Localization in GPS-Obstructed Environments with Android Devices

D A V I D   C H A N G

KTH Computer Science
and Communication

Master of Science Thesis
Stockholm, Sweden 2012

Localization in GPS-Obstructed Environments with Android Devices

D A V I D   C H A N G

2D1021, Master's Thesis in Computer Science (30 ECTS credits)
Degree Progr. in Computer Science and Engineering 270 credits
Royal Institute of Technology year 2012
Supervisor at CSC was Hedvig Kjellström
Examiner was Danica Kragic

TRITA-CSC-E 2012:093
ISRN-KTH/CSC/E--12/093--SE
ISSN-1653-5715

Royal Institute of Technology
School of Computer Science and Communication

KTH CSC
SE-100 44 Stockholm, Sweden

URL: www.kth.se/csc

Abstract

Localization has many useful purposes and applications, but is in many cases reliant on the Global Positioning System, which might not always be accessible. In this project, an alternative localization approach using dead reckoning and WLAN localization is suggested as a complement to the traditional GPS where the satellite signal is obstructed. The dead reckoning approach utilizes step detection and a dynamic step length, modeled as a relation between step period and step length through linear regression. The WLAN localizer is a particle filter with a discrete state space. The two location estimates are then fused in a Kalman filter for a final estimate. The test results indicate that this implementation is an improvement over GPS even when a GPS signal is accessible but disturbed. An average deviation of 5-10 m from the correct position makes this implementation suitable for wide indoor spaces, such as shopping malls, warehouse buildings, etc, but not necessarily office environments with small rooms.

Referat

Lokalisering i GPS-blockerade miljöer med Androidenheter

Lokalisering har många användbara tillämpningsområden, men är i många fall beroende av GPS som inte alltid är tillgängligt. I denna rapport föreslås ett alternativ som utnyttjar död räkning och WLAN-lokalisering, som ett komplement till den traditionella GPS-lösningen i områden där satellitsignalen är blockerad. Död-räkningsmetoden använder sig av stegdetektering och en variabel steglängd som ett samband mellan stegperiod och steglängd. Sambandet hittas med minstakvadratmetoden. WLAN-lokaliseraren är ett partikelfilter med ett diskret antal tillstånd. Dessa positionsuppskattningar kombineras genom ett Kalmanfilter för att få en slutgiltig positionsuppskattning. Testresultat visar att denna implementation är en förbättring gentemot GPS även när en GPS-signal är tillgänglig men störd. En genomsnittlig avvikelse på 5-10 m från den korrekta positionen gör denna implementation passande för stora och öppna inomhusmiljöer som till exempel köpcentrum och lagerbyggnader men inte nödvändigtvis för kontorsmiljöer med mindre rum.

Contents

1 Introduction **1**

2 Related Work **3**
- 2.1 Dead Reckoning . . . 3
  - 2.1.1 Heading Direction . . . 3
  - 2.1.2 Distance . . . 4
- 2.2 Error Correction . . . 5

3 Theory **11**
- 3.1 Global Coordinate System . . . 11
- 3.2 Normal (Gaussian) distribution . . . 11
- 3.3 Linear Regression . . . 12
- 3.4 Kalman Filter . . . 13
- 3.5 Particle Filter . . . 14

4 Implementation **19**
- 4.1 Equipment . . . 19
- 4.2 Dead Reckoning . . . 19
- 4.3 WLAN Localization . . . 24
- 4.4 Combining Dead Reckoning, WLAN and GPS . . . 27

5 Testing **35**
- 5.1 Testing Conditions . . . 35
- 5.2 Results . . . 35
  - 5.2.1 Dead Reckoning and Kalman Filter Performance on a Cyclic Course . . . 35
  - 5.2.2 WLAN Convergence . . . 36
  - 5.2.3 Randomly Walking . . . 37

6 Discussion **43**

7 Conclusion **45**
- 7.1 Summary . . . 45
- 7.2 Future Work . . . 45

Chapter 1

Introduction

The Global Positioning System (GPS) is a well-established method for determining the absolute location of an object, in terms of global coordinates. However, the method does rely on a satellite signal, which cannot be guaranteed under all circumstances or in all locations. For example, the satellite signal can be obstructed by ceilings and walls in indoor environments, or tall surrounding buildings in an outdoor location.

The purpose of this project is to examine the alternative positioning techniques or a combination of several, for satellite signal-obstructed areas. The project will be carried out with an Android device in mind, as devices of this kind are currently regarded as everyday accessories. They are also equipped with the sensors needed for measuring the acceleration and the orientation of the device. These measurements will be useful, or even crucial, for the *dead reckoning* (DR) approaches that will be examined and eventually required for the final implementation. Dead reckoning is a collection of various techniques for determining an object's current location based on movement from a known start position, making it a relative positioning technique. A drawback and a consequence of dead reckoning, being a relative positioning approach, is the error accumulation over distance. Error sources could include systematic errors in the sensors, sensor inaccuracy, measurement noise or perhaps even the positioning algorithms themselves. Also, sloped terrain can cause overestimated position displacements, if not taken into consideration. To address these issues, we (any reader who is going to set up a localization system like this one) need to find ways to correct the errors.

One approach to limit the error accumulation in dead reckoning is feeding updates of correct and absolute position values to the positioning system. This could be done periodically in terms of distance or time, or whenever a reliable absolute position is available. Temporary GPS signals and Wireless Local Area Network (WLAN) access points (APs) are two examples of this, and we will look further into the alternative positioning techniques using wireless access points. By fusing several positioning methods, we prevent one source of error from having too large of an impact on the position reporting.

The goal of the project is a prototype application for Android devices for determining the position of the device in areas where GPS is not available. The system needs to operate in real-time. We will also have to consider the demands on the user when using the system. Considerations like whether it should be required that the user holds the device in a fixed orientation in relation to himself or herself and the movement direction, or if the user is free to place it in a pocket that can cause undesired motion readings. The system will be naive in the way that it cannot detect if the user is deliberately trying to trick the sensors.

Chapter 2

Related Work

2.1 Dead Reckoning

There are a few different approaches to dead reckoning. The approach is generally to accumulate motion; the heading direction and the distance traveled in that particular direction. Sensors for acceleration and orientation are used, and the placements vary depending on the dead reckoning approach.

For pedestrian scenarios, they can for example be placed on a helmet [4], on the lower back and centered around the waist [13], in the trousers' pocket [16, 25], on the shoe [3], or be handheld [15]. Jirawimut states the compass can be placed anywhere on the body because of the compass bias correction, in his conclusion [13].

2.1.1 Heading Direction

The heading direction is a crucial estimate for dead reckoning systems as incorrect estimations would lead the user off-course and yield an inaccurate position.

Orientation sensors such as gyroscopes can be used to retrieve the rotation of the body. The rotation angle is used to rotate the body coordinate system to the world coordinate system before adding the position displacement [3, 11]. The solution in [3] proved to handle side-stepping and backward movement adequately, but it requires the sensor device to be fixed to the user's foot.

In [4], magnetic compass measurements and GPS were used separately and then compared, and the position offset turned out to be about 10 m after 1 km. The authors claim the path offset would have been absent if they had fused the *pedestrian dead reckoning* (PDR) and the GPS measurements up to the entry of the building [4]. The authors point out that the orientation given from a compass is not necessarily the same as the direction of travel, and that the compass is attached in a fixed orientation in relation to the body [4].

In [13], the heading is read from the yaw signal of the magnetometer, and is corrected with a compass bias value that is in turn continuously adjusted by a compass bias error. The compass bias error is predicted and corrected with a *Kalman filter* (KF) that receives GPS input whenever such a signal is available.

The correction approach turned out to give more accurate and consistent results than both the GPS and a Kalman-filtered GPS approach. In the test without the GPS signal, it is clear that the heading was affected by magnetic field disturbances and a deviation from the GPS reference is noticeable.

Both [13] and [4] show the importance of frequently correcting the position estimate from dead reckoning and keeping the accumulated error low by updating the absolute position, whenever possible. This is what we aim to do with periodic updates from GPS and WLAN access points, and we will return to this topic later.

The heading direction can also be retrieved from acceleration measurements through *principle component analysis* (PCA) [16, 25]. In [16], the user is assumed to face forward while walking relatively straight. PCA will yield the motion axis that is parallel to the movement direction, but not yield the forward direction itself. Kunze et al. suggest integrating over the acceleration, and determining the forward direction from the positive integral. On the other hand, integrating over the acceleration did not yield a "robust estimate", perhaps because of the mentioned 180° ambiguity [25] caused by the leg backswing. Instead, the authors in [25] chose to align the rotation axis, for the positive angular movement just before foot impact, to the right side of the body. Rotating this rotation axis 90° will give an approximate forward direction for comparison [25]. This is also examined as an independent method to determine the heading, without the use of PCA [25]. Four PCA approaches were examined: 2D acceleration axes (PCA2D), 2D acceleration axes with a low-pass filter (PCA2Df), 3D acceleration axes with the components projected to the horizontal plane (PCA3Df), and finally, PCA on the gyroscope measurements (gyroPCA). The advantage of a PCA-based approach is that it can detect backward and sideway movement, using a known body orientation as a reference.

The PCA2Df approach proved to be the most accurate one with an average median orientation error of 5.7° across 23 traces from 8 different persons for a total of 30 km trace length. Results of device placements in both the right and the left trouser pocket were compared to results of a statically fixed sensor that worked as ground truth, and it is noted that the results vary for different pocket shapes and sizes. The experiment over a 30 m straight path in [16] showed a mean of 5° and standard deviation of 2.5° when compared to the reference GPS results.

2.1.2 Distance

There are different approaches for retrieving the position displacement depending on how the device is moving. In a pedestrian scenario, various kinds of step detection, step length estimation and step counting can be used to calculate the traveled distance. To detect steps, a commonly used approach takes advantage of the fluctuations in the acceleration, which means setting a threshold value and using the peaks of the acceleration that exceed it [3, 4, 15, 25]. Another approach is to use the pitch measurements from an orientation sensor, when wearing the sensor around the center of the waist [13]. For this degree project, using a handheld Android device, it is unclear whether pitch will be retrievable considering that we then would have to demand that the user attaches the device in a static orientation in relation to her own body coordinate system.

Following the step detection, the step length needs to be determined. The step length can be pre-determined and static, if the step length variance (across the step lengths of one person or several) is low enough to give an accurate velocity and while the angle error is expected to be the largest source of error [25].

On the other hand, step lengths can vary plenty depending on the environment, for example in crowded areas where the user is forced to walk slower with shorter step lengths. Therefore Jirawimut et al. present a variable step length approach using an extended Kalman filter to continuously correct the step length error. The corrected step length error is used to update the step length that in turn is used for the velocity calculation [13].

Beauregard and Haas suggest estimating the step length from a neural network using interpolation between GPS position fixes [4]. It is also pointed out that the step length from the neural network is given for one individual only, hence only that individual can use the trained step model with accurate results. The total distance deviation is "only a few percent" and the authors attribute the result to the neural-network-based step length estimation [4].

Kröger et al. propose calculating the distance between the left and the right heel, using the standard deviation of the acceleration [15]. The study presented no results for distance deviation as the experiment focused on detection of different movement states rather than positioning accuracy.

Our system and its testing will focus mainly on *pedestrian dead reckoning* and indoor use to begin with, but using a purely step-based approach would rule out all other forms of movement, including undetected movement that we want to include to preserve accuracy, for example horizontal escalators. Therefore, an acceleration-based approach utilizing double integration for calculating the position displacement is an option [3, 24]. On the other hand, double integration can lead to inaccuracies because of sampling losses and requires a high and consistent sampling frequency to mitigate the losses [24]. Furthermore they claim the double integration approach is useful as long as the displacement precision is not extremely critical, and is light in terms of computational load, but these properties are likely the results of this specific implementation not using any floating point calculations [24]. On the contrary, double integrating acceleration values turned out successful in the case of [3], where it outshone the GPS reference in sharp turns.

The different approaches for detecting vertical movement in [3] and [15] *might* also be of interest as the project progresses.

2.2 Error Correction

Since dead reckoning is very susceptible to errors and error accumulation, we need methods to control the error accumulation and to keep it low, preferably under a certain threshold. As mentioned earlier, one method is to feed periodic updates of correct absolute positions to the system. The more trusted positions can be retrieved from for example a temporary GPS signal or various positioning methods using WLAN access points with known locations. The advantage of positioning with WLAN access points is that a WLAN is often already in place with the technical properties necessary for positioning, hence does not require any additional installations and therefore is a cost-efficient option [23].

There are many different approaches for indoor positioning with WLAN access points, but they can be divided into three major categories: 1) triangulation that uses the geometry between the access points and the target, 2) scene analysis (or location fingerprinting), or 3) proximity (or cell-of-origin) [12, 18, 20, 23].

Triangulation can be divided further into lateration and angulation techniques [20]. Lateration techniques make use of the "distance" between the access points and the target device. This distance value can be estimated from a signal strength value [18] or the signal propagation time between the target device and each of the access points [12,20,23]. The time-based approach is considered to be more accurate than the signal strength and cell-of-origin approaches, but requires all the access points and the target devices to be very precisely synchronized in time [12, 23] and does even require a specific kind of network infrastructure that is not commonly established in contemporary WLANs [12].

Time of arrival (TOA) is a lateration technique that uses the intersection of at least three circles (for 2D positioning) to estimate the position, where the radius of each circle is the distance from the respective access point to the target device. The accuracy is about 4–5 m, using signal strength for distance estimation [18].

