UNIVERSITY OF TECHNOLOGY SYDNEY
Centre for Autonomous Systems
Faculty of Engineering and Information Technology

Research on 2D general feature based SLAM algorithm for mobile robot

by

Jiaheng Zhao

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

Doctor of Philosophy

Sydney, Australia

2021

Certificate of Authorship/Originality

I, Jiaheng Zhao certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for the award of Doctor of Philosophy, in the Faculty of Engineering and Information Technology 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.

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of the requirements for a degree at any other academic institution except as fully acknowledged within the text. This thesis is the result of a Collaborative Doctoral Research Degree program with Beijing Institute of Technology.

Signed: Production Note: Signature removed prior to publication.

Date: 21/07/2021

ABSTRACT

Research on 2D general feature based SLAM algorithm for mobile robot

by

Jiaheng Zhao

Simultaneous Localization and Mapping (SLAM) is a fundamental research problem for autonomous robot navigation and map construction. This thesis studied the problem of improving the performance of localization and mapping for mobile robots, including pre-fitting features with ellipse representation, representing features with implicit functions, parameterization in Fourier series, and submap joining.

The conventional planar scan matching approach cannot cope well with the open environment as lacking of sufficient edges and corners. A SLAM algorithm with pre-fitted conic features via 2D lidar is presented, which is named as Pre-fit SLAM and can be adapted to an open environment nicely. The novelty of this work includes threefold: (1) defining a conic feature based parameterization approach; (2) developing a SLAM method to utilize feature's conic geometric information and odometry information since open environments are short of regular linear geometric features. Synthetic and practical experiments demonstrated that the proposed SLAM algorithm can get accurate and convincing results for the open environment and the map in our representation can express accurately the environment situation.

In order to avoid information loss during pre-fitting progress and to enlarge the scope of feature representation, a post-count framework for 2D lidar SLAM with implicit functions on general features is studied. Since 2D laser data reflect the distances from the robot to the boundary of objects in the environment, it is natural to use the boundary of the general objects/features within the 2D environment to describe features. Implicit functions can be used to represent almost arbitrary shapes from simple (e.g. circle, ellipse, line) to complex (e.g. a cross-section of a bunny

model), thus it is worth studying implicit-expressed feature in 2D laser SLAM. The main contributions are the specific problem formulation and algorithm framework for 2D laser SLAM with general features represented by implicit functions (named as Implicit-SLAM). Furthermore, ellipses and lines are used as examples to compare the proposed SLAM method with the traditional pre-fit method. Simulation and experimental results show that Implicit-SLAM has a better performance compared with Pre-fit SLAM and other methods, demonstrating the potential of this new SLAM formulation and method.

A 2D laser SLAM approach with Fourier series based feature parameterization (called Fourier-SLAM) and submap joining is studied to improve the efficiency of convergence and the accuracy of method using implicit functions. The Fourier series are introduced to parameterize irregular closed shape features and the SLAM problem with Fourier series feature parameterization is formulated. A submap joining process is also derived in order to reduce the high dependence on precise initial guess and the computing time. Fourier-SLAM has been evaluated on both synthetic and actual data and is able to obtain accurate trajectory and feature boundaries. We also prove that submap joining method can improve the calculation efficiency without loosing too much accuracy.

Acknowledgements

Throughout the writing of this dissertation I have received a great deal of support and assistance.

I would first like to express my sincere gratitude to A/Prof Shoudong Huang for providing me with an opportunity to do a PhD at UTS. I went from being a rookie who didn't know how to do scientific research at all to graduating with a PhD. He was the most helpful to me in this process. There have been countless times when I constantly doubted my abilities, it is him who is always there, giving me courage and encouragement. I remember he once said that the most important thing between supervisors and students is mutual trust. Whether it is research or life, we all trust each other, and this trust has also helped me overcome many difficulties. Thanks for the assurance when I was feeling doubt about myself. Thanks for the empowerment when I was in great fear. Finally, thanks for the comfort when I was feeling vulnerable.

I would also like to extend my sincere thanks to my co-supervisor Dr. Liang Zhao for invaluable supervision and support during my PhD degree. He is not only a teacher, but more like my brother. He always shared with me knowledge and experience in research methodologies and skills, and gave me many helpful feedback and suggestions on my projects and papers. He set an example for me on how to be a good researcher.

I would like to thank all the excellent researchers I have met during my PhD study. Enormous thanks to Miao Zhang, Yanhao Zhang, Tianming Wang, Huan Yu for their company in my research and life. I would like to thank Yongbo Chen, Jun Wang, Jingwei Song, Fang Bai, Felix H Kong, Tiancheng Li, Shuai Zhang, Zhehua Mao, Yu He, Liyang Liu, Brenton Leighton, Andrew To for many inspiring discussions on research. I would like to thank Tong Yang, Yue Wang at Zhejiang

University for the discussion and advice on my research. I would like to thank Beijing Institute of Technology for providing me with the opportunity to study at University of Technology Sydney.

I would like to thank my better half Yi Yang. Thank you for your comfort and encouragement during my many emotional breakdowns. Although I thank you constantly, I'll never be able to say it enough.

Finally, I would like to thank my parents. It is the meticulous care and encouragement of my parents that support me to move forward in difficult times. From the bottom from my heart, thanks for all the sacrifices, thanks for all the supports and thanks for all the assurance.

Jiaheng Zhao

Sydney, Australia, 2021.

List of Publications

1. **Zhao, Jiaheng**, Tiancheng Li, Tong Yang, Liang Zhao, and Shoudong Huang. "2D Laser SLAM With Closed Shape Features: Fourier Series Parameterization and Submap Joining." *IEEE Robotics and Automation Letters* 6, no. 2 (2021): 1527-1534.
2. **Zhao, Jiaheng**, Liang Zhao, Shoudong Huang, and Yue Wang. "2D laser SLAM with general features represented by implicit functions." *IEEE Robotics and Automation Letters* 5, no. 3 (2020): 4329-4336.
3. **Zhao, Jiaheng**, Shoudong Huang, Liang Zhao, Yongbo Chen, and Xiao Luo. "Conic feature based simultaneous localization and mapping in open environment via 2D lidar." *IEEE Access* 7 (2019): 173703-173718.
4. **Zhao, Jiaheng**, Shoudong Huang, and Liang Zhao. "Constrained Gaussian mixture models based scan matching method." In *Australasian Conference on Robotics and Automation, ACRA*. 2018.
5. Kong, Felix H., **Jiaheng Zhao**, Liang Zhao, and Shoudong Huang. "Analysis of Minima for Geodesic and Chordal Cost for a Minimal 2-D Pose-Graph SLAM Problem." *IEEE Robotics and Automation Letters* 5, no. 2 (2019): 323-330.
6. Jia, Yan, Xiao Luo, Baoling Han, Guanhao Liang, **Jiaheng Zhao**, and Yuting Zhao. "Stability criterion for dynamic gaits of quadruped robot." *Applied Sciences* 8, no. 12 (2018): 2381.

Contents

List of Figures

Abbreviation

SLAM - Simultaneous Localization and Mapping

RMSE - Root Mean Square Error

2D: Two-dimensional

3D: Three-dimensional

ICP - Iterated Closest Point

NDT - Normal Distribution Transform

GMM - Gaussian Mixture Models

ToF - Time-of-Flight

Nomenclature and Notation

The semicolon is to represent vertical vector concatenation. Lowercase letters indicate scalars, bold lowercase letters indicate vectors, and uppercase letters indicate matrices. Some special symbols are listed below.

The observed points have zero-mean Gaussian noise $\mathbf{n}_z \in \mathbb{R}^2 \sim N(\mathbf{0}, \Sigma_z)$.

${}^{\{j\}}\mathbf{p} \in \mathbb{R}^2$ denotes an observed point in the frame j.

${}^{\{j\}}\mathbf{P}_k = [{}^{\{j\}}\mathbf{p}_1^k, \cdots, {}^{\{j\}}\mathbf{p}_M^k]$ is a 2D point set observed of feature k at the frame j. ${}^{\{G\}}\mathbf{P}_k$ is usually abbreviated as $\mathbf{P}_k$ since it is relative to the global frame.

ϕ is an angle within the range $[-\pi, \pi)$.

$R(\phi) \in SO(2) = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}$ is the corresponding rotation matrix which is abbreviated as R.

$\mathbf{t} = [t_x, t_y]^\intercal$ is the translation vector.

$R_{ij}, \mathbf{t}_{ij}$ means the rotation and translation from frame i to frame j.

If i is the global frame $\{G\}$, it is usually omitted in order to simplify the formula.

$\mathbf{\Xi}_j = [\mathbf{t}_j; \phi_j]$ is a robot pose.

$T(\mathbf{\Xi}_j, {}^{\{j\}}\chi)$ represents the process of transforming a point/point cloud/feature from frame $\{j\}$ to global frame $\{G\}$.

$T^{-1}(\mathbf{\Xi}_j, {}^{\{G\}}\chi)$ represents the process of transforming a point/point cloud/feature from global frame $\{G\}$ to frame $\{j\}$.

Feature $\mathbf{\Phi}_k$ is in closed shape, whose boundary point set is denoted by $\mathbf{P}_k$.

$\|\mathbf{e}\|_{\Sigma}^2$ is the weighted L2 vector norm with a covariance of Σ.

Chapter 1

Introduction

1.1 Background

Robotics technology can complete specific tasks in different operating environments and has been applied in various fields such as industry, medical, military, deep space, and underwater (as shown in Fig. 1.1). As the most widely used category in the field of robotics, intelligent mobile robots have flexible motion capabilities and can complete autonomous navigation in the environment. Therefore, mobile robotics have been a scenic spot for researchers and achieve good performance in industry, agriculture, medical care, and service industry (Rubio et al., 2019).

(a) Industrial robot

(b) Service robot

(c) Surgical robot

(d) Sweeping robot

(e) Logistics robot

(f) Grazing robot

Figure 1.1 : The application of mobile robots in different scenarios.

How to judge whether a mobile robot is "intelligent"? One of the core criteria is the robot's performance in an unknown environment. In an unknown environment, intelligent mobile robots need to be able to complete the following three tasks:

1. Mapping. Mobile robots should have the ability to build a map of the unknown environment.

2. Localization. Mobile robot should be aware of its position and posture in the current environment.

3. Navigation. Mobile robot should be able to autonomously plan a route to a given target location.

The combination of tasks 1 and 2 is called Simultaneous Localization and Mapping (SLAM), which is the computational problem of constructing or updating a map of an unknown environment while simultaneously keeping track of a robot's location within it (Cadena et al., 2016; Folkesson and Christensen, 2007; Cieslewski and Scaramuzza, 2017; Durrant-Whyte and Bailey, 2006; Bailey and Durrant-Whyte, 2006).

Nowadays, mobile robots have been able to utilize given absolute positions to navigate in a known environment. The most typical application is to employ navigation satellites in the Global Positioning System (GPS) with given road map. However, mobile robots cannot determine their position and orientation in unknown environments due to limitations of GPS technique:

- Unable to obtain surrounding environmental information. GPS can only provide a globally unified three-dimensional geocentric coordinate, but cannot construct the environment around the sensor.

- The civilian localizing accuracy only reaches the meter level. Although on a global scale, the meter-level localizing accuracy can basically meet the de-

mand, but in a large number of small-scale scenarios, the meter-level localizing accuracy is too rough to guarantee the robot's working accuracy at all.

- GPS cannot guarantee 100% positioning accuracy, and indoor positioning is even more inaccurate. GPS positioning will be affected by many aspects, including buildings, viaducts, radio waves, and so on. Generally speaking, the more open the GPS is, the more accurate the positioning is. The actual environment cannot guarantee that the surrounding environment is open enough. The inability to accurately locate indoors directly limits the application of GPS to mobile robots.

Therefore, the study of SLAM problems is necessary.

As series of researches indicated that different sensors equipped on a robot have significant influences on its potential application, it is worth mentioning that SLAM system based on lidar is considered as an effective and accurate way for robots to construct a map and locate themselves (Yin et al., 2014; Song et al., 2018, 2017; Pedrosa et al., 2017).The robot platform used for experiments in this thesis is the "Fetch robot", and part of the experiment is done on the Fetch mobile platform and the corresponding simulator. Fetch robot is a mobile platform developed by Fetch Robotics in the United States for academic research (Wise et al., 2016). The platform is favored by many scientific researchers due to its robust design, superior performance, and relatively low cost (as shown in Fig. 1.2).

1.2 Motivation

In the last few years, one application that has been widely adopted by industry and academy is planar SLAM based on 2D lidar or laser rangefinder, and the number of approaches has increased (Ren et al., 2019; Wilson et al., 2019; Badue et al., 2019; Hess et al., 2016; Labbé and Michaud, 2019).

Figure 1.2 : Fetch robot.

Currently, the two main common approaches to 2D laser SLAM are scan matching based approach and feature based approach. In a typical scan matching based approach, nearby scans are registered to obtain the relative poses, and then a pose-graph optimization is performed to obtain the optimized poses. Finally, the map is built via the optimized poses and the laser scans. Two classes of scan points registration methods are adopted in recent years' research, namely, Iterated Closest Point (ICP) based methods (Besl and McKay, 1992; Sharp et al., 2002; Yang et al., 2013) and Gaussian Mixture Models (GMM) based methods (Jian and Vemuri, 2011).

The latter one can be extended to a famous special case named Normal Distribution Transform (NDT) (Biber and Straßer, 2003; Stoyanov et al., 2012; Zhao et al., 2018). On this basis, some state-of-the-art 2D lidar-based SLAM algorithms have been developed for many indoor scenarios (Hess et al., 2016; Olson, 2015; Konolige et al., 2010; Martín et al., 2014; Li et al., 2018), especially those constituted by regular, obvious and sufficient lines or corners.

Although it is beneficial for scan matching not making assumption on environment, a prior knowledge of the geometrical information is helpful to improve the accuracy. In the industrial environment with multiple stacks, for instance, the boundary description of manufactured objects can be easily obtained from the manufacturer, which is workable to model the boundary via implicit functions for stacks. Furthermore, there is difficulty involved for scan matching method in accurately fusing information from consecutive scans.

Thus, in order to overcome problems of incorrect scan matching caused by environmental influences, some researchers seek help from features. Feature based SLAM problem focuses on parameterizing objects in the environment and estimating robot locations and orientations as well as feature parameters. In the applications of 2D laser rangefinder, various features are parameterized and used in SLAM problems such as point features, circular features, and polynomial features. For an object in the environment, researchers generally intend to either estimate a representative center of a feature or approximate the boundaries. One common sense of the former way is that the estimation of features should not be affected by their own geometrical structures, such as landmarks or similar stacks. In (Guivant and Nebot, 2002), the centers of tree trunks are regarded as point features to be estimated. The latter way takes the shape of features into account and tries to represent features via selected functions. For instance, line function (Kim and Oh, 2008) and curve function (Liu et al., 2011) have been studied as feature parameterization these years. However,

it is not easy to seek out an appropriate way to parameterize features since simple structural objects (circle, ellipse, rectangle, etc.) are rare and feature boundaries are not always in regular shapes. In order to address such problems, some researchers seek help from polynomial functions. Gee et al. (Gee et al., 2016) utilized a cubic polynomial function to fit sparse laser points to acquire dense observation, but they do not estimate features with the polynomial function. Instead, the polynomial function is used to interpolate consecutive sparse laser observations and generate a dense model. However, they still focus on fitted points without considering the features in the environment.

Based on the content mentioned above, three research questions are raised:

1. When the scan matching method is prone to failure, how to use features to obtain correct poses?

2. How to ensure the generality of feature representation under the premise of being as accurate as possible?

3. How to improve the accuracy of the results while meeting the needs?

Inspired by these three questions, we started with special cases and studied feature-based SLAM problems in a targeted manner.

1.3 Contributions

With the aforementioned research questions, the main contributions of this thesis consist of four aspects:

- A conic feature based SLAM algorithm called Pre-fit SLAM was proposed. The method is aimed in open environment using 2D laser sensor. Tradition scan matching methods are not competent for working in an open environment

where sufficient edges and corners do not exist. We proposed a conic feature based method to represent features instead of matching scan points. First, the raw data was fitted to the feature parametrization proposed in this thesis. Then a factor graph optimization was adopted to obtain pose estimates as well as the map in our representation. Simulation and practical environments demonstrated that the proposed algorithm can get accurate and convincing results for the open environment and the map in our representation can accurately express the environment situation.

- A clear problem formulation and a solution framework for implicit function based SLAM problem (named as Implicit-SLAM) were proposed. Two challenges involved in this novel SLAM problem are addressed. One is finding the covariance of the noises involved in implicit energy terms. Another is handling the asymmetry of the energy terms for closed shape features. Simulation results using ellipse and line features as examples shows that the proposed method is more robust to observation noises and outperforms the pre-fit methods. It is also shown that using hybrid features can achieve better accuracy in SLAM compared with SLAM with only ellipses or lines. Practical experiment illustrates that this method has the ability to acquire accurate result.

- A 2D feature based SLAM approach utilizing Fourier series as feature parameterization (named as Fourier-SLAM) was proposed. Compared to implicit function based method, simulated experiments concluded that this method does not rely significantly on initial guess and can provide close-to-real feature boundaries. Practical experiment shows that Fourier-SLAM surpass Cartographer under certain scenarios and has the ability to be applied to the general environment.

- A submap joining method with the Fourier series parameterization was for-

mulated. Submap joining method is able to speed up the calculation without leading to an unacceptable result.

1.4 Thesis Organization

This thesis is organized as follows:

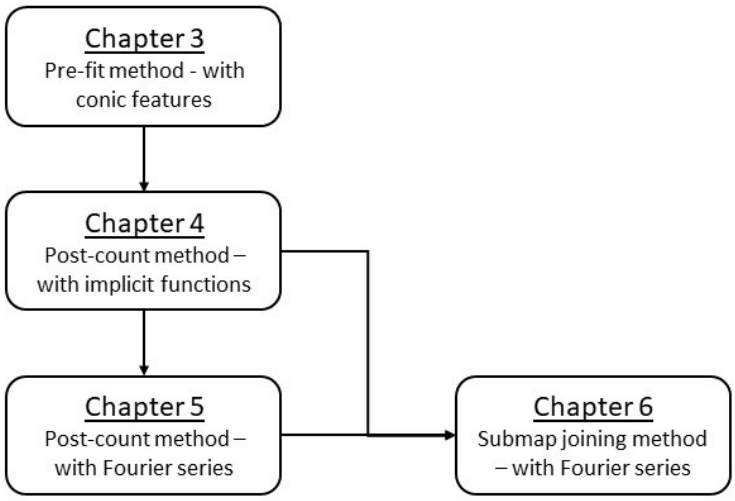


Figure 1.3 : Thesis organization.

- *Chapter 2*: This chapter presents a survey of feature based SLAM on mobile robot's application.
- *Chapter 3*: A pre-fit conic feature based SLAM algorithm called Pre-fit SLAM is derived in this chapter.
- *Chapter 4*: This chapter presents a post-count feature based SLAM framework utilizing implicit function, which is named as Implicit-SLAM.
- *Chapter 5*: A post-count feature based SLAM framework with Fourier series (named as Fourier-SLAM) is introduced in this chapter.

- *Chapter 6*: This chapter presents a submap joining method for the post-count feature based method proposed in this thesis.

- *Chapter 7*: A brief summary of the thesis contents and its contributions are given in the final chapter. Recommendation for future works is given as well.

The relationship among the main chapters is illustrated in Fig. 1.3. This thesis starts from researching on pre-fitting features with a certain shape (Chapter 3). Chapter 4 is an extension of Chapter 3, which reduces the error impact caused by the loss of information during the pre-fitting. On the basis of Chapter 4, Chapter 5 focuses on solving the problem of being too sensitive to the initial value, and can express more complex features. Chapter 6 is mainly to improve the calculation speed of the post-counting method.

Chapter 2

Literature Survey

As mentioned earlier, SLAM is a process where a robot locates itself in an unknown environment and builds a map of the surrounding environment. In the past few decades, SLAM technology has received a lot of attention and research (Grisetti et al., 2007; Kaess et al., 2008; Olson et al., 2006). In the application of mobile robot, multiple algorithms or techniques have been proposed and implemented.

2.1 General SLAM methods for Mobile Robot

According to the different types of sensors equipped in the robots, SLAM methods can be roughly divided into vision based SLAM (Qin et al., 2017; Klein and Murray, 2007; Forster et al., 2016) and lidar based SLAM (Bry et al., 2015, 2012).

Vision based SLAM focuses on solutions to different sensors, such as Monocular camera (Davison, 2003), Binocular camera (Mahon et al., 2008), ToF camera (Kim and Oh, 2008) and RGB-D camera (Shi et al., 1994). Representative methods include RGB-D SLAM (Endres et al., 2013) and ORB-SLAM (Mur-Artal et al., 2015; Mur-Artal and Tardós, 2017), which extract and match feature points from a series of continuous images to obtain robot poses.

Davison et al. (Davison, 2003) proposed a pioneering visual SLAM solution. They used a monocular camera to collect environmental information, then use the Shi And Tomasi operator (Shi et al., 1994) to extract sparse features in the environment, and utilized the normalized sum of squares to match the new features with the observed features for data union and map construction. Klein and Murray first

proposed Parallel Tracking and Mapping (PTAM) in 2007 (Klein and Murray, 2007). In recent years, with the development of GPU, the attention of some researchers has gradually shifted from the sparse 2D or 3D SLAM problem to the dense 3D reconstruction. Newcombe and Davison (Newcombe and Davison, 2010) successfully used a monocular camera to obtain a dense 3D environment model in real time. Henry et al. (Henry et al., 2012) took the lead in using an RGB-D camera (using a Microsoft Kinect sensor in their work) to implement an RGB-D-based mapping method. They used RGB images and depth images to reconstruct the dense three-dimensional environment, and at the same time estimated the pose of the camera with 6 degrees of freedom. They extracted SIFT (Scale-Invariant Feature Transform) features in each frame first, and then used Calonder descriptor (Besl and McKay, 1992) to match the two consecutive frames. Finally RANSAC is used to eliminate abnormal matches.

Bachrach et al. (Bachrach et al., 2012) proposed an unmanned aerial vehicle SLAM system using RGB-D cameras. The system relies on extracting FAST features from successive pre-processed images of different pyramid levels, and then restricts the size of the sliding window for feature matching for initial rotation estimation. The matching is performed by squaring the descriptor vectors and finding the pair with the lowest sum of each other. The greedy algorithm is also applied to refine the matching and obtain the corresponding pairs, and then use the result to estimate the relative relationship between consecutive frames movement. In order to reduce drift in motion estimation, they recommend matching the current frame with selected key frames instead of matching consecutive frames.

Except for vision based SLAM algorithm, another trend is lidar based SLAM approaches. Lidar based SLAM is considered to be an effective and accurate method for robots to build maps and locate themselves (Yin et al., 2014; Song et al., 2018, 2017; Pedrosa et al., 2017). The lidar odometry methods or SLAM algorithms which implement lidar as the research object have been improved in recent years. The

LOAM proposed by Zhang et al. (Zhang and Singh, 2014) is a well-known framework in lidar visual odometry. They used a laser rangefinder with an additional motor controlling the laser sensor for 360 degree rotation. Instead of processing every single point, they extracted edge feature points and surface feature points for scan matching. Jiang et al. (Jiang et al., 2019b) proposed IMLS-SLAM which used the scan-model matching form to estimate the relative pose transformation. First, the 3D scan needs to be sampled, and then the implicit least squares method is used for surface reconstruction to improve the quality of matching. Graeter et al. (Sehgal et al., 2019) proposed an odometry estimation framework that combines lidar and monocular cameras. They calculated features in the image first and estimated the depth through lidar data corresponding to these features, and then used bundle adjustment to fuse sensors and estimate the pose.

Tang et al. (Tang et al., 2019) proposed a method called simultaneous trajectory estimation and mapping (STEAM). They used the ground truth to train the Gaussian process. The input of the system is the well-detected features extracted from the point cloud, and the output of the system is the predicted pose calculated using the estimator and ground truth. At a deeper level, this algorithm starts with down-sampling point clouds, and uses normalized intensity values to transfer the large amount of data in the point cloud to sparse points (which are also called key points). If a point meets certain conditions of the proposed algorithm, it can be selected as the key point. Then these sparse point clouds are matched according to the minimized Euclidean distance. To estimate the trajectory, they adopted the STEAM framework, in which the continuous-time trajectory was estimated by Gaussian process regression. Ji et al. (Ji et al., 2018) proposed a closest probability and feature grid based SLAM algorithm that enables unmanned vehicles to locate in off-highway scenes. Behley and Stachniss (Behley and Stachniss, 2018) proposed a localizing method using surfel map, which can estimate the change of robot pose

through the data association between the current scan of and the model view of the surface map.

In addition to exploit diverse sensors, researchers also explored different mathematical thinking to solve SLAM problems. The two representative ideas are either based on filtering or based on optimization. Zhang et al. (Zhang et al., 2017a) compared extended Kalman filter (EKF-SLAM), unscented Kalman filter (UKF-SLAM) and unscented FastSLAM (UFastSLAM) and concluded the priority of UFastSLAM. Murphy and Russell (Murphy and Russell, 2001) proposed the first solution to SLAM applying Rao-Blackwellized particle filter method. Grisetti et al. (Grisettiyz et al., 2005) developed an open sourced SLAM algorithm called Gmapping utilizing Rao-Blackwellized particle filter and the improved proposal distribution. Gmapping has good performance in less-feature environment (e.g. corridor) and small scaled scenes, requiring less calculation without loosing accuracy. Steux and El Hamzaoui (Steux and El Hamzaoui, 2010) developed a 200-lines-of-code method, which estimates robot pose with particle filter. Kohlbrecher et al. (Kohlbrecher et al., 2011b) proposed Hector SLAM, which is an algorithm fusing Inertial Measurement Unit (IMU) and 2D lidar by EKF to acquire 2D pose.

2.2 Feature parameterization for 2D laser SLAM

The general vision-based SLAM methods tend to extract feature points of images to perform feature-based SLAM, whether it is using filter-based approaches or optimization-based approaches. On the other hand, lidar-based SLAM methods are mainly divided into two categories, one is the method using scan-matching, and the other is the method using features.

2.2.1 Point and line feature parameterization

A general way in feature based SLAM problem is to consider lidar data as point feature or line feature via series of segmentation (Guo et al., 2019; Wang et al., 2019a; Gargoum and El Basyouny, 2019). Holỳ (Holỳ, 2016) combined points and lines for the scan matching to decrease the computation time and increase the accuracy. Nguyen et al. (Nguyen et al., 2005) compared six line feature extraction methods on 2D laser scans for indoor environment. Wu et al. (Wu et al., 2018) put forward a point and line features based 2D laser SLAM applied in the underground tunnel environment. They extracted point features via curvature changes and extracted line features by Principal Component Analysis (PCA) algorithm. The feature state is combined by point features and the end points of line features. Chen and Peng (Chen and Peng, 2019) presented a corridor feature detector by evaluating the main direction of the 2D lidar scan via PCA. With the corridor indicator η they then adjusted the scan matching threshold for a better performance. Walter et al. (Walter et al., 2007) proposed an exactly sparse extended information filter to solve feature-based SLAM problem. de la Puente and Rodríguez-Losada (de la Puente and Rodríguez-Losada, 2014) introduced a prior knowledge and presented a way to discover new structures by hierarchically including different kinds and levels of features.

Zhang and Ghosh (Zhang and Ghosh, 2000) employed 2D laser rangefinder to locate the robot and built a corresponding map via extracting line segments as basic elements. Coincidentally, Li et al. (Li et al., 2016) proposed a point and line features based SLAM method. They firstly distinguished point and line features via a Split-and-Merge algorithm, then optimized poses by minimizing l_q-norm distance (Marjanovic and Solo, 2012, 2014).

2.2.2 Complex geometric feature parameterization

However, line features are not suitable for open environments. Some recent studies have considered the use of conic curves to calculate the location of features (Bailey et al., 2006). Zhang et al. (Zhang et al., 2019) combined 2D lidar and gyroscope to navigate a robot in the forest. They utilized circles to fit scan points and estimated poses by feature-based extended Kalman filter. Different from point-feature based methods, their research focused on the initial application assuming all features as circles. Hu et al. (Hu et al., 2019) analyzed the reflective characters of 2D points hit on artificial landmarks and parameterized features with the intensity and the number of hit points. Their feature parameterization method is highly relied on reflectors and is not adapted to a general environment.

For an object in the environment, some researchers generally intended to either estimate a representative center of a feature or approximate the boundaries. One common sense of the former way is that the estimation of features should not be affected by their own geometrical structures, such as landmarks or similar stacks. In (Guivant and Nebot, 2002), the centers of tree trunks are regarded as point features to be estimated. The latter way takes the shape of features into account and tries to represent features via selected functions. For instance, line function (Kim and Oh, 2008), ellipse function(Zhao et al., 2019) and curve function (Liu et al., 2011) have been studied as feature parameterization these years. However, it is not easy to seek out an appropriate way to parameterize features since simple structural objects (circle, ellipse, rectangle, etc.) are rare and feature boundaries are not always in regular shapes. In order to address such problems, some researchers seek help from polynomial functions. Gee et al. (Gee et al., 2016) utilized a cubic polynomial function to fit sparse laser points to acquire dense observation, but they do not estimate features with the polynomial function. Instead, the polynomial function is used to interpolate consecutive sparse laser observations and generate

a dense model. A unified formulation named *matchable* was employed in (Aloise et al., 2019) to represent point, line and plain features. Zhang et al. (Zhang et al., 2017b) utilized remote and near feature parametrization to improve the robustness of rotation estimation. Rao et al. (Rao et al., 2012) extracted Bézier curves and used four control points to parameterize curve features, then the optimization problem was solved by Levenberg-Marquardt algorithm. Pedraza et al. (Pedraza et al., 2009) were the first to use spline to parameterize features and then optimized robot poses and control points simultaneously.

Another potential option of fitting features in this thesis is Fourier series. The process of fitting shape with Fourier series is a typical application on shape retrieval in image processing (Zhang and Lu, 2001; Puhan et al., 2011). Shapes can be sampled first and fitted through Fast Fourier Transform. Rakshit et al. (Rakshit and Monro, 2007) used Fourier series to extract non-circular human pupil iris boundaries. Su and Xiang (Su and Xiang, 2020) proposed a Fourier series based approach to characterize 2D general-shape particles. Jiang et al. (Jiang et al., 2019a) implemented Fourier Transform to assist scan matching.

2.3 Submap joining in SLAM

In the SLAM research, submap joining method is one subject which focuses on reducing the complexity of calculation and increasing the approximation speed. Huang et al. (Huang et al., 2008a) proposed an Extended Information Filter based sparse local map submap joining approach, then they extended the application using multiple iterations to improve the consistency (Huang et al., 2008b). Chen et al. (Chen et al., 2018) applied submap joining to improve the real-time ability of their algorithm. Ni et al. (Ni et al., 2007) proposed a divide-and-conquer scheme to solve the optimizing problem. Zhan et al. (Zhan et al., 2020) presented a method to construct a global map via corresponding undirected connected graph. Nevertheless,

all of them focused on point features.

Wang et al. (Wang et al., 2019b) combined points, patches and planes in local map building and proposed a submap joining method using both point and plane features. Burguera Burguera and Bonin-Font (Burguera Burguera and Bonin-Font, 2019) optimized the trajectory with the help of map joining by adding a link between the joined map nodes. Zhao et al. (Zhao et al., 2010) combined local maps to reduce the computational cost during large-scale map building. These works focused on vision SLAM and cannot represent closed shape features.

Sun et al. (Sun et al., 2020) optimized series submaps acquired during exploration. But their work only focuses on frontier detection. Aulinas et al. (Aulinas et al., 2010a) used local maps to correct errors in the global map built in the underwater scenarios. They also used the submap joining method to reduce the computational cost for the Victoria dataset (Aulinas et al., 2010b). Ahmad et al. (Ahmad et al., 2012) combined local maps via least squares optimization to maintain the estimation consistency in a range-only SLAM problem.

2.4 Summary

It can be found from the literature survey that there have been rich researches on 2D laser SLAM. However, researches on feature based SLAM are insufficient, most of which prefer to use point features or line features represented by end points instead of intuitive geometric information in the environment.

We aim to maximize the use of environment information to solve feature based SLAM problem. In this thesis, we investigate the possibility of directly observing the geometric shape of features in Chapter 3. Furthermore, we study methods of preserving original information to the greatest extent in Chapter 4 and Chapter 5. Finally, we explore the method to improve computational efficiency in Chapter 6.

Chapter 3

A pre-fit feature based SLAM method in open environment: Pre-fit SLAM

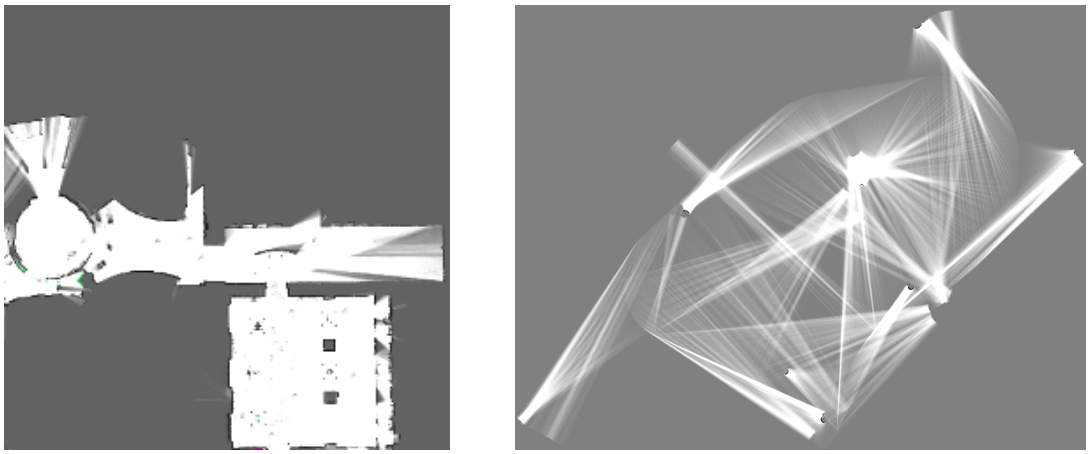
Open outdoor environments are common application scenarios for mobile robots. One property of open outdoor environment is that there are less buildings and typically exist sparse trees, which makes scan-matching approaches easy to fail. Generally, academics always treat such sparse trees as point features and pre-fit features once receiving observation data. In this chapter, we focus on a pre-fit feature based SLAM method and propose a novel conic feature based SLAM algorithm for mobile robot working in open environment. In contrast with existing planar SLAM systems or algorithms, the method in this chapter takes advantage of the conical properties of points scattered on the contour of the objects, while conventional methods only consider relationship between points or extraction of line segments. More than just utilizing geometric information, a further progress has been made that we defined a new parameterization approach for such conic feature and constructed corresponding factor graph optimization model. We also represented the map with conic features instead of occupancy grid map. In order to verify the feasibility of the proposed approach, we carried out experiments in simulated and real environments respectively. Also we evaluated Pre-fit SLAM on a public dataset. The main contributions of this chapter include:

1. Defining a conic feature based parameterization approach.

2. Developing a method called “Pre-fit SLAM” to utilize feature’s conic geometric information and odometry information since open environments are short of

regular linear geometric features.

The remainder of this chapter is organized as follows: Section 3.1 analyzes the difficulty of the problem and briefly describes the algorithm flow. Specific methodological theory and implementation are suggested in Section 3.2. Section 3.3 presents detailed comparative experiments and analysis. Finally, conclusions and summaries are drawn in Section 3.4.



(a) Indoor. (b) Outdoor.

Figure 3.1 : Two typical environments. (a) Indoor environment consisting of sufficient lines and corners; (b) Outdoor environment lacking of sufficient lines and corners.

3.1 Problem description and algorithm structure

In recent years, researchers have developed a series of advanced technologies based on 2D lidar for many indoor scenes, especially environments rich in regular, obvious and sufficient features such as lines or corners (Hess et al., 2016; Olson, 2015; Konolige et al., 2010; Martín et al., 2014; Li et al., 2018), as is shown in Fig. 3.1a.

Compared with the indoor environment, the outdoor environment is more complicated and cannot be treated as a general scenario (as shown in Fig. 3.1b). An open environment presents a series of challenges including:

1. Lacking of regular linear geometric information. Different from indoor environment, an open area is often composed of scarce trees, which has a significant influence on the performance of scan matching. Assume the robot moves around an object with cylindrical surface, then data acquired by lidar equipped on the robot is distributed on the robot-facing edge. In cases where the robot is operated in an indoor environment with boundaries and polygon features, such edge-distributed data will not increase the error apparently because its weight is diluted by other lines or corners. Once the scenario is lack of polygon features or boundaries, the performance of matching is dramatically declined.

2. Inconsecutive observations. Due to the sparse objects within open environments, observations can not be obtained steadily over time. The discontinuous observation makes frame-to-frame scan matching easily fail since the transformation between two valid adjacent scans could be enormous, which leads to unpredictable estimated relative poses. The wrongly estimated poses also cause accumulated error when registering several frames into a map, which also make frame-to-map matching inaccurately. Therefore, scan matching cannot be executed in order to make sure the robustness.

3. Large scale maps. One common method to build map is via occupancy grid map (OGM). However, constructing OGM in an open environment is ineffective since the large proportion of the environment consists of free space. Eventually, building OGM still causes a waste of computational memory because much data are stored as “free grid”.

In response to the above challenges, this chapter proposes a SLAM method based

on the parameterization of conic features. As shown in Fig. 3.2, the architecture of this method consists of two main parts, data processing and back-end optimization. After data collection is completed by lidar, the first stage called data processing begins, and the point set is fitted by a conic equation. The scan point is under the suggested parameterization after fitting. Note that although the points have been fitted to the cone feature in this step, the original points are still stored. Then, using the prior of the odometer information, if the current feature has never appeared before, the feature list will be updated, otherwise, the new edge between the current step and the feature will be linked to help close the loop. After data processing, the problem continues to be optimized by iterative nonlinear least squares method (such as Gauss-Newton method or Levenberg-Marquardt algorithm). This stage is called graph optimization or backend. Finally, the pose, feature parameters and final graph structure of the robot are obtained after the back end.

3.2 Data pre-procession for open outdoor environment

Conic feature parameterization and association are finished at this section. We proposed a conic feature parameterization to model conic features for solving SLAM problem. The conic feature can be fitted on the basis of Ahn's work (Ahn et al., 1999). We also studied the uncertainty flow from sensor to parameterized feature which makes the fitted result reliable.

3.2.1 Feature parameterization

There are four basic types of conics: circles, ellipses, hyperbolas, and parabolas. Fortunately, it is unnecessary to utilize all of the four types. In real world, circular or elliptical shaped objects such as trees and pillars appear more frequently, which means it is easy to implement circle or ellipse equation when denoting actual features. Further more, circle is the special case of ellipse where the major axis and the minor

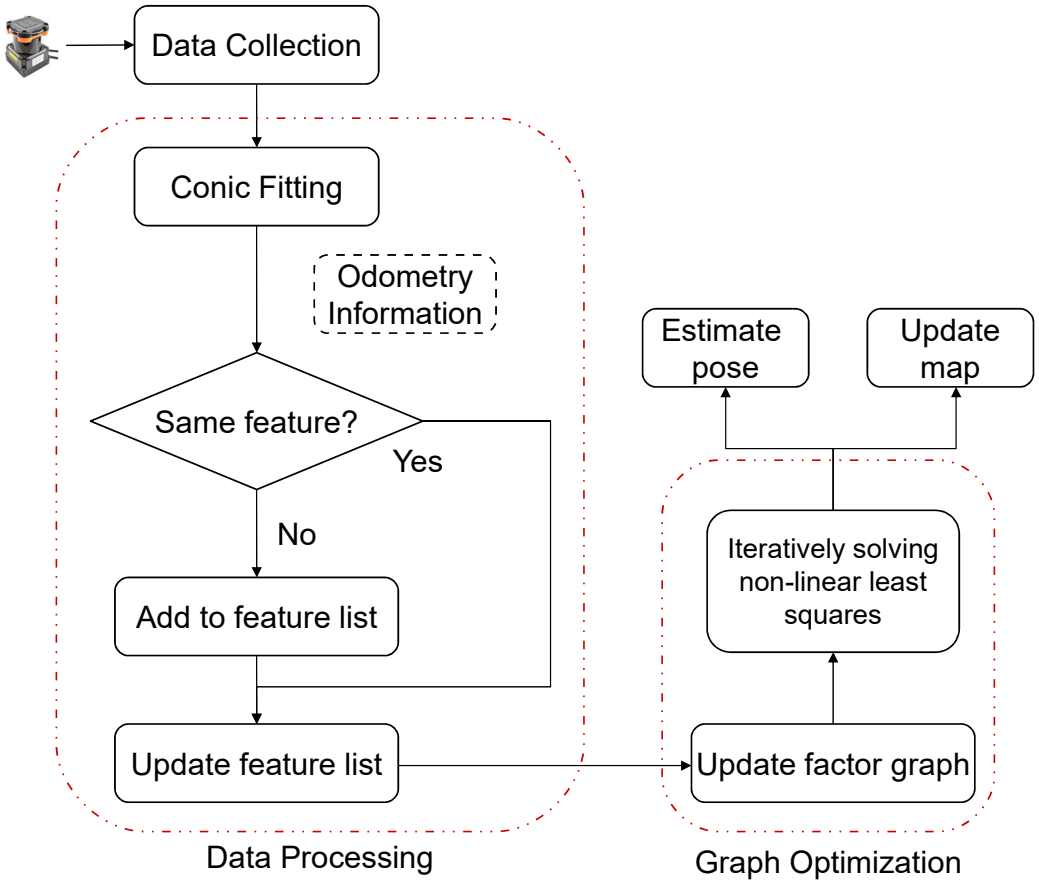


Figure 3.2 : The flowchart of Pre-fit SLAM

axis have the same dimension. Hence we can take advantage of ellipse equation to express features.

