AirLoc: Pedestrian dead reckoning for passenger localization

An airport specific indoor localization system for airlines.

A.F.A. de Moes
AirLoc: Pedestrian dead reckoning for passenger localization
An airport specific indoor localization system for airlines.

by

A.F.A. de Moes

for the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday January 17, 2016 at 14:00.

Student number: 1354728
Thesis committee:
Prof. dr. ir. K.G. Langendoen, TU Delft, Chair
Dr. ir. R.R. Venkatesha Prasad, TU Delft, supervisor
Dr. ir. C.J.M. Verhoeven, TU Delft
Dr. C. Sarkar, TU Delft, supervisor

An electronic version of this thesis is available at http://repository.tudelft.nl/.
Abstract

Travelling through airports can be quite a stressful experience for passengers. Airlines, such as KLM, want to make the journey of their passengers through airports as comfortable as possible. This thesis proposes an indoor localization system for airlines called AirLoc. The research question was: "Which airport specific indoor localization method can be developed, that is ideal for airlines and can be applied at multiple airports?". Different localization methods were discussed and pedestrian dead reckoning was chosen as the most suitable method, because it is infrastructure free and requires no additional investments. AirLoc is implemented on an Android smartphone and uses a motion model as a sensing stage, pedestrian dead reckoning as the localization method and a particle filter combined with landmarks as the refinement method. The location based services include the passenger journey and airport navigation. A particle filter was created that can be used on a complex polygon based map, by using polygon geometry and by creating special collision detection rules and optimizations, such as obstacle distance measurements. Map data of different airports can be included to make the system work at multiple airports.

A complete motion model was created that does step detection, step length estimation and heading estimation dynamically. The heading estimation can be used while the phone is in different orientations, by using an initialization stage in which the GPS heading is used to determine the walking direction relative to the orientation of the phone. This estimation has an error of 30 degrees.

AirLoc was compared to the Polestar system, which is based on Bluetooth fingerprinting and is used in the Schiphol app. The system was tested in two orientations, hand-held and in a pocket. AirLoc has a mean error of 6.17 meters when held in hand and an accuracy of 7.23 meters when put inside the pocket. The estimated location of AirLoc is always in the correct area and at a reachable location thanks to the collision detection of the particle filter and it removes the glass wall problem introduced by infrastructure based methods. This makes AirLoc more suitable for location based services such as airport navigation. AirLoc consumes only half of the energy consumed by infrastructure based systems, such as Polestar.

A system is also proposed that calculates queuing times at the airport security, by using activity monitoring combined with an x-ray landmark. Testing shows that the queueing time could be estimated with an error of 14 seconds.
Preface

This Master thesis is the final part of my Master of Science in Embedded Systems at Delft University of Technology. The presented work was done at the headquarters of KLM at Amstelveen and is a collaboration between KLM and the Embedded Software Group led by Prof. dr. Koen Langendoen. This thesis describes the research that was done while developing a possible solution and implementation for KLM’s location based services.

In my work I describe a practical, airport specific, indoor localization method based on pedestrian dead reckoning. This report was written for the master thesis committee and for KLM to see what is possible for their smartphone application in the near future. A paper with the title; “Airloc: Pedestrian dead reckoning for passenger localization” will be submitted to the IEEE IoT Journal. Video’s of this project can be found on my Youtube channel:

https://www.youtube.com/user/Alexanderdemoes

For any questions or remarks, do not hesitate to contact me.

A.F.A. de Moes
Delft, December 2016
Acknowledgments

This thesis describes the work done by me during the past 8 months at KLM for a Master of Science degree in Embedded Systems at the Delft University of Technology.

I wish to express my deepest gratitude, to Marco van Heerde, Bart Bruijnesteijn and Jon Eshuijs for giving me the opportunity to do this great project and for guiding me through the world of KLM. I would like to thank the entire KLM mobile team, including Frederik Kempe and Koen Vos, for letting me be part of the team. I would also like to thank my supervisors, Dr. Ranga Rao Venkatesha Prasad and Dr. Chayan Sarkar for guiding me through the project. Special thanks to Willem de Lang, for helping me with the images in this thesis and Joost van der Gaag, for providing Android development tips. Lastly, I would like to thank my parents, who have aided me throughout my entire studies and have given me their unconditional support.
List of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KLM</td>
<td>Koninklijke Luchtvaart Maatschappij N.V. (Royal Dutch Airlines)</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>BLE</td>
<td>Bluetooth Low Energy</td>
</tr>
<tr>
<td>app</td>
<td>Smartphone application</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>AP</td>
<td>Wireless Access point</td>
</tr>
<tr>
<td>RSSI</td>
<td>Received signal strength indication</td>
</tr>
<tr>
<td>UUID</td>
<td>Universally unique identifier</td>
</tr>
<tr>
<td>AoA</td>
<td>Angle of arrival</td>
</tr>
<tr>
<td>AoD</td>
<td>Angle of departure</td>
</tr>
<tr>
<td>SDK</td>
<td>A software development kit</td>
</tr>
<tr>
<td>API</td>
<td>Application program interface</td>
</tr>
<tr>
<td>PDR</td>
<td>Pedestrian dead reckoning</td>
</tr>
<tr>
<td>LIDAR</td>
<td>Light Detection and Ranging</td>
</tr>
</tbody>
</table>
Contents

Abstract iii
Preface v
Acknowledgments vii
List of Acronyms ix

1 Introduction 1
  1.1 Background ................................................. 1
  1.2 Motivation .................................................. 2
  1.3 Problem statement ........................................... 3
  1.4 Solution approach ......................................... 3
  1.5 Contributions .............................................. 4
  1.6 Outline ..................................................... 4

2 State-of-the-art 5
  2.1 Basic indoor localization techniques ......................... 5
    2.1.1 Trilateration ........................................... 6
    2.1.2 Triangulation .......................................... 6
    2.1.3 Fingerprinting .......................................... 7
    2.1.4 Proximity sensing ...................................... 7
    2.1.5 Pedestrian Dead Reckoning .............................. 8
    2.1.6 Comparison ............................................. 9
  2.2 Advanced indoor localization techniques ..................... 9
  2.3 Enterprise solutions ........................................ 11
    2.3.1 Indoor Atlas ........................................... 11
    2.3.2 Polestar ................................................. 11
    2.3.3 Apple Core Location .................................. 11
    2.3.4 Accengage ............................................... 11
  2.4 Queueing .................................................. 11
    2.4.1 Bliptrack .............................................. 11
  2.5 Indoor mapping ............................................ 12
    2.5.1 Locus Labs ............................................. 12
    2.5.2 Google indoor maps .................................. 12
    2.5.3 Apple venue maps .................................... 12

3 System Design 13
  3.1 Introduction .............................................. 13
  3.2 AirLoc architecture ......................................... 13
  3.3 Location based services .................................... 14
  3.4 Queueing time ............................................. 15

4 The motion model 17
  4.1 Introduction .............................................. 17
  4.2 Sensor orientation .......................................... 17
  4.3 Step detection ............................................. 18
  4.4 Step length estimation ..................................... 19
  4.5 Heading estimation ......................................... 19
    4.5.1 Related research ....................................... 19
    4.5.2 Acceleration based heading estimation .................. 20
    4.5.3 Relative heading estimation ............................ 20
    4.5.4 Absolute heading direction ............................. 21
Introduction

Travelling through airports can be quite a stressful experience for passengers. They are usually on a tight schedule and they don’t always know where they must go, especially at unfamiliar airports. Thus many airlines want to make the journey of their passengers through airports as stress-free as possible. In this age of smartphones, a smartphone based solution is expected to be the most viable one.

Traditionally, the smartphone application of an airport takes care of navigating passengers through the airport. Recent research however, has shown that passengers do not download apps of the airport they are travelling through, but prefer the app of the airline they are travelling with [19]. This can be seen in the amount of downloads that airport apps have compared to airline apps. For example, the KLM (the Royal Dutch Airlines) app has 1 million users, while the Schiphol (the airport in Amsterdam) app has only 100,000 users. Similarly, the British Airways app has almost 5 million users, while the Heathrow (the main airport in London) app has only 500,000 downloads. A simple reason is that a person may fly with a particular airline in majority of the time, but travels through various airports. That is why airlines want to integrate indoor navigation into their smartphone application.

1.1. Background

The goal of this work is to develop an app-based indoor navigation system that can be used at various airports by an airline. The app should tell the passenger how to get to his/her destination at the airport and how much time it will take to get there. This work is carried out in consultation with KLM at the Schiphol airport. Thus, majority of the requirements are driven by their needs. However, the system presented in this work is generic enough (also the requirements and assumptions) that it can be utilized by any airlines at any airport.

KLM decided to divide location based services into three parts; Indoor wayfinding, track & trace and location based push. Indoor wayfinding focusses on showing the user’s current position on a map and getting customizable maps for multiple airports. Track & trace focusses on mapping passengers on an airport and location based push focusses on sending push messages based on the current location of the passenger. Figure 1.1 shows three concept designs of how KLM wants to implement indoor localization and navigation into their app. The design on the left shows amount of time it will take for the passenger to arrive at his gate. The design on the left shows amount of time it will take for the passenger to arrive at his gate. The design in the middle shows point to point navigation. The location of the user is placed on the map of the airport and the direction is shown. The design on the right shows the passenger’s journey from the entrance of the airport all the way to the gate. It shows which checkpoints are crossed and which checkpoints are still ahead.

The smartphone application will be referred to as ‘app’ for the remainder of this thesis.
1. Introduction

(a) Walking time to gate.  (b) Point to point navigation.  (c) Customer journey.

Figure 1.1: Concept designs for location based services.

1.2. Motivation

In order to navigate a passenger through an airport, the first thing that has to be determined is the passenger's location. In outdoor situations, GPS can be used in order to determine someone's location. However, in indoor environments such as airports, GPS is too unreliable, and that is why indoor localization techniques have to be applied. Current indoor localization methods at airports mostly rely on fingerprint based methods based on Wi-Fi or Bluetooth signal strength. For example, "Apple Core location" is a location based service based on Wi-Fi fingerprinting, Polestar use fingerprints of Bluetooth beacons for indoor localization, etc. The Schiphol airport also uses the Bluetooth fingerprinting provided by Polestar. Moreover, the integration of map data makes the app bulkier. For example, the Schiphol app uses ESRI map SDK [50], which makes it around 80 MB. The product owners of the app want to keep the size of the app below 100MB, because smartphone users are less likely to install and more likely to uninstall an app above 100MB [48]. Moreover, apps above 100 MB can only be installed when the smartphone is connected to a Wi-Fi connection. Thus there is a need to build apps that are low in footprint.

The inclusion of location based services in any airline's app has many benefits for customers, the airport and Airline itself.

- For the customer it can remove uncertainty and stress by directing them to the required destination at the airport and telling them how much time it will take to get there.
- It can also optimize the customer's journey by adding extra services such as having a taxi waiting at the right time.
- The airline can determine how many of their passengers will make the flight and provide their passengers with location based offers such as an upgrade to business class, while they are waiting at the gate or transfer their customers to an earlier flight when the customer arrives early.
- The airline can locate lost passengers, such as unaccompanied minors, VIP's and disabled people. It costs KLM 250 Euro per delayed passenger [12], so if KLM knows a passenger will not make it to the plane on time, the passenger can be rebooked and the plane can leave on time. Airlines have to pay the airport when their flights are delayed and they also have to fly faster, which burns more fuel, in order
to arrive at the destination on time. Furthermore, transfer passengers that miss their flight because of a delayed KLM flight need to be given shelter until the next flight. All these factors combined result in 250 Euro per passenger on average. This could be reduced by providing location based services. Also, a seamless customer journey, by providing relaxed and tailored door-to-door journeys, helps KLM to become a more customer intimate and efficient airline.

