LEARNING-BASED LOCALIZATION IN WIRELESS AND SENSOR NETWORKS

by

JUNFENG PAN

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
in Computer Science and Engineering

November 2007, Hong Kong

Authorization

I hereby declare that I am the sole author of the thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

_____

JUNFENG PAN

LEARNING-BASED LOCALIZATION IN WIRELESS AND SENSOR NETWORKS

by

JUNFENG PAN

This is to certify that I have examined the above Ph.D. thesis

and have found that it is complete and satisfactory in all respects,

and that any and all revisions required by

the thesis examination committee have been made.

PROF. QIANG YANG, THESIS SUPERVISOR

PROF. LIONEL M. NI, HEAD OF DEPARTMENT

Department of Computer Science and Engineering

9 November 2007

ACKNOWLEDGMENTS

First of all, I would like to sincerely and gratefully express my thanks to my supervisor Prof. Qiang Yang, for his valuable advice and in-depth guidance throughout my doctoral research. With his help, I learned how to survey a new field, identify interesting problems, formalize them and seek for suitable solutions. He gave me helpful feedback on conducting presentations and writing papers. His extensive discussions and insightful explorations have been of great value for this work. This dissertation would never have been completed without his consistent and generous support.

I would like to thank James Kwok, Jeffrey Xu Yu, Lionel Ni, Andrew Lim, Dit-Yan Yeung, Fangzhen Lin, Brian Mak, and Yunhao Liu for serving as the committee members of my qualifying, proposal and thesis defenses. Their valuable comments are of great help in my research work.

I would like to thank the members of the Artificial Intelligence & Statistics research group – Dr. Rong Pan, Dr. Jie Yin, Dr. Hong Chang, Dr. Dou Shen, Sinno Jialin Pan, Vincent Wenchen Zheng, Kai Zhang, Ivor Tsang and so on – for their constructive discussions and friendly help. I would also wish to thank Hejun Wu, Wenwei Xue, Jian Ma, Quanbin Chen and Jingdong Wang from the database, the networking and the vision groups for their useful guidance.

During my internship at Google New York, I got generous help from my mentors Robert Doorenbos, Thatcher Ulrich and Alice Bonhomme-Biais. I also got great help from Yu Chen and Bijun He. I would also thank many friends, Edmond Kayi Lee, Justin Wan, Yuanyuan Zhao, Ye Zhou, Ming Lin, Ning Hu, Hua Yu, Julian Qian, Tim Pan, Pinxing Ye, Jinhai Rao and Xiao Wu for their generous help.

Especially, I would like to thank Andrew Moore for his guidance in Google Pittsburgh. I would thank Eugene Fink and Carlos Guestrin for hosting me in Carnegie Mellon University; Henry Kautz, Matt Post and Benjamin Van Durme for hosting me in University of Rochester; Jianhui Xie, Benfeng Chen and Hoifung Poon for hosting me in Seattle; Microsoft for hosting me in Redmond.

During my internship at the Computer Science and Artificial Intelligence Laboratory in Massachusetts Institute of Technology, I would like to thank Seth Teller, John

Fisher, Federico Mora, Yoni Battat, Jun-Geun Park and Been Kim for their guidance and discussion.

Last but not least, I would like to give special thanks to my parents Zhifu Pan and Jianzhen Zhong, my brother Junbiao Pan. I just can not tell how much they love me and I love them. Meanwhile, I would like to give great thanks to Yanli Cai. She always encourages me when I feel depressed and brings me fun. It is their love, support and understanding that make everything I have possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LEARNING-BASED LOCALIZATION IN WIRELESS AND SENSOR NETWORKS

by

JUNFENG PAN

Department of Computer Science and Engineering

The Hong Kong University of Science and Technology

ABSTRACT

Accurately tracking mobile devices in wireless and sensor networks using received-signal-strength (RSS) values is a useful task in robotics and activity recognition. It is also a difficult task since radio signals usually attenuate in a highly nonlinear and uncertain way in a complex environment where client devices may be moving. Many existing RSS localization systems suffer from the following problems: first, many of them are inaccurate. Second, to increase their accuracy, many require costly manual calibration. Third, many of them cannot cope with changing data as users move in a dynamic environment. In this thesis, we will describe our learning-based solution using kernels, manifolds and graph Laplacian for solving the these problems. We will demonstrate the effectiveness of our algorithms for tracking static and mobile devices in complex indoor environments using wireless local area network, wireless sensor networks and radio frequency identification networks with much less calibration effort.

CHAPTER 1

INTRODUCTION

With the recent advance in pervasive computing technology and mobile devices, location-sensing and context-aware systems have attracted intense attention. Location-based systems [24, 27, 11, 65, 51] have a variety of applications from object tracking, people positioning [24, 27, 82, 5, 51, 13], content delivery, security [90], environment [88] monitoring and activity recognition [97, 15, 58, 59].

Accurately tracking *mobile* nodes in wireless sensor networks using received-signal-strength (RSS) values is a complex and difficult task. Radio signal usually attenuates in a way that is highly nonlinear and uncertain in a complex environment, which may be further corrupted when introducing the mobility of sensor nodes. Two main approaches have been taken, as follows:

- The first method is a propagation-based approach for localizing mobile nodes in a wireless sensor network. These methods [80, 76] usually consist of two main steps, by first transforming sensor reading into a distance measure and then recovering the most probable coordinates of sensor nodes. These approaches usually rely on a sophisticated signal propagation model and extensive hardware support. A problem with this approach is that it usually has low accuracies due to the noisy environment.

- A second approach is the learning-based approach, [64, 5, 60], which can bypass the ranging process but needs relatively more calibration data. A problem with this method is that manual calibration is expensive, and the calibrated data can be outdated easily.

- In a dynamic wireless environment, the data are changing dynamically. Furthermore, a user to be tracked may be moving constantly. Thus, the incoming data correspond to a data stream. How to learn a localization model from a stream of data is yet another open problem.

In summary, we have the following three problems we try to solve:

- Increase the accuracy of localization
- Reduce the calibration effort
- Online and incremental model update

1.1 Thesis Outline

- Chapter 2 gives an overview of tracking system from several perspectives: the infrastructure, the architecture and the algorithm. Considering the infrastructures, it can be wireless local area networks, wireless sensor networks and radio frequency identification networks. We study the radio characteristics of the three networks. Considering what is carried in the object being tracked, it can be receiver-oriented, transmitter-oriented, transceiver-oriented and non-intrusive models. Considering the algorithms used for location estimation, it can be propagation-based or learning-based models. We discuss and compare the strength and the weakness of propagation-based and learning-based models. We also survey several works on localization and user behavior analysis.

- Chapter 3 studies the problem of increasing the localization accuracy using kernel methods. Our aim is to obtain an accurate mapping between the signal space and the physical space. This location estimation problem has traditionally been tackled through probabilistic models trained on manually labelled data, which are expensive to obtain. In contrast, our algorithm adopts Kernel Canonical Correlation Analysis (KCCA) to build a nonlinear mapping between the signal-vector space and the physical location space by transforming data in both spaces into their canonical features. This allows the pairwise similarity of samples in both spaces to be maximally correlated using kernels. We use a Gaussian kernel to adapt to the noisy characteristics of signal strengths and a Matérn kernel to sense the changes in physical locations.

- Chapter 4 presents a manifold regularization approach to calibration-effort reduction for tracking a mobile node in a wireless sensor network. Many previous approaches to the location-estimation problem assume the availability of sufficient calibrated data. However, to obtain such data requires great effort. In this chapter, we propose a semi-supervised manifold learning technique for sensor-network-based tracking. We compute a subspace mapping function between the

signal space and the physical space by using a small amount of labelled data and a large amount of unlabelled data. This mapping function can be used online to determine the location of mobile nodes based on the signals received.

- Chapter 5 addresses the problem of recovering the locations of both mobile devices and access points from radio signals, a problem which we call co-localization, by exploiting both labelled and unlabelled data from mobile devices and access points. We first propose a solution using Latent Semantic Indexing to construct the relative locations of the mobile devices and access points when their absolute locations are unknown. We then propose a semi-supervised learning algorithm based on manifold to obtain the absolute locations of the devices. Both solutions are finally combined together in terms of graph Laplacian.

- Chapter 6 extends the co-localization model to an online and incremental one so that it can deal with streaming calibration data. Many tracking systems function in two phases: an offline training phase and an online localization phase. In the training phase, models are built from a batch of data that are collected offline. Most of them can not cope with a dynamic environment in which calibration data may come sequentially. These systems may gradually become inaccurate without a manually costly re-calibration. To solve this problem, we proposed an online co-localization method that can deal with labelled and unlabelled data stream based on semi-supervised manifold-learning techniques.

- Chapter 7 applies localization algorithms for location-based data sharing. With the proliferation of wireless and sensor techniques, data can be shared conveniently through the air. However, wireless communication is vulnerable since normal packets may be eavesdropped and malicious packets may be injected without being physically connected. To enhance data security and protect user privacy, data must be sent to the right machine at the right place at the right time. Many previous works have addressed the problem of building secure connection using cryptographic techniques and well-designed protocols. However, seldom works have been done to control the data sharing within an expected physical area. How to track a wireless client and verify its location claim is still an open problem. In this chapter, we propose *digital wall*, a power-efficient location-based data sharing system to control data communication within a customized space. We use learning-based techniques to determine the location of a mobile

client based on received signal strengths. We study how the performance may be affected when the controlled area is confined by physical or virtual walls. Experimental results show that we can well distinguish whether a client device is located within an expected *digital wall*.

- Chapter 8 describes PANE, a WiFi signal data collecting and location labelling tool for positioning and navigation experiments. To evaluate the performance of a localization model, we may need to collect large amount of labelled data. Similarly, to set up a tracking system, sufficient data shall be collected for calibration. In an outdoor environment, GPS may be employed to give ground truth locations. However, GPS may not work in an indoor environment and data collection is usually manually done. This is a time consuming and error prone process. In this chapter, we introduce a convenient tool developed by us for building maps, collecting signal strength values and labelling locations. More specifically, it supports general wireless devices. It has rich functions for creating and maintaining multi-layer maps, collecting received signal strength values and marking down user trajectory and access point locations. Many of our WiFi experimental dataset are obtained with this tool.

- Chapter 9 propose VEST, a video-based evaluation system for mobile tracking in wireless sensor networks. To evaluate the performance of tracking mobile objects is challenging problem. The real time trajectory of the moving object must be recorded accurately so that it can be treated as the ground truth. This trajectory is not possible to manually mark down when the object keeps changing its location. Instead, we may use special hardware to automate this process. In this chapter, we use video camera array to track the location of a moving target on a two-dimensional floor and in a three-dimensional space. Multiple cameras must be set up to cover the whole test-bed. They are calibrated using the landmarks on the floor. After that, we apply an image segmentation algorithm to recognize the moving object based on many features such as motion, color and shape. Multiple cameras are synchronized and work collaboratively so that local coordinates in each camera are combined and transformed to global ones. By carefully calibrating the cameras, we can track an object within a few centimeters. The objective of VEST is to extract ground truth locations for evaluation rather than to replace an RSS-based tracking system. Visual tracking may involve too much

computational overhead.

1.2 Main Contribution

In this thesis, we study the problem of localization and tracking in wireless and sensor networks from the machine learning perspective. The main contributions can be summarized as follows:

- We applied cutting edge machine learning techniques for tracking in wireless and sensor networks. We built an accurate mapping between signal space and location space. We reduced calibration effort by incorporating unlabelled data.
- We proposed co-localization, a general and flexible framework for recovering the location of access points and mobile devices using labelled and unlabelled data from access points and mobile devices. Co-localization can be operated in a two-phase manner or purely online.
- We did a thorough study of our localization algorithm in wireless local area networks, wireless sensor networks and radio frequency identification networks, in which the target could be static or moving.
- We developed vision-based accurate evaluation system and convenient tool for wireless and sensor network experiments.
- We spent huge amount of time collecting several data sets. Other researchers may play around with these data sets and save a lot of time on data collection.

CHAPTER 2

LOCALIZATION IN PERVASIVE COMPUTING ENVIRONMENTS

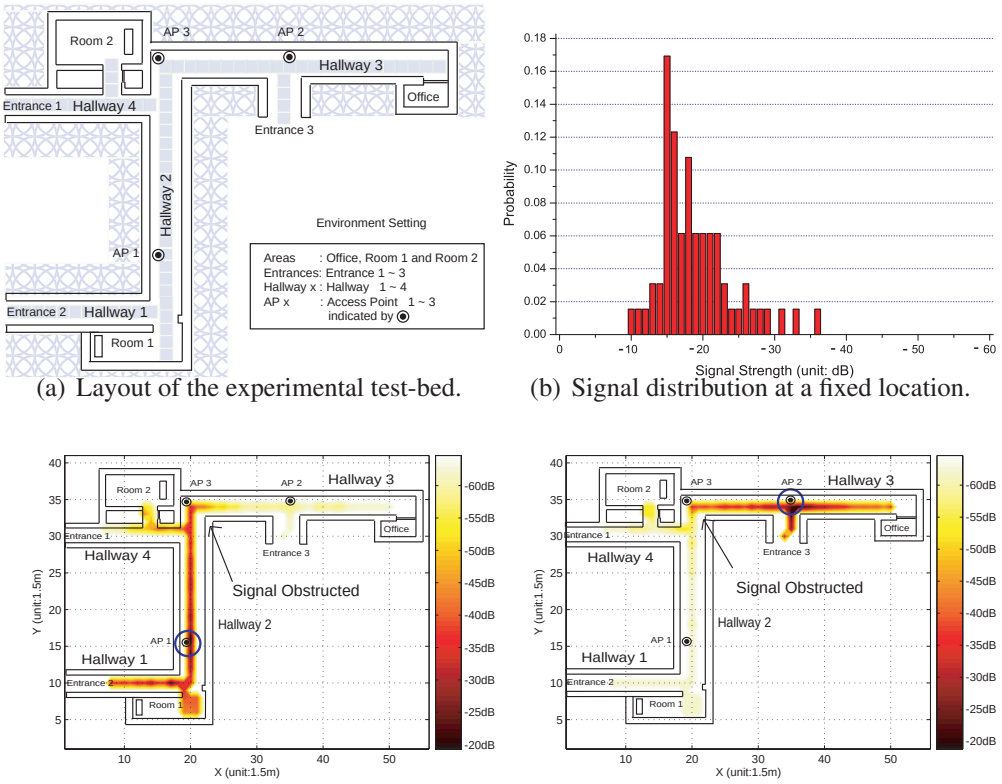
Localization is a fundamental and important problem in robotics, location-based service and content delivery [27, 11, 65, 5, 88, 97, 59]. As the recent advancement of wireless and sensor techniques, tiny sensors are deployed more densely than before in the environment, objects and human bodies. Sensor readings such as temperature, pressure, humidity, lightness, signal strength, acceleration can be easily obtained and analyzed in real time for many applications, from low-level location estimation [92, 25, 57, 64, 65, 5] to high-level activity recognition [97, 15, 58, 59] and abnormal detection [100]. Among these sensors, received signal strength (RSS) is perhaps the most general in wireless devices. An RSS-based localization system can be classified based on different criteria. Considering the infrastructure, it can be wireless local area networks, wireless sensor networks and radio frequency identification networks. Considering what is carried in the object being tracked, it can be receiver-oriented, transmitter-oriented, transceiver-oriented and non-intrusive models. Considering the algorithms used for location estimation, it can be propagation-based or learning-based models. We discuss and compare the strength and the weakness of propagation-based and learning-based models.

2.1 Pervasive Computing Environments

We mainly set up three test-beds for our tracking experiments. They are wireless local area networks, wireless sensor networks and radio frequency identification networks. They are used in the experiments in [67, 68, 70, 69, 71]. We describe these test-beds and their radio characteristics as follows.

2.1.1 Wireless Local Area Networks

A person carrying an IBM© T42 notebook, which has an Intel© Pro/2200GB internal wireless card, walks in an indoor environment of about $60m \times 50m$ in size as shown in



(a) Layout of the experimental test-bed. (b) Signal distribution at a fixed location.

(c) Signal distribution from AP 1. (d) Signal distribution from AP 2.

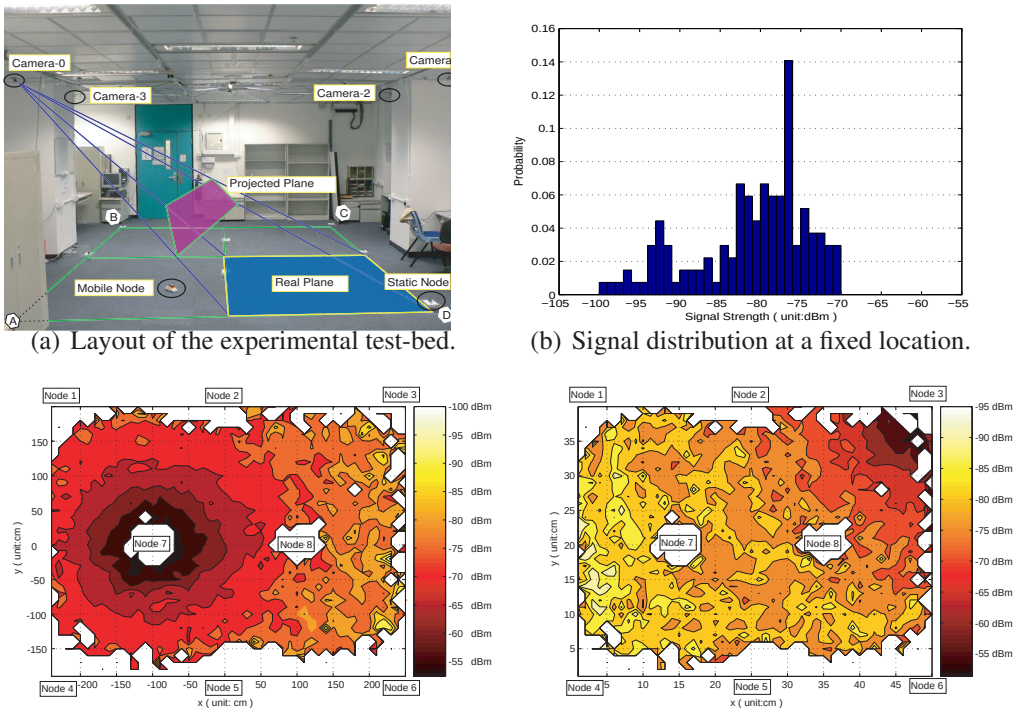
Figure 2.1: WLAN experimental test-bed and radio propagation characteristics.

Figure 2.1(a). It is equipped with an IEEE 802.11b/g wireless network in the 2.4GHz frequency bandwidth. We can detect more than 20 access points and only 3 is shown in Figure 2.1(a). The person walks in the hallways and collect data with sample rate $2Hz$. The ground-truth location labels are obtained by referring to landmark points such as doors, corners and dead-ends. The localization area is composed by one-dimensional hallways. Detailed experiments based on the test-bed can be found in [67, 68, 69, 71].

The IEEE 802.11b/g wireless local area networks use radio in the $2.4GHz$. Accurately predicting signal strength is a difficult task since radio signal propagates in a complex way in an indoor environment [38, 33]. It may also be weaken or obstructed by walls and human bodies. Due to reflection, refraction, scattering and absorption of radio waves by structures inside a building, a transmitted signal can reach the receiver through different paths, each having its own amplitude and phase. Moreover, with disturbance from other equipments such as cordless phones, the received signal is usually distorted and noisy. In Figure2.1(a), there are three access points (APs). Figure 2.1(b) shows the typical signal distribution at a particular location from a fixed access point. As can be seen, this noisy signal can be as weak as -36dB and as strong as -10dB. Its

empirical distribution (*radio map*) is thus difficult to obtain, especially when the training samples are scarce. Figures 2.1(c) and 2.1(d) show the average signal strength distributions of AP_1 and AP_2 respectively. As can be seen, the signal strength changes sharply at the corner intersecting hallways 2 and 3 due to obstruction of the walls. It does not obey the outdoor radio propagation model anymore. As a result, AP_3 is installed at that corner to enhance coverage and stability of the signal.

2.1.2 Wireless Sensor Networks



(a) Layout of the experimental test-bed.

(b) Signal distribution at a fixed location.

(c) Signal distribution of Wireless Sensor Node 7.

(d) Signal distribution of Wireless Sensor Node 3.

Figure 2.2: WSN experimental test-bed and radio propagation characteristics.

The sensor-based tracking experiment is performed in the Pervasive Computing Laboratory (Figure 2.2(a)), Department of Computer Science and Engineering. The room is large enough for us to set up an experimental test-bed of 5.0 meter by 4.0 meter. In figure 2.2(a), $|P_1P_3| = |P_4P_6| = 5.0m$ and $|P_1P_4| = |P_3P_6| = 4.0m$. More specifically, we use CrossBow MICA2 and MICA2Dot to construct a wireless sensor network. We program these sensor nodes to broadcast and detect beacon frames periodically so that they could measure the Radio Signal Strength (RSS) of each other. By combining the RSS from different nodes we can estimate locations of these nodes. Figure 2.2(a) shows that there are eight *static* sensor nodes deployed on the test-bed:

six around the rectangular boundary denoted as $1, 2, \ldots, 6$; another two around the central area denoted as $7, 8$. There is one *mobile* node, which is attached on top of a toy car. We config all the nine nodes so that each one could measure the RSS from the remain eight nodes in every 0.5s.

It is well known that RSS is quite complicated in real environments due to signal reflection, diffraction, and scattering. Figure 2.3 shows the relation of signal strength and distance between two wireless sensor nodes (MICA2). The signal attenuates while increasing the distance. We could also see that the variance at a longer distance is larger than that of a shorter distance.

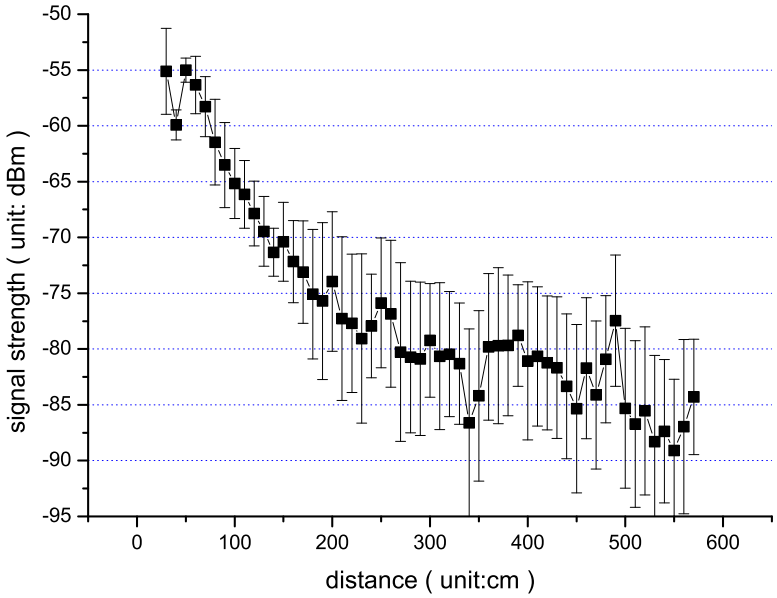


Figure 2.3: Relation of signal strength and distance

Similarly, there are a lot of factors that may affect signal strength in wireless sensor networks. Even when the distance between a transmitter and a receiver is fixed, the signal may change from time to time. Factors that affect signal strength may be battery voltage, emission power, temporal and spatial dynamics.

When all these factors are combined together, the signal strength we observed is rather uncertain. Our experiment is performed in the Pervasive Computing Laboratory. See Figure 2.2(a). The room is large enough for us to set up an experimental test-bed of 5.0 meter by 4.0 meter. Figure 2.2(b) shows that the signal changes at a fixed location. Figure 2.2(c) and 2.2(d) show the signal distribution of sensor node 7 and 3

respectively. We may see that the signal attenuates roughly in the shape of co-centric circles when the distance is close to the transmitter in Figure 2.2(c). However, when the distance between the receiver and transmitter is large, the signal becomes more noisy in Figure 2.2(c). More detail can be found in [70, 69].

2.1.3 Radio Frequency Identification Networks

The RFID experiment is again done in the Pervasive Computing Laboratory (Figure 2.4), the Hong Kong University of Science and Technology. We used 4 Mantis readers (AP) and 30 tags (MD) from RF Code©. They were all deployed as stationary nodes, which is shown in Figure 2.4. All the 30 tags are deployed at 6×5 grid points on the $5.0m \times 4.0m$ floor. The data are sent to a central computer through a hub. The ground truth locations are marked down manually. The signal strength values of RFID have a similar characteristics as wireless sensor networks. However, it is more stable since all tags are deployed in a static environment. RFID experiments are done in [69].

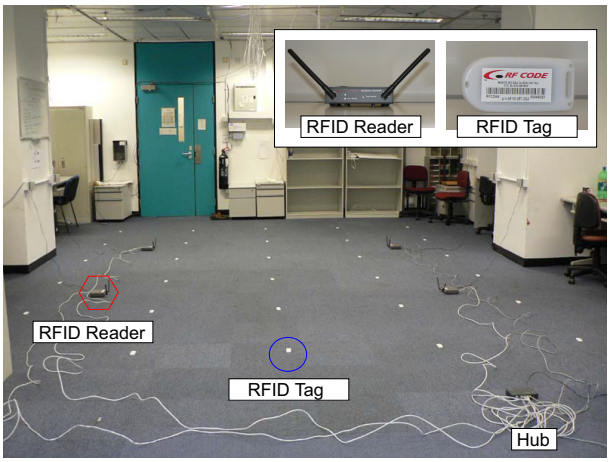


Figure 2.4: RFID Test-bed

2.2 General Architecture of Tracking Systems

We can abstract and classify different localization systems by what is carried in the object. Typical models are receiver-oriented model, transmitter-oriented model, transceiver-oriented model and non-intrusive model. This section is a joint work with Prof. Lionel Ni and his PhD student Jian Ma.

2.2.1 Receiver-Oriented Model

The receiver-oriented model [5, 69, 71, 25, 57] is shown in Figure 2.5, in which the object carries a receiver and receives the messages from different transmitters in the external infrastructure. The object can estimate its distances to these transmitters by measuring the RSS values on received messages. Based on at least three different distance estimations, the object computes its own location. The receiver-oriented model is a natural extension of the GPS mechanism without using timing in-formation. It is mainly used in 802.11-based localization systems, in which the object is usually a person carrying a notebook computer or Personal Digital Assistant (PDA) with 802.11 card and the transmitters are the access points.

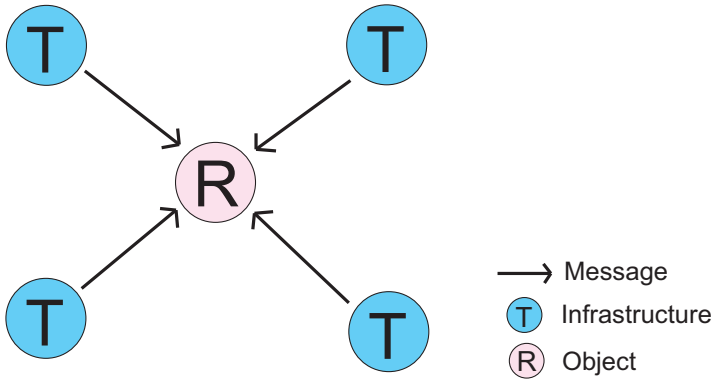


Figure 2.5: Receiver-Oriented Model

2.2.2 Transmitter-Oriented Model

As shown in Figure 2.6, the transmitter-oriented model [65, 50] exchanges the roles of transmitters and receivers in the receiver-oriented model. After receiving the messages from the transmitter car-ried by the object, the receivers in the external infrastructure estimate the distances to the object. By collecting these distance estimations, the external infrastructure computes the location of the object without involving the object in computation. A common scenario for this model is the lo-calization systems that employ Active Radio Frequency Identification (RFID) technique. In RFID-based localization systems, the object carries an active RFID tag periodically broadcasting messages, and the RFID readers are deployed as the receivers. Compared with notebook com-puter and PDA, a RFID tag has very weak computing capability so that the localization computa-tion must be handled by the external infrastructure.

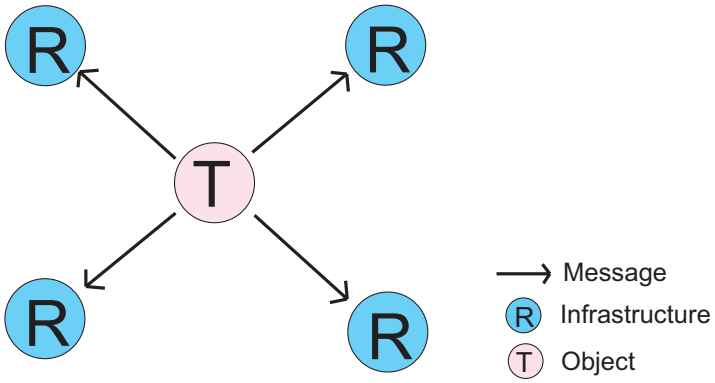


Figure 2.6: Transmitter-Oriented Model

2.2.3 Transceiver-Oriented Model

In the transceiver-oriented model [83, 73, 93] shown in Figure 2.7, an object acts both as a transmitter and as a receiver. As the combination of the receiver-oriented and transmitter-oriented model, the object not only estimates distance based on received messages, but also transmits messages so that other radios can estimate their distances to the object. The common scenario for this model is sensor networks that are commonly dense-deployed without enough infrastructure support. On one side, an object is hard to locate itself within its own neighborhood due to lacking of infra-structure support. On the other side, the object can find many neighbor objects that have their own distance estimation information. It can collaborate with its neighboring objects to solve the localization problem. Furthermore, its neighboring objects may seek the help from their own neighboring transceivers too. In other words, the collaboration could involve of the objects that are multi-hop away. The model reflects the key idea of sensor networks - the collaboration between networked sensor nodes.

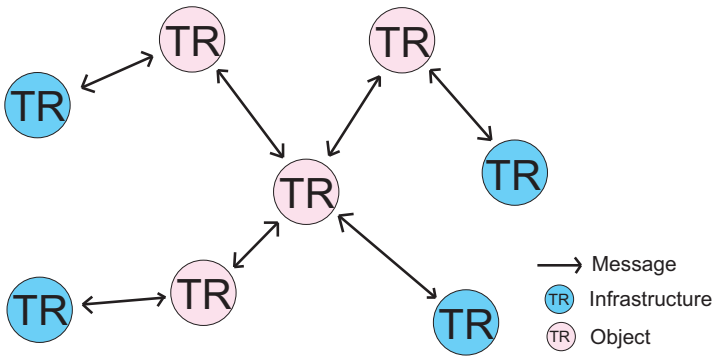


Figure 2.7: Transceiver-Oriented Model

2.2.4 Non-Intrusive Model

Finally, can the object be located without carrying radio or other tag? In many scenarios such as security surveillance, it is inconvenient or even impossible to require an object to carry a radio. This model, called non-intrusive model [104], is shown Figure 2.8. Visual tracking falls into this category. For example, multiple cameras may be deployed to cover an interesting area. These camera have to work collaboratively to track an object since each camera may just sense some part of the area.

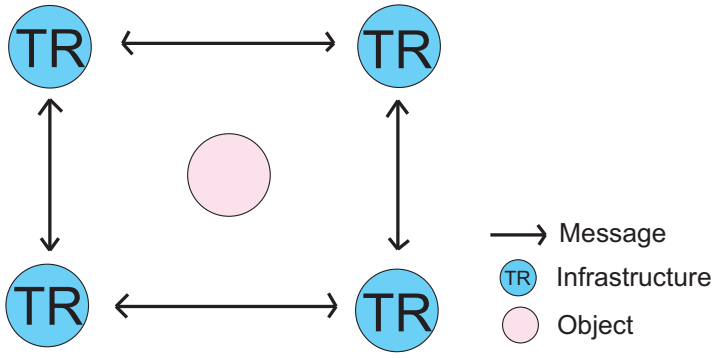


Figure 2.8: Non-Intrusive Model

2.3 General Algorithms of Tracking Systems

Location estimation systems based on radio techniques could be classified into different categories according to different criteria. Considering the system architecture, they could be client-based [5, 51, 62, 102] or infrastructure-based [30, 50]; When emphasizing the use of probability, they could be deterministic [5, 87, 65, 10] or distribution-based [16, 46, 51, 78, 79, 95]. In this chapter, we follow [5] and classify location estimation systems into two main categories: (1) *Propagation-based Models* (2) *Learning-based Models*. The latter may not rely on the knowledge of radio propagation.

2.3.1 Propagation-based Models

Location estimation systems that adopt *Propagation-based Models* benefit from the knowledge of radio propagation. Observing the nonlinear and noisy patterns of radio signal strength, people who develop this kind of systems usually go to a *low-level* analysis and try to identify the hidden factors that cause these patterns. Typical factors

include path loss, shadowing and multi-path, which lead to log-distance , log-normal and more sophisticated multi-path channel models [33, 38, 46, 78]. Generally speaking, a more accurate formula needs additional information about the physical environment and the network. For example, the location of access points needs to be given in many radio propagation models. When the building structure (or even the material) is known, the wall attenuation factors and the corridor effect could be encoded and form a more accurate propagation model [5, 62]. Collecting environment information may raise the calibration effort. There are many algorithms that apply radio propagation models. Take trilateration for example, we could first transform signal into distance information from client to a few number of access points (ranging). Then a least square fit is used to estimate the most probable coordinate [80]. As an alternative, RADAR [5] discretizes the localization area to grid locations and *theoretically* compute the signal strength at these locations in the offline phase. Then, nearest neighbor heuristics and triangulation methods are used to infer a client's location in the online phase. Martin *et al.* [62] explore that different access points should behave similarly. Their work presents a Bayesian network model that encodes knowledge about radio-propagation models, which make use of similarity among the access points and other factors. It also points out that, by incorporating additional knowledge such as the motion constraint of a user, the calibration effort can be further reduced.

Free-Space Propagation Models

As we know, a signal strength value falls off as the signal propagates further. The transmission path between the transmitter and the receiver can vary from simple line-of-sight to one that is severely obstructed by buildings, mountains, and foliages. Various propagation models have been developed to predict the local average received signal strength for an arbitrary transmitter-receiver distance under different environments [38]. In most propagation models, the received power is given as a function of the distance raised to some power. The free space model is the most famous one in which the transmitter and the receiver have a clear, unobstructed line-of-sight path between them. The free space power received by a receiver that is separated from a transmitter by a distance d, is given by the Friis free space equation,

$$P_r = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2} \tag{2.1}$$

where P_t is the transmitted power in watts, P_r is the received power in watts, G_t is the transmitter antenna gain, G_r is the receiver antenna gain, d is the T-R separation distance in meters, and λ is the wavelength in meters. However, a single direct path between the transmitter and the receiver is seldom the only propagation path. The ground reflection model considers both the direct path and a ground reflected propagation path between the transmitter and the receiver. The received power at a T-R separation distance d can be expressed as

$$P_r = \frac{P_t G_t G_r h_t^2 h_r^2}{(4\pi)^2 d^4} \tag{2.2}$$

where h_t is the height of transmitter antenna in meters and h_r is the height of receiver antenna in meters. The two models are typical ideal propagation models, in which RSS is only determined by the transmitter-receiver separation distance. The difference between them is only the rate of signal attenuation.

