Smartphone-based Indoor Positioning Using Wi-Fi Fine Timing Measurement Protocol

Master of Science Thesis
University of Turku
Department of Future Technologies
Communication Systems
2019
Sami Huilla

Supervisors:
Ethiopia Nigussie, Adj. Prof.
Nanda Thanigaivelan, M.Sc.(Tech.)

Advisor:
Seppo Horsmanheimo,
Principal Scientist

UNIVERSITY OF TURKU
Department of Future Technologies

SAMI HUILLA: Smartphone-based Indoor Positioning Using Wi-Fi Fine Timing Measurement Protocol

Master of Science Thesis, 69 p.
Communication Systems
September 2019

Location-based services have grown popular since GPS and other satellite navigation systems became available for consumers. However, because satellite signals are absent inside buildings, other means of positioning need to be used to enable similar services as outdoors. In the case of mobile phones, Wi-Fi received signal strength has been a widely studied option for positioning. Indoor environment is challenging, because the readings have significant fluctuation due to interference from walls, furniture and people.

Fine Timing Measurement (FTM) is a new addition to the IEEE 802.11 WLAN standard. It provides Wi-Fi positioning that relies on the time of flight of the signal instead of its received strength. Time of flight information is supposed to be more reliable compared to signal strength, providing more accurate distance estimates to be used in positioning. FTM is claimed to provide meter-level positioning accuracy.

In this thesis, the FTM protocol is introduced, and a smartphone positioning system is implemented. The system includes two alternative Android applications for recording and visualizing FTM data, and two algorithms for calculating position estimates. With an FTM-enabled smartphone and Wi-Fi access points, the positioning accuracy of FTM is evaluated with field measurements in two different office environments. Using an Unscented Kalman Filter algorithm, mean positioning error of 0.72 meters was achieved in a large, open room. In a more scattered AP constellation across multiple rooms, the mean error was 2.07 meters. The results show that meter-level positioning accuracy is possible with FTM, although here it was achieved with favourable AP placements around a single room. In the more realistic setting, room-level accuracy was achieved.

Keywords: Indoor positioning, Fine Timing Measurement, WLAN, UKF, Android

TURUN YLIOPISTO
Tulevaisuuden teknologioiden laitos

SAMI HUILLA: Smartphone-based Indoor Positioning Using Wi-Fi Fine Timing Measurement Protocol

Diplomityö, 69 s.
Tietoliikennetekniikka
Syyskuu 2019

Sijaintitietoon perustuvat palvelut ovat yleistyneet GPS:n ja muiden satelliittipaikannusjärjestelmien tultua kuluttajien käyttöön. Koska satelliittien signaalit eivät kantaudu rakennusten sisälle, vaihtoehtoiset paikannuskeinot ovat tarpeen samanlaisten palveluiden mahdollistamiseksi kuin ulkotiloissa. Älypuhelimien tapauksessa Wi-Fi -signaalinvoimakkuus on paljon tutkittu paikannuskeino. Sisätilat ovat haasteellisia, koska seinät, kalusteet ja ihmiset aiheuttavat merkittävää vaihtelua vastaanotetussa signaalinvoimakkuudessa.

Fine Timing Measurement (FTM) on uusi lisäys IEEE 802.11 WLAN -standardiin. Se mahdollistaa Wi-Fi -paikannuksen, joka perustuu signaalin vastaanotetun voimakkuuden sijasta sen lentoaikaan. Lentoaikatiedon oletetaan olevan signaalinvoimakkuutta luotettavampi ja tarkempi keino etäisyyksien arviointiin. FTM:n väitetään mahdollistavan metritason paikannustarkkuuden.

Tässä tutkielmassa käsitellään FTM-protokollan toiminta ja toteutetaan sitä hyödyntävä älypuhelimen paikannusjärjestelmä. Toteutus sisältää kaksi vaihtoehtoista Android-sovellusta FTM-datan keruuseen ja visualisointiin sekä kaksi algoritmia sijaintiarvioiden laskemiseen. FTM:n paikannustarkkuutta arvioidaan suorittamalla koemittauksia kahdessa erilaisessa toimistoympäristössä käyttäen protokollaa tukevaa älypuhelinta ja Wi-Fi -tukiasemia. Unscented Kalman Filter -algoritmilla paikannusvirheen keskiarvoksi saatiin 0,72 metriä isossa, avoimessa huoneessa. Asettelemalla tukiasemat harvemmin useaan huoneeseen keskivirheeksi saatiin 2,07 metriä. Tulosten mukaan metritason paikannustarkkuus on mahdollista FTM:n avulla, joskin tässä tapauksessa se saavutettiin suotuisalla tukiasemien sijoittelulla yhden huoneen ympärille. Todenmukaisemmalla sijoittelulla saavutettiin huonetason paikannustarkkuus.

Asiasanat: Sisätilapaikannus, Fine Timing Measurement, WLAN, UKF, Android

Acknowledgements

This thesis was done at VTT Technical Research Centre of Finland Ltd, based on the work done in projects 5G-FORCE and LuxTurrim5G funded by Business Finland.

