Computer Engineering, Degree Project, Advanced Course,
15 higher education credits

INDOOR POSITIONING USING WLAN

Arvid Norlander and Pierre Andersson

Computer Engineering Programme, 180 higher education credits

Örebro, Sweden, Spring 2012

Examiner: Mathias Broxvall

INNOMHUSPOSITIONERING MED WLAN

Örebro universitet
Institutionen för
naturvetenskap och teknik
701 82 Örebro

Örebro University
School of Science and Technology
SE-701 82 Örebro, Sweden

Abstract

This report evaluates various methods that can be used to position a smartphone running the Android platform, without the use of any special hardware or infrastructure and in conditions where GPS is unavailable or unreliable; such as indoors. Furthermore, it covers the implementation of such a system with the use of a deterministic fingerprinting method that is reasonably device independent, a method which involves measuring a series of reference points, called fingerprints, in an area and using those to locate the user.

The project was carried on behalf of Sigma, a Swedish software consulting company.

Sammanfattning

Denna rapport evaluerar olika metoder för att bestämma positionen av en smartphone som använder sig av Android-plattformen. Metoden skall inte använda sig av någon speciell hårdvara eller infrastruktur samt kunna hantera förhållanden där GPS är otillgängligt eller opålitligt, som till exempel inomhus. Den beskriver också implementation av ett sådant system som använder sig av en deterministisk fingerprinting-metod som någorlunda väl kan hantera enheter av olika modeller, en metod som innebär att man mäter upp en mängd med referenspunkter, kallade fingerprints, och använder dessa för att placera användaren.

Projektet utfördes på uppdrag av Sigma, ett svenskt mjukvarukonsultbolag.

Preface

We want to express our gratitude to Peter Lorenz and Sigma for giving us the opportunity to realize this project. We would also like to thank Thomas Padron-McCarthy for being our supervisor at Örebro University and Holger Schellwat for helping us understand certain parts of compressive sensing.

Contents

1 Introduction

1.1 Background

The project was carried out on the behalf of Sigma, a consulting firm founded in 1986. Sigma operates in the areas of system development, management services, business systems and technical documentation.

The idea of the project was to develop a system for indoor positioning for Android smartphones that works in environments where GPS-based positioning does not. Since GPS uses microwave signals, it is attenuated by walls and similar obstacles and thus, does not perform well in such environments [1]. As requested by Sigma, the system should work with "off-the-shelf" hardware only, mainly to keep costs of potential deployment of the system down. This leaves two technologies as the most promising candidates, Bluetooth and Wireless Local Area Network (WLAN, or Wi-Fi) [2]. Both are radio-based techniques that most Android smartphones have the capabilities to take advantage of [2].

Research of the subject found that several different solutions in the subject had been explored previously. The different solutions can be divided into multiple types, those based on distance measurement, those based on angle of arrival, and fingerprinting.

Distance measurement requires multiple transmitters with known locations. Positioning is done by triangulating the location of the receiver based on the distance to the transmitters, measured by either using propagation time of the radio signals or by using the received signal strengths (RSS) to estimate it. [3]

In angle of arrival systems, like in distance measurement systems, several transmitters with known locations are also needed. The receiver measures the angle it receives signals from, generating a straight line to each transmitter used. The point where those lines all intersect is where the receiver is located. [3]

Finally, fingerprinting is based on the idea of selecting a number of reference points ("fingerprints", hence the name) and measuring the signal strengths of visible transmitters at those points, thus recording the radio signature there. During the actual positioning the signal strengths of the various visible transmitters is compared to those recorded for the fingerprint to find the best match or matches and then use some method of interpolation between those. Unlike the other systems, the locations of the transmitters do not need to be known. [4]

1.1.1 WLAN and Bluetooth

WLAN transmitters, often referred to as access points (or APs), are commonly available in most electronic stores and are already deployed in many locations in sufficient numbers to be usable for positioning [5].

The other option that also would only require "off-the-shelf" hardware is to use Bluetooth-

based positioning. It should be possible to position a user using the same techniques for Bluetooth as for WLAN, and as shown by [8] it may be possible to achieve higher accuracy with this technology than with WLAN. Even so, they are both subject to the same difficulties, as Bluetooth also is a radio-based technology that uses the same frequency band. [9][10]

The biggest advantage WLAN has over Bluetooth is however that it is more widespread, with many locations already having dedicated WLAN-networks installed. Also the potential gain in accuracy is caused by the lower range of the Bluetooth device, which means that they may require deployment of more of them to achieve sufficient coverage for accurate positioning in larger areas.

1.1.2 Cellular Positioning

Another possible method to position the client would be using received signal strengths from cellular towers [5]. Unlike WLAN and Bluetooth, GSM operates on a licensed radio band which results in significantly less interference from other electronic devices using the same frequency band [11]. As with WLAN a fingerprinting approach could be used, however the median error is as high as 4-5 meters [5]. Another issue with this method would be if there are few towers within range of the area of interest it would be difficult to extend coverage by adding additional towers.

1.2 Project

The project consisted of three separate parts, evaluation, implementation and evaluation of the implementation. In the evaluation stage, previous solutions were to be explored, their viability evaluated and any potential obstacles were to be identified. The implementation stage involved choosing one of the solutions and implementing it. In the final stage the performance of the implementation was to be evaluated.

1.3 Objective

The goal of the project was to develop a system that could be used to position a user in an indoor environment; an example of an application for it (which was given by Sigma) was to help customers navigate in a mall, in order to find specific stores or even shelves. Other possible customers for the system would include companies interested in helping people navigate to the right shelf in a warehouse. However, Sigma did not have any specific application in mind, but simply wanted to get access to the technology.

1.4 Requirements

At the end of the project the following requirements were to be satisfied:

- The main requirement was a system that with reasonable reliability could estimate the position of the user, in situations where GPS and similar satellite-based positioning systems are unreliable or unavailable.
- In this case an average error of approximately 3.5 meters was considered reasonable, although the goal was to get a much better result.
- A basic functional implementation of the positioning system was to be done for the Android platform.

Apart from these requirements there were a couple of points that were not requirements for the project to be considered successful, but which should be considered a bonus:

- Using existing infrastructure and hardware.
- Cheap infrastructure and hardware.

1.5 Android application development

Since the environment that Android applications run in differs significantly from that of a program targeted at a normal PC, a brief introduction to the model follows to introduce the reader to the topic.

While Android applications are written in Java; unlike PC implementations of the language, Android does not actually execute Java byte code. Instead the code is compiled (on the PC of the developer) from Java byte code to a custom byte code format for the Dalvik VM. [12]

Furthermore, Android does not use the complete Java class library. For example, neither of the GUI frameworks, AWT and Swing, are available. Rather Android uses a large subset of the non-GUI parts of the standard library and a custom GUI toolkit. Other Android specific namespaces and classes are included as well.

The GUI toolkit of Android is based on the idea of "activities". As described by the Android SDK documentation:

> An Activity is an application component that provides a screen with which users can interact in order to do something, such as dial the phone, take a photo, send an email, or view a map. Each activity is given a window in which to draw its user interface. The window typically fills the screen, but may be smaller than the screen and float on top of other windows. [13]

Another important aspect of Android application development to keep in mind is that when an application is closed, it is in fact not terminated; rather the operating system lets the application continue to exist, until the operating system decides to reclaim the memory from it, at which point the process is killed.

In addition, each Android application has a manifest file that describes it. This includes the name of the application and any system permissions it needs in order to be able to perform its tasks.

2 Methods and tools

2.1 Methods

The viability of different solutions were to be evaluated primarily through studying papers detailing their advantages, disadvantages and difficulties, but also through experimentation and data gathering.

As Android is built for Java, this was to be the primary programming language for the application. However, performance demanding algorithms could have required C and thus the Android NDK (Native Development Kit) for implementation as it typically provides superior performance.

It was also assumed that a simple server program had to the developed in order to provide the client application with any static and/or pre-measured data needed for positioning. Depending on amount and complexity of the data stored by the server, a simple data-file might have been enough, though a relational database was thought to most likely have been required. Encryption of the communication between the server and the client was also meant to be implemented, in order to protect the users' integrity, though it was not seen as strictly required for the project.

2.2 Tools

Eclipse with the Android Development Tools plugin was used as the development environment for the Java code. Android SDK was used as the underlying layer providing the actual compiler for the target platform.

The Android platform includes the SQL-database engine SQLite (providing a custom Java API for accessing it) which was used to log data on the mobile device for the several applications that were written in order to evaluate different solutions.

When a server-based solution was needed later it was written in Java as a JSON-based web service. This solution was selected due to technical limitations in the network setup at Sigma AB. Microsoft SQL Server was used for data storage since it was already installed on the server.

2.3 Other resources

As the system was to be implemented on an Android cell phone, such a device was required for testing purposes. Sigma provided an LG Optimus 2X (LG-P990) cell phone running Android 2.2.2 as well as a Samsung Galaxy Nexus running Android 4.0.2 and a HTC Legend with Android 2.2.

Seven wireless access points of the model D-Link DAP-1360 was also provided, to give us access to APs we had control over.

Furthermore computers to do the development on were of course required and provided by Sigma.

3 Problem analysis

As described above, several possible models were to be evaluated to determine their viability. Angle-of-arrival and timing based methods where dismissed out of hand, mainly because the Android platform does not provide enough information about surrounding Wi-Fi-networks for those models to be implementable on that platform. Thus only signal strength based methods remained.

3.1 Signal Noise and Antenna Quality

A potential problem with all possible models that made use of signal strength was thought to be signal noise, unpredictable variation of the RSS (Received Signal Strength, measured in dBm) caused by surrounding radio noise and non-static obstructions such as humans moving around. This could cause any positioning model dependent on that value to yield highly varying results.

Antenna quality of the receiver was also a matter of concern. While a poor-quality antenna could reduce the RSS significantly, that part should not pose a problem as any mathematical model could simply compensate for that fact. Though, in the case of poor antennas, the transmitters would have to be better and/or more numerous in order to achieve the required coverage of the positioning system.

The big problem would lie in how a poor antenna could affect signal noise and the distribution of it. In order to test this, a smart phone and a laptop were placed next to each other, both running programs that collected data about the RSS from a specific AP over time, while in line of sight. The results can be seen in Figure 1 and Figure 2.

The same smart phone was later used to collect data about RSS from another location, with many static and mobile obstacles in-between in order to evaluate how it affected noise. The results of that can be seen in Figure 3.

As can be seen in the figures, the results from the laptop are somewhat normally distributed though left-skewed with a few outliers. However, while the results from the smart phone are not strictly normally distributed, it can still be considered a reasonable approximation of a right skewed one. Taking this into the account for the positioning system may be required. Interesting enough, having many static obstructions seemed to reduce variance and not increase it as initially thought. Determining why this is the case is very difficult, but possible reasons may be that the signal is less direct. Reflection and scattering plays a larger role in how the client receives the signal, and thus temporary interferences may affect it less. Another possible reason may be that interferences are less noticed on the already relatively weak signal. Regardless of the reason, these results show that avoiding line-of-sight to access points when positioning may yield more accurate results.

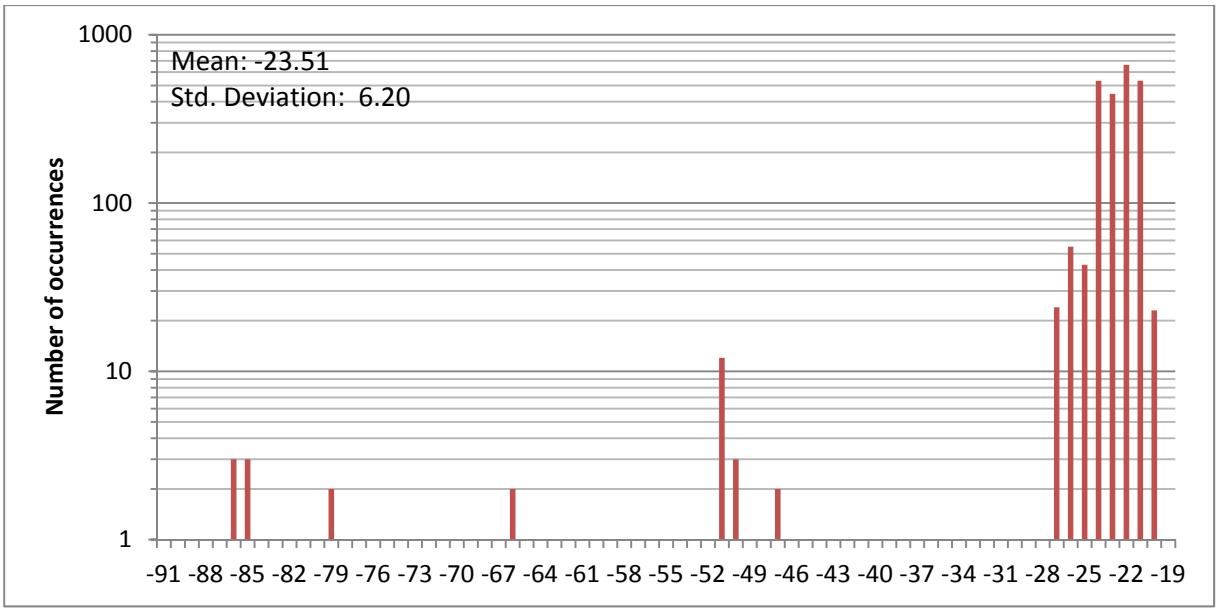

Figure 1: Histogram of signal strengths (in dBm) captured on a laptop (a Lenovo Thinkpad R500 running Linux) within line of sight from the AP. Note that the vertical axis uses a logarithmic scale. The data was captured using Wireshark (a network packet analyser software) to record the beacon frames from the AP, then the data was dumped to a text file and a shell script were used to extract the signal strength for each packet.

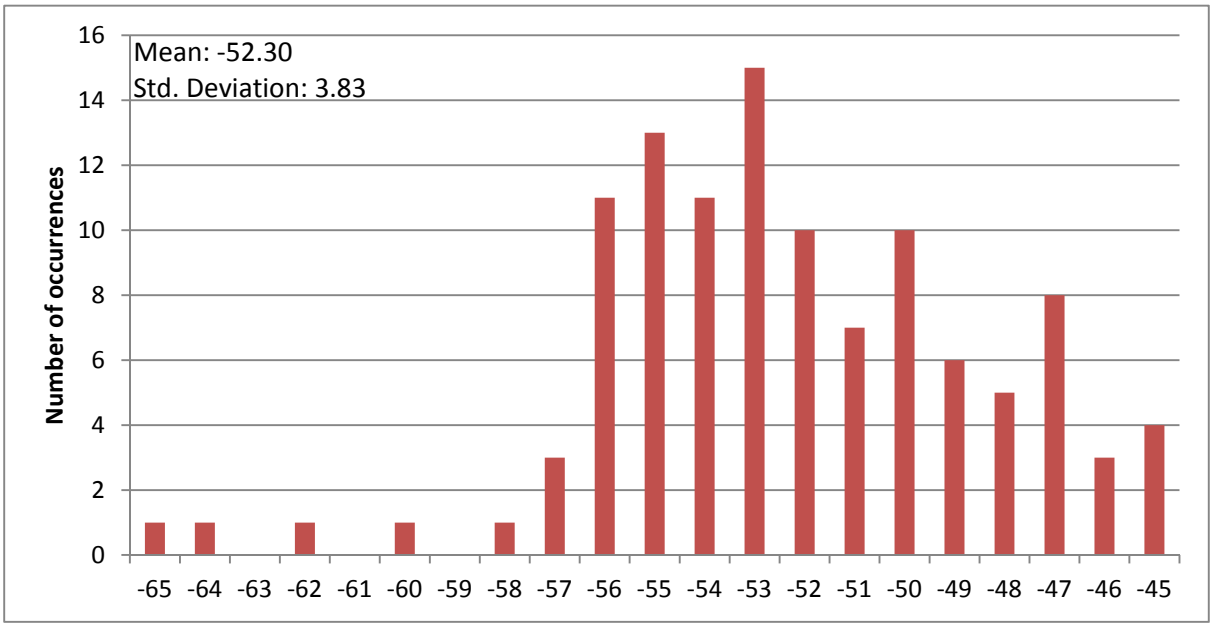

Figure 2: Histogram of signal strengths (in dBm) captured on an LG-P990 smartphone with line of sight to the AP. This diagram contains far fewer captures than the diagram from the laptop, since packet capturing is unavailable and thus scanning had to be used, which is rather slow. In both cases however the actual capturing ran for the same amount of time.

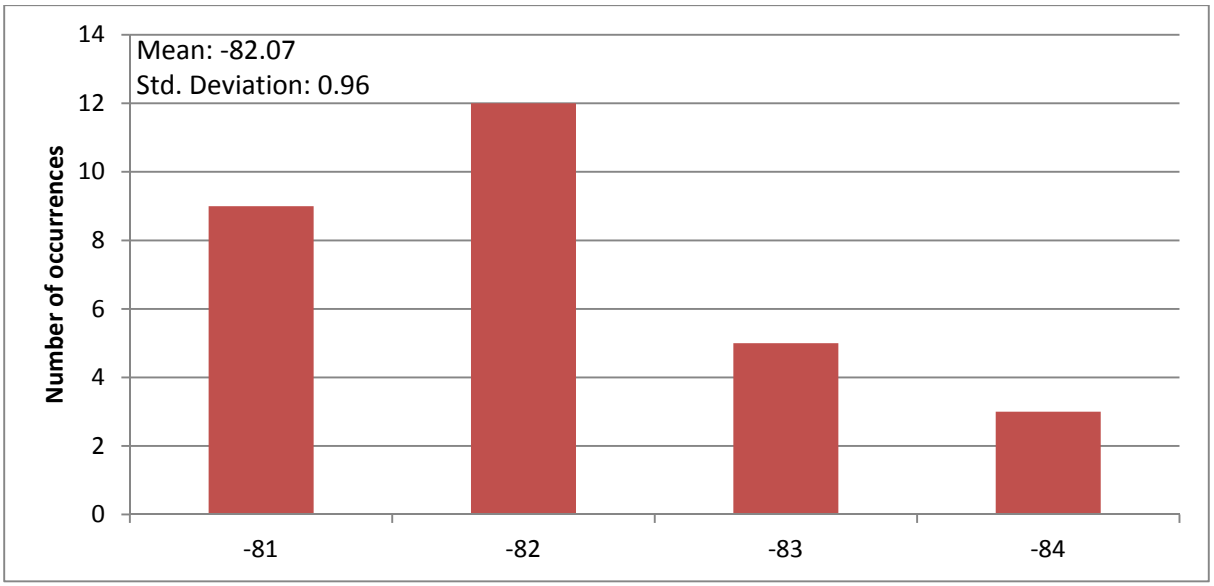

Figure 3: Histogram of signal strengths (in dBm) captured on an LG-P990 smartphone from a different floor (without line of sight to the AP). The same method of capture was used as in Figure 2.

3.2 Distance Measurements

One of the first things that was tested was the possibility of estimating the distance between a wireless transmitter and a wireless receiver by calculating it from the measured path-loss (the difference between the transmit power and the received power).

The model tested was the International Telecommunication Union's recommended model for indoor path-loss, slightly altered to predict distance from pass-loss rather than predicting pass-loss from distance. [14]

The resulting model can be described as thus:

$$\log d = \frac{1}{N}(L - 20\log f + 28 + L_A)$$

Where d is the distance between the transmitter and receiver, L is the measured path-loss; N is the distance power loss coefficient, f is the frequency and L_A is an additional path-loss factor introduced in order to compensate for the poor antenna quality of the receiver; as a high quality antenna will receive a higher RSS than a poor quality one, as shown in Figure 1 and Figure 2.

The frequency was measured to 2472 MHz, which is not surprising given the frequencies which WLAN makes use of. N was set to 30, according the recommendations from the ITU for this category of test-environment [14]. L_A was set to -20, after studying the different RSS values between a laptop and the smartphone used in these tests. While a different value for that constant might have yielded more accurate results, it would not change the underlying problem that the result from this test illustrates.

The data for the diagram in Figure 4 was captured in an office environment (thus noise from other APs and clients were present) using the LG phone. It illustrates that the estimated

distance with this model is not accurate. All data points used the median value (to reduce the effect of outliers) of five scans. Apart from the noise, it can be seen that at a distance of 4 meters, the signal was significantly stronger than at a distance of 3 meters. While the cause for this is unknown, and not easy to determine, a possible explanation might be the signal reflected off something in the environment. It should also be pointed out that the measurements at 7 meters did not have line of sight since due to walls the max distance with line of sight available was approximately 5 meters. Figure 5 contains the median RSS values used (without the L_A factor). Figure 6 contains measurements for another phone.

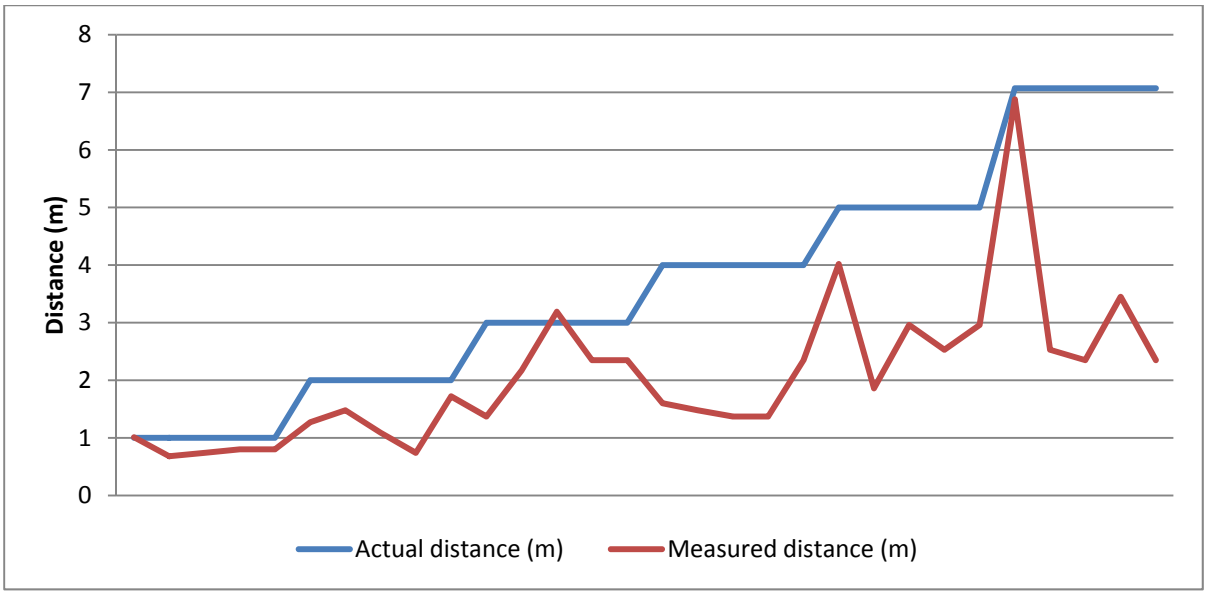

Figure 4: Distance estimation using ITU Indoor radio propagation model.

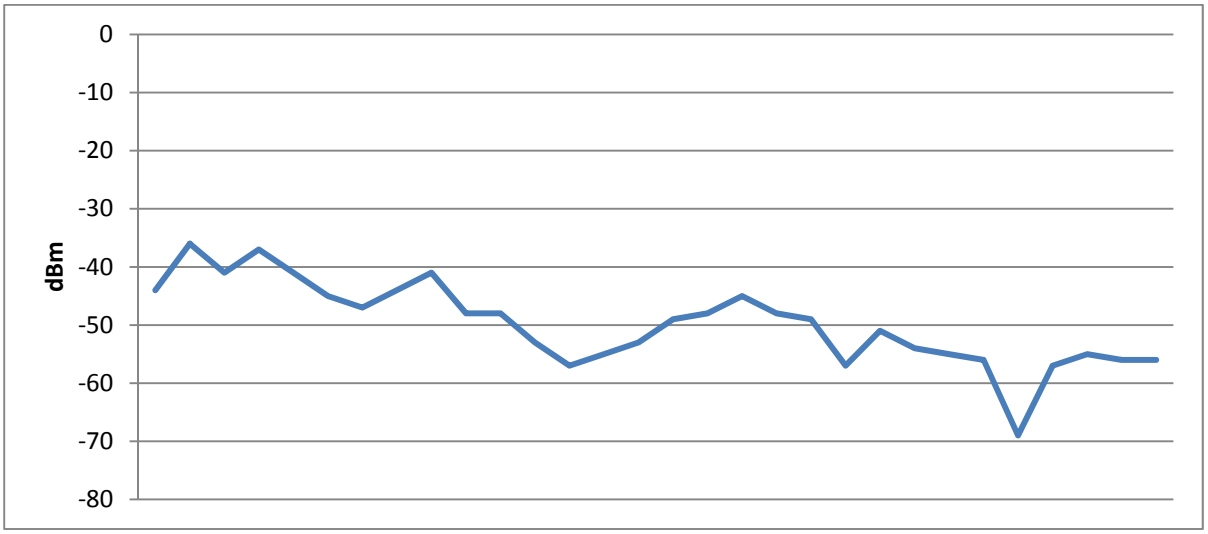

Figure 5: Raw RSS values for the measurement in Figure 4.

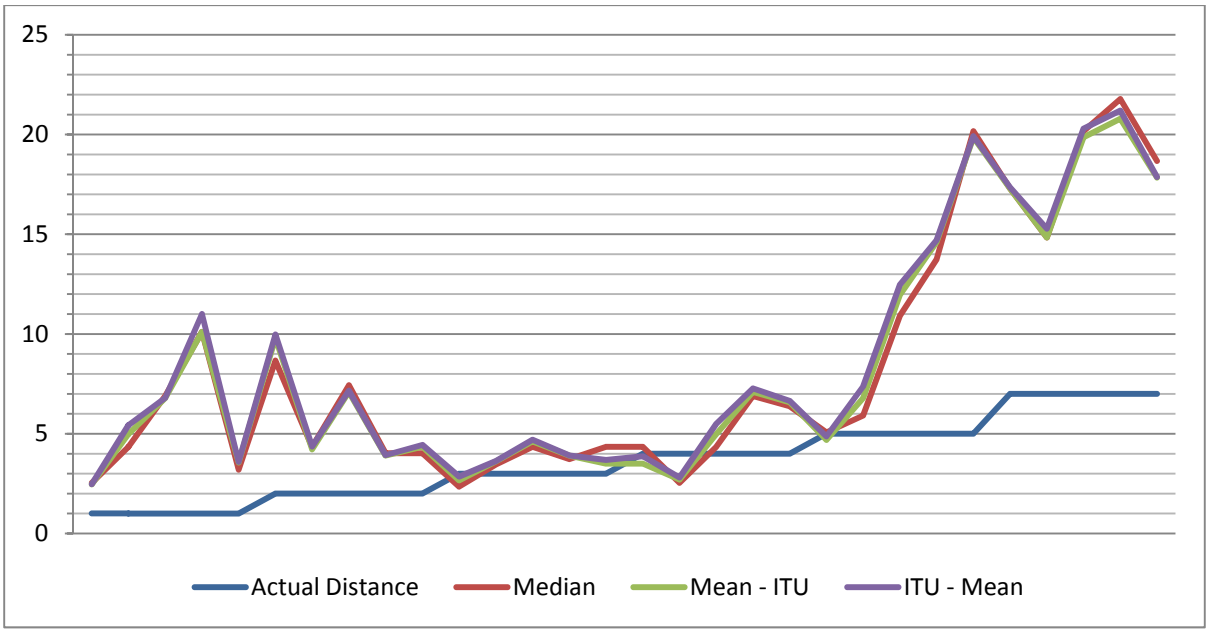

Figure 6: Estimated distance using data from the Samsung phone, using an antenna compensation factor of 0. This diagram shows different ways to calculate the distances based on the RSS values. For each data point, 5 measurements were used as input. In this case the ITU model was applied at different stages to compare these ways. In the median case the ITU model was applied on each of the 5 values and then the median was taken. In the "Mean – ITU" case, the mean of the RSS values were taken then the ITU model was applied. In the "ITU – Mean" case, the ITU model was applied on each measurement and then the mean distance was computed. While this demonstrates that an antenna compensation factor is required, as the distance was frequently over-estimated, no single value for this factor would work for all distances.

The effect of shadowing from moving objects, such as people, should be minimal in these tests as the measurements were taken in lab-environment. In a real environment this effect has the potential to make the noise even worse. Radio-propagation models often account for this using a random variable, though such a factor would be difficult to account for in a positioning system. [5]

3.3 Fingerprinting

The second viable option was a method known as 'fingerprinting'. This method is divided into two phases, a *training phase* (also known as *calibration phase)* and a *positioning phase*.

During the *training phase*, the area that should be calibrated for positioning is prepared by pre-measuring several, well-chosen, points (called fingerprints or reference points), storing the RSS values measured at each. During the *positioning phase* those values are then compared to the values measured by the client in order to determine a position.

The fingerprinting method was concluded to be the most viable method for WLAN-based indoor positioning, as concluded several others as well [4][5].

This method bypasses several problems that affect other positioning methods, making the effect of static signal obstructions essentially irrelevant, as the fingerprinting model works by

comparing differences and does not care about how high or low a certain RSS value is in absolute terms.

Several mathematical and statistical methods can also be used in order to reduce the problems posed by random factors, such as shadowing and noise caused by other sources. It would still not protect the system against large temporary disturbances; so modelling possible movement of the client may be required in order to achieve better accuracy.

3.3.1 Fingerprinting Positioning Methods

Even when it comes to fingerprinting, there are several different methods for using the data collected at fingerprints to estimate the user's position, often divided into two different categories; the deterministic and the probabilistic approaches. The defining difference between these two approaches is that the deterministic approach takes one or more measurements at each reference point and then calculates some form of single value from them, which is then used for the actual positioning. The probabilistic approach takes more of the data into account when determining the position of the user. [5][15][16]

Note that when mentioning the difference between two RSS values it is assumed that the quotient between the raw numbers is used as the RSS is given in dBm, which is a logarithmic scale.

3.3.1.1 The Probabilistic Approach

In mathematical terms, the core problem that the probabilistic approach is trying to solve is to determine the posterior probability $p(\mathrm{l}|\mathrm{o})$. That is, the probability that the observer (in practice, the observer is the device being positioned) is at location l given the observation o. Bayes rule can be used to determine the posterior distribution for the user's location [16].

$$p(\mathrm{l}|\mathrm{o}) = \frac{p(o|l)p(l)}{\Sigma_{i \in \lambda}\, p(o|i)p(i)}$$

Where $p(l)$ is the probability that the observer is at location l before making any observations, and λ is a set of all possible locations and assumed to be discrete and limited, as a continuous set of locations would be difficult to implement in practice. Several heuristics can be used to estimate $p(l)$ once a position has been determined at least once in the current execution of the positioning system, such as by incorporating distance to last point or other domain knowledge; if no such knowledge exists, $p(l)$ can be considered to have a uniform distribution. The largest challenge of the probabilistic approach is to determine the likelihood function $p(o|l)$, which gives the probability of making observation o given location l. There are several methods for this, but as this project was concluded using a deterministic approach they will only be covered briefly. [5][16]

Phone	Mean Scan Time
LG Optimus 2X	1011 ms
Samsung Galaxy Nexus	1384 ms
HTC Legend	1611 ms

Table 1: Mean time for a single Wi-Fi scan on the phones used for the experiments

Once the probability of all interesting locations has been estimated for the current positioning pass, the position with the highest probability is typically used. An alternative is to interpolate the positions with the k-highest probabilities, similar to the deterministic method of k-nearest neighbour.

The downside of the probabilistic approach is that it typically requires more measurements at a given fingerprint in order to attain enough data. Given the time it may take to calibrate a large area (see Table 1 for RSS-scanning times) a deterministic approach may often be preferable. The highest standard deviation observed, with a large series of 20 samples, was as high as 22.22, meaning that quite a few samples would need to be collected in order to generate a reasonably accurate image of the measurement distribution for any given AP.

Bellow follows a short overview of two common methods for obtaining the likelihood functions required for the probabilistic approach.

3.3.1.1.1 Kernel

One method for determining the likelihood function is known as the kernel method, where the likelihood function can be obtained through the following formula:

$$p(o|l) = \frac{1}{T}\sum_{i=1}^{T} K(o;\ o_i)$$

Where T is the number of calibration vectors, o is the observation made during the positioning phase and o_i is the calibration vector i . $K(o;\ o_i)$ is the kernel function of choice with the Gaussian kernel being one of the more popular ones. Regardless of choice of kernel function, this method can end up computationally expensive compared to the deterministic approaches as it takes into account each of the individual training samples. [16]

3.3.1.1.2 Histogram

This method uses the frequency of appearance of RSS values for each AP in use, and uses those for the likelihood function at a given location. It requires that one sets the constant k, where k is the number of 'bins' to use, to a value that should be as small as possible but large enough to cover entire possible range of RSS values. The boundaries of the bins are also a parameter that needs to be determined. There are several methods to use these values to determine the likelihood density of a particular RSS value, a simple one would be to look at a given RSS-values relative frequency. [5][16]

3.3.1.2 The Deterministic Approach

As mentioned above, the deterministic approach computes a value from the training samples which is then used for positioning; typically only the mean or median of these values are used. The deterministic approach is attractive as it typically requires fewer training samples than the probabilistic approach, but it still requires a moderate sized set of samples as too few risks skewing the results due to temporary variations and noise.

3.3.1.2.1 K-Nearest Neighbour

K-nearest neighbour is a deterministic method that involves calculating a weight based on current measurements, and possibly other factors, for each fingerprint and then selecting the k fingerprints with the best weight, where k is pre-defined a constant often set to 3 or 4. The final position is often determined through using those weights to do a weighted interpolation between the physical locations of the chosen fingerprints; this approach is often referred to as 'weighted k-nearest neighbour'. The methods used to determine the weights of each fingerprint vary as well; several are investigated in 3.3.1.3.

3.3.1.2.2 Compressive Sensing

This is another deterministic method, one that involves restructuring the problem into one of sparse nature. This enables one to apply compressive sensing, a method to restore an unknown sparse signal vector with just a few samples with minimal error [17]. The use of this method for fingerprint based positioning has been explored in depth by Au [5].

3.3.1.3 Fingerprinting Weighting Methods

If the deterministic approach is used, one often has to come up with some methodology for weighting individual fingerprints by looking at the measurement results acquired by the device to be positioned. The most straightforward one is to simply look at the Euclidian distance, the L^2 norm. Although using L^1norm, the Manhattan-distance, has been shown to provide a slight improvement in accuracy [18].
The L^2 norm is typically expressed as:

$$w = |o - o_t| = \sqrt{\sum_{i=1}^{n} \left(o_{i-} o_{t_i}\right)^2}$$

While the L^1 norm can be expressed as:

$$w = |o - o_t|_1 = \sum_{i=1}^{n} |o_{i-} o_{t_i}|$$

Where o is the vector of measurements obtained during positioning and o_t is the vector of measurement values determined from the values obtained during fingerprint calibration.

In both cases, the fingerprints with the lowest total distances are the ones that are best matched with the devices measurements. These methods all work fine when the number of access points used are the same for each fingerprint, while this is often a good idea it is not always practical and more importantly it does not address the problem of APs that are only seen sporadically from a fingerprint, which can be a significant hurdle as shown in Table 2 and Table 3. As can be seen from these tables, the detection rate of the LG phone was worse in general. However this is likely related to that it having the shortest scan time, thus giving less time for the phone to detect visible access points. This HTC phone (which has the longest scan time of the three phones) appears to have a lower quality antenna, since it was much worse at detecting the access point with the weakest signal. Even though Table 2 shows that the HTC has a better detection rate than the LG for the third access point, it performed much more unevenly, sometimes not seeing the access point at all during long periods. The LG on the other hand generally performed much more evenly over time.

	LG	**Samsung**	**HTC**
Access Point 1	82.75 %	99.75 %	100 %
Access Point 2	83.25 %	100 %	100 %
Access Point 3	44.50 %	87.25 %	57.25 %

Table 2: Detection rates for the data in Figure 7. Even though the HTC has better detection rate for access point 3 than the LG, the detection rate for the HTC varied much more over time than for the LG.

	LG	**Samsung**	**HTC**
Access Point 1	81.75 %	99.75 %	100 %
Access Point 2	88.00 %	100 %	100 %
Access Point 3	11.75 %	51.00 %	3.50 %

Table 3: Detection rates for the data in Figure 8.

An alternative to the weighting methods used above is to simply use the average difference between the RSS values measured during the training phase and those measured during the positioning phase. This handles fingerprints that use different number of APs better; you can also use the weighted difference and place different weights on different APs. This means that you can place more importance on APs that have proven themselves reliable, for instance, without fully discarding the information provided by those that are considered unreliable or unimportant.

These methods all use the absolute RSS values, which can vary significantly based on the model of device that is used for the positioning, as can be seen in Figure 7 and Figure 8. One solution to this issue would be to build one radio map for each area and each device model supported, but this can quickly become extremely costly and time consuming if one wishes to support more than just a few different models. However, as can be seen by comparing these diagrams, the values vary over time as well.

Another solution to this issue that has been suggested is to divide the APs in pairs and use the differences between the RSS values of these pairs (actually quotients, as the RSS are received in dBm; a logarithmic scale) as the values of comparison [8]. The differences between these values should be smaller between most models than the differences in absolute values [8]. The computational cost of this method is higher when using more than 3 APs however, as instead of getting one comparable value per AP one instead gets a number of comparable values equal to $\frac{n(n-1)}{2}$, where n is the number of APs in use. This increases the time complexity of the value comparison from $O(n)$ to $O(n^2)$.

There is also a possibility that this may reduce the effect of noise a bit, since when an external effect worsens the readings of one AP it is likely to worsen the readings of another AP. When pairing APs, these effects may cancel each other out. If noise changes the readings in different directions however, pairing APs may increase those effects. It is, however, unusual for the environment to not be at least somewhat noisy in all directions so this should be a minor concern at best. Experiments have proven that this does result in acceptable accuracy between devices as demonstrated in chapter 5.

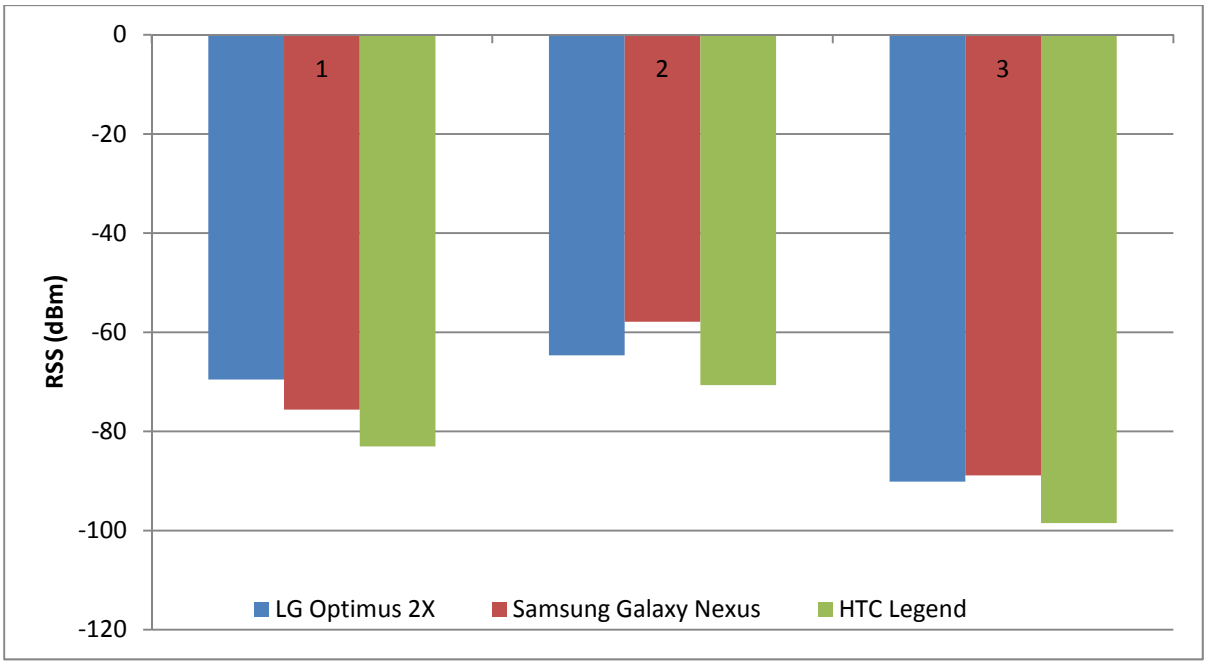

Figure 7: The average of 400 RSS readings towards three different APs, for three different phones at the same location. The phones did not always detect all access points in all scans. Since this may not always be caused by a poor signal the average does not include those data points. Also note that the data was not captured at exactly the same time, since placing the phones too close to each other might affect the radio performance, and placing them far enough apart to avoid that issue would have rendered the comparison useless.

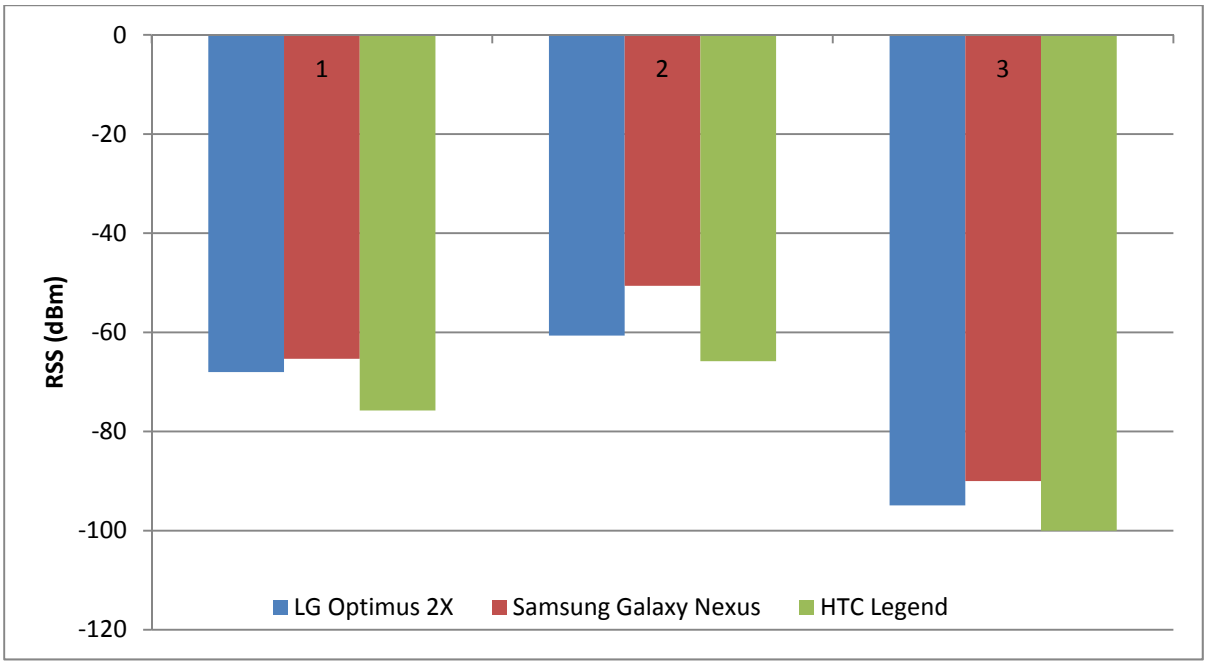

Figure 8: The average of 400 RSS readings towards three different APs, for three different phones at the same location. This was recorded at the same location used in Figure 7 but one day later.

3.3.1.4 Movement modelling

An additional issue with fingerprinting is that users tend to move around. In combination with the high chance that a device fails to observe an AP during a single scan (see Table 2 and Table 3) and with the long scan times (see Table 1) positioning a moving user becomes more problematic. If the system only uses a single scan, it is likely to miss useful data, but if it uses several scans per repositioning pass it is likely that the user moved during this time and the scan results will not reflect values measured from a single point. This makes the idea of modelling user movement an attractive one. In this section some models for this are investigated.

3.3.1.4.1 Kalman Filter

A Kalman Filter is an algorithm for determining estimates of unknown variables from measurements that vary with time, and is somewhat resistant to noise in those measurements. [19]

The filter can be split into two parts, the first being the dynamic system and the second being the filtering process.

The dynamic system should be described using the following formulas:

$$x_k = Ax_{k-1} + Bu_k + w_{k-1}$$

$$z_k = Hx_k + v_k$$

Where x_k is the new state of the dynamic system, u_k is a control signal, z_k is the measured value, w_{k-1} and v_k are Gaussian noise functions. A, B and H are domain-specific matrices.

The filtering process is itself split into two parts, the time update which is described with the formulas below:

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$$

$$P_k^- = AP_{k-1}A^T + Q$$

The second part is the measurement update, which can be described using these formulas:

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$

$$P_k = (I - K_k H)P_k^-$$

Where K_k is the current Kalman gain, P_k is the error, R and Q are noise covariances that are usually experimentally determined. The system is given an initial state, and both the time update and the measurement update are executed during each iteration. Properly used Kalman filters can serve to minimize the mean of the square error of a dynamic system. [19]

Kalman filters are commonly used for positioning and navigational problems. By using a dynamic system that describes the movements of the user for the Kalman filter, it can work

together with a measurement system to better estimate the position of the user. Dead reckoning is one such possible dynamic system, see 3.3.1.4.3. [5]

3.3.1.4.2 Particle Filters

Kalman filters have certain limitations; they assume that the system has a Gaussian distribution and that it is linear; though the last one can be overcome, with extended Kalman filters for instance. [6]

One method that can be employed in order to overcome the Gaussian distribution one is known as a particle filter. This method estimates the state of a dynamic system by using a set of N system states (known as 'particles') with weights assigned to them, where a higher N may yield a better result at the cost of requiring more computational power. The actual state of the system is then approximated by one of several methods, such as using the highest weighted particle, a weighted sum of all particles or a weighted sum of a subset of particles. [6][7]

The method is recursive and, like Kalman filters, only depend on its previous state. During each iteration, the state of each particle is updated through a model of the system with an addition of random noise in order to simulate the noise in the system being simulated. Afterwards each particle gets its weight recalculated based on the measured state. The final approximation of the systems current state is then estimated. Note that the particle weights are normalized such as that the sum of all weights is equal to 1. [6][7]

As the filter is iterated through, particles may drift a significant distance from any new measured states and a large amount of computational power may be devoted towards updating these outliers that have negligible impact on the estimated state. To handle this, particles that have drifted too far away from the measured state are periodically eliminated; often when the set of outliers have reached a certain threshold. Over time this may result in the system only using a single particle, thus new ones are usually created to take the place of those eliminated. This process is called resampling. The new particles are often duplicates of high-weight ones, though if the system model has no or very little noise this may eventually result in each particle having very similar states; there are methods to handle this however [6]. Ideally the system state estimated by the particle set after resampling should be identical or very close to the one estimated before the resampling. [6][7]

3.3.1.4.3 Dead Reckoning

Dead reckoning, that is estimating the current position based on a previously known position, a known direction and a known speed, could potentially be used to counteract lag in the estimated position when the user is moved. However experiments showed that the basic method of estimating the current speed and direction based on previous positions did not work well, more often than not noise in the signal resulted in incorrect direction and speed estimates.

The possibility of inertial navigation was investigated (combining dead reckoning with sensor data from, for example, accelerometers and gyroscopic sensors). However experiments showed that the built in accelerometer of Android phones was not precise enough to even distinguish between a person holding the phone and starting to walk, and a person standing still but putting the phone down in a pocket or on a table. Even if only the former situation was considered, the resulting speed estimate was highly inaccurate.

Furthermore, using this to estimate the direction of movement would require calculating the orientation of the phone relative the user, which might be possible using the gyroscopic sensors. However this is a complicated problem of itself, since it is not possible to assume that the user is actively looking at the phone at any specific point, the user might for example hold the phone in his/her hand by his/her side while walking.

Using an inertial navigation system with only smartphone sensors may still be possible, but is outside the scope of this project.

4 Implementation

The final system was decided to be an implementation of the fingerprinting method, for the reasons described in chapter 3. It was decided that the system should consist of four separate programs:

- A Calibration Tool for collection of RSS data during the training phase, see section 4.1. This application runs on the Android platform.
- A program to perform the actual positioning of the user. This program also runs on the Android platform.
- A program to use for planning out an area which was to support the positioning system, mainly a graphical user interface to place fingerprints.
- A server application that handled storage of data gathered during the training phase and communication of said data to any of the two client applications above.

4.1 System Phases

As all fingerprinting systems, the system had to be divided into two different distinct phases, a *training phase* and a *positioning phase*.

During the *training phase*, a map of the area that was to support the positioning system had to be defined in a custom vector-based format. The reason for this are two-fold, first it was to provide a simple but clear graphical user interface for the calibration tool, the positioning application and the planning application but the main reason was to support systems that made use of user-movement modelling.

Next, fingerprints had to be predefined with the planning application. The main reason for this was to support careful planning of the fingerprinting radio map. Finally, data had to be gathered from these pre-defined fingerprints using the calibration tool and then uploaded to the server by going out and taking these measurements at the fingerprint point.

As the RSS readings can vary by the orientation of receptor, RSS readings were taken in four different directions as suggested by Au [5]. For familiarities sake, the directions where called *North, East, South* and *West* though they did not correspond to the magnetic directions of the same names. It is not important which direction these actually point towards, but they should be the same for all fingerprints on a given map, and they should each be separated by 90°. This was thought to be sufficient to collect all required information about the WLAN signals at that point. Once all of this is

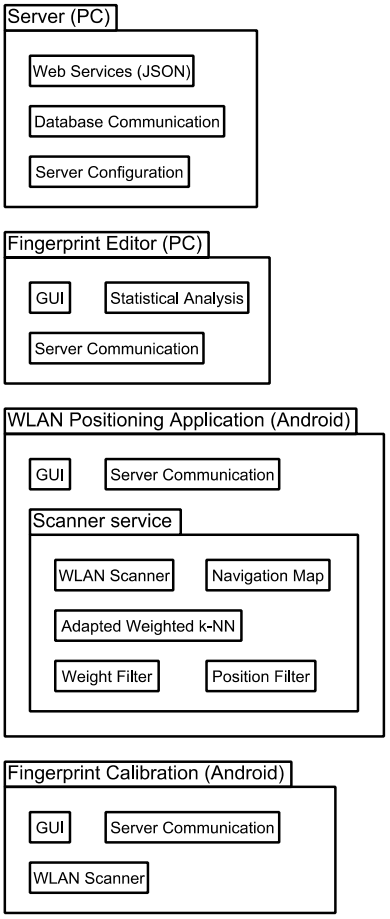

Figure 9: An overview of the system architecture, the applications that make up the entire systems and their functions.

done, one could proceed to the *positioning phase*.

In the *positioning phase* the positioning client is informed by the server in which supported area it is (if any), only given a choice if several areas are probable. It is then sent all required information about that area. Now the client can begin to position itself, which is all handled automatically by the positioning system. See 4.3 for details on how this is implemented.

If an area changes dramatically, such as heavy metal shelves being moved, walls being removed or added or possibly even with the moving of a lot of furniture the *training phase* might need to be redone as the radio signature of the fingerprints might have changed greatly. Not redoing it runs the risk of losing accuracy in the system.

4.2 Architecture

As described above, the system consists of four different applications that each serve different purposes. For an overview of these and their functionality see Figure 9. Below follows a more detailed explanation of their structure and functionality.

4.2.1 Server

The server provides web services that allow the mobile clients and the fingerprint editor to access and modify the map data. The server stores the data in a database management system (DBMS). The data interchange format JSON was used for providing the data since it was simple to handle on Android.

4.2.2 Fingerprint Editor

The fingerprint editor, which runs on normal personal computers rather than phones, is used to place fingerprints on the map. The editor also includes a feature which provides statistical information for the recorded data from the mobile devices. This can then be used to evaluate which APs to use for the actual positioning.

4.2.3 Fingerprint Calibration Application

The fingerprint calibration tool consists of a WLAN scanner, a server communication module and a GUI that allows the user to record data for fingerprints, selected from those on the areas map.

4.2.4 WLAN Positioning Application

Finally the WLAN positioning application consists of a GUI, a server communication module (shared with the fingerprint calibration application) and a scanner service. The scanner service handles the actual positioning. The WLAN scanner module from the fingerprint calibration application is used here as well to retrieve the RSS measurements from the Android operating system.

Figure 10 shows the general flow of operation in this application. When the application is started it first retrieves a list of plausible maps from the server (based on the currently visible access points). Then the user can select one of these, or in case there is only one, it is automatically selected. The web server used does not support the HTTPS protocol, and

regular HTTP is used so this communication is unsecure. For a deployment of this system, in an environment other than the test-bed used in this project, it is recommended that the underlying webserver is changed to one that supports the HTTPS protocol so that this communication does not reveal the general location of the user against their wishes by revealing which map is downloaded.

Once the map is loaded the application will pre-process the measurement data sent to it from the server, and generate a navigation map from the vector map data that can be easily used to determine the 'walk-distance' between two points. It consists of a grid, where each cell is flagged as either blocking or not. The cell density is configurable; but a higher density results in higher costs when one estimates the distance between two points as more cells has to be considered. Using a density of 3x3 nodes per square meter was determined to give a good compromise between detail and computational cost. On the non-blocked spaces the distance from the point of interest (the previous location estimate) is computed using Dijkstra's algorithm, thus making the Manhattan walk-distance to the previous position available to later stages of the positioning system. Due to the limited computational power of phones the algorithm will only walk up to 7 meters away from the point of origin, leaving the distance to cells beyond that unknown. This is generally not a problem as the user is highly unlikely to walk more than 7 meters in a single repositioning pass.

Using the observed signal strengths, the pre-recorded map data (the fingerprints) and the navigation map, each fingerprint is then assigned a weight. In this stage the navigation map is used to check if a fingerprint (reference point) is too far away (more than 7 meters) or even unreachable, in which case the fingerprint is ignored. This helps prevent the estimated position from jumping through solid walls or extreme distances and also to reduce the amount of computation required. (It should be noted that due to the limited resolution of the navigation map, it is important to ensure that a fingerprint is far enough away from a wall as to not appear to be inside a wall on the navigation map. The fingerprint calibration application contains sanity checking code for that very purpose.)

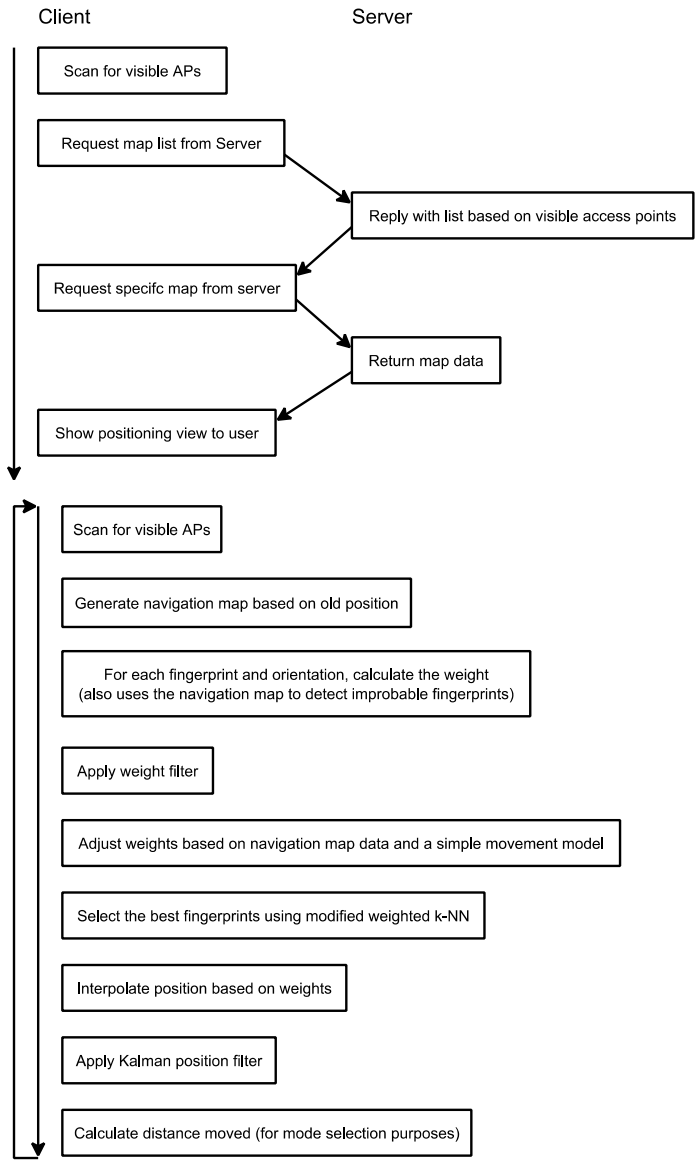

Figure 10: Operation of the WLAN positioning application.

For details on how the weights are determined and how the algorithm works see 4.3.

After the weights have been calculated, they are passed through a simple filter that provides some resistance to sudden and improbable large weight changes. This improves the system accuracy slightly, allowing it to resist large amounts of noise.

Using the navigation map and a simple model for how far a walking human can move, the weight of fingerprints beyond the probable walk-distance is slightly increased as it is improbable that the user has managed to walk that far.

A modified version of weighted k-NN is then used to select the most likely fingerprints. The modification consists of selecting the most likely fingerprint, then generating a new navigation map with the distances from that point. The new navigation map is then used to prevent the algorithm from selecting additional points too far away from the most likely one. In the current implementation up to the three most likely are selected, though fewer may end up being used if not enough of them are in range. Once the most likely fingerprints have been selected, their positions are interpolated, using their weights, yielding an estimated position of the user. Too reduce the effect of noise on user position and make it appear more accurate; very slight changes in position are thrown away at this stage.

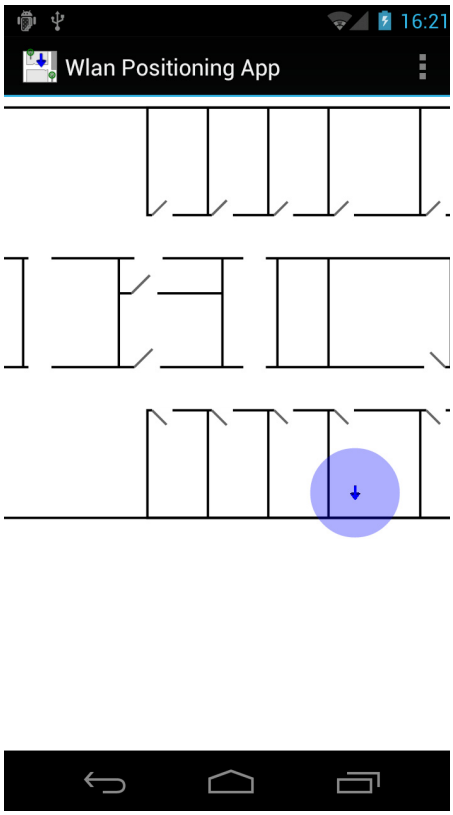

Figure 11: The positioning application in action

Finally, this new position is fed into a Kalman filter, this filter should in general have a better dynamic model to estimate user movement, but due to the issues encountered with such models (see 3.3.1.4.3) it assumes that the user is simply stationary. It does improve accuracy noticeably even with this assumption.

The positioning phase described above has several modes, giving more precise or less precise results. These differ in the number of scans they perform in the first step, before doing the calculations. A model with more scans is used when standing still in order to increase the precision. To reduce lag when moving fewer scans are done in this case. The mode is selected by looking at how the position has changed recently, with large shifts resulting in the assumption that the user is moving around.

4.2.4.1 Access Point Selection

Another aspect of the system that was considered was how to choose which APs to use for the actual positioning, while the calibration tool gathers and sends data about all APs it can find not all of them are used in the actual *positioning phase*. Computation time is certainly one factor, but the fact that many of the access points have unknown reliability is an even more important consideration, as you cannot be certain that any visible AP will continue to be

visible or change physical location. This will cause the reliability of the system to degrade over time, assuming that the system does not limit itself to use only APs that are known to be reliable.

In the implementation of this system, the server only sends data about APs that are flagged as "usable" to the positioning client, thus avoiding this degradation effect.

4.3 Algorithm Details

As mentioned before, the final algorithm is a deterministic one; mainly in order to reduce labour costs in the training phase. It can be split into several parts, each contributing a bit towards the final position. It has been tested to work reasonably well with a varying number of scans (see section 5.1) during the positioning phase, and some parts behave slightly different based on the number of scans done.

Regardless of the number of scans per positioning pass, only a single RSS value per AP in use is utilized for the positioning. The RSS used is the mean of all RSS values read from that AP during this pass. Note that only the number of actual values read is used for this and not the number of scans as the number of scans can be higher than the number of values read from any given AP. This is necessary as the chance of an AP being seen might vary significantly from device to device, as seen in Table 2 and Table 3. Only APs that were not seen at all are given special treatment; in which case they are assigned a mean RSS of -110, which is lower than any readings taken from any device.

Note that the mean is used in all cases as it has a slightly lower standard deviation than the median, as seen in Figure 12, in spite of the distribution of RSS values not being Gaussian.

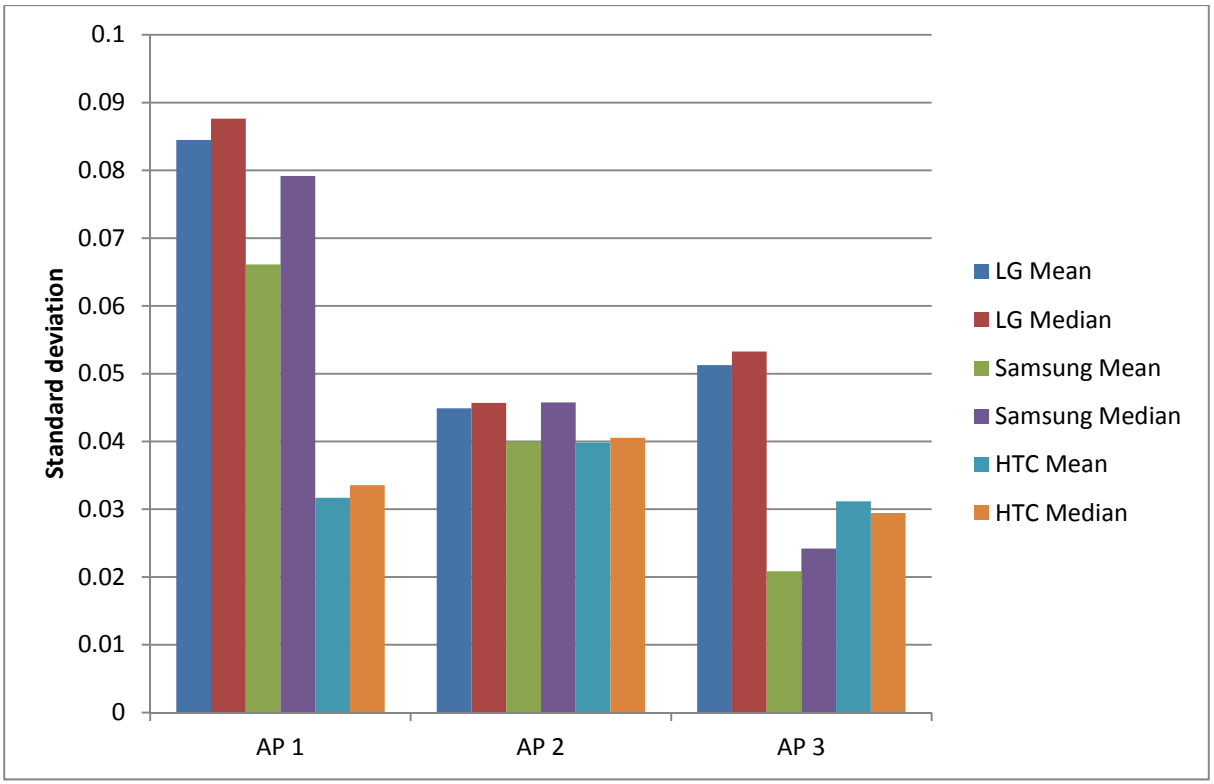

Figure 12: Standard deviation per phone, using mean and median pairings. The measurements were taken from the same location for three different APs.

Also note that the algorithm itself is explained here, ignoring implementation details such as how walk-distance is calculated or which data types are used.

Each orientation that has had measurements taken for it is considered to be a separate fingerprint for the purpose of this algorithm.

4.3.1 Fingerprint Weight

Each fingerprint is pre-processed before any positioning is done, where the AP readings of all APs seen from that fingerprint are divided into $\frac{n(n-1)}{2}$ unique pairs. The difference between the mean RSS values from those are then recorded, as well as which APs where involved in that pair and in which order they were applied.

In the *positioning phase,* for each positioning pass the first thing the algorithm does is to calculate a weight for each fingerprint; if the previous user position is unknown all fingerprints are considered valid, otherwise only the fingerprints within 7 meters Euclidian distance from the user's previous position are considered. This serves mainly to limit the computational cost of the algorithm, as it would otherwise scale with area size; leaving large enough areas too expensive to apply this algorithm to. Should every fingerprint be disqualified for any reason, every fingerprint is then considered qualified as it must mean that the last position has ended up somewhere where it should not be able to.

The weighting method of this algorithm uses is the weighted average difference variant,

mainly to better handle APs missed when doing just a few scans per re-position pass. The weight of a given fingerprint, w_{fp}, is calculated through the following formulas:

$$w_{fp} = \sum_{i=1}^{n_p} \frac{|d_i - d'_i| w_{p_i}}{w_{tot}}$$

$$w_{tot} = \sum_{i=1}^{n_p} w_{p_i}$$

Where d_i is the difference between the RSS values of training pair i and d'_i is the difference between the RSS values of live pair i. n_p is the total number of pairs. w_{p_i} is the weight of pair i and is always 1 unless the number of scans done for this repositioning pass is just 1, in which case w_{p_i} is 1/10 if any of the APs RSS values are -110 (i.e. not seen at all). This is due to the high chance of missing even reliable APs when doing just a single scan, and thus minimizes the consequence of that miss. Whether a reading was received or not may be an important part of a fingerprints identity, so they must still be given some weight.

Note that this gets you the weighted average difference between the pairs training values and live values, and thus the fingerprint with the lowest weight is the one which the user most likely is closest too.

4.3.2 Weight Filtering

In order to dampen the effects of large, sudden, changes in fingerprint weight the weights of all fingerprints are passed through a simple filter, where they are adjusted based on their previous values. This filter only fully trusts new values if it has nothing to go on from before, such as when one is being positioned for the first time. The new weight of a fingerprint, w'_{fp}, is determined through the following formula:

$$w'_{fp} = w_{fp} + R(w_{fpm} - w_{fp})$$

Where R is en experimentally determined constant which in this implementation it was set to 0.7.

4.3.3 Navigational Map Filtering

After the weights have been adjusted by the filter, they are passed through a second filter that adjusts them based on data available from the navigational map, based on the following formula:

$$w'_{fp} = w_{fp} D$$

$$D = \begin{cases} 1, & d_f \leq 1 \\ \sqrt[n]{d_f}, & d_f > 1 \end{cases}$$

$$d_f = \frac{d}{d_w t}$$

Where d_w is a constant that denotes the user's probable walking speed and t is the time since the last update, d is the walk-distance to the fingerprint from the user's last known position. Finally, n is an empirically determined constant that was set to 3 for this project. This filter is

ignored if the user has no last known position.

4.3.4 Conditional Weighted K-Nearest Neighbour

At this point, the weights of each fingerprint for this pass have been finalized. They are sorted in order and the most likely fingerprint is selected, that is, the one with the lowest weight. An additional k-1 fingerprints are selected, in ascending order according to their weights, from the set of fingerprints that are within a certain empirically determined distance from the most likely fingerprint. This distance was set to 2.5 m for this project. The parameter k was set to 3 in accordance to the recommendations of Li, et al. [4]. The reason for this condition is that noise may on occasion give excessively strong weight to a fingerprint far away from the other most likely ones, likewise regions with a low density of fingerprints gain in accuracy when selecting fewer of them [4].

The position of the user is interpolated through the following method:

$$p = \sum_{f \in \varphi} \frac{\frac{1}{w_f} p_f}{\sum_{f' \in \varphi} \frac{1}{w_{f'}}}$$

Where φ is the set of all selected fingerprints, including the most likely one. Should w_f or $w_{f'}$ have the value 0, they are replaced with a very small value instead.

If the new position is less than 0.5 meters way (Euclidian distance) from the old position, the new position is discarded and the old position is reused. This improves apparent accuracy by not slightly shifting the position every repositioning pass, but may not necessarily improve actual accuracy.

4.3.5 Position Kalman Filter

As the final step of the algorithm, the newly estimated position is passed through an adapted Kalman filter. The filter should use a good dynamic model on how the user moves, as that may dramatically improve accuracy. However, for reasons discussed in 3.3.1.4.3 the Kalman filter in this project simply assumes that the user is stationary.

While particle filters were considered for this project, they could be very computationally expensive [5]. In addition they would require a somewhat accurate user movement model, which as mentioned above was not available.

The Kalman filter noise covariances R and Q (see 3.3.1.4.1) were both set to 2. While initial error was also set to 2. The adjustment however, is that R changes from update to update. An additional factor is added to its base value, with the factor being calculated through the following formula, which was experimentally determined:

$$R^+ = \sqrt{d}\left(\frac{d}{d_w}\right)^2$$

Where d is simply the distance between the last and the new position. Note that the new value is not kept between passes. This addition only has a minor negative effect on users walking at a normal pace, but reduces max error by quite a bit as it the large position shifts that can occur when measured values are changed due to a short burst of highly increased noise. If there is

no last position to be had, this filter simply returns the measured position.

The position determined by this filter is the one the user is considered to be at.

5 Result

Here are the empirical results of testing the positioning system described in chapter 4; all experiments used the same radio map, where 15 measurement samples had been taken for each fingerprint and each of the four directions.

5.1 Methodology for accuracy measurements

To perform accurate measurement when measuring the error of the estimated position versus the real position a method was used wherein the phone was positioned at a known location, measured using trigonometry from various corners with known coordinates (based on a floor plan that was provided by Sigma). The coordinates estimated by the positioning system was then recorded. For stationary tests 10 positions were used, and 5 estimated positions recorded at each, in varying orientations. The measurements were taken during normal work hours, with people moving in the area. See Figure 13 for the spatial layout of the measurement locations.

Only the Samsung Galaxy Nexus and the LG Optimus 2X were used in practice due to limited resources on the HTC Legend making it near unusable with the resource-heavy algorithms used. The fingerprint map for the location had been recorded using the LG Optimus 2X phone 10 days earlier. 15 access points were used for the positioning.

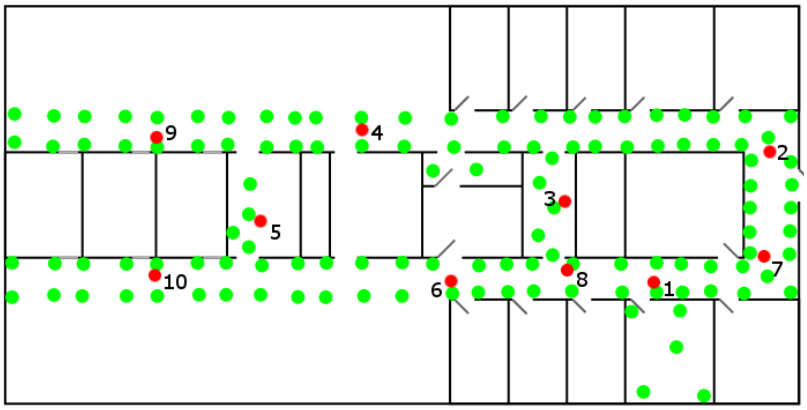

Figure 13: The fingerprint map with test positions. The red circles are the approximate test positions used. The green circles are the fingerprints.

5.1 Stationary tests

These tests were performed by a stationary user at the pre-defined points in Figure 13.

5.1.1 Three scans

First, the phone was set to use the result of three scans for each positioning pass. The results of these accuracy tests are in Table 4.

Phone	Mean	Median	Minimum	Maximum	Std. Deviation
LG	1.26 m	0.94 m	0.15 m	5.20 m	1.18 m
Samsung	1.52 m	1.28 m	0.25 m	4.54 m	0.98 m

Table 4: Accuracy results taken from 50 different positioning attempts, each done with three scans.

It should be noted that these values varied considerably from point to point as can be seen in Figure 14. Although the LG in general had the best result, there were some outliers, with 3.17 meters being the worst mean error for a given point.

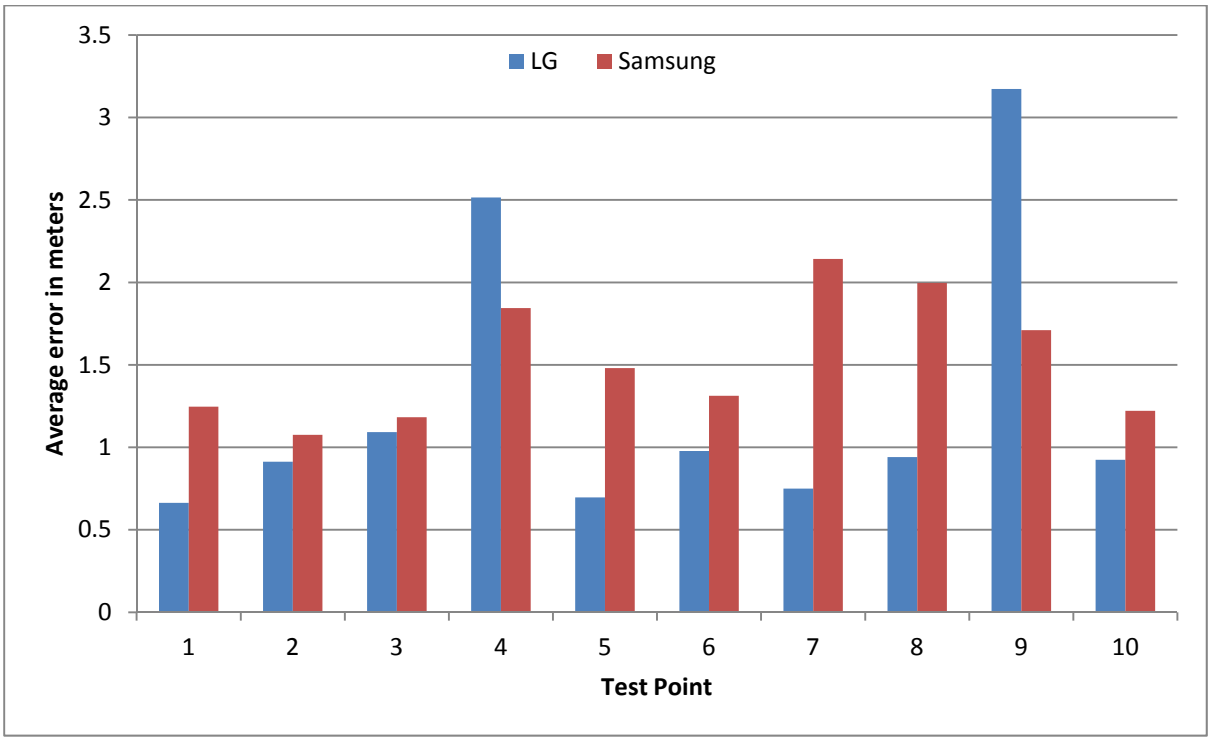

Figure 14: The mean error for each test location when using three scans per positioning pass.

5.1.2 One scan

When using only one scan per positioning pass the result were as expected worse. This model is normally used when the system estimates that the user is moving, though in this case they were still taken from a stationary user. See Table 5 for the results.

Phone	Mean	Median	Minimum	Maximum	Std. Deviation
LG	1.45 m	1.12 m	0.12 m	4.33 m	0.92 m
Samsung	1.67 m	1.25 m	0.25 m	5.26 m	1.17 m

Table 5: Accuracy results with one scan.

As can be seen in Figure 15, the tenth location has the worst mean error for these measurements.

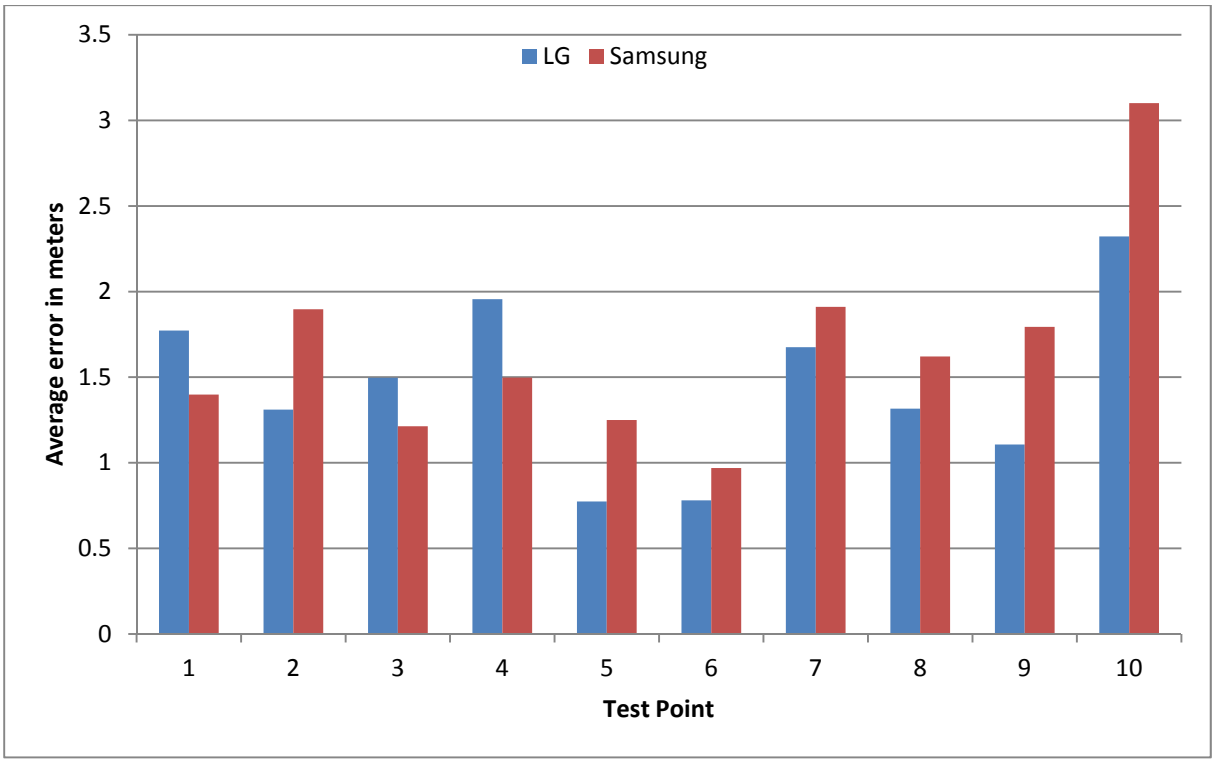

Figure 15: The mean error for each test location when using one scan per positioning pass.

5.1.3 Analysis

Analysing the results above indicate that location 4, 9 and 10 all have less precision than most of the others locations. All three are positioned in a noisy environment with a significant number of people moving around, along with a lot of other electronics. That these only showed up as outliers in one or the other of Figure 14 and Figure 15 is likely just due to chance: when people happened to be moving around or not.

Adding more reference points around these locations or adding extra access points near such locations could mitigate these issues, the latter would however require re-recording a large set of fingerprints to account for the newly visible access point in any location it can be seen from. If the issue is people moving around, placing access points in the ceiling or other tall locations where moving people are less likely to obstruct the signal could also reduce the issue. Yet another possible solution might be to capture a set of alternative fingerprints at the problematic locations, for example, one set recorded during the peak time and one set when the location is empty or near empty. Even selecting a different subset of the access points seen when calibrating the location might substantially improve the result, though possibly at the expense of getting problem locations elsewhere instead.

Since the LG had been used to capture the fingerprint map, the smaller error of that phone is not surprising. However, as demonstrated in Table 2 and Table 3, the Samsung phone has a better hit rate, which could give a slight advantage compared to the LG when only one scan is used. In practice however, this effect is somewhat reduced since the implementation lessens the impact of a missed access point by assigning those a lower weight than the others in the one-scan model, see chapter 4.3.

5.2 Mobile tests

These tests were conducted in much the same manner as the stationary ones, with the exception that they were not done on a pre-defined set of points and that the tester moved around a bit before position was taken and recorded.

The system was used fully here, with it trying to adapt the number of scans to user movement as explained in 4.2.4. Thus, an individual position estimation may have been done with either one or three scans. See Table 6 for the results.

Phone	Mean	Median	Minimum	Maximum	Std. Deviation
LG	2.35 m	2.2 m	0.05 m	4.96 m	1.49 m
Samsung	2.12 m	1.89 m	0.47 m	4.73 m	1.22 m

Table 6: Mobile user accuracy results

5.2.1 Analysis

As expected, given that the system constantly assumes a stationary user, the results are worse though still within an acceptable and useful level. It should be noted that almost every data point lagged behind the user, which is expected. Since the position updated roughly once every second, when the measurement was taken relative to the last position update was observed to have significant effect on the margin of error.

One can conclude that a full movement model would have to be implemented in order to attain good accuracy for a moving user, but that it's still usable in the current state.

6 Discussion

The system provided for surprisingly good accuracy for a user that stood still with both a single scan and with three scans, beyond what we first expected when we started working on this project. While we had hoped for better precision for a mobile user; we also know that the movement models, that all expect the user to remain stationary, sabotage this by having the positioning system lag a bit behind the actual position. Tests ran before all these systems were fully implemented resulted in a mean error of 1.7m for a mobile user, suggesting that it might be a good idea to reduce the impact of these systems or remove them entirely in situations where the user is highly mobile. Then again, these issues may be completely resolved by implementing a model that does not assume a stationary user.

Regardless, it is still useful as it is in any environment where high precision is not required. Should it be used in a shopping mall, for instance, it can still guide a customer to a given store without trouble, as the individual stores tend to enjoy more spacing between each other than then side rooms in the office environment we used as our test environment.

It is also unfortunate that we did not have time for a more extensive test; it would have been interesting to see how the system performed in a more realistic environment. But given how noisy the test bed was, with many people coming and going, a lot of electronics active and a large amount of radio noise due to the presence of quite a few Wi-Fi networks, the results should not be far from reality except for in the most ill-suited environments.

6.1 Further improvements

Due to time constraints, several features that could improve the system was considered but never implemented. Here, they are covered in broad detail along with what advantages they would have.

6.1.1 Clustering

The system described in this report only considers fingerprints up to a cut-off distance away from the previous estimated position in order to make the system run fast enough to be usable on phones. This generally works well, but sometimes causes the system to get stuck at an incorrect location. The cause of this is that somehow the user moved outside the evaluated area, for example by walking in one direction and random noise causing the estimate to move in another direction. Then if the incorrect location has a local maximum of the weight function (the maximum of the evaluated area) and the user stops moving, the system gets stuck at that position.

One way to solve this would be to use a set of pre-generated clusters and use a classification algorithm as a coarse localization stage to select the most likely clusters. This would reduce the amount of computation needed without the risk of the system getting stuck at a local maximum. [5]

6.1.2 Improved Navigational Map

The current implantation of the navigational map of an area, as mentioned in section 4.2.4, converts the vector map into a cell-based map with a provided density and then uses Dijkstra’s algorithm to calculate distances. While this does give acceptable results, it has the

limitations that it cannot handle fingerprints placed too close to a wall as those may end up associated with the same node as the wall itself and will thus be treated as inaccessible. Any calculated distance between two cells is also the shortest Manhattan distance between those cells, as opposed to the shortest Euclidian distance which is usually more desirable.

An alternative implementation that would resolve both issues would be to make use of 'navigational meshes' with path smoothing, as it would both provide a close-to optimal Euclidian distance with obstacles taken into account and allow one to place fingerprints near walls without risking them being placed inside the wall vector. Implementing such a system could also result in faster distance calculations, provided that the map obstacles do not result in too complex a set of navigational meshes. [20]

6.1.3 Kalman Filter and navigation paths

While the Kalman filter implemented in this system uses the navigation map when it estimates the error of the measurements, it does not use it when it determines the final position. Instead it uses a Euclidian interpolation between those points, while this generally gives good results it can cause problems around corners where the interpolated position ends up in another room. Even though the system typically does not have a problem recovering from this, it is still an undesirable behaviour.

One approach towards solving this issue is not to use the raw distance for this interpolation, but instead interpolate along a path, i.e. a new point that is halfway between the last position and the new measured one would be halfway along the shortest path between these. This was not really implementable due to how the navigational map was implemented, as using a cell based approach for this would necessitate that the new position is on a cell rather than at an arbitrary point in space. A better map implementation could make this alternative viable.

6.1.4 Complementing Inertial Navigation System

Many smart phones come with one or more sensors, such as digital compasses and gyros, which can be used to implement an inertial navigation system. While such a system would likely have limited utility on its own, due to cumulative errors, it could be used as extra input to estimate the user's movement before each repositioning pass after the initial one to improve the accuracy of the system further, possibly by using it as a model for the Kalman Filter rather than the current one where the user is assumed to be stationary.

6.1.5 Probabilistic Pairing

One alternative that could potentially provide for better accuracy without disregarding the cross-phone compatibility that the current deterministic approach provides is to adapt one of the probabilistic methods with the pairing method, as the probabilistic method may have better accuracy than k-NN [16], though not in every case [5].

Instead of comparing absolute values seen from a given location, pair differences would instead be compared. As mentioned in 3.3.1.1, it would increase calibration time for an unknown change in accuracy. Having access to a probabilistic system with higher accuracies may be of value for areas where accuracy is far more important than calibration time. Further evaluation is recommended.

6.1.6 Fictitious fingerprints

One alternative to manually adding more fingerprints in area with poor accuracy might be fictitious fingerprints [8]. In this method extra "fictitious" fingerprints are generated based on the real fingerprints. A quick test of this, using a simple interpolation model between the three nearest real fingerprints with line of sight to each other, was performed in some problematic areas in the test environment; however this did not lead to any noticeable improvement in accuracy. Using a more complex algorithm for the generation might help. Further study is needed to answer this question.

6.2 Grading criteria

In order to comply with university rules, the following section discusses how well the grading criteria of the course were satisfied.

- Wide knowledge within the relevant field of technology and in depth knowledge of the specific technology and methods used for the thesis:
  In depth research of radio localisation and several related fields (such as compressive sensing, probability theory and statistics) was performed. Mathematics was used throughout the work in order to describe the theory behind the system, evaluate alternative solutions and analyse the results.

- Problem analysis and evaluation of solutions:
  Several models were evaluated to find the best solution for localising a smartphone using off-the-shelf hardware. The reasons for selecting WiFi were explained, and the advantages and disadvantages with cell tower, Bluetooth and WiFi based solutions were examined. The advantages and disadvantages with probabilistic and deterministic approaches were also discussed.

- Scientific quality and research:
  We had very little help with research from our supervisor, only getting a few ideas for starting off at the very beginning. While these proved useful, a significant amount of time was spent on finding and reading peer reviewed papers. Except in a few cases where peer reviewed resources were not available (such as when describing the limitations of the Android platform) the references used were of scientific quality.

- Using and relating to existing technology and solutions:
  Indoor positioning using WiFi is a relatively new area of research, though other radio based positioning systems exist and are a well explored area of research. Due to the limitations of the Android platform and the WiFi infrastructure we could not make use of technologies such as positioning by angle of attack or by using timestamps.
  Existing solutions that were applicable (such as the signal strength difference pairing) were researched and used.

- Planning:
  While more time would have been welcome in order to further improve the system, we managed to meet the requirements within the allotted time. The time schedule worked fairly well, although it had to be adjusted after the first couple of weeks.

- Team work:
  Due to the nature of the project we worked fairly independently from the normal projects at Sigma, distributing the workload between us by ourselves. It was fairly evenly divided.

- Discussing ideas:
  We have discussed ideas between ourselves through the entire project. At several points we also discussed problems and solutions with our supervisor at Sigma as well as contacting experts within mathematics and other fields at the university.

- Report and presentation:
  A significant amount of work was spent on the report and presentation in order to ensure it was easily accessible for not just experts in the field of WLAN positioning. A basic understanding of computers and engineering is of course still required.

- Scientific and social aspects and insight into the potential and limitations of the technology:
  Some possible applications for the technology developed in this thesis are described, though there could of course be many more use cases discovered in the future. Several possible future improvements to bypass the current limitations (such as problems with handling moving users) are discussed.

- Expanding knowledge:
  During this thesis we had to learn several to us previously unknown areas of science, such as new mathematical techniques and radio propagation models. While there is always more to learn, we did manage to learn enough to solve the problem and satisfy the requirements.

- Professionalism and documentation:
  We cooperated with Sigma, and adjusted our choices of technology to fit their infrastructure. During the weekly meetings with our supervisor at Sigma we discussed what we had done last week and what we were going to do. It was also interesting to see what the other students were doing. Furthermore the solution was well documented, the theoretical background in this thesis, and the specifics of the implementation in the form of Javadoc in the code as well as an architectural overview document.

7 References

[1] Wan Bejuri, Wan Mohd Yaakob; Mohamad, Mohd. Murtadha; Sapri, Maimunah, *Ubiquitous Positioning: A Taxonomy for Location Determination on Mobile Navigation System.* Signal & Image Processing : An International Journal(SIPIJ) Vol.2, No.1, 2011/03, pp 24-34.
Downloaded 2012-02-17
URL: http://airccse.org/journal/sipij/papers/2111sipij03.pdf

[2] *Android 4.0 Compatibility Definition*. Google, Inc, January 2011, pp. 20-21
Downloaded: 2012-05-11
URL: http://source.android.com/compatibility/4.0/android-4.0-cdd.pdf

[3] Bose A & Foh C, *A Practical Path Loss Model For Indoor WiFi Positioning Enhancement.* Singapore: Proceedings on the 6th international conference on Information, Communications & Signal Processing, 2007.
ISBN: 978-1-4244-0983-9

[4] Li, Binghao; Salter, James; Dempster, Andrew G.; Ri, Chris, *Indoor Positioning Techniques Based on Wireless LAN.* Sydney: University of New South Wales, 2006.
Downloaded: 2012-02-20
URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.72.1265&rep=rep1&type=pdf

[5] Au, Anthea; *RSS-based WLAN Indoor Positioning and Tracking System Using Compressive Sensing and Its Implementation on Mobile Devices.* Toronto: University of Toronto, Department of Electrical and Computer Engineering, 2010 Master of Science Thesis
Downloaded: 2012-02-20
URL: http://www.wirlab.utoronto.ca/wirlab/thesis/Au-Anthea-WS-201011-MASc-thesis.pdf/view

[6] Arulampalam, Sanjeev; Maskell, Simon; Gordon, Neil; Clapp, Tim, *A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking*. IEEE Transactions on Signal Processing, vol.50, no.2, pp.174-188, Feb 2002
Downloaded: 2012-05-29
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=978374&isnumber=21093

[7] Rekleitis, Ioannis, *A Particle Filter Tutorial for Mobile Robot Localization TR-CIM-04-02*. Québec: McGill University, Centre for Intelligent Machines, 2004
Downloaded: 2012-05-29
URL: http://www.cim.mcgill.ca/~yiannis/ParticleTutorial.html

[8] A. K. M. Mahtab Hossain, H. N. Van, Y. Jin, & W.-S. Soh, *Indoor Localization Using Multiple Wireless Technologies*. IEEE International Conference on Mobile Adhoc and Sensor Systems, 2007. MASS 2007, October 2007, pp. 1 –8. Singapore
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4428622&isnumber=4428592

[9] *Specification of the Bluetooth System Core 3.0 + HS*
Bluetooth SIG, 2009
Downloaded: 2012-05-08
URL: https://www.bluetooth.org/Technical/Specifications/adopted.htm

[10] *IEEE Standard for Information technology – Telecommunications and information exchange between systems Local and metropolitan area networks – Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, New York: The Institute of Electrical and Electronics Engineers, Inc, 2012, pp. 1566
ISBN 978-0-7381-7245-3

[11] Varshavsky, Alex; de Lara, Eyal; Hightower, Jeffrey; LaMarca, Anthony; Otsason, Veljo, *GSM Indoor Localization*. Pervasive and Mobile Computing Journal (PMC), Vol. 3, No. 6, December 2007, pp. 698–720.
Downloaded 2012-05-09
URL: http://www.sciencedirect.com/science/article/pii/S1574119207000478

[12] *What is Android?* Google, Inc
Downloaded: 2012-04-05
URL: http://developer.android.com/guide/basics/what-is-android.html

[13] *Activities*. Google, Inc
Downloaded: 2012-04-05
URL: http://developer.android.com/guide/topics/fundamentals/activities.html

[14] *Recommendation ITU-R P.1238-7*. International Telecommunications Union
Downloaded: 2012-04-05
URL: http://www.itu.int/ITU-R/index.asp?category=publications&rlink=latest-recommendations&lang=en

[15] Youssef, Moustafa; Agrawala, Ashok; *The Horus WLAN Location Determination System*. International Conference On Mobile Systems Applications And Services, Volume: 14, Issue: 3, Publisher: Springer, pp: 205 – 218 2005
Downloaded: 2012-05-10
URL: www.cs.umd.edu/~moustafa/papers/horus_usenix.pdf

[16] Roos, Teemu; Myllymäki, Petri; Tirri, Henry; Misikangas, Pauli; Sievänen, Juha *A Probabilistic Approach to WLAN User Location Estimation*. International Journal of Wireless Information Networks Vol. 9 No. 3, July 2002, pp 155-164
Downloaded: 2012-05-10
URL: http://kom.aau.dk/group/05gr999/reference_material/basis_of_ekahau/ijwin02.pdf

[17] Candes, E; Wakin, M; *An Introduction To Compressive Sampling*. Sign Processing Magazine IEEE, vol.25, no.2, pp.21-30, March 2008
doi: 10.1109/MSP.2007.914731
URL: http://ieeexplore.ieee.org.db.ub.oru.se/stamp/stamp.jsp?tp=&arnumber=4472240&isnumber=4472102

[18] Li, B.; Wang, Y.; Lee, H.K.; Dempster, A.; Rizos, C. *Method for yielding a database of location fingerprints in WLAN*. Communications, IEE Proceedings, Vol. 152, Issue 5, October 2005, pp. 580-586
Downloaded: 2012-05-10
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1522067&isnumber=32550

[19] Welch, Greg & Bishop, Gary, *An Introduction to the Kalman Filter*. University of North Carolina at Chapel Hill
Downloaded: 2012-04-05
URL: http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf

[20] van Toll, Wouter; Cook, Atlas F.; Geraerts, Roland, *Navigation meshes for realistic multi-layered environments*
Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, vol., no., pp.3526-3532, 25-30 Sept. 2011
doi: 10.1109/IROS.2011.6094790
Downloaded: 2012-05-12
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6094790&isnumber=6094399