Indoor Propagation Models

In a realistic environment, many propagation models have been developed to represent different RSS characteristics under various kinds of environments. Here, we introduce two popular propagation models used in an indoor environment. RADAR [5] applied a Wall Attenuation Factor (WAF) model, which is described by the following equation, to represent the relationship between the loss in transmitted signal and the distance.

$$P(d)[dBm] = P(d_0)[dBm] - 10nlog\tfrac{d}{d_0} - \{ \begin{array}{ll} nW * WAF, & nW < C \\ C * WAF, & nW \geq C \end{array} \tag{2.3}$$

where n indicates the rate at which the path loss increases with distance, $P(d_0)$ is the signal power at some distance d and d_0 is the transmitter-receiver separation distance. C is the maximum number of obstructions (walls) up to which the attenuation factor makes a difference, nW is the number of obstructions between the transmitter and the receiver, and WAF is the wall attenuation factor. Based on this relationship, the indoor positioning system can compute the signal strength at every grid of locations as reference radio map. The RADAR then applies k nearest neighbor (KNN) methods to estimate the locations of a mobile node in the online phase.

[1, 39] proposed to use another propagation model to generate a radio map. This propagation model is known as channel model that is for WLAN environment. In this

model, the path loss relationship can be formulated as follows:

$$\begin{aligned} L(d) &= L_{FS}(d) && d \geq d_{BP}, \\ L(d) &= L_{FS}(d_{BP}) - 10\alpha_2 log_{10}\frac{d}{d_{BP}} + x, && d > d_{BP}, \\ L_{FS}(d) &= L_0 + 10\alpha_1 log_{10}(d) \end{aligned} \tag{2.4}$$

where d, d, d_{BP}, α_1 and α_2 represent path loss, distance, breakpoint distance, power-distance gradient, coefficients before and after the breakpoint, respectively. x is the shadow fading component with a zero mean Gaussian probability distribution:

$$f_X(x) = \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{x^2}{2\sigma_x^2}}$$

where σ_x is defined for different models. [27, 66, 3] applied variant indoor propagation models to generate radio maps. The accuracy of these approaches depends on how well the selected propagation model matches the real indoor environment. Thus, the major issue of these approaches is how to build the indoor radio propagation model in a real WLAN environment. If the indoor positioning system considers all the environmental factors (wall, floor, persons' moving), this could make the propagation model too complicated to be solved. However, if the system uses a simple version of propagation model, this could make the model inaccurate in a complex indoor environment.

2.3.2 Learning-based Models

Another class of location estimation systems is based on *Learning-based Models*, which employ machine learning techniques. Path loss, shadowing and multipath are caused by a complicated underlying process in a complex environment. When all these factors are mixed together, they show a *high-level* of nonlinear and noisy patterns. These patterns can be captured when sufficient empirical data are manually collected at different locations [102, 16] or automatically by additional hardware [50, 65, 99]. These methods need less information about physical layout and network configuration. The input is usually implicitly encoded into radio maps [102] at different locations. In such cases, the locations of access points are not needed. Typical pattern descriptions include histogram [102, 79, 16], mixture of gaussian[79], kernel matrix [67], Akima Spline [50], or simply the mean value of signal strength at different locations [5, 65].

For example, the LANDMARC system [65] uses reference tags to dynamically construct and update radio map. It alleviates the effects caused by the fluctuation in RF signal strength. The method first computes the distances between the signal-strength vectors received from the tracking tags and those from different reference-tags' respectively. It then uses k nearest reference tags' coordinates to calculate the approximate coordinate of the tracking tag. A similar technique is used in LEASE [50] and [99].

2.3.3 User Models and Others

More recently, intense research efforts have been taken to seek additional knowledge in order to boost the accuracy of location determination systems. For example, [4, 43, 51, 62, 95] take the sequential characteristics of user traces into consideration by posing a reasonable assumption that a user could not walk irregularly, run too fast or go through a wall. These models come closer to the general framework of Bayesian filtering [27]. For example, [51] suggests a sensor fusion model and shows a strong correlation between consecutive locations. The robotics-based location sensing system in [51] applies Bayesian inference to compute the conditional probabilities over locations based on received signal-strength samples from various access points. Then a post-processing step, which utilizes the spatial constraints of a user's movement trajectories, refines the location estimation and rejects the estimates that show significant changes in the physical location space. Depending on whether the post-processing step is used or not, the accuracy of this method is 83% or 77% within 1.5 meters. [4, 16, 50, 65, 99, 101] study the dynamic features of signal strength and update their models to adapt the change. [50, 65, 99] employ additional hardware such as sniffers to help recalibrate the radio map periodically. [4] improve the localization performance by profiling the radio characteristics into "busy" and "non-busy" hours. [16, 62] could use unlabelled data to advance the performance so that they are capable to dynamically update their model to fit the characteristics of the radio environment. [101] use an autoregressive model to reduce fluctuation between consecutive signal strength.

In practice, location-estimation systems may combine both propagation-based and learning-based models, and capture the signals and the user dynamics by filtering in both signal and location spaces. Thus, the above models can *complement* each other and improve the performance. A summary of related works is shown in Table 2.1. The table shows a high-level overview of different location-estimation algorithms. Thus,

items in the table only highlight the main and common features of these algorithms. In this table, some papers propose more than one algorithm so that they fall into multiple categories. For example, RADAR [5] and its enhanced version [4] have two variants : one is based on radio propagation and the other is based on machine learning, although both use K-Nearest-Neighbor in the online localization phase.

Table 2.1: A Summary of Common Location Estimation Methods.

Method	Propagation-based	Signal dynamics	User dynamics
Multilateration [80], Statistical [78], Fingerprinting [46], RADAR [5]	✓		
RADAR+ [4], Bayesian-Indoor [62]	✓	✓	
Active-Campus [10], PHD [87], Robust-Indoor [34], RADAR+ [4]	✓		✓
RADAR [5], Horus [102], Probabilistic [79], KCCA [67]			
LEASE[50], Reduce[16], Horus+[101], Adaptive [99], LANDMARC [65]		✓	
RADAR+ [4], MCL [43], Robotics [51], State-Machine [95]			✓

2.4 Comparison of Propagation and Learning Models

In general, RSS-based location estimation systems have two phases: an offline training phase and an online localization phase. In the offline training phase, a radio map is built for location estimation. In the online localization phase, the radio map is used to estimate the location of mobile nodes. The performance of the system relies on the accuracy of the underlying radio map. Based on different methods to generate a reference radio map in indoor environment, localization approaches can be classified into two categories: propagation model based approaches and learning model based approaches.

2.4.1 Comparison of Calibration Effort

We define three types of calibration effort in an indoor positioning system: calibration of AP locations, calibration in generating the radio map and calibration of adapting the system to dynamic environment. The comparison of calibration effort between propagation-based models and learning-based models is shown in Table 2.2.

We can see from the table, propagation-based models need to know all APs' locations, since these models need to calculate the distances from APs to mobile nodes, while learning-based models do not need to know these information. This is convenient for client-based system. However, since the motivation of learning-based models is to learn knowledge from data automatically, thus these models need a lot of training data (the on site measurements at various locations) to extract the knowledge. In order to reduce the calibration effort in generating the radio map, [70, 25] apply semi-supervised and unsupervised machine learning techniques for this propose, respectively. Because of the properties of indoor RSS [47], the values of RSS can change significantly over different time periods. For propagation-based models, the path loss function represents the relationship between RSS and transmitter-receiver separation distance. If the value of RSS at transmitter changes, the value of RSS at receiver would change. This makes the radio map always up to date. However, for learning-based models, the mapping function is learnt from the out-of-date data, this could make localization performance of the system be inaccurate in new time period. In order to deal with this shortcoming, [35, 99] propose different adapting methods to update the mapping function while minimize the additional calibration.

Table 2.2: Calibration Effort Comparison

	Calibration of AP Locations	Calibration in Generating the Radio Map	Calibration of Adapting the Positioning System to Dynamic Environment
Propagation-based Models	Need to know	Simulate signal strength for various locations by propagation models. Either collect signal strength at a few location to tune the coefficient of the propagation model or use the default coefficient by prior knowledge.	Do not need additional calibration effort. Since the path loss function represents the relationship between RSS and transmitter-receiver (T-R) separation distance. When the value of RSS at transmitter changes, the value of RSS at receiver changes at the same time. [21]
Learning-based Models	Do not need to know	Need to collect signal strength at various locations	Need to recollect several signal strength at various locations. [4]

As mention above, propagation-based models need to consider environmental factors. For instance, in Equation 2.3 and Equation 2.4, the positioning system determines a client location by considering different indoor environments. If the propagation model does not fit the indoor environment well, the localization performance is

poor. However, in a complex indoor environment, it is difficult to formulate the propagation model perfectly. Learning-based models do not need know anything about environmental factors. Most of these models can consider these factors implicitly in the learning process.

2.4.2 Comparison of Localization Performance

Table 2.3: Performance Comparison of different Models

Reference	Propagation-based Model	Learning-based Model	Test Bed (m^2)	APs
[5]	4.3 m (50th percentile)	2.94 (50th percentile)	22.5×43.5, > 50 rooms	3
[66]	1.53m (mean)	1.08m (mean)	22×34, 4 hallways	3
[3]	4.62m (mean)	1.78m (mean)	11×23, 7 rooms	5
[61]	Up to 12m	2.45m (mean)	60×60, 50 rooms	7

Table 2.3 shows a performance comparison between propagation-based models and learning-based models from some previous works [5, 66, 3, 61].

We also compare the performance of propagation-based model and leraning-based model in an 802.11 WLAN as shown in Figure 2.9. A person carried an IBM T42 laptop which is equipped with an Intel Pro/2200GB internal wireless card and walked in the environment. A total of 2000 examples are collected sequentially with sample rate 2Hz. The ground-truth locations are obtained by referring to landmark points such as doors, corners and dead-ends.

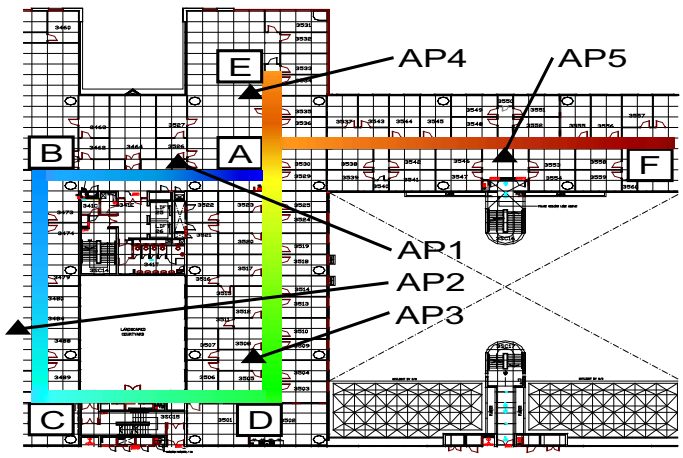


Figure 2.9: WLAN Test-bed

For propagation-based model, we use a subset of examples to fit the parameters and generate the radio map. For learning-based model, we just the subset as the training data. In the online localization phase, we use k-nearest-neighbor for location estimation for both compared models. In each experiment, we randomly pick up a subset of examples for training and the rest for testing. Figure 2.10 shows the error distance while varying the number of examples. All results are averaged over 10 repetitions for reducing statistical variability. As can be seen, learning-based model has a smaller error distance than the propagation-based model. The performance of Propagation-based model converges with less number of calibration data because it has a simpler parameter space (degree of freedom). Therefore, our experimental results lead to a similar conclusion compared to previous study in Table 2.3.

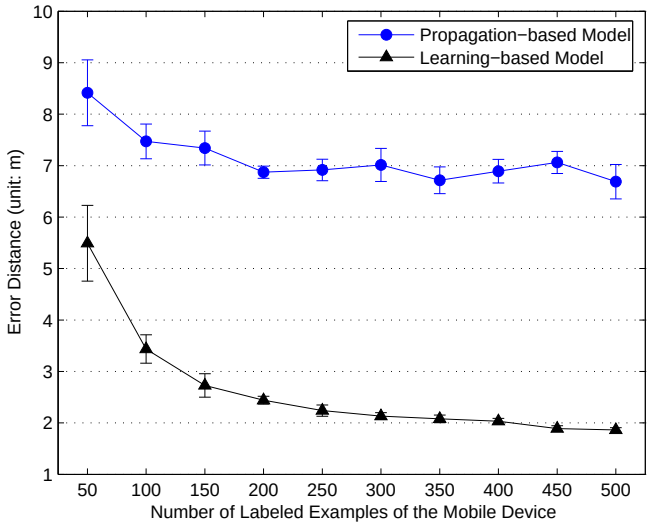


Figure 2.10: Comparison Result of Propagation-based and Learning-based Models

2.5 Localization and User Behavior Analysis

Location estimation and use behavior recognition are research problems that are closely related. In the past, they have been separately investigated. The current action and potential goal of a user may be reflected on his or her recent locations. For example, a professor is likely to give a course in a classroom and take some drink in a cafe. Location is important information for understanding the user behavior. In the following, we survey some user behavior recognition models that are related to location estimation. Figure 2.11 illustrates a person with three sensor nodes attached in an indoor

environment for behavior analysis. Sensor readings such as temperature, light, sound and acceleration can be collected from these sensor nodes.

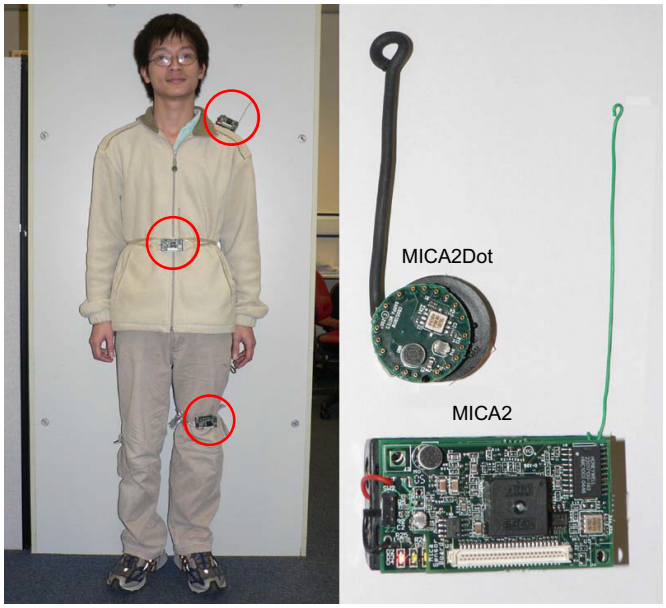


Figure 2.11: Sensor-based User Behavior Analysis

2.5.1 User Behavior Analysis in Wireless Local Area Networks

Yin et al. [97, 96] presents an integrated plan-recognition model that combines low-level sensory readings with high-level goal inference. A two-level architecture is proposed to infer a user's goals in a complex indoor environment using an RF-based wireless network. The novelty of their work derives from the ability to infer a user's goals from sequences of signal trajectory, and the ability to make a tradeoff between model accuracy and inference efficiency. The model relies on a dynamic Bayesian network to infer a user's actions from raw signals, and an N-gram model to infer the users' goals from actions.They presents a method for constructing the model from the past data and demonstrate the effectiveness of the proposed solution through empirical studies using some real data collected.

Chai et al. [15] breaks a common assumption made by most approaches: a user either has a single goal in mind, or achieves several goals sequentially. However, in real-world environments, a user often has multiple goals that are concurrently carried out, and a single action can serve as a common step towards multiple goals. They formulate the multiple-goal recognition problem and exemplify it in an indoor environment where an RF-based wireless network is available. They propose a goal-recognition

algorithm based on a dynamic model set and show how goal models evolve over time based on pre-defined states. Experiments with real data demonstrate that their method can accurately and efficiently recognize multiple interleaving goals in a user's trace.

A major issue in activity recognition in a sensor network is how to automatically segment the low-level signal sequences in order to optimize the probabilistic recognition models for goals and activities. Yin et al. [98] point out that past efforts have relied on segmenting the signal sequences by hand, which is both time-consuming and error-prone. Segments should correspond to atomic human activities that enable a goal-recognizer to operate optimally; the two are intimately related. They present a novel method for building probabilistic activity models at the same time as segmenting signal sequences into motion patterns. They model each motion pattern as a linear dynamic model and the transitions between motion patterns as a Markov process conditioned on goals. They use EM learning algorithm to simultaneously learn the motion-pattern boundaries and probabilistic models for goals and activities, which in turn can be used to accurately recognize activities in an online phase. A major advantage of the algorithm is that it can reduce the human effort in segmenting and labeling signal sequences. They demonstrate the effectiveness of the algorithm using the data collected in a real wireless environment.

A summary of the above methods is shown in Table 2.4.

Table 2.4: WLAN-based User Behavior Analysis

	Motivation	Model	Advantage	Disadvantage
[97]	Traditional plan recognition could not well adapt to uncertainty.	DBN	No sliding-windows	Could not deal with multiple goals.
[15]	People usually have multiple goals	Dynamic Model Set	Deal with multiple goals.	Model set is not easy to control.
[98]	building probabilistic activity models at the same time as segmenting signal sequences into motion patterns	LDS+EM	No localization needed. Reduce calibration effort	Time-consuming

2.5.2 User Behavior Analysis in Wireless Sensor Networks

Accurate recognition and tracking of human activities is an important goal of ubiquitous computing. Recent advances in the development of multi-modal wearable sensors

enable us to gather rich datasets of human activities. However, the problem of automatically identifying the most useful features for modeling such activities remains largely unsolved.

Lester et al. [56] present a hybrid approach to recognizing activities, which combines boosting to discriminatively select useful features and learn an ensemble of static classifiers to recognize different activities, with hidden Markov models (HMMs) to capture the temporal regularities and smoothness of activities. They tested the activity recognition system using over 12 hours of wearable-sensor data collected by volunteers in natural unconstrained environments. The models succeeded in identifying a small set of maximally informative features, and were able identify ten different human activities with an accuracy of 95%. Related hybrid model refer to [2, 77]

Taipa et al. [91] present a system for recognizing activities in the home setting using a set of small and simple state-change sensors is introduced. The sensors are designed to be "tape on and forget" devices that can be quickly and ubiquitously installed in home environments. The proposed sensing system presents an alternative to sensors that are sometimes perceived as invasive, such as cameras and microphones. Unlike prior work, the system has been deployed in multiple residential environments with non-researcher occupants. Preliminary results on a small dataset show that it is possible to recognize activities of interest to medical professionals such as toileting, bathing, and grooming with detection accuracies ranging from 25% to 89% depending on the evaluation criteria used.

The ability to determine what day-to-day activity a person is performing is of interest in many application domains. A system that can do this requires models of the activities of interest, but model construction does not scale well: humans must specify low-level details, such as segmentation and feature selection of sensor data, and high-level structure, such as spatio-temporal relations between states of the model, for each and every activity. As a result, previous practical activity recognition systems have been content to model a tiny fraction of the thousands of human activities that are potentially useful to detect.

Perkowitz et al. [74] present an approach to sensing and modeling activities that provides scalability for a much larger class of activities than before. They show how a new class of sensors, based on Radio Frequency Identification (RFID) tags, can directly yield semantic terms that describe the state of the physical world. These sensors allow

them to formulate activity models by translating labeled activities, such as “cooking pasta”, into probabilistic collections of object terms, such as “pot”. Given this view of activity models as text translations, they show how to mine definitions of activities in an unsupervised manner from the web. They have used our technique to mine definitions for over 20,000 activities. They experimentally validate our approach using data gathered from actual human activity as well as simulated data.

A fundamental difficulty in recognizing human activities is obtaining the labeled data needed to learn models of those activities. Given emerging sensor technology, however, Wyatt et al. [94] show that it is possible to view activity data as a stream of natural language terms. Activity models are then mappings from such terms to activity names, and may be extracted from text corpora such as the web. They show that models so extracted are sufficient to automatically produce labeled segmentations of activity data with an accuracy of 42% over 26 activities, well above the 3.8% baseline. The segmentation so obtained is sufficient to bootstrap learning, with accuracy of learned models increasing to 52%. To our knowledge, this is the first human activity inferencing system shown to learn from sensed activity data with no human intervention per activity learned, even for labeling.

A summary of the above methods is shown in Table 2.5.

Table 2.5: WSN-based User Behavior Analysis

	Motivation	Model	Advantage	Disadvantage
[56]	Recognize basic human activities	Generative-Discriminative Hybrid	fast and accurate	could not well recognize high-level activities
[91]	Use simple sensors that detect changes in state of objects and devices to recognize activities	Naive Bayesian	Simple	Sliding Window size is not easy to set reasonably since different activities have different duration
[74]	labeling is expensive, time-consuming	Plan Generated from the Web, Inference by DBN	No Training, recognize thousands of plans	Format of Web text should be well organized for generating plans. Traces are hand-segmented.
[94]	Plans are usually not described step by step on web.	Web Mining + HMM	Requires no additional input beyond the natural language names of activities and tags	

CHAPTER 3

SUPERVISED LOCALIZATION USING KERNELS

In this chapter, we present an algorithm for increasing the accuracy of machine-learning based localization. We first review related works in this area. We then present our approach to solving this problem, using multidimensional vector regression on data that are highly uncertain and nonlinear.

3.1 The Inaccuracy Problem

In localization, our aim is to obtain an accurate mapping between the signal space and the physical space without requiring too much human calibration effort. This location estimation problem has traditionally been tackled through probabilistic models trained on manually labelled data, which are expensive to obtain. The problem of indoor location estimation can be considered as one of building a mapping from radio-frequency signals to a multi-valued function that corresponds to locations. Indoor location estimation is a major area of pervasive computing applications that range from context-dependent content delivery to object tracking [20] and people monitoring [58, 97]. Many systems utilize the signal strength values received from the access points to infer the location of mobile devices [5, 31, 51, 65, 102, 18]. Similarly, in a sensor network, the location information must also be inferred from signals received from various deployed sensors [43, 80].

In general, location-estimation systems using radio frequency (RF)-based signal-strength values function in two phases: an *offline training phase* and an *online localization phase*. In the offline training phase, data are collected that correspond to each location and labelled by hand with the location information. Then a model is trained by considering the signal strength values received from the access points at selected locations in the area of interest. In the online localization phase, the learned model is applied to new signals. The real-time signal strength samples received from the access points are used to estimate the current location based on the learned model.

Models used in location-estimation systems could be broadly categorized into two classes: *Radio-Propagation Models* and *Empirical-Fit Models*. For building *Radio-Propagation Models*, we usually try to identify different factors that may affect the fluctuation and attenuation of radio signal and encode these factors as parameters into radio propagation equations [34, 46, 62, 79]. For building *Empirical-Fit Models*, we would empirically describe the signals in terms of their mean values, histogram or spline interpolation without formulating a closed-form propagation model [5, 50, 51, 65, 102]. In order to calibrate both kinds of models, efforts need be spent to collect signal samples from different locations. Thus, how to maintain a high-level of accuracy while reducing calibration effort is a challenging task, since the data collection process is time-consuming. For example, in our earlier test it took many hours to collect and label the signal-strength data in a small indoor environment. Furthermore, the mapping between the signal and physical location spaces is very difficult to construct with only limited labelled data due to uncertainty and nonlinearity. Nonlinearity exists when similar locations have very different signal signatures. Therefore, a direct mapping between the two spaces may not always work well, even when much data are collected.

In this chapter, we present a multidimensional vector regression method for building a mapping function by addressing the problems of uncertainty and nonlinearity directly. Our main intuition is to perform a kernel-based transformation of the signal and physical location spaces to capture the nonlinear relationship between the signals and locations. Furthermore, we perform kernel canonical correlation analysis (KCCA) in the two spaces for feature extraction. This allows the pairwise similarity of samples in both spaces to be maximally correlated. We use a Gaussian kernel to adapt to the noisy characteristics of signal strengths and a Matérn kernel to sense the changes in physical locations. In comparison to previous methods, a major advantage of our proposed technique is that we can obtain higher accuracy while reducing the training cost by requiring only a fraction of the labelled samples.

Although this chapter focuses on the location estimation problem in pervasive computing, the proposed method has more general practical implications. In effect, we are addressing a new type of machine learning problems where there are many classes but limited amount of training examples. For this type of problems, traditional machine learning algorithms will usually result in overfitting. Our idea is to make use of the inherent relationships between the different classes. In this paper, we rely on the similarity relationship imposed by distance relations among the different locations. As will

be shown later in the paper, this relationship can be captured using kernel functions [81]. Consequently, the accuracy of the classification model can be greatly enhanced with the use of this new piece of information.

The rest of this chapter is organized as follows. Section 3.2 then describes the proposed KCCA-based location estimation algorithm. Results on a series of WLAN location estimation experiments using data collected in a realistic environment are presented in Section 3.3, and the last section gives some concluding remarks.

3.2 Methodology

3.2.1 Problem Statement

Consider the two-dimensional location estimation problem[1]. Its goal is to obtain a mapping between the space of signal strengths obtained at p access points

$$\mathcal{S} = \{\mathbf{s} \equiv [s_1, s_2, \ldots, s_p]' \in \mathbb{R}^p\} \tag{3.1}$$

and the physical location space $\mathcal{P} = \{\boldsymbol{\ell} \equiv [x, y]' \in \mathbb{R}^2\}$. Methods such as trilateration deal with the mapping between signal and location spaces in two step: first transform signal into distance with a nonlinear equation (ranging) and then recover the most probable coordinate from distance with least square methods. In this chapter, we directly build a nonlinear mapping between signal and location spaces. We consider x and y together and emphasize the *correlation* between signal and physical location spaces, observing that the pairwise similarity in the signal space should match the pairwise similarity in the physical location space. For example, in Figure 3.1, signal S_A should be more similar to S_B than S_C, since A is closer to B in the physical location space. Consequently, we consider both x and y together and use KCCA to learn the mapping between the two spaces.

3.2.2 (Kernel) Canonical Correlation Analysis

Canonical Correlation Analysis (CCA)

Given two sets of variables $\mathbf{x}$ and $\mathbf{y}$, canonical correlation analysis (CCA) [42] attempts to find a basis for each set such that the correlation between the projections of the

[1]Extension to the three-dimensional (or even higher-dimensional) case is straight-forward.

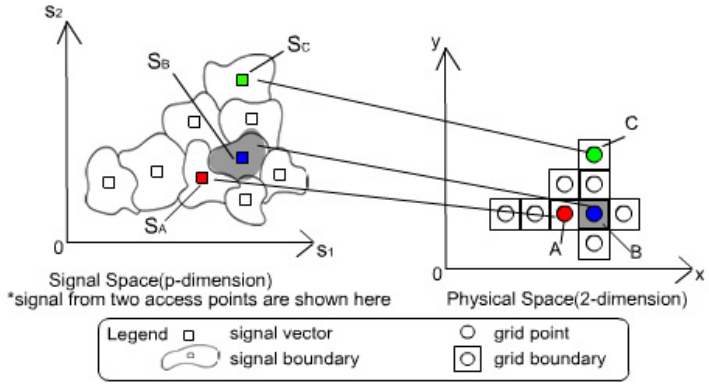


Figure 3.1: Correlation between the signal and physical location spaces.

variables onto these basis vectors are mutually maximized. Mathematically, given n instances

$$S = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\},$$

CCA finds directions (*canonical vectors*) $\mathbf{w_x}$ and $\mathbf{w_y}$ so that the sets of transformed variables (*canonical variates*)

$$S_\mathbf{x} \equiv S_\mathbf{x}(\mathbf{w_x}) = (\langle \mathbf{w_x}, \mathbf{x}_1 \rangle, \langle \mathbf{w_x}, \mathbf{x}_2 \rangle, \ldots, \langle \mathbf{w_x}, \mathbf{x}_n \rangle)$$

and

$$S_\mathbf{y} \equiv S_\mathbf{y}(\mathbf{w_y}) = (\langle \mathbf{w_y}, \mathbf{y}_1 \rangle, \langle \mathbf{w_y}, \mathbf{y}_2 \rangle, \ldots, \langle \mathbf{w_y}, \mathbf{y}_n \rangle)$$

are maximally correlated.

Define the total covariance matrix by[2]

$$\mathbf{C} = \hat{\mathbb{E}}\left[\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}'\right] = \begin{bmatrix} \mathbf{C_{xx}} & \mathbf{C_{xy}} \\ \mathbf{C_{yx}} & \mathbf{C_{yy}} \end{bmatrix},$$

where $\hat{\mathbb{E}}[\cdot]$ is the empirical expectation operator, $\mathbf{C_{xx}}, \mathbf{C_{yy}}$ are the within-sets covariance matrices and $\mathbf{C_{xy}} = \mathbf{C}'_{\mathbf{yx}}$ is the between-sets covariance matrix. The correlation

[2]In this chapter, vector/matrix transpose is denoted by the superscript '.

coefficient between $S_{\mathbf{x}}$ and $S_{\mathbf{y}}$ can be written as

$$\rho = \frac{\hat{\mathbb{E}}[\langle \mathbf{w_x}, \mathbf{x} \rangle \langle \mathbf{w_y}, \mathbf{y} \rangle]}{\sqrt{\hat{\mathbb{E}}[\langle \mathbf{w_x}, \mathbf{x} \rangle^2] \hat{\mathbb{E}}[\langle \mathbf{w_y}, \mathbf{y} \rangle^2]}} \tag{3.2}$$

$$= \frac{\mathbf{w}_{\mathbf{x}}' \hat{\mathbb{E}}[\mathbf{x}\mathbf{y}'] \mathbf{w_y}}{\sqrt{\mathbf{w}_{\mathbf{x}}' \hat{\mathbb{E}}[\mathbf{x}\mathbf{x}'] \mathbf{w_x} \cdot \mathbf{w}_{\mathbf{y}}' \hat{\mathbb{E}}[\mathbf{y}\mathbf{y}'] \mathbf{w_y}}}$$

$$= \frac{\mathbf{w}_{\mathbf{x}}' \mathbf{C_{xy}} \mathbf{w_x}}{\sqrt{\mathbf{w}_{\mathbf{x}}' \mathbf{C_{xx}} \mathbf{w_x} \cdot \mathbf{w}_{\mathbf{y}}' \mathbf{C_{yy}} \mathbf{w_y}}}. \tag{3.3}$$

It can be shown that $\mathbf{w_x}$ can be obtained by solving the generalized eigenproblem [37]

$$\mathbf{C_{xy}} \mathbf{C_{yy}^{-1}} \mathbf{C_{yx}} \mathbf{w_x} = \lambda^2 \mathbf{C_{xx}} \mathbf{w_x}.$$

Subsequently, $\mathbf{w_y}$ can then be obtained as $\mathbf{w_y} = \frac{1}{\lambda} \mathbf{C_{yy}^{-1}} \mathbf{C_{yx}} \mathbf{w_x}$. Moreover, it can be shown that the λ obtained is equal to the ρ in (3.2).

Kernel Canonical Correlation Analysis (KCCA)

A major limitation of CCA is that it can only exploit linear relationships between $\mathbf{x}$ and $\mathbf{y}$. As is now well-known, the use of kernels offers efficient, nonlinear extensions for many standard linear procedures [81]. In kernel canonical correlation analysis (KCCA) [37] implicitly maps $\mathbf{x}$ and $\mathbf{y}$ to $\phi_x(\mathbf{x})$ and $\phi_y(\mathbf{y})$, and then performs traditional CCA in the two high-dimensional feature spaces. Using the dual representations for the projection directions $\mathbf{w}_{\phi_x(\mathbf{x})}$ and $\mathbf{w}_{\phi_y(\mathbf{y})}$:

$$\mathbf{w}_{\phi_x(\mathbf{x})} = \mathbf{S}_{\phi_x(\mathbf{x})}' \boldsymbol{\alpha}, \quad \text{and} \quad \mathbf{w}_{\phi_y(\mathbf{y})} = \mathbf{S}_{\phi_y(\mathbf{y})}' \boldsymbol{\beta},$$

where $\mathbf{S}_{\phi_x(\mathbf{x})} = [\phi_x(\mathbf{x}_1), \ldots, \phi_x(\mathbf{x}_n)]'$, $\mathbf{S}_{\phi_y(\mathbf{y})} = [\phi_y(\mathbf{y}_1), \ldots, \phi_y(\mathbf{y}_n)]'$, and $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^n$. Denote the corresponding kernel functions by $k_{\mathbf{x}}(\cdot, \cdot)$ and $k_{\mathbf{y}}(\cdot, \cdot)$, and the kernel matrices (defined on all n instances) by $\mathbf{K_x}$ and $\mathbf{K_y}$. The kernelized counterpart of (3.3) is

$$\rho = \frac{\boldsymbol{\alpha}' \mathbf{K_x} \mathbf{K_y} \boldsymbol{\beta}}{\sqrt{\boldsymbol{\alpha}' \mathbf{K_x^2} \boldsymbol{\alpha} \cdot \boldsymbol{\beta}' \mathbf{K_y^2} \boldsymbol{\beta}}} \tag{3.4}$$

which is then maximized w.r.t. $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. However, it can be shown that one can always obtain perfect correlation and consequently uninteresting result given that the kernel matrices $\mathbf{K_x}$ and $\mathbf{K_y}$ are invertible [37]. To control the flexibility of the projections, the

norms of the associated weight vectors are thus penalized as in other kernel methods. Hence, instead of maximizing (3.4), we maximize the regularized version

$$\frac{\boldsymbol{\alpha}'\mathbf{K}_\mathbf{x}\mathbf{K}_\mathbf{y}\boldsymbol{\beta}}{\sqrt{(\boldsymbol{\alpha}'\mathbf{K}_\mathbf{x}^2\boldsymbol{\alpha}+\kappa\boldsymbol{\alpha}'\mathbf{K}_\mathbf{x}\boldsymbol{\alpha})\cdot(\boldsymbol{\beta}'\mathbf{K}_\mathbf{y}^2\boldsymbol{\beta}+\kappa\boldsymbol{\beta}'\mathbf{K}_\mathbf{y}\boldsymbol{\beta})}}, \tag{3.5}$$

where κ is a user-defined regularization parameter. It can be shown that $\boldsymbol{\alpha}$ can be obtained by solving the generalized eigenproblem

$$(\mathbf{K}_\mathbf{x}+\kappa\mathbf{I})^{-1}\mathbf{K}_\mathbf{y}(\mathbf{K}_\mathbf{y}+\kappa\mathbf{I})^{-1}\mathbf{K}_\mathbf{x}\boldsymbol{\alpha}=\lambda^2\boldsymbol{\alpha}, \tag{3.6}$$

where $\mathbf{I}$ is the identity matrix. Subsequently, $\boldsymbol{\beta}$ can be obtained as

$$\boldsymbol{\beta}=\frac{1}{\lambda}(\mathbf{K}_\mathbf{y}+\kappa\mathbf{I})^{-1}\mathbf{K}_\mathbf{x}\boldsymbol{\alpha}. \tag{3.7}$$

Given any $\tilde{\mathbf{x}}$, its projection on $\mathbf{w}_{\phi_x(\mathbf{x})}$ is given by

$$P_\mathbf{x}(\tilde{\mathbf{x}})=\phi_x(\tilde{\mathbf{x}})'\mathbf{w}_{\phi_x(\mathbf{x})}=\mathbf{k}'_{\tilde{\mathbf{x}}}\boldsymbol{\alpha}, \tag{3.8}$$

where $\mathbf{k}_{\tilde{\mathbf{x}}}=[k_\mathbf{x}(\tilde{\mathbf{x}},\mathbf{x}_1),k_\mathbf{x}(\tilde{\mathbf{x}},\mathbf{x}_2),\ldots,k_\mathbf{x}(\tilde{\mathbf{x}},\mathbf{x}_n)]'$. Similarly, the projection of any $\tilde{\mathbf{y}}$ onto $\mathbf{w}_{\phi_y(\mathbf{y})}$ is

$$P_\mathbf{y}(\tilde{\mathbf{y}})=\phi_y(\tilde{\mathbf{y}})'\mathbf{w}_{\phi_y(\mathbf{y})}=\mathbf{k}'_{\tilde{\mathbf{y}}}\boldsymbol{\beta},$$

where $\mathbf{k}_{\tilde{\mathbf{y}}}=[k_\mathbf{y}(\tilde{\mathbf{y}},\mathbf{y}_1),k_\mathbf{y}(\tilde{\mathbf{y}},\mathbf{y}_2),\ldots,k_\mathbf{y}(\tilde{\mathbf{y}},\mathbf{y}_n)]'$.

