Indoor-localization using a mobile phone

Ruud Henken

May 14, 2014

Master Thesis

Artificial Intelligence

University of Groningen, The Netherlands

Internal supervisor:

Dr. M.A. Wiering (Artificial Intelligence, University of Groningen)

External supervisor:

T.P. Schmidt, Msc. (Sense Observation Systems B.V., Rotterdam)

university of groningen / faculty of mathematics and natural sciences / artificial intelligence

Abstract

In an era of mobile communication, the demand for indoor-localization is increasing. For instance, users could benefit from indoor-localization systems in complex indoor environments such as large shopping malls, museums, and location-based services (e.g. warn travelers to hurry to their gate). It is commonly known that GPS is unsuitable and inaccurate for these kinds of applications. Therefore, in this thesis the feasibility of indoor localization based on mobile phone motion and WiFi sensors is researched. Methods such as circular lateration, K-nearest neighbours, and extended Kalman filter simultaneous localization and mapping are assessed on their accuracy and applicability. The effects of building characteristics and mobile-phone features on the performance of these methods are discussed. Furthermore, it is shown that model-based methods such as K-nearest neighbours outperform memory-less methods like fingerprinting.

Contents

Chapter 1

Introduction

In an era of mobile communication the demand for indoor-localization is increasing. Users could for instance benefit from indoor-localization systems in complex indoor environments such as large shopping malls, hospitals, and museums. An increasing amount of mobile devices (such as mobile phones) use the Global Positioning System (GPS) [15] for outdoor localization. GPS devices use a technique called trilateration to determine receiver location. A key aspect of this technique is satellite signal strength. If the GPS device is unable to receive a proper signal from at least four satellites, the GPS device fails its task. Indoor-localization systems might benefit from using GPS but it is commonly known that mobile phone GPS sensors fail to operate in indoor localization situations. Furthermore, GPS devices are often not accurate enough for indoor applications.

Applications of an indoor-localization system are plentiful, to name a few:

1. Victim search: locate victims and fire fighters inside large buildings.
2. Tracking expensive or important objects. For instance artworks in a museum or the portable X-ray machine in a hospital.
3. Indoor navigation at large conferences or expos.
4. Location-based services such as: (1) warn travelers to hurry to their gate, (2) offer relevant discounts to shoppers, or (3) additional information to art in a museum.

This thesis reports on indoor-localization and indoor-localization and mapping methods based on mobile phone sensor data, without installing additional devices in the building (such as RFID tags) that might aid the localization process. Users should be able to use the localization system 'off-the-shelf'. The system should be applicable in any building, without having to install additional devices such as RFID tags or other transmitter/receiver related systems (e.g. site management or building management should not have to make adjustments to the building before users could benefit from the proposed system).

1.1 Related work

Techniques used for localization have changed over time, but the problem has always remained the same: how to determine your position, relative to some point in space. Let it be a combination of satellites orbiting around earth (GPS) or the Pole star. Indoor Positioning Systems (IPS) are already commercially available, but all of these solutions rely on additional transmitters/receivers which provide feedback about the current position of the receiver. In this thesis we aim to develop a solution which operates without any additional hardware.

Indoor-localization and mapping using mobile phones shows great resemblance with localization challenges in robotics. In both situations the phone/robot has no prior knowledge about the environment and is interested in its location. The key difference between locating mobile phones and robot localization is the available information on which localization is performed. In robotics, odometry and landmark based navigation are two commonly used features [12]. However, these are not available in mobile phones, in particular odometry. In robotics is the usage of Simultaneous localization and mapping (SLAM) [7] is a commonly used technique for simultaneous map making and localization in previously unseen environments. The SLAM problem is an example of a chicken-and-egg problem: a map is needed for localization, while accurate knowledge about your current location is needed in order to build that map. Nowadays several implementations for SLAM are available, but existing versions of SLAM are not applicable because these algorithms rely on information which is not available in mobile phones. An interesting related method is FootSLAM [28]. FootSLAM uses data from an inertial sensor mounted on a shoe to track user movement. These movements are used to build maps of the environment and thereafter used for localization. Inertial sensors are commonly integrated sensors in mobile phones which makes FootSLAM an interesting method. However, observations made by foot-mounted inertial sensors may greatly differ from those in mobile phones. Dead reckoning might be used in a similar way using inertial sensors [32]. However, dead reckoning suffers greatly from cumulative error.

Nowadays most non-residential buildings have multiple WiFi access points installed throughout the building. Each access point has a unique identifier (MAC-address) and signal strength corresponding to the distance to that access point. Earlier studies report on localization based on WiFi access point radio fingerprinting [16]. IPS may greatly benefit from WiFi fingerprinting. WiFi fingerprinting might provide the landmarks needed for the above mentioned techniques.

In literature, the best performance is achieved by using a combination of inertial sensors and WiFi access points. Some studies use only inertial sensors [25] or only WiFi access points [9], others use hybrid methods of inertial sensors and WiFi access points [4]. The latter is called WiSLAM and shows an increase in performance over FeetSLAM

[29] (which is the successor of FootSLAM) by adding signal strength measurements. In WiSLAM, RSSI readings are added to (multiplied with) the likelihood function of the FootSLAM algorithm.

The drawback of the studies discussed above is that users often require additional hardware such as a foot mounted IMU [29]. In this thesis we aim at methods that require only a mobile phone without any additional hardware.

1.2 Research questions

The main challenge in this thesis is to describe the environment and develop an indoor-localization system based on observations made solely by mobile phone sensors. For instance, we would like to be able to distinguish a coffee corner from the entrance of a building based on mobile phone sensor data. Ideally, mobile phone sensor readings provide enough information to accurately localize the mobile phone. This leads to the following research questions:

1. *How can mobile phone sensors be utilized for (1) indoor-localization and (2) indoor-localization and mapping?*

   (a) *What is the maximum accuracy of the evaluated methods, fingerprinting, K-nearest neighbours, and circular lateration, on indoor-localization?*

   (b) *Which of the evaluated methods, dead reckoning, extended Kalman filter simultaneous localization and mapping (EKF SLAM), and multi-user EKF SLAM, performs best on device tracking?*

   (c) *Which of the evaluated methods, EKF SLAM and multi-user EKF SLAM, performs best on localization and mapping?*

1.3 Outline

In Chapter 2 mobile phone sensors are discussed that are potentially interesting for indoor-localization. Chapter 2 also reports on mobile phone sensor characteristics and algorithms to translate raw sensor data into meaningful input for localization and mapping algorithms. In Chapter 3 multiple methods for localization and mapping are presented. The first part of Chapter 3 is focused on motion-based methods, followed by Radio Frequency (RF) based methods. The last part of Chapter 3 deals with methods combining motion sensors and RF-sensors. In Chapter 4 the performance of all localization and mapping methods presented in Chapter 3 are discussed. In Chapter 5 this thesis is concluded and research questions posed in the previous section are answered. Chapter 5 also reports on open challenges for future research.

Chapter 2

Sensor Selection and Environment

In this Chapter sensors for mapping and localization are discussed. Android OS was chosen as the platform for testing localization and mapping algorithms because it is the platform with the most open character and provides the most features to developers. Depending on the manufacturer and type of mobile phone, Android offers access to a multitude of interesting sensors. The most commonly available sensors include:

- Motion sensors: accelerometer, gyroscope, magnetic field, and orientation
- Environmental sensors: WiFi, Bluetooth, NFC, and GSM
- Ambient sensors: barometer, thermometer, and photometer

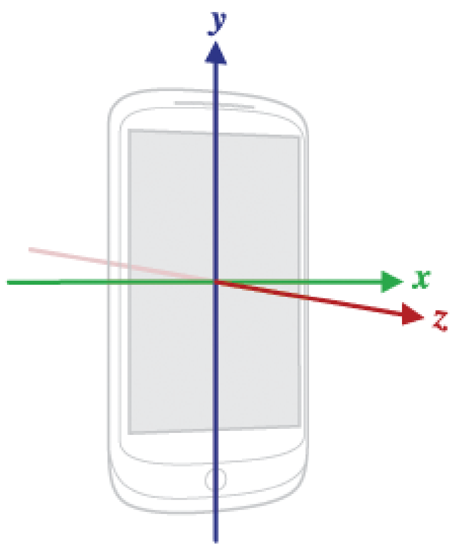
Sensors that are useful for localization and mapping are discussed below.

2.1 Coordinate systems

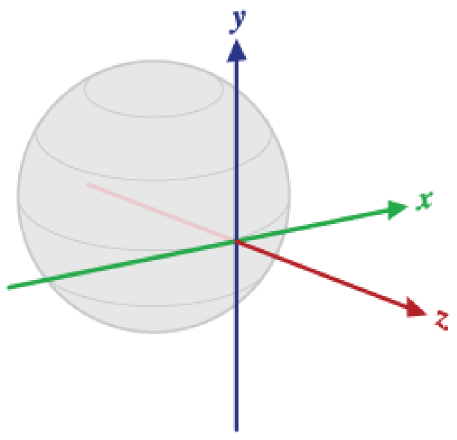
Nearly all Android motion sensors measure their values with respect to the coordinate system of the device (see Figure 2.1(a)). It is important to realize that this is not the same as the earth coordinate system. For instance, an increase of gyroscope values in a particular direction creates insight in the angular velocity of the phone but does not say anything about the absolute orientation of the phone in the physical world. In order to make the sensory output useful for mapping and localization purposes, it is necessary to create a mapping from Android sensors to the earth coordinate system (see Figure 2.1(b)). This translation from one coordinate system to the other is discussed below, but first some properties of the coordinate systems are discussed in more detail.

In the Android coordinate system, the x, y, and z axes are defined relative to the screen of the phone. If the phone is held in front of the user facing the screen, the x-axis is defined as horizontal with positive values on the right. The y-axis points vertically with positive values at the top. The z-axis points outward of the screen with positive values towards the user. In the earth coordinate system, the x-axis always points east and is tangential to the ground at the current location of the device. The y-axis is also tangential to the ground at the device's current location but points toward the geomagnetic North Pole. The z-axis points toward the sky and is perpendicular to the plane defined by the x and y axes.

For convenience device orientation is represented using quaternions. A quaternion is a vector of length 4, where the last three numbers of this vector are reserved to again represent a vector, and the first number of the quaternion is used to represents a rotation around this axis. So instead of a vector, a quaternion is actually a composite structure where the first part is a real number and the latter part is a vector of length 3. Quaternions are commonly used in data visualization applications to rotate objects. An advantage of quaternions over other methods such as Euler angles and matrices is that of compactness and relatively simple object rotation. Even though quaternions are non-commutative they remain much easier to operate on compared to other methods.



(a) Coordinate system of an Android device



(b) Coordinate system for the earth

FIGURE 2.1: Coordinate systems. Figure (a) depicts the coordinate system used for an Android device. Figure (b) depicts the coordinate system used for the earth.

2.2 Motion sensors

Motion sensors are interesting for localization and mapping because they provide valuable information about the distance and bearing travelled by a user. A typical Inertial

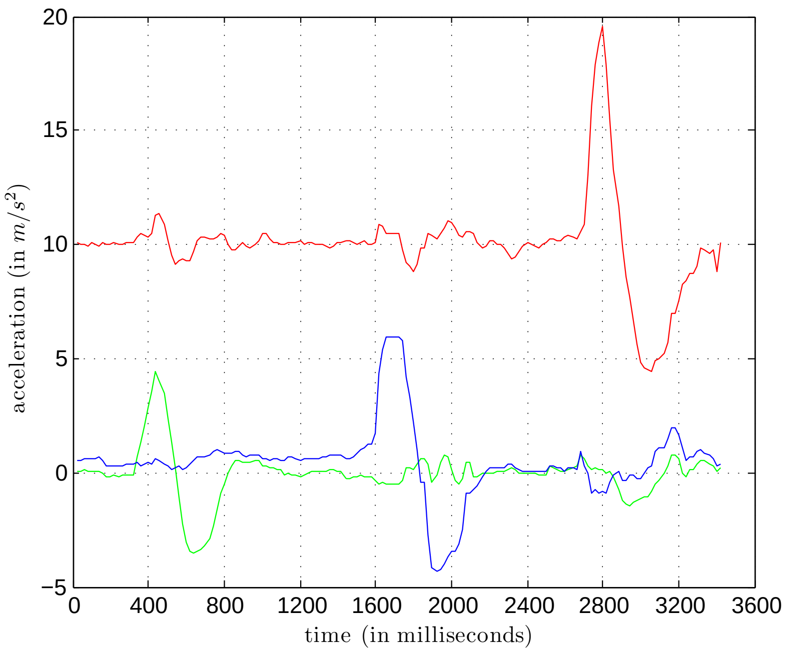
Measurement Unit (IMU) such as used in [28] consists of a six degrees of freedom unit (6DOF) composed of a three-axial accelerometer and a three-axial gyroscope. In this thesis the phone's three-axial magnetic field sensor is also included as a motion sensor. The magnetic field sensor is included because it can be used to measure orientation of the phone and hence provide information on the direction of a motion. Self-evidently the magnetic field sensor can be operated like a compass. This set of motion sensors (accelerometer, gyroscope, and magnetic field sensor) distinguishes itself from other sensors in the sense that they do not require any external hardware. Motion sensors detect ever present physical events such as gravity and magnetic fields, whereas other sensors - such as Bluetooth - require external devices before they become useful.

Typical output of an Android motion sensor (accelerometer) is depicted in Figure 2.2. Each motion sensor measures in three dimensions. The green, blue, and red lines represent the x, y, z axis respectively. Sensor data presented here is generated by consecutively moving the device in the positive direction of the x, y, and z axis.

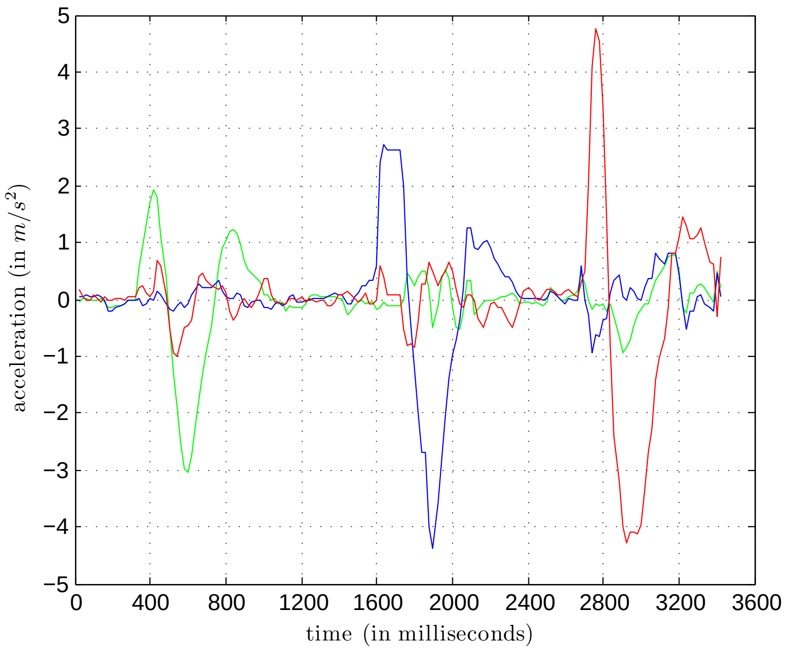
2.2.1 Accelerometer

Nearly all modern phones are equipped with three-axial accelerometers. The abundant availability of this sensor makes it a compelling sensor to consider for mapping and localization. Accelerometer values are measured with respect to the device (see Figure 2.1(a)) and are measured in standard SI-units of m/s^2. Some typical output of the accelerometer is shown in Figure 2.2. This activity was generated by consecutively displacing the phone along the x, y, and z axis. The whole sequence of displacements takes about $3.5s$. Regardless of the orientation of the phone there is the ever present gravitational force of approximately 9.81 m/s^2 pulling at the device. The gravitational component is also visible in the accelerometer readings. Unless the device is in free fall, of course. Together with other sensors, an accelerometer can be used to measure activity and orientation of the phone. Theoretically it is possible to obtain speed and travelled distance from the accelerometer by doubly integrating accelerometer values over time. However, this method is known to have a rather poor accuracy. In Section 2.2.4, doubly integrating accelerometer values and an alternative method are discussed for measuring travelled distance using an accelerometer.

Android also provides a 'linear acceleration' sensor which is basically the same as the default accelerometer but without the gravity component. The linear acceleration sensor can therefore be considered as the net acceleration. Figure 2.2(b) displays sensor data from the linear accelerometer which was generated by exactly the same sequence of displacements as used for the accelerometer.



(a) Accelerometer



(b) Linear accelerometer

FIGURE 2.2: Acceleration data obtained from a Samsung SII by consecutively moving the phone in the positive direction of the x, y, and z axis. The axes are represented by the green, blue, and red lines respectively. Figure (a) depicts accelerometer data, Figure (b) depicts linear accelerometer data (i.e. without the gravity component).

2.2.2 Gyroscope

A gyroscope measures angular velocity and can be used to assist the magnetic field sensor and accelerometer in acquiring high precision motion information. Gyroscope sensor values are measured in radians per second (rad/s) with respect to the device's coordinate system (see Figure 2.1(a)). Nowadays it is very common for phones to be equipped with a three-axial gyroscope which measures angular velocity at all three axes of a device. A gyroscope might be considered redundant if there is also a magnetic field sensor available, because both a gyroscope and a magnetic field sensor can be used to measure changes in rotation of the phone. But surely a gyroscope can't replace a magnetic field sensor considering a magnetic field sensor is necessary to obtain a device's orientation with respect to the North Magnetic Pole. The latter can't be obtained using just a gyroscope. Gyroscope measurements can therefore only be used to aid the magnetic field sensor for better precision. In general the magnetic field sensor in a phone is much slower compared to the gyroscope. Utilizing both provides short-term information on quick changes via the gyroscope and long-term overall orientation information via the magnetic field sensor, which makes it a perfect coalescence.

Typical output of a gyroscope is depicted in Figure 2.3. The data shown here are generated by consecutively rotating the phone along the x, y, and z axis of the device. The whole sequence of rotations takes about $8s$.

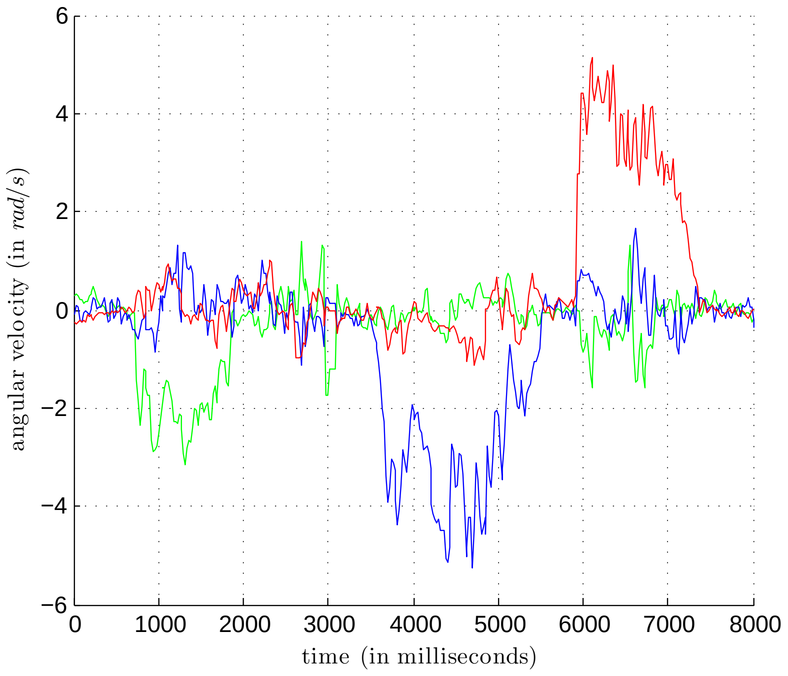


FIGURE 2.3: Gyroscope sensors values generated by rotating the phone consecutively along the x, y, and z axis. These axes are represented by the green, blue and red lines respectively.

2.2.3 Magnetic field

A magnetic field sensor measures orientation of the phone with respect to the magnetic North. Sensor values are measured in Earth's magnetic field in each direction (x, y, and z), and can be utilized to detect the orientation of the phone with respect to the

Earth's surface. Magnetic field sensor values are measured in μT (micro-Tesla). It is commonly known that the magnetic field sensor of mobile phones is very sensitive to local disturbances and can easily be fooled. Electronic devices such as a refrigerator or even some loudspeakers can easily disturb measurements. Other local disturbances can be caused by structural parts of buildings such as reinforced concrete. Local disturbances in the magnetic field are considered as noise, but can, on the other hand, also be exploited as features. In a study by [36], it was found that local disturbances in the magnetic field can be used for localization purposes. In [36] 'Organic Landmarks' are proposed which consist of a combination of local disturbances in the magnetic field, and may include radio based features like 3G and WiFi, and optionally activity from motion sensors. These features cannot be defined a priori and have to be learnt online.