Motivated by the above insight, the conic feature suggested in this chapter is parameterized by

$$\mathbf{\Phi} = [{}^{\{G\}}F_x, {}^{\{G\}}F_y, F_\phi, F_{r_1}, F_{r_2}]^\intercal \tag{3.1}$$

As is shown in Fig. 3.3, $[{}^{\{G\}}F_x, {}^{\{G\}}F_y]^\intercal$ is the central coordinate of the ellipse in the global frame, F_ϕ is the angle between ellipse's major axis and the world frame x-axis. $[F_{r_1}, F_{r_2}]^\intercal$ are the absolute dimensions of the major axis and the minor axis respectively.

If the feature has circular shape, it is apparently confused to decide the spe-

cific position of the major or minor axis as well as the angle. Fortunately, we can still adopt this expression because of the elaboration of feasibility and validity of proposed feature parameterization in Section 3.2.3 and Section 3.4.2.

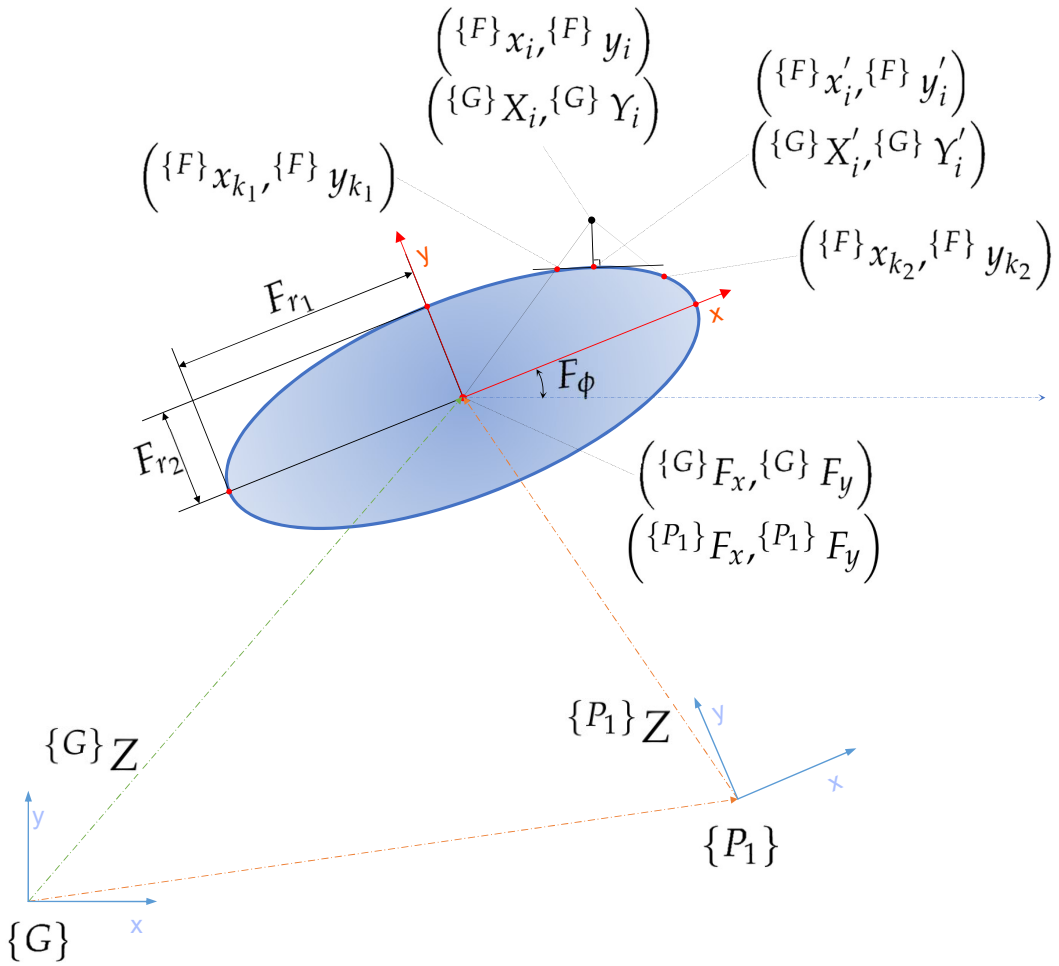


Figure 3.3 : Schematic diagram of conic feature parameterization.

3.2.2 Feature extraction

Many studies on fitting points into ellipses have been conducted. Our method is based on (Ahn et al., 1999) and (Zhang, 1997), supplementing studies on uncertainty flow from sensors to features. This section introduces the implementation approaches and uncertainty transmission.

- Polynomial fitting

The most common and widely used method to fit an ellipse should be polynomial fitting. Given a cluster of points $\mathbf{P} = \{p_i = (x_i, y_i) \mid i = 1, 2, 3, \cdots, n\}$ aligned on the surface of an arbitrary ellipse, obviously all of these points must satisfy the conic equation. The main and minor axes are normalized by $F_{r1} \geq F_{r2}$. With the parameterization mentioned above it is easy to give the equation at each point as follows:

$$\begin{aligned} f(p_i) = & \frac{\left((x_i - F_x)\cos F_\phi + (y_i - F_y)\sin F_\phi\right)^2}{F_{r_1}^2} \\ & + \frac{\left((x_i - F_x)\sin F_\phi - (y_i - F_y)\cos F_\phi\right)^2}{F_{r_2}^2} - 1 \approx 0 \end{aligned} \tag{3.2}$$

After series of simplification and like terms combination, a general polynomial form of conic equation is denoted by the following equation:

$$f(p_i) = Ax_i^2 + 2Bx_iy_i + Cy_i^2 + 2Dx_i + 2Ey_i + F \approx 0 \tag{3.3}$$

where A, B, C, D, E, F, are polynomial coefficients (Zhang, 1997) .

Obviously a trivial solution that all of the coefficients are equal to 0 is good for nothing. To avoid such a situation, several normalization ways can be employed. In this chapter, we normalized $A + C = 1$. Then for all the n points Eq. (3.3) can be revised into vector form:

$$f(\mathbf{v}) = W\mathbf{v} - \mathbf{b} \tag{3.4}$$

where

$$\begin{aligned} W &= [\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3, \cdots, \mathbf{W}_n]^\intercal \\ \mathbf{b} &= [b_1, b_2, b_3, \cdots, b_n]^\intercal \\ \mathbf{v} &= [A, B, D, E, F]^\intercal \\ \mathbf{W}_i &= \left[x_i^2 - y_i^2, 2x_iy_i, 2x_i, 2y_i, 1\right]^\intercal \\ b_i &= -y_i^2 \end{aligned} \tag{3.5}$$

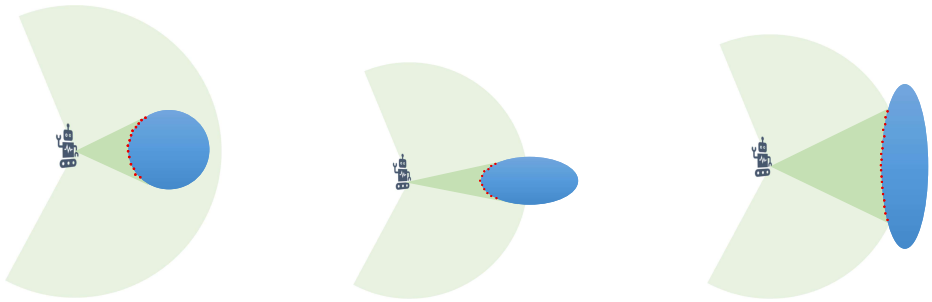
Hence the linear least squares problem becomes

$$\min \mathcal{F}_{\mathbf{v}} = \frac{1}{2} \left(W\mathbf{v} - \mathbf{b} \right)^{\intercal} \left(W\mathbf{v} - \mathbf{b} \right) \tag{3.6}$$

This problem has a closed-form solution:

$$\mathbf{v}^{*} = \left(W^{\intercal} W \right)^{-1} W^{\intercal} \mathbf{b} \tag{3.7}$$

Generally an ellipse can be fitted with a series of points through the polynomial method talked above. Nevertheless, a robot can always "see" the object in one single direction as is shown in Fig. 3.4 which implies proper distributed points result in good ellipse fitting. For example, when the robot is in the position as is illustrated in Fig. 3.4a, the results obtained by Eq. (3.7) are highly reliable. But if the robot happens to observe the extreme narrow end or extreme flat end of an ellipse (Fig. 3.4b and Fig. 3.4c), the observed points may contribute badly to the polynomial function, leading to a totally wrong result or even a complex solution. Therefore an enhanced method should be imposed.



(a) Circular feature. (b) Narrow end. (c) Wide end.

Figure 3.4 : Different situations when the robot observes a conic feature.

- Orthogonal point fitting

In order to overcome the problem discussed in the previous section, one enhanced approach is to minimize the orthogonal distance which is invariant to rigid

transformations in Euclidean space and which presents low curvature bias. Fig. 3.3 depicts various intermediate variables needed in the derivation process. The coordinate transformation of point cloud $\mathbf{P}$ between global coordinate $\{G\}$ and feature coordinate $\{F\}$ is defined by rotation matrix $R = \begin{bmatrix} \cos F_\phi & -\sin F_\phi \\ \sin F_\phi & \cos F_\phi \end{bmatrix}$ and ellipse center ${}^{\{G\}}\mathbf{F_c} = \left[{}^{\{G\}}F_x, {}^{\{G\}}F_y\right]^\intercal$ by

$$ {}^{\{F\}}\mathbf{P} = R^{-1}\left({}^{\{G\}}\mathbf{P} - {}^{\{G\}}\mathbf{F_c}\right) \tag{3.8} $$

Because the feature coordinate $\{F\}$ is defined in the standard ellipse form, we can directly apply standard ellipse equation to the point cloud ${}^{\{F\}}\mathbf{p}$.

For any given point ${}^{\{F\}}\mathbf{p}_i = (x_i, y_i) \in {}^{\{F\}}\mathbf{P}$ in ellipse frame, it is easy to find the orthogonal point ${}^{\{F\}}\mathbf{p}_i^{'} = \left(x_i^{'}, y_i^{'}\right)$ located on the ellipse by solving tangent line equation and the standard ellipse equation as follows:

$$ f\left({}^{\{F\}}\mathbf{p_i}^{'}\right) := \begin{cases} f_1(x_i^{'}, y_i^{'}) = \frac{1}{2}\left(F_{r_1}^2 y_i^{'2} + F_{r_2}^2 x_i^{'2} - F_{r_1}^2 F_{r_2}^2\right) \\ f_2(x_i^{'}, y_i^{'}) = F_{r_2}^2 x_i^{'}\left(y_i^{'} - y_i\right) - F_{r_1}^2 y_i^{'}\left(x_i^{'} - x_i\right) \end{cases} \tag{3.9} $$

Given the point ${}^{\{G\}}\mathbf{p}_i : (X_i, Y_i)$ in coordinate $\{G\}$, transform the point to coordinate $\{F\}$ firstly obtaining ${}^{\{F\}}\mathbf{p}_i : (x_i, y_i)$, then the orthogonal point ${}^{\{F\}}\mathbf{p}_i^{'} : \left(x_i^{'}, y_i^{'}\right)$ can be found by adapting generalized Newton method iteratively to Eq. (3.9) through the following functions:

$$ \begin{aligned} H &= \left.\frac{\partial f}{\partial\left(\mathbf{x}\right)}\right|_{\mathbf{x}={}^{\{F\}}\mathbf{p}_i^{'}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_i^{'}} & \frac{\partial f_1}{\partial y_i^{'}} \\ \frac{\partial f_2}{\partial x_i^{'}} & \frac{\partial f_2}{\partial y_i^{'}} \end{bmatrix} \\ &= \begin{bmatrix} F_{r_2}^2 x_i^{'} & F_{r_1}^2 y_i^{'} \\ \left(F_{r_2}^2 - F_{r_1}^2\right) y_i^{'} - F_{r_2}^2 y_i & \left(F_{r_2}^2 - F_{r_1}^2\right) x_i^{'} + F_{r_1}^2 x_i \end{bmatrix} \\ H\Delta &= -f\left(\mathbf{x}_k\right) \\ \mathbf{x}_{k+1} &= \mathbf{x}_k + \Delta \end{aligned} \tag{3.10} $$

The initial guess $\mathbf{x}_0$ for solving Eq. (3.10) can be given by approximately calculating the midpoint of two intersection points, where one point $\left({}^{\{F\}}x_{k_1}, {}^{\{F\}}y_{k_1}\right)$ is the intersection of line $\overrightarrow{{}^{\{F\}}F_{\mathbf{c}}\,{}^{\{F\}}\mathbf{p}_i}$ and the ellipse. The other point $\left({}^{\{F\}}x_{k_2}, {}^{\{F\}}y_{k_2}\right)$ is the intersection of the perpendicular line at ${}^{\{F\}}\mathbf{p}_i$ with respect to the ellipse's major axis and the ellipse. The three points are calculated by:

$$ {}^{\{F\}}\mathbf{x}_k = \frac{1}{2}\left({}^{\{F\}}\mathbf{x}_{k_1} + {}^{\{F\}}\mathbf{x}_{k_2}\right) \tag{3.11} $$

where

$$ \mathbf{x}_{k_1} = \begin{pmatrix} {}^{\{F\}}x_{k_1} \\ {}^{\{F\}}y_{k_1} \end{pmatrix} \cdot \frac{F_{r_1}F_{r_2}}{\sqrt{F_{r_2}^2\,{}^{\{F\}}x_{k_1}^2 + F_{r_1}^2\,{}^{\{F\}}y_{k_1}^2}} $$

$$ \mathbf{x}_{k_2} = \begin{cases} \begin{bmatrix} {}^{\{F\}}x_{k_2} \\ \operatorname{sign}\left({}^{\{F\}}y_{k_2}\right) \cdot \frac{F_{r_2}}{F_{r_1}}\sqrt{F_{r_1}^2 - {}^{\{F\}}x_{k_2}^2} \end{bmatrix} & \textbf{if } \left|{}^{\{F\}}x_{k_2}\right| < F_{r_1} \\ \begin{bmatrix} \operatorname{sign}\left({}^{\{F\}}x_{k_2}\right) \cdot F_{r_1} \\ 0 \end{bmatrix} & \textbf{if } \left|{}^{\{F\}}x_{k_2}\right| \geqslant F_{r_1} \end{cases} \tag{3.12} $$

The orthogonal point ${}^{\{F\}}\mathbf{p}_i^{'} : \left({}^{\{F\}}x_i^{'}, {}^{\{F\}}y_i^{'}\right)$ is finally obtained after iterative calculation of Eq. (3.10). At last the orthogonal error distance is minimized by the equation:

$$ \min \sum_i \|e_i\|_\Sigma^2 = \sum \left\| {}^{\{G\}}\mathbf{p}_i - {}^{\{G\}}\mathbf{p}_i^{'} \right\|_\Sigma^2 \tag{3.13} $$

after transferring the orthogonal point from feature frame to the global frame ${}^{\{G\}}\mathbf{p}_i^{'} : \left({}^{\{G\}}X_i^{'}, {}^{\{G\}}Y_i^{'}\right)$. Σ is the covariance matrix of the intrinsic noise of the sensor.

Noted that we have defined ellipse parameters vector $\mathbf{\Phi}$ in global frame, deriva-

tives of $\mathbf{\Phi}$ can be found through Eq. (3.8) and Eq. (3.9):

$$
\begin{aligned}
J_{\{F\}\mathbf{p}_i',\Phi} &= \left(\frac{\partial \mathbf{x}}{\partial \Phi}\right)\Big|_{\mathbf{x}=\{F\}\mathbf{p}_i'} \\
&= \begin{bmatrix} -\cos F_\phi & -\sin F_\phi & y_i' & 0 & 0 \\ \sin F_\phi & -\cos F_\phi & -x_i' & 0 & 0 \end{bmatrix}\Bigg|_{\mathbf{x}=\{F\}\mathbf{p}_i'} \\
J_{\{G\}\mathbf{p}_i',\Phi} &= \left(\frac{\partial \mathbf{X}}{\partial \Phi}\right)\Big|_{\mathbf{X}=\{G\}\mathbf{p}_i'} \\
&= R^{-1}\left(\frac{\partial \mathbf{x}}{\partial \Phi}\right)\Big|_{\mathbf{x}=\{F\}\mathbf{p}_i'} \\
&+ \begin{bmatrix} 1 & 0 & -x_i'\sin F_\phi - y_i'\cos F_\phi & 0 & 0 \\ 0 & 1 & x_i'\cos F_\phi - y_i'\sin F_\phi & 0 & 0 \end{bmatrix}\Bigg|_{\mathbf{x}=\{F\}\mathbf{p}_i'}
\end{aligned}
\tag{3.14}
$$

The Jacobian matrix is to be derived after series of reductions by

$$
J_{\{G\}\mathbf{p}_i',\Phi} = \left(R^{-1}H^{-1}C\right)\big|_{\mathbf{x}=\{F\}\mathbf{p}_i'} \tag{3.15}
$$

where H is the Jacobian matrix from Eq. (3.10), and $C = (\mathbf{C_1}, \mathbf{C_2}, \mathbf{C_3}, \mathbf{C_4}, \mathbf{C_5})$:

$$
\begin{aligned}
\mathbf{C}_1 &= \begin{bmatrix} F_{r_2}^2 x_i' \cos F_\phi - F_{r_1}^2 y_i' \sin F_\phi \\ F_{r_2}^2 \left(y_i - y_i'\right)\cos F_\phi + F_{r_1}^2 \left(x_i - x_i'\right)\sin F_\phi \end{bmatrix} \\
\mathbf{C}_2 &= \begin{bmatrix} F_{r_2}^2 x_i' \sin F_\phi + F_{r_1}^2 y_i' \cos F_\phi \\ F_{r_2}^2 \left(y_i - y_i'\right)\sin F_\phi - F_{r_1}^2 \left(x_i - x_i'\right)\cos F_\phi \end{bmatrix} \\
\mathbf{C}_3 &= \begin{bmatrix} \left(F_{r_1}^2 - F_{r_2}^2\right) x_i' y_i' \\ \left(F_{r_1}^2 - F_{r_2}^2\right)\left(x_i'^2 - y_i'^2 - x_i' x_i + y_i' y_i\right) \end{bmatrix} \\
\mathbf{C}_4 &= \begin{bmatrix} F_{r_1}\left(F_{r_2}^2 - y_i'^2\right) \\ 2F_{r_1} y_i'\left(x_i - x_i'\right) \end{bmatrix} \\
\mathbf{C}_5 &= \begin{bmatrix} F_{r_2}\left(F_{r_1}^2 - x_i'^2\right) \\ -2F_{r_2} x_i'\left(y_i - y_i'\right) \end{bmatrix}
\end{aligned}
\tag{3.16}
$$

Finally, the value of ellipse parameters will be solved through iteratively minimizing the orthogonal error distance Eq. (3.13) among all of the given points.

Noticed that an initial guess is still inevitable even using orthogonal point fitting method. Thus in this research the result of Eq. (3.7) is considered as initial guess. The algorithm of ellipse fitting is explicated in Algorithm 1.

Algorithm 1: Ellipse fitting

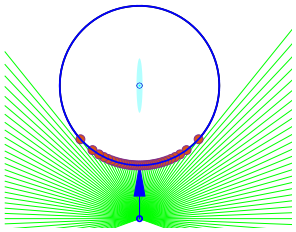
Input: Scan points p_i in local frame

Output: Parameters $\tilde{F}$ in global frame

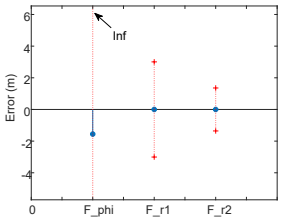
```
if Enough points then
    Solve initial guess F̃_0 by polynomial fitting via Eq. (3.7)
    if Curvature change of point sets > λ_φ then
        while Not Converged do
            Calculate F̃ with F̃_0 by orthogonal point fitting via Eq. (3.13)
            iteratively
        end
    else
        Wait for the next observation
    end
end
```

Notation: λ_ϕ is an empirical criterion to filter near flat distributed points.

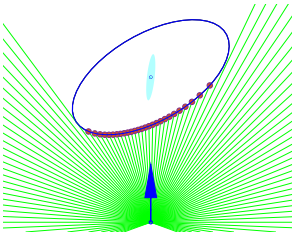
Several steps are taken to improve the fitting accuracy. At first, enough input points are necessary to start fitting. If the number of observed points is larger than a certain value, the fitting process will continue. The initial value $\tilde{F}_0$ is given by fitting with polynomial Eq. (3.7). Then calculate the curvatures of both ends of the points and finding the difference. If the difference is acceptable after compared with an empirical criterion λ_ϕ, $\tilde{F}$ is to be solved by iteratively doing Eq. (3.13).



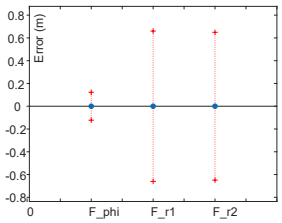
(a) Circular feature (no noise). Axes are in meters.



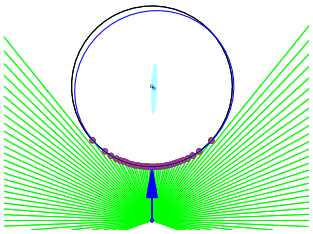
(b) Angle error and axis error of circular feature (no noise).



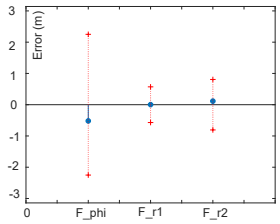
(c) Elliptic feature (no noise). Axes are in meters.



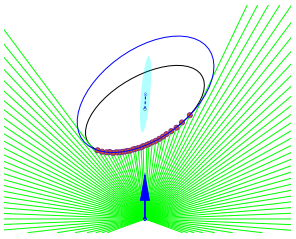
(d) Angle error and axis error of elliptic feature (no noise).



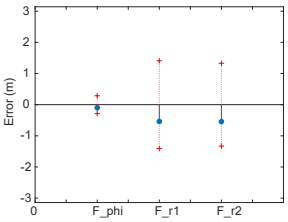
(e) Circular feature (with noise). Axes are in meters.



(f) Angle error and axis error of circular feature (with noise).



(g) Elliptic feature (with noise). Axes are in meters.



(h) Angle error and axis error of elliptic feature (with noise).

Figure 3.5 : Uncertainty after fitting process. Left side figures show the error and uncertainty of translation (Error is depicted by dash lines, uncertainty is depicted by light blue elliptical range). Right side figures show the error and uncertainty of angle and axis dimension (From left to right each bar is corresponded of angle, major axis, and minor axis respectively).

3.2.3 Uncertainty transmission

As we know, "ellipse fitting" is an approximation of the raw data, there is information loss during the procedure. It is untrusted to utilize fitted parameters without analyzing uncertainty transmission process. If we denote sensor's covariance matrix as Σ_s, the information matrix of parameterized feature Σ_f can be calculated by:

$$\Sigma_f = J^{-\intercal}\Sigma_s J^{-1} \tag{3.17}$$

It should be noted that two cases will cause ill-condition problem of Jacobian J, one is that the point $\mathbf{p_i}$ locates at the ellipse center, while the other case is when the ellipse has two similar axes (close to a circle, which is a special case of ellipse equation).

Fig. 3.5 compares errors and uncertainty of one feature's individual parameter. Firstly, considering the theoretical cases where no noise exists in the observation, error of each parameter is always zero without any doubt (See Fig. 3.5a - Fig. 3.5d). A remarkable part is that all the errors are strictly within the scope of their corresponding uncertainty except circular feature's angle which is not zero and the corresponding uncertainty is marked as infinite. It is caused by the same dimension of major and minor axis that deriving angle turns to be unreliable. Noise cannot be ignored when a robot is handled in real world (See Fig. 3.5e - Fig. 3.5h). As we have analyzed the uncertainty transmission above, uncertainty caused by sensors are evidently transmitted to the fitted parameters. All the errors are significantly limited in the range of calculated uncertainty. Even for a badly fitted result (Fig. 3.5g), the fitted ellipse diverges from the exact model but all the errors are reasonably in the range of uncertainty.

3.2.4 Data association

Data association is a difficult problem in SLAM, especially in certain complicated environment. When a robot works in an open environment, there are two cases where observations from lidar sensors do not always occur: one is no object exists within a valid lidar range, another is no acceptable feature parameters fitted in one single observation.

Due to the sparse observation distribution, a valid odometry information is needed to handle such no-observation situation. Then data association is easily done with the odometry information since features are widely dispersed. Also, if observations from a single view are relatively dense, the association can be done by determining whether there are break points or not. Different from conventional lidar SLAM, each feature defined in our parameterization possesses a center, an angle and a pair of geometry dimensions, and these parameters can be taken into account if correspondences are found, since using pure points is more complicated to solve nearest neighbor for the sake of large size and dense distribution. Noted that if no valid fitted features appear at a certain step, this step is marked by "no observation" and no edges are added between this step's node and other feature poses' node (discussed in Section 3.3). Then Pre-fit SLAM can overcome the challenge of no consecutive observations.

As is shown in Fig. 3.6, a newly fitted feature as well as odometry information are considered at the same time to determine whether this feature appeared or not. If this newly fitted feature appears for the first time (not the same feature in the feature list), it is added to the feature list. Otherwise, this fitted feature will be labeled as the occurred feature within the feature list. For example, the temporary robot pose $\tilde{\boldsymbol{\Xi}}_i$ is calculated by accumulating odometry information, with which a certain observed feature's center coordinate $\tilde{\mathbf{Z}}_{\text{feature},i,j}$ can be transformed to the

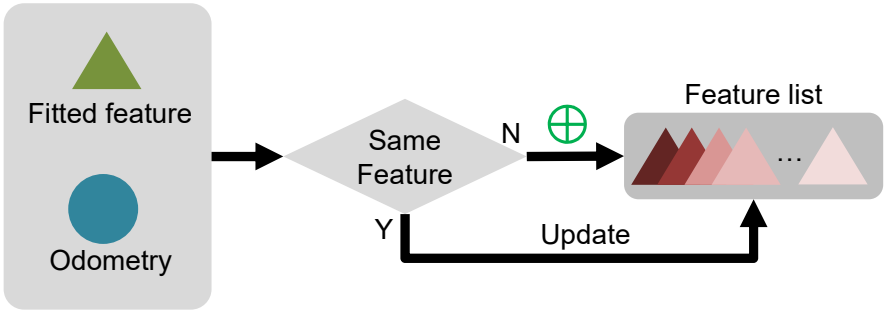
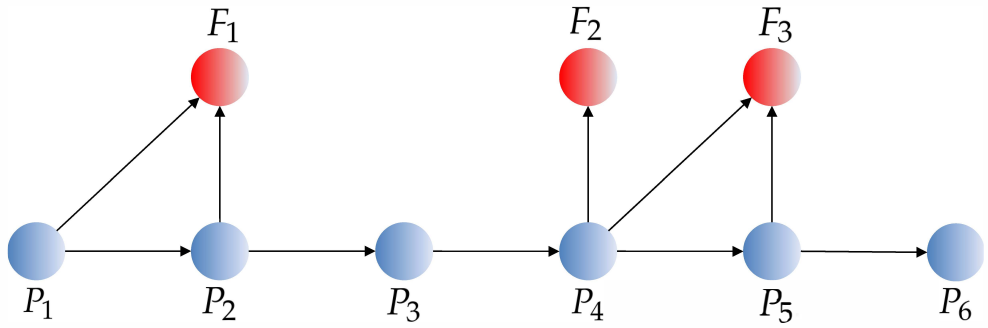


Figure 3.6 : Flow chart of data association.

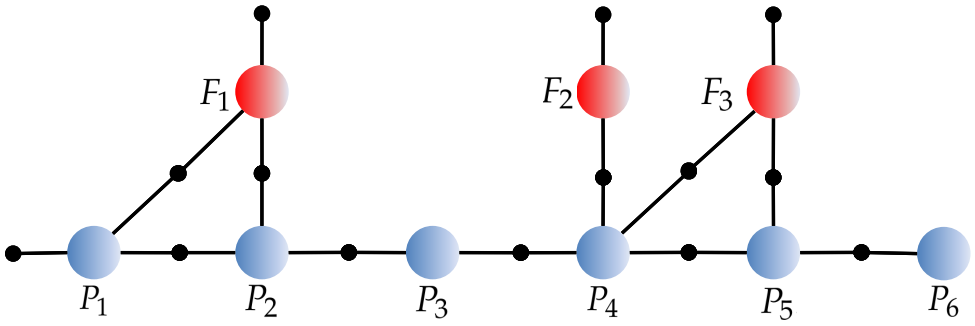
3.3 Graph-optimization

This section focuses on the back-end optimization of the proposed algorithm. Thanks to the data processing section we are provided with an initial graph of robot poses and features. The remainder of this section briefly introduces factor graph SLAM for our problem.

When a robot is moving in a 2D space, its state vector can be described by $\boldsymbol{\Xi} = (x, y, \theta)$. Remarkably, according to the parameterization discussed above, we form the first three parameters $[F_x, F_y, F_\phi]$ as the feature's "pose" $\tilde{\mathbf{F}}$ with geometry properties $\mathbf{F}_r$. With our feature parameterization approach each feature can be re-expressed as a feature pose and a dimensional part denoted by $\mathbf{F}$, which is a combination of $\tilde{\mathbf{F}}$ and $\mathbf{F}_r$. Let us assume a simplified structure (Fig. 3.7a). Blue nodes are robot poses and red nodes are features, and each observation is represented by an arrow edge. When express this structure via factor graph (Fig. 3.7b), one observation is represented by an edge with a black point. Robot poses follow a motion model with the input $\mathbf{u} = (\delta x, \delta y, \delta\theta)$ and edges linking features and poses follow an observation model similarly. Noticed that every feature node is connected



(a) Graph structure.



(b) Factor graph.

Figure 3.7 : Optimization structure.

In the factor graph $\mathcal{F} = (\mathcal{M}, \mathcal{X}, \varepsilon)$, we can denote factors, variables and edges as $\phi_i \in \mathcal{M}$, $x_i \in \mathcal{X}$ and $e_{ij} \in \varepsilon$, respectively. Writing all of the variables for an assignment to the set X_i, we can define the global factorized optimization problem of the example as:

$$\underset{X}{\text{argmax}} \quad \phi(X) = \prod_i \phi_i(X_i) \tag{3.18}$$

where ϕ_i is the factor linked with a certain node and has no specific definition. By

taking the negative logarithm of Eq. (3.18), the actual optimization problem in this chapter can be written by:

$$\begin{aligned} \underset{\boldsymbol{\Xi},\mathbf{F}}{\operatorname{argmin}} \quad E_{\text{total}} = \frac{1}{2}\sum_{i=1}^{n}\Big(&\|\mathbf{Z}_{\text{odom},i} - T^{-1}(\boldsymbol{\Xi}_{i-1},\boldsymbol{\Xi}_i)\|^2_{\tilde{\Sigma}_{\text{odom}}} \\ &+ \|\tilde{\mathbf{Z}}_{\text{feature},i,j} - T^{-1}(\boldsymbol{\Xi}_i,\mathbf{F}_j)\|^2_{\tilde{\Sigma}_{\text{feature},i,j}}\Big) \end{aligned} \tag{3.19}$$

where $\mathbf{Z}_{\text{odom},i}$ and $\tilde{\mathbf{Z}}_{\text{feature},i,j}$ are observations of odometry and fitted features, respectively. Note that although $\mathbf{F}_j$ occurs in the frame transformation operator $T^{-1}(\cdot)$, the actual transformed element is $\tilde{\mathbf{F}}_j$ while the operation of $\mathbf{F}_r$ is independent on the frame transformation.

It is easy to solve Eq. (3.19) by general non-linear least squares method such as Gauss-Newton method or Levenberg-Marquardt method.

3.4 Experiment and analysis

3.4.1 System setup

The experiment overview on synthetic and practical data is interpreted in this section. The platform in our experiment is the Fetch robot (Wise et al., 2016). It is equipped with a SICK 2D laser scanner at 15 Hz and has a 220 degree field of view with an angular resolution of 0.3323 degree and a 25 meter valid range. The synthetic data was collected via the simulator designed for Fetch robot, including Fetch model and working environment. All the simulation parameters are set the same as a real Fetch while we assumed the observation noise and odometry noise obey zero mean Gaussian distribution $n_s \sim \mathcal{N}(\mathbf{0}, \Sigma_s)$ and $n_o \sim \mathcal{N}(\mathbf{0}, \Sigma_o)$ respectively, which are supposed to be similar to the real robot's noise. During the simulation Σ_s was set to $(0.02\text{m}, 0.02\text{m})$ with regard to $(\delta x, \delta y)$ for the laser point in Cartesian coordinate, and Σ_o was set to $(0.05\text{m}, 0.05\text{m}, 0.001\text{rad})$ with regard to $(\delta x, \delta y, \delta\theta)$ for the odometry.

We built three different simulated environments and one real scenario to test our algorithm and compared with common algorithms, namely Cartographer (Hess et al., 2016), ICP (Bergström and Edlund, 2014) and NDT (Biber and Straßer, 2003). We also conducted experiments to analyze the uncertainty during fitting. Furthermore, we compared the uncertainty between the proposed method and filter-based method. In our case, an EKF approach in the proposed feature parameterization was selected.

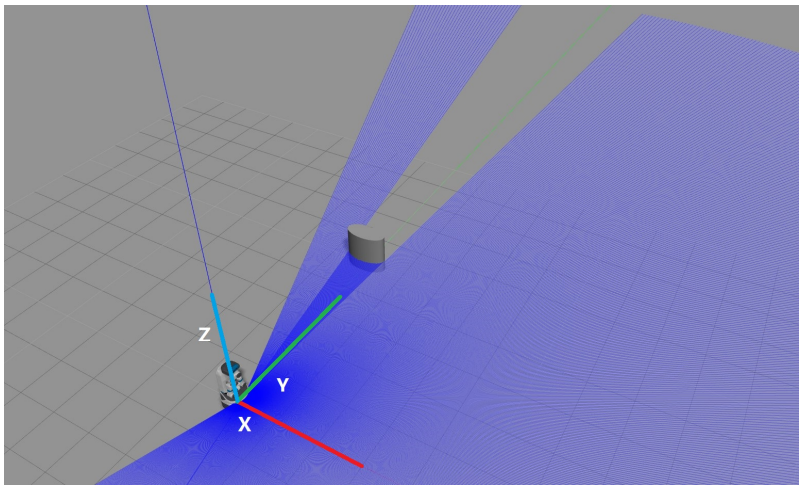


Figure 3.8 : Schematic diagram of uncertainty analysis experiment.

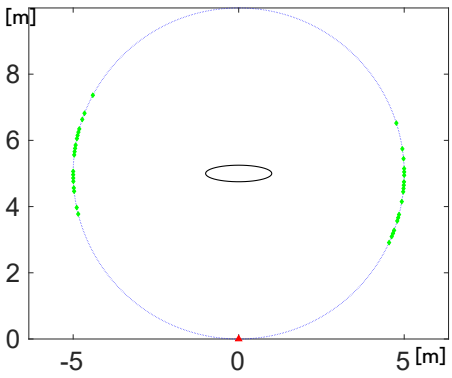
3.4.2 Results on accuracy of feature fitting using simulations

We tested the fitting process at various observing angles because the mobile robot cannot observe objects ideally. In the simulator, Fetch robot made a counter-clockwise circular motion around the object 3 meters from the robot on y-axis (as shown in Fig. 3.8). Because the dimension of two axes are able to judge the fitting performance intuitively, we only compare F_{r_1} and F_{r_2} in this case. Table 3.1 shows the average error percentage of different dimensional features: Feature $1 \sim (F_{r_1} = 1\text{m}, F_{r_2} = 0.25\text{m})$, Feature $2 \sim (F_{r_1} = 1\text{m}, F_{r_2} = 0.5\text{m})$, Feature $3 \sim (F_{r_1} = 1\text{m}, F_{r_2} = 0.75\text{m})$, and Feature $4 \sim (F_{r_1} = 1\text{m}, F_{r_2} = 1\text{m})$. The fitting process in each case was performed for 15 times before the final results were obtained.

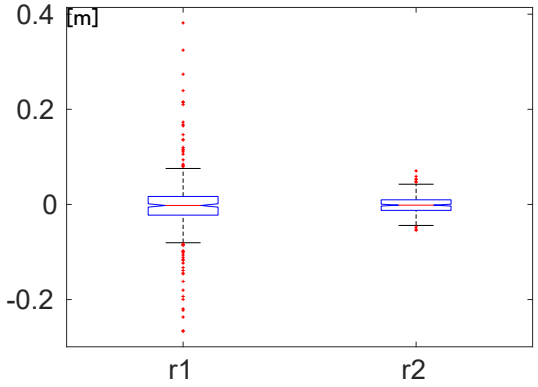
Table 3.1 : Average error percentage of estimated axis dimension for different features. (Unit: m)

		Feature 1	**Feature 2**	**Feature 3**	**Feature 4**
	Groundtruth	1	1	1	1
F_{r1}	Estimate	1.0117	1.0027	0.9976	1.0018
	Error %	1.17%	0.27%	0.24%	0.18%
	Groundtruth	0.25	0.5	0.75	1
F_{r2}	Estimate	0.2518	0.5015	0.7489	0.9991
	Error %	0.72%	0.3%	0.15%	0.09%

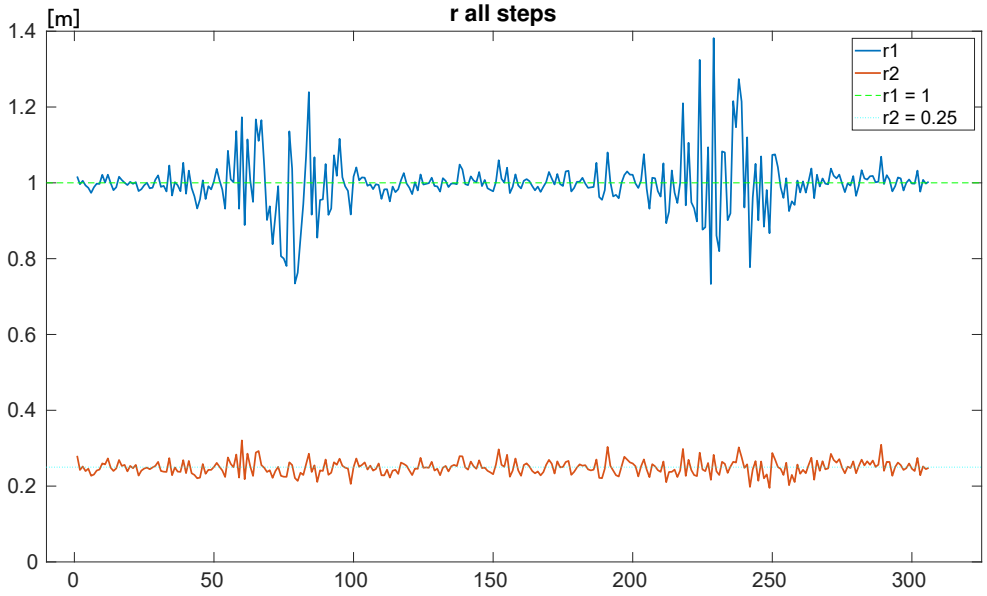
It can be seen errors of four features are reasonably small. The average error percentage of F_{r_1} and F_{r_2} descends with the decreasing of axis ratio $\tau = \frac{F_{r_1}}{F_{r_2}}$. Feature 1 has the largest τ and the biggest error percentage. As is shown in Fig. 3.9c, F_{r_1} has two distinct growth with the increase of steps, but F_{r_2} doesn't represent the similar regularity. The reason of such phenomenon has been explained by Okatani and Deguchi (see Okatani and Deguchi, 2009, Sec 4.2 and Eq. 11), showing that the main axis dimensions will be more biased if the curvature of fitting points is large. The distinct growth occurs when the robot faces the narrow end, where the curvature is larger than that in the flat end. According to Fig. 3.9b, errors of F_{r_2} are not significantly large compared with F_{r_1}. If marking positions where the error percentage of F_{r_1} is over 50% with green diamond markers (see Fig. 3.9a), we can obviously see that these large errors are mostly found at positions where the robot observes feature's narrow end. Fig. 3.9d illustrates error and 3-sigma bounds among all steps. As we can see, the error of F_{r_1} reaches the maximum around 1.5m at the $90th$ step but it is still located within 3-sigma bounds as well as errors of all the other steps.