The generalized eigenproblem in (3.6) can be solved by using the (complete) Cholesky decomposition [75]. However, the kernel matrices $\mathbf{K}_\mathbf{x}$ and $\mathbf{K}_\mathbf{y}$ are of size n, and so obtaining the Cholesky decomposition can become computational expensive for large training sets. In this case, the incomplete Cholesky decomposition or partial Gram-Schmidt orthogonalization (PGSO) can be used instead [37]. The basic idea is to find a low-rank approximation of the kernel matrix, and the incomplete Cholesky decomposition differs from the complete Cholesky decomposition in that all pivots below a certain threshold are simply skipped. The decomposition is obtained by picking columns of the kernel matrix one at a time, at each step choosing the column that leads to the greatest reduction in the approximation error. Once a column (or basis vector) is selected, the other columns are then orthogonalized by the Gram-Schmidt algorithm.

3.2.3 The LE-KCCA Algorithm

Our approach builds a similarity based mapping function by making full use of the continuous location information in kernel-based transformation.

Offline Training Phase

In the training phase, the following steps are taken:

1. Signal strengths are collected at various grid locations.

2. KCCA, with appropriate choices for the two kernels (Section 3.2.4), is used to learn the relationship between the signal and physical location spaces. In particular, λ_i's and $\boldsymbol{\alpha}_i$'s are obtained from the generalized eigenproblem in (3.6), and the corresponding $\boldsymbol{\beta}_i$'s from (3.7).

3. For each training pair $(\mathbf{s}_i, \boldsymbol{\ell}_i)$, its projections

$$P(\mathbf{s}_i) = [P_1(\mathbf{s}_i), P_2(\mathbf{s}_i), \ldots, P_T(\mathbf{s}_i)]' \tag{3.9}$$

   on the T canonical vectors are obtained from (3.8).

Online Localization Phase

In the localization phase, the location of a new signal strength vector $\tilde{\mathbf{s}}$ is estimated as follows:

1. Use (3.8) to project $\tilde{\mathbf{s}}$ onto the canonical vectors and obtain

$$P(\tilde{\mathbf{s}}) = [P_1(\tilde{\mathbf{s}}), P_2(\tilde{\mathbf{s}}), \ldots, P_T(\tilde{\mathbf{s}})]'.$$

2. Among the projections (3.9) from the training samples, find the K neighbors closest to $P(\tilde{\mathbf{s}})$. In this chapter, the weighted Euclidean distance

$$d_i = \sum_{j=1}^{T} \lambda_j (P_j(\tilde{\mathbf{s}}) - P_j(\mathbf{s}_i))^2 \tag{3.10}$$

   is employed in finding the neighbors.

3. Interpolate these neighbors' physical locations to predict the physical location of $\tilde{\mathbf{s}}$. In this chapter, we simply output the median (or mean for continuous location estimation) of the (x, y) coordinates of these K neighbors.

Note that this is essentially a variant of the weighted K-nearest neighbor method. Here, we use λ_j as the weight in (3.10). As mentioned in Section 3.2.2, λ_j is equal to the correlation coefficient ρ in (3.2), and thus serves as a reasonable measure for the importance of the corresponding canonical vector.

3.2.4 Choice of Kernels

The choice of kernels is highly dependent on the nonlinear and noisy characteristics of the location estimation problem we are addressing due to possible path loss, shadowing and multipath effects, etc [33]. In Figure 3.2(a), there are three access points (APs). Figures 3.2(b) and (c) show the average signal strength distributions of AP1 and AP2 respectively. As can be seen, the signal strength changes sharply (nonlinear) at the corner intersecting hallways 2 and 3 due to shadowing effect of the walls. Figure 3.2(d) shows the typical signal distribution at a particular location from a fixed access point. As can be seen, this noisy signal can be as weak as -36dB and as strong as -10dB. Its empirical distribution (*radio map*) is thus difficult to obtain, especially when the training samples are scarce. As a first approximation, the Gaussian distribution has been used in characterizing the nonlinearity of the signal strengths [79]. Hence, in this chapter, we also use the Gaussian kernel

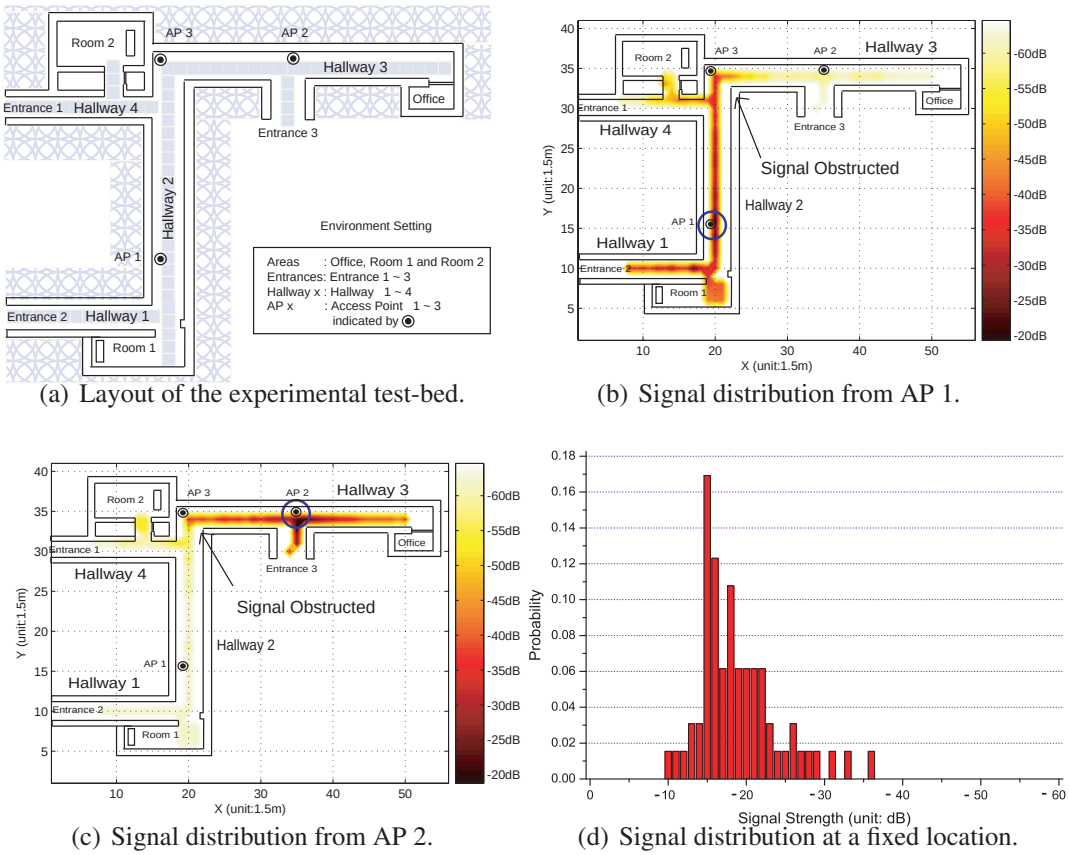
$$G(\mathbf{s}_1, \mathbf{s}_2) = \exp(-w_G^2 \|\mathbf{s}_1 - \mathbf{s}_2\|^2) \tag{3.11}$$

for the signal space. Here, $\|\cdot\|$ denotes the Euclidean norm and w_G is a user-defined parameter that reflects the smoothness of the radio map.

On the other hand, measurements of the physical locations are relatively clean. Intuitively, it seems that Euclidean distance best represents the "similarity" between two locations. Unfortunately, Euclidean function is not a valid kernel since a valid kernel should satisfy the positive definite property [32]. For the commonly used Gaussian kernel, it has been argued by [89] that sample paths of Gaussian model are "infinitely smooth", thus often leading to unreasonably low predictive variance [82]. Consider the Euclidean distance $|x_i - x_j|$. This distance measure cannot be used directly as a kernel without first transforming it into a valid kernel. Therefore, in this chapter, we adopt the Matérn kernel, which is a function that reflects the Euclidean distance [82]. (Figure 3.3)

$$M(\mathbf{x}_1, \mathbf{x}_2) = \frac{2(\sqrt{\nu} w_M \|\mathbf{x}_1 - \mathbf{x}_2\|)^\nu}{\Gamma(\nu)} K_\nu(2\sqrt{\nu} w_M \|\mathbf{x}_1 - \mathbf{x}_2\|). \tag{3.12}$$

Here, ν is a user-defined smoothness parameter, $\Gamma(\nu)$ is the Gamma function $\Gamma(\nu) = \int_0^\infty e^{-t} t^{\nu-1} dt$ and $K_\nu(r)$ is the modified Bessel function of the second kind with degree ν: $K_\nu(r) = \left(\frac{r}{2}\right)^\nu \sum_{k=0}^\infty \frac{(\frac{r^2}{4})^k}{k!\Gamma(\nu+k+1)}$ [75]. It can be shown that when $\nu \to \infty$, the



(a) Layout of the experimental test-bed.

(b) Signal distribution from AP 1.

(c) Signal distribution from AP 2.

(d) Signal distribution at a fixed location.

Figure 3.2: Experimental test-bed and radio propagation characteristics.

Matérn kernel degenerates to the Gaussian kernel. On the other hand, when $\nu = 0.5$, it degenerates to the exponential kernel

$$E(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\sqrt{2} w_M \|\mathbf{x}_1 - \mathbf{x}_2\|).$$

Note that both the Gaussian and Matérn kernels are isotropic[3], and so are invariant to the location of origin and to arbitrary rotation.

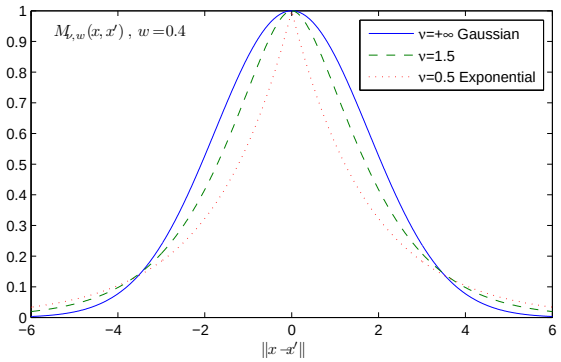


Figure 3.3: The Matérn kernel.

3.2.5 Time Complexity Analysis

In location estimation, the training phase can be performed offline and so its speed is not very important. On the other hand, the localization phase has to be performed online. In this Section, we compare the time complexities for online localization as required by LE-KCCA and other popular location estimation algorithms, namely, support vector machine (SVM) [14], support vector regression (SVR) [14], model trees [52], maximum likelihood estimation (MLE) [102] and RADAR [5]. These algorithms will also be compared experimentally in Section 3.3.

In the following, let A be the number of access points, L the number of locations, V the number of support vectors, S the number of training samples, and T the number of canonical vectors.

1. LE-KCCA: Localization involves three steps (Section 3.2.3). The first step projects $\tilde{\mathbf{s}}$ onto the T canonical vectors, each being a linear combination of the

[3]A kernel is isotropic if $k(\mathbf{x}_i, \mathbf{x}_j)$ depends only on the distance $\|\mathbf{x}_i - \mathbf{x}_j\|^2$.

S training samples in the feature space[4]. The time for one kernel evaluation is $O(A)$. Thus, this step takes $O(AST)$ time. In the second step, we compute the weighted Euclidean distance, and this takes $O(ST)$ time. These distances are then ranked, but as ranking mainly involves comparison, its time is small compared to those of others and so will be dropped. The total time complexity is thus $O(AST + ST)$.

2. SVM: Our SVM treats location estimation as a multiple-class classification problem, by using the signal strengths from the A access points as input and the L locations as output. There are $O(V)$ support vectors at each location, and so the time complexity for localization (testing) is $O(AVL)$.

3. SVR: Again using the signal strengths from the A access points as input, our SVR builds a regression model for each output dimension, x and y. As there are only two outputs, the time complexity for localization is $O(AV)$.

4. Model tree: We build two model trees, one for each output dimension. The time required is related to the height of the tree and the regression computation at the leaf nodes.

5. RADAR and MLE methods: We have to calculate the distance or probability of the new incoming signal to each location, which can be done in $O(A)$ time. As there are L candidate locations, the time complexity for both methods are $O(AL)$.

3.3 Experiments

In this Section, we perform a series of WLAN location estimation experiments in a realistic environment shown in Figure 3.2(a), the office area of Department of Computer Science, the Hong Kong University of Science and Technology. Its area is about 64m by 40m, with three entrances and four hallways. It is equipped with an IEEE 802.11b wireless network in the 2.4GHz frequency bandwidth. All data are collected with a IBM laptop computer with an external Linksys Wireless-B USB network adapter. We

[4]In fact, as the $\boldsymbol{\alpha}$ vector is typically sparse, not all training samples will be involved in the computation of the canonical vectors. By eliminating the corresponding kernel evaluations, the first step can be performed much faster.

divide the four hallways in Figure 3.2(a) into a total of 99 grids, with each grid measuring 1.5m $\times$ 1.5m. There are three access points shown in the figure, though a total of eight access points (including some from other floors) are detected. Each sample is thus an 8-dimensional signal strength vector, with the measurements averaged in one second. 100 such samples are collected at the center of each grid, with a total of 9,900 samples obtained. We randomly use 65% of the 9,900 samples for *training* and the rest for *testing*. All data are not normalized and no other preprocessing is performed. For comparison with LE-KCCA, we also run:

1. Support vector machine (SVM) [14];
2. Support vector regression (SVR) [14];
3. Model tree [52];
4. Maximum likelihood estimation (MLE) [102]; and
5. RADAR [5].

As in LE-KCCA, both SVM and SVR use the Gaussian kernel. All implementations are in C++, and experiments are performed on a 533MHz Celeron-II machine. To reduce statistical variability, results here are based on averages over 10 repetitions.

3.3.1 Setting the LE-KCCA Parameters

To determine the tunable parameters (w_M, ν and w_G) in the various methods, we further split the whole *training* data set into two parts: 75% for tentatively building the model, while the remaining 25% is used as a *validation* set for evaluating the performance. We enumerate a list of values for different parameters and empirically pick up those values, with which the model performs well in the *validation* set. Once parameters are determined, we re-combine the two parts of data (the whole training set) to build the model and evaluate the performance in the *testing* set. In this Section, we discuss the selection of the LE-KCCA parameters in more detail. There are five parameters in LE-KCCA:

- w_G in the Gaussian kernel (3.11),
- w_M and ν in the Matérn kernel (3.12),

- the regularization parameter κ in (3.6), and
- the number of neighbors K in step 3 of the online localization phase (Section 3.2.3).

Initially, we set K to 7, w_M and w_G to some random number around 0.05, and κ to some random number around 0.1. We then tune the parameters in the order of ν, K, w_M, w_G and κ (Figure 3.4).

As can be seen from Figures 3.4(a)-(b), the highest accuracy is obtained at $\nu = 0.5$. In this case the Matérn kernel degenerates to the simple exponential kernel, which is more computationally efficient. Note that in comparison to the Gaussian kernel (which corresponds to setting $\nu \to \infty$ in the Matérn kernel), the exponential kernel drops off more rapidly than the Gaussian kernel at small values of $\|\mathbf{x}_1 - \mathbf{x}_2\|$ (Figure 3.3). Hence, as the physical location measurements are relatively clean, the exponential kernel is more sensitive than the Gaussian kernel to small changes in the physical locations.

In the online localization phase, the most important parameter is K. Figures 3.4(c)-(d) show that when K increases from 1 to 7, the accuracy improves quickly; when K is from 7 to about 21, the accuracy stays about the same; then the accuracy gradually decreases when K is further increased. In the following, we set $K = 15$ (an odd number, so that the median in Step 3 of the online localization phase is always well-defined). It is interesting to compare this with RADAR [5], which also uses nearest neighbor heuristics. Unlike LE-KCCA, RADAR benefits little in varying the value of K. As discussed in Section 4.1.2 of [5], a small value of K shows minor improvement in the accuracy, while a large value leads to rapid performance degradation. This is because the RADAR neighbors in the signal space may *not* be neighbors in the physical location space. On the other hand, LE-KCCA uses location information as feedback to guide the feature extraction process so that projections of the signal and physical location spaces are maximally correlated. Consequently, neighbors in the signal projected space are usually neighbors in the physical location projected space. More discussions will be presented in Section 3.3.3 and Figure 3.7.

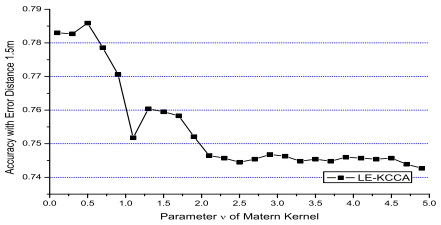
The w_M parameter reflects sensitivity of the Matérn kernel to changes in the physical location $\|\mathbf{x}_i - \mathbf{x}_j\|$. When fixing ν, the Matérn kernel M (3.12) is a function of the term $w_M\|x_i - \mathbf{x}_j\|$ with domain $[0, +\infty)$ and range $(0, 1]$. Intuitively, the kernel M here could be viewed as a distance similarity measurement (and yet a valid kernel,

positive-definite) between two location x_i and x_j where $M = 1$ means that they are in the same location (most similar) and $M \to 0$ indicates that they are far away from each other (most dissimilar). From Figure 3.3, we could also see that M drops faster with a smaller $\|x_i - x_j\|$ than with a larger value. This implies that M is likely to be sensitive to "small distance change". Here, what is "small distance change" is controlled by the scaling parameter w_M. For example, $w_M = 0$ is an extreme that $M \equiv 1$ becomes a constant so that any two points are equally similar. $w_M \to +\infty$ is another extreme that M drops sharply so that neighbor points would be dissimilar. Both the two extremes could not well capture the distance information in the physical location space. Instead, we should choose a balanced w_M value. By observing Figures 3.4(e)-(f), we empirically set $w_M = 0.05$. The role of w_G in the signal space is similar. By observing Figures 3.4(g)-(h), we set $w_G = 0.15$.

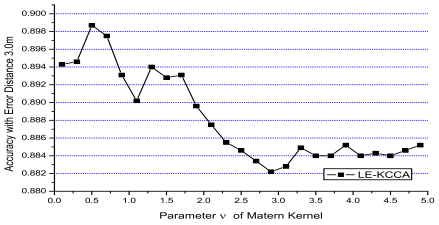
The regularization parameter κ is used to control the flexibility of the canonical vectors. As discussed in Section 3.2.2, one can always obtain perfect correlation and thus uninteresting results when $\kappa = 0$. On the other hand, if κ is too large, KCCA will be over-penalized and cannot capture the correlation between signals and physical locations. By observing Figures 3.4(i)-(j), we choose $\kappa = 1.5$.

Note that w_G and w_M are essentially scaling factors that apply to data in signal and location spaces respectively. Thus, when the basic unit is changed in either space, the corresponding parameter may be changed as well. For example, if we use *foot* rather than *meter* as the basic unit in location space, w_M should be re-scaled by 0.30 since $1ft \approx 0.30m$.

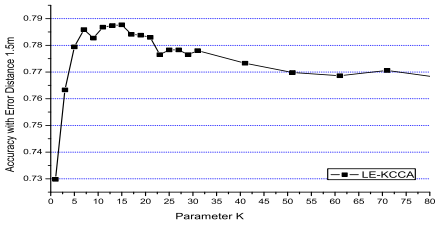
Furthermore, w_G may be affected by the hardware difference of network adapters from different manufacturers. Although this chapter does not conduct research on the effect of hardware devices, we note that [35] shows that there is approximately a linear relationship between adaptors from different hardware vendors. Thus, an alternative is to use a linear transformation as a preprocessing step, which parameters are obtained by a small number of calibration data, for adapting to new hardware. In such case, we don't need to change w_G. Once w_G or the linear transformation is properly set, the difference caused by new hardware would be reduced. Consequently, new data would tend to have the same characteristics as the old so that we don't need to change ν, K and κ.



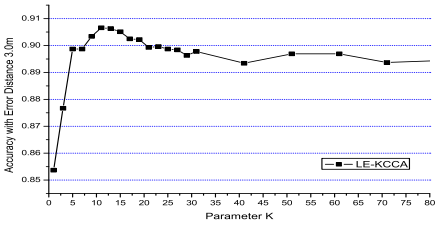
(a) Different ν's (error distance = 1.5m).



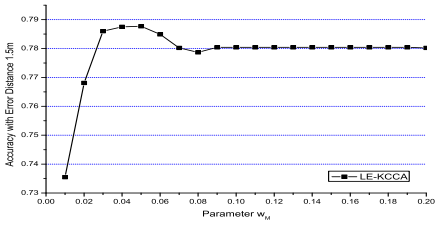
(b) Different ν's (error distance = 3.0m).



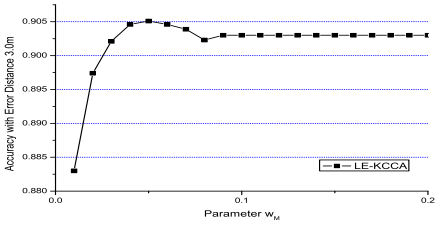
(c) Different K's (error distance = 1.5m).



(d) Different K's (error distance = 3.0m).

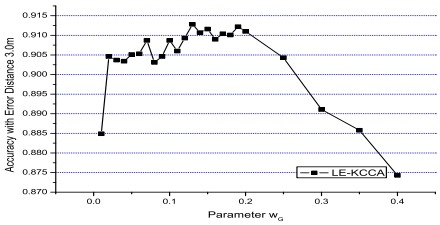


(e) Different w_M's (error distance = 1.5m).

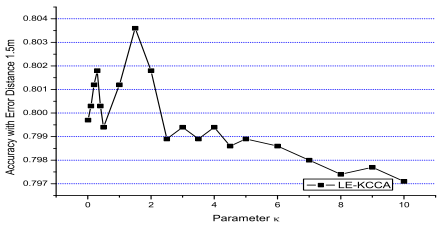


(f) Different w_M's (error distance = 3.0m).

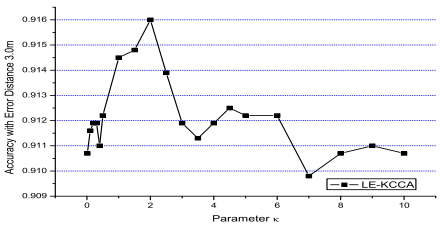
(g) Different w_G's (error distance = 1.5m).



(h) Different w_G's (error distance = 3.0m).



(i) Different κ's (error distance = 1.5m).

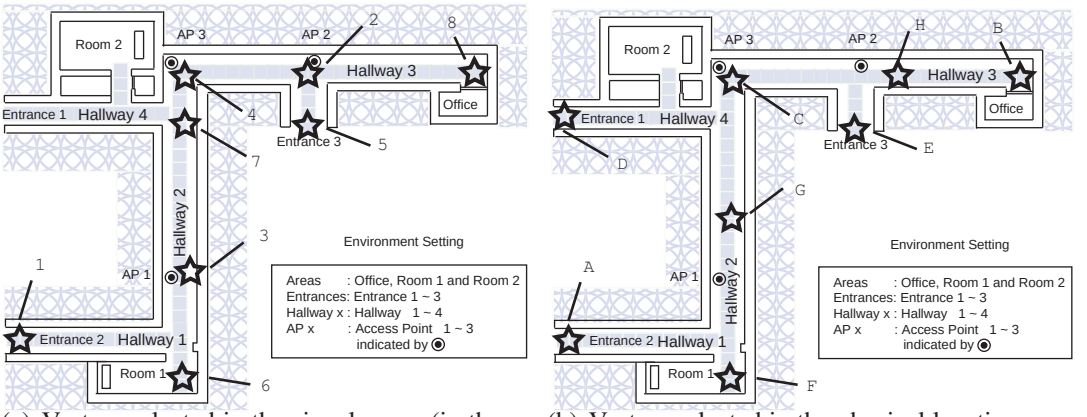


(j) Different κ's (error distance = 3.0m).

Figure 3.4: Validation set accuracies at different values of ν, K, w_M, w_G and κ's.

3.3.2 Basis Vectors Selected by PGSO

As discussed in Section 3.2.2, partial Gram-Schmidt orthogonalization (PGSO) can be used in place of the complete Cholesky decomposition to reduce the computational requirement of KCCA on large data sets. In our experiments, PGSO is applied to both the signal space and physical location space. A total of 200-300 vectors and 20-30 vectors are picked in the signal space and physical location space respectively. As vectors are sequentially added by greedily minimizing the approximation error, they tend to spread out in the kernel-induced feature space. Figure 3.5(a) shows the positions corresponding to the first eight vectors[5] selected in the signal space, while Figure 3.5(b) shows those corresponding to the first eight vectors selected in the physical location space. As can be seen, they correspond to the access points in the signal space; while, in the physical location space, they correspond to the ends, corners and centers of the hallways.



(a) Vectors selected in the signal space (in the order $1, 2, \ldots, 8$).

(b) Vectors selected in the physical location space (in the order A,B,. . .,H).

Figure 3.5: Corresponding locations of the first eight vectors selected by PGSO in the signal space and the physical location space respectively.

3.3.3 Accuracy and Speed

Figure 3.6 plots the average testing accuracies and the corresponding standard deviations at different acceptable error distances. In particular, at an error distance of 3.0m, the accuracy of LE-KCCA is 91.6% while those of SVM, model tree, SVR, MLE and RADAR are 87.8%, 88.3%, 89.1%, 86.1% and 78.8% respectively. When the accept-

[5] As all the candidate basis vectors are normalized to have unit norm, the first vector (in both the signal space and physical location space) has to be manually selected by the user.

able error distance is 1.5m, the accuracy of LE-KCCA, SVM, model tree, SVR, MLE, and RADAR are 81.7%, 77.7%, 73.7%, 76.7%, 75.8%, 47.3%. Thus, by utilizing the pairwise distance similarities in physical locations, LE-KCCA can perform better than the other methods. The SVM, by treating each (x, y) location as a class, also considers x and y together. However, it cannot utilize the information that neighboring locations should have similar signal strengths, and thus it is not as accurate as LE-KCCA. Note that SVR and the model tree, which treat the physical location dimensions x and y separately, can also perform relatively well at an error distance of 3.0m. However, at an error distance of 1.5m, the model tree has much degraded performance because its locally linear heuristics are not capable of capturing the nonlinearity of the signal-location mapping. Finally, RADAR does not perform well over the whole range. It is a deterministic method and cannot well adapt to the noisy characteristics of the signal.

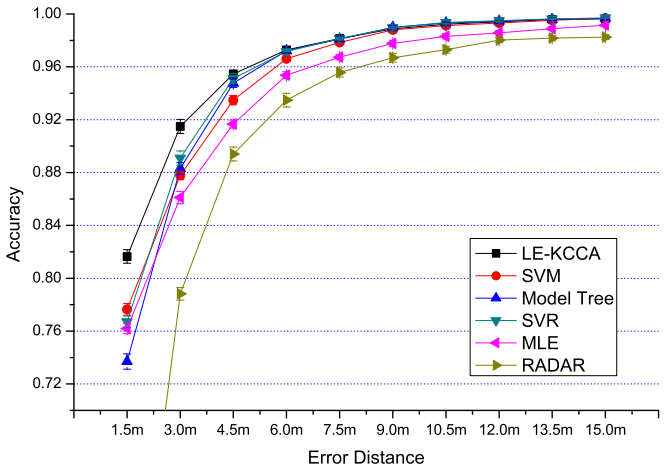


Figure 3.6: Testing accuracies at different error distances.

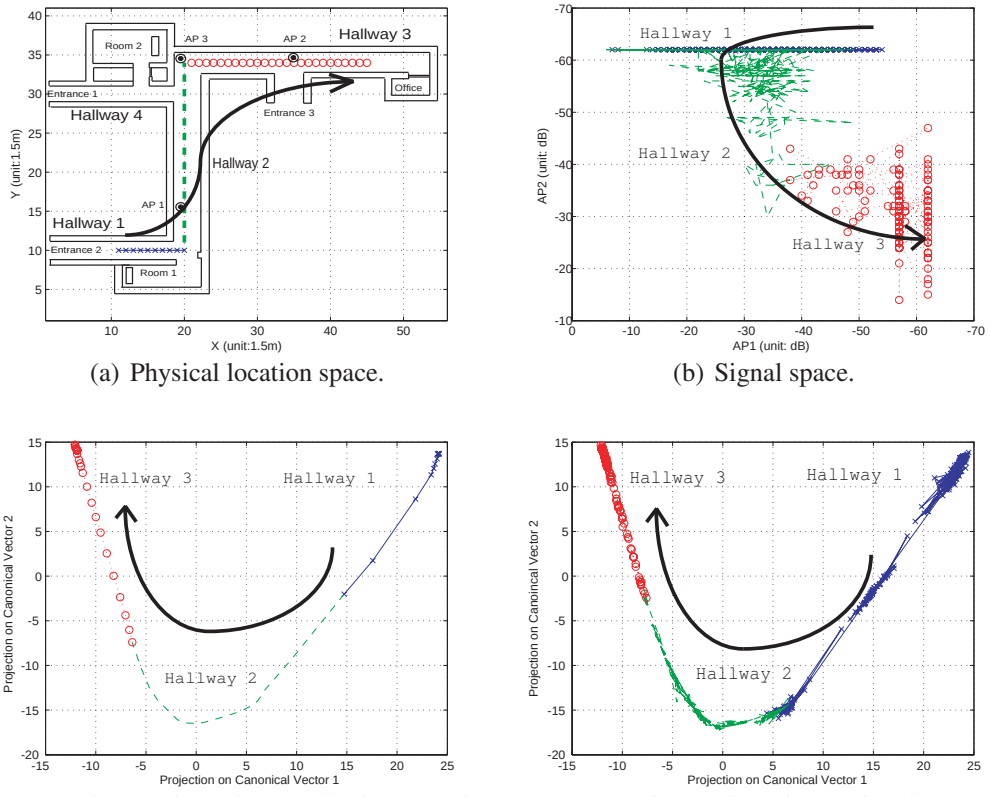
Table 3.1 compares the average CPU time for predicting a new position using the various methods. As can be seen, LE-KCCA is the slowest. Nevertheless, as this only takes 0.025 seconds, the online localization phase can still be performed in real-time.

Table 3.1: Average CPU time (in seconds) for online localization of one new position.

method	LE-KCCA	SVM	SVR	model tree	MLE	RADAR
time	0.025	0.012	0.0084	0.00027	0.00030	0.00031

With the physical location similarity as feedback, samples from the same locations move closer together, while those from different locations are pushed away under the

feature-space mapping built by LE-KCCA. This is further demonstrated in Figure 3.7. When we walk from hallway 1 through hallway 2 to hallway 3 (Figure 3.7(a)), we first come closer to AP1 and then leave AP1. We then come closer to AP2 and then leave AP2. Thus, the signal strengths of AP1 and AP2 first reach their maximum and then weaken one after another (Figure 3.7(b)). We can also see that the signal is very noisy and unstable so that neighbors in the signal space may not be neighbors in the physical location space [5]. However, after the feature mapping with KCCA, the projections in both feature spaces are maximally correlated and the trajectories become very similar to each other (Figures 3.7(c)-(d)). Consequently, nearby physical locations have similar values in the projected signal space. In this way, even though there are not enough samples at each individual physical location, we can "borrow strength" from the nearby locations. When a new signal arrives, its nearest neighbors will be closer to the true location after the KCCA mapping than those in the original space [5], and consequently we can have better location estimation.



(a) Physical location space.

(b) Signal space.

(c) Canonical variates in the physical location space.

(d) Canonical variates in the signal space.

Figure 3.7: Trajectories in the different spaces as one walks from hallway 1 through hallway 2 to hallway 3. Note that different sections of the trajectories are color-coded.

3.3.4 Different Variants of LE-KCCA

As mentioned earlier, we believe that the success of LE-KCCA stems from considering information from both physical dimensions x and y together. To validate this claim, we consider the following three variants of LE-KCCA:

1. Discrete-KCCA: It treats all the physical locations independently, which is achieved by using a large value[6] for w_M in the Matérn kernel.

2. 1D-KCCA: Here, the LE-KCCA algorithm is modified so that the signal space is correlated with x and y *separately*.

3. Linear CCA: Here, instead of using the Gaussian and Matérn kernels, we use the linear kernel in both the signal and physical location spaces.

We use all the 99 grid locations but only a random subset of the signal samples available at each location for training. Figure 3.8 shows the testing accuracies at error distances of 1.5m and 3.0m. As can be seen, all three variants lead to degraded performance as expected. In particular, linear CCA has an accuracy that is lower than 60% on every test set, and so its results are not shown in the figure.

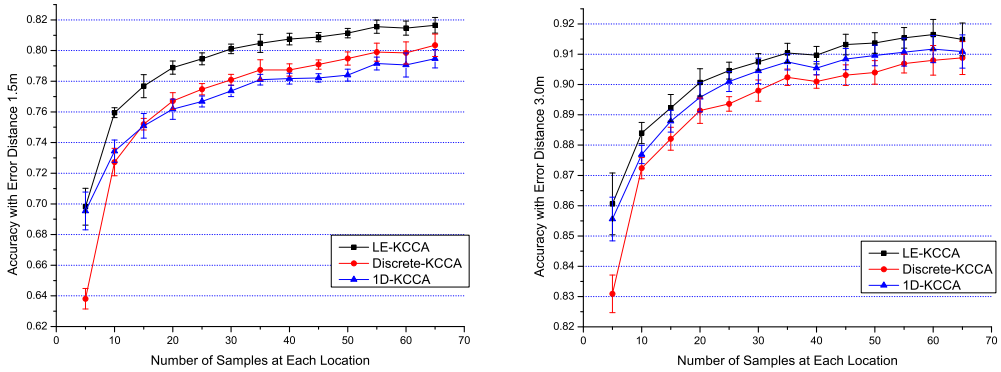


Figure 3.8: The effect when the information in both physical dimensions (x and y) are *not* considered together. Discrete-KCCA considers each grid location separately, while 1D-KCCA considers x and y independently. Linear CCA has even lower accuracy and the curve is below the bottom of the figure.

[6]In the experiment, we set $w_M = 50$.

