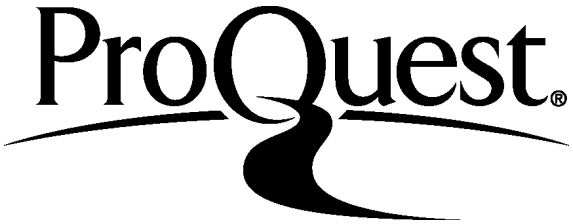
Unsupervised Indoor Localization for Smartphones

by

Shervin Shahidi

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Electrical and Computer Engineering
University of Toronto



Abstract

Unsupervised Indoor Localization for Smartphones

Shervin Shahidi
Doctor of Philosophy
Graduate Department of Electrical and Computer Engineering
University of Toronto
2016

In this work, the problem of indoor localization for smartphones is investigated. An indoor localization system is designed, that relies on two general techniques: received signal strength based localization and dead reckoning. The system is designed to be trained by the collected data via crowdsourcing, with no need for active human intervention. Sensor fusion techniques are used to resolve different issues of the localization problem. The combination of accelerometer and gyroscope are used as a dead reckoning system to measure the user's displacement.

Graph matching algorithms are used to merge crowdsourced data to make a unified data structure, in a fully unsupervised fashion. We investigate the problem of translating topological indoor localization to geographical localization, by modeling the building map and the semantic maps as graphs. Two matching algorithms are proposed along with a node similarity measure based on finding the minimum distance between all sets of permutations of two vectors. We provide an efficient algorithm to calculate the similarity measure, and prove its correctness via a theorem. The proposed systems are implemented and shown to perform well, both in simulations and on real data.

A novel magnetic sensor based localization system is also introduced and implemented that does not require any available infrastructure in the indoor area. Sensor fusion techniques are used to improve the system's accuracy.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Professor Shahrokh Valaee, for his constant support and his precious comments. I will always be indebted to him for his guidance and motivation during this thesis, which will accompany me throughout my future life. I would like to extend my gratitude to all of my fellow labmates in the Wireless and Internet Research Lab (WIRLab), particularly Hamed Sadeghi and Kamran Jafari whom I had the chance to work with more closely, for all of the thoughtful discussions and their invaluable feedbacks. I also thank Prof. Hatzinakos, Prof. Kundur, Prof. Adve, Prof. Liang, and Prof. Banihashemi for serving in my thesis committee and their insightful comments.

I am grateful to the love of my life, Parisa, for if it wasn't for her unconditional support and motivating words, I would not have been standing where I am now.

My sincere thanks also goes to my parents, my sister, and my in-laws who have always encouraged me. Without their spiritual support and patience, I would never have been able to complete this thesis.

Last but not least, I am grateful to my dear friends for their help, kind support and understanding during this period. Specifically, I would like to thank Navid, Peyman, Arin, Kaveh, Shadi, Iraj, Keyvan, Nazanin, and Veria for all of the fun and the adventures we had during this period.

Contents

List of Tables

List of Figures

List of Acronyms

AOA Angle Of Arrival

AP Access Point

BFS Breadth-First Search

CDF Cumulative Distribution Function

DR Dead Reckoning

DTW Dynamic Time Warping

GNSS Global Navigation Satellite System

GSM Global System for Mobile Communications

HMM Hidden Markov Model

IMU Inertial Measurement Unit

INS Inertial Navigation Systems

KNN K Nearest Neighbour

LBS Location Based Services

MDS MultiDimensional Scaling

MEMS Micro Electro-Mechanical Systems

MMSE Minimum Mean Square Error

MSE Mean Square Error

NN Nearest Neighbour

PDR Pedestrian Dead Reckoning

RSS Received Signal Strength

SLAM Simultaneous Localization And Mapping

TDOA Time Difference Of Arrival

TOA Time Of Arrival

Chapter 1

Introduction

1.1 Motivation

Although the problem of localizing wireless mobile sensor nodes has attracted notable attention in the last decade, designing a real-time and efficient solution for indoor area is still a challenging and open problem. The notable attention attracted by indoor localization recently is mostly due to the various emerging location based services (LBS), that can only be provided by knowing user's location, especially in indoor areas. According to studies since 1990s, it is reasonable to assume that more than 80% of human's life is spent in indoor area [32, 45].

Almost all of the location based services are based on one or a combination of three major problems: localization, tracking, and navigation. The indoor localization problem is defined to be the task of finding the location of a user in a given area, such as a shopping mall or a department. The tracking problem entails following a user's location and increasing the localization accuracy by exploiting the correlation between the user's current state and it's previous state. The problem of navigation is more related to finding a feasible and optimal (if possible) path from a given location (usually the user's current location) to a destination.

Providing a practical and sensible solution for the localization problem (as the core of LBS,) opens a door to various possible services that will make a significant impact on many aspects of human's everyday life, including safety and accessibility. To name a few, the location based services can be extended from outdoors to indoors, providing indoor navigation to the location of an office in a department, a store in a shopping mall, or an item of interest in a store. There are also numerous applications for people with special needs, such as providing indoor navigation for the blind, locating the elderly or children inside an indoor area, or to help the police in locating the ones in danger in a

large (possibly multi-floor) indoor area, as opposed to the current systems that can only roughly identify the building's location.

1.2 Background

The indoor localization problem differs from the outdoor settings for two reasons: first, global navigation satellite system (GNSS) which is the most acceptable solution for outdoor environments is not effective for indoors due to lack of line of sight. Second, the common triangulation and trilateration techniques that are utilized for outdoor localization, perform poorly in indoor areas due to the heavy multipath and fading in the complex structure of buildings.

Generally, most of the existing literature try to address the indoor localization via one or a combination of three common approaches: trilateration and triangulation, received signal strength (RSS) based fingerprinting (also known as scene analysis [50]) and proximity. Recently, thanks to the advance in micro electro-mechanical systems (MEMS) technology which is currently available on smartphones, inertial navigation systems (INS) have also become very attractive.

- The RSS-fingerprinting methods [3, 40], take advantage of the available received signal strength from surrounding signals, such as Wi-Fi signals, Global System for Mobile Communications (GSM) signals, Bluetooth etc., and find the location of a user by exploiting the variation of patterns in signal strength of each location. These methods have shown to be promising for localizing wireless nodes in indoor areas, since they provide acceptable accuracy with small to no need for extra infrastructure installment in the indoor area. However, these techniques often suffer from the need of time and manpower consumption for site-survey and training, making these approaches hard to scale. In the training (or calibration) phase, a trained technician is often required to gather signal fingerprints from the area of interest, also known as collecting "labeled data". The set of labeled data for each location of the building are then used to generate a signal space heat map of the building, called the building's *radio map*. In the online (or operation) phase, available signals in the location of interest are scanned and matched against the radio map to obtain the location estimation.

- Trilateration and triangulation techniques, such as the time-of-arrival (TOA) [35], time difference of arrival (TDOA) [49], and angle of arrival (AOA) [87, 81], estimate

the location of a device based on its range or the angle of received signals from several transmitters. These techniques mostly require precise time synchronization, as well as line of sight between the transmitter and the user [50]. Although lateration techniques have been extensively investigated in the literature, most of the works have considered outdoor applications, where having line of sight and minor fading is a more realistic assumption. In indoor environment, implementation of such systems usually requires extra infrastructure costs for installation of proper transmitters with known locations. On the other hand, the RSS-fingerprinting methods can be implemented in most of the indoor areas without the need for any extra infrastructure, since the transmitters will be Wi-Fi hotspots or GSM transmitters and the locating devices can be off-the-shelf smartphones, available to most of the users.

- Proximity approaches, which are usually more case specific, require proximity sensors such as RFID tags [36] or IR sensors [16] or Bluetooth unique tags [61] to recognize a user's vicinity once he/she passes by one of the installed tags with known positions. Depending on the application and the coverage area, these methods can be easy or very costly to implement.

- Inertial navigation Systems (INS), are the systems that perform the navigation according to inertial measurement units (IMUs), i.e., accelerometer, gyroscope, and possibly magnetic field sensors. Given the initial position, the location at each time can be calculated by finding the displacements in the previous time steps iteratively. This process is also known as *dead reckoning* (DR). These approaches, although accurate in the beginning, start to accumulate noise and their accuracy degrades after a short while. There are several works investigating DR for the purpose of pedestrian tracking, as well as indoor localization, such as [65, 31, 92, 78].

Note that in the first three categories, the exploited signals used for localization are not limited to specific signals and can be any possible fields or waves, such as electromagnetic (e.g., Wi-Fi and Bluetooth), sound [85], optical [34], or magnetic waves [38].

In this work, we mainly focus on the RSS fingerprinting methods, specifically using Wi-Fi signals, since these signals are generally more available in indoor areas. We are interested in a practical solution to the indoor localization problem, one that satisfies three requirements: *reliability*, *scalability*, and *feasible cost*. The RSS based techniques, as mentioned, are reliable and cheap. In order to overcome the scalability requirement, we investigate solutions for reducing the initial training costs as well as maintaining the radio map. To this aim, we investigate inertial navigation systems, as we believe a final

answer to the indoor localization problem should be a combination of different techniques. We use DR to remove the training phase efforts of the RSS fingerprint method. The same technique can also be used in the online phase to track the user's location, while the drift error of DR is being limited using RSS based locationing ([64, 76]).

To further remove the dependency on infrastructure, we also investigate the possibility of using magnetic field sensor readings as the signals that populate the radio map. The main motivation for using magnetic field based localization is that magnetic fields are available in all of the indoor areas, even where Wi-Fi signals are unavailable or when the building is out of power in emergency situations.

1.3 Related work

There are similar works attempting to reduce the training phase efforts for RSS-based techniques. Some of the works suggest using semi-supervised learning algorithms, rather than supervised algorithms, to reduce the amount of needed labeled data for training. These approaches usually require a portion of the data as labeled and the rest as *unlabeled data*. The unlabeled data are RSS values scanned in the building with no physical location information. To name a few, [22, 60, 57, 51] have shown promising results. Using semi-supervised techniques, the amount of needed labor for calibration phase of the localization system can be reduced, but not completely eliminated.

Other approaches suggest using a combination of localization techniques to help overcome the issues of each approach, or introduce a way to gather labeled data in a less costly manner. For example, [6] suggests recording the time data is gathered, and asking the user to label the data asynchronously in an offline mode. While, this method might ease the data labeling process, it can only work when an accuracy at room level is desired; otherwise, the user will not remember his exact location at each time, during the day.

Jin et al. [33] use inertial navigation for localization, while correcting the estimations via installed radio frequency and ultra sonic beacons of cricket [62] sensors. Woodman et al. [91, 92] utilize a foot mounted IMU to locate the user's location and floor via DR. They use particle filters as well as map constraints to limit the drift error of DR, while using Wi-Fi RSS readings to only estimate the initial location of the user. A similar approach is also used in [58]. All of the mentioned works, and many similar ones, aim to improve the reliability, cost, or scalability issues of indoor positioning by using one technique as the main estimator, and correct the estimations via a side information source, such as floor plan, or a second localization technique (also c.f. [89, 80]).

Some works have proposed to overcome the localization problem via simultaneous

localization and mapping (SLAM). These systems usually apply sensor fusion on inertial sensors and RSS fingerprints. The most common tools for sensor fusion in SLAM (both for human and robot localization) are Kalman filtering and sequential Monte Carlo methods, also known as particle filtering. Using SLAM approaches, gives the ability to reduce the need for human intervention. It also brings the possibility of using crowdsourced data for building a location's radio map [78, 39, 23].

Crowdsourcing is one of the most promising approaches to tackle the scalability problem of localization systems [39, 78, 96, 64]. Crowdsourcing can be specifically beneficial to RSS-based methods, since the training phase cost for generating the radio map makes it unscalable. In such systems, each sensor node (or user) contributes to the training data-set by collecting data from a part of the whole area, i.e., its own movement trace. The collected data in the form of movement traces are then matched and merged, similar to pieces of a puzzle, to obtain a semantic pathway map, that represents the passable areas of a building. The obtained pathway map can be used to localize the user within the explored pathways of the building. In order to obtain the location of the user geologically on the floor plan, after successful matching and merging of the collected data, the pathway map should be matched to the geological map of the indoor area.

Most of the related works output a semantic map of passable areas (the pathway map), and are able to locate the user within that map. However, the semantic map is generally scale and direction free and matching such semantic map to the actual floor plan is often not addressed well, is too complex, or is done manually by finding a proper direction and scale to fit the map to the floor plan. To name a few, [78, 77, 59] have proposed systems that obtain such semantic maps using smartphones.

Some related works use different models to exploit the floor plan's information and match their obtained semantic maps to the floor plan, such as [64, 58, 96]. In [64, 58], particle filters are utilized to correct the path obtained from inertial sensors and Wi-Fi signals to build and maintain the radio map. Using particle filters, these works can directly utilize the building's passable areas to find the user's location. Unfortunately, particle filters have a high complexity since each particle should be updated at every step in the online phase. Therefore, the preference is with the approaches that do the data processing in an offline calibration phase, preferably on a server to avoid draining the wireless device battery. To be more specific, the common method used in the literature for representing the floor plan is by defining impassable and passable objects (polygons). At each step, all particles are checked to be within allowed polygons and if a particle passes through an impassable object (such as a wall,) its weight becomes zero. Since the initial number of particles to cover a large indoor area should be very high and each

particle must be separately evaluated, such approaches impose a high complexity to the system. Furthermore, representation of the floor plan by defining all the passable and impassable polygons is a time consuming process.

In [96], multidimensional scaling (MDS) is used to convert the floor plan and its obstacles into a relaxed (obstacle free) N-dimensional space, where distances are simply the Euclidean distances between the points of the map. After converting both the floor plan and semantic maps into relaxed maps, the two maps are compared based on several different levels to find the corridors, their directions, the entrance doors, and the rooms. However, since the result for a 2 or 3 dimensional space obtained from MDS is not necessarily unique, the two relaxed maps are not necessarily similar. Hence, there is a possibility of not obtaining a meaningful response.

Overall, in the mentioned works, the main concentration is usually not on the calibration phase, but on the operation phase of the system and its accuracy, possibly on a semantic map. However, after generating the semantic maps from sensory information, matching the floor map to the semantic map and the accuracy of such matching is not discussed or investigated well.

Our key concentration in this work is mainly on the calibration phase of the system, aiming for a low cost practical system, with no need for supervision or extra infrastructure and making only minor assumptions about the environment or the users. Once such calibration is done, a wide range of RSS based and DR based localization algorithms can be applied in the operation phase.

The current off-the-shelf smartphones are not only equipped with wireless transmitters and receivers, but also several sensors such as accelerometers, gyroscopes and magnetic sensors. Such sensors, turn each smartphone into a mobile sensor node that can perceive various data from the environment and obtain meaningful information, such as location and orientation. Therefore, we consider the smartphones as the mobile sensors to be located within an indoor environment. Regarding the environment, we do not require any extra infrastructure in the building. Only presence of the Wi-Fi signals and availability of an indoor floor plan is assumed. Furthermore, we do not require a specific placement of Wi-Fi access points (AP), nor do we have need of the AP locations.

We design and implement a system that is trained in an unsupervised manner via crowdsourcing. Each smartphone user can both use the system and also contribute to the training data. Once enough unsupervised training data is collected, we generate a reliable radio map. Then, any localization technique can be employed in the operation phase of the system.

1.4 Contributions

As stated above, we investigate the RSS fingerprinting techniques as the core of our indoor localizer. These techniques have been largely investigated recently, since they can be implemented without the need for any extra infrastructure. We use the already installed Wi-Fi access-points in the buildings and off-the-shelf smartphones.

Along the main concentration of this work, we briefly describe our implementation of a DR system for counting steps and detecting heading direction (also known as pedometer), as a building component for the unsupervised localization system. To make the pedometer practical, we have tried to make minimal assumptions about the user's gestures, walking speed or phone orientation during the training. The user is allowed to make occasional stops during his walk. The DR module is used as a component in the rest of the work.

In our main work, we propose and implement a crowdsourced based system to generate the radio map completely unsupervised, by exploiting information from the existing users that regularly walk in the area. The users do not need to provide any information such as labeling their current location or marking their walking path. We keep track of the users' movements as well as scanning the RSS values while the users do not do any active contribution to the process. Once enough data is collected, we merge all of the users' data, and build a radio map separately in an offline phase. Once the radio map is built (in an unsupervised and offline fashion), variety of supervised or semi-supervised localization techniques can be employed to perform the online localization task. Such system requires several main components that will be investigated and addressed in this work successively:

- Extracting location information from the floor map in the physical domain.
- Extracting the location fingerprints of the indoor area in the RSS domain.
- Modeling the building floor and the radio map as a graph.
- Problem formulation as a graph matching problem and a proper solution for the problem.

We model the floor plan and the collected user traces as graphs. Our graph model for the paths in the floor, not only provides the possibility of low complexity matching and radio map error correction, but also significantly reduces the complexity of the tracking and navigation tasks.

After representing the data on graphs, we define a graph matching problem, not only to generate the pathway map of the users via crowdsourced data, but also to automatically match the obtained pathway map to the floor plan with notable accuracy. To that aim, we have developed two graph matching techniques. For the first approach, we have defined a graph similarity measure and an efficient way to calculate it, along with an efficient matching algorithm we call K-best fits, which is a fast heuristic[1]. The second approach is a probabilistic method, based on hidden Markov models (HMM)[2]. The K-best fits algorithm uses the similarity of nodes based on their global topological location in the graph, while the HMM-based approach uses the local properties of nodes and edges for comparison.

Our proposed measure and matching techniques are not limited to such application and can be applied to any two graphs with similar settings. In fact, they can also be applied to some of the related works that produce a semantic pathway map, to improve the accuracy of radio map, by adjusting the location tags in the radio maps, and also translation of topological localization (finding the location on the semantic pathway map) to geological localization (finding the location on the indoor map). Our method is independent of the localization algorithm used in the online phase. Therefore, it is compatible with the majority of the conventional RSS based methods[3].

As a further step, in order to relax the assumption about the availability of Wi-Fi signals, we also investigate the possibility of localization using the distortions of magnetic field waves, caused by the metal infrastructure of the buildings. These signal distortions exist in every building with metal infrastructure, even when building is out of power. We propose a novel and promising technique for using the magnetic field readings for indoor localization. The novelty of our magnetic based system has several aspects: the method magnetic sensor is used, the model we have introduced in fusing the DR information into magnetic pattern matching results in relaxing many common assumptions and constraints introduced in the state-of-the-art approaches[4].

1.5 Organization of the thesis

The organization of this thesis is as follows:

In Chapter 2, we express the problem formulation, establish the notations, and explain the solution we have proposed on the crowdsourced based indoor localization. The

[1]Published as [71]
[2]Published as [73]
[3]The system is U.S. patent pending. The system is also submitted for peer-review [70])
[4]An early version of the work is published as [72]

implementation details and results of the crowdsourced system are given in Chapter 3. In Chapter 4, we present a DR system and its implementation. The DR module is used as an element in the rest of the work. We investigate the possibility of using magnetic field sensor values for indoor localization, in Chapter 5. Finally, Chapter 6 is devoted to conclusion and proposed future works.

Chapter 2

Designing a Crowdsourced Indoor Localization System

2.1 Modeling via Graph theory

As mentioned in Chapter 1, we are interested in the problem of unsupervised indoor localization. In order to train the system in an unsupervised fashion, we use crowdsourced data to generate a training data-set known as radio map.

In this section, after a brief introduction to graph theory and establishing necessary notations, we describe how we model the problem of radio map generation, using graph theory. More detailed definitions and properties of graphs can be found in renown graph theory references, e.g., [90, 7].

- A graph $G = (V, E)$ is an ordered pair of two sets, where V is a set of vertices (also called nodes), and E is a set of edges. For a directed graph, each edge is an ordered pair of vertices $e_{v_1,v_2} = (v_1, v_2), v_1, v_2 \in V$, where in an undirected graph, $e_{v_1,v_2} \in E$ implies $e_{v_2,v_1} \in E$.

- An edge weighted graph, or simply a "weighted graph", $G = (V, E, w)$ is a graph with a set of values w associated to each of its edges, such that $w_{e_{i,j}} \in w, \forall e_{i,j} \in E$. For simplicity, we denote the weight $w_{e_{i,j}}$ as $w_{i,j}$.

- A graph is called planar if it can be embedded in a plane, so that all of its edges and vertices are in the plane, with no intersecting edges.

- A bipartite graph is a graph that its edges divide its vertices into two disjoint sets, where each edge of the graph connects one of the members of the first set, to a member of the other set.

- A "path" through a graph is a sequence of nodes (with no repetitions) connected by edges [98]. A graph is called connected if there exists a path between all of its nodes.

- A graph can be represented via either a set of pairs, known as "adjacency list", describing the edges and the vertices it connects, or an "adjacency matrix" $A = [x_{ij}]_{i,j=1,\dots,n}$, describing the connectivity and edge weights of each pair of nodes, such that the element $x_{ij} = w_{ij}$ if nodes i and j are connected via an edge with weight w_{ij}, and $x_{ij} = 0$, otherwise.

Figure 2.1 demonstrates examples of directed,undirected, weighted, connected, and planar graphs. Below, we bring some extra definitions in graph theory that will be used throughout this work.

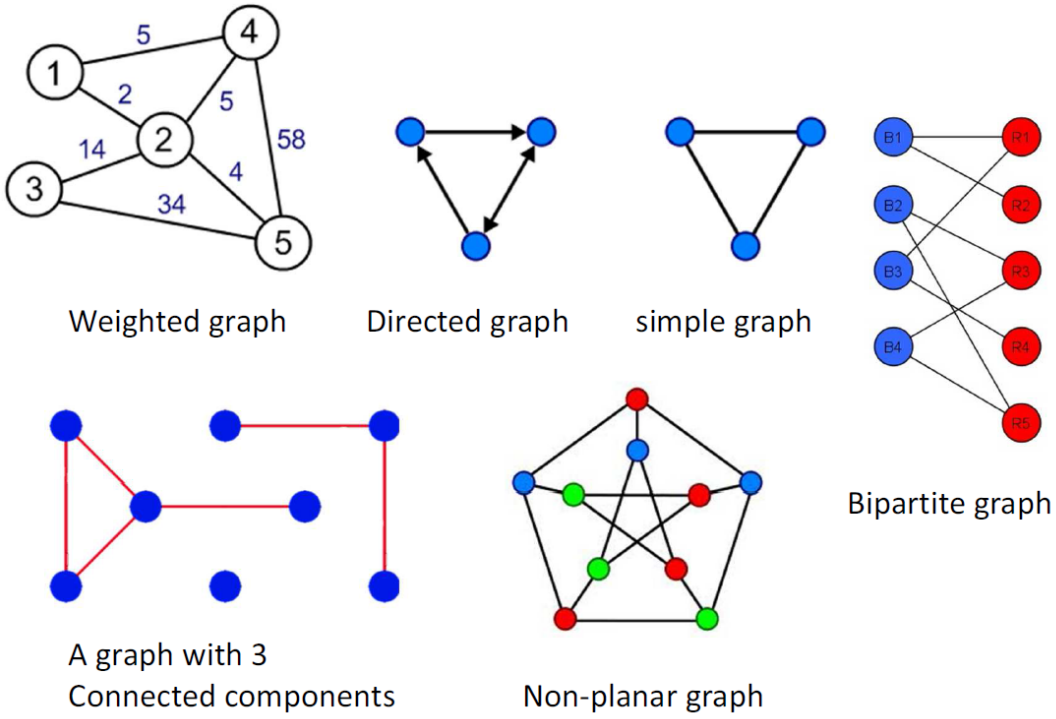


Figure 2.1: Samples of different types of graphs

- "The shortest path" problem on a weighted graph is the problem of finding a path between two vertices, such that the sum of the weights of the edges along the path is minimum. An "all pairs shortest path problem", seeks the shortest path between each pair of vertices in the graph. There are well-known shortest path algorithms, such as Dijkstra algorithm, Bellman-Ford algorithm and Floyd-Warshall algorithm that can all calculate all pairs shortest path. The details of

these algorithms as well as computational complexity analysis are found in [46]. Particularly, the time complexity of Floyd-Warshall algorithm is $\mathcal{O}(|V|^3)$ and the time complexity of Johnson's algorithm (which uses a combination of Bellman-Ford and Dijkstra algorithms) is $\mathcal{O}(|V||E| + |V|^2 \log |V|)$, where $|V|$ is the number of vertices and $|E|$ denotes the number of the edges in the graph.

- The graph matching (or edge independent set) problem is the problem of finding a subset of graph edges, such that none of the edges share vertices. This problem is more interesting on bipartite graphs, where the matching is similar to assigning the vertices of one part of the graph to another, with no vertex appearing twice in the assignment. A specific case of this problem is the "maximum weighted bipartite matching", which is defined as a matching on a bipartite graph with maximum sum of the weights of selected edges. This problem is also known as the "assignment problem". There are well-known algorithms for this problem such as the Hungarian algorithm (originally called by this name in [42]). Another method to solve this problem is to run the Bellman-Ford shortest path algorithm on an augmented graph.

- The graph rigidity; if we consider a graph as a system of rods and joints, a graph can form a rigid structure. A "rigid graph" is a graph that its edges and vertices are like rods and joints, so that the edges have fixed length (usually equal to the weight of the edge) and vertices cannot freely move without changing the rigid structure of the graph. For example, if the weights of the edges be the Euclidean distance of the nodes, in a rigid graph, each node can only move in a way that its distance to the other connected nodes is kept constant and equal to the connecting edge's weight.

We now describe the problem formulation by modeling two graphs: the *ground truth graph* which is built offline, based on the building map, and the *data graph*, which is built in an unsupervised fashion, based on the readings obtained from users walking in the environment.

As mentioned previously, a radio map is needed for RSS-based localization, which consists of a data-set of RSS fingerprints, tagged with their physical coordinates (labeled points). In our approach, instead of manually building such data-set, we first define a graph, called the ground truth graph $\mathcal{G}_T$, that only contains the physical coordinates of specific points. Such graph can be easily built using a graphical interface and a given floor plan. These points have physical location labels, but no RSS fingerprints. Therefore, we define a second graph that is built from crowdsourced data. The second graph, called

the data graph $\mathcal{G}_D$, is collected while the users with unknown locations are walking in the building, engaged in their regular daily activities. While the users walk, RSS samples with unknown physical locations, along with the user's relative displacements are collected and merged into $\mathcal{G}_D$. $\mathcal{G}_D$ has the RSS information but no absolute physical location information. However, using a dead reckoning system introduced in Chapter 4, $\mathcal{G}_D$ also contains relative distance information between its nodes. Once both $\mathcal{G}_T$ and $\mathcal{G}_D$ are obtained, we apply a graph matching algorithm to find a correspondence between the nodes of $\mathcal{G}_T$ and $\mathcal{G}_D$ based on their topological structure and yield a set of points with both physical coordinates and RSS samples, i.e., the radio map. Figure 2.2 demonstrates the scheme of our approach.

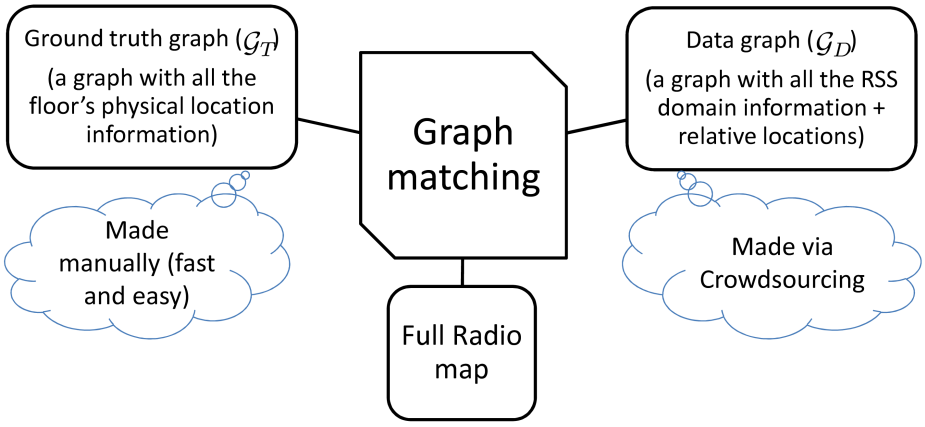


Figure 2.2: The schematic diagram of the proposed crowdsourced system.

Below, we present a formal definition of the two graphs $\mathcal{G}_T$ and $\mathcal{G}_D$.

2.2 Modeling the building map as the ground truth graph $\mathcal{G}_T$

In order to model the building map as a graph, we discretize the floor plan by considering the walkable areas as the edges of a graph. The vertices of the graph will be considered as the joints to connect the edges. In other words, the walkable areas in the floor plan will be modeled by rods and joints, considered as the edges and vertices of a rigid graph. In order to reduce the complexity of the model, we have limited the angles in which the rods can join in to 4 (allowing only horizontal and vertical edges). For example, a straight narrow hallway can be modeled by a set of points put on straight lines along the passable area (Figure 2.3). Using 8 possible angles will also allow diagonal movements in the building when large hallways exist.

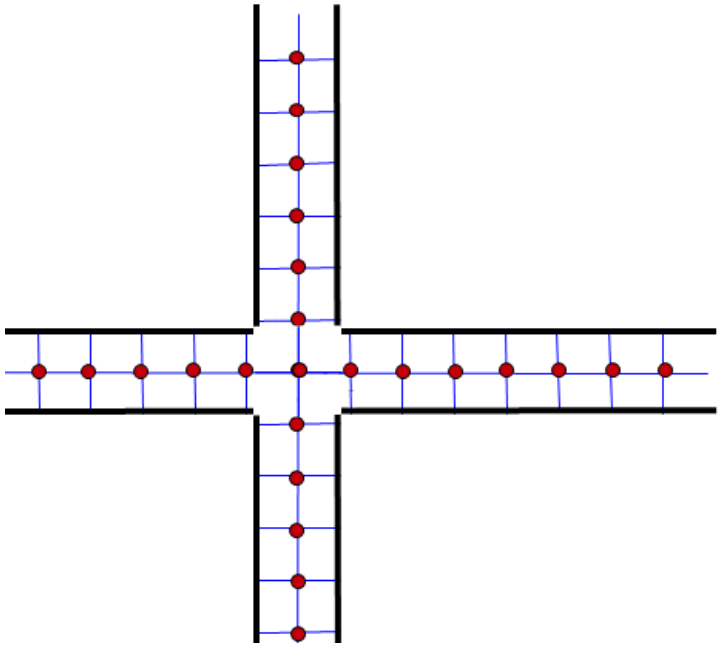


Figure 2.3: Set of labeled points in a hallway (the red dots)

Although this modeling introduces a quantization noise to the localization system, it is vital for tasks such as gathering RSS readings on the labeled points to generate the radio map. Localization to a set of discrete points also reduces the complexity of localization, as well as drift error in tracking. Furthermore, in navigation, these points are necessary for finding a path from the starting point to the destination. The quantization noise can be reduced by increasing the density of the labeled points on the floor map, and/or performing interpolation between the labeled points, specially in the operation phase of the system. Collecting a discrete set of labeled points and performing interpolation is common in most of RSS fingerprinting methods, such as [3, 97, 60].

Define the set of labeled points, $\mathcal{L}$ to be the set of vertices of a graph, representing the floor map. The set of edges $e_{i,j} \in \mathcal{E}$, where $i, j \in \mathcal{L}$ is also defined as the ordered path (i, j) if one can walk from i to j without passing through any other graph vertex. Note that the graph is undirected since a walkable path from i to j implies a walkable path from j to i. We also define the Euclidean distances between two labeled points on the floor to be the weight of edges connecting two points, such that $w_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$, where x_i and y_i denote the physical coordinate of the labeled point i on the floor, according to a given coordinate origin. Denote the resulting weighted, undirected graph by *ground truth graph* $\mathcal{G}_T = (\mathcal{L}, \mathcal{E}, W)$.

The resulting graph, not only is used to build the radio map in the training phase, but also can be used for tracking (to limit the error of locationing using the movement trace

of the user on $\mathcal{G}_T$) and navigation (to show the shortest path to the destination using the underlying graph $\mathcal{G}_T$) in the operation phase of the system. Another advantage of the graph model is to yield a density of appearance of the users on each node in the operation phase, estimating the distribution of user presence inside a building. Such a distribution can be useful for resource management purposes, as well as improving online localization accuracy for algorithms that can benefit from having prior location distribution.

Looking back at the main problem, in order to localize a device on the floor plan in the operation phase, we need a radio map. The radio map, in its simplest version, consists of a data-set of labeled points, i.e., points with RSS fingerprints tagged with their physical location coordinates. In a sense, the ground truth graph only forms a set of candidate labeled points based on their physical coordinates. The next step will be filling such candidate points with RSS information. Such information is populated in a second graph, i.e., the data graph.

2.3 Modeling the data graph $\mathcal{G}_D$

The *data graph* $\mathcal{G}_D$ is the graph built using the data generated by the users. In order to build the data graph, we need to recognize the points and define them as vertices. Note that due to the unsupervised nature of the problem, we do not have the location information. However, we can use some techniques to estimate the relative distance of the vertices and distinguish the loop closures and repetitive points.

Different proposed techniques for finding the relative distance between the points will be discussed shortly, below. To find the loop closures and repetitive points, we use the information in the unlabeled RSS readings. The system is designed to recognize the overlapping parts of two walks in the building and merge them, based on the common RSS readings. This process has some similarity to the operation phase of localization systems with a data-set of labeled data, except that in this case, instead of finding the physical location of the user *geologically*, only matching unlabeled points (repetitive points) are found and merged. This process, can be considered as *topological* localization on the data graph, since no physical coordinate is obtained. A major difference between this process and the operation phase is that the received RSS scans may not always be matched (and merged) to any previous point, since it might belong to a new unknown location that was not explored by the users previously. However, if the point was visited by the users previously, say a few days a go or even by another user, the system would recognize it and merge it to the proper point in the radio map.

Overall, the process of finding overlaps in user traces based on RSS scans and merg-

ing them to build $\mathcal{G}_D$ has some similarities and differences with geological localization. We present the details and challenges of practically designing this system as a separate component of the overall system, explained in Section 3.2. For the rest of this chapter, we assume that the problem of recognizing repetitive points is handled successfully and dedicate this chapter to other details.

Once we will be able to merge repetitive points, we can merge the unlabeled data points gathered by all users at all times and obtain a set of unlabeled points $\mathcal{U}$ for the floor map, that is expected to cover the whole walkable floor area or a sub-region in that area. Calling such unlabeled points as the vertices v_i^D, $i \in \mathcal{U}$, similar to the case for $\mathcal{G}_T$, we define the edges $e_{ij}^D \in \mathcal{E}^D$ to exist between the vertices if there exists a direct walkable path between those points.

The graph $\mathcal{G}_D = \{\mathcal{U}, \mathcal{E}_D, W^D\}$ is defined to be weighted, with weights w_{ij}^D equal to the relative distance between i, j, i.e., the distance from i to j.

The resulting graph will form a chain in a hall way, and a grid in big rooms (the grid estimates the paths inside the room). Also note that the relative distance between two points can either be a distance vector (having both magnitude and angle information) or a distance scalar (having the magnitude information only). In our work, for the sake of keeping the generality of the problem, we investigate both cases, with or without angle information, since some phones (or wireless mobile sensors) may not be equipped with the required sensors to provide angle information.

In order to find the relative distances and directions between the vertices, i.e., the edge weights of the $\mathcal{G}_D$, several methods are proposed and investigated.

Using accelerometer to count the steps

The accelerometer sensor can be used to count the steps, as a measure of distance. In Chapter 4, where the DR module is explained in details we present our implementation of a step counter that effectively counts the user steps, in arbitrary phone orientations and several gestures of holding the phone. The step counter provides distance information between two edges.

Using compass to find the direction information

The heading direction can be found using compass (magnetic field sensor). However, as will be seen in Chapter 4, such sensors in the smartphones have low accuracy and receive a large amount of disturbance in indoor area. Although the new phones provide a better accuracy via better firmwares, we still experienced some issues with the accuracy of these

sensors at certain locations, where the field is affected by large objects made of metal in the building, e.g., elevators, server rooms, air canals, etc. As a result, we are currently not relying on magnetic field sensor information.

Using gyroscope and gravity sensor to find the heading displacements

Another approach for finding heading direction is to use the gyro and gravity sensor (which in turn is obtained by combining gyroscope and accelerometer sensors). We can project the rotations to the X-Y plane and find the heading direction *displacements*, rather than the absolute heading direction, like what compass provides. The detailed description of this method, which is a part of the DR module, is provided in Section 4.6.

Using the correlation of RSS values to find relative distance and heading direction displacements

Another possible method of estimating the distance vector is using the RSS pattern while walking. Upon moving in a direction or making turns, the RSS values follow a specific pattern on each access point (AP) in the area. It is possible to use these patterns via angulation techniques to find relative distances, or changes in heading direction. For example, as shown in Figure 2.4, it is expected that while a user moves from point A to B, the RSS from AP1 decreases, the RSS from AP2 increases, and the RSS from AP3 remain almost constant. However, upon moving from B to C, the RSS from AP1 should remain constant, while RSS of AP2 and AP3 decrease and increase respectively. This kind of changes in the pattern may be used to detect changes in the heading direction, even without knowing the location of the access point. Using a propagation model for the signal strength, the amount of decrease/increase in the RSS values may also be used to find relative distances.

In some scenarios however, analyzing these patterns requires the knowledge of AP locations. Furthermore, the indoor area multi-path and fading of the signals introduce a large error in the distance/direction estimation. Overall, the lateration methods for localization often require perfect knowledge or an estimation of AP locations, such as the works in [27, 49, 87].

Using the floor map information to limit the number of possible heading directions

Given the location on the map, one can only move inside walkable areas. For example, if the user is inside a narrow hallway, he/she can only keep walking straight or return and

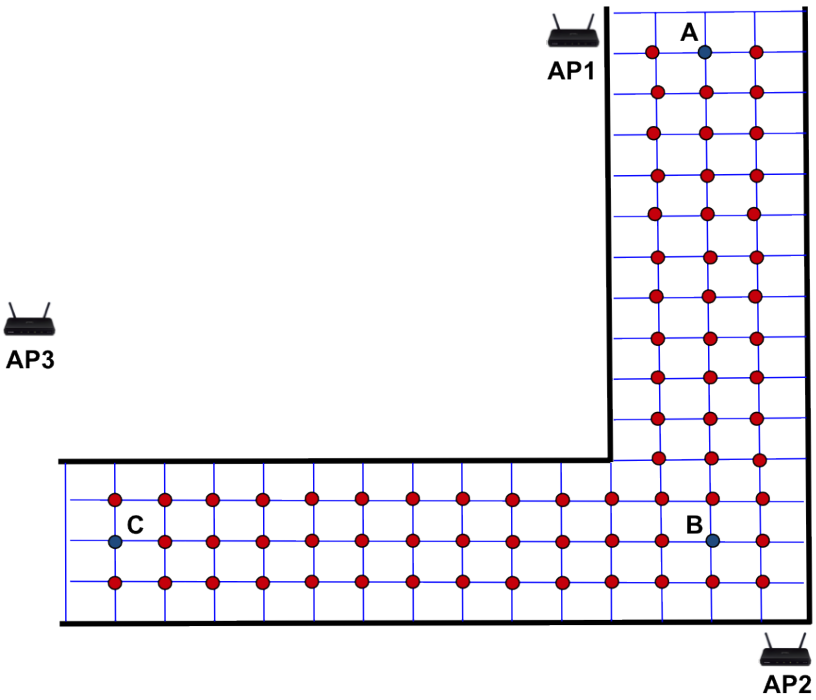


Figure 2.4: Using RSS change pattern to detect turns when the user walks from point A to points B and C

walk in the opposite direction. Similarly, on a turn, a user is expected to either turn or (unlikely) return and walk back toward his starting point. These information can also be extracted using the graph model we described by modeling the user's movement on the graph. However, employing the floor map information requires an estimation of user's current location, which is normally not available during the unsupervised training phase. Such information can be useful for confirming the accuracy of generated radio map, in a post-processing/refinement step, as well as the online phase, as a tracking component, to correct the estimation of user's location. Similar interpretation in also used in our hidden Markov model based graph matching approach, based on a sequence of random walk on the data graph (Section 2.6.4).

Graph realization and multidimensional scaling (MDS)

One of the other ideas to infer heading information is adding a post-processing step, after $\mathcal{G}_D$ is generated without heading direction information. Although such post-processing cannot be used in the process of generating $\mathcal{G}_D$, it can be beneficial in the matching process, when the $\mathcal{G}_D$ is matched to $\mathcal{G}_T$.

To estimate the heading directions after $\mathcal{G}_D$ is generated, we investigated the possibility of realizing $\mathcal{G}_D$ as a rigid graph with the given connectivity and relative distance

values represented as the edge weights of the graph. This problem is known as *graph realization problem*: given a weighted graph $\mathcal{G} = (V, E, w)$, a graph is called *realizable* in $\mathcal{R}^k$ if and only if

$$\exists\ p_1, p_2, \ldots, p_{|V|}, \quad \text{such that } \|p_i - p_j\|_k = w_{ij}, \quad \forall (i,j) \in E.$$

The graph realization problem is to find such realization or to determine that no set of such points exists. Although polynomial time algorithms exist for several classes of graphs to answer this problem, such as [8], it is shown that the problem is NP-hard for some cases, e.g., the realization problem for minimum spanning trees with Euclidean distance in $\mathcal{R}^2$ [18]. Later, Saxe has proved that the realization of an (incomplete) edge-weighted graph into Euclidean spaces is strongly NP-hard [68].

A relaxed version of the problem that can be more useful in our case, is the multi-dimensional scaling (MDS) problem. The MDS problem aims to realize the objects of a dataset in an N-dimensional space, such that a given distance matrix Δ between the objects is satisfied as well as possible. The problem has several variations, such as classical MDS, metric MDS, generalized MDS, etc. In general, this problem can be formulated as an optimization problem as follows: for a set of data-points $V = \{v_1, v_2, \ldots, v_n\}$, define the distance matrix

$$\Delta \triangleq \begin{pmatrix} \delta_{1,1} & \delta_{1,2} & \cdots & \delta_{1,I} \\ \delta_{2,1} & \delta_{2,2} & \cdots & \delta_{2,I} \\ \vdots & \vdots & & \vdots \\ \delta_{I,1} & \delta_{I,2} & \cdots & \delta_{I,I} \end{pmatrix},$$

where $\delta_{i,j} = \|v_i - v_j\|$; find the set of N-dimensional points $\{p_1, p_2, \ldots, p_n\},\ p_i \in \mathcal{R}^N$ such that

$$P = \arg \min_{p_1, p_2, \ldots, p_n} \sum_{i,j} \left| \|p_i - p_j\| - \delta_{i,j} \right|^k,$$

where k is a value indicating the distance norm, for minimization (usually $k = 2$). Note that $\|\cdot\|$ can be any arbitrary metric or distance function. In fact, since we aim to realize the vertices of the data graph $\mathcal{G}_D$ on a grid, the Manhattan distance (L_1 distance) might be a more useful metric than the common Euclidean distance (L_2 distance). One of the solutions of the classical scaling problem analytically obtains the result via spectral decomposition [17, Ch. 2]

MDS is mostly used as a visualization tool for indicating the level of similarity between the objects of a data set. However, it has also been used in other areas, as such localization of wireless networks, in the context of vehicular or wireless sensor networks [53, 20, 74].

As stated in Chapter 1, it has also been used in indoor localization problem [96].

In MDS based localization, the network nodes are considered as graph nodes, and based on the connectivity model, the graph nodes are connected to each other via weighted edges, while the edge weights represent a distance measure between the nodes. Then, the MDS tools are used to find a graph realization in a (usually) 2 or 3 dimensional plane, which leads to finding the location of each node of the graph.

It is important to note that our graph realization problem differs from network localization problem from several aspects. First, the graph nodes in our problem are not users or sensors, but they are labeled points; some hypothetical points that we consider to ease the localization process. As a result, in contrast with wireless networks, we do not seek a distributed approach in which the nodes communicate and need to find their own/their neighbours locations. Our problem is centralized since the information about all nodes of the graph are present in one processing entity. Second, since we are modeling static labeled points on the floor, the mobility assumption is not required, while in mobile sensor networks or vehicular network settings, mobility assumption is important. Finally and most importantly, in our problem the node degrees are usually not more than 4 (in intersections), while in network localization, usually wireless broadcasting is used for localization, so that in each sub-region of the network several nodes are in the broadcast range of each other and therefore, the node degrees are usually very high. In fact, the classical MDS problem assumes that all pairwise distances are available, which translates to having a complete graph.

The mentioned node degree difference is very critical since in order for a graph to be realizable on a 2D plane, its node degrees should have certain conditions. For example, an average degree of 10 or higher is assumed in [53] to achieve 100% localization. Clearly, if a node degree is 1 or 2, it cannot be realized on a 2 or higher dimensional plane (the root locus for a point of certain distance from a certain point is a circle, and the root locus of a point with given distance from two fixed points is one, two, or no points in general). As a result, the nodes with degrees 1 or 2 cannot be realized in general. Furthermore, nodes of higher degrees may also not be uniquely realizable due to "flip and flex ambiguities" as noted in [53] (Figure 2.5). It can be intuitively seen that for a general graph, the possibility of flip and flex decreases as the node degrees increase.

Considering the above differences, it can be concluded that MDS, with the defined settings, cannot be very useful, due to non-uniqueness of obtained results.

Based on our experiments of the mentioned methods, we decided to calculate the distances and directions in $\mathcal{G}_D$ based on the DR module that is explained in Chapter

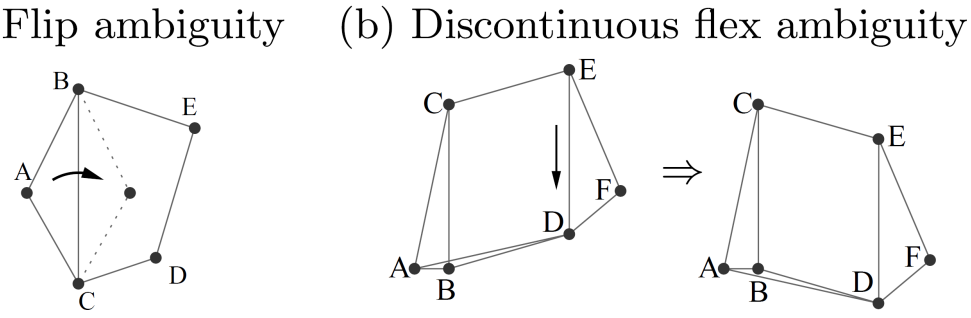


Figure 2.5: Vertex A can be reflected across the line connecting B and C with no change in the distance constraints. (b) Discontinuous flex ambiguity. If edge AD is removed, then reinserted, the graph can flex in the direction of the arrow, taking on a different configuration but exactly preserving all distance constraints [53].

4. The employed module uses an accelerometer based step counter to find the relative distance and, if available, the combination of gyroscope and accelerometer to find the relative heading direction. However, note that any method that results in a topological map (a.k.a. pathway map) with distance (and potentially heading direction) information, such as the works mentioned earlier [39, 78, 96, 64], can be used to generate the data graph.

Once we merge all of the walks and find the relative distances, the components of the resulting graph $\mathcal{G}_D$, will be matched against the components of $\mathcal{G}_T$ using their topological similarities. Note that we can topologically localize ourselves everywhere on $\mathcal{G}_D$. Hence, if we can map the nodes of $\mathcal{G}_D$ to correct nodes in $\mathcal{G}_T$, the location on the floor map can be found geologically.

Regarding the graph model for a building floor, it is also important to mention that since the building floor is practically a continuous area and we are modeling it by a discrete graph, the user may not exactly walk on the graph edges to reach his destination. Also note that we are finding the relative distances between two points via inertial sensors. Hence, there is a chance of estimating incorrect edge distances between two neighbour points in the graph, if a user walks via different paths from one point to another. For example, as a part of $\mathcal{G}_T$ consider a discrete grid in a room (Figure 2.6). One can move from point A to B without passing from C while one may do. Taking a straight path from A to B will cause the graph $\mathcal{G}_D$ to have an edge from A to B with weight $\approx d\sqrt{2}$, while such an edge does not exist in $\mathcal{G}_T$. Besides, since the straight walk from A to B is not defined on $\mathcal{G}_T$, the distance from A to B on $\mathcal{G}_T$ is considered to be the path going through C resulting $2d$. These differences between the structure and weights of $\mathcal{G}_T$ and $\mathcal{G}_D$, introduce noise to the output, which makes the matching task extremely difficult.

Therefore, for this research, we only consider corridors in which non-horizontal and non-vertical movements are less likely to happen. The small rooms can also be represented by single leaf nodes that are walkable from corridors.

As an extension point for this work, the matching results for large corridors can be made more accurate if two graphs $\mathcal{G}_T$ and $\mathcal{G}_D$ are structured more similarly. Our proposed approaches for dealing with the noise in generation of the data graph are the following: noting that if the accurate absolute heading direction using compass was available, the structural noise could be properly eliminated. Therefore, the advances in the MEMS technology as well as developing a proper method to detect the accurate absolute heading direction can be used to detect diagonal movements, versus vertical/horizontal movements. One of our other suggested approaches is to consider the grid points to have 8 connections to their neighbours (adding edges for diagonal movements). Such modifications to the model can approximate most of the possible movements, while adding manageable complexity to the system. Another approach to deal with the mentioned problem is to represent the distance between two nodes via a probability distribution, rather than a deterministic number. This kind of presentation makes the distance calculation more robust to noise and also can handle the problem of multiple paths from A to B up to some extent.

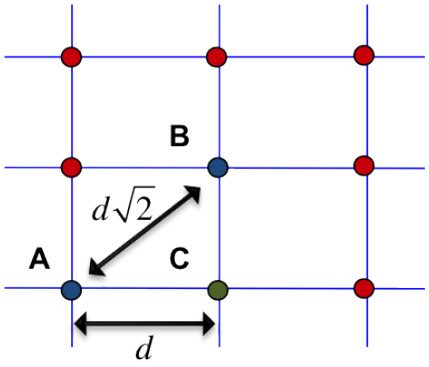


Figure 2.6: Differences between the distance of two points in $\mathcal{G}_T$ and the estimated distance in $\mathcal{G}_D$

2.4 Building the radio map

After generating $\mathcal{G}_D$ and $\mathcal{G}_T$, we match the two graphs to obtain a correspondence between the physical locations and the RSS readings, i.e., the radio map. In the rest of this chapter, we address the matching problem: given two graphs $\mathcal{G}_T$ and $\mathcal{G}_D$, how to find a matching (or partial matching) between the nodes of the two graphs. We have proposed

several approaches to perform such a task. At this stage, the matching will take place as a one time and offline task, after the two graphs $\mathcal{G}_D$ and $\mathcal{G}_T$ are generated. Iterative and online generation of the radio map using this technique can also be applied with minor modifications.

There are various problems formulated in to comparing two graphs, or (two sets of nodes with neighbourhood information) for different purposes. The approaches for each application are also numerous. For instance, in the context of image processing, shape matching, and computer vision, many works such as [4, 84, 75, 13, 55], have proposed different robust shape matching algorithms. Most of the works in those areas are designed for a set of features of an image, such as a set of points that form the edge of a shape, its skeleton, or a mesh network. These methods match the corresponding points of two sets by modeling one set as a non-linear transformation of the other point set. This approach works well in such areas since the two images being compared are usually two shots of the same 3D object from different angles or distances. However, in our context, since the heading direction information may not be available due to unavailability of gyroscope sensor, the two graphs are generally shapeless. In addition, the two graphs that we compare do not have many features to be treated as images and more importantly, are generated through two different processes. Hence, modeling the correspondence as a non-linear transformation function is not relevant, and for the same set of reasons, computer vision-based techniques such as template matching that do rigid point-set matching are also not generally applicable in this setting.

The matching problem is also addressed in the contexts of search engines [5], DNA pattern matching [29], and many more in last three decades [15].

In our work although the main focus is on the specific type of graphs that will be built, namely $\mathcal{G}_T$ and $\mathcal{G}_D$, we have tried to keep the generality of the matching algorithms, so that they can be applied to other matching problems, with minor modifications. We first find a proper similarity/dissimilarity measure between the components of two graphs that can capture the topological structure of the graph for matching. Then we look for proper matching algorithms to minimize the total dissimilarity by assigning the nodes/edges of the two graphs properly. We also investigate other matching approaches that do not require an explicit similarity measure to find the correspondence between the nodes of two graphs.

Along with investigating several existing matching algorithms, we explain our two proposed graph matching algorithms. We first design our techniques to work without heading direction information. Subsequently, we will show that with some modifications, the heading direction information can be added to our proposed algorithms to improve

accuracy.

2.5 Similarity measures

A common notion that runs through many problems modeled by graph theory is the notion of similarity. The similarity between two graphs is defined differently in each application. The definition can address a variety of topological characteristics of a graph. One might be interested in finding out whether two graphs are identical copies of each other (the graph isomorphism problem [99, 24]) or whether a subgraph of one is identical to a subgraph of the other (such as maximum common subgraph isomorphism [9]). Sometimes a graph has changed due to some unknown operations and we are interested in finding out what operations should be done on a graph to revert it back to its original shape (such as error correcting graph matching).

A large number of graph matching tools have high computational complexity. For example the complexity of determining whether two graphs are isomorphic is still undecided [98]. Hence, by increasing the size of graphs, expensive tools with high computational complexity become rapidly inapplicable. In such cases, usually approximation methods are proposed, in which either a relaxation-optimization algorithm is performed, or certain topological measures of the graph, such as degree distribution, vertex centrality, or graph diameter, are the only considered property for measuring the similarity of two graphs. To name a few, node centrality measure [25], coupled node-edge scoring [98] and RASCAL (RApid Similarity CALculation) [66] all have defined similarity measures to compare the nodes of two graphs for different purposes.

In our case, the graph sizes can be moderately large (100-10000 nodes). As a result, we investigate topological similarity measures that suits our matching problem. The structure of the graphs of our interest are very specific, since passable areas in most of the buildings are similar: they usually have no nodes with degrees higher than 4 (intersections), they are planar, and have a single connected component.

In the following, we first discuss the similarity measures we investigated for our matching problem. We first discuss a brief overview of two existing measures in the literature and then we present our two proposed similarity measures (Sections 2.5.3 and 2.5.3). Note that in some cases, instead of defining the similarity, we define the dissimilarity of two nodes, since both definitions can be interchangeably used in the matching process by finding the minimum possible dissimilarity rather than maximum possible similarity.

2.5.1 A measure of similarity between graph vertices by Blondel, et. al.

The similarity approach suggested in [5] is an iterative approach that obtains the similarity of all pairs of nodes for two generally directed graphs G_1, G_2 with arbitrary number of vertices, and connected components. According to [98], this method is a generalization of an influential work by Kleinberg [41], which proposes a graph-based web searching iterative approach.

Adopting the similarity measure in [5], for a node v_1 in the undirected graph G_1 and a node v_2 in the undirected graph G_2 the similarity at iteration k, $\mathrm{sim}_B^k(v_1, v_2)$ is defined as follows:

$$\mathrm{sim}_B^k(v_1, v_2) = \sum_{r:(r,v_1)\in E_{G_1}, s:(s,v_2)\in E_{G_2}} \mathrm{sim}_B^{k-1}(r, s). \tag{2.1}$$

Each iteration is followed by a normalization. The resulting measure defines the similarity between two nodes, as the similarity between the neighbours of one node and the neighbours of the other node. Since the similarity of each two nodes rely on their neighbours, this measure compares the two nodes mostly based on the relative position of each node in the graph. If we define a similarity matrix $\mathrm{SIM}_B^k(G_1, G_2) = [\mathrm{sim}_B^k(v_1, v_2)]_{v_1\in V_{G_1}, v_2\in V_{G_2}}$, then (2.1) can be written in matrix form

$$\mathrm{SIM}_B^k(G_1, G_2) = A(G_1) \times \mathrm{SIM}_B^{k-1}(G_1, G_2) \times A(G_2)^t, \tag{2.2}$$

where $A(G_1)$ denotes the adjacency matrix for the graph G_1 and the superscript t represents matrix transposition.

Note that in general, the initial value for the similarity matrix SIM is very important. It is shown in [98, 5] that under certain conditions, the mentioned iterative method converges to an explicitly computable value.

2.5.2 A measure of similarity between graph vertices by Heyman, et. al.

As a related similarity model to the one in [5], Heyman et. al. have independently suggested an interesting measure. In [29], they compare a set of unweighted directed graphs representing enzyme interactions to determine the evolutionary relationship between the species. Adopting the method used in [29] for finding the similarity between two nodes

of two undirected and unweighted graphs, we have:

$$\begin{aligned} \mathrm{SIM}_H^k(G_1, G_2) &= A(G_1) \times \mathrm{SIM}_H^{k-1}(G_1, G_2) \times A(G_2)^T \\ &+ (\mathbf{1} - A(G_1)) \times \mathrm{SIM}_H^{k-1}(G_1, G_2) \times (\mathbf{1} - A(G_2)^T) \\ &- A(G_1) \times \mathrm{SIM}_H^{k-1}(G_1, G_2) \times (\mathbf{1} - A(G_2)^T) \\ &- (\mathbf{1} - A(G_1)) \times \mathrm{SIM}_H^{k-1}(G_1, G_2) \times A(G_2)^T, \end{aligned} \tag{2.3}$$

where $\mathbf{1}$ is a matrix of all ones, with proper dimensions. The justification for such similarity update is that each pair of nodes receives similarity scores both if the two nodes in the pair are connected to similar nodes or are disconnected from similar nodes, and lose score if one is connected, and the other is disconnected. Note that this measure only works for the case of unweighted graphs, so that the term $(\mathbf{1} - A(G_1))$ represents the unconnected nodes. In order to use this similarity update measure, we define all of the weights to be 1.

We experimented with this measure on our graph matching algorithms and due to the fact that graph edge weights could not be effectively used, the results were not satisfactory in our setting.

2.5.3 Dissimilarity based on sum of all shortest paths

For each node of a graph $\mathcal{G} = (V, E)$, we define the sum of all shortest paths to be

$$\mathrm{sum}_G(v) \triangleq \sum_{i \in V} \text{shortest path}(v, i).$$

In order to compare two nodes within a graph or two nodes in two graphs, the dissimilarity can be defined as:

$$\mathrm{dissim}_{\mathrm{sum}}(v, w) = \|\mathrm{sum}_{G_1}(v) - \mathrm{sum}_{G_2}(w)\|_p, \quad p \geq 1$$

where the operator $\|\cdot\|_p$ is defined to be L_p norm of a vector, and is defined to be

$$\|\mathbf{x}\|_p = \left(\sum_{x_i \in \mathbf{x}} |x_i|^p \right)^{1/p}.$$

The above measure is undefined if the graph is not connected. In our work, it is always assumed that the graph is connected.

As can be seen, such dissimilarity measure is very simple to calculate. The compu-

tational complexity of calculation, further than finding the all pairs shortest path for the two graphs, is $\mathcal{O}(n_1 + n_2)$, where n_1, n_2 are the number of nodes in the two graphs. On the other hand, this dissimilarity measure, is too general and does not represent the graph very well. Experimentally, it was observed that, $\text{sim}(v, w) = 0$ can easily happen even when v and w are totally different nodes.

2.5.4 Dissimilarity based on all pairs shortest paths

A more distinct measure for representing nodes in a graph $\mathcal{G} = (V, E)$ is considering the vector of all shortest paths:

$$\begin{aligned} \text{SP}_G(v) \in \mathcal{R}^n &\triangleq (x_1, x_2, \ldots, x_n), \\ x_i &= \text{shortest path}(v, i), \quad i \in V, \end{aligned}$$

where $n = |V|$ is the cardinality of V. Such measure represents each node of the graph with a vector of length $|V|$.

The idea is that each point in the graph is expected to have a unique representative vector. Therefore, for similar graphs, the corresponding nodes should have the same (or very similar) vectors. Note that the order of the elements of $\text{SP}_G(v)$ is not necessarily the same in two graphs. Therefore, the dissimilarity (distance) between two nodes v, w of two graphs is defined to be the difference between the permutation of the elements of v and w that minimizes the L_p distance between them, for a given $p \geq 1$:

$$\text{dissim}_{\text{SP}}(v, w) = \min_{\pi_w \in \prod(\text{SP}_{G_2}(w))} \|\text{SP}_{G_1}(v) - \pi_w\|_p, \tag{2.4}$$

where $\prod(\text{SP}_{G_2}(w))$ is the set of all permutations of vector $\text{SP}_{G_2}(w)$.

As an example, for a given p, if $\text{SP}_{G_1}(v) = (1, 2, 3, 4)$ and $\text{SP}_{G_2}(w) = (3, 2, 1, 5)$, then according to the definition, $\text{dissim}_{\text{SP}}(v, w) = \|1\|_p = 1$.

For the cases where the size of two vectors $\mathbf{x}$ and $\mathbf{y}$ is not the same, the dissimilarity is defined for a subset of $\mathbf{x}$ and $\mathbf{y}$ of size $\min(|\mathbf{x}|, |\mathbf{y}|)$ that minimizes the sum given in (2.4). For example, if $\text{SP}_{G_1}(v) = (2, 1, 3, 4)$ and $\text{SP}_{G_2}(w) = (4, 1, 2)$, the resulting dissimilarity for a given p will be $\text{sim}_{\text{SP}}(v, w) = \|(1, 2, 3) - (1, 2, 4)\|_p = \|1\|_p = 1$.

Note that (2.4) refers to a dissimilarity measure. However, for ease of explanation, we refer to the values as *similarities* by misusing the word.

In order to efficiently calculate the similarity based on all pairs shortest path, we provide Theorem 2.1, which is proven using the following lemma:

Lemma 2.1. *For all $p \geq 1$ and $a_1, a_2, b_1, b_2 \in \mathcal{R}$*

$$|a_1 - b_2|^p + |a_2 - b_1|^p \geq |a_1 - b_1|^p + |a_2 - b_2|^p, \tag{2.5}$$

where $a_1 \leq a_2$ and $b_1 \leq b_2$.

Proof. The proof is obtained by using the properties of submodular functions. A function $f : X \to \mathcal{R}$, where $X \subset \mathcal{R}^n$ is submodular if for any $\mathbf{x}, \mathbf{x}' \in X$,

$$f(\mathbf{x}) + f(\mathbf{x}') \geq f(\mathbf{x} \uparrow \mathbf{x}') + f(\mathbf{x} \downarrow \mathbf{x}'), \tag{2.6}$$

whenever $f(\mathbf{x} \uparrow \mathbf{x}'), f(\mathbf{x} \downarrow \mathbf{x}') \in X$, where $\uparrow$ and $\downarrow$ are the element-wise maximum and minimum operators on $\mathbf{x}, \mathbf{x}'$. According to Theorem 2.3.4, part (c) of [79], it is straightforward to see that $f(\mathbf{a}, \mathbf{b}) \triangleq \sum_{i=1}^{2} |a_i - b_i|^p$ is submodular for $p \geq 1$. Letting $\mathbf{a} = (a_1, a_2), \mathbf{b} = (b_1, b_2), \mathbf{a}' = (a_2, a_1), \mathbf{b}' = (b_2, b_1)$, we have:

$$\begin{aligned} & f(\mathbf{a}, \mathbf{b}') + f(\mathbf{a}', \mathbf{b}) \geq f(\mathbf{a} \uparrow \mathbf{a}', \mathbf{b}' \uparrow \mathbf{b}) + f(\mathbf{a} \downarrow \mathbf{a}', \mathbf{b}' \downarrow \mathbf{b}) \\ \Rightarrow\ & 2\Big[|a_1 - b_2|^p + |a_2 - b_1|^p\Big] \geq 2\Big[|a_1 - b_1|^p + |a_2 - b_2|^p\Big] \\ \Rightarrow\ & |a_1 - b_2|^p + |a_2 - b_1|^p \geq |a_1 - b_1|^p + |a_2 - b_2|^p, \end{aligned}$$

proving the lemma. □

Theorem 2.1. *For two vectors $\mathbf{x},\ \mathbf{y}$ with lengths n and $p \geq 1$*

$$\min_{\pi_y \in \prod(\mathbf{y})} \|\mathbf{x} - \pi_y\|_p = \|sort(\mathbf{x}) - sort(\mathbf{y})\|_p, \tag{2.7}$$

where sort(·), denotes a permutation of the elements of a vector, that is sorted in increasing order.

Proof of Theorem 2.1. The minimization problem can be considered as finding an assignment between the elements of $\mathbf{x}$ to $\mathbf{y}$ so that the p-norm distance is minimized. Such assignment is a one-on-one mapping, shown by $M(x_i) = y_j$, where x_i, y_j are the elements of vectors $\mathbf{x}, \mathbf{y}$ respectively. Therefore, the problem can be equivalently formulated as seeking an assignment M that minimizes the distance d, where $d(\mathbf{x}, M(\mathbf{x})) \triangleq \|\mathbf{x} - (M(x_1), M(x_2), \ldots, M(x_n))\|_p$.

In order to prove the theorem, we show that the optimal assignment cannot have a "tie", according to our definition. An assignment has a *tie* if there exist two pair of elements x_i, x_j and $y_{i'}, y_{j'}$, with $x_i < x_j$ and $y_{i'} < y_{j'}$, such that $M(x_i) = y_{j'}$ and $M(x_j) = y_{i'}$ (Figure 2.7). For each two pairs of elements that have a tie, we define the

untying process as swapping the assignment of two tied pairs so that $M(x_i) = y_{i'}$ and $M(x_j) = y_{j'}$. Now, we prove that *untying a tie* will not increase $d(\mathbf{x}, M(\mathbf{x}))$ for arbitrary assignment M.

Before untying is done, the total distance can be written as:

$$d(\mathbf{x}, M_1(\mathbf{x})) = \Bigg[\sum_{k=1,\neq i,j}^{n} |x_k - M(x_k)|^p + |x_i - y_{j'}|^p + |x_j - y_{i'}|^p \Bigg]^{1/p}.$$

After the untying, the total distance is equal to:

$$d(\mathbf{x}, M_2(\mathbf{x})) = \Bigg[\sum_{k=1,k\neq i,j}^{n} |x_k - M(x_k)|^p + |x_i - y_{i'}|^p + |x_j - y_{j'}|^p \Bigg]^{1/p}.$$

Now we have:

$$\begin{aligned} & d(\mathbf{x}, M_1(\mathbf{x}))^p - d(\mathbf{x}, M_2(\mathbf{x}))^p \\ = \; & |x_i - y_j|^p + |x_j - y_i|^p - |x_i - y_i|^p - |x_j - y_j|^p \geq 0 \\ \Rightarrow \; & d(\mathbf{x}, M_1(\mathbf{x}))^p \geq d(\mathbf{x}, M_2(\mathbf{x}))^p \\ \Rightarrow \; & d(\mathbf{x}, M_1(\mathbf{x})) \geq d(\mathbf{x}, M_2(\mathbf{x})), \end{aligned}$$

where the 2nd line's inequality is obtained from Lemma 2.1, and the last inequality holds since both sides of the inequality are non-negative numbers.

As a result, untying a tie will not increase the distance $d(\mathbf{x}, M_1(\mathbf{x}))$, which immediately results that the assignment with no ties will have the minimum distance, since any changes in that assignment will lead to having ties, i.e., an equal or greater distance. Note that there only exists one assignment with no ties and that is assigning the elements of $\mathbf{y}$ to elements of $\mathbf{x}$ in order of values, so that $M(x_i) = y_i, i = 1, 2, \ldots, n$, where $x_1 < x_2 < \ldots < x_n$, and $y_1 < y_2 < \ldots < y_n$. Figure 2.7, schematically shows the optimal assignment and a tie in an assignment. Given the optimal assignment, it can be seen that the minimum distance is equal to $\|\text{sort}(\mathbf{x}) - \text{sort}(\mathbf{y})\|_p$. □

Using Theorem 2.1, the computational complexity for calculating the dissimilarity between two nodes via this measure, after finding the shortest path length to all pairs of nodes in the graph, requires sorting the two shortest path vectors, which is $\mathcal{O}(|V|log|V|)$.

For the cases that $|\mathbf{x}| < |\mathbf{y}|$ (or vice versa), we can still use Lemma 2.1 to show that a necessary condition for having an assignment from elements of $\mathbf{x}$ to a subset of $\mathbf{y}$ with minimum distance, is no ties in the assignment. As a result, for such cases, a search algorithm, can be applied to the sorted elements of $\mathbf{x}$ and $\mathbf{y}$ to find the optimal subset

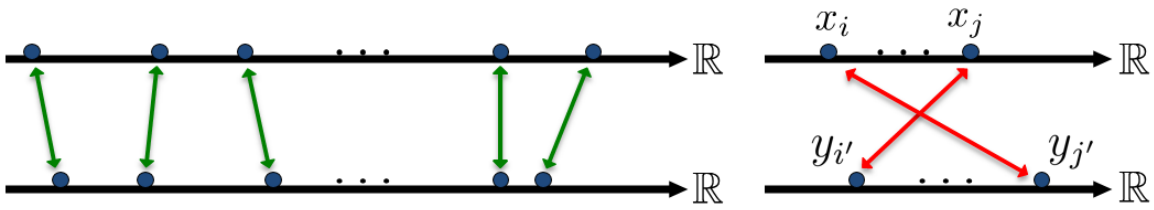


Figure 2.7: On the left; the optimal assignment of elements of two vectors with the same number of elements. No tie exists. On the right; a tie in an assignment. The black line is the axis of real numbers.

of $\mathbf{y}$ and assign them to elements of $\mathbf{x}$, one by one and in order.

Assuming $|\mathbf{x}| < |\mathbf{y}|$, our suggested algorithm is the following:

1. Sort $\mathbf{x}$ and $\mathbf{y}$ and initialize $i \leftarrow 1$.
2. If $i < |\mathbf{x}|$, for the i-th element of $\mathbf{x}$, i.e., x_i, find the best match in $\mathbf{y}$; otherwise, quit.
3. If mapping x_i to its best match doesn't cause a tie (x_i's best match in $\mathbf{y}$ is free), map x_i to its best match. Then increment i and go to 2. Otherwise:
4. Set the *total distance* $\leftarrow \infty$.
5. Temporarily map x_i to the smallest element in $\mathbf{y}$ that does not cause a tie.
6. Calculate the *new total distance* up to here. If equal to or less than the previous *total distance*, keep the current mapping as the best mapping. Otherwise, increment i and go to 2.
7. Map x_i to the element on the left of its current mapped element (this causes a tie in the mapping).
8. For all elements x_j, $j < i$, shift the mapping to left (smaller elements) until no more tie exists. If shift to left is no more possible, increment i and go to 6.

Algorithm 2.1 describes an implementation of the above procedure.

Note that when calling the *shiftleft* function for resolving conflicts (ties), not all of the mappings will be necessarily shifted; only the ones that are conflicting must be shifted. For example, if we have $M = (2, 4, 5)$ (meaning that x_1, x_2 and x_3 are mapped to y_2, y_4 and y_5 respectively), in order to shift the elements to left, since there is a gap between 2 and 4, the shifted mapping becomes $M = (2, 3, 4)$, leaving element 2 unaltered.

To calculate the complexity of the algorithm, note that the most time-consuming part of the algorithm is doing the left-shifts and recalculating the total distance. Since each element is only left-shifted until its place in the final mapping is found, the whole assignment only consists of at most $|\mathbf{y}|$ left-shifts. To be more exact, for each node, we may do one extra left-shift to find out that more shifts will not decrease the distance and break the loop at Line (22). Therefore, the whole assignment requires at most $2|\mathbf{y}| = \mathcal{O}(|\mathbf{y}|)$ left-shifts. Since each left-shift requires a calculation time of at most $|\mathbf{x}|$ points, the computational complexity of the Algorithm is $\mathcal{O}(|\mathbf{x}||\mathbf{y}|)$. The following theorem, proves the correctness of Algorithm 2.1.

Theorem 2.2. *The pseudo code given in Algorithm 2.1 optimally assigns each element of* $\mathbf{x}$ *to an element of* $\mathbf{y}$ *with minimum total* L_p *distance, given* $|\mathbf{x}| \leq |\mathbf{y}|$ *and* $p \geq 1$.

Proof. We first note that the optimal mapping, does not have any ties due to Theorem 2.1. Hence, we first sort the elements of two vectors. We also note that the algorithm initially seeks the best individual match, called $inx \in \mathbf{y}$ for each element $x_i \in \mathbf{x}$ and then either assigns x_i to inx, or an element with a greater index.

Using induction (with an obvious base case), assume for the inductive step that the optimal matching for a vector $\mathbf{x}_1^{k-1} \triangleq (x_1, x_2, \ldots, x_{k-1})$ is found.

The optimal matching for $\mathbf{x}_1^k$ should be obtained by a certain number of *shiftleft* operations in which, the node x_k is trying to get closer to its optimal individual match inx to decrease the total cost. However, each *shiftleft*, also moves the first k elements of $\mathbf{x}$ away from their optimal position in the previous step, resulting in an increase in the total cost.

Since the total cost function $\sum_{j=1}^{k} |x_j - y_j|^p \quad y_j \in \mathbb{R}$ is a convex function on $y_1, y_2, \ldots, y_k$, the function is also convex on a discrete subset of $\mathbb{R}$, namely, the set of the elements of $\mathbf{y}$. Therefore, the minimum distance can be found by calculating the total distance for all *shiftlefts* one by one, until an increase in the total cost is observed or no more shift is possible, as the algorithm suggests. □

We investigated the accuracy of our graph matching techniques, explained in next section, using the proposed similarity measures. I was observed that the similarity measure based on all pairs shortest path drastically outperformed other measures, since it is designed to fit our matching problem.

Algorithm 2.1: Optimal assignment of the elements of two vectors $\mathbf{x}$, $\mathbf{y}$ for minimum total L_p distance, with $|\mathbf{x}| \neq |\mathbf{y}|$

input : Two vectors $\mathbf{x}$ and $\mathbf{y}$ with lengths $n_1 \leq n_2$, norm parameter p
output: The minimum L_p distance between the elements of $\mathbf{x}$ and all possible subsets of $\mathbf{y}$ with size n_1

$\mathbf{x} = \texttt{Sort}(\mathbf{x})$
$\mathbf{y} = \texttt{Sort}(\mathbf{y})$
define the mapping array $M[1 \dots n_1] = [0 \dots 0]$
best $= \infty$
$M[1] = \texttt{argmin}(|\mathbf{x}[1] \times \mathbf{1}_{1\times n_2} - \mathbf{y}|)$
best $= (\mathbf{x}[1] - \mathbf{y}[M[1]])^p$
for $i = 2$ **to** n_1 **do**
    inx $= \texttt{argmin}(|\mathbf{x}[i] \times \mathbf{1}_{1\times n_2} - \mathbf{y}|)$
    if $M[i-1] <$ inx **then**
        $M[i] =$inx
        best $=$ best $+|\mathbf{x}[i] - \mathbf{y}[M[i]]|^p$
    else
        $M[i] = M[i-1] + 1$
        best $=$ best $+|\mathbf{x}[i] - \mathbf{y}[M[i]]|^p$
        /* look for the best mapping by shifting */
        temp_$M = M$
        for $j = 1$ **to** $\min(M[i-1] -$ inx $+ 1, M[i-1] - i + 1)$ **do**
            temp_$M = \texttt{shiftleft}($temp_$M)$
            if $\sum_{k=1}^{i} |\mathbf{x}[k] - \mathbf{y}[$temp_$M\ [k]]|^p <$ best **then**
                best $= \sum_{k=1}^{i} |\mathbf{x}[k] - \mathbf{y}[$temp_$M\ [\mathrm{k}]]|^p$
                $M =$ temp_M
            else
                break
            end
        end
    end
end
best $=$ best $^{1/p}$

2.6 Graph matching

After calculating the similarities of node pairs, a matrix of pairwise node similarities SIM is created to match the two graphs. The problem of matching can be considered as an assignment problem, where each node from $\mathcal{G}_D$ must be assigned to its representative node in $\mathcal{G}_T$ based on SIM. Another formulation of the problem is the maximum weighted bipartite graph matching, mentioned in Section 2.1. Given the set of vertices of two graphs $\mathcal{G}_D, \mathcal{G}_T$, and the similarity matrix SIM, both formulations find a mapping between the vertices of one graph to the other. According to the problem formulation and the used algorithm, the mapping might be one-on-one or many-to-one.

In this section, we first review two well-known matching algorithms: stable matching [26] and Hungarian method [42]. Afterward, we present our proposed algorithm, *K-best fits*, present a pseudo code, and discuss why the proposed algorithm outperforms the two other techniques in the current settings. Finally, we present our proposed hidden Markov model (HMM) based algorithm, that does not use an explicitly defined similarity measure, rather uses a notion of local similarity of nodes and edges to find the corresponding components of nodes. Both of our proposed algorithms are capable of matching graphs generated both with and without heading direction information.

2.6.1 Stable matching problem

The stable matching problem or stable marriage problem, is a well known problem in which, a set of men M and women W want to marry each other. Each man has his own list of preferences of the set of all women. Also, each woman has her own list of preferences of the set of all men. A marriage between m, w is defined to be stable, if two conditions are satisfied:

1. w prefers m to any other man who has proposed to her.

2. m prefers w to any other woman to whom he could be married (i.e., who would have said yes to him if he proposed).

The problem is seeking for an assignment of men and women, where all marriages are stable. The algorithms for solving such problem have variety of applications. For the cases where the set of men and women is not equal and one-to-many mapping is allowed, the problem is also known as the college admission problem. A well known optimal polynomial time algorithm for such problem is given by Gale and Shapley in [26] with worst case running time $\mathcal{O}(V^3)$, where $|m| = \mathcal{O}(V)$ and $|w| = \mathcal{O}(V)$. It is also proved in

[26] that a stable matching always exists. I implementation of the solution to the stable matching problem is given in Algorithm 2.2.

Algorithm 2.2: Stable matching

```
   input: A list of preference for each element of M and W
 1 initialize all m ∈ M and all w ∈ W to free
 2 while ∃ free man m who still has a woman w to propose to do
 3 |   propose to the woman w whom m prefers the most and has not yet proposed to
 4 |   if w is free then
 5 |   |   (m, w) become engaged
 6 |   else
   |   |                                   /* some pair (m', w) already exists */
 7 |   |   if w prefers m to m' then
 8 |   |   |   (m, w) become engaged
 9 |   |   |   m' becomes free
10 |   |   else
11 |   |   |   (m', w) remain engaged
12 |   |   end
13 |   end
14 end
```

For the cases where the mapping is not one-on-one ($|V_{\mathcal{G}_D}| \neq |V_{\mathcal{G}_T}|$), the algorithm can be adapted to be optimal for the first set M or the second set W. We have modified the algorithm to allow one-to-K mappings, where K is the maximum allowed number of assigned members of W to the set M. As a result, for the cases where the density of the nodes of $\mathcal{G}_D$ and $\mathcal{G}_T$ is not the same, the algorithm can still provide meaningful answers. Letting $K = 1$ will reduce the algorithm to the classical version given in Algorithm 2.2.

Stable matching algorithm is optimal in providing an assignment that is stable (a game theoretic equilibrium). However, in the context of matching $\mathcal{G}_D$ to $\mathcal{G}_T$ the stable matching algorithm is not the best choice because it does not consider two facts. First, the neighbourhood information of the nodes within each graph is not considered in the stable matching algorithm. For example, in Figure 2.8, if nodes A and A' are matched to each other, the algorithm matches B to C' if their measured similarity is even slightly greater than the similarity of B to B', even due to noise. However, using the fact that A is already matched to A', it can be inferred that B' is more likely to be a match to B since they are neighbours of an already matched pair. Since stable matching is not designed to consider such side information, it will be missed, unless the information is contained implicitly in the similarity matrix. Therefore, the stable matching algorithm will obtain a good result only if it is fed with a similarity matrix that has optimally extracted all of the neighbourhood information.

The second fact that is missed by stable matching algorithm is that a similarity matrix not only provides a ranking for each node, but also specifies how well two nodes fit to each other and how much a fit is better than another pair. For example, if the similarity of node A and A' is 10, while the similarity of node A and B' is 1, A should be assigned to A'. However, if SIM(A, B')=9.5, assigning A to A' should be less confident. This comparison is not possible if only the ranking of similarities is used.

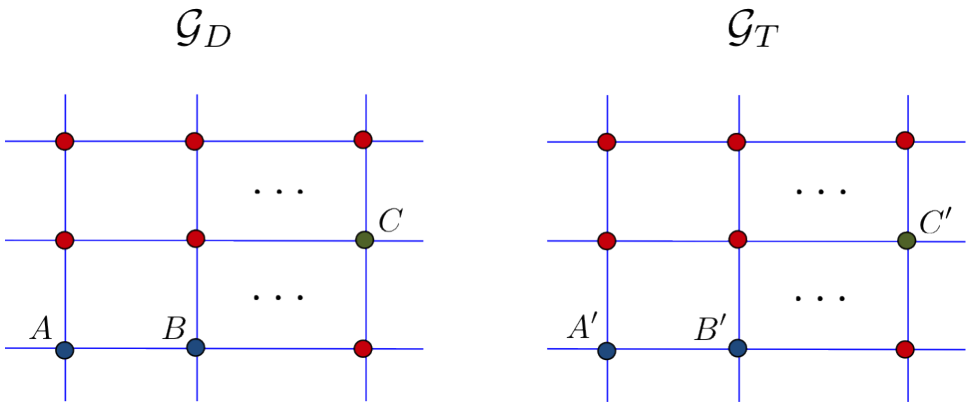


Figure 2.8: If A and A' are matched together, and B has a slightly more similarity score with C' rather than B', it should still be matched to B'.

2.6.2 The Hungrarian assignment

The Hungarian method [42], is a well-known algorithm for maximum weighted bipartite graph matching problem, also known as *the assignment problem*. Given a set of tasks and a set of workers with different costs for doing each task (a cost matrix), this algorithm finds the optimal assignment of tasks to the workers, minimizing the total cost. Therefore, it is optimal in minimizing the total cost (maximizing the total similarity) of the matching. The problem can be mathematically formulated as given the two sets $\mathcal{X}$ and $\mathcal{Y}$ together with the similarity matrix SIM$_{|\mathcal{X}|\times|\mathcal{Y}|}$, finding the mapping $M : \mathcal{X} \to \mathcal{Y}$ that maximizes the following utility function

$$\sum_{\forall x \in \mathcal{X}} SIM(x, M(x)).$$

In its matrix interpretation, the problem can be solved by a sequence of matrix manipulations, non of which will change the result of the matching, until the matching becomes trivial. In the graph interpretation, the problem can be solved via a modified shortest path search in the augmenting path algorithm. A common implementation of the Hungarian method, which is efficient for sparse graphs, has complexity $\mathcal{O}(V^2E)$ [19].

This method has been commonly used in different areas, for matching components of two objects, e.g., in shape matching and object recognition [4].

To use this algorithm, we define our problem as an assignment of the nodes of $\mathcal{G}_T$ to $\mathcal{G}_D$ with a given similarity matrix SIM, and find the assignment with maximum total weight. As the graph model interpretation of the problem, if we can consider the nodes of the graphs $\mathcal{G}_T$, $\mathcal{G}_D$ as the two parts of a bipartite graph, with weights being the similarity between the nodes, our assignment problem can be considered as a weighted bipartite matching problem with the maximum total weight (similarity). Unlike the stable matching algorithm, here the similarity values (rather than the similarity rankings) are used. Therefore, for our problem, it is expected that the Hungarian method outperforms the stable matching algorithm. In Chapter 3, it will be verified that using the similarity values, rather than their ordering, adds to the accuracy as well as noise resiliency of the solution. However, the Hungarian method is more complex than the stable matching and still does not use the neighbourhood information of the nodes.

2.6.3 K-best fits matching algorithm

In order to use the similarity values between the graphs, as well as the neighbourhood information of the nodes within each graph, we propose an algorithm that iteratively matches the nodes and incorporates the neighbourhood information of the matched points, as the assignment of nodes is being completed.

In order to elaborate the contrast between this algorithm and the Hungarian method, the propose heuristic can be considered as an approximation the following optimization problem: given the two graphs $G = (V, E, W)$ and $G' = (V', E', W')$ with together with the similarity matrix between the nodes of the two graphs $\text{SIM}_{|V|\times|V'|}$, finding the mapping $M : V \to V'$ that maximizes the following utility function

$$\sum_{\forall v \in \mathcal{V}} \left[SIM(v, M(v)) - f\left(\Big| \sum_{\forall i, e_{v,i} \in E} w_{v,i} - \sum_{\forall j, e'_{M(v),j} \in E'} w'_{M(v),j} \Big| \right) \right],$$

where $e_{i,j} \in E$ denotes the edge connecting the nodes $i, j \in V$, $w_{i,j} \in W$ represents the weight of $e_{i,j}$, and $f(\cdot)$ denotes the penalty function used for incorporating the graph neighbourhood information.

The algorithm traverses the graph $\mathcal{G}_D$ in a breadth-first fashion (similar to the BFS graph traversal algorithm [90]), starting with a random node. Thereafter, a sequence of visits is made, with each node having a predecessor in the visit sequence. When a node is visited, we select "the best K matches" of that node in $\mathcal{G}_T$ and add them to the K-best

mappings. The final answer is obtained when all nodes of $\mathcal{G}_D$ are visited. According to the implementation, one-to-many mappings may or may not be allowed.

The most important factor about this algorithm is the way "the best K matches" are selected. For each node v, we pick the best K matches not only based on the similarity matrix, but also based on the distance between v and its predecessor $pre(v)$. If direction information is available, we also include the heading direction information in the distance measure. To be more specific, if the distance of v from $pre(v)$ is d, we expect the distance of matches of v to the matches of $pre(v)$ to be also close to d. Therefore, the similarities between v and each node $u \in \mathcal{G}_T$ is decreased with a penalty kernel $f(|D_{\mathcal{G}_D}(v, pre(v)) - D_{\mathcal{G}_T}(u, M_i[pre(v)])|)$, where $D_{\mathcal{G}}(a, b)$ is the shortest path distance between nodes a and b in a graph $\mathcal{G}$ and M_i is the i-th best mapping before visiting v. Since we calculate the K best matches for v based on the K best mappings for π_v, K^2 resulting mappings will be obtained, where only the best K ones will be kept as the mappings to be used for updating the similarities for the next node to be visited.

In our experiments, we used a linear penalty kernel $f(d) \triangleq \alpha d$, where α is a constant factor, we set to 10. Therefore, the similarity between $v \in \mathcal{G}_D$ and $u \in \mathcal{G}_T$ given the i-th mapping, was updated via

$$\text{sim}(v, u) \leftarrow \text{sim}(v, u) - \alpha |D_{\mathcal{G}_D}(v, pre(v)) - D_{\mathcal{G}_T}(u, M_i[pre(v)])|.$$

With similar reasoning, a kernel can be designed to reward candidates for v, if they have the expected relative heading direction, with respect to the 2 previously traversed nodes $pre(v)$ and $pre(pre(v))$. For the case of using heading direction, we multiplied the above distance kernel by a factor $0 < \beta \leq 1$ as a reward, when the heading direction of v was the one expected. Letting $\beta = 1$ is equivalent to not having direction information. For our reported experiments in Chapter 3, we set the constants α and β to the values 10 and 0.6 respectively.

Algorithm 2.3 describes our proposed method in more details. If the algorithm is run N times with N random starting points, the complexity becomes $\mathcal{O}(NK^2V^2)$. A proper value for K was observed to be as low as 2 or 3 and N can be at most V.

2.6.4 Hidden Markov model (HMM) based graph matching

The HMM has been used for graph matching in several other contexts such as image processing, i.e., shape matching [63]. However, due to the differences in the problem settings, we investigate and solve a different problem, using an augmented variation of HMM with continuous emission distributions depending not only on the current state,

Algorithm 2.3: K-best matching

input : similarity matrix $S_{n\times m}$, adjacency matrices $D_{\mathcal{G}_D}, D_{\mathcal{G}_T}$ for graphs $\mathcal{G}_D, \mathcal{G}_T$, all pairs shortest path matrix P for $\mathcal{G}_T$, distance penalty kernel $f()$, and the K value

output: K best found assignments between the rows of S (elements of $V_{\mathcal{G}_D}$) and the columns of S (elements of $V_{\mathcal{G}_T}$

) define K empty mappings $M_{1\dots K}$ of size n
pick a random node v from $\mathcal{G}_D$ and `Enqueue(`v`)`
`/* find the most` K `similar nodes of` $\mathcal{G}_T$ `with` *head* `according to` S `*/`
for $i = 1$ **to** K **do**
    in $M_i[head]$ = the i-th most similar node in $\mathcal{G}_T$ to *head*
end
while Q *is not empty* **do**
    if *head has no free neighbours* **then**
        `Dequeue(`*head*`)`
        continue;
    end
    pick a free neighbour of *head*, v and `Enqueue(`v`)`
    for $i = 1$ **to** K **do**
        $\mathsf{temp} = S[v, 1\dots m] - f(|D_{\mathcal{G}_D}(head, v) - D_{\mathcal{G}_T}(M_i(head), 1\dots m)|) \times \mathbf{1}_{1\times m}$
        for $j = 1$ **to** K **do**
            $\mathsf{fit}[i, j]$ = the total similarity from M_i+ similarity of the j-th most similar node of $\mathcal{G}_T$ to v according to temp
        end
    end
    `/* out of the` K^2 `elements of matrix` fit`, pick the` K `first with highest fits */`
    for $i = 1$ **to** K **do**
        M_i = the mapping with the i-th best total similarity, including mapping of node v, according to fit
    end
end

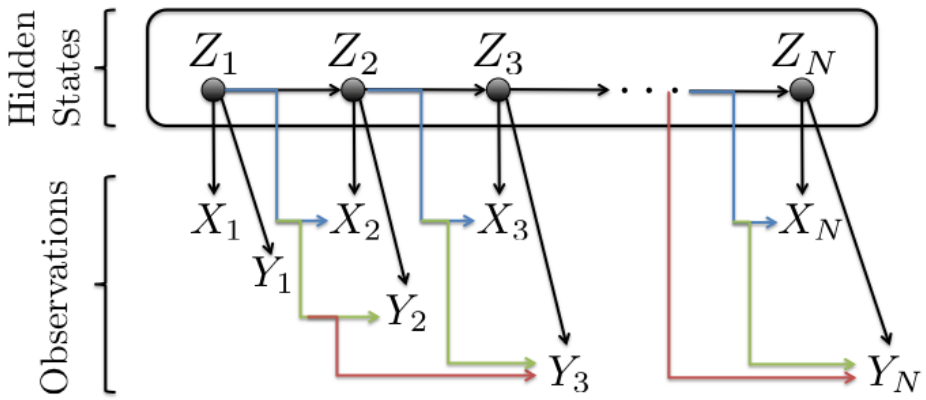


Figure 2.9: The HMM schematic representation. The arrows indicate dependence.

but also on the previous states.

Similar to the previous approach, we generally assume that the heading direction information is not available for the matching. However, we design the algorithm to be able to use such side information, if available. Note that in the presence of heading direction information, $\mathcal{G}_D$ only has the *relative* heading direction information, while $\mathcal{G}_T$ can also have the absolute direction information between the nodes, that can be kept in an incidence matrix, showing which node (if any,) is on top, left, right, or bottom of each node.

To match the graphs, we first generate a random sequence of nodes to mimic a hypothetical user walking on $\mathcal{G}_D$: $\mathbf{S} = \{S_n\}_{n=1}^{N}, S_n \in \{1, 2, \ldots, |\mathcal{U}|\}$, where $|\mathcal{U}|$ is the number of nodes in $\mathcal{G}_D$. The sequence $\mathbf{S}$ is a random traverse on $\mathcal{G}_D$, with a uniform random initial node. Once such sequence is generated, a sequence of edge weights $\mathbf{X} = \{X_n\}_{n=1}^{N}$, representing walked distances in each state transition is observed. A sequence of relative changes in the heading direction (if available,) is also observed, while transitioning between the states. We refer to such sequence as $\mathbf{Y} = \{Y_n\}_{n=1}^{N}$.

The idea of the matching is that the walk (generated sequence) on $\mathcal{G}_D$, can be considered as a walk on $\mathcal{G}_T$ that was subject to an unknown deterministic noise, that caused $\mathcal{G}_T$ to be transformed into $\mathcal{G}_D$. We note that although the labeling orders of the nodes in $\mathcal{G}_T$ and $\mathcal{G}_D$ are not identical, the observed set of edge weights and heading directions is expected to be similar in both graphs, due to their similar topology. We aim to compensate the deterministic noise and find the correct mapping between the nodes of $\mathcal{G}_D$ and $\mathcal{G}_T$ by observing the sequence of walked distances and relative heading directions (if available,) to find the most probable sequence of states on $\mathcal{G}_T$ that could produce the same sequence. Once such sequence is obtained, it can be compared to the actual sequence walked on

$\mathcal{G}_D$, to find a correspondence between the node pairs in the two graphs.

Define the set of nodes in $\mathcal{G}_T$ as the set of possible states, represented by the latent variable $Z \in \{1, 2, \ldots, |\mathcal{V}|\}$, where $|\mathcal{V}|$ is the number of nodes in $\mathcal{G}_T$. The equivalent sequence of states in $\mathcal{G}_T$ is represented by $\mathbf{Z} = \{Z_n\}_{n=1}^N$, with the transition probabilities

$$\Pr(z_n|z_{n-1}) = \begin{cases} 0 & \text{if } A_{\mathcal{G}_T}(z_n, z_{n-1}) = 0 \\ 1/\text{degree}(z_n) & \text{otherwise.} \end{cases}, \tag{2.8}$$

where z_n represents the outcome of Z_n, indicating the latent state at time n, $A_{\mathcal{G}_T}$ is the adjacency matrix for the graph $\mathcal{G}_T$, and degree(z_n) denotes the number of states to which Z_n can transition, with non-zero probability.

The sequence of emissions (walked distances) are represented by $\mathbf{X}$, where unlike the conventional HMMs, X_n has a continuous probability distribution that depends on both the current state Z_n and the previous state:

$$\Pr(x_n|z_n, z_{n-1}) \sim \mathcal{N}(\mu, \sigma), \tag{2.9}$$

where $\mathcal{N}(\mu, \sigma)$ denotes a Gaussian distribution with mean values set to be $\mu = A_{\mathcal{G}_T}(z_n, z_{n-1})$ and standard deviation σ set according to the observations of noise in measuring the distance. For our experiments, we set $\sigma = 0.1\mu$, since the variance of the noise grows as the walked distance grows, due to drift error. Note that in practice, the Gaussian distribution should be truncated for values less than zero, since x_n represents distance.

Since the distribution of emission is a function of the current, the previous state and the adjacency matrix, we have

$$\Pr(X_n|Z_n, Z_{n-1}, A_{\mathcal{G}_T}) \sim \mathcal{N}(\mu_{Z_n,Z_{n-1}}, \sigma_{Z_n,Z_{n-1}}), \tag{2.10}$$

where $\mu_{i,j} = A_{\mathcal{G}_T}(i, j)$.

If the heading direction information is also available, there exists an extra observation, which is the relative changes in the heading direction when transitioning from one state to another. Such observation can be also considered as a second element of the emission sequence $\mathbf{X}$. Here, for simplicity of notations, we define it as a separate sequence, denoted by the random process $\mathbf{Y} = \{Y_n\}_{n=1}^N, Y_n \in \{0, 90, -90, 180\}$. Note that Y_n depends on the current state, as well as the previous two states, such that

$$\Pr(y_n|z_n, z_{n-1}, z_{n-2}) \begin{cases} 1 - \epsilon & \text{if } \angle(z_n, z_{n-1}, z_{n-2}) = y_n \\ \epsilon/3 & \text{other 3 possible cases,} \end{cases} \tag{2.11}$$

where ϵ is the probability of false heading detection and $\angle(a, b, c)$ denotes the angle that appears when moving from a to b and then to c. In practice, we set $\epsilon = 10^{-2}$ for our experiments.

Figure 2.9 shows a schematic diagram of the modeled HMM. Given the transition and emission probabilities, the probability of observing a sequence of emissions $\{\mathbf{X}, \mathbf{Y}\}$, caused by a sequence of occurred states $\{\mathbf{Z}\}$, with model parameters $\Theta = \{A_{\mathcal{G}_T}, \pi, \epsilon\}$, where π is the prior probability of the states, can be written as

$$\begin{aligned} &\Pr(\mathbf{X}, \mathbf{Z}, \mathbf{Y}|\Theta) = \\ &\Pr(z_1|\pi)\left[\prod_{n=2}^{N}\Pr(z_n|z_{n-1})\right]\times \\ &\Pr(X_1|Z_1)\left[\prod_{m=2}^{N}\Pr(X_m|Z_m, Z_{m-1}, A_{\mathcal{G}_T})\right]\times \\ &\Pr(Y_1|Z_1)\times\Pr(Y_2|Z_2, Z_1)\left[\prod_{l=3}^{N}\Pr(Y_l|Z_l, \ldots, Z_{l-2}, \epsilon)\right]. \end{aligned} \tag{2.12}$$

Letting the initial states to be equiprobable and using the Bayes probability rule, the most probable sequence of states on $\mathcal{G}_T$, based on the observed sequence of emissions can be formulated as

$$\begin{aligned} \hat{\mathbf{Z}} &= \arg\max_{\mathbf{Z}}\Pr(\mathbf{Z}|\mathbf{X}, \mathbf{Y}, \Theta) \\ &= \arg\max_{\mathbf{Z}}\Bigg[\sum_{n=2}^{N}\log\Pr(Z_n|Z_{n-1}) \\ &+ \sum_{m=2}^{N}\log\Pr(X_m|Z_m, Z_{m-1}, A_{\mathcal{G}_T}) \\ &+ \sum_{l=3}^{N}\log\Pr(Y_l|Z_l, Z_{l-1}, Z_{l-2}, \epsilon)\Bigg]. \end{aligned} \tag{2.13}$$

If the direction information is unavailable, the last line of (2.13) will be omitted.

Using the formulation given in (2.13), the most probable sequence of states is calculated using the Viterbi algorithm with complexity $\mathcal{O}(|\mathcal{U}||\mathcal{V}|N)$. If the number of the nodes of two graphs is the same, since N is also selected to be an order of the number of nodes (so that all nodes are walked at least once,) the complexity will be $\mathcal{O}(N^3)$.

Once $\hat{\mathbf{Z}}$ is calculated, we define a mapping table $M = [m_{i,j}], i \in \{1, \ldots, |\mathcal{U}|\}, j \in \{1, \ldots, |\mathcal{V}|\}$ by comparing the state sequences $\mathbf{S}$ and $\hat{\mathbf{Z}}$ in the following way:

- initially set $m_{i,j} = 0, \forall i, j$;

- for $k = 1, \ldots, N$ do $m_{S_k, \hat{Z}_k} = m_{S_k, \hat{Z}_k} + 1$.

Note that if the mapping is unitary, after normalizing each row of M, the obtained table should be close to a unitary matrix, as in Figure 2.10, where a sample matching table on simulated graphs is shown. Since $\hat{\mathbf{Z}}$ may have some errors, in order to obtain the final mapping, we apply the stable matching algorithm [26] to map each node of $\mathcal{G}_D$ to its most stable match. If the number of nodes in $\mathcal{G}_T$ and $\mathcal{G}_D$ is not the same, we allow one to many mappings in the algorithm.

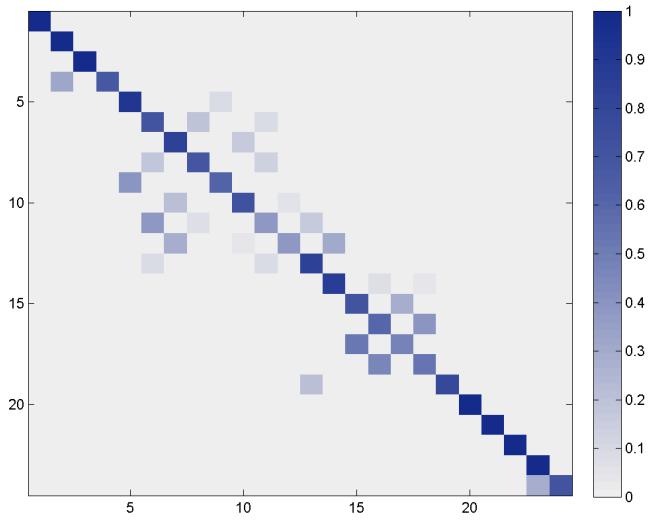


Figure 2.10: Representation of a sample mapping table for 24 nodes and unitary mapping.

After finding a matching between the nodes of $\mathcal{G}_D$ and $\mathcal{G}_T$, all the information for generating the radio map is available. In the next chapter, we will present the details of our implementation of the proposed system, as well as the obtained results of generating graphs and running matching algorithms both in simulation and on real data.

Overall, the described algorithms can be summarized in Table 2.1. Note that although the complexity of the algorithms is generally important, it does not affect the real-time performance of the system, since the matching is done in the offline training phase.

Table 2.1: Summary of the properties of two well known matching algorithms, along with the two proposed algorithms. The stated computational complexities are given for the matching of two graphs with nodes of order $\mathcal{O}(V)$ and edges of order $\mathcal{O}(E)$. Worst case complexity, including similarity calculation is considered.

Algorithm	**Pre-calculated similarity?**	**Complexity**	**Comments**
Stable Matching [26]	Yes, ranking only	$\mathcal{O}(V^3)$	Not using similarity values or neighbourhood info.
Hungarian Method [19]	Yes	$\mathcal{O}(V^2E)$	Not using neighbourhood info.
K-best fits	Yes	$\mathcal{O}(NK^2V^2)$	Faster, local error affects all similarities
HMM based matching	No	$O(V^3)$	Slower, local error has less effect on all similarities

Chapter 3

Implementing the Crowdsourced Localization System

In this chapter, we describe our implementation of the crowdsourcing based localization system, proposed in Chapter 2. In our implementation, the system is designed to work in two phases, i.e., calibration phase and operation phase. In such implementation, most of the work regarding the use of crowdsrouced data, generating the two graphs $\mathcal{G}_T$ and $\mathcal{G}_D$ and matching them happens in the calibration phase. The output of the calibration phase is the radio map which is used in the online phase for localization.

Note that using the same model discussed in Chapter 2, the system can be designed to work in a single phase. In such system, initially no data is available in the data-set. However, gradually the users who are using the system in the single phase of operation, populate the data-set with sensor readings that will be used for improving the accuracy and coverage of the system by obtaining a more and more complete data graph. In such system, the matching of $\mathcal{G}_D$ to $\mathcal{G}_T$ can happen repeatedly over certain periods of time, to further train the system with recently obtained crowdsrouced data.

3.1 Building the ground truth graph $\mathcal{G}_T$

As mentioned earlier, $\mathcal{G}_T$ is simply generated by discretizing the floor plan and defining a graph on the floor, based on passable/impassable areas.

Note that according to this definition, both of the ground truth graph and data graph are defined to have a node every k meters or few taken steps, and the edges are the walked traces that connect the nodes. Therefore, the number of nodes along a hallway in $\mathcal{G}_T$ could vary arbitrarily and is not necessarily equal to the number of nodes on the corresponding hallway in $\mathcal{G}_D$, making $\mathcal{G}_T$ and $\mathcal{G}_D$ generally look different. To deal with

the unequal number of nodes in the matching phase, we filter the nodes of $\mathcal{G}_T$ and $\mathcal{G}_D$ in a way that only the nodes representing turns, intersections, or leaf nodes are considered in the matching. The graph containing only such filtered nodes is denoted as *macro graph*. Note that since detecting sharp turns requires the heading direction information, for the cases that such data is not available, the macro graph will only contain intersection points and leaf nodes on both $\mathcal{G}_T$ and $\mathcal{G}_D$. In such cases, there is a chance of having two edges connecting the same nodes, e.g., when two parallel hallways exist and one is connected to the other via two 90 degree turns (similar to the hallways in Figure 3.1). The macro graph for such cases can be modeled by either allowing multi-graphs (graphs that can have multiple edges between two nodes), or simply discarding the longer one during the filtering phase, keeping the shorter edge between the nodes.

After filtering, the matching would be more robust to the unequal number of nodes since the macro graphs for both $\mathcal{G}_T$ and $\mathcal{G}_D$ are expected to look very similar.

Figure 3.1 shows a sample ground truth graph for a part of a building floor along with its macro graph containing only the representative nodes of the graph. Once the graph is generated, its adjacency matrix, along with the location of all nodes is stored for the matching phase.

3.2 Building the data graph $\mathcal{G}_D$

In order to build $\mathcal{G}_D$, we use crowdsourced data. When each user walks through the pathways of the building, the user's smartphone collects and stores information about the status of the phone and the environment. To estimate the user's relative displacement, we use the DR module explained in Chapter 4 which is capable of counting the steps and detect relative changes in the heading directions. Due to the structure of the hallways in common building areas, we are only interested in major changes in the heading direction, i.e., 90 degree turns.

Using the DR module, the user's walking trace can be recorded with respect to an (unknown) initial point with an initial heading direction. While such trace is being calculated, we also store available RSS samples and assign each RSS sample to its relative displacement with respect to the starting point of the walking trace.

Once the user reaches the destination and stops walking, the data collection is ended. To avoid large drift errors using DR, long traces are broken to smaller pieces, as if several users walked shorter traces. The obtained traces can be considered as a graph with nodes at each k taken steps and edges connecting the consecutive nodes, with k indicating the density of graph nodes. The value k cannot be arbitrarily small as the overlap of traces

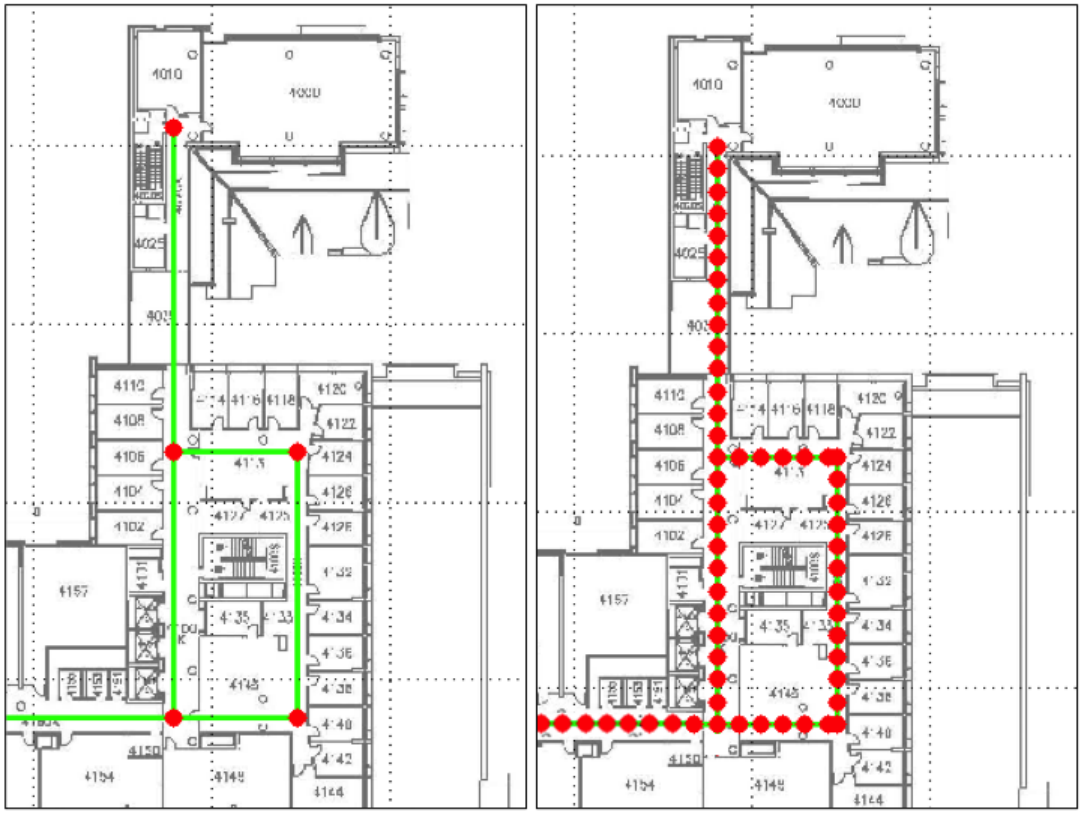


Figure 3.1: A sample ground truth graph (on the right) and its resulting macro graph (on the left). If the direction info. is not available, the vertical two nodes on the right and their edges will be either removed or replaced with a single edge.

will be found by matching the Wi-Fi signals, making very close nodes indistinguishable. Typically, the minimum possible value for k can be 3 to 5 steps. Higher k values lead in a lower density of RSS points and hence, a more coarse localization. From now on, we will use the terms "traces" and "graphs" exchangeably, for ease of explanation.

After collecting a set of traces, we merge them one by one to build a unified data graph. Note the traces do not have any information about the actual map locations in the building, since each trace is stored with respect to a different starting point and initial heading direction. Therefore, to merge two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, we need to transform one to the other's coordinate system. We transform all the node coordinates of G_2 to the coordinate system of G_1. The transformation entails finding a proper origin offset $\mathbf{O} = [\alpha, \beta]^T$, a rotation matrix $\mathbf{R}_{2\times2}$, and scale factor $\mathbf{S} = s \times I_{2\times2}$,

which can be described via the following relation:

$$\begin{aligned} v' &= \mathbf{S} \times \mathbf{R} \times v + \mathbf{O} \\ &= \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} + \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \end{aligned}$$

where θ is the clockwise rotation angle between the coordinate system of the two graphs. In this work, for simplicity and due to observations, we consider the scale factor to be 1. Note that this is not a limiting factor, as the scale factor can be estimated. However, estimating an extra parameter requires a larger minimum overlap between the graphs. With $s = 1$, the only remaining transformation parameters to estimate are θ, α and β.

To find the transformation parameters, we compare the collected RSS values in the nodes of two graphs. Finding the similarity (distance) of two RSS readings is largely investigated in the literature; we adapt a common method, which consists of comparing the L_p norm of RSS differences (with p usually 1 or 2) as in [94, 95, 88]. For each two RSS samples $\mathbf{f}_i, \mathbf{f}_j$, we find the union of visible access-points (AP) and calculate the norm-p distance of the two RSS vectors. For each AP that is only visible in one of the points, we add a penalty to the distance by replacing the invisible AP RSS values to be a very small received power (e.g. -110 dBm which indicates no signal in Wi-Fi standard). The distance is then normalized based on the number of common APs. Since having more common APs indicates a higher chance of having corresponding nodes on the two graphs, the normalized distance is then rewarded (decreased) with a concave function of the count of common APs, since concave functions are more distinctive among small values. In this work, we use the log function for the rewarding scheme. Therefore, the overall distance can be calculated via the following relation:

$$D(\mathbf{f}_i, \mathbf{f}_j) = \frac{||\mathbf{f}_i - \mathbf{f}_j||_p}{n} - \log n,$$

where n is the number of common APs and the invisible APs are set to have the RSS value -110 dBm. Since finding the union of two sets of size N can be calculation with computational complexity of $\mathcal{O}(N)$ (e.g., using hash tables), the complexity of calculating such distance for all pair of nodes is $\mathcal{O}(AV^2)$, where A is the maximum number of visible APs in a single point, and V is the number nodes in each trace.

Once the distance between all RSS sample pairs of G_1 and G_2 is calculated, we use the stable matching algorithm (described in Subsection 2.6.1) to assign the points of G_1 to corresponding points in G_2. If the distance of the two points is less than a threshold, the assigned points from the two graphs are considered to be on the same physical location,

indicating an overlap between the two traces. Given three or more overlapping points, the parameters θ, α and β are calculated. The traces that do not have enough overlap are kept to be compared again later, when more traces are merged.

Note that due to the variations in the RSS readings for each point, in order to successfully identify more than 3 overlapping points, a larger number points should overlap in practice. In other words, since the RSS readings of a single point on two traces may vary due to reading noise, in practice, a larger number of points should have overlap to have a portion of them being recognized successfully.

To estimate the parameters $\alpha, \beta, \text{and} \gamma$, define an ordered set of overlapping points in G_1 as P_1 and their corresponding points in G_2 as P_2. The offset $\mathbf{O} = [\alpha, \beta]^T$ is calculated by finding the minimum mean square error (MSE), as follows:

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \frac{1}{|P_1|} \sum_{i=1}^{|P_1|} \left(P_1(i)_x - P_2(i)_x \right) \\ \frac{1}{|P_1|} \sum_{i=1}^{|P_1|} \left(P_1(i)_y - P_2(i)_y \right) \end{bmatrix}.$$

As an alternative, in order to reduce the effect of noise, estimating α and β can be done via techniques that remove outliers, e.g., robust locally weighted regression [14].

The rotation angle θ is hard limited so that it can only be a multiple of 90 degree turns (for the 4 possible heading directions). It is straightforward to extend the algorithm to include turns at 45 degree angles. Due to the small set of possible rotation angles, θ can be calculated by trial and error over the four possible rotation matrices of 0, 90, 180, and 270 degree rotation, and picking the rotation angle that obtains the minimum MSE after removing the offset.

Once the second graph is transformed to the coordinate system of the first graph, the graphs can be merged. We merge the graphs based on their distance in the physical domain. In other words, each two points from the two graphs are merged if their Euclidean distance is less than a threshold. Once each two points are merged, their location, neighbourhood information, and RSS values are also merged.

In order to merge the location coordinates and RSS values, we use a dynamic weighted averaging method. Let a data entry (a node location or an RSS value) D_W be a weighted average of W previous samples. Define the new data entry being merged ($W+1$-st entry) by d_{new}. The resulting average value for the data entry D_{W+1} is defined to be

$$D_{W+1} = \gamma \left(\frac{W D_W + d_{new}}{W+1} \right) + (1-\gamma) \left(\frac{D_W + d_{new}}{2} \right), \tag{3.1}$$

where $0 \leq \gamma \leq 1$ is a forgetting factor that keeps the data set up-to-date by taking a

weighted sum of an averaging term that uniformly values all the $W + 1$ merged data entries (first term) and an averaging relation term that assigns half of the weight to the most recently merged entry (second term). Note that by using the averaging method in (3.1), at the early steps that the data set graph and the newly found trace are both unconfident, say $W = 1$, both of the points will have a similar weight in the average. As the number of merged points grows, W grows and a larger weight is assigned to the merged data set. On the other hand, because of the γ factor, the old data values will be forgotten in long run. In our experiments, we used the value 0.4 for the γ forgetting factor.

The result of such merge is a single graph, with the information from both of the traces. The obtained graph is then used as the incomplete data graph for the next trace to be merged with. The merging continues until all overlapping graphs are merged. Note that regarding the averaged RSS values, the system can also be extended in a straightforward manner to keep all of the observed RSS values for each point, so that instead of a single average RSS value per point, a distribution of RSS values for each point is kept for the online phase localization.

In order to calculate the computational complexity of the data graph generation for each merge, assuming the graphs have similar number of nodes and nodes have similar number of visible access points, we need to sum the complexity of calculating the RSS distance, $(\mathcal{O}(AV^2))$ with the stable matching algorithm (with worst case complexity $\mathcal{O}(V^3)$). The rest of the operations, including transformation of coordinate system of traces, and merging the data points are dominated by the first two operations. Therefore, for each collected trace, a complexity of order $\mathcal{O}(AV^2+V^3)$ is required to merge the trace to the existing data graph.

The minimum theoretical number of required traces to generate a complete data graph, is a number of traces that covers the whole walkways in the area, while each trace also has at least 3 overlapping Wi-Fi readings with another trace. However, in practice, since some of the trace overlaps may not be detected, more overlap between the traces and more traces are required.

Figure 3.2 shows a set of sample traces collected from a part of a building via smartphone. Figure 3.3 represents the built data graph from that part of the building. The two presentations of the graph denote the two possible cases of having or not having the heading direction information on the graph.

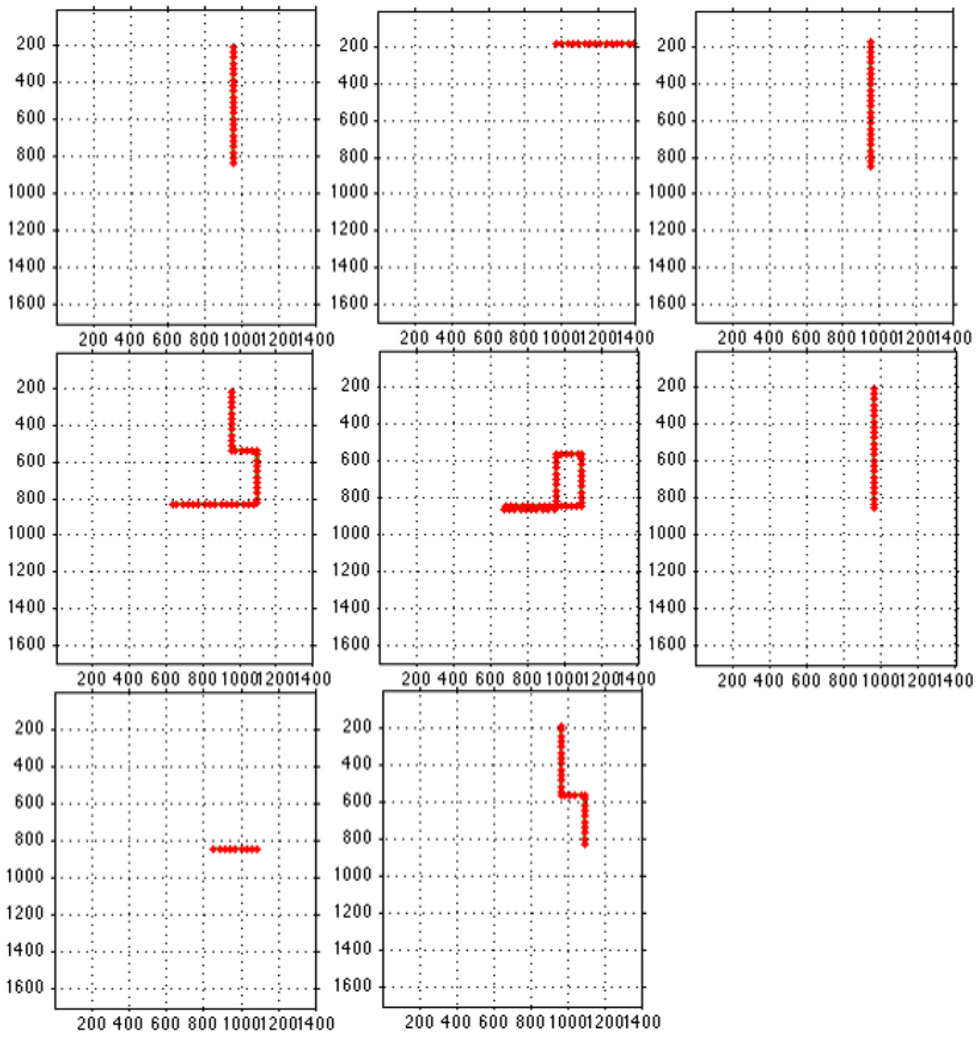


Figure 3.2: A set of sample collected traces.

3.3 Results

In this section, we will demonstrate the results obtained from the implemented crowdsource based system. We first present the results for the data graph generation. Next, we assess the performance of the matching algorithms. Finally, we present an overall performance evaluation of the system, in comparison with the state-of-the-art algorithms.

3.3.1 Unsupervised data graph generation

To evaluate the performance of data graph generation based on pedometer (DR module) and matching Wi-Fi signals, we collected 47 walk sequences on the 4th floor of "Bahen Centre for Information Technology" building (almost $100 \times 100m^2$ with about 305m of

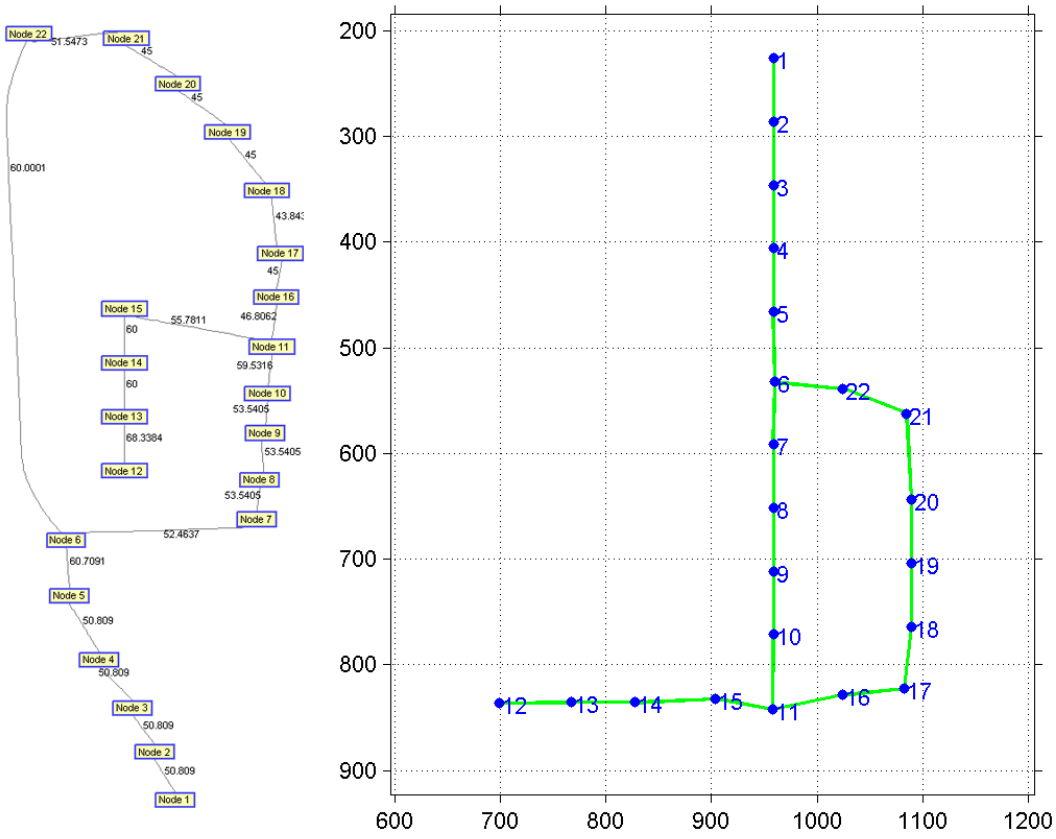


Figure 3.3: A sample data graph with (right side) and without (left side) heading direction information.

hallway), located at the University of Toronto campus (Figure 3.7 background image on the right). The traces where collected by 2 persons (a male and a female) using 3 smartphones. An overall of 6113 steps were collected, along with 911 Wi-Fi scans. In order to collect each trace, the users casually walked in the building, without making unnecessary stops, as if they have a destination room in mind and are walking toward that destination. The phone was not required to be kept in a specific direction and could be kept in several gestures such as in front, swinging in the hang or on the ear.

Since our algorithm is designed to depend on the order of merging graphs (using the weighted averaging scheme mentioned in Section 3.2), we were interested in the dependence of the resulting data graph on the order in which graphs were given to the system. Note that relying on the order of input graphs is generally an intended behavior to keep the generated data graphs up-to-date over time, by relying more on new data and forgetting the old data after a while. However, this behavior may initially make the generated data graph rely too much on the first set of traces. Hence, if the initial traces are too noisy or very short, they may drastically affect the out-coming $\mathcal{G}_D$.

To evaluate such a dependence on the ordering, we generated 300 random orderings of the collected traces and used them to generate data graphs. The resulting graphs could be categorized to 5 general categories, seen in Table 3.1. The graphs in each category were mostly similar to each other. A sample graph from each category is shown in Figure 3.4, along with the target answer.

Table 3.1: Evaluation of generated data graphs using 300 different permutations on input traces. The categorization is in terms of how much of the graph was built and how much topological difference was seen w.r.t. the target graph.

id	Category	%	used walks ratio
0	more than 3 differences	5	0.42
1	1-3 incorrect nodes/edges	13	0.36
2	less than half of the graph was generated	36	0.05
3	more than half, not complete, minor errors	29	0.32
4	complete, minor errors	17	0.42

Of the categories in Table 3.1, the only one that can lead to big errors in the graph matching phase is category 0, where the generated data graph has a big topological difference from the ground truth graph. Since the percentage of these cases is very low (5% in this experiment), they can be detected by running the algorithm several times and picking the majority of the resulting data graphs that have more resemblance. Category 1 is expected to lead to minor errors in the matching.

For Category 2, we investigated the graphs that were merged with others. It was observed that there exist 7 traces in the data set that did not merge to any graphs, once they were selected as the initial graph. The reason was mainly because these traces had a small length and the number of overlapping traces with them was not high enough. Of the 109 cases belonging to this category, 52 started with one of these 7 traces. We refer to these cases as *bad starting points*. In practice, bad starting points can be avoided by selecting larger traces as the initial graphs. A mechanism can also detect and replace bad starting points if many traces are compared to them and nothing gets merged.

Category 3 and 4 (total of 46% of cases) produced acceptable results. We expect that given more traces, the graphs in category 3 (which are a partial graph of the target graph) will also move toward category 4 (the complete graph, with minor errors).

Overall, in all of the categories, an average of 25% of the traces were merged. The average of the merged walks for all categories except category 2 was 36%. Therefore, we expect that if bad starting points are avoided, an average of 3 walks per path can generate a complete radio map.

Aside from the subjective analysis of generated data graphs, we calculated the average

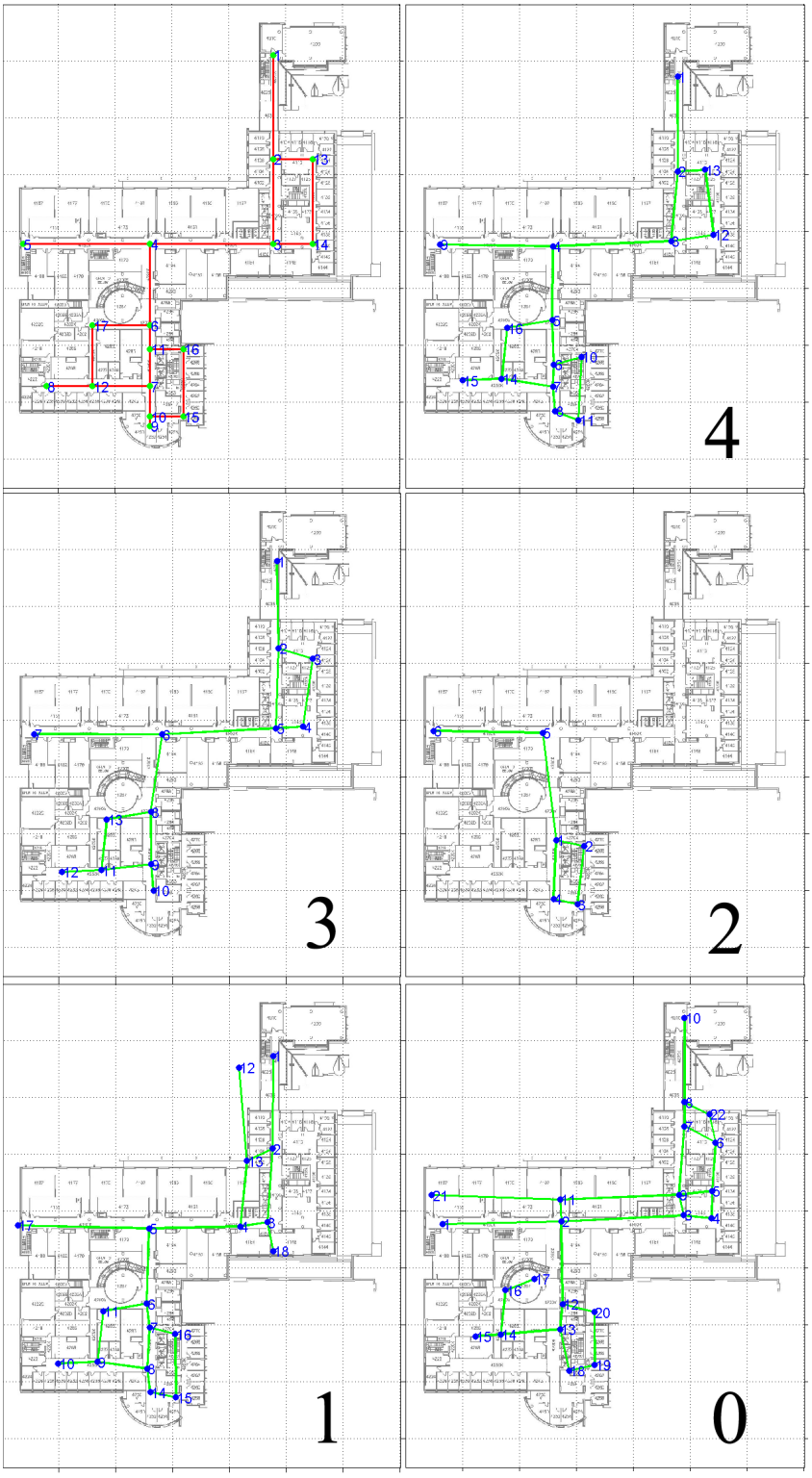


Figure 3.4: The 5 categories of generated data graphs, along with the target ground truth graph (top left). In Category 0 and 1, the extra incorrect nodes going through walls are seen. In Category 2 less than half of the graph is generated. In Category 3, the bottom right portion of the target graph is missing. Category 4 is a complete data graph.

distance of the nodes in $\mathcal{G}_D$ from their corresponding nodes in $\mathcal{G}_T$. We collected the ground truth location of the users during data collection, by marking the initial location of the user and updating their location via dead reckoning. Since the estimated locations were collected using dead reckoning, an average error of about 1 meter was observed in the reference location information. The calculated average distance from the estimated trace locations for complete data graphs (Category 4) was 2.65 meters, which suggests that the generated data graph $\mathcal{G}_D$ and $\mathcal{G}_T$ are very much similar. This distance, which is due to the noise in data collection and generation of $\mathcal{G}_D$ was expected to be partly compensated in the graph matching phase. Note that as mentioned before, the matching phase can also be applied to any semantic pathway map (including the ones in many related works), to reduce their mapping error.

3.3.2 Graph matching, simulations

We first tested the performance of our matching algorithm and similarity measures via simulations, under different conditions.

Sensitivity to noise

In this experiment, $\mathcal{G}_D$ was a noisy version of $\mathcal{G}_T$. In order to validate the performance of our proposed similarity measure and matching algorithm, we performed 1000 simulations on randomly generated graphs. In each round, we generated a random graph with 60 to 100 nodes (20 to 60 nodes after filtering). The ground truth graphs $\mathcal{G}_T$ were generated by simulating a random 2-dimensional (2D) walk on 2D plane. The obtained graphs had certain structural limits, such as having maximum node degree of 4, having a single connected component, and being planar, so that the graph could actually be in correspondence with an imaginary building floor. A noisy copy of the graph was also generated when building $\mathcal{G}_T$ by adding a Gaussian noise to each edge weight. Two sample simulated graphs (after filtering) and their noisy copy are shown in Figure 3.5. Note that we evaluated the system's performance with noise standard deviations up to $\sigma = 0.3$ of step length, which is very pessimistic in comparison with our observations of the user step length variations and the DR module's high accuracy.

In all of the matching (except the HMM-based method that does not need an explicit similarity measure), we used the vector of all pairs shortest path similarity measure, as described in Section, 2.5 due to its superior performance. Table 3.2 shows the results of the simulation for different matching techniques described in Section 2.6. It can be seen that the K-best fits algorithm is more robust to noise than the comparing algorithms. If

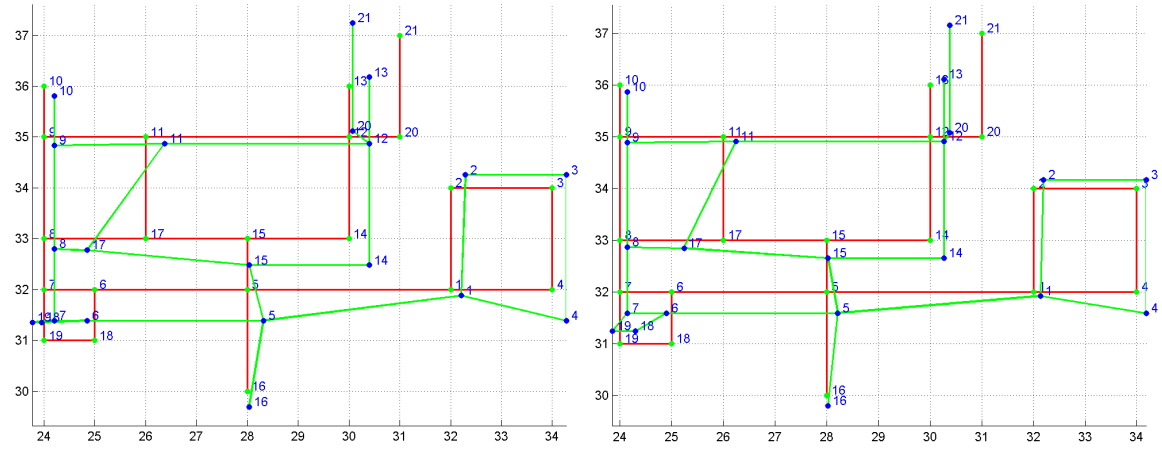


Figure 3.5: Simulated ground truth graphs (in red) and their noisy copy as data graph (in green). On the left; $\sigma = 0.3$. On the right; $\sigma = 0.2$.

the heading direction information is available, the hidden Markov model (HMM)-based approach is also as accurate as the K-best fits, outperforming the other two algorithms. For noise standard deviation $\sigma = 0.2$, the proposed K-best fits algorithm without heading direction information is observed to match an average of 96% of the nodes in the simulated graphs correctly. The K-best fits with direction has achieved 100% for the same settings. The equivalent numbers for the proposed HMM-based algorithm is 99% and 92% respectively. The reason for the bigger gap between the performance of HMM-based in presence of direction information is that this method relies more on the heading direction, as well as high variance in the length of edges, which was not the case in our simulated graphs.

A comparison of the matching accuracy before and after filtering the nodes (as mentioned in Section 3.1) is shown in Table 3.3. It can be seen that filtering the nodes before the matching, makes the matching more robust in all of the shown algorithms. The filtering also decreases the computational complexity of the problem, since the number of nodes in the macro graphs is generally less than the number of nodes in the original graphs. For the specific case of HMM based matching, the filtering was found to be more important, due to its higher complexity. In such cases, the algorithm did not perform well in the absence of heading direction, due to the fact that without filtering, the edges connected to each node have the same weights. Therefore, the nodes do not have much local distinction to be used by the HMM-based matching approach.

It is also important to note that the high accuracy of matching in all of the shown algorithms confirms that the suggested similarity measure (based on all pairs shortest path) successfully captures the topological characteristics of the graphs.

Table 3.2: Simulation results for the accuracy (in percentage of correct assignments) of different matching algorithms, with and without heading direction information. HMM= Hidden Markov model based matching, SM= stable matching, HM = Hungarian method. σ is the standard deviation of the added noise to $\mathcal{G}_D$.

	Kbest w dir.	Kbest w/o dir.	HMM w dir.	HMM w/o dir.	SM	HM
$\sigma = 0.1$	100	98	100	96	97	98
$\sigma = 0.2$	100	96	99	92	87	88
$\sigma = 0.3$	99	91	95	82	76	78

Table 3.3: Simulation results for the accuracy (in percentage of correct assignments) of different matching algorithms, before and after node filtering. SM= stable matching, HM = Hungarian method. σ is the standard deviation of the added noise to $\mathcal{G}_D$.

	Kbest w dir.	Kbest w/o dir.	SM	HM
Sim. before filtering $\sigma = 0.2$	98	78	66	67
Sim. before filtering $\sigma = 0.3$	93	60	47	49
Sim. after filtering $\sigma = 0.2$	100	96	87	88
Sim. after filtering $\sigma = 0.3$	99	91	76	78

Table 3.4: Simulation results for the accuracy (in percentage of correct assignments) in presence of heading direction noise. σ = std. dev. of added noise to $\mathcal{G}_D$. p is the probability of having wrong heading direction.

	HMM w/ dir		HMM	K-best w/ dir		K-best
	$p = 0.05$	$p = 0$	w/o dir	$p = 0.05$	$p = 0$	w/o dir
Simulation $\sigma = 0.1$	100	100	96	100	100	98
Simulation $\sigma = 0.2$	99	99	92	100	100	96

We also tested the sensitivity of the K-best and HMM-based matching algorithms to noise in the heading direction. We tested the matching accuracy in the presence of 5% noise in the heading direction, which is a realistic assumption based on the experiments on our DR module, and it was observed that both of the algorithms are very robust to such noise. In Table 3.4 it can be seen that both of the HMM matching and the K-best fits algorithms have less than 1% deterioration in performance in the presence of the heading direction noise.

Partial matching

We also studied the issue of partial matching, which can occur when $\mathcal{G}_D$ is only a part of the ground truth graph $\mathcal{G}_T$. We used the same method mentioned above to generate

1000 random graphs but only kept a portion of noisy copy of $\mathcal{G}_T$ as $\mathcal{G}_D$. The results of the matching algorithms for partial matching with randomly generated graphs are shown in Table 3.5. It can be seen that the K-best fits algorithm has a superior performance than the other algorithms. Also, when the heading direction is available, K-best fits is capable of exploiting such information to further improve the accuracy of matching. The HMM matching algorithm is also performing equally well in the presence of heading direction information.

Table 3.5: Simulation results for the accuracy (in percentage of correct assignments) of different matching algorithms with or without heading direction information: K-best fits, HMM-based matching (HMM), stable matching (SM), Hungarian method (HM). σ = std. dev. of added noise to $\mathcal{G}_D$.

	K-best	HMM	SM	HM
Sim. 75% of nodes in $\mathcal{G}_D$, $\sigma = 0.2$ w/o dir.	76	75	60	61
Sim. 75% of nodes in $\mathcal{G}_D$, $\sigma = 0.2$ with dir.	84	83	60	61
Sim. 75% of nodes in $\mathcal{G}_D$, $\sigma = 0.3$ w/o dir.	68	65	48	50
Sim. 75% of nodes in $\mathcal{G}_D$, $\sigma = 0.3$ with dir.	81	78	48	50
Sim. 50% of nodes in $\mathcal{G}_D$, $\sigma = 0.2$ w/o dir.	68	69	50	49
Sim. 50% of nodes in $\mathcal{G}_D$, $\sigma = 0.2$ with dir.	77	79	50	49

Sensitivity to scale factor

One of the side observations was that in order for the similarity measure to have meaningful values, the graphs should have the same scaling, which is not the default case since $\mathcal{G}_D$ and $\mathcal{G}_T$ generally have different scales. The scaling factor can be known in many situations, e.g. if the map scale is known and an average number of pixels per step can be estimated.

For cases with unknown scale, the normalization factor can be estimated. The measure we chose for unifying the scales was the sum of all the pairwise shortest path distances between the nodes of each graph. This measure worked well for complete graphs, but for the partial matching case, the performance was sometimes degraded by up to 70%. Therefore, the reported accuracies for partial matching are based on known scaling factors. Figure 3.6 shows the accuracy of the scaling method for the generated 1000 graphs. As can be seen, our normalization method is very effective, with an average ratio of estimated value to correct value equal to 1.004.

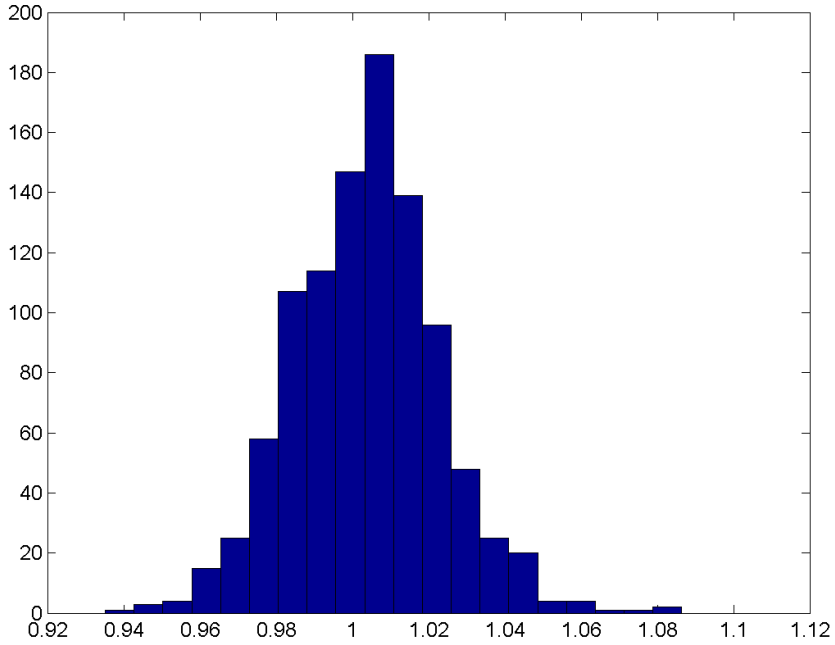


Figure 3.6: A histogram of the accuracy of scale estimation for 1000 simulated graphs, in terms of the ratio $\frac{\text{estimated scale value}}{\text{correct scale value}}$. Mean = 1.004, Std. deviation = 0.019.

3.3.3 Graph Matching, real data

We also tested the matching algorithm on real measured data. The ground truth graph $\mathcal{G}_T$ was manually drawn based on the floor plan of the 4^{th} floor of the Bahen Centre for Information Technology in the University of Toronto. Figure 3.7 (right) shows the drawn ground truth graph, matched on the building's floor plan.

We used the graphs of Category 4 as completely generated data graphs ($\mathcal{G}_D$). Note that $\mathcal{G}_D$ had no absolute location information; only movement and Wi-Fi samples collected by smartphones sensors. We also verified the partial matching accuracy on Category 3 graphs as $\mathcal{G}_D$. A sample of the used data graphs is shown in Figure 3.7(left). Note that the obtained $\mathcal{G}_D$ does not necessarily have 90 degree angles at each intersection since it is the results of averaging over several traces. As mentioned in Subsection 3.3.1, the average distance of the nodes of $\mathcal{G}_D$ from their corresponding node in $\mathcal{G}_T$, due to noise, was 2.65 meters. The error in $\mathcal{G}_D$ was expected to be reduced by the matching algorithms. The angle imperfections were also expected to be resolved after being matched to $\mathcal{G}_T$.

Table 3.6 shows the matching results of the proposed algorithms for Category 4 and Category 3 generated data graphs. It can be seen that the K-Best fits algorithm and HMM matching using all pairs shortest path vector, have obtained 89% and 88% average accuracy, respectively, in matching the points of the filtered $\mathcal{G}_D$ to $\mathcal{G}_T$. For partial matching, an accuracy of 75% for K-best fits on real data is achieved. After fitting the

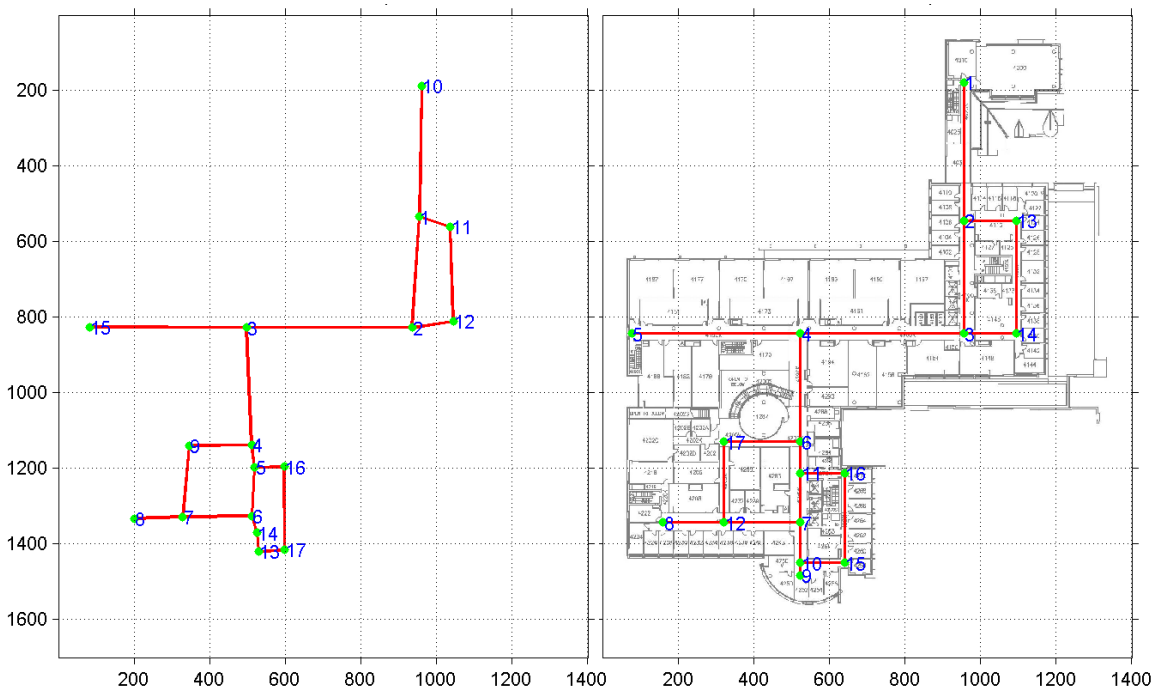


Figure 3.7: On the left; the filtered data graph $\mathcal{G}_D$ of the complete floor obtained from crowdsourced data. On the right; the ground truth graph $\mathcal{G}_T$ of the complete floor.

$\mathcal{G}_D$ on top of $\mathcal{G}_T$ based on the matched nodes, the average error on the location of the nodes was 1.96 and 2.59 meters for the K-Best fits algorithm with and without heading direction information. The equivalent matching errors for HMM matching algorithm were 3.6 and 7.2 meters.

The improvement in the average distance of the nodes of $\mathcal{G}_D$ from $\mathcal{G}_T$ confirms that not only the matching algorithm automatically labels the nodes of $\mathcal{G}_D$, it also reduces the error in the location of the nodes, as the 2.65 meters of error that occurred during the generation of $\mathcal{G}_D$ is reduced to 1.96 meters using the K-best fits algorithm.

The excellent performance of proposed algorithms "K-best fits using the defined similarity measure" and "HMM matching" can be explained by noting that both of the proposed algorithms successfully exploit the neighbourhood information of the nodes in the matching, as well as being capable of using the direction information, when available.

3.3.4 Overall localization performance

Although, this work is more focused on a novel method of radio map generation (calibration phase) and not localization algorithms (operation phase), as a measure for comparison, we also evaluated the operation phase performance using a simple localization algorithm, i.e., the nearest neighbours (NN) method, using the generated radio map.

Table 3.6: Matching results of different matching algorithms on Category 3 and 4 Data graphs, based on real data, with and without heading direction information. The accuracy is in percentage of correct assignments. HMM = HMM-based matching, SM= stable matching, HM = Hungarian method.

	Kbest	HMM	SM	HM
Category 4 (full) graphs, after filtering w/o dir.	84	73	66	68
Category 4 (full) graphs, after filtering with dir.	89	88	66	68
Category 3 (partial) graphs, after filtering w/o dir.	71	52	46	45
Category 3 (partial) graphs, after filtering with dir.	75	72	46	45

We first generated a complete radio map based on the matching of the macro graphs of $\mathcal{G}_D$ and $\mathcal{G}_T$. The operation phase accuracy was evaluated by collecting 985 RSS samples on random locations of the building (in the hallways) and comparing the distance between the actual locations and the localized points according to the NN localization algorithm on the radio maps generated based on Category 4 data graphs. Figure 5.6 presents the empirical cumulative distribution function (CDF) of the localization error over all of the radio maps, using both of the K-best fits and HMM matching for radio map generation. A median error of 3.8 meters and a 90 percentile error of 10.3 meters is observed for the radio maps generated using K-best fits algorithm and using NN algorithm for localization. Table 3.7 presents the average and median error for NN algorithm.

The projection error in localization, representing the distance between the actual location and the closest point in the radio map had an average of 2 meters. This component of error is counted as an inherent mapping error, due to the density of labeled points rather than the accuracy of the radio map. Note that more complicated algorithms and particularly adding a user tracking module (using the user trajectory to improve accuracy) can improve the localization accuracy significantly. For example, as can be seen in the same table, the error in LIFS[94] is reported to decrease from a median of 4.5 meters down to 1.5 meters, once trajectory mapping, i.e., tracking is applied. However, since the actual output of our work is the radio map, and not a localization system that provides location in the online phase, it is best to evaluate the performance of the system in terms of average radio map error, rather than online phase accuracy. Therefore, we are not applying more complicated online phase localization algorithms or tracking to report the online locationing performance. In order words, the online accuracy with or without tracking is not only a function of the used radio map, but is also dependent on the used localization algorithm and the tracking techniques applied on top of it. Therefore, since the concentration of this work is on the training phase of the problem and not the locationing algorithms, the online phase accuracy is not necessarily a good representative of

the proposed system's performance.

Comparison to related works

As a qualitative measure for comparison, Table 3.7 also shows the localization accuracy of some of the related works. It is important to note that a quantitative comparison between these methods is not fair, since as stated above, the difference between such systems is not only in the method of generating radio map, but also in many other factors such as the tested indoor area, its size and shape, the number of labeled points and number of RSS samples per point, and the applied localization algorithm. However, it can be qualitatively seen that our proposed system is showing promising accuracy in comparison with the methods that report errors without using tracking, while not using any direct manpower for training.

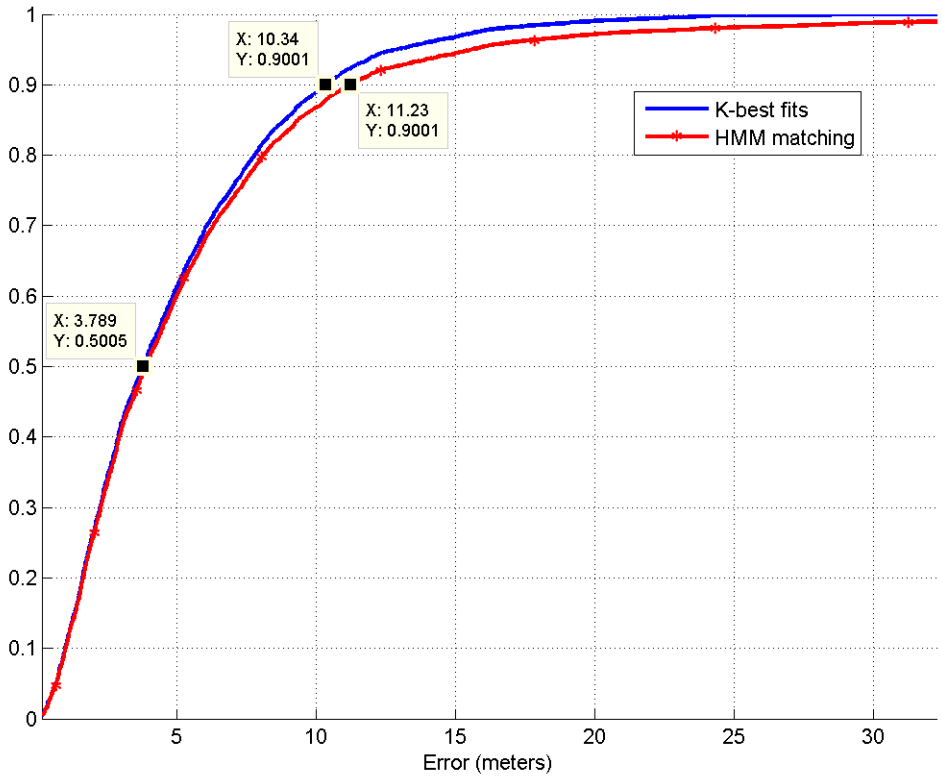


Figure 3.8: The empirical error commutative distribution function for the graphs of Category 4, using NN localization algorithm.

Table 3.7: The accuracy, testbed site size, accuracy, and features for related works along with the proposed approach.

System	**site-size**	**Reported error** (m)	**Tracking?**
SmartSLAM [78]	$1600m^2$	mean $\approx$ 3.5	Yes
Unloc [88]	3 sites: 1750,3000,4000 m^2	mean 1-2	Yes
Walkie-Markie [76]	2 sites: length: $260m$,$310m$	mean 1.65	Yes
ZeeHorus [64]	$2275m^2$ (length: $\approx 230m$)	med. $\approx$7	Yes
LIFS-TM [94]	$1600m^2$	med. 1.5	Yes
LIFS [94]	$1600m^2$	mean 5.88, med. 4.5	No
Kim *et al.*[39]	$4000m^2$	mean 6.9	No
Proposed system	$10000m^2$ (length: $305m$)	mean 5, med. 3.8	No

Chapter 4

Indoor Localization Using Inertial Sensors

4.1 Auto-labeling

In this Chapter, we present our implementation of a dead reckoning (DR) module which consists of a step counting algorithm, as well as a technique for finding the heading direction, to measure the relative displacement with respect to a starting point.

Along the implementation of the DR module, we also implement a technique, we refer to as *auto-labeling*, that is used for speeding up the training phase of RSS based localization. Auto-labeling is a fast site-survey in offline phase by labeling the data via a DR module, using embedded inertial sensors in smartphones. This technique is only applied to offline phase for fast data collection (for building the radio map of the building), but can also be applied in online phase to improve accuracy. Generally, the online phase of the localizer system can be any of the state-of-the-art algorithms and is not discussed here. However, for the purpose of evaluation in the online phase, we present results using the common classification algorithm, known as the K-nearest neighbours (KNN).

Auto labeling can be as fast as walking the entire indoor area regularly, by holding the phone in hand, with an arbitrary gesture and therefore it can be done by any untrained volunteer or contributing user. Using auto-labeled data is experimentally shown to be reliable and a speed up of more than 20 times can be achieved in comparison with the old fashion point by point manual data-labeling phase in RSS based methods.

In the next section, as a background, we will go through a brief overview of two possible inertial navigation methods. Thereafter, we will describe our DR module. The experimental results from the implementations on both Matlab and on our real-time

Android application are given in Section 4.8.

4.2 Inertial navigation systems

There are two common methods for inertial navigation:

1. **Double integration of acceleration**
   According to classical physics, it can be stated that

$$\begin{aligned} \Delta x_t &\triangleq x_t - x_{t-1} = \iint a_t d^2t + \int v_t dt, \\ \Delta v_t &\triangleq v_t - v_{t-1} = \int a_t dt, \end{aligned} \tag{4.1}$$

   where a_t, v_t, x_t denote the acceleration,velocity, and location at time t, respectively. The 3D displacement during each interval Δt can be calculated via the above relation, given the acceleration in the 3 dimensions.

   Note that we are interested in the location of the user with respect to the world's coordinate system, and not the displacements of the user with respect to it's own coordinate system. These two coordinate systems are not generally the same, since the device can rotate freely in the space and the device's frame of reference will change as the device rotates. A gyroscope can be used to find the orientation of the device with respect to the word's reference frame. Each gyroscope sensor will sense the angular velocity of the device according to its own frame of reference in one dimension. Given the 3D gyroscope readings, the rotation angle of the device at each interval Δt can be obtained by integrating the sensed angular velocity:

$$\Delta\theta_t \triangleq \theta_t - \theta_{t-1} = \int \mathbf{v}_r(t) dt,$$

   where, θ_t and $\mathbf{v}_r(t)$ are the orientation angle and the angular velocity at time t. The orientation angle at each time is calculated with respect to its own coordinate system. The calculated orientation, along with the acceleration and magnetic sensor readings can be used to obtain the orientation of the device with respect to the world's coordinate system. More details about how the transformation is expressed and calculated is given in Section 4.3. Once the orientation of the device is found, the 3D acceleration values sensed from the device are transformed to readings with respect to the world's frame of reference via a linear transformation. Thereafter, (4.1) can be used to find the displacement of the user.

The double integration of acceleration, along with a given initial location and velocity at time $t = 0$ can theoretically obtain the location of the user for an arbitrary time t. However, the noise due to double integration starts to accumulate quadratically with time. Therefore, the error becomes intolerable very fast. The noise due to integration is also known as *drift error*, which is the most dominating noise source for such approach. Note that the drift error exists both for integrating the gyro's reading for computing the orientation and also for double integration of the accelerometer readings. Such error is expected to grow quadratically with time due to double integration. There are other error sources such as constant bias, thermo-mechanical noise, bias instability, and temperature effects [93]. It is shown in [93] that the most dominating error source in DR is the drift error due to gyroscope readings. For a quantitative study of error in inertial navigation systems, c.f. [93, 12].

Double integration of acceleration usually works well when the noise is negligible or a set of observations can bound the propagation of error. For example, for systems that have a limitation on their possible moving directions or have a smooth change in their speed or direction, such as robot in certain scenarios. Such system can perform well after proper filtering and correction techniques. Unfortunately, for a mobile device, this is not the case. Although pedestrians have very smooth changes in their speed and heading direction, their phone can have bursty and unpredictable movements in their hand or on their body. As a result, since we do not force the user to attach the phone to his body or keep it in a consistent gesture, the reading noise can lead to a drastic error.

2. **Counting steps (integration of speed)**
   The second technique is dead reckoning. For a pedestrian, a measure of speed can be given via a pedometer (step counter). A pedometer will detect and count the number of steps taken by the user. Then, the covered distance can be calculated via

$$\Delta x_t = \int v_t dt,$$

   or in terms of the number of steps taken in a unit of time:

$$\Delta x_t = nL\Delta t,$$

   where n is the number of taken steps per unit of time and L is the length of each step. This DR technique generally works best for vehicles or wheeled robots where

a more accurate speed measurement instead of nL can be obtained via sensing the wheel rotations. However, using a pedometer for DR is a commonly accepted solution in the literature (c.f., [33, 91, 92, 89]).

In this work, the sensor data is taken from the MEMS sensors incorporated in smartphones. Such sensors are less accurate in comparison with the inertial measurement units (IMUs) that are separately sold in the market. Furthermore, we have tried to have no assumptions about user movements or gestures. Therefore, the observed data is rather noisy. Hence, after trying both of the above techniques, as expected, the second approach was observed to be more accurate.

It is important to note that both techniques update the location based on previously estimated location. As a result, both techniques are subject to error accumulation and cannot be used for a long time without a correction phase that compensates the accumulated error. However, integration of speed is expected to grow linearly with time since the error is accumulated via a single integration, while the double integration method accumulates error quadratically. To bound such error, in the offline phase, we used the DR module with a reliable starting point for a short time. In the online phase, the drift can be bounded by location estimations received from the Wi-Fi localizer or map information.

4.3 Expressing rotation

As mentioned above, since the phone's reference frame is not necessarily aligned with the world's reference frame, we need to find the 3D acceleration values in the world's coordinate system by applying a linear transformation to the phone's 3D readings and convert them to reading with respect to the world's reference frame, i.e., global coordinate system.

The phone's orientation can be expressed by the angle of rotation around each of the 3 axes in the reference frame, known as "yaw", "pitch", and "roll" (Euler angles). Once we know these rotation angles, the linear transformation can be expressed by a 3×3 rotation matrix

$$R = R_x(\gamma)\, R_y(\beta)\, R_z(\alpha), \tag{4.2}$$

where R_x, R_y, and R_z are rotation matrices along each of the 3 axes, whose yaw, pitch,

and roll angles are α, β, and γ, respectively,

$$\begin{aligned} R_x(\gamma) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & \sin\gamma \\ 0 & -\sin\gamma & \cos\gamma \end{bmatrix} \\ R_y(\beta) &= \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix} \\ R_z(\alpha) &= \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned}$$

Note that in a 3-dimensional space, the order of rotation along the 3 axes is important, such that doing the rotation in a different order, results in a different rotation matrix. In (4.2), the *z-y-x* convention is considered for rotation, i.e., the rotation is done along the z, y, and x axes, respectively. The rotation can also be expressed using a rotation vector and an angle of rotation around that vector, known as the axis and angle rotation expression. For a 3D space, such rotation can be represented by a quaternion

$$\left\langle \cos\frac{\theta}{2}, v_x \sin\frac{\theta}{2}, v_y \sin\frac{\theta}{2}, v_z \sin\frac{\theta}{2} \right\rangle, \tag{4.3}$$

where $V = (v_x, v_y, v_z)$ is the rotation vector and θ is the angle of rotation.

In order to find the orientation of the device with high accuracy as well as quick dynamic response, a combination of gyroscope sensor (used for speeding up the dynamic response in updating the orientation), the accelerometer (which is used to indicate the gravity direction) and the magnetic field sensor (for finding the magnetic north pole direction) are used. All of these sensors are embedded in current smartphones. Once we know the direction of gravity and magnetic north, we can find the world's reference frame by defining the z axis to be the opposite of gravity vector, and the y axis to be the geographic north pole (which can be calculated from magnetic north pole). The x axis is defined to be an axis orthogonal to both y and z in a right handed coordinate system, as shown in Figure 4.1.

One of the practical problems with current phone embedded sensors is that the magnetic field sensor is very sensitive to outer noise, particularly in indoor areas. This sensor, which is commonly implemented using the Hall effect (the variations in the electric current in the presence of magnetic field), can be affected easily by the circuitry and rotating

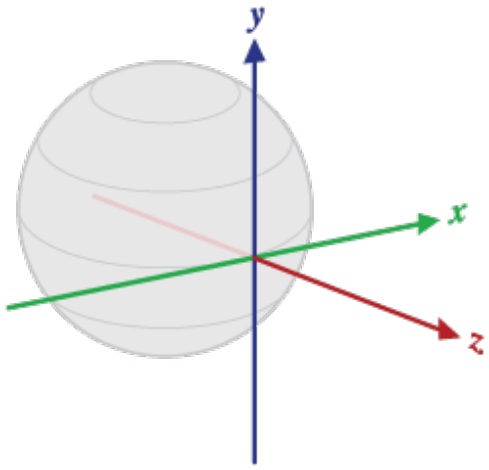


Figure 4.1: The world's reference frame definition [1]

devices inside the phone, as well as the metal foundation of the building and the environment's metal furniture. In Section 4.6, an experiment's result is shown to assess the reliability of magnetic sensors in indoor area. In fact, the same magnetic disturbance is used in Chapter 5 as the fingerprints for indoor landmarks, and used for localization.

Without reliable magnetic field sensor, the 3D orientation of the device w.r.t. the world's coordinate system cannot be calculated. However, we will see in Section 4.6 that even without the magnetic sensor, the relative changes in the heading direction can be calculated using only gyroscope and accelerometer. Consequently, we have designed the step counter to only use the acceleration magnitude (so that it does not require the 3D orientation information), and find the relative heading direction changes without the use of magnetometer.

4.4 Counting the steps

There are many step counter algorithms in the literature, based on the readings from accelerometer and gyroscope, that use a variety of patterns to detect the steps, such as detecting the peaks on accelerometer, zero-crossings of the z-axis acceleration, or correlation of the acceleration readings with some step profiles [21, 65, 31]. Any of the existing approaches, as long as they are accurate and invariant to users, gestures, and phones, can be applied to our system. Our implementation is using a peak detector similar to the method in [33].

In order to detect and count the steps, we use the smartphone's accelerometer. Each acceleration sample is a 3D value, indicating the acceleration in terms of m/s^2, with respect to the phone's coordinate system.

Currently, we implement the step counter using the acceleration magnitude, which is a scalar value independent of the phone's orientation and has the same value, both in the body and world coordinate systems. Consequently, the sensing noise and decalibration, as well as the extra complexity of finding the orientation is avoided, while the user is still not forced to keep the phone in a predefined orientation.

Note that in all of the cases, the acceleration sensors are constantly sensing the gravity along the world's z axis. Therefore, before calculating the magnitude of the acceleration, the gravity's effect should be removed. There are well known methods for such task, such as applying a low pass filter on the samples or combining the information from gyroscope and accelerometer (usually using a Kalman filter) to obtain more accuracy and less delay. We have used the APIs already implemented on Android phone's [1].

After the filtering, the acceleration readings can be broken into two 3D components: *the linear acceleration*, which is assumed to have 0 magnitude (or a constant bias) for a stationary phone, and *gravity*, which is assumed to have the magnitude of 9.81 m/s^2 along the world's Z axis, such that

$$\texttt{raw acceleration} = \texttt{gravity} + \texttt{linear acceleration}.$$

In order to detect the steps from the magnitude of the accelerometer, we implemented a peak detector that can recognize the peaks of the sample reading. Our implementation is a modified version of the method in [33]. These peaks are used to find the user's taken steps. Investigations have shown that the human's walking has a pattern, in which for each taken step, a peak and a local minimum in the acceleration along the z-axis happens. After removing the gravity and taking the magnitude of the linear acceleration, the pattern for each taken step appears as two peaks and two local minima in the sequence of acceleration magnitude samples (Figure 4.2). As a result, with a proper rate of sampling, the steps of a user can be detected by finding the peaks caused by walking. However, there are a variety of sources that introduce error in the peak detection process, including the non-walking gestures that cause ups and downs in the acceleration readings, as well as the sensor noise. In order to eliminate the peaks that are not caused by walking, we define a set of constraints on the detected peaks to remove the invalid peaks by looking at the frequency and the amplitude of the peaks. A peak is considered to be valid only if:

- Its amplitude is the local maximum of a number of its neighbours. The number is selected according to the sampling rate, such that the peak is the local maximum in a window of about 0.1 second.

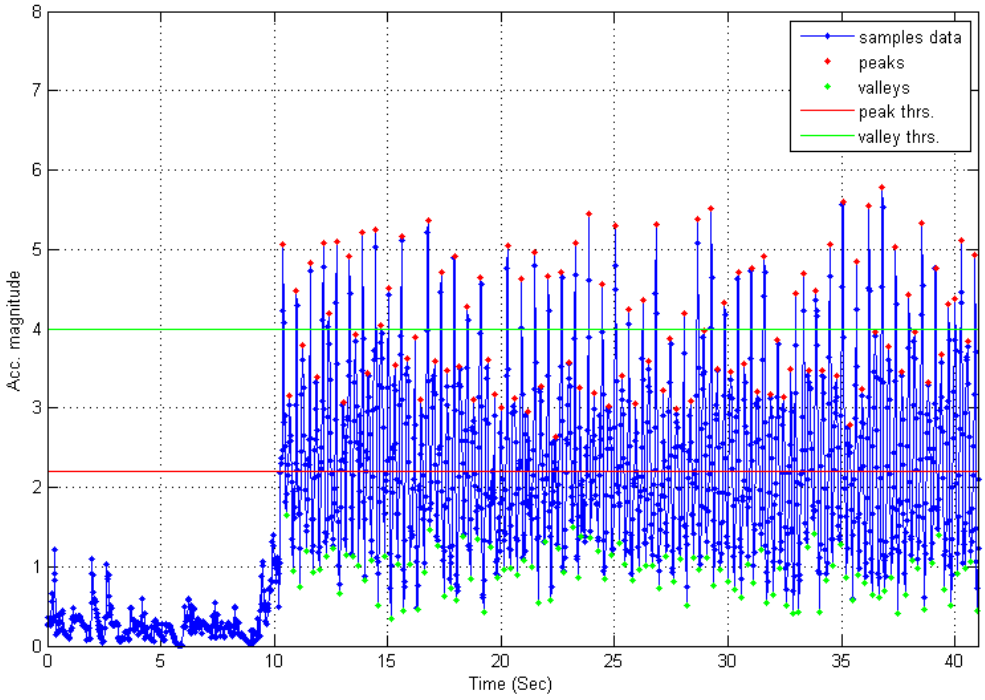


Figure 4.2: Accelerometer magnitude for regular walking. The red dots are the valid peaks. The green dots are the valid valleys. Each two peaks indicate one step taken.

- It is higher than a threshold. We define the threshold to be a factor of the average sensor readings up to the sensed sample, so that the potential bias in the sensor readings will be compensated. Furthermore, depending on the user, phone brand, and on the gesture for holding the phone, a different range of sensor readings is observed. In order for the algorithm to be working on a broader set of these possible combinations, the threshold needs to be selected according to the sensor readings.

- The valid peaks cannot have a magnitude more than a specific threshold. Similar to the previous condition, the threshold is selected to be a factor of average readings in an interval, aiming to weed out the random spikes caused by non-walking events, such as dropping the phone.

- It is observed during a specific allowed time interval after a valid local minimum called *valley*.

- The number of occurred peaks in a time unit cannot be more than a given number (limiting the number of steps that can take place in a unit of time). This constraint implies a limitation in human walking speed.

Similar conditions are defined for detecting valid valleys. The threshold values for the step counter are determined experimentally.

4.5 Estimating the step length

Stride length may vary when different users are carrying the sensors. However, simple calibrations can be performed to estimate the pedometer parameters for a specific user. For example, the stride length can be determined by counting the number of steps taken for a user to cover a fixed distance [33]. In [89], the step length is estimated as a function of the acceleration sensors readings

$$L \approx \sqrt[4]{A_{\max} - A_{\min}} \times C,$$

where L is the step length estimate, $A_{\max}, A_{\min}$ denote the maximum and minimum of acceleration magnitude in one step, and C is a constant value, which is obtained via calibration.

Note that even if the step length is estimated in meters, an extra parameter is required to be calibrated to convert the step length from standard distance units into the floor plan's equivalent distance units. In other words, since each building's map generally has a different resolution and scale, one pixel in floor plan will indicate a different distance in real world. As a result, a pixel-to-meter conversion ratio is required to convert the actual walked distance into pixels and represent it on the monitoring device's screen. In our implementation, we have integrated the two parameters by defining the distance unit in pixels. In the first run, we simply ask the user to determine his beginning and ending position on map shown on the screen, and estimate the step length according to the following relation:

$$L = \frac{\texttt{walked distance}}{\texttt{number of taken steps}} \qquad [\textit{map pixels}].$$

The next time that user walks in the same building, with the same map resolution and scale, only the beginning point is asked from the user. After the walk is complete, user can see the estimation result on the map. If satisfied with the result, user can choose to use the previously estimated step length; otherwise, he/she can update the estimation by providing the end point again and updating the step length estimation. Note that in this approach, is it assumed that each user is walking with a fixed stride length, while users may have time varying stride lengths. We do not discuss such cases, since it is beyond the scope of our main objective in this work. However, adaptive stride length

detection techniques in the literature, such as [44, 11] can be also applied here to improve accuracy for cases with time varying stride length.

In the crowdsourced version of our work, the concentration is primarily on building the data set via a graph model using graph matching techniques. Therefore, similar to [78], we manually set the stride length of each user in our implementations. Although setting the stride length to a constant value is not very accurate and the accuracy can benefit from thorough estimation of the stride length, it is shown in Chapter 3 that our matching algorithm is able to compensate for such inaccuracies, effectively.

4.6 Finding the heading direction

As mentioned in Section 4.4, the system is theoretically able to find the heading direction (similar to an electric compass) by employing the magnetic sensor readings. Given the phone's orientation, one can apply a linear transformation to the sensor readings and obtain the values with respect to the world's reference frame. The angle of rotation from the geographic north (calculated as yaw) will then indicate the heading direction. Note that here, we assume that the user always walks in the $x-y$ plane of world's reference frame, i.e., the user only walks horizontally and will not take the stairs or elevator and therefore, the user will not change the floors. Detecting floor changes can be handled as a separate problem.

In Figure 4.3, we have shown a simple experiment to test the reliability of the magnetic sensors. In this figure, the solid and the dashed curves are the estimated path for a single actual path, walked twice. It can be seen that using heading direction read from the magnetic sensors of the device, the estimated paths show very different heading directions. The major difference between the two curves in the figure seems to be a constant rotation angle, which indicates an unknown constant bias on the readings of heading direction. This inaccuracy introduces even more errors when the walking path becomes more complicated.

To avoid the noisy magnetic sensor readings, instead of calculating the *absolute* heading direction of the user based on magnetic field sensor, we use the gyroscope to find the *relative* heading direction of the user. The gyroscope measures the angular velocity in 3 dimensions with respect to the phone's reference frame. Since the user walks in a plane along the world's horizon, the only rotation angle of interest, which is the user's heading, is equivalent to yaw or rotation angle around the Z axis of the world (also known as azimuth). Therefore, we decompose the rotations of phone, into two orthogonal components: rotation around the world's Z axis, and rotation around some axis in

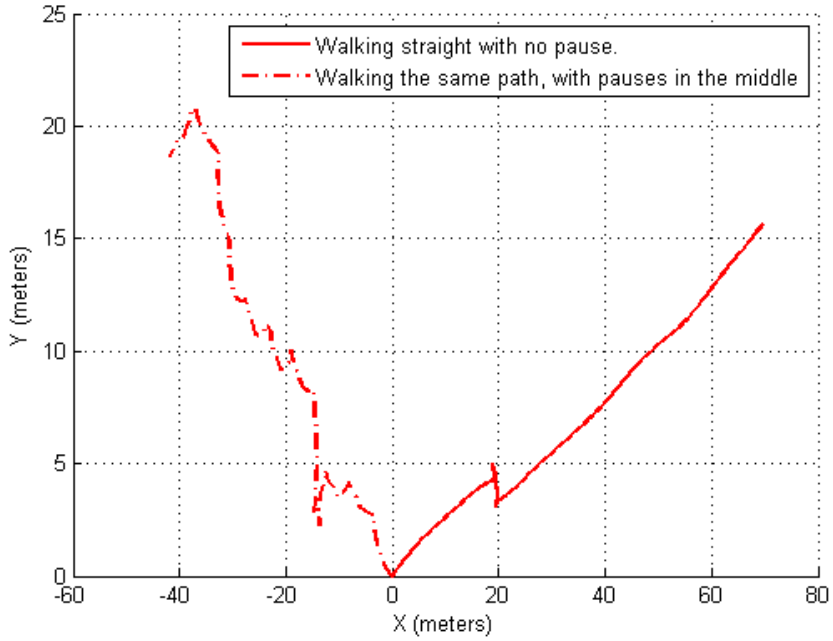


Figure 4.3: The calculated path for a single straight path, walked twice (using a Motorolla Z smartphone).

the $Y - X$ plane, i.e., $\mathbf{v} = \mathbf{v}_Z + \mathbf{v}_{XY}$, where $\mathbf{v}$ denotes the angular velocity of the device in 3 dimensions, while $\mathbf{v}_Z$ represents the angular velocity of the device with respect to the world's Z axis.

To find $\mathbf{v}_Z$, we project the angular velocity (sensed by the gyroscope) to the world's Z axis and then we integrate over time to obtain the rotation angle. The world's Z axis is provided by the gravity sensor of the phone (calculated from accelerometer), which always provides a 3D vector pointing to the earth's centre. Therefore, for each time t

$$\begin{aligned} \mathbf{Z}(t) &= -\mathbf{g}(t)/\|\mathbf{g}(t)\|, \\ \mathbf{v}_Z(t) &= \mathbf{Z}(t).\mathbf{v}(t), \\ d\theta &= \mathbf{v}_Z(t)dt, \qquad [rad] \end{aligned} \tag{4.4}$$

where $\mathbf{g}$ is the gravity vector, $\mathbf{Z}$ is the vector pointing to world's Z axis, $d\theta$ is the amount of rotation around the world's Z axis during the interval dt, and the "." notation represents scalar product.

Since the cumulative displacements of heading direction are subject to drift error, such approach can only be used for short durations of time. In order to eliminate the drift, instead of continuous integration and finding the heading direction, we only look at a short

sliding window of time and detect turns when a user's relative heading direction, goes above a threshold. For example, for changes in the heading direction that are between $[\frac{\pi}{4}, \frac{3\pi}{4}]$, a 90 degree turn is detected. In our current system, we have implemented a module that detects all coefficients of 90 degree turns towards left or right. In Table 4.2, the results of testing the system several times with different orientations, users, and phones is shown, which will be discussed in Section 4.8. Less sharp turns can also be detected with similar approach and different thresholds, with the cost of having less accuracy in detection.

4.7 Employing the inertial localizer in labeling the training data for RSS localizer

As mentioned above, given the initial location, the direction and the speed, the current location can be calculated. In this section, we explain our technique of using the inertial localizer module for data labeling. The inertial localizer module in this study, consists of a pedometer, an estimation of step length, a module for detecting turns using gyroscope and gravity sensor, and a given starting point with initial heading direction. Our proposed approach for auto-labeling the training data is the following:

- The user or technician who intends to perform the training, starts from an arbitrary location in the building, marks his current location on the interactive map, shown on the screen, and indicates his heading direction.
- User keeps walking on a straight path along the given direction and makes occasional turns in the hallway, when necessary.
- The user is allowed to hold the phone in any arbitrary orientation or gesture (such as holding the phone in front of himself, talking on the phone, etc.)
- The user can observe the real-time inertial localizer's estimation of his location.
- When the desired area on the straight line is covered, the user informs the system to stop the inertial navigation module. In this state, a post processing module in the device will start to label the RSS readings during the walk.
- The user will be asked to choose whether the estimation was correct or not. If the estimation was not correct, which happens when the step length factor is not yet calibrated, the user can mark his ending point on the interactive map. The step length will then be calibrated and the data will be labeled accordingly.

According to our DR implementation, the system can cover a broad collection of gestures for holding the phone, but not all possible gestures are covered yet. For example, the system requires a separate set of parameters for detecting the steps if the device is kept in the user's trousers pocket. Note that, the concentration of in this work is more on the use of DR module in RSS based localization, and not on implementing more robust DR modules.

In order to label the data, the system will keep track of the number of steps taken by the time that each RSS sample is received, calculates the cumulative distance from the initial point and assigns the received signal strength values to their corresponding physical locations on the floor plan.

For better accuracy of radio map, further smoothing/filtering can then be performed on the data by merging the close data points and averaging their RSS values, or applying a moving average or any outlier detection algorithm on the data to eliminate the noisy readings.

4.8 Results

The step counter algorithm and the turn detector were tested on several phones, for several gestures and several users. The results are shown in Tables 4.1 and 4.2. As the results suggest, the step counter has a very high accuracy, ranging from 97 to 100% and is the best when the phone is kept in front of the user (which is the default gesture when the user is trying to read its location from the phone's screen). The turn detector module is also very robust to users and gestures and for most of the cases, it provides 100% accuracy. The sampling rate for acceptable accuracy was 50 Hz. With 15 Hz sampling rate, the system still worked, but about 30 percent degradation in the accuracy of the step counter was observed.

#	Phone brand	Tester id	gesture	Detected steps	Actual taken steps	Accuracy %
1	Samsung S1	P1	unknown	39	40	97.5
2	Samsung S1	P1	unknown	60	60	100
3	Samsung G. Tab	P1	unknown	60	60	100
4	Motorrola RAZR	P1	unknown	79	80	98.8
5	HTC Desire Z	P2	unknown	49	50	98
6	LG Nexus 4	P3	unknown	98	100	98

Table 4.1: Step counter results.

#	Phone brand	Tester id	Detected Turns	Actual taken Turns	Accuracy %
1	Samsung S3 (1)	P1	30	30	100
2	Samsung S3 (2)	P1	30	30	100
3	Samsung S3 (1)	P2	30	30	100
4	Samsung S3 (2)	P2	30	30	100
5	Samsung S3 (1)	P3	29	30	96.7
6	Samsung S3 (2)	P3	28	30	93.3

Table 4.2: Results of detecting heading direction changes. 3 users, 2 different phones with the same brand, and arbitrary gestures and a set of possible turns $\{90, 180, 270, 360\}$ degrees were selected.

4.8.1 MATLAB Implementation

In the Matlab implementation, we gathered actual data from the 4th floor of the building "Bahen centre for information technology", in the University of Toronto (about 70m $\times$ 80m floor dimensions), and processed the data offline. Since the process was offline, we had the chance to optimize the step counter and observe the details of the experiment. We designed an experiment to answer the question: *how reliable is auto-labeling?*

We gathered auto-labeled data over a straight line of length 60 meters. In a separate data-set, we also manually labeled the received data. For the auto-labeling, the user simply walked through the path, indicating the starting point and heading direction. For manual labeling, each point was selected on the map and chosen to be assigned to a specific set of RSS readings. Both of the data-sets were then used to test the performance of the system, using the KNN algorithm as the localizer (here we set $K = 1$). In the experiment, each time a portion of the data-points was chosen as the training data and the rest were chosen as the test data. Figure 4.4 shows the results of our experiment. It can be seen that the performance of the system that uses auto labeled data is almost identical to a system that uses manually labeled data. However, the auto-labeled data was collected by just walking through the area of interest, while the manually labeled data required to repetitively walk a few steps, manually label a point on the map, and stand still for a while to collect RSS data.

4.8.2 Android Implementation

We also implemented our algorithm on Android devices to observe the real-time performance of the system with actual data. Everything related to the training was performed

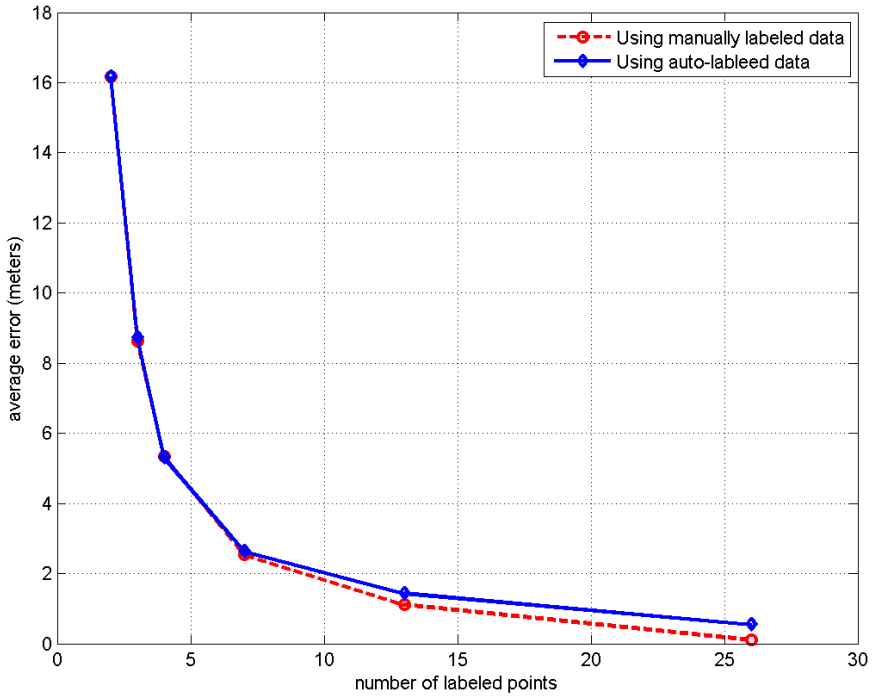


Figure 4.4: Comparison of localization accuracy using auto-labeled data vs. manually labeled data

on the phone. The collected data could then be used in the phone or on the server to perform localization. By auto-labeling, we gathered 347 labeled points in the Bahen center's 4th floor in less than 12 minutes (about 2 seconds per point). For the manual version it took us about 15 minutes to gather 21 labeled points (about 40 seconds per point). Using auto-labeling, for each portion of the building, the user only needed to indicate his initial location and heading direction and walk in the building without taking stops, while for manual labeling, the user had to stop at each point, collect data, and go to the next point.

The system trained by auto-labeled data, performed very good even without any filtering or smoothing. We used a simple KNN algorithm for localization (with the number of close neighbours $K = 3$). Figure 4.5 shows the labeled data in the building (both for manual and auto-labeled data). Figure 4.6 shows a qualitative view of the system's performance. It can be seen that the collected data can be reliably used as a radio map for the KNN online localization algorithm.

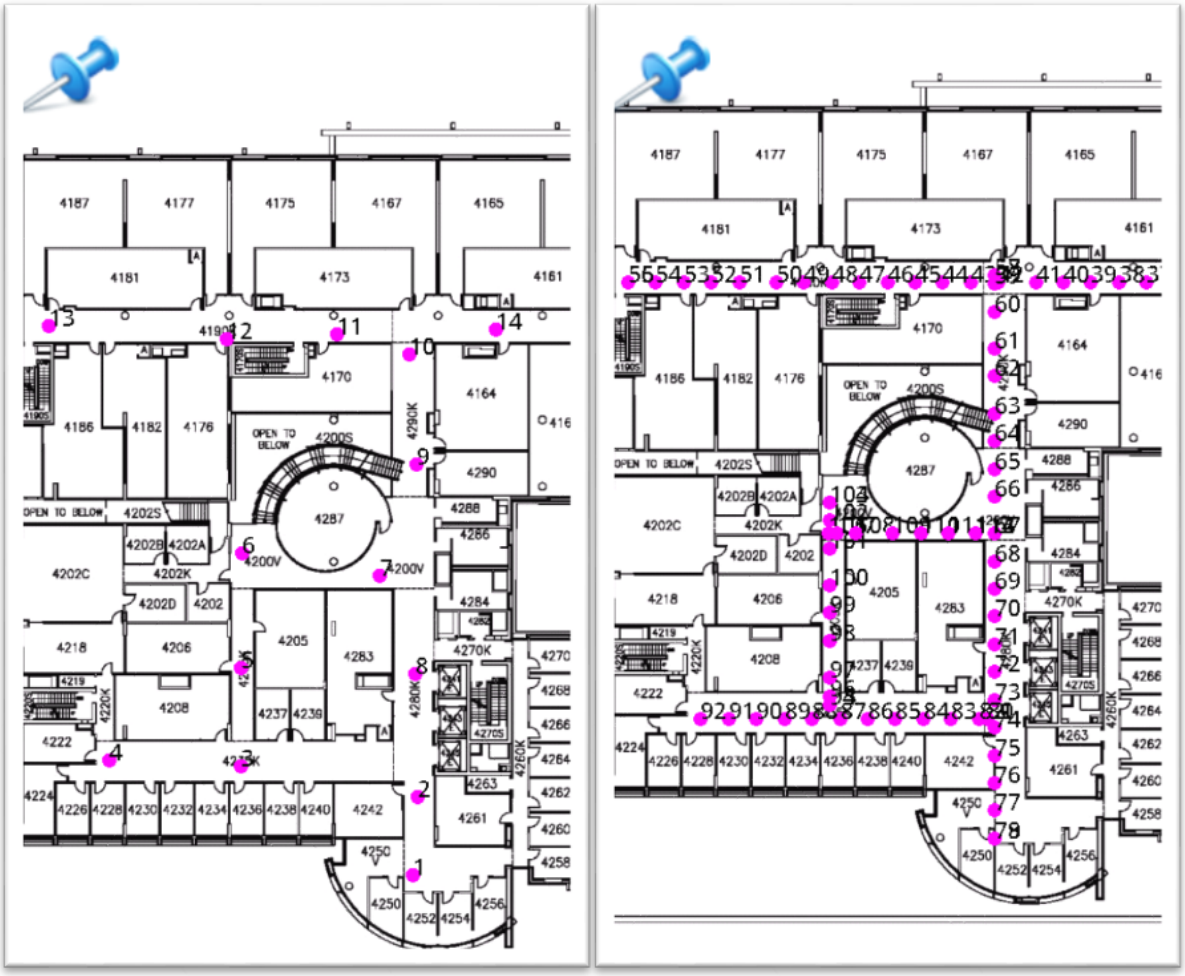


Figure 4.5: Comparison of the auto-labeled data with manually labeled data. The red dots on the right picture are the auto-labeled data (347 points). The pink dots on the left picture are the manually labeled data (21 points).

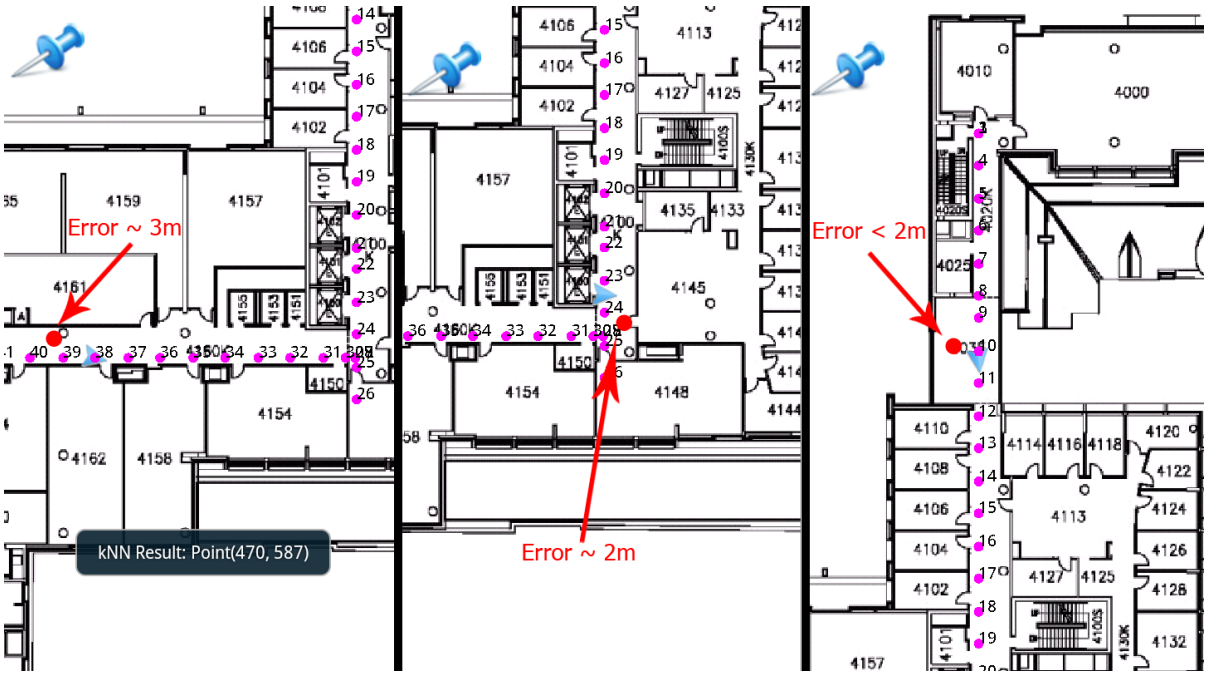


Figure 4.6: A screen-shot of the phone application showing the performance of system, using auto-labeled data. The blue marker is the location estimated by the system. The red circle is the actual location. The pink dots are auto-labeled data, in the training set.

Chapter 5

Indoor Localization Using Magnetic Sensors

5.1 Introduction

As mentioned earlier, the problem of indoor localization has attracted a lot of attention in the last decade, due to being one of the main ingredients for providing location based services. One of the main challenges in this area is that the majority of existing solutions, either rely on extra infrastructure to be installed in areas of interest, such as techniques based on ranging (triangulation/trilateration techniques [35]), or rely on existing infrastructure assuming the availability, such as received signal strength (RSS) based techniques that use available ambient signals, e.g., from Wi-Fi access points (APs) [3, 97]. Although these approaches can work in many settings, they are not useful in many other scenarios, such as the cases where the infrastructure is unavailable, is only available in part of the area, where the ambient signals are subject to occasional change, and especially in emergency rescue missions, when power is out.

In this chapter, we investigate the magnetic sensors readings as a potential signal for the purpose of positioning. Magnetic fields are always available in any location of the earth, due to earth's geomagnetic field, and is also affected by many passive and active magnetic sources, such as the metal infrastructure of the building and the electrical devices.

We first concentrate on the feasibility of using low cost magnetic sensors for indoor positioning, under practical conditions. We introduce a novel low-complexity approach for indoor positioning, by matching the geomagnetic pattern disturbance in indoor area, using smartphone sensors (accelerometer, gyroscope and magnetic field sensor). We show

that with a simple tracking via hidden Markov models, a sequence of magnetic sensor readings along a path can be combined with the correlation of the user's movement trace to track a user's location. Our algorithm only requires users to hold a smartphone, in arbitrary orientation and walk in the building. Since we only use the sensors embedded on the smartphones, no installed infrastructure is needed in the building.

Next, we present a more complex tracking approach, that is based on a combination of dead reckoning (DR) and magnetic sensor readings. The system employs the magnetic sensor readings to limit the drift error of the DR system. The second proposed approach, is more accurate and has a better dynamic response time.

Note that although the methods discussed in this chapter do not require any extra infrastructure such as Wi-Fi APs, with minor modifications, the system can be fused with existing infrastructure-free or infrastructure-relying solutions to become more reliable.

In the proposed method, training dataset can also be collected by a smartphone, simply by walking in the building without the need for device calibration, making the deployment of the system very easy and quick, hence inexpensive.

Note that both of the proposed systems, although promising, are at the early stages of practical magnetic based indoor positioning for smartphone users. The purpose of this study is to show the feasibility of implementing such systems, while making a step toward practice. A complete and reliable solution however, requires more intensive investigation which is beyond the scope of this work.

Related work

As discussed in Chapter 4, the use of pedestrian DR (PDR), i.e., detecting the user's displacement based on the taken steps and heading direction has been around for indoors as well as outdoor navigation since 1990s [47]. The issue with merely using DR for localization is that since the displacement at each time is obtained via integrating over all previous displacements, it accumulates error and soon drifts away from the actual location. The well-known solution to limit the drift error is to combine the DR module with another source of location information with non-accumulative error. The location information can be obtained based on several sources. In [92], the author uses the limitations of floor plan and a foot mounted DR system, however it takes a long time (76 steps on average) to converge to the correct location. RSS based techniques can also correct the DR drift [64, 76] but as mentioned, requires extra or already installed infrastructure.

The works that use magnetic sensors in indoor area are generally divided into two

categories: first; the ones that use the magnetic field to estimate the heading direction, such as [43, 37, 10]. For these works, the indoor anomalies of magnetic readings due to the reinforced concrete and steel architecture of buildings is counted as distortion and the aim is to filter such distortion to obtain a reliable heading direction, similar to outdoor areas where the sensed magnetic fields are majorly due to the earth's magnetic field.

The second category, which is our topic of interest, aims at exploiting the anomalies in the indoor magnetic fields to obtain a unique signature for each location in the indoor area and use them for localization, similar to RSS-based techniques.

There has been various investigations in this area, ranging from assessment of the feasibility of using magnetic sensors for indoor localization [48] to very fine grained localization (centimeter levels [2]) for assigning a path to a robot to follow a certain track on magnetic space [82]. Many of related works are proposing solutions for robots, such as [2, 82, 83], or for wearable devices [67] or generally for more accurate sensors and higher sampling rates than the ones provided in smartphones [69, 28].

Some of the proposed solutions require a partial or completely known heading/orientation of the sensing device [82, 69]. This requirement is not valid when a smartphone with arbitrary orientation is used.

Other approaches suggest using the magnitude of the 3D magnetic sensor readings [38, 67, 52, 30], to make the readings heading/orientation independent. Calculating a reliable magnitude of magnetic field requires calibration of the sensor readings, since each axis of the 3D sensor readings usually has a different bias that gradually changes over time and since the magnitude function is a non-linear function of the magnitude values, the bias of the values needs to be compensated before magnitude calculation.

The calibration of magnetic sensors, specially the low cost smartphone sensors, usually entails the users to move the phone/device in every possible direction in the air, while keeping the center of movement constant [2], which can be an inaccurate and time consuming practice for an untrained user, especially if needed to be done before each usage. To avoid such process, we make the sensor readings orientation independent by using a linear function of the 3D axis readings. Since the bias does not make rapid changes, the linear function keeps a device's bias constant for arbitrary orientations.

In the majority of the related works, the magnetic sensors are used similar to the way RSS-fingerprinting methods are deployed, i.e., by measuring the intensity of the magnetic readings (either as a 3D vector or just the magnitude). This method raises the need for calibration and accuracy of the readings at each location even further, since a single reading can provide at most 3 features for each location (readings along x,y,z axis), which may not be unique enough, specially if exposed to noise or an offset, in comparison

with Wi-Fi fingerprints where the number of scanned APs for a single location can be easily more than 50.

In this work, we have proposed using magnetic sensor readings as a sequence, as opposed to single reading vectors that are more effective in RSS-fingerprinting based techniques. We conjecture that the magnetic sequence patterns along a path, if long enough, can be considered as a unique signature for that path. A similar idea has also been presented in [56], for using magnetic sensors in combination with vision based localization for a wearable camera system. We show that using magnetic sequences along paths rather than single values on grids is both unique and also robust enough to be collected with low cost smartphone sensors, with lower sampling rates at arbitrary device orientations.

In the next section, we explain our first proposed approach, along with an experimental study of the nature of the problem. We discuss the limitations and required assumptions in the state of the art magnetic based approaches, and how our proposed solution enhance some of those limitations. In the following section, we explain a more complex tracking algorithm that combines the magnetic information with DR to improve reliability and response time.

5.2 Geomagnetic indoor positioning using smartphones

As mentioned above, we are interested in the problem of indoor positioning using smartphones, under practical settings, with no need for any extra infrastructure. Similar to RSS based methods, in the training phase, we generate a training dataset, we refer to as the *magnetic profiles dataset*. In the online operation phase, we match the queries with the training dataset to obtain the location estimation. Both of the training and online data go through the same data processing, as Figure 5.1 suggests. In the following, we explain the blocks depicted in Figure 5.1.

5.2.1 Training phase: generating magnetic profiles dataset

Figure 5.1 shows the different components of our proposed system. We use the sequence of magnetic readings as signature of a path, rather than the single readings as signatures of each location. Using a sequence instead of a single sample for comparison, makes the readings more distinct and more robust to noise. The readings in the training phase are used as a training dataset, to locate the users in the online phase.

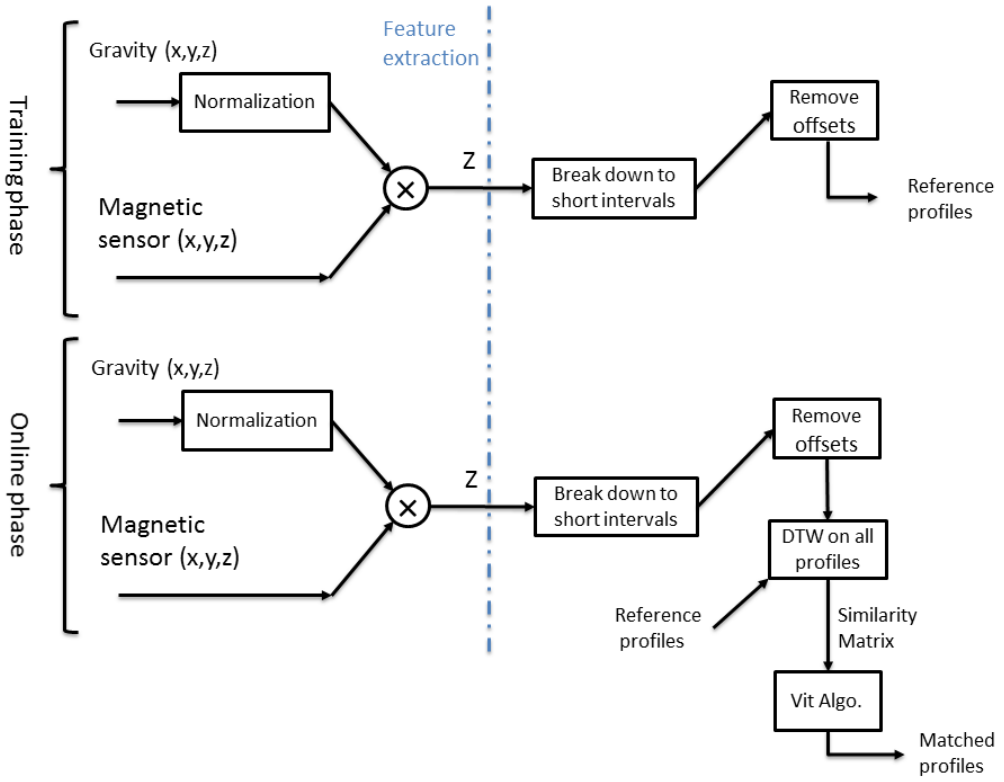


Figure 5.1: The schematic diagram of proposed localization system in training and online phase.

To build the training dataset, a person walks along the paths of the building with a phone in hand. While the path is being walked, the physical location is also recorded to tag the sensor readings with their corresponding location. The location information can be calculated via different methods, such as time-stamping several checkpoints along the paths, while assuming constant walking speed between each two points, or measurement devices such as dead reckoning modules or laser range finders.

To collect data for big rooms or the hallways that are wider than 2-3 meters, several parallel paths can be collected. Note that the paths do not necessarily need to be parallel and any walked trace with known location can be used to generate reference profiles.

Feature extraction

Once a recording of raw magnetic readings tagged by the location is obtained, we make the readings independent of the phone's orientation. This process is necessary since we require no restrictions on the gesture and orientation of the phone when walking. Ideally, this process is done by estimating the orientation of the device and transforming the

sensors' readings (which are measured in terms of the phone's reference frame, $x,\ y,\ z$) to the world's reference frame $X,\ Y,\ Z$, where X is pointing to the geographic north pole, Z points to the opposite direction of gravity and Y is orthogonal to both X and Z with its direction in a way that the system $X,\ Y,\ Z$ builds a right handed coordinate system.

Ironically, estimating the inclination from the magnetic north heading (used for finding the geographic north) requires undistorted magnetic readings, which is not available in indoor areas. Therefore, a reliable transformation cannot be done completely in indoor areas. In Section 4.4 we used the magnitude of the 3D accelerometer readings, to make the reading orientation independent. However, due to the bias in the readings of the magnetic sensors, the magnitude of biased readings cannot be used as an orientation independent feature of the readings. Note that for accelerometer readings, estimating the bias can be done reliably by detecting the device being at rest and knowing the fact that a device at rest should only sense the earth's gravity, while calibrating the bias of magnetic sensors is more complicated and less user friendly.

We suggest a novel feature of the magnetic readings to relax the constraint of orientation free data collection with no need to bias calibration. Although a complete 3D magnetic reading along the world's reference frame cannot be accurately calculated due to unreliable estimation of the angle of rotation with respect to magnetic north pole, the rotation angle with respect to the world's Z axis can be estimated with high accuracy by sensing the direction of gravity, similar to the technique used for detecting heading direction in Section 4.6.

Using the gravity vector, we decompose the 3D magnetic sensor readings of the device to two components: a component along the world's (reference frame's) Z axis and a component orthogonal to worlds Z axis, i.e., in the world's XY plane. The first component can be obtained by projecting the 3D magnetic readings onto the world's Z axis, provided by the gravity vector. Since the readings along the Z axis are independent of the phone or the user's heading, we can use them as heading and orientation independent features for each location. Furthermore, as the signatures are obtained in sequence, there is a possibility of reversing the sequence of readings to obtain a signature of a path walked in the opposite direction.

According to our observations the need for calibration of the magnetic sensor readings on smartphones is mostly due to a rather constant 3D bias, since the reading scales are almost accurate. Assume a random 3D sequence $\mathbf{M} = (M_x, M_y, M_z)$ to be representative of the 3D magnetic readings along the world's $X,\ Y,\ Z$ axes in a hallway. For an arbitrary, but fixed, orientation of the device, the sensors will sense $\mathbf{M}$ in a rotated coordinate

system with the axes $x,\ y,\ z$. Denote the sensed sequence as $\mathbf{m} = (m_x, m_y, m_z)$. Given the gravity vector $\mathbf{g}$ in the rotated coordinate system, the readings along Z can be obtained from $\mathbf{m}$ by the following projection:

$$M_z = \mathbf{m}.(\mathbf{g}/|\mathbf{g}|),$$

where $|\cdot|$ denotes the L_2 norm of the vector.

In practice, the rotated coordinate system is shifted by a bias, therefore $\mathbf{m} = \mathbf{m}_{unbiased} + \mathbf{b}$, where $\mathbf{b}$ is the bias vector, assumed to be constant in short intervals. The estimated projection of the magnetic signal on the world Z axis, M_z will then be affected by a constant bias, depending only on the bias values and the orientation of the device [1]:

$$\begin{aligned} \hat{M_z} &= (\mathbf{m}_{unbiased} + \mathbf{b}).(\mathbf{g}/|\mathbf{g}|) \\ &= \underbrace{\mathbf{m}_{unbiased}.(\mathbf{g}/|\mathbf{g}|)}_{\text{The actual } M_z \text{ value}} + \underbrace{\mathbf{b}.(\mathbf{g}/|\mathbf{g}|)}_{\text{constant w.r.t. } \mathbf{m}} . \end{aligned} \tag{5.1}$$

The common alternative for making the readings orientation independent is using magnitude of m. In such case, the offset affects the magnitude in the following way:

$$\begin{aligned} |\mathbf{m}| &= |(\mathbf{m}_{unbiased} + \mathbf{b})| \\ &= \sqrt{(m_x + b_x)^2 + (m_y + b_y)^2 + (m_z + b_z)^2} \\ &= \sqrt{\underbrace{m_x^2 + m_y^2 + m_z^2}_{\text{the actual magnitude}} + \underbrace{b_x^2 + b_y^2 + b_z^2}_{\text{constant value}} + \underbrace{2(m_x b_x + m_y b_y + m_z b_z)}_{\text{value dependent on both offset and } \mathbf{m}}} , \end{aligned} \tag{5.2}$$

which is clearly affected by a non-constant value for different readings of magnetic sensor.

To better illustrate why the Z axis projection feature is more reliable than the magnitude of the 3D readings, we show a simple example. In Figure 5.2, the projection of $\mathbf{M}$ along Z axis, as well as the magnitude of the sequence for both of the original signal $\mathbf{M}$ and its rotated biased version $\mathbf{m}$ are seen. It can be observed that the projection on the Z axis is only affected by an offset, while the magnitude changes are not constant. Figure 5.3 elaborates the difference between these two features, by comparing the difference between the values of magnitude feature with and without calibration and the values of Z feature with and without calibration.

It is important to note that if the biases are removed, both features work reliably. Otherwise, in the presence of sensing bias, the projection onto the Z axis will be off by

[1] Note that we assume arbitrary but fixed orientation of the device, as we break down the sequences into shot intervals, in which the device can be assumed to have a fixed orientation

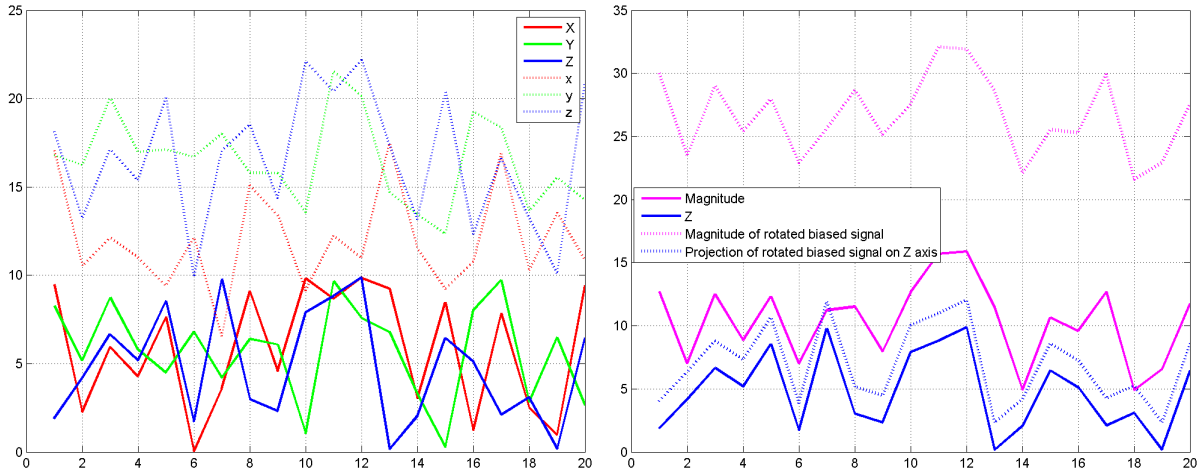


Figure 5.2: Left; A random 3D signal (X, Y, Z) and its rotated and biased version (x, y, z) (rotation Euler angles: $(48, 55, 27)$, offsets: $(10, 10, 10)$. Right; The magnitude and the Z axis projection of the actual signal and its rotated and biased version.

a constant offset, while the magnitude values also depend on the signal values, due to nonlinearity of L_2 norm. Since we aim to avoid the need for the calibration of the device, using the magnitude feature on an uncalibrated set of magnetic readings will lead to a random error, while using the Z axis feature, only leads to a constant offset.

Breaking the sequence into short sub-intervals

After feature extraction, the recorded magnetic readings along the world's Z axis are broken into smaller segments, each covering a few seconds of the walked area. Each of these walked segments is called a *profile*. The profiles, each consisting of a line segment (with known starting and ending coordinates) and a sequence of observed magnetic readings along the segment are stored in a reference dataset. To allow for better matching, profiles can overlap. For our tests, we used 5 second intervals with 2.5 seconds of overlap. Breaking a long walked path into several profiles allows for partial path matching of the magnetic readings, when a user walks through a portion of a long sequence.

Removing the offsets

As mentioned above, since we assume no calibration on the sensor readings, a bias in the readings leads to a constant offset in the estimated sequence of readings along the world's Z axis. To be able to compare the profiles, this offset should be removed. We make the profiles offset independent, by removing the mean of each profile. Making

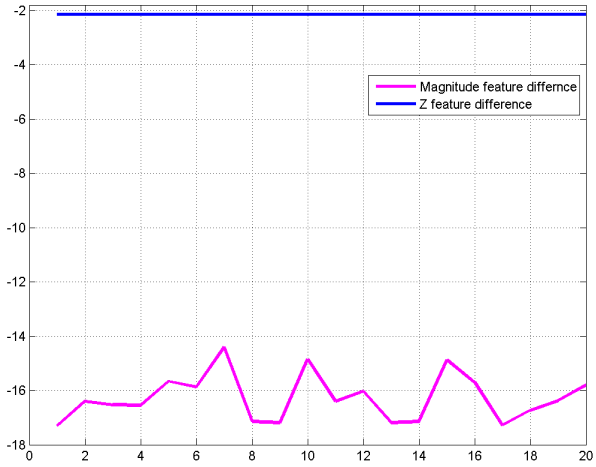


Figure 5.3: The effect of offset on the signals from Figure 5.2. The magenta curve is the difference between the magnitude of the actual signal and the magnitude of the biased rotated signal. The blue curve is the difference between the Z projection of the actual signal and its biased rotated version. As can be seen, the Z feature has only changed by a constant value.

both of the dataset profiles and the online reading intervals zero-mean relaxes the need for calibration of the device. Note that although removing the mean makes the profiles offset independent, it also removes some of the uniqueness features of the profiles, as two sequences with similar pattern but different offsets may be referring to two different parts off the building. Therefore, estimating the offset via calibration, although not necessary, can certainly improve accuracy.

5.2.2 Online phase: Matching the online readings to the magnetic profiles in dataset

In the online phase, when a user walks in the building, a sequence of magnetic readings, similar to the reference profiles, is observed. The online data is first pre-processed similar to the profile dataset. Next, every few seconds, the recent sequence of magnetic readings is matched to the reference profiles. Although not necessary, the length of the observation is typically equal to the length of the profiles. Since we compare the sequence of observed magnetic readings along a path, the user is assumed to be walking, during each time interval. The occasional stops along the path can be detected by a movement detection tool based on the accelerometer sensor or a DR system that recognizes the number of steps. In such intervals, since user's location does not change, no profile matching takes place.

Using dynamic time warping for comparing the profiles to online data

In order to match the online readings with the reference profiles, a distance measure needs to be defined between the reference profiles and online readings. The used distance measure needs to be invariant to different sequence time shifts and time scales.

The difference in the timescale of two sequences can happen if they are collected when walking with different speeds, or if they are collected at different sampling rates. This difference makes the sample-by-sample comparison of the two sequences inaccurate. One of the well-known algorithms to compare two sequences that may vary in time-scale is the dynamic time warping algorithm (DTW) [54]. This algorithm maps the samples of one sequence to a sub-sequence of the other sequence (with allowed many to one mappings) in a way that the minimum distance between the two sequences is achieved (Algorithm 5.1).

Algorithm 5.1: Dynamic time warping algorithm.

input : two sequences S_1, S_2 and a distance measure D
initialize Matrix $\mathrm{DTW}_{|S_1|\times|S_2|}$, where $\mathrm{DTW}(0,0)=0$, $\mathrm{DTW}(1\ldots|S_1|,0)=1$ and $\mathrm{DTW}(0,1\ldots|S_2|)=1$
for $i=1$ *to* $|S_1|$ **do**
  for $j=1$ *to* $|S_2|$ **do**
    $\mathrm{DTW}[i,j]=D[i,j]+\min\left(\mathrm{DTW}[i-1,j],\mathrm{DTW}[i,j-1],\mathrm{DTW}[i-1,j-1]\right)$
  end
end
output: $\mathrm{DTW}(|S_1|,|S_2|)$

Note that the dynamic time warping algorithm, in its original form, forces all of the elements of each sequence to be matched to an element of the other sequence. Therefore, although the results will be a time-scale independent matching, it is not time-shift independent. In other words, if one of the signals is a shifted version of the other signal, the DTW may output a high distance. The time-shift difference happens when one of the sequences has a time offset w.r.t. the other sequence. This can happen if the online readings have a partial overlap with a profile, but are not exactly the same as the profile readings.

There are more complicated versions of the algorithm that allow for sequences with partial overlap to be matched. For example, the algorithm can be extended to run several times, from different starting points on the first row/column of the calculation matrix DTW. Exemplifying, if the algorithm starts from DTW(1,10) (instead of DTW(1,1)), the first 9 samples of S_2 will be ignored, allowing for a shifted version of the second sequence

to be matched to S_1. Calculating the distance by allowing shifted versions of sequences, obtains a more reliable distance value, at the cost of higher computational complexity. Since we are interested in a real-time localization system, we have used the original DTW distance, rather than more complicated versions.

Tracking via HMM

Consider a user's movement trace to be broken into N smaller traces, each consisting of the user's movement, in one time-step. To locate the user, at certain time intervals, we calculate the distance of the recently observed sequence of magnetic readings with all of the reference profiles to find the best matching profile. To further improve the accuracy of the matching, we track the user by only allowing movements between neighbour profiles. The tracking is modeled via a hidden Markov model (HMM).

We aim to map the user's location at each time step to one of the profile locations in the training dataset, indexed from 1 to M: $\mathcal{S} = \{1, 2, \ldots, M\}$, where M is the number of profiles in the training data-set. Each profile in the data-set is represented by the coordinate of the ending point of the profile's segment, as well as a sequence of magnetic sensor readings transformed along the world's Z axis, with removed average.

Similar to the movement trace, the sequence of magnetic observations along the user's walked path is also broken into N subsequences $\mathbf{X} = (X_1, X_2, \ldots, X_N)$ where X_i is a sequence of magnetic readings during the i-th time-step.

The localization problem for the whole sequence of the walked path can be formulated as estimating the best matching sequence of walked profiles $\hat{\mathbf{Z}}$:

$$\hat{\mathbf{Z}} = \arg\max_{\mathbf{Z}} \Pr(\mathbf{Z}|\mathbf{X}), \tag{5.3}$$

where $\mathbf{Z} = (Z_1, Z_2, \ldots, Z_N)$ is the sequence of states (data-set profiles) the user was located at each time step, with each $Z_i \in \mathcal{S}$. Applying the Bayes rule, we have

$$\hat{\mathbf{Z}} = \arg\max_{\mathbf{Z}} \Pr(\mathbf{Z}, \mathbf{X}). \tag{5.4}$$

Using the product rule, the Markov property of the user's location, and noting that

the observation X_i at time-setp i only depends on the state at time-step i, we have

$$\begin{aligned}\Pr(\mathbf{Z},\mathbf{X}) &= \Pr(Z_1)\prod_{i=2}^{N}\Pr(Z_i|Z_{i-1},\ldots,Z_1)\times \\ &\quad \Pr(X_1|\mathbf{Z})\prod_{i=1}^{N}\Pr(X_i|X_{i-1},\ldots,X_1,\mathbf{Z}) \\ &= \Pr(Z_1)\prod_{i=2}^{N}\Pr(Z_i|Z_{i-1})\times\prod_{i=1}^{N}\Pr(X_i|Z_i).\end{aligned} \tag{5.5}$$

Taking the logarithm of equation (5.5) and letting the initial states to be equiprobable, the problem can be written as:

$$\hat{\mathbf{Z}} = \arg\max_{\mathbf{Z}}\left[\sum_{i=2}^{N}\log(\Pr(Z_i|Z_{i-1})) + \sum_{i=1}^{N}\log(\Pr(X_i|Z_i))\right]. \tag{5.6}$$

For ease of notation, we represent the probability of the occurrence of an event z_i, $\Pr(Z_i = z_i)$ as $\Pr(z_i)$. Using (5.6), We can calculate $\hat{\mathbf{Z}}$ via the Viterbi algorithm as explained below.

We model the transition probabilities to be uniform around a neighbourhood of each state

$$\Pr(z_n|z_{n-1}) = C\times\begin{cases}1 & \text{if } D(z_n,z_{n-1}) < thrs \\ 0 & \text{otherwise.}\end{cases}, \tag{5.7}$$

where z_n represents the outcome of Z_n, $thrs$ is a constant value, selected based on the maximum possible distance a user can walk during a time-step, and D is the Euclidean distance between the end points of two profiles, The observation probabilities $X_i, i \in \{1,\ldots,N\}$ given the state z_i are modeled to be

$$\Pr(x_i|z_i) = C\times\exp\left(-DTW(x_i,z_i)\right),$$

where $DTW(x_i,z_i)$ is the DTW based distance between the observed magnetic sequence x_i and the magnetic sequence of the profile z_i, while C is a normalization factor.

To calculate $\hat{\mathbf{Z}}$ using the Viterbi algorithm, we build a trellis of all possible states $\mathcal{S}$, and at each time-step i, we calculate the most probable sequence of walked states $(\hat{Z}_1,\hat{Z}_2,\ldots,\hat{Z}_i)$ for all $Z_i \in \mathcal{S}$ based on the best path until the $(i-1)^{\text{th}}$ time step, using the metric

$$\log(\Pr(Z_i|Z_{i-1})) + \log(\Pr(X_i|Z_i)). \tag{5.8}$$

For each $Z_i \in \mathcal{S}$, we have to maximize the above relation over all $Z_{i-1} \in \mathcal{S}$, leading to $|\mathcal{S}|^2$ comparisons, where $|\mathcal{S}|$ is the size of the state space. In addition, the comparison of the sequence X_i with length L_1 with each of the Z_is with length L_2 using DTW has complexity $\mathcal{O}(L_1 L_2)$. Therefore, calculating the most probable sequence according to the above metric has the complexity $\mathcal{O}(|\mathcal{S}|^2 + |\mathcal{S}|L_1 L_2)$. This calculation is repeated for each time step in the online phase.

The complexity can be further optimized if a list of neighbouring profiles is kept to avoid searching for most probable paths when two profiles are not neighbours. To be more specific, since many of the transitions between states are expected to be impossible according to (5.7), a list of neighbour states (the states that have non-zero transition probability) can be kept for each state. Using such neighbourhood list, assuming that the number of neighbour points for each state is at most a constant $K \ll M$, the maximization of the first term is only calculated over at most $K|\mathcal{S}|$ states, where K is a constant, leading to the complexity $\mathcal{O}(K|\mathcal{S}| + |\mathcal{S}|L_1 L_2) = \mathcal{O}(|\mathcal{S}|L_1 L_2)$.

Implementation and results

In this section, we demonstrate the results of several experiments on an implementation of the system on a real testbed.

The system was partly implemented for Android operating system and partly on Matlab. We collected both of the profile dataset and online data using a logging algorithm for Android operating system. To collect the ground truth location, we used two methods. Initially, we used the dead reckoning (DR) module we have implemented in Chapter 4. The DR performance was first tested and carefully calibrated to obtain reliable location for ground truth purpose.

To assure that the ground truth locations are accurate and that the results are replicable using other methods, we repeated some of the experiments while collecting the ground truth by setting 12 checkpoints along the path, and measuring the time when each checkpoint was passed. The location between the checkpoints was obtained via interpolation, assuming the walking speed was constant between each two checkpoints. This method was estimated to have less than 1 meters of average error, since the time stamp was manually recorded and could happen during a one-step period, which could be about 1 meters out of sync with the actual time of passing each checkpoint. Figure 5.4 depicts the comparison between the DR and the time based ground truth path obtained for one of the experiments. The first part (top right part) is an exact overlap of the two paths. However, as the user has walked toward the end of the training path, the DR

has slightly drifted away. Overall, it can be seen that the two methods have measured very similar traces. It is also observed that as the end of the path is reached (the square area), more difference between the two paths is accumulated. The reason is that the DR accumulates error while the timestamp-based method does not.

After collecting the data, calculations were done on a computer to simulate real-time localization, as well as non-causal localizations. For each testing trace, the online localization results were calculated by taking the most likely location at all time-steps according to the Viterbi algorithm, while the non-causal tracking results were obtained by going through the whole online trace and then back-tracking through the path with the highest probability.

We did several tests with different scenarios to assess the performance of the system. Our tests were done on a single floor of the Bahen Centre for Information Technology, in the University of Toronto. The building structure was mainly reinforced concrete. Figure 5.5 shows the floor plan of the building as well as the sample path that was walked through the tests. Each time, either the whole path, or only part of it was walked, without making stops through the path. The full length of the path was about 170 meters.

We first collected a profile data-set by holding the phone in horizontal direction in front of the body and walking through the whole path. This process simply took a few minutes, as the path was only needed to be walked once. Next, the same path was walked several times as a test with different phone gestures, such as horizontal, vertical, and randomly changing the orientation of the device. Figure 5.6 shows the cumulative distribution function (CDF) of the error in localization of the proposed system. It can be seen that median error of 2.1 meters is obtained while the 90 percentile error is about 13.8 meters for the real-time localization, which is very promising since no extra locationing information other than the geomagnetic field sensor is used and all of the data is collected via smartphones, by walking naturally in the building. The red starred curve is the non-causal localization results. As expected, the non-causal results are more accurate, since they are obtained after the whole path is walked. Such results can be useful, if the system is being embedded in a crowdsourced system, where the data can be processed in an offline manner to produce more reliable training datasets.

We also tested the consistency of the profile datasets from several aspects. Table 5.1 demonstrates the median and mean error of test traces collected on the same path with different gestures, as well as a path walked 1 meters aside from the training path, a trace collected with 3 month difference in time, and a trace collected in the reverse order.

The results for repeating the experiment with the time-stamping based ground truth

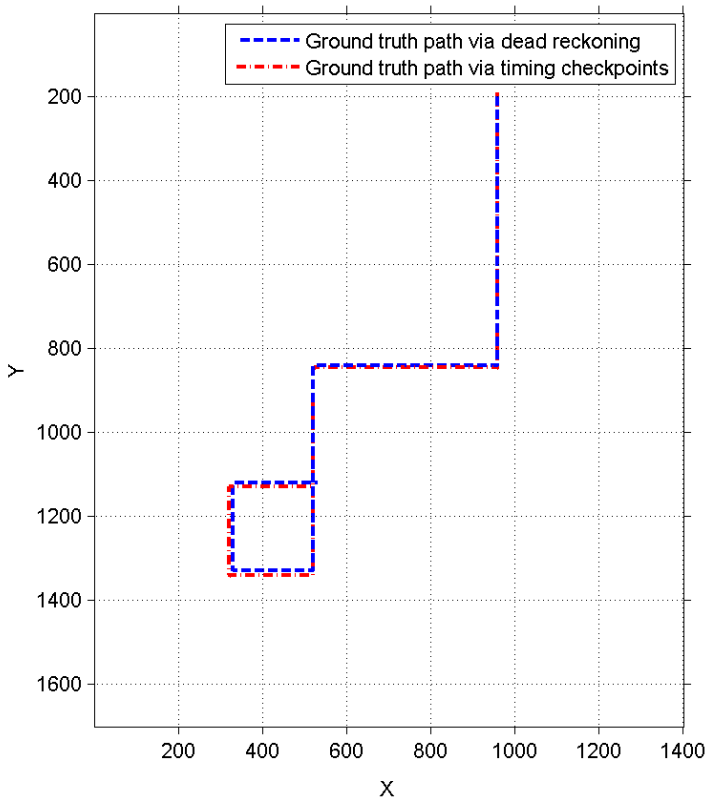


Figure 5.4: The estimated ground truth path collected using two different methods. The paths overlap completely at the beginning (top right corner). The DR-based path gradually accumulates error as the distance grows.

collection is shown in Table 5.2. It can be seen that the results are generally similar to the ones in Table 5.2 confirming the reliability of the experiments. The difference between the values of the repeated experiments is due to both the difference in the method of ground truth collection and the fact that the experiments are done on real environment, which is subject to unpredictable noise. The noise is expected to be canceled if the results are averaged over hundreds of experiments, whereas we only repeated the experiment a few times.

The high variance between the real-time values (specially the mean values), is due to the fact that in some of the experiments, the HMM converged after several time steps, leading to a high location error in the beginning of the trace. Generally, the real-time accuracy relied very much on when the tracking converged to the correct position, with a slight chance of not converging at all. Therefore, indoor localization using merely magnetic sensors on smartphones, although promising, does not guarantee certain localization accuracy. The early convergence chance can be increased if prior information about the initial whereabouts of the user is known, or as will be seen in next section, side

Figure 5.5: The testbed floor plan with a sample path.

information via extra sources of location information, such as inertial sensors is combined with magnetic pattern matching.

One of the interesting observations in Table 5.1 is that the system is able to recognize a path in reverse, once it is compared to the dataset readings in reverse order. This makes the data collection process faster and less expensive, as the same profiles used for one direction can also be used in the reverse direction by reversing the order of magnetic readings.

Table 5.1: Median and mean of error for experiments on real data in different scenarios, for the DR based ground truth collection. (Errors are in meters.)

	Real-time median	Real-time mean	Non-Causal median	Non-Causal mean
Phone in different orientations	2.2	5.2	2.2	2.7
Test and training collected with 3 month time different	1.5	9.1	1.5	4.0
Walked only a partial path on reference data	2.8	7.2	2.8	4.2
Trace collected 1 meter on the left of reference data	6.9	12.7	3.2	6.4
Walked the path in reverse direction	1.7	3.9	1.1	2.0

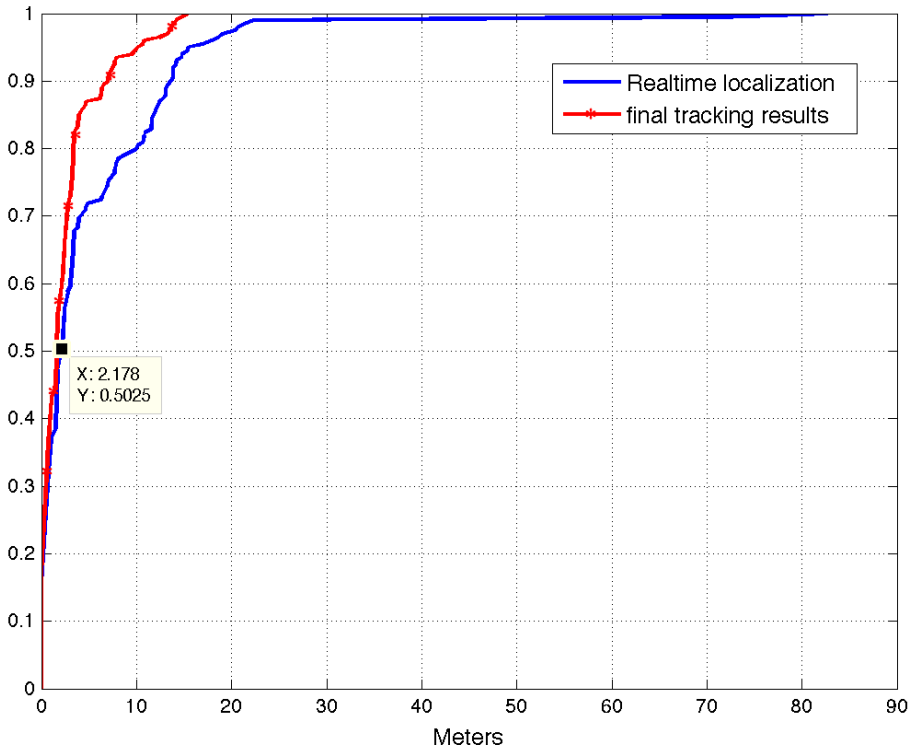


Figure 5.6: The error cdf of localization for 4 walks with different phone orientations along a 170m path.

Table 5.2: Median and mean of error for experiments on real data in different scenarios, for the timestamp based ground truth collection. (Errors are in meters.)

	Real-time median	Real-time mean	Non-Causal median	Non-Causal mean
Phone in different orientations	1.8	3.8	1.2	2.8
Walked the path in reverse direction	1.8	7.3	1.0	2.3

5.3 Using DR to improve accuracy

In the previous section, we showed that magnetic field disturbance in indoor areas can be used for localization. In this section, we explain our proposed approach for exploiting the information obtained from our DR module to improve the accuracy of the system described in Section 5.2. We use a filtering approach (which is a combination of Particle filters and the Viterbi algorithm) for the purpose of sensor fusion, to combine the information of inertial sensors with the magnetic profiles and user's movement trace. Figure 5.7 depicts the proposed adjustments to the system described in Section 5.2.

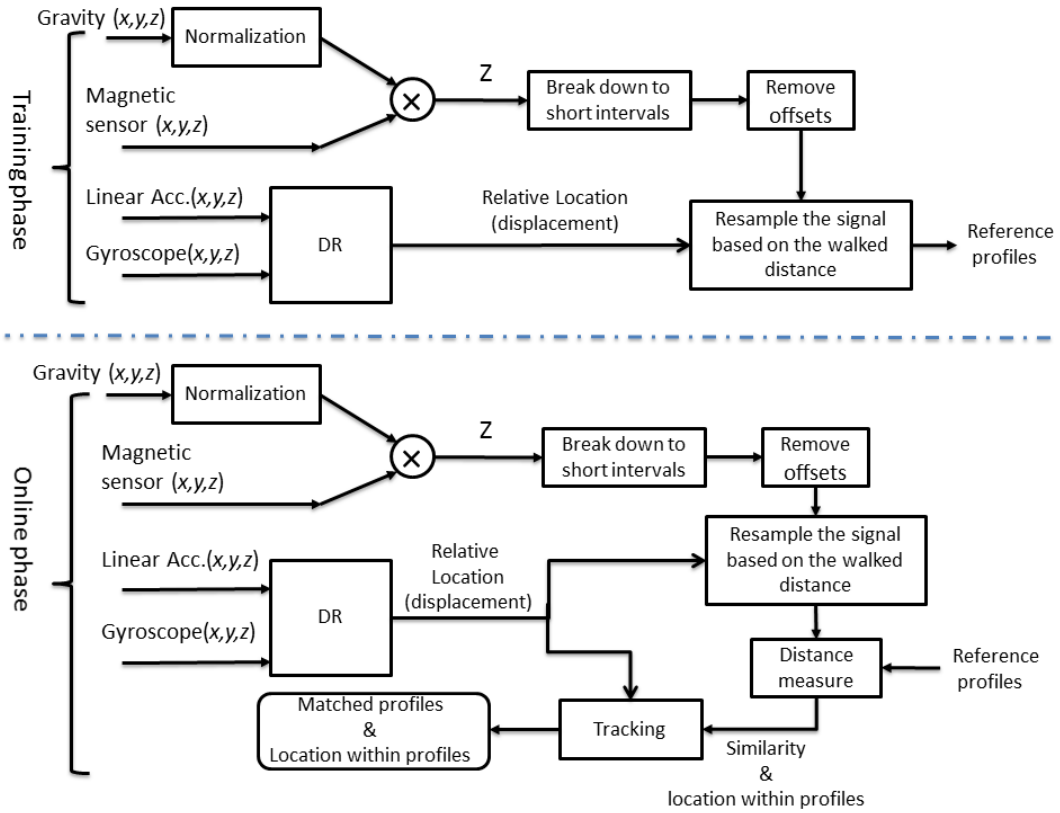


Figure 5.7: The schematic diagram of the proposed system.

5.3.1 Training phase

As can be seen in Figure 5.7, the training phase is similar to the one shown in Figure 5.1: the collected raw magnetic sensor readings along a walked path are first projected onto the world's Z axis to become independent of the pone's orientation. Next, the signals are broken into short intervals, each called a *profile* and then the mean of the signals in each profile is removed, making it offset independent. The only extra step here is resampling the magnetic sensor readings based on the output of the DR module. The resampling of the signals is done to spread the magnetic readings uniformly along the walked space, rather than the collection time of the sample. This technique, unifies sampling rate of different smartphones, as well as different walking speeds. In other words, the resamling of the readings based on the walked distance enables us to compare the signals in the covered physical domain, rather than time domain. Since the profiles are selected to be a short portion of the walked path, we assume that the user is walking with constant speed during that time, hence results in magnetic profiles that all have the same number of samples per meter.

5.3.2 Online phase

In order to match the online readings with the reference profiles, we are interested in a similarity/distance measure between two sequences of magnetic readings $\mathbf{X}, \mathbf{Y}$, that is robust to reading offset, time scale, and time shift.

The difference in the offset of the readings, as mentioned, can be compensated by removing the mean of the two sequences. Since the process of resampling the signals based on the walked distance is done on both of the training and online signals, the signals are expected to have the same scale, relaxing the need for a time-scale invariant distance measure. Therefore, instead of the DTW algorithm, we proposed two more effective similarity measures between the reference profiles and online readings, which are invariant to time shift:

1. **Similarity based on Correlation:**
   In this measure, we use the correlation coefficient between the two sequences with all possible time shifts and pick the highest found correlation coefficient as the similarity value of the two sequences.

$$sim(\mathbf{X}, \mathbf{Y}) = \max_{t \in \{1,2,\ldots,|X|+|Y|-1\}} \text{Corr}(\mathbf{X}^t_{t-|\mathbf{Y}|}, \mathbf{Y}) \times W(t), \tag{5.9}$$

   where Corr() denotes the Pearson correlation coefficient, $\mathbf{X}^t_{t-k}$ denotes a subsequences of $\mathbf{X}$ with zero padding, i.e., $(x_{t-k}, x_{t_k+1}, \ldots, x_t)$, $x_t = 0; t \leq 0$ and $t > |\mathbf{X}|$, $|\mathbf{X}|$ is the length of the sequence, and $W(t)$ is a window function [86] (also known as the tapering function). The window function is designed to remove the edge effect by applying a weight on the correlation values based on the amount of overlap. The window function we applied for this measure normalized the correlation value by the length of signal overlap. In practice, to further reduce the edge effect, we ignored the time-shifts with very small overlaps. The time shift with the highest correlation is also kept to be used later in the tracking application, when the location of the user within a profile sequence is estimated.

2. **Distance based on variance**
   In this measure, we use the variance of the difference between all possible time shifts of the two sequences and picked the shift with the lowest variance:

$$D(\mathbf{X}, \mathbf{Y}) = \min_{t \in \{1,2,\ldots,|X|+|Y|-1\}} \text{Var}(\mathbf{X}^t_{t-|\mathbf{Y}|} - \mathbf{Y}) \times W(t), \tag{5.10}$$

   where Var() denotes the variance, $\mathbf{X}^t_{t-k}$ denotes a subsequences of $\mathbf{X}$ with zero

padding, i.e., $(x_{t-k}, x_{t_k+1}, \ldots, x_t)$, $x_t = 0; t \leq 0$ and $t > |\mathbf{X}|$, $|\mathbf{X}|$ is the length of the sequence, and $W(t)$ is a window function. The window function we used for this measure normalized the variance of the sequences by the logarithm of the amount of overlapping elements. Same as correlation based similarity, in our tests, we ignored the time-shifts with the overlaps smaller than a threshold. Since the variance is independent of offset, this measure is offset independent. Therefore, when using this measure, there is no need to remove the mean of the signals in each profile.

To locate the user, at certain time intervals, we calculate the displacement of the user with respect to the previous time step via the DR and fuse the estimated current location to the similarity of the observed sequence of magnetic readings with the reference profiles. We have adopted the HMM used in Section 5.2 to fuse this information, and find the most likely profile in which user is located at each time. In order to estimate the user's location within the profiles, we apply a particle filter based method. The next subsection explains how the problem is modeled as an HMM and how the most likely location is estimated.

5.3.3 Tracking via HMM

Similar to Section 5.2, consider a user's movement trace to be broken into N smaller traces, each consisting of the user's movement, in one time-step. Define the user's location at the beginning of each time-step as the sequence $\mathbf{L} = (L_1, L_2, \ldots, L_N)$, representing the user's location during the whole trace. Note that if $\mathbf{L}$ is known, user's real-time location during each time step can be approximated via DR with high accuracy, as the time-steps are short enough to have negligible drift error.

We map the user's movement on a discrete state space, in the sense that at each starting point L_i, $i \in \{1, \ldots, N\}$, the user's location is assumed to be located on a finite set of profiles in the data-set. Note that the user can be located anywhere, while the user's starting point L_i can only lie on a set of finite line segments, i.e., each L_i is a coordinate projected on a segment in the dataset. If the number of profiles in the data-set is large enough, the user's location can be mapped to the line segments with a small projection error.

The sequence of observations along the user's walked path can also be broken into subsequences $\mathbf{X}$ and $\mathbf{Y}$, where $X_i, i \in \{1, \ldots, N\}$ is a sequence of magnetic readings during the i-th time-step, and $Y_i, i \in \{1, \ldots, N\}$ is the sequence of 2D relative displacements of the user during the i-th time-step, obtained by the DR module. Note that the relative displacements obtained from the DR do not contain absolute heading information, e.g.,

a sample displacement can be in the form of *2 meters straight (* 0° *change in heading angle)* or *2 meters to the left (*90° *counter-clock wise change in heading angle)* without knowing the actual inclination of straight heading from the North direction.

Define the set of profiles (line segments), as the set of possible latent states $Z \in \mathcal{S} = \{1, 2, \ldots, M\}$, where M is the size of the reference profile data-set. Each profile consists of a line segment with known coordinates (represented by starting point Z^s and ending point Z^e) as well as a sequence of magnetic readings along the line segment, shown by Z^{Mag}. The sequence of states on which the sequence $\mathbf{L}$ was located is defined as $\mathbf{Z} = \{Z_i\}_{i=1}^N$, where L_i lies on Z_i for all i (represented as $L_i \in Z_i, \forall i$). Note that since $\mathbf{Z}$ forms a Markov chain, it can be characterized by a prior probability π_Z and the transition probabilities

$$\Pr(z_n|z_{n-1}) = C \times \begin{cases} 1 & \text{if } D(z_n, z_{n-1}) < thrs \\ 0 & \text{otherwise.} \end{cases}, \tag{5.11}$$

where z_n represents the outcome of Z_n, $thrs$ is a constant value, selected based on the maximum possible distance a user can walk during a time-step, and D is distance between two profiles, defined to be the Euclidean distance between the center of two line segments. C is the normalization factor. According to this model, the probability of changing states is uniform between the states in a certain distance radious of the current state.

Using the notations above, the problem of interest is estimating the most probable sequence $\mathbf{L}$ based on the magnetic readings, $\mathbf{X}$, and the observations of DR, $\mathbf{Y}$. Using the Bayes rule, we have

$$\begin{aligned} \hat{\mathbf{L}} &= \arg\max_{\mathbf{L}} \Pr(\mathbf{L}|\mathbf{X}, \mathbf{Y}) \\ &= \arg\max_{\mathbf{L}} \Pr(\mathbf{L}, \mathbf{X}, \mathbf{Y}). \end{aligned}$$

Using the total probability theorem, this problem can be broken down to two sub problems: estimating the most likely sequence of profiles $\mathbf{Z}$, and the most likely location of L_i, given that L_i lies on Z_i:

$$\begin{aligned} \Pr(\mathbf{L}, \mathbf{X}, \mathbf{Y}) &= \sum_{\forall \mathbf{Z}} \Pr(\mathbf{L}, \mathbf{X}, \mathbf{Y}, \mathbf{Z}) \\ &= \sum_{\forall \mathbf{Z}} \left(\Pr(\mathbf{L}, \mathbf{X}, \mathbf{Y}|\mathbf{Z}) \Pr(\mathbf{Z})\right). \end{aligned} \tag{5.12}$$

For simplicity of explanation, assume for now that the line segments of the profiles

do not overlap. As a result, for each sequence $\mathbf{L}$, there only exists one sequence $\mathbf{Z}$, for which the term $\Pr(\mathbf{L}|\mathbf{X},\mathbf{Y},\mathbf{Z})\Pr(\mathbf{Z})$ is not equal to zero, i.e., for each $\mathbf{L}$ all the terms in the sum given (5.12) are zero, except the one that satisfies $L_i \in Z_i, \forall i$.

To calculate $\hat{\mathbf{L}}$, we can now model the problem as a hidden Markov model, with latent variables $\{Z_i\}_{i=1}^N$, and observations $\{X_i\}_{i=1}^N$, $\{Y_i\}_{i=1}^N$. Equation (5.12) can be further expanded using the product rule:

$$\begin{aligned} \Pr(\mathbf{Z}) &= \Pr(Z_1)\prod_{i=2}^{N}\Pr(Z_i|Z_{i-1}), \\ \Pr(\mathbf{L},\mathbf{X},\mathbf{Y}|\mathbf{Z}) &= \Pr(\mathbf{Y}|\mathbf{Z})\Pr(\mathbf{X}|\mathbf{Y},\mathbf{Z})\Pr(\mathbf{L}|\mathbf{X},\mathbf{Y},\mathbf{Z}). \end{aligned} \tag{5.13}$$

resulting in

$$\Pr(\mathbf{L},\mathbf{X},\mathbf{Y}) = \underbrace{\Pr(\mathbf{Z})\Pr(\mathbf{Y}|\mathbf{Z})}_{\text{inter-profile level}}\underbrace{\Pr(\mathbf{X}|\mathbf{Y},\mathbf{Z})\Pr(\mathbf{L}|\mathbf{X},\mathbf{Y},\mathbf{Z})}_{\text{intra-profile level}}, \quad L_i \in Z_i, \forall i. \tag{5.14}$$

Note that the first two terms of the equation can be paraphrased as the "profile level localization", i.e., finding the most likely sequence of visited profiles based on the DR information (inter-profile level). The latter two terms, however, aim to localize within a given profile, via DR and magnetic readings (intra-profile level).

We now model each of the probability distributions in (5.14):

- $\Pr(\mathbf{Z})$: For the transition probability between the states, we do not assume a prior information about the user's location. Therefore, as stated above we consider the transition probability to be uniformly distributed among all of the neighbours of each state, within a given radius, as shown in (5.11).

- $\Pr(\mathbf{Y}|\mathbf{Z})$: The observations $Y_i, i \in \{1,\ldots,N\}$ have a continuous probability distribution and depend on 3 parameters: from which state the observation started, to which state the observation transitioned, and what was the absolute initial heading direction in that time step. The absolute initial heading direction is estimated based on the vector connecting the starting point of the last visited two states, i.e.,

$$\theta = \angle(z_{i-1}^s - z_{i-2}^s)$$

  We model the probability of observing a displacement, based on the previous states as a 3D Guassian Random variable, stretched along the line segment representing the profile, as shown in 5.8. The probability of observation, noting the Markov

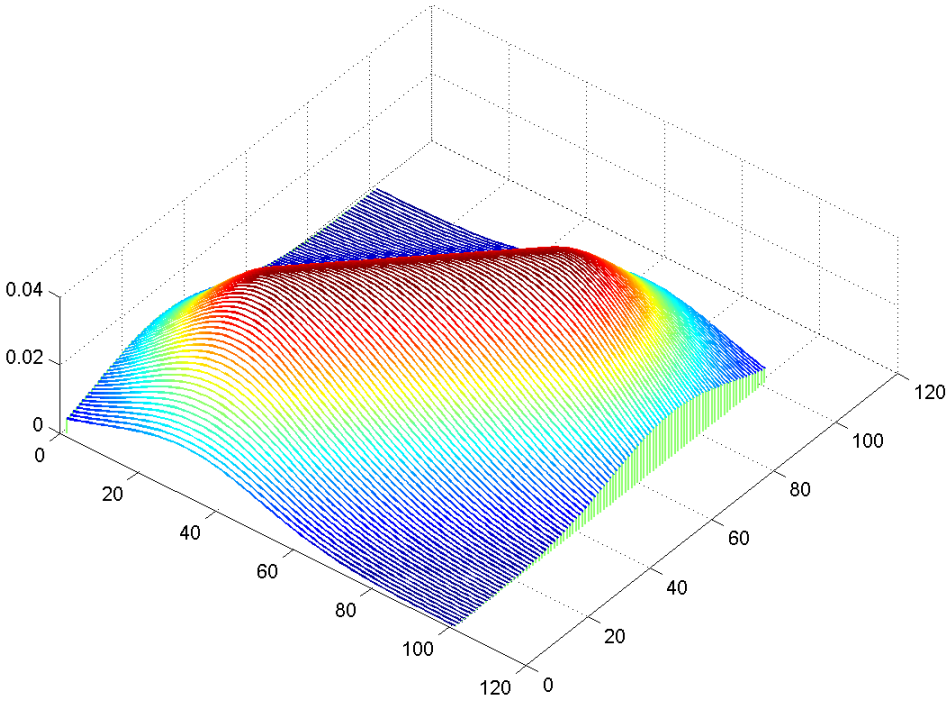


Figure 5.8: the probability distribution of displacement observation for a sample line segment.

property of the sequence of states, is calculated via:

$$\begin{aligned} \Pr(y_i|z_i, z_{i-1}, \ldots, z_1) &= \Pr(y_i|z_i, z_{i-1}, z_{i-2}) \\ &= C \times \exp\left(-\frac{d(p, z_i)}{2\sigma^2}\right), \\ p &\triangleq z_{i-1}^s + y_i \times \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cot\theta \end{pmatrix}, \end{aligned} \tag{5.15}$$

where $d(p, z_{i+1})$ is the Euclidean distance between the point p and the line segment of profile z_i, p is defined to be the estimated location of the user at the end of the i-th time-step, based on the displacement given via DR and heading direction θ. The constant C is a normalization factor. Figure 5.8 shows the probability distribution of a sample displacement observation over a sample line-segment segment.

- $\Pr(\mathbf{X}|\mathbf{Y}, \mathbf{Z})$: To calculate the probability of observing a sequence of magnetic readings given a displacement by DR and the previous visited states, similar to the training phase, we first resample the magnetic readings based on the walked dis-

tance and then model the probability based on the similarity/distance measures introduced in (5.9) and (5.10). Noting the fact that in each time-step, the observation of a magnetic sequence only depends on the states in which the observation has happened and on the displacement in the profile,

$$\begin{aligned} \Pr(x_i|z_i, z_{i-1}, \ldots, z_1, y_i, y_{i-1}, \ldots, y_1) &= \Pr(x_i|z_i, y_i) \\ &= C' \times \exp\left(-\text{Dist}(x_i, z_i)\right), \end{aligned} \quad (5.16)$$

where C' is a normalization constant and $\text{Dist}(\cdot)$ represent the distance measure used for comparing the magnetic observation x_i with the magnetic observation of profile (state) z_i. If the correlation similarity measure is used instead of distance, the negative sign in the exponent is removed.

- $\Pr(\mathbf{L}|\mathbf{X}, \mathbf{Y}, \mathbf{Z})$: The probability of location $L_i = l_i$ given the sequence of states $\mathbf{Z}$ and the observation of DR module and magnetic sensor only depends on the previous location L_{i-1} where the transition state started, the current observations of DR Y_i, the initial heading direction θ, and the magnetic sensor, X_i. The above combination, depends on the current state, as well as the last two states, i.e., $Z_i, Z_{i-1}, \text{and } Z_{i-2}$. We use an approximation of such probability based on particle filters. At each time step, we estimate the current location based on the initial heading angle θ and the displacement observed from DR. This estimation is done by distributing a number of particles located on the line-segment of Z_i, with a Gaussian distribution $\mathcal{N} = (\mu, \sigma)$, where

$$\begin{aligned} \mu &= l_{i-1} + y_i \times \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cot\theta \end{pmatrix}, \\ \sigma &= c|y_i|, \end{aligned}$$

  where $|y_i|$ is the length of the displacement and c is a constant. Note that the standard deviation of the Gaussian kernel grows with the displacement, to represent the accumulation of error in the DR module.

  The particles are then weighted by the magnetic observation, based on the similarity/distance value that is calculated at each time-shift using (5.9) or (5.10). Therefore, the most likely location at time-step i, given profile Z_i is represented by the location of the particle with the maximum weight in that profile.

To track the user along a sequence of observations, similar to Section 5.2, we apply the Viterbi algorithm, with the metric obtained from (5.14) and the probability models

described above. In each time step, the probability of being located in each profile is calculated and then, within each profile, using the particle filter based method, we estimate the user's location based on particle with the highest weight. Therefore, the location within the floor plan is estimated to be the location with the highest profile probability and the highest particle weight, i.e., the location that maximizes (5.12).

Implementation and results

The implementation and testing of the system was done in a way similar to the system described in Section 5.2, i.e., the data was collected using an Android application and was analyzed in Matlab. To calculate the displacements, we used the Matlab implementation of our DR module, described in Chapter 4.

The ground truth data was collected by setting several checkpoints in the area and measuring the time each checkpoint was passed. The magnetic and DR recordings were collected and time stamped along the walks.

For generating the profile data-set, we chose 7 steps (about 5.2 meters) of walking for each profile, with 3 steps (2.2 meters) overlap between the neighbouring profiles. The magnetic samples were resampled based on the distance measured by DR, so that all of the traces have 33 samples per meters. In the operation phase, the time-steps for updating the observations and estimating the new location was selected to be every 3.5 seconds. To measure the similarity of the magnetic readings, we used the distance based on the variance given in (5.10) since it was experimentally observed to be more effective.

We did two general tests on a single floor of the Bahen Center for Information Technology. For the first set of tests, we used the same data collected for the previous system, i.e., over a full path of about 170 meters. Figure 5.9 represents the CDF of the error using the new enhanced system. As can be seen, the accuracy is significantly more than the version without DR, shown in Section 5.2. This improvement in accuracy confirms the proposed model can successfully exploit the DR information to locate the user within the walking path, while the magnetic readings allow for quick convergence to the correct location in real-time localization case. The mean and median of error for this experiment were about 0.65 and 0.85 meters for both of real-time tracking and final tracking (allowing back-tracing through the optimal path), respectively. Note that in this case, the traces that were collected with constantly moving the device in arbitrary orientations could not be used, due to the unrealistic error they introduce in the step count and heading direction.

The seconds set of tests evaluated the system over a longer path that covered almost

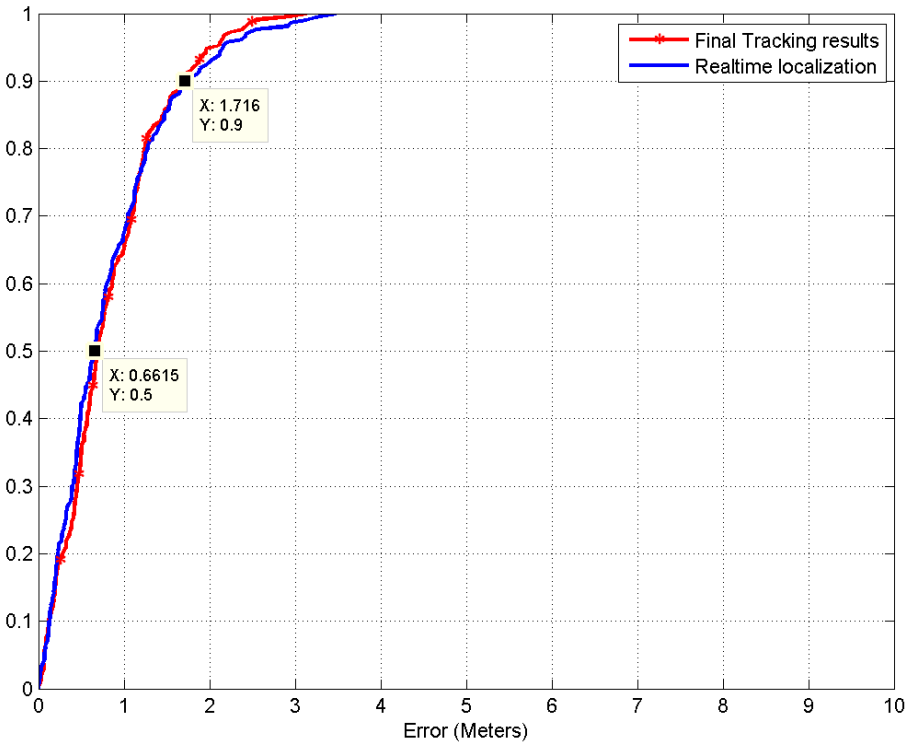


Figure 5.9: The error cdf of localization for 3 cross validated walks with different phone orientations along a 170m path, using DR to improve accuracy.

all of the walkable areas in the floor, with length of about 300 meters. The aim for collecting the second data-set was to assess the uniqueness of the collected traces over larger data-sets. Figure 5.10 shows the floor plan as well as the trace used for training.

The results of the experiments over the big data-set indicated that the rate of convergence notably decreases as the size of the data-set grows, with a higher chance of no convergence. Figure 5.12 depicts the localization error throughout the time of walking for a sample trace. It can be seen that in the real-time tracking case, the system has taken about 40 seconds to converge, while the final tracking result has been able to successfully trace back to the correct location. The median and mean results for several testing over the 300 meter trace were 1.4 and 8.4 meters respectively for the real-time tracking, and 1.1 and 3.3 meters for the final tracking with back-tracing through the optimal path. The error CDF of the tests is shown in Figure 5.11.

The reason for the degradation in accuracy in comparison with the smaller data-set, specifically for the real-time case, is believed to be the fact that as the size of the trace grows, the number of magnetic profiles grows, so that in each time-step the observations

Figure 5.10: The testbed floor plan with the coverage of 2nd data-set (path length is about 300 meters).

must be compared and classified to a bigger set of profiles. Since the readings along a profile are highly correlated, there are limited features that make each magnetic profile unique and hence, chances of correct classification reduces as the number of profiles grow. As a result, the tracking system keeps receiving a sequence of DR module displacements that cannot distinguish between different straight hallways and a sequence of magnetic profiles that are less likely to be uniquely distinguishable, until a unique feature (such as a turn according to DR or a unique magnetic pattern) in the path is observed. Such behavior leads to a large delay in converging to the correct location with a higher chance of no convergence at all.

To make the magnetic profiles more unique over a larger trace, one approach can be increasing the profile lengths. However, in order for the bigger profiles to be more effective, the lengths of observations in the online phase must also increase, leading to larger delay in updating the location of the user. Therefore, it can be concluded that although the results have shown to be promising over small or medium data-sets, it is expected that the proposed approach, in its current settings, may not scale well with larger indoor areas.

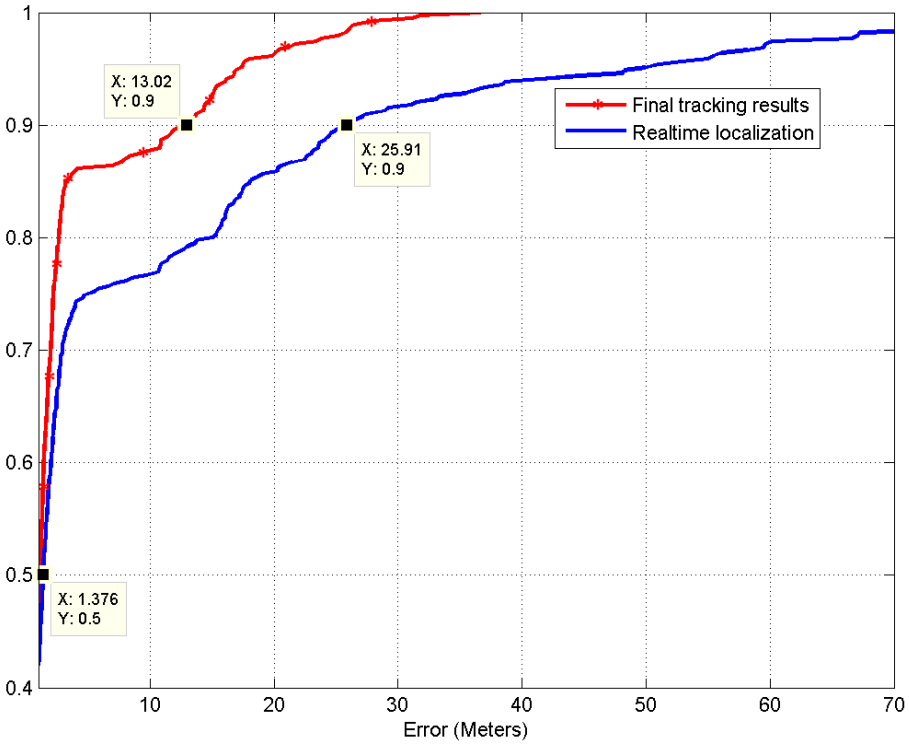


Figure 5.11: The error cdf of localization for 3 cross validated walks with different phone orientations along a 300m path, using the DR to improve accuracy. The high 90 percentile in comparison with median is due to the late convergence of the traces to the correct path.

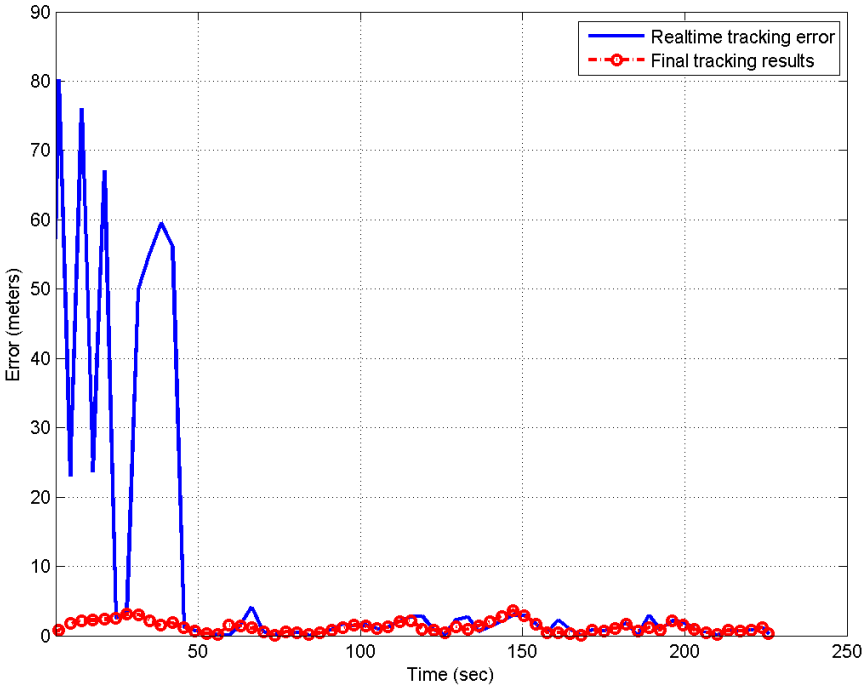


Figure 5.12: The plot of error in each time step of walking a trace (path length is about 300 meters).

Chapter 6

Conclusion and Future Work

In this work, we investigated the problem of effort-free radio map generation for RSS-based indoor localization. Unlike the traditional techniques, our method does not require any active user intervention during the training. The system automatically collects data from inertial sensors along with Wi-Fi signals and builds a radio map by merging the collected data from the users, engaged in their daily activities.

We modeled the problem as a matching of weighted undirected graphs and proposed two different algorithms with different characteristics, to effectively solve the problem. For the first matching algorithm, K-best fits, we defined a similarity measure to capture the global topological properties of graphs and provided a technique to efficiently calculate it, while theoretically proving the correctness of our efficient algorithm. For the second matching algorithm, we approached the problem in a probabilistic fashion, while exploiting the local characteristics of nodes and edges in the graphs. The proposed system was implemented and tested on both simulation and real data and was shown to be highly accurate.

We also investigated a new method for indoor localization by exploiting magnetic sensors on smartphones using a combination of geomagnetic signals and inertial sensors, with no need for extra infrastructure. The system was shown experimentally to have promising results, while also benefiting from many relaxed constraints of practical smartphone usage, such as the ability of quick dataset collection with low-cost sensors used in smartphones, arbitrary device orientation, and no need for extra infrastructure.

We used sensor fusion via hidden Markov models to further improve the accuracy and stability of the system. The implemented system was tested on real data and the results were presented.

This work can be extended from several different aspects. Here, we briefly present an itemized overview of the proposed future works:

- The crowdsourcing system is designed to work in a single floor of a building. It can be merged with global positioning systems as well as a floor detection sub-system to act completely transparent both in indoors and outdoors, without the need for a user to select his/her current building or floor number.

- In the current implementation, due to the rectangular modeling of movements and floor plans, wide pathways such a big halls or atriums are not accurately modeled. One of the topics of future study is improving the modeling of floor plans to better represent such indoor areas.

- The proposed magnetic based indoor localization system is in early stages. There still exists a gap between the proposed system and a practical system. Such practicality limitations can be studied and relaxed. To name a few, improving the tracking system to converge to the user's location faster and more reliably, more effective usage of magnetic readings, and improving the accuracy and stability are a few expansion points of the magnetic system.

- The nature of magnetic fields and the fact that they can have spatial rapid changes due to disturbance, makes the usage of our crowdsourced based solution for Wi-Fi signals seem impractical for magnetic field signals. The possibility of training the magnetic based system using crowdsourced data is certainly one of the interesting topics for further research.

Bibliography

[1] Android documentation of motion sensors. `http://developer.android.com/guide/topics/sensors/sensors_motion.html`. Accessed: 2010-09-30.

[2] Michael Angermann, Martin Frassl, Marek Doniec, Brian J Julian, and Patrick Robertson. Characterization of the indoor magnetic field for applications in localization and mapping. In *International Conference on Indoor Positioning and Indoor Navigation*, volume 13, page 15th, 2012.

[3] Paramvir Bahl and Venkata N Padmanabhan. RADAR: an in-building RF-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 775–784 vol.2, 2000.

[4] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):509–522, Apr 2002.

[5] Vincent D Blondel, Anahí Gajardo, Maureen Heymans, Pierre Senellart, and Paul Van Dooren. A measure of similarity between graph vertices: Applications to synonym extraction and web searching. *SIAM review*, 46(4):647–666, 2004.

[6] Philipp Bolliger, Kurt Partridge, Maurice Chu, and Marc Langheinrich. Improving location fingerprinting through motion detection and asynchronous interval labeling. In *Location and Context Awareness*, pages 37–51. Springer, 2009.

[7] John Adrian Bondy and Uppaluri Siva Ramachandra Murty. *Graph theory with applications*, volume 290. Macmillan London, 1976.

[8] Prosenjit Bose, William Lenhart, and Giuseppe Liotta. Characterizing proximity trees. *Algorithmica*, 16(1):83–110, 1996.

[9] Horst Bunke, Xiaoyi Jiang, and Abraham Kandel. On the minimum common supergraph of two graphs. *Computing*, 65(1):13–25, 2000.

[10] Michael J Caruso. Applications of magnetic sensors for low cost compass systems. In *Position Location and Navigation Symposium, IEEE 2000*, pages 177–184. IEEE, 2000.

[11] Seong Yun Cho and Chan Gook Park. Mems based pedestrian navigation system. *Journal of Navigation*, 59(01):135–153, 2006.

[12] Rolf Christensen and Nikolaj Fogh. Inertial navigation system. Technical report, Aalborg university, Tech. Rep. 08gr1030a, 2008.

[13] Haili Chui and Anand Rangarajan. A new algorithm for non-rigid point matching. In *IEEE Conference on Computer Vision and Pattern Recognition, 2000. Proceedings.*, volume 2, pages 44–51. IEEE, 2000.

[14] William S Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American statistical association*, 74(368):829–836, 1979.

[15] Donatello Conte, Pasquale Foggia, Carlo Sansone, and Mario Vento. Thirty years of graph matching in pattern recognition. *International journal of pattern recognition and artificial intelligence*, 18(03):265–298, 2004.

[16] Jonathan D Courtney and Anil K Jain. Mobile robot localization via classification of multisensor maps. In *IEEE International Conference on Robotics and Automation, 1994. Proceedings., 1994*, pages 1672–1678 vol.2, 1994.

[17] Trevor F Cox and Michael AA Cox. *Multidimensional scaling*. CRC Press, 2010.

[18] Peter Eades and Sue Whitesides. The realization problem for euclidean minimum spanning trees is NP-hard. *Algorithmica*, 16(1):60–82, 1996.

[19] Jack Edmonds and Richard M Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, 19(2):248–264, 1972.

[20] Tolga Eren, David Kiyoshi Goldenberg, Walter Whiteley, Yang Richard Yang, A Stephen Morse, Brian DO Anderson, and PN Belhumeur. Rigidity, computation, and randomization in network localization. In *INFOCOM 2004. Twenty-third Annu. Joint Conf. of the IEEE Comput. and Commun. Societies*, volume 4, pages 2673–2684 vol.4, 2004.

[21] Lei Fang, Panos J Antsaklis, Luis A. Montestruque, M Brett McMickell, Michael Lemmon, Yashan sun, Hui Fang, Ionnis Koutroulis, Martin Haenggi, Min Xie, and Xiaojuan Xie. Design of a wireless assisted pedestrian dead reckoning system - the NavMote experience. *IEEE Trans. Instrum. Meas.*, 54(6):2342–2358, Dec 2005.

[22] Chen Feng, Wain Sy Anthea Au, Shahrokh Valaee, and Zhenhui Tan. Received-signal-strength-based indoor positioning using compressive sensing. *IEEE Trans. Mobile Comput.*, 11(12):1983–1993, 2012.

[23] Brian Ferris, Dieter Fox, and Neil D Lawrence. WiFi-SLAM using gaussian process latent variable models. In *IJCAI*, volume 7, pages 2480–2485, 2007.

[24] Scott Fortin. The graph isomorphism problem. Technical report, Technical Report 96-20, University of Alberta, Edomonton, Alberta, Canada, 1996.

[25] Linton C Freeman. Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239, 1979.

[26] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[27] Mohammad Heidari, Nayef Ali Alsindi, and Kaveh Pahlavan. UDP identification and error mitigation in ToA-based indoor localization systems using neural network architecture. *IEEE Trans. Wireless Commun.*, 8(7):3597–3607, 2009.

[28] Hendrik Hellmers, Abdelmoumen Norrdine, Jorg Blankenbach, and Andreas Eichhorn. An IMU/magnetometer-based indoor positioning system using Kalman filtering. In *Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on*, pages 1–9. IEEE, 2013.

[29] Maureen Heymans and Ambuj K Singh. Deriving phylogenetic trees from the similarity analysis of metabolic pathways. *Bioinformatics*, 19(suppl 1):i138–i146, 2003.

[30] Chengkai Huang, Gong Zhang, Zhuqing Jiang, Chao Li, Yupeng Wang, and Xueyang Wang. Smartphone-based indoor position and orientation tracking fusing inertial and magnetic sensing. In *Wireless Personal Multimedia Communications (WPMC), 2014 International Symposium on*, pages 215–220. IEEE, 2014.

[31] H.-J. Jang, J.W. Kim, and D.H. Hwang. Robust step detection method for pedestrian navigation systems. *Electronics Letters*, 43(14):–, 2007.

[32] Peggy L Jenkins, Thomas J Phillips, Elliot J Mulberg, and Steve P Hui. Activity patterns of californians: use of and proximity to indoor pollutant sources. *Atmospheric Environment. Part A. General Topics*, 26(12):2141–2148, 1992.

[33] Yunye Jin, M. Motani, Wee-Seng Soh, and Juanjuan Zhang. Sparsetrack: Enhancing indoor pedestrian tracking with sparse infrastructure support. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9, 2010.

[34] Soo-Yong Jung, Swook Hann, and Chang-Soo Park. TDOA-based optical wireless indoor localization using LED ceiling lamps. *Consumer Electronics, IEEE Transactions on*, 57(4):1592–1597, 2011.

[35] M. Kanaan and K. Pahlavan. A comparison of wireless geolocation algorithms in the indoor environment. In *Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE*, volume 1, pages 177–182 Vol.1, 2004.

[36] Songkran Kantawong. Development of fire evacuation path selective using adaptive routing algorithms and rfid traffic cone-based observation with shadowing method. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2012 9th International Conference on*, pages 1–4, 2012.

[37] Mohammed Khider, Patrick Robertson, Martin Frassl, Michael Angermann, Luigi Bruno, Maria Garcia Puyol, Estefania Munoz Diaz, and Oliver Heirich. Characterization of planar-intensity based heading likelihood functions in magnetically disturbed indoor environments. In *Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on*, pages 1–10. IEEE, 2013.

[38] Seong-Eun Kim, Yong Kim, Jihyun Yoon, and Eung Sun Kim. Indoor positioning system using geomagnetic anomalies for smartphones. In *Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on*, pages 1–5. IEEE, 2012.

[39] Yungeun Kim, Yohan Chon, and Hojung Cha. Smartphone-based collaborative and autonomous radio fingerprinting. *IEEE Trans. Syst., Man, Cybern. C*, 42(1):112–122, Jan 2012.

[40] Mikkel Baun Kjærgaard. A taxonomy for radio location fingerprinting. In *Location- and Context-Awareness*, pages 139–156. Springer, 2007.

[41] Jon M Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[42] Harold W Kuhn. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[43] Woong Kwon, Kyung-Shik Roh, and Hak-Kyung Sung. Particle filter-based heading estimation using magnetic compasses for mobile robot navigation. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2705–2712. IEEE, 2006.

[44] Quentin Ladetto. On foot navigation: continuous step calibration using both complementary recursive prediction and adaptive Kalman filtering. In *Proceedings of ION GPS*, volume 2000, pages 1735–1740, 2000.

[45] JA Leech, K Wilby, E McMullen, and K Laporte. The Canadian human activity pattern survey: report of methods and population surveyed. *Chronic Diseases in Canada*, 17(3-4):118–123, 1995.

[46] Charles E Leiserson, Ronald L Rivest, Clifford Stein, and Thomas H Cormen. *Introduction to algorithms*. The MIT press, 2001.

[47] Robert W Levi and Thomas Judd. Dead reckoning navigational system using accelerometer to measure foot impacts, December 10 1996. US Patent 5,583,776.

[48] Binghao Li, Thomas Gallagher, Andrew G Dempster, and Chris Rizos. How feasible is the use of magnetic field alone for indoor positioning? In *International Conference on Indoor Positioning and Indoor Navigation*, volume 13, page 15th, 2012.

[49] Xinrong Li, Kaveh Pahlavan, Matti Latva-aho, and Mika Ylianttila. Comparison of indoor geolocation methods in DSSS and OFDM wireless LAN systems. In *Vehicular Technology Conference, 2000. IEEE-VTS Fall VTC 2000. 52nd*, volume 6, pages 3015–3020 vol.6, 2000.

[50] Hui Liu, Houshang Darabi, Pat Banerjee, and Jing Liu. Survey of wireless indoor positioning techniques and systems. *IEEE Trans. Syst., Man, Cybern. C*, 37(6):1067–1080, 2007.

[51] Shaoshuai Liu, Haiyong Luo, and Shihong Zou. A low-cost and accurate indoor localization algorithm using label propagation based semi-supervised learning. In *Mobile Ad-hoc and Sensor Networks, 2009. MSN '09. 5th International Conference on*, pages 108–111, 2009.

[52] Piotr Mirowski, Tin Kam Ho, Saehoon Yi, and Michael MacDonald. Signalslam: simultaneous localization and mapping with mixed wifi, bluetooth, LTE and magnetic signals. In *Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on*, pages 1–10. IEEE, 2013.

[53] David Moore, John Leonard, Daniela Rus, and Seth Teller. Robust distributed network localization with noisy range measurements. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 50–61. ACM, 2004.

[54] Meinard Müller. Dynamic time warping. *Information retrieval for music and motion*, pages 69–84, 2007.

[55] Andriy Myronenko and Xubo Song. Point set registration: Coherent point drift. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(12):2262–2275, 2010.

[56] Jason Orlosky, Takumi Toyama, Daniel Sonntag, Andras Sarkany, and Andras Lorincz. On-body multi-input indoor localization for dynamic emergency scenarios: fusion of magnetic tracking and optical character recognition with mixed-reality display. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on*, pages 320–325. IEEE, 2014.

[57] Robin Wentao Ouyang, Albert Kai-Sun Wong, Chin-Tau Lea, and Mung Chiang. Indoor location estimation with reduced calibration exploiting unlabeled data via hybrid generative/discriminative learning. *IEEE Trans. Mobile Comput.*, 11(11):1613–1626, 2012.

[58] James Pinchin, Chris Hide, and Tyler Moore. A particle filter approach to indoor navigation using a foot mounted inertial navigation system and heuristic heading information. In *Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on*, pages 1–10, 2012.

[59] André Pomp, Dipl-Inform JóAgila Bitsch Link, Ing Klaus Wehrle, and Bernhard Rumpe. Smartphone-based indoor mapping. 2012.

[60] Vahid Pourahmadi and Shahrokh Valaee. Indoor positioning and distance-aware graph-based semi-supervised learning method. In *Global Communications Conference (GLOBECOM), 2012 IEEE*, pages 315–320, 2012.

[61] Phongsak Prasithsangaree, Prashant Krishnamurthy, and Panos K Chrysanthis. On indoor position location with wireless lans. In *Personal, Indoor and Mobile Radio*

Communications, 2002. The 13th IEEE International Symposium on, volume 2, pages 720–724. IEEE, 2002.

[62] Nissanka B Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket location-support system. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 32–43. ACM, 2000.

[63] Xiaoning Qian and Byung-Jun Yoon. Shape matching based on graph alignment using hidden markov models. In *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages 934–937, March 2010.

[64] Anshul Rai, Krishna Kant Chintalapudi, Venkata N Padmanabhan, and Rijurekha Sen. Zee: Zero-effort crowdsourcing for indoor localization. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pages 293–304. ACM, 2012.

[65] Cliff Randell, C. Djiallis, and H. Muller. Personal position measurement using dead reckoning. In *Wearable Computers, 2003. Proceedings. Seventh IEEE International Symposium on*, pages 166–173, 2003.

[66] John W Raymond, Eleanor J Gardiner, and Peter Willett. Rascal: Calculation of graph similarity using maximum common edge subgraphs. *The Computer Journal*, 45(6):631–644, 2002.

[67] Patrick Robertson, Michael Angermann, and Bernhard Krach. Simultaneous localization and mapping for pedestrians using only foot-mounted inertial sensors. In *Proceedings of the 11th international conference on Ubiquitous computing*, pages 93–96. ACM, 2009.

[68] James B Saxe. Embeddability of weighted graphs in k-space is strongly NP-hard. *17th Allerton Conf. Commun. Control Comput.*, pages 480–489, 1979.

[69] Abhinav Saxena and Maciej Zawodniok. Indoor positioning system using geomagnetic field. In *Instrumentation and Measurement Technology Conference (I2MTC) Proceedings, 2014 IEEE International*, pages 572–577. IEEE, 2014.

[70] Shervin Shahidi and Shahrokh. Valaee. CBIL: Crowdsourcing based indoor localization via graph matching. *Submitted to IEEE Trans. Mobile Comput.*

[71] Shervin Shahidi and Shahrokh Valaee. Graph matching for crowdsourced data in mobile sensor networks. In *Signal Processing Advances in Wireless Communications (SPAWC), 2014 IEEE 15th International Workshop on*, pages 414–418, June 2014.

[72] Shervin Shahidi and Shahrokh Valaee. GIPSy: Geomagnetic indoor positioning system for smartphones. In *Indoor Positioning and Indoor Navigation (IPIN), 2015 International Conference on*, pages 1–7. IEEE, 2015.

[73] Shervin Shahidi and Shahrokh Valaee. Hidden markov model based graph matching for calibration of localization maps. In *Communications (ICC), 2015 IEEE International Conference on*, pages 4606–4611. IEEE, 2015.

[74] Yi Shang and W. Ruml. Improved MDS-based localization. In *INFOCOM 2004. Twenty-third Annu. Joint Conf. of the IEEE Comput. and Commun. Societies*, volume 4, pages 2640–2651 vol.4, 2004.

[75] Larry S Shapiro and J Michael Brady. Feature-based correspondence: an eigenvector approach. *Image and vision computing*, 10(5):283–288, 1992.

[76] Guobin Shen, Zhuo Chen, Peichao Zhang, Thomas Moscibroda, and Yongguang Zhang. Walkie-markie: indoor pathway mapping made easy. In *Proc. of USENIX NSDI*, 2013.

[77] Hyojeong Shin and Hojung Cha. Wi-Fi fingerprint-based topological map building for indoor user tracking. In *Int. Conf. on Embedded and Real-Time Computing Syst. and Applicat. (RTCSA), 2010 IEEE 16th*, pages 105–113, Aug 2010.

[78] Hyojeong Shin, Yohan Chon, and Hojung Cha. Unsupervised construction of an indoor floor plan using a smartphone. *IEEE Trans. Syst., Man, Cybern. C*, 42(6):889–898, Nov 2012.

[79] David Simchi-Levi, Xin Chen, and Julien Bramel. Convexity and supermodularity. In *The Logic of Logistics*, Springer Series in Operations Research, pages 13–32. Springer New York, 2005.

[80] Yueming Song and HongYi Yu. A rss based indoor tracking algorithm via particle filter and probability distribution. In *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on*, pages 1–4, 2008.

[81] Petre Stoica and Randolph L Moses. *Introduction to spectral analysis*, volume 1. Prentice hall New Jersey:, 1997.

[82] William Storms, Jeremiah Shockley, and John Raquet. Magnetic field navigation in an indoor environment. In *Ubiquitous Positioning Indoor Navigation and Location Based Service (UPINLBS), 2010*, pages 1–10. IEEE, 2010.

[83] S Suksakulchai, S Thongchai, DM Wilkes, and K Kawamura. Mobile robot localization using an electronic compass for corridor environment. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, volume 5, pages 3354–3359. IEEE, 2000.

[84] Hari Sundar, Deborah Silver, Nikhil Gagvani, and Sven Dickinson. Skeleton based shape matching and retrieval. In *Shape Modeling International, 2003*, pages 130–139, May 2003.

[85] Stephen P Tarzia, Peter A Dinda, Robert P Dick, and Gokhan Memik. Indoor localization without infrastructure using the acoustic background spectrum. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 155–168. ACM, 2011.

[86] Thomas Theußl, Helwig Hauser, and Eduard Gröller. Mastering windows: Improving reconstruction. In *Proceedings of the 2000 IEEE symposium on Volume visualization*, pages 101–108. ACM, 2000.

[87] Anneleen Van Nieuwenhuyse, Lieven De Strycker, Nobby Stevens, J-P Goemaere, and Bart Nauwelaers. Analysis of the realistic resolution with angle of arrival for indoor positioning. In *Next Generation Mobile Applications, Services and Technologies (NGMAST), 2011 5th International Conference on*, pages 200–205, 2011.

[88] He Wang, Souvik Sen, Ahmed Elgohary, Moustafa Farid, Moustafa Youssef, and Romit Roy Choudhury. No need to war-drive: unsupervised indoor localization. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 197–210. ACM, 2012.

[89] Hui Wang, Henning Lenz, Andrei Szabo, Joachim Bamberger, and Uwe D Hanebeck. WLAN-based pedestrian tracking using particle filters and low-cost MEMS sensors. In *Positioning, Navigation and Communication, 2007. WPNC '07. 4th Workshop on*, pages 1–7, 2007.

[90] Douglas Brent West. *Introduction to graph theory*, volume 2. Prentice hall Englewood Cliffs, 2001.

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

[92] Oliver Woodman and Robert Harle. RF-based initialisation for inertial pedestrian tracking. In *Pervasive Computing*, pages 238–255. Springer, 2009.

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

[94] Chenshu Wu, Zheng Yang, and Yunhao Liu. Smartphones based crowdsourcing for indoor localization. *IEEE Trans. Mobile Comput.*, 14(2):444–457, Feb 2015.

[95] Chenshu Wu, Zheng Yang, Yunhao Liu, and Wei Xi. WILL: Wireless indoor localization without site survey. *IEEE Trans. Parallel Distrib. Syst.*, 24(4):839–848, April 2013.

[96] Zheng Yang, Chenshu Wu, and Yunhao Liu. Locating in fingerprint space: wireless indoor localization with little human intervention. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pages 269–280. ACM, 2012.

[97] Moustafa Youssef and Ashok Agrawala. The Horus WLAN location determination system. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 205–218. ACM, 2005.

[98] Laura Zager. *Graph similarity and matching*. PhD thesis, Massachusetts Institute of Technology, 2005.

[99] Viktor N Zemlyachenko, Nickolay M Korneenko, and Regina I Tyshkevich. Graph isomorphism problem. *Journal of Soviet Mathematics*, 29(4):1426–1481, 1985.