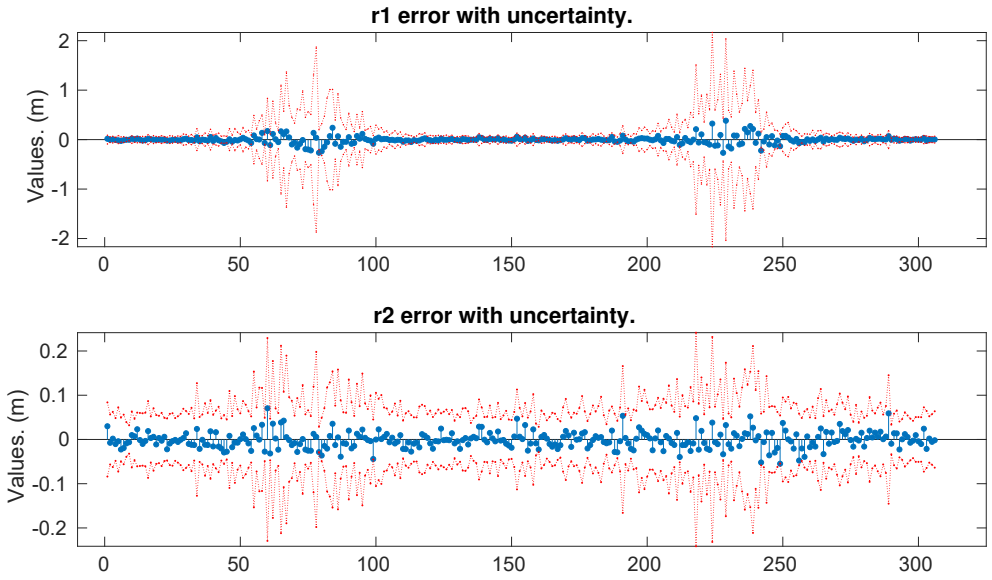
(a) Positions when error percentage is over 50%.



(b) Error distribution for F_{r_1} and F_{r_2}.



(c) Dimensions of F_{r_1} and F_{r_2} each step in one loop



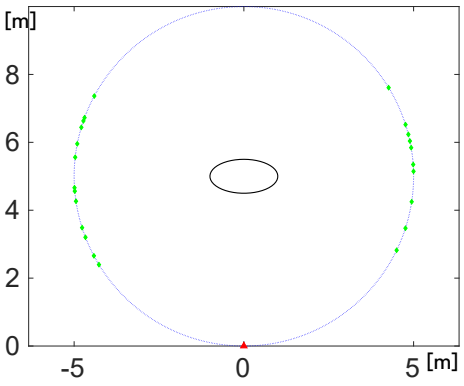
(d) Error and 3-sigma bounds at each step in one loop

Figure 3.9 : Error of F_{r_1} and F_{r_2} for Feature 1.

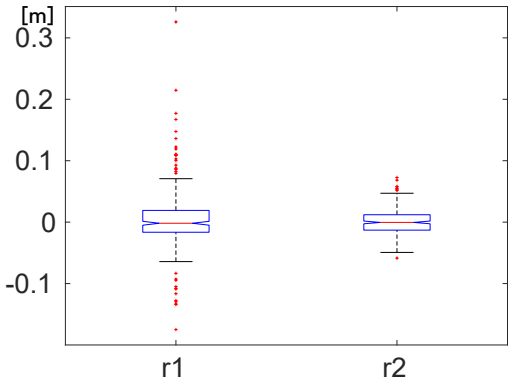
Fig. 3.10a, Fig. 3.11a and Fig. 3.12a show the error occurrence for Feature 2, Feature 3, and Feature 4 respectively. It can be seen that the error-prone positions are most likely occurred when observing the narrow end of features, but the probability of large-error occurrence and the value of errors descend with the increase of τ. By comparing all the four features' error and 3-sigma bounds, it can be found that Feature 1 possesses the largest errors for F_{r_1} and F_{r_2} by around 0.5m and 0.1m (Fig. 3.9d), Feature 2's largest errors locate at 0.29m and 0.09m (Fig. 3.10d), Feature 3 possesses the largest errors for F_{r_1} and F_{r_2} by around 0.17m and 0.08m (Fig. 3.11d) and Feature 4 has the largest errors for 0.1m and 0.1m (Fig. 3.12d). All the errors are strictly limited within the 3-sigma bounds.

3.4.3 Results on simulation for open scenario

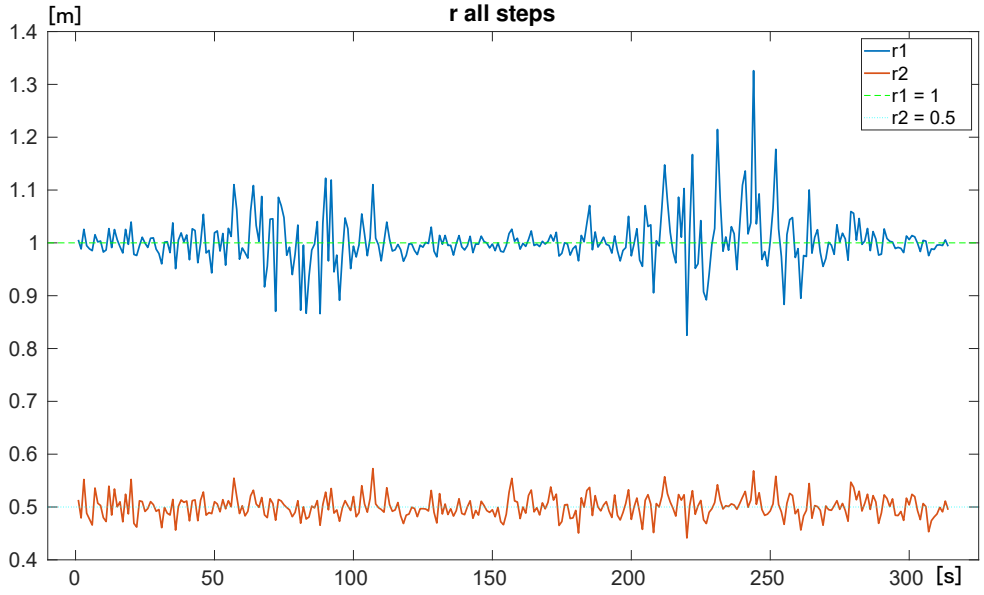
In this section, we constructed multiple expected working scenarios (as shown in Fig. 3.13), and compared Pre-fit SLAM with the state of the art 2D SLAM system Cartographer (Hess et al., 2016), and other widely used algorithm: ICP without an initial guess (set initial guess to zero, denoted by ^{0}ICP), ICP with a good initial guess (set initial guess to the odometry value, denoted by ^{1}ICP), NDT without an initial guess (set initial guess to zero, denoted by ^{0}NDT), and NDT with a good initial guess (set initial guess to the odometry value, denoted by ^{1}NDT). Noted that the results of Cartographer is under the configuration "using odometry information", while the other four approaches does not utilize odometry. We evaluated our algorithm with other methods by comparing difference with groundtruth in x, y, θ via Root-Mean-Square-Error (RMSE) and error per step. In order to clarify the advantages of factor graph optimization in our algorithm, a general extended Kalman filter based SLAM algorithm (Huang and Dissanayake, 2007) was adopted as a controlled group. Since the original algorithm uses point feature, the state vector in the control group is modified to be adapt for the feature parameterization in this chapter. Different from



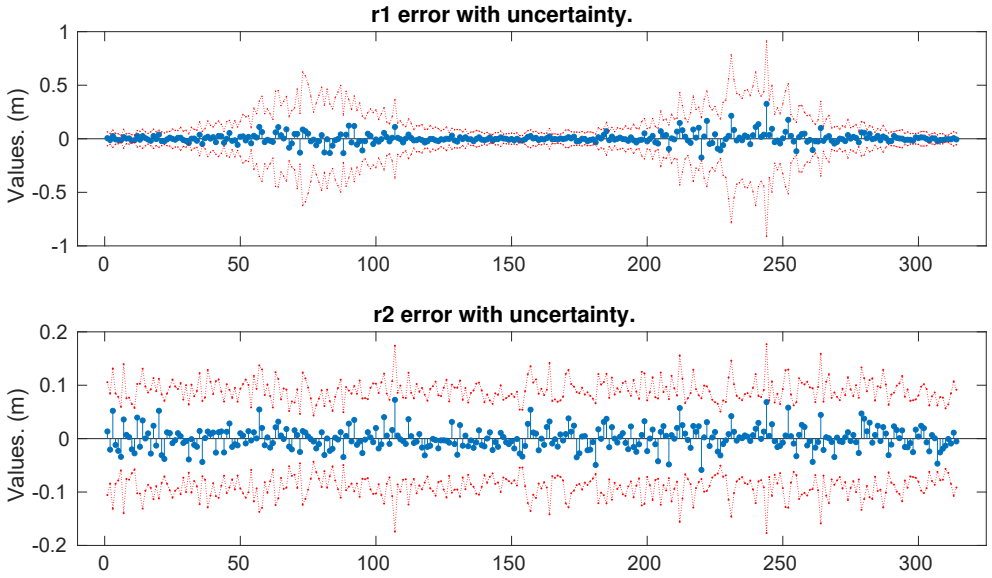
(a) Positions when error percentage is over 50%.



(b) Error distribution for F_{r_1} and F_{r_2}.

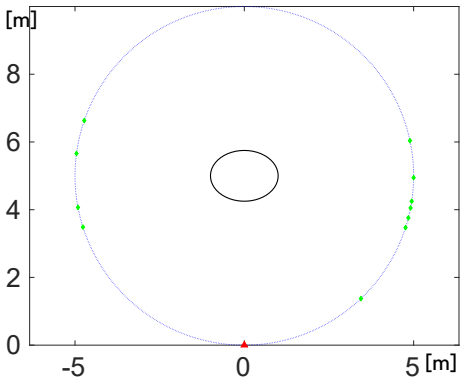


(c) Dimensions of F_{r_1} and F_{r_2} each step in one loop.

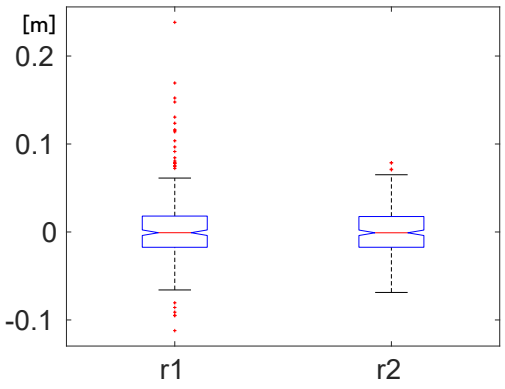


(d) Error and 3-sigma bounds at each step in one loop.

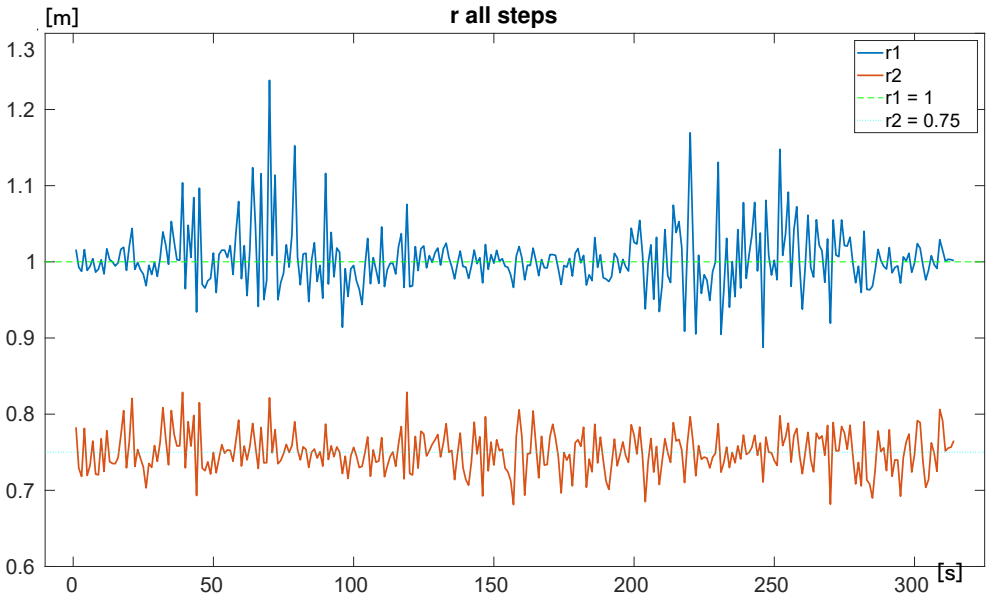
Figure 3.10 : Error of F_{r_1} and F_{r_2} for Feature 2.



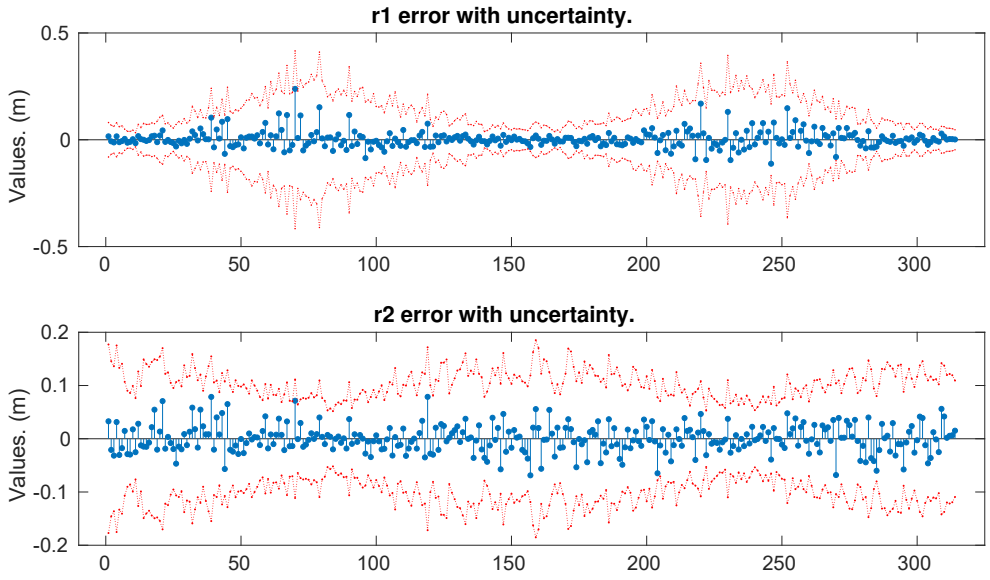
(a) Positions when error percentage is over 50%.



(b) Error distribution for F_{r_1} and F_{r_2}.

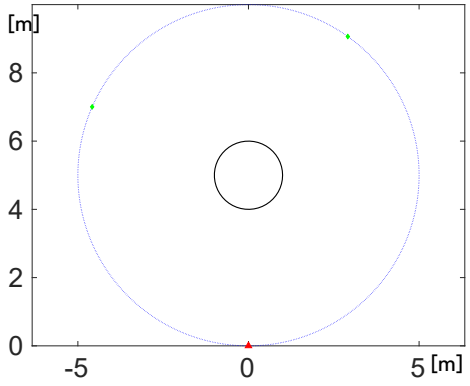


(c) Dimensions of F_{r_1} and F_{r_2} each step in one loop.

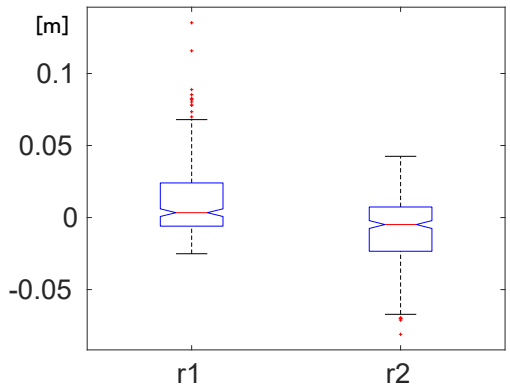


(d) Error and 3-sigma bounds at each step in one loop.

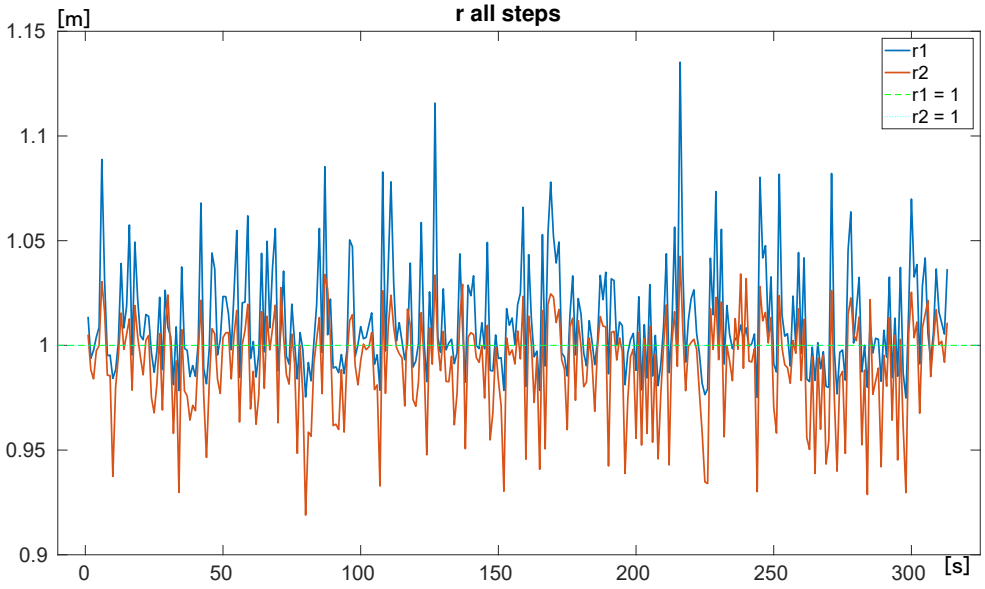
Figure 3.11 : Error of F_{r_1} and F_{r_2} for Feature 3.



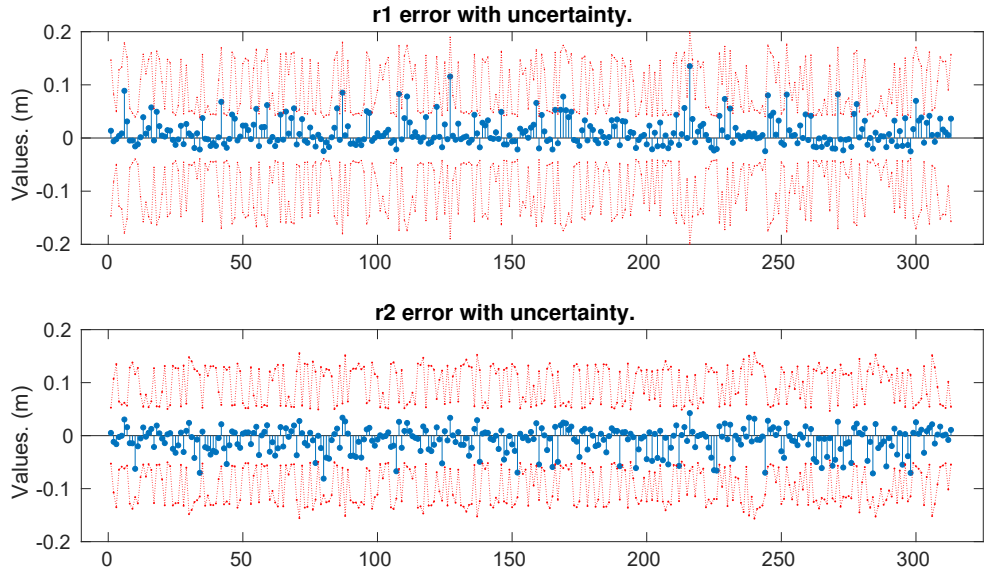
(a) Positions when error percentage is over 50%.



(b) Error distribution for F_{r_1} and F_{r_2}.

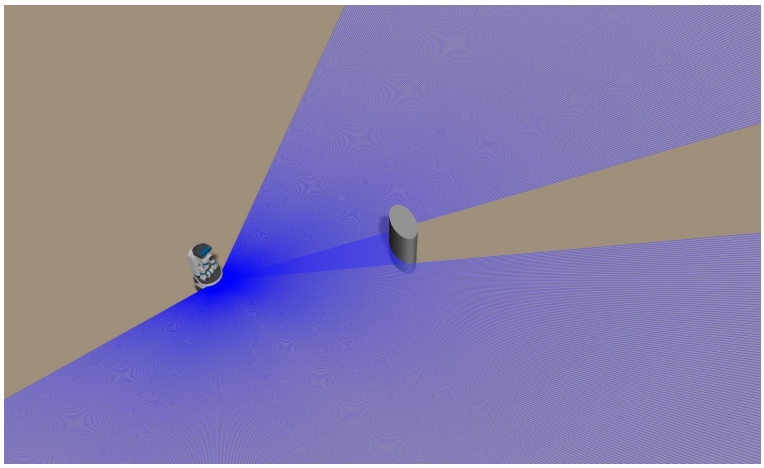


(c) Dimensions of F_{r_1} and F_{r_2} each step in one loop.

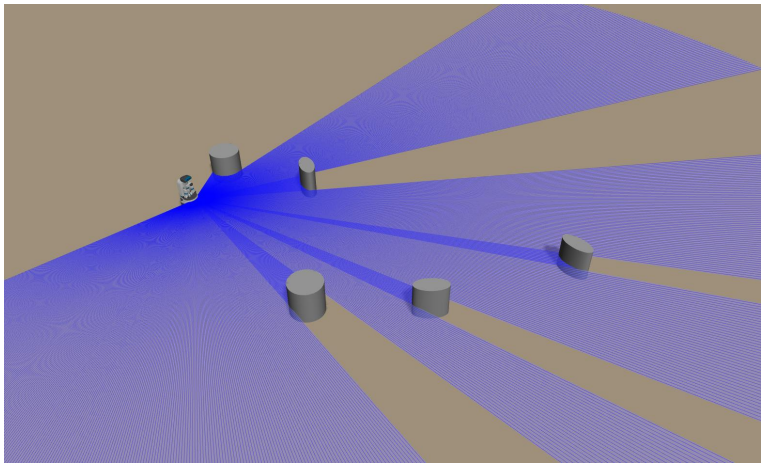


(d) Error and uncertainty at each step in one loop.

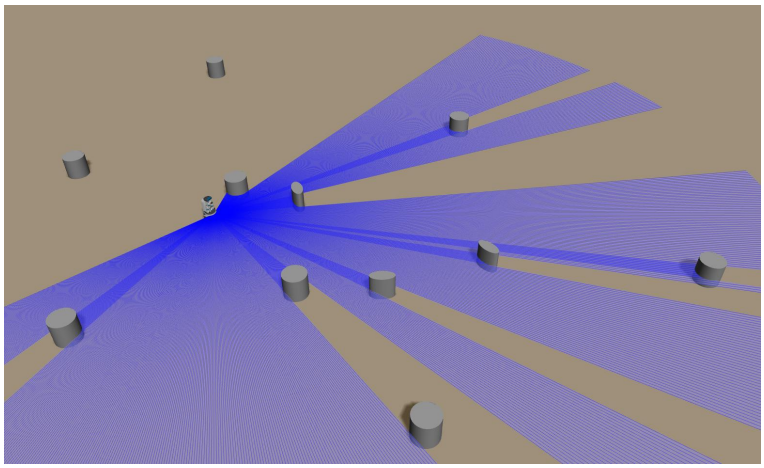
Figure 3.12 : Error of F_{r_1} and F_{r_2} for Feature 4.



(a) Case 1: Single feature.



(b) Case 2: Five features.



(c) Case 3: Eleven features.

Figure 3.13 : Simulation environment. Case 1 contains one single feature, the robot moves around the feature. Case 2 contains five features, the robot moves around all the features. Case 3 contains eleven features, the robot moves though and around the features.

common EKF SLAM, the state vector is composed of current pose and parameterized features under the parameterization of the proposed algorithm (denoted by EKF). Note that estimated features at the last step as well as estimated pose at each individual step are chosen for the purposes of comparison.

Table 3.2 : Feature parameters groundtruth.

		F_x /m	F_y /m	F_ϕ /rad	F_{r_1} /m	F_{r_2} /m
Case 1	F_1	3	2.5	2.3562	0.5	0.25
Case 2	F_1	3	2.5	2.3562	0.5	0.25
	F_2	6	-2.4	-	0.5	0.5
	F_3	9	-1	0.5	0.5	0.25
	F_4	12	2.4	2.7416	0.5	0.25
	F_5	0	2	-	0.5	0.5
Case 3	F_1	9	-1	0.5	0.5	0.25
	F_2	6	-2.4	-	0.5	0.5
	F_3	7	12	-	0.5	0.5
	F_4	3	2.5	2.3562	0.5	0.25
	F_5	0	2	-	0.5	0.5
	F_6	0	-8	-	0.5	0.5
	F_7	12	2.4	2.7416	0.5	0.25
	F_8	13	-6	-	0.5	0.5
	F_9	20	5.5	-	0.5	0.5
	F_{10}	-9	12	-	0.5	0.5
	F_{11}	-8	0	-	0.5	0.5

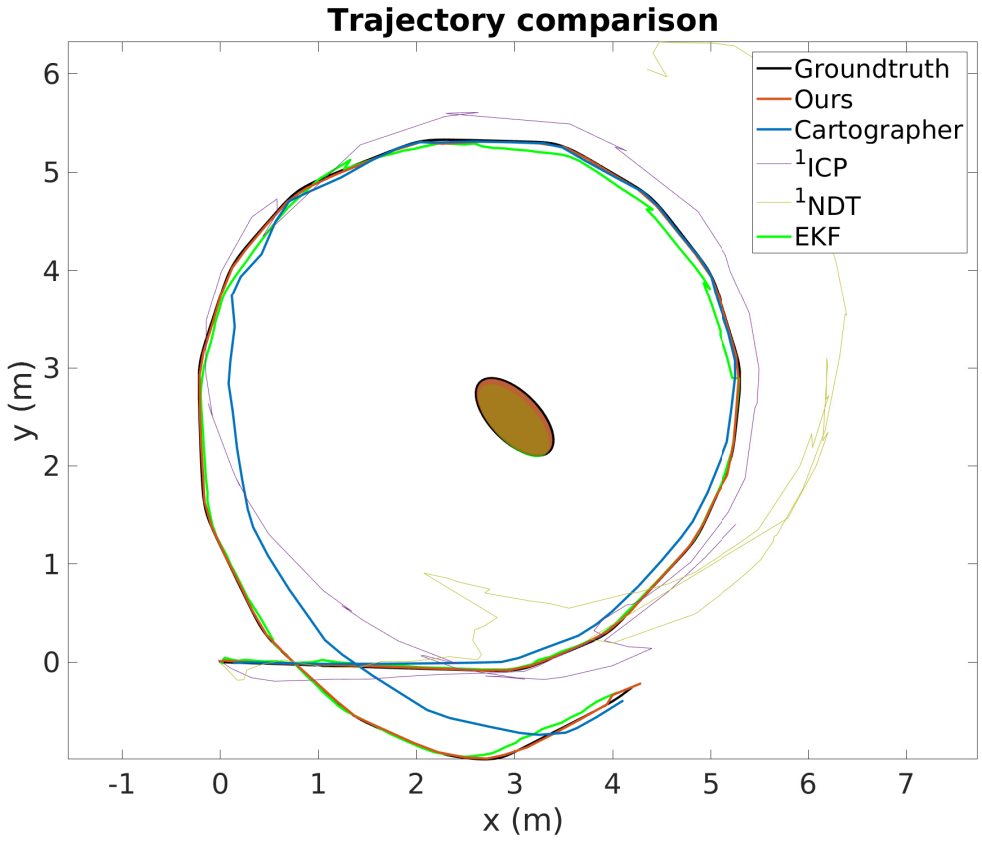
* F_ϕ is denoted by "-" if that feature is circular shaped.

In order to simulate trees in an open environment without importing model-

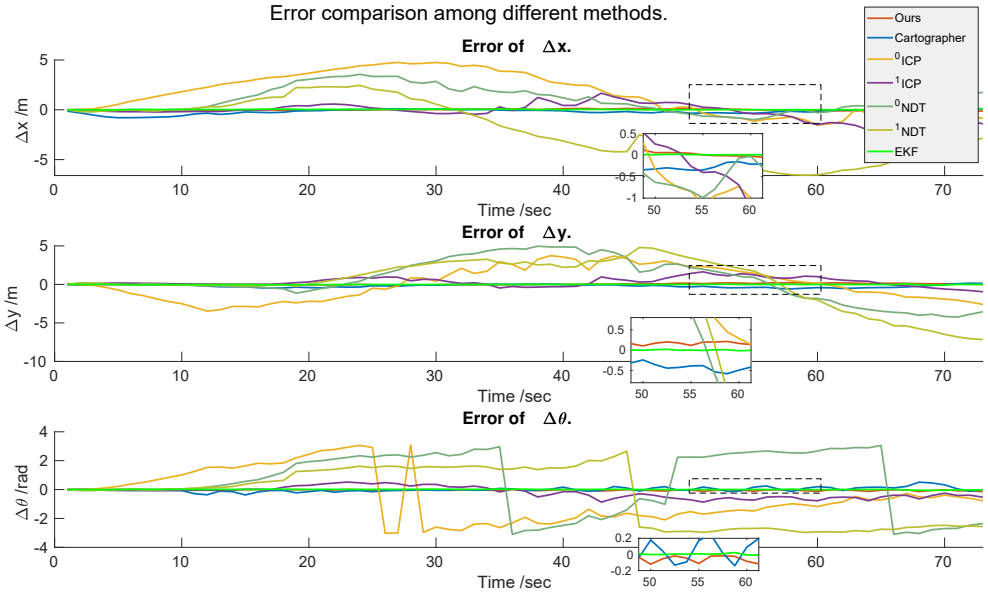
ing complexity, we built features with selected dimensions (0.25m and 0.5m). The dimensions are chosen by considering the fitting difficulty. The fact is that our algorithm still works with other dimensions if a proper outlier elimination strategy is adopted. Table 3.2 gives groundtruth of feature parameters in each case. F_ϕ is denoted by "-" if that feature is circular shaped. All the features are built in Gazebo. To enhance the visualization performance, real feature is filled with brown shadow and estimated result by Pre-fit SLAM is filled with orange color in the trajectories comparison figures (Fig. 3.14a, Fig. 3.15a, Fig. 3.16a, and Fig. 3.20).

In Case 1 (Fig. 3.13a), the robot moved around one single elliptical feature. Trajectory comparison is illustrated in Fig. 3.14a. ^{0}ICP and ^{0}NDT are not depicted for the sake of completely wrong results. A turn back exists in the trajectory of ^{1}NDT which is caused by the similar shape at both sides. ^{1}ICP is better than ^{1}NDT but is still worse than Cartographer. The trajectory of Pre-fit SLAM is the closest to the groundtruth. Fig. 3.14b demonstrates the estimated error of robot pose in $\delta x, \delta y$ and $\delta\theta$ varying with time. Both ICP and NDT cannot provide reasonable result, the maximum errors of δx and δy exceed 4 meters and the variation of rotation error is even greater. Cartographer and Pre-fit SLAM can maintain the error within a small range. In the dash rectangle we enlarged part of the error curve from 50s to 60s. It can be found that the absolute translation error of Cartographer is around 0.4m while ours is within 0.1m. The peak of rotation error of both Cartographer and Pre-fit SLAM can reach 0.2rad but it is clearly seen that ours has a lower average level than that of Cartographer.

In Case 2 (Fig. 3.13b), the robot moved around five features including circular and elliptical shape. Trajectory comparison is illustrated in Fig. 3.15a. ^{0}ICP and ^{0}NDT are not depicted for the sake of completely wrong results. ^{1}ICP and ^{1}NDT still perform badly. The trajectory of Pre-fit SLAM is the closest to the groundtruth. There is a relatively large jump in the Cartographer's trajectory, the reason of



(a) Trajectories comparison among different methods.



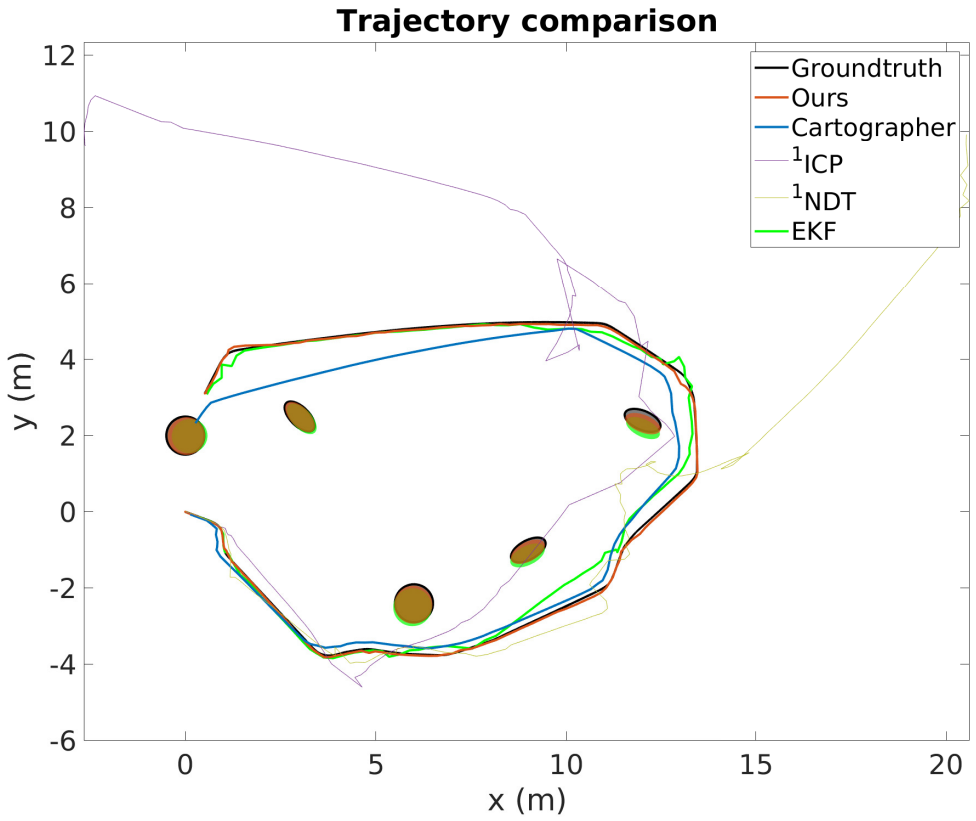
(b) Case 1: Error comparison among different methods.

Figure 3.14 : Case 1: Trajectory and error varying with time.

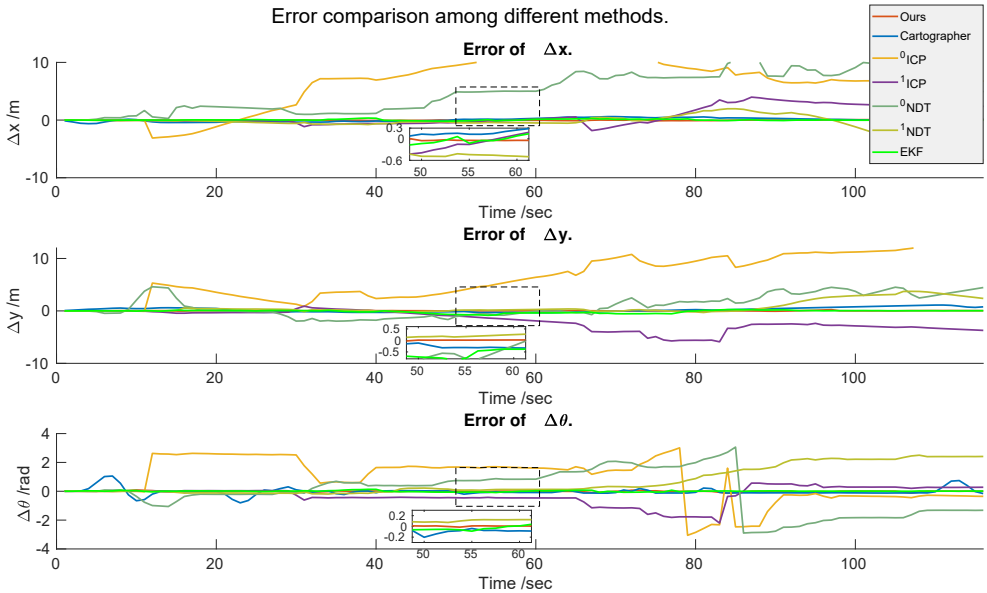
which is that the observed scan points are located on the other side of features in contrast to previous observation. Thus the registration process considers point sets on both side and tries to make the point sets on either side coincide during scan matching, instead of stitching in the shape of an ellipse. Fig. 3.15b demonstrates the estimated error of robot pose in $\delta x, \delta y$ and $\delta\theta$ varying with time. Both ^{0}ICP and ^{0}NDT cannot provide reasonable result, where the translation error and rotation error are too large. The results in the first half part of ^{1}ICP and ^{1}NDT are roughly near the real trajectory, but in the remainder part they diverged because of wrong matching. Cartographer and Pre-fit SLAM can maintain the error within a small range. In the dash rectangle we enlarged part of the error curve from 50s to 60s. It can be found clearly that our error is less than that of Cartographer. The peak of rotation error of Cartographer is even beyond 0.2rad while ours keep the error level stick to near 0. It is worth saying that ^{1}NDT possesses a smaller rotation error and δy error than Cartographer dramatically, due to the increased amount of features.

In Case 3 (Fig. 3.13c), the robot moved through and around eleven features including circular and elliptical shape. Trajectory comparison is illustrated in Fig. 3.16a. ^{0}NDT is not depicted for the sake of completely wrong result. In this case ^{0}ICP, ^{1}ICP and ^{1}NDT are partly trusty when the robot can observe features from both left and right sides. Then the trajectories starts to drift since it can only observe features from a single side. Our trajectory is still the nearest to the groundtruth and Cartographer in this case performs the best compared with the other two cases. Fig. 3.16b demonstrates the pose error of the robot in $\delta x, \delta y$ and $\delta\theta$ varying with time. ^{0}NDT in this case is the method that generate the worst result. In the dash rectangle we enlarged part of the error curve from 80s to 120s. The error of Pre-fit SLAM is still the smallest one.