The travelling time at airports is not fixed and heavily depends on so called choke points. These are points all passengers must pass through and take the most time to pass, such as security and passport checks. In order to provide valid travelling times, the queuing times at these choke points need to be estimated. Airlines are not able to install any hardware in airports, so a solution that does not require any additional hardware would greatly benefit airlines. In order to provide a valid travelling estimate, the queuing times at these choke points need to be estimated.

1.3. Problem statement

The goal of this research is to investigate the best indoor localization method for an airline. An airline operates at many airports and does not always have access to the infrastructure of an airport. Moreover, an airline is not able to install hardware, such as beacons, on an airport and for this reason the best indoor localization method for an airline is an infrastructure free localization method. An infrastructure free indoor localization method is a method of localization that does not require any information about the environment, like the location of beacons or Wi-Fi access points. A generic solution will not suffice for an airport. An airport specific solution enables a look at features that do not exist elsewhere, such as airport specific landmarks. In addition, the amount of people at airports change dramatically during the day and airports have very large and complex maps compared to the offices, which are used in most localization research. For this reason, an airport specific solution is proposed in this thesis. The main research question is formulated as follows:

| Which airport specific indoor localization method can be developed, that is ideal for airlines and can be applied at multiple airports? |

To obtain an answer to this research question, the following more detailed questions must also be answered:

1. How can a pedestrian dead reckoning system be created that supports multiple floors?
2. How can a complex polygon based map be used for particle filtering?
3. How can a complete motion model be created that does step detection, step length estimation and heading estimation dynamically?
4. How can landmarks be used to improve localization accuracy?
5. Is there an infrastructure free method to estimate queuing times dynamically?
6. How can location based services improve the passenger journey?

1.4. Solution approach

An ideal indoor localization method for airlines is a localization method that does not require any infrastructure, because airlines cannot install any hardware at airports. Furthermore, fingerprinting requires an off-line phase and since airports are so big, it takes a lot of time to fingerprint the airport. In order to provide navigation in an airport, the estimated location should be in the correct zone. For these reasons AirLoc was created. AirLoc is a localization system that uses pedestrian dead reckoning as the localization method, a motion model as the sensing stage and particle filters combined with landmarks. This method is ideal because it requires no additional sensors and no off-line phase. The particle filter is added as a refinement method to
improve accuracy and to make sure that the estimated location cannot move through walls or be at unreachable places. Creating AirLoc throws quite some challenges. First a motion model had to be created that can detect steps, determine the step length dynamically and estimates the correct heading for persons of different heights and sizes and also of different nationality. The particle filter had to be extended in order to work on a giant map by using polygon geometry and optimizations that make the processing time fast enough to run on a smartphone. Landmarks had to be added in order to switch between different floor levels. In order to find these landmarks, sensor analysis had to be performed at these locations. AirLoc is also able to provide location based services to the smartphone user. In order to provide accurate walking times, an off-line method to detect queueing times had to be found. This was done by detecting the current activity of the user during queueing. In order to detect these activities, sensor analysis during queueing was performed. This data can then be provided to other smartphone users. the above challenges are addressed in the remaining chapters of this thesis.

1.5. Contributions

This thesis proposes an airport specific, particle filter based indoor localization method that requires no additional hardware. The contributions of this thesis can be summarized in the following topics:

- Practical airport specific localization technique that does not require any set up time or investment cost and can be used at any airport as long as the airport map is available, which is an ideal solution for airlines. This is a suitable solution for airlines since they do not require any information of the infrastructure other than the map, which is always needed for any form of localization.

- Particle filter on a polygon based map, which support multiple floors by using landmarks. Airport specific landmarks such as escalators, elevators and x-ray scanners allow for recalibration of the particle filter and floor change for a pedestrian dead reckoning indoor localization system.

- GPS hand-off allows for a self-sustaining indoor localization method. The initial position and heading are determined by the last known reliable GPS heading.

- Completely dynamic motion model that is able to determine the walking distance and heading estimation independent of the orientation.

- Queue estimation that does not require any additional hardware.

- The proposed system removes the Glass wall problem introduced in infrastructure based indoor localization method, by making sure that the estimated location cannot move through walls or be at unreachable places.

1.6. Outline

In chapter 2 an introduction into every related aspect of airport localization is given. The localization system and location based services are discussed in Chapter 3. Chapter 4 describes the complete motion model and how each variable in the motion model is estimated. Chapter 5 explains how the airport map is constructed and how the data can be used for particle filtering. Chapter 6 discusses how the particle filter and landmarks at the airport can refine the indoor localization accuracy. The system is evaluated in Chapter 7. In chapter 8 conclusions are drawn and opportunities for future work are discussed.
State-of-the-art

The Global positioning system (GPS) offers an accurate solution for outdoor localization based on satellite trilateration [6]. Most modern smart phones are equipped with GPS, because it provides the location of the smartphone to the user. The accuracy of GPS is 4 to 8 meters outside. GPS signals are transmitted at 1575.42 MHz and are classified as part of the ultra-high frequency band [39]. These high frequency signals are easily scattered and attenuated by walls, roofs and other objects of which a building consists, which is why the GPS signal quality is poor inside buildings. This makes GPS a poor choice for determining the location of a person inside a building and this is the reason why new methods for indoor localization had to be developed. Many different indoor localization methods have been suggested in research, but a universal solution has not yet emerged. Some of these solutions have been implemented by the industry and are in use today at airports. This chapter explores the current state of the art in research and the enterprise solutions currently in use by airports.

2.1. Basic indoor localization techniques

Indoor localization is a system to localize devices or people inside a building by using available resources such as radio waves, magnetic fields or sensor information collected by a device. The most common indoor localization techniques are; fingerprinting, triangulation, trilateration and pedestrian dead reckoning. An indoor localization system consists of three parts, sensing, localization and position refinement. This is shown in figure 2.1. In the sensing stage, the required data is gathered and used in the localization. Mathematical algorithms, such as Bayesian or Kalman filters then are used for position refinement in order to improve the localization accuracy. An overview of the common indoor localization techniques can be found in table 2.1 and these methods are discussed in the following subsections.

![Figure 2.1: The three pillars of an indoor localization system. A sensing stage, the localization technique and the position refinement.](image)
2.1.1. Trilateration

In trilateration localization, the received signal strength indication (RSSI) from an access point or beacon, is received by the smartphone and first translated into a distance. The distance from at least 3 access points or beacons is required in order to determine the current location. Trilateration is an infrastructure based localization method, because it requires information of the environment, in this case the locations of the access points or beacons. Kaishun et al. investigated the error of a simple trilateration system and achieved an accuracy of 1.2m meters [52]. A graphical representation of trilateration can be found in figure 2.2.

![Figure 2.2: A visual representation of trilateration.](image)

2.1.2. Triangulation

In triangulation, either the angle of arrival (AoA) or the angle of departure (AoD) is measured. Triangulation is an infrastructure based localization method, because it requires information of the environment, in this case the locations of the access points or beacons. The angle of arrival is the angle between the signal send by a Wi-Fi access points or Bluetooth beacons and received by a mobile device. This angle can be measured using several antennas placed side by side and by measuring the phase difference between the signals received by the antenna array [51]. Since no commonly used smartphone has multiple Bluetooth or wi-fi antennas, this technology cannot be used in this research. In angle of departure triangulation, the angle between the signal send from the smartphone is measured by the access points or beacons and the angles between the smartphone and the access points are acquired by the phase difference between the signals received by the access point or beacon array. Yao et al. investigated angle of departure localization in [53] by using an L-shaped antenna array. Also, Gunhardson achieves an accuracy of 4 meters using this method in [23]. A graphical representation of angle of departure triangulation can be found in figure 2.3.
2.1. Basic indoor localization techniques

2.1.3. Fingerprinting

Unlike triangulation and trilateration, fingerprinting is an infrastructure free localization method, meaning it does not need to measure a relation between some measurable size and location. Instead, the received signal strength indication (RSSI) is gathered throughout the mapped area. Fingerprinting consists of an on-line and an off-line phase. In the off-line phase, a map of fingerprints is generated, in which ambient signals are measured and matched with physical locations. These fingerprinted signals depend on what is available in the environment and can be Wi-Fi, Bluetooth or even magnetic fields. During the on-line phase, a device that observes collects a real-time fingerprint and compares it to the fingerprint map. The location is then estimated by using a pattern matching and refinement algorithm. Basic fingerprinting has an accuracy of 3 to 5 meters according to Ding et al.\cite{18}. A graphical representation of fingerprinting can be found in figure 2.4.

2.1.4. Proximity sensing

Proximity sensing is a localization method which is based purely on the proximity of the smartphone to a known location. This can be the location of a Bluetooth beacon or Wi-Fi access point. For this reason proximity sensing is an infrastructure based localization method. Proximity sensing does not provide a location in
the form of coordinates, but instead in the form of sets of possible locations, namely the locations of the beacons or access points. This method is often used in combination with Bluetooth beacons, where an action, such as a push message to a smartphone, is triggered when the smartphone is in proximity of a beacon [21]. Proximity sensing with Bluetooth beacons is investigated by Naya et al. They show that Bluetooth devices can exchange their mutual proximity information at a rate of more than 1 Hz [37]. An average localization error of 0.38m is achieved, but this is only for a single access point and does not give an absolute location, but instead gives the location of the access point. A graphical representation of proximity sensing can be found in figure 2.5.

Figure 2.5: A visual representation of proximity sensing.

2.1.5. Pedestrian Dead Reckoning
Pedestrian dead reckoning is a localization (PDR) method where the current position is determined by a previously determined position. Dead reckoning is subjected to cumulative errors, because of this. To implement this localization method on a smartphone, the motion of the user has to be analysed. The travelled distance and heading direction need to be calculated in order to localize the user. This allows for a localization method that is infrastructure free and does not require any additional hardware like beacons. Pratama et al. achieved an accuracy of 1.4 meters using a basic PDR method on a smartphone [40]. A visual representation of PDR is shown in figure 2.6.

Figure 2.6: A visual representation of PDR localization.
2.2. Advanced indoor localization techniques

2.1.6. Comparison

The suitable localization methods need to work on smartphones and need to be usable at airports. All methods can be used at airports, but triangulation will require special beacons or access points that can calculate the angle of departure. Since smartphones do not possess an antenna array, the angle of arrival cannot be calculated on a smartphone. For these reasons, triangulation is not a suitable method for airport localization. The infrastructure based methods, trilateration, triangulation and proximity sensing can be used at airports, but they require accurate infrastructure information, such as the location and ID of beacons or Wi-Fi access points. At airports where no Wi-Fi or beacons are available, only a PDR approach is possible, since all other methods require external hardware. Buying this hardware infrastructure for an entire airport require heavy investments. Also, PDR is the only solution that does not require any app permissions for localization. This is an advantage, because users are less likely to use indoor localization when user permissions are required [19]. The accuracy of all methods is according to the requirements and usable for airport localization [48]. Proximity sensing seems to have the highest accuracy, but it does not allow absolute localization and the accuracy is only for a single access point. An overview of advantages and disadvantages of the basic localization methods can be found in table 2.1.

