Development of an adaptive navigation system for indoor mobile handling and manipulation platforms

M.Sc. Qiucheng Li

Supervisor: Prof. Dr.-Ing. habil. Thomas Rauschenbach

Department of Computer Science and Automation
Ilmenau University of Technology

This dissertation is submitted for the degree of
Doktor-Ingenieur (Dr.-Ing.)

October 17th, 2018

Development of an adaptive navigation system for indoor mobile handling and manipulation platforms

Dissertation zur Erlangung des
akademischen Grades Doktor-Ingenieur (Dr.-Ing.)

vorgelegt der Fakultät Informatik und Automatisierung
der Technischen Universität Ilmenau

von M.Sc. Qiucheng Li

1. Gutachter: Prof. Dr.-Ing. habil. Thomas Rauschenbach
   (TU Ilmenau, Institutsteil Angewandte Systemtechnik (AST) des Fraunhofer IOSB)
2. Gutachter: Univ.Prof. Dr.-Ing. habil. Pu Li
   (TU Ilmenau)
3. Gutachter: Prof. Dr.-Ing. Rolf Findeisen
   (Otto-von-Guericke-Universität Magdeburg)

Tag der Einreichung: 17.10.2018
Tag der wissenschaftlichen Aussprache: 12.04.2019

urn:nbn:de:gbv:ilm1-2019000090

I would like to dedicate this thesis to my loving wife, my son and my parents for their support. Thanks to Prof. Thomas Rauschenbach and Prof. Andreas Wenzel, they have always provided insights and advice on my work. Thanks to Dr. Fabian Müller, with whom I have lots of interesting and fruitful discussions. I would also like to thank my friendly colleagues at Fraunhofer IOSB-AST, they have provided plenty of help in my work and my life in Germany. Thanks to the reviewers of my thesis. Thanks to China Scholarship Council for the financial support, and thanks to Prof. Zhenbo Li and Prof. Yingyi Chen as the surety.

Acknowledgements

I would like to acknowledge China Scholarship Council (CSC) for the financial support from 09.2012 till 09.2016.

Abstract

A fundamental technology enabling the autonomous behavior of mobile robotics is navigation. It is a main prerequisite for mobile robotics to fulfill high-level tasks such as handling and manipulation, and is often identified as one of the key challenges in mobile robotics. The mapping and localization as the basis for navigation are intensively researched in the last few decades. However, there are still challenges or problems needed to be solved for online operating in large-scale environments or running on low-cost and energy-saving embedded systems.

In this work, new developments and usages of Light Detection And Ranging (LiDAR) based Simultaneous Localization And Mapping (SLAM) algorithms are presented. A key component of LiDAR based SLAM algorithms, the scan matching algorithm, is explored. Different scan matching algorithms are systemically experimented with different LiDARs for indoor home-like environments for the first time. The influence of properties of LiDARs in scan matching algorithms is quantitatively analyzed. Improvements to Bayes filter based and graph optimization based SLAMs are presented. The Bayes filter based SLAMs mainly use the current sensor information to find the best estimation. A new efficient implementation of Rao-Blackwellized Particle Filter based SLAM is presented. It is based on a pre-computed lookup table and the parallelization of the particle updating. The new implementation runs efficiently on recent multi-core embedded systems that fulfill low cost and energy efficiency requirements. In contrast to Bayes filter based methods, graph optimization based SLAMs utilize all the sensor information and minimize the total error in the system. A new real-time graph building model and a robust integrated Graph SLAM solution are presented. The improvements include the definition of unique direction norms for points or lines extracted from scans, an efficient loop closure detection algorithm, and a parallel and adaptive implementation. The developed algorithm outperforms the state-of-the-art algorithms in processing time and robustness especially in large-scale environments using embedded systems instead of high-end computation devices.

The results of the work can be used to improve the navigation system of indoor autonomous robots, like domestic environments and intra-logistics.

Zusammenfassung

Eine der grundlegenden Funktionen, welche die Autonomie in der mobilen Robotik ermöglicht, ist die Navigation. Sie ist eine wesentliche Voraussetzung dafür, dass mobile Roboter selbständig anspruchsvolle Aufgaben erfüllen können. Die Umsetzung der Navigation wird dabei oft als eine der wichtigsten Herausforderungen identifiziert. Die Kartenerstellung und Lokalisierung als Grundlage für die Navigation wurde in den letzten Jahrzehnten intensiv erforscht. Es existieren jedoch immer noch eine Reihe von Problemen, z.B. die Anwendung auf große Areale oder bei der Umsetzung auf kostengünstigen und energiesparenden Embedded-Systemen.

Diese Arbeit stellt neue Ansätze und Lösungen im Bereich der LiDAR-basierten simultanen Positionsbestimmung und Kartenerstellung (SLAM) vor. Eine Schlüsselkomponente der LiDAR-basierten SLAM, die so genannten Scan-Matching-Algorithmen, wird näher untersucht. Verschiedene Scan-Matching-Algorithmen werden zum ersten Mal systematisch mit verschiedenen LiDARs für den Innenbereich getestet. Der Einfluss von LiDARs auf die Eigenschaften der Algorithmen wird quantitativ analysiert. Verbesserungen an Bayes-filterbasierten und graphoptimierten SLAMs werden in dieser Arbeit vorgestellt. Bayes-filterbasierte SLAMs verwenden hauptsächlich die aktuellen Sensorinformationen, um die beste Schätzung zu finden. Eine neue effiziente Implementierung des auf Partikel-Filter basierenden SLAM unter der Verwendung einer Lookup-Tabelle und der Parallelisierung wird vorgestellt. Die neue Implementierung kann effizient auf aktuellen Embedded-Systemen laufen. Im Gegensatz dazu verwenden Graph-SLAMs alle Sensorinformationen und minimieren den Gesamtfehler im System. Ein neues Echtzeitmodel für die Grafenerstellung und eine robuste integrierte SLAM-Lösung werden vorgestellt. Die Verbesserungen umfassen die Definition von eindeutigen Richtungsnormen für Scan, effiziente Algorithmen zur Erkennung von Loop Closures und eine parallele und adaptive Implementierung. Der entwickelte und auf eingebetteten Systemen eingesetzte Algorithmus übertrifft die aktuellen Algorithmen in Geschwindigkeit und Robustheit, insbesondere für große Areale.

Die Ergebnisse der Arbeit können für die Verbesserung der Navigation von autonomen Robotern im Innenbereich, häuslichen Umfeld sowie der Intra-Logistik genutzt werden.

Table of contents

List of figures **xiii**

List of tables **xix**

1 Introduction **1**
1.1 Indoor autonomous robotics 1
1.1.1 History of autonomous navigation robotics 2
1.2 Classification of indoor navigation systems 3
1.3 Motivation 4
1.4 Contributions 5

2 Simultaneous Localization and Mapping Techniques **7**
2.1 Introduction 7
2.2 Bayesian theorem based methods 8
2.2.1 Linear filtering 10
2.2.2 Nonlinear filtering 10
2.3 Graph Optimization 12
2.4 Important models often used in SLAM algorithms 17
2.4.1 Motion model 17
2.4.2 Observation model 18
2.4.3 Map representation 19
2.5 State-of-the-art SLAM algorithms 19
2.5.1 GMapping 19
2.5.2 Hector SLAM 20
2.5.3 Cartographer 20
2.6 Summary 21
2.6.1 Comparison of different SLAM techniques 21
2.6.2 Challenges of laser scanner based SLAM techniques 21

3 Contributions to 2D Bayesian Based SLAM 23
3.1 Benchmark system and reference system . . . 23
3.1.1 Benchmark system . . . 23
3.1.2 Reference system . . . 24
3.2 Comparison of different scan matching algorithms with different laser scanners 26
3.2.1 Introduction . . . 26
3.2.2 Related works . . . 26
3.2.3 Scan matching algorithms . . . 28
3.2.4 Experiment . . . 30
3.2.5 Results . . . 33
3.2.6 Conclusion . . . 38
3.3 EMB-SLAM: An embedded efficient implementation of Rao-Blackwellized Particle filter based SLAM . . . 39
3.3.1 Introduction . . . 39
3.3.2 Rao-Blackwellized Particle Filter . . . 40
3.3.3 Improved scan matching algorithm based on GMapping . . . 43
3.3.4 The parallel programming models . . . 45
3.3.5 Hardware and software configuration . . . 45
3.3.6 Algorithm implementation and evaluation . . . 46
3.3.7 Experimental results . . . 49
3.3.8 Discussion . . . 52
3.3.9 Conclusion . . . 53
3.4 Summary . . . 53

4 Contributions to 2D real-time robust graph based SLAM 55
4.1 Introduction . . . 55
4.1.1 The history of graph optimization based SLAM and the theory . . . 55
4.1.2 Graph optimization related sensors and applications . . . 56
4.1.3 The components of graph optimization based SLAMs . . . 56
4.1.4 State-of-the-art of the graph building techniques . . . 56
4.1.5 Description of the existing problems . . . 62
4.1.6 Structure of the Chapter . . . 63
4.2 Preliminaries . . . 63
4.2.1 Error propagation . . . 63
4.2.2 The ellipsoid representation of laser scans . . . 66
4.2.3 Extracting segments from a single scan . . . 69
4.3 System overview . . . 73

4.4 Robust basic edge building . . . 73
4.4.1 Introduction . . . 73
4.4.2 A long-term + short-term sliding window based basic edge builder . 75
4.4.3 A new nonlinear distance weight function for the goodness function of the scan matching algorithm . . . 77
4.4.4 An environment-aware enhancement for the brutal force based scan matching algorithm . . . 78
4.4.5 The filtering of dynamic objects . . . 82
4.4.6 A comparison of persistent-mode vs frame-to-frame scan matching for basic edge building . . . 84
4.5 Efficient ellipsoid based loop closure edge building . . . 91
4.5.1 Preprocessing . . . 92
4.5.2 Loop closure detection . . . 98
4.5.3 Loop closure validation . . . 106
4.6 Implementation of the adaptive Graph SLAM . . . 115
4.6.1 Incremental Graph Optimization . . . 115
4.6.2 Fine backward loop closure detection . . . 115
4.6.3 Parallel processing of graph based SLAM . . . 117
4.7 Results . . . 119
4.7.1 ACES Building . . . 119
4.7.2 Intel Research Lab . . . 120
4.7.3 MIT Stata Building . . . 122
4.7.4 MIT Killian Court . . . 122
4.7.5 Fraunhofer IOSB-AST Building . . . 124
4.7.6 Real-time performance of the Graph SLAM . . . 127
4.8 Summary . . . 128

5 Applications of the robust and adaptive SLAM in real environments 129
5.1 Project Othello . . . 129
5.1.1 Introduction . . . 129
5.1.2 The author's work in the project . . . 131
5.2 Project Klara . . . 134
5.2.1 Introduction . . . 134
5.2.2 The author's work in the project . . . 137

6 Summary 141
6.1 Conclusion . . . 141

6.2 Outlook 143

References **147**

List of figures

1.1 The applications of indoor autonomous robotics. 2
1.2 The automated guided vehicles used in industrial environments. 3
1.3 Classification of indoor localization methods 4

2.1 Odometry 8
2.2 A graphical model of the Bayesian filter applied for the SLAM problem. . . 9
2.3 An illustration of the graph model. The graph is made of nodes (robot poses x or landmarks l) and edges (data associations between robot poses or landmarks). A loop closure is indicated by the red arrow. The presence of landmarks is optional. 12
2.4 An illustration of the error function. x_i and x_j are nodes, z_{ij} is the real observation and z'_{ij} is the expected observation of x_j from x_i. The error function is $e_{ij}(x_i, x_j)$. 13
2.5 Found loop closure and graph optimization based backwards correction. . . 15
2.6 Graph optimization after found loop closure. 16
2.7 Correction of poses by Graph optimization after found loop closure. 16
2.8 An illustration of the motion model from Thrun. 17
2.9 An example of scan beams projected from the orange cell. The gray cells indicate obstacles in the environment. 18

3.1 The explanation of accuracy benchmarking. The left image illustrates absolute error, and the right image illustrates relative error. Blue circles stand for ground truth poses, black circles stand for estimated poses, and orange arrows stand for the errors. 23
3.2 Illustration of the SICK NAV350 based reference system (a lab environment). 25
3.3 The scan matching problem. 26
3.4 The classification of scan matching algorithms. 28

3.5 The simulated home-like environment. The white line is the trajectory, and the red points are from a scan observation. 30
3.6 Flowchart of the experiment. 32
3.7 The robustness comparison of CSM, ICP, MbICP, PSM and NDT. 34
3.8 The accuracy comparison of CSM, ICP, MbICP, PSM and NDT. 35
3.9 The processing comparison of CSM, ICP, MbICP, PSM and NDT. 36
3.10 A look-up table example at 0.05m resolution. Unit in the right figure stands for 0.05m. 44
3.11 Embedded computation unit ODROID-XU4 46
3.12 Processing loop of parallel Rao-Blackwellized particle filter 48
3.13 The comparison of speedup obtained using multi-processing. 50
3.14 The comparison of memory usages of different multi-processing models. The memory usage of GMapping is used a reference. 51
3.15 The average execution time per update using EMB-SLAM for Intel Research Lab dataset and ACES Building dataset on ODROID-XU4. 53

4.1 Error propagation from scan matcher 65
4.2 Ellipsoid example 67
4.3 The rotation and expansion of an ellipsoid. 68
4.4 Extract segment features from a single scan. 70
4.5 Merge line features from multi scans. 71
4.6 Merge segment features from multi scans. 72
4.7 Outline of the Ellipsoid based Graph SLAM pipeline 73
4.8 The flowchart of on-line adding long term and short term nodes in the robust basic edge building method. 76
4.9 The flowchart of on-line long term and short term nodes based robust basic edge building method. 77
4.10 False view side effect. In (a)(c)(d), the red points stand for the old local map, the blue points stand for the current scan. The translational result from low resolution search at the correct angle is shown in (b), where two yellow cells stand for high score. The higher one corresponds to the pose in (b), and the lower one corresponds to the pose in (c). The vanilla CRSM algorithm will come to the result in (d) which has the highest goodness score; with the extension of the new developed outlier check method, the correct result is found in (c). 79

4.11 Long corridor effect. In (a)(b)(c), the red points stand for the old local map, the blue points stand for the current scan. The translational result from low resolution search at the correct angle is shown in (d), where an array of yellow cells stand for high score. The highest one corresponds to the pose in (b), and the correct one corresponds to the pose in (c). The vanilla CRSM algorithm will come to the result in (b) which has the highest goodness score; with the extension of the new developed ambiguity check method, the correct result is found in (c). 81
4.12 The simulated warehouse. 85
4.13 Comparison for basic edge building. The left column is PLICP, the right column is the author's new method. Red · is the first scan, black · is the second scan, blue · is the transformed scan. 86
4.14 Histogram of relations between nodes in Intel Research Lab and ACES Building. The left column is Intel research Lab dataset, the right column is ACES Building dataset. The unit resolution in the histogram is 0.1 meter for translation, and 0.1 radian for rotation. 87
4.15 Comparison for basic edge building. The left column is Intel research Lab dataset, the right column is ACES dataset. 88
4.16 Comparison for basic edge building. The left column is Intel research Lab dataset, the right column is ACES building dataset. 90
4.17 The cost for different MNT values. 91
4.18 Ellipsoid based loop closure detection 91
4.19 Check whether ellipsoids intersect with each other, the size of the shared free area in this case is 7.38 m^2. 94
4.20 An example for demonstrating the improvement of the overlay rate after using dynamic local map generation. The top figure is scan to scan, and in the bottom figure is local map to local map. The blue color stands for source node, and the red color stands for target node. The magenta color stands for the shared area. 99
4.21 Voxel filter based scan down-sampling. The left column top to down are resolution factor from 1 to 5, and in the right column top to down are resolution factor from 6 to 10. The resolution unit is 0.02 m. 101
4.22 Example of candidate nodes in ordered layers. Different colors stand for different layers. Dashed lines stand for candidate loop closures. 104

4.23 Histogram intersection example. The size of the cell in 2D histogram is 0.1m*0.1m. The histogram intersection result is 0.33425 in this case. The geometry complexity of the shared area is 0.865371. 107
4.24 Loop closure hypothesis with high correlative metric but low complexity metric example. The scan matching goodness of the reference map is 0.701, and the false scan matching pose has the goodness of 0.751. So the output pose will be the false scan matching pose. The histogram intersection result is 0.276 in this case. The geometry complexity of the shared area is 0.039. . 108
4.25 Closest correspondences between reference frame and target frame. 109
4.26 Negative segment pair check example. The scan matching goodness of the reference map is 0.383, and the false scan matching pose has the goodness of 0.399. So the output pose will be the false scan matching pose. The histogram intersection result is 0.259 in this case. The geometry complexity of the shared area is 0.935. 110
4.27 From the images at old and new viewpoints, we can see the two sides of the wall (ca. 30 cm thick). Scan matching algorithms will normally ignore the thickness of the wall and merge them a line. 111
4.28 Local Graph optimization for validating the loop closure hypothesis. 113
4.29 A ROC for Dataset MIT Killian is shown to demonstrate the effectiveness of different validation methods. Let assume 0.01 false positive rate is allowed, all three criteria method has 0.727 accuracy with matching threshold 0.263, correlative threshold 0.136, complex threshold 0.171, correlative and complex method has 0.721 accuracy, matching and complex has 0.665 accuracy. 114
4.30 The pipeline of backward fine optimization 116
4.31 UML Sequence diagram of Graph SLAM 118
4.32 ACES trajectory. 119
4.33 ACES maps. 120
4.34 Intel odometry trajectory and optimized trajectory. 120
4.35 Intel. 121
4.36 MIT Stata. 122
4.37 MIT Killian. 123
4.38 FhG IOSB-AST odometry path and corrected path. 125
4.39 FhG IOSB-AST maps. 126
4.40 FhG IOSB-AST graph. 126

5.1 The robot developed for Project Othello. 131

5.2 Othello platform in the demonstration environment of project partner Focal Meditech BV in Tilburg NL (Othello house). 132
5.3 The test in Othello house. 133
5.4 The test in Evoluon Endhoven. 134
5.5 The robot used in project Klara. SICK S300 LIDAR; IFM TOF; two Fotonic ASTRA 3D; ODroid-XU3, Plug-In IPC, Speed Real-Time Target Machine. 136
5.6 The robot used in project Klara. SICK S300 LIDAR; IFM TOF; two Fotonic ASTRA 3D; ODroid-XU3, Plug-In IPC, Speed Real-Time Target Machine. The test area is about $23m \times 28m$. 137
5.7 The robot is equipped with a Hokuyo UTM-30LX Scanning Laser Rangefinder. The test area is about 22,000 m^2, $130m \times 165m$. 139
5.8 The robot is equipped with a Hokuyo UTM-30LX Scanning Laser Rangefinder. The test area is a factory about 22,000 m^2, $130m \times 165m$. 140

6.1 An example of objects detected from an image by a trained model. 144
6.2 The simulated semantic labeling based SLAM. 145

List of tables

2.1 Comparison of SLAM theories 21

3.1 Parameters for investigating the effect of FOV 33
3.2 Required field-of-view with odometry input 37
3.3 Required field-of-view without odometry input 38
3.4 Computation units 46
3.5 Datasets information 48
3.6 The mean processing time of the particle update method in GMapping and EMB-SLAM 50
3.7 The execution time of different approaches for Intel dataset in seconds . . . 51
3.8 The execution time of different approaches for ACES Building dataset in seconds 51
3.9 Mean translational error ε_{trans} and mean rotational error ε_{rot} 53

4.1 WCRSM compared with PLICP for basic edge building 84
4.2 Parameter setting of experiment 1 85
4.3 Datasets used in experiment 1 85
4.4 Parameters for adding new vertex 90
4.5 Parameters for adding new mini vertex 91
4.6 Datasets used in preprocessing experiment 96
4.7 Comparison of preprocessing methods based on scan matching pose and covariance (max 100 candidates per node) 97
4.8 Comparison of preprocessing methods based on scan matching pose and covariance (max 30 candidates per node) 97
4.9 Processing time for VoxelGrid filter with resolution from 0.02m to 0.2m. . . 102
4.10 The real-time performance of Graph SLAM on ODROID-XU4 127
4.11 The real-time performance of Graph SLAM on laptop Lenovo T540p . . . 127

Chapter 1

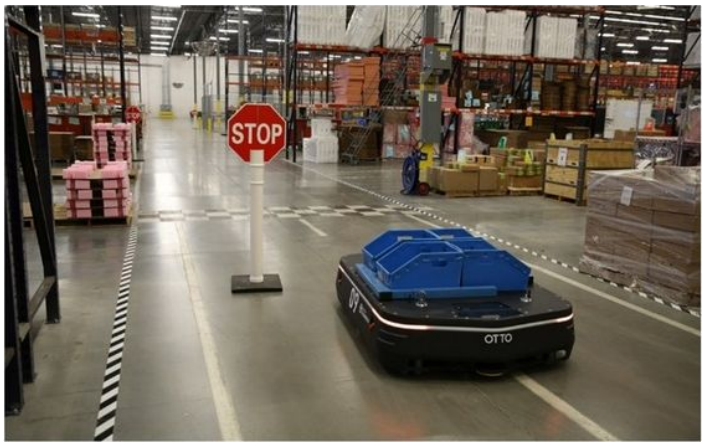
Introduction

1.1 Indoor autonomous robotics

Indoor autonomous robotics are normally battery-driven, unmanned vehicles that have been widely used for guidance [17][94], health-care [35], logistic [88], search and rescue [51], security [4][21] and so on.

The aim for Automated Guided Vehicle (AGV) is simple: To replace human operators working manually or using traditional material handling equipments such as fork trucks or pallet jacks with an automated vehicle to handle the same load. The aim is not replacing human workers in general, but helping and protecting human workers from work which may do potential damage to the human health, for example, there is a Germany law [1] for protecting human workers by limiting the load of occasional lifting and carrying. With the help of autonomous robots, the human worker can focus on more productive work. As shown in Fig. 1.1a, a robot is used for the transporting of goods in the industrial environment. Autonomous robotics are also designed to help take care of the disabled or elderly people at home. In countries like Japan (27,87% are aged 65 or above, 2017), Germany(22.06% are aged 65 or above, 2017), Sweden (20.26% have passed the standard retirement age of 65, 2017) [23], there are more and more old people. The service robot can play an important role for helping the daily life of the elderly people. An example of service robots is shown in Fig. 1.1b.

[1] https://www.gesetze-im-internet.de/lasthandhabv/__2.html



(a) OTTO 1500.

(b) Korea service robot.

Fig. 1.1 The applications of indoor autonomous robotics.

1.1.1 History of autonomous navigation robotics

A major branch of autonomous robotics is automated guided vehicles which are battery-driven, unmanned vehicles that have been transporting materials and products for over 50 years. In the early 1950s, the first AGV systems were introduced in warehouses and consisted of modified towing tractors pulling carts following an overhead wire. By the early 1960s, towing AGVs were deployed in many types of manufacturing and warehousing operations. In the 1970s when the unit loads AGV was introduced, with this advancement, an electronic frequency was included in wires that were buried in the floor. By the late 1980s, non-wire-guided AGVs were introduced, allowing for more system flexibility and accuracy. Magnets or tapes are used for guiding the robot [99]. Recently, an increasing number of AGVs use laser-guided or vision-guided technology with zero infrastructure installed in the environment. An example of AGV which is able to follow the ground lines is shown in Fig. 1.2a, and an example of AGV without the need of ground lines is shown in Fig. 1.2b.
Despite advances with laser guidance and other improvements, AGVs are still viewed by many as being inflexible – slow and difficult to change movement paths. This is a weakness in today's highly dynamic distribution environment, as facility layout and related changes generally happen much more frequently in manufacturing facilities.

(a) AGV from company SSI Schäfer with ground lines.

(b) AGV from company Seegrid Vision without ground lines.

Fig. 1.2 The automated guided vehicles used in industrial environments.

1.2 Classification of indoor navigation systems

Many technologies enable the autonomous behavior of indoor robotics. A fundamental ability or technology which an autonomous robot must be capable of is navigation. The definition of navigation is the following: *Navigation is a field of study that focuses on the process of monitoring and controlling the movement of a craft or vehicle from one place to another* [26]. In our case, the field of navigation is indoor land navigation. The navigation problem which is a main prerequisite for an autonomous mobile robot to fulfill high-level tasks such as handling and manipulation is often identified as one of the key challenges in mobile robotics that must be solved before robots can become part of everyday life in domestic homes and the workplace. In the field of navigation, there are several components, for example, mapping and localization, path planning, controlling, obstacle avoidance and so on. In this work, the research is focused on the mapping and localization tasks, which are also the fundamental parts of the navigation.

A classification of the state-of-the-art indoor navigation systems is presented in Fig. 1.3. The indoor navigation system can be dived into two major classes, with infrastructure or without infrastructure installed in the environment. With the help of extra infrastructure installed in the environment, the robot can receive its current pose from external sources, most of them work like indoor Global Positioning System (GPS). The real GPS, due to the technology request of line-of-sight when connecting to satellites suffers from weak signal and low accuracy in indoor environment. A survey of active and passive indoor localization systems is presented in [30][68][58]. For example, Ultra Wide Band (UWB) based indoor localization system can achieve very high precision, up to approximately 15 cm for 95% of

the readings [90]. The disadvantage is also obvious, extra infrastructure need to be installed, which may be not always possible.
Another class of navigation systems requires zero infrastructure installed in the environment, for example, using magnetic field for positioning is discussed in [67], localization is done by comparing the current fingerprint of the magnetic field with a pre-recorded fingerprint database which requires also a lot of effort. Vision or optical based indoor positioning systems are discussed in [73]. Optical indoor positioning systems are ego-motion systems, where a mobile camera is to be located. Current optical indoor positioning approaches achieve accuracy levels between a couple of µm and dm. But the disadvantage is obvious, the lighting condition has a huge impact on the optical based solutions. Another popular approach is using Laser range finder for localization and mapping purpose. For the autonomous operation, the robot needs to know the environment, when the operation environment is previously unknown, the robot has to build a map of the environment, for building a high quality map, the robot has to localize itself in the map. So a method called Simultaneous mapping and localization (SLAM) is developed to solve this problem. More detailed classification and comparison of SLAMs will be discussed in Chapter 2.
There is no standard indoor navigation system, thus the selection of an existing system need to be done based on the operation environment and the accuracy required.

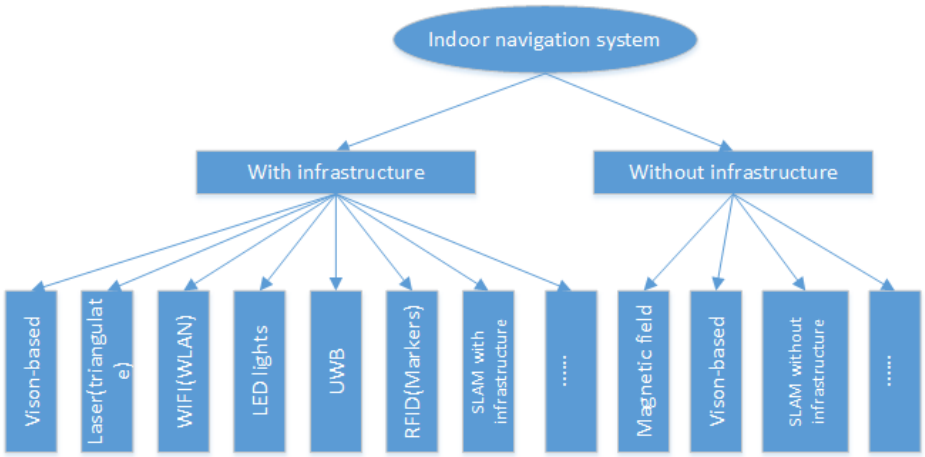


Fig. 1.3 Classification of indoor localization methods

1.3 Motivation

The navigation module of autonomous robotics relies on the intelligent software together with the on-board sensors to control the vehicle's movement by knowing the operation

environment and the current position in the environment, an improved flexible approach to indoor robot control. Using the laser range finders and other sensors, a map for the indoor robot in its environment can be built. As a result, the robots can generate their own "guide path" for the movement of materials from different points in the environment and with much less restrictive tolerances for making those moves than traditional indoor robot technologies. In the past decades, although significant improvements of the navigation system have been achieved, the navigation problem which is a main prerequisite for an autonomous mobile robot to fulfill high-level tasks such as handling and manipulation is still identified as one of the key challenges in mobile robotics that must be solved before robots can become part of everyday life in domestic homes and the workplaces. For the mapping and localization part, the sensor noise and the perceptual similarities in the environment are difficult to be handled by state-of-the-art methods especially in large-scale environments. Another problem of the current navigation systems is the requirement of high-end computation units which are expensive and not energy efficient for mobile robotics.
The goal of this work is to develop an adaptive navigation system for indoor robotics without the need of many changes in the infrastructure. Further, the system should be efficient and able to run in real-time on low-cost and energy-efficient embedded devices for large-scale areas.

1.4 Contributions

In this work, the research is focused on the mapping and localization part of the navigation system.
A key component of LiDAR (light detection and ranging) based SLAM algorithms, scan matching algorithms, is explored. Different scan matching algorithms are systemically experimented with different kind of laser scanners for indoor home-like environments for the first time .
A new efficient implementation of Rao-Blackwellized Particle Filter based SLAM is presented. It is based on the usage of a pre-computed lookup table and the parallelization of the particle updating. Different task-based parallelizing models are implemented, and the performance is compared for the parallelization task. The new implementation runs efficiently on recent multi-core embedded systems which fulfill low cost and energy efficiency requirements.
A new real-time graph building model and a full integrated robust Graph SLAM solution are presented. The improvements include the definition of unique direction norms for scan observations, efficient loop closure detection algorithms, and a parallel and adaptive

implementation of the proposed solution. The proposed solution outperforms the state-of-the-art algorithms in processing time and robustness especially in large-scale environments using embedded systems instead of high-end computation systems.
This thesis is organized as follows. In Chapter 2, the background knowledge of the Simultaneous Mapping and Localization (SLAM) is given, the state-of-the-art SLAM algorithms are reviewed and compared, and important models used in SLAM are introduced. In Chapter 3, after the introduction of the benchmark system and reference system, the simulation based comparison of different scan matching algorithms with different laser scanners are described, and an embedded system based efficient implementation of Rao-Blackwellized Particle filter SLAM is developed. In Chapter 4, the contributions to the graph optimization based SLAM are explained. The developments of a real-time robust basic edge building method and an efficient ellipsoid based loop closure edge building method are explained, and an integrated solution is proposed and implemented. In Chapter 5, the applications of the newly developed methods are shown in two projects. In Chapter 6, the summary of the author's work and an outlook are presented.

Chapter 2

Simultaneous Localization and Mapping Techniques

2.1 Introduction

Autonomous robotics should be able to build maps of a prior unknown environments and localize themselves in the map. For the map building, the robot needs to know its pose in the environment for accumulating the scan observation into the map; for the localization, the robot requires a map first. So the mapping and localization problem is like a "chicken and egg" situation, one solution to this problem is the Simultaneous Localization and Mapping (SLAM) which the mapping and localization are handled at the same time.
In the last few decades, plenty of researchers have worked on this topic. In this work, the Laser Range Finder based SLAM techniques are discussed. One reason why the SLAM problem is difficult is that sensor measurements are normally noisy, especially after long time error accumulation, the drift can be significant as shown in Fig. 2.1. Another difficulty is the perceptual similarities in the environment which are difficult to be handled, for example, a long corridor without features or two places look the same. Those cases are sometimes even difficult for the human to distinguish. In the literature, plenty of SLAM algorithms, like EKF SLAM [102], FastSLAM [76], GMapping [42], Graph SLAM [41] and so on, have been proposed by researchers. Well-known SLAM algorithm GMapping requires a mobile robot that provides odometry data and is equipped with a horizontally mounted fixed laser range finder. GMapping is a Rao-Blackwellized Particle Filter (RBPF) based approach. The motion of the mobile robot can be modeled as a probabilistic model, and so is the current estimation of the robot pose [96]. Instead of using Bayesian models, another approach uses graph-based SLAM for solving the mapping and localization problem. In [97], an offline application of

graph based SLAM for large-scale mapping of urban structures is demonstrated. With the advancement of high update rate laser range finders, HectorSLAM [59] is developed based on a robust scan matcher, and the odometry information is not required.
As a brief conclusion of state-of-the-art SLAM algorithms, there are mainly two branches, one is Bayesian filtering based, another one is graph theory based. The related theories for two approaches will be explained in the following sections. A comparison of the state-of-the-art algorithms will be shown at the end of this chapter.

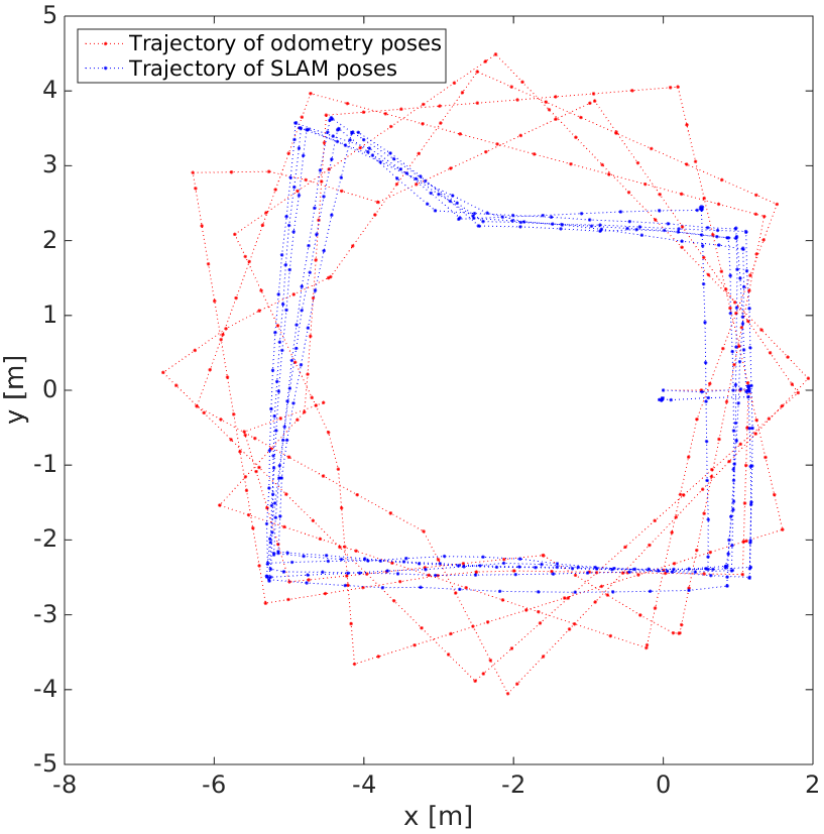


Fig. 2.1 A demonstration of the noisy raw odometry data from our robot Pioneer-3AT driving in the laboratory at Fraunhofer IOSB-AST.

2.2 Bayesian theorem based methods

A state-space model for a general nonlinear system [103] is given by

$$x_{k+1} = f(x_k, u_k) + w_k, \quad w_k \sim p(w_k) \tag{2.1}$$

$$z_k = h(x_k) + e_k, \quad e_k \sim p(e_k) \tag{2.2}$$

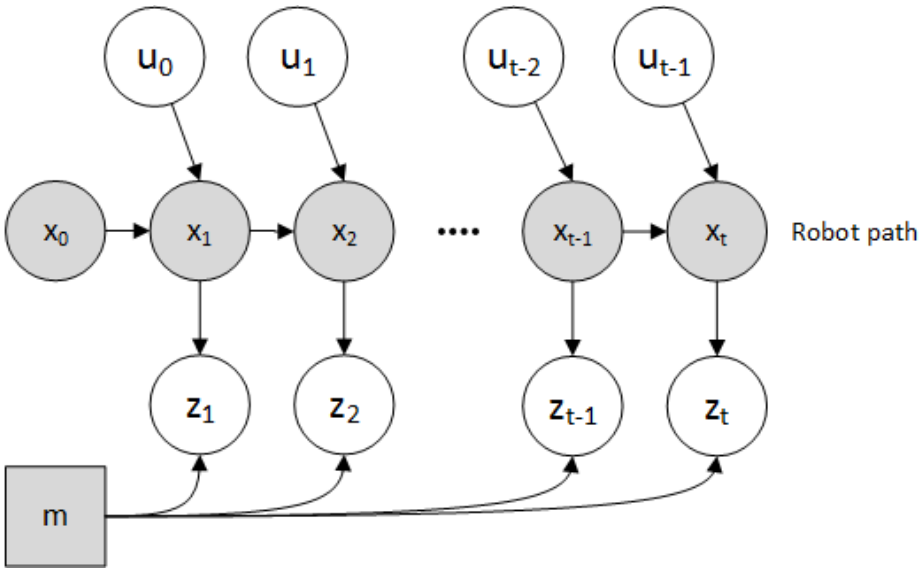


Fig. 2.2 A graphical model of the Bayesian filter applied for the SLAM problem.

where x_k is the state vector, z_k is the measurement, u_k is an external control input (for example, a driving command). w_k is a disturbance caused by the model error and the noise in u_k. e_k is the noise in the measurement z_k. The distributions of w_k and e_k can be arbitrary, but their probability density functions $p(w_k)$ and $p(e_k)$ are assumed known at the filter design time. Only z_k and u_k are measurable. A graphical model of the Bayesian filter applied for the SLAM problem is illustrated in Fig. 2.2, where the x stands for the robot pose and m represents the map.

Considering Bayes' theorem conditioned on variable $z_{1:k-1}$ (k is the time step, $z_{1:k}$ stands for the measurements from time step 1 till step k.)

$$p(x_k|z_k, z_{1:k-1}) = \frac{p(z_k|x_k, z_{1:k-1})p(x_k|z_{1:k-1})}{p(z_k|z_{1:k-1})}, \tag{2.3}$$

recognizing the Markov property of the state-space models yields

$$p(x_k|z_{1:k}) = \frac{p(z_k|x_k)p(x_k|z_{1:k-1})}{p(z_k|z_{1:k-1})}, \tag{2.4}$$

where $p(z_k|x_k)$ is called the likelihood of the measurement. The denominator in the above formulation can be expressed by using the law of the total probability

$$p(z_k|z_{1:k-1}) = \int_{\mathbf{R^n}} p(z_k|x_k)p(x_k|z_{1:k-1})dx_k. \tag{2.5}$$

Using Bayes' rule conditioned on $z_{1:k}$,

$$p(x_{k+1}, x_k | z_{1:k}) = p(x_{k+1}|x_k) p(x_k|z_{1:k}), \tag{2.6}$$

integration on both sides with respect to x_k over the entire state space using the law of the total probability yields

$$p(x_{k+1}|z_{1:k}) = \int_{\mathbf{R^n}} p(x_{k+1}|x_k) p(x_k|z_{1:k}) dx_k. \tag{2.7}$$

By using Equation (2.4) and (2.7) recursively and initiating by

$$p(x_0|z_0) = p(x_0), \tag{2.8}$$

we arrive at the general Bayesian recursion equations

$$p(x_k|z_{1:k}) = \frac{p(z_k|x_k) p(x_k|z_{1:k-1})}{p(z_k|z_{1:k-1})} \tag{2.9}$$

$$p(z_k|z_{1:k-1}) = \int_{\mathbf{R^n}} p(z_k|x_k) p(x_k|z_{1:k-1}) dx_k \tag{2.10}$$

$$p(x_{k+1}|z_{1:k}) = \int_{\mathbf{R^n}} p(x_{k+1}|x_k) p(x_k|z_{1:k}) dx_k \tag{2.11}$$

that need to be solved to arrive at the filtering and prediction densities.

2.2.1 Linear filtering

For linear systems with Gaussian noise distributions it can be proven that Kalman filter is an optimal solution. The general problem of trying to estimate the state $x \in \mathfrak{R}^{\mathbf{n}}$ of a discrete-time controlled process that is governed by the linear stochastic difference equation is addressed by the Kalman filter [56].

2.2.2 Nonlinear filtering

The extended Kalman filter (EKF) [74] is an extension of the Kalman filter that allows systems with minor nonlinearities in the system dynamics and Gaussian or almost-Gaussian noise distributions. Another approach unscented Kalman filter (UKF) [52] lets a number of points in the state-space propagating through the model and afterwards recovering the distribution mean and covariance. In some cases, this method allows prediction functions with greater nonlinearities than the EKF can handle. Both EKF and UKF can handle models with small nonlinearities, but when the process noise term becomes too big and the filter yields

very little useful information. To handle arbitrary distributions and greater nonlinearities it is better to use other statistical methods, such as a point mass filter (PMF) [10] or particle filter (PF) [69].

Particle filter

Particle filter, as a Bayes filter, is an efficient way to represent non-Gaussian distribution. There are two basic principles: particles are a set of state hypotheses and they are survival-of-the-fittest. In the SLAM content, SLAM can be represented as a sample based Posterior by particles as following:

$$S = \{< s^{(i)}, w^{(i)} > | i = 1, ..., N\} \tag{2.12}$$

$$p(x) = \sum_{i=1}^{N} w^{(i)} \cdot \delta_{s^{(i)}}(x), \tag{2.13}$$

where $s^{(i)}$ is the ith sample, $w^{(i)}$ is the weight of the ith sample, and N is the number of samples.
Particle filters have been successfully applied to localization task, and later researchers extended the particle filter to solve SLAM problem too. The difference between localization and SLAM can been seen in the following:

$$p(x|m,z,u) \longrightarrow p(x,m|z,u), \tag{2.14}$$

where x is the robot pose, m is the map, z is the observation, u is the control input. The left side is the localization posterior, and the right side is the SLAM posterior. For the localization task, the map is known. For the SLAM task, the map is initially unknown. If robot poses are known, it is easy to build the map by adding observations into corresponding known poses. Rao-Blackwellization is the process of splitting the robot pose estimation (localization) and the mapping. The factorization is first introduced by Murphy in 1999 [33] as following:

$$p(x_{1:t}, m|z_{1:t}, u_{0:t-1}) = p(x_{1:t}|z_{1:t}, u_{0:t-1}) \cdot p(m|x_{1:t}, z_{1:t}), \tag{2.15}$$

where t is the time step, $x_{1:t}$ is the robot trajectory, m is the map, $z_{1:t}$ is the observations and $u_{0:t-1}$ is the movements. $p(x_{1:t}, m|z_{1:t}, u_{0:t-1})$ is the SLAM posterior. $p(x_{1:t}|z_{1:t}, u_{0:t-1})$ is the robot path posterior. $p(m|x_{1:t}, z_{1:t})$ is mapping with known poses which is easy. In RBPF-SLAM, every particle carries a potential trajectory of the robot and its own map. Each particle survives with a probability proportional to the observations relative to its own map. All particles together represent a joint posterior about the poses of the robot and the map.

Since each map is quite big in case of grid maps and each particle has its own map, the number of particles should be kept small. In [42], scan matching poses are used as input to the RBPF instead of the raw odometry. Fewer particles are needed, since scan matching provides a locally consistent pose correction which has smaller error than the raw odometry. According to [42], the optimal proposal distribution with respect to the variance of the particle weight is as following:

$$p(x_t|m_{t-1}^{(i)},x_{t-1}^{(i)},z_t,u_{t-1}) = \frac{p(z_t|m_{t-1}^{(i)},x_t)p(x_t|x_{t-1}^{(i)},z_t,u_{t-1})}{p(z_t|m_{t-1}^{(i)},x_{t-1}^{(i)},u_{t-1})}. \tag{2.16}$$

Using that proposal, the computation of weights becomes

$$w_t^{(i)} \propto w_{t-1}^{(i)} \cdot p(z_t|m_{t-1}^{(i)},x_{t-1}^{(i)},u_{t-1}) \tag{2.17}$$

$$= w_{t-1}^{(i)} \cdot \int p(z_t|x')p(x'|x_{t-1}^{(i)},u_{t-1})dx'. \tag{2.18}$$

More detailed derivation can be found in [7][42].

2.3 Graph Optimization

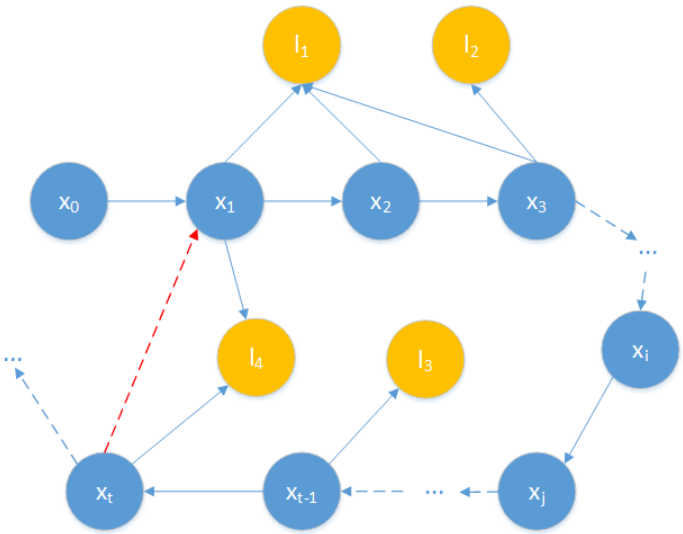


Fig. 2.3 An illustration of the graph model. The graph is made of nodes (robot poses x or landmarks l) and edges (data associations between robot poses or landmarks). A loop closure is indicated by the red arrow. The presence of landmarks is optional.

Besides Bayesian based methods, the SLAM problem can also be modeled as a graph, where every node includes a robot pose and a laser measurement, an edge between two nodes

represents a data-dependent spatial constraint between the nodes, then the graph can be treated a least squares problem. Once we have the most likely path by "moving" the nodes, then we can get the map based on the moved poses. The idea of graph based SLAM was first proposed by Lu and Milios [72]. They were historically the first to represent the SLAM prior as a set of edges between robot poses to use all the frames for the consistent registration. Lu and Milios's algorithm was successfully implemented by Gutmann and Konolige [44]. The Graph based SLAM algorithm was first presented in the information therory form by Thrun et al. [95]. Recently the formulation of the Graph based SLAM algorithm is further extended to model the state vector using relative information instead of absolute information [48]. Many works have been done in this area for effectively solving such problems. A general Graph based SLAM algorithm interleaves two steps, graph construction and graph optimization. An illustration of the graph model is shown in Fig. 2.3, where the x stands for the robot pose, l stands for the observed landmark in the environment and arrows stand for data associations. The presence of landmarks is optional. A loop closure is indicated by the red arrow.

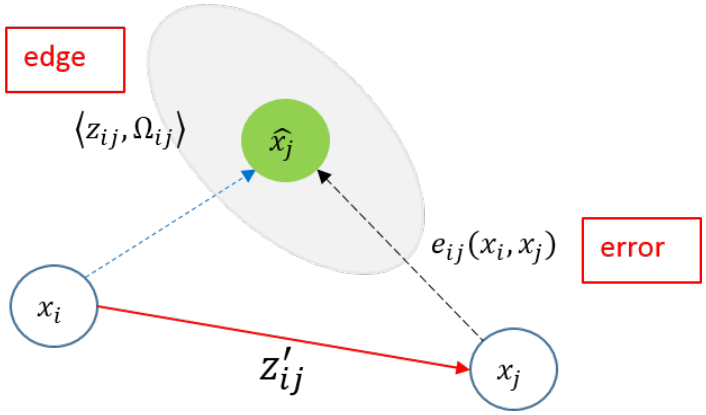


Fig. 2.4 An illustration of the error function. x_i and x_j are nodes, z_{ij} is the real observation and z'_{ij} is the expected observation of x_j from x_i. The error function is $e_{ij}(x_i, x_j)$.

Let $x_{1:n} = \{x_1, x_2, \ldots, x_n\}$, each node x_i is a 2D or 3D transformation representing the pose of the robot at time i. An edge z_{ij} is defined as a constraint between the node x_i and the node x_j. The constraint could be the robot observed the same part of the environment from both x_i and x_j or from the measurement of odometry.

To account for the noisy nature of the observations, an information matrix Ω_{ij} is added to the edge to encode the uncertainty of the edge. The "bigger" (in matrix sense) Ω_{ij} is, the more the edge "matters" in the optimization procedure.

