Machine learning based WiFi location fingerprinting

Khuong An Nguyen
Computer Science Department
Royal Holloway University of London

A thesis submitted for the degree of
Doctor of Philosophy

September 2016

Declaration

I, Khuong An Nguyen, declare that the work in this thesis is my own except where specifically indicated in the text. This thesis is the result of work carried out in the Computer Science department, at Royal Holloway University of London, between October 2012 and September 2016. No part of this dissertation has been or is currently being submitted for a degree, or diploma or any other qualification. This thesis does not exceed 50,000 words including references and footnotes.

Khuong An Nguyen
September 2016

Acknowledgements

Foremost, I would like to thank my supervisor, Dr. Zhiyuan Luo, for his kind guidance over the last four years, for giving me the opportunities to learn from other world-class researchers, for allowing me to present my work at the top conferences and in the top journals, for his patience in so many long discussions, and finally, for arranging the financial assistance. Without him, I cannot have reached this stage of my academic career.

I am grateful to many people I have had the pleasure to collaborate with during the course of my Ph.D. In particular, I would like to thank Prof. Chris Watkins for inspiring me with the topic of epidemic tracking. I would like to thank the IT support team at Royal Holloway University for helping with some technical aspects of my work. Finally, I wish to acknowledge the Computer Science department for the warm working environment.

My biggest gratitude goes to my parents for their unconditional love and support whenever I needed through-out my journey, and for always standing by me through all the ups and downs of my career. I would like to thank my mother Duoc for always listening to me, my father Chanh for providing the personal financial support. This work is dedicated to them.

Abstract

Indoor localisation provides the positioning service to the indoor users, where the GPS coverage is not available. Much research effort has been invested into ‘Location Fingerprinting’, which is considered one of the most effective indoor tracking methods to date. Fingerprint-based approaches piggyback on top of the existing indoor communication layers such as the WiFi network to provide the location-based service. However, the challenges of fingerprinting are the huge training database, the dynamic indoor environment, and the WiFi fingerprints may struggle to provide fine-grained positioning accuracy at certain indoor positions. This thesis addresses the mentioned problems using several machine learning algorithms and additional information observed from the users and the indoor environment.

While the literatures considered fingerprinting as an un-supervised learning problem to divide the training database into clusters based on the received signal strength only, this thesis realises that fingerprinting is a supervised learning problem with the Cartesian label attached to each signal reading. By exploiting this information, the thesis introduces a two-level clustering process with mixture of Gaussians clustering to improve the database’s inquiring speed and the prediction accuracy.

The WiFi signal strength is notoriously noisy. Therefore, the thesis examines the use of the magnetic field strength in the form of a continuous sequence. This sequence represents a walking trajectory, which will be demonstrated to be highly repeatable with time. The thesis introduces a new training dataset called the routine database to facilitate the user’s walking habits, in order to predict in advance the potential destination and the intended route, which is a novel feature for fingerprinting.

Most existing fingerprint-based approaches imply a strong assumption that the estimated user position is correct, to the best of their knowledge. However, there is no such guarantee, given the noisiness nature of the training database. To address this challenge, the thesis proposes a confidence measure to reflect the uncertainty of the positioning prediction.

Finally, the thesis presents a scheme to detect and estimate the signal strength of the missing WiFi Access Points. This problem may happen at any time because the Access Point is accidentally switched off, or because of the nature of the WiFi scanning process. However, it is conveniently overlooked in most fingerprinting researches.

The proposed approaches in this thesis have been validated in the real offices with challenging indoor environments. One test bed has multiple buildings and floors, and has been previously used in the EvAAL 2015 indoor positioning competition, which provides a relative baseline for the proposed techniques. In particular, the regression and classification algorithms in this thesis were ranked second and third out of the *5* contestants, under the same competition’s test domain. In addition, they performed up to *20%* more accurate than traditional Naïve Bayes and W-KNN algorithms.

Publications

This thesis is based in part on work included in the following publications.

1. Khuong Nguyen and Zhiyuan Luo. Conformal prediction for indoor localisation with fingerprinting method. *In 12th International Conference on Artificial Intelligence Applications and Innovations (AIAI)* (pp. 214-223), Springer, 2012.

2. Khuong Nguyen and Zhiyuan Luo. Evaluation of Bluetooth properties for indoor localisation. *Progress in Location-Based Services* (pp. 127-149), Springer, 2013.

3. Khuong Nguyen and Zhiyuan Luo. Enhanced conformal predictors for indoor localisation based on fingerprinting method. *In 13th International Conference on Artificial Intelligence Applications and Innovations (AIAI)* (pp. 411-420), Springer, 2013.

4. Khuong Nguyen and Zhiyuan Luo. Selective mixture of Gaussians clustering for location fingerprinting. *In 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous)* (pp. 328-337), ACM, 2014.

5. Khuong Nguyen, Zhiyuan Luo and Chris Watkins. On the feasibility of using two mobile phones and the WLAN signal to detect co-location of two users for epidemic prediction. *In 11th International Symposium on Location-Based Services* (pp. 63-78), Springer, 2015.

6. Khuong Nguyen and Zhiyuan Luo. Reliable indoor location prediction using conformal prediction. *Annals of Mathematics and Artificial Intelligence, Volume 74* (pp. 133-153), Springer, 2015.

7. Khuong Nguyen and Zhiyuan Luo. Dynamic fingerprinting route prediction with the magnetic field data. *International Journal of Wireless and Mobile Computing*, 2017.

Contents

List of Figures

List of Tables

Nomenclature

α	Conformity measure
ξ	Significance level
AOR	Angle-of-Arrival
APs	Access Points
CDF	Cumulative Distribution Function
CP	Conformal Prediction
DTW	Dynamic Time Warping
FCM	Fuzzy c-Means
KM	K-Means
MFS	Magnetic Field Strength
MFV	Magnetic Field Vector
MoG	Mixture of Gaussians
MSE	Mean Squared Error
PDR	Pedestrian Dead Reckoning
RSS	Received Signal Strength
ToF	Time-of-Flight
W-KNN	Weighted K-Nearest Neighbours

Chapter 1

Introduction to indoor positioning

"Our success used to be determined by delivering the right message, to the right audience, at the right time. Now we can do it in the right place."
– Donald Dodge, The next big thing (2013).

This chapter presents the motivations of this thesis, the research objectives, the main contributions and the structure of the thesis.

1.1 What is indoor positioning, and why is it useful?

Global Navigation Satellite Systems (GNSS) such as GPS have been successfully deployed in the past two decades, and are indispensable for outdoor navigation. However, people spend most of their times indoors, where limited or no GNSS service is available at all, because the satellites' signals are too weak to penetrate the building. In addition, GPS cannot provide the indoor users with the positioning accuracy they need for room-level tracking. The growing demand for an indoor navigation system from the industry (e.g. the supermarkets and the hospitals) to provide the location-based services for their clients has encouraged much interest in the indoor localisation research. Some noticeable events include the founding of InLocation Alliance[1] in *2012* by *22* major mobile companies (e.g. Samsung, Sony, Qualcomm) and Apple's *$20* million big-money acquisition of the start-up WiFiSlam[2] in *2013*. They all have one goal in mind: 'To bring the location services indoors!'

Furthermore, because the majority of the people's activities happen indoors, the positioning systems have much more potential to provide personalised services tailored to the user experience, rather than simply how to travel from one location to another, as normally seen in outdoor GPS localisation. The customers do not only want to pinpoint the location of a particular product, but also prefer to be notified of the latest in-store offers of other relevant nearby products. The managers are not only interested in knowing what the customers buy, but also want to find out how long they dwell on a particular shelf. All these services may only become feasible if the user's position is known. The applications of indoor positioning can be classified into two general categories, which are 'active tracking' and 'passive monitoring'. The active tracking system can be viewed as an on-demand service that guides the user to the required destination when requested, which is equivalent to the indoor version of Google Maps. The passive monitoring system, however, runs in the background to continuously monitor the user's positions, so that the system is aware of the user's whereabouts in real-time to re-act accordingly. Table 1.1 compares some specific applications of indoor positioning for the three popular areas of the society, that are healthcare (e.g. hospitals), retailer (e.g. shopping malls) and industry (e.g. warehouses, museums). The tracking devices can be the users' mobile phone, or a tag attached to the push trolleys.

[1]http://inlocationalliance.org - last accessed in Sep/2016.
[2]https://angel.co/wifislam - last accessed in Sep/2016.

Table 1.1 The applications of indoor positioning.

	Active tracking	Passive monitoring
Healthcare	**For patients** • Route-finding to the doctor's room. **For doctors** • Finding the equipment's position. • Quickly find the patient in an emergency.	**For patients** • Automatic checking-in/-out upon arriving or leaving. • Automatic wheel-chair navigation to a destination. **For doctors** • Informed when the patient arrives. • 24/7 monitoring of patients with mental illness.
Retailer	**For customers** • Product-finder to navigate to a particular shelf. **For managers** • Identifying real-time staff location (No need for public announcements).	**For customers** • Showing the latest deals when the items of interest are nearby. • Buying assistant for personalised shopping experience. **For managers** • Creating geo-fences to automatically send out e-coupons when the user is near a particular shelf. • Identifying congestions and footfall. • Finding areas of high or low density to improve store's layout.
Industry	**For clients** • Seat-finding in the theatre. • Friend-finder app for the cinema. • Luggage-tracking at the airport. **For administrators** • Item-finder to search for products in the warehouse.	**For clients** • Automatic computer guided tours of museums. **For administrators** • Informed when a luxury item leaves the secured area. • Automatically adjusting conveyor speed when transporting heavy items in the warehouse. • Automatic door opening/closing when the staff are nearby.

1.2 Indoor positioning systems

The existing indoor positioning systems employ a variety of technologies to support the process of estimating the user position (see Figure 1.1). Some of these technologies are ubiquitous (e.g. WiFi, FM, inertial sensors on the smart phone), whereas the others are manually deployed (e.g. Ultrasonic, infrared).

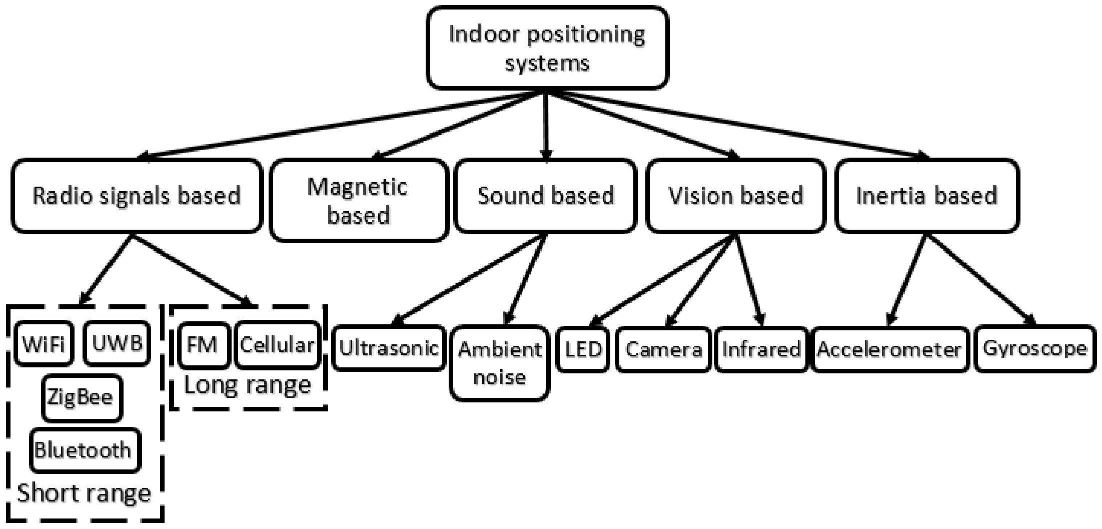


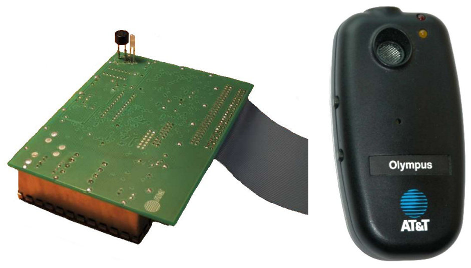
Fig. 1.1 The taxonomy of the indoor positioning systems.

Overall, based on how the systems interact with the indoor environment, they can be divided into two broad categories, which are infrastructure-based systems and infrastructure-free systems.

With the infrastructure-based approach, the system relies on a particular piece of hardware that needs to be installed into the building. These hardware are often specifically designed for indoor positioning, and are costly to install and to maintain. Some of the most notable systems include the Active Bat which is still the industry benchmark to date [70, 88]. The system comprises of a network of ultrasonic beacons mounted at fixed known positions on the ceiling to capture the pulses emitted at regular intervals from the tags (see Figure 1.2). These tags are worn by the users, so that the beacons can measure the time-of-flight the pulse taken to travel from the tag to the beacons. The system achieves up to *3* centimetre accuracy, *95%* of the time. However, only *9* institutions in the world can afford it[3]. The most recent technology in this category is the LED-based system which has been adopted by Philips

[3]In the UK, they are University of Cambridge, University of Kent, Imperial College London, Lancaster University. From overseas, they are University of Twente (Netherlands), Xerox PARC, DEC research laboratories, Bellcore and MIT (all from the US).

for the French supermarkets in 2015. For this system, normal light bulbs are replaced by special LEDs which are designed to blink rapidly at imperceptible human-eyes level. The smart phone's front camera is used to capture and to decode these optical luminaries [49]. Overall, the infrastructure-based systems often provide excellent positioning accuracy, at the expense of additional hardware, which is also challenging to set up without conflicting with other existing infrastructure already operating in the building.



(a) Bat beacon. (b) Bat tag.

Fig. 1.2 The components of the Active Bat ultrasonic positioning system [88].

In contrast, the infrastructure-free systems are self-contained and require no additional changes to the indoor environment. These systems piggyback on top of the structures that already exist in the building (e.g. the WiFi network) to provide the positioning service. The earliest and most basic form of this category was the proximity-based tracking system [11]. When a user is within the coverage distance of the wireless transmitter, his location is determined as the known location of the transmitter itself. However, the positioning accuracy was not great because the broadcasting range of the transmitter was often large. For the past decade, the most discussed approach in this category is WiFi fingerprinting, which generates a radio map of the building to serve as a training database. This data are later used to estimate the position of the user in real-time [5, 81, 86, 89, 98]. The majority of infrastructure-free systems employed ubiquitous devices such as the mobile phones to obtain inputs from the environment. Others introduced their own sensors. However, it is worth noting that they are mostly affordable. Due to the increasing popularity of the smart phones, Pedestrian Dead-Reckoning (PDR) based systems have been the new trend in recent years. These PDR systems use the inertial sensors (i.e. accelerometer, compass, gyroscope) in the phone to provide relative positioning feedbacks [16, 61, 71, 87]. The challenges for these systems are that the sensors in current smart phones are particularly noisy, because they are included for basic app support, rather than for the robust positioning purpose. Overall, the advantage of the infrastructure-free approach is the ease of deployment, at the expense of a lower positioning accuracy than the infrastructure-based approach.

Table 1.2 Summary and comparisons of the latest infrastructure-based systems.

Authors	Citation	Sensors used	Accuracy	Summary of technology
Want et al. (Active Bat)	[88]	Ultrasonic	0.03 m	Measuring the time-of-flight the ultrasonic pulse taken to travel from the user tag to the beacons on the ceiling.
Kuo et al. (Luxapose)	[49]	LED	0.07 m	Modified LEDs blink rapidly at different rates. These coded optical pulses are captured by the phone camera to decode the user location and orientation.
Aitenbichler et al. (IRIS-LPS)	[1]	Infrared, Stereo-camera	0.16 m	A stationary camera mounted on the wall measures the angle of arrival of the light rays emitted from the infrared tags worn by the users.
Hammer et al.	[31]	Loudspeaker, microphone	0.25 m	Microphone worn by the user records the 40 kHz ultrasound constantly played by the loudspeakers at fixed positions. The time-of-flight of these audible signals is calculated.
Markham et al.	[60]	Magneto-inductive transceivers	0.45 m	Magnetic transmitters are set up in the environment to provide low frequency magnetic field.
Campbell et al. (DecaWave)	[12]	Ultra-wideband transceiver	0.6 m	Measuring the time-of-flight between the receivers using minimum least squares trilateration.
Klipp et al.	[47]	Bluetooth, Accelerometer, Gyroscope	0.78 m	Inertial foot sensor to detect step-length. Bluetooth beacons are set up at the doors to correct the drifting errors.
Fang et al.	[26]	ZigBee	1.5 m	Low-power ZigBee sensors are installed at fixed known positions, whose signal are used to infer the user's position.
Ni et al. (LANDMARC)	[65]	RFID	1.6 m	RFID readers attached on the wall constantly look for the RFID tags worn by the users.
Mirshekari et al.	[12]	Geophone, Amplifier	8 m	Measuring ground vibrations induced by the human footsteps with geophone.

Tables 1.2 and 1.3 overview the design, the positioning accuracy and the underlying technologies of some notable infrastructure-based and infrastructure-free systems in the literature. Only the most prominent systems with the highest positioning accuracy for each technology were selected to review. It is important to remind that the reported accuracy was obtained by each author's own testing environment. In general, sub-metre positioning accuracy can be expected for the infrastructure-based systems, whereas, *2* to *5* metre accuracy was popular with the infrastructure-free ones.

Table 1.3 Summary and comparisons of the latest infrastructure-free systems.

Authors	**Citation**	**Sensors used**	**Accuracy**	**Summary of technology**
Berkovich (SPIRIT)	[7]	Smart Phone, WiFi	1.5 m	PDR, WiFi fingerprinting, geomagnetic fingerprinting and map matching are fused together using Particle Filter to estimate the user position.
Azizyan et al.	[4]	Microphone, Camera, WiFi	4 m	The light, sound, colour and WiFi signal from the ambient environment are recorded to generate a fingerprint database.
Varshavsky et al.	[84]	GSM	4 m	GSM signals from nearby cellular towers are captured to general a fingerprinting database.
Chung et al.	[15]	Magnetometer	4.7 m	The magnetic field strength in the building is captured by the mobile phone to create a fingerprinting dataset.
Faragher et al.	[28]	Bluetooth Low Energy[4]	4.8 m	RSS from iBeacons are used to construct a BLE fingerprinting database.
Zhang et al. (Navi-Glass)	[99]	Smart Glass	5.7 m	Relative trajectories of the users are obtained via the inertial sensors of the glass. The glass' camera provides visual sensing to correct the drifting error.
Yoon et al.	[97]	FM receiver	6 m	RSS from commercial FM radio stations are used to construct a FM fingerprinting database.

[4]This system may be classified as infrastructure-based. However, the authors argued that iBeacons are becoming as ubiquitous as WiFi, and there is no need to deploy them manually in the building.

1.3 Research questions

This thesis presents the use of several machine learning techniques and additional information to address the challenges of WiFi fingerprinting, to be discussed in detail in the next chapters. In doing so, it aims to answer the following research questions.

(i) **How to address the volatility and the noisiness of the indoor environment?**

Firstly, it is common to find several different indoor positions with a similar WiFi signal observation, because of the signal multi-path issue. Chapter 3 addresses this challenge by pre-processing the training database to divide the training examples into different groups. Secondly, the real-time signal sample may not fully match the training example previously observed in the same position, because the environment has slightly changed. Chapter 4 will introduce a novel parameter to reflect the uncertainty of the positioning prediction. Lastly, the WiFi signals may also be distorted by other environmental factors and the WiFi scanning process, which result in an incomplete signal observation. Chapter 6 will rectify this issue by identifying and estimating the missing signals.

(ii) **Is the WiFi received signal strength alone sufficient to achieve fine-grained sub-metre positioning accuracy for fingerprinting?**

Most existing fingerprint-based systems rely totally on the WiFi signal. However, trying to push the boundary of fingerprinting into sub-metre with only the WiFi signal strength is challenging. To aid the WiFi readings, using other sources of wireless information is a compelling option, yet, not all information are robust or ubiquitous enough to maintain the infrastructure-free concept of fingerprinting. Chapter 5 will present some extra information that can be easily found indoors and are specifically tailored to the users.

(iii) **Are the public outdoor WiFi Access Points suitable for outdoor mobile phone co-localisation?**

Fingerprinting is a well-known approach for indoor positioning. However, it is challenging to apply the technique for outdoor localisation, because the environment is much bigger and more dynamic. Through-out a series of experiments with the outdoor WiFi signals collected in several measurement campaigns in the real world, Chapter 7 assesses the feasibility to detect the outdoor co-localisation of the people, a sub-category of outdoor localisation, using the mobile phones and the public WiFi Access Points.

1.4 Contributions to this field of research

Firstly, most authors working on fingerprinting have mainly been concerned with the use of the WiFi signal strength only. Secondly, many fingerprint-based researches in the literature either made no assumption about the quality of the positioning estimation, or firmly believed that the prediction is the correct one. Thirdly, the positioning labels were mostly ignored in clustering-based fingerprinting. Fourthly, most fingerprinting researchers overlooked the missing WiFi APs problem during a real-time scan. Finally, fingerprinting has often been limited to identify a single position of the user.

The work I present in this thesis consists of the use of additional information beside the WiFi signals to support the positioning estimation. In particular, the thesis identifies the following contributions.

(i) I use the magnetic field to predict not just a single position, but also the whole walking trajectory and the potential destination of the user in real-time.

(ii) I introduce a confidence measure to reflect the uncertainty of the positioning predictions, and to deliver a prediction set.

(iii) I propose a novel two-level soft clustering process that involves both the WiFi RSS and the Cartesian label to reduce the searching time and tackle the signal borderline challenge at the same time.

(iv) I address the missing APs problem in the real-time WiFi scan by proposing a scheme to detect and estimate the value of the missing APs using only the WiFi RSS from the previous position.

(v) I present the results of a series of experiments with the data collected through-out several measurement campaigns in the real world to assess the feasibility of using the public outdoor WiFi APs and the mobile phones to co-locate the people's position.

1.5 Limitations of scope

This thesis focuses on evaluating the research questions posed above to address the challenges circling around the application of fingerprinting in real-world environments. However, it is not the intention of this thesis to provide an in-depth discussion of every aspect of fingerprinting. In particular, the following limitations apply.

(i) The privacy of the users' position is important for any tracking system. However, the thesis does not discuss such feature. No data encryption is performed at any time on the mobile client or the fingerprinting server.

(ii) The thesis is concerned with the problem of tracking people. It does not consider object tracking, although the algorithms described in Chapter 4 can be applied to track the position of any object with an equipped WiFi-enabled device. Other algorithms presented in Chapters 5, 6 and 7 make use of specific knowledge such as the user's routine, and therefore, are unsuitable for tracking arbitrary objects.

(iii) The WiFi signal strength is used as the main measurement in this thesis, due to its ubiquity, and the ease of measuring. Other wireless signals (e.g. FM, Cellular, Bluetooth) may also be used. Most algorithms presented in this thesis may be applied to those signals, although the thesis does not experiment all of them in detail.

(iv) The positioning result is provided in the form of the numeric Cartesian co-ordinate in this thesis, which is understood by the system to evaluate the algorithms' performance. To present this result to the normal users, the system needs to translate the co-ordinate into geo-coded information (e.g. 'The user is currently in Room 121, and is near the front door'). This process requires an extra layer running on top of the positioning system, which is beyond the scope of this thesis.

(v) The algorithms in this thesis were designed with 2-dimensional positioning data in mind, although they may also be applicable for 3-dimensional data. The terrains are assumed to be flat. The escalator, stair case and other non-flat surfaces were not tested.

(vi) Although all the proposed ideas in each chapter contribute towards a fingerprint-based indoor positioning system, it is worth noting that the thesis does not aim to deliver a complete product that is suitable to sell immediately to the public. There are still many challenges for fingerprinting in particular and for the indoor positioning field in general to be addressed.

1.6 Thesis outline

The structure of the remainder of this thesis is outlined below. Each chapter aims to address parts of the research questions posed above. The logical progression of work is described graphically in Figure 1.3.

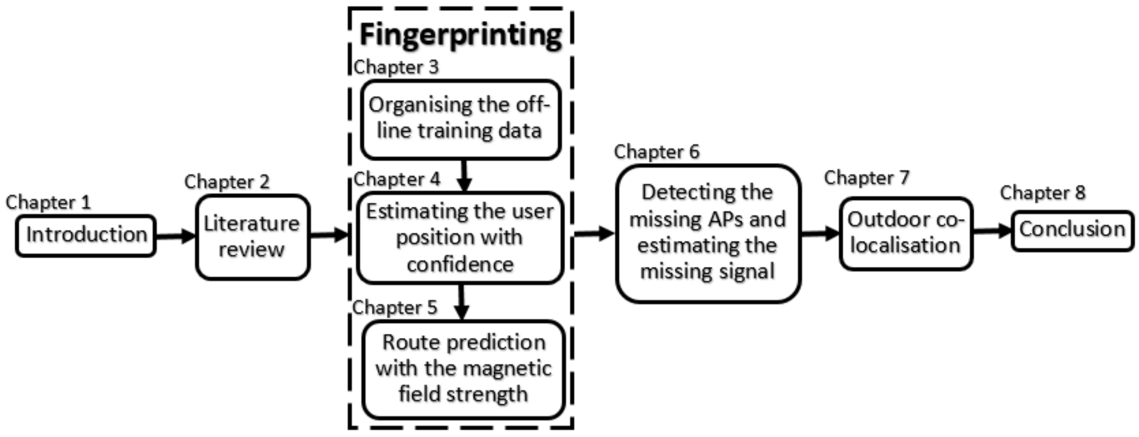

Fingerprinting
Chapter 1
Introduction
Chapter 2
Literature review
Chapter 3
Organising the off-line training data
Chapter 4
Estimating the user position with confidence
Chapter 5
Route prediction with the magnetic field strength
Chapter 6
Detecting the missing APs and estimating the missing signal
Chapter 7
Outdoor co-localisation
Chapter 8
Conclusion

Fig. 1.3 Thesis progression.

Chapter 2 presents the concept of location fingerprinting, for which the remainder of the thesis will build on. Some popular machine learning techniques which have been used for fingerprinting along with their performance accuracies in the literature are discussed. The chapter also presents the datasets to be used in this thesis.

Chapter 3 describes a clustering algorithm to improve the inquiring speed of the fingerprinting training database.

Chapter 4 describes the concept of Conformal Prediction, and presents two underlying classification and regression algorithms to provide a confidence level alongside the prediction of the user position.

Chapter 5 demonstrates a novel feature for fingerprinting to allow the system to predict in advance the intended route and the potential destination for a regular user. It explains the use of the magnetic field strength in the form of the walking trajectories, and compares it to the WiFi signal strength.

Chapter 6 presents an approach to detect the missing Access Points during a WiFi scan, and outlines a scheme to compensate for this unwanted problem by estimating the missing signals.

Chapter 7 presents the results of a series of experiments to assess the feasibility of using the public outdoor WiFi Access Points and the mobile phones to co-locate the people's position. It describes two algorithms to calculate the matching rate of two WiFi signal vectors, and compares the results with the ground-truth GPS distance.

Chapter 8 summaries the results obtained from the work carried out in this thesis, and outlines possible avenues for further research.

Chapter 2

WiFi location fingerprinting

"I do not think that the wireless waves I have discovered will have any practical application."

– Heinrich Hertz.

WiFi fingerprinting is a decade mature technology, which has been the subject of much research interest from the academia, and has also attracted developments from the big companies such as Google and Apple. Given such promising attentions, what is it that prevents the widespread adoption of fingerprinting for every building in the real world?

This chapter outlines the state-of-the-art and the challenges of WiFi fingerprinting, to be addressed in the later chapters of this thesis. It presents the machine learning approaches to fingerprinting, along with the description of the most popular learning algorithms. Then, it compares the performance of these algorithms implemented in the most noticeable fingerprint-based systems in the literature. It reviews the performance of fingerprinting amongst other well-known techniques, including those competing in the recent Microsoft 2014 indoor positioning competition. Finally, the chapter concludes by introducing the test beds to be used in this thesis.

2.1 Introduction to fingerprinting

Compared to the outdoor space, the indoor environment is more challenging for most wireless signal based technologies to work reliably. While the standard satellite signals such as GPS struggle to penetrate the building structure, other indoor wireless technologies such as WiFi or Bluetooth could not rely on their standard properties such as the time-of-flight (ToF), the angle-of-arrival (AoR) or the received signal strength (RSS) measure to calculate the distance between two positions, because of the complex layout of the building. As the wireless signals travel in the air, they reflect from the metal objects, diffract around sharp corners, scatter off the walls, floors and ceilings, which result in multiple copies of the original signal travelling in different directions. When two in-phase waves of the signal meet, constructive interference forms a new stronger wave of signal. In contrast, two out-phase waves will cancel each other out, resulting in a weaker version. The receiving signal at the end user is a combination of these distorted products. This phenomenon is known as the signal multi-path problem. Furthermore, the building is often crowded with many users who move around to create a harsh and dynamic environment.

Fingerprinting, however, uses this challenging environment as its core function. The indoor positions are manually calibrated to capture the full dynamic of the signal characteristic at each position. Thus, the more diverse and tricky the signal propagation is, the more unique the ‘location fingerprint’ is. Those training fingerprints will be matched against the user’s real-time fingerprint to estimate his position. More details about the fingerprinting processes will be discussed in the next section.

Fingerprinting was originally proposed with the WiFi RSS recorded by a laptop from the nearby WiFi Access Points (APs) [5]. Since then, other wireless signals have been tested with fingerprinting (e.g. Bluetooth, FM, Cellular). This thesis decides to use the WiFi RSS and a smart phone as the main components to perform fingerprinting for two reasons. Firstly, the WiFi AP is becoming a norm, and it is common to have many WiFi APs indoors. Furthermore, they are starting to move beyond the buildings to provide a seamless transitional coverage from the indoor space to the outdoor space. Secondly, most people carry a smart phone with them wherever they are. These devices have the computing power and the storage of a mini computer, as well as including a WiFi receiver. The properties of the WiFi signal for fingerprinting are well-researched, and need not repeating in this thesis [41, 42, 48, 56]. Instead, a more novel use of the WiFi RSS will later be discussed.

2.2 The two phases of fingerprinting

This section discusses the inner processes of fingerprinting and the difficulties for each step (see Figure 2.1).

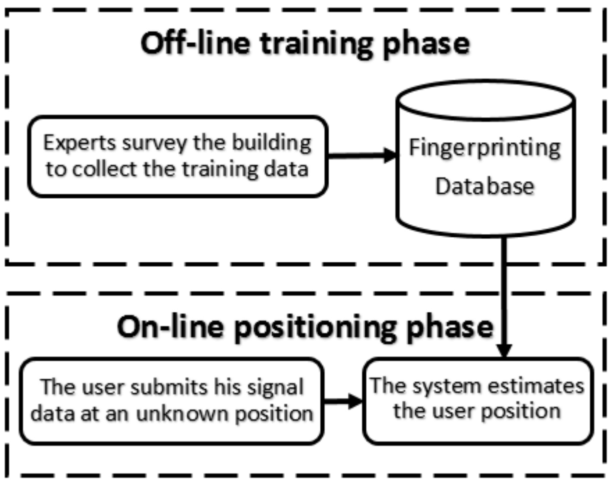

Off-line training phase
Experts survey the building to collect the training data
Fingerprinting Database
On-line positioning phase
The user submits his signal data at an unknown position
The system estimates the user position

Fig. 2.1 The two phases of fingerprinting.

2.2.1 The off-line phase of fingerprinting

This phase is also known as the training or planning phase. The purpose of this phase is to generate a training database (i.e. the fingerprinting database) to reflect how the signal propagates inside the building. For WiFi fingerprinting, it is normally done by an expert holding a WiFi-enabled device (e.g. smart phone, laptop) and walking around the building to record the WiFi RSS at different training positions. Some key issues for the experts to take into considerations are:

- **How granular the tracking space is?** The higher the granularity is, the more training positions the expert needs to cover. The tracking zone is normally divided into a metre-by-metre grid for the ease of planning.
- **How often the measures are taken at each position?** The WiFi signal is noisy, thus, it is recommended to capture the full histogram distribution of the WiFi RSS by measuring them repeatedly at each training position.
- **How to label the signal data?** Most fingerprint-based systems used their own co-ordinate metric (e.g. the Earth's latitude and longitude) to label the training WiFi RSS. Others used a more human-readable presentation (e.g. room number).

The first two plans directly decide the type of the fingerprinting algorithm to be used in the positioning phase. In most cases, they also affect the performance accuracy of the system. For instance, it may not be reasonable to expect a low-grained fingerprinting dataset with too few signal measures to perform well with any algorithm.

The challenges at this phase are the sheer amount of the building space to be meticulously surveyed, the time it takes to perform such process, and the lack of physical reference for the training positions. Some works have attempted to alleviate the first two challenges with a robot [33, 40, 64]. However, the robot does have its own problem of knowing where it exactly is in the building. Others relied on crowd-sourcing to train the system automatically [10, 14]. However, the lack of ground-truth references is a major challenge to label the crowd-sourced data. Other researchers set up landmarks in the building, where the users can contribute manually [52, 53]. Others used an independent tracking system (e.g. Active Bat) to provide the location references for fingerprinting [88]. Despite the above challenges, this training phase normally needs to be performed once at the beginning for every building.

2.2.2 The on-line phase of fingerprinting

This phase is known as the positioning phase or the estimation phase. It will be performed whenever the user wishes to discover his whereabouts. For WiFi fingerprinting, the user needs to carry a WiFi-enabled device (e.g. a smart phone) to measure the WiFi signal at his current unknown position. Given this real-time WiFi reading, the system looks up the training database generated in the previous phase to estimate the user position based on the surveyed co-ordinate labels.

There are two main challenges at this phase. Firstly, the new WiFi signal sample from the user may not fully represent his current unknown position, because the environment is distorted at the time of measuring, or because of the nature of the WiFi scanning process. This challenge will be addressed later in Chapter 7. Secondly, the system must decide in advance the type of algorithm to estimate the user position. If the training database includes multiple signal readings per location, a probabilistic approach is preferred. In contrast, if only a single signal reading is available per training location, a deterministic approach is the only option. The algorithms for each category will be discussed later in this chapter.

2.3 State-of-the-art and challenges

Most indoor positioning systems were judged solely on their affordability and accuracy. However, a highly adoptable system should also address the following criteria, which will be discussed from WiFi fingerprinting's viewpoint.

The availability is fingerprinting's main strength, thanks to the ubiquitous indoor WiFi network. Other long range outdoor signals such as the FM radio and the Cellular network may be used in conjunction with WiFi to boost the indoor coverage. Theoretically, fingerprint-based systems are always available anywhere within the coverage distance of the indoor WiFi signals.

The installation is simple from the users' viewpoint. They only need to install an app on their mobile devices, or a piece of software on their laptops, tablets, and allow it to sniff the WiFi signals to enable the tracking capability. However, the set-up process is more challenging for the administrators. A new building will require manual calibrations to generate the training database at the beginning.

The reliability of the fingerprint-based systems is normally high, because they piggyback on top of the existing WiFi network in the building, which is highly robust to provide the constant communication service to the users.

The cost to deploy a fingerprint-based system is minimal, because there is no need to install additional infrastructure. The WiFi network is already widely available in many buildings. In addition, most people possess a smart phone, which has the computing power and the storage to process the fingerprinting database. In case the training database is too large, it may be stored on a central server.

The scalability is one of the weaknesses of fingerprinting, if the central server is involved in the fingerprint-based system. The server may struggle to process a large number of inquiries from many users simultaneously, especially when the fingerprinting database is huge.

The maintainability is challenging for fingerprinting. The training database will become outdated over time because the old furniture is re-arranged or the new one is added into the environment; and to re-survey the whole tracking zone requires much labour work.

The complexity and speed of fingerprinting mostly depend only on the size of the training database. In most cases, every training example needs to be considered once to estimate the user position, which may be a burden for large buildings. However, it can be addressed via techniques such as clustering, which will be discussed later in this thesis.

The accuracy (or positioning error) of WiFi fingerprint-based systems are widely expected to be around room-level on average for most indoor environments, which is moderate, compared to other sub-metre infrastructure-based systems which makes use of dedicated hardware. Overall, the past literature suggested that the amount of infrastructure needed to deploy a positioning system and its accuracy seem to be correlated (see Figure 2.2). It is clear that the lower left corner of the figure is still missing where none of the current indoor positioning systems can yet fill in. Fingerprinting is the most balanced approach, and this thesis will attempt to bring it further left on the x-axis.

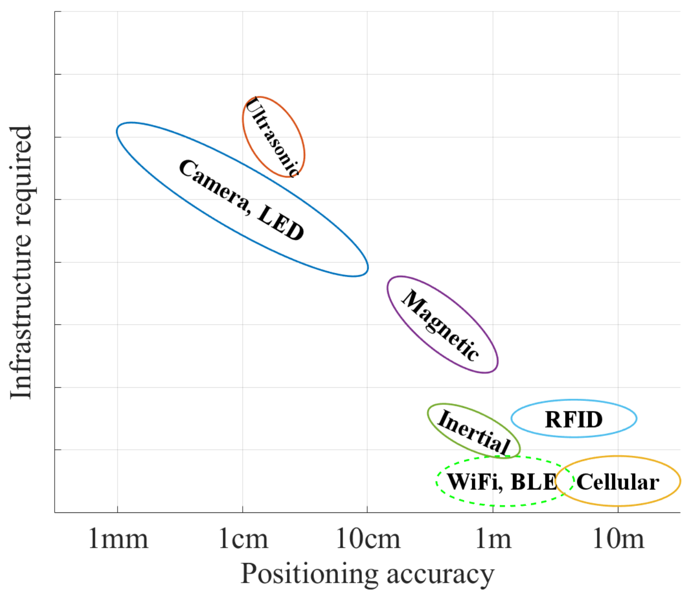


Fig. 2.2 A rough correlation of the amount of required infrastructure for the indoor positioning technologies and their accuracies. The x-axis is presented in log scale.

The precision of the fingerprint-based systems depends on the quality of the wireless signal. With WiFi fingerprinting, the receiving WiFi RSS from the mobile device has a high degree of variation, because of the indoor wireless signal's multipath effect. Furthermore, the density of the indoor users and their movements have an impact on the WiFi signal too. This is a common challenge for the application of radio signals for fingerprinting, since they were not originally designed with the positioning purpose in mind.

Other aspects such as security, risk were not discussed since they are beyond the scope of this thesis. The challenges of fingerprinting including the scalability, the accuracy and the precision will be addressed in this thesis. The approaches to be discussed in the later chapters are software-based, which rely solely on the wireless signals' information. The author believes this direction is important to maintain the core infrastructure-free concept of fingerprinting.

2.4 Machine learning approaches to fingerprinting

There are three classes of machine learning in general based on the design of the training database, which are ‘Supervised learning’, ’Unsupervised learning’ and ‘Reinforcement learning’. While the supervised learning approaches require all training examples to have both the object and the label associated with it, un-supervised learning ones deal with non-labelled training examples. Reinforcement learning, on the other hand, lies between supervised learning and un-supervised learning, since it does not have a label for all training examples, but only sparse and time-delayed feedbacks, which are called the rewards.

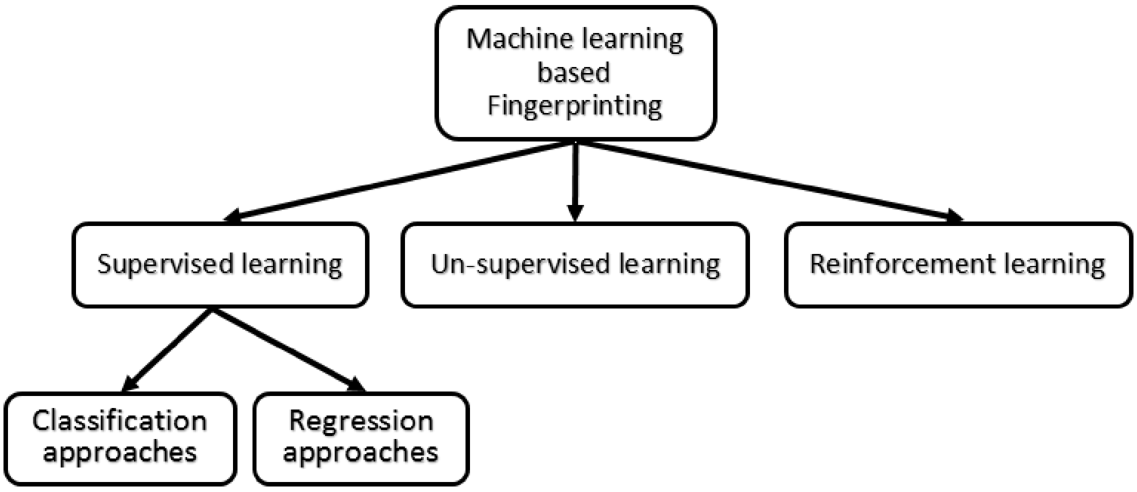


Fig. 2.3 Machine learning approaches to fingerprinting.

For fingerprinting, the training database has both the WiFi RSS (i.e. the object) and the associated Cartesian co-ordinate (i.e. the label). Therefore, it clearly favours a supervised learning approach. However, it is possible to ignore the Cartesian label, and applies un-supervised learning approaches (e.g. clustering) to divide the training examples into groups of similar WiFi RSS, which will be used to speed up the inquiring process, to be discussed later on in this thesis. Reinforcement learning approaches, however, are not truly suitable for fingerprinting, because there is often no such rewards telling the positioning system of how well it is performing at any stage. The remaining of this thesis will mainly dedicate on supervised learning approaches for fingerprinting.

For supervised learning, there are two main approaches based on the type of the training labels. They are ‘Classification approach’ and ‘Regression approach’ (see Figure 2.3). With classification, the training labels are seen as discrete finite variables, whereas, they are considered as infinite random variables with regression. At a quick glance, the Cartesian

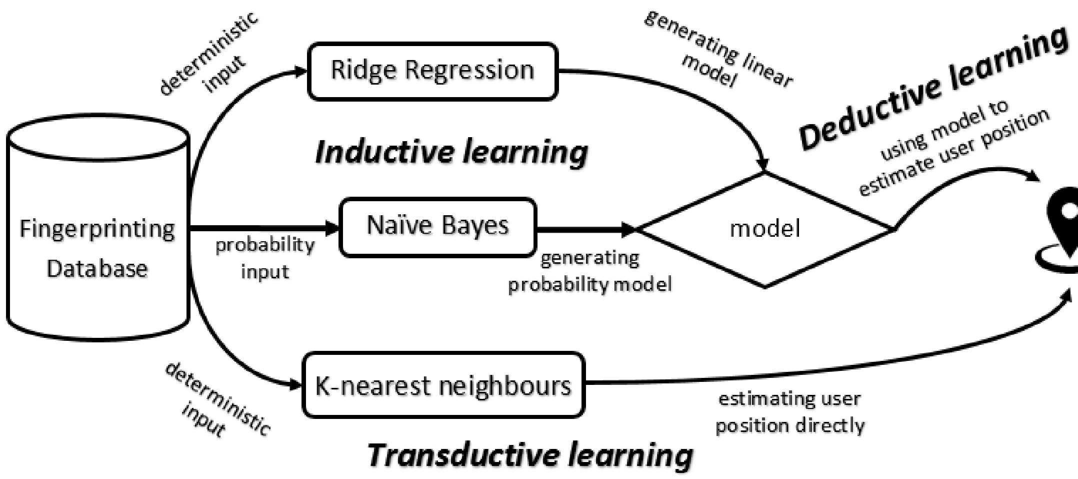


Fig. 2.4 Fingerprinting supervised learning models to be covered in this thesis.

labels of the WiFi RSS represent discrete finite indoor positions, thus, the classification approaches are overwhelming in favour to be the candidate for fingerprinting. However, at a closer look, these labels are real numbers. Thus, when there are too many training positions, the classification approaches may execute too slowly, and a regression model will be more suitable. In summary, both classification and regression approaches can be used for fingerprinting. A more preferred means to categorise supervised learning approaches for fingerprinting is to look at how they estimate the user position. Most approaches first use the training database to create a model, then use that model to predict the outcome (the label) for a new sample. The algorithms to be discussed in this chapter are Naïve Bayes which uses a probability model, and ridge regression which uses a linear model. However, there are other approaches that skip the model part and go directly from the training examples to predicting the outcome, such as K-nearest neighbours (see Figure 2.4). These algorithms and their real world performances will be discussed in this chapter.

2.4.1 Modelling the WiFi fingerprinting problem

Without loss of generality, the location fingerprinting problem is formally modelled as follows. The database indexes M training examples $T_i = (\overrightarrow{RSS_i}, \overrightarrow{L_i})$ $(1 \leq i \leq M)$. Each example T_i represents a training position, where $\overrightarrow{RSS_i}$ is the WiFi RSS vector observed at that position, and $\overrightarrow{L_i} = (d_x^i, d_y^i)$ is the 2-dimensional Cartesian label of the position. The WiFi RSS vector contains all the individual RSS from N nearby APs, $\overrightarrow{RSS_i} = (AP_1^i, \ldots, AP_N^i)$, where AP_j is the RSS received from the AP j^{th} $(1 \leq j \leq N)$. It is possible that there are duplicated

$\overrightarrow{L_i}$ in the training set, because the WiFi RSS are captured several times at the same position. Table 2.1 illustrates such training database. There may be other features attached to each training example such as the user orientation, the time of measure, the calibration device.

Table 2.1 An example of the fingerprinting database.

Position label	WiFi RSS measurement
...	......
(51, 136)	(-57, -41, -62, -59, -86, -91)
(51, 136)	(-57, -42, -60, -58, -87, -93)
(51, 146)	(-59, -36, -65, -58, -82, -95)
(51, 156)	(-56, -38, -69, -58, -84, -93)
(51, 156)	(-54, -35, -69, -57, -84, -95)
(51, 166)	(-63, -32, -73, -57, -87, -99)
(51, 176)	(-67, -33, -68, -60, -91, -99)
...	......