Table 2.1: Comparison of common localization methods.

<table>
<thead>
<tr>
<th>Localization method</th>
<th>Trilateration</th>
<th>Triangulation</th>
<th>Proximity</th>
<th>Fingerprinting</th>
<th>Pedestrian dead reckoning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>1.2m</td>
<td>4m</td>
<td>0.38m</td>
<td>3-5m</td>
<td>1.4m</td>
</tr>
<tr>
<td>Requires no external hardware</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Requires no off-line phase</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Infrastructure free</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Investment cost</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>None</td>
</tr>
<tr>
<td>User permissions required</td>
<td>Wi-Fi/Bluetooth</td>
<td>Wi-Fi/Bluetooth</td>
<td>Wi-Fi/Bluetooth</td>
<td>Wi-Fi/Bluetooth</td>
<td>None</td>
</tr>
<tr>
<td>Usable on smartphones</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Usable at airports</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

2.2. Advanced indoor localization techniques

To improve the localization accuracy, more advanced indoor localization techniques are developed. These methods use advanced refinement techniques or a combination of different localization methods. An overview can be found in table 2.2. Chen et al. proposes a sensor fusion framework, which combines Wi-Fi, PDR and landmarks [13]. A Kalman filter is applied for refinement. The proposed system was implemented on an Android smartphone and an accuracy of 1 meter.

Ubicarse emulates an antenna array using a Synthetic Aperture Radar (SAR) implementation, to identify the spatial direction of incoming RF signals [32]. Ubicarse uses regular Wi-Fi access points to perform triangulation to improve noisy sensor data from accelerometer, magnetometer and gyroscope. et al. combine RF localization with stereo-vision algorithms to localize objects that have no RF source. It is implemented on a tablet and a median error of 39 cm is achieved.

Chronos is a system that uses a single Wi-Fi access point to localize users. Chronos uses an algorithm that computes the time of flight using off-the-shelf Wi-Fi cards. The time-of-flight and the speed of light are multiplied and an access point, with multiple antennas, computes the distance between each antenna and the client. Chronos uses optimization techniques to improve accuracy, such as eliminating packet detection delay, separating the propagation delays of different signal paths and phase offset correction. Chronos achieves an accuracy of up to 0.65 meters.

SmartPDR is another indoor localization system based on pedestrian dead reckoning [29]. SmartPDR tracks users without the need of additional devices or infrastructure. The system was implemented on off-the-shelf smartphones and the performance of the system was tested in several buildings. SmartPDR achieves an accuracy of 1.62 meters. SEAMLOC is a fingerprinting based localization method and is based on the creation of a database of Wi-Fi RSSI data at pre-chosen calibration points[43]. It uses a novel interpolation algorithm which allows for up to four times fewer calibration points. Location refinement is achieved by interpolating between access points using probabilistic Bayesian functions in which a system of two non-linear equations are solved. SEAMLOC achieves an accuracy of up to 1.8 meter.

SAIL, short for Single Access point Indoor Localization, is an indoor localization system that uses a single ac-
cess point and calculates the distance between a client and an access point by using the propagation delay between them [35]. This distance is combined with dead reckoning on a smartphone. The system is tested on a smartphone with 10 different users and an accuracy of up to 2.3m is achieved. ZEE, is an indoor localization system that makes the off-line phase of fingerprinting zero effort, by using a crowd-sourced PDR approach in which the Wi-Fi fingerprints are collected by smartphone users [42]. ZEE uses a particle filter and the particles are initialized based on the collected fingerprint data, to enable faster convergence. an overall accuracy of 5 meters is achieved and in 80% of the time an accuracy of 2.3m is achieved. Gu et al. present an indoor localization technique that uses compressive sensing for recovering absent fingerprints [22]. Missing entries in the fingerprint matrix are found using a K-Nearest Neighbour algorithm. It is shown that all fingerprint data can be recovered with an error rate below 6.6%, by collecting only half of the fingerprints. An overall accuracy of less than 10 meters is achieved. WaP, short for Wi-Fi assisted Particle filter, is an indoor localization system that combines fingerprinting with pedestrian dead reckoning [24]. WaP requires the floor plan and the locations of the access points reside are required. An accuracy of 0.71 meters is achieved with WaP.

The current state-of-the-art does not address the problems introduced, when indoor localization is used at an airport. The discussed systems do not provide a solution for big, polygon based maps and do not support multiple floor levels. That is why these problems are solved in this thesis.

Table 2.2: An overview of advanced indoor localization systems.

<table>
<thead>
<tr>
<th>Localization system</th>
<th>Technique</th>
<th>Sensors</th>
<th>Refinement technique</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chen [13]</td>
<td>PDR, WPL</td>
<td>Accelerometer, magnetometer, gyroscope, barometer, Wi-Fi</td>
<td>Kalman, Landmarks</td>
<td>1.00m</td>
</tr>
<tr>
<td>Ubicarse [32]</td>
<td>Triangulation</td>
<td>Wi-Fi, accelerometer, magnetometer, gyroscope</td>
<td>SAR</td>
<td>0.39m</td>
</tr>
<tr>
<td>SmartPDR [29]</td>
<td>PDR</td>
<td>Accelerometer, gyroscope, magnetometer</td>
<td>none</td>
<td>1.62m</td>
</tr>
<tr>
<td>Chronos [49]</td>
<td>Triangulation</td>
<td>Wi-Fi</td>
<td>packet delay removal, multipath separation, phase offset correction</td>
<td>0.85m</td>
</tr>
<tr>
<td>SEAMLOC [43]</td>
<td>Fingerprinting</td>
<td>Wi-Fi</td>
<td>Naive Bayes, interpolation</td>
<td>1.80m</td>
</tr>
<tr>
<td>Sail [35]</td>
<td>PDR, triangulation</td>
<td>wi-fi</td>
<td></td>
<td>2.3m</td>
</tr>
<tr>
<td>ZEE [42]</td>
<td>PDR, fingerprinting</td>
<td>accelerometer, magnetometer, gyroscope, Wi-Fi</td>
<td>particle filter</td>
<td>5.00m</td>
</tr>
<tr>
<td>Gu [22]</td>
<td>Fingerprinting</td>
<td>Wi-Fi</td>
<td>KNN</td>
<td>10m</td>
</tr>
<tr>
<td>WaP [24]</td>
<td>PDR, fingerprinting</td>
<td>Wi-Fi, accelerometer, magnetometer, gyroscope</td>
<td>Particle filter</td>
<td>0.71m</td>
</tr>
</tbody>
</table>
2.3. Enterprise solutions

In this section, enterprise localization solutions currently used by airports are discussed.

2.3.1. Indoor Atlas

Buildings have predictable magnetic fields caused by machinery, wiring and other magnetic interferences. Indoor Atlas used this magnetic noise to fingerprint an entire building and can achieve an accuracy of less than 3 meters when combined with Wi-Fi or Bluetooth fingerprinting and trilateration. Fingerprinting disturbances of the Earth’s magnetic field was researched by Chung et al. in [14], in which they achieve an accuracy of 1 meter.

2.3.2. Polestar

Polestar is the current global leader and pioneer of indoor localization in industry. Since it was created in 2002, it has applied its indoor technology in numerous indoor environments, including full coverage of Charles de Gaulle and Schiphol Airport. Polestar uses Wi-Fi or Bluetooth fingerprinting for localization. At Schiphol airport, 2000 beacons were placed throughout the airport to provide indoor localization in their app using the polestar SDK.

2.3.3. Apple Core Location

Apple Core Location is an indoor localization technology developed by Apple specifically for their iOS smartphone operating system. Apple Core Location uses Wi-Fi fingerprinting and works directly in their default maps app. Core Location takes over from GPS, in areas where Apple Core Location is available. Apple is working hard to fingerprint as many airports as possible. During this research Apple already has all major airports fingerprinted, including Schiphol airport. For this reason, Apple Core Location is initially the preferred method for indoor localization in the KLM app.

2.3.4. Accengage

Accengage is a company that uses proximity sensing for localization and is used for sending push messages to smartphones[4]. They use either GPS, where they use geofencing, to detect if a smartphone is in a certain area, or Bluetooth beacons. Their SDK can be implemented into a smartphone app, such as the KLM app and when the user is near a beacon or inside a geofence, the app sends a push message to the user.

2.4. Queueing

This section discusses the current state of the art in queueing time estimation. Corporate implementations and the latest research are discussed.

2.4.1. Bliptrack

For airlines and airports it is important to know the waiting times at choke points such as security and passport control[7]. Airports can manage the throughput through the airport and airlines can calculate the travelling times at the airport. Bliptrack was developed for this reason. BlipTrack is a system which is used by many major airports, such as Schiphol, JFK, Cincinnati airport and Edinburgh Airport and measures queue time, capacity and throughput. BlipTrack places sensors at strategic points in a specified area, in the case of airports at the beginning and end of queues. The sensors detects Wi-Fi or Bluetooth devices, such as mobile phones and tablets. By re-identifying devices again at the end of the queue, the waiting times can be calculated.
2.5. Indoor mapping

Mapping is an important part of localization. Without a map, a location provided by an indoor localization algorithm is just a coordinate. A map provides a visual representation of the current location to the user. Furthermore, map data is required to determine if a location is valid or invalid, by using a particle filter for example. This section explores the current state of the art in both enterprise solutions and research related topics.

2.5.1. Locus Labs

Locus Labs focuses on the venue map generation, maintenance, and user experience of location based services in smartphone apps. They deploying indoor location at scale, such as airports. They use a reality capture work flow to map large environments. Locus labs uses Light Detection and Ranging (LIDAR), which is a remote sensing method that uses light from of a pulsed laser to measure distances and depths, combined with vSlam [30], which is an algorithm for simultaneous localization and mapping. Simple web and mobile tools enable them to keep the content up-to-date. They integrate several indoor positioning solutions as well, which are deployed based on the customer’s requirements. They use a hybrid solution that uses fingerprinting of Wi-Fi, or magnetic interference from Indoor Atlas. They also use trilateration of Bluetooth beacons for background monitoring.

2.5.2. Google indoor maps

Google indoor maps is a project by google to integrate indoor maps into their google maps platform. Google integrates popular building such as airports, malls and stadiums into their global maps. The drawback of this platform is that Google becomes the owner of the map data and airports are not willing to hand over their map data due to security reasons. The airports available in Google indoor maps are mostly crowd sourced by users and are not always accurate [48].

2.5.3. Apple venue maps

Apple venue maps is used by apple to aid the apple core location on the iOS platform. Apple manually translates building schematics into their apple venue format, which is based on the GeoJSON format. The main advantage of Apple venue maps is that the map data is owned by the building owners and only they are allowed to distribute the data, which makes it an ideal solution for airports, since they want to make their map data public.
System Design

In this section, the system overview of the localization algorithms and the smartphone applications are discussed. In the introduction, the reasoning behind the chosen localization method is discussed. In Section 3.2, the architecture of the localization method is explained and in Section 7.1, the smartphone hardware is discussed.