Let $e(x_i, x_j, z_{ij}) = e_{ij}(x_i, x_j)$ be the function that computes the difference between the expected observation and real observation acquired by the robot. A graphical illustration of the error

function is shown in Fig. 2.4. x_i and x_j are nodes, z_{ij} is the real observation and z'_{ij} is the expected observation of x_j from x_i. The error function is $e_{ij}(x_i, x_j)$. Let $\mathscr{C}$ be the set of pairs of indices for which a constraint z exists.

$$e_{ij}(x_i, x_j) = z_{ij} - z'_{ij}(x_i, x_j) \tag{2.19}$$

The minimization of the system error becomes finding the configuration of the nodes $\hat{X}$ that minimize the negative log likelihood of all the observations, thus we seek to solve the following equation:

$$\hat{X} = argmin \sum_{<i,j>\in\mathscr{C}} e_{ij}^T \Omega_{ij} e_{ij} \tag{2.20}$$

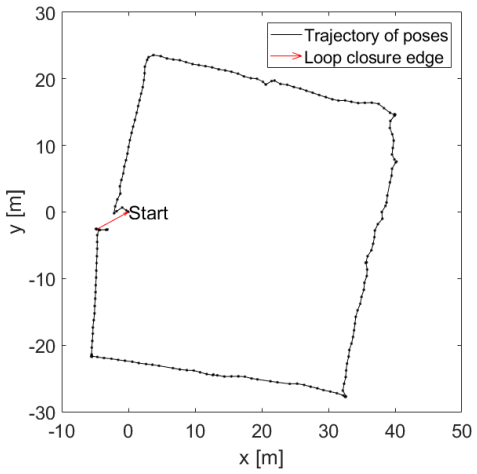
$$e_{ij}(X) = e_{ij}(X_i, X j) \tag{2.21}$$

The popular Levenberg-Marquardt or Gauss-Newton algorithms can be used to obtain the numeric solution of Eq. 2.20. The idea is to approximate the error functions around an initial guess X via Taylor expansion:

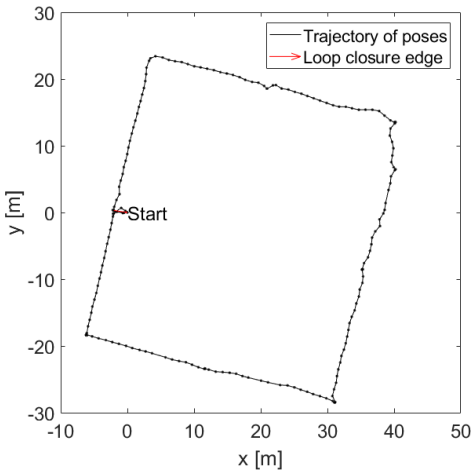
$$e_{ij}(X + \triangle X) \cong e_{ij} X + J_{ij} \triangle X \tag{2.22}$$

$$J_{ij} = \frac{\partial e_{ij}(X)}{\partial X} \tag{2.23}$$

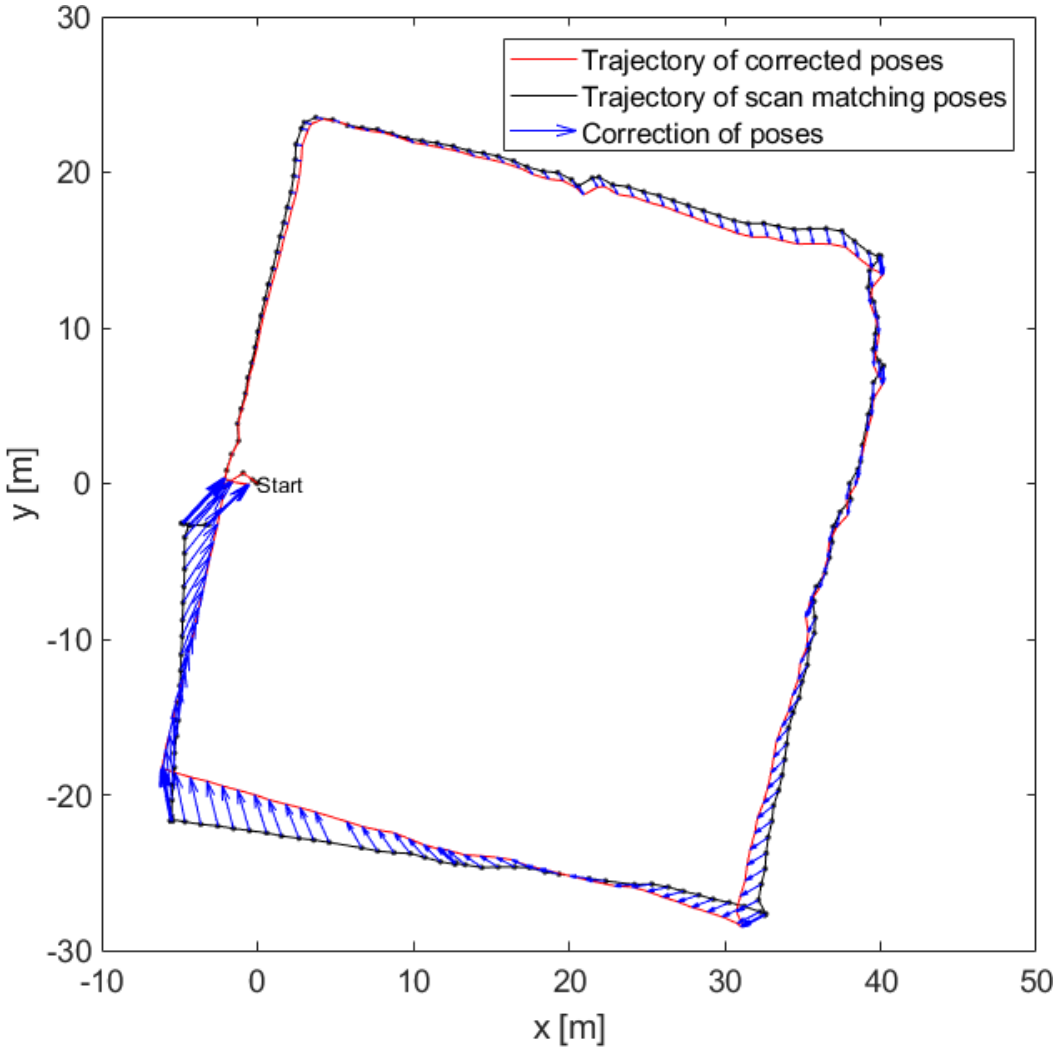
More detailed derivation can be found in [41].
A demonstration of Graph Optimization is shown Fig. 2.5. The trajectory before optimization is shown in Fig. 2.5a, and the optimized trajectory is shown in Fig. 2.5b. The corresponding maps can be seen in Fig. 2.6, after optimization the map becomes consistent. A graphic illustration of the absolute correction for every node in the graph is indicated using blue arrows in Fig. 2.5c. A plot of the relative and absolute corrections is shown in Fig. 2.7, the maximal relative translational error is below 0.0002 m and the maximal relative rotational error is below 0.002 rad which means even though at every step there is only a minor error, after 216 steps (number of nodes) the absolute error can already be significantly drifted from the correct pose. With graph optimization, those minor errors in every step can be 'moved' to achieve the trajectory in Fig. 2.5b. By using all the information in the graph, graph optimization is able to correct the errors in the past nodes.



(a) Found loop closure.



(b) Optimized trajectory.



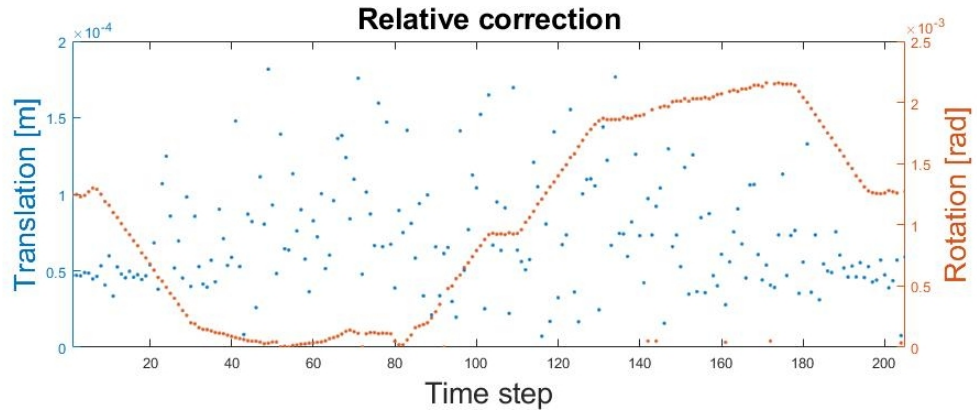
(c) Backwards correction.

Fig. 2.5 Found loop closure and graph optimization based backwards correction.

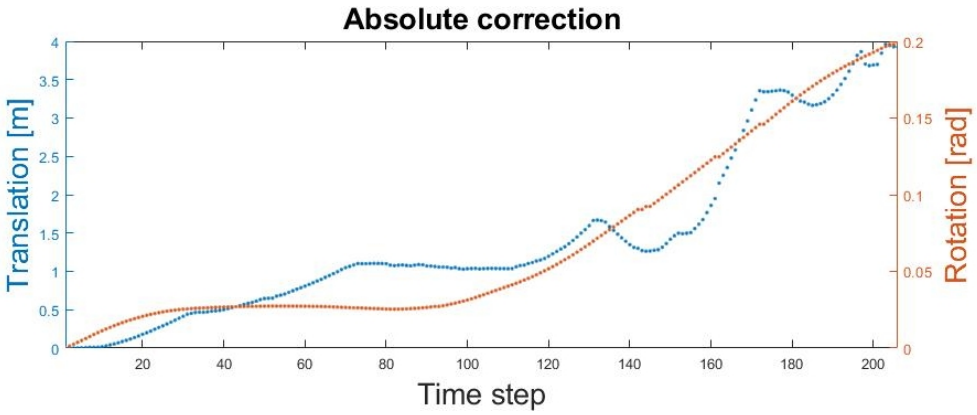
(a) Before graph optimization

(b) After graph optimization

Fig. 2.6 Graph optimization after found loop closure.



(a) Relative correction of poses after graph optimization



(b) Absolute correction of poses after graph optimization

Fig. 2.7 Correction of poses by Graph optimization after found loop closure.

2.4 Important models often used in SLAM algorithms

2.4.1 Motion model

Odometry motion model [95] is described as following: denote the odometry reading as $(\triangle x, \triangle y, \triangle \phi)$, and the prior robot pose is assumed to be $(0,0,0)$, which means that the search space is the robot increment, not the final robot pose. Then the new robot pose is

$$\begin{pmatrix} x' \\ y' \\ \phi' \end{pmatrix} = \begin{pmatrix} \cos\hat{\delta}_{rot1} & 0 & 0 \\ \sin\hat{\delta}_{rot1} & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} \hat{\delta}_{trans} \\ \hat{\delta}_{rot1} \\ \hat{\delta}_{rot2} \end{pmatrix} \tag{2.24}$$

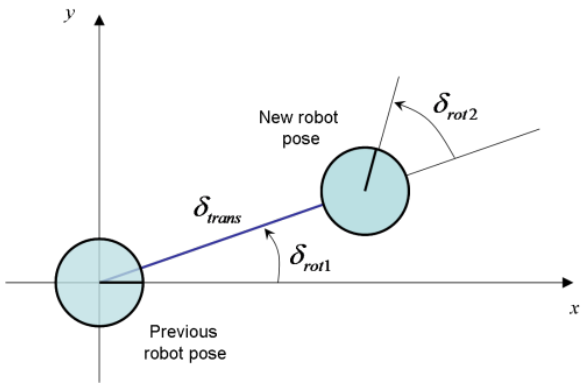


Fig. 2.8 An illustration of the motion model from Thrun.

where the variables $\hat{\delta}_{trans}$, $\hat{\delta}_{rot1}$ and $\hat{\delta}_{rot2}$ are the result of adding a Gaussian, zero-mean random noise to the actual odometry readings:

$$\begin{aligned} \hat{\delta}_{trans} &= \delta_{trans} + \varepsilon_{trans} & \varepsilon_{trans} &\sim \mathcal{N}(0, \sigma^2_{trans}) \\ \hat{\delta}_{rot1} &= \delta_{rot1} + \varepsilon_{rot1} & \varepsilon_{rot1} &\sim \mathcal{N}(0, \sigma^2_{rot1}) \\ \hat{\delta}_{rot2} &= \delta_{rot2} + \varepsilon_{rot2} & \varepsilon_{rot2} &\sim \mathcal{N}(0, \sigma^2_{rot2}) \end{aligned} \tag{2.25}$$

δ_{trans}, δ_{rot1} and δ_{rot2} are defined as following and illustrated in Fig. 2.8:

$$\delta_{trans} = \sqrt{\triangle x^2 + \triangle y^2} \tag{2.26}$$

$$\delta_{rot1} = atan2(\triangle y, \triangle x) \tag{2.27}$$

$$\delta_{rot2} = \triangle \phi - \delta_{rot1} \tag{2.28}$$

The model described in [95] employs the following approximations for the values of the standard deviations required for the equations above:

$$\begin{aligned} \sigma_{rot1} &= \alpha_1|\delta_{rot1}| + \alpha_2\delta_{trans} \\ \sigma_{trans} &= \alpha_3\delta_{trans} + \alpha_4(|\delta_{rot1}| + |\delta_{rot2}|) \\ \sigma_{rot2} &= \alpha_1|\delta_{rot2}| + \alpha_2\delta_{trans} \end{aligned} \tag{2.29}$$

where $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ are parameters used for describing the uncertainty of the motion model.

2.4.2 Observation model

Beam sensor model For laser scans, normally the beam sensor model is used by assuming every beam in the scan is independent. An example of scan beams is illustrated in Fig. 2.9.

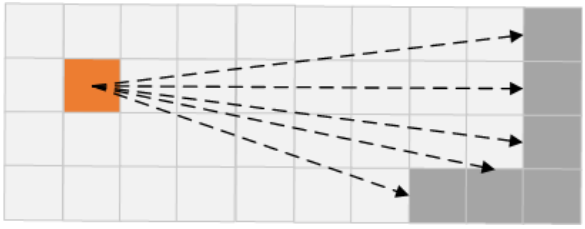


Fig. 2.9 An example of scan beams projected from the orange cell. The gray cells indicate obstacles in the environment.

Likelihood field model Based on the "beam model", where every individual scan beam is assumed to be independent, the likelihood could be calculated as following:

$$p(z_t|m_{t-1},x_t) = \prod_j p(z_t^j|m_{t-1},x_t), \tag{2.30}$$

where z_t^j is the jth beam of a scan z at time-step t. The likelihood of a beam is computed based on the distance between the projected endpoint using robot pose x_t and the closest obstacle from that point in the previous map m_{t-1} . Normally a Gaussian function is used for approximating $p(z_t^j|m_{t-1},x_t)$.

2.4.3 Map representation

The frequency based occupancy grid

The map representation used here is occupancy grid map. The world is divided into discrete cells, the size of those cells are defined by the map resolution. The map can be generated from the accumulation of observations at different poses. Each cell has a value indicate the state of the cell, whether it is free, occupied or unknown. For each cell $c_{x,y}$ of the map m, they carry two numbers: the number of times the cell has been visited $v_{x,y}$, and the number of times that the cell has been found occupied $o_{x,y}$. The probability of a cell is occupied is $p(c_{x,y}) = \frac{o_{x,y}}{v_{x,y}}$. Beam model is used here for updating the map, which means the scan observation at time t is represented as $z_t = (z_t^1, \ldots, z_t^n)$. Each beam will be transformed into the current robot pose x_t. $\widehat{z}_t = (\widehat{z}_t^1, \ldots, \widehat{z}_t^n) = z_t \oplus x_t$. Then a ray tracing procedure will be called to incrementally update the cells of the map for each beam $(x_t, \widehat{z}_t^i)$, the endpoint of the beam is occupied, while the points before endpoint are marked as free and updated accordingly.

The Normal Distribution Transform map

The Normal Distribution Transform (NDT) map is originally introduced in [12] for efficient laser scan matching. Similar to Occupancy Grid map, the mapped area is subdivided into voxels. The points inside the voxel are represented by a normal distribution.

The feature or landmark based map

In the computer vision community, manually placed landmarks or features extracted from the image could be used to track the movement of the robot, and therefore a feature or landmark based map is built.

2.5 State-of-the-art SLAM algorithms

2.5.1 GMapping

GMapping [42] is a Rao-Blackwellized particle filter (RBPF) based SLAM algorithm. The main contribution of GMapping is the reduction of the number of particles by taking into account not only the movement of the robot but also the most recent observation for computing an accurate proposal distribution. The improved proposal is acquired by executing a scan-matching algorithm using the movement from the motion model to compare the current observation with the map of the corresponding particle. The assumption of the improved

proposal relies on that the observation from the laser scanner is much more accurate than the odometry, so man can see that GMapping relies on good laser scanners and good scan-matcher algorithms. There is no loop closure technique used. GMapping uses a resampling threshold for avoiding particle depletion. The resampling threshold is measured by the effective sample size. The resampling function can help closing the loop, but this is not active loop closure technique, it works well when the error accumulation from before is small, so the correct solution can still be found. If the accumulated error is big, the map will be catastrophic. The gradient descent based scan matching algorithm is used in GMapping. The map representation used is occupancy grid map.

2.5.2 Hector SLAM

HectorSLAM [59] is a scan matching based mapper. There is no front-end theory like Bayesian theory or back-end theory like graph optimization used. The algorithm relies on high update rate laser scanner with low noise. To avoid being trapped in the local minima, a multi-resolution map representation is used. HectorSLAM has no loop closure technique. The scan matching problem is modeled as a least squares minimization problem, and the rigid-body transform is acquired by solving the Gaussian-Newton equation. All points or beams in the scan observation are modeled to fitting to the map best by transforming a delta pose.

2.5.3 Cartographer

Google's Cartographer [45] is a system that provides real-time simultaneous localization and mapping (SLAM) in 2D and 3D across multiple platforms and sensor configurations. The source code is open source available [1]. The system requirements of Cartographer are 64-bit, modern CPU (e.g. Intel 3rd generation i7) and 16 GB RAM. Cartographer is a pose graph optimization based approach. A scan is inserted into a submap after using a Ceres [3] based scan matcher. The loop closure detection is based on matching scans with those submaps. The building of the constraints for the graph is based on a branch-and-bound approach for computing scan-to-submap matches. The Ceres [3] is used for solving the optimization problem. In Chapter 5, the author's method is compared with Cartographer in a large factory environment.

[1] https://github.com/googlecartographer/cartographer

2.6 Summary

2.6.1 Comparison of different SLAM techniques

A comparison of Bayesian filter with graph optimization is shown in Table 2.1. The applied systems, posterior density assumptions and solution types are included.

Table 2.1 The comparison of SLAM techniques

Methods		Applied system	Posterior density assumption	Solution type
Bayesian filter	Kalman filter	Linear	Gaussian	Optimal
	Extended Kalman filter	Nonlinear	Gaussian	Approximate
	Particle filter	Nonlinear	Non-Gaussian	Approximate
Graph optimzation	Batch optimization	Nonlinear	Gaussian	Approximate
	Incremental optimization	Nonlinear	Gaussian	Approximate

2.6.2 Challenges of laser scanner based SLAM techniques

The laser scanner based SLAM algorithms are intensively researched in the last two decades, but still there are challenges or problems needed to be solved:

- Indoor human-made environments are normally smooth or continuous, but the scan observation acquired by laser scanners is consist of discrete points and every time only a slice of the environment is acquired.

- The perceptual similarity in the environment, for example a long corridor or two rooms look exactly the same, is a difficult problem even for human themselves.

- Algorithms like HectorSLAM require expensive high update rate laser scanners.

- High-performance computation devices are required for algorithms like GMapping and Cartographer.

All those challenges will be improved in this thesis and the solutions will be presented in the following chapters.

Chapter 3

Contributions to 2D Bayesian Based SLAM

3.1 Benchmark system and reference system

3.1.1 Benchmark system

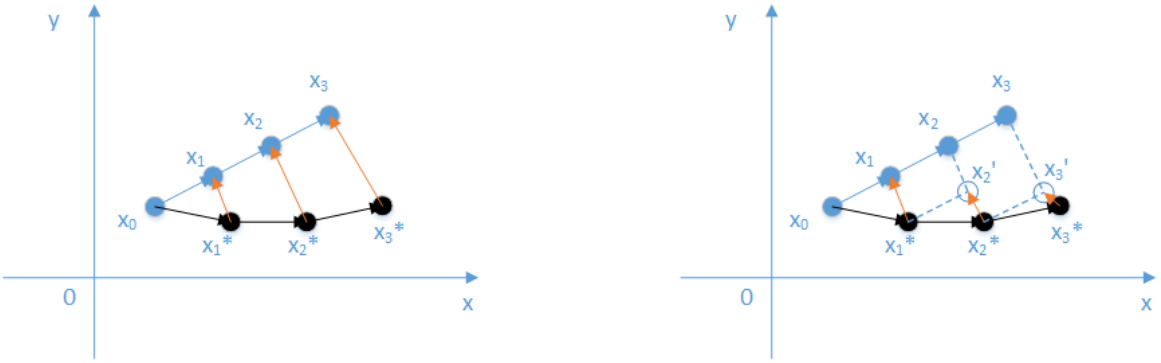


Fig. 3.1 The explanation of accuracy benchmarking. The left image illustrates absolute error, and the right image illustrates relative error. Blue circles stand for ground truth poses, black circles stand for estimated poses, and orange arrows stand for the errors.

For comparing different SLAM approaches, a benchmark system is needed. In this work, for benchmarking accuracy, absolute error metric and relative error metric are used. In Fig. 3.1, the left image illustrates absolute error, and the right image illustrates relative error. Blue circles stand for ground truth poses, black circles stand for estimated poses, and orange arrows stand for the errors.

The relative error metric following [60] is shown below. Let $\delta_{i,j} = x_j \ominus x_i$ be the relative transformation that moves the node x_i onto x_j and similarly $\delta^*_{i,j} = x^*_j \ominus x^*_i$.

$$\varepsilon(\delta) = \frac{1}{N}\sum_{i,j} trans(\delta_{i,j} \ominus \delta^*_{i,j})^2 + rot(\delta_{i,j} \ominus \delta^*_{i,j})^2, \tag{3.1}$$

where $\ominus$ is the inverse of standard motion composition operator, N is the number of relative relations, $\delta^*_{i,j}$ is the estimated relative transformation, $\delta_{i,j}$ is the reference relative transformation, $trans(\cdot)$ is the translation part of the transformation and $rot(\cdot)$ is accordingly the rotation part.
The absolute error metric is shown below.

$$\varepsilon(\delta) = \frac{1}{N}\sum_{i} trans(x_i \ominus x^*_i)^2 + rot(x_i \ominus x^*_i)^2, \tag{3.2}$$

where $\ominus$ is the inverse of standard motion composition operator, N is the number of nodes, x^*_i is the estimated pose, x_i is the reference pose, $trans(\cdot)$ is the translation part of the transformation and $rot(\cdot)$ is accordingly the rotation part.
The CPU and memory usage are also recorded, as it is desirable to run the program on embedded devices. The processing time or execution time is also an important criterion. The experiments are conducted under different computation platforms and different environments.

3.1.2 Reference system

For benchmarking the accuracy of SLAM poses, a reference system is needed. In simulated environments, for example, Player/Stage[100], Gazebo[1], the ground truth is always available. In real environments, the ground truth information could be provided by a high-accuracy positioning system, or one can use open source datasets with ground truth information provided.

Related works

The ground truth data associations could be acquired manually, Frese et al. [36] placed artificial landmarks in the environment, the position of the landmark is then given as a relative 2D coordinate in the robot frame. Burgard et al. [18] determined the close-to-true relative displacements between poses using manual work for five open-source available datasets. With the manual work incorporating human background knowledge about the environment,

[1] http://gazebosim.org//

the ground truth quality is close-to-true, the only disadvantage is that the intensive human effort is required.
There are also highly precise indoor positioning systems, like Symeo LPR®-2D [38], which could reach the accuracy of 5-20 cm like indoor GPS. In Optitrack and Vicon[2] a multi-camera based tracking system is introduced for tracking drones, ground and industrial robotics. In [34], an external motion capture system from MotionAnalysis is used for tracking.

Landmark based reference system

The SICK NAV350 navigation scanner is used as a global reference system for the experiment. The working principle is similar to Frese's work [36], only the manual measurement of relative poses of landmarks is not required here that will be automatically measured. Before the collection of datasets, the reflectors needed by the NAV350 are mounted in the operation environment. A map of those artificial landmarks is pre-built for later providing the reference localization pose using Extended Kalman Filter. The user interface of the reference system is shown in Fig. 3.2, which includes the scan measured by the sensor (the blue dots) and the reflector landmarks (the sign which has a cross in the center of a circle). By using the map of landmarks, the reference pose of the robot can be acquired.

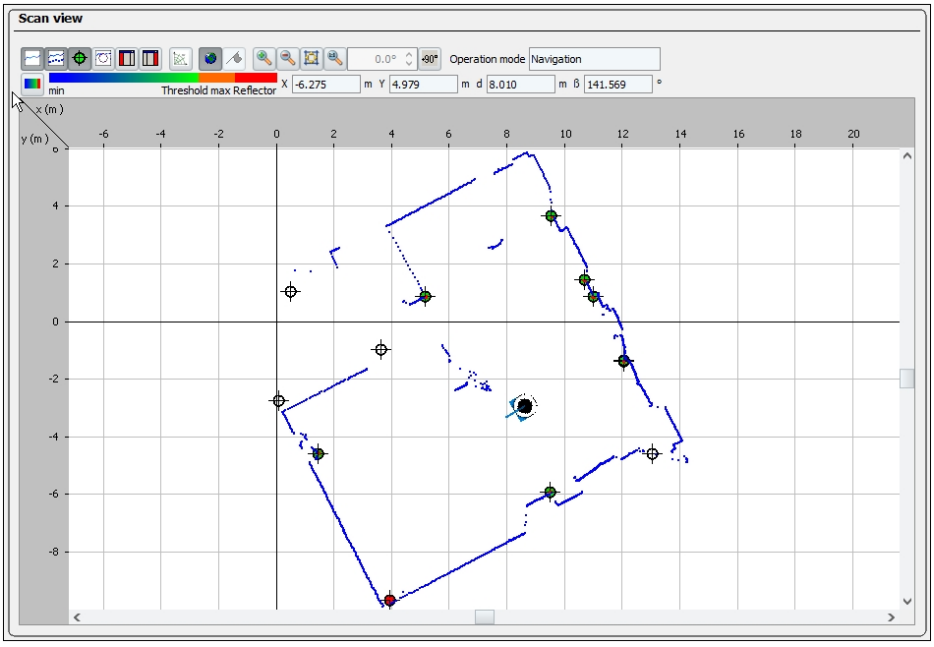


Fig. 3.2 Illustration of the SICK NAV350 based reference system (a lab environment).

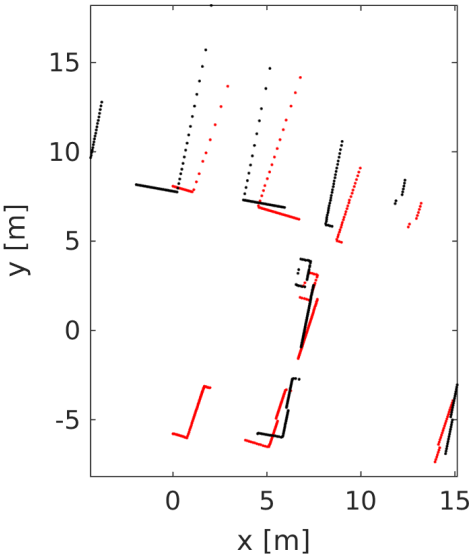
[2]https://www.vicon.com/motion-capture/engineering

3.2 Comparison of different scan matching algorithms with different laser scanners

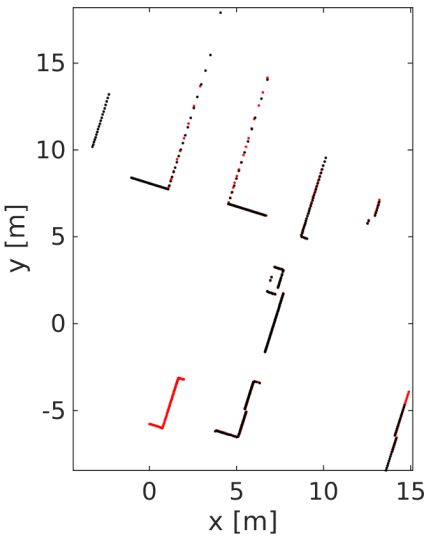
3.2.1 Introduction

A scan matching technique is a key component for the rangefinder-based Simultaneous Localization and Mapping (SLAM). A lot of scan matching techniques have been developed, and normally when a new scan matching technique is developed, it will be compared with the existing techniques, but normally only one specified type of rangefinders is used, which cannot fully demonstrate the improvement. Nowadays there are plenty types of rangefinders, which differ in the maximal range, resolution, field of view, accuracy and of course price. In this work, different rangefinders with different scan matching techniques are systemically examined for the first time, and a benchmarking system is developed for measuring the performance of each combination. The result of this work can provide suggestions for which kind of rangefinder works best with which kind of scan matching techniques in indoor environment, and therefore can be used as guidelines for the choosing of rangefinder in indoor SLAM applications to reduce the price of the system.

3.2.2 Related works



(a) Two scans at different time-steps are falsely aligned using raw odometry.



(b) Two scans at different time-steps are correctly aligned using scan matching algorithm.

Fig. 3.3 The scan matching problem.

The 2D scan matching problem can be stated as: *given a scan and a map, or a scan and a scan, or a map and a map, find the rigid-body transformation (translation and rotation) that aligns the current scan data with the reference data best.* An example is shown in Fig. 3.3.

In the past 30 years, plenty of scan matching methods have been developed. The well-known Iterative Closest Point (ICP) algorithm is one of the first algorithms for registering two different scans, it is initially developed for 3D scan matching [11], but the idea is also widely applied to 2D scan matching methods. Many variants of ICP have been developed to solve problems like, the correspondence search is too computationally expensive [22], bad results are frequent with occlusion and outliers [85]. Other variants of ICP have been developed to further improve the performance by using different distance metrics, for example Iterative Dual Correspondences (IDC) [72], Metric-Based ICP (MbICP) [75], Canonical Scan Matcher (CSM) [19] and so on. There are also approaches trying to solve the problem from other directions, for example, Hough Scan Matching (HSM) which transfers the problem from Cartesian coordinate system into Hough domain [20], scan matching using histogram correlations [16], Normal Distribution Transform (NDT) [12] which uses normal distribution transformation instead of individual points for matching, Polar Scan Matcher (PSM) [31] which uses the inherent property of laser scans in polar coordinate system. The brutal-force based correlative scan matching algorithm [83] is developed to avoid being trapped in the local minima. There are also feature extraction based scan matching methods presented in [24] [47] [46] [55].
Normally when a new scan matching method is proposed, it will be compared with the state-of-the-art methods. For example, Censi [19] compared his CSM with MbICP, classic ICP and IDC using a rangefinder with 180° field of view. By using only one specific kind of rangefinders for the performance comparison of the new method with the old ones, the effect of the properties of the laser scanner, for example, field of view, maximal range, noise level, resolution is not fully explored on scan matching algorithms. There are also papers which only focus on the comparison of scan matching techniques. In [43], the authors examined three different scan matching approaches: matching by assigning points to line segments, matching by using the cross correlation function and matching by point-to-point assignments. The first approach in [43] requires line extraction from the environment, which works only in environments that have rich polygonal features. The second approach requires a 360° rangefinder. The third approach is IDC. Lastly, the authors combined 2 methods for improving the robustness of the self-localization. In [65], the authors compared ICP, HSM and PSM for a SLAM technique with a rangefinder (Field Of View (FOV): 240°, maximal

range: 5 meter). The processing time and the accuracies of the x-y translation and rotation angle are compared.
Most of the works use only one type of rangefinder which of course cannot fully demonstrate the performance of different scan matching methods. In this work, plenty of combinations of different rangefinders and different scan matching methods will be examined to find out which kind of rangefinder works best with which scan matching technique in indoor environments, and also the threshold value of FOV and maximal range for guiding the selection of rangefinder for indoor applications. Five popular scan matching approaches are selected which are classic ICP, CSM, MbICP, PSM and NDT (2D version) for comparison. Because of the long computation time needed for each scan matching, IDC is not taken into consideration. HSM is not used here, because of its bad performance in accuracy. A short introduction of those five selected algorithms are described in the following part.

3.2.3 Scan matching algorithms

The scan matching algorithms can be classified into two classes based on whether the search of correspondence is required or not, see Fig. 3.4. In this work, due to the time effort, only the correspondence class is experimented with.

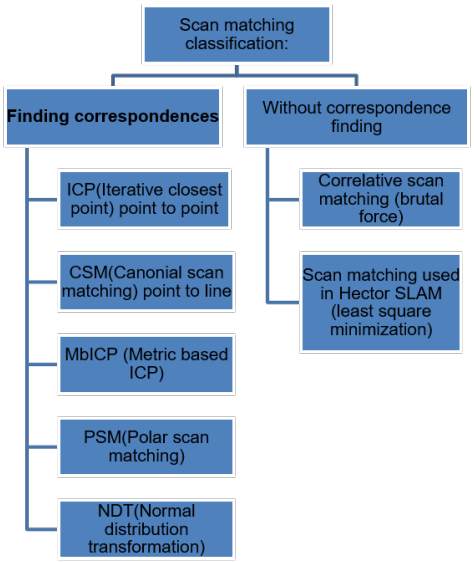


Fig. 3.4 The classification of scan matching algorithms.

Classic ICP

The classic ICP uses point to point metric as the error metric, and iteratively minimizes the error, starting from a first guess and then generating a new guess for the next generation. The

ICP algorithm will always converge to a local minimum with respect to the mean-square distance objective function in [11]. The algorithm is stated as following:

Algorithm 1 Classic ICP algorithm

1: $P \leftarrow \{p_i, i \in [1, N_p]\}$ ▷ A source point set
2: $X \leftarrow \{x_i, i \in [1, N_x]\}$ ▷ A target point set
3: $P_0 \leftarrow P, t \leftarrow [0,0,0]_t, k \leftarrow 0$ ▷ Initialization
4: **while** Not Converged **do**
5: Compute the closest points: $Y_k = C(P_k, X)$
6: Compute the registration $t = Q(P_k, Y_k)$
7: Apply the registration $P_{k+1} = t(P_k)$
8: Terminate the iteration when the change in mean square error falls below a preset threshold τ
9: **end while**

MbICP

The Euclidean distance is used in the ICP algorithm. The Metric-based Iterative Closet Point scan matching takes into account both translational and rotational inaccuracies of the sensor within the iterative closest point framework.

CSM

CSM (Or PLICP) is a point to line metric based ICP algorithm. Instead of finding the closet point, CSM finds the 2 closest points which can be formed as a line, and then calculates the distance to the line as the error metric.

PSM

Polar Scan Matching is an iterative method that takes advantage of the polar structure of the laser scan, making the correspondence search much more efficient. The current scan is projected into the coordinate system of the reference scan. Prior to matching, the current and the reference scans are preprocessed to remove erroneous measurements. The two are then aligned by minimizing the sum of the squared differences between the reference scan and the points calculated by linear interpolation using the current scan. During the alignment process the algorithm alternates between the translation estimation and the orientation estimation.

NDT

The idea behind NDT scan matching is that the environment can be characterized by a grid whose cells describe the probability of registering a measurement at a certain location thus

providing a compact representation that is piecewise continuously differentiable. The pose can then be computed by minimizing the $L2$ Norm between the probability density functions of the current and the reference scan using Newton's method.

3.2.4 Experiment

Five scan matching approaches are tested namely classic ICP, CSM, MbICP, PSM and NDT (2D version). In this section, the statistical analysis on simulated data is shown. Experiments were carried out on a single core 2.1 GHz CPU.

Data

Simulation data The main reason to do simulated experiments was the availability of the ground truth and easy configuration of the environment. A home environment simulated in the player/stage simulator [100] is used for these experiments in this work.

Simulation environment The experiment datasets were collected based on a differential-drive robot navigating in an indoor home environment (a travel of 50 m with 1680 scans) shown in Figure 3.5. The idea here is to cover most representative indoor environments, such as unstructured rooms and structured corridors.

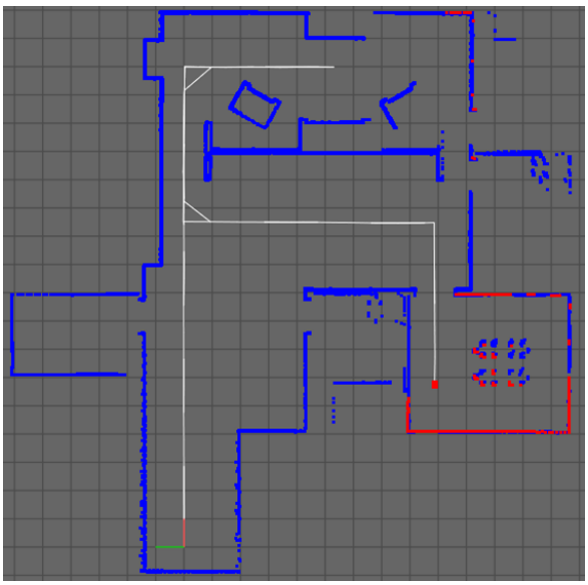


Fig. 3.5 The simulated home-like environment. The white line is the trajectory, and the red points are from a scan observation.

Rangefinders

Rangefinders used in the simulation environment are configured in different FOVs and Maximal ranges, and resolution is set to 0.5° in order to simplify the problem. The FOV starts with 60° and goes up to 270° by increasing 30° every time. The maximal range starts with 5 meters and goes up to 10 meters by increasing 1 meter every time, in addition, 20 meters is also included.

Odometry and dead reckoning

For the tests, odometry can be used as an initial guess for the scan matching methods. Dead reckoning hereby means that when the scan matching methods return an invalid estimation and the odometry is available, the invalid estimation will be replaced by the raw odometry increment.

Movement threshold for doing scan matching

Based on the global reference system, the ground truth of the robot is known. The movement threshold is defined as the movement of the rangefinder instead of the movement of the robot. When the rangefinder is placed in the rotation center of the robot, the movement of the rangefinder and the robot are the same. When the rangefinder and the robot are not in the same rotation center, the movement of the rangefinder and the robot are different. As the effect of the movement threshold for scan matching methods will be explored here, the movement of the rangefinder is selected.

Protocol

Various scan matching methods were compared by running a ROS-based [3] program. It computes the errors of the scan matching pose compared with ground truth for each method. It also stores the processing time for every scan matching. At the beginning, all variables are initialized by the ROS parameter server. All calculations are based on the data of a rosbag file given to the program. The rosbag contains the robot odometry measurements, the laser scans and the ground truth poses. As only those ROS messages with matching timestamps can be used for benchmarking, up to six messages are stored in a buffer. This is due to the fact that it is not guaranteed that the laser scanner and the odometer take measurements at the same frequency. Once the buffer is full, the program searches for synchronized triplet e.g. robot odometry poses, reference poses and laser scan messages logged at the same time.

[3] http://www.ros.org/

This is done by selecting the first messages of each type from the buffer and comparing the timestamps. If they do not match, the first rosbag message will be deleted and the process is repeated. Otherwise the information is passed to the chosen scan matching method which aligns the current laser scan to the previous one and returns the corrected pose. This pose is then compared to the ground truth by calculating the absolute and relative error. The results and the processing time are written into a text file. So they can be accessed later for the evaluation. The processed messages are deleted from the buffer. This procedure is repeated until the buffer contains less than three elements and thus no more triplets. At this time the buffer is filled again. The program terminates when there are no more messages in the rosbag file. Based on the program mentioned above, a python script is written for doing the experiments in parallel. Another python script is written for processing the results generated by the first python script. In the end, the output result will be analyzed in Matlab. In order to compare fairly, the same values for common parameters are used. The maximal iteration time is set to 50, the convergence condition for both translation and rotation are set to 0.001, and the maximal correspondence distance is set to 0.3 m. For each test, the procedure is repeated 10 times, which makes 53760 runs in total.

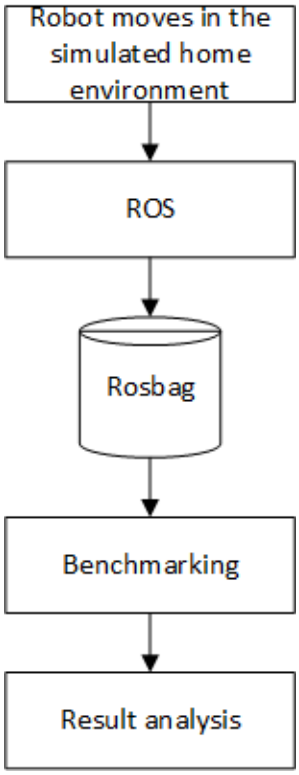


Fig. 3.6 Flowchart of the experiment.

Benchmark

Table 3.1 Parameters for investigating the effect of FOV

Parameter	Value
Low movement threshold	0.0 meter or 0.0 radian
High movement threshold	0.2 meter or 0.2 radian
Small range	5 meter
High range	20 meter
Rotation error threshold	0.005 radian
Translation error threshold	0.01 meter
Robustness threshold	0.8

The performance is given by the processing time, precision, and robustness. Robustness is defined as the percentage of precise scan matching times. Precise scan matching is defined as both the translation and rotation estimations are below certain thresholds. Those parameters are listed in Table 3.1.

3.2.5 Results

The results of experiments are summarized in Fig. 3.7, 3.8 and 3.9. The rotational and translational errors are plotted separately, but precise scan matching should have small errors both in translation and rotation. In Fig. 3.7, 3.8 and 3.9, the solid lines stand for cases which have odometry input, and the dashed lines stand for cases which have no odometry input. Due to space constraints, only plots about emphasized aspects of the benchmark are shown here. Here mainly the influence of FOV on different scan matching methods is introduced, because the influence of maximal range of rangefinders is smaller compared with FOV which could be seen from the results. The reason is also obvious, that the size of a normal indoor home environment is limited.

Robustness For most of the scan matching methods, the performance increases with increasing field of view. In general, the scan matching performs better when the odometry is available. CSM and MbICP are more depend on the odometry input, and classic ICP and NDT are not so depend on the odometry. MbICP depends on the odometry input as an initial guess the most. In Fig. 3.7 (a) and (b), when odom is used, MbICP and CSM outperform the others, but in Fig. 3.7 (c) and (d), when the movment between two scans is high, CSM can still keep the robustness, but MbICP drops significantly.

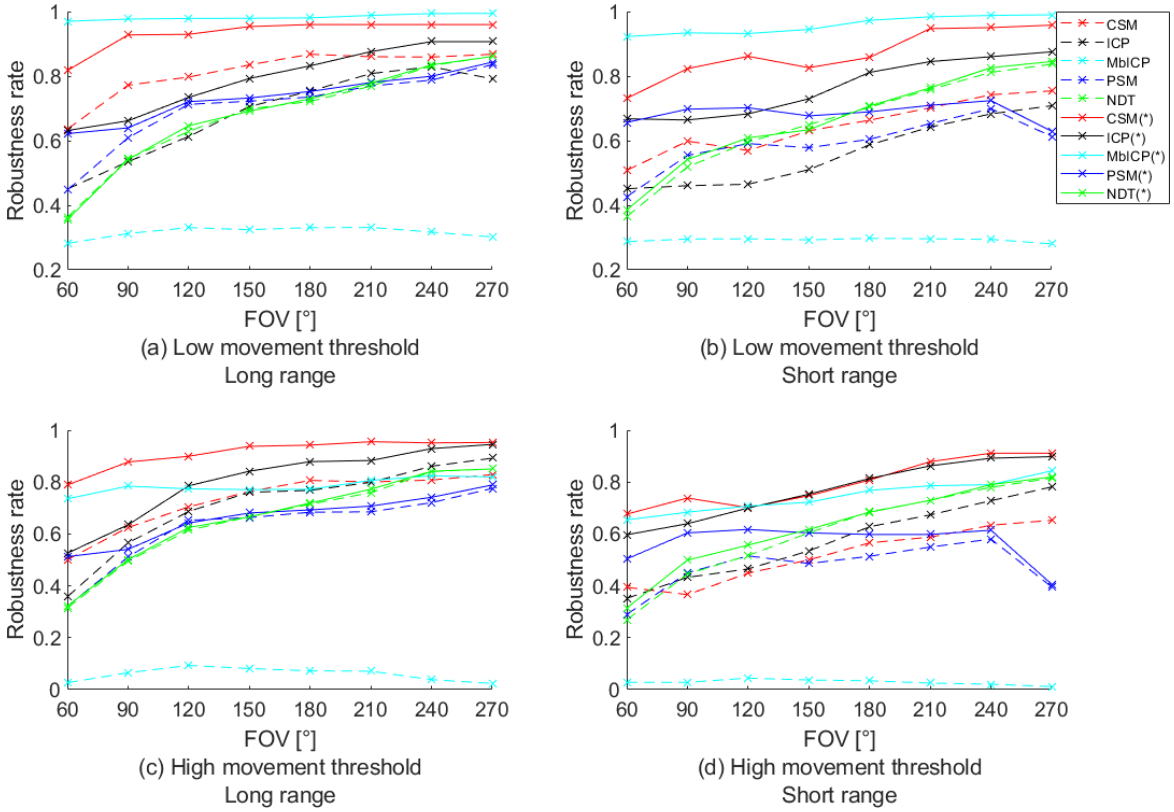
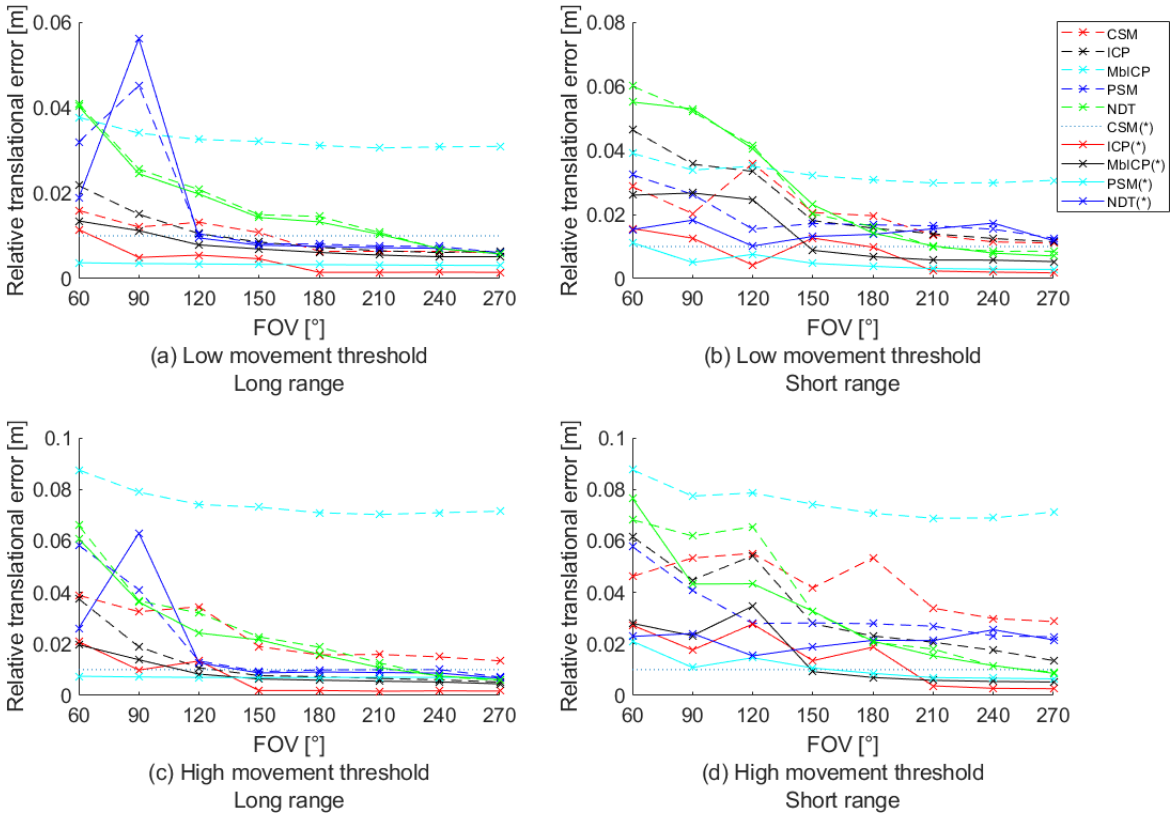
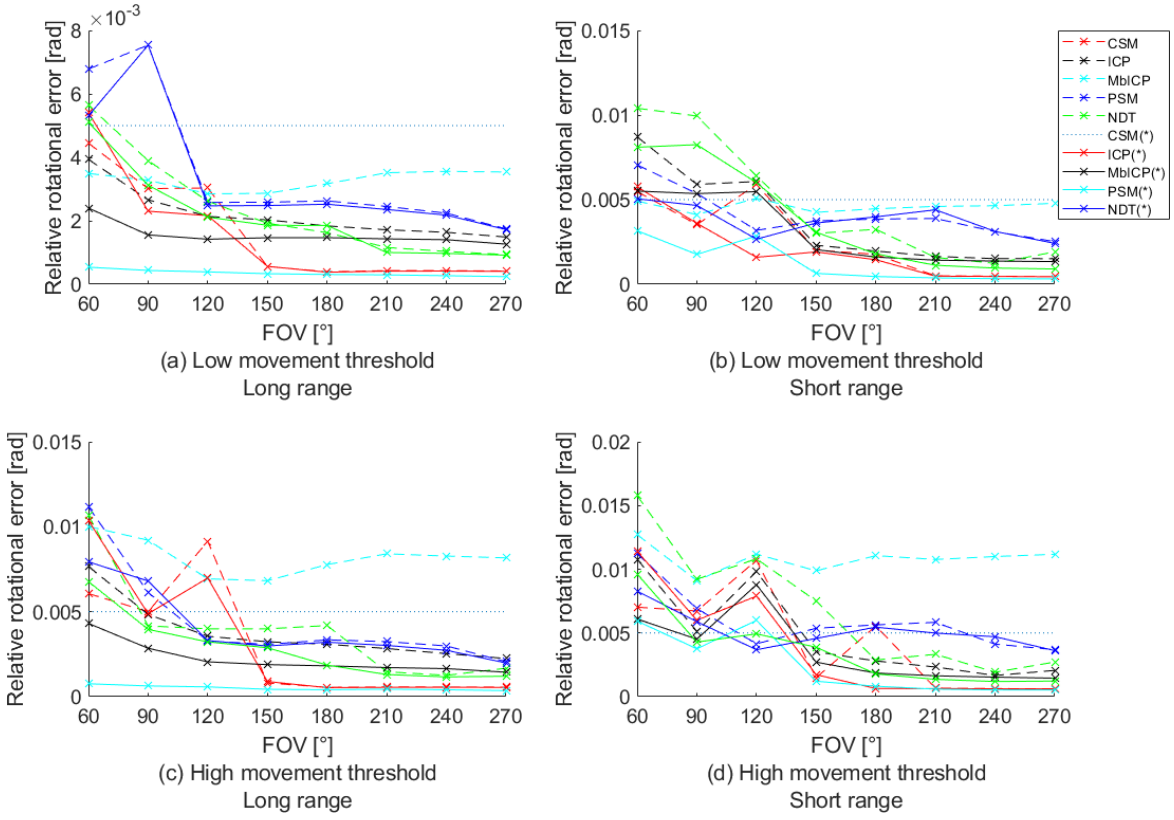


Fig. 3.7 The robustness comparison of CSM, ICP, MbICP, PSM and NDT.