3.4 Summary

In this chapter, we defined the machine learning based approach to the localization problem in a WSN environment. We reviewed the previous works, and presented our solutions for increasing the accuracy of localization using a multidimensional vector regression method. Experiments show a better performance than SVR and model tree that treat each output dimension separately. We found that kernel transformation and CCA allow the construction of an accurate mapping between the physical location space and the signal space.

One advantage is the higher accuracy obtained in localization with much less calibration effort. We use a Gaussian kernel for the signal space to adapt to the noisy characteristics of radio-propagation channels, and a Matérn kernel for the physical location space.

Note that the proposed method can be applied to a wider range of problems that have many classes but few examples. In this chapter, we showed how to leverage the distance-based similarity relationship between classes to enhance classification accuracy. In the future, we will extend this for other types of inter-class relationships, and develop a general framework to address this new type of learning problems.

CHAPTER 4

SEMI-SUPERVISED LOCALIZATION USING MANIFOLD

The ability to accurately detect the location of a mobile node in a sensor network is important for many artificial intelligence (AI) tasks that range from robotics to context-aware computing. Many existing approaches to the location-estimation problem assume the availability of calibrated data. For example, machine learning based localization requires a large amount of labelled data that are usually manually collected. However, to obtain such data requires great effort. In this chapter, we consider how to reduce the calibration effort of localization. We present a manifold regularization approach known as `LeMan` to calibration-effort reduction for tracking a mobile node in a wireless sensor network.

4.1 The Calibration Effort Problem

Wireless sensor networks have recently attracted great interests in AI communities. Many tasks ranging from robotics [6, 11] to context-aware computing [58, 56] can now be realized with the help of distributed wireless sensor networks. Researchers have successfully applied learning techniques in a sensor network from localization [64] to activity recognition [58]. Many of these applications depend on the ability for a sensor node to be aware of its location.

Accurately tracking *mobile* nodes in wireless sensor networks using radio-signal-strength (RSS) values is a complex and difficult task. Radio signal usually attenuates in a way that is highly nonlinear and uncertain in a complex environment, which may be further corrupted when introducing the mobility of sensor nodes. In the past, many researchers have developed ranging-based algorithms for localizing mobile nodes in a wireless sensor network. These methods [80, 76] usually consist of two main steps, by first transforming sensor reading into a distance measure and then recovering the most probable coordinates of sensor nodes. These approaches usually rely on a sophisticated signal propagation model and extensive hardware support. Learning-based approaches

[64, 5, 60] can bypass the ranging process but need relatively more calibration data. However, few of them consider the mobility of sensor nodes. Even though [63] can accurately track *mobile* nodes, it needs special hardware such as ultrasonic transceivers. Adding the required hardware increases the cost and size of sensor nodes. How to estimate the location of mobile nodes when the signal environment is noisy, and when we have much less calibrated (labelled) data and hardware support, is still an open problem.

In this chapter, we address the calibration-effort reduction problem in sensor-network based tracking by proposing a semi-supervised learning [12] approach for learning a mapping between the signal space and the physical space, which is essentially a regression problem. Our approach, called `LeMan`, is based on *manifold regularization* [8], which exploits the manifold structure of data to avoid poor generalization due to limited labelled data. We first use a mobile robot to collect a small quantity of labelled data at various locations. Then, we apply manifold regularization to solve the regression problem, in a semi-supervised manner, using a small amount of labelled data and a large amount of unlabelled data. The mapping function learned can be used online to determine the location of a mobile node in a sensor network based on the signals received.

We have tested our algorithm `LeMan` using a sensor network implemented using the Crossbow MICA2. Experimental results show that we can achieve a higher accuracy with much less calibration effort. We have found that with a much lower proportion of labelled data, `LeMan` achieves much higher accuracy than many state-of-the-art algorithms. We have also found that with an increase of the amount of unlabelled data, the accuracy can be greatly improved.

4.2 Methodology

4.2.1 Problem Statement

Consider a two-dimensional tracking problem[1]. Assume that there are N sensor nodes fixed in an area $\mathcal{C} \subseteq \mathbb{R}^2$ that we are interested in. These *beacon* nodes periodically send out beacon signals to other nodes. The locations of *beacon* nodes are not necessarily known. There are one or more *mobile* nodes of unknown locations. At time t, a *mobile*

[1]Extension to the three-dimensional case is straight-forward

node can measure the RSS sent by the N *beacon* nodes by detecting their signals which gives a vector $\mathbf{s}_t = (s_{t1}, s_{t2}, \ldots, s_{tN})' \in \mathbb{R}^N$. In addition, we have collected a small amount l of labelled training data which are signal-location pairs $\{(\mathbf{s}_{t_i}, \boldsymbol{\ell}_{t_i})\}_{i=1}^{l}$ at various locations, and u unlabelled data $\{\mathbf{s}_{t_j}\}_{j=l+1}^{l+u}$.

Our objective is to determine the location $\boldsymbol{\ell}_t = (x_t, y_t)' \in \mathcal{C}$ of a *mobile* node based on the signal vector $\mathbf{s}_t$ measured at time t. Figure 4.3 shows an example of the floor in our experimental test-bed, which consists of $N = 8$ *beacon* nodes and one *mobile* node.

4.2.2 Domain Characteristics

To establish the feasibility of our manifold regularization approach, we first examine the signal distribution properties of the sensor-network environment. Figure 4.2(a) shows the distance-signal correspondence between a *mobile* node and a *beacon* node. As can be seen, the signal attenuates in a way that is highly nonlinear and noisy. The uncertainty is relatively larger at a farther distance. Even when the *mobile* node is fixed in one location, the signal would not be stable.

When there is little noise, every two-dimensional location would uniquely determine a signal vector in a high-dimensional signal space and the localization area would be mapped to a two-dimensional manifold. However, when we take into account the uncertainty introduced by environment noise and node mobility, the manifold is corrupted. However, the manifold regularization is still feasible for our problem. As shown in Figure 4.2(b), the average signal change between two time points which is measured by Euclidean norm $\|\mathbf{s}_{t_i} - \mathbf{s}_{t_j}\|$, grows with the physical distance $\|\boldsymbol{\ell}_{t_i} - \boldsymbol{\ell}_{t_j}\|$. Similarly, the signal change also grows with the time interval $|t_i - t_j|$ because of robot movement, which is shown in Figure 4.2(c). These two figures support the fact that neighboring locations have more similar signal vectors than those that are far away. This fact also holds at the dimension of time since a *mobile* node can not move too fast. Furthermore, we found that such signal change has a stable statistical pattern. More specifically, the signal change is likely to obey a Gaussian distribution at a fix physical distance or time interval even if the signal itself is not Gaussian at a fixed location or time. One example is shown in Figure 4.2(d). A more detailed hypothesis testing of the above hypothesis will be provided in our future work. Our problem basically satisfies the assumption of manifold regularization that similar signal vectors in the

intrinsic geometry lead to similar location distribution. Figure 4.1 illustrates how the unlabelled examples can supplement the sparsity of labelled ones in the sample space. A dashed line shows that two examples are similar in signal or time dimensions.

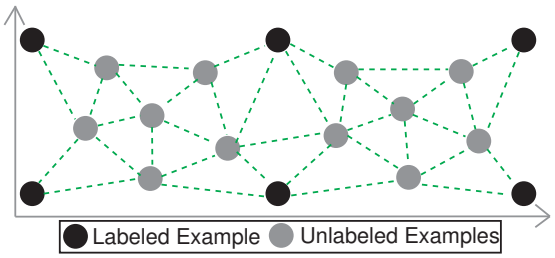
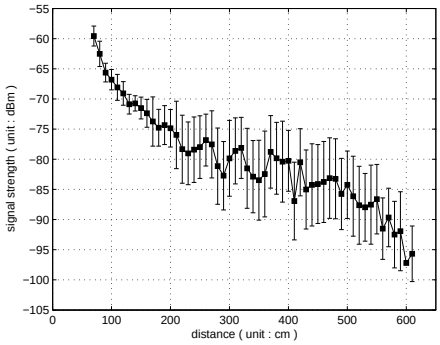
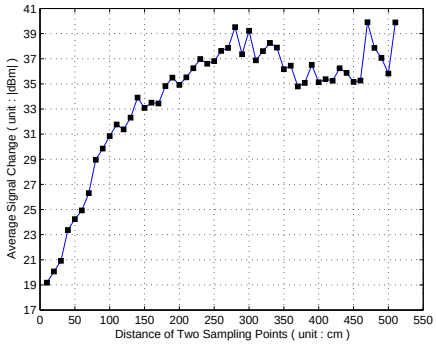


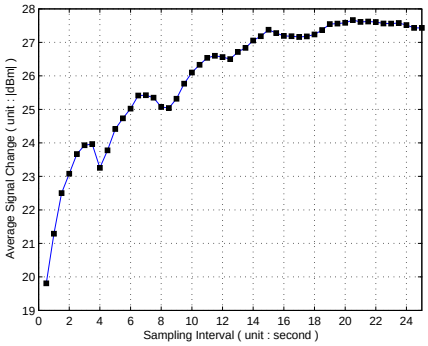
Figure 4.1: The use of labelled and unlabel examples



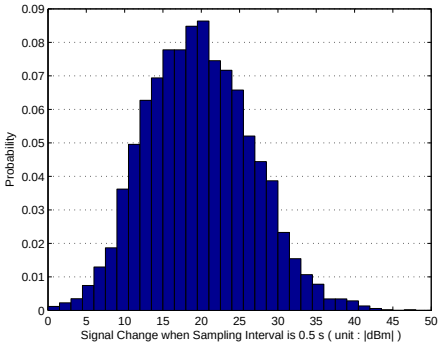
(a) Signal attenuation



(b) Signal change over space



(c) Signal change over time



(d) Distribution of signal change

Figure 4.2: Radio Characteristics

4.2.3 Manifold Regularization

We try to learn a mapping between the signal space and the physical space in sensor-network based tracking under the semi-supervised learning setting. To avoid poor generalization due to limited labelled data, the learning problem is formulated under

a regularization framework. Specifically, it is a data-dependent regularization framework called *manifold regularization* [8] that exploits the geometric structure of the marginal distribution of the data in the signal space. The basic underlying assumption of manifold regularization is that if two points are close in the intrinsic geometry of the marginal distribution, then their conditional distributions are similar. For classification problems, this implies that they are likely to have the same label. For regression problems, their regression function values are similar. Manifold regularization introduces to the regularized risk functional an additional regularizer that serves to impose this assumption on the learning problem.

Let us now define the learning problem more formally. Suppose we have a set of l labelled examples $\{(r_i, z_i)\}_{i=1}^{l}$ and a set of u unlabelled examples $\{r_j\}_{j=l+1}^{l+u}$, where r_i and r_j are sampled from the input space $\mathcal{X}$ according to the marginal distribution $\mathcal{P}_{\mathcal{X}}$ and $z_i \in \mathbb{R}$ is governed by the conditional distribution $\mathcal{P}(z|r)$. The learning problem corresponds to solving the following optimization problem:

$$f^* = \underset{f \in \mathcal{H}_K}{\arg\min} \frac{1}{l} \sum_{i=1}^{l} V(r_i, z_i, f) + \gamma_A \|f\|_K^2 + \gamma_I \int_{\mathcal{M}} \langle \nabla_{\mathcal{M}}, \nabla_{\mathcal{M}} \rangle, \tag{4.1}$$

which finds the optimal function f^* in the reproducing kernel Hilbert space (RKHS) $\mathcal{H}_K$ of functions $f : \mathcal{X} \to \mathbb{R}$ corresponding to a Mercer kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.

The first term of the regularized risk functional is defined based on the loss function V which measures the discrepancy between the predicted value $f(r_i)$ and the actual value z_i. The second term controls the complexity of f in terms of the norm $\| \cdot \|_K$, with γ_A being the regularization parameter. The third term is specific to manifold regularization and is based on the assumption that the support of $\mathcal{P}_{\mathcal{X}}$ forms a compact submanifold $\mathcal{M}$. It controls the complexity of f in the intrinsic geometry of $\mathcal{P}_{\mathcal{X}}$, with γ_I being the corresponding regularization parameter.

In [8], the third term is approximated using the graph Laplacian [19] defined on all $l + u$ labelled and unlabelled examples without using the label information. Hence the optimization problem can be reformulated as:

$$f^* = \underset{f \in \mathcal{H}_K}{\arg\min} \frac{1}{l} \sum_{i=1}^{l} V(r_i, z_i, f) + \gamma_A \|f\|_K^2 + \frac{\gamma_I}{(u+l)^2} \hat{f}^T L \hat{f}, \tag{4.2}$$

where $\hat{f} = (f(r_1), \ldots, f(r_{l+u}))'$ and L is the graph Laplacian.

From the extended Representer Theorem [8], the optimal function can be expressed

in the following form:

$$f^*(r) = \sum_{i=1}^{l+u} \alpha_i K(r_i, r). \tag{4.3}$$

When focusing on regression, V in (4.2) is set to be the squared loss function $V(r_i, z_i, f) = (z_i - f(r_i))^2$ to give the Laplacian Regularized Least Squares (LapRLS) algorithm [8]. It can be shown that the optimal solution $\alpha^* = (\alpha_1^*, \ldots, \alpha_{l+u}^*)'$ is given by

$$\alpha^* = (JK + \gamma_A lI + \frac{\gamma_I l}{(u+l)^2} LK)^{-1} Z, \tag{4.4}$$

where K is the $(l+u) \times (l+u)$ Gram matrix over all labelled and unlabelled examples, Z is an $(l+u)$-dimensional label vector given by $Z = (z_1, \ldots, z_l, 0, \ldots, 0)'$, and $J = \text{diag}(1, \ldots, 1, 0, \ldots, 0)$ is an $(l+u) \times (l+u)$ diagonal matrix with the first l diagonal entries being 1 and the rest being 0.

4.2.4 The `LeMan` Algorithm

Our location estimation algorithm `LeMan`, which is based on manifold regularization, has two phases : an offline training phase and an online localization phase.

• Offline Training Phase

1. Collect l labelled signal-location pairs $\{(\mathbf{s}_{t_i}, \boldsymbol{\ell}_{t_i})\}_{i=1}^{l}$ at various locations and u unlabelled signal examples $\{\mathbf{s}_{t_j}\}_{j=l+1}^{l+u}$ with a mobile robot, on which top we attach a sensor node. The mobile robot runs and stops around the test-bed so that these signal vectors form a long trace.

2. Scale each component of every signal vector $\mathbf{s}$ to $[0, 1]$ where $\mathbf{s} \in S = \{(s_{t_p 1}, \ldots, s_{t_p N})\}_{p=1}^{l+u}$. This step is commonly used in kernel methods.

3. For each signal vector $\mathbf{s}_{t_p} \in S$, connect $\mathbf{s}_{t_p}$ to its k nearest neighbors, which distance is measured by Euclidean norm $\|\mathbf{s}_{t_p} - \mathbf{s}_{t_q}\|$ where $p, q \in [1, l+u]$. We further link those $\mathbf{s}_{t_p}$ and $\mathbf{s}_{t_q}$ together if $|t_p - t_q| < \Delta T$. These two kinds of connections are based on our analysis from Figure 4.2(b) and 4.2(c). After that, the adjacency graph and weight matrix can be used to form the graph Laplacian L.

4. Laplacian Regularized Least Square, with proper choice of kernel, is used to learn the mapping from signal vector $\mathbf{s}$ to location $\boldsymbol{\ell}$. In particular, the optimal α_x^* and α_y^* are obtained from Equation (4.4) for x and y coordinates.

• **Online Localization Phase**

1. At time t, the *mobile* node collects a signal vector $\tilde{\mathbf{s}}_t$. For those *beacon* nodes that are far away, which signals are too weak to detect, we fill in a small value, e.g. -95dBm.

2. Scale each component of $\tilde{\mathbf{s}}_t$ to $[0, 1]$ in a similar way as Step 2 in the training phase.

3. $\hat{\boldsymbol{\ell}}_t = (\hat{x}_t, \hat{y}_t)' = (f_{\alpha_x}^*(\tilde{\mathbf{s}}_t), f_{\alpha_y}^*(\tilde{\mathbf{s}}_t))'$ is the location estimation at time t, where α_x^* and α_y^* are obtained from Step 4 in the training phase and f^* is from Equation (4.3).

4. Optionally, applying a Bayes Filter [27, 43] would be a bonus. For example, Kalman Filter, which encodes a proper motion model, can be used to smooth the trajectory and enhance the performance of *any* localization algorithm.

4.3 Experimental Setup

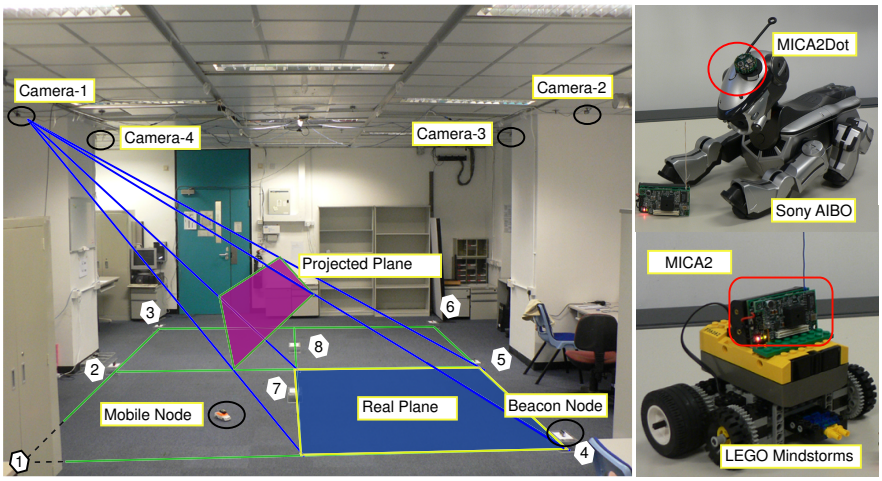


Figure 4.3: Experimental Test-Bed

Our experiment is performed in a laboratory of HKUST (Figure 4.3). The room is large enough for us to set up an experimental test-bed of 5.0 meter by 4.0 meter. In

Figure 4.3, $\|P_1P_3\| = \|P_4P_6\| = 5.0m$ and $\|P_1P_4\| = \|P_3P_6\| = 4.0m$. There are three main components of our setup:

- *Wireless Sensor Networks* can be constructed with Crossbow MICA2 or MICA2Dot. Figure 4.3 shows that there are eight *beacon* sensor nodes deployed on the floor denoted as $1, \ldots, 8$, which are set to broadcast beacon signals periodically. *Mobile* nodes are attached on top of robots.

- *Mobile Robots* are used for supporting mobility. However, in practice, sensor nodes can be attached to any object. We tried different robots that can run freely around the floor such as LEGO Mindstorms and Sony AIBO dogs.

- *A Camera Array* is used to record experiments for supporting time-dependent location information (ground truth) of mobile robots. Figure 4.3 shows that the whole test-bed is monitored by four USB cameras attached at the ceiling. They can record videos at $20fps$ with resolution 320*240 pixels and track the location of mobile nodes within error distance $2.5cm$ after calibration.

4.4 Experimental Results

In this section, experiments are performed to evaluate the location estimation accuracy, calibration effort reduction and the robustness of `LeMan`. For comparison, we also run: (1) Regularized Least Square (RLS); (2) Support Vector Regression (SVR); (3) LANDMARC [65]; (4) RADAR [5]; (5) MoteTrack [60]; (6) Basic Multilateration [80].

In applications, we can attach sensor nodes to any object. In our experiments, we control a mobile robot to run and stop around the test area (Figure 4.3) for collecting data with sampling interval $0.5s$. Two data sets are collected at different times within two days, each forming a trace of total length about $1,400\ meters$ or $7,000$ examples. We spent nearly two months setting up cameras and exporting locations from videos. Labels were expensive to obtain. For every experiment below, we randomly picked up a subset of data from one data set for training and evaluate the performance on the other. To reduce statistical variability, results here are based on averages over 20 repetitions.

We use a Gaussian kernel $exp(-\|\mathbf{s}_{t_p} - \mathbf{s}_{t_q}\|^2/2\sigma^2)$ for Equations (4.3) and (4.4) since it is widely used in localization problems for adapting the noisy characteristic of radio signal [64] and σ is set to 0.5. The number of nearest neighbors k and the time interval ΔT for constructing graph Laplacian L are set to 5 and $0.5s$ respectively. For γ_A and γ_I in Equation (4.4), we refer to the handwritten digit and spoken letter recognition experiments in [8] and set $\gamma_A l = 0.005$ and $\frac{\gamma_I l}{(u+l)^2} = 0.045$.

4.4.1 Location Estimation Accuracy

In this section, experiments are done on the whole testing set ($7,000$ examples) when using another 100 labelled examples for all compared methods and additional 500 unlabelled examples for `LeMan` from the training set. Figure 4.4(a) plots the cumulative probability with respect to error distance. More detailed results are summarized in Table 4.1. As can be seen, `LeMan` has a better performance than the others. Note that RLS is a special case of `LeMan` when no unlabelled examples are used. Furthermore, `LeMan` has the smallest mean error distance and standard deviation. In the worst case, `LeMan` may predict the location within about 290cm. `LeMan`, with the help of unlabelled examples, improves the performance of tracking *mobile* nodes.

Figure 4.4(b) illustrates an estimated trajectory of about 30 $seconds$. The trajectory is not smooth since we have not yet applied a Bayes Filter (Step 4) in the online phase. However, we test that, by employing a filter, the performance of *all* compared methods can be enhanced by about 9% to 12%. What is more, the maximal error distance can be greatly reduced because the filter smooths the trajectory. In practice, Bayes Filters are generally used. However, in this chapter, we prefer to see the original performance of all compared methods. Thus, experimental results shown in this and the following sections are done without any Bayes Filter.

Figure 4.4(c) shows the sensitivity of error distance while varying parameters $\gamma_A l$ and $\gamma_I l/(u+l)^2$ in Equation (4.4). As can be seen, the result is stable when $\gamma = \gamma_A l + \gamma_I l/(u+l)^2$ ranges in $[10^{-1.5}, 10^{-0.5}]$. When the penalty that controls the function complexity in Equation (4.1) or (4.2) is higher, say , $\gamma > 10^{-0.5}$, the function mapping from signal to location becomes simpler and tends to underfit. On the other hand, if $\gamma < 10^{-1.5}$, the error distance goes up and overfits.

We test the average time for predicting a new position using the various methods on Matlab with a $3.2GHz$ CPU just for easy control of parameters and evaluation.

Table 4.1: Performance of Different Methods

Method	Mean (cm)	Std. (cm)	Max (cm)	Accuracy at 100cm	Time (ms)
LeMan	* 67	* 39	290	* 82%	0.242
RLS	78	46	358	73%	0.047
SVR	79	40	* 257	75%	0.045
RADAR	86	59	391	68%	0.106
MoteTrack	85	61	418	69%	0.106
Multilateration	108	77	1592	53%	0.125
LANDMARC	118	59	372	42%	0.085

However, our sensor data are collected on a realistic sensor network based on Crossbow MICA2. The result is shown in Table 4.1. `LeMan` is relatively slower than the others. However, it is still suitable in real time.

4.4.2 Calibration Effort Reduction

In the first experiment, we test the performance of `LeMan` in reducing the calibration effort. We set the number of labelled examples to 50, 100, 200 and 400 while varying the number of unlabelled ones from 0 to 1000 in each setting, which results are shown in Figure 4.4(d). For example, the error distance is $75cm$ when given 50 labelled and 250 unlabelled examples. Compared to $87cm$ if no unlabelled examples are available, the performance is enhanced by $(87 - 75)/87 = 14\%$ and is better than the result with 100 labelled examples only. `LeMan`, with the use of unlabelled examples, does help reduce calibration effort when the number of labelled examples is rare.

In the second experiment,we test how the number of labelled examples affects all compared methods. We fix the total number of examples $l + u = 500$ and vary the ratio of labelled part $l/(l + u)$ from 5% to 100%. Figure 4.4(e) shows that most of the methods benefit from the increasing number of labelled examples. `LeMan` again has a better performance than the others with the help of unlabelled examples. However, if more labelled examples are available, the unlabelled examples become less important.

4.4.3 Robustness to the Number of Beacon Nodes

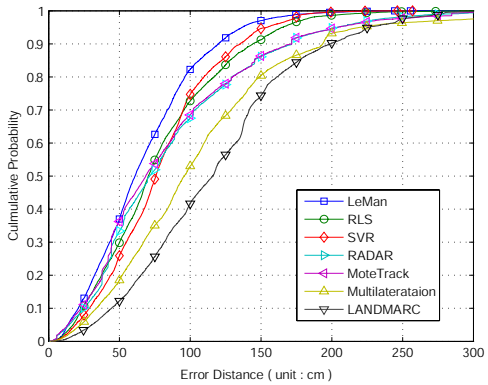
Low density of nodes due to sparse deployment and node failure may affect the performance. In this experiment, we study how the number of *beacon* nodes affect the performance. We set the number of labelled and unlabelled examples to 100 and 500 respectively and randomly select a subset of beacon nodes for the experiment. The

result is shown in Figure 4.4(f). As can be seen, LeMan is relatively more robust than the others.

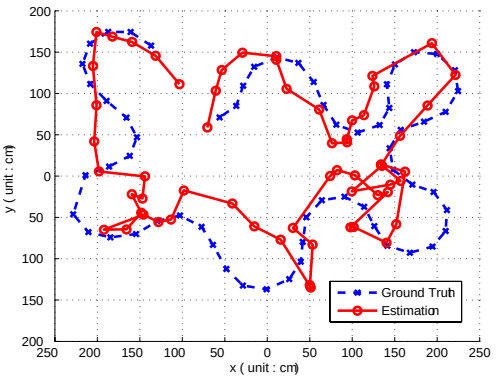
4.5 Summary

In this chapter, we described the problem of how to reduce the calibration effort in learning a localization model. We reviewed previous works, and presented our solution LeMan, a manifold regularization approach to solving this problem. Our model is based on the observation that similar signals from *beacon* nodes imply close locations. A mapping function between the signal space and the physical space is learned by using a small amount of labelled data and a large amount of unlabelled data. This function can then be used online to determine the location of *mobile* nodes. Experimental results show that we can achieve a higher accuracy with much less calibration effort. It is robust to changes in the number of *beacon* nodes, too. Furthermore, tracking multiple *mobile* nodes would not burden the network since each *mobile* node can estimate its own location by passively listening to *beacon* signals.

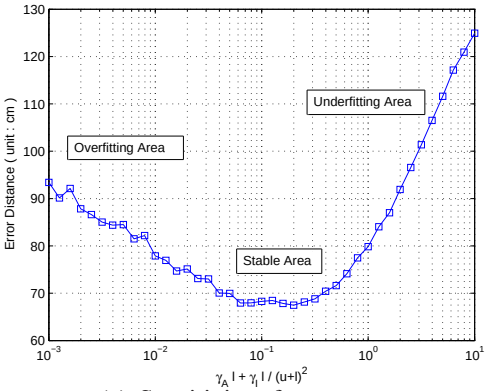
We are encouraged by the results and plan to extend this work in a few directions. First, we would like to move the experiment from the lab to a more complex environment to check the performance and robustness. Secondly, we may consider "distributed storage" of calibration data. Each *beacon* node stores some local calibration data nearby and broadcasts these data in their *beacon* frames so that we can reduce the storage cost in each *mobile* node. Third, if *beacon* nodes are densely deployed and their locations can be determined with any available hardware or algorithm [83, 64], we can use their signal reading and location as the labelled examples. By combining them with unlabelled examples from *mobile* nodes, our algorithm can be fully automatic.



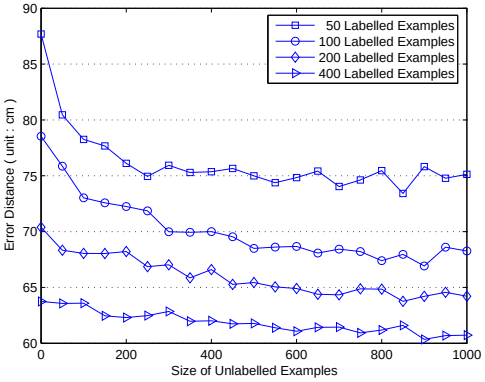
(a) Comparison of accuracy



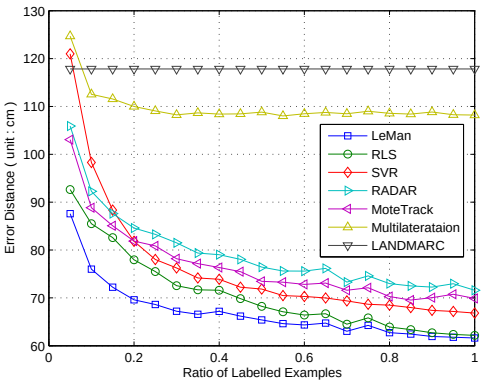
(b) An estimated trajectory of LeMan



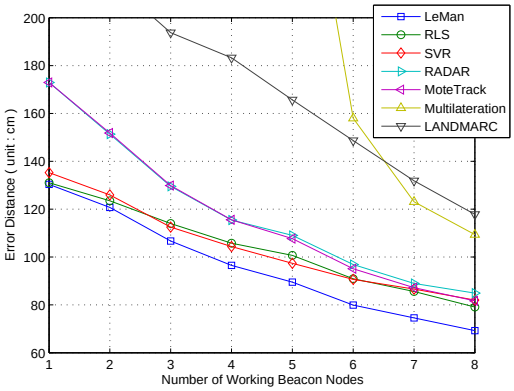
(c) Sensitivity of parameters



(d) Vary the number of unlabelled examples



(e) Vary the ratio of labelled examples



(f) Vary the number of *beacon* nodes

Figure 4.4: Experimental Results

CHAPTER 5

SEMI-SUPERVISED CO-LOCALIZATION BY DIMENSION REDUCTION

This chapter addresses a second aspect of the problem of reducing the calibration effort, in that we now have more information about the wireless environment. Suppose that we know some of the locations of both mobile devices and access points. We call the problem of recovering the remaining locations and unlabelled access points the *co-localization* problem. Our approach exploits both *labelled* and *unlabelled* data from mobile devices and access points, and combines dimensionality reduction in information retrieval with manifold learning. Extensive experiments are conducted in wireless local-area networks, wireless sensor networks and radio frequency identification networks.

5.1 The Co-Localization Problem

Accurately tracking *mobile* devices in wireless networks using received-signal-strength (RSS) values is a useful task in robotics and activity recognition. It is also a difficult task since radio signals usually attenuate in a highly nonlinear and uncertain way in a complex environment where client devices may be moving. Existing approaches to RSS localization fall into two main categories [26]: (1) radio propagation models [62, 80], which rely on the knowledge of access point locations; (2) statistical machine learning models [64, 57, 5], which require a large amount of costly calibration.

However, in cities and large buildings where wireless networks are set up by different network suppliers, it is not easy to ask them to share the location information of all access points for business or privacy reasons. Besides, a mobile device may also want to locate access points for obtaining stable connections or to spot them in hostile areas. In all these cases, sufficient calibration (*labelled*) data on mobile devices and access points may not always be available due to the lack of GPS coverage or costly human effort.

In this chapter, we address the problem of simultaneously recovering the locations of both mobile devices and access points, a problem which we call *co-localization*, using *labelled* and *unlabelled* RSS data from both mobile devices and access points. We take two steps for solving this problem. In the first step, we assume that only *unlabelled* RSS data are given. In such case, we show that the problem can be solved by Latent Semantic Indexing (LSI) or Singular Value Decomposition (SVD) [22], techniques that are popular in information retrieval. Consequently, the relative locations of APs and mobile device trajectory can be determined. In the second step, we assume that a small amount of *labelled* RSS data from mobile devices and access points are given. To determine the absolute locations of the devices and access points, we apply a semi-supervised algorithm with graph Laplacian and manifold learning [19, 7, 36]. Finally, we provide a unified framework for both the above unsupervised and semi-supervised solutions.

We tested our *co-localization* algorithms in different indoor environments using both static and mobile client devices. We also tested the algorithms with different hardware such as 802.11 Wireless Local Area Networks (WLAN), Wireless Sensor Networks (WSN) and Radio Frequency Identifiers (RFID). Experimental results showed that we can achieve a higher accuracy with much less calibration effort in different environments, motion patterns and with different hardware.

5.2 Methodology

5.2.1 Problem Statement

Consider a two-dimensional *co-localization* problem. Assume that a user holds a mobile device and navigates in an indoor wireless environment $\mathcal{C} \subseteq \mathbb{R}^2$ of n access points, which can periodically send out beacon signals. At some time t_i, the RSS values from all the n access points are measured by the mobile device to form a row vector $\mathbf{s}_i = [s_{i1}\ s_{i2}\ \ldots\ s_{in}] \in \mathbb{R}^n$. A sequence of m signal strength vectors form an $m \times n$ matrix $S = [\mathbf{s}'_1\ \mathbf{s}'_2\ \ldots\ \mathbf{s}'_m]'$, where "prime" is used to denote matrix transposition. Here, the locations of some access points and the mobile devices at some time t are known or *labelled*, while the rest are *unlabelled*.

Our objectives are stated as follows: We wish to estimate the $m \times 2$ location matrix $P = [\mathbf{p}'_1, \mathbf{p}'_2, \ldots, \mathbf{p}'_m]'$ where $\mathbf{p}_i = [p_{i1}\ p_{i2}] \in \mathcal{C}$ is the location of the mobile device at

Table 5.1: Signal Strength (unit:dBm)

	AP_1	AP_2	AP_3	AP_4	AP_5
t_A	-40		-60	-40	-70
t_B	-50	-60		-80	
t_C		-40	-70		
t_D	-80		**-40**	-70	
t_E	-40		-70	-40	-80
t_F	-80			-80	-50

(All values are rounded for illustration)

time t_i and the $n \times 2$ location matrix $Q = [\mathbf{q}'_1, \mathbf{q}'_2, \ldots, \mathbf{q}'_n]'$ where $\mathbf{q}_j = [q_{j1}\ q_{j2}] \in \mathcal{C}$ is the location of the j access points. Our objectives are to determine the locations of all of the remaining access points and the trajectory of the mobile device. We call this problem *co-localization*.

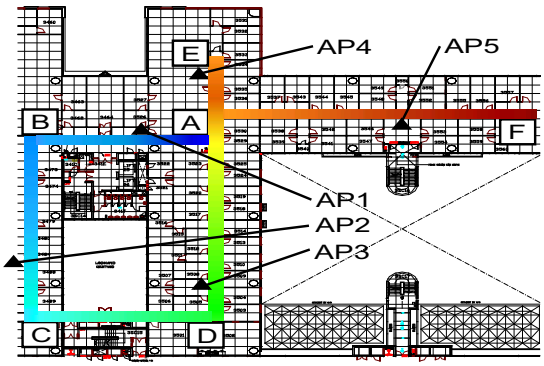
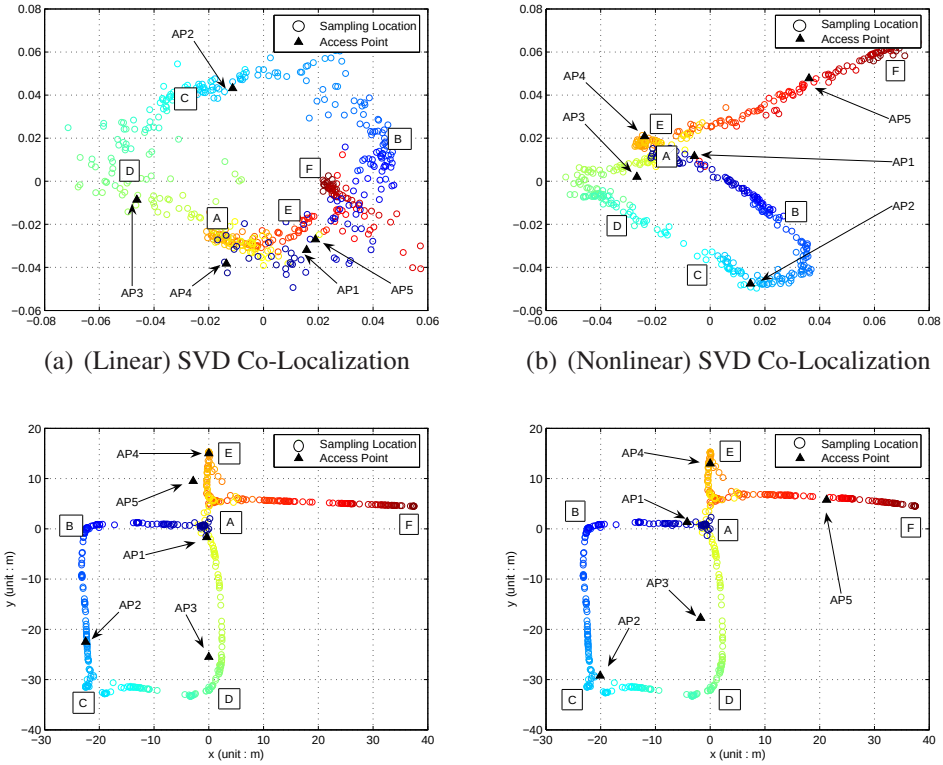


Figure 5.1: WLAN Test-bed

Example 1 As an example, Figure 5.1 shows an indoor 802.11 wireless LAN environment of size about $60m \times 50m$. It is equipped with $n = 5$ access points. A user with an IBM T42 notebook that is equipped with an Intel Pro/2200BG internal wireless card walks from A through $B, \ldots, E$ to F at time $t_A, t_B, \ldots, t_F$. $m = 6$ signal strength vectors are extracted and the 6×5 matrix S is shown in Table 5.1. By walking from A to $B, \ldots, E$ and finally to F in the hallways, we collected 500 signal strength vectors from 5 access points. Note that the blank cells denote the missing values, which we can fill in a small default value, e.g., $-100dBm$.

Our task is to estimate the trajectory matrix P of the mobile device at all times and to determine the location matrix Q of the access points $AP_1, AP_2, \ldots, AP_5$.



(a) (Linear) SVD Co-Localization

(b) (Nonlinear) SVD Co-Localization

(c) Manifold-based Co-Localization

(d) The Unifying Framework

Figure 5.2: 802.11 Wireless LAN test in an indoor environment

5.2.2 SVD-based Relative Co-Localization

Given *unlabelled* data only, we can determine the relative locations of the mobile device and the access points. This problem is called relative *co-localization*. Intuitively, we may observe the following characteristics of the data (see Table 5.1):

1. Considering two *rows* of the data, the mobile device at two different time may spatially close to each other if their signal strengths are similar when received from most access points, e.g., the time t_A and t_E.

2. Considering two *columns* of the data, two access points may be spatially close to each other if the signal strengths to the mobile device be similar most of the time, e.g., AP_1 and AP_4.

3. Considering a *single cell* s_{ij} of the data, the mobile device and the j access point may spatially close to each other at time t_i if the signal be strong, e.g., the mobile device is close to AP_3 at time t_D.

Note that the access points and the mobile device are treated in a completely symmetric manner here. (1) The location of the mobile device is represented in terms of the signal strengths to different access points. (2) The location of an access point is expressed in terms of the signal strengths to the mobile device estimated in different time. (3) The distance between the mobile device and an access point is one hidden factor that determines the signal strength.

The above observations enabled us to relate co-localization with information retrieval. Not surprisingly, the *co-localization* is closely related to the Latent Semantic Indexing (LSI) [22]. In this view, we treat an access point as a term and a mobile device at some time as a document. The above three observed characteristics would be mapped to the similarities of document-document, term-term and document-term respectively. Estimating the positions of the mobile device and the access points corresponds to discovering the latent semantics of documents and terms in some concept space.

More specifically, we can estimate the relative coordinates by performing Singular Value Decomposition (SVD).

1. Transform the signal matrix $S = [s_{ij}]_{m \times n}$ to a non-negative weight matrix $A = [a_{ij}]_{m \times n}$ by a *linear* function $a_{ij} = s_{ij} - s^{min}$ where s^{min} is the minimal signal strength detected, e.g., the noise level or $-100dBm$.

2. Normalize the weight matrix by $A_{\mathcal{N}} = D_1^{-1/2} A D_2^{-1/2}$. Here, D_1 and D_2 are both diagonal matrices such that $D_1 = diag(d_1^1, d_2^1, \ldots, d_m^1)$ where $d_i^1 = \sum_{j=1}^{n} a_{ij}$ and $D_2 = diag(d_1^2, d_2^2, \ldots, d_n^2)$ where $d_j^2 = \sum_{i=1}^{m} a_{ij}$.

3. Perform SVD on the normalized weight matrix by $A_{\mathcal{N}} \approx U_{m \times r} \Sigma_{r \times r} V'_{n \times r}$. The columns of $U_{m \times r} = [\mathbf{u}_1 \ldots \mathbf{u}_r]$ and $V_{n \times r} = [\mathbf{v}_1 \ldots \mathbf{v}_r]$ are the left and right singular vectors. The singular values of the diagonal matrix $\Sigma_{r \times r} = diag(\sigma_1, \sigma_2, \ldots, \sigma_r)$ are ranked in *non-increasing* order.

4. The (latent) location matrices of the mobile device P and that of the access points Q can be estimated using $P = D_1^{-1/2}[\mathbf{u}_2\ \mathbf{u}_3]$ and $Q = D_2^{-1/2}[\mathbf{v}_2\ \mathbf{v}_3]$. Note that we skip the first singular vectors $\mathbf{u}_1$ and $\mathbf{v}_1$ which mostly capture some constant since matrix $A_{\mathcal{N}}$ is not centering.

As an example, after performing SVD on data in **Example 1**, we obtained the latent coordinates of the mobile device and the access points, which are shown in Figure 5.2(a). In this example, it is easy to see that the hallway structure is not well preserved by comparing the true location sequence shown in Figure 5.1. This is because SVD assumes a *linear* subspace, while the correlation of RSS values and distance to APs is often nonlinear [64].

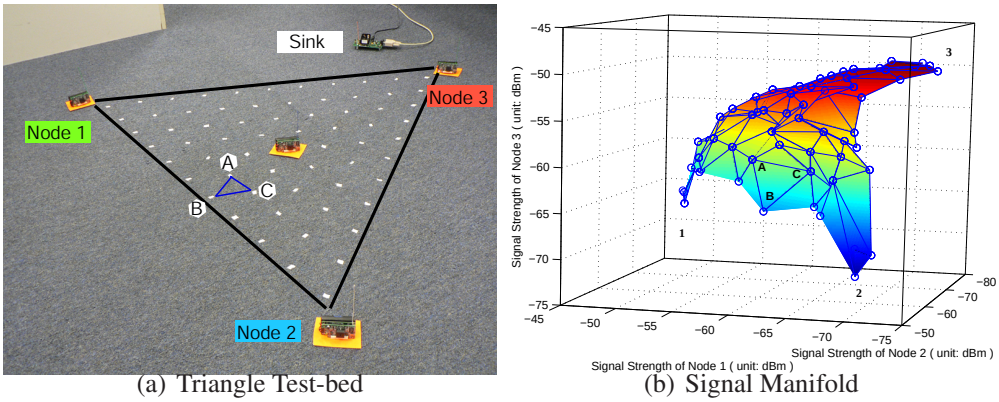
A better solution is using Kernel SVD [84] or simply transforming signal strengths to weights by some *nonlinear* function. More specifically, we transform the signal matrix $S = [s_{ij}]_{m \times n}$ to a new weight matrix $A = [a_{ij}]_{m \times n}$ by a Gaussian kernel $a_{ij} = exp(-|s_{ij} - s^{max}|^2/2\sigma_A^2)$ where s^{max} is the maximal signal strength detected, e.g., the signal strength around an access point or $-30dBm$. Figure 5.2(b) plots the *co-localization* result using P and Q. Intuitively, the reconstructed hallway structure and the locations of access points are better than that shown in Figure 5.2(a) while referring to the ground truth illustrated in Figure 5.1.

5.2.3 Manifold-based Absolute Co-Localization

When the physical locations of some access points and the mobile device at some time are known, we can ground the unknown coordinates by exploiting the geometry of the signal distribution. More specifically, we can use manifold-based learning, which generally assumes that if two points are close in the intrinsic geometry of the marginal distribution, their conditional distributions are similar [8, 86, 36]. This implies that the mobile device shall be spatially close to each other if their signal vectors are similar along some manifold structure [72, 70]. For example, the mobile device at time t_A and t_E shall be spatially close to each other (Figure 5.1) since their signal strengths are similar (Table 5.1).

A more concrete example is shown in Figure 5.3. As can be seen in Figure 5.3(a), there is a two-dimensional triangle localization area with three beacon nodes placed at the vertices. The corresponding signal strengths forms a two-dimensional nonlinear signal manifold in a three-dimensional space in Figure 5.3(b). Point A, B and C are neighbors in both location and signal spaces.

When the manifold assumption holds, the optimal solution is give by $\mathbf{f}^* = \arg\min \Sigma_{i=1}^{l} |f_i - y_i|^2 + \gamma \mathbf{f}^T L \mathbf{f}$ [36] where the first term measures the fitting error and the second term poses the smoothness along the manifold and L is the graph Laplacian [19]. For our



(a) Triangle Test-bed (b) Signal Manifold

Figure 5.3: Neighborhood Preserving

problem, the objective is to optimize:

$$P^* = \arg\min_{P \in \mathbb{R}^{m \times 2}} (P - Y_P)' J_P (P - Y_P) + \gamma_P P' L_P P \tag{5.1}$$

Here, P is the coordinate matrix of the mobile device to be determined; $J_P = diag(\delta_1, \delta_2, \ldots, \delta_m)$ is an indication matrix where $\delta_i = 1$ if the coordinate of the mobile device at time t_i is given and otherwise $\delta_i = 0$; $Y_P = [\mathbf{y}'_1, \mathbf{y}'_2, \ldots, \mathbf{y}'_m]'$ is an $m \times 2$ matrix supplying the calibration data where $\mathbf{y}_i$ is the given coordinate of the mobile device at time t_i if $\delta_i = 1$ and otherwise the value of $\mathbf{y}_i$ can be any, e.g., $\mathbf{y}_i = [0\ 0]$; γ_P controls the smoothness of the coordinates along the manifold; $L_P = D_P - W_P$ is the graph Laplacian; $W_P = [w_{ij}]_{m \times m}$ is the weight matrix and $w_{ij} = exp(-\|\mathbf{s}_i - \mathbf{s}_j\|^2 / 2\sigma_P^2)$ if $\mathbf{s}_i$ and $\mathbf{s}_j$ are neighbors along the manifold and otherwise $w_{ij} = 0$; $D_P = diag(d_1, d_2, \ldots, d_m)$ and $d_i = \sum_{j=1}^{m} w_{ij}$.

Setting the derivative of Equation (5.1) to zero,

$$2J_P(P - Y_P) + 2\gamma_P L_P P = 0 \tag{5.2}$$

the optimal solution is given by [36]

$$P^* = (J_P + \gamma_P L_P)^{-1} J_P Y_P \tag{5.3}$$

Similarly, the coordinates of the access points are given by

$$Q^* = \arg\min_{Q \in \mathbb{R}^{n \times 2}} (Q - Y_Q)' J_Q (Q - Y_Q) + \gamma_Q Q' L_Q Q \tag{5.4}$$

and

$$Q^* = (J_Q + \gamma_Q L_Q)^{-1} J_Q Y_Q \tag{5.5}$$

where $L_Q = D_Q - W_Q$ is the graph Laplacian, W_Q is the weight matrix and D_Q is constructed from W_Q.

Thus, when the locations of the mobile device and the access points are partially known, we can *co-localize* them by solving Equations (5.3) and (5.5) respectively. Alternatively, we can combine them into a single equation as

$$R^* = (J + \gamma_B L_B + \gamma_C L_C)^{-1} JY \tag{5.6}$$

Here, $R = [P'\ Q']'$ is the coordinate matrix of the mobile device and the access points; $Y = [Y'_P\ Y'_Q]'$ gives the label information; $J = \left[\begin{smallmatrix} J_P & 0 \\ 0 & J_Q \end{smallmatrix}\right]$ is the indication matrix; $L_B = \left[\begin{smallmatrix} L_P & 0 \\ 0 & 0 \end{smallmatrix}\right]$ and $L_C = \left[\begin{smallmatrix} 0 & 0 \\ 0 & L_Q \end{smallmatrix}\right]$ are the graph Laplacians.

In practice, the graph Laplacians L_B and L_C in Equation (5.6) are normalized [7, 85]. Figure 5.2(c) shows an example of the manifold-based *co-localization* when the locations of the mobile device at time t_A, t_B, t_C, t_D, t_E, t_F and the access points AP_2, AP_3, AP_4 are known. As can be seen, the trajectory of the mobile device is well grounded when compared to the ground truth shown in Figure 5.1. However, due to the limited number of access points, their locations are estimated badly, e.g., the location of AP_5.

In the following, we will combine the SVD-based and the Manifold-based *co-localization* together so that we can align the mobile device and the access points to the ground truth and to each other.

5.2.4 A Unifying Framework

So far, we have formulated the unsupervised *co-localization* based on SVD and the semi-supervised *co-localization* based on the manifold assumption using Equation (5.6) by exploiting the correlation within the mobile device and the access points. In this section, we integrate them through a unifying theory. Essentially, performing SVD on $A_{\mathcal{N}}$ is equivalent to solving the generalized eigenvalue problem [23]

$$L_A Z = D_A Z \Lambda \tag{5.7}$$

where $L_A = D_A - W_A$ is the graph Laplacian [19], $W_A = \left[\begin{smallmatrix} 0 & A \\ A' & 0 \end{smallmatrix}\right]$ and $D_A = \left[\begin{smallmatrix} D_1 & 0 \\ 0 & D_2 \end{smallmatrix}\right]$. The eigenvalues of the diagonal matrix $\Lambda = diag(\lambda_1, \lambda_2, \ldots, \lambda_{m+n})$ are ranked in *non-decreasing* order. $Z = [\mathbf{z}_1, \mathbf{z}_2, \ldots \mathbf{z}_{m+n}]$ are the eigenvectors. $[P'\ Q']' =$

$[\mathbf{z}_2\ \mathbf{z}_3]$. Note that we skip the first eigenvector $\mathbf{z}_1$ since the solution is trivial. Furthermore, it is interesting to see that we have $\lambda_i = 1 - \sigma_i$ where $i = 1, 2, \ldots, r$ [23]. Detailed analysis and comparison of LSI, SVD and graph Laplacian can be found in Latent Semantic Indexing [22, 23, 40], Bipartite Co-Clustering [23, 103] and Fiedler Embeddings [40].

Putting these together, our objective is to optimize:

$$R^* = \underset{R \in \mathbb{R}^{(m+n)\times 2}}{\arg\min}\ (R - Y)' J (R - Y) + \gamma R' L R \tag{5.8}$$

The first term measures the fitting error and the second term constrains the smoothness among the mobile device and the access points. $L = \gamma_A L_A + \gamma_B L_B + \gamma_C L_C = D - W$. The solution is given by:

$$R^* = (J + \gamma L)^{-1} J Y \tag{5.9}$$

Assuming that there is no uncertainty in our calibration matrix Y, γ is set to a small positive value. The result is directly related to harmonic functions, which are smooth functions on the graph such that the coordinate or a mobile device or an access point $\mathbf{r}_i$ in $R = [\mathbf{r}'_1, \mathbf{r}'_2, \ldots, \mathbf{r}'_{m+n}]'$ is determined by the average of its neighbors:

$$\mathbf{r}_i = \frac{\sum_j w_{ij} \mathbf{r}_j}{\sum_j w_{ij}} \tag{5.10}$$

where $W = [w_{ij}]_{(m+n)\times(m+n)} = \begin{bmatrix} \gamma_B W_P & \gamma_A A \\ \gamma_A A' & \gamma_B W_Q \end{bmatrix}$.

In practice, we optimize the objective function over the *normalized* graph Laplacian [7, 85] to balance the weights of vertices by substituting $R = D^{-1/2} F$ into Equation (5.8)

$$F^* = \underset{F \in \mathbb{R}^{(m+n)\times 2}}{\arg\min}\ (D^{-1/2} F - Y)' J (D^{-1/2} F - Y) + \gamma_{\mathcal{N}} F' L_{\mathcal{N}} F$$

where $L_{\mathcal{N}} = D^{-1/2} L D^{-1/2}$ is the *normalized* graph Laplacian. The optimal F is given by

$$F^* = (J D^{-1/2} + \gamma_{\mathcal{N}} L_{\mathcal{N}})^{-1} J Y \tag{5.11}$$

Substituting $F = D^{1/2} R$ back to Equation (5.11), the locations of the mobile device and the access points are given by

$$R^* = D^{-1/2} (J D^{-1/2} + \gamma_{\mathcal{N}} L_{\mathcal{N}})^{-1} J Y \tag{5.12}$$

An example of applying the above *co-localization* algorithm using Equation (5.9) is shown in Figure 5.2(d) when the locations of the mobile device at time t_A, t_B, t_C, t_D t_F and the access point AP_4 are known. As can be seen, most of the locations are correctly recovered while using less calibration data than that in Figure 5.2(c).

5.3 Experiments

5.3.1 Experimental Setup

We evaluated the performance of the *co-localization* algorithm on three sets of different devices and test-beds:

(1) Wireless Local Area Network (WLAN): a person carrying an IBM© T42 notebook, which is equipped with an Intel© Pro/2200GB internal wireless card, walked in an indoor environment of about $60m \times 50m$ in size as shown in Figure 5.1. A total of 2000 examples are collected with sample rate $2Hz$. The ground-truth location labels are obtained by referring to landmark points such as doors, corners and dead-ends. The localization area is composed by one-dimensional hallways.

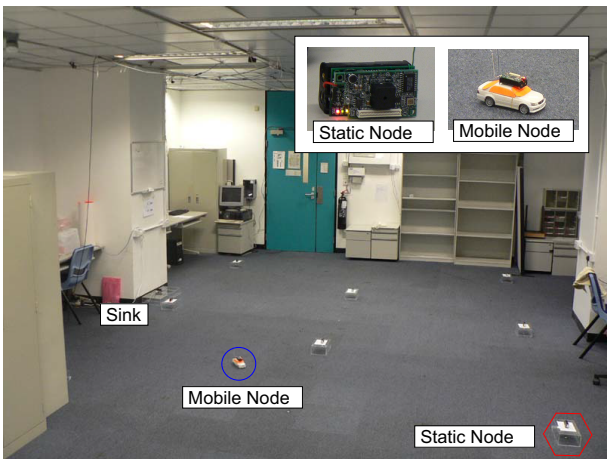


Figure 5.4: WSN Test-bed

(2) Wireless Sensor Network (WSN): We used a number of MICA2 sensors from Crossbow© for experiments. As can be seen from Figure 5.4, 8 static nodes (AP) were placed in a room of size $5m \times 4m$. One mobile node (MD) was attached on the top of a robot that moved around freely in this domain. A total of 4000 examples are

Table 5.2: The experimental setups of WLAN, WSN and RFID

Infrastructure	AP	MD	Test-bed	Scale	Motion Pattern
WLAN	5 Access Points	1 Notebook	Hallway	$60m \times 50m$	Mobile (robot)
WSN	8 Static Nodes	1 Mobile Node	Room	$5m \times 4m$	Mobile (human)
RFID	4 RFID Readers	30 RFID Tags	Room	$5m \times 4m$	Static

collected with sample rate $2Hz$. The ground-truth location labels of the mobile node were supported by the cameras deployed on the ceiling. The localization area is a two-dimensional plane.

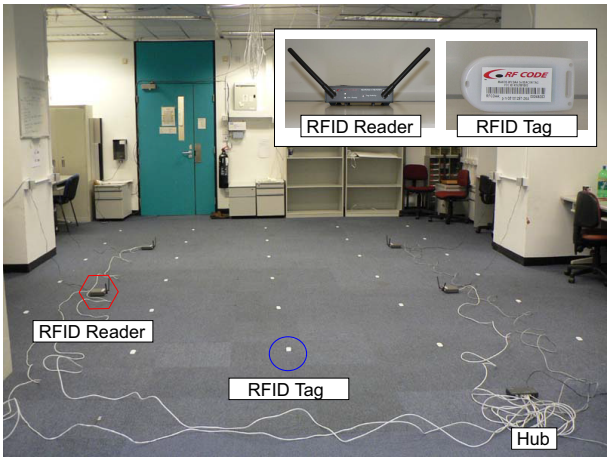


Figure 5.5: RFID Test-bed

(3) Radio Frequency Identification (RFID): We used 4 Mantis readers (AP) and 30 tags (MD) from RF Code©. They were all deployed as stationary nodes, which is shown in Figure 5.5. A total of 2000 examples were collected. The ground truth locations were marked down manually.

We summarize our three experimental setups in Table 5.2.

5.3.2 Experimental Results

For comparison, we also run the following baseline algorithms (1) LANDMARC, a nearest-neighbor weighting based method designed for RFID localization [65]; (2) Support Vector Regression (SVR), a simplified variant of a kernel-based method used for WSN localization [64]; (3) RADAR, a K-Nearest-Neighbor method for WLAN localization [5].

In each experiment, we randomly pick up 500 examples for training and the rest

for testing. The training data is further split into *labelled* and *unlabelled* parts. The results shown in Figure 5.7 are averaged over 10 repetitions for reducing statistical variability. All results are measured in *relative error distances*, which are error distances in percentage while referring to the maximal error distance in each figure for easy comparison. All parameters are determined from a validation subset. LANDMARC, RADAR and SVR use the *labelled* part of training data only.

In contrary, the *co-localization* method used both *labelled* and *unlabelled* data. We will show how our algorithm benefits from the additional *unlabelled* data and reduces calibration effort. In all, we tested on two configurations for the *co-localization* method: (1) '**Co-Localization no AP**' uses partially *labelled* data from mobile devices for training, in which we tries to recover the locations of the access points; and (2) '**Co-Localization with AP**' repeats the same experiments with the locations of all access points known.

Figures 5.6(a), 5.6(c) and 5.6(e) show the localization error of different mobile devices by varying the number of labelled examples in a training subset which size is fixed to be 500. The three figures could be read in two directions. First, if we compare the results vertically in each figure, we can see how the *unlabelled* data help improve the result in the proposed methods. For example in Figure 5.6(e), most compared methods have a relative error distance of around 80% when using 50 *labelled* examples. In contrary, the proposed methods have an error of around 40% by employing additional 450 *unlabelled* examples. Secondly, if we compare the results horizontally in each figure, we can find how our methods reduce calibration effort. For example in Figure 5.6(a), most compared methods have a relative error distance of around 60% when all 500 examples are *labelled*. The proposed '**Co-Localization with AP**' has a similar performance when using 50 *labelled* and 450 *unlabelled* examples. We save the calibration effort.

We found that the mobility of the mobile device and the environment complexity are two main factors that affected the performance of the *co-localization* algorithm. In a static and plane-shaped test-bed (Figure 5.6(a)), the radio signals are less noisy and the '**Co-Localization no AP**' configuration demonstrated similar performance as RADAR, LANDMARC and SVR when the number of *labelled* examples is small. In a mobile and complex environment, as shown in (Figure 5.6(e)), the radio signal is more noisy and the '**Co-Localization no AP**' performed much better and more robust than

the compared methods. We have also tried some other combinations of experiments that led to a similar conclusion, such as using RFIDs in a mobile scenario.

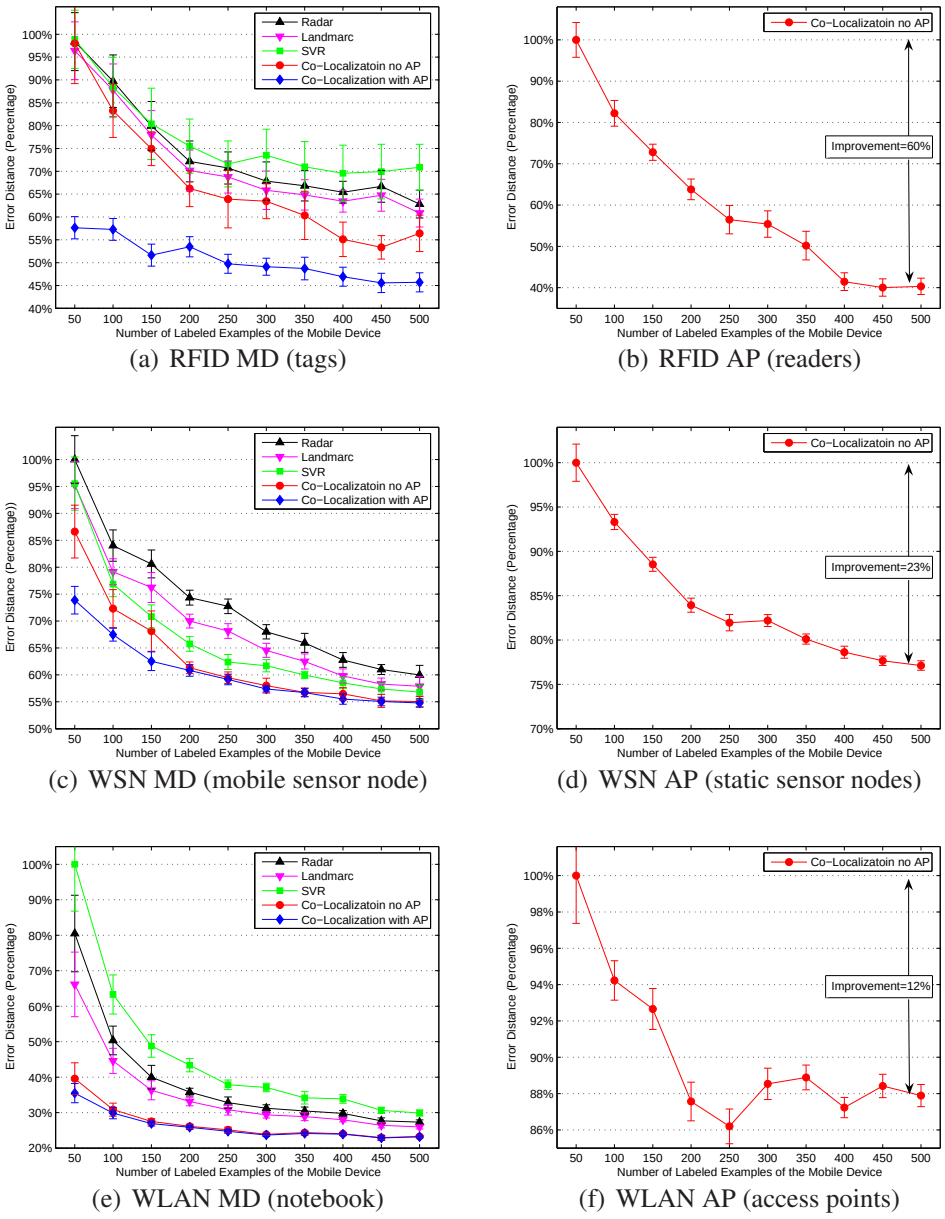
While comparing the results of '**Co-Localization no AP**' and '**Co-Localization with AP**' in Figures 5.6(a), 5.6(c) and 5.6(e), we can find that knowing the locations of access points is more helpful for localizing the mobile devices in a static and planar scenario (Figure 5.6(a)) than in a mobile and complex environment (see Figure 5.6(e)).

Similarly, we can see from Figures 5.6(b), 5.6(d) and 5.6(f) that knowing the locations of mobile devices are more helpful for localizing access points in a static and plane-shaped scenario rather than a mobile and complex environment.

5.4 Summary

In this chapter, we have considered a second aspect of the localization problem, where we recover both the locations of the client and APs together when only part of the data are labelled. We have developed a novel graph Laplacian approach to solve the problem of simultaneously recovering the locations of both mobile devices and access points. In our co-localization framework, we find the relative locations of mobile devices and access points by exploiting a SVD based method, and find the absolute locations using a small collection of labelled data through graph Laplacian methods. Our extensive experiments in three different configurations showed that we can achieve high performance with much less calibration effort as compared to several previous approaches.

The significance of the work is that we can leverage both the knowledge of the access point locations and the mobile device trajectories to obtain more accurate localization. Indeed this is one of our future works. Besides, we would try to evaluate the performance in a large-scale and dynamic environment, e.g., in a city level and in different time. We may also vary more parameters such as number of access points and their deployment density and study the robustness.



(a) RFID MD (tags)

(b) RFID AP (readers)

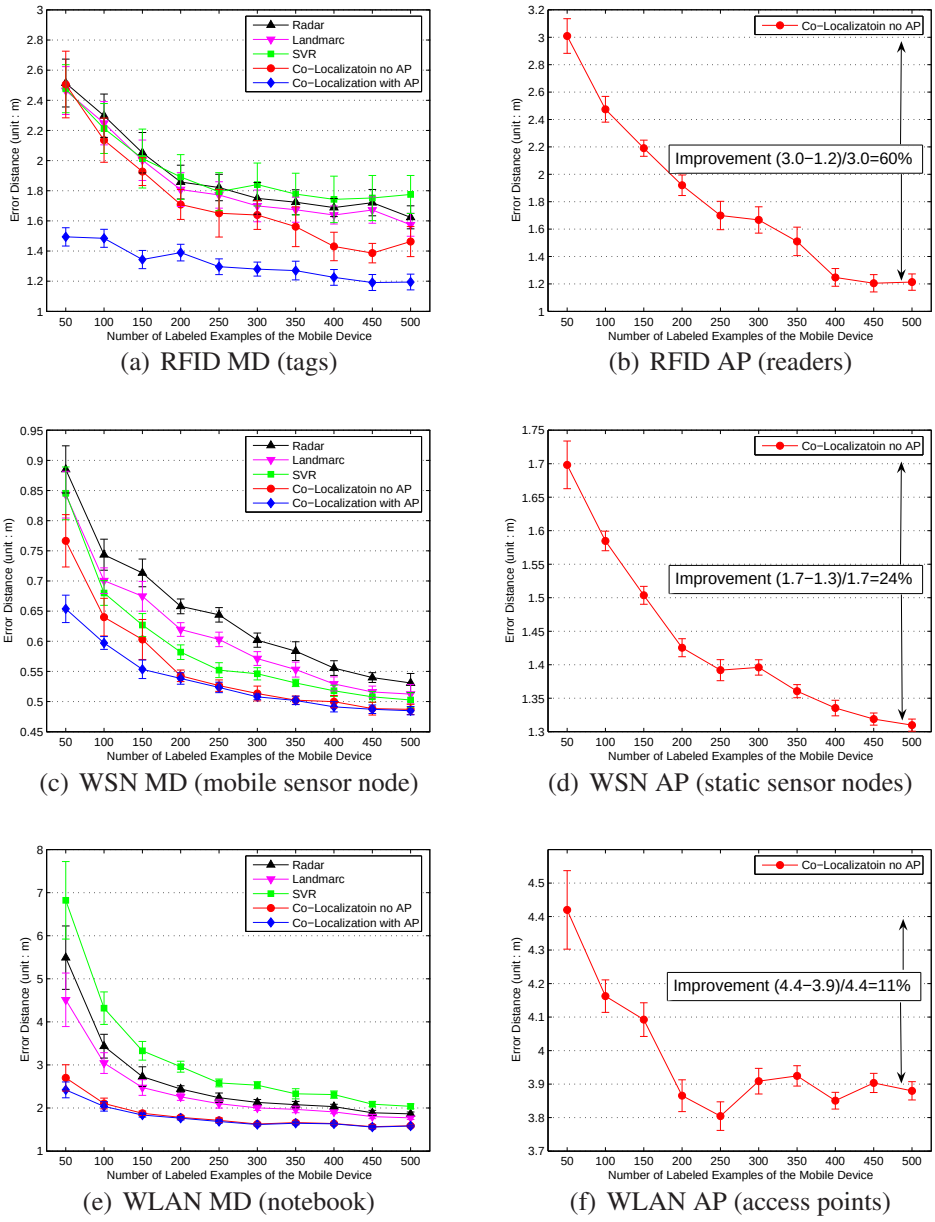
(c) WSN MD (mobile sensor node)

(d) WSN AP (static sensor nodes)

(e) WLAN MD (notebook)

(f) WLAN AP (access points)

Figure 5.6: Experimental Results over 10 Repetitions (Mean and Std.): MD for Mobile Device; AP for Access Point



(a) RFID MD (tags)

(b) RFID AP (readers)

(c) WSN MD (mobile sensor node)

(d) WSN AP (static sensor nodes)

(e) WLAN MD (notebook)

(f) WLAN AP (access points)

Figure 5.7: Experimental Results over 10 Repetitions (Mean and Std.): MD for Mobile Device; AP for Access Point

CHAPTER 6

ONLINE CO-LOCALIZATION BY DIMENSION REDUCTION

This chapter addresses the problem of recovering the locations of both mobile devices and access points from radio signals that come in a stream manner, a problem which we call *online co-localization*, by exploiting both labelled and unlabelled data from mobile devices and access points. Many tracking systems function in two phases: an *offline training phase* and an *online localization phase*. In the training phase, models are built from a batch of data that are collected offline. Many of them can not cope with a dynamic environment in which calibration data may come sequentially. In such case, these systems may gradually become inaccurate without a manually costly re-calibration. To solve this problem, we proposed an *online co-localization* method that can deal with labelled and unlabelled data stream based on semi-supervised manifold-learning techniques. Experiments conducted in wireless local area networks show that we can achieve high accuracy with less calibration effort as compared to several previous systems. Furthermore, our method can deal with online stream data relatively faster than its two-phase counterpart.

6.1 The Online Co-Localization Problem

With the recent advance in pervasive computing and mobile technology, tracking wireless devices using received-signal-strength (RSS) has attracted intense interest in many research communities. It is a useful task in robotics and activity recognition. It is also a difficult task since radio signals usually attenuate in a highly nonlinear and uncertain way in a complex environment where client devices may be moving. Existing approaches to RSS localization fall into two main categories [26]: (1) radio propagation models [62, 80], which rely on the knowledge of access point locations; (2) statistical machine learning models [64, 57, 5], which require a large amount of costly calibration.

In general, machine-learning-based systems using RSS values function in two phases [68]: an *offline training phase* and an *online localization phase*. In the offline phase, a

probabilistic model is trained by considering the signal strength values received from the access points at selected locations in the area of interest. These values comprise the training data gathered from a physical region, which are used to calibrate a probabilistic location-estimation system. In the online localization phase, the real-time signal strength samples received from the access points are used to estimate the current location based on the learned model.

However, in many applications, access points are not deployed in a static environment in which calibrated and uncalibrated data come in a stream manner. Access points may be removed, relocated and added for better coverage and link quality. In either case, a localization system may gradually become inaccurate without a manually costly re-calibration and re-run the whole training process. It is also wasteful to discard previous computation results. A better idea is to construct an online model where calibration data come in stream. Previous works have been done in online learning over graph [41], incremental neighborhood graphs [105], incremental ISOMAP [55, 54] and incremental LLE [49]. [44] models the real-time scenario with a spatio-temporal manifold. More specifically in wireless and sensor networks, [29] proposes an online algorithm for localizing beacon nodes with a moving target. [28] describes a similar problem when beacon nodes are cameras. [92] presents a framework for simultaneously localization and mapping with ultrasonic sensors based on Baysian Filter [27]. [25] shows WiFi SLAM using Gaussian Process model.

In this chapter, we address the problem of recovering the locations of both mobile devices and access points from radio signals that come in a stream manner, a problem which we call *online co-localization*, by exploiting both labelled and unlabelled data from mobile devices and access points. The proposed method is based on online and incremental manifold-learning techniques [54, 49, 44], semi-supervised techniques that can cope with labelled and unlabelled data that come sequentially.

We test our *online co-localization* in a Wireless Local Area Network (WLAN). Experiments show that we can achieve high accuracy with less calibration effort as compared to several previous systems. Furthermore, our method can incrementally deal with data stream *online* relatively faster than its two-phase counterpart.

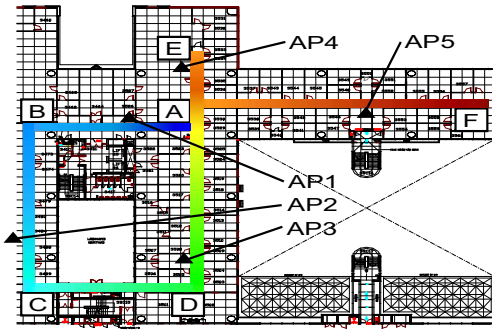


Figure 6.1: WLAN Test-bed

Table 6.1: Signal Strength (unit:dBm)

	AP_1	AP_2	AP_3	AP_4	AP_5
t_A	-40		-60	-40	-70
t_B	-50	-60		-80	
t_C		-40	-70		
t_D	-80		**-40**	-70	
$t_{A'}$	-40		-70	-40	-60
t_E	-40		-70	-40	-80
t_F	-80			-80	-50

(All values are rounded for illustration)

6.2 Methodology

6.2.1 Problem Statement

Co-localization addresses the problem of recovering the locations of both mobile devices and access points, by exploiting both *labelled* and *unlabelled* data from both mobile devices and access points [69]. *Co-localization* can be done in a traditional Two-Phase manner: an *Offline Training Phase* and an *Online Localization Phase*. However, in a dynamic environment where calibration data come sequentially, it will be inefficient to build the model repeatedly. A better idea is to adjust the current model *online*.

Consider a 2-dimensional *online co-localization* problem: Assume that a user holds a mobile device and navigates in an indoor wireless environment $\mathcal{C} \subseteq \mathbb{R}^2$ of n access points, which can periodically send out beacon signals. At some time t_i, the RSS values from all the n access points are measured by the mobile device to form a row vector $\mathbf{s}_i = [s_{i1}\ s_{i2}\ \ldots\ s_{in}] \in \mathbb{R}^n$. As time elapses, these row vectors come in a stream manner. After m time ticks, we get a sequence of m signal strength vectors form an $m \times n$ matrix $S = [\mathbf{s}'_1\ \mathbf{s}'_2\ \ldots\ \mathbf{s}'_m]'$, where "prime" is used to denote matrix transposition.

Here, the locations of some access points and the mobile devices at some time t are known or *labelled*, while the rest are *unlabelled*.

Our objectives are stated as follows: In an *online and incremental* manner, we wish to estimate the $m \times 2$ location matrix $P = [\mathbf{p}'_1, \mathbf{p}'_2, \ldots, \mathbf{p}'_m]'$ where $\mathbf{p}_i = [p_{i1}\ p_{i2}] \in \mathcal{C}$ is the location of the mobile device at time t_i and the $n \times 2$ location matrix $Q = [\mathbf{q}'_1, \mathbf{q}'_2, \ldots, \mathbf{q}'_n]'$ where $\mathbf{q}_j = [q_{j1}\ q_{j2}] \in \mathcal{C}$ is the location of the j access points.

Our objectives are to determine and update the locations of all of the remaining access points and the trajectory of the mobile device in real-time as partially calibrated data come sequentially. Note that m is not a constant value. As time elapses, m may increase from 1, 2, ..., to any number. We want to dynamically adjust the model when observing new data without (or with) an *offline training phase*. We call this problem *online co-localization*.

Example 1 As an example, Figure 6.1 shows an indoor 802.11 wireless LAN environment of size about $60m \times 50m$. It is equipped with $n = 5$ access points. A user with an IBM T42 notebook that is equipped with an Intel Pro/2200BG internal wireless card walks from A through B, C, D, A, E to F at time $t_A, t_B, t_C, t_D, t_{A'}, t_E, t_F$. Correspondingly, a total number of $m = 1, 2, \ldots, 7$ signal strength vectors are incrementally extracted. The final 7×5 matrix S is shown in Table 6.1. By walking from A to F in the hallways, we collected 500 signal strength vectors from 5 access points. Note that the blank cells denote the missing values, which we can fill in a small default value, e.g., $-100dBm$.

Our task is to dynamically update the trajectory matrix P of the mobile device at each time when new data come and to determine the location matrix Q of the access points $AP_1, AP_2, \ldots, AP_5$ in an *online* manner.

6.2.2 Domain Characteristics

There are four main characteristics about received-signal-strengths by observing the data in Table 6.1:

1. Considering two *rows* of the data, the mobile device at two different time may be spatially close if their pairwise signal strengths are similar from most access

points, e.g., the time t_A and $t_{A'}$.

2. Considering two *columns* of the data, two access points may be spatially close if their pairwise signal strengths are similar most of the time, e.g., AP_1 and AP_4.

3. Considering a *single cell* s_{ij} of the data, the mobile device and the j access point may be spatially close to each other at time t_i if the signal is strong, e.g., the mobile device is close to AP_3 at time t_D.

4. Considering two *neighbored rows* of the data, the mobile device at two consecutive time may be spatially close if their time interval is small by assuming that a user may not move too fast or too irregularly. For example, the locations of the mobile device at time $t_{A'}$ and t_E are close since $|t_{A'} - t_E| < \Delta T$.

It is not surprising that the above observations are related to the assumption of manifold-learning techniques: When the locations of some access points and the mobile device at some time are known, we can ground the unknown coordinates by exploiting the geometry of the signal distribution. Manifold-based methods generally assume that if two points are close in the intrinsic geometry of the marginal distribution, their conditional distributions are similar [8, 36], which approximately holds in our above observations and several previous works [70, 69]

6.2.3 Solution I: *Two-Phase Co-Localization*

First of all, we can solve the *co-localization* problem in two phases: an *offline training phase* and an *online localization phase*. In the offline phase, suppose that we have collected a batch of data, which are partially *labelled* and the labels can be from mobile devices and access points. We wish to build a model using the whole batch of data. In the online phase, we use the model to estimate the unknown coordinates one by one when signal vectors come in a stream manner.

Offline Training Phase

When the manifold assumption holds, the optimal solution is give by $\mathbf{f}^* = \arg\min_{\mathbf{f}} \Sigma_{i=1}^{l} |f_i - y_i|^2 + \gamma \mathbf{f}^T L \mathbf{f}$ [36] where the first term measures the fitting error and the second term poses the smoothness along the manifold and L is the graph Laplacian [19].

First of all, we can express the similarity within all the collected signal vectors $\mathbf{s}_i$ $(i = 1, 2, \ldots, m)$ by constructing the neighborhood graph and its graph Laplacian matrix L_P [19]. The objective is to optimize:

$$P^* = \underset{P \in \mathbb{R}^{m \times 2}}{\arg\min} (P - Y_P)' J_P (P - Y_P) + \gamma_P P' L_P P \tag{6.1}$$

where P is the coordinate matrix of the mobile device to be determined; $J_P = diag(\delta_1, \delta_2, \ldots, \delta_m)$ is an indication matrix where $\delta_i = 1$ if the coordinate of the mobile device at time t_i is given and otherwise $\delta_i = 0$; $Y_P = [\mathbf{y}'_1, \mathbf{y}'_2, \ldots, \mathbf{y}'_m]'$ is an $m \times 2$ matrix supplying the calibration data where $\mathbf{y}_i$ is the given coordinate of the mobile device at time t_i if $\delta_i = 1$ and otherwise the value of $\mathbf{y}_i$ can be any, e.g., $\mathbf{y}_i = [0\ 0]$; γ_P controls the smoothness of the coordinates along the manifold; $L_P = D_P - W_P$ is the graph Laplacian; $W_P = [w_{ij}]_{m \times m}$ is the weight matrix and $w_{ij} = exp(-\|\mathbf{s}_i - \mathbf{s}_j\|^2 / 2\sigma_P^2)$ if $\mathbf{s}_i$ and $\mathbf{s}_j$ are neighbors along the manifold and otherwise $w_{ij} = 0$; $D_P = diag(d_1, d_2, \ldots, d_m)$ and $d_i = \sum_{j=1}^{m} w_{ij}$.

Setting the derivative of Equation (6.1) to zero, the optimal solution is given by [36]

$$P^* = (J_P + \gamma_P L_P)^{-1} J_P Y_P \tag{6.2}$$

We also construct the neighborhood graph for access points in a similar way, the optimal solution is given by

$$Q^* = \underset{Q \in \mathbb{R}^{n \times 2}}{\arg\min} (Q - Y_Q)' J_Q (Q - Y_Q) + \gamma_Q Q' L_Q Q \tag{6.3}$$

where $L_Q = D_Q - W_Q$ is the graph Laplacian, W_Q is the weight matrix and D_Q is constructed from W_Q.

Furthermore, we encode the similarity between access points and mobile devices by transforming the signal matrix $S = [s_{ij}]_{m \times n}$ to a non-negative weight matrix $A = [a_{ij}]_{m \times n}$ by a Gaussian kernel $a_{ij} = exp(-|s_{ij} - s^{max}|^2 / 2\sigma_A^2)$ where s^{max} is the maximal signal strength detected, e.g., the signal strength around an access point or $-30dBm$. From the weight matrix A, we construct the graph Laplacian $L_A = D_A - W_A$ where $W_A = \begin{bmatrix} 0 & A \\ A' & 0 \end{bmatrix}$ and $D_A = \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix}$. We also reform L_P and L_Q to larger matrices with $L_B = \begin{bmatrix} L_P & 0 \\ 0 & 0 \end{bmatrix}$ and $L_C = \begin{bmatrix} 0 & 0 \\ 0 & L_Q \end{bmatrix}$. Now L_A, L_B and L_C are graph Laplacians of the same size. L_A describes the similarity between mobile devices and access points. L_B and L_C express the similarity within them respectively.

Putting these together, our objective is to optimize:

$$R^* = \underset{R \in \mathbb{R}^{(m+n)\times 2}}{\arg\min} \; (R-Y)'J(R-Y) + \gamma R'LR \tag{6.4}$$

where $R = [\mathbf{r}'_1, \mathbf{r}_2, \ldots, \mathbf{r}'_{m+n}]' = [P'\ Q']'$ is the coordinate matrix of the mobile device and the access points; $Y = [Y'_P\ Y'_Q]'$ supports the partial labels; $J = \left[\begin{smallmatrix} J_P & 0 \\ 0 & J_Q \end{smallmatrix}\right]$ is the indication matrix; $L = \gamma_A L_A + \gamma_B L_B + \gamma_C L_C = D - W$ is the graph Laplacian. The optimal solution is given by:

$$R^* = (J + \gamma L)^{-1} JY \tag{6.5}$$

We can export the estimated coordinates of the mobile device trajectory P^* and access point locations Q^* from $R^* = [P^{*\prime}\ Q^{*\prime}]'$.

Online Localization Phase

In the localization phase, the location of a new signal strength vector $\mathbf{s}_i$ is predicted as follows:

1. Find the k neighbors closest to $\mathbf{s}_i$ in the training data $S = [\mathbf{s}'_1\ \mathbf{s}'_2\ \ldots\ \mathbf{s}'_m]'$. Let $\mathcal{C}_i$ be the index set of the k nearest neighbors. Besides, we link $\mathbf{s}_i$ to those access points from which we can detect the radio signal. We also link $\mathbf{s}_i$ to $\mathbf{s}_{i-1}$ in order to pose the temporal constraint by assuming that a user may not move too fast ($t_i - t_{i-1} < \Delta T$). Denote the index set for these additional links as $\mathcal{B}_i$.

2. Approximately, we can predict the location using a property of harmonic functions [106, 36], which are smooth functions on the graph such that $\mathbf{r}_i$ is determined by the weighted average of its neighbors. This property holds if there is no uncertainty in the labelled locations of matrix P during training ($\gamma \to 0$).

$$\tilde{\mathbf{r}}_i \approx \frac{\sum_{j \in \mathcal{C}_i \cup \mathcal{B}_i} w_{ij} \mathbf{r}_j}{\sum_{j \in \mathcal{C}_i \cup \mathcal{B}_i} w_{ij}} \tag{6.6}$$

   Note that the above $\tilde{\mathbf{r}}_i$ is an approximation because adding $\mathbf{s}_i$ to the existing neighborhood graph from the train data may slightly change the graph structure: We have linked the i^{th} node to the node set $\mathcal{C}_i$; However, we have not yet eliminated any existing edge in the graph to maintain the k-neighbor relationship among all nodes.

6.2.4 Solution II: *Online Co-Localization*

We will extend the above *Two-Phase Co-Localization* model to an *online* version. We wish that it can dynamically adjust itself when new data come sequentially in real-time. The key point is how to add the new data into the learned graph by updating the k-neighbor relationship and the corresponding weight matrix W. This can be done repeatedly in two online steps: Predict and Update.

Predict

Given a new signal vector $\mathbf{s}_i$ at time t_i, we find its k nearest neighbors and use Equation (6.6) in the above *online localization phase* for predicting the location $\tilde{\mathbf{r}}_i$.

Update

The addition and deletion of nodes can modify the neighborhood graph and the corresponding graph Laplacian. We use the method described in [54] for updating the neighborhood graph structure locally.

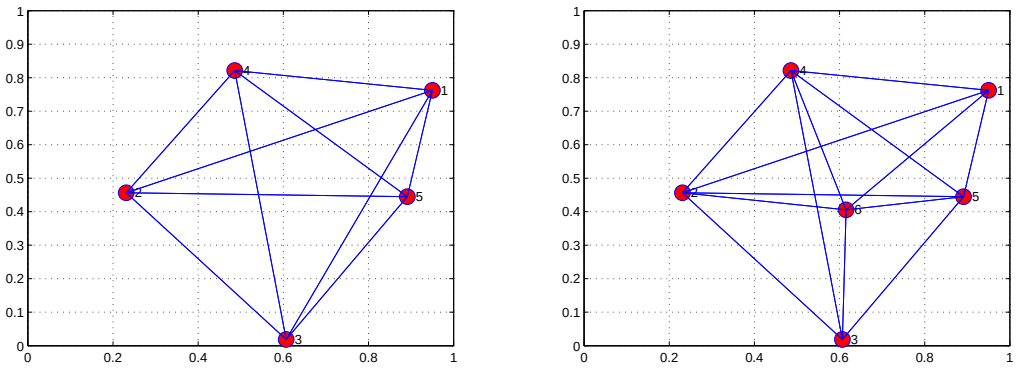
•**Node Addition** Let $\mathcal{A}_i^+$ and $\mathcal{D}_i^+$ be the set of edges to be added and deleted after inserting v_i to the neighborhood graph, respectively. Let τ_i be the index of the k^{th} nearest neighbor of v_i. So, v_j is in the k nearest neighborhood of v_i if $\Delta_{ij} \leq \Delta_{i,\tau_i}$. When $\Delta_{i,\tau_i} > \Delta_{i,\tau_i}$, v_{n+1} replaces v_{τ_i} in the knn neighborhood of v_i. It is easy to see that:

$$\mathcal{A}_i^+ = \{e(j, n+1) \ : \ j \in \mathcal{C}_i \ or \ \Delta_{j,\tau_j} > \Delta_{ji}\}$$

$$\mathcal{D}_i^+ = \{e(j, \tau_j) \ : \ \Delta_{j,\tau_j} > \Delta_{j,i} \ and \ \Delta_{\tau_j,j} > \Delta_{\tau_j,l_j}\}$$

where l_i is the index of the k^th nearest neighbor of v_{τ_j} after inserting v_i in the graph.

•**Node Deletion** Similarly, let $\mathcal{A}_i^-$ and $\mathcal{D}_i^-$ denote the set of edges to be added and deleted after removing v_i from the neighborhood graph, respectively. The graph update can be done as follows:



(a) A 4-Nearest-Neighbor graph that has 5 nodes (b) Add Node 6 to the graph

Figure 6.2: Edge addition and deletion for adding Node 6

$$\mathcal{A}_i^- = \{e(i,h_i)\} \text{ where } h_i \text{ is the } (k+1)^{th} \text{ nearest neighbor before removing } v_i \text{ in the graph.}$$

$$\mathcal{D}_i^- = \{e(i,j) \ : \ j \in \mathcal{C}_i\}$$

Fig. 6.2 shows an example of node addition. Fig. 6.2(a) is a 4-nearest-neighbor graph that has 5 nodes. By adding Node 6 to the graph, edges are added and deleted to maintain the neighborhood consistency. As can be seen in Fig. 6.2(b), five edges are added from the existing nodes to the new node. The edge between Node 1 and 3 is removed. Note that, a node may have more than K edges since Node i and j are connected either if Node i is within the K nearest neighbors of Node j or the other way around.

After updating the neighborhood graph, it is straight-forward to modify the corresponding weight matrix W. For an added edge $e(i,j)$, we set both the values of w_{ij} and w_{ji} because the neighborhood graph is symmetric. If it is a deleted edge, we clear the values of w_{ij} and w_{ji}. The graph Laplacian $L = D - W$ can be updated in a similar way.

Finally, we have to re-estimate the location matrix $R = [P'Q']'$ of the mobile devices and the access points so that it can reflect the change of the neighborhood graph and the new graph Laplacian L. Instead of using Equation (6.5) for solving R, we update R by iteration. In each iteration cycle, we apply:

$$\mathbf{r}_i^{new} \leftarrow \frac{\sum_{j \in \mathcal{C}_i \cup \mathcal{B}_i} w_{ij} \mathbf{r}_j^{old}}{\sum_{j \in \mathcal{C}_i \cup \mathcal{B}_i} w_{ij}} \ (i = 1, 2, \ldots, m+n) \tag{6.7}$$

We use the predicted $\tilde{\mathbf{r}}_i$ $(i = 1, \ldots, m+n)$ as the initial values for iteration. Furthermore, the weight matrix W does vary too much after addition or deletion. We can obtain very good estimation after a few iterations.

Example 2 A user with a mobile device walks in the office area shown in Figure 6.1. The mobile device periodically collects signal vectors. The user can mark down his location when he walks by some landmark points such as corners and dead-ends of the hallways $(A, B, \ldots, F)$. Thus, the data that come in stream are partially *labelled*. By applying the *online co-localization* method, we continuously update the recovered locations of the mobile devices and the access points. Figure 6.3 shows the *online co-localizaiton* results at six key frames when the user walks by $A, B, \ldots, F$. As can be seen, the locations of the user trajectory and the access points are dynamically calibrated when obtaining new data. For example, AP_3 gradually converges to its true location.

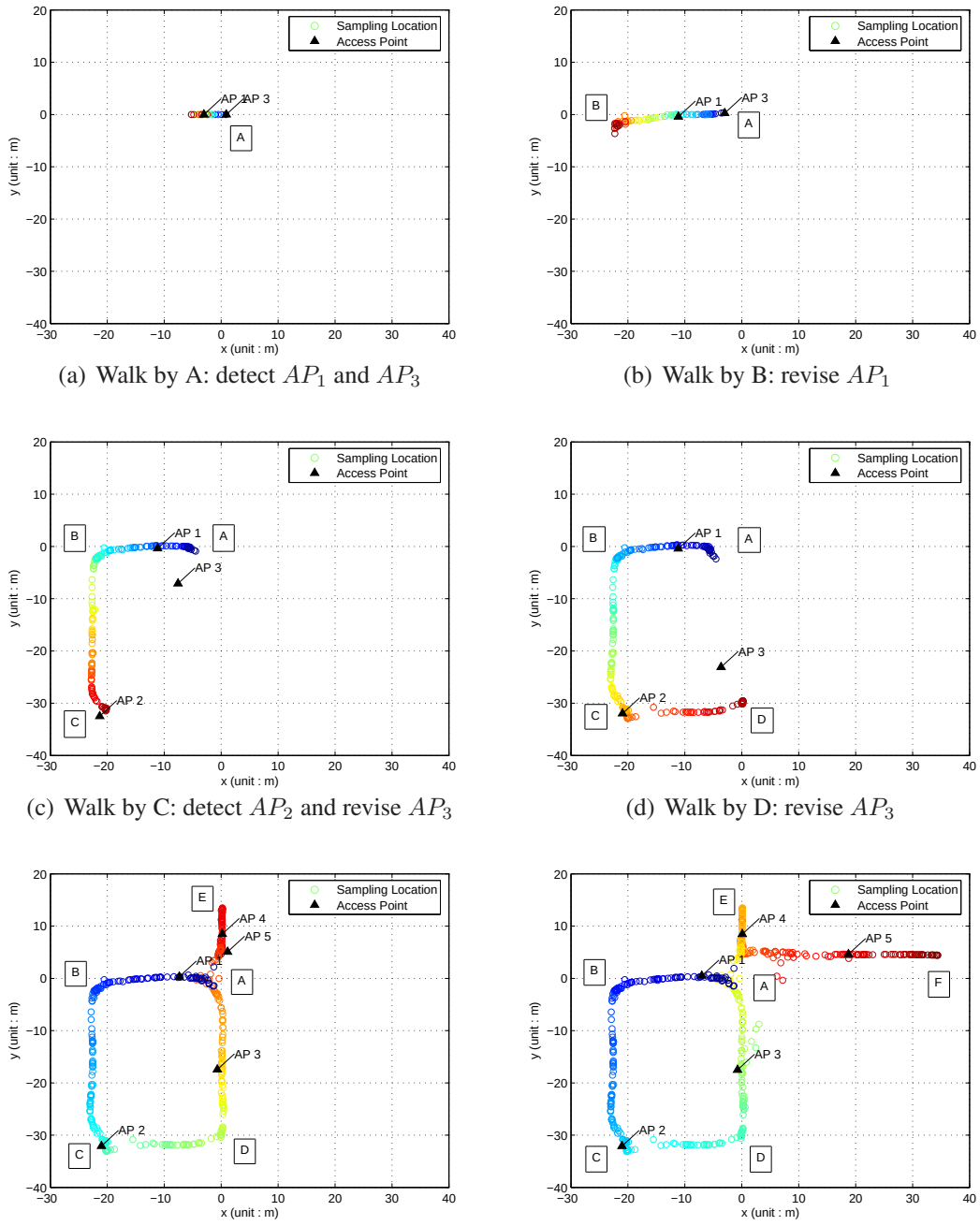
6.3 Experiments

6.3.1 Accuracy and Calibration Effort

We evaluated the performance of the *co-localization* algorithm in an 802.11 WLAN as shown in Figure 6.1. A person carried an IBM T42 laptop which is equipped with an Intel Pro/2200GB internal wireless card and walked in the environment. A total of 2000 examples are collected sequentially with sample rate $2Hz$. The ground-truth location labels are obtained by referring to landmark points such as doors, corners and dead-ends.

For comparison, we also run the following baseline algorithms (1) LANDMARC, a nearest-neighbor weighting based method [65]; (2) Support Vector Regression (SVR), a simplified variant of a kernel-based method [64]; (3) RADAR, a K-Nearest-Neighbor method [5].

In each experiment, we randomly pick up a sequence of 500 examples for training and the rest for testing. The training data are further split into *labelled* and *unlabelled*



(a) Walk by A: detect AP_1 and AP_3

(b) Walk by B: revise AP_1

(c) Walk by C: detect AP_2 and revise AP_3

(d) Walk by D: revise AP_3

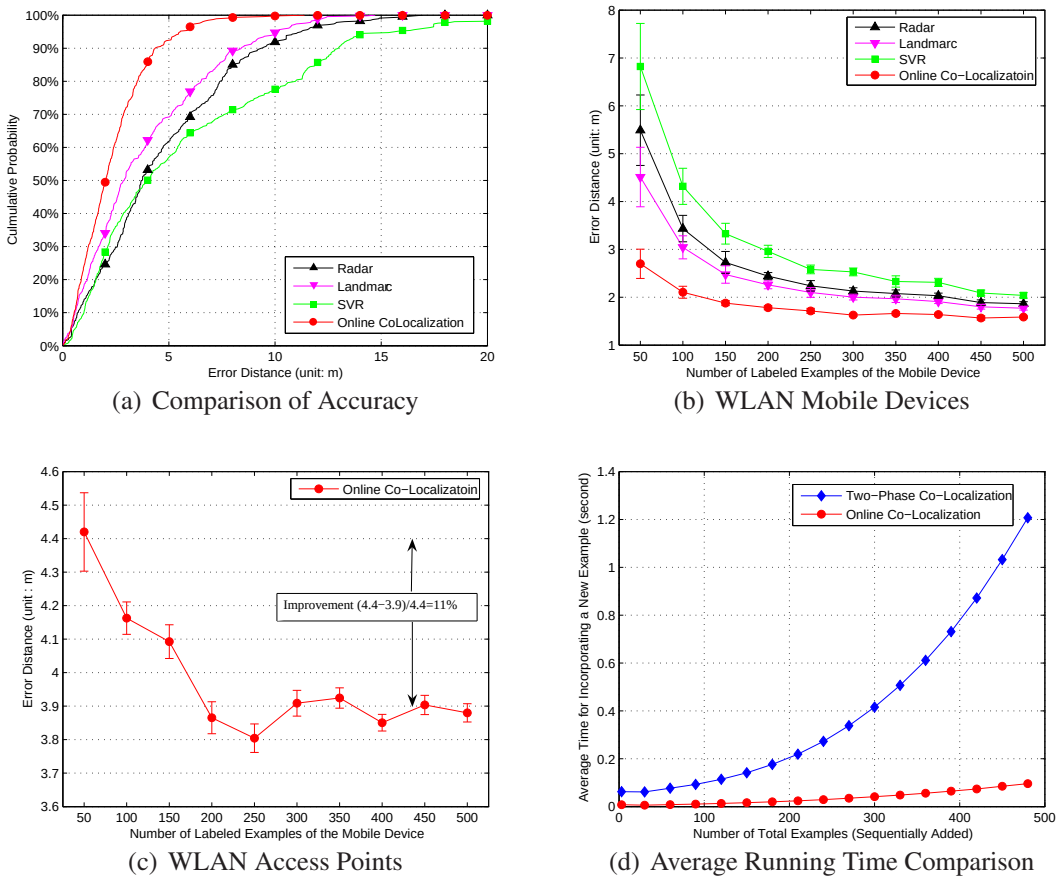
(e) Walk by E: detect AP_4 and AP_5

(f) Walk by F: revise AP_5

Figure 6.3: Illustration of the Online Co-Localization when a user walks from A through $B, \ldots, D$ to F

parts. The results shown in Figure 6.4 are averaged over 10 repetitions for reducing statistical variability. LANDMARC, RADAR and SVR use the *labelled* part of training data only. In contrary, the *online co-localization* method uses both *labelled* and *unlabelled* data. Figure 6.4(a) plots the cumulative probability with respect to error distance. As can be seen, the proposed *online co-localization* benefits from the addi-

tional *unlabelled* data and increases the accuracy. Figure 6.4(b) and 6.4(c) show the localization error of the mobile device and access points by varying the number of labelled examples in a training subset which size is fixed to 500. Again, the proposed method performs relatively better than the baselines. By employing the *unlabelled* data, we save the calibration effort.



(a) Comparison of Accuracy

(b) WLAN Mobile Devices

(c) WLAN Access Points

(d) Average Running Time Comparison

Figure 6.4: Experimental Results over 10 Repetitions (Mean and Std.)

6.3.2 Speed Test

We compare the speed of Two-Phase *co-localization* method and the *online* version. Suppose that data come one by one sequentially. Once we get a new signal vector, the Two-Phase method adds it as a training example and rebuilds the whole model. In comparison, *Online co-localization* updates the estimation incrementally. Figure 6.4(d) shows the average running time for adding a new vector. The test is done in Matlab on a computer that has a 2.0GHz CPU. Experimental results show that we can greatly reduce the time for the model adaption in an online manner. For example, when the

training dataset size is incrementally enlarged to about 500, the Two-Phase method needs $1.2s$ to re-estimate everything while the online method spends no more than $0.1s$. The *online* method is more than ten times faster.

6.4 Summary

We have developed a manifold-based online and incremental approach to solve the problem of recovering the locations of both mobile devices and access points from radio signals that come in a stream manner. In our *online co-localization* framework, we exploit both labelled and unlabelled data from mobile devices and access points *online* when data come in stream. Experiments conducted in an 802.11 WLAN show that we can achieve high accuracy with less calibration effort as compared to several previous systems. Furthermore, our method can deal with data stream *online* relatively faster while compared to its Two-Phase counterpart. In the future, we would continue to study the environment in which access points may be removed, relocated and added for better coverage and link quality and how the algorithm can adapt the change dynamically.

CHAPTER 7

DIGITAL WALL: A SOLUTION FOR LOCATION-BASED DATA SHARING

With the proliferation of wireless and sensor techniques, data can be shared conveniently through the air. However, wireless communication is vulnerable since unauthorized machine may try to intrude a server without being physically connected. In this chapter, we wish to control the communication between a wireless client and the infrastructure based on the client location. Our idea is to implement a digital wall, which is a user-defined boundary so that access is allowed within the boundary and denied outside the boundary. To do this, we need to do accurate location estimation since the decision around the boundary line is critical. Furthermore, timing is also important since we expect that a connection can be cut shortly after the client is outside the digital wall. To do this, we need to refresh the estimated location frequently. The algorithm shall be power efficient to avoid computational overhead. In this chapter, we propose k-nearest-neighbor, a learning-based method to determine the location of a mobile client based on received signal strength values. We further use information gain as a feature selection criterion to reduce the estimation time. We study how the performance may be affected when the controlled area is confined by physical or virtual walls. Experimental results show that we can well distinguish whether a client device is located within an expected digital wall.

7.1 The Location-based Data Sharing Problem

As the pervasive environments become common, user activity context now turns out to be a sensitive issue for protecting user's privacy [53] and the network security [90]. The research community become increasingly interested in the context sensitive access control, which use different information, such as identity, location, for determining whether a user should be authorized for retrieving data from a network. Among those used information, location is a primary piece for context-aware computing, considering a usual case to confine data communication and sharing within an expected area. This suggests a location-based access control.

Digital wall can be treated as a special application of location based access control, which extends the notion of privacy provided by physical walls to the virtual realm. A *digital wall* is a user-define boundary so that access is allowed inside the boundary and denied outside.

Essentially, *Digital wall* is a virtual boundary as proposed in [48] for securing the users' privacy in wireless networks. However, in [48], work were focused on developing a policy language to formalize the semantics of access control using virtual wall. Other similar systems include *Digital Territory project* [9], *pawS* [53], *etc*. They all left the problem unsolved of actively detecting user to decide whether the user should be authorized for data sharing. That is exactly the issue we will address.

Accurate location estimation is needed for solving *digital wall* since the estimation shall be sensitive around the virtual boundary. In this chapter, we propose k nearest neighbor, a learning-based method for determining whether a client is inside or outside a customized space. More specially, our solution has two phase: an offline training phase and an online localization phase. In the training phase, we customize a virtual boundary and collect data inside and outside the boundary. In the localization phase, k-nearest-neighbor method is used to make a decision based on signal strength values detected in real time.

Meanwhile, timing is also important in *digital wall*, which shall respond quickly to the change of client locations. Especially, the connection shall be cut shortly after the client leaves the virtual space to protect privacy. Estimated location shall be updated frequently, which may involve too much computational overhead. Too solve this problem, we use information gain to selectively pick up a portion of access points for localization.

We study how the performance may be affected when the controlled area is confined by physical or virtual walls. We set up several testing cases and vary many locations on both sides of a digital wall. We also vary the number of access points used for localization based on information gain. Experimental results show that we can well distinguish whether a client device is located within an expected *digital wall*.

7.2 Methodology

7.2.1 System Architecture

We can abstract and classify different RSS-based localization models by what is carried in the object. The *receiver-oriented model* is shown in Figure 7.1, in which the object carries a receiver and receives the messages from different transmitters in the external infrastructure. The object can estimate its distances to these transmitters by measuring the RSS values on received messages. Based on at least three different distance estimations, the object computes its own location. The receiver-oriented model is a natural extension of the GPS mechanism without using timing information. It is mainly used in 802.11-based localization systems, in which the object is usually a person carrying a notebook computer or Personal Digital Assistant (PDA) with 802.11 card and the transmitters are the access points. Installing *digital wall* on client site is possible. The client itself can detect signal strengths, estimate its location and determine whether it shall response to the infrastructure and send out data packets. However, the infrastructure may not trust in the location the client claims. In such case, we need to verify the client location from the infrastructure without relying on the computation from the client device.

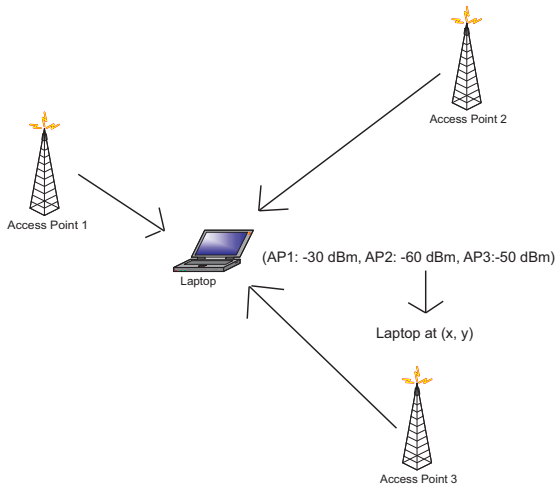


Figure 7.1: receiver-oriented model

The *transmitter-oriented model* exchanges the roles of transmitters and receivers in the receiver-oriented model as shown in Figure 7.2. After receiving the messages from the transmitter carried by the object, the receivers in the external infrastructure estimate the distances to the object. By collecting these distance estimations, the ex-

ternal infrastructure computes the location of the object without involving the object in computation. This model is best fit if the infrastructure want to track a client device without changing anything on that device and to verify the location the client claims.

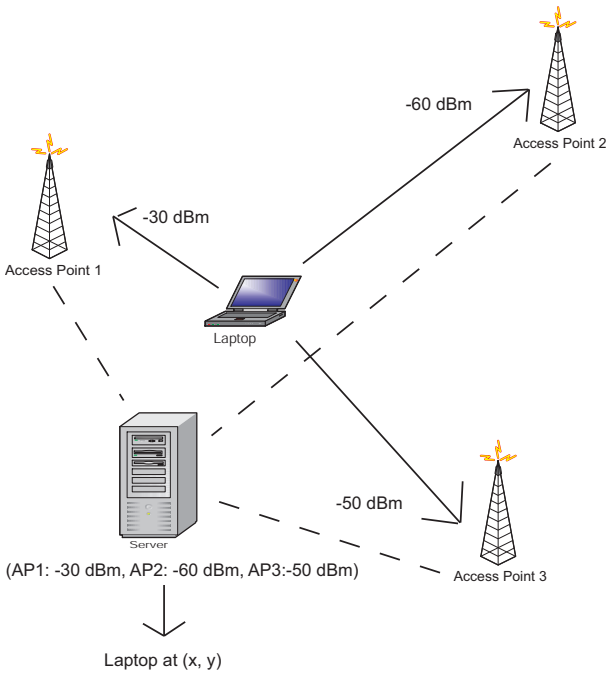


Figure 7.2: transmitter-oriented model

7.2.2 Problem Statement

We define the *digital wall* problem in a *receiver-oriented* model as follows. Assume that there are N access points fixed in an area that we are interested in. These access points periodically send out beacon signals. The locations of the access points are not necessarily known. There are one or more *mobile* devices of unknown locations. At time t, a *mobile* device can measure the RSS sent by the N access points by detecting their signals which gives a vector $\mathbf{s}_t = (s_{t1}, s_{t2}, \ldots, s_{tN})' \in \mathbb{R}^N$. In addition, we have collected l labeled training data which are signal-location pairs $\{(\mathbf{s}_{t_i}, \boldsymbol{\ell}_{t_i})\}_{i=1}^{l}$ at two locations. One location is inside a digital wall and the other is outside.

Our objective is to determine the location $\boldsymbol{\ell}'_t \in \{inside, outside\}$ of a *mobile* node based on the signal vector $\mathbf{s}_t$ measured at time t.

7.2.3 Power-efficient Classification

Our solution to the *digital wall* has two phases : an offline training phase and an online localization phase. In the offline phase, we customize the digital wall, collect data and select access points based on a feature selection criterion. In the online phase, we apply k nearest neighbor method to determine whether the current location of a mobile device is inside or outside the digital wall.

• Offline Training Phase

1. Customize the digital wall in an area we are interested in. It may be physical walls around a room or virtual ones inside a room.

2. Collect l labelled signal-location pairs at two locations. One location is inside the digital wall and the other is outside. Denote the dataset as $\{(\mathbf{s}_{t_i}, \boldsymbol{\ell}_{t_i})\}_{i=1}^{l}$.

3. Compute and rank Information Gain for the N access points. Pick up the top M discriminative access points which Information Gains are higher than some threshold θ. Denote the subset of M access points as $\mathfrak{F}$.

The discriminative power of an access point is measured by the Information Gain when its value is known. Specifically, it is calculated as the reduction in entropy as:

$$InfoGain(AP_i) = H(G) - H(G|AP_i)$$

where $H(G)$ is the entropy of the two sampling locations and $H(G|AP_i)$ is the conditional entropy when the value of AP_i is known.

In such a way, we can pick up the topmost useful access points and avoid computational overhead.

• Online Localization Phase

1. At time t, the *mobile* device collects a signal vector $\tilde{\mathbf{s}}_t$. For those access points that are far away, which signals are too weak to detect, we fill in a small value, e.g. -100dBm.

2. Pick up the signal values that are in the subset of selected access points $\mathfrak{F}$.

3. Apply k nearest neighbor method and determine whether the mobile device is inside the digital wall or outside.

7.3 Experiments

7.3.1 Experimental Setup

We perform our experiments in the laboratory area of the Computer Science and Engineering Department in the Academic Building of the Hong Kong University of Science and Technology (HKUST) as shown in Figure 7.3. The mobile device used to collect training and test data is a T40 IBM laptop. There are totally sixty different access points detected in the laboratory area. We evaluate our digital wall solution in two different scenarios: *physical wall* and *virtual wall*. For physical wall scenario, we test our system in four offices. For each office, we use the laptop to collect RSS records in a fixed location in-office and another fixed location out-of-office as training data. For testing, we collect RSS records at variant locations in-office and out-of-office as test data. Our goal is to predict whether the laptop is in office or out of office. For virtual wall scenario, we test our system around a council board in a office. We image that there were a virtual wall in the middle of the council board, as shown in Figure 7.4. We collect RSS records in a fixed location in either side of the boundary as training data, separately. For testing, we collect several RSS records in various locations in each side of the boundary (as shown in Figure 7.4, we collect test data in location a, b, ...,n). Our goal is to predict whether the laptop is in the side above the boundary or below the boundary. In order to measure the performance, we use location prediction accuracy as the evaluation criterion.

7.3.2 Experimental Results

Experiments on Physical Wall

The first experiments evaluate the performance of our system in the physical wall scenario. In each office region, our system first apply *CaDet* to select important access points, then based on the selected access points, our system use k-nearest-neighbor algorithm to predict locations of the laptop.

Figure 7.5 shows the average performance of our system in different distances from

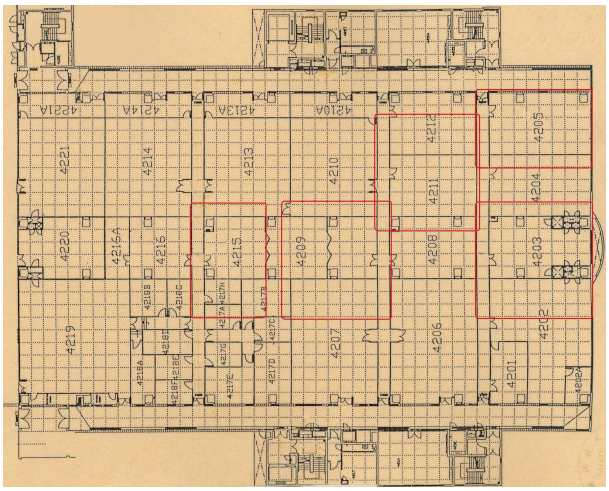


Figure 7.3: Test Bed in the laboratory area of the Computer Science and Engineering Department

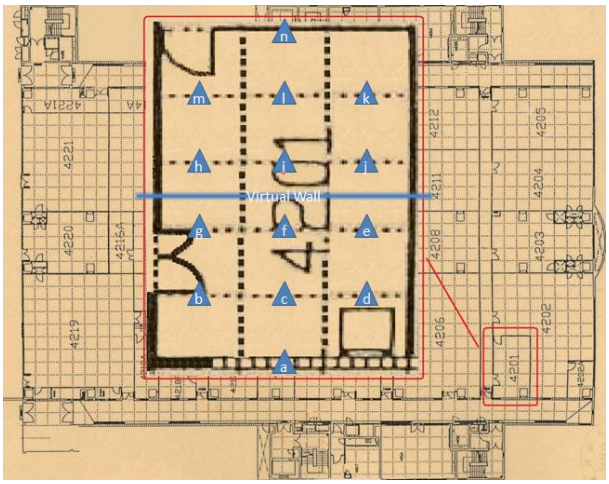


Figure 7.4: Virtual boundary in an office

the laptop to the door of the office. We can see that, our system can get high accuracy when the distance from the laptop to the door is equal to or lager than 2 meters. That means when a user walks into an office and is not nearby the door, our system can support wireless services correctly.

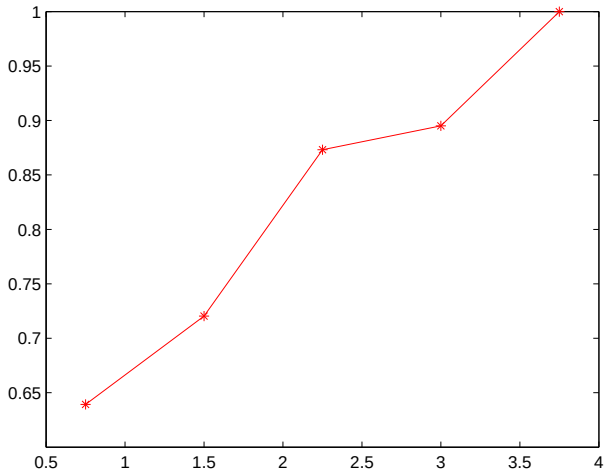


Figure 7.5: Location estimation accuracy versus the distance from the laptop to the door of an office

Figure 7.6 shows the effort of the number of access points on average performance of different office regions, where we fix the distance between the laptop and the door being 2 meters. In order to reduce the computation cost used to locate the mobile device, we expect to select as few as access points to achieve a high accuracy. It can be seen from the figure that when the number of access points equals to eleven, our system can achieve the best accuracy of 87%.

Experiments on Virtual Wall

In this section, we evaluate the performance of our system in the virtual wall scenario. We first apply *CaDet* to selecte access points, then use k-nearest-neighbor algorithm to predict locations. Figure 7.7 shows the performance of the system in different the distance from the laptop to the virtual boundary. It can be seen that the performance of the system is not high when the laptop is nearby the boundary. However, when the distance from the laptop to the boundary is larger than or equal to 2 meters, our system can get around 85% accuracy.

In the above experiment, we use the first five access points selected by *CaDet*. The

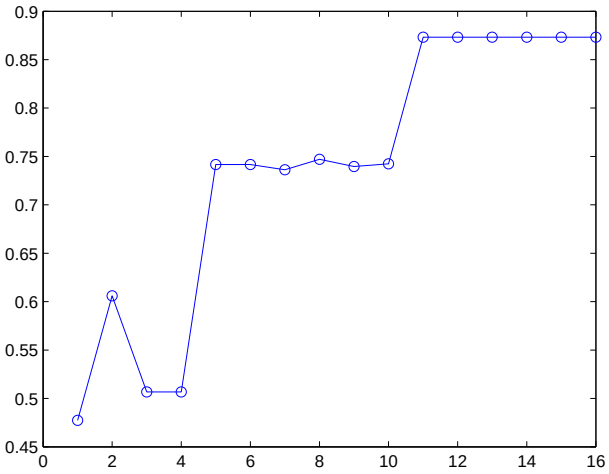


Figure 7.6: The number of APs versus average accuracy in physical wall

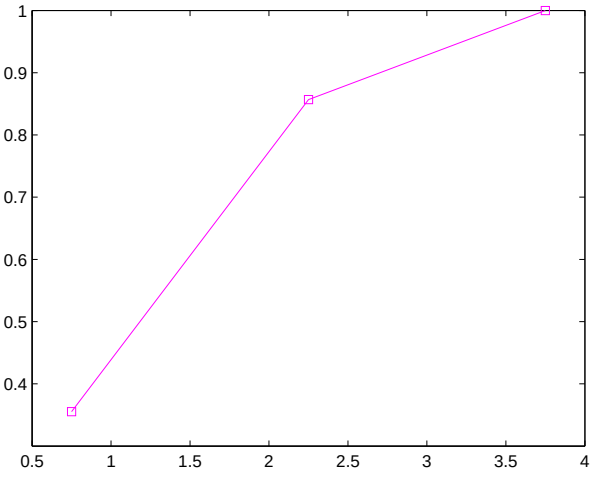


Figure 7.7: Location estimation accuracy versus the distance from the laptop to the virtual boundary

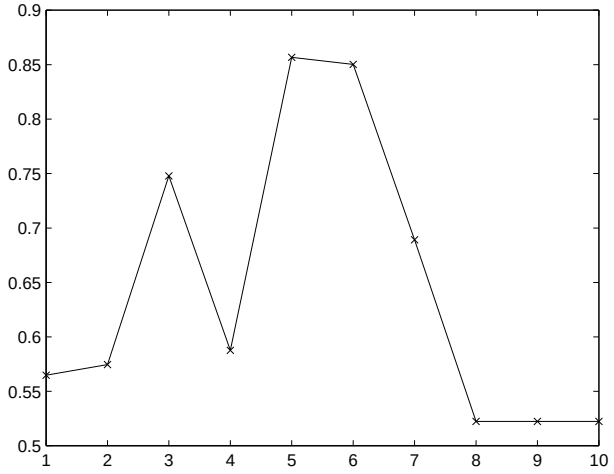


Figure 7.8: The number of APs versus accuracy in virtual wall

effort of the number of access points on performance is shown in Figure 7.8, where we fixed the distance from the laptop to the boundary being around 2 meters. We can see that when selecting access points not only can reduce the computation cost but also improve accuracy. The performance using five access points is much larger than that using all access points.

7.4 Summary

In this chapter, we describe *digital wall*, a power-efficient solution for location-based data sharing. To apply this system, we first have to define a digital wall in an interesting area. In the offline training phase, we collect training data at two locations. One is inside the digital wall and the other outside. After that, Information Gain is used to select the most useful access points and reduce computational overhead. In the online localization phase, we apply k nearest neighbor to determine a mobile client is inside a digital wall or not. We vary many parameters to verify the effectiveness of the algorithm. We use different rooms, the wall of which may be physical or virtual. We vary the number of access points and the sampling locations. Experimental results show that we can well distinguish whether a client device is located within an expected *digital wall*.

CHAPTER 8

PANE: A WIFI TOOL FOR POSITIONING AND NAVIGATION EXPERIMENTS

This chapter describes a WiFi signal collecting and location labelling system for positioning and navigation experiments. To evaluate the performance of a tracking system using received signal strengths, we need to collect large amount of calibrated (labelled) data. Meanwhile, to deploy a tracking system, we also need sufficient labelled data. Labels can be automatically given in an outdoor environment by employing a GPS. However, we have to manually mark down the locations for all detected signal vectors in an indoor environment since a GPS may not work. This is a time consuming and error-prone process. In this chapter, we introduce a convenient tool for data collection and analysis, which is developed in a Windows XP platform. It supports general wireless cards and has convenient functions for map building, signal collecting, location labelling and data visualization.

8.1 System Overview

To test the performance of a tracking system at different time and environment, we need to frequently collect signal strengths and their associative locations. A convenient tool is needed to relief our burden and reduce potential error.

In this chapter, we introduce a WiFi signal collecting and labelling tool for positioning and navigation experiments. The main graphical user interface (GUI) is shown in Figure 8.1. The main window can be split into three parts:

- **Main Control Area** The buttons, scroll bars and check boxes on the upper-left part of the window are used for controlling wireless device, map building, data collection, localization and signal visualization.
- **Signal Indication Area** The list view on the bottom-left part of the window shows the information of detected access points, including media access control (MAC) address, service se identifier (SSID), received signal strength, etc.

- **Interative Mapping Area** The frame on the right is the user interaction area. It shows the multi-layer maps, the user trajectory and the access point locations that we mark down in real-time.

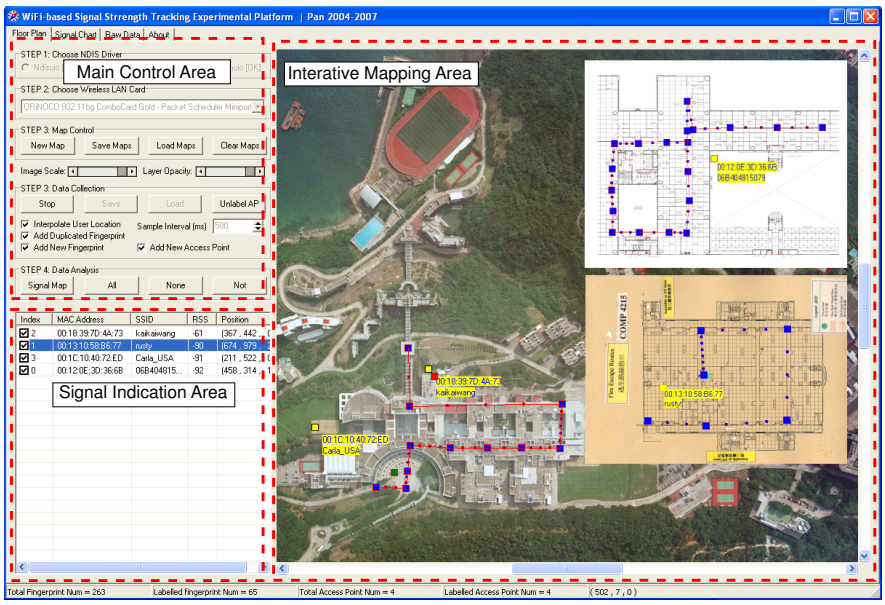


Figure 8.1: The Experimental Platform for WiFi Signal-Strength-based Tracking

More specifically, our WiFi experimental platform has the following features:

- **Support of general wireless devices**. Our platform is developed in Windows XP. It uses Network Driver Interface Specification User Mode Input / Output (NDISUIO) to detect beacon frames from different access points and measure their signal strengths. NDISUIO is initially used by Wireless Zero Configuration Service (WZCSVC). Thus, if a wireless card can run WZCSV well, it is mostly likely to be supported by our platform.

- **Multi-layer map building tool**. We can create, drag, rescale, remove, load and save maps conveniently.

- **Label access points and client devices**. By supporting a map, we can mark down the current locations of client devices and the readings of signal strengths. We can also selectively mark down the locations of access points on the map. We visualize all labelled access points and client devices. Each map has its own local coordinate system.

- **Received-signal-strength based Tracking**. We implement two tracking algorithms based on RSS readings. One is a simple propagation model that relies on the locations of access points. Another is a K-Nearest-Neighbor method which uses the calibrated data from client devices.

- **Signal coverage visualization of access points**. After we collect enough labelled data in the experimental test-bed, we can visualize the signal coverage easily.

- **Export data to Matlab for further analysis**. A matlab file is automatically generated for further analysis when we save the collected data.

Although the system is developed in Windows XP, the kernel codes are mostly ready for migrating to other operating systems since the platform-dependent codes are well separated from the kernel modules.

8.2 Signal Coverage Visualization

We can do some simple signal coverage visualization. First we have to stop the data collection before any analysis. Then we can select and un-select access points for calculating signal level. Alternative, we can click “All”, “None” and “Not” to pick up access points for analysis. By clicking “Signal Map”, we can obtain a rough signal distribution as shown in Figure 8.2.

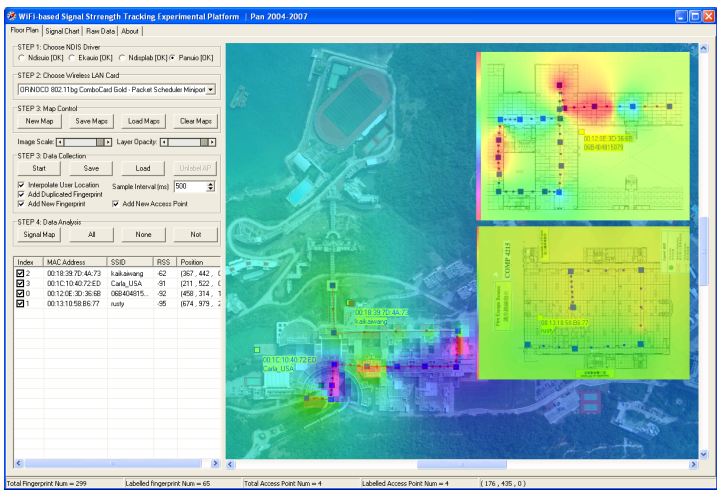
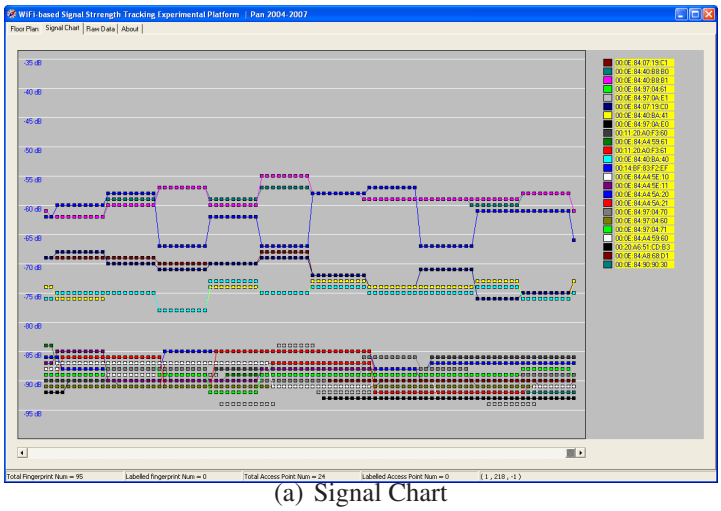


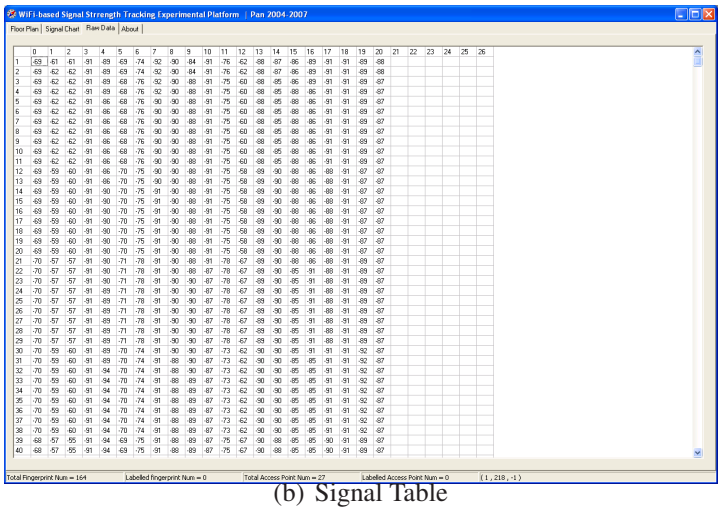
Figure 8.2: Visualization of Signal Coverage

8.3 Signal Chart and Table

This tool can show a real-time and historical view of received signal strength values as shown in Figure 8.3(a). The horizontal axis is the time line. The vertical axis denotes different signal levels. The signal strength of multiple access points are shown in the figure. Similarly, the signal values can be listed in a table as shown in Figure 8.3(b).



(a) Signal Chart



(b) Signal Table

Figure 8.3: Visualization of real-time and historical signal strength values

CHAPTER 9

VEST: A VISION-BASED EVALUATION SYSTEM FOR MOBILE TRACKING

This chapter describes VEST, a Vision-Based Evaluation System for Mobile Tracking done in most of our sensor-network-based experiments. As we know, to test the accuracy of different tracking models, it is necessary to mark down the ground truth locations for the associated sensor readings, e.g., received signal strengths (RSS). Ground truth can be obtained by manually labelling or through the assistance of special hardware. It is easy to mark down the locations of a static object by referring to landmark points such as corners, doors, turns and texture on floor and ceiling. However, it is not straightforward to mark down the real-time locations of a moving person since his/her position may change from time to time. In this chapter, we introduce a vision-based tracking system that uses off-the-shelf USB video cameras to form a video array to track the motion of sensor nodes in an experimental environment. The estimated trajectory is accurate enough to be treated as ground locations.

9.1 The Problem of Obtaining Ground Truth Location

To build an accurate tracking system is not an easy job. To evaluate the performance of the tracking model may even harder. Ground truth locations must be obtained and compared to the estimated ones. If the target objects are fixed, we could manually mark down their positions one by one. However, it is not that easy if the objects are moving from time to time. In an outdoor and suburban area, GPS may be employed to record the real-time trajectory. However, in urban and indoor environment, GPS does not work well, especially if we require that the ground truth shall have small error, e.g., within some centimeter. Although laser scanners are possible, they are expensive and not ubiquitous. We propose a vision-based tracking system that can track an object accurately when it is carefully calibrated in an experimental area. The goal of the vision system is not to replace the RSS-based tracking system but to support ground truth for evaluation. Vision-based tracking may be accurate. However, they demand

high speed computers and consume more power. Cameras also need to be carefully calibrated before they can track an object cooperatively. Instead, RSS-based tracking is simpler and it is not sensitive to the environment change such as daylight.

9.2 The System Architecture

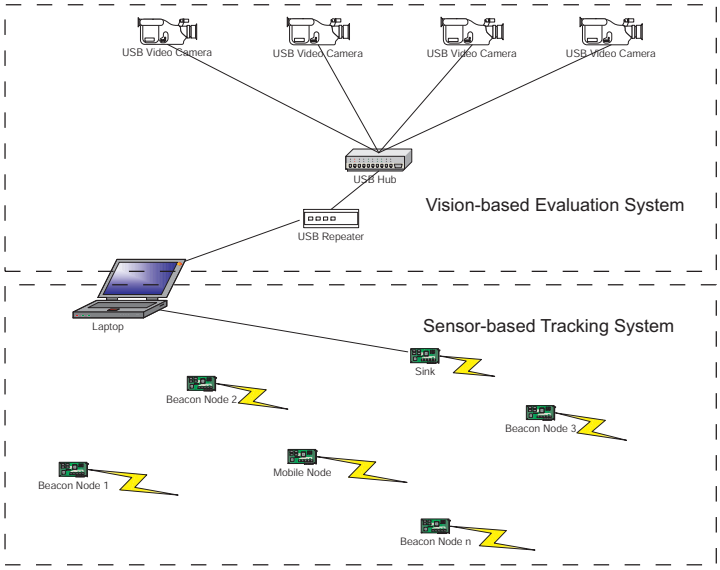


Figure 9.1: The Architecture of Tracking and Evaluation System

As shown in Figure 9.1, the whole tracking and evaluation system can be split into two parts: the sensor-based tracking system and the vision-based evaluation system. All data are streamed to the central computer for synchronization and further analysis. Typically, the vision-based evaluation system has:

- **USB**[1] **Video Camera**. We use four cameras deploy at different locations to capture the target area. We choose V-Gear TalkCam Pro manufactured by Asiamajor Inc. It has a $1/4''$ CMOS sensor, adjustable focus and exposure control with a resolution 640×480 pixels. The stand and clip can be easily attached to the ceiling of our experimental room. The cable length is about $1.5m$. We extend the cable by $1.0m$ with a common USB extension cable. A total of four cameras are used for our experiments. Note that a longer extension cable may cause problem. The camera may not be correctly powered, recognized or the image data may not be transmitted fluently.

[1]Universal Serial Bus (USB) is a serial bus standard to interface devices.

- **USB Hub**. All four cameras are connected to a 4-port Polar^{TM} USB hub. The hub shall be powered to drive all the cameras.

- **Boosted USB Extension Cable**. We connect the USB hub to a central computer. To make the evaluation system a bit more easy to deploy, we use a long USB extension cable to connect them, e.g., a $5.0m$ cable. General extension cable is not applicable since signal may be corrupted. Instead, we shall use high-quality USB extension cable such as Comsol^{TM} USB 2.0 extension cable. It is essentially a $5.0m$ USB repeater with chip inside. A common 5.0m cable line would not work.

- **Central Computer**. The USB extension cable is connected to an IBM T60 laptop. The computer is power enough to processing videos at $20fps$ each from all four cameras simultaneously by setting the resolution to 320×240 pixels.

The sensor-based tracking system has:

- **Wireless Sensor Networks**. We use CrossBow MICA2 and MICA2Dot to construct a wireless sensor network. We program these sensor nodes to broadcast and detect beacon frames periodically so that they could measure the Radio Signal Strength (RSS) of each other. By combining the RSS from different nodes we could estimate locations of these nodes.

- **Sink Node and Interface Board**. A sink node is composed of a sensor node such as MICA2 or MICA2Dot, and an interface board MIB510. The sensor node listens to all the communication packets and transmits them to the central computer through the interface board.

- **USB-to-RS232 cable**. This component is optional. Our laptop computer IBM T60 does not have a COM port to communicate with the interface board MIB510. We need a USB-to-RS232 converter to link the laptop and the interface board together. We also require high quality converter since a general converter may not work correctly. The model we use is a Aten^{TM} UC-232A USB to Serial (RS-232) Converter.

- **Central Computer**. Collect sensor communication packets and extract RSS vectors. The signals are synchronized with the ground truth locations by analyzing video streams.

The toolkit for video tracking is shown in Figure 9.2.

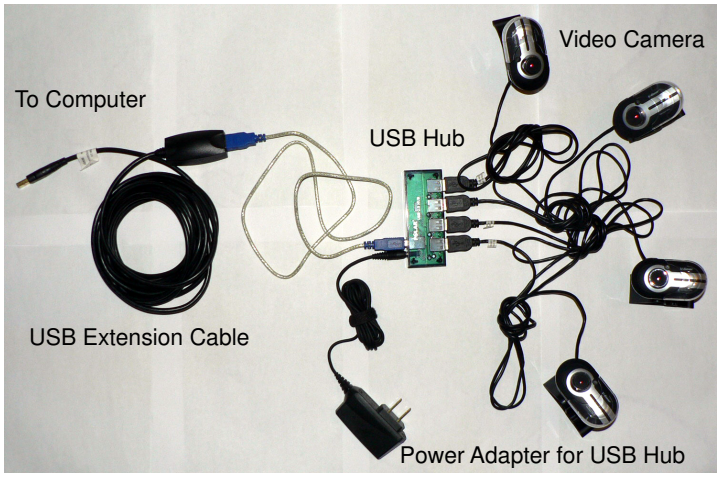


Figure 9.2: The Toolkit for Visual Tracking

9.3 Camera Calibration

We use landmarks [45] on floor to calibrate the position and the direction of a camera. More specifically, we use a rectangle. As shown in Figure 9.3, $ABCD$ is a rectangle on the floor. $AB//CD$ and $AD//BC$. All four angles are $90°$. A camera is located at point E and capture an image in rectangle $JKLM$. The projected plane of $ABCD$ is $A'B'C'D'$ in the captured image. Lines $A'D'$ and $B'C'$ intersect at H, which is a vanishing point. Similarly, G is also a vanishing point.

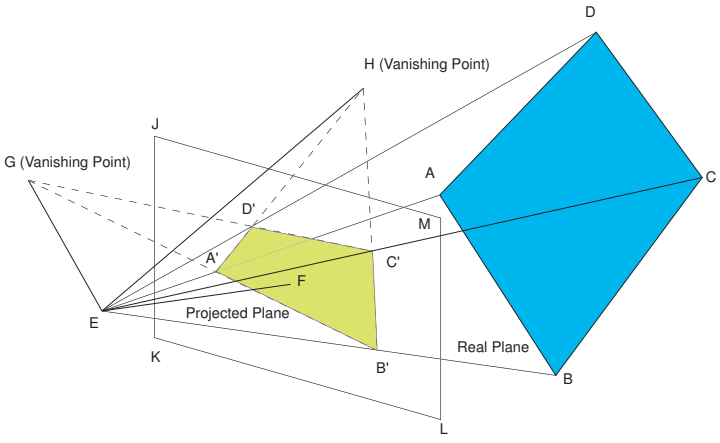


Figure 9.3: Camera Calibration based on a Rectangle Landmark

We will show that lines EG and EH are both parallel to the real plane $ABCD$. First, it is easy to see that lines EA' and ED' intersect lines EA and ED at points

A and D respectively. H is a vanishing point. Line EH intersects line AD at an infinite position and thus is parallel to line AD and plane $ABCD$. Similarly, line EG is parallel to line BA and plane $ABCD$. The plane determined by points E, G and H is parallel to plane $ABCD$. The direction of plane $ABCD$ is the cross product of vector $\vec{EH}$ and vector $\vec{EG}$. If we use formula $ax + by + cz + d = 0$ to represent a plane in a 3-dimensional space, we have fixed a, b and c so far. It is straightforward to solve d since we know the length of line segment AB and BC. In such a way, we can compute the coordinates of A, B, C and D from the projected plane, which in turns determine the position and the direction of a camera given a real plane.

Figure 9.4 shows the graphical user interface (GUI) of our camera calibration process. In the window, we first pick up a camera for calibration. The real-time video of the camera is shown on the left of the window. We need to set up the size of a landmark rectangle, e.g., $|AB|$ and $|BC|$. We may also have to set up a ratio that can convert pixels to physical length in centimeter. After that, we can manually drag and drop points A, B, C and D in the video, which positions shall match the landmark rectangle.

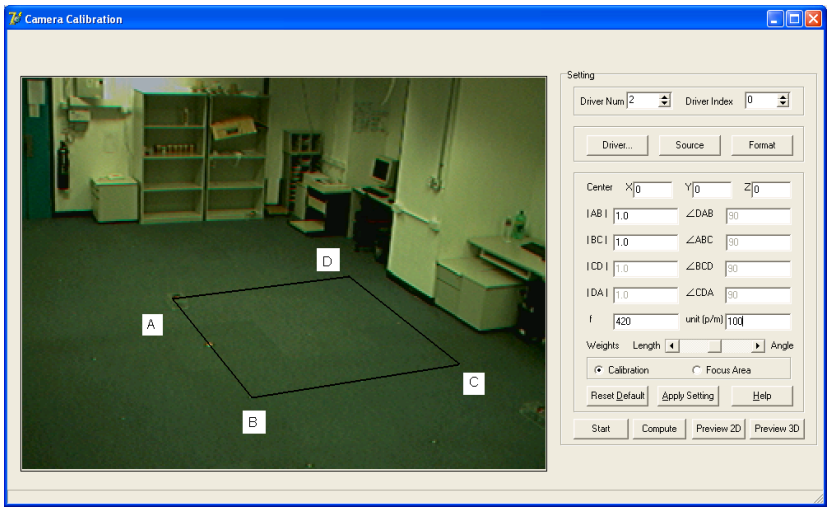


Figure 9.4: GUI for Camera Calibration

9.4 Visual Tracking

Video streams from different cameras are then synchronized with each other as well as the packet flows from the wireless sensor networks. They would be analyzed frame by frame through an image segmentation [17] and a feature extraction (lightness, color,

size and shape) process. Locations of mobile nodes are detected first on projected planes (video frames) and transformed to coordinates on the real plane (the floor). Our USB2.0 cameras could record videos at 20fps with resolution 320*240 pixels when all four are worked concurrently on a computer with a 3.2GHz CPU. Since the floor has a pure color and the mobile nodes have simple shape and fixed size, we could track the location of mobile nodes within error distance in some centimeter, which is treated as the ground truth.

Figure 9.5 shows the console of the video tracking system. On the left of the window, there are videos that come from four cameras, each cover a portion of the test-bed. The options on the right are used to adjust the parameters for visual tracking. Typical parameters are color, size and shape. The crossing denotes the recognized moving object in the video frames.

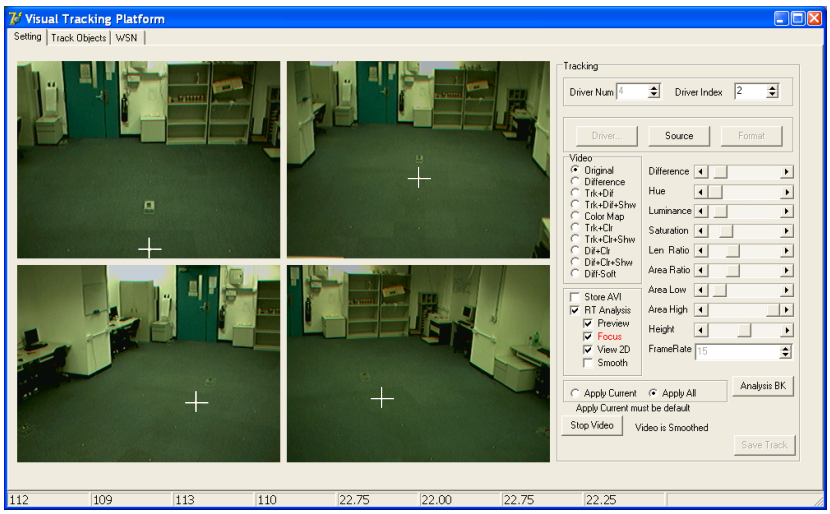


Figure 9.5: GUI for Vision-based Tracking

9.5 Post Processing

Recognizing individual objects in each video camera is not enough. We have to check whether two recognized positions refer to the same object or not in consecutive frames. Furthermore, to enable three-dimensional tracking, an object may be captured by more than one camera. Tracking from multiple cameras is a complex task. We design an heuristic algorithm to fuse the trajectories from multiple cameras collaboratively, which forms the final three-dimensional trace in the test-bed.