We also utilized EKF method to evaluate the accuracy of importing factor graph. From Fig. 3.14 to Fig. 3.16 it can be seen directly that Pre-fit SLAM via factor graph

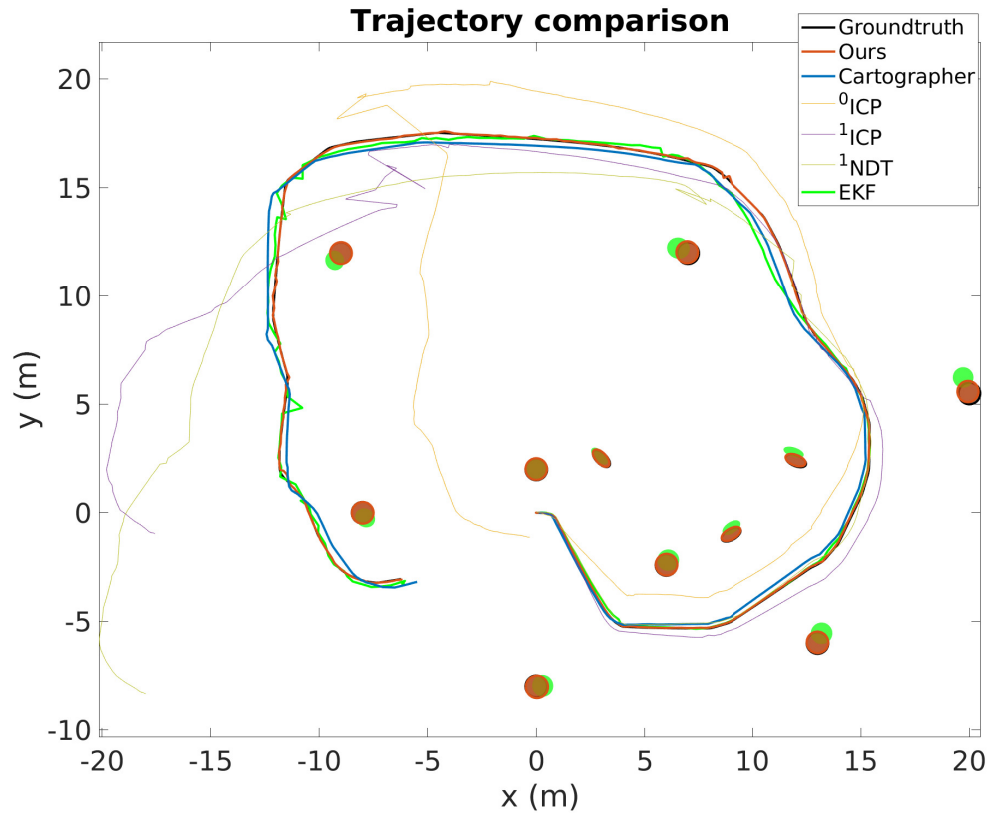


(a) Trajectories comparison among different methods.

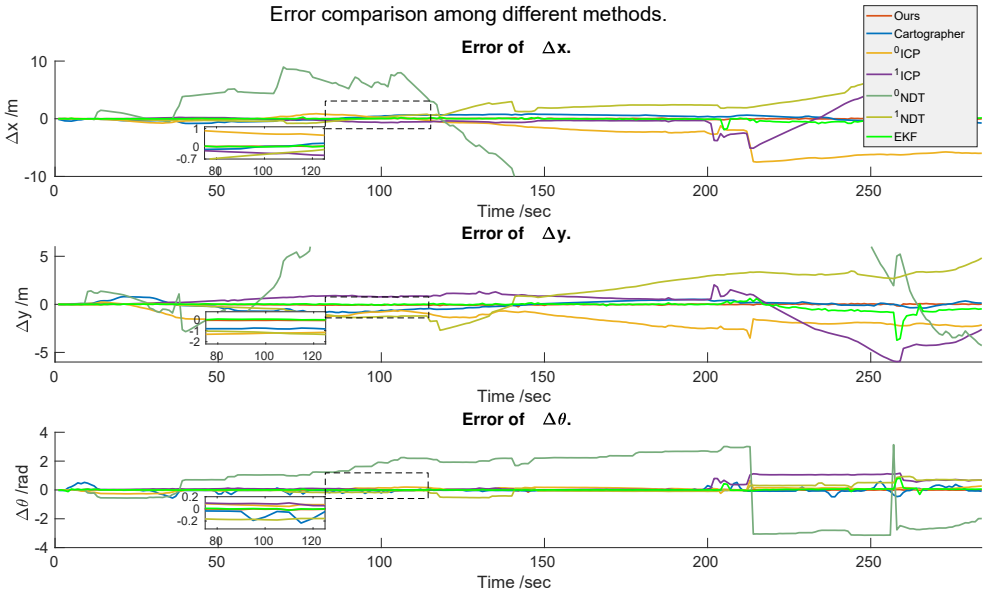


(b) Case 2: Error comparison among different methods

Figure 3.15 : Case 2: Trajectory and error varying with time.



(a) Trajectories comparison among different methods.



(b) Case 3: Error comparison among different methods.

Figure 3.16 : Case 3: Trajectory and error varying with time.

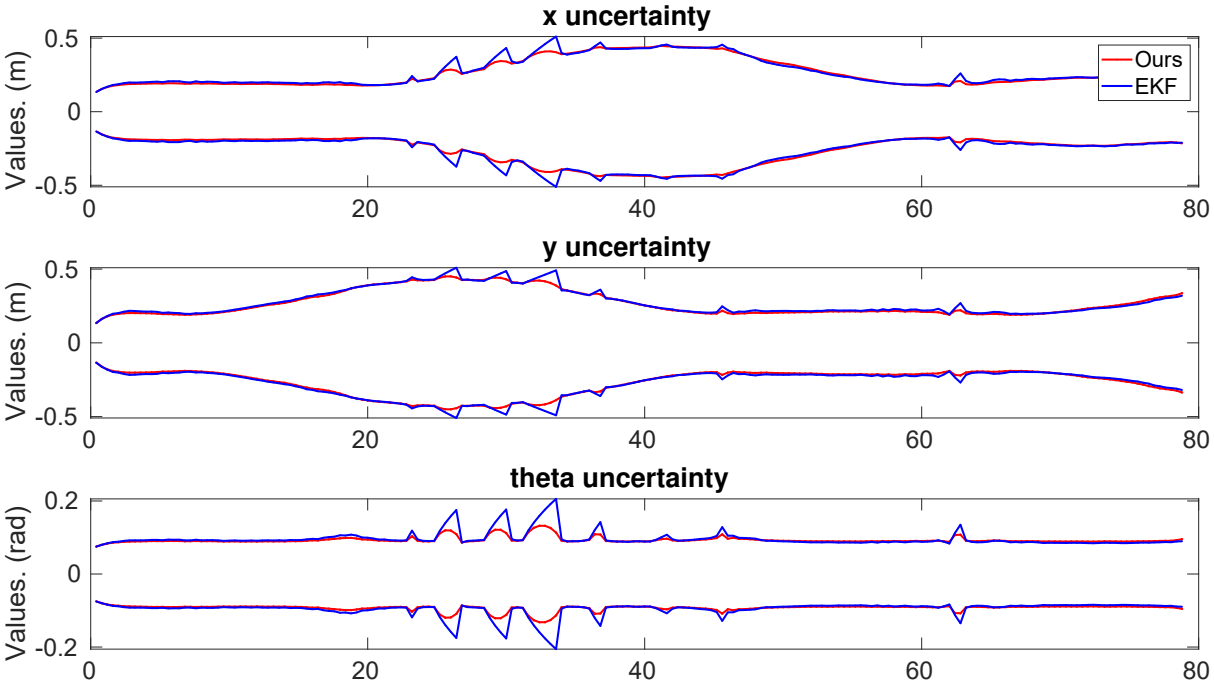
possesses a more accurate result compared with EKF. In order to demonstrate the superiority of factor graph, we depicted the uncertainty comparison between EKF frame and factor graph frame as is shown in Fig. 3.17. Obviously the uncertainty curve of factor graph performs more continuously and smoothly than that of EKF. It is worth noting that although in Case 2 and Case 3 the uncertainty of factor graph exceeds that of EKF and the θ uncertainty of EKF is dramatically small in Case 3 compared with that of factor graph, we still cannot regard that EKF is more accurate than factor graph. The reason has been proved in (Huang and Dissanayake, 2007):

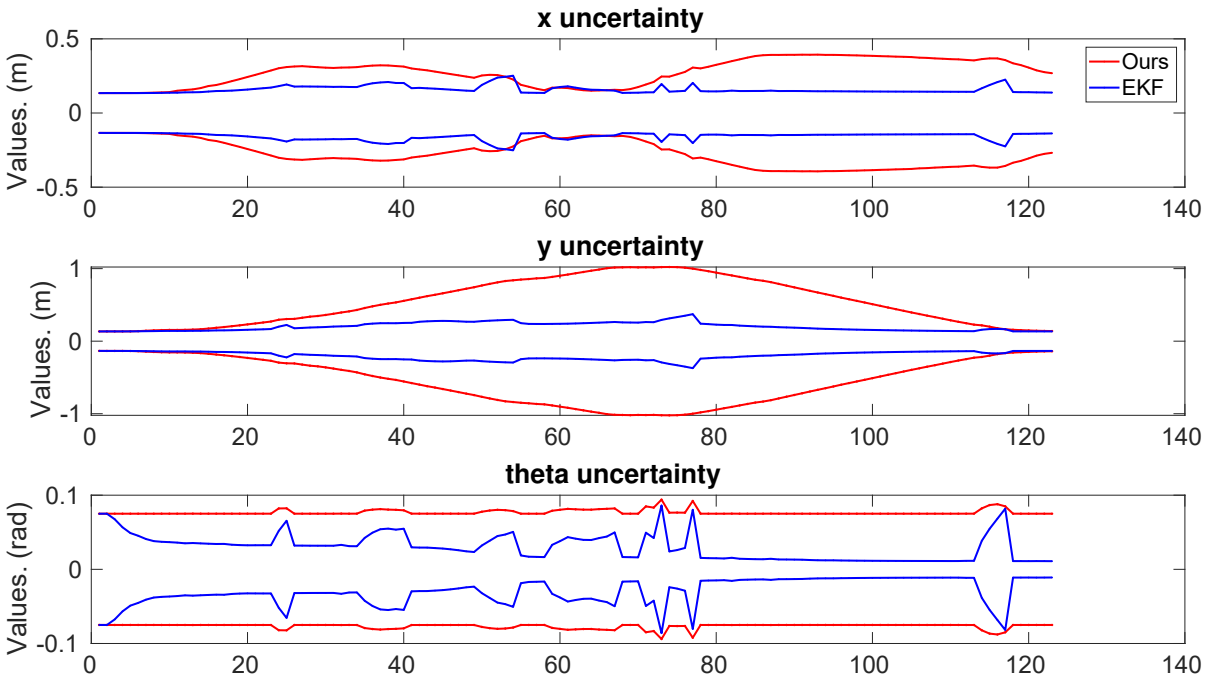
- *The inconsistency of EKF SLAM may cause the variance of the robot orientation estimate to be incorrectly reduced to zero.*
- *The linearization process of EKF SLAM can introduce errors to make the estimated uncertainty smaller than the true uncertainty.*

The error of each pose is depicted as is shown in Fig. 3.17c. It can be found that errors of Pre-fit SLAM are distributed within the uncertainty range, while EKF's errors exceed the corresponding uncertainty greatly even if the uncertainty looks fairly small.

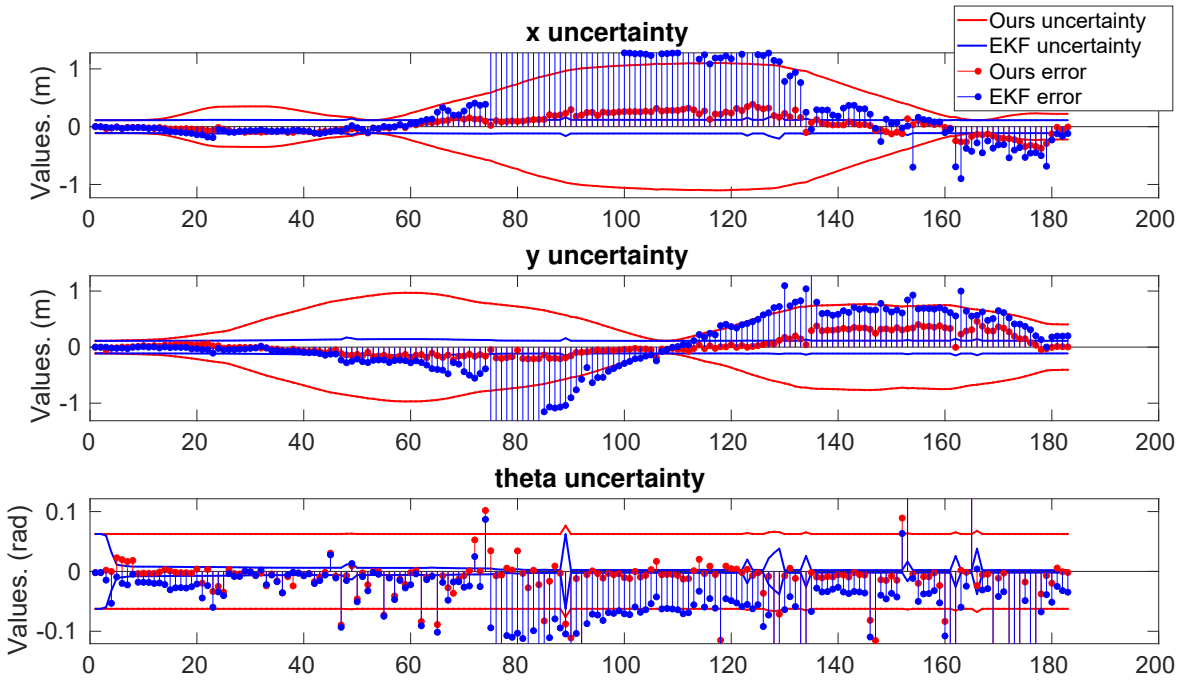
The uncertainty of poses and features in the map is displayed as is shown in Fig. 3.18 (black ellipses are true features). Two conclusions can be drawn: (a) If only one single feature exists, the performance of factor graph and EKF are slightly different, but the difference is indeed small. (b) If there are more than one feature in the environment, factor graph can obtain more accurate estimate compared with EKF especially when loop closure happens.

Table 3.3 provides the RMSE for all the methods with respect to three cases. Table 3.4 lists feature parameters error comparing estimated features obtained by Pre-fit SLAM with groundtruth.





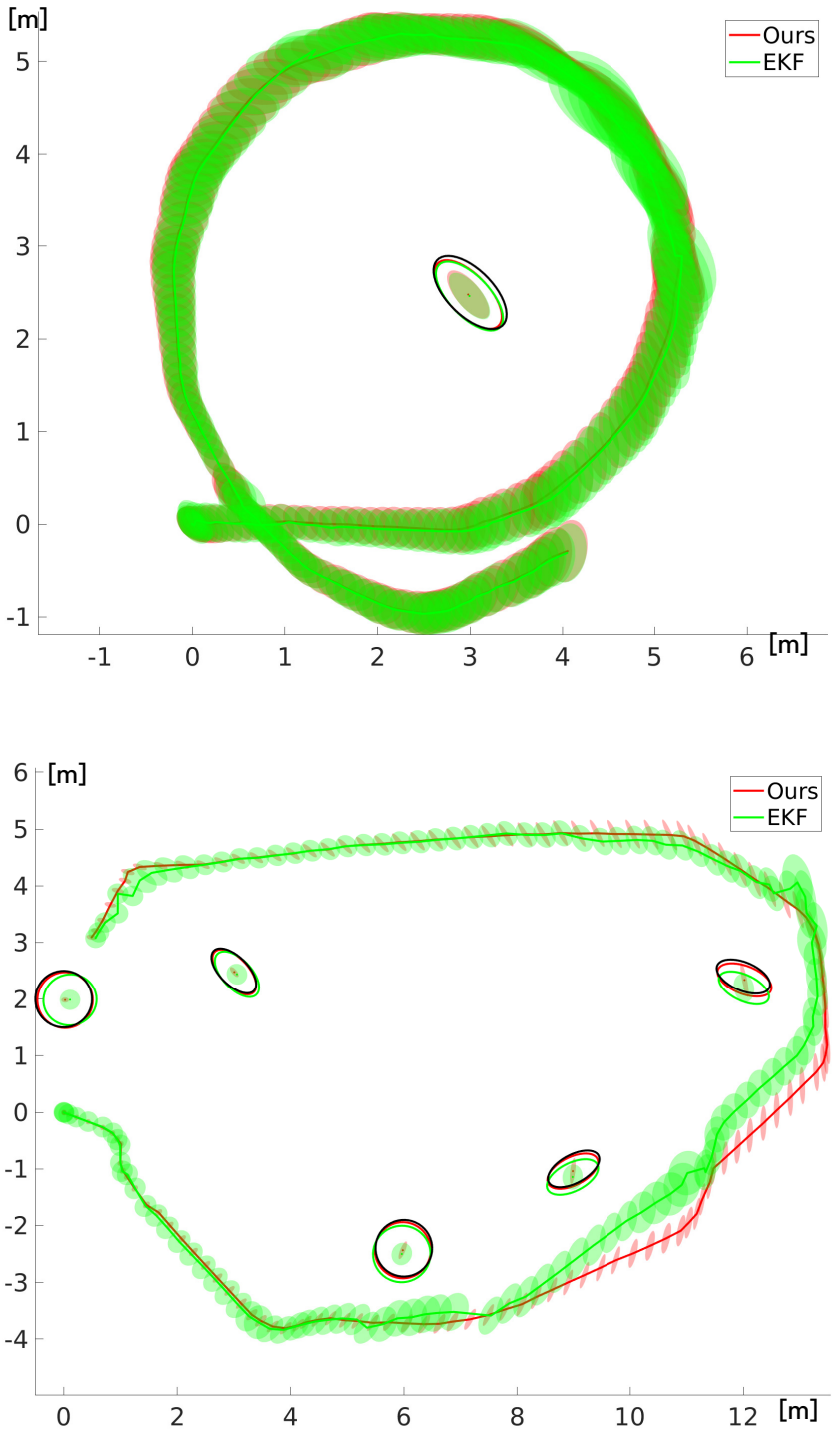
(b) Case 2: Pose uncertainty



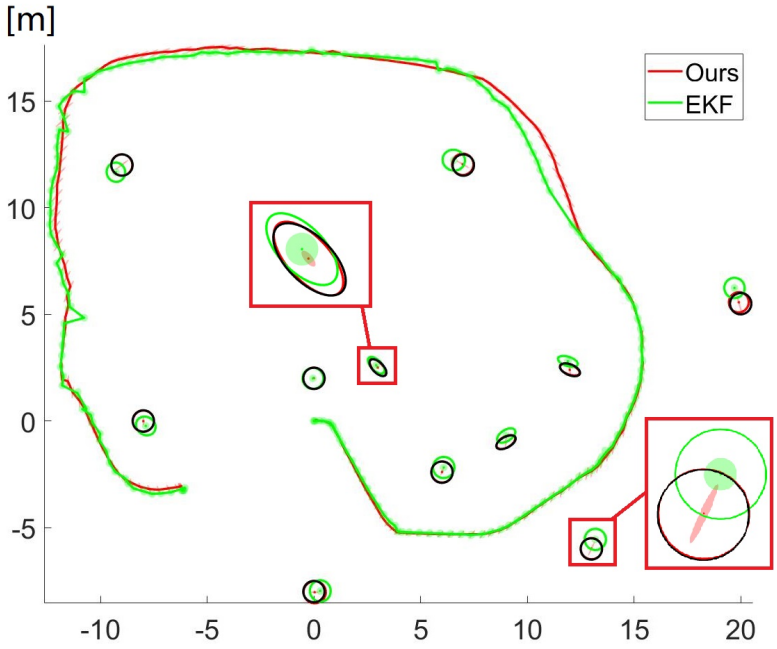
(c) Case 3: Pose uncertainty

Figure 3.17 : Pose 3-sigma bounds comparison between EKF and factor graph. From top to bottom at each sub graph illustrates the uncertainty of x, y and θ.

It can be concluded from Table 3.3 that Pre-fit SLAM possesses the minimal RMSE for three simulated cases except for θ in Case 1 and Case 2, while Cartographer performs better than the other four methods. Nevertheless, the difference between Pre-fit SLAM via factor graph and EKF for Case 1 and Case 2 are quite minor that the difference is no more than 0.007rad. Another conclusion is that NDT method is less adapted to the open environment than ICP method because of sparsely distributed sensor data. Fortunately, a good initial guess for both ICP and NDT can improve the accuracy significantly, but they cannot reach Cartographer's accuracy. By comparing errors in Cartographer and Pre-fit SLAM from Case 1 to Case 3, the accuracy of Pre-fit SLAM enhanced with the increasing feature amounts, while Cartographer is not influenced. This phenomenon is due to the compact graph structure as we associate data before graph optimization, which makes the result



(b) Case 2: Uncertainty of trajectories and features.



(c) Case 3: Uncertainty of trajectories and features.

Figure 3.18 : Uncertainty comparison displayed in the map for each case.

robust and accurate.

3.4.4 Results on actual data

In this section, we conducted a real world environment (Fig. 3.19) with 7 features surrounded by glass walls. Laser data is not reliable hitting transparent glass. The origin position is manually measured as well as features' positions with respect to the predetermined coordinate at the origin point, and the accuracy of measurement is within 0.1m. Because we only have odometry information and features' manually measured position, we did not compare pose errors. Instead, the odometry and measured features' position were used to roughly distinguish the trajectory and evaluate the mapping performance.

Fig. 3.20 depicts trajectories obtained by Pre-fit SLAM, GMapping (**?**), Hector

Table 3.3 : RMSE comparison.(Unit: m)

Dataset	**Case 1**			**Case 2**			**Case 3**		
Method	x/m	y/m	θ/rad	x/m	y/m	θ/rad	x/m	y/m	θ/rad
Ours	**0.10**	**0.10**	0.06	**0.08**	**0.05**	**0.04**	**0.05**	**0.05**	**0.01**
Cartographer	0.30	0.28	0.16	0.32	0.48	0.25	0.49	0.43	0.16
^{0}ICP	2.47	2.15	1.71	7.08	9.87	1.47	3.52	1.62	0.14
^{1}ICP	0.82	0.72	0.42	1.98	3.47	0.61	3.75	1.92	0.49
^{0}NDT	1.80	2.72	2.13	5.21	4.38	1.36	27.92	17.68	2.12
^{1}NDT	0.73	0.65	0.32	7.79	1.98	1.59	4.19	2.22	0.33
EKF	0.11	0.12	**0.05**	0.11	0.29	**0.04**	0.29	0.44	0.08

SLAM (Kohlbrecher et al., 2011a) and Cartographer *. The valid range of Cartographer is limited to 4m in order to keep the fairness of comparison. Both ICP and NDT failed in obtaining an acceptable solution. Real features are plotted in grey shadow via rough measurement, and orange/red features are the estimated result by Pre-fit SLAM. Some estimated features are around 0.5 meter from the actual corresponding features, and the axes dimension of one elliptical feature shrinks. But the trajectory of Cartographer is obviously untrusted because it goes through a feature. The trajectories of GMapping and Hector SLAM present discontinuity. Especially at the end of Hector SLAM, the pose jumped to the middle of the figure because of the wrong scan-matching result. Pre-fit SLAM performs better in this case.

Fig. 3.21a depicts pose uncertainty comparison between Pre-fit SLAM and EKF

*It should be noted that Cartographer failed achieving reasonable result under current valid range and the default configuration. Hence, we tuned weights of Cartographer as well as other parameters to keep the valid range remaining the same. The parameters of GMapping and Hector SLAM were default.

approach. Same phenomenon occurs resembling simulation experiments. Fig. 3.21b shows the "real" features lie in the uncertainty range of ours, while one estimated feature by EKF exceeds the reliable range.

We also evaluated Pre-fit SLAM on public dataset Victoria Park (Guivant et al., 2000) as is shown in Fig. 3.22b. However, cartographer cannot be adopted on this dataset due to the data format difference. Hence, we only compared Pre-fit SLAM with a point feature based approach (Huang et al., 2012) which is marked by blue line. GPS data of the dataset is marked by black dot, and Pre-fit SLAM is expressed by red line. Red ellipses (which look like red points because of the scaled display) are estimated features by Pre-fit SLAM. It is not easy to evaluate the accuracy of trajectory quantitatively since the GPS data is untrustworthy. Also, if looking at the turn on the right it can be seen that Pre-fit SLAM drifts a little compared with point feature SLAM. This is because features at that turn are rare which makes the ellipse fitting process unstable. But the performance is better in the middle part and the left part since features are trustful. Such phenomenons are also reflected to some extent in Fig. 3.22a. The 3-sigma bound of Pre-fit SLAM from step 2000 to 2400 (on the turn right) is broader than that of the point feature based method, but it is narrower around step 5000 and step 6000 (on the middle and left part).

3.4.5 Map performance comparison

In this section we compared the mapping performance between Cartographer and Pre-fit SLAM. Mapping by Cartographer is an occupancy grid map which is widely used in robotics algorithms. However, the accuracy of occupied grid map is affected by the size of the grid. The mapping representation of Pre-fit SLAM directly expresses features with conic equation. The advantages include three aspects: 1. Mapping is continuous so that mapping accuracy won't be influenced by the grid size. 2. The map records features only, where the storage required is less than occupancy

Figure 3.19 : Real world environment.

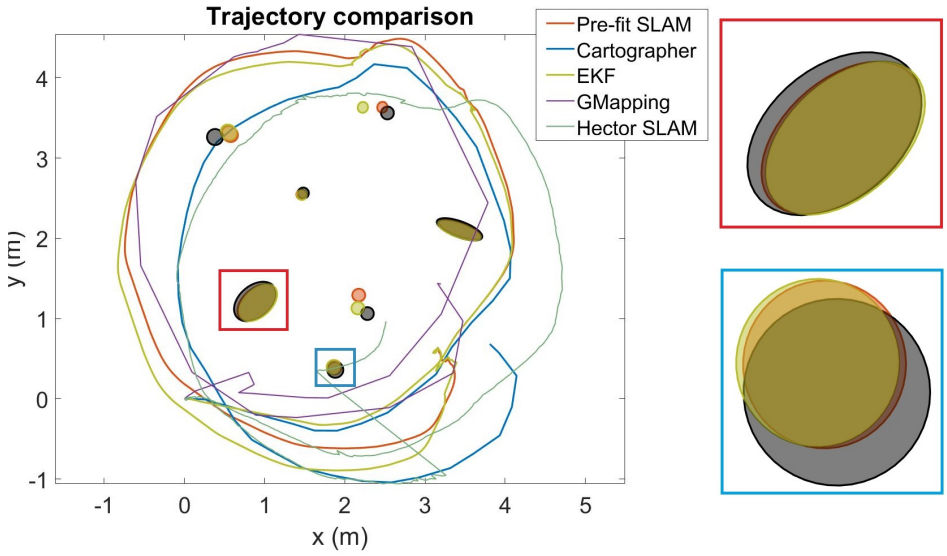
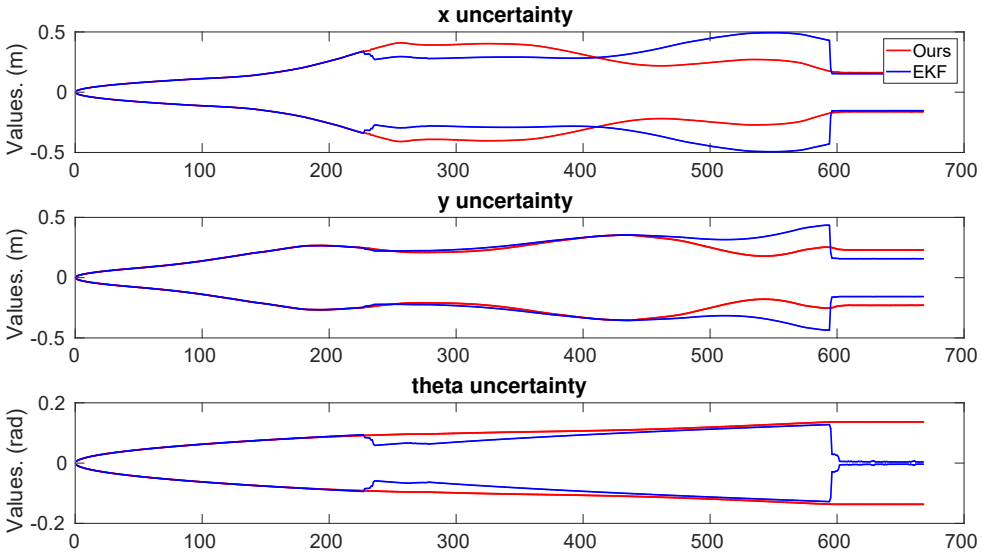


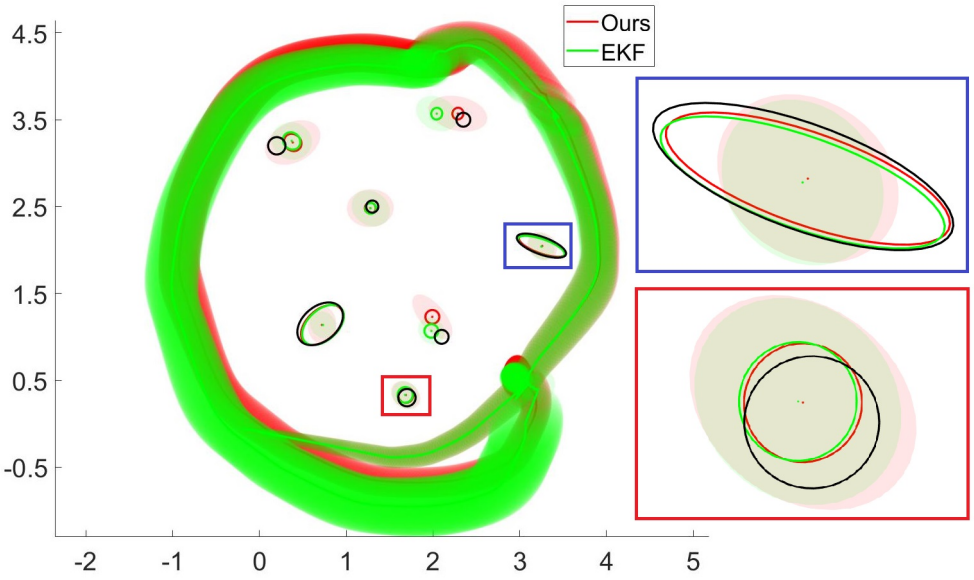
Figure 3.20 : Real world: Trajectories comparison among different methods.

grid map. 3. The representation is human friendly and easy for visualization.

Fig. 3.23 compares mapping performance of Cartographer and Pre-fit SLAM. The first row from Fig. 3.23a to Fig. 3.23c is obtained by Cartographer and the



(a) Pose uncertainty.



(b) Uncertainty comparison displayed in the map.

Figure 3.21 : Uncertainty comparison between factor graph and EKF.

second row is by Pre-fit SLAM. The map of practical experiment was also obtained and compared by GMapping and Hector SLAM, as shown in Fig. 3.23d. It is clear that Pre-fit SLAM can obtain a good map. In the first row of Case 1, Case 2, and Real Case, it is easy to find abundant duplicated and overlapping curves which

should be assembled to the same feature. But Cartographer did well for Case 3 because peripheral features are observed on one side which will not introduce too much error into the optimization problem.

We also compared the estimated features with groundtruth (Groundtruth: Case 1 to Case 3. Rough measurement: Real experiment) to check the accuracy numerically and to make figures friendly to read we filled real features and estimated results with black and orange shadow respectively in the trajectories comparison figures (Fig. 3.14a, Fig. 3.15a, Fig. 3.16a, and Fig. 3.20). As is shown in Table 3.4, for simulation experiments, the accuracy of majority estimated features is around 2 centimeters except a few large error terms. But even these large terms are within 10 centimeters. However, real experiment performs worse than simulations, the largest error term reaches 20 centimeters (As shown in Fig. 3.20, two estimated features are far away from the measured true features). Also, mapping of the Victoria Park by Cartographer is not built due to the capability limitation, while Pre-fit SLAM built the map shown in Fig. 3.22b. Nevertheless, Pre-fit SLAM is able to provide a continuous, well-performed and robust map in contrast to Cartographer.

3.5 Summary

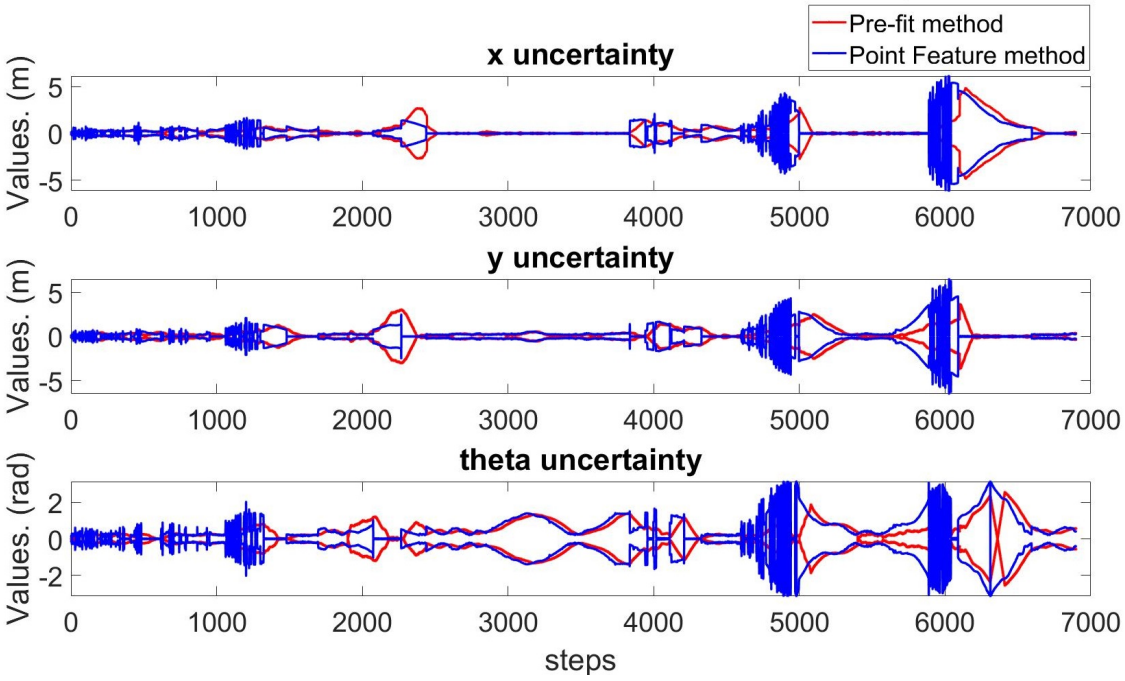
A pre-fit feature based SLAM algorithm in open environment via 2D lidar was proposed in this chapter. Traditional scan matching methods are not competent for working in an open environment where sufficient edges and corners do not exist. We proposed a conic feature based method to represent features and reformed corresponding graph structure instead of matching scan points in a traditional approach. First, the raw data was processed with the prior information of odometry to associate data. Then conic feature fitting was applied to transform points to the feature parameterization proposed in this chapter. At last a factor graph optimization was adopted to obtain pose estimates as well as the map in our representation. Simula-

Table 3.4 : Features absolute error.

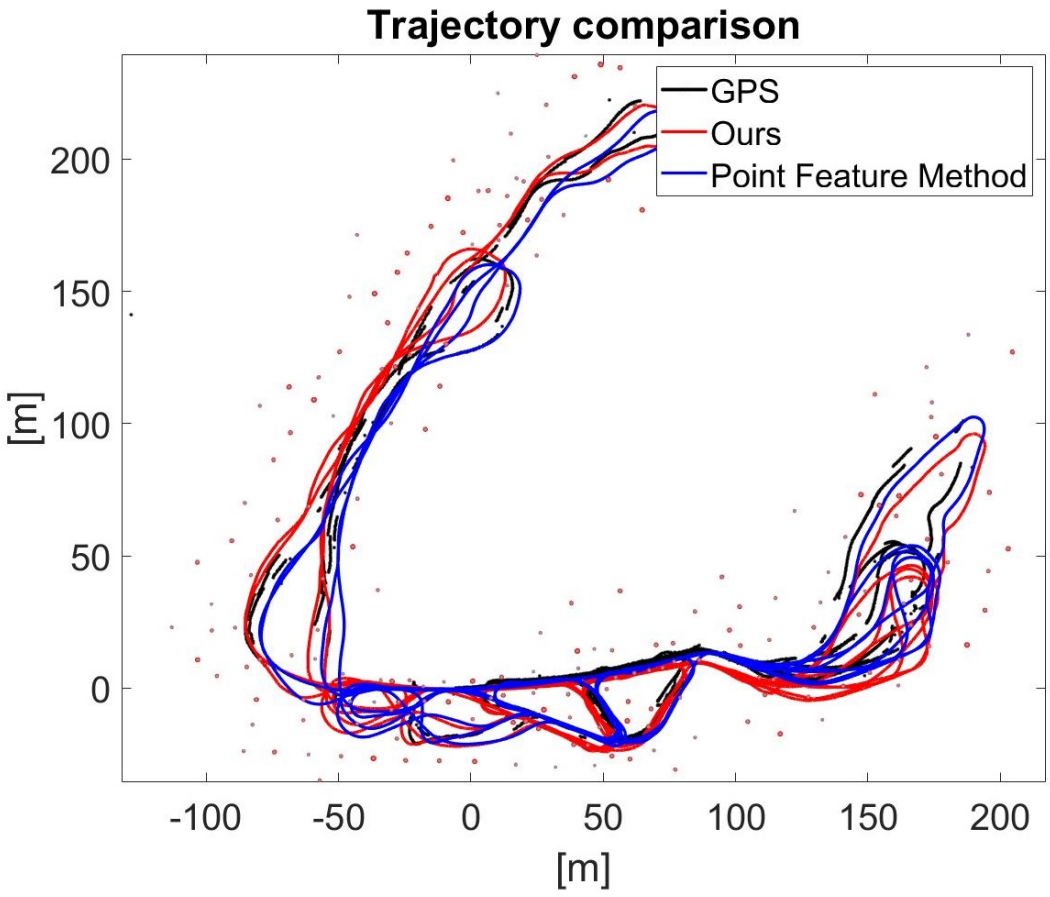
		ΔF_x /m	ΔF_y /m	ΔF_ϕ /rad	ΔF_{r_1} /m	ΔF_{r_2} /m
Case 1	F_1	0.0019	0.0036	0.0123	0.0060	0.0055
	F_1	0.0121	0.0148	0.0176	0.0018	0.0032
	F_2	0.0155	0.0356	-	0.0011	0.0002
Case 2	F_3	0.0127	0.0470	0.0045	0.0095	0.0025
	F_4	0.0102	0.0871	0.0228	0.0075	0.0189
	F_5	0.0283	0.0123	-	0.0029	0.0101
	F_1	0.0039	0.0284	0.0121	0.0021	0.0011
	F_2	0.0066	0.0137	-	0.0011	0.0017
	F_3	0.0426	0.0448	-	0.0043	0.0011
	F_4	0.0120	0.0187	0.0440	0.0045	0.0020
	F_5	0.0024	0.0149	-	0.0003	0.0051
Case 3	F_6	0.0293	0.0216	-	0.0146	0.0117
	F_7	0.0158	0.0276	0.0316	0.0139	0.0049
	F_8	0.0017	0.0310	-	0.0028	0.0061
	F_9	0.0684	0.0887	-	0.0079	0.0124
	F_{10}	0.0028	0.0048	-	0.0062	0.0015
	F_{11}	0.0073	0.0016	-	0.0003	0.0002
	F_1	0.1074	0.2297	-	0.0010	0.000
	F_2	0.0629	0.0708	-	0.0121	0.0169
	F_3	0.1838	0.0381	-	0.0062	0.0065
Real	F_4	0.0223	0.0180	-	0.0070	0.0074
	F_5	0.0210	0.0125	-	0.0150	0.0259
	F_6	0.0132	0.0297	-	0.0102	0.0127
	F_7	0.0089	0.0039	0.0107	0.0168	0.0128

tion experiments and real environment test demonstrated that our proposed SLAM algorithm can get accurate and convincing results for the open environment and the map in our representation can accurately express the environment situation.

During the experiments we found that the results will be greatly affected if the features are not well fitted. Proper outlier elimination approaches were adapted in this chapter, otherwise the estimated poses will deviate from the groundtruth. Starting from the mentioned concern, we proceeded to the work in next chapter.

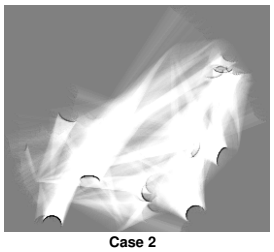


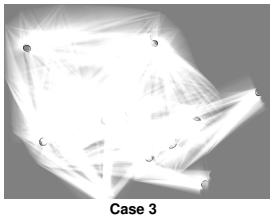
(a) Pose uncertainty.



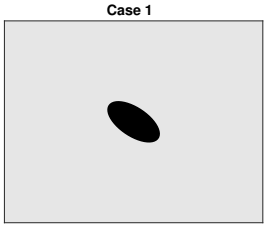
(b) Trajectory and features on Victoria Park.