(a) The translational accuracy comparison.



(b) The rotational accuracy comparison.

Fig. 3.8 The accuracy comparison of CSM, ICP, MbICP, PSM and NDT.

Accuracy As expected, with the increasing of FOV, the error goes down. For cases whose FOV are greater than 150°, the errors drop only slightly. As shown in the Fig. 3.8, with low movement threshold and long range, scan matching methods go into coverge at certain FOV. CSM and MbICP can reach the smallest error.

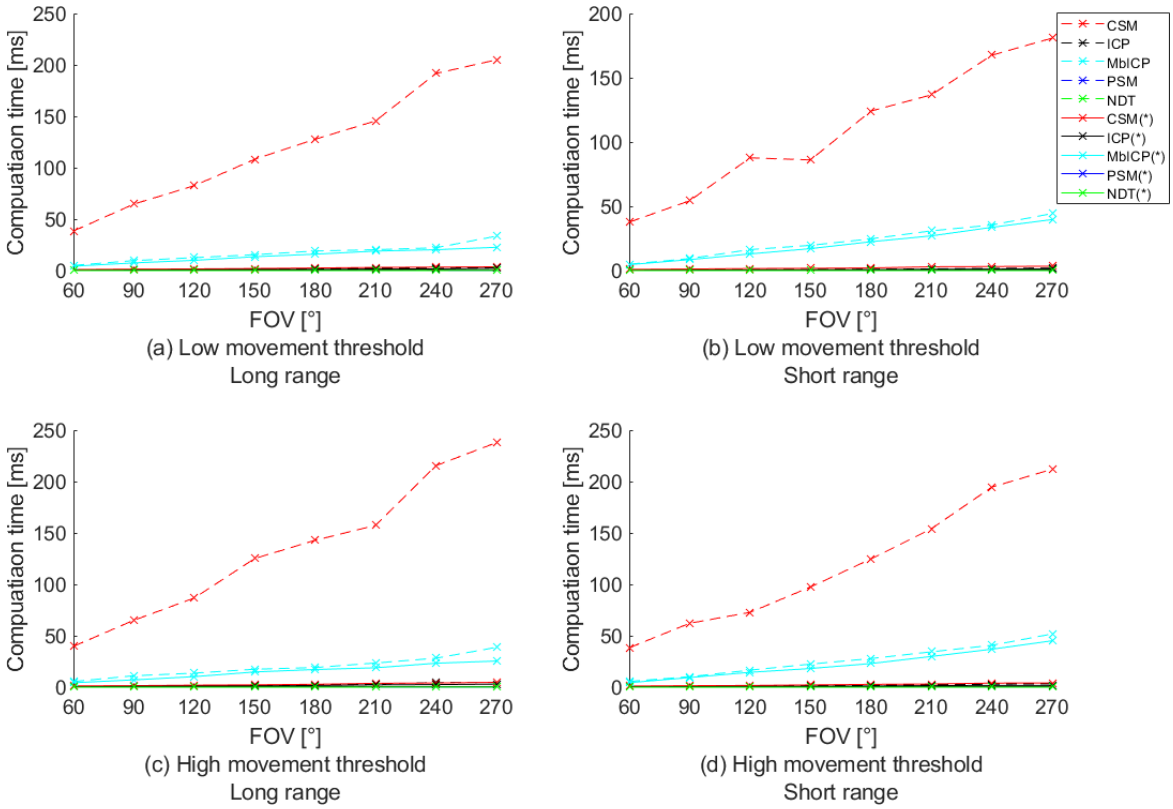


Fig. 3.9 The processing comparison of CSM, ICP, MbICP, PSM and NDT.

Processing time In general, with the increasing of FOV (resolution is constant), the processing time increases. When there is no initial guess given, the CSM costs obviously more time than other methods. The processing time of other methods is not affected by there is odometry as initial guess or not. One thing need to be mentioned is that MbICP is the second most time expensive method.

Influence of FOV sizes For all methods, the field of view increases and the accuracy and robustness mostly also increase. Here for the purpose of demonstrating the effect of field of view for rangefinders, 4 scenarios are selected, which are combination of high/low movement threshold and long/small range. The definition of thresholds used here is shown in Table 3.1. The results are shown in Table 3.2 and Table 3.3. In the Table 3.2 and Table 3.3, the

minimal field-of-views for meeting the corresponding threshold is displayed for robustness, average translational error and average rotational error. The column Min FOV is the value which can meet all robustness, rotational and translational thresholds. The value NA stands for no value could satisfy the corresponding threshold. When the odometry input exists, MbICP has the smallest minimal requirement for field of view when the movement is low, but when the movement is big, the minimal requirement for field of view increases significantly. When there is no odometry available, there is no field-of-view which can satisfy all the three thresholds. When the odometry input exists, CSM has the second smallest minimal requirement for field of view when the movement is small. For NDT2D, the minimal required FOVs for all 4 corresponding scenarios in Table 3.3 have the same value as in Table 3.2, although as seen in Table 3.2 and Table 3.3, NDT2D needs a big FOV to get good results for all 4 scenarios. The effect of the presence of an odometry or not on NDT2D is trivial. Because of the robustness problem, the minimal required FOV values for PSM are either the biggest FOV value or no value.

Table 3.2 Required field-of-view with odometry input

Scenario	**Method**	**Robustness**	**Mean rot/tr error**	**Min FOV (°)**	**Processing time (ms)**
Low movement long range	CSM	60	90/90	90	1.40
	ICP	180	60/120	180	1.14
	MbICP	60	60/60	180	4.21
	PSM	270	120/120	270	0.01
	NDT	240	90/240	240	0.06
Low movement small range	CSM	90	90/120	120	1.56
	ICP	180	150/150	180	0.70
	MbICP	60	60/90	90	8.48
	PSM	NA	90/NA	NA	NA
	NDT	240	150/240	240	0.03
High movement long range	CSM	90	90/90	90	1.51
	ICP	150	60/120	150	1.21
	MbICP	210	60/60	210	18.84
	PSM	NA	120/150	NA	NA
	NDT	240	90/240	240	0.07
High movement small range	CSM	180	150/210	210	3.10
	ICP	180	90/150	180	1.17
	MbICP	270	90/180	270	45.14
	PSM	NA	120/NA	NA	NA
	NDT	270	90/270	270	0.04

Table 3.3 Required field-of-view without odometry input

Scenario	Method	Robustness	Mean rot/tr error	Min FOV (°)	Processing time (ms)
Low movement long range	CSM	150	60/180	180	127.70
	ICP	210	60/150	210	2.12
	MbICP	NA	60/NA	NA	NA
	PSM	270	120/150	270	0.01
	NDT	240	90/240	240	0.07
Low movement small range	CSM	NA	90/NA	NA	NA
	ICP	NA	150/NA	NA	NA
	MbICP	NA	60/NA	NA	NA
	PSM	NA	120/NA	NA	NA
	NDT	240	150/210	240	0.03
High movement long range	CSM	180	150/NA	NA	NA
	ICP	240	90/150	240	4.48
	MbICP	NA	NA/NA	NA	NA
	PSM	NA	120/150	NA	NA
	NDT	240	90/240	240	0.08
High movement small range	CSM	NA	150/NA	NA	NA
	ICP	NA	150/NA	NA	NA
	MbICP	NA	NA/NA	NA	NA
	PSM	NA	120/NA	NA	NA
	NDT	270	180/270	270	0.03

3.2.6 Conclusion

This section presented a simulation-based performance comparison of different rangefinders with different scan matching techniques for indoor robotics, examined and analyzed the influence of maximal range and field of view of rangefinders, displacement size of two scans and the input of odometry on different scan matching methods. In conclusion, when there is odometry input, CSM and MbICP are good choices for rangefinders with limited FOV and limited maximal range. When there is no odometry input, CSM can still keep good performance with the cost of high computation time, but when there is no odometry input, the performance of MbICP drops significantly, so MbICP is not suggested when odometry is not available. ICP performs next to CSM, but its processing time is more stable. For NDT2D, a wide FOV is required, but small maximal range is already enough. One interesting property of NDT2D is the independence from odometry, which may be interesting for odometry-free robot systems. In general, PSM does not perform good compared to other methods. The processing time for scan matching algorithms in indoor home environment is affected by the field of view of rangefinders rather than the maximal range of rangefinders. Because in general in indoor environment, the length of the room is limited, so field of view plays a more important role for the number of beams which will result in the processing time. When there is no initial guess given, CSM can reach 200 ms of processing time, which is slow for scenarios requiring a high updating rate. MbICP is the second slowest method which needs 20 ms, but this is already acceptable for indoor robotics. The average processing time of the rest methods are all below 10ms. In Chapter 5, different scan matching methods with

different real sensors are compared in real environments, and the result is applied to the choosing of scan matching methods in rangefinder based SLAMs.

3.3 EMB-SLAM: An embedded efficient implementation of Rao-Blackwellized Particle filter based SLAM

3.3.1 Introduction

Simultaneous localization and mapping (SLAM) algorithms are an essential component for autonomous mobile robotics to be able to operate in a priori unknown environments. In the last two decades, plenty of SLAM algorithms have been developed and also a number of optimizations have been done for those algorithms. But rarely optimization approaches to low-cost and energy-efficient embedded systems which are suitable for indoor robotics have been done. The benefit of the development of embedded system should be explored. With the emerging of new technologies (multi core, ARM® NEON™) which can greatly accelerate the processing speed, rethinking the implementation of algorithms should be done. In this work, a new embedded efficient Rao-Blackwellized particle filter based Simultaneous Mapping and Localization (EMB-SLAM) implementation is presented. It is based on the co-design with the multi-core embedded hardware, a SLAM algorithm and an optimization methodology. EMB-SLAM is tested with real datasets. Experiments show the real-time performance of this implementation, and demonstrate that the embedded system is suitable for realizing SLAM applications under real time constrains.

To efficiently solve many tasks desired to be carried out by mobile robots including logistic, home-care, search and rescue, or guidance, robots need models of their environment. Building maps has therefore been a major research focus in the robotic community over the last decades. Learning maps and localizing the robot itself in the maps at the same time is the simultaneous localization and mapping (SLAM) problem. In the literature, plenty of SLAM algorithms, like EKF SLAM [102], FastSLAM [76], GMapping [42], Graph SLAM [41] and so on, have been proposed by researchers. Most of them focus on the improving of accuracy, consistency or robustness, only few works deal with the implementation and optimization of SLAM for low-cost embedded systems [101] [66] [70]. Traditional SLAM algorithms are too computational to run on embedded devices. At least laptop level computation devices are needed which are not suitable for low-cost and energy efficient purposes. Researchers have already realized and paid attention to deal with this problem. Vincke et al. [101] introduced an embedded system based SLAM application. They presented an efficient implementation of the EKF SLAM algorithm on a multi-core CPU architecture, and proved

that the SLAM could be performed and optimized on the ARM based architecture. A camera was used as input for feature extraction and matching, and visual landmarks were used in the application. Lee et al. [66] presented an embedded visual SLAM. In Lee's approach, an upward-looking camera was used for indoor environment perception which could take advantage of the orthogonal structure and the invariance property of indoor environment. An optimized graph optimization was integrated into the standard Kalman filter. Llofriu et al. [70] presented an embedded particle filter SLAM implementation. In this approach, a camera and identifiable artificial landmarks were used, and the map consists of landmarks. In [39], OpenMP [14] is used for the parallelizing of GMapping. Here, we further explored two other popular task-based parallel programming modes in C++ programming language, Boost Thread [15] and Intel® TBB [25]. A new efficient implementation of calculating the score of a scan at a specified pose in the map for the scan matching problem is used here by pre-computing a lookup table instead of doing computational expensive closest point searching every time. The well-known GMapping is compared with our EMB-SLAM implementation. The work presented in this section includes embedded RBPF-SLAM implementation, optimization and performance evaluation method. The section is organized as follows. After explaining the theory of Rao-Blackwellized particle filter in SLAM problem, the lookup table and parallel programming models are described, and implementation details are provided. Experiments carried out on real datasets are presented. Finally, results and conclusions are discussed. Results prove that EMB-SLAM has gained speedup using lookup table and parallel programming models.

3.3.2 Rao-Blackwellized Particle Filter

Particle filter is a Bayes filter. Particle filters are an efficient way to represent non-Gaussian distribution. There are two basic principles: particles are a set of state hypotheses and they are survival-of-the-fittest. In the SLAM content, SLAM can be represented as a sample based Posterior by particles as following:

$$S = \{< s^{(i)}, w^{(i)} > | i = 1, ..., N\} \tag{3.3}$$

$$p(x) = \sum_{i=1}^{N} w^{(i)} \cdot \delta_{s^{(i)}}(x), \tag{3.4}$$

where $s^{(i)}$ is the ith sample, $w^{(i)}$ is the weight of the ith sample, and N is the number of samples.

Particle filters have been successfully applied to localization task, and later researchers extended the particle filter to solve SLAM problem too. The difference between localization

and SLAM can been seen in the following:

$$p(x|m,z,u) \longrightarrow p(x,m|z,u), \tag{3.5}$$

where x is the robot pose, m is the map, z is the observation, u is the control input. The left side is the localization posterior, and the right side is the SLAM posterior. For the localization task, the map is known. For the SLAM task, the map is initially unknown. If robot poses are known, it is easy to build the map by adding observations into corresponding known poses. Rao-Blackwellization is the process of splitting the robot pose estimation (localization) and the mapping. The factorization is first introduced by Murphy in 1999 [33] as following:

$$p(x_{1:t},m|z_{1:t},u_{0:t-1}) = p(x_{1:t}|z_{1:t},u_{0:t-1}) \cdot p(m|x_{1:t},z_{1:t}), \tag{3.6}$$

where t is the time step, $x_{1:t}$ is the robot trajectory, m is the map, $z_{1:t}$ is the observations and $u_{0:t-1}$ is the movements. $p(x_{1:t},m|z_{1:t},u_{0:t-1})$ is the SLAM posterior. $p(x_{1:t}|z_{1:t},u_{0:t-1})$ is the robot path posterior. $p(m|x_{1:t},z_{1:t})$ is mapping with known poses which is easy. In RBPF-SLAM, every particle carries a potential trajectory of the robot and its own map. Each particle survives with a probability proportional to the observations relative to its own map. All particles together represent a joint posterior about the poses of the robot and the map. Since each map is quite big in case of grid maps and each particle has its own map, the number of particles should be kept small. In [42], scan matching poses are used as input to the RBPF instead of the raw odometry. Fewer particles are needed, since scan matching provides a locally consistent pose correction which has smaller error than the raw odometry. According to [42], the optimal proposal distribution with respect to the variance of the particle weight is as following:

$$p(x_t|m_{t-1}^{(i)},x_{t-1}^{(i)},z_t,u_{t-1}) = \frac{p(z_t|m_{t-1}^{(i)},x_t)p(x_t|x_{t-1}^{(i)},z_t,u_{t-1})}{p(z_t|m_{t-1}^{(i)},x_{t-1}^{(i)},u_{t-1})}. \tag{3.7}$$

Using that proposal, the computation of weights becomes

$$w_t^{(i)} \propto w_{t-1}^{(i)} \cdot p(z_t|m_{t-1}^{(i)},x_{t-1}^{(i)},u_{t-1}) \tag{3.8}$$

$$= w_{t-1}^{(i)} \cdot \int p(z_t|x')p(x'|x_{t-1}^{(i)},u_{t-1})dx'. \tag{3.9}$$

More detailed derivation can be found in [42].

Algorithm 2 Rao-Blackwellized Particle Filter SLAM

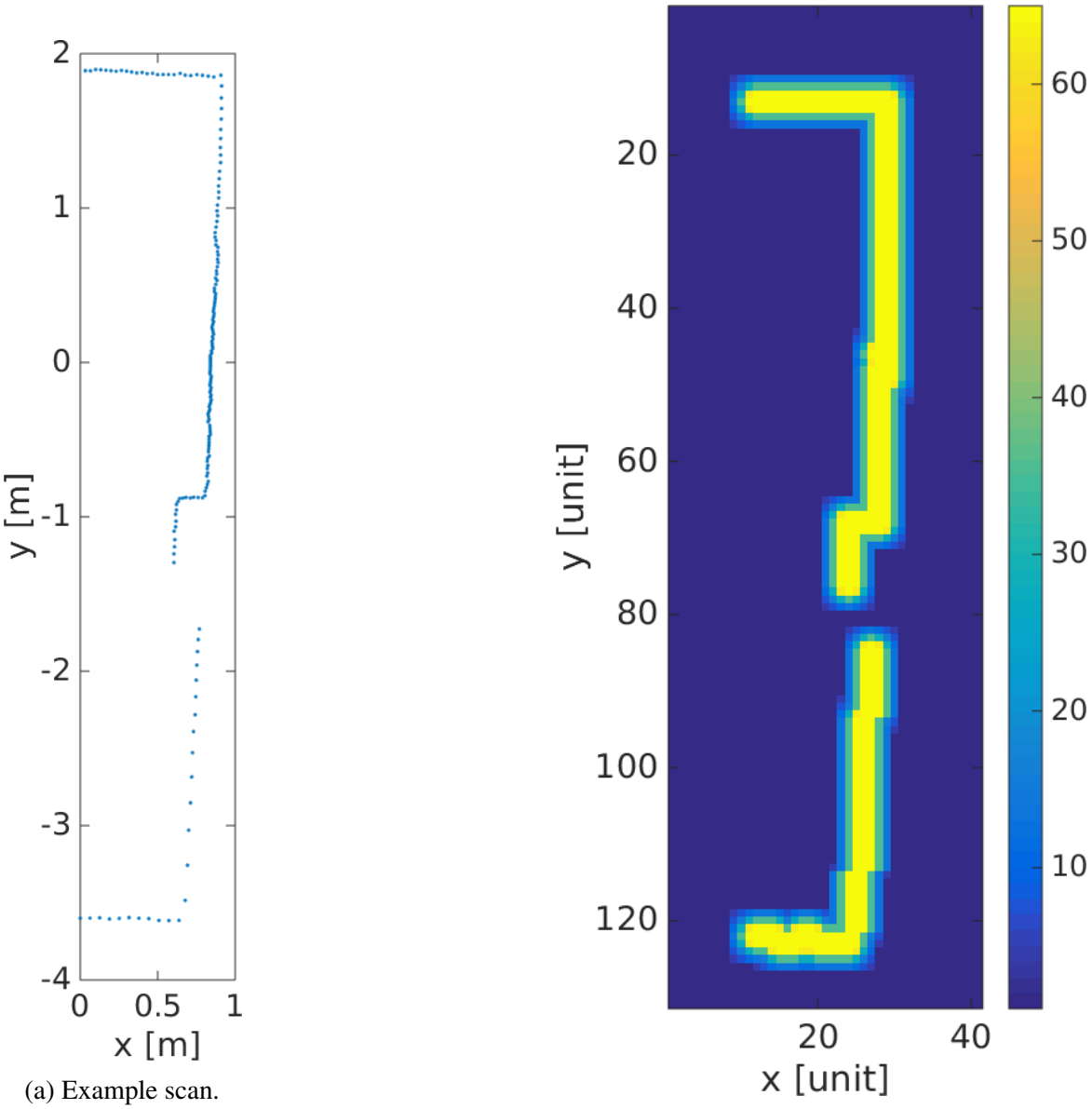
for $i = 1 : N$ **do**
    $w_0^i = 0$ ▷ Initialization
    $m^i \leftarrow z_0$
    $x_0^i \leftarrow 0$
end for
while `New` z_t u_t **do** ▷ Particle Update
    for $i = 1 : N$ **do**
        Draw sample $\hat{x_t^i} \sim p(x_t | x_{t-1}^{(i)}, u_{t-1})$
        Scan matching using $\hat{x_t^i}$ as initial guess to find $x_t^i = argmax p(z_t^i | m^{(i)})$
        Update the particle weight $w_t^{(i)}$ according to (3.8)
    end for
end while
Calculate total weight $t = \sum_{i=1}^{N} w_t^{(i)}$ ▷ Normalize
for $i = 1 : N$ **do**
    $w_t^{(i)} = t^{-1} w_t^{(i)}$
end for
Calculate $N_{eff} = \frac{1}{\sum_{i=1}^{N} (w_t^{(i)})^2}$ ▷ Resample
if $N_{eff} < N_T$ **then**
    Resmapling
end if
for $i = 1 : N$ **do** ▷ Map update
    $m^i \leftarrow z_t$
end for

3.3.3 Improved scan matching algorithm based on GMapping

GMapping is one of the most famous and widely used SLAM algorithms in the robotic community. In Gmapping, scan matching algorithm "vasco" from Carnegie Mellon Robot Navigation Toolkit (CARMEN) is used. "vasco" finds the local maximum based a gradient descent search. In most cases, vasco is able to compute the best alignment between a reference map $m_{t-1}^{(i)}$ and the current observation z_t given an initial guess $x_t^{'}$. Gradient descent or hill-climbing based method could be trapped in the local maximum, so a more throughly method is proposed in [83]. In [83], a brutal-force searching is performed to overcome the problem of being trapped in the local maximum, further a lookup-table is calculated for fast score computation. With the use of a lookup table, the computational expensive closest obstacle point searching is avoided. Scan points are transformed into the map coordinate, and projected to the lookup table. The score of one scan point is the value in the hit cell in the table. The total score is the sum of score of single points. The accuracy of the brutal-force method is limited by the resolution of the lookup-table, the higher the resolution, the higher the accuracy. The problem with high-resolution is that the memory consumption is $O(n^2)$, where n is the number of pixels per meter. In order to balance the accuracy and the memory consumption, the brutal-force method is used in the work for rough estimation, and the gradient descent method is used for fine estimation. The resolution for look-up table used here is 0.05 m. An example of the look-up table is shown in 3.10b.
The calculation of likelihood $p(z_t|m_{t-1}^{(i)}, x_t)$ is calculated based on the "beam endpoint model", where every individual scan beam is assumed to be independent, so the likelihood could be calculated as following:

$$p(z_t|m_{t-1}^{(i)}, x_t) = \prod_j p(z_t^j|m_{t-1}^{(i)}, x_t), \tag{3.10}$$

where z_t^j is the jth beam of a scan. The likelihood of a beam is computed based on the distance between the endpoint and the closest obstacle from that point. Normally a Gaussian function is used for approximating $p(z_t^j|m_{t-1}^{(i)}, x_t)$.



(a) Example scan.

(b) Look-up table.

Fig. 3.10 A look-up table example at 0.05m resolution. Unit in the right figure stands for 0.05m.

3.3.4 The parallel programming models

A brief introduction of the programming models are given in this section.

Boost Thread

Boost Thread enables the use of multiple threads of execution with shared data in portable C++ code. Classes and functions for managing the threads themselves are provided, along with others for synchronizing data between the threads or providing separate copies of data specific to individual threads. Boost Thread has been supported by the standard library since C++11. The difference between the standard C++11 thread and Boost thread are in many regards, Boost Thread offers extensions.

Threading Building Blocks (TBB)

TBB is a portable C++ template library for task parallelism. Like Boost.Thread, TBB does not require special compilers, any reasonable modern C++ complier is able to build a program using TBB. It is designed to promote scalable data parallel programming. TBB takes tasks and maps onto threads in an efficient manner for execution. In this way, the programmer is abstracted from the platform-specific threading libraries, and also from the details of the hardware, for example the the number of physical cores available in the hardware.

OpenMP (Open Multi-Processing)

OpenMP consists of a number of complier directives, introduced into the code using the #pragma keyword , library routines and environment variables. OpenMP uses a portable, scalable model that gives programmers a simple and flexible interface for developing parallel applications. The master thread forks off a number of threads which execute blocks of code in parallel. Different from Boost.Thread and TBB, OpenMP requires a complier that supports OpenMP. Most modern C++ compliers support OpenMP.

3.3.5 Hardware and software configuration

In order to test and validate the EMB-SLAM algorithm, experiments were conducted with real datasets. For the evaluation, the computation unit used here is a multi-processor embedded board computer (ODROID-XU4). The board integrates a Samsung Exynos5422 CPU. The Exynos 5422 is equipped with four “big” cores (ARM® Cortex® -A15™ up to 2.0GHz) and four “little” cores (ARM® Cortex® -A7™ up to 1.4 GHz). The Exynos5422 supports HMP (Heterogeneous Multi-Processing). HMP refers to the ability for the Operation System

(OS) to use and schedule threads on all 8 cores at the same time, putting those threads with low performance requirements on the “little” cores and high performance threads on the “big” cores. The multi-core structure is suitable for accelerating the processing speed of RBPF-SLAM by parallelizing the processing of particles. A laptop (Lenovo T540p) is used for comparison. More detailed information about ODROID-XU4 and Lenovo T540p is shown in Table 3.4, and a picture of ODROID-XU4 is shown in Fig. 3.11. For both computation units, the operation system is Ubuntu 16.04 LTS, and the Robot Operation System (ROS) "Kinetic" is used as middle-ware for interfacing various sensors and the robot controlling.

Table 3.4 Computation units

Name	ODROID-XU4	Lenovo T540p
CPU	Samsung Exynos5422	Intel Core i7-4700MQ
RAM	2 Gbyte	16 Gbyte
Disk	32 Gbyte	500 Gbyte
Power	5V/4A	20V/6.75A
Size	83 x 58 x 20 mm	340.5 x 233 x 30.5 mm
Price	$100	$1800

Fig. 3.11 Embedded computation unit ODROID-XU4

3.3.6 Algorithm implementation and evaluation

Task analysis

As shown in Algorithm 2, the RBPF-SLAM is similar to scan matching on a per-particle base with some extra noise, which can be divided into five modules. Those five modules are pose prediction, scan matching, weight calculation and normalization, resampling and map update. Particle filter update: for each particle, a) draw a sample from motion model based on the last pose and control input, calculate an improved pose distribution using scan to map

based scan matching algorithm to align the scan measurement to the last map; b) predict new particle by sampling from the improved pose proposal; c) calculate the weight by evaluating the likelihood of poses sampled from the newly predicted particle pose combined with the scan measurement in the map. Particle weight normalize: For each particle, calculate the normalized weight. For all particles, calculate the effective sample size [57] which indicates how equally the weights of particles are distributed. Particle resample: If the distribution of weights varies a lot, particles will be resampled. The resampling method used here is multinomial resampling [32]. The particles which are not survived will be replaced by the survived ones. Map update: For each particle, the map is updated with the estimated pose and the corresponding scan measurement.

Evaluation datasets and methods

Two widely used datasets for benchmarking SLAM in the community, which are Intel and ACES datasets, are used here for benchmarking purpose as shown in Table 3.5. The particle update happens when the robot moves 0.5 meter or rotates 0.5 radian. The number of processed nodes is also shown in Table 3.5. In [39], ROSBAG format is used for evaluation, and the disadvantage of using ROSBAG format is that if the processing speed of the SLAM algorithm is slower than the incoming speed of sensor measurements, many scan measurements will be jumped. Different from the work in [39], the datasets are firstly processed in the CARMEN format to make sure every scan which should be processed is processed for a fair comparison. The processing time now only depends on the algorithm itself, if the algorithm is fast enough, many test time will be saved. ROSBAG format is used in the end for the validation of the real-time performance. The number of particles is set to 30, 60 and 90. The benchmark criterion includes mean error of estimated poses, calculation time per iteration, and maximal CPU and memory usage of the program. a) Accuracy is measured by using the relative error metric described in previous Section. 3.1.2. b) Processing time is measured by calculating the difference between the start time and the end time of the processing loop inside the program. c) CPU and memory usage is measured outside of the program by using sysstat[4], which could continuous monitor the CPU and memory usage. The maximal value is used here for the benchmarking purpose. As commonly used in the parallel programming, the speedup metric is used here to measure the improvement of a parallel program. The speedup s is defined as:

$$s = T_{SP}/T_{MP}$$

[4]http://sebastien.godard.pagesperso-orange.fr/

where T_{SP} represents the total processing time of the serial processing method, and T_{MP} the time of the multi-processing method in the same exact condition.

Table 3.5 Datasets information

Name	Intel	ACES
Duration (s)	2691	1365
Size (m^2)	56×58	28.5×28.5
Nodes	1140	575

Computation Time analysis

As discussed in the task analysis part, the major modules of RBPF-SLAM are defined. The default implementation is not optimized, and the computation time (unoptimized) of each module is measured. Two open-source available datasets which are widely used in the robotic community are used here for the analysis of the computation time of each component. The result is shown in Table 3.6.

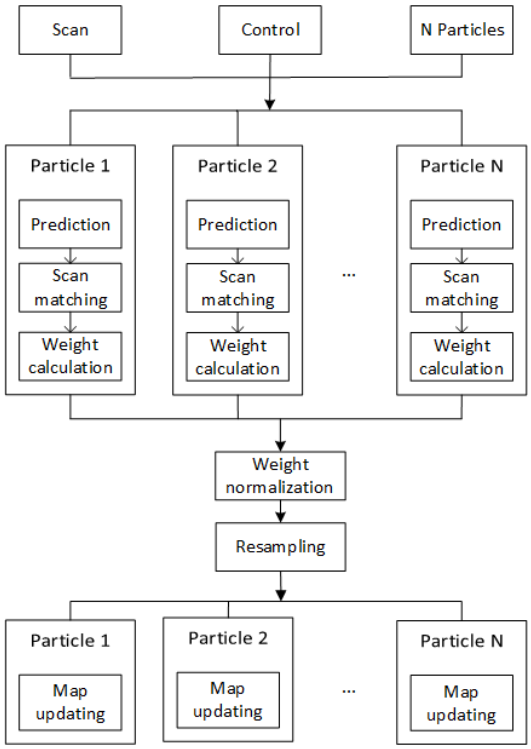


Fig. 3.12 Processing loop of parallel Rao-Blackwellized particle filter

Hardware-software optimization

As shown in Table 3.6, the code section particle update needs the most time (more than 94 %) in one processing loop. Because of the multi-core architecture, multiple threads can run in parallel which can accelerate the processing speed of parallelizable programs. The particle filter update module includes the update of each particle which can be scheduled into independent threads. In this case the number of CPU cores is eight. The map update module is also parallelized for each particle to take advantage of the multi-core architecture for accelerating computation speed. Furthermore the map update only happens when the robot has traveled certain distance or certain angular or the time interval since last map update is more than certain time. The optimized Rao-Blackwellized particle filter processing loop is shown in Fig. 3.12.

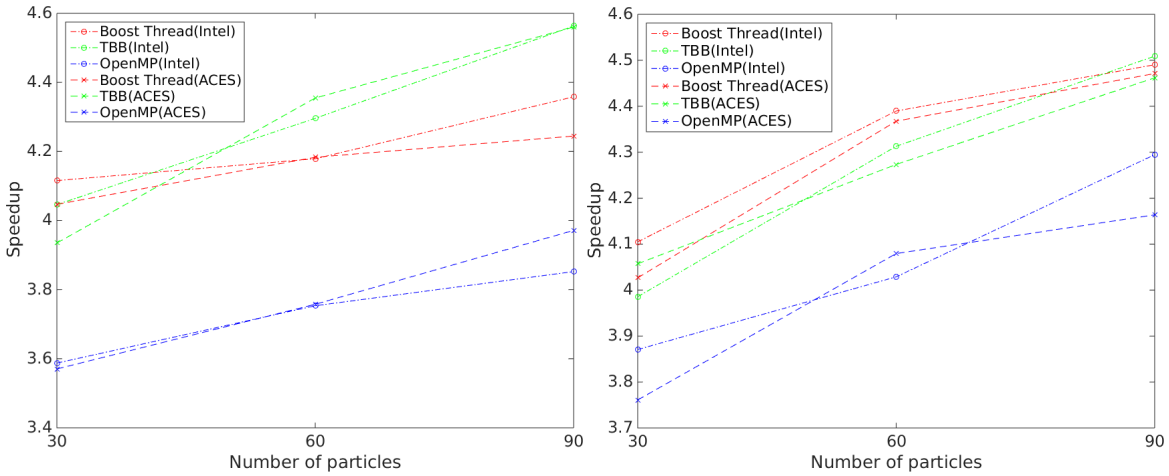
3.3.7 Experimental results

The comparison between the time usage of the scan matching algorithm used in GMapping and the author's new improved method is shown in Table 3.6. The new improved method using a pre-computed lookup table has a mean speedup of 9.9 times on ODROID-XU4, and a mean speedup of 1.8 times on Lenovo T540p. The cost for the speedup is that more memory is used for storing the look-up table, which is a Time-Memory Tradeoff. The maximal memory usages for serial processing and multi-processing are shown in Fig. 3.14a and 3.14b. Because of the use of lookup table, the author's serial implementation requires in average 3.68 % more system memory compared with GMapping' implementation for Intel dataset, and in average 1.78% more for ACES Building dataset. In both test datasets, the memory usages of three parallel programming models satisfy the order that $MEM_{BoostThread} > MEM_{TBB} > MEM_{OpenMP} > MEM_{Serial}$. All three parallel models require more memory than the serial model, because extra memory is needed for the scheduling and management of threads. The memory usage grows with the increasing number of particles. The memory usage of OpenMP is the smallest among all three parallel programming models, and the memory usage using Intel TBB is the second smallest one. The memory usages on the PC (refers to Lenovo T540p) and ODROID (refers to ODROID-XU4) under the same condition are the same, so the figure for the memory usage on the PC is not shown here. The maximal CPU usage of a serial program is one kernel, and for parallel programs, all kernels of the CPU could be used. The speedup for three different parallel programming models with different number of particles are shown in Fig. 3.13a and Fig. 3.13b. The speedups on PC for both datasets are all below 8, which is the number of CPU threads. On Odroid, the maximal speedup is 4.56, which is also below the number of the CPU kernels (8). In Odorid,

the speedup of Boost thread $s_{BoostThread} \in [4.05, 4.36]$ and Intel TBB $s_{TBB} \in [3.84, 4.56]$ are clearly better than OpenMP $s_{OpenMP} \in [3.57, 3.97]$. The difference between ODROID and PC is that, the PC CPU has 4 kernels (8 threads) with the same computation power, and the ODROID CPU has 4 "big" cores and 4 "little" cores. From the experiment results, one can see that the speedups of Boost Thread and Intel TBB are better than OpenMP in both platforms. The total execution time of different approaches are shown in Table 3.7 and 3.8. Under the same configuration, PC has an average speedup of 4.6 times against ODROID.

Table 3.6 The mean processing time of the particle update method in GMapping and EMB-SLAM

Dataset	Particles (N)	ODROID-XU4 (ms)		Lenovo T540p (ms)	
		GMapping	EMB-SLAM (serial)	GMapping	EMB-SLAM (serial)
Intel	30	2815.908	313.663	131.498	80.285
	60	5702.258	625.396	256.326	163.91
	90	8489.265	923.716	381.66	237.883
ACES	30	2999.833	276.989	136.209	70.886
	60	5961.353	564.622	270.042	142.14
	90	8833.583	849.584	400.798	213.83



(a) ODROID-XU4.

(b) Lenovo T540p.

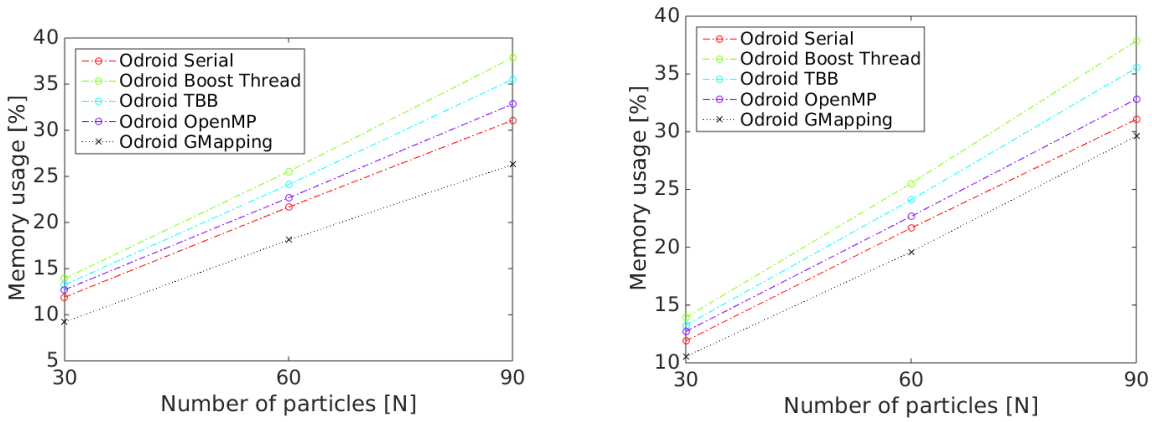
Fig. 3.13 The comparison of speedup obtained using multi-processing.

Table 3.7 The execution time of different approaches for Intel dataset in seconds

Platform	Particles (N)	GMapping	EMB-SLAM			
			Serial	Boost Thread	TBB	OpenMP
ODROID-XU4	30	3940.99	1081.78	262.835	267.354	301.552
	60	7712.9	2161.49	517.222	503.126	575.869
	90	11820.1	3241.55	743.759	710.352	886.142
Lenovo T540p	30	189.722	246.049	59.9404	61.7368	63.5674
	60	377.135	490.338	111.703	113.692	121.708
	90	566.766	734.004	163.463	162.784	170.911

Table 3.8 The execution time of different approaches for ACES Building dataset in seconds

Platform	Particles (N)	GMapping	EMB-SLAM			
			Serial	Boost Thread	TBB	OpenMP
ODROID-XU4	30	2166.51	599.855	148.22	152.438	168.011
	60	4280.6	1206.19	288.329	276.968	320.991
	90	6495.46	1806.74	425.637	396.197	454.924
Lenovo T540p	30	103.128	130.426	32.3908	32.1457	34.6789
	60	206.172	261.972	59.9816	61.3039	64.2162
	90	305.488	391.294	87.5161	87.6986	93.9779



(a) Intel dataset.

(b) ACES Building dataset.

Fig. 3.14 The comparison of memory usages of different multi-processing models. The memory usage of GMapping is used a reference.

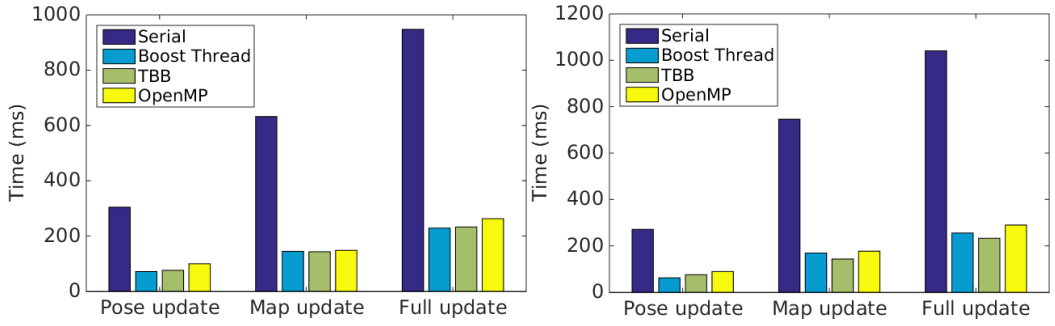
3.3.8 Discussion

According to Table 3.6, the average speedup of serial processing EMB-SLAM for particle updating is 9.9 times against the un-optimized version in GMapping on ODROID-XU4, which is a significant speedup due to the computation load is significantly decreased by replacing the computational closest point search for every point in the scan at every specified pose with a pre-computed lookup table. The pre-computed lookup trades extra memory usage for the speedup, but as shown in Fig. 3.14a and 3.14b, the increasing of maximal memory usage is not significant. The maximal speedup of the parallelized EMB-SLAM for particle update is 4.56 against the serial processing version of EMB-SLAM on ODROID-XU4. The reason for the improved processing speed is by taking advantage of the multi-core CPU architecture. ODROID-XU's CPU Exynos 5422 is an eight core CPU, theoretic the processing speedup s can be 8, on one hand, the accelerating rate of the processing speed is normally smaller than the number of CPU cores, because of the extra time and resource are needed by the OS to handle the thread list. On the other hand, Exynos 5422 has a "big" CPU and a "little" CPU, because of the big-little structure, the processing thread assigned to the "little" core will be slower than threads assigned to the "big" core. The total execution time will depend on the thread which has the longest execution time. As shown in Table 3.7 and 3.8, the mean execution time of one update for all particles using parallelized EMB-SLAM is at least 3.57 times speedup of the mean execution time of single particle's update process. Then considering the time for scheduling all those threads and the allocation and free of memory, the at least 3.57 times speedup against the serial processing version is a reasonable result. In the experiments, the effect of the number of particles is explored, as shown in Fig. 3.14a and Fig. 3.14b. As every particle has its own path and map. The average processing time is linearly related with the number of particles used. The maximal memory usage also linearly related with the number of particles used. A comparison of the accuracy with the well-known GMapping is explored. The result in Table 3.9 shows the accuracy between RBPF SLAM and GMapping is similar, but the average processing time of RBPF SLAM is 12 times faster than the average processing time of GMapping with the same number of particles. In other words, EMB-SLAM is able to use more particles, but could still keep the real-time performance. Because the SLAM posterior is represented preciser with more particles, the improvement done in this work improves not only the processing speed of RBPF based SLAMs, but also improves the robustness of the SLAM algorithms. The average execution time per update using EMB-SLAM for Intel Research Lab dataset and ACES Building dataset on ODROID-XU4 is shown in Fig. 3.15. Pose update includes scan matching and likelihood evaluation. Map update includes the update of the map and also

the update of the lookup table. Full update is the full update including predict, pose update, resampling and map update.

Table 3.9 Mean translational error ε_{trans} and mean rotational error ε_{rot}

Algorithm	Intel		ACES Building	
	ε_{trans} (m)	ε_{rot} (radian)	ε_{trans} (m)	ε_{rot} (radian)
EMB-SLAM (N=30)	0.041±0.001	0.010±0.0003	0.060±0.002	0.009±0.0003
GMapping (N=30)	0.038±0.001	0.009±0.0003	0.064±0.002	0.010±0.0003



(a) Intel dataset (N=30). (b) ACES dataset (N=30).

Fig. 3.15 The average execution time per update using EMB-SLAM for Intel Research Lab dataset and ACES Building dataset on ODROID-XU4.

3.3.9 Conclusion

In this work, a RBPF SLAM is implemented and optimized for a multi-core architecture embedded platform. A benchmark system is also introduced for evaluating the performance of the approaches. The results show the EMB-SLAM implementation is 12 times faster than GMapping without a huge increase in the Memory consumption. The high process speed can potentially be used for improving the robustness of mobile robotic with high speed. It shows that energy efficient and low-cost embedded devices are suitable for SLAM approaches.

3.4 Summary

The scan matching algorithm is a key component of the laser scanner based SLAM algorithms. A lot of scan matching techniques have been developed, and normally when a new scan matching technique is developed, it will be compared with existing techniques, but normally only one specified type of laser scanners is used, which cannot fully demonstrate

the improvement and the performance difference between different laser scanners. Further the well-known laser scanner based GMapping algorithm is widely used in the robotic community for SLAM, but it is not able to run efficiently on low-cost and energy-efficient embedded devices.

In this chapter, a simulation based comparison of different scan matching algorithms with different laser scanners is conducted, and the result gives us a better understanding of the performances of different scan matching algorithms with different laser scanners. And based on the understanding of the principle of the RBPF-SLAM and the computation time analysis of the scan matching algorithm used in GMapping, a new embedded efficient implementation of RBPF-SLAM (EMB-SLAM) is presented.

The EMB-SLAM parallelizes the most computational task into different threads. Three parallel programming models are implemented and compared for the speedup performance. For the scan matching algorithm, a time-memory trade-off is made by using extra memory storing a lookup table instead of searching the closest correspondences at each iteration. In total, a speedup of 12 times is achieved on an embedded platform.

Chapter 4

Contributions to 2D real-time robust graph based SLAM

4.1 Introduction

4.1.1 The history of graph optimization based SLAM and the theory

Pose-graph optimization has been widely used in the SLAM research community to suppress the nonlinear error by optimizing robot or sensor poses [45][79]. The idea of graph based SLAM was first proposed by Lu and Milios [72]. They were historically the first to represent the SLAM prior as a set of edges between robot poses to use all the frames for the consistent registration. Lu and Milios's algorithm was successfully implemented by Gutmann and Konolige [44]. The Graph based SLAM algorithm was first presented in the information theory form by Thrun et al. [95]. Recently the formulation of the Graph based SLAM algorithm is further extended to model the state vector using relative information instead of absolute information [48]. Many works have been done in this area for effectively solving such problems. At the beginning, Graph based SLAM was identified as a batch algorithm, not an online algorithm like EKF SLAM or RBPF SLAM. Recent developments show that the Graph based SLAM could also be incremental or online. Typical Pose-Graph SLAM implementations consist of two distinct phases. First, the information from sensors are pre-processed to identify potentially spurious readings as well as to associate sensor readings with the parts of the environment that have been previously observed. This is known as the front-end, which is specific to the sensors used and the application domain. The Graph SLAM back-end uses optimization algorithms to provide an estimate of robot poses and a representation of the environment.

4.1.2 Graph optimization related sensors and applications

Graph SLAM front-end is mostly sensor and robot specific. Depends on the sensor used, the graph SLAM could be classified into two classes, one is LiDAR based which uses Laser scanners or LiDARs, the other is vision based which uses monocular cameras [2] [93], stereo cameras [71] or RGB-D cameras [104]. Graph SLAM is widely used for land vehicles [9], unmanned aerial vehicles and unmanned underwater vehicles [71].