The magnetic field sensor is an important sensor that will provide necessary input for dead reckoning experiments (see Section 3.1.1) and hybrid methods such as SLAM (see Section 3.3).

Next, two algorithms are discussed for reliably measuring distance and orientation/bearing in respectively Section 2.2.4 and Section 2.2.5.

2.2.4 Distance measurement

The three motion sensors discussed above (accelerometer, gyroscope, and magnetic field sensor) provide a powerful combination to measure distance and bearing. Unfortunately only bearing can be measured directly using the magnetic field sensor. Speed (and hence also distance) on the other hand can only be measured indirectly using accelerometers. Probably the most obvious solution to estimate phone displacements is by doubly integrating accelerometer values over time (accelerometer values are measured in m/s^2). However, this requires knowledge about the initial orientation and velocity of the phone. If the orientation is off or if the measurement algorithm is started while the phone is already moving (e.g. in a car, travelling at a velocity of 100 km/h), accelerometers would indicate incorrect information. Furthermore, doubly integrating accelerometer values is known to be prone to drift errors. In [23], doubly integrating accelerometer values is used to determine device velocity and position and a Kalman filter is used to correct drift errors.

The approach taken here is slightly different. Instead of measuring displacements by integrating accelerometer values, a step counting algorithm is used. The advantage of counting steps over other methods is that steps are easy to detect and are not prone to drift errors. Methods based on integrating accelerometers need to be calibrated periodically by use of external measurements of position [23]. On the other hand, step counting algorithms have the disadvantage that steps are only indirectly correlated to distance.

Every person has a unique gait which implies a unique step size for every person. Knowledge about the number of steps therefore does not directly imply knowledge of travelled distance. According to human physiology studies there are similarities between human gaits and according to [18] the average person's stride length is approximately 0.8 meter with a variance of 0.1 meter. This and other knowledge on physical limitations of human gaits provide further information to define a step counting algorithm. In general, humans are unlikely to walk any faster than 5 steps per second. Using this and other information, a step counting algorithm is proposed in Algorithm box 1. The algorithm presented here closely resembles step counting algorithms presented in [34] and [26]. Both [34] and [26] propose a step counting algorithm based on accelerometer values. In [34] a sliding window approach is used. For each window the minimum and maximum values are determined, and based on these minimum and maximum values a dynamic threshold value is determined. If the currently considered data point exceeds the dynamic threshold and is within a specified time window, it is considered as the detection of a step. An important difference between the algorithm presented here and the one presented in [34], is that the method here operates on all three axes simultaneously whereas the method of [34] first determines the axis with highest activity (as this is considered to contain the most information). By taking into account all three axes simultaneously, step detection is made independent of device orientation. The algorithm presented in [26] closely resembles a peak counter, but is designed for offline usage. In [26] a control point is used to determine whether a peak has occurred. Every data point that resembles a peak is counted. For each peak it is also determined whether the peak lies below or above a time window average. As soon as the end of the dataset is reached, the total number of steps is determined by subtracting all under average peaks from the total number of peaks. All three methods have in common that (1) there is in some way noise cancellation, either directly by setting a parameter, or indirectly by computing a dynamic threshold value, and (2) successive steps cannot follow within a certain time window, and (3) step detections are limited by human physiological capabilities.

The algorithm presented here and described in Algorithm box 1 is basically a peak counter. Activity is generated by the linear accelerometer (See Section 2.2.1). In the algorithm presented here, ϵ is a threshold value. Any activity generated below this threshold is considered to be noise. τ is a minimum step interval (in seconds) based on limitations of human physiology. In a pilot experiment τ was determined to be approximately $20ms$. δ holds the timestamp at which the last step was detected, and Σ is the total number of steps counted up to time t. μ is the average activity of the interval considered at time t, and, depending on the sampling frequency, contains approximately 50 datapoints of the last second (for 50 Hz). a_{tn} is the activity at time tn, where $n = \{0, \ldots, -3\}$.

A side note about step counting algorithms is that they are reactive. The algorithm described in Algorithm box 1 is an example of an odometer (retrospective), which differs

Algorithm 1 Step counting algorithm

```
1:  $\epsilon \leftarrow activity\ threshold\ (in\ m/s^2)$
2:  $\psi \leftarrow window\ size\ (in\ s)$
3:  $\tau \leftarrow minimum\ step\ interval\ (in\ s)$
4:  $\delta \leftarrow (current\ time - \tau)\ (in\ s)$
5:  $C \leftarrow 0$
6:  while true do
7:    if $(new\ sensor\ data)$ then
8:      $t \leftarrow current\ time$
9:      $\mu \leftarrow average\ activity\ (accelerometer\ magnitude)$
10:     $a_{tn} \leftarrow activity\ at\ time\ tn,\ where\ n = \{0, \dots, -3\}$
11:     if $(a_{t0} > \epsilon)$ then
12:       if $(a_{t0} > \mu)$ then
13:         if $(a_{t0} > a_{t-1}\ AND\ a_{t0} > a_{t-2}\ AND\ a_{t0} > a_{t-3})$ then
14:           if $(\delta + \tau < t)$ then
15:             $\delta \leftarrow current\ time$
16:             $C++;$
17:           end if
18:         end if
19:       end if
20:     end if
21:   end if
22: end while
```

from prediction based methods (pro-active). This subtle difference is important for mapping and localization methods because of a timing component. Information on a new location at time t_0 might either be predicted by some probabilistic method (FIR-filter, function fitting, slow speed decay) or, it might be reactive (peak counter, odometer) and represent the actual distance travelled. It makes sense to be aware of this because the bearing sensor presented in the next Section operates in real time, whereas the step counter presented here does not. If there is a shift in time between the bearing sensor and step counter (even if the offset is very small), motion sensing would eventually diverge. By introducing a short delay in the bearing sensor (use the bearing of the previous state in the current prediction state), the step counter and bearing sensor are always in sync.

The performance of the algorithm presented in Algorithm box 1 is evaluated with several user tests. The algorithm was implemented in a simple Android application and multiple experiments have been conducted to test the performance on step counting and distance predictions. Results of these experiments are presented in Chapter 4.

2.2.5 Bearing measurement

By default Android is provided with an orientation sensor. This sensor indicates the orientation of the phone with respect to the ground at its current location. The orientation sensor is extremely useful if we are interested in the walking direction of a user (i.e.

bearing). The drawback of this default Android sensor is that: (1) it is slow. The maximum frequency is about 10 Hz, which limits the maximum frequency at which mapping and localization algorithms can operate, and (2) it is inaccurate. In a pilot experiment it was noted that compared to actual orientation, the Android orientation sensor is sometimes off by more than 180 degrees, which, obviously, is catastrophic for mapping and localization methods. And finally (3), it does not take into account measurements from the gyroscope. The default Android orientation sensor only considers the accelerometer and magnetic field sensor. These two sensors combined provide sufficient information to deduce orientation, but it would be beneficial to include the gyroscope into the equation. Mainly because a gyroscope improves detail and stability compared to the magnetic field sensor. As discussed before, the sampling rate of the gyroscope is much faster compared to the magnetic field sensor. Furthermore, SLAM methods are known to be very sensitive for any error in bearing. It is therefore interesting to develop an improved bearing sensor that combines accelerometer, magnetic field sensor, and gyroscope for a better and more reliable orientation sensor.

Such an algorithm is proposed in [31]. The algorithm presented in [31] is basically a sensor fusion algorithm designed for a 9DOF inertial measurement unit. The proposed algorithm is designed for a specific inertial measurement unit, but in this thesis it is applied to the set of three-axial mobile phone sensors (accelerometer, gyroscope, and magnetic field sensor). The method is based on a two-stage Extended Kalman Filter (EKF). Generally EKF consists of a prediction step and a correction step [37]. However, in the proposed method, the filter is modified to include two correction steps instead of one. The two correction steps are called stages, hence the name. At first (the prediction step), gyroscope values are used to set an orientation estimate of the phone. Next up, in the first correction stage, the estimated orientation is corrected using the accelerometer. The accelerometer is used to correct the plane defined by the x and y axes by means of the gravitational component. The correction component for the z-axis is set to 0 in this correction stage because the accelerometer does not provide information about the bearing anyway. In the second correction stage, the phone orientation is corrected using data from the magnetic field sensor. Performance of this algorithm compared to the Android orientation sensor is presented in Chapter 4.

It is important to realize that device orientation is not equal to bearing. Having information about the orientation of the phone is one thing, but extracting bearing from this orientation is another. Bearing is related to the force exerted on the phone corrected by the phone's orientation. The only information available at this point is the orientation of the phone in the physical world. Extracting information about bearing, based on an already unreliable prediction of phone orientation, is left for further research. In this thesis the user is restricted to point the top of the phone in the direction of movement with the screen facing towards the user.

2.3 Environmental sensors

In this Section the properties and suitability of WiFi, Bluetooth, GSM, and GPS will be discussed. These environmental sensors are interesting because they provide additional information about the device location. In many situations environmental sensors are considered to provide redundant information with respect to the motion sensors. For instance, walking with a phone in your pocket will result in high activity of the motion sensor, and signal strength of WiFi, Bluetooth, and GSM are expected to change accordingly. Assuming the uncertainty of motion sensors and environmental sensors are uncorrelated, exploiting the redundancy can reduce overall localization uncertainty.

2.3.1 WiFi

Mapping and localization methods based on WiFi-signals mostly rely either on Received Signal Strength Indication (RSSI) or angle-of-arrival. In most of these systems signal strength is used. Angle-of-arrival is used less often because it requires more sensitive equipment [21]. A key property to recognize when using RSSI for localization is that of signal attenuation. All RSSI-based localization methods rely on the observation that a relation exists between distance between transmitter and receiver and signal strength. In general signal strength decreases when distance increases. A simplified model known as the 'path loss model' describes signal attenuation as a function of antenna characteristics and environmental properties, and is given by [10, pp. 38-47]:

$$P_r = P_t + 10log_{10}K - 10\gamma log_{10}(\frac{d}{d_0}) - \psi(dBm) \tag{2.3.1}$$

where

- P_r is the received signal strength indication (RSSI) measured in dBm.
- K is a unitless constant that relates to antenna characteristics. K is completely determined given d_0 and γ and can be computed using the following equation: $K = -20log_{10}(4\pi d_0/\gamma)$.
- d_0 is a reference distance (i.e. the scope of the transmitter). d_0 is assumed to be $1-10m$ for indoor applications and $10-100m$ outdoors.
- γ is a parameter to describe signal propagation through the environment. γ has values for office buildings with multiple floors between $2 \leq \gamma \leq 6$, and $2 \leq \gamma$ for more open places.
- P_t is the transmitted power of an access point measured in dBm.
- $\psi(dBm)$ is a Gaussian distributed noise component with mean zero and variance $\sigma^2_{\psi_{dBm}}$.

Note that predicates such as 'indoor' and 'outdoor' are on a continuous scale. In terms of signal attenuation there is not really such a thing as 'indoor' or 'outdoor', there is only clutter. A roofed airport terminal would still count as an 'indoor' environment, but 'outdoor' values for d_0 and γ apply. In general, γ-values increase as clutter increases.

Normally signal attenuation is considered to be a bad thing. For localization purposes on the other hand signal attenuation is exploited as a feature. The main challenge of using signal attenuation as a feature is the non-linear aspect. Consider Figure 2.4. Here a typical signal attenuation curve is depicted for non-residential buildings with values $\gamma = 4$, $d_0 = 20$, and $P_t = -40$. From this Figure it becomes clear that measurements at further distances from an access point (AP) are covered by a much smaller range of dBm values compared to measurements closer to it. Distances from 15 to 45 meters are covered by a range of only 20 dBm, whereas distances from 0-15 meters are covered by a much wider range of 70 dBm. This affects the accuracy of distance predictions based on dBm for larger distances. For predictions based on dBm it holds that small changes in the lower regions of dBm result in very large distance fluctuations. Because of this phenomenon distance predictions based on very low dBm values can become unreliable.

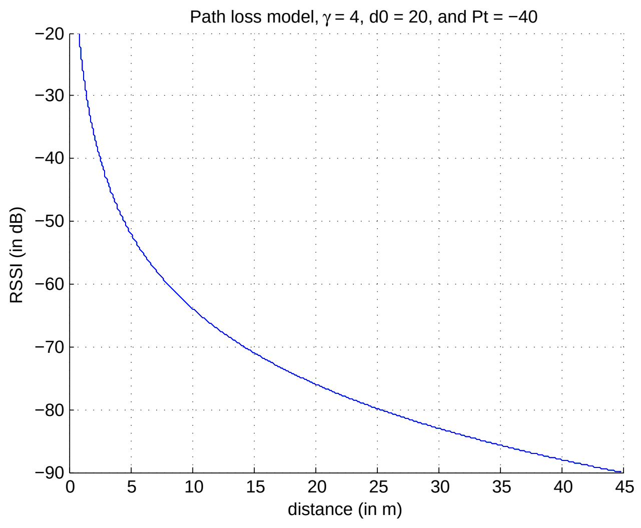


FIGURE 2.4: A typical signal attenuation model for non-residential buildings, with parameter settings $\gamma = 4$, $d_0 = 20$, and $P_t = -40$.

2.3.2 Bluetooth

Similar to WiFi, Bluetooth signal attenuation can be used to determine relative distances. However, the whole idea behind the development of the Bluetooth protocol was to get rid of wired communication systems in favor of having more mobility. As a result, Bluetooth observations do not necessarily provide information about location. Even worse: wearing a mobile Bluetooth headset would cause contradictory results between motion sensors and environmental sensors. The Bluetooth sensors would hint that the user is not moving because the headset is moving along with the user, while at the same time motion sensors might indicate high activity. In a situation where only sensor information is available without feedback, there is by no means an option of eliminating this contradictory information.

Both Bluetooth and WiFi 802.11x operate in the 2.4GHz short-range radio frequency band. As shown by [38], this unmistakably leads to signal interference, even if preventive measures such as Adaptive Frequency Hopping (AFM) [22] for Bluetooth technology are used. Zeng et al. [38] have shown that the use of Bluetooth in combination with 802.11b WiFi signals can severely degrade the performance of the latter and causes significant signal attenuation. It might therefore be argued that it is best to prevent the use of Bluetooth devices at all if we are interested in exploiting WiFi signal attenuation. At least actively promoting the use of Bluetooth devices should be prevented because of signal attenuation issues. The trade-off we face in using Bluetooth devices is in whether considering these devices adds more knowledge (at the cost of WiFi signal attenuation) than is lost by not using Bluetooth devices for localization and mapping.

Mainly because of the high mobility argument mentioned above, Bluetooth observations are not considered in this thesis. One more note about Bluetooth is that it comes in many different protocol versions. A commonly used Bluetooth protocol version in phones is $v3.0 + HS$. In combination with Class 2 this protocol has a signal range up to approximately 10 meters. This is an often used technology for other systems such as wireless game controllers. In theory, Bluetooth communication can be boosted up to far larger distance than 10 meters, but this is not feasible using phones because it requires much more power to transmit messages over larger distances. A more recent development is the availability of Bluetooth Smart/low energy in Android devices (as of Android API level 18). Bluetooth low energy is - as the name suggests - less battery consuming than older versions and can reach up to far larger distances compared to earlier Bluetooth versions. The already fuzzy relation between distance and RSSI values becomes even more obfuscated because of these new protocol versions.

2.3.3 GSM towers

A high density of the GSM network is very important for maximum location accuracy based on GSM towers, but even in urban areas density is often not high enough to include GSM sensors as a feature. In [21], it is shown that for urban areas the maximum accuracy for GSM towers is in the order of 10 meters, but only for outdoor applications.

Compared to other techniques such as WiFi and Bluetooth, GSM towers can only assist in coarse positioning. For instance, in [39], a device is classified as inside or outside a building. In this study the variance in GSM signal strength is exploited (together with other sensors) for indoor/outdoor detection. In [39] it was a deliberate decision to leave out WiFi scanning in favour of cell tower signal strength. The main reason to opt for cell tower RSSI over WiFi RSSI is that by default any phone is already scanning for cell towers, whereas sensors such as WiFi require extra battery consumption. Furthermore, in many outdoor situations WiFi signals are of poor quality and might not even be present at all. Continuously scanning for WiFi networks would then only result in unnecessarily consuming battery power.

For indoor applications a localization error similar to that of GSM errors is unacceptable. A typical office building has corridors and rooms that would not be classified correctly with this amount of uncertainty. It would therefore not be very helpful to include GSM RSSI as a feature for an indoor localization and mapping framework. However, a common problem in SLAM alike methods is drift. In high density GSM areas it might be possible to use GSM cell tower RSSI to eliminate large amounts of drift.

2.3.4 GPS

GPS is a commonly used technique for localization and navigation. The technique depends on satellite measurements and is most frequently used outdoors. Indoor applications are often not considered possible due to poor accuracy. In indoor applications, the accuracy highly depends on the building type and quality of the measuring device. Considering only situations where a GPS fix is possible, meaning that sufficient satellites are visible for the technique to operate normally, and using an embedded GPS sensor from a mobile phone, accuracy is approximately $10m$ in wooden and brick buildings, and even decreases to more than $10m$ in shopping malls [17]. In some cases it might take up to one minute and longer to get a good GPS fix, which is extremely inconvenient for battery powered devices, and sometimes it is not possible to get a GPS fix at all.

Others have reported slightly better results $(6 - 12m)$ for GPS applications [27]. For outdoor applications, such as navigation on highways, an accuracy of approximately $10m$ is most often sufficient, but for indoor applications these results are unacceptable.

GPS may however be included as an additional feature for indoor applications to overcome drift problems at GPS-friendly locations in the building. GPS could also be used to situate a SLAM generated map on existing GPS based maps.

2.4 Ambient and other sensors

Unlike what you might expect, ambient sensors can provide valuable information for localization and mapping purposes. Clever use of ambient sensors is in some cases key to success. In this Section some interesting ambient and other remaining sensors are discussed.

2.4.1 Barometer

Some of the newer phones are equipped with a pressure sensor. The pressure sensor is basically a barometer that measures ambient pressure in units of Pascal. In [35], a pressure sensor is used successfully to detect floor changes independent of the environment (i.e. context-agnostic). Results show that their algorithm can detect any floor changes with any mode of transportation (elevator or stairs). Their algorithm is based on relative pressure differences and a user dynamics mode, which consists of walking, sitting, and moving up/down with an elevator or stairs. Relative pressure differences can of course equally well be used for mapping purposes.

Methods for mapping and localization in 2-dimensional environments are unlikely to benefit from a pressure sensor. But for modelling 3-dimensional environments it is definitely interesting to include pressure sensors. In fact, a pressure sensor together with the algorithm presented in [35], provide direct insight in one of the state variables.

The focus in this thesis is on 2-dimensional environments and the barometer is therefore not used.

2.4.2 Camera

Cameras are a common feature in modern phones and camera images can be used in a variety of solutions. In [11], targeted at mobile phones, a monocular-SLAM solution is used to estimate a local map and device's trajectory. A Parallel Tracking and Mapping (PTAM) algorithm is used to allow for asynchronous updates of camera pose and mapping optimization. However, by using a camera to solve SLAM alike problems a completely different approach is needed. By using a camera to solve the SLAM problem a couple of new challenges are introduced:

- Obviously visual based methods require the camera to record the environment at all times. In general people tend to wear their phone inside their pocket/purse which causes visual based methods to fail. Assumptions must be made on how the phone is retained for the success of visual based methods. For example, passive use of a localization and mapping algorithm (with the mobile phone located inside a pocket/purse) is not feasible.
- Camera images themselves are not very useful for localization. By using a camera for mapping and localization purposes there is the need of extracting features from the camera images and identifying a uniqueness property of landmarks. With WiFi signals this is not so much of an issue because each beacon simultaneously emits a unique identifier (BSSID). In a visual approach, feature extraction algorithms are needed to identify a landmark as such. In the full SLAM problem this is also

known as the "unknown correspondence problem" [33]. Images require careful processing and feature extraction before these features can be used in a localization framework. Image processing and feature extraction methods are known to be computationally heavy processes (i.e. methods such as SIFT [24], and FAST [30]). While mobile devices are getting increasingly powerful in terms of processing power, it may be a problem to perform heavy computation like image processing.

- Clearly, having the camera turned on at all times and processing images to extract features is extremely power consuming, which is rather inconvenient for battery powered devices.

2.4.3 Microphone

Even the phone's microphone can be used for localization purposes. An interesting technique is to create an 'acoustic fingerprint' of the environment. For instance Aloui et al. [2] have used a microphone in mobile devices for localization. However, their method assumes pre-installed static microphones and speakers. Furthermore this technique assumes a priori information of the environment such as the location of speakers. To our knowledge this method is never used in combination with SLAM-based techniques.

Fingerprinting is non-trivial to include in a SLAM-based method. Fingerprinting is ideally suited to describe a location, but it is unclear how this should aid SLAM-based methods. The microphone is therefore not considered as an input method for localization and mapping algorithms in this thesis.