3.1. Introduction

Due to the advantages mentioned in Section 2.1.6, a pedestrian dead reckoning method was chosen as the indoor localization solution for airlines. A PDR approach does not require any infrastructure or time consuming data collection phase, like in fingerprinting. The only thing that is required is the map, which is needed anyway to show the user’s location. In this way the presented indoor localization method minimizes the amount of dependencies required and improves the deployment at multiple airports. The system is called AirLoc. AirLoc requires a motion model, which analyses the sensor data to determine the motion of the user. For positioning refinement, a particle filter was chosen. The three pillars shown in Figure 2.1 of AirLoc are the motion model, pedestrian dead reckoning and the particle filter. The three pillars of AirLoc are shown in Figure 3.1.

3.2. AirLoc architecture

AirLoc is structured according to the three pillars mentioned in Section 2.1. The localization method used is pedestrian dead reckoning. As mentioned in Section 2.1.5 The sensing part consists of an accelerometer,
gyroscope, magnetometer and barometer. These sensors are fed into the motion model. The motion model analyses the motion of the user and consists of step detection, heading direction and step length estimation. The motion model is discussed in chapter 4. A particle filter is used as position refinement combined with landmarks. Regular PDR localization methods use simple maps of office. AirLoc uses complex polygon based maps based on GeoJSON data. Furthermore, AirLoc uses landmarks to recalibrate the particle filter and to change floors. An overview of AirLoc can be found in figure 3.2. When a floor change landmark is triggered, the particles are redistributed on the new polygon on the other floor. When the location is in the area of queueing area, the queue detection is enabled and calculates the waiting time of that queue. This information is useful for KLM to accurately estimate the walking times for other passengers that wish to know their walking times. The estimated location and waiting time is then used in the app to provide location based services to the user.

![Figure 3.2: An overview of the indoor localization architecture of AirLoc.](image)

### 3.3. Location based services

The localization system is developed for KLM in order to provide location based services. Location based services available in the system include waiting time estimation, airport localization and airport navigation. The concepts provided in Figure 1.1 are used to create the app. The app supports indoor navigation shown in Figure 1.1(b) and the passenger journey shown in Figure 1.1(c). The passenger journey is the journey from the starting point of the journey all the way to the point of departure. During the journey, different milestones need to be passed until departure. These milestones include; arriving at the airport, passing security, reaching the gate and departure. A passenger at an airport can have different passenger journey's. The assumed journey in this project is departure at the Shengen zone. The Schengen zone is the area that includes 26 European countries that have abolished passport and any other type of border control at their mutual borders. If a passenger travels to a Schengen country, no passport control milestone is passed, so that is why the passenger's journey is different when travelling to a non-Schengen country. Other passengers journey's include transfer from and to a Shengen or non-Shengen country and arrival from a Shengen or non-Shengen country. The airport milestone is reached when the user enters a geofence around the airport. This geofence includes a distance measurement. If the user is less than 2 KM away from the center of the airport, the airport milestone is reached. When the x-ray landmark, discussed in Section 6.2, is triggered, the security milestone is triggered and when the user is less than 10 meters away from the gate, the gate milestone is reached. When the aircraft departs the departure milestone is triggered. This can be done by detecting a change in air pressure, but is not implemented in this system. Figure 3.3(a) shows the passenger journey in the app. Airport navigation is provided by showing the map of the airport. Point to point navigation is required by
3.4. Queueing time

Airlines cannot install any infrastructure at an airport that is why solutions like bliptrack cannot be used by airlines. For this reason an infrastructure free method to measure queueing time was developed. This method uses activity monitoring to determine the current activity of the user. The activity monitor monitors the lower level walking patterns Walking and Standing. The time between these activities determines the higher level activity monitoring. When a walking time of less than 3 seconds is recorded and when a standing time of more than 3 seconds is recorded it is recognised as a potential queue. When more than 5 potential queues are detected, all potential queues are recorded as the activity Queueing. The queue is ended when, a Walking activity of more than 5 seconds is recorded, or when the x-ray landmark is triggered. Landmarks are discussed in Section 6.2. This data can then be provided to other smartphone users, to provide them with accurate walking times. A visual overview of the queue length estimation principle is shown in Figure 3.4.
The motion model

This chapter discusses the motion model. First the sensor orientation principle is explained. After that each piece of the motion model is discussed separately.

4.1. Introduction

A complete motion model on a smartphone should consist of three parts. Step detection, walking direction estimation and step length estimation. A smartphone contains sensors that can measure motion and direction. The Nexus 5X smartphone contains an accelerometer, gyroscope, magnetometer and barometer, which are used for the motion model. As shown in section 7.1, the accelerometer, gyroscope and magnetometer are 3 axis sensors. These local coordinates must first be transformed into global coordinates, because we are interested in the motion in global coordinates. Solutions such as in [47] have fixed sensors, so both the orientation and the heading direction of the sensor is fixed. A smartphone can be orientated into any direction and the heading direction can also be in any direction. For this reason, finding the heading direction is challenging. The step length or stride length provides the biggest challenge in the motion model. The step length of a person is based on his length and movement speed. Since the smartphone is never on a fixed position on the body, a static solution will not be enough. The accelerometer was positioned on the ankle and the step length was estimated by integrating the acceleration twice in [20]. For smartphones, a different solution must be found, because a smartphone can be in any position and orientation, such as in a chest pocket, pants pocket, or in hand.

4.2. Sensor orientation

The accelerometer, gyroscope and magnetometer are 3-axis sensors as shown in Table 7.1. These are the local coordinates of the smartphone and are shown in figure 4.1. For the motion model we are interested in the sensor data relative to the world coordinates, because this can provide us with data about the direction of motion. For this reason, a coordinate transformation must be performed in order to transform the local coordinates of the smartphone into these world coordinates represented in Figure 4.1. This is done by using a rotation matrix. The gravity vector $\vec{G} = [G_x \ G_y \ G_z]^{-1}$ is extracted from the accelerometer data and the magnetic north vector $\vec{E} = [E_x \ E_y \ E_z]^{-1}$ is acquired from the magnetometer data. In order to construct the world coordinate system, a third vector pointing east is constructed by taking the cross product of the magnetic north and gravity vector $\vec{H} = \vec{E} \times \vec{A}$ is constructed to make sure the gravity and magnetic north vector are perpendicular; $\vec{M} = \vec{G} \times \vec{H}$. Both vectors $\vec{H}$ and $\vec{M}$ are normalized. The rotation matrix $R$ is presented in Equation 4.1.

$$R = \begin{bmatrix} \hat{H}_x & \hat{H}_y & \hat{H}_z \\ M_x & M_y & M_z \\ G_x & G_y & G_z \end{bmatrix}$$  \hspace{1cm} (4.1)
4. The motion model

Table 4.1: left: smartphone axes in local coordinates. right: axes in world coordinates.

4.3. Step detection

Step detection is done by using the accelerometer of the smartphone. Raw accelerometer data is gathered, the gravity component of the acceleration is extracted and multiplying the rotation matrix $R$ with the gathered accelerometer data, the downward acceleration on the $w2$ axis can be measured, independent of the orientation of the smartphone. The downward acceleration is then processed to determine if a step was taken. The frequency of the accelerometer data is 50 Hz. To minimize false positives due to noise, a low-pass filter is applied so that no more than 3 steps can be recorded per second. Step detection based on the accelerometer has been thoroughly investigated in [17] [47] [3]. The step detection is based on a peak detection combined with zero crossing. The initial peak must be at least $4.5 \text{ m/s}^2$. The signal must reach zero no more than $250 \text{ ms}$ after the peak was detected. If this is the case, a step is detected. This is shown in Figure 4.1.

Figure 4.1: Overview of the step detection.
4.4. Step length estimation

Ten test runs were performed where hundred steps were taken both when the phone was in the hand and inside the pocket. On average in the hand 97 steps were recorded and in the pocket 103 steps were recorded on average.

4.4. Step length estimation

To determine the distance travelled, the distance after each step should be determined as accurately as possible. The step length changes every time, so the step length should be determined dynamically. In [40], different step length methods are compared and the scarlet method, shown in Equation 4.2, is shown to be the best dynamic method. In this equation, \( SL \) is the step length, \( k_s \) is the calibration factor, \( N \) is the amount of samples recorded between each detected step, \( a \) is the magnitude of a recorded acceleration sample.

\[
SL = k_s \cdot \sum_{k=1}^{N} \frac{|a_k| - a_{\text{min}}}{a_{\text{max}} - a_{\text{min}}} \quad (4.2)
\]

The drawback of this method is that it requires a calibration factor \( k_s \) which should be determined experimentally. For this system to work, the calibration should be done automatically. To determine the calibration factor, the static step length method shown in Equation 4.3 is used. In this equation \( k_h \) is the conversion constant and \( H \) is the height of the user. Acceleration data of 5 consecutive steps are stored and the median step is said to be equal to the fixed step length determined.

\[
SL = k_h \cdot H \quad (4.3)
\]

The height \( H \) of a person is based on the average length of the user's nationality. The Android API can be used to find these user specific variables. Also apps with user information, such as the KLM app, already have access to this information. The KLM app can obtain the nationality and gender of the user. After testing with 10 different people, the constant \( k_h \) in Equation 4.3 is determined to be 0.314. Tests were performed with 10 different people of different heights and different nationalities. Using this method, an accuracy of 90% was achieved. In the system, the calibration factor \( k_s \) is determined outside in the calibration phase.

4.5. Heading estimation

This section explains how the walking direction of the user is estimated in the motion model. The problem finding the heading estimation is that the heading direction relative to the orientation of the smartphone must be found. The smartphone can be in any orientation relative to the direction of movement. The smartphone can be in a hand, pants pocket, chest pocket or bag for example.

4.5.1. Related research

In the ideal case, the smartphone only accelerates into the walking direction. The walking process, or gait cycle however, introduces a lot of noise, which makes it difficult to get the heading direction from the accelerometer signal. In [5] the unwanted peaks are filtered and the heading is then estimated by using Equation 4.4.

\[
\theta = \text{atan2}(V_x, V_y) \quad (4.4)
\]

In [33] this is extended to obtain the global walking direction. The compass and gravity vector are used to generate a rotation matrix, which transforms the axis of the phone into the global plane. In [44], an elaborate walking analysis is done. The gait cycle, which is the walking pattern of a human being, is analysed and it is found that the walking direction is found by sign changing and averaging the acceleration signal at the point just before the foot is placed on the on the ground. It also cancels magnetic interference from the compass data, to acquire a more accurate rotation matrix. In [26] the unique behaviour of the measured acceleration data during the stance phase of the walking cycle is used for detecting the direction of the user. uDirect is improved by developing a calibration process without requiring noisy compass readings in [16]. These papers focus on finding the heading estimation, but do not use them for localization. In [27], the heading estimation for a PDR approach is found by using the rotation of the gyroscope to find the heading direction. This paper assumes the initial direction and position is known, because they claim it can be found by looking at the last known GPS location and heading.
4.5.2. **Acceleration based heading estimation**

A heading estimation algorithm based on acceleration was developed based on the findings of the previous subsection. As proposed by Ayub *et al.* in [5], the unwanted peaks are removed with a low pass filter. The heading angle is based on Equation 4.4. Tests were performed while holding the phone in hand and in a fixed position. Even in this ideal position the error was at least 21 degrees. Putting the phone in the pants pocket produced no useful results. A more advanced algorithm was produced using the principles developed by Roy *et al.* in [44]. They claim that the heading direction can be extracted from a fixed moment in the gait cycle.