The task is, given a WiFi RSS vector $\overrightarrow{RSS_u} = (AP_1^u, \ldots, AP_N^u)$ at an unknown position u lying somewhere in the tracking zone, the system estimates the Cartesian label $\overrightarrow{L_u} = (d_x^u, d_y^u)$ for this position.

2.4.2 Classification fingerprinting with weighted K-nearest neighbours

Given a set of M labelled training examples $T_i = (\overrightarrow{RSS_i}, \overrightarrow{L_i})$ $(1 \leq i \leq M)$, and an un-labelled new sample $S_u = (\overrightarrow{RSS_u}, \overrightarrow{L_u})$, the most convenient method to estimate the position $\overrightarrow{L_u}$ is to find a training example T_i that is closest to S_u in terms of their RSS vectors' difference. A possible means to calculate the difference of two RSS vectors is by using the Euclidean distance as follows.

$$Dist(\overrightarrow{RSS_i}, \overrightarrow{RSS_u}) = Dist((AP_1^i, \ldots, AP_N^i), (AP_1^u, \ldots, AP_N^u)) = \sqrt{\sum_{p=1}^{N} (AP_p^i - AP_p^u)^2} \ . \quad (2.1)$$

Since there may be more than one training example with a similar Euclidean distance to S_u, or the nearest training example may not necessary be the correct one, it is recommended to take a set of K training examples with the smallest Euclidean distances. A weight will be introduced to penalise the training examples that are further away. This weight may be calculated as the inverse Euclidean distance of the WiFi RSS between each K training example and the new sample [35]. The estimated weighted location L_u is calculated below

(ε is a small constant to avoid division by zero).

$$L_u = \frac{\sum_{i=1}^{K} \frac{1}{Dist(\overrightarrow{RSS_i}, \overrightarrow{RSS_u}) + \varepsilon} \overrightarrow{L_i}}{\sum_{i=1}^{K} \frac{1}{Dist(\overrightarrow{RSS_i}, \overrightarrow{RSS_u}) + \varepsilon}} \ . \tag{2.2}$$

This approach is known as Weighted K-nearest neighbours (W-KNN) [37, 77]. It is amongst the simplest learning algorithms. The disadvantage of this approach is the need to go through all training examples to calculate the Euclidean distance every time a new sample is presented. It also requires a pre-determined K value which may impact the prediction result if not chosen properly.

2.4.3 Classification fingerprinting with Naïve Bayes

The probabilistic approach to fingerprinting addresses one important challenge of the indoor WiFi signal. That is, the WiFi RSS observed from the same AP varies, even in the same position. It is possible that the training RSS and the real-time RSS are not similar even though they come from the same position. Therefore, with this approach, the WiFi RSS are captured repeatedly at each training position to record the full signal distribution of each AP.

Given the real-time RSS vector $\overrightarrow{RSS_u}$ at an unknown position, the algorithm calculates the posterior probability $P(T_i|\overrightarrow{RSS_u})$ of this unknown RSS vector being observed at the training position T_i $(1 \leq i \leq M)$. The training position with $\max_i(P(T_i|\overrightarrow{RSS_u}))$ will be chosen as the estimated position. Following Bayes' rule, this probability is calculated as [98]:

$$P(T_i|\overrightarrow{RSS_u}) = \frac{P(\overrightarrow{RSS_u}|T_i)\ P(T_i)}{P(\overrightarrow{RSS_u})} \ . \tag{2.3}$$

As $P(T_i)$ and $P(\overrightarrow{RSS_u})$ are known, the remaining problem is to calculate the reverse probability $P(\overrightarrow{RSS_u}|T_i)$. However, calculating this probability is challenging, because the combination of the RSS from different APs for this vector $\overrightarrow{RSS_u}$ may not be recorded at any training example. However, by assuming independence for individual RSS, the Naïve Bayes approach can be applied to calculate the individual posterior probability for each RSS as follows.

$$P(\overrightarrow{RSS_u}|T_i) = \prod_{j=1}^{N} P(AP_j^u|T_i) \ . \tag{2.4}$$

Since the WiFi APs mostly do not interfere with each other, this is a fair assumption. The individual posterior probability $P(AP_j^u|T_i)$ can be calculated based on how often the individual RSS of the AP_j appears at the training location T_i, which is much easier than calculating the probability of the whole RSS vector.

$$P(AP_j^u|T_i) = \frac{number\ of\ times\ AP_j\ appears\ at\ location\ T_i}{total\ number\ of\ readings\ observed\ at\ location\ T_i}\ . \tag{2.5}$$

Overall, the probabilistic approach takes into account the probability distribution of individual RSS reading, whereas W-KNN uses a single RSS vector per location. The disadvantage of the probabilistic approach, however, is that capturing the full signal distribution at each training location is time-consuming.

2.4.4 Regression fingerprinting with ridge regression

The regression technique is only applicable if the signal label is in the numeric format (e.g. latitude and longitude). While the above classification approaches try to classify the new $\overrightarrow{RSS_u}$ into one of the training examples, the regression approach tries to find a model that fit all training examples $T_i = (\overrightarrow{RSS_i}, \overrightarrow{L_i})$ $(1 \leq i \leq M)$. This model will be used to predict the Cartesian label $\overrightarrow{L_u}$ for the new vector $\overrightarrow{RSS_u}$. Ridge regression is one of the most popular machine learning techniques for regression [22]. Before introducing ridge regression, let's re-cap the mean-squared error (MSE) method, for which ridge regression was based on. When the user provides the signal $\overrightarrow{RSS}$, the trained model predicts the positioning label as $f(\overrightarrow{RSS})$. Assumed that the true positioning label is known as $y(\overrightarrow{RSS})$, the MSE can be calculated as.

$$MSE = \langle (y(\overrightarrow{RSS}) - f(\overrightarrow{RSS}))^2 \rangle\ . \tag{2.6}$$

where $\langle \dots \rangle$ indicated the averaging. This MSE score reports how good the averaging prediction is. It can be further described as.

$$MSE = (y(\overrightarrow{RSS}) - \langle f(\overrightarrow{RSS}) \rangle)^2 + \langle (f(\overrightarrow{RSS}) - \langle f(\overrightarrow{RSS}) \rangle)^2 \rangle\ . \tag{2.7}$$

where $(y(\overrightarrow{RSS}) - \langle f(\overrightarrow{RSS}) \rangle)^2$ is called the bias, and $\langle (f(\overrightarrow{RSS}) - \langle f(\overrightarrow{RSS}) \rangle)^2 \rangle$ is called the variance.

Ridge regression is similar to MSE, however, it introduces a tuning parameter called the ridge factor $\lambda \geq 0$ to control the balance between fitting the data and avoiding the penalty.

The ridge regression model is described below.

$$\mathscr{M} = \sum_{i=1}^{M} (\overrightarrow{L_i} - w \cdot \overrightarrow{RSS_i})^2 + \sum_{j=1}^{N} \lambda ||w_j||^2 \ . \tag{2.8}$$

where w_j $(1 \leq j \leq N)$ are the weights for the WiFi APs. The closer the value of λ is to zero, the more linear the regression model becomes, or the tighter the data will be fit. The bigger the value of λ is, the closer the model is shrank to zero, or the larger the weights are. It is preferred to have a balanced λ to fit a linear model, and to shrink the co-efficient at the same time. This approach may introduce some bias, but has been proved to greatly reduce the variance, which results in a better MSE [22].

The objective is to find the optimal weights w^* so that $\mathscr{M} \to \texttt{min}$. The prediction Cartesian label $\overrightarrow{L_u} = (d_x^u, d_y^u)$ for the new sample $\overrightarrow{RSS_u}$ is obtained as follows. For fingerprinting, where there are two Cartesian labels x, y, each dimension is predicted separately with a different model.

$$d_x^u = w_x \cdot \overrightarrow{RSS_u} \ . \tag{2.9}$$

$$d_y^u = w_y \cdot \overrightarrow{RSS_u} \ . \tag{2.10}$$

The benefit of the regression approach is the high estimation speed for any new sample using the model, which was generated in advance to represent the whole training database. There is no need to go over each training example to find the best match as in the case of classification.

2.5 A comparative performance review of fingerprinting

Fingerprinting is a decade mature technology. How is it keeping up with other infrastructure-based systems, and especially with the latest trends such as inertial based tracking in the same infrastructure-free category? This section assesses the performance accuracy of WiFi fingerprinting in a Microsoft competition where all contestants were ranked under the same test domain. The underlying algorithms of fingerprinting are then analysed individually to understand which options are suitable to perform fingerprinting.

2.5.1 Performance review of fingerprinting at the Microsoft IPSN competition

Since 2014, Microsoft have been organising a yearly indoor positioning competition, where the competitors from the academia, the industry and start-ups come together to evaluate their latest technologies in a realistic, unfamiliar environment. For Microsoft IPSN 2014, the *2,500* square feet evaluation area includes two rooms and the hallway surrounding them[1]. It is interesting to see how well fingerprint-based systems performed in this same test environment with other systems. There were two pure WiFi fingerprint-based only systems in this competition (MapUme & Nanyang). Inertial based tracking dominated the selection of the remaining contestants in the infrastructure-free category. There was no WiFi fingerprint based only systems enrolled in the following two subsequent years.

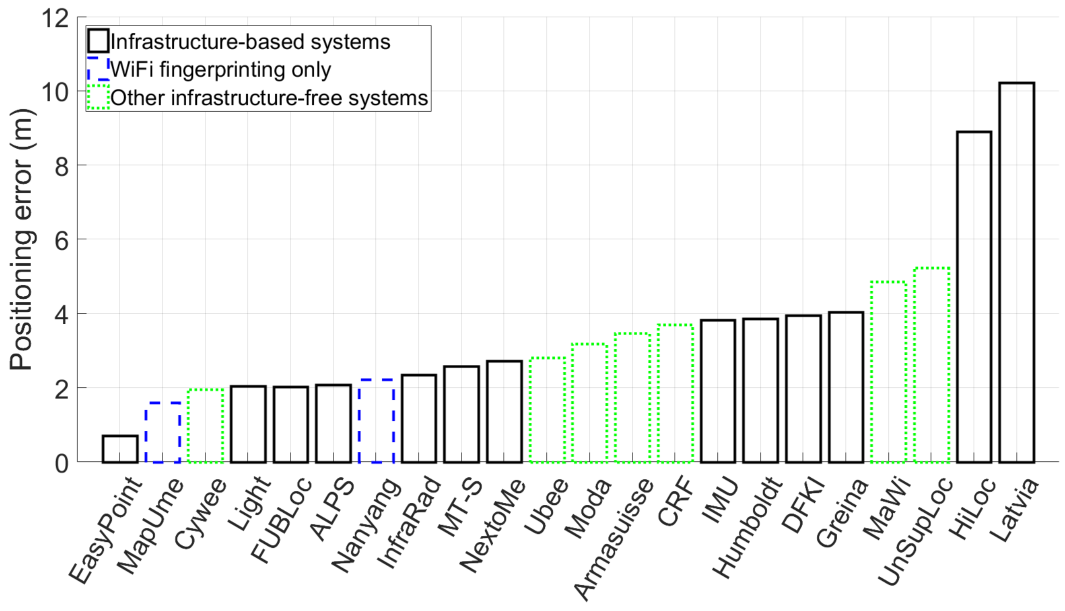


Fig. 2.5 Performance accuracy of fingerprinting at Microsoft IPSN 2014 competition.

Figure 2.5 compares the performance accuracy of the systems enrolled in this competition, where WiFi fingerprinting ranked 2^{nd} and 7^{th} with the positioning accuracy of *1.6* m and *2.22* m respectively, out of *22* contestants including some best papers at the international conferences. In particular, the same fingerprinting systems came 1^{st} and 3^{rd} amongst all *9* infrastructure-free systems. The big surprise was that some hybrid fingerprinting and inertial tracking systems performed less accurate than these two pure fingerprint-based systems.

[1]http://research.microsoft.com/en-us/events/ipsn2014indoorlocalizatinocompetition - last accessed in Sep/2016.

Perhaps the sensor noises from the mobile phone degrade the positioning accuracy of those hybrid systems. Overall, this is a highly encouraging result for fingerprint-based research.

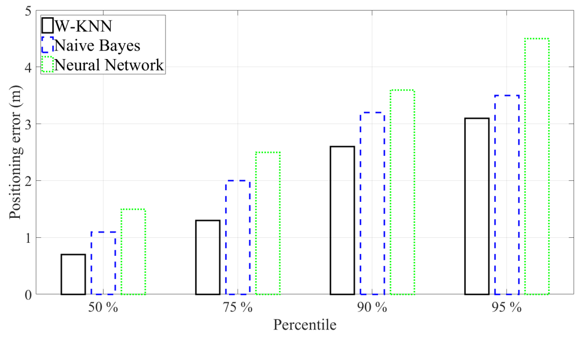
It is worth noting that many of the systems participating in this competition came from the industry and did not reveal much of their underlying algorithms. The next section assesses the impact of different machine learning algorithms for fingerprinting.

2.5.2 Performance review of the machine learning approaches to WiFi fingerprinting

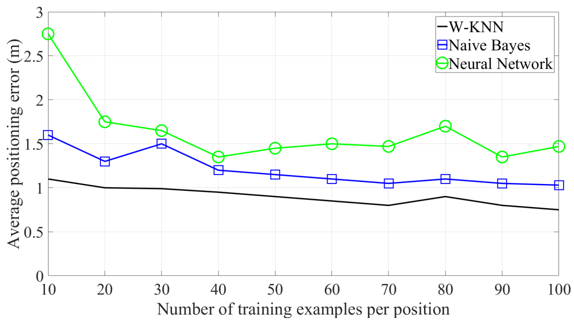
This section assesses the most popular machine learning algorithms for fingerprinting in the literature. In particular, they are W-KNN, Naïve Bayes, Neural Network and Histogram. The Histogram method simply compares the WiFi RSS distribution of both the test position and the training position. The neural network approach uses a mathematical model to recognise the patterns in the database. Once the neural network has been trained on the examples of the fingerprinting database, it is able to predict by detecting similar patterns for the new samples. The experiments were conducted and reported under the same test domain by three independent review papers [20, 35, 56]. To emphasise on the performance of the algorithms, all reviewed systems are WiFi RSS based only. No other technique apart from fingerprinting was employed in these systems.

The first review was conducted in a corridor of *24.6* m by *17.6* m, with at least *5* nearby WiFi APs [56]. A total of *84* training positions were recorded, with *100* readings of the WiFi RSS per position. The training dataset's granularity was *1* m. Figure 6.5a demonstrated that W-KNN had the most accurate performance at *3.1* m, 95% probability. However, it was suggested that the higher the number of training examples per location is, the more performance gain the Naïve Bayes approach may have (see Figure 6.5b).

The second review was conducted on a floor of *2,160* square metres, which is five times larger than the first review's [20]. However, the training points were much sparser with a granularity of *6.2* m covering *56* positions. This sparse training set was compensated by a denser histogram of *224* WiFi RSS per position, covering all *4* orientations (N/W/S/E). This review compared *17* variations of K-NN and Naïve Bayes. Giving such strong RSS coverage for each training position, Naïve Bayes was expected to triumph. However, W-KNN slightly edged out again at *2* m positioning error on average, compared to that of *2.3* m for the Naïve Bayes approach (having applied the filter mode) (see Table 2.2). This review noticed that the Naïve Bayes approach achieved competitive performance with fewer APs than W-KNN.



(a) Performance accuracy of the three algorithms.



(b) Impact on the number of training examples.

Fig. 2.6 Performance accuracy of W-KNN, Naïve Bayes and Neural Network, reported in the first review [56].

Table 2.2 Performance accuracy of nearest neighbours and Naïve Bayes, report in the second review [20].

Algorithms	**Average positioning error (m)**
Weighted K-nearest neighbours, with best orientation	2.2
Naïve Bayes with filtering	2.3
Average of all K-nearest neighbours	2.7
Weighted K-nearest neighbours, closest to all 4 orientations	3.2
Probabilistic Naïve Bayes	3.3
1-nearest neighbour	5.3

The third review was conducted over *3* floors in a university building with a total of *177* training positions, although it was unclear what the granularity of these training points was [35]. The WiFi RSS was recorded repeatedly for *60* seconds at each position, which is expected to collect about half the amount of the WiFi RSS as in the second review. This review compared the performance accuracy of W-KNN, Naïve Bayes and the Histogram method. The Histogram approach compares the WiFi RSS distribution of both the test position and the training position, using Kullback-Leibler, Lissack-Fu. This method may not be applicable in the real-world, where it is difficult to obtain more than one reading per location for the moving users in real-time. In this review, Naïve Bayes came up on top at *12.3* m positioning error, *95%* probability, while W-KNN was just slightly behind at *13.7* m, *95%* probability.

In summary, all three review papers suggested that with only the WiFi RSS as the mea-

Table 2.3 Performance accuracy of nearest neighbours and Naïve Bayes, report in the third review [35].

Algorithms	**Average positioning error (m)**	**Median (m)**	**Max (m)**	**95 % (m)**
Probabilistic Naïve Bayes	5.4	4.1	98.6	12.3
Weighted K-nearest neighbours	5.6	4.4	119.6	13.7
Histogram	7.0	5.0	106.0	19.4

surement metric, many complex algorithms may not perform as well as simpler ones. Despite its simplicity, W-KNN excelled in most fingerprinting reviews. It is worth noting that the MapUme system that came second over *22* contestants in the Microsoft IPSN 2014 competition also employed W-KNN as the main underlying algorithm. However, the Naïve Bayes approach improves its accuracy as the number of training examples per location is high, which is an indication that beside the WiFi RSS, additional information will be needed to enhance the performance of fingerprinting further.

2.6 The fingerprinting test beds

This thesis uses three test beds. The first one (Royal Holloway) was manually collected by the thesis' author in a standard office's environment using a smart phone. The second one (Cambridge) was collected automatically by a robot designed by the thesis' author in a fairly ideal environment, supported by an independent tracking system for ground-truth. The last one (UJIIndoorLoc) covers a huge indoor area spanning across *3* buildings in a very challenging environment. All three test beds are available for further research[2]. The UJIIndoorLoc dataset is publicly available from the UCI Machine Learning Repository[3], and has recently been used for the EvAAL 2015 fingerprinting competition[4]. Each test bed has a large training database which includes both the signal measure (i.e. the WiFi RSS) and its label (i.e. the positioning co-ordinate); and a smaller test database which was collected randomly and separately to provide the test samples that may not be covered in the training set. Their detailed information are described below.

[2]http://www.cs.rhul.ac.uk/∼wruf265/datasets/ - last accessed in Sep/2016.
[3]https://archive.ics.uci.edu/ml/datasets/UJIIndoorLoc - last accessed in Sep/2016.
[4]http://evaal.aaloa.org/current-competition/track-3 - last accessed in Sep/2016.

2.6.1 Royal Holloway test bed

This test bed was collected on the ground floor of the McCrea building at Royal Holloway, University of London. The floor plan of *45.4* m by *32.6* m composes of *3* corridors and *27* offices. Most of the tracking space was in the corridors (see Figure 2.7). There were *9* WiFi APs directly inside the building to provide strong RSS (see Figure 2.8). There were also many other weaker APs from nearby buildings which brought the total number of WiFi APs in the training set to *131*. Any training position can observe at least *28* APs.

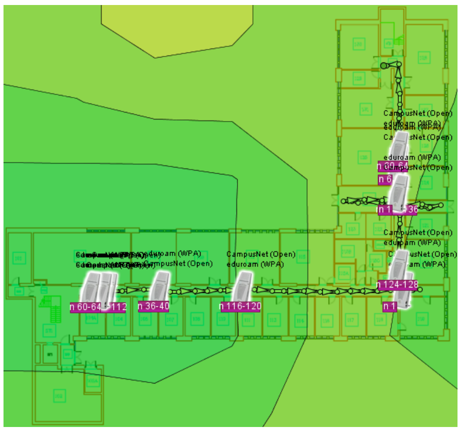
(a) South corridor. (b) Southeast corridor. (c) East corridor.

Fig. 2.7 The Royal Holloway test environment.

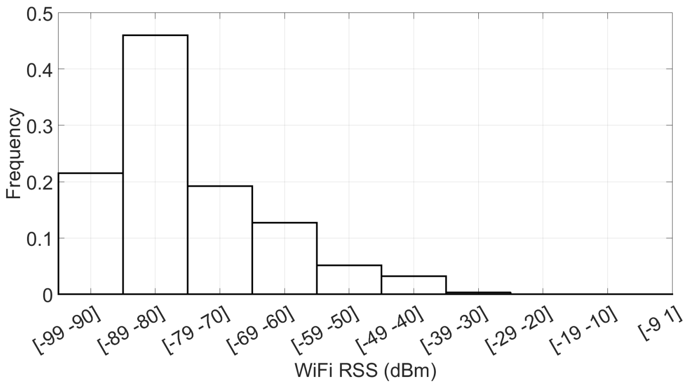
This dataset was generated in a standard manner, taking into considerations of the lessons learnt to produce a good training set. For instance, the granularity was *1* m to cover most positions in the building. Each training position recorded *200* readings to capture the full WiFi RSS variation, and also for the probabilistic methods to work well. The orientation of each training example also covered the four main cardinal directions (N/W/S/E). The collection time was short to avoid the temporal environmental changes. Furthermore, this dataset covers both the WiFi RSS and the magnetic field data, which will be useful for an approach to be discussed later in Chapter 5. The mobile device used to collect the fingerprints was the Nexus 5. An app was developed to support the training process (see Figure 2.9). Out of the three test beds, this one has the highest number of measure per training location.

2.6.2 Cambridge test bed

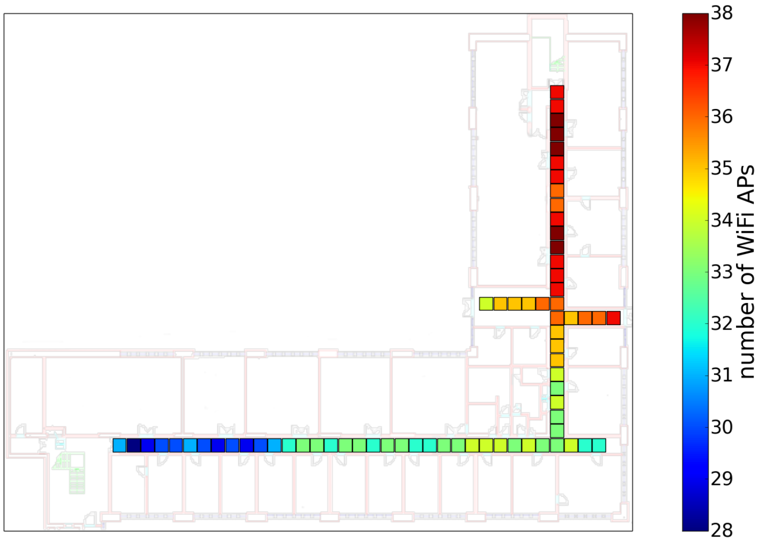
This test bed was collected on the second floor in the North corridor of the Computer Lab at the University of Cambridge. The tracking space contains a long corridor of *45* m by *1.7* m, and a single room of about *29.3* square metres (*6.1* m by *4.8* m) (see Figure 2.10). There were *5* WiFi APs inside this area to provide strong RSS, in which *4* APs were positioned at



(a) The rough position of the 9 WiFi APs inside the building, generated by Ekahau.



(b) The WiFi RSS distribution of the Royal Holloway test bed.



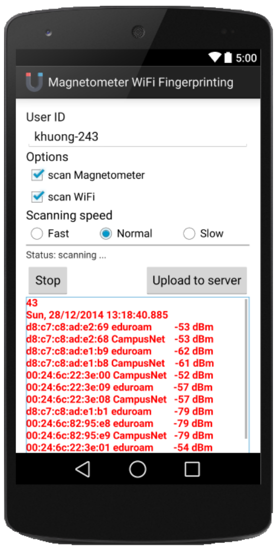
(c) The number of WiFi APs covered by the training positions.

Fig. 2.8 The coverage of the WiFi APs of the Royal Holloway test bed.

the two ends of the corridor, and *1* AP was in the middle of the corridor. There were also weaker APs from the surrounding areas. In total, there are *450* training positions along the corridor and *1,500* training positions in one single room.

A robot designed by the thesis' author was used to collect the training fingerprints (see Figure 2.11). It carries a netbook (Sony P-115) on its back, and an Active Bat tag attached to its head. The netbook has a Java program which is responsible to co-ordinate the robot movements, and to record the WiFi RSS and the Cartesian label provided by the Active Bat system. More details about this robot can be found in [64].

The highlights of this test bed are the high precision of the reference labels provided



User ID
khuong-243
Options
[✓] scan Magnetometer
[✓] scan WiFi
Scanning speed
Fast [✓ Normal] Slow
Status: scanning ...

```
43
Sun, 28/12/2014 13:18:40.885
d8:c7:c8:ad:e2:69 eduroam      -53 dBm
d8:c7:c8:ad:e2:68 CampusNet    -53 dBm
d8:c7:c8:ad:e1:b9 eduroam      -62 dBm
d8:c7:c8:ad:e1:b8 CampusNet    -61 dBm
00:24:6c:22:3e:00 CampusNet    -52 dBm
00:24:6c:22:3e:09 eduroam      -57 dBm
00:24:6c:22:3e:08 CampusNet    -57 dBm
d8:c7:c8:ad:e1:b1 eduroam      -79 dBm
00:24:6c:82:95:e8 eduroam      -79 dBm
00:24:6c:82:95:e9 CampusNet    -79 dBm
00:24:6c:22:3e:01 eduroam      -54 dBm
```

Fig. 2.9 The Android app used to collect the fingerprints for the Royal Holloway test bed.

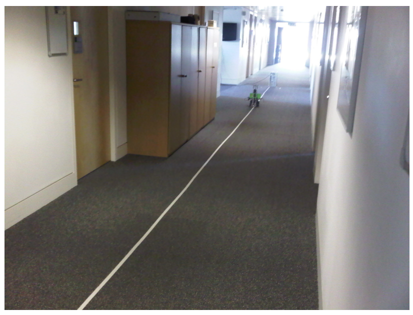
by the Active Bat system (up to *3* cm error, *95%* probability), and the fine-grained *10* cm resolution of the training space (*30* cm resolution for the corridor) that was made possible only with a robot. The environment of this test bed was fairly ideal, with a wide and long corridor, and an empty room with no furniture. The training set was compiled over the weekend with no people around.

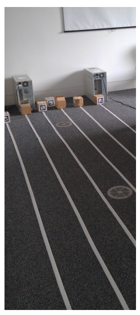
2.6.3 UJIIndoorLoc test bed

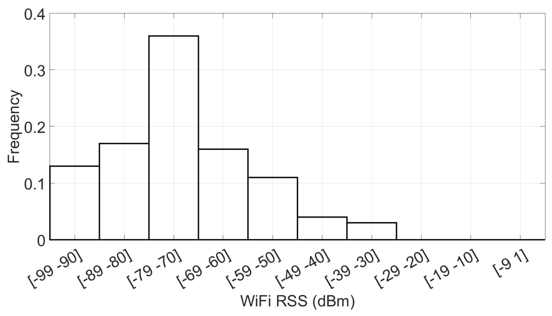
This dataset covers *3* buildings of *110,000* square metres of the Universitat Jaume I in Spain, with a total of *13* floors (see Figures 2.12 and 2.13). Out of the three test beds used in this thesis, this is the only one covering multiple buildings and floors. It was used in the EvAAL fingerprinting competition in 2015[5].

The challenge of this dataset is not just the sheer amount of tracking space, but also the nature in which the dataset was generated. Firstly, there were *25* different mobile devices involved in the process. Secondly, *18* contributors used those devices to record the WiFi RSS without any pre-arrangement. Thirdly, the user taps on the touch screen to label the recorded WiFi RSS. Then, a central server converts this rough estimation into a numeric latitude and longitude label. With so many different contributors, their expectations may be different, therefore, the labels they provide may not be uniform. Lastly, the data collection time was long (*20* days) and the two test sets were collected *3* months and *18* months afterwards,

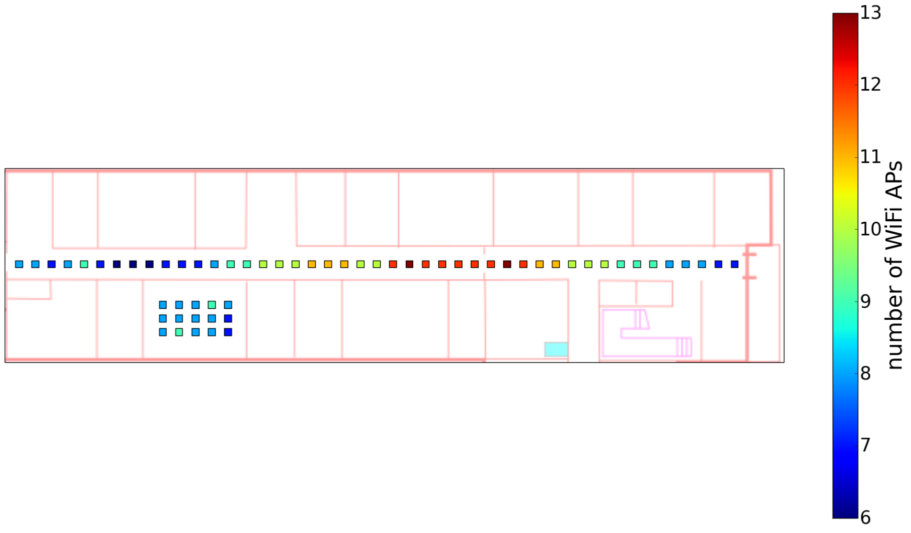
[5]http://evaal.aaloa.org/current-competition/track-3 - last accessed in Sep/2016.


(a) The 45 m x 1.7 m corridor.


(b) The 6.1 m x 4.8 m room.



(c) The WiFi RSS distribution of the Cambridge test bed.



(d) The number of WiFi APs covered by some training positions, at 1 m resolution.

Fig. 2.10 The Cambridge test bed.

which no longer reflect the original state of the indoor infrastructure. These conditions make this test bed highly challenging for any algorithm to estimate the user position.

2.6.4 Summary of the three test beds

In summary, the three fingerprinting datasets used in this thesis were selected to represent three different indoor environments, which include a standard one (Royal Holloway), an ideal one (Cambridge) and a challenging one (UJIIndoorLoc). The Royal Holloway dataset has the highest number of measure per location, while the Cambridge one has the highest training resolution. Out of the three test beds, the UJIIndoorLoc dataset covers the widest indoor space. The majority of the WiFi RSS in all three datasets were around *[-80 -70]*

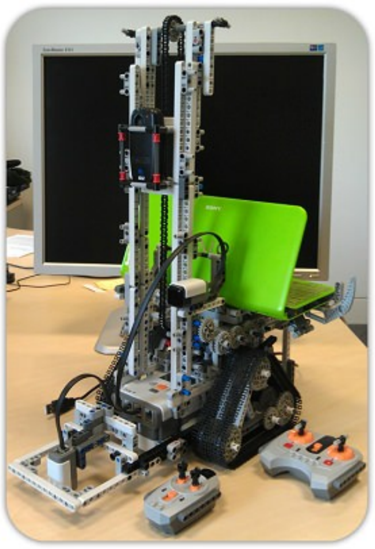


Fig. 2.11 The LEGO robot used to collect the fingerprints for the Cambridge test bed.

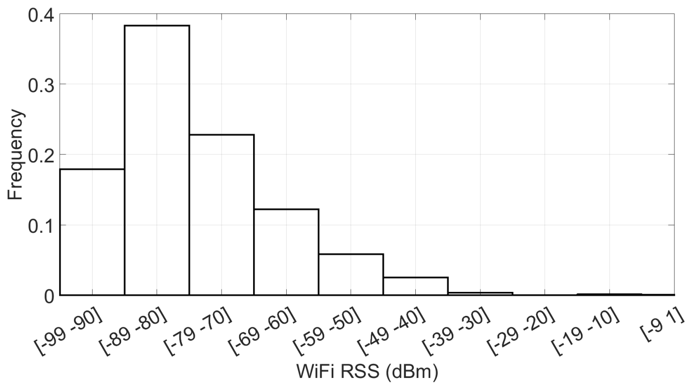


Fig. 2.12 The WiFi RSS distribution of the UJIIndoorLoc test bed.

(dBm). Table 2.4 summaries the characteristics of the three test beds.

2.7 Summary

Fingerprinting has demonstrated its promising potential over the years not just on the research papers, but also in the official competitions with other positioning systems. The strategy to design a suitable algorithm for fingerprinting depends on how the training database was generated. A fine-grained dataset with a dense histogram distribution of the WiFi RSS for each training position will favour a probabilistic approach, although much effort is needed to compile such dataset. Whereas, a sparser training set will favour a determinis-

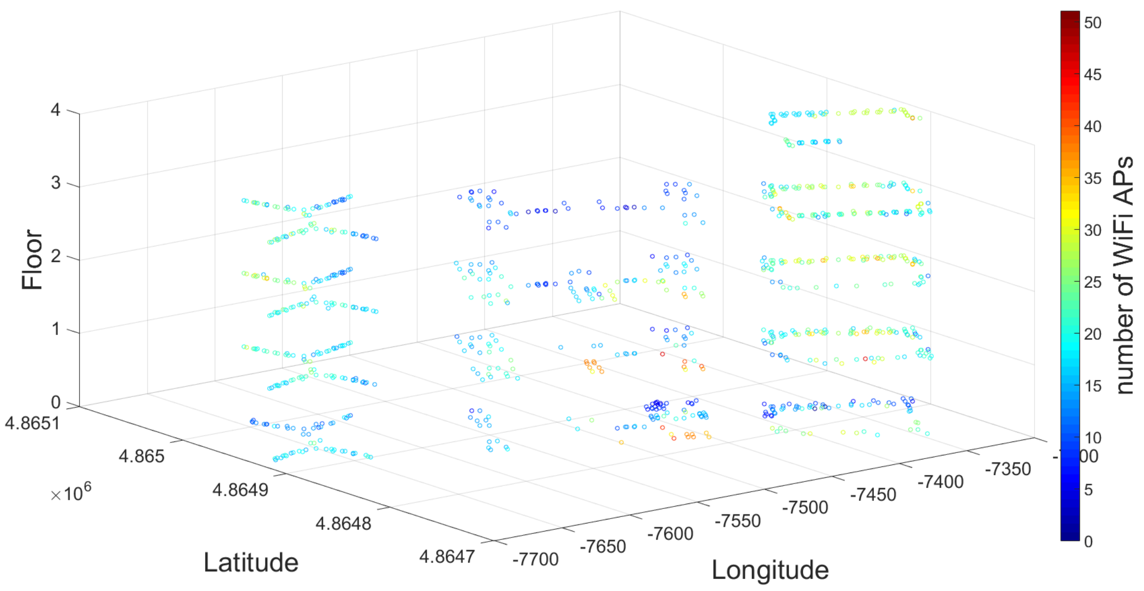


Fig. 2.13 The coverage of the WiFi APs of the UJIIndoorLoc test bed.

tic approach. Many review papers suggested that W-KNN is still the most accurate approach for fingerprinting, despite its simplicity. This is also an indication that complex algorithms may not improve the performance of fingerprinting much further with the WiFi RSS as the only measure. Therefore, the next chapters in this thesis will consider other additional information which may be useful for the fingerprinting research.

Table 2.4 Summary of the three fingerprinting test beds used in this thesis.

	Royal Holloway	**Cambridge**	**UJIIndoorLoc**
Abbreviation	RH	Cam	UJI
Training examples	13,600	78,000	19,937
Training area	1,480 m^2	540 m^2	110,000 m^2
Surveyed space	3 corridors	1 corridor, 1 room	3 buildings
Training time	1 day	2 days	20 days
Granularity	1 m	10 cm for room 30 cm for corridor	Up to 2 m
Test samples	150	100	1,111
Last test sample	Same day	1 day later	3 months later
Fingerprint type	Individual + Sequence	Individual	Individual
Fingerprint metric	WiFi RSS Magnetic field strength	WiFi RSS	WiFi RSS
Measuring time	Working hours	Weekend	Working hours
Measures per location	200	40	Up to 30
Distinct positions	68	1,950	933
Orientations per location	4	4	Unknown
Label type	Cartesian (metre scale)	Cartesian (metre scale)	Longitude & Latitude
Label generator	Manually by the surveyor	Automatically by Active Bat system	Manually by the surveyor
Total building(s)	1	1	3
Total floor(s)	1	1	13
WiFi Access Points	131	43	529
Device(s) used	1 phone	1 netbook	25 phones
Surveyor(s)	1 person	1 robot	18 people

Chapter 3

Off-line fingerprinting database processing with clustering

"Divide each difficulty into as many parts as is feasible and necessary to resolve it."

– Rene Descartes, Discourse on the method (1637).

One of the challenges of fingerprinting is the database inquiring process, which does not scale well with the increasing number of simultaneous users, and the huge amount of tracking space to be covered. This chapter addresses this issue with the popular clustering technique. However, the chapter realised the key difference between fingerprinting and other un-supervised clustering problems is that the Cartesian label of the signal data is known in advance with the fingerprinting dataset. This information was mostly ignored in the literature, where fingerprinting clustering was based solely on the WiFi RSS. By exploiting this concept, the chapter introduces a two-level clustering process, involving both the WiFi RSS and the Cartesian label. This chapter will demonstrate that the proposed scheme does not only reduce the searching time, but also improves the positioning accuracy.

3.1 Introduction to fingerprinting clustering

Clustering is an un-supervised learning technique to partition the data into groups with similar characteristic. It is particularly useful to find the structure of the non-labelled datasets. The clustering techniques are diverse, and can be classified into two general categories. Firstly, is the technique hard clustering or soft clustering? The former partitions the dataset into disjoint groups (e.g. K-Means (KM)), while the latter allows the data point to be a member of more than one group (e.g. Mixture of Gaussians (MoG), Fuzzy C-Means (FCM)). Secondly, is the method flat or hierarchical? Flat clustering such as KM generates a set of clusters with no explicit information to describe the relationship amongst those clusters. On the other hand, hierarchical clustering such as agglomerative uses a tree-like structure to describe the clusters.

Since fingerprinting involves a large training database, clustering has been a compelling approach to organise this dataset for faster real-time searching. For fingerprinting, all of the above clustering techniques can be applied. However, most fingerprinting clustering-based approaches only utilised the WiFi RSS to structure the dataset [2, 21, 55, 83, 92]. The label of the WiFi RSS has mostly been ignored in the main clustering process. In this chapter, four modular steps of a novel clustering scheme which involves both the WiFi RSS and the Cartesian label will be discussed. The chapter opted for a soft clustering technique (MoG) with a two-level tree-like structure, which it believes is a suitable match for fingerprinting for three reasons. Firstly, when the user stands at the borderline of two clusters (e.g. near the wall between the two rooms), he may be misallocated to the wrong cluster. This is a challenging scenario for hard clustering techniques such as KM. Secondly, recent researches pointed out that other clustering techniques such as KM and FCM are sensitive to the training dataset whose training examples are not distributed homogeneously in all dimensions [95]. This problem is well addressed with a mixture model, where the membership function is defined as a probability and the objective function is defined as a likelihood function, to be discussed later in this chapter. Finally, the fingerprinting database is noisy, therefore, the training examples that are grouped into the same cluster in terms of the WiFi RSS, may not be near each other at all in terms of their Cartesian co-ordinate. This is because of the signal multi-path problem described earlier in Chapter 2. This challenge will be addressed with the clustering processes, to be discussed in the next sections.

3.2 Selective mixture of Gaussians clustering

This section presents the four modular steps of the proposed clustering process. Their details are as follows.

3.2.1 Finding overlapped clusters from the training database with mixture of Gaussians

Mixture of Gaussians (MoG) is a probabilistic model-based clustering technique, which uses the Gaussian distribution to model each cluster [9]. This chapter does not attempt to model the WiFi RSS distribution, which may not be Gaussian at all. Instead, it tries to find a model that best fit the entire fingerprinting database. To achieve this goal, a Gaussian distribution model is fit into each group of the WiFi RSS from the training database. The entire dataset is a sum or a mixture of these Gaussian distributions. In contrary to the popular belief that a Gaussian distribution can only be used to represent normally distributed data, a mixture of a sufficient number of Gaussians can represent any data model. The Gaussian distribution is chosen to model each cluster, because of its simplicity, and the ease to estimate the model parameters, which will be discussed later.

The following multivariate Gaussian distribution function is used to calculate the probability that a WiFi RSS vector $\overrightarrow{RSS_x} = (AP_1^x, \ldots, AP_N^x)$ can be observed, with AP_i^x is the RSS (dBm) of AP i^{th} $(1 \leq i \leq N)$, given the mean vector μ and the covariance matrix Σ of a certain cluster.

$$P(\overrightarrow{RSS_x}|\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{N}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\overrightarrow{RSS_x}-\mu)'\Sigma^{-1}(\overrightarrow{RSS_x}-\mu)} \ . \tag{3.1}$$

Let's assume that the fingerprinting database D can be modelled by k Gaussian components (clusters). More details about choosing the value of k will be discussed later. Each Gaussian component $\mathcal{N}(\mu_i, \Sigma_i)$ is given a non-negative weight ω_i to represent the likelihood of occurrence of that component. These weights are important to help deciding which component (cluster) a new sample belongs to later on. For example, it is more likely that an un-observed sample belongs to the cluster with the highest weight. All these weights must sum up to *1* to maintain the probability distribution property of the model. The entire fingerprinting database's model is a mixture, or a weighted sum of these k clusters.

$$P(D|\omega,\mu,\Sigma) = \omega_1\mathcal{N}(D|\mu_1,\Sigma_1) + \cdots + \omega_k\mathcal{N}(D|\mu_k,\Sigma_k) \ . \tag{3.2}$$

$$P(D|\omega,\mu,\Sigma)=\sum_{i=1}^{k}\omega_i\mathcal{N}(D|\mu_i,\Sigma_i),\sum_{i=1}^{k}\omega_i=1\ . \tag{3.3}$$

The objective is to maximise the above probability $P(D|\omega,\mu,\Sigma)$. In other words, the mission is to estimate the value of all the parameters ω_i, μ_i, and Σ_i $(1 \le i \le k)$ simultaneously to maximise the probability of observing the fingerprinting database D. A well-known solution to estimate said parameters is the Expectation-Maximisation (EM) algorithm [62]. It is a two-step iterative procedure, which starts from random guesses of the parameters. In the Expectation step, the probabilities are calculated with the current values assigned to the parameters. In the Maximisation step, these probabilities are then used to update the parameters [95]. The Gaussian Mixture Model package of Matlab[1] was used to perform the EM algorithm in this chapter.

In summary, by the end of this step, a model which best fits the WiFi RSS vectors from the fingerprinting database is generated. Based on this model, a set of k overlapped clusters is found to group the similar WiFi RSS vectors together. An WiFi RSS vector may belong to more than one cluster, and each cluster has a weight to determine its likelihood of occurrence in the probabilistic model. The advantage of such weight and the selection of k will be discussed later on.

3.2.2 Deriving sub-clusters within a cluster

One of the challenges for indoor positioning is the multi-path problem, for which two further positions in the building may have a similar WiFi RSS observation. When such training positions are put together in the same cluster based on their RSS, some of them may not be near each other at all in the Cartesian space. Figure 3.1 illustrates such phenomenon found on the ground floor of the RH test bed. With 7 clusters generated by KM, there were two clear islands which formed the black cluster, and three islands for the blue clusters. Some of the blue members were found inside the cyan cluster, and they did indeed possess similar WiFi RSS with the remaining members of the blue cluster. If a user happens to stay in the minority portions of this blue cluster, his positioning prediction result estimated by considering all members of the cluster will be pulled toward the majority of the blue cluster on the left.

To tackle this challenge, the clustering process is performed again for each cluster found in the previous step. However, the members of each cluster will be judged on their Cartesian (x,y) co-ordinate, rather than the WiFi RSS vector used in the previous step. Ideally, if all

[1]http://www.mathworks.co.uk/help/stats/gmdistribution-class.html - last accessed in Sep/2016.

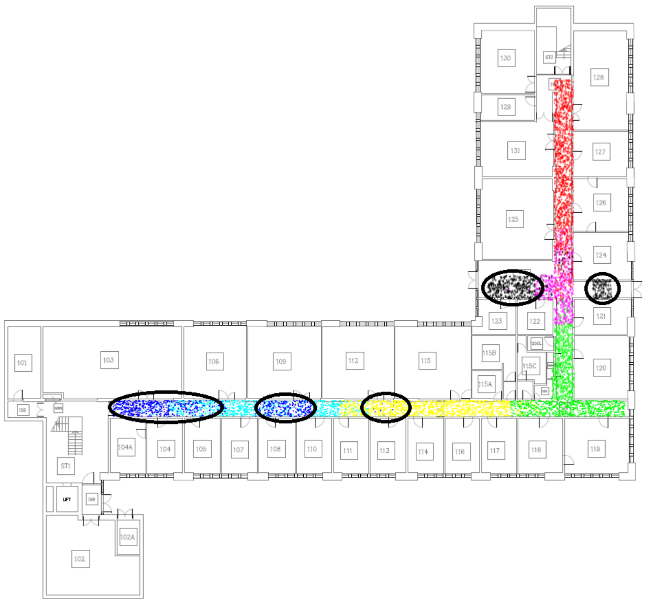


Fig. 3.1 Islands with similar WiFi RSS visualised on a 2-D floor plan of the RH test bed.

members within the cluster are also close in the Cartesian space, only one big cluster will be seen after this process. Otherwise, multiple sub-clusters will be generated. These sub-clusters form a second layer in the proposed two-level clustering scheme. The first layer is the parent cluster of these sub-clusters.

At the end of this step, many sub-clusters will be obtained, for which more than one may have a similar WiFi RSS. However, all sub-clusters are guaranteed to possess different Cartesian characteristics. For consistency, the term sub-cluster will be simply referred to as cluster for the remaining of this chapter. The next part will discuss the strategy to select a set of clusters to represent the user's new signal data at his unknown position.

3.2.3 Selective clusters for on-line positioning

When a user wishes to discover his current position, he uses his mobile device to measure the WiFi data RSS_u at his location. The system will identify which clusters the user's data belongs to. This process is the most challenging and error-prone one for many clustering algorithms. If the user's signal data is put into the wrong cluster, the positioning accuracy will degrade as a consequence. While most previous work picked only a single best matched cluster, this chapter proposes to select a group of clusters instead. Since the probability model of the fingerprinting database has already been constructed, the probability of any new signal data to be associated to each of the k clusters can be calculated by substituting the new signal data RSS_u and the two parameters μ_i, Σ_i $(1 \leq i \leq k)$ of each cluster into Equation 3.4.

$$P(\overrightarrow{RSS_u}|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{\frac{N}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\overrightarrow{RSS_u}-\mu_i)'\Sigma_i^{-1}(\overrightarrow{RSS_u}-\mu_i)} \ . \tag{3.4}$$

By definition, these k probabilities will sum up to *1*. Therefore, a threshold (e.g. *90%*) can be specified and all clusters with the largest probability adding up to the threshold will be selected. For example, with $k = 3$, and the three probabilities are *0.0, 0.2, 0.8*, both the second and third cluster will be picked, for a *90%* threshold. A high threshold will demand more clusters to satisfy, while a smaller one needs fewer clusters. Without loss of generality, given a threshold value θ, $(0 \leq \theta \leq 1)$ and a decreasing order vector of k probabilities $P_k = (P_k^1, \ldots, P_k^k)$, a set of t probabilities $(t \leq k)$ to minimise the below equation are found.

$$arg\ \min_t [\sum_{i=1}^{t} P_k^i \geq \theta] \ . \tag{3.5}$$

The next section will discuss how to estimate the user's position, given the set of clusters found in this step.

3.2.4 Finding the user position

At this stage, a set of clusters has been identified given the user's signal data. this new set of chosen clusters can be treated as a new, smaller fingerprinting database. The readers may apply any favourite algorithm to estimate the user's position, such as Naïve Bayes or W-KNN. Since the main purpose of this chapter is to accelerate the speed of fingerprinting by reducing the search space via clustering, this section will not go over these algorithms in detail. In the next chapter, a more extensive discussion of the algorithms to estimate the user position will be outlined.

3.2.5 Tuning the value of k

Choosing the optimal number of clusters k is challenging. Importantly, a bigger value of k does not necessarily mean longer running time, which is one of the criteria to judge the clustering algorithm. Instead, the criteria for choosing k should be based on the structure of the training database, which, unfortunately, is different for every training set. However, since the clustering process needs to be done just once, it is possible to trial multiple values of k, and a manual decision based on the result from each k is made. On the other hand, a well-known technique to automatically choose the value of k is based on information criterion such as AIC (Akaike Information Criterion) and BIC (Bayesian Information Criterion), which are popular for evaluating model-based clustering [25]. These information criteria measure the information loss corresponding to the model being used. The model with the smallest information criterion (AIC or BIC) is often selected. Previous researchers inclined to favour BIC for model-based clustering, based on their theoretical and empirical studies [18, 44, 73]. The performance of both AIC and BIC will be compared later on.

3.2.6 Bringing it all together

Figure 3.2 illustrates the complete proposed clustering scheme. Given a fingerprinting database mapping the WiFi RSS to the Cartesian co-ordinate, MoG is used to find k clusters based on the WiFi RSS only. Next, MoG is applied again for individual cluster with the Cartesian co-ordinate as the clustering criteria. By the end of the first two steps, a big probability model for the entire fingerprinting database, and a smaller probability model for each cluster are obtained. When a user submits his signal data to discover his current location, the system uses the probability model of the entire fingerprinting database to identify the clusters that the user's signal may belong to. Then, it uses the probability model of each of these clusters to highlight the sub-clusters that best fit the user's signal data. Finally, the individual WiFi RSS members inside the chosen clusters are compared directly to the user data to estimate his position.

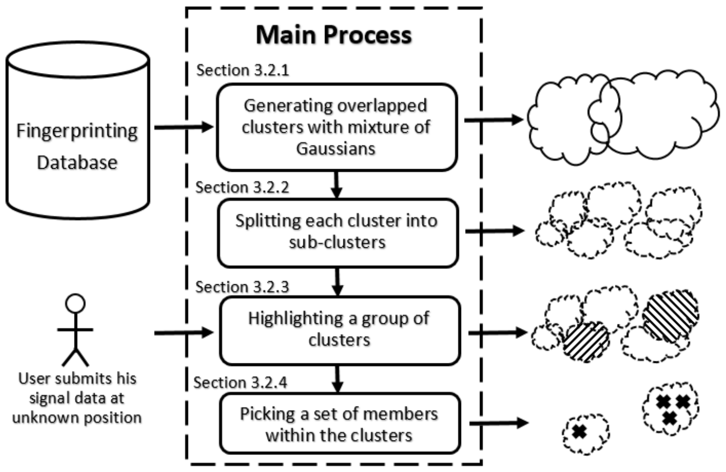

Main Process
Section 3.2.1
Fingerprinting Database
Generating overlapped clusters with mixture of Gaussians
Section 3.2.2
Splitting each cluster into sub-clusters
Section 3.2.3
Highlighting a group of clusters
User submits his signal data at unknown position
Section 3.2.4
Picking a set of members within the clusters

Fig. 3.2 The progress of the proposed clustering scheme.

3.2.7 Related work

In [21, 55], the authors used Affinity Propagation to divide the fingerprinting database based solely on the WiFi RSS, and only when the matching cluster has been identified, the Cartesian labels are used to output the user's location at the end. Similarly, [2, 58, 83, 92] used KM and fuzzy logic to cluster the fingerprinting database with the WiFi RSS only. The proposed approach in this chapter combines the Cartesian label and the WiFi RSS of each training examples to tackle the signal multipath problem. In addition, while other works use only a single best cluster to represent the user's signal data, this chapter selects a group of clusters instead, to minimise the chance of picking up the wrong cluster.

3.3 Evaluation of performance

This section evaluates the performance of the proposed method, using the three test beds described in Chapter 2. In doing so, it aims to verify the following research questions.

(i) **Can clustering improve the positioning accuracy?** Ideally, the main target is to maintain the same positioning accuracy with or without clustering involved in. However, by pinpointing the most relevant clusters, the training examples which are further away may be avoided, and the final prediction result may be improved.

(ii) **How much computational overhead the system can reduce?** There are two computationally heavy processes for fingerprinting clustering. The first one is generating clusters from the off-line training data, and the second one is finding the correct clusters for on-line positioning. Both processes will be assessed here.

The proposed clustering scheme will be compared against KM and FCM. KM is a well-known hard clustering algorithm which partitions data into distinct clusters [32, 59]. It will be interesting to observe how the performance of the overlapped clusters compares to that of the non-overlapped clusters under the same training sets. More details about KM can be found in Appendix A. On the other hand, FCM is a popular clustering algorithm to produce overlapped clusters, which is one of the features of the proposed algorithm in this chapter [8, 23]. Thus, it is natural to choose FCM as a competitor. More details of FCM can be found in Appendix B. The implementations of KM[2] and FCM[3] are provided by Matlab. All three algorithms used *200* iterations for a single cycle, and *100* random cycles

[2]http://www.mathworks.com/help/stats/kmeans.html - last accessed in Sep/2016.

[3]http://www.mathworks.com/help/fuzzy/fcm.html - last accessed in Sep/2016.

with different initial values (different initial centroids) were repeated to warrant data point convergence.

Since the system's design is modular, each individual step will be evaluated separately, and the final result of the entire process will be shown at the end.

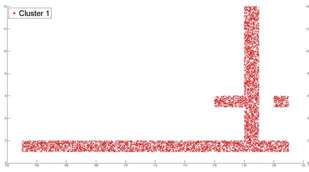
3.3.1 Generating clusters from the fingerprinting database evaluation

To understand how the clusters look like in the real world, with respect to the soft clustering and hard clustering techniques, Figure 3.3 visualises the whole fingerprinting database under different clusters using MoG for soft clustering and KM for hard clustering with the RH dataset. The training examples are clustered based on the WiFi RSS, and then plotted on a 2-D floor map.

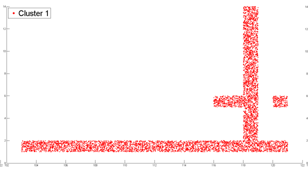
With hard clustering, each cluster is virtually separated, whereas soft clustering allows a much higher degree of freedom for overlapped clusters. With MoG, since each training example has a posterior probability corresponding to the likelihood of appearance for each cluster, the highest probability was used to decide which cluster the training examples belongs to.

The main challenge at this step is to identify an optimal total number of clusters k, so that the clustering algorithms can generate the clusters accordingly. As discussed earlier, AIC and BIC will be used to recommend such number. Figure 3.4 suggests that the lowest value of BIC is around $k = 17$ for the RH test bed, while AIC suggests a much lower k, tested with all the number of clusters between $[1,50]$. The allocation and positioning accuracy (to be evaluated later) proved that BIC was a better indication to select k for the fingerprinting dataset. This result was similar to other reports that AIC tends to overestimate the number of clusters when used in the mixture models [3, 13, 78]. The same experiment suggests that the optimal number of clusters for the Cam test bed is around $k = 11$, while that number is higher at $k = 42$ for the UJI test bed (see Figure 3.4). Although AIC and BIC serve as a convenient means to estimate the number of clusters, many real world applications still decide such number manually by observing each value of k to find out the most optimal number for their datasets. Although this is a time-consuming process, it is only needed to perform just once. For the evaluation purpose, most experiments in this section will examine all the numbers of clusters between $[1,50]$.

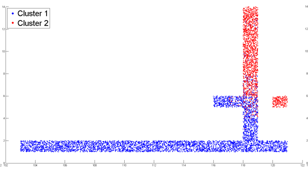
Figure 3.5 demonstrates the size of the individual cluster, the mean and variation of the whole set of clusters generated by each $k = [1,50]$. As the total number of clusters k increases, the box plot of MoG spreads out more, which shows a high degree of variation



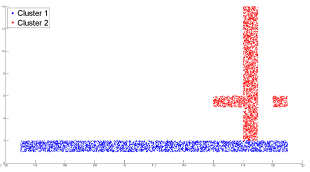
(a) 1-cluster MoG



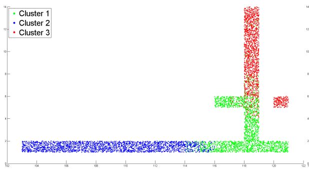
(b) 1-cluster KM



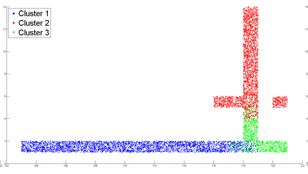
(c) 2-cluster MoG



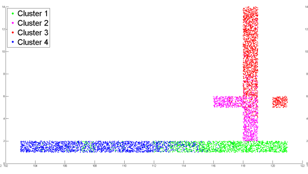
(d) 2-cluster KM



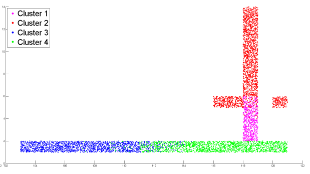
(e) 3-cluster MoG



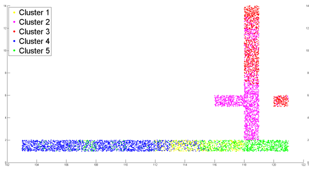
(f) 3-cluster KM



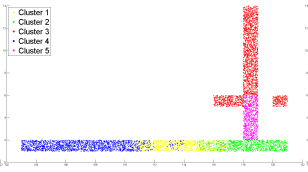
(g) 4-cluster MoG



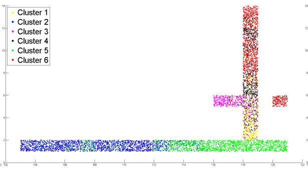
(h) 4-cluster KM



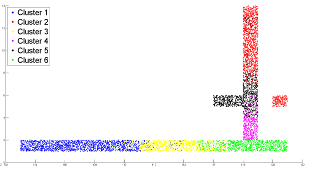
(i) 5-cluster MoG



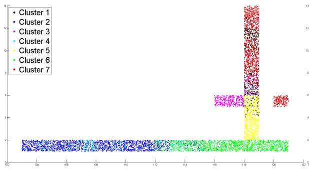
(j) 5-cluster KM



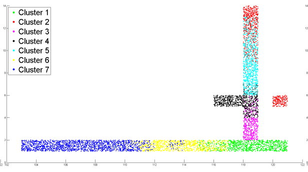
(k) 6-cluster MoG



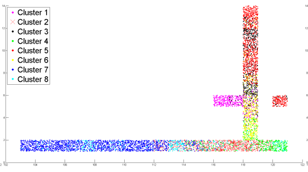
(l) 6-cluster KM



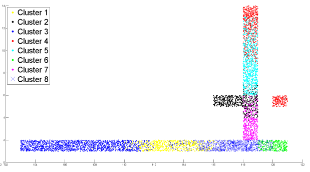
(m) 7-cluster MoG



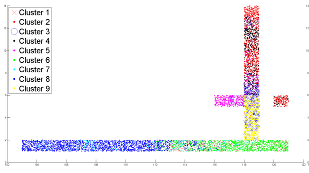
(n) 7-cluster KM



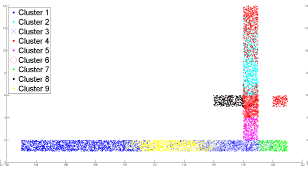
(o) 8-cluster MoG



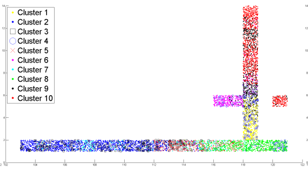
(p) 8-cluster KM



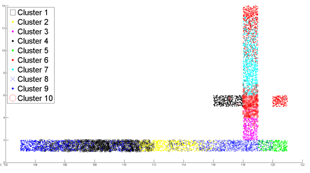
(q) 9-cluster MoG



(r) 9-cluster KM

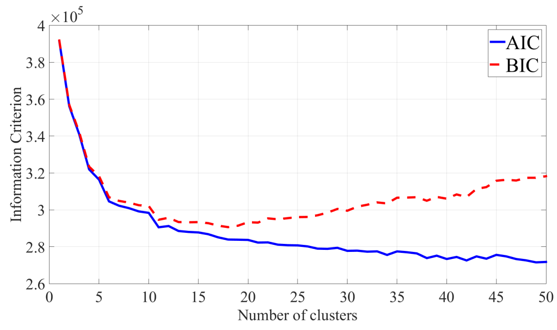


(s) 10-cluster MoG

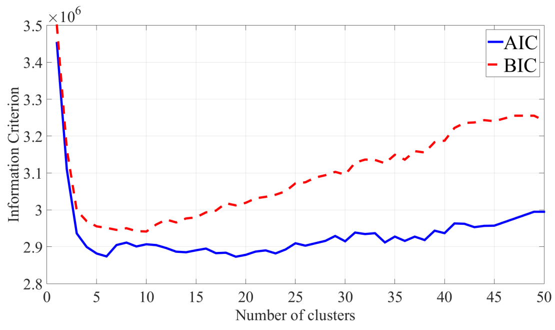


(t) 10-cluster KM

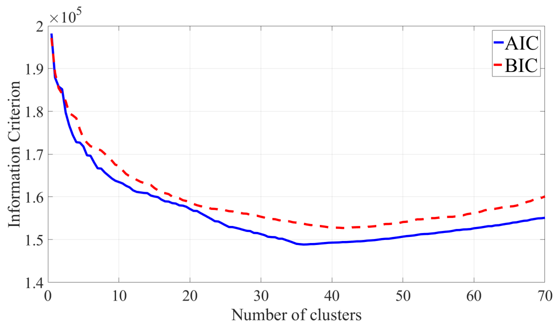
Fig. 3.3 The RH fingerprinting dataset visualised on a 2-D floor plan under soft and hard clustering.



(a) RH test bed.



(b) Cam test bed.

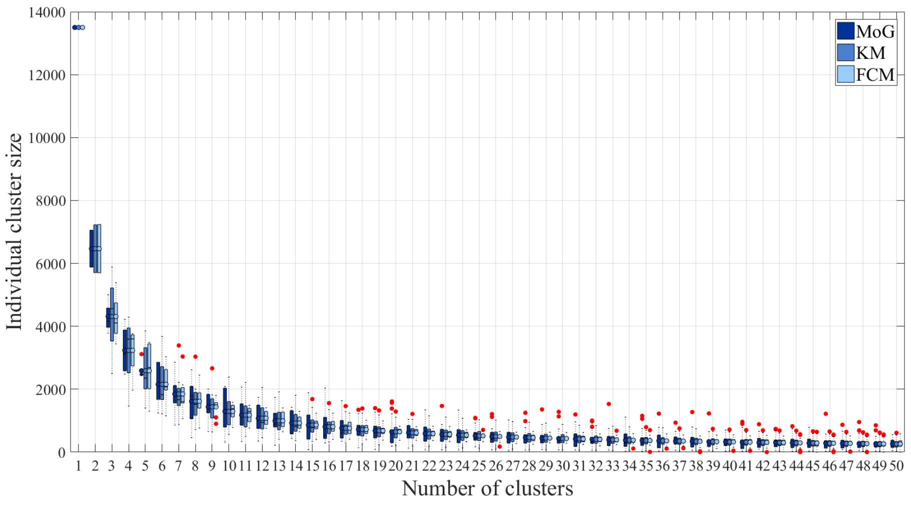


(c) UJI test bed.

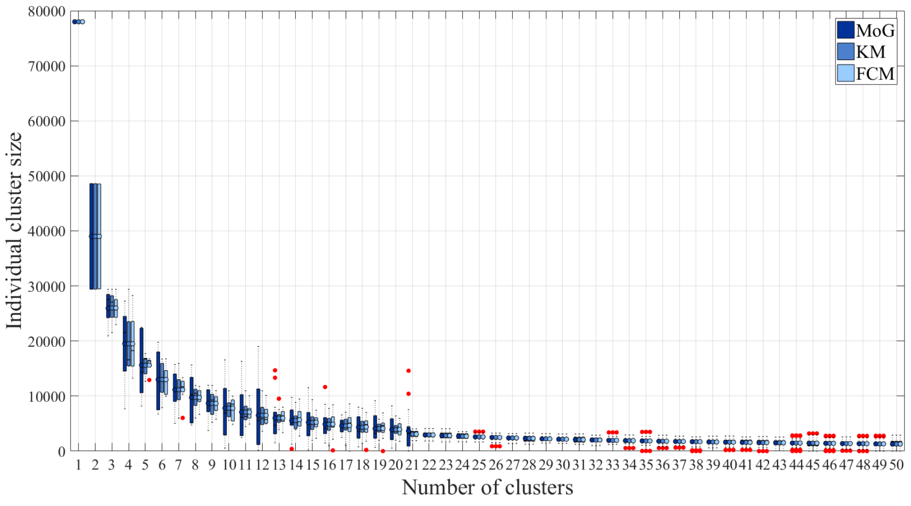
Fig. 3.4 Optimal number of clusters recommended by AIC and BIC for MoG.

amongst the clusters' size. With MoG, very big clusters were noticed. On the other hand, KM maintained a relatively steady size for its clusters, with very little variation for all three test beds.

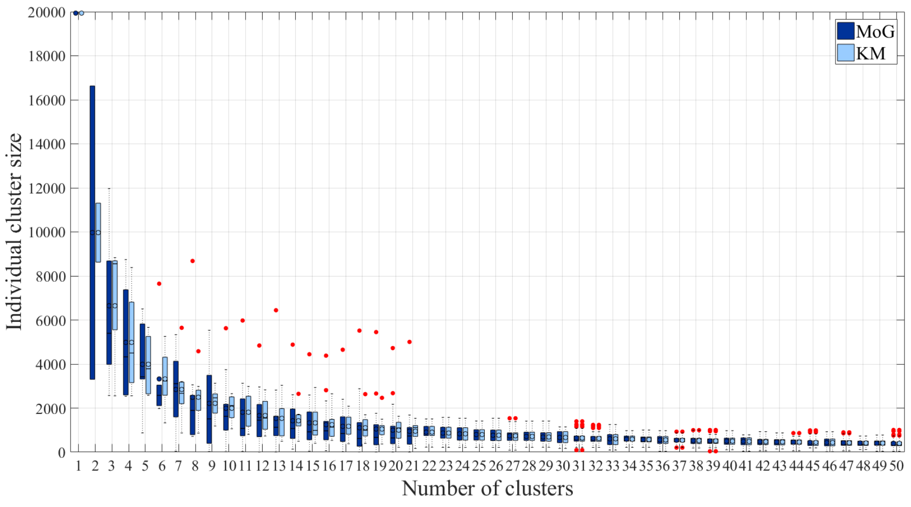
Importantly, FCM failed to produce the correct number of clusters, for certain high values of k, such as *38,42,44,45,48,49* and *50* for the RH test bed. In addition, FCM failed with many k for the UJI test bed, which has a high number of WiFi APs per fingerprint. Thus, the result of FCM for the UJI dataset was not included for the experiments in this chapter. Despite the efforts in attempting different search iterations, FCM did not manage to assign any member to all required k clusters, resulting in several empty clusters. This result coincides with recent researches, which also reported that FCM has problems with high dimensional dataset and high number of clusters [90]. It was explained that the objective function of FCM has a local minimum that is in the centre of the training set. When the number of dimension is high, all centroids run into the same middle point of the data [90]. Previous works in FCM-based fingerprinting did not report this issue, although it was noticed that their datasets were much sparser and did not have as many WiFi APs as the ones used in this thesis [83, 100].



(a) RH test bed.



(b) Cam test bed.



(c) UJI test bed.

Fig. 3.5 Individual cluster size, mean and variation. The red dots are the outliers, which are very big and very small clusters.

3.3.2 Deriving sub-clusters evaluation

After the clusters have been generated in the last step based on the WiFi RSS with MoG, each cluster should have a similar WiFi RSS observation. This step attempts to split each cluster into sub-clusters with MoG, this time, based on the Cartesian labels, as explained earlier in Section 3.2.2. BIC will be used to recommend how many sub-clusters to split, in the same manner as it is used to recommend the optimal number of cluster in the last step. Table 3.1 summaries the amount of sub-clusters for the three test beds with this approach. The next section will evaluate the benefit of these sub-clusters, in terms of the performance accuracy using real test samples.

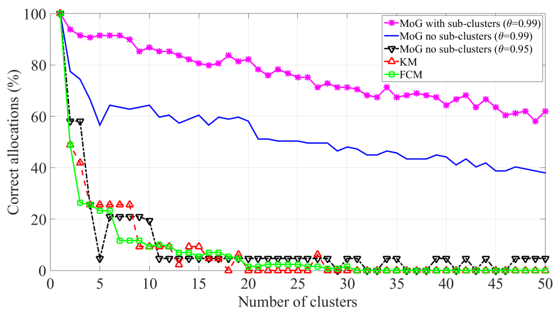
Table 3.1 Total recommended number of clusters and sub-clusters for the three test beds with MoG.

	RH	**Cam**	**UJI**
Total clusters (excluding sub-clusters)	17	11	42
Total clusters (including sub-clusters)	46	52	122
Average sub-clusters per cluster	2.7	4.7	2.9
Max sub-clusters per cluster	4	6	5
Min sub-clusters per cluster	1	1	1

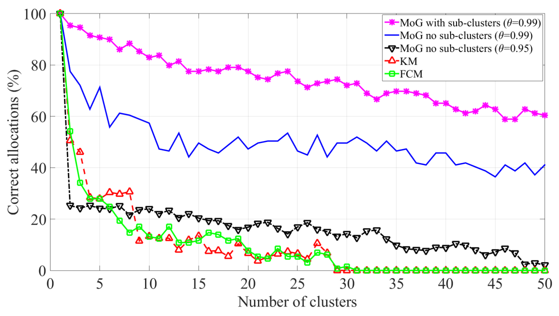
3.3.3 Selective clusters evaluation

This experiment uses the independent test sets to evaluate how successful the clustering algorithms allocate the test samples into their correct clusters generated earlier from the training database. These test sets were collected separately and may contain samples which were not seen in the training sets, as discussed earlier in Chapter 2. For each test sample, the true Cartesian co-ordinate was used to judge the decision making of the clustering algorithm. The test sample is declared to be correctly allocated to the cluster X, if the Euclidean distance between the true label and the label of the centroid of X is minimal. If there exists another cluster Y whose centroid is closer to the true label, this test sample's allocation is marked as incorrect.

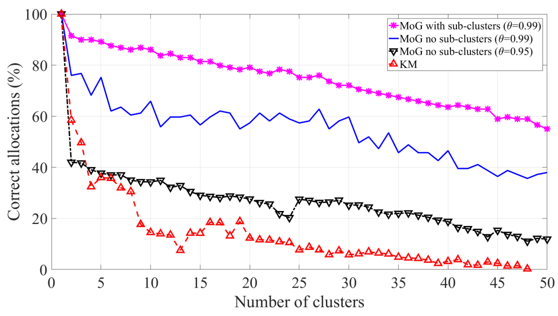
This experiment emphasises on two observations. Firstly, how the overlapped clusters may benefit the new sample's cluster allocation process, compared to rigid distinct clusters. Secondly, what is the advantage of having sub-clusters that were split based on the Cartesian



(a) RH test bed.



(b) Cam test bed.



(c) UJI test bed.

Fig. 3.6 Number of correct allocations based on the Cartesian label.

labels. The result with FCM for the UJI test bed was not included since it failed to generate clusters for this particular test bed, as explained earlier in Section 3.3.1. Figure 3.6 gives a good indication that having overlapped clusters clearly allows more test samples to be put into their correct clusters. In particular, at $k = \{17, 11, 42\}$ for the RH, Cam and UJI test beds respectively, MoG (without sub-clusters) achieved almost *60%*, *45%* and *40%* correct allocations. While the results of KM at the same number of clusters were only about *8%*, *12%* and *5%* for the same test beds.

Although the results with overlapped clusters outperformed those with distinct clusters, the correct allocations still were not particularly high. This was because the clusters were generated based on the WiFi RSS only, therefore, the members within each cluster may not be close at all in terms of the Cartesian labels. By continuing to derive each cluster into sub-clusters, using the Cartesian label as the clustering criterion, the correct allocation was increased from about *60%*, *45%* and *40%* to about *80%*, *83%* and *67%* at the same k for the RH, Cam and UJI test beds respectively. For all three test beds, the highest observed probability for a single cluster was above *90%* on average, which showed that the majority of the test samples were confident that they were in their correct clusters. The value $\theta = 0.99$ was set to allow more than one cluster to be considered with MoG.

Overall, with MoG, just about *20%* of wrong allocations were found for the majority of k, while KM and FCM struggled from $k = 10$ onwards for all three test beds. The next section will discuss how these wrongly allocated test samples impact the positioning accuracy.

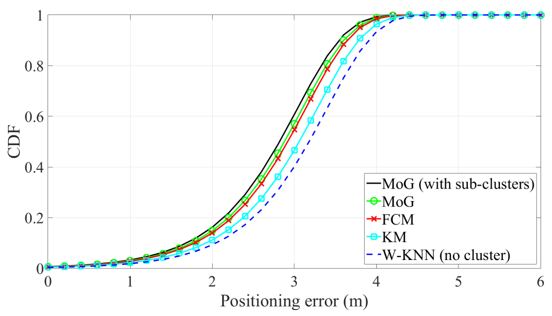
3.3.4 Positioning error evaluation

When the user submits his new signal data for the system to estimate his position, it is interesting to know how accurate the proposed clustering solution is. Under the same fingerprinting database, each clustering algorithm produces a different set of clusters. Each algorithm also has its own means to recommend the clusters for a new test sample. To evaluate the performance of each clustering algorithm, after the algorithm has recommended the clusters for the test sample, these clusters will be treated as a new smaller training database. Then, W-KNN will be used to estimate the Cartesian label for each test sample. This estimated label is compared to the true label of the test sample to work out the positioning error. All the values of neighbours from $[1,50]$ for W-KNN will be tested, and the final results are averaged. The performance of the three test beds will be compared under the Cumulative Distribution Function (CDF) to highlight the benefit of using sub-clusters, the accuracy enhancement (if any) with clustering, and the overall positioning accuracy of MoG, KM and FCM.

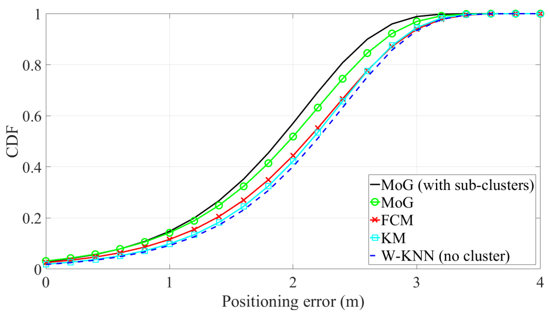
At $k = 17, 11$ and 42 clusters for the RH, Cam and UJI test bed respectively (the optimal k for each test bed), the positioning precision was improved from about *3.5* metres, *70%* probability when no clustering was involved, to about *3.5* metres, *90%* probability, with MoG for the RH test bed (see Figure 3.7). With the Cam test bed, the precision was improved from about *2.5* metres, *70%* probability without clustering, to about *2.5* metres, *85%* probability, with MoG. With the UJI test bed, the precision was improved from *10* metres, *85%* probability without clustering, to *10* metres, *95%* probability, with MoG. This results confirmed that, by pinpointing the most relevant clusters, it was possible to avoid further away training examples and improved the performance accuracy. Compared to KM, MoG and FCM produced better estimation, which confirmed the benefit of having overlapped clusters for fingerprinting.

3.3.5 Processing speed evaluation

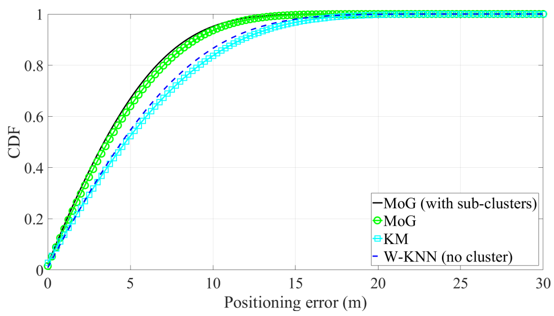
The preparation time, from which the clusters are generated, is undoubtedly the most time-consuming process. The larger the fingerprinting database is, the longer it takes to generate clusters. This experiment measures the total time needed to generate the clusters for the



(a) RH test bed with $k = 17$ clusters.



(b) Cam test bed with $k = 11$ clusters.



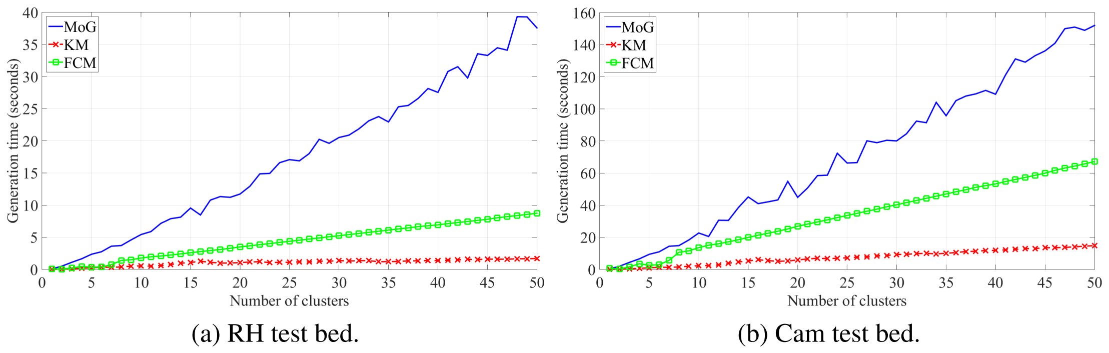
(c) UJI test bed with $k = 42$ clusters.

Fig. 3.7 Positioning accuracy of clustering with the three test beds.

training database of each test bed, using the Matlab clustering package, running on a Core i7 3.5 GHz CPU. Figure 3.8 points out that KM was the fastest option amongst the three algorithms to generate clusters, whereas MoG needed a much longer time. All three algorithms used *200* iterations for a single cycle, and *100* random cycles with different initial values (different initial centroids) were repeated for each number of cluster to warrant data point convergence. The total processing time for MoG also included the sub-clusters generation time.

However, when it comes to identifying which clusters the new WiFi RSS sample belongs to, MoG performed as quick as KM (see Figure 3.9). This is expected, because only the new WiFi RSS data is needed as an input for the Gaussian equation, along with the mean vector μ and the covariance matrix Σ found in the model, to calculate the probability of this new signal data for each of the k clusters (see Equation 3.4). This is an advantage of MoG, considering that its prediction accuracy was better than that of KM and FCM as evaluated previously. A similar result in terms of the new sample's allocation speed was observed for all three test beds.

Without clustering, every single training example must be searched through at least once for each test sample. With the proposed clustering approach, the system only needs to look



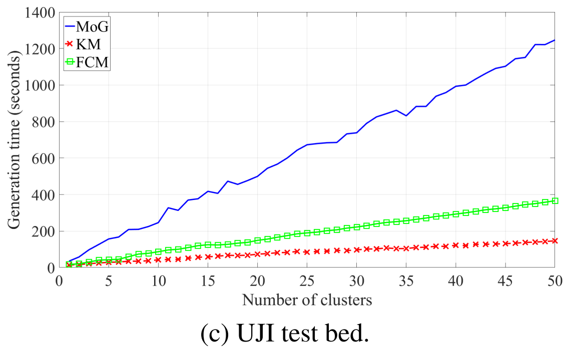
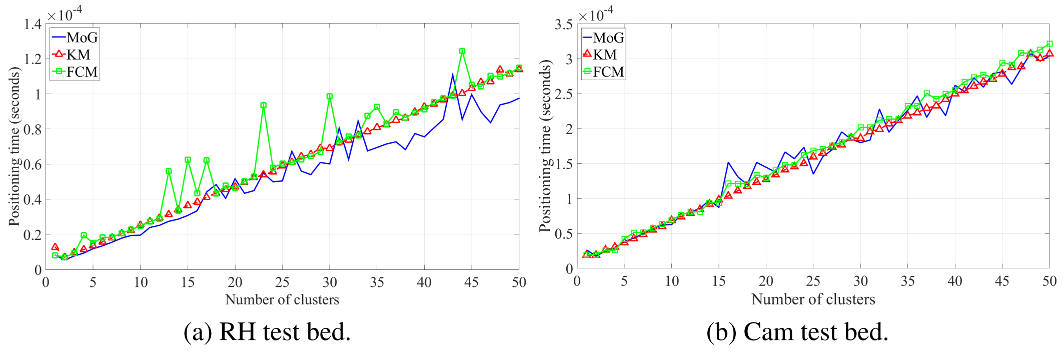


Fig. 3.8 Cluster generation time from the fingerprinting training database.



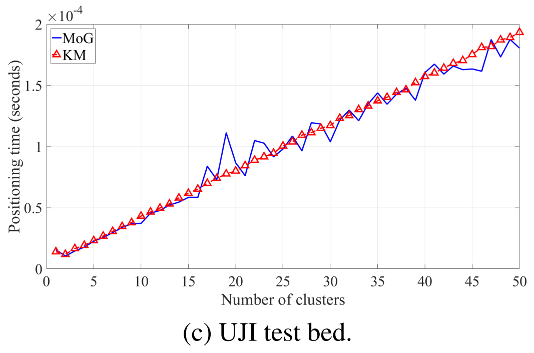
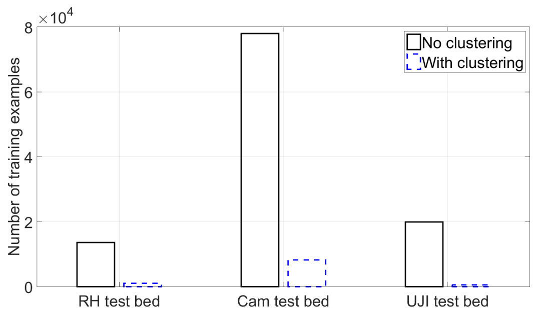
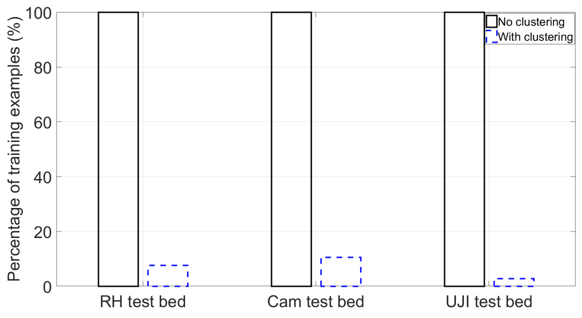


Fig. 3.9 Average time to identify cluster for the new sample.

through all clusters and their sub-clusters once to decide the most relevant candidates. Then, it needs to go through every member inside these chosen clusters to calculate the Cartesian position for the test sample. The smaller the number of clusters, the quicker the first process will finish, yet, the size of the individual clusters will be large. Hence, the second process will run slower. In contrast, the bigger the number of clusters, the faster the second process will finish. Figure 3.10 demonstrates the average reduction in the computational overhead for all three test beds, using the optimal number of clusters suggested by BIC as previously discussed in Sections 3.3.1 and 3.3.2. With clustering, the system saves up to *90%* in terms of the training examples needed to go through for the RH and Cam test beds, and up to *95%* for the UJI test bed.



(a) Average reduction in the number of training examples.



(b) Percentage of reduction in the number of training examples.

Fig. 3.10 Computational overhead reduction via clustering.

3.4 Summary and further work

This chapter has proposed and demonstrated a novel idea to organise the fingerprinting training database for faster searching, which will be useful for the later on-line positioning phase. The proposed approach addresses the problem of the user standing at the borderline of two clusters, where his signal data may belong to either cluster. Furthermore, the chapter realised the key difference between fingerprinting and other un-supervised learning problems, that is, the Cartesian label of the signal data is known in advance. This key information was mostly ignored in the previous works, where the fingerprinting data was clustered based solely on the WiFi RSS. By exploiting this information, the chapter described a two-level clustering process, where each cluster is further divided into sub-clusters based on their Cartesian label. Finally, in contrary to the convention of choosing a single best cluster for the new signal data, the proposed approach made use of the probability information of each cluster offered by mixture of Gaussians to select a group of clusters.

The proposed clustering scheme was tested under three test beds. The empirical results concluded that the approach effectively reduced the positioning time. More importantly, the proposed clustering method with MoG allowed the clusters to overlap which outperformed hard clustering methods such as KM. In addition, it maintained a lower positioning error than non-clustering algorithms, thanks to the attempt in pinpointing the most relevant areas of interest (clusters).

The method of selecting the number of clusters and sub-clusters in this chapter was based on AIC and BIC. Some potential further work is to investigate how to determine an optimal number of clusters, based on the size of the clusters. Some possible considerations for further investigation are outlined below.

(i) The total number of clusters k should be small, in proportional to the size of the fingerprinting database. Since MoG already allows the clusters to overlap, a high number of clusters may not benefit from having sub-clusters.

(ii) The size of the cluster should be large. If a cluster is too small, it may not be beneficial to split it much further.

In summary, this chapter has demonstrated a novel method to organise the fingerprinting training database, so that the training examples can be accessed faster and easier in later inquiries. The next chapter will demonstrate how to use this training database to estimate the user position.

Chapter 4

Estimating the user position with confidence machine

"The enemy of human progress is absolute certainty."

– Brian Cox, The values of science (2016).

Machine learning algorithms have been playing an increasingly important role for fingerprint-based systems to estimate the user position. There is a plethora of machine learning options that are all capable of using the training database to make predictions. However, most of them make no assumption about the quality of the predicted position. In reality, given the noisy nature of the indoor environment reflecting in the training data, it will be useful to provide some information about the uncertainty of the positioning estimation.

This chapter proposes a novel supervised confidence learning algorithm based on Conformal Prediction, that is, to the best of the author's knowledge, the first approach to provide a confidence measure for the fingerprinting research. This confidence measure does not only reflect the uncertainty of the positioning prediction, but is also capable of adjusting the size of the prediction set. The chapter will discuss two implementations of this confidence machine approach under classification and regression. Their performances are directly compared to the traditional W-KNN and Naïve Bayes to demonstrate the improvement in the positioning accuracy.

4.1 Introduction to confidence machine

The concept of confidence machine is that a prediction made by any learning algorithm should be governed by a confidence parameter measuring the belief of the algorithm on this prediction. The confidence learning algorithm that is used in this chapter is called Conformal Prediction (CP), which produces a set of predictions, given a new sample, a training database and a confidence level [75, 85]. For instance, given a *95%* confidence level, a training database with *3* training examples $\{t_1, t_2, t_3\}$ and a new sample s, the set of predictions that CP produces is $\{t_1, t_2\}$, which can be interpreted as *"I am 95% confident that 's' belongs to this prediction set. However, there is 5% chance that I may be wrong"*. It has been mathematically proven that in the on-line setting, where many predictions are performed one after another with the new samples added into the training data after each prediction, CP is correct in the sense of maintaining such error rate. Even with off-line learning using the same training database to make predictions, the confidence level offered by CP can adjust the size of the prediction set. For instance, using the same above example, given a *100%* confidence level and a new sample s, CP produces a prediction set $\{t_1, t_2, t_3\}$, which is actually the whole training data. This is interpreted as *"I am 100% confident that 's' must belong to this prediction set, and there is no chance that I can be wrong"*. This prediction set is not too useful, however, although it is statistically correct. As the confidence level decreases, CP automatically reduces the size of the prediction sets accordingly. Ideally, it is preferred to have a high confidence level with a small prediction set, which may be achieved with different nonconformity measures. More details of how this process works will be discussed in the next section. Overall, the use of confidence machine offers the following benefits.

(i) Each prediction has an associated confidence level to express how likely the prediction is correct.

(ii) The predictions produced by CP are statistically correct under the on-line setting.

(iii) The confidence level can be adjusted to produce a bigger or a smaller prediction set.

CP has been successfully tested in some real-world applications such as cancer diagnosis, image analysis and network traffic prediction [6, 19, 50, 51]. This is the first time that CP is used for the indoor positioning research. Empirical studies at the end of this chapter will demonstrate that the proposed algorithms with CP perform up to *20%* more accurate than other systems without confidence machine.

4.2 Fingerprinting positioning estimation with Conformal Prediction

Conformal Prediction is a machine learning algorithm which makes predictions based on how well the new sample can fit into the training data. To do so, it uses a function called the nonconformity function to measure the difference between the new sample and other training examples. The smaller the value of this function is, the higher the probability that the new sample will fit in. Ideally, when predicting the label for a new sample, it is preferred that this sample is similar to all training examples. So that, the training sequence's randomness can be maintained. The motivational idea of CP came from Kolmogorov's algorithmic notion of randomness for computable approximations [75, 85]. The only assumption which CP requires is that the training examples are exchangeable, that is, the order they appear does not have an impact on the prediction. With fingerprinting, this is not a problem since the training positions were surveyed independently. CP is flexible, in the sense that it allows any nonconformity function to be used. However, the efficiency of the predictions will depend on such function. In this chapter, two nonconformity functions will be presented, one for classification and the other one for regression, because fingerprinting may be considered as either a classification or a regression problem. Their efficiencies will be evaluated later on.

4.2.1 Classification conformal prediction for fingerprinting

Without loss of generality, the CP classification fingerprinting is formally modelled as follows. Given the training database $T = (T_1, \dots, T_M)$ where each training example $T_i = (\overrightarrow{RSS_i}, \overrightarrow{L_i})$ maps the WiFi RSS vector $\overrightarrow{RSS_i}$ to its Cartesian co-ordinate $\overrightarrow{L_i} = (d_x^i, d_y^i)$ $(1 \leq i \leq M)$, a new sample $T_{M+1} = (\overrightarrow{RSS_{M+1}}, \overrightarrow{L_{M+1}})$ at an unknown location $\overrightarrow{L_{M+1}}$, with known $\overrightarrow{RSS_{M+1}}$, and a confidence level $(1-\xi)$, with ξ is the significance level, CP will find a set of training examples for the new sample T_{M+1} in the following three steps.

Firstly, the new sample T_{M+1} is added into the training database, with the label $\overrightarrow{L_{M+1}}$ being assumed as one of the training labels $\overrightarrow{L_i}$ $(1 \leq i \leq M)$. Then, given a nonconformity function $A(T, T_i)$ (which will be discussed soon, for now such function is assumed to exist), the nonconformity measure $\alpha_i = A(\{T_1, ..., T_{i-1}, T_{i+1}, ..., T_{M+1}\}, T_i)$, $(1 \leq i \leq M+1)$ is calculated for every training example. This α score demonstrates the difference between the example T_i and all other training examples including the new sample. Intuitively, the algorithm wants to observe how well the new sample with the assumed label fits into the whole training database. However, for this reason, since this nonconformity function may be designed in any way the readers want, the score α_{M+1} of the new sample, by itself does not tell how similar A finds T_{M+1} to be. For that purpose, α_{M+1} needs to be compared to other α_i.

Thus, the second step uses the α_i $(1 \leq i \leq M)$ of all training examples to calculate a p-value which represents the adaptability of the assumed label L for the new sample, as follows [75, 85].

$$p(L) = \frac{\#|\{j = 1, \ldots, M+1 : \alpha_j(L) \geq \alpha_{M+1}(L)\}|}{M+1} \quad . \tag{4.1}$$

This p-value lies within $\frac{1}{M+1}$ and 1 to indicate the fraction of the training examples that are similar to the new sample. The higher the p-value is, the better it indicates that the assumed Cartesian label L helps the new sample T_{M+1} fit into the training data. Otherwise, the lower the p-value is, which means α_{M+1} is much bigger than the majority of other α_i, the stronger the indication is that the assumed label L makes this new sample an outlier. This process is repeated for the remaining training labels to calculate a p-value for each label.

In the third and final step, the algorithm outputs the predictions based on the user's requirement. If the user wants a single prediction, the label with the biggest p-value is chosen as the predicted label. If the user prefers a prediction set, the algorithm will ask for a confidence level $(1-\xi)$ from *0%* to *100%*, where ξ is called the significance level. Then, any training label with a p-value greater than ξ will be included in the prediction set Γ^{ξ}.

$$\Gamma^{\xi}(T_1, \ldots, T_{M+1}) = \{L | p(L) > \xi\} \quad . \tag{4.2}$$

The last objective of this section is to define the nonconformity function $A(T, T_i)$ to calculate the difference between a training example T_i and all other training examples, as mentioned above. It is worth reminding that CP produces valid predictions with any nonconformity function, although the prediction's efficiency may vary. The guidelines in [75, 85] recommend that the nonconformity function should involve both the label and the object. For CP classification, the function should compute the difference between the nearest training example with the same label and the nearest training example with a different label [75, 85]. However, with fingerprinting, the nearest training example may not necessarily be the optimal one due to the noisiness of the WiFi RSS and the signal multi-path problem as explained in Chapter 2. Therefore, this chapter employs W-KNN to consider a set of K training examples, where different values of K will be evaluated later on. In principle, the idea is to group these K training examples into one weighted averaged position, and follows the same guidelines as described above. The details of such nonconformity function will be explained below. The concept of W-KNN has been described earlier in Section 2.5.1.

The first step in calculating the nonconformity function $A(T,T_i)$ for the training example T_i is to find K other training examples $\{T_1,\dots,T_K\}$ which have the smallest Euclidean distance in terms of the WiFi RSS to T_i. These examples must also have a different Cartesian label from T_i. In the second step, these K training examples are combined into one weighted average position $\overrightarrow{e} = (d_x^e, d_y^e)$ as follows, ε is a small constant to avoid division by zero.

$$\overrightarrow{e} = \frac{\sum_{j=1}^{K} \frac{1}{Dist(\overrightarrow{RSS_j},\overrightarrow{RSS_i})+\varepsilon}\overrightarrow{L_j}}{\sum_{j=1}^{K} \frac{1}{Dist(\overrightarrow{RSS_j},\overrightarrow{RSS_i})+\varepsilon}} \ . \tag{4.3}$$

The nonconformity measure α_i is the difference between the Cartesian label $\overrightarrow{L_i}$ of T_i and the above weighted position $\overrightarrow{e}$.

$$\alpha_i = ||\overrightarrow{L_i} - \overrightarrow{e}|| = \sqrt{(d_x^i - d_x^e)^2 + (d_y^i - d_y^e)^2} \ . \tag{4.4}$$

The implementation of CP classification is summarised in Algorithm 1.

4.2.2 Regression conformal prediction fingerprinting

The challenge of implementing CP for regression is that it needs to examine every possible training label, which is not feasible for infinite real number labels. With ridge regression, however, there is a way to avoid having to examine the infinite number of labels, as previously discussed in [75, 85]. The idea of ridge regression is that one needs to find the weight value w for the following general regression model so that

$$\lambda||w||^2 + \sum_{i=1}^{M}(y_i - w\cdot x_i)^2 \rightarrow min \ . \tag{4.5}$$

where $\lambda > 0$ is the ridge factor constant, y_i is the label and x_i is the object of the training example i^{th} $(1 \le i \le M)$. To predict the label for the new sample x_{new}, ridge regression calculates $w \cdot x_{new}$. More details about ridge regression was previously discussed in Section 2.4.4.

Normally, the ridge regression procedure is presented in a matrix form. With fingerprint-

```
Data: Training database B = $\{T_1, \dots, T_M\}$, significance level $\xi$, new sample $T_{M+1}$
      with known RSS
Result: Prediction region $R$
/* Calculate the Euclidean distance of two RSS vectors                    */
Function Dist(P, Q)
    distance = 0;
    /* N is the total number of APs                                        */
    for $i = 1 \rightarrow N$ do
        $distance = distance + (AP_i^P - AP_i^Q)^2$;
    end
    return $\sqrt{distance}$;
/* Calculate the weighted average position                                 */
/* Kset contains $K$ nearest neighbours to $z$                             */
Function Weighted_location (Kset, z)
    for $i = 1 \rightarrow K$ do
        $weight1 = 1/(Dist(Kset.RSS_i, z.RSS) + \varepsilon) * Kset.L_i$;
        $weight2 = 1/(Dist(Kset.RSS_i, z.RSS) + \varepsilon)$;
    end
    return $weight1/weight2$;
Function Conform(B, z)
    /* Find K nearest neighbours to $z$, with different labels             */
    $Kset$ = {};
    for $i = 1 \rightarrow M$ do
        if $(T_i.L \neq z.L)$ and $min(Dist(T_i.RSS, z.RSS))$ then
            $Kset = Kset + \{T_i\}$;
        end
    end
    $e = Weighted\_location(Kset, z)$;
    return $\sqrt{(z.L - e)^2}$;
/* Y is the set of training positions                                      */
for $y \in Y$ do
    $T_{M+1} = (RSS_{M+1}, y)$;
    $B = B + \{T_{M+1}\}$;
    for $T_j$ in $B$ do
        $\alpha_j = Conform(B, T_j)$
    end
    $p(y) = \frac{\#\{j=1,\dots,M+1 : \alpha_j \geq \alpha_{M+1}\}}{M+1}$
end
Prediction set $R = \{y : p(y) > \xi\}$;
return R;
```

Algorithm 1: Fingerprinting Conformal Prediction with W-KNN.

ing, the training positioning labels are organised as a column vector $Y_M = (L_1, \ldots, L_M)'$, and the training WiFi RSS vectors are organised as a M by P matrix, where P is the total number of WiFi APs observed in the training data, $X_M = (\overrightarrow{RSS_1}, \ldots, \overrightarrow{RSS_M})'$. By substituting these matrices into Equation 4.5, the equation becomes:

$$\lambda ||w||^2 + ||Y_M - X_M w||^2 \to min \ . \tag{4.6}$$

which is equivalent to

$$Y_M' Y_M - 2w' X_M' Y_M + w'(X_M' X_M + \lambda I_P) w \to min \ . \tag{4.7}$$

with I_p is the identity matrix. By taking the derivative in w, the equation becomes:

$$2(X_M' X_M + \lambda I_P) w - 2X_M' Y_M = 0 \ . \tag{4.8}$$

which is equivalent to

$$w = (X_M' X_M + \lambda I_P)^{-1} X_M' Y_M \ . \tag{4.9}$$

For CP ridge regression, the nonconformity measure α_i of training example i^{th} can be defined as the difference between L_i and $\hat{L}_i$, where $\hat{L}_i$ is the prediction of ridge regression for $\overrightarrow{RSS_i}$ using the whole training data $(\overrightarrow{RSS_1}, \overrightarrow{L_1}), \ldots, (\overrightarrow{RSS_M}, \overrightarrow{L_M})$.

$$\alpha_i = |L_i - \hat{L}_i| \ . \tag{4.10}$$

The prediction $\hat{Y}_i$ for X_i are

$$X_M (X_M' X_M + \lambda I_P)^{-1} X_M' Y_M \ . \tag{4.11}$$

which is called a hat matrix H_M. The vector of the nonconformity measure $(\alpha_1, \ldots, \alpha_M)'$ can be re-written as follows [75, 85].

$$|Y_M - H_M Y_M| = |(I_M - H_M) Y_M| \ . \tag{4.12}$$

According to [75, 85], given the confidence level $(1 - \xi)$, with y is a potential label for x_{M+1} (the new sample), and $Y = (y_1, \ldots, y_M, y)'$, the vector of the nonconformity measure can be presented as $|A + By|$, where

$$A = (I_M - H_M)(y_1, \ldots, y_M, 0)' \ . \tag{4.13}$$

$$B = (I_M - H_M)(0,\dots,0,1)' \ . \tag{4.14}$$

Thus, the nonconformity measure α_i of each training example can be expressed as a piecewise-linear function of y [75, 85]. It was proven that the p-value p(y) can only change at points in which $\alpha_i(y) - \alpha_{M+1}(y)$ changes sign ($1 \leq i \leq M+1$). This means there is no need to compute the p-value of every possible training label, instead one only needs to compute the set of points on the real line that have a p-value greater than the significance level ξ. The following CP ridge regression was adapted from [75, 85] for the fingerprinting problem (see Algorithm 2).

The benefit of CP regression is that it produces a prediction region that may lie in between the training examples, whereas, CP classification picks a prediction set that contains the precise training examples. For each dimensional label, CP regression produces a prediction range (i.e. an interval). With a 2-dimensional tracking space, the prediction region is therefore a rectangle. If the application requires a single prediction, the simplest positioning output can be the centre of this rectangle. A slightly more computationally heavier approach is to search through all the training examples to find which ones are inside this rectangle prediction region, and takes the weighted average of these examples to obtain the final prediction (see Algorithm 3). The next section compares the performance accuracy of the algorithms described so far.

4.3 Evaluation of performance

This section compares the performance accuracy of the proposed CP classification and regression algorithms to other popular machine learning algorithms for fingerprinting in the literature such as Naïve Bayes and W-KNN. The advantage of using the confidence measure will be evaluated to assess whether it produces valid predictions, and if there is any improvement on the positioning result. The positioning error between the estimated position and the true position will be calculated using the Euclidean distance. The implementations of CP classification and CP regression in R and Java are also available[1].

4.3.1 The validity of CP evaluation

The so-called 'error rate' of CP is the percentage indicating how often the algorithm does not produce a prediction set (in the case of classification) or a prediction region (in the case of regression) that contains the true location. The algorithm is considered valid under a

[1] http://www.cs.rhul.ac.uk/~wruf265/CP/ - last accessed in Sep/2016.

```
Data: Training database B= $\{T_1,\dots,T_M\}$, significance level $\xi$, new example $T_{M+1}$
      with known RSS, $A$ and $B$ matrices are initialised from Eq. 4.15 and 4.16
Result: Prediction region $R$

for $i = 1 \to M$ do
    if B[i] < 0 then
        A[i] = -A[i]; B[i] = -B[i];
    end
end

P = { };
for $i = 1 \to M$ do
    if (B[i] $\neq$ B[M]) then
        P = P + -(A[i] - A[M])/(B[i] - B[M]) + -(A[i] + A[M])/(B[i] + B[M]);
    end
    if (B[i] == B[M]) and (A[i] $\neq$ A[n]) then
        P = P + -(A[i] + A[M])/2 B[i];
    end
end

/* sort P in ascending order from 1 to N.                          */
P[0] = $-\infty$; P[N+1] = $\infty$;
for $j = 0 \to N$ do
    N[j] = 0; M[j] = 0;
end
for $i = 1 \to N$ do
    for $j = 0 \to M$ do
        if (|A[i] + B[i] * P[j]| $\geq$ |A[N] + B[N] * P[j]|) and (|A[i] + B[i] * P[j+1]|
          $\geq$ |A[N] + B[N] * P[j+1]|) then
            N[j] = N[j] + 1;
        end
    end
end

for $i = 1 \to N$ do
    for $j = 0 \to M$ do
        if (|A[i] + B[i] P[j]| $\geq$ |A[N] + B[N] * P[j]|) then
            M[j] = M[j] + 1;
        end
    end
end

PRegion = { };
for $k = 1 \to K$ do
    PRegion = $(\cup_{j:N[j]/N>\xi}(y_j, y_{j+1})) \cup \{y_j : M[j]/N > \xi\}$;
end
return PRegion;
```

Algorithm 2: Fingerprinting Conformal Prediction with ridge regression [85].

```
Data: New sample $T_{M+1}$, predicted range produced by Algorithm 2:
      $x_{start}, x_{end}, y_{start}, y_{end}$
Result: Single prediction $T_w$
P = {};
/* go through all training examples                              */
for $i = 1 \rightarrow M$ do
    /* this training example is inside the predicted region      */
    if $(T[i].L[x] \geq x_{start})$ and $(T[i].L[x] \leq x_{end})$ and $(T[i].L[y] \geq y_{start})$ and
     $(T[i].L[y] \leq y_{end})$ then
        P = P + {T[i]};
    end
end
/* calculate the single weighted location for all examples in P.
   The Weighted_location function was defined in Algorithm 1     */
$T_w = Weighted\_location(P, T_{M+1})$;
return $T_w$;
```

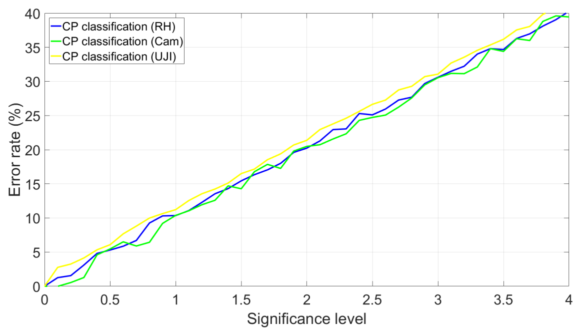
Algorithm 3: Computing single prediction with CP ridge regression.

confidence level $(1-\xi)$, if the error rate does not exceed ξ. The formulae to computes this error rate for a given confidence level is defined below, where N is the total number of tests, L_i is the Cartesian label and R_i is the prediction set or region of the test sample i^{th} $(1 \leq i \leq N)$.

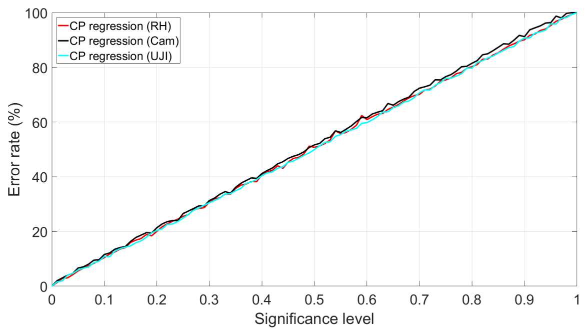
$$Error\_rate = \frac{\#|i = 1,\ldots,N : L_i \notin R_i|}{N} \tag{4.15}$$

To study the validity of the predictions, the first experiment performs 10-fold cross validation on each training database of the three test beds. This is the special case of leave-one-out validation, where each training example is left out and is used as the test sample. However, due to the high number of training examples in all three datasets, it is preferred to use 10-fold cross validation. The training examples are randomly divided into *10* roughly equal portions, for which *9* of them will be used as the training set and the single remaining portion will be used for testing. In particular, each fold has about *1,360* examples for the RH test bed, *7,800* examples for the Cam test bed and *1,990* examples for the UJI test bed. This process is repeated *9* times for the other portions, so that every training example has a chance to be considered as test sample. The error across all *10* trials will be averaged to obtain the error rate.

Figures 4.1 demonstrates that both CP classification and CP regression are valid on all three datasets. The error rates were around the specified confidence level in all cross-validations (subject to statistical fluctuation). When the significance level is zero, which is equivalent to *100%* confidence level, there is no error since the predictions are the whole training set. With a significance level of *1*, which is equivalent to *0%* confidence level, the error rate is *100%* since the prediction set is empty. With CP classification, the test did not go below *60%* confidence level, because many test samples started to return empty prediction sets at this level. This result also indicates that this is the threshold for the fingerprinting datasets used in this thesis under CP classification.



(a) The validity of CP classification.



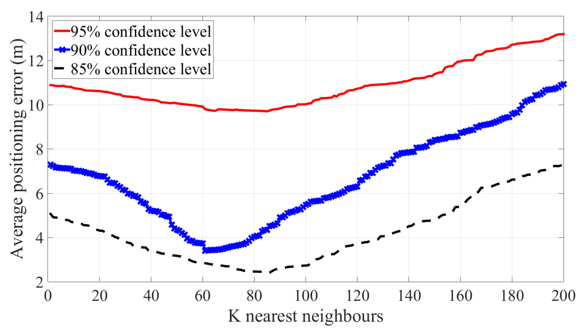
(b) The validity of CP regression.

Fig. 4.1 The validity of CP under three test beds.

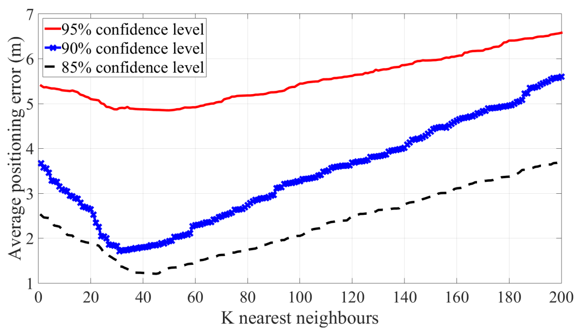
This experiment has demonstrated the validity of the predictions provided by CP, under different significance (or confidence) levels. However, it is more interesting to the fingerprinting researchers that the prediction set (or the prediction region) is small. Thus, the next experiment evaluates the narrowness of the prediction sets and regions, with respect to the positioning accuracy and the confidence level. The same training sets from the above 10-fold cross validation were used. For each confidence level, the positioning prediction accuracy for all test samples were calculated and then averaged.

4.3.2 The performance of CP classification evaluation

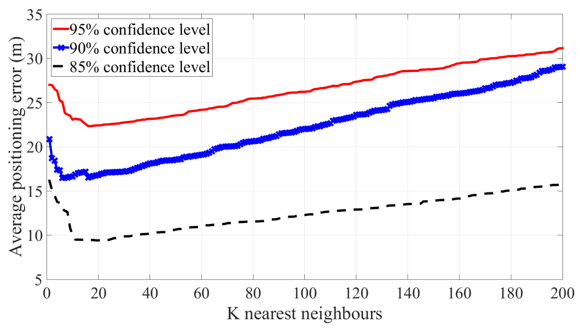
For CP classification, there are two parameters to control, which are the *K* neighbours and the confidence level. The value of *K* decides how many training examples to be considered to calculate an average position. A big *K* includes the examples that are too far away in terms of the Euclidean distance. A small *K* (such as 1-nearest neighbour) may not include the correct prediction, because the training database is noisy. For CP classification, the value of *K* for optimal positioning prediction was empirically shown to be around *60* to *80* for the RH test bed, *30* to *40* for the Cam test bed, and *10* to *20* for the UJI test bed (see Figure 4.2). Depending on the chosen confidence level, the value of *K* slightly varies. For the ease of comparison, a fixed *K* of *70*, *35* and *15* were selected for the RH, Cam and UJI test bed respectively.



(a) RH test bed.



(b) Cam test bed.



(c) UJI test bed.

Fig. 4.2 Performance accuracy of CP classification under different nearest neighbours.

Using the optimal values of *K* found above, the next experiment evaluates the performance of CP classification under different confidence levels to understand the size of the prediction set and the performance accuracy. Since CP classification produces a set of predictions, there are two methods to obtain a single prediction to compare with the true position. The first method is averaging the whole prediction set to a single position. The second method is simply using the training position with the largest p-value. Their performances will be evaluated below.

Tables 4.1, 4.2 and 4.3 demonstrate that the size of the prediction set strictly decreases as the confidence level decreases for all three test beds, as expected. At *100%* confidence, the full training set of all *9* folds was returned. At *60%* confidence levels for all three test beds, several test samples started to return empty prediction sets, which indicated that this seems to be the threshold. Overall, the recommended confidence levels for CP classification are *75%* for the RH test bed, *80%* for the Cam test bed and *65%* for the UJI test bed, which achieve *2.42* metre error, *0.7* metre error and *9.3* metre error respectively by averaging the prediction set.

Table 4.1 Performance of CP classification for the RH test bed.

Confidence level	**Significance ξ**	**Positioning error (avg)**	**Predicted positions (avg)**	**Error rate (avg)**
100%	0	18.2 m	62	0%
95%	0.05	6.7 m	46	4.8%
90%	0.1	3.8 m	31	10.3%
85%	0.15	2.44 m	19	11.2%
80%	0.2	2.5 m	14	19.6%
75%	0.25	2.42 m	9	25.3%
70%	0.3	2.58 m	9	29.7%
65%	0.35	2.51 m	8	34.8%
60%	0.4	2.53 m	4	39.6%

Table 4.2 Performance of CP classification for the Cam test bed.

Confidence level	**Significance ξ**	**Positioning error (avg)**	**Predicted positions (avg)**	**Error rate (avg)**
100%	0	7.6 m	1,755	0%
95%	0.05	4.9 m	1,312	4.6%
90%	0.1	1.8 m	1,106	9.2%
85%	0.15	1.2 m	879	14.7%
80%	0.2	0.7 m	513	19.8%
75%	0.25	0.75 m	307	24.3%
70%	0.3	0.82 m	156	29.5%
65%	0.35	0.88 m	79	34.8%
60%	0.4	0.83 m	38	39.6%

When the training position with the largest p-value was used as the predicted position, the positioning error was smaller than that when the whole prediction set was used, for

Table 4.3 Performance of CP classification for the UJI test bed.

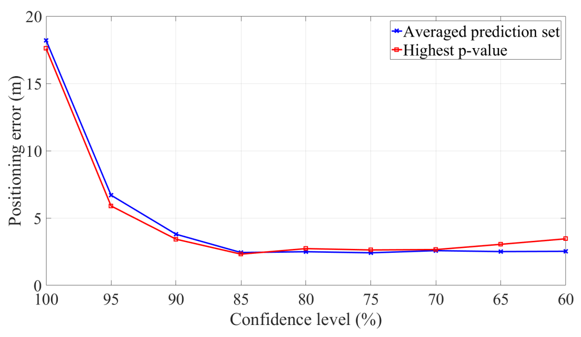
Confidence level	**Significance ξ**	**Positioning error (avg)**	**Predicted positions (avg)**	**Error rate (avg)**
100%	0	37.4 m	837	0%
95%	0.05	22.3 m	592	5.6%
90%	0.1	16.5 m	522	10.8%
85%	0.15	9.8 m	469	15.8%
80%	0.2	10.2 m	317	21.2%
75%	0.25	10.1 m	143	26.7%
70%	0.3	9.7 m	82	31.3%
65%	0.35	9.3 m	29	36.4%
60%	0.4	10.6 m	16	41.5%

the confidence levels within *85%* and *100%*. However, as the confidence level decreases from *85%* downwards, the positioning error got larger with the largest p-value prediction, for all three test beds (see Figure 4.3). This is because when the confidence level is high, the prediction set contains many predictions including those that are not close to the true position. Hence, the overall positioning error when using the whole training set was higher. However, the prediction with the largest p-value is not always the correct prediction, due to the noisiness of the training database. Thus, its positioning error slightly suffers when the size of the prediction set decreases.

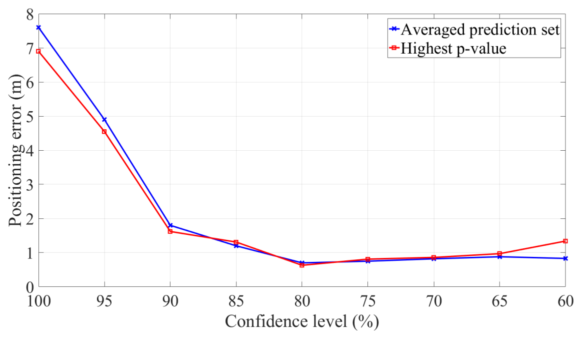
4.3.3 The performance of CP regression evaluation

With CP regression, the two parameters to calibrate are the ridge factor λ and the confidence level. Unlike the K neighbours which is an integer parameter, the ridge factor is a real number, thus, it is more challenging to calibrate. Earlier discussions in Sections 2.4.4 and 4.4.4 pointed out that it is preferred to have a balance λ to fit a linear model, and shrink the co-efficient at the same time. Figure 4.4 uses a step length of *0.1* to test the ridge factors from *0.01* to *10*, which suggested that the optimal ridge factor was around *0.1* to *1* for all three test beds. A fixed λ of *0.7*, *0.6* and *0.3* was chosen for the RH, Cam and UJI test bed respectively.

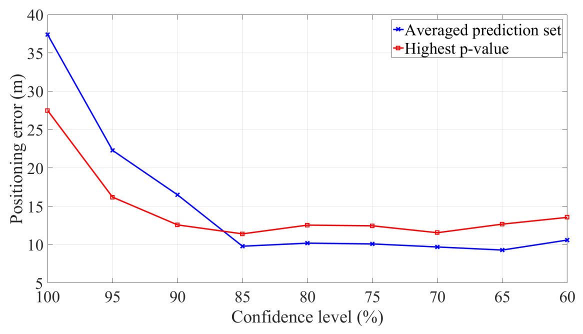
Using these recommended ridge factors, the next experiment evaluates the performance



(a) RH test bed.



(b) Cam test bed.



(c) UJI test bed.

Fig. 4.3 Comparison of the performance accuracy of the single prediction with CP classification, under the averaged prediction set and largest p-value prediction.

of CP regression under different confidence levels. Different from CP classification which predicts a set of individual positions, CP regression produces a region. For fingerprinting with 2-dimensional Cartesian label, this region is a rectangle, therefore, this experiment reports the width and the height of the region. Algorithm 3 was used to deduce a single positioning prediction from the region. The positioning error is the difference between the true position and this single prediction. Tables 4.4, 4.5 and 4.6 demonstrate that when the confidence level decreases, the narrowness of the prediction region increases, as expected. At 100% confidence level, the prediction region was $[-\infty\ +\infty]$ which can be interpreted as the whole tracking zone. The experiment stopped at 40% confidence level since the region got too small, and the prediction region of many test samples did not include the true position.

Overall, the recommended confidence level with CP regression is *60%* for the RH test bed, *65%* for the Cam test bed, and *50%* for the UJI test bed to achieve the most accurate positioning prediction under these three test beds.

Table 4.4 Performance of CP regression for the RH test bed.

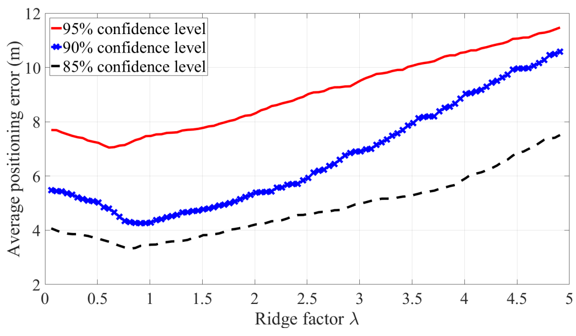
Confidence level	**Significance** ξ	**Positioning error (avg)**	**Width (avg)**	**Height (avg)**	**Error rate (avg)**
95%	0.05	7.13 m	9.2 m	13.8 m	4.2%
90%	0.1	4.24 m	7.2 m	11.3 m	9.4%
85%	0.15	3.71 m	6.3 m	7.4 m	14.1%
80%	0.2	3.23 m	6.1 m	6.9 m	18.4%
75%	0.25	2.72 m	5.8 m	6.6 m	24.5%
70%	0.3	2.67 m	5.2 m	6.2 m	28.7%
65%	0.35	2.66 m	4.7 m	5.6 m	33.6%
<u>60%</u>	0.4	<u>2.65 m</u>	4.6 m	5.3 m	38.2%
55%	0.45	2.67 m	3.9 m	4.8 m	43.1%
50%	0.5	2.71 m	3.2 m	4.2 m	51.2%
45%	0.55	2.68 m	2.6 m	3.7 m	56.8%
40%	0.6	2.69 m	1.9 m	3.4 m	62.4%

Table 4.5 Performance of CP regression for the Cam test bed.

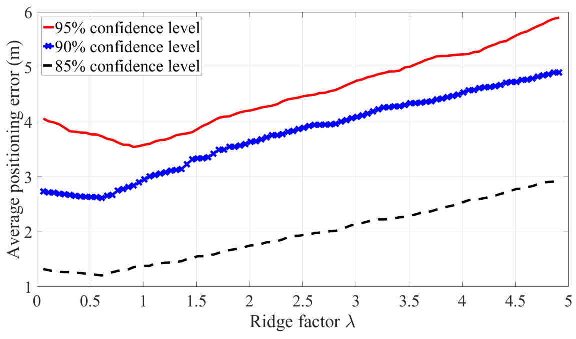
Confidence level	**Significance ξ**	**Positioning error (avg)**	**Width (avg)**	**Height (avg)**	**Error rate (avg)**
95%	0.05	3.52 m	2.75 m	1.63 m	4.7%
90%	0.1	2.63 m	2.64 m	1.51 m	9.7%
85%	0.15	1.19 m	2.38 m	1.47 m	14.5%
80%	0.2	1.08 m	2.24 m	1.41 m	19.4%
75%	0.25	0.97 m	2.21 m	1.32 m	24%
70%	0.3	0.83 m	2.17 m	1.26 m	29.3%
65%	0.35	0.63 m	1.91 m	1.22 m	34%
60%	0.4	0.65 m	1.69 m	1.13 m	39.4%
55%	0.45	0.74 m	1.52 m	1 m	45.5%
50%	0.5	0.65 m	1.41 m	0.97 m	50.7%
45%	0.55	0.82 m	1.25 m	0.93 m	56.8%
40%	0.6	0.84 m	1.16 m	0.88 m	61.7%

Table 4.6 Performance of CP regression for the UJI test bed.

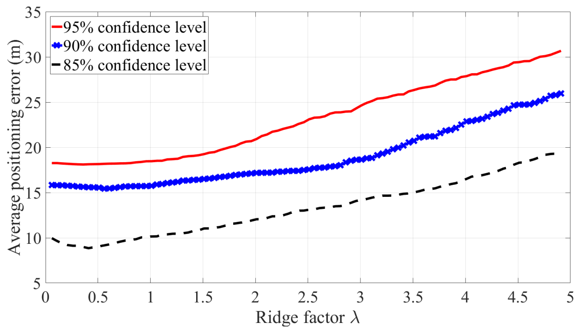
Confidence level	**Significance ξ**	**Positioning error (avg)**	**Width (avg)**	**Height (avg)**	**Error rate (avg)**
95%	0.05	17.82 m	21.6 m	18.3 m	5.1%
90%	0.1	15.2 m	18.7 m	17.4 m	10.7%
85%	0.15	9.2 m	16.9 m	15.4 m	15.9%
80%	0.2	9.5 m	14.1 m	12.4 m	21.8%
75%	0.25	8.78 m	13.4 m	10.7 m	26.2%
70%	0.3	8.62 m	12.6 m	9.3 m	31.8%
65%	0.35	9.3 m	10.3 m	8.8 m	36.7%
60%	0.4	9.38 m	9.8 m	7.9 m	41.6%
55%	0.45	8.86 m	8.1 m	7.6 m	46.4%
<u>50%</u>	0.5	<u>8.71 m</u>	7.5 m	6.8 m	51.8%
45%	0.55	9.28 m	6.6 m	6.1 m	56.5%
40%	0.6	8.96 m	6.3 m	5.7 m	61.3%



(a) RH test bed.



(b) Cam test bed.



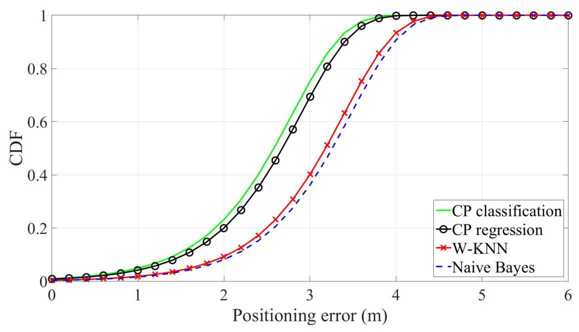
(c) UJI test bed.

Fig. 4.4 Performance accuracy of CP regression at different ridge factors.

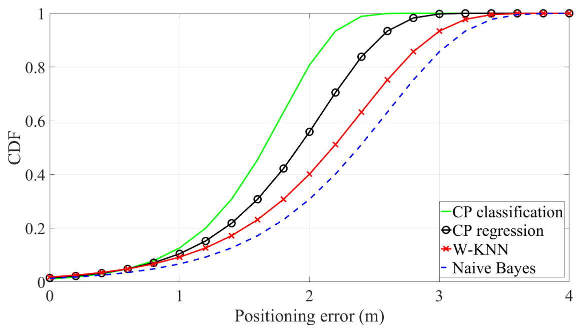
4.3.4 Overall prediction accuracy evaluation

To understand how the proposed CP algorithms perform in the real world, they will be verified with the test sets which have been independently collected and may contain samples that were not seen in the training sets. This experiment uses the most optimal parameters found from the cross-validation experiments in the earlier sections. In doing so, it compares the performance of confidence machine against other well-known traditional approaches in fingerprinting including Naïve Bayes and W-KNN. The test set of the UJI test bed was the public version collected *3* months after the training set. There is another competition version that was generated *18* months later, which will be evaluated in the next section.

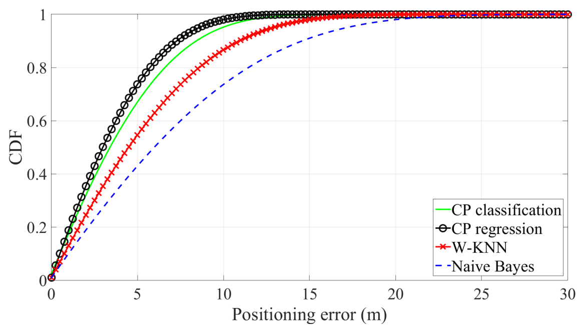
Figure 4.5 uses the Cumulative Distribution Function (CDF) to demonstrate the superiority of the proposed confidence learning algorithms, which outperform the traditional fingerprinting algorithms by *15%* to *20%* on average in all three independent test sets. In particular, compared to W-KNN, CP reduced the positioning error from *4.2* metre error to under *3.6* metre error, *95%* probability, for the RH test set. For the Cam test bed, the positioning error was reduced from *3.2* metres to *2.2* metres, *95%* probability. For the UJI test bed, the improvement was more significant, from *17* metres to under *7* metres, *95%*



(a) RH test bed.



(b) Cam test bed.



(c) UJI test bed.

Fig. 4.5 Performance accuracy of confidence machine learning.

probability. It was noted that CP classification performed slightly better than CP regression in the RH and Cam test sets. However, CP regression edged out in the UJI competition test set, due to its sparser training set. W-KNN performed better than Naïve Bayes in all three test sets, which suffered heavily with so few measures per location in the UJI training set.

4.3.5 UJI's floors and buildings hit rate evaluation

Out of the three test beds, the UJI test bed has multiple floors and buildings. Although the proposed algorithms in this chapter were designed with 2-dimensional space in mind, it would be interesting to assess how well they identify the user position in 3-D. Since CP classification returns a set of predictions containing the precise training positions, a simple majority will decide which floor and building for the whole prediction set. With CP regression, a prediction interval of the floor is first obtained, and the centre of this interval will indicate which floor the user is on. Then, the prediction region of the longitude and latitude in a rectangle shape is obtained. Finally, a majority of all training positions within this estimated rectangle will decide the building of the user. It is possible to use a regression model to directly estimate the interval of the building. However, this is un-necessary because

Table 4.7 Building and floor's successful hit rate for the UJI public test set.

	Building hit rate	**Floor hit rate**
CP classification	99.4%	78.2%
CP regression	99.6%	81.5%

knowing the longitude and latitude will also reveal the building number.

Out of *1,111* test samples, Table 4.7 demonstrates a very high *99.6%* successful hit rate for the building prediction, and a bit lower *81.5%* hit rate for the floor prediction with CP regression. Out of *18.5%* test samples that were estimated on the wrong floors, most of them were only *1* floor below or above the correct one (see Figure 4.6).

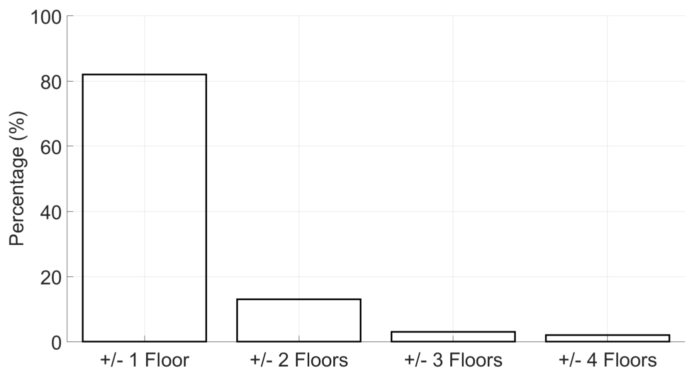


Fig. 4.6 Performance of CP regression with the UJI public test set.

4.3.6 EvAAL competition test set evaluation

With the UJI test bed, the training set is un-changed. However, there are two versions of the test set. The first one is publicly available to anyone to self-evaluate their systems, which has been used in the previous sections. The second one is un-labelled so that it can be used for the EvAAL 2015 competition. This test set may be obtained by contacting the test bed's author directly. Its performance is presented here.

In the EvAAL competition, each entrant was allowed *5* submissions, and the most accurate one was chosen as the final result. The author decided to enter *3* CP regression entries and *2* CP classification entries with $K = \{10, 15\}$ and $\lambda = \{0.3, 0.5, 0.7\}$ which are around the optimal parameters suggested by the earlier experiments. CP regression was given more entries because the above experiments with the UJI public test set suggested that it performed a bit better than CP classification. It is worth noting that although the proposed algorithms were not officially entered in the competition, the submissions were evaluated using the same criteria as with other competitors. In particular, the positioning error was calculated as follows, where R is the true positioning label, and E is the estimated positioning label. For every wrongly predicted building, a *50* metre penalty is added into the result, and for every wrongly predicted floor, a *4* metre penalty is added.

$$Error(R,E) = Euclidean\_Distance(R,E) + (50 * buildingfail) + (4 * floorfail) \quad (4.16)$$

The final results put CP regression as 2nd with *9.1* metre error, *75%* probability, and CP classification as 3rd with *10.4* metre error, *75%* probability, out of the *5* competitors (see Figure 4.7). The detailed performance of CP classification and CP regression under the competition test set is also available[2].

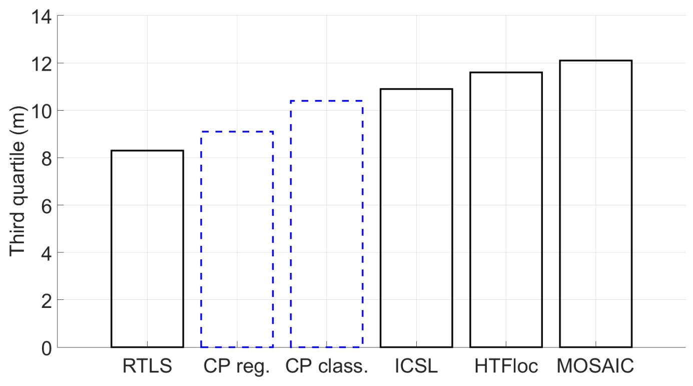


Fig. 4.7 Performance of CP regression with the EvAAL competition test set.

The information of other competitors can be found on the competition website[3]. It is worth noting that some of them came from the industry and did not reveal much of their underlying algorithms. The RTLS system that came first in this competition also used W-KNN. However, it applies several filtering methods to the training database, and then divides it into smaller training sets. In other words, it was designed to tackle this dataset specifically. The proposed CP classification and CP regression in this chapter were tested with the full complete training database. In addition, the proposed confidence learning approach in this chapter offers more information (i.e. the confidence level) which was not utilised at all since the competitors were ranked solely on the positioning error. More importantly, the proposed algorithms can produce a prediction set or a prediction region, which was forced to reduce into a single prediction for evaluation.

[2]http://www.cs.rhul.ac.uk/~wruf265/CP/ - last accessed in Sep/2016.

[3]http://evaal.aaloa.org/2015/competition-home - last accessed in Sep/2016.

4.3.7 Summary of the experimental results

This section summarises the results obtained from the previous experiments to deduce what lessons to learn from, in order to apply the proposed ideas into the real world.

The positioning predictions from CP have been demonstrated to be valid under different confidence levels. They constantly outperformed traditional Naïve Bayes and W-KNN by *15%* to *20%* more accurate on average, under the same training and test domains. By comparing the performance of CP classification and CP regression under the same confidence level, with optimal K nearest neighbours and ridge factor λ, the experiments suggested that CP regression achieved a noticeably smaller positioning error for the UJI test bed where the training space is huge and the training locations are sparser than the other two test beds. CP classification achieved a slightly lower positioning error with the RH and Cam test beds, where the training locations are denser.

It was understood that the values of the K parameter, the ridge factor and the confidence level did impact the quality of the final prediction. It is challenging to find the optimal values that work for all new test samples, which are not known in advance in the real world. A possible approach to estimate the optimal value for these parameters is to perform cross-validation with the training database under different parameters, as seen in this chapter.

4.4 Summary and further work

This chapter has proposed a novel confidence learning approach for the fingerprinting problem using Conformal Prediction. Two confidence-based algorithms for classification and regression have been discussed. Their positioning accuracies have been empirically shown to be around *2.4* metres, *75%* probability under a normal test bed, around *70* cm, *75%* probability under an ideal test bed, and around *8.8* metres, *75%* probability under a challenging test bed. These results outperformed non-confidence machine learning algorithms tested on the same test beds by up to *20%* more accurate. The prediction sets produced by the proposed algorithms have also been demonstrated to be valid. The proposed confidence learning algorithms were compared to other machine learning algorithms in the EvAAL 2015 competition under the same testing criteria. The result ranked the two proposed algorithms as 2^{nd} and 3^{rd} out of the *5* competitors.

Some potential further work is to examine other nonconformity measures for CP classification and CP regression. Since CP regression estimates each label separately, it is desirable to find new nonconformity measures that combine multiple labels together in one single prediction.

In summary, this chapter has demonstrated how to estimate the user position using the fingerprinting database. The next chapter will present a novel method to predict the user's walking route and the potential destination in advance.

Chapter 5

Dynamic route prediction with the magnetic field data

"If you have to forecast, forecast often."
– Edgar Fiedler, The three Rs of economic forecasting (1977).

Most existing fingerprint-based systems were designed as an on demand service to guide the user to his wanted destination. This chapter introduces a novel feature that allows the positioning system to predict in advance which walking route the user may use, and the potential destination. To achieve this goal, a new so-called routine database will be used to maintain the magnetic field strength in the form of the training sequences to represent the walking trajectories. The benefit of the system is that it does not adhere to a certain predicted trajectory. Instead, the system dynamically adjusts the prediction as more data are exposed through-out the user's journey.

The proposed system was tested in a real indoor environment to demonstrate that the system did not only successfully estimate the route and the destination, but also improved the single positioning prediction.

5.1 Passive monitoring for the regular users

The previous chapter demonstrates fingerprinting in an active tracking context, where the user submits a new WiFi RSS sample for the system to discover his current position. This chapter considers fingerprinting in a passive tracking scenario, where the system has the permission to monitor the user positions continuously to react in a timely manner when his position changes (i.e. to open the door, to switch off the light). Crucially, with passive monitoring, the system is guaranteed to have access to a sequence of the signal data, which provides useful insights into the walking trajectory. In addition, these users normally have established presence in the building, hence, their personal routines should have been well-observed to help the system predict their intended travelling path in advance.

However, the use of the WiFi RSS in this scenario is challenging because of its noisiness and its high variation. This chapter will investigate the magnetic field data as a new source of information, which has been suggested in some earlier work in fingerprinting [33, 45, 80]. There is a fundamental difference in the application of the magnetic field data in this chapter, that is, they are considered as a sequence. The chapter will later explain why this is a more suitable option than using it in a traditional single position fingerprint. This chapter will present a novel dataset called the ‘routine database’ to reflect the user’s walking routine, based on the magnetic field data. The benefit of this approach, and the details of the implementation will be discussed in the remaining sections of this chapter.

5.2 Extra information from the magnetic field

This section explores the use of the indoor magnetic field for fingerprinting. It only examines the features that are most relevant to fingerprinting, including the sensitivity and the uniqueness. More detailed experiments of the indoor magnetic field data can be found in [45, 54, 93].

5.2.1 How to measure the magnetic field?

An Android phone - one of the most popular devices to perform fingerprinting, was used to collect the magnetic field for all experiments in this chapter. The geomagnetic field sensor on the Android phone measures the strength and the direction of the magnetic field at a point in space, where the phone is held. This measure is presented as a 3-dimensional vector $\vec{m} = (x,y,z)$, where x,y,z (measured in microTesla or μT) are the ambient strength of the magnetic field in the corresponding axis (see Figure 5.1). The fastest sampling frequency on the experimented Nexus 5 phone was about *3* samples per second.

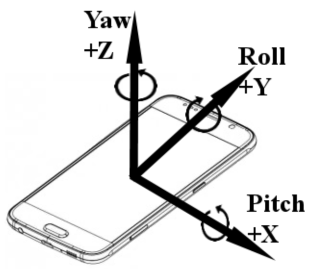


Fig. 5.1 The three axes of the geomagnetic field sensor.

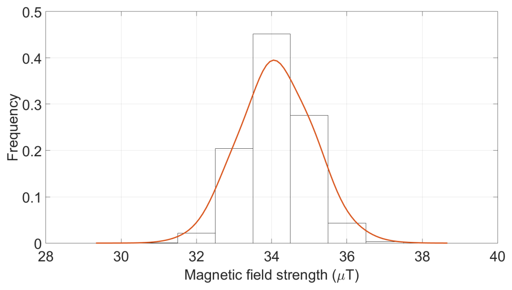
It is important to note that these axes are relative to the position of the mobile phone in space. Therefore, the magnitude of each axis may differ as the device's orientation changes, even when it stays in the same spot. The accelerometer included on the phone may be used to detect the direction of the gravity, which helps to deduct the 3-D position of the phone. However, since the focus of this chapter is mainly about the magnetic field, this idea is left out for future work. A simple solution to this problem is to combine the magnetic field vector into one scalar magnitude as follows (see Equation 5.1).

$$||\vec{m}|| = \sqrt{x^2+y^2+z^2}\ (\mu T)\ . \tag{5.1}$$

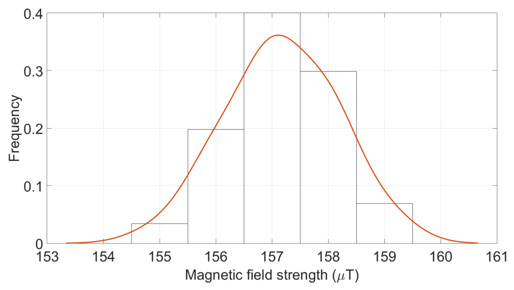
However, with this approach, there is only measure left for each position. For the rest of this chapter, this single scalar magnitude will be referred to as the magnetic field strength (MFS), and the whole magnetic field vector with all three magnitudes of the three axes will be referred to as the MFV.

5.2.2 The sensitivity of the magnetic field strength

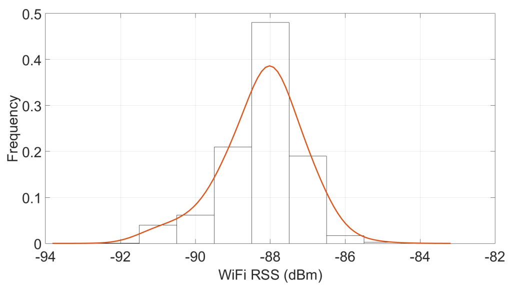
The reliability of fingerprinting depends on the stability of the fingerprints. This is the ability to repeat the same measure at the same position at any time in the near future. This requirement is challenging, because the structure of the building may have changed, or because of other external factors that happened at the time of data measuring (e.g. moving people or furniture). The former may happen gradually which allows the system to cope with, although it may require the whole training data to be re-surveyed. The latter is normally caused by the indoor users' movements, which may be avoided if the MFS has low sensitivity. Two experiments were performed to assess the sensitivity of the MFS in real-time and over long periods of time.



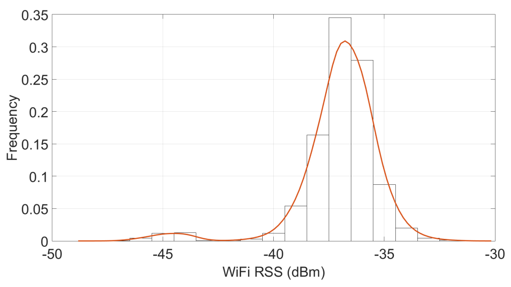
(a) Weak magnetic field strength from a central heater measured at a fixed position. It varied from 31 to 37μT (7 μT variation). The normalised range is [0.84 1].



(b) Strong magnetic field strength measured from the same central heather at a different fixed spot. It varied less between 155 to 159μT (5 μT variation). The normalised range is [0.97 1].



(c) Weak WiFi RSS from a single AP varied from -85 to -93dBm (9 dBm variation). The normalised range is [0.91 1].

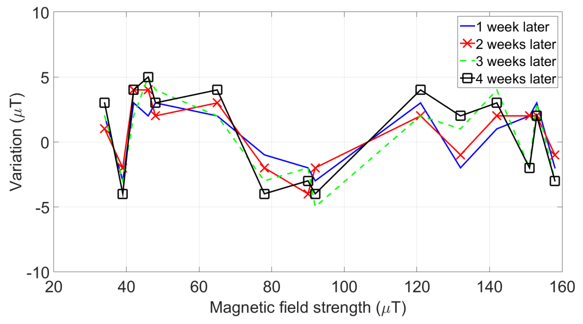


(d) Strong WiFi RSS from the same AP varied much more from -30 to -48dBm (19 dBm variation). The normalised range is [0.62 1].

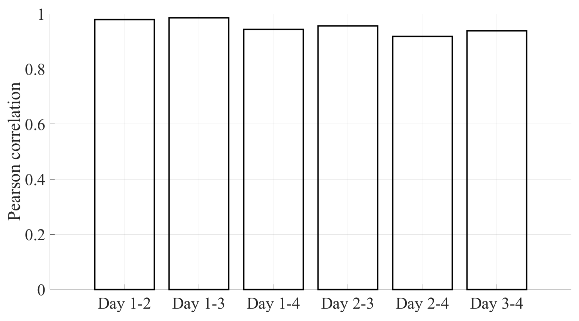
Fig. 5.2 Histograms of the WiFi RSS and MFS in fixed positions from the same magnetic source and AP, over *6* hours during working hours. Stronger MFS were more robust than weak MFS. Strong WiFi RSS were less robust than weak WiFi RSS.

The first experiment measures the variation of the MFS at a fixed position. It is preferred that the variation is low to avoid the second challenge posed above. The experiment was performed over *6* hours during the working hours to reflect a busy environment, with many people walking in the building. Compared to the WiFi RSS, the MFS vary much less rapidly in its measurement unit in fixed positions. This experiment also suggested that the stronger the field strength was (the closer it is to the magnetic source), the more stable the MFS was, which was exactly the opposite of the WiFi RSS, where strong RSS was more sensitive than weak one (see Figure 5.2).

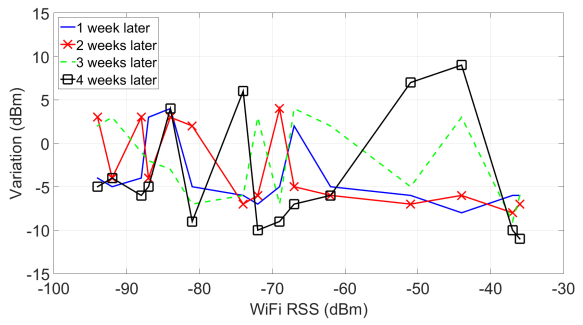
The second experiment assesses the repeatability of the MFS at the same positions. For this experiment, *15* positions in the building were marked with duck-tape. They were re-



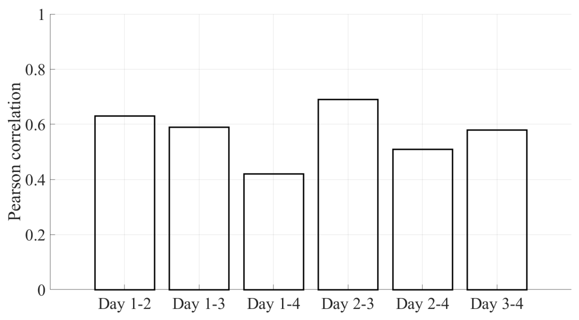
(a) MFS change over time.



(b) Pair-wise comparison of MFS sequences with Pearson correlation. The closer it is to 1, the more similar the two sequences are.



(c) WiFi RSS change over time.



(d) Pair-wise comparison of WiFi RSS sequences with Pearson correlation. The closer it is to 1, the more similar the two sequences are.

Fig. 5.3 The change of the MFS and WiFi RSS at *15* observation points over *1* month. Strong MFS was slightly less sensitive, while strong WiFi RSS was more sensitive. MFS sequences were highly correlated, while WiFi RSS ones were not.

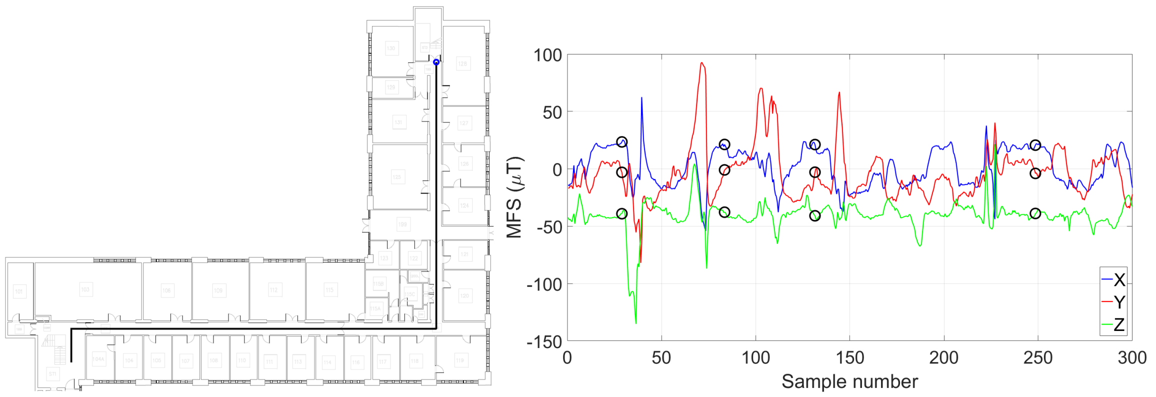
visited weekly, over *1* month to measure the MFS. Only *5* measures were taken at each observation point, and the average was used for this experiment. It was suggested that the variation was minimal, compared to the WiFi RSS observed at the same locations, with up to only $3\ \mu T$ for the positions with strong MFS, and up to $4\ \mu T$ for other positions with weak MFS (see Figure 5.3). Such difference was well-within the short-term variation range of the magnetic field reported in the last experiment. Also, there may be a slight displacement of the phone's position when the experiment took place, which explains this small change.

In summary, the magnetic field has low sensitivity and is highly repeatable with time, compared to the highly volatile WiFi RSS. The major difference of the WiFi RSS and MFS is that strong MFS was observed to be more robust than weak one, whereas, strong WiFi RSS was more sensitive than weak RSS. This is an important suggestion to apply the magnetic field to fingerprinting, since most positioning systems would favour the strong signals.

5.2.3 The uniqueness of the magnetic fingerprints

The positioning accuracy of fingerprinting relies on how unique the surveyed fingerprints are. If all the training examples are similar, it will be challenging for any algorithm to use them to estimate the user position. With the magnetic field, the building is made of large ferrous metal structures (i.e. pipes, steel shells) which greatly distort the magnetic readings at many indoor positions (see Figure 5.5). However, their influence is mostly local, and the greater the anomalies, the more unique the magnetic fingerprint is.

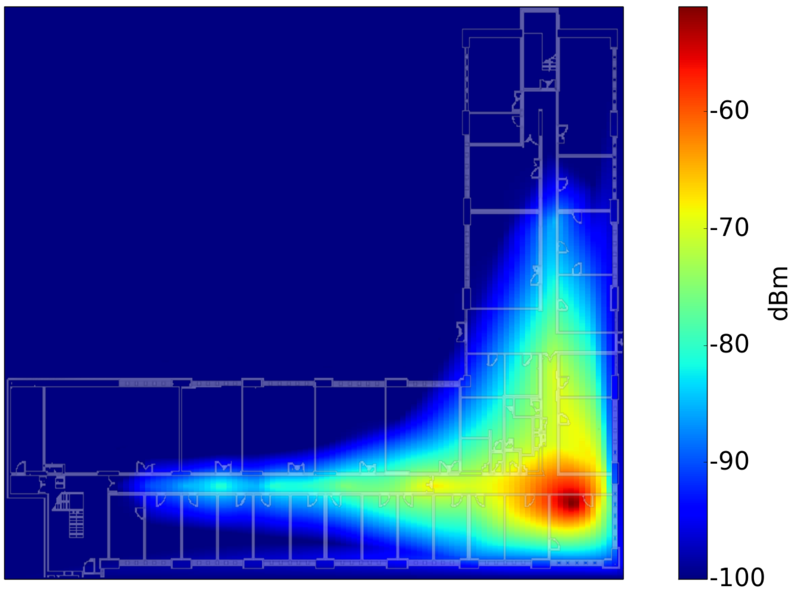
The major challenge when using the magnetic data for fingerprinting is that it only contributes a maximum of three measurements corresponding to the magnitude of the three axes for each fingerprint, whereas, the WiFi signal provides a big vector of many RSS readings from several nearby WiFi APs. A single walk through the ground floor of the RH test bed exposed several different positions with a similar MFV, in which the difference in the magnitude of each axis was just $2\ \mu T$ (see Figure 5.4).



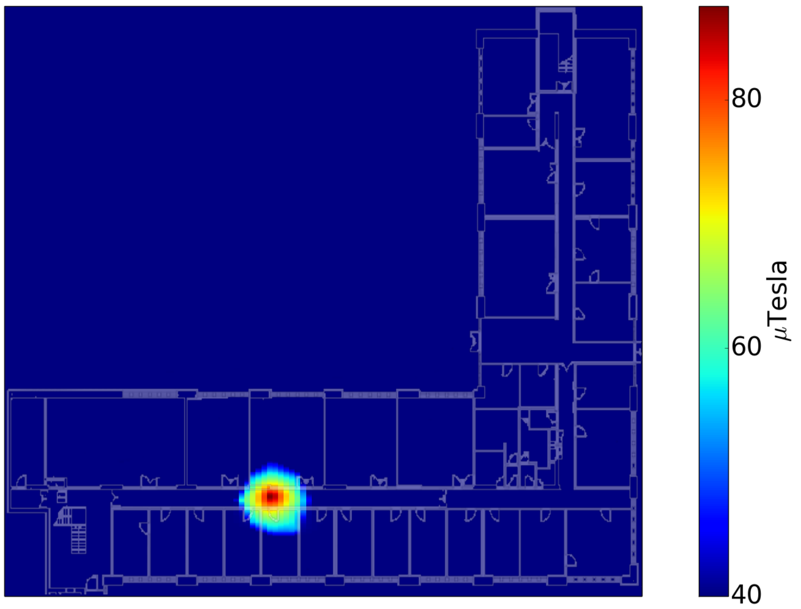
(a) The walking trajectory. (b) The magnetic readings.

Fig. 5.4 The magnitude of each of the three axes measured from a single walk through non-repeated positions on the ground floor of the RH test bed. The black circle pinpoints the positions with a similar reading.

For this reason, the magnetic field alone should not be employed to represent a single position, as in the case with the WiFi RSS. To take advantage of the robust magnetic measure, however, this chapter proposes to merge them into a long sequence to represent a walking trajectory, which is much more distinctive. Intuitively, it is less likely to have two different trajectories with the same magnetic field representation. These trajectories form the user's routine database to be explored in the next section.



(a) The propagation of WiFi RSS from a single AP in an office. The non-uniform signal coverage was caused by signal passing through doors and walls. Positions further down the corridor may still see the AP, although the signal was relatively weak.



(b) Magnetic field strength from a central heater attached on the wall in the same indoor corridor. It provides very fine scale reading over short distance.

Fig. 5.5 Magnetic field strength is more resilient than WiFi measurement from its transmitting shape. Distance-wise, WiFi coverage is far greater than magnetic field, from a single power source.

5.3 The personal routine database

In addition to the fingerprinting database, a routine database is introduced for each user. This new database describes the user's travelling history in the form of the frequently visited places, and the routes amongst them. Since each user normally has different preferred trajectories, he should have his own routine database. Each training example of the routine database represents a walking trajectory, whereas each training example in the fingerprinting database represents a single position. The training trajectory is a sequence of continuous MFV readings, mapped to their Cartesian co-ordinates. It is essential to clarify that the main purpose of this routine database is not to substitute the fingerprinting counterpart, but to provide prior knowledge to predict where the user may go next. The fingerprinting database will still be needed when the user takes a completely new route, which has not been recorded in the routine database. This routine database was introduced based on three observations.

(i) The indoor user often takes the same route to travel between familiar places.

(ii) The user often walks in a straight path.

(iii) For a moving user, there is often not enough time for the tracking system to collect multiples readings at every single position in real-time.

This section outlines the process of generating the routine database, and the formal model of such database.

5.3.1 The off-line phase to generate the routine database

Initially, the routine database is generated manually in the same manner as the fingerprinting database. The two differences are that the magnetic field is collected instead of the WiFi RSS, and each training example in the routine database is a sequence of MFV. The expert first identifies the route he wants to cover. He then uses an Android phone (with the app described in Chapter 2) to record the magnetic data at different positions along the route, and provides them with the corresponding Cartesian labels. At each position, the magnetic data should also be measured several times and an averaged measure is computed, although the number of repeated measures need not be as high as in the case with the WiFi RSS, because of the low variation of the magnetic field as previously discussed in Section 5.2.2. The starting and ending positions of the trajectory are also labelled by the room or office number, so that the system can provide a human-readable response when needed. The only criterion the expert should consider at this phase, is how long the gap between the two consecutive positions in the trajectory is. Ideally, it is preferred that this gap is small. The granularity of the fingerprinting database can be used as a reference to decide this gap (e.g. *1* metre between consecutive positions). A method to compare two trajectories with different gaps will be discussed in the next section. The expert will attempt to cover as many trajectories as possible, which is similar in the sense that he also tries to cover as many individual training positions for the fingerprinting database as possible.

It may also be useful to incorporate the user's personal timetable into his routine database, by looking for the starting and ending positions of the events on the calendar. However, such approach is beyond the scope of this chapter. For the experiments described here, the routine database is manually generated by one user to reflect his personal routine.

5.3.2 Modelling the routine database

Without any loss of generality, the routine database RD is formally modelled as a set of M examples $RD = \{T_1, \ldots, T_M\}$. Each training example $T_i = (R_1, \ldots, R_K)$ $(1 \leq i \leq M)$ represents a walking route, where R_j $(1 \leq j \leq K)$ is the data of the position j^{th} along the route, with K is the total number positions on the route. Each position $R_j = (\overrightarrow{MFV_j}, \overrightarrow{L_j})$ contains the magnetic field vector $\overrightarrow{MFV_j} = (x, y, z)$ recorded at that position, along with its Cartesian label $\overrightarrow{L_j} = (L_x^j, L_y^j)$, with x, y, z is the magnitude (μT) of the three axes.

The task is, given a magnetic sequence without the Cartesian label $(\overrightarrow{MFV_1}, \ldots, \overrightarrow{MFV_L})$, the system finds the best training trajectory that matches this sequence.

5.3.3 Estimating the user trajectory with the routine database

The user will have an app running in the background to collect the magnetic data automatically. This is also the common assumption for most passive monitoring systems. The recording process is triggered when the accelerometer readings are above a certain threshold for a window period of a few seconds which indicates that the user is moving (see Figure 5.6). This same process will stop when the user no longer moves, for which the accelerometer readings are within the threshold for a window period of time. It is worth noting that the use of accelerometer in this chapter is basic, although these accelerometer measures can tell more information about the user movements as in WiFi Simultaneous Localization and Mapping (SLAM) research [27, 45, 61, 87].

When triggered, the magnetic data is recorded continuously to generate a real-time trajectory. It may be possible that some false recordings will get through when the phone is momentarily picked up (e.g. to make a call). This issue may be avoided by setting an initial

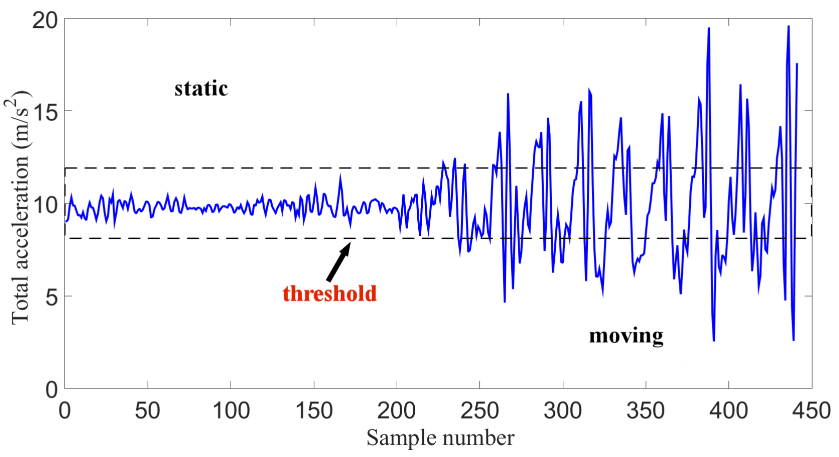


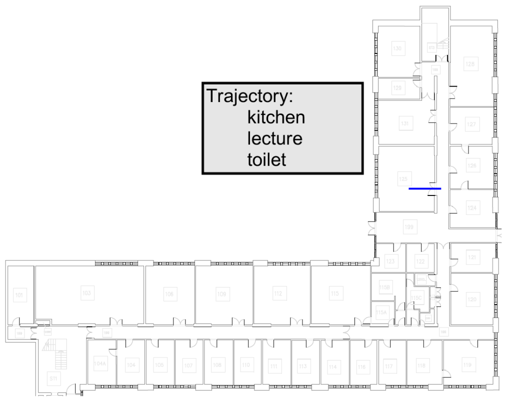
Fig. 5.6 Acceleration reading with the phone in the trouser pocket. When the device is static, the reading is around $9.8m/s^2$, which is equivalent to the earth's gravitational force [34].

threshold for the trajectory's length, so that the prediction module will only engage when the trajectory has accumulated at least a certain amount of data. This approach is left for future research.

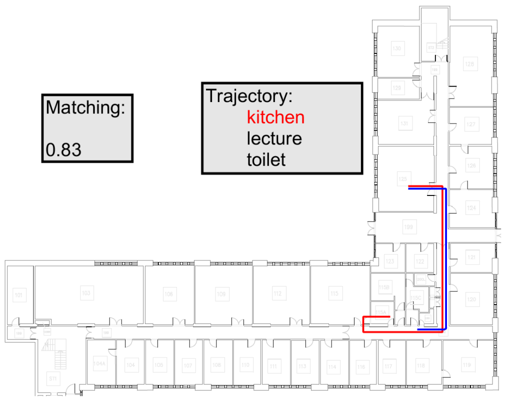
As the user travels, the system assumes that the training trajectory which closest matches the current real-time sequence is the correct route. This assumption, in turn, tells the system what the potential destination that the user may arrive at. However, the system does not stick to a single predicted trajectory. Through-out the journey, the user's magnetic sequence gets larger, and also becomes more unique. Every time a new reading arrives, the system re-calculates the measure to the training examples to find a better trajectory, if any. Therefore, the potential destination is dynamically adjusted as the user navigates the building.

To compare the difference of two magnetic sequences, this chapter employs Dynamic Time Warping (DTW). The benefits of DTW are to mitigate the difference in the measuring time of the training and real-time sequences, the ability to compare signal sequences of different lengths, and most crucially is to eliminate the temporal variance in walking speed [63, 66, 79]. The smaller the DTW measure is, the closer the two trajectories are. The variant of DTW employed in this chapter is called open-end DTW (OE-DTW) [29]. Normal DTW will stretch the shorter sequence to match the longer one up until the end, for which, this OE-DTW version relaxes the end-point constraint. It serves the purpose of this chapter better, because in the early stage of the journey, the user's MFV sequence is short, and may introduce certain bias when compared to the full training trajectories, even though they start from the same position. In principle, OE-DTW achieves its target by constructing different incomplete versions of the longer sequence, and picks the best match. In order to compare the DTW measure at different points of the journey, these measures will be normalised by dividing with the length of DTW aligned sequence. More details of OE-DTW[1] can be found in Appendix C.

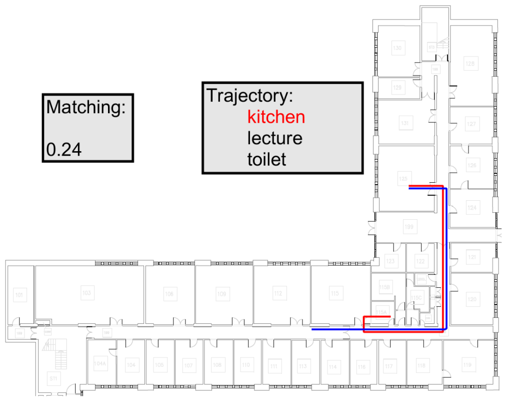
Figure 5.7 demonstrates an example of this estimation process in real-time[2]. The route prediction algorithm with the routine database is summarised below (see Algorithm 4).



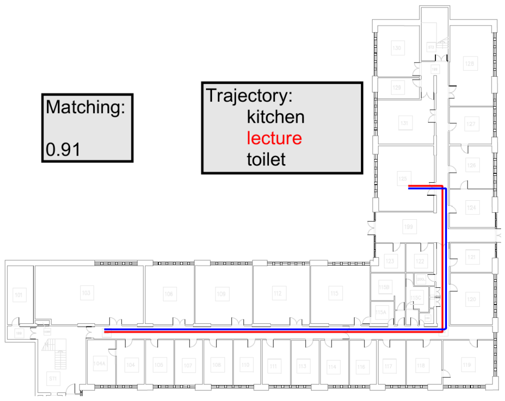
(a) The user starts moving from his office. The blue line demonstrates the user path.



(b) The red line shows the current highest matching route.



(c) However, the user turns to a different route in mid-way. The matching measure dropped as he navigates away from the predicted route (kitchen), which signalled a wrong prediction.



(d) The system switches to the next matching route (lecture room).

Fig. 5.7 An example of the dynamic route prediction with the routine database. The *3* training trajectories were Kitchen-bound, Lecture room-bound, and Toilet-bound. The matching number on the left is calculated by DTW and is scaled to [0,1] for demonstration purpose. The closer it is to 1, the more similar the predicted route is to the actual taken route.

[1]For the rest of the chapter, the term OE-DTW will be simply referred to as DTW, and DTW measure means normalised OE-DTW measure.

[2]A video demonstrating this process can be viewed at http://www.cs.rhul.ac.uk/~wruf265/dtw.mp4

If the DTW measures between the real-time sequence and all training trajectories are low, in other words, no training trajectory matches the current sequence, the system will simply make a single position-by-position prediction with the fingerprinting database. This situation may happen because the user is relatively new to the building, or because his walking routines have not been well-recorded. A more in-depth discussion of how to estimate the single user position based on the fingerprinting database was covered in the previous Chapter 4.

```
Data: Routine database with $M$ training trajectories $\{T_1, \dots, T_M\}$.
Result: Route prediction.

while user is still moving do
    read Magnetic field strength $MFV_k$ at current location $k$;
    /* merge it into a sequence                                   */
    $mag_{cur} = mag_{cur} + (MFV_k)$;
    /* Find the closest training trajectory                       */
    min = $\infty$ ;
    $T_{ref} = T_1$;
    for $i = 2 \rightarrow M$ do
        if $DTW(T_i, mag_{cur}) < min$ then
            min = DTW($T_i$, $mag_{cur}$);
            $T_{ref} = T_i$;
        end
    end
    return $T_{ref}$;
end
```

Algorithm 4: Estimating the trajectory with the routine database.

5.4 Related work

The earliest use of the magnetic field for indoor positioning was first seen in robotics navigation. It helps the robot orientate itself by maintaining a correlation to the magnetic North [40, 82]. However, the structure of most buildings greatly distorts the direction of the magnetic field (e.g. compass does not work reliably indoors).

Some early work suggested the use of the magnetic field with fingerprinting. However, they shared the same idea that the position's fingerprint is directly represented by the magnetic data [15, 54]. This approach is challenging, as explained earlier, since the magnetic field only contributes a maximum of three measures per location. The work described in this chapter makes use of the robust magnetic data via a sequence of measures.

5.5 Evaluation of performance

The experiments with the routine database in this chapter were performed on the RH test bed only, because this is the only test bed with the magnetic field data. The R implementation of OE-DTW[3] was used to conduct the experiments.

5.5.1 Criteria of evaluation for the routine database

Due to the difficulties in recruiting more experts to generate the routine database for many users, this chapter only has *1* routine database with *50* training trajectories, covering all three corridors on the ground floor of the RH test bed, with the starting and ending points in different offices (see Figure 5.8). The individual training trajectory is included in Appendix D. In particular, the longest trajectory has *62* MFV measures, and the average length is *34*. The magnetic field data was collected with the phone held in the user's hand at chest level, and the impact of the device's orientation is assumed to be negligible.

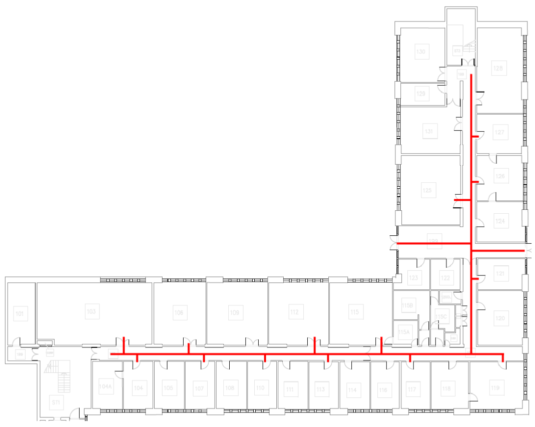
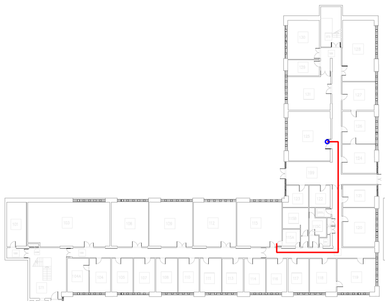
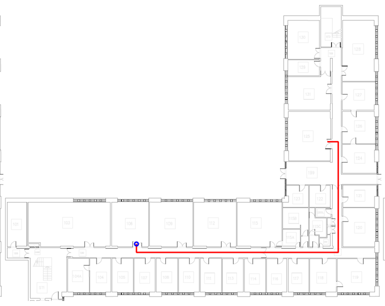


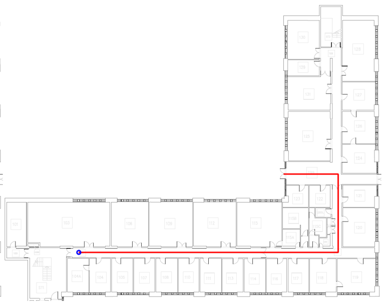
Fig. 5.8 Coverage of all training routes.

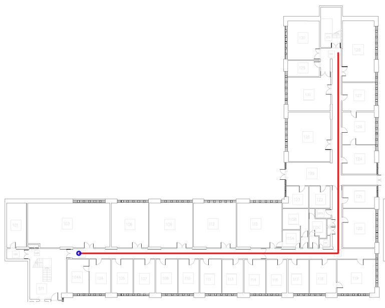
The test set contains *23* test trajectories measuring separately at different times from the training set, for which *10* of them are fully covered in the training set (see Figure 5.9). For every fully observed test trajectory, there are at least *3* training trajectories that start from the same position.

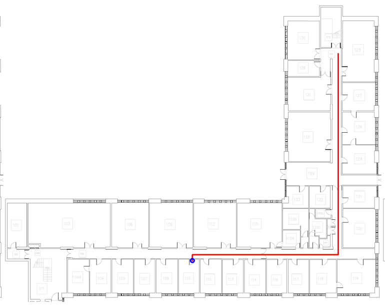
[3]http://cran.r-project.org/web/packages/dtw/ - last accessed in Sep/2016.

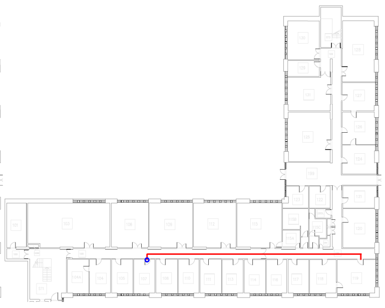

(a) From 125 to 115

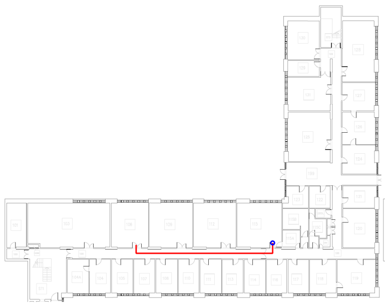

(b) From 106 to 125

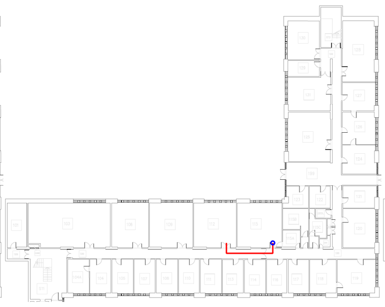

(c) From 19 to 99

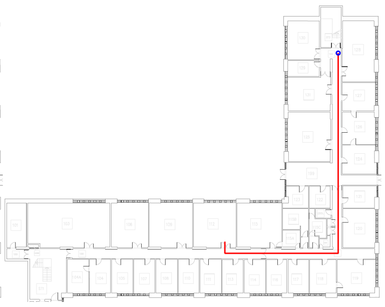

(d) From 19 to 199


(e) From 110 to 199


(f) From 107 to 119


(g) From 115 to 106


(h) From 115 to 112


(i) From 199 to 112

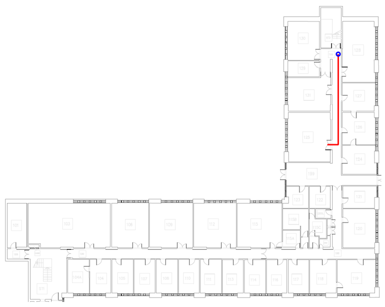
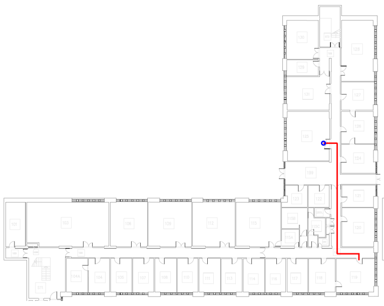
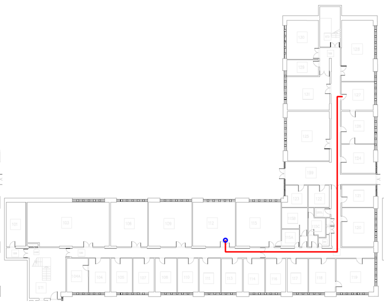
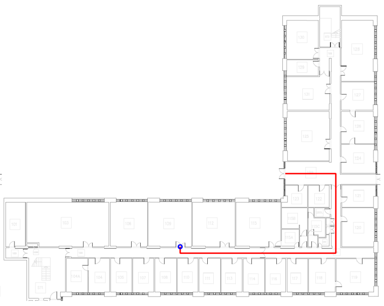
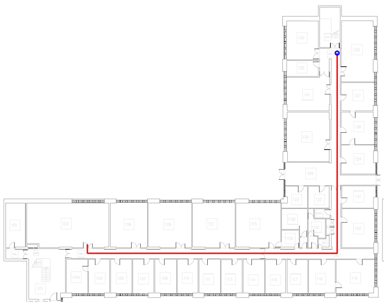

(j) From 199 to 125

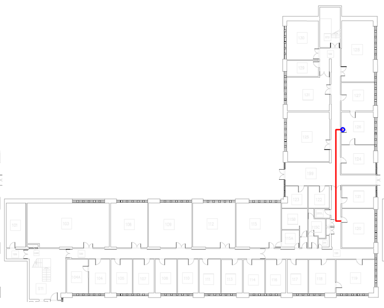
Fig. 5.9 The *10* fully observed test trajectories. The blue circle denotes the starting point.

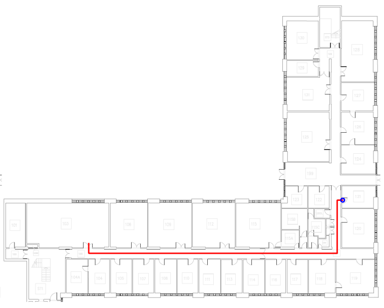

(a) From 125 to 119

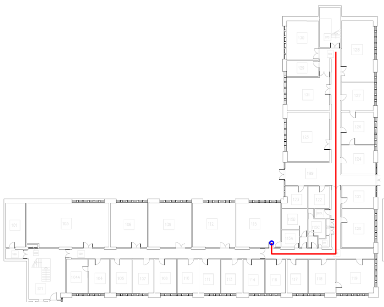

(b) From 112 to 127

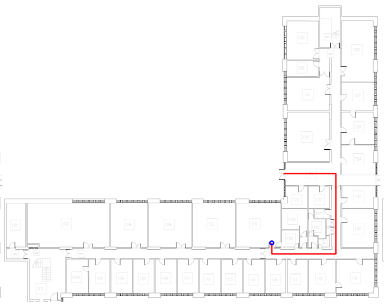

(c) From 109 to 99

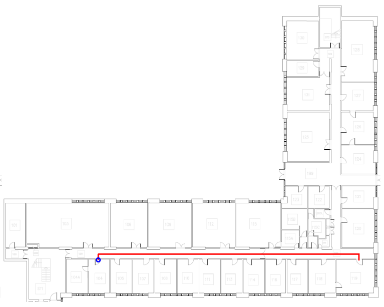

(d) From 199 to 103


(e) From 126 to 120


(f) From 121 to 103


(g) From 115 to 199


(h) From 115 to 99


(i) From 104 to 119

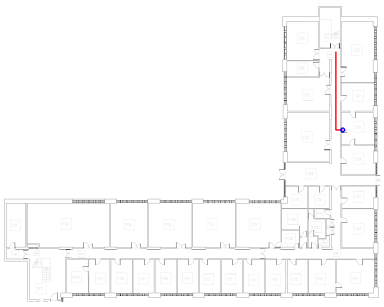
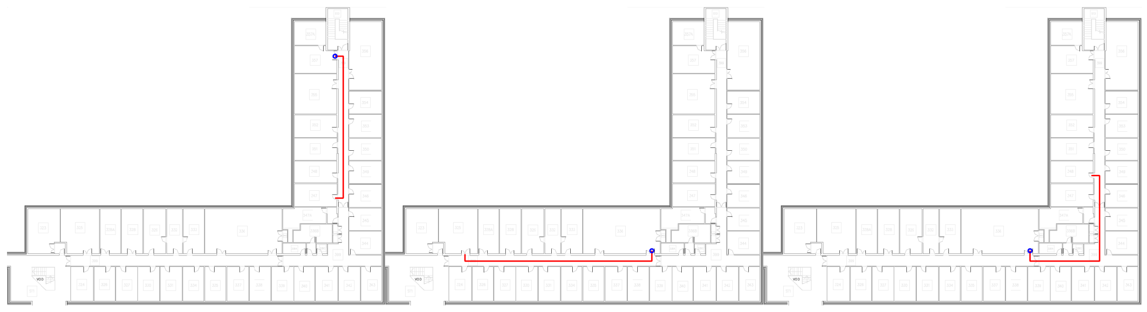

(j) From 126 to 199

Fig. 5.10 The *10* partly observed test trajectories. The blue circle denotes the starting point.

The other *10* test trajectories are partly covered so that only the beginning portion of the route was recorded in the training set. These routes either finished in unknown positions or the remaining half of their trajectories was not observed (see Figure 5.10).

The remaining *3* test route are completely new trajectories, which were recorded two floors above the training space in the same building (see Figure 5.11). There is no information of these routes in the training database.



(a) From 357 to 347 (b) From 336 to 325 (c) From 336 to 348

Fig. 5.11 The *3* completely new test trajectories measured on a different floor. The blue circle denotes the starting point.

Using this routine database, the chapter aims to address the following questions.

(i) **How well the routine database predicts the route and the potential destination?** The main purpose of the routine database is to estimate the user's trajectory and destination. Therefore, it is natural to assess how well it performs, in terms of how quickly the system picks the correct trajectory and how often the system switches trajectory during the journey.

(ii) **Can the routine database improve the accuracy of the single positioning prediction?** If the routine database identifies the correct route the user is taking, can it also improve the positioning prediction, compared to the result produced by the fingerprinting database?

For the experiments in this chapter, the test trajectory is broken down into ordered individual MFV which represents a single position on the trajectory. Each MFV will be accumulated one-by-one into a sequence to simulate a walking user in real-time. At each step, the system computes the DTW measure of the current sequence to every training trajectory. The training trajectory with the smallest measure will be chosen as the predicted route.

5.5.2 The destination and route prediction evaluation

The first experiment evaluates the accuracy of the route and the destination prediction. Two observations stand out from this experiment. Firstly, in terms of destination prediction, the accuracy was not particularly high. At all *317* positions on the *10* fully observed test trajectories, the system managed to suggest the correct destination for only *156* positions, which was less than *50%* (see Figure 5.12). This result is understandable, because there are several training trajectories that start from the same position for each test trajectory. Thus, the system was not always able to pick the training trajectory that leads to the correct destination. Unsurprisingly, none of the remaining *13* partly observed and new test trajectories suggested the correct destination.

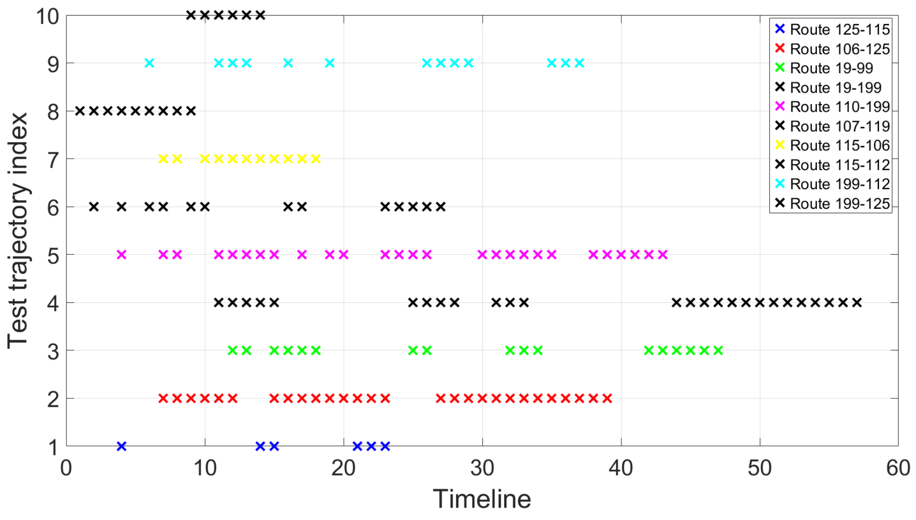
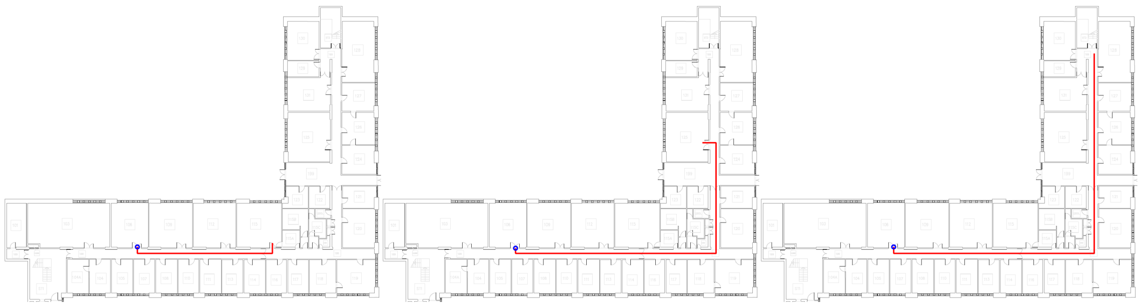


Fig. 5.12 Destination prediction accuracy with the routine database. Each horizontal line corresponds to the test trajectory. Each cross demonstrates a position along the journey where the destination prediction is correct.

However, in terms of route prediction, each position is deemed correctly predicted as long as the predicted trajectory overlaps with the correct trajectory from the beginning up until the current position. For example, considering test trajectory 106-125, which goes from office 106 in the lower-left to office 125 in the top-right (see Figure 5.9b), there are *3* training trajectories that start from the same office with this test trajectory but end in different positions (see Figure 5.13, the full routine database is listed in Appendix D).

In this experiment, when the user's sequence was short, the system matched it to the shortest training route 106-115 (see Figure 5.13a). Although the user's actual route was 106-125, this taken route overlaps the predicted route 106-115. In other words, the user was



(a) Training route 106-115. (b) Training route 106-125. (c) Training route 106-199.

Fig. 5.13 The *3* training trajectories that start from office 106. The blue circle denotes the starting point.

actually following the same trajectory up until this point. In terms of positioning prediction, this predicted route did not make much difference, which will be evaluated in the next section. When the user's magnetic sequence grew larger as he went past room 115 - the destination of the current predicted route, the system re-calculated the DTW measure and found the correct matching route 106-125. Overall, at all *317* positions on the *10* fully observed test trajectories, the system managed to suggested the correct trajectory for *286* positions, which was almost *90%* (see Figure 5.14). This result also illustrated that all the wrong route predictions happened at early positions when the sequence was short. For this routine database, all test positions with at least *9* MFV found their correct routes.

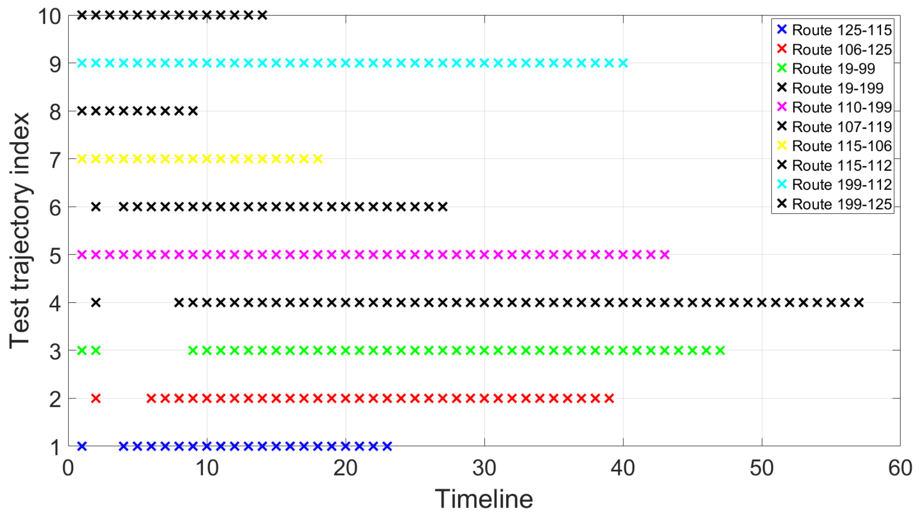
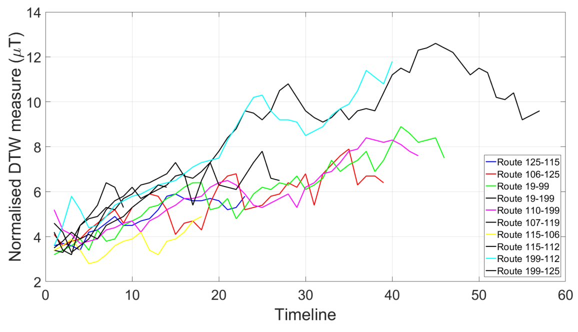
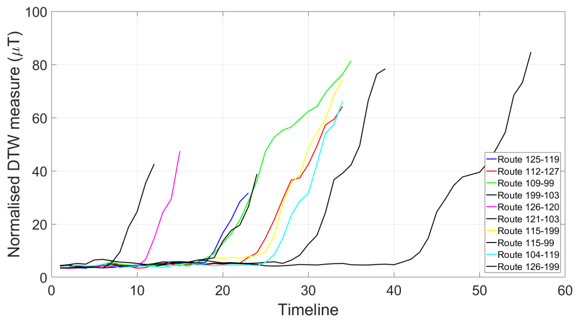


Fig. 5.14 Route prediction accuracy with the routine database. Each horizontal line corresponds to the test trajectory. Each cross demonstrates a position along the journey where the route prediction is correct.

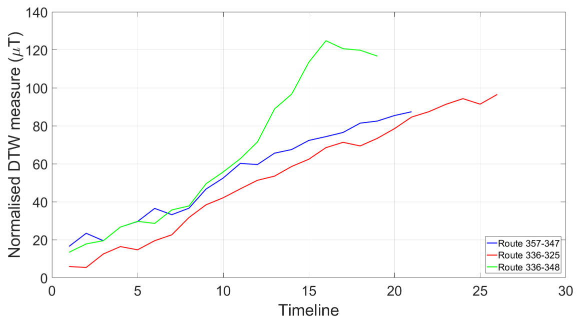
The second experiment evaluates the value of the DTW measure between the test trajectories and the training trajectories. For all *10* fully observed test trajectories, the DTW measure at each position on the test trajectory was relatively low, with the highest measure recorded at just *12.6* (μT) (see Figure 5.15a). This low measure is understandable because there are several training trajectories that match these fully observed test trajectories. For the *10* partly observed test trajectories, the DTW measures were also low, up until the point where the user strayed away from all training trajectories, from which the DTW measure quickly grew up (see Figure 5.15b). For the *3* completely new test trajectories, the DTW measures went up rapidly right from the very beginning with the biggest measure recorded at *116.7* (μT), because there was no training trajectories that match these new test trajectories (see Figure 5.15c). Table 5.1 summaries the DTW measures for the three categories of the test trajectories.



(a) Fully observed test trajectories. The DTW measure is relatively low for all positions.



(b) Party observed test trajectories. The DTW measure remains low until the mid-way positions where the user took a new route.



(c) New test trajectories. The DTW measure increases rapidly from the very beginning.

Fig. 5.15 The minimum DTW measure at each position on the test trajectory through-out the journey.

This experiment also exposed how often the system changed trajectory during the journey, since it always picked the training trajectory with the lowest DTW measure to the

Table 5.1 Comparison of the normalised DTW measure for the test trajectories.

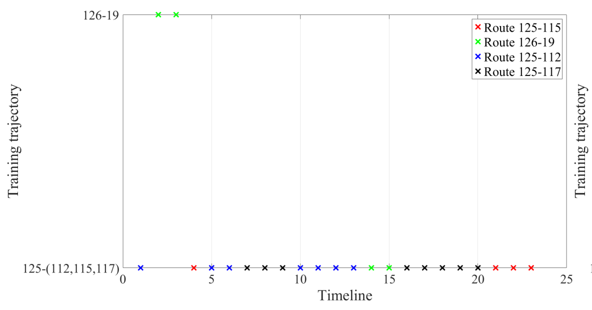
	Max DTW measure	**Min DTW measure**	**Average DTW measure**
Fully observed test trajectories	12.6	3.2	6.8
Partly observed test trajectories	84.7	3.3	31.2
Completely new test trajectories	116.7	5.4	84.1

current sequence at each position. Figure 5.16 pointed out that for this routine database, when the test trajectory had less than *10* MFV, the system switched routes about *5* times on average across all *23* test trajectories. In contrast, when the length of the test trajectory was within *[10 20]* MFV, the system switched route only *2* times on average. For all *10* fully observed test trajectories, there were *4* routes being used on average through-out each test trajectory. Overall, the longer the test trajectory was, the more successful the system could match it to the correct training trajectory, and the less frequent the system switches routes.

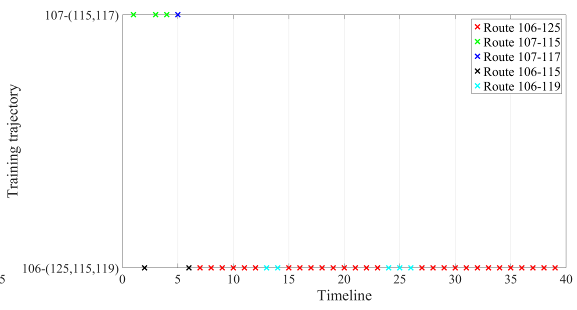
5.5.3 Single positioning estimation evaluation

Although the main purpose of the routine database is to predict the potential walking trajectory and the destination, it is interesting to investigate the positioning accuracy while the user is travelling on the predicted route. At any stage of the journey, the user's position will be regarded as the Cartesian label of the last position on the training trajectory returned by DTW. It is worth recalling that the OE-DTW variant of DTW employed in this chapter tests different incomplete versions of the training trajectory by relaxing the end-point constraint to find the best matched version. For this experiment, all *317* positions on the *10* fully observed test trajectories will be used as test positions. To compare the positioning estimation with the fingerprinting database, the WiFi RSS vector at each test position was used to estimate the user position.

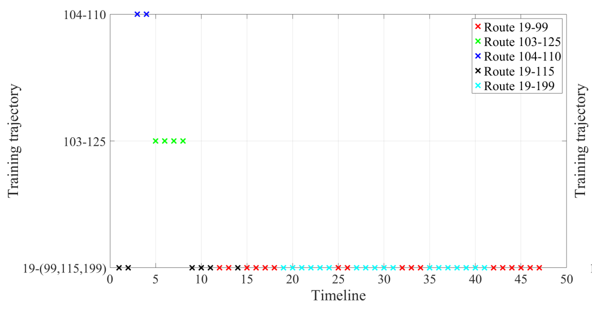
Figure 5.17 demonstrates a remarkable positioning accuracy using the routine database with up to *2.2* metre positioning error, *95%* probability with the routine database, compared to *3.5* metres, 95% probability with the fingerprinting database. This result emphasises that when the system is certain of the user's walking trajectory, it can estimate the user's whereabouts using the individual positions recorded on the training trajectories much quicker with higher accuracy. In contrast, the fingerprinting approach must go through the whole training database to make prediction for every position.



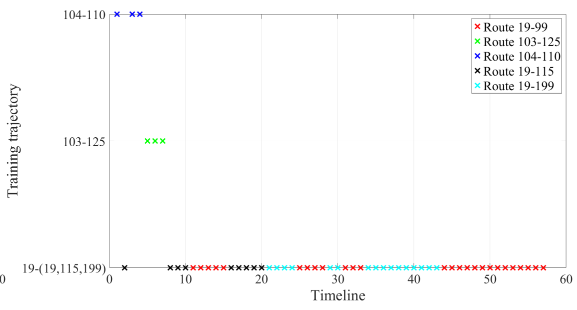
(a) Test trajectory 125-115



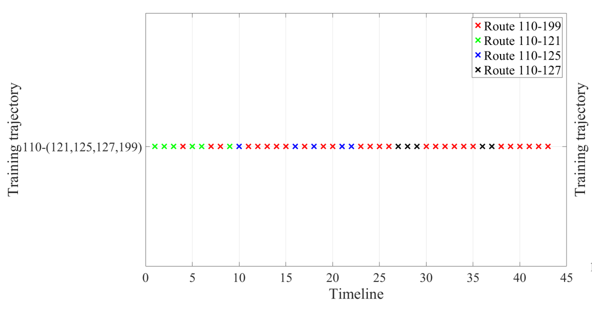
(b) Test trajectory 106-125



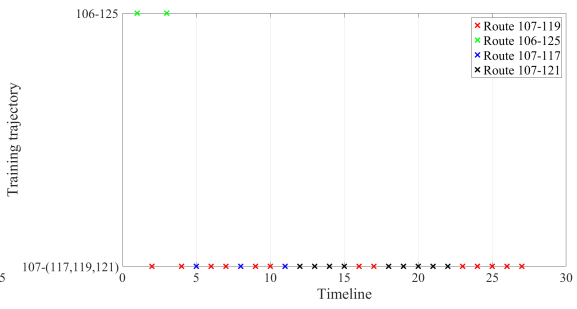
(c) Test trajectory 19-99



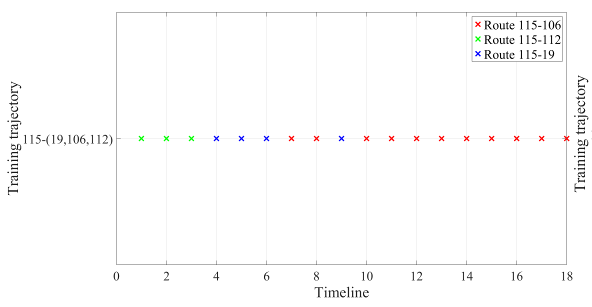
(d) Test trajectory 19-199



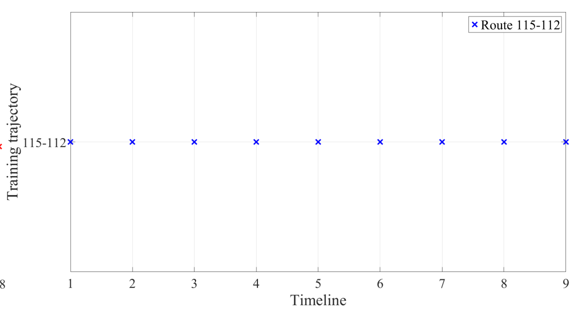
(e) Test trajectory 110-199



(f) Test trajectory 107-119

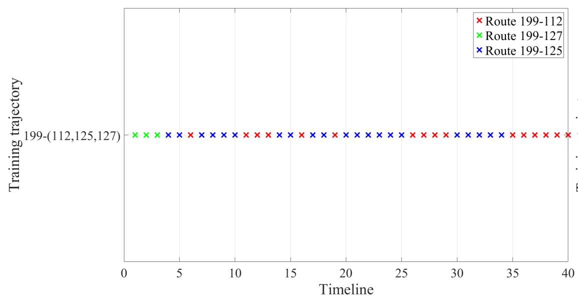


(g) Test trajectory 115-106

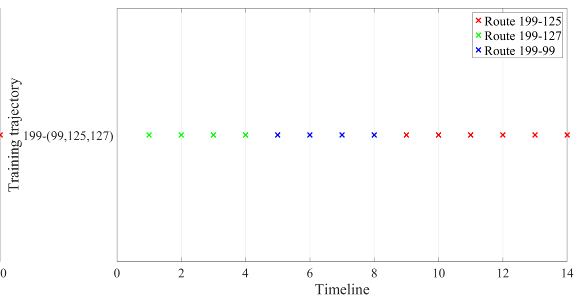


(h) Test trajectory 115-112

Fig. 5.16 How often the system switches route, through-out the journey of the whole test trajectory.



(i) Test trajectory 199-112



(j) Test trajectory 199-125

Fig. 5.16 How often the system switches route, through-out the journey of the whole test trajectory.

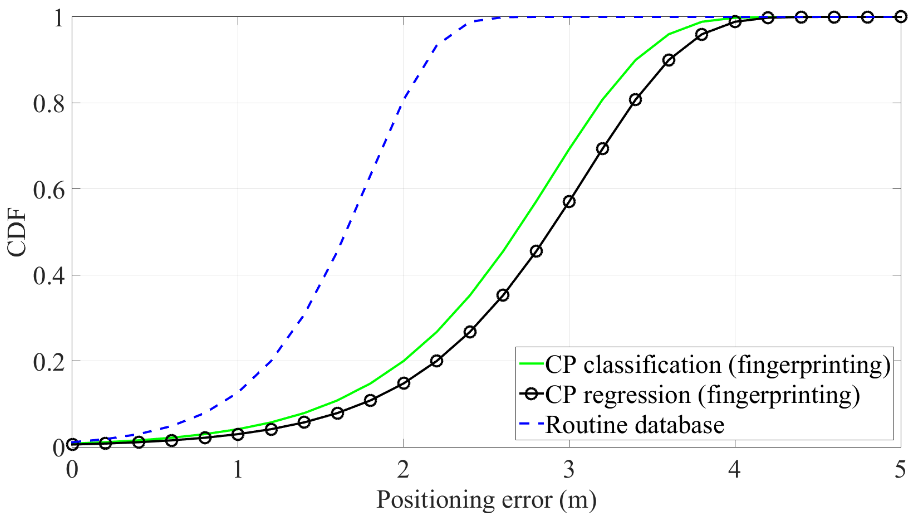


Fig. 5.17 Performance accuracy of single position estimation with the routine database.

There are two reasons why the estimated position using the routine database may not perfectly match the true position. Firstly, the training trajectory does not line up completely with the real-time trajectory. The version of the training trajectory returned by DTW may be a few measures before or after the real-time trajectory. Secondly, there are many training trajectories starting in the same position, the current predicted one is not necessarily the correct one.

5.6 Summary and further work

Traditional fingerprinting has normally been considered as a tracking service to guide the user to his wanted destination. This chapter addresses fingerprinting as a sentient service for the regular users to allow the system to predict their intended walking route and the potential destination in advance. The chapter has shown that such objective was possible by using the magnetic field data to generate a routine database for the user. The major difference of the proposed approach and other fingerprint-based ones is that the real-time readings are accumulated as a sequence to distinctively represent the user's walking trajectory. Using this routine database, the system can predict the user position with higher accuracy than using just the fingerprinting database. The chapter also demonstrated that the longer the system observes the magnetic field data, the more accurate the predicted routes are, and the less frequent the system changes route during the journey.

The limitation of the proposed approach is that it is more feasible for the buildings with narrow corridors (e.g. those in the RH test bed). It was not designed to identify the user's position in a large area (e.g. in the middle of the lecture hall), for which the fingerprinting database will be more suitable.

Lastly, some practical discussions are outlined here. These discussions were not presented in the main body to maintain the focus of the chapter.

(i) **How does the system cope when the user temporarily turns off his phone, or when he takes a break during his journey?** When the walking route is broken, the system simply makes a single position-by-position prediction using the fingerprinting dataset.

(ii) **What if the user turns on the tracking app in the middle of his journey?** With the current design, the system predicts the trajectory at the beginning of the user's journey. It may be possible to make prediction in mid-way, although this concept has not yet been examined in this chapter.

(iii) **Why not use a universal routine database for every user in a similar manner as the fingerprinting database?** Firstly, the chapter assumes that most users often follow their own preferred path to navigate the building. Secondly, a universal database may contain too many trajectories, for which, certain paths may not be used at all by some users.

This chapter has laid the foundation for using the routine database to predict the walking trajectory. Some potential further works to improve this concept are outlined below.

(i) The personal routine database can be further personalised by adding the approximate starting time for each journey. Depending on the user's timetable, he may favour one particular route at a certain time, and this extra information may improve the accuracy of the decision making process. More importantly, the timetable also tells how often the event takes place, which can provide the initial value for the frequency of the trajectory. For example, the timetable indicates the user has *4* lectures per week, thus, the F(office, lecture) = *4*.

(ii) It may also be useful to set a threshold for the user's real-time trajectory, so that the prediction engine only engages when the trajectory reaches a certain length. This approach has two benefits. Firstly, it guarantees that the accumulated user trajectory is long enough to avoid many initial similar matchings with the training trajectories. Secondly, it avoids the scenario where the recording process is falsely triggered by the user temporarily picks up the phone to make a call.

(iii) Since the routine database is constantly updated with new WiFi RSS and magnetic field data, it is also a great resource for crowd-sourcing. The training examples from the fingerprinting database may be updated with the latest data extracted from the individual routes in the routine database.

(iv) It may be beneficial to reduce the number of times the system has to search through the routine database by introducing a DTW measure threshold. There are two scenarios for this threshold.

   (a) If the DTW measure between the current training trajectory and the real-time trajectory is below the threshold, the system believes that the user still follows the predicted route. Hence, it does not bother checking other training trajectories.

   (b) If the measure rises above the threshold, the system believes that the predicted trajectory is no longer correct. It re-calculates the measure to other training trajectories to find a better match.

   There are two parameters to consider here. Firstly, a DTW measure threshold needs to be decided in advance, so that the system knows when to drop the current predicted trajectory to find a better one. The challenges here are that if this DTW measure threshold is low, the system keeps following the wrong training trajectory, although the user has turned to a new route. In contrast, if this DTW threshold is high, the system becomes too sensitive, and may drop the correct training trajectory because of the noises recorded in the real-time trajectory. Secondly, a time window parameter also needs to be considered to tackle the challenge with the magnetic noises. This time window ascertains that the system does not accidentally drops the correct training trajectory. Only when several low DTW measures are recorded within this window that the current training trajectory is dropped.

All the chapters discussed so far in this thesis assume that the new WiFi RSS sample from the user is fully observed from his unknown position. However, there are certain factors including the operation of the WiFi scanning process that may result in incomplete signal observations. The next chapter will investigate this phenomenon.

Chapter 6

Detecting and handling missing Access Points in the real-time WiFi scan

"When you realise you've made a mistake, take immediate steps to correct it."

– Dalai Lama XIV.

The reliability of WiFi fingerprinting depends much on the availability of the indoor WiFi APs. An AP which goes missing during the real-time positioning phase results in an incomplete signal strength observation, which does not match any recorded training examples. To further complicate the problem, the missing AP is not just caused by the human intervention, but also happens due to the nature of the WiFi scanning process. Surprisingly, this problem is conveniently overlooked in most fingerprinting researches.

This chapter addresses the aforementioned issues in two phases. In the detection phase, given the user's real-time WiFi RSS data, the system attempts to spot any potential missing APs, based on the positioning prediction and the APs' behaviours from the previous reading. In the recovery phase, it then estimates the RSS of the missing APs by testing all potential RSS's values. The proposed technique has been validated to confirm its success in detection and recovery of the missing WiFi APs.

6.1 The WiFi scanning process

There are two methods to perform WiFi scanning on an Android phone, which are active scanning and passive scanning. Both of them operate at the IEEE 802.11 MAC layer. This section discusses the mechanism of each method to understand how they affect the missing AP problem.

With passive scanning, the phone listens on each channel for the beacons sent by the APs periodically. The IEEE 802.11 standard specifies that the WiFi APs must send out beacons every *100* ms on all channels simultaneously [38]. The author looked into the Android source code to find out that the Android phone is implemented to stay on each channel for *120* ms. Theoretically, this extra *20* ms is sufficient for the phone to discover at least one beacon per channel. With *13* channels of the European *2.4* GHz spectrum, it will take at least *1,560* ms to scan all nearby APs. In reality, the total scanning time is longer because of the information processing delay. Although passive scanning consumes less battery power, the device cannot pick up hidden APs, which are configured not to send out any beacon.

With active scanning, the phone sends the probe request frames on each channel, then listens for a minimum of *MinChannelTime (ms)* for the probe responses from the WiFi APs, before moving on to the next channel. If more than one probe response is heard, the phone continues to listen till the *MaxChannelTime (ms)* has elapsed. However, the IEEE 802.11 does not define an absolute value for *MinChannelTime* and *MaxChannelTime*. The author looked into the Android source code and found that Google's implemented just a constant *MaxChannelTime* waiting time of *30* ms per channel. Thus, in theory, it will take at least *390* ms to scan all *13* channels.

Table 6.1 Passive and active scanning time in *5* hours with the two experimented phones.

	Passive Scanning		**Active Scanning**	
	Nexus 4	Samsung S5360	Nexus 4	Samsung S5360
Slowest scan	4561 ms	4418 ms	1045 ms	1532 ms
Fastest scan	3712 ms	3041 ms	944 ms	1400 ms
On average	4023 ms	4014 ms	962 ms	1419 ms

To experiment the waiting time of active scanning and passive scanning in the real-world, two Android phones (Google Nexus 4 and Samsung Galaxy S5360 Y) were used to continuously scan for WiFi APs in one spot in the building of the RH test bed. More than

30 APs were discovered on average per scan. Table 6.1 suggests that the actual scanning time for both active scanning and passive scanning were longer than the theoretical ones. Overall, it took about a second per scan for active scanning, and about *4* seconds per scan for passive scanning. Clearly, active scanning is the preferred choice for fingerprinting with shorter scanning time, and the ability to pick up hidden APs. However, there is a higher chance that one or more APs are not seen in a single scan with this option. In addition, the user is often on the move, thus, there will not be sufficient time to perform several scans per position in real-time.

To verify the impact of the Android active scanning process on the number of missing APs, the WiFi APs were scanned at *5* different positions in the building of the RH test bed, where the APs' coverage of these positions is known. For the static-phone experiment, the phone was left at these fixed positions. For the moving-phone experiment, the phone was held by a user who slowly moved it in the same position. Both experiments were performed in the same amount of time of *10* minutes. The measuring process took place outside the office hours to minimise the impact of the human movements and the environmental changes. This experiment revealed some interesting correlations between the number of missing APs and the WiFi RSS in the same period of time. Firstly, with over *3,000* static WiFi scans, about one third of them had missing APs. Interestingly, for the same number of moving WiFi scans in the same positions, about *43%* of them contained missing APs, which was a higher number. It may be the instability and the sensitivity of the phone's antenna that contributed to this problem. Secondly, it was suggested that the weaker the RSS was, in other words, the further the user was to the AP, the more likely the AP was missed during a WiFi scan (see Figure 6.1). This is understandable, because the signal of those APs was fading quicker.

6.2 Detecting and correcting missing WiFi APs

To detect the missing APs, it is assumed that the user is present when the APs go missing. In other words, the phone can observe the complete signal data at position A when all APs are functional, and the incomplete signal data at the consecutive position B when some APs are missing. An AP is declared as missing if:

- A strong AP was observed at position A. However, the same AP is not seen at B; or
- There is a big difference in positioning prediction from A to B. All the weak APs that appeared in A but disappeared in B will be assumed to be missing.

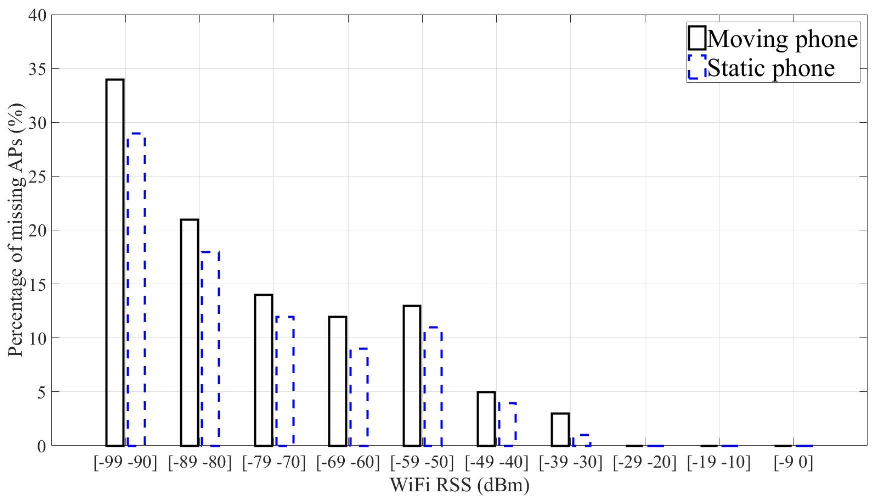


Fig. 6.1 The likelihood of missing APs based on the WiFi RSS. Weak APs are prone to be missing. Moving phones during WiFi scans are more likely to experience missing APs.

The above detection phase serves as a pre-caution to alert the administrator to investigate the presumed missing APs. The recovery phase, however, is only triggered by the second condition, that is, there is a big difference in the positioning prediction between A and B, because, ultimately, the only criterion the system cares about is the positioning accuracy. The positioning threshold to trigger this phase should include the gap between A and B, as well as taking into account the average positioning error when A and B are in the same position. For instance, the distance between A and B is *1* metre, the underlying algorithm used to estimate the user position is W-KNN, which has a positioning error of *3.5* metres on average for the RH test bed. Thus, the threshold should be about *4.5* metres. More details about this positioning threshold will be evaluated later on. The steps to estimate the RSS for the suspected missing APs are outlined below.

(i) A list of all APs which appear in the last position A, but disappear in the current position B is complied. The list is sorted in decreasing RSS, so that, the APs with stronger RSS rank first: $L = \{AP_1, \ldots, AP_N\}$.

(ii) At each step, one AP in this list, starting from the strongest AP_1 will be considered. At step K, all APs $\{AP_1, \ldots, AP_K\}$ will be considered.

(iii) For each suspected missing AP, eight WiFi RSS $\{-30, -40, \ldots, -100\}$ (dBm) within the *[1, -100]* (dBm) theoretical interval with a step length of *10* dBm will be tested.

(iv) For each step, a positioning prediction using the estimated RSS for all suspected missing APs will be made.

(v) The process terminates as soon as the distance between the last position A to the positioning prediction is below the threshold discussed earlier. The estimated RSS of the current positioning prediction will be used as the final result.

This recovery phase was based on three observations. Firstly, stronger APs have more impact on the positioning prediction, hence, they should be prioritised over weaker ones. Secondly, it is not necessary to test all *100* theoretical WiFi RSS, because some very high RSS within *[-1, -20]* (dBm) did not appear in the training database of this thesis, and may never be observed in the real-world at all. In addition, the indoor WiFi RSS has a roughly *10* dBm variation, as discussed in Chapter 5. Thirdly, if the number of suspected missing APs is low, it may be possible to trial all the RSS with all the suspected missing APs, and the estimated RSS with the smallest positioning distance to the previous position A will be chosen.

6.3 Evaluation of performance

This section assesses the proposed detection and recovery phases for the missing APs. The experiments in this chapter were performed on the RH and the Cam test beds only, because the training examples in the UJI dataset are a mixture of contributions from multiple devices, and thus there is no continuity of the WiFi RSS in consecutive training positions. To work out the positioning prediction, Naïve Bayes and W-KNN will be used. For the detection phase, when an AP is missing, it will be given a *-100* dBm measure in the case of W-KNN. In the case of Naïve Bayes, a small constant will be given to each posterior probability of the AP to avoid the probability of the training example being zero.

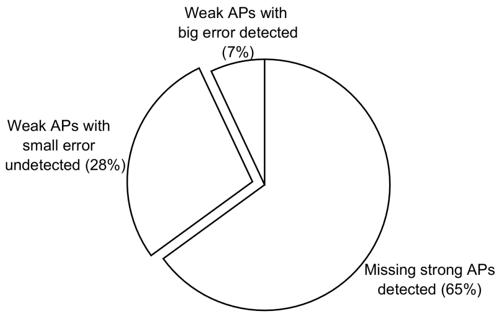
6.3.1 Generating the fingerprints with missing APs

For this evaluation, the training set was separated into pairs of fingerprints (r, r') of continuous positions. Since there are several fingerprints at each training position, only the most complete fingerprints with the highest number of recorded APs were selected for this experiment. To create a fingerprint with missing APs, the RSS of each AP_i $(1 \leq i \leq N)$ in $r' = (AP_1, \ldots, AP_N)$ was removed to generate N incomplete versions of the same fingerprint. This process was repeated with two APs being removed, until $(N-1)$ APs were removed. However, due to the high number of APs in each fingerprint (up to *38* for the RH test bed, and *13* for the Cam test bed), a smaller test set of *1,000* fingerprints with missing APs was randomly selected for each test bed. The chosen test fingerprints cover all possible number of missing APs, from *1* to 37 for the RH test bed and from *1* to 12 for the Cam test bed.

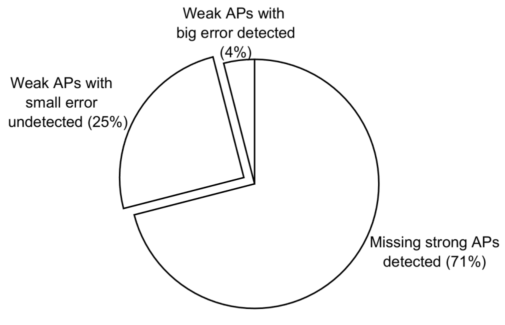
For each pair of test samples, the system tries to detect the missing APs using the complete reading *r* of the previous position, based on the two criteria set out in Section 7.3. For the evaluation purpose, W-KNN and Naïve Bayes approaches will be used as the algorithms to estimate the position of the fingerprint.

6.3.2 Detection of the missing APs evaluation

Figures 6.2 and 6.3 demonstrate that out of all the fingerprints with missing APs, *72%* of them were successfully flagged with Naïve Bayes and *69%* with W-KNN, for the RH test bed. This result implies that Naïve Bayes seems to be more sensitive to the missing APs than W-KNN. Both algorithms had no issue with the first cue when a strong AP was not seen in the fingerprint. The difference in the detection rate was caused by the second cue, that is, there is a big difference in the positioning prediction between the two fingerprints, and W-KNN seems to cope well by returning close positioning predictions even with the missing APs. A similar result was observed for the Cam test bed. For the evaluation purpose, the positioning difference to trigger the second condition was set at *4.5* metres for the RH test bed, *2.8* metres for the Cam test bed, and strong APs were those higher than *-70* dBm. The assumption was that the user may not travel more than *4.5* metres within two consecutive WiFi active scanning of about a second apart, including the *3.5* metre average error for the RH test bed. For the Cam test bed, the distance between two consecutive positions is *30* centimetres, and the average positioning error is about *2.5* metres.



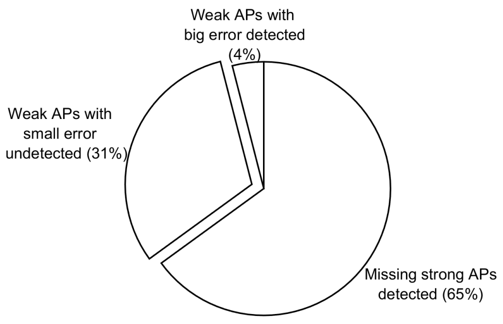
(a) RH test bed.



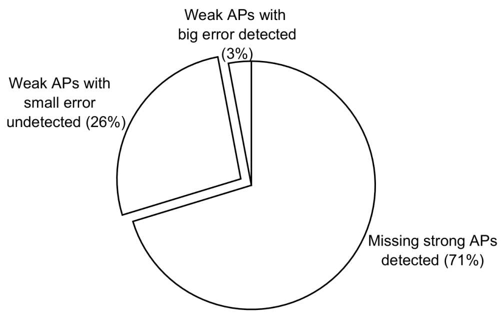
(b) Cam test bed.

Fig. 6.2 Successful detection rate of missing APs using Naïve Bayes.

The readers may be concerned of the high number of un-detected cases, with *28%* to *31%* undetected test cases for the RH test bed, and *25%* to *26%* for the Cam test bed using



(a) RH test bed.

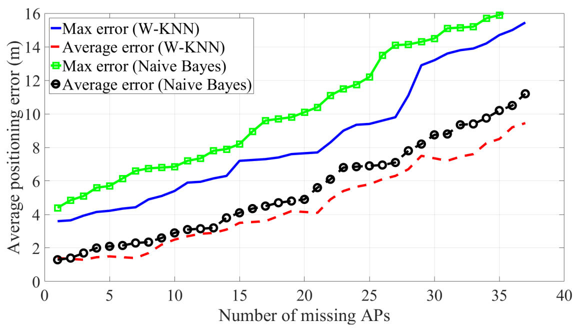


(b) Cam test bed.

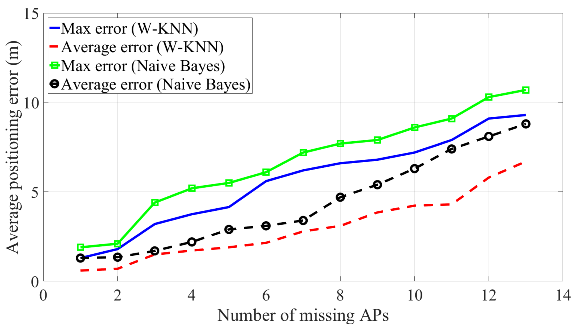
Fig. 6.3 Successful detection rate of missing APs using W-KNN.

W-KNN and Naïve Bayes. Nevertheless, it is worth noting that these undetected cases had weak missing APs in the range of *[-99, -70]* (dBm) and the difference in their positioning estimation was minimal (less than *3.5* metres for the RH test bed, and *1.8* metres for the Cam test bed), which was not picked up by either algorithm.

Figure 6.4 demonstrates the relationship between the number of missing APs and the positioning error. This result suggests that the fewer the APs left, the less accurate the positioning prediction was. The accuracy went from about *4* metre error when just two APs were missing, to *7* metre error when *10* APs were missing for RH test bed using Naïve Bayes. Performance-wise, W-KNN was more resilient against missing APs than the Naïve Bayes, with up to about *2* metres more accurate on average for the RH test bed, and about *1* metre for the Cam test bed. This is understandable, because Naïve Bayes relies much on the probability of the missing APs' RSS. Although this experiment showed all the possible scenarios including when all APs but one went missing, these cases are unusual, and it is more likely that only a few APs may go missing in the real-world.



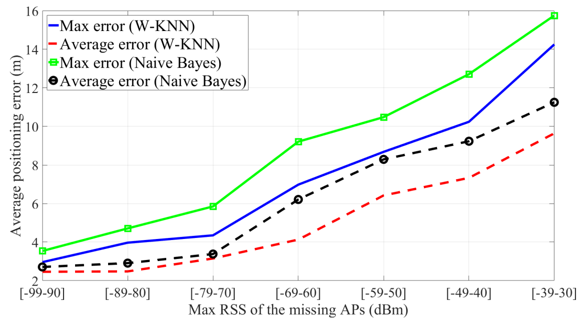
(a) RH test bed. The maximum number of APs observed for a single RSS vector was 38.



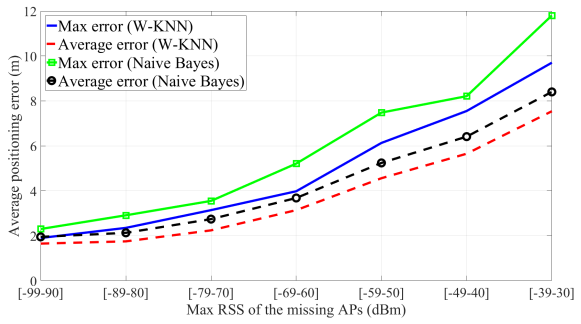
(b) Cam test bed. The maximum number of APs observed for a single RSS vector was 13.

Fig. 6.4 The impact of the number of missing APs on the positioning estimation.

To understand how much the strength of the RSS may influence the positioning estimation, Figure 6.5 illustrates the relationship between the maximum RSS of the missing APs and the positioning error it causes. It suggests that the stronger the missing APs' RSS were, the bigger the positioning error would be, for both W-KNN and Naïve Bayes. The changes in the positioning error were relatively small for weak RSS within *[-99 -70]* (dBm), the transitional period happened from *-70* dBm onwards, where up to *9* metre error was observed for the RH test bed, and up to *5* metre error was observed for the Cam test bed with Naïve Bayes.



(a) RH test bed.



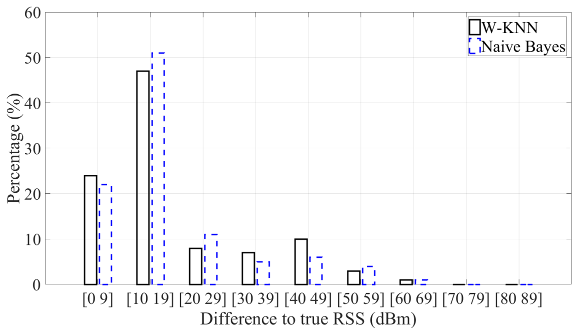
(b) Cam test bed.

Fig. 6.5 The impact of the strength of the RSS on the positioning estimation.

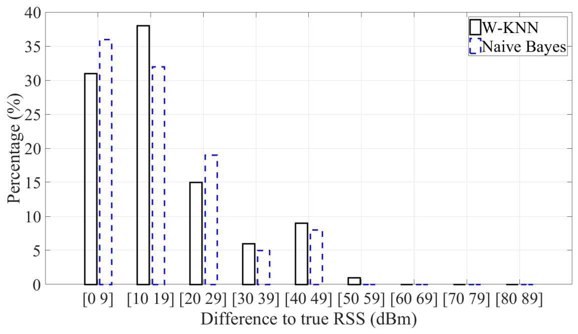
6.3.3 Estimating the missing APs' RSS evaluation

For this evaluation, the system tries to estimate the RSS for those missing APs, according to the algorithm discussed in Section 6.2. Ideally, the estimated RSS is preferred to be as close to the actual RSS as possible. Figure 6.6 illustrates that the missing APs' RSS were successfully estimated using both W-KNN and Naïve Bayes with minimal difference to the actual RSS. For both methods, the majority of the estimated RSS was well within *20* dBm of the true RSS.

To understand the benefit of the above estimated RSS, the next experiment compares the positioning prediction of the fingerprints with the estimated RSS and the same fingerprints with the missing APs. In other words, this experiment demonstrates how much improvement the positioning estimation will recover with the estimated RSS. Figure 6.7 illustrates a strong performance improvement from the fingerprints using the estimated RSS in this chapter. Compared to the same fingerprints where the missing APs were simply ignored by assigning a *-100* dBm for W-KNN, or a small constant to avoid zero probability for Naïve Bayes, the positioning error was reduced from *10.5* metres, *80%* probability to about *6* metres, *80%*



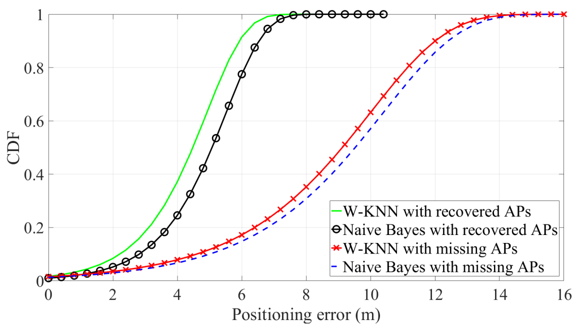
(a) RH test bed.



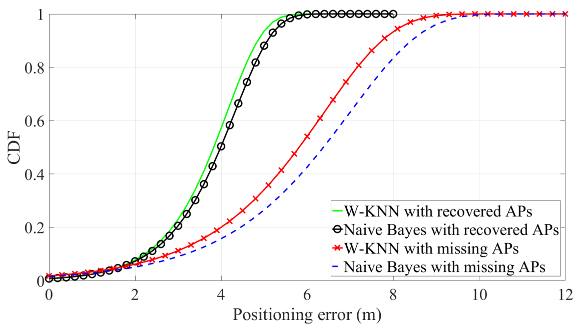
(b) Cam test bed.

Fig. 6.6 The difference between the estimated RSS and the actual RSS.

probability with W-KNN, which is about *40%* improvement for the RH test bed. With the Cam test bed, the positioning error was reduced from *7* metres, *80%* probability to about *4.5* metres, *80%* probability with W-KNN.



(a) RH test bed.



(b) Cam test bed.

Fig. 6.7 The performance accuracy of the fingerprints with the estimated RSS.

6.4 Summary and further work

This chapter tackles the missing APs problem, which happens when the WiFi RSS are collected in real time. A method to identify the potential missing APs and how to recover them was outlined. The approach used in this paper is entirely programmatic and self-contained. The empirical experiments demonstrated that the higher the number of APs that went missing, the less accurate the positioning estimation became. In addition, the stronger the RSS of the missing APs were, the bigger the positioning error would be. The RSS estimation process was demonstrated to successfully recover the RSS for the missing APs to within *20* dBm of the actual RSS. Finally, the chapter highlights the strong improvement of the positioning prediction using these estimated RSS, compared to the situations where the missing APs were ignored. A potential extension of this work is to incorporate more information such as using the whole RSS sequence to spot the potential missing APs, rather than just two consecutive readings.

So far, this thesis has explored the use of the indoor WiFi RSS for location fingerprinting. The next chapter will assess a different use of the WiFi RSS for outdoor positioning, in the form of co-localisation.

Chapter 7

Outdoor co-localisation with the mobile phone and the public WiFi Access Points

"We must never make experiments to confirm our ideas, but simply to control them."

– Claude Bernard (1865).

Mobile phones and the internet have integrated into daily human life. More importantly, the wireless infrastructure has improved significantly in recent years to help transferring the information amongst the devices. People leave real-time digital footprints everywhere. These footprints may be used to track and to study the human contacts. However, this information seems to be largely neglected so far.

In this chapter, the possibility of people co-location detection is investigated through a series of experiments with the outdoor WiFi RSS and the mobile phones. As most people own a smart phone, a contact between two persons can be regarded as a handshake of the two phones. The novel feature is that the proposed method does not require GPS coverage or real-time mapping. Each person only needs to install an Android app which silently records the WiFi RSS in the phone's background. This data will be analysed off-line to detect the potential contacts. Several measurement campaigns in the real world were conducted to verify the feasibility of detecting such co-location.

7.1 Introduction to mobile phones outdoor co-localisation

Co-localisation is the process of identifying whether two persons are in the same position at the same time. If they are co-located, there is a possibility that both devices observe the same signal fingerprint. Given a time-stamp for each recorded signal, these contacts may be detected. This approach is considerably different from the general outdoor localisation, since there is no need to maintain a map, nor require any knowledge of the exact location of the phone at any moment. Instead, each mobile device collects its own signal trait privately through-out the user's journey. These data will be analysed off-line by an administrator to detect the possibility that the users were in the same position at some points during their journey. The key features of this approach are:

(i) Only co-locations are recorded, when the two phones are close together without the need of detecting exactly where they are.

(ii) An Android app running in the phone's background to take care of the recording process. No signals are sent out.

(iii) The users have full control of whether to allowing tracking themselves.

(iv) The tracking process uses the existing outdoor WiFi APs. The proposed idea has been tested in challenging, uncontrolled city environments.

Theoretically, it is possible to invert the proposed system to have the WiFi APs track the mobile phones. This structure has the advantage of requiring no additional code on the phones, at the expense of a higher processing load on the APs, which may have to track a huge number of users simultaneously. In addition to the scalability issue, it is unlikely such changes can be made on the public APs, without permission from the network providers. On the security side, it would be undesirable for the users to be forcibly tracked by the APs. The users' anonymity is broken, because he can be identified by his GSM mobile SIM card number or the phone's WiFi MAC address. In contrast to the original approach, the users can simply stop the tracking app without disrupting normal WiFi usages.

7.2 The applications of WiFi outdoor co-localisation

Some potential practical applications of mobile phone co-localisation include epidemic tracking, with a network of registered participants. Each patient uses his mobile phone to input his current symptoms, which are managed by a server. At any time, the system can work out the probability that the user may be infected. To discover the origin of an unknown disease, the doctor can backtrack the patients' contact history with other registered people in the same system.

In 2011, the FluPhone project was launched to track the spreading of a disease during a pandemic [96]. It captures the physical proximity information of other nearby users via Bluetooth. Both of the proposed system in this chapter and FluPhone aimed to provide mobile phone localisation for epidemic tracking. However, there are two key differences. Firstly, this chapter do not record the physical location of the users. Secondly, while FluPhone uses GPS and the Bluetooth signal to discover nearby handsets in real-time, the proposed approach analyses the off-line signal data to discover such contact.

Outdoor vehicles' tracking without GPS has also been reported [43, 57, 76]. These systems made use of the cellular network or WiFi to provide the positioning service in the absence of GPS. However, they all used the signal strength from the station to work out the distance to the vehicle. Whereas, the proposed approach in this chapter looks at the pattern of the WiFi RSS to consider if two devices are in close proximity.

7.3 The wireless signals candidates

Signal availability is the most important criterion to decide what type of wireless signal to be used. Ideally, the signal candidate should be available in both the indoor and outdoor space for maximum coverage. Two stand-out candidates to be considered are the WiFi signal, and the cellular signal.

The cellular signal coverage has increased significantly in recent years, thanks to the deployment of more and more phone towers [39]. However, the Android *NeighboringCellInfo* class used to access the cellular information is phone-dependent and network-dependent. At the time of testing, many Samsung phones did not work.

The WiFi networks are popular indoors, but were not so popular outdoors a few years ago. In recent years, the communication service companies such as BT and Vodafone have been actively set up new outdoor WiFi APs to provide the wireless services to their customers on the streets. In particular, over *5* millions outdoor WiFi APs were estimated to be available in the UK in *2012*. In the conducted experiments, there were always at least *10* available APs at any experimented position in London (see Figure 7.1). Many areas in the central London had more than *40* available APs. Another example is the commercialised Skyhook project[1], which provides a world-wide WiFi RSS to physical location mapping to

[1]http://www.skyhookwireless.com - last accessed in Sep/2016.

alleviate the need of GPS coverage [68].

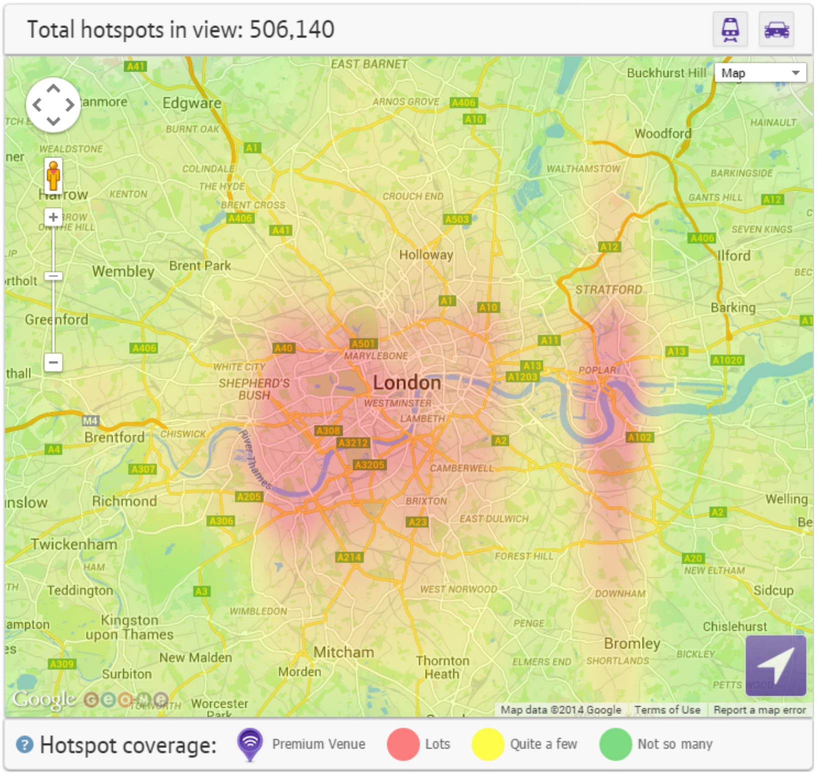


Fig. 7.1 The coverage of BT WiFi hotspots in London, recorded in November 2015.

Some popular wireless signals such as Bluetooth and Infrared have restricted range, and are not available outdoors. Other long-range signals such as radio FM are available outdoors, but require additional decoder on the handset to read them. Such decoders are not widely supported by current smart phones. Thus, the WiFi signal remains the most viable candidate.

7.4 The statistical properties of the outdoor WiFi RSS

In this section, two Android phones (Google Nexus 4, Samsung Galaxy Y S5360) with an Android app (as described in Chapter 2) were used to collect the WiFi RSS. The experiments in this section were carried out in different outdoor locations to assess the following criteria, which are important for the feasibility of outdoor co-localisation.

(i) The WiFi RSS vector recorded from any two co-located phones are similar.

(ii) The WiFi RSS vector recorded at different positions are distinguishable.

While the majority of the literature investigated the WiFi RSS for indoor APs [41, 42, 48], the experiments in this chapter looked at the signal properties of the outdoor APs for two reasons. Firstly, the outdoor environment changes much faster and is more volatile than the indoor counterpart, because of the dynamic of the people movements. Secondly, the public outdoor APs are commercial long-range transmitters that offer further broadcasting range than the indoor ones.

7.4.1 The similarity of the WiFi RSS in fixed positions

When the two phones are co-located, it is assumed that they should hear the same RSS from nearby APs. Figure 7.2 demonstrates the histogram distribution of the RSS from the same WiFi AP, when the two phones were positioned next to each other in a static position. In total, *6,130* WiFi RSS were collected in half a day. In this experiment, the RSS range was as wide as *20* dBm, which contrasts the *10* dBm interval reported for indoor WiFi RSS [41]. The maximum and minimum readings observed from the Google Nexus phone were *-60* dBm and *-77* dBm respectively, while they were *-56* dBm and *-75* dBm for the Galaxy Y phone. However, the majority of signals concentrated around the most occurred RSS for both phones. Of all the histograms being observed at different times, all of them had a near Gaussian left-skewed distribution.

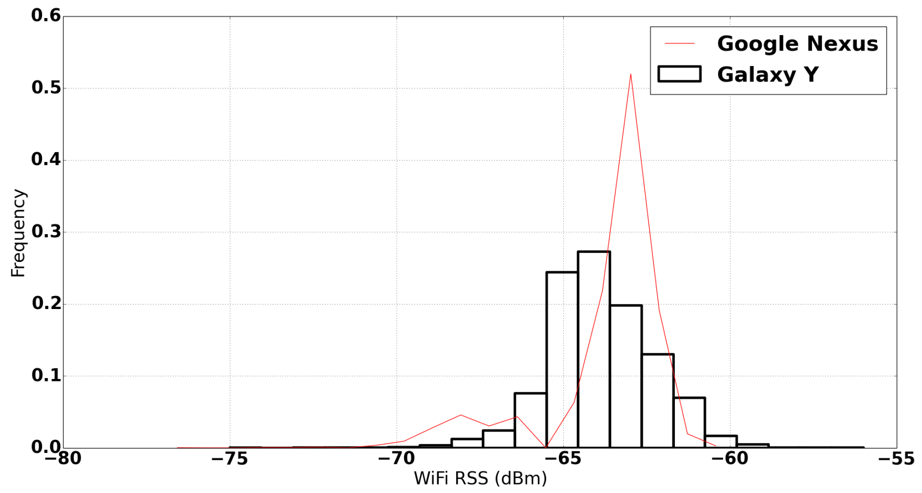


Fig. 7.2 Outdoor WiFi RSS distribution histogram.

Despite the *20* dBm variation over half a day of each phone, by comparing the signal reading of the two phones at any moment, the maximum difference was found to be just *11* dBm (see Figure 7.3). The signal difference was noticeably small during night time, and got bigger by lunch time.

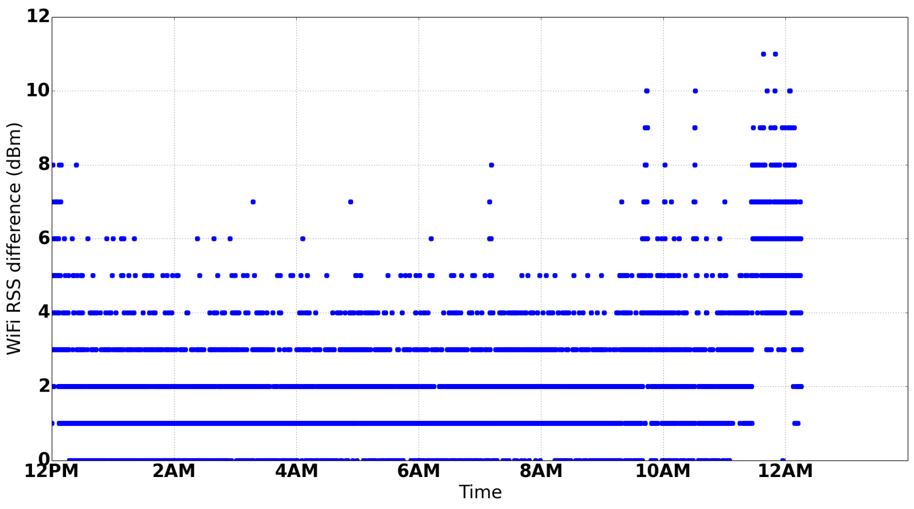


Fig. 7.3 WiFi RSS difference between the two phones over times.

Out of all the observations, *91%* of the signal difference between the two phones was less than *4* dBm. They had the exact same RSS reading for *16%* of the time. The majority of the signal difference was just *1* dBm, *34%* of the time (see Figure 7.4).

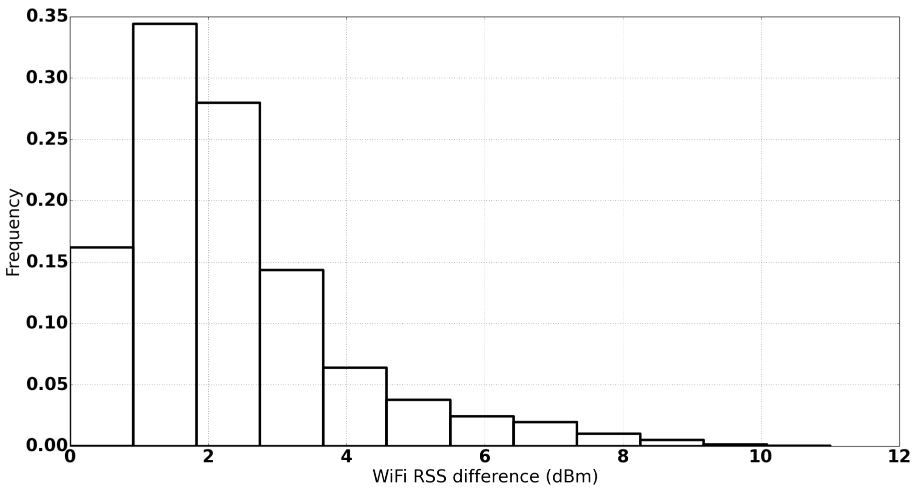


Fig. 7.4 Percentage of RSS difference between the two phones.

With the same data, in terms of the number of discovered APs, the Google Nexus phone consistently observed *9* to *10* APs per scan, while that number varied from *5* to *10* for the Galaxy Y phone. However, both models found *10* APs for most of the time in this experiment (see Figure 7.5).

These experiments suggested that the RSS from more than one APs should be considered when comparing the two RSS vectors. Despite the large *20* dBm variation of the outdoor WiFi AP over half a day, an instant snapshot of the two RSS of the same AP from both

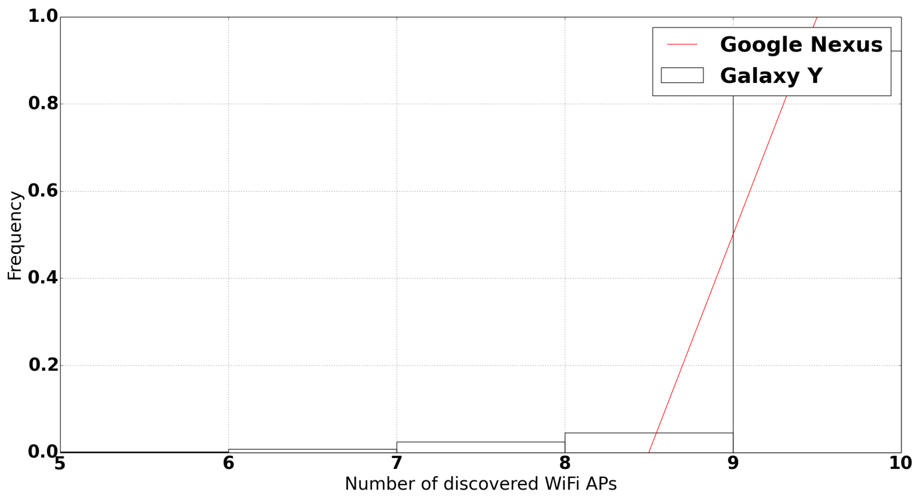


Fig. 7.5 The distribution of the number of discovered APs per scan.

phones has less than *4* dBm difference, *91%* probability.

7.4.2 The variation of the WiFi RSS at different positions

When the distance between the mobile phone and the AP increases, the RSS becomes weaker. This feature is important to distinguish two non-colocated signal vector. The remaining question is how far should the two handsets be, so that their signal readings can be clearly distinguishable. Figure 7.6 demonstrates the correlation between the WiFi RSS and the distance from the phone to the AP. The starting point is the location where the strongest RSS can be heard. The phone was then moved away in a straight line from that position, and stopped approximately every *30* metres to measure the RSS.

It was observed that the signal variation was high for stronger RSS, and became smaller as the signals faded away. The WiFi signal was overwhelming amongst all the ambient noises at a distance of about *170* metres from the initial position. This result shows a much greater range of the industrial outdoor AP, compared to the short *15* to *20* metre distance of an indoor AP. This experiment suggests an approximate *5* to *10* dBm difference between two consecutive locations of *30* metres apart, assuming they are in a relative straight line from the AP.

7.4.3 The correlation of the WiFi RSS at the same AP

If the environment is stable, it is assumed that the WiFi RSS should not change over time. To verify if there is any signal interference amongst the WiFi RSS, this experiment applies

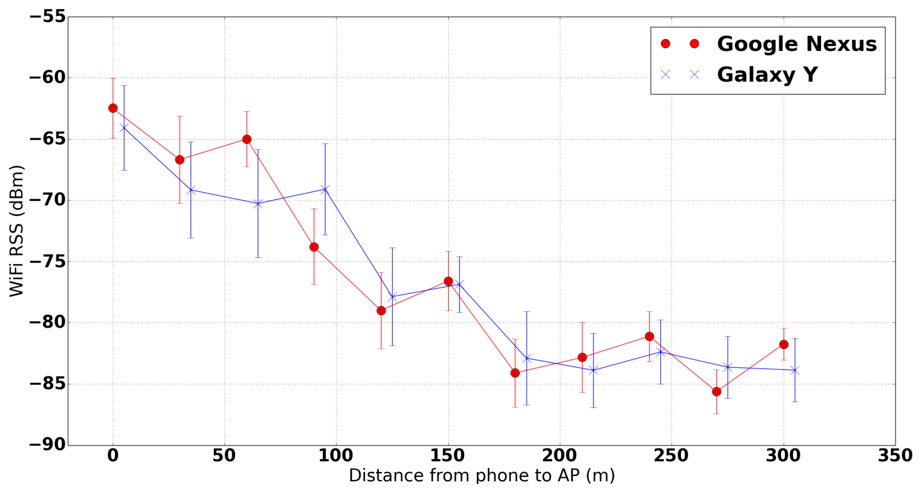


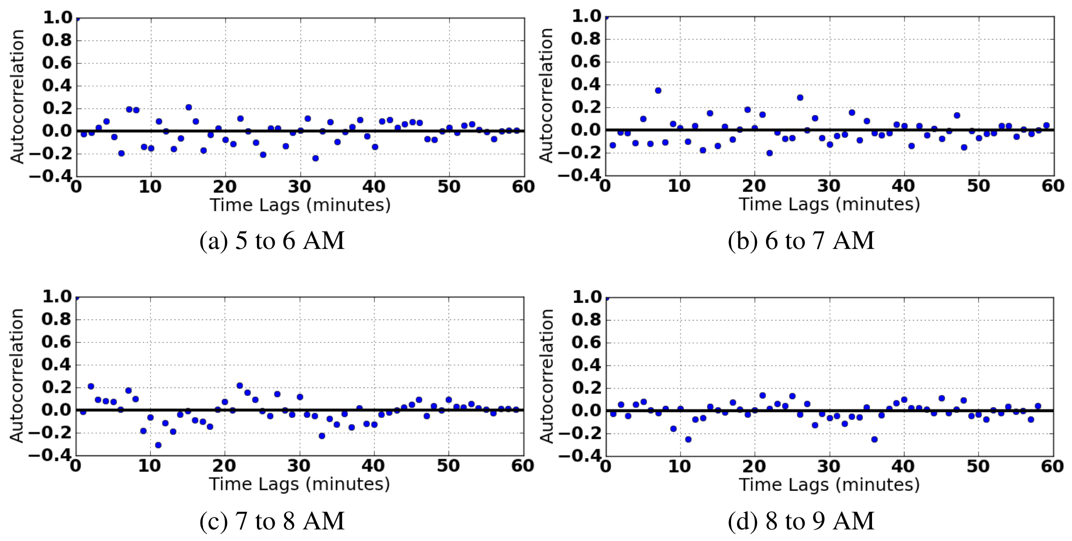
Fig. 7.6 WiFi RSS large scale variation.

the Wide-Sense Stationary (WSS) process to verify the stationary property of the RSS at different times for the same AP [69]. Two conditions must be met for a signal reading to be stationary. Firstly, the mean value of the signal distribution is a constant. Secondly, the autocorrelation function of the WSS process is independent of the time difference at the same AP. The WiFi RSS were recorded in a static location near Waterloo station in London, over *4* hours in the early morning, when the environment was relatively stable. The signal sequence was divided into chunks of *1* hour to compare the segments. Table 7.1 shows that the mean and standard deviation of the four sequences were similar, which satisfies the first WSS condition. Despite the effort to eliminate the ambient noises, there may be some minor changes in the environment, which explains the small differences in the statistical values.

Table 7.1 WiFi RSS correlation at the same AP.

	5-6am	**6-7am**	**7-8am**	**8-9am**
Mean	-68.3 dBm	-68.9 dBm	-68.9 dBm	-68.8 dBm
Standard Deviation	2.74	2.48	2.68	2.35

Figure 7.7 demonstrates the *4* correlograms of 1-hour segment. Interestingly, although the mean and the standard deviation values of the *4* segments were similar, the correlograms suggested that the individual signal was mostly independent. There were some correlations among the consecutive signals in the early hours (see Figures 7.7a, 7.7b, 7.7c). However, the last hour's correlogram was non-correlated because of the signal variation caused by more people going to work (see Figures 7.7d).

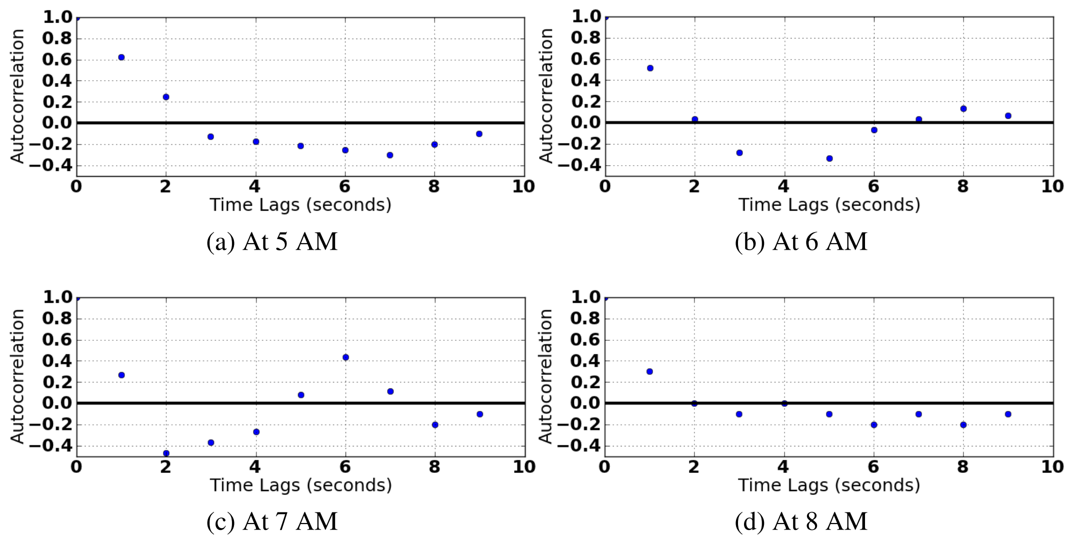


(a) 5 to 6 AM (b) 6 to 7 AM

(c) 7 to 8 AM (d) 8 to 9 AM

Fig. 7.7 1-hour signal correlograms at the same AP.

Although the correlograms illustrates that the WiFi RSS seemed to be random over a long period of time, Figure 7.8 demonstrates that the RSS were correlated in just 10 seconds.



(a) At 5 AM (b) At 6 AM

(c) At 7 AM (d) At 8 AM

Fig. 7.8 10-seconds signal correlograms at the same AP.

This experiment was repeated with different segment sizes of *10* minutes, *15* minutes and *30* minutes to confirm the same results. The results suggested that the outdoor signals are independent for a long time interval. This finding is important to assure that the two signal readings at two different times are distinct and non-related. However, the signals are stable and correlated in a short period of time, which means it is possible to match two signal readings in a short interval with good accuracy, without having them precisely co-located.

7.4.4 The correlation of the WiFi RSS at different APs

There are just *13* WiFi channels in which many devices are operating on. Therefore, the signal interference is likely to happen. For this experiment, the WiFi RSS were collected from three APs at the same time in *1* hour. Each AP was at least *10* metres away from the others. Table 7.2 shows a strong distinction of the mean and the standard deviation for each AP, from the handset's location.

Table 7.2 WiFi RSS's mean and standard deviation from different APs.

	Mean	**Standard Deviation**
AP 1	-69.65 dBm	3.98
AP 2	-84.9 dBm	2.7
AP 3	-89.1 dBm	1.67

Each pair of signal sequences from the *3* APs are compared using the Pearson formula to find any correlation caused by the signal interference. Table 7.3 shows that all correlation co-efficients are negative (+1 means fully related, and -1 means non-related), which confirms that the WiFi RSS from different stations are independent. The IEEE 802.11 protocol may have helped to minimise the interference amongst the APs on the same channel. This experiment suggests that the WiFi RSS from multiple stations may be considered at the same time to better match the signal vectors.

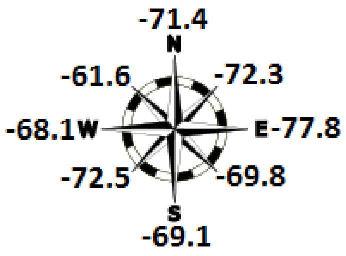
Table 7.3 WiFi RSS correlation amongst different APs.

	Correlation co-efficient
AP 1 - AP 2	-0.198
AP 1 - AP 3	-0.292
AP 2 - AP 3	-0.012

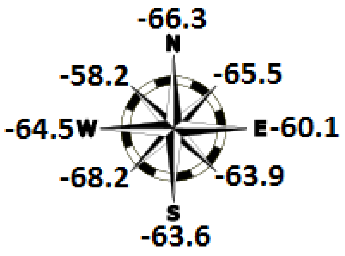
7.4.5 Influences by the human body on the WiFi RSS

The majority of the human body are water, which is a strong signal attenuator. Since the mobile phone is often held close to the body, and in most cases, inside the trousers' pocket, the WiFi RSS may be affected. This section analyses the influences of the human body on the WiFi RSS. Firstly, the RSS difference, based on how the phone is held, is investigated. Most smart phones are equipped with an omni-directional antenna, which can broadcast and receive the signals in full *360* degrees. A mobile phone was set up on a flat table in a clear outdoor area. The phone was then rotated to each of the *8* cardinal directions (N, NE, E, SE, S, SW, W, NW). For each direction, it measured the WiFi RSS several times for a few seconds when there was nobody near the table. The experiment was repeated with a person holding the phone.

Figure 7.9 suggested that although the human presence did have an impact on the signal, the RSS of each direction was comparable in both cases. There was up to *6* dBm difference in the RSS with the human presence. There was no linear relationship among the signals in different directions. Thus, to reliably compare two RSS vectors, each measurement should be accompanied by a cardinal direction which is provided by the compass sensor of the mobile phone.



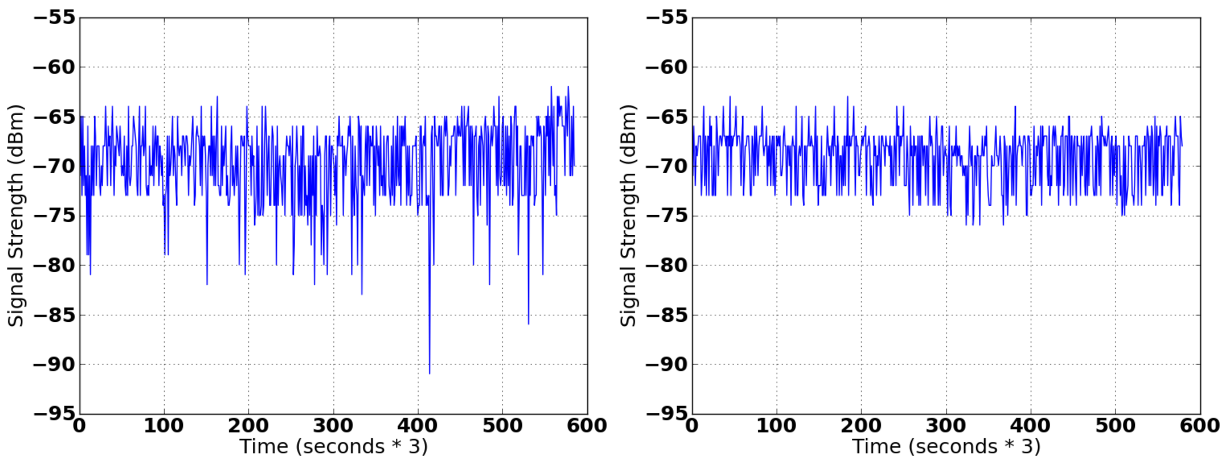
(a) With human presence



(b) Without human presence

Fig. 7.9 WiFi RSS (dBm) in different orientations with/without the human presence.

The second experiment investigated the stability of the WiFi RSS over *30* minutes in Bond Street, a crowded street in central London during rush hours, and in the same street at the same location in the early hours with fewer people. The signal was collected every *3* seconds. Figure 7.10 demonstrates a strong signal variation during the rush hours with a standard deviation of *3.98*, compared to *2.86* of that during the early hours. There is no simple solution for such phenomenon, because the environment changed drastically in a short period of time. The experiment suggested that the WiFi RSS cannot be compared reliably in the same position in a crowded environment in different times.



(a) Rush hours with many people (b) Early hours with fewer people

Fig. 7.10 WiFi RSS during and before the rush hours.

7.5 Handling the problem of heterogeneous mobile devices

The mobile phone manufacturers implement their own WiFi antenna and chip, which may potentially affect the receiving RSS on the mobile device. For example, in the experiments, the Nexus phone was a newer phone with a bigger antenna than the old Samsung Y, thus it was able to discover more APs. The attempts to tackle the heterogeneous devices issue in the literature can be broadly split into two categories, which are calibration-based and algorithm-based. With the first approach, the new device is calibrated manually or automatically within the tracking system, using known landmarks in the building [53]. With the algorithm-based approach, some authors used a linear transformation model and kernel estimation [30, 67], while the others compared two pairwise signal vectors directly [39, 46]. Since the chapter's goal is off-line co-localisation detection, the WiFi RSS of each mobile device are simply normalised so that they are directly comparable.

Without loss of generality, given an RSS vector representing the WiFi RSS of N nearby WiFi APs, $\overrightarrow{RSS} = (s_1, \ldots, s_N)$, with s_i is the RSS (dBm) observed from AP_i. Each individual s_i is divided by the strongest RSS s_{max} observed in the whole journey.

$$\overrightarrow{RSSI_{norm}} = (\frac{s_1}{s_{max}}, \frac{s_2}{s_{max}}, \ldots, \frac{s_N}{s_{max}}) = (s'_1, s'_2, \ldots, s'_N) \ . \tag{7.1}$$

7.6 Two algorithms to calculate the matching rate of two WiFi RSS vectors

Given two WiFi RSS traits from two phones which represent the journey of two persons, the system identifies at which parts of the journey they were co-located. To achieve this goal, it needs a method to calculate the similarity of two WiFi RSS vectors. This section outlines two algorithms to compute such measure.

The first algorithm is based on the appearance of the APs alone. The benefit of this algorithm is that it ignores the magnitude of the WiFi RSS, which may be highly fluctuating due to the harsh outdoor environment, as discussed earlier. The two WiFi RSS vectors *A* and *B* are considered fully matched, or very similar, if they have the same number of recorded APs, and for every AP recorded by Phone 1, it is also observed by Phone 2, and vice-versa. If the two signal vectors share some APs in common, but also contain their own discovered APs, the similarity is calculated as the number of overlapped APs, divided by the total number of APs. This measure will be called the matching rate, and the closer it is to *100%*, the more similar the two signal vectors are. The idea of this measure comes from the Jaccard index [72].

$$matching\_rate\_AP(A,B) = \frac{\#(AP \in A \& AP \in B)}{\#(AP \in A || AP \in B)} \quad . \tag{7.2}$$

The above approach worked relatively well when a large number of APs was observed. However, the measure was not high when the number of nearby APs was low. With the second algorithm, the magnitude of the WiFi AP is considered. The assumption is that the overlapped RSS from the same APs that any two co-located devices share should be strong. If one device sees a strong AP, while the other does not, they are unlikely to be co-located. Without loss of generality, given the WiFi RSS vector of Phone 1: $\overrightarrow{A} = (s_1, \ldots, s_n)$, and Phone 2: $\overrightarrow{B} = (s'_1, \ldots, s'_m)$, the matching rate is calculated as follows, with AP_i and AP'_i are the AP observed by Phone 1 and Phone 2 respectively.

$$matching\_rate\_RSS(A,B) = \frac{\sum_{i=1}^{n} s_i + \sum_{j=1}^{m} s'_j, AP_i, AP'_j \in A \cap B}{\sum_{i=1}^{n} s_i + \sum_{j=1}^{m} s'_j, AP_i, AP'_j \in A \cup B} \quad . \tag{7.3}$$

A high measure means the two devices are close, and a low one means they are further away. Ideally, the matching measure should be closely correlated to the actual physical distance between the two devices. The above two measures will be evaluated and compared to the ground-truth GPS distance in the next section.

7.7 Evaluation of performance

In the first experiment, two persons walked side by side, with a mobile phone in their pockets. The two phones were synchronised to invoke WiFi scanning simultaneously every *30* seconds. When called, each phone initiates *3* continuous scans. There were *1,643* APs recorded in an hour journey. Since the two phones were side by side, it was expected that all *253* recorded readings to be *100%* matched. However, none of the reading pair was fully matched at any moment through-out the one hour journey using both of the above two measures. This problem was caused by the WiFi APs being missed during a scan, which has been discussed in Chapter 7. However, since this chapter is not using any training database like fingerprinting, and it is not estimating the user position to discover the missing APs, as suggested in the last chapter, a simpler approach was used to alleviate the issue, that is, three continuous scans were combined as follows. Since there is a delay in scanning the APs, it is challenging to perform many scans within a short period of time.

(a) The RSS of the same APs within *3* continuous scans are averaged.

(b) If a new AP is found within *3* continuous scans, it is added into the signal vector.

(c) All APs with very weak signal ≤ *-90 dBm* are removed.

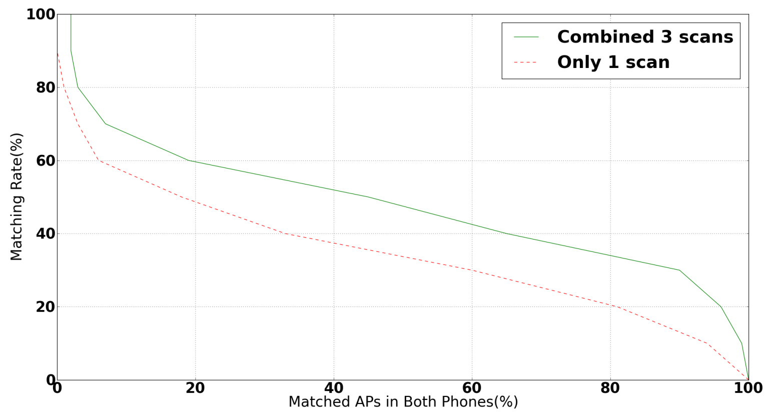


Fig. 7.11 Improvement in the matching rate measure based on the AP appearance by combining three continuous scans.

With this approach, the number of samples with *100%* matching rate was increased from *0%* to *2%*, and the same number for *50%* matching rate was increased from *18%* to *45%* based on the AP appearance. The majority of WiFi RSS vectors' pair were around *40%*

matched with this approach (see Figure 7.11). A real-time visualisation of this experiment can be viewed on-line[2].

For the second experiment, two persons started in the same position. They walked in different directions at *14:42* on the time line, then re-joined halfway at *14:56*. The journey finished at *15:01*. All other settings were kept the same as the first experiment. Figure 7.12 demonstrates the appearance of the top *10* commonly shared APs for the whole journey.

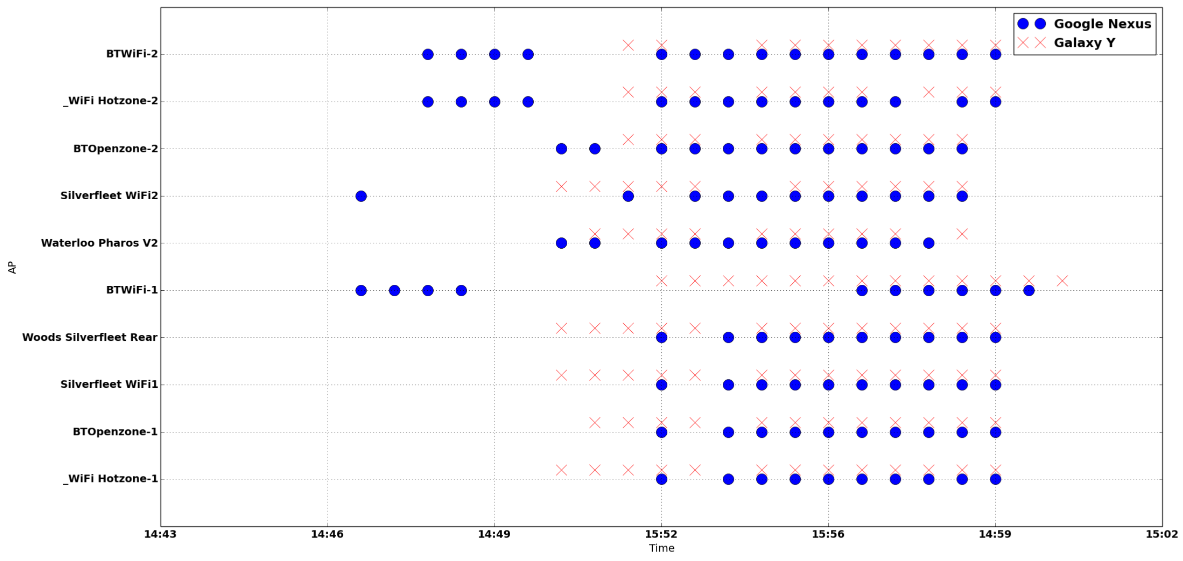


Fig. 7.12 The density of AP appearance over times seen by the two phones.

The matching rate started off positively at the beginning, when both phones were together (see Figure 7.13). As they went off in different routes, the matching rate decreased. Figure 7.14a demonstrates that the phones still observed some similar APs, although they were further apart, because of the outdoor open space. However, the observed RSS was weak for these similar APs, as a result. The matching rate increased again as the phones approached each other, and remained stable as they re-joined (see Figure 7.14b). Both phones observed many strong RSS, when the matching rate was high. A real-time visualisation of this experiment on Google Maps can be viewed on-line[3].

After applying the above method to combine the WiFi scans, there were *412* APs recorded by the Google Nexus phone, and *554* APs recorded by the Galaxy Y phone. Of *252* APs found by both phones during the *20* minute journey, the highest number of appearance of a single AP was *15* for Google Nexus and *13* for Galaxy Y (see Table 7.4).

[2]http://www.cs.rhul.ac.uk/~wruf265/Map
[3]http://www.cs.rhul.ac.uk/~wruf265/Map2

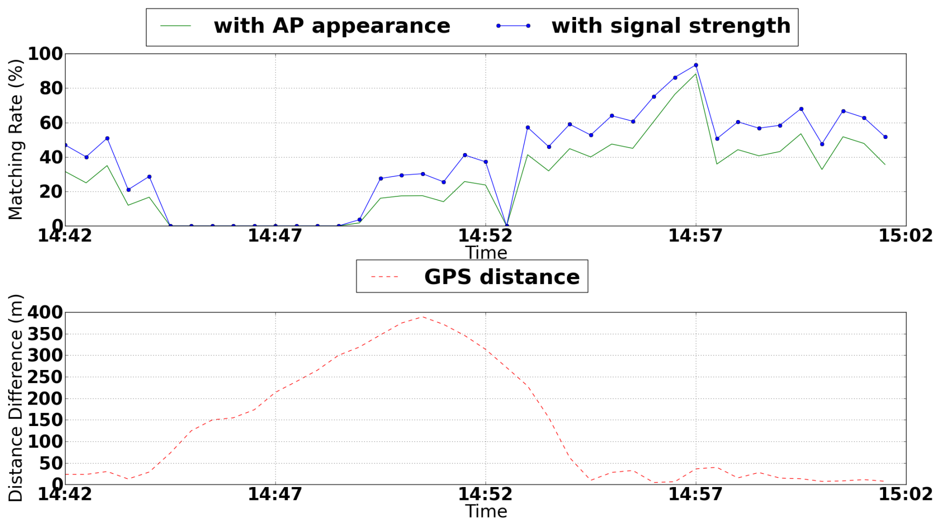
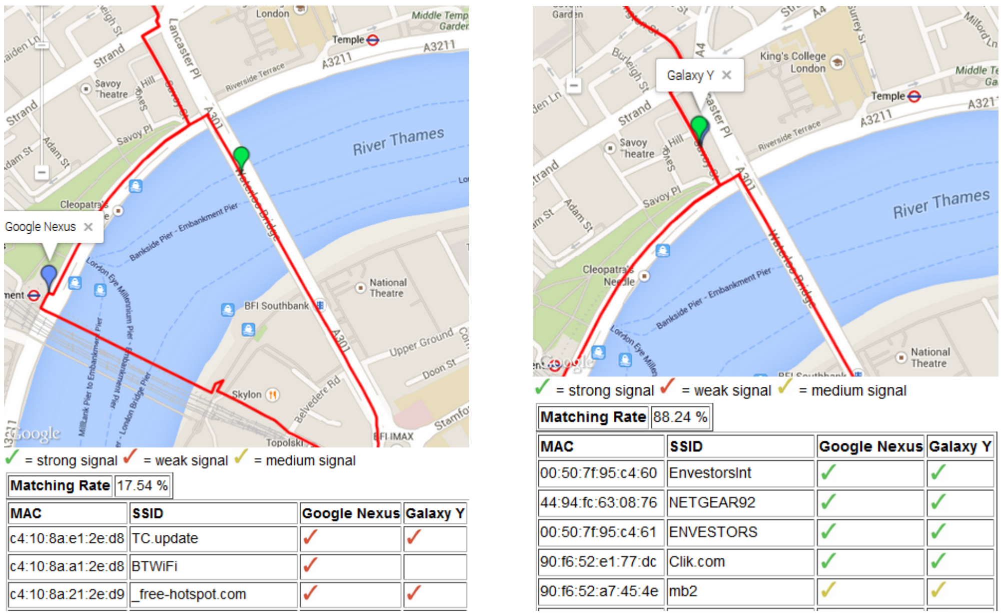


Fig. 7.13 Matching rate vs. GPS distance.



✓ = strong signal ✓ = weak signal ✓ = medium signal

Matching Rate	17.54 %

MAC	SSID	Google Nexus	Galaxy Y
c4:10:8a:e1:2e:d8	TC.update	✓	✓
c4:10:8a:a1:2e:d8	BTWiFi	✓	
c4:10:8a:21:2e:d9	_free-hotspot.com	✓	✓

(a) Low matching rate for separated phones

✓ = strong signal ✓ = weak signal ✓ = medium signal

Matching Rate	88.24 %

MAC	SSID	Google Nexus	Galaxy Y
00:50:7f:95:c4:60	EnvestorsInt	✓	✓
44:94:fc:63:08:76	NETGEAR92	✓	✓
00:50:7f:95:c4:61	ENVESTORS	✓	✓
90:f6:52:e1:77:dc	Clik.com	✓	✓
90:f6:52:a7:45:4e	mb2	✓	✓

(b) High matching rate for co-located phones

Fig. 7.14 The matching rates of mobile phones co-localisation visualised on Google maps.

Table 7.4 APs summary of the two phones.

	Google Nexus	**Galaxy Y**
Total number of scans	120	120
Total recorded APs	504	558
Remaining APs after processing	412	554
Shared APs	252	252
Highest number of single AP appearance	15	13

7.8 Summary and further work

This chapter has demonstrated the feasibility of using the mobile phones and the outdoor public WiFi APs for the purpose of off-line co-location detection. An Android app was used to silently collect the WiFi RSS while the users travel. This data can be matched off-line to spot the RSS similarity, which confirms if the users were in close proximity. The chapter has described two algorithms to calculate such similarity based on a measure called the matching rate. This measure can be calculated with just the number of available APs due to the high number of outdoor APs, or incorporated with the RSS. Several measurement campaigns in the real world were carried out to assess the matching rate measure with the ground-truth GPS data. The empirical results demonstrated that the proposed measure closely reflect the GPS distance, and the possibility of using the outdoor WiFi APs for outdoor co-localisation is positive.

The calculation of the matching rate in this chapter was fairly basic, and was just sufficient to demonstrate the feasibility of the proposed idea. A potential extension of this work is to research other methods to compute such measure. The author has been in the process of applying this idea to the epidemic tracking research to predict in advance the possibility of a future outbreak.

Chapter 8

Conclusions and further work

"Endings to be useful must be inconclusive."

– Samuel Delany.

In this chapter, the results obtained individually from the previous chapters are summarised and assessed in relation to the research objectives set out earlier. The chapter concludes by discussing the possible avenues for further research in fingerprinting.

8.1 Summary of the main contributions

Fingerprinting has been a remarkably popular method for infrastructure-free indoor positioning in the past decade, and looks here to stay. A great deal of effort has been invested to improve the performance of fingerprinting. However, it is commonly expected that the positioning error of WiFi fingerprinting to be around room level on average, and may not be able to consistently achieve sub-metre error under real-world conditions. This thesis tackles the challenges of WiFi fingerprinting with machine learning algorithms and additional information beside the WiFi RSS. The techniques described in this thesis are flexible and can be adapted to most existing fingerprint-based systems. With respect to the research questions posed in Chapter 1, the thesis identifies the following contributions.

(i) This thesis has demonstrated that the WiFi RSS alone is too noisy for fine-grained fingerprinting. It suggested to incorporate the magnetic field to assist the WiFi RSS. The use of the magnetic field in this thesis is original in the sense that it merges the MFV into a sequence to represent a walking trajectory. The novel feature of the system is the ability to predict in advance the route, and the potential destination for the users. The thesis tested the proposed idea in a real indoor environment to demonstrate that it did not only successfully estimate the route and the destination, but also achieved a single positioning prediction error of *2.2* metres, *95%* probability, compared to *3.5* metres, *95%* probability when only the WiFi RSS of the fingerprinting database were used.

(ii) This thesis introduces a novel supervised confidence learning algorithm, that is, to the best of the author's knowledge, the first approach to provide a confidence measure for the fingerprinting research. This confidence measure is not only useful in reflecting the uncertainty of the positioning predictions, but is also capable of delivering a prediction set or a prediction region. Its performance has been empirically illustrated to be around *15%* to *20%* more accurate than the traditional W-KNN and Naïve Bayes under the same test domains.

(iii) While most clustering-based fingerprinting approaches used only the WiFi RSS to structure the training data, this thesis realised that the clustering process should be applied as a supervised learning problem for fingerprinting. It proposes a novel two-level soft clustering process that involves both the WiFi RSS and the Cartesian label. Furthermore, the proposed clustering algorithm tackles the signal borderline challenge by allowing the training examples to be associated with more than just one cluster.

The empirical results demonstrated that the proposed scheme did not only reduce the searching time, but also improved the positioning accuracy.

(iv) The thesis addresses the missing APs problem in the real-time WiFi scan, which has been conveniently overlooked in the fingerprinting literature. It proposes a scheme to detect and estimate the value of the missing APs using only the WiFi RSS from the previous position. The empirical results showed that the proposed scheme successfully recovered the RSS value for the missing APs to within *20* dBm of the true RSS. Compared to the common approach where the missing APs are simply given a *-100* dBm value, the proposed scheme reduced the positioning error up to *40%*.

(v) Lastly, this thesis presents the results of a series of experiments with the data collected through-out several measurement campaigns in the real world to assess the feasibility of using the public outdoor WiFi APs and the mobile phones to co-locate the people's position. It describes an algorithm to calculate the matching rate of two WiFi RSS vectors, and compares the results with the ground-truth GPS distance to demonstrate their similarity.

8.2 Further work

This section outlines five general directions to extend fingerprinting further. The detailed extension of the proposed idea for each chapter has been discussed separately at the end of the according chapter, and will not be repeated here.

8.2.1 Extra information from a site map

Although the approaches presented in this thesis do not require a site map of the building at all, such map can be combined to provide extra ground-truth information to remove the violated predictions, where the user walks through the walls. This has been a popular idea with robot-based Simultaneous Localization And Mapping (SLAM) [24, 36, 91]. The building map is also often available for most buildings. However, the challenge of the system is to correctly map the positioning prediction co-ordinate to the relevant point on the map.

8.2.2 Scalable fingerprinting

Although the proposed clustering technique in Chapter 3 helps to alleviate the computational overhead to process the training database, the scalability of the whole system still partly depends on the total number of active users. Some potential further work to lift this burden completely out of the system is providing the user with a portion of the training database to work out his current position by himself. The mobile clients will communicate amongst themselves to share their own portions of the training database they need when moving to a new location. Thus, the central server only has to serve the clients at the beginning when everyone first enters the building.

8.2.3 Beyond the received signal strength

The WiFi RSS is noisy since it was not designed with the localisation purpose in mind. Some recent works started to delve into the PHY layer below to obtain the Channel State Information (CSI). This measure was reported to be as robust as the magnetic field strength, without losing the long-range property of the WiFi signal. Sub-metre tracking systems have been reported [17, 74, 94]. However, CSI fingerprint-based systems fall into the infrastructure-based category, because they require a custom WiFi driver to be installed on the WiFi APs to allow capturing of the CSI data, which is a real challenge for real-world implementation. Furthermore, only one WiFi adapter - the Intel NIC 5300 is currently supported by the custom firmware. The major disadvantage of using CSI is that only one AP can be inquired at a time, which is not practical unless the user is static. Nevertheless, the information on the physical layer provides much more insights into the distance of the user than the RSS, and future CSI developments may reduce the above limitations.

8.2.4 Different means to perform fingerprinting

The smart phone is a ubiquitous device that most people carry with them. It is small and contains all popular sensors. In recent years, it is getting even smaller, in the forms of the smart watch, the smart glass and the smart bracelet. In addition, they are able to operate alone without a Bluetooth or a WiFi connection to the phone. Thus, the users have more options to access the positioning service. More importantly, these new technologies change the way people hold and use them. While the phones are normally restricted in the user's pocket, the smart watch and smart bracelet can only be worn on the user's wrist, thus, their movements are more predictable than the phone's. The smart glass, on the other hand, is always next to the user's eyes. Thus, it can observe the direction the user is looking at, as well as being able to capture the full image of the surroundings. These new wearable devices do not only provide new ways to collect data, but also deliver additional information that may be useful for fingerprinting.

8.2.5 Reinforcement learning with the user's feedbacks

A highly adaptable system should be able to improve itself over time. With reinforcement learning, the system changes its model based on the rewards received from time to time to maximise its performance. The challenge to apply reinforcement learning for fingerprinting is that there is often no such rewards telling the positioning system of how well it performs at any stage. A possible means to introduce such rewards is by asking the users directly. At the end of the journey when the user has reached his destination, he is asked to rank the system's performance accuracy. Based on this feedback, the system's action is to prioritise the training examples that lead to a good feedback, and avoids those that result in bad feedbacks. However, the limitation of this approach is that the expectations from the users are different. Some users are more tolerant than others, hence, they tend to give higher rewards. This challenge can be addressed by a questionnaire, or by asking simple 'yes/no' question (e.g. Are you inside this room?) that help the system uniformly quantifies the answers from all users. Furthermore, it may be beneficial to group the answers from the same user over time, because his feedbacks are expected to be consistent.

References

[1] Erwin Aitenbichler and Max Mühlhäuser. An ir local positioning system for smart items and devices. In *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on*, pages 334–339. IEEE, 2003.
Cited on page 6.

[2] Bulut Altintas and Tacha Serif. Improving rss-based indoor positioning algorithm via k-means clustering. In *Wireless Conference 2011-Sustainable Wireless Technologies (European Wireless), 11th European*, pages 1–5. VDE, 2011.
Cited on pages 38 and 44.

[3] Rick L Andrews and Imran S Currim. A comparison of segment retention criteria for finite mixture logit models. *Journal of Marketing Research*, 40(2):235–243, 2003.
Cited on page 45.

[4] Martin Azizyan, Ionut Constandache, and Romit Roy Choudhury. Surroundsense: mobile phone localization via ambience fingerprinting. In *Proceedings of the 15th annual international conference on Mobile computing and networking*, pages 261–272. ACM, 2009.
Cited on page 7.

[5] Paramvir Bahl and Venkata N Padmanabhan. Radar: An in-building rf-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 775–784. Ieee, 2000.
Cited on pages 5 and 14.

[6] Tony Bellotti, Zhiyuan Luo, Alex Gammerman, Frederick W Van Delft, and Vaskar Saha. Qualified predictions for microarray and proteomics pattern diagnostics with confidence machines. *International Journal of Neural Systems*, 15(4):247–258, 2005.
Cited on page 58.

[7] Gennady Berkovich. Accurate and reliable real-time indoor positioning on commercial smartphones. In *Indoor Positioning and Indoor Navigation (IPIN), 2014 International Conference on*, pages 670–677. IEEE, 2014.
Cited on page 7.

[8] James C Bezdek. *Pattern recognition with fuzzy objective function algorithms*. Springer Science & Business Media, 2013.
Cited on page 44.

[9] Christopher M Bishop. Pattern recognition. *Machine Learning*, 128, 2006.
Cited on page 39.

[10] Daren C Brabham. Crowdsourcing as a model for problem solving an introduction and cases. *Convergence: the international journal of research into new media technologies*, 14(1):75–90, 2008.
Cited on page 16.

[11] Nirupama Bulusu, John Heidemann, and Deborah Estrin. Gps-less low-cost outdoor localization for very small devices. *Personal Communications, IEEE*, 7(5):28–34, 2000.
Cited on page 5.

[12] Bradford Campbell, Prabal Dutta, Benjamin Kempke, Ye-Sheng Kuo, and Pat Pannuto. Decawave: Exploring state of the art commercial localization. *Ann Arbor*, 1001:48109.
Cited on page 6.

[13] Gilles Celeux and Gilda Soromenho. An entropy criterion for assessing the number of clusters in a mixture model. *Journal of classification*, 13(2):195–212, 1996.
Cited on page 45.

[14] Saad Chaudhry. Indoor location estimation using an nfc-based crowdsourcing approach for bootstrapping. 2013.
Cited on page 16.

[15] Jaewoo Chung, Matt Donahoe, Chris Schmandt, Ig-Jae Kim, Pedram Razavai, and Micaela Wiseman. Indoor location sensing using geo-magnetism. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 141–154. ACM, 2011.
Cited on pages 7 and 92.

[16] Ionut Constandache, Romit Roy Choudhury, and Injong Rhee. Towards mobile phone localization without war-driving. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.
Cited on page 5.

[17] Riccardo Crepaldi, Jeongkeun Lee, Raul Etkin, Sung-Ju Lee, and Robin Kravets. Csi-sf: Estimating wireless channel state using csi sampling & fusion. In *INFOCOM, 2012 Proceedings IEEE*, pages 154–162. IEEE, 2012.
Cited on page 138.

[18] Abhijit Dasgupta and Adrian E Raftery. Detecting features in spatial point processes with clutter via model-based clustering. *Journal of the American Statistical Association*, 93(441):294–302, 1998.
Cited on page 43.

[19] Mikhail Dashevskiy and Zhiyuan Luo. Reliable probabilistic classification of internet traffic. *International Journal of Information Acquisition*, 6(2):133–146, 2009.
Cited on page 58.

[20] Brett Dawes and Kwan-Wu Chin. A comparison of deterministic and probabilistic methods for indoor localization. *Journal of Systems and Software*, 84(3):442–451, 2011.
Cited on pages xvii, 26, and 27.

[21] Genming Ding, Zhenhui Tan, Jinbao Zhang, and Lingwen Zhang. Fingerprinting localization based on affinity propagation clustering and artificial neural networks. In *Wireless Communications and Networking Conference (WCNC), 2013 IEEE*, pages 2317–2322. IEEE, 2013.
Cited on pages 38 and 44.

[22] Norman R Draper and Harry Smith. *Applied regression analysis*. John Wiley & Sons, 2014.
Cited on pages 23 and 24.

[23] Joseph C Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. 1973.
Cited on page 44.

[24] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.
Cited on page 137.

[25] Brian Everitt, Sabine Landau, and Morven Leese. *Cluster Analysis*. Wiley, 2011.
Cited on page 43.

[26] Shih-Hau Fang, Chu-Hsuan Wang, Ting-Yu Huang, Chin-Huang Yang, and Yung-Sheng Chen. An enhanced zigbee indoor positioning system with an ensemble approach. *Communications Letters, IEEE*, 16(4):564–567, 2012.
Cited on page 6.

[27] R Faragher and R Harle. Smartslam–an efficient smartphone indoor positioning system exploiting machine learning and opportunistic sensing. In *ION GNSS*, volume 13, pages 1–14. IEEE, 2013.
Cited on page 89.

[28] Ramsey Faragher and Robert Harle. Location fingerprinting with bluetooth low energy beacons. *Selected Areas in Communications, IEEE Journal on*, 33(11):2418–2428, 2015.
Cited on page 7.

[29] Toni Giorgino et al. Computing and visualizing dynamic time warping alignments in r: the dtw package. *Journal of statistical Software*, 31(7):1–24, 2009.
Cited on page 90.

[30] Andreas Haeberlen, Eliot Flannery, Andrew M Ladd, Algis Rudys, Dan S Wallach, and Lydia E Kavraki. Practical robust localization over large-scale 802.11 wireless networks. In *Proceedings of the 10th annual international conference on Mobile computing and networking*, pages 70–84. ACM, 2004.
Cited on page 128.

[31] F Hammer, M Pichler, H Fenzl, A Gebhard, and C Hesch. An acoustic position estimation prototype system for underground mining safety. *Applied Acoustics*, 92: 61–74, 2015.
Cited on page 6.

[32] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28 (1):100–108, 1979.
Cited on pages 44 and 151.

[33] Janne Haverinen and Anssi Kemppainen. Global indoor self-localization based on the ambient magnetic field. *Robotics and Autonomous Systems*, 57(10):1028–1035, 2009.
Cited on pages 16 and 82.

[34] Weikko Aleksanteri Heiskanen. The earth and its gravity field. 1958.
Cited on pages xiv and 90.

[35] Ville Honkavirta, Tommi Perälä, Simo Ali-Löytty, and Robert Piché. A comparative survey of wlan location fingerprinting methods. In *Positioning, Navigation and Communication, 2009. WPNC 2009. 6th Workshop on*, pages 243–251. IEEE, 2009.
Cited on pages xvii, 21, 26, 27, and 28.

[36] Andrew Howard. Multi-robot simultaneous localization and mapping using particle filters. *The International Journal of Robotics Research*, 25(12):1243–1256, 2006.
Cited on page 137.

[37] Chih-Ning Huang and Chia-Tai Chan. Zigbee-based indoor location system by k-nearest neighbor algorithm with weighted rssi. *Procedia Computer Science*, 5:58–65, 2011.
Cited on page 22.

[38] Ping-Jung Huang, Yu-Chee Tseng, and Kun-Cheng Tsai. A fast handoff mechanism for ieee 802.11 and iapp networks. In *Vehicular Technology Conference, 2006. VTC 2006-Spring. IEEE 63rd*, volume 2, pages 966–970. IEEE, 2006.
Cited on page 108.

[39] Mohamed Ibrahim and Moustafa Youssef. Enabling wide deployment of gsm localization over heterogeneous phones. In *Communications (ICC), 2013 IEEE International Conference on*, pages 6396–6400. IEEE, 2013.
Cited on pages 119 and 128.

[40] Jongdae Jung, Seungmok Lee, Hangeun Kim, Byeolteo Park, and Hyun Myung. Mobile robot relocation using ambient magnetic fields and radio sources. In *Control, Automation and Systems (ICCAS), 2013 13th International Conference on*, pages 1766–1768. IEEE, 2013.
Cited on pages 16 and 92.

[41] Kamol Kaemarungsi and Prashant Krishnamurthy. Properties of indoor received signal strength for wlan location fingerprinting. In *Mobile and Ubiquitous Systems:*

Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on, pages 14–23. IEEE, 2004.
Cited on pages 14 and 121.

[42] Kamol Kaemarungsi and Prashant Krishnamurthy. Analysis of wlan's received signal strength indication for indoor location fingerprinting. *Pervasive and mobile computing*, 8(2):292–316, 2012.
Cited on pages 14 and 121.

[43] Sneha Kamble, Chinmaya Godbole, Rohini Gaikwad, and AP Yadav. Vehicle tracking system. *International Journal for Innovative Research in Science and Technology*, 2(11):415–419, 2016.
Cited on page 119.

[44] Christine Keribin. Consistent estimation of the order of mixture models. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 49–66, 2000.
Cited on page 43.

[45] Seong-Eun Kim, Yong Kim, Jihyun Yoon, and Eung Sun Kim. Indoor positioning system using geomagnetic anomalies for smartphones. In *Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on*, pages 1–5. IEEE, 2012.
Cited on pages 82 and 89.

[46] Mikkel Baun Kjærgaard. Indoor location fingerprinting with heterogeneous clients. *Pervasive and Mobile Computing*, 7(1):31–43, 2011.
Cited on page 128.

[47] Konstantin Klipp, Jonas Willaredt, Helge Rosé, and Ilja Radusch. Low cost high precision indoor localization system fusing inertial-and magnetic field sensor data with radio beacons.
Cited on page 6.

[48] Prashant Krishnamurthy. *Wifi location fingerprinting*. CRC Press, 2013.
Cited on pages 14 and 121.

[49] Ye-Sheng Kuo, Pat Pannuto, Ko-Jen Hsiao, and Prabal Dutta. Luxapose: Indoor positioning with mobile phones and visible light. In *Proceedings of the 20th annual international conference on Mobile computing and networking*, pages 447–458. ACM, 2014.
Cited on pages 5 and 6.

[50] A Lambrou, H Papadopoulos, and A Gammerman. Evolutionary conformal prediction for breast cancer diagnosis. In *2009 9th International Conference on Information Technology and Applications in Biomedicine*, pages 1–4. IEEE, 2009.
Cited on page 58.

[51] Antonis Lambrou, Harris Papadopoulos, Efthyvoulos Kyriacou, Constantinos S Pattichis, Marios S Pattichis, Alexander Gammerman, and Andrew Nicolaides. Assessment of stroke risk based on morphological ultrasound image analysis with conformal prediction. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pages 146–153. Springer, 2010.
Cited on page 58.

[52] Jonathan Ledlie, Jun-geun Park, Dorothy Curtis, André Cavalcante, Leonardo Camara, Afonso Costa, and Robson Vieira. Molé: A scalable, user-generated wifi positioning engine. *Journal of Location Based Services*, 6(2):55–80, 2012.
Cited on page 16.

[53] Minkyu Lee and Dongsoo Han. Qrloc: User-involved calibration using quick response codes for wi-fi based indoor localization. In *Computing and Convergence Technology (ICCCT), 2012 7th International Conference on*, pages 1460–1465. IEEE, 2012.
Cited on pages 16 and 128.

[54] Binghao Li, Thomas Gallagher, Andrew G Dempster, and Chris Rizos. How feasible is the use of magnetic field alone for indoor positioning? In *Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on*, pages 1–9. IEEE, 2012.
Cited on pages 82 and 92.

[55] Kejiong Li, John Bigham, Laurissa Tokarchuk, and Eliane L Bodanese. A probabilistic approach to outdoor localization using clustering and principal component transformations. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International*, pages 1418–1423. IEEE, 2013.
Cited on pages 38 and 44.

[56] Tsung-Nan Lin and Po-Chiang Lin. Performance comparison of indoor positioning techniques based on location fingerprinting in wireless networks. In *2005 International Conference on Wireless Networks, Communications and Mobile Computing*, volume 2, pages 1569–1574. IEEE, 2005.
Cited on pages xiii, 14, 26, and 27.

[57] Henghui Lu, Sheng Zhang, Xingchuan Liu, and Xiaokang Lin. Vehicle tracking using particle filter in wi-fi network. In *Vehicular Technology Conference Fall (VTC 2010-Fall), 2010 IEEE 72nd*, pages 1–5. IEEE, 2010.
Cited on page 119.

[58] Jun Ma, Xuansong Li, Xianping Tao, and Jian Lu. Cluster filtered knn: A wlan-based indoor positioning scheme. In *World of Wireless, Mobile and Multimedia Networks, 2008. WoWMoM 2008. 2008 International Symposium on a*, pages 1–8. IEEE, 2008.
Cited on page 44.

[59] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.
Cited on pages 44 and 151.

[60] Andrew Markham, Niki Trigoni, David W Macdonald, and Stephen A Ellwood. Underground localization in 3-d using magneto-inductive tracking. *Sensors Journal, IEEE*, 12(6):1809–1816, 2012.
Cited on page 6.

[61] Eladio Martin, Oriol Vinyals, Gerald Friedland, and Ruzena Bajcsy. Precise indoor localization using smart phones. In *Proceedings of the international conference on*

Multimedia, pages 787–790. ACM, 2010.
Cited on pages 5 and 89.

[62] Todd K Moon. The expectation-maximization algorithm. *Signal processing magazine, IEEE*, 13(6):47–60, 1996.
Cited on page 40.

[63] Meinard Müller. Dynamic time warping. *Information retrieval for music and motion*, pages 69–84, 2007.
Cited on page 90.

[64] Khuong An Nguyen. Robot-based evaluation of bluetooth fingerprinting. Master's thesis, Computer Lab, University of Cambridge, 2011.
Cited on pages 16 and 30.

[65] Lionel M Ni, Yunhao Liu, Yiu Cho Lau, and Abhishek P Patil. Landmarc: indoor location sensing using active rfid. *Wireless networks*, 10(6):701–710, 2004.
Cited on page 6.

[66] Ralph Niels. Dynamic time warping. *Artificial Intelligence*, 2004.
Cited on page 90.

[67] Jun-geun Park. *Indoor localization using place and motion signatures*. PhD thesis, Massachusetts Institute of Technology, 2013.
Cited on page 128.

[68] Ling Pei, Jingbin Liu, Robert Guinness, Yuwei Chen, Torsten Kroger, Ruizhi Chen, and Liang Chen. The evaluation of wifi positioning in a bluetooth and wifi coexistence environment. In *Ubiquitous Positioning, Indoor Navigation, and Location Based Service (UPINLBS), 2012*, pages 1–6. IEEE, 2012.
Cited on page 120.

[69] Maurice Bertram Priestley. Non-linear and non-stationary time series analysis. 1988.
Cited on page 124.

[70] Nissanka B Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket location-support system. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 32–43. ACM, 2000.
Cited on page 4.

[71] Anshul Rai, Krishna Kant Chintalapudi, Venkata N Padmanabhan, and Rijurekha Sen. Zee: zero-effort crowdsourcing for indoor localization. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pages 293–304. ACM, 2012.
Cited on page 5.

[72] Raimundo Real and Juan M Vargas. The probabilistic basis of jaccard's index of similarity. *Systematic biology*, 45(3):380–385, 1996.
Cited on page 129.

[73] Kathryn Roeder and Larry Wasserman. Practical bayesian density estimation using mixtures of normals. *Journal of the American Statistical Association*, 92(439):894–902, 1997.
Cited on page 43.

[74] Souvik Sen, Romit Roy Choudhury, Bozidar Radunovic, and Tom Minka. Precise indoor localization using phy layer information. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, page 18. ACM, 2011.
Cited on page 138.

[75] Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *The Journal of Machine Learning Research*, 9:371–421, 2008.
Cited on pages 58, 59, 60, 61, 63, and 64.

[76] Kush Shah. Implementation and integration of cellular/gps-based vehicle tracking system with google maps using a web portal. In *Proceedings of the International Congress on Information and Communication Technology*, pages 513–520. Springer, 2016.
Cited on page 119.

[77] Beomju Shin, Jung Ho Lee, Taikjin Lee, and Hyung Seok Kim. Enhanced weighted k-nearest neighbor algorithm for indoor wi-fi positioning systems. In *Computing Technology and Information Management (ICCM), 2012 8th International Conference on*, volume 2, pages 574–577. IEEE, 2012.
Cited on page 22.

[78] G Soromenho. Comparing approaches for testing the number of components in a finite mixture model. *Computational Statistics*, 9(1):65–78, 1994.
Cited on page 45.

[79] Kalyan Pathapati Subbu, Brandon Gozick, and Ram Dantu. Indoor localization through dynamic time warping. In *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, pages 1639–1644. IEEE, 2011.
Cited on page 90.

[80] Kalyan Pathapati Subbu, Brandon Gozick, and Ram Dantu. Locateme: Magnetic-fields-based indoor localization using smartphones. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(4):73, 2013.
Cited on page 82.

[81] Fazli Subhan, Halabi Hasbullah, Azat Rozyyev, and Sheikh Tahir Bakhsh. Indoor positioning in bluetooth networks using fingerprinting and lateration approach. In *Information Science and Applications (ICISA), 2011 International Conference on*, pages 1–9. IEEE, 2011.
Cited on page 5.

[82] S Suksakulchai, S Thongchai, DM Wilkes, and K Kawamura. Mobile robot localization using an electronic compass for corridor environment. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, volume 5, pages 3354–3359. IEEE, 2000.
Cited on page 92.

[83] Dwi Joko Suroso, Panarat Cherntanomwong, Pitikhate Sooraksa, and J-i Takada. Location fingerprint technique using fuzzy c-means clustering algorithm for indoor localization. In *TENCON 2011-2011 IEEE Region 10 Conference*, pages 88–92. IEEE, 2011.
Cited on pages 38, 44, and 47.

[84] Alex Varshavsky, Eyal De Lara, Jeffrey Hightower, Anthony LaMarca, and Veljo Otsason. Gsm indoor localization. *Pervasive and Mobile Computing*, 3(6):698–720, 2007.
Cited on page 7.

[85] Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*. Springer, 2005.
Cited on pages 58, 59, 60, 61, 63, 64, and 65.

[86] Bang Wang, Shengliang Zhou, Laurence T Yang, and Yijun Mo. Indoor positioning via subarea fingerprinting and surface fitting with received signal strength. *Pervasive and Mobile Computing*, 2015.
Cited on page 5.

[87] He Wang, Souvik Sen, Ahmed Elgohary, Moustafa Farid, Moustafa Youssef, and Romit Roy Choudhury. No need to war-drive: unsupervised indoor localization. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 197–210. ACM, 2012.
Cited on pages 5 and 89.

[88] Roy Want, Andy Hopper, Veronica Falcao, and Jonathan Gibbons. The active badge location system. *ACM Transactions on Information Systems (TOIS)*, 10(1):91–102, 1992.
Cited on pages xiii, 4, 5, 6, and 16.

[89] Mark Weber, Ulrich Birkel, Ralf Collmann, and Julia Engelbrecht. Comparison of various methods for indoor rf fingerprinting using leaky feeder cable. In *Positioning Navigation and Communication (WPNC), 2010 7th Workshop on*, pages 291–298. IEEE, 2010.
Cited on page 5.

[90] Roland Winkler, Frank Klawonn, and Rudolf Kruse. Problems of fuzzy c-means clustering and similar algorithms with high dimensional data sets. In *Challenges at the Interface of Data Analysis, Computer Science, and Optimization*, pages 79–87. Springer, 2012.
Cited on page 47.

[91] Denis F Wolf and Gaurav S Sukhatme. Mobile robot simultaneous localization and mapping in dynamic environments. *Autonomous Robots*, 19(1):53–65, 2005.
Cited on page 137.

[92] Chenshu Wu, Zheng Yang, Yunhao Liu, and Wei Xi. Will: Wireless indoor localization without site survey. *Parallel and Distributed Systems, IEEE Transactions on*, 24 (4):839–848, 2013.
Cited on pages 38 and 44.

[93] Hongwei Xie, Tao Gu, Xianping Tao, Haibo Ye, and Jian Lv. Maloc: a practical magnetic fingerprinting approach to indoor localization using smartphones. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 243–253. ACM, 2014.
Cited on page 82.

[94] Zheng Yang, Zimu Zhou, and Yunhao Liu. From rssi to csi: Indoor localization via channel response. *ACM Computing Surveys (CSUR)*, 46(2):25, 2013.
Cited on page 138.

[95] Zheng Rong Yang. *Machine learning approaches to bioinformatics*, volume 4. World scientific, 2010.
Cited on pages 38 and 40.

[96] Eiko Yoneki. Fluphone study: Virtual disease spread using haggle. In *Proceedings of the 6th ACM workshop on Challenged networks*, pages 65–66. ACM, 2011.
Cited on page 119.

[97] Sungro Yoon, Kyunghan Lee, and Injong Rhee. Fm-based indoor localization via automatic fingerprint db construction and matching. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, pages 207–220. ACM, 2013.
Cited on page 7.

[98] Moustafa Youssef and Ashok Agrawala. The horus wlan location determination system. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 205–218. ACM, 2005.
Cited on pages 5 and 22.

[99] Yongtuo Zhang, Wen Hu, Weitao Xu, Hongkai Wen, and Chun Tung Chou. Naviglass: Indoor localisation using smart glasses. In *Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks*, pages 205–216. Junction Publishing, 2016.
Cited on page 7.

[100] Hao Zhou and Nguyen Ngoc Van. Indoor fingerprint localization based on fuzzy c-means clustering. In *Measuring Technology and Mechatronics Automation (ICMTMA), 2014 Sixth International Conference on*, pages 337–340. IEEE, 2014.
Cited on page 47.

Appendix A

K-Means clustering

KM is a well-known hard clustering algorithm which partitions the training data into distinct clusters. It aims to put each training example into the cluster whose centroid is closest. The centroid is the mean of the cluster. Without loss of generality, given the training examples $(T_1, \ldots, T_M)$ and the total number of clusters k, KM partitions the examples into k clusters $S = \{S_1, \ldots, S_k\}$ so as to minimise the following function, which sum the distance of each training example in the cluster to the centroid, where μ_i is the centroid of the cluster S_i.

$$arg\ \underset{S}{min} \sum_{i=1}^{k} \sum_{j=1}^{M} ||T_j - \mu_i||^2 \quad . \tag{A.1}$$

The *4* steps to perform KM are briefly summarised below.

(i) Given a number of clusters k, the algorithm selects k training examples to become the centroids, one for each cluster. These centroids are chosen to be as far away from each other as possible.

(ii) The remaining training examples are assigned to the cluster whose centroid is closest to them.

(iii) The k centroids are re-calculated to be the middle examples of the newly formed clusters.

(iv) The steps (ii) and (iii) are repeated until the centroids no longer change.

Although it has been proved that the above steps always terminate, the efficiency of KM does not just depend on the number of k, but also relies on the initial chosen centroids [32, 59]. KM is normally run several times with different initial centroids on the same training set and the best result is selected.

Appendix B

Fuzzy c-Means clustering

With FCM, the cluster is associated with a function to demonstrate the degree to which each training example belongs to the cluster. Given a number of clusters c, FCM partitions the training database $\{T_1, \ldots, T_M\}$ into c clusters by minimising the following squared error function

$$J_n = \sum_{i=1}^{M} \sum_{j=1}^{c} u_{ij}^n ||T_i - c_j||^2, 1 \leq n \leq \infty \ . \tag{B.1}$$

where $||T_i - c_j||$ is the distance between the training example T_i and the cluster centre c_j, u_{ij} is the degree of membership of T_i in the cluster j, and n is any positive real number which controls the influence of the membership (i.e. the degree of fuzzy overlap).

The *4* steps to perform FCM are briefly summarised below.

(i) Matrix U=[u_{ij}] is initialised.

(ii) At step k^{th}, the cluster centre is calculated as follows. C(k)=[cj] with U(k)

$$c_j = \frac{\sum_{i=1}^{M} u_{ij}^n \cdot T_i}{\sum_{i=1}^{M} u_{ij}^n} \ . \tag{B.2}$$

(iii) Updating $U^{(k)}$ and $U^{(k+1)}$

$$u_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{||x_i - c_j||}{||x_i - c_k||} \right)^{\frac{2}{n-1}}} \ . \tag{B.3}$$

(iv) Returning to step (ii) until $||U^{(k+1)} - U^{(k)}|| < \varepsilon$, where ε is a termination constant between [0,1].

Appendix C

Dynamic Time Warping

Given two variable length sequences $s = \{s_1, \ldots, s_n\}$ and $t = \{t_1, \ldots, t_m\}$, DTW will measure the distance between them by finding an optimal alignment based on dynamic programming. The idea is to find a mapping between all the points of s and t. The DTW algorithm is described as follows.

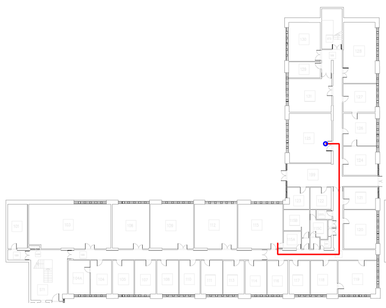
```
**Data:** Two sequences $s = \{s_1, \ldots, s_n\}$ and $t = \{t_1, \ldots, t_m\}$.
**Result:** Positioning prediction.

DTW[0, 0] = 0;
for $i = 1 \to m$ do
    DTW[0, i] = ∞ ;
end
for $i = 1 \to n$ do
    DTW[i, 0] = ∞ ;
end
for $i = 1 \to n$ do
    for $i = 1 \to m$ do
        distance = s[i] - t[j];
        DTW[i, j] := distance + min(DTW[i-1, j], DTW[i , j-1], DTW[i-1, j-1]);
    end
end
return DTW[n, m];
```

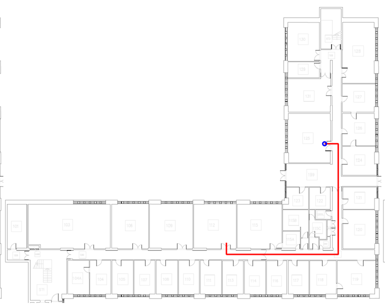
Algorithm 5: Dynamic Time Warping algorithm.

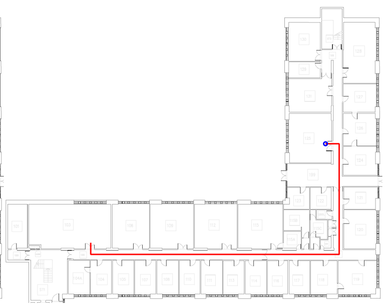
Appendix D

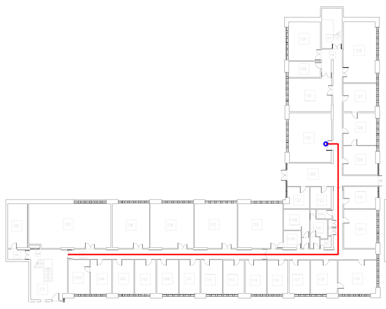
The routine training database

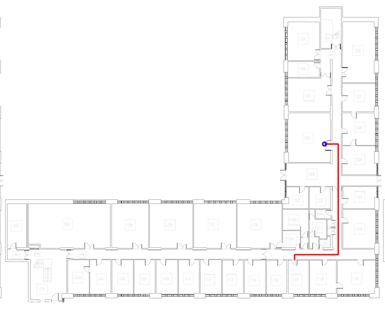
This section illustrates all *50* training trajectories used in Chapter 5.


From 125 to 115


From 125 to 112


From 125 to 103


From 125 to 19


From 125 to 117

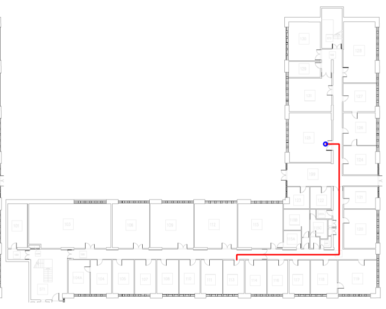
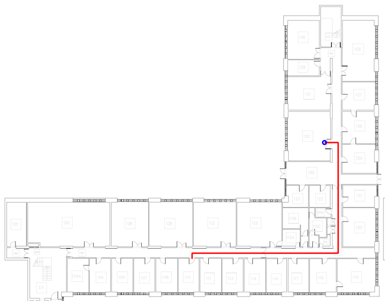
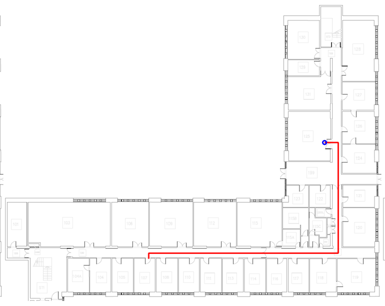
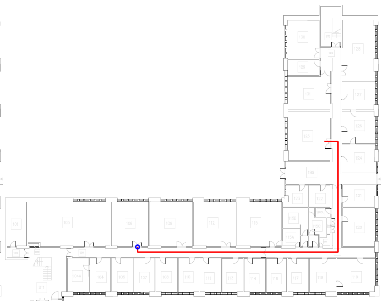
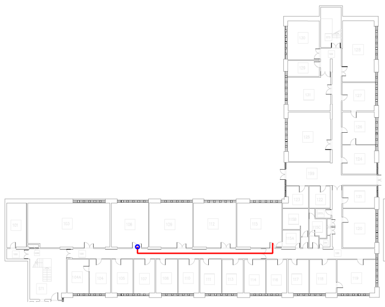

From 125 to 113

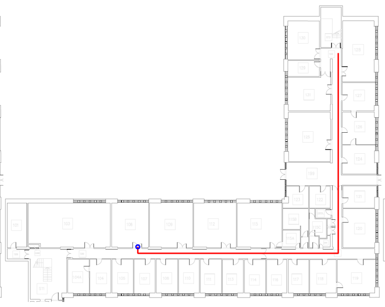
Fig. D.1 The *50* training trajectories. The blue circle denotes the starting point.

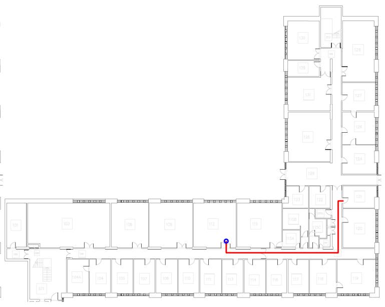

From 125 to 110

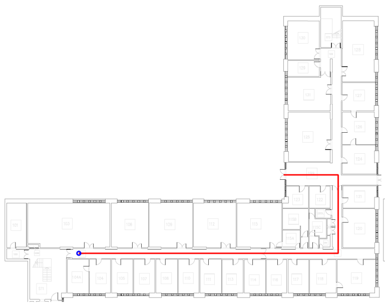

From 125 to 107

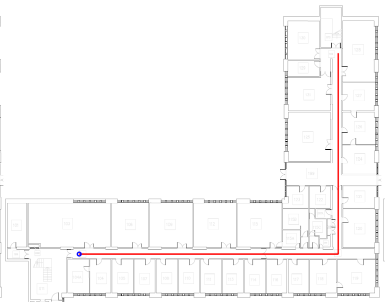

From 106 to 125

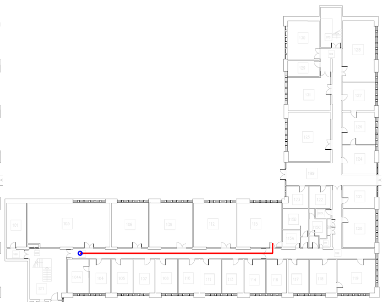

From 106 to 115

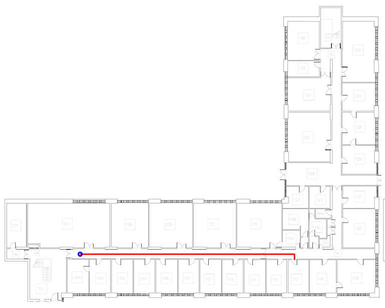

From 106 to 199

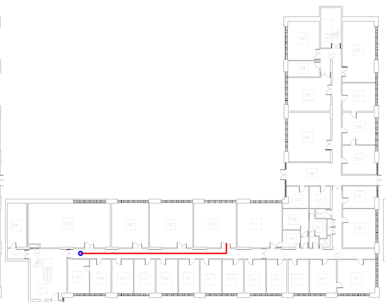

From 112 to 121


From 19 to 99


From 19 to 199


From 19 to 115


From 19 to 117


From 19 to 112

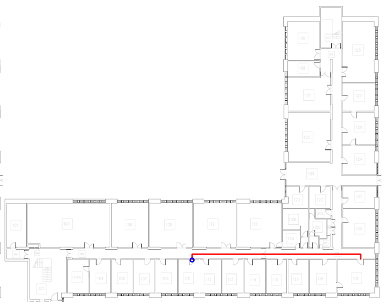
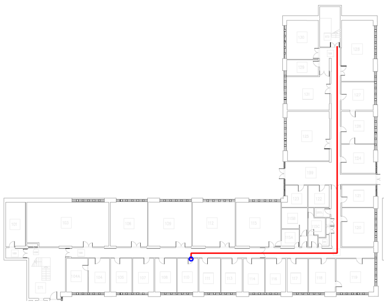
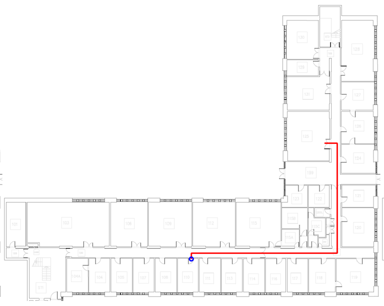
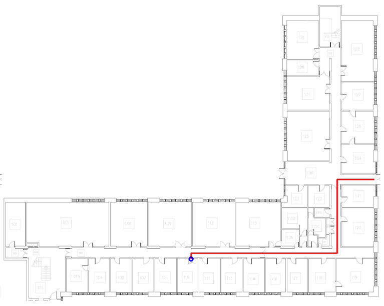
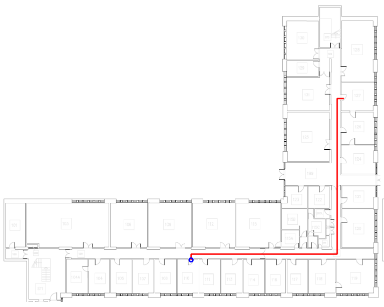

From 110 to 119

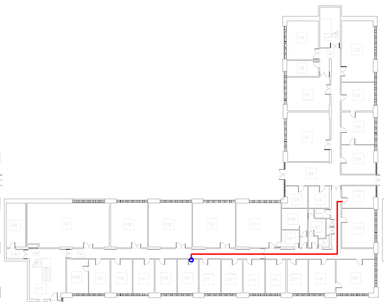
Fig. D.1 The *50* training trajectories. The blue circle denotes the starting point.

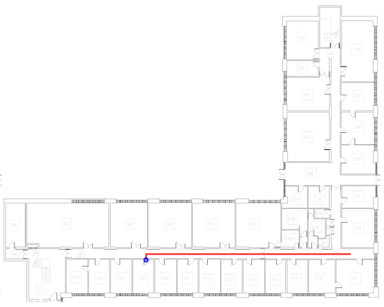

From 110 to 199

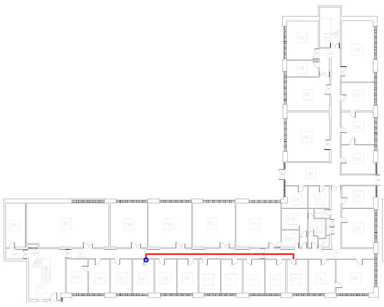

From 110 to 125

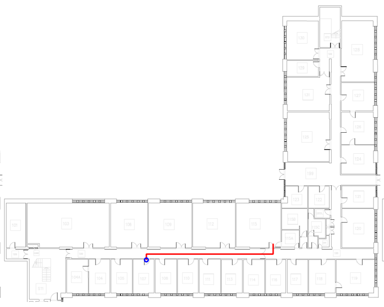

From 110 to 100

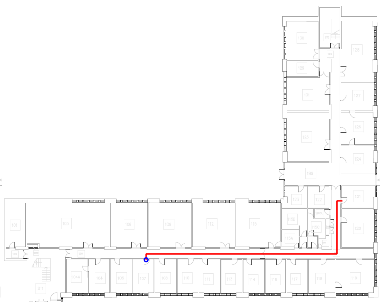

From 110 to 127

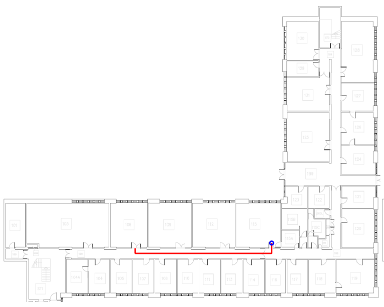

From 110 to 121

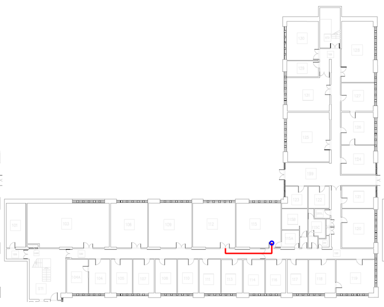

From 107 to 119


From 107 to 117


From 107 to 115


From 107 to 121


From 115 to 106


From 115 to 112

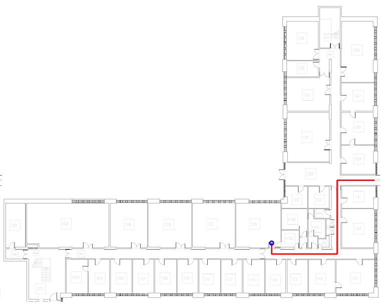

From 115 to 100

Fig. D.1 The *50* training trajectories. The blue circle denotes the starting point.

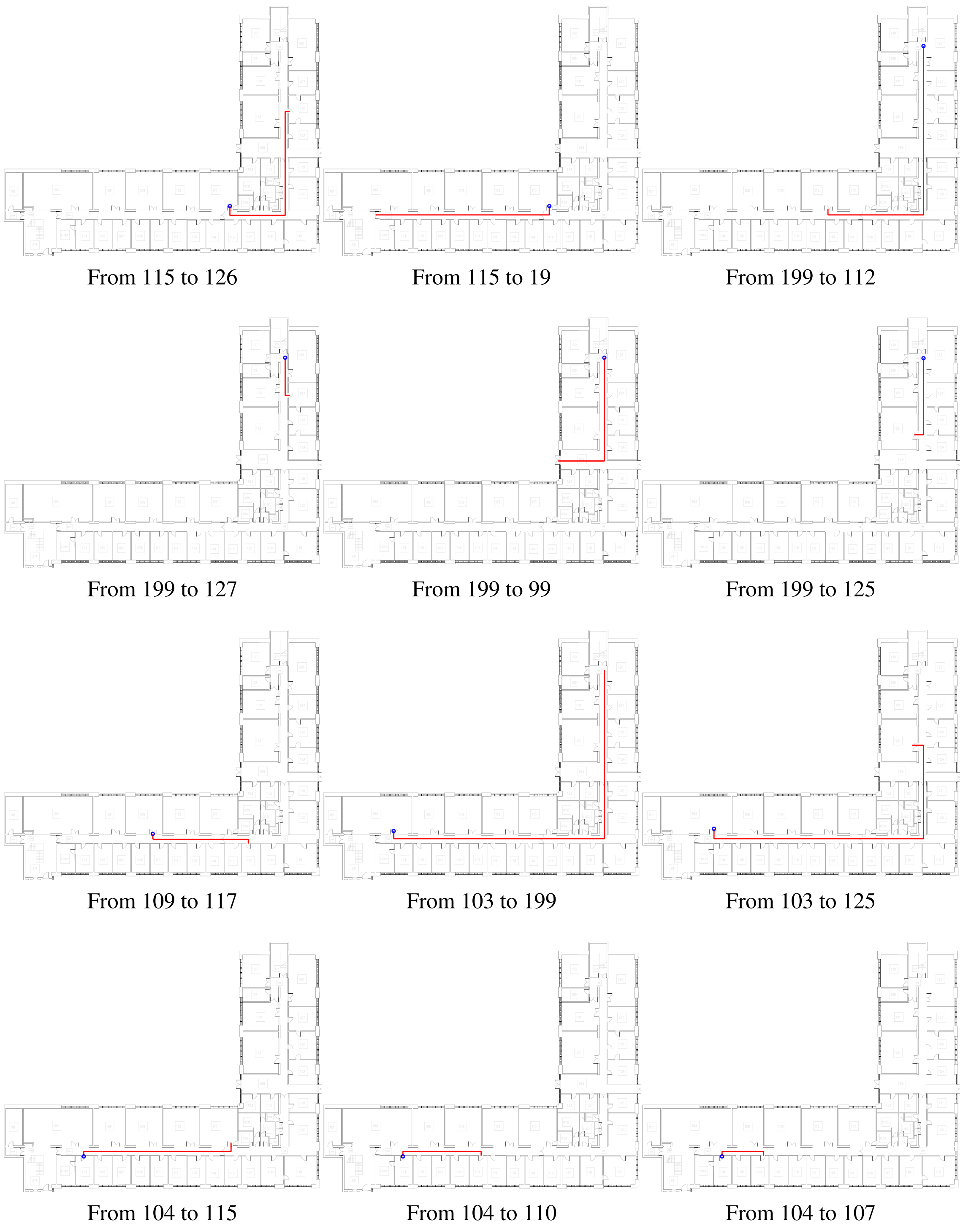


Fig. D.1 The *50* training trajectories. The blue circle denotes the starting point.

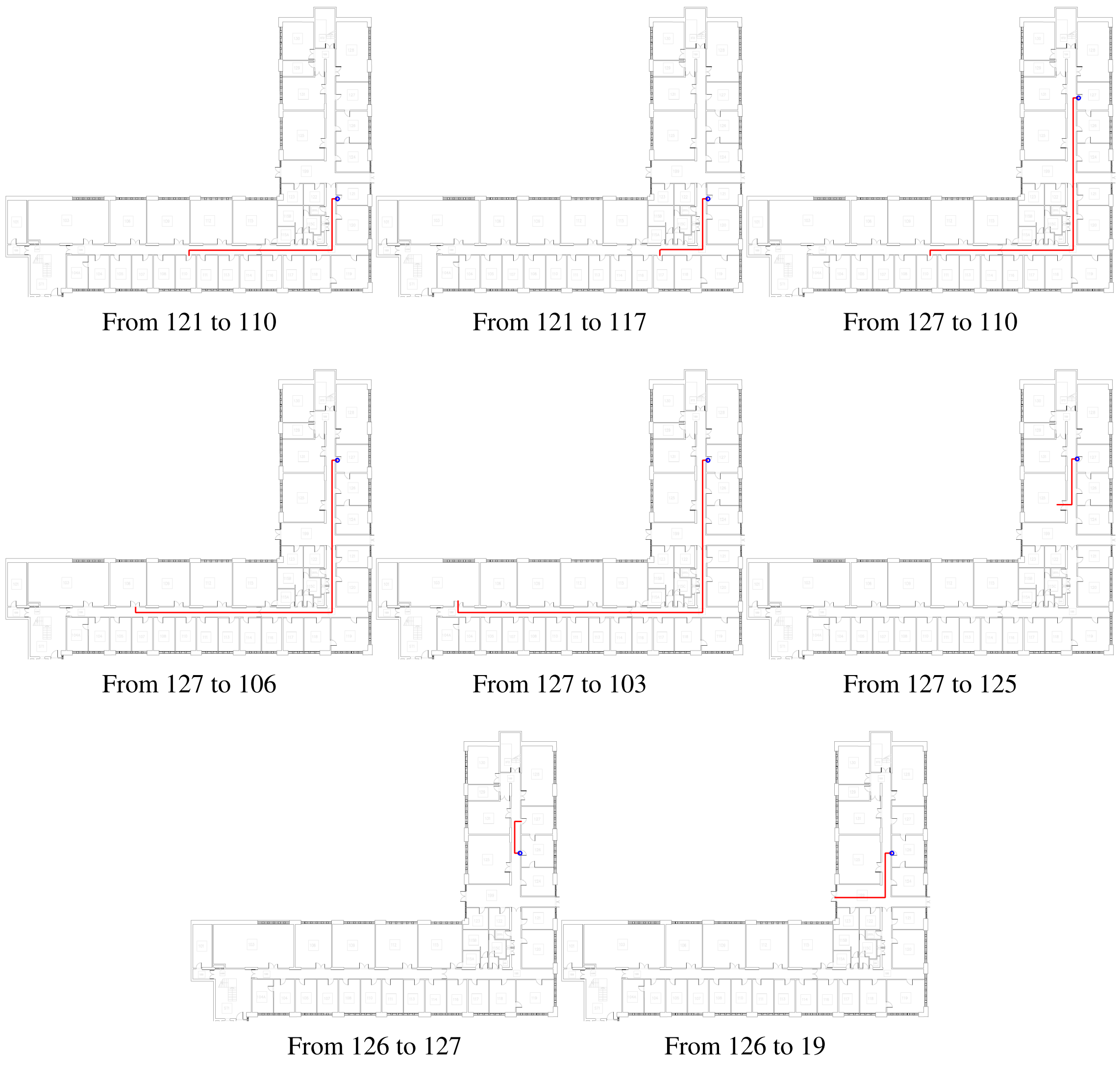


Fig. D.1 The *50* training trajectories. The blue circle denotes the starting point.