Figure 3.22 : Evaluated on Victoria Park.

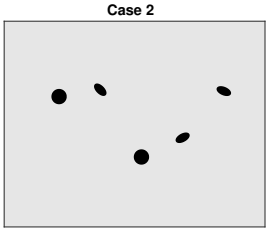




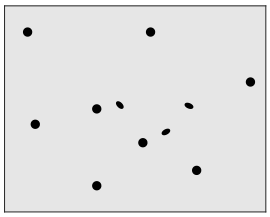
Case 1



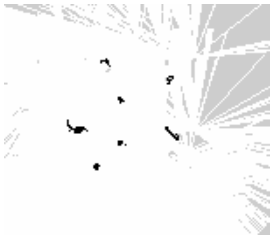
Case 2

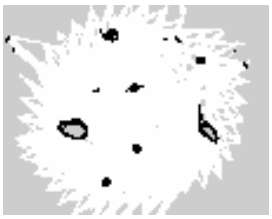


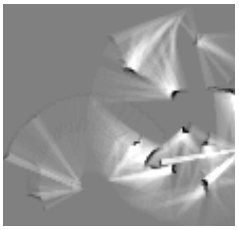
Case 3



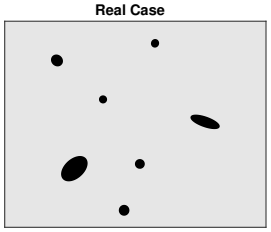
(a) Case 1. (b) Case 2. (c) Case 3.







Real Case



(d) Real world. From left to right are results of GMapping, Hector SLAM, Cartographer, and Pre-fit Method.

Figure 3.23 : Maps by different methods. (a) Case 1. Up: Cartographer; Down: Pre-fit SLAM. (b) Case 2. Up: Cartographer; Down: Pre-fit SLAM. (c) Case 3. Up: Cartographer; Down: Pre-fit SLAM. (d) Real world.

Chapter 4

A post-count feature based SLAM approach on implicit functions: Implicit-SLAM

Chapter 3 has proposed a SLAM algorithm based on pre-fitting features from original observations. This chapter continues on the basis of feature-based SLAM, and conducts more in-depth theoretical mining for general features. In contrast to Pre-fit SLAM, this chapter proposes a potential framework which can be adopted for all the types of features represented by implicit functions. To illustrate the new proposed SLAM technique, ellipse and line features are used as two examples to demonstrate how the proposed problem can be solved by iterative methods. The performance comparison between the new technique in this chapter and the Pre-fit SLAM clearly shows the advantages of the proposed approach.

The main contributions of this chapter are:

- We propose a framework called Implicit-SLAM to solve SLAM problem with implicit functions by clearly formulating the problem with implicit functions to represent features, computing corresponding implicit covariance rigorously, and presenting a framework for solving the problem using iterative methods.
- By taking ellipse and line features as examples, we compared the proposed method with Pre-fit SLAM and proved the superiority of Implicit-SLAM.
- We develop a novel logarithmic form objective function for features with closed shapes to enhance the convergence of the iteration based algorithms.

4.1 Motivation

Feature based approach estimates the parameters of the feature present in the environment. One basic feature based SLAM is point feature SLAM where the feature parameter is the position of the point feature. Other features used in laser SLAM include line feature, ellipse feature, curve feature and so on.

However, most of the existing feature based SLAM methods including the proposed method in Chapter 3 need to pre-fit features before estimating robot pose, which are limited in the following aspects:

- The process of pre-fitting features is an approximation via geometrical or mathematical properties, which may abandon valid environmental information.
- Badly-fitted observations can drive the estimated result into errors if outliers are not eliminated appropriately.
- Most sensors observe the object from one side, which makes fitting features difficult to predict, because the specific shape of the other side of the object is not known. This property makes pre-fitting features only handle limited types of geometric features.

In reality, laser scans reflect the boundary of an object, which could be of arbitrary shape and cannot be easily described with feature parameters. On the other hand, most of the boundaries can be expressed by implicit functions (every point on the boundary satisfies the function, referred to Section 4.2.2). Thus we would like to ask the question: Is it possible to use implicit functions as features in SLAM?

It should be noted that implicit functions cover general geometric features as special cases. As is shown in Fig. 4.1, features like circle, ellipse, diamond, or even irregular closed curve can be expressed by implicit functions. Thus, SLAM with

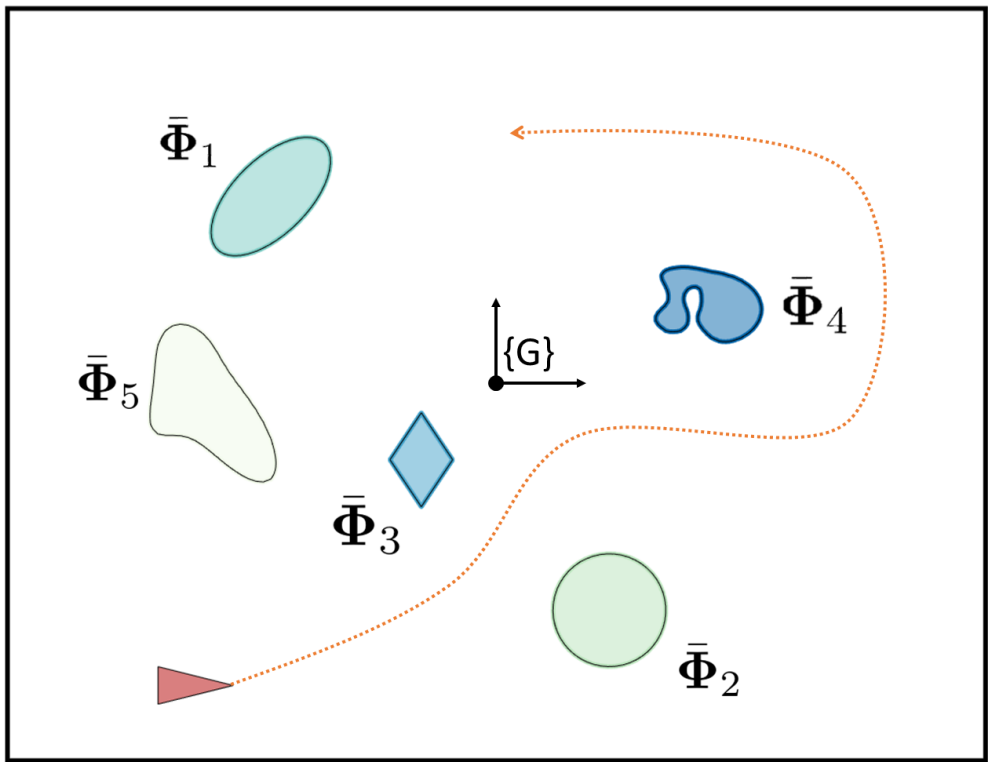


Figure 4.1 : Schematic diagram of Implicit-SLAM. The red triangle is the robot starting at [-8, -8]$^{\intercal}$. Features with implicit functions are defined by:

$$\bar{\mathbf{\Phi}}_1 \overset{\text{def.}}{=} 0.6\mathbf{x}^2 - 0.8\mathbf{xy} + 0.6\mathbf{y}^2 + 11.2\mathbf{x} - 10.8\mathbf{y} + 1400.4$$

$$\bar{\mathbf{\Phi}}_2 \overset{\text{def.}}{=} \mathbf{x}^2 - 6\mathbf{x} + \mathbf{y}^2 + 12\mathbf{y} + 42.74$$

$$\bar{\mathbf{\Phi}}_3 \overset{\text{def.}}{=} 1.2|\mathbf{x} + 2| + 0.8|\mathbf{y} + 2| - 1$$

$$\bar{\mathbf{\Phi}}_4 \overset{\text{def.}}{=} 3(1 - 5(\mathbf{x} - 5))^2 e^{-(5(\mathbf{x}-5)^2)-(5(\mathbf{y}-2)+1)^2} - 10((\mathbf{x} - 5) - 5(\mathbf{x} - 5)^3 - 3(\mathbf{y} - 2)^5)e^{-2(\mathbf{x}-5)^2-5(\mathbf{y}-2)^2} - \frac{1}{3}e^{-(5(\mathbf{x}-5)+1)^2-4(\mathbf{y}-2)^2} - 0.1$$

$$\bar{\mathbf{\Phi}}_5 \overset{\text{def.}}{=} 1.2\mathbf{x}^4 + 0.4\mathbf{y}^4 + 2\mathbf{xy} + 2.3\mathbf{x}^3\mathbf{y} - 2$$

Noted that $\mathbf{x}$ and $\mathbf{y}$ are points belonging to each feature in global frame.

implicit functions as features is a very general feature based SLAM and has the potential to be applied in different scenarios (3D surfaces are implicit functions in 3D).

This chapter studies the 2D laser SLAM problem utilizing implicit functions as features. To the best of our knowledge, no clear researches are made on formulating features as implicit functions. We clearly formulate the problem as an optimization problem, and (a) correctly compute the observation covariance matrix, (b) formulate a symmetry energy term for closed shapes. Then, we propose a potential framework which can be adopted for all the types of features represented by implicit functions.

4.2 Problem formulation

In this section, we formulate a general SLAM problem with features represented by implicit objective functions and elaborate the feasibility and approaches of solving such a problem. Consider a general feature-based SLAM problem (see Fig. 4.1). Assume a robot moves n steps in the scenario containing features $\mathbf{\Phi}_1, \mathbf{\Phi}_2, \cdots, \mathbf{\Phi}_q$. At each step, the robot collects laser points hitting on features. The state is defined by:

$$\mathbf{\Psi} \overset{\text{def.}}{=} \begin{Bmatrix} \mathbf{\Xi}_1 & \mathbf{\Xi}_2 & \cdots & \mathbf{\Xi}_n \\ \mathbf{\Phi}_1 & \cdots & \mathbf{\Phi}_q & \end{Bmatrix} \overset{\text{def.}}{=} < \mathbf{\Xi}, \mathbf{\Phi} > \tag{4.1}$$

under the assumption that initial pose $\mathbf{\Xi}_0 = [0; 0; 0]$. Thus, the problem is minimizing the energy function:

$$\underset{\mathbf{\Psi}}{\operatorname{argmin}}\ E_{\text{total}} = E_{\text{odom}} + \underbrace{\sum_{j=1}^{q} E_{\text{feature},j}}_{E_{\text{feature}}} \tag{4.2}$$

and each term in Eq. (4.2) is defined by:

$$\begin{aligned} E_{\text{odom}} &= \frac{1}{2} \sum_{i=1}^{n} \| f(\mathbf{Z}_{\text{odom},i}, \mathbf{\Xi}_{i-1}, \mathbf{\Xi}_i) \|^2_{\Sigma_{\text{odom},i}} \\ E_{\text{feature},j} &\overset{\text{def.}}{=} \frac{1}{2} \sum_{i=1}^{n} \| g(\mathbf{Z}_{\text{feature},i,j}, \mathbf{\Xi}_i, \mathbf{\Phi}_j) \|^2_{\Sigma_{\text{feature},i,j}} \end{aligned} \tag{4.3}$$

where $\mathbf{Z}_{\text{odom},i}$ is the observation vector of i^{th} odometry, $\mathbf{Z}_{\text{feature},i,j}$ is the observation vector of feature j at pose i.* $f(\mathbf{Z}_{\text{odom},i}, \mathbf{\Xi}_{i-1}, \mathbf{\Xi}_i)$ and $g(\mathbf{Z}_{\text{feature},i,j}, \mathbf{\Xi}_i, \mathbf{\Phi}_j)$ are the cost functions for the two entries, respectively. $\Sigma_{\text{odom},i}$ is the odometry covariance at the i^{th} step. $\Sigma_{\text{feature},i,j}$ is the covariance of feature j's observation from pose i.

The energy term of a typical odometry observation (measuring relative pose) is:

$$f(\mathbf{Z}_{\text{odom},i}, \mathbf{\Xi}_{i-1}, \mathbf{\Xi}_i) = \mathbf{Z}_{\text{odom},i} - \begin{bmatrix} R_{i-1}^{\intercal}(\mathbf{t}_i - \mathbf{t}_{i-1}) \\ \text{dist}(\phi_i - \phi_{i-1}) \end{bmatrix}_{3\times 1} \tag{4.4}$$

where $\text{dist}(\phi_i - \phi_{i-1})$ is the angle distance between the i^{th} and the $(i-1)^{th}$ robot orientation, and $\mathbf{Z}_{\text{odom},i}$ is the odometry observation at pose i in the form of $[\Delta x_i, \Delta y_i, \Delta\phi_i]^{\intercal}$. In this chapter, geodesic distance is used to find the difference of angles, which is also known as "*wrap*".

4.2.1 Problem formulation of Pre-fit SLAM

In Pre-fit SLAM introduced in Chapter 3, the observation $\tilde{\mathbf{Z}}_{\text{feature},i,j}$ is obtained by fitting the raw data following a certain parameterization, and the covariance $\tilde{\Sigma}_{\text{feature},i,j}$ is from the fitting result. Hence, the optimization problem aims to minimize the energy cost $\tilde{E}_{\text{feature},j}$ between actual observation $\tilde{\mathbf{Z}}_{\text{feature},i,j}$ and theoretical observation $T^{-1}(\mathbf{\Xi}, \mathbf{\Phi})$. To obtain the theoretical observations, we need to transform feature states from global frame to the corresponding local frame. In order to distinguish annotations from other methods, we use $\mathbf{F}$ to denote features. Taking feature j as an example, the energy cost can be expressed as:

$$\tilde{E}_{\text{feature},j} = \frac{1}{2}\sum_{i=1}^{n} \|\tilde{\mathbf{Z}}_{\text{feature},i,j} - T^{-1}(\mathbf{\Xi}_i, \mathbf{F}_j)\|^2_{\tilde{\Sigma}_{\text{feature},i,j}}. \tag{4.5}$$

*Without loss of generality, we assume feature j is observed from all the poses 1 to n. For different SLAM formulations, the format of $\mathbf{Z}_{\text{feature},i,j}$ is different, as seen in Sections 4.5.3 and 4.5.4 (Pre-fit SLAM), Sections 4.5.1 and 4.5.2 (Implicit-SLAM).

It is worth noting that $\mathbf{F}_j$ depends on how the feature is parameterized, instead of only in the point form. And $\tilde{\mathbf{Z}}_{\text{feature},i,j}$ is always in the same format of the feature state. It is easy to find $\tilde{\Sigma}_{\text{feature},i,j}$ if features can be observed directly or fitted by raw data in advance.

4.2.2 Problem formulation of Implicit-SLAM

Suppose an implicit function $\bar{\mathbf{\Phi}}_j(\mathbf{P}) = \mathbf{0}$ holds[†] for a point set $\mathbf{P}$ that belongs to feature $\mathbf{\Phi}_j$ and $\mathbf{P}$ is in the global coordinate as well as $\mathbf{\Phi}_j$. As shown in Fig. 4.1, for example, feature $\mathbf{\Phi}_1$ to $\mathbf{\Phi}_5$ are in complex shapes, and the corresponding implicit functions $\bar{\mathbf{\Phi}}_1 = \mathbf{0}$ to $\bar{\mathbf{\Phi}}_5 = \mathbf{0}$ hold for every point locating on each feature respectively.

If the perfect observation of $\mathbf{\Phi}_j$ at pose i is a point set denoted by ${}^{\{i\}}\mathbf{P}$, it must satisfy the implicit function after transforming it to global frame, that is:

$$\bar{\mathbf{\Phi}}_j({}^{\{G\}}\mathbf{P}) = \bar{\mathbf{\Phi}}_j(T^{-1}(\mathbf{\Xi}_i, {}^{\{i\}}\mathbf{P})) = \mathbf{0}. \tag{4.6}$$

Eq. (4.6) is obviously an implicit function of $\mathbf{\Xi}_i$,$\mathbf{\Phi}_j$ and ${}^{\{i\}}\mathbf{P}$. Since observations always contain noises, still taking feature j as an instance, the energy term of feature j turns to be:

$$E_{\text{feature},j} = \frac{1}{2}\sum_{i=1}^{n} \left\| \bar{\mathbf{\Phi}}_j \left(T^{-1}(\mathbf{\Xi}_i, \mathbf{Z}_{\text{feature},i,j}) \right) \right\|^2_{\Sigma_{\mathbf{\Phi}_j,i}} \tag{4.7}$$

where $\mathbf{Z}_{\text{feature},i,j}$ are raw points belonging to feature j at pose i, and $\Sigma_{\mathbf{\Phi}_j,i}$ is the corresponding covariance, which will be computed in the next section. The method solving SLAM problem with implicit functions is called post-count method in this thesis. To denote with other post-count approaches, the algorithm in this chapter is denoted as Implicit-SLAM.

The comparison of pre-fit and Implicit-SLAM method is shown in Tab. 4.1.

[†]We use $\mathbf{\Phi}_j$ to represent the feature j and use $\bar{\mathbf{\Phi}}_j$ to represent the feature's implicit function.

Table 4.1 : Comparison of Pre-fit SLAM and Implicit-SLAM

	Pre-fit SLAM	**Implicit-SLAM**
Observation	Depending on feature parameterization	Raw points
Annotation	$\tilde{\mathbf{Z}}_{\text{feature},i,j}$	$\mathbf{Z}_{\text{feature},i,j}$
Properties	Need parameterization† $\mathbf{F}_j$	Need Implicit function‡ $\bar{\mathbf{\Phi}}_j(\mathbf{p})$
Objective function†‡	$\frac{1}{2}\sum_{i=1}^{n} \|\tilde{\mathbf{Z}}_{\text{feature},i,j} - T^{-1}(\mathbf{\Xi}_i, \mathbf{F}_j)\|^2_{\tilde{\Sigma}_{\text{feature},i,j}}$	$\frac{1}{2}\sum_{i=1}^{n} \left\|\bar{\mathbf{\Phi}}_j\left(T^{-1}(\mathbf{\Xi}_i, \mathbf{Z}_{\text{feature},i,j})\right)\right\|^2_{\Sigma_{\mathbf{\Phi}_j,i}}$
Variables		
Frame	From $\{G\}$ to $\{L\}$	From $\{L\}$ to $\{G\}$
Covariance	$\Sigma_{\text{feature},i,j}$	$\Sigma_{\mathbf{\Phi}_j,i}$
Dependence	Depend on feature's fitting; fixed	Depend on feature's implicit function; vary in each iteration step
Information loss	Accumulate with time and poses	No accumulation

† Pre-fit SLAM: different feature-based SLAM methods have unequal parameterization way. Implicit-SLAM: Each kind of features possesses a unique implicit function which holds for all the points belonging to the feature.

‡ Odometry part is omitted in the table.

4.3 Solution to Implicit-SLAM

4.3.1 Uncertainty transmission and implicit covariance

In typical least squares problems, the energy term is $E = \|\mathbf{z} - f(\mathbf{x})\|^2_\Sigma$ and Σ is the covariance of the noise in $\mathbf{z}$. However, the covariance of implicit functions cannot be obtained directly from observations. Thus the following lemma is proposed to link the raw observation and the implicit items and calculate the covariance $\Sigma_{\mathbf{\Phi}_j,i}$ in Eq. (4.7).

Lemma 1: Consider a least squares problem with energy term $E = \|f(\mathbf{x}, \mathbf{z})\|^2_{\Sigma_f}$ with variables $\mathbf{x}$ and observations $\mathbf{z}$. Assume $\mathbf{z} = \mathbf{z}_0 + \mathbf{n}_\mathbf{z}$, where $\mathbf{n}_\mathbf{z} \sim N(\mathbf{0}, \Sigma_z)$ is a zero-mean Gaussian noise. Since $f(\mathbf{x}_0, \mathbf{z})$ approximately follows Gaussian distribution around $\mathbf{z} = \mathbf{z}_0$ as:

$$f(\mathbf{x}_0, \mathbf{z}) \sim N(f(\mathbf{x}_0, \mathbf{z}_0), J_\mathbf{z} \Sigma_z J_\mathbf{z}^\intercal) \tag{4.8}$$

where

$$J_\mathbf{z} = \left.\frac{\partial f}{\partial \mathbf{z}}\right|_{\mathbf{z}=\mathbf{z}_0} \tag{4.9}$$

a good choice of Σ_f in the least squares problem is

$$\Sigma_f = J_\mathbf{z} \Sigma_z J_\mathbf{z}^\intercal \tag{4.10}$$

Proof 1 (Lemma 1): Let $f(\mathbf{x}_0, \mathbf{z}_0) = f_0$ and $J_\mathbf{x} = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{z}=\mathbf{z}_0, \mathbf{x}=\mathbf{x}_0}$. Expand $f(\mathbf{x}, \mathbf{z})$ around $\mathbf{z}_0$ and $\mathbf{x}_0$:

$$f(\mathbf{x}, \mathbf{z}) \approx f_0 + J_\mathbf{z}(\mathbf{z} - \mathbf{z}_0) + J_\mathbf{x}(\mathbf{x} - \mathbf{x}_0) \tag{4.11}$$

and $f(\mathbf{x}_0, \mathbf{z})$ around $\mathbf{z}_0$ is:

$$f(\mathbf{x}_0, \mathbf{z}) \approx f_0 + J_\mathbf{z}(\mathbf{z} - \mathbf{z}_0) = \mathbf{Z}. \tag{4.12}$$

Since $\mathbf{z}$ has zero-mean Gaussian noise $\mathbf{n}_\mathbf{z}$, the probability distribution of $\mathbf{Z}$ yields:

$$\mathbf{Z} \sim N(f_0, J_\mathbf{z} \Sigma_z J_\mathbf{z}^\intercal)) \tag{4.13}$$

Hence, $f(\mathbf{x}_0, \mathbf{z})$ approximately follows:

$$f(\mathbf{x}_0, \mathbf{z}) \sim N(f_0, J_{\mathbf{z}} \Sigma_z J_{\mathbf{z}}^{\intercal}) \tag{4.14}$$

Then the minimizing problem can be rewriten by:

$$\begin{aligned} \underset{\mathbf{x}}{\operatorname{argmin}}\ F &\approx \|f_0 + J_{\mathbf{z}}(\mathbf{z} - \mathbf{z}_0) + J_{\mathbf{x}}(\mathbf{x} - \mathbf{x}_0)\|_{\Sigma_f}^2 \\ &= \|\mathbf{Z} - A\Delta\mathbf{x}\|_{\Sigma_f}^2 \end{aligned} \tag{4.15}$$

where

$$A = -J_{\mathbf{x}}, \quad \Delta\mathbf{x} = \mathbf{x} - \mathbf{x}_0 \tag{4.16}$$

Thus a good choice of Σ_f is $\Sigma_f = J_{\mathbf{z}} \Sigma_z J_{\mathbf{z}}^{\intercal}$.

It is impossible to obtain $\mathbf{z}_0$ during practical experiments, which is the groundtruth of $\mathbf{z}$. Because the noise influence of observed points is similar to that of groundtruth points when the observations are near the exact positions, we use the observed points to approximately calculate the covariance in Eq. (4.7). The experiment in Section 4.6.2 shows the validity.

4.3.2 Optimization of Implicit-SLAM

One important difference between the new SLAM problem and Pre-fit SLAM is: Feature $\mathbf{\Phi}_j$ in Eq. (4.1) is expressed by an implicit function instead of a finite-dimensional vector. Thus $\mathbf{\Psi}$ in Eq. (4.1) is not a typical state vector and the problem cannot be directly solved using iterative methods.

However, if we can identify some "changeable parameters" in each feature $\mathbf{\Phi}_j$, then the problem is to find these changeable parameters together with the poses such that the total energy is minimized.

Suppose the "changeable parameters" in feature $\mathbf{\Phi}_j$ are defined by s elements in a vector form $\overrightarrow{\mathbf{\Phi}_j} = [\Phi_{j_1}, \Phi_{j_2}, \cdots, \Phi_{j_s}]^{\intercal}$, then standard iterative methods such as Gauss-Newton and Levenberg-Marquardt can be used to solve the problem.

Suppose the incremental Δ is defined by:

$$\begin{aligned}\Delta &\overset{\text{def.}}{=} \begin{Bmatrix} \Delta\boldsymbol{\Xi}_1 & \Delta\boldsymbol{\Xi}_2 & \cdots & \Delta\boldsymbol{\Xi}_n \\ \Delta\overrightarrow{\boldsymbol{\Phi}}_1 & \cdots & \Delta\overrightarrow{\boldsymbol{\Phi}}_q & \end{Bmatrix} \\ &\overset{\text{def.}}{=} < \Delta\boldsymbol{\Xi}, \Delta\overrightarrow{\boldsymbol{\Phi}} > .\end{aligned} \tag{4.17}$$

Since we need to find Δ, an $\oplus$ operator is defined to apply the increment Δ to $\boldsymbol{\Psi}_{\text{old}}$ as:

$$\begin{aligned}\boldsymbol{\Psi}_{\text{new}} &\overset{\text{def.}}{=} \boldsymbol{\Psi}_{\text{old}} \oplus \Delta \\ &\overset{\text{def.}}{=} \begin{Bmatrix} \boldsymbol{\Xi}_1 \oplus \Delta\boldsymbol{\Xi}_1 & \cdots & \boldsymbol{\Xi}_n \oplus \Delta\boldsymbol{\Xi}_n \\ \boldsymbol{\Phi}_1 \oplus \overrightarrow{\boldsymbol{\Phi}}_1 & \cdots & \boldsymbol{\Phi}_q \oplus \overrightarrow{\boldsymbol{\Phi}}_q \end{Bmatrix} \\ &\overset{\text{def.}}{=} < \boldsymbol{\Xi}_{\text{old}} \oplus \Delta\boldsymbol{\Xi}, \boldsymbol{\Phi}_{\text{old}} \oplus \Delta\overrightarrow{\boldsymbol{\Phi}} > .\end{aligned} \tag{4.18}$$

The step increment Δ can be calculated by LM method. The Jacobian of feature j alone is given as an example:

$$J_j = \begin{bmatrix} \frac{\partial\bar{\boldsymbol{\Phi}}_j}{\partial\boldsymbol{\Xi}} & \frac{\partial\bar{\boldsymbol{\Phi}}_j}{\partial\Phi_{j_1}} & \cdots & \frac{\partial\bar{\boldsymbol{\Phi}}_j}{\partial\Phi_{j_s}} \end{bmatrix} \tag{4.19}$$

the first element in J_j is the derivative with respect to all the poses.

It is worth noting that the covariance adaptively varies according to Lemma 1, and the implicit function allows very flexible representation of the features in the environments. The "changeable parameters" is a way to parameterize the feature and adjust an initial value for the general feature when the more detailed shape information of the feature becomes available. Currently, the number of parameters is determined manually to help optimization. One empirical strategy to in determining the parameters number is to select the polynomial equation which can get the approximate shape in advance. This step can be assisted by some visualization tools. It should be noted that the number of polynomials should not be excessive, otherwise the convergence difficulty will be exacerbated.

4.4 Improved objective function for closed shape features

4.4.1 Asymmetry issue of general objective function

One inevitable problem of implicit functions is: for features with closed shapes, the value of implicit functions $\bar{\mathbf{\Phi}}_j$ in the error term $E_{\text{feature},j}$ varies from $-\varrho$ (inside the boundary, where $\varrho > 0$, varying from different implicit functions) to 0 (at the boundary) and then from 0 to $+\infty$ (outside the boundary). ϱ is generally a small number. The value of $\bar{\mathbf{\Phi}}_j$ changes slightly within the boundary, while the change outside the boundary is dramatic. This will make it difficult for the energy term to quickly decline to the optimal solution when the transformed observation points with respect to the estimated poses are inside the boundary during iterations.

4.4.2 Improved objective functions

We propose to improve $\bar{\mathbf{\Phi}}_j$ to $\bar{\mathbf{\Phi}}_j^\star$, following:

$$\bar{\mathbf{\Phi}}_j^\star = \log(\frac{1}{\varrho}\bar{\mathbf{\Phi}}_j + 1) \tag{4.20}$$

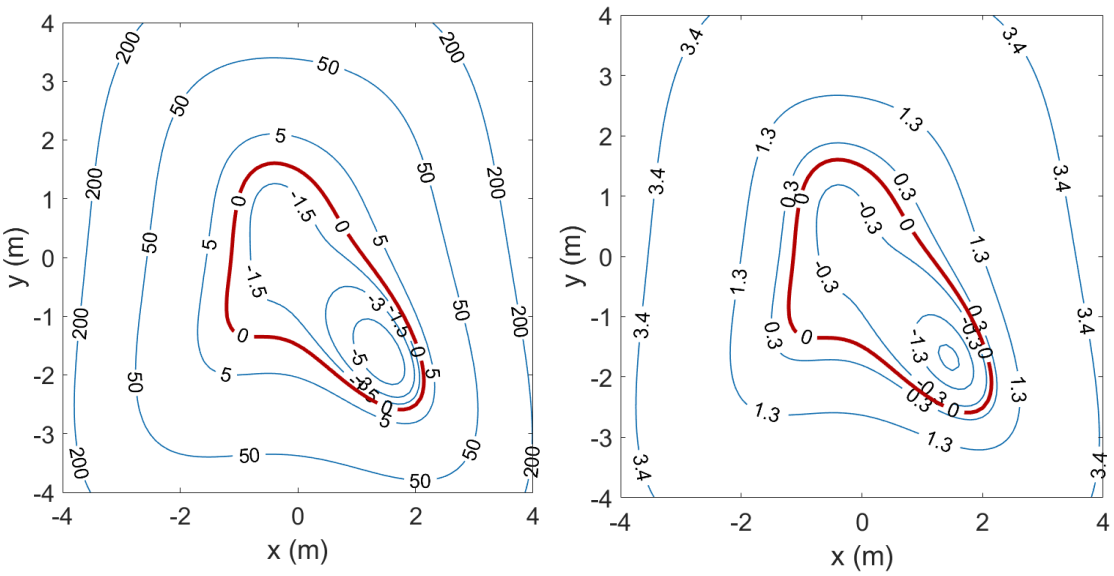
then the value of $\bar{\mathbf{\Phi}}_j^\star$ varies from $-\infty$ (inside the boundary) to $+\infty$ (outside the boundary).

Fig. 4.2a shows a general implicit function defined by the following function, taking $\mathbf{\Phi}_5$ (in Fig. 4.1) as an example:

$$\bar{\mathbf{\Phi}}_5(\mathbf{P}) = 1.2\mathbf{x}^4 + 0.4\mathbf{y}^4 + 2\mathbf{x}\mathbf{y} + 2.3\mathbf{x}^3\mathbf{y} - 2 \tag{4.21}$$

where a point set is $\mathbf{P} = [\mathbf{x}, \mathbf{y}] \in \mathbb{R}^{w\times 2}$ belonging to $\mathbf{\Phi}_5$. Obviously, the value of $\bar{\mathbf{\Phi}}_5$ outside the boundary grows rapidly (from 0 to 200), while that inside the boundary does not change much (from 0 to -5). Such value distribution can cause an inefficient descending problem.

Hence, a logarithmic function is applied to reconstruct the original implicit func-



(a) Value of $\bar{\mathbf{\Phi}}_5$

(b) Value of $\bar{\mathbf{\Phi}}_5^\star$

Figure 4.2 : Comparison of implicit functions for $\mathbf{\Phi}_5$. Red line is the boundary of feature $\mathbf{\Phi}_5$. Blue contours are the values of $\bar{\mathbf{\Phi}}_5$ and $\bar{\mathbf{\Phi}}_5^\star$, respectively.

tion as:

$$\bar{\mathbf{\Phi}}_5^\star(\mathbf{P}) = \log(\frac{1}{\varrho}\bar{\mathbf{\Phi}}_5(\mathbf{P}) + 1) \tag{4.22}$$

where $\varrho = 0.1543$ for $\mathbf{\Phi}_5$.

Fig. 4.2b depicts the improved implicit function. The value of $\bar{\mathbf{\Phi}}_5^\star$ inside and outside the boundary changes relatively evenly, showing a symmetry property and leading to a steady convergence.

One inevitable issue is how to choose a proper ϱ. Features in regular geometrical shapes, such as ellipse, line, or rectangle, have standard equations to describe the features' properties, which makes it easy to find ϱ. However, it is not easy to obtain a ϱ in advance if the shape of a feature is irregular and unknown. In this chapter, we assume the prior knowledge of features' shape is known, whether it is accurate or not. Then the ϱ is dynamically searched after every iteration.

4.5 Instance analysis

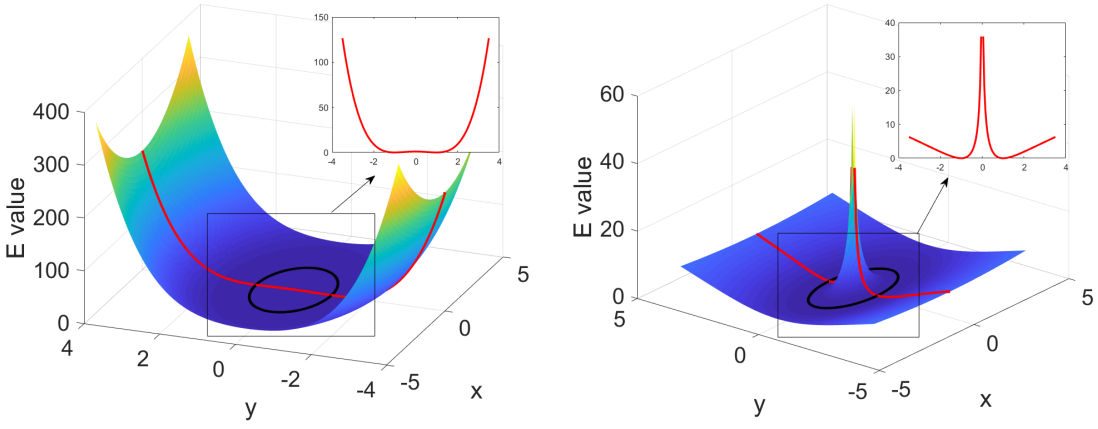
In this section, we use two examples (ellipse feature and line feature) to compare Implicit-SLAM and Pre-fit SLAM.

4.5.1 Implicit-SLAM: Ellipse feature

Suppose for an arbitrary point $\mathbf{p} = (x, y)^\intercal$ defined in the global frame. Then $\bar{\mathbf{\Phi}}_j$ is:

$$\begin{aligned}\bar{\mathbf{\Phi}}_j(\mathbf{p}) =& \frac{\left((x - F_x)\cos F_\phi + (y - F_y)\sin F_\phi\right)^2}{F_{r_1}^2} \\ &+ \frac{\left(-(x - F_x)\sin F_\phi + (y - F_y)\cos F_\phi\right)^2}{F_{r_2}^2} - 1\end{aligned} \tag{4.23}$$

where (F_x, F_y) is the center of the ellipse, F_ϕ is the angle between the major axis of the ellipse and x axis, F_{r_1} and F_{r_2} are the major and minor axis respectively, and the vector form of changeable parameters for ellipse feature is $\overrightarrow{\mathbf{\Phi}}_j = [F_x, F_y, F_\phi, F_{r_1} F_{r_2}]^\intercal$. The reason of formulating $\overrightarrow{\mathbf{\Phi}}_j$ as the given way is to facilitate comparison with the Pre-fit SLAM, which requires reasonable feature parameterization (discussed in Section 4.5.3 and Section 4.5.4).



(a) Value of $E_{\text{feature},j}$.

(b) Value of $E^\star_{\text{feature},j}$.

Figure 4.3 : Comparison of energy terms utilizing Eq. (4.23) and Eq. (4.24) for ellipse feature.

We reduce the dimensions of Eq. (4.23) to 2 by fixing the last three parameters of $\overrightarrow{\bar{\mathbf{\Phi}}_j}$ and robot poses in order to illustrate the convergence ability. Fig. 4.3a depicts the energy term $E_{\text{feature},j} = \left\|\bar{\mathbf{\Phi}}_j(\mathbf{p})\right\|_{\Sigma}^2$, and the black ellipse is the groundtruth ellipse $[0, 0, 0, 2, 1]^\intercal$ depicted in xy-plane. It seems that the function can converge well for any point from a macro perspective. However, the fact is that $E_{\text{feature},j}$ is hard to converge when the observed points locate within the ellipse area. As shown in the sub-figure in Fig. 4.3a, we drew a line where $x = 0$ as an instance. The value of $E_{\text{feature},j}$ varies slightly if $|y| \leq 1$, which means the gradient is too small. It is worth mentioning that this figure is only a two-dimensional example. The real function is in high-dimension, and it is far more difficult for the energy function to decline in the right direction when the observed points are inside the ellipse area.

As the result, we reconstruct the implicit function by:

$$\bar{\mathbf{\Phi}}_j^\star(\mathbf{p}_a) = \log(\bar{\mathbf{\Phi}}_j(\mathbf{p}_a) + 1). \tag{4.24}$$

Depict the new energy term in a similar way $E_{\text{feature},j}^\star = \left\|\bar{\mathbf{\Phi}}_j^\star(\mathbf{p})\right\|_{\Sigma}^2$ and the result is shown in Fig. 4.3b. In this example, the new objective function provides a sharp decline when points fall inside the ellipse. It is still easy to descend due to a large change in energy function.