![Figure 4.2: Acceleration on three axis in world coordinates.](image)

Figure 4.2 shows the gait cycle in world coordinates. According to Roy *et al.*, the acceleration of the angle between the $w_1$ axis and $w_2$ should give the right heading direction at the area between 11.1 and 11.2 second. The acceleration at this point in the gait cycle however always is around 0 m/s$^2$ and the angle between the axis depends only on the noise. The heading angle could not be estimated using the suggested method.

4.5.3. **Relative heading estimation**

Since no accurate heading estimation algorithm based on acceleration could be created, other methods were investigated. Kang *et al.* show that the heading can be estimated by using the gyroscope in [29]. In their research the heading direction and the initial position is known. They state that the initial heading can be determined by the direction and location of the entrance. Finding the initial position is discussed in Subsection 4.5.4. Using the gyroscope for heading estimation introduces another problem, which is drift. Drift is the error that occurs in the estimated angle due to the integration of biased sensor signals. Due to this drift, the gyroscope was not used in AirLoc. AirLoc only uses the magnetometer orientation to get the orientation of the smartphone relative the magnetic north. A low-pass filter, shown in equation 4.5, was used to filter the high frequency noise caused by magnetic interference. In this filter, $\alpha$ is 0.25.

$$y[n] = (1 - \alpha)x[n] + \alpha \ y[n - 1]$$ (4.5)
4.5.4. Absolute heading direction
To find the absolute heading direction, the initial heading must first be found. This initial heading is the
direction the user is pointing at the start of the indoor localization. The easy assumption could be made that
the user is simply looking straight into the entrance of the door. The problem however is that the entrances
at Schiphol are all revolving doors and for this reason, the direction error can be up to 90 degrees using this
assumption. This is due to the fact that the user can enter and exit the revolving door with many different
angles. Furthermore, the localization could be enabled just before the user enters the revolving door, giving
a completely inaccurate heading angle.
For this reason a more suitable solution was found. During the calibration phase, samples from the GPS
sensor are used to determine the heading vector. This vector is normalized and shown in Equation 4.6.
\[
\vec{V}_g = \begin{bmatrix} \hat{V}_\phi \\ \hat{V}_\theta \\ 0 \end{bmatrix}
\] (4.6)

The smartphone is assumed to remain in the same position throughout the calibration and during the
localization. Due to this assumption, the global heading estimate can be transformed back into the local
coordinate system with Equation 4.7. In this equation \( \vec{V}_l \) is the heading direction vector relative to the orien-
tation of the smartphone and \( R^T \) is the transposed rotation matrix.
\[
\vec{V}_l = R^T \vec{V}_g
\] (4.7)

When the indoor localization system is initiated, The local heading vector \( \vec{V}_l \) is transformed back into the
global coordinate system by using the most recent rotation matrix.
\[
\vec{V}_l = \begin{bmatrix} \hat{V}_x \\ \hat{V}_y \\ \hat{V}_z \end{bmatrix}
\] (4.8)

The absolute heading direction \( \theta_n \) is then calculated by using Equation 4.9.
\[
\theta_n = \text{atan2}(\hat{V}_\phi/\hat{V}_\theta)
\] (4.9)
This chapter explains how the map is constructed and the necessary tools required to interpret the map data. Different coordinate systems are explained and the mathematical calculations required to calculate polygon properties.

5.1. Introduction

Finding the right map data of Schiphol airport has been a very difficult task. Due to the PDR localization method, the physical information of walls, doors and obstacles are required. For this reason, a simple figure of the airport could not be used for this research. Schiphol does not give their map data to outsiders, due to security risks. Finding the right map data for this research became an intensive but necessary task. Halfway through this research, Schiphol gave Apple permission to give KLM the map data they gathered for their core location platform. This map data, which is a collection of GeoJSON data is used for this research. The map data size is only 1 MB, which makes it ideal for the KLM app, since they want a small sized system. The map data is used in the localization system for the following features:

- Map rendering in the app.
- Particle distribution and movement.
- Collision detection.
- Floor change and recalibration.

5.2. Map rendering

After acquiring the necessary map data, the first essential task was constructing the airport map. No manual or SDK was provided, so the map renderer was reversed engineered in order to both display the data in the app and allowing the particle filter to interpret the map data. The map data consists of different files that each have their own function. The most important being the Buildings, Levels, Units, Fixtures and Openings. The Buildings file contains the outer contour of the airport. The Levels file contains the contour of each floor level. In the Units file each object inside the airport can be found. This includes Walkway, Room, Non-Public, Stairs, Restroom, Escalator, Elevator, Restaurant, Moving Walkway, Parking, Open to Below. The map contains no walls, but instead a wall is the boundary of each unit. Each entrance to a unit can be found in the Openings file and each obstacle in the map such as columns and baggage belts can be found in the fixtures file. The data is stored per floor level. Each floor level contains all contour data, units and fixtures. Each unit, fixture and opening has its own object in which all their properties and geometry are stored. All this information is needed for the indoor localization system. A single unit is shown in Listing 5.1.
5. Airport modelling

Listing 5.1: Code snippet of a single UNIT

```json
{
    "type": "FeatureCollection",
    "features": [
        {
            "type": "Feature",
            "properties": {
                "UNIT_ID": "U04133",
                "CATEGORY": "Escalator",
                "LEVEL_ID": "L001",
                "NAME": "null",
                "SUITE": "null",
                "RESTRICTED": "Restricted_admission",
                "SOURCE": "null",
                "ADDRESS_ID": "A00001"
            },
            "geometry": {
                "type": "MultiPolygon",
                "coordinates": [
                    [ [ 4.764990529000045, 52.307636984000055 ],
                      ... ,
                      [ 4.764990529000045, 52.307636984000055 ]
                ]
            }
        }
    ]
}
```

The map is rendered using the Mapbox SDK [34]. The Mapbox SDK is able to draw polygons on top of its own map data. Each map feature is drawn separately and therefore it is possible to give each feature a different colour. The colour scheme is based on the KLM design guidelines [1]. Table A.1 shows the colours given to each map item.

### 5.3. Coordinate systems

The system uses three different coordinate systems. WGS-84 coordinates, local coordinates and rijksdriehoek-coordinaten. The map data is in WGS-84 coordinates $(\psi, \varphi)$, which is the standard coordinate system used by GPS. All information from Schiphol, such as walking routes and points of interest are in rijksdriehoek-coordinaten, which is the coordinate system used by the Dutch government. Local coordinates, $(x, y)$ are used to calculate the properties of the polygon. The local coordinates are calculated by finding the minimum coordinate and calculate the distance between that point. The local coordinate system is used to calculate the properties of the polygon. The rijksdriehoek-coordinaten are approximated by using the approximation Equations in [28]. The system can transform coordinates from WGS-84 to rijksdriehoek-coordinaten and vice versa.

### 5.4. Map geometry

This section explains the geometric calculations that are done in order to acquire the properties from the map data that is needed for the localization system. All polygons used in this system are assumed to be non-convex and non-self-intersecting.
5.4. Point in polygon problem
For collision detection, it is crucial to know if a particle is inside a polygon or not. This problem is known as the point in polygon problem [25]. The point in polygon algorithm is based on the Jordan Theorem, which states that for each loop without self-intersection, each point is either inside the polygon or outside. An infinite ray is cast from the particle position into any direction and intersecting this ray with any edge of the polygon. The algorithm is a ray-casting algorithm. Each iteration of the loop, the test point is checked against one of the polygon’s edges. First, the algorithm checks if the point’s y coordinate is within the edge’s scope. After that, the algorithm tests if the point is to the left of the line. If that is the case, the line drawn rightwards from the test point crosses that edge. By inverting the returned Boolean whenever an intersection is found, the algorithm counts how many times the line crosses the polygon. If the line crosses an even number of times, then the point is outside, the Boolean returns false and if an even number, the point is inside and the Boolean returns true. The point in polygon algorithm is at the heart of both the particle distribution and collision detection. When a particle is distributed or moved to an invalid polygon, the particle is destroyed and redistributed.

![Figure 5.1: A graphical representation of the point in polygon algorithm. The point is inside if the amount of orange dots on each side of the red is an odd number.](image)

5.4.2. Distance
The Haversine method is a method to calculate the distance between two points on a sphere from their latitudes and longitudes and is therefore ideal to measure distances on the surface of the earth [38]. The Haversine method is shown in Equation 5.1 and is used to calculate the distance between objects on the map. The distance between polygons is measured by calculating the center point of the polygon and using the Haversine method to calculate the distance between the center points. $d$ is the distance between two points with longitude and latitude $(\psi, \varphi)$ and $r$ is the radius of the Earth, which is 6378.137 KM.

$$d = 2r \sin^{-1}\left(\sqrt{\sin^2\left(\frac{\psi_2 - \psi_1}{2}\right) + \cos(\varphi_1) \cos(\varphi_2) \sin^2\left(\frac{\psi_2 - \psi_1}{2}\right)}\right)$$

(5.1)

5.4.3. Area
The area of a polygon must be calculated in order to determine the weight of every individual polygon. The coordinates of the polygon are first transformed into local coordinates. The local coordinates are calculated by finding the minimum latitude and minimum longitude of the entire polygon and setting this coordinate as the $(0,0)$ position. The rest of the coordinates are acquired by using the Haversine method to calculate the distance between the center points. $d$ is the distance between two points with longitude and latitude $(\psi, \varphi)$ and $r$ is the radius of the Earth, which is 6378.137 KM. The area of a non-self-intersecting polygon, which has vertices $(x_1, y_1)$ until $(x_n, y_n)$, is calculated by formula 5.2.

$$A = \frac{1}{2} \left| \begin{array}{cc} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ \vdots & \vdots \\ x_n & y_n \end{array} \right|$$

(5.2)
5.4.4. Opening intersection
The openings file contains all openings such as doors and entrances to escalators or stairs. These openings can be restricted or non-restricted. In order to check if an opening is crossed, a line intersection algorithm was used. If indeed an opening crossed and it is non-restricted, the passenger is allowed to enter. To check this, a line intersection algorithm is used [2]. The line intersection works in three stages. In the first stage, it is checked if the bounding boxes of the two lines intersect. If the bounding boxes intersect, the algorithm moves to the next stage. In this stage, it is checked if line a intersects line segment b, by checking if the cross product is zero. The crossproduct has the characteristic that if it is zero, the two lines intersect. If the two lines intersect, it is checked if line b intersects line segment a. The algorithm checks if b crosses a by checking if the end points of b are on different sides of a. A visual representation of the line intersection algorithm is shown in Figure 5.2.

![Visual representation of the line intersection algorithm.](image)

5.4.5. Uniform distribution on a polygon
Each unit contains polygons which cover an area on the map. The particles required for the particle filter can only be distributed at possible locations and these include all units except for Non-Public and Open to Below. A uniform distribution of coordinates has to be created for every valid polygon. In order to do this two algorithms are constructed. In the first algorithm, shown in Figure 5.2, a random location between the minimum and maximum point in the polygon. The point in polygon algorithm is then used to check if the position lies inside or outside the polygon. Figure 5.3 (a) is a visual representation of this algorithm.

