Technical Report

UCAM-CL-TR-901
ISSN 1476-2986

Number 901

UNIVERSITY OF
CAMBRIDGE

Computer Laboratory

Signal maps for smartphone localisation

Chao Gao

February 2017

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

This technical report is based on a dissertation submitted August 2016 by the author for the degree of Doctor of Philosophy to the University of Cambridge, King's College.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

http://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

Signal maps for smartphone localisation

Chao Gao

Summary—Indoor positioning has been an active research area for 20 years. Systems based on dedicated infrastructure such as ultrasound or ultra-wideband (UWB) radio can provide centimetre-accuracy. But they are generally prohibitively expensive to build, deploy and maintain. Today, signal fingerprinting-based indoor positioning techniques, which use existing wireless infrastructure, are arguably the most prevalent. The fingerprinting-based positioning system matches the current signal observations (fingerprints) at a device to position it on a pre-defined fingerprint map. The map is created via some form of survey. However, a major deterrent of these systems is the initial creation and subsequent maintenance of the signal maps. The commonly used map building method is the so-called manual survey, during which a surveyor visits each point on a regular grid and measures the signal fingerprints there. This traditional method is laborious and not considered scalable. An emerging alternative to manual survey is the path survey, in which a surveyor moves continuously through the environment and signal measurements are taken by the surveying device along the path. A path survey is intuitively better than a manual survey, at least in terms of speed. But, path surveys have not been well-studied yet.

This thesis assessed the path survey quantitatively and rigorously, demonstrated that path survey can approach the manual survey in terms of accuracy if certain guidelines are followed. Automated survey systems have been proposed and a commodity smart-phone is the only survey device required. The proposed systems achieve sub-metre accuracy in recovering the survey trajectory both with and without environmental information (floor plans), and have been found to outperform the state-of-the-art in terms of robustness and scalability.

This thesis concludes that path survey can be streamlined by the proposed systems to replace the laborious manual survey. The proposed systems can promote the deployment of indoor positioning system in large-scale and complicated environments, especially in dynamic environments where frequent re-survey is needed.

Acknowledgements

I could not have completed my PhD without the help of many people. In particular, I wish to thank my supervisor Dr. Robert Harle for his time and patience, I wish to thank my parents for their love and support, and I wish to thank my friends and brothers for being with me along my PhD journey. I wish to thank everyone in the Digital Technology Group for the kind advice, fun and friendly atmosphere they provided. Finally, I wish to thank the Cambridge Commonwealth, European & International Trust and the China Scholarship Council for their financial support through the CSC Cambridge Scholarship.

Contents

1 Introduction **15**
1.1 Research questions 16
1.2 Limitations of scope 17
1.3 Thesis outline 18

2 Related work **21**
2.1 Dedicated infrastructure solutions 21
2.2 Existing infrastructure solutions 22
2.3 Infrastructure-free solutions 23
2.4 Combined/Fused solutions 24
2.5 Summary 24

3 Quantitative assessment of path survey **25**
3.1 Introduction 25
3.2 Path surveying techniques 26
3.3 A ground-truthed survey dataset 27
3.3.1 Environment and devices 27
3.3.2 Manual survey 28
3.3.3 Path survey walks 28
3.3.4 GP regressed map 29
3.4 Point comparison on regressed maps 30
3.5 Positioning comparison on GP regressed maps 32
3.5.1 Positioning method 32
3.5.2 Map directionality and space coverage of survey path 33
3.5.3 Segment survey evaluation 36
3.5.4 Sample density evaluation 36
3.6 Conclusions and further work 37

4 Automated signal survey **39**
4.1 Introduction 39
4.2 Related work 40
4.3 Magnetic mapping but not positioning 41
4.4 Magnetic sequence-based loop detection 41
4.4.1 Matching algorithm 42
4.4.2 Match validation 42
4.5 A magnetic surveying system 43
4.5.1 PDR 43
4.5.2 PDR line filter (optional) 45
4.5.3 SLAM front-end and back-end 46

4.5.4 Map generation . . . 46
4.5.5 Positioning . . . 46
4.6 Testbeds . . . 48
4.6.1 Floor plans . . . 48
4.7 Evaluation . . . 49
4.7.1 SLAM trajectory accuracy . . . 49
4.7.2 Positioning accuracy . . . 51
4.7.2.1 WGB2a . . . 51
4.7.2.2 WGB1 and WGB2 . . . 53
4.7.2.3 ENG, KX, and RUTH . . . 53
4.8 Conclusions and further work . . . 55

5 Optimised graphslam back-end 57
5.1 Introduction . . . 57
5.2 Related work . . . 58
5.3 Problem formulation . . . 58
5.4 State-of-the-art algorithms . . . 60
5.4.1 g2o . . . 60
5.4.2 SGD . . . 61
5.4.3 Toro . . . 61
5.4.4 Datasets . . . 61
5.4.5 Evaluation . . . 62
5.4.5.1 Algorithm settings . . . 62
5.4.5.2 Odometry constraints . . . 63
5.4.5.3 Metrics . . . 65
5.4.5.4 Results and analysis . . . 66
5.5 Optimised back-end design . . . 67
5.5.1 Motivation . . . 67
5.5.2 Optimisation based on tree operation and approximation . . . 68
5.5.2.1 Linear-time pose computation . . . 68
5.5.2.2 Tree-based H estimation . . . 69
5.5.2.3 Speeding up H estimation by approximation . . . 70
5.5.3 Error propagation over selected constraints only . . . 72
5.5.3.1 Structural feature of loop closures and constraints . . . 72
5.5.3.2 Constraints selection and error propagation . . . 75
5.6 Evaluation . . . 78
5.7 Conclusions . . . 80

6 A robust survey trajectory recovery system 81
6.1 Introduction . . . 81
6.2 Dedicated surveying . . . 82
6.3 Related work . . . 82
6.4 Trajectory recovery by state-of-the-art algorithms . . . 83
6.4.1 Inputs . . . 83
6.4.2 Methods . . . 83
6.4.3 Result analysis . . . 85
6.5 Motivations . . . 85
6.6 System overview . . . 89

6.7 PF1 90
6.8 Straight line filter 90
6.9 Loop closure detection and validation 91
6.10 PF2 94
6.11 Evaluation 95
6.11.1 Loop closure detection and validation 96
6.11.2 Trajectory outputs, particle clouds and room ambiguities 96
6.11.3 Signal map results 96
6.12 Conclusions 98

7 Conclusions **99**
7.1 Research contributions 99
7.2 Contributions to the research questions 100
7.2.1 Research question 1: how well can a path survey approach a manual survey? 101
7.2.2 Research question 2: how should the path survey be performed in order to provide best positioning performance? 102
7.2.3 Research question 3: how can the surveyor's path be recovered accurately and efficiently? 102
7.2.4 Research question 4: what inputs are needed for accurate trajectory recovery? 103
7.3 Limitations and future work 103
7.3.1 Overcome environmental limitation 103
7.3.2 Fuse multiple survey paths 104
7.3.3 Achieve 3D signal survey 104
7.3.4 Automatic update of signal maps 105
7.3.5 Preserve user privacy 105
7.4 Final words 106

A Datasets and evaluation results for SLAM **107**

Bibliography **127**

List of Figures

1.1 Manual survey and path survey. 16
1.2 Thesis outline. 19

2.1 Sample GP regression map built from a path survey. 23

3.1 The test area. 28
3.2 The survey points 28
3.3 Survey walk paths 29
3.4 Sample signal surveys and the resulting GP maps. 30
3.5 RSS$_{90}$ maps from point comparison of various GP maps (W1–W4) with the manual survey (MS) GP map. 31
3.6 RSS$_{90}$ distribution (CDF). 31
3.7 Corridor directionality for BLE and WiFi 33
3.8 Positioning results for directionality analysis. 34
3.9 The positioning results of W1$_{MS}$ (W1 maps evaluated against manual survey data) broken down by distance of manual test point from the W1 survey path. 35
3.10 The positioning CDF results for W1, W2 and W3 maps evaluated against fingerprints generated from W4 data. 36
3.11 Sample density evaluation. 37

4.1 Example magnetic loop closure for both scalar and vector magnetic fields. 42
4.2 An example of a false loop closure. 44
4.3 System architecture. 44
4.4 The value of the PDR line filter. 45
4.5 Sample SLAM component outputs. 47
4.6 Sample WiFi Gaussian Processes maps from survey walks. 47
4.7 Result of magnetic SLAM. 50
4.8 SLAM accuracy for WGB2a testbed. 51
4.9 Positioning errors in the WGB2a testbed. 52
4.10 The WiFi positioning CDF results for Ideal-SLAM maps created from W1 and W3 broken down by distance of test point from the survey trajectory. 52
4.11 Positioning errors in WGB1 and WGB2. 53
4.12 Result of magnetic SLAM. 54
4.13 Error CDFs for RUTH, ENG and KX, separated by distance of test points from the survey trajectory. 55

5.1 An example of constraints in a pose graph. 59
5.2 The illustration of loop closures and loop closure constraints. 59
5.3 Two sample datasets (WGB2a-1 and WGB2-3) used for evaluation of SLAM back-end algorithms. 62

5.4 g2o results when odometry constraints are discarded. 64
5.5 χ^2 errors and SO errors. 66
5.6 SLAM results on WGB2a-1 and WGB2-3. 67
5.7 Example of linear-time pose computation. 69
5.8 Example of two types of loop closures. 72
5.9 Illustration of constraints and their affected poses. 73
5.10 Illustration of solving selected constraints in the two types of loop closures respectively shown in Figure 5.8. 74
5.11 MSGD results on WGB2a-1 and WGB2-3. 78
5.12 CPU time per iteration of SGD, MSGD$^-$ and MSGD. 79
5.13 χ^2 errors for different algorithms on dataset WGB2-3. 79

6.1 Example scaling errors from SLAM without a floor plan. 81
6.2 The groundtruth of *Path-1*. 85
6.3 Example outputs of the conventional particle filter plus smoother approach. 86
6.4 Example outputs of the conventional particle filter plus smoother approach on longer walks. 87
6.5 Relationship between execution time and number of particles. 88
6.6 The work flow of the proposed trajectory recovery system. 89
6.7 Examples of *PF1-traj*, segment pair and maximum segment pair. 91
6.8 Loop closure detection/validation examples for two maximum segment pairs. 92
6.9 Illustration for MSP finding algorithm. 93
6.10 Loop closure validation. 95
6.11 The results of the proposed system on the path *Path-1*, *Path-2* and *Path-3*. 97
6.12 The error CDFs when applying a FL smoother and our proposed system on *Path-1* data. 97
6.13 The estimated trajectory of *Path-3* in the table area. 97
6.14 Sample signal map and final positioning result 98

A.1 Dataset of WGB2a. 108
A.2 Dataset of WGB1. 109
A.3 Dataset of WGB2. 110
A.4 Dataset of ENG, RUTH and KX. 111
A.5 SLAM results on WGB2a. 112
A.6 SLAM results on WGB1. 113
A.7 SLAM results on WGB2. 114
A.8 SLAM results on ENG, RUTH and KX. 115
A.9 MSGD results on WGB2a. 116
A.10 MSGD results on WGB1. 117
A.11 MSGD results on WGB2. 118
A.12 MSGD results on ENG, RUTH and KX. 119

List of Tables

3.1 Number of samples for a single WiFi/BLE beacon in different directional datasets. . 33
3.2 Survey durations 37

4.1 WGB2a SLAM Accuracy Summary. 49

5.1 Statistics of all datasets 63

6.1 Statistics on loop closures before and after validation. 96

Chapter 1

Introduction

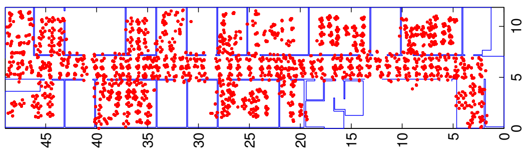
Determining the location of people indoors is a crucial enabler for pervasive computing applications. Research into indoor location systems began more than 20 years ago [58, 51, 2]. Multiple techniques have been used for positioning. For example, the Active Badge determines a person's location to room accuracy based on Infra-red signals [91]. The Active Bat [37, 92, 97] and the Cricket system [74, 82, 6] provide centimetre-level accuracy by the use of ultrasonic sensors. However, these systems require dedicated infrastructures which are prohibitively expensive to build, deploy and maintain.

Today, arguably the most prevalent indoor location systems are based on signal fingerprinting: devices are provided with a pre-existing signal map for the environment, and an online positioning phase matches the current signal observations at a device to position it on this map. This kind of system is usually based on WiFi [5], mainly due to its ubiquity, but many other signals such as Bluetooth Low Energy (BLE) [22], cellular [73], DECT [53], and magnetic field strength [12] can be used. The advantage of fingerprinting systems is the capability of fine-grained continuous positioning and tracking. The prerequisite of fingerprinting is a map of some spatially-variant but temporally-stable signals. For instance, WiFi fingerprinting requires us to build a map that associates WiFi Received Signal Strength (RSS) with positions. However, the initial creation and the subsequent long-term maintenance of the signal maps are a major challenge for fingerprinting systems.

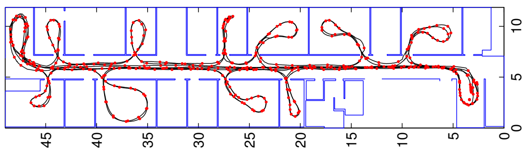
In early work, *manual surveys* were used to build the signal maps, which required a surveyor to visit each point on a regular grid and measure the signal fingerprint there. In 2010, a WiFi-based indoor positioning system was deployed at the COEX complex in Seoul, Korea, which took 15 surveyors 2 weeks to complete the signal survey work in an area of more than 450,000m^2 [35]. In dynamic environments like this huge shopping mall, frequent re-surveying is necessary but prohibitively expensive. To illustrate, Figure 1.1a shows the survey points for a typical manual survey (but in a much smaller area). Obviously, this process has many disadvantages, which are summarised below:

- It is laborious and not considered scalable. For large-scale environments like a shopping mall, this process is especially tedious and costly in terms of man-hours.
- The way of manually specifying the position of current survey point is often inaccurate, which can cause erroneous signal maps and affect the ultimate positioning performance.
- If the environment is very dynamic, regular resurveys may be needed, but performing the manual survey frequently is not scalable.

An alternative survey method is what we term a *path survey*, during which the surveyor moves continuously through the environment and signal measurements are taken by the surveying device



(a) An example of grid-based manual survey. Red points are the survey points. 206 minutes were used for this survey.



(b) An example of a path-based survey in the same environment. Trajectory in black is the survey path the surveyor took. Red points are the survey points where signal samples were taken along the path. 9.6 minutes were used for this survey.

Figure 1.1: Manual survey and path survey. Please note that the two sample surveys were performed with the aid of an high-accuracy external positioning system (i.e. the Bat system [1]) to provide the groundtruthed position of the sample device (surveyor) in real time. Without Bat, the grid-based manual survey could take even longer time and it would be a challenge for the path survey to recover survey trajectory accurately.

along the path. Figure 1.1b shows a typical path survey. The paths could be pre-defined or entirely ad-hoc. For example, applications such as WiFiSLAM, Google Indoor Maps [61] and IndoorAtlas [44] require the surveyor to move in manually-specified straight line segments. A more efficient way is to estimate the survey path by Pedestrian Dead Reckoning (PDR) algorithms applied to commodity inertial sensors found in smartphones. However, PDR algorithms drift quickly, so the error needs to be constrained by floor plans if available, or by the application of Simultaneous Localisation and Mapping (SLAM) algorithms[1]. Once the survey path is recovered (manually or via the algorithms like SLAM), it is interpolated at the times signal measurements were made to produce a sequence of survey points. Signal maps are then generated from signals collected along these paths rather than a regular grid of points.

In terms of speed, a path survey is undoubtedly better than a traditional grid-based manual survey. However, path surveys are not well-studied. This thesis investigates key aspects of path surveys, with quantitative and rigorous analysis results given. Based on these, automated signal survey systems are proposed to streamline the signal survey process. Efficient and robust algorithms are proposed and evaluated against state-of-the-art alternatives.

1.1 Research questions

How well can a path survey approximate a manual survey? The signal maps generated from a path survey can be seen as approximations to those that result from a detailed manual survey. But how well can the path survey approximate the manual survey? A quantitative and rigorous analysis is provided in Chapter 4.

[1]In robotics, SLAM algorithms localise a robot in an unknown environment and simultaneously build a map of this environment. In the signal survey context, SLAM algorithms can be used to estimate the walking trajectory of the surveyor. Observations like wireless and environmental information are typically used by SLAM to achieve this task. Please refer to [86] for details of various SLAM algorithms.

How should the path survey be performed in order to provide best positioning performance? A path survey can be carried out in different ways in an environment (e.g. with different degrees of coverage). How a path survey is taken can affect the ultimate performance. Clear guidelines need to be provided and followed properly.

How can the surveyor's path be recovered accurately and efficiently? The key for a path survey is the accurate recovery of the survey trajectory. PDR algorithms drift quickly so cannot be used alone. How PDR errors can be corrected robustly and efficiently is not well-studied. Automated path survey systems need to be designed and evaluated to answer this question.

What inputs are needed for accurate trajectory recovery? Floor plans can be used to constrain PDR errors, but they are not always available. Is a floor plan indispensable to a path survey? How should the survey system deal with the cases both with and without floor plans?

1.2 Limitations of scope

This thesis focuses on a single component of a complete indoor positioning system—the signal survey component. The following limitations apply:

Signal mapping but not positioning. A signal map associates some property of the signal (usually signal strength) with locations in an unknown environment. This thesis is concerned with how to build this signal map, the key to which is to recover the survey trajectory accurately, efficiently and robustly. To evaluate the signal maps being built by the proposed automated survey systems, some commonly used positioning techniques (e.g. the Gaussian Process Regression-based fingerprinting) are adopted. This is a straight-forward way to prove the reliability of the signal maps. And it is also useful for the comparisons between performance of different survey techniques. But we should note that multiple factors can affect the ultimate positioning performance, e.g. whether using the survey data as input to a regression algorithm to generate a continuous map, what kinds of algorithms are used to achieve positioning and tracking. How these factors affect the positioning accuracy and how to achieve better positioning results will not be covered in this thesis. The basic idea of this thesis is that better trajectory recovery results in better signal maps, and better maps give better positioning regardless of the positioning algorithms. Therefore, this thesis focuses on the signal mapping but not positioning.

Empirical fingerprinting only. In some works a signal propagation model is built to achieve positioning by estimating the distance to signal sources (e.g. through trilateration). Signal survey for this kind of positioning scheme could be seen as mapping for the signal sources in the environment. This can be achieved through typical SLAM algorithms. However, a reliable signal propagation model is hard to build for complicated environments as found indoors. And this approach cannot be used on signals without point sources associated with specific measurements (e.g. magnetic fields). So this thesis deals with the signal survey for empirical fingerprinting purpose only. For this purpose, we associate only the signal strength information with locations but do not infer the positions of signal sources explicitly (although it is achievable from the maps we build).

Signal survey by dedicated surveyor but not crowdsourcing.. Some researchers have advocated crowdsourcing the radio maps. In some sense, the survey is then 'free': a survey point is created by any device that knows its location (or can be subsequently located) and can measure the local signals. A portion of the radio map is created *and maintained* simply by one or more devices being in that area and willing to report measurements. This is a highly scalable concept, but there are a number of practical issues that have prevented wide usage of it to date:

- Device heterogeneity makes it difficult to combine crowdsourced measurements. A number of previous works have reported significant differences in the signal strength measurements made on one phone model to those made on another in the same context [48].

- Even the same device can record a different fingerprint at the same location according to its context. For example, being carried in-hand vs in-pocket vs in-bag vs within a dense crowd.

- The crowdsourced data collection is battery-intensive. It usually requires all of the inertial sensors to be on, continuous WiFi scanning, etc. Typical users are reluctant to run sensors they are not directly using since they reduce battery life; heat up the phone; and interfere with normal usage (e.g. repeated WiFi scanning adversely impacts the WiFi performance).

- Map quality can vary dramatically according to the volume and quality of the crowdsourced data in a specific space. This results in inconsistent location accuracy, which is difficult for location-aware applications to handle.

- Security and privacy are potentially at risk. Malicious users could contrive to adapt the map to their advantage, and privacy is at risk unless the data are carefully anonymised (which may be difficult given that devices must be individually profiled for best results).

Based on the above consideration, this thesis investigates a different approach. We retain the notion of a *dedicated surveyor* (i.e. a user who is willing to follow specific guidelines to explicitly collect the fingerprint data needed), and focus instead on how to make their job much easier. By removing the laborious manual survey, we enable efficient signal surveying without accepting the disadvantages of crowdsourcing listed above. The remaining cost is the need for a dedicated surveyor to do the survey task, perhaps on a (semi-) regular basis. We are able to reduce this cost to a simple walk that passes within a few metres of anywhere that positioning is required. Surveying a typical office space takes only minutes and may even be carried out by security personnel or cleaning staff (both of which are expected to visit all of the building regularly).

1.3 Thesis outline

The outline of the remainder of this thesis is as follows, with a graphical illustration shown in Figure 1.2.

Chapter 2 describes existing indoor positioning systems, highlights both the advantages and disadvantages of fingerprinting-based systems, and motivates the remainder of this thesis.

Chapter 3 quantitatively evaluates path surveys with reference to a detailed manual survey. To explore the upper-bound of real-world path survey performance, an alternative high-accuracy location system is used to compare best-case path surveys to manual surveys in terms of map quality and positioning performance. Guidelines about how the path survey should be performed are provided. This chapter demonstrates that a path survey can provide maps of equivalent quality to a manual survey if the provided guidelines are followed and an accurate trajectory is estimated.

Chapter 4 introduces a SLAM-based dedicated surveying system that uses the fast-varying magnetic field to constrain the trajectory, which is then used to generate maps of *other* signals such as WiFi/BLE. The proposed system achieves high accuracy and efficiency using only a smartphone. No external infrastructure or floor plans are required. It also has features such as supporting free movement of the surveyor and allowing one-shot (push-to-fix) positioning. Extensive experiments demonstrate the efficiency and reliability of the proposed system,

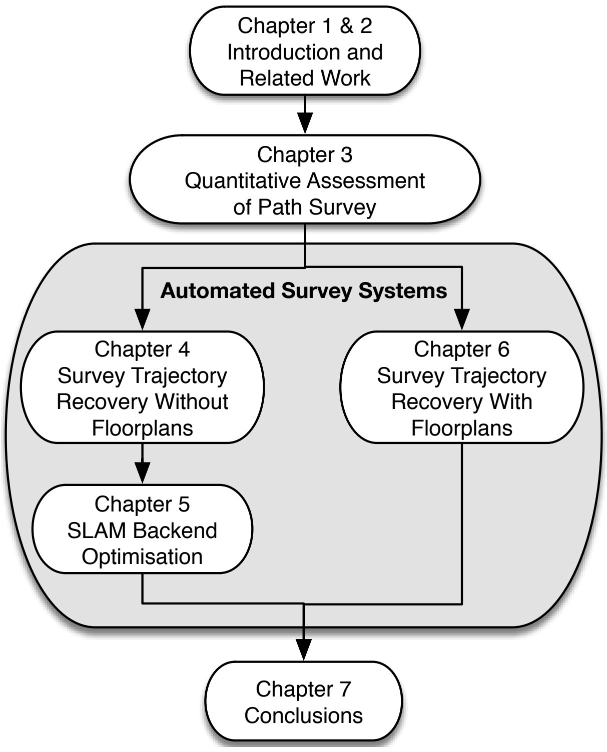

Chapter 1 & 2
Introduction and Related Work
Chapter 3
Quantitative Assessment of Path Survey
Automated Survey Systems
Chapter 4
Survey Trajectory Recovery Without Floorplans
Chapter 6
Survey Trajectory Recovery With Floorplans
Chapter 5
SLAM Backend Optimisation
Chapter 7
Conclusions

Figure 1.2: Thesis outline.

Chapter 5 optimises the SLAM back-end component used in the system proposed in the previous chapter. State-of-the-art SLAM back-end algorithms are evaluated comprehensively. An optimised back-end algorithm is proposed and evaluated against the state-of-the-art. The proposed algorithm is shown to outperform state-of-the-arts in some key aspects.

Chapter 6 proposes a survey system that uses floor plans and magnetic sequence-based loop closures to recover the survey trajectory. It differs from the system proposed in Chapter 2 by producing trajectories consistent with a floor plan. The proposed system is evaluated against state-of-the-art algorithms and shown to be more robust to noisy PDR results.

Chapter 7 concludes the thesis with summarised contributions and answers to the research questions posed in this chapter. The outline for future work is given.

Chapter 2

Related work

The larger aim of this work is to provide indoor location and so it is important to review the current state-of-the-art. Existing indoor positioning systems can be divided into three categories: the systems based on dedicated infrastructure, the systems based on existing infrastructure and infrastructure-free systems. We will consider each in turn.

2.1 Dedicated infrastructure solutions

This type of system requires the deployment of dedicated infrastructure to provide positioning service indoors. Typical technologies adopted by these systems include infra-red (IR), Bluetooth Low Energy (BLE), ultrasound, and ultra-wideband (UWB). The most significant advantage of them is the ability to achieve great robustness and high accuracy. They are usually based on the concept of proximity or on properties of a signal as it passes through space (e.g. directionality or time of flight).

A good proximity example is the *Active Badge* system, which is an indoor location system based on IR sensor networks [91]. It provides room scale accuracy based on a simple idea: IR signals cannot penetrate walls so it can robustly position a user to a room. Despite its robustness (because of the simple rationale it adopts), the coarse location accuracy is its major drawback.

More recently, many proximity-based systems are based on BLE beacons. A well-known solution is the Apple's *iBeacon* [43]. This kind of system deploys BLE beacons at specific points of interest. These beacons communicate with nearby mobile devices. The physical location of a device is inferred from the information embedded in the messages being transmitted (e.g. distance could be inferred based on the received signal strength when receiving a BLE message). An iBeacon works like a location marker. So a large number of BLE beacons are needed if continuous positioning is required, which is expensive for large-scale environments like a shopping mall.

Many signals have been used for timing-based positioning. For example, ultrasound-based solutions give much higher positioning accuracy than proximity systems. A representative ultrasound-based system is the *Active Bat* [37, 92, 97]. This system requires to deploy ultrasonic beacons in the environment, and attach a ultrasonic sensor to each user. By estimating the distance between a user and 3 or more beacons, the position of the user could be inferred by trilateration. Active Bat locates people to within 3 cm 95% of the time with a 10–15Hz update rate. Another similar system is the Cricket system [74, 82, 6].

Other systems are based on radio signal, notably UWB [78, 60, 31]. These systems work in a similar manner as the ultrasound-based systems: UWB beacons are required to be installed in the environments and UWB sensor needs to be attached to the target being tracked. Typical accuracy of UWB positioning system is within 10 cm [60].

However, the most important disadvantage of all the systems described above is the cost to build, deploy and maintain these dedicated infrastructure. For example, a large number of sensors need not only to be deployed, but also to be charged. Power consumption of both ultrasound and UWB sensors are high, so battery-based charging solution is not suitable for long term deployment. Therefore, powering the sensor network by mains electricity is necessary. But wiring cables throughout a building for hundreds or even thousands of sensors is not practical. So although these systems exist commercially (e.g. both *Ubisense* [1] and *DecaWave* [2] provide UWB-based positioning solutions) and can provide high-accuracy positioning performance, they are usually deployed for only niche applications and not likely to be used in daily life. Moreover, most modern smartphones are not equipped with any IR, ultrasound or UWB sensors, which further limits the popularity of such systems.

2.2 Existing infrastructure solutions

This type of indoor positioning system is built upon re-using the infrastructure that has already been deployed. For example, currently more and more WiFi routers are deployed in public areas. If a WiFi signal propagation model could be built, then the distance between a mobile device and each WiFi access point could be estimated. With three or more such estimations, the positioning of this mobile device could be inferred by trilateration. However, building a reliable radio propagation model for the complicated indoor environments is hard. Instead, the most prevalent existing infrastructure solutions are based on the empirical fingerprinting method. Rather than building a radio propagation model, this method matches the current signal observations (fingerprints) at a device to position it on a pre-defined fingerprint map. The idea began with the RADAR system [5]. Since then there have been a plethora of research and commercial systems using indoor fingerprinting (e.g. [38, 99, 47]). These systems have generally used WiFi signals or BLE signals [22], but cellular [73], DECT [53], and magnetic field strength [12] have also been used.

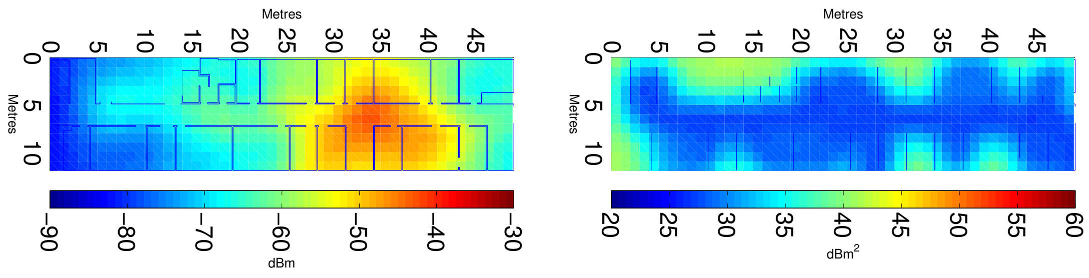
The fingerprinting-based systems have significant advantages like low deployment cost (infrastructure is already there for use, especially in public areas) and reasonable accuracy (typically between 2–5m on average). But the initial creation and subsequent maintenance of the signal maps is a major deterrent of these systems. As described in Chapter 1, the signal map is usually created via a manual survey. We advocate that a path survey is a promising alternative to manual survey but it has not been well-studied yet.

Manual surveys result in a set of survey points distributed throughout the space of interest. These points can be used as-is, forming a *raw map*. Positioning is then usually achieved through a k-NN approach to find the best matching k survey points, where k typically lies in the range 1–5 [38].

An alternative approach is to use the survey data as input to a regression algorithm to generate a continuous map. For a dense manual survey there is often little sensitivity to the sophistication of the regression algorithm since it is only being tasked with predicting the values a short distance from one or more inputs.

However, for a path survey the interpolation or regression stage is often crucial. Because the path only samples an irregular subset of the space, there is a greater need to predict values further from the survey points. It is difficult to predict the variation of WiFi-like signals with distance in an indoor environment and so non-parametric regression techniques are favoured. Gaussian Process (GP) regression has emerged as the de-facto regression algorithm for WiFi data [25, 24]. Full details are available elsewhere ([75]). Here we note only that the technique provides a normal *distribution* for the signal value at any given point in space, rather than a single value. So, we therefore have an

[1]http://ubisense.net
[2]http://www.decawave.com/



(a) The expected value at each point in space.

(b) The variance estimate at each point in space.

Figure 2.1: Sample GP regression map built from a path survey.

expected value and a variance estimate at each point in space (Figure 2.1).

2.3 Infrastructure-free solutions

The infrastructure-free systems do not rely on external infrastructure to achieve positioning. Instead, they track the motion of the human (or any moving object) using sensors mounted on the body. Typical sensors used by these systems include the inertial measurement unit (IMU) and visual sensors (cameras).

Visual odometry (VO) systems estimate the motion of a moving object (e.g. a smartphone or a robot) by active vision [80], which can achieve sub-pixel accuracy in ideal case. However, vision-based systems are easily affected by environmental factors. For example, good light condition and rich visual features in the scene are key factors to the success of VO systems. Also, vision algorithms usually require high computational cost, so special hardware like the graphics processing unit (GPU) is often needed. Finally, they require the camera to be outside the pocket and have privacy implications.

IMU is usually used to implement the inertial navigation system (INS). An IMU-based INS integrates the acceleration and rotation to track the motion of the target relative to its initial position [88]. Typical IMUs used for human tracking – the Pedestrian Dead Reckoning (PDR) systems – are based on the microelectromechanical systems (MEMS) [95]. The advantage is that they can be small and light so convenient to be mounted on human body. But a common problem is that the inevitable drift accumulates quickly over time. So, instead of making no assumptions about the movement of a pedestrian, the prior knowledge of human gait cycle could be used to constrain the INS [28, 45]. A commonly used method is the zero–velocity updates (ZUPTs) [81, 67, 66], which requires mounting the IMU on a pedestrian's foot in order to correct the INS when the foot is grounded (so the velocity should be zero). This method could largely reduce the drift but a dedicated foot-mounted IMU is needed.

PDR algorithms could also be implemented on a modern smartphone [36]. However, mounting a smartphone on user's foot is not suitable in most cases, and ZUPTs cannot be applied on a handheld smartphone (the movement of hands is arbitrary). So smartphone-based PDR systems are usually based on step detection technique: the steps could be detected using simple techniques such as thresholding the magnitude of the accelerometer [8], the direction of each step can be inferred by gyroscope output, and the step length can be estimated empirically or learned based on a mathematical model [3], then, the motion of a pedestrian can be tracked using a handheld smartphone only. However, due to the gyroscope noise (MEMS sensors on modern smartphone) and possible errors in the step detection as well as step length estimation, drift is also inevitable in the step-based PDR systems.

In summary, without applying any external constraints on the pedestrian's movement, a pure PDR system (either running on a smartphone or using a foot-mounted IMU) would give noisy trajectory estimate. But by combing building floor plans and particle filters, PDR can track a pedestrian to the metre level [93, 96, 94, 49, 52].

The common disadvantage of the infrastructure-free systems is that they can only provide relative positioning. The initial position of a user cannot be determined in a "push-to-fix" style, i.e. they cannot tell where the user starts. Although the initial position may be inferred eventually if floor plans and particle filters are available, many applications still cannot be enabled without immediate position fix.

2.4 Combined/Fused solutions

Each of the three kinds of solutions described above has its own drawbacks, but we believe that by combining/fusing different kinds of systems, more practical solutions could be achieved. The focus of this thesis is on how to fuse the existing infrastructure solutions (i.e. fingerprinting systems) and the infrastructure-free methods (i.e. PDR or VO systems) to achieve low-cost positioning systems. As mentioned before, the traditional method to build the signal map (i.e. the manual survey) for fingerprinting systems is expensive and not scalable for frequent re-survey. Then if we can use the infrastructure-free methods to enable efficient path survey, the initial creation and subsequent maintenance of the signal maps could be low-cost. By achieving this, the fingerprinting-based positioning systems could be easily deployed and maintained in large-scale environments.

The key to path survey is accurate trajectory recovery. VO systems could achieve this but they typically have strict requirements on environmental conditions and are much more computationally intensive than PDR algorithms. So, PDR systems are favoured. They are ideally suited to a dedicated surveyor performing a path survey in an arbitrary environment: the surveyor perform the survey by simply walking around holding the survey device, and then the survey path can be recovered by the PDR system.

Rai et al. proposed a similar approach for crowdsourcing [4]. In practice, however, consumers are resistant to permanently turn on the necessary sensors and processing due to high battery consumption. Turning it on for short periods might be acceptable, but it is typically difficult to set a start position on the floor plan when initialising the system from cold. In addition, PDR algorithms are not yet robust enough to deal with the arbitrary positions and movements of a smartphone seen when crowdsourcing.

The problem with the PDR systems is that drift accumulates as time goes on. Some systems have applied Simultaneous Localisation and Mapping (SLAM) techniques to constrain PDR drift. These systems use the spatially-varying signals to correct PDR error. Once a trustworthy trajectory is established, the observed signals are re-used to form a path survey. Most SLAM path survey approaches are based on graph optimisation (e.g. [42, 24]), although particle filters are also used [23], as well as hybrids [21]. We discuss the technical details of such algorithms where they arise later in this thesis.

2.5 Summary

The Fingerprinting-based indoor positioning systems are promising for large-scale deployment. But the traditional method to build the signal maps for these systems is not scalable. This thesis focuses on an efficient alternative map building method, i.e. the path survey method. The rest of this thesis first gives quantitative assessment of the path survey. After that, robust, efficient and accurate path survey systems are proposed and evaluated. More related work is also given in relevant chapters.

Chapter 3

Quantitative assessment of path survey

3.1 Introduction

Indoor location systems have a rich history of technologies and techniques [58, 51, 2]. However, no single system has yet emerged to provide ubiquitous indoor positioning, primarily due to the need to deploy local infrastructure of some sort. The most prevalent systems are those that are able to leverage existing infrastructure, and of those most are based on applying signal fingerprinting techniques to WiFi signals [5]. Devices are provided with a pre-existing signal survey (map) for the building, which they use to position themselves. The fingerprinting approach in general is a flexible one that can be applied to any spatially-varying but temporally stable signal. Furthermore, the fingerprint maps may have uses other than positioning (for example to assist the deployment of WiFi access points to ensure uniform coverage and minimal overlap).

The key issue with fingerprinting is the creation of the signal maps (often called the *survey* or *offline* process) and their maintenance over the long term. Early work in this area used manual surveys, requiring a surveyor to visit each point on a regular grid and measure the signal fingerprint there. This is a laborious and time-consuming process that is a deterrent to the creation and use of such systems.

An emerging approach is the use of what we term *path surveys*. Here the surveying device is somehow tracked as it moves continuously through the environment and records signal measurements. Signal maps are then generated from signals collected along these paths rather than from a regular grid of points. The paths themselves may come from an alternative location system, or be jointly estimated based on the signal to be mapped using a variant of a Simultaneous Localisation and Mapping (SLAM) algorithm. These maps can be seen as approximations to the conventional manual survey. To date, however, there is no quantitative and rigorous analysis about how well such path surveys approximate their manual equivalents, and no guidelines about how the path survey should be performed.

In this chapter, we explore these issues in detail. We make four primary contributions:

1. We describe the collection of a dataset containing a detailed manual survey and ground-truthed path surveys.
2. We consider Bluetooth Low Energy in addition to the usual WiFi signal.
3. We use an accurate ground truth to compare best-case path surveys to manual surveys in terms of map quality and positioning performance, which gives the upper bound of real-world path survey performance.
4. We provide guidelines for path survey based on our quantitative analysis; We demonstrate that by following our guidelines, a path survey can achieve good efficiency and accuracy.

3.2 Path surveying techniques

A path survey collects signal survey points along a continuous path through space rather than at discrete, evenly-distributed points. Like the signal it is measuring, the path may be dedicated to the survey task (where a surveyor moves with the sole purpose of building the maps); or opportunistic (where crowd-sourced measurements are collected from users moving with some other purpose in mind).

The primary advantage of a dedicated surveyor is that the movement can be constrained: the surveyor can be instructed to provide good spatial coverage; to provide occasional manual position fixes; and to move in a way that produces good position estimates. For example, some proposed schemes use some form of Pedestrian Dead Reckoning (PDR) algorithm applied to inertial sensors in the survey device. This can give high quality trajectory estimates if the user keeps the device in one position and only ever walks in the direction it points in (i.e. no side steps or back steps) at a constant speed. Applications such as WiFiSLAM, Google Indoor Maps [61] and IndoorAtlas [44] took this one step further and require the surveyor to move in manually-specified straight line segments. The segments are then interpolated at the times signal measurements were made to produce a line of survey points. More advanced algorithms from the SLAM family may also be applied by having the surveyor perform particular actions (usually walking in loops).

Crowd-sourced data bears some similarity to the segment-walking technique just described: multiple users are expected to contribute short segments that can be combined over time to form a comprehensive survey. Since the users behave naturally, a higher density of measurements will be associated with more popular regions. Thus crowdsourced maps are naturally more detailed in high footfall areas such as corridors, at least in principle. In practice, crowdsourcing is challenging due to the lack of constraint in the trajectories and movements: patchy spatial coverage; devices being held in different ways; segments being non-linear with complex movements that are difficult to characterise. Even if a trajectory can be established, it is often difficult to anchor it to the floor plan, and thus to provide enough reliable survey points to make a comprehensive map.

Some have argued that crowd-sourcing is the solution to the maintenance of a map surveyed through some other means. Assuming there is sufficient redundancy in the observed signals to accurately position the user using only a subset of the received signals, a random sample consensus (RANSAC) algorithm can be applied to identify and correct outlier measurements [26]. The key challenge here is verifying that the estimated position is sufficiently accurate to drive an update. To date there has been no detailed study of the validity of crowd-sourced signal maps, although it shows promise.

The key requirement of the path survey is to recover the survey path taken by the surveyor. We divide the above-mentioned techniques into two classes:

- **PDR-based Survey.** Fusing step detection PDR algorithms running on a consumer device with a building floor plan can provide accurate tracking around a building [93, 96]. The best results are obtained when the device user moves smoothly and continuously; turns regularly; and holds the device in one position. In addition to the floor plan, various Simultaneous Localisation and Mapping (SLAM) techniques can be applied to correct PDR error. The core requirement is that the path contains loops that the system can observe by monitoring the signal environment.

- **Segment Survey.** This uses the approach popularised by Google Indoor Maps and IndoorAtlas. The surveyor manually marks a small number of points on a floor plan image and then walks in straight lines between them. Unlike the PDR-based survey, this manual path specifying method is difficult to deal with complicated survey routes. So typically only a few segments along corridors are used.

The accuracy of the path derived by either method is affected by multiple factors. The most obvious one is the noisy sensors in the survey devices (typically smartphones). The algorithms (SLAM, particle filter etc.) used to correct PDR error have impacts on the PDR-based survey accuracy. The user's behaviour mode (e.g. whether the user walks along the specified segment accurately) can affect the segment survey accuracy.

To quantitatively evaluate the performance of a path survey, we use an accurate positioning system—the Bat system [1]—to simulate best-case path surveys. The Bat system is an ultrasonic time-of-flight system installed throughout the test environment capable of 3D positioning to an accuracy of 3 cm 95% of the time with a 10–15Hz update rate. We require the surveyor to hold the survey device and behave just like conducting a normal PDR-based survey or a segment survey, but use the Bat system to accurately recover the survey path. In this way, we are able to give an upper bound for the real-world path survey results.

3.3 A ground-truthed survey dataset

We have collected an extensive dataset in order to bound and assess the performance of path surveys in a real-world environment using consumer devices. A detailed manual survey of WiFi and BLE signals was taken in addition to a series of ground-truthed path survey walks.

3.3.1 Environment and devices

Our tests were performed in a wing of the William Gates Building housing the Computer Laboratory at the University of Cambridge. The building itself is a three-storey office building constructed from brick, steel and concrete. Internal walls are constructed from plasterboard, wood and metal.

We gathered signal data using a custom application running on an Android smartphone (Google Nexus 5). The application recorded WiFi RSSI and BLE RSSI data which we considered to be amenable to fingerprinting. The Android WiFi subsystem was set to use 2.4 GHz only since this reduced the scan period from around 4 s to 1.5 s: a significant reduction that allowed us to better localise a scan spatially when the device was moving. All the WiFi internet connections (and BLE connections as well) were disconnected manually throughout the experiments, such that the signal RSSI can be recorded continuously without interruption.

When performing path surveys we also recorded magnetometer data (magnetic field without hard-iron compensation and estimated hard-iron bias along the x, y and z axis), accelerometer data (acceleration force along the x, y and z axis including gravity) and gyroscope data (rate of rotation around the x, y and z axis) by Inertial Measurement Unit (IMU) on the smartphone for PDR. The sample rate for IMU data is above 100 Hz. Android system API calls were used to record all these sensor data mentioned here. Also, we shut down any applications that may use IMU sensors in order not to interfere with data recording.

For accurate ground truth, the Bat location system was synchronised with the Android phone via NTP (Network Time Protocol). The active tag that the Bat system tracks was firmly attached to the back of the smartphone. So the Bat system was able to track the smartphone to within 3 cm 95% of the time with a 10–15Hz update rate.

During the experiments, the smartphone is handheld flat in front of the body of the surveyor (as if being used to navigate). The surveyor was asked not to wave the smartphone intentionally during the data recording, thus to keep the smartphone in a fixed position relative to the human body.

The test environment contained three WiFi APs; additional APs from the floor below and adjacent areas were also observed. The RSSI information of all these observed beacons was recorded. The APs had been previously deployed by the building's IT staff to provide good communications coverage throughout the building (i.e. explicitly not with positioning in mind). As such they were representative of most office buildings. Seven BLE beacons were available in the area, each set to advertise a unique identifier at 12 Hz. These were deployed solely for the purpose of positioning. The beacons were attached to the ceiling (corridors) or on ledges above windows (offices). Figure 3.1 illustrates the test area and the positions of the WiFi APs and BLE beacons within it.