Contents

List of Figures v

List of Tables vii

List of Acronyms viii

1 Introduction 1

1.1 Motivation 2

1.2 Thesis Statement and Research Questions 3

1.3 Objective of the Thesis 3

1.4 Thesis Structure 3

2 Indoor Positioning Techniques 5

2.1 Ranging 5

2.1.1 Received Signal Strength 6

2.1.2 Time of Arrival 7

2.1.3 Time Difference of Arrival 8

2.1.4 Round-Trip Time 9

2.2 Angulation 10

2.2.1 Direction Finding 10

2.2.2 Triangulation 10

2.3 Fingerprinting 11

2.4 Trilateration . . . 13
2.4.1 Nonlinear Least Squares . . . 14
2.4.2 Kalman Filter . . . 15
2.5 Fine Timing Measurement . . . 17
2.5.1 Discovery and Negotiation . . . 18
2.5.2 Measurement Exchange . . . 19
2.5.3 Android RTT API . . . 21
2.6 Related Work . . . 23

3 Implementation **25**
3.1 Access Points . . . 26
3.2 Mobile Device . . . 27
3.2.1 Range Measurements . . . 28
3.2.2 Location Visualization . . . 30
3.3 Positioning Algorithms . . . 31
3.3.1 NLS . . . 31
3.3.2 UKF . . . 35

4 Measurement Setup **41**
4.1 Reference Location . . . 41
4.2 Visual Analysis . . . 42
4.3 Sites . . . 42
4.4 Calibration Measurements . . . 43
4.5 Mobile Measurements . . . 46

5 Results and Discussion **48**
5.1 Ranging Accuracy . . . 48
5.1.1 Line-of-Sight . . . 49
5.1.2 Non-Line-of-Sight . . . 52

5.2 Positioning Performance 52
5.2.1 Site A: Favourable Environment with Plenty of LOS Conditions . 54
5.2.2 Site B: Challenging Environment with Mostly NLOS Conditions . 56
5.3 Discussion 59
5.4 Comparison with Other Technologies 60

6 Conclusions **63**
6.1 Future Work 64

References **65**

List of Figures

2.1 TDOA positioning 8
2.2 Triangulation in 2-D 11
2.3 Trilateration in 2-D 13
2.4 Fine Timing Measurement exchange example 20

3.1 Google Pixel 2 XL and Compulab FTM Responder 25
3.2 Android applications using Wi-Fi RTT 28
3.3 The ranging process 29
3.4 Native to JS interaction in FTM Demo App 30

4.1 Measurement robot 42
4.2 Test sites and their measurement areas 43
4.3 Device arrangement in the stationary measurement 44
4.4 Calibration measurement arrangement 45
4.5 Measurement route at site A 47
4.6 Measurement route at site B 47

5.1 Calibration range measurements to two access points 49
5.2 Ranging error in LOS conditions 50
5.3 CDF of absolute ranging error 51
5.4 Range measurements in partial NLOS conditions 53
5.5 Ranging error in partial NLOS conditions 53

5.6 Position estimates at site A 54
5.7 CDF of positioning error at site A 55
5.8 Position estimates at site B (raw range measurements) 57
5.9 Position estimates at site B (corrected range measurements) 57
5.10 CDF of positioning error at site B 58
5.11 FTM positioning performance against other technologies at site B 62

List of Tables

2.1 Timestamps captured during an FTM frame exchange 19
2.2 Information in a ranging result 22
4.1 Ranging offsets in the stationary measurement 44
5.1 Ranging error statistics in the calibration measurement 51
5.2 Positioning error statistics 60

List of Acronyms

AOA angle of arrival.

AOSP Android open source project.

AP access point.

API application programming interface.

BFGS Broyden–Fletcher–Goldfarb–Shanno.

BSSID basic service set identifier.

CDF cumulative distribution function.

CG conjugate gradient.

COBYLA constrained optimization by linear approximation.

DF direction finding.

DOP dilution of precision.

EKF extended Kalman filter.

FTM fine timing measurement.

GNSS global navigation satellite system.

HTML hyper text markup language.

JS JavaScript.

KF Kalman filter.

kNN k-nearest-neighbor.

L-BFGS-B limited-memory BFGS bound-constrained.

LBS location-based service.

LCI location configuration information.

LOP line of position.

LOS line-of-sight.

LTE long term evolution.

MAC media access control.

NGP next generation positioning.

NLOS non-line-of-sight.

NLS nonlinear least squares.

RSS received signal strength.

RSSI received signal strength indicator.

RTT round-trip time.

SLAM simultaneous localization and mapping.

SMP smallest m-vertex polygon.

SSID service set identifier.

STA station.

SVM support vector machine.

TDOA time difference of arrival.

TOA time of arrival.

ToF time-of-flight.

UDP user datagram protocol.

UKF unscented Kalman filter.

UWB ultra-wideband.

WLAN wireless local area network.

Chapter 1

Introduction

Location information is used in a large variety of applications by both industry and consumers. Location-based services (LBSs) such as navigation, geotagging and resource tracking, are growing more popular than ever. One key enabler for this is the emergence of global navigation satellite systems (GNSSs) including GPS, GLONASS, BeiDou and Galileo. Today, most smartphones have a GNSS receiver, which provides the device with a location estimate. While GNSS is a popular and rather reliable positioning method outdoors, its performance suffers from shadowing objects between the satellites and the receiver. Tall buildings, trees and indoor structures affect the satellite signal propagation in a way that causes problems in acquiring the signal [1].

The lack of GNSS signals indoors forces us to use alternate methods for positioning. Radio signals, inertial sensors, optical radars, or even images can be utilized when determining position inside a building. Due to their popularity and existing infrastructure, wireless communication technologies like Wi-Fi and Bluetooth have been widely researched as means of positioning. These signals can be utilized in multiple ways such as measuring the received signal strength (RSS), time-of-flight (ToF) or angle of arrival (AOA). By combining these measurements with basic trigonometry, the position of the wireless device can be determined.

Signal attenuation has been a rather widely studied approach to estimating the dis-

tance between two wireless devices. While in free space the signal path loss is proportional to the square of travel distance, obstacles both outdoors and indoors bring many additional variables to the equation. Consistent and accurate distance estimation is challenging because the received signal strength usually has fluctuation caused by reflections and attenuating objects such as walls, furniture and humans. Time-based range estimation is considered less prone to these challenges since the time-of-flight of a radio signal does not fluctuate as much as the signal strength. However, as radio signals travel effectively at the speed of light, the time-of-flight needs to be measured with a great precision in order to get accurate distance estimations. This causes hardware requirements that have been outside of large-scale commercial wireless devices like smartphones or Wi-Fi access points (APs).