4.1.3 The components of graph optimization based SLAMs

Two major components of graph based SLAMs are graph building and graph optimization. Many graph optimization algorithms have been proposed for accurately and effectively solving the graph (SLAM back-end), and thanks to the maturity of the SLAM problem, a lot of open-source libraries for graph solving, like Ila's SLAM++ [49], Kümmerle's General Graph Optimization(g2o) [86], Kaess's iSAM2 [54] are available. So solving the graph is not a big problem, the building of the graph is, the discussion in this chapter will be focused on the graph construction part. The front-end is responsible for the building of the graph. Depends on the topological distance of two connected nodes, the edges can be classified as basic edges and loop closure edges. Basic edges are normally built from odometry like sensors which connect two neighbor nodes. Loop closure edges are built by finding the transformation between two topologically far away, but physically close nodes. Loop closure could also be described as the robot has the ability to recognize places that it has visited before. The constraints between loop closure nodes will be added to the graph as edges. Loop closure edges are essential for pose graph optimization algorithms, those extra added edges provide important constraints in the cost function to be minimized by "moving" all the nodes which is done by graph optimization. The building of loop closure edges can be further divided into three parts, which are preprocessing step for generating loop closure hypotheses, transformation calculation and verification of loop closure hypotheses.

4.1.4 State-of-the-art of the graph building techniques

For the construction of the graph, in the early times, the graph is built manually, with the help of well designed user interfaces, the workload could be decreased. In [60], the initial estimates are computed by SLAM algorithms at hand, and then for indoor environments the initial estimates are inspected by a human who will decide accept, refine, or discard a match and also add missing relations by incorporating the background knowledge about the

environment; for outdoor environments, additional relations could be derived from satellite image data, and a human operator is still needed to accept or decline all relations found by the approach. For the purpose of generating close to ground truth poses, the manual work is tolerable, for practical real-time usage, it will not be practical. Later, many effective methods have also been proposed to build the graph (SLAM front-end) automatically without manual work. In [41], the addition of a new basic edge is done when the robot moves more than 0.5 meters or rotates more than 0.5 radians. The searching of candidate nodes for loop closures is limited to the ones whose 3σ marginal covariances contains the current robot pose. The validation technique of the candidate loop closures is purely based on the matching goodness in the paper. Since the uncertainty of the state estimate can be very large before closing a large loop, traditional data association based on geometric information together with the estimated uncertainty may not be adequate for loop closure detection. Olson [82] presented a front-end with outlier rejection. Nüchter et al. [81] presented a loop closure method for 3D SLAM. The detection of loop closures is heavily researched in the vision based approaches. Zhang et al. [106] explored a multi-restraint method of closed-loop detection based on the time limit, data association and the location deviation. Normally for doing loop closure detection, the current observation need to be compared with all the old observations, with the increasing of the internal map, eventually limiting the on-line processing. Angeli et al. [5] presented an online method that makes it possible to detect when an image comes from an already perceived scene using local shape information. Their approach extends the bag of visual words method used in image recognition to incremental conditions and relies on Bayesian filtering to estimate loop-closure probability. For LiDAR based approaches, scan feature descriptors are used to extract features from scans to improve the computational cost, like FLIRT [98], key point feature [55]. In order to reduce the size of the dimension of laser scans, 20 features are calculated [40]. Machine learning algorithm Adaboost is used to train the detection of loop closures. Perhaps the most similar work to the author's is [45], no extra feature extraction is needed.

Ratter et al. [87] presented their encoder-free graph building method. OG-MbICP [13] alignment for robot pose estimation is used instead of directly using odometry as basic constraints, and a local map which is accumulated from multiple neighbor scans is used instead of a single scan. A new local map is created when the robot moves more than 2 meters. Instead of a one by one chained structure of the basic edges, the tree structure is constructed. When a loop closure is found and it is good enough to combine them together, the old node will be the parent of the new node. The selection of candidate nodes for loop closure detection is similar to Olson's approach [83], by comparing the absolute value of the difference in global position between the centers of two local maps with the covariance,

only close enough nodes will be further exterminated. The rough transformation between two local maps are computed by the correlation of their histograms, then the result is refined by ICP. The rejection of bad loop closures is solely based on the convergence time of ICP and the number of matching points in the ICP algorithm. Himstedt et el. their graph based framework for building maps from laser range scans. The basic edge building is based on a point-to-line based ICP algorithm, the loop closure detection is based on the FLIRT features extracted from single scans, and switchable loop closure constraints are used as robust back-end to identify false positive loop closures. The map quality is further improved by applying Sparse surface Adjustment algorithm to jointly optimize the settings of robot poses and laser measurements. They have only provided results on their own datasets, so it is hard to compare the author's method with theirs. Most importantly, their framework post-processes datasets, which is not aimed to run in real-time. Yin et al. [105] presented a full off-line approach for Graph SLAM. The building of basic edges is based on point-to-line matching algorithm for estimating basic edge transformations. A filtering of dynamic parts in the raw LIDAR data is done, then a conditioned-hough-transform and linear regression based line segment detection is called to detect line features from the rest of LIDAR data. In the end, a global line feature map is constructed. Puente et al. [28] presented a general framework for the handling structure constraints into a feature-based graph SLAM. They demonstrated the improvement by including structure constraints at the case no global data association methods is employed. Here the author wants to argue that the result could be further improved with efficient global data association methods. Labbe and Michaud [61] presented an on-line multi-session graph-based SLAM. For the loop closure detection, the bag-of-words approach is used to calculate the transformation between two nodes using RGB images and depth images, and the transformation can be refined with the ICP algorithm using laser scans. A loop closure is accepted when a minimum of inliers from RANSAC are found. A memory management method is proposed to ensure the real-time performance of loop closure detection by limiting the number of locations used for loop closure detection, which of course sacrifices the number of loop closured to be found. Mur-Artal et al. presented a bag of words place recognizer with ORB features [93] for camera based SLAM, the selection of a loop candidate requires the current keyframe matches with 3 previous keyframes, and the validation of loop closure hypotheses is done by geometrical verification using similarity transformation. To solve the computational cost, in the vision community, PTAM [78] splits the tracking and mapping into different threads on CPU. Because there are two threads running in parallel so that the computation cost of the tracking is not affected by the mapping. A visual feature based data association algorithm is presented for loop closure detection by Gil et al. [37]. A distance transform is applied to the local occupancy grids first for a

better SURF feature extraction extracted from local sub-maps, as Gil et al. suggested that the distance transformation emphasizes the structure of the environment. The data association for both basic edges and loop closures is based on the SURF feature matching. A landmark-scan based Graph SLAM is applied to AGV localization in a large-scale warehouse with SICK NAV350 laser scanner which is able to output laser scans together with reflectors (landmarks) [9]. Because of the long detection range (70 meters) and 360 degree Field-of-View of SICK NAV350 and not so large environment ($130m \times 100m$) compared with the maximal range of the laser scanner, they claim they don't need extra loop closure techniques. High level feature representation like rectangles is incorporated into a featured based graph SLAM for structured environments [29]. They showed impressive results from the using of structure constraints, but they were lack of a global data association algorithm for large loop closures which they need to do manually. A stereo graph SLAM for Autonomous Underwater Vehicles which optimizes the trajectory and treats the feature out of the graph is presented in [71]. The basic edge is added when the robot travels every n meters, and the transformation is calculated from Visual odometry. The search of loop closure candidates is limited to nodes inside a spherical Region of Interest centered at the current camera pose. The estimation of the transformation of loop closures is achieved by Perspective N-Point algorithm. This method is only suitable for small environments. g2o is used as the back-end optimizer. In [45], the building of basic edge is based on scan-to-submap principle, and the building of loop closing constraints is based on scan to submap or submap to submap principle. The branch-and-bound approach and several precomputed grids are used for improving the real-time performance. Ceres scan matching is used here for calculating transformation. A workstation is needed for the computation. A hybrid metric-topological 3D occupancy grid maps for large scale mapping is presented in [89], metric sub-maps and a global topological graph are used to represent the environment. A NDT based graph SLAM in presented in [50]. A hybrid method which combines NDT and the appearance similarity method [53] is presented to find the appropriate loop based on RGB-D camera. A genetic search and fractional distance metric based scan matching algorithm (FGSM) [27] is presented for establishing transformation estimation between loop closure constraint using 2D laser scanner under unknown initial condition. Loop closure candidates are detected from camera data or are manally defined, then the FGSM is used to establish the transformation, last two verification metrics are used for validation loop closure candidates.

After a loop closure hypothesis is found, it is important to validate the loop closure to prevent erroneous loop closures which could destroy the global map. Corso [27] ascribed the poor loop closure transformation to either that the transformation is falsely identified or the geometry is ill-conditioned. Two verification metrics are used, one is correlation

metric which compares the shared geometry between matched scans using a 2D histogram. The author argues that the bin size must be small enough to distinguish falsely identified transformation, which will lead to the too small goodness value, because the number of points in the bin is too small. Another metric which is called complexity metric measures the complexity of the overlapping geometry, which is calculated by the ratio of eigenvalues from the matrix composed from shared points. The complexity metric is able to represent the complexity of the environment, for example, if the environment is a long corridor, the metric will be very small, if the environment has many corners, the metric value will be very high. The high complexity of environment has a high probability to lead to correct transformation, so it is also used in this work as one of the verification criteria.

Even though the validation step work well in most cases, there is still no perfect front-end SLAM which produces graphs without outliers. How to deal with outliers (false positive loop closure hypotheses) is of great importance for the robustness of the SLAM. Bad loop closures undermine the robustness of pose graph SLAM algorithms. To improve the robustness of pose graph SLAM, many researchers have presented work to improve the front-end by correctly rejecting data association errors and false positive loop closure detections. For robust pose graph SLAM, many researchers have also done approaches in the back-end part. Sunderhauf and Protzel [91] presented their back-end system which is able to recognize and reject outliers during the optimization by using switchable constraints. Switchable constraints which mean that the constraints in the graph are not fixed is achieved by introducing a new type of hidden variable in to the problem formulation. The switchable approach achieves impressive result even under high percentage of outliers, the limitation of this method is for the case when the graph is very sparse. A large part of the generated false positive loop closures in [91] could be easily rejected in the front-end by limiting the search space using error propagation law and effective validation techniques. Lee et el. presented their robust back-end with Bayesian network formulation, an additional set of variables is introduced as weights for the loop constraints, and solved the problem with Classification Expectation Maximization algorithm [64]. Agarwal et al. [2] proposed a method called dynamic covariance scaling to generalize classical gating and dynamically reject outliers based on switchable constrains, the improvement is that they avoid the use of additional variables for constraints subjected to be an outlier in the optimizer which leads to faster convergence and better performance in sparse graphs. Latif et al. [62] presented the Realizing, Reversing and Recovering (RRR) algorithm to detect and remove past incorrect loop closures. The assumption of their algorithm requires that the front-end generates all basic edges close to truth. The consensus-based detection of false positive loop closures is achieved by firstly clustering the loop closure assumptions based on the same portions of the trajectory, then testing the loop closure clusters and

every loop closure in every cluster based on the χ^2 test. RRR algorithm rejects false loop closure, while those previous algorithms keep the false loop closures with a low weight. Sunderhauf and Protzel [92] compared the Switchable Constraints, Max-Mixture Models and RRR to robust pose graph SLAM, and they concluded that no one of the three evaluated algorithms performed clearly superior on all of the examined datasets. Latif et al. [63] further compared and analyzed those robust Graph SLAM back-ends by using the benchmark dataset from KITTI Vision Benchmark, they examined the need for tunning parameters of different algorithms , and they argued that maintaining false positive loop closures with low weight leads to more fill-in in the pose graph and at the same time could result false path planning solutions. Further, Pfingsthorn and Birk [84] proposed a generalized graph SLAM for solving local and global ambiguities through multimodal Mixture of Gaussians and hyperedge constraints.

State-of-the-art of the graph SLAM In the literature, the building of the graph is often limited to small to middle size environment, especially for real-time usage. Mostly the authors process the recorded data as a whole off-line. The real-time graph construction for large scale environment is not fully researched yet. A lot problems will occur due to large scale environment. The difficulties include *Perceptual aliasing* in the environment, large drift after long driving, keep real-time performance in large scale environment. *Perceptual aliasing* means that two different places can be perceived as the same. For example, in a building, it is nearly impossible to determine a location solely with the visual information, because all the corridors may look the same. Because the computation time maybe very large due to the current observation is ambiguous to many previous visited place, but at certain point the robot may stand still, then maybe there is no computation done. The time plays a important role in the real-time system, to achieve this goal, the system needs to be good designed. Contributions from the author are a new real-time graph slam framework and a fast loop closure hypothesis generator.

Efficient implementation The SLAM problem currently has two major directions. One direction is Bayesian state-space model based, which includes Extended Kalman Filer, Particle Filter, and Rao-Blackwellized Particle Filter. The Bayesian based methods are front-end solutions, which means they mainly process the most recently messages and discard the messages acquired in the past. To fulfill the real-time requirement of the system, discarding the old messages is not wrong, but of course loss of information. In general, the Bayesian based methods work well, but they cannot correct the errors in the past. The possible inconsistency in EKF is determined by its close to linear assumption which relies on a first-order linear ap-

proximation and is close related to the partial observability of the SLAM problem. Therefore, another direction for solving SLAM problem tries to use all the information and optimizes them in the end to get a better result. All the information in the past are added into a graph (mathematically a large matrix) which is made of nodes (robot poses) and edges (constraints between different nodes). The graph optimization method has two major stages, one is the building of the graph, and another is the optimization of the graph. For the optimization of the graph, normally the Gauss-Newton algorithm and Levenberg-Marquardt algorithm are used. The Gaussian-Newton algorithm may get worse result during the iterative optimization phase, and the Levenberg-Marquardt algorithm may trap in the local minima. Yes, both of them are not perfect, but they work well in most cases. Recent research showed that the optimization based SLAM methods improves the consistency of SLAM as the Jacobians are repeatedly computed at the most recent parameter estimate, and in the limit at the optimal state as the solution converges [48]. Normally the graph SLAM formulation could includes features in the state vector, like points, lines or segments, the features and robot pose will be optimized at the same time. It is also possible to formulate the Graph SLAM without features. Rectangle featured based graph SLAM is also presented, but the environment needs to be structured which limits its usage, even though it is a nice approach to explore high level of features for optimization.

4.1.5 Description of the existing problems

Scan matching related problems The scan matching algorithm doesn't consider the inherent property of the scans which contents observed area and unknown area. If only the points itself are considered, and the search region is not precise enough, mismatch will happen, for example, the thickness of the wall can be merged to zero. The physical property of the environment could be taken into consideration, for example, new research results in the thesis, by the definition of a unique normal vector of lines in the environment, the author could tell the difference between two sides of a thin wall.

Drawbacks of current graph building and loop closure techniques The execution time for loop closure detection could affect the real-time performance of the system. For achieving real time performance, loop closure detection which is the most important and computational expensive part must be efficient. The quality of the map will heavily rely on the quality of loop closure assumptions. With false loop closure assumptions, the real environment will not be recovered. State-of-the-art graph optimization based SLAMs have difficulties in large scale indoor scenarios.

Not considering the robot state in the SLAM system The robot state is not modeled into the SLAM system. A new research result in the thesis, the whole process of Graph SLAM could be further optimized by taking the robot state into consideration.

4.1.6 Structure of the Chapter

This chapter is organized as follows. Section two presents work related to existing data association and loop-closure identification techniques. This chapter has presented a range-bearing sensor based graph-SLAM algorithm especially addressed to improve the localization of indoor robots equipped with a laser scanner. The graph management is based on the iSAM library [54] and the data association is reinforced by exploring the error propagation law and the line segment feature of the environment. The efficiency of the system is further improved by the parallel implementation of the graph building and graph optimization. The robot state is modeled into the system for further backwards optimization ability.

4.2 Preliminaries

4.2.1 Error propagation

Error propagation and node graph building

When a new node is added into the graph, the mean and covariance is also calculated. The last pose is $(x_0, y_0, \theta_0)^T$, the new movement is $(\Delta_x, \Delta_y, \Delta_\theta)^T$. The new mean pose $(x, y, \theta)^T$ is calculated as following

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} cos\theta_0 & -sin\theta_0 & 0 \\ sin\theta_0 & cos\theta_0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \Delta_x \\ \Delta_y \\ \Delta_\theta \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \\ \theta_0 \end{bmatrix} \tag{4.1}$$

.

The new covariance is calculated by following *error propagation law* [6]

$$C_Y = F_X C_X {F_X}^T \tag{4.2}$$

, where C_X is a $n \times n$, C_Y a $p \times p$ covariance matrix and F_X some matrix of dimension $p \times n$.

In 2D case, the error propagation between two connected node is shown as following:

$$F_X = \begin{bmatrix} cos\theta_0 & -sin\theta_0 & 0 \\ sin\theta_0 & cos\theta_0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

$$C_{\Delta_X} = \begin{bmatrix} \sigma(\Delta_x,\Delta_x) & \sigma(\Delta_x,\Delta_y) & \sigma(\Delta_x,\Delta_\theta) \\ \sigma(\Delta_y,\Delta_x) & \sigma(\Delta_y,\Delta_y) & \sigma(\Delta_y,\Delta_\theta) \\ \sigma(\Delta_\theta,\Delta_x) & \sigma(\Delta_\theta,\Delta_y) & \sigma(\Delta_\theta,\Delta_\theta) \end{bmatrix} \tag{4.4}$$

$$C_X = \begin{bmatrix} \sigma(x_0,x_0) & \sigma(x_0,y_0) & \sigma(x_0,\theta_0) \\ \sigma(y_0,x_0) & \sigma(y_0,y_0) & \sigma(y_0,\theta_0) \\ \sigma(\theta_0,x_0) & \sigma(\theta_0,y_0) & \sigma(\theta_0,\theta_0) \end{bmatrix} \tag{4.5}$$

$$C_Y = F_X C_{\Delta_X} {F_X}^T + C_X \tag{4.6}$$

which F_X is the transformation function between the old pose and new pose, C_{Δ_X} is the covariance between the transformation which could come from scan matcher or motion model, C_X is the covariance in the old pose, and the covariance at the new pose is C_Y.
To calculate the search space between two nodes inside a loop closure hypothesis, directly accumulating the error between those two nodes from consecutive nodes between them is often prone to accumulate too large error (search space), especially when there are loop closures found before. The error has already be minimized. Here a shortest path between the two nodes are searched by using Dijkstra's Algorithm, if there are loop closures found before, the shortest path will also contain the loop closure edges, then the error propagation will be done based the shorted path.
As shown in Fig. 4.1, the ellipses represent uncertainty at each corresponding step, the confidence probability for those ellipses is 0.999.

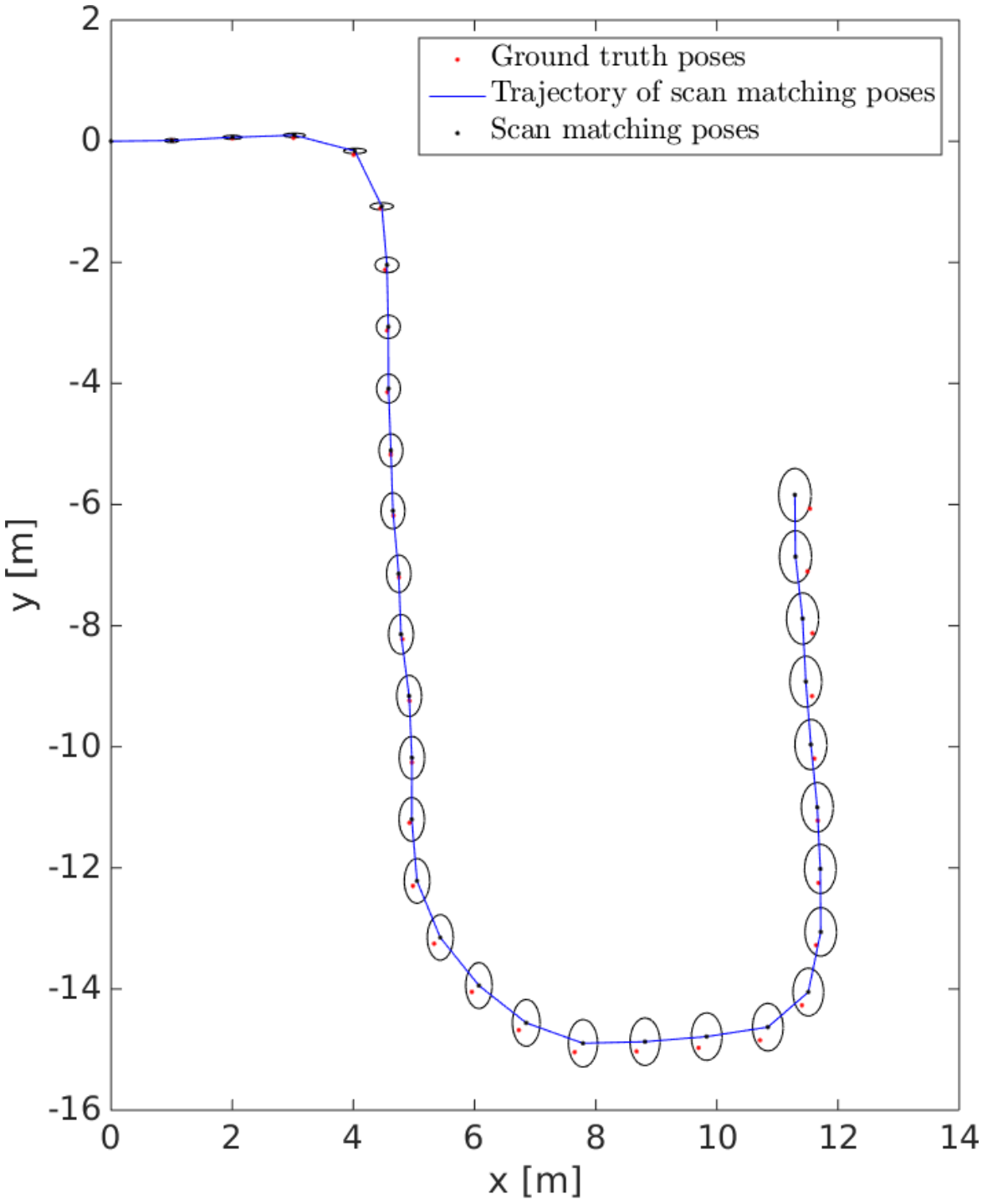


Fig. 4.1 Error propagation from scan matcher

4.2.2 The ellipsoid representation of laser scans

To accelerate the loop closure detection, a new ellipsoid-based approach is presented and implemented. Each laser scan observation is represented by an ellipsoid which could cover all the points in the scan with a minimal volume using only a 2x3 matrix, and the detection of loop closure is converted into the intersection detection of two ellipsoids. The intersection detection is solved by using Lagrange multiplier. The ellipsoid based approach runs very fast, and takes into account the maximal range of the laser scanner which means the robot doesn't have to be physically in the places where it visited before. To explore the inherent property of laser scans, an ellipsoid representation of laser scans is presented here as shown in Fig. 4.2. The calculation of the minimum volume ellipsoid enclosing all scan points is following the method described in [77]. In Fig. 4.2, the tolerance is set to 0.01. There are several benefits from using an ellipsoid representation. Firstly, the representation of an ellipsoid that is only a 2×3 matrix is much smaller than the original scan representation that includes many measured points. Secondly, the ellipsoid covers the region where the scan currently observed. So the intersection check of two nodes is converted to the intersection check of two ellipsoids, which will greatly improve the time needed for rejection of nodes which are not related. Thirdly, the ellipsoid representation enables the search of possible loop closures nodes which are not only the neighbors around the current robot pose, but the nodes which could share common areas, as shown in Fig. 4.3b. The current ellipsoid is expanded by the maximal range of the laser range finder, robot poses inside this expanded ellipsoid will be possible to have loop closure connection with the current node. Nodes which are outside the ellipsoid will have no possibility, and could be safely excluded. In addition to expanded by the maximal range of laser range finder, the ellipsoid will also be expanded by the uncertainty between the two related nodes. The calculation of the uncertainty will be explained in the next section.

Ellipsoid representation An arbitrarily oriented ellipsoid centered at c, is defined by solutions x to the equation

$$(x-c)^T A(x-c) = 1 \tag{4.7}$$

, where A is a positive definite matrix,which contains all the information regarding the shape of the ellipsoid. x, c are vectors.
The eigenvectors of A defines the orientation of the ellipsoid and the eigenvalues of A are the reciprocals of the squares of the semi-axes: a^{-2}, b^{-2}. Those information could be acquired by doing singular value decomposition (SVD).

The expansion and rotation of an ellipsoid follows the equations in the following.

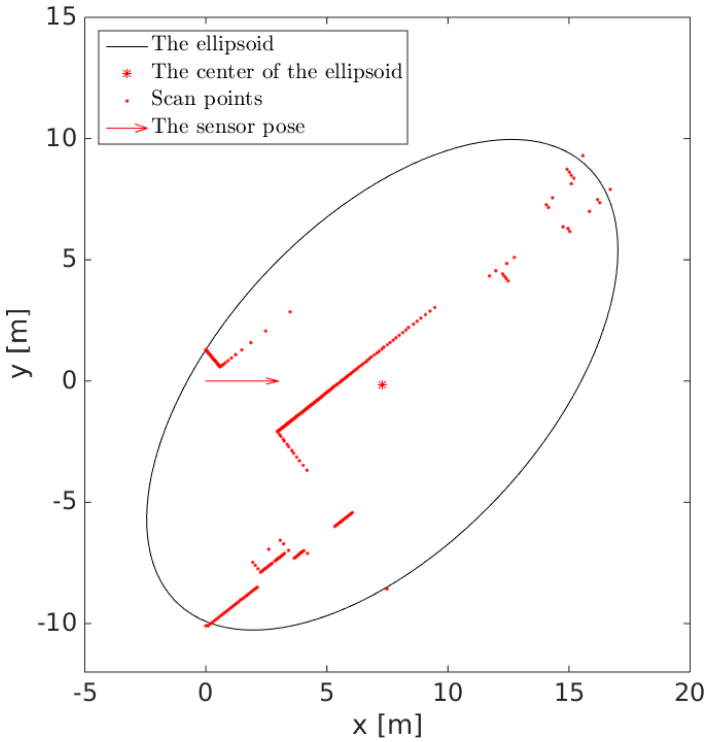


Fig. 4.2 Ellipsoid example

$$[UQV] = svd(A) \tag{4.8}$$

$$a = 1/\sqrt{Q_{00}} \tag{4.9}$$

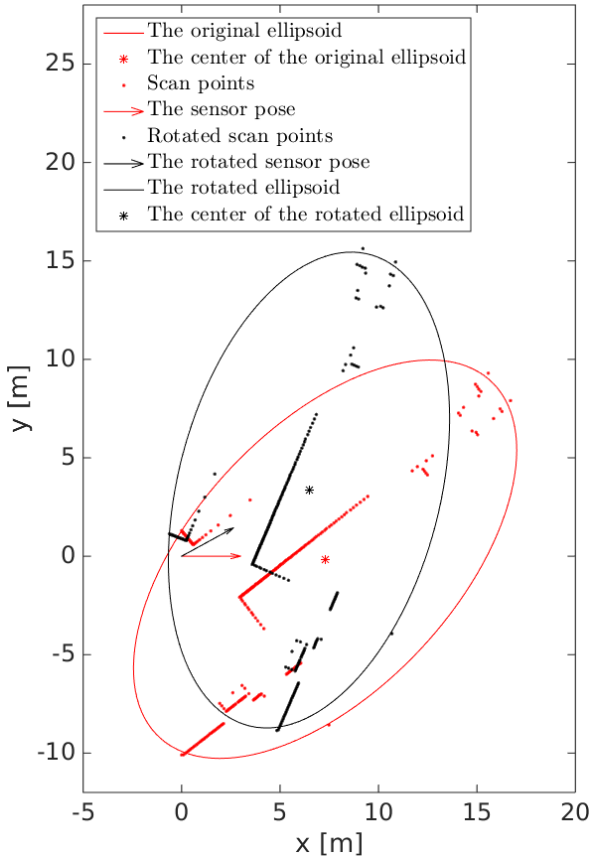
$$b = 1/\sqrt{Q_{11}} \tag{4.10}$$

$$F = \begin{bmatrix} [a^2/(l+a)^2 & 0 \\ 0 & b^2/(l+b)^2] \end{bmatrix} \tag{4.11}$$

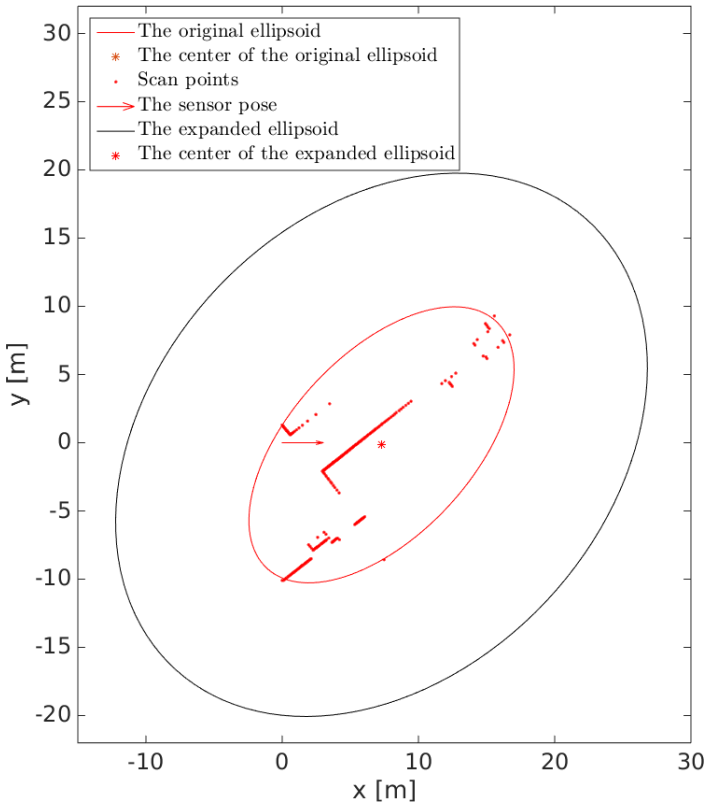
$$c_{expanded} = c \tag{4.12}$$

$$A_{expanded} = U(QF)V \tag{4.13}$$

$$(x - c_{expanded})^T A_{expanded}(x - c_{expanded}) = 1 \tag{4.14}$$



(a) A rotated ellipsoid.



(b) An expanded ellipsoid.

Fig. 4.3 The rotation and expansion of an ellipsoid.

$$R = \begin{bmatrix} cos\theta_r & -sin\theta_r \\ sin\theta_r & cos\theta_r \end{bmatrix} \tag{4.15}$$

$$c_{rotated} = Rc \tag{4.16}$$

$$A_{rotated} = [UQ(RV)^*] = [UQV^*R^*] = AR^* \tag{4.17}$$

$$(x - c_{rotated})^T A_{rotated}(x - c_{rotated}) = 1 \tag{4.18}$$

4.2.3 Extracting segments from a single scan

Extracting segments from laser scans is a common approach in the mobile robotic community, and many researchers have developed segment extraction algorithms. For real-time applications, Split-and-Merge algorithm is the best choice by its superior speed [80]. A heuristic breakpoint based Split-and-Merge algorithm is used in this work for segment feature extraction. An example of the extracted segment features is shown in Fig. 4.4a. The normal of each segment $\vec{n}_i$ is defined to consistently towards the viewpoint, the following equation need to be satisfied

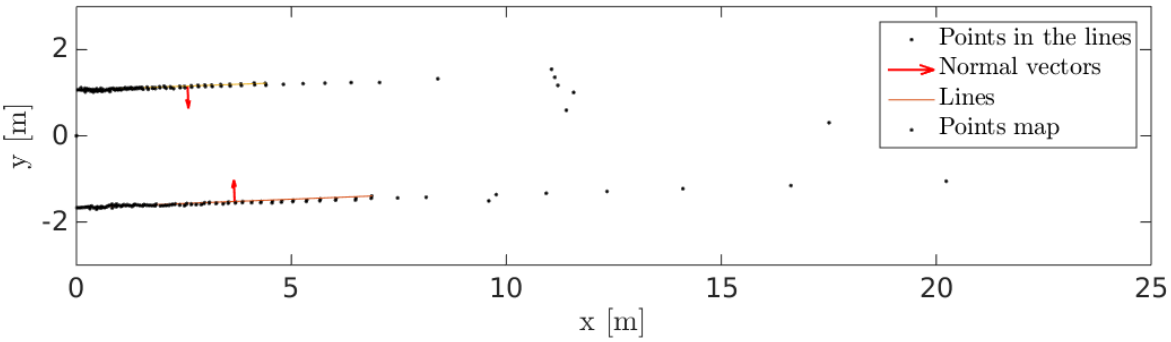
$$\vec{n}_i \cdot (v_p - p_i) > 0, \tag{4.19}$$

where v_p is the viewpoint and p_i is the middle point of the segment.
The breakpoints are determined from local max distances instead of breaking from the middle or from the max distance point.

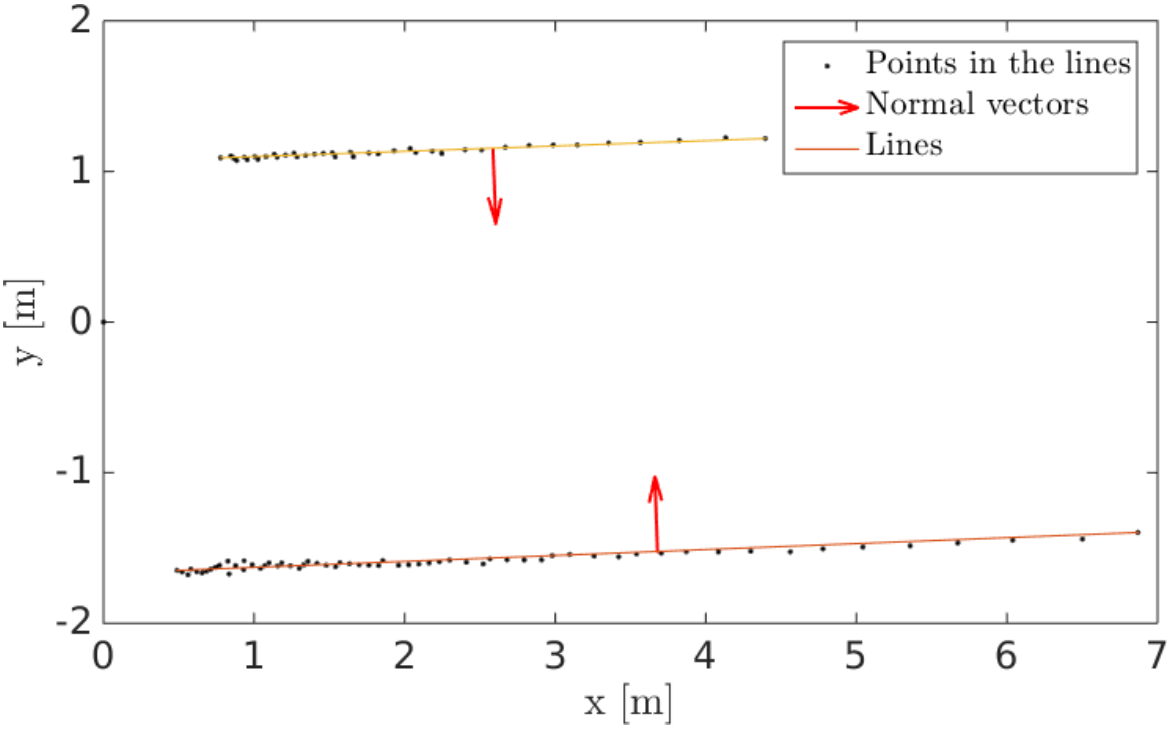
Merging segments from multi scans The merge of segments from multi scans is described in the following algorithm. An example of merging 19 consecutive scan frames is shown in Fig. 4.5, and another example of merging 7 consecutive scan frames is shown in Fig. 4.6.

Algorithm 3 Merge segments from multiple scans

1: $Z = z_0, z_1, \ldots, z_i$ ▷ a set of scans
2: $P = p_0, p_1, \ldots, p_i$ ▷ a set of poses
3: $S = s_0, s_1, \ldots, s_i$ ▷ a set of segments
4: **for all** Z, P **do**
5: Segment detection in sensor frame
6: Transform the detected segments into global frame
7: **end for**
8: **for** $s_i \in S$ **do**
9: Merge s_i with s_0 based on Normal vector and distance between segments
10: Save the merged result in s_0
11: **end for**

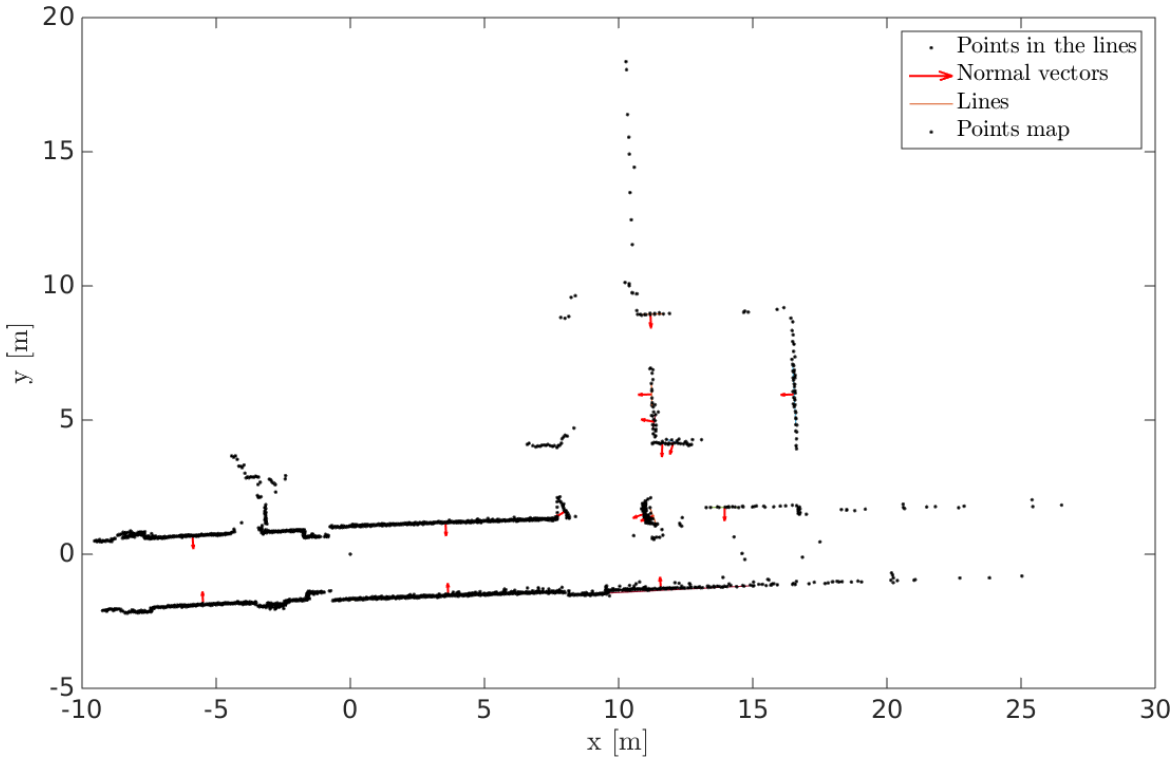


(a) Detected segments in original scan.

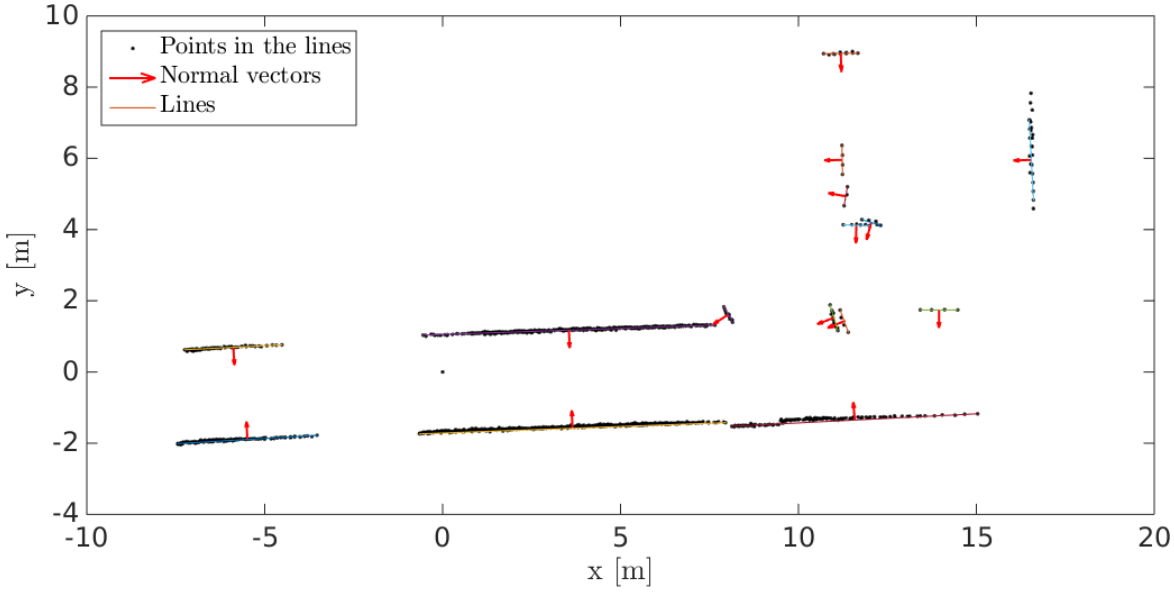


(b) Detected segments together with inlier points.

Fig. 4.4 Extract segment features from a single scan.

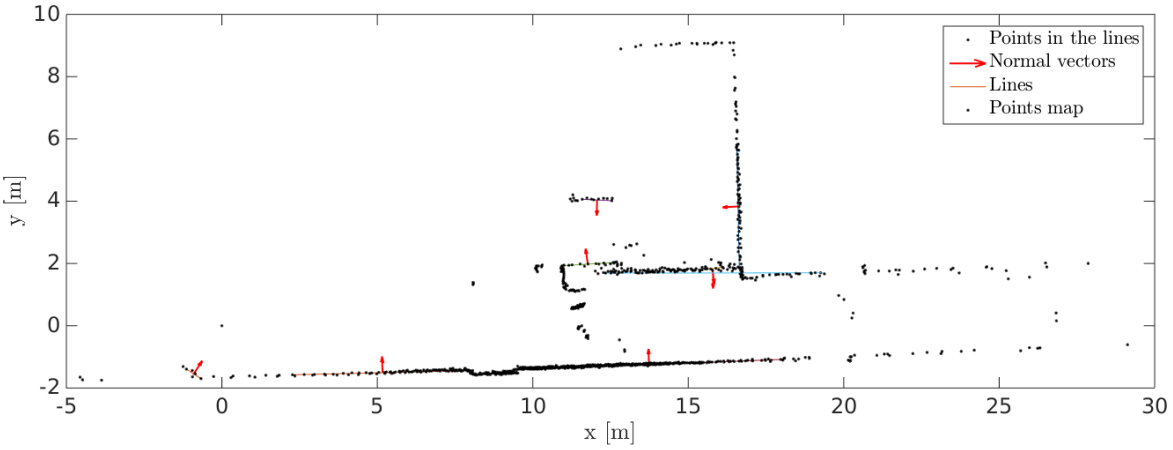


(a) Merged segments from 19 scan frames.

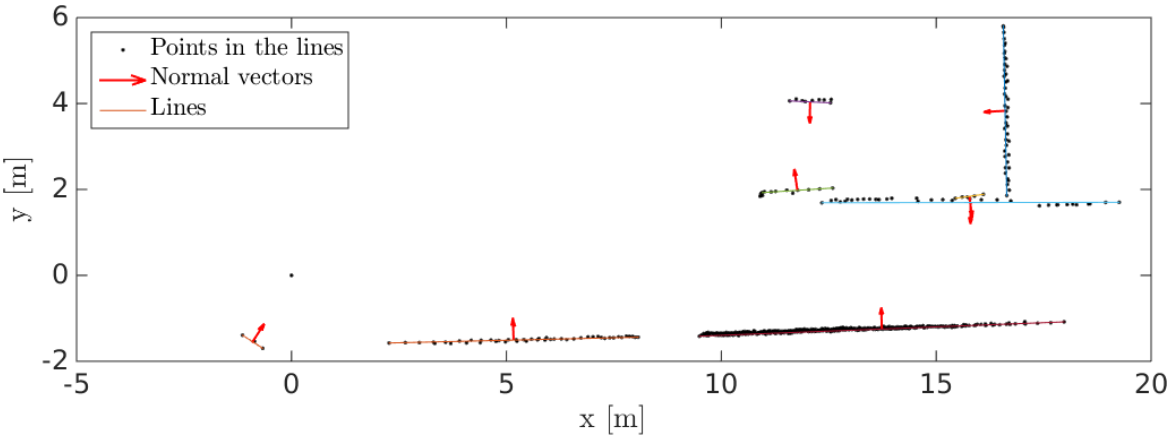


(b) Merged segments together with inlier points from 19 scan frames.

Fig. 4.5 Merge line features from multi scans.



(a) Merged segments from 7 scan frames.



(b) Merged segments together with inlier points from 7 scan frames.

Fig. 4.6 Merge segment features from multi scans.

4.3 System overview

A system overview is shown in Fig. 4.7. The system takes odometry measurements and laser scans as input. Sensor messages are first synchronized before they are used for the graph building. The graph building includes the basic edge building and the loop closure edge building. The graph will be optimized when a loop closure is found. The complete system is optimized for the multi-kernel CPUs and the standby state of the robot is modeled into the system for the further performance optimization. In order to achieve a real-time robust Graph SLAM, the following new methods are researched in the thesis: (1) a novel basic edge building model is proposed, (2) a highly efficient ellipsoid based loop closure edge building model is presented, (3) a multi-threading implementation of the system is achieved. Each of them will be discussed in the following sections.

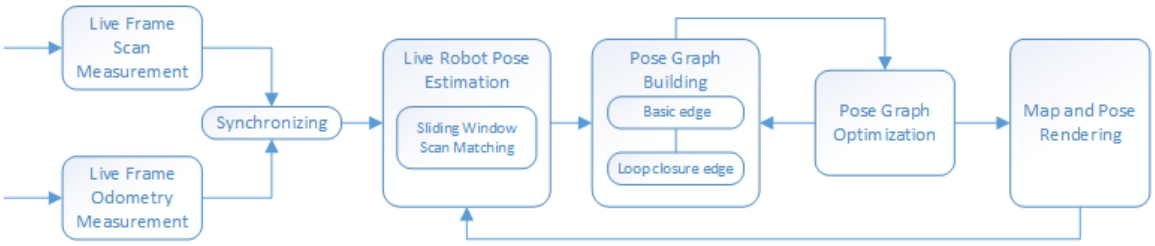


Fig. 4.7 Outline of the Ellipsoid based Graph SLAM pipeline

4.4 Robust basic edge building

4.4.1 Introduction

In laser range finder based Graph SLAM, every node corresponds to a robot pose and to a laser measurement or a submap which is made of multiple consecutive laser measurements. A basic edge represents a data-depend spatial constraint between two successive nodes. When an odometer is available, the odometry measurement could be used for basic edges. However, due to odometry's slippery problem, the odometry measurement tends to drift quickly, mostly scan matching algorithms are used to register laser measurements associated with nodes to correct the odometry measurement. At cases where there are no odometry like sensors, the movement of the robot could be tracked by continually registering every incoming laser measurement with previous laser measurements. In both cases, scan matching algorithms are an important component for the basic edge building. In the literature, various scan matching algorithms have been used. In [41], the brutal force scan matcher CRSM described by Olson [83] is used, a new node is created when the robot moved more than 0.5 meter or 0.5 radian.

A point-to-line based ICP variant (PLICP) from Censi [19] is applied in [47], no detailed information about the threshold for adding a new node is provided. Matching the current scan in the current node against the scan in the previous node is the typical approach as shown in [41][47]. In recent works, matching current scan with local submaps is applied instead of single frame to single frame scan. A moving window based Normal Distributions Transformation (NDT) algorithm is used in [50], every incoming scan is registered against a moving window NDT map. Another similar work, every scan is registered against a local occupancy grid submap using Ceres based scan matcher (CSM) in [45]. Mostly the matching of current with a local map generates better result, as the local map contains more information about the environment.

In this work, a new robust and real-time basic edge building framework will be developed. The sliding window approach is used here, because this approach could keep a short history of the environment, at the same time avoids large error accumulation from using a global map, and the memory usage and the computation time could be kept relatively constant. The new developed approach compared with those previous approaches has two distinctions. Firstly, the author extended CRSM to a sliding window based scan matching method. As in [83], Olson suggests the CRSM could also be extended into matching scan against local maps. Secondly, the principled estimation of uncertainty of the transformation from CRSM rather than over-confident covariances or no covariance from other scan matching algorithms could be used to limit the search area of loop closure candidates. CRSM outputs a covariance estimated from large areas instead of a few sampled volume, which describes the uncertainty of the pose more close to the truth. Several specifications of the graph building, for example, the threshold for adding a new node into the graph and the covariance matrix which could later be used for loop edge building, are also taken into consideration. In this way, the Graph SLAM system will benefit from fast and less false selection of loop closure candidates, which will be presented in the section 4.5. In the following sections, the accuracy, robustness and computation time of the author's approach is compared with other scan matching algorithms under different kinds of environments (complex environment, corridor environment) and different levels of odometry noise to show the performance of the new developed method.