![Visual representation of the point in polygon algorithm.](image)

Listing 5.2: square distribution algorithm.
```
get max X and max Y and min X and min Y
get random position between min X and max X and between min Y and max Y
if point_in_polygon
    distribute particle on this location
```

To speed up the particle distribution, a second algorithm was constructed. To improve the probability that a location is inside the polygon, a line is constructed between two random points in the polygon. A random point on the line is chosen and to verify that the position is inside the polygon, the point in polygon algorithm is used. A visual representation is shown in Figure 5.3 (b), and the algorithm is shown in Listing 5.3.
Using the line distribution method, a speed up of 14.6% is achieved compared to the squared distribution algorithm. The drawback of this method is that the distribution is only uniform if the points that make up the polygon are also uniformly distributed.

(a) squared distribution.  
(b) line distribution.  

Figure 5.3: visual representations of distribution algorithms.
This chapter described the different refinement methods used in the localization system. First it describes how the particle filter is used. In the following section landmarks are discussed.

6.1. Particle filter

A particle filter was used in this system as a refinement method. In the particle filter each particle represents a possible location. The advantages of using a particle filter are threefold. Firstly, it is able to capture the error in both step length and heading estimation by moving all particles in a random direction based on the mean error of the motion model. Secondly it is able to refine the estimated position with collision detection. When a particle collides with a wall, it is destroyed and redistributed on a random location of an active particle. Thirdly, it is able to estimate a position after a series of particle movements, when no initial position is known.

6.1.1. Particle generation

The system can distribute particles in three different ways: random distribution over an entire map, random distribution on a single polygon and random distribution around a given position. When the initial position is unknown the system will distribute the particles over the entire airport. When the initial position is at an entrance, the particles can be distributed on the entrance. If the initial position is a given location, the system can distribute all particles around this point.

Classical particle generation generates particles inside squares. A value is taken between the minimal and maximum value of both x and y. In case of squares, this point will always be inside the map. In case of a complex map, a polygon can take any shape, so the particle can be distributed outside the polygon. That is why a point in polygon algorithm must be used to check if the generated particle is inside or outside the polygon. To make sure the particle distribution is uniform, the number of particles have to be evenly distribution over each polygon. To achieve this, the area of each polygon is calculated and the weight of each particle is determined by the percentage of the entire area each polygon covers. The particle distribution algorithm is shown in Listing 6.1.

```plaintext
int totalParticleCount
for each valid polygon calculate weight
for each polygon
    particleCount = totalParticleCount * weight
    while (i < particleCount)
        get randomPosition
        if randomPosition inside valid polygon
            distribute particle
            i++
```

Listing 6.1: particle distribution algorithm.
6.1.2. Particle movement

The output of the motion model is inserted into the particle filter. The total distance travelled $D_n$ and the heading direction $\theta_n$. The total distance travelled $D_n$ is the sum of the step length estimates. For each particle, a random value between the minimum mean and maximum mean is added to both the heading angle and the distance travelled, to take the error of the motion model into account.

\begin{align}
\Delta x_n &= D_n \cos(\theta_n) \quad (6.1) \\
\Delta y_n &= D_n \sin(\theta_n) \quad (6.2)
\end{align}

$\Delta x_n$ and $\Delta y_n$ must then be transformed back into WGS-84 coordinates. Equation 6.3 shows the resulting longitude. In this equation $\psi_n$ is the previous WGS-84 location, $\Delta x_{n+1}$ is the changed position in the $x$ direction and $r$ is the radius of the earth, which is 6378.137 KM. Equation 6.4 shows the resulting latitude position. In this equation $\varphi_n$ represents the previous WGS-84 location and $\Delta y_{n+1}$ presents the changed position in the $y$ direction.

\begin{align}
\psi_{n+1} &= \psi_n + \frac{180 \Delta y_{n+1}}{\pi r} \quad (6.3) \\
\varphi_{n+1} &= \varphi_n + \frac{180 \Delta x_{n+1}}{\pi r \cos(\frac{\psi_{n+1} \pi}{180})} \quad (6.4)
\end{align}

The system first checks to which unit the particle moved by comparing the category of the current unit with the previous unit. If the particle is still in the same unit, it checks if the particle crossed or is inside a fixture. If it is inside a fixture, it is destroyed and if it is not, the particle is moved to the new location. If the particle moved to a different unit, it checks if the particle crossed a non-restricted opening. The particle is then only allowed to move if the particle did not cross or is not inside a fixture. An overview of the collision detection rules is shown in Figure 6.1.

---

Figure 6.1: Graphical representation of the collision detection in the particle filter.
6.1.3. Obstacle Distance measurements

Calculating collisions for all particles with all obstacles every iteration is a very computationally intensive task. It is obvious that particles do not collide with obstacles that are very far away. In order to optimize the particle filter, the distance between the centre point of each obstacle and the mean position is calculated. The centre point is calculated by taking the average latitude and the average longitude of each polygon. The maximum distance in which obstacles are calculated is experimentally determined to be 30 meters. This distance is required to take both the travel distance and the obstacle length into account. This feature offers a significant speed up. At the ground floor the average computation time is 300 ms without distance measurement and 100 ms with distance measurement. At the first floor, the computation time is 1200 ms without distance measurements and 300 ms with distance measurement. This is a speed up of up to 4 times. Without this distance measurement, the system would be too slow to be used in practice, so obstacle distance measurements are an essential part in the particle filter.

6.1.4. Backtracking

When no initial position is known and the particles are distributed uniformly on the airport map, the particles will eventually converge at the current location. The path can then be backtracked to visualise the taken path. This is demonstrated by Klepal et al. in [31]. Each time a particle moves, the previous position of the particle is saved. Each particle has a parent ID variable, which connects the current particle to the particle in the previous step from which it originates. Using the parent ID in step \( n \), it can be determined which particles were not destroyed in step \( n - 1 \) to generate all the particles in step \( n \). The same principle can be used to determine which particles were not destroyed in step \( n - 2 \), to generate all the particles in step \( n \). This can be continued for each saved dataset to calculate the precise position in each moment of time. When further on in the process the particles converge closer, a backtrack function can be executed to calculate from which location the particles originate. A line is drawn on the map to each of these positions showing the path which was taken. When the initial position is known the taken path can also be backtracked. This is shown in Figure 6.2.

![Figure 6.2: Graphical representation of the backtracking algorithm.](image)
6.1.5. Distribute around a given location

When the initial position is known, the system can initiate the particle filter by distributing all particles around a given location. The particle locations are a random value in a circle with radius $\text{MAXIMUM\_RADIUS}$, which is the uncertainty. Each particle is distributed on a random position, where $r$ is a random value between 0 and $\text{MAXIMUM\_RADIUS}$ and $\theta$ is a value between $-\pi$ and $\pi$.

![Graphical representation of the particle distribution on a given location.](image)

6.2. Landmarks

This section explores the possibility to use landmarks as a way to refine the localization algorithm. Different airport specific landmarks are investigated and tested.

6.2.1. Introduction

Landmarks are specific sensor patterns, with known locations and for this reason they are identifiable in the environment. This allows for a more accurate localization than is possible with the localization algorithm. The reason for introducing landmarks is that the accuracy of PDR localization depends on the accuracy of the initial location estimation. A landmark can serve as an initial location estimation. Landmarks can also serve as a recalibration point for the PDR algorithm. The particles can be recalibrated around the location of the landmark. Additionally, landmarks are also able to reset the cumulative error introduced by incorrect output from the motion model, like a wrong heading estimation, incorrect step detection or step length estimation. Landmarks investigated in this section include, elevators, escalators, stairs and entrances. The sensor analyzer app was used to record sensor data.

6.2.2. Altitude change

Escalators, elevators and stairs are very prominent objects in an airport. Sensor data was gathered at these objects to see if landmarks could be extracted. It was suggested by Chen et al. in [13] that the upward acceleration could be recorded to detect an elevator or stairs. The sensor data did record an upward acceleration, however it was within the noise region. A simple hand movement could produce the same sensor data. For this reason, it was concluded that the acceleration could not be used to detect changes in altitude. The barometer data showed more promising results. A barometer is present in most modern smartphones, excluding cheaper models. Therefore, a barometer is a useful tool in this system. Figure 6.4 shows the change in air pressure for all possible ways to change floors; up the escalator, up and down the elevator and up and down the stairs.
If the air pressure changes with the floor level difference, the floor change is triggered. This difference is 0.60 mBar between the ground floor and the first floor. In order to differentiate between these five different options, a classification was made. This classification is based on the rate of change and is shown in Table 6.1.

Table 6.1: landmark change rates.

<table>
<thead>
<tr>
<th>Landmark</th>
<th>change rate [mBar/s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>elevator</td>
<td>$p &gt; 0.06$</td>
</tr>
<tr>
<td>stairs</td>
<td>$0.01875 &lt; p &lt; 0.06$</td>
</tr>
<tr>
<td>escalator</td>
<td>$p &lt; 0.01875$</td>
</tr>
</tbody>
</table>

### 6.2.3. X-ray detection

The security is the pivot point of an airport. Every departing passenger has to pass it and it is the entrance from landside to airside. Every smartphone has to pass the x-ray scanner and passengers that are airside on time will most likely make it to their flight. For these reasons, a landmark at the security could be extremely useful. Furthermore, The estimated waiting time can be determined, giving a more accurate time to gate estimate for other passengers, by triggering the end of the security queue.

To investigate the possibility of a landmark at an x-ray scanner, tests were done at Scarabee Aviation Group [45]. Scarabee maintains all x-ray scanners at Schiphol airport and other airports worldwide. At the time of these experiments they were testing the new Siemens 6040 x-ray scanner [46] that is now deployed throughout the airport. The test set-up is shown in Figure 6.5.
Figure 6.5: Test setup for X-ray scanner testing at Scarabee.

The magnitude of the recorded magnetometer data is shown in Figure 6.6. The figure shows three distinct peaks. The first two are rather small, but the third one changes from about $50 \mu T$ to about $160 \mu T$. According to Mensch [36], the magnetic signal is caused by the electronics at the side powering the x-ray scan.

Figure 6.6: Magnetic magnitude reading of a smartphone inside an X-ray machine.

The x-ray landmark is triggered when the difference between the maximum value is bigger than $150 \mu T$. 
the difference between the minimum value and the maximum value is bigger than $100\mu T$ and the minimum
and maximum occur with $500\,ms$. An overview of the x-ray landmark values is shown in Table reftab:xrayvalues.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peek value</td>
<td>$&gt; 150\mu T$</td>
</tr>
<tr>
<td>minimum difference</td>
<td>$100\mu T$</td>
</tr>
<tr>
<td>Maximum delay</td>
<td>$500,ms$</td>
</tr>
</tbody>
</table>

6.3. Other landmarks

Other landmarks were also investigated. Finding a landmark at the entrance, could also be very useful. That
is why sensor data was recorded while entering and leaving the airport. There was a slight difference in air
pressure while entering and leaving the airport. This difference was only $0.2\,mBar$ and it heavily depended
on the temperature outside. For this reason, a fixed change rate could not be extracted and no landmark
could be created. The next signal change that was investigated was the Wi-Fi and Bluetooth signal strength,
but due to the glass walls at the entrances of the airport, the change in signal strength was insignificant. For
this reason, Wi-Fi and Bluetooth signal strength could also not be used as an entrance landmark.