Fine timing measurement (FTM) is a protocol introduced in the IEEE 802.11-2016 standard [2]. It specifies a way for two wireless local area network (WLAN) devices to perform round-trip time (RTT) measurements between each other. It is similar to the existing Timing Measurement protocol, but the measurements in FTM are more accurate, making them useful in indoor positioning. The Wi-Fi Alliance has established a certification for Wi-Fi Location, which is given to Wi-Fi products that are capable of using FTM. Google has included Wi-Fi RTT measurements as part of the Android operating system in 2018. Given the standardization and the support from Wi-Fi Alliance, hardware manufacturers and Google, FTM is a worthy option to study for indoor positioning.

1.1 Motivation

Accurate indoor positioning based on received signal strength is difficult because of multipath propagation and varying attenuation factors in the environment. In a typical office environment with a WLAN infrastructure, non-line-of-sight (NLOS) situations are common. That makes RSS-based ranging challenging.

With the addition of the FTM protocol to the 802.11 standard, a more accurate time-of-flight-based positioning solution for Wi-Fi devices is possible. In this thesis, an indoor positioning system utilizing FTM is implemented, aiming at accuracy and latency suitable for real-time navigation. FTM features of an Android phone and suitable access points are tested. The positioning performance is compared with other technologies.

1.2 Thesis Statement and Research Questions

It has been claimed [3, 4], that the FTM protocol enables meter-level accuracy in Wi-Fi-based indoor positioning, providing a new, feasible positioning solution that uses existing network infrastructure. The thesis will evaluate this claim by answering the following three research questions:

1. How does the FTM protocol work?
2. How does FTM-based positioning differ from other methods?
3. What is the positioning accuracy of an FTM-based indoor positioning system?

1.3 Objective of the Thesis

The main objective of this thesis is to implement an indoor positioning system that uses Wi-Fi Fine Timing Measurements. The system is capable of providing a real-time position estimate accurate enough for indoor navigation.

1.4 Thesis Structure

In chapter 2, the common techniques of indoor positioning are introduced. The chapter focuses on techniques that are essential in radio signal-based positioning. Description of the FTM protocol is also given. Chapter 3 introduces the parts of the implemented

positioning system and their functions. The measurement tools and test locations are discussed in chapter 4. In chapter 5, the results from ranging and positioning tests are introduced and compared with other technologies. The final conclusions of the work are presented in chapter 6.

Chapter 2

Indoor Positioning Techniques

Indoor positioning has been a widely researched topic since the emergence of GPS-based location services. Because GNSS signals have poor performance indoors, there is a continuous demand for an indoor positioning system as functional as GNSS. There is a large variety of possible technologies to use in determining location indoors, from sensors and cameras to radio signals. In this chapter, different methods used in indoor positioning are discussed, mainly concentrating on techniques involving radio signals.

2.1 Ranging

Ranging is the procedure of determining the distance between two wireless devices, e.g. a mobile phone and an AP. In radio-based positioning, the distance is usually derived from the time-of-flight of the signal or from signal strength [5]. When distances to three or more APs with known locations are known, a two-dimensional position estimate of the mobile device can be calculated using trilateration. Ranging techniques such as time difference of arrival (TDOA) provide the difference in distance to two transmitters, which can be used to determine the possible locations of the receiver.

2.1.1 Received Signal Strength

The distance between two radio antennas is possible to calculate when the transmitted and received power are known. Received signal strength (RSS) measurements have been widely used in positioning research. In free space the signal path loss is proportional to the square of the signal travel distance.

The formula for the free space loss is [6]:

$$L_{FSPL} = 20 \log \left(\frac{4\pi R}{\lambda} \right) \qquad \text{dB} \tag{2.1}$$

where R is the distance and λ is the wavelength. In indoor environments however, the path loss modelling becomes a more complex task. In addition to walls and furniture causing additional attenuation, corridors tend to reduce attenuation by guiding the waves [7]. To take these challenges into account, several indoor path loss models have been developed. The ITU Indoor Path Loss Model [8] introduces a distance power loss coefficient and a floor penetration loss factor. Tables for their values in different environments and frequencies are provided in the recommendation.

The total indoor path loss is calculated as follows [8]:

$$L_{IPL} = 20 \log_{10} f + N \log_{10} d + L_f(n) - 28 \qquad \text{dB} \tag{2.2}$$

where:

N: distance power loss coefficient

f: frequency in MHz

d: distance in meters

L_f: floor penetration loss factor

n: number of floors between the devices.

There are other, more complex indoor propagation models such as the log-distance path loss model, the Ericsson multiple breakpoint model, and the attenuation factor model [9]. By using an attenuation model, the distance between devices can be estimated from the RSS value [7]. However in real situations, the indoor challenges such as reflections, obstacles, and interference cause the distance estimation to be difficult [7]. Because fitting an analytical propagation model to a real indoor environment is challenging, empirical calibration can be done in addition [7]. Also, instead of mapping RSS values to distance, signal strength information can be used for positioning in fingerprinting, which is discussed in section 2.3.

2.1.2 Time of Arrival

Time of arrival (TOA) is a time-based ranging method that uses the signal propagation time to calculate the distance between devices [10]. In one-way TOA, the transmitter sends a signal at a known time t_t, and the receiver measures the time of arrival t_a. The subtraction of the time of transmission from the time of arrival gives the propagation time. The distance d between the transmitting and receiving antennas is then calculated by multiplying the propagation time by the speed of light c:

$$d = (t_a - t_t)c \tag{2.3}$$

when sources of error are not taken into account [5].