Time difference of arrival (TDOA) is a relative approach in contrast to the absolute TOA approach, which bases its position estimation on the differences in TOA between the measurements from an access point to the extra reference points and to the target device [20].

Measurements of both TOA and TDOA are susceptible to errors caused by multipath propagation of signals, which means the signals can be delayed by taking alternative routes [20].

Angulation uses the angles formed between the angle direction lines originating from reference points to the intersection and the straight lines back to the access point [20]. The advantage of angulation is that only one reference point for each desired dimension in the position estimate, is required [20]. Although no precise time synchronization is required here, the angulation approach requires specific and complex hardware for handling the *direction finding* [20]. Angulation is also affected by multipath propagation in the way that the signals can come from misleading angles [20].

Scene analysis, or location fingerprinting, is an approach which makes use of signal strength values to known reference locations stored from a training phase. The reference data are then compared to the live measurements to estimate the location of the device. The signal strength values are measured between the target device and each individual access point [12, 18, 20, 23]. Two seemingly common approaches are the *k nearest neighbors* (KNN) approach and the probabilistic approach [18,20].

Neural networks, support vector machines (SVM) and smallest m-vertex polygon (SMP) are also mentioned as alternative approaches [20].

The user's position can be estimated by choosing the closest reference point in terms of Euclidean or Manhattan distance as the estimation (the nearest neighbor approach, NN) [12, 18]. The Euclidean or Manhattan distance is calculated in signal strength units rather than actual distance units. In k nearest neighbors, the coordinates of the k closest reference points are used to calculate an average that works as the estimated position [18]. Li et al. emphasize a balanced k and state that the average of the three or four nearest neighbors will generally give the best results, as selecting too few neighbors will ignore useful information while too many will include too distant values and hence skew the average [18].

In short, the probabilistic approach estimates the location based on the probability of being in a specific location (one of the reference points) given a specific set of signal strength readings [18, 20]. The simplest form of the probabilistic approach assumes that all locations are equally probable when performing the localization. To improve the accuracy, information about previous locations together with the possible future locations within a certain time frame can be used to narrow down the alternatives [18]. In contrast to the KNN approach, the probabilistic approach can only assume discrete positions [20].

Experiment results show that using some form of KNN is the most accurate in general but no method is consistently the most accurate, as the most basic NN method prove to be the most suitable for low-resolution grids [18]. Comparing the probabilistic approach to the NN method with $k = 2$, the results show a slight improvement for static localization and a great improvement for mobile localization, both tests in favour of the probabilistic approach [18]. However, the comparison in [18] would have been more interesting and useful if the three or four nearest neighbors had been used to represent the NN method instead, as it is explicitly stated and shown earlier in the same article that they were the better choices of k for NN in general.

Results also confirm that a higher number of reference points gives a better accuracy up until a point where the increase eventually stagnates [18]. In order to achieve adequate accuracy with fingerprinting, a dense grid of reference points is required and that will lead to a large training phase [18]. It is also noted in [18] that the reference point database has to be updated to reflect the new signal strengths each time any change to the environment can affect the signal strengths. To facilitate the training phase, the spatial information in the reference points can be utilized in interpolating between neighbors, to create a denser database of reference points [12, 18, 19].

Proximity, or cell-of-origin, relies on knowing the ID and the absolute position of the access point one is connected to. It is then also assumed it is the closest access point and from there the position can be estimated [12, 20, 23]. This might yield a quite inaccurate result if the signal range of the reference point is long and the grid of reference points is sparse. The MAC addresses of a device can be used for identification [12] and the advantage of this method is its simplicity [12, 20, 23]. In the case of overlapping signals, the device can just connect to the access point giving the strongest signal [20], but devices do not automatically switch to the connection with the strongest signal according to the WLAN standard [23]. In summary, cell-of-origin appears to be simple low-cost option that suffers in terms of accuracy and flexibility.

The major drawback for positioning by WLAN is that the accuracy is very susceptible to disturbances, and obstacles in the line of sight is a problem affecting all the WLAN positioning methods, in one way or another.

Both [20] and [5] suggest hybrid solutions in their conclusions to cover the flaws of the respective techniques. A hybrid method of fingerprinting and trilateration proved to be a significant improvement to trilateration alone, but slightly worse than fingerprinting with nearest neighbor and a medium training phase (33 reference points in a 11×23 m^2 test area) [17].

Measurement values can be affected by noise or inaccurate sensors. The Kalman filter is an algorithm to estimate a series of (more likely) values given a series of measurement values and information about confidence and noise. It works in two phases, predicting and correcting, where the algorithm continuously updates the prediction and the confidence we have in that prediction. It weighs the algorithm's own prediction against a measurement and depending on the confidence levels in either values, a weighted output value (a new prediction) is produced [28]. A Kalman filter might be an option for fusing dead reckoning, GPS and WLAN location estimates in this project.

In [14], a Kalman filter for integrating dead reckoning, map matching and GPS positioning was proposed. The method relies on dead reckoning to track movement from a known starting position, and uses periodic updates (in distance or time) from GPS positioning or map matching to correct the accumulated errors from the dead reckoning tracking. Map matching depends on having distinct and recognizable movement patterns to be able to match them to a map. Therefore, problematic situations for map matching include long straight roads and "undetected vehicular movement", such as ferry crossing, towing, etc. It is presumed that GPS positioning and map matching are accurate for different situations and therefore complete each other by correcting the position when the situation suits each method the most. The map matching correction is said to be the most suitable solution for areas with many distinct turns or movement patterns. In contrast, the GPS system will be able to update the position along long and straight roads in open environments where the number of tall buildings that obstruct the GPS signal is assumed to be significantly lower. The results show that dead reckoning and map matching alone can contain the accumulated errors within 20 m in areas with several turns while the error increases to as much as 60 m where map matching cannot be performed. However, with a GPS signal the error can be contained within 20 m throughout the entire 13.5 km long test route.

Sloped terrain can also be a source of error and can be addressed with rotation matrices to rotate and match the body coordinate system with the world coordinate system before adding the displacement [3, 11]. However, this would require the sensor to be attached in a fixed orientation in relation to the coordinate system of the moving body. This requires too much effort from the user and is not a viable approach at this point.

Chapter 3

Theory

This chapter presents the theoretical concepts used in this project.

3.1 Global Coordinate System

To localize an object on earth, global coordinates are needed. A global coordinate consists of two components, latitude and longitude. These are measurements given relative to two prime meridians, the equator and the Greenwich meridian. Latitude and longitude measure angles and the unit for both is degrees. The latitude measurement originates at the equator (0°) and spans 0°-90° to the north and 0°-(-90°) to the south. The longitude measurement originates at the Greenwich meridian (0°) and spans 0°-180° to the east and 0-(-180°) to the west.

3.2 Normal (Gaussian) distribution

Probability distributions are models for displaying the probable behavior of a stochastic variable in a set of data. For example, what values the variable is most likely to assume given a known set of conditions. The probability distribution of importance in this degree project is the *normal distribution* (also known as the *Gaussian* distribution). It is a continuous probability distribution with a bell curve-shaped probability density function which has a single peak centered at the expected value μ and a spread determined by the standard deviation σ. This single-peak property is also referred to as *unimodal.* The probability density function is as follows [6]:

$$p = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.1}$$

where p is the probability of x occurring.

The normal distribution is a very common probability distribution among measured data, and it can often be assumed that measurement errors are of approximately Gaussian nature as well. Another reason for using the distribution is that it is also easy to parameterize with μ and σ only. Gaussian curve example in figure 3.1.

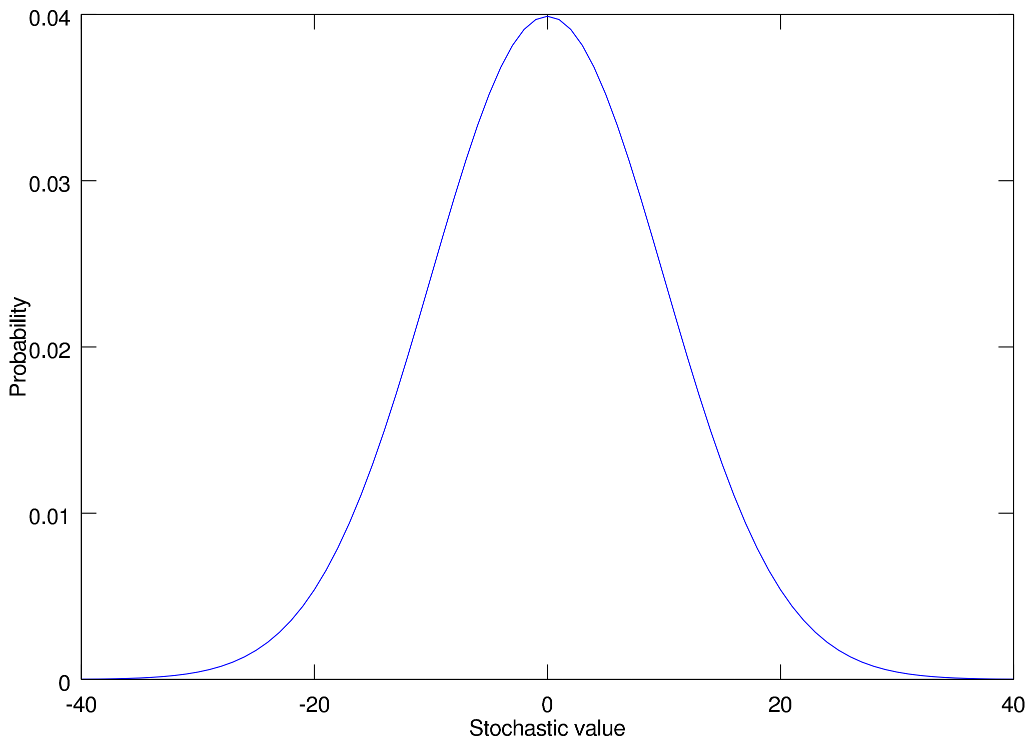

Figure 3.1. *Gaussian curve with* $\mu = 0$ *and* $\sigma^2 = 100$

3.3 Linear Regression

Linear regression [10] is a method to fit a line or a curve to a set of data and is useful in cases where we want to create a generalized model of the data, for instance in the case of prediction. This can help answering questions on how a system behaves once it reaches beyond the scope of the existing set of data. The method produces a line or a curve that is optimally fit to the set of data in the sense that it minimizes the mean of the squared error.

If there are measurement data $\mathbf{x} = \{x^{(1)}, ..., x^{(m)}\}$ and $\mathbf{y} = \{y^{(1)}, ..., y^{(m)}\}$ they might be approximately related by $\mathbf{Ac} \approx \mathbf{y}$:

$$\begin{pmatrix} 1 & x_1^{(1)} \\ . & . \\ . & . \\ . & . \\ 1 & x_1^{(m)} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \approx \begin{pmatrix} y^{(1)} \\ . \\ . \\ . \\ y^{(m)} \end{pmatrix} \tag{3.2}$$

where the superscript (number) denotes the number of measurement data pairs while the subscript denotes the number of features (in this case y only depends on x). Solving for the coefficients $\mathbf{c}$ can then produce an equation for the line that approximates the behavior of the system. This can be done by setting up and solving the normal equations for the coefficients $\mathbf{c}$.

$$\mathbf{c} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y} \tag{3.3}$$

3.4 Kalman Filter

The localization problem can be compared to the problem of a hidden state where we cannot observe the state directly but must rather use measurements to relate to and infer the state. The *Kalman filter* (KF) [28] is a state estimation algorithm for linear systems. The purpose of the algorithm is to produce a more accurate estimate of an uncertain value by using a series of measurements sequentially. It gives the optimal estimate (minimizes the mean of the squared error) by weighing the two values together based on how confident we are in respective values. The confidence is stated by a noise value (high noise means low confidence) and this value must also be Gaussian, for the Kalman filter to provide an optimal estimate. The algorithm is recursive and for each discrete time-step there are two phases, the *time update* (for prediction) followed by the *measurement update* (for correction). The time update uses the *posterior* (the output from the measurement update) of the last time-step or an initial estimate, and predicts the behavior of it and its noise as they transition into the next time-step. The output of the time update can be regarded as a "first guess" and is also referred to as the *prior*, in the sense that it is the estimate before being "corrected" in the measurement update. The purpose of the measurement update is then to adjust the prior and yield an estimate closer to the "true value", by using a measurement and the confidence in that particular measurement. The corrected estimate, after a measurement has been taken into account, is referred as the *posterior* and is the final estimate of its time-step. This posterior can then be used as the input to the time update of the next time-step.

The equations for the *discrete Kalman filter* (the "standard" version for linear problems) are listed next, starting with the time update. The equations to predict the prior $\hat{x}_k^-$ and the prior noise P_k^- are the following:

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \tag{3.4}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{3.5}$$

The equations represent the generalized, multi-dimensional case where the various variables are represented by vectors and matrices. The superscript (-) indicates that it is a prior value, while the subscript indicates the time-step. A is the state transition matrix for the state, in case the state varies from one time-step to the next. B is the state transition matrix for the user control input u in case the process is user-controlled. Q is the Gaussian process noise introduced by the system itself and can be represented by a covariance matrix.