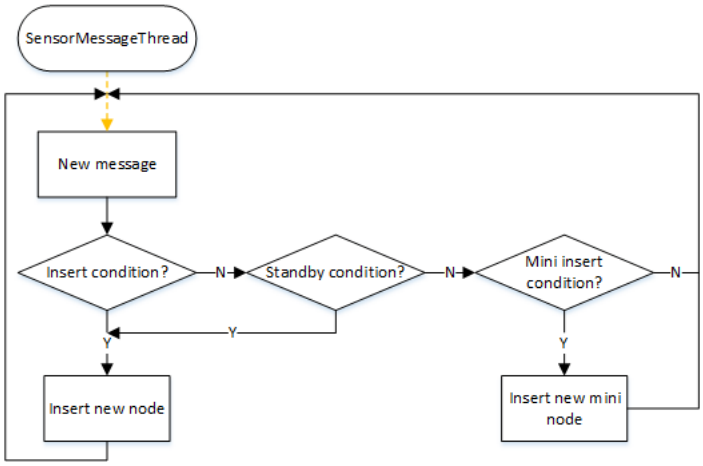
In related works, it is common to use only long term vertexes [41] or use only short term vertexes [27][59][45]. By only using long term vertexes, the state vector of the graph could be smaller, and a lot of memory usage and computation time could be saved, but because the relative large gap between the two neighborhood vertexes, it is possible that the odometry measurement could drift far away from the real movement because of unpredictable slippages. As a result, the motion model could fail to cover the real pose, and the basic edge transformation is not likely to be corrected by scan matcher. By only using short term vertexes, it is

possible to overcome the problem mentioned before, but the cost is also obvious, the density of the graph increases significantly, or the system requires specified sensor, for example high update rate laser scanner is required in [59]. The times of mismatching also increase significantly. Here a new improved method is proposed to take advantage of both the long term method and short term method, at the same time to get rid of the corresponding disadvantages.

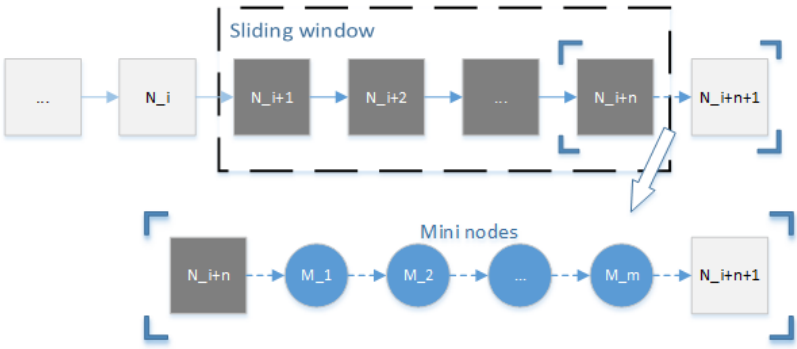
4.4.2 A long-term + short-term sliding window based basic edge builder

The graph $G = (V, E)$ is made of vertexes V and edges E. For every vertex v in V, it contains a pose and a laser scan at this pose. E includes two types of edges, one is the edge connects neighborhood vertexes, the other one is the edge connects far away vertexes (could also be called loop closure edges). For robustly building the basic edges, a long-term plus short term memory based sliding window based scan matcher is used. The flowchart is shown in Fig 4.8a. The key idea here is the vertexes generated from the insert condition and the standby condition are stored for the long term, and the mini vertexes between two long term vertexes are stored only for the short term which will be erased after usage.
The addition of a new vertex into the graph is determined by the translation distance, rotation angle or the time difference. If the odometry measurement of the robot has moved more than the *translation distance* or the *rotation angle* compared with the pose of the last vertex, a new vertex will be added into the graph. Another case is when the robot has stand still for more than the *standby time*, a new vertex will also be added into the graph. The detection of the standby mode and inserting the pose at the standby mode keep the visualization of the robot pose consistent by invoking graph optimization at the standby pose.The addition of new mini vertexes is determined by the parameters which are similar to the addition of a new vertex. If the odometry measurement of the robot is bigger than the *mini translation distance* or *mini rotation angle*, at the same time the time difference is bigger than *time increment*, a new mini pose together with the observed laser scan will be saved. Short term here means those mini vertexes will be cleaned after usage as shown in Algorithm 4. The reason why *time increment* is used instead of odometry measurement is that the physical linear velocity and angular velocity of the robot are limited, so the transformation between *time increment* is also limited. Without the interference from the odometry, the transformation between two neighborhood vertexes could be recovered with the help of those mini vertexes.

A sliding window based CRSM is applied for the basic edge building to make sure the memory consumption of the scan matcher is not growing with the size of the global map, at the same time the accumulation of the error from basic edge building is also limited in

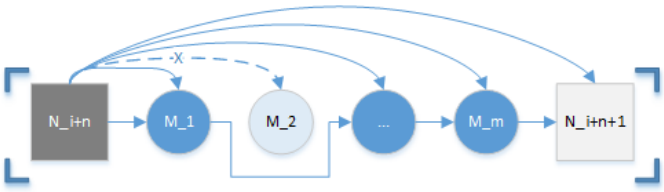


(a) The flowchart of on-line adding long term and short term nodes in the robust basic edge building method.

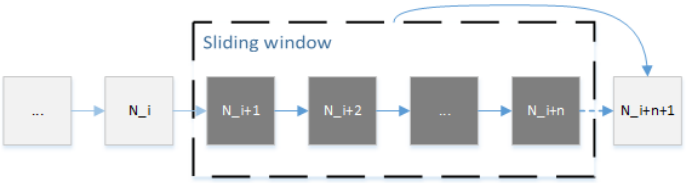


(b) Long term and short term nodes for robust basic edge.

Fig. 4.8 The flowchart of on-line adding long term and short term nodes in the robust basic edge building method.



(a) Matching mini scans for robust basic edge.



(b) Matching scans for robust basic edge.

Fig. 4.9 The flowchart of on-line long term and short term nodes based robust basic edge building method.

the local level. Even the frame to frame matching could achieve 99% robustness, there is still 1% of mismatching. Those mismatched basic edges will distort the final map if there is no enough re-traverse to correct the error. Improving the robustness of basic edge building could narrow the searching area of loop closures edges, and then improving the quality of the final map. The generated long term and short term vertexes are shown in Fig 4.8b, N_i stands for long term vertexes and M_i stands for mini nodes. The flowchart of basic edge building is shown in Fig 4.9. The matching of mini nodes between two long-term nodes is executed first before the matching of the current node to the sliding window. As shown in Fig 4.9a, the matching of mini vertexes is not matching $M_i \rightarrow M_{i+1}$ and then $M_{i+1} \rightarrow M_{i+2}$, instead, all the mini vertexes are matched against last long term vertex N_{i+n}. The long term vertex works like an anchor here. The probability which belows a certain threshold will be rejected, for example, M_2, the input for motion model will be the movement between M_3 and M_1. Thus the uncertainty is always limited to as small as possible.

4.4.3 A new nonlinear distance weight function for the goodness function of the scan matching algorithm

Each individual lidar return z_j is assumed to be independent like previous work, the new score function is written as:

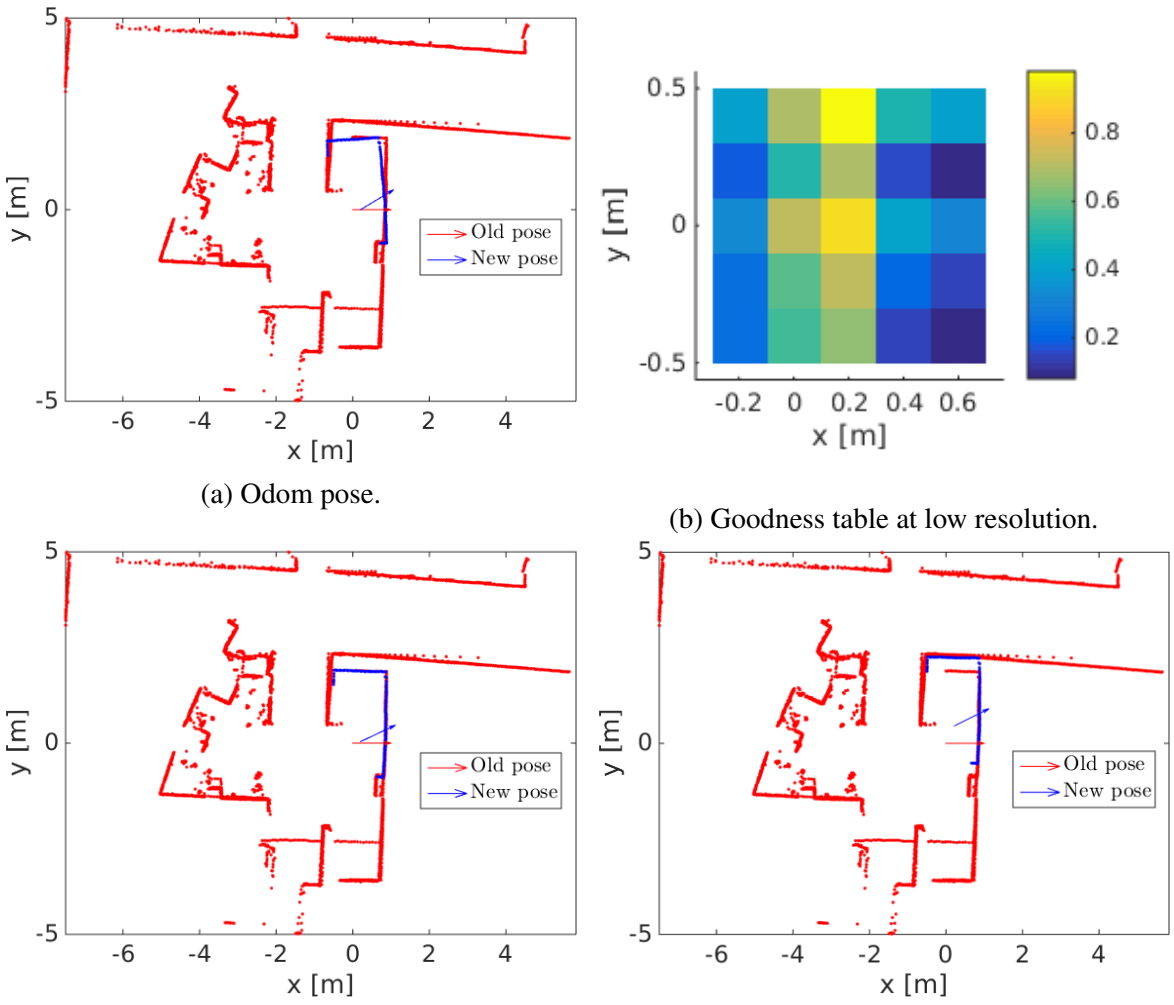
$$p(z|x_i,m) = \prod_j p(z_j|x_i,m) \cdot \frac{\|d_j\|}{d_{max}} \tag{4.20}$$

The probability distribution for the current observation z at pose x_i in map m is $p(z|x_i, m)$. Here the distance of each lidar return d_j is used as weight in the probability distribution for getting better matching result. d_{max} is the maximal range of the laser scanner. Because of the inherent property of the laser scanner, the points at near range is often denser compared with points at far range. By using the distance as weight, the far away points which carry rich information will get higher weight in the matching process, which will minimize the uneven distribution of scan points.

4.4.4 An environment-aware enhancement for the brutal force based scan matching algorithm

As in Olson's paper [83], CRSM does not consider occlusion cases which could fail at cases, for example, robot will consider the two sides of a thin wall as one line. The original CRSM algorithm is solely score-oriented, which means a wrong transformation with the highest score will be accepted as the output. In most cases, the occlusion problem could be solved by limiting the searching area of CRSM, but because of the noise level of different robots vary from each other, for robots with high noise odometry, it is difficult to limit the search area to a small area. Here the author presented an environment-aware extension of the CRSM algorithm to settle the occlusion and corridor problem. The idea is to test the rate of outliers in results from low resolution search. Outliers are points on the current scan which have angle difference bigger than $0.8 * \pi$. The sorted from high score to low score results in low resolution will be filtered by the rate of outliers, if the rate is bigger than 0.05, the result will be removed from the further search in the high resolution check table. An example is shown in Fig. 4.10b. The highest scores are checked with normal vector match with the reference scan. The assumptions with the lowest number of negative points or contours or lines will be accepted as the solution. In this way, the correct solution will survived even it is not the one with highest score. The explicitly sensor viewpoint and corridor aware formulation allows the approach to operate on challenging environments.

Long corridors without features are difficult cases for scan matching algorithms to determine the correct transformation. For those cases, when all are with minimal negative number or no negative number, the one which is closest to the initial guess which is normally the odometry measurement will be accepted as the solution. The idea is to test the distribution of high score assumptions, if the distribution varies, then the assumption which is closest to the initial guess pose will be accepted as solution. An example is shown in Fig. 4.11. The solution which is closest to the initial guess which may not be the correct solution, but because the error from the odometry at short term is limited, it is acceptable to accept the



(a) Odom pose.

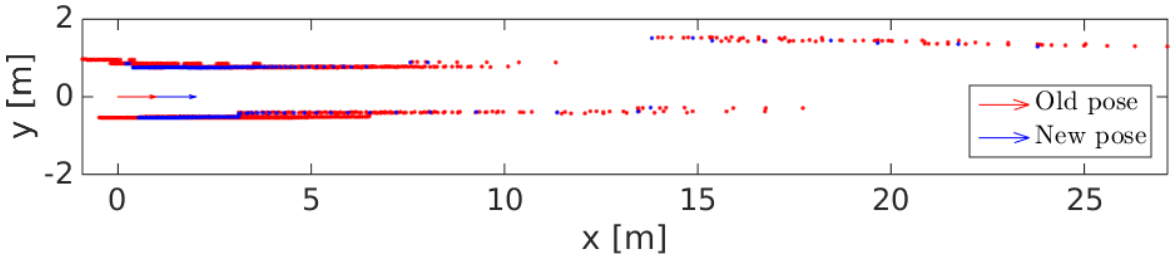
(b) Goodness table at low resolution.

(c) Correct match pose.

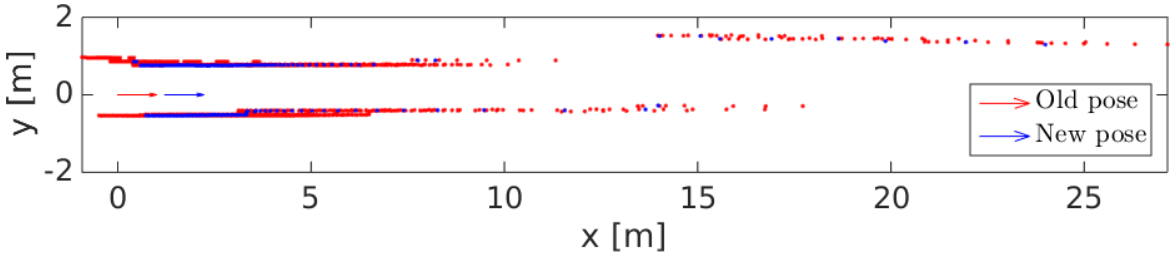
(d) Wrong match pose.

Fig. 4.10 False view side effect. In (a)(c)(d), the red points stand for the old local map, the blue points stand for the current scan. The translational result from low resolution search at the correct angle is shown in (b), where two yellow cells stand for high score. The higher one corresponds to the pose in (b), and the lower one corresponds to the pose in (c). The vanilla CRSM algorithm will come to the result in (d) which has the highest goodness score; with the extension of the new developed outlier check method, the correct result is found in (c).

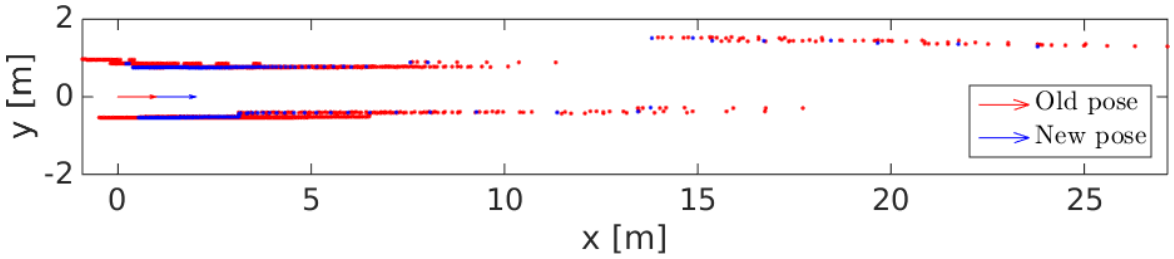
proposed solution, and later when loop closure edges are found, the error from here will be further reduced.



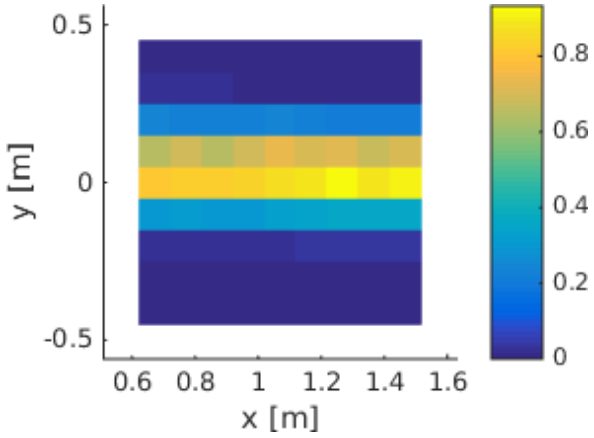
(a) Odom pose.



(b) Wrong match pose.



(c) Correct match pose.



(d) Goodness table at low resolution.

Fig. 4.11 Long corridor effect. In (a)(b)(c), the red points stand for the old local map, the blue points stand for the current scan. The translational result from low resolution search at the correct angle is shown in (d), where an array of yellow cells stand for high score. The highest one corresponds to the pose in (b), and the correct one corresponds to the pose in (c). The vanilla CRSM algorithm will come to the result in (b) which has the highest goodness score; with the extension of the new developed ambiguity check method, the correct result is found in (c).

4.4.5 The filtering of dynamic objects

In real environment, dynamic objects are common, for example, people and other mobile robot platforms. In contrast to previous work with a static environment assumption, here the dynamic objects are detected and removed from the current observation. The detection of moving objects are executed in the mini node level, when a valid scan matching result is acquired, the current mini node will be projected into the occupancy grid map built from last long term vertex and previous accepted mini vertexes. Points in the current mini vertex which projected into free cell of the occupancy grid will be identified as moving objects and removed from the scan.
The detailed algorithm of the robust edge building is described in Algorithm 4.

Algorithm 4 Robust basic edge building algorithm

```
1: procedure BASICEDGEBUILD(nodes)                          ▷ Build basic edges from input nodes
2:     last_node_id ← 1
3:     while system :: ok() do
4:         if current_node_num ≥ last_node_id then                          ▷ Process new nodes
5:             node_done ← False
6:             for i = last_node_id; i ≤ current_node_num; i = i+1 do
7:                 can_be_done ← False
8:                 if current_node_num ≥ i then
9:                     can_be_done ← True
10:                end if
11:                if processed_num == 0 then                    ▷ Reset for new mini node cycle
12:                    T_mini.reset()
13:                    T_mini.update(SCANS[i−1])
14:                    M_dynamic.reset()
15:                    M_dynamic.update(SCANS[i−1])
16:                    mini_id_old ← 0
17:                    last_accum_pose ← (0,0,0)
18:                end if
19:                start_id ← 0
20:                if mini_node_num > processed_num then                ▷ Process new mini nodes
21:                    start_id ← processed_num
22:                else
23:                    break;
24:                end if
25:                result_mini ← 0
26:                for i_mini = start_id; i_mini ≤ current_mininode_num; i_mini = i_mini + 1 do
27:                    init_guess ← MINI_POSES[i_mini] − MINI_POSES[mini_id_old]
28:                    (result_mini, incr_pose) ← match(MINI_SCANS[i_mini], SCANS[i−1], init_guess)
29:                    if result_mini > T_mini then                ▷ Use mini node with good match result
30:                        M_dynamic.update(MINI_SCANS[i_mini])
31:                        T_mini.update(MINI_SCANS[i_mini])
32:                        last_accum_pose ← last_accum_pose + incr_pose
33:                        mini_id_old ← i_mini
34:                    end if
35:                end for
36:                processed_num ← current_mininode_num
37:                if can_be_done then
38:                    init_guess ← odom_pose
39:                    if result_mini > T_node then ▷ Use pose estimated from mini nodes with good match result
40:                        init_guess ← last_accum_pose
41:                    end if
42:                    e ← T_node.match(SCANS[i], init_guess)
43:                    G ← e                                    ▷ Add new basic edge into the graph
44:                    mini_nodes ← empty                                    ▷ Remove mini nodes
45:                    node_done ← True
46:                end if
47:            end for
48:            if node_done then
49:                last_node_id ← i+1
50:            end if
51:        end if
52:    end while
53: end procedure
```

4.4.6 A comparison of persistent-mode vs frame-to-frame scan matching for basic edge building

To prove the robustness of the proposed method, experiments are done in both real environment and simulated environment. The evaluation is carried out on a laptop PC with an Intel Core i7 operating at 2.5GHz with 16GB of RAM. The relative error metric [60] is used here for accuracy benchmarking.

$$\varepsilon(\delta) = \frac{1}{N}\sum_{i,j} trans(\delta_{i,j} \ominus \delta^*_{i,j})^2 + rot(\delta_{i,j} \ominus \delta^*_{i,j})^2, \tag{4.21}$$

where N is the number of relative relations, $\delta^*_{i,j}$ is the estimated relative transformation, $\delta_{i,j}$ is the reference relative transformation, $trans(\cdot)$ is the translation part and $rot(\cdot)$ is the rotation part.

Simulated environment

The proposed method is compared with PLICP [19] to demonstrate the robustness. The size of the simulated environment is 54 m x 58.7 m. A laser scanner with max range of 20 m and FOV 180° at resolution 0.25°. The scan matching will happen between two consecutive nodes, the generating of new nodes follows the principle described above in Figure 4.9. The simulated environment is also shown in the Fig. 4.12. In total there are 657 nodes. The number of error above 0.1 m or rad is 10 times. 4 out of 10 scan matchings are shown in the Fig. 4.13. The detailed accuracy result is shown in Table 4.1. The reason why the proposed method is superior to PLICP is that the Weighted CRSM (WCRSM) could use the scan frames from before which is determined by its lookup table accumulation principle.

Table 4.1 WCRSM compared with PLICP for basic edge building

Method Precision (m,rad)	WCRSM %	PLICP %
<0.05	100	97.41
(0.05,0.1)	0	1.07
>0.1	0	1.52

Real environment

The new proposed robust basic edge building algorithm involves three parameters which are mini node threshold MNT, the number of nodes in the sliding window NUM_{win} and

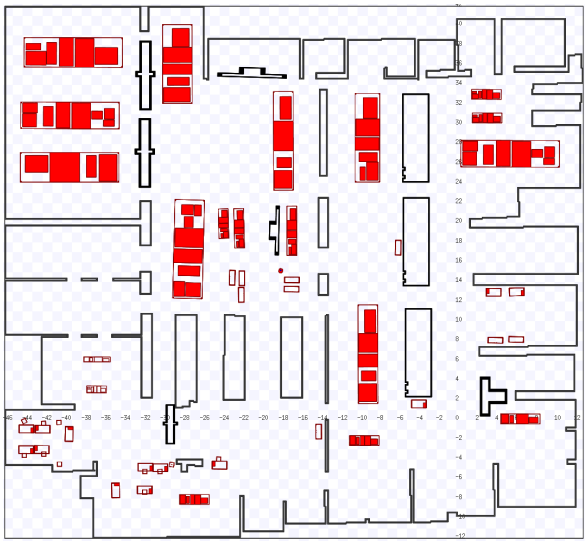


Fig. 4.12 The simulated warehouse.

the boolean parameter USE_{dw}. To determine the optimal values for the parameters, two experiments are designed. The first experiment, the parameter setting is shown in Table 4.2. The test data information is shown in another Table 4.3. For both datasets, the maps and the histogram of relations between nodes is illustrated in Fig. 4.14.

Table 4.2 Parameter setting of experiment 1

Parameter names	Parameter values
MNT (m or rad)	0 0.05 0.1
NUM_{win}	0 10 20 30 40 50
USE_{dw}	true false

Table 4.3 Datasets used in experiment 1

Dataset	Intel Research Lab	ACES Building
Number of scans used	13462	7238
Number of nodes	910	440
Environment size	28 m x 28 m	54 m x 57 m
Dataset duration	2691 s	1365 s
Time per node	2.957 s	3.102 s
Scan Filed of view	180°	180°
Scan angle resolution	1°	1°
Scan Max range	50 m	50 m

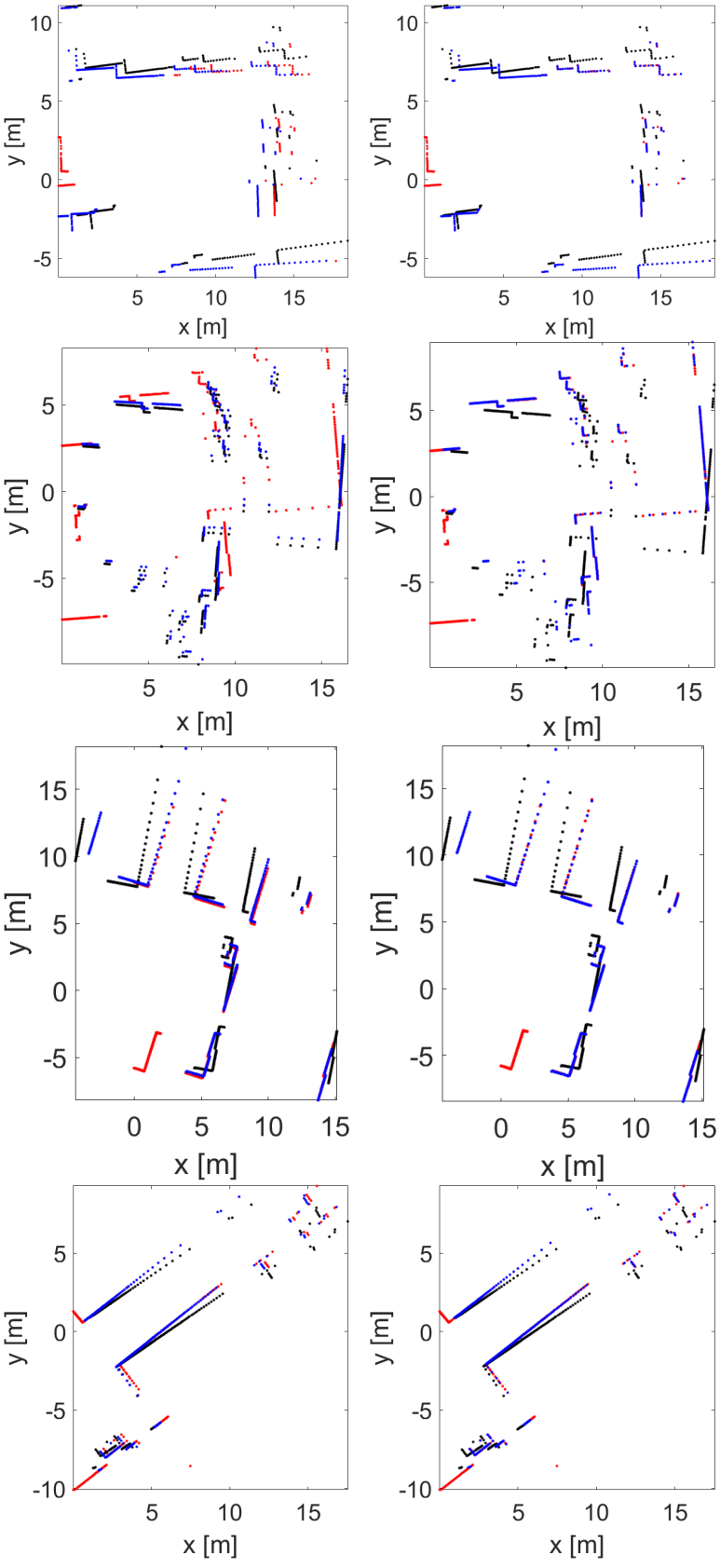
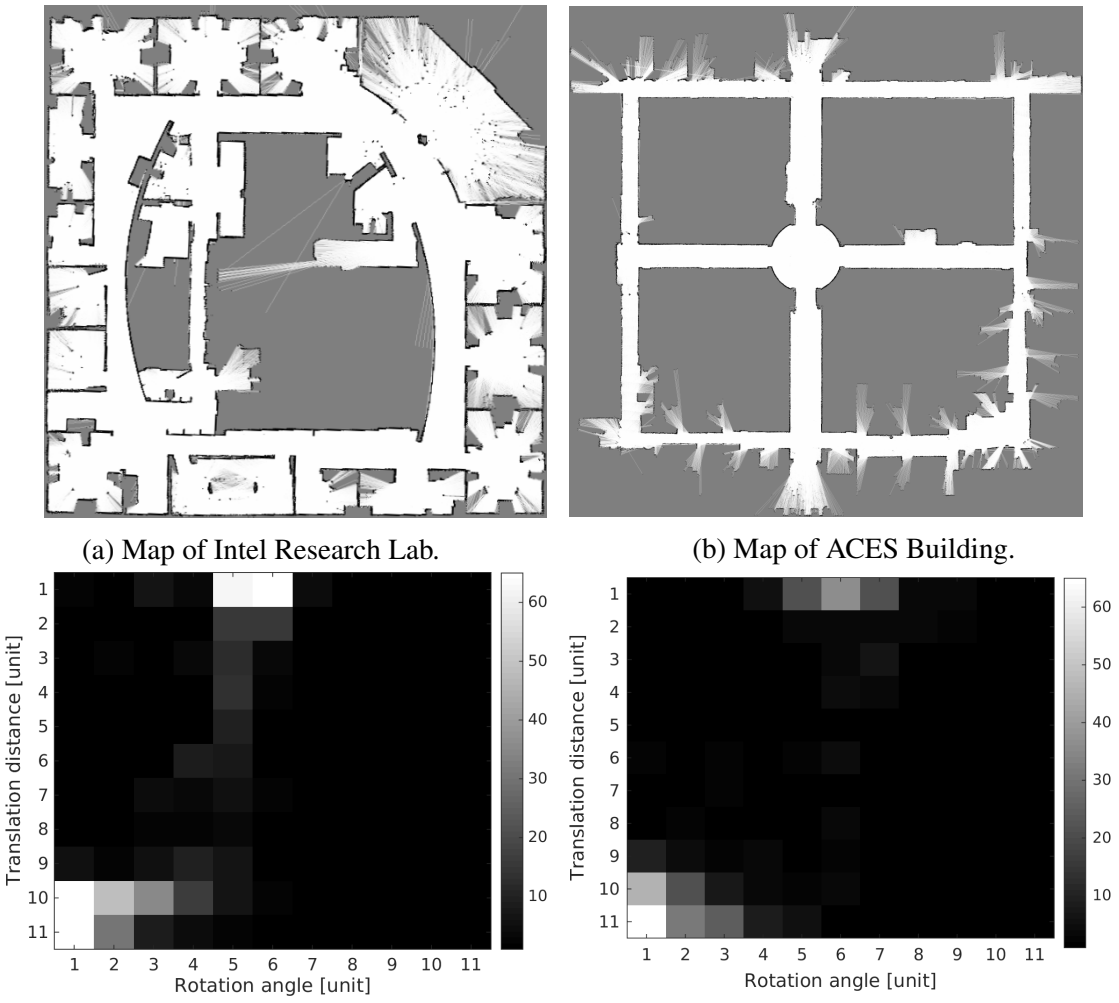


Fig. 4.13 Comparison for basic edge building. The left column is PLICP, the right column is the author's new method. Red · is the first scan, black · is the second scan, blue · is the transformed scan.



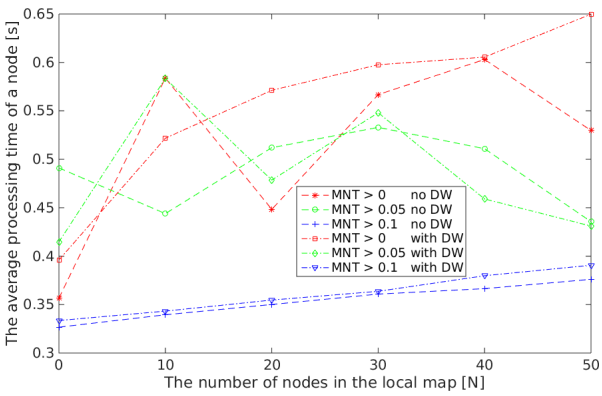
(a) Map of Intel Research Lab.

(b) Map of ACES Building.

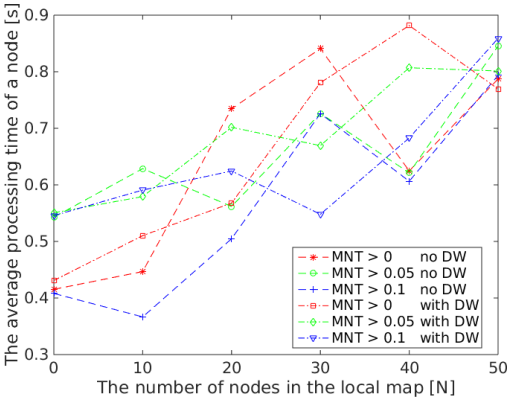
(c) Normalized histogram of Intel nodes.

(d) Normalized histogram of ACES nodes.

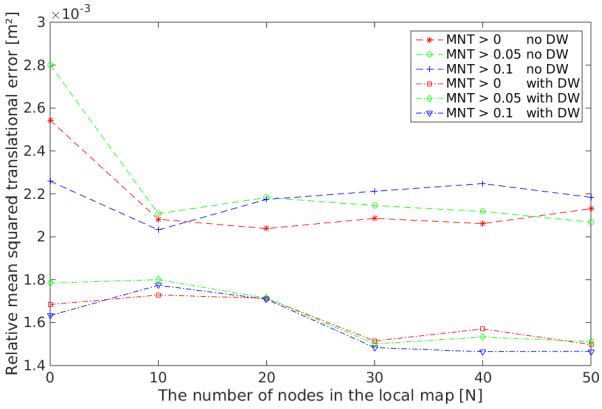
Fig. 4.14 Histogram of relations between nodes in Intel Research Lab and ACES Building. The left column is Intel research Lab dataset, the right column is ACES Building dataset. The unit resolution in the histogram is 0.1 meter for translation, and 0.1 radian for rotation.



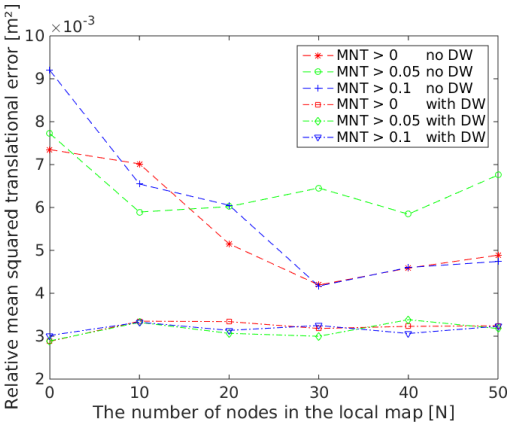
(a) Intel average processing time for each node.



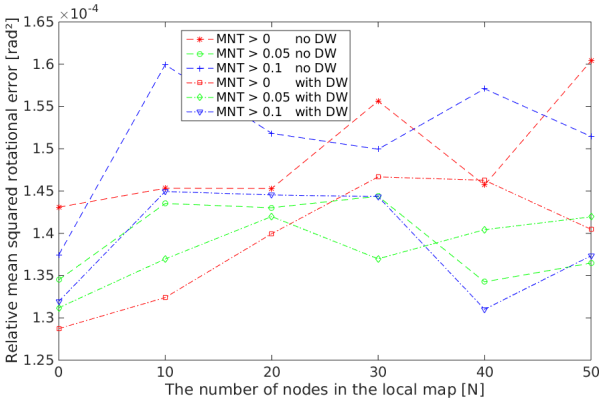
(b) ACES average processing time for each node.



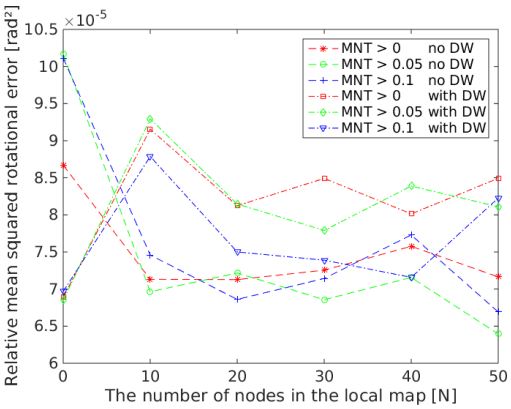
(c) Intel relative squared translational error.



(d) ACES relative squared translational error.



(e) Intel relative squared rotational error.

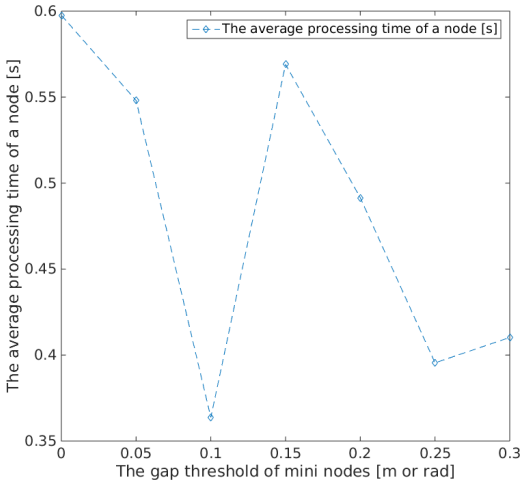


(f) ACES relative squared rotational error.

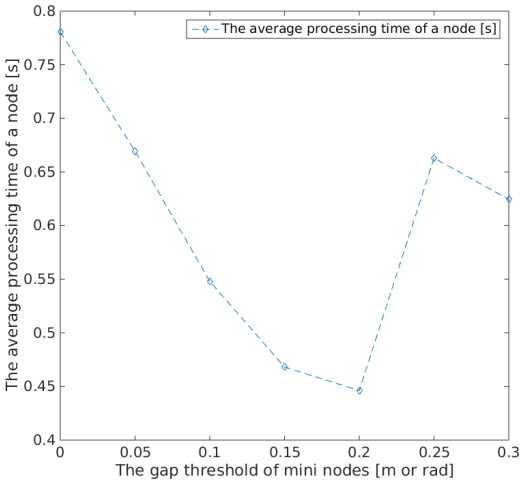
Fig. 4.15 Comparison for basic edge building. The left column is Intel research Lab dataset, the right column is ACES dataset.

Results

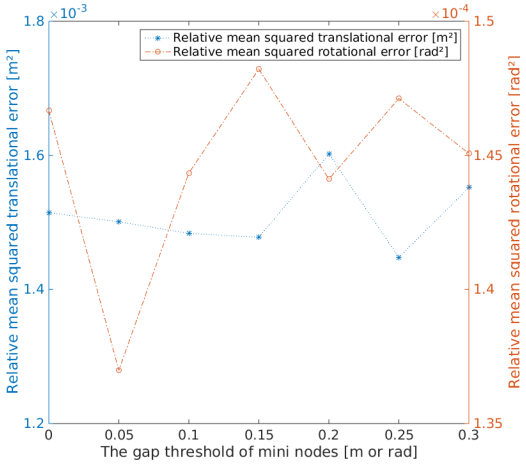
In Fig. 4.15, the translational error is reduced after using distance weight in all three MNT cases for both datasets. For the rotational error part, in the Intel dataset, the error is reduced by using distance weight in all three MNT cases; in the ACES dataset, the error is not reduced by using distance weight, but the result between using distance weight and not using distance weight is close, both are very accurate (errors are smaller than $10^{-4} rad^2$). The processing time increases only slightly by using distance weight. So one can safely use the distance weight to achieve higher accuracy without the worry of significant extra computation time.
For the choosing of parameter NUM_{win}, the translation error is relatively constant when $NUM_{win} \geq 30$ for all six cases for both datasets. For the rotational error part, the break point for NUM_{win} is not as obvious as for the translation part, the errors are small and close. The processing time generally increases with the increasing of NUM_{win}, but the accuracy doesn't increases consequently, because too old scans in the sliding window could not be observed by the current scan anymore. When there are too many old scans in the windows, the result may get worse because of too many accumulated error from the past. NUM_{win} depends on a variety of conditions, for example the maximal sensing range of the sensor, the size of the operating environment and even the way of robot driving. In this work, 30 is empirically used for NUM_{win} when MNT is set to 0.1.
Experiment 2 is designed to further investigate the optimal value for parameter MNT after NUM_{win} and USE_{dw} are fixed. The test value of MNT is set to $[0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3]$, and $NUM_{win} = 30$ $USE_{dw} = true$. The result is shown in Fig 4.16. There is no unified breakpoints for two datasets under three criteria. In order to get a unified result, a cost function for each criterion is adopted, for example, in Figure 4.16a, cost value is assigned from 0 to 6 according to the value of time from small to big. For translational error and rotational error, cost value is assigned from 0 to 6 according to the value of error from small to big. The sum cost is computed by adding the cost value at all six criteria at corresponding MNT value. The result is shown in Figure 4.17. A clear breakpoint is shown at 0.1, which indicates that the optimal solution is not process every incoming observation, and also not too wide (more than 0.1). The average processing time for a new node in Intel dataset is 0.364s which is 8 times faster than the real observation time. The average processing time for a new node in ACES dataset is 0.548s which is 5.7 times faster than the real observation time. The real average time between nodes is shown in Table 4.2. So the proposed basic edge building algorithm is able to satisfy the real-time requirement.
In this work, the addition of a new vertex into the graph is determined by parameters in Table 4.4, and the addition of a new mini vertex into the graph is determined by parameters in Table 4.5.



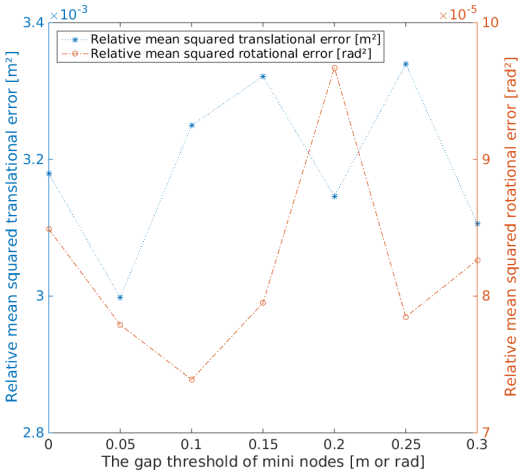
(a) Intel average processing time for each node.



(b) ACES average processing time for each node.



(c) Intel relative squared translational error.



(d) ACES relative squared translational error.

Fig. 4.16 Comparison for basic edge building. The left column is Intel research Lab dataset, the right column is ACES building dataset.

Table 4.4 Parameters for adding new vertex

Parameter name	Value
Translation distance (meter)	1
Rotation angle (degree)	45
Standby time (second)	5

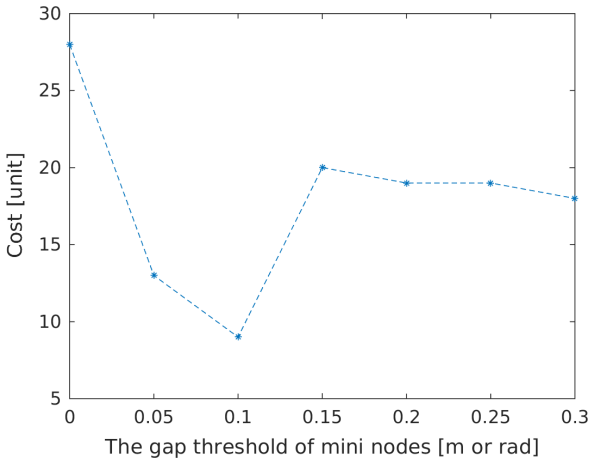


Fig. 4.17 The cost for different MNT values.

Table 4.5 Parameters for adding new mini vertex

Parameter name	Value
Mini translation distance (meter)	0.01
Mini rotation angle (degree)	0.01
Time increment (second)	0.1

4.5 Efficient ellipsoid based loop closure edge building

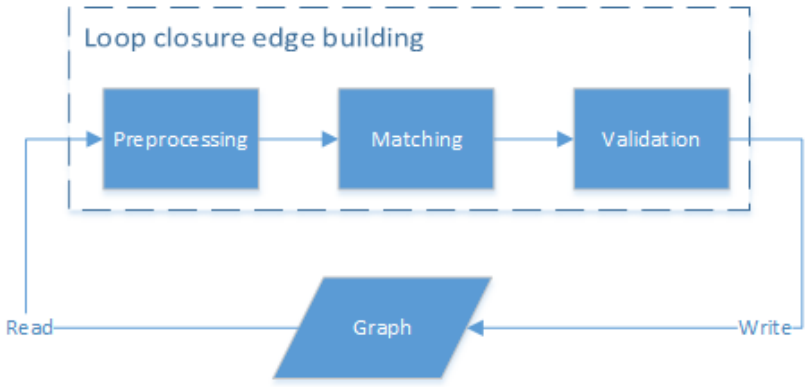


Fig. 4.18 Ellipsoid based loop closure detection

In order to get real-time performance in the graph based SLAM algorithm, several optimizations are needed, here a new ellipsoid based loop closure detection method is proposed which could achieve both real-time performance and robustness. The flow chart of the ellipsoid based loop closure detection method is shown in Fig. 4.18. There are in total three steps,

which are *Preprocessing*, *Matching* and *Validation*. Each step includes a few small steps. Details will be explained in the following section.

4.5.1 Preprocessing

Normally the searching of loop closures is greedy and computational expensive. If every node is checked with the other nodes, the total check time will be $\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$, where n is the number of nodes in the graph. With the increasing number of nodes in the graph, it will be difficult for the algorithm to run in real-time. So it is important to do a preprocessing process that will reduce the number of unrelated nodes in the graph for finding loop closures. In related works, most of the preprocessing methods are only based on the relationship between poses, for example, only nodes whose 3σ contains the current robot pose are considered, where σ is the marginal covariance [41]. A spherical Region of Interest (ROI) centered at the current pose is used in [71]. By limiting the search area the computation time could be reduced. The closest work related with the author's work is that the laser scan is modeled as a circle with c_a as the centroid of end points of laser beams, with r_a as the average distance of laser beams to the centroid in [82]. The overlapping check is converted into the detection of circle intersection. The state-of-the-art methods consider only the physically close poses without taking the laser scans coupled with the poses into consideration, or use a simplified representation of the laser scan to find physically far away nodes. By taking advantage of the coupled laser scan, a better connection between nodes that are not physically close but have shared observation areas using the full information of the scan could be established. To fulfill the requirement of real-time performance and precision at the same time, every scan is represented by an ellipsoid which could cover all the points with a minimal volume [77]. The first two steps in the preprocessing step are only ellipsoid based operations which could run extremely fast, and unrelated nodes could be quickly filtered out. The third step will further check the number of points from two scans in the intersected area, if the number is above certain threshold, the size of the shared observation area will be checked, if the size of the shared area is above certain threshold, the node will be stored into a candidate queue for finding loop closures. The queue will be sorted based on the size of the shared area, which means the node which has a bigger shared area size with the current node will be check earlier. Let's define all the nodes in the graph as N, node with id i is defined as $n_i (n_i \in N, i = 0, \ldots, cnt)$, where cnt is the current node id.

Step 1: select nodes whose poses are inside the expanded ellipsoid representation of the current observation

The current ellipsoid A_t will be expanded by the maximal range of the laser scanner *max_r* and the covariance $\sum = (\sigma_x, \sigma_y, \sigma_\theta)^T$ of the current node n_t using Dijkstra Projection [82]. Factor $f_1 \in (0,1]$ is used to scaling *max_r*. $\sigma_{max} = max(\sigma_x, \sigma_y)$. Factor f_2 is used to scaling σ_{max}. Those selected nodes are defined as $N_1, N_1 \subset N$. The orientation uncertainty σ_θ is used to sample two new ellipsoids when $\sigma_\theta > \theta_{threshold}$ ($\theta_{threshold} = 1$ is used here), otherwise σ_θ will be ignored. The expansion length $l = (f_1 * max\_r + f_2 * \sigma_{max})$, and the rotation angle $\theta_r = \pm\sigma_\theta$. The expanded ellipsoid is represented as $\hat{A}_t$.