In TOA, it is important that the transmitter and receiver clocks are synchronized, because the range calculation is based on timestamps that are recorded in both devices. However, there is always some offset between the clocks in practice, which causes error to the range measurements (one nanosecond offset results in 30 cm distance error). Additionally in positioning scenarios, there are multiple transmitters, and their clocks have to be synchronized [5]. Besides clock offsets and drifting, the positioning error is affected by propagation errors and the receiver's capabilities in detecting the reception of the signal [5].

2.1.3 Time Difference of Arrival

Time difference of arrival (TDOA) method is based on the difference in the signal arrival time at multiple receivers that lie in known locations [10]. After a transmitter at an unknown location has transmitted a signal, the receivers receive it at different points of time depending on their distance from the transmitter. Provided the receivers are time synchronized, the TDOA can be measured. Two-dimensional TDOA positioning, also known as hyperbolic positioning, is illustrated in figure 2.1. The TDOA measurements between two pairs of receivers, $\{A, B\}$ and $\{A, C\}$, form hyperbolic lines of position (LOP) for the transmitter P. The position of P is where the two LOPs intersect.

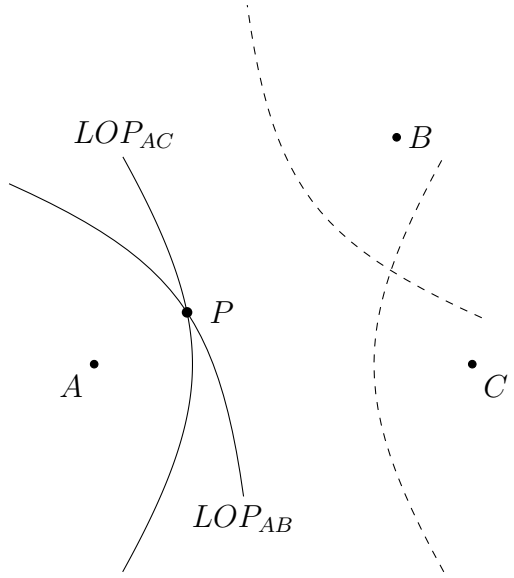


Figure 2.1: TDOA positioning

The time of arrival t_i in receiver i is

$$t_i = t_0 + \frac{d_i}{c} \tag{2.4}$$

where t_0 is the time of transmission at the transmitter, d_i is the distance from the transmitter to the receiver, and c is the speed of light [11]. By combining two of these equations, i.e. calculating the difference in the arrival times at two receivers (t_1 and t_2), the unknown

time t_0 is eliminated:

$$t_2 - t_1 = \frac{d_2 - d_1}{c} \tag{2.5}$$

This equation represents a hyperboloid that defines the (LOP) of the transmitter. Now, with the Pythagorean theorem, the equation can be represented using transmitter coordinates x_0,y_0,z_0 and receiver coordinates x_1,y_1,z_1 and x_2,y_2,z_2:

$$\begin{aligned} &\sqrt{(x_2 - x_0)^2 + (y_2 - y_0)^2 + (z_2 - z_0)^2} \\ -&\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2} = c(t_2 - t_1) \end{aligned} \tag{2.6}$$

The 3-D transmitter coordinates can be solved by combining three of such equations [11].

2.1.4 Round-Trip Time

Ranging by round-trip time (RTT) is done by measuring the time it takes for a message to travel to a receiver and back [7]. RTT is the time that passes between the transmission of an initiating signal and the reception of a corresponding response signal. The signal's time-of-flight (ToF) between the initiator and the responder is obtained by subtracting the turnaround time at the responder (τ) from the RTT, and dividing the result by two. Ultimately, the ToF is multiplied by the speed of light, resulting in the distance between the devices [5]:

$$d = \frac{1}{2}(t_{RTT} - \tau)c \tag{2.7}$$

RTT ranging has the advantage of omitting synchronization between devices. The round-trip time t_{RTT} of the signal is measured by only the initial transmitter, therefore time synchronization with the receiver is unnecessary. The value of τ can be defined as fixed or it can be calculated by the receiver [5]. The main error source is the timing resolution, since the clock offset of the initiator is mostly cancelled in the two-way exchange, and the clock offset of the responder has only little effect if the turnaround time is short [5]. The RTT measurement procedure used in the 802.11 Fine Timing Measurement protocol is described in more detail in section 2.5.

2.2 Angulation

Angulation is based on measuring directions from an unknown position to at least two known positions. To calculate the 2-D location of a transmitter, two receivers at known locations need to measure the angle of arrival (AOA) of a signal that comes from the unknown location [10]. The angle measurement is done with direction finding techniques.

2.2.1 Direction Finding

Direction finding (DF) requires use of directional antennae or an array of antennae [10]. A rotating directional antenna detects the AOA usually from the minimum RSS, due to minima being sharper than maxima in the gain pattern [5]. Two directional antennas placed orthogonally can be used to detect the direction of arrival without the need of physical rotation. DF systems that use directional antennas have degree-level accuracy, but their performance decreases as the signal propagation environment worsens [5].

Antenna arrays are another popular method for DF. They consist of specifically arranged antennas that receive a signal at different times and phases [12]. The antennas are usually placed fractions or multiple wavelengths apart. They produce time- and phase-delayed outputs that are processed in order to solve the direction of arrival. DF accuracy improves with high number of antennas, and practical antenna array systems work with high-frequency signals due to size limitations [5].

2.2.2 Triangulation

When the angles of arrival have been measured in known positions, the transmitter position can be calculated. In figure 2.2, the baseline length l between receivers A and B is known. The AOAs are measured in relation to a common reference, such as the baseline l or true north. The angles define the LOPs for the transmitter, and the transmitter position is in the intersection of those lines.