The equations for the Kalman gain K_k, the posterior $\hat{x}_k$ and the posterior noise P_k in the measurement update are as follows:

$$K_k = \frac{P_k^- H^T}{H P_k^- H^T + R} \tag{3.6}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{3.7}$$

$$P_k = (I - K_k H)P_k^- \tag{3.8}$$

Once again, the equations represent the generalized case, and the superscript (-) and the subscript denote prior values and time-step of the variable, respectively. K_k is the weight which decides in which direction (towards the prior or the measurement) the Kalman filter will "pull" the estimate. Also, while the prediction phase increases the noise P_k^-, the correction phase decreases P_k because the confidence is stronger in the estimate after Kalman-filtering it. H is the measurement matrix for mapping between the state and the measurement, in case the measurement is not an observation of the state itself. For example, if we were to estimate a temperature value by measuring weight. R is the measurement noise and is similar to Q, a covariance matrix in the generalized case. I is the identity matrix. It can be noted that A, H, Q and R are previously known and user-supplied matrices, and a simplified version of the Kalman filter is used for this project , which will be explained later.

The recursive nature of the algorithm makes it suitable for real-time processing as not all measurement data need to be ready at the time of calculation. There is also no need to store entire series of values of either posteriors or measurements, other than to keep track of the posterior state from the latest time-step, together with its associated noise.

3.5 Particle Filter

Even though the Kalman filter does produce the optimal estimate it also requires the strict assumptions of a linear and Gaussian system to hold. This is not always possible. An alternative state estimation algorithm with looser restrictions is the *particle filter* [1]. Similarly to the Kalman filter it uses measurements to infer the hidden state and operates recursively, which makes it suitable for real-time applications. It also has two phases similar to the prediction-correction cycle for each discrete time-step, but which work slightly differently. The particle filter represents the estimates in so called "particles". Each particle stores a guess of the state and an importance weight to signify the confidence in the guess based on measurements.

The *transition* phase is the equivalent to the time update and uses a *transition model* to dictate the possible movements of a particle. The transition phase is performed by moving each particle to a, for that specific particle, possible state, given the state of that particle at the previous time-step.

$$x_k^i \sim p(x_k \mid x_{k-1}^i) \tag{3.9}$$

The variable x_k^i denotes the state of the ith particle at time-step k.

Then how do we translate a measurement into an importance weight? The particle filter requires a *measurement model* which relates the measurement to a probability value (the *likelihood*), indicating how likely it is to make *that* particular observation (measurement) in a specific setting. More formally, the importance weight w_k^i is the probability of observing z_k given the state x_k^i:

$$w_k^i = p(z_k \mid x_k^i) \tag{3.10}$$

where the superscript denotes the particle index and the subscript denotes the time-step. The calculation of the likelihood probability is comparable to the measurement update of the Kalman filter.

The main reason for choosing a probabilistic approach is its ability to cope with the uncertainties of the environment and the sensors, which in turn leads to a more error-tolerant and robust implementation [26].

In localization, the measurement model does not differ much in principle from a real-world map. As an example, imagine we are traveling through Gothenburg, then we are more likely to see a road sign pointing to the amusement park *Liseberg* than one pointing to *Gröna Lund* (the Stockholm equivalent). The measurement model can be constructed in various ways, ranging from for example, histograms to discrete or continuous probability distributions. Once we have fetched the importance weight from the measurement model we can infer the most likely posterior by looking at the particles with the highest importance weights.

The algorithm works recursively by resampling the particles, with replacement, in proportion to the normalized importance weights. This means, assuming a static number of particles in the particle filter, that a higher percentage of particles will gather in the vicinity of the most probable posteriors. At the same time the particle population that are representing the less likely posteriors becomes sparser or disappears entirely. We can consider the resampling process as a filter, where posterior estimates survive or die out. There are several different variants of particle filters, and for this project we will use the *Sequential Importance Resampling* (SIR) particle filter, where the resampling process occurs at each discrete time-step. The particle filter algorithm in algorithm 1 describes the SIR procedure.

The resampling is done by constructing a *cumulative distribution function* (CDF) from the weights of the current particle set and traversing it taking equally spaced steps (the uniform probability $1/n$). This method is known as *systematic resampling*. The systematic resampling algorithm in algorithm 2 describes the procedure for resampling the particles in proportion to their weights.

The general steps of the SIR particle filter are ordered as follows [2, 9]:

1. Initialize particle set S_0 (uniformly or according to prior knowledge) at time-step $k = 0$
2. Receive new measurements and increment time-step $k \leftarrow k + 1$
3. Move each particle i from the result particle set of the previous time-step, S_{k-1}, according to the transition model $x_k^i \sim p(x_k \mid x_{k-1}^i)$
4. Calculate the importance weight $w_k^i = p(z_k \mid x_k^i)$ for each particle i
5. Normalize the importance weights so that they sum up to 1
6. Use the importance weights for estimation
7. Resample the particles based on importance weights according to algorithm 2 and keep the resampled particle set S_k for the next time-step
8. Go to step 2

Algorithm 1 SIR Particle filter

```
function SIR_PARTICLEFILTER($S_{k-1}$, $z_k$)
    $S_k = \emptyset$, t = 0
    for i = 1,...,n do
        Move particle according to transition model $x_k^i \sim p(x_k \mid x_{k-1}^i)$
        Calculate importance weight $w_k^i = p(z_k \mid x_k^i)$
        $t = t + w_k^i$
        Add particle$(x_k^i, w_k^i)$ to $S_k$
    end for
    for i = 1,...,n do
        Normalize weights: $w_k^i = \frac{w_k^i}{t}$
    end for
    Compare importance weights and estimate posterior
    Resample according to weights in particle set $S_k$
    return $S_k$
end function
```

Algorithm 2 Systematic resampling

```
function SYSTEMATIC_RESAMPLING($S_{k-1}$)
        $S_k = \emptyset, c_1 = w^1_{k-1}$
    for $i = 2, ..., n$ do
        Construct CDF: $c_i = c_{i-1} + w^i_{k-1}$
    end for
        Initialize threshold uniformly between 0 and 1/n: $u_1 \sim U[0, 1/n]$
        Start at the bottom of the CDF: $i = 1$
    for $j = 1, ..., n$ do
        while $u_j > c_i$ do
        $i = i + 1$
        end while
        Add particle($x^i_{k-1}$,$1/n$) to $S_k$ with a uniform weight
        Increment threshold $u_{j+1} = u_j + 1/n$
    end for
        return $S_k$
end function
```

Chapter 4

Implementation

This chapter describes how the previously explained theoretical concepts are applied in the implementation.

4.1 Equipment

The prototype was developed to run on the Android device *HTC Magic (Vodafone edition)*, with access to the accelerometer, the orientation sensor (a magnetic compass) and the Wifi sensor.

4.2 Dead Reckoning

The dead reckoning component, as mentioned earlier, uses the sensor data from the accelerometer and the compass of the Android device to estimate the location. The sensors for the dead reckoning implementation are both sampled at approximately 25 Hz. To smoothen out the possible noise and misreadings by the sensors such as spikes, a moving average is used to represent the latest sensor reading. This smoothing procedure is performed by the implementation on both sensors that are relevant for dead reckoning. The latest four accelerometer readings are averaged while the latest five are averaged for the compass. The different values of the latest readings to average were chosen as a result of perceived difference in responsiveness from the averaged values.

The dead reckoning implementation has also been developed with a few usage presumptions and restrictions in mind. It is presumed that the user carries the Android device with its positive y-axis in the heading direction and its positive z-axis upwards during the dead reckoning localization. Figure 4.1 illustrates expected user behavior in 2D. With that said, how are the sensors used for dead reckoning?

The two main components of dead reckoning are *direction* of travel (*bearing*) and the *distance* traveled in that direction (*displacement*). To determine the bearing, we simply use the compass reading. The value given by the sensor spans 0°-359° where 0° is presumed to point to the magnetic north. Since we only use the compass

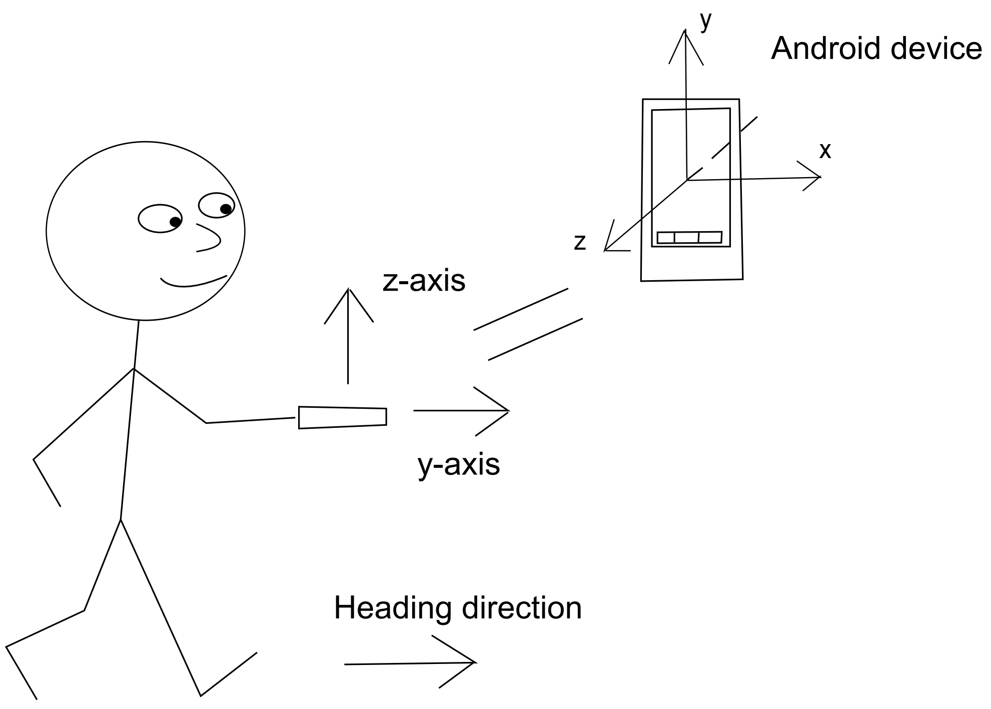

Figure 4.1. *Expected user behavior.*

to determine the bearing we need to consider the *magnetic declination* [8] that is different for different parts of the world. This is because the magnetic north is not perfectly aligned with the true north, which is the north used in for example, maps. A small adjustment of $+5°$ was made to the orientation reading according to [7].

The other dead reckoning component is the distance displacement. For this, a pedestrian perspective is assumed as detecting and counting steps is a common and relatively robust approach to estimating the position displacement of a person. First, a step needs to be detected, before a displacement can be added. The implementation utilizes the readings from the z-axis of the accelerometer to detect steps, and excludes the magnitude of gravity using the Android API. A static value of $1.1\ \mathrm{m/s}^2$ was empirically chosen as the threshold to detect steps. In other words, all smoothened accelerometer values that exceed the threshold are *potential* step occurrences. It is made sure that the potential step occurrence has been preceded by negative z-axis readings since the last valid step occurrence, before being regarded as a valid step. This is to prevent one valid step from being falsely interpreted as multiple steps, by utilizing the fact that regular walking follows a periodic pattern and that the accelerometer readings shift accordingly. Figure 4.2 illustrates the step detection.

Other than detecting steps, we must also know the step length to accurately estimate distance displacement from walking. It is safe to assume that the step

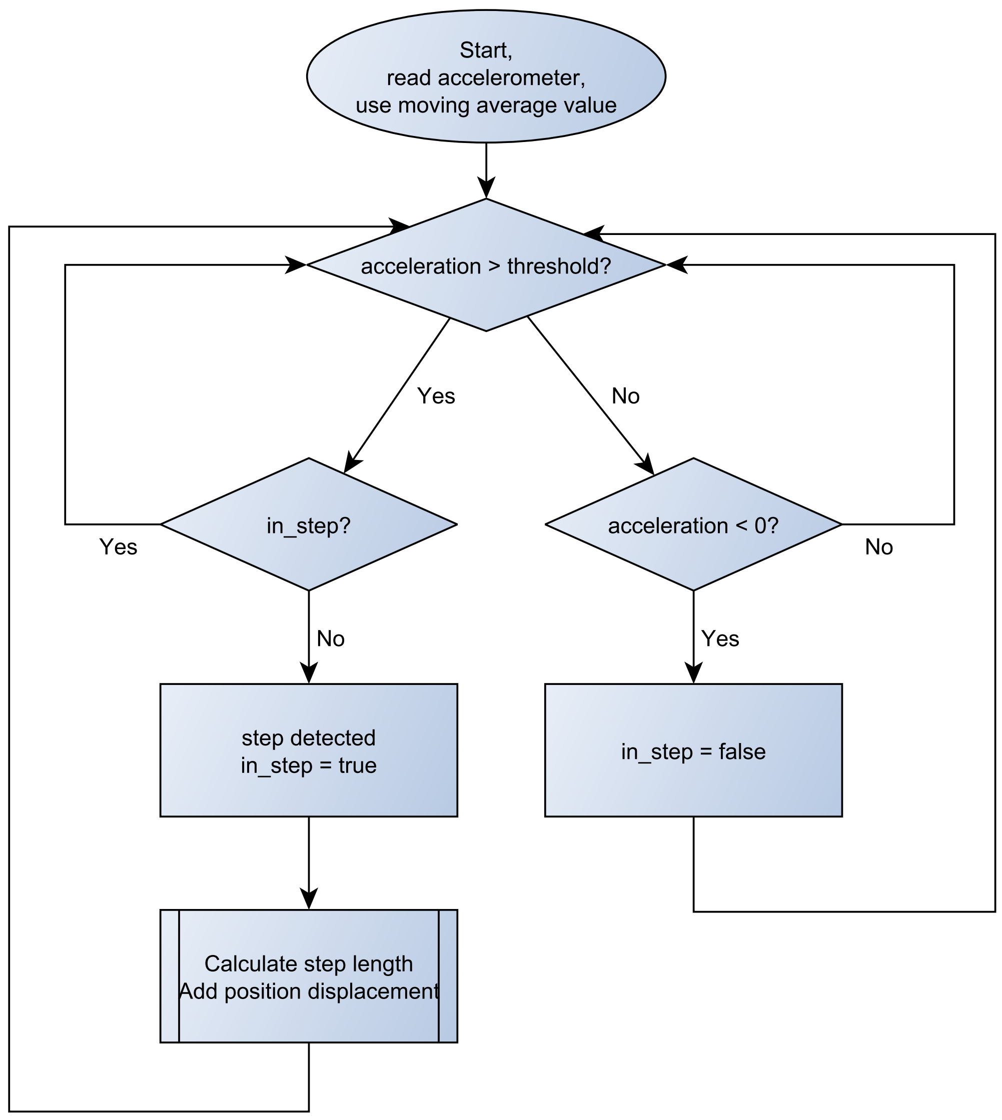

Figure 4.2. *Step detection procedure.*

length is dynamic among pedestrians. Because of the need to find a dynamic step length model, and out of curiosity, the author chose to examine the relation between the step period and the step length to see if there was any connection that could be utilized. This was done through linear regression. Initially, data had to be collected in order to perform linear regression. More specifically, we needed to collect data that related the step period (s/step) to the step length (m/step). Note that this approach tailors the dead reckoning implementation for the author, since step

Step period (s/step)	Step length (m/step)
0.6	0.76
0.61	0.75
0.72	0.6
0.72	0.6
0.55	0.9
0.57	0.88
0.55	0.85
0.59	0.84
0.57	0.86
0.57	0.88
0.63	0.73
0.65	0.72
0.61	0.75
0.62	0.75
0.72	0.62
0.77	0.6
0.71	0.63
0.77	0.6

Table 4.1. *Step period and step length data*

periods and step lengths can differ between different persons. The author walked a distance of approximately 36 m 18 times in total, in three different walking paces and six times for each: slow, normal and fast walking. This is to gain a spread in the data. During each walk, the number of steps were counted and the elapsed walking time was measured. From these measurements, we can calculate the step period and the step length for each of the 18 walks. See table 4.1.