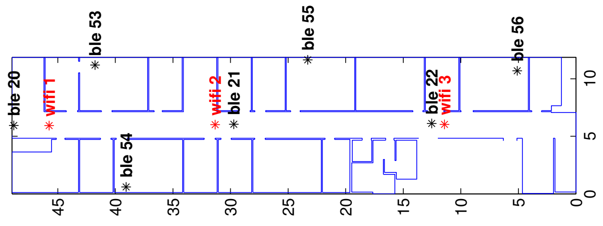


Figure 3.1: The test area.

3.3.2 Manual survey

We performed a manual survey in the test environment. Unlike previous works, we did not attempt to establish a set of positions on a regular grid since this was both very laborious and error-prone.[1] Instead, we used an approximate grid but measured the position of the device accurately at each survey point using the Bat system. In this way we achieved comprehensive spatial coverage and accurate survey positions. We aimed for an inter-point spacing of 1 m to provide a detailed survey—Figure 3.2 illustrates the survey points on a floor plan (rooms without survey points were not accessible during the survey).

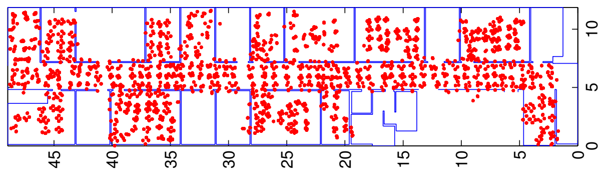


Figure 3.2: The survey points

During surveying the device was handheld as if being used to navigate. To examine body shadowing effects the surveyor rotated about a vertical axis through the phone to point in each of the four cardinal directions of the building.[2] 2.5 s of data were collected in each orientation. This period allowed for tens of BLE beacon measurements, and up to two measurements of each WiFi AP per survey point per orientation.

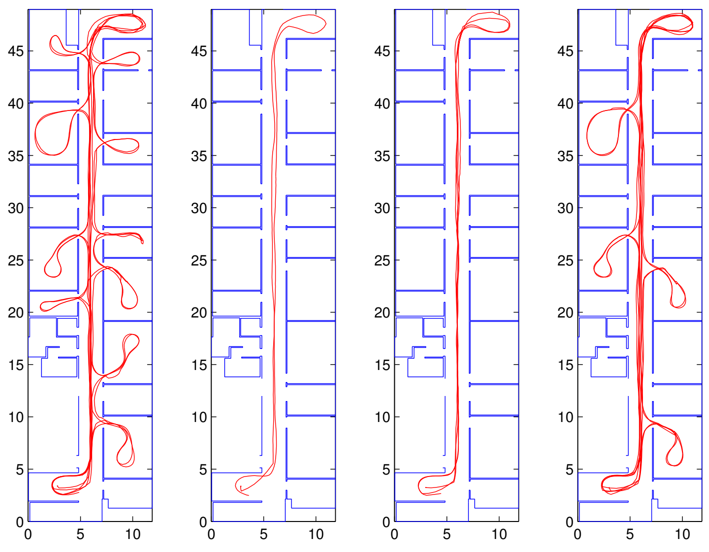
3.3.3 Path survey walks

We performed four different survey walks (labelled W1, W2, W3, and W4) with the same device held by the surveyor as if navigating. The Bat system provided ground truth location. W1 was carefully chosen to represent a dedicated surveyor visiting every room and area covered by the manual survey; W2 and W3 involved walking the corridor multiple times (to simulate segment survey); W4

[1]It is difficult to accurately mark a 1 m grid across a large indoor space and even more difficult to ensure that a device is being held over a marked position when surveying

[2]These were approximately aligned with the magnetic cardinal directions. For simplicity we assume them to be the same herein.

was free-form and visited many, but not all, of the surveyed rooms (see Figure 3.3). In total the walks contained 2,191 steps.



(a) W1 (b) W2 (c) W3 (d) W4

Figure 3.3: Survey walk paths

3.3.4 GP regressed map

Both manual surveys and path surveys result in a set of survey points distributed throughout the space of interest. The difference is that the survey points given by a manual survey are usually distributed more even and denser than those given by a path survey. To generate a continuous map, Gaussian Process (GP) regression can be used to predict signal strength distribution on arbitrary points based on the measurements collected on the survey points (Section 2.2). Please refer to [75] for more details about GP regression. Here we note only that it provides a normal distribution (represented with mean and variance) for the predicted signal strength distribution at an arbitrary point in space.

For every survey, we generated a GP map for each signal source[3]. Figure 3.4 shows sample WiFi and BLE RSS measurements taken during different surveys and the GP regressed maps (mean and variance at each point) that result from them. It can be seen that for the BLE GP maps, the variance grows quickly the further from survey points/paths we move, correctly encapsulating the growing signal propagation uncertainty. But this is less obvious for the WiFi GP maps because the WiFi signals attenuate much slower (so the correlation is stronger) over space than BLE signals[4]. Figure 3.4i shows an extreme case of this: the variance of the WiFi manual survey GP map is almost evenly distributed. This means that WiFi signals have strong correlation over space. So when good survey coverage is given (manual survey), the uncertainty of the prediction made by the GP is low everywhere.

Please note that the manual survey and all the path surveys used the accurate Bat system to infer sample positions. This is arguably a best-case scenario that would be very hard to reproduce in an arbitrary environment today. Nonetheless it serves as an important bound.

[3]We adopt the open source library provided by [75] to train the survey data and generate GP maps.

[4]Please note that the WiFi beacons are deployed for the purpose of better wireless network coverage, while the BLE beacons are deployed for positioning purpose only. So the signal strength of WiFi is generally much stronger than that of BLE.

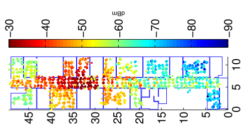
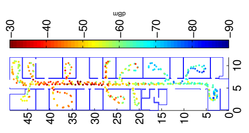
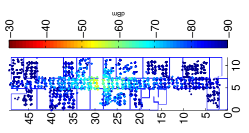
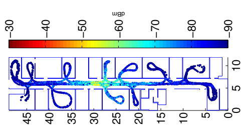
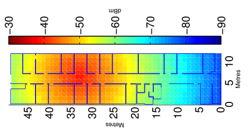
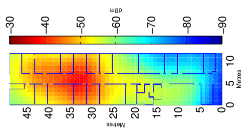
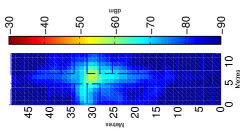
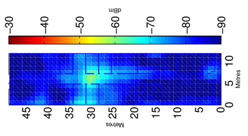
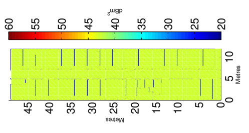
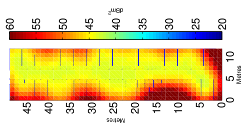
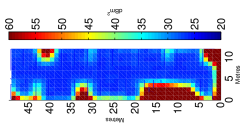
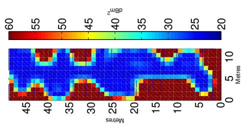
	WiFi		BLE	
	Manual Survey	Path Survey W1	Manual Survey	Path Survey W1
Aligned Signals	 (a)	 (b)	 (c)	 (d)
GP μ	 (e)	 (f)	 (g)	 (h)
GP σ^2	 (i)	 (j)	 (k)	 (l)

Figure 3.4: Sample signal surveys and the resulting GP maps.

3.4 Point comparison on regressed maps

When surveying for the purposes of wireless network deployment, the GP map itself is the end goal. Furthermore, we might expect intuitively that better map approximations allow for better online positioning. To meaningfully compare GP maps directly is a challenge since each position is associated with a normal *distribution* rather than a scalar value. Hence we must first establish a metric to compare normal distributions.

Comparison of normal distributions. For two normal distributions, $\mathcal{N}(\mu_1, \sigma_1^2)$ and $\mathcal{N}(\mu_2, \sigma_2^2)$, we form the distribution of the differences, which has the form $\mathcal{N}(\mu_1 - \mu_2, \sigma_1^2 + \sigma_2^2)$. Since we are only interested in the magnitude of the difference, we convert this normal distribution to a folded normal distribution, the CDF of which is given by:

$$F(X; \mu, \sigma) = \frac{1}{2}\left(\text{erf}\left(\frac{x+\mu}{\sqrt{2}\sigma}\right) + \text{erf}\left(\frac{x-\mu}{\sqrt{2}\sigma}\right)\right) \tag{3.1}$$

for μ=$\mu_1 - \mu_2$ and σ^2=$\sigma_1^2 + \sigma_2^2$. We then estimate the RSS difference corresponding to the 90% confidence interval, RSS_{90}. Hence 90% of the time the two distributions would agree to within RSS_{90}.

Comparison of GP maps. We evaluate $\text{RSS}_{90,i}$ at each of the grid point positions covered by the manual survey since these points are associated with the lowest uncertainty in the manual survey-derived GP map. The distribution of $\text{RSS}_{90,i}$ acts as a measure of the agreement between the two maps. Two very similar maps would show a strong concentration of RSS_{90} values within the expected measurement noise, whilst differing maps would not.

We visualise the RSS_{90} by heatmap (Figure 3.5) and CDF (Figure 3.6). The colour (value) c of a point on the RSS_{90} map represents that at this point, the predicted RSS values by a path survey GP map and a manual survey GP map agree to within c dBm 90% of the time. Similarly, the interpretation of a point (x,y) on the CDF is the proportion of grid points (y) where we expect the two maps to agree to within x dBm 90% of the time.

Figure 3.6b shows that for the BLE signal, the GP maps of W1 and W4 (which covered the space more comprehensively than W2 and W3) are very similar to the manual survey. This is because these

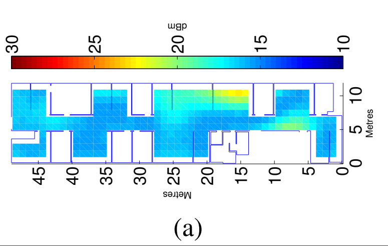
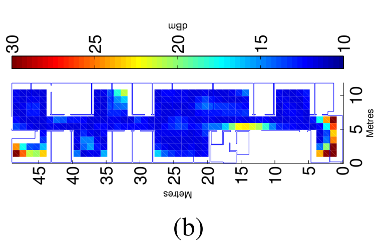
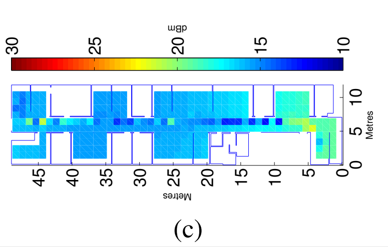
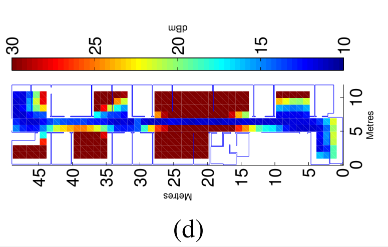
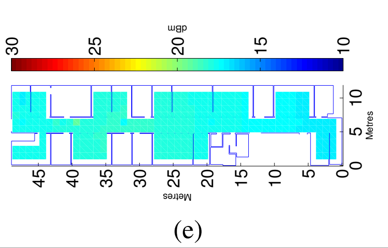
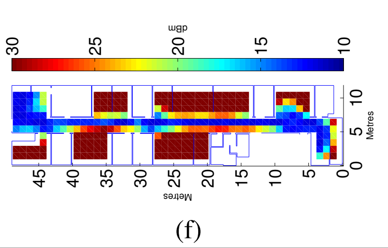
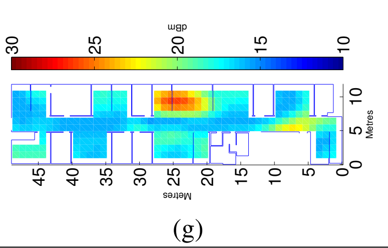
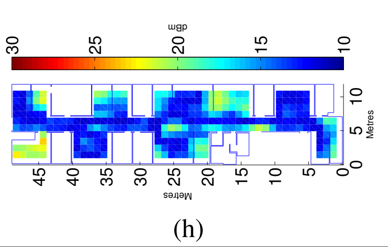
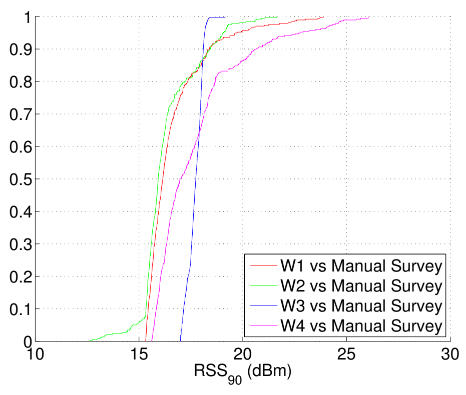
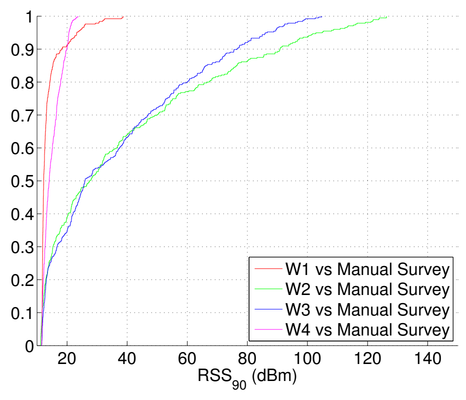
	RSS$_{90}$ Map - WiFi	RSS$_{90}$ Map - BLE
W1 vs MS	 (a)	 (b)
W2 vs MS	 (c)	 (d)
W3 vs MS	 (e)	 (f)
W4 vs MS	 (g)	 (h)

Figure 3.5: RSS$_{90}$ maps from point comparison of various GP maps (W1–W4) with the manual survey (MS) GP map.



(a) WiFi



(b) BLE

Figure 3.6: RSS$_{90}$ distribution (CDF).

survey paths cover the space comprehensively. Conversely W2 and W3 (which only visit a subset of the rooms in the manual survey and hence require prediction further from the survey path) exhibit good results only within a metre or so of the survey path.

However, this phenomenon is less obvious for the WiFi signal. One possible reason is that the WiFi signal variance over space is lower than that of the BLE signal. As can be seen in Figure 3.4, the WiFi signal attenuates more slowly than BLE over space. This is confirmed by the GP regression results—as is common we use a *squared exponential* (SE) covariance function (or "Gaussian kernel") to encode the correlation between nearby measurements:

$$k_{SE}(r) = \sigma_f^2 exp\left(-\frac{r^2}{2l^2}\right) \tag{3.2}$$

where r is the distance between two spatial locations, σ_f^2 is the signal variance, and l is the *characteristic length-scale* that determines how the correlation weakens with distance [75, 25]. The signal variance σ_f^2 and the characteristic length-scale l are usually learnt from the training data. For the W1 WiFi and BLE maps in Figure 3.4, the learned values of the signal variance σ_f^2 are 44.08 and 193.72 respectively, and the learned values for the characteristic length-scale l are 4.96 m and 2.50 m respectively. These values show that WiFi signals in distant locations correlate better than their BLE counterparts. This means it is easier for GP regression to predict the signal strength distribution in more distant locations for WiFi than BLE.

However, we note that the RSS_{90} values can be affected by many factors. For example the signal sample density, multipath, body shadowing, beacon deployment, environmental settings. So, it is complicated to take into account every possible factor that affects the final GP maps and a rigorous interpretation is hard to provide. The proposed visualisation method of GP map difference gives us an intuitive feeling about how the path survey map differs from the manual survey map. We see that when doing the path survey for BLE signal, the survey path should pass all possible locations where positioning requests may happen as closely as possible. The following sections evaluate the positioning performance of the GP maps to cross-validate the analysis results given in this section.

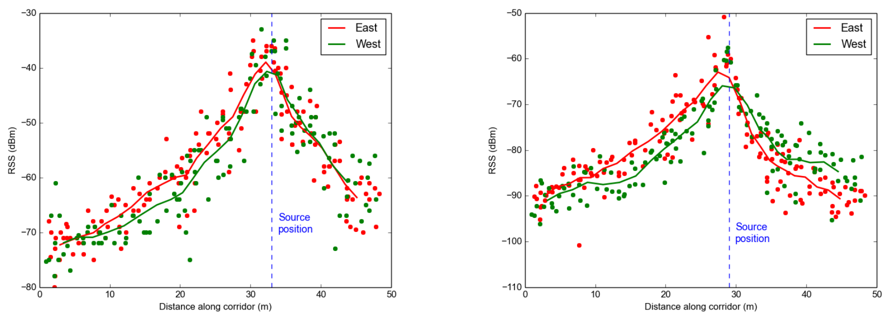
3.5 Positioning comparison on GP regressed maps

Positioning using GP maps requires the use of multiple maps (or equivalently, signals from multiple sources). It is therefore feasible that good positioning results could be achieved even when one or more GP maps are very poor approximations to the true map. To see this, consider a GP map with high variances associated with most of the evaluation points (which is an indication that the GP map is not of high quality). A sensible positioning algorithm would derive very little information from such a map—in effect, it should be discarded. Provided there are enough low-variance evaluations for other GP maps, a good positioning results is then possible. In this section we study the results of positioning using maps from path and manual surveys.

3.5.1 Positioning method

We adopt the signal strength-based location estimation method described in [25] to evaluate the positioning performance. We first use the raw data to generate GP maps and then use a standard Bayesian localisation algorithm to incorporate the variance estimate as well as the mean at each GP map location. To estimate a person's location, x, conditioned on observation of a set of signal strength measurements (fingerprints), z, we used:

$$p(x|z) \propto p(x)p(z|x) \tag{3.3}$$



(a) WiFi (b) BLE

Figure 3.7: Corridor directionality for BLE and WiFi

where $p(x)$ is the prior probability of the person being at x; and $p(z|x)$ is the likelihood of observing a set of signal strength measurements at location x. $p(z|x)$ is directly given by the GP regression.

We divided the environment into a grid of square cells of length 1 m or less, For each step at each cell we computed $p(x|z)$ and selected the cell with the maximum as the person's location. Signal strength measurements lower than -90 dBm were discarded. Note that the positioning algorithm did *not* assume prior information—i.e. our results are for one-shot positioning. This is achieved by setting $p(x)$ to a uniform distribution over the whole space. A recursive tracking scheme that would seed each positioning calculation with the posterior from the last would be expected to perform better, but could mask a bad positioning result.

3.5.2 Map directionality and space coverage of survey path

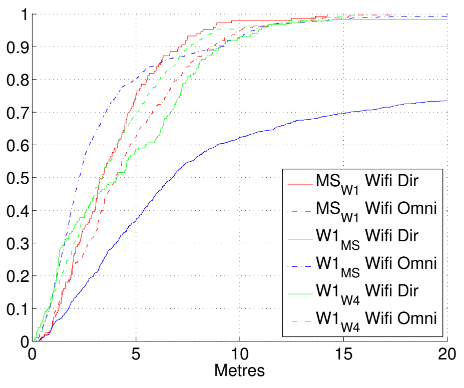
Table 3.1: Number of samples for a single WiFi/BLE beacon in different directional datasets.

	Manual Survey	Path Survey W1
East-facing WiFi	1460	77
West-facing WiFi	1452	70
East-facing BLE	1374	703
West-facing BLE	1407	626

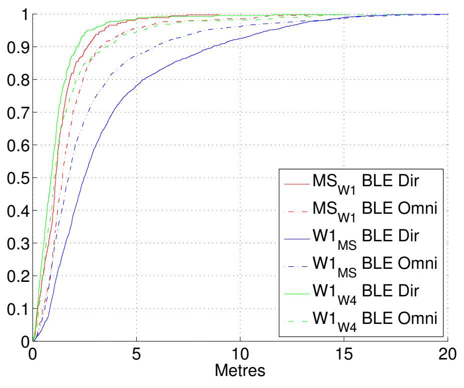
We first study the directionality of the survey data. Previous works have highlighted an orientation dependency in RSS measurements on handheld consumer devices. The dependency is primarily caused by body shadowing: the attenuation of a signal that must pass through the human body as it travels directly from source to receiver. The extent of this dependency is important for a path survey, since it may demand paths be traversed in both directions. This would clearly add to the time and complexity of the surveying task. In addition, it affects how we use the orientation information of the survey data. If the orientation plays an important role in the positioning accuracy, than more complicated orientation-based positioning algorithm is preferred.

We expect directionality to be strongest along vectors that pass through the source, and it is thus most easily observed along the straight corridor in our data. Figure 3.7 shows the East-facing and West-facing RSS values observed at the survey points in the corridor, plotted against the distance along the corridor (the distance increases in the westerly direction). The general trends are shown by the two smoothed lines. In both cases there is a clear offset between the trends that corresponds to higher RSS when approaching a source compared to receding from it.

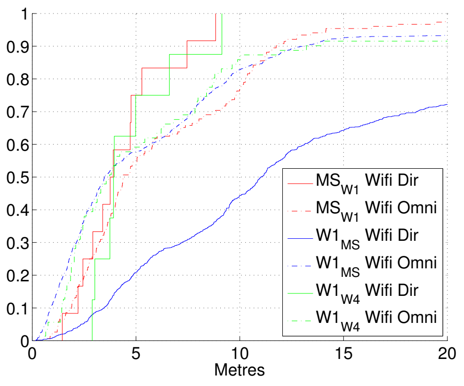
Our survey data contained orientation information, allowing us to generate both:



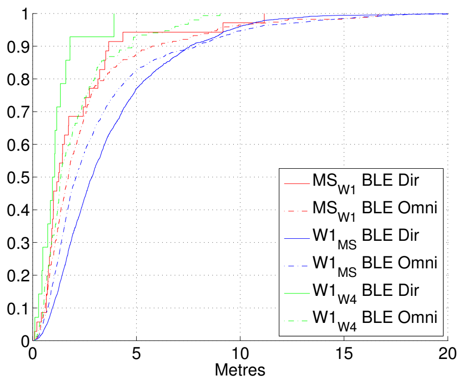
(a) Inside corridor area - WiFi



(b) Inside corridor area - BLE



(c) Outside corridor area - WiFi



(d) Outside corridor area - BLE

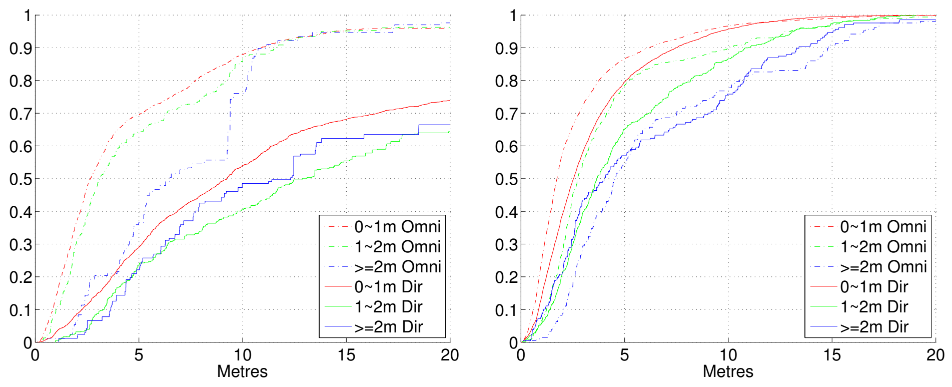
Figure 3.8: Positioning results for directionality analysis. AA$_{BB}$ implies a map generated from AA data tested using the data from BB. Directional maps are labelled 'Dir', non-directional maps 'Omni'.

- **Directional Maps**. For each signal source, divide corresponding signal strength sample dataset into several subsets according to the surveyor's direction when taking each sample. Then, create a map for each cardinal direction. When positioning, use only the map corresponding to the direction of movement. Here we only use the east-facing and west-facing datasets because these correspond to the major direction of the building. The number of samples in each dataset for a single signal source is shown in Table 3.1.

- **Omnidirectional Map**. For each signal source, create a single map using all the survey data *after* discarding the orientation labels. The directionality should then manifest as a larger variance in the values collected at each survey point.

We produce these two kinds of map using various datasets and evaluate their positioning performance using different dataset as fingerprints (input). We consider the corridor separately from the rest of the space because we expect more significant directionality there. The analysis results are shown in Figure 3.8.

First we look at the positioning results of the manual survey map (MS$_{W1}$ in Figure 3.8). For both inside and outside corridor areas, WiFi directional map outperforms omnidirectional map (about 1.46 m and 4.00 m at the 90th percentile respectively). It is the same for BLE but less significantly (about 0.65 m and 2.21 m at the 90th percentile respectively). This shows that map directionality *does* affect the positioning performance. If a manual survey is affordable, conducting the signal sampling in different directions *do* improve positioning performance.



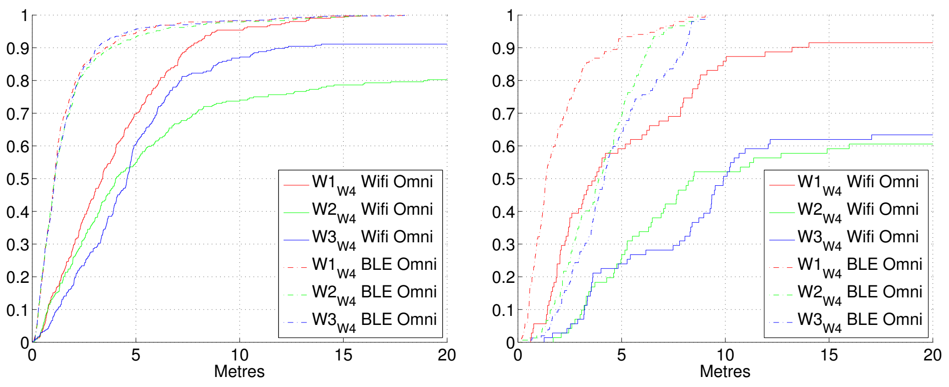
(a) WiFi (b) BLE

Figure 3.9: The positioning results of W1$_{MS}$ (W1 maps evaluated against manual survey data) broken down by distance of manual test point from the W1 survey path.

However, the omnidirectional map generated from W1 achieves better results than the directional map when evaluated using the manual survey data (W1$_{MS}$). This is especially significant for WiFi. One of the key factors is that the amount of training data used to generate the directional map is very small (as shown in Table 3.1). Hence the directional map is not a good representation of the signal distribution over the space. The results also show that W1 maps (W1$_{MS}$) are generally outperformed by manual survey maps (MS$_{W1}$). One important reason is that the W1 map is evaluated using the manual survey data as fingerprint input, so most of the test points are away from the W1 survey path, which requires good signal distribution prediction to achieve good positioning results. But the sparse training data is not sufficient to produce a good regressed map, that is why the positioning performance is poorer. Figure 3.9 classifies the positioning results of (W1$_{MS}$) by the distances between the test points and the (nearest points on the) survey path. It shows positioning accuracy decreases dramatically when the distance exceeds 2 m. This is less significant for WiFi directional map because its positioning accuracy is already very poor.

In real life, people tend to follow some common path when walking indoors. For example, people may take the same route when walking around a room (the furniture somehow "define" the route) and people tend to walk along the centre of a narrow corridor instead of walking close to the walls. This means in reality, the locations where a positioning request may happen are likely to be on those common paths. The W1 survey path is such a path that going around the whole space following the most probable pedestrian route. The W4 survey path can be seen as a subset of W1, so, if we use W4 data as fingerprint input to evaluate the performance of W1 map, the real life positioning performance can be simulated. In this case, the distances between test points to the survey path are almost zero, which is supposed to give better accuracy. This is confirmed by the positioning results shown in Figure 3.8: the positioning performance of W1 map evaluated by W4 data (W1$_{W4}$) is similar or even better than manual survey map (MS$_{W1}$). So for a path survey, good space coverage is necessary to achieve good positioning performance. This requires the survey path to cover the locations where positioning request can possibly happen within 1$\sim$2 m. In this way, all positioning requests are in the high-confidence areas on the GP regressed map and it can achieve good positioning performance as the manual survey does.

The results on Figure 3.8 also show that when evaluating W1 maps against W4 data (W1$_{W4}$), only for the WiFi signal inside the corridor area, omnidirectional map outperforms directional map but in all other cases, a directional map gets slightly better results. From these results we conclude that a path survey should cover paths in both directions for optimal results. But not doing so is not, however, likely to degrade the result significantly if good space coverage is provided (but the possibility of



(a) Inside corridor area. (b) Outside corridor area.

Figure 3.10: The positioning CDF results for W1, W2 and W3 maps evaluated against fingerprints generated from W4 data.

applying more sophisticated orientation-based positioning algorithm to achieve more significant improvements in positioning accuracy cannot be excluded). Also, considering that omnidirectional map has better performance for test points further away from the survey path (Figure 3.9), we recommend to use omnidirectional map for the path survey data. The additional advantage of using omnidirectional maps is that we do not need to produce multiple maps for each direction, which reduces the map storage as well as the position computation cost, and excludes the need for reliable orientation estimation at all times.

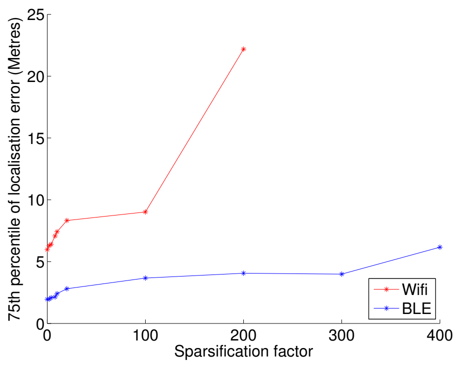
3.5.3 Segment survey evaluation

In Section 3.2 we have mentioned a special kind of path survey – the segment survey, in which typically only a few segments along corridors are used. The W2 (walking up and down corridor once) and W3 (walking up and down corridor twice) are good representations of segment survey. By the analysis given so far, we can infer that segment survey cannot provide good positioning performance as a PDR-based survey like W1 does. We confirm this by using the W4 data as fingerprint input to evaluate W1, W2 and W3 omnidirectional maps. Results are given in Figure 3.10. It shows that outside the corridor area, W1 outperforms W2 and W3 significantly for both WiFi and BLE. This is because the test points are all on the low confidence areas of the W2 and W3 maps. It also shows that inside the corridor area, W1 outperforms W2 and W3 for WiFi positioning but they have indistinguishable results on BLE positioning. This shows the importance of sample density. W1 has more samples in the corridor area than W3, and W3 has more than W2, so WiFi positioning performance of W1 is better than W2 and W3. But the sample rate of BLE is much higher than that of WiFi (10 Hz vs 0.5 Hz) on the android phone we use, which mitigates the difference in the sample density of the three surveys. So the difference in the BLE positioning performance is less significant than the difference in the WiFi positioning performance.

So we conclude that, segment survey which only covered the corridor area of the environment does not give good positioning performance. Good space coverage is necessary when taking the path survey.

3.5.4 Sample density evaluation

To quantitatively evaluate how the sample density affects the positioning performance of the GP maps, we sparsify the training data of W1 to generate different GP omnidirectional maps. These maps



(a) WiFi.

Figure 3.11: Sample density evaluation. The training data of W1 are sparsified to different extent (get 1 out of n samples) and resultant GP maps are evaluated using fingerprints generated from W4 data. When the sparsification factor reaches 300, there are too few WiFi samples to generate a valid GP regressed map.

are evaluated using W4 data as fingerprint input. Results are shown in Figure 3.11. It shows that the positioning performance decreases with the extent of sparsification, but with WiFi map decreases much more dramatically than BLE map. This is because the sample rate of WiFi is extremely low compared with BLE (about 0.5 Hz vs 10 Hz). Therefore, repeating some part of the paths multiple times is necessary when taking the path survey because this can improve the map quality by increasing the number of signal samples (especially for WiFi positioning).

3.6 Conclusions and further work

In this chapter we proposed a method to visualise the difference between GP maps generated from manual survey and path survey. Straight-forward positioning experiments were also conducted to show various features of path survey maps. Based on our analysis we propose several guidelines about how the path survey should be taken. By following these guidelines, path survey can be a low-cost alternative for the laborious manual survey: in our experiments the manual survey took more than 3 hours but each of the W1–W4 took less than 10 minutes (Table 3.2).

Table 3.2: Survey durations

Survey	W1	W2	W3	W4	Manual
Time (mins)	9.6	2.1	3.3	7.5	206

The guidelines are as follows:

1. The survey path should pass within 1$\sim$2 m of any given point where positioning might be required. This is to provide good space coverage so that all positioning requests happen in the high-confidence areas of the GP regressed map.

2. The surveyor should repeat some parts of the path to increase the signal sampling density. This is especially necessary for WiFi positioning because the WiFi sample rate on a modern smartphones is typically as low as 0.5$\sim$1.0 Hz.

3. For the directionality of the wireless signals, we have found that ignoring this does not degrade the positioning performance significantly. But considering the possibility of more sophisticated

orientation-based positioning algorithm being proposed, the surveyor can still try to pass in both directions (e.g. walking up and down a corridor) to record the directional information for potential use. This will also increase the sample density of the signals.

Our survey guidelines are simple to follow and so anyone equipped with a smartphone can quickly perform a survey. Many public buildings have security or building management personnel who regularly walk through them—these offer an ideal opportunity to preform regular surveys.

Please note that in most cases, a high-accuracy external positioning system like the Bat system used here is seldom available. PDR algorithms can be used to recover the survey path but with large errors and drifts in the results. Then repeating some parts of the path is very valuable not only because it can increase the signal sampling density, but also because it can provide spatial constraints between sample points on the survey path. These constraints are called *loop closures*. The loop closures can be used to correct the PDR result so that the survey path can be accurately recovered without an external positioning system (details about this are in the following chapters of this thesis).

So, this chapter has proved the feasibility of the path survey, and provides guidelines about how the path survey should be taken, which is the foundation of our following works.

Chapter 4

Automated signal survey

The previous chapter shows that a path survey can significantly lower the survey efforts and provide similar positioning performance as a manual survey given that it follows the suggested guidelines. However, the path survey evaluated in the previous chapter is based on a high-accuracy external positioning system (i.e. the Bat system) to recover the survey trajectory. With this groundtruthed survey trajectory, the recorded signal strengths can be aligned to the environment accurately. In real life, such a high-accuracy system is rarely available for the signal survey, and the survey device is typically a commodity smartphone with erroneous sensors. So accurate path recovery is difficult. If the survey path cannot be recovered accurately, the signal strength information cannot be aligned to the environments correctly, which causes errors in the signal maps and hence lowers the performance of the path survey. To solve this problem, this chapter proposes a high-efficiency and high-accuracy path survey system which uses a commodity smartphone as the survey tool. We test our system in a range of testbeds, prove that it offers a robust and fast easy way to survey an environment.

4.1 Introduction

Path surveys require that we estimate the trajectory of the device accurately. The estimated trajectory itself can be built using Pedestrian Dead Reckoning (PDR) algorithms applied to commodity inertial sensors found in smartphones. However, PDR algorithms drift quickly and techniques must be used to constrain the error. Recent work has successfully limited this drift by the application of constraints such as floor plans, or the application of Simultaneous Localisation and Mapping (SLAM) algorithms (introduced in more detail in Section 4.2).

There is also a question of whether the survey process is *dedicated* (whereby the device is purposely moved to build a fingerprint map, as per our case) or *opportunistic* (where the trajectories are crowdsourced as multiple devices move through the space for other purposes—see e.g. [40]). Although appealing, opportunistic surveying faces many challenges that have yet to be solved, including heterogeneity of devices, uncontrolled movements, changes in device orientation, etc.

In this chapter we introduce a SLAM-based dedicated surveying technique that uses the fast-varying magnetic field to constrain the trajectory, which is then used to generate maps of *other* signals such as WiFi/BLE. Our contributions over established work include an algorithm for sequence matching of magnetic signals rather than point matching; support for free movement of the surveyor; the use of magnetic signals to survey but other signals that are more amenable to regression and allow one-shot or push-to-fix positioning; and detailed experimental testing in five testbeds.

4.2 Related work

SLAM algorithms were originally developed in robotics to allow joint estimation of a robot's local map and its position or path within it [17]. Most approaches are designed with accurate robot odometer and precise sensors (usually laser scanners) in mind. In moving to a smartphone platform, accurate odometer is typically replaced with approximate relative motions from PDR [8] and accurate room scans by noisy signal measurement processes. For a path survey, as the user walks the smartphone collects signal scans and their path is estimated using PDR. To combat the inevitable drift that accrues in the PDR path a SLAM algorithm is used to correct the path. The process is usually considered in terms of a front-end and back-end; the former determines constraints between parts of the trajectory; the latter solves the system under these constraints. When used with PDR broadly two classes of SLAM algorithm may be used:

- **Full SLAM**. The SLAM engine has a model for the propagation of the signal source (e.g. a WiFi propagation model). The back-end jointly estimates the locations of the signal sources and the trajectory in the same space. Once done the signal fingerprint at any point (i.e. the map) can be estimated. This approach is very valuable when we have point signal sources (e.g. WiFi) but cannot be used on signals without point sources associated with specific measurements (e.g. magnetic fields).
- **Path smoothers**. The front-end derives constraints from *loop closures*. The idea is that, when the user revisits position $p(t_a)$ at time t_b, the system will observe the same (or very similar) signal measurements. Thus it can infer a loop has been travelled and it can infer the constraint that $p(t_a) = p(t_b)$. The back-end then estimates the path.

Based on the consideration that building a reliable radio (WiFi/BLE) propagation model for the complicated indoor environments is hard, we use the path smoother SLAM algorithm and the spatially-variant but temporally-stable magnetic field signal to infer loop closures. In the context of pedestrian surveying, this requires the user to repeat parts of their path, forming loops. For the purpose of path smoothing, many different SLAM algorithms exist: for pedestrian SLAM important approaches include those based on graph optimisation (e.g. [42]) and particle filtering (e.g. [18, 23]).

We are not the first to consider location determination using sequences of magnetic readings: both the LocateMe system [83] and IndoorAtlas [44] use a similar approach to us. Our work differs in a number of ways from LocateMe. Firstly, we use the magnetic sequence matching within a SLAM framework for surveying rather than for online positioning; secondly, we use different signals (WiFi, BLE, etc.) for online positioning; thirdly, we perform more comprehensive signal matching (LocateMe matches to a corridor while we show how to match to a co-ordinate position). The IndoorAtlas system is not openly described so is hard to compare to. However, from the limited information available in patents, similar differences apply.

The MagSLAM system proposed by Robertson et al. ([76]) uses magnetic magnitude within a SLAM framework. Their work differs in a number of ways. Firstly they use a particle filter implementation of SLAM (based on FootSLAM) applied to a foot-mounted Inertial Measurement Unit (IMU). This produces a much better PDR trajectory estimate since the drift can be constrained by the application of ZUPTs and ZARUs before SLAM is applied. Here we use an unconstrained smartphone as the sensor, producing a very noisy PDR input. Accounting for this noise in a particle filter is very difficult to do since the particle number must increase dramatically to account for it. Consequently we use a graph-based offline SLAM approach. MagSLAM also produces magnetic maps for subsequent positioning. In our experience such maps are not temporally or spatially stable and we advocate their use only during the survey—we discuss our reasoning in detail in the next section. Lastly, while we explicitly consider sequences of magnetic measurements, MagSLAM does so only implicitly. As a user walks and a distinctive sequence is observed, confidence will grow in the particles representing hypotheses following the correct path. We return to this presently.

4.3 Magnetic mapping but not positioning

Magnetic signals often vary fast indoors, which can allow for accurate detection of loop closure points. This makes it a very good signal for surveying with. However, we have found magnetic fingerprint maps to be a poor choice for online positioning for a number of reasons:

- **Ease of decalibration.** The magnetometers inside consumer smart devices are susceptible to noise from nearby electrical components and are affected by nearby ferrous materials (e.g. keys in a pocket). Manual recalibration is needed quite frequently to get reliable magnetic measurements.

- **Accurate positioning needs movement.** A given magnetic measurement is not spatially unique and does not vary predictably. Thus one-shot positioning (where we estimate position from a single measurement) is highly unreliable. Movement allows for sequences of measurements that are more spatially unique. Systems that implicitly consider sequences (e.g. MagSLAM) will struggle to initialise a position based only on a magnetic map.

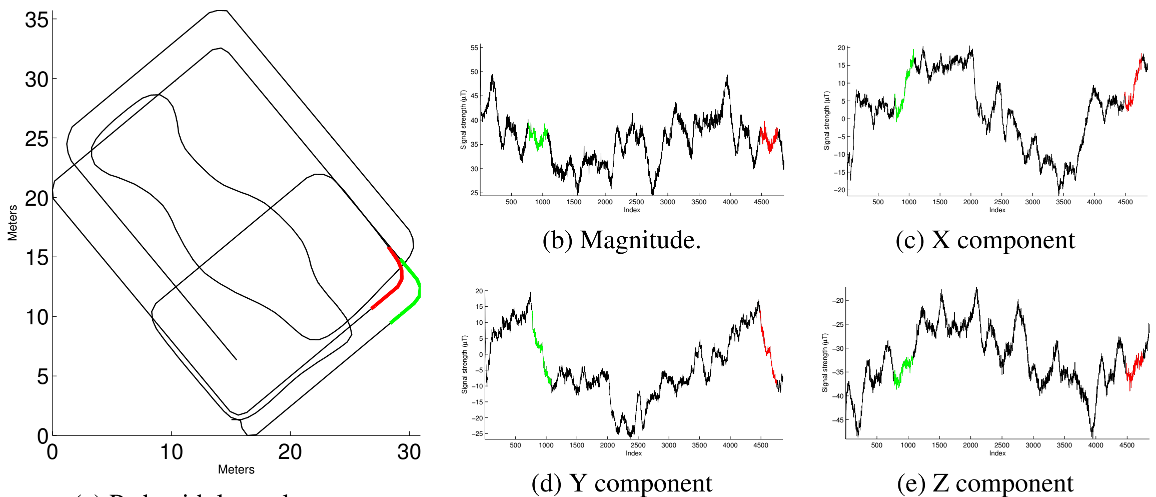
- **Limited regression suitability.** Generating 2D fingerprint maps from linear trajectories requires the use of regression. However, the fast-varying nature of magnetic signals means that it is impossible to predict the magnetic signal even a short distance from the trajectory (the typical magnetic coherence length is only 30 cm). Ultra high frequency (UHF) signals, by contrast, have a longer coherence length of around 3 m, allowing better regression.

- **Device orientation is rarely known.** In a general positioning scenario the device orientation is rarely known to any accuracy and may even vary during the positioning computations. Thus we are limited to using the scalar magnitude of the field, which introduces more location ambiguity.

- **Areas of low variability.** A typical indoor space will have subareas where the magnetic field is broadly uniform, especially when considering only magnetic field magnitude. Map-based positioning is not available in such areas.

4.4 Magnetic sequence-based loop detection

We use a path-smoother SLAM algorithm that depends on us providing exteroceptive constraints, most notably loop closures. We cannot reliably detect loops based on comparing the current magnetic fingerprint to those collected earlier in the survey since very disparate locations can exhibit very similar magnetic fingerprints. We argue that matching *sequences* of magnetic measurements rather than points can provide more robust localisation. To match, we take windows of the magnetic field vector and search for similar magnetic windows in the history of the walk. We use the full 3D magnetic vector since this provides further spatial locality (Figure 4.1), and *we can instruct a dedicated surveyor to hold the device in (approximately) the same orientation relative to their body.* Since the surveyor may traverse a path in either direction, we search both chronologically and reverse chronologically.



(a) Path with loop closure.
(b) Magnitude.
(c) X component
(d) Y component
(e) Z component

Figure 4.1: Example magnetic loop closure for both scalar and vector magnetic fields. The loop closure in Figure 4.1a consists of two segments marked in red and green respectively on the survey path. Figure 4.1b, 4.1c, 4.1d and 4.1e show the magnitude and vector magnetic fields collected along the full survey path, with magnetic sequences correspond to the loop closure marked in corresponding colours. This example shows the similarity between magnetic sequences for loop closure segments.

