Thesis Results

Divye Kapoor

June 28, 2011

CANDIDATE'S DECLARATION

I hereby declare that the work being presented in the dissertation work entitled "**Particle Filter Based Scheme for Indoor Tracking on an Android Smartphone**" towards the partial fulfillment of the requirement for the award of the degree of **Integrated Dual Degree in Computer Science and Engineering (with specialization in Information Technology)** and submitted to the **Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, India** is an authentic record of my own work carried out during the period from May, 2010 to June, 2011 under the guidance and provision of **Dr. Manoj Misra, Professor, Department of Electronics and Computer Engineering, IIT Roorkee**.

I have not submitted the matter embodied in this dissertation work for the award of any other degree and diploma.

Date: June, 2011

Place: Roorkee

(DIVYE KAPOOR)

CERTIFICATE

This to certify that the work contained in the dissertation entitled "**Particle Filter Based Scheme for Indoor Tracking on an Android Smartphone**" by Divye Kapoor of Integrated Dual Degree in Computer Science and Engineering (with specialization in Information Technology), has not been submitted elsewhere for a degree or diploma to the best of my knowledge.

Date: June, 2011

Place: Roorkee

Dr. Manoj Misra
Professor,
E&CE Department
IIT Roorkee, India

Acknowledgements

I would like to take this opportunity to extend my most heartfelt gratitude to my guide and mentor - **Prof. Manoj Misra**, Department of Electronics and Computer Engineering, IIT Roorkee. His guidance, attention to detail and drive for excellence have contributed greatly towards my education and also towards ensuring that sterling work is done as part of my Masters dissertation. His encouragement and unstinting faith have been essential in helping me learn, grow and push the limits of my capabilities. I am fortunate to have been able to do my dissertation under him.

My sincere thanks are due to **Dr. S. N. Sinha**, Professor and Head of the Department of Electronics and Computer Engineering for providing the facilities and hardware without which this thesis would have been all but impossible. Thanks are also due to **Dr. Raheja**, Institute Architect, IIT Roorkee for providing the blueprints of Ravindra Bhawan, the test site for this thesis. My work would have been immensely more difficult without his help and cooperation. The **Mahatma Gandhi Central Library** of IIT Roorkee also deserves a mention in the Acknowledgements. They have provided unfettered access to the world's highest quality research published in the IEEE and ACM journals. I have always felt well supplied with information thanks to their hard work.

Sincere gratitude is also due to the authors of the many open source software projects that have been very useful in the making of this thesis. The list of projects is very long and it is impossible to name all, but I would like to give a special mention to the authors and contributors of **LaTeX, GNUPlot, the GNU R Language, the GNU Core Utilities, Python, Linux, Android** and **Eclipse**. This thesis is the result of standing on the shoulders of giants and I am indebted to you for your efforts in maintaining such high quality tools for the benefit of the world. Many many thanks.

Acknowledgements aren't complete without thanking **friends and family**. They are the ones who have given unconditional encouragement and support. They have shared my happiness on the days that things have worked and they have helped get me through the days when things haven't. They have stood by me through thick and thin and for that I am very grateful.

Abstract

The development of stable outdoor positioning and tracking algorithms has ushered in a new era of technology. Today GPS based navigators are ubiquitous and GPS based positioning and navigation has been integrated into smartphones. This has made life easy for navigation over large distances in an outdoor scenario. A new horizon for research has developed which aims at providing that same ease of use in navigation and tracking in an indoor context.

Over the last few years, advances in technology have allowed for the development and integration of MEMS sensors into smartphones. These sensors (an accelerometer, a gyroscope and a magnetometer) can be leveraged to develop an indoor tracking solution. However, the resource constrained nature of smartphones necessitates modifications to prior work achieving the same in a non-mobile context. Additionally, the fact that the tracking device lies in a user's palm and not on a protected, stable surface (as is the case in robotics) generates a new set of challenges.

This thesis proposes and implements an online particle filter based algorithm for indoor positioning on an Android smartphone for pedestrians. The proposed algorithm is not only able to work well on the resource constrained device, it actually yields an accuracy better than prior work in this area. Such results have been made possible by a careful approach to the problem with decisions made on factual data determined before and during the implementation of the proposed work. The final system is able to achieve a mean error of $0.5m$ with a very low standard deviation value of $0.24m$ and provides a clean, intuitive tracked path. Results for comparison are produced using a simple dead reckoning approach and a Nearest Neighbour Wifi corrected tracking algorithm. Both the algorithms are tested in the same environment as the proposed work.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

The world is an exciting place today. We are surrounded by technology that has woven itself seamlessly into our lives. Today, many of us see the time on our mobile phones, connect with friends and family through mobile versions of popular social networks accessed through our smartphones. We talk to people across the globe using that same mobile device and the smartphone doubles up as a gaming console and eBook reader. The availability of a GPS sensor on smartphones has been a growing trend for the past few years and is now nearly a de facto standard. Implementation of algorithms produced by researchers over the past two decades on smartphones has resulted in tracking capability that is quite accurate and suitable for navigation at outdoor locations. In fact, latest generation devices like the Samsung Nexus S even ship with a beta mobile application that doubles up as a standard car based GPS navigation device. So, while engineers at companies are working furiously to solve technical challenges for accurate tracking and navigation outdoors, a different problem looms on the horizon:

> How do we take such tracking and navigational aids for mobiles indoors where GPS based systems are virtually useless?

This thesis will analyze this question and show multiple systems that provide indoor positioning and tracking using not much more than a cutting edge smartphone. But before moving further, let me state my reasons for picking this up as my dissertation proposal.

1.1 Motivation

For my dissertation, I wished to take up a topic that was current, technologically intensive and had a broad scope of application. I had prior exposure to location based services and map matching algorithms via my seminar report[2] in 2010 and I was firmly convinced that Location Based Services are going to play a very valuable part in the evolving future of mobile devices. To enable a future that has relevant location based content, reliability of location of the user - both indoors and outdoors is a high priority. The outdoor location and tracking problem was being solved by using GPS, map matching and other filtering techniques. Long distance navigational aids were commercially available and the field is rather mature. In contrast, the field of indoor positioning was highly fragmented, technologically diverse with no clear winners and few commercial offerings. The area was thus ripe for research.

1.2 Potential Applications

There are numerous potential applications for research in this area. Some are obvious - Indoor navigation and tracking, applications as navigational aids for the differently abled, and even location based services that aim to improve customer experiences in malls and shopping complexes. However, some of the more interesting applications of this technology are in fields that are just opening up to research and implementation.

1.2.1 Navigational aids for the abled and differently abled

Having a stable, highly accurate tracking solution is a prerequisite to provide navigational aids in unfamiliar surroundings like a customer at a mall or inside a huge departmental store. Visitors to huge office complexes could also benefit with such navigational aids. The biggest beneficiaries of such technology are the differently abled - who will be able to leverage such tools in coordination with highly mature devices customized to their needs so that they can move in indoor surroundings with just as much confidence as they do while using other navigational aids like GPS outdoors.

1.2.2 Inventory tracking and retreival

There are a number of large industries (eg. shipping industry, online ecommerce vendors, private couriers, wholesalers etc.) that invest heavily in large warehouses. One of their primary concerns is inventory management and retreival. Most of the companies in these industries use some form of inventory tracking software that tracks what is present in the warehouses. They

also evolve conventions for storage that allow for easy retrieval of inventory. An indoor tracking and navigation solution integrated with the inventory system can help workers easily and efficiently locate objects present in their warehouses and reduce incidents of misplacements or loss.

1.2.3 Augmented Reality and Reality Fusion Gaming

Among all the applications of indoor tracking - this is one that holds the greatest excitement.Augmented reality can be defined concisely as:

> A technology that superimposes a computer-generated image on a user's view of the real world, thus providing a composite view. (Adapted from [3])

The field of AR is not new, however, technology has advanced to the point that AR applications can now be built to run on smartphones while providing exceptional engagement and interactivity. Researchers at Georgia Tech University demonstrated such integration through a two player interactive AR game running on an Android device at the 2010 Uplinq conference. The game was playable anywhere provided a simple mat denoting the gaming area was placed on the floor. The virtual characters (or sprites) were then created and layered on a camera view of the mat. The two players could interact with the characters virtually and their view of the sprites was altered in near real-time as they moved around the mat. Another major application of AR to mobile devices has been shown by Layar wherein they overlay 3D models and informative content directly onto a user's view of a location.

Integration of a stable indoor tracking solution with such AR applications will allow the development of a new class of applications where a user's action and location in real-life will correspond to a sprite's action in the virtual world thus opening up arenas for AR based first person shooters etc. The possibilities opened up by such technology are immense and limited only by imagination.

1.3 Problem Statement

All the applications mentioned above put different constraints of accuracy, deployability, mobility and scalability on the tracking algorithm. We concentrate on a specific type of the indoor tracking problem and thus limit the scope of the thesis to the well defined problem statement below:

> Develop and characterize a smartphone based particle filter sensor fusion approach for online indoor tracking of pedestrians.

1.4 Organization of the Thesis

The thesis is split into various chapters for the convenience of the reader.

- Chapter 2 details the flow of ideas and prior work by researchers in this area.
- Chapter 3 specifies the specific methods proposed by the author for implementation of an indoor tracking system. The methods detail the modifications proposed for implementation on a mobile device as well as the choices guiding the approach.
- Chapter 4 discusses the implementation details of the algorithms on an Android device and explains the architectural organization of the entire codebase.
- Chapter 5 specifies the field work that was done to characterize the Microelectromechanical Systems (MEMS) sensors and Wifi receiver on the smartphone and includes the results of a project that implemented an indoor positioning system based on the received signal strengths from different Wifi Access Points. This work was done to determine a choice of suitable parameters for the algorithms specified in Chapter 3.
- Chapter 6 discusses the performance of various components of the solution and shows the performance of the final proposed method contrasted with a naive dead reckoning approach and a Wifi based indoor tracking approach.
- Chapter 7 finally summarizes the experiences derived from the thesis and describes scope for future work in this area.
- Additional test results which were too voluminous to include in the main text of this thesis are present in the Appendix.
- The papers and other sources of information used in this thesis are listed in the bibliography at the end of this thesis.

2

Literature Review

2.1 Technologies for Unaided Indoor Positioning

Indoor positioning has been a topic of active research for the past decade with the first research system using RF signals for distance estimation and indoor positioning being produced by Microsoft Research (RADAR, 2000)[4]. Of course, systems like Active Badge[5] have been developed as far back as 1992 to locate targets indoors but they have shown limited utility on account of the requirement to deploy specialized sensors for detecting the radio tags deployed within the system.

Since RADAR, a number of improvements have been made in the core algorithms and technologies for positioning. Over time, indoor positioning has progressed from simply using RF signals to using ubiquitous 802.11 Access Points as radio sources. With the development of UWB (Ultra Wide Band) technology, very fine grained ranging and tracking results have been obtained over target distances of the order of a few hundred feet from the radio source.[6]

However, the primary scale of interest for commercial exploitation of indoor positioning and tracking is of the order of the size of warehouses and malls (approximately of the order of $2500m^2$) with an accuracy that is preferably of the order of a few meters. Ultimately, the following goal needs to be met:

> Create a low complexity, low cost indoor positioning system that is maintainable, reasonably accurate and highly scalable.

UWB, unfortunately, does not scale easily to such large spaces without requiring that a significant number of sensors and repeaters be installed in the target positioning area specifically for this purpose. It is also an expensive technology. On the other hand, IEEE 802.11 AP based positioning has low complexity and cost and it has a large installed base, thus making it attractive for positioning purposes as a source of radio signals.

Thus it is interesting to explore the limitations of accuracy, scalability and maintainability of indoor positioning systems that are based on IEEE 802.11 Access Points.

Other technologies for indoor positioning include laser range finding, ultrasound ranging etc.

2.2 Indoor Tracking

Once having worked with indoor positioning systems, it is but natural to ask: Can we extend all the technology developed for indoor positioning to track objects and devices? Also, can we do it fast enough to provide a near-realtime experience?

Research shows that the answer to both the questions is in the affirmative and papers as far back as 2000 [4] have attempted to achieve this

2.3 Progression of Ideas

The progression of ideas in this field spans a decade of intensive research. However, the central themes can be easily summarized:

- Initial seminal work was done on indoor location and tracking based on RF signal strength data.[4]
- The work was improved with the aid of more powerful filters and better algorithms and was applied to different signal sources such as Wifi and Bluetooth.[7][8]
- Analysis of Wifi signals indicated a strong effect of orientation on the sample data. This observation was incorporated into various systems to improve indoor localization accuracy.[9][10]
- The advent of Microelectromechanical Systems (MEMS) allowed the development of compact accelerometers on chips and subsequently gyroscopes have also been created.

These were sufficient for the development of portable, low cost, Inertial Navigation Systems (INS). An attempt was made to merge the benefits of an INS with a Wifi based approach to indoor localization.[11][12]

- Particle filters were explored for indoor localization.[13][14][15]
- Implementation of Wifi RSSI based indoor positioning and tracking systems on mobile devices start to appear.[16]

Subsequent sections of this chapter describe the papers and their important ideas and limitations in greater detail in a chronological format.

2.4 Early work (2000-2003)

Bahl and Padmanabhan[4] (2000) proposed RADAR as the first system to utilize RF signals to achieve indoor positioning using a PC. They used a WaveLAN NIC to get signal strength and SNR values from a received and from statistical data collected they were able to carry out indoor location and tracking. Their work was seminal and formed the basis of many indoor localization methods. The central ideas in their paper were twofold:

1. Treat the problem as a signal processing problem that aims to compensate for free space path loss and other multipath and obstructive effects of the RF source signals by using a propagation model, thus allowing the use of trilateration for positioning. OR
2. Treat the problem as a nearest neighbor problem for a set of predefined signal samples associated with fixed locations in the target area. This was called the location fingerprinting method.

These ideas were central to the research carried out over the next few years.

Kotanen, Hannikainen, Leppakoski et al[7] (2003) took forward RADAR by exploring alternative radio sources and produced an indoor positioning system using Bluetooth devices as their radio source. Their major contribution was the use of Kalman Filters and other signal processing techniques to improve the accuracy of the location estimate. The major limitation of this method for achieving the stated objectives is that Bluetooth capable anchor points are not deployed in sufficiently large numbers and a Bluetooth based indoor location technology would require additional hardware deployment and does not have sufficient range to justify the extra cost and complexity of the solution.

Other systems created during this period include SpotOn[8] by Intel Research which took indoor localization to 3 dimensions using RFID tags. All systems in this period had significant accuracy limitations with best case accuracies approaching $3m$.

2.5 Wifi Positioning Algorithm Maturation (2004-2006)

2004 was a year when indoor positioning really started to generate research interest. The primary progress during this period was the analysis of signal characteristics of IEEE 802.11 systems in the 2.4 GHz band as well as the maturation of approaches to positioning based on probabilistic and fuzzy models.

2.5.1 Analysis of the properties of Wifi RSSI

Kaemarungsi and Krishnamurthy[1] (2004) did a thorough study of the properties exhibited by IEEE 802.11 signals in the 2.4 GHz ISM band as they apply to a real world positioning scenario in this paper. Their major contribution was the statistical analysis of Received Signal Strength datasets. Kaemarungsi and Krishnamurthy first collected data from a single access point to determine the impact of the presence of a user as well as the statistical properties of the dataset (the statistical distribution, the signal stationarity, the variation of the signal at different times of the day etc.). After a thorough analysis, they extended their analysis to datasets generated from multiple access points to determine the degree of independence of signals from the different access points and the similarity of the statistical properties of the access points in a multi-access point scenario. The final part of their work dealt with the differences between RSS fingerprints of two locations to identify commonalities and differences between fingerprints. The major results of their work are summarized in Table 2.1.

These results are very important as subsequent systems have failed to take the results of this paper into account while designing their algorithms. Their results as well as the groundwork done as part of this thesis forms a significant informing factor for choosing appropriate algorithms for the problem.

2.5.2 Orientation Augmented Positioning and Tracking Systems

Ladd et al (2003, 2005)[9] took up a different approach to the problem of indoor localization and tracking they preferred to use the probabilistic viewpoint and take an approach keeping in mind its practical application in robotics. Their paper, published in 2003 as part of IEEE MOBICOM and later in a revised form in 2005 by Springer used a fairly advanced probabilistic model to predict the position of a robot. They used a Bayesian Inference Algorithm to

Statistical Property	Variations
Effect of Users presence	Upto 5 dBm variation in the properties
Effect of Users body on Standard Deviation	Increases from 0.68 dBm to 3.00 dBm
Effect of Users orientation	Deviations of upto 10 dBm including complete loss of signal strength for Line of Sight obstruction in low signal strength scenarios
Statistical Distribution	Although other published results claim a lognormal distribution of RSS values they fail to mention the presence or absence of a user. As per the authors, in the presence of a user, the lognormal distribution is violated with no clear distribution fit.
Standard Deviation	Except when a users orientation blocks an Access Point, the standard deviation values at a particular point are relatively stable.
Stationarity of the RSS	The RSS distribution was found to be fairly stable and exhibited ergodic properties at small time scales but stationarity was violated over longer time scales of the order of hours due to environmental changes.
RSS Correlation between APs	Fairly uncorrelated, nearly independent
Interference from multiple APs on same channel	Nearly independent on account of MAC layer avoiding simultaneous broadcasts
Clustering of RSS values	RSS values corresponding to different APs show significant clustering behaviour around radio sources and may thus be used with a discriminant to distinguish between locations. Also, the number of distinct tuples for each location are fewer than the number of samples.

Table 2.1: Kaemarungsi and Krishnamurthy Analysis Results of RSS datasets (summarized from [1])

generate probability values for motion and position. They also attempted to coarsely estimate the orientation of a user based on the sample data provided to compensate for orientation effects mentioned in [1]. Their model though fairly complex and quite accurate is solely built upon a Wifi sample dataset augmented by rudimentary orientation information (discretized to 8 directions) and assumes a gaussian distribution of Wifi signal strengths around a sample (an assumption which is not valid according to [1]). However, its central ideas are excellent and the results were the best among its contemporaneous systems.

In 2006, King et al[10] provided the first description of any system that utilized digital compasses to determine user orientation. Their probabilistic algorithm utilized Bayes Rule and an assumed Gaussian distribution of signal strengths around the reference points to generate location candidates. Orientation information from digital compass measurements was used to limit the dataset to only those samples in the training set that had a similar orientation as the current test datapoint. Simple averaging of the resulting most probable data points was returned to the user as the most likely location. Bayes' rule was used for localization and no tracking applications were evaluated.

With the number of systems attempting to provide indoor positioning and tracking solutions ballooning, IEEE published a survey[17] by Hui Liu et al which compared the accuracies and features of the variety of indoor positioning systems. Interested readers may refer to the survey paper for a huge list of positioning and tracking systems and a concise description of the ideas and methodologies behind them as well as a comparative analysis of their features. Such comparisons have been omitted here for the sake of brevity.