The final ellipse implicit function written in vector form and the energy function is:

$$\begin{aligned} g(Z_{\text{ep},i,j}, \mathbf{\Xi}_i, \mathbf{F}_j) &= g(Z_{\text{ep},i,j}, \mathbf{\Xi}_i, \mathbf{\Phi}_j) = \bar{\mathbf{\Phi}}_j^\star(T(\mathbf{\Xi}_i, Z_{\text{ep},i,j})) \\ &= \log\left(\text{sum}_2\left(A^\intercal M \odot A^\intercal\right)\frac{1}{F_{r_1}^2} + \text{sum}_2\left(A^\intercal N \odot A^\intercal\right)\frac{1}{F_{r_2}^2}\right) \\ E_{\text{feature},j}^\star &= \frac{1}{2}\sum_i^n \left\|\bar{\mathbf{\Phi}}_j^\star(T(\mathbf{\Xi}_i, Z_{\text{ep},i,j}))\right\|_{\Sigma_{\mathbf{\Phi}_j,i}}^2 \end{aligned} \tag{4.25}$$

where $\odot$ is the Hadamard product, sum_2 returns a column vector containing the

sum of each row, and

$$
\begin{aligned}
A &= T(\boldsymbol{\Xi}_i, Z_{\text{ep},i,j}) - \mathbf{F}_{j_{xy}} \mathbf{1}_{w^\star}^\intercal \\
M &= \begin{bmatrix} c^2 & cs \\ cs & s^2 \end{bmatrix} \quad N = \begin{bmatrix} s^2 & -cs \\ -cs & c^2 \end{bmatrix}
\end{aligned} \tag{4.26}
$$

$\Sigma_{\boldsymbol{\Phi}_j,i}$ can be calculated according to Lemma 1:

$$
\begin{aligned}
\Sigma_{\boldsymbol{\Phi}_j,i}^{-1} &= \nabla g_{ij}^{-\intercal} \Sigma_z^{-1} \nabla g_{ij}^{-1} \\
\nabla g_{ij} &= \left. \frac{\partial \bar{\boldsymbol{\Phi}}_j^\star (T(\boldsymbol{\Xi}_i, Z_{\text{ep},i,j}))}{\partial p_{\boldsymbol{\Phi}_j}} \right|_{\{i\}p_{\boldsymbol{\Phi}_j}, \boldsymbol{\Xi}_i, \boldsymbol{\Phi}_j} \\
&= \frac{1}{C_0} \left(\frac{2\Delta^\intercal M R}{F_{r_1}^2} + \frac{2\Delta^\intercal N R}{F_{r_2}^2} \right) \\
C_0 &= C|_{\{i\}p_{\boldsymbol{\Phi}_j}, \boldsymbol{\Xi}_i, \boldsymbol{\Phi}_j} \\
\Delta &= T(\boldsymbol{\Xi}_i, {}^{\{i\}}p_{\boldsymbol{\Phi}_j}) - \mathbf{F}_{xy}
\end{aligned} \tag{4.27}
$$

We give a simplified example of one point in a feature. Suppose a single point $\mathbf{p} = [x_i; y_i]$. Also denote $\boldsymbol{\Phi}_{j_{xy}} = [F_x; F_y]$, $\cos(F_\phi)$ as c, and $\sin(F_\phi)$ as s. Thus, the updated Ellipse cost function turns to:

$$
\begin{aligned}
E_{\text{feature},j}^\star &= \frac{1}{2} \left\| g(Z_{\text{ep},i,j}, \boldsymbol{\Xi}_i, \boldsymbol{\Phi}_j) \right\|_{\Sigma_{\boldsymbol{\Phi}_j,i}}^2 \\
&= \frac{1}{2} \left\| \log \left(\frac{1}{F_{r_1}} \underbrace{\left(\mathbf{p} - \boldsymbol{\Phi}_{j_{xy}}\right)^\intercal M \left(\mathbf{p} - \boldsymbol{\Phi}_{j_{xy}}\right)}_{A} + \frac{1}{F_{r_2}} \underbrace{\left(\mathbf{p} - \boldsymbol{\Phi}_{j_{xy}}\right)^\intercal N \left(\mathbf{p} - \boldsymbol{\Phi}_{j_{xy}}\right)}_{B} \right) \right\|_{\Sigma_{\boldsymbol{\Phi}_j,i}}^2 \\
&= \left\| \log \left(\underbrace{\frac{A}{F_{r_1}^2} + \frac{B}{F_{r_2}^2}}_{C} \right) \right\|_{\Sigma_{\boldsymbol{\Phi}_j,i}}^2
\end{aligned} \tag{4.28}
$$

Then the corresponding Jacobian can be obtained by partially differential g with related to the state vector:

$$
\frac{\partial g}{\partial \boldsymbol{\Psi}} = \sum^{n} \frac{1}{C} \cdot \frac{\partial C}{\partial \boldsymbol{\Psi}} \tag{4.29}
$$

where

$$\frac{\partial C}{\partial \mathbf{t}} = \frac{1}{F_{r_1}^2}\frac{\partial A}{\partial \mathbf{t}} + \frac{1}{F_{r_2}^2}\frac{\partial B}{\partial \mathbf{t}} \Rightarrow \begin{cases} \frac{\partial A}{\partial \mathbf{t}} = 2\left(\mathbf{p} - \mathbf{\Phi}_{j_{xy}}\right)^{\intercal} M = 2\mathbf{D}^{\intercal} \\ \frac{\partial B}{\partial \mathbf{t}} = 2\left(\mathbf{p} - \mathbf{\Phi}_{j_{xy}}\right)^{\intercal} N = 2\mathbf{E}^{\intercal} \end{cases}$$

$$\frac{\partial C}{\partial \phi} = \frac{1}{F_{r_1}^2}\frac{\partial A}{\partial \phi} + \frac{1}{F_{r_2}^2}\frac{\partial B}{\partial \phi} \Rightarrow \begin{cases} \frac{\partial A}{\partial \phi} = 2\mathbf{D}^{\intercal}(dR)^{\{i\}}\mathbf{p} \\ \frac{\partial B}{\partial \phi} = 2\mathbf{E}^{\intercal}(dR)^{\{i\}}\mathbf{p} \end{cases}$$

where dR is the derivative of R

$$\frac{\partial C}{\partial \mathbf{\Phi}_{j_{xy}}} = \frac{1}{F_{r_1}^2}\frac{\partial A}{\partial \mathbf{\Phi}_{j_{xy}}} + \frac{1}{F_{r_2}^2}\frac{\partial B}{\partial \mathbf{\Phi}_{j_{xy}}} = -\frac{\partial C}{\partial \mathbf{t}} \Rightarrow \begin{cases} \frac{\partial A}{\partial F_{xy}} = -2D \\ \frac{\partial B}{\partial F_{xy}} = -2E \end{cases}$$

$$\frac{\partial C}{\partial F_\phi} = \frac{1}{F_{r_1}^2}\frac{\partial A}{\partial F_\phi} + \frac{1}{F_{r_2}^2}\frac{\partial B}{\partial F_\phi} \Rightarrow \begin{cases} \frac{\partial A}{\partial F_\phi} = \left(\mathbf{p} - \mathbf{\Phi}_{j_{xy}}\right)^{\intercal} dM \left(\mathbf{p} - \mathbf{\Phi}_{j_{xy}}\right) \\ \frac{\partial B}{\partial F_\phi} = \left(\mathbf{p} - \mathbf{\Phi}_{j_{xy}}\right)^{\intercal} dN \left(\mathbf{p} - \mathbf{\Phi}_{j_{xy}}\right) \end{cases}$$

$$\text{where } dM = \begin{bmatrix} -2cs & -s^2 + c^2 \\ -s^2 + c^2 & 2cs \end{bmatrix} \quad , \quad \text{and } dN = \begin{bmatrix} 2cs & s^2 - c^2 \\ s^2 - c^2 & -2cs \end{bmatrix}$$

$$\frac{\partial C}{\partial F_{r_1}} = -\frac{2A}{F_{r_1}^3} \quad , \quad \frac{\partial C}{\partial F_{r_1}} = -\frac{2B}{F_{r_2}^3} \tag{4.30}$$

4.5.2 Implicit-SLAM: Line feature

Similarly, assume ${}^{\{i\}}\mathbf{P}_{\mathbf{\Phi}_k}$ is the homogeneous point set of k^{th} line feature in the pose i. The observations are defined by $Z_{\text{lp},i,k} = {}^{\{i\}}\mathbf{P}_{\mathbf{\Phi}_k}$. $\overrightarrow{\mathbf{\Phi}_k} = [l_\alpha, p]^{\intercal}$ is the vector form of changeable parameters normalized by $p \geq 0$, l_α is the angle between the line and global x-axis, p is the distance from the origin point to the line. The implicit function of $\mathbf{\Phi}_k$ is:

$$\begin{aligned} g(Z_{\text{lp},i,k}, \mathbf{\Xi}_i, \mathbf{\Phi}_k) &= \bar{\mathbf{\Phi}}_k(T(\mathbf{\Xi}_i, Z_{\text{lp},i,k})) \\ &= \cos l_\alpha \mathbf{x} + \sin l_\alpha \mathbf{y} - p \end{aligned} \tag{4.31}$$

$\mathbf{x}$ and $\mathbf{y}$ are column vectors of $Z_{\text{lp},i,k}$. Since p is a scalar, the "-" operator represents p is subtracted from each element in the preceding term.

The line feature's implicit function can be formulated with the help of Lemma 2:

$$\begin{aligned} g(Z_{\text{lp},i,k}, \boldsymbol{\Xi}_i, \boldsymbol{\Phi}_k) &= \bar{\boldsymbol{\Phi}}_k(T(\boldsymbol{\Xi}_i, Z_{\text{lp},i,k})) \\ &= T(\boldsymbol{\Xi}_i, Z_{\text{lp},i,k})^\intercal \overrightarrow{\boldsymbol{\Phi}}_k^\circ \end{aligned} \tag{4.32}$$

where $\overrightarrow{\boldsymbol{\Phi}}_k^\circ = [\cos l_\alpha, \sin l_\alpha, -p]^\intercal$ is the normalized vector of $\overrightarrow{\boldsymbol{\Phi}}_k$. Hence, the energy functions of line is:

$$E_{\text{feature},k} = \frac{1}{2} \sum_i^n \left\| \bar{\boldsymbol{\Phi}}_k(T(\boldsymbol{\Xi}_i, Z_{\text{lp},i,k}) \right\|^2_{\Sigma_{\boldsymbol{\Phi}_k,i}} \tag{4.33}$$

The covariance $\Sigma_{\boldsymbol{\Phi}_k,i}$ is calculated as:

$$\begin{aligned} \Sigma^{-1}_{\boldsymbol{\Phi}_k,i} &= \triangledown h_{ik}^{-\intercal} \Sigma_z^{-1} \triangledown h_{ik}^{-1} \\ \triangledown h_{ik} &= \left. \frac{\partial h_{ik}}{\partial p_t} \right|_{{}^{\{i\}}p_{\boldsymbol{\Phi}_k}, \boldsymbol{\Xi}_i, \boldsymbol{\Phi}_k} \\ &= \overrightarrow{\boldsymbol{\Phi}_{k_{12}}}^{\circ\intercal} \end{aligned} \tag{4.34}$$

$\overrightarrow{\boldsymbol{\Phi}_{k_{12}}}^\circ$ is the first two elements of $\overrightarrow{\boldsymbol{\Phi}}_k^\circ$.

Similarly, a simplified example is given based on the existing normalization. Let $\overrightarrow{\boldsymbol{\Phi}}_k = [l_\alpha, p]^\intercal$ be the normalized feature state and $\overrightarrow{\boldsymbol{\Phi}}_k^\circ = [\cos l_\alpha, \sin l_\alpha, -p]^\intercal$. Then the Jacobian matrix is derived by:

$$\begin{aligned} \frac{\partial E_{\text{feature},k}}{\partial \mathbf{t}} &= \overrightarrow{\boldsymbol{\Phi}_{k_{12}}}^{\circ\intercal} \\ \frac{\partial E_{\text{feature},k}}{\partial \phi} &= \overrightarrow{\boldsymbol{\Phi}_{k_{12}}}^{\circ\intercal} (dR)^{\{i\}}\mathbf{p} \\ \frac{\partial E_{\text{feature},k}}{\partial l_\alpha} &= T^{-1}(\boldsymbol{\Xi}_i, {}^{\{i\}}\mathbf{p})^\intercal \begin{bmatrix} -\sin l_\alpha \\ \cos l_\alpha \end{bmatrix} \\ \frac{\partial E_{\text{feature},k}}{\partial p} &= -1 \end{aligned} \tag{4.35}$$

4.5.3 Pre-fit SLAM: Ellipse feature

The raw data can be used to fit an ellipse, one method is presented in (Zhao et al., 2019). Denote $\breve{\mathbf{Z}}_{\text{ef},i,j}$ as the observation of the j^{th} ellipse feature at pose i, then $\breve{\mathbf{Z}}_{\text{ef},i,j} = {}^{\{i\}}\breve{\mathbf{F}}_j$, where ${}^{\{i\}}\breve{\mathbf{F}}_j$ is the actual observation of ${}^{\{i\}}\mathbf{F}_j$.

Hence the pre-fit ellipse observation function can be written as:

$$g(\breve{\mathbf{Z}}_{\text{ef},i,j}, \mathbf{\Xi}_i, \mathbf{F}_j) = {}^{\{i\}}\breve{\mathbf{F}}_j - \begin{bmatrix} T^{-1}(\mathbf{\Xi}_i, \mathbf{F}_{j_{xy}}) \\ \text{dist}(F_{j_\phi} - \phi_i) \\ \mathbf{F}_{j_{r1,r2}} \end{bmatrix}_{5\times 1} \tag{4.36}$$

where $\mathbf{F}_{j_{xy}}$,F_{j_ϕ} and $\mathbf{F}_{j_{r1,r2}}$ are the position, angle and axis dimensions of the j^{th} feature in global frame, respectively. An extra *wrap* step is still needed for the 3^{rd} element.

The ellipse feature uncertainty $\Sigma_{\text{ef},i,j}$ is easily computed by $\Sigma_{\text{ef},i,j} = J^{-\intercal}\Sigma_z J^{-1}$ (Discussed in Chapter 3.2.3).

4.5.4 Pre-fit SLAM: Line feature

We replace $\mathbf{F}_k$ with $\mathbf{L}_k$ to represent the line feature to distinguish it from ellipse features. Suppose the k^{th} line state vector is parameterized by $\mathbf{l}_k = [\alpha_k, p_k]^\intercal (p_k \geq 0)$. Then the corresponding line feature $\mathbf{L}_k$ normalized by $\mathbf{l}_k$ is $\mathbf{L}_k = [\cos\alpha_k, \sin\alpha_k, -p_k]^\intercal$. It is worth noting that the line feature state includes normalization process. Suppose the k^{th} line state vector is parameterized by $\mathbf{l}_k = [\alpha_k, p_k]^\intercal (p_k \geq 0)$. Then the corresponding line feature $\mathbf{L}_k$ normalized by $\mathbf{l}_k$ is $\mathbf{L}_k = [\cos\alpha_k, \sin\alpha_k, -p_k]^\intercal$ which satisfies $\mathbf{L}_k^\intercal \hat{\mathbf{p}} = 0$ where $\hat{\mathbf{p}}$ is a point on the line feature in homogeneous coordinate. This mapping is denoted by $\mathbf{l}_k \mapsto \mathbf{L}_k$ and is reversible.

The observation of line features is obtained by intuitively minimizing the distance from discrete points to the line. Suppose a point ${}^{\{i\}}\mathbf{p}_w$ is defined in the local frame $\{i\}$ at pose i belonging to $\mathbf{L}_k$ and denote $\breve{\mathbf{Z}}_{\text{lf},i,k}$ as the observation of the k^{th} line feature at pose i. In order to avoid excessive mapping, the observations of line $\breve{\mathbf{Z}}_{\text{lf},i,k}$ are in the form of $\mathbf{L}_k$ and calculated by minimizing:

$$\underset{{}^{\{i\}}\mathbf{L}_k}{\text{argmin}} \sum_w {}^{\{i\}}\mathbf{L}_k^\intercal {}^{\{i\}}\hat{\mathbf{p}}_w \tag{4.37}$$

thus $\breve{\mathbf{Z}}_{\text{lf},i,k} = {}^{\{i\}}\mathbf{L}_k$.

Here, we propose Lemma 2 in order to help derive the following equations.

Lemma 2: Assume an arbitrary line feature $\mathbf{L}_k$ is defined in the global frame $\{G\}$. Suppose $T_i \in \mathrm{SE}(2)$ denotes the transformation from local frame $\{i\}$ to global frame $\{G\}$, then the corresponding line feature in the frame $\{i\}$ is:

$$^{\{i\}}\mathbf{L}_k = T_i^{\intercal}\mathbf{L}_k \tag{4.38}$$

Proof 2: Suppose an arbitrary point $^{\{i\}}\hat{\mathbf{p}}$ is allocated on the line $^{\{i\}}\mathbf{L}_k$ and both the point and the line are defined in the frame $\{i\}$. Then the point in the frame $\{G\}$ can be obtained by $\hat{\mathbf{p}} = T_i{}^{\{i\}}\hat{\mathbf{p}}$, which yields $\hat{\mathbf{p}}^{\intercal}\mathbf{L}_k = 0$. By direct calculation,

$$\hat{\mathbf{p}}^{\intercal}\mathbf{L}_k = (T_i{}^{\{i\}}\hat{\mathbf{p}})^{\intercal}\mathbf{L}_k = {}^{\{i\}}\hat{\mathbf{p}}^{\intercal}(T_i^{\intercal}\mathbf{L}_k) \tag{4.39}$$

Hence $^{\{i\}}\mathbf{L}_k = T_i^{\intercal}\mathbf{L}_k$.

The detailed derivation of pre-fit line model can be derived according to Lemma 2 as follows:

$$g(\breve{\mathbf{Z}}_{\mathrm{lf},i,k}, \boldsymbol{\Xi}_i, \mathbf{l}_k) = {}^{\{l\}}\mathbf{L}_k - T^{-1}(\boldsymbol{\Xi}_i, \mathbf{L}_k) \tag{4.40}$$

Remarked here that an implicit mapping $\mathbf{l}_k \leftarrow \mathbf{L}_k$ is done to accomplish the state vector.

The uncertainty of line energy function $\Sigma_{\mathrm{lf},i,k}$ can be calculated by

$$\Sigma_{\mathrm{lf},i,k} = \mathrm{diag}(\Sigma_z, 0) \tag{4.41}$$

according to Zhao et al. (Zhao et al., 2015, Eq. (19)).

4.6 Experiment and analysis

In this section several numerical examples were considered to analyze the performance of the proposed method. Firstly, we investigated the validity of Lemma 1; secondly, we compared the results of Implicit-SLAM and Pre-fit SLAM. Different

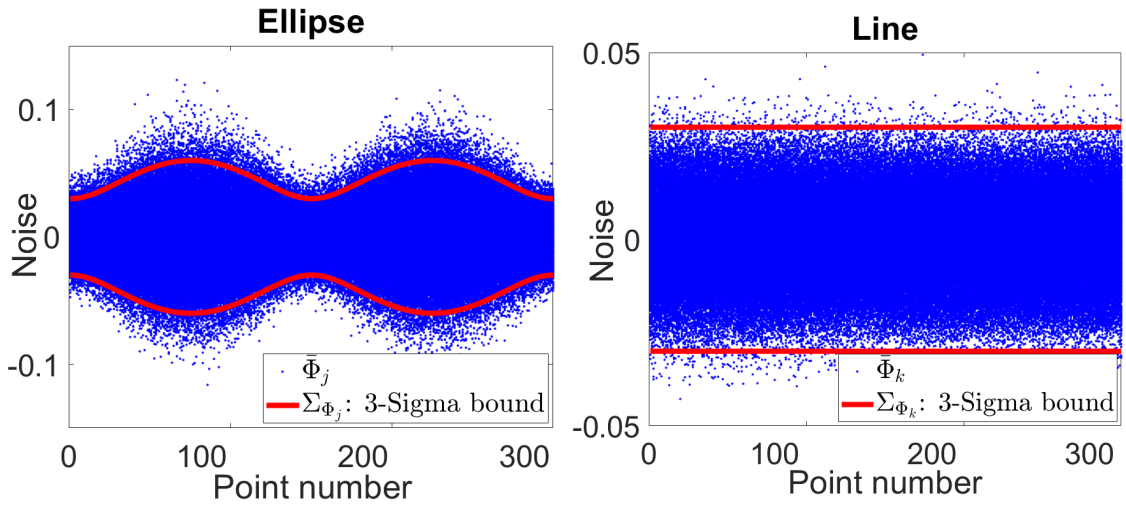
from elliptical feature based Pre-fit method in Chapter 3, the Pre-fit SLAM utilized in this chapter also takes line features into account. Then we tested Implicit-SLAM by fixing the covariance to evaluate Lemma 1. We also compared the performance between the improved implicit functions for ellipse feature with the original functions. Finally, we tested the robustness to observation noise level and checked the influence of fusing different types of features on both methods.

4.6.1 Simulation setup

The simulated environment is a 15 m×8 m space containing walls and ellipse features. The robot starts at $[0, 0, 0]^\intercal$ and odometry information is provided via a virtual wheel encoder with a random Gaussian noise $\text{diag}(0.4^2, 0.4^2, 3e^{-6})$. The initial observation noise is a random Gaussian noise $\mathbf{n}_z \sim N(\mathbf{0}, \text{diag}(0.05^2, 0.05^2))$. A 2D lidar is simulated with the valid range of 10m and the angle resolution of 0.33°. Range-Azimuth model is adopted for simulation, but the range-bearing data is transferred to Cartesian coordinate to form the observation. Only points within valid range and hit on features can be observed.

Our algorithm was tested in multiple settings: Pre-fit SLAM with ellipse feature only (denoted as pfE), with line feature only (pfL), and with both ellipse and line feature (pfEL); Implicit-SLAM method with ellipse feature only (denoted as pcE), with line feature only (pcL), and with both ellipse and line feature (pcEL). All the three Implicit-SLAM methods implemented variable covariance for ellipse and line features according to Lemma 1. As a comparison, Implicit-SLAM method with a given unchanged covariance for both ellipse and line features is prepared (pcEL_fixCov). Another two comparisons are Implicit-SLAM method with original ellipse objective function ($E_{\text{feature},j}$ by Eq. (4.23)) (pcEL_oldFun) and the same configuration except fixing covariance matrix (pcEL_oldFun_fixCov).

4.6.2 Validation of uncertainty transmission



(a) Noise of ellipse feature.

(b) Noise of Line feature.

Figure 4.4 : Uncertainty comparison. Red line is the 3-Sigma bound calculated by Lemma 1. Blue points are real values of implicit functions obtained by repeated experiments.

In this part, we used ellipse and line feature for verification of Lemma 1. We firstly verified whether $g(\mathbf{Z}_{\text{feature},i,j}, \mathbf{\Xi}_i, \mathbf{\Phi}_j)$ yields to $\Sigma_{\mathbf{\Phi}_j,i}$ via feature points by Monte Carlo experiment. 300 points on the edge of an ellipse and a line are selected respectively. Under the given noise, we repeatedly calculated both ellipse and line's implicit functions $\bar{\mathbf{\Phi}}_j$ and $\bar{\mathbf{\Phi}}_k$ with noisy observation and noisy $\mathbf{\Psi}$ by 1000 times and marked all results at each point as blue dots, as is illustrated in Fig. 4.4. The red line represents 3-sigma bound that obtained by Lemma 1. For both ellipse and line features, over 90% sampled data are strongly limited in 3-sigma bound, which verifies Lemma 1 statistically.

Since it is impossible to obtain $\mathbf{z}_0$ during practical experiments, which are the groundtruth of $\mathbf{z}$. Because the noise influence of observed points is similar to that of groundtruth points when the observations are near the exact positions, we use

the observed points to approximately calculate the covariance in Eq. (4.7). As is shown in Fig. 4.4, although $\Sigma^{-1}_{\mathbf{\Phi}_j,i}$ cannot cover more than 99% error points, it is still acceptable. And the imported errors in ellipse feature can be amended by the proposed improved objective function.

4.6.3 Result comparison in general environments

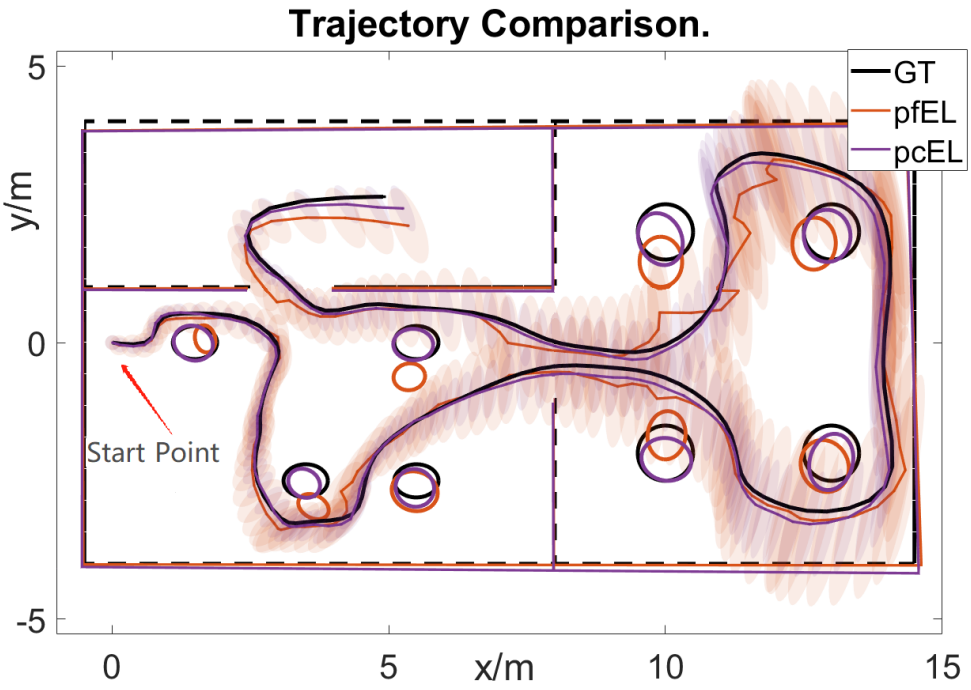


Figure 4.5 : Trajectory comparison. 3-sigma bound for robot's positions are depicted by shadowed ellipse in specific color.

We compared the results of pcEL and pfEL, as shown in Fig. 4.5. Because the line parameters in this chapter cannot represent line segments, the end points of each line at the first observation are maintained dependently (Do not participate in the optimization. We assume each observed line feature contains all the points.) and the resulted line features are drawn by transforming the end points to global frame. They are not accurate lines but to make the results look better.

Obviously, the trajectory of pcEL is much better than that of pfEL. Some sharp

"jump" occurred for pfEL due to the badly-fitted observations. The RMSE of pose is shown in Tab. 4.2. A main conclusion is that the pre-fit model usually possesses a higher error level than Implicit-SLAM model. The position error of pfEL is 0.1386m, while that of pcEL is 0.0912m, which is much smaller than pfEL. Also, the final covariance of position of pcEL is smaller than that of pfEL according to Fig. 4.5. It is noticed that pcEL_oldFun possesses an even smaller error in x-axis. It may related to the arrangement of the elliptical features (the main axis angle). If the angles are changed, the result may perform differently.

Table 4.2 : RMSE Comparison of multiple settings. The definition of abbreviations is in Section 4.6.1.

	x/m	y/m	$\mathbf{t}/m$	θ/rad
pfE	0.0974	0.1713	0.1971	0.0058
pfL	0.1038	0.1332	0.1689	0.0068
pfEL	0.0806	0.1128	0.1386	0.0059
pcE	0.0940	0.0762	0.1210	0.0067
pcL	0.0776	0.1695	0.1864	0.0059
pcEL	0.0749	**0.0520**	**0.0912**	**0.0043**
pcEL_fixCov	0.0780	0.0693	0.1043	0.0044
pcEL_oldFun	**0.0614**	0.0872	0.1066	0.0078
pcEL_oldFun_fixCov	0.0940	0.1608	0.1863	0.0047

Remark: pf stands for pre-fit; pc stands for Implicit-SLAM; E stands for ellipse; L stands for line.

Influence of Implicit covariance

We evaluated pcEL_fixCov in the same simulation environment. In Tab. 4.2, it can be found that the RMSE of pcEL_fixCov is slightly larger than pcEL, but smaller than any pre-fit approaches and Implicit-SLAM approaches.

Influence of improved cost function for closed shape features

By comparing pcEL_oldFun and pc_EL, It can be found that even if the RMSE at x-axis of pcEL_oldFun is the smallest, its translational and rotational RMSE (0.1066 and 0.0078) are bigger than those of pcEL (0.0912 and 0.0043). Hence, one conclusion is that the improved cost function for closed shape features can improve the accuracy of pose estimation.

Influence of different features

A secondary conclusion can be derived from Tab. 4.2 is that the combination of ellipse and line features can effectively improve the accuracy of the results compared with settings only using one type of feature, whether it is Pre-fit SLAM or Implicit-SLAM method.

4.6.4 Results on robustness against noise level

In the last experiment, we tested both Pre-fit SLAM and Implicit-SLAM with observing noise level increasing. The observing noise increases from 0.05 m to 0.1 m, and for each level we tested both algorithms by 50 Monte Carlo experiments and used the average error to depict Fig. 4.6.

It is clear that Implicit-SLAM is more robust to noise than Pre-fit SLAM on both position error and rotation error. The reason that pre-fit model performs badly is that the larger the noise is, the less accurate the fitted features are.

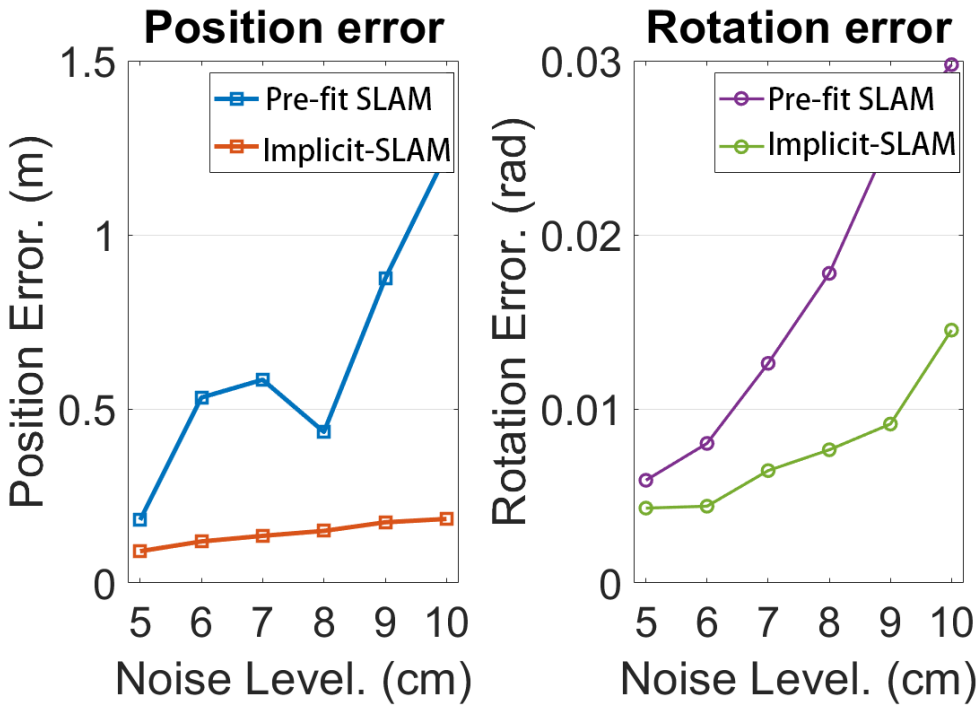
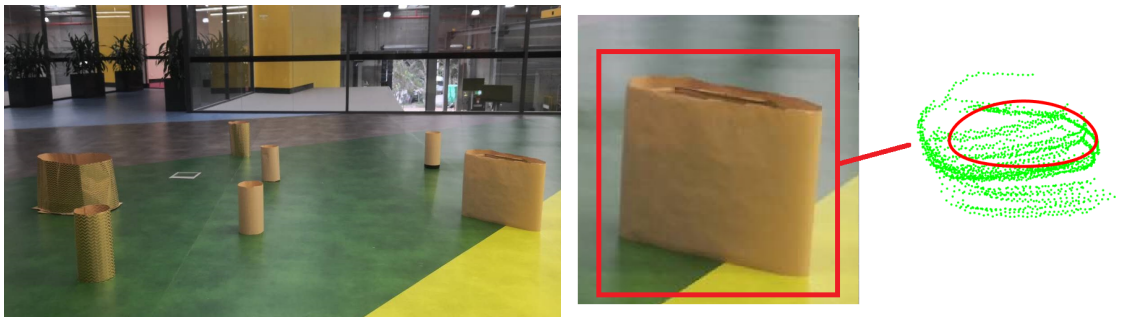


Figure 4.6 : Error changes with noise increasing.

4.6.5 Results on practical scenario

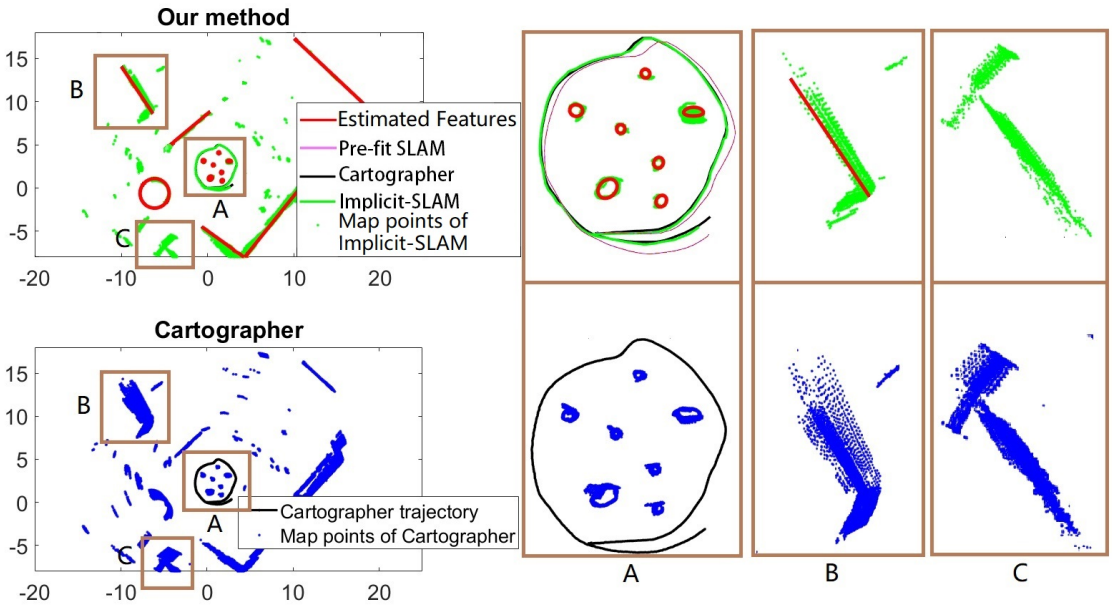
In this part we implemented Implicit-SLAM on a real scenario. As is shown in Fig. 4.7, the experiment environment is similar to the Chapter 3, where placed 7 artificial near-elliptical features and 1 round sofa inside glass walls. The data is collected via Fetch robot. The data association is executed by two aspects: 1. we clustered the discrete points of elliptical features by roughly projecting points back to the initial frame via odometry information since the number of features is known and features are sparsely placed; 2. a simple way is used to associate lines. We first extract lines at each single scan and then projecting line parameters to the initial frame via odometry. As each line is represented by the distance to the line and the angle of its normal line, a threshold is selected to determine to merge the same lines.

The experiment compared Implicit-SLAM and the state of the art Cartographer



(a) Practical scenario: a lounge consisting of several near-elliptical features.

(b) The irregular feature. On the right is the estimated feature by Implicit-SLAM.



(c) Comparison of Pre-fit SLAM, Implicit-SLAM and Cartographer. Region A, B and C are highlighted to compare the results.

Figure 4.7 : Practical experiment.

(Hess et al., 2016). The estimated features of Pre-fit SLAM are not drawn due to the bad performance of line features. Instead the trajectory result of Pre-fit SLAM was depicted as a comparison.

We chose 7 ellipse features and 1 irregular feature with quartic implicit function (the irregular feature) in order to evaluate Implicit-SLAM on general shape features. Since the specific equation of the irregular feature is not able to be obtained, we

use a quartic equation to roughly approximate the feature starting from assuming it as an ellipse initially. Then the global guess of irregular feature's parameters is given by quartic parameters. In this chapter, the observed points include all sides of features. The case where observation are open sets will be discussed in Chap. 5.

The results of Pre-fit SLAM, Implicit-SLAM and Cartographer are depicted in Fig. 4.7c. Similar to simulation, line feature are depicted as segments for a better visualization. In the first row we depicted results of Cartographer, Pre-fit SLAM and Implicit-SLAM together to show the difference. First of all, Pre-fit SLAM deviated from the other two methods because of the error imposed by wrongly estimated features. Since the groundtruth in real scenario is not available, we cannot quantitatively evaluate the two methods. However, it is possible to compare the results by re-projecting scan points back to the initial frame via estimated poses of either method. Three rectangle areas are highlighted in the figure. In region A, more points of Cartographer exceed features' boundaries, while Implicit-SLAM can maintain the basic shape of features. In region B and C, Cartographer's results show a clear dispersion compared with Implicit-SLAM.

4.7 Summary

In this chapter, a clear problem formulation and a solution framework for implicit function based SLAM problem are proposed. Two challenges involved in this novel SLAM problem are addressed. One is finding the covariance of the noises involved in implicit energy terms. Another is handling the asymmetry of the energy terms for closed shape features. Simulation results using ellipse and line features as examples shows that the proposed method is more robust to observation noises and outperforms the Pre-fit SLAM. It is also shown that using hybrid features can achieve better accuracy in SLAM compared with SLAM with only ellipses or lines. Practical experiment illustrates that Implicit-SLAM has the ability to acquire accurate

result.

Chapter 5

A post-count feature based SLAM approach on Fourier series: Fourier-SLAM

Last chapter proposed a post-count feature based SLAM approach using implicit functions. However, Implicit-SLAM with polynomial feature variables are sensitive to the initial guess, which will lead the algorithm to failure. This chapter continues digging into the post-count method and proposed another feature based SLAM approach with the help of Fourier series, which is called Fourier-SLAM in this chapter.

The main contributions of this chapter is that we propose a feature based SLAM approach implementing Fourier series to parameterize closed shape features (Section 5.2). Results (Section 5.4) show that using Fourier series as feature parameterization increases the accuracy of estimated trajectory as well as providing precise feature boundaries.

5.1 Motivation

In the applications of 2D laser rangefinder, various features are parameterized and used in SLAM problems such as point features, circular features, and polynomial features. Some researchers tried to estimate the contour of features during optimization and imported polynomial functions. In the last chapter, we took pre-defined polynomial functions as features and proposed an implicit function parameterization approach to solve feature based SLAM problem. The approach can estimate features in general shape with prior knowledge and calculate the uncertainty theoretically. However, it is still challenging to provide proper polynomial coefficients

as an initial guess if the prior knowledge of complex closed shape features is not available, resulting in being sensitive to the initial guess.

In this chapter, we propose Fourier-SLAM method. The advantages of Fourier series include:

1. Fourier series representation can express a feature's boundary accurately;
2. The number of coefficients in Fourier series is changeable, which makes the feature representation flexible. Truncated Fourier series is actually removing the part of data with high frequencies, which is usually the case of the random noise;
3. Feature's center has less influence on the estimated boundary, which makes the result reliable since an accurate initial guess from the noisy raw data is not available.

5.2 Closed shape feature parameterization

This section elaborates the problem formulation via Fourier series.

5.2.1 Fourier series coefficients estimation

Fourier series can be used to represent a periodic function via a sum of sine and cosine functions. For example, suppose $f(x)$ is a continuous function with period T and x is the indeterminate. $f(x)$ can be decomposed into triangular functions with different periods as :

$$f(x) := \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos\frac{2\pi n x}{T} + \sum_{n=1}^{\infty} b_n \sin\frac{2\pi n x}{T} \tag{5.1}$$

To make the problem clear, we discuss the estimation of coefficients from points within one time sequence and omit the superscript for simplification. Given the boundary set $\mathbf{P}_k$, the estimation of Fourier series coefficients a_n, b_n, are well defined

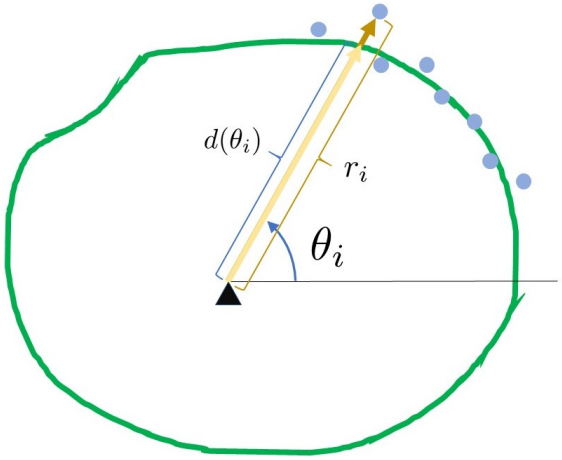


Figure 5.1 : Illustration of $r_i, \theta_i, d(\theta_i)$. The black triangle is the feature center.

and can be calculated. As shown in Fig. 5.1, denote r_i as the actual distance from point $\mathbf{p}_i^k$ to the feature's center, θ_i^k as the corresponding angle of $\mathbf{p}_i^k$ with respect to the x-axis in global frame. We use $d(\theta_i^k)$ to denote the fitted distance from the known or assumed center $\mathbf{C}_k$ to the ith point on the boundary, where $\theta_i^k \in [-\pi, \pi)$. Based on the above conception, one assumption is made:

Assumption 1: All features in the environment are in closed shapes (as shown in Fig. 5.6a). For any bearing beam pointing from the feature center to infinity, it has only one intersection with the feature's boundary.

The periodic distance function $d(\theta_i)$ of a feature can be represented by:

$$d(\theta_i) = \sum_{n=0}^{N} [a_n \cos(n\theta_i) + b_n \sin(n\theta_i)] \tag{5.2}$$

where the period is 2π according to Eq. (5.1). N can be any positive finite number. The larger the N, the closer the fitted boundary is to the observed points.

The coefficients of the Fourier series is a group of numbers such that the estimated boundary and the observed data has minimal difference, i.e., the solution of the following cost function (Rakshit and Monro, 2007):

$$\underset{a_n, b_n}{\operatorname{argmin}} \; F = \sum_{i=1}^{M} (d(\theta_i) - r_i)^2 \tag{5.3}$$

where M is the size of $\mathbf{P}_k$.

Noting that b_0 can be neglected since it has no influence on the above equation. By taking the first order differential of Eq. (5.3) with respect to a_k, b_k where $k = 0, \cdots, N$, and let them equal to zero:

$$\begin{aligned} \frac{\partial F}{a_k} &= 2\sum_{i=1}^{M} (d(\theta_i) - r_i)\cos(k\theta_i) = 0 \\ \frac{\partial F}{b_k} &= 2\sum_{i=1}^{M} (d(\theta_i) - r_i)\sin(k\theta_i) = 0 \end{aligned} \tag{5.4}$$

Since θ_i and r_i are determined numbers from the observation, Eq. (5.3) is actually a linear function of a_n, b_n, thus Eq. (5.4) is a quadratic cost function of a_n, b_n. The optimization problem can be solved through calculating $\mathbf{X} = [a_0, a_1, \cdots, a_N, b_1, \cdots, b_N]^\intercal$ from $A\mathbf{X} - \mathbf{S} = 0$, where

$$S = S_{(2N+1)\times 1} = [S_1; (\mathbf{S}_2)_{N\times 1}; (\mathbf{S}_3)_{N\times 1}] \tag{5.5}$$

where

$$\begin{cases} S_1 = \sum_{i=1}^{M} r_i \\ (\boldsymbol{S}_2)_p = \sum_{i=1}^{M} r_i \cos(p\theta_i) \\ (\boldsymbol{S}_3)_p = \sum_{i=1}^{M} r_i \sin(p\theta_i) \end{cases} \tag{5.6}$$

and

$$A \triangleq \begin{pmatrix} M & (\boldsymbol{A}_{12})_{1\times N} & (\boldsymbol{A}_{13})_{1\times N} \\ (\boldsymbol{A}_{21})_{N\times 1} & (\boldsymbol{A}_{22})_{N\times N} & (\boldsymbol{A}_{23})_{N\times N} \\ (\boldsymbol{A}_{31})_{N\times 1} & (\boldsymbol{A}_{32})_{N\times N} & (\boldsymbol{A}_{33})_{N\times N} \end{pmatrix} \tag{5.7}$$

where

$$\begin{cases} (\boldsymbol{A}_{12})_{1q} = \sum_{i=1}^{M} \cos(q\theta_i) \quad , \quad \boldsymbol{A}_{21} = \boldsymbol{A}_{12}^{\intercal} \\ (\boldsymbol{A}_{13})_{1q} = \sum_{i=1}^{M} \sin(q\theta_i) \quad , \quad \boldsymbol{A}_{31} = \boldsymbol{A}_{13}^{\intercal} \\ (\boldsymbol{A}_{22})_{pq} = \sum_{i=1}^{M} \cos(p\theta_i)\cos(q\theta_i) \\ (\boldsymbol{A}_{23})_{pq} = \sum_{i=1}^{M} \cos(p\theta_i)\sin(q\theta_i) \\ (\boldsymbol{A}_{32})_{pq} = \sum_{i=1}^{M} \sin(p\theta_i)\cos(q\theta_i) \\ (\boldsymbol{A}_{33})_{pq} = \sum_{i=1}^{M} \sin(p\theta_i)\sin(q\theta_i) \end{cases} \tag{5.8}$$

then the optimal solution is $\mathbf{X} = A^{-1}\mathbf{S}$.

5.2.2 Partial observation complement and center estimation

The estimation of Fourier series coefficients is based on the assumption that all points are referenced at the feature's center. Therefore, feature points need to be converted from global frame to feature's local coordinate in advance.

For any feature point set $\mathbf{P}_k$ with the center $\mathbf{C}_k$, the corresponding points in feature's local coordinate are denoted as $\hat{\mathbf{P}}_k$. One possible way to get $\mathbf{C}_k$ is finding the average center of $\mathbf{P}_k$. Thus for each point within the point set, $\hat{\mathbf{p}}_i^k = \mathbf{p}_i^k - \mathbf{C}_k$, where $\mathbf{C}_k = [C_{k,x}, C_{k,y}]^{\intercal}$ is the center of the point set belonging to the feature $\boldsymbol{\Phi}_k$.

However, one problem of this way is that the center will be shifted if only part of the boundary is observed, leading to wrong fitting. Starting from this concern, we propose a complement process. As shown in Fig. 5.2, partial points will lead to a bad average center for fitting. Thus, we initially use a circle equation to approximately find a center from the partial points. This center is served as the complemented center. Then partial points are transformed to the frame defined in the complemented center, after which sort all points in the order of ascending angles θ. Next,

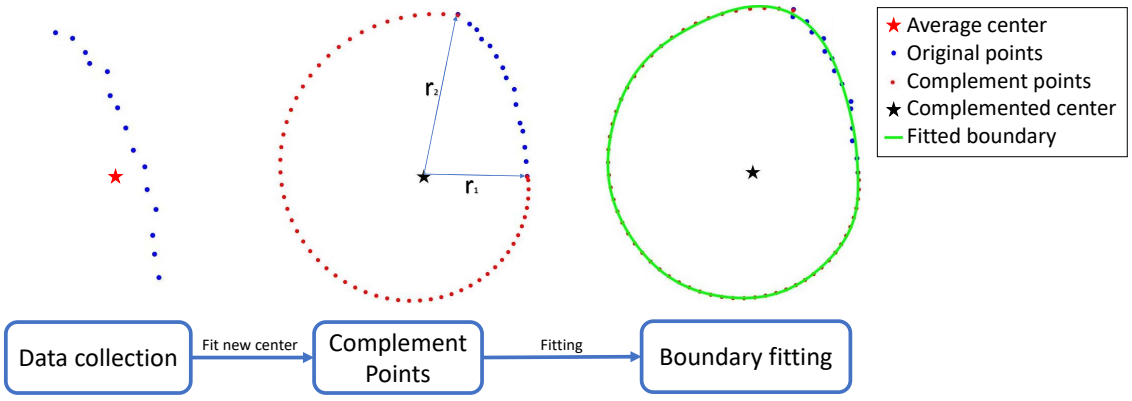


Figure 5.2 : Coefficients fitting process.

the starting point and the end point among these partial points are selected (with the help of ascending angles), denoting their distances to the feature's center as r_1 and r_2, respectively. The complemented points can be predicted by linearly interpolating distance between r_1 and r_2. Finally, the boundary is fitted using original partial points and the complemented points together.

The complemented center is still inaccurate as the unknown points are unpredictable. The complemented center could be different from the actual center. Fortunately, thanks to the advantage of Fourier series fitting, it is reasonable to process center complement since the fitted boundary is less-influenced by the selected feature center. As shown in Fig. 5.3, the boundary is well-fitted among different selected fitting centers even if the center is not located at the groundtruth. The only condition to the selected center is that it must locate within the boundary and satisfy Assumption 1.

The complemented points are helpful during initializing Fourier series. Note that once the coefficients are initialized, the complemented points will be dropped and do not join in the SLAM optimization. With the initialized coefficients the estimation process is able to deal with features which can only be observed partially.

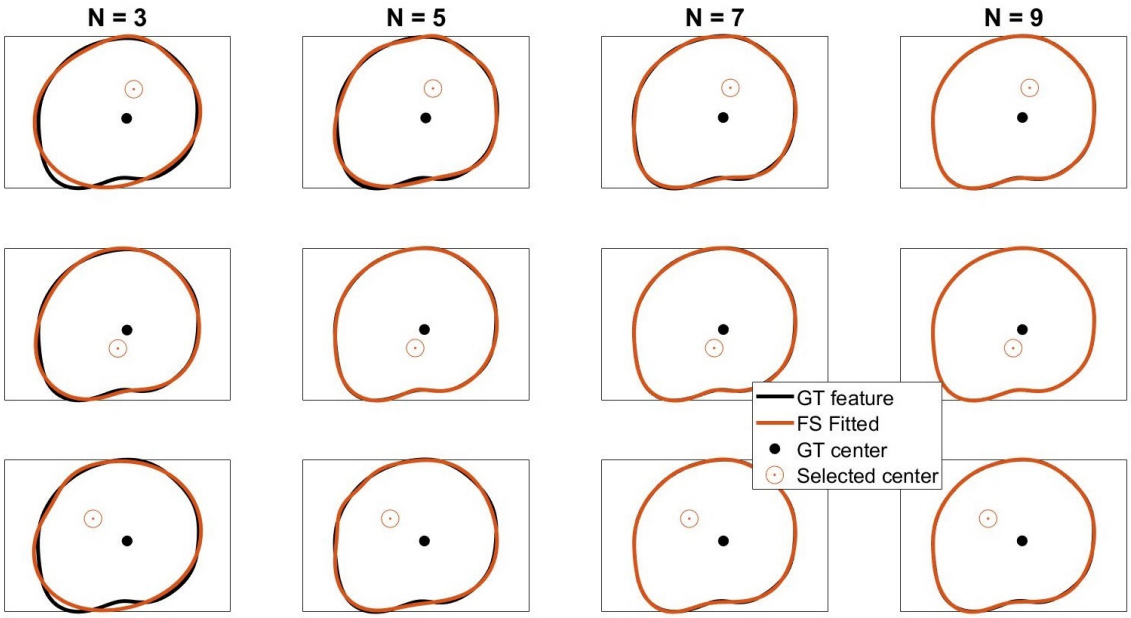


Figure 5.3 : Different feature centers result in the same boundary. The larger the N is, the closer the fitted boundary is to the groundtruth.

5.2.3 Choice of N

In this chapter, N is a chosen number such that using truncated Fourier series to estimate the boundary of the feature possesses enough precision, and N can be increased to acquire a more accurate result. As shown in Fig. 5.3, the fitted boundary is fitted well enough for $N \geq 7$, while $N = 3$ shows an approximated boundary. As N increases, the resultant boundary gets closer to the actual points. Note that N should be limited to less than the number of fitting points, otherwise the result is over-fitting causing the boundary oscillating. Empirically, N can be set to 5 or 7 for most of the smooth closed shape features, and it can also be set to 15 or 17 for most of the rectangular features.

In order to evaluate the accuracy of the feature coefficients estimate, we need the groundtruth of the Fourier series coefficients. The feature's groundtruth are generated by three steps: (a) set feature's groundtruth center and manually select

30 boundary points for each feature; (b) fit points with Fourier series with a pre-set N; (c) use the fitted Fourier series as the groundtruth to generate the feature shape using Eq. (5.2).

5.3 Problem definition of Fourier-SLAM

In this section, the problem definition of Fourier-SLAM is introduced in details.

5.3.1 Problem formulation of Fourier-SLAM

The original observation in Fourier-SLAM is raw laser points. One inevitable issue is that it is difficult to acquire correct data association in general. However, considering the application environment of Fourier-SLAM, we are able to associate laser points on the same feature aided by odometry information. Hence, in order to make formulations easy to understand, an assumption on data association is made:

Assumption 2: The raw laser points are segmented into point sets corresponding to different features. The data association is done by clustering points.

The raw observation points represent a real position in Euclidean space and it cannot be used directly to estimate coefficients of Fourier series. Consequently, laser points in Euclidean coordinates should be transformed to each feature's local polar coordinates first, and the center of these points are kept for the following optimization.

Assume the initial pose $\mathbf{\Xi}_0 = [0; 0; 0]$ coincides with the global frame, and a point set ${}^{\{j\}}\mathbf{P}_k$ with a size of M_k^j observed in the jth frame belongs to feature $\mathbf{\Phi}_k$. Only one feature is considered in this section for simplification. For the pose set $\mathbf{\Xi}_{j=1:L}$, we aim to optimize vector $\mathbf{X} = [\mathbf{\Xi}_{j=1:L}; \mathbf{\Phi}_k]$ where $\mathbf{\Phi}_k \overset{\text{def.}}{=} [\mathbf{C}_k; \mathbf{V}_k]$ and $\mathbf{V}_k = [a_0^k, a_1^k, b_1^k, \cdots, a_N^k, b_N^k]^\intercal$. Then the optimization problem is to minimize the

following energy function:

$$\underset{\mathbf{X}}{\operatorname{argmin}}\ F(\mathbf{X}) = \sum_{\mathbf{\Xi}_{j=1}}^{L} \left(E_{\mathrm{o},j} + \sum_{k}^{K} \left[E_{\mathrm{f},j,k} + E_{\mathrm{c},j,k} \right] \right) \tag{5.9}$$

$E_{\mathrm{o},j}$ is the energy cost of the odometry, $E_{\mathrm{f},j,k}$ is the distance cost between estimated boundaries and observed points, $E_{\mathrm{c},j,k}$ is the energy cost of observed feature centers fitted at each step. Each term in Eq. (5.9) is defined by:

$$\begin{aligned} E_{\mathrm{o},j} &= \| f(\mathbf{Z}_{\mathrm{o},j}, \mathbf{\Xi}_{j-1}, \mathbf{\Xi}_j) \|^2_{\Sigma^{-1}_{\mathrm{o},j}} \\ E_{\mathrm{f},j,k} &= \left\| \sum_{n=0}^{N} \left[a_n^k \cos(n\boldsymbol{\theta}^{jk}) + b_n^k \sin(n\boldsymbol{\theta}^{jk}) \right] - \mathbf{r}^{jk} \right\|^2_{\Sigma^{-1}_{\mathrm{f},j,k}} \\ E_{\mathrm{c},j,k} &= \left\| T^{-1}(\mathbf{\Xi}_j, \mathbf{C}_k) - {}^{\{j\}}\mathbf{C}_k \right\|^2_{\Sigma^{-1}_{\mathrm{c},j,k}} \end{aligned} \tag{5.10}$$

where $\mathbf{Z}_{\mathrm{o},j}$ is the observation vector of jth odometry, $\Sigma^{-1}_{\mathrm{o},j}$ is the odometry covariance at the jth step. $\Sigma^{-1}_{\mathrm{f},j,k}$ is the covariance of feature k's boundary error at jth step. $\Sigma^{-1}_{\mathrm{c},j,k}$ is the covariance of fitted center at jth step. ${}^{\{j\}}\mathbf{C}_k$ is obtained through a circle-fitting process at each time sequence. The coefficients a_n^k and b_n^k are initialized by the complemented points. $\boldsymbol{\theta}^{jk} = [\theta_1^{jk}, \cdots, \theta_{M_k^j}^{jk}]^\intercal$ and $\mathbf{r}^{jk} = [r_1^{jk}, \cdots, r_{M_k^j}^{jk}]^\intercal$, where θ_i^{jk} and r_i^{jk} are calculated by:

$$\begin{aligned} \theta_i^{jk} &= \arctan(\frac{\hat{y}_i^{jk}}{\hat{x}_i^{jk}}) \quad , \quad r_i^{jk} = \sqrt{(\hat{x}_i^{jk})^2 + (\hat{y}_i^{jk})^2} \\ \begin{bmatrix} \hat{x}_i^{jk} \\ \hat{y}_i^{jk} \end{bmatrix} &= \begin{bmatrix} c_j {}^{\{j\}}x_i^k - s_j {}^{\{j\}}y_i^k + t_{j,x} - C_{k,x} \\ s_j {}^{\{j\}}x_i^k + c_j {}^{\{j\}}y_i^k + t_{j,y} - C_{k,y} \end{bmatrix} \end{aligned} \tag{5.11}$$

where $\hat{\mathbf{p}}_i^{jk} = [\hat{x}_i^{jk}, \hat{y}_i^{jk}]^\intercal$ is the ith point in feature k transformed from the jth frame, and θ_i^{jk} is the angle of $\hat{\mathbf{p}}_i^{jk}$, r_i^{jk} is the distance from $\hat{\mathbf{p}}_i^{jk}$ to the feature's center $\mathbf{C}_k$, and both of them are calculated by global points transformed from the jth frame. c_j, s_j represent $\cos\phi_j$ and $\sin\phi_j$.

5.3.2 Optimization and uncertainty transmission

Eq. (5.9) is able to be solved by iterated methods such as Newton's method or Levenberg-Marquardt algorithm. In this chapter, Levenberg-Marquardt algorithm

is taken as the problem solver. The covariance matrices $\Sigma_{\mathrm{o},j}$, $\Sigma_{\mathrm{f},j,k}$ and $\Sigma_{\mathrm{c},j,k}$ of the problem will change in each iteration, which are used as the weights for the three terms in Eq. (5.9).

Since a common odometry model is utilized, here we only consider the feature part for the sake of simplification.

Denote dR_j as the derivative of R_j with respect to ϕ_j and let B and D represent:

$$\begin{aligned} B &= \sum_{n=0}^{N} \left[a_n^k \cos(n\boldsymbol{\theta}^{jk}) + b_n^k \sin(n\boldsymbol{\theta}^{jk})\right] - \mathbf{r}^{jk} \\ D &= T^{-1}(\boldsymbol{\Xi}_{je}, \mathbf{C}_k) - {}^{\{j\}}\mathbf{C}_k \end{aligned} \tag{5.12}$$

The Jacobian of D is $J_D = \begin{bmatrix} \frac{\partial D}{\partial \mathbf{t}_j} & \frac{\partial D}{\partial \phi_j} & \frac{\partial D}{\partial \mathbf{C}_k} \end{bmatrix}$:

$$\frac{\partial D}{\partial \mathbf{t}_j} = -R_j^{\intercal} \ , \ \frac{\partial D}{\partial \phi_j} = dR^{\intercal}(\mathbf{C}_k - \mathbf{t}_j) \ , \ \frac{\partial D}{\partial \mathbf{C}_k} = R_j^{\intercal} \tag{5.13}$$

For each row in B, the Jacobian of B_i with respect to the state vector is:

$$J_{B_i} = \begin{bmatrix} \frac{\partial B_i}{\partial \mathbf{t}_j} & \frac{\partial B_i}{\partial \phi_j} & \frac{\partial B_i}{\partial \mathbf{C}_k} & \frac{\partial B_i}{\partial a_{s_1=0:N}} & \frac{\partial B_i}{\partial b_{s_2=1:N}} \end{bmatrix} \tag{5.14}$$

where

$$\begin{aligned} \frac{\partial B_i}{\partial \mathbf{t}_j} &= \sum_{n=0}^{N} [a_n \frac{\partial \cos(n\theta_i^{jk})}{\partial \mathbf{t}_j} + b_n \frac{\partial \sin(n\theta_i^{jk})}{\partial \mathbf{t}_j}] - \frac{\partial r_i^{jk}}{\partial \mathbf{t}_j} \\ \frac{\partial \cos(n\theta_i^{jk})}{\partial \mathbf{t}_j} &= \frac{n \sin(n\theta_i^{jk})}{(\hat{\mathbf{p}}_i^{jk})^{\intercal} \hat{\mathbf{p}}_i^{jk}} (\hat{\mathbf{p}}_i^{jk})^{\intercal} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \\ \frac{\partial \sin(n\theta_i^{jk})}{\partial \mathbf{t}_j} &= \frac{n \sin(n\theta_i^{jk})}{(\hat{\mathbf{p}}_i^{jk})^{\intercal} \hat{\mathbf{p}}_i^{jk}} (\hat{\mathbf{p}}_i^{jk})^{\intercal} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \\ \frac{\partial r_i^{jk}}{\partial \mathbf{t}_j} &= \frac{(\hat{\mathbf{p}}_i^{jk})^{\intercal}}{|\hat{\mathbf{p}}_i^{jk}|} \end{aligned} \tag{5.15}$$

$$
\begin{aligned}
\frac{\partial B_i}{\partial \phi_j} &= \sum_{n=0}^{N}[a_n \frac{\partial \cos(n\theta_i^{jk})}{\partial \phi_j} + b_n \frac{\partial \sin(n\theta_i^{jk})}{\partial \phi_j}] - \frac{\partial r_i^{jk}}{\partial \phi_j} \\
\frac{\partial \cos(n\theta_i^{jk})}{\partial \phi_j} &= -n\sin(n\theta_i^{jk}) \frac{({}^{\{j\}}\hat{\mathbf{p}}_i^k)^\intercal R_j^\intercal \hat{\mathbf{p}}_i^{jk}}{(\hat{\mathbf{p}}_i^{jk})^\intercal \hat{\mathbf{p}}_i^{jk}} \\
\frac{\partial \sin(n\theta_i^{jk})}{\partial \phi_j} &= n\cos(n\theta_i^{jk}) \frac{({}^{\{j\}}\hat{\mathbf{p}}_i^k)^\intercal R_j^\intercal \hat{\mathbf{p}}_i^{jk}}{(\hat{\mathbf{p}}_i^{jk})^\intercal \hat{\mathbf{p}}_i^{jk}} \\
\frac{\partial r_i^{jk}}{\partial \phi_j} &= \frac{1}{|\hat{\mathbf{p}}_i^{jk}|} ({}^{\{j\}}\hat{\mathbf{p}}_i^k)^\intercal R_j^\intercal \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \hat{\mathbf{p}}_i^{jk} \\
\frac{\partial B_i}{\partial \mathbf{C}_k} &= -\frac{\partial B_i}{\partial \mathbf{t}_j} \\
\frac{\partial B_i}{\partial a_{s_1}} &= \cos(s_1\theta_i^{jk}) \quad , \quad \frac{\partial B_i}{\partial b_{s_2}} = \sin(s_2\theta_i^{jk})
\end{aligned}
\tag{5.16}
$$

The covariance in this problem is not explicit to acquire. We adopt implicit covariance Lemma 1 in Chapter 4.3 to approximately calculate the covariance matrices.

The observed center ${}^{\{j\}}\mathbf{C}_k$ is fitted through circle function, which is applicable for Lemma 1. Then:

$$
\begin{aligned}
f_1 &= \sum_{i=1}^{M_k^j} \left\| |{}^{\{j\}}\hat{\mathbf{p}}_i^k - {}^{\{j\}}\mathbf{C}_k|^2 - r_i^2 \right\|_{\Sigma_{f_1}^{-1}}^2 \\
J_{f_{1z}} &= 2({}^{\{j\}}\hat{\mathbf{p}}_i^k - {}^{\{j\}}\mathbf{C}_k)^\intercal \ , \ J_{f_{1x}} = -2({}^{\{j\}}\hat{\mathbf{p}}_i^k - {}^{\{j\}}\mathbf{C}_k)^\intercal \\
\Sigma_{f_1} &= J_{f_{1z}} \Sigma_z J_{f_{1z}}^\intercal \qquad , \ \Sigma_{{}^{\{j\}}\mathbf{C}_k}^{-1} = J_{f_{1x}}^\intercal \Sigma_{f_1}^{-1} J_{f_{1x}}
\end{aligned}
\tag{5.17}
$$

hence, $\Sigma_{\mathrm{c},j,k}^{-1}$ is calculated by:

$$
\Sigma_{\mathrm{c},j,k}^{-1} = J_D^\intercal \Sigma_{{}^{\{j\}}\mathbf{C}_k}^{-1} J_D \tag{5.18}
$$

The calculation of $\Sigma_{\mathrm{f},j,k}$ also needs Lemma 1. For each row in B, the Jacobian

of B_i w.r.t. the observed points is:

$$\begin{aligned} J_{B_i,z} &= \sum_{n=0}^{N}[a_n \frac{\partial \cos(n\theta_i^{jk})}{\partial^{\{j\}}\mathbf{p}_i^k} + b_n \frac{\partial \sin(n\theta_i^{jk})}{\partial^{\{j\}}\mathbf{p}_i^k}] - \frac{\partial r_i^{jk}}{\partial^{\{j\}}\mathbf{p}_i^k} \\ \frac{\partial \cos(n\theta_i^{jk})}{\partial^{\{j\}}\mathbf{p}_i^k} &= \frac{\partial \cos(n\theta_i^{jk})}{\partial \mathbf{t}_j} \cdot R_j \\ \frac{\partial \sin(n\theta_i^{jk})}{\partial^{\{j\}}\mathbf{p}_i^k} &= \frac{\partial \sin(n\theta_i^{jk})}{\partial \mathbf{t}_j} \cdot R_j \\ \frac{\partial r_i^{jk}}{\partial^{\{j\}}\mathbf{p}_i^k} &= \frac{\partial r_i^{jk}}{\partial \mathbf{t}_j} \cdot R_j \end{aligned} \tag{5.19}$$

where $J_{B,z} = [J_{B_1,z}; \cdots ; J_{B_M,z}]$. Following Lemma 1, $\Sigma_{\mathrm{f},j,k}$ is calculated by:

$$\Sigma_{\mathrm{f},j,k} = J_{B,z} \Sigma_z J_{B,z}^{\intercal} \tag{5.20}$$

5.4 Experiment and analysis

In this section several simulations and experiments were performed and Fourier-SLAM is compared with Implicit-SLAM to analyze the performance and evaluated the validity. The algorithm is tested on the laptop with Intel i7, 16GB RAM. All the code are finished in MATLAB.

5.4.1 Simulation setup

The simulation environment is a 15m×10m space containing several irregular closed shape features. The robot starts at $[0, 0, 0]^{\intercal}$ and odometry information is provided via a virtual wheel encoder with a random zero-mean Gaussian noise $\mathrm{diag}[0.05^2, 0.05^2, 4e^{-6}]$ for (x, y, ϕ). The noise of raw observed points is a random zero-mean Gaussian noise $\mathrm{diag}[0.05^2, 0.05^2]$. A 2D lidar is simulated with the valid range of 10m and the angle resolution of 0.33°. Range-Azimuth model is adopted for simulation, but the range-bearing data is transformed to Cartesian coordinate to form the observation. Only points within valid range and hit on features can be observed.

5.4.2 Accuracy evaluation

The results of Implicit-SLAM and Fourier-SLAM are compared and shown in Fig. 5.4. Both algorithms share the same initial guess of the robot poses*. A second order implicit function is used in the Implicit-SLAM. $N = 7$ is used in our Fourier-SLAM.

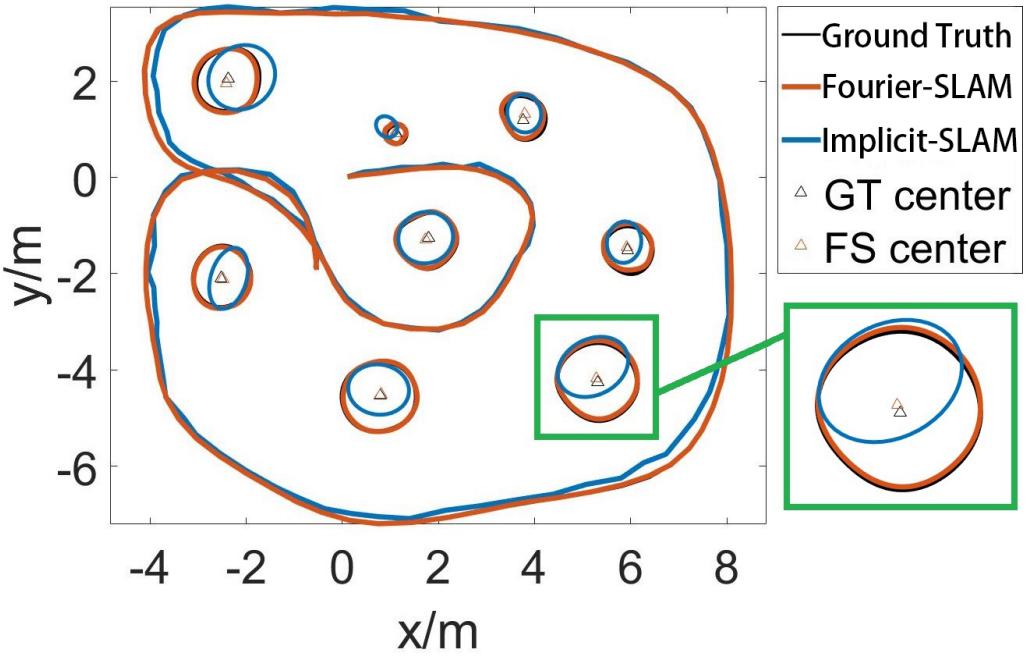


Figure 5.4 : Trajectory comparison between Implicit-SLAM and Fourier-SLAM. GT center means the groundtruth of feature centers. FS center means the estimated feature centers of Fourier-SLAM.

The estimated trajectories of both the two methods are close to the groundtruth, while the one obtained from Implicit-SLAM drifts at the bottom. Features estimated in Fourier-SLAM coincide approximately with the groundtruth features' boundary, which illustrates that Fourier-SLAM has the ability to represent real features as closely as possible. It should be noted that the estimated centers in Fourier-SLAM

*Since Implicit-SLAM is sensitive to initial guess, we use actual poses with perturbation as initial guess for both methods.

do not have a significant influence on the estimation process. Even though the estimated center does not always locate at the feature's real center, the estimated boundary still shows a good performance.

Figure 5.5 : Pose error of every step. The dash line in each sub figure is the average difference between the estimated result and the groundtruth for all the steps.

Fig. 5.5 illustrate pose error of each step. It can be seen that the average error of Fourier-SLAM in x, y and θ is smaller than that of Implicit-SLAM. The final RMSE in translation and rotation of Fourier-SLAM are 0.0526m and 0.01rad, while that of Implicit-SLAM are 0.1513m and 0.0125rad.

5.4.3 Results on practical experiments

In this part we implemented Fourier-SLAM on real scenarios, as is shown in Fig. 5.6a. The first scenario is the same one as conducted in Chapter 3 and Chapter 4. The valid laser range is set to 4m in order to filter outlier laser points hit on or pass through the glass walls. The data association is executed by clustering the discrete points of features. We project the points back to the initial frame via odometry

(a) Scenario A: a lounge consisting of several irregular closed shape features.

(b) Scenario B: underground car park.

Figure 5.6 : Environment of practical experiments.

information since the number of features is known and features are sparsely placed. After that points of each feature are segmented and associated among all time sequences.

The experiment compared Fourier-SLAM with Implicit-SLAM and the state of the art Cartographer (Hess et al., 2016). The results are depicted in Fig. 5.7. In Fig. 5.7, the top-left image shows the difference in trajectory among three methods. Since the groundtruth is not accessible, one intuitive way to compare trajectories is to evaluate the performance of back-projected laser points via estimated poses of all the methods. The top-right image shows the back-projected laser points of

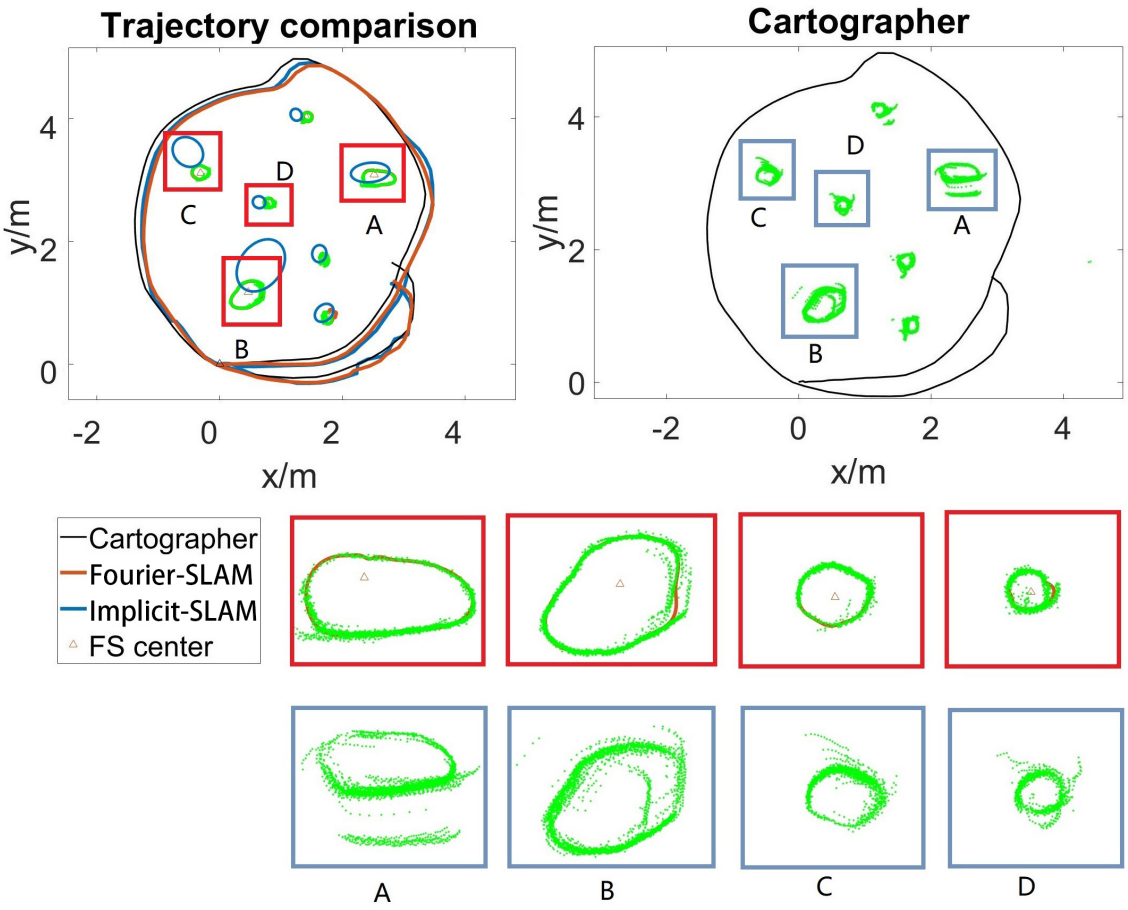


Figure 5.7 : Trajectory comparison in practical experiment. All laser points are back-projected to the global frame. The first row is the result of Fourier-SLAM, and the second row is the result of Cartographer.

Cartographer. Four rectangle areas are highlighted. By comparing from region A to region D, it is clear to see that Fourier-SLAM provides a close-to-real boundary, while Cartographer shows a larger dispersion during the end of the trajectory. Although the trajectory of Implicit-SLAM is close to Fourier-SLAM, it shows inconsistency in the bottom of the trajectory.

We also intend to illustrate the validity of Fourier-SLAM in general environment. Two experiments were conducted in this section. One is a simulated environment consisting of rectangles, irregular closed shape features and a rectangular room

boundary. The simulation properties were set the same as the first simulation. The number of Fourier series coefficients for each type of features were set differently. In this example, N of irregular closed shape features, rectangles and room boundary are chosen based on experience, which are 5, 15 and 33 respectively. The result is shown in Fig. 5.8. Clearly, the irregular features are close to the groundtruth, while the fitted rectangular features oscillate at the boundary. Nevertheless, it can still represent the feature reasonably well.

The second environment is an underground car park with width of about 50m and length of over 80m (the environment is shown in Fig. 5.6b, and the result is shown in Fig. 5.9). The robot was driven along the aisle and passed through the parking place, providing laser scan messages with a valid range of 20m and odometry messages. Since the main purpose of this part is to prove Fourier-SLAM can be used under actual application, we undertook strict data pre-processing such as manually selecting interesting features, filtering environment noise and assigning correct data association using prior information. The data was also tested via Cartographer with default settings (we did not focus on improving the performance of Cartographer by tuning parameters). It can be found that there are no observed points in the left part of the figure but the fitted wall boundary still exists, this is caused by our feature initialization strategy. The estimated boundary in the left part has no meaning since no observations occur. But the estimated boundary is close to the actual shape where the observations exist.

We use the two experiments to validate that Fourier-SLAM is able to be adapted to more general cases.

5.5 Summary

In this chapter, we proposed a novel 2D feature based SLAM approach utilizing Fourier series as feature parameterization. Compared to Implicit-SLAM, simulation

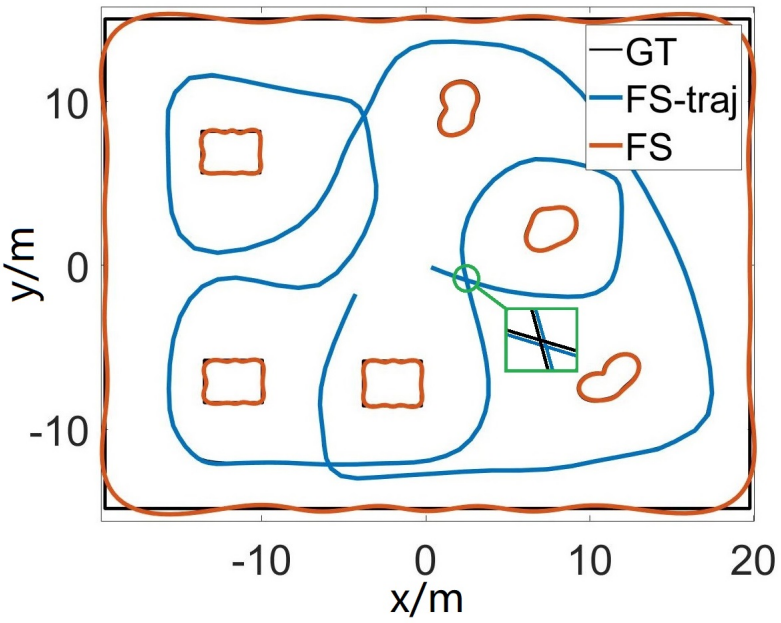


Figure 5.8 : General environment result: simulated environment. GT means groundtruth. FS-traj means the trajectory of Fourier-SLAM. FS denotes the estimated feature boundaries of Fourier-SLAM

and experimental results show that Fourier-SLAM does not rely significantly on initial guess and can provide close-to-real feature boundaries. One remaining challenge is that the data association is pre-processed, which means the observed points need to be clustered beforehand. In view of the potential of Fourier-SLAM to be applied to general scenarios, online data association is worthy of attention.

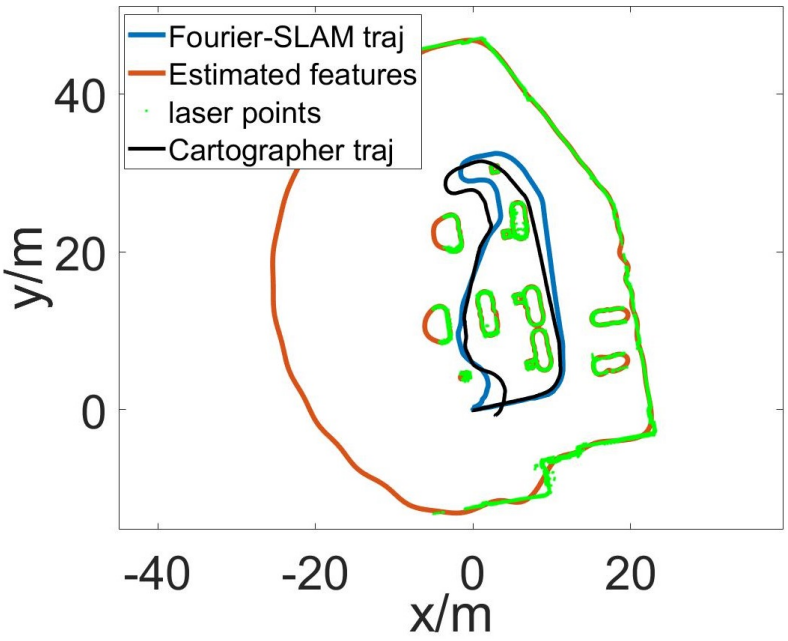


Figure 5.9 : General environment result: underground car park. The estimated boundary can represent actual features if observation is sufficient. It can also handle not-closed observed features.

Chapter 6

Improving accuracy and performance by submap joining

In the previous chapter, we proposed a Fourier series aided feature based SLAM method to conquer the problem of seeking proper parameters. This chapter focuses on another important aspect, which is the sensitivity to initial guess, and implements submap joining approach to improve the performance of Fourier-SLAM algorithm.

The main contributions of this chapter is that we formulate submap joining problem with closed shape features represented by Fourier series parameterization and develop a submap joining framework. Results demonstrate the improvement of importing submap joining method.

6.1 Problem description

Normally, closed shape feature representation method such as implicit function based method is sensitive to initial guess. Besides parameterizing features with Fourier series, another efficient way to reduce the impact of inaccurate initial guess is submap joining. Since the size of local submaps is usually small, the noisy odometry and laser information can be locally regarded as trustful, reducing the requirement of good initial guess. Moreover, submap joining method is less time-consuming than solving the problem in full least squares way.

Fig. 6.1 illustrates the process of map joining. As the odometry information is available, the observed points are projected to the start robot pose of one local submap, and then the points are re-sorted in ascending angle (w.r.t. the start robot

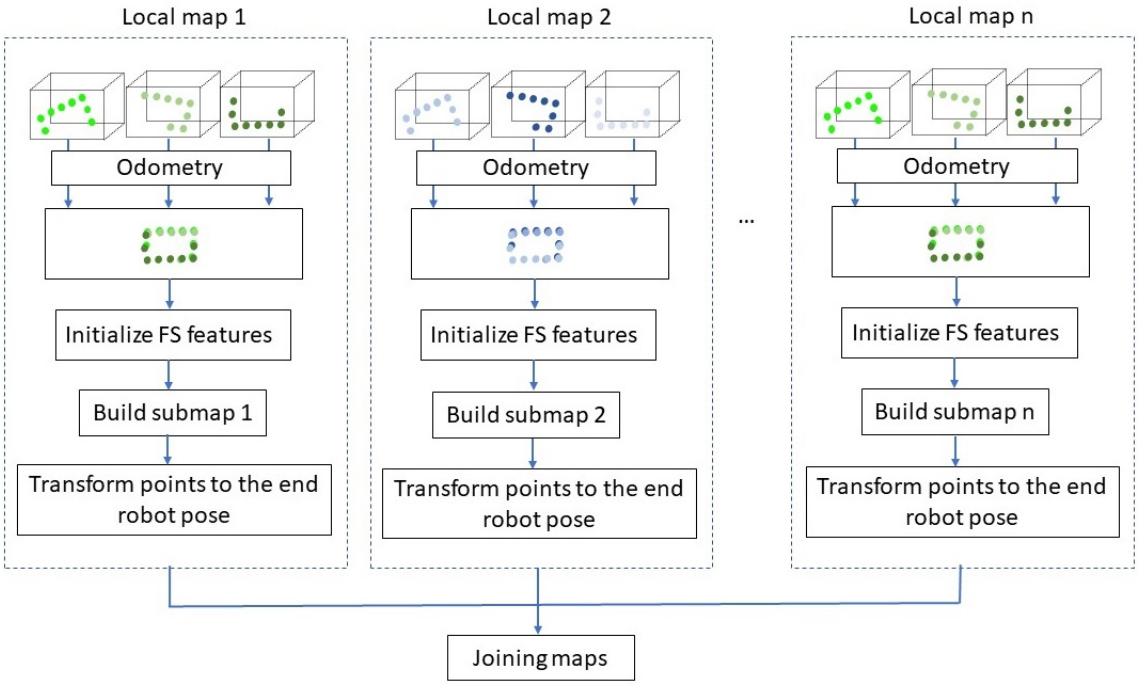


Figure 6.1 : Flow chart of submap joining process.

pose). The coefficient of Fourier series is then initialized through fitting process and then form the state vector, after which the submap building optimization will be done.