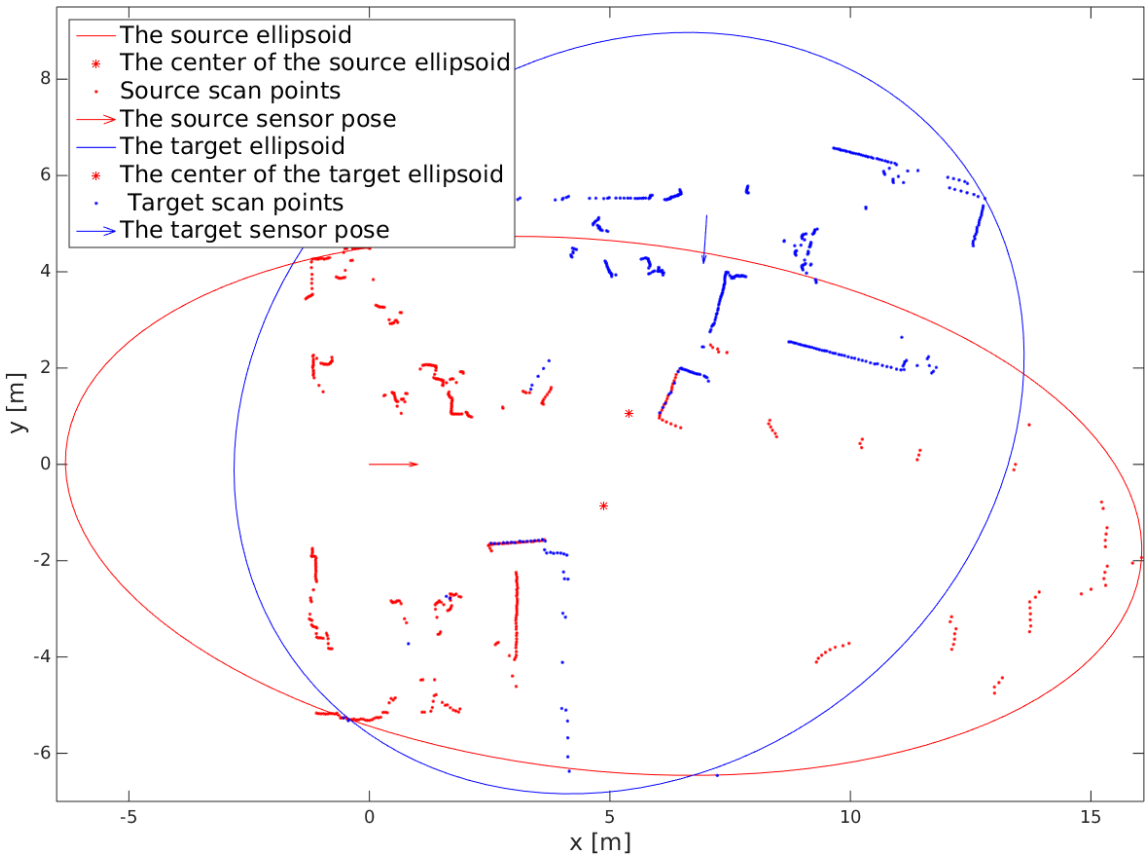
$$N_1 = \{n_i | n_i \in \hat{A}_t\} \tag{4.22}$$

Step 2: select nodes whose ellipsoids intersect with the current ellipsoid

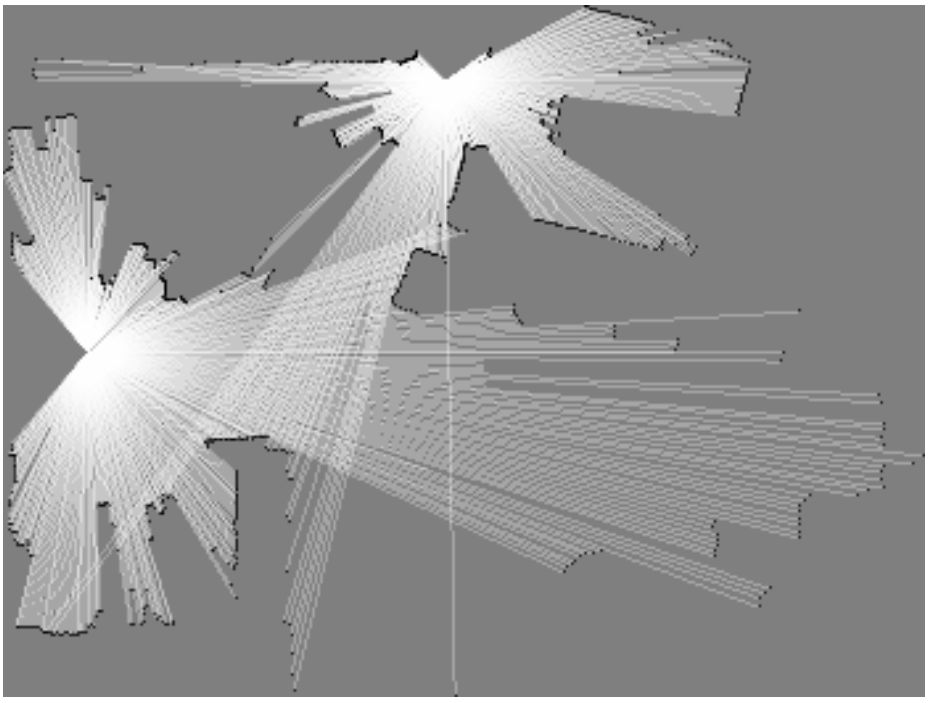
The selection in Step 2 will based on N_1. The covariance $\sum_{rt} = (\sigma_x, \sigma_y, \sigma_\theta)^T$ is the relative uncertainty between n_r ($n_r \in N_1$) and n_t. The current ellipsoid A_t will be expanded by the $\sigma_{max} = max(\sigma_x, \sigma_y)$. The orientation uncertainty σ_θ is used to sample two new ellipsoids. The intersection check is formed as solving a maximal 4th degree polynomial equation using Lagrange multiplier. The equation from Lagrange multiplier is $\mathscr{L}(x,y,\lambda) = f(x,y) - \lambda \cdot g(x,y)$. The equation could have 4 real results, 2 real results or no real results. The check should be two-directions to avoid getting false result when one ellipsoid includes the other one. The orientation uncertainty σ_θ is used to sample two new ellipsoids when $\sigma_\theta > \theta_{threshold}$ ($\theta_{threshold} = 1$ is used here), otherwise σ_θ will be ignored. The expansion length $l = f * cov_{max}$, and the rotation angle $\theta_r = \pm\sigma_\theta$. If two ellipsoids intersect with each other, n_r will be added into N_2. The selected nodes are defined as $N_2, N_2 \subset N_1$.

Step 3: select nodes that have many points in the intersection area

The selection in Step 3 is based on N_2. In Step 2, only ellipsoids which intersect with the current ellipsoid are considered. Even two ellipsoids intersect with each other, they may have no points in the intersection area. In this step, the number of points from two scans which are inside the intersection area is calculated. If the number is above certain threshold, the node will be checked further in step 4, otherwise it will be rejected. The selected nodes are defined as $N_3, N_3 \subset N_2$.



(a) Check whether ellipsoids intersect with each other



(b) Overlaid occupancy grid map using ray tracing

Fig. 4.19 Check whether ellipsoids intersect with each other, the size of the shared free area in this case is 7.38 m^2.

Step 4: select nodes that have large shared free area

The selection in Step 4 is based on N_3. It is possible that, even two ellipsoid intersect with each other and have enough points in the intersection area, they don't really have shared region. In this step, both scans will be projected into the same coordinate and generate an occupancy grid map using ray tracing, the number of cells which are free in both occupancy grid maps are calculated, nodes which pass the threshold *min_area_size* will be selected. $\sum_{rt}$ is used in the same way as in Step 2. Those selected nodes will be defined as $N_4, N_4 \subset N_3$. The calculation of the intersection area has two cases, one is small gap, the other is large gap. If the uncertainty between the current node and the target node is more than a certain threshold, a gradient descent algorithm with the relative pose as the initial pose is used to search the space limited by the relative uncertainty to maximize the shared area size. In this way, the criteria of shared free area size is consistent as a measuring of the probability of nodes to have loop closure with the current node, even nodes which have initially small shared area size will be considered. A graphical illustration of the ellipsoid based intersection area check is shown in Fig. 4.19. In the end of preprocessing, the node set $N_4, N_4 \subset N_3$ will be checked for loop closure. Only a maximal number of candidates will be searched. Under the limitation of error propagation, the establish of obvious false loop closure hypothesis is prevented. Because of the robust basic edge building algorithm , the rotational drift in the preprocessing phase is normally small and could be ignored.

Algorithm 5 Preprocessing for searching loop closure candidates

```
1: targetID = currentID
2: A = computeEllipsoid(targetID)
3: checkID = targetID − ignoreNum
4: eExtended = extendEllipsoid(A, maxRange, cov)
5: while checkID > 0 do
6:     checkID − −
7:     if !checkPoseInEllipsoid(checkID, eExtended) then
8:         continue
9:     end if
10:    if !checkEllipsoidIntersect(checkID, targetID) then
11:        continue
12:    end if
13:    if !checkEllipsoidIntersectPointsRate(checkID, targetID) then
14:        continue
15:    end if
16:    if checkEllipsoidIntersectSize(checkID, targetID) then
17:        candidateList.push_back(checkID)
18:        checkID − −                                   ▷ skip the next node
19:    end if
20: end while
```

Comparison of different preprocessing methods

The ellipsoid based preprocessing method is compared with 3-sigma method and 3-sigma-circle method. Three public available datasets which are Intel research lab, ACES building and MIT Killian Court, are used for benchmarking the performance. The related information of those datasets are shown in Table. 4.6. A lap with a Intel CPU running at 2.4 GHz is used for the experiment. Basic edges are generated by using the method from the previous section, every 0.5 meter or 0.5 radian, a new node will be added into the graph, the pose and the corresponding covariance for each node are calculated. The preprocessing procedure will be based on the basic edges. For every node, a maximal number of candidates will be selected, in the experiment, 100 and 30 are selected, then those candidates will be validated by analyzing the shared area and the complexity of the shared area using the ground truth pose and corresponding scans. If the share area is large and complex, the candidate will be accepted as a validate candidate. The total execution time, the found number, the valid number and the analysis of the distribution of validate candidates are compared. The result for maximal 100 candidates per node is shown in Table 4.7. The result for maximal 30 candidates per node is shown in Table 4.8. The result shows that 3-sigma-circle method could find the largest candidate number, but their method is not good at discriminate the valid ones from invalid ones, as this method produces smaller valid candidates than the author's ellipsoid based method. When the maximal candidate number decreased from 100 to 30, the number of validate candidates drops significantly. The 3-sigma method is the fastest method in both experiments, but the limitation of 3-sigma method is also obvious, the candidates are all in 3 sigma, and a lot of far away validate candidates cannot be found. The author's ellipsoid based method is the slowest method, but compared with the sum time of test datasets, the preprocessing time is negligible. In both experiments, the author's ellipsoid method produces the highest number of valid candidates, and the author's method could find candidate nodes which are far away from the current node.

Table 4.6 Datasets used in preprocessing experiment

Dataset	Intel Research Lab	MIT Killian Court	ACES Building
Number of scans used	13462	17469	7238
Number of nodes	910	1941	440
Environment size	28 m x 28 m	230 m x 185 m	54 m x 57 m
Dataset duration (s)	2691	7677	1365
Scan Filed of view (°)	180	180	180
Scan angle resolution (°)	1	1	1
Scan Max range (m)	50	50	50

Table 4.7 Comparison of preprocessing methods based on scan matching pose and covariance (max 100 candidates per node)

Data	Method	Time used(s)	Found number	valid number	near	middle	far
Intel	Ellipsoid	27.167	24985	5860	1201	1647	3012
	3-sigma-circle	2.765	73023	3748	847	1092	1809
	3-sigma	0.015	2630	1399	1399	0	0
ACES	Ellipsoid	7.798	4011	666	185	138	343
	3-sigma-circle	0.731	29896	446	157	111	178
	3-sigma	0.004	365	207	207	0	0
MIT Killian	Ellipsoid	11.776	7112	1339	509	477	353
	3-sigma-circle	8.408	180285	493	165	148	180
	3-sigma	0.064	2942	719	719	0	0

Table 4.8 Comparison of preprocessing methods based on scan matching pose and covariance (max 30 candidates per node)

Data	Method	Time used (s)	Found number	valid number	near	middle	far
Intel	Ellipsoid	27.729	16194	5066	1086	1462	2518
	3-sigma-circle	1.249	24583	1078	145	197	736
	3-sigma	0.015	2630	1399	1399	0	0
ACES	Ellipsoid	7.842	3719	660	185	138	337
	3-sigma-circle	0.525	11346	130	40	26	64
	3-sigma	0.004	365	207	207	0	0
MIT Killian	Ellipsoid	12.769	7082	1334	509	472	353
	3-sigma-circle	3.077	57505	77	11	18	48
	3-sigma	0.064	2942	719	719	0	0

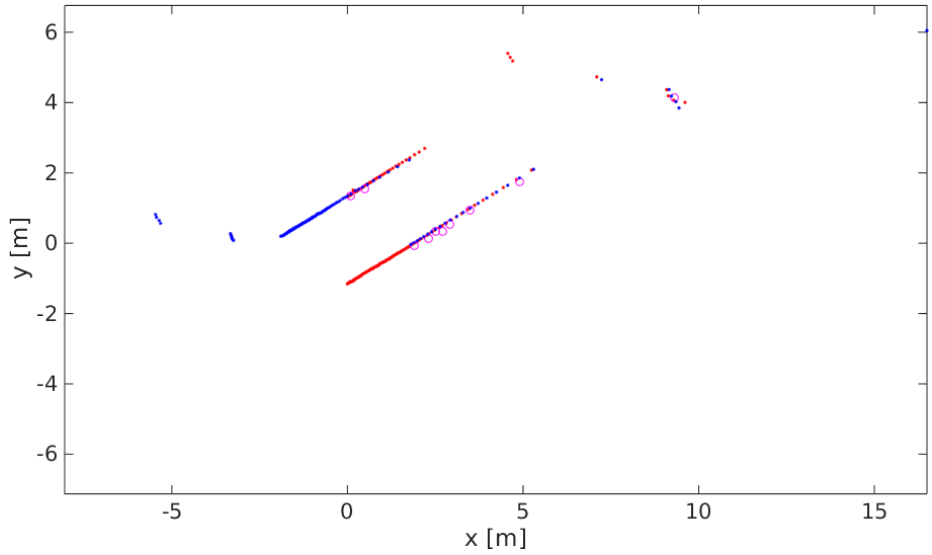
4.5.2 Loop closure detection

Different from the basic edge building (incremental scan matching), enough overlapping area is not guaranteed for the generation of loop closure hypothesis, and the initial guess pose could be far away from the real pose. This makes the generation of loop closure hypothesis a difficult case for normal scan matching methods. Because, for example, the famous vanilla ICP algorithm assumes 100% overlapping. Under limited overlapping area, the ambiguity of the environment is another difficulty for generating loop closure hypothesis. Further a large uncertainty in the initial guess will result in plenty searching time. To achieve real time performance and find correct transformations of loop closure edges, the large uncertainty and limited overlapping conditions should be overcame. New techniques for dealing with those problems mentioned above are presented in the following paragraphs.

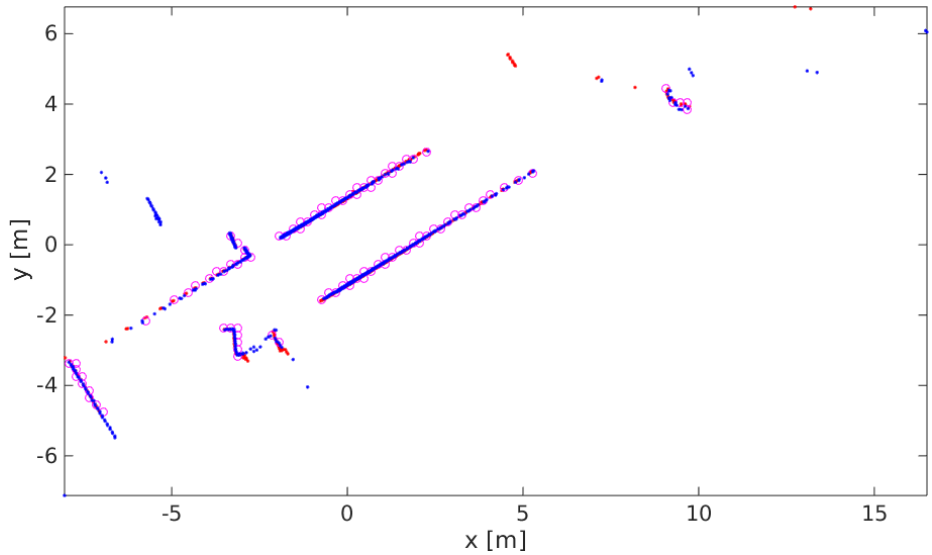
Dynamic local map generation

The building of a loop closure edge is through the scan matching algorithm to find the transformation between two topologically far away but physically close nodes. The larger the size of the shared area between two scans, the better the scan matching result will be. Because of the arbitrary property of the trajectory of the robot, if only scan pairs with high overlapping rates are searched, a lot of valid loop closures will be missed. To improve the overlapping area of candidate pairs with low overlapping areas, local map which is made from a continual sequence of neighborhood scans is used instead of a single scan. Local maps will be generated for both the target node and the source node. For the target node, the generation of a local map around the target node is static, because normally the target node is the newest node, neighbors exist only in the backward direction, certain number of neighbors will be added into the target local map. If the current node is not the newest one, nodes ahead the current nodes will also be used, only the total number of nodes used is constant. For the source local map, the source node normally has neighbors in backward and forward directions. Instead of using a static number of neighbors in each direction, the source local map is generated by maximizing the overlapping area with the source local map. The algorithm uses the ellipsoid representation and the circle representation of scans. The selection of neighbor nodes is based on the decreasing of the distance between two circle centers.
Here the loop closure is calculated based on a brute-force based scan matcher. The look up table is calculated from the $node_{from}$ together with the neighbors of the $node_{from}$, currently 6 neighbors are used (3 neighbors at each side). The current observation from $node_{to}$ together with 2 nodes from the past forms the local map. If there are nodes which are from the "future"

exist, maximal 2 "future" nodes will be inserted into the current local map. The points from the current local map will be fused and sampled before input into brute-force searching procedure to accelerating the speed. An example for demonstrating the improvement of the overlay rate after using dynamic local map generation is shown in Fig. 4.20, larger shared area between the target frame and the source frame are achieved.



(a) Scan to scan



(b) Local map to local map

Fig. 4.20 An example for demonstrating the improvement of the overlay rate after using dynamic local map generation. The top figure is scan to scan, and in the bottom figure is local map to local map. The blue color stands for source node, and the red color stands for target node. The magenta color stands for the shared area.

VoxelGrid filter based down-sampling

The matching of a local map to another local map could generate more accurate transformations by using more information. But there are also two obvious disadvantages of the local to local based matching approach, one is that the matching of local map to local map could be computational expensive. Another disadvantage is that because of the local map is overlaid from multiple scans, the raw local map could have unequal density of points, the brutal force scan matching method will tend to match the high density part which could lead to false result. So the down sampling of the local map is not only accelerating the computation speed but also increasing the accuracy of the result. The computation time is dependent to the number of points in the local map which can be seen later in the experiment result. The straight forward down-sampling way is skipping every certain points. The result from skipping based down-sampling is not stable, because the physical distribution of points is ignored, important points are equally removed during the down-sampling procedure that is why the skipping based down-sampling method works poorly. So now the question is that can down-sampling at the same time keep the important or information-rich points.
A new down-sampling method based on VoxelGrid filter is proposed here. VoxelGrid filter uses a voxelized grid (think about a voxel grid as a set of tiny 2D squares in plane). A 2D voxel grid is created over the input scan points. Then, in each voxel(i.e, 2D square), all the points presented will be approximated (i.e., downsampled) with their centroid. The idea behind the VoxelGrid filter based down-sampling algorithm is that the local points-map is converted to a points-map with every point represents a slice of the environment without duplication. Here an example of the down-sampled scans from factor 1 to 10 is shown in Figure. 4.21. The experiment is done at a computer with Intel i7 with quad-core at 2.80 GHz. The result of the down-sampling time at each factor is shown in Table 4.9. The search space is $(1.0m, 1.0m, 1.0rad)$. It is worthy to mention that for different down-sampling factor, the matching results are the same.

Brutal force based scan matching

CRSM is used for finding the transformation of loop closures. When the uncertainty between two nodes is small, CRSM will be directly used for fine searching. When the uncertainty between two local maps is high (large search space), an initial guess step will be executed before doing the fine searching.

Establish loop closure edge between nodes with large displacement

For two nodes with large displacement as a loop closure candidate, it will be too time-

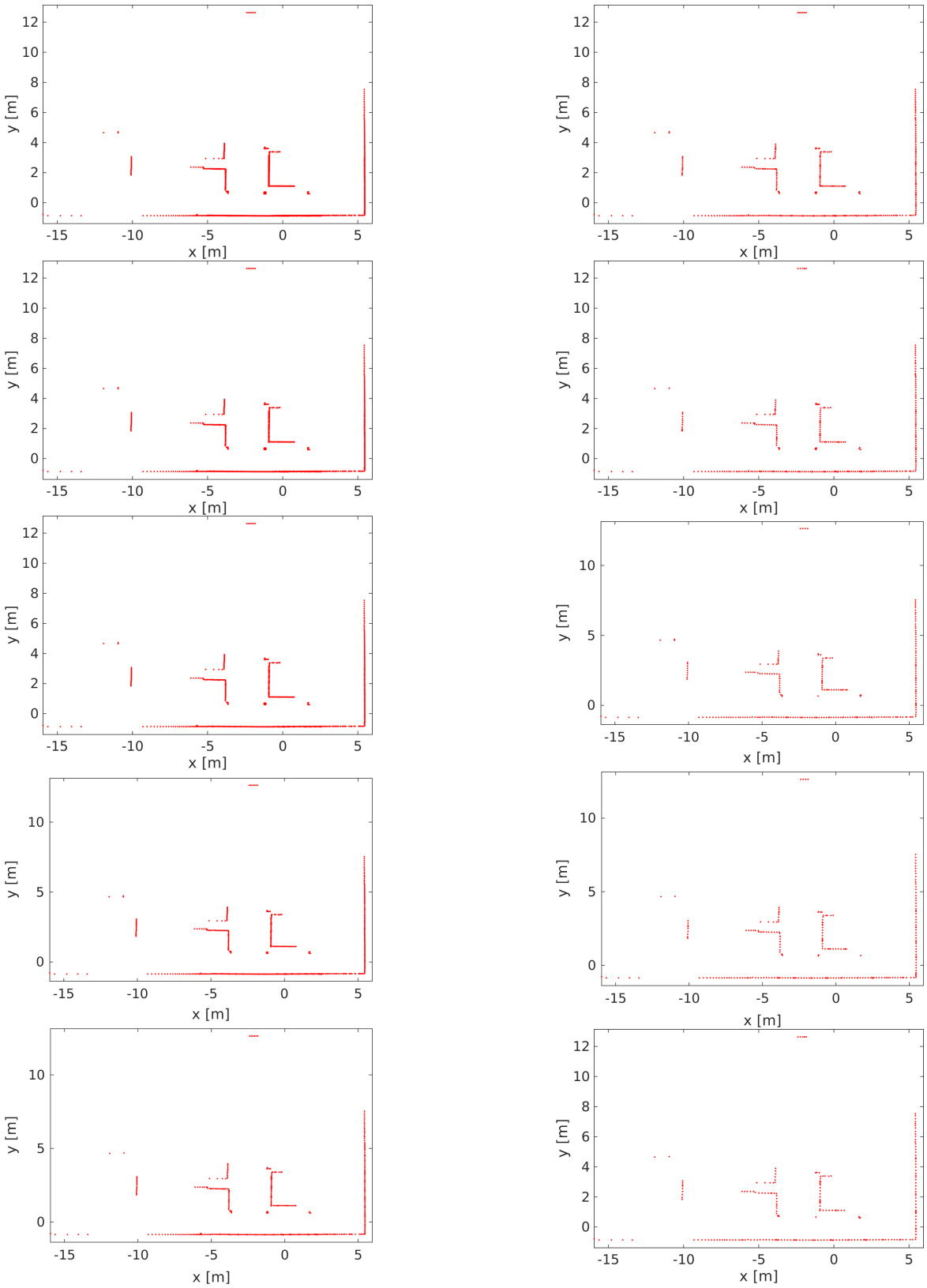


Fig. 4.21 Voxel filter based scan down-sampling. The left column top to down are resolution factor from 1 to 5, and in the right column top to down are resolution factor from 6 to 10. The resolution unit is 0.02 m.

Table 4.9 Processing time for VoxelGrid filter with resolution from 0.02m to 0.2m.

Factor	Num of points	Time used for down-sampling (ms)	Time used in total (ms)
0	2292	0.599	644.977
1	1829	73.907	529.915
2	1312	23.714	393.385
3	994	11.044	284.091
4	817	4.075	234.29
5	683	3.587	201.031
6	598	1.408	174.65
7	527	1.249	152.601
8	480	0.9	146.75
9	431	0.878	137.775
10	400	0.673	122.818

consuming by directly using brutal force method for calculating the transformation in a large search space. It will be faster if a rough initial guess could be found first and apply the rough initial guess to the brutal force method for fine searching.

To find the initial rough estimation, a brutal force based line feature matching algorithm is proposed in [82], where line features are extracted from the laser scans. If two lines from one scan (correspondences are selected from another scan) are selected, the intersection point of two lines will provide the translation, and the rotation could be calculated also. Initially, the correspondences of lines in two scans are unknown, so a brutal force based method is used for every possible combination of two correspondences lines between the reference and target scan. It is obvious that the line feature based method requires strong line features existing in the environment.

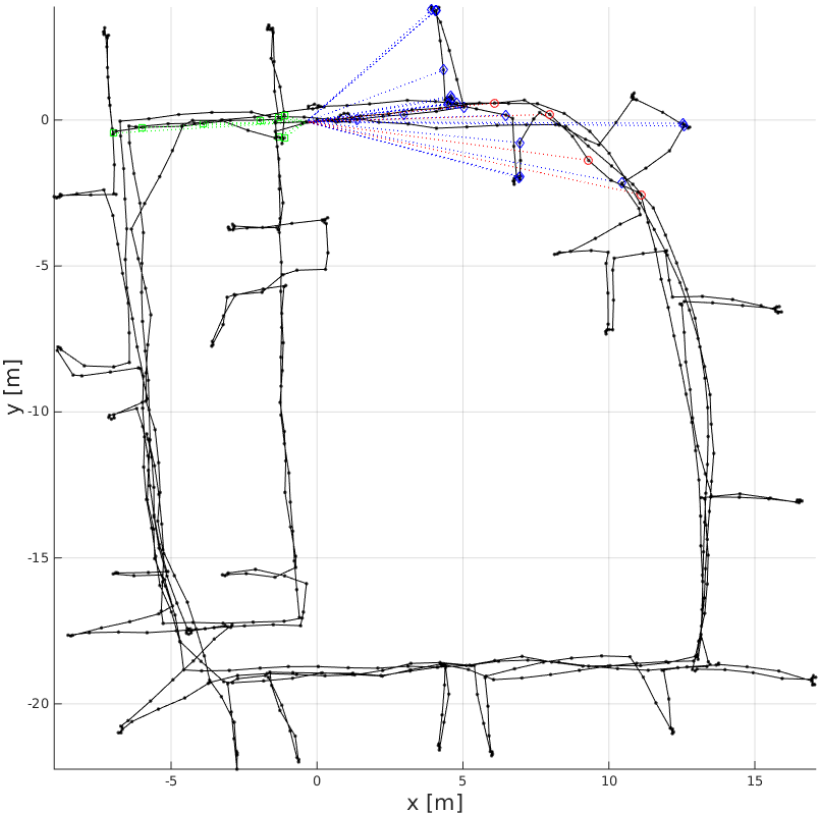
Here the author proposed a new method for searching the rough estimation by doing brutal-force search in a lookup table at ultra low resolution without the need of extra features, which is a more general approach for searching the rough estimation. The ultra low resolution for rough initial estimation is set to 0.5 meter, and when the odometry displacement is bigger than 1.0 meter, the ultra low resolution will be used to calculate a rough initial guess. If the match score is above a certain threshold (empirically 0.2 is used here), the solution will be accepted for the next step doing accurate matching, otherwise, the loop closure candidate will be ignored.

To test the performance of ultra-low resolution CRSM, 58 loop closures candidates with Mahalanobis distance bigger than 6 are manually selected from MIT-Killian dataset with ground truth information for benchmarking. The search space for ultra-low resolution CRSM is set to three times of the uncertainty between two nodes in the loop closure candidate.

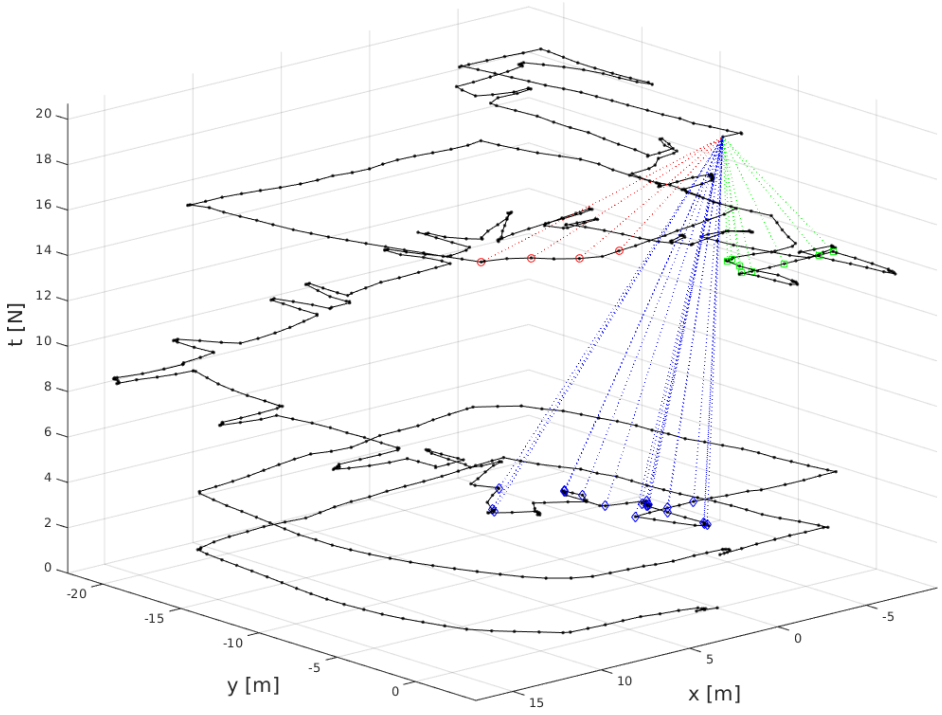
50 out of 58 loop closures are successfully found, 8 failed cases are due to the perceptual aliasing in the environment, for example, featureless corridors or two places with similar features. Those failed cases will be handled in the validation step in the following section. The ultra-low resolution CRSM takes average 0.04 seconds in a laptop with Intel i7 operating at 2.8 GHz. Its runtime is negligible.

Priority based loop closure searching

Those selected nodes N_4 in *Preprocessing* in Fig. 4.18 will be clustered into several groups based on their node IDs, and candidate nodes in each group will be sorted based on the size of their intersection area with the current node, the larger, the fronter. An example of clustered groups is shown in Fig. 4.22. The loop closure detection will be executed based on clustered groups, if one loop closure is found in one group, then the rest candidate nodes in this group will be ignored. The process goes into the next group following the same principle until all groups have been searched. The purpose of this process is to keep the real time performance and without losing important loop closure edges. The detailed algorithm is summarized in Algorithm 6.



(a) Top down view of candidate nodes in ordered layers.



(b) 3D view of candidate nodes in ordered layers.

Fig. 4.22 Example of candidate nodes in ordered layers. Different colors stand for different layers. Dashed lines stand for candidate loop closures.

Algorithm 6 Priority based loop closure searching

Require: $CandidatelistL = [c_i], c_i = (p_i, n_i), i \in [1, N]$
Require: $d_{layer}, s_{depth}, max_{check}, max_{num}$
Ensure: $N > 0$

```
checkedlist
num_found
layers = [[]]
layer = []
Add c_1 into layer
for i = 2:N do
    if (n_{i-1} − n_i) > d_layer then
        From big to small sort layer based on the value of p
        Add layer into layers
        Clean layer
        Add c_i into layer
    else
        Add c_i into layer
    end if
end for
if layer not empty then
    From big to small sort layer based on the value of p
    Add c_i into layers
end if
while max_check > 0&&num_found < max_num do
    doSearch
end while
```

4.5.3 Loop closure validation

Even though the local to local based CRSM algorithm is a robust method to generate loop closure transformations, the recognition and rejection of erroneous transformations are still needed. The loop closure assumption $l(from\_id, to\_id)$ suggested from the previous paragraphs will be validated before adding to the graph. The validation criteria include matching goodness check, negative segment correspondence check, and local graph optimization check.

Local to local based matching goodness metric

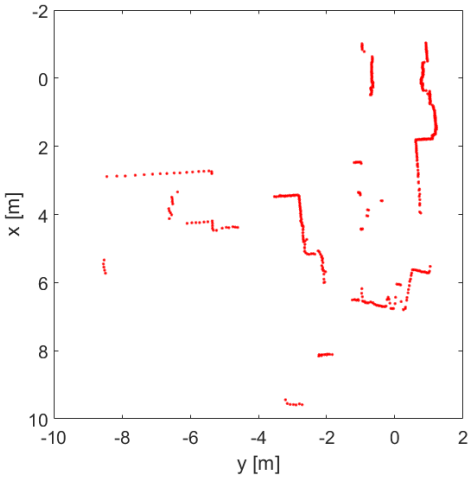
The result from the local-to-local based scan matching algorithm will output a goodness score of the current estimation. This metric provides a metric for using full area from both target and source local map.

Histogram intersection metric

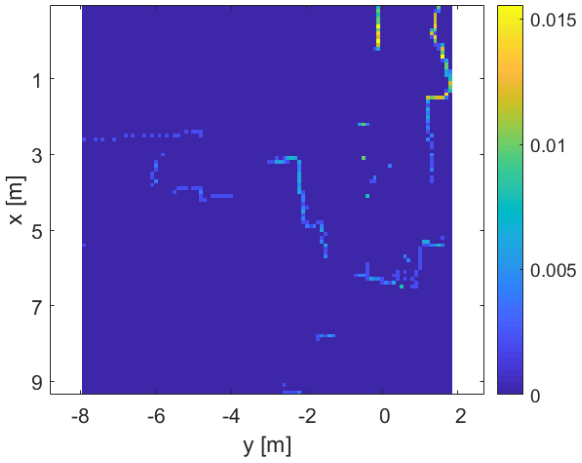
The histogram intersection metric is first proposed in [8] for the classifying of images, then this metric is applied for measuring the similarity of matched 2D laser scans in [27]. In this work, the first time the metric is extended to measure the similarity of matched local maps. Using the estimation from the local to local map matching algorithm, the local maps are projected into the same coordinate. Then, for each local map, a two-dimensional histogram is built by inserting points in the local map into a 2D occupancy grid map. Next the two histograms are correlated using the intersection kernel [8]:

$$c = \sum_{i,j} min(h_1(i,j), h_2(i,j)) \tag{4.23}$$

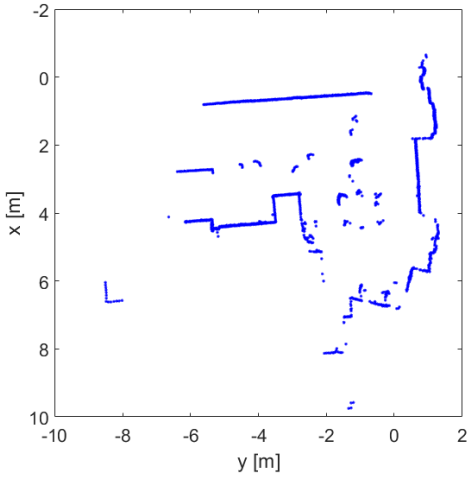
where $(h_1(i,j)$ and $(h_2(i,j)$ stands for the i,j-th cell of the respective histograms. When $c = 1$ means the histograms are identical and $c < 1$ when they differ.
Because of the accumulation of neighbor scans into the local map, the density property of the local map cannot correctly reflect the histogram intersection metric, the local map is first filtered by a voxel grid filter in a high resolution, and then the filtered local map is applied into the calculation of the histogram intersection metric at a low resolution. The histogram intersection metric provides a different view for measuring similarity compared with the local to local based matching goodness metric, because here only the intersection area between two local maps are used for the calculation. An example of histograms for both local map and the intersected part are shown in Figure.4.23.



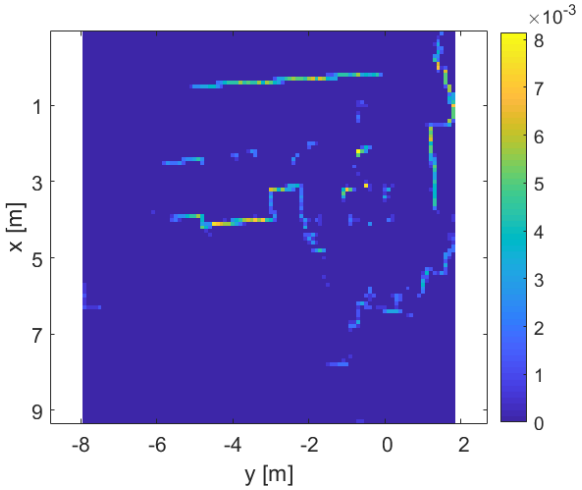
(a) source local map.



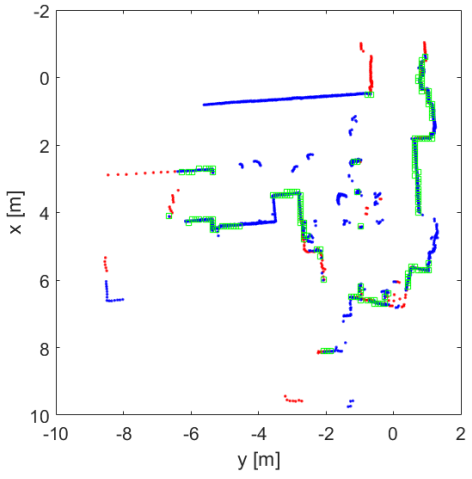
(b) source histogram.



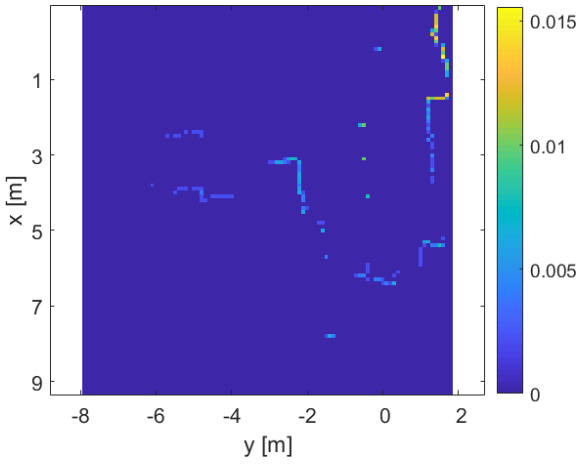
(c) target local map.



(d) Target histogram.



(e) Shared area.



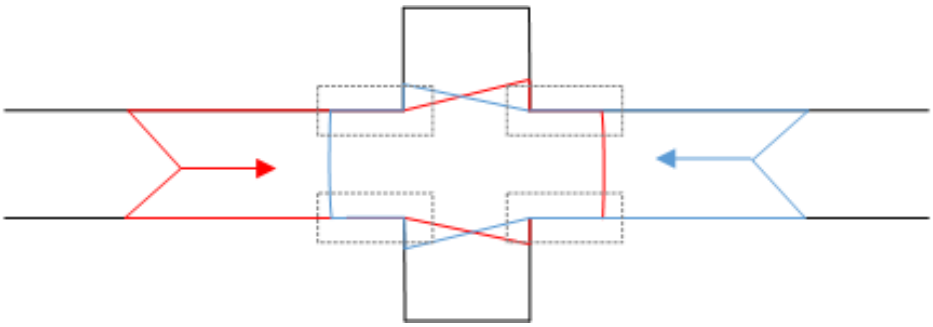
(f) Correlative histogram.

Fig. 4.23 Histogram intersection example. The size of the cell in 2D histogram is 0.1m*0.1m. The histogram intersection result is 0.33425 in this case. The geometry complexity of the shared area is 0.865371.

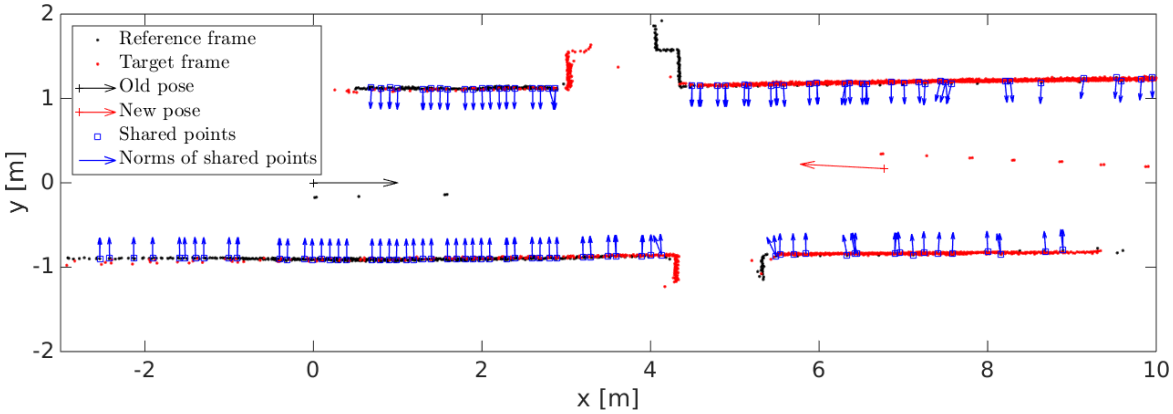
Complexity metric of the shared area

The normal vectors of the points in the shared area will be calculated as $N = [\vec{n}_1, ..., \vec{n}_p]^T$. The correlation matrix $R = N^T N$, the eigenvalues of R are λ_1, λ_2, assuming $\lambda_1 \leq \lambda_2$. The complexity metric is defined as $c = \lambda_1 / \lambda_2$ following the work in [27]. If all the points are in one direction, c will close to 0, otherwise if the geometry of the environment is complex enough, c will close to 1.

Both the scan matching goodness and the correlative metric measure the similarity of two local maps, the difference between them is the scan matching goodness metric measures the full scale, and the correlative metric measures only the intersected rectangle area. As shown in Figure 4.24b, when the scan matching goodness or the correlative metric is very high, it does not mean the loop closure hypothesis is true.



(a) Illustration of the corridor effect, only the horizontal lines inside the dashed rectangles can be observed from both sensor poses.



(b) Loop closure hypothesis with high correlative metric but low complexity metric.

Fig. 4.24 Loop closure hypothesis with high correlative metric but low complexity metric example. The scan matching goodness of the reference map is 0.701, and the false scan matching pose has the goodness of 0.751. So the output pose will be the false scan matching pose. The histogram intersection result is 0.276 in this case. The geometry complexity of the shared area is 0.039.

The number of matched segments with opposite norms metric

Because of the inherent property of the brutal force based scan matching algorithm will always try to maximize the goodness, the sensor viewpoint-awareness is used here for validation. The corridor-like environment awareness is not used here because the accumulated scan matching error could be large. When a transformation between a local map with another local map is established, the corresponding segments will be found based on the closet neighborhood principle after transforming those two local maps into the same coordinate using the transformation hypothesis. The main idea here is to identify line features in the local maps, and find corresponding line features by considering unique normal vectors of line features. By taking the unique normal vector, the correspondent lines from two sides of a wall could be avoided. For a successfully matched loop closure hypothesis, the norms of segment correspondences will agree with each other as shown in Fig. 4.25, all five correspondences marked with blue dashed lines have similar norms. When a falsely matched loop closure hypothesis is found in Fig. 4.26, for better understanding, the images of the two view-points is shown in Fig. 4.27. The thickness of the wall is ignored by the scan matching algorithm, but with the detection of segment correspondences, this false hypothesis can be safely rejected due to two pairs of correspondences with opposite norms. The line correspondence and complexity check takes only a few milliseconds for computation. This metric provides a aspect of the view of the sensor to the environment. Even though at different view points, the same line in the environment will always have the same norm through the author's definition of norm vector direction.

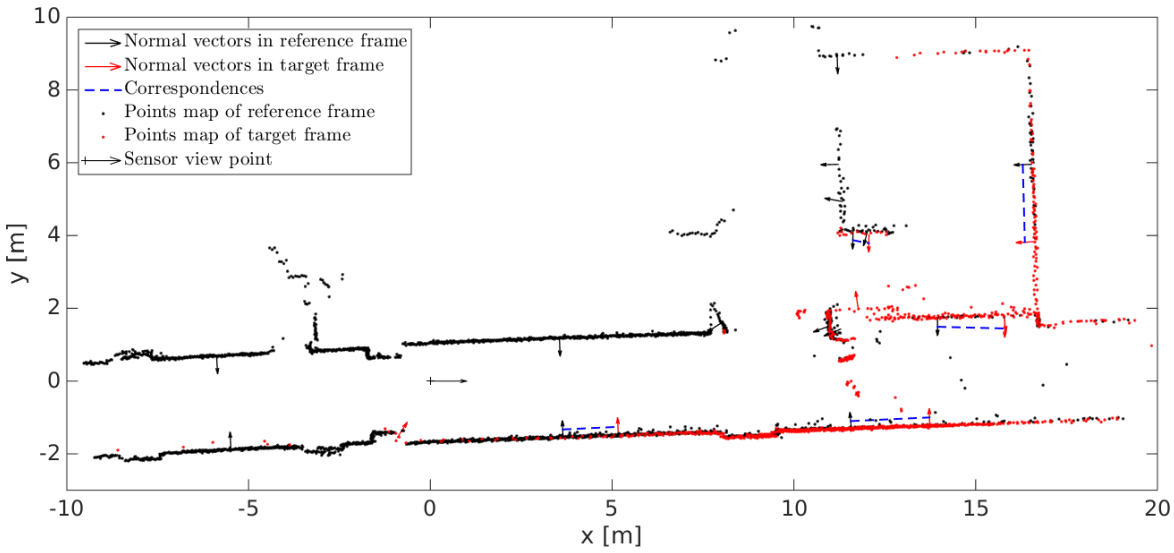
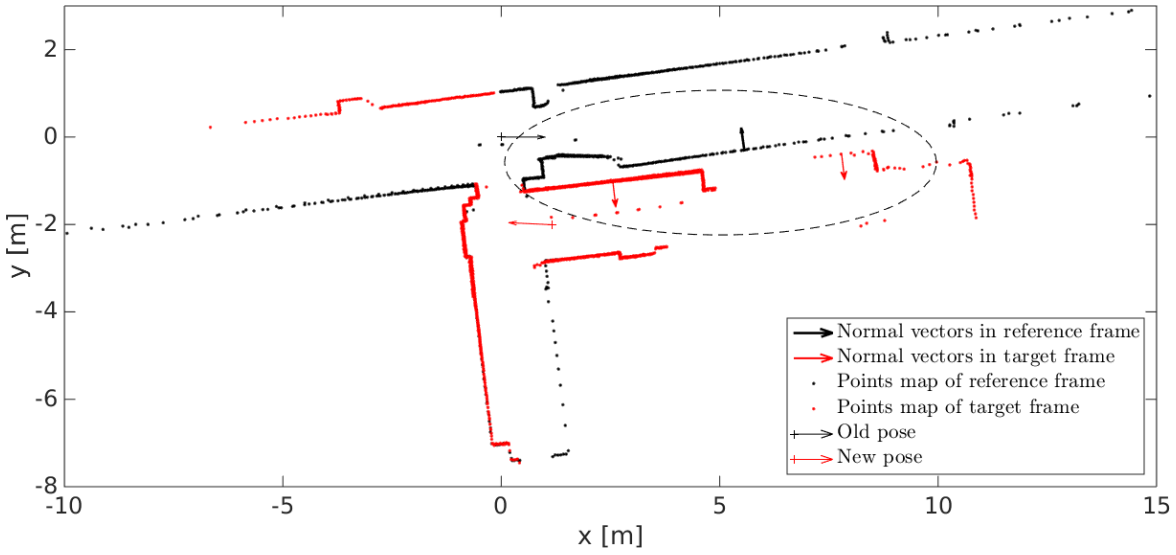
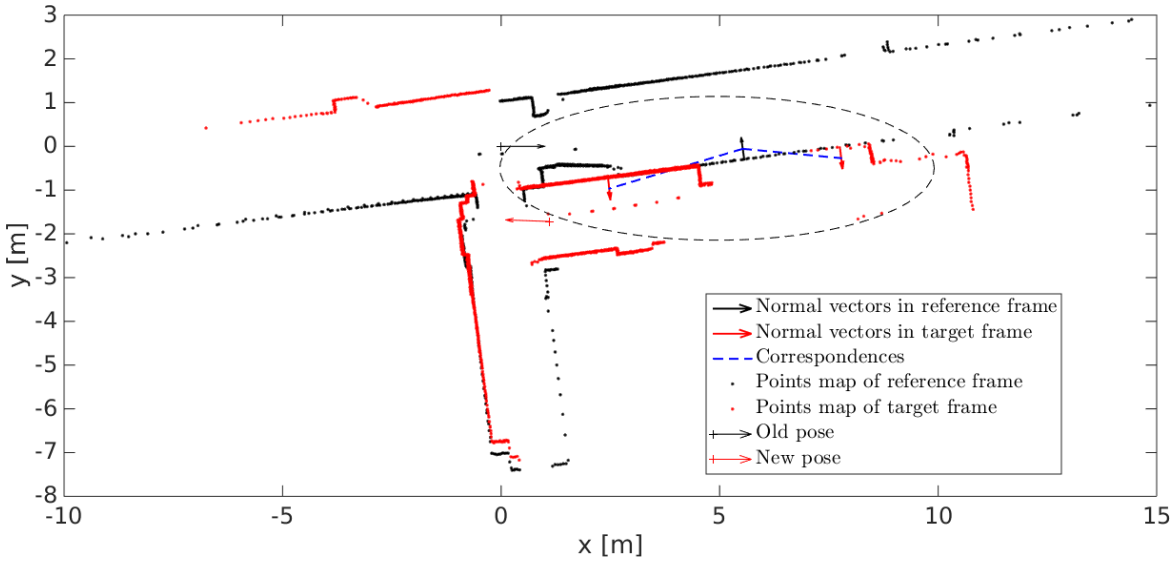


Fig. 4.25 Closest correspondences between reference frame and target frame.



(a) Reference map.



(b) Map with false scan matching pose.

Fig. 4.26 Negative segment pair check example. The scan matching goodness of the reference map is 0.383, and the false scan matching pose has the goodness of 0.399. So the output pose will be the false scan matching pose. The histogram intersection result is 0.259 in this case. The geometry complexity of the shared area is 0.935.

(a) The image from the old viewpoint in Fraunhofer IOSB-AST building.

(b) The image from the new viewpoint in Fraunhofer IOSB-AST building.

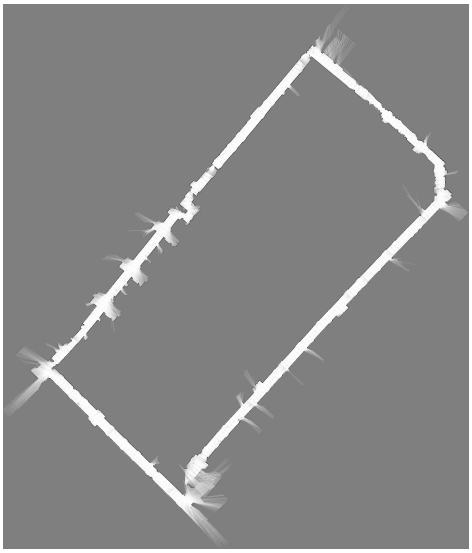
Fig. 4.27 From the images at old and new viewpoints, we can see the two sides of the wall (ca. 30 cm thick). Scan matching algorithms will normally ignore the thickness of the wall and merge them a line.

Local graph optimization

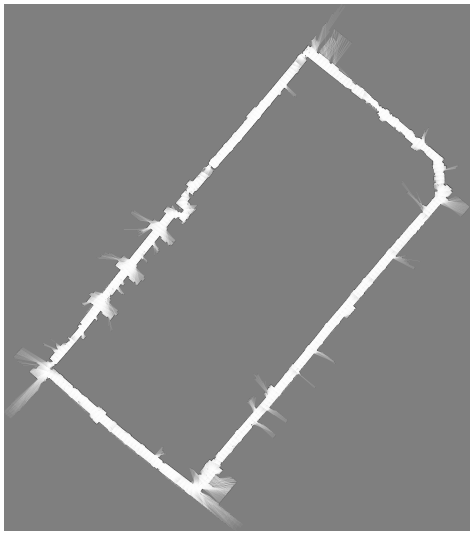
Even scan matching goodness, correlative metric and complex metric all satisfy the corresponding threshold, the loop closure hypothesis still could be false because of perceptual aliasing, for example similar structure at different places. Those kind of errors could be detected by combing more information from the graph. Instead of directly adding the loop closure into the global graph, the loop closure hypothesis is tested in a local graph which is built based on the shortest path connecting two nodes in the loop closure hypothesis. Based on the Dijkstra's Algorithm, a shortest path which connects $node_{from}$ and $node_{to}$ will be found. Nodes and edges in the shortest path will be added into the local graph, in the end, the loop closure hypothesis will also be added into the graph as an edge. Then the local graph will be optimized, if the entropy of the map after graph optimization decreases, then the loop closure hypothesis will be accepted and added into the global graph, otherwise the hypothesis will be rejected as shown in Fig. 4.28. The hypothesis in Fig. 4.28a and 4.28b will be accepted, and the hypothesis in Fig. 4.28c and 4.28d will be rejected.

Experiment

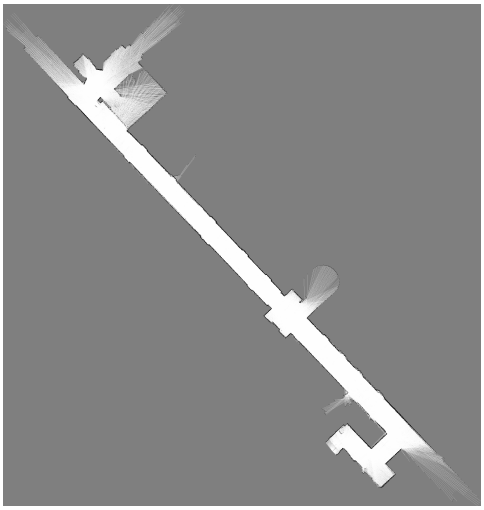
The selection of the thresholds for scan matching goodness, correlative rate and complex rate is analyzed using the MIT Killian Court dataset. Because the MIT Killian Court dataset is one of the largest indoor datasets, includes a lot of long corridors, and the ground truth is provided. Using the loop closure detection methods described in the last section, a set of loop closure hypotheses is generated. The combination of different correlative rates and complex rates is analyzed in Fig. 4.29a. The combination of different scan matching goodness scores and complex rates is analyzed in Fig. 4.29b. A ROC is plotted in Fig. 4.29c. The combination of all three criteria outcomes the best result, and the correlative rate metric is more effective the scan matching goodness metric.



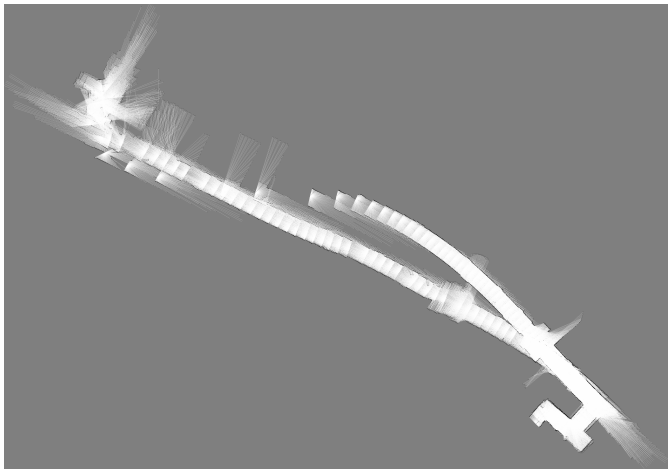
(a) Before local graph optimization, entropy 92339.9



(b) After local graph optimization, entropy 87080.7

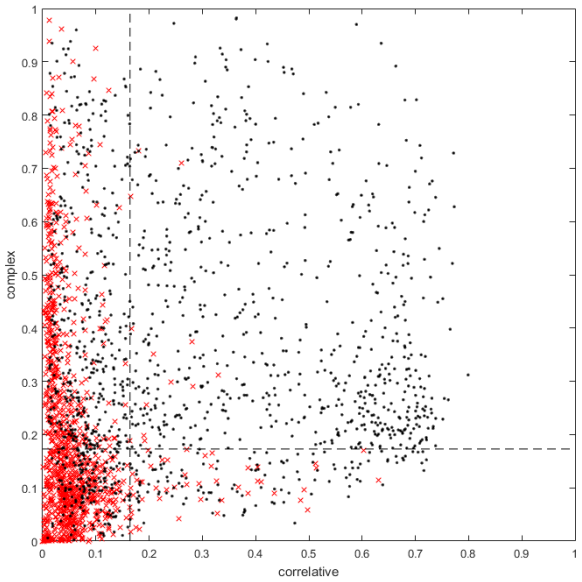


(c) Before local graph optimization, entropy 28039.5

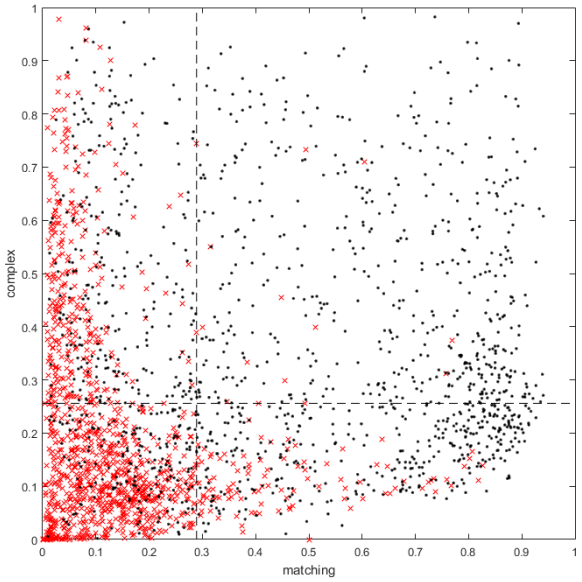


(d) After local graph optimization, entropy 96260.4

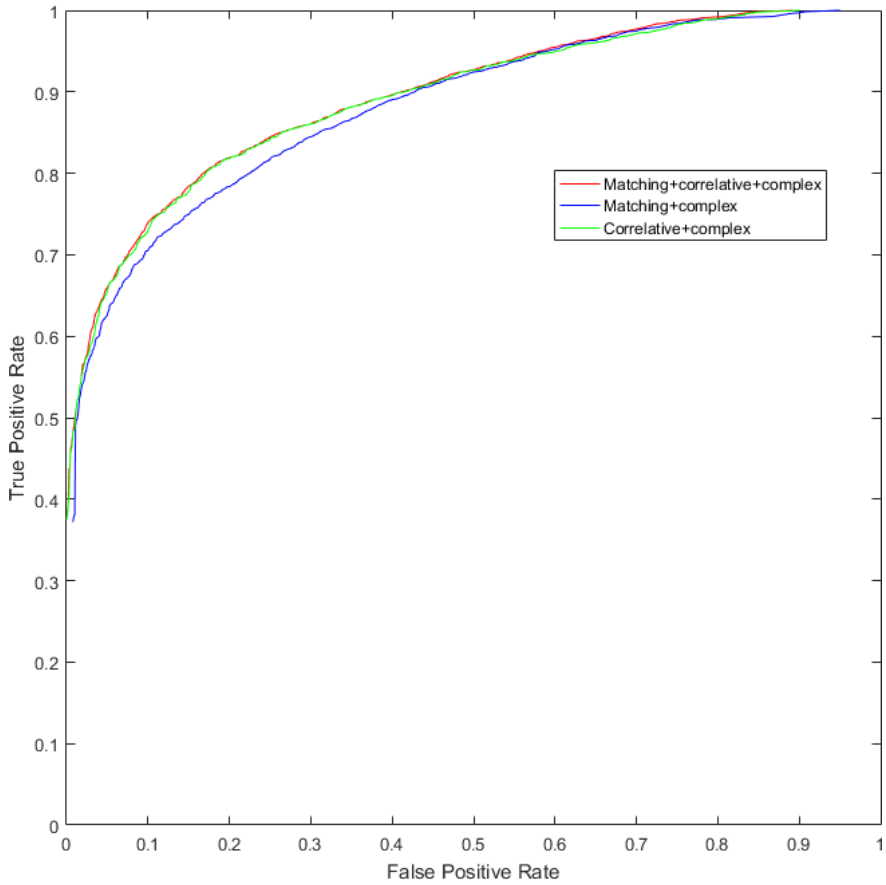
Fig. 4.28 Local Graph optimization for validating the loop closure hypothesis.



(a) correlative with complex: correlative threshold at 0.164 and complex threshold at 0.173.



(b) matching with complex: matching threshold at 0.289 and complex threshold at 0.256.



(c) ROC.

Fig. 4.29 A ROC for Dataset MIT Killian is shown to demonstrate the effectiveness of different validation methods. Let assume 0.01 false positive rate is allowed, all three criteria method has 0.727 accuracy with matching threshold 0.263, correlative threshold 0.136, complex threshold 0.171, correlative and complex method has 0.721 accuracy, matching and complex has 0.665 accuracy.

4.6 Implementation of the adaptive Graph SLAM

4.6.1 Incremental Graph Optimization

Most popular way of doing graph optimization is batch optimization which means the graph is optimized after the full graph is built. Different from batch graph optimization, incremental graph optimization optimizes the graph when one or a few loop closure edges are added into the graph. The benefit of incremental graph optimization is that the optimized nodes could limit the searching area of future nodes for finding loop closures, and the best estimate is available at any time. The Incremental Mapping and Smoothing (iSAM) [54] solver is used as the graph optimization back-end in this work to serve the incremental purpose.

4.6.2 Fine backward loop closure detection

To further improve the quality of graph optimization based SLAM, when a loop closure is accepted and the graph is optimized, the nodes between two nodes of the accepted loop closure are rechecked and saved into a waiting list for later checking. To limit the computation time, every node in the graph has a limited number of chances being checked for finding loop closures, when the checking time of a node is above the threshold, this node will be not checked any more. The nodes in the waiting list will only be checked when the robot is at standby mode and all the new messages are processed. The pipeline of the backward fine optimization is shown in Fig. 4.30.

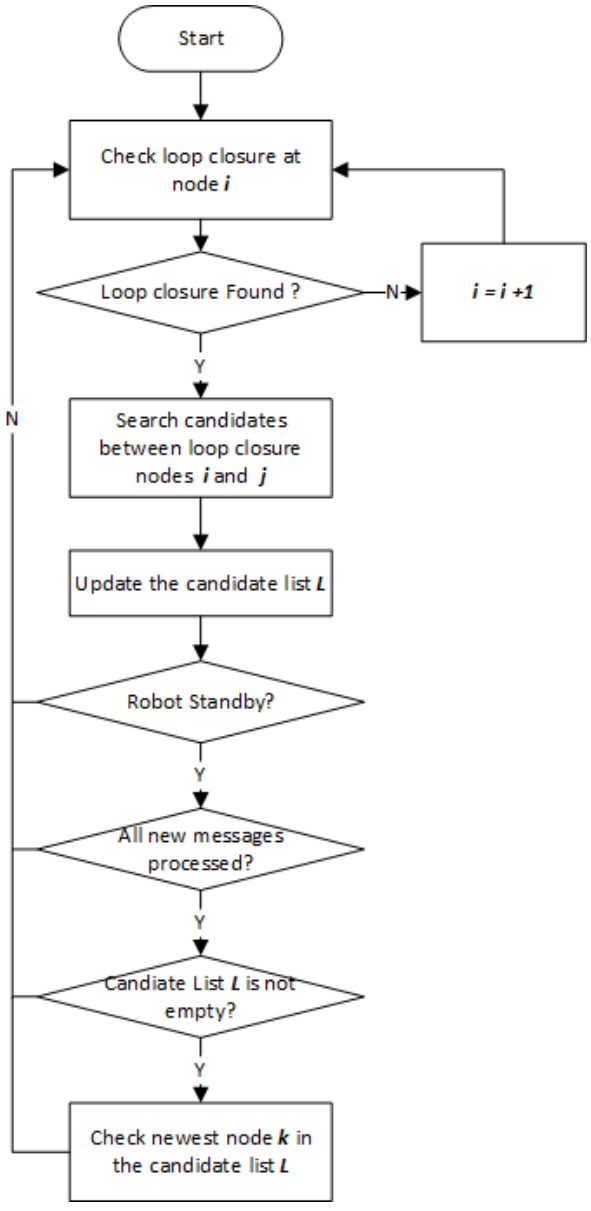


Fig. 4.30 The pipeline of backward fine optimization

4.6.3 Parallel processing of graph based SLAM

The real-time Graph-based SLAM is designed as a multiprocessing program to satisfy different requirements. To fulfill the real-time requirement, a separate thread listens for incoming messages from the sensors with the highest priority. The processing of the messages may be time consuming and slower than the rate of incoming sensor messages, at the same time all required messages need to be processed, so messages need to be stored and the process runs in the background in a separate thread. Because normally the transformation from scan matcher is much more accurate than odometry, the basic edges are built by matching two consecutive scans. The loop closure detection is based on the basic edges to minimize the search area (computation power and time are saved). When a loop closure is found, and graph optimization is executed, the map and trajectory should also be updated, this is done in a separate thread. To make sure the read and write of the same memory block are not contradicted, all related data must be thread safe. The design of the related data structure and multi-threading structure are presented in Fig. 4.31. In total, there are six threads used. Boost [1] is used for the implementation of the multi-threading program. The synchronization of different threads depends on boost::mutex.

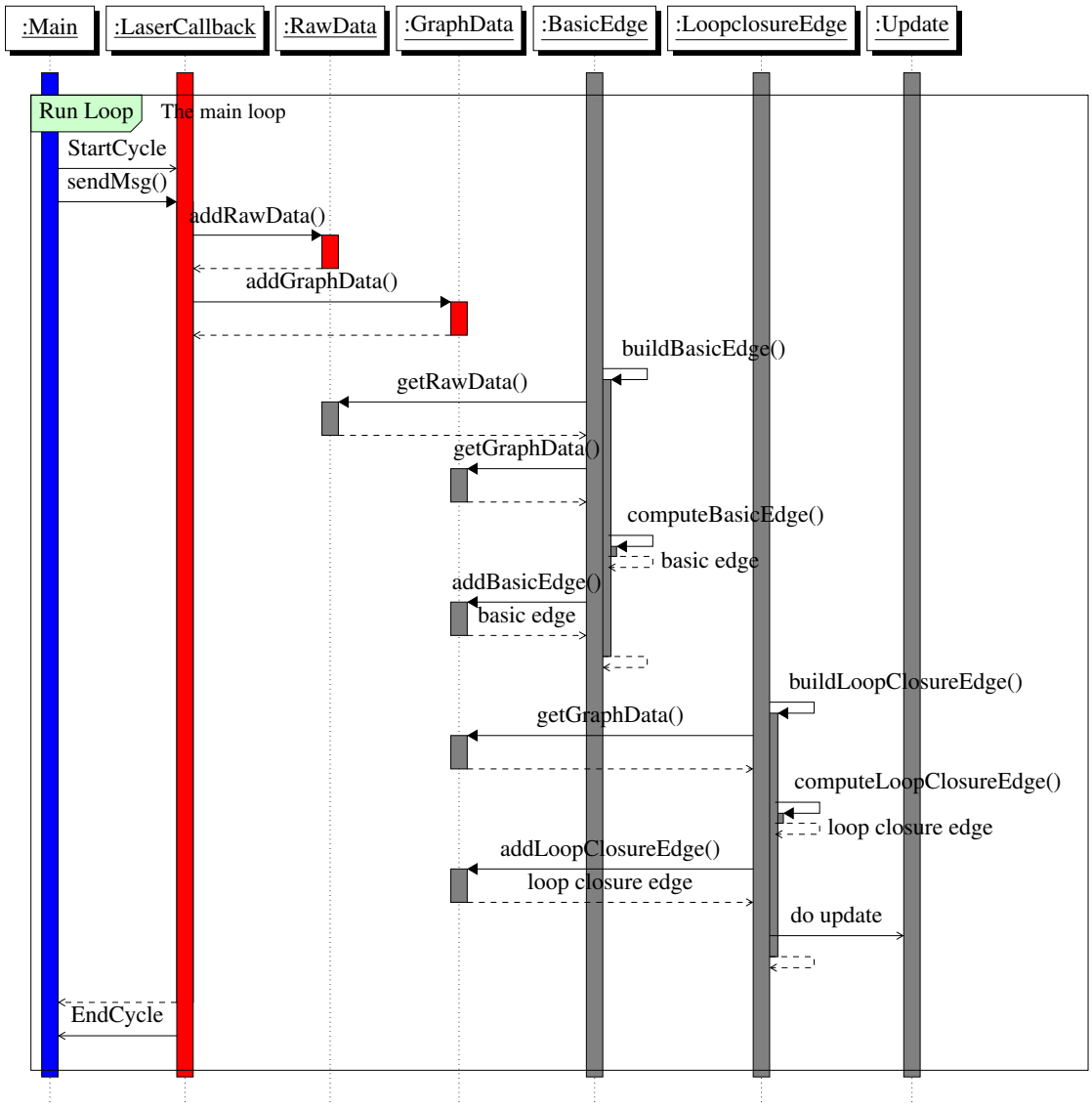
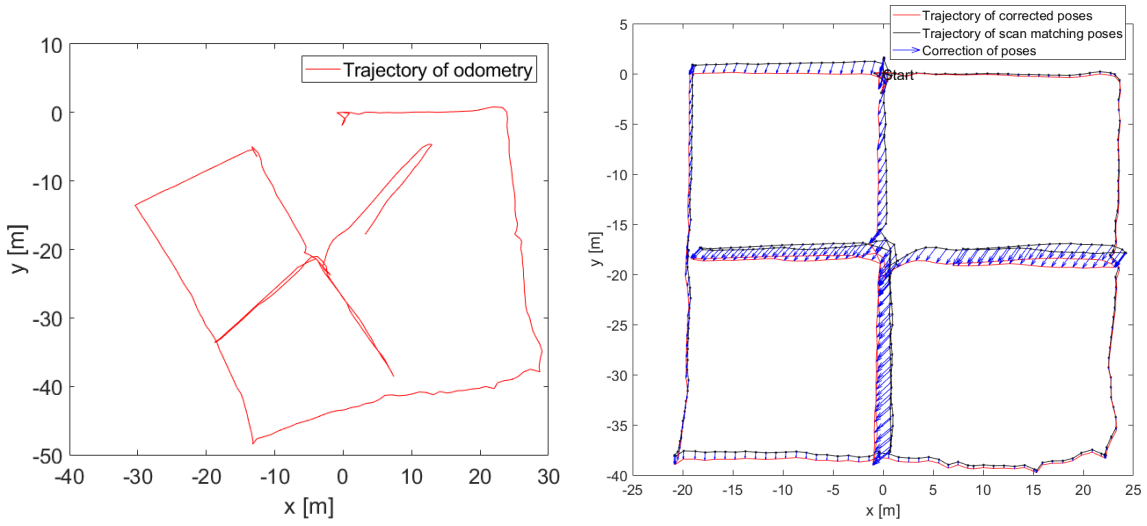


Fig. 4.31 UML Sequence diagram of Graph SLAM

4.7 Results

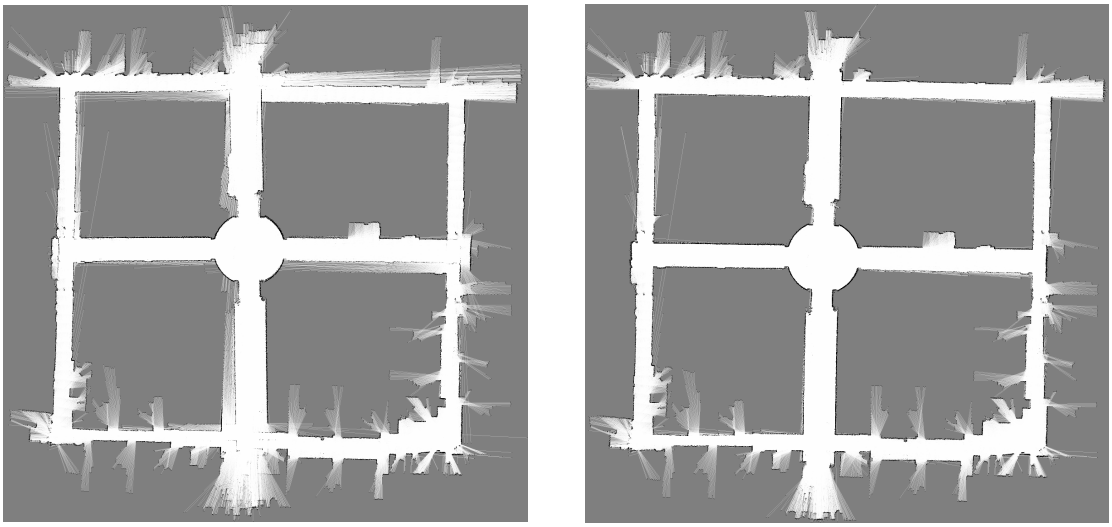
4.7.1 ACES Building

Applied Computational Engineering and Sciences (ACES) Building is located on the University of Texas at Austin campus, which was provided by Patrick Beeson. The raw odometry path is shown in Figure. 4.32a. The path of scan matching poses and the path after graph optimization is shown in Figure. 4.32b, and the corresponding maps are shown in Figure. 4.33a and Figure. 4.33b.



(a) ACES odometry trajectory (b) ACES corrected trajectory

Fig. 4.32 ACES trajectory.

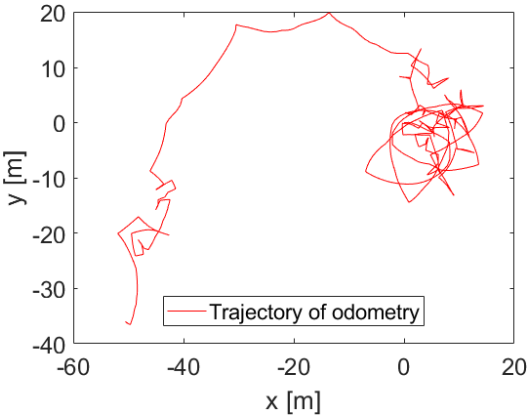


(a) ACES map from scan matcher poses (b) ACES map from optimized poses

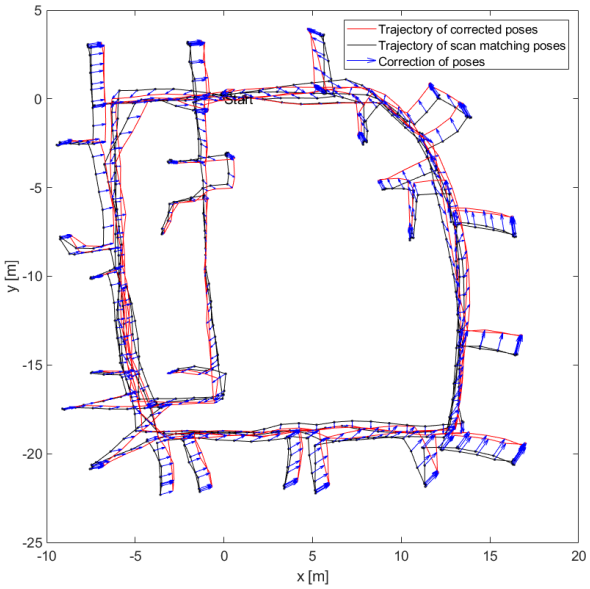
Fig. 4.33 ACES maps.

4.7.2 Intel Research Lab

Intel Research Lab (Seattle) dataset was provided by Dirk Hähnel. The odometry drifts significantly in this dataset as shown in Fig. 4.34a. The author's basic edge building algorithm could provide close to truth scan matching poses. The graph is shown in top-down view and 3D view, and the map from optimized poses is displayed in Fig. 4.35.



(a) Odometry path.

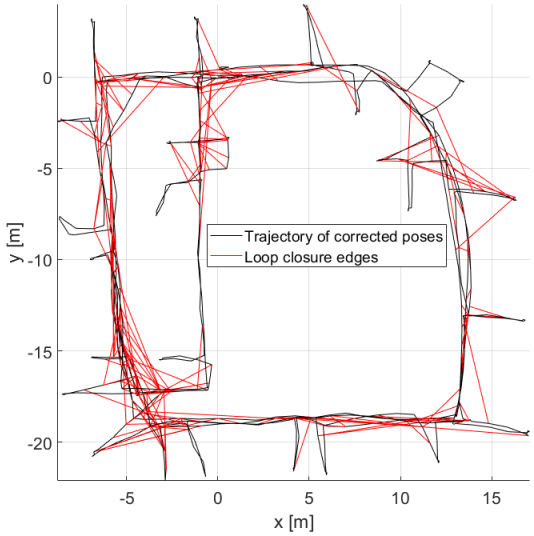


(b) Optimized path.

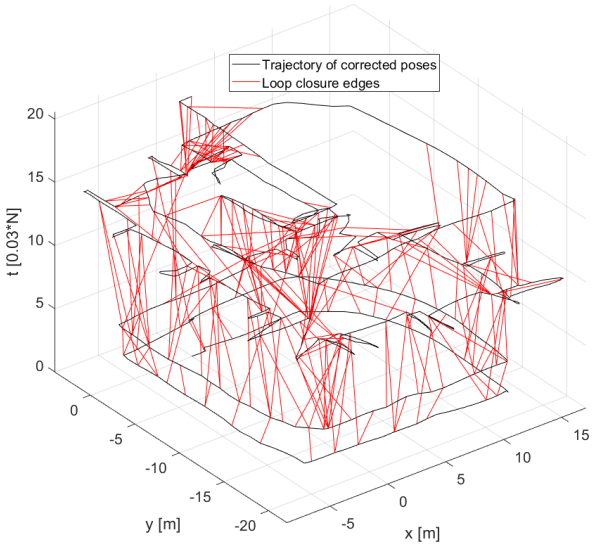
Fig. 4.34 Intel odometry trajectory and optimized trajectory.

(a) Intel map from scan matcher poses

(b) Intel map from optimized poses



(c) Top-down view.

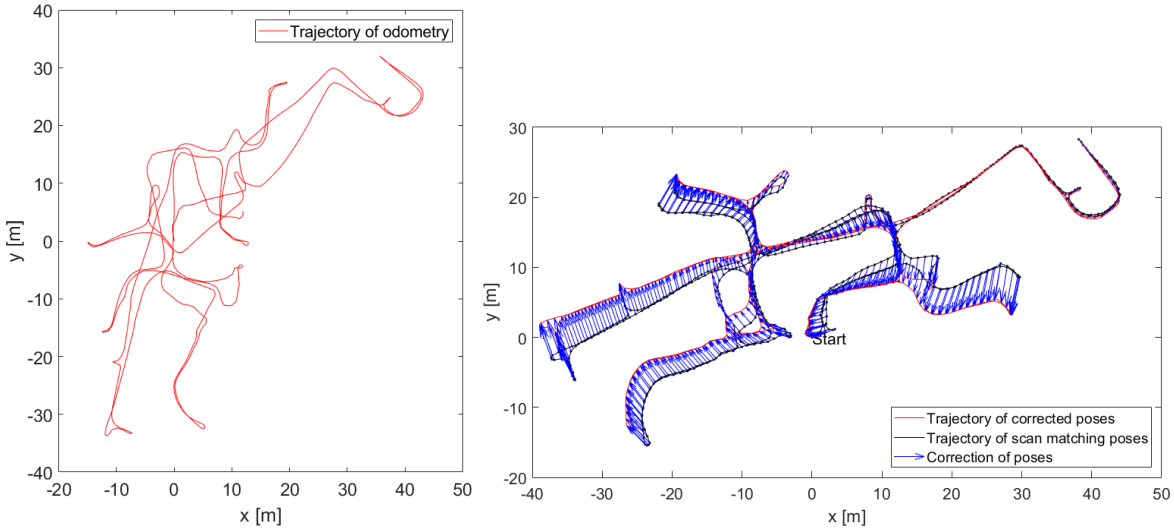


(d) 3D view.

Fig. 4.35 Intel.

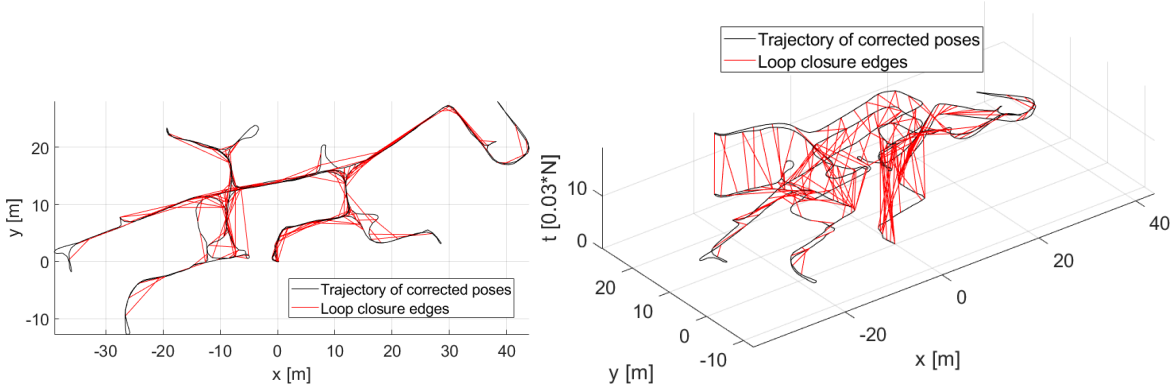
4.7.3 MIT Stata Building

The MIT Stata Building dataset is record by a PR2, the dataset represents typical clustered indoor environments. the result is shown in Fig. 4.36.



(a) Odometry path.

(b) Optimized path.



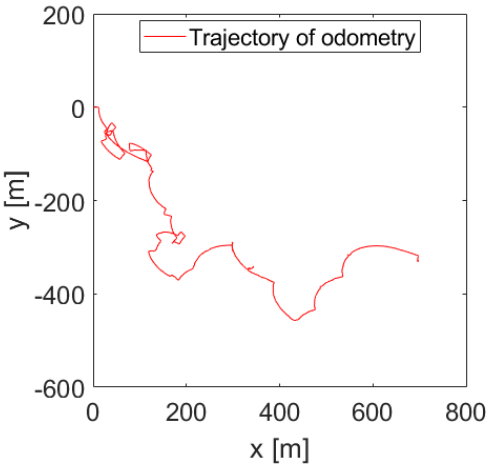
(c) Top-down view.

(d) 3D view.

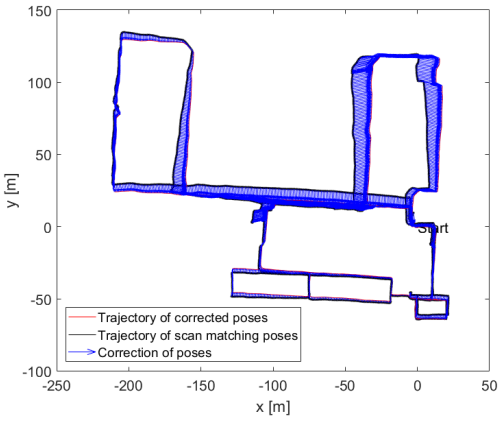
Fig. 4.36 MIT Stata.

4.7.4 MIT Killian Court

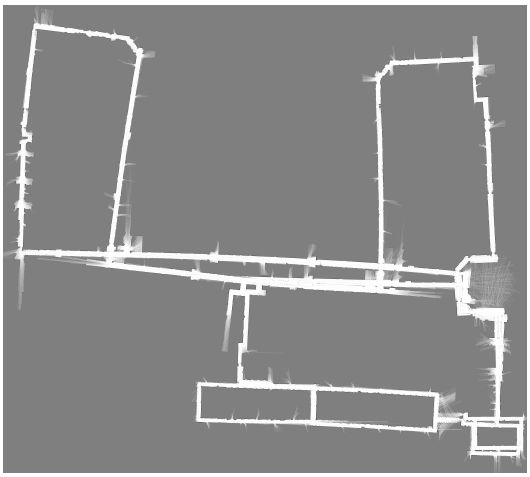
MIT Killian Court dataset was provided by Mike Bosse and John Leonard. The trajectory of this dataset is approximately 1.9 km, which is the longest one among all datasets used in this work. The raw odometry is quite noisy as shown in Fig. 4.37a, and the trajectory is made of corridors (another name of this dataset is called as "Infinite Corridor"). The author's proposed method could find large gap loop closures effectively in this difficult case, and the result is shown in Fig. 4.37.



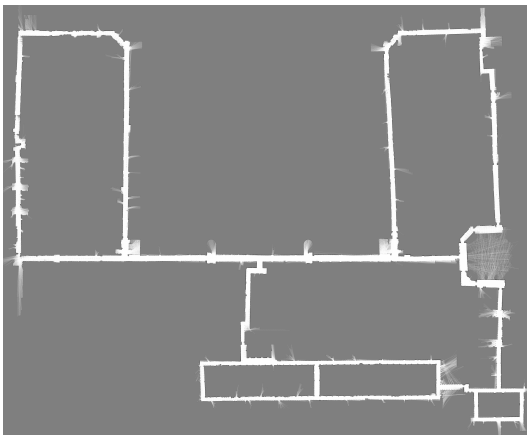
(a) Odometry path.



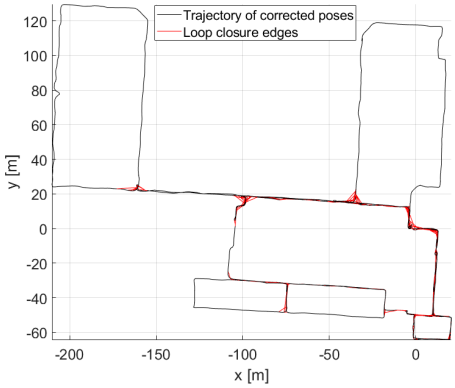
(b) Optimized path.



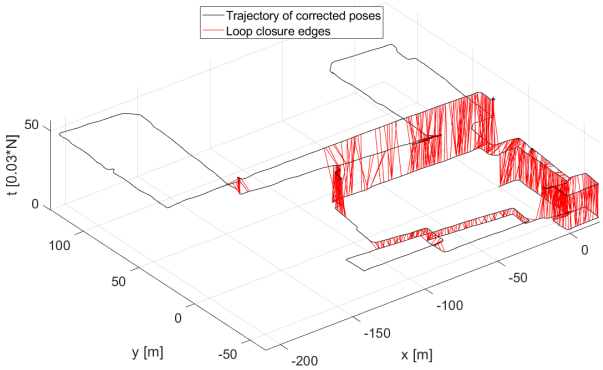
(c) Map from scan matcher poses



(d) Map from optimized poses



(e) Top-down view.



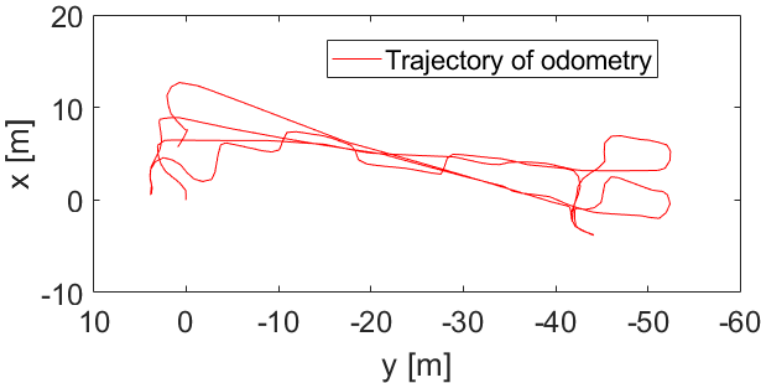
(f) 3D view.

Fig. 4.37 MIT Killian.

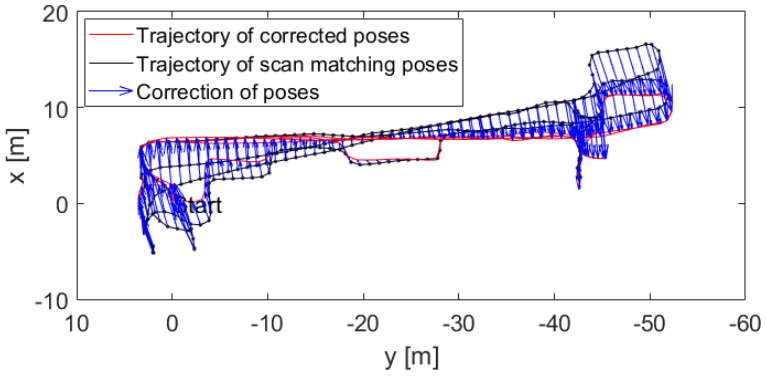
4.7.5 Fraunhofer IOSB-AST Building

Fraunhofer IOSB-AST building is located in Ilmenau, Germany. A research facility for Advanced System Technology. The ground plan of this building is shown in Fig. 4.38a. The map generated from the author's method is overlaid with the ground plan, and they fit very good. Only one door matches not so good, the reason is that the door is renovated. The path of odometry poses, path of scan matching poses and path of optimized poses are shown in Fig. 4.39. The graph built from the author's method is shown in Fig. 4.40b, the z axis stands for the time when the node is created.

(a) FhG IOSB-AST ground plan

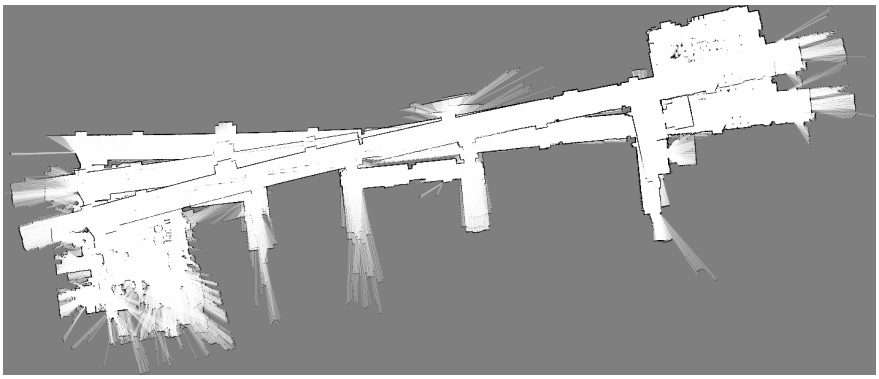


(b) Odometry path.

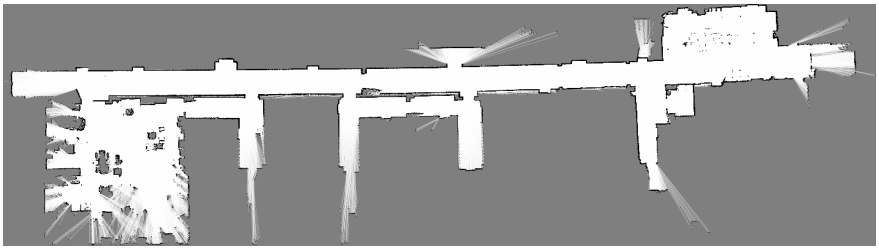


(c) Optimized path.

Fig. 4.38 FhG IOSB-AST odometry path and corrected path.

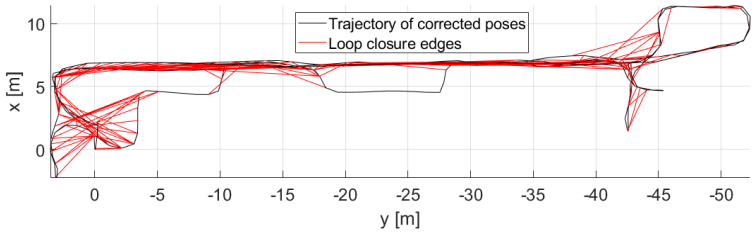


(a) Intel map from scan matcher poses

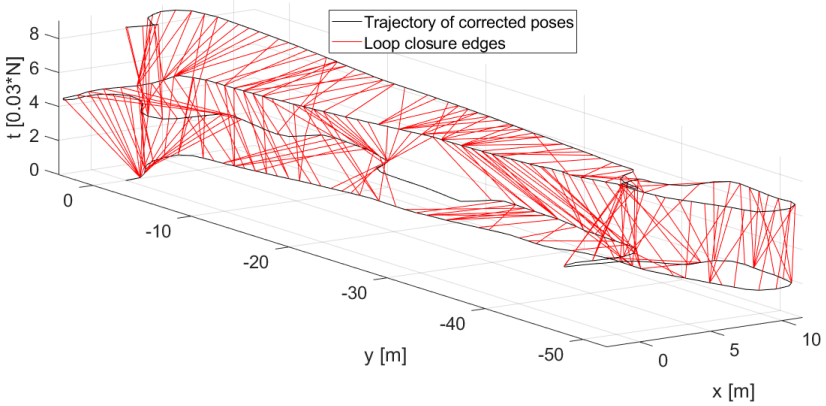


(b) Intel map from optimized poses

Fig. 4.39 FhG IOSB-AST maps.



(a) Top-down view.



(b) 3D view.

Fig. 4.40 FhG IOSB-AST graph.

4.7.6 Real-time performance of the Graph SLAM

To measure the real-time performance of the proposed Graph SLAM, three datasets, namely Intel Research Lab, ACES Building and MIT Killian Court, are published at speed from factor 1 to 4. Speed factor 1 means the original publish rate, and speed factor 4 means 4 times of the original publish rate. An embedded computation unit ODROID-XU4 and a laptop Lenovo T540p are used for the experiment, and detailed information is shown in Chapter 3 in Table 3.4. The real-time performance t_{rt} is measured by the time difference between the time t_{scan} when the last scan measurement is acquired and the time t_{end} when all the nodes in the graph are optimized (For measuring the real-time performance, the time used in the backward optimization phase is not included).

$$t_{rt} = t_{end} - t_{scan} \tag{4.24}$$

Table 4.10 The real-time performance of Graph SLAM on ODROID-XU4

Dataset name	t_{rt}			
	x1	x2	x3	x4
Intel	1.3s	2.1s	2.0s	4.4s
ACES	4.5s	6.5s	5.0s	6.5s
MIT-Killian	3.2s	210.6s	869.0s	1883.4s

Table 4.11 The real-time performance of Graph SLAM on laptop Lenovo T540p

Dataset name	t_{rt}			
	x1	x2	x3	x4
Intel	0.3s	0.3s	0.2s	0.5s
ACES	0.8s	0.8s	0.9s	0.9s
MIT-Killian	0.5s	0.4s	0.5s	0.5s

The results are shown in Table 4.10 and 4.11. For the laptop, all the real-time performance t_{rt} are below 1 second. It means that when the last scan observation is acquired, the full solution can be delivered in 1 second which includes the preprocessing time, the scan matching time, the validation time and the optimization time. For running on the embedded computation unit at the original publish speed, even for the large area dataset MIT Killian Court, the real-time performance t_{rt} can be below 5 seconds. For small datasets like Intel Research Lab and ACES Building, the time needed at higher publish rates only increase slightly. For MIT Killian Court, the time needed at higher publish rates increases significantly due to the large number of nodes in the graph, and the difference in computation power between an embedded computation unit and a laptop with Intel 4th generation i7 is also clearly demonstrated.

4.8 Summary

Graph optimization is a powerful way of solving the SLAM problem by taking all the information into consideration. The graph optimization based SLAM has two stages which are graph building (front-end) and graph optimization (back-end). Efficiently optimizing the graph is already intensively researched in the robotic community with a lot of success. The building of the graph is not efficient and robust enough to be real-time which makes it the bottleneck of the graph optimization based SLAMs. The automatic real-time robust graph building is a difficult problem, because of the massive number of possible data associations and perceptual similarities in the environments.