2.5 Summary

Sensors that will be used for localization and mapping include all motion sensors (accelerometers, gyroscopes, and magnetic field sensors). The raw data of these three sensors are first processed by a step detection algorithm as described in Section 2.2.4 and an improved bearing sensor as described in Section 2.2.5. Furthermore, the WiFi sensor is used to correct the location prediction made by the motion sensors. A key notion for exploiting the WiFi sensor for mapping and localization methods is that of signal attenuation, which is modelled by the signal attenuation model (See Eq. 2.3.1) described in Section 2.3.1.

Finally, Table 2.1 shows an overview of all relevant Android sensors, units in which sensor values are measured, and for each motion sensor the coordinate system in which sensor values are measured.

Sensor	**Units**	**Coordinate system**
Motion sensors		
Accelerometer	m/s^2	Android
Linear accelerometer	m/s^2	Android
Gyroscope	rad/s	Android
Magnetic field	μT (micro-Tesla)	Earth
Orientation	$degrees$	Earth
Environmental sensors		
WiFi	dBm	-
Bluetooth	dBm	-
GSM	dBm	-
GPS	-	-
Ambient sensors		
Barometer	Pa	-
Camera	-	-
Microphone	-	-

Table 2.1: Overview of Android sensors, measurement units, and coordinate system in which sensor values are measured

Chapter 3

Mapping and Localization Methods

In this Chapter techniques for localization and mapping are presented. Both supervised and unsupervised methods are discussed. All supervised methods presented in this Chapter require some offline training phase and additional training time whereas unsupervised methods do not require this initial training phase.

The first part of this Chapter, Section 3.1, deals with localization methods based on motion sensors. In the second part of this Chapter, Section 3.2, localization methods based on radio signals (WiFi) are discussed. Methods presented in Section 3.2 are all supervised methods. Finally, in the last part of this Chapter, Section 3.3, hybrid methods are discussed in which both motion sensors and WiFi sensors are used. This last Section also covers unsupervised mapping, which is not considered in any other method presented in this Chapter.

3.1 Inertial navigation methods

An inertial navigation method is a localization method that relies on bearing and covered distance. The composite of these two enables reconstructing the user's walking path. Many versions of Inertial Navigation Systems (INS) exist and have applications in many fields, but most of these systems all include some form of Inertial Measurement Unit (IMU) to determine distance/speed and bearing [13]. In an ideal situation where the frequency is high enough and IMU sensor readings are freed from noise, path reconstruction resembles vector addition, in which vector magnitudes are given by the user's walking speed per time step, and the vector direction by the user's walking direction.

3.1.1 Pedestrian dead reckoning

Dead reckoning or Pedestrian Dead Reckoning (PDR) is a localization technique based on estimating a user's current location by projecting bearing and speed on a previously known location. It is a widely used positioning technique and has been extensively studied in the past. PDR methods are based either on shoe-mounted or waist-mounted measurement units. In Kourogi et al. [20] a PDR method is presented using waist-mounted inertial measurement units. Shoe-mounted approaches exploit so called zero velocity updates (ZUPTs) which occur when a shoe is touching the ground. Waist-mounted methods on the other hand are unable to exploit ZUPTs, but have the advantage of easier detection of body orientations and transitions such as from sitting down and standing up. In all experiments presented in this thesis, a mobile phone is used as if being an IMU. Motion sensors of the phone are used to detect steps, speed, and orientation. In case the user's mobile phone is located in a waist pocket the mechanics resemble those of a waist-mounted IMU, but compared to shoe-mounted and waist-mounted devices, it is much more difficult to detect steps from a hand-held device. The damping effect of a stretched out arm makes it even harder to detect ZUPTs. To overcome the damping effect, a step detection algorithm was presented for hand-held devices in Section 2.2.4, and in Section 2.2.5 an algorithm was presented for bearing detection using hand-held devices.

A well known problem of PDR is that errors accumulate over time. This problem is also known as drift. Some applications tend to overcome this problem by regularly calibrating the system, for instance by using GPS. At times when GPS signals are unavailable, the dead reckoning system takes control and predicts the user's location. At times the GPS signal is reliable enough, the user's location is reset again using the GPS device. Indoors, GPS is most often unavailable or unreliable and therefore other methods are needed for calibration. In the last part of this Chapter, hybrid models will be discussed to cancel out drift. PDR will then be used as input for these hybrid models. It is therefore interesting to investigate the boundaries of a mobile phone PDR approach before plugging the results into hybrid methods. Results of a mobile phone based PDR approach are presented in Chapter 4.

3.2 RF-localization methods

Localization techniques based on RF signals (e.g. WiFi) can coarsely be divided into two categories: either with or without signal propagation modelling. Propagation models aspire to relate signal strength and distance. Localization techniques based on signal modelling can be tricky because signal propagation is rather easily influenced. Obstacles of any kind will alter and reflect WiFi signals, which makes it difficult to device a general propagation formula. One attempt to such a signal propagation model was presented in Section 2.3.1. Localization methods that do not require signal modelling also exist. These models evaluate the signal locally such as in K-nearest neighbours (see Section 3.2.2). Since any error in the modelling technique propagates to the model based method, some have argued that locally evaluated methods will be guaranteed to always outperform modelled methods. An open challenge in both approaches (with or without signal modelling) is interpreting missing values. Modelling *not* observing an Access Point is not trivial as it might just be a glitch in the WiFi signal or it could indicate where the user is not located.

3.2.1 Fingerprinting

Fingerprinting is a positioning technique based on the idea that a location can be represented by some unique identifier. Theoretically any measure can be used for fingerprinting. In Section 2.4.3 the phone's microphone was already hinted as a feature for fingerprinting, and obviously WiFi signal strength or RSSI can be used likewise. Others have suggested exploiting local fluctuations in the magnetic field as fingerprinting features [36]. The process of fingerprinting is very similar to human fingerprinting, hence the name, and consists of two phases. The first (offline) phase is creating a lookup table of RSSI-measurements and locations. For every location, often called reference point (RP), multiple RSSI-measurements are collected. Multiple measurements are needed to cancel out noise due to dynamic signal characteristics and to create a reliable fingerprint of that location. In the second (online) phase, RSSI-measurements are classified using for instance a K-means classifier[1] or a least-square estimator.

The advantage of a fingerprinting approach is that it does not require signal modelling. Location descriptions are based on local information only, therefore making it unnecessary to model signal propagation as is done in eq. 2.3.1. The downside is that a large amount of training data is needed to create reliable cluster prototypes.

Fingerprinting is most often regarded as manually generating an RSSI-lookup table with a significant amount of work, while knowing that the list is useless after some time due to the dynamic characteristics of RF signals [27]. The dynamic characteristic of RF-signals calls for a different approach in the long run.

3.2.2 K-nearest neighbours

A K-nearest neighbours approach for indoor localization is probably the most straightforward method one can adopt. In environments where a dense amount of access points

[1]In K-means classification a data vector is assigned to the cluster with the smallest Euclidean distance. In Fingerprinting, clusters represent physical locations (i.e. the reference point locations). The resemblance with a K-means classifier is the usage of classifying a data vector (in signal space – measured in dBm) to a cluster prototype which represents a physical location by means of the smallest Euclidean distance (in signal space).

are installed, it can have surprisingly good results. The K-nearest neighbours method (for $K = 1$) is a localization approach that determines the user's location as being the location of the access point with the loudest dBm-values (i.e. the user's location is determined as the location of the nearest transmitter). K-nearest neighbours localization is unmistakably based on the idea of signal attenuation. The closer a phone is located to a transmitter, the better the signal strength and the more likely the phone is at the location near the transmitter.

The K-nearest neighbours approach is appealing because it does not require signal modelling. Other methods such as circular lateration (see Section 4.2.5) require signal modelling techniques to estimate the distance between access points and the device location. In a K-nearest neighbours approach, the device location is determined by selecting the K loudest transmitters (in signal space, dBm values) and averaging between the locations of these K transmitters (in physical space). The performance of this approach stands or falls with the density of access points. If only a limited amount of access points is available in a large area, it is unlikely this method will produce satisfactory results. But this also applies to other localization methods. More important is the value chosen for K. If K is chosen too small (e.g. $K = 1$) performance is directly related to the density of access points. On the other hand, if K is chosen too large in an environment where only a limited amount of access points is available, this approach will not have satisfactory results either. Note that if K is equal to the amount of available access points, the predicted device location, X, is the same in every prediction and can be written as

$$X = \frac{1}{N} \sum_{i=1}^{N} (x_i, y_i) \tag{3.2.1}$$

where N is the amount of available access points, and x_i and y_i refer to the x and y coordinates of the i^{th} landmark respectively. Hence, K needs to be chosen carefully. Optimal values for K and results of a K-nearest neighbours experiment are discussed in Chapter 4.

3.2.3 Circular lateration

Circular lateration is a deterministic localization method using multiple fixed reference points (i.e. access points) to determine the location of a mobile device. Circular lateration requires the location of the access points in Cartesian coordinates and distance measurements (RSSI measurements) from the mobile device to the fixed reference points in order to determine the location of the mobile device. A signal modelling technique such as presented in Section 2.3.1 is needed to convert RSSI values into distance measurements. Locating a mobile device in 2-dimensional space using circular lateration requires at least three reference points, and at least four reference points are needed to locate a mobile device in a 3-dimensional space. If access point locations are known, circular lateration can be regarded as a geometrical problem. The geometrical problem that needs to be solved is that of finding the intersection of multiple circumferences of circles which have a diameter proportional to the distance between transmitter (access point) and receiver (mobile device).

Recall that RF-signal propagation is not symmetrical in all directions (see Section 2.3.1), and therefore in practice the circumferences might not even overlap, or there might be a solution area rather than one single solution. There might be situations where some or all of the circles don't have overlapping regions, or there might be situations where one or more circles have overlapping regions. In the latter situation the solution would be an area rather than one location. In the former, no real solution is available. A least-squares method can be used to determine the location of the mobile device in case of (non-)overlapping regions. As stated in [21], the problem can be written in the form

$$HX = B, \tag{3.2.2}$$

where X is a vector with the desired device location, and is defined as

$$X = \begin{bmatrix} x & y \end{bmatrix}^T, \tag{3.2.3}$$

and the matrices H and B are defined as

$$H = \begin{bmatrix} x_2 - x_1 & y_2 - y_1 \\ \vdots & \vdots \\ x_n - x_1 & y_n - y_1 \end{bmatrix}, \tag{3.2.4}$$

$$B = \frac{1}{2}\begin{bmatrix} (r_1^2 - r_2^2) + (x_2^2 + y_2^2) - (x_1^2 + y_1^2) \\ \vdots \\ (r_1^2 - r_n^2) + (x_n^2 + y_n^2) - (x_1^2 + y_1^2) \end{bmatrix}, \tag{3.2.5}$$

where x_i and y_i represent the x and y coordinates of the i^{th} landmark respectively, and r_i represents the Euclidean distance to the i^{th} landmark. X can then be obtained as follows [21]:

$$X = (H^T H)^{-1} H^T B \tag{3.2.6}$$

The pitfall of using least-squares is that it is not robust against RSSI outliers and noise. Outliers can cause dramatic results because the method can not recognize outliers as such. Others have therefore proposed to use a probabilistic method instead or methods such as smallest polygon [27].

Circular lateration is expected to be superior to K-nearest neighbours, since circular lateration is able to locate the device in a continuous space, whereas a K-Nearest Neighbour is bounded to a finite number of access points. Also, if the model of signal attenuation is well defined, circular lateration might outperform fingerprinting methods since fingerprinting can only converge to a known 'fingerprinted' location, whereas with circular lateration, the actual location can be approximated more precisely by interpolating in between fingerprinted locations (i.e. circular lateration has a continuous scale of solutions whereas K-Nearest Neighbour and fingerprinting do not). On the other hand, since Circular lateration depends on RSSI modelling, the performance of Circular lateration is confined by the performance of the RSSI modelling technique. An overview of Circular lateration experiments and corresponding results are presented in Section 4.2.5.

3.3 Hybrid method (SLAM)

Localization methods presented so far rely on one modality at the time (i.e. either motion sensors of a mobile phone or dBm-values of WiFi access points) and require an offline training phase. In this Section, a method is presented to merge multiple sensor modalities into one localization and mapping method without a priori knowledge. The need for such an approach to localization is twofold: (1) In many applications the environment is unknown. No map data is available which requires map making at first. And (2) a priori maps are usually: costly to obtain, inaccurate, incomplete, and out of date.

A method used in robotics to overcome these problems, is Simultaneous Localization and Mapping (SLAM). As stated in [7], SLAM is a process by which a mobile robot can build a map of an environment and at the same time use this map to deduce its location. SLAM builds a map in which the location of landmarks (access points) and the mobile device are determined simultaneously. The simultaneous localization and mapping problem has no closed form solution and is known as a chicken-and-egg problem. In order to know either the robot location or the map, you need to know the other. In SLAM no prior knowledge of either the environment or the robot location is needed. Both are estimated online. The problem has been heavily studied in the past and nowadays many different approaches have been proposed. All solutions include a recursive method that consists of two stages, a prediction stage and a correction stage. In the prediction stage the robot pose is estimated based on motion information and information from the previous pose. In the correction stage, the predicted position of the robot and landmarks is corrected using measurements made by the robot. While moving, the robot can concurrently improve its own pose and landmark locations. This two-step recursive prediction correction method is the essence of the SLAM algorithm.

The system state in SLAM consists of pose information of the device, and all landmark locations. Besides the state vector, a covariance matrix is used to maintain the probability of each of these system state properties. During a time update the new system state is predicted based on the previous system state and a motion model. Existing SLAM methods often use robot odometry for the motion model, see for instance [5]. In contrast to existing methods, odometry is not available when using a mobile phone for the motion model. Instead, measurements from the phone's motion sensors are applied to the previous state to predict a new system state. Below it will become clear how exactly the motion sensors are used to create this prediction. In the correction step the predicted system state is corrected with device measurements. Existing methods have used a wide variety of measurement types for the correction step, for instance laser range data [5] or camera images [6]. In case of a mobile phone, measurements from the WiFi sensor can be used for correcting the predicted state. Below it will become clear how exactly WiFi measurements are used to correct the predicted system state.

Note that the problem presented here greatly resembles a 2-dimensional Range-Only SLAM problem (RO-SLAM) [1]. Despite the important applications of RO-SLAM, relatively few research has been addressed to the SLAM problem based on range-only sensors. The key aspects that make RO-SLAM an extremely challenging problem are related to the measurements. First of all, there is a high number of outliers in measurements due to the nature of the sensor (RF sensors are known to have a high number of outliers, as will become clear in Section 4.2.1). Secondly, there is a high ambiguity in the measurements (only distance information is available from the measurement sensor, no bearing information is available at all, meaning measurements have a wide range of conflicting solutions) [3].

A popular and commonly used solution to the SLAM problem is by the use of an extended Kalman filter [7]. Theory and principles of the extended Kalman filter (EKF) are a key aspect of the EKF SLAM algorithm and are therefore discussed below in detail before diving into details of the EKF SLAM algorithm.

3.3.1 Terminology

Commonly used terminology for the (extended) Kalman filter and EKF SLAM is listed below. Note that state representations at a discrete time step k are written as q_k. In literature x_t is often used instead of q_k to denote state representations. We prefer using the latter notion because the use of x_t might be confusing in the discussion of locations.

- q_k, system state at time k. Here, $q_k \in \mathbb{R}^3$, with x_k, y_k, and θ_k to describe the position and orientation of the robot/measuring device.
- P_k, system covariance matrix. A key component of the EKF SLAM framework and can intuitively be considered as representing the uncertainty about device pose and orientation and landmark locations. As the values in this matrix approach zero, the framework converges to a solution. On the other hand, if this matrix still contains larger values, there remains larger uncertainty.

- z_k, complete set of (landmark) observations at time k. For instance, z_1 refers to the set of observations at time $k = 1$ and contains distance observations of n landmarks. Note that no assumptions are made on the landmark locations. z_k is just a vector with noisy observations at time k.
- u_k, the control vector with actions the robot has to perform. This vector is not always directly observable.
- A_k, $n \times n$ matrix that relates the state at the previous time step, $k - 1$, with the state at the current time step, k. n represents the number of dimensions needed to describe the system state. For localization systems, commonly $n \in \mathbb{R}^3$ so it can describe position and orientation. If state transitions remain constant over time, the subscript k may be left out.
- B_k, $n \times l$ matrix that relates the optional control input to the state q, with l being the number of control inputs. In real life situations B_k and u_k are mostly unknown (we don't know the intentions of the person operating a mobile phone - it would be an interesting factor to take into account though).
- k, discretization factor of time.
- K_k, vector called the Kalman Gain. The Kalman Gain is used to 'weight' residual values. The simplest way to understand the effect of the Kalman Gain is to think of it as a weighting function. If the error covariance matrix R approaches zero it means that the measurements are more reliable and get more weight. If, on the other hand, the estimated error covariance is large, the measurements are unreliable and more weight is given to the predicted state. The effect of the Kalman Gain will become more clear in the following Sections.
- w_k, process noise vector. Subscript can be left out if measurement noise remains constant over time. Usually this is just a fixed number and subscript is left out.
- v_k, measurement noise vector. Subscript can be left out if measurement noise remains constant over time. Usually this is just a fixed number and subscript is left out.
- m_i, a vector of size 2 (with x and y coordinates), describing the i^{th} landmark. m (without subscript), is a matrix which describes the complete set of all landmarks. m refers to the set of landmarks that have been initialized and are added to the EKF state vector and covariance matrix as landmarks. This differs from the set of observations of landmarks required by the landmark initialization techniques.

3.3.2 Kalman filter and extended Kalman filter

The Kalman filter [14] is one of the first successful implementations of a Bayesian filter and is used as a state estimation method. A Kalman filter alternates between a *prediction* and a *correction* step (commonly known as a predictor-corrector structure). In the prediction step, using the previous system state q_{k-1} and a control input u_k, a prediction $\bar{q}_k$ is made for the next state. In the correction step, the predicted system state $\hat{q}_k$ is corrected using the measurement z_k to form the new state q_k. By alternating between the prediction and correction step, an accurate state estimation is achieved.

3.3.2.1 Kalman filter

The idea behind the Kalman filter is to represent an estimation problem using a predictor-corrector structure. In the time update (prediction) the next system state is estimated based on the previous known system state and state transition properties. In the measurement update (correction) the predicted values are corrected using weighted observations. The state equation, q_k, and measurement equation, z_k, form the basis of the Kalman filter theory. The state equation q_k is modeled as

$$q_k = Aq_{k-1} + Bu_{k-1} + w_{k-1} \tag{3.3.1}$$

where w_{k-1} is a zero-mean Gaussian noise parameter that is modeled as $P(w) = \mathcal{N}(0, Q)$. A is a matrix that relates the state at time $k-1$ to the state at time k. A might change over time, but most often it remains constant. The matrix B relates the control input u (e.g. odometry) at time $k-1$ to the system state at time k. Measurements z_k are modeled as

$$z_k = Hq_k + v_k \tag{3.3.2}$$

where v_k is a zero-mean Gaussian noise parameter and is modeled as $P(v) = \mathcal{N}(0, R)$. These equations form the basis of the Kalman filter algorithm. Now let's use these equations to make a prediction on the next state estimate $\hat{q}_k$

$$\hat{q}_k = Aq_{k-1} + Bu_{k-1} \tag{3.3.3}$$

And predicted error covariance as

$$\hat{P}_k = AP_{k-1}A^T + Q \tag{3.3.4}$$

In the measurement update, the Kalman Gain, K_k, is computed to 'weight' the measurements z_k by multiplying the residual values $(z_k - H\hat{q}_k)$ with the Kalman Gain K_k. The Kalman Gain is computed as

$$K_k = \hat{P}_k H^T (H\hat{P}_k H^T + R)^{-1} \tag{3.3.5}$$

Having computed the Kalman Gain, the next state probability is corrected using the residual values and the Kalman Gain as

$$\hat{q}_k = \hat{q}_k + K_k(z_k - H\hat{q}_k) \tag{3.3.6}$$

And the new error covariance matrix is computed as

$$P_k = (I - K_k H)\hat{P}_k \tag{3.3.7}$$

Kalman Gain

A little word on the Kalman Gain. The Gain is important because it 'weights' the predicted value $\hat{q}_k$ and measurements z_k as can be seen in equation 3.3.6. This can be made clear if the Kalman Gain equation is rewritten in the following form:

$$K_k = \frac{\hat{P}_k H^T}{H\hat{P}_k H^T + R}. \tag{3.3.8}$$

From this notation it is easy to see that as P_k goes to zero (the predicted error covariance goes to zero), the Kalman Gain also approaches zero and all 'weight' is given to the predicted value q_k (See eq. 3.3.6). This seems intuitive because if the predicted values have little error, they can be trusted more and most of the weight is given to the predicted values. On the other hand, if the measurement error covariance matrix R approaches zero, the Kalman Gain approaches H^{-1}. This seems intuitive because if the measurement error is low, meaning the measurements are more reliable, more weight is given to the measurements z_k.

An overview of the Kalman filter algorithm is given in Algorithm box 2. For a more detailed derivation of the Kalman filter equations see [37, 33].