Solving $\mathbf{c} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}$ gives $c_1 = 1.6481$ and $c_2 = -1.4177$, which means we can calculate the step length y given step period x:

$$y = 1.6481 - 1.4177x \tag{4.1}$$

During localization the step period is simply the time difference between two valid steps, namely measured from the last valid step until a new valid step. There are two special cases to consider. The first is where the time difference is very high, meaning the user is walking very slowly or it could indicate a first step in a series if the user had previously stopped moving. A default minimum step length is given to represent these cases. The second is where the time difference is very low, meaning the user is stepping forward very quickly and might even be running. Since the training did not include several different paces of running, running has not been properly modeled nor tested, and therefore the implementation is limited to walking and does not account for running. A default maximum step length is used for this case. In other words the step length model is only used within a window specified by an upper and a lower limit of the step period. Figure 4.3 shows how the line fits the training data and the independent test data.

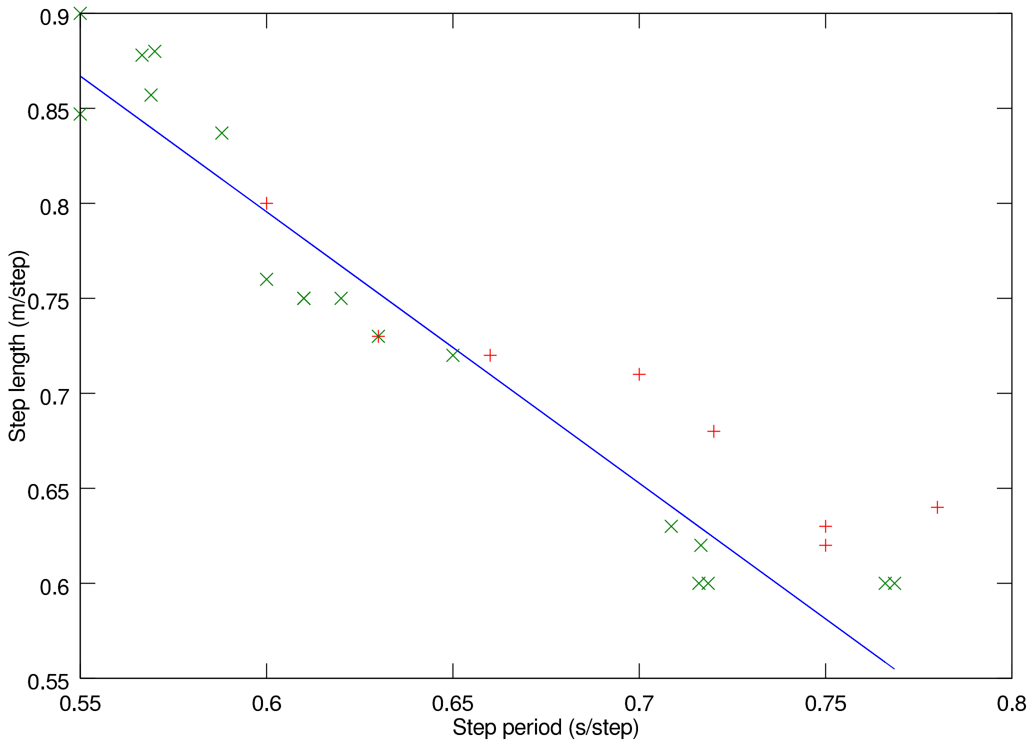

Figure 4.3. *The line which represents the dynamic step length and its fit to the training data ('x') and the independent test data ('+')*

After calculating the step length we need to translate the displacement into a movement in terms of global coordinates, latitude and longitude. The *haversine formula* [27] is method to calculate the shortest distance between two points via the surface of a sphere. We use the equations derived from the haversine formula for this translation [27]. The new global coordinates (lat_2,lon_2) after moving distance d with the bearing θ from the original position (lat_1,lon_1) are given by:

$$lat_2 = \arcsin(\sin(lat_1)\cos(\frac{d}{R}) + \cos(lat_1)\sin(\frac{d}{R})\cos(\theta)) \tag{4.2}$$

$$lon_2 = lon_1 + \arctan 2(\sin(\theta)\sin(\frac{d}{R})\cos(lat_1), \cos(\frac{d}{R}) - \sin(lat_1)\sin(lat_2)) \tag{4.3}$$

where R is the earth radius (set to 6367.5 km in the implementation). This approach approximates the earth to a sphere, which might generate distance errors up to 0.55 % [27]. Also note that all angles converted to radians for these calculations. Figure 4.4 illustrates the dead reckoning procedure.

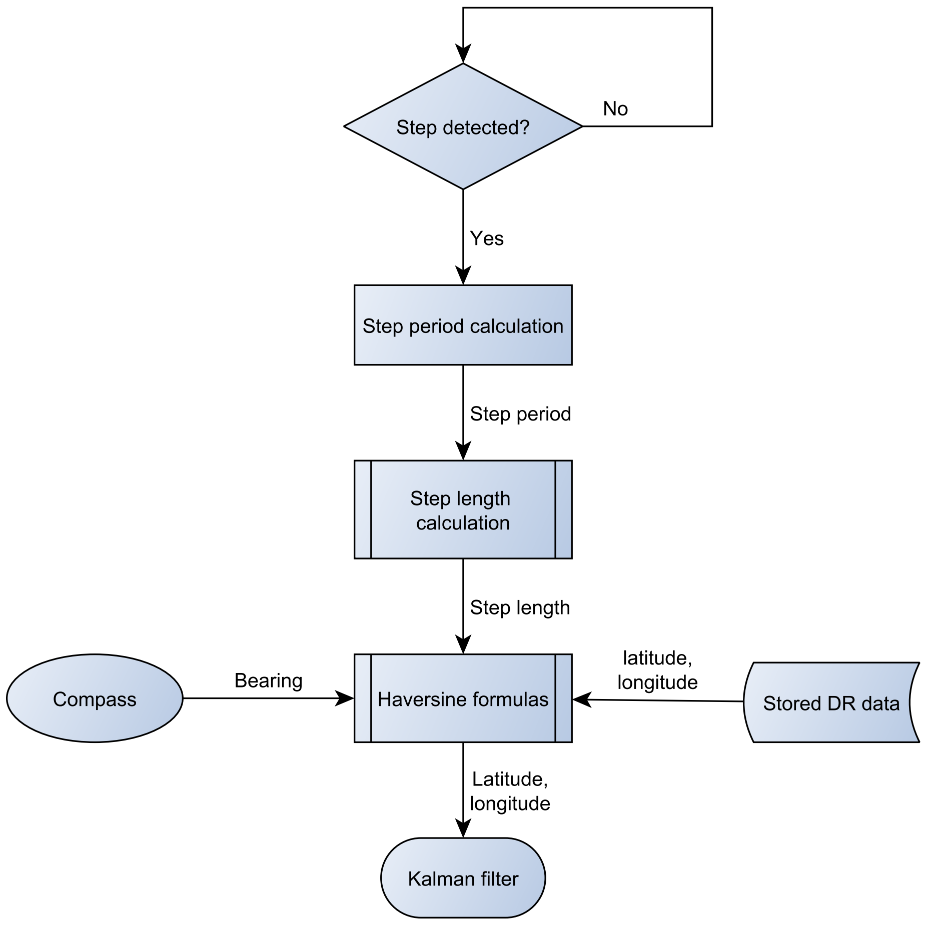

Figure 4.4. *Dead reckoning procedure.*

4.3 WLAN Localization

To localize with the help of WLAN, we will use the SIR particle filter, which was introduced earlier. The key measurements in this implementation are the WLAN signal strength values (also referred to as *RSSI* in some literature), which will build the foundation of the localizer. As described earlier, the particle filter requires a measurement model for correctly assessing the RSSI. Therefore we divide the WLAN localization into two phases: the *offline* phase where we construct the measurement model, and the *online* phase where we use it and actual localization takes place.

To build the measurement model (remember the geographic map comparison), a total of 21 reference points (RPs) were placed throughout the office floor in a 3×7 grid with the spacing of 3.5 m horizontally and vertically. This means that the WLAN localizer cannot produce an estimate with higher accuracy than 3.5 m. Each reference point was associated with a known global coordinate and the reference points covered an area of 147 m^2 (7 m $\times$ 21 m) in total. Figure 4.5 shows the placements of all the 21 reference points on the office floor.

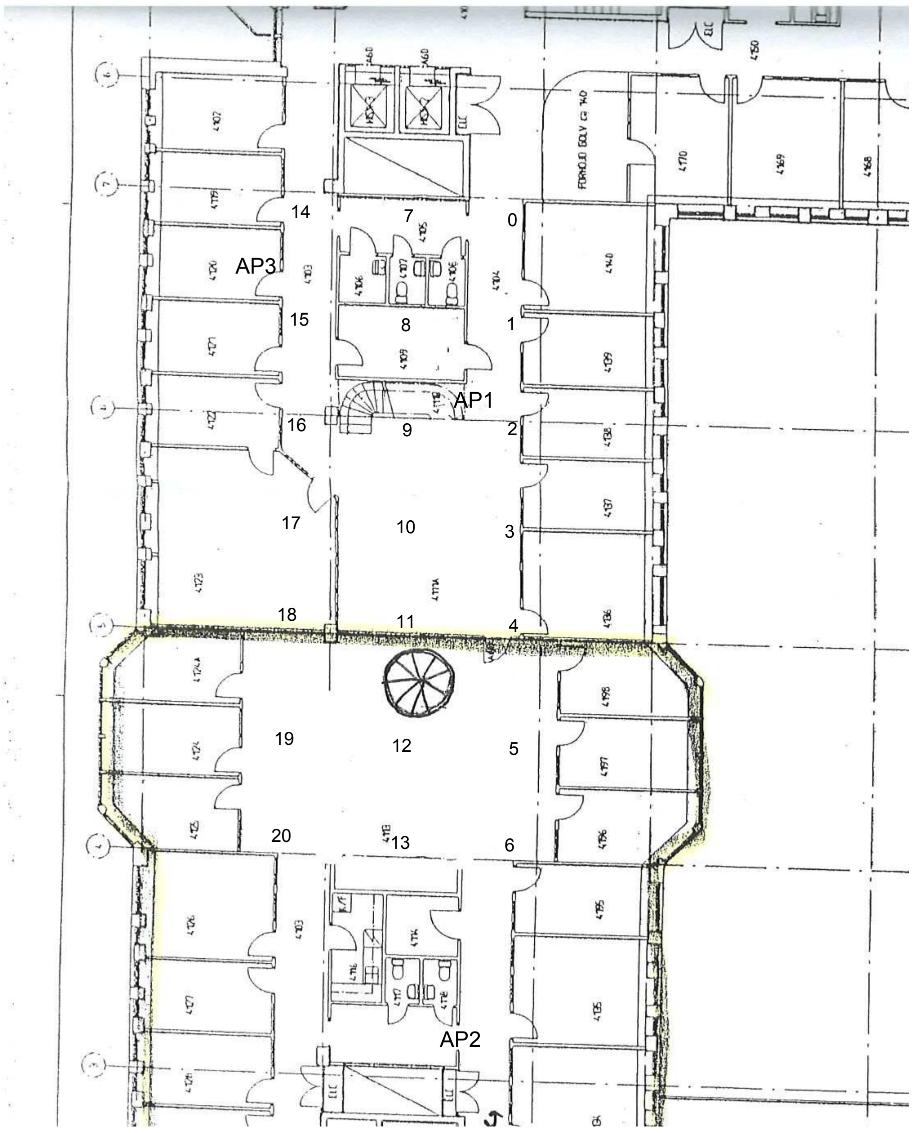

Figure 4.5. *Map of the reference points and WLAN access points. The reference points are numbered 0-20, and the WLAN access points are marked as AP1-AP3.*

RSSI received from three separate WLAN access points (APs) are used for each reference point in the localization, and we make the assumption that these values are Gaussian as [20] suggests. Therefore the likelihood distribution for each reference point can be modeled by the normal distribution. Since there are 21 reference points, with each reference point receiving RSSI from three access points, we have $21 \cdot 3 = 63$ different likelihood distributions so far. Measurements from one access point can be sufficient for localization, but utilizing more than one access point can help distinguish between two locations that otherwise would give the same observations. This reduces the risk of having "mirrored" situations and increases the accuracy. See figure 4.5 for the placements of the three access points. One of the access points (AP2) was located one floor below the rest of the access points and reference points.

To address the impact of directionality mentioned in [21], one likelihood distribution was created for each of the two opposite directions, instead of combining measurements from two different directions into one likelihood distribution. These opposite likelihood distributions for each reference point will be referred to as "Down" ($145° \leq$ bearing $< 325°$) or "Up" (bearing $< 145°$ or bearing $\geq 325°$). This means we have a total of $21 \cdot 3 \cdot 2 = 126$ different likelihood distributions in the measurement model. During the online phase, the compass is used to determine when to use which likelihood distribution.

To construct the likelihood distributions, for each likelihood 20 RSSI samples were measured to represent the probability distribution and should be sufficient according to [21]. From these samples the necessary parameters for the normal distributions μ and σ could be calculated. The arithmetic mean of the data was used as the expected value μ:

$$\frac{x_1 + ... + x_n}{n} \tag{4.4}$$

The variance σ^2 of the data is calculated as follows [10]:

$$\sigma^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \mu)^2 \tag{4.5}$$

The standard deviation σ is the square root of the variance: $\sigma = \sqrt{\sigma^2}$.

Occasionally during the offline phase, we receive no-signal RSSI values that must also be taken into consideration when constructing the measurement model. This is treated by setting all incoming no-signal measurements to a default signal value that is too low to be sensed by the WLAN signal scanner. We also manually increase the variance of the measurement series which are dominated by no-signal values. This is to compensate for the low variance that many default "filler" values give, when in actuality the measurements in the set are highly uncertain.

Besides the measurement model, a transition model is necessary to spread the particles in the transition phase (which can compared to the prediction step of the Kalman filter). For this, an adjacency list of all the reference points was created to represent the possible routes of a particle (or the user). For example, walls are considered as blocking and prevents a particle from moving through it even if the reference points are neighbors. The transition means that a particle transitions to a new guess of the location. Figure 4.6 illustrates the transition procedure, which takes place after the resampling.

In this implementation and for the testing, we use 21 particles, as many as there are reference points. These are initialized locally and spread uniformly among the five reference points that are closest to the initialization coordinate, which is inserted manually or retrieved from GPS. The distance d between the points (lat_1,lon_1) and (lat_2,lon_2) is calculated with the haversine formula in the following procedure [27]:

$$a = \sin^2\left(\frac{lat_2 - lat_1}{2}\right) + \cos(lat_1)\cos(lat_2)\sin^2\left(\frac{lon_2 - lon_1}{2}\right) \tag{4.6}$$

$$c = 2 \cdot \arctan 2(\sqrt{a}, \sqrt{1-a}) \tag{4.7}$$

$$d = R \cdot c \tag{4.8}$$

where R is the earth radius. Again, the angles are converted to radians for the calculations. Figure 4.7 illustrates the initialization of particles.

We also make the assumption that the RSSI values from the three different access points are independent of each other, which allows us to just multiply the probability values together when combining them into an importance weight [20]. This facilitates calculations during the online phase.

Generally, using more particles gives a more accurate posterior estimate, but since this implementation is limited to only 21 discrete and unique states (the reference points), the need for particles is not as high as for continuous state spaces. During the online phase, the particles move across the discrete reference points to calculate the likelihood of observing the received RSSI values in their respective locations. The probability density function of the normal distribution are used for all likelihood calculations and the three probabilities are then multiplied together into an importance weight. Figure 4.8 illustrates the computation of importance weights. After normalization, to make the sum of all importance weights add up to 1, the importance weight is then used for estimating the posterior and for the resampling step, according to the SIR particle filter described in section 3.5. The implementation of the WLAN particle filter is illustrated in figure 4.9.

4.4 Combining Dead Reckoning, WLAN and GPS

The Kalman filter is used to combine the results from the dead reckoning implementation and the WLAN localizer into a posterior estimate, and to track the location as time passes. The distance displacements calculated from dead reckoning are used as prediction in the time update of the Kalman filter, while the WLAN localizer estimates are used as the correcting measurements in the measurement update. We assume that the results from the dead reckoning implementation and the WLAN localizer are both Gaussian. The Kalman filter runs as soon as it has received a result from the particle filter, which is approximately once every two seconds, since that is the interval of the WLAN signal scans. The prior and the prior noise of the Kalman filter is initialized by the last known GPS location, manually or preset in the implementation, depending on test scenario. There is not necessarily one time update for each measurement update, but more likely several time update phases for each measurement update if the user is moving, or vice versa if the user is stationary.

The Kalman filter equations described in section 3.4 are used in the one-dimensional case for latitude and longitude estimation, respectively. We use the haversine equations described in section 4.2 as the prediction phase and not the traditional time update equations. Therefore we can disregard the state transition matrix A and the control input u with its associated matrix B. Also, we do not need to translate our measurements between different state spaces, therefore $H = 1$ in the measurement update equations. Our simplified measurement update phase looks as follows:

$$K_k = \frac{P_k^-}{P_k^- + R} \tag{4.9}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - \hat{x}_k^-) \tag{4.10}$$

$$P_k = (1 - K_k)P_k^- \tag{4.11}$$

where the equations 4.9, 4.10 and 4.11 correspond to the equations 3.6, 3.7 and 3.8 respectively. The measurement update is applied to both the latitude and longitude estimates after each run of the WLAN particle filter.

It is important to note that the uncertainty of the estimated position increases for each position displacement, and therefore noise needs to be added. This takes place in the prediction phase of the Kalman filter. Usually, in regular estimation cases zero-mean Gaussian noise from the process noise Q is added to the prior $\hat{x}_k^-$ and the equivalent to the measurement drawn from the measurement noise R, but in this project the author believes it is more interesting to review the results without the addition of random elements to the implementation. Therefore only the prior noise P_k^- is increased when steps are detected. This increase can be a static value or a percentage of the step length for that particular step. The latter is chosen for this implementation and the increase was empirically set to 20 % during development. The flowchart in figure 4.10 displays the overall structure and data flows of the implementation.

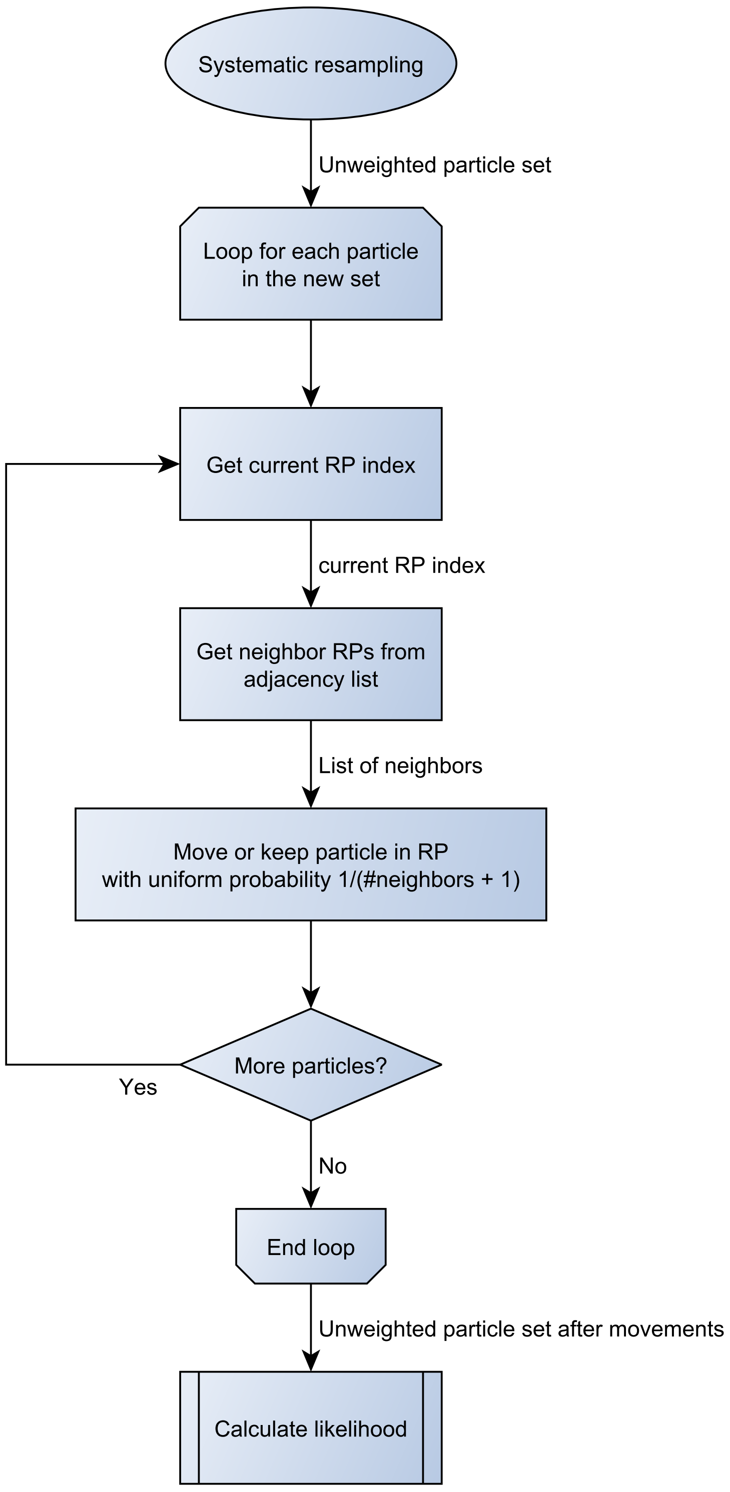
Systematic resampling
Unweighted particle set
Loop for each particle in the new set
Get current RP index
current RP index
Get neighbor RPs from adjacency list
List of neighbors
Move or keep particle in RP with uniform probability 1/(#neighbors + 1)
More particles?
Yes
No
End loop
Unweighted particle set after movements
Calculate likelihood

Figure 4.6. *Moving recently resampled particles in the WLAN particle filter transition phase.*

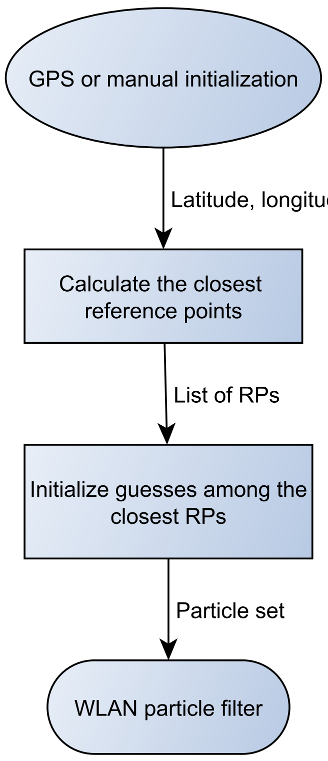
GPS or manual initialization
Latitude, longitude
Calculate the closest reference points
List of RPs
Initialize guesses among the closest RPs
Particle set
WLAN particle filter

Figure 4.7. *Initiating particles from initial location.*

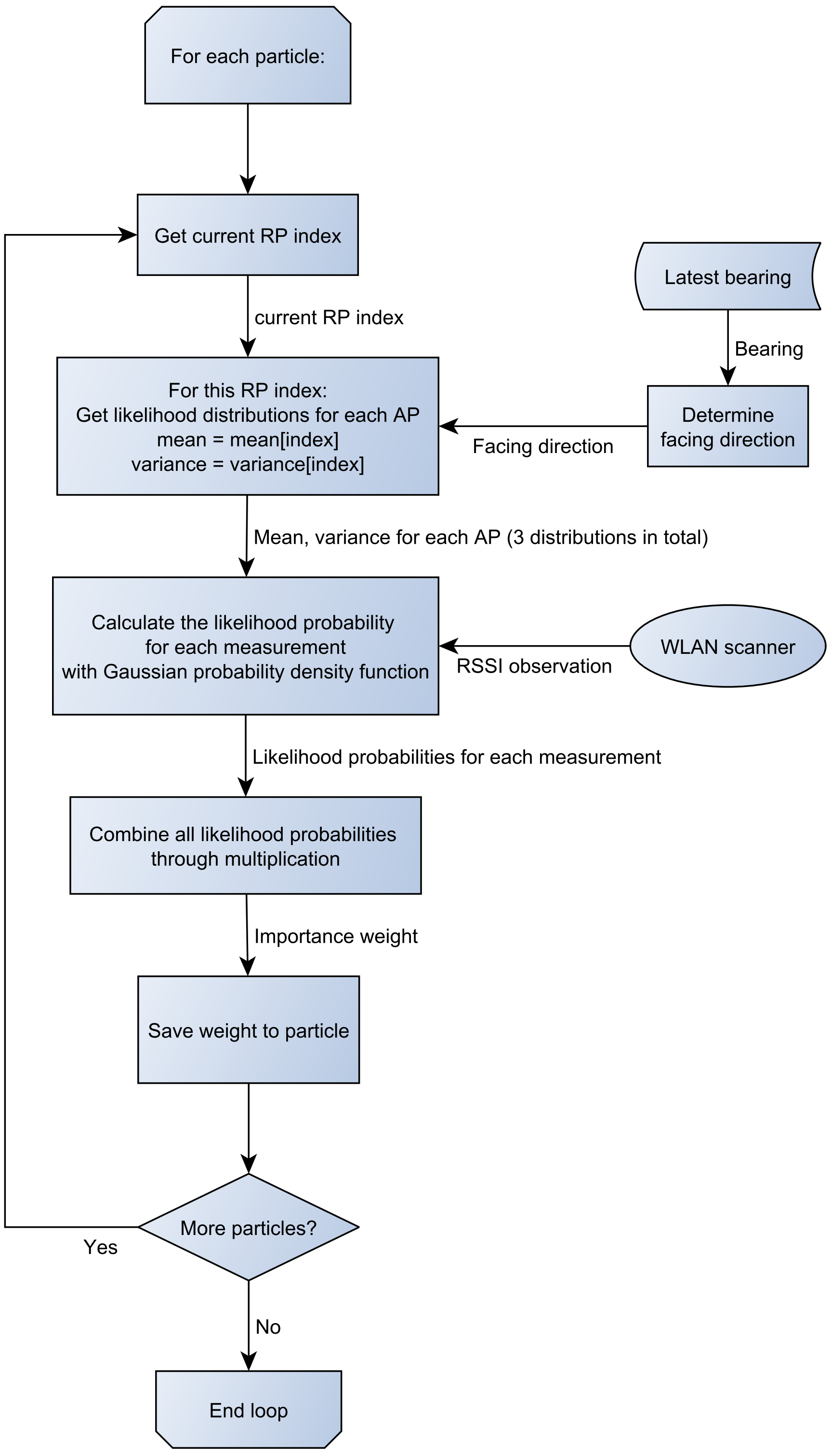
For each particle:
Get current RP index
current RP index
Latest bearing
Bearing
For this RP index:
Get likelihood distributions for each AP
mean = mean[index]
variance = variance[index]
Facing direction
Determine facing direction
Mean, variance for each AP (3 distributions in total)
Calculate the likelihood probability for each measurement with Gaussian probability density function
RSSI observation
WLAN scanner
Likelihood probabilities for each measurement
Combine all likelihood probabilities through multiplication
Importance weight
Save weight to particle
More particles?
Yes
No
End loop

Figure 4.8. *Likelihood calculation in the WLAN particle filter.*

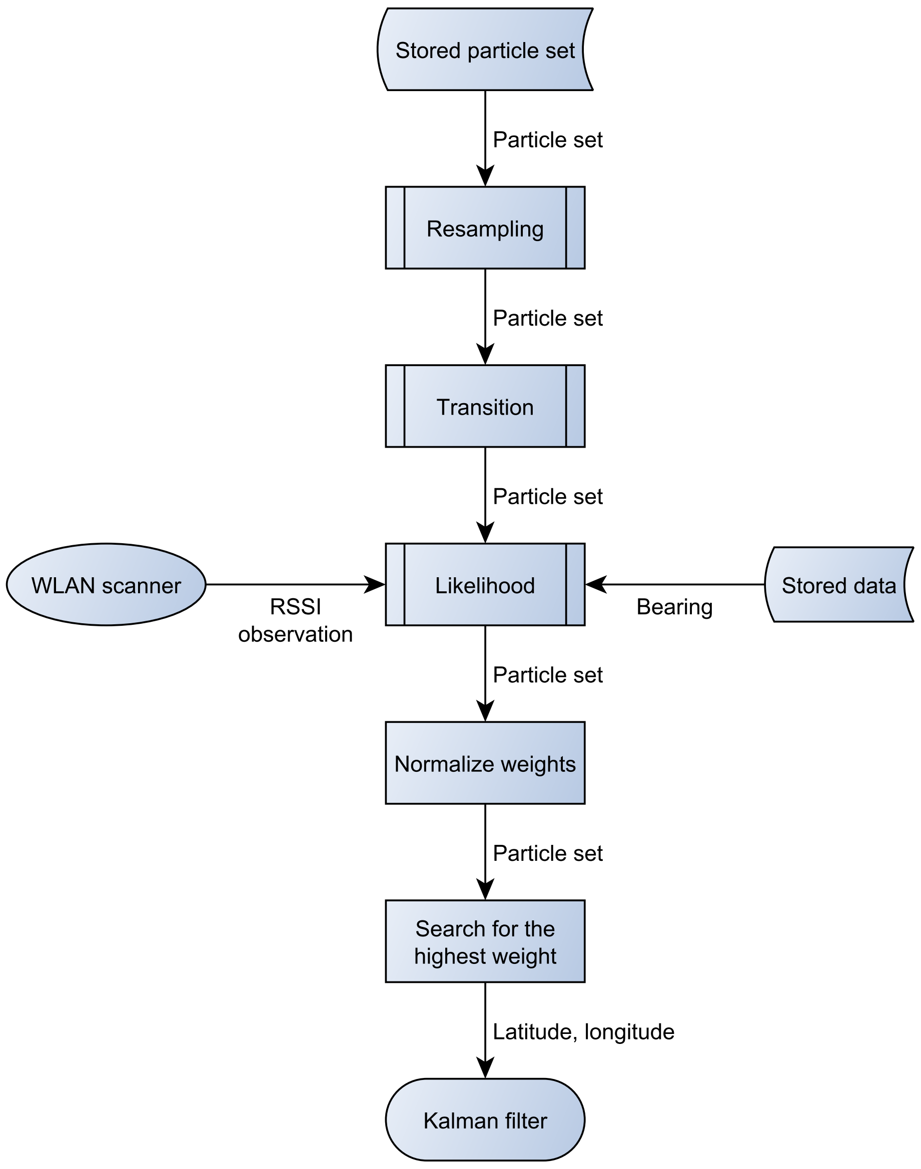

Figure 4.9. *The WLAN particle filter.*

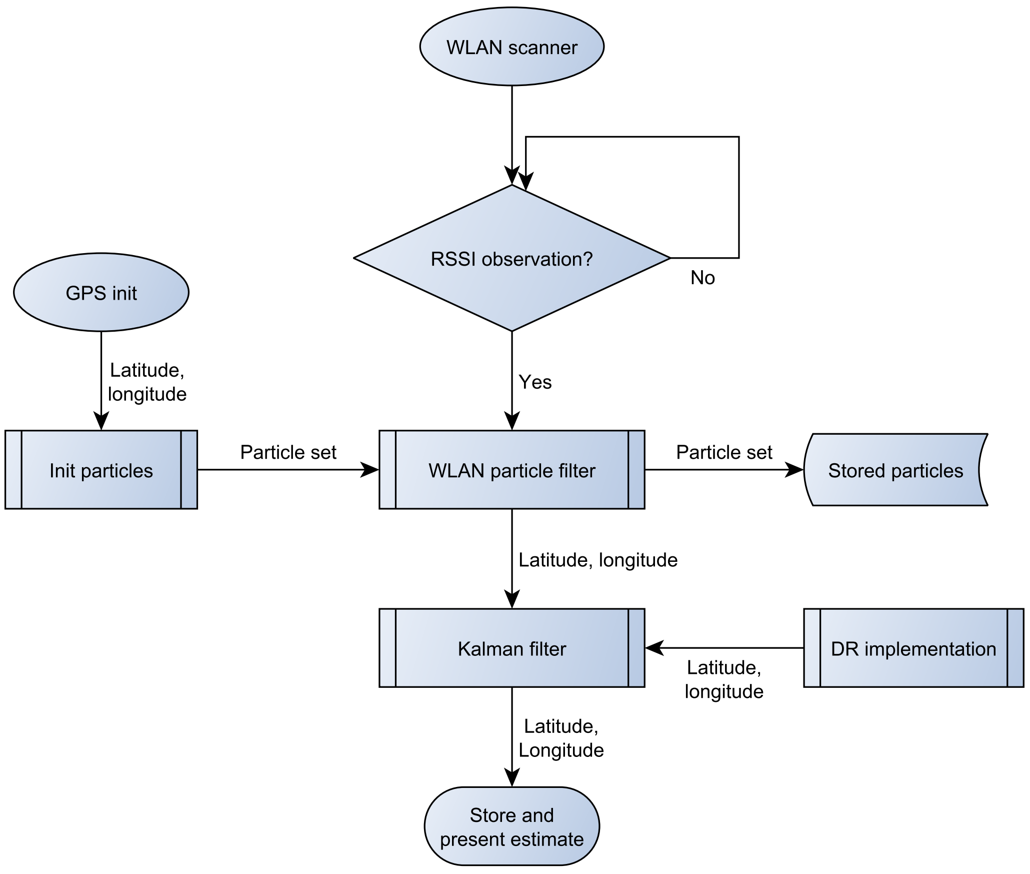

Figure 4.10. *Flowchart displaying all the components.*

Chapter 5

Testing

5.1 Testing Conditions

Testing was performed using the Android device *HTC Magic (Vodafone edition)*.

The tests were performed on the office floor that the constructed measurement model represents, and all three access points were used during testing. There might have been both static and moving objects in the environment not present at the time of measurement model construction, which could possibly have disrupted the WLAN signals from any of the three access points and in that way have affected the measurements, and in turn the localization results as well. The same applies to objects present at the time of the measurement model construction, but which were missing at the time of testing. In other words, it was not a static environment.

The implementation does not currently account for sloped terrain, but the tests were performed on a non-sloped floor. Therefore we assume no errors are introduced from sloped terrain.

During localization testing, the Android device is held and kept statically with its positive y-axis facing forward and positive z-axis facing upwards to the best of the tester's abilities, according to the usage presumptions stated earlier to minimize the error introduced from incorrect use.

5.2 Results

5.2.1 Dead Reckoning and Kalman Filter Performance on a Cyclic Course

Tests were performed to compare a dead reckoning-only solution to the a Kalman-filtered solution with WLAN input (KF). The tests were performed on a cyclic course with known start and end points. Initial location was manually set. Deviations from the correct position were measured after 35, 70, 140 and 280 m. The deviation test data for various initial prior noise P_k^- and measurement noise values R are found in tables 5.1, 5.2, 5.3 and 5.4. Table values are average deviation values.

Distance (m)	DR deviation (m)	KF deviation (m)
35	13.16789572	4.2987687
70	16.75127595	7.77131082
140	18.2856323	9.2380716
280	26.55886989	9.41494586

Table 5.1. *Initial prior noise P_k^- = 20 m, measurement noise R = 3.5 m*

The corresponding plots are illustrated in figures 5.1, 5.2, 5.3 and 5.4.

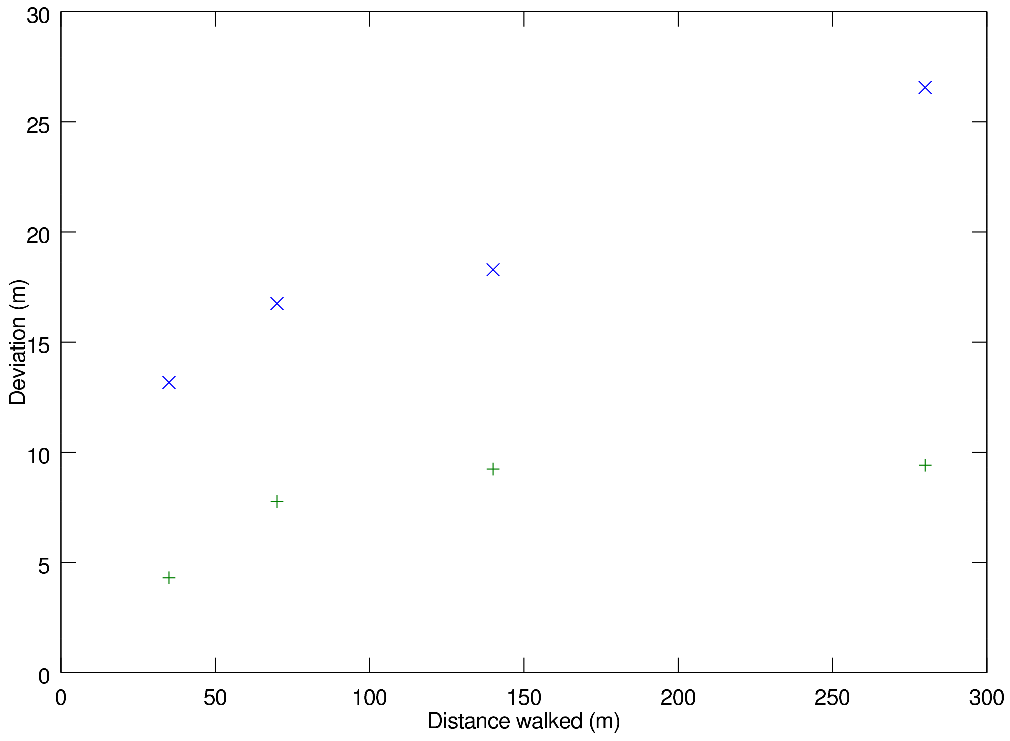

Figure 5.1. *DR deviation ('x') and KF deviation ('+') with initial prior noise P_k^- = 20 m, measurement noise R = 3.5 m*

5.2.2 WLAN Convergence

WLAN convergence was tested by standing still at a known location (a reference point), observe the deviations after 10 and 20 Kalman filter iterations. Elapsed time was 25-26 s and 50-52 s respectively for 10 and 20 iterations. Initial location estimate was given by GPS (at 0 iterations) and all tests were performed at the same reference point (RP 17). The reason for this was the close access to a window

Distance (m)	DR deviation (m)	KF deviation (m)
35	13.16789572	5.16492262
70	16.75127595	9.56725292
140	18.2856323	9.702293
280	26.55886989	8.13004692

Table 5.2. *Initial prior noise P_k^- = 40 m, measurement noise R = 3.5 m*

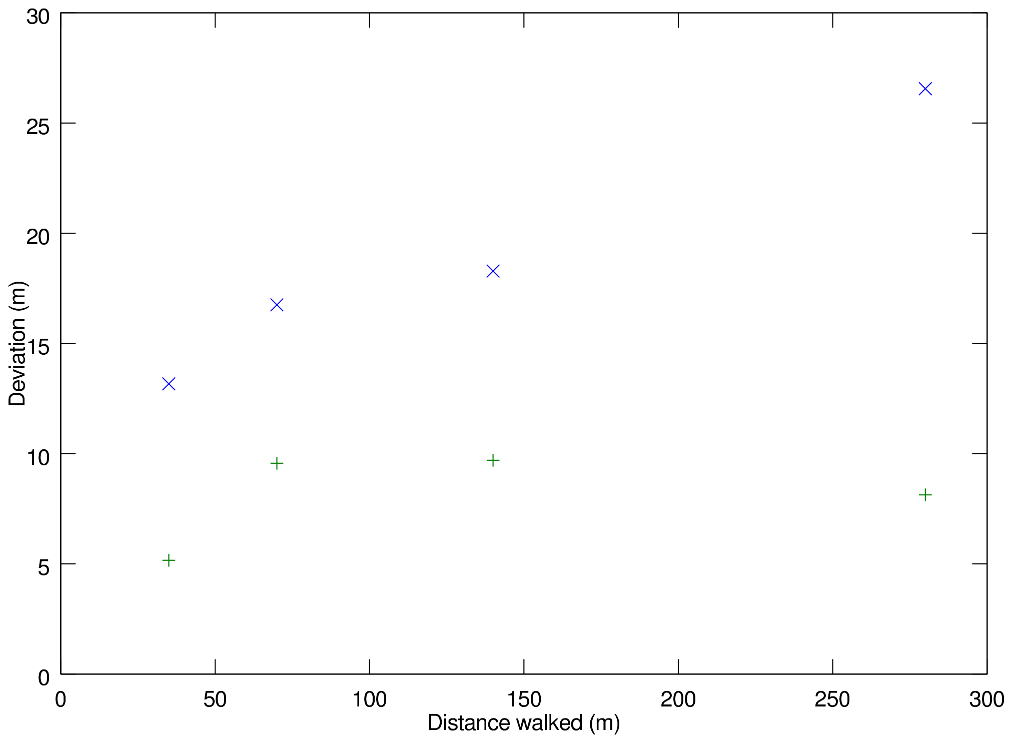

Figure 5.2. *DR deviation ('x') and KF deviation ('+') with initial noise P_k^- = 40 m, measurement noise R = 3.5 m*

and the quick detection of a GPS signal. The tests were performed with varying measurement noises and facing directions. Deviation data for measurement noise R = 3.5 m are shown in table 5.5 and figure 5.5. For $R = 7.0$, the test data are shown in table 5.6 and figure 5.6. Tables show the average values from four samples.

5.2.3 Randomly Walking

Finally, a test of a randomly walking scenario was performed to test the system under "realistic" conditions. The start location was initialized by a GPS signal and walking speed as well as the number of starts and stops varied, and the total

Distance (m)	DR deviation (m)	KF deviation (m)
35	13.16789572	5.69040238
70	16.75127595	9.1037334
140	18.2856323	8.5707426
280	26.55886989	7.39010336

Table 5.3. *Initial prior noise P_k^- = 20 m, measurement noise R = 7.0 m*

Distance (m)	DR deviation (m)	KF deviation (m)
35	13.16789572	4.99291904
70	16.75127595	7.16024096
140	18.2856323	9.20066766
280	26.55886989	7.1496143

Table 5.4. *Initial prior noise P_k^- = 40 m, measurement noise R = 7.0 m*

Kalman filter iterations	Deviation up (m)	Deviation down (m)
0	92.4250825	36.60327175
10	6.6991302	10.18915213
20	7.4846652	10.357684

Table 5.5. *WLAN convergence test data for measurement noise = 3.5 m*

Kalman filter iterations	Deviation up (m)	Deviation down (m)
0	35.0394975	41.2108385
10	7.504577	6.1907002
20	7.43186325	6.0717485

Table 5.6. *WLAN convergence test data for measurement noise = 7.0 m*

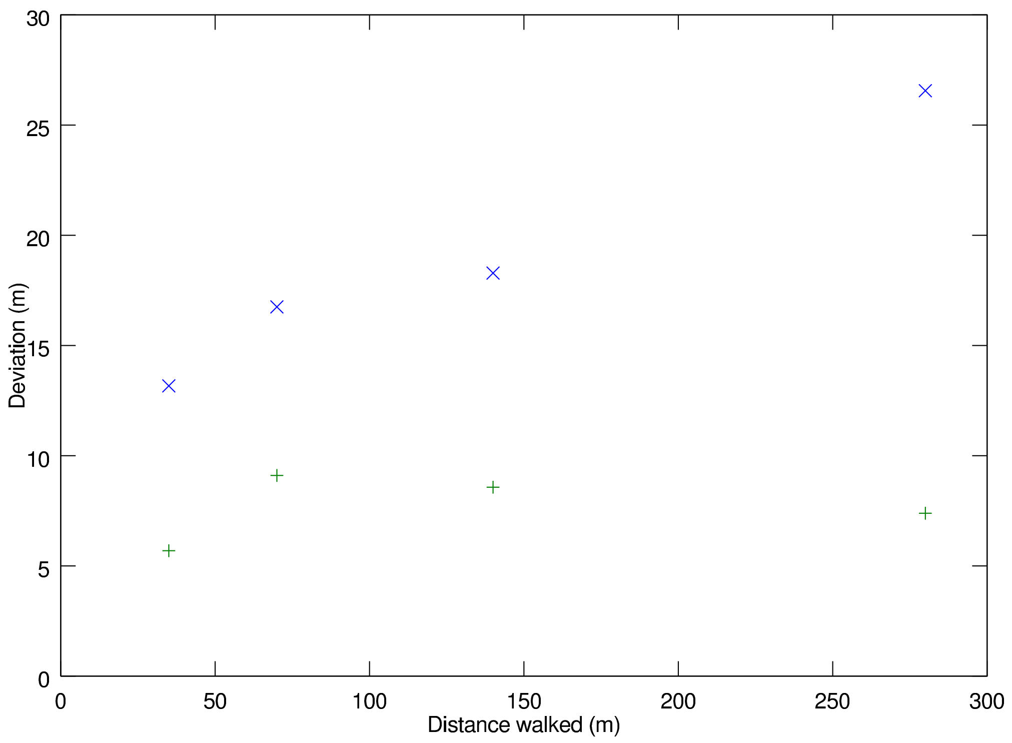

Figure 5.3. *DR deviation ('x') and KF deviation ('+') with initial prior noise $P_k^- = 20$ m, measurement noise $R = 7.0$ m*

distance traversed is unknown. No particular course was taken into consideration when walking, and moving and staying stationary did not follow a regular pattern. The test was to measure the ability of the implementation to converge to the nearest reference point or one of its closest neighbors within 50 iterations of the Kalman filter, and to successfully keep the tracking as close for at least 20 consecutive iterations. The success rate was 60 % (9 of 15).

At the occasion of failure, it did seem like the estimate was "stuck" in the corner of RP 20 and 19.

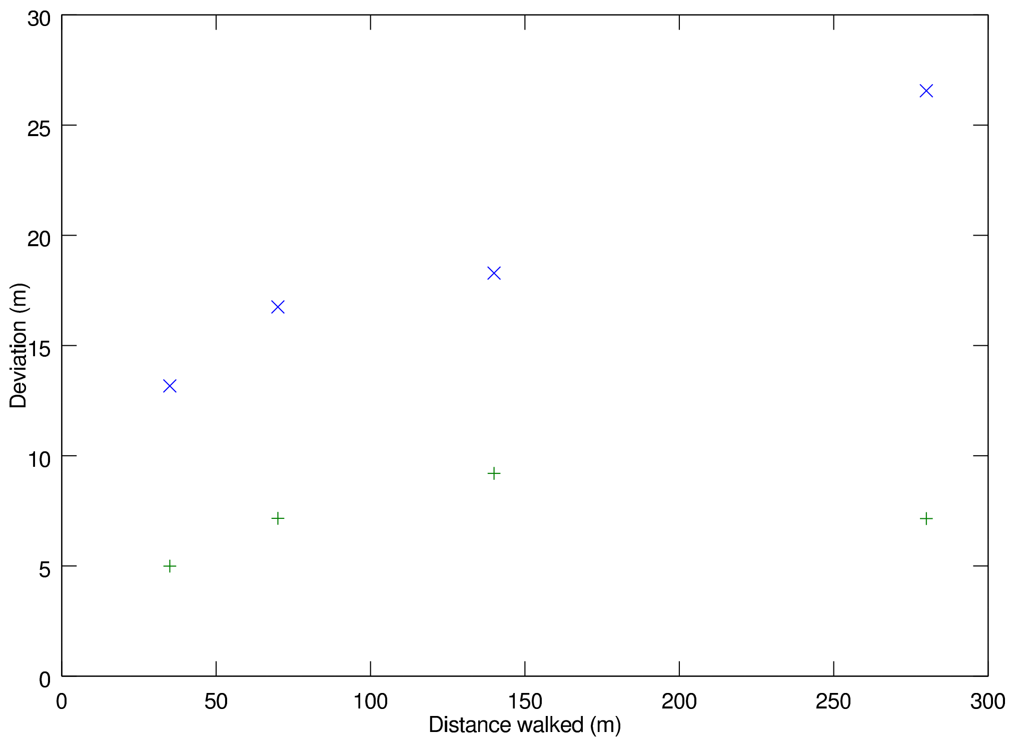

Figure 5.4. *DR deviation ('x') and KF deviation ('+') with initial prior noise* $P_k^- = 40$ *m, measurement noise* $R = 7.0$ *m*

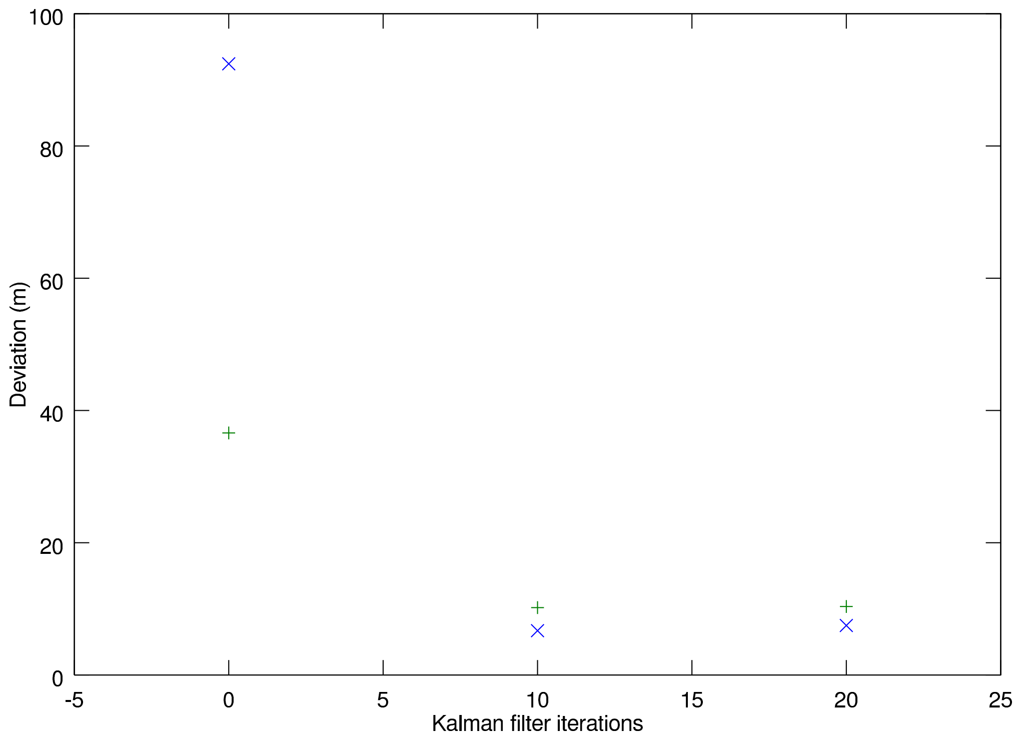

Figure 5.5. *WLAN convergence test (measurement noise 3.5 m): Direction up ('x'), direction down ('+')*

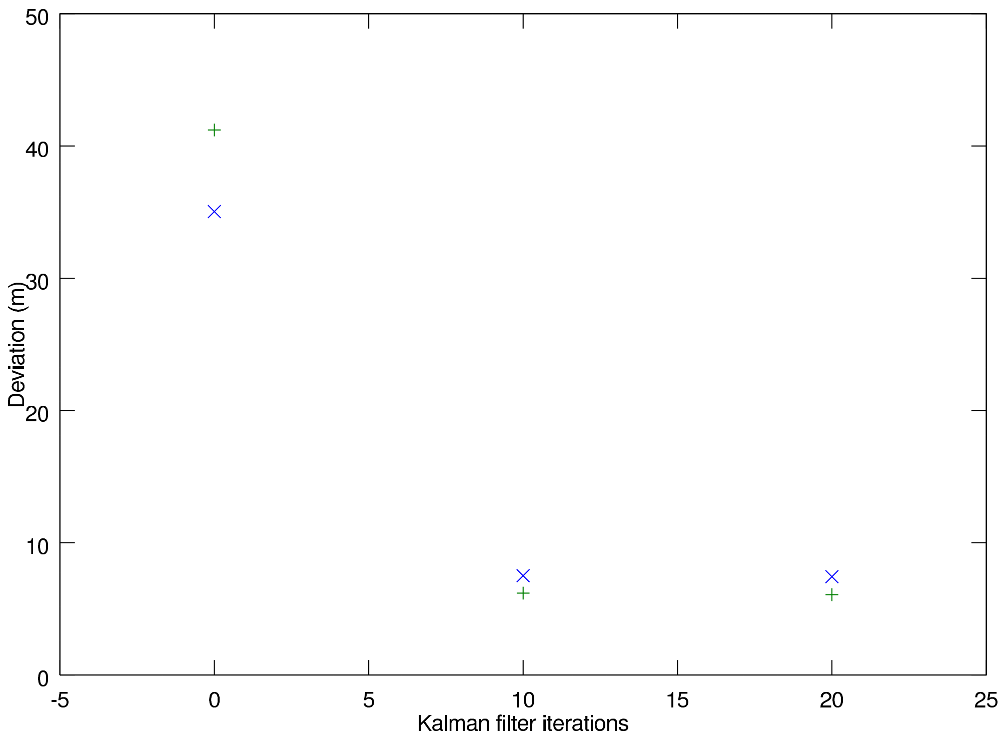

Figure 5.6. *WLAN convergence test (measurement noise 7.0 m): Direction up ('x'), direction down ('+')*

Chapter 6

Discussion

The cyclic course tests in section 5.2.1 show no real difference for the different choices of initial prior noise, nor different measurement noise. It is possible that the different choices of noise values were not separated enough and that more varied values would have been needed for noticeable differences. However, since our grid of reference points does not allow us to determine the WLAN estimate more accurately than 3.5 m it is only reasonable to test for higher measurement noises R. This will only lower the significance of the WLAN localization estimate and does mean that the final estimate from the Kalman filter will favor (and be "pulled" towards) the dead reckoning estimate.

However, the tests do show that combining dead reckoning with WLAN localization yields consistently better localization results than using the dead reckoning-only approach, with the deviation kept below 10 m as opposed to a slowly increasing deviation up to about 26 m after 280 m of walking. An error of 10 m might not be accurate enough to place users in the correct room in office environments but should be sufficient concerning larger indoor environments, for example shopping malls, warehouse buildings, etc. In the case where the dead reckoning estimate is placed outside of the grid of reference points, the implementation will "pull" the final estimate closer to the grid and also the correct position. Therefore it can be argued that the map of reference points covered a too small area and that it would be interesting to see how this WLAN localizer would fare on a larger map of reference points.

The WLAN convergence tests in section 5.2.2 show that, without the addition of uncertainty from walking, the localization estimate will converge to a position which deviates about 7 m from the correct position within ten iterations of the Kalman filter, with initial deviations ranging from 35 m to 92 m. The exception is when using the *faced-down* likelihood distribution for measurement noise $R = 3.5$ m, where the deviation is a bit higher than the other three results. This is a peculiar result considering that the same likelihood distribution also yielded the best estimate for $R = 7.0$. This could have been logically explained by a flawed measurement model and that the Kalman filter weighs the WLAN estimate more lightly as R increases, unless this test was carried out with WLAN localization only. Therefore we would need to re-measure for a larger set of samples to see if the average result changes. Again, a deviation of about 6-7 m is not quite accurate enough for smaller indoor environments such as office spaces, but close. The tests also show that the implementation is an improvement over location estimates from the GPS, but again, testing over a larger area would be interesting to confirm.

The final test scenario, the "randomly walking" scenario in section 5.2.3 showed that the implementation succeeds in the majority of the cases, even if it is not by a large margin. What is worth analyzing here is how the Kalman filter estimate was stuck to a corner, in a majority of the times that it failed. There can be several reasons for this. This can be caused by a flawed measurement model that steers the Kalman filter estimate to the wrong area, followed by convergence, which means that the Kalman filter noise P_k has become so low that the Kalman filter barely takes new measurements (WLAN particle filter estimates) into consideration in the correction phase. So while walking under these conditions yields correct relative displacements, the location estimates are incorrect in absolute terms. The fact that the localizer has a very strong belief in an incorrect estimate makes it potentially very difficult to recover from such errors, unless we manually increase or reset the Kalman filter noise P_k and let the localizer start over again. Along with increasing the noise of the Kalman filter, we can redistribute the particles of the WLAN particle filter *globally* (instead of locally) as re-initialization. Globally in this context means across the entire map of reference points. This might not be a desirable solution though, in case the map is very large while the number of particles are low and/or the computation power of the localizer hardware is low.

These recovery approaches presume that the user or the localizer itself is aware of the tracking error. Therefore, the first step to recovering from such tracking errors is to make the localizer aware of them. One such detection method is referred to as *validation gates* [22]. This is based on the presumption that the measurement residual will be relatively high when a tracking error has occurred, which for this project means that the WLAN localizer estimate will differ significantly from the final estimate of the Kalman filter. By looking at the behavior of the measurement residuals, we can detect unexpectedly high measurement residuals and suspect that something is out of order.

Chapter 7

Conclusion

7.1 Summary

In this project, a localization solution for GPS-obstructed areas has been suggested and tested. The implementation uses a Kalman filter to fuse the estimates from a dead reckoning localizer and a WLAN localizer. The dead reckoning localizer assumes pedestrian use, and utilizes step detection and a dynamic step length to update the location estimate as the user strolls along. The dynamic step length is the relation between the step length and the step period, determined by linear regression. This dead reckoning location estimate is corrected by the WLAN localizer result, which uses a probabilistic particle filter approach to estimate the location based on signal strength observations from the WLAN infrastructure. Tests show that WLAN localization helps to reduce the errors introduced by the dead reckoning approach, and that the average deviations from the correct position mostly are in the range of 5-10 m depending on the situation. This might not be accurate enough to place the user in the correct office in narrow office environments but should be sufficient for localization in open and larger indoor spaces, such as shopping malls, warehouse buildings, etc.

7.2 Future Work

To improve the general accuracy of this localization implementation, there are several methods to examine. The most immediate actions are to experiment with redistributing the particles in the particle filter and changing the confidence of the Kalman filter estimate during online localization. This is to recover from localization errors, if we suspect that the localization has failed. This could be activated manually, or automatically by monitoring specific values and triggering at specific thresholds: for example, if the WLAN estimate would deviate "too much" from the DR/KF estimates for a consecutive number of WLAN particle filter iterations. To detect potential tracking errors, we can implement some form of *validation gates* [22], which is one such automatic detection approach, as mentioned in chapter 6.

Additionally, to keep the localizer “on track”, we can utilize a map or a transition model to rule out “impossible movements" such as walking through walls, and reset the estimate to the nearest possible location, for example a narrow passage, elevator, escalator, etc. This is related to the map matching [14] mentioned in section 2.2.

Regarding improvements to the measurement model itself, an alternative is to try another probability distribution for representing the likelihood distribution, for example the Weibull distribution as suggested in [21]. We can also try to use a denser grid of reference points.

Additionally, we can increase the number of particles (make more guesses) in the case of a large map that is not sufficiently covered. This could be done statically in the implementation or dynamically during online localization. Note, as mentioned earlier, that the need for a large particle set is not as critical on a map with a limited number of discrete reference points.

Looking beyond the WLAN localization, in order to make the system more convenient to use, we need to loosen the restrictions for using the system. That is, the user should not necessarily have to hold the device in a specific pose during localization but instead be able to place the device freely in a shirt, trouser pocket, on a backpack, etc. There is some interesting related work mentioned in section 2.1 about these experiments.

Bibliography

[1] M Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2), February 2002.

[2] Haris Baltzakis. Kalman/particle filters tutorial. Presentation slides, November 2004.

[3] Stéphane Beauregard. Omnidirectional pedestrian navigation for first responders. *4th Workshop on Positioning, Navigation and Communication, 2007*, pages 33–36, 2007.

[4] Stéphane Beauregard and Harald Haas. Pedestrian dead reckoning: A basis for personal positioning. *Proceedings of the 3rd workshop on positioning, navigation and communication, (WPNC'06)*, pages 27–36, 2006.

[5] R Bill, C Cap, M Kofahl, and T Mundt. Indoor and outdoor positioning in mobile environments — a review and some investigations on wlan-positioning. *Geographic Information Sciences*, 10(2):91–98, December 2004.

[6] Gunnar Blom, Jan Enger, Gunnar England, Jan Grandell, and Lars Holst. *Sannolikhetsteori och statistikteori med tillämpningar.* Studentlitteratur, 5 edition, 2005.

[7] NOAA's Geophysical Data Center. URL `http://www.ngdc.noaa.gov/geomagmodels/Declination.jsp`. Fetched November 2011.

[8] Wei Chen, Zhongqian Fu, Ruizhi Chen, Yuwei Chen, Octavian Andrei, Tuomo Kröger, and Jianyu Wang. An integrated gps and multi-sensor pedestrian positioning system for 3d urban navigation. *2009 Urban Remote Sensing Joint Event*, 2009.

[9] Keith Copsey. Tutorial on particle filters. Presentation slides.

[10] Gerd Eriksson. *Numeriska algoritmer med MATLAB.* Nada, KTH, 2002.

[11] Yasutaka Fuke and Eric Krotkov. Dead reckoning for a lunar rover on uneven terrain. *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, pages 411–416, 1996.

[12] Uwe Grossmann, Christof Röhrig, Syuzanna Hakobyan, Thomas Domin, and Matthias Dalhaus. Wlan indoor positioning based on euclidian distance and interpolation (isobars), 2006.

[13] Rommanee Jirawimut, Piotr Ptasinski, Vanja Garaj, Franjo Cecelja, and Wamadeva Balachandran. Method for dead reckoning parameter correction in pedestrian navigation system. *IEEE Transactions on Instrumentation and Measurement*, 52(1):209–215, February 2003.

[14] Edward J. Krakiwsky, Clyde B. Harris, and Richard V.C. Wong. A kalman filter for integrating dead reckoning, map matching and gps positioning. *Position Location and Navigation Symposium, 1988. Record. Navigation into the 21st Century. IEEE PLANS '88*, pages 39–46, 1988.

[15] Tuomo Kröger, Yuwei Chen, Ling Pei, Tomi Tenhunen, Heidi Kuusniemi, Ruizhi Chen, and Wei Chen. A method of pedestrian dead reckoning using speed recognition. *Ubiquitous Positioning Indoor Navigation and Location Based Service (UPINLBS)*, pages 1–8, 2010.

[16] Kai Kunze, Paul Lukowicz, Kurt Partridge, and Bo Begole. Which way am i facing: Inferring horizontal device orientation from an accelerometer signal. *2009 International Symposium on Wearable Computers)*, pages 149–150, 2009.

[17] Binghao Li, Andrew Dempster, Chris Rizos, and Joel Barnes. Hybrid method for localization using wlan. *Spatial Sciences Conference*, September 2005.

[18] Binghao Li, James Salter, Andrew G. Dempster, and Chris Rizos. Indoor positioning techniques based on wireless lan. *LAN, First IEEE International Conference on Wireless Broadband and Ultra Wideband Communications*, 2006.

[19] Binghao Li, Yufei Wang, Hyung Keun Lee, Andrew Dempster, and Chris Rizos. Method for yielding a database of location fingerprints in wlan. *IEE Proceedings - Communications*, 152(5):580–586, October 2005. ISSN 1350-2425.

[20] Hui Liu, Houshang Darabi, Pat Banerjee, and Jing Liu. Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics - part C: Applications and Reviews*, 37(6):1067–1080, November 2007. ISSN 1094-6977.

[21] Jingbin Liu, Ruizhi Chen, Ling Pei, Wei Chen, Tomi Tenhunen, Heidi Kuusniemi, Tuomo Kröger, and Yuwei Chen. Accelerometer assisted robust wireless signal positioning based on a hidden markov model. *IEEE*, 2010.

[22] Rudy Negenborn. Robot localization and kalman filters. Master's thesis, Utrecht University, 2003.

[23] Ulf Rerrer and Odej Kao. Suitability of positioning techniques for location-based services in wireless lans. *Proceedings of the 2nd workshop on positioning, navigation and communication (WPNC'05) & 1st Ultra-wideband expert talk (UET'05)*, pages 51–56, 2005.

[24] Kurt Seifert and Oscar Camacho. Implementing positioning algorithms using accelerometers. *Freescale Semiconductor Application Note*, 2007.

[25] Ulrich Steinhoff and Bernt Schiele. Dead reckoning from the pocket - an experimental study. *Conference on Pervasive Computing and Communications (PerCom), 2010 IEEE International*, pages 162–170, 2010.

[26] Sebastian Thrun. Probabilistic algorithms in robotics, April 2000.

[27] Chris Veness. Calculate distance and bearing between two latitude/longitude points using haversine formula in javascript, 2010. URL `http://www.movable-type.co.uk/scripts/latlong.html`. Fetched January 2012.

[28] Greg Welch and Gary Bishop. An introduction to the kalman filter. *UNC-Chapel Hill*, 2006. TR 95-041.

TRITA-CSC-E 2012:093
ISRN-KTH/CSC/E--12/093-SE
ISSN-1653-5715