2.6 INS and Wifi Integration (2006-2008)

The location system that I found the most interesting and suitable for application to the task at hand was the integration of an Intertial Navigation System (INS) with Wifi measurements done by Frederic Evennou and Francois Marx of the R&D division of TECH/IDEA, France Telecom. Their paper [11] attempted to fuse an inertial navigation system using accelerometer and gyroscope data with Wifi location fingerprints for continuous tracking of a user. Their reported accuracy of $1.53m$ was significantly better than that of pure Wifi location fingerprinting solutions reported previously. However, in their work, they reported that the particle filter used was extremely intensive computationally and was thus not suitable for implementation on mobile devices. Additionally, they used the log-normal approximation relationship for signal strength versus distance - this relationship has been shown to not be valid for indoor scenarios by work done by Kaemarungsi [1]. This is apparently a major factor in degrading their results. This led them to believe that the algorithm was not suitable for implementation on a mobile

device. To quote:

> Due to the large number of particles, the algorithm is too complex to be implemented on handheld devices. A way to cut down this number of particles must be found.[11]

Another major system of this kind was published by Wang et al (2007)[12] which fused Wifi Received Signal Strength Information (RSSI) with a step counting algorithm based on [18] and map information to develop a pedestrian tracking algorithm based on particle filters. Their system was reported to have a mean error of $4.3m$ with a standard deviation in error of $2.8m$ in comparison to a pure KNN based system with a mean error of $6.44m$ and a standard deviation of $6.84m$. Though the reported accuracy of their system was poor, the paper had a number of interesting ideas.

To the best of the author's knowledge, none of these systems were ever implemented on a mobile device.

2.7 Smartphone based Indoor Positioning and Tracking Systems (2008-2011)

Not many examples of smartphone based indoor tracking systems are found. Redpin[16] was an implementation of a Wifi based room level accurate system on a Nokia N95. The implementation was probably finished around 2008. Subsequent work has produced an iPhone and a slightly unstable Android client for the Redpin server based system - both of which are available in the public subversion repository of the project. Work on this system is ongoing and is concentrating on Wifi as its primary source of positioning information.

Au [19] implemented a Wifi based compressive sensing method for indoor positioning and tracking as part of a Masters thesis in 2010. A simple dead reckoning scheme with error correction based on multiple mobile devices on the same pedestrian has been proposed by [20] in March 2011. Lukianto, Honniger and Sternberg[21] claim to be developing an implementation on a Nokia N900 (released 2010) but have not published any algorithms or results. (Note: the author was unaware of these systems at the time of system design and implementation given that they are so recent in nature).

2.8 Research Gaps

Based on a review of a wide variety of papers besides those mentioned in the previous section, the following issues have been identified:

- Many systems make assumptions of log-normal decay of signals strengths from Wireless Access Points. Such assumptions are not valid as per the research done in [1] and verified as part of the groundwork described in Chapter 5.

- A decade of research on Wifi assisted indoor positioning system has not provided a satisfactory increase in accuracy of positioning using Wifi data. Even the best systems report errors of the order of $2m$ with a very high standard deviations.

- The lower limit of resolution for a Wifi assisted system is 1 wavelength. For systems based on the $2.4GHz$ IEEE 802.11 standard, this limit is approximately $0.125m$ in the absence of noise. However, given the amount of noise incurred in the Wifi Signal Strengths (as evaluated in Chapter 5), it is unlikely that this limit will ever be reached and we shall have to be content with systems that give at best error bounds of around 10x the theoretical maximum resolution.

- Very few systems appreciate the enormous informative content present in maps and the possibility of using map information for continuous error correction in systems.

- None of the dead reckoning solutions being implemented on mobile devices are aiming to use map information in any way to aid tracking.

- Nearly all the indoor positioning systems were aiming to use Wifi data in some way to decrease the error of their systems. However, this may not always be appropriate.

The proposed work aims to improve upon prior work in this area and demonstrate the implementation of indoor positioning systems using a modern Android based smartphone. The discussion of the proposed algorithms is in Chapter 3.

3

Particle filter Based Map and Sensor Fusion technique for Indoor Tracking using an Android Smartphone

3.1 Relevant Prior Work

As described in Chapter 2 (Literature Review), the pedestrian navigation system described in [12] shows the most promise for accurate tracking of subjects in indoor tracking scenarios. The results of [11] which implements a similar system integrating an INS with Wifi data are very encouraging. However, we are cautioned against the computational complexity of a particle filter based solution and its attendant implications while implementing the same on a mobile device by Evennou and Marx[11]. As we shall later see, these cautionary words are well founded.

Our proposed method thus builds upon the work of Evennou and Marx[11] and that of Wang et al[12]. However many modifications specific to implementation on a resource constrained device like an Android smartphone are made. The proposed device for implementation is the Samsung I9020 (also called the Samsung Nexus S which is co-branded with Google).

3.2 System Inputs

Given the choice of hardware, we are constrained to use only specific data sources for our algorithm. The sensors available on a typical Android Smartphone are:

1. A 3-axis accelerometer for measuring acceleration of the device.
2. A 3-axis magnetometer for measuring the surrounding magnetic field.
3. A gyroscope for providing angular velocity information.
4. A camera for providing visual information.
5. A Wifi Network Card with the ability to provide measurements of the received signal strengths. (RSSI)

Besides the sensors mentioned above, additional sources of information that we can make available to the system are:

1. Map information detailing the test environment.
2. A site survey of the test environment which provides Wifi location fingerprints augmented by orientation information of the reciever when the fingerprint was taken.

Characterizations of these data sources as well as other relevant parameters will be done as ground-work in Chapter 5.

3.3 Device Limitations

Being a resource constrained device, a number of limitations are imposed on software for these systems. Some of the ones most affecting the implementation are:

- Applications are restricted to a maximum memory utilization of 16 MB on older devices and newer devices get limits based on the size of their RAM. The Nexus S has a maximum limit of 32 MB, other devices usually have lower limits of 24 MB or 16 MB.
- The CPU usage of the application has to be controlled. Applications that use the CPU intensively for more than a few seconds are detected as misbehaving applications and the user is prompted to kill them.
- Sensor events are dropped if the event delivery thread is busy processing the last delivered event. This puts significant limits on the amount of inter sample processing that may be performed while

3.4 Background on Particle Filters

Particle filters belong to a class of Sequential Monte Carlo algorithms that depend essentially approximating a probability distribution function based on a Monte Carlo simulation of the system using observation samples. Particle filters have been known to be very useful for tracking in systems where joint probability distribution functions are either not completely known or hard to sample.

In this case, our problem of indoor tracking is essentially the determination of a latent variable x_t which represents the position of the user after we receive a series of observations $(o_0 \dots o_t)$ from the sensors on the Android device. Effectively, we wish to approximate the the probablity distribution $p(x_t|o_t, o_{t-1} \dots, o_0)$ at any time instant t so that we may determine the position of the device using the expectation function $E[x]$ on the probability distribution $p(x_t|o_t, o_{t-1} \dots, o_0)$ and we wish to use a particle filter to help us do so.

Particle filters are a huge topic in and of themselves. Thus, in the interests of brevity, the reader is referred to the excellent expositions of particle filters for tracking applications in [14] and to [13] for particle filters in the context of non-linear, non-gaussian Bayesian tracking.

3.5 Proposed Method

Keeping the availability of the inputs described above as well as the device limitations described in Section 3.3 in mind, a less computationally intensive method is proposed in Algorithm 1.

Algorithm 1 is very general. Also, though it is stated in a sequential manner, the it needs to be implemented in an event-driven fashion to allow for online tracking. The **ForEach** loop in the algorithm is essentially an event loop that depends on step detection.

Procedures to determine *FirstFix*, detect steps from sensor data, *MapSelect*, *ApplyDynamicalEquations*, and *PerfomRecovery* are explained below.

Comparison of the proposed method will be done with 2 other algorithms, also implemented on the smartphone:

1. A simple step based dead reckoning solution that directly uses sensor data.
2. A simple step based dead reckoning solution with a Wifi based nearest neighbour algorithm incorporated as a drift correction method.

These algorithms are being implemented as control algorithms for comparisons in order to provide suitable comparative information since maps and test environments differ substantially and make comparative analysis difficult.

Input: Sensor Events from device sensors
Result: An estimate of the location of the device at any time through the variable L
begin
  $L \longleftarrow FirstFix()$
  $P \longleftarrow N$ particles around L corrupted by gaussian noise
  foreach *stepEvent from sensors* **do**
    $P' \longleftarrow ApplyDynamicalEquations(P)$
    $P' \longleftarrow MapSelect(P', P)$
    if P' *is* $\emptyset$ **then**
      $R \longleftarrow PerfomRecovery(P)$
      if R *is not* $\emptyset$ **then**
        $P \longleftarrow R$
      end
    else
      $P \longleftarrow P'$
    end
    Resample set P uniformly to a fixed sized set and corrupt the new particles thus generated with gaussian noise
    $L \longleftarrow Mean(P)$
  end
end

Algorithm 1: High Level View of the System

3.6 First Fix

Dead reckoning refers to a method of position determination where the current location is estimated based on a known starting location and offsets from the known location measured via instruments.

For any dead reckoning algorithm, it is important to provide a good starting point. Low error in the starting location contributes to better performance. Our proposed method is based on the dead reckoning concept and thus requires a (reasonably) accurate starting point (also called a first fix).

In order to provide the starting point to the algorithm two approaches are proposed:

1. Directly choose the starting location on a map using the capacitive touchscreen

2. Select the starting location based on QRCode recognition using the onboard camera. The QRCodes were printed and pasted at specific locations in the test environment and the user is required to simply take a picture of it using the onboard camera.

Figure 3.1: A Sample QRCode

3.6.1 QR Codes

QRCodes are a kind of 2 dimensional machine readable code (an example of which is shown in Figure 3.1). The information encoded in this format can be read off by processing an image acquired from a camera.

QRCodes can contain a variety of content - from URLs to Contact information. They can also contain plain text.

For this application, we use the plain text information mode of QRCodes to provide encoding of a human-readable text representation of map points. This human and machine readable representation is achieved with the lightweight JavaScript Object Notation (JSON) representation. Details of the format chosen for this application are mentioned below:

QR Code information format

A typical format used for storing all the relevant information in a QRCode is shown in Figure 3.2. As is proper for any data serialization format, it stores a Version number and a Type field to distinguish itself from other data serialization formats. It also includes critical information about the point itself - the X and Y locations on the relevant map as scaled on a 0-1 scale. A full description of the fields is given in Table 3.1.

The fields are descriptive and quite human readable. However, because they are in JSON, a basic check of well-formedness can be made by programs. The data format retains extensi-

```
{
   "Version": 1,
   "Type": "MapPoint",
   "MapURL": "http://www.iitr.ernet.in/path/to/sample/map.png",
   "Scale": 0.010716161,
   "X":0.70625,
   "Y":0.16479166666666667

}
```

Figure 3.2: A sample data point encoded as text information in a QRCode

bility as it is able to accommodate additional fields which will be ignored by applications that are not built to expect the presence of those fields.

3.7 Noise filtering

It is well known that while measuring real world information, most sensors pick up noise from the environment that affects the readings of the sensed variables. The MEMS accelerometer present on a typical smartphone is no different.

In general, we assume that the accelerometer sensor noise is below a small threshold T when the device is static on a firm surface such as a table. A higher threshold Q is required when the device is being kept at the palm of a hand because of small noisy variations introduced by the palm itself.

The filtering process uses a simple clamping mechanism. The filter rejects a reading of the accelerometer based on the following constraints:

$$NoiseFilter(a_i) = \begin{cases} 0 & \text{if } |a_i| \leq Q, \\ a_i & \text{otherwise} \end{cases} \tag{3.1}$$

The values of T and Q are determined as part of the groundwork in Chapter 5.

The filtering process is very important for the step counting algorithms to detect accurate peaks.

3.8 Distance Estimation

The acceleration of the human body as part of the act of walking is small and very impulsive in nature. Unfortunately, the linear acceleration values produced by it are so small that they

Field Name	Field Description
Version	Constant value 1. May be incremented if additional fields are added to the format.
Type	Represents the type of QRCode waypoint sample that was just acquired. Type ”MapPoint” indicates that the QRCode represents a point on a Map. Alternative and additional information can be provided by choosing other Types. For example, a Type of ”WifiAP” could be used to provide location information about Wifi Access Points in the surroundings.
MapURL	This field always refers to a publicly available map associated with the environment where the QRCode was pasted. A png map type has been indicated, but any other resource type that a client is able to handle should be accepted.
Scale	Scale is an optional parameter of a MapPoint. It represents the scaling factor between distances on a map with distances in the real world. It can be omitted if it is expected that the client will have some other way of figuring out the scale of the map based on information encoded within the map itself.
X and Y	These are values encoded in the real range [0-1] which represent the location of the QRCode on a map referred by it. These values are used for providing a first fix to the reckoning algorithms.

Table 3.1: Explanation of the fields used in the QRCode Information Format

are virtually indetectible from the sensor noise of the MEMS accelerometer of the Android device. However, the act of putting a foot on the ground generates an impulse along the Z axis of the accelerometer which is clearly distinguishable in the sensor readings. Thus, we use the step count method described in [12] and [18] to detect motion of the user holding the device.

By relating the acceleration impulses to the size of a step, an inertial navigation system may be created. An empirical formula from [22] is used to estimate step sizes from the readings sensed by the accelerometer. Details of all the methods used to achieve the same follow:

3.8.1 Step detection procedure

Zero Crossings

This step counting method is described in [12] is based on a similar method devised for outdoor pedestrian navigation by [18]. It is a simple formulation that counts the number of zero crossings of a filtered version of the raw accelerometer signal which represent double the number of steps taken. However, this method, when implemented, generates step events at zero crossings which correspond to a body in motion. Since each step has to be associated with an angle of motion, it might be advantageous to ensure that step events are associated with points where the foot of the user is on the ground, leading to better stability of angle readings. Thus, the Peak and Valley hunting method is also proposed.

Peak and Valley hunting

The Peak and Valley hunting procedure is an alternate method being proposed for step detection. This method will detect the same number of crossings as the zero crossing method but will raise the step events whenever the foot of the user strikes the ground for the beginning of the next step.

In this method a 2-state machine is constructed according to Table 3.2. This method outputs a detected step at the first positive peak in the accelerometer sensor data that corresponds to the beginning of the next step. Also, since it only processes data at points where the first derivative of the accelerometer signal is zero and the accelerometer signal itself is non-zero, it can reduce computational effort for cases where zero crossing would perform considerable testing on account of being at the value 0, waiting for a transition. Additionally, the values of A_{max} and A_{min} that will be required later are updated only during these intervals, thus saving further computational resources compared the Zero Crossing method. The disadvantage of this method though is the additional lag introduced between a step being taken and its event being delivered to the application.

State	Accel. Value	New State	Action
q_0	Positive Peak (P) Detected	q_1	Update A_{max} if peak $P > A_{max}$. Output a step event if $flag = 1$. Reset $flag$ to 0.
	Other values	q_0	Ignore
q_1	Negative Trough (T) Detected	q_0	Update A_{min} if $T < A_{min}$. Set $flag$ to 1.
	Other values	q_1	Ignore

Table 3.2: State table of the step detection state machine

3.8.2 Step Size Estimation

Engineers from Analog Device have published an empirical relationship between acceleration values and step size in [22]:

$$Step - size = C \sqrt[4]{A_{max} - A_{min}} \tag{3.2}$$

The constant C is a scaling factor that is used as a constant of proportionality to scale the step-values to real world distances and A_{max} and A_{min} represent maximum and minimum acceleration values corresponding to the peaks and troughs associated with a step.

This equation is used to obtain an estimate of the step size when a user takes a step.

3.8.3 Determining the Training Constant

The training constant for each user was determined experimentally. Users were asked to perform a short walk between 2 QRCoded locations present in a straight line along a corridoor. Since, the QRCodes represent fixed locations in the real world, the actual distance between them can be found using simple Euclidean geometry. From simple step counting, we are able to figure out the number of steps taken. Also, by means of the step size formula mentioned in (3.2), we can estimate the distance travelled based purely on the accelerometer values. We can then find the training constant by simply plugging in the values into the following equation:

$$C = \frac{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}{\sum_{i=1}^{stepcount} \sqrt[4]{(A_{max_i} - A_{min_i})}} \tag{3.3}$$

Here,

C	is the training constant
$(x_1, y_1), (x_2, y_2)$	are the anchor point locations provided by the QRCodes
A_{max_i}, A_{min_i}	represent the maximum and minimum acceleration values corresponding to the i^{th} step.
$stepcount$	The number of steps detected by the algorithm in section 3.8.1

3.9 Dynamical Equations for the System

With the preliminaries now out of the way, we can develop the dynamical model of our proposed solution. The basic step update equation for the system can be written as:

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} + d_i \begin{bmatrix} -cos(\theta_i) \\ sin(\theta_i) \end{bmatrix} \tag{3.4}$$

Here,

x_i, y_i	represent location of the device after the i^{th} step
x_0, y_0	are the first fix values obtained via the methods of Section 3.6
d_i	the predicted step distance as per the step estimate equation (3.2)
θ_i	the angle associated with the detected step based on a magnetometer reading

These equations are written assuming a coordinate system for the map where the origin is at the top left corner of the map, the x axis being positive towards right and y axis positive downwards with $TrueNorth$ of the map pointing upwards. Effectively, equation (3.4) represents a raw dead reckoning solution. Unfortunately, these equations do nothing to counteract the biggest disadvantage of dead reckoning: unbounded error growth. The dynamical representation is also overtly simplistic because it fails to take into account other issues such as sensor drift and sensor bias. The magnetometer is the biggest culprit in this regard as it is highly sensitive to stray magnetic effects in the environment.

3.9.1 Modelling Orientation Sensor Noise

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} + d_i \begin{bmatrix} -cos(\theta_i + \vartheta) \\ sin(\theta_i + \vartheta) \end{bmatrix} \tag{3.5}$$

We modify the dynamical equation to add an additional parameter ϑ which is a random variable that represents random variations akin to white noise in the reading from the true value of the magnetometer. θ_i now represents the true angle associated with the step motion.

3.9.2 Accounting for Orientation Sensor Bias

Angular readings from the magnetometer often turn out to be biased from $TrueNorth$. This bias can creep in due to 2 different reasons - the first being specific, environmental magnetic fields which distort the actual detection of $TrueNorth$ in the system and the second being a bias that creeps in due to the way the user holds the smartphone in the palm of his hand and the offset thus produced. To take into account such offsets, we modify the dynamical equations as follows:

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \\ \theta_{b_{i+1}} \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ \theta_{b_i} \end{bmatrix} + \begin{bmatrix} d_i & d_i & 1 \end{bmatrix} \begin{bmatrix} -cos(\theta_i + \theta_{b_i} + \vartheta) \\ sin(\theta_i + \theta_{b_i} + \vartheta) \\ \theta_t \end{bmatrix} \tag{3.6}$$

$$\theta_t \sim \mathcal{N}(0, \sigma_{angle}) \tag{3.7}$$

In this modified version of the dynamical equations, we have added a slowly varying term θ_{b_i} that represents an explicit bias in the readings from the magnetometer at the time of the i^{th}. This bias value is updated at each step by altering it with a small gaussian noise of zero mean an a variance determined by the characteristics of the magnetometer. The choice of σ_{angle} was made on the basis of the groundwork done. Corrections for the sensor bias will be made when $MapSelect$ is applied to the particles by the overall algorithm.

3.9.3 Accounting for Varying Step Sizes and Missing Steps

The second issue at hand is step size variation and missing steps. Steps may be missed by the step detection procedures or phantom steps may be detected due to sensor noise or other events. Changes in step sizes due to changes in footwear or floor material as well as the leads and lags produced due to a slightly incorrect calibration constant also contribute to deviations of apparent distance travelled versus the apparent distance travelled. To account for this bias in step size detection, we introduce an additional parameter d_b in the dynamical system that accounts for the step bias. Thus, the new equations for the dynamical system are:

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \\ \theta_{b_{i+1}} \\ d_{b_{i+1}} \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ \theta_{b_i} \\ d_{b_i} \end{bmatrix} + \begin{bmatrix} (d_i + d_{b_i}) & (d_i + d_{b_i}) & 1 & 1 \end{bmatrix} \begin{bmatrix} -cos(\theta_i + \theta_{b_i} + \vartheta) \\ sin(\theta_i + \theta_{b_i} + \vartheta) \\ \theta_t \\ d_t \end{bmatrix} \tag{3.8}$$

$$d_t \sim \mathcal{N}(0, \sigma_{step}) \tag{3.9}$$

In this representation, d_{b_i} is a slowly varying bias variable on the step size and the gaussian variance used during each update step is d_t. The determination of the variance of d_t again depends on sensor to sensor and is evaluated empirically for our system to be close to $step\_size/25$.

Note that the equations mentioned in Sections 3.9.2 and 3.9.3 will work only if the map information is dense enough to eventually eliminate groups of particles with incorrect orientations and step drifts. Otherwise, the only effect of maintaining the bias variables will be to slightly increase the variance of the variables x_i and y_i. An explanation of how these bias variables actually help determine overall angle and step bias is provided in Section 3.11

3.9.4 Ensuring particle diversity

The presence of the variables θ_t and d_t represents an intentional introduction of randomness into the system to ensure particle diversity. For example, if two particles start out with the same state vectors, say $[x_i y_i \theta_{b_i} d_{b_i}]$ and $[x_j y_j \theta_{b_j} d_{b_j}]$, they will in the immediate next time step tend to diverge due to possibly different samples of the random variables θ_t and d_t. So, when evolved over a large number of steps, similar samples will take slightly different paths through the state space, thus ensuring particle diversity of the particle filter. Note however, that the choice of magnitudes of the variance of θ_t and d_t is critical. If the values chosen are too small, states that were possible physically are not achieved by the particle filter and if the values chosen are too large, computational effort will be wasted for unachievable states. For our case of a highly resource constrained machine, this is very undesirable for we wish to maximize the effectiveness of our particles. In this regard, it is also very advantageous if we have a highly detailed map of the area for unachievable states are quickly pruned and computational effort is redirected towards states with more likelihood. The integration of map information is described in the subsequent section.

3.10 Integrating Map Information

Map information can be integrated into the system in a number of ways. The simplest way to use map information as a selection function which is similar to [12]:

$$MapSelect(p_{i+1}, p_i) = \begin{cases} 1 - P_{wall} & \text{if } p_i + \alpha(p_{i+1} - p_i) \text{ for } \alpha \in [0,1] \\ & \text{does not cross a wall} \\ P_{wall} & \text{otherwise} \end{cases} \tag{3.10}$$

Here,

p_{i+1} and p_i	are tuples corresponding to map locations (x_{i+1}, y_{i+1}) and (x_i, y_i) respectively.
P_{wall}	represents the probability of selection of the particle if it crosses a wall.
α	is a parametric variable used to represent a line between the two points p_{i+1} and p_i

Wang et all[12] suggest that a P_{wall} of 0 be used as such a motion is impossible. Though simplistic, this approach works well by swiftly removing unreachable particles. It is very light computationally too. However, given our very limited budget for particles, it might not be prudent to remove a particle simply because it was crossing a wall very close to a door. Thus, more advanced map integration techniques may well be envisaged with P_{wall} varying based on the distance of the intersection point from the closest door in the wall.

3.11 Determination of overall orientation and step biases

The integration of map information with the dynamical equations of the system via a selection function $MapSelect$ allows us to actually determine the extent of orientation and step bias. The idea behind it is simple. However, before explaining it, let us define the following two terms:

$$\theta_b = \frac{1}{N} \sum_i \theta_{b_i} \tag{3.11}$$

$$d_b = \frac{1}{N} \sum_i d_{b_i} \tag{3.12}$$

Here,

N	is the total number of live particles in the system
d_{b_i} and θ_{b_i}	retain their meanings from the previous sections
θ_b	represents overall system orientation bias at the i^{th} stage
d_b	represents overall system step bias at the i^{th} stage

When no walls exist in the vicinity of the particles, map information is sparse and few

particles are eliminated because of it. Thus, we have insufficient feedback from our system to determine the value of orientation bias (θ_b) as all positions are equally valid. However, in a confined space like a doorway or a corridoor, there exist only a few valid range of angles over a sequence of steps. Thus errant particles will be swiftly rejected by $MapSelect$ and we will get a highly accurate estimate of the degree of orientation bias present in our system.

The justification for step bias (d_b) lies similarly in the fact that while moving along corridoors, we will have accumulated a certain degree of step drift. This drift ensures that we are slightly uncertain of our exact position along the corridoor after walking through it for a long time as we have not received any feedback from $MapSelect$ along our direction of motion. Thus, some particles representing our location will be further along the corridoor than others and we have no way to choose between them other than by treating them at par. However, whenever we make a sharp turn on the map, the presence of walls and other obstacles immediately before and after the junctions on the map will cause a number of particles to be rejected by $MapSelect$. The values of d_{b_i} remaining represent the net bias in the step size of the system. Thus, their mean d_b is a good representation of the step bias of the system.

3.12 Recovering from particle insufficiency

Particle insufficiency is a weakness of particle filters. It arises if no valid successor states p_{i+1} can be found from the current set of particles p_i when the observation o_i is taken into account and applied to the dynamical equations of the system. (That is, all successor states p_{i+1}, have $MapSelect(p_{i+1}, p_i) = 0$). This possibility is further accentuated in resource constrained systems like ours which have a hard limit on the maximum number of particles that can be updated between two steps (since the algorithm is working in an online fashion with a location estimate being available after every step).

The insufficiency of particles can be handled by a number of ways. The simplest one is: increase the number of particles being updated at each step and ensure enough particle diversity so that you always have a few particles that survive incorrect decisions of the particle filter.

Widyawan [15] suggests an alternative method: maintaining particle history information so that an incorrect decision of the particle filter can be rolled back a few steps in the past and an alternative decision can be made that does not cause the particle filter to suffer particle insufficiency. However, the method suggested is expensive in terms of memory usage and computational cost.

The recovery algorithm ($PerformRecovery$) being proposed is stated in Algorithm 2.

There are a few interesting things to note about the recovery algorithm. The first being that

Input: A set of particles P such that $ApplyDynamicalEquations(P)$ yields $\emptyset$, σ_X and σ_Y are the positioning errors of the Wifi based positioning system
Result: A new set of particles R which approximate the location of the device or $\emptyset$

begin
  Try to recover by finding particles in the surroundings by random sampling
  $retryCount \longleftarrow 0$
  while $retryCount < MaxRetries$ **do**
    for j *in* $[1..N]$ **do**
      $r_j \longleftarrow (\mathcal{N}(0, \sigma_X), \mathcal{N}(0, \sigma_Y))$
      $selectParticle \longleftarrow false$
      foreach $p_i \in P$ **do**
        if $MapSelect(r_j, p_i)$ **then**
          $selectParticle \longleftarrow true$
          break
        end
      end
      if $selectParticle$ *is* $true$ **then**
        $R \longleftarrow R \cup \{r_j\}$
      end
    end
    $R' \longleftarrow ApplyDynamicalEquations(R)$
    if R' *is not* $\emptyset$ **then**
      return R
    end
    $retryCount \longleftarrow retryCount + 1$
  end
  We have failed to find suitable particles by random sampling, fall back on Wifi
  $wifiLocation \longleftarrow PerformWifiPositioning()$
  for j *in* $[1..N]$ **do**
    Corrupt the wifi location with gaussian noise to approximate the error
    $r_j \longleftarrow wifiLocation + (\mathcal{N}(0, \sigma_X), \mathcal{N}(0, \sigma_Y))$
    $R \longleftarrow R \cup \{r_j\}$
  end
  $R' \longleftarrow ApplyDynamicalEquations(R)$
  if R' *is not* $\emptyset$ **then**
    return R
  else
    We've failed to find suitable particles, return failure and handle it by skipping the step information.
    return $\emptyset$
  end
end

Algorithm 2: Recovery algorithm for particle insufficiency

it retries finding suitable particles. This is an optimization as we don't want to spend too much effort at recovering unless we really have to since the inner algorithm has an O($N \times |P|$) complexity. In case we fail to recover directly from random sampling around the location, we retry with Wifi Positioning data. If that fails too, then we return an empty set. The system handles this failure by skipping the step event (discarding it as possibly invalid sensor readings).

3.13 Summary

The above algorithms represent a lower complexity particle filter that is proposed to be suitable for implementation on resource constrained smartphones. The performance of the proposed methods is analyzed in Chapter 6.

4
Implementation Details

4.1 Background on Android Application Development

4.1.1 The Android OS

The Android OS was initiated as an open source project by Google to make application development easier for mobile devices. In order to do so, Google engineers created a customized Java Virtual Machine (called Dalvik VM) with resource saving features to make running Java applications on embedded devices feasible. A large section of the Java Standard Library was then bolted onto the JVM by leveraging the open source Apache Harmony project.The typical structure of an Android Application built on the Android OS is shown in Figure 4.1.

4.1.2 Application Development Tools

Applications for Android are usually written in Java and are run on the custom Dalvik VM. Embedded programming is usually difficult because of limited visibility of the way the code runs on the target device (a smartphone). To make development of apps for Android easier, custom tooling for Android has been developed that integrates with the Eclipse IDE toolchain. The tools help to interface with a development smartphone running an Android OS. They provide a direct install-uninstall mechanism for applications under development and they also

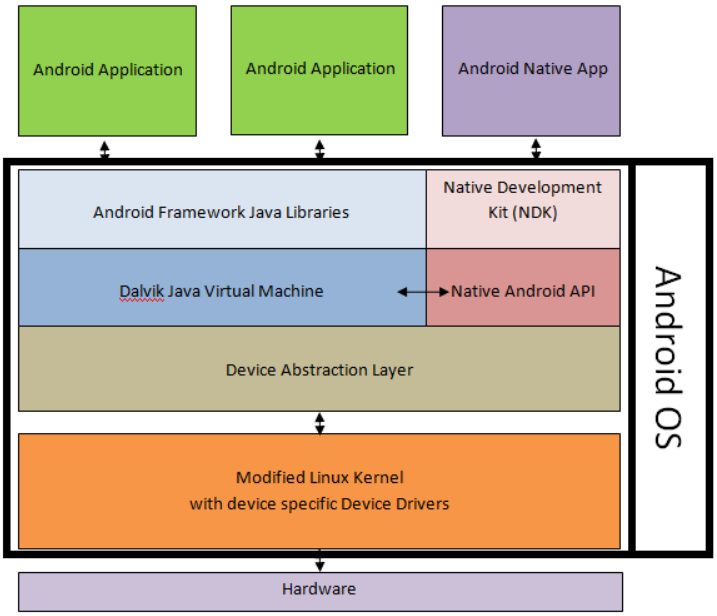

Figure 4.1: Typical Structure of Android OS

provide a debug bridge - called the ADB through which the Application code running on the smartphone may be debugged via debugging tools on the host computer. A logging mechanism is also provided wherein application logs running on the device may be made visible in realtime on the host computer through the ADB. These tools go a long way in making development easier and help the programmer detect root causes of bugs faster.

4.2 The Android Application Lifecycle

Android applications are very different from the normal system executables that we are used to programming. Executables once launched have complete system control during execution. They demand resources and release them at will depending on how they have been programmed. They are pre-empted and resumed transparently by the Kernel scheduler. However, such a mode of execution is unsuitable for applications built for the mobile device. These applications need to be aware of their internal state at all times. The user may pre-empt any application at any time and the system may suspend or even kill applications without warning to reclaim resources. Applications are therefore expected to be written to follow an event machine lifecycle and are expected to save their state whenever they get pre-empted. The lifecycle of a typical Android application is shown in Figure 4.2.

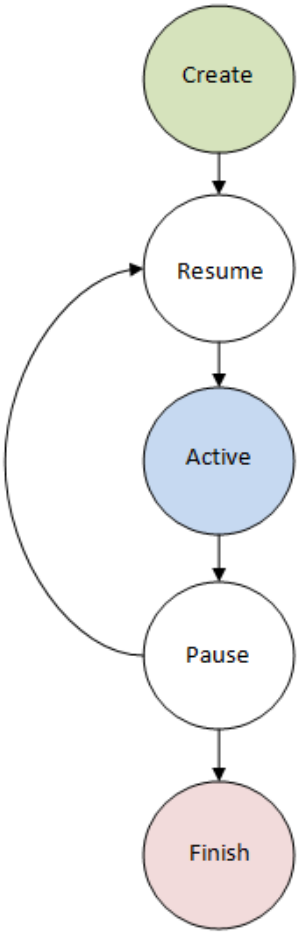

Figure 4.2: Android Application Lifecycle (Simplified)

The application typically starts off the 'Created' state, progresses to the 'Resumed' state. It may oscillate between the 'Resumed' and 'Paused' states several times during the course of the execution lifetime of the application and then it finally transitions to the 'Finished' state when it is killed by the system either automatically or in response to a user action. All processing in the application is done either in response to a User action, a state change in the application lifecycle or in response to System messages delivered asynchronously to the application via *BroadcastReceivers* or *Intents*. (*BroadcastReceiver* and *Intent* are important classes in the Android framework).

4.2.1 Sensor Event Delivery Mechanism

Sensor events are not delivered synchronously to the system in response to a function call as this would require a polling mode of operation which is a great drain on system resources - especially the battery. Instead, system events are delivered asynchronously to the application code in a thread created by the Android OS and which is owned by the Android Sensor Manager API. Thus any code written to handle sensor event delivery must be multithreaded aware and should be able to handle race conditions. The act of registering and unregistering for sensor events with the Sensor Manager API still remains a synchronous operation though. Figure 4.3 clarifies via a diagram the nature of the sensor event delivery mechanism and the synchronous and asynchronous operations involved.

4.3 System Architecture

As is clear from the previous section, the Android OS expects applications to be event-driven. In our case, events are being generated by the user not through direct interaction with the application via the UI but through indirect interaction via the sensors onboard the device. This naturally leads to a bottom-up design of the system on the lines similar to protocol stacks seen commonly in the case of Computer Networks. The analogy being drawn here is that a packet receive event in the case of a protocol stack is similar to a sensor event generated for our system.

The complete layered structure of the application is shown in Figure 4.4.

At the bottom we have the **Sensor Unification Layer**. This layer is important as the Android API separates the event delivery mechanism for a Wifi SCAN event and the event delivery mechanism for a sensor sample event and a unification mechanism is thus required. The rationale behind the separation of concerns in the Android API is that sensors are passive environment samplers while the Wifi NIC is an active communication device and is not

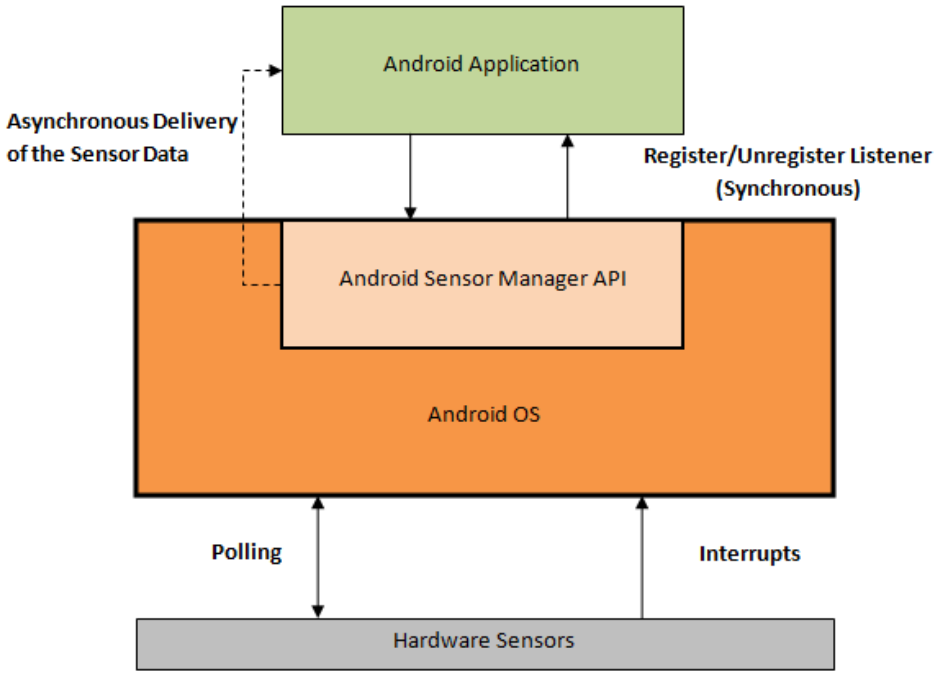

Figure 4.3: Synchronous and Asynchronous operations involved in registering for sensor events

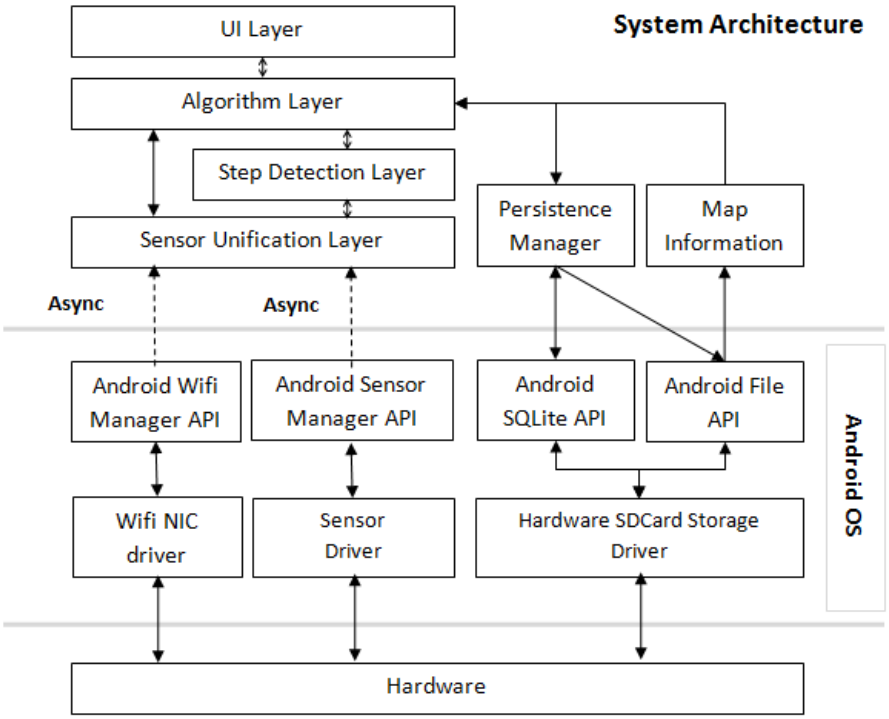

Figure 4.4: System Architecture

typically intended to be used as a sensor. The **Sensor Unification Layer** works very hard to unify Wifi `SCAN_RESULTS_AVAILABLE` event results and sensor sample results into a single uniform events system to which a higher layer can subscribe by registering a callback object. Whenever a sensor event takes place for which higher layers have requested notification, all registered callbacks for that event are executed by this layer. Thus, higher layers can request access to sensor events by just registering an appropriate callback with this layer. The unification layer takes care of interfacing with the Android SensorManager API and the Android WifiManager API and exposes lifecycle methods - `pause` and `resume` which free up higher layers from performing appropriate registrations and de-registrations of sensor event handlers when they intend to go to sleep. The class that provides the API for the higher layers is the *SensorLifecycleManager* class and the critical class for implementation of this layer is the *HWSensorEventListener* and *WifiScanEventListener* classes.

The layer immediately above the **Sensor Unification Layer** is the **Step Detection Layer**. This layer is responsible for accepting accelerometer and magnetometer sensor events from the device and notifying callbacks that a step event has taken place. The callbacks receive notification of the step event along with parameters which indicate the `step size`, the `step angle` and a `timestamp` of the event.

The **Step Detection Layer** essentially feeds the **Algorithm Layer**. The **Algorithm Layer** may depend directly on the step information to implement tracking (as is in the case of the Simple Dead Reckoning solution and our proposed method), or it may register it's own callback for other events such as Wifi `SCAN_RESULTS_AVAILABLE` events from the **Sensor Unification Layer** to implement tracking algorithms that depend on Wifi information. In case the **Algorithm Layer** requires access to a Wifi Site Survey data, it requests a reference to a DB storing such data from the *PersistenceManager* class. Utility modules developed at the **Algorithm Layer** allow population, display and modification of samples in the survey database. Map information is brought in by reading a Map downloaded to the Android device. The compressed version of the image is expanded and cached for access by the Algorithm layer.

The **UI Layer** is built using HTML5 technology as well as native Android Views and simply taps into the **Algorithm Layer** to get interesting values regarding the current position of the device or the tracked path. This access can be made synchronous if done using Android views or it can be made asynchronous by exposing the algorithm layer via a JavaScript object to a webpage running in a sandboxed browser. Currently, the JavaScript route has been chosen as it allows us to remove a UI update from the sensor processing loop. Algorithm data is tapped using a timer that refreshes the UI at 1 Hz.

Java packages and the layers to which code in them belongs is stated in Table 4.1.

Package	Classes	Layer
in.ernet.iitr.divyeuec.ui	*LaunchActivity*, *DeadReckoningActivity*, *WifiSiteSurveyActivity*...	UI
in.ernet.iitr.divyeuec.algorithms	*DeadReckoning*, *WifiSnappedDeadReckoning*, *ParticleFilteredReckoning*	Algorithm
in.ernet.iitr.divyeuec.algorithms	the callback functions of *DeadReckoning*	Step Detection
in.ernet.iitr.divyeuec.sensors	*SensorLifecycleManager*, *WifiScanResults*	Sensor Unification Layer
in.ernet.iitr.divyeuec.db	*PersistenceFactory*, *FileDB*, *SQLiteDB*, *LocationFingerprint*	Persistence Manager

Table 4.1: List of Classes and their Corresponding Layers

4.4 System Implementation of the Lifecycle

The Android Application Lifecycle referred in Section 4.2 primarily deals with high level user actions such as going idle, switching off the screen, exiting the application etc. We preempt idle states from occurring in the application by acquiring a *WakeLock* from the `POWER_MANAGER` system service. This allows us to keep the display running even when no user interaction has taken place with the device (as is the case when the system is tracking and the user is just observing the screen). The operations that involve the highest complexity in the lifecycle are *Pause* and *Refresh*. In these operations, the application is supposed to let go of all Sensor resources. Since we have a single class managing all the Sensor data, it exposes the `pause` and `resume` methods which are called by the UI layer whenever the UI is signalled to pause or signalled out of the paused state and asked to resume.

4.5 Computational Complexities

Everything below and including the *Step Detection Layer* needs necessarily to be of the lowest possible computational complexity in order to keep the application responsive to sensor data being generated asynchronously. There is a slight leeway above the step detection layer to use algorithms that require more computational resources as the step events are not generated as frequently.

4.6 Extensibility

The architecture of the system is very sound. In fact, the layered system made it very easy to develop logging and graphing utilities for the device without causing major modifications to

parts of the system.

A simple module, developed at the appropriate layer and exposed to the UI layer could be built independently and used by using only the features of the lower layers.

Separation of concerns was not an issue in this design as no layer below the UI layer was concerned about how to expose the data to the user.

Loose coupling ensures that there is ample scope for extending the application via modules that can replace the functionality for entire layers. In fact, the entire system can be turned into a simulation platform by just replacing the **Sensor Unification Layer** with a class that simulates sensor events from data recorded in files and replicates the same interface as the SensorLifecycleManager.

5 Groundwork

While implementing any system, it is imperative to understand the data at hand so as to make correct analytical decisions and set correct parameters for the algorithms used. To characterize the input data sources, set parameters of the algorithms and optimize the resulting implementation, groundwork was done.

5.1 Hardware

The hardware on which the system is implemented is the Samsung Nexus S (cobranded with Google) running Android 2.3.4. The official specifications of the device can be obtained from the official website of the device[23]. A short listing of the sensors present on the device is given below based on how the device reports the sensors via the software APIs:

1. KR3DM 3-axis Accelerometer
2. AK8973 3-axis Magnetic field sensor
3. AK8973 Orientation sensor
4. GP2A Light sensor

5. GP2A Proximity sensor
6. K3G Gyroscope sensor
7. Gravity Sensor
8. Linear Acceleration Sensor
9. Rotation Vector Sensor
10. An 802.11 b/g/n compatible Wifi NIC

It may be noted that of these sensors, the Gravity sensor, the Linear Acceleration Sensor and the Rotation Vector Sensor are derived sensors. They depend on filtering the raw data sensed via the accelerometer and the magnetometer to generate their sensed values. In essence, the Gravity sensor is a low pass filter over the accelerometer values (since gravity is essentially stable for the device at a particular orientation), the Linear Acceleration Sensor is a high pass filter over the accelerometer values (the high frequency changes in acceleration are presumed to occur due to device motion and thus correspond to linear acceleration of the device) and the Rotation Vector Sensor is a composite sensor that fuses Gravity information derived from the 3 axis Accelerometer and the magnetic field information derived from the 3 axis Magnetic field sensor to orient the device in the 3 Dimensional World Coordinate space. The actual method used to do so is described in the API documentation [24] and further discussion is out of scope for this thesis.

5.2 Accelerometer

The MEMS accelerometer on the Nexus S was subject to simple tests to determine its noise floor and the degree of separation between signal and noise. Figure 5.2 shows a plot of the accelerometer's raw output along the Z axis for a stationary device kept on a table. The characteristics of the accelerometer signal are summarized in Table 5.1. As can be clearly observed, the signal degrades while in the palm of a user who is standing.

The noise level for the accelerometer was determined to be always less than $0.6m/s^2$. The acceleration peaks corresponding to regular walk steps usually lie around $2.0m/s^2$. To allow for a reasonable noise margin and provide sufficient cushion for additional noise introduced due to the dynamic nature of walking, we choose a threshold of $1.3m/s^2$ which is the mean value of the two peak values. If the absolute value of the Z-axis acceleration sample is less than this threshold, then the sample will be clamped to zero. For a smartphone with a similar

Figure 5.1: Google Nexus S manufactured by Samsung

	$a_{x,table}$	$a_{y,table}$	$a_{z,table}$	$a_{x,hand-held}$	$a_{y,hand-held}$	$a_{z,hand-held}$
Min	-0.180217	-0.209914	-0.352813	-0.281559	-0.238432	**-0.571722**
Median	-0.002995	0.001779	0.003011	0.004029	-0.005922	0.003617
Mean	-0.002065	-0.002809	0.004762	0.007291	-0.003648	-0.004827
Max	0.206112	0.211490	0.306364	0.577652	0.207061	**0.595963**
Std. Dev	0.06985715	0.06902411	0.08939619	0.1100473	0.08232208	**0.1603238**

Table 5.1: Characterization of the Accelerometer (values in m/s^2)

accelerometer sensor but with different noise characteristics, the values of the threshold can be varied accordingly. For our purposes, the threshold Q is chosen as $1.3m/s^2$.

5.3 Magnetometer

The Nexus S comes with a built in MEMS magnetometer. It measures the local magnetic field strength in μT.

In practical tests, the performance is reasonable but the magnetometer has a lot of sensor noise and suffers from sensor bias. For example, rotation of the smartphone through large angles introduces bias in the readings from the magnetometer. Similary, the magnetometer shows a small offset when the device is turned through 360 degrees.

Figure 5.3 shows the behaviour of sensor readings under the 2 circumstances when the device is on a table and when it is held in the palm of the hand of a user.

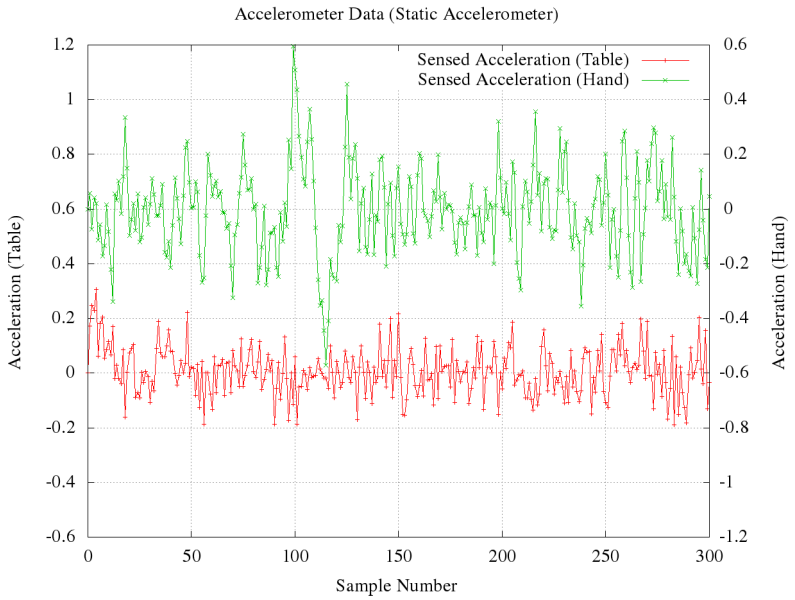

Figure 5.2: Accelerometer Z axis readings when device stationary

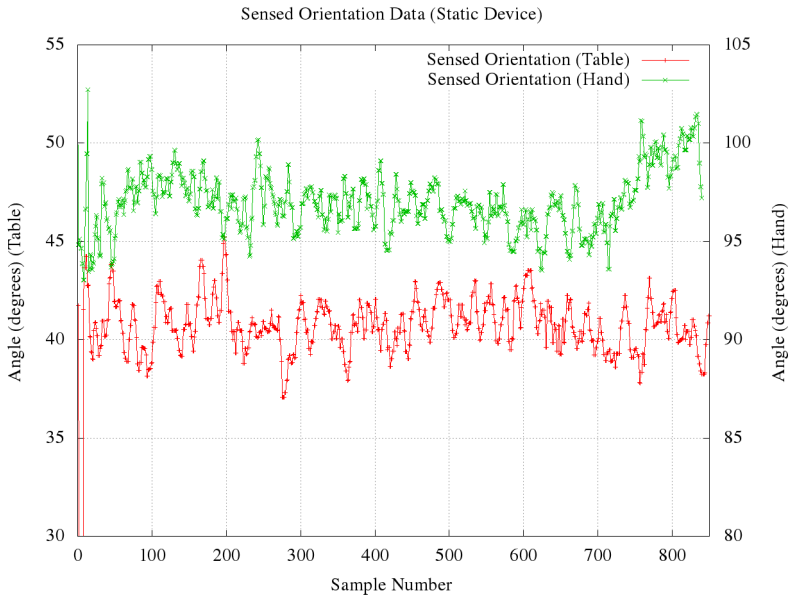

Figure 5.3: Derived azimuth angle when smartphone on table

	θ_{table}	$\theta_{standing}$
Maximum Deviation	8.138436	9.688619
Std. Dev.	1.234318	1.521967

Table 5.2: Characteristics of a Stationary Magnetometer (degrees)

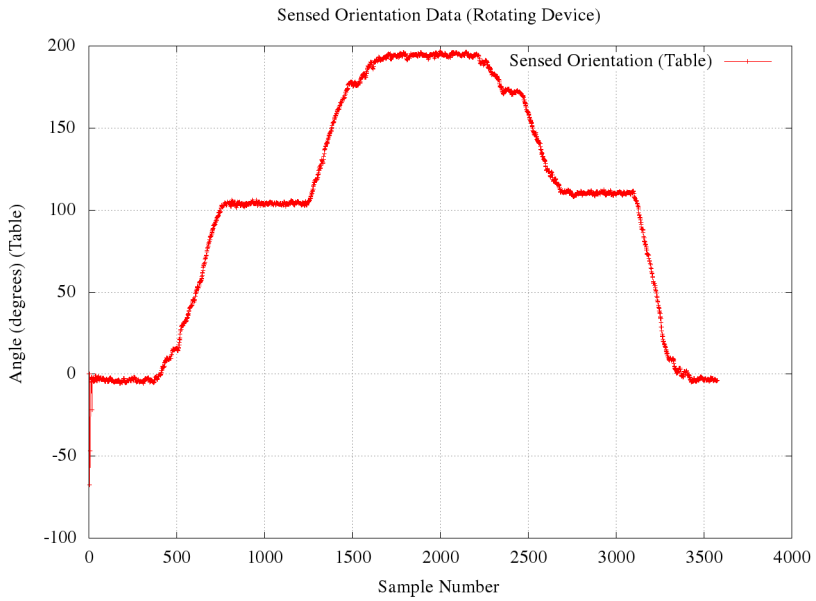

Figure 5.4: Rotation of the phone through 180 degrees (90 degree turns)

These values are required to determine choice of parameters for the particle filter that will be introduced later.

5.3.1 Bias study

Performing rotations in a slow and steady manner yields good results with little or no bias and sensor lag. See figure 5.4 which shows how the angle measurements behave when the rotated slowly and steadily. There is only a slight sensor lag and bias visible.

5.3.2 Effect of motion

Although the sensor measurements are stable when the device is on a table, the measurements go haywire when the device is in a human's hand. See figure 5.5 - a human is walking along a corridoor and he walks back. The angle measurements of the return trip are offset by 180 degrees and the two sets of measurements are compared. You can easily see a large variation

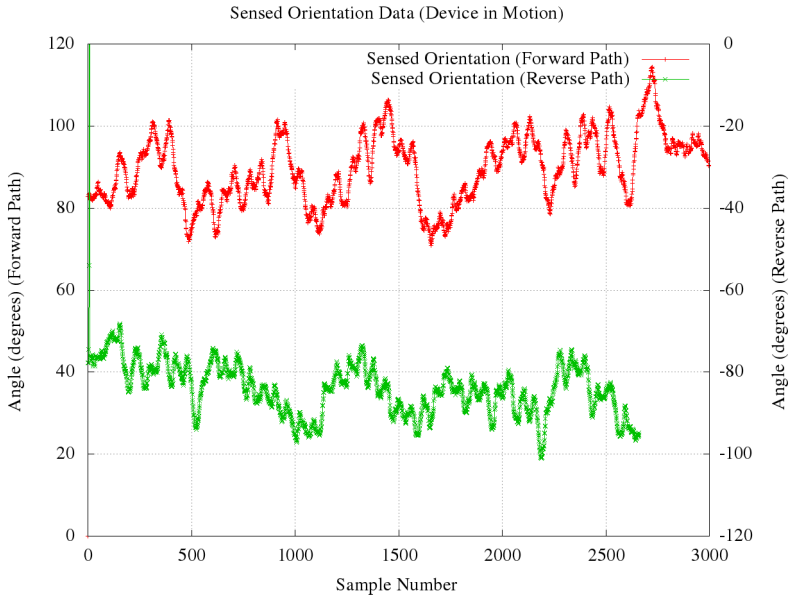

Figure 5.5: Angle measurements when moving in opp directions.

	Forward Path	Return Path
Angular Range (Max - Min)	43.48589	32.7897
Standard Deviation	8.307481	5.955799

Table 5.3: Effect of Motion on Orientation Values

in the angles due to the motion of the human and the dynamic nature of the environment. Sensor lag and long term sensor value drift is evident in the graph. This dynamic variation of the measurement of the angle from magnetic north introduces error in the inertial navigation system.

The values of Table 5.3 indicate the maximum range of values over which the angle varies during a walk along a straight corridoor. These values are used to set the parameters θ_t in the algorithm. The angular range is large but the middle 50% of the values lie between 11 to 13 degrees apart which corresponds with the standard deviation values measured.

5.4 Test Environment

The location chosen for performing all experimental verification of the proposed work was Floor 3, Wing 7 and 8 of Ravindra Bhawan at the IIT Roorkee campus. The environment is suitable for this purpose due to a number of reasons:

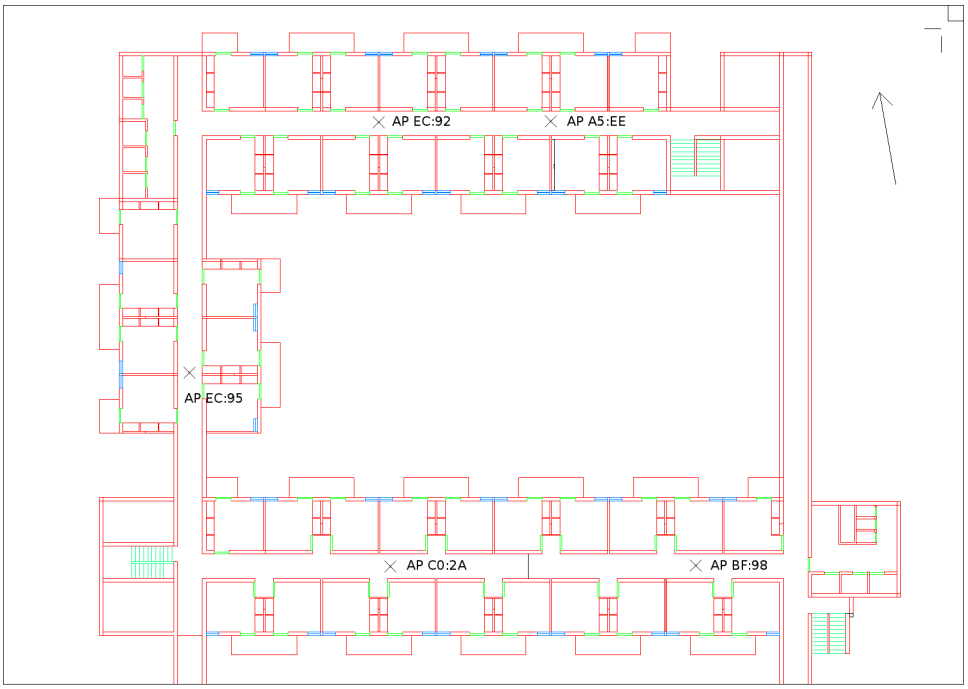

Figure 5.6: A map of the experimental environment

1. There are a large number of Wifi Access Points in close proximity installed on the premises.

2. Wifi Access Points of neighbouring wings and floors provide diversity to the location fingerprints.

3. Long corridoors of around 30-32 metres each provide long stretches of similar environment for evaluation of algorithm performance over larger distances.

A map of the location is shown in Figure 5.6. The red lines indicate walls, blue lines indicate windows and green lines indicate doors. Wifi Access Points are marked with stars on the map. This map has been used for all the experiments detailed below and while determining the results as stated in Chapter 6.

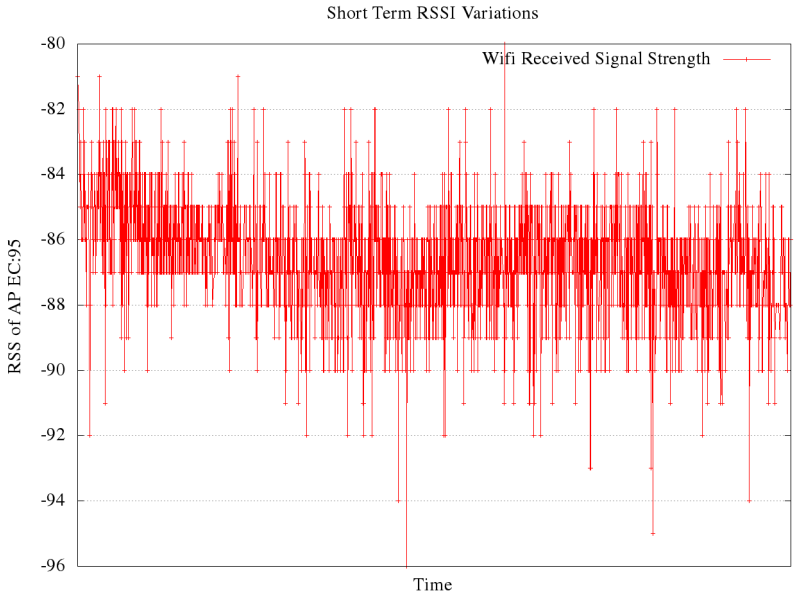

Figure 5.7: Variation of RSSI for AP EC:92

5.5 Wifi Signal Strength Distribution

Wifi signals are used in our proposed system during the recovery process. Other systems and the control algorithms use Wifi for correcting positioning drift. Hence, it is important to understand the strengths and limitations of Wifi based positioning. In order to do so, we need to first characterize the Wifi signal.

The behaviour of the wifi signal strength distribution for a stationary laptop and smartphone was analyzed to characterize the input wifi signal.

Short term duration

Figure 5.7 shows the variation of signal strength of AP EC:95 for a smartphone placed within a room in a Non-Line Of Sight configuration.

5.5.1 7 day study

The Wifi Access Point RSSI values were monitored over a 7 day period (Jan 5 2011 - Jan 13 2011) from room S-152 and the resulting signal strength data was analyzed.

As you can see in Figure 5.8, the signal strength distribution for the closest access point is highly non-gaussian. This concurs with the conclusions of Kaemarungsi and Krishnamurthy

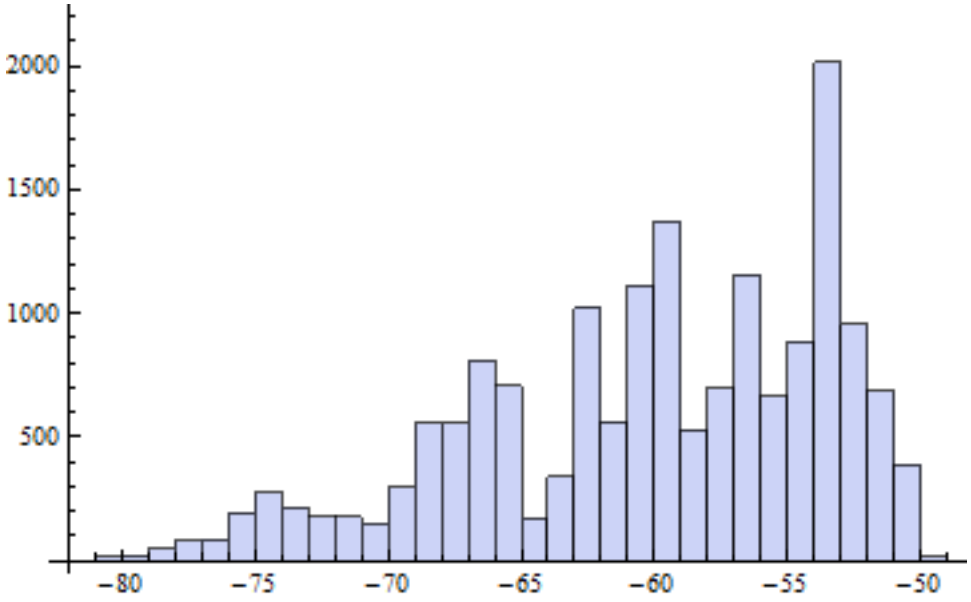

Figure 5.8: Distribution of RSSI values AP EC:92

in [1].

In contrast, Figure 5.10 doesn't show such a pronounced non-gaussian nature whereas Figure 5.9 shows more leanings towards a non-gaussian distribution.

Table 5.4 states the features of the signal with respect to 3 different APs. However, this table does not state the entire story.

Figure 5.11 is the time domain plot of the signal strength samples smoothed as a moving average over 300 samples as received by a fixed location. It is quite evident that the variations in the signal strength are highly localized with significant peaks and valleys. Plots for other APs show this kind of clustering behaviour too but to a lower extent. Given the uncorrelated nature of wifi signals, it is highly likely that signal variations will not move in a correlated fashion and thus even location fingerprinting based systems will find it very difficult to create a distance function that will be able to estimate the location of a device reliably from the RSSI values given that the survey database will always be subsampling the signal.

Thus it is quite evident that the input wifi signal is a highly noisy and unreliable source of information.

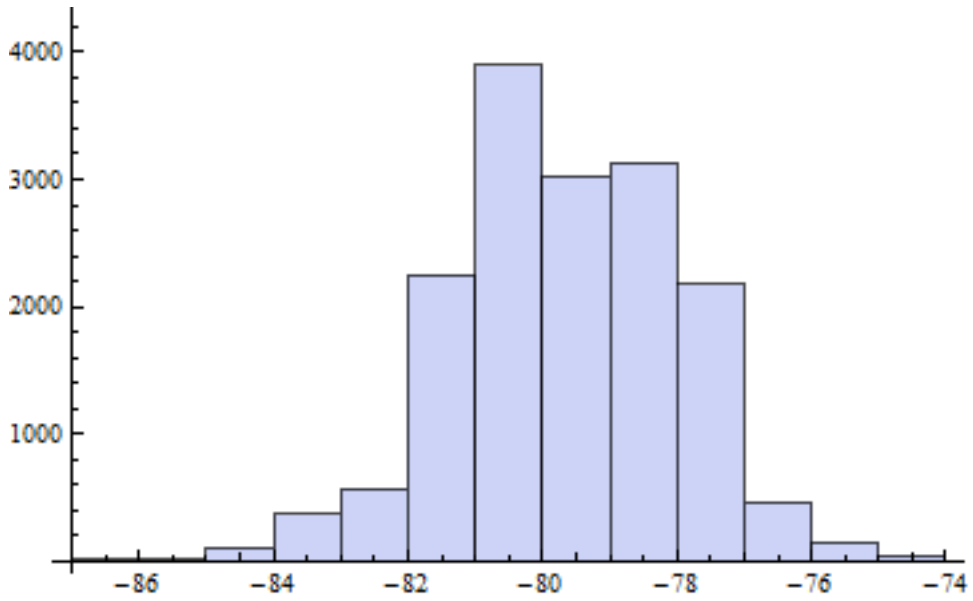

Figure 5.9: Distribution of RSSI values AP EC:95

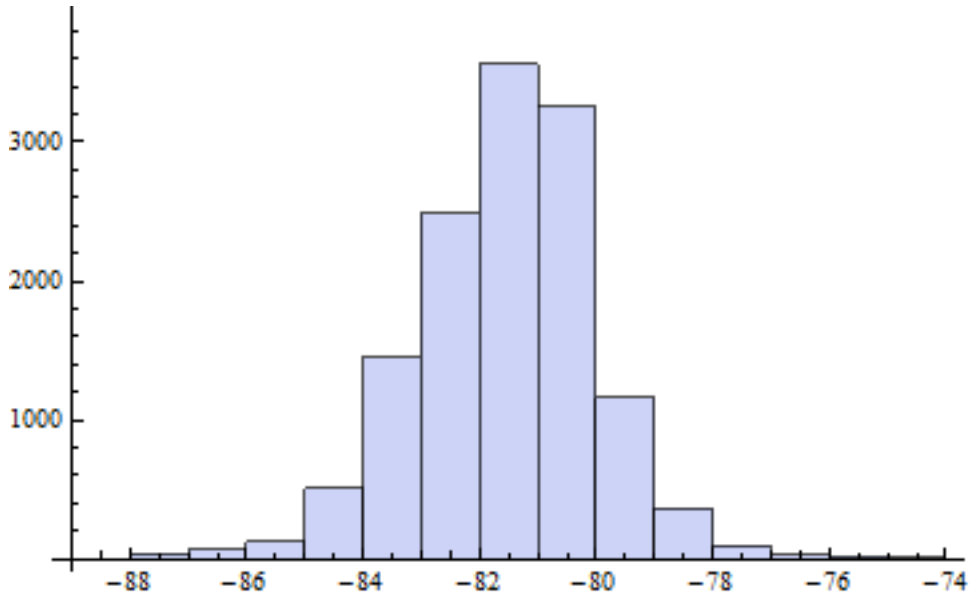

Figure 5.10: Distribution of RSSI values for the AP A5:EE

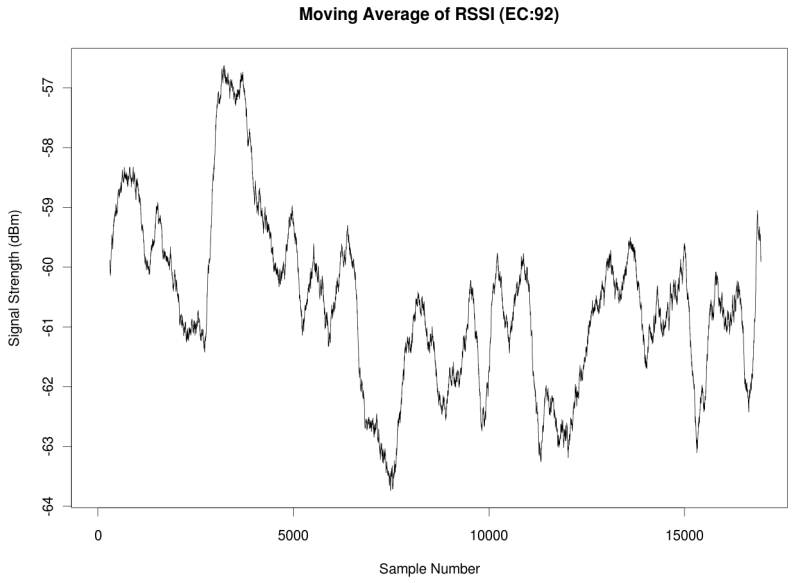

Figure 5.11: 3 hour moving average of RSSI values for the 7 day survey

	Min	1^{st} quartile	Median	Mean	3^{rd} quartile	Max	NA	SD
AP EC:92	-82.00	-66.00	-60.00	-60.62	-55.00	-49.00	27.36%	6.638592
AP EC:95	-87.00	-81.00	-80.00	-80.16	-79.00	-74.00	30.64%	1.691343
AP A5:EE	-89.00	-83.00	-82.00	-82.05	-81.00	-73.00	43.48%	1.580761

Table 5.4: Wifi Signal Analysis

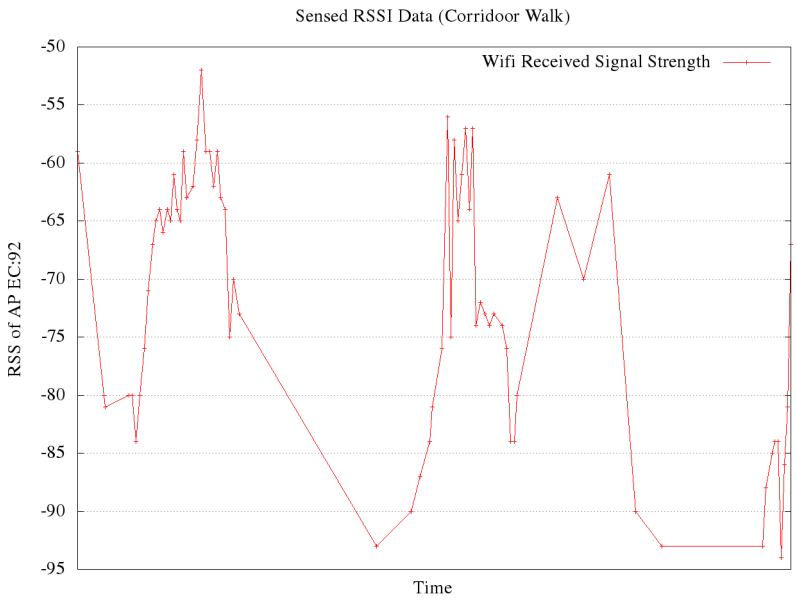

Figure 5.12: AP EC:95 RSSI: Walk up and down the corridoor

5.6 Effect of motion on signal strengths

To analyze the effect of motion on the RSSI values from the APs, a simple walking test was done along a corridoor. The variation of RSSI values seen from the AP in the corridoor is shown in 5.12.

5.6.1 Inferences

From the figure, it can be clearly seen that orientation with respect to AP is important. There is significant noise in the signal even though the AP is in the line of sight. Thus retaining orientation data with respect to an AP is an important criteria while surveying the test environment to collect location fingerprints.

5.7 Generating a Location Fingerprinting Dataset

A Wifi Survey was carried out in the test environment to generate a location fingerprint dataset. To ensure uniform sampling of the environment, a $1m \times 1m$ grid was set up on the ground and Wifi RSSI readings were taken at 8 different orientations per point on the floor. Figure 5.13 shows a snapshot of a small area of the test environment. Grid points have been marked out

Figure 5.13: Picture of the $1m \times 1m$ grid drawn on the floor

on the floor. Wifi RSSI samples were taken at these grid points along 8 different orientation values. Some samples, however, could not be taken due to obstructions present at the site.

5.8 Testing Wifi Positioning Accuracy using KNN

A KNN based location system was implemented to test location accuracy for positioning using Wifi Signal Strengths. This was work done as part of the Masters project in the same test location. The results of the project are included here. Although the project was done using a laptop instead of a smartphone and with a different set of survey data, it is expected that the results will be comparable as the test environment was the same.

Table 5.5: Performance of KNN based indoor positioning system

	Mean Error $\bar{e}$ (m)	σ_{error} (m)	$\bar{e} + 2\sigma_{error}$
$K = 1$	2.8	2.7	8.2
$K = 2$	2.8	2.7	8.2
$K = 3$	2.7	2.7	8.1
$K = 4$	2.3	2.8	7.9

Parameter	Value
For Noise Filtering	
Threshold T	> 0.352813
Threshold Q	$Q > 0.595963$ and $Q < 2.0$
For Orientation Bias	
σ_{angle}	8.307481 degrees
For NN based positioning	
Wifi Orientation Tolerance	> 43.48589 degrees
Wifi Positioning Error	$> 2.8m$ with S.D. $> 2.7m$

Table 5.6: Algorithm Parameters Determined by Ground Work

5.9 Summary

To conclude, we can state that by doing all the experiments and other ground-work mentioned in this chapter, we have obtained a fair idea about the inputs of the system and various system parameters required for the algorithms. The groundwork thus helps in guiding the implementation of the system. Since groundwork was being done in conjunction with algorithm design, some feedback from the groundwork was also incorporated into improving the design of the proposed method of Chapter 3.

By doing the groundwork, we have been able to derive the ranges of parameters of the proposed algorithm. These ranges are listed in Table 5.6. Besides finding the ranges of parameters, the ground work has been useful in exposing the behaviour of the input signals. This is very beneficial in diagnosing implementation bugs and determining source of errors when the application does not behave as expected.

6

Experiments and Results

All the experiments and results of this Chapter are performed in the same test environment as described in Section 5.4 using the Android Application implementation described in Chapter 4. Based on the groundwork performed, the parameters of the algorithms were chosen. This chapter discusses the performance of the proposed approach and the two control algorithms based on the same set of selected parameters as the proposed algorithm.

6.1 The Filtering Algorithm

The threshold (Q) for the filtering algorithm described in Section 3.7 was chosen as the mean of the ranges determined in the groundwork to provide adequate noise margin. Therefore, the threshold for the noise filter was set at $1.3m/s^2$.

The clamping technique described in Section 3.7 was employed to filter the raw acceleration.

The effect of the clamping on the raw accelerometer data is shown in Figure 6.1. Figure 6.2 shows the sensor noise that is rejected around zero enabling clean detection of peaks corresponding to the steps.

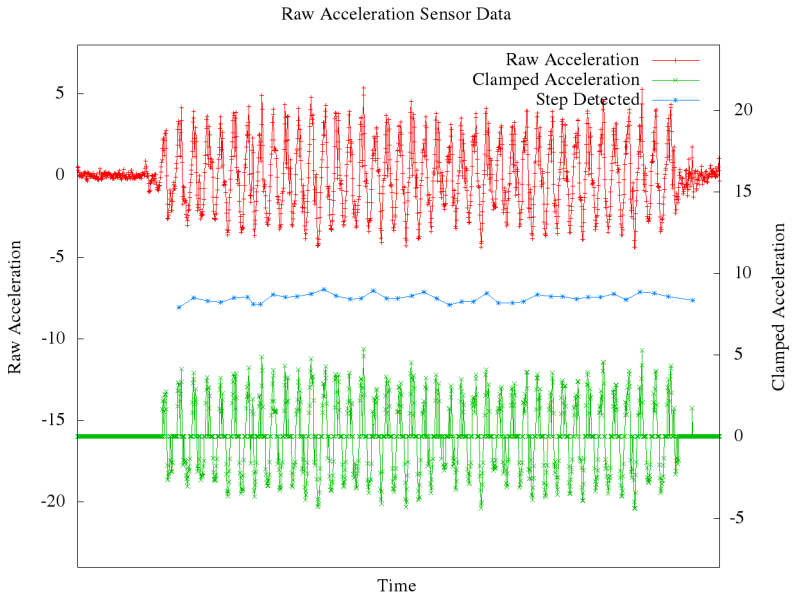

Figure 6.1: Output of the Filtering algorithm

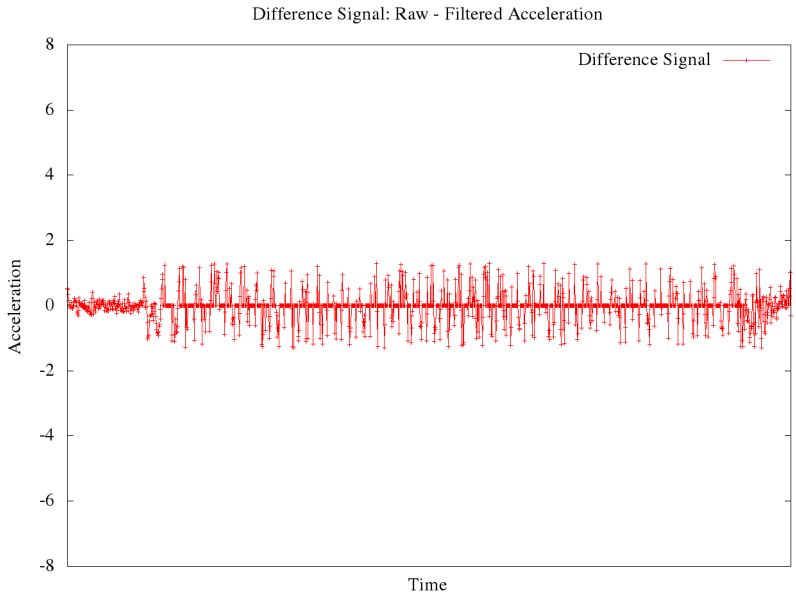

Figure 6.2: Rejected signal from the Accelerometer sensor

6.2 Step detection procedure

The step detection procedures of Section 3.8.1 were analyzed using sensor data of the accelerometer on the Android mobile phone. The simple zero crossing method on unfiltered data was used as a control algorithm to estimate the improvement in step detection.

The comparative graph of the performance of the step detection procedures is shown in Figure 6.3. Table 6.1 shows how the methods performed in relation with the actual number of steps detected.

Algorithm	**Steps Detected**
Simple Zero Crossings Count	109
Zero Crossings on Filtered Data	41
Peaks and Valley Method	41
Actual Number of Steps	**40**

Table 6.1: Step Detection Performance

It can be seen that a simple Zero Crossing method performs similarly to the Peaks and Valley method. From the time based graph plotted in Figure 6.3, it can be seen that the Peaks and Valley method slightly lags the Zero Crossing method in time. This is to be expected as this method detects a step only when the next peak is detected and thus suffers a lag of about half a step.

6.3 First Fix

This section analyzes the two different methods of 3.6 used for obtaining the first fix for the algorithm.

6.3.1 Experimental setup

All the experments were performed with the same Android device (the Samsung Nexus S) and with the same application software. Users were familiar with touch screen phones and were well aware of the layout of the building. None of the users had had any prior exposure to the map of the building and did not know the direction of True North. Details of the experiments performed are specified in Sections 6.3.2 and 6.3.3

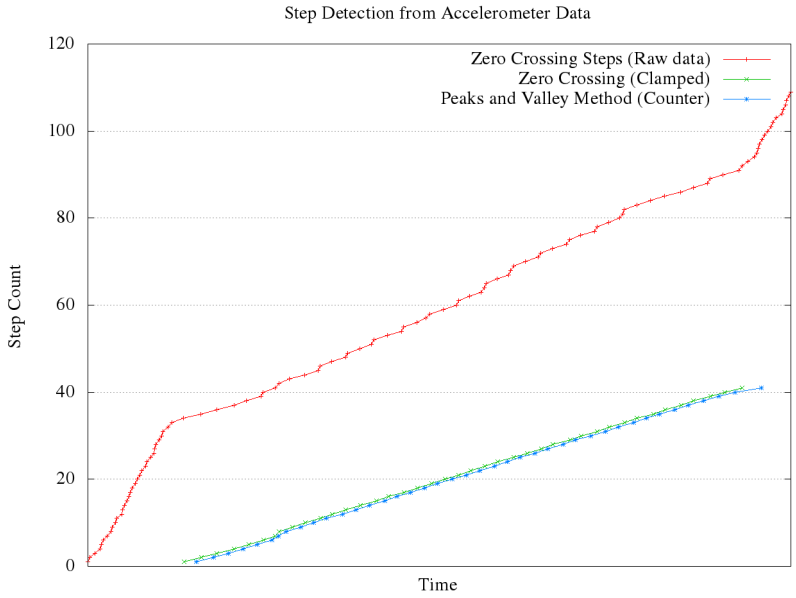

Figure 6.3: Performance of Step Detection Algorithms

6.3.2 Direct Selection of Location on a Map

Five test subjects were asked to mark out their current position on a map by simply touching the corresponding location on the map shown on the touchscreen of the device. The map was shown at a fixed resolution to all test subjects in order to make their responses comparable. Their positions on the touchscreen were snapped to the nearest 1 m grid pixel for ease of use.

The test were also allowed to retry locating themselves on the map in case they were not satisfied with their initial results. All their location attempts were logged for later analysis.The following facts were observed during the process of selection:

1. Some users had an initial difficulty in orienting themselves with regard to the map supplied.

2. As the map contained blocks of rooms that were similar looking, initial errors were made in orientation. However, they attempted to correct their errors when the mistake was realized.

3. Some users felt extremely frustrated by the direct selection procedure primarily because they could not mark out their locations precisely due to the large surface area of contact of their fingers with the touch screen. The variability of selection of the contact point while using the touchscreen contributed to an increase in the number of attempts

User	Retries	Comments
1	6	This user had difficulty orienting himself with respect to the map.
2	0	–
3	2	–
4	2	–
5	15	This user had difficulty with the touchscreen contact areas. He was not a user of a touch screen phone. He also had an initial difficulty with orientating himself on the map.

Table 6.2: Number of Retries for a Satisfactory First Fix

required to correctly mark their positions on the map.

6.3.3 QRCode Based Selection of Location

To evaluate the efficiency of determining first fix via QRCodes in comparison to the direct location selection method, an experiment was set up. The experiment was performed in conjunction with the experment of Section 6.3.2, users were asked to locate themselves by snapping a QRCode pasted on the wall instead of marking their starting location on a map.

The users were then positioned at various distances from the QRCode and asked whether they would make the effort to go to the QRCode to snap their starting location while using the application or whether they would prefer the manual method.

The following observations were made regarding the QRCode method for first fix:

1. QRCode selection via the camera was a fast process, usually not requiring more than 10 seconds on behalf of the user.

2. Users were consistently located at distances less than 1 meter from the QRCode. In most cases, the users were at a distance of around 30 cm from the QRCode pasted on the wall.

3. There was an instance where poor lighting delayed the capture of the QRCode and the QRCode could be successfully captured only after a fluorescent tubelight was switched on to provide adequate lighting.

User Number	First fix by manual method (Ease of use: scale of 5)	First fix by QRCodes (Ease of use: scale of 5)	Building familiarity required for manual method (scale of 5)
1	1	5	5
2	2	3	4
3	2	4	5
4	3	5	4
5	1	5	4
Mean	**1.8**	**4.4**	**4.4**

Table 6.3: Comparison of QRCode based first fix with the Manual Marking Method

6.3.4 Comparison between the two methods

After processing the results of the experiments, it was determined that users overwhelmingly preferred the QRCode method for first fix acquisition. They were willing to walk up to $5m$ to the nearest QRCoded location in order to take advantage of the facility. The results of the comparative survey are listed in Table 6.3.

6.3.5 User Feedback

Users in their comments indicated that potential improvements to the application were possible. For example, if the application could determine roughly their current location and then provide just that subset of the map in which the user was located, the speed of providing the first fix location by a manual method could be improved.

6.4 Determining the Training Constant

In a separate experiment, three test subjects were asked to walk between 2 QRCoded locations. The first fix was taken using the QRCode method and the training constant was thus estimated based on the number of steps detected and the known distance between the 2 QRCoded locations.

The locations used for this test were along a corridoor and were about $30m$ apart. The tests were repeated thrice for each subject. The step constants for the users were different but consistent.

6.5 Step variation analysis

To see the performance of the algorithm to step variations, an experiment was performed wherein a test subject walked at three distinct speeds along a straight corridoor. The subject paused for a short while between the three distinct segments to maintain separation of speeds.

The three segments of the walk had the following characteristics:

1. The test subject walked with very light steps, almost shuffling along.
2. The test subject walked with regular steps along the corridoor.
3. The test subject walked with large steps along the corridoor in a motion that approximated running.

The accelerometer graphs and the consequent step detections for this experiment are shown in Figure 6.4. The tabulated results of the step detection process are shown in Table 6.4. Note that the poor performance for shuffling steps is due to acceleration peaks not being sufficiently distinguishable from sensor noise and for the fast heavy steps is due to generation of spurious peaks while coming to an abrupt halt. A smooth motion doesn't generate such issues and thus the step detection procedure is very accurate for that range.

Step Segment	Actual Number of Steps	Steps Detected
Light Steps (Shuffle)	10	5
Regular steps	10	10
Fast (Heavy) steps	15	18

Table 6.4: Step detections under step variations

6.6 Test Paths

Five different test paths were used to test the performance of the proposed algorithm. The features of these paths are described below. The paths themselves are marked out in Figure 6.5.

1. Straight walk down a corridoor (A to B)
2. Straight walk down a corridoor, a U turn followed by a straight walk to the starting point. (A to B and then B to A)
3. Zig-Zag walk down a corridoor (A to B).

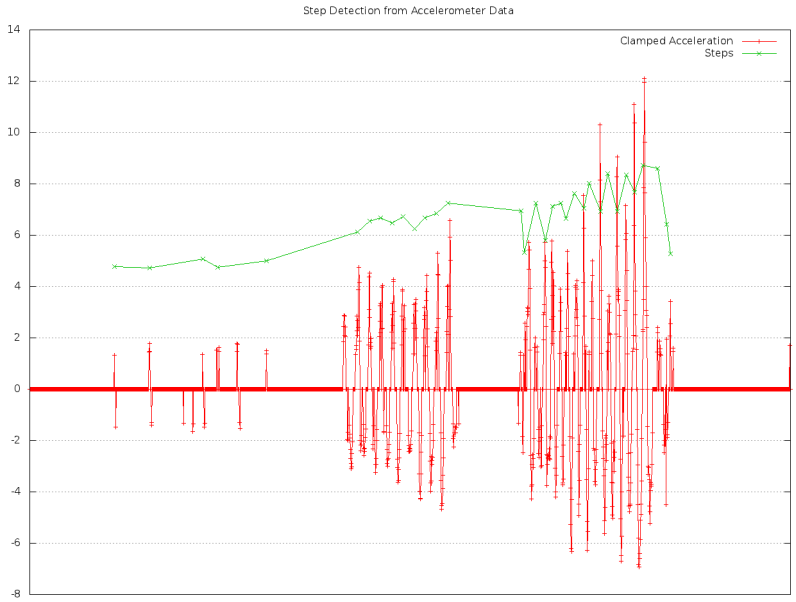

Figure 6.4: Difference in step sizes and the corresponding variations in step detection and step estimates

4. A U shaped walk along all three corridoors. (A-B-C-D)

5. A U shaped walk along all three corridoors followed by a U turn and a walk back to the starting location. (A-B-C-D-C-B-A)

6.7 Simple Dead Reckoning

As discussed in the Literature Review (Chapter 2) and in the Proposed Work (Chapter 3), it is impossible to directly obtain displacement information by double integration of the accelerometer data. Hence, the step detection method was employed for tracking the motion of the device as held by a walking human. The steps are integrated with orientation information received from the magnetometer to create a simple dead reckoning system in accordance with the dynamical equation (3.4). The initial position for dead reckoning was obtained via QRCodes in accordance with the description in Section 3.6.1. This algorithm is a control algorithm to be used for comparison with the proposed method.

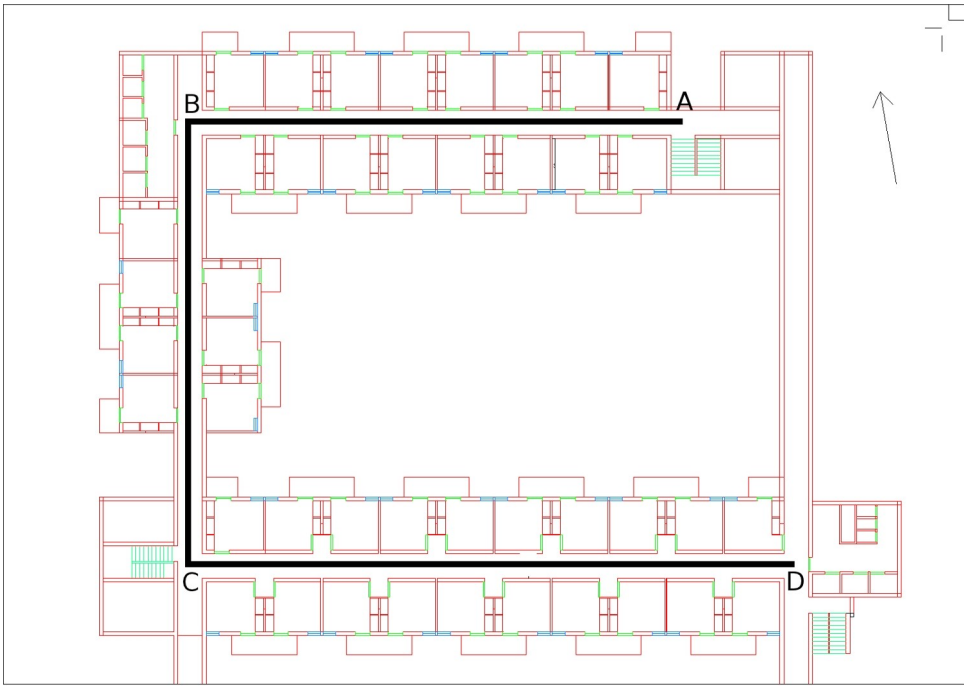

Figure 6.5: Test paths for algorithm

6.7.1 Simple Dead Reckoning Results

One of the well known weaknesses of the dead reckoning approach is that it diverges quickly from the ground truth unless corrective measures are taken. As a basic control for the results, a simple dead reckoning based tracking algorithm was developed.

Figure 6.6 shows the quick divergence of the algorithm from the expected path A-B. In fact, at the end of a simple straight walk of around $30m$, the estimate for the location has diverged a lot from the actual location. It should also be noted that the path taken passes through rooms and walls and thus, doesn't look like a natural path for the user.

Figure 6.7 shows the extreme divergence that takes place over long distances. Table 6.5 chronicles the error growth that takes place for this path at the turning points of the path. It is amply clear that Simple Dead Reckoning is not suitable for tracking if implemented as-is.

6.8 Reckoning with NN Wifi correction

As per our groundwork described in Section 5.8 we know that the mean error for positioning using KNN is around $2.8m$ with a standard deviation that is nearly the same as the mean. Given the large errors incurred by the dead reckoning solution, we wish to analyze the performance

Location	Distance Travelled	Error
A	0	0
B	30 m	3.761 m
C	56 m	4.982 m
D	92 m	5.416 m

Table 6.5: Error growth over distance for Simple Dead Reckoning

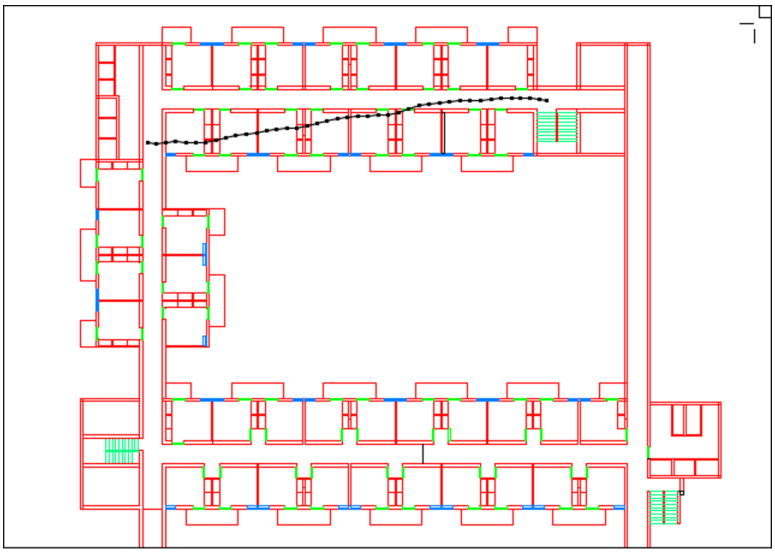

Figure 6.6: Performance of Dead Reckoning for Test Path 1

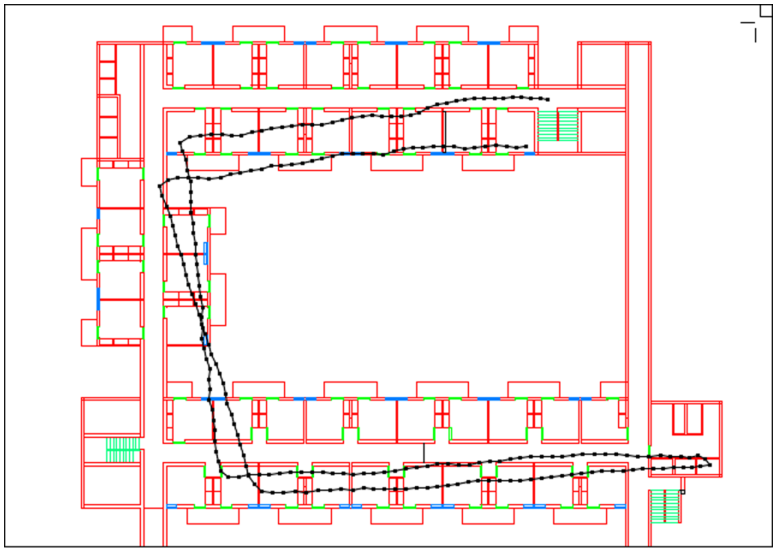

Figure 6.7: Performance of Dead Reckoning for Test Path 5

of Dead Reckoning after Wifi data is integrated into the system.

6.8.1 Wifi Survey

We carried out a Wifi Survey of the test area and collected 1562 samples by carefully laying down a $1m \times 1m$ grid and collected samples. 8 different samples were collected per grid point at orientations that were detected to be 45 degrees apart by the onboard magnetometer. By relying on the magnetometer itself to specify orientation, the effect of local magnetic fields was taken into account during the survey. However, sensor noise causes the angle values actually recorded with the Wifi Signal Strength data to vary from the expected orientation of the user.

6.8.2 Tracking Performance

While implementing and testing out the implementation of the KNN based tracking solution, the resource and scaling limitations of the system came to fore.

Because of it being a "lazy" algorithm that consults the entire training dataset on every query for a position, the KNN algorithm is computationally very intensive. Queries for obtaining a position based on Wifi information was a slow process that even disrupted the UI of the application due to its heavy workload. Optimizations involving cacheing of data and short-circuiting the distance calculation step based on the orientation angle of the training sample viz

	Error Value (m)
Min	1.888e-07
1^{st} Quartile	1.448
Median	2.948
Mean	3.748
3^{rd} Quartile	5.064
Max.	14.87
Std. Dev.	2.953

Table 6.6: Error in tracking for the NN Algorithm

a viz the current device orientation were implemented. With these optimizations, we were able to successfully run a hybrid Dead Reckoning system that utilized Wifi Access Point Received Signal Strengths and Step motion information to track the device.

The performance of the algorithm was again judged based on the test paths mentioned in Section 6.6. Table 6.6 describes the statistical nature of the error incurred for Path 1 with respect to the actual position of the device. The values obtained agree with those reported in papers like [12] as well as our groundwork. Figure 6.8 shows the track for Test path 1. A quick glance will indicate that the tracked endpoints are reasonably accurate ($error < 2m$), however, the subjective nature of the tracked path is very poor.

6.9 Reckoning with Integrated Map Information

The results of the Wifi corrected reckoning as well as the data collected as part of the implementation of the KNN positioning system indicates that the location information utilizing Wifi data is very unstable. On the other hand there is significant drift if we depend solely on the accelerometer and the magnetometer to provide intertial tracking. In order to get better accuracy from the system, we require constant feedback from the environment. A way to achieve that is to use the fact that the device holder is always moving through "passable" spaces on the map. Any motion contrary to the allowed locations on the map would indicate tracking drift and a feedback loop would be set up to correct for such errors.

In this context, the particle filter's performance is analyzed with the same test tracks as the other two algorithms.

As can be seen amply in Figure 6.10 and Figure 6.11 the tracking performance of this algorithm is excellent. The drift errors of the dead reckoning solution are compensated for by the map information which ensures that only states consistent with the motion of the device are retained. Long term behaviour of the algorithm is also remarkably stable. Even after moving

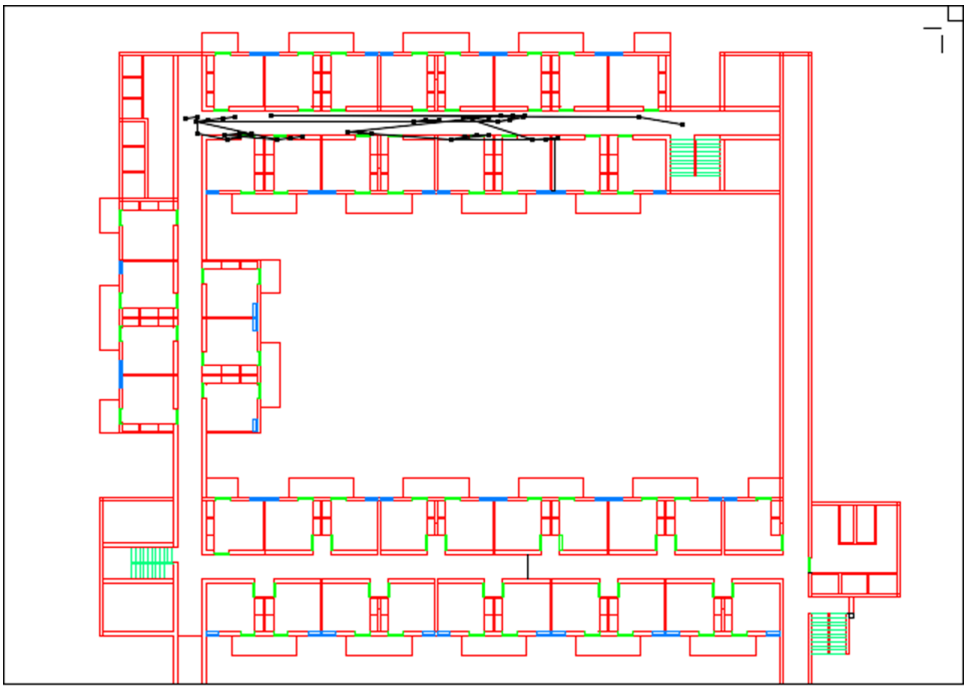

Figure 6.8: Performance of the KNN algorithm for Test Path 1

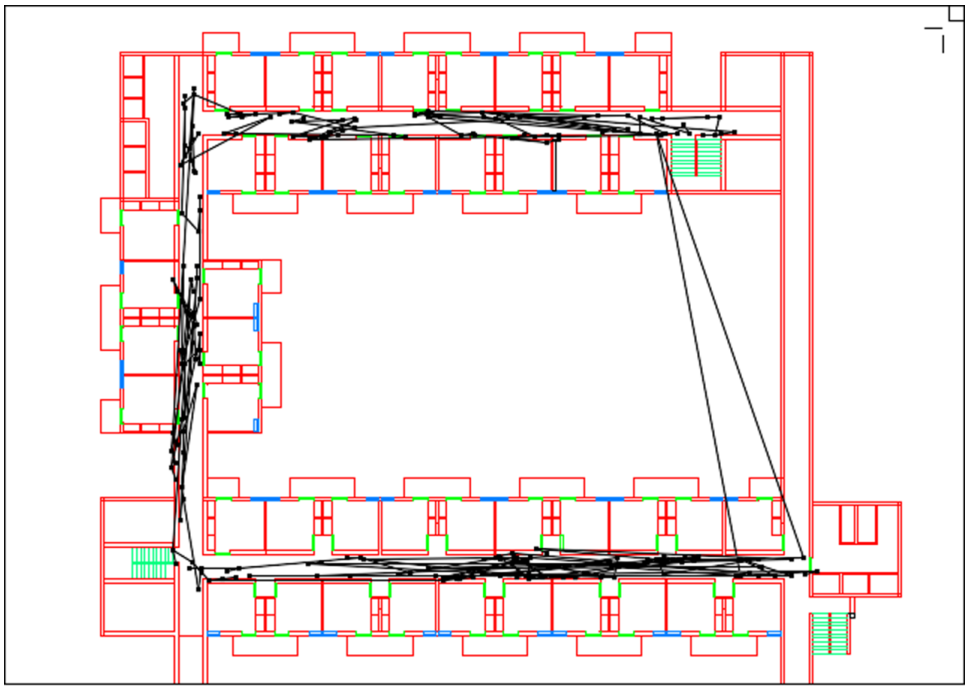

Figure 6.9: Performance of the KNN algorithm for Test Path 5

	Error Value (m)
Min.	1.888e-07
1^{st} Quartile	0.2750
Median	0.5218
Mean	0.5064
3^{rd} Quartile	0.7209
Max.	0.9412
Std. Dev.	0.2391

Table 6.7: Tracking Error of Proposed Algorithm for Path 1

close to 100 m, the algorithm was able to maintain a tracking error of less than 1 m. This is illustrated in Figure 6.11.

To analyze the algorithm more carefully, the ground truth was mathematically modelled as a series of even sized steps along the path A-B for a test done on Path 1. The resulting tracking errors were determined and their statistical analysis was done. The error distribution over the distance of 30 m are summarized in Table 6.7.

As is clear from this error distribution, the error incurred is low. Also, more importantly, the standard deviation of the error distribution is low. Hence it is unlikely that we will get large deviations from the tracked path.

6.10 Recovery from Particle Insufficiency

For a typical test path, the variation in the number of live particles is shown in Figure 6.12. As can be seen clearly, the particle filter does suffer dips in the number of live particles but the number of particles remaining is around 5. Thus maximum value of 50 particles seems to be a reasonable value for our tracking algorithm. An empirical determination of the computational limits of the implemented algorithm seems to indicate that the maximum number of particles that the device is able to update per computation step lies around 200.

The dips seen in Figure 6.12 correspond to events where the particle filter is adapting its location estimate based on feedback received from $MapSelect$. The algorithm may get lost if the dips reach a value 0. In this case the recovery algorithm is invoked. Figure 6.13 shows how the particle filter recovered successfully from a “lost” scenario to continue tracking of the device, albeit with some error.

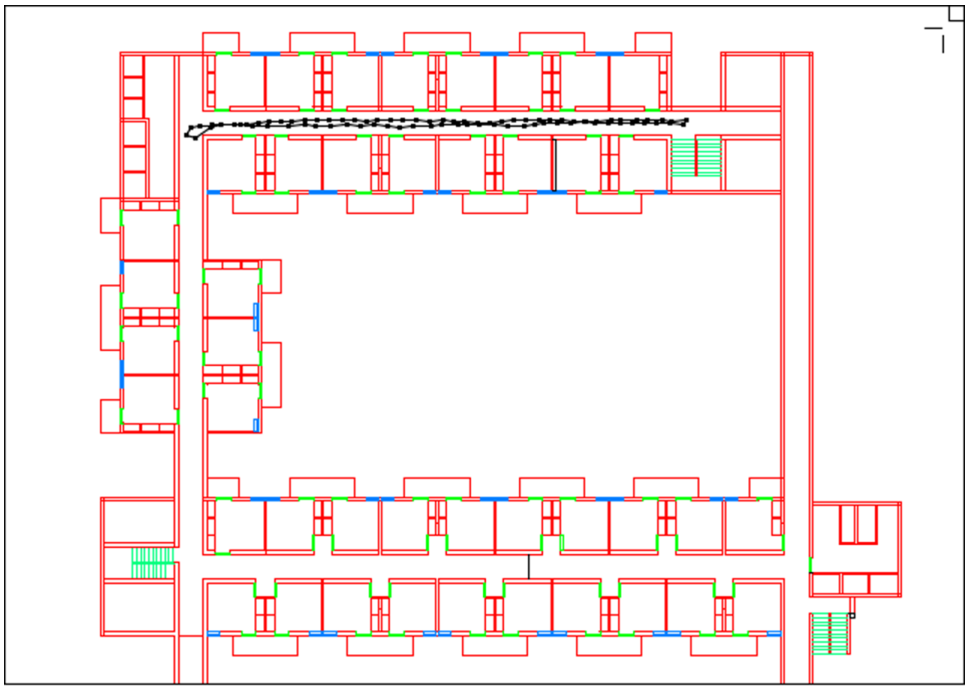

Figure 6.10: Performance of Reckoning with Map info for Test Path 2.

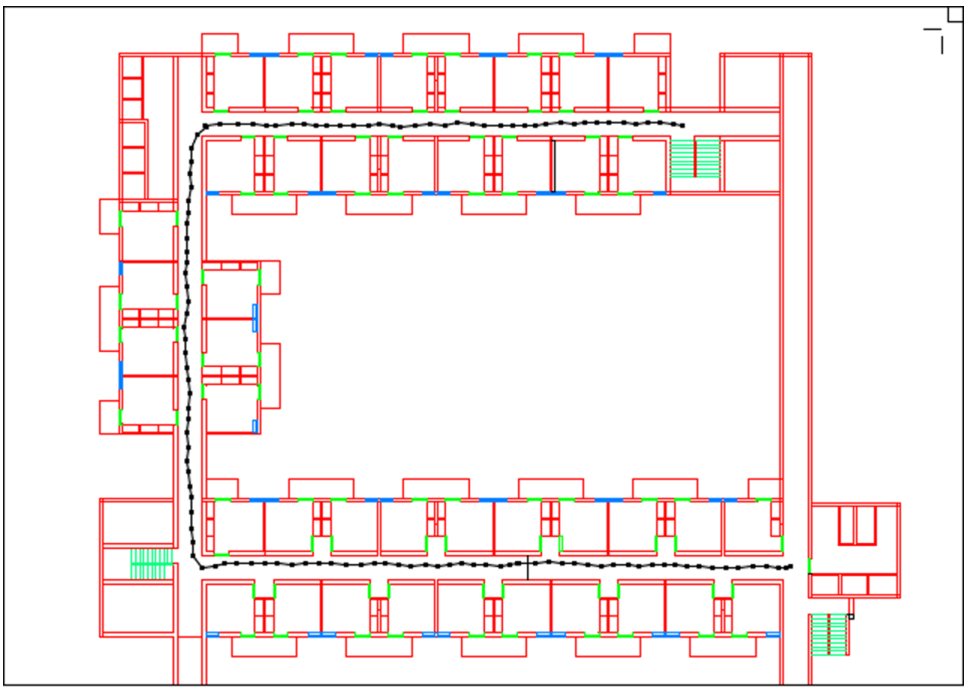

Figure 6.11: Performance of the Reckoning algorithm for Test Path 4.

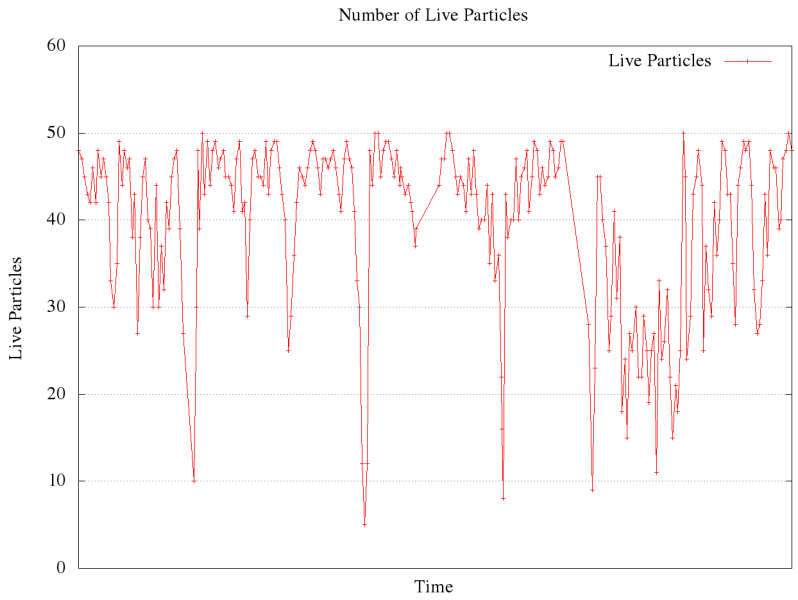

Figure 6.12: Number of Live Particles for a Path

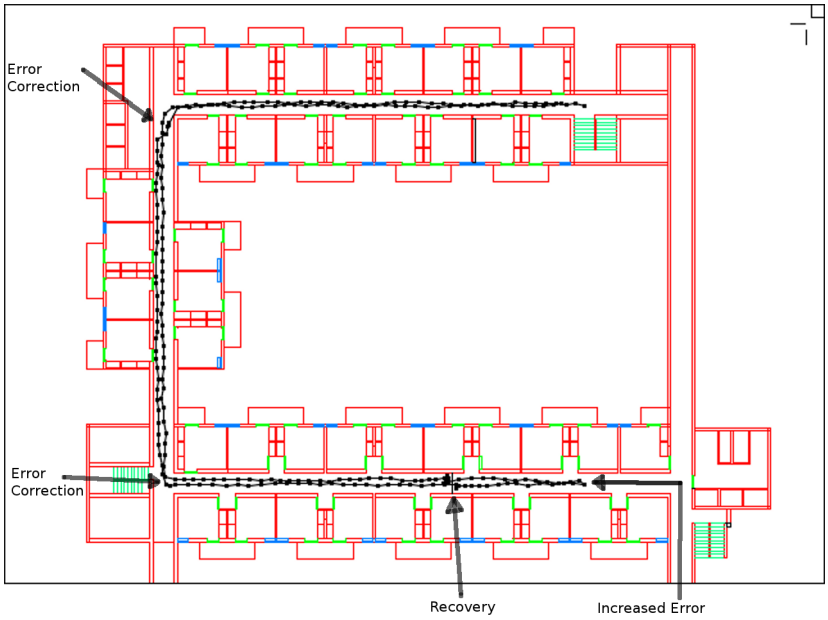

Figure 6.13: Result of a successful recovery

6.11 Comparison with published results

With additional information at hand as well as suitable modifications to the dynamical equations of the system and care in avoiding particle degeneracy, a significantly better accuracy has been achieved by this system as compared to previous published works.

The comparison of the 3 implemented methods is shown in Table 6.8.

Algorithm	Mean Error (m)	Std. Dev. (m)
Wang[12] KNN only	6.44	6.84
Wang[12] PF with RSSI	5.57	3.9
Wang[12] PF with RSSI + Steps	4.54	3.52
Wang[12] PF with RSSI + Steps + Map	4.30	2.80
Evennou[11] PF with RSSI	1.86	N/A
Evennou[11] PF with RSSI + INS	1.53	N/A
Control Algorithm: NN Wifi corrected Reckoning	**3.748**	**2.953**
Proposed Work: PF with INS + Map	**0.5064**	**0.2391**

Table 6.8: Comparison with Published Results

6.12 Complete listing of the results

In order to maintain the flow of the thesis and due to the voluminous nature of reporting the results for each of the test paths for the 3 algorithms, the results of those tests have been reported in the appendices.

7

Summary and Future Work

7.1 Summary of the Work

As per the problem statement described in Section 1.3, an indoor tracking solution based on a particle filter approach for pedestrians has been developed for the Android smartphone. The performance of the algorithm has been evaluated and it has shown an improvement against prior work in the area.

Based on the insights gained during this work, a few points need to be stated in conclusion:

1. QRCodes are excellent tools for providing first fixes to tracking algorithms. They are reasonably scalable and extremely easy to use.

2. Wifi is a poor choice of signal source to correct for drift errors for step estimate based tracking solutions. This is because Wifi signal sources are too noisy that they are only able to correct gross drift errors. In fact, using Wifi data for drift correction probably degrades accuracy rather than improves it.

3. Having a Wifi survey of an area is a very time consuming matter. However, it provides an additional way to resolve situtations where the particle filter gets lost.

4. Treating all particles at par (as is done in this algorithm) reduces computational complexity of the particle filter by eliminating the reweighting done at each step.

5. It is very important to analyze the parameters of the sensors to choose appropriate values for the algorithm. Doing so ensures particle conservation and thus reduces the computational load on the system.

The system has shown excellent performance in the tests and as an outcome of this thesis, it is recommended that only map information be used for drift correction instead of a combination of Map information and Wifi data.

Of course, as no system is perfect, this system too has its shortcomings and limitations:

1. Due to the lower number of particles allowed, the algorithm has a higher tendency to get lost.

2. Once lost, it is hard to recover and recovery is usually associated with some tracking error.

3. The system requires a Wifi Survey of the test area for error recovery which is time consuming and non-scalable. This may be avoided however if we are willing to sacrifice a better chance at recovery in case the algorithm gets lost.

4. Using the lazy Nearest Neighbour algorithm is expensive in time and memory consumption and is not very scalable.

5. Small steps cannot be detected due to limitations of the step detection algorithm.

6. Maps with large open spaces provide very little feedback to the tracking algorithm and thus the tracking quality degrades.

All these points provide scope for future work on this system.

7.2 Limitations

The one thing that this algorithm is unable to take into account is when a user decides to move backwards while retaining the same direction of the smartphone. Since the algorithm makes an assumption of forward motion only, moving backwards significantly affects the accuracy of the tracking algorithm and the algorithm quickly gets lost. This is a limitation of the system.

7.3 Future enhancements

There is a lot of scope for improving a number of aspects of the algorithms proposed. For example, we need to look at improving $MapSelect$, integrating information from barometric sensors as and when they are made available on handheld devices. We also need to figure out how to modify the information to be put into first fix QRCodes to link up maps of the same building. SVM can also be used to increase the accuracy of the recovered position on the lines of the work done by [25] for Redpin[16].

On the implementation front, performance of the solution, though adequate, can be significantly improved to enable a greater number of particles to be evolved. For example, improvements can be achieved by leveraging thread parallelism and a producer consumer model to allow for higher system lag in the case of a recovery event. The UI can be made event driven from the current polling model to increase responsiveness. Also, the Android NDK can be used to write the performance intensive parts of the solution in C.

7.4 Crowdsourcing a Signal Strength Map

An application of this thesis that arises naturally from the results is that the base algorithm can be used to set up a crowdsourcing system that will eventually generate a Wifi signal strength map of a large test area based solely on user inputs generated while they are using the tracking application. This would be beneficial for bootstrapping projects like [16] for providing lower complexity solutions to the indoor positioning problem. This aspect requires greater thought and research.

A

Test results for all algorithms

A.1 Simple Dead Reckoning

Figures A.1, A.2, A.3, A.4, A.5 show the results of simple dead reckoning on the Test Paths 1-5.

A.2 Wifi Corrected Reckoning

Figures A.6, A.7, A.8, A.9, A.10 show the results of Wifi corrected reckoning on the Test Paths 1-5.

A.3 Reckoning with Map Information

Figures A.11, A.12, A.13, A.14, A.15 show the results of particle filtered reckoning with map information on the Test Paths 1-5.

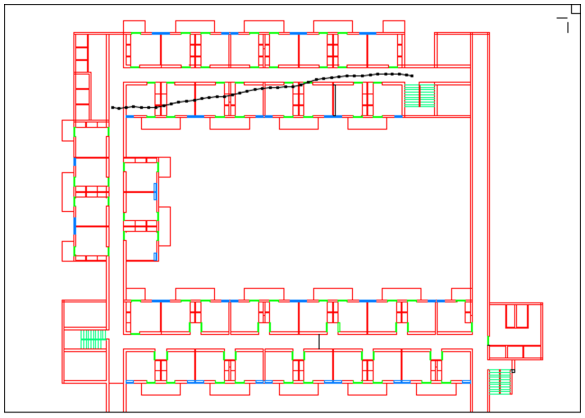

Figure A.1: Simple Dead Reckoning Path 1

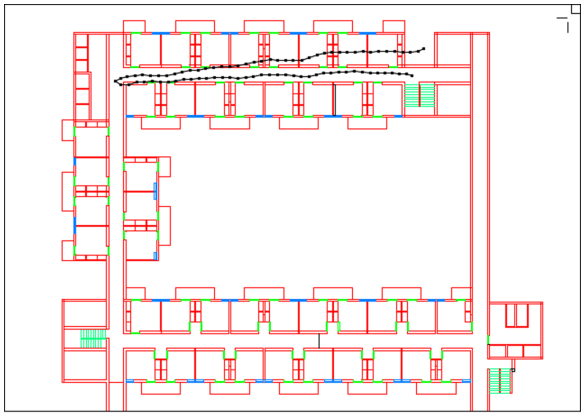

Figure A.2: Simple Dead Reckoning Path 2

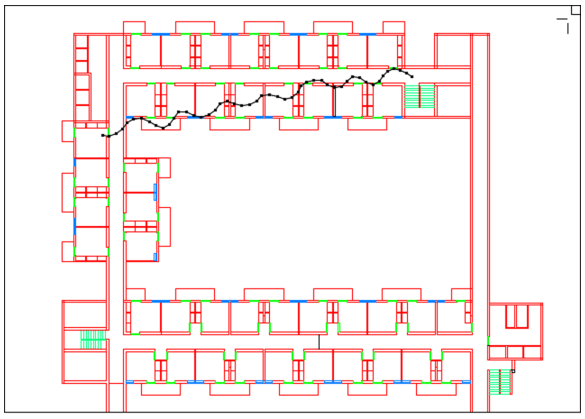

Figure A.3: Simple Dead Reckoning Path 3

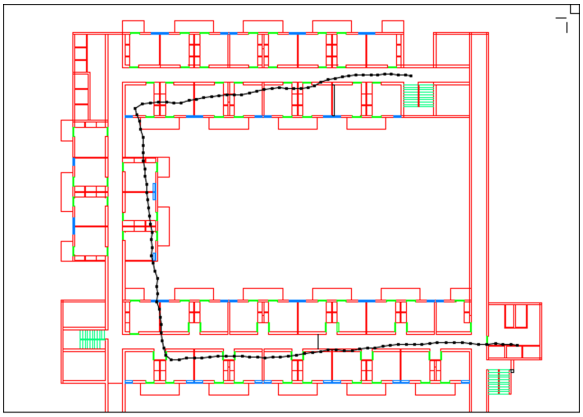

Figure A.4: Simple Dead Reckoning Path 4

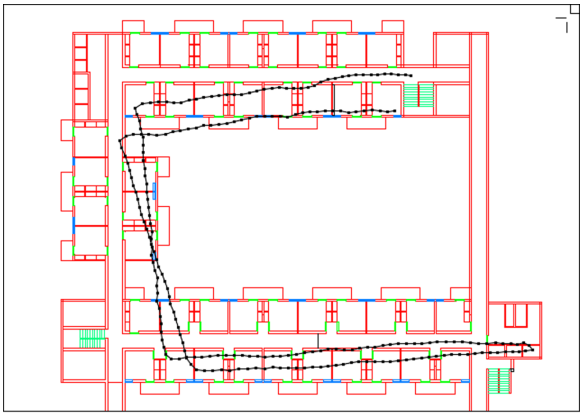

Figure A.5: Simple Dead Reckoning Path 5

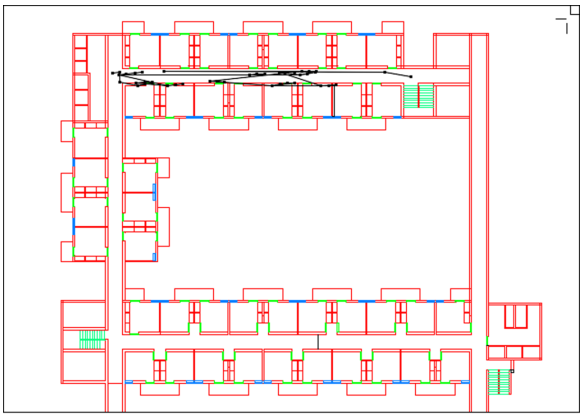

Figure A.6: Wifi Corrected Reckoning Path 1

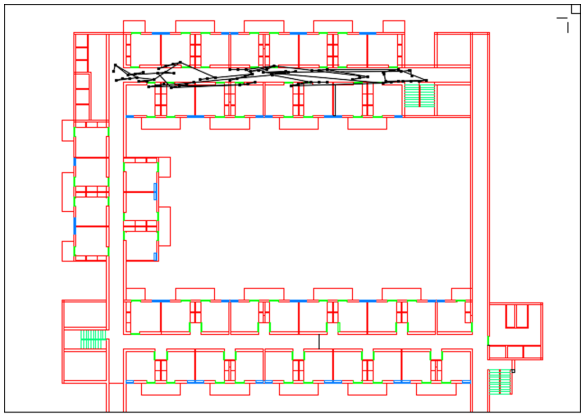

Figure A.7: Wifi Corrected Reckoning Path 2

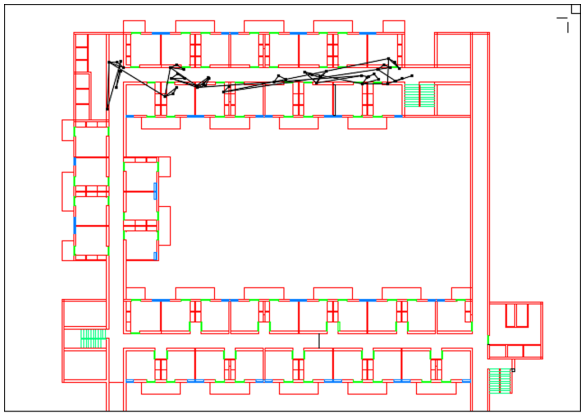

Figure A.8: Wifi Corrected Reckoning Path 3

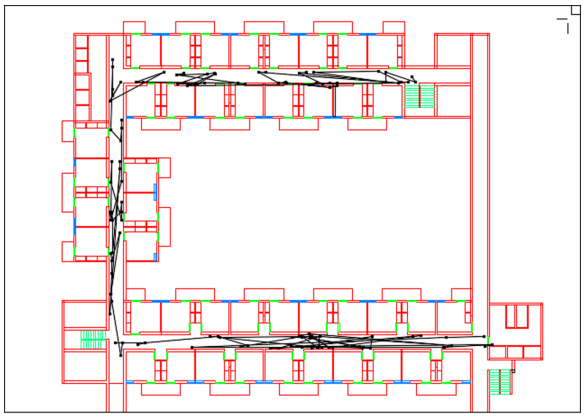

Figure A.9: Wifi Corrected Reckoning Path 4

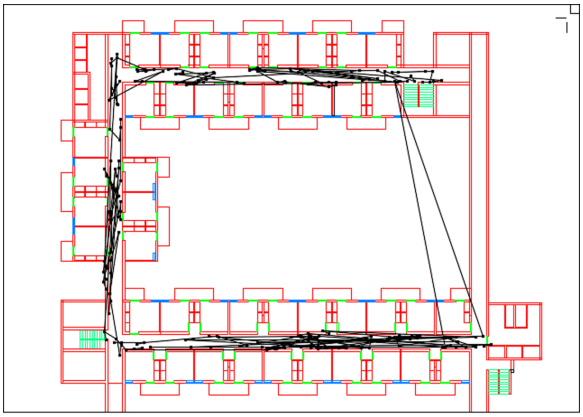

Figure A.10: Wifi Corrected Reckoning Path 5

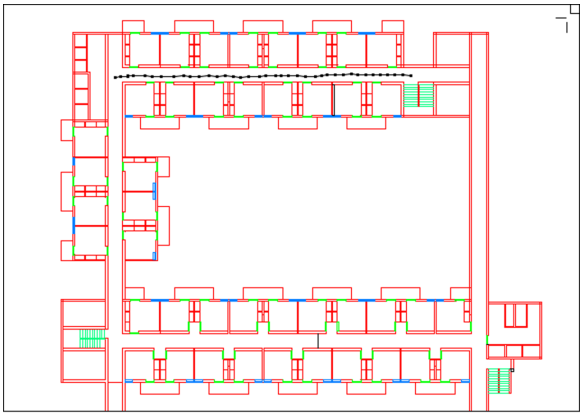

Figure A.11: Reckoning with Map Information Path 1

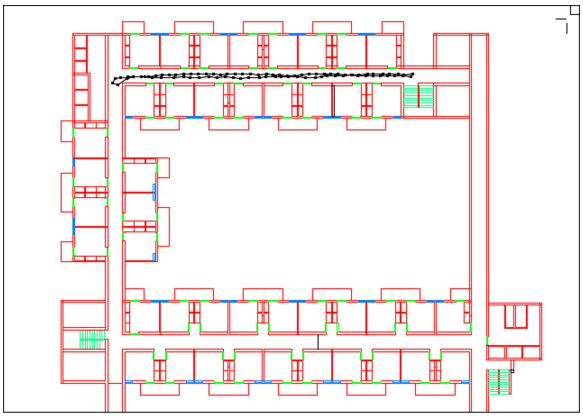

Figure A.12: Reckoning with Map Information Path 2

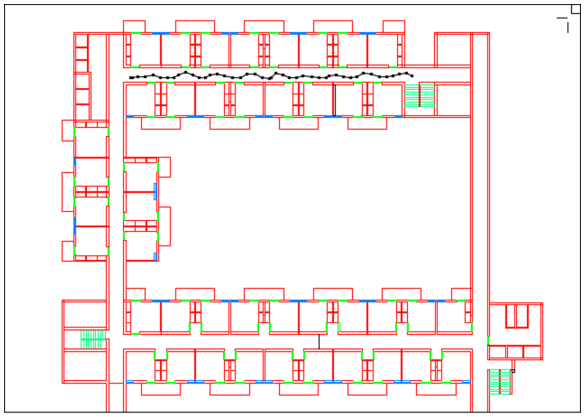

Figure A.13: Reckoning with Map Information Path 3

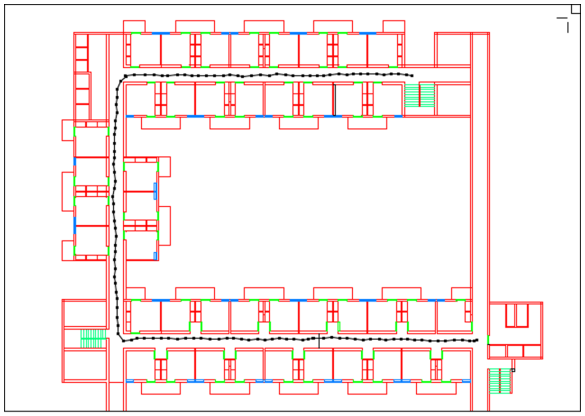

Figure A.14: Reckoning with Map Information Path 4

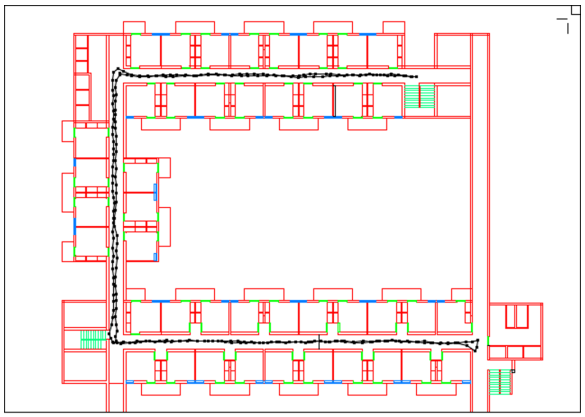

Figure A.15: Reckoning with Map Information Path 5

Bibliography

[1] K. Kaemarungsi and P. Krishnamurthy, "Properties of indoor received signal strength for wlan location fingerprinting," in *International Conference on Mobile and Ubiquitous Systems: Networking and Services*, 2004, pp. 14–23.

[2] D. Kapoor, "Location Acquisition, Accuracy Constraints and Map Matching Algorithms for Navigation Oriented Location Based Services," 2010, Indian Institute of Technology, Roorkee (Unpublished).

[3] "The Encyclopaedia of Virtual Environments," 1993. [Online]. Available: http://www.hitl.washington.edu/scivw/EVE/IV.Definitions.html

[4] P. Bahl and V. N. Padmanabhan, "RADAR: An In-Building RF-based User Location and Tracking System," in *IEEE INFOCOM*, 2000.

[5] R. Want, A. Hopper, V. Falcão, and J. Gibbons, "The active badge location system," *ACM Trans. Inf. Syst.*, vol. 10, no. 1, pp. 91–102, Jan. 1992. [Online]. Available: http://dx.doi.org/10.1145/128756.128759

[6] O. Gremigni and D. Porcino, "UWB Ranging Performance Tests in Different Radio Environments," in *London Communications Symposium*, 2006. [Online]. Available: http://www.ee.ucl.ac.uk/lcs/previous/LCS2006/53.pdf

[7] A. Kotanen, M. Hännikäinen, H. Leppäkoski, and T. Hämäläinen, "Experiments on Local Positioning with Bluetooth," in *International Symposium on Information Technology*, 2003, pp. 297–303.

[8] Jeffrey, "SpotON: An Indoor 3D Location Sensing Technology Based on RF Signal Strength," *Intel Research*, 2000. [Online]. Available: http://seattle.intel-research.net/people/jhightower/pubs/hightower2000indoor/hightower2000indoor.pdf

[9] A. M. Ladd, K. E. Bekris, A. Rudys, L. E. Kavraki, and D. S. Wallach, "Robotics-Based Location Sensing Using Wireless Ethernet," *Wireless Networks*, pp. 189–204, 2005, pub: Springer.

[10] T. King, S. Kopf, T. Haenselmann, C. Lubberger, and W. Effelsberg, "COMPASS: A Probabilistic Indoor Positioning System Based on 802.11 and Digital Compasses," in *Proceedings of the First ACM International Workshop on Wireless Network Testbeds, Experimental evaluation and CHaracterization (WiNTECH)*, Los

Angeles, CA, USA, Sep. 2006. [Online]. Available: http://www.informatik.uni-mannheim.de/pi4/publications/King2006g.pdf

[11] F. Evennou and F. Marx, "Advanced Integration of WiFi and Inertial Navigation Systems for Indoor Mobile Positioning," *Eurasip Journal on Advances in Signal Processing*, vol. 2006, pp. 1–12, 2006.

[12] H. Wang, H. Lenz, A. Szabo, J. Bamberger, and U. D. Hanebeck, "WLAN-Based Pedestrian Tracking Using Particle Filters and Low-Cost MEMS Sensors," in *Positioning, Navigation and Communication, 2007. WPNC '07. 4th Workshop on*, Mar. 2007, pp. 1–7. [Online]. Available: http://dx.doi.org/10.1109/WPNC.2007.353604

[13] M. S. Arulampalam, S. Maskell, and N. Gordon, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, pp. 174–188, 2002.

[14] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman Filter-Particle Filters for Tracking Applications*. Artech House Publishers, 2004, (Book).

[15] Widyawan, M. Klepal, and S. Beauregard, "A novel backtracking particle filter for pattern matching indoor localization," in *Proceedings of the first ACM international workshop on Mobile entity localization and tracking in GPS-less environments*, ser. MELT '08. New York, NY, USA: ACM, 2008, pp. 79–84. [Online]. Available: http://doi.acm.org/10.1145/1410012.1410031

[16] P. Bolliger, "Redpin - adaptive, zero-configuration indoor localization through user collaboration," in *Proceedings of the first ACM international workshop on Mobile entity localization and tracking in GPS-less environments*, ser. MELT '08. New York, NY, USA: ACM, 2008, pp. 55–60. [Online]. Available: http://doi.acm.org/10.1145/1410012.1410025

[17] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of Wireless Indoor Positioning Techniques and Systems," *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 6, pp. 1067–1080, Nov. 2007. [Online]. Available: http://dx.doi.org/10.1109/TSMCC.2007.905750

[18] Ladetto, Quentin, "On Foot Navigation: continuous step calibration using both complementary recursive prediction and adaptive Kalman filtering," in *ION GPS 2000, Salt Lake City, Utah*, 2000. [Online]. Available: http://www.ladetto.ch/LAQ/publications.php

[19] A. W. S. Au, "Rss-based wlan indoor positioning and tracking system using compressive sensing and its implementation on mobile devices," Masters Thesis, Graduate Department of Electrical and Computer Engineering, University of Toronto, 2010.

[20] W.-S. S. Yunye Jin, Hong-Song Toh and W.-C. Wong, "A robust dead-reckoning pedestrian tracking system with low cost sensors," in *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, Seattle, Washington, USA, March 2011, pp. 222–230.

[21] H. S. Christian Lukianto, Christian Hönniger, "Pedestrian smartphone-based indoor navigation using ultra portable sensory equipment," in *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Zurich, Switzerland, 15-17 September 2010.

[22] A. Devices, "Using the ADXL202 in Pedometer and Personal Navigation Applications," application Note AN602, Analog Devices.

[23] Google Inc., "Tech Specs - Nexus S," 2010. [Online]. Available: www.google.com/phone/detail/nexus-s

[24] Google Inc, "Android Reference API - SensorManager," 2011. [Online]. Available: http://developer.android.com/reference/android/hardware/SensorManager.html

[25] L. Rogoleva, "Crowdsourcing Location Information to Improve Indoor Localization," Master's thesis, Swiss Federal Institute of Technology (ETH) Zurich, 2010.