6.4. Floor change

For the system to work on an airport, it has to work on different floors. Indoor localization systems are usually
2 dimensional and no research on particle filters supporting multiple floors was found. Shen et al. created an
autonomous multi-floor indoor navigation system in which a drone can be localized by mapping the environ-
ment in 3d. The system can change floors in two stages. First a landmark is triggered. A classification is made
in order for the system to be able to differentiate between stairs, escalators and elevators. Next it is checked
if particles are on an escalator polygon. If this is the case, the system looks for the closest empty space on the
floor other floor. The map data does not contain any objects at the exit of an escalator or elevator, this
is an empty space. The distance between the center point of the escalator and the empty space on the other
floor are calculated and all particles are distributed on this empty space polygon. A graphical representation
of the floor change is shown in Figure 6.7. On the left, the particles move inside the escalator. An escalator
landmark is triggered and the system redistributes the particles on the empty space. This is shown on the
right.

![Figure 6.7: Visual representations of the floor change after an escalator landmark is triggered.](image)
Evaluation

This chapter evaluates the system. Test results of Landmarks and localization. The accuracy of AirLoc is shown and evaluated. All testing was done at the landside of Schiphol airport. Access to airside, through the security could not be provided by KLM. For this reason, the queueing time at security and the dynamic walking times suggested in this thesis could not be tested. Before each test, the magnetometer was thoroughly calibrated in order for the phone to provide the best magnetometer data.

7.1. Hardware overview

The KLM app works on both Android and iOS, but the suggested solutions was only implemented on the Android platform. The app will be developed for the Nexus 5X smartphone, because it was designed especially for development purposes and runs stock android, which means that there are no extra modifications or layers added on top of the operating system. This benefits the efficiency and testing capabilities of the smartphone, because these layers consume resources on the smartphone. The phone runs Android 7.1.1 Nougat, which is the latest version of Android at the time of the project. The Nexus 5X contains all sensors currently available in smartphones, a list of the relevant sensors, the model number of the chip and their properties are shown in Table 7.1.

<table>
<thead>
<tr>
<th>Model nr.</th>
<th>Sensor type</th>
<th>Axis</th>
<th>Output</th>
<th>power consumption.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bosch BMI160 [9]</td>
<td>Accelerometer</td>
<td>3</td>
<td>m/s</td>
<td>1.58mW</td>
</tr>
<tr>
<td>Bosch BMP280 [10]</td>
<td>Barometer</td>
<td>1</td>
<td>mBar</td>
<td>8.1µW</td>
</tr>
<tr>
<td>Bosch BMM150 [8]</td>
<td>Geomagnetic field sensor</td>
<td>3</td>
<td>µT</td>
<td>275µW</td>
</tr>
<tr>
<td>Qualcomm QCA6174 [41]</td>
<td>Wi-Fi</td>
<td>2</td>
<td>RSSI, UUID</td>
<td>227.7mW</td>
</tr>
<tr>
<td></td>
<td>Bluetooth</td>
<td>2</td>
<td>RSSI, UUID</td>
<td>39.6mW</td>
</tr>
</tbody>
</table>

The power consumption is shown in Table 7.1 is the power consumption for a sensor running in Normal mode, which is 50 hz for the accelerometer, gyroscope and magnetometer. The power consumption of Wi-Fi and Bluetooth is the power consumption during scanning. It can be seen from the table that using all internal sensors of the phone does not consume more power than using Wi-Fi or Bluetooth. Most sensors shown in Table 7.1 are used in this project. The accelerometer is used to detect steps and to estimate the step length. The magnetometer is used in the heading estimation and the barometer is used to detect changes in altitude. Wi-Fi and Bluetooth are not used in this system.
7.2. Landmark results

The floor change landmarks were tested by testing each landmark 10 times. Tests were performed at the ground floor and the first floor, because they were accessible for testing. The results are shown in Figure 7.1. The landmark is triggered when the change of air pressure is more than 0.6 mBar. Which landmark is triggered depends on the rate of change. Elevator triggers with \( p > 0.06 \text{ mBar} \), stairs trigger \( 0.01875 < p < 0.06 \text{ mBar} \) and escalator \( p < 0.01875 \text{ mBar} \). With an average of 0.62 mBar, this results in an elevator landmark triggering when the change occurs in less than 10 seconds, a stairs landmark is triggered when the rate of change is between 10 and 32 seconds and an escalator landmark triggering when the change occurs in more than 32 seconds. Figure 7.1 shows that all landmarks triggered within these boundaries.

![Air pressure landmark classification](image)

Figure 7.1: Air pressure landmark results.

The x-ray landmark was tested 30 times and triggered 28 times. The problem with the x-ray landmark are the false positives. As soon as a magnetic object is near the smartphone, the landmark is triggered. Different methods were tried to remove the false positives, such as a more complex algorithm that takes the smaller peeks into account. This did not change the amount of false positives. Due to the fact that no testing could be done at the airport security, the probability of a false positive could not be measured.

7.3. Localization results

When no initial position is known, the particles are randomly distributed over the airport. All particles move based on the motion model data. The ideal path on the airport map is a long straight line. The path shown in Figure 7.2 was taken. It took the particle filter 190 meters in order for the particles to converge on a single location, so the user has to walk at least 190 meters in order to get an estimated location. This takes to much time and for this reason an initial position is required in AirLoc. The initial position is estimated by calculating the distance between the current GPS location and the location the an entrance. When this distance is smaller than four meters, the particles are distributed around the location of the door with a distribution radius of four meters. Four meters was initially chosen, because when the user is less than four meters away from the airport building, the GPS location becomes unreliable and the actual location is somewhere in this four meter radius. This was tested and 8 out of 10 times, the particle filter was enabled when the user was inside this four meter radius. The radius was increased to 4.5 meters to achieve a 10 out of 10 result.
To test the accuracy of AirLoc, a fixed path was chosen to test different systems. On this path, the error between the actual position and the estimated location is tested. The tested path is shown in Figure 7.3a. First the Polestar solution used by the Schiphol app was tested. This is shown in Figure 7.3b. This localization system is based on Bluetooth fingerprinting. Due to the fact that the location has no direct relation to the map data, the location can be at impossible locations such as position 12. *The glass wall problem*, which is a problem which occurs when a user is near a glass wall, causes inaccuracy in fingerprinting, because the glass walls hardly block the Bluetooth signal. This problem is clearly visible at locations 6 and 7. The user is not able to cross the wall, though the location system does move to the other side of the wall. The minimum error during the test was 3.69 meters and the maximum error was 24.84 meters. An overview of the error at each tested location is shown in Table 7.2.

AirLoc was tested in two orientations. The first test was done while the phone was in hand. In this test the heading is determined by only by the orientation of the phone. The estimated hand held path is shown in Figure 7.4a. The path shows an error between point 4 and 5 and between point 8 and 9. This error is probably caused by the magnetic interference of the escalators. AirLoc estimates an accurate position at points 6 and 7, because the particles cannot cross the glass wall. The particle filter corrects the error caused by the magnetic interference between point 4 and 5. The minimum error during this test was 0.81 meters and the maximum error was 21.38 meters.

The next test was to test the system in the pocket. Due to practical reasons, the phone was held outside the pocket in the orientation it would be if it was inside a pocket. For this test, the heading vector $\vec{V}_l$ was first determined with a calibration process. In this process, 10 steps were taken into the same direction. The calibration process, is started as soon as the GPS location is stabilized. The angle and distance between the first step and the current step is calculated and transformed to the local coordinate system. The mean of these 10 estimates is used as the local direction vector $\vec{V}_l$. The error between the actual heading and the calculated heading is 30 degrees. The system was not implemented to run in the background of the phone and keeping the screen on inside a pocket, enabled the touch, causing the app to be closed. Instead, the phone was held next to the pocket and in the orientation it would be if it was inside a pocket. The heading vector for this orientation was determined in the calibration phase and was added to the system manually. The tested route is shown in Figure 7.4b. The path shows a clear error towards the right from point 2 to 3. The magnetic interference between 4 and 5 causes the system to head more to the left and due to the particles colliding against the shops, the system was able to recover back to point five. Between 4 and 5 and between 8 and nine, the error of
7. Evaluation

(a) Fixed path to test the localization accuracy.

(b) Polestar accuracy.

Figure 7.3

(a) AirLoc Hand-held path.

(b) AirLoc pocket path.

Figure 7.4: AirLoc backtracked results.
7.3. Localization results

The magnetic interference cancels out the error of the heading estimation. The contextual location is always correct. The comparison of errors of the three test results are shown in Table 7.2.

<table>
<thead>
<tr>
<th>Nr.</th>
<th>Error Polestar</th>
<th>Error Airloc hand-held</th>
<th>Error Airloc pocket</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>18.84</td>
<td>2.49</td>
<td>1.53</td>
</tr>
<tr>
<td>2</td>
<td>12.85</td>
<td>13.88</td>
<td>13.73</td>
</tr>
<tr>
<td>3</td>
<td>5.29</td>
<td>1.31</td>
<td>7.22</td>
</tr>
<tr>
<td>4</td>
<td>5.22</td>
<td>4.52</td>
<td>23.35</td>
</tr>
<tr>
<td>5</td>
<td>24.84</td>
<td>21.38</td>
<td>4.90</td>
</tr>
<tr>
<td>6</td>
<td>18.94</td>
<td>3.23</td>
<td>3.47</td>
</tr>
<tr>
<td>7</td>
<td>21.76</td>
<td>3.03</td>
<td>3.37</td>
</tr>
<tr>
<td>8</td>
<td>4.22</td>
<td>1.11</td>
<td>2.46</td>
</tr>
<tr>
<td>9</td>
<td>3.69</td>
<td>9.49</td>
<td>11.96</td>
</tr>
<tr>
<td>10</td>
<td>10.88</td>
<td>5.63</td>
<td>5.01</td>
</tr>
<tr>
<td>11</td>
<td>4.45</td>
<td>7.15</td>
<td>8.04</td>
</tr>
<tr>
<td>12</td>
<td>7.79</td>
<td>0.81</td>
<td>1.76</td>
</tr>
</tbody>
</table>

The accuracy of the three tests are shown in Table 7.3. The error is defined as the distance between the estimated position and the actual position. The accuracy is defined as the mean error.

<table>
<thead>
<tr>
<th>Localization method</th>
<th>Minimum error [m]</th>
<th>Maximum error [m]</th>
<th>Accuracy [m]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polestar</td>
<td>4.22</td>
<td>24.84</td>
<td>11.56</td>
</tr>
<tr>
<td>Airlock hand-held</td>
<td>0.81</td>
<td>21.38</td>
<td>6.17</td>
</tr>
<tr>
<td>Airlock pocket</td>
<td>1.53</td>
<td>23.35</td>
<td>7.23</td>
</tr>
</tbody>
</table>

The power consumption of both systems were also tested. During a single test, Polestar consumed 10% battery life and AirLoc only 5%. The battery capacity of the Nexus 5X is 10.3 Wh, so Polestar consumed 1.03 Wh and AirLoc only 0.515 Wh, which makes AirLoc twice as efficient as Polestar, while at the same time providing a better accuracy.