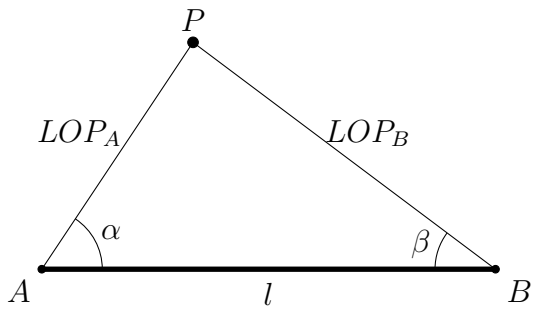


Figure 2.2: Triangulation in 2-D

Accurate positioning by triangulation requires accurate angle measurements [10]. Like RSS and TOA measurements, AOA measurements suffer from shadowing and multipath effect. Positioning accuracy also decreases if the receivers are far away from the transmitter. Complex equipment needed for the angle measurements is a disadvantage, too [10].

2.3 Fingerprinting

Fingerprinting, or pattern matching, is a positioning method that compares measured signal properties with previously recorded samples, and chooses the location of the closest match as the current position [5]. RSS fingerprinting using existing wireless network infrastructure is a viable method for positioning with accuracy of few meters [13]. The implementation complexity and positioning accuracy depend on how the sample database is gathered and how the closest match is found.

The sample database can be established by either empirical measurements or calculations with a signal propagation model [13]. In the empirical approach, this off-line phase consists of measuring RSS values from multiple APs in many locations within an area, such as an office. One sample in the database typically includes the coordinates of the measurement point, and the mean of several RSS values measured in that location. Averaging of the RSS values over time is necessary to decrease the effect of RSSI fluctuation

[7]. In RSS fingerprinting particularly, it can be beneficial to include user orientation information to the samples as well, since the body of the user attenuates the signals [13]. The mean RSS values from each AP form combinations that can be considered unique for each sampling location.

The RSS values for the samples can also be estimated with a propagation model, instead of empirically measuring them [13]. This semi-analytical approach omits the offline measurement phase, but requires development of a suitable indoor radio propagation model. Furthermore, a propagation model that takes attenuation caused by walls into account, may require empirical attenuation measurements [13]. The resulting sample for one location is similar to that of the empirical method: The RSS value computed by the propagation model corresponds to the mean of empirically measured values.

In addition to the sample database i.e. training data, a fingerprinting system needs a mapping technique that estimates the position of a new measurement [14]. There are multiple implementation options for the mapping technique, such as k-nearest-neighbor (kNN), probabilistic methods, support vector machine (SVM), neural networks, and smallest M-vertex polygon (SMP) [10].

Fingerprinting systems have the advantage of robustness against the challenging indoor radio propagation conditions [14]. This means that a system trained in certain conditions (e.g. furniture placement, closed doors and number of people) usually performs well in similar conditions in terms of positioning accuracy. The downside is that the training data must be updated appropriately if there are changes in the environment. Together with the effort required for the initial training, the update requirements make fingerprinting systems somewhat expensive and laborious to setup and maintain [14]. Hardware costs on the other hand are reasonable, because the system does not need any special hardware in addition to existing WLAN infrastructure.

2.4 Trilateration

The distance estimates produced by the chosen ranging method are used to calculate the unknown position by trilateration. Trilateration (or multilateration) in 2-D is a problem of solving the intersection point of at least three circles given their center coordinates and radii. In ranging-based positioning methods, the anchor stations are in the centers and the estimated ranges are the radii of the circles. In figure 2.3, A, B, and C denote the anchor positions and P is the user position to be solved. The circles represent the LOP for the user, and their radii r_A, r_B and r_C equal the distance between the user and the respective anchor.

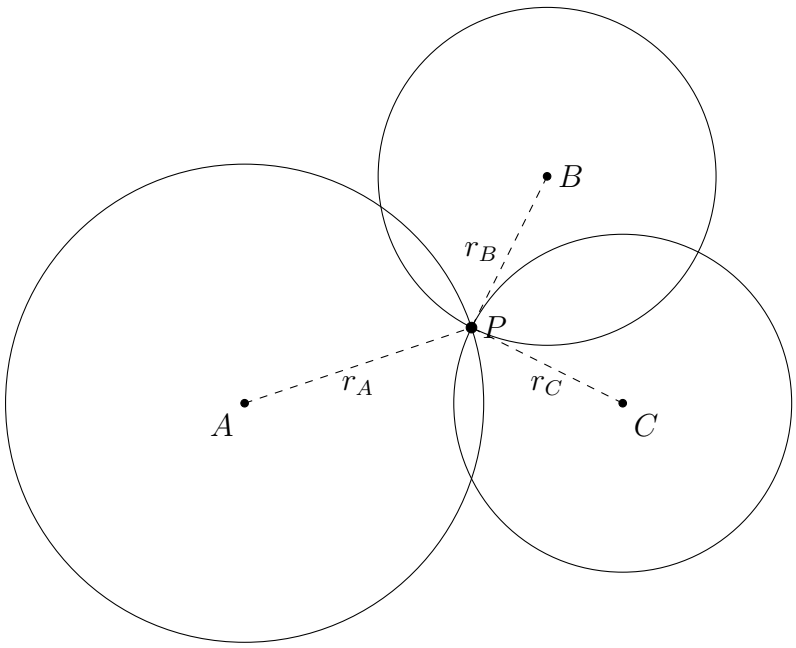


Figure 2.3: Trilateration in 2-D

Finding the exact intersection point of the circles is possible if the distance measurements are exact. It is done by e.g. solving a linearized system of the circle equations [15]. However, in navigation and positioning applications, the trilateration method usually involves inaccurate range measurements. When there is no exact intersection point between the circles or spheres, it is better to search for the best approximate solution with e.g. the nonlinear least squares (NLS) method [16]. The position can also be approximated with

a Kalman filter (KF), which handles the coordinates as states that are updated based on predictions and new range measurements. Overviews of using these two methods in positioning are given in the following.