4.4.1 Matching algorithm

We use Dynamic Time Warping (DTW) to match sequences of magnetic vectors. Given two sequences, seq_a and seq_B (assuming seq_B is longer than seq_a), DTW finds the part of seq_B (denoted as seq_b) that best matches seq_a. Crucially it will compress or stretch one sequence to find the best match, which helps to mitigate speed changes as the surveyor walks. The variant of DTW used here is called unconstrained or open-begin-end DTW (OBE-DTW), for which full details are available in [32]. To reduce the computational load, we use a local matching strategy, whereby subsequences of the historical trajectory are selected based on their proximity to the last known position of the device (we used a range of 10 m)

4.4.2 Match validation

The DTW algorithm terminates having found the best match of the subsequence to the larger sequence. There is always a match found, albeit not always a good one. A typical survey walk of a few minutes therefore produces many hundreds of ‘matches’, many of which are false positives. These false positive loop closures can have catastrophic effects on the SLAM result hence need to be dealt with properly. In principle, a robust SLAM *back-end* (the part of a SLAM system responsible for optimising the output given the constraints) is supposed to have the ability to reject or deactivate false positive constraints [85, 84, 70, 57, 56, 11]. However, when a large proportion of constraints are false positive, the back-end can become confused. Therefore, a robust front-end that can reject enough false positive loop closures is important. Here we focus on the loop closure validation in the front-end. We filter the DTW output to identify the matches that are accurate and consistent (these hopefully represent valid loop closures). We prove by experiments that by filtering out false positive loop closures in the front-end, a normal SLAM back-end is able to give good SLAM result. But we should still note that it could definitely increase the system robustness if robust SLAM back-end is used together with our front-end validation method (though higher computational cost is needed).

To validate matches found by the DTW, we apply five simple but effective empirical criteria as follows:

1. The **normalised distance between matches must be less than 10** μT. This quantity is computed as the sum of the distances between the aligned elements normalised by the number of elements [32].

2. The **compression ratio must be less than 5**. for two matching sequences, seq_a and seq_b, of lengths n_a and n_b samples, the compression ratio is $C_r = \frac{max(n_a,n_b)}{min(n_a,n_b)}$. It represents how stretched or compressed DTW has made a sequence to match it—we expect minimal stretching or compression corresponding to minimal speed changes.

3. The **temporal distance must be greater than 10 s**. Two consecutive sequences are likely to generate false positive matches. We impose a minimum temporal separation between the first measurements of each proposed matching sequence pair.

4. The **physical distance must be within 10 m**. If a match is found for two sequences physically close in the PDR trajectory, there is a greater chance that the match is valid.

5. The **segment topology should be similar**. The spatial shape of the matched sequences in the PDR trace should also match. For two matched sequences seq_a and seq_b returned by DTW, we have spatial co-ordinates X_A, Y_A, X_B, Y_B. We form two new sequences, $seq_{diffX} = X_A - X_B$, $seq_{diffY} = Y_A - Y_B$ and use the variances of these two sequences to indicate similarity. We use an empirical threshold of 0.5 m^2.

Many DTW schemes filter based on criteria 1 and 2 above. However, these are insufficient for magnetic sequence matching—Figure 4.2 provides an example where criteria 1 and 2 are met since the magnetic waveforms are similar. In this example, the topology criterion would cause the match to be discarded.

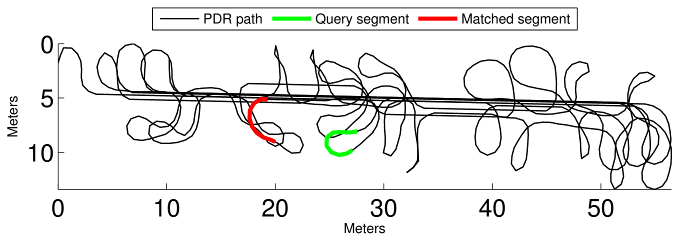
One thing to note is that the proposed topology comparison method (the fifth criteria proposed above) is a simple linear algorithm. There are more advanced methods which could achieve the same goal. For example, Horn et al. [39] proposed a closed-form solution to compute the optimal rigid-body transformation T (consists of rotation and translation) and the scale s that best transform one set of point to match another corresponding point set. Then the topological similarity can be inferred from s and the magnitude of T. Similar methods to assess topological similarity of two segments include the Iterative Closest Point (ICP) [90]. The loop closure validation could be more robust by the use of these more advanced methods, but the computational cost would be higher. For the ease of implementation, we adopted the proposed low-cost method which works well by our experiments.

4.5 A magnetic surveying system

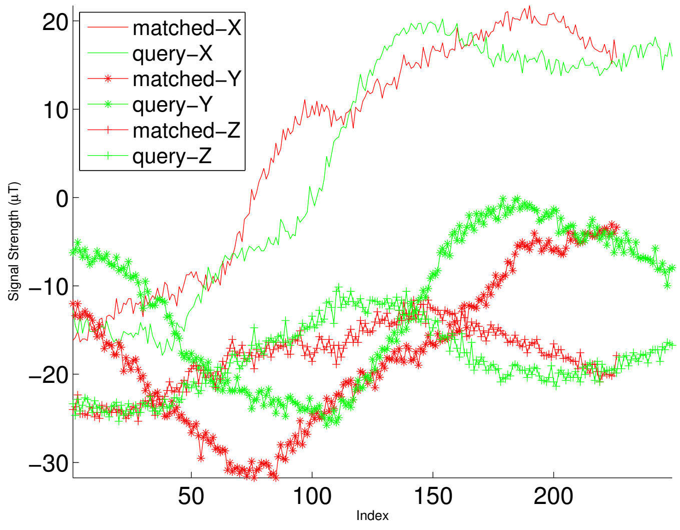
We have incorporated the loop detection and validation algorithms in a system that operates in two modes: *survey* (in which the signal maps are built by a dedicated surveyor) and *positioning* (in which a device is positioned using the surveyed maps). A block diagram of the key survey components is shown in Figure 4.3, each of which is described in this section.

4.5.1 PDR

The PDR component provides a trajectory estimate based on dead reckoning algorithms applied to the accelerometer and gyroscope sensors. Many factors that affect the performance of PDR, most notably how and with what consistency the user holds the phone. In the scenario we propose, PDR is only used during the surveying process, which is carried out by a dedicated surveyor. It is therefore



(a) The surveyor's trajectory with an incorrect sequence match highlighted.



(b) The magnetic waveform. The green and red sequences correspond to the coloured segments in 4.2a

Figure 4.2: An example of a false loop closure. The normalised DTW distance is 3.53 μT with a compression ratio of 2.09.

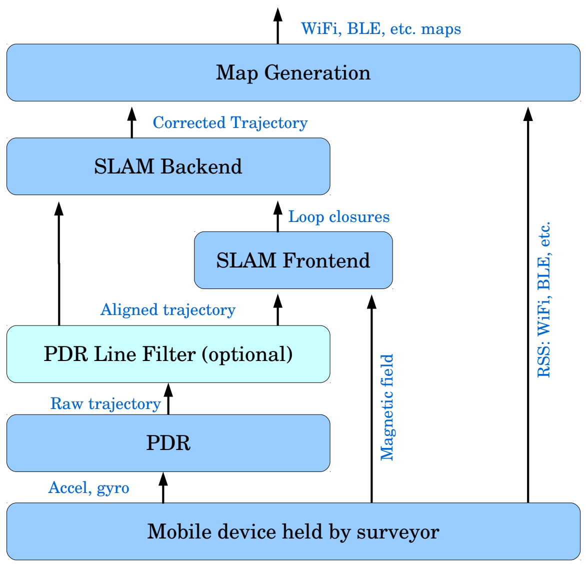


Figure 4.3: System architecture.

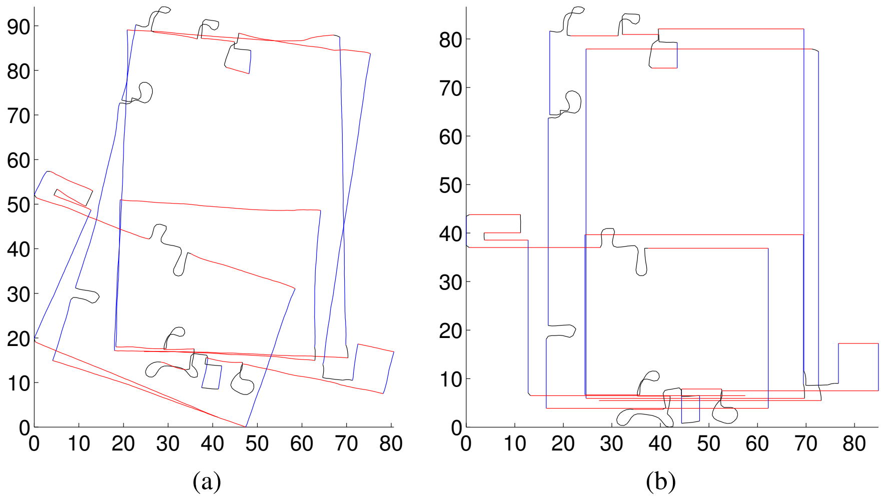


Figure 4.4: The value of the PDR line filter. (a) The raw PDR trajectory. (b) the filtered trajectory, which is significantly closer to the true path taken.

reasonable to assert the phone is held in a convenient location—we assume it is held out in front as if navigating, and the user's heading matches that of the phone. Under these circumstances even simple PDR algorithms can be used successfully [8][1]. We take a representative and commonly-used PDR algorithm based on thresholding the magnitude of the accelerometer (as described in [8]). Figure 4.4a shows an example PDR output.

4.5.2 PDR line filter (optional)

Humans tend to walk in straight lines wherever possible. However, in this work we rely on gyroscopes for orientation tracking and these typically exhibit bias errors that are only partially compensated for in smartphones. These bias errors cause estimated trajectories to bend incorrectly, which is particularly problematic for the higher-level algorithms we apply. This component identifies likely straight-line segments and filters them to be straight. The filter works by looking at the difference in heading between consecutive PDR steps (the *turning angle*):

1. Loop over all steps in PDR result, grouping consecutive steps with a turning angle of less than 3°. Flag each group with more than 15 steps as a straight line.

2. Reset the the turning angle of each step within a straight line group to zero.

3. (Optional, if the building is rectangular). Set the heading of the first straight line as a reference orientation. Loop from the second straight line, if the difference of the orientation with the first straight line is within P_{thresh}=15°, then assume this straight line is parallel to the first line; if the orientation difference is within [90-P_{thresh}, 90+P_{thresh}], then assume this line is perpendicular to the first line and change the line orientation accordingly.

Figure 4.4 illustrates the value of the PDR line filter on a typical rectangular building. For this kind of survey path which consists of many long straight lines, the proposed line filter is necessary, because

[1]Please note that there is a growing body of research on PDR aimed at relaxing the assumption about device placement. Our proposed system can use these more advanced PDR algorithms as inputs. However, the stability and robustness of these PDR algorithms in uncontrolled crowd-sourcing contexts has yet to be proven. We choose simple PDR algorithms and ensure their robustness by constraining the device usage in order to focus on the higher levels of the system.

the SLAM back-end algorithm does not have sufficient information to explicitly correct lines which are wrongly bent (it simply “moves” lines towards each other if implied by a given loop closure). But for the survey path consists of mostly curved lines (e.g. the one shown in Figure 4.5a), applying this line filter or not causes no significant difference in the final results. Also, the proposed line filter would straighten *any* shallow curve in the path, even those that were genuinely curves. However, it is rare to find buildings with curved corridors so this problem is minor (the line straightening component can be omitted entirely in buildings with curved paths). Therefore, we decide to apply the PDR line filter or not on a case-by-case basis.

4.5.3 SLAM front-end and back-end

SLAM frameworks are typically split into two tasks: the *front-end* (which identifies constraints) and the *back-end* (which optimises the system under those constraints). We use the GraphSLAM (or graph-based SLAM) back-end to optimise the entire trajectory based on all loop closures [86]. GraphSLAM does this using a pose graph. The front-end builds the graph where each node in the graph corresponds to a position estimate for a given time step. Consecutive steps correspond to nodes linked by edges that represent the constraints of the step itself (length, turning angle). Loop constraints determined as in Section 4.4 are added by connecting the two nodes where the loop occurs by an edge. The GraphSLAM back-end optimises the spatial configuration of the nodes to best satisfy the constraints.

Many modern implementations of GraphSLAM adopt a least-squares error minimisation technique in the back-end [59]. To cope with the high frequency of magnetic measurements (which leads to very large matrices to be solved) we adopted a stochastic gradient-descent GraphSLAM (SGD-SLAM) [71, 72]. SGD-SLAM solves a single constraint at a time instead of trying to solve all at once. This breaks the problem into many tiny problems and speeds up the process. Figure 4.5 gives two examples of the front-end and back-end outputs.

4.5.4 Map generation

Fingerprint maps were generated using Gaussian Processes regression applied to the SLAM trajectory estimate and the signals measured en-route. Each location was associated with a signal mean and variance; this is illustrated in Figure 4.6. Note that the variance climbs steeply within a few metres off the trajectory, correctly capturing the high uncertainty that comes from having no measurements there.

4.5.5 Positioning

As in Section 3.5.1, a standard Bayesian localisation algorithm was used to position based on the signal maps, which is briefly repeated here. To estimate the person’s location, x, conditioned on observation of a set of signal strength measurements (fingerprints), z, we used:

$$p(x|z) \propto p(x)p(z|x) \tag{4.1}$$

where $p(x)$ is the prior probability of the person’s location and $p(z|x)$ is the likelihood of observing a set of signal strength measurements at location x. The signal map generated from the Gaussian Processes regression was used to estimate $p(z|x)$ as described in [25].

In practice we divided the environment into a grid of square cells of length 1 m or less, For each step at each cell we computed $p(x|z)$ and selected the cell with the maximum as the person’s location. Signal strength measurements lower than -90 dBm were discarded. We tested two priors: a uniform

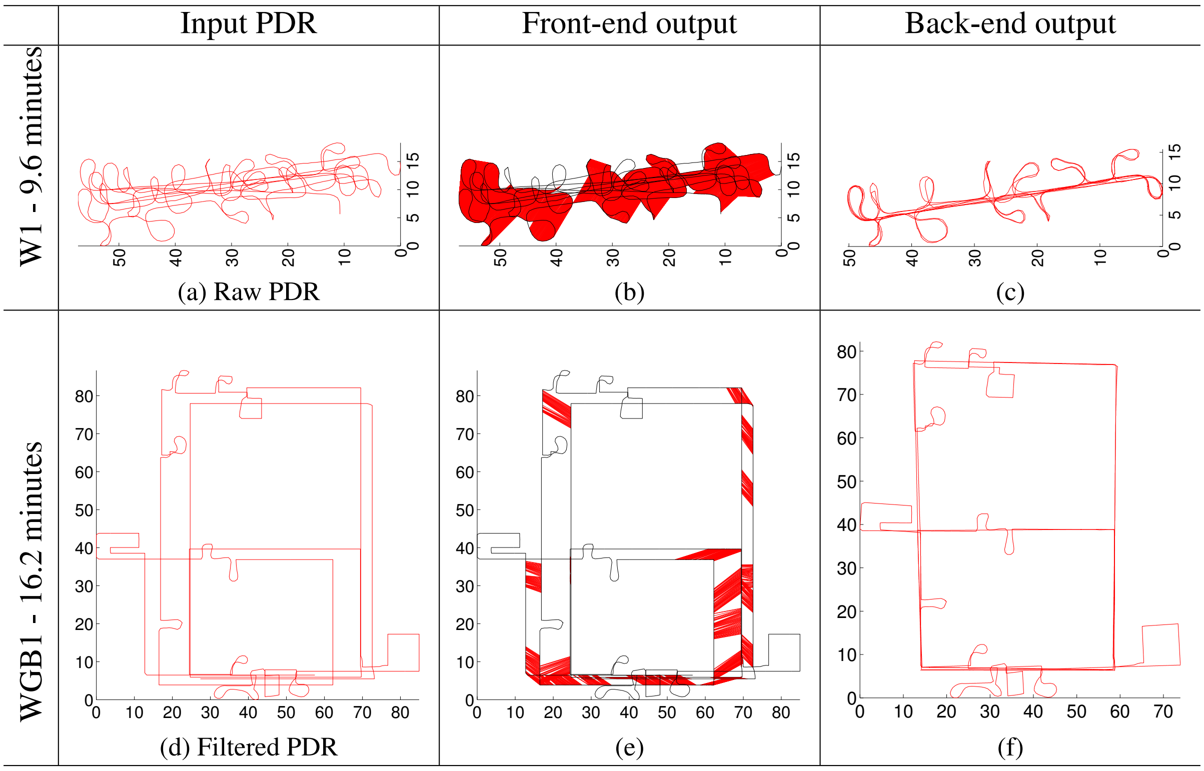


Figure 4.5: Sample SLAM component outputs. Each red lines in the middle column indicates a selected loop closures.

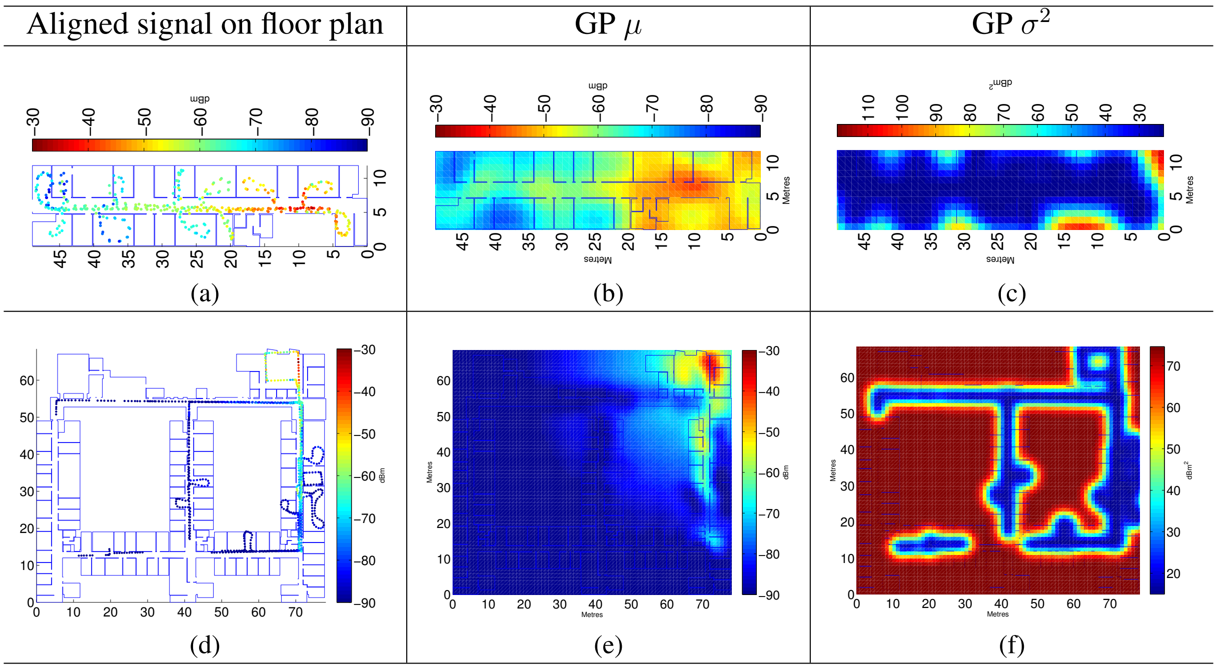


Figure 4.6: Sample WiFi Gaussian Processes maps from survey walks.

distribution across cells (representing one-shot or push-to-fix positioning); and the previous posterior distribution (representing a tracking scenario). As expected, the tracking prior gave a slightly higher accuracy, although the difference in error was rarely less than 1 m. For brevity we focus on the results of the uniform prior since this assesses the performance of the system in the most difficult scenario.

4.6 Testbeds

We used a number of different testbeds to assess our survey system:

1. **WGB1**. The first floor of the William Gates Building at the University of Cambridge. This is an office environment with long corridors, small office rooms and large lecture halls.
2. **WGB2**. The second floor of the William Gates Building with long corridors connecting individual office as per WGB1.
3. **ENG**. A large office area in the Engineering Department at the University of Cambridge. It featured a large open space and computers scattered throughout.
4. **KX.** The concourse in London King's Cross train station. A very large, roughly rectangular open space with lots of people moving through it.
5. **RUTH**. The Rutherford building at the University of Cambridge. A concrete building featuring corridors and (restricted access) laboratories,

One wing of the WGB2 testbed (**WGB2a**) was instrumented with the Bat systems [1] that allowed for accurate trajectory ground truth in order to assess the system more quantitatively. Additionally, the manual survey of the WiFi and BLE signals in all non-restricted areas of WGB2a from the previous chapter was available. We also use the walks W1–W4 as described in Section 3.3: W1 visited all non-restricted rooms; W2 was a simple walk up and down the corridor (only); W3 traversed the corridor (only) multiple times; and W4 visited only a subset of the non-restricted rooms.

For the remaining testbeds, ground truth was obtained in two ways. The first involved a series of checkpoints that the surveyor logged as they passed them. The second involved manually matching turning points in the trajectory to known locations on a floor plan. Neither approach gives an accuracy comparable to that available in the WGB2a testbed, but the results from different testbeds ensure the results have more general relevance.

When testing, the surveyor walked continuously with an Android smartphone held out as if navigating. The smartphone recorded the inertial sensor outputs and the Received Signal Strengths (RSS) of WiFi and Bluetooth Low Energy (BLE) signals. All recorded data were then transferred to a desktop machine for offline processing. For a practical system, we do not envisage running the various optimisation algorithms on mobile devices, but rather offloading such computations to the cloud—this approach has worked well for similar systems e.g. WiFiSLAM. For all results presented here, the survey was carried out by an author, in line with our belief that the survey should be carried out by a dedicated surveyor fully aware of the how the system works.

4.6.1 Floor plans

Accurate digital floor plans were available for WGB1, WGB2 and WGB2a, but not for the other testbeds. The surveying system does not rely on such floor plans, although they can help with visualisation and context derivation for the subsequent positioning service. For visualisation in this work we manually align and scale our outputs to bitmaps of the relevant floor plan if a digital version is unavailable. In general such bitmaps might be obtained by photographing the fire evacuation sign or similar.

4.7 Evaluation

4.7.1 SLAM trajectory accuracy

We assess the SLAM accuracy within testbed WGB2a (walks W1–W4) since it provided an accurate ground truth trajectory. Figures 4.5a, 4.5b, and 4.5c show the outputs of the various stages of the system for W1 and Figure 4.7 shows the result for W2–W4. In Figure 4.7, the first column shows the raw PDR output; the second shows the line-filtered output; the third highlights the selected loop closures in red; and the fourth shows the final trajectory, which can be compared with the ground truth in the fifth column. Subjectively we observe that the final trajectory is significantly better than the original input. To assess more quantitatively we use two metrics commonly found in the robotics literature:

- **Absolute Error (ABS)**. This is the distance between the estimated position and the ground truth position at each epoch. It requires a manual alignment of the SLAM trajectory to the floor plan.
- **Subjective-Objective Error (SO)**. This metric aims to characterise how well the SLAM trajectory reproduces the loop closures in the ground truth. In essence it is the average distance between points on the SLAM trajectory that match to loop closure points on the ground truth trajectory. Thus a perfect SLAM trajectory would have an SO error close to 0 m, and increasing values indicate a worse SLAM result. More complete definitions can be found in [7, 42].

Table 4.1: WGB2a SLAM Accuracy Summary.

	ABS (m,m^2) (mean, variance)	SO (m,m^2) (mean, variance)
PDR	N/A	(1.27,5.78)
SLAM	(0.73,0.18)	(0.22,0.15)

Table 4.1 shows the aggregate values of these metrics for the test walks, whilst Figure 4.8 gives the CDF of the values. As expected from visual inspection, the raw PDR trace has high error[2]. The SLAM output has significantly reduced errors, validating the use of the magnetic sequence matching for trajectory correction even on consumer-grade smartphones. Please also note that our magnetic SLAM accuracy (0.22 m and 0.15 m^2 for mean and variance respectively) also outperforms the accuracy of the state-of-the-art *WiFi GraphSLAM* algorithm (2.23 m and 1.56 m^2 for mean and variance respectively) which is reported in [42].

[2]ABS error for raw PDR was not computed since it cannot be unambiguously aligned to the ground truth.

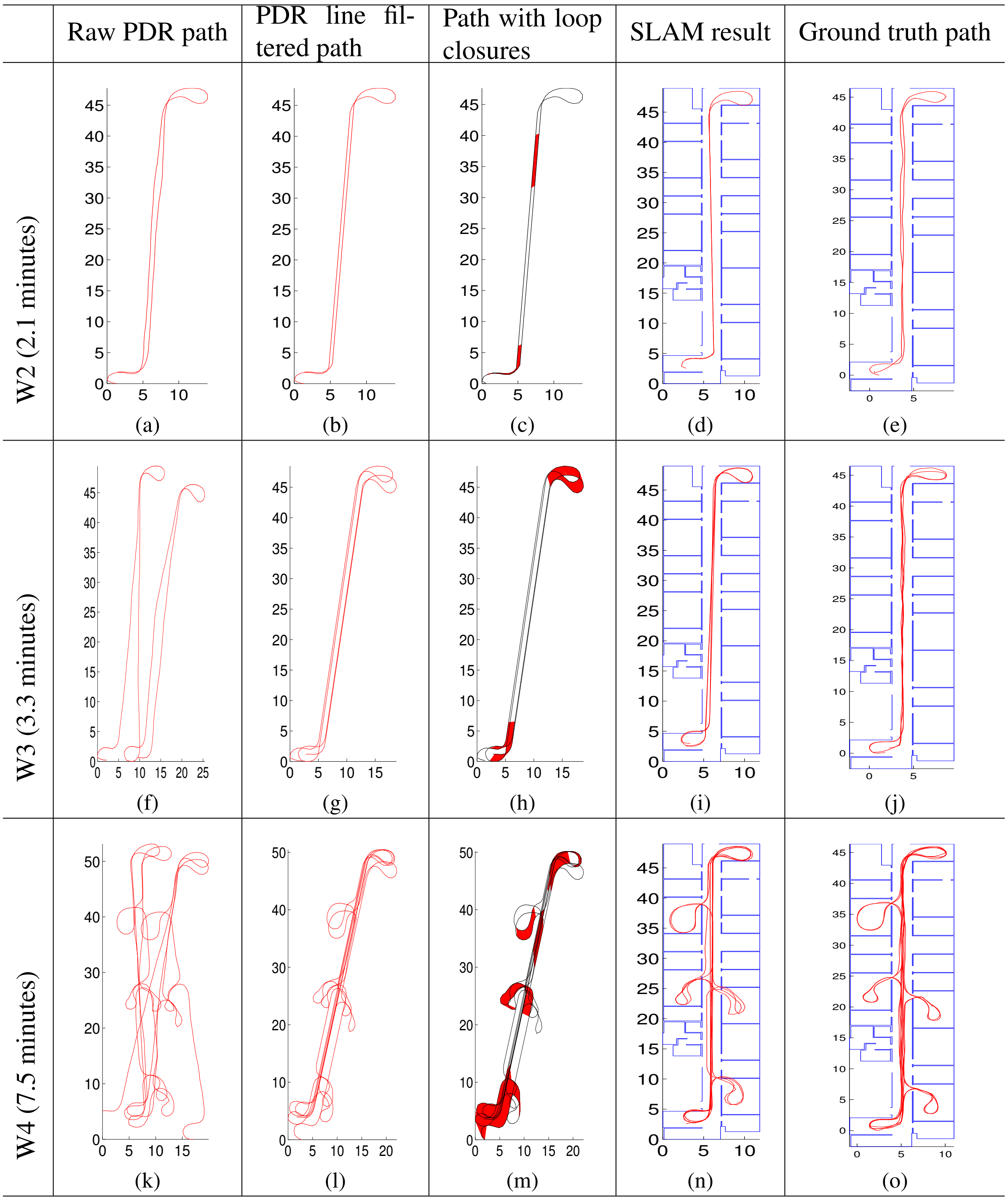


Figure 4.7: Result of magnetic SLAM. X-axis and Y-axis are coordinates measured in metres. Note that the results were aligned to the floor plan manually.

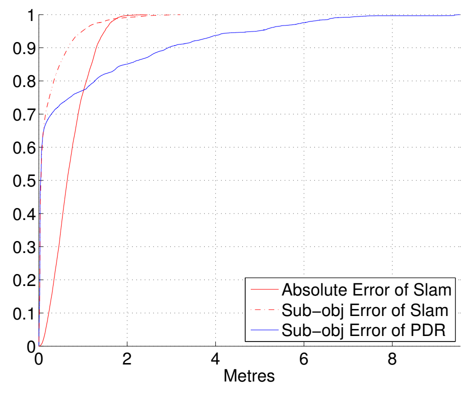


Figure 4.8: SLAM accuracy for WGB2a testbed.

4.7.2 Positioning accuracy

In this subsection we quantitatively assess the positioning accuracy achieved in each testbed using the auto-generated signal maps. We present results for 'push-to-fix' positioning (i.e. each position estimate was based on a uniform prior) since this represents the most challenging case for indoor location systems, and something that cannot be achieved using magnetic signal maps.

Note that the positioning accuracy is a function of the map quality, the user's movement pattern, the positioning algorithm in use, and many other factors. As such, the previous SLAM accuracy metrics are arguably the fairest assessment of the magnetic SLAM performance. However, the positioning accuracy metric is well understood and all that was available in the other testbeds.

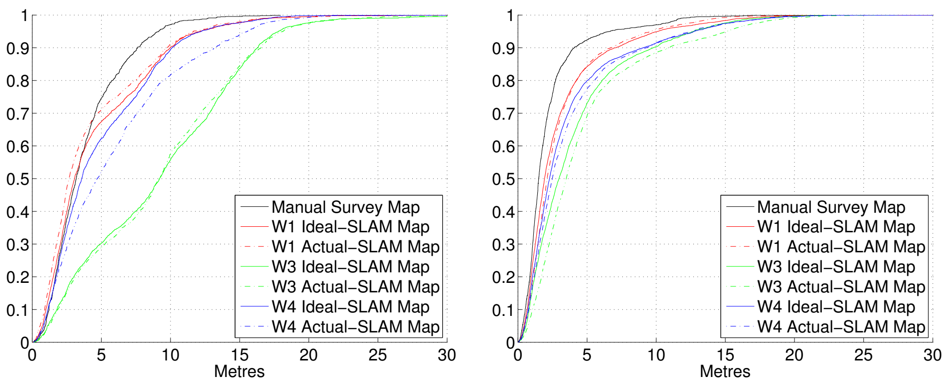
4.7.2.1 WGB2a

The ground truth and manual survey available for testbed WGB2a allowed for a detailed positioning accuracy assessment. We present results from three types of map: a **Manual-survey** map created from the data in the manual survey only; **Ideal-SLAM** maps created from the high accuracy ground truth trajectories of W1, W3, and W4[3] (representing the best possible result that could come from SLAM); and **Actual-SLAM** maps created from the SLAM trajectory output for survey walk W1, W3 and W4 by the system described in this chapter.

For a baseline we used the fingerprints observed for survey walks W1, W2, W3 and W4 to position using the manual-survey map. This represents the positioning accuracy that would have been achieved on those walks if the manual survey was used. To evaluate the walks directly, we used the measurements taken during the manual survey as input to be matched to the SLAM-derived maps. The results for both Ideal-SLAM and Actual-SLAM are shown in Figure 4.9.

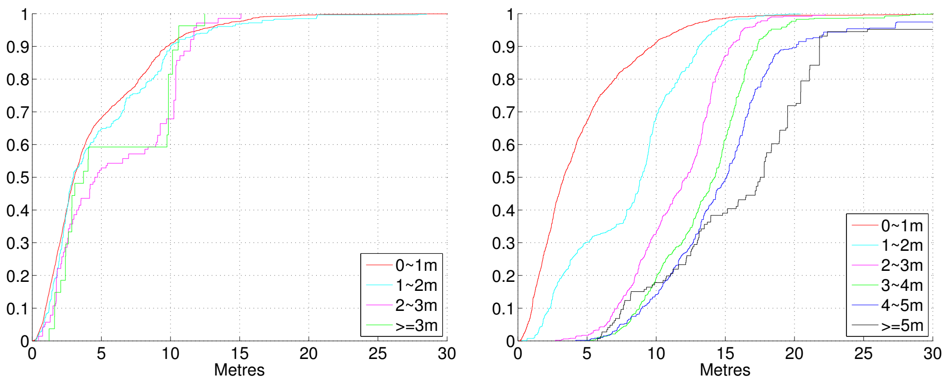
For both WiFi and BLE maps, we observe that the manual map provided the best positioning accuracy (7.6 m and 4.1 m, 90^{th} percentile respectively). The W1 maps performed better than the W4 maps, which performed better than the W3 maps. This is as expected: W1 visited every room in the manual survey, whilst W4 only visited a subset and W3 only visited the corridor. Thus W3 required greater extrapolation since there were more test points further away from the survey path, and hence more positioning occurring in regions of low map confidence. Figure 4.10 demonstrates this by classifying the errors by their distance from the survey trajectory. For the W3 map (Figure 4.10b), we see that the test points near the trajectory (i.e. along the corridor) gave similar error profiles to those for the W1 map (around 10 m, 90^{th} percentile). However, there are a large number of test points away

[3]The short duration of walk W2 meant very few WiFi scans were completed, so it is hard to build a meaningful map with good signal prediction quality.



(a) WiFi (b) BLE

Figure 4.9: Positioning errors in the WGB2a testbed.



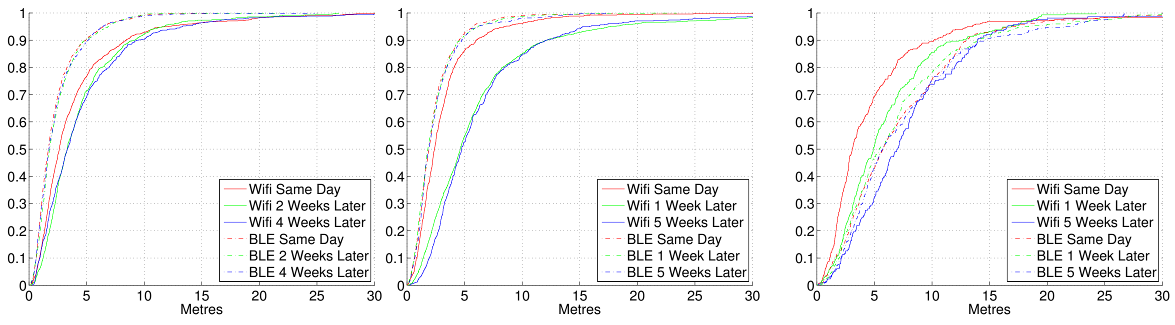
(a) W1 (b) W3

Figure 4.10: The WiFi positioning CDF results for Ideal-SLAM maps created from W1 and W3 broken down by distance of test point from the survey trajectory.

from the trajectory, and these exhibit far higher errors (around 20 m, 90th percentile). When taken as a whole, the error profile in Figure 4.9a is thus much worse for W3.

The Ideal-SLAM and Actual-SLAM lines in Figure 4.9 are generally very similar, with the Ideal-SLAM performing marginally better. This is further evidence that the magnetic SLAM scheme we describe in this work is able to approximate the trajectory to a high accuracy. It is interesting to note that even the Ideal-SLAM maps, with their 'perfect' trajectories did not reproduce the manual map accuracy. We attribute this to three factors: the first is that the inputs to the manual-survey map are distributed regularly and comprehensively through the environment and regression further than a metre from a survey point was not necessary; the second is that more data were collected at each of the manual survey points, allowing a better picture of the variation to be built there (to emphasise this point, the manual survey was built from 5847 WiFi scans, whilst W1, the longest of the survey walks, used only 485 scans); and the third is that the Ideal-SLAM maps are evaluated against fingerprints generated from manual survey data, most of which were taken 1$\sim$2 meters or further away from the survey path.

Overall, these results confirm the need for the trajectory to fully explore the space, by which we mean it must pass within a few metres off any given point where positioning might be required. Crucially, the generated maps within this range of the trajectory are a good approximation of the manual survey maps. Whilst better accuracy can always be achieved from a detailed manual survey, this must be weighed against the time taken to survey. For context the WGB2a manual and walk



(a) WGB1 (b) WGB2: Dense signal zone (c) WGB2: Sparse signal zone

Figure 4.11: Positioning errors in WGB1 and WGB2.

survey durations are given in Table 3.2.

4.7.2.2 WGB1 and WGB2

For each of these larger testbeds we generated a signal map from a single survey walk and tested its positioning performance on a test walk. To see how the maps degraded over time, the WGB1 test walk was repeated two and four weeks later, and the WGB2 test walk one and five weeks later.

The results for WGB1 are shown in Figure 4.11a. They show a small degradation over time, suggesting that a monthly resurvey would be appropriate. The degradation is more noticeable for WiFi than BLE positioning. We attribute this to the much higher density of BLE beacons than WiFi access points, making the positioning more resilient (and of higher accuracy in general). We emphasise this trend by splitting the WGB2 results into errors for a subarea with dense WiFi and BLE coverage (Figures 4.11b) and one with sparse coverage (Figure 4.11c). For both WiFi and BLE the positioning accuracy falls with increased sparsity—indeed, the BLE coverage is so poor that WiFi positioning generally outperformed BLE in the sparse area. Again we note the temporal degradation of the accuracy, although there is now a marked fall in WiFi accuracy within a week of the survey. We attribute this to a physical change to the access point, based on three observations. Firstly, the BLE positioning in the same area was unaffected; secondly, the positioning performance after 1 and 5 weeks is very similar; and thirdly we have not seen this behaviour before.

Given both WGB1 and WGB2 are typical office environments, it seems reasonable from these results that dedicated signal surveys could be carried out every one or two months. This seems like a feasible demand given that a survey is non-specialist and takes only a few minutes.

4.7.2.3 ENG, KX, and RUTH

The positioning results for the remaining testbeds are based on WiFi measurements collected on a survey walk followed by a test walk. Sample results are shown in Figure 4.12, with the error CDFs in Figure 4.13. Unlike the WGB1 and WGB2 testbeds, the survey paths did not cover the space completely. Therefore we would expect areas of good quality positioning (near the trajectory) and areas of poor quality (away from it). When such variable quality is present and we do not uniformly test the space, the error CDF is highly dependent on the test path taken. For example, if we were only to test along the survey path, we would observe only low errors. Consequently we present CDFs for groupings of test points based on their distance from the survey trajectory.

Overall we see that the office-like environment of RUTH exhibited slightly better accuracies than the WGB1 and WGB2 test data. We attribute this to a combination of factors (e.g. the building

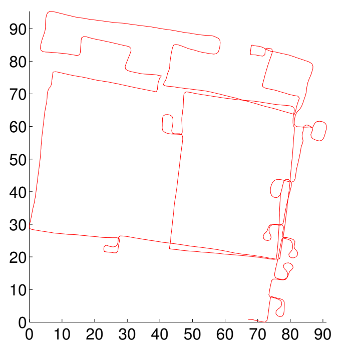
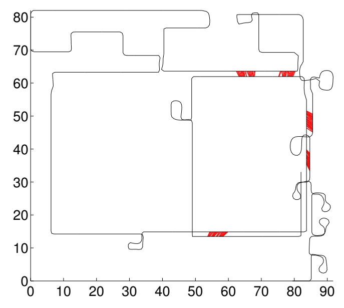
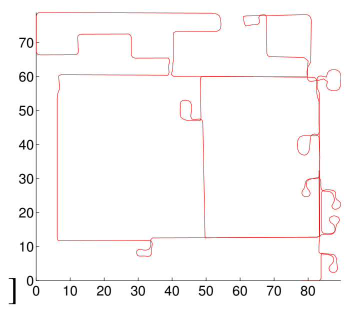
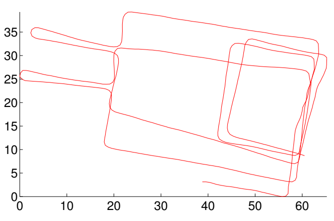
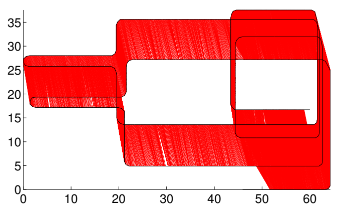
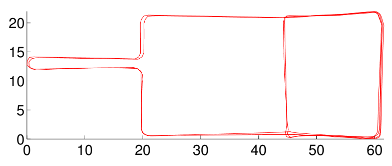
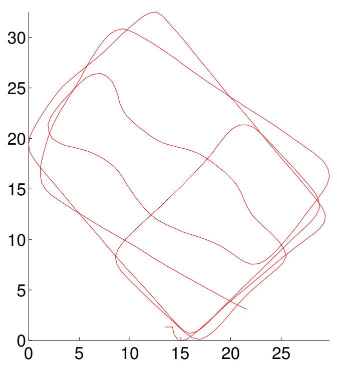
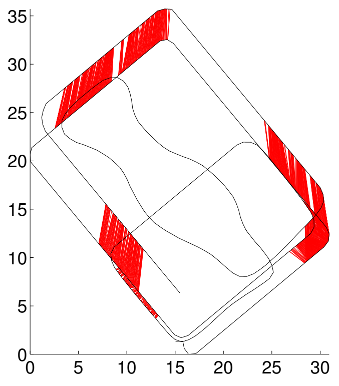
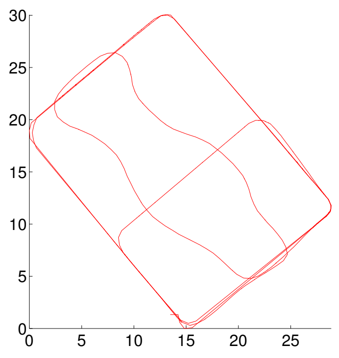
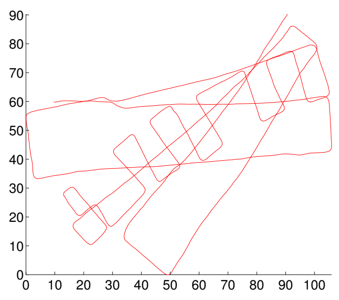
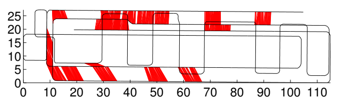
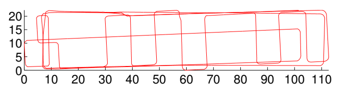
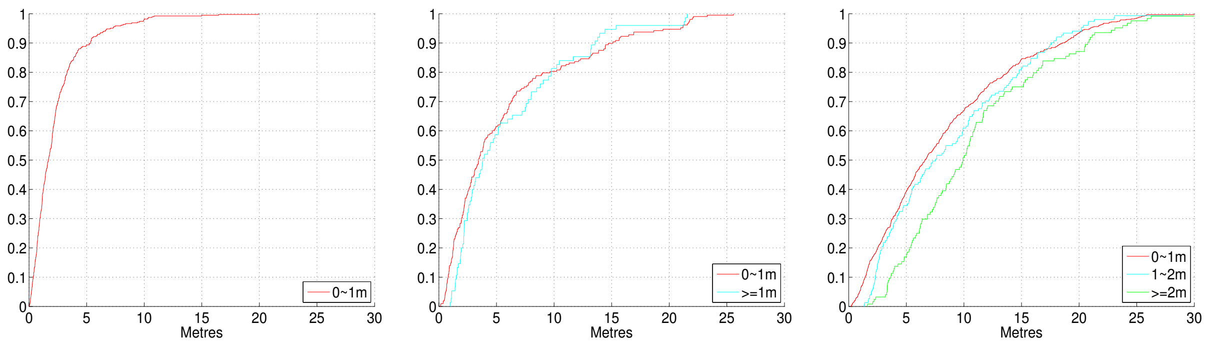
	Raw path	Filtered path with loop closures	SLAM result
WGB2 (12.6 mins)	 (a)	 (b)	 (c)
Ruth (6.6 mins)	 (d)	 (e)	 (f)
Eng (4.2 mins)	 (g)	 (h)	 (i)
KX (13.6 mins)	 (j)	 (k)	 (l)

Figure 4.12: Result of magnetic SLAM. X-axis and Y-axis are coordinates measured in meters.



(a) RUTH (b) ENG (c) KX

Figure 4.13: Error CDFs for RUTH, ENG and KX, separated by distance of test points from the survey trajectory.

construction materials), but also the less accurate (and less frequent) ground truth may have masked some slightly larger errors.

The errors for ENG and KX are notably worse. We attribute this to both testbeds containing large, open space with relatively sparse WiFi coverage near the centre (in such large spaces, WiFi is inevitably deployed to give comprehensive coverage with minimal overlap of base station ranges). In the case of KX, it is also a particularly dynamic environment which does not lead to the radio signal stability desirable for map-based positioning. Nonetheless, the accuracy is around 20 m at the 95% confidence level, which still provides useful positioning in such a large area (approximately 30 m by 125 m). Given that regular manual signal surveying is not possible in such an area, this accuracy is highly valuable, especially since it derives from a simple 13.6 minute walk.

4.8 Conclusions and further work

This chapter presented a robust, time-efficient scheme for signal surveying using a dedicated surveyor walking around a space with a consumer-level mobile device such as a smartphone. Specifically we found:

- Repeated parts of a trajectory (loops) can be accurately detected by comparing *sequences* of magnetic measurements. We have described a robust technique to detect and validate loop closures using this signal.

- A SLAM scheme can be used to estimate the trajectory from a Pedestrian Dead Reckoned (PDR) input from a smartphone accelerometer and gyroscope. Using a testbed with high accuracy ground truth available, we showed a point on the estimated trajectory has an average error of 73 cm, with a subjective-objective error under 0.22 m.

- Gaussian Processes regression applied to radio signals observed during the survey can provide good positioning results within a few metres off the trajectory, but not beyond. The errors are greater than if the map were constructed using a detailed manual survey (around 2 m at the 90% confidence level for WiFi and push-to-fix positioning). However this loss in accuracy is offset by the much shorter survey time (a matter of minutes rather than hours). In many environments a detailed manual survey is not a realistic possibility as a one-off event, never mind a regular one.

We have evaluated our survey system in five different environments, from which we make the following recommendations for the best results:

- A *dedicated* surveyor should be used to ensure trustworthy PDR and to allow the use of the magnetic vector (not just the scalar magnitude) for loop detection.
- Signals other than the magnetic signal should be favoured for the ultimate positioning. This is because magnetic signals can change quickly and they have poor locality (to the extent they cannot reliably support push-to-fix positioning)
- The survey trajectory should enter into every room and pass within 3–4 m of everywhere (or 1–2 m for the best performance) that a position may be requested. Regression schemes cannot reliably predict the fingerprints beyond this range.

Once a trusted fingerprint map has been constructed, it may be possible to update or refine the map using crowdsourced PDR traces. There are many challenges inherent in this, not least that people tend to avoid the loops necessary to constrain the PDR drift. We hope to investigate this in detail in the future.

Chapter 5

Optimised graphslam back-end

5.1 Introduction

The previous chapter introduced a high-efficiency and high-accuracy path surveying system which only uses a commodity smartphone as the survey tool. This system was tested in a range of testbeds, proving that it offers a robust and fast way to survey an environment. In this context, the noisy trajectory was recovered by a state-of-the-art PDR algorithm and pre-processed by a line filter (optionally). The loop closures were detected by matching magnetic signal sequences using a DTW algorithm. False positive loop closures were filtered out by the validation strategy described before. The filtered trajectories and validated loop closures were then processed by another critical component of this system, the SLAM back-end. This back-end takes the noisy trajectory and the loop closures as input, and generates a corrected trajectory that is the most consistent with all the loop closures. The problem the back-end solves is the well studied GraphSLAM (graph-based SLAM) problem in the area of robotics [86]. GraphSLAM optimises the trajectory according to all the available information from motion sensors and observations (loop closures)[1]. It does this using a *pose graph*. The nodes of the graph are formed by the poses (i.e. positions in the environment) of the survey device for every magnetic signal scan during the survey. The PDR outputs (steps, step length and turning angles) give an initial estimate of the poses. Consecutive nodes are linked by edges that represent the constraints between them (distance, turning angle). These constraints are called odometry constraints in robotics. Loop closure constraints (determined as described in Section 4.4) are added to the pose graph as edges connecting corresponding nodes where the loop occurs. This kind of edge represents a transformation with zero translation and zero turning angle, which means the two nodes should be in the same physical position in the space, i.e. they should have same pose values (x,y and heading). The GraphSLAM back-end optimises the spatial configuration of all the nodes to best satisfy the odometry constraints and loop closure constraints.

Many modern implementations of GraphSLAM adopt a least-squares error minimisation technique [59]. This technique requires to iteratively solve a linear system whose size is proportional to the number of nodes (poses). When the number of nodes is large (more than thousands), the time needed to solve the linear system becomes the bottleneck of this algorithm. Because the magnetic scanning frequency is high ($50 \sim 100$ Hz), a 10-minute walk can generate a graph with 30,000 to 60,000 nodes, leading to very large matrices to solve, so an efficient and scalable GraphSLAM al-

[1]Please note that we can alternatively treat the trajectory correction problem as a filtering problem and solve it using the algorithms like a particle filter. However, without enough environmental constraints (e.g. a floor plan), it is difficult to represent the posterior of the system efficiently (particles may grow unboundedly). So we choose to treat the trajectory correction problem as an optimisation problem and use GraphSLAM to solve it. We return to the alternative approach in the next chapter.

gorithm is necessary. This chapter first evaluates three state-of-the-art GraphSLAM algorithms (g2o, Toro, SGD) using the datasets from the automated surveying system. The evaluation results shows that the SGD algorithm outperforms others in this particular scenario. It adopts a stochastic gradient-descent method [71, 72], solving only a single constraint per time instead of trying to solve all at once. This strategy breaks the problem into many tiny problems and speeds up the process. However, SGD is not optimised for this use case. We show how it can be improved so that it solves our problems more efficiently. At the end of this chapter, we evaluate the improved SGD algorithm (MSGD) using the same datasets to demonstrate its value.

5.2 Related work

Graph-based optimisation has been studied intensely in the area of robotics. Many algorithms utilise the sparsity of the *information matrix* [2] of the pose graph to efficiently optimise the graph [87, 20]. However, they irrevocably introduce linearisation error that can lead to poor estimates. Lu and Milios [59] suggested a brute-force nonlinear least squares implementation to optimise the pose graph. However, this method has unaffordable time complexity in real applications. Many more efficient algorithms have been proposed since then. For example, some algorithms introduced Gauss-Seidel relaxation to speed up the graph optimisation [16, 29]. Olson et al. [71] introduced the stochastic gradient descent method which was originally used to train artificial neural networks into GraphSLAM. Their method (denoted as SGD herein) did not try to solve the large linear system formed by all the constraints in the graph directly. Instead, it attempted to solve only one constraint per time and optimised the graph iteratively. They proposed an incremental state space to capture the cumulative nature of the robot motion. Based on this parameterization, the update can be performed efficiently. Grisetti et al. [34] further extended the parameterization of the state space by adopting a tree structure to perform the stochastic gradient descent update more efficiently (denoted as Toro herein). More recently, with the development of modern linear solvers, a series of algorithms attempt to solve the large non-linear least square graph optimisation problem by iteratively linearising the current state and solving the large linear system. For instance, Preconditioned Conjugate Gradient (PCG) and sparse Cholesky decomposition were both used to solve large linear systems efficiently [63, 46, 50]. The most general and state-of-the-art algorithm in this category is g2o [55], which was found to outperform many other algorithms when applied to the graph optimisation problem in robotics. To summarise, the state-of-the-art GraphSLAM algorithms can be roughly classified into two categories: those based on stochastic gradient descent (representative algorithms are SGD and Toro); and those based on modern linear solvers to solve the linear systems formed by linearisation of the non-linear least square problem at the current state (the typical algorithm is g2o). When solving graph-based optimisation problems, SGD and Toro have been found to be more robust to poor initialisations but g2o has been found to converge faster. This chapter evaluates the abilities of the three representative algorithms to solve the SLAM problem in the automated signal survey context. Based on the evaluation results, an optimised algorithm is proposed to solve the specific SLAM problem more efficiently.

5.3 Problem formulation

The GraphSLAM problem is to minimise the errors in a pose graph based on the constraints (edges) between nodes (Figure 5.1). This pose graph has two kinds of constraints: the odometry constraints that connect consecutive poses (denoted as C_{odo}), and the loop closure constraints based

[2]The information matrix is also called the *inverse covariance matrix*. The elements of this matrix encode the constraints information between different poses, providing information to correct errors in the pose graph.

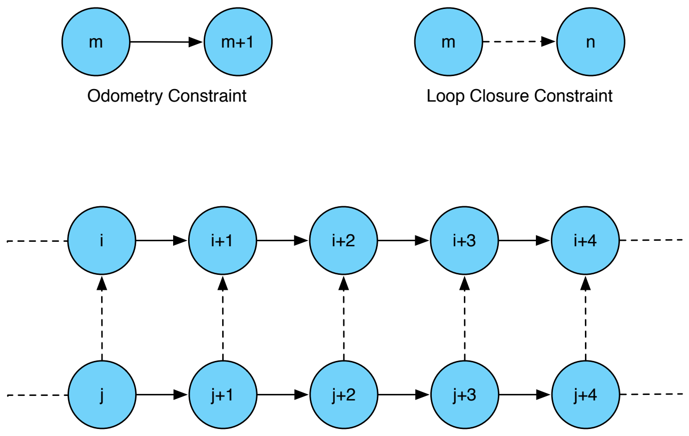


Figure 5.1: An example of constraints in a pose graph. Each node of the graph represents a pose of the system. Constraints between consecutive poses are odometry constraints; the other constraints are loop closure constraints.

on observations (denoted as C_{lp}). Here, the observations are the magnetic loop closures. Once a loop closure is detected, a group of constraints are added to the system (Figure 5.2). Let $C = C_{odo} \cup C_{lp}$ be the set of all constraints in the pose graph, X be the state vector of all the poses. The chi-squared (χ^2) errors of the whole pose graph can be formulated as

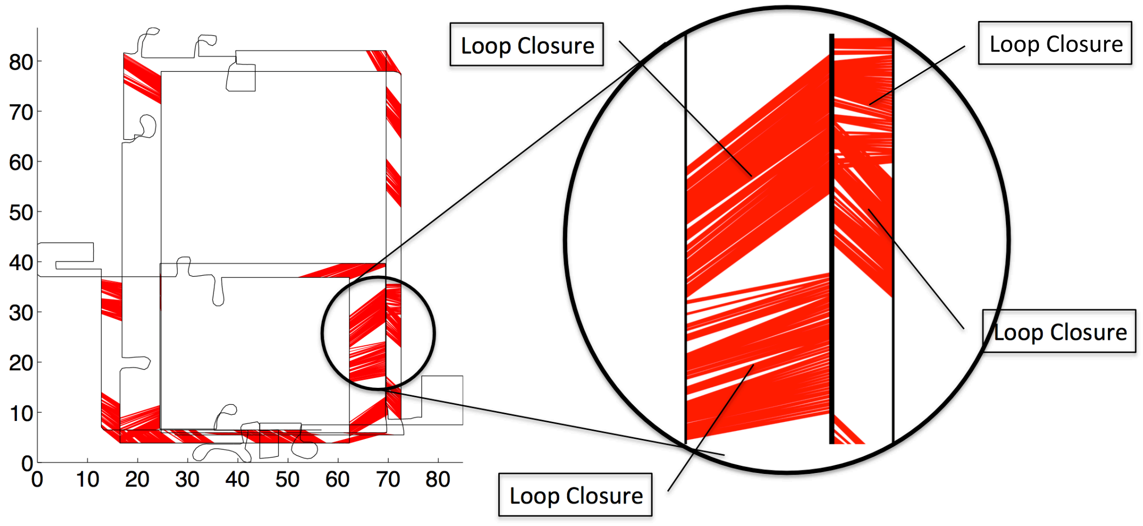


Figure 5.2: The illustration of loop closures and loop closure constraints. The figure on the left is a survey trajectory with identified loop closures; the figure on the right is the zoomed-in view of four loop closures. Because we detect loop closures by matching magnetic sequences, when a loop closure is detected, a sequence of magnetic scans are matched to another. Then each scan-to-scan matching forms a constraint in this loop closure, which is represented by a red line connecting corresponding points on the trajectory. It can be seen that the detection of a loop closure will cause a group of constraints to be added to the system.

$$F(X) = \sum_{i \in C} F_i = \sum_{i \in C} e_i^T \Omega_i e_i \tag{5.1}$$

where Ω_i is the information matrix (as mentioned in Section 5.2) of a particular constraint i, e_i is the residual error of this constraint. If the related poses of constraint i are pose m and pose n, let z_{mn} be the observed value of the observation between the two poses and h_{mn} the predicted observation value. Then the error of constraint i is

$$e_i = z_{mn} - h_{mn}$$

The GraphSLAM algorithm tries to find a configuration of X (X^*) that minimises F:

$$X^* = \arg\min_X F(X)$$

By linearisation (first order Taylor expansion) of the error e_i at the current state X we can get

$$e_i(X + \Delta X) \simeq e_i + J_i \Delta X$$

where, e_i is the error at current state X, J_i is the Jacobian of e_i at current state. Then the χ^2 error F_i becomes

$$\begin{aligned} F_i(X + \Delta X) &= e_i(X + \Delta X)^T \Omega_i e_i(X + \Delta X) \\ &\simeq (e_i + J_i \Delta X)^T \Omega_i (e_i + J_i \Delta X) \\ &= e_i^T \Omega_i e_i + 2 e_i^T \Omega_i J_i \Delta X + \Delta X^T J_i^T \Omega_i J_i \Delta X \end{aligned} \tag{5.2}$$

differentiating Equation 5.2 with respect to ΔX we obtain:

$$\frac{\partial F_i}{\partial \Delta X} = 2 J_i^T \Omega_i J_i \Delta X + 2 J_i^T \Omega_i e_i$$

then differentiating Equation 5.1 with respect to ΔX we get:

$$\begin{aligned} \frac{\partial F}{\partial \Delta X} &= \sum_{i \in C} \frac{\partial F_i}{\partial \Delta X} \\ &= \sum_{i \in C} (2 J_i^T \Omega_i J_i \Delta X + 2 J_i^T \Omega_i e_i) \\ &= 2 J^T \Omega J \Delta X + 2 J^T \Omega e \end{aligned} \tag{5.3}$$

let Equation 5.3 equal 0:

$$\begin{aligned} 2 J^T \Omega J \Delta X + 2 J^T \Omega e = 0 \\ J^T \Omega J \Delta X = - J^T \Omega e \end{aligned} \tag{5.4}$$

then we obtain the value of ΔX that minimises F:

$$\Delta X = -(J^T \Omega J)^{-1} J^T \Omega e \tag{5.5}$$

and the solution is

$$X^* = X + \Delta X \tag{5.6}$$

Most algorithms iteratively solve this problem until convergence.

5.4 State-of-the-art algorithms

This section gives a brief introduction to three state-of-the-art algorithms: g2o, Toro and SGD. Please refer to [55, 71, 34] for detailed and complete descriptions about these algorithms.

5.4.1 g2o

This is a conventional non-linear least square method that iteratively linearises the error F at current state X, computes ΔX (as in Equation 5.5) and adds ΔX to current state X (as in Equation 5.6). Because the size of the linear system (Equation 5.4) is proportional to the number of poses, without a modern linear solver, this method is impractical. The idea of g2o is to use modern linear solver such as sparse Cholesky decomposition to solve the linear system efficiently. The prerequisite is that the information matrix $J^T \Omega J$ is sparse and positive definite.

5.4.2 SGD

The SGD algorithm also solves the linear system iteratively, but in a different way: when computing ΔX, it does not consider all the constraints simultaneously, instead, it computes ΔX based on just one single constraint selected randomly. The update equation becomes:

$$X^* = X + \lambda H^{-1} J_i^T \Omega_i e_i \tag{5.7}$$

where $H = J^T \Omega J$, and λ is the learning rate, which decreases with the iteration. The use of λ is necessary for the system to converge to an equilibrium point because the pose graph generally contains antagonistic constraints. The H is used as the pre-conditioner to scale and distribute errors according to the importance of each constraint and node. To avoid expensive computation in the inversion of H, it is approximated as

$$H \simeq diag(H)$$

Thus the inversion of H can be computed trivially by inverting its diagonal elements. Using the proposed incremental state space [71], the update caused by solving a constraint is actually to distribute weighted residual errors to a sequence of consecutive poses. For example, if a constraint connects two poses with indices a and b respectively, then solving this constraint is to distribute the errors among poses $[a+1, a+2, ...b]$. A naive implementation requires $O(N)$ (assume the graph contains N poses) time to perform the update of a single constraint. A special binary tree was proposed to speed up this process. It can help to perform each update in $O(logN)$ time. More discussion about this tree is given in Section 5.5.2.1, and please refer to Section 4.3.4 of [69] for detailed description.

5.4.3 Toro

The Toro algorithm is based on the same stochastic gradient descent method as SGD, but it adopts a tree parameterization of the pose graph. With this parameterization, the number of nodes involved in the update of each constraint depends only on the topology of the environment, and the interactions between antagonistic constraints are kept small. In this way, Toro will typically converge much faster than SGD.

5.4.4 Datasets

The inputs of the SLAM back-end are the outputs of the SLAM front-end, i.e. the trajectory needed to be corrected and the identified loop closures. We use a number of real datasets collected in various testbeds to evaluate the three state-of-the-art SLAM back-end algorithms. All the datasets are processed by the SLAM front-end (discussed in Chapter 4) to get the filtered paths and loop closures. Two examples of these processed results are shown in Figure 5.3. All the datasets are shown in Appendix A, where Figure A.1 shows 4 datasets collected in WGB2a testbed, Figure A.2 shows 5 datasets collected in WGB1 testbed, Figure A.3 shows 4 datasets collected in WGB2 testbed, Figure A.4 shows 2 datasets collected in ENG testbed, 2 datasets collected in RUTH testbed and 2 datasets collected in KX testbed. Details about the testbeds, groundtruth and data collecting were given in Section 4.6.

Please note that we applied the line-filter algorithm (Section 4.5.2) to most of the original PDR trajectories. This is because if a loop closure connects two wrongly bent straight-line segments, the back-end will simply move the segments towards each other, but cannot straighten them. Also the magnetic loop closures do not provide rich enough information about how the orientation of a pose relates to the environment. So we simply assume the poses connected by a loop closure constraint should have the same orientation. Therefore, to get good final results, it is necessary to use the line

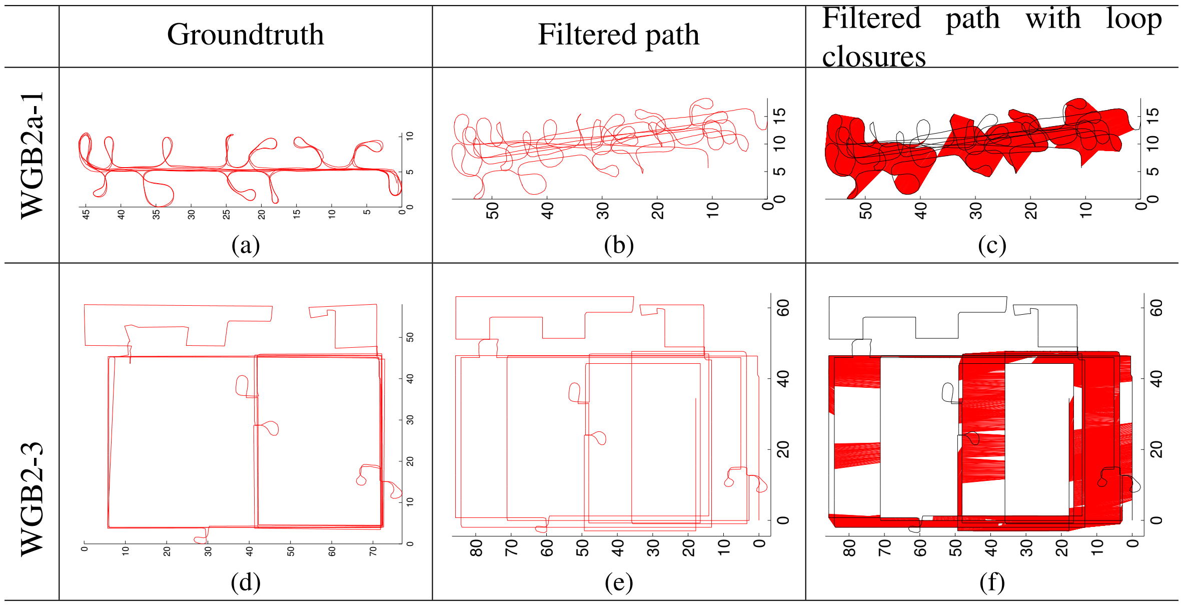


Figure 5.3: Two sample datasets (WGB2a-1 and WGB2-3) used for evaluation of SLAM back-end algorithms.

filter to identify likely straight-line segments in the original PDR trajectory and filter them to be straight.

For all the datasets, the numbers of poses, odometry constraints, loop closure constraints and loop closures are shown in Table 5.1. Each pose corresponds to one magnetic scan during the survey. An odometry constraint is the spatial relationship between two consecutive magnetic scans. This relationship is inferred from PDR result. Loop closure constraints correspond to the loop closures detected by the SLAM front-end. As described in Section 4.4, the front-end matches magnetic sequence windows to detect loop closures, so each time a loop closure is detected, a group of constraints are added. The relationship between a loop closure and the loop closure constraints it introduces is not used by g2o, Toro and SGD, but in Section 5.5 we discuss how this information can be used to design a more efficient SLAM back-end algorithm.

5.4.5 Evaluation

5.4.5.1 Algorithm settings

We adopt an open source implementation of the g2o [3]. This open source package incorporates many different robust kernels as well as various modern linear solvers/optimisers. We found by experiment that different combinations of these components produced similar results on our datasets. So we chose the default settings:

- Robust kernel: None;
- Optimiser: "gn_var_cholmod", i.e. "Gauss-Newton: Cholesky solver using CHOLMOD (variable blocksize)".

We also adopt an open source implementation of Toro [4] and use the default settings as well.

We implemented the SGD algorithm using the C++ language. It randomly selected a constraint to process each time (as in [69]) rather than iterating through all constraints in a fixed order (as

[3]https://openslam.org/g2o.html
[4]https://www.openslam.org/toro.html

Table 5.1: Statistics of all datasets

Dataset	Poses	Odometry Constraints	Loop Closure Constraints	Loop Closures
WGB2a-1	26262	26261	10382	29
WGB2a-2	3982	3981	596	2
WGB2a-3	8177	8176	996	4
WGB2a-4	20671	20670	7221	29
WGB1-1	18551	18550	1370	26
WGB1-2	25958	25957	16213	109
WGB1-3	14639	14638	2820	19
WGB1-4	13353	13352	1338	9
WGB1-5	29941	29940	790	5
WGB2-1	7490	7489	152	5
WGB2-2	15692	15691	2690	27
WGB2-3	17645	17644	9690	130
WGB2-4	16195	16194	401	5
ENG-1	4314	4313	237	4
ENG-2	12748	12747	1125	7
RUTH-1	7218	7217	7200	60
RUTH-2	7980	7979	4117	23
KX-1	16402	16401	1165	20
KX-2	21499	21498	6094	41

in [71]) because we found randomisation (which is better for escaping local minima) gave much better performance for all our datasets.

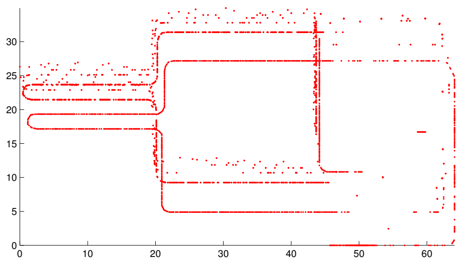
5.4.5.2 Odometry constraints

The inputs for the g2o and Toro include all the poses, odometry constraints and loop closure constraints. But the odometry constraints are excluded for SGD to avoid over-constraining the system. Unlike g2o and Toro, SGD does not rely on the odometry constraints to maintain the topology of the trajectory. It simply loops over each loop closure constraint and distributes errors among poses related to this constraint (the poses related to a constraint include the two poses connected by it and all the poses between them on the trajectory). The topology of the pose graph is maintained naturally by the mechanism of the algorithm.

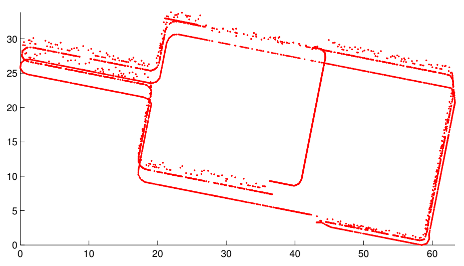
Contrarily, odometry constraints are compulsory for both g2o and Toro. Toro relies on the connectivity of the input pose graph to build the special tree structure it adopts. The lack of enough odometry constraints can cause the pose graph to be non-connected, which results in the failure of the tree building process. For g2o, the Cholesky solver it adopts requires the information matrix H of the system to be positive-definite. The lack of odometry constraints can cause some diagonal elements of H to be zero, which causes it to be non-positive definite. The use of the LM (Levenberg-Marquardt) algorithm can seemingly relax this restriction because LM will add a damped identity matrix I to the information matrix:

$$H' = H + \lambda I$$

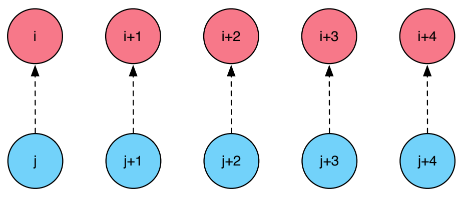
where λ is the damping factor. However, in this case, g2o is still not working for the following reasons: g2o corrects the errors in the system by 'moving' poses in the pose graph to the 'right' place, and it moves the poses according to the gradient of the system. The gradient of the system is computed based on every constraint the system has. So different constraints related to a single pose act



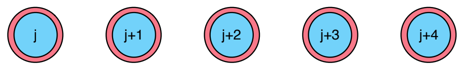
(a) g2o result on RUTH-1 (with odometry constraints discarded).



(b) g2o result on RUTH-2 (with odometry constraints discarded).



(c) The pose graph in Figure 5.1 (with odometry constraints discarded).



(d) g2o result on the pose graph shown on the left. The sizes of the blue poses are deliberately reduced to show the overlap more clearly.

Figure 5.4: g2o results when odometry constraints are discarded. Figure 5.4a and 5.4b show the g2o results on the two RUTH datasets (which are shown in Figure A.4i and A.4l). The odometry constraints were discarded when applying g2o. The results showed that any poses not connected by any constraint just stay where they are. Each result has a χ^2 error of zero because any two poses connected by a constraint are simply moved towards each other until overlap. This is more clear if using the pose graph in Figure 5.1 to illustrate: if all the odometry constraints are discarded (Figure 5.4c), then g2o will cause pose j to overlap with pose i, pose $j+1$ to overlap with pose $i+1$,..., i.e. any two poses connected by a loop closure constraint will ultimately overlap with each other (Figure 5.4d).

as 'springs' to constrain where this pose should be moved to. Specifically, the odometry constraints between consecutive poses act as a spring system. This spring system tries to maintain the topology of the trajectory against the effects of the loop closure constraints. Without the odometry constraints, most poses are connected by only a loop closure constraint. Then any two poses connected by a loop closure constraint will be simply moved towards each other regardless how the topology of trajectory is, and this can easily damage the trajectory topology.

This difference between SGD and g2o comes from their state space choices. SGD adopts an incremental state space (incremental pose) while g2o adopts a global state space (global pose). The state space representation determines how errors will be spread over poses when solving a constraint: In SGD, the error embedded in a constraint will be spread not only over the two poses connected by this constraint, but also over the poses between them on the trajectory, by which the topology of the trajectory is naturally maintained. However, g2o distributes error only over the two poses connected by this constraint, and this will cause two problems as shown in Figure 5.4:

1. Any poses not connected by any constraints will just stay where they are.

2. Any two poses connected by a single constraint will be simply moved towards each other until overlap.

Therefore, g2o cannot give reasonable results when there are no odometry constraints to maintain the topology of the pose graph.

5.4.5.3 Metrics

In robotics literature, there are three common metrics used to evaluate the performance of a SLAM back-end[5]:

- **Chi-Squared Error (χ^2 error).** This kind of error is used when the groundtruth data about the trajectory is unavailable. Equation 5.1 is the χ^2 error of the whole system, and this is what the GraphSLAM algorithms explicitly attempt to minimise. We can normalise this error by the number of constraints and use the normalised χ^2 error as a direct indicator of the performance of different SLAM back-end algorithms:

$$\chi^2_{normed} = mean(\sum_{i \in C} F_i) = mean(\sum_{i \in C} e_i^T \Omega_i e_i) \tag{5.8}$$

  However, as reported in [69], what χ^2 error tells us is how much error is embedded in the constraints, but not how well the resultant trajectory approaches the groundtruth, i.e, good χ^2 error does not necessarily mean good map (trajectory) quality. Figure 5.5a shows the χ^2 error of different algorithms on dataset WGB2-3. It shows that the χ^2 error of all the algorithms being tested converged to 0. However, as we will show later in this section, the corresponding results can be far from the groundtruth. We found this is the common case when testing the three state-of-the-art algorithms. In the signal survey context, a good SLAM back-end is required to return corrected trajectories that well approach the groundtruth. Therefore, from a practical perspective, we do not use the χ^2 error to evaluate the performance of these back-end algorithms. To avoid confusion, please note that the SLAM back-end algorithms are still trying to minimise the χ^2 error when correcting the pose graph because no groundtruth information is available (otherwise we do not need to solve it). So we emphasise that *the χ^2 error is only used by the back-end algorithms themselves but not for our evaluation purpose when comparing the performance of different back-end algorithms.*

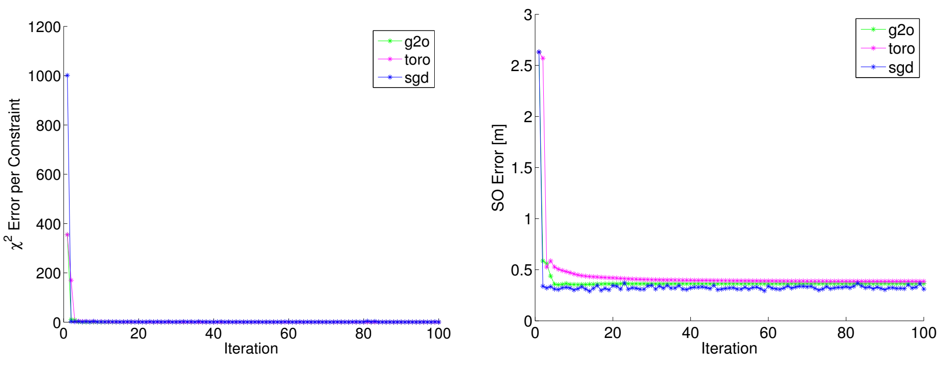
- **Subjective-Objective Error (SO error).** This metric is used frequently to evaluate SLAM back-end results (and was mentioned in Section 4.7.1. We give a brief review here to compare with other kinds of metric). Like the χ^2 error, the SO error is also used when no groundtruth information is available. The intuitive interpretation of this error is: it evaluates how accurately a SLAM back-end algorithm corrects the errors related to the loop closure constraints. A corrected trajectory returned by a perfect SLAM back-end should have an SO error close to 0. We refer the reader to [7, 24, 42] for its mathematical description and application examples.

- **State-Squared Error (SS error).** This metric is proposed in [69]. For a graph with n poses, denote the configuration of the result trajectory by X, where $X = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, and denote the groundtruth configuration by $\bar{X}$, where $\bar{X} = \{(\bar{x}_1, \bar{y}_1), (\bar{x}_2, \bar{y}_2), ..., (\bar{x}_n, \bar{y}_n)\}$. Then :

$$SS_{error} = mean\{dis_i^2 \mid dis_i^2 = (x_i - \bar{x}_i)^2 + (y_i - \bar{y}_i)^2\} \tag{5.9}$$

  This is actually the mean squared Euclidean error between the (x, y) pose positions on the result trajectory and the groundtruth trajectory. However, this metric is relevant to the rigid-body transformation of the pose graph. For example, rotating, scaling or translating either the result trajectory or the groundtruth trajectory does not affect the topology of the trajectory (nor the χ^2 error of the result), but it affects the SS error computed by Equation 5.9. To give an

[5]We have mentioned two metrics in Section 4.7.1. Here, we introduce several more advanced metrics for better evaluation of back-end algorithms.



(a) χ^2 errors. (b) Subjective-objective errors.

Figure 5.5: χ^2 errors and SO errors when applying different algorithms on dataset WGB2-3.

unambiguous comparison, as suggested by [69], a rigid-body transform T that minimises the SS error should be computed and applied to either the result trajectory or the groundtruth trajectory beforehand. Given the configurations of the result trajectory and the groundtruth trajectory, this transform T can be computed by the algorithm proposed in [39]. Then, Equation 5.9 is used to compute the SS error of the SLAM results.

Empirically we found that each SLAM back-end algorithm will give a χ^2 error approaching 0 after enough iterations regardless how good the quality of the result trajectory is. Both SO and SS errors are better indicators of the result quality. However, the SO error is not as sensitive to the SLAM result quality as SS error is. For example, Figure 5.5b shows the SO errors of different algorithms on dataset WGB2-3. Compared with the SS errors (Figure 5.6h) and the result trajectories (Figure 5.6e, 5.6f and 5.6g), we find that SS errors are more sensitive to the result quality. Therefore, although SO error is a good metric to measure how well a back-end algorithm optimises the trajectory according to the loop closures (Section 4.7.1), to better compare the performance of different back-end algorithms, we use the SS error. [6] We first compute the rigid-body transform T that minimises the SS error, and then apply it to the groundtruth trajectory to get the transformed groundtruth configuration $\bar{X}_t$. Then we use $\bar{X}_t$ to compute the SS error by Equation 5.9. We run 100 iterations of g2o, SGD and Toro on each dataset and present the results below.

Figure 5.6 show the results of each algorithm on two sample datasets. It can be seen from these figures that SS error is a good indicator of the quality of the SLAM result: lower SS errors correspond to better result (map) quality, as proved by SGD results which approach the groundtruth well; higher SS errors correspond to worse quality, as shown by the results of g2o and Toro, which are distorted to an unacceptable extent.

5.4.5.4 Results and analysis

The results of Toro, g2o and SGD on each dataset are shown in Appendix A (Figure A.5, A.6, A.7 and A.8). These figures show that in most cases, g2o and Toro failed to converge to acceptable results, but SGD worked well on each dataset. As described before, all the three back-end algorithms explicitly minimise the χ^2 error of the pose graph (because of the lack of groundtruth), but we use the SS error to evaluate their results (because it is a good indicator about how well a trajectory approaches the groundtruth). We should note that minimising the χ^2 error can cause the SS error to be minimised

[6]Please note again that without the groundtruth information, the SLAM back-end algorithms are still trying to minimise the χ^2 error so as to correct the pose graph. The SS error is only used for evaluating the back-end results when we have groundtruth.

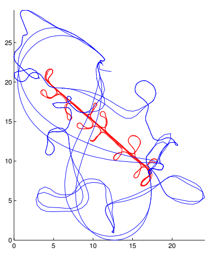
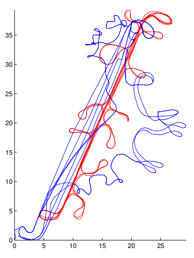
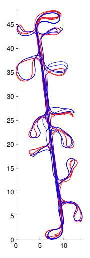
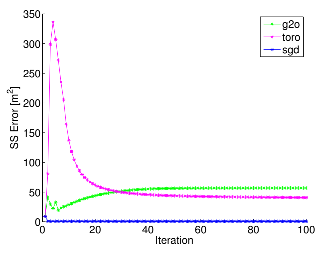
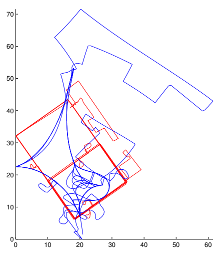
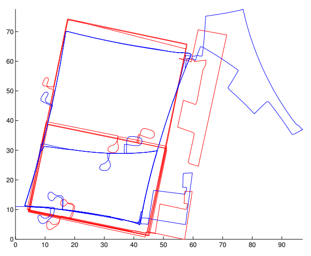
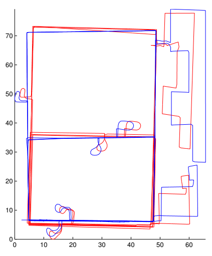
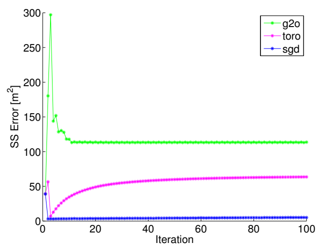
	g2o	Toro	SGD	State Squared Error
WGB2a-1	 (a)	 (b)	 (c)	 (d)
WGB2-3	 (e)	 (f)	 (g)	 (h)

Figure 5.6: SLAM results on WGB2a-1 and WGB2-3. The trajectories (the red is the transformed groundtruth and the blue the slam result) and the SS errors are the results of the 100th iteration of different algorithms.

but not necessarily. Actually, as stated before, the χ^2 error of all the three algorithms converges to zero after enough iterations. However, whether an algorithm converges to the minimum SS error (or converges to a result that has the best quality from a practical perspective) depends on how this algorithm approach the minimum χ^2 error. Generally speaking, the methods based on Cholesky decomposition like g2o can typically find the lowest χ^2 error because they usually take the path directly leading to it. However, these methods can easily get stuck in local minima because the state space of the non-linear optimisation problem have long valleys with small χ^2 error. In contrast, the randomised algorithms like SGD are capable of escaping local minima and finding the path leading to the global minimum when exploring the state space. Actually, Toro has a very similar error distribution mechanism as SGD to optimise the graph. However, Toro iterates over each constraint in a fixed order pre-defined by the tree structure it adopts, and, it is this fixed order that keeps the high efficiency of Toro. This fixed order makes Toro much less randomised than SGD, which leads to worse results.

In summary, the SGD algorithm achieves the best performance on our problem. The evaluation results presented here are similar to the results reported in [69], but here we compare the SGD algorithm with state-of-the-art algorithms on a specific problem.

5.5 Optimised back-end design

5.5.1 Motivation

In the context of automated signal survey, the number of loop closure constraints can be very large. Table 5.1 shows the statistics of all the datasets collected during each survey for this work. The duration of each survey ranges from 2 minutes to 15 minutes. Because the magnetic scanning frequency is high (usually between 50 to 100 Hz), in larger areas where the survey time is much longer, the number of loop closure constraints in the pose graph can be even larger than shown in Table 5.1. Figure 5.12 shows that the execution time per iteration of SGD increases dramatically as the number of constraints increases, which means SGD is not scalable to deal with larger datasets.

Another motivation for more efficient back-end is that from a practical perspective, the surveyor usually needs to know the SLAM result immediately after a survey has been done so that she knows whether she needs to re-do the survey (sometimes the survey path cannot be successfully recovered because the path does not contain enough loop closures). In this case real-time feedback of the SLAM result is very valuable, by which the surveyor can take action immediately after the SLAM goes wrong.

To increase the efficiency and scalability of the automated survey system, a possible way is to reduce the size of the pose graph being built, i.e. reduce the number of nodes (poses) and edges (constraints). For example, we can convert the pose graph consisting of high-frequency magnetic sample points (about 100 hz) and related constraints to the pose graph consisting of PDR step points (about 1.0 hz) and related constraints. This would reduce the size of the pose graph but could result in loss of information. For instance, the magnetic loop closure constraints provide much richer information about how a PDR step should be stretched, compressed or aligned, which is better to fine-tune the trajectory in the correction process. There are also many research works about how to sparsify the pose graph based on more generalised and advanced techniques [54, 9, 10, 89, 62, 41].

Obviously, another way to increase the system efficiency is to optimise the back-end algorithm itself. This is the focus of this section. We show how SGD can be optimised according to the features of magnetic-sequence based loop closures. We demonstrate that the optimised algorithm is more efficient for large datasets than the original SGD algorithm, so more suitable for large-scale deployment of indoor positioning.

5.5.2 Optimisation based on tree operation and approximation

5.5.2.1 Linear-time pose computation

As described in Section 5.4.2, SGD requires to estimate a pre-conditioner H to scale the distribution of residual error. The process to estimate H is shown in Line 6 to 15 of Algorithm 1 (where H is approximated by its diagonal elements stored in M), during which the function $getPose$ needs to be called E times for a graph with E constraints. $getPose$ computes current pose values by adding all the changes made to a pose. These changes are made by solving corresponding constraints: solving a constraint connecting two poses with indices a and b respectively will distribute an array of weighted errors among poses with indices $[a+1, a+2, ...b]$. The trivial way to implement this is using an array to hold all the pose values, which means $O(N)$ time (assuming this pose graph has N nodes/poses) is needed to perform updates over corresponding poses.

SGD adopts a special binary tree proposed in [69] to speed up this process. Each node i ($0 \leq i < N$) of the tree holds a *node value* n_i and maintains a pose value v_i. The pose value v_i is defined as the sum of the node value n_j from node i up to the root of the tree along the ancestry chain. Because it is a binary tree, so each v_i can be computed in $O(logN)$ time. The most important advantage of the tree is that it supports the operation of adding some amount to each v_i with $i \geq I$, where I is an arbitrary index and $0 \leq I < N$. This can be done by the tree in $O(logN)$ time because the amount only needs to be added to a set of n_j with $j \in J$, where J is the index set of nodes such that each path from node $j \geq I$ to the root goes through exactly one node with its index in J. Section 4.3.4 of [69] has shown that the size of J is $O(logN)$ and also given an example of this operation. As described above, solving a constraint requires to distribute an array of weighted errors over a consecutive sequence of poses with indices $[a+1, a+2, ...b]$. Then by this tree, we can first perform an $O(logN)$ operation to distribute some amount over poses $[a+1, a+2, ...N]$ and then perform another $O(logN)$ operation to distribute some amount over poses $[b+1, a+2, ...N]$ to achieve the update of the constraint [69].

In summary, we need $O(logN)$ time to solve a constraint and the same time to read a pose value v_i. The latter is the $getPose$ operation in Algorithm 1. So $O(ElogN)$ time is needed to compute all necessary poses for a single estimation of H. This time complexity is high for a graph with dense constraints, especially for the pose graph built from the magnetic sequence-based automated survey. Therefore, if all the poses can be computed before the estimation of H in a faster way, the time complexity can be reduced. Obviously, the naive implementation to compute all the poses by the tree needs $O(NlogN)$ time. Below we show how to utilise the special structure of the tree to compute all the poses in linear time ($O(N)$).

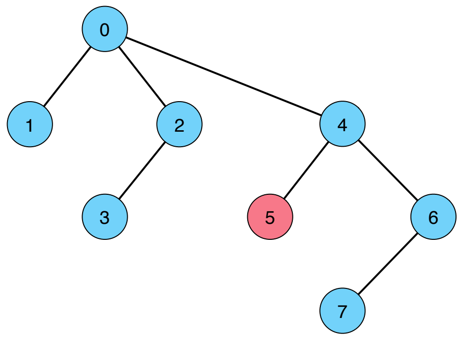


Figure 5.7: Example of linear-time pose computation. This is a binary tree adopted by SGD to maintain pose values. We take the node 5 as an example. The pose value v_5 is the sum of node values n_5, n_4 and n_0, i.e. $v_5 = n_5 + n_4 + n_0$. Similarly, we have $v_4 = n_4 + n_0$. Then we can get $v_5 = n_5 + v_4$. So, if we can compute the pose values of any parent nodes (node 4 here) earlier than any children (node 5 here), we can compute all pose values in $O(N)$ time. For example, the breadth fist search order (BFS) meets this requirement: $v_0 = n_0, v_1 = n_1 + v_0, v_2 = n_2 + v_0, v_4 = n_4 + v_0, v_3 = n_3 + v_2, v_5 = n_5 + v_4, v_6 = n_6 + v_4, v_7 = n_7 + v6$.

Recall that each node of the tree represents a pose in the graph (and holds the pose value v_i), and computing a pose value is to sum the node value n_j from node i up to the root along the ancestry chain. Then, computing a pose value v_i can be seen as adding the node value n_i to its parent node's pose value. So, if we can compute the pose values of all the nodes in an order that the pose value of any parent node is always computed earlier than its children, we are able to compute all the pose values in linear time ($O(N)$). Fortunately, the breadth first search (BFS) (or depth first search, DFS) order of the tree meets this requirement and can be found trivially when the tree is built. When we need to compute all the poses of the tree, we only need to compute the pose values of all the nodes in the BFS/DFS order, which can be done in $O(N)$ time. An example is shown in Figure 5.7.

In this way, we can compute all the poses in linear time before the computation of H. In a graph with dense constraints ($E \simeq N$), the time complexity is then much lower than the original $O(ElogN)$.

5.5.2.2 Tree-based H estimation

With the above improvements, as shown in Algorithm 1, the calculation of $M = diag(H)$ requires $O(EN)$ time ($getPose$ only needs $O(1)$ time now). By careful inspection, we find that when estimating M, each constraint causes the same amount of value distributed to a continuous part of the diagonal elements of H (Line 11 to 14), which is the similar process as updating the poses when solving a constraint. Therefore, we can use the same tree-based mechanism to maintain the elements of M and speed up the M estimation. Thus each constraint needs only $O(logN)$ time to update M. So the M estimation process can be done in $O(Elog(N))$ time.

After M is computed, it is then used in the stochastic gradient descent process to compute $totalWeight$ for each constraint (Line 17 to 23). Each $totalWeight$ requires $O(N)$ time to compute, so a single iteration requires $O(EN)$ time to compute $totalWeight$ for all the constraints. We can also accelerate this by the method proposed in Section 5.5.2.1 to compute all the tree values. Because M is now computed using the tree, we can first recover each element of M ($M_i, i \in [1 : numPoses]$)

in linear time by the method proposed in Section 5.5.2.1. Then, define $cmWeight_i = \sum_{j\in[1,i]} 1/M_j$, so that the $totalWeight$ for each constraint can be computed in $O(1)$ time as shown in Algorithm 2. Now, each iteration needs only $O(N + E)$ time to compute $totalWeight$ for all constraints.

Algorithm 1 Pre-Conditioner Estimation in SGD

```
1: $iters = 0$
2: **loop**
3:     $iters$++
4:     ...
5:     // *Update approximation* $H = J^T\Omega J$, $M = diag(H)$
6:     $M = zeros(numPoses, 3)$
7:     **for all** $a, b, t_{ab}, \Omega_i$ in $Constraints$ **do**
8:         $poseA$ = getPose(a)
9:         $R$ = rotation matrix of $poseA$
10:        $W = R\Omega_i R^T$
11:        **for** $i$ = a+1 **to** $b$ **do**
12:            $M_{i,1:3} = M_{i,1:3} + diag(W)$
13:            ...
14:        **end for**
15:    **end for**
16:
17:    // *Modified Stochastic Gradient Descent*
18:    **for all** $a, b, t_{ab}, \Omega_i$ in $Constraints$ **do**
19:        ...
20:        $totalWeight = zeros(1, 3)$
21:        **for** $i$ = a+1 **to** $b$ **do**
22:            $totalWeight = totalWeight + 1/M_i$
23:        **end for**
24:        ...
25:    **end for**
26: **end loop**
```

5.5.2.3 Speeding up H estimation by approximation

In the context of the magnetic sequence matching, when a loop closure is detected, a group of constraints are added to the graph. This feature can be used to further speed up the estimation of H. As shown in Algorithms 1 and 2, a value $W = R\Omega_i R^T$ needs to be computed for each constraint when estimating M. W is actually the information matrix of the constraint in the global reference frame. The computation of W needs two matrix operations. From Figures A.1, A.2, A.3, A.4 we can see that most loop closures are detected along straight line segments, so it is reasonable to assume that the orientations of poses in a consecutive pose sequence (related with a group of constraints) are mostly the same. Also, the information matrix Ω_i of any constraint in a group is the same. Therefore, we can use the W of any constraint in a group to approximate the W for the other constraints in the same group. In this way, W only needs to be computed once for a group of constraints. For the high frequency magnetic scans, a group usually contains hundreds of constraints, so this approximation makes a significant difference.

Algorithm 2 Improved Pre-Conditioner Estimation

```
 1: $iters = 0$
 2: loop
 3:     $iters$++
 4:     ...
 5:     // Update approximation $H = J^T \Omega J$, $M = diag(H)$
 6:     $M = zeros(numPoses, 3)$
 7:     {Compute all poses in linear time}
 8:     for all $a, b, t_{ab}, \Omega_i$ in $Constraints$ do
 9:         $poseA =$ getPose(a)                                  // This is $O(1)$ now
10:         $R =$ rotation matrix of $poseA$
11:         $W = R\Omega_i R^T$
12:         // Distribute values in $O(log(N))$ time by tree
13:         {Distribute $diag(W)$ over $M_i, i \in [a+1, b]$}
14:         ...
15:     end for
16:
17:     {Recover $M_i, i \in [1, numPoses]$ in O(N) time by proposed tree operation}
18:
19:     // Compute cumulative weight
20:     $cmWeight_1 = 1/M_1$
21:     for $i$ = 2 to $numPoses$ do
22:         $cmWeight_i = cmWeight_{i-1} + 1/M_i$
23:     end for
24:
25:     // Modified Stochastic Gradient Descent
26:     for all $a, b, t_{ab}, \Omega_i$ in $Constraints$ do
27:         ...
28:         $totalWeight = cmWeight_b - cmWeight_a$
29:         ...
30:     end for
31: end loop
```

5.5.3 Error propagation over selected constraints only

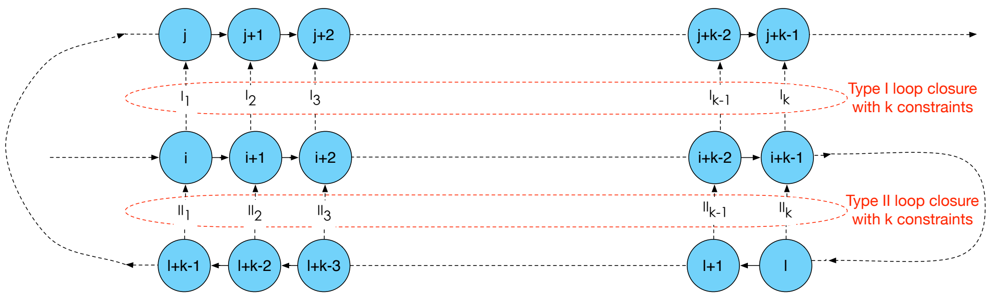


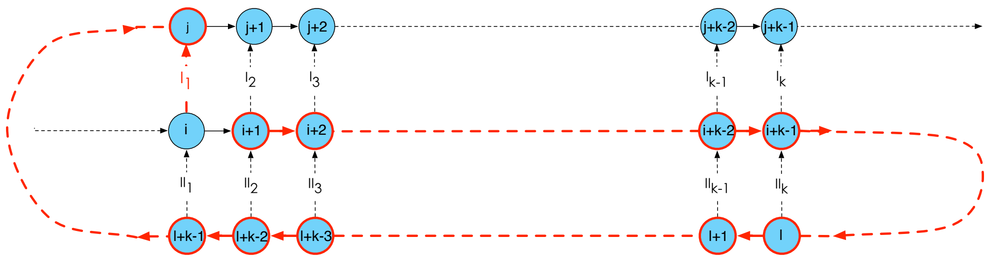
Figure 5.8: Example of two types of loop closures. The pose graph showed here is typical for the situation when the surveyor walks up and down a corridor. Two loop closures are shown in the pose graph: a Type I loop closure with k constraints I_1 to I_k; a Type II loop closure with k constraints II_1 to II_k. The rule to classify loop closures is: write the index number of poses in the older sequence (i.e. the pose sequence with smaller index numbers) in ascending order (i.e. $i, i+1, ...i+k-1$), if the index number of poses in the later sequence (i.e. the pose sequence with larger index numbers) is increasing (i.e. $j, j+1, ..., j+k-1$), then this is a Type I loop closure; otherwise (i.e. $l+k-1, l+k-2, ..., l$), this is a Type II loop closure.

5.5.3.1 Structural feature of loop closures and constraints

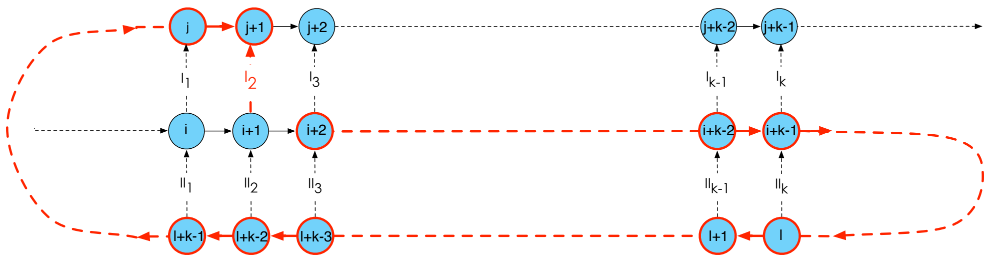
As described above, a detected loop closure adds a group of constraints to the system. The original SGD algorithm solves every constraint iteratively in the pose graph, which can fine tune the solution to approach the global minimum state. Figure 5.8 shows two abstracted loop closures and their corresponding constraints (each loop closure introduced k constraints). Please note that a loop closure relates an older pose sequence to a later one. So if we always write the index number of poses in the older sequence in ascending order, the loop closures can be classified into two categories based on the order (increasing or decreasing) of the index number of poses in the later sequence.

Solving a constraint in SGD is equivalent to distributing the error over a consecutive sequence of poses. For example, in Figure 5.8, solving the constraint I_1 in Type I loop closure spreads the error of I_1 over $Pose_{i+1}$, $Pose_{i+2}$,..., ,$Pose_{j-1}$, $Pose_j$. Similarly, solving the constraint II_1 in Type II loop closure spreads the error of II_1 over $Pose_{i+1}$, $Pose_{i+2}$,..., $Pose_{l+k-2}$, $Pose_{l+k-1}$. Figure 5.9 gives more examples about which poses are affected by solving corresponding constraints in Figure 5.8. In Figure 5.10, we draw which poses are affected by solving a certain constraint on each row, and align the poses with the same indices vertically, thus to show the fact that solving constraints within the same loop closure group affects many common poses. For instance, solving I_1 in the Type I loop closure affects $Pose_{i+1}$ to $Pose_j$ and solving I_2 in the Type I loop closure affects $Pose_{i+2}$ to $Pose_{j+1}$. So, the poses affected by solving I_1 and solving I_2 respectively have $Pose_{i+2}$ to $Pose_j$ overlapped.

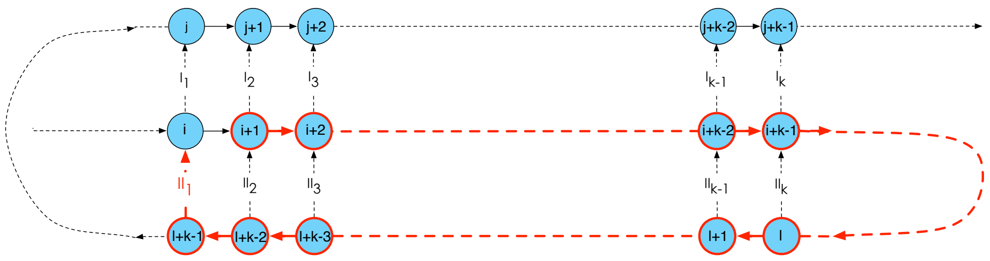
Because in either kind of loop closure, the poses affected by different constraints in the same loop closure group overlap heavily, solving a constraint can affect the states of other constraints within the same loop closure group. For example, solving a constraint C may cause other constraints that consistent with C to be solved or nearly solved, but it might cause constraints that antagonistic to C being pushed off their solved states (if they have been solved or nearly solved). The SGD algorithm uses a decreasing learning rate to modulate the interactions between constraints. During an iteration, each constraint in a loop closure group will be solved. But in practice, as we demonstrate later, solving only two or three selected constraints in a loop closure group is enough to find a reasonable solution, and this can largely reduce the computational cost.



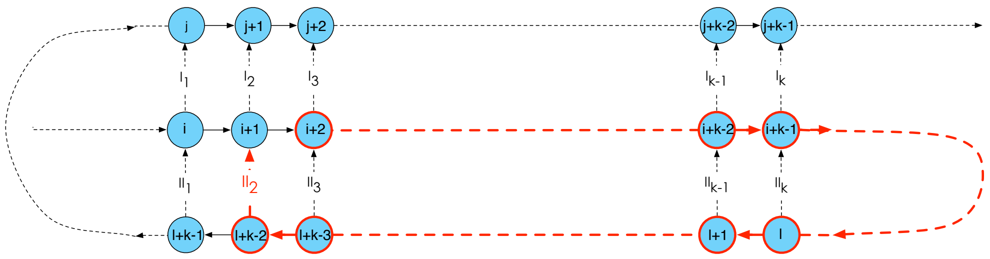
(a) Solving the constraint I_1 in the Type I loop closure affects $Pose_{i+1}$ to $Pose_j$.



(b) Solving the constraint I_2 in the Type I loop closure affects $Pose_{i+2}$ to $Pose_{j+1}$.

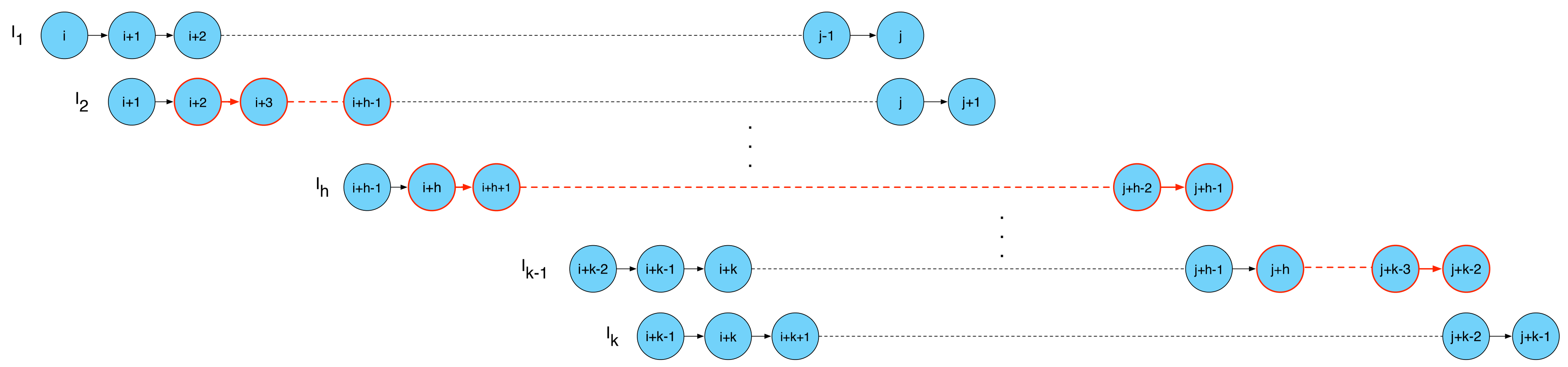


(c) Solving the constraint II_1 in the Type II loop closure affects $Pose_{i+1}$ to $Pose_{l+k-1}$.

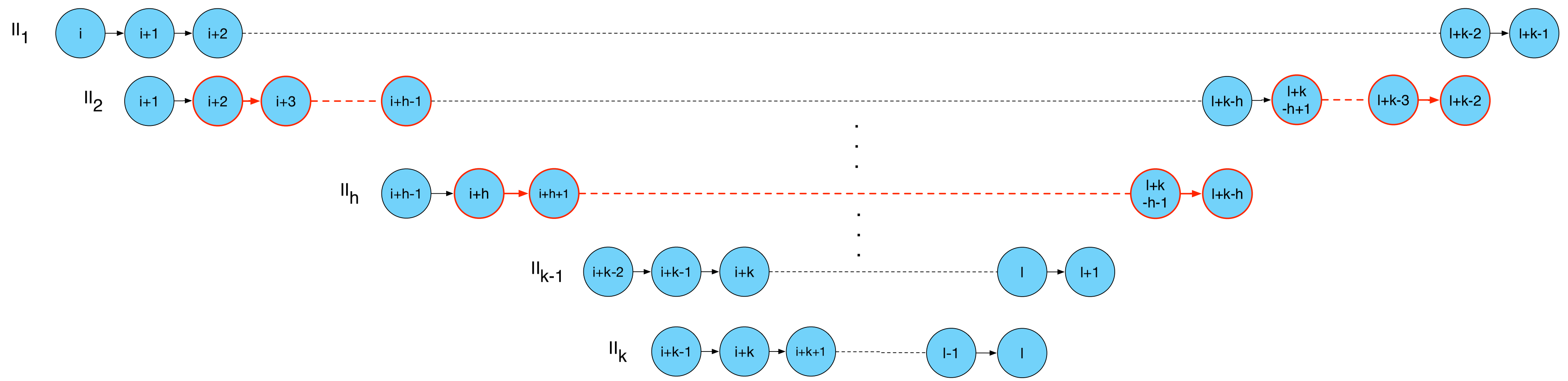


(d) Solving the constraint II_2 in the Type II loop closure affects $Pose_{i+2}$ to $Pose_{l+k-2}$.

Figure 5.9: Illustration of constraints and their affected poses. The constraint and the poses it affects are marked in red.



(a) Solving constraints in Type I loop closure. Assume that based on the rules described in Section 5.5.3, constraint I$_h$ is the max error constraint (i.e. the C_{max}), constraint I$_2$ and I$_{k-1}$ are the selected constraints (i.e. C_{I_a} and C_{I_b} respectively). When solving I$_2$ and I$_{k-1}$, the bounded residual errors are distributed among only the poses that not covered by I$_h$ (i.e. $Pose_{i+2}$ to $Pose_{i+h-1}$ and $Pose_{j+h}$ to $Pose_{j+k-2}$).



(b) Solving constraints in Type II loop closure. Assume that based on the rules described in Section 5.5.3, constraint II$_h$ is the max error constraint (i.e. the C_{max}), constraint II$_2$ is the selected constraint (i.e. the C_{II}). When solving II$_2$, the bounded residual errors are distributed among only the poses that not covered by II$_h$ (i.e. $Pose_{i+2}$ to $Pose_{i+h-1}$ and $Pose_{l+k-h+1}$ to $Pose_{l+k-2}$).

Figure 5.10: Illustration of solving selected constraints in the two types of loop closures respectively shown in Figure 5.8.

5.5.3.2 Constraints selection and error propagation

Below is the proposed method to solve only two or three selected constraints in a loop closure group:

1. For a group of constraints, first solve the constraint with the max χ^2 error (denote this constraint as C_{max}). This step requires us to compute the χ^2 errors for all constraints and find the one with the max χ^2 error. The χ^2 error e of a constraint is:

$$e = r^T W r \simeq r^T r \tag{5.10}$$

   where r is the residual error of this constraint and $W = R\Omega_i R^T$, which is assumed constant for a given group as per Section 5.5.2.3. Please note that approximating e as $r^T r$ is not appropriate in theory because different components of r need to be scaled properly. But this makes little difference in the converged results as we tested.

2. Next, we solve a very small set of constraints that are consistent with C_{max}. Define the consistency between two constraints as: a constraint C_a with residual error r_a is consistent with another constraint C_b with residual error r_b if [7]:

$$r_a * r_b > 0$$

   We select the set of constraints that needs to be solved based on these rules:

   (a) All constraints in this set are consistent with C_{max};
   (b) The constraints in this set cover as many poses as possible;
   (c) This set contains as few constraints as possible.

   Solving these selected constraints can push the solution in the same direction, and the conflicting effects of antagonistic constraints can be attenuated. Since very few constraints need to be solved for a loop closure group, the computational cost can be lowered. The selection process is trivial (Figure 5.10):

   - For the Type I loop closure, loop from the first constraint to C_{max} until a constraint that is consistent with C_{max} is found (represent it by C_{I_a}); then loop from the last constraint to C_{max} until a consistent constraint is found (represent it by C_{I_b}).
   - For the Type II loop closure, loop from the first constraint to C_{max} until a consistent constraint is found (represent it by C_{II}).

   Thus, the constraints to be solved for the Type I loop closure are C_{max}, C_{I_a} and C_{I_b}; the constraints to be solved for the Type II loop closure are C_{max} and C_{II}. However, it should be noted that, when solving C_{I_a}, C_{I_b} and C_{II}, we do not distribute errors over the poses that overlapped with the poses affected by C_{max}, because this will disturb the solved state of C_{max}. Instead, we distribute errors only over the poses that are not covered by C_{max}. And, to avoid over optimisation, we bound the residual errors conservatively before distribution. Figure 5.10 provides an example of this process. The detailed algorithm is shown in Algorithm 3.

   In summary, instead of solving all the constraints in a loop closure group, only two or three selected ones are solved so as to largely reduce the computational cost.

[7] Here we assume r is a scalar for simplicity's sake. In practice, r is a vector (e.g. $r = x, y, heading$) and solving a constraint requires to distribute errors separately for each component. In this case, our method can be applied in the same way for each component.

Algorithm 3 Error Propagation

```
1: $iters = 0$
2: loop
3:     $iters$++
4:     ...
5:     // Modified Stochastic Gradient Descent
6:     // We show only how the Type I loop closure is solved
7:     // Solving Type II is similar so omitted here for brevity
8:     // Each loop closure corresponds to a mini-batch (group) of constraints
9:     for all $Mini - batch$ in $Constraints$ do
10:        {Find max error constraint $C_{max} = \{a_{max}, b_{max}, e_{max}, t_{max}, \Omega_{max}\}$}
11:        $weight_{max} = cmWeight_{b_{max}} - cmWeight_{a_{max}}$
12:        Solve $C_{max}$
13:
14:        // Loop from first constraint to $C_{max}$
15:        for $C = \{a, b, e, t_{ab}, \Omega_i\}$ from first constraint to $C_{max}$ do
16:            if $C$ is not consistent with $C_{max}$ then
17:                continue;
18:            end if
19:            $weight_{overlap} = 0$
20:            if $b > a_{max}$ then
21:                $weight_{overlap} = cmWeight_b - cmWeight_{a_{max}}$
22:            end if
23:            $e_{overlap} = e_{max} * weight_{overlap}/weight_{max}$
24:            $e_{residual1} = e - e_{overlap}$
25:            // Check consistency again
26:            if $e * e_{residual1} < 0$ then
27:                continue;
28:            end if
29:            $totalWeight = cmWeight_b - cmWeight_a$
30:            $weight_{residual} = totalWeight - weight_{overlap}$
31:            $e_{residual2} = e * weight_{residual}/totalWeight$
32:            // Bound the error conservatively to avoid over optimisation
33:            $e_{residual} = (|e_{residual1}| < |e_{residual2}|)?e_{residual1} : e_{residual2}$
34:            $W = R\Omega_i R^T$
35:            $d = 2We_{residual}$
36:            $\lambda \propto \frac{1}{iters}$                                // the learning rate
37:            $\beta = (a_{max} - a) * \lambda * d$
38:            if $|\beta| > |e_{residual}|$ then
39:                $\beta = e_{residual}$
40:            end if
41:            {Distribute $\beta$ over $pose_{a+1}$ to $pose_{a_{max}}$ by tree operation}
42:            break;
43:        end for
```

Algorithm 3 Error Propagation (continued)

```
44:         // Loop from last constraint to $C_{max}$
45:         for $C = \{a, b, e, t_{ab}, \Omega_i\}$ from last constraint to $C_{max}$ do
46:             if $C$ is not consistent with $C_{max}$ then
47:                 continue;
48:             end if
49:             $weight_{overlap} = 0$
50:             if $a < b_{max}$ then
51:                 $weight_{overlap} = cmWeight_{b_{max}} - cmWeight_a$
52:             end if
53:             $e_{overlap} = e_{max} * weight_{overlap}/weight_{max}$
54:             $e_{residual1} = e - e_{overlap}$
55:             // Check consistency again
56:             if $e * e_{residual1} < 0$ then
57:                 continue;
58:             end if
59:             $totalWeight = cmWeight_b - cmWeight_a$
60:             $weight_{residual} = totalWeight - weight_{overlap}$
61:             $e_{residual2} = e * weight_{residual}/totalWeight$
62:             // Bound the error conservatively to avoid over optimisation
63:             $e_{residual} = (|e_{residual1}| < |e_{residual2}|)?e_{residual1} : e_{residual2}$
64:             $W = R\Omega_i R^T$
65:             $d = 2We_{residual}$
66:             $\beta = (b - b_{max}) * \lambda * d$
67:             if $|\beta| > |e_{residual}|$ then
68:                 $\beta = e_{residual}$
69:             end if
70:             {Distribute $\beta$ over $pose_{b_{max}+1}$ to $pose_b$ using tree operation}
71:             break;
72:         end for
73:     end for
74: end loop
```

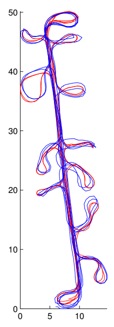
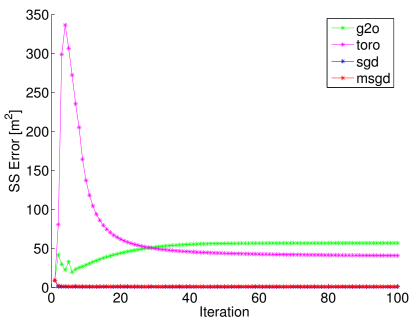
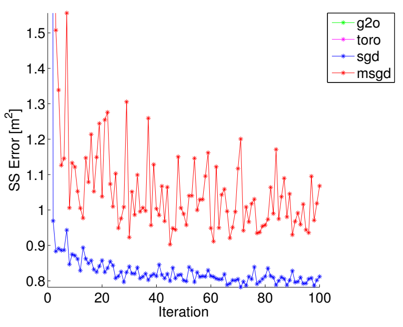
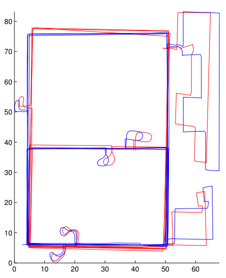
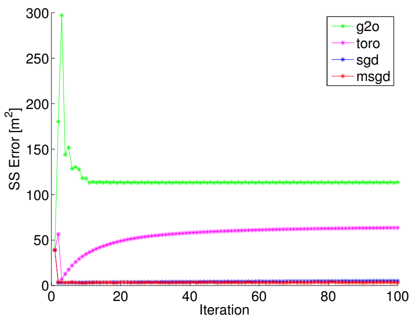
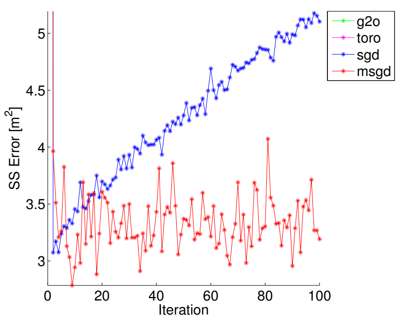
	MSGD Result	State Squared Error	State Squared Error (Zoomed In)
WGB2a-1	 (a)	 (b)	 (c)
WGB2-3	 (d)	 (e)	 (f)

Figure 5.11: MSGD results on WGB2a-1 and WGB2-3. The trajectories (the red is the transformed groundtruth and the blue the slam result) and the SS errors are the results of the 100th iteration.

5.6 Evaluation

We use all the datasets used to test g2o, Toro and SGD to test the optimised SGD algorithm (we refer to it as **MSGD** herein, where 'M' means magnetic mapping or 'mini-batch' [8]). Two sample results of MSGD (100th iteration) are shown in Figure 5.11 (we also repeat the SS errors of SGD, Toro and g2o in these figures for ease of comparison). We show all the results of MSGD in Appendix A (Figure A.9, A.10, A.11 and A.12).

On four datasets (WGB1-1, ENG-2, RUTH-1 and KX-2) MSGD provided similar performance to SGD (Figure A.10c, A.12f, A.12i and A.12r). We should note that the SS errors of both SGD and MSGD vary within a small range after convergence. This is because both of them take stochastic steps in the state space around a minimum. However, the amplitude of the fluctuations in the SS errors of MSGD is larger than that of SGD. This is because MSGD is 'more randomised' than SGD: SGD simply solves every constraint in the system, while MSGD solves only a few selected constraints based on their importance. In different iterations MSGD solves different constraints, which causes different changes to the SS error, and that is where the large fluctuation comes from. But as long as the amplitude of the fluctuation is within a small range (not exceeds 1 m^2 in most cases), the final result is still stable and approaches the groundtruth.

In terms of the SS error, for these five datasets: WGB2a-1, WGB2a-4, WGB1-4, ENG-1 and KX-1, SGD provides better performance than MSGD (Figure A.9c, A.9l, A.10l, A.12c and A.12o). But for these ten datasets: WGB2a-2, WGB2a-3, WGB1-2, WGB1-3, WGB1-5, WGB2-1, WGB2-2,

[8]'Mini-batch' is a terminology used in machine learning area, which means in the training process, a mini-batch of information is used per time rather than a single piece of information nor all the information (the whole batch). In our case, we process a mini-batch of constraints per time instead of process a single constraint or all the constraints together.

WGB2-3, WGB2-4 and RUTH-2, MSGD performs better (Figure A.9f, A.9i, A.10f, A.10i, A.10o, A.11c, A.11f, A.11i, A.11l and A.12l). In terms of the trajectory quality shown on corresponding figures, the difference between these two algorithms are subtle. So, MSGD achieves almost the same or better performance as SGD.

However, MSGD is more scalable than SGD as shown in Figure 5.12. In this figure, *MSGD$^-$ is the MSGD without the constraints selection in Section 5.5.3.2 (with the optimisations described in Section 5.5.2 only)*. This is to show the effects of different optimisations. It shows that with the optimisation by tree and approximation (Section 5.5.2), MSGD$^-$ largely reduces the time cost of SGD. The constraints selection (Section 5.5.3.2) then achieves further cost reduction: full MSGD gains up to 40% improvements in execution time compared with MSGD$^-$. Please note that this improvement also increases as the number of constraints increases, which demonstrates the scalability of MSGD. The full MSGD algorithm is 3 to 8 times faster than SGD and the time per iteration of MSGD increases much more slowly than that of SGD when the number of constraints is increasing. Actually, the computational cost of MSGD is proportional to the number of loop closures but not the number of constraints. A loop closure can add hundreds of loop closure constraints to the pose graph, so MSGD is a more scalable solution for large-scale automated survey.

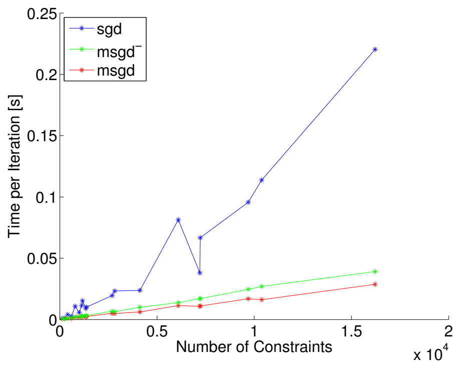


Figure 5.12: CPU time per iteration of SGD, MSGD$^-$ and MSGD. MSGD$^-$ is the MSGD without the constraints selection in Section 5.5.3.2 (with the optimisations described in Section 5.5.2 only). The data of the execution time was obtained by running each algorithm on every dataset shown in Table 5.1. Odometry constraints are discarded.

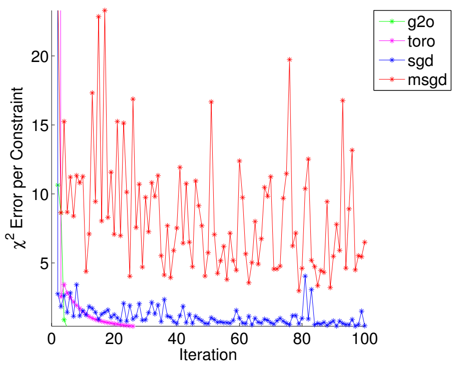


Figure 5.13: χ^2 errors for different algorithms on dataset WGB2-3.

We should also notice that SGD exhibited slight divergence on 10 datasets: WGB2a-2, WGB2a-3, WGB2a-4, WGB1-2, WGB1-3, WGB2-2, WGB2-3, WGB2-4, RUTH-1 and RUTH-2 (Figure A.9f, A.9i, A.9l, A.10f, A.10i, A.11f, A.11i, A.11l, A.12i and A.12l). However, MSGD does not have the same problems even though it has larger amplitude of fluctuation. This can be attributed to the strategy of MSGD again. MSGD solves as few consistent constraints as possible. Only the *consistent* constraints with the most importance and the most coverage (affect more poses than others) are selected as candidates to be solved. In this way, MSGD keeps the interaction between antagonistic constraints small, thus achieving better consistency in the system states after solving a group of constraints within a loop closure. So, the solution will not be pushed away from the minimum area. However, SGD simply solves every constraint within a loop closure regardless their impacts to the stability of the system states, which causes divergence in these results. Please recall that both SGD and MSGD explicitly minimise the χ^2 error instead of the SS error. We take the SLAM results on WGB2-3 as an example. It can be seen that the χ^2 error of SGD (Figure 5.13) gradually converged to zero (with small fluctuation) as expected while the SS error (Figure 5.11f) slightly diverged. However, both the χ^2 error and SS error of MSGD did not show divergence though exhibited fluctuation. This is because when minimising the χ^2 error, MSGD takes the consistency between constraints into account (as described in Section 5.5.3.2, MSGD selects only the consistent constraints in a loop closure group to solve). However, SGD simply picks up a constraint randomly with no consideration about if this constraint is antagonistic to the others. So although the SGD minimises the χ^2 error gradually, the SS error is not necessarily minimised (This confirms with the analysis given in Section 5.4.5.4). That is why SGD exhibited divergence in some datasets.

5.7 Conclusions

We have introduced a novel variant of the SGD algorithm that is optimised for magnetic sequence SLAM, MSGD. It has been demonstrated that although MSGD has slightly larger fluctuations in the SS errors (after convergence) than SGD, they achieve equivalent trajectory quality. MSGD is also found to be more efficient and scalable when dealing with much larger datasets. In addition, MSGD does not have the problem of divergence as the SGD does. Therefore, for large-scale automated survey in indoor environments, we believe MSGD is a better solution.

However, we should note that the SLAM back-end discussed here works without any environmental information (e.g. a floor plan). So, the resultant trajectory may have incorrect scale and might violate the environmental constraints (examples are given in the next chapter). Without a floor plan, this is the best we can get. When we *do* have a floor plan, it can be incorporated to the path survey system so that environment-consistent survey trajectory can be recovered. This case is discussed in the next chapter.

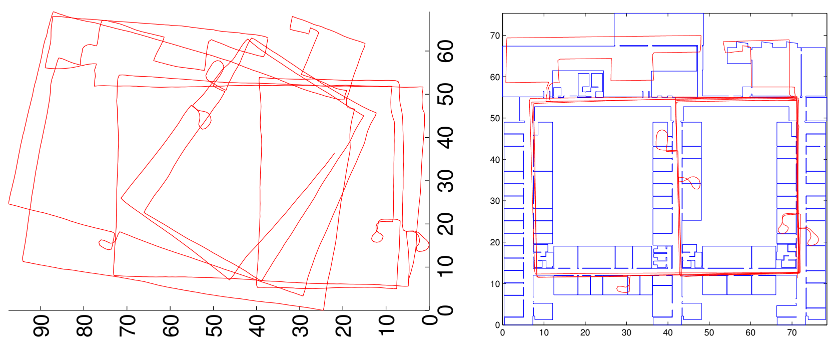
Chapter 6

A robust survey trajectory recovery system

6.1 Introduction

Chapter 4 and 5 have demonstrated a GraphSLAM-based system to estimate a trajectory for a dedicated surveyor when a floor plan is unavailable. The loop closures are detected by a window-based searching scheme that matches similar magnetic sequences recorded along the survey path. However, these loop closures only give information about the spatial relationships between the surveyor's positions at different time points/steps, so the optimised trajectory may have incorrect scale and might violate the environmental constraints. Figure 6.1 gives an example where the SLAM output appears visually correct until manually aligned with the floor plan. We observe that the path crosses the walls and different parts of the trajectory have different scaling errors.

We should note that when a digital floor plan is not available, this is the best result we can get. This chapter, however, deals with the opposite case: if we *do* have a digital floor plan, how can we best recover the survey trajectory? This chapter proposes PFSurvey, a system that uses smartphone accelerometer, gyroscope and magnetometer data to estimate a dedicated surveyor's trajectory post-hoc using Simultaneous Localisation and Mapping (SLAM) techniques and particle filters to incorporate a building floor plan. We show how the survey trajectory can be recovered so that it not only satisfies all the loop closure constraints but also has correct scale and remains consistent with the environment (floor plan).


(a) PDR result.

(b) SLAM result plotted on floor plan.

Figure 6.1: Example scaling errors from SLAM without a floor plan. (a) the raw PDR result, which is the input. (b) the output result after manual alignment to the floor plan.

6.2 Dedicated surveying

As in previous chapter, we emphasise that we assume a dedicated surveyor here. This is motivated by our belief that PDR algorithms are not sufficiently mature to robustly estimate the trajectories of arbitrary multi-purpose devices. Assuming a dedicated surveyor afford us a number of advantages:

- the surveyor will carry the smartphone consistently;
- the surveyor will cover the area comprehensively following best-practice guidelines; and
- a start position can be manually specified

Since a live position estimate is not required, we can afford to use greater computational resources to post process the data.

6.3 Related work

Floor plans have been used to constrain PDR drift. A particle filter is typically used to ensure the trajectory remains consistent with the floor plan. Systems such as [93, 52] provide *instantaneous* location estimates—i.e. at time $t = T$ they sample the probability distribution for the current position based on all measurements and state for $0 < t < T$. The position estimate is usually taken as the weighted mean of the samples.

Here, however, we do not require instantaneous location estimation as the data arrive but rather a best estimate of the trajectory post-hoc. Thus at time $t = T$ we want to estimate the probability distribution given the events at all epochs, even the 'future' ones ($t > T$). Essentially, knowing where we are now may allow a better estimate of where we were, particularly if we were uncertain at the time (e.g. a multi-modal distribution). Particle *smoother* algorithms are commonly used to provide the best post-hoc estimates. For example Fixed Lag Smoothing (FL) [49], Forward Filter Backward Smoother (FFBS) [14] and Forward Filter Backward Simulation (FFBSi) [33]. These involve retaining the particle distributions at each epoch (which can be costly in terms of storage) and reprocessing at various stages (which can be computationally costly). They are discussed in more detail in Section 6.4.

A simpler but less formally correct approach was described anonymously in [18] (DP-SLAM)—we refer to it as particle pruning. An ancestor tree for each particle is retained as before. However, when a particle is not resampled we walk up its ancestor branch, removing any parent particles in previous epochs that have no other child ('pruning'). At the end of the filter, the position at each epoch is computed as the weighted mean of the remaining particles for that epoch. The pruning approach is less resource-intensive but needs a large number of particles to ensure older epochs do not suffer particle depletion.

An alternative post-hoc approach is like the work in Chapter 4, which uses external spatially-variant signals (possibly even those we wish to map) to enable SLAM. The core idea is to search the external observations for evidence of loops in the trajectory (the external observations will return to values recorded earlier in the trajectory). These form *loop closures* that are used to constrain the post-hoc trajectory estimate. The SLAM algorithms are either graph-based (e.g. GraphSLAM [42]) or use particle filters (e.g DP-SLAM [18] and FastSLAM [64]). They have been used when a floor plan was unavailable, giving *unanchored* trajectories.

6.4 Trajectory recovery by state-of-the-art algorithms

6.4.1 Inputs

The primary goal of this work is to produce the best trajectory estimate consistent with and anchored to a floor plan. The simplest solution is to feed the raw PDR trajectory (herein referred to as *PDR-traj*) into a wall-sensitive particle filter and then use a particle smoother to recover the optimal trajectory. An alternative to *PDR-traj* is the SLAM-corrected PDR results as demonstrated in Chapter 4 and 5, which we refer to as *SLAM-traj*.

6.4.2 Methods

The particle filter algorithm approximates the system state x given all the measurements y up to time step t by a set of particles and weights $\{x_t^i, w_t^i\}_{i=1}^N$. Each particle holds a hypothesis about the state x_i with a weight w_i associated. By incorporating the floor plan and treating the *PDR-traj* or *SLAM-traj* as a sequence of *step events*, a particle filter can track incrementally the position of the surveyor during the whole survey process, and thus align the survey trajectory to the floor plan. In the proposed scenario of signal survey, the initial state (position and heading) of the surveyor is unknown. So a large number of particles are spread over the environments (over the whole floor or the room/corridor where the survey starts at) to initiate the particle filter. As a step event is reported (from the *PDR-traj* or *SLAM-traj*), a new generation of particles is sampled. The states of the particles in the new generation are estimated by '*moving*' one step from their original states (i.e. their parents' states) according to the step event. The new sates will be weighted based on how well their observations and states match. For example, with the incorporation of the floor plan, a particle that moves across a wall will be given a weight of 0 (killed), because it violates the fact that a wall is impassable. After that, another step event is reported and a newer generation of particles will be sampled and weighted...

A particle filter estimates the posterior distribution of state x_t given observations z up to time step t:

$$p(x_t|z_{1:t})$$

which is most tracking applications care about. But in the signal survey case, we need to estimate the surveyor's positions (system states) at all time steps, i.e. to recover the whole survey trajectory. So what we need to know is the system state at an arbitrary time step given all the measurements taken during the survey. Actually, this is the marginal smoothed distribution:

$$p(x_t|z_{1:T}) \text{ with } T > t$$

Typically, a particle smoother achieves this goal. There are many variants of particle smoother algorithm [79, 15]. To evaluate, we select the most commonly used ones in indoor localisation or trajectory generation applications [68, 49, 77]:

1. **Fixed Lag Smoothing (*FL*)**. This particle smoother (also called *Backtracking* [49]) approximates the marginal smoothing density $p(x_{t-L}|z_{1:t})$ by the weights at the latest time step t:

$$p(x_{t-L}|z_{1:t}) \approx \sum_{i=1}^{N} w_t^i \delta(x_{t-L} - x_{t-L}^i)$$

This smoother is trivial to implement but the value of L is difficult to decide. Small L tends to give poor approximation while large L causes degeneracy problem (because most particles at time step t may share common ancestors). Empirically, a value between 20 to 50 was found to achieve good performance [15].

2. **Forward Filter Backward Smoother (*FFBSm*).** This is a less degenerate particle smoother [14], which is also called *Reweighting Particle Smoother* or *Marginal Particle Smoother*. It approximates the smoothing distribution by reweighting the particles as follow:

   - Start by setting $w^i_{T|T} = w^i_T$ for $i = 1, 2, ..., N$.
   - From time step $t = T-1, T-2, ..., 0$, new weights are computed as

   $$w^i_{t|T} = \sum_j w^j_{t+1|T} \frac{w^i_t p(x^j_{t+1}|x^i_t)}{[\Sigma_l w^l_t p(x^j_{t+1}|x^l_t)]}$$

   After the reweighting, the marginal smoothing distribution can be approximated as

   $$p(x_t|z_{1:T}) \approx \sum_{i=1}^{N} w^i_{t|T}\delta(x_t - x^i_t)$$

   An obvious disadvantage of FFBSm is the high computational complexity, which is $O(N^2T)$.

3. **Forward Filter Backward Simulation (*FFBSi*).** This is suggested in [33], which approximates the smoothed distribution by simulation of M trajectories backwards. A single trajectory is simulated as follows:

   - Choose $\tilde{x}_T = x^i_T$ with probability w^i_T
   - For time step $t = T-1, T-2, ..., 0$
     (a) New weights are computed as

     $$w^i_{t|t+1} = w^i_t p(\tilde{x}_{t+1}|x^i_t)$$

     (b) Choose $\tilde{x}_t = x^i_t$ with probability $w^i_{t|t+1}$

   M iterations result in M sample trajectories that approximate the smoothing distribution with equal weights. The computational cost of FFBSi is O(MTN). An efficient implementation of FFBSi algorithm is proposed in [13].

In addition to particle smoothers, a low-cost approach as mentioned in Section 6.3 is the ***particle pruning***:

- Start by setting

  $$\tilde{S}_T = S_T = \{x^i_T, w^i_T\}, i = 1, 2, ..., n$$

  .

- For time step $t = T-1, T-2, ..., 0$,
  - Setting $\tilde{S}_t = S_t$.
  - For each particle p in $\tilde{S}_t$, if there is not a particle q in $\tilde{S}_{t+1}$ such that q is p's child, then delete p from $\tilde{S}_t$.

The basic idea of this strategy is that if a particle is not being re-sampled, it is less likely that it holds the true hypothesis about the system state. So we iteratively remove parent particles without any child. Particle pruning can be integrated into the filtering process to dynamic prune the particles at each time step, which can greatly save memory usage.

After smoothing or pruning, the system state – the position of the surveyor – at each time step can be estimated by the weighted mean of the smoothed (or pruned) particle cloud.