![Power consumption](image.png)

Figure 7.5: Power consumption of the two tested localization methods.
7.4. Queueing time results

The activity monitoring was designed to be activated as soon as the estimated location was in the area of the security. When the walking pattern shown in Figure 3.4 is detected, the queuing time estimation is started and when the x-ray landmark is triggered, the queue time ends. The calculated time can then be uploaded and used for others to get an accurate walking time estimate. Since no access to the security or airside could be provided by KLM, testing this system was not possible. That is why the queueing time and walking time estimation moved to the background of this project. The queueing time system was demonstrated and tested at Scarabee. These results were gathered to show that the system works, rather than to show the accuracy of the system. At that time it was assumed that access to airside would be provided. The gathered results are shown in Table 7.4. During this test a queue was simulated near the x-ray scanner. The waiting time is the actual time that was simulated and the measured waiting time is the time that was calculated by the app. The simulated time was measured from the start of the queue until the phone was taken out of the x-ray scanner.

<table>
<thead>
<tr>
<th>Nr.</th>
<th>Waiting time [s]</th>
<th>Measured waiting time [s]</th>
<th>Landmark triggered</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>316</td>
<td>331</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>321</td>
<td>333</td>
<td>✓</td>
</tr>
<tr>
<td>3</td>
<td>319</td>
<td>332</td>
<td>✓</td>
</tr>
<tr>
<td>4</td>
<td>327</td>
<td>341</td>
<td>✓</td>
</tr>
<tr>
<td>5</td>
<td>312</td>
<td>326</td>
<td>✓</td>
</tr>
</tbody>
</table>

The landmark is triggered when the peak is detected and the queueing time is ended when the phone is taken out of the machine. In Figure 6.6 it can be seen that the time between the peak and the moment the phone is out of the machine is about 9 seconds. That is why the error is always at least 9 seconds. Other causes of error could be that the initial potential queue is not registered properly. The mean error is 14 seconds. The result show that the system works and that it can be used to measure the waiting time accurately. However, the system is heavily influenced by the false positives that are triggered by magnetic equipment such as laptops, that is why more testing should be performed at the airport security to test the feasibility of this system.
Conclusions and future work

In this chapter, the work is concluded and propositions for future work are discussed. Section 8.3 contains the take-home message of this project.

8.1. Conclusions

Let us revisit our important question with which we started this work: "which airport specific indoor localization method can be developed, that is ideal for airlines and can be applied at multiple airports?". In this thesis different localization methods have been investigated, including fingerprinting, trilateration, triangulation and pedestrian dead reckoning. This thesis states that an indoor localization system consists of three stages, the sensing, the localization technique and the optimization technique. A comparison is made between these different localization methods and pedestrian dead reckoning was chosen to be the most suitable solution for airlines, because they are infrastructure free, require no off-line phase and require no additional investment costs and no permissions are needed from the user to use the technique.

AirLoc was introduced as a system that is ideal for airlines. The system uses a motion model in the sensing stage, pedestrian dead reckoning as the localization technique and a particle filter combined with landmarks as the refinement technique. A pedestrian dead reckoning system that supports multiple floors can be created by using landmarks. The pressure change was used to switch to different floors and to recalibrate the particle filter. Testing has shown that a system can make a clear distinction between elevators, escalators and stairs. A particle filter was created that could be used on a complex polygon based map, by using polygon geometry and by creating special collision detection rules and optimizations such as obstacle distance measurements. Map data of different airports can be included to make the system work at multiple airports, but this was not tested.

A complete motion model was created that does step detection, step length estimation and heading estimation dynamically. The heading estimation can be used in different orientations by using an initialization stage in which the GPS heading is used to determine the walking direction relative to the orientation of the phone. This estimation has an error of 30 degrees, which is still high for a heading estimation, but a more accurate heading estimation could not be found. This error is visible in the test where the phone is in the orientation of the pocket. The particle filter however is able to partially correct this error. In order to improve the accuracy of AirLoc, the heading estimation must be improved.

AirLoc was compared to the Polestar implementation that is used in the Schiphol app. The system was tested in two orientations, hand-held and in a pocket. Polestar has a minimum error of 4.22 meters, a maximum error of 24.84 meters and a mean error of 11.56 meters. Airlock hand-held has a minimum error of 0.81 meters, maximum error of 21.38 meters and a mean error of 6.17 meters. Airlock in pocket orientation has a minimum error of 1.53 meters, maximum error of 23.35 meters and a mean error of 7.23 meters. This shows that the infrastructure free method can give better results as compared to an infrastructure based fingerprinting method. The most important difference between AirLoc and Polestar is that AirLoc is always in the correct area, while Polestar is not, due to the glass wall problem and due to the fact that the location of Polestar is not related to the map data. AirLoc cannot move through walls or give a location estimate at an unreachable location thanks to the collision detection of the particle filter. This makes AirLoc more suitable.
for location based services such as airport navigation. When a location on the wrong side of a wall is estimated, the airport navigation system will show the incorrect route, which is not possible with AirLoc. Power consumption of the two systems were also compared. During the test run Airloc consumed 0.515 Wh and Polestar consumed 1.03 Wh, making AirLoc twice as efficient.

A system is proposed that calculates queuing times at the airport security by using activity monitoring combined with an x-ray landmark. Initial testing shows that the queuing time could be estimated with an error of 14 seconds. Since no access to the airport security could be provided, this system could not be tested in practice. Especially the false positives caused by magnetic objects could make this system less accurate.

### 8.2. Future work

The localization accuracy is heavily dependent on the motion model. More research on the motion model is required, especially on the heading estimation. Finding the correct heading direction of a smartphone user can make pedestrian dead reckoning a great localization method for the industry, since there are no additional costs involved.

The localization system itself also offers room for future work. In the current system, the initial position is assumed to be outside the airport. If the app is opened inside the airport the particles are distributed uniformly over the entire airport and a location can only be provided once the particles have converged. A solution for this is to collect Wi-Fi fingerprints while users walk around the airport and upload these fingerprints to a server. Users that open the app can receive an initial position based on these fingerprints. Such a system was demonstrated in [15], but was not implemented here due to a lack of time. More research should also be done in the hand-off procedure, which decides the moment GPS is turned off and indoor localization is enabled. This could improve the accuracy of the initial position. Furthermore, the moment when indoor localization is shut down and GPS is turned on should be investigated. Altitude detection, to instantly detect the current floor level can be combined with the fingerprint to determine the initial position. Airports have very accurate weather stations, thus this information could be suitable for this purpose.

There are also a lot opportunities for future work in landmark research. More landmarks should be investigated to improve accuracy. Moving walkways exist at almost every airport and investigating them could improve pedestrian dead reckoning since particles should move with the speed of the moving walkway once they enter these polygons. Also full body scanners could be an important landmark, because they could tell the airline the passenger is ready for boarding.

The current queue estimation could not be tested at the airport security, therefore, more research should be done to improve the accuracy of the queueing time and to remove the false positives from the x-ray detection. This information should be uploaded to a server, so it can be used by other passengers. The accuracy of such a system also provides room for future research.

More investigation should be done with respect to the map data. The provided map data did not provide all the required information. Exits of escalators and entrances should for example be added to be more suitable for pedestrian dead reckoning. Also more information about which polygons can be crossed and which cannot be crossed should be included in the data. Future map research also includes a way to divide airports into different zones. Crowd sourced data could also be used here.

Routing information for indoor environment should be further investigated. The current routing information is inaccurate and does no show an obvious walking path, but rather the shortest path. A system should be created that can dynamically change this routing information when the airport map changes.

### 8.3. Localization vs. context

During this research I found that all problems that require input from the environment are treated as localization problems, while actually there are two types of problems; localization based problems and context based problems. When the current walking time to the gate must be shown in the app, this is a localization problem, because it requires the current location of the user to calculate the walking time to the gate. However, when a push message is send to the user for a discount only when he is seated, the solution is to place beacons at every seating location, without monitoring the activity of the user. This problem is treated as a localization problem, but is in fact a context based problem, because the app should detect when the user is seated. The
best example of such a problem comes from the NS, the Dutch national railway company. They wanted to remind their passengers to "check out" their public transportation cards by placing Bluetooth beacons in every "check out pole" [11]. The problem they introduced is that when passengers are not in the proximity of such a pole, no push message is send. This problem is therefore treated as a localization problem, while in fact this is a context problem. The context here is that a passenger is leaving the train and should therefore be reminded to "check out". Instead of measuring Bluetooth beacons the NS app should monitor if the user exits the train, by measuring the magnetic interference of the train for example. This can improve accuracy, save the NS a lot of money, does not require a permission from the user to enable Bluetooth and saves energy, because no Bluetooth scanning is required. Even though localization and context are two different problems they often go hand in hand. When a passenger for example should be given a discount to business class when he is waiting at the gate, this is both a context and a localization problem. The localization problem here is; Is the user at his gate? The context problem is; is the user seated? In this project landmarks are used to optimize localization and change of floors. The power of landmarks is that they offer both a context and a location. In case of the x-ray landmark, the context is; going through security. The location of the x-ray scanner is fixed on the map. More research on these context aware systems is needed.
Figure A.1: Ground floor.
Figure A.2: First floor.
Figure A.3: Second floor.

Figure A.4: Third floor.
Figure A.5: Basement.

Table A.1: Overview of colours used for each entry in the map

<table>
<thead>
<tr>
<th>entry</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walkway</td>
<td>■■■</td>
</tr>
<tr>
<td>Room</td>
<td>■■■</td>
</tr>
<tr>
<td>Non-Public</td>
<td>■■■</td>
</tr>
<tr>
<td>Stairs</td>
<td>■■■</td>
</tr>
<tr>
<td>Restroom</td>
<td>■■■</td>
</tr>
<tr>
<td>Escalator</td>
<td>■■■</td>
</tr>
<tr>
<td>Elevator</td>
<td>■■■</td>
</tr>
<tr>
<td>Restaurant</td>
<td>■■■</td>
</tr>
<tr>
<td>Moving Walkway</td>
<td>■■■</td>
</tr>
<tr>
<td>Parking</td>
<td>■■■</td>
</tr>
<tr>
<td>Open to Below</td>
<td>■■■</td>
</tr>
<tr>
<td>Fixture</td>
<td>■■■</td>
</tr>
<tr>
<td>Opening - non restricted</td>
<td>■■■</td>
</tr>
<tr>
<td>Opening - restricted</td>
<td>■■■</td>
</tr>
</tbody>
</table>
Smartphone apps

(a) Passengers journey.

(b) Indoor navigation.
52 B. Smartphone apps

(a) Sensor analyzer.

(b) Motionmodel test app.
Listing C.1: Input JSON data for route information

```json
{
  "features": [
    {
      "geometry": {
        "x": "529577",
        "y": "6856570",
        "z": "10"
      },
      "attributes": {
        "Name": "Current_position",
        "RouteName": "Route_A"
      }
    },
    {
      "geometry": {
        "x": "529577",
        "y": "6856570",
        "z": "10"
      },
      "attributes": {
        "Name": "Gate_G02",
        "RouteName": "Route_A"
      }
    }
  ]
}
```

53


[41] *802.11ac Wi-Fi 2x2 MIMO Combo SoC*. Qualcomm, 9 2014. 1.0.