2.4.1 Nonlinear Least Squares

NLS is a method in which the best estimate for the position is found by minimizing the sum of squared distance errors [15]. The distance error is the difference between the measured range r and the exact distance $\hat{r}$ from the current position estimate to the respective anchor. In 3-D positioning with measurements to n anchors, the sum to be minimized can be presented as a function

$$F(x, y, z) = \sum_{i=1}^{n} (\hat{r}_i - r_i)^2, \tag{2.8}$$

where

$$\hat{r}_i = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2}. \tag{2.9}$$

The minimum of $F(x, y, z)$ can be found iteratively, e.g. with the Newton's method [15]. The method needs an initial guess for (x_0, y_0, z_0) in order to converge to the minimum. The stationary anchors can be utilized for the initial guess in various ways, such as using the coordinates of the anchor with the shortest measured range, or calculating the midpoint of a closed path formed by the anchors [17].

An NLS positioning algorithm has the advantage of being able to make use of range measurements from more than three anchors, and thereby handle the effects of errors more reliably [16]. The accuracy of NLS trilateration suffers from bad geometry between the actual position and anchor positions. Another problem is the possibility of converging to a local minima, if the initial guess is far away from the true position [17].

Murphy et.al. [15] compared different methods for solving 3-D trilateration with approximate distances. NLS proved out to be the most effective method, against linearized equations and linear least squares.

2.4.2 Kalman Filter

Kalman filter (KF) is a state estimation algorithm that is commonly used in navigation systems [5]. It is named after Rudolf E. Kálmán, who introduced the fundamental technique in 1960 [18]. After further development together with Richard S. Bucy in [19], the filter has been implemented in many applications. Practical development by Stanley F. Schmidt in NASA's Ames Research Center led up to Kalman filter being an important part of the navigation system of the Apollo Project [20]. Since then, authors have used several different notations and naming conventions for the elements and steps of the filter. Naming and notation used in [5] will be followed in the next overview.

The Kalman filter estimates a system by maintaining states of selected system parameters, for example the xy-position and velocity of a mobile device. The set of states is called the *state vector*. It is updated by predicting new values based on previous ones, and by incorporating new measurements to the estimates. Each of the state estimates has some uncertainty, and they are represented in an *error covariance matrix*. It holds information of the errors in the estimates and the correlation between them. The state vector and error covariance matrix change with time according to the *system model*. The xy-position state, for example, changes as the integral of the velocity state during a time period. Also, the state errors in the error covariance matrix increase with time, as the estimates will become outdated without new measurements. [5]

A set of new measurements (e.g. ranges to wireless anchors) is brought to the system in a *measurement vector*. The measurements have noise, which is described in a *measurement noise covariance matrix*. The noise can be defined as constant or dynamic. The relation between the measurement vector and the state vector is defined by the *measurement model*. It includes calculation of a *measurement matrix*, which tells for example how range measurements from wireless anchors change with the xy-position of the mobile receiver. [5]

The process can be divided into two phases: system propagation and measurement

update. A whole iteration is executed every time new measurements have been taken. The system propagation phase begins the process with predictions of the state vector and error covariance matrix. It is also known as the time propagation phase, because the state and error covariance propagation is predicted from the time of the previous measurements to the present. In a practical example, the position state is predicted based on the velocity state and the elapsed time. [5]

In the measurement update phase, the predicted states and error covariances are updated with new measurement data. Important step in this phase is the calculation of the Kalman gain. It defines how trustworthy the new measurements are when updating the estimates, based on measurement noise and the uncertainty in the current estimates. If the measurements are very noisy, they have only a minor effect on the state update. Also, if the current estimates have low uncertainty, the measurement update changes them only slightly. After the state estimate update, the error covariances are updated as well according to the new measurement information. [5]

The standard Kalman filter assumes that the relation between the measurement and state vectors is linear. In a real navigation system based on range measurements, the relation is however nonlinear. Extended Kalman filter (EKF) can be used for these kind of nonlinear processes [21]. EKF is an extension to KF that linearizes the nonlinear state and measurement transitions with Taylor series expansions. Due to these linearizations, EKF performs poorly with problems of high nonlinearity. [22] However, if the errors caused by the linearization are significantly smaller than the errors caused by system and measurement noise, EKF is useful. Trilateration, for example, is one of these so called quasi-linear problems. [20] For highly nonlinear problems, the unscented Kalman filter (UKF) performs better than EKF. Instead of linearizing the system and measurement models, UKF makes a nonlinear transformation and approximates the resulting probability density [22]. Kalman filters are well suited for Gaussian measurements and noise distributions, but with estimation problems that have non-Gaussian distributions, a particle filter performs better

[5]. As a downside, particle filters come with a higher computational cost.

2.5 Fine Timing Measurement

Fine timing measurement (FTM) was introduced to the IEEE 802.11 Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications in 2016. The amendment is also known as 802.11mc, after the IEEE 802.11 REVmc Task Group that published it. After standardization, FTM has been adopted in Wi-Fi CERTIFIED Location [3], which is a certification program by Wi-Fi Alliance. At the time of writing, there are Wi-Fi Location certified WLAN chipsets from several vendors, but FTM-enabled commercial products are yet to emerge in large scale.

FTM allows a WLAN device to measure accurate round-trip time (RTT) to another device and back [2]. After a successful measurement, the distance between the devices can be calculated as discussed in 2.1.4. When multiple FTM frames are transferred periodically, the initiating station (STA) can keep track of its position relative to other STAs nearby [2]. Positioning by FTM is based on ToF ranging and trilateration.