Figure 9.6 demonstrates the recognized trajectories within each individual camera. A red object is moved in the air by a person. It is captured by all four cameras from different perspective. By computing line-line intersection, we can export the three-dimensional location sequence.

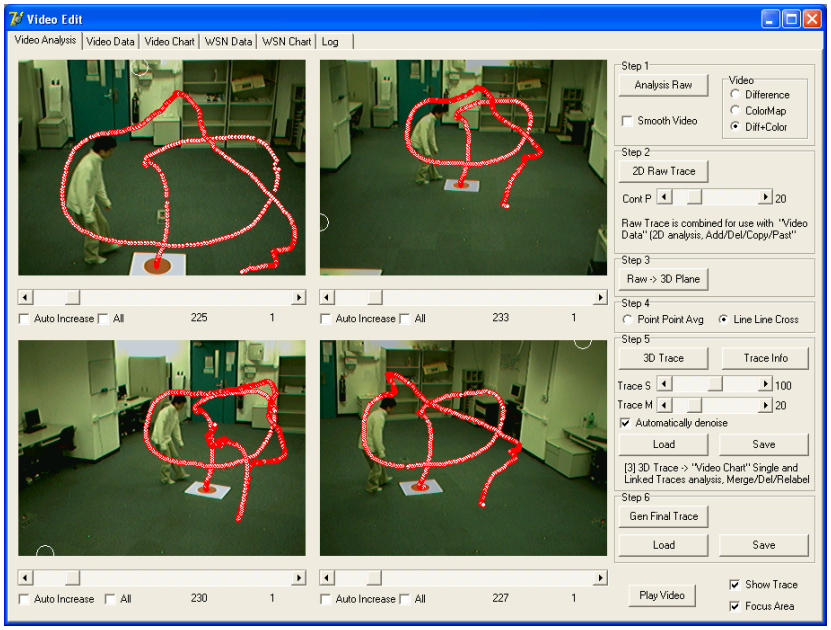


Figure 9.6: GUI for Video Postprocessing

9.6 Two-Dimensional Tracking Test-bed

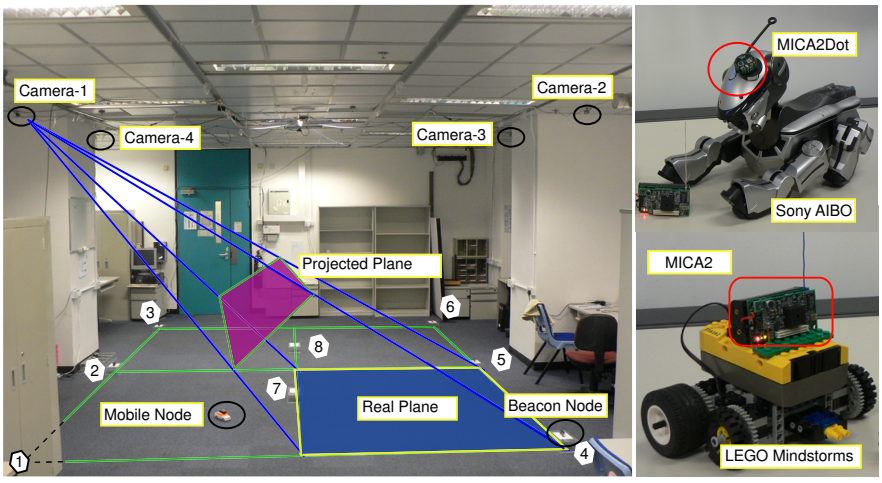


Figure 9.7: WSN 2D Test-Bed

Our two-dimensional tracking experiment is performed in the Pervasive Computing Laboratory (Figure 9.7), Department of Computer Science. The room is large enough

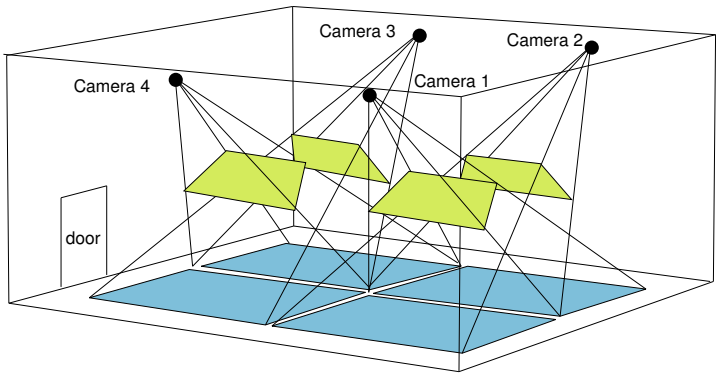


Figure 9.8: Camera Calibration for Two-Dimensional Tracking

for us to set up an experimental test-bed of 5.0 meter by 4.0 meter. In Figure 9.7, $|P_1P_3| = |P_4P_6| = 5.0m$ and $|P_1P_4| = |P_3P_6| = 4.0m$. The sensor network and the video cameras are configurated as below.

- *Sensor-based Tracking System*. Figure 9.7 shows that there are eight *static* sensor nodes deployed on the test-bed: six around the rectangular boundary denoted as $1, 2, \ldots, 6$; another two around the central area denoted as $7, 8$. There is one *mobile* node, which is attached on top of a toy car. We config all the nine nodes so that each one could measure the RSS from the remain eight nodes in every 0.5s. We try different kinds of robots that could run freely around the floor such as Sony AIBO dogs, LEGO Mindstorms and off-the-shelf toy cars. Figure 9.7 shows that a sensor node is attached on top of a toy car which could be remotely controlled by radio at the speed of 0.4 m/s.

- *Vision-based Evaluation System* is used to record experiments for supporting location information (ground truth) of mobile robots. Figure 9.7 shows that there are four cameras deployed at four corners of the ceiling (2.8m height) respectively. Each camera monitors at least one-fourth part of the test-bed. The central area is covered by all four cameras. After calibration, we can get the position and the direction of all cameras as shown in Figure 9.8. An example of the ground truth and the estimated trajectories of about 30 seconds is illustrated in Figure 9.9.

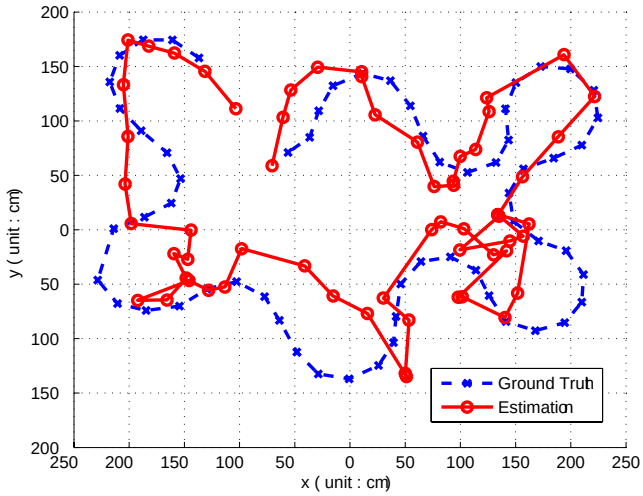


Figure 9.9: Ground Truth and Estimated Trajectories

9.7 Three-Dimensional Tracking Test-bed

The three-dimensional tracking experiment is again done in the Pervasive Computing Laboratory as shown in Figure 9.10. We use the space efficiently so that the whole test-bed fills up a cubic space of $6.0m \times 6.0m \times 2.0m$. In the test-bed, we have ten static nodes that send out beacon signals. Five of them are deployed on the floor and the rest on the ceiling. There is one more node that moves freely around the environment for tracking experiments. In Figure 9.10, the mobile node is located at the intersection point E of lines CD and AB estimated from two different cameras.

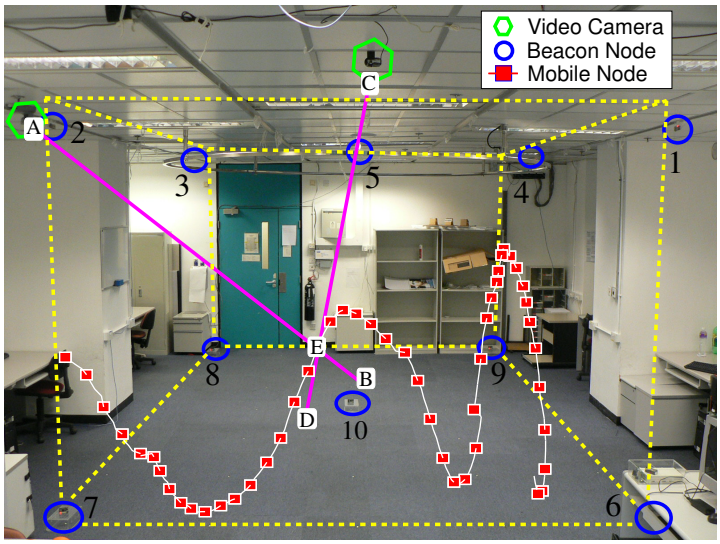


Figure 9.10: WSN 3D Test-bed

9.8 Evaluation of the Vision-based System

We use the vision-based system to evaluate the accuracy of sensor-based system. However, how can we evaluate the accuracy of the vision-based system? We haven't finished a thorough study on this issue. Instead, we pick up dozens of points from the test-bed and estimate the accuracy. By further considering the latency of video transmission, we conclude that the error is within some centimeter. The error is much smaller than that from a sensor network in general. We treat it as the ground truth.

9.9 Summary

In this chapter, we introduce VEST, a vision-based evaluation system for tracking experiments. Evaluation is a process that can not be missed in most system designs. Obtaining ground truth is critical or we may have problems in quantizing the performance of different tracking models. To evaluate the performance of localizing static nodes, we can manually mark down their true positions one by one. However, to test the performance of tracking a mobile node is much more difficult because we have to track the changing location of a mobile node in real time. It can not be manually done.

Building an evaluation is challenging since it usually calls for a much higher accuracy than a general tracking system can do. We solve the problem by deploying an array of cameras in the interesting area. We build a vision-based tracking system that can track the locations of a mobile node in real time with reasonable accuracy by carefully calibrating all cameras. The vision-based evaluation system works well in an experimental area. It is not used to replace the sensor-based tracking system since it involves very high computational overhead.

CHAPTER 10

CONCLUSION AND FUTURE WORK

The theme of this thesis is to study the problem of localization and tracking from noisy and nonlinear radio signal strength. In this chapter, we conclude our work and point out possible directions for future work.

10.1 Conclusion

This thesis reviewed several learning-based approaches for localization and tracking in wireless and sensor networks, and discussed three main problems. We first reviewed the wireless and sensor network environments and related works. We then discussed how to improve the accuracy of localization using the LeKCCA [67] method, a kernel-based supervised-learning algorithm for locating client devices in wireless local area networks. We then discussed how to reduce the calibration effort using the LeMan [70], which is a manifold-based semi-supervised method for tracking mobile nodes in Wireless Sensor Networks. We also proposed co-localization [69], a general and flexible framework for recovering the locations of both mobile devices and access points from radio signals by exploiting both labeled and unlabeled data from mobile devices and access points. We further extend the framework into an online method that can cope with calibration data that come sequentially [71]. The characteristics of all these models are shown as in Table 10.1.

Table 10.1: Characteristics of Our Models

Method / Calibration	Access Point		Mobile Device		Trace	Online
	Unlabeled	Labeled	Unlabeled	Labeled		
LeKCCA	✓			✓		
LeMan	✓		✓	✓		
Co-Localization	✓	✓	✓	✓		
Online Co-Localization	✓	✓	✓	✓	✓	✓

We have published the following papers (in chronological order):

Jeffrey J. Pan, Qiang Yang, Sinno J. Pan. Online Co-Localization in Indoor Wireless Networks by Dimension Reduction. In Proceedings of the *Twenty-Second National Conference on Artificial Intelligence* (AAAI-07), Vancouver, Canada, 2007.

Jeffrey J. Pan, Qiang Yang. Co-Localization from Labeled and Unlabeled Data Using Graph Laplacian. In Proceedings of *Twentieth International Joint Conference on Artificial Intelligence* (IJCAI-07), Hyderabad, India, January, 2007.

Jeffrey J. Pan, Qiang Yang, Hong Chang, Dit-Yan Yeung. A Manifold Regularization Approach to Calibration Reduction for Sensor-Network Based Tracking. In Proceedings of the *Twenty-First National Conference on Artificial Intelligence* (AAAI-06), Boston, 2006.

Jeffrey J. Pan, James T. Kwok, Qiang Yang, Yiqiang Chen. Multidimensional Vector Regression for Accurate and Low-Cost Location Estimation in Pervasive Computing. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*. 2006.

Jeffrey J. Pan, James T. Kwok, Qiang Yang, Yiqiang Chen. Accurate and low-cost location estimation using kernels. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence* (IJCAI-05), pages 1366-1371, Edinburgh, United Kingdom, July 2005.

10.2 Limitations and Future Works

We will continue to explore possible research work along the following directions.

- **Sample Selection** We have proposed online co-localization, a framework that can encode new data and delete old data online and incrementally. As time elapses, the model may grow bigger and bigger if we do not selectively pick up and delete data for model update. In practice, the indoor environment is dynamic, obsoleted access points may be removed; new ones may be deployed and existing ones may be relocated. New calibration data comes in and old go out. The environment would be different time to time. How to intelligently select new data and remove old data is still an open problem.

- **Encode User Model** Popular tracking methods are Kalman Filter, Particle Filter and Gaussian Process, which could encode user models as a prior to improve accuracy. We may consider this sequence information online to smooth the predicted trajectory.

- **Encode Action Label** Sometimes we can collect acceleration data. With these data, we can infer the activity of a user, say, whether he or she is walking, making

a left turn or right turn. By recognizing these actions, we can further improve the localization performance in an indoor environment.

REFERENCES

[1] K. P. A. Hatami, "A comparative performance evaluation of rss-based positioning algorithms used in wlan networks," vol. 4, pp. 2331 – 2337, March 2005.

[2] K. T. Abou-Moustafa, M. Cheriet, and C. Y. Suen, "Classification of time-series data using a generative or discriminative hybrid," in *Proceedings of the 9th International Workshop on Frontiers in Handwriting Recognition*, (Tokyo, Japan), October 2004.

[3] C. R. B. Li, A. Dempster and J. Barnes, "Hybrid method for localization using wlan," in *Proceedings of Spatial Sciences Institute Biennial Conference*, 2005.

[4] P. Bahl, A. Balachandran, and V. Padmanabhan, "Enhancements to the RADAR user location and tracking system," technical report, Microsoft Research, February 2000.

[5] P. Bahl and V. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in *Proceedings of the Conference on Computer Communications*, vol. 2, (Tel Aviv, Israel), pp. 775–784, March 2000.

[6] M. A. Batalin, G. S. Sukhatme, and M. Hattig, "Mobile robot navigation using a sensor network," in *IEEE International Conference on Robotics and Automation*, (New Orleans, LA, USA), pp. 636–642, April 2004.

[7] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[8] M. Belkin, P. Niyogi, and V. Sindhwani, "On manifold regularization," in *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pp. 17–24, Society for Artificial Intelligence and Statistics, January 2005.

[9] L. Beslay and H. Hakala, "Digital territory: Bubbles," *European Visions for the Knowledge Age*, 2005.

[10] E. Bhasker, S. Brown, and W. Griswold, "Employing user feedback for fast, accurate, low-maintenance geolocationing," in *IEEE International Conference on Pervasive Computing and Communications*, (Orlando, FL, USA), pp. 111–120, March 2004.

[11] R. Biswas and S. Thrun, "A distributed approach to passive localization for sensor networks," in *Proceedings of the Twentieth National Conference on Artificial Intelligence*, (Pittsburgh, PA, USA), pp. 1248–1253, July 2005.

[12] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the eleventh annual conference on Computational learning theory*, (New York, NY, USA), pp. 92–100, ACM Press, 1998.

[13] C. Bregler, "Learning and recognizing human dynamics in video sequences," in *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition*, (San Juan, Puerto Rico), pp. 568–575, 1997.

[14] M. Brunato and R. Battiti, "Statistical learning theory for location fingerprinting in wireless LANs," *Computer Networks*, vol. 47, no. 6, pp. 825–845, 2005.

[15] X. Chai and Q. Yang, "Multiple goal recognition from low-level signals," in *Proceedings of the Twentieth National Conference on Artificial Intelligence*, (Pittsburgh, PA, USA), pp. 3–8, July 2005.

[16] X. Chai and Q. Yang, "Reducing calibration effort for location estimation using unlabeled samples," in *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications*, (Kauai Island, HI, USA), pp. 95–104, 2005.

[17] H. Chang and D. Yeung, "Robust path-based spectral clustering with application to image segmentation," in *Proceedings of the Tenth IEEE International Conference on Computer Vision*, (Beijing, China), pp. 278–285, Octorber 2005.

[18] Y. Chen, Q. Yang, J. Yin, and X. Chai, "Power-efficient access-point selection for indoor location estimation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, pp. 877–888, July 2006.

[19] F. Chung, *Spectral Graph Theory*. American Mathematical Society, 1997.

[20] A. Civilis, C. S. Jensen, and S. Pakalnis, "Techniques for efficient road-network-based tracking of moving objects," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, pp. 698–712, 2005.

[21] L. F. M. de Moraes and B. A. A. Nunes, "Calibration-free wlan location system based on dynamic mapping of signal strength," in *Proceedings of the international workshop on Mobility management and wireless access*, (New York, NY, USA), pp. 92–99, ACM Press, 2006.

[22] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society of Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[23] I. S. Dhillon, "Co-clustering documents and words using bipartite spectral graph partitioning," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 269–274, ACM Press, 2001.

[24] P. Enge and P. Misra, "Special issue on GPS: The global positioning system," *Proceedings of the IEEE*, pp. 3–172, January 1999.

[25] B. Ferris, D. Fox, and N. Lawrence, "Wifi-slam using gaussian process latent variable models," in *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, (Hyderabad, India), pp. 2480–2485, 2007.

[26] B. Ferris, D. Hahnel, and D. Fox, "Gaussian processes for signal strength-based location estimation," in *Proceedings of Robotics: Science and Systems*, (Philadelphia, Pennsylvania, USA), August 2006.

[27] D. Fox, J. Hightower, L. Liao, and D. Schulz, "Bayesian filtering for location estimation," *IEEE Pervasive Computing*, vol. 2, no. 3, pp. 24–33, 2003.

[28] S. Funiak, C. Guestrin, M. Paskin, and R. Sukthankar, "Distributed localization of networked cameras," in *Proceedings of the fifth international conference on Information processing in sensor networks*, (New York, NY, USA), pp. 34–42, ACM Press, 2006.

[29] A. Galstyan, B. Krishnamachari, K. Lerman, and S. Pattem, "Distributed online localization in sensor networks using a moving target," in *Proceedings of the*

third international symposium on Information processing in sensor networks, (New York, NY, USA), pp. 61–70, ACM Press, 2004.

[30] S. Ganu, A.S.Krishnakumar, and P.Krishnan, "Infrastructure-based location estimation in wlan networks," in *IEEE Wireless Communications and Networking Conference*, (Atlanta, GA, USA), March 2004.

[31] C. Gentile and L. Berndt, "Robust location using system dynamics and motion constraints," in *Proceedings of the IEEE International Conference on Communications*, pp. 1360–1364, June 2004.

[32] M. G. Genton, "Classes of kernels for machine learning: A statistics perspective," in *Journal of Machine Learning Research*, vol. 2, pp. 299–312, December 2001.

[33] A. Goldsmith, *Wireless Communications*. Cambridge: Cambridge University Press, 2005.

[34] Y. Gwon, R. Jain, and T. Kawahara, "Robust indoor location estimation of stationary and mobile users," in *Proceedings of the Conference on Computer Communications*, (Hong Kong), 2004.

[35] A. Haeberlen, E. Flannery, A. M. Ladd, A. Rudys, D. S. Wallach, and L. E. Kavraki, "Practical robust localization over large-scale 802.11 wireless networks," in *Proceedings of the Tenth Annual International Conference on Mobile Computing and Networking*, (Philadelphia, PA, USA), pp. 70–84, 2004.

[36] J. Ham, D. Lee, and L. Saul, "Semisupervised alignment of manifolds," in *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pp. 120–127, Society for Artificial Intelligence and Statistics, January 2005.

[37] D. Hardoon, S. Szedmak, and J. Shawe-Taylor, "Canonical correlation analysis; an overview with application to learning methods," *Neural Computation*, vol. 16, pp. 2639–2664, 2004.

[38] H. Hashemi, "The indoor radio propagation channel," in *Proceedings of the IEEE*, vol. 81, pp. 943–968, July 1993.

[39] A. Hatami and K. Pahlavan, "Comparative statistical analysis of indoor positioning using empirical data and indoor radio channel models," in *Proceedings of IEEE Consumer Communications and Networking Conference*, vol. 2, pp. 1018 – 1022, Janurary 2006.

[40] B. Hendrickson, "Latent semantic analysis and fiedler embeddings," in *Proceedings of SIAM Workshop on Text Mining*, April 2006.

[41] M. Herbster, M. Pontil, and L. Wainer, "Online learning over graphs," in *Proceedings of the Twenty-Second International Conference on Machine Learning*, (New York, NY, USA), pp. 305–312, ACM Press, August 2005.

[42] H. Hotelling, "Relations between two sets of variates," *Biometrika*, vol. 28, pp. 312–377, 1936.

[43] L. Hu and D. Evans, "Localization for mobile sensor networks," in *Proceedings of the Tenth Annual International Conference on Mobile Computing and Networking*, (Philadelphia, PA, USA), pp. 45–57, 2004.

[44] O. C. Jenkins and M. J. Mataric, "A spatio-temporal extension to isomap nonlinear dimension reduction," in *Proceedings of the twenty-first international conference on Machine learning*, (New York, NY, USA), p. 56, ACM Press, 2004.

[45] G. Jiang and L. Quan, "Detection of concentric circles for camera calibration," in *Proceedings of the Tenth IEEE International Conference on Computer Vision*, pp. 333–340, Octorber 2005.

[46] K. Kaemarungsi and P. Krishnamurthy, "Modeling of indoor positioning systems based on location fingerprinting," in *Proceedings of the International Conference on Computer Communications*, no. 1, (Hong Kong), pp. 1013–1023, 2004.

[47] K. Kaemarungsi and P. Krishnamurthy, "Properties of indoor received signal strength for wlan location fingerprinting," in *First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, vol. 00, (Los Alamitos, CA, USA), pp. 14–23, IEEE Computer Society, 2004.

[48] A. Kapadia, T. Henderson, J. Fielding, and D. Kotz, "Virtual walls: Protecting digital privacy in pervasive environments," in *Proceedings of the Fifth International Conference on Pervasive Computing*, vol. 4480, pp. 162–179, May 2007.

[49] O. Kouropteva, O. Okun, and M. Pietikainen, "Incremental locally linear embedding algorithm," *Pattern Recognition*, vol. 38, pp. 1764–1767, October 2005.

[50] P. Krishnan, A. S. Krishnakumar, W. Jun, C. Mallows, and S. Ganu, "A system for LEASE: Location estimation assisted by stationary emitters for indoor rf wireless networks," in *Proceedings of the Conference on Computer Communications*, (Hong Kong), 2004.

[51] A. M. Ladd, K. E. Bekris, A. Rudys, G. Marceau, L. E. Kavraki, and D. S. Wallach, "Robotics-based location sensing using wireless ethernet," in *Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking*, (Atlanta, GA, USA), pp. 227–238, September 2002.

[52] N. Landwehr, M. Hall, and E. Frank, "Logistic model trees," in *Proceedings of the European Conference on Machine Learning*, (Cavtat-Dubrovnik, Croatia), pp. 241–252, September 2003.

[53] M. Langheinrich, "A privacy awareness system for ubiquitous computing environments," in *Proceedings of the 4th international conference on Ubiquitous Computing*, (London, UK), pp. 237–245, Springer-Verlag, 2002.

[54] M. H. C. Law and A. K. Jain, "Incremental nonlinear dimensionality reduction by manifold learning," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 28, no. 3, pp. 377–391, 2006.

[55] M. H. C. Law, N. Zhang, and A. K. Jain, "Nonlinear manifold learning for data stream," in *Proceedings of the Fourth SIAM International Conference on Data Mining*, (Lake Buena Vista, Florida, USA), April 2004.

[56] J. Lester, T. Choudhury, N. Kern, G. Borriello, and B. Hannaford, "A hybrid discriminative/generative approach for modeling human activities," in *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, (Edinburgh, Scotland), pp. 766–772, 2005.

[57] J. Letchner, D. Fox, and A. LaMarca, "Large-scale localization from wireless signal strength," in *Proceedings of the 20th National Conference on Artificial Intelligence*, (Pittsburgh, USA), pp. 15–20, July 2005.

[58] L. Liao, D. Fox, and H. Kautz, "Learning and inferring transportation routines," in *Proceedings of the 19th National Conference on Artificial Intelligence*, (San Jose, CA, USA), pp. 348–353, July 2004.

[59] L. Liao, D. Fox, and H. Kautz, "Location-based activity recognition using relational markov networks," in *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, (Edinburgh, Scotland), pp. 773–778, 2005.

[60] K. Lorincz and M. Welsh, "Motetrack: A robust, decentralized approach to RF-based location tracking," in *Personal and Ubiquitous Computing, Special Issue on Location and Context-Awareness*, Springer, 2006.

[61] M. S. J. N. M. Ocana, L.M. Bergasa and R. Flores, "Indoor robot localization system using wifi signal measure and minimizing calibration effort," vol. 4, pp. 1545– 1550, June 2005.

[62] D. Maligan, E. Elnahrawy, R. Martin, W. Ju, P. Krishnan, and A. Krishnakumar, "Bayesian indoor positioning systems," in *Proceedings of the Conference on Computer Communications*, vol. 2, (Miami, FL, USA), pp. 1217–1227, March 2005.

[63] D. Moore, J. Leonard, D. Rus, and S. Teller, "Robust distributed network localization with noisy range measurements," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, (Baltimore, MD, USA), pp. 50–61, ACM Press, 2004.

[64] X. Nguyen, M. I. Jordan, and B. Sinopoli, "A kernel-based learning approach to ad hoc sensor network localization," *ACM Transactions on Sensor Networks*, vol. 1, no. 1, pp. 134–152, 2005.

[65] L. Ni, Y. Liu, Y. Lau, and A. Patil, "LANDMARC: Indoor location sensing using active RFID," in *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*, (Dallas, TX, USA), pp. 407–416, March 2003.

[66] V. M. Olivera, J. M. C. Plaza, and O. S. Serrano, "Wifi localization methods for autonomous robots," *Robotica*, vol. 24, no. 4, pp. 455–461, 2006.

[67] J. J. Pan, J. T. Kwok, Q. Yang, and Y. Chen, "Accurate and low-cost location estimation using kernels," in *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, (Edinburgh, Scotland), pp. 1366–1371, 2005.

[68] J. J. Pan, J. T. Kwok, Q. Yang, and Y. Chen, "Multidimensional vector regression for accurate and low-cost location estimation in pervasive computing," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, pp. 1181– 1193, September 2006.

[69] J. J. Pan and Q. Yang, "Co-localization from labeled and unlabeled data using graph laplacian," in *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, (Hyderabad, India), pp. 2166–2171, 2007.

[70] J. J. Pan, Q. Yang, H. Chang, and D. Y. Yeung, "A manifold regularization approach to calibration reduction for sensor-network based tracking," in *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, (Boston, USA), pp. 988–993, July 2006.

[71] J. J. Pan, Q. Yang, and S. J. Pan, "Online co-localization in indoor wireless networks by dimension reduction," in *Proceedings of the Twenty-Second National Conference on Artificial Intelligence*, (Vancouver, Canada), pp. 1102–1107, 2007.

[72] N. Patwari and A. O. Hero, "Manifold learning algorithms for localization in wireless sensor networks," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, May 2004.

[73] N. Patwari and A. O. Hero, "Adaptive neighborhoods for manifold learning-based sensor localization," in *Proceedings of the 2005 Signal Processing and Wireless Communications Conference*, June 2005.

[74] M. Perkowitz, M. Philipose, K. Fishkin, and D. J. Patterson, "Mining models of human activities from the web," in *Proceedings of the 13th international conference on World Wide Web*, (New York, NY, USA), pp. 573–582, ACM Press, 2004.

[75] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C*. New York: Cambridge University Press, 2nd ed., 1992.

[76] N. B. Priyantha, A. K. Miu, H. Balakrishnan, and S. Teller, "The cricket compass for context-aware mobile applications," in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, (Rome, Italy), pp. 1–14, 2001.

[77] R. Raina, Y. Shen, A. Y. Ng, and A. McCallum, "Classification with hybrid generative/discriminative models," in *Advances in Neural Information Processing Systems 16* (S. Thrun, L. Saul, and B. Schölkopf, eds.), Cambridge, MA: MIT Press, 2004.

[78] T. Roos, P. Myllymäki, and H. Tirri, "A statistical modeling approach to location estimation," *IEEE Transactions on Mobile Computing*, vol. 1, no. 1, pp. 59– 69, 2002.

[79] T. Roos, P. Myllymaki, H. Tirri, P. Misikangas, and J. Sievanen, "A probabilistic approach to WLAN user location estimation," *International Journal of Wireless Information Networks*, vol. 9, pp. 155–164, July 2002.

[80] A. Savvides, C. Han, and M. B. Strivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, (Rome, Italy), pp. 166–179, 2001.

[81] B. Schölkopf and A. Smola, *Learning with Kernels*. Cambridge, MA: MIT Press, 2002.

[82] A. Schwaighofer, M. Grigoras, V. Tresp, and C. Hoffmann, "GPPS: A Gaussian process positioning system for cellular networks," in *Advances in Neural Information Processing Systems* (S. Thrun, L. Saul, and B. Schölkopf, eds.), vol. 16, (Cambridge, MA), MIT Press, 2004.

[83] Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz, "Localization from mere connectivity," in *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, (New York, NY, USA), pp. 201–212, ACM Press, 2003.

[84] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. New York, NY, USA: Cambridge University Press, 2004.

[85] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[86] V. Sindhwani, P. Niyogi, and M. Belkin, "Beyond the point cloud: from transductive to semi-supervised learning," in *Proceedings of the 22nd international conference on Machine learning*, (New York, NY, USA), pp. 824–831, ACM Press, 2005.

[87] A. Smailagic and D. Kogan, "Location sensing and privacy in a context-aware computing environment," *IEEE Wireless Communications*, vol. 9, no. 5, pp. 10–17, 2002.

[88] D. C. Steere, A. Baptista, D. McNamee, C. Pu, and J. Walpole, "Research challenges in environmental observation and forecasting systems," in *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking*, (Boston, MA, USA), pp. 292–299, 2000.

[89] M. Stein, *Interpolation of Spatial Data Some Theory for Kriging*. Springer Verlag, June 1999.

[90] P. Tao, A. Rudys, A. M. Ladd, and D. S. Wallach, "Wireless lan location-sensing for security applications," in *Proceedings of the 2003 ACM workshop on Wireless security*, (New York, NY, USA), pp. 11–20, ACM Press, 2003.

[91] E. M. Tapia, S. S. Intille, and K. Larson, "Activity recognition in the home using simple and ubiquitous sensors," in *Second International Conference Pervasive Computing*, pp. 158–175, 2004.

[92] C. Taylor, A. Rahimi, J. Bachrach, H. Shrobe, and A. Grue, "Simultaneous localization, calibration, and tracking in an ad hoc sensor network," in *Proceedings of the fifth international conference on Information processing in sensor networks*, (New York, NY, USA), pp. 27–33, ACM Press, 2006.

[93] K. F. Wong, I. W. Tsang, V. Cheung, S. H. Chan, and J. T. Kwok, "Position estimation for wireless sensor networks," in *Proceedings of the IEEE Global Telecommunications Conference*, vol. 5, pp. 2772–2776, 2005.

[94] D. Wyatt, M. Philipose, and T. Choudhury, "Unsupervised activity recognition using automatically mined common sense," in *Proceedings of the Twentieth National Conference on Artificial Intelligence*, pp. 21–27, July 2005.

[95] Z. Xiang, S. Song, J. Chen, H. Wang, J. Huang, and X. Gao, "A wireless lan-based indoor positioning technology," *IBM Journal of Research and Development*, vol. 48, no. 5/6, pp. 617–626, 2004.

[96] Q. Yang, Y. Chen, J. Yin, and X. Chai, "LEAPS: A location estimation and action prediction system in a wireless lan environment," in *International Conference on Network and Parallel Computing Workshop on Building Intelligent Sensor Networks*, (Wuhan, China), pp. 584–591, 2004.

[97] J. Yin, X. Chai, and Q. Yang, "High-level goal recognition in a wireless LAN," in *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, (San Jose, CA, USA), pp. 578–584, July 2004.

[98] J. Yin, D. Shen, Q. Yang, and Z. N. Li, "Activity recognition through goal-based segmentation," in *Proceedings of the Twentieth National Conference on Artificial Intelligence*, (Pittsburgh, PA, USA), pp. 28–33, July 2005.

[99] J. Yin, Q. Yang, and L. Ni, "Adaptive temporal radio maps for indoor location estimation," in *Proceedings of the 3rd Annual IEEE International Conference on Pervasive Computing and Communications*, (Kauai Island, HI, USA), pp. 85–94, 2005.

[100] J. Yin, Q. Yang, and J. J. Pan, "Sensor-based abnormal human-activity detection," *Special Issue on Intelligence and Security Informatics, IEEE Transactions on Knowledge and Data Engineering*, 2008.

[101] M. Youssef and A. Agrawala, "Handling samples correlation in the Horus system," in *Proceedings of the International Conference on Computer Communications*, vol. 2, (Hong Kong), pp. 7–11, March 2004.

[102] M. Youssef, A. Agrawala, and U. Shankar, "WLAN location determination via clustering and probability distributions," in *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*, (Fort Worth,TX, USA), pp. 143–150, March 2003.

[103] H. Zha, X. He, C. Ding, H. Simon, and M. Gu, "Bipartite graph partitioning and data clustering," in *Proceedings of the tenth International Conference on Information and Knowledge Management*, (New York, NY, USA), pp. 25–32, ACM Press, 2001.

[104] D. Zhang, J. Ma, Q. Chen, and L. M. Ni, "An rf-based system for tracking transceiver-free objects," in *Fifth Annual IEEE International Conference on Pervasive Computing and Communications*, pp. 135–144, 2007.

[105] D. Zhao and L. Yang, "Incremental construction of neighborhood graphs for nonlinear dimensionality reduction," in *Proceedings of the 18th International Conference on Pattern Recognition*, (Washington, DC, USA), pp. 177–180, IEEE Computer Society, 2006.

[106] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the Twentieth International Conference on Machine Learning*, (Washington DC, USA), pp. 912–919, August 2003.