Algorithm 2 KALMAN FILTER

```
1: $\hat{q}_k = Aq_{k-1} + Bu_{k-1}$
2: $\hat{P}_k = AP_{k-1}A^T + Q$
3: $K_k = \hat{P}_k H_k^T (H_k \hat{P}_k H_k^T + R)^{-1}$
4: $q_k = \hat{q}_k + K_k(z_k - H_k\hat{q}_k)$
5: $P_k = (I - K_k H_k)\hat{P}_k$
6: return $q_k, P_k$
```

The parameters that enter the KF-algorithm include the system state q_{k-1} and corresponding covariance matrix P_{k-1}, and noise parameters R and Q. These noise parameters can be estimated each iteration of the KF, but can also be fixed parameters. In the case of SLAM using a mobile phone, the noise matrices are kept fixed. At line 1 and line 2, the prediction $\hat{q}_k$ and covariance $\hat{P}_k$ are made based on the previous system state q_{k-1} and motion input u_{k-1}. At line 3 the Kalman Gain is computed. Finally, on line 4 and line 5 the new system state q_k and new covariance matrix P_k are computed using the previous state and Kalman Gain.

3.3.2.2 Extended Kalman filter

The main difference between a Kalman filter and an extended Kalman filter is to replace the prediction and correction function with non-linear equations $f(\cdot)$ and $h(\cdot)$, where the non-linear function $f(\cdot)$ models the prediction model and the non-linear function $h(\cdot)$ models the measurement model [33]. See Table 3.1 for an overview.

	Kalman filter	Extended Kalman filter
state prediction	$Aq_{k-1} + Bu_{k-1}$	$f(q_{k-1}, u_{k-1})$
measurement prediction	$H\hat{q}_k$	$h(\hat{q}_k)$

TABLE 3.1: Key differences between the Kalman filter and the extended Kalman filter.

By substituting the EKF equations into the Kalman filter algorithm, the new system state equation, q_k, becomes

$$q_k = f(q_{k-1}, u_{k-1}) + w_k \tag{3.3.9}$$

and the measurement model z_k becomes

$$z_k = h(q_k) + v_k \tag{3.3.10}$$

This set of non-linear equations forms the essence of the extended Kalman filter. In the time update (prediction) step, the predicted system state, $\hat{q}_k$, is given as

$$\hat{q}_k = f(q_{k-1}, u_{k-1}) \tag{3.3.11}$$

and associated error covariance matrix $\hat{P}_k$ becomes

$$\hat{P}_k = AP_{k-1}A^T + W_k Q_{k-1} W_k^T \tag{3.3.12}$$

In the measurement update again the Kalman Gain is computed first after which the residual values are weighted using this Kalman Gain. The Gain is computed as

$$K_k = \hat{P}_k H_k^T (H\hat{P}_k H^T + V_k R_k V_k^T)^{-1} \tag{3.3.13}$$

where V_k is the Jacobian matrix that relates to the measurement noise, and H_k is the Jacobian of $h(\hat{q}_k)$. Next the predicted system state is corrected using the Kalman Gain K_k and residual values $(z_k - h(\hat{q}_k))$ as

$$q_k = \hat{q}_k + K_k(z_k - h(\hat{q}_k)) \tag{3.3.14}$$

And finally the new error covariance matrix P_k is computed as

$$P_k = (I - K_k H_k)\hat{P}_k \tag{3.3.15}$$

The extended Kalman filter algorithm is described below in Algorithm box 3. For a detailed derivation of the extended Kalman filter equations, see [37, 33].

Algorithm 3 Extended Kalman filter

```
1: $\hat{q}_k = f(q_{k-1}, u_{k-1})$
2: $\hat{P}_k = AP_{k-1}A^T + Q$
3: $K_k = \hat{P}_k H_k^T (H_k \hat{P}_k H_k^T + R)^{-1}$
4: $q_k = \hat{q}_k + K_k(z_k - h(\hat{q}_k))$
5: $P_k = (I - K_k H_k)\hat{P}_k$
6: return $q_k, P_k$
```

The EKF algorithm will form the basis of the EKF SLAM solution for the problem stated earlier. The EKF filter needs to be modified a little before it can be used in SLAM. In the next Section the details of EKF SLAM will be discussed.

3.3.3 EKF SLAM

In EKF SLAM, the EKF state vector and covariance matrix need to be extended with landmark locations. That is, landmarks are included in the state vector and covariance matrix. The size of this *combined state vector* then becomes $2N + 3$, where N is the number of landmarks. The size of the covariance matrix is a square matrix of similar dimensions (e.g. $\mathbb{R}^{(2N+3)\times(2N+3)}$) . If a 3D-model is used, the device location consists of three parameters to describe the location x, y, z, and one, θ, to describe the orientation. For landmarks this becomes x, y, z to describe the locations. Hence, with a 3D-model and N landmarks, the combined state vector becomes size $3N + 4$. But for now, a $2D$ model is assumed.

Analogous to the EKF algorithm, EKF SLAM consists of a *motion model* and an *observation model* [7]:

- The motion model describes the probability of a state change given the previous state and a control input (e.g. motion sensors of the mobile phone). Using the estimated device location at a previous time step, q_{k-1}, and a control input, u_k, the new robot position can be described with probability $P(q_k|q_{k-1}, u_k)$. State transitions are assumed to be Markovian. This means the current state only depends on the directly preceding state.

- The observation model describes the probability of making an observation given the robot and landmark locations. Using the estimated device location q_k and landmark locations m, the observation model describes the probability of making an observation z_k, and can be described by the probabilistic form $P(z_k|q_k, m)$.

The use of an extended Kalman filter (EKF) is a commonly used solution to the SLAM problem. Here the system state is described with a vector q_k, at discrete time steps

k, where $q_k \in \mathbb{R}^3$ with x_k, y_k, and θ_k for position and orientation. Orientation, θ_k, is determined using a sensor fusion algorithm (as described in Section 2.2.5) which relies on sensory input from gyroscope, magnetic compass and accelerometer. Further details on individual sensors are described in Chapter 2. Speed (and distance) is determined using a step counting algorithm described in Section 2.2.4.

By using the stepcounter evaluated distance ΔD as

$$\Delta D = numSteps \cdot stepSize \tag{3.3.16}$$

and using a sensor fusion algorithm for an indication of heading, the EKF equations can be written as follows. The EKF state equation becomes

$$f(q_{k-1}, u_{k-1}) = \begin{bmatrix} x_{k-1} + \Delta D_{k-1}\cos(\theta_{k-1}) \\ y_{k-1} + \Delta D_{k-1}\sin(\theta_{k-1}) \\ \theta_{k-1} \end{bmatrix} \tag{3.3.17}$$

with ΔD_{k-1} the traveled distance between the previous system state q_{k-1} and current system state q_k, and w_k the system process noise. The matrix $A_k \in \mathbb{R}^{3x3}$ relates the state q_{k-1} from the previous time step to the current state q_k and can be written as

$$A_k = \frac{\partial f}{\partial q_{k-1}} = \begin{bmatrix} 1 & 0 & -\Delta D_{k-1}\sin(\theta_{k-1}) \\ 0 & 1 & \Delta D_{k-1}\cos(\theta_{k-1}) \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3.18}$$

The measurement model (correction) uses the signal attenuation model (Eq. 2.3.1) and RSSI-values as predicted landmark distances. This is the non-linear Euclidean distance function $d_{q_k}^{\lambda}$ as described in Eq. 3.3.20. The Jacobian H_k of $d_{q_k}^{\lambda}$ is given by

$$\begin{aligned} H_k = \frac{\partial h}{\partial q_k} &= \begin{bmatrix} \frac{x_k - x_\lambda}{\sqrt{(x_k-x_\lambda)^2+(y_k-y_\lambda)^2+\eta_s}} & \frac{y_k - y_\lambda}{\sqrt{(x_k-x_\lambda)^2+(y_k-y_\lambda)^2+\eta_s}} & 0 \end{bmatrix} \\ &= \frac{1}{d_{q_k}^{\lambda}} \begin{bmatrix} x_k - x_\lambda \\ y_k - y_\lambda \\ 0 \end{bmatrix}^T \end{aligned} \tag{3.3.19}$$

3.3.4 Landmark initialization

New landmarks need to be added to the state vector and covariance matrix of the EKF SLAM framework. There are basically two approaches to add new landmarks: either online (without delay) or with some delay for initialization. The distinction is important because it can have considerable consequences on the performance of the SLAM method. The advantage of undelayed initialization of landmarks obviously is that the position estimation of the device can immediately benefit from this landmark. However, this is only possible if the undelayed initialization makes any sense. In case the landmark location uncertainty is too large, it might be better to use the delayed initialization method.

A welcome side effect of using APs as landmarks is that they are easy to distinguish using their BSSID (i.e. MAC-address). Every access point has its own MAC-address which gives it a unique identifier. Many SLAM implementations face the problem of unknown correspondences which is the problem of recognizing landmarks by a unique identifier. This mainly occurs when landmarks have overlapping features. For instance, a yellowish cone in the left corner of a robot soccer field looks very much like a yellowish cone on the right corner of a robot soccer field. Additional information is needed to distinguish between those two landmarks. But fortunately access points are easily distinguishable using their BSSID. This boils down to using an EKF SLAM framework with known correspondences [33].

Once access point locations are known, the distance at time k from device location (x_k, y_k) to landmark location (x_λ, y_λ), denoted by $d^\lambda_{q_k}$, is given by:

$$d^\lambda_{q_k} = \sqrt{(x_k - x_\lambda)^2 + (y_k - y_\lambda)^2} + \eta_s \tag{3.3.20}$$

where λ refers to the i^{th} landmark, and η_s is a zero-mean Gaussian noise with variance σ_s^2. But before landmarks can enter the EKF SLAM framework, an initial location estimate is needed as well as an estimate on the precision of this estimate. Landmark locations can be estimated using a technique called trilateration. Trilateration is a technique to determine a location of an access point by finding the intersection of circumferences from reference points with a radius as large as the distance to the AP (see Figure 3.1). Reference points in this context refer to locations on a travelled path where measurements were taken. At least three reference points that *do not lie on a straight line* are needed to unequivocally determine the position of the AP, but it is best to use more observations to cancel out measurement errors. Three (or more) reference points that lie on a straight line will yield ambiguous results with two optional landmark positions, one on either side of the line. For instance, if there are observations from P_1 and P_2 there are two remaining solutions, one marked by the red star on the lower left, and the other marked by the black star. A third observation P_3 is needed to pinpoint the AP location.

Landmark initialization is even more unreliable if it is the very first observed landmark. This means that no landmarks have entered the SLAM EKF framework and the measurement update from the SLAM method can't be used. Feedback (correction) from the landmark measurements is not yet available. This also implies that reference point locations are not very reliable because their locations are derived only from motion sensor measurements. In [8] it is shown that trilateration with noisy measurements falls in the NP-complete category of problems. Another method for landmark initialization is discussed below.

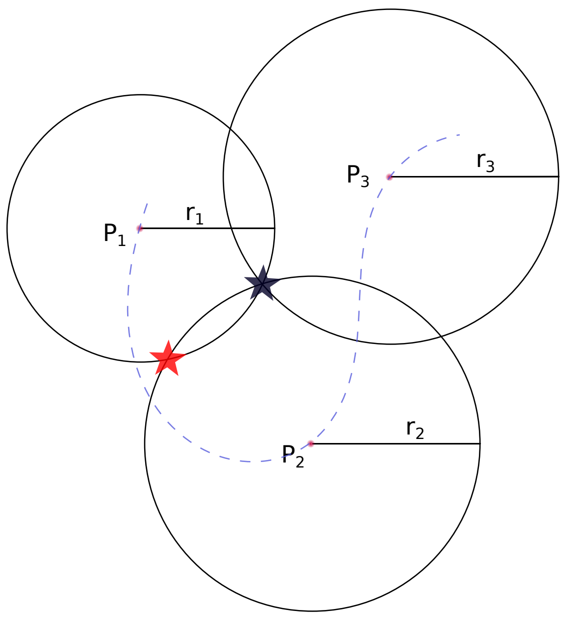


FIGURE 3.1: *Trilateration.* The black star represents an AP-location. The dashed blue line is an example path with reference points at P_1, P_2, and P_3 with respectively radius r_1, r_2, and r_3. Starting near P_1, as we reach P_2, there are two possible solutions remaining, one marked by the red star on the left, the other marked by the black star in the middle. After taking into account reference point P_3, only one final solution remains.

3.3.4.1 Gauss-Newton method

The problem of finding the intersection of circumferences can be written as a non-linear least squares problem and can be solved using the Gauss-Newton method. This method has the advantage to always converge, however it is not guaranteed to converge to the global minimum. Non-linear least squares methods solve the problem of minimizing $\|Ax-b\|^2$ by finding a vector $\hat{x}$ such that $\|A\hat{x}-b\|^2 \leq \|Ax-b\|^2$ for all x. This problem has a known solution at $\hat{x} = (A^TA)^{-1}A^Tb$. The objective function to minimize is defined in terms of residual distances. Basically we want to find an $\hat{x}$ such that the residual distances to all reference points is lowest. This function is of the form

$$g(x) = \sum_{i=1}^{m} d_i(x)^2 \tag{3.3.21}$$

where $d_i(x)$ is the non-linear Euclidean distance function between the landmark (access point) location and the i^{th} reference point, and is given by

$$d_i(x) = \sqrt{(AP_x - P_{x,i})^2 + (AP_y - P_{y,i})^2} + \rho_i \tag{3.3.22}$$

AP_x and AP_y refer to the location of the access point we are trying to locate, $P_{x,i}$ and $P_{y,i}$ refer to the i^{th} reference point locations, and ρ_i is the noise related to the measurements for the i^{th} reference point.

This model is tested on convergence rate and accuracy. Figure 3.2 shows an example Gauss-Newton trace. The dashed blue line is an example walking path, the green circles represent noisy distance measurements at several reference points and the red graph is a Gauss-Newton trace. A pilot experiment showed that after approximately 6 iterations the method has sufficiently converged (see Figure 3.3). Furthermore, it was found that if the residual training distance is below a certain threshold, the landmark location can be accepted as the true landmark location. If the residual distance measurement is higher than this threshold value, it is almost certain that the method is stuck in a local minimum and there exists another solution with the same (or higher) probability of being the actual access point location. It is therefore wise to re-run the Gauss-Newton method with a different initialization and converge until a better estimate is available. In a second experiment the Gauss-Newton method is performed multiple times with a different initialization. The accepted final solution is the result with the lowest residual distance.

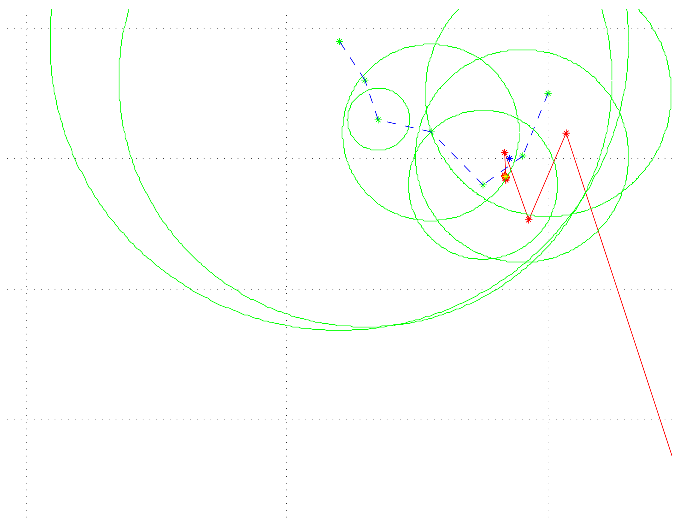


FIGURE 3.2: Visualization of Gauss-Newton convergence. The red graph indicates iterations of the algorithm, the blue line is a simulated walking route. Green asterisks indicates locations at which WiFi measurements were taken along the walking route. The green circles indicate possible AP-locations based on RSSI measurements at the green-asterisk locations. The signal attenuation model is used to translate RSSI values to distances (see Eq. 2.3.1). The yellow asterisk indicates the true AP-location.

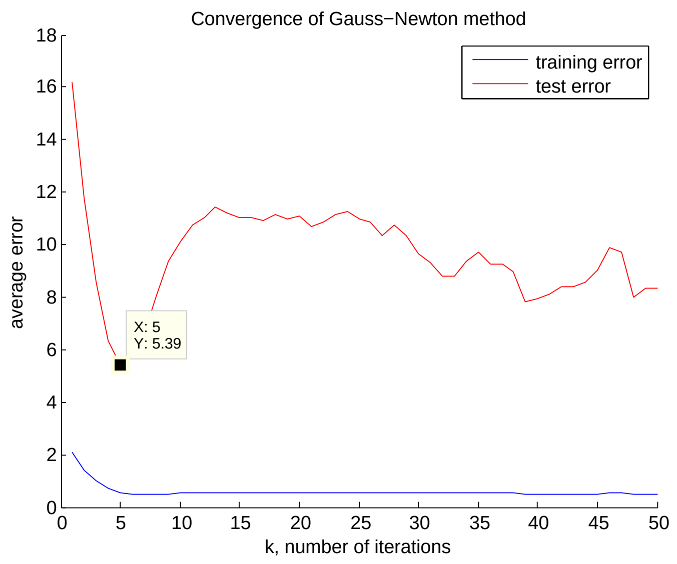


FIGURE 3.3: *Gauss-Newton convergence.* Training and test error of Gauss-Newton method

3.4 Multi-user EKF SLAM

The state vector, q_k, and noise matrix, P_k, contain valuable information for future users. They contain information about the relative landmark (access point) locations and related uncertainties. Only the first three elements of the state vector contain information about the device relative to the map. The rest of the state vector is the map itself. This also holds for the noise matrix: the first three rows and first three columns store information about uncertainty between the device and landmarks (and vice versa). All remaining parts of the noise matrix contain map data. In a follow up study it would be interesting to evaluate the effect of a so called 'multi-user'-approach where the map-related parts of the state vector and noise matrix are reused.

In Section 4.3 results of a multi-user EKF SLAM experiment are presented in which the state vector and noise matrix are reused. In these experiments results of dead reckoning and EKF SLAM will we compared to multi-user EKF SLAM. Theoretically the best performing method should be the multi-user variant of EKF SLAM, followed by EKF SLAM and dead reckoning. The advantage of multi-user EKF SLAM over the other two methods is that it can benefit from earlier determined landmark locations to overcome drift errors. Furthermore, the multi-user variant is expected to outperform the single-user method because the single-user variant needs to initialize landmarks before they can be exploited to reduce drift errors whereas the multi-user variant can reuse map data (landmark locations) from earlier experiments. Dead reckoning does not take advantage of WiFi measurement for localization correction and is therefore expected to perform worst. The assumption made is that by integrating motion sensor measurements and WiFi measurements, the localization prediction becomes more accurate.

Chapter 4

Results and Discussion

In this Chapter the performance of localization and mapping algorithms is presented. This Chapter is divided into three parts. In the first Section (see Section 4.1), all results related to motion-based methods are given and discussed. A noise analysis of motion sensors is made and the performance of localization algorithms based on motion sensors are discussed. Performance of the step counter and bearing sensors are also presented in this Section. In the second Section (see Section 4.2), all results related to WiFi sensors are presented. In the last Section (see Section 4.3) the performance of (multi-user) EKF SLAM is discussed, which combines measurements from motion and WiFi sensors. In Section 4.4 the results presented in this Chapter are briefly summarized.

4.1 Motion-based methods

In this Section results of methods based on the mobile phone's motion sensors are presented. Datasets used in this Section are the motion noise dataset (see Appendix A.2) for evaluating the precision of motion sensors, and the user tracking datasets (see Appendix A.4) which are used to assess the bearing sensor and dead reckoning.

4.1.1 Noise analysis

All localization and mapping algorithms presented in this thesis receive input from mobile phone sensors and are confined by the precision of these sensors. Therefore a noise analysis experiment of motion sensors is discussed below. The data for this experiment is obtained from a (1) Samsung Galaxy SII, which is the phone model that was used most frequently for data gathering during this thesis. Other phone models used to gather data are a (2) HTC One, and a (3) Samsung Galaxy Ace.

Data in this noise analysis experiment was gathered with an Android application built specifically for this task (see Appendix B.2). The application measures sensor values of three different sensors (gyroscope, accelerometer and magnetic field sensors) every $50ms$ for exactly ten minutes. During the experiment the phone was placed stationary on a table with the screen facing upwards. A delay of 5 seconds was introduced to prevent recording noise generated by touching the phone to start the experiment. See Appendix A.2 for details about the dataset. Results are presented in Table 4.1.

The most interesting results are those from the gyroscope and the linear accelerometer. Measurements obtained from the accelerometer and compass are more difficult to interpret. Results from the accelerometer should resemble those of the linear accelerometer, the only difference between those sensors is that the gravity component included in the accelerometer makes it harder to interpret the results. The same holds for the magnetic field sensor. Measurements of this sensor directly relate to the orientation of the phone during the experiment.

SENSOR	MEAN			SD		
	x	y	z	x	y	z
Accelerometer (m/s^2)	0.0283	-0.1209	10.1209	0.0153	0.0231	0.0408
Linear accelerometer (m/s^2)	-0.0000	0.0010	0.0022	0.0116	0.0182	0.0327
Gyroscope (rad/s)	0.0098	-0.0088	0.0091	0.0027	0.0023	0.0028
Magnetic field (μT)	-33.4681	27.2340	-59.7049	0.8047	0.3012	0.5255

TABLE 4.1: Noise analysis of motion and orientation sensors.

Table 4.1 shows that when the phone is placed stationary on a table, motion sensors are quite accurate. The only sensor that has slightly off SD-values is the orientation sensor. But there is a simple explanation for this effect. Accelerometer and gyroscope don't receive input when the phone is stationary on the table, but the magnetic field detected by the compass is much harder to divert or shut down. Magnetic field sensor measurements presented here might as well indicate fluctuations in the magnetic field instead of sensor precision. It might as well indicate fluctuations due to changes in Earth's magnetic field or external sources such as loudspeakers, as was discussed in Section 2.2.3.

4.1.2 Step counter

Other studies have reported performance of stepcounting algorithms as a function of walking pace (BPM) (see for instance [26]). In a simple step counting experiment the results of the step counting algorithm was evaluated in the same manner. Figure 4.1 displays the performance of the step counting algorithm by comparing actual steps to predicted steps. Each asterisk indicates the difference between actual and predicted steps for a single experiment. The closer to the diagonal the better. For optimal performance, where the amount of predicted steps is equal to the amount of actual steps, all results would lie on the diagonal for all experiments.

Very few studies report on algorithm performance of rejecting false positives (e.g. fake steps). Most often it is very well possible to trick step counting algorithms into detecting fake steps by making 'step motions' manually by shaking the phone in a walking alike motion. A clever method to overcome these tricks is by including geolocation measurements such as WiFi or GPS (if available) to filter such fake motions. On the other hand, the latter approach would also filter steps made on a treadmill. In the approach presented here, the only limitation on faking steps is based on human physiological constraints. A walking pace of $> 20 steps/s$ would for instance be impossible. In [26] a step counting algorithm is proposed and the performance of commercially available pedometers is reported. Comparing our results to those presented in [26], the performance of the step counter algorithm presented in this thesis outperforms commercially available methods and resembles performance reported in [26].

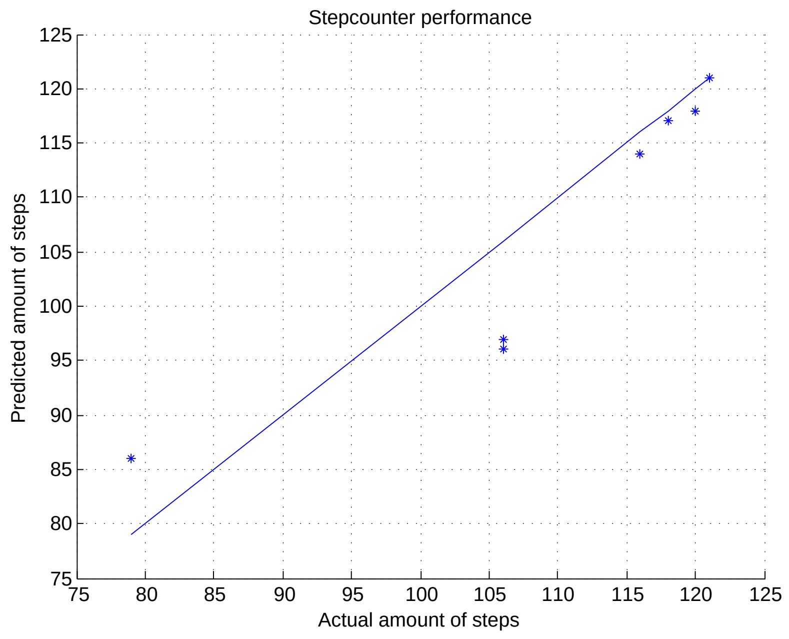


FIGURE 4.1: Step counter performance. The asterisks indicate results of individual experiments. In case of optimal performance, all results would lie on the diagonal line.

4.1.3 Bearing sensor and dead reckoning

In this Section the performance of a 2-stage EKF sensor fusion algorithm (see Section 2.2.5) is compared to the default Android orientation sensor by means of a dead reckoning experiment. The 2-stage EKF sensor fusion algorithm was developed to improve the default Android orientation sensor by incorporating measurements from the phone's gyroscope. This algorithm relies on measurements from three motion sensors: (linear) accelerometer, gyroscope, and orientation sensor. The only requirement for this algorithm to improve the Android orientation sensor is the presence of a gyroscope in the mobile phone. Some mobile phone manufacturers choose not to equip their phone models with a gyroscope which makes this algorithm useless. If the gyroscope is absent, the performance of this 2-stage EKF sensor fusion algorithm is comparable to the performance of the default orientation sensor in Android. Data for this experiment was gathered using the Tracking application (see Appendix B.4 for details).

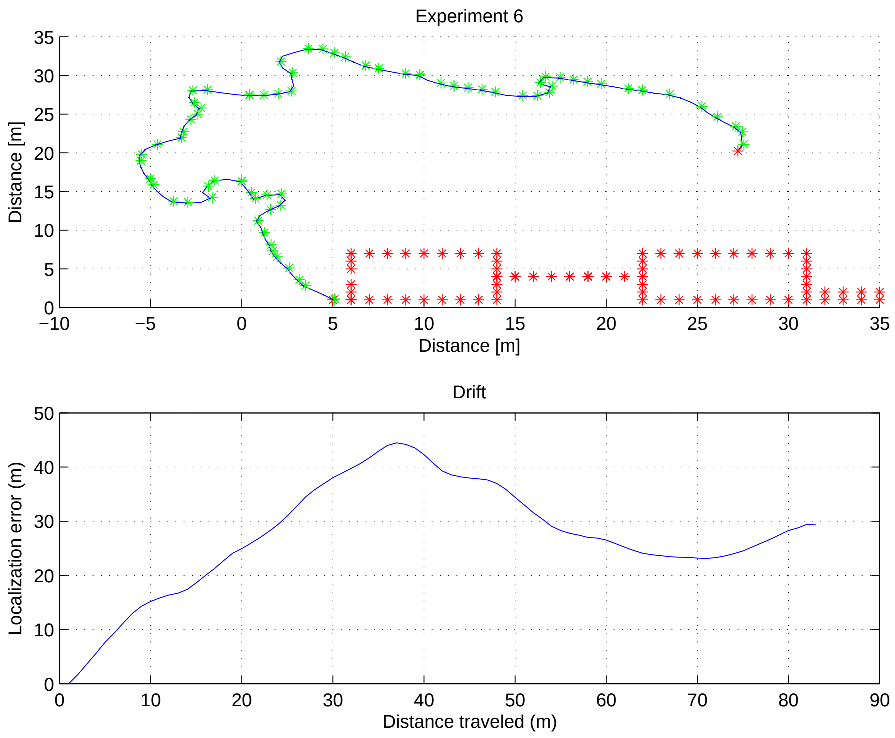


FIGURE 4.2: Dead reckoning experiment based on the default Android bearing sensor. The red marks in the top Figure indicate actual locations, green marks indicate estimated locations. The blue line on the top graph depicts the reconstructed path. The blue line in the bottom Figure depicts the error of the estimated path (m). Participants were instructed to walk the route counter clockwise visiting all red marks, starting at $(5, 1)$.

The result of a dead reckoning experiment based on the default Android orientation sensor is presented in Figure 4.2. The result of the same experiment, but this time with the bearing determined by the 2-stage EKF algorithm, is presented in Figure 4.3. In both Figures, the top graphs display the reconstructed path. The red marks indicate actual locations, the green marks indicate locations determined by dead reckoning, and the blue line depicts the reconstructed path of the user. In both Figures, the bottom graphs shows the error based on the Euclidean distance between the green marks and corresponding red marks (i.e. localization error).

Consider the results of the Android orientation sensor presented in Figure 4.2. In short, the Android orientation sensor has really poor performance for this environment. The

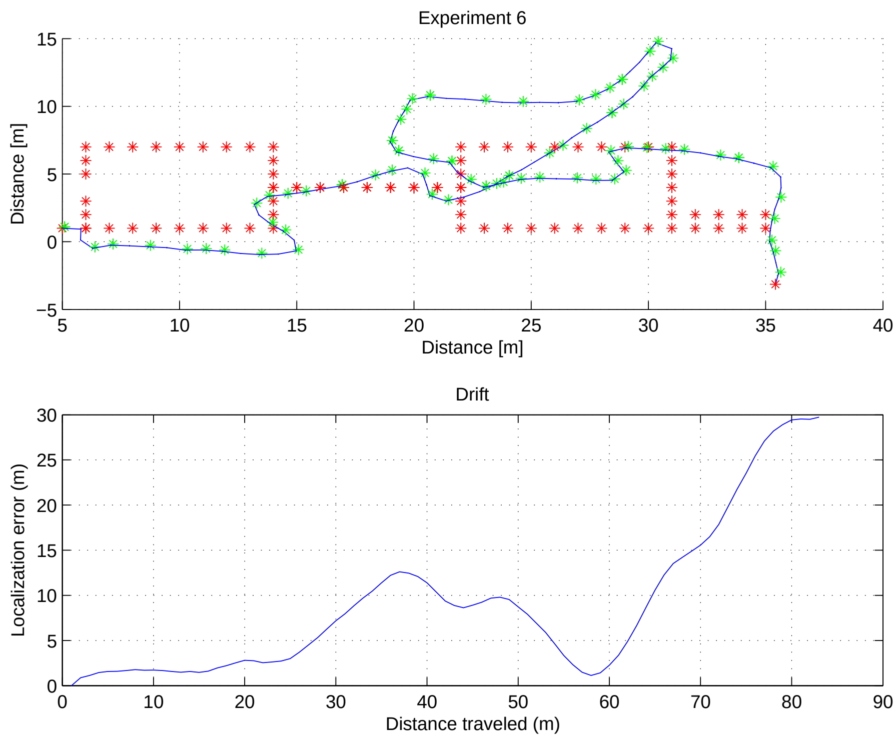


FIGURE 4.3: Dead reckoning experiment based on a 2-stage EKF sensor fusion algorithm. The red marks in the top Figure indicate actual locations, green marks indicate estimated locations. The blue line on the top graph depicts the reconstructed path. The blue line in the bottom Figure depicts the error of the estimated path (m). Participants were instructed to walk the route counter clockwise visiting all red marks, starting at $(5, 1)$.

orientation at the start of the experiment is already off, and it continues to fail throughout the experiment. Rotations are not detected correctly and overall localization error increases to approximately $30 - 40m$. In multiple pilot experiments it was discovered that orientation sensors of all tested phone models and brands failed in this environment. Even a field compass failed to indicate true magnetic North consistently. The foremost cause of these failures is due to building structure. Walls and support pillars are made of reinforced concrete which cause orientation sensors to digress into wrong predictions. This is most likely to occur in other environments also. Therefore, every localization method based on orientation sensors needs to take into account such possible environments and requires additional techniques to overcome these deficiencies.

Now let's consider the results of the same experiment based on the 2-stage EKF filter. Interesting to note is the development of drift in the 2-stage EKF algorithm displayed in Figure 4.3. The first approximate $60m$ of the reconstructed path is dominated by a steady increase in drift (the path is deviating more and more towards the north). After a distance of almost $60m$ the predicted bearing is off by a full 180 degrees. This steady increase in drift is most likely due to the local disturbances of the magnetic field and the correction stage of the algorithm which uses the Android orientation sensor for corrections. Quick changes in bearing are determined by the phone's gyroscope which causes the performance increase compared to the default Android orientation algorithm. But on the long term the Android orientation sensor is used to align the sensor with the true magnetic North, which causes it to fail on the long run.

By comparing the results of dead reckoning based on the default Android sensor and the 2-stage EKF algorithm, it becomes immediately clear that, although the performance is quite poor in general, the 2-stage EKF algorithm outperforms the Android orientation sensor by far.

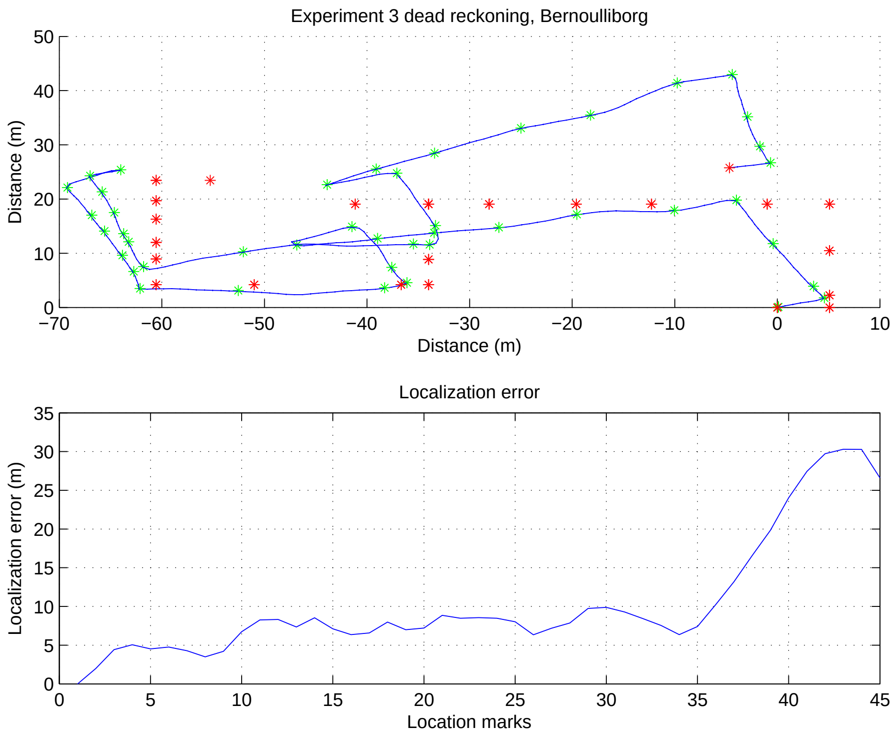


FIGURE 4.4: Dead reckoning experiment based on a 2-stage EKF sensor fusion algorithm. The red marks in the top Figure indicate actual locations, green marks indicate estimated locations. The blue line on the top graph depicts the reconstructed path. The blue line in the bottom Figure depicts the error of the estimated path (m).

4.2 RF-based methods

In this Section results of all methods based on RF-signals are presented. Datasets that are used in this Section are the fingerprint dataset (see Appendix A.1) for the signal attenuation model, K-nearest neighbours, circular lateration, and fingerprinting. The WiFi noise dataset (see Appendix A.3) is used for evaluating WiFi characteristics, and is discussed next.

4.2.1 WiFi noise

Data in this experiment was gathered using an Android application that was specifically designed for this task (see Appendix B.3 for details). Every 30 seconds the application measures RSSI-values of all visible access points and stores the information in a data file. The dataset used in this experiment was gathered over the course of multiple days (just over 80 hours) to account for temporal effects, resulting in a dataset of at most 9600 data points for each individual access point.

Results of this experiment are shown in Figure 4.5, in which the RSSI-values of the 9 individual access points are plotted. Mean and standard deviation for each of these APs are displayed in Table 4.2. But note that these values are more difficult to interpret. What cannot be seen from the plotted RSSI-values in Figure 4.5 is that some of the access points have missing values. It sometimes happens that access points are unobserved during a WiFi scan. There are multiple explanations for the non-occurrences (e.g. signal blocked by moving objects, noise in the transmitter, noise in the receiver), but the main point is that these non-occurrences are not taken into account when the mean and SD are computed.

AP	Mean (dBm)	SD (dBm)	Missing (%)	Distance (m)*
1	-81.6173	1.5862	15.39	22.47
2	-79.6210	1.2320	1.12	22.00
3	-73.8517	2.4356	12.99	16.00
4	-76.4868	1.0983	6.08	16.12
5	-59.4780	2.3388	1.03	9.43
6	-58.7839	1.9845	1.06	8.00
7	-29.2970	4.5512	0.11	2.00
8	-62.7986	1.8510	0.62	11.31
9	-63.3485	1.2333	0.00	8.00

TABLE 4.2: WiFi noise. **Euclidean distance between transmitter (access point) and receiver (phone).*

4.2.2 Signal attenuation modeling

The signal attenuation model is used to relate physical distance between transmitter (access point) and receiver (mobile phone) to RSSI measurements (and vice versa). Except for fingerprinting, K-nearest neighbours, and dead reckoning all localization methods in this thesis rely on signal modelling. It is therefore important to have a good understanding of the characteristics of signal modeling and the limitations of the model.

It might be preferred to set a delimiter for the maximum range an access point can cover based on building characteristics and physical limitations of transmitters (access points) and receivers (mobile phones). The delimiter could be set to the distance at which it is

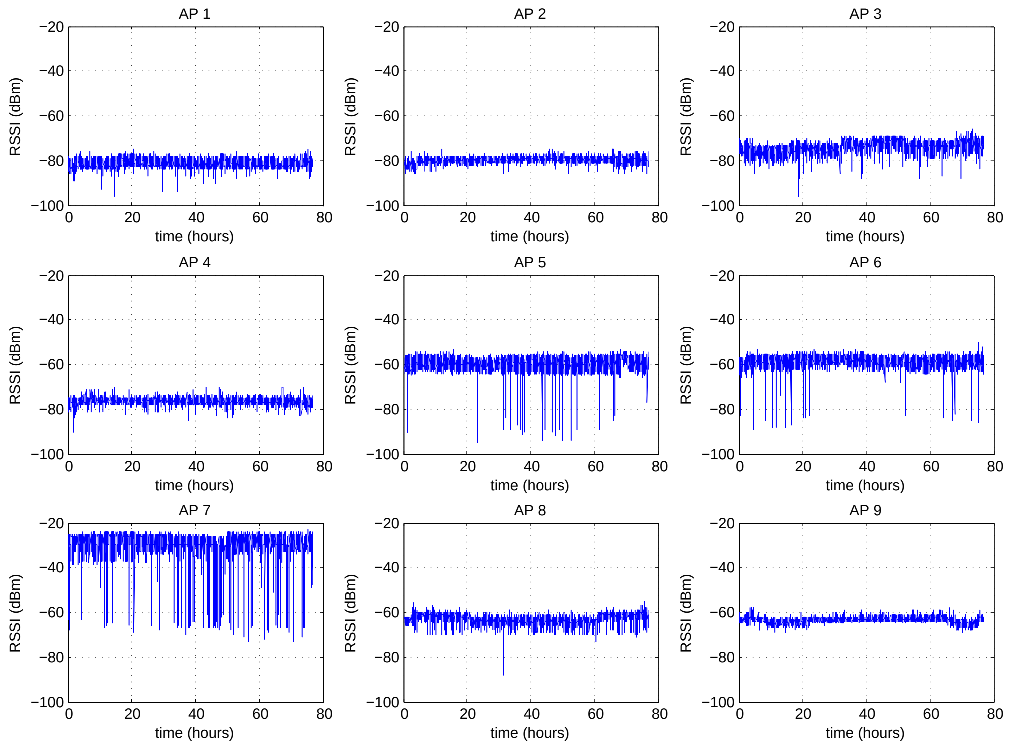


FIGURE 4.5: RSSI measurements of 9 individual access points. WiFi samples were taken every 30 seconds for the duration of approximately 80 hours.

absolutely certain that no signal could have been received from the transmitter. Using brute force testing it was found that the best performance on the attic dataset at Sense HQ (see Appendix A.1 for details) was achieved when no delimiter is used. The brute force method optimizes for the standard deviation of the error (e.g. try to fit a smallest as possible distribution). Also the mean error is then automatically minimized. Parameter optimization stopped as soon as the mean error started to increase. This resulted in a normal distribution around a mean prediction error of $\mu = 0.09$, and $\sigma = 4.40$ (See Figure 4.6). The parameter settings $\gamma = 5.2$, $d_0 = 30.0$, and $P_t = -46.0$, without using a delimiter resulted in the best performance for this particular environment. The model parameters determined here are also used in further experiments that rely on signal modelling. Note that these results are location specific. Model parameters presented here are fitted on this dataset and by no means representative for any other environment, even though most model parameters resemble default values for this type of environment.

In Figure 4.7, RSSI measurements of the fingerprint dataset are displayed in green, along with the signal attenuation model displayed as the blue line. Signal attenuation model parameters are as described above. The signal strength data plotted here illustrate the main challenge of modeling RF-signal attenuation. RSSI measurements show extremely high variance as can be seen in Figure 4.7. At a distance of $2m$, signal strength varies between $-35dBm$ and $-80dBm$, whereas signal strength measurements at a range of $25m$ are $-80dBm$ on average. That is, if an RSSI measurement of $-80dBm$ is observed, it might be that the distance between transmitter and receiver is somewhere between

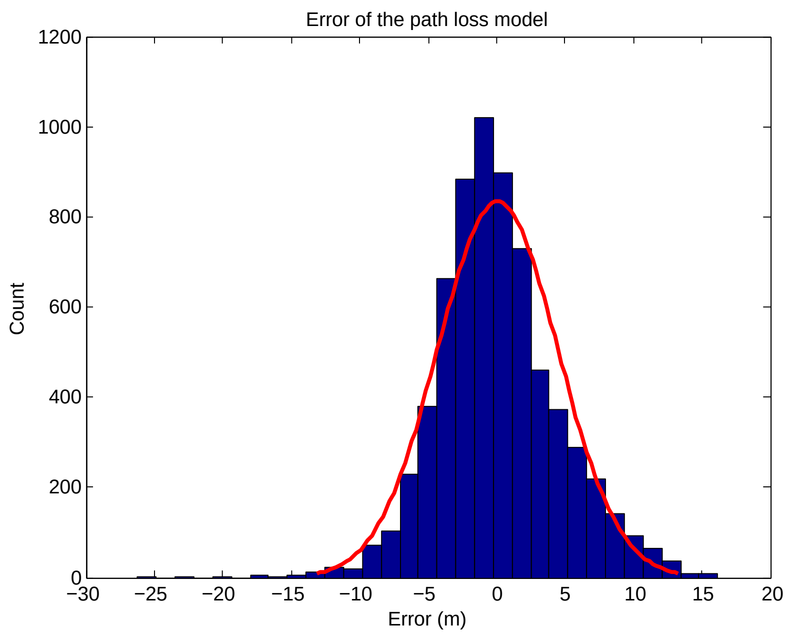


FIGURE 4.6: Prediction peformance of the signal attenuation model with parameter settings $\mu = 0.09m$, $\sigma = 4.40m$.

$2m$ and $25m$ or even more. This huge variance in signal strength measurements makes it extremely hard to reliably predict distances between transmitter and receiver based on RF-signals.

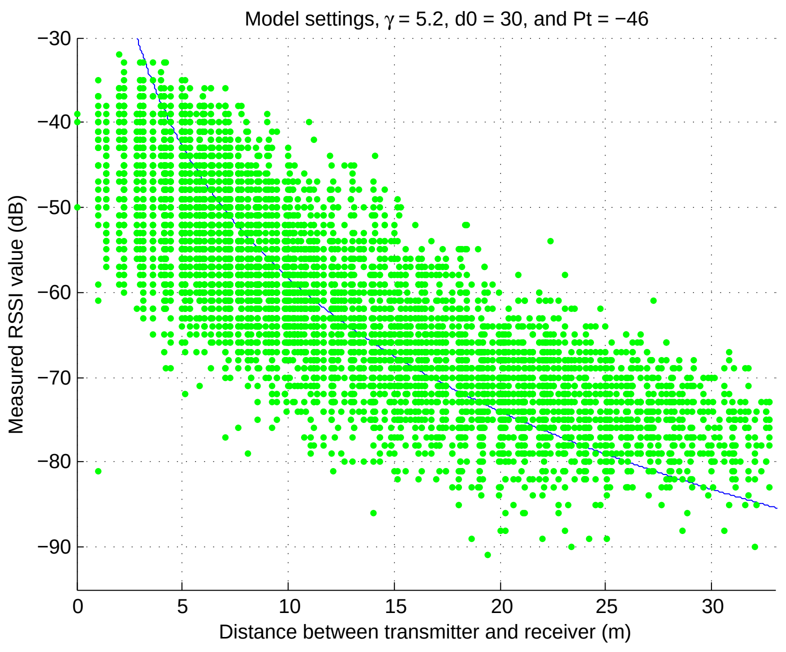


FIGURE 4.7: Signal attenuation model with parameter settings $\gamma = 5.2$, $d_0 = 30.0$, and $P_t = -46.0$, is shown in blue. RSSI measurements are depicted in green.

4.2.3 Fingerprinting

For this fingerprinting experiment, a dataset was created at the attic of Sense HQ. This environment is approximately $35m$ long, $7m$ wide, and $4m$ high with only minor obstructions in the center of the attic. At both ends of the attic, there are two smaller rooms which are separated by a gypsum block wall. A structure of wooden beams supports the roof throughout the whole environment (see Appendix A.1 for a schematic overview). The dataset for this experiment was created using a simple Android application which creates data points that consist of WiFi measurements[1] and measurement locations[2] (see Appendix B.1 for details about the application). The environment was divided into 212 squares of $1m^2$, and 4 measurements were taken at each square. This resulted in a dataset of 848 data vectors of WiFi signal strength measurements (e.g. $[AP_1, AP_2, ..., AP_K]$) divided into 212 clusters. Signal strength in this dataset varied between $-30dBm$ and $-91dBm$. Some data vectors contained missing values or non-observations, meaning that the access points were too far away to detect.

In this fingerprint experiment, data vectors were classified to the cluster which had the least Euclidean distance between cluster prototype and data vector. The best results were observed after inserting the lowest observed RSSI value ($-91dBm$) in data vectors where non-observations occurred. The experiment showed that only 32.8% of all data vectors were classified correctly. However, in case of incorrect classification the matching cluster is often very close to the actual location. The average distance between the actual cluster and the matched cluster is $2.16m$. An overview of classification performance and error radius is given below in Table 4.3.

Radius	**Correct (%)**
0m	32.78
1m	43.40
2m	56.25
3m	70.17
4m	79.36
5m	88.44
6m	94.10
7m	97.05
8m	98.47
9m	98.82
10m	99.17

TABLE 4.3: Fingerprint classification accuracy. For an error radius of $10m$, 99.17% is classified correctly.

[1]A WiFi measurement consists of BSSID (MAC-address), RSSI (received signal strength), SSID (network name), and Frequency of all visible WiFi access points.

[2]Users of the Android application need to manually enter the location (e.g. x, and y coordinates) to create this dataset.

With a total of 212 clusters, the amount of correct classified clusters definitely outperforms random guesses, even at a radius of $0m$. In Figure 4.8 this result is visualized by means of a similarity matrix. The similarity matrix is used as a measure for the similarity of data points with the clusters they belong to. If all data vectors were to be classified correctly, the similarity matrix resembles a perfect diagonal. The similarity matrix shown in Figure 4.8 indicates that there are a lot of data vectors classified incorrectly. However, the distance between the cluster they belong to is often not very large (e.g. data vectors are classified correctly or close to - in physical space - the correct cluster).

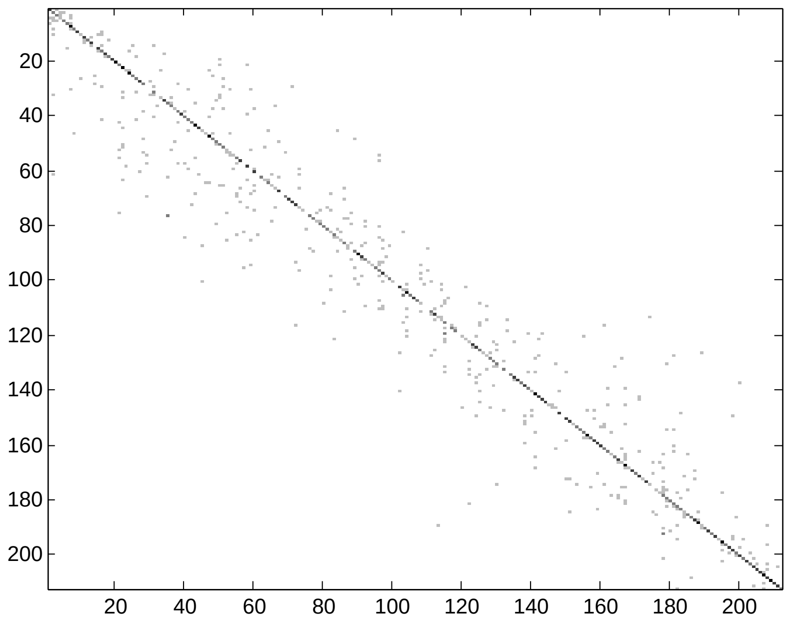


FIGURE 4.8: Similarity matrix of clusters in a Fingerprinting experiment.

These results might also be influenced by the amount of access points that are used in the data vectors. Intuitively, using a large amount of access points should result in higher classification performance compared to using a small amount of access points. Exploiting a large amount of access points increases the discriminability of clusters, and should therefore achieve higher precision and accuracy. For instance, if only one access point is used, it is impossible to differentiate in more than one dimension. This is also reflected in the results. Figure 4.9 shows the classification performance as a function of distance from the actual location for multiple amounts of access points. Each individual graph is the result of using a different amount of access points. Each graph is generated by averaging over 50 individual experiments, where one experiment consists of randomly selecting a fixed amount of access points out of all available ones and computing the precision at different distances ($0m - 10m$). Figure 4.9 shows that at a radius of approximately $8m$, a classification performance of nearly 100% is possible.

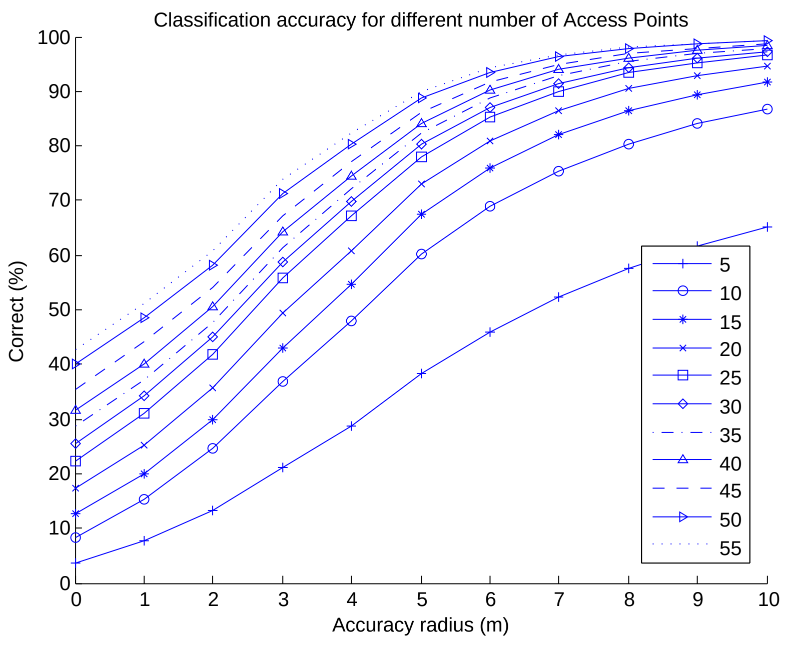


FIGURE 4.9: Fingerprint accuracy and precision. Each graph displays the accuracy after using a different amount of access points.

4.2.4 K-nearest neighbour

The K-nearest neighbours experiment reuses the dataset created for fingerprinting (see Appendix A.1 for details about the dataset). Mean localization error and standard deviation of the K-nearest neighbours experiment for $K = \{1, ..., 9\}$ are presented below in Figure 4.10. For this type of localization technique, the distribution and density of the access points and shape of the environment can have a notable effect on the performance of this technique. In the current dataset, K-nearest neighbours is most likely ineffective because nearly all access points are located on the edge of the environment, and the environment is shaped in a rectangular way. Meaning that for $K > 2$, predicted locations will inevitably result in a location prediction pulled towards the center of the environment, because the third or fourth 'nearest neighbours' are almost certainly on the opposite side of the rectangular environment (see Appendix A.1 for map details). Locating a device on the edge of the environment will therefore most likely have consistent bad performance for larger values of K. Theoretically, performance can only degrade for most locations in this map with increasing values of K.

These expectancies are also reflected in the results. Optimal performance is observed for $K = 3$, closely followed by $K = 2$, and $K = 4$. For $K = 3$, the average localization error is $\mu = 3.75$ and $\sigma = 1.94$. For $K > 4$, localization error starts to increase significantly.

The increase in localization error due to larger values of K is visualized in Figure 4.11. This Figure displays the the performance of K-nearest neighbours for $K = \{1, ..., 9\}$, for each location on the map. Colors indicate the localization performance per location. This Figure clearly shows a positive correlation between larger values of K and increasing

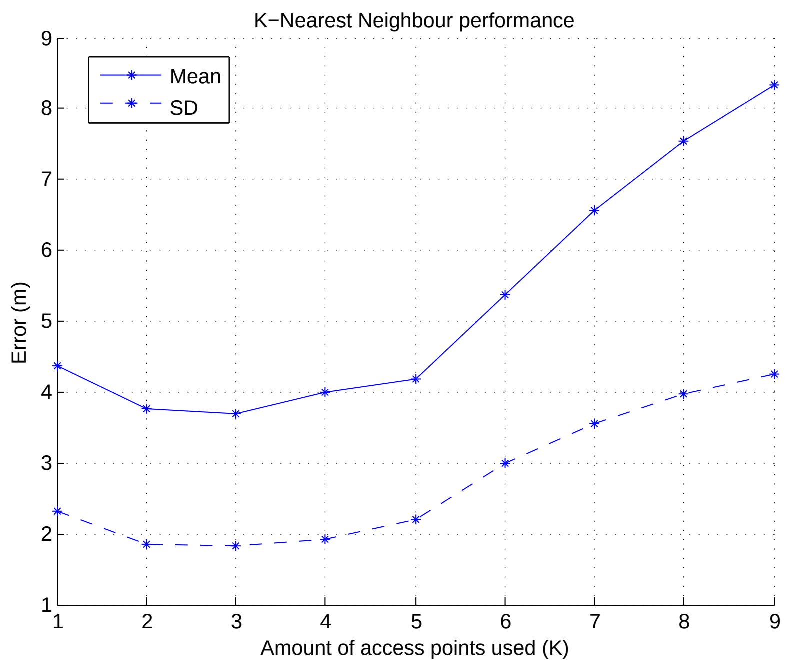


FIGURE 4.10: K-Nearest Neighbour performance for $K = \{1, ..., 9\}$. Mean and standard deviation are shown.

localization error for the outer edges. For $K = 9$, only the center locations are localized correctly. Note that for larger values of K it is not always possible to construct a localization error because not all access points are observed. For $K > 6$, gaps start to emerge because of missing data.

4.2.5 Circular lateration

In this circular lateration experiment the fingerprint dataset recorded at the attic of Sense HQ is used, see Appendix A.1 for details. The effect of averaging WiFi measurements over time is also evaluated.

The performance of the circular lateration experiment is presented in Figure 4.12. Results for default circular lateration are shown in blue and results of a so called 'averaged' version in which WiFi measurements are averaged over time are displayed in red. In the averaged version, multiple WiFi measurements recorded at different points in time are averaged in order to obtain more reliable RSSI values. The assumption is made that by averaging multiple measurements the arbitrary effects of RF-signals is cancelled out. Theoretically the averaged version should outperform the default version because the effect of outliers is cancelled out. Now, for both the default and averaged versions, the circular lateration performance is shown as a function of the amount of access points. For lower amounts of access points the nearest – loudest in signal space (dBm) – access points were selected. For instance, for the situation with 3 access points, the 3 loudest or closest access points are used for the experiment. It is always best to select the

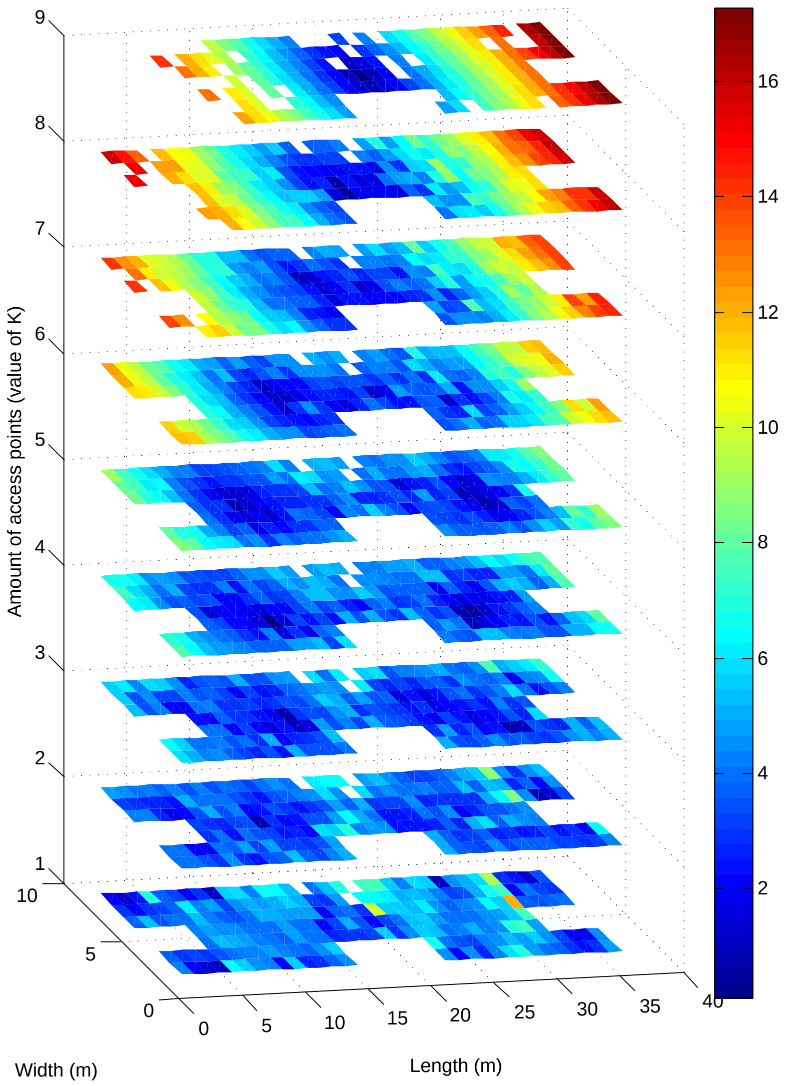


FIGURE 4.11: K-nearest neighbours performance per location, for $K = \{1, ..., 9\}$. Colors indicate the estimation error (m).

loudest access points because the signal attenuation model is more accurate in the range closer to the access point compared to predictions that are further away (see Figure 4.7). A pilot experiment with randomly selecting access points confirmed that performance dramatically decreases if the selected access points happen to be access points far away from the device location. Note that RSSI measurements are evaluated in signal space and are used to estimate the location of a mobile device, but the performance of circular lateration is reported in Euclidean distance measured in physical space.

In [19] a method is presented to locate mobile devices using RSSI-based circular lateration. Results in [19] show that using their method, accuracy is at best 5 meters, and degrades to as far as 20 meters using less reliable circular lateration methods. Our results reflect these results for large amount of access points, but never degrade into localization errors of more than $\mu = 5.5$ and $\sigma = 2.6$ (for using 9 access points). Furthermore, the 'averaged version' outperforms all results presented in [19] regardless of the amount of access points. Our results show that the best performance is achieved if 4 or 5 closest access points are used. The localization error of the averaged version when using 4 access points is $\mu = 2.93m$ and $\sigma = 1.65m$. Localization error for the non-averaged version is $\mu = 4.09$ and $\sigma = 2.91$. It must be noted that the density of access points is rather high for this environment. It might be that the results are influenced in a positive way by the high density of access points.

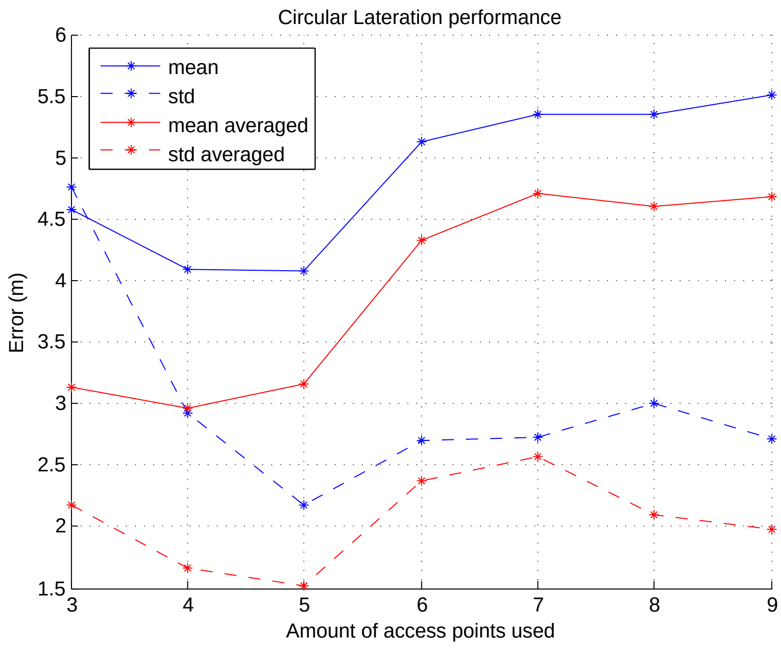


FIGURE 4.12: Circular Lateration performance. The graphs depict mean, μ, and standard deviation, σ, for $AP = \{3, ..., 9\}$.

Also note that the results presented here are obtained from WiFi measurements where participants were stationary. It is unclear how moving subjects might influence the results. Especially the outcome of the averaged version is hard to predict. If multiple measurements from different locations are averaged, the predicted device location might very well be 'lagging behind' the actual location.

4.3 Simultaneous localization and mapping

The extended Kalman filter simultaneous localization and mapping (EKF SLAM) algorithm, as presented in Section 3.3, was first assessed using a simulated dataset on overall performance and convergence using a simulated environment. The simulated environment is approximately $50m$ long and $40m$ wide. In total the simulation environment contains 10 landmarks distributed evenly across the area. A schematic 2-dimensional map of the simulation environment is shown in Figure 4.13. The blue line depicts the user's walking route, the red asterisks represent landmark locations. The walking route is traversed 10 times, so every location is visited 10 times in total. The dataset resulting from the simulated environment consists of simulated motion and WiFi measurements (e.g. simulated steps, bearing, and WiFi observations).

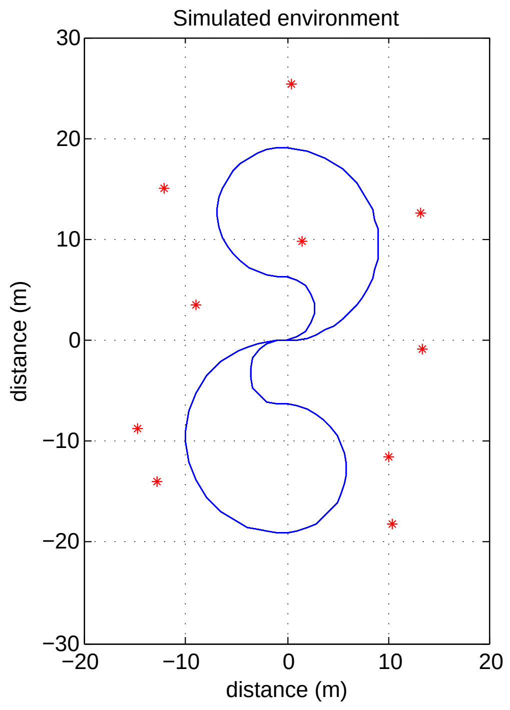


FIGURE 4.13: Schematic 2-dimensional map for evaluating the EKF SLAM algorithm performance. The blue line indicates walking route. Red asterisks indicate landmark locations.

A graphical user interface (GUI) was developed to assess the performance of the algorithm visually. The GUI can be used to visualize real world data obtained from data gathering applications or to visualize simulated datasets. Figure 4.14 displays the GUI after an experiment based on the simulated environment. The final localization error after 900 prediction-correction steps for the user location is $0.36m$, with an uncertainty boundary of $0.53m$ in the horizontal direction, and $0.47m$ in the vertical direction. These results are very satisfying and provide some insight in the maximum achievable performance on real world data. These results might however be somewhat misleading because the simulated environment and dataset contain an idealized situation (e.g. low amount of noise).

The mapping and localization algorithm can also be assessed in terms of landmark location convergence. Results of landmark convergence for the 10 landmarks from the simulated environment are shown in Figure 4.15. Each graph in this Figure depicts the convergence of a single landmark. After 900 prediction-correction steps, the average error of all landmarks is $\mu = 1.64$, with a standard deviation of $\sigma = 1.02$. Note that some landmarks are not initialized correctly and errors increase for some landmarks even after initializations. This is most likely to happen in real world situations also. In the simulation environment eventually all landmarks converge to a stable situation.

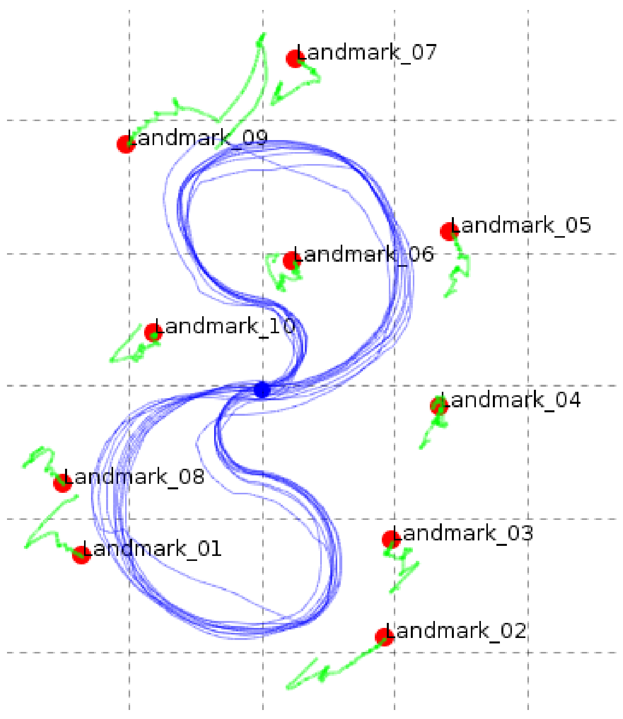


FIGURE 4.14: 2-dimensional GUI used for assessment of the EKF SLAM implementation. The blue line represents the estimated trajectory of the device. The green lines indicate state changes of landmarks. The red marks indicated the final estimated locations of landmarks. The dashed gray lines plotted on the background are used as a reference grid. The distance between grid lines is $10m$.

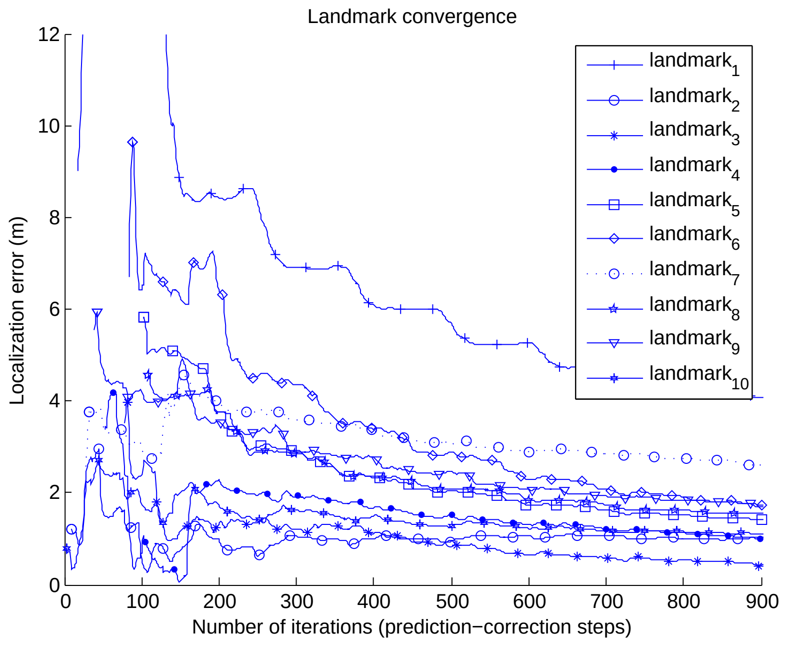


FIGURE 4.15: Landmark convergence for simulated data. Each graph depicts the localization error over time of an individual landmark. The mean localization error of all landmarks after 900 iterations is $\mu = 1.64$, and $\sigma = 1.02$.

Performance on the simulated dataset looks promising and suitable to apply on real world data. The algorithm converges to a stable situation for both the device location and all landmarks. Below the results of three experiments based on real world data are discussed. In the experiments below, the performance of dead reckoning is compared to EKF SLAM and multi-user EKF SLAM. Datasets for these experiments were gathered in the Bernoulliborg at the University of Groningen using the dataset gatherer application (see Appendix A.4 for details).

The results of these three experiments are presented in Figures 4.16, 4.17, and 4.18. For each experiment, the top graph displays the actual path in red and the reconstructed path is displayed in blue. The asterisks indicate marked locations[3] which are used to determine the localization error. Red asterisks represent actual locations, green asterisks are the estimated device locations. Participants moved from $(0, 0)$, via $(-41, 19)$, visiting $(-34, 19)$ twice, towards $(-55, 23)$, and moved back to $(0, 0)$ following the same route in reverse. In the bottom graph the localization error is displayed by means of Euclidean distance between actual location (red marks) and reconstructed location (green marks) for three algorithms: (1) dead reckoning, (2) EKF SLAM, and (3) multi-user EKF SLAM[4].

In all three experiments, multi-user EKF SLAM outperforms the other methods as expected. Unfortunately the expected reduction of drift by combining motion sensors and WiFi sensors is far less than hoped for. Figure 4.16 shows the results of the first localization and mapping experiment. The reconstructed path displayed in the top graph is determined with multi-user EKF SLAM, which was the best performing method as can be seen in the bottom graph. Striking about this experiment is the huge increase in localization error of the EKF SLAM method midway the walking route. The localization error increases to approximately $30m$ and drops back to $15m$ after which the localization error continues to diminishes even further. Interestingly this steep increase in localization error is not observed in either dead reckoning or the multi-user variant. Further analysis showed that the increase in error is due to wrong landmark initializations. Incorrectly initialized landmarks result in incorrectly predicted device locations as is shown here. Eventually landmarks converge towards their actual locations and device localization error is decreased. The convergence of landmark locations is confirmed by the multi-user experiment. In the multi-user variant the steep increase of localization error is not observed because this method continues with the map data from the single-user variant in which landmark locations already converged.

Results of the second experiment are shown in Figure 4.17. Conditions and experimental setup are equal to those of the first experiment. However, for the second experiment it must be noted that at the end of the recording of the user's track, the data gathering

[3]Participants can mark their location by pressing a button in the Android application during the recording of their walking path. Visually high recognizable locations were chosen as 'marked locations'. See Appendix A.4 for details.

[4]The horizontal axis displayed in the bottom graph is not equal to traveled distances. The horizontal axis in the bottom graph displays results for subsequently visited marked locations, which is related to traveled distance, but not the same. The traveled distance is larger because marked locations are more than $1m$ apart.

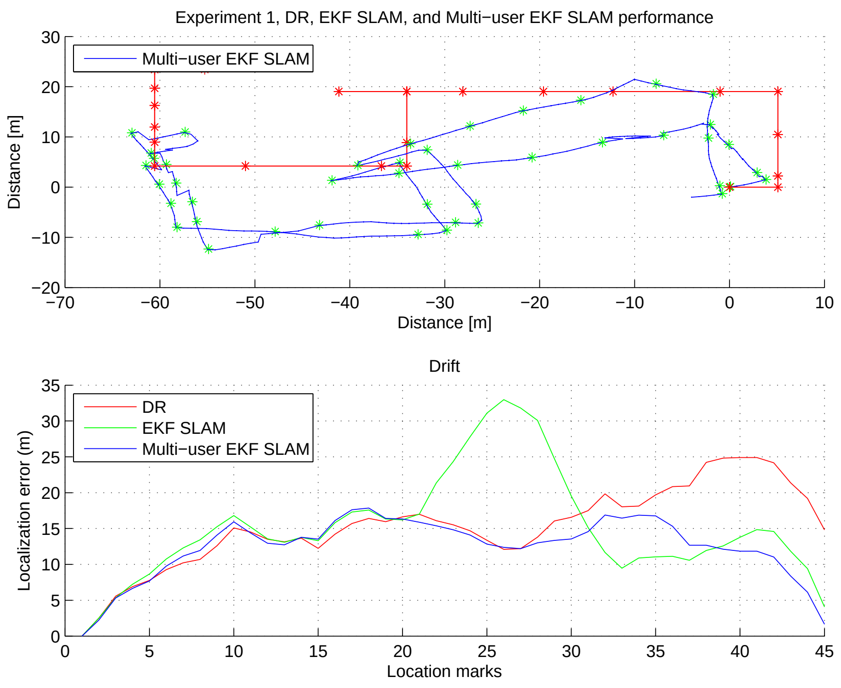


FIGURE 4.16: Performance of dead reckoning, EKF SLAM and multi-user EKF SLAM. The top Figure depicts the reconstructed path (blue line) obtained from multi-user EKF SLAM and the actual path (red line). The bottom Figure displays the error between predicted and actual path. Participants walked from $(0, 0)$, via $(-41, 19)$, towards $(-55, 23)$, and back again to $(0, 0)$.

application failed to measure WiFi samples (i.e. unable to apply the correction step using WiFi measurements in the (multi-user) EKF SLAM approach). If, for whatever reason, corrections are unavailable, performance of all three methods is expected to resemble that of dead reckoning. This effect is clearly visible in the sudden increase in localization error for all three methods at the end of the route. And similar to the first experiment, the EKF SLAM approach displays a significant increase in localization error somewhere midway the walking route due to incorrect landmark initializations. Localization error increases from approximately $5m$ to $20m$ for a short period of time, after which the localization error diminishes to approximately $10m$. The multi-user variant does not suffer from this large increase in error because it reuses map information. The overall best performing method in this experiment is multi-user EKF SLAM, of which the reconstructed path is displayed in the top graph.

Finally, results of the third experiment are shown in Figure 4.18. For this experiment conditions and setup are again equal to those in the first and second experiment. Out of all three localization and mapping experiments the results of this third experiment resulted in the best overall performance for all methods. The localization error varies between $5m$ and $10m$ for the larger part of the experiment. Performance decreases to a localization error of approximately $15m$ for both EKF SLAM variants and twice as much for the dead reckoning approach. The increase in localization error is caused by faulty measurements of the orientation (motion) sensors. Interestingly the error introduced by

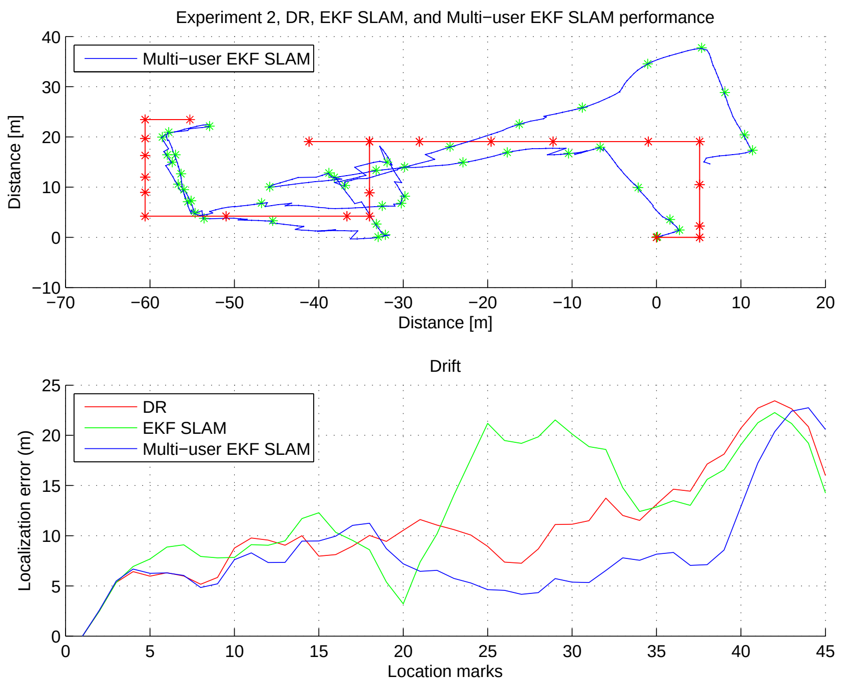
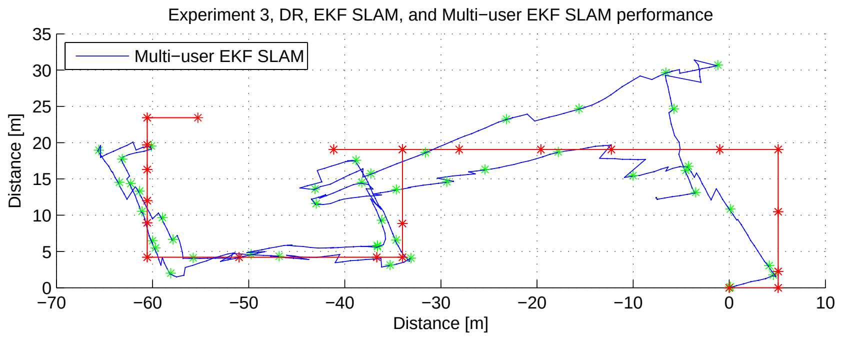


FIGURE 4.17: Performance of dead reckoning, EKF SLAM and multi-user EKF SLAM. The top Figure depicts the reconstructed path (blue line) generated by multi-user EKF SLAM and the actual path (red line). The bottom Figure displays the error between predicted and actual path.

faulty motion measurements, can indeed be corrected using WiFi measurements as is shown in the superior performance of both EKF SLAM methods. Both EKF SLAM and multi-user EKF SLAM outperform dead reckoning in the last part of this experiment because the dead reckoning is unable to compensate for erroneous motion measurements whereas both EKF SLAM methods are able to compensate for the faulty motion measurements using landmark corrections. Although both EKF SLAM and multi-user EKF SLAM suffer from the erroneous measurements, the reduced increase in error indicates the effectiveness of correcting localization prediction with WiFi measurements.

4.4 Summary

In this Chapter the performance of localization and mapping methods was evaluated. In the first part of this Chapter, the performance of motion-based methods was presented by means of a dead reckoning experiment. In this experiments a 2-stage EKF sensor fusion algorithm was compared to the default Android orientation sensor. The 2-stage EKF sensor fusion algorithm showed a considerable performance increase compared to the default Android orientation sensor. Although the performance of both methods was quite poor in general, the 2-stage sensor fusion algorithm showed a huge improvement over the Android sensor.



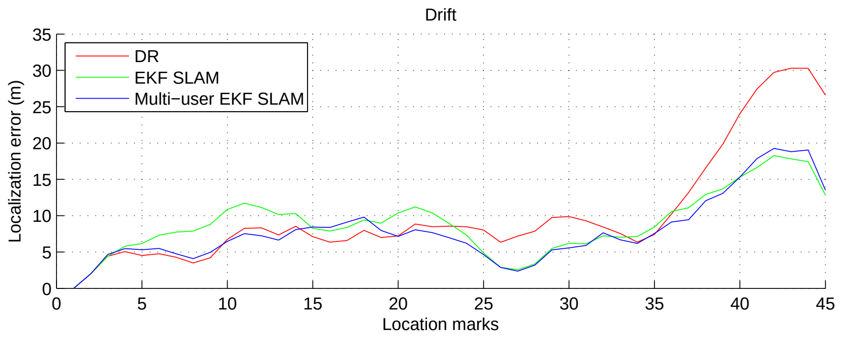


FIGURE 4.18: Performance of dead reckoning, EKF SLAM and multi-user EKF SLAM. The top Figure depicts the reconstructed path (blue line) generated by multi-user EKF SLAM and the actual path (red line). The bottom Figure displays the error between predicted and actual path.

In the second part of this Chapter, results of three RF-based methods were presented: (1) fingerprinting, (2) K-nearest neighbours, and (3) circular lateration. All of these experiments were conducted on the fingerprint dataset (see Appendix A.1). Out of these three methods, fingerprinting requires the most labour to configure. For each location, multiple WiFi measurements need to be recorded which can be a tedious job for large amounts of locations. In order to locate a device, K-nearest neighbours and circular lateration only require knowledge about access point locations. The best performing method was circular lateration with an average error of $\mu = 2.93m$ and $\sigma = 1.65$. Although this might be considered to be an unfair comparison with the other methods because these results were obtained after averaging multiple WiFi measurements recorded at different points in time. Averaging was not applied to the other methods, and is left for future research. Furthermore, averaging is not always possible, for instance if users are walking with a fast pace while using circular lateration, the predicted device location might 'lag behind' the actual location.

Optimal performance of K-nearest neighbours was observed for $K = 3$, with an average localization error of $\mu = 3.75m$ and $\sigma = 1.94m$. Fingerprinting showed close to 100% correct classification if an error radius of approximately $8m$ is used.

Finally, in the third part of this Chapter, the effect of combining motion sensors and WiFi sensors was evaluated by means of (multi-user) EKF SLAM. Performance of multi-user EKF SLAM was tested with three individual experiments and compared to dead reckoning and EKF SLAM. Note that the problem of localization *and* mapping is a much harder problem compared to RF-based methods such as circular lateration and fingerprinting in which map data is constructed offline. In all three experiments the multi-user EKF SLAM method outperformed dead reckoning and EKF SLAM. Unfortunately the expected reduction in drift errors by combining motion data and WiFi measurements is less than expected. The best results were observed in the third experiment where localization error varied between $5m$ and $10m$ for the larger part of the experiment.

Chapter 5

Conclusion and Future Work

5.1 Conclusions

In this thesis methods for indoor-localization and indoor-localization and mapping based on mobile phone sensors were presented. The goal of this thesis was to accurately locate mobile devices in indoor situations using only mobile phone sensors, without changing the environment (ceteris paribus).

This thesis was divided into three categories of methods: (1) indoor-localization methods, which locate the device based on a priori map information, (2) device tracking methods, which consists of a model-based approach to track devices, and (3) indoor-localization and mapping methods, which locate the device and build a map online. Out of these three categories, the latter is by far the most difficult since no a priori map information is available and map information must be build online. In all of these categories, mobile phone motion sensors and WiFi sensors provided the necessary input for all algorithms. Mobile phone motion sensors were used to track and predict the next locations (e.g. in tracking experiments). Mobile phone WiFi sensors provided localization information by means of signal attenuation modeling and information on WiFi access point locations.

For indoor-localization methods (1), fingerprinting, K-nearest neighbours, and circular lateration, were assessed on their overall performance and maximum accuracy. Fingerprinting requires by far the most effort to configure. All locations that need to be distinguished, need to be 'fingerprinted' first, whereas K-nearest neighbours and circular lateration only require access point locations to operate. Results showed that circular lateration was the best performing method with the highest accuracy, closely followed by K-nearest neighbours. The best observed localization performance of circular lateration was with a localization error of $\mu = 2.93m$ and $\sigma = 1.65m$. The best performance of K-nearest neighbours was observed for $K = 3$, with a localization error of $\mu = 3.75$ and

$\sigma = 1.94$. Fingerprinting showed close to 100% correct classification when using an error radius of 8 meters.

For device tracking methods (2), dead reckoning, extended Kalman filter simultaneous localization and mapping (EKF SLAM), and multi-user EKF SLAM were tested. These three methods were compared and evaluated on their performance of device tracking and the latter two methods are furthermore assessed on convergence. In three individual experiments it was found that multi-user EKF SLAM outperforms the other two methods consistently. Drift errors observed in dead reckoning were successfully corrected using the multi-user EKF SLAM approach. In two out of three experiments the EKF SLAM approach showed a temporarily increase in localization error due to incorrectly initialized landmark locations. However, eventually the EKF SLAM method was able to recover from this increase in error. Even though multi-user EKF SLAM outperformed dead reckoning and EKF SLAM on device tracking, localization error varied between $5m - 20m$.

Finally, for indoor-localization and mapping methods (3), EKF SLAM and multi-user EKF SLAM were compared on their overall performance and convergence. In three individual experiments it was shown that multi-user EKF SLAM consistently outperforms EKF SLAM on localization and convergence. In all of these experiments the multi-user variant could benefit from earlier established landmarks which showed an increase in localization performance compared to the EKF SLAM variant.

Next, the research questions posed in Section 1.2 are answered.

1. *How can mobile phone sensors be utilized for (1) indoor-localization and (2) indoor-localization and mapping?*

   In this thesis mobile phone sensors were successfully used to locate a mobile phone in an indoor environment using a priori map information and mobile phone WiFi sensors. Methods such as K-nearest neighbours and circular lateration showed localization error of at best $\mu = 2.93m$ and $\sigma = 1.65m$. Furthermore, mobile phone motion sensors and WiFi sensors were successfully combined for localization and mapping. Multi-user EKF SLAM consistently outperformed dead reckoning and showed localization errors varying between $5m - 20m$.

   (a) *What is the maximum accuracy of the evaluated methods, fingerprinting, K-nearest neighbours, and circular lateration, on indoor-localization?*

   The maximum accuracy on indoor-localization was observed for circular lateration after averaging multiple WiFi measurements over time. The average localization error of circular lateration when using 4 nearest access points is $\mu = 2.93m$ and $\sigma = 1.65m$.

(b) *Which of the evaluated methods, dead reckoning, extended Kalman filter simultaneous localization and mapping (EKF SLAM), and multi-user EKF SLAM, performs best on device tracking?*

Multi-user EKF SLAM consistently showed to outperform dead reckoning and EKF SLAM on device tracking. In three individual experiments it was shown that, multi-user EKF SLAM was able to successfully compensate drift errors observed in dead reckoning.

(c) *Which of the evaluated methods, EKF SLAM, and multi-user EKF SLAM, performs best on localization and mapping?*

Three individual experiments showed that multi-user EKF SLAM consistently outperformed EKF SLAM on localization performance and convergence. It was found that multi-user EKF SLAM could successfully benefit from earlier established and converged landmarks. Although multi-user EKF SLAM outperformed EKF SLAM, overall localization error varied between $5m - 20m$.

5.2 Future work

Some parts in this thesis are left open for future research. The effect of motion on the performance of RF-based methods such as circular lateration, fingerprinting, and K-nearest neighbours is unclear. Future studies might focus on assessment of localization prediction of moving targets. It might very well be that, if not accounted for, predicted device locations lag behind the actual location of moving targets.

Probably the K-nearest neighbours approach can be improved by applying averaging to WiFi measurements. Averaging greatly improved results of circular lateration, mainly due to the inability of circular lateration to deal with outliers. K-nearest neighbours might benefit from a similar approach.

For future research it would be interesting to evaluate the performance of multi-user EKF SLAM with hundreds or more participants. Converging map data from a large amount of users could possibly greatly improve certainty and accuracy of landmark locations. Signal attenuation modeling is unlikely to improve in the near future, but if landmark locations are known more accurately it would certainly aid in device localization.

Appendices

Appendix A

Datasets

To assess the performance of all localization and mapping methods presented in Chapter 3, multiple datasets were created. A brief overview of all datasets is provided below.

Datasets were generated at two different locations: (1) the attic at Sense HQ, Rotterdam, and (2) the second floor of the Bernoulliborg, faculty building of mathematics and natural sciences, University of Groningen.

A.1 Fingerprints

The fingerprint dataset contains signal strength measurements at 212 locations (gray squares in figure A.1) at the attic of Sense HQ. This dataset is used to determine optimal parameters for the signal attenuation model (see Eq. 2.3.1), and to assess the performance of fingerprinting (see Section 3.2.1), K-nearest neighbours (see Section 3.2.2), and circular lateration (see Section 4.2.5). Data is gathered using the fingerprinting application (see Appendix B.1).

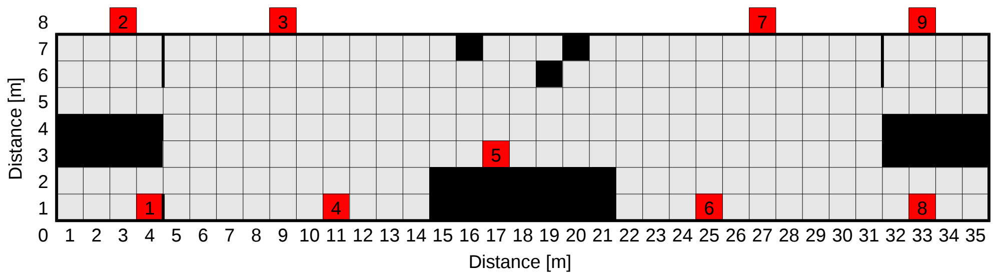


FIGURE A.1: 2-dimensional schematic floor plan of the attic at Sense HQ. The labelled red squares indicate installed access point locations (9 in total). The gray areas represents the fingerprinted area, and the areas marked black are walls or other obstacles.

At each of the 212 location, 4 measurements were recorded. This resulted in a dataset of 848 data points, where one data point consists of (1) location coordinates (x, y) and

(2) a WiFi measurement of all visible access points. A WiFi measurement consists of BSSID (MAC-address), RSSI (received signal strength) measured in dBm, SSID (network name), and frequency.

A.2 Motion noise

The motion noise dataset is a relatively small dataset which was gathered at the Bernoulliborg using the motion noise application (see Appendix B.2). This dataset was used to evaluate the precision of motion sensors. A noise analysis experiment based on this dataset is performed in Section 4.1.1. The dataset contains measurements of all mobile phone motion sensors such as gyroscope, (linear) accelerometer, and orientation sensor. Measurements were recorded at a frequency of approximately $20Hz$ for about 10 minutes.

A.3 WiFi noise

The WiFi noise dataset is gathered at the attic of Sense HQ using the WiFi noise application (see Appendix B.3), and is used to evaluate the characteristics of WiFi signal strength (dBm) over time. The dataset contains WiFi measurements recorded over the course of approximately 80 hours and an interval of 30 seconds. A WiFi noise analysis was performed with this data and is presented in Section 4.2.1.

A.4 User tracking

The user tracking datasets were gathered (1) at the attic of Sense HQ and (2) at the second floor of the Bernoulliborg, faculty building of mathematics and natural sciences, University of Groningen. The datasets were gathered using the dataset gatherer application (See Appendix B.4), and were used to compare the performance of dead reckoning with EKF SLAM, and multi-user EKF SLAM.

The datasets are stored in JSON-format, each datapoint in the dataset contains: a (1) date, the time stamp at which the data point was created, (2) location time stamp, a variable used to reconstruct the times at which a user visited a location. Users are instructed to walk a certain path and click a button in the application once they reach a marked location. In this manner the user's path can be reconstructed, (3) (linear) accelerometer measurements, (4) gyroscope measurements, (5) orientation sensor measurements, (6) bearing measurement based on 2-stage EKF algorithm (see Section 2.2.5), and finally (7) a list of WiFi measurements. Data points were created with an interval of approximately $50ms$.

Figure A.2 depicts a 2-dimension map of the Bernoulliborg and the 23 unique marked locations used for performance assessment. The green large circles are supposed to be access point locations. In this environment three user tracking experiments were performed.

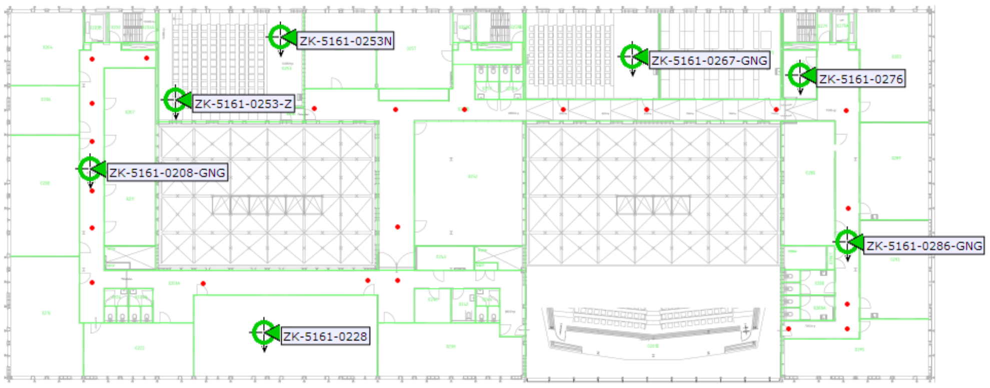


FIGURE A.2: Map of the second floor of the Bernoulliborg, faculty building of mathematics and natural sciences, University of Groningen. Red dots indicate marked locations.

Figure A.3 displays a schematic 2-dimensional floor plan of the attic at Sense HQ and an example path participants were instructed to walk depicted in blue. Red squares indicate access point locations.

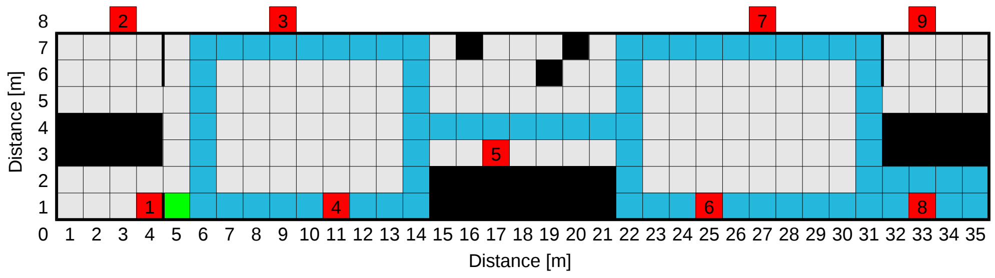


FIGURE A.3: 2-dimensional schematic floor plan of the attic at Sense HQ. The labelled red squares indicate installed access point locations (9 in total). The gray areas represent the fingerprinted area, and the areas marked black are walls or other obstacles.

Appendix B

Android Applications

This Appendix contains a brief overview of Android applications written during this thesis. The applications were built to (1) analyse noise of sensors, (2) record walking tracks of users, and (3) to create datasets to evaluate localization algorithms.

B.1 Fingerprinter

The main and only goal of this fingerprinting application is to generate a fingerprint dataset (see Appendix A.1 for details about the dataset). After pressing a refresh button the home screen of the application displays a list of all observed access points. For each list item (access point) the application displays the network name (SSID) and signal strength (RSSI), as well as an estimated distance to the access point based on the signal attenuation model. The application also features an option to submit the data to the common-sense platform. Before submitting the data, an x and y location need to be entered manually. Data are uploaded to the platform in JSON-format and can easily be retrieved in the same format as it was submitted. Submitted data contains a lot more details than is displayed in the application. See Appendix A.1 for detailed information about the exact data format and resulting fingerprint dataset.

B.2 Motion noise analyser

The noise analyser application is used to evaluate the precision of motion sensors. It is a simple application that measures the phone's gyroscope, (linear) accelerometer, and orientation sensor at a frequency of approximately $20Hz$. Data are stored in JSON-format on the phone. See Appendix A.2 for details about the dataset resulting from this application.

B.3 WiFi noise analyser

The WiFi noise analyser application was developed to simplify the measurement of WiFi signals over the course of a large period of time. The application can run in the background, but for practical reasons the phone displays a warning screen about a running experiment. The application records WiFi measurements at a frequency of $2Hz$ and can run indefinitely. WiFi measurements consist of BSSID (MAC-address), RSSI (received signal strength), SSID (network name), and channel frequency.

See Appendix A.3 for the resulting dataset.

B.4 Tracking application

The user tracking application was created to track the walking route of users in multiple experiments. The application records both motion and WiFi sensors. The motion sensor operates at a frequency of approximately $20Hz$ and measures sensor values of (linear) accelerometer, gyroscope, and orientation sensor. The WiFi sensor operates at the maximum allowed frequency by the phone model, which in most cases does not exceed $\sim 0.25Hz$. The WiFi sensor records BSSID (MAC-address), RSSI (received signal strength), SSID (network name), and channel frequency.

This application is designed to gather data for EKF SLAM experiments and dead reckoning. Participants need to click a button on every marked location to keep track of their true location. The main screen of the application displays a large button participants can click on intersections. The main screen also shows some additional information like the last time a WiFi scan was received, WiFi scan interval, name of the data file to which data is written, and a start/stop button to start/stop the recording.

See Appendix A.4 for details about datasets gathered with this application.

Bibliography

[1] A. Ahmad, S. Huang, J. Wang, and G. Dissanayake. A new state vector for range-only SLAM. In *Control and Decision Conference (CCDC), 2011 Chinese*, pages 3404–3409, may 2011.

[2] N. Aloui, K. Raoof, A. Bouallegue, S. Letourneur, and S. Zaibi. A novel indoor localization scheme based on fingerprinting technique And CDMA signals. In *Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on*, pages 1–5, 2012.

[3] J.-L. Blanco, J. Fernandez-Madrigal, and J. Gonzalez. Efficient probabilistic Range-Only SLAM. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 1017–1022, 2008.

[4] L. Bruno and P. Robertson. WiSLAM: Improving FootSLAM with WiFi. In *Indoor Positioning and Indoor Navigation (IPIN), 2011 International Conference on*, pages 1 –10, sept. 2011.

[5] D. Cole and P. Newman. Using laser range data for 3D SLAM in outdoor environments. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1556–1563, 2006.

[6] A. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1403–1410 vol.2, Oct 2003.

[7] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part I. *Robotics Automation Magazine, IEEE*, 13(2):99–110, june 2006.

[8] C. Evrendilek and H. Akcan. On the Complexity of Trilateration with Noisy Range Measurements. *Communications Letters, IEEE*, 15(10):1097–1099, october 2011.

[9] B. Ferris, D. Fox, and N. Lawrence. WiFi-SLAM using Gaussian process latent variable models. In *Proceedings of the 20th international joint conference on Artifical intelligence*, IJCAI'07, pages 2480–2485, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[10] A. Goldsmith. *Wireless Communications.* Cambridge University Press, New York, NY, USA, 2005.

[11] S. Hilsenbeck, A. Moller, R. Huitl, G. Schroth, M. Kranz, and E. Steinbach. Scale-preserving long-term visual odometry for indoor navigation. In *Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on*, pages 1–10, 2012.

[12] H. Hu and D. Gu. Landmark-based navigation of industrial mobile robots. *Industrial Robot*, 27(14):458–467, 2000.

[13] A. Jimenez, F. Seco, C. Prieto, and J. Guevara. A comparison of Pedestrian Dead-Reckoning algorithms using a low-cost MEMS IMU. In *Intelligent Signal Processing, 2009. WISP 2009. IEEE International Symposium on*, pages 37–42, Aug 2009.

[14] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[15] E. D. Kaplan. *Understanding GPS Principles and Applications.* Artech House Mobile Communications, 1996.

[16] Y. Kim, Y. Chon, and H. Cha. Smartphone-Based Collaborative and Autonomous Radio Fingerprinting. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 42(1):112–122, January 2012.

[17] M. B. Kjaergaard, H. Blunck, T. Godsk, T. Toftkjaer, D. L. Christensen, and K. Gronbaek. Indoor positioning using GPS revisited. In *Proceedings of the 8th international conference on Pervasive Computing*, Pervasive'10, pages 38–56, Berlin, Heidelberg, 2010. Springer-Verlag.

[18] V. B. Kokshenev. Dynamics of Human Walking at Steady Speeds. *Phys. Rev. Lett.*, 93:208101, Nov 2004.

[19] J. Koo and H. Cha. Localizing WiFi Access Points Using Signal Strength. *Communications Letters, IEEE*, 15(2):187–189, February 2011.

[20] M. Kourogi, T. Ishikawa, and T. Kurata. A method of pedestrian dead reckoning using action recognition. In *Position Location and Navigation Symposium (PLANS), 2010 IEEE/ION*, pages 85–89, 2010.

[21] A. Kushki, K. N. Plataniotis, and A. N. Venetsanopoulos. *WLAN Positioning Systems: Principles and Applications in Location-Based Services.* Cambridge University Press, New York, NY, USA, 2012.

[22] S.-H. Lee and Y.-H. Lee. Adaptive frequency hopping for bluetooth robust to WLAN interference. *Communications Letters, IEEE*, 13(9):628–630, 2009.

[23] H. Liu and G. Pang. Accelerometer for mobile robot positioning. *Industry Applications, IEEE Transactions on*, 37(3):812–819, may/jun 2001.

[24] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[25] T.-L. Nguyen, Y. Zhang, and M. L. Griss. ProbIN: Probabilistic inertial navigation. In *MASS*, pages 650–657. IEEE, 2010.

[26] M. Oner, J. Pulcifer-Stump, P. Seeling, and T. Kaya. Towards the run and walk activity classification through step detection - An android application. In *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, pages 1980–1983, 28 2012-sept. 1 2012.

[27] S. Pandey and P. Agrawal. A survey on localization techniques for wireless networks. *Journal of the Chinese Institute of Engineers*, 29(7):1125–1148, 2006.

[28] P. Robertson, M. Angermann, and B. Krach. Simultaneous Localization and Mapping for Pedestrians Using Only Foot-mounted Inertial Sensors. In *Proceedings of the 11th International Conference on Ubiquitous Computing*, Ubicomp '09, pages 93–96, New York, NY, USA, 2009. ACM.

[29] P. Robertson, M. G. Puyol, and M. Angermann. Collaborative Pedestrian Mapping of Buildings Using Inertial Sensors and FootSLAM. *Proceedings of the 24th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2011)*, 24:1366–1377, sept. 2011.

[30] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Proceedings of the 9th European conference on Computer Vision - Volume Part I*, ECCV'06, pages 430–443, Berlin, Heidelberg, 2006. Springer-Verlag.

[31] S. Sabatelli, M. Galgani, L. Fanucci, and A. Rocchi. A Double-Stage Kalman Filter for Orientation Tracking With an Integrated Processor in 9-D IMU. *Instrumentation and Measurement, IEEE Transactions on*, 62(3):590–598, 2013.

[32] U. Steinhoff and B. Schiele. Dead reckoning from the pocket - An experimental study. *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 162–170, 2010.

[33] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series)*. The MIT Press, 2005.

[34] K. Tumkur and S. Subbiah. Modeling Human Walking for Step Detection and Stride Determination by 3-Axis Accelerometer Readings in Pedometer. In *Computational Intelligence, Modelling and Simulation (CIMSiM), 2012 Fourth International Conference on*, pages 199 –204, sept. 2012.

[35] S. Vanini and S. Giordano. Adaptive context-agnostic floor transition detection on smart mobile devices. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*, pages 2–7, 2013.

[36] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury. No need to war-drive: unsupervised indoor localization. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, MobiSys '12, pages 197–210, New York, NY, USA, 2012. ACM.

[37] G. Welch and G. Bishop. An Introduction to the Kalman Filter. Technical report, Chapel Hill, NC, USA, 1995.

[38] Z. Zeng, B. Allen, Z. Liu, and A. Aghvami. Evaluation of IEEE 802.11b and Bluetooth coexistence in an office environment. In *3G Mobile Communication Technologies, 2004. 3G 2004. Fifth IEE International Conference on*, pages 54–58, 2004.

[39] P. Zhou, Y. Zheng, Z. Li, M. Li, and G. Shen. IODetector: a generic service for indoor outdoor detection. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, SenSys '12, pages 113–126, New York, NY, USA, 2012. ACM.