The RTT is calculated from timestamps that are captured at departure and arrival of FTM frames and their respective acknowledgements. The protocol itself is similar to the former 802.11v Timing Measurement protocol; one of the most notable improvements is the increase in timestamp resolution from 10 nanoseconds to 100 picoseconds [23]. In the FTM procedure, an initiating STA starts an FTM session with a responding STA. A STA may hold multiple sessions at the same time (e.g. a STA measuring distance to multiple APs, or an AP responding to multiple STAs) [2]. During a session, the STAs exchange FTM frames in bursts that are triggered by the initiator. In total, an FTM session consists of three phases: negotiation, measurement exchange and termination. The next two sections give an overview to the negotiation and measurement exchange phases, according to the standard [2].

2.5.1 Discovery and Negotiation

WLAN STAs discover each other by periodically sending out probe requests or beacon frames. Probe requests are typical for e.g. mobile phones scanning for wireless networks, whereas access points broadcast beacon frames to let other STAs know about them. Probe responses and beacon frames have elements that carry information about the capabilities of the STA: the Capability Information and Extended Capabilities. Information about FTM capabilities are advertised in the Extended Capabilities element. The element has a field of n octets (8 bits) indicating support for various features. Bit 70 is set to 1 if the STA acts as an FTM responder, and bit 71 is set to 1 if the STA acts as an FTM initiator.

A STA that works as a FTM initiator, begins the procedure by transmitting a Fine Timing Measurement Request frame. The first instance of this type of frame is called the *initial* Fine Timing Measurement Request. The frame includes a Trigger field that indicates whether to start, continue or stop the measurement process, together with a Fine Timing Measurement Parameters element, that holds information about the scheduling and operational details of the FTM session. One of them is the ASAP field, which the initiator uses to request either scheduled or immediate measurement. Within the Parameters element, the initiator also suggests values to be used in parameters such as burst duration, FTM frames per burst, bandwidth, and minimum time between consecutive FTM frames. The Fine Timing Measurement Request can optionally include also request elements for location configuration information (LCI) and Location Civic measurements. An LCI Report includes information about the location of the responding STA, e.g. the geographic coordinates in a format defined in IETF RFC 6225 Section 2.2 [24], floor number and height above floor. A Location Civic Report contains the street address of the STA either in a vendor specific or IETF RFC 4776 [25] format. LCI and Location Civic reports, if available, can be utilized by a STA for self-positioning purposes.

After sending the initial Fine Timing Request frame, the initiating STA stands by for receiving an initial Fine Timing Measurement frame. It should be transmitted by the

responding STA within 10 ms. This frame also includes the Parameters element that has the settings supported by the responding STA. The settings should be the same or similar as those requested by the initiating STA, if the responding STA supports them. The Status Indication field indicates if the request was successful or not. If the responding STA is incapable or unable (at that moment) to meet the requirements of the request, it sets the Status Indication field accordingly, and the FTM session ends. Otherwise, the negotiation is complete and the request successful.

2.5.2 Measurement Exchange

If the STAs agreed on a scheduled measurement (ASAP=0), the initiating STA begins the first burst by sending a new FTM Request, this time without the FTM Parameters element. The Trigger field is set to 1, which indicates that the initiator is available for the rest of the burst. The responding STA sends an Ack in response, and then transmits Fine Timing Measurement frames according to the agreed number of bursts, and frames per burst. The initiating STA responds to those frames by sending an Ack after each one. Timestamps captured from a single Fine Timing Measurement frame exchange are shown in table 2.1. If the measurement was agreed to begin as soon as possible (ASAP=1), the timestamps are captured already from the initial Fine Timing Measurement frame. The message exchange in the ASAP=1 case is shown in figure 2.4.

Table 2.1: Timestamps captured during an FTM frame exchange

Timestamp	Captured by	Event
t_1	responding STA	FTM frame is transmitted
t_2	initiating STA	FTM frame arrives
t_3	initiating STA	Ack is transmitted
t_4	responding STA	Ack arrives

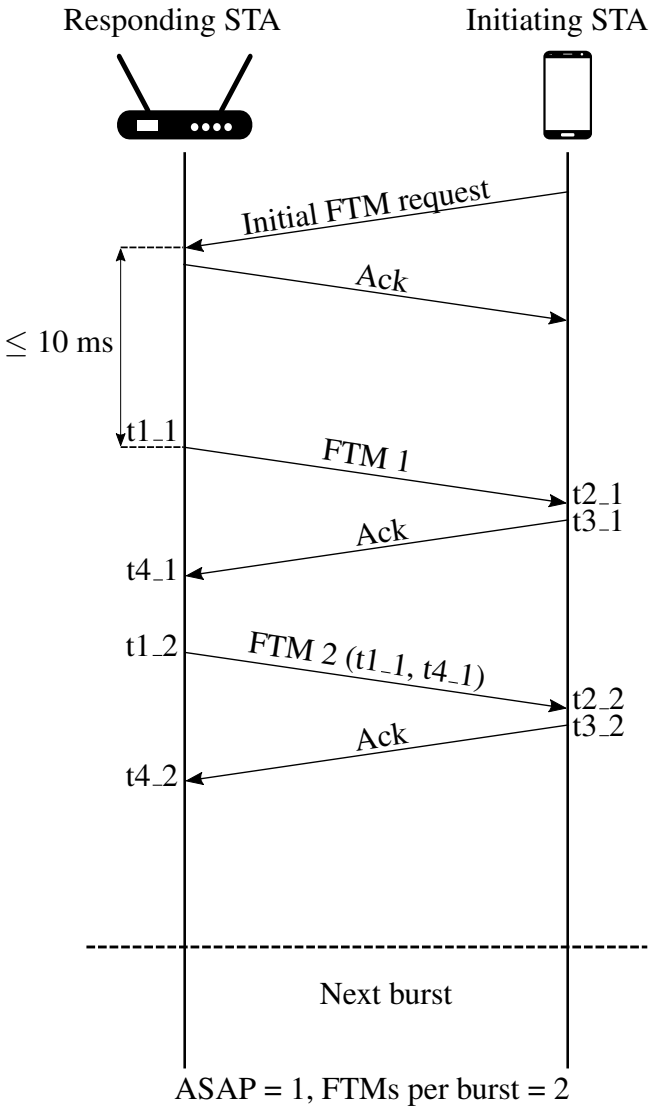


Figure 2.4: Fine Timing Measurement exchange example

The responding STA sends timestamps t_1 and t_4 to the initiating STA as part of the next Fine Timing Measurement frame. The timestamps are reported in picoseconds. Together with timestamps t_2 and t_3, the initiating STA can calculate the round-trip time t_{RTT}, turnaround time τ and one-way time-of-flight t_{TOF} as follows:

$$t_{RTT} = t_4 - t_1 \tag{2.10}$$

$$\tau = t_3 - t_2 \tag{2.11}$$

$$t_{TOF} = \frac{t_{RTT} - \tau}{2} \tag{2.12}$$

By multiplying t_{TOF} with the speed of light, the STA calculates the distance to the responding STA as described in section 2.1.4.

2.5.3 Android RTT API

The support for Fine Timing Measurements was added to the Android OS version 9 in 2018. The measurements are referred to as Wi-Fi RTT measurements in the public Android application programming interface (API). According to the Android Developers Guide, the device position can be estimated with 1-2 meters accuracy by using Wi-Fi RTT measurements and multilateration. The use of Wi-Fi RTT on Android requires that the device has hardware that supports FTM, and runs Android 9 Pie or later. The user must enable location services and Wi-Fi scanning on the device, and grant location permissions to the app that is using the feature. [4]

The main phases of RTT ranging with an Android application are:

1. Scanning for Wi-Fi networks
2. Extracting FTM-enabled APs from the scan results
3. Building a ranging request with the selected APs
4. Handling results from completed ranging operations

Each ranging operation returns a `RangingResult` object that holds information described in table 2.2. The information in one result comes from a burst of RTT measurements done with the responding device.

Table 2.2: Information in a ranging result

Name	Description
Status	Measurement success/failure
MAC address	The MAC address of the responding device
Responder location[1]	Object containing the LCI configured in the AP
Attempted measurements	The number of attempted measurements in the exchange
Successful measurements	The number of successful measurements that are used to calculate the distance
Distance	The average measured distance in mm
Distance std.	The standard deviation in the measured distance
RSSI	The average RSSI in dBm
Timestamp	The time when the measurement was done in ms since boot

If the measurement failed (indicated by the status field), all other fields except status and MAC address are null. Therefore, the handling of every measurement result should begin with identifying the responding device and checking the status. In case of a successful measurement, the rest of the information can then be extracted and used in a positioning algorithm within the same application or e.g. on a remote server. In Android 10, the method for acquiring the responder location from the LCI was added. This enables self-positioning on the phone provided that multiple APs have their location configured.

[1] Added in Android 10

2.6 Related Work

WLAN RTT-based positioning has been studied for a while before the FTM protocol. In 2005, Günther et al. [26] used affordable commercial WLAN products to estimate distance based on the RTT of a WLAN data packet. The hardware was capable of only 1 μs resolution timestamps, so they used statistical smoothing over multiple samples to refine the time delay resolution. They concluded that RTT correlates with distance much better than RSSI, and should therefore be used in distance estimation. Schauer et al. [27] achieved mean distance error of 1.33 meters in optimal environment, but stated that the distance estimates still had too large deviation for practical indoor positioning. The authors noted that more precise clock were needed in WLAN devices.

Since FTM-enabled devices with precise clocks have emerged, their ranging and positioning performance has been under research. Ibrahim et al. [28] implemented a tool to analyse the ranging accuracy of devices equipped with Wi-Fi chipsets by Intel and Qualcomm. From extensive measurements in various conditions, they concluded that meter-level ranging accuracy is possible in open space after calibration, but in an indoor office environment, ranging and positioning errors increase to around 5 meters. Using corrected range estimates and 80 MHz bandwidth signals, they achieved 3.5 m median and 4.7 m 90th percentile positioning error.

Banin et al. [29] used FTM measurements to compare an extended Kalman filter with a Bayesian filter that was complemented with floor layout information. In an office environment they found that the ranging accuracy stays within one meter for 60% of measurements at less than 10 meter distances, but degrades due to multipath propagation at longer distances. They had 3 m 90th percentile positioning error with the EKF, and less than 2 m 90th percentile error with the Bayesian filter that was enhanced with map information and smoothing.

FTM is the ranging method for next generation 802.11 WLAN positioning systems. The goal of IEEE 802.11az next generation positioning (NGP) Task Group is to enhance

accuracy and scalability as well as reduce wireless medium usage in WLAN positioning [30]. Following these objectives, Banin et al. [31] proposed a collaborative ToF system, in which a mobile STA positions itself by observing broadcast beacons from stationary STAs. The system also has a network-based client tracking mode, in which the stationary STAs observe broadcast beacons sent from mobile STAs. Pefkianakis and Kim [32] presented a system called *mWaveLoc*, which is based on 60 GHz 802.11ad devices that use direction finding and FTM ranging to locate client STAs. They achieved median ranging error of 4 cm with FTM and decimeter-level 3-D positioning accuracy in line-of-sight (LOS) conditions. However, they noted that a human body blocking the LOS of the 60 GHz signal poses a ranging error of 1.14 meters and a 70% decrease in TCP throughput.

Chapter 3

Implementation

The proposed indoor positioning system consists of a mobile phone and five Wi-Fi FