Spatial model-aided indoor tracking

Xu. Weilin

Master of Science Thesis

TU Delft Delft University of Technology

Geomatics

Spatial model-aided indoor tracking

Master of Science Thesis

For the degree of Master of Science in Geomatics at Delft University of Technology

Xu. Weilin
Student number: 4246004
Email: W.Xu@student.tudelft.nl
Address: Zusterlaan 252, Delft

June 24, 2014

Main tutor:
Ph.D.Liu Liu
L.Liu-1@tudelft.nl
Graduation professor:
Ass.Prof.Dr.S.Zlatanova
s.zlatanova@tudelft.nl

Advisor:
ir.Wouter Penard
w.penard@cgi.com
Co-reader:
Ph.D.Pirouz Nourian
P.Nourian@tudelft.nl

Faculty of Architecture and the Built Environment (Arch) · Delft University of Technology

The work in this thesis was supported by CGI. Their cooperation is hereby gratefully acknowledged.

Abstract

In order to address the problem of indoor pedestrian tracking, this thesis reports a research on spatial models' ability to reduce tracking error of a WiFi positioning system. There are three main objectives in this research. First, it is to build a suitable spatial model for tracking. Second, it is to develop a tracking algorithm that can make full use of the spatial model. Last, the tracking algorithm should be tested in a live environment.

Based on literature study, a grid-based spatial model is chosen to be built because it is easy to design and maintain, has high flexibility, has accurate location data and is powerful for computation. The thesis explores various geometric, topological and semantic features of the grid model and select out the most useful features upon tracking purposes. Among geometric features, coordinate, buffer, orientation vector and Euclidean distance are used. Among semantic features, space, obstacle, and door are employed. Among topological features, the difference between straight-line distance and shortest path distance is chosen.

We develop the tracking algorithm combining multiple tracking techniques. In addition to the spatial model and WiFi positioning system, the algorithm also includes magnetometers and grid filters. The former one measures the orientation of a pedestrian. The latter one allows integrating all selected features of the grid model with the measurements from both the WiFi positioning system and the magnetometer to compute the location recursively.

To test the algorithm's performance, we built a tracking system with database, web service and mobile client. Several experiments are carried out using the system in a real environment. The experiment results show that the algorithm is able to determine locations at reasonable places (in the correct space, outside obstacles and connected to the previous location) and derive the accurate moving direction of a pedestrian.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisor Ph.D. Liu Liu for his support, guidance and advice throughout this project, Ass. Prof. Dr. S. Zlatanova and Ph.D. Pirouz Nourian for their valuable opinions to improve this work. I also thank Wouter Penard and the Company CGI Group for their valuable input and help on the thesis.

I owe my special thanks to the researchers of Wuhan University for the WiFi positioning system provided by them, especially Taizhou Li and Jianjan wang for their help on implementing the WiFi positioning system. I would also like to thank my friend hailin Li for his help on collecting data as well as all those who have supported me during the course of this master degree.

Delft, University of Technology Xu. Weilin
June 24, 2014

Chapter 1

Introduction

1-1 Context

The wide deployment of wireless networks promotes location-based services (LBS) on mobile devices. The precise location is the foundation of most LBS applications. For example, navigation, monitoring and emergency services, people and assets tracking, mobile advertising and pedestrian moving pattern-based analysis, etc. Hence, there is an increasing interest in developing effective positioning and tracking systems.

Global Positioning System (GPS) is the widely used in outdoor environment, but it fails indoor due to the poor signals of satellites inside buildings. Therefore, many techniques (e.g. Bluetooth, WiFi, RFID, GSM) are considered as alternatives for localization in indoor environments. The boom of WiFi networks over the past few years let the WiFi localization technology obtain popularity rapidly. The cost of implementation of this technology is relatively low because it makes use of the existing infrastructures and does not rely on extra hardware except the mobile devices of users. Also, the number of private and public access points (APs) is increasing in the cities and the WiFi coverage is getting higher, thus higher precision can be provided [1].

Based on WiFi network, the location can be calculated through Cell identity, Time of Arrival(TOA), Time Difference of Arrival(TDOA), Angle of Arrival(AOA) or signal strength based methods [2]. Cell identity simply matches the target's location with its connected AP. This method does not require complex conditions such as time synchronization and multiple APs, but it has very low accuracy due to its simplicity. The TOA, TDOA and AOA methods receptively measure the travel time from a transmitter to a receiver, the difference of travel time from several transmitters to a receiver and the angle from a transmitter to a receiver. Multi-path effects of indoor environment influence these three approaches greatly. Moreover, the TOA and TDOA require accurate time synchronization between transmitters and receivers. Hence, they are not optimal solutions for indoor positioning. The signal strength based approaches utilizes the distance-to-signal-strength relationship to estimate location of a mobile device. An example of the relation between signal strength and distance is given

in figure 1-1. Because it is difficult to represent the relation between the signal strength and distance using simple analytical function [3], the fingerprinting method is often employed for location determination. This method measures the received signal strength (RSS) from Aps in two phases: off-line phase where the signal strength at various predefined locations is mapped, and on-line phase where the possibility of a location is estimated by comparing the signal strength at this location with the signal strength map obtained in the off-line phase. For this thesis, the WiFi fingerprinting method is chosen since it can provide acceptable localization accuracy and that it is not too complicated to be implemented.

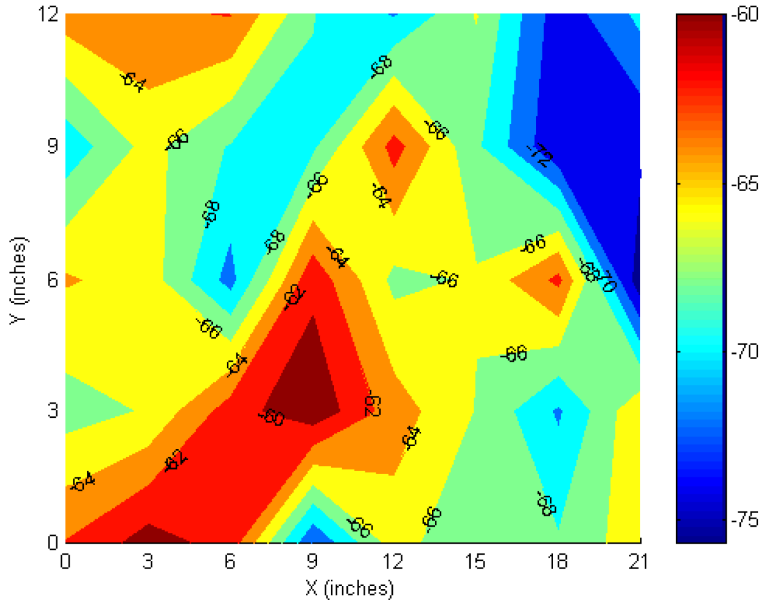


Figure 1-1: Relation between signal strength and distance [3]

Spatial model is employed for various indoor applications. By far the most common application discussed in the literature is navigation, however there is also research on using spatial model for robotic exploration [4, 5], indoor tracking and spatial analysis [6]. Some examples of use of spatial models for tracking are given below. To facilitate tracking application, Jensen, Lu and Yang [7] utilize topological graph of indoor space to increase tracking accuracy and Lee and Chen [8] use a floor model including the accessibility of different zones of the floor to deduce a user current location. Spatial models can be categorized based on their representations, i.e. geometry, topology and semantics. Pure geometrical, topological or semantic model is rarely used, instead hybrid models with various levels-of-abstraction [9]. Two typical spatial models are network model and grid model. The former one uses vertex-edge representation and has an irregular decomposition of the space. The latter one represents the indoor space using a group of regular cells. By reviewing literature, we find that there is very little research on using complete features of indoor space (geometry, topology and semantics) for tracking. In the context of this thesis, geometric features are shapes and coordinates of indoor space and structures, topological features refer to spatial relations of geometric primitives, which can be adjacency or connectivity relation, and semantics is defined as information about high-level entities such as rooms, corridors, doors, and their relationships. This thesis builds a hybrid spatial model including the three types of features and uses this model to decrease tracking errors.

1-2 Challenges in indoor tracking

There are some challenges in WiFi fingerprinting technology. First this method suffers the electromagnetic interference. The WiFi signal frequency is in the 2.4 GHz public band which is also used by many other wireless signal transmitters, e.g., phone, bluetooth, microwave oven and video devices. In the online phase of fingerprinting, any other device in this frequency can change the RSS patterns of the positioning device.

Second since the WiFi fingerprinting is divided into two phases, this method suffers from the RSS variance problem caused by differences in device type, antenna orientation, and environment changes between two phases [10].

Moreover, the accuracy of this method depends on sample points' sampling space. Large amount of sample points are needed for a high positioning accuracy. Any changes in the WiFi network such as APs replacing and facilities upgrading can lead to re-sampling these points. Thus to build and maintain the fingerprinting database is a very time consuming work.

These shortages of WiFi fingerprinting method indicate that the measured locations are not always reliable. In addition, it turns out to be worse when the target is moving. Tracking errors like improper locations (e.g. at a wrong room, a wall, or a table), wrong heading and jumps between consequent locations are often seen from the results of WiFi positioning system.

To solve these tracking issues, some research has been conducted based on signal processing and combining multiple sensors (WiFi, Bluetooth and inertial measurement unit (IMU), etc.). However very little research makes use of the user's movement and the indoor environment to enhance localization accuracy [8]. Thus, the thesis also confronts challenges in building an appropriate model for tracking purposes. Though there are several standards for indoor modeling such as IFC, KML, CityGML LoD4, they do not consider the tracking as the main purpose. Another standard for indoor spatial information, IndoorGML, which takes indoor navigation and tracking into consideration, is still under development. A set of problems in indoor modeling are summarized in the survey of Zlatanova [11]. Among them, the problems for tracking application are determining the composition of space and granularity of model.

1-3 Objectives and research questions

Spatial models which can represent geometry, topology and semantics of indoor space include constraints information about a person's location and moving path in an indoor environment. Therefore spatial models can contribute to detecting and correcting poor positioning results. The goal of this thesis is to propose and implement a spatial model-aided tracking algorithm that is able to reduce tracking errors of WiFi positioning systems and provide reasonable location estimation when the positioning results is poor or missing temporally. The main research question to be addressed is:

What contributions can a spatial model make to decrease tracking errors of WiFi technology for indoor navigation?

This research question is divided into three underlying questions:

1. What features of spatial model could be employed for tracking? A spatial model may have geometric, topological and semantic features. We think all of them can contribute to tracking, but what kind of geometric, topological and semantic features are suitable needs to be explored.
2. In what way can the selected features be integrated together to decrease tracking errors of a WiFi positioning system? Inaccurate locations measured by a WiFi positioning system can be filtered out according to model's geometrical, topological and semantic features. For example, a location can be matched to the model based on geometric coordinates and its accessibility can be checked based on connectivity and semantics of the spatial model. Thus, it is significant to develop a tracking algorithm which can make use of the model's features efficiently.
3. How can the spatial model aided tracking algorithm be implemented and tested? In order to test the algorithm in a live environment, a tracking system for mobile devices needs to be developed. Thus, the methodology of how to implement the system is studied.

1-4 Research scope

The core of this research is to use spatial models to decrease the tracking errors of WiFi positioning system upon navigation purposes. The spatial model is the base of the tracking algorithm and it needs to be built by the author. Not all types of spatial models are looked at, but two most commonly used spatial models: grid-based model and network-based model. The tracking algorithm is not aimed to improve absolute positioning accuracy but decrease unreasonable locations. Finally the spatial model aided tracking method needs to implemented based a WiFi positioning system and tested in a live environment. However, There are several things not involved in this research:

- The development of localization system. This research will use a WiFi positioning system developed by other researchers and implement the spatial model aided -tracking algorithm based on the localization system.
- The construction efficiency of spatial models. The aim is to use spatial models, not to build spatial models. Thus the spatial models are created in a convenient way.
- The moving pattern of a pedestrian. This thesis will not take various moving patterns of person into consideration, only the most common case that walking in a constant speed is considered by the tracking algorithm.
- The implementation of navigation. The locations provided by the tracking system could be used for navigation purposes but the implementation of navigation service is not concerned by this research.

1-5 Contributions

The values of this thesis research can be seen from both a scientific point of view and a societal point of view:

1-5-1 Scientific contribution

The main contribution of this thesis in scientific field is that it proposes a tracking algorithm that makes use of the geometry, topology and semantics of a spatial model. From the literature review in the next chapter, it can be seen that most commonly used tracking methods are dead reckoning, Bayesian filters and map matching. These approaches employ very limited spatial information. Moreover, it is hard to find any research on how to integrate geometrical, topological and semantic features of indoor environment for tracking. Thus, this thesis first studies the suitable spatial model for tracking purposes by comparing various proposed spatial models in former research, then develops a tracking algorithm based on the selected spatial model, finally implements and tests the algorithm in a real environment. These three steps respectively answer the three research questions given. We consider this thesis can contribute to the joint research of spatial model and indoor tracking.

Moreover, we develop the tracking algorithm based on a WiFi localization system developed by the researchers of Wuhan University. In addition, our algorithm can improve the initial positioning results of the system. Thus, the project can also benefit their research.

1-5-2 Societal contribution

Tracking service is an important aspect of location-based service (e.g., tracking children, elderly, friends, customers, etc.) and is the basis of other location-based services (e.g., navigation, evaluation, mobile advertising, etc.).

The growth of indoor LBS market is very impressive these years. The report of MarketsandMarkets [12] estimates indoor location market will grow from $ 448.6 million in 2013 to $2.6 billion in 2018 and there are at least 170 companies today working on indoor location, indoor maps, in-building tracking, and wayfinding (navigation) inside buildings.

However, providing precise location inside buildings is challenging. The improvement of current tracking systems are expected by the market. With the boom in WiFi network, the WiFi based positioning technology gains great popularity now. However the shortages of WiFi positioning systems (introduced in 1-1) prevent this technology to be ubiquitous. The output of this research can improve the tracking performance of WiFi positioning system so that it could be more widely used.

This project is carried out at the Company CGI Group that is a leading IT consulting, systems integration, outsourcing, and solutions company. The company has great interest in the field of indoor localization and navigation. Some customers of the company are looking for a solution for indoor localization and tracking issues. The company also owns the research results of this project and they may use it for real applications in future.

1-6 Chapters overview

This thesis consists of 5 chapters which are organized as follows:

Chapter 1 introduces the context of this research, the challenges in indoor tracking, the objective and scope of the thesis.

Chapter 2 presents an overview of the existing indoor modeling methods and the common used tracking methods. Decisions on what types of model and what tracking methods should be used are made in this chapter based on literature study.

Chapter 3 explains what features of the spatial model are selected upon tracking purposes and how a tracking algorithm including these features is designed. The requirements of the algorithm are first analyzed and then how the algorithm is developed based these requirements is illustrated.

Chapter 4 presents the implementation of the tracking system and test results of the system in a live environment.

Chapter 5 concludes this thesis research.

Chapter 2

Background

This Chapter introduces the background of the thesis. The working principle of the WiFi positioning system is first explained. Then a literature study is carried out to find the suitable spatial model for the tracking. The grid model is selected according to the comparison of various models. At last, four existing tracking techniques are introduced and we determine to integrate three of them for this thesis: grid model, magnetometer and grid filter.

2-1 WiFi positioning system

The WiFi positioning system used for this thesis is provided by the researchers of Wuhan University, which uses WiFi fingerprinting method to localize the mobile client. The framework of the system is depicted in figure 2-1.

In the offline training phase, for each sampling point the RSS from several APs are measured. The system assumes that the RSS values from different APs are independent and have Gaussian distribution. The system estimates the distribution's parameters rss_i_para (mean and standard deviation) using a number of RSS from an access point $AP(i)$. The coordinates of each sampling point x with its RSS distribution parameters for several APs ($rss1\_para$,$rss2\_para$,$rss3\_para\ldots$) are stored in database, also known as the radio map. In order to facilitate the matching process in the online phase, the system clusters the radio map's sampling points according to the four strongest APs covering each point. The sampling points sharing common APs (four strongest APs) are grouped. In the later process, the system only searches the corresponding cluster of sampling points rather than the whole radio map.

In the online location determination phase, the mobile client measures a signal strength vector $rss = (rss1, rss2, rss3\ldots)$ from available APs at its location, then the positioning algorithm matches the vector rss to the radio map to find the sampling point x that maximizes the probability $P(x|rss)$, i.e. we want $\arg\max_x [P(x|rss)]$.

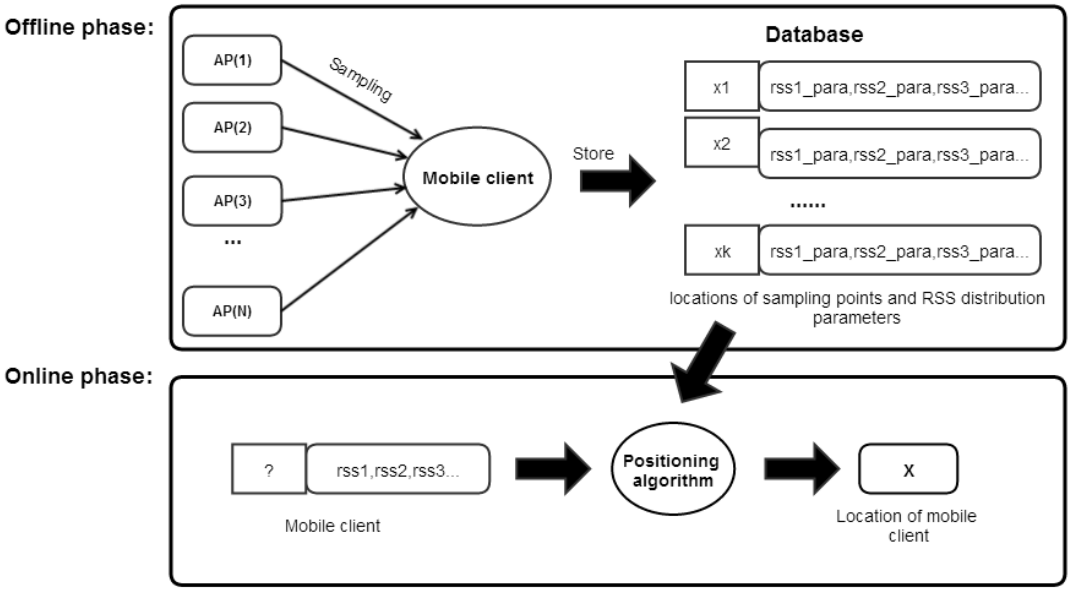


Figure 2-1: WiFi fingerprinting positioning system [13]

The positioning algorithm is based on Bayes' theorem [3]:

$$P(x|rss) = \frac{P(rss|x)P(x)}{P(rss)} \tag{2-1}$$

since $P(rss)$ is a constant value for all x and the algorithm assumes that all sampling points are equally likely, i.e. $P(x) = 1/C$ where C is the number of sampling points. We can get the equation 2-2 [13]:

$$\arg\max_x [P(x|rss)] = \arg\max_x [P(rss|x)] \tag{2-2}$$

Therefore the problem now is to find the $\arg\max_x [P(rss|x)]$. The $P(rss|x)$ is calculated using the radio map [13]:

$$P(rss|x) = \prod_{i=1}^{k} P(rss_i|x), i = 1, 2, \ldots, k \tag{2-3}$$

Where $P(rss_i|x)$ represents the probability of receiving signal strength rss_i from an access point $AP(i)$ at location x, which can be obtained from the corresponding Gaussian distribution in the radio map. k is the number of available APs at location x.

Due to the fact that the mobile client is continuously moving rather than static, the developers use time-average window (see formula 2-4 [13]) to smooth the resulting location estimate. It obtains the location estimate by averaging the last W locations estimates. Their tests show the optimal value for W is around 6.

$$\bar{x}_t = \frac{1}{min(W,t)} \sum_{t-min(W,t)+1}^{t} x_i \tag{2-4}$$

In addition, the system implements device self-calibration based on the RSS histogram to reduce the influence of device variance. There is a linear relation between the RSS histogram of the user's device which is created and updated in real time and the RSS histogram of the reference device which is obtained from the existing radio map [13]. According to the linear relation, the measured RSS can be adjusted and matched to the radio map.

The system has an accuracy of 3.4m for 90% locations when the mobile client is static. However, we find that the accuracy of the system for a moving mobile client is lower according to our tests. In order to track a pedestrian in a complicated indoor environment, other technologies are needed to improve the performance of the positioning system.

2-2 Modeling indoor space

2-2-1 Representations of spatial model

Various indoor spatial models have been proposed by former researchers [14, 15, 16, 6, 9]. By reviewing these models, we find that they can be distinguished by their representation, i.e. geometry, topology and semantics. There are few pure geometric, topological or semantic model, but hybrid models using various levels-of-abstraction [9]. In this thesis, we want to construct a spatial model integrating geometric, topological and semantic features, thus in below the approaches to represent each type of feature are studied.

The geometry of a spatial model depends on the subdivision of indoor space. The indoor space can be partitioned by either real boundaries of the subspace (walls and doors) or arbitrary boundaries defined by developers. The former method is not considered for this thesis because it lacks capability to represent the spatial entities at low level (e.g. sub room and objects inside room) and it is less suitable for navigation services [17]. The later method, also known as cell-based partition, tessellates the indoor space into a finite number of areas. These areas could be regular shapes like square grids, or irregular shapes such as triangles, trapezoids and Voronoi diagrams [17]. Several commonly used cell-based models are compared in the next section to determine which one is more suitable for tracking. The granularity of subdivision (the size of cell) determines level of detail of the spatial model (e.g. room-level, sub room-level, object-level). According to the survey of Domínguez, García and Feito [9], most presented spatial models use point, line and polygon to represent the indoor space and have detail in room-level.

The topology of an indoor space may be modeled either as a fully 3D space, using for example cell complexes or as a linked collection of 2D layers [6]. Only the later approach is considered in this thesis because it is easier to implement and sufficient for tracking purposes. Two types of topological relations should be distinguished: connectivity and adjacency. Two spaces are connected only if there are doors or windows between them but they are adjacent as long as they share one item (like a wall). In the review of Domínguez, García and Feito [9], most presented spatial models utilize connectivity graph to represent topology of the indoor space. The connectivity graph can provide information about the accessibility between two locations that is useful for location determination. Thus our spatial model should have connectivity representation.

The semantics of a spatial model describes the basic spatial and structural concepts of indoor environments [16]. The semantics is often referred to as ontology when it is not only used

for representational purpose but also for reasoning [6]. Worboys [6] thinks it is significant to make distinction between the fixed structures which support indoor space (e.g. walls, doors, windows and floors) and the entities which are usually moveable (e.g. furniture and equipment). More detailed definitions of indoor ontology are given by Goetz and Zipf [18] as well as Tsetsos and Anagnostopoulos [16]. Goetz and Zipf classify the indoor space into room, corridor, hall and vertical passage, and defines obstacle and point of interest to represent indoor entities. Tsetsos and Anagnostopoulos define three subclasses for indoor space: corridor, room and floor, while the vertical passage are categorized to the class of path_element. Concepts of obstacle and point of interest are also used by them. For this thesis, we think it should also distinguish two semantic types: space (e.g., room, corridor, vertical passage) and object (e.g., obstacle and door).

2-2-2 Cell-based models

This section introduces and compares several commonly used cell-based models. We classify these models into two categories according to their decompositions of the space. The first class is network model, which is generated by irregular subdivision, and the second class is grid model, which is obtained by regular subdivision.

There are several approaches to build network models, such as Constrained Delaunay triangulation, Voronoi diagram, convex polygon-based and trapezoidal-based subdivision. This model has vertex-edge representation where vertices represent irregular cells, and edges represent connections between them. Moving from one vertex to the other one is allowed only when there is an edge between them. The edge indicates the distance or travel time between vertices. Moreover, the vertex can include semantic information about the location, e.g., room vertex, door vertex, obstacle vertex, etc. Four examples of the network model are given below.

Network model based on functional cells (given in figure 2-2): Lorenz, et al. [19] divide the free space of the building into non-overlapping cells. The large room and long corridor are decomposed into several cells due to their size. The room with multiple functional areas is also subdivided according to their functionality, for example, an airport lounge may feature waiting areas, meeting points, areas in front of the different counters and security checks, passport control, etc. All of them serve a different purpose and they need to be represented by different cells. The cell centers are extracted as the vertices of network model, and two cells are connected by an edge. The model also contains semantic information. The vertices extracted from the room's cells and the corridor's cells are distinguished, the door that connects two rooms or a room and a corridor is represented by an edge. Their model is created mainly for way finding purposes.

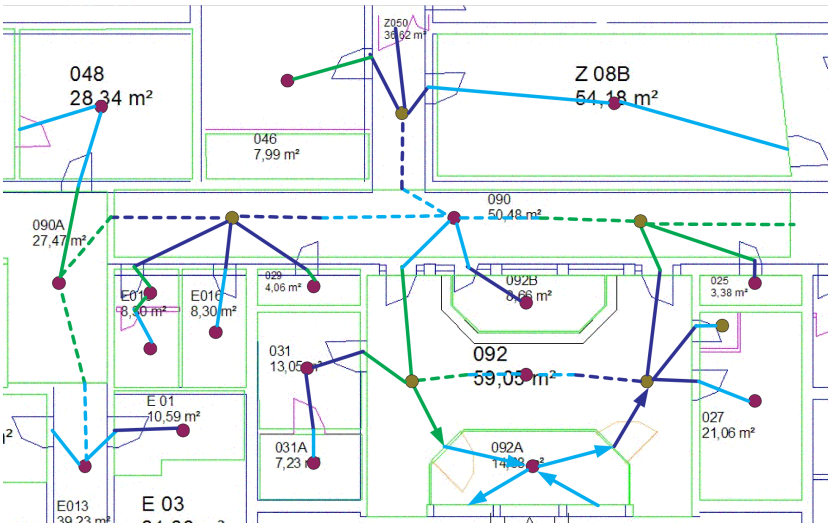
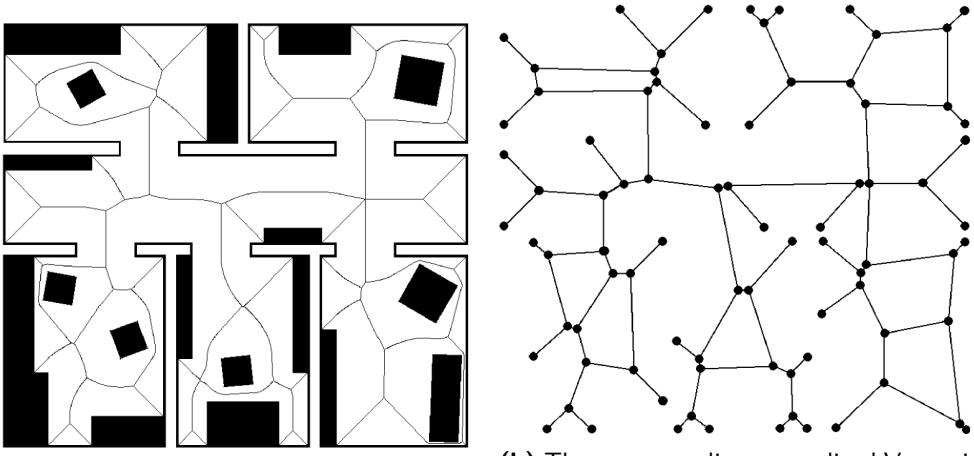


Figure 2-2: Network based on functional cells [19]

Network model based on generalized Voronoi graph (GVG) (given in figure 2-3): Wallgrün [20] derive the GVG from its corresponding generalized Vorinoi Diagram (GVD). The GVD is a retraction of the free space onto a network of curves reflecting the connectivity of free space. Figure 2-3a shows a simple two-dimensional environment and the corresponding GVD (fine lines) consisting of curves that intersect at meet points and end up in corners of the environment. Figure 2-3b shows the GVG, which is the graph corresponding to the GVD in figure 2-3a with vertices corresponding to meet or corner points and edges connecting vertices joined by Voronoi curves. This model well represents the topology of the environment and is suitable for path planning and spatial reasoning.



(a) The generalized Voronoi diagram

(b) The corresponding generalized Voronoi graph

Figure 2-3: Network model based on Vorinoi diagram [20]

Network model based on Constrained Delaunay triangulation: we construct the network model using constrained Delaunay triangulation (see figure 2-4). Because the initial result of triangulation of a complicate space has many skinny tangles, we combine these small triangles

with the neighbor that has the smallest area. The centroids of triangles are extracted as vertices of space, a door or obstacle inside the space is represented by a vertex as well. To build the connectivity, the vertices inside a space are connected with their neighbors and the vertices in different spaces are connected through the door vertices.

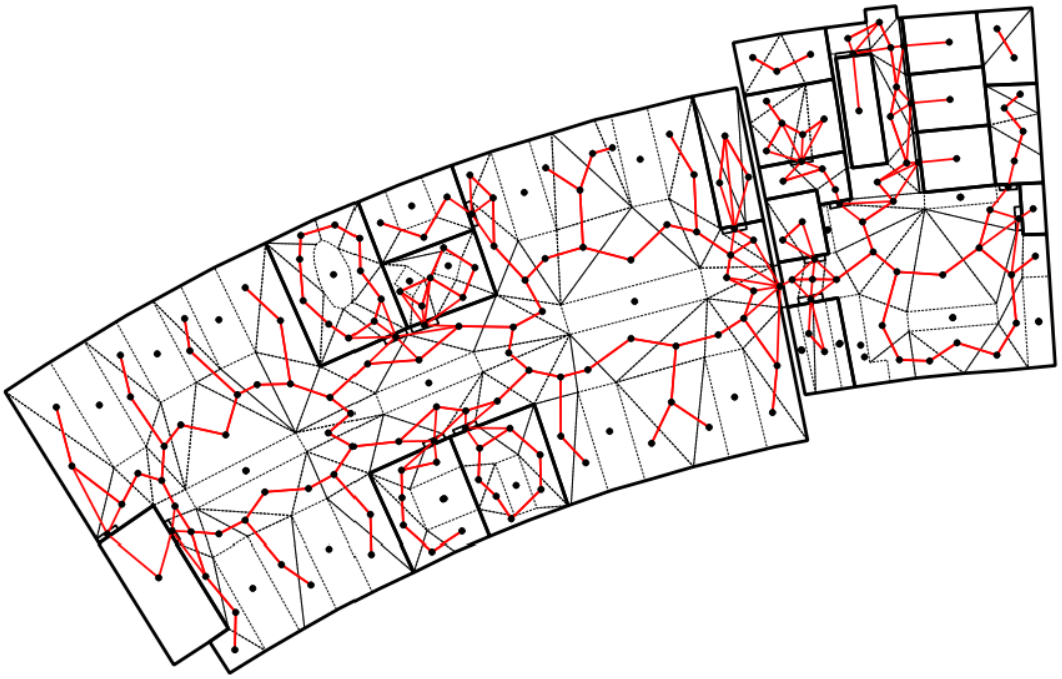
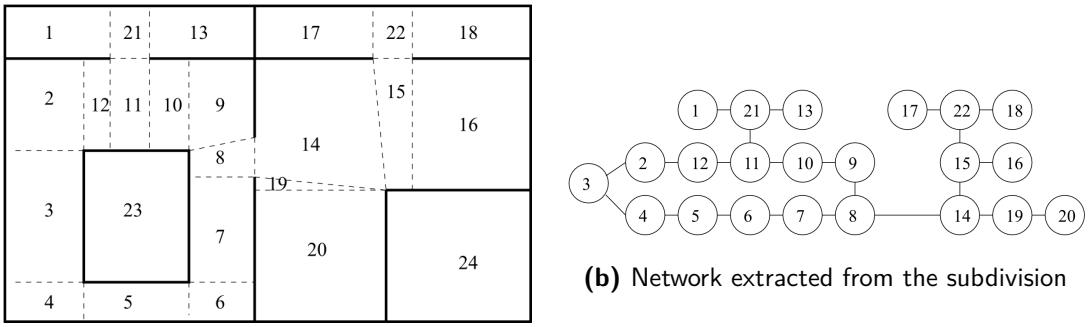


Figure 2-4: Network based on Constrained Delaunay triangulation

Network model based on convex polygon: Lamarche and Donikian [21] derive the network model from convex polygons of the environment. Extra constraint segments are added to the parallel walls and corners of structures firstly, then the Delaunay triangulation is applied under the constraints of the added segments and walls, finally the triangles are integrated to generate convex cell (see figure 2-5a). Once the convex cell subdivision is computed, a graph containing topological relations is extracted. A vertex of this graph is a convex cell and an edge represents a free segment shared by two adjacent cells with a length greater than the width of the humanoid (see figure 2-5b). This spatial subdivision accurately maps the environment geometry and the extraction of narrow areas automatically identifies the most constrained parts of this environment. Thanks to this information, accessibility between adjacent cells can be filtered before any path planning computation by using the humanoid width. This network model is also designed for path finding.

In this thesis, we only consider the grid model with square cells because it is the simplest and most often used shape. The grid model decomposes the space into regular units with meaning (semantic elements), for example a piece of room is considered a grid unit, and each grid unit is linked to its neighbors [14]. Since the grid model does not abstract the space, it is able to describe the location accurately and continuously. By reviewing literature, we find that less grid models are employed for indoor pedestrian navigation, however a similar model, called occupancy grid, are widely used for mobile robot navigation and tracking. In below, a grid

model of indoor spaces is first introduced, and then the occupancy grid is explained.

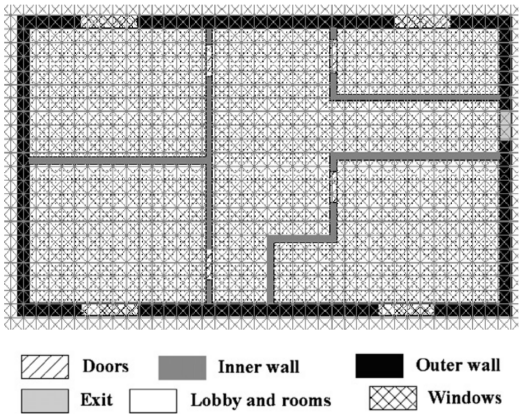


(a) The subdivision of the environment

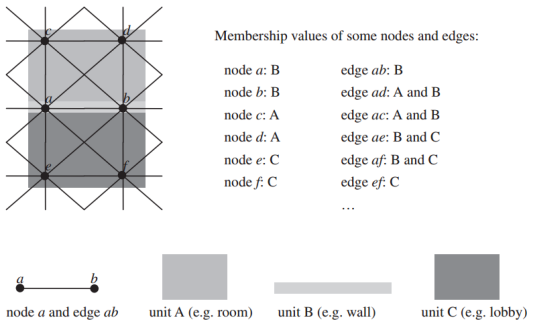
(b) Network extracted from the subdivision

Figure 2-5: Network model based on convex polygon [21]

Li, Claramunt, and Ray [14] propose a grid graph model (see figure 2-6). They first represent the indoor space using cellular units (different elements of the indoor space like a room, a wall, etc.) and then overlap a grid graph on the cellular units, which is illustrated in figure 2-6a. Each vertex of the grid graph is connected to its eight neighbors (apart from the ones located on the boundary of the grid model) with horizontal, vertical, and diagonal edges. Finally, in order to reflect the geometrical information of cellular units in an indoor space, vertices and edges of the grid graph are labeled according to their memberships to the underlying cellular units. Each vertex that is contained by one and only one cellular unit has one and only one membership value, while the membership value of an edge is multivalued when this edge intersects several cellular units (see the figure 2-6b). The topological relationships among cellular units can be extracted from edges with multiple membership values. Their research also indicates that the extent and granularity are most important parameters for a grid model. The extent of the grid model is usually set as the extent of the indoor environment and the granularity of the grid model needs to be determined carefully according to the purpose of the application.



(a) Overlap of the grid graph and cellular units.



(b) Intersections between cellular units and graph components (vertices and edges).

Figure 2-6: Grid graph model [14]

An occupancy grid is a regular matrix consisting of equally-sized cells, and each cell can

be connected to its eight neighboring cells. A high probability is assigned to cells within accessible space, while a low probability to cells occupied by obstacles. Simplicity and metric embeddedness are two advantages of the occupancy grid approach [22]. An example of occupancy grid is depicted in the figure 2-7.

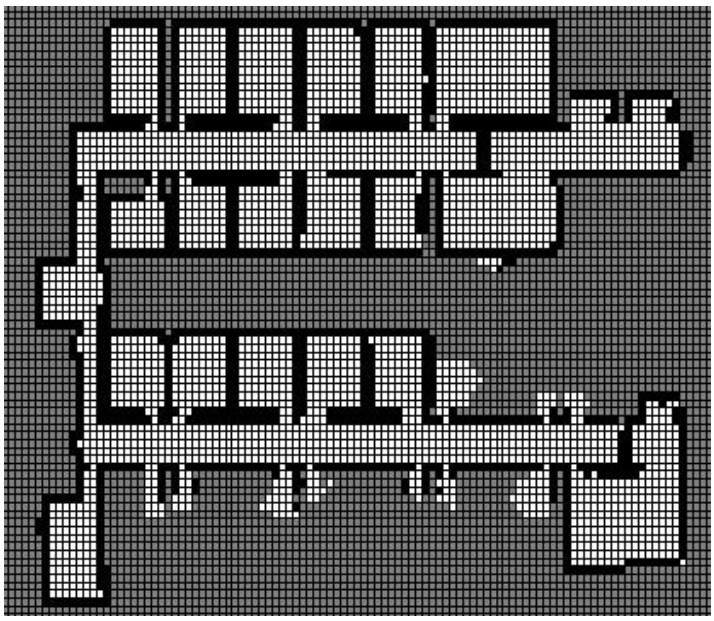


Figure 2-7: Occupancy grid [23]

The pros and cons of both cell-based models are summarized in the table 2-1.

Model	Advantages	Disadvantages
network	location based on an irregular tessellation object-based or empty space related analysis easy to tessellate the space efficient because more compact	inaccurate location data
grid	accurate location data continuous analysis easy to design and maintain high flexibility	Consumes excessive amounts of memory and processor time in large spaces

Table 2-1: Comparison of grid model and network model [17]

By reviewing all these models, we find that the network model has good representation of the topology of the environment and is often used for path planning. However, this model is considered less accurate since they lack geometric details on entities and places indoor. The grid model is selected for this thesis due to the following reasons:

- The grid model is able to describe accurate location. The grid model usually has small cell size (less than 1 meter), thus almost any locations of the indoor environment can be reached through this model.

- The grid model has high flexibility on the granularity. According to the complexity of indoor environment, different granularity of decomposition may be desired. For a simple environment with very little obstacles, a coarse decomposition is feasible, however for a complex environment with many sub-spaces and obstacles, a fine decomposition is needed to represent all details of the environment. It is very easy to adjust the granularity using grid model because its cells have same size and can be scaled together. But the granularity of cells of the network model is not uniform and it is more difficult to obtain a fine decomposition through this method.
- The grid model is suitable for computation. The grid model can be also regarded as a matrix which enable many matrix-based computation.
- The grid model is easy to implement and maintain.

2-3 Indoor tracking methods

Various technologies have been proposed in the literature to solve tracking problems. They are briefly introduced below.

2-3-1 Dead reckoning

Dead reckoning calculates a person's current location by advancing a known position using course, speed, time and distance to be traveled. This data is measured by inertial measurement unit (IMU) equipped on the tracked device, for example, accelerometer for steps, gyroscope and magnetometer for heading. The uncertainty of dead reckoning positions grows with time thus it is necessary to check the position regularly [24]. As a result, this method is usually combined with other positioning technology (e.g. WiFi, RFID) to track the moving object.

2-3-2 Grid filter

Grid filter is a discrete Bayesian filter, which probabilistically estimates a target's location based on observations from sensors. Grid filter tessellates the environment into small patches, typically between 10 cm and 1 m in size. Each grid cell contains the probability that the person or object is in the cell [25].

To make the computation tractable, grid filter is based on the Markov chain model. The Markov Markov chain model is a system that its future state only depends on its current state and not the previous states. An illustration of the Markov chain model is given in figure 2-8. For location estimation, it assumes that sensor measurements depend only on an object's current location and that an object's location at time t depends only on the previous location at time $t-1$. Locations before $t-1$ provide no additional information.

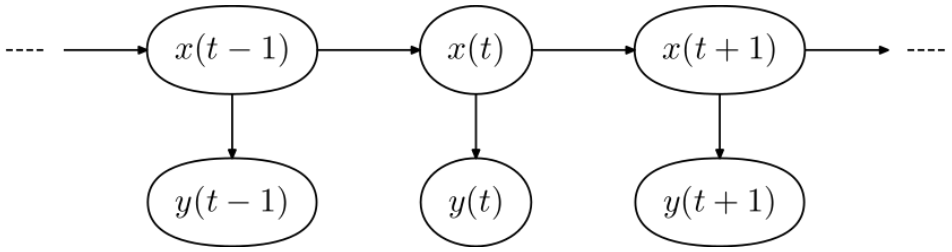


Figure 2-8: Markov chain model [26]: $x(t)$ is the state at t, $y(t)$ is the observation at t.

The grid filter methods are widely used in the field of robotics [4, 5]. It computes the location in two phases: the prediction phase where the prior probability p_t^- of location x_t is estimated based on the previous location x_{t-1} , the motion model u_t (the control of the robot's motion, mostly the velocity) and the map of tracking environment m, and the update phase where the posterior probability p_t is computed by multiplying the prior probability with the conditional probability $p(z_t|x_t)$ that the measurement z_t may be measured at x_t. Thrun and Burgard and Fox [4] give the prediction formula 2-5 and update formula 2-6 of mobile robot localization and tracking.

$$p_{k,t}^- = \sum_i p(x_{k,t} \,|\, u_t, x_{i,t-1}, m)\, p_{i,t-1} \tag{2-5}$$

$$p_{k,t} = \alpha\, p(z_t \,|\, x_{k,t}, m)\, p_{k,t}^- \tag{2-6}$$

where $x_{k,t}$ and $x_{i,t-1}$ denote individual locations (grid cells) at t and $t-1$, $p_{k,t}^-$ and $p_{k,t}$ are the prior and posterior probability over a grid cell $x_{k,t}$, $p_{i,t-1}$ is the posterior probability over a grid cell $x_{i,t-1}$, z_t refers to measurements obtained from $t-1$ to t, u_t denotes the motion control of the robot, m refers to the map of the environment, α is a normalizing constant which lets $p_{k,t}$ between $[0, 1]$.

An example of one dimensional grid filter is given in figure 2-9. In this example, the robot carries a door-sensing camera that cannot distinguish different doors. Each frame depicts the robot's position in the hallway and the current belief $b(x)$: The step (a) is the initial state where all grid cells are assigned the same probability since the actual location is unknown, in step (b) when the measurement is input, the probability of the robot's location is updated. In step (c), the probability of location is predicted based on the previous measurement and the robot's motion. When the new measurement is obtained in step (d), the probability is updated again. The two phases are repeated to obtain new locations.

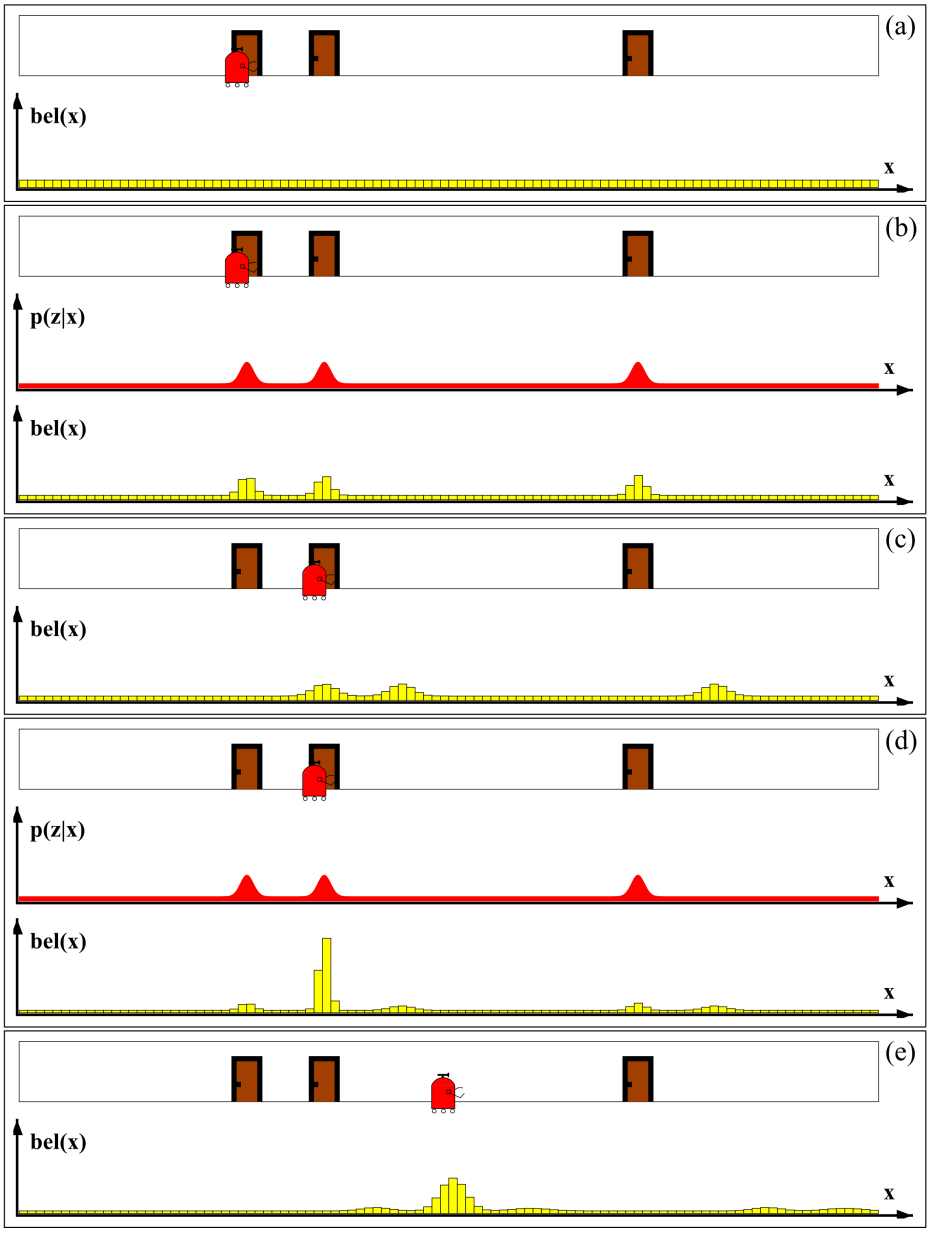
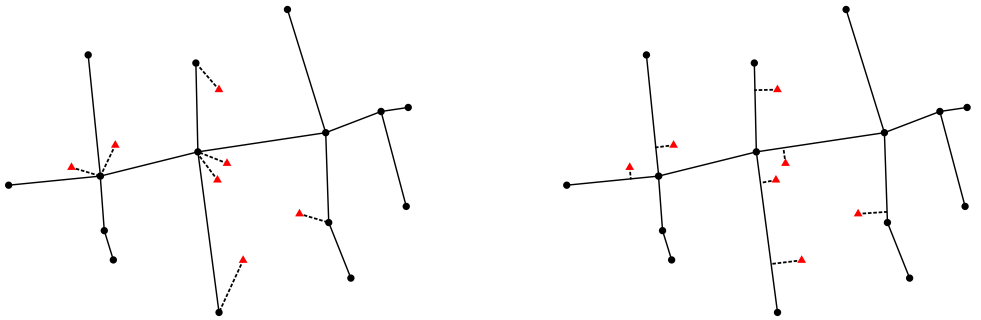


Figure 2-9: An illustration of grid filter [4]:(a) the robot's location is unknown, $bel(x)$ refers to initial probability $\{p_{k,0}\}$. (b) the sensor detects a door, $bel(x)$ refers to the posterior probability $\{p_{k,1}\}$ at $t = 1$. (c) the robot moves, $bel(x)$ refers to the prior probability $\{p^{-}_{k,2}\}$ at $t = 2$. (d) the sensor detects another door, $bel(x)$ refers to the posterior probability $\{p_{k,2}\}$ at $t = 2$. (e) the robot moves again, $bel(x)$ refers to the prior probability $\{p^{-}_{k,3}\}$ at $t = 3$. Additionally, (b) and (d) depict the p(z|x), the conditional probability of observing a door at different locations in the hallway.

2-3-3 Map matching

Map matching assumes the person can be only located along certain routes. The constraints of indoor construction are used to correct estimates of the position of a person moving within a building, e.g. people and objects do not pass through walls, they pass along corridors and through doorways [24]. Two commonly used map matching techniques are point-to-vertex matching (match the measured location to a vertex of a route) in figure 2-10a and point-to-edge matching (match the measured location to an edge of a route) in figure 2-10b. Spassov [27] implements the later technique for continuous localization. Their results show there is 85% edges are identified correctly. This method performs quite well in corridor environment. There are no jumps of positions and the average distance to the middle of the corridor is acceptable.



(a) Point-to-vertex matching **(b)** Point-to-edge matching

Figure 2-10: Map matching

2-3-4 Model based

Model based methods uses a vector model of the indoor environment to improve the estimation of the user's location. This method considers more factors than the constraints of indoor space, which can be taken as an expansion of the mapping matching method. Girard et al. [28] define model based tracking is a method that combines the use of model features (such as walls, open areas or obstacles), information from the sensors (such as speed and direction) and information from the user (such as mean velocity and velocity variance [29]) and they use a grid model together with particle filter, ultrasound range sensors, and foot-mounted IMU for indoor pedestrian tracking. Jensen, Lu and Yang [7] propose a base graph model which represents the connectivity and accessibility of indoor space and use the model for RFID-based tracking. More specifically, an deployment graph model of RFID reader is constructed from the base graph model and this model is then used in their algorithms for constructing and refining trajectories from raw RFID readings. Their experimental results show that the method can considerably improve the indoor tracking accuracy at low computational cost.

2-3-5 Integration of the selected tracking technologies

The four techniques above are commonly used tracking methods. However, there are some disadvantages of these methods: the dead reckoning is very dependent on the hardware (e.g. accelerometer, gyroscope, magnetometer, etc.), the grid filter needs continuous inputs of measurements from sensors, the map matching method is very simple but may have large matching errors, the model-based method also needs information from sensors or users. In order to overcome the disadvantage of an individual technique, we will use a combination of grid model, magnetometer and grid filter to improve the locations measured by WiFi positioning system. By integrating the grid model and grid filter, the geometric, topological and semantic features of the grid model can be used to predict the prior probability of the person's location. In the later phase, the probability is updated using the measurements from WiFi positioning system. However, the WiFi measurements could have very low quality for the reasons given in section 1-2. Thus, it is better to also make use of measurements from other sensors rather than only the WiFi positioning system. Due to the restriction of hardware, this thesis only employs the magnetometer to measure the orientation of the walking person. When the WiFi measurement is not applicable, the orientation data can be used instead to update the location's probability.

Chapter 3

Spatial model-aided tracking algorithm

As the search question 1 and 2 are strongly related, they are studied together in this chapter.

Question 1: *What features of a spatial model could be employed for tracking?*

Question 2: *In what way can the selected features be integrated together to decrease tracking error of WiFi positioning system?*

The section 3-1 analyzes various geometric, topological and semantic features of a spatial model and explains what features are considered to be useful for tracking. The section 3-2 describes the design processes of the tracking algorithm. In this section the requirements of the algorithm is looked at first. Then the flowchart of the algorithm and specific explanations of important processes in the flowchart are given in sub sections.

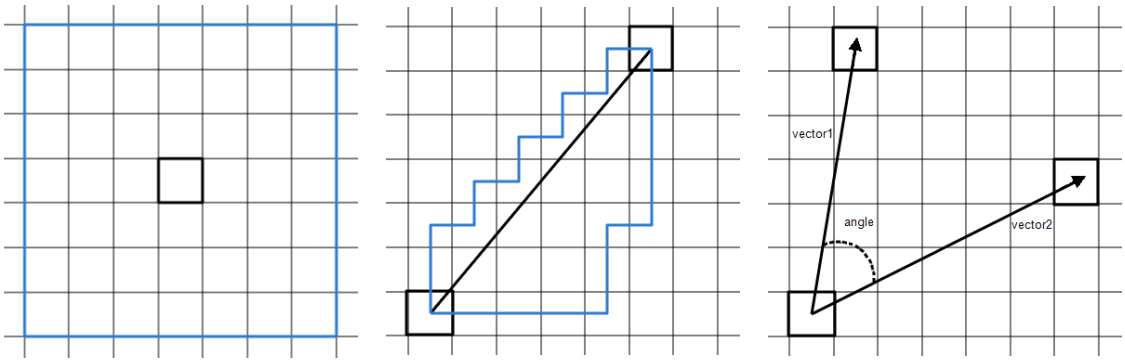
3-1 Spatial model's feature

As introduced in the previous chapter, the grid model is selected for this thesis. Thus, we will analyze all features of the model and figure out what features can contribute to the improvement of tracking performance. The features are classified into three categories: geometry, topology and semantics. They are looked at respectively in the following.

3-1-1 Geometric features

The geometry is the base of a grid model. To make the model accurate, the grid cell size should be small (usually between 10 cm and 1m). For this thesis, 0.7m is used. More explanations about how the cell size is determined are given in section 4-1-1. Four geometric features of the model are introduced below.

Geometric coordinates: We represent grid cells using center points of cells. With the coordinates of center points, the measured locations of WiFi positioning system can be mapped to the model using the point-to-vertex approach (introduced in section 2-3-3).



(a) Buffer: the blue rectangle is the buffer of the black grid cell with a radius of 3 times the cell size.

(b) Distance: the black line is Euclidean distance and the blue lines are two paths that have the same Manhattan distance.

(c) Orientation vector: two orientation vectors and the angle between them.

Figure 3-1: Geometric features:(a) buffer, (b) distance, (c) orientation.

Buffer: It costs a lot of computational processors to compute a location using the whole model because it commonly covers a large area and has fine-resolution grids. Thus, it is necessary to extract the relevant part of the model for the current location and only use this part for location determination. We use a buffer of the known previous location to represent the search region of the current location. The range of the buffer relays on the walking speed and the time interval of location determination. An example of buffer feature is depicted in figure 3-1a.

Distance: For a grid model, there are two common approaches to compute the distance between two grid cells: Euclidean distance and Manhattan distance. The former one is the ordinary distance between two points and the latter one is the sum of the absolute differences of their Cartesian coordinates [30]. For Manhanttan distance, multiple paths could have the same distance value. An example of two distances are given in figure 3-1b.

Orientation vector: We use an orientation vector to represent the direction from one grid cell to another. The angle between two vectors (e.g., $v1(x_1, y_1)$ and $v2(x_2, y_2)$) can be obtained through computing the dot product of two vectors [31] (see formula 3-1). An example of orientation vector is given in figure 3-1c.

$$\theta = \arccos\left(\frac{v1 \cdot v2}{\|v1\| \, \|v2\|}\right) \tag{3-1}$$

3-1-2 Semantic features

According to the literature (see section 2-2-1), semantics such as floor, space (e.g., room, corridor and vertical passage), door, wall, obstacle and point of interest are commonly used. We think following semantic features are useful upon tracking purpose:

Floor: the floor feature can help determine the change of floors.

Space: we define a space is a region that is separated from other regions by a real boundary (e.g., room and hall) or a region that has a specific function (e.g., vertical passage). The wall

can be represented by the boundary of the space, so it is not necessary to have semantics of wall. The space concept is important for tracking because users expect to be localized at the correct space even if the accurate location inside the space is hardly estimated. Moreover, the space of vertical passage can be used for tracking between floors.

Door: we define a door as a connector between two separated spaces. The person has to pass the door when walking from one space to the other one.

Obstacle: we define an obstacle as an object or a region that blocks the way that is not reachable. The obstacle is needed for building the connectivity graph of the grid model. And the user should not be localized inside any obstacle.

3-1-3 Topological features

As introduced in the previous chapter, two types of topological relations are distinguished: adjacency and connectivity. We consider that the connectivity is more useful for tracking because it indicates if a grid cell can be reached from a predefined location. The adjacency relationship is less important but it is needed to build the connectivity graph of the grid model. We define the rule of connectivity as below: A grid cell is connected to its neighboring cells (at most 8) which do not belong to any obstacle and are in the same space of the grid cell. For grid cells in different spaces, they can be connected through the grid cell of a door.

In addition to the connectivity graph, two other features based on this topology could also contribute to tracking:

Shortest path: The shortest path here refers to the path with shortest distance. Based on the connectivity graph, there are several methods to compute the shortest path, e.g., Dijkstra's algorithm, A^* search algorithm, breadth-first-search, best-first-search, etc. A^* search algorithm is used for this thesis because it has high computational efficiency in this case. The shortest path can not only represent the path between two grid cells but also gives an estimation on if one grid cell can be reached from the other one within a certain time period as it returns the path length.

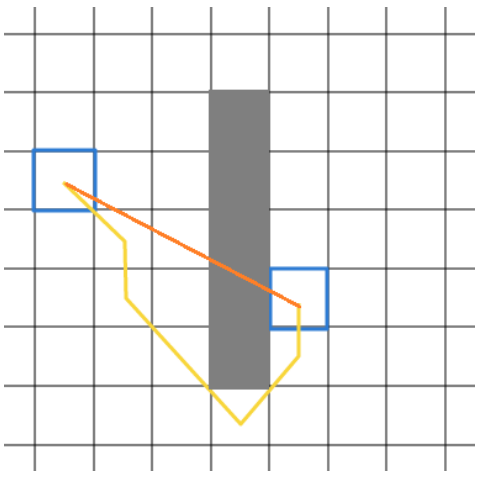


Figure 3-2: An example of distance difference: the orange line denotes the straight-line distance and the yellow line denotes the shortest path length returned by A^* algorithm.

Distance difference: we define the distance difference as the difference between the straight-line distance (Euclidean distance) and the shortest path distance from one grid cell to another. An example of the distance difference is depicted in figure 3-2. It is notable that the distance difference is small if there is no obstacle between two grid cells and it increases with the growth of obstruction between two grid cells.

3-2 Design of tracking algorithm

3-2-1 Requirements analysis

Before starting the design of tracking algorithm, several important questions need to be answered:

Question 1: What kind of data does the algorithm deal with? The data used for this algorithm is listed in table 3-1, which can be categorized into two classes. The first type of data is real-time data obtained from mobile device. The two-dimensional location data is obtained per half second from WiFi positioning system but their quality is not always high. The time of when the location is acquired and the mobile device's orientation at this time are measured using the clock and magnetometer of the device. The measured orientation has relative high accuracy when the device is kept forward the moving direction. The second type of data is model data. The grid model is used, which contains geometrical, topological and semantic information of the indoor environment.

Question 2: What assumptions are made for this algorithm? Since the motion of human has great diversity and uncertainty, it is difficult to consider all cases in this thesis. Some assumptions are made to let the tracking situation be less complicated.

- Assumption1: The start location of the person is known.
- Assumption2: The pedestrian is walking in a constant speed.
- Assumption3: The pedestrian is likely to walk in a constant direction and inside one space within a short time.
- Assumption4: The mobile device's forward orientation is the moving direction of the person.

Question 3: What requirements must the algorithm meet? There are three requirements for this algorithm:

1. Spatial model-aided. The algorithm need to make full use of the geometric, topological and semantic features of the grid model and these features should be integrated in an efficient way.
2. Adaptability. The adaptability refers to two aspects. Firstly, the algorithm is able to deal with different walking cases, e.g. walking with turning, walking between different space and floor. Secondly, the algorithm can handle the exceptions like the WiFi positioning results are extremely poor or missing temporally.

3. Effectiveness. The effectiveness is that the algorithm can improve the tracking accuracy of the WiFi positioning system.

Data	Value	Quality
location	x,y	medium (mostly around 3-5 meters)
time	nanoseconds	high
orientation	angle between the magnetic north direction and the y-axis of mobile device	high
grid model	geometry, connectivity graph, semantics	high

Table 3-1: Data quality analysis

3-2-2 Flowchart of algorithm

We design the tracking algorithm for the WiFi positioning system by integrating three other tracking techniques: magnetometer, grid model and grid filter. The grid filter (see section 2-3-2) with recursive predict and update steps is the base of the tracking algorithm in which the grid model is used instead of the map and the WiFi positioning system as well as the magnetometer are inputs of measurements. The measurements are acquired with a high frequency (every half second) but do not always have high accuracy. Thus, we use a set of measurements during a short time period for each location determination to avoid the effect of an individual inaccurate measurement. In this thesis, we use measurements obtained every 3 seconds to compute a new location because the moving distance of a pedestrian within 3 seconds is short and 6 measurements can be acquired on average.

We use the formulas introduced in section 2-3-2 to compute the prior and posterior probability grids. Unlike the mobile robot, the motion of a pedestrian cannot be controlled, and to make the situation to be less complicated we assume the pedestrian walks in a constant speed. Thus, u_t is excluded in our computation. The formulas 2-5 and 2-6 can be rewritten as below:

$$p^{-}_{k,t} = \sum_i p(x_{k,t} \,|\, x_{i,t-1}, m)\, p_{i,t-1} \tag{3-2}$$

$$p_{k,t} = \alpha\, p(z_t \,|\, x_{k,t}, m)\, p^{-}_{k,t} \tag{3-3}$$

where $x_{k,t}$ and $x_{i,t-1}$ denote individual locations (grid cells) at t and $t-1$, $p^{-}_{k,t}$ and $p_{k,t}$ are prior probability and posterior probability over a grid cell $x_{k,t}$, $p_{i,t-1}$ is the posterior probability over a grid cell $x_{i,t-1}$, m represents the grid model, z_t is the measurements obtained from $t-1$ to t and α is a normalizing constant.

To reduce computational load, we convert the $\{p_{i,t-1}\}$ to a degenerate distribution [32] for the prediction of x_t, suppose x_{t-1} denotes the grid cell with maximum posterior probability at $t-1$, then

$$p_{i,t-1} = \begin{cases} 1 & \text{if} \quad x_i = x_{t-1} \\ 0 & \text{else} \end{cases} \tag{3-4}$$

So the formula 3-2 can be simplified further as below:

$$p_{k,t}^{-} = p(x_{k,t} \mid x_{t-1}, m) \tag{3-5}$$

The usage of grid filter is to combine the different features of the grid model, also the measurements from the mobile device. By using them to compute the prior probability and posterior probability of the locations, their influence is integrated and accumulated. The flowchart of the algorithm in figure 3-3 gives an overview of the algorithm, where we can see there are three steps to track a pedestrian: initialization, prediction and update.

The initialization step initializes the probability grids $\{p_{k,0}\}$, which is used as the input to predict the first location ($x_{t=1}$).

In prediction step, we employ the previous location, the grid model as well as the previous moving direction to calculate the prior probability of the current location. When the location is at $t = 1$ or after a turning, the previous moving direction is not known or not feasible for prediction.

In the update step, we use the measurements obtained from mobile devices (WiFi positioning system and magnetometer) to update the prior probability grids and find the grid cell with maximum posterior probability. This grid cell is taken as the current location and used as the input for the prediction of the next location.

The latter two steps, prediction and update are repeated over time until the tracking is over. The pseudocode of the tracking algorithm is given in Algorithm 1, the functions of 'Initialization', 'Prediction' and 'Update' are respectively explained in the following sections. One thing should be noticed that the flowchart does not include the case of tracking between floors. Since the WiFi positioning system can only provide locations in two dimension, the change of floors is determined in a different way which is explained briefly in section 3-2-6.

Data:
x_0 :the start location
M :the grid model
z :measurements including measured locations and orientations
Result:
a trajectory

Initialization(x_0, M);
while $t < T$ **do** `// t starts from 1, T is the end time of the tracking`
  Prediction(x_{t-1}, M);
  Update(z_t, M);
  t = t+1;
end

Algorithm 1: Spatial model-aided tracking algorithm

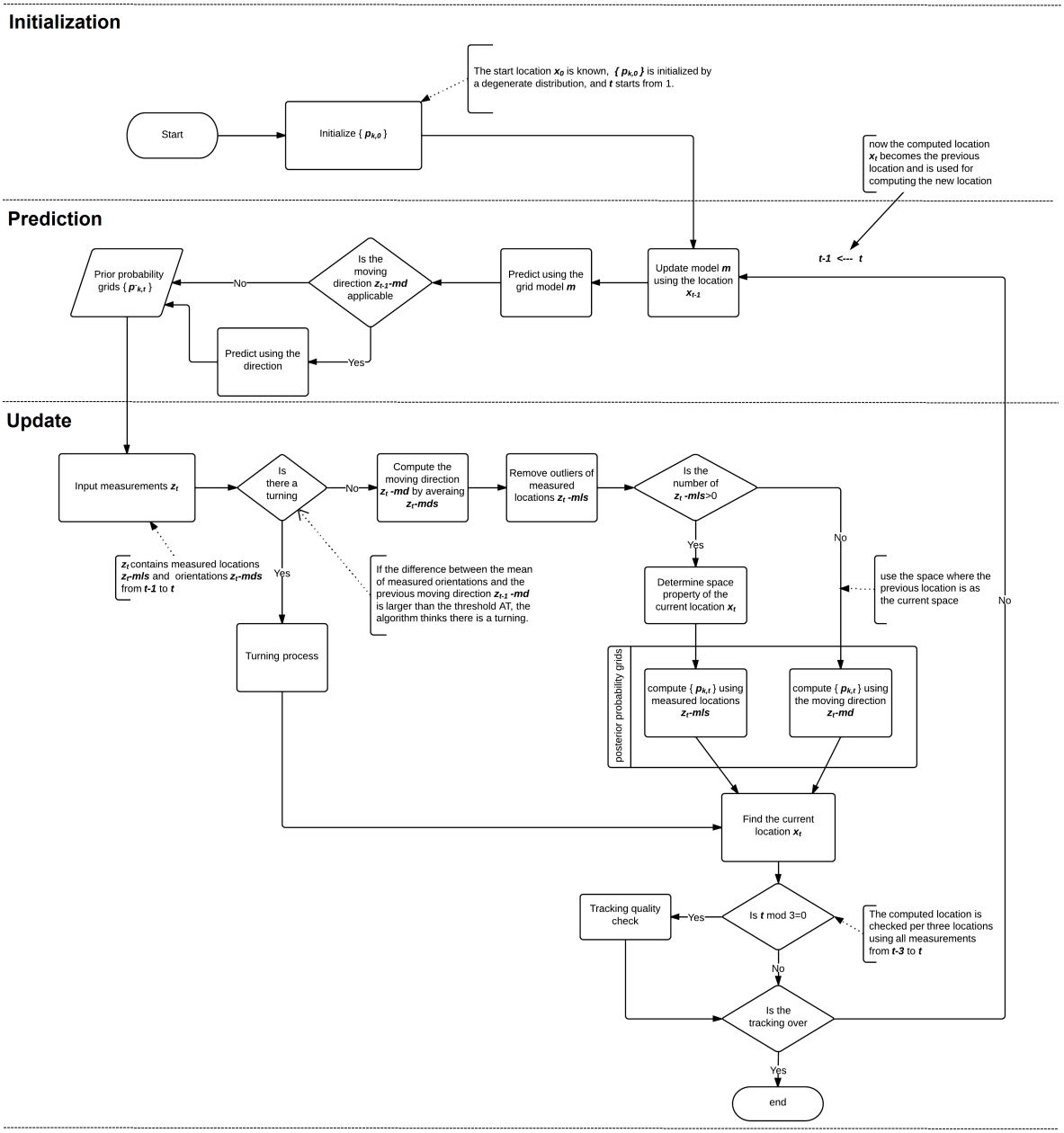


Figure 3-3: The flowchart of the tracking algorithm

The important terms and concepts used to describe the algorithm are clarified here to make the description in later sections understandable:

- Computed location: the location computed by this algorithm, which is a grid cell.
- WiFi measured location: the location measured by the WiFi positioning system.
- Measured orientation: the orientation of the mobile device when WiFi location is acquired, which is measured by the compass of the mobile device.

- Orientation vector: the vector from the center of one grid cell to another.
- Tracking interval: the time interval between two consecutive computed locations.
- Measurements: all measurements acquired by the mobile device during tracking interval, each measurement contains three values: WiFi measured location, measured orientation, and time.
- In moving direction: when we say a cell is in the moving direction, we mean the angle between the orientation vector of this cell and the moving direction is less than an threshold.
- Clustering Distance threshold (CDT): the clustering threshold for WiFi measured locations.
- Probability grids: the grid cells with initial, prior or posterior probabilities.

3-2-3 Initialization

Since the start location is known, the $\{p_{k,0}\}$ is initialized using the degenerate distribution (given in equation 3-4) where x_{t-1} is replaced by x_0. The pseudocode for this step is given in Function Initialization.

Input:
  x_0 :the start location
  $\{x_{k,0}\}$:grid cells at $t = 0$
Output:
  $\{p_{k,0}\}$:initial probability grids
forall the $x_{k,0}$ **do**
  if $x_{k,0} = x_0$ **then**
    $p_{k,0} \leftarrow 1$;
  else
    $p_{k,0} \leftarrow 0$;
  end
end
return $\{p_{k,0}\}$

Function Initialization

3-2-4 Prediction

In prediction step (the pseudocode for this step is given in Function Prediction), the prior probability grids are computed based on three factors:

The previous location. As it is very inefficient to use the entire model of the tracking environment to predict and update the location's probability, we derive a sub model from the entire model based on the previous location, and the sub model is updated at each prediction step. Only the grid cells inside a buffer of the previous location are taken into consideration in later

processes while other grid cells obtain a probability of 0 by default. We set the *radius* of the buffer is 2 times the product of the average walking speed of human (1.4 m/s [33]) and the tracking interval (3 seconds). We think the person should not be outside the buffer within 3 seconds.

The grid model. In order to determine the probability of the derived sub model, we make use of its semantic and topological features. Two semantic features are employed: obstacle and space. If the grid cell is an element of an obstacle, it gets probability as 0, and if the grid cell is at a different space than the space of the last location, its probability is decreased by dividing a constant value W_s. We set W_s as 2 in this case because we assume the probability of the current location in a new space is half of the probability in the old space. Among all topological features introduced in section 3-1-3, the distance difference is chosen. This is because that it can represent the connectivity between two grid cells and it is suitable for probability computation. The shortest path is not used because without the knowledge of the real walking speed, it is difficult to determine the distance threshold of the path. Moreover, the grid cells that are far from the previous location have been filtered out by the buffer so it is not necessary to use an absolute distance. We assume there is a negative relation between the distance difference and the probability, therefore the grid cell with larger distance difference is assigned lower probability. We use a discrete probability distribution for distance difference (see the equation 3-6).

$$P(\Delta d) = \begin{cases} P1 & \text{when} \quad |\Delta d| \leq d1 \\ P2 & \text{when} \quad d1 < |\Delta d| \leq d2 \\ P3 & \text{when} \quad d2 < |\Delta d| \leq d3 \\ P4 & \text{when} \quad |\Delta d| > d3 \end{cases} \tag{3-6}$$

where Δd denotes the distance difference, $d1$, $d2$, $d3$, $d4$ are thresholds of distance difference and $P1$, $P2$, $P3$, $P4$ are corresponding probabilities.

In this research the $d1$, $d2$, $d3$ are set respectively as 1 m, 3 m and 5 m, $P1$, $P2$, $P3$, $P4$ are set receptively as 0.9, 0.7, 0.5 and 0.1. This is because we suppose that there are no obstacles between two grid cells if their distance difference is smaller than 1 m, there are no or a small obstacle between two grid cells if their distance difference is above 1 m but smaller than 3 m, and there are obstacles between two grid cells if their distance difference is larger than 5 m for our testing environment. Thus, the grid cells with distance difference smaller than 3 m get high probability (0.9 or 0.7) and cells that have distance difference larger than 5 m get low probability (0.1). Other cells get medium probability (0.5).

The previous moving direction. If there is no turning at the previous location, we assume that there is a very high probability ($P_{high} = 0.8$) that the current moving direction is same as the previous one and a low probability ($P_{low} = 0.2$) that the moving direction is changed. So we decrease the probability of grid cells outside the moving direction by dividing a constant value W_{md}. W_{md} is 4 in this case because P_{low} is a quarter of P_{high}. To determine if a grid cell is inside the moving direction, we compare the angle between the orientation vector of the grid cell and the moving direction with a angle threshold AT. We set AT as 45 ° in this thesis.

In this step, the spatial model's geometric features (buffer and orientation vector) is used to extract the sub model and compare with the previous moving direction, its semantic (space

and obstacle) and topological features (distance difference) are employed to determine the prior probability.

Input:
x_{t-1} :the location at $t-1$
z_{t-1} - md :the moving direction at $t-1$
M :the entire grid model
Output:
$\left\{ p_{k,t}^{-} \right\}$:prior probability grids

```
// derive the sub model based on the previous location
```
$m \leftarrow$ UpdateModel(M, x_{t-1}) ;
foreach $x_k \in m$ **do**
```
    // compute prior probability using the grid model's features
```
    $p_{k,t}^{-} \leftarrow$ PredictbyModel(x_k, x_{t-1}, m) ;
```
    // compute prior probability using the previous moving direction
```
    if x_k *outside* z_{t-1} - md **then**
        $p_{k,t}^{-} \leftarrow p_{k,t}^{-}/W_{md}$
    end
end

Function Prediction

Input:
M :the entire grid model
x_{t-1} :the location at $t-1$
Output:
m :the sub model

foreach $m_k \in M$ **do**
    $B \leftarrow$ SetBuffer$(x_{t-1}, radius)$;
    if m_k *inside* B **then**
        add m_k to m ;
    end
end
return m;

Function UpdateModel

Input:
x_k :a candidate location at t
x_{t-1} :the location at $t-1$
m :the sub model
Output:
$p(x_k \mid x_{t-1}, m)$:the prior probability of x_k

// $m_{obstacle}$ denotes the obstacle feature of x_k in m
if $m_{obstacle}$ *of* x_k *is true* **then**
  $p(x_k \mid x_{t-1}, m) \leftarrow 0$;
else
  // $m_{\text{distancediff}}$ denotes the distance difference feature of x_k in m
  if $m_{distancediff}$ *of* $x_k < d1$ **then**
    $p(x_k \mid x_{t-1}, m) \leftarrow P_1$;
  else if $m_{distancediff}$ *of* $x_k < d2$ **then**
    $p(x_k \mid x_{t-1}, m) \leftarrow P_2$;
  else if $m_{distancediff}$ *of* $x_k < d3$ **then**
    $p(x_k \mid x_{t-1}, m) \leftarrow P_3$;
  else
    $p(x_k \mid x_{t-1}, m) \leftarrow P_4$;
  end
  // m_{space} denotes the space feature of a cell in m
  if m_{space} *of* $x_k \neq m_{space}$ *of* x_{t-1} **then**
    $p(x_k \mid x_{t-1}, m) \leftarrow p(x_k \mid x_{t-1}, m)/W_s$;
  end
end
return $p(x_k \mid x_{t-1}, m)$

Function PredictbyModel

3-2-5 Update

In this step, the probability grids are updated using new measurements. The pseudocode of this step is given in Function Update. As introduced before, to overcome the influence of a single inaccurate measurement, We use the measurements per 3 seconds to compute a new location. But the orientation of each measurement is checked once it is acquired. If its orientation is different from the previous moving direction, the process to deal with turning is started (see section Turning process). We update the moving direction using the mean value of new measured orientations and use it to filter WiFi measured locations (see section Remove outliers of WiFi measured locations). After removing outliers of WiFi measured locations, the next step is to determine the space property of the current location x_t, namely to check if the new location is at a different space from previous location (see section Determine space property of the current location). Then the probability grid is updated based on the space property and filtered locations. If all WiFi measured locations are filtered out, the prior probability is updated based on the current moving direction (see section Compute posterior probability). The grid cell with maximum posterior probability is taken as the current location. In addition, we check the quality of the computed location every three tracking intervals (9 seconds) using all measurements obtained in this period (18 measurements on average). Details of this process are given in section Tracking quality check.

Remove outliers of WiFi measured locations

We use a set of measurements (z_t) in 3 seconds to compute the person's current location. As the accuracy of WiFi positioning system is influenced by the dynamic environment, there could be outliers inside WiFi measured locations z_t - mls. The number of z_t - mls is not large, so we cannot eliminate the effect of outliers by just averaging all WiFi measured locations. Instead, we remove outliers using hierarchical clustering which groups WiFi measured locations to clusters based on the distance between themselves [34]. This method does not provide a single partitioning of the data set, but instead provide an extensive hierarchy of clusters that merge with each other at certain distances, also known as dendrogram. By cutting the dendrogram at different distance threshold, different clusters can be obtained. There are also multiple choices for distance function, such as single-linkage (the minimum of object distances), average-linkage (the average of object distances) and complete-linkage (the maximum of object distances). Based on our experiments, the average-linkage hierarchical clustering with distance threshold (CDT) of 4 m has best performance and is used by the algorithm. A result of the clustering of WiFi measured locations is given in Figure 3-4, in which the six locations obtained in 3 seconds are clustered into three groups by a threshold $4m$. The locations of the largest cluster (purple points) are considered more accurate than other measured locations (pink and yellow points in the bottom figure). Moreover, We think the moving direction z_t - md is reliable and we also use it to detect the outliers of the WiFi measured locations.

The process to filter out the outliers is given below: At first we look at the number of z_t - mls, if there are very little measurements (only one or two), it indicates that the WiFi signal strength during this period is very poor and WiFi positioning system does not work normally. So we think these WiFi locations are not reliable, namely they are outliers. Otherwise, we cluster the measurements using average-linkage hierarchical clustering. We assume that the distance

between locations in 3 seconds should be smaller than 4 m. If there is more than one cluster, we think only one is trustful while others are outliers. In order to find the most reliable cluster, we compare two features of these clusters: direction and size. The direction of cluster is the orientation vector from the previous location to the cluster center. Finally the cluster whose direction inside the current moving direction and size is largest is kept. The pseudocode of this process is given in Function RemoveOutliers.

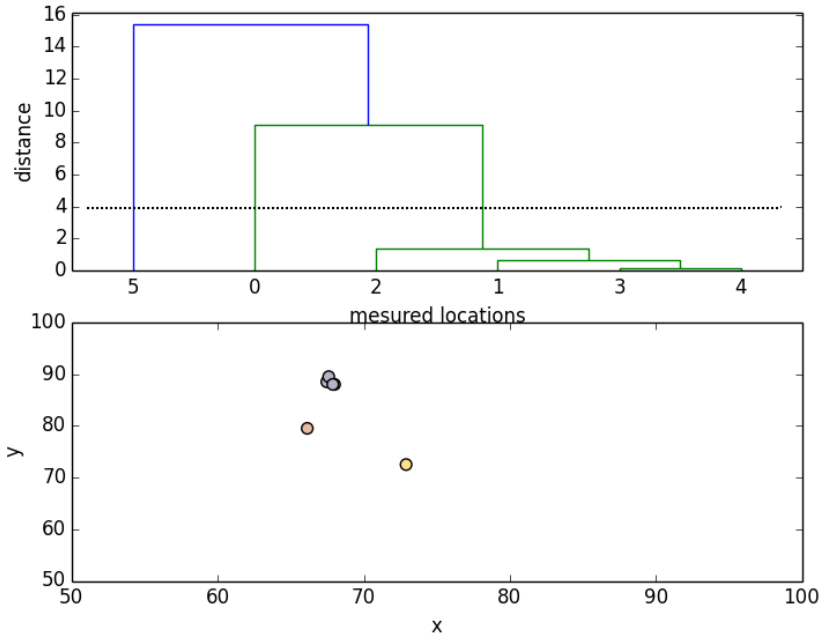


Figure 3-4: The top figure is the dendrogram of average-linkage hierarchical clustering, the bottom figure is the clustering result with distance threshold as 4 m.

Determine space property of the current location

This process is applied after removing outliers of WiFi measured locations. We think the remaining locations are reliable and use them to determine the space property of the current location. In order to check the space property of WiFi measured locations, they are mapped to the grid model based on geometric coordinates. If most WiFi measured locations are still in the previous space, we believe the space of current location is also inside the previous space. When most WiFi measured locations are in a new space, further checking is carried out to confirm the change of space. To avoid ambiguity between two space, we use the door between two space as a transition zone. So when the location is not in a new space for certain, we put the location at the door. The pseudocode of this process is given in Functon DetermineSpace. From it we can see there are two conditions to enter a new space:

1. the door connected two spaces is in the moving direction
2. the previous location is at the door

Only when both conditions are met, the current location is considered to be at the new space. If only condition 1 is met, the current location is determined at the door. Otherwise, the current location is still at the previous space. In this way, the condition for entering a new space is relatively strict. The person has to be first localized at the door before getting into the new space, which also means most WiFi measured locations have to be in the new space twice. The advantage of this strict rule is that ambiguous situations such as locations are measured at wrong space temporally and the person going into and out the space very quickly can be dealt with.

Compute posterior probability

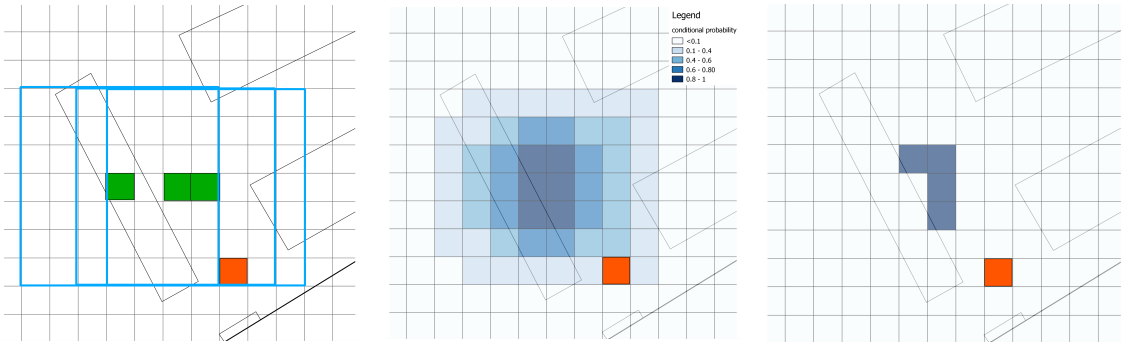
There are two ways to update the position probability grids. If there is no WiFi measured location is available, the probability is updated using the current moving direction, otherwise it is updated using WiFi measured locations. For the later approach, the current moving direction is not reused as that it has been applied to filter the WiFi measured locations.

The second approach is looked at firstly since it is the common way to update the probability. We use a rectangular window function $W(d)$ [35] to compute the conditional probability $\left\{p(z_t^j \mid x_{k,t}, m)\right\}$ based on each WiFi measured location z_t^j. As shown in equation 3-7, the function $W(d)$ is the average of three sub window function with different window size. The usage of the average window function rather than a single window function is to make the conditional probability decrease with the increase of the distance to a WiFi measured location. In order to allow the windows of different measured location overlap, we set the largest windows size as CDT and other two size receptively as two-thirds and third of the CDT. Thus, the grid cell that is close to all WiFi measured locations get highest conditional probability. Then the conditional probability estimated by all WiFi measured locations are accumulated and normalized to get the final conditional probability $\{p(z_t \mid x_{k,t}, m)\}$. Lastly, the posterior probability of a grid cell is computed by multiplying its prior probability and conditional probability. An illustration of this approach is given in figure 3-5.

$$
\begin{aligned}
W_i(d) &= \begin{cases} 1 & \text{for} \quad |d| < \frac{1}{2}n_i \\ 1/2 & \text{for} \quad |d| = \frac{1}{2}n_i \\ 0 & \text{for} \quad |d| > \frac{1}{2}n_i \end{cases} \\
W(d) &= \frac{1}{3}\sum_{i=1}^{3} W_i(d) \\
(n_1 = CDT, n_2 = \frac{2}{3}CDT, n_3 &= \frac{1}{3}CDT)
\end{aligned}
\tag{3-7}
$$

where $|d|$ denotes the distance from a grid cell to the a WiFi measured location, n_i is the size of the rectangular window W_i.

When all WiFi measured locations are filtered out, the posterior probability is calculated based on the current moving direction. We compute orientation vectors from the previous location to the grid cells with the highest prior probability, then find the grid cell whose orientation vector is closest to the current moving direction. Except this grid cell, other grid cells are updated to 0. The pseudocode to compute posterior probability grids in both ways are given in the function ComputePosteriorProbability.



(a) rectangular window function **(b)** conditional probability **(c)** posterior probability

Figure 3-5: Update using WiFi measured locations: the red cell is the previous location, (a) the green cells are WiFi measured locations and blue rectangles are window function, (b) the conditional probability is represented by the darkness of the blue, the darker cell the higher probability, (c) the blue cells are cells with maximum posterior probability.

Find the current location

The grid cell with largest posterior probability is considered to be the current location. However, there could be multiple grid cells with the maximum probability, in this case, we choose the grid cell that is closest to the center of these grid cells.

Tracking quality check

Since the locations may be continuously estimated without using WiFi measured locations, there could be errors caused by inaccurate moving directions. In order to control this error, the location x_t is checked per three tracking intervals. We use three tracking intervals (from $t-3$ to t) because on average 18 measurements (in 9 seconds) can be obtained during this period and we suppose there should be reliable locations inside these measurements. All measurements obtained during this period are used to compute a location x_{check} and an average moving direction md_{check}. We compute x_{check} using the same processes to compute other locations but with different parameter settings. The buffer to update the grid model and the clustering threshold are enlarged because we use measurements of a longer period (three tracking intervals). Then the location x_t is qualified by checking its distance to x_{check} and the angle difference between its orientation vector z_t - md and md_{check}. The threshold of distance (DT) and angle (AT) are set respectively as $8m$ and $45\,^\circ$ because we think the x_{check} is at the middle of the path from $t-3$ to t and its distance to x_t should not be larger than 8 m according to the average walking speed (1.4 m/s [33]). If the location x_t is unqualified, it is replaced by x_{check}. The pseudocode of this process is given in function TrackingqualityCheck.

Turning process

The turning process is triggered when a measured orientation is outside the previous moving direction. In order to make sure this change is really caused by turning rather than other coincidences (e.g., rotation of the mobile device), more measurements need to be acquired. If

the average orientation of these measurements is also outside the previous moving direction, then we confirm there is a turning.

In order to adapt the general tracking procedure (prediction and update) to the case with turning, we introduce the concept of turning point. We assume that the direction change occurs only at the turning point. Thus, the locations before this point have the previous moving direction and locations after this point have a new moving direction. We use the turning point to replace the previous location, and then the general tracking procedure is suitable again. The pseudocode of this process is given in function TurningProcess.

The approach to find the turning point is explained below: The main criterion for the turning point is that it must be inside the moving direction of the previous location as well as the opposite moving direction of the current location. To find the turning points, we firstly estimate the current location at the center of WiFi measured locations, then use this estimated location to back track the turning point. Since there could be multiple grid cells inside the intersection of two directions, more constraints are applied to find the most reasonable one: it should be at the same space of previous location, it should not be inside any obstacle, the paths from the previous location to the turning point and from the turning point to the current location should not blocked by any obstacle. An example of turning point is given in figure 3-6.

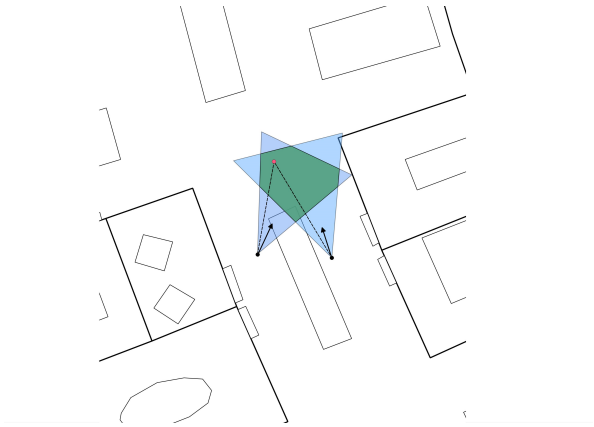


Figure 3-6: Turning point: the right black point and vector (with the blue rectangle) are the previous location and its moving direction, the left black point and vector (with the blue rectangle) are the current location and its opposite moving direction. The red point is the turning point and the dash line is estimated turning path.

3-2-6 Floor change process

As the spatial model and WiFi measured locations are all two dimensional, it is impossible to estimate the moving path between floors. But is possible to detect the change of floors. In the offline training phase of WiFi positioning (see section 2-1), the radio map is measured at each floor and the floor number is also stored. Thus in the online localization phase of WiFi positioning, the floor number of the matched location can be obtained. We define the condition of floor change below: If a WiFi measured location is matched to the radio map in different floor and the previous location is close to a vertical passage, we consider the person

is walking between floors, and when most WiFi measured locations of a tracking interval are located in a new floor, we assume the person is at the new floor now.

Input:
$\left\{ p_{k,t}^{-} \right\}$:prior probability grids
z_t :measured orientations $z_t\text{-}mds$ and locations $z_t\text{-}mls$ from $t-1$ to t
$\{x_{t-1},\ z_{t-1}\text{-}md,\ S_{t-1}\}$:the location, the moving direction and the space at $t-1$
m :the sub model
Output:
x_t :the location at t
$z_t\text{-}md$:the moving direction at t

if $\forall\, md_i \in z_t\text{-}mds\ :\ md_i$ *inside* $z_{t-1}\text{-}md$ **then**
  $z_t\text{-}md \leftarrow average(z_t\text{-}mds)$;
  $z_t\text{-}mls \leftarrow RemoveOutliers(z_t\text{-}md, z_t\text{-}mls)$;
  if $size(z_t\text{-}mls) > 0$) **then** `// update using WiFi measured locations` $z_t\text{-}mls$
    $Z_t \leftarrow z_t\text{-}mls$;
    $S_t \leftarrow DetermineSpace(z_t\text{-}mls, z_t\text{-}md, S_{t-1}, m)$;
    $\{\, p_{k,t} \,\} \leftarrow ComputePosteriorProbability(Z_t, S_t, \left\{ p_{k,t}^{-} \right\})$;
  else `// update using the moving direction` $z_t\text{-}md$
    $Z_t \leftarrow z_t\text{-}md$;
    $S_t \leftarrow S_{t-1}$;
    $\{\, p_{k,t} \,\} \leftarrow ComputePosteriorProbability(Z_t, S_t, \left\{ p_{k,t}^{-} \right\})$;
  end
else
  $\{\, p_{k,t} \,\} \leftarrow TurningProcess(x_{t-1}, z_{t-1}\text{-}md, z_t, M)$;
end
$x_t \leftarrow FindCurrentLocation(\{\, p_{k,t} \,\})$;
`// the computed location is checked per three tracking intervals`
if $t \bmod 3 = 0$ **then**
  $x_t \leftarrow TrackingqualityCheck(x_t, z_t\text{-}md, M)$;
end
return $x_t,\ z_t\text{-}md$

Function Update

Input:
z_t - mls :measured locations
z_t - md :the moving direction at t
Output:
$\bar{z}_t$ - mls :filtered locations

$C \leftarrow$ hierarchical clustering of z_t - mls ;
if $size(C) > 1$ **then**
  foreach $c \in C$ **do**
    if c *outside* Z_t - md **then**
      remove c from C ;
    end
  end
  `// return the cluster containing most measured locations`
  foreach $c \in C$ **do**
    if $size(c) = max$ **then**
      return c;
    end
  end
else
  `// when there is only one cluster, all measured locations are returned`
  return z_t - mls;
end

Function RemoveOutliers

Input:
z_t - mls :measured locations(filtered),
z_t - md :the moving direction at t
S_{t-1} :the space at $t-1$
m :the sub model
Output:
S_t :the space at t

// S_{majority} denotes the space property of the majority of measured locations
find S_{majority} of z_t - mls ;
if $S_{majority} = S_{t-1}$ **then**
  $S_t \leftarrow S_{t-1}$;
else
  if $\exists\, door \in m : door$ *connects* $S_{majority}$ *and* S_{t-1} **and** *door is inside* z_t - md **then**
    if S_{t-1} *at door* **then**
      $S_t \leftarrow S_{\text{majority}}$;
    else
      $S_t \leftarrow door$;
    end
  else
    $S_t \leftarrow S_{t-1}$;
  end
end
return S_t

Function DetermineSpace

Input:
Z_t :the selected measurements at t, could be z_t - mls or z_t - md
S_t :the space at t
$\left\{p_{k,t}^{-}\right\}$:prior probability grids
Output:
$\{p_{k,t}\}$:posterior probability grids

if $Z_t = z_t$ - mls **then** `// update using WiFi measured locations` z_t - mls
  forall the k **do**
    `// initialize` $p(z_t \mid x_{k,t}, m)$
    $p(z_t \mid x_{k,t}, m) \leftarrow 0$;
    if $x_{k,t} \in S_t$ **then**
      `// apply the rectangular window function to each measured location`
      foreach $z_i \in z_t$ - mls **do**
        $d_i \leftarrow distance(x_{k,t}, z_i)$;
        compute $W(d_i)$;
      end
      $p(z_t \mid x_{k,t}, m) \leftarrow \prod W(d_i)$
    end
    $p_{k,t} \leftarrow \alpha\, p(z_t \mid x_{k,t}, m)\, p_{k,t}^{-}$;
  end
else `// update using the current moving direction` z_t - md
  forall the k **do**
    `// initialize` $p(z_t \mid x_{k,t}, m)$
    $p(z_t \mid x_{k,t}, m) \leftarrow 0$;
    if $x_{k,t} \in S_t$ **and** $p_{k,t}^{-} = max$ **then**
      compute $angle_{diff}$ between $vector(x_{t-1}, x_{k,t})$ and z_t - md ;
      if $angle_{diff} = min$ **then**
        $p(z_t \mid x_{k,t}, m) = 1$;
      end
    end
    $p_{k,t} \leftarrow \alpha\, p(z_t \mid x_{k,t}, m)\, p_{k,t}^{-}$;
  end
end
return $\{p_{k,t}\}$

Function ComputePosteriorProbability

Input:
x_t :the location at t
z_t - md :the moving direction at t
M :the entire grid model
Output:
:the qualified location

x_{t-3}, z_{t-3} - $md \leftarrow$ get the previous location and moving direction at $t-3$;
$z_{t-3:t} \leftarrow$ get all measurements from $t-3$ to t ;
$\left\{p_{k,t}^{-}\right\}, m \leftarrow Prediction(x_{t-3}, z_{t-3}\text{-}md, M)$;
$x_{check}, md_{check} \leftarrow Update(\left\{p_{k,t}^{-}\right\}, z_{t-3:t}, z_{t-3}\text{-}md, m)$;
if *DistanceDiff*$(x_t, x_{check}) < DT$ **and** *AngleDiff*$(z_t\text{-}md, md_{check}) < AT$ **then**
  return x_t ;
else
  return x_{check};
end

Function TrackingqualityCheck

Input:
x_{t-1} :the location at t-1
z_{t-1} - md :the moving direction at t-1
z_t :measured orientations and locations from $t-1$ to t
M :the entire grid model
Output:
turning result

$z_{t+1} \leftarrow$ get more measurements ;
$md_{average} \leftarrow Average(\{z_{t+1}\text{-}mds\})$;
if *AngleDiff*$(z_{t-1}\text{-}md, md_{average}) > AT^\circ$ **then** `// AT is a threshold of angle difference`
  $x_{turn} \leftarrow$ find the turning point ;
  $\left\{p_{k,t}^{-}\right\}, m \leftarrow Prediction(x_{turn}, M)$;
  $x_t \leftarrow Update(\left\{p_{k,t}^{-}\right\}, z_{t:t+1}, m)$;
  return *true*;
else
  return *false*;
end

Function TurningProcess

Chapter 4

Implementation and Testing

To answer research question 3: *How can the spatial model aided tracking algorithm be implemented and tested?*

This chapter introduces the architecture of tracking system and explains how each layer of this system is implemented. Five tests are done using this system to check the performance of this tracking algorithm in a live environment. They respectively look at the algorithm's ability to track a pedestrian in different situations, such as in one space, between two spaces, with turning and in a short distance.

4-1 Preprocessing

4-1-1 Build grid model

This section describes how a hybrid grid model containing geometry, topology and semantics is constructed. The raw data to build the model is the floor plan of the selected environment. The general modeling flow is depicted in figure 4-1.

Extract geometry

The grid cells are generated automatically by using GIS software, but two parameters need to be set: extend and granularity (grid size). Apparently, the former one should be set as the extent of the floor plan.

The criteria for choosing a suitable grid size are:

1. Important indoor objects e.g., door, obstacle can be represented by grid cells. Namely, the grid size should be smaller than these objects' size.

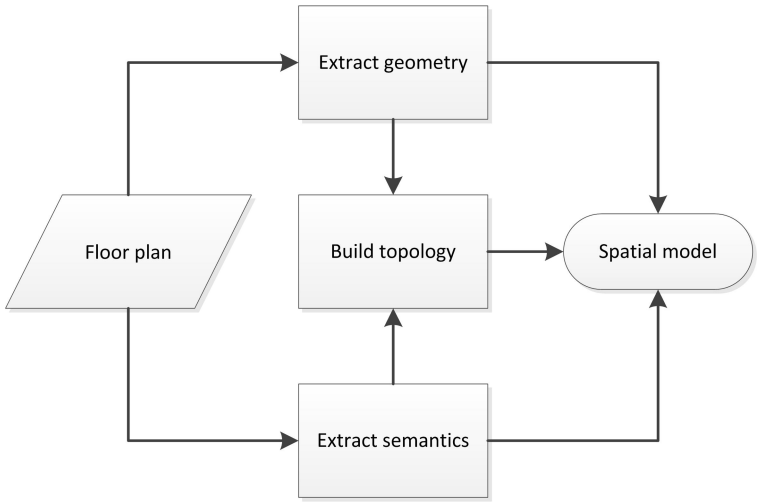


Figure 4-1: Modeling flow

2. The occupancy of a grid should be as few as possible, the ideal situation is that each grid cell is only occupied by one space or object. This enables a grid cell to have a unique attribute.

3. The human factors such as step length or average walking speed should also be taken into consideration.

4. The size of the model should not be too large. With increasing the number of grid cells, the storage and computational load also go up and thus a too small cell size should be avoided.

In order to find appropriate grid granularity for the tracking environment, a series of grid sizes ranging from 0.5 meters to 4 meters are compared in table 4-1. The total number of grid cells, their intersections with indoor space and objects and the uniqueness of grid cells are listed with respect to the size of grid. More intersections indicates more details of the environment can be represented. The uniqueness of grid cells shows the percentage of grid cells that have unique occupancy.

granularity	number of grid cells	intersection	uniqueness
4m	72	231	31.2%
2m	257	528	48.6%
1m	944	1442	65.5%
0.7m	1725	2554	67.5%
0.5m	3598	4442	81%

Table 4-1: Comparison of grid granularity

From the comparison in table 4-1, it is notable that when the granularity is smaller than 1 meter, most grid cells have unique occupancy. With the decrease of granularity, the intersections and uniqueness grows up but the number of grid cells also increase. Considering the localization frequency of WiFi positioning system is 0.5 second and the averaging walking

speed of human is 1.4 m/s [33] per second. A very fine granularity (less than 0.7m) is not necessary for the localization purpose. Moreover in the algorithm, we integrate a grid filter with the model and the common used size of grid filter is between 10 cm and 1 m according to the literature (see section 2-3-2). Therefore 0.7 m is used by this thesis.

Extract semantics

There are three types of semantic features: space, door, obstacle (introduced in section 3-1-2). For each grid cell, it must have one and only one space feature, but may have null or one object feature (door or obstacle). We derive the space and object feature of a grid cell by intersecting it with the floor plan. To ensure each grid cell has a unique space or obstacle feature, a filtering based on area is applied to the initial results of intersection. If a grid cell intersects with multiple spaces, than only the space that has the most occupancy is assigned to this grid cell. If a grid cell is occupied by obstacles the grid cell is determined to be obstacle only when more than half of it is occupied. As the door's geometry in the floor plan is small and it hardly occupies half or more of a cell when it intersects with multiple cells, we assign door feature to a cell once it intersects with a door. However when a cell intersects with more than one door, only the door that has the most occupancy is assigned to this grid cell.

Build topology

We first create the adjacency graph that describes eight neighbors of a grid cell (for the grid cells at the boundary, there are less than eight neighbors). Then the connectivity of grid cells is derived based on the adjacency graph. A grid cell, which is not obstacle, is connected to its adjacent neighbors that have the same space property. Grid cells in different space are connected only through door cells. The final grid model is shown in figure 4-2.

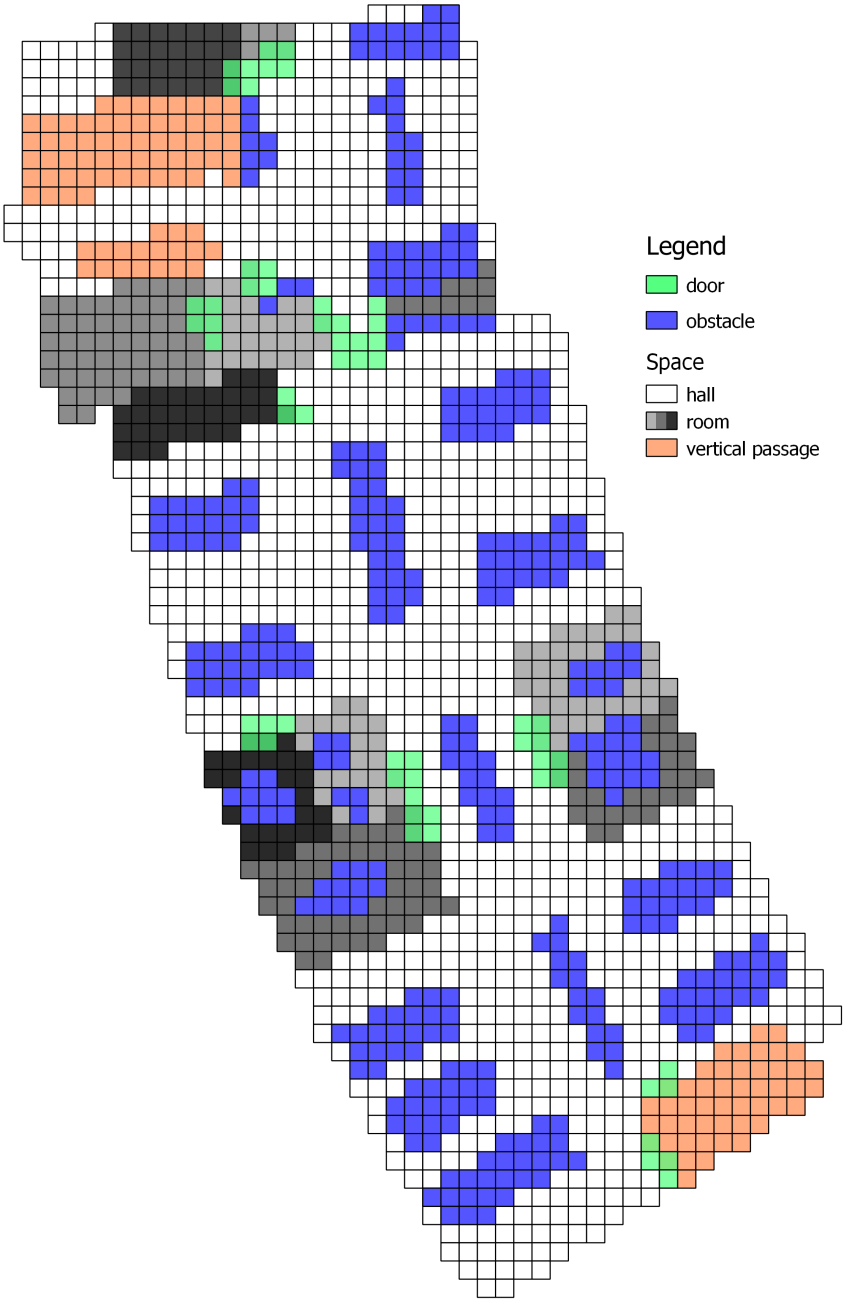


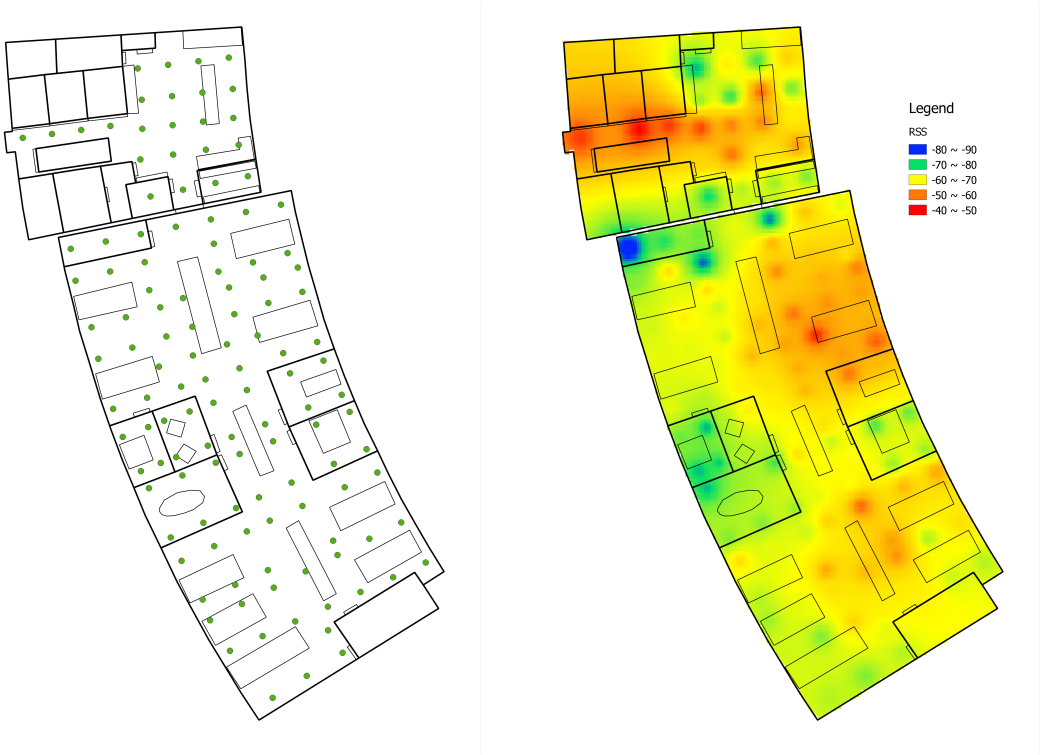
Figure 4-2: Grid model

4-1-2 Set up Mobile device

Two types of data are measured real time for tracking: location and orientation. The first one is obtained by the WiFi positioning system and the second one is acquired by the magnetometer sensor of the mobile device.

WiFi positioning system

The first step is to create the radio map of the given area based on the RSSI data from all fixed APs. To get high positioning accuracy, we have dense sampling points (see figure 4-3a). The sampling interval ranges from 2 to 3 meters influenced by the obstacles in the indoor space. Then the sampling data is processed to generate the radio map (see figure 4-3b). During the online phase of WiFi positioning, the live RSS values are then compared to the radio map to find the closest match.



(a) WiFi fingerprint sampling points **(b)** Visualization of the radio map

Figure 4-3: Radio map

The accuracy of WiFi positioning system is tested. According to the developer of the WiFi positioning system, 90% locations have an accuracy of 3.4m. However, in our testing environment the accuracy is lower. This may be due to the influence of the Wireless LAN Controller in our environment that adjusts the signal strength of access point based on the environment. We compare the positioning accuracy in 4 locations table 4-2. It is notable that the position accuracy is not the same everywhere, in some places the wifi locations are very accurate (0.9m) but in other place the positioning error is 10 times higher (9.2m).

location	stand deviation of x	stand deviation of y	accuracy
l1	1.68m	3.02m	9.2m
l2	1.14m	0.31m	4.8m
l3	1.89m	1.98m	3.2m
l4	1.47m	1.75m	0.9m

Table 4-2: Comparison of positoning accuracy

Magnetometer

We use the magnetometer of the mobile device to measure its orientation. The coordinate system of mobile device is shown in figure 4-4. We use the angle between the magnetic north direction and the y-axis of mobile device to represent the orientation of the person because we ask people to hold the mobile device flat with the top forward. We test the accuracy of the measured orientation in two cases: still and walking in straight line. The angle's stand deviation of the first case is around 2 degree and the stand deviation of the second case is around 9 degree. The person is likely to shake the mobile device during walking and thus the latter case has larger variance.

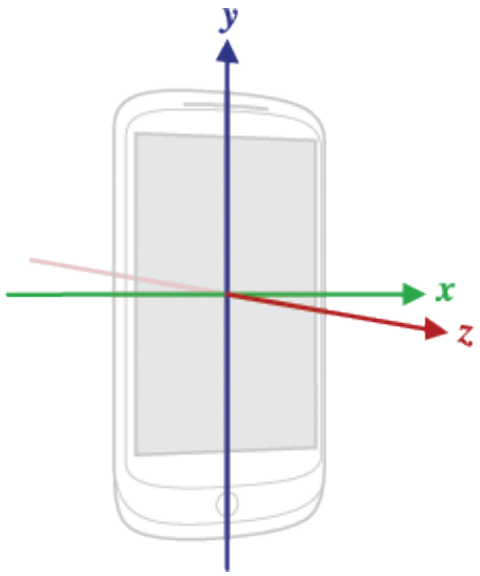


Figure 4-4: Mobile device axis

4-2 Implementation of tracking system

The architecture of the system is given in figure 4-5. The system has a three-layer architecture: the bottom layer is database where the spatial model is stored, the middle layer is web service where the tracking algorithm is implemented, the top layer is mobile device, where the measurements (location, orientation and time) are acquired and the tracking results are presented. The web service also has three layers: data access object (DAO) layer where the data from database is mapped to the web server, tracking service layer where the tracking

algorithm is implemented, and controller layer where the communication between mobile device and service is controlled. However due to the limitation of time, the two parts indicated by dash line are not implemented. Currently the mobile client is not connected to the service in real time, thus the tracking is offline, which means the trajectory is computed after the walking, and the computed trajectories are stored in files instead of in database now. However, these two functions do not influence the research of this thesis. The performance of the tracking algorithm can still be tested using the incomplete system.

As mentioned before, this thesis is carried out in the company CGI group who has a lot experience on system development. The experts from the company recommend the architecture of the system considering following advantages:

- The mobile client is lightweight. We put most computation load at the sever layer so that the mobile client is only used to collect data and present the tracking results.
- The low requirements for data transmission. The model data in need can be sent to the sever at start of the application, and during the tracking process the sever can retrieve the model data locally. Between the mobile client and server, only a few data is exchanged so it will not cause transmission problem.
- The system has high efficiency. Since the server has strong computational capacity, the performance of the system is not restricted by the time complexity of the tracking algorithm.

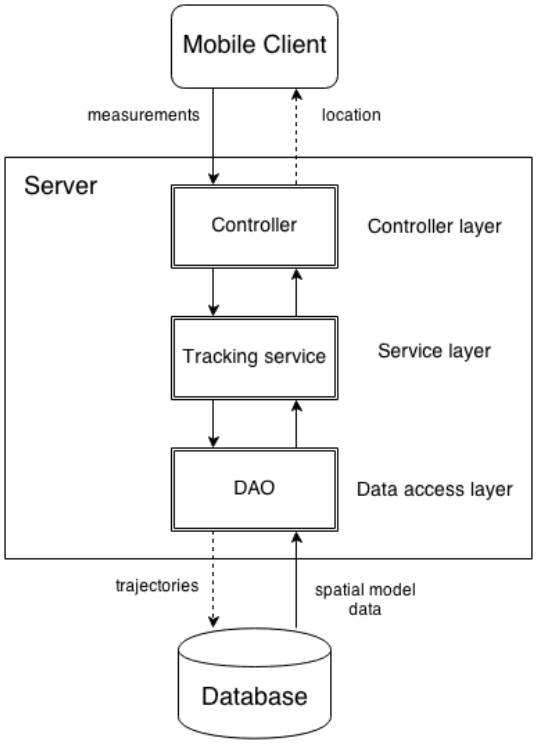


Figure 4-5: Tracking system structure

4-2-1 Database

For this system, the main function of database is to store the grid model. The open source spatial database PostGIS is used by this thesis. The entity-relationship diagram (ER diagram) of the grid model which illustrates the relationships between entities in the database [36], is given in figure 4-6. We can see the core component of the model is the grid cell that has relationships with all other components. It has many to one relationship with space, obstacle and door. Each space, obstacle or door is comprised of several grid cells. Every grid cell must be related to one and only one space but may be related none or one object (door or obstacle). To represent the connectivity between grid cells, the grid table is also related to itself. A grid cell can have at most 8 neighboring cells. The grid cell entity has an attribute of geometric coordinates. The space entity has three attributes: space_type which describes the structure type of the space (e.g., room, vertical passage, hall, etc.), usage_type of the space (e.g., work, print, washroom, etc.) and the floor of the space. The obstacle has an attribute of obstacle type (e.g., table, couch, shelf, etc.). The door is the connection between two spaces so that it has an attribute of the connected spaces (we assume the door can be opened from both sides, so the direction of connection is not included). All entities and the relationship 'Has neighbors' are converted to tables in the database and other relationships are represented through foreign keys.

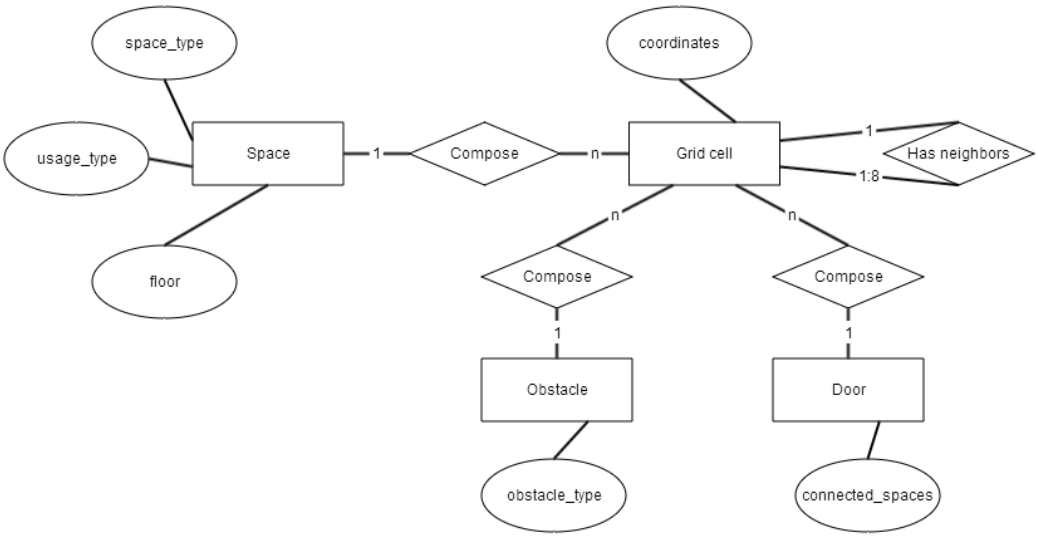


Figure 4-6: Entity-relationship diagram

4-2-2 Web service

The web service is the most important part in this system, because most of the computation is executed here. We build the web service using java technology. The program on the web service is developed using Spring framework (it gives infrastructural support for programming) and published on an open source web server Apache Tomcat. The combination of

spring framework and tomcat server is widely used for web-based enterprise application and recommend by CGI Group Company.

The web service has three layers: Data access object (DAO) layer is responsible for communication with database. It maps the model data from database to the server so that the computations based on the model can be carried out locally. This increases the efficiency of the system greatly. The DAO layer is implemented using the DAO module of Spring framework [37] with an open source object-relational mapping (ORM) library hibernate spatial [38]. The benefits of DAO module and ORM are that they allow data persistence and high performance of the application.

The tracking service layer is where the proposed tracking algorithm is implemented. The program on this layer is mostly written by the author except two libraries used for shortest path computation (PatherFinder [39]) and clustering (Hac [40]).

The controller layer is created to manage request-response between the mobile client and the server.

4-2-3 Mobile client

The mobile client is developed based on Android 4.0+ operating system. The radio map of the indoor environment is stored on the mobile client in xml format. The main function this mobile client is to measure the location, orientation, and time of the mobile device. We developed an application which integrates the WiFi positioning system with the module to obtain the orientation and time from magnetometer and the device's clock. Due to the limitation of time, the real-time communication between the mobile client and the web server is not realized yet.

4-3 Experiment results and analysis

4-3-1 Experiment contexts

The testing environment is the office of CGI Group company, at the A wing of the 8th floor (around 780 m^2) and the middle part of the 10th floor (around 220 m^2). On the 8th floor 63 WiFi APs are used for WiFi positioning and On the 10th floor 30 WiFi APs are used. These APs are not only from 8th or 10th floor but also from other floors.

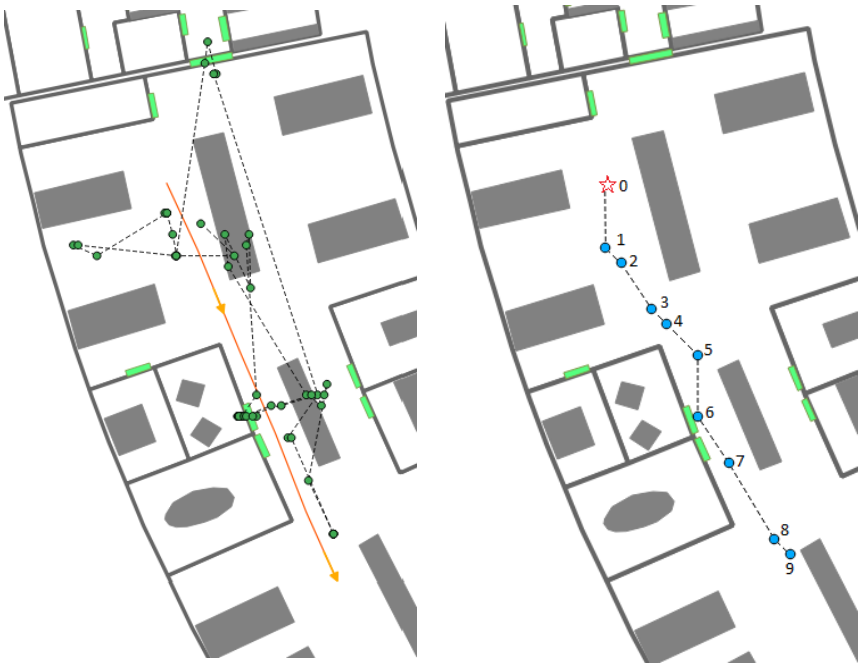
The mobile device used for testing is Motorola Moto G. Its OS is Android 4.4.2 and it has WiFi function and magnetometer sensor.

We set the tracking interval of the algorithm as 3 seconds. The tests given below were carried out at different times.

4-3-2 Case 1: Walking inside a space

The first case is walking inside a space. The objective of this test is to check if the proposed tracking algorithm can derive the moving trend of the walking person correctly and localize

the person at an appropriate place (not inside obstacle and not in wrong space). The test results are given in figure 4-7. From the result, we can see though there are some obvious errors in the raw measurements of the WiFi positioning system, our algorithm still gets an accurate trajectory of the person. No location is inside obstacle or at a wrong space and the moving direction is correct.



(a) wifi measured locations **(b)** tracking results

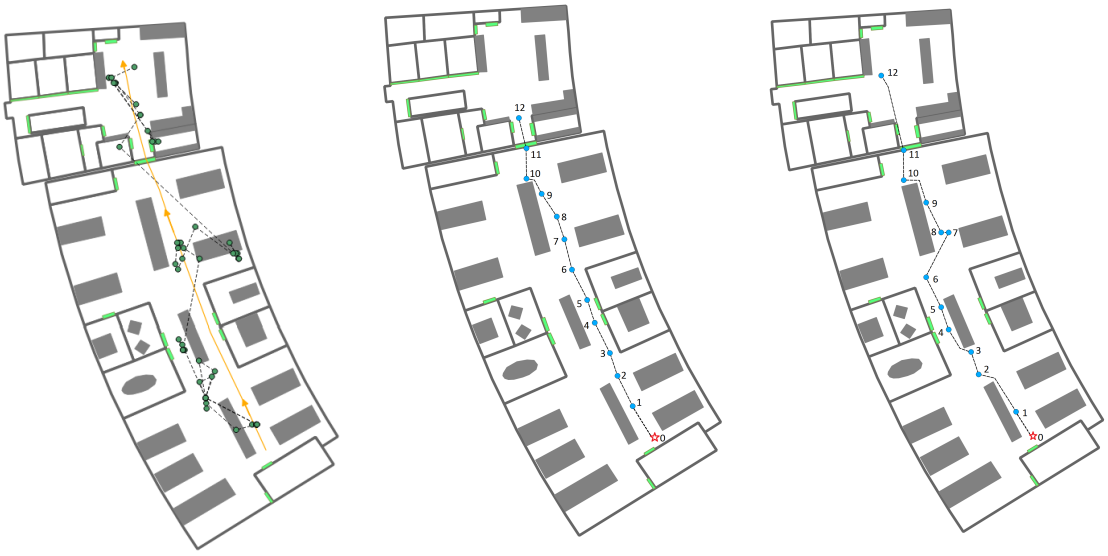
Figure 4-7: Tracking inside a space: (a) the green points are WiFi measured locations, the orange line is the actual walking path of the person. (b) the red star is the start location and the blue points with the sequence numbers are locations computed by the tracking algorithm.

As introduced in section 3-2-5, a location can be computed based on either WiFi measured locations or the moving direction. Later we call the location obtained via the first method measured location and the location obtained via the second method estimated location. For the tracking result of this case, the first two location point 1 and 2 are measured locations, points 3 to 5 are estimated due to the fact that no accurate WiFi measured location are available during this period. After that the location 6, 7 and 8 are measured locations again. It is notable that even though many WiFi measured locations used to compute location 6 are located inside a neighboring space, the location 6 is still calculated correctly in the old space by our algorithm. The last location 9 is estimated. In summary, the proposed algorithm has good performance on filtering unreliable WiFi measured locations, and it can provide reasonable estimation of person's location when no accurate WiFi measured location is available.

4-3-3 Case 2: Walking between spaces

The second case is walking between spaces. This test aims to check the algorithm's capacity on switching the person's locations between two spaces. We can see both locations 11 in

figure 4-8b and figure 4-8c are at the door connecting two spaces. This is because it's first time most WiFi measured locations are in a new space and the algorithm is not sure that the person really enters the new space. So the location is put at the transition zone between two spaces. The next time when most WiFi measured locations are in the new space again, the algorithm confirms that the person enters the new space and computes the location based on WiFi measured locations (location 12). It can be seen that there is a delay for entering a new space caused by the algorithm's conformation process. However the delay is not obvious for one tracking interval (3 seconds). So we think the tracking algorithm's performance on space change is also quite well.



(a) WiFi measured locations

(b) tracking results with more estimated locations

(c) tracking results with more measured locations

Figure 4-8: Tracking between two spaces: (a) the green points are WiFi measured locations, the orange line is the actual walking path of the person. (b) and (c) the red star is the start location and blue points with sequence numbers are locations computed by the tracking algorithm.

In addition, this test also looks at the influence of the ratio of measured locations and estimated locations in a trajectory on the accuracy. Since there are always inaccurate WiFi measured locations during tracking, the trajectories always consist of both measured locations and estimated locations. It is significant to know which type of location is more accurate. A comparison is given by figure 4-8b and figure 4-8c. They use the same data to compute the person's walking trajectory, but the test of figure 4-8b has a smaller tolerance for inaccurate WiFi measured locations than the test of 4-8c . Therefore more locations in this test are estimated, for example the locations from 2 to 6 are all estimated location in 4-8b while in the test of figure 4-8c, these locations are all measured location except location 6. It's notable that the estimated locations are closer to the actual path in this case. We think the estimated location could be more accurate than measured location for a short walking distance when the moving direction is not changing, however there is also error caused by continuous estimations which can be seen in the later test.

4-3-4 Case 3: Walking with turning

The third case is to check if the algorithm is able to deal with turning situations. The experiment result in figure 4-9 shows that the algorithm can find a reasonable turning point in the indoor environment and properly handle the change of moving direction.

In this test, it can be seen that in the middle part of the trajectory, locations are computed at the wrong side of the obstacle. On the way from bottom to top, the error firstly happens at location 4 which is located at the wrong side of the obstacle, after that even though new WiFi measured locations are in the right side of the obstacle, the next location is still put at wrong side because the path to the other side is blocked. Until the location 8 that is at the end of the obstacle, the trajectory is adjusted. The same situation can be also seen on the way back from location 14 to 18. This type of error is caused by two factors:

1. the WiFi positioning system is not very accurate. This causes the algorithm to determine the location at the wrong side.
2. the algorithm takes the constraint of indoor environment into consideration so that the error can not be corrected immediately. We think this error is acceptable therefore it does not influence the general tracking trend and can be corrected over time.

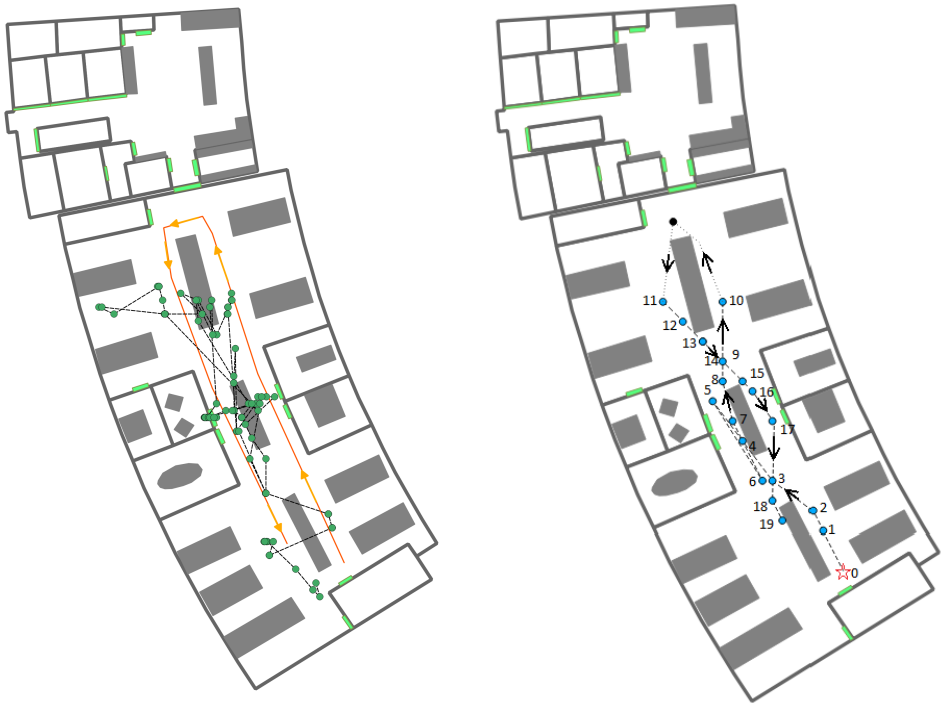
In addition, it is notable that the moving direction from 5 to 6 is opposite to the actual moving direction. This is due to the control of estimation error which is explained in the section 4-3-6.

4-3-5 Case 4: Comparison of tracking performance on short and long path

The fourth case studies if the tracking distance has an effect on the performance of the algorithm. From the figure 4-10, we can see both the trajectories of short distance and long distance deviate from the actual path inside the orange box. This error looks less serious for the long trajectory because it is corrected later. The similar example can also be seen in the result of the last test. Based on these results, we think the tracking performance can be improved over time or tracking distance.

4-3-6 Case 5: Estimation error control

The fifth case looks at the error caused by estimation and how the algorithm prevents the growth of this error. The test in figure 4-11a shows the maximum estimation error allowed by the algorithm. In figure 4-11b the location 1 and 2 are estimated locations because the WiFi measured locations are not inside the moving direction of the location 0. When computing location 3, the estimation error is over the threshold. The location 3 is adjusted by using WiFi measured locations obtained in a long period (from 1 to 3). Thus, the location 3 goes back. After that, location 4 and 5 are computed using WiFi measured locations. It proves that the estimation error will not increase over time.



(a) WiFi measured locations **(b)** tracking results

Figure 4-9: Tracking with turning: (a) the green points are WiFi measured locations, the orange line is the actual walking path of the person. (b) the red star is the start location, the blue points with sequence numbers are locations computed by the tracking algorithm and the black point is the turning point.

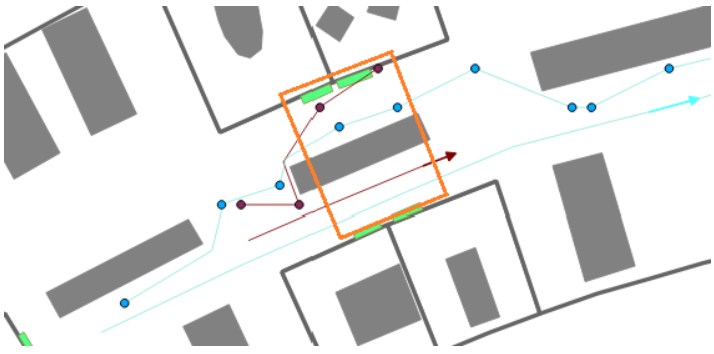
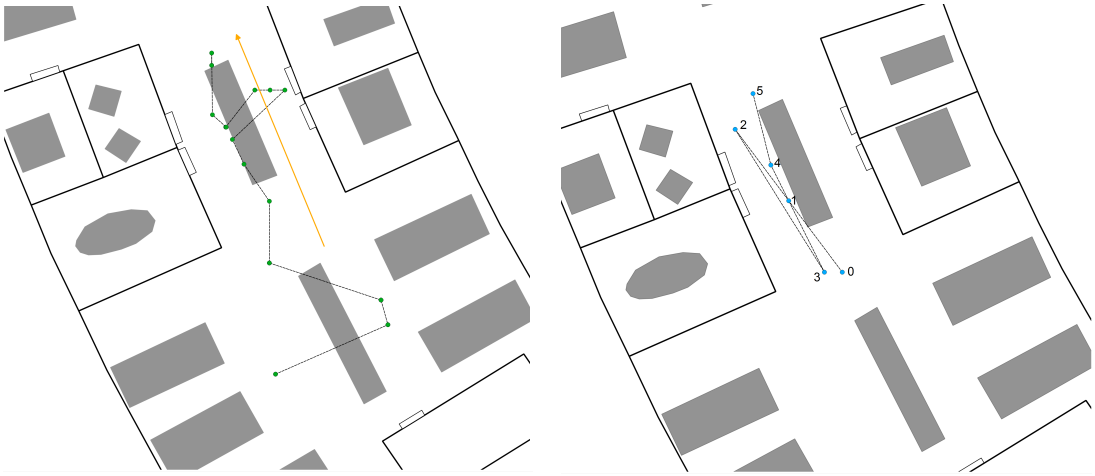


Figure 4-10: Comparison of short and long path: the brown points and vector represent the locations computed by the algorithm and actual path of a short walking, the blue points and vector represent the locations computed by the algorithm and actual path of a long walking.



(a) WiFi measured locations **(b)** tracking results

Figure 4-11: Control of estimation error: (a) the green points are WiFi measured locations for the locations 1 to 5 in (b), the orange line is the actual walking path of the person. (b) the the blue points with sequence numbers are locations computed by the tracking algorithm.

Chapter 5

Conclusion and future work

5-1 Conclusion

In this thesis, a spatial model-aided tracking algorithm based on WiFi positioning system is proposed and its performance is tested in a live environment. The conclusion is made with respect to the three sub research questions:

1. what features of spatial model could be employed for tracking?
2. In what way can the selected features be integrated together to decrease tracking errors of WiFi positioning system?
3. How can the spatial model aided tracking algorithm be implemented and tested?

5-1-1 Spatial model

The grid model is suitable for tracking purpose. We compared several spatial models based on different decompositions of the indoor environment through literature study. We consider the grid model with regular subdivision of the indoor space is more appropriate because of following advantages: It is easy to design and maintain. It has high flexibility. It is able to provide accurate location data. Moreover, it is powerful for computation.

All geometric, topological and semantic features can contribute to improving the tracking performance. As for geometric features, geometric coordinates, buffer, orientation, and distance are useful for tracking. The geometric coordinates is employed to match the measurements of WiFi positioning system to the model. The buffer of the previous location is used to derive a smaller model for computing the current location. The orientation vector from the previous location to a grid cell is compared with the measured direction by the mobile device to determine if the grid cell has a high probability to be the current location. The distance is used to derive the distance difference feature that is one determinant of the prior probability of the current location. As for semantic features, space, door and obstacle are employed.

First these semantic features with the geometric coordinates are used together to derive the connectivity graph of the grid model. Then they are included in both prediction and update steps of the tracking algorithm. The 'space' is a determinant of the prior probability and it is used to determine the space property in the update step. The 'obstacle' is also a determinant of the prior probability it is used with the space to find the turning point. The 'door' plays an important role in the change of spaces. It works as a transition zone between two spaces and successfully avoids ambiguous situations between spaces. As for topological features, various features are derived, e.g., adjacency graph, connectivity graph, shortest path distance and distance difference. These features are strongly related but not all of them are used. The distance difference and connectivity graph are employed by the algorithm respectively for prediction and finding the turning point.

5-1-2 Tracking algorithm

The tracking algorithm integrates all selected features of the spatial model as well as measurements from WiFi positioning system and magnetometer using the grid filter. The grid filter is a technique initially used for mobile robot localization and tracking, but we successfully adapted it to pedestrian tracking. In the prediction step of the grid filter, the geometric features (buffer, orientation vector), topological features (distance difference) and semantic features (space, obstacle) are combined to determine the prior probability of the location. In the update step of the grid filter, the geometric coordinate feature is used to match the measured locations to the model, and the combinations of different types of features are employed for special cases, such as space change and turning. The orientation vector together with semantic features of space and door are used to determine the space property of locations at the border of two spaces. The orientation vector, the semantic features of space and obstacle, and connectivity graph are integrated to find the turning point.

The tracking algorithm can reduce the tracking errors of the WiFi positioning system. We have tested the tracking algorithm for various cases in a live environment tracking. Our test results shows that the algorithm can not only adjust the inaccurate locations measured by the WiFi positioning system to a more reasonable place but also estimate a reliable location when the quality of WiFi positioning is too poor. Moreover, the algorithm has the ability to deal with complicated tracking cases like turning and passing spaces. However, it is difficult to assess the tracking results quantitatively because we don't know the precise walking path. But by qualitatively comparing the computed trajectory and the actual path in the test result, we can see the algorithm is able to derive the moving direction of the pedestrian correctly and locate the pedestrian is always at a reasonable place (in the right space and outside obstacles), and there is no obvious jumps between locations.

The performance of the algorithm is dependent on the input parameters, the indoor environment and the tracking distance. The test case 2 shows the tolerance of outliers of WiFi measured location can influence the tracking accuracy. From the test case 3 and 4, it can be seen that the accuracy of the trajectory is affected by the constraints of the environment and distance to track. The algorithm shows weakness on determining the location at the correct side of the obstacle, but this error can be corrected with the increase of the tracking distance.

5-1-3 Implementation

A tracking system is implemented to test the proposed algorithm. We developed a tracking system using java technology. The system has three layers: database where the spatial model is stored, web service where the tracking algorithm is implemented and mobile client where the location and orientation data is measured. The web service also has three layer: DAO layer, tracking service layer and control layer. The architecture of the system is recommended by the CGI group company because it has following advantages: lightweight mobile client, low requirements for data transmission, and high efficiency.

The tracking algorithm is tested offline, because the mobile client is not connected to web service in real time, the algorithm is applied after the collection of all measurements. But the system has the ability to track the pedestrian online when it is fully implemented.

5-2 Future work

The proposed tracking algorithm can provide a relatively accurate trajectory of the pedestrian under the given assumptions. But the following aspects of the algorithm and tracking system could be improved in future:

The algorithm uses a constant walking speed that is not suitable for real situations. The walking speed of the person could be updated according to traveled distance and time. In order to measure the speed, other devices such as inertial measure unit (IMU) and pedometer are needed. With accurate speed, the algorithm can adjust the search region of the current location (buffer) dynamically.

The algorithm uses the direction of the y-axis of mobile device to represent the person's orientation. However, this direction may be not the walking direction of the person if the mobile device is rotated. It is necessary to adjust the direction based on the position of the mobile device.

Most parameters of the algorithm are determined by experience or limited experiments. A research on how to optimize these parameters should be carried out in the future.

The 2D grid model is used by this system. However, this model has limitations to represent the vertical space between floors. In order to estimate the trajectory between different floors, a 3D model could be considered.

The system should be fully implemented and tested in the case of online tracking. The algorithm should be tested in a larger and more complex environment such as a large open space or a place with repetitive architectural patterns. It is significant to know if the size and complexity of the environment will influence the tracking performance.

A quantitative evaluation method to assess the performance of the tracking algorithm needs to be studied.

Bibliography

[1] J. Lloret, J. Tomas, A. Canovas, and I. Bellver, "Geowifi: A geopositioning system based on wifi networks," in *ICNS 2011, The Seventh International Conference on Networking and Services*, pp. 38–43, 2011.

[2] A. Bose and C. H. Foh, "A practical path loss model for indoor wifi positioning enhancement," in *Information, Communications & Signal Processing, 2007 6th International Conference on*, pp. 1–5, IEEE, 2007.

[3] M. Youssef, "Horus: A wlan-based indoor location determination system," *Department of Computer Science, University of Maryland*, 2004.

[4] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[5] W. Burgard, D. Fox, and D. Hennig, "Fast grid-based position tracking for mobile robots," in *KI-97: Advances in Artificial Intelligence*, pp. 289–300, Springer, 1997.

[6] M. Worboys, "Modeling indoor space," in *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness*, pp. 1–6, ACM, 2011.

[7] C. S. Jensen, H. Lu, and B. Yang, "Graph model based indoor tracking," in *Mobile Data Management: Systems, Services and Middleware, 2009. MDM'09. Tenth International Conference on*, pp. 122–131, IEEE, 2009.

[8] D. L. Lee and Q. Chen, "A model-based wifi localization method," in *Proceedings of the 2nd international conference on Scalable information systems*, p. 40, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007.

[9] B. Domínguez, Á. García, and F. Feito, "Semantic and topological representation of building indoors: an overview," *CNKI Proc., The International Society for Photogrammtery and Remote Sensing (ISPRS)*, 2011.

[10] Y. Kim, H. Shin, and H. Cha, "Smartphone-based wi-fi pedestrian-tracking system tolerating the rss variance problem," in *Pervasive Computing and Communications (PerCom), 2012 IEEE International Conference on*, pp. 11–19, IEEE, 2012.

[11] S. Zlatanova, G. Sithole, M. Nakagawa, and Q. Zhu, "Problems in indoor mapping and modelling," *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 1, no. 4, pp. 63–68, 2013.

[12] F. Donovan, "Indoor location market to reach $ 4 billion in 2018, predicts abi." http://www.fiercemobileit.com/story/indoor-location-market-reach-4-billion-2018-predicts-abi/2013-10-18, October 2013.

[13] M. Zhao, "The wifi-based positioning technologies with smart mobile device," Master's thesis, Wuhan University, 2013.

[14] X. Li, C. Claramunt, and C. Ray, "A grid graph-based model for the analysis of 2d indoor spaces," *Computers, Environment and Urban Systems*, vol. 34, no. 6, pp. 532–540, 2010.

[15] E.-P. Stoffel, B. Lorenz, and H. J. Ohlbach, "Towards a semantic spatial model for pedestrian indoor navigation," in *Advances in Conceptual Modeling–Foundations and Applications*, pp. 328–337, Springer, 2007.

[16] V. Tsetsos, C. Anagnostopoulos, P. Kikiras, and S. Hadjiefthymiades, "Semantically enriched navigation for indoor environments," *International Journal of Web and Grid Services*, vol. 2, no. 4, pp. 453–478, 2006.

[17] I. Afyouni, C. Ray, and C. Claramunt, "Spatial models for context-aware indoor navigation systems: A survey," *Journal of Spatial Information Science*, no. 4, pp. 85–123, 2014.

[18] M. Goetz and A. Zipf, "Extending openstreetmap to indoor environments: bringing volunteered geographic information to the next level," *Proceedings of the Urban and Regional Data Management: Udms Annual 2011*, pp. 47–58, 2011.

[19] B. Lorenz, H. J. Ohlbach, and E.-P. Stoffel, "A hybrid spatial model for representing indoor environments," in *Web and Wireless Geographical Information Systems*, pp. 102–112, Springer, 2006.

[20] J. O. Wallgrün, "Hierarchical voronoi-based route graph representations for planning, spatial reasoning, and communication," in *Proceedings of the 4th International Cognitive Robotics Workshop (CogRob-2004)*, pp. 64–69, Citeseer, 2004.

[21] F. Lamarche and S. Donikian, "Crowd of virtual humans: a new approach for real time navigation in complex and structured environments," in *Computer Graphics Forum*, vol. 23, pp. 509–518, Wiley Online Library, 2004.

[22] G. Franz, H. Mallot, J. Wiener, and K. Neurowissenschaft, "Graph-based models of space in architecture and cognitive science-a comparative analysis," in *Proceedings of the 17th International Conference on Systems Research, Informatics and Cybernetics*, pp. 30–38, 2005.

[23] J. Röning, J. Riekki, and T. Seppänen, "Annual report 2008 of intelligent systems group." http://www.infotech.oulu.fi/Annual/2008/isg.html, 2008.

[24] L. E. Miller, *Indoor navigation for first responders: a feasibility study*. Citeseer, 2006.

[25] D. Fox, D. Schulz, G. Borriello, J. Hightower, and L. Liao, "Bayesian filtering for location estimation," *IEEE pervasive computing*, vol. 2, no. 3, pp. 24–33, 2003.

[26] Wikipedia, "Hidden markov model." http://en.wikipedia.org/wiki/Hidden_Markov_model, 2014. [Online; accessed 21-June-2014].

[27] I. Spassov, "Algorithms for map-aided autonomous indoor pedestrian positioning and navigation," *EPFL Thesis, Lausanne*, 2007.

[28] G. Girard, S. Côté, S. Zlatanova, Y. Barette, J. St-Pierre, and P. Van Oosterom, "Indoor pedestrian navigation using foot-mounted imu and portable ultrasound range sensors," *Sensors*, vol. 11, no. 8, pp. 7606–7624, 2011.

[29] M. Khider, S. Kaiser, P. Robertson, and M. Angermann, "The effect of maps-enhanced novel movement models on pedestrian navigation performance.," in *The 12th annual European Navigation Conference (ENC 2008)*, 2012.

[30] Wikipedia, "Taxicab geometry." http://en.wikipedia.org/wiki/Manhattan_distance, 2014. [Online; accessed 10-June-2014].

[31] MathWorld, "Dot product." http://mathworld.wolfram.com/DotProduct.html, 2014. [Online; accessed 22-June-2014].

[32] Wikipedia, "Degenerate distribution." http://en.wikipedia.org/wiki/Degenerate_distribution, 2013. [Online; accessed 14-June-2014].

[33] Wikipedia, "Preferred walking speed." http://en.wikipedia.org/wiki/Preferred_walking_speed, 2014. [Online; accessed 15-June-2014].

[34] S. Theodoridis and K. Koutroumbas, *Pattern recognition*. Academic Press, 2009.

[35] MathWorld, "Rectangle function." http://mathworld.wolfram.com/RectangleFunction.html, 2014. [Online; accessed 14-June-2014].

[36] R. Ramakrishnan and J. Gehrke, *Database management systems*, vol. 3. McGraw-Hill New York, 2003.

[37] Spring, "Spring framwork." http://spring.io/projects, 2014. [Online; accessed 22-June-2014].

[38] Hibernate, "Hibernate ogm." http://hibernate.org/ogm/, 2014. [Online; accessed 22-June-2014].

[39] P. Lager, "Path finder library." http://www.lagers.org.uk/pfind/index.html, 2014. [Online; accessed 23-April-2014].

[40] Software and P. E. R. Group, "Hac - a java class library for hierarchical agglomerative clustering." http://sape.inf.usi.ch/hac.html, 2013. [Online; accessed 15-May-2014].

Glossary

LBS location-based services

GPS Global Positioning System

RSS received signal strength

RSSI received signal strength indicator

APs access points

IFC Industry Foundation Classes

KML Keyhole Markup Language

GVD generalized Vorinoi Diagram

GVG generalized Vorinoi Graph

IMU inertial measurement unit

OS operating system

ER diagram entity-relationship diagram

ORM object-relational mapping

DAO Data access object

Appendix A: Mobile client

```
1  public class Tracking extends ActionBarActivity {
2  private ArrayList<String> locationData = new ArrayList<String>();
3  private Location location;
4  private SensorManager mSensorManager;
5  private Sensor mOrientationSensor,mAccelerometerSensor;
6  private float mTargetDirection,mAccelerometer;
7  private Boolean  hasOritensionSensor,hasAccelerometerSensor,
       hasRegisterSensorTOSensorManger,hasRegisterSensorTASensorManger;
8  float x;
9  float y;
10 String result;
11 private TextView text1;
12 private TextView text2;
13 boolean start;
14 long starttime;
15 long stoptime;
16 double measuretime;
17 @Override
18 protected void onCreate(Bundle savedInstanceState) {
19   super.onCreate(savedInstanceState);
20   setContentView(R.layout.activity_tracking);
21 }
22
23 @Override
24 public boolean onCreateOptionsMenu(Menu menu) {
25   // Inflate the menu; this adds items to the action bar if it is
         present.
26   getMenuInflater().inflate(R.menu.tracking, menu);
27   return true;
28 }
29
30 @Override
31 public boolean onOptionsItemSelected(MenuItem item) {
32   // Handle action bar item clicks here. The action bar will
33   // automatically handle clicks on the Home/Up button, so long
34   // as you specify a parent activity in AndroidManifest.xml.
35   int id = item.getItemId();
```

```
36      if (id == R.id.action_settings) {
37        return true;
38      }
39      return super.onOptionsItemSelected(item);
40    }
41
42    /*
43     * start tracking
44     */
45    public void start(View view){
46      start=true;
47      text1=(TextView) findViewById(R.id.textView1);
48      text1.setText("processing");
49      localize();
50      starttime=System.nanoTime();
51
52    }
53
54    /*
55     * stop tracking and record measurements
56     */
57    public void stop(View view){
58      start=false;
59      removeSensorFromSensorManger();
60      stoptime=System.nanoTime();
61      text2=(TextView) findViewById(R.id.textView2);
62      text2.setText("save");
63      /*String Path = Environment.getExternalStorageDirectory()
64          .getPath();*/
65      String Path = "/storage/emulated/legacy/whuapp";
66      /*String Path = Environment.getDataDirectory().getPath();*/
67      try {
68        FileWriter writer = new FileWriter(Path+"/test.txt",true);
69        writer.append("\n");
70        for(String point:locationData){
71          writer.append(point+"\n");
72
73        }
74        double seconds= (double)(stoptime-starttime) / 1000000000.0;
75        writer.append("time:"+seconds);
76        writer.append("\n");
77        writer.close();
78      } catch (IOException e) {
79        // TODO Auto-generated catch block
80        e.printStackTrace();
81      }
82    }
83
84    /**
85     * initial sensor
86     *
87     * @param mSensorManager
88     *            SensorManagerâśźèšą
```

```
89     */
90    private void initSensor(SensorManager mSensorManager)
91    {
92      this.mSensorManager = mSensorManager;
93      if ((mOrientationSensor = mSensorManager
94          .getDefaultSensor(Sensor.TYPE_ORIENTATION)) != null)
95      {
96        hasOritensionSensor = true;
97      }
98      if ((mAccelerometerSensor = mSensorManager
99          .getDefaultSensor(Sensor.TYPE_ACCELEROMETER)) != null)
100     {
101       hasAccelerometerSensor = true;
102     }
103   }
104
105   private SensorEventListener mySensorEventListener = new
        SensorEventListener(){
106     public void onAccuracyChanged(Sensor sensor, int accuracy) {
107       }
108     public void onSensorChanged(SensorEvent event)
109     {
110       Sensor sensor = event.sensor;
111       if(sensor.getType()==Sensor.TYPE_ORIENTATION){
112         //get direction
113         float direction = event.values[SensorManager.DATA_X] * -1.0f;
114         // change direction to 0~360 degree
115         mTargetDirection = normalizeDegree(direction);
116         mTargetDirection = normalizeDegree(mTargetDirection * -1.0f);//
117       }else if(sensor.getType()==Sensor.TYPE_ACCELEROMETER){
118         // get accelerometer
119         mAccelerometer=getAccelerometer(event);
120       }
121
122     }
123   };
124
125   /**
126    *  register sensor to SensorManger
127    */
128   protected void registerSensorToSensorManger()
129   {
130     if (hasOritensionSensor)
131     {
132       mSensorManager.registerListener(mySensorEventListener,
            mOrientationSensor,
133           SensorManager.SENSOR_DELAY_GAME);
134       hasRegisterSensorTOSensorManger = true;
135     }
136     if (hasAccelerometerSensor)
137     {
138       mSensorManager.registerListener(mySensorEventListener,
            mAccelerometerSensor,
```

```
139             SensorManager.SENSOR_DELAY_GAME);
140         hasRegisterSensorTASensorManger = true;
141       }
142     }
143
144     /**
145      * remove sensor from SensorManger
146      */
147     protected void removeSensorFromSensorManger()
148     {
149       if (hasOritensionSensor||hasAccelerometerSensor)
150       {
151         mSensorManager.unregisterListener(mySensorEventListener);
152         hasRegisterSensorTOSensorManger = false;
153         hasRegisterSensorTASensorManger = false;
154       }
155
156     }
157
158     /**
159      * change angle to 0~360 degree
160      *
161      * @param measured degree
162      *
163      * @return normalized angle
164      */
165     private float normalizeDegree(float degree)
166     {
167       return (degree + 720) % 360;
168     }
169
170     private float getAccelerometer(SensorEvent event) {
171         float[] values = event.values;
172
173         // Movement
174         float x = values[0];
175         float y = values[1];
176         float z = values[2];
177         float accelationSquareRoot =z-SensorManager.GRAVITY_EARTH;
178         return accelationSquareRoot;
179       }
180
181     public void localize() {
182       mSensorManager = (SensorManager) getSystemService(Context.
              SENSOR_SERVICE);
183       location = Location.getInstance(mSensorManager,
184           Tracking.this, getIntent());
185       initSensor(mSensorManager);
186       if(hasOritensionSensor||hasAccelerometerSensor){
187         registerSensorToSensorManger();
188       }
189       location.startLocation();
```

```
190     location.addOnLocationChangedListener(new onLocationChangedListener()
            {
191       public void parseLocation(LocationEvent e) {
192         if (start) {
193           x = e.getX();
194           y = e.getY();
195           measuretime=(System.nanoTime()-starttime)/ 1000000000.0;
196           result=e.getResult();
197           x=x+507400;
198           y=y+6791600;
199           String measure =  x + "," + y +","+measuretime+","+
                mTargetDirection+","+mAccelerometer+","+result;
200           locationData.add(measure);
201         }
202       }
203     });
204   }
205 }
```

Appendix B: Tracking algorithm (parts)

```
1   public Grid tracking(Grid pgrid,List<Geometry> measurements,Map<String,
        String> params){
2     Double ori = Double.parseDouble(params.get("ori"));
3     Double velocity= Double.parseDouble(params.get("velocity"));
4     Double tinterval=Double.parseDouble(params.get("tinterval"));
5     Double oriweight = Double.parseDouble(params.get("oriweight"));
6     Double preori=Double.parseDouble(params.get("preori"));
7     List<Double> mpara = new ArrayList<Double>();
8     Collection<Grid> mgrids =new HashSet<Grid>();
9     Map<String,Integer> spacevalue =new HashMap<String,Integer>();
10    Map<Grid,Double> posterior = new HashMap<Grid,Double>();
11    mpara.add(velocity);
12    mpara.add(tinterval);
13    Collection<Grid> gridmodel = updateGridModel(pgrid,mpara,getGrids());
14    Map<Grid,Double> priori = prediction(pgrid,gridmodel,preori,oriweight
          );
15    if(measurequalityCheck(pgrid,measurements,params).size()>0){
16      measurements=measurequalityCheck(pgrid,measurements,params);
17      mgrids = mapPointtomodel(measurements,gridmodel);
18      spacevalue=spaceCheck(mgrids,pgrid,ori,velocity*tinterval);
19      posterior = probabilityUpdate(mgrids,spacevalue,gridmodel,priori
            ,3.0);
20      /*Grid location= pgrid;*/
21      Grid location = findMaxpos(posterior);
22      return location;
23
24    }else{
25      return estimateLocation(priori,ori,pgrid);
26
27    }
28
29  }
30  /**
31   * derive the sub model
32   */
```

```
33      public Collection<Grid> updateGridModel(Grid pgrid,List<Double> para,
            Collection<Grid> model){
34      Double ratio= 2.0;
35      Double extend = ratio*para.get(0)*para.get(1);
36      Collection<Grid> newgridmodel = new ArrayList<Grid>();
37      for(Grid grid:model){
38        if(grid.getGeometry().coveredBy(pgrid.getGeometry().buffer(extend))
              ){
39          newgridmodel.add(grid);
40
41        }
42
43      }
44      return newgridmodel;
45    }
46  /**
47     * prediction step:
48     * compute priori probability
49     */
50    public Map<Grid,Double> prediction(Grid pgrid, Collection<Grid>
          gridmodel,Double ori,Double oriweight){
51      Map<Grid,Double> priori = new HashMap<Grid,Double>();
52      Double totalpos=0.0;
53      for(Grid grid:gridmodel){
54        Double pos;
55        if(grid.getObs_id()==null){
56          GraphNode[] path= getRoute(pgrid.getId(),grid.getId());
57          Double shortestpath=getRouteDistance(path);
58          Double distance = pgrid.getGeometry().getCentroid().distance(grid
                .getGeometry().getCentroid());
59          Double pos0=getPreprob(Math.abs(shortestpath-distance));
60          if(grid.getSpace_id()==pgrid.getSpace_id()){
61            pos=pos0;
62          }else{
63            pos=pos0*0.5;
64          }
65        }else{
66          pos=0.0;
67        }
68        if(ori<=2*Math.PI){
69          if(oriCheck(pgrid.getGeometry(),grid.getGeometry(),ori,10.0,Math.
                PI/4)){
70            pos=pos*oriweight;
71          }
72        }
73
74        priori.put(grid, pos);
75      }
76      return priori;
77    }
78    /**
79     * remove outliers of WiFi measured location
80     */
```

```
81    private List<List<Integer>> removeOutliers(List<List<Integer>> clusters
          ,List<Double[]> clustercenters,Grid pgrid,Map<String,String> params)
          {
82      Double ori = Double.parseDouble(params.get("ori"));
83      Double velocity= Double.parseDouble(params.get("velocity"));
84      Double tinterval=Double.parseDouble(params.get("tinterval"));
85      Double range = 2*velocity*tinterval;
86      int msize = Integer.parseInt(params.get("msize"));
87      List<List<Integer>> newclusters = new ArrayList<List<Integer>>();
88      List<Double[]> newclustercenters= new ArrayList<Double[]>();
89      int size=clustercenters.size();
90
91      for(int i =0; i<size;i++){
92        GeometryFactory gf = new GeometryFactory();
93        Point pt =gf.createPoint(new Coordinate(clustercenters.get(i)[0],
              clustercenters.get(i)[1]));
94        if(oriCheck(pgrid.getGeometry(),pt,ori,range,Math.PI/3)){
95          newclusters.add(clusters.get(i));
96          newclustercenters.add(clustercenters.get(i));
97        }
98      }
99      clusters.clear();
100     clustercenters.clear();
101     clusters=newclusters;
102     clustercenters=newclustercenters;
103
104     if(clusters.size()>1){
105       int max=0;
106       int ind=0;
107       int sum=0;
108       for(int i=0;i<clusters.size();i++){
109         sum=sum+clusters.get(i).size();
110         if(clusters.get(i).size()>max){
111           max=clusters.get(i).size();
112           ind=i;
113         }
114       }
115       if(max>=sum/2){
116         List<Integer> maincluster = clusters.get(ind);
117         clusters.clear();
118         clusters.add(maincluster);
119       }else{
120         clusters.clear();
121       }
122     }
123     return clusters;
124   }
125   /**
126    * determine space property
127    */
128 private Map<String, Integer> spaceCheck(Collection<Grid> mgrids, Grid
      pgrid, Double ori, Double range){
129     Map<Integer, Integer> map = new HashMap<Integer, Integer>();
```

```
130     Door spacedoor= new Door();
131     for(Grid mgrid:mgrids){
132       Integer count = map.get(mgrid.getSpace_id());
133       map.put(mgrid.getSpace_id(),(count==null)?1:count+1);
134     }
135     Integer max =0;
136     for(Integer value:map.values()){
137       if(value>max){
138         max=value;
139       }
140     }
141     Integer space=-1;
142     Integer sdoor =-1;
143     Map<String, Integer>spacemap = new HashMap<String, Integer>();
144     Integer previousspace=pgrid.getSpace_id();
145     for(Integer key:map.keySet()){
146       if(map.get(key)==max){
147         if(key==previousspace){
148           space=previousspace;
149           break;
150         }else{
151           space=key;
152         }
153       }
154     }
155     if(space!=previousspace){
156       for(Door door:getDoors()){
157         int sp1=door.getSpace1();
158         int sp2=door.getSpace2();
159         if((space==sp1&&previousspace==sp2)||(space==sp2&&previousspace==
              sp1)){
160           spacedoor = door;
161           break;
162         }
163       }
164       if(spacedoor.getId()<0){
165         space = previousspace;
166       }else {
167         Set<Grid> doorgrids = spacedoor.getGrid();
168         for(Grid doorgrid:doorgrids){
169           if(oriCheck(pgrid.getGeometry(),doorgrid.getGeometry(),ori,
                range,Math.PI/4)){
170             if(pgrid.getDoor_id()==null||pgrid.getDoor_id()!=spacedoor.
                  getId()){
171               sdoor=spacedoor.getId();
172             }
173           }
174         }
175       }
176     }
177     if(sdoor<0){
178       spacemap.put("space", space);
179
```

```
180     }else{
181       spacemap.put("door", sdoor);
182     }
183     return spacemap;
184   }
185  /**
186    * update step:
187    * compute posterior probability
188    */
189   private Map<Grid,Double> probabilityUpdate(Collection<Grid> mgrids,Map<
          String,Integer> spacevalue,Collection<Grid> gridmodel,Map<Grid,
          Double> priori,double range){
190     if(spacevalue.get("door")==null){
191       for(Grid grid:gridmodel){
192         Double count=0.0;
193         for(Grid mgrid:mgrids){
194           if(grid.getSpace_id()==spacevalue.get("space")){
195             if(mgrid.getGeometry().buffer(range/3).contains(grid.
                  getGeometry())){
196               count=count+3;
197             }else if(mgrid.getGeometry().buffer(2*range/3).contains(grid.
                  getGeometry())){
198               count=count+2;
199             }else if(mgrid.getGeometry().buffer(range).contains(grid.
                  getGeometry())){
200               count=count+1;
201             }
202           }
203         }
204         Double pos=count/mgrids.size();
205         priori.put(grid,priori.get(grid)*pos);
206       }
207     }else{
208       try {
209         Door door = getProjectService().getDoorByID(spacevalue.get("door"
              ));
210         Double pos;
211         for(Grid grid:gridmodel){
212           if(hasDoor(door,grid)){
213             pos=1.0;
214           }else{
215             pos=0.0;
216           }
217           priori.put(grid,priori.get(grid)*pos);
218         }
219       } catch (ProjectServiceException e) {
220         // TODO Auto-generated catch block
221         e.printStackTrace();
222       }
223     }
224     return priori;
225   }
226
```

```
227   private Grid estimateLocation(Map<Grid,Double> priori, Double ori,Grid
          pgrid){
228     List<Grid> candidates =new ArrayList<Grid>();
229     for(Grid key :priori.keySet()){
230       if(priori.get(key)>0.5&&oriCheck(pgrid.getGeometry(),key.
              getGeometry(),ori,10.0,Math.PI/4)&&key.getSpace_id()==pgrid.
              getSpace_id()){
231         GraphNode[] path= getRoute(pgrid.getId(),key.getId());
232         Double distance=getRouteDistance(path);
233         if(distance<4){
234           candidates.add(key);
235
236         }
237       }
238     }
239     if(candidates.size()==0){
240       return pgrid;
241     }
242     else if(candidates.size()==1){
243       return candidates.get(0);
244     }else{
245       double max=0.0;
246       Grid location = pgrid;
247       for(Grid each:candidates){
248         Double[] vector1={Math.sin(ori),Math.cos(ori)};
249         Double[] vector2={each.getGeometry().getCentroid().getX()-pgrid.
                getGeometry().getCentroid().getX(),each.getGeometry().
                getCentroid().getY()-pgrid.getGeometry().getCentroid().getY()
                };
250         if(getAngleofvectors(vector1,vector2)>max){
251           max=getAngleofvectors(vector1,vector2);
252           location=each;
253         }
254       }
255       return location;
256     }
257   }
```