In this chapter, an integrated graph optimization based SLAM is presented with the capability of robustly building and optimizing the graph in real-time. The current solver (back-end) used here is iSAM, but it can also be changed to other stage-of-the-art solvers, for example G2O, SPA, or Lago and so on. The new proposed Graph SLAM exploits the inherent relationship of sensor observations with the environment, and further monitors the stage of robot for a better computation task arrangement. The parallel design and implementation of the proposed SLAM further boosts the performance of the algorithm by benefiting from the multi-core CPU architecture of current computation hardwares.

What is new in this chapter is the framework for graph building including basic edges and loop closure edges. The well designed basic edge building method ensures the low error of basic edges even under difficult perceptual similarity situations which benefits the loop closure edge building by limiting the searching area for loop closures. The boosting based cascade loop closure preprocessing method uses the ellipsoid representation of the laser scan together with the robot pose for fast rejection of unrelated nodes and uses a gradient descent based intersection area size check algorithm for accurately identifying potential data associations. The validation criteria of the loop closure hypothesis could robustly distinguish true loop closures from false loop closures. With the new proposed method, big loop loops can also be efficiently closed which makes it also suitable for large scale environments.

The algorithm is very fast, and the real-time performance of the proposed SLAM is proved even on embedded devices.

Chapter 5

Applications of the robust and adaptive SLAM in real environments

5.1 Project Othello

5.1.1 Introduction

Othello is the abbreviation of *Autonomous Object Taxi and Handler for Handicapped and Elderly persons*. In the project Othello, the research, development and validation of components of a mobile handling aid in the home-care robotics area haven been done. This mobile handling aid has automated driving functions. For the increasing aging population who wants to live independently in their home environment, this will become more and more of a factor due to the growing lack of care workers problem. The system developed in this project is designed to help people with restricted mobility restraints and disability in coping with everyday tasks such as to help with food and drink without a caregiver being permanently present. The special adaption of apartment for the disabled person should also be avoided or delayed. The system Othello should be able to handle items, grab and transport. For example. the robot can transport a bottle of water from the kitchen to the living area.
To fulfill this vision of a self-determined life up to a high age, the Fraunhofer IOSB-AST works on a new 3D sensor based indoor localization system and navigation methods. The project partners Focal Meditech BV and Götting KG take care of the development of the robot platform including tools for grabbing and carrying. In this joint project granted by the German ministry of education and science, a robot shall be developed, to assist elderly and handicapped people to deal with their daily life. The main tasks of this household robot are fetch and bring services. To fulfill this tasks a robot arm will be integrated. Also it should drive to the charging station autonomously.

The challenge in this project is the field of domestic application, because this is a complex working area for a robot. Therefore the Fraunhofer IOSB-AST develops an intelligent control system. This needs to be robust and designed for the "near human use". Furthermore, it has to work nearly without human interventions. The robot shall recognize its environment autonomously and detect and handle changes in its surroundings accordingly. This information is necessary for the system to "find its way" through an apartment or to avoid obstacles. Furthermore, the interaction with the user is important to improve acceptance and usability and is treated as a principal point in the project OTHELLO. For this task, guided and autonomous learning skills are compiled for the assistance robot.
For the secure use in a home, a comprehensive perception of the environment is necessary. For this reason 3D cameras are used, and for those an appropriate data processing to evaluate the 3D data is developed. In particular a data registration for multiple cameras into one map has to be realized, which is used for the path planning and reactive navigation. For the reactive navigation "potential-field" approaches shall be refined for the special characteristic of the OTHELLO platform. The aim is to react on planning insecurities due to moving obstacles or sensor noise of the 3D cameras in a real-time capable control loop. For the intuitive handling of the OTHELLO assistance robot a gesture based controlling will be developed, which adapts itself to the user. Last but not least, OTHELLO should be a cost-effective and energy-saving mobile assistance system. For this reason small, embedded components are used for realization of the intelligent control.

(a) The Othello robot at an early stage of development.

(b) The final Othello robot with a robot arm.

Fig. 5.1 The robot developed for Project Othello.

5.1.2 The author's work in the project

The author's work in the project is mainly the mapping and localization part. The goal here is to reduce the system cost by using cheap and energy efficiency components. Different SLAM algorithms have been tested and further developed. Different kinds of sensors are tested. In the end, the cheap ARM-based computation units which are also widely used in the communication devices (smart phones, tablets and so on) are used to replace the normal computers. To replace the expensive laser scanners, the use of cheap 3D cameras (like Microsoft Kinect) with significant reduced Field-of-View (FOV) is explored. A more detailed experiment is examined by using configurations of different scan matching algorithms and different sensors in Chapter 3. The experiment shows that one Kinect is generally not enough for the mapping of unknown home-like environments, so an addition Kinect is used to extend the total FOV. For generating a 2D occupancy grid map, the 3D point cloud from the Kinect is collapsed into a 2D laser scan.

The map generated in the Othello house The project partner Focal Meditech BV has also created a home-like demonstration environment in its premises. This has also been used for validation purposes in relation to the application scenario. A section of this is shown in Fig. 5.2. With the living room, bedroom and bathroom, it represents a complete

Fig. 5.2 Othello platform in the demonstration environment of project partner Focal Meditech BV in Tilburg NL (Othello house).

non-adapted housing unit with the typical dimensions. For this purpose, the scenario of approaching a target position for the purpose of transport or manipulation was examined. The review scenario was first in the assignment task for the selected area. Then the autonomous localization attempt was carried out in this area and finally the entire navigation chain was tested with additional path planning and path control components. All components could be successfully tested in the described test environment. It was possible in no time to create a map using the selected Rao-Blackwellized particle filter based SLAM method (EMB-SLAM). Fig. 5.3 shows the result map for navigation. This is an occupancy grid map, which describes the assignment of the individual cells with gray values. All obstacles are marked black. Gray cells describe the unexplored area. Passable cells are displayed in white. The sensors used were the three "ASTRA" 3D cameras from Fotonic (see Fig. 5.1a) together with the miniature laser scanner TIM650 from SICK. The upper camera was used exclusively for navigation.

The map generated in the Evoluon building in Eindhoven A further test of the entire system was conducted during the exhibition, which was affiliated to Robocup in Eindhoven. A map of the Evoluon, a circular building on the Philips campus, was successfully recorded when the platform navigated between visitors. Here, during the mapping, no special care had to be taken on the visitors of the fair, as they were detected by the method used as moving obstacles and thus no entry was made in the map. Due to the speed of this technology, a short command by hand started the navigation function, which ultimately makes the system quickly ready for use and flexible. Another challenge was the size and shape of the building's ground

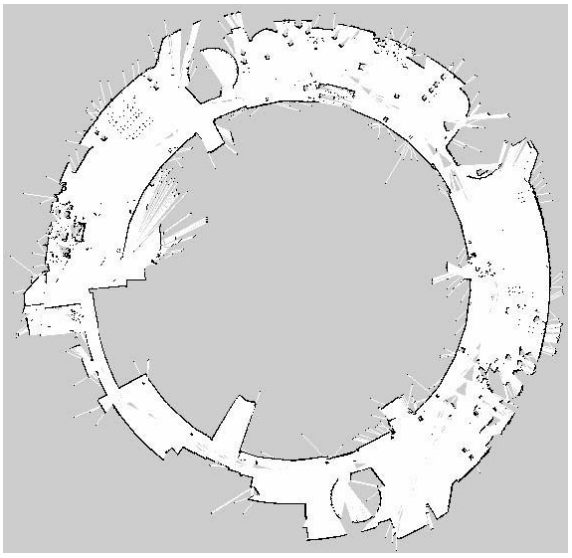
(a) The plan of the test area in project partner Focal Meditech BV.

(b) The map generated using EMB-SLAM with two Kinects.

Fig. 5.3 The test in Othello house.

floor. Fig. 5.4a shows the round building in which the map Fig. 5.4b was recorded. In this test, the benefits of the chosen structure emerged clearly. Since the mobile Othello system was not always in the immediate vicinity of the operating computer (tablet, PC), it was an advantage that the computers for path planning / navigation were integrated directly on the mobile platform. Thus, the operating computer was only responsible as an exchangeable interface between the user and the mobile Othello system. As a result, the mobile system was able to independently work off the received order even with a WiFi connection tear-down to the operating computer. In addition, this was also the first test with integrated robot arm. The robot arm was not fully functional at this time. However, the effect on controller behavior could be tested with the arm attached. Fig. 5.1b shows the driving platform with robot arm attached.

(a) The image of the Evoluon building in Eindhoven.



(b) The recorded map during the test in Evoluon Endhoven.

Fig. 5.4 The test in Evoluon Endhoven.

5.2 Project Klara

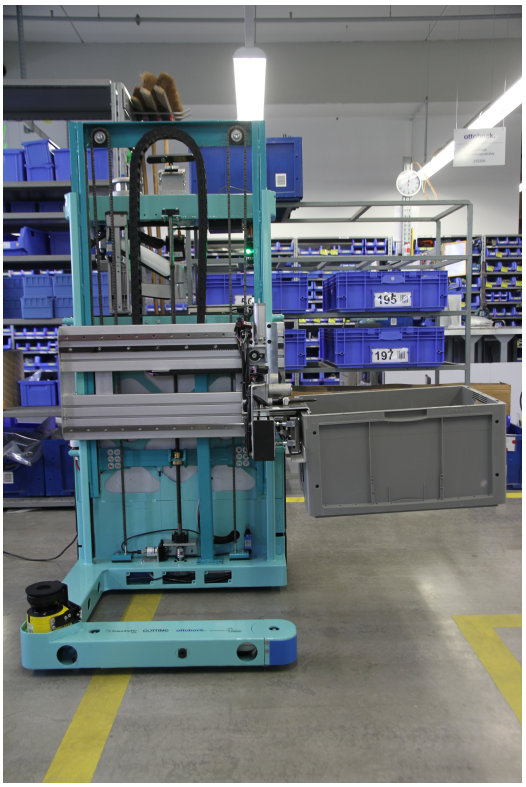
5.2.1 Introduction

Klara is the abbreviation of German project name *Unterstützungsfunktionen für kleine Transport- und Handhabungshilfen zu autonomer Mobilität und Lernfähigkeit*, in English means *Small Autonomous Moving And Handling Aid*. In many companies, manufacturing and assembly processes require lifting and carrying of loads. These areas are therefore particularly relevant for occupational health and safety. Currently, many people cannot be employed in manufacturing and assembly processes since they belong to the group of people who cannot be expected to handle heavy loads. To provide these employees with the ability to participate in this area of activity and to minimize the risk of chronic skeletal health problems, it is necessary to reduce the strain of lifting and handling heavy objects. Many aids already exist for moving heavy loads. Not so, however, for small loads up to 50 kg: existing tools are used mainly in areas with relatively rigid work-flows, which largely excludes flexible usage of these aids.

In project KLARA granted by the German ministry of education and science, a personal, versatile lifting and moving aid for loads of up to 50 kg is being developed. Rather than replacing employees in production, the aim is to provide employees with an intelligent "sidekick". To be able to perform this task, the KLARA platform must have the following

capabilities:(1) Free navigation and automated driving, (2) 3D environment detection, (3) Recognition, picking up and lifting of small loads, (4) Simple, intuitive and personalized operation for specific tasks.
The main task of Fraunhofer IOSB-AST consists of the development of tasks related to navigation, which are implemented as a system of "embedded hardware modules". A task is therefore the development of an automatic navigation system with scalable assistance and sensor-based environment detection for reliable mission guidance based on LiDAR scanners and 3D sensor technology. The navigational aspects encompass sensor data preprocessing and fusion as well as automatic mapping. With the Gesture Move & Drive system, approaches to gesture-based control of the KLARA platform will be incorporated in the project.
Positioning is being developed on the basis of the Simultaneous Localization and Mapping (SLAM) technique, which, in a learning phase, generates a sensor-based map of the environment, with which the KLARA platform can identify its location at any time in operation. The process for industry-capable use is being further developed in the KLARA project. In particular, in contrast to the state of the art, "learning" of the map is to take place largely automatically. Inexpensive 3D sensors will be used for positioning. The entire data processing chain is implemented with a distributed architecture of embedded, heterogeneous computer components. These modules are based on ARM or DSP processors and form a flexible modular network that can be easily adapted to various industrial requirements and enables a high re-usability.
Unlike standard PC hardware, the computer units are exceptionally small and energy saving, so the solution has a high usability potential from both a technology and an economic perspective. Data processing is specifically optimized for this purpose and utilizes the Robot Operating System (ROS) to achieve an open architecture that is capable of further development.

(a) The robot used in Project Klara in a factory.



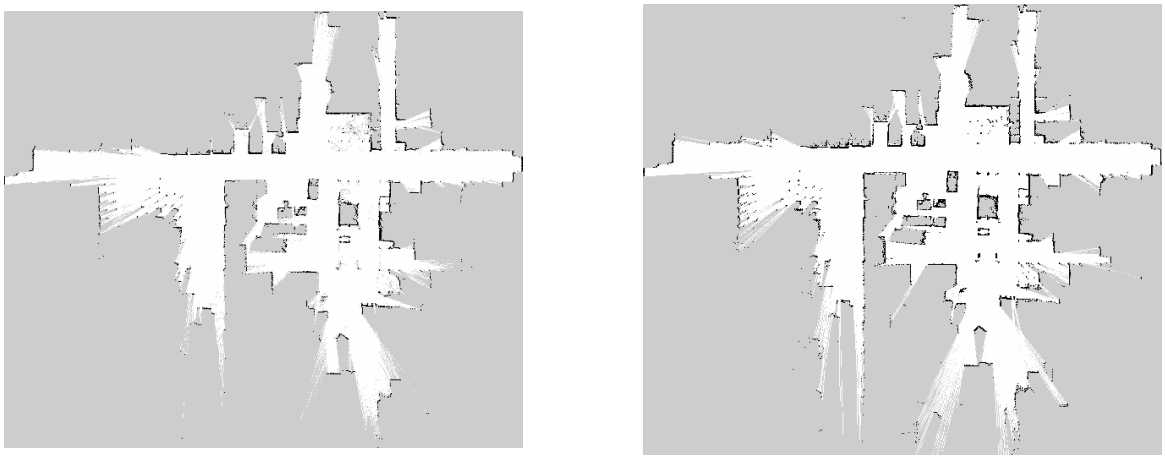
(b) The robot used in Project Klara in a factory.

Fig. 5.5 The robot used in project Klara. SICK S300 LIDAR; IFM TOF; two Fotonic ASTRA 3D; ODroid-XU3, Plug-In IPC, Speed Real-Time Target Machine.

5.2.2 The author's work in the project

Various well-known SLAM algorithms have been tested and further developed to meet the requirement of the Project. To deal with large scale environments, Graph-SLAM is developed and used here.

The Graph-SLAM is tested in a wheelchair-manufacturing factory and a car-manufacturing factory. As shown in Fig. 5.6, The author's EMB-SLAM (Bayesian filter based method) and Graph-SLAM (graph optimization based method) are tested in a small patch of the wheelchair-manufacturing factory using the same dataset. The results from EMB-SLAM and Graph-SLAM are similar and good. The platform developed in this project is shown in Fig. 5.5. The robot is able to understand human gesture commands, grabs and lifts the boxes from the shelf for transporting.



(a) The map in a factory using EMB-SLAM.

(b) The map in a factory using Graph-SLAM.

Fig. 5.6 The robot used in project Klara. SICK S300 LIDAR; IFM TOF; two Fotonic ASTRA 3D; ODroid-XU3, Plug-In IPC, Speed Real-Time Target Machine. The test area is about $23m \times 28m$.

A further comparison between the author's methods with the corresponding state-of-the-art methods in a large-scale car-manufacturing factory environment (about $22,000m^2$) is shown in Fig. 5.7 and Fig. 5.8. A laptop with a 4th generation Intel i7 running at 2.4GHz is used for the experiment. EMB-SLAM and GMapping both failed to represent the real model of the factory, due to the inherent limitation of particle filter based methods, with the increasing of the uncertainty of the poses, more particles are required to correctly approximate the posterior. If we increase the number of particles, the result may be better, but as every particle carries a path and its own map (occupancy grid maps of large environments are pretty large in memory consumption), the total memory consumption will be too large for the embedded devices. Different from Bayesian based method, the author's Graph-SLAM (only one occupancy grid

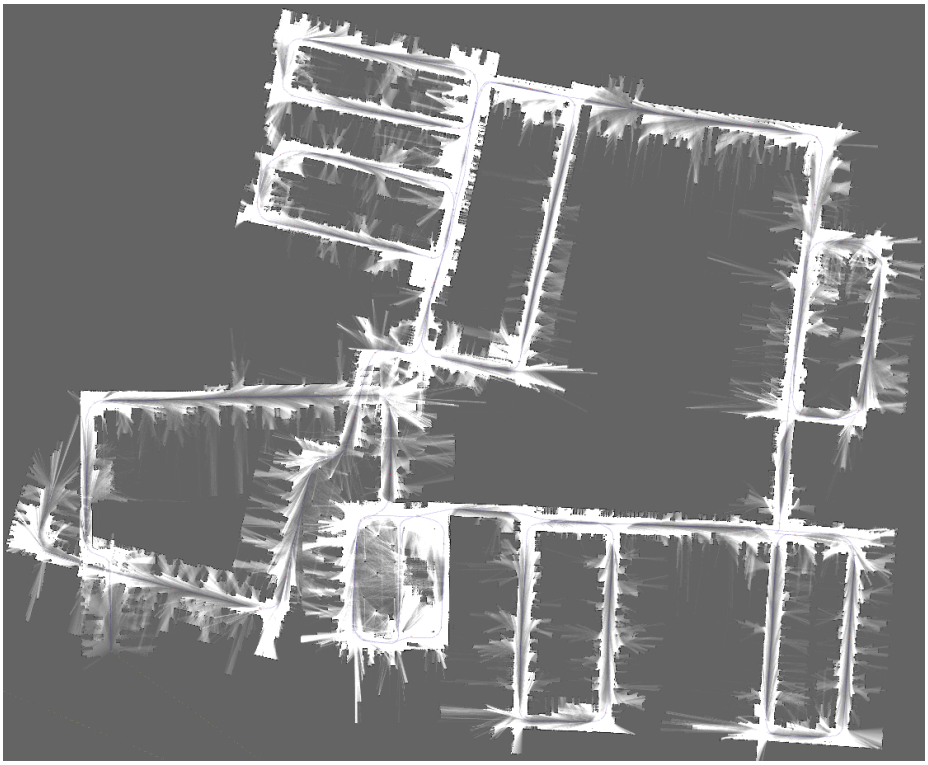
is needed) successfully recovered the model of the factory with real-time performance as shown in Fig. 5.8a, and the Google Cartographer failed to close the last two big loops and resulted as an inconsistent map. The test in the real complex large-scale factory environment proves that the author's method is better than the state-of-the-arts in processing speed and robustness.

(a) The map in a factory using EMB-SLAM (N=60).

(b) The map in a factory using GMapping (N=60).

Fig. 5.7 The robot is equipped with a Hokuyo UTM-30LX Scanning Laser Rangefinder. The test area is about 22,000 m^2, $130m \times 165m$.

(a) The map generated using Graph-SLAM at 4 times of the real speed.



(b) The map generated using Google Cartographer at the real speed.

Fig. 5.8 The robot is equipped with a Hokuyo UTM-30LX Scanning Laser Rangefinder. The test area is a factory about 22,000 m^2, $130m \times 165m$.

Chapter 6

Summary

6.1 Conclusion

The navigation problem which is a main prerequisite for autonomous mobile robots to fulfill high-level tasks such as handling and manipulation is often identified as one of the key challenges in mobile robotics. In the field of navigation, there are several components, for example, mapping and localization, path planning, control, obstacle avoidance and so on. In this work, the author contributes mainly to the improvement of the mapping and localization part of the navigation system.

A key component of laser scanner based SLAM algorithms, the scan matching algorithm, is explored. Different scan matching algorithms are systemically experimented with different kinds of simulated laser scanners for indoor home-like environments for the first time. The influence of properties of laser scanners (maximal range, field-of-view) in scan matching algorithms is quantitatively analyzed. The difference between scan matching algorithms in terms of robustness, accuracy and computation time is given. Five well-known scan matching algorithms, namely Classic ICP, MbICP, CSM, PSM and NDT, are compared with different laser scanners. Results show that the field-of-view of laser scanners plays an important role in the robustness of scan matching algorithms. For robust scan matching results, with odometry input, a laser scanner requires at least 90° field-of-view; without odometry input, at least 180° field-of-view is required. Even though the odometry input is normally noisy, the computation time of scan matching algorithms is reduced by the usage of odometry as the initial guess. When the odometry input is available, CSM and MbICP are good choices for range finders with limited field-of-view and short maximal range. When there is no odometry input, CSM can still keep good performance with the cost of high computation time, but the performance of MbICP drops significantly, so MbICP is not recommended when odometry is not available. ICP performs almost as good as CSM, but its processing time is stable regardless of odometry

availability. For NDT, a wide field-of-view is required. One interesting property of NDT is its independence from odometry, which may be interesting for odometry-free robot systems. In general, PSM does not perform well compared to the other methods.
A new efficient implementation of Rao-Blackwellized Particle Filter based SLAM is presented. It is based on the usage of a pre-computed lookup table and the parallelization of the particle updating. Different task-based parallelizing models are compared for the parallelization task. The computational expensive part of the scan matching algorithm 'vasco' for the iterative searching of closest correspondences in GMapping is accelerated by using a pre-computed lookup table (a time-memory tradeoff), and a mean speedup of 9.9 times is achieved on ODROID-XU4. The inherent property of particle filters that each particle carries a map and its own path enables the parallel implementation of RBPF-SLAM for current multi-core architecture computation systems. The comparison of three parallel-programming models shows that the Intel TBB and Boost Thread perform better than the previous approach using OpenMP. Compared to the state-of-the-art method GMapping, EMB-SLAM has a speedup of 12 times running on recent multi-core embedded systems which fulfill low cost and energy efficiency requirements. The speedup enables the on-line usage of the algorithm on high speed mobile robotics with low-cost computation units.
A novel real-time graph building method and a full integrated robust Graph SLAM solution are presented to solve the problem of current graph optimization based SLAMs with the difficulty of real-time graph building in large-scale environments on low-cost computation devices. The proposed solution outperforms the state-of-the-art algorithms in processing time and robustness especially in large-scale environments using embedded systems instead of high-end computation systems. The improvements include enhancements of Olson's brutal force scan matching algorithm, the definition of unique direction norms for scan observations, efficient loop closure detection algorithms, and a parallel and adaptive implementation of the proposed solution.
The key points are:

- Olson's brutal force scan matching algorithm is enhanced by using a distance weight to compensate the uneven distribution of points in scan observations that has high density at the near region and low density at the far region. A new environment-aware enhancement is proposed to actively detect corridor-like featureless cases and identify matched points with opposite norms, and as a result, the scan matching converges to the correct solution instead of the solution with the highest score at those difficult cases.

- A new long-term + short-term sliding window based basic edge builder is proposed to robustly construct basic edges. The long-term sliding window limits the accumulated error in

the local area, and the short-term sliding window prevents the influence of spurious odometry measurements (for example slippage).

• The new ellipsoid representation of scan observations together with the error propagation from scan matching poses enables faster and more effective selection of loop closure candidates than before. The ellipsoid based cascade method is slower than the state-of-the-art preprocessing methods, but compared with the sum time of datasets, the preprocessing time is negligible (less than 1% of the sum time of datasets). The selected loop closure candidates will be clustered into different groups based on their topological distances, and candidates in each group will be sorted based on the size of their intersection areas for efficient loop closure searching.

• The local map to local map based matching is used for finding the transformations of loop closure edges. A dynamic local map generation algorithm is developed for maximizing the overlapping area between two nodes forming the loop closure candidate. A VoxelGrid filter based down-sampling method is applied to the target and source local maps to accelerate the computation speed at the same time keeping the important points.

• The unique direction norms of segments extracted from scan observations are defined by taking the sensor viewpoint into consideration. Difficult false loop closure cases like two different sides of a wall merged as one line can be recognized correctly. Using the segment norm check together with other validation criteria like scan matching score, correlative rate, complex rate and entropy check in the local graph optimization, false loop closures are correctly rejected.

• The parallel design and implementation of sensor message acquiring, basic edge building, loop closure edge building and graph optimization tasks and the monitoring of the idle state of the robot further boost the efficiency of the system.

In summary, an adaptive navigation system for indoor robotics with zero infrastructure is developed, which is efficient and able to run in real-time on low-cost and energy-efficient embedded devices for large-scale areas.

6.2 Outlook

The methods presented in this thesis enable robots to operate in larger areas efficiently and robustly on embedded systems using mainly range-only sensors (laser scanners) than before. Range-only senors could provide distance measurements with centimeter accuracy

which is much better than humans estimation without aids, but the information acquired about the environment is limited, for example, the sensors see only whether it is free or occupied, in contrast, the bearing-based sensors, for example, cameras can provide much more information about the environment. With the current developments in machine learning, the objects in the image can be detected as shown in Fig. 6.1. If we can fuse the detected objects with the occupancy-grid map, the generated map will be more friendly for the humans and hence the robot can assist the human better. The localization task can also be improved by using information from the image. For example in a home-like environment, with the information about the existing of a television (TV), the kidnapped robot can limit its search area mainly in the living room instead of spreading assumptions over the whole environment. A simulated home environment is shown in Fig. 6.2a. A sensor with the ability to detect objects like tables, chairs, sofas and so son is simulated, and a map of the labeled objects is built using EKF as shown in Fig. 6.2b. The estimated poses fit well with the ground truth poses. In the simulated environment, the center of the object is simplified, but in real world, the centers of objects are more divisible and difficult to determine. It is important for the further development of the SLAM problem by incorporating achievements from other scientific branches.

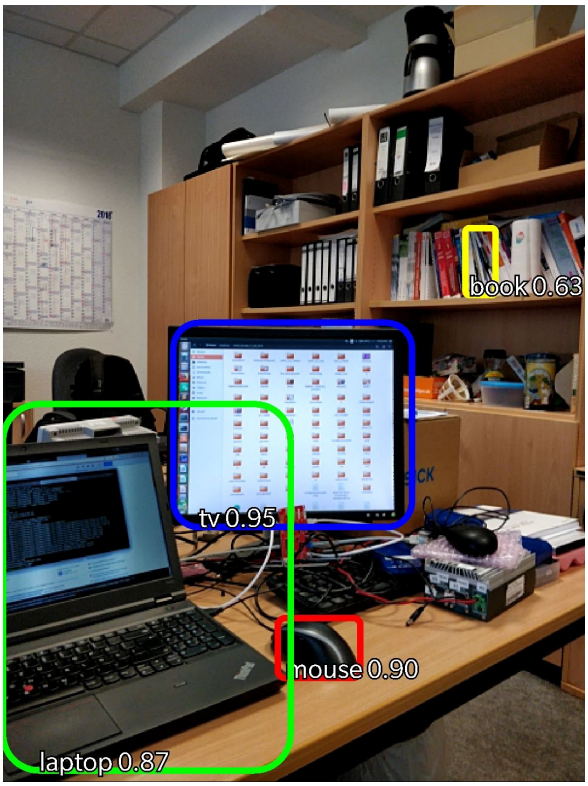
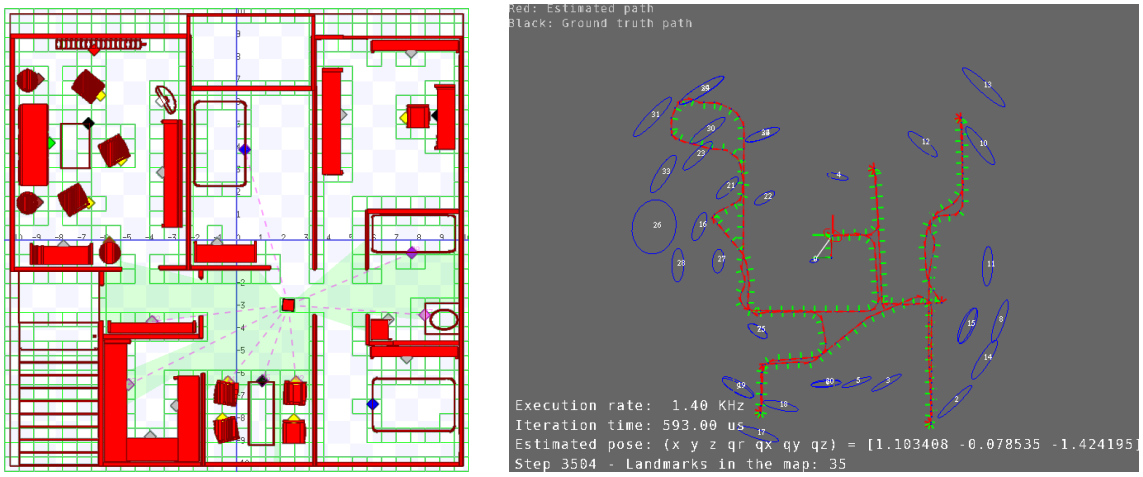


Fig. 6.1 An example of objects detected from an image by a trained model.



(a) The simulated sensor with the ability to detect objects in a simulated environment using Player/Stage.

(b) A map of objects from the simulated environment is built by EKF SLAM. The blue ellipsoids represent the detected objects with uncertainty. The estimated path in red fits with the ground truth path in black.

Fig. 6.2 The simulated semantic labeling based SLAM.

References

[1] (2017). Boost Software. http://www.boost.org/. [Online; accessed 01-January-2017].

[2] Agarwal, P., Tipaldi, G. D., Spinello, L., Stachniss, C., and Burgard, W. (2013). Robust map optimization using dynamic covariance scaling. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 62–69.

[3] Agarwal, S., Mierle, K., and Others (2010). Ceres solver. http://ceres-solver.org.

[4] Andreasson, H., Magnusson, M., and Lilienthal, A. (2007). Has somethong changed here? autonomous difference detection for security patrol robots. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3429–3435.

[5] Angeli, A., Doncieux, S., Meyer, J. A., and Filliat, D. (2008). Real-time visual loop-closure detection. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1842–1847.

[6] Arras, K. and Arras, K. (1998). An Introduction to Error Propagation: Derivation, Meaning, and Examples of Equation cy= fx cx fx. *Lausanne: Swiss Federal Institute of Technology Lausanne (EPFL)*, (September):98–01.

[7] Arulampalam, M., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-Gaussian\nBayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188.

[8] Barla, A., Odone, F., and Verri, A. (2003). Histogram Intersection kernel for image classification. In *Proceedings 2003 International Conference on Image Processing*, volume 3, pages 513–516.

[9] Beinschob, P. and Reinke, C. (2015). Graph SLAM based mapping for AGV localization in large-scale warehouses. *Proceedings - 2015 IEEE 11th International Conference on Intelligent Computer Communication and Processing, ICCP 2015*, pages 245–248.

[10] Bergman, N. (1999). Recursive bayesian estimation: Navigation and tracking applications. Technical report.

[11] Besl, P. J. and McKay, N. D. (1992). A Method for Registration of 3-D Shapes.

[12] Biber, P. and Strasser, W. (2003). The normal distributions transform: a new approach to laser scan matching. *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, 3(October):2743 – 2748 vol.3.

[13] Bibuli, M., Caccia, M., and Lapierre, L. (2011). A Method for Fast Encoder-Free Mapping in Unstructured Environments. *Journal of Field Robotics*, 28(6):817–831.

[14] Board, T. O. A. R. (2018). OpenMP. https://www.openmp.org/. Last accessed 2018-05.

[15] Boost (2018). Boost C++ Libraries. http://www.boost.org/. Last accessed 2018-05.

[16] Bosse, M. and Roberts, J. (2007). Histogram matching and global initialization for laser-only SLAM in large unstructured environments. *Proceedings - IEEE International Conference on Robotics and Automation*, (April):4820–4826.

[17] Burgard, W., Cremers, A., and Fox, D. (1998). The interactive museum tour-guide robot. *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial AAAI '98/IAAI '98 intelligence*, pages 11–18.

[18] Burgard, W., Stachniss, C., Grisetti, G., Steder, B., Kümmerle, R., Dornhege, C., Ruhnke, M., Kleiner, A., and Tardös, J. D. (2009). A comparison of slam algorithms based on a graph of relations. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2089–2095.

[19] Censi, A. (2008). An ICP variant using a point-to-line metric. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 19–25.

[20] Censi, A., Iocchi, L., and Grisetti, G. (2005). Scan matching in the hough domain. *Proceedings - IEEE International Conference on Robotics and Automation*, 2005:2739–2744.

[21] Chang, C.-W., Chen, K.-T., Lin, H.-L., Wang, C.-K., and Jean, J.-H. (2007). Development of a patrol robot for home security with network assisted interactions. In *SICE Annual Conference 2007*, pages 924–928.

[22] Chetverikov, D., Svirko, D., Stepanov, D., and Krsek, P. (2002). The Trimmed Iterative Closest Point algorithm. *Object recognition supported by user interaction for service robots*, 3:545–548.

[23] CIA (2017). Cia world factbook. https://www.cia.gov/library/publications/the-world-factbook/fields/2010.html. [Online; accessed 20-January-2018].

[24] Collier, J., Se, S., Kotamraju, V., and Jasiobedzki, P. (2012). Real-time lidar-based place recognition using distinctive shape descriptors. *Proceedings of SPIE - The International Society for Optical Engineering*, 8387(November 2016):83870P–83870P–11.

[25] Cooperation, I. (2018). Intel® Threading Building Blocks. https://www.threadingbuildingblocks.org/. Last accessed 2018-05.

[26] Corke, P. I. (2017). *Robotics, vision and control: fundamental algorithms in MATLAB®*. Springer.

[27] Corso, N. and Zakhor, A. (2013). *Loop Closure Transformation Estimation and Verification Using 2D LiDAR Scanners*. PhD thesis.

[28] De La Puente, P. and Rodriguez-Losada, D. (2014). Feature based graph-SLAM in structured environments. *Autonomous Robots*, 37(3):243–260.

[29] De La Puente, P. and Rodriguez-Losada, D. (2015). Feature based graph SLAM with high level representation using rectangles. *Robotics and Autonomous Systems*, 63(P1):80–88.

[30] Deak, G., Curran, K., and Condell, J. (2012). A survey of active and passive indoor localisation systems. *Computer Communications*, 35(16):1939–1954.

[31] Diosi, A. and Kleeman, L. (2007). Fast Laser Scan Matching using Polar Coordinates. *The International Journal of Robotics Research*, 26(10):1125–1153.

[32] Douc, R. and Cappe, O. (2005). Comparison of resampling schemes for particle filtering. *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.*

[33] Doucet, A., Freitas, N. d., Murphy, K. P., and Russell, S. J. (2000). Rao-blackwellised particle filtering for dynamic bayesian networks. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, UAI '00, pages 176–183, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[34] Engelhard, N., Endres, F., Burgard, W., and Cremers, D. (2012). A Benchmark for the Evaluation of RGB-D SLAM Systems. *IEEE IROS*.

[35] Evans, J. M. (1994). Helpmate: an autonomous mobile robot courier for hospitals. In *Intelligent Robots and Systems '94. 'Advanced Robotic Systems and the Real World', IROS '94. Proceedings of the IEEE/RSJ/GI International Conference on*, volume 3, pages 1695–1700 vol.3.

[36] Frese, U. (2008). DLR Spatial Cognition Data Set. http://www.informatik.uni-bremen.de/agebv/en/DlrSpatialCognitionDataSet.

[37] Gil, A., Juliá, M., and Reinoso, Ó. (2015). Occupancy grid based graph-SLAM using the distance transform, SURF features and SGD. *Engineering Applications of Artificial Intelligence*, 40:1–10.

[38] GmbH, S. (2017). Symeo GmbH. http://www.symeo.de.

[39] Gouveia, B. D., Portugal, D., and Marques, L. (2014). Speeding up rao-blackwellized particle filter SLAM with a multithreaded architecture. *IEEE International Conference on Intelligent Robots and Systems*, (Iros):1583–1588.

[40] Granström, K., Callmer, J., Ramos, F., and Nieto, J. (2009). Learning to detect loop closure from range data. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 15–22.

[41] Grisetti, G., Kummerle, R., Stachniss, C., and Burgard, W. (2010). A tutorial on graph-based SLAM. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43.

[42] Grisetti, G., Stachniss, C., and Burgard, W. (2007). Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46.

[43] Gutmann, J.-s. (1996). AMOS : Comparison of Scan Matching Approaches for Self-Localization in Indoor Environments.

[44] Gutmann, J.-s., Freiburg, D., Konolige, K., and Park, M. (1999). Incremental Mapping of Large Cyclic Environments. *Proc. of the IEEE Int. Symposium on Computational Intelligence in Robotics and Automation (CIRA)*.

[45] Hess, W., Kohler, D., Rapp, H., and Andor, D. (2016). Real-time loop closure in 2D LIDAR SLAM. *Proceedings - IEEE International Conference on Robotics and Automation*, 2016-June:1271–1278.

[46] Himstedt, M., Frost, J., Hellbach, S., Bohme, H. J., and Maehle, E. (2014). Large scale place recognition in 2D LIDAR scans using Geometrical Landmark Relations. *IEEE International Conference on Intelligent Robots and Systems*, (Iros):5030–5035.

[47] Himstedt, M., Keil, S., Hellbach, S., and Böhme, H. (2013). A robust graph-based framework for building precise maps from laser range scans. . . . *on Robust and Multimodal Inference in* . . . .

[48] Huang, S. and Dissanayake, G. (2016). A critique of current developments in simultaneous localization and mapping. *International Journal of Advanced Robotic Systems*, 13(5):1–13.

[49] Ila, V., Polok, L., Solony, M., and Svoboda, P. (2016). Highly Efficient Compact Pose SLAM with SLAM++ 1. (2010):1–21.

[50] Jelínek, L. (2016). *Graph-based SLAM on Normal Distributions Transform Occupancy Map*. PhD thesis, Charles University.

[51] Jennings, J. S., Whelan, G., and Evans, W. F. (1997). Cooperative search and rescue with a team of mobile robots. In *Advanced Robotics, 1997. ICAR '97. Proceedings., 8th International Conference on*, pages 193–200.

[52] Julier, S. and Uhlmann, J. K. (2004). Unscented filtering and nonlinear estimation. *Proc. IEEE*, 92(3):401–422.

[53] Junqin, L., Zhihong, C., Yanbo, W., Jiayu, L., and Yu, L. (2017). Research on loop-closure detection in SLAM based on RGB-D camera. In *2017 Chinese Automation Congress (CAC)*, number 1.

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

[55] Kallasi, F., Rizzini, D. L., and Caselli, S. (2016). Fast Keypoint Features from Laser Scanner for Robot Localization and Mapping. *IEEE Robotics and Automation Letters*, 1(1):176–183.

[56] Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35.

[57] Kass, R. E. (1998). Markov chain monte carlo in practice: A roundtable discussion. *The American Statistician*, 52(2):93–100.

[58] Khoury, H. M. and Kamat, V. R. (2009). Evaluation of position tracking technologies for user localization in indoor construction environments. *Automation in Construction*, 18(4):444–457.

[59] Kohlbrecher, S., Von Stryk, O., Meyer, J., and Klingauf, U. (2011). A flexible and scalable SLAM system with full 3D motion estimation. *9th IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR 2011*, pages 155–160.

[60] Kümmerle, R., Steder, B., Dornhege, C., Ruhnke, M., Grisetti, G., Stachniss, C., and Kleiner, A. (2009). On measuring the accuracy of SLAM algorithms. *Autonomous Robots*, 27(4):387–407.

[61] Labbé, M. and Michaud, F. (2014). Online global loop closure detection for large-scale multi-session graph-based SLAM. *IEEE International Conference on Intelligent Robots and Systems*, (Iros):2661–2666.

[62] Latif, Y., Cadena, C., and Neira, J. (2013). Robust loop closing over time for pose graph SLAM. *International Journal of Robotics Research*, 32(14):1611–1626.

[63] Latif, Y., Cadena, Ć., and Neira, J. (2014). Robust graph SLAM back-ends: A comparative analysis. *IEEE International Conference on Intelligent Robots and Systems*, (Iros):2683–2690.

[64] Lee, G. H., Fraundorfer, F., and Pollefeys, M. (2013). Robust pose-graph loop-closures with expectation-maximization. *IEEE International Conference on Intelligent Robots and Systems*, pages 556–563.

[65] Lee, H.-C., Lee, S.-H., Lee, S.-H., Lee, T.-S., Kim, D.-J., Park, K.-S., Lee, K.-W., and Lee, B.-H. (2011). Comparison and analysis of scan matching techniques for Cooperative-SLAM. *2011 8th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 165–168.

[66] Lee, S. and Lee, S. (2013). Embedded visual SLAM: Applications for low-cost consumer robots. *IEEE Robotics and Automation Magazine*, 20:83–95.

[67] Li, B., Gallagher, T., Dempster, A. G., and Rizos, C. (2012). How feasible is the use of magnetic field alone for indoor positioning? *2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, (November):1–9.

[68] Liu, H., Darabi, H., Banerjee, P., and Liu, J. (2007). Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 37(6):1067–1080.

[69] Liu, J. S. and Chen, R. (1998). Sequential monte carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044.

[70] Llofriu, M., Andrade, F., Benavides, F., Weitzenfeld, A., and Tejera, G. (2013). An Embedded Particle Filter SLAM implementation using an affordable platform. In *Advanced Robotics (ICAR), 2013 16th International Conference*, pages 2–7.

[71] Lluis, P., Carrasco, N., Bonin-font, F., and Codina, G. O. (2016). Stereo Graph-SLAM for Autonomous Underwater Vehicles. 302:351–360.

[72] Lu, F. and Milios, E. (1997). Globally Consistent Range Scan Alignment for Environment Mapping. *Autonomous Robots*, 4(4):333–349.

[73] Mautz, R. and Tilch, S. (2011). Optical indoor positioning systems. *Ieee*, (September):21–23.

[74] McElhoe, B. A. (1966). An assessment of the navigation and course corrections for a manned flyby of mars or venus. *IEEE Transactions on Aerospace and Electronic Systems*, AES-2(4):613–623.

[75] Minguez, J., Montesano, L., and Lamiraux, F. (2006). Metric-based iterative closest point scan matching for sensor displacement estimation. 22(5):1047–1054.

[76] Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. (2002). FastSLAM: A factored solution to the simultaneous localization and mapping problem. *Proc. of 8th National Conference on Artificial Intelligence/14th Conference on Innovative Applications of Artificial Intelligence*, 68(2):593–598.

[77] Moshtagh, N. (2009). Minimum volume enclosing ellipsoids. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.116.7691&rep=rep1&type=pdf.

[78] Mur-Artal, R., Montiel, J. M., and Tardos, J. D. (2015). ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, 31(5):1147–1163.

[79] Mur-Artal, R. and Tardos, 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.

[80] Nguyen, V., Martinelli, A., Tomatis, N., and Siegwart, R. (2005). A Comparison of Line Extraction Algorithms using 2D Laser Range nder for Indoor Mobile Robotics. *Autonomous Systems Laboratory Ecole Polytechnique F´ erale de Lausanne*.

[81] Nüchter, A., Lingemann, K., Hertzberg, J., and Surmann, H. (2005). 6D SLAM with approximate data association. *2005 International Conference on Advanced Robotics, ICAR '05, Proceedings*, 2005:242–249.

[82] Olson, E. B. (2008). Robust and Efficient Robotic Mapping. *Work*, 31:265–272.

[83] Olson, E. B. (2009). Real-time correlative scan matching. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4387–4393.

[84] Pfingsthorn, M. and Birk, A. (2016). Generalized graph SLAM: Solving local and global ambiguities through multimodal and hyperedge constraints. *The International Journal of Robotics Research*, 35(6):601–630.

[85] Phillips, J. M., Liu, R., and Tomasi, C. (2007). Outlier robust ICP for minimizing fractional RMSD. *3DIM 2007 - Proceedings 6th International Conference on 3-D Digital Imaging and Modeling*, (3dim):427–434.

[86] Rainer, K., Grisetti, G., and Konolige, K. (2011). g 2 o : A General Framework for Graph Optimization. In *ICRA*.

[87] Ratter, A., Sammut, C., and McGill, M. (2013). GPU accelerated graph SLAM and occupancy voxel based ICP for encoder-free mobile robots. *IEEE International Conference on Intelligent Robots and Systems*, pages 540–547.

[88] Schilling, K., Arteche, M. M., Garbajosa, J., and Mayerhofer, R. (1997). Design of flexible autonomous transport robots for industrial production. In *Industrial Electronics, 1997. ISIE '97., Proceedings of the IEEE International Symposium on*, pages 791–796 vol.3.

[89] Schmuck, P., Scherer, S. A., and Zell, A. (2016). Hybrid Metric-Topological 3D Occupancy Grid Maps for Large-scale Mapping. *IFAC-PapersOnLine*, 49(15):230–235.

[90] Steggles, P. and Gschwind, S. (2005). The Ubisense smart space platform. *Adjunct Proceedings of the Third . . .*, pages 73–76.

[91] Sunderhauf, N. and Protzel, P. (2012). Switchable constraints for robust pose graph SLAM. *IEEE International Conference on Intelligent Robots and Systems*, pages 1879–1884.

[92] Sunderhauf, N. and Protzel, P. (2013). Switchable constraints vs. max-mixture models vs. RRR - A comparison of three approaches to robust pose graph SLAM. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 5198–5203.

[93] Tard, J. D. (2014). Fast Relocalisation and Loop Closing in Keyframe-Based SLAM. In *IEEE International Conference on Robotics & Automation (ICRA)*.

[94] Thrun, S., Bennewitz, M., Burgard, W., Cremers, A., Dellaert, F., Fox, D., Hahnel, D., Rosenberg, C., Roy, N., Schulte, J., and Schulz, D. (1999). MINERVA: a second-generation museum tour-guide robot. *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, 3(May):1999–2005.

[95] Thrun, S., Burgard, W., and Fox, D. (2005a). *Probabilisic Robotics*.

[96] Thrun, S., Burgard, W., and Fox, D. (2005b). *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press.

[97] Thrun, S. and Montemerlo, M. (2006). The GraphSLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures. *Robotics Research, The International Journal of*, 25(5-6):403–429.

[98] Tipaldi, G. D., Braun, M., and Arras, K. O. (2014). FLIRT: Interest regions for 2D range data with applications to robot navigation. *Springer Tracts in Advanced Robotics*, 79:695–710.

[99] Ullrich, G. (2015). *Automated Guided Vehicle Systems A Primer with Practical Applications*. Springer-Verlag Berlin Heidelberg.

[100] Vaughan, R. (2008). Massively multi-robot simulation in stage. *Swarm Intelligence*, 2(2-4):189–208.

[101] Vincke, B., Elouardi, A., and Lambert, A. (2010). Design and evaluation of an embedded system based SLAM applications. *2010 IEEE/SICE International Symposium on System Integration: SI International 2010 - The 3rd Symposium on System Integration, SII 2010, Proceedings*, pages 224–229.

[102] Vincke, B., Elouardi, A., Lambert, A., and Merigot, A. (2012). Efficient implementation of EKF-SLAM on a multi-core embedded system. *IECON Proceedings (Industrial Electronics Conference)*, pages 3049–3054.

[103] Welch, G. and Bishop, G. (2006). An Introduction to the Kalman Filter. *In Practice*, 7(1):1–16.

[104] Whelan, T., Kaess, M., Leonard, J. J., and McDonald, J. (2013). Deformation-based loop closure for large scale dense RGB-D SLAM. *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 548–555.

[105] Yin, J., Carlone, L., Rosa, S., and Bona, B. (2014). Graph-based robust localization and mapping for autonomous mobile robotic navigation. *2014 IEEE International Conference on Mechatronics and Automation, IEEE ICMA 2014*, pages 1680–1685.

[106] Zhang, S., He, B., Nian, R., Liang, Y., and Yan, T. (2013). SLAM and a novel loop closure detection for autonomous underwater vehicles. *Oceans - San Diego, 2013*, pages 1–4.

Declaration

I certify that I prepared the submitted thesis independently without undue assistance of a third party and without the use of others than the indicated aids. Data and concepts directly or indirectly taken over from other sources have been marked stating the sources.

Further persons were not involved in the content-material-related preparation of the thesis submitted. In particular, I have not used the assistance against payment offered by consultancies or placing services (doctoral consultants or other persons). I did not pay any money to persons directly or indirectly for work or services which are related to the content of the thesis submitted.

So far the thesis have not been submitted identically or similarly to an examination office in Germany or abroad.

I have been notified that any incorrectness in the submitted above mentioned declaration is assessed as attempt to deceive and, according to § 7 para. 10 of the PhD regulations, this leads to a discontinuation of the doctoral procedure.

Ilmenau, 17.10.2018 Qiucheng Li

(City, Date) *(Signature)*