6.4.3 Result analysis

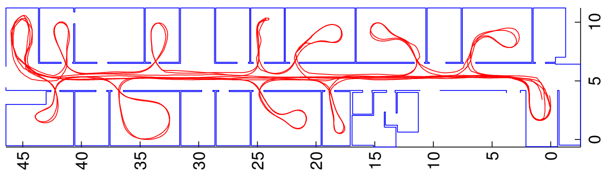


Figure 6.2: The groundtruth of *Path-1*.

The left column of Figure 6.3 shows the results when using *PDR-traj* as the input of conventional particle filter and smoother (or pruning). The raw PDR-traj path exhibits typical drift issues seen when walking a path such as the ground truth (Figure 6.2, obtained using a high accuracy ultrasonic positioning system [1]). The error is sufficiently high that various smoothers give significantly different trajectory results (Figures 6.3b–6.3f). It is shown that FL smoother improves the non-smoothed result better than both FFBSM and FFBSi do. This is partially because we choose a small M for FFBSi and use much less particles for FFBSm due to the high computational costs. Moreover, the trajectories are not truly consistent with the floor plan, since they cross walls (or enter wrong rooms) at various points. The explanation for this can be seen in Figure 6.3g, which shows an instantaneous particle distribution corresponding to the point marked with a blue dot in the preceding images. The high PDR uncertainty results in multi-modal distributions spanning multiple rooms. The position estimate is a weighted average of these particles and so can cross walls. This *room ambiguity* is very serious for a path survey: a small perturbation to any of these systems could easily result in signal data being assigned to the incorrect room and the subsequent radio map containing serious errors.

A natural adaptation of this system would be to take the SLAM-corrected trajectory (i.e. *SLAM-traj*) as input. The right column of Figure 6.3 illustrate this idea. Compared to *PDR-traj* (with both heading and scaling errors) the *SLAM-traj* has much lower heading noise as can be seen in Figure 6.3h. However, the scale errors persist and we typically find that the final result is only marginally better. For these runs we see that only the FL smoother was able to correctly recover the path (in fact part of the trajectory still penetrates walls if observed carefully, but the error is negligible). However, closer inspection of the particles at the position marked with a blue dot in the smoother outputs shows that the distribution was still multi-modal (Figures 6.3g and 6.3n). This is more obvious for longer walks such as those in Figure 6.4, where the FL smoother produced poor results (highlighted in magenta) when faced with multi-modal distributions.

6.5 Motivations

We believe that a fundamental reason for the ambiguities in the *PDR-traj*-based filtering results is the lack of sufficient constraints to distribute weights reasonably over the particles. The wall-sensitive particle filter and smoother kills all particles violating environmental constraints and gives all the surviving particles the same weight, e.g. a weight of $\frac{1}{N}$ for number of surviving particles, N. Thus, all surviving particles have equal probability of being re-sampled no matter how likely it is they hold the true hypothesis of the system state. For the results in Figure 6.3 and 6.4, the particle clusters spread in the wrong locations have similar weight sums with the particle clusters in the correct place. This causes ambiguities and incorrectness in the final result. While loop closures can assist by limiting the drift (i.e. keeping the particle clouds small), the common techniques used to process them cannot incorporate floor plan constraints.

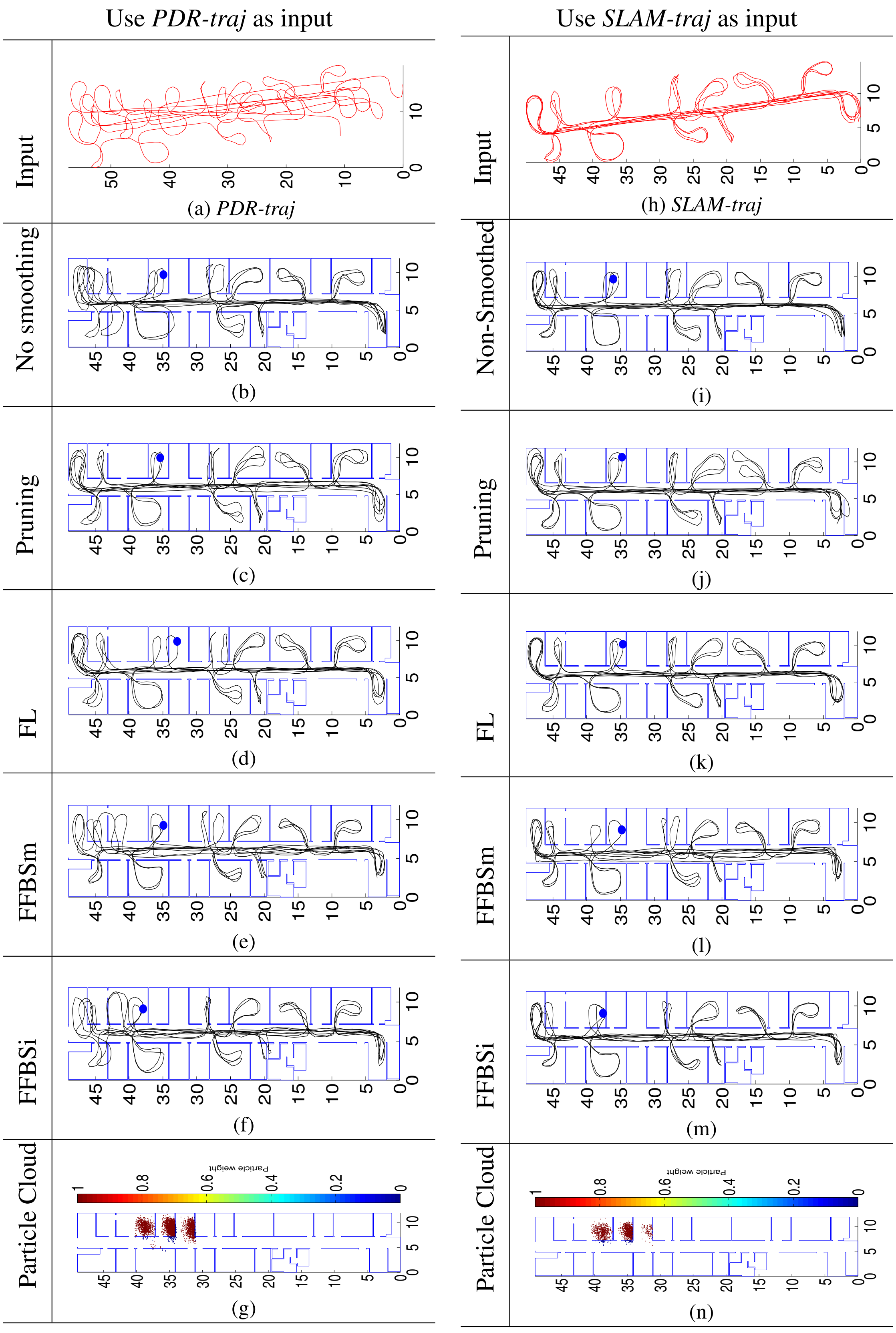


Figure 6.3: Example outputs of the conventional particle filter plus smoother approach. The groundtruth trajectory is shown in Figure 6.2. The lag L was set to 50 for the FL smoother. M was set to 100 for FFBSi.

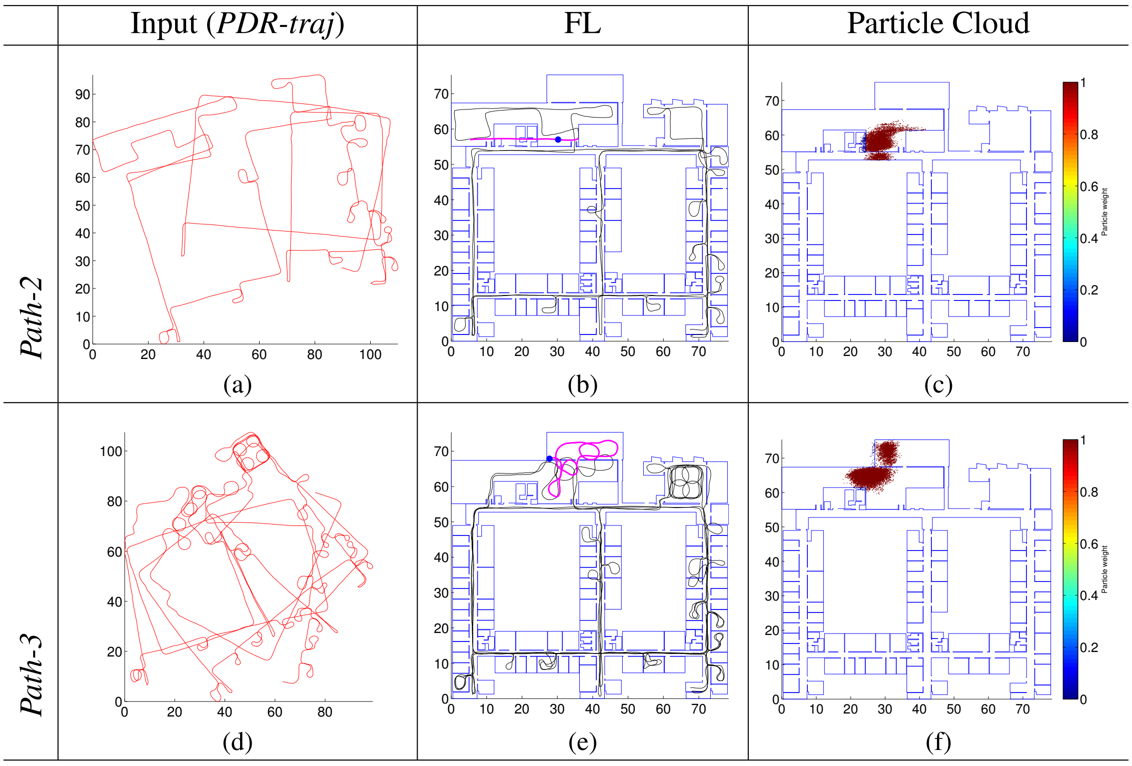


Figure 6.4: Example outputs of the conventional particle filter plus smoother approach on longer walks.

To solve this problem, the proposed PFSurvey attempts to use both the floor plan and loop closures simultaneously to produce a more robust and accurate trajectory estimate. The core fusion process is a particle filter, but a series of pre-processing steps are necessary to create suitable loop closures.

Chapter 4 shows how the loop closures can be detected by looking for similarities in magnetic sequences. It also shows that a pure signal similarity based detection method can produce lots of false positive loop closures (Figure 4.2). Therefore, it is critical to validate the detected loop closures. Section 4.4 shows how they can be validated by comparing the closeness, topological similarities, etc. between the corresponding path segments. This method requires the topology of the *PDR-traj* to be good enough to help distinguish between true and false loop closures. However, the *PDR-traj* can be heavily distorted because of the gyroscope bias errors. Hence, a straight-line filter was applied to the *PDR-traj* to improve its topology before loop closure detection (Section 4.4). This straight-line filter naïvely assumes all straight-line segments are either parallel or vertical to each other because no environmental information is available. It cannot deal with more complicated survey path and environments where this assumption does not hold. Also, it does nothing but straighten segments that appear to be almost straight. So, large drifts still exist in the filtered trajectory. These can cause confusions and difficulties in the topology-based loop closures detection/validation: a strict parameter setting [1] results in many false negative loop closures while a more relaxed parameter setting produces many false positive ones. Also, an empirical parameter setting might not work for different cases because the drifts in the PDR results are quite unpredictable (could range from 2 to 10 metres or even larger).

With the availability of the floor plan, we propose a better loop closure detection method, which uses a wall-constraints-only particle filter to improve the topology of *PDR-traj* at the very beginning. This pre-processing can be kept low-cost because the time complexity of a particle filter is (generally)

[1]For example, we can assert that any two segments which are more than n meters away cannot contain true loop closures, then a very small value of n is a strict parameter setting and a very large n is a relaxed parameter setting.

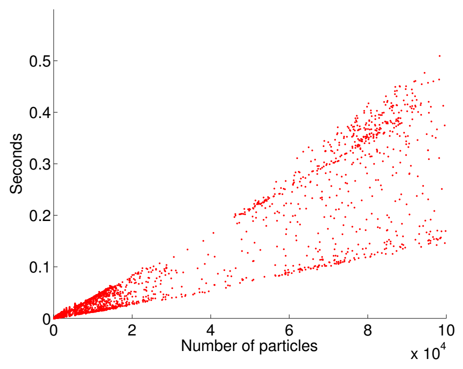


Figure 6.5: Relationship between execution time and number of particles when applying particle filter on the *PDR-traj* shown in Figure 6.3a.

proportional to the number of particles it uses in the filtering process. Figure 6.5 shows the relationship between the execution time and the number of particles of a typical particle filter algorithm. The results were got by running the particle filter multiple times using different numbers of particles. It can be seen that when the particle number is as low as hundreds, the cost to run a particle filter is minimal for an offline survey trajectory recovery system (less than 5 seconds for the 10-minute survey walk *Path-1* with about 950 steps as shown in Figure 6.3a)[2]. With this minimal cost, although the filtering result is still incorrect (we can take the result shown in Figure 6.3d as an example, although it uses much more particles than hundreds), it *does* have a better topology than the original PDR result, which is more suitable for loop closure detection/validation. For example, the distances between path segments that could possibly contain loop closures are all within a reasonable range which can be captured by an empirical threshold (e.g. $2 \sim 3$ metres). Inspired by this, we run a light-weight particle filter to improve the *PDR-traj* as the very first step, and apply an improved loop closure detection/validation scheme later. This scheme greatly increases both the quality and quantity of the detected loop closures. These loop closures will then be used by another particle filter to formally recover the survey trajectory and give robust results.

Please note that the major motivation of the proposed system is to solve the room ambiguity problem in the particle filter results. The basic assumption is that the filtering process succeeds in the trajectory recovery (the particle cloud holds the true hypothesis of the system state at any given time point). It is possible that the particle filter fails during the filtering process, e.g., all the particles are killed because of violating the wall constraints. An important reason is that the number of particles being used is insufficient so that the particle cloud cannot well approximate the true probability distribution of the system state. In this case, restarting the particle filter with more particles (through tuning the parameters which control the number of particles) can help solve this problem. However, the computational resource is not unlimited, so the trade-off between success rate and computational cost needs to be made. A feasible method for higher efficiency is to use discrete system state space, i.e., discretise the floor plan [98]. But in this case another trade-off between the discretisation granularity (e.g., discretise the floor plan to be $1 * 1\ m^2$ or $2 * 2\ m^2$ grids) and the tracking accuracy (larger granularity typically gives lower tracking accuracy) needs to be made. In our experiments, particle filters can always succeed when using a reasonable error model and sufficient particles. So this thesis does not deal with the failure case, and focuses on solving the tracking

[2]The particle filter was implemented in C++ and the laptop used for this experiment has a 2.8 GHz Intel Core i7 CPU and 16 GB memory. The execution time shown here is the time used to run a single filtering process ("Re-sampling - Propagation - Correction" as described in [93]) for a step event. To estimate the total run time needed for a whole survey walk, simply multiply the execution time shown in this figure by the number of steps taken during the walk.

ambiguity problem only.

6.6 System overview

The system architecture is illustrated in Figure 6.6 and contains a series of components:

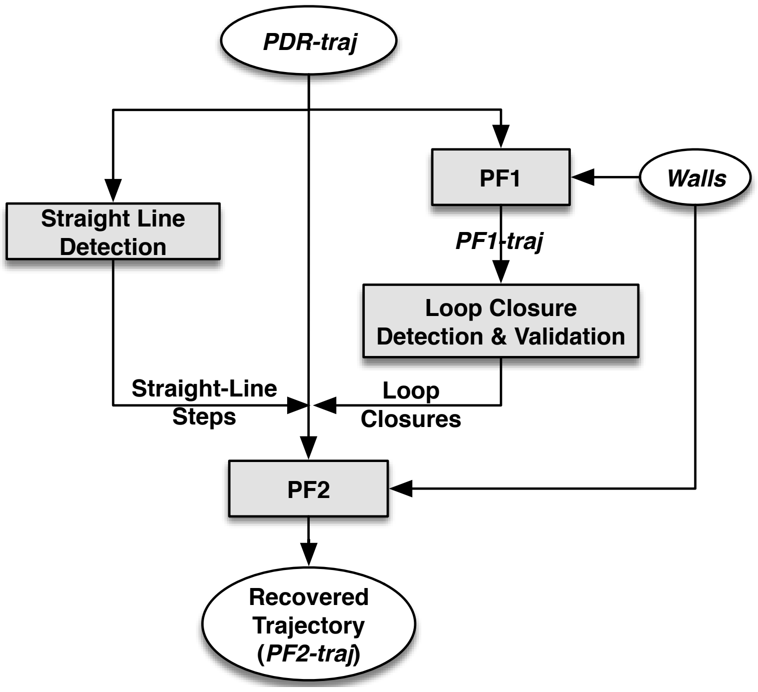


Figure 6.6: The work flow of the proposed trajectory recovery system.

1. **Particle Filter 1 (*PF1*)**. This step takes the *PDR-traj* as input, runs a wall-constraints-only particle filter to improve the topology of *PDR-traj*. This is to bound the PDR drift using the floor plan to reduce heading errors. We denote the resultant trajectory of this step as *PF1-traj*.

2. **Straight line constraints**. We typically build our environments to be rectilinear and thus tend to move in straight lines. However, in the PDR results straight line steps often bend due to bias errors from the gyroscopes. We use a simple threshold-based method to identify straight-line steps in the PDR results. Then we weight particles in PF2 according to how well their headings match the orientations suggested by the environments (rooms or corridors).

3. **Loop closure detection & validation**. This step detects and validates loop closures using the partially-corrected *PF1-traj*. Without the initial PF1 pass, robustly identifying true loop closures is often all but impossible. We provide details of our detection and validation scheme in Section 6.9.

4. **Particle Filter 2 (*PF2*)**. This step adopts a customised particle filter to produce an accurate and correct survey trajectory that is consistent with the environment. It uses the wall, straight-line and loop closure constraints to weight the particles using PDR-traj as input. We denote the resultant trajectory of this step as *PF2-traj*, which is the final output.

The design of PF1 and PF2 were largely inspired by previous work [93]—a particle filter-based tracking in 3D environments using a foot-mounted IMU. This foot-mounted IMU can accurately measure the height changes in the step events, which is a critical contributor to the convergence and accuracy of the particle filter. We will show that without the accurate 3D movement measurements, we are still able to achieve good performance. In our proposed system, PF1 adopts a framework that is very similar to the particle filter described in [93, 96]. PF2 is based on PF1 but has two more kinds of constraints incorporated in the weighting process. Both PF1 and PF2 adopt a particle pruning strategy to largely ease the cost of particle smoothing. The following sections describe the PF1, PF2 and the loop closure detection/validation methods in detail.

6.7 PF1

The primary aim of PF1 is to correct large heading errors using the floor plan. PF1 is based on the filter described in [93, 96]. We adapt it to use 2D step vectors from handheld smartphones, which are more noisy than the inputs used in the original work. In particular the step length is unknown. We represent a step event as $m_i = (l, \delta\theta_i)$, where l is a fixed step length of 0.75 m and $\delta\theta_i$ is the heading change of the surveyor during this step as estimated by the gyroscope. The error models for these two components are assumed to be independent and Gaussian:

$$\begin{aligned} e_l &\sim \mathcal{N}(0, \sigma_l^2) \\ e_{\delta\theta} &\sim \mathcal{N}(0, \sigma_{\delta\theta}^2) \end{aligned} \tag{6.1}$$

We set the uncertainty in the heading change $\sigma_{\delta\theta} = 0.5°$ and the step length uncertainty to $\sigma_l = \lambda l$[3]. We set $\lambda = 0.5$ to capture our lack of knowledge of the true step length.

We use the KLD adaptive resampling algorithm [27, 93] to dynamically vary the number of particles appropriately at each step. The parameters used for KLD were set empirically: bin sizes $\Delta_x = \Delta_y = 2.0\ m$ and $\Delta_\theta = 30°$; bounding parameters $\delta = 0.01$ and $\epsilon = 0.0109238$; and $n_{min} = 504$. With these parameters PF1 typically uses hundreds of particles and can finish within 3$\sim$5 seconds for a 10-minute survey walk (about 950 steps), assuming the surveyor provides the initial room or corridor (a reasonable expectation for a dedicated surveyor).

To generate an output path we apply the pruning smoother. Since we do not care about room ambiguities in the output of PF1, it offers the least resource-intensive solution. Other smoothers could be applied but would come at additional complexity for no particular accuracy gain.

The result of applying PF1 on the *PDR-traj* in Figure 6.3a is shown in Figure 6.7. This improvement in topology can greatly facilitate the work of the loop closure detection and validation.

6.8 Straight line filter

This component first identifies candidates for *straight-line steps* from the original PDR-traj. It uses a simple threshold-based method: we identify consecutive steps where the turning angle is less than 5° to form candidate straight line steps. Where there are 10 or more candidates in a row, we assert that they are all straight line steps. The remaining candidates are discarded. Figure 6.10d, 6.10e and 6.10f show examples of identified straight-line steps (highlighted in blue) in *PDR-traj*.

[3]Our step variance is proportional to the step length, although that quantity is a constant here.

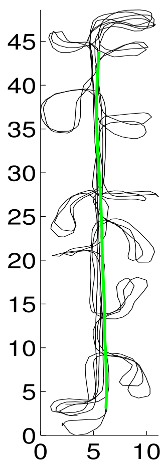
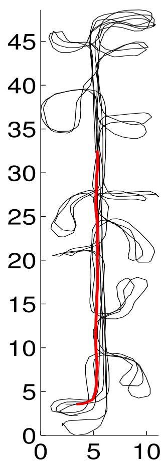
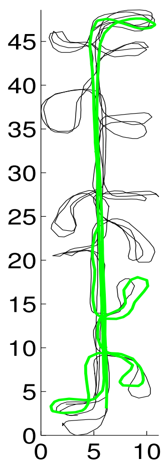
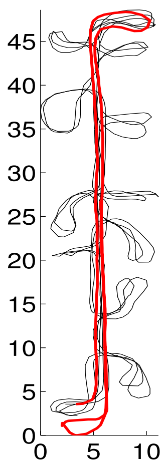
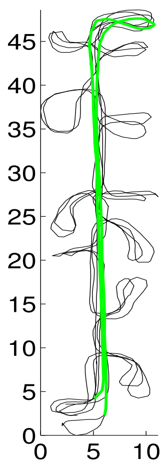
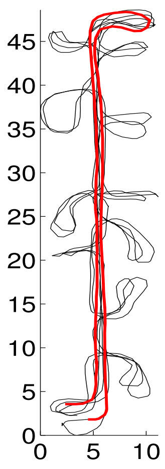
Segment Pair	Maximum Segment Pair 1	Maximum Segment Pair 2
 (a) Segment A.  (b) Segment B.	 (c) Segment A.  (d) Segment B.	 (e) Segment A.  (f) Segment B.

Figure 6.7: Examples of *PF1-traj*, segment pair and maximum segment pair. The black trajectories are the *PF1-traj* generated by applying PF1 on the *PDR-traj* in Figure 6.3a. A segment pair and two maximum segment pairs are shown in the *PF1-traj*. Each segment pair/maximum segment pair consists of two segments A and B, which are marked in green and red respectively. The segment pair shown in the first column is contained by the Maximum Segment Pair 1 in the second column.

6.9 Loop closure detection and validation

For optimal results, this component must detect a sufficient number of true-positive loops in the trajectory without also detecting many false-positive loops. In principle loops can be detected directly from the PDR-traj (e.g. by looking for the same external signal values at different times). This leads to a large number of false positive closures since signals can reasonably adopt the same values at different spatial locations. We seek instead to find loop closures that link parts of the estimated trajectory that are already spatially close. This is the motivation for the PF1: it corrects the large heading errors that result in even true loop closures being spatially far apart (and hence difficult to distinguish from false positive closures).

We have developed closure detection algorithms based on monitoring *sequences* of magnetic readings (Chapter 4). Magnetic signals are ideally suited to the task: they have strong variance over space and the sensors are low power with frequent updates. We have not found them to be a good signal to map for subsequent localisation because they are very easily influenced by small changes to the environment and hence transient in nature. But during a single dedicated survey walk lasting minutes not hours, the signal is stable. The technique proceeds as follows:

1. **Generate PF1-traj** as in Section 6.7.

2. **Maximum Segment Pair (MSP) search**. A segment is any *consecutive* part of the position sequence, $s = \{i_s : i_e\}$, where i_s and i_e are the start and end indices in *PF1-traj*. We process *PF1-traj* to find *segment pairs*, (s_1, s_2), which are spatially close and hence candidates for being loop closures (we verify this latter property in the next point).

   Clearly an arbitrary segment could be contained within another (e.g $s_j = \{i_4, i_6\}$ is contained by the longer $s_k = \{i_4, i_9\}$). To avoid duplicating effort in subsequent steps we wish to find the Maximum Segment Pairs (MSPs), which are simply the segment pairs containing the longest

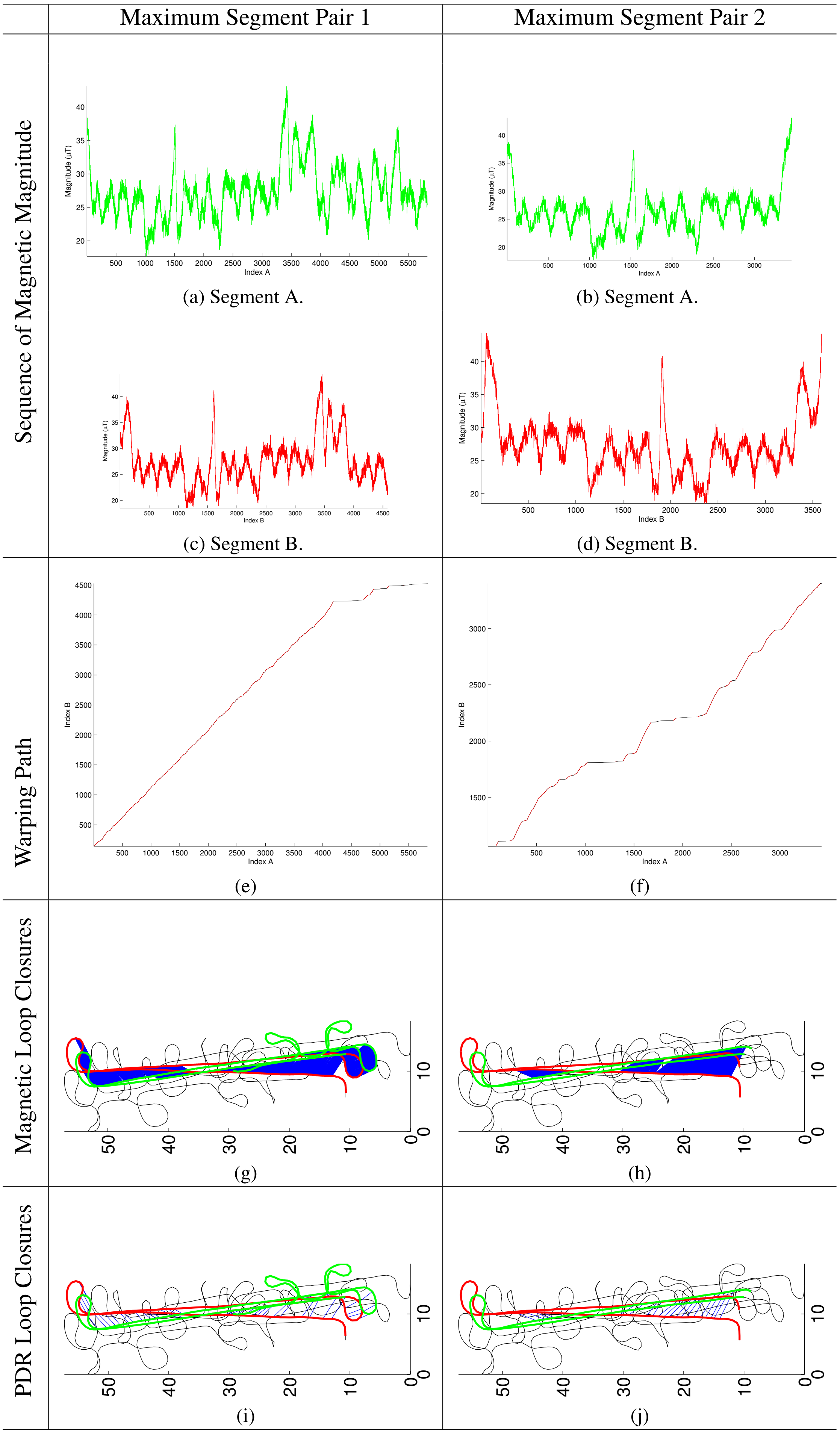


Figure 6.8: Loop closure detection/validation examples for two maximum segment pairs in Figure 6.7.

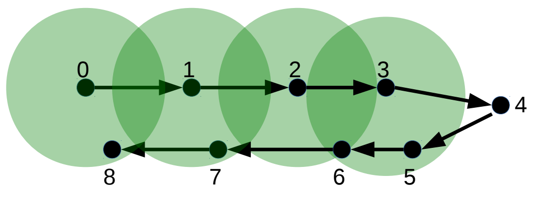


Figure 6.9: Illustration for MSP finding algorithm.

segments possible without their elements violating the spatial proximity rule. Examples are given in Figure 6.7.

Our algorithm for finding the MSPs is described briefly here for the situation illustrated in Figure 6.9, which shows a short 9-step trajectory. We first visit each index of *PF1-traj*, linking it with all other indices that lie within a distance, R (shown as green circles for the first few steps). We then make a sequential pass over *PF1-traj*, checking the linked indices to see if they are increasing (or decreasing) in sequence, indicating they run in parallel. In the example, we move from 0 to 3 and simultaneously observe 8,7,6,6,5 (we observe 6 twice due to variance in the step length causing it to be in range of multiple earlier steps). Thus we find an MSP $\{\{0:3\},\{8:5\}\}$

3. **Sequence-based magnetic loop closure detection**. Having found an MSP, we must then determine the loop closures it contains. We do so using the magnetic signal strength. We associate each magnetic measurement with a PF1-traj position using time interpolation.

   Figure 6.8a and 6.8c, Figure 6.8b and 6.8d are typical examples of magnetic signals in MSPs. We seek to align the waveforms to get point-to-point correspondences (loop closures). Since the two segments within an MSP may be of different length, we apply Open-Begin-End Dynamic Time Warping (OBE-DTW) with an 'asymmetric' step pattern [32] to find these correspondences. OBE-DTW compresses or stretches the time series to create a 'warping path' between the segments. A point (i, j) on the warping path means the i^{th} element of M_1 matches to the j^{th} element of M_2 (Figure 6.8e and Figure 6.8f). A horizontal segment in the warping path means DTW has either stretched one of the signals to fit (accounting for a speed difference) or mapped a chunk of the segment to one value on the the other segment since that chunk does not match anything. The latter situation leads to false positive closures.

   We therefore filter the warping path, splitting the segments into sub-segments at each horizontal part of the warping path. For example, several sub-segments are created from Figure 6.8e and Figure 6.8f respectively (all shown in red). The sub-segments are carried forward as possible matchings (sequences of loop closures).

4. **Closure validation**. The closure detection algorithm produces a large number of potential closures based solely on the magnetic observations. We apply a series of spatial criteria to reject matched subsequences, N and M, that we are not confident in. The criteria are based on empirical constants chosen to be aggressive in culling closures—we would rather have a few true positive closures than a lot of true positives mixed with false positives. As such the constant values are not particularly sensitive.

   - *Either $length(N)$ or $length(M)$ must be larger than 2.5 metres* (about 3~4 human steps). We expect that at least 3~4 steps are contained in a valid loop closure sequence.

- Assume $length(N) > length(M)$, then *the* $\frac{length(N)}{length(M)}$ *must be less than 2.0*. This is to ensure the magnetic time series are not over compressed (stretched) because speed changes are not expected to be great.
- *M and N must be physically close.* Because PF1 has corrected the trajectory to within some scaling errors we do not trust any any instances where M and N are not physically close. To assess this we compute the mean spatial distance between the matched points of M and N on PF1-traj, D_{mean}. We require this value empirically to within $3.0\ m$.
- Additionally we expect the shapes of the two segments to be similar. To capture this we use the variance of the distances between the matched points, D_{var}. We require this value empirically to lie within 1.0 m^2. Please note that more advanced methods to compare the shapes of two segments could be used as discussed in Section 4.4.2.

5. **Closure step mapping**. The final step is to take the dense set of validated closures (Figure 6.8g and 6.8h) and map them to closures at the detected step times—i.e. map the closures to links between steps (Figure 6.8i and 6.8j) rather than arbitrary points along the trajectory. This can be done using interpolation by time.

6.10 PF2

The previous stages can be seen as pre-processing for this major pass with a particle filter. The filter is an extended version of PF1 where we seek higher accuracy output. To that end we adapt the KLD resampling parameters such that $\Delta_x = \Delta_y = 0.5\ m$, $\Delta_\theta = 1°$. and $n_{min} = 16433$. We must also incorporate the straight line and loop closure constraints at the particle reweighting stage. The full reweighting procedure for particle $p_{i,t}$ incorporating step s_t is:

1. Initialise the particle weight to 1: $w_{i,t} = 1$
2. If the step crosses a wall, set $w_{i,t} = 0$ and return.
3. If s_t is marked as a straight-line step, find the room/area wall that is closest to parallel to the step direction. We model the acute angle α between this wall and the step vector as a random variable drawn from a folded normal distribution with mean 0 and variance σ_α^2. We then multiply the particle weight by the probability of the measured α:

$$w_{i,t} = w_{i,t} \cdot Trunc\mathcal{N}(\alpha \mid 0, \sigma_\alpha^2) = w_{i,t} \cdot \frac{\sqrt{2}}{\sigma_\alpha \sqrt{\pi}} e^{-\frac{\alpha^2}{2\sigma_\alpha^2}}$$

   where σ_a is set to 2.5°, which is half of the turning angle threshold for straight line detection.

4. If s_t is associated with a loop closure we compute the Euclidean distance d between the new particle position and the other point specified by this loop closure. We model this distance using a folded normal distribution with mean 0 and variance σ_d^2. We multiply the weight by the probability of the observed distance:

$$w_{i,t} = w_{i,t} \cdot Trunc\mathcal{N}(d \mid 0, \sigma_d^2) = w_{i,t} \cdot \frac{\sqrt{2}}{\sigma_d \sqrt{\pi}} e^{-\frac{d^2}{2\sigma_d^2}}$$

   where $\sigma_d = 1.0$ m empirically.

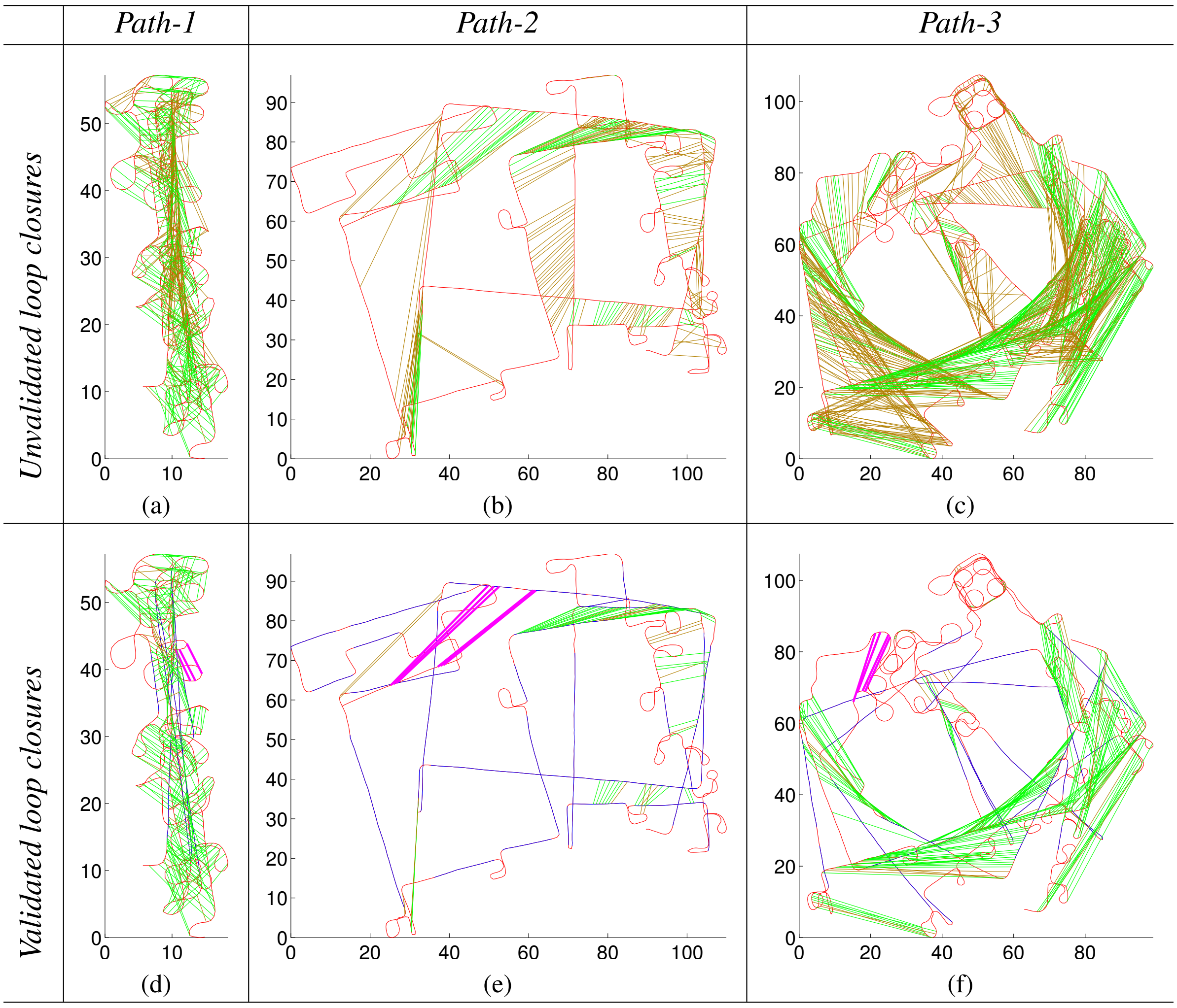


Figure 6.10: Loop closure validation.

Since the loop closures remove the room ambiguities, the PF2 output does not require a complicated smoother. Here we use the simple pruning approach again. More advanced smoothers can be applied, although at significant additional cost for marginal or no gain in our experience. With this approach PF2 typically finishes within 2.5 minutes for a 10-minute survey walk (about 950 steps).

6.11 Evaluation

The data used to demonstrate and evaluate this work were collected in the William Gates Building, a three-storey office building at the University of Cambridge, UK. Data collection was done using a consumer Android smartphone that logged WiFi and BLE scan results, along with the accelerometer, gyroscope and magnetometer sensor values (Section 3.3). A variety of path surveys were carried out in the building, of which three are used throughout this work: *Path-1* covered a single corridor where high accuracy ground truth location was available; while *Path-2* and *Path-3* cover the entire floor but with only coarse ground truth available (the sequence of rooms visited and the actions performed in each were recorded).

6.11.1 Loop closure detection and validation

Figure 6.10 illustrates the loop closures detected from the *PF1-traj*, drawn onto the *PDR-traj* (since this represents the input to PF2). Both unvalidated and validated loop closures are shown in the top and bottom row respectively to demonstrate the need for the validation. Note that the classification of a loop closure was done post-hoc using the final trajectory—it would not be known at this stage. In the images, green lines represent true-positive closures while brown lines denote false-positives. The validated row also shows detected straight-line steps in blue and highlights some true-positive loop closures that are considered in more detail in Section 6.11.2.

	Path-1		*Path-2*		*Path-3*	
	Before	After	Before	After	Before	After
True	304	284	84	68	280	196
False	217	37	158	21	565	53
Ratio	0.58	0.88	0.35	0.76	0.33	0.79

Table 6.1: Statistics on loop closures before and after validation.

Table 6.1 summarises the statistics on loop closures for the three paths. We observe that the validation algorithm was highly conservative as intended; it rejected a large number of closures. In all cases it significantly boosted the percentage of true loop closures above 0.7 as intended. False positives were thus moved to a minority and could not adversely impact the results (which are shown in the next Section 6.11.2).

6.11.2 Trajectory outputs, particle clouds and room ambiguities

Figure 6.11 shows some sample outputs from PFSurvey for the three paths. The top row shows the estimated trajectory in black, with a magenta highlight corresponding to the magenta loop closures in Figure 6.10. The bottom row shows the probability distribution corresponding to the positions marked with a blue dot in the top row. Note these positions match those used to generate Figure 6.3g, 6.3n, 6.4c and 6.4f, which gave multi-modal distributions spanning multiple rooms.

Considering *Path-1* first: we see that the addition of loop closures has produced a uni-modal distribution that means there is no room ambiguity (Figure 6.11d). The availability of accurate ground truth for *Path-1* also allows us to plot the error CDF for this run of PFSurvey and a typical run of the conventional filter plus smoother (Figure 6.12). We observe that the PFSurvey result is more accurate in general: 1.1 m rather than 1.4 m 90% of the time. Note that the CDF is in some senses misleading sine it does not capture the room ambiguity clearly.

The outputs of PFSurvey for the larger scale Path-2 and Path-3 are also shown in Figure 6.11. Although these survey walks did not have accurate ground truth available, the estimated paths are visually indistinguishable from the paths taken and enter no erroneous rooms, unlike their equivalents in Figure 6.4. The magenta-highlighted loops in Figure 6.11c result from walking around some large tables that were *not* in the floor plan. We subsequently measured the positions of the tables and we show them overlaid with the estimated trajectory in Figure 6.13 to illustrate the quality of the result from PFSurvey.

6.11.3 Signal map results

Figure 6.14a shows the *Path-1* input to the signal map generation process for a particular WiFi access point. The focus of this work is the generation of this input, but we show a typical regressed

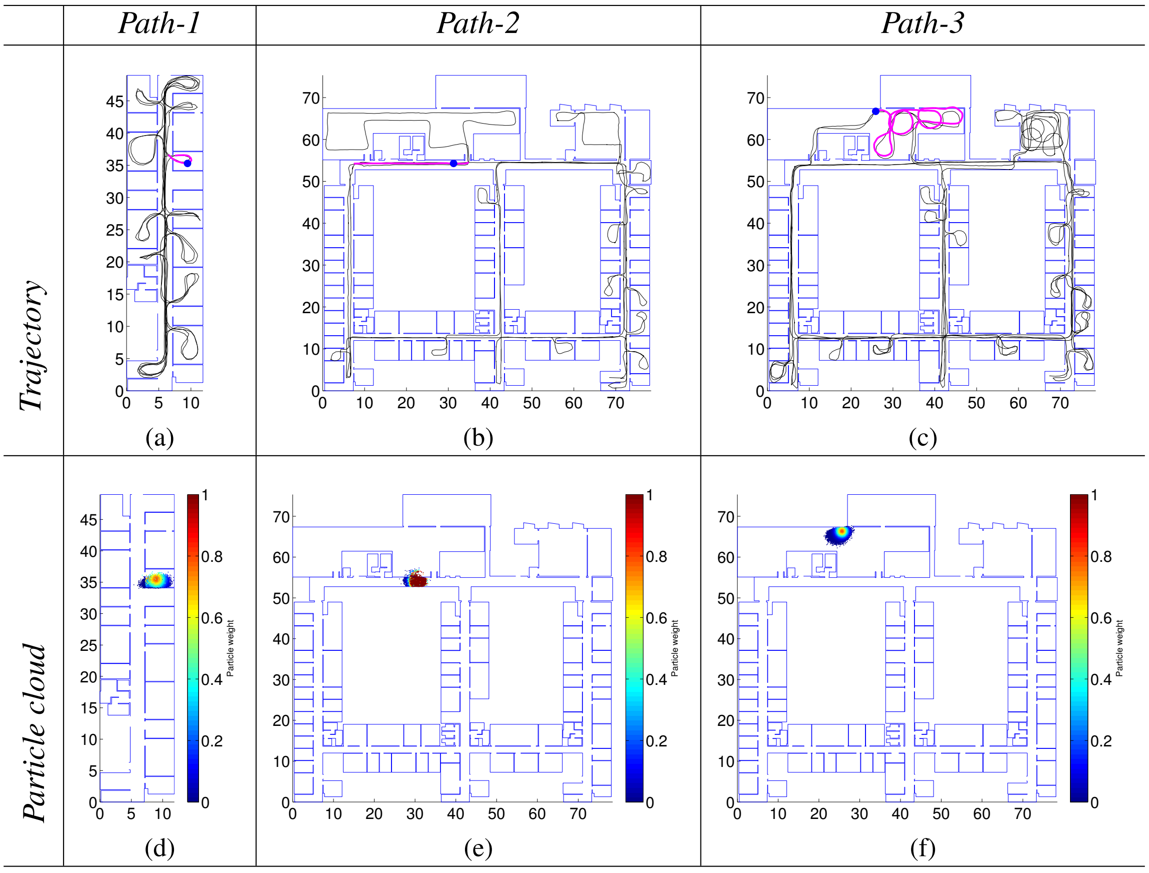


Figure 6.11: The results of the proposed system on the path *Path-1*, *Path-2* and *Path-3*.

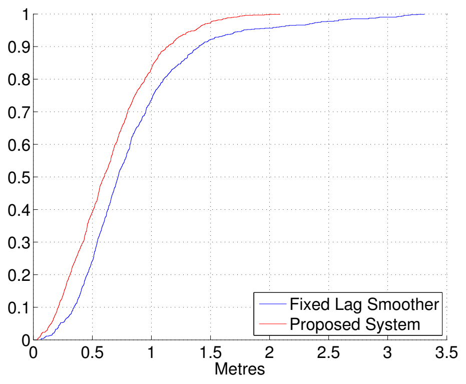


Figure 6.12: The error CDFs when applying a FL smoother and our proposed system on *Path-1* data. Mean errors are 0.83 and 0.65 metre respectively. The errors are measured by the high-accuracy Bat system.

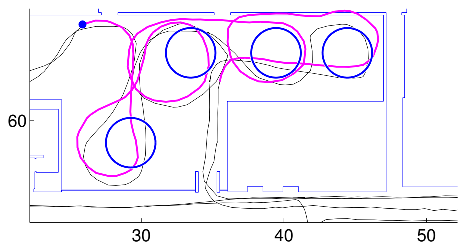
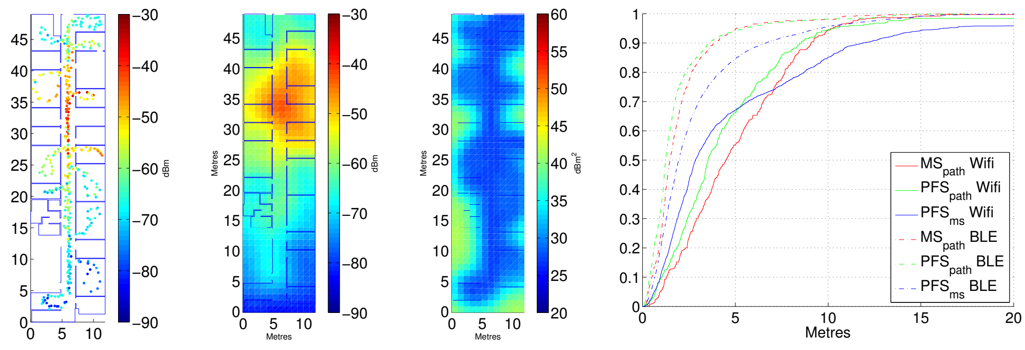


Figure 6.13: The estimated trajectory of *Path-3* in the table area. Four large round tables (not in the floor plan) have been added.



(a) PFSurvey output (b) GP map (μ) (c) GP var (σ^2) (d) Positioning CDF

Figure 6.14: Sample signal map and final positioning result

output using Gaussian Processes (Figures 6.14b and 6.14c) for illustration. Figure 6.14d gives the final positioning results for a separate test walk in the *Path-1* area. The results are gathered using the setup and methodology in Chapter 4 and show WiFi and BLE accuracy when using both the generated maps and a conventional manual survey (MS) as described in Chapter 4. It can be seen that the accuracy is marginally better using the generated maps. This is a consequence of the test path not straying far from the path survey input (Figure 6.14a). This is not an unreasonable occurrence since we would assume a dedicated surveyor to walk through all accessible areas, leaving few spaces where a user could be far from the survey trajectory.

6.12 Conclusions

In this chapter we have described and evaluated PFSurvey, a system designed to allow a dedicated surveyor to build a signal survey for a space in a matter of minutes using a commodity smartphone, assuming a floor plan of the space is available. The system uses a series of pre-processing steps to generate reliable loop closures between points on a noisy dead-reckoned trajectory. These are then fused with the floor plan to provide a robust, accurate trajectory that can be used to generate maps of any quantity measured during the survey.

We have evaluated PFSurvey in a large building and shown that it can successfully solve the room ambiguity problem that typically results from using solely the floor plan to constrain the dead-reckoning drift. We have demonstrated that the PFSurvey trajectory can be used to build detailed signal maps that allow positioning accuracy on a par with conventional, laborious manual surveying.

Chapter 7

Conclusions

7.1 Research contributions

This thesis has considered low-cost techniques to create signal maps for a fingerprinting-based indoor positioning system. The signal maps associate some property of the signal (usually signal strength) with locations. The fingerprinting-based positioning system matches the current signal observations (fingerprints) at a device to position it on this map. There are two kinds of method to build a signal map. The first one is the traditional *manual survey*, in which signal samples are collected in regular grid positions; the second is called *path survey* which collects signal samples continuously when a surveyor is walking along the (pre-defined or arbitrary) survey path. Path surveys are supposed to be more efficient than manual survey but have not been well-studied before this work.

The first research contribution of this thesis was to quantitatively evaluate path surveys with reference to a detailed manual survey:

- Smartphone-grade equipment was used to map both WiFi and Bluetooth Low Energy (BLE) signatures throughout our testbed.
- Upper-bound of real path survey performance has been provided by the use of an alternative high-accuracy location system to simulate perfect trajectory input.
- Gaussian Process (GP) regression maps were generated for both path survey and manual survey. A method to visualise the difference between these Gaussian maps was proposed.
- The impact of signal directionality on the positioning result has been investigated.
- Guidelines about how the path survey should be performed have been provided. It was demonstrated that by following our guidelines, a path survey can provide similar accuracy to a manual survey but with lower cost. Our survey guidelines are simple to follow, and many public buildings have security or building management personnel who regularly walk through them. These offer an ideal opportunity to preform regular path surveys.

The key to a path survey is accurate trajectory recovery. Based on these analytical results and the guidelines provided, an automated survey system was proposed:

- We advocated the use of a dedicated surveyor to hold the survey device in a convenient and stable manner, so that even simple PDR algorithm can successfully (but only roughly) recover the survey path.

- A SLAM system that corrects the noisy PDR results based on loop closures (i.e. revisits to the same locations) was proposed. This system consists of a front-end that matched sequences of magnetic measurements within a survey walk to identify loops in the trajectory, and a back-end that corrects the noisy PDR results based on the magnetic loop closures.
- High accuracy was achieved by the proposed system. For a 10 minute walk in our experiments, this system achieved an average error of any point 73 cm, and a subjective-objective error of 22 cm.
- The resultant signal maps have been found to give good positioning accuracies within 3–4 m of the trajectory (best performance was achieved within 1–2 m) and come close to the manual survey accuracies, despite requiring significantly less manual effort.
- The proposed system uses a commodity smart-phone as the only survey tool. No external infrastructures or floor plans are required. It supports free movement of the surveyor, and allows one-shot (push-to-fix) positioning.

This system relies on a good SLAM back-end. The back-end takes the noisy PDR trajectory and the loop closures as input, produces a corrected trajectory that is the most consistent with all the loop closures. The following contributions have been made to make the back-end more efficient and scalable:

- Three state-of-the-art GraphSLAM algorithms (g2o, Toro and SGD) have been evaluated against an extensive dataset. The evaluation has shown that SGD algorithm outperforms others.
- Because SGD is not optimised for the case of signal survey, a novel variant MSGD algorithm was proposed and demonstrated to achieve higher robustness and efficiency. For large-scale datasets, MSGD was found to be about 8 times faster than SGD (on the largest dataset being tested), and the improvements in speed are supposed to be more significant for even larger datasets.

So far, the proposed system can recover the survey trajectory based on the loop closure information. But the system has no awareness of whether the recovered trajectory satisfies the environmental constraints. When no environmental information like a floor plan is available, this is the best we can get. When we *do* have a digital floor plan, we have shown how we can take this one step further, i.e. recovering the survey trajectory so that it is consistent with the environment:

- The trajectory recovery problem was formulated as a tracking problem: given the rough initial position of the surveyor, we track incrementally the position of the surveyor during the whole survey process so as to recover the survey trajectory. The most commonly used algorithm that achieves this goal is particle filter and smoother.
- Experiments demonstrated that the state-of-the-art particle filter & smoother algorithms are not robust enough to recover the survey trajectory. The reason why they fail has been analysed and a more robust survey trajectory recovery system has been proposed. This system detects the loop closures in a more robust way, incorporates both the magnetic loop closures and the environmental information (i.e. the floor plan) in the filtering process, achieves high-accuracy (a mean error of 0.65 metre) with more robustness than the state-of-the-art alternatives.

7.2 Contributions to the research questions

This thesis has addressed the research questions listed in Section 1.1. They are examined and summarised in turn below.

7.2.1 Research question 1: how well can a path survey approach a manual survey?

To compare path survey and manual survey, quantitative and rigorous analysis of their various aspects (the map quality, map directionality, positioning performance, etc.) have been given.

In Chapter 3, RSS_{90} maps were proposed to visualise how the GP regression maps generated from path survey data approach GP maps from manual survey data. *To eliminate the trajectory recovery error, the path surveys were performed using an external high-accuracy positioning system to simulate best-case trajectory recovery*. The results have shown that the similarities between these two kinds of maps degrades faster in BLE maps than WiFi maps as the distance from the survey path increased. This can be explained by the trained parameters for GP regression: WiFi typically has a longer characteristic length-scale than BLE (4.96 m vs 2.50 m in the example shown in Section 3.4). This means that WiFi signals in distant locations correlate better than their BLE counterparts, so it is easier for GP regression to predict the signal strength distribution in more distant locations for WiFi than BLE. However, it should be noted that the generation of a GP map is affected by multiple factors and a rigorous interpretation is hard to provide. The proposed visualisation method of GP map difference just gives us an intuitive feeling about how the path survey map approaches the manual survey map for different signals.

To cross-validate these analysis results, the positioning performance of different GP maps has been evaluated. The directionality of the survey data was studied first. By inspection the RSS values taken in the corridor area where strong directionality is expected, a clear offset in RSS values has been shown when the measurements are taken in different directions. Both directional maps and omnidirectional maps were generated to examine whether this kind of offset affects the positioning results. It was shown that the path survey map differs from the manual survey map in this aspect. For a manual survey map, it was found that map directionality *does* affect the positioning performance, which causes 2.54 m and 1.56 m improvements (at the 90^{th} percentile) in positioning accuracy for WiFi and BLE respectively. But for path survey map, omnidirectional maps were found to outperform directional maps because too little training data is available for the generation of directional maps (especially for WiFi because modern smartphones have much lower WiFi sample frequency than that of BLE). Another important conclusion is that the positioning accuracy is inversely proportional to the distance between where the positioning request is issued and the nearest point in the survey path (this has been also confirmed by the positioning results in Chapter 3). The results have shown that best positioning performance for a path survey map can be achieved when this distance is within $1 \sim 2$ m. In this case, all positioning requests are in the high-confidence areas on the GP regressed map and it can approach the positioning performance of the manual survey map. This has cross-validated the analysis results given by RSS_{90} based GP map comparison.

It can be inferred from these results that segment survey, in which typically only a few segments along corridors are used, cannot provide good positioning performance as a PDR-based survey or a manual survey with better space coverage. And, in addition to the space coverage, positioning performance should be proportional to the signal sample density as it affects GP map quality. Both of these have been confirmed by quantitative evaluation results.

Summary *Based on our analysis, we can conclude that given sufficient density of measurements, a path survey produces accurate results within 1$\sim$2 m of the path if the trajectory is accurately estimated. It may in fact have more data* ***along the path*** *than a manual survey, hence giving a better result on it. This is good in general since we often travel along set paths. But it is of little use away from the path.*

7.2.2 Research question 2: how should the path survey be performed in order to provide best positioning performance?

The answer to the last question has shown how a path survey approaches a manual survey in different situations. Based on this, we have proposed several guidelines about how the path survey should be performed. By following our guidelines, the path survey can provide as good performance as manual survey but with minimal cost. We repeat the guidelines below:

1. The survey path should pass within $1\sim2$ m of any given point where positioning might be required. This is to provide good coverage so that all positioning requests happen in the high-confidence areas of the GP regressed map.

2. The surveyor should repeat some parts of the path to increase the signal sampling density. This is especially necessary for WiFi positioning because the WiFi sample rate on a modern smartphone is typically as low as $0.5\sim1.0$ Hz.

3. For the directionality of the wireless signals, although we have found that not considering this does not degrade the positioning performance significantly. But considering the possibility of more sophisticated orientation-based positioning algorithm being proposed, the surveyor can still try to pass a path in both directions (e.g. walking up and down a corridor) to record the directional information for potential use. And this can also increase the sample density of the signals.

   Summary *By following these simple guidelines, path survey can be a low-cost alternative for the laborious manual survey: a typical manual survey takes more than 3 hours in a 500 m^2 area, while a path survey typically takes less than 10 minutes.*

7.2.3 Research question 3: how can the surveyor's path be recovered accurately and efficiently?

To streamline the path survey, two automated signal survey systems were proposed. They dealt with the cases with and without floor plans, respectively. The system architectures for both systems are shown in Figure 4.3 and 6.6 respectively. Both of them make the path survey a simple and efficient process. What the surveyor needs to do is hold the survey device (e.g. a commodity smartphone) and walk around the environment freely, but following the guidelines from Section 7.2.2. After walking, the proposed survey systems can recover the survey trajectory accurately and robustly. Once the survey path is recovered, they are interpolated at the times signal measurements were made to produce a sequence of survey points. Signal maps can then be built and indoor positioning can be achieved. Compared with the manual survey, the proposed systems minimise the operations that need human involvements, which keeps the survey process automated and accurate.

The key of these systems is the ability to recover the survey path accurately and robustly. For both systems, the use of a dedicated surveyor was advocated to conduct the path survey. It is therefore reasonable to assert the phone is held in a convenient location consistently. Under these circumstances even simple PDR algorithms can recover the survey path successfully. However, the PDR drifts quickly so the raw PDR results are noisy and erroneous. The proposed systems take the raw PDR results as input and recover the survey path by correcting PDR errors. To achieve this, it is necessary to have sufficient information about how the PDR results should be corrected (or what is wrong with the PDR results in other words). This information comes in the form of *constraints*. We propose to use sequence-based magnetic loop closure constraints for the PDR trajectory correction. Robust algorithms have been proposed to detect and validate magnetic loop closures both with and without floor plans. When floor plans are not available, the trajectory correction problem can be formulated as a Graph optimisation (GraphSLAM) problem.

An efficient GraphSLAM algorithm (MSGD) was proposed and found to outperform state-of-the-art algorithms in terms of robustness and scalability. A mean subjective-objective error of 22 cm was achieved using our system, which significantly outperforms the state-of-the-art algorithms (1.27 m). However, without floor plan information, the best we can get is to recover the trajectory so that it is consistent with all the loop closure constraints, but no environmental constraints are considered. This means the recovered trajectory might have incorrect scale and be inconsistent with the environments.

When we *do* have a floor plan, two more kinds of constraints have been proposed to work with the magnetic loop closure constraints in recovering the trajectory: the straight line constraints and the floor plan constraints. In this case, the trajectory recovery problem is formulated as a filtering problem. We have shown that how the three kinds of constraints can be used by particle filters to recover the trajectory. We have demonstrated by experiments that the proposed system is more robust than the state-of-the-art particle filter & smoother algorithms. A mean error of 0.65 m was achieved by our system.

Summary *The path survey can be streamlined to an extremely simple process: the surveyor just needs to hold the survey device in a stable manner and walk around, then signal strength information can be automatically recorded by the device and the waking trajectory can be recovered using the proposed systems. High robustness and sub-metre accuracy can be achieved by the proposed systems.*

7.2.4 Research question 4: what inputs are needed for accurate trajectory recovery?

As described in the answer to the last question, the proposed automated survey systems use a commodity smart-phone as the survey tool, which is the *only* hardware we require. We have demonstrated that sub-metre accuracy can be achieved even when no floor plan information is available. What we relied on to recover the trajectory was just the sensor data available in a modern smart-phone, such as the acceleration, angular rate and magnetic fields data. When floor plans are available, we can make the trajectory recovery more robust: recover the trajectory so that it satisfies not only the loop closure constraints but also the environmental constraints.

Summary *We conclude that the minimum inputs for an automated survey system are the inertial sensor, the magnetic sensor and a sensor for whatever signal is to be used (these sensors could be from a modern smart-phone) and (optionally) a floor plan.*

7.3 Limitations and future work

The automated survey systems proposed in this thesis have been proven to provide robust and accurate performance. They can help to deploy indoor positioning service in large-scale and complicated environments efficiently. However, they still have some limitations when applied in real life. Here we briefly discuss some of these limitations and motivate potential future work.

7.3.1 Overcome environmental limitation

One of the basic ideas of the proposed systems is to robustly detect loop closures in the survey path, which is based on matching sequences of magnetic signals. However, due to the fast-varying nature of magnetic signals, it is hard to predict magnetic signal strength distribution over space. This means that the prerequisite for robust sequence-based matching is that the surveyor should exactly repeat the same path segments more than once and record the same magnetic sequence each time she traverses a segment. Therefore, different environments could affect the performance of the proposed systems. In office-like environments where clear walking paths are "pre-defined" by narrow corridors and/or furniture, the surveyor's walking trajectory is well constrained and it is more likely that some path segments are repeated exactly by the surveyor more than once. But if the signal survey is performed in the environments with large open spaces, the surveyor may not be able to exactly repeat enough path segments because there is no clear paths to follow. In this case, if the surveyor does not deliberately follow history walking trajectory, it is difficult to detect loop closures in the survey path by matching magnetic sequences. Despite this limitation, the cost of magnetic field-based SLAM (including sequence-based loop closure detection and trajectory optimisation) is much lower than more flexible systems like the vision-based SLAM systems [65, 19]. Vision-based systems are able to detect loop closures without the limitation of exact path segment repeating, e.g., the state-of-the-art *Bags of Binary Words* method which uses environmental features to detect loop closures [30]. So, it is interesting to integrate the low-cost magnetic field-based SLAM into more expensive but more flexible vision-based systems. The purpose is to develop a robust, low-cost and flexible SLAM system that can be used in various environments for signal survey.

7.3.2 Fuse multiple survey paths

This thesis only considers the case when there is only one surveyor surveying an environment in one go. But it is possible that the interested environment is very large so that a single surveyor may not be able to cover the whole area in a single attempt. In this case, either the surveyor needs to perform several survey walks, or multiple surveyors should be used to simultaneously survey this environment (with different surveyors surveying different areas). Both of these would produce multiple survey paths that need to be fused to generate a complete signal map. Then how to fuse multiple survey paths is an interesting research area and several potential challenges need to be solved. When the floorplan is unavailable, a basic requirement of successful trajectory fusion is that there are sufficient parts of these paths overlapping with each other. Then a problem needs to be solved is how to efficiently detect loop closures both intra-trajectory and inter-trajectory. We have shown that how intra-trajectory loop closures can be detected efficiently and robustly. But inter-trajectory loop closure detection is more challenge. In this case, there is no spatial constraints on the searching range of potential loop closures. Exhaustive searching between any segments on each path is very inefficient. Also, this causes that false positive loop closures are more difficult to eliminate. In the case when a floorplan is available. We can first fuse long paths by locating them to the floor plan using particle filters. For short paths that could not be well located to the floor plan by particle filters, we can fuse them with those long paths by loop closures as in the no-floorplan case. Then similar problems about intra-trajectory and inter-trajectory loop closure detection need to be solved.

7.3.3 Achieve 3D signal survey

The proposed systems are assumed to work in 2D environment. The trajectory recovery techniques they adopted cannot deal with 3D movements yet. This is a limitation of the proposed systems. For example, if the environment we are surveying is not a flat area, the survey path cannot be recovered properly. Or, if we'd like to deploy an indoor positioning system in a multiple-storey building, we need to conduct signal survey floor-by-floor, and the signal information in the staircase areas cannot be easily mapped. Therefore, 3D mapping capability is useful in real applications. To deal with 3D movement, the system should be able to detect the surveyor's changes in height. A foot mounted IMU is capable of 3D tracking but requires extra hardware. Modern smart-phones are usually equipped with barometers that can track the altitude of the device. But they are too noisy to achieve accurate height tracking. Another possibility is to adopt the visual odometry techniques which use cameras to infer 3D motion of the device [80]. However, vision system typically require high computational costs to deal with large amount of image data. This is not a problem if the image processing is conducted offline on a desktop PC, but might be a challenge if the images are processed on the smartphone (e.g., to give real-time feedback to the surveyor). Also, combining the smart-phone camera with its IMU sensors to achieve efficient and accurate 3D tracking is an interesting research area.

7.3.4 Automatic update of signal maps

Real environments could be dynamic. For examples, WiFi access points could be turned off or moved, new beacons can be deployed, furniture might be rearranged. These factors can affect the signal strength distribution in the indoor environment. For an indoor positioning system deployed for long-term usage, frequent re-survey is needed to deal with dynamic environments. Re-surveying using the proposed systems is efficient, but still requires a dedicated surveyor to perform the survey every time. To minimise the cost, it is interesting to explore the possibility of automatic update of signal maps in dynamic environment. When an initial signal survey is done using the proposed system, we've build a signal map for each signal source (e.g. every WiFi and BLE beacon). If only a small portion of them are changed, fingerprinting might still be able to position users to a certain accuracy. Then the changes in the affected signal sources could be detected by comparing the latest signal information (passed to the positioning system from the users) with corresponding signal maps, and updates can be made to correct outdated signal maps. To achieve this, lots of research problems need to be solved. For example, how to fuse the positioning uncertainty information and signal measurement noise to achieve robust signal map updating. A potential method is to select well-localised user path (e.g., a path which is long enough and could be localised to the floor plan accurately by simply a PDR algorithm and a particle filter) and create mini-signal maps by Gaussian process regression using only the data collected during this path. Then, for a signal source, we have two Gaussian distributions for the signal strength distribution at each spatial point: one from the initial map and the other from the latest path. By checking the statistical compatibility of these two distribution (e.g., by χ^2 test), we can choose to fuse these two hypotheses or simply discard the old one.

7.3.5 Preserve user privacy

This thesis focuses on the low-cost deployment method of indoor positioning system. Hopefully, in the future such systems will be deployed in more and larger indoor areas and people's location can be tracked really accurately indoors. On the bright side, more location-based services could help improve people's life. However, the problem about how to preserve user privacy could arise. Therefore, an important future research area is how to design the indoor positioning system so that it could not only track people accurately but also preserve user privacy. A possible system architecture for privacy-preservation purpose is that the users download the signal maps on their smartphones and compute their locations locally. Data transmission from user to the server should be strictly restricted in order to protect user privacy as much as possible. However, if no user data is uploaded to the server, it is difficult to achieve automatic map updates like mentioned before. If data transmission from user to server is necessary, proper encryption methods should be used. We leave this for future work.

7.4 Final words

This thesis assessed the path survey quantitatively and rigorously, demonstrated that path survey can approach the manual survey in terms of accuracy if certain guidelines are followed. Automated survey systems have been proposed and a commodity smart-phone is the only survey device required. The proposed systems achieve sub-metre accuracy in recovering the survey trajectory both with and without environmental information (floor plans), and have been found to outperform the state-of-the-art in terms of robustness and scalability. For the indoor positioning systems which are supposed to be deployed in large-scale and complicated environments, the proposed systems can greatly ease the signal survey process. For dynamic environment where frequent re-survey is needed, our proposed systems are especially valuable.

Appendix A

Datasets and evaluation results for SLAM

All the datasets that used to evaluate different SLAM back-end algorithms are shown in Figure A.1, A.2, A.3 and A.4. Figure A.1 shows 4 datasets collected in WGB2a testbed. Figure A.2 shows 5 datasets collected in WGB1 testbed. Figure A.3 shows 4 datasets collected in WGB2 testbed. Figure A.4 shows 2 datasets collected in ENG testbed, 2 datasets collected in RUTH testbed and 2 datasets collected in KX testbed. Details about the testbeds, groundtruth and data collecting were given in Section 4.6

The evaluation results of Toro, g2o and SGD on each dataset are shown in Figure A.5, A.6, A.7 and A.8.

The evaluation results of MSGD on each dataset are shown in Figure A.9, A.10, A.11 and A.12.

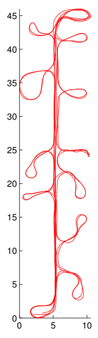
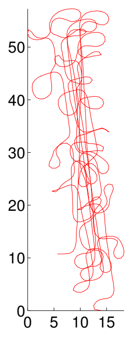
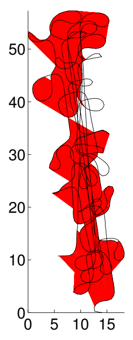
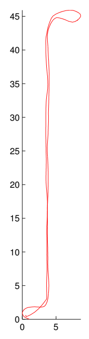
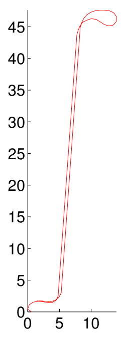
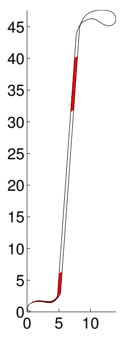
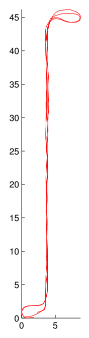
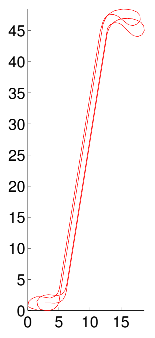
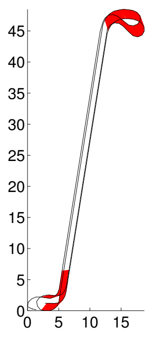
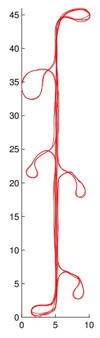
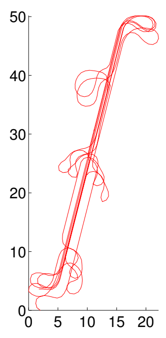
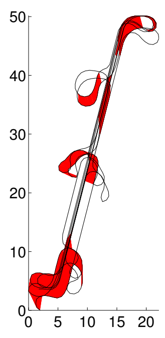
	Groundtruth	Filtered path	Filtered path with loop closures
WGB2a-1	 (a)	 (b)	 (c)
WGB2a-2	 (d)	 (e)	 (f)
WGB2a-3	 (g)	 (h)	 (i)
WGB2a-4	 (j)	 (k)	 (l)

Figure A.1: Dataset of WGB2a.

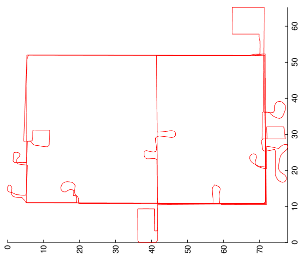
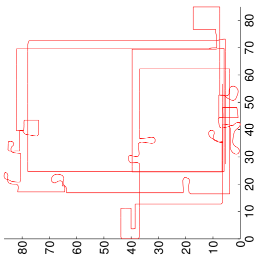
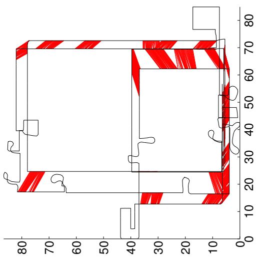
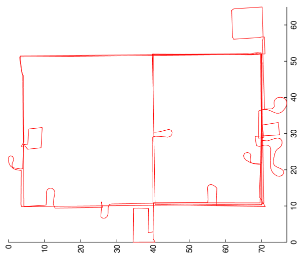
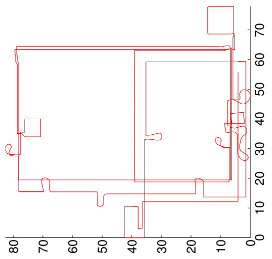
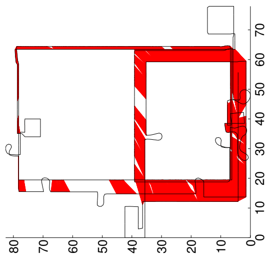
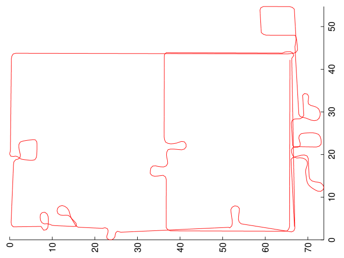
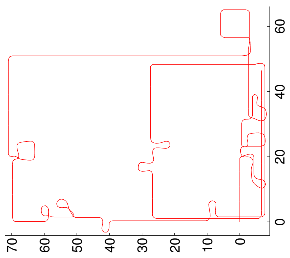
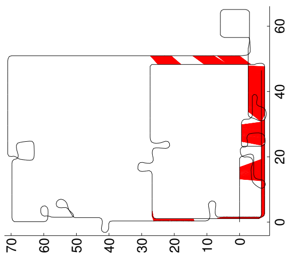
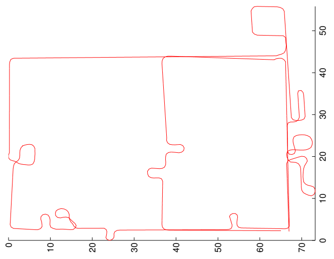
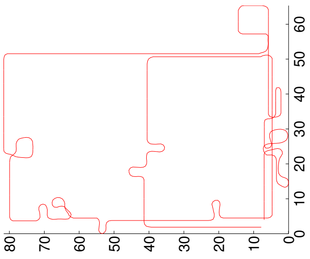
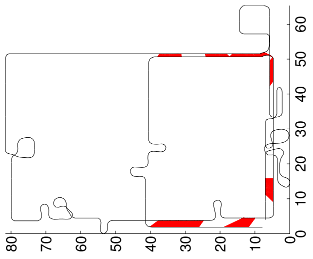
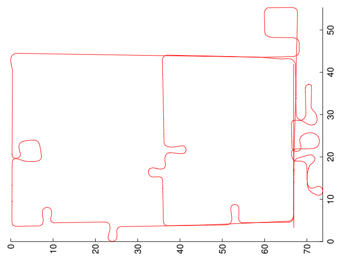
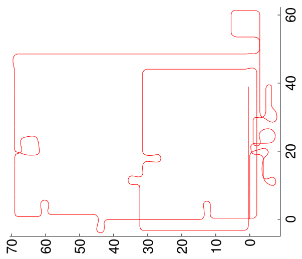
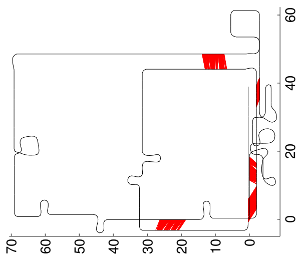
	Groundtruth	Filtered path	Filtered path with loop closures
WGB1-1	 (a)	 (b)	 (c)
WGB1-2	 (d)	 (e)	 (f)
WGB1-3	 (g)	 (h)	 (i)
WGB1-4	 (j)	 (k)	 (l)
WGB1-5	 (m)	 (n)	 (o)

Figure A.2: Dataset of WGB1.

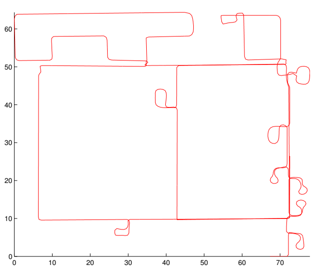
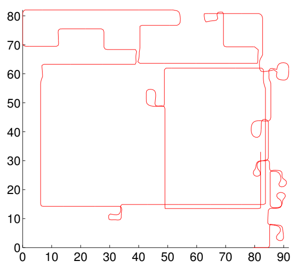
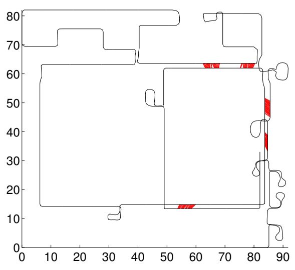
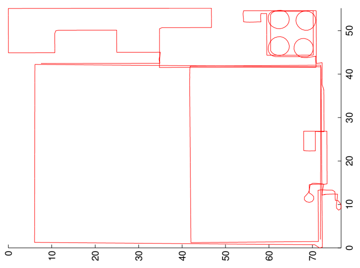
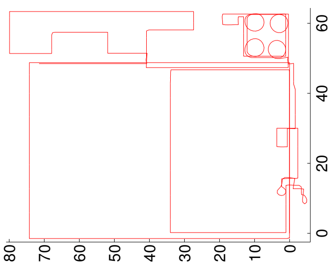
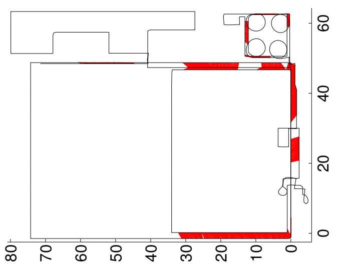
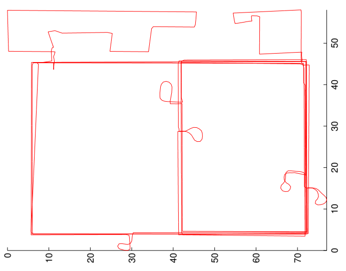
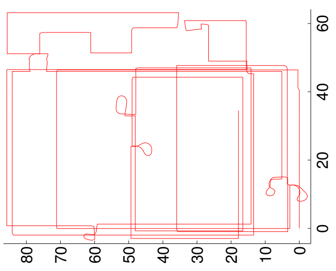
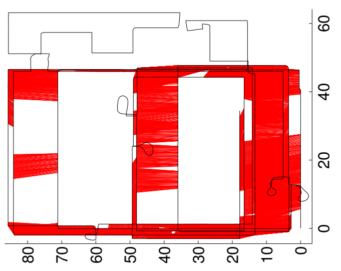
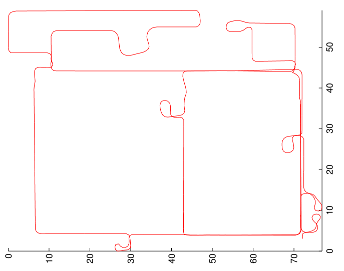
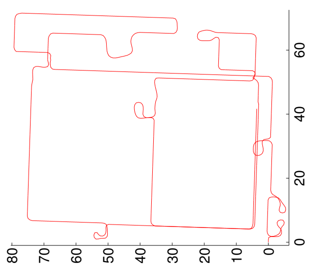
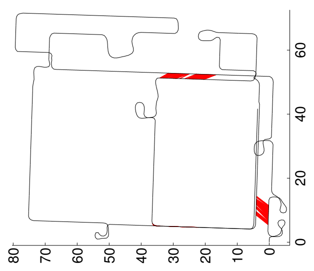
	Groundtruth	Filtered path	Filtered path with loop closures
WGB2-1	 (a)	 (b)	 (c)
WGB2-2	 (d)	 (e)	 (f)
WGB2-3	 (g)	 (h)	 (i)
WGB2-4	 (j)	 (k)	 (l)

Figure A.3: Dataset of WGB2.

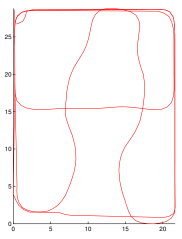
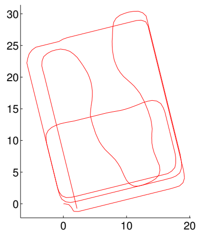
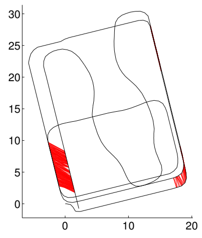
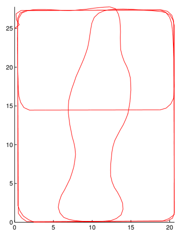
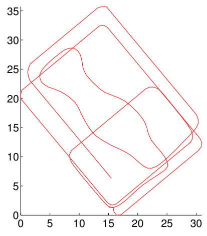
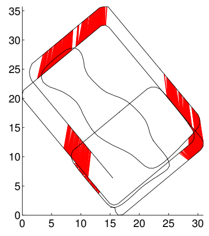
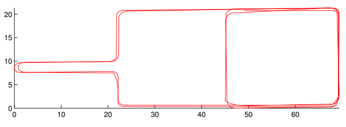
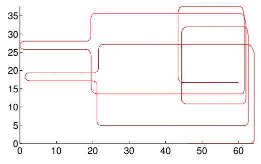
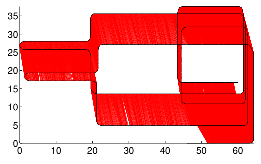
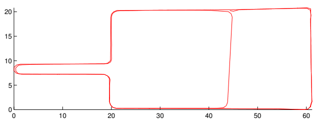
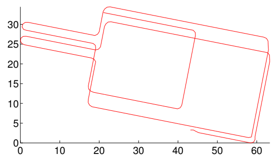
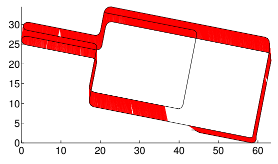
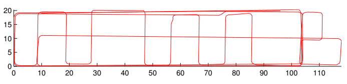
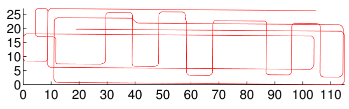
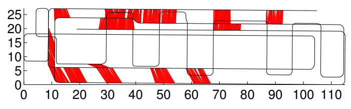
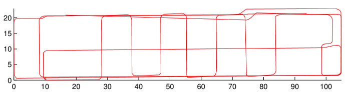
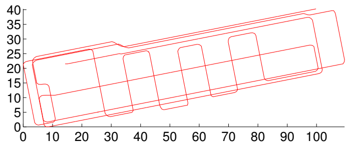
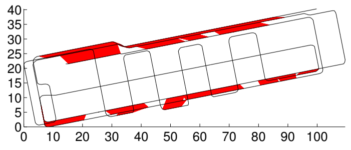
	Groundtruth	Filtered path	Filtered path with loop closures
ENG-1	 (a)	 (b)	 (c)
ENG-2	 (d)	 (e)	 (f)
RUTH-1	 (g)	 (h)	 (i)
RUTH-2	 (j)	 (k)	 (l)
KX-1	 (m)	 (n)	 (o)
KX-2	 (p)	 (q)	 (r)

Figure A.4: Dataset of ENG, RUTH and KX.

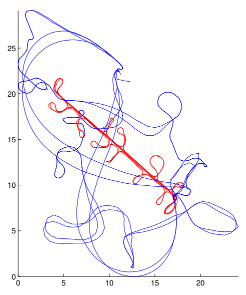
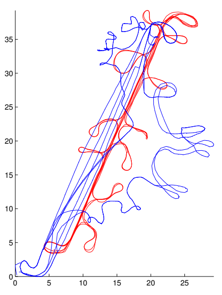
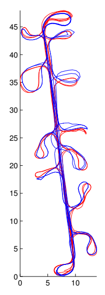
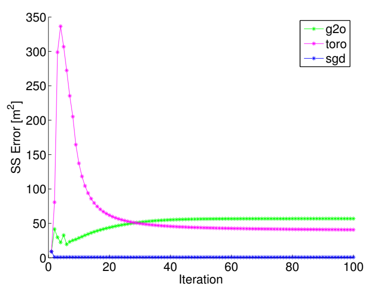
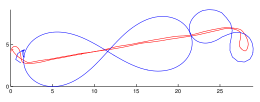
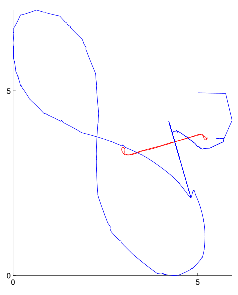
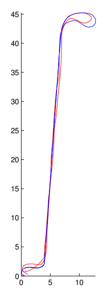
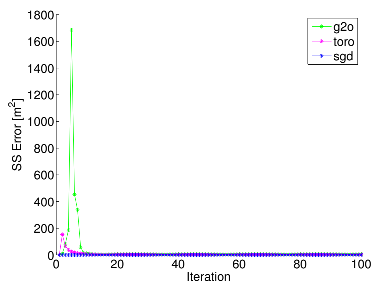
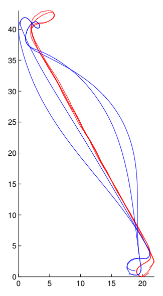
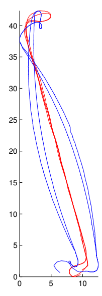
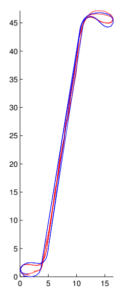
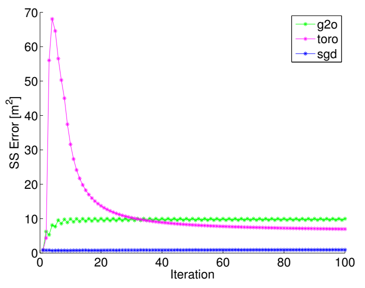
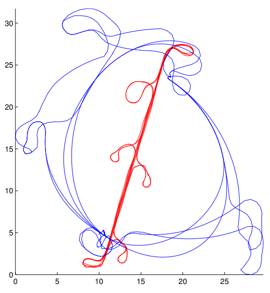
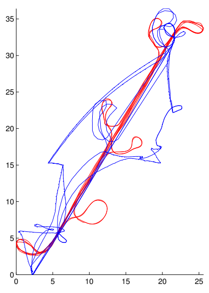
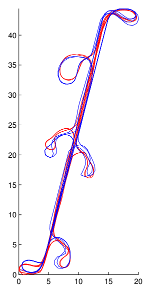
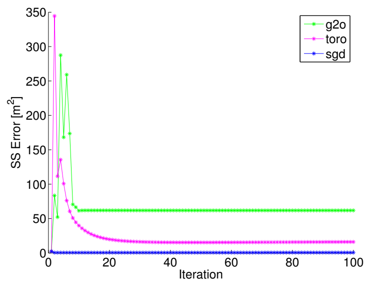
	g2o	Toro	SGD	State Squared Error
WGB2a-1	 (a)	 (b)	 (c)	 (d)
WGB2a-2	 (e)	 (f)	 (g)	 (h)
WGB2a-3	 (i)	 (j)	 (k)	 (l)
WGB2a-4	 (m)	 (n)	 (o)	 (p)

Figure A.5: SLAM results on WGB2a. The trajectories (the red is the transformed groundtruth and the blue the slam result) and the SS errors are the results of the 100th iteration of different algorithms.

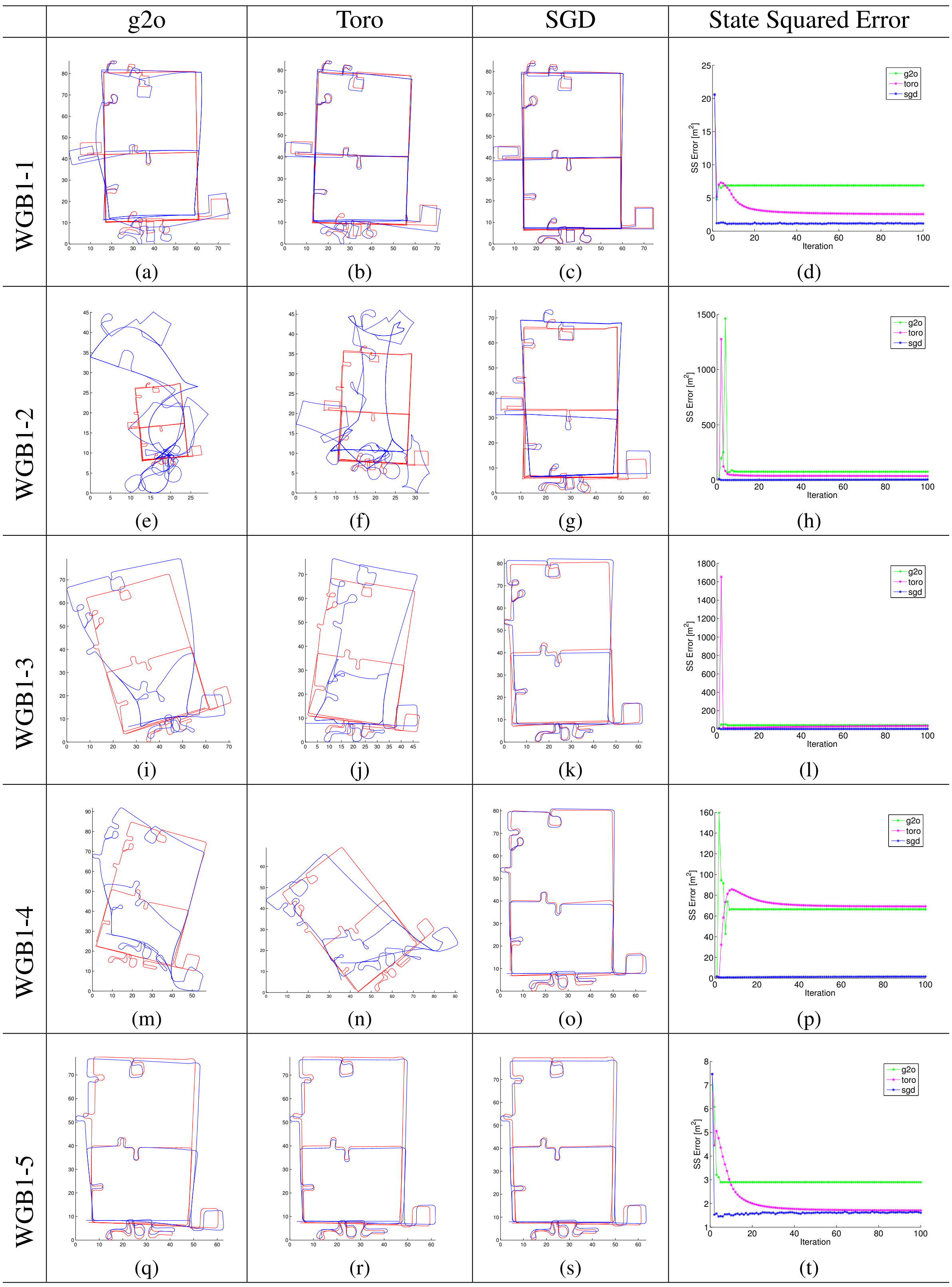


Figure A.6: SLAM results on WGB1. The trajectories (the red is the transformed groundtruth and the blue the slam result) and the SS errors are the results of the 100th iteration of different algorithms.

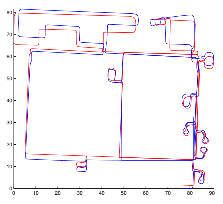
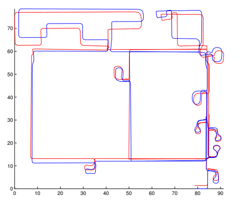
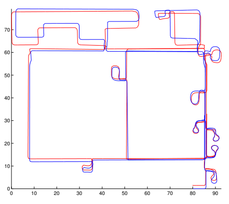
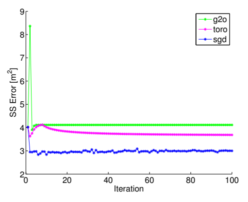
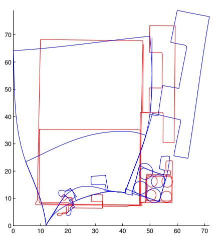
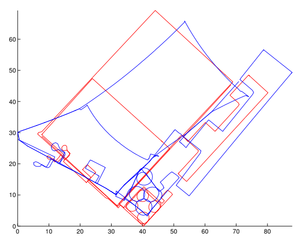
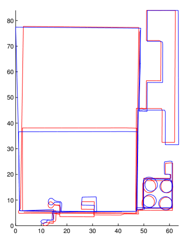
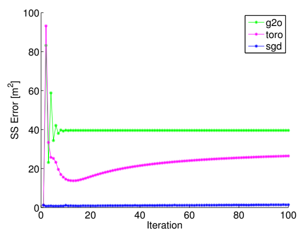
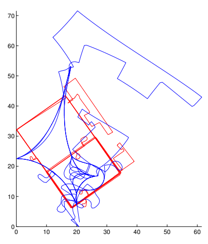
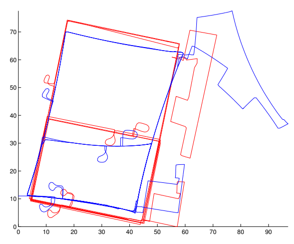
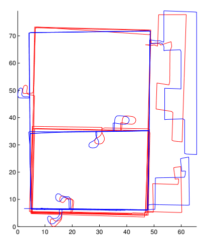
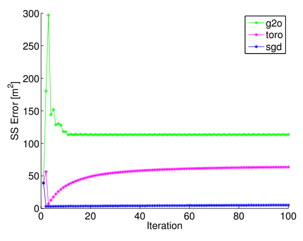
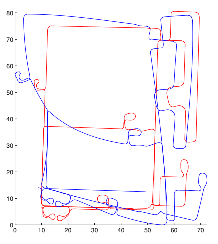
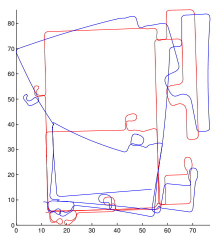
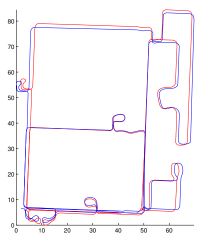
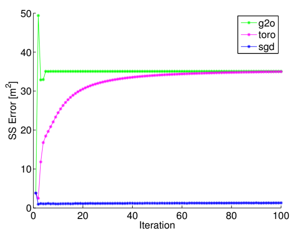
	g2o	Toro	SGD	State Squared Error
WGB2-1	 (a)	 (b)	 (c)	 (d)
WGB2-2	 (e)	 (f)	 (g)	 (h)
WGB2-3	 (i)	 (j)	 (k)	 (l)
WGB2-4	 (m)	 (n)	 (o)	 (p)

Figure A.7: SLAM results on WGB2. The trajectories (the red is the transformed groundtruth and the blue the slam result) and the SS errors are the results of the 100th iteration of different algorithms.

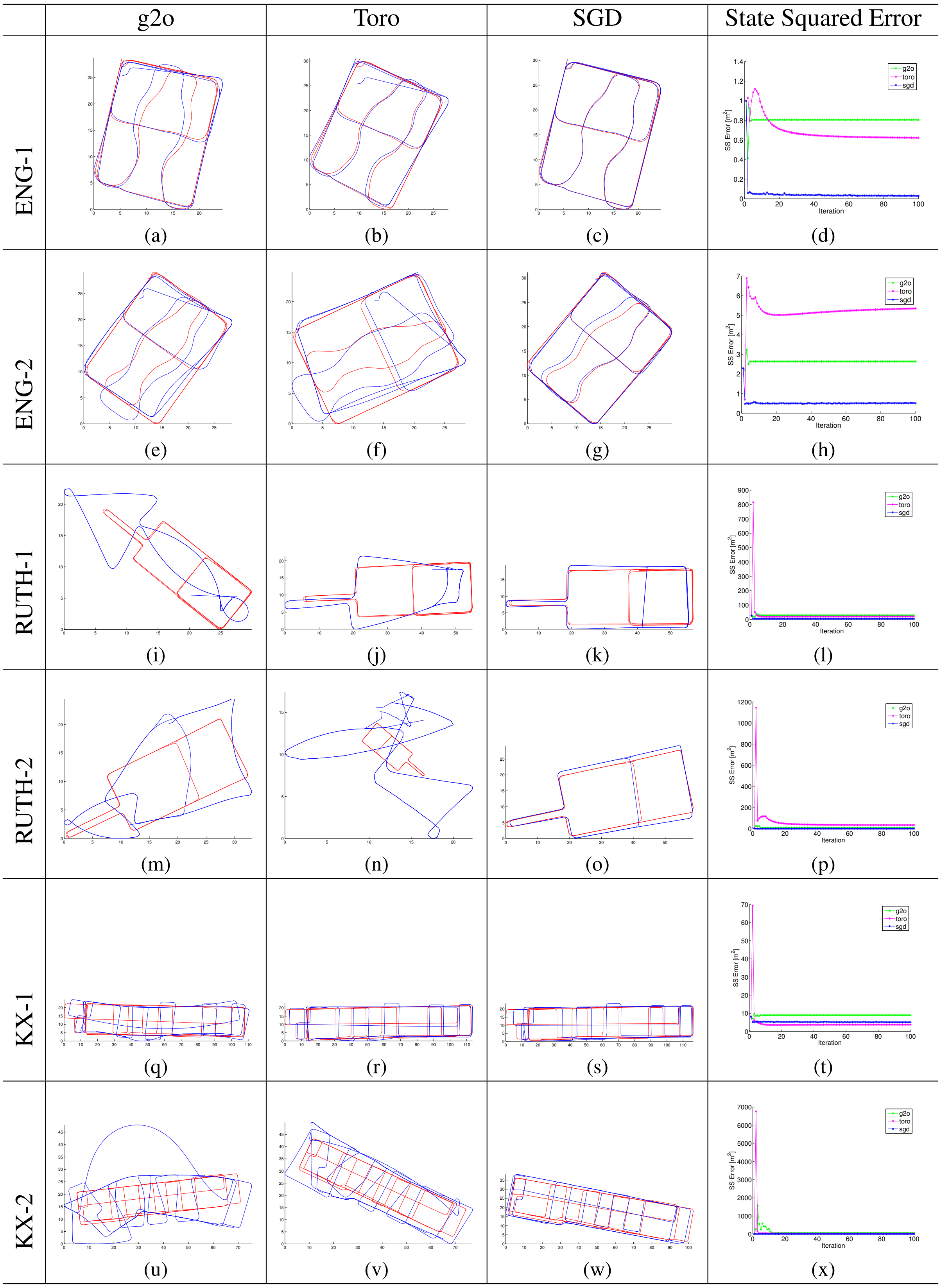


Figure A.8: SLAM results on ENG, RUTH and KX. The trajectories (the red is the transformed groundtruth and the blue the slam result) and the SS errors are the results of the 100th iteration of different algorithms.

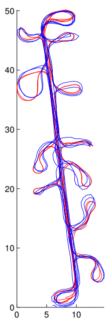
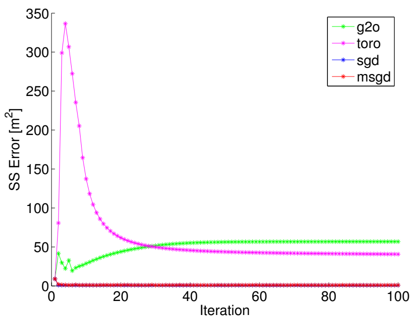
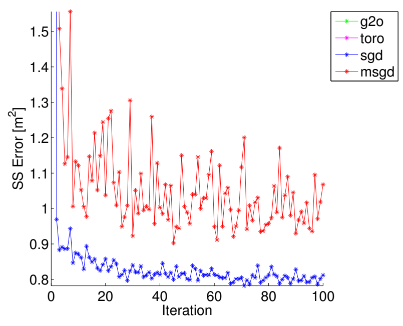
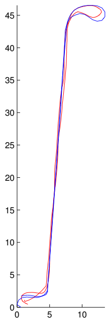
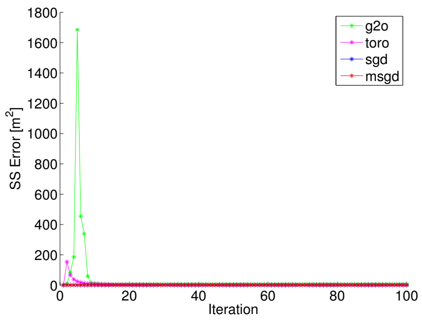
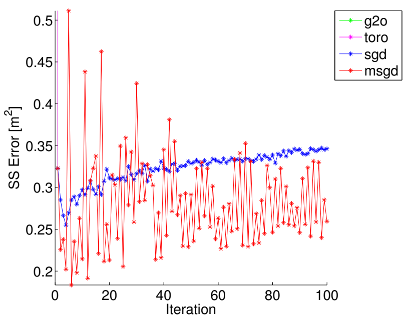
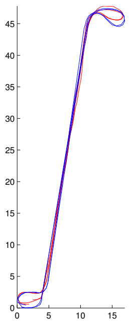
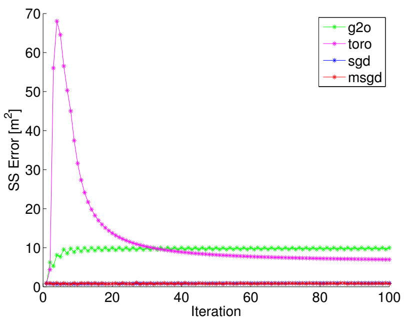
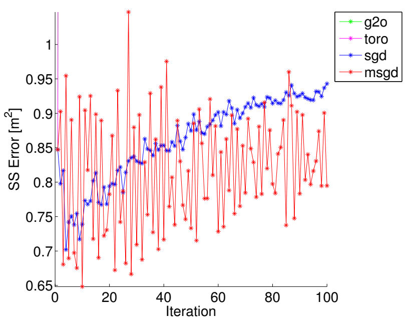
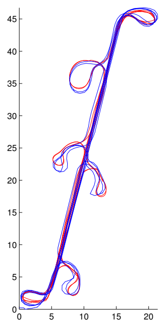
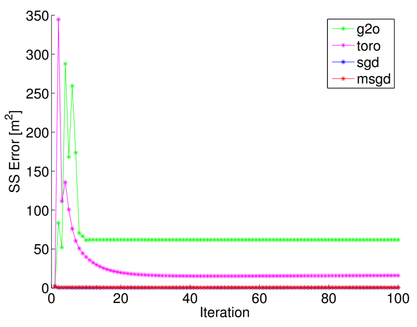
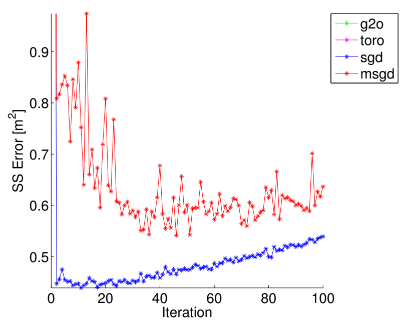
	MSGD Result	State Squared Error	State Squared Error (Zoomed In)
WGB2a-1	 (a)	 (b)	 (c)
WGB2a-2	 (d)	 (e)	 (f)
WGB2a-3	 (g)	 (h)	 (i)
WGB2a-4	 (j)	 (k)	 (l)

Figure A.9: MSGD results on WGB2a. The trajectories (the red is the transformed groundtruth and the blue the slam result) and the SS errors are the results of the 100th iteration.

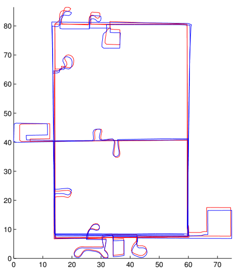
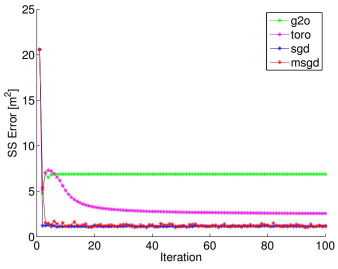
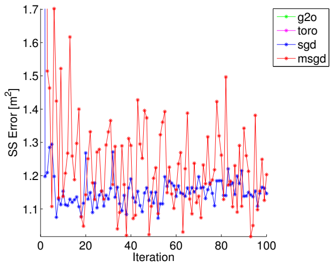
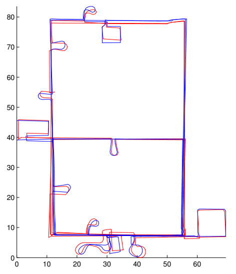
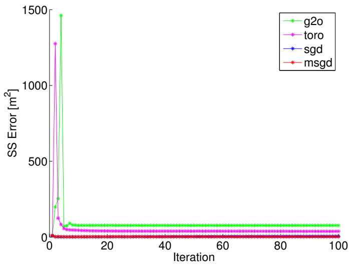
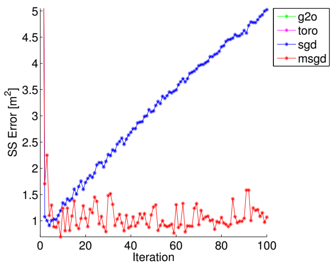
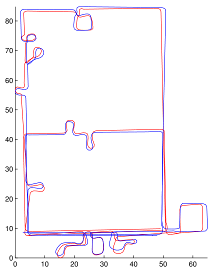
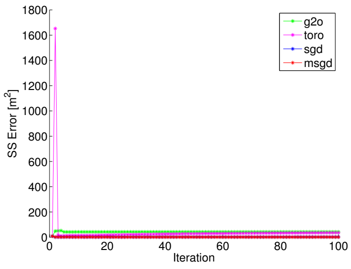
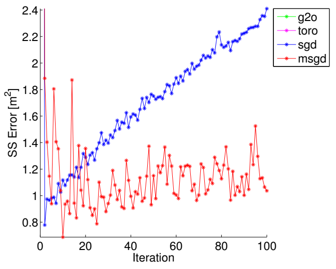
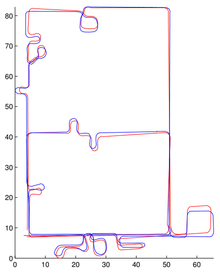
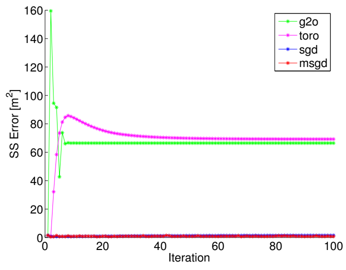
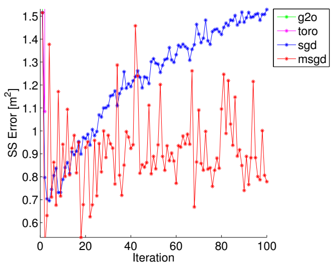
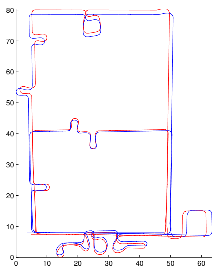
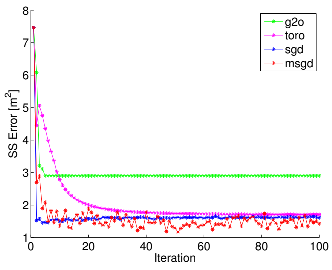
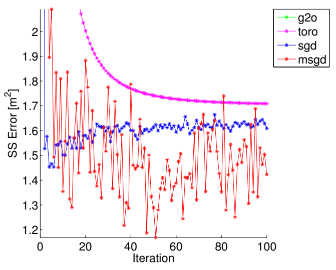
	MSGD Result	State Squared Error	State Squared Error (Zoomed In)
WGB1-1	 (a)	 (b)	 (c)
WGB1-2	 (d)	 (e)	 (f)
WGB1-3	 (g)	 (h)	 (i)
WGB1-4	 (j)	 (k)	 (l)
WGB1-5	 (m)	 (n)	 (o)

Figure A.10: MSGD results on WGB1. The trajectories (the red is the transformed groundtruth and the blue the slam result) and the SS errors are the results of the 100th iteration.

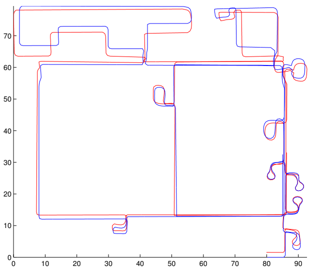
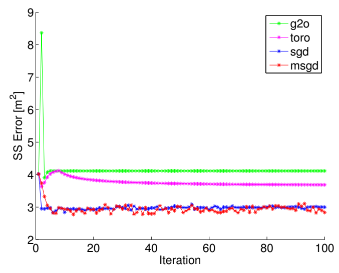
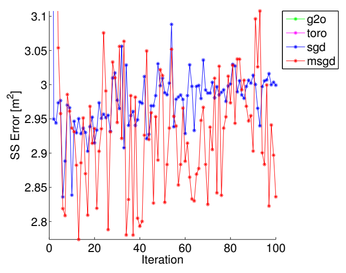
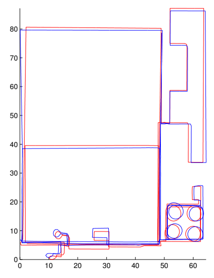
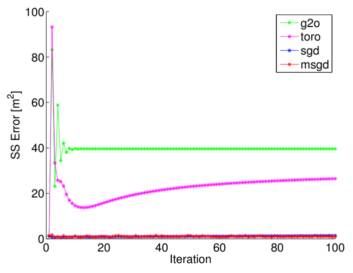
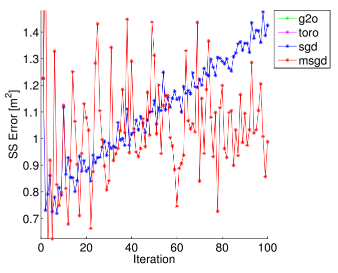
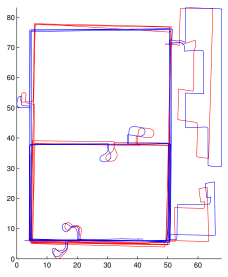
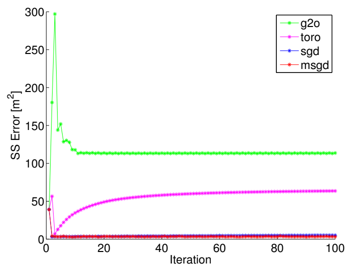
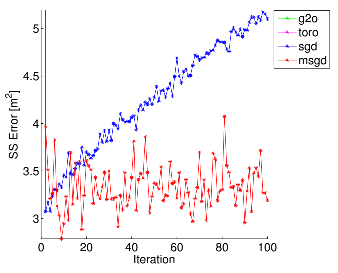
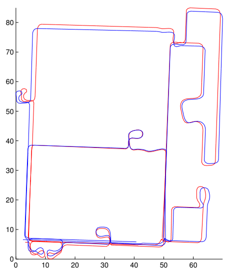
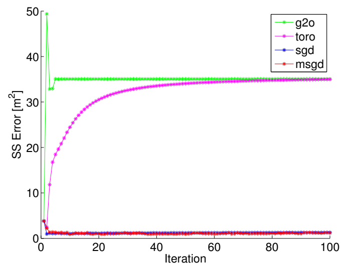
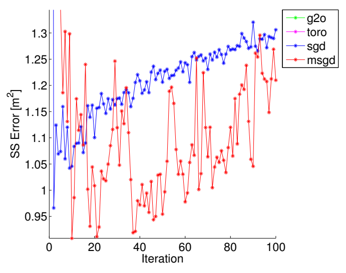
	MSGD Result	State Squared Error	State Squared Error (Zoomed In)
WGB2-1	 (a)	 (b)	 (c)
WGB2-2	 (d)	 (e)	 (f)
WGB2-3	 (g)	 (h)	 (i)
WGB2-4	 (j)	 (k)	 (l)

Figure A.11: MSGD results on WGB2. The trajectories (the red is the transformed groundtruth and the blue the slam result) and the SS errors are the results of the 100th iteration.

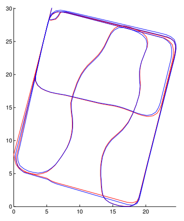
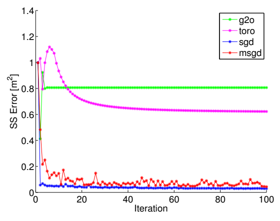
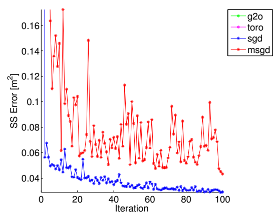
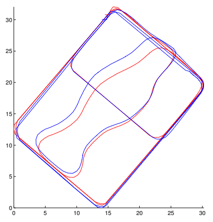
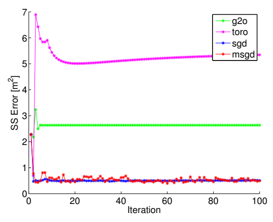
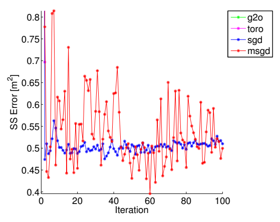
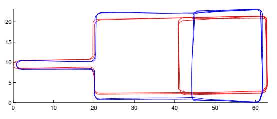
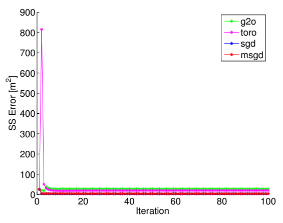
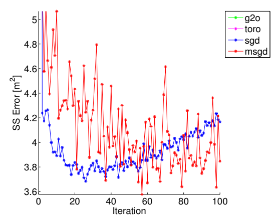
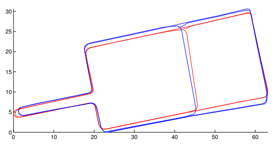
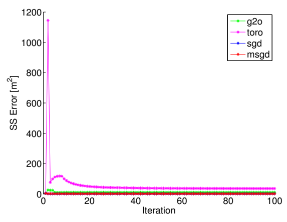
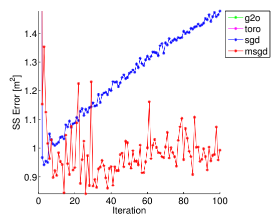
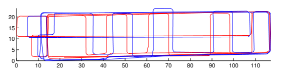
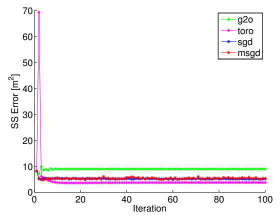
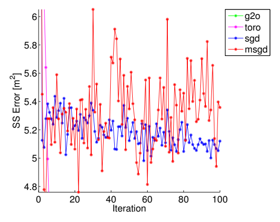
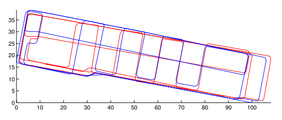
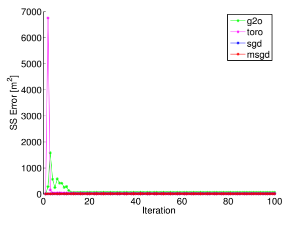
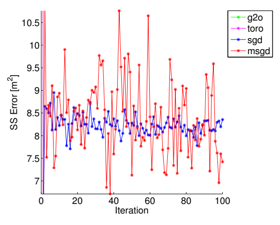
	MSGD Result	State Squared Error	State Squared Error (Zoomed In)
ENG-1	 (a)	 (b)	 (c)
ENG-2	 (d)	 (e)	 (f)
RUTH-1	 (g)	 (h)	 (i)
RUTH-2	 (j)	 (k)	 (l)
KX-1	 (m)	 (n)	 (o)
KX-2	 (p)	 (q)	 (r)

Figure A.12: MSGD results on ENG, RUTH and KX. The trajectories (the red is the transformed groundtruth and the blue the slam result) and the SS errors are the results of the 100th iteration.

Bibliography

[1] M. Addlesee, R. Curwen, S. Hodges, J. Newman, P. Steggles, A. Ward, and A. Hopper. Implementing a sentient computing system. *IEEE Computer*, 34(8), August 2001.

[2] Klaithem Al Nuaimi and Hesham Kamel. A survey of indoor positioning systems and algorithms. In *Innovations in Information Technology (IIT), 2011 International Conference on*, pages 185–190. IEEE, 2011.

[3] Diego Alvarez, Rafael C González, Antonio López, and Juan C Alvarez. Comparison of step length estimators from weareable accelerometer devices. In *Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE*, pages 5964–5967. IEEE, 2006.

[4] Anshui Anshul Rai, Krishna Kant Chintalapudi, Padmanabhan Venkat, and Rijurekha Sen. Zee : Zero-effort crowdsourcing for indoor localization. In *Proceedings of The 18th Annual International Conference on Mobile Computing and Networking (MobiCom)*, august 2012.

[5] Paramvir Bahl, Venkata N. Padmanabhan, and Anand Balachandran. Enhancements to the radar user location and tracking system. Technical report, 2000.

[6] Hari Balakrishnan and Nissanka Bodhi Priyantha. The cricket indoor location system: Experience and status. In *2003 Workshop on Location-Aware Computing*, page 7, 2003.

[7] Michael Bowling, Dana Wilkinson, Ali Ghodsi, and Adam Milstein. Subjective localization with action respecting embedding. In *Robotics Research*, pages 190–202. Springer, 2007.

[8] Agata Brajdic and Robert Harle. Walk detection and step counting on unconstrained smartphones. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 225–234. ACM, 2013.

[9] Nicholas Carlevaris-Bianco and Ryan M Eustice. Generic factor-based node marginalization and edge sparsification for pose-graph slam. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 5748–5755. IEEE, 2013.

[10] Nicholas Carlevaris-Bianco and Ryan M Eustice. Conservative edge sparsification for graph slam node removal. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 854–860. IEEE, 2014.

[11] Luca Carlone, Andrea Censi, and Frank Dellaert. Selecting good measurements via 1 relaxation: A convex approach for robust estimation over graphs. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2667–2674. IEEE, 2014.

[12] Jaewoo Chung, Matt Donahoe, Chris Schmandt, Ig-Jae Kim, Pedram Razavai, and Micaela Wiseman. Indoor location sensing using geo-magnetism. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, MobiSys '11, pages 141–154, New York, NY, USA, 2011. ACM.

[13] Randal Douc, Aurélien Garivier, Eric Moulines, Jimmy Olsson, et al. Sequential monte carlo smoothing for general state space hidden markov models. *The Annals of Applied Probability*, 21(6):2109–2145, 2011.

[14] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and computing*, 10(3):197–208, 2000.

[15] Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 12(656-704):3, 2009.

[16] Tom Duckett, Stephen Marsland, and Jonathan Shapiro. Fast, on-line learning of globally consistent maps. *Autonomous Robots*, 12(3):287–300, 2002.

[17] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *Robotics &amp; Automation Magazine, IEEE*, 13(2):99–110, 2006.

[18] Austin Eliazar and Ronald Parr. Dp-slam: Fast, robust simultaneous localization and mapping without predetermined landmarks. In *in Proc. 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-03*, pages 1135–1142. Morgan Kaufmann, 2003.

[19] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

[20] Ryan M Eustice, Hanumant Singh, and John J Leonard. Exactly sparse delayed-state filters. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2417–2424. IEEE, 2005.

[21] Ramsey Faragher and Rob Harle. Smartslam - an efficient smartphone indoor positioning system exploiting machine learning and opportunistic sensing. In *Proceedings of the 26th International Technical Meeting of the Satellite Division of the Institute of Navigation, ION GNSS+ 2013, Nashville, Tennesse*, September 2013.

[22] Ramsey Faragher and Robert Harle. Location fingerprinting with bluetooth low energy beacons. *Selected Areas in Communications, IEEE Journal on*, 33(11):2418–2428, 2015.

[23] R.M. Faragher, C. Sarno, and M. Newman. Opportunistic radio slam for indoor navigation using smartphone sensors. In *Position Location and Navigation Symposium (PLANS), 2012 IEEE/ION*, pages 120 –128, April 2012.

[24] Brian Ferris, Dieter Fox, and Neil Lawrence. Wifi-slam using gaussian process latent variable models. In *Proceedings of the 20th international joint conference on Artifical intelligence*, IJCAI'07, pages 2480–2485, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[25] Brian Ferris, Dirk Haehnel, and Dieter Fox. Gaussian processes for signal strength-based location estimation. In *In proc. of robotics science and systems*. Citeseer, 2006.

[26] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[27] Dieter Fox. Kld-sampling: Adaptive particle filters. In *Advances in neural information processing systems*, pages 713–720, 2001.

[28] E. Foxlin. Pedestrian Tracking with Shoe-Mounted Inertial Sensors. *IEEE Computer Graphics and Applications*, 25(6):38–46, November 2005.

[29] Udo Frese, Per Larsson, and Tom Duckett. A multilevel relaxation algorithm for simultaneous localization and mapping. *Robotics, IEEE Transactions on*, 21(2):196–207, 2005.

[30] Dorian Gálvez-López and Juan D Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012.

[31] Sinan Gezici and H Vincent Poor. Position estimation via ultra-wide-band signals. *Proceedings of the IEEE*, 97(2):386–403, 2009.

[32] Toni Giorgino. Computing and visualizing dynamic time warping alignments in r: the dtw package. *Journal of statistical Software*, 31(7):1–24, 2009.

[33] Simon J. Godsill, Simon J. Godsill, Arnaud Doucet, and Mike" West. Monte carlo smoothing for non-linear time series. *JOURNAL OF THE AMERICAN STATISTICAL ASSOCIATION*, 99:156–168, 2004.

[34] Giorgio Grisetti, Cyrill Stachniss, Slawomir Grzonka, and Wolfram Burgard. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *Robotics: Science and Systems*, 2007.

[35] Dongsoo Han, Sukhoon Jung, Minkyu Lee, and Giwan Yoon. Building a practical wi-fi-based indoor navigation system. *IEEE Pervasive Computing*, 13(2):72–79, 2014.

[36] R. Harle. A survey of indoor inertial positioning systems for pedestrians. *Communications Surveys Tutorials, IEEE*, 15(3):1281–1293, Third 2013.

[37] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster. The anatomy of a context-aware application. *Wireless Networks*, 8(2):187–197, 2002.

[38] V. Honkavirta, T. Perala, S. Ali-Loytty, and R. Piche. A comparative survey of wlan location fingerprinting methods. In *Positioning, Navigation and Communication, 2009. WPNC 2009. 6th Workshop on*, pages 243–251, March 2009.

[39] Berthold KP Horn. Closed-form solution of absolute orientation using unit quaternions. *JOSA A*, 4(4):629–642, 1987.

[40] AKM Mahtab Hossain and Wee-Seng Soh. A survey of calibration-free indoor positioning systems. *Computer Communications*, 2015.

[41] Guoquan Huang, Michael Kaess, and John J Leonard. Consistent sparsification for graph optimization. In *Mobile Robots (ECMR), 2013 European Conference on*, pages 150–157. IEEE, 2013.

[42] J. Huang, D. Millman, M. Quigley, D. Stavens, S. Thrun, and A. Aggarwal. Efficient, generalized indoor wifi graphslam. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1038–1043, May.

[43] iBeacon. https://support.apple.com/en-gb/ht202880 (accessed july 2016).

[44] IndoorAtlas. http://www.indooratlas.com (accessed september 2014).

[45] Antonio Ramón Jiménez, F Seco, José Carlos Prieto, and Jorge Guevara. Indoor pedestrian navigation using an ins/ekf framework for yaw drift reduction and a foot-mounted imu. In *Positioning Navigation and Communication (WPNC), 2010 7th Workshop on*, pages 135–143. IEEE, 2010.

[46] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. isam: Incremental smoothing and mapping. *Robotics, IEEE Transactions on*, 24(6):1365–1378, 2008.

[47] Thomas King, Stephan Kopf, Thomas Haenselmann, Christian Lubberger, and Wolfgang Effelsberg. Compass: A probabilistic indoor positioning system based on 802.11 and digital compasses. In *Proceedings of the 1st international workshop on Wireless network testbeds, experimental evaluation & characterization*, WiNTECH '06, pages 34–40, New York, NY, USA, 2006. ACM.

[48] Mikkel Baun Kjærgaard. Indoor location fingerprinting with heterogeneous clients. *Pervasive and Mobile Computing*, 7(1):31–43, 2011.

[49] Martin Klepal and Stephane Beauregard. A Backtracking Particle Filter for fusing building plans with PDR displacement estimates. *2008 5th Workshop on Positioning Navigation and Communication*, 2008:207–212, 2008.

[50] Kurt Konolige, Giorgio Grisetti, Rainer Kümmerle, Wolfram Burgard, Benson Limketkai, and Regis Vincent. Efficient sparse pose adjustment for 2d mapping. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 22–29. IEEE, 2010.

[51] Hakan Koyuncu and Shuang Hua Yang. A survey of indoor positioning and object locating systems. *IJCSNS International Journal of Computer Science and Network Security*, 10(5):121–128, 2010.

[52] Bernhard Krach and Patrick Robertson. Integration of foot-mounted inertial sensors into a Bayesian location estimation framework. *2008 5th Workshop on Positioning Navigation and Communication*, 2008(2):55–61, 2008.

[53] Matthias Kranz, Carl Fischer, and Albrecht Schmidt. A comparative study of dect and wlan signals for indoor localization. In *Pervasive Computing and Communications (PerCom), 2010 IEEE International Conference on*, pages 235–243. IEEE, 2010.

[54] Henrik Kretzschmar and Cyrill Stachniss. Information-theoretic compression of pose graphs for laser-based slam. *The International Journal of Robotics Research*, 31(11):1219–1230, 2012.

[55] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g 2 o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3607–3613. IEEE, 2011.

[56] Yasir Latif, César Cadena, and José Neira. Robust loop closing over time for pose graph slam. *The International Journal of Robotics Research*, page 0278364913498910, 2013.

[57] Gim Hee Lee, Friedrich Fraundorfer, and Marc Pollefeys. Robust pose-graph loop-closures with expectation-maximization. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 556–563. IEEE, 2013.

[58] Hui Liu, Houshang Darabi, Pat Banerjee, and Jing Liu. Survey of wireless indoor positioning techniques and systems. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(6):1067–1080, 2007.

[59] Feng Lu and Evangelos Milios. Globally consistent range scan alignment for environment mapping. *Autonomous robots*, 4(4):333–349, 1997.

[60] Mohamed R Mahfouz^1, Aly E Fathy, Michael J Kuhn^1, and Yahzou Wang. Recent trends and advances in uwb positioning. 2009.

[61] Google Indoor Maps. https://www.google.com/maps/about/partners/indoormaps/ (accessed september 2014).

[62] Mladen Mazuran, Gian Diego Tipaldi, Luciano Spinello, and Wolfram Burgard. Nonlinear graph sparsification for slam. In *Proc. Robot.: Sci. &amp; Syst. Conf*, pages 1–8, 2014.

[63] Michael Montemerlo and Sebastian Thrun. Large-scale robotic 3-d mapping of urban structures. In *Experimental Robotics IX*, pages 141–150. Springer, 2006.

[64] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *AAAI/IAAI*, pages 593–598, 2002.

[65] Raul Mur-Artal, JMM Montiel, and Juan D Tardós. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[66] John-Olof Nilsson, Amit K Gupta, and Peter Händel. Foot-mounted inertial navigation made easy. In *Indoor Positioning and Indoor Navigation (IPIN), 2014 International Conference on*, pages 24–29. IEEE, 2014.

[67] John-Olof Nilsson, Isaac Skog, Peter Händel, and KVS Hari. Foot-mounted ins for everybody-an open-source embedded implementation. In *Position Location and Navigation Symposium (PLANS), 2012 IEEE/ION*, pages 140–145. IEEE, 2012.

[68] Henri Nurminen, Anssi Ristimaki, Simo Ali-Loytty, and Robert Piché. Particle filter and smoother for indoor localization. In *Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on*, pages 1–10. IEEE, 2013.

[69] Edwin Olson. *Robust and efficient robotic mapping*. Phd thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, June 2008.

[70] Edwin Olson and Pratik Agarwal. Inference on networks of mixtures for robust robot mapping. *The International Journal of Robotics Research*, 32(7):826–840, 2013.

[71] Edwin Olson, John Leonard, and Seth Teller. Fast iterative alignment of pose graphs with poor initial estimates. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2262–2269. IEEE, 2006.

[72] Edwin Olson, John J Leonard, and Seth J Teller. Spatially-adaptive learning rates for online incremental slam. In *Robotics: Science and Systems*, 2007.

[73] Veljo Otsason, Alex Varshavsky, Anthony LaMarca, and Eyal De Lara. Accurate gsm indoor localization. In *UbiComp 2005: Ubiquitous Computing*, pages 141–158. Springer, 2005.

[74] Nissanka Bodhi Priyantha. *The cricket indoor location system.* PhD thesis, Massachusetts Institute of Technology, 2005.

[75] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[76] Paul Robertson, Martin Frassl, Michael Angermann, Marek Doniec, Brian J Julian, Maria Garcia Puyol, Mohammed Khider, Michael Lichtenstern, and Luca Bruno. Simultaneous localization and mapping for pedestrians using distortions of the local magnetic field intensity in large indoor environments. In *Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on*, pages 1–10. IEEE, 2013.

[77] Michael Roth, Fredrik Gustafsson, and Umut Orguner. On-road trajectory generation from gps data: a particle filtering/smoothing application. In *Information Fusion (FUSION), 2012 15th International Conference on*, pages 779–786. IEEE, 2012.

[78] Zafer Sahinoglu, Sinan Gezici, and Ismail Guvenc. Ultra-wideband positioning systems. *Cambridge, New York*, 2008.

[79] Simo Särkkä. *Bayesian filtering and smoothing*, volume 3. Cambridge University Press, 2013.

[80] Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry [tutorial]. *Robotics &amp; Automation Magazine, IEEE*, 18(4):80–92, 2011.

[81] Isaac Skog, John-Olof Nilsson, and Peter Händel. Evaluation of zero-velocity detectors for foot-mounted inertial navigation systems. In *Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on*, pages 1–6. IEEE, 2010.

[82] Adam Smith, Hari Balakrishnan, Michel Goraczko, and Nissanka Priyantha. Tracking moving devices with the cricket location system. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 190–202. ACM, 2004.

[83] Kalyan Pathapati Subbu, Brandon Gozick, and Ram Dantu. Locateme: Magnetic-fields-based indoor localization using smartphones. *ACM Trans. Intell. Syst. Technol.*, 4(4):73:1–73:27, October 2013.

[84] Niko Sünderhauf and Peter Protzel. Switchable constraints for robust pose graph slam. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1879–1884. IEEE, 2012.

[85] Niko Sünderhauf and Peter Protzel. Towards a robust back-end for pose graph slam. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1254–1261. IEEE, 2012.

[86] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics.* MIT press, 2005.

[87] Sebastian Thrun, Yufeng Liu, Daphne Koller, Andrew Y Ng, Zoubin Ghahramani, and Hugh Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *The International Journal of Robotics Research*, 23(7-8):693–716, 2004.

[88] D. Titterton and J. Weston. *Strapdown Inertial Navigation Technology*. The American Institute of Aeronautics and Astronautics, 2004.

[89] John Vial, Hugh Durrant-Whyte, and Tim Bailey. Conservative sparsification for efficient and consistent approximate estimation. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 886–893. IEEE, 2011.

[90] Sen Wang, Hongkai Wen, Ronald Clark, and Niki Trigoni. Keyframe based large-scale indoor localisation using geomagnetic field and motion pattern. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 1910–1917. IEEE, 2016.

[91] R. Want, A. Hopper, V. Falcão, and J. Gibbons. The active badge location system. *ACM Transactions on Information Systems (TOIS)*, 10(1):91–102, 1992.

[92] Andy Ward, Alan Jones, and Andy Hopper. A new location technique for the active office. *Personal Communications, IEEE*, 4(5):42–47, 1997.

[93] O Woodman and R Harle. Pedestrian localisation for indoor environments. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 114–123. ACM, 2008.

[94] Oliver Woodman and Robert Harle. RF-Based Initialisation for Inertial Pedestrian Tracking. *Pervasive Computing 7th International Conference Pervasive 2009 Nara Japan May 1114 2009 Proceedings*, 5538:238–255, 2009.

[95] Oliver J Woodman. An introduction to inertial navigation. *University of Cambridge, Computer Laboratory, Tech. Rep. UCAMCL-TR-696*, 14:15, 2007.

[96] Oliver J. Woodman. *Pedestrian localisation for indoor environments*. Phd thesis, Computer Laboratory, University of Cambridge, September 2010.

[97] Oliver J Woodman and Robert K Harle. Concurrent scheduling in the active bat location system. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, pages 431–437. IEEE, 2010.

[98] Zhuoling Xiao, Hongkai Wen, Andrew Markham, and Niki Trigoni. Lightweight map matching for indoor localisation using conditional random fields. In *Information Processing in Sensor Networks, IPSN-14 Proceedings of the 13th International Symposium on*, pages 131–142. IEEE, 2014.

[99] Moustafa Youssef and Ashok Agrawala. The horus wlan location determination system. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, MobiSys '05, pages 205–218, New York, NY, USA, 2005. ACM.