Once the submap is built, all the points are transformed to the end robot pose of this submap. Note that the calculated Fourier series coefficients in the local map are dropped from now on and all the transformed points will be kept.

After all the submaps are built, the map joining process will take into account of all the projected points and fitted centers, and then optimize global Fourier series coefficients with a global N.

6.2 Local map building process

A local map $\mathcal{L}_l$ is denoted by:

$$\mathcal{L}_l := \left\{ {}^{\{\mathcal{L}_l\}}\boldsymbol{\Xi}_e, {}^{\{\mathcal{L}_l\}}\bar{\boldsymbol{\Phi}}_1, \cdots, {}^{\{\mathcal{L}_l\}}\bar{\boldsymbol{\Phi}}_k \right\} \tag{6.1}$$

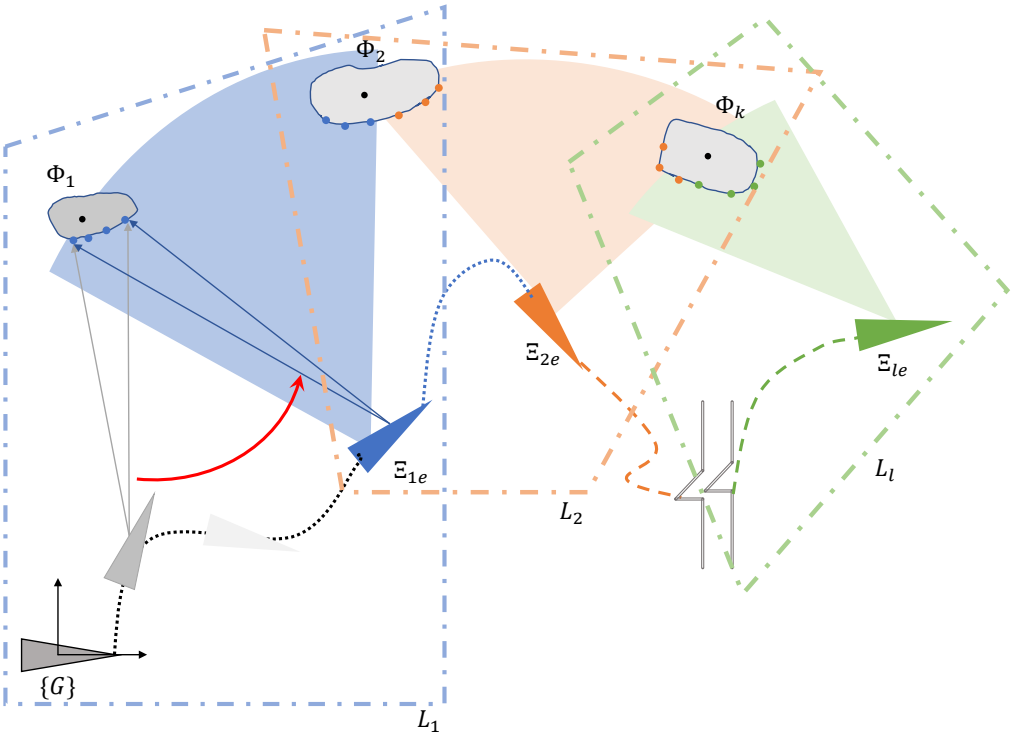


Figure 6.2 : Illustration of submap joining. The origin of the first local map $\mathcal{L}_1$ coincides with the global map frame. The robot end pose of each local map (e.g. $\mathcal{L}_1$) is the robot start pose of the next local map (e.g. $\mathcal{L}_2$). A local map is build by a series robot poses and feature observations.

where $\mathcal{L}_l$ denotes the lth local map, $\{\mathcal{L}_l\}$ is the frame of $\mathcal{L}_l$ and is defined at the first robot pose within the local map, ${}^{\{\mathcal{L}_l\}}\boldsymbol{\Xi}_e$ is the robot end pose in $\mathcal{L}_l$, ${}^{\{\mathcal{L}_l\}}\bar{\boldsymbol{\Phi}}_k$ are local features and ${}^{\{\mathcal{L}_l\}}\bar{\boldsymbol{\Phi}}_k \overset{\text{def.}}{=} [{}^{\{\mathcal{L}_l\}}\mathbf{C}_k, {}^{\{\mathcal{L}_l\}}\mathbf{P}_k]$. ${}^{\{\mathcal{L}_l\}}\mathbf{P}_k$ denotes all the points transformed to the robot end pose which are observed at each pose. Similar to Huang et al. (2008a), the robot end pose of local map $\mathcal{L}_{l-1}$ is the same as the robot start pose of local map $\mathcal{L}_l$, and the first local map starts at the same origin as the global map (as is shown in Fig. 6.2).

A new local map is built following two rules: (1) after the robot has several new observations, or (2) after the robot moves for a certain distance. The building of each local map can be regarded as a "full" SLAM problem starting at the first local

robot pose. The process can be solved via Eq. (5.9). Once all the poses and features are optimized, the estimated feature's center and laser points are transformed to the robot end pose of this local map. After the local map is built, all the poses except the robot end pose are removed. The results of one local map consist of transformed points, feature centers, and the robot end pose.

6.3 Map joining process

After all the local maps are built, submap joining process will optimize the robot end poses and features by considering all the local maps as integrated observations. Consider a simplified case: l local maps are built containing K features. The state vector $\mathbf{X}^{\mathcal{G}}$ in submap joining process is defined as

$$\mathbf{X}^{\mathcal{G}} = [\boldsymbol{\Xi}_{1e}; \cdots; \boldsymbol{\Xi}_{le}; \boldsymbol{\Phi}_1; \cdots; \boldsymbol{\Phi}_K] \tag{6.2}$$

where $\boldsymbol{\Xi}_{1e}, \cdots, \boldsymbol{\Xi}_{le}$ are the global robot end poses of each local maps; $\boldsymbol{\Phi}_1, \cdots, \boldsymbol{\Phi}_K$ are global features.

The aim of submap joining is to obtain the global map by merging and optimizing local maps. The optimization problem is:

$$\underset{\mathbf{X}^{\mathcal{G}}}{\operatorname{argmin}}\ M(\mathbf{X}^{\mathcal{G}}) = \sum_{\boldsymbol{\Xi}_{je}, j=1}^{l} \left(\tilde{E}_{\mathrm{o},je} + \sum_{k=1}^{K} \left(\tilde{E}_{\mathrm{f},j,k} + \tilde{E}_{\mathrm{c},j,k} \right) \right) \tag{6.3}$$

where

$$\begin{aligned} \tilde{E}_{\mathrm{o},je} &= \left\| f(\mathbf{Z}_{\mathrm{o},je}, \boldsymbol{\Xi}_{(j-1)e}, \boldsymbol{\Xi}_{je}) \right\|^2_{\tilde{\Sigma}^{-1}_{\mathrm{o},je}} \\ \tilde{E}_{\mathrm{f},j,k} &= \left\| d(\boldsymbol{\theta}^{jk}) - \mathbf{r}^{jk} \right\|^2_{\tilde{\Sigma}^{-1}_{\mathrm{f},j,k}} \\ \tilde{E}_{\mathrm{c},j,k} &= \left\| T^{-1}(\boldsymbol{\Xi}_{je}, \mathbf{C}_k) - {}^{\{j\}}\mathbf{C}_k \right\|^2_{\tilde{\Sigma}^{-1}_{\mathrm{c},j,k}} \end{aligned} \tag{6.4}$$

Noting that $\mathbf{Z}_{\mathrm{o},je}$ is the estimated robot end pose ${}^{\{\mathcal{L}_j\}}\boldsymbol{\Xi}_e$ of local map $\mathcal{L}_j$, and $f(\cdot)$ is under the same definition as Eq. (5.10). $\boldsymbol{\theta}^{jk}$ and $\boldsymbol{r}^{jk}$ are defined similarly to Eq. (5.11) with respect to the jth robot end pose. Since the calculation of covariance

matrices is quite similar to computing the covariance matrices in Eq. (5.10), we omit the specific derivation in this chapter.

A larger N in building local maps and a smaller N in submap joining process is helpful during submap joining. The accumulated error caused by sensor noises does not influence results significantly because of the limited size of poses when building local maps. Hence, a larger N contributes more precise feature boundaries and more accurate local maps. On the other hand, joining maps has to handle all the points corresponding to every feature from each local map, where the accumulated error is really large. In this situation, a smaller N can lower the impact of noisy points both in fitting boundaries and solving the optimization problem. Moreover, a smaller N can also reduce the calculation time because of the decreased size of state vectors.

6.4 Experiment and analysis

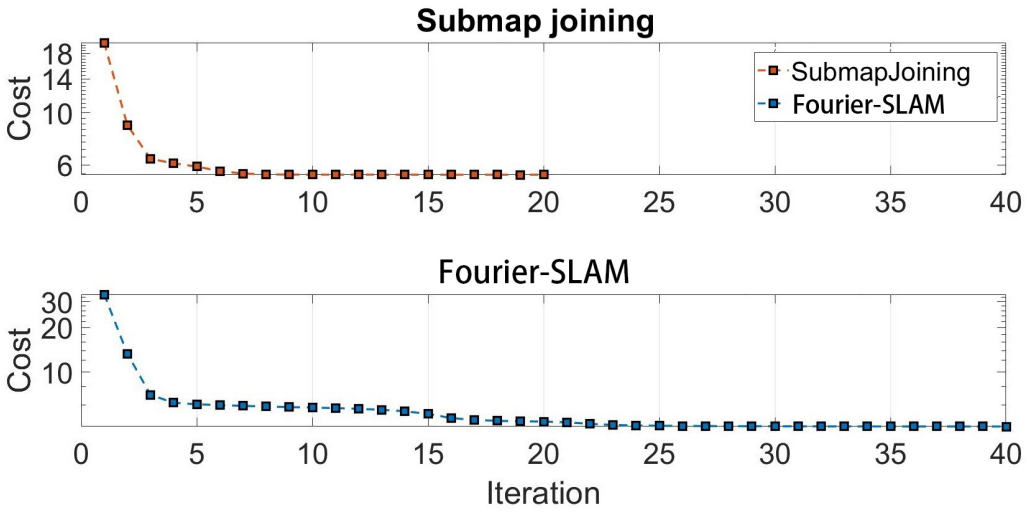


Figure 6.3 : Cost value changes with iteration. The iteration number of Fourier-SLAM is 148 and truncated at 40. Submap joining stops after 20 iterations. The Y axis is scaled by logarithm operation. It is clear that Fourier-SLAM continues iterating from 5 to 16 with slight changes.

In this part, we test the submap joining on the synthetic experiment and practical experiment scenario A. A local map is built every 5 valid sequences (a sequence is regarded valid only if the odometry increment is beyond a threshold. In this chapter, the thresholds for translation and rotation are set to 0.1m and 2°). Within each local map, the accumulated error caused by odometry noise is small, so the odometry information can be approximately regarded as accurate. As a result, the algorithm can directly import odometry as the initial guess without importing unexpected local minima.

We evaluate submap joining method and Fourier-SLAM in two aspects: map performance and time consumption. Both methods are provided the same odometry information as initial guess and share the same weights. Both methods run 50 times with different random noise at each run to compare average runtime.

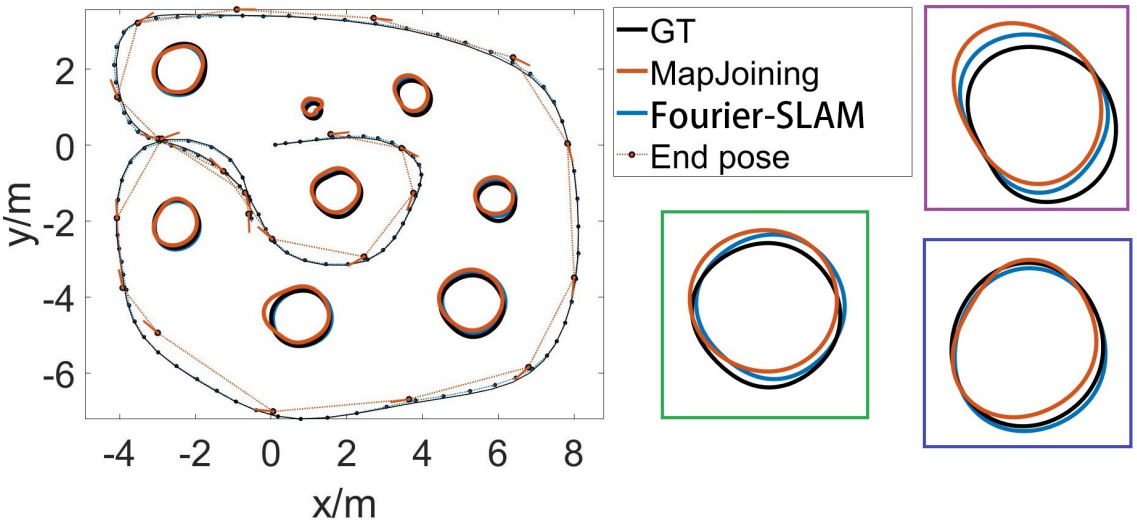


Figure 6.4 : Submap joining. Each point-line marker denotes the end pose of the local map with respect to the global frame.

The results of one run are shown in Fig. 6.4. In contrast to the result of Fourier-SLAM, the estimated features of submap joining undergo subtle changes against the groundtruth features, while the Fourier-SLAM is closer to the groundtruth.

However, submap joining process wins considering time consumption. According to the calculation time of the last step, the average runtime among 50 times experiments of Fourier-SLAM is 17.133 seconds, while the submap joining method is 7.3657 seconds. Noting here the runtime of submap joining does not include the processing time of building local maps, the reason is that local maps are already built with time increasing.

We also compared the cost function value changes w.r.t. iteration number in Fig. 6.3. It can be found that Fourier-SLAM does not descends as quickly as submap joining, and Fourier-SLAM needs more iterations to get converged, which proves that submap joining effectively improve the calculation performance.

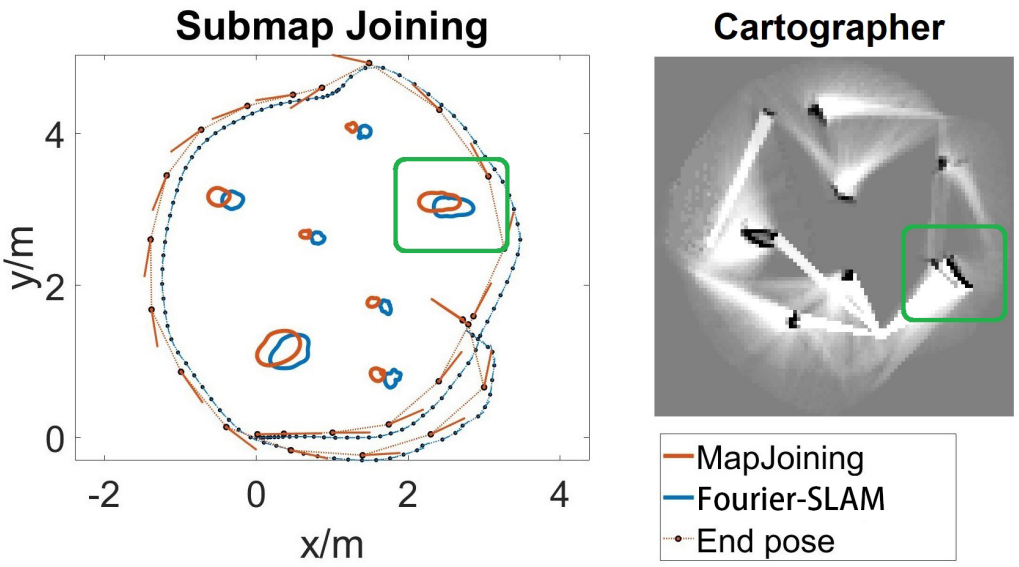


Figure 6.5 : Submap joining result and the map from Cartographer.

Fig. 6.5 shows the map of submap joining, Fourier-SLAM and Cartographer, respectively. Compared with submap joining and Fourier-SLAM, submap joining method provides a good feature boundary even if its trajectory drifts slightly, and the runtime of submap joining is less than that of Fourier-SLAM. The map of Cartographer is also displayed on the right. In the circled area, it can be found the map

of Cartographer is ambiguous.

6.5 Summary

In this chapter, we formulated a submap joining method based on Fourier-SLAM. Compared to original Fourier-SLAM, synthetic and practical experiments show that submap joining method is able to speed up the calculation without leading to an unacceptable result.

Chapter 7

Conclusion

The Simultaneous Localization and Mapping technology for mobile robots is an important embodiment of robots' intelligent and autonomous. As the most widely used sensor in SLAM application, 2D lidar is often one of the necessary equipment for mobile robots, and related algorithms and researches become inevitable. Different from conventional SLAM methods that directly utilizing discrete points information, feature based SLAM approaches can better integrate geometric information of the environment. In this thesis, we studied 2D general feature SLAM algorithms, proposing pre-fit method and post-count methods.

7.1 Summary of the contributions

The contributions of this thesis are as follows.

7.1.1 Propose Pre-fit SLAM

The open environment usually has large scales and lacks of regular geometric information, where observations from mobile robots could be discontinuous. Aiming at the problem of lacking of regular linear geometric features in the open environment, we proposed a pre-fit feature based SLAM method taking advantage of ellipse features. The original point cloud data is parameterized by ellipse information and ranging information, and the problem is redefined by the parameterized features. At last the factor graph framework is used for the final optimization solution. The experimental results show that the method is superior to the most advanced methods and can ensure that the robot works normally in an open environment. And com-

pared with the traditional occupancy grid map, the use of parameterized features for map expression occupies less resources, and the map representation is clearer.

7.1.2 Propose Implicit-SLAM

Aiming at problems that the general environment is not conducive to explicit feature expression and the way of pre-fitting features causes a large amount of information to be lost, a deeper theoretical study is carried out on the feature-based SLAM. A post-count theoretical framework aided by implicit function is proposed. The implicit function is used to define the characteristics of features with any shapes, and the corresponding objective optimization equation is constructed accordingly. On the basis of the corresponding lemmas derived from the research, we can get the inner connection between the uncertainty of the original data and the expression characteristics of the redefined implicit equation. At the same time, the implicit function expression of the closed shape feature is improved to ensure the convergence performance when the virtual observation falls inside the closed shape boundary during the iteration process. The experiment demonstrated the feasibility and potential value of this SLAM framework with implicit function expression characteristics.

7.1.3 Propose Fourier-SLAM

The post-count method with implicit functions are able to express general closed shape features. However, a sensitive initial guess should be provided to the problem. To conquer it, we proposed to represent closed shape features with Fourier series, and raised a post-count approach implementing Fourier series. Similar to but different from implicit functions, the method via Fourier series imports the concept of "center" and converts the distance of the boundary from Euclidean space to frequency domain. The final robot poses are then optimized by constructing constrains of Fourier series, centers and robot poses. Compared to the post-count method using implicit function, simulated and practical experiments concluded that using Fourier

series does not rely significantly on initial guess and can provide close-to-real feature boundaries.

7.1.4 Propose submap joining method for Fourier-SLAM

In order to further reduce the sensitivity to the initial value and speed up the calculation, we studied the feasibility of submap joining. When the robot moves in a short period of time, the odometer information can be approximately regarded as accurate. On this basis, we proposed a submap joining method and formulated equations of the method with the Fourier series parameterization. Synthetic and practical experiments show that by joining submaps the iteration number during optimization is decreased significantly, while the relative pose error is not increased too much.

7.2 Future works

Nowadays, the value of mobile robots has been increasingly reflected as an important carrier of artificial intelligence. Combining the research content of this thesis and the problems encountered in the work, further research can be carried out in the following aspects.

7.2.1 Points clustering

In the theoretical study of post-count method, we studied implicit functions and Fourier series to help solve the SLAM problem. One actual problem during applying our methods to realistic scenario is the data association. Current way to associate features is implemented based on the odometry information, which means we need to know the number of features in advance and the size of features cannot be too large. One of the future works is aimed to study on automatically clustering method to tackle this problem.

7.2.2 Registration by features

Another potential step is to study the probability of point registration via feature parameters directly. At present, in addition to assuming the data association is known, we also rely on the point cloud registration method. If the registration are done with the help of feature parameterization, the result will be improved.

7.2.3 Multi-sensor fusion

The mobile robots working in 2D are the most widely used and the most need of improved performance robots. A lidar-only system cannot meet the demands in multiple situations. In the future work, merging lidar with other sensors, such as IMU and camera, will be our important research direction.

Bibliography

Ahmad, A., Huang, S., Wang, J. J., and Dissanayake, G. (2012). A new state vector and a map joining algorithm for range-only slam. In *2012 12th International Conference on Control Automation Robotics & Vision (ICARCV)*, pages 1024–1029. IEEE.

Ahn, S. J., Rauh, W., and Recknagel, M. (1999). Ellipse fitting and parameter assessment of circular object targets for robot vision. In *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No. 99CH36289)*, volume 1, pages 525–530. IEEE.

Aloise, I., Della Corte, B., Nardi, F., and Grisetti, G. (2019). Systematic handling of heterogeneous geometric primitives in graph-slam optimization. *IEEE Robotics and Automation Letters*, 4(3):2738–2745.

Aulinas, J., Lladó, X., Salvi, J., and Petillot, Y. R. (2010a). Selective submap joining for underwater large scale 6-dof slam. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2552–2557. IEEE.

Aulinas, J., Lladó, X., Salvi, J., and Petillot, Y. R. (2010b). Slam based selective submap joining for the victoria park dataset. *IFAC Proceedings Volumes*, 43(16):557–562.

Bachrach, A., Prentice, S., He, R., Henry, P., Huang, A. S., Krainin, M., Maturana, D., Fox, D., and Roy, N. (2012). Estimation, planning, and mapping for

autonomous flight using an rgb-d camera in gps-denied environments. *The International Journal of Robotics Research*, 31(11):1320–1343.

Badue, C., Guidolini, R., Carneiro, R. V., Azevedo, P., Cardoso, V. B., Forechi, A., Jesus, L., Berriel, R., Paixão, T., Mutz, F., et al. (2019). Self-driving cars: A survey. *arXiv preprint arXiv:1901.04407*.

Bailey, T. and Durrant-Whyte, H. (2006). Simultaneous localization and mapping (slam): Part ii. *IEEE robotics & automation magazine*, 13(3):108–117.

Bailey, T., Nieto, J., and Nebot, E. (2006). Consistency of the fastslam algorithm. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 424–429. IEEE.

Behley, J. and Stachniss, C. (2018). Efficient surfel-based slam using 3d laser range data in urban environments. In *Robotics: Science and Systems*.

Bergström, P. and Edlund, O. (2014). Robust registration of point sets using iteratively reweighted least squares. *Computational optimization and applications*, 58(3):543–561.

Besl, P. J. and McKay, N. D. (1992). Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics.

Biber, P. and Straßer, W. (2003). The normal distributions transform: A new approach to laser scan matching. In *IROS*, volume 3, pages 2743–2748.

Bry, A., Bachrach, A., and Roy, N. (2012). State estimation for aggressive flight in gps-denied environments using onboard sensing. In *2012 IEEE International Conference on Robotics and Automation*, pages 1–8. IEEE.

Bry, A., Richter, C., Bachrach, A., and Roy, N. (2015). Aggressive flight of fixed-wing and quadrotor aircraft in dense indoor environments. *The International Journal of Robotics Research*, 34(7):969–1002.

Burguera Burguera, A. and Bonin-Font, F. (2019). A trajectory-based approach to multi-session underwater visual slam using global image signatures. *Journal of Marine Science and Engineering*, 7(8):278.

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., and Leonard, J. J. (2016). Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332.

Chen, L.-H. and Peng, C.-C. (2019). A robust 2d-slam technology with environmental variation adaptability. *IEEE Sensors Journal*, 19(23):11475–11491.

Chen, Y., Huang, S., Fitch, R., and Yu, J. (2018). Efficient active slam based on submap joining, graph topology and convex optimization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE.

Cieslewski, T. and Scaramuzza, D. (2017). Efficient decentralized visual place recognition using a distributed inverted index. *IEEE Robotics and Automation Letters*, 2(2):640–647.

Davison, A. J. (2003). Real-time simultaneous localisation and mapping with a single camera. In *null*, page 1403. IEEE.

de la Puente, P. and Rodríguez-Losada, D. (2014). Feature based graph-slam in structured environments. *Autonomous Robots*, 37(3):243–260.

Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110.

Endres, F., Hess, J., Sturm, J., Cremers, D., and Burgard, W. (2013). 3-d mapping with an rgb-d camera. *IEEE transactions on robotics*, 30(1):177–187.

Folkesson, J. and Christensen, H. I. (2007). Graphical slam for outdoor applications. *Journal of Field Robotics*, 24(1-2):51–70.

Forster, C., Zhang, Z., Gassner, M., Werlberger, M., and Scaramuzza, D. (2016). Svo: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*, 33(2):249–265.

Gargoum, S. A. and El Basyouny, K. (2019). A literature synthesis of lidar applications in transportation: feature extraction and geometric assessments of highways. *GIScience & Remote Sensing*, pages 1–30.

Gee, T., James, J., Van Der Mark, W., Delmas, P., and Gimel'farb, G. (2016). Lidar guided stereo simultaneous localization and mapping (slam) for uav outdoor 3-d scene reconstruction. In *2016 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, pages 1–6. IEEE.

Grisetti, G., Grzonka, S., Stachniss, C., Pfaff, P., and Burgard, W. (2007). Efficient estimation of accurate maximum likelihood maps in 3d. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3472–3478. IEEE.

Grisettiyz, G., Stachniss, C., and Burgard, W. (2005). Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 2432–2437. IEEE.

Guivant, J. and Nebot, E. (2002). Simultaneous localization and map building: Test case for outdoor applications. In *IEEE Int. Conference on Robotics and Automation*.

Guivant, J., Nebot, E., and Durrant-Whyte, H. (2000). Simultaneous localization and map building using natural features in outdoor environments. In *Intelligent Autonomous Systems*, volume 6, pages 581–586.

Guo, Z., Cai, B., Jiang, W., and Wang, J. (2019). Feature-based detection and classification of moving objects using lidar sensor. *IET Intelligent Transport Systems*.

Henry, P., Krainin, M., Herbst, E., Ren, X., and Fox, D. (2012). Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The International Journal of Robotics Research*, 31(5):647–663.

Hess, W., Kohler, D., Rapp, H., and Andor, D. (2016). Real-time loop closure in 2d lidar slam. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1271–1278. IEEE.

Holỳ, B. (2016). Registration of lines in 2d lidar scans via functions of angles. *IFAC-PapersOnLine*, 49(5):109–114.

Hu, M., Ao, H., and Jiang, H. (2019). Experimental research on feature extraction of laser slam based on artificial landmarks. In *2019 Chinese Control And Decision Conference (CCDC)*, pages 5495–5500. IEEE.

Huang, S. and Dissanayake, G. (2007). Convergence and consistency analysis for extended kalman filter based slam. *IEEE Transactions on robotics*, 23(5):1036–1049.

Huang, S., Wang, H., Frese, U., and Dissanayake, G. (2012). On the number of local minima to the point feature based slam problem. In *2012 IEEE International Conference on Robotics and Automation*, pages 2074–2079. IEEE.

Huang, S., Wang, Z., and Dissanayake, G. (2008a). Sparse local submap joining filter for building large-scale maps. *IEEE Transactions on Robotics*.

Huang, S., Wang, Z., Dissanayake, G., and Frese, U. (2008b). Iterated slsjf: A sparse local submap joining algorithm with improved consistency. In *Proceedings of the 2008 Australasian Conference on Robotics and Automation, ACRA 2008*.

Ji, K., Chen, H., Di, H., Gong, J., Xiong, G., Qi, J., and Yi, T. (2018). Cpfg-slam: a robust simultaneous localization and mapping based on lidar in off-road environment. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 650–655. IEEE.

Jian, B. and Vemuri, B. C. (2011). Robust point set registration using gaussian mixture models. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1633–1645.

Jiang, G., Yin, L., Liu, G., Xi, W., and Ou, Y. (2019a). Fft-based scan-matching for slam applications with low-cost laser range finders. *Applied Sciences*, 9(1):41.

Jiang, J., Wang, J., Wang, P., and Chen, Z. (2019b). Pou-slam: Scan-to-model matching based on 3d voxels. *Applied Sciences*, 9(19):4147.

Kaess, M., Ranganathan, A., and Dellaert, F. (2008). isam: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378.

Kim, S. and Oh, S.-Y. (2008). Slam in indoor environments using omni-directional vertical and horizontal line features. *Journal of Intelligent and Robotic Systems*, 51(1):31–43.

Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, pages 225–234. IEEE.

Kohlbrecher, S., Meyer, J., von Stryk, O., and Klingauf, U. (2011a). A flexible and scalable slam system with full 3d motion estimation. In *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE.

Kohlbrecher, S., Von Stryk, O., Meyer, J., and Klingauf, U. (2011b). A flexible and scalable slam system with full 3d motion estimation. In *2011 IEEE international symposium on safety, security, and rescue robotics*, pages 155–160. IEEE.

Konolige, K., Grisetti, G., Kümmerle, R., Burgard, W., Limketkai, B., and Vincent, R. (2010). Efficient sparse pose adjustment for 2d mapping. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 22–29. IEEE.

Labbé, M. and Michaud, F. (2019). Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, 36(2):416–446.

Li, J., Zhong, R., Hu, Q., and Ai, M. (2016). Feature-based laser scan matching and its application for indoor mapping. *Sensors*, 16(8):1265.

Li, L., Liu, J., Zuo, X., and Zhu, H. (2018). An improved mbicp algorithm for mobile robot pose estimation. *Applied Sciences*, 8(2):272.

Liu, M., Huang, S., and Dissanayake, G. (2011). Feature based slam using laser sensor data with maximized information usage. In *2011 IEEE International Conference on Robotics and Automation*, pages 1811–1816. IEEE.

Mahon, I., Williams, S. B., Pizarro, O., and Johnson-Roberson, M. (2008). Efficient view-based slam using visual loop closures. *IEEE Transactions on Robotics*, 24(5):1002–1014.

Marjanovic, G. and Solo, V. (2012). On l_q optimization and matrix completion. *IEEE Transactions on signal processing*, 60(11):5714–5724.

Marjanovic, G. and Solo, V. (2014). $l\_\{q\}$ sparsity penalized linear regression with cyclic descent. *IEEE Transactions on Signal Processing*, 62(6):1464–1475.

Martín, F., Triebel, R., Moreno, L., and Siegwart, R. (2014). Two different tools for three-dimensional mapping: De-based scan matching and feature-based loop detection. *Robotica*, 32(1):19–41.

Mur-Artal, R., Montiel, J. M. M., and Tardos, J. D. (2015). Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163.

Mur-Artal, R. and Tardós, J. D. (2017). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262.

Murphy, K. and Russell, S. (2001). Rao-blackwellised particle filtering for dynamic bayesian networks. In *Sequential Monte Carlo methods in practice*, pages 499–515. Springer.

Newcombe, R. A. and Davison, A. J. (2010). Live dense reconstruction with a single moving camera. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1498–1505. IEEE.

Nguyen, V., Martinelli, A., Tomatis, N., and Siegwart, R. (2005). A comparison of line extraction algorithms using 2d laser rangefinder for indoor mobile robotics. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1929–1934. IEEE.

Ni, K., Steedly, D., and Dellaert, F. (2007). Tectonic sam: Exact, out-of-core, submap-based slam. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 1678–1685. IEEE.

Okatani, T. and Deguchi, K. (2009). On bias correction for geometric parameter estimation in computer vision. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 959–966. IEEE.

Olson, E. (2015). M3rsm: Many-to-many multi-resolution scan matching. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5815–5821. IEEE.

Olson, E., Leonard, J., and Teller, S. (2006). Fast iterative alignment of pose graphs with poor initial estimates. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2262–2269. IEEE.

Pedraza, L., Rodriguez-Losada, D., Matia, F., Dissanayake, G., and Miró, J. V. (2009). Extending the limits of feature-based slam with b-splines. *IEEE Transactions on Robotics*, 25(2):353–366.

Pedrosa, E., Pereira, A., and Lau, N. (2017). Efficient localization based on scan matching with a continuous likelihood field. In *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 61–66. IEEE.

Puhan, N. B., Sudha, N., and Kaushalram, A. S. (2011). Efficient segmentation technique for noisy frontal view iris images using fourier spectral density. *Signal, Image and Video Processing*, 5(1):105–119.

Qin, T., Li, P., and Vins-mono, S. S. (2017). A robust and versatile monocular visual-inertial state estimator. *arXiv preprint arXiv: 1708.03852*.

Rakshit, S. and Monro, D. M. (2007). Pupil shape description using fourier series. In *2007 IEEE Workshop on Signal Processing Applications for Public Security and Forensics*, pages 1–4. IEEE.

Rao, D., Chung, S.-J., and Hutchinson, S. (2012). Curveslam: An approach for vision-based navigation without point features. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4198–4204. IEEE.

Ren, R., Fu, H., and Wu, M. (2019). Large-scale outdoor slam based on 2d lidar. *Electronics*, 8(6):613.

Rubio, F., Valero, F., and Llopis-Albert, C. (2019). A review of mobile robots: Concepts, methods, theoretical framework, and applications. *International Journal of Advanced Robotic Systems*, 16(2):1729881419839596.

Sehgal, A., Singandhupe, A., La, H. M., Tavakkoli, A., and Louis, S. J. (2019). Lidar-monocular visual odometry with genetic algorithm for parameter optimization. In *International Symposium on Visual Computing*, pages 358–370. Springer.

Sharp, G. C., Lee, S. W., and Wehe, D. K. (2002). Icp registration using invariant features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):90–102.

Shi, J. et al. (1994). Good features to track. In *1994 Proceedings of IEEE conference on computer vision and pattern recognition*, pages 593–600. IEEE.

Song, J., Wang, J., Zhao, L., Huang, S., and Dissanayake, G. (2017). Dynamic reconstruction of deformable soft-tissue with stereo scope in minimal invasive surgery. *IEEE Robotics and Automation Letters*, 3(1):155–162.

Song, J., Wang, J., Zhao, L., Huang, S., and Dissanayake, G. (2018). Mis-slam: Real-time large-scale dense deformable slam system in minimal invasive surgery based on heterogeneous computing. *IEEE Robotics and Automation Letters*, 3(4):4068–4075.

Steux, B. and El Hamzaoui, O. (2010). tinyslam: A slam algorithm in less than 200 lines c-language program. In *2010 11th International Conference on Control Automation Robotics & Vision*, pages 1975–1979. IEEE.

Stoyanov, T., Magnusson, M., Andreasson, H., and Lilienthal, A. J. (2012). Fast

and accurate scan registration through minimization of the distance between compact 3d ndt representations. *The International Journal of Robotics Research*, 31(12):1377–1393.

Su, D. and Xiang, W. (2020). Characterization and regeneration of 2d general-shape particles by a fourier series-based approach. *Construction and Building Materials*, 250:118806.

Sun, Z., Wu, B., Xu, C.-Z., Sarma, S. E., Yang, J., and Kong, H. (2020). Frontier detection and reachability analysis for efficient 2d graph-slam based active exploration. *arXiv preprint arXiv:2009.02869*.

Tang, T. Y., Yoon, D. J., and Barfoot, T. D. (2019). A white-noise-on-jerk motion prior for continuous-time trajectory estimation on se (3). *IEEE Robotics and Automation Letters*, 4(2):594–601.

Walter, M. R., Eustice, R. M., and Leonard, J. J. (2007). Exactly sparse extended information filters for feature-based slam. *The International Journal of Robotics Research*, 26(4):335–359.

Wang, C., Shu, Q., Wang, X., Guo, B., Liu, P., and Li, Q. (2019a). A random forest classifier based on pixel comparison features for urban lidar data. *ISPRS journal of photogrammetry and remote sensing*, 148:75–86.

Wang, J., Song, J., Zhao, L., Huang, S., and Xiong, R. (2019b). A submap joining algorithm for 3d reconstruction using an rgb-d camera based on point and plane features. *Robotics and Autonomous Systems*, 118:93–111.

Wilson, G., Pereyda, C., Raghunath, N., de la Cruz, G., Goel, S., Nesaei, S., Minor, B., Schmitter-Edgecombe, M., Taylor, M. E., and Cook, D. J. (2019). Robot-enabled support of daily activities in smart home environments. *Cognitive Systems Research*, 54:258–272.

Wise, M., Ferguson, M., King, D., Diehr, E., and Dymesich, D. (2016). Fetch and freight: Standard platforms for service robot applications. In *Workshop on autonomous mobile service robots*.

Wu, D., Meng, Y., Zhan, K., and Ma, F. (2018). A lidar slam based on point-line features for underground mining vehicle. In *2018 Chinese Automation Congress (CAC)*, pages 2879–2883. IEEE.

Yang, J., Li, H., and Jia, Y. (2013). Go-icp: Solving 3d registration efficiently and globally optimally. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1457–1464.

Yin, J., Carlone, L., Rosa, S., Anjum, M. L., and Bona, B. (2014). Scan matching for graph slam in indoor dynamic scenarios. In *The Twenty-Seventh International Flairs Conference*.

Zhan, Z., Jian, W., Li, Y., Wang, X., and Yue, Y. (2020). A slam map restoration algorithm based on submaps and an undirected connected graph. *arXiv preprint arXiv:2007.14592*.

Zhang, C., Yong, L., Chen, Y., Zhang, S., Ge, L., Wang, S., and Li, W. (2019). A rubber-tapping robot forest navigation and information collection system based on 2d lidar and a gyroscope. *Sensors*, 19(9):2136.

Zhang, D. and Lu, G. (2001). Shape retrieval using fourier descriptors. In *In Proceedings of 2nd IEEE Pacific Rim Conference on Multimedia*. Citeseer.

Zhang, F., Li, S., Yuan, S., Sun, E., and Zhao, L. (2017a). Algorithms analysis of mobile robot slam based on kalman and particle filter. In *2017 9th International Conference on Modelling, Identification and Control (ICMIC)*, pages 1050–1055. IEEE.

Zhang, H., Hasith, K., and Wang, H. (2017b). A hybrid feature parametrization for improving stereo-slam consistency. In *2017 13th IEEE International Conference on Control & Automation (ICCA)*, pages 1021–1026. IEEE.

Zhang, J. and Singh, S. (2014). Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2.

Zhang, L. and Ghosh, B. K. (2000). Line segment based map building and localization using 2d laser rangefinder. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 3, pages 2538–2543. IEEE.

Zhang, Z. (1997). Parameter estimation techniques: A tutorial with application to conic fitting. *Image and vision Computing*, 15(1):59–76.

Zhao, J., Huang, S., and Zhao, L. (2018). Constrained gaussian mixture models based scan matching method. In *Australasian Conference on Robotics and Automation*.

Zhao, J., Huang, S., Zhao, L., Chen, Y., and Luo, X. (2019). Conic feature based simultaneous localization and mapping in open environment via 2d lidar. *IEEE Access*, 7:173703–173718.

Zhao, L., Huang, S., Yan, L., and Dissanayake, G. (2015). A new feature parametrization for monocular slam using line features. *Robotica*, 33(3):513–536.

Zhao, L., Huang, S., Yan, L., Wang, J. J., Hu, G., and Dissanayake, G. (2010). Large-scale monocular slam by local bundle adjustment and map joining. In *2010 11th International Conference on Control Automation Robotics & Vision*, pages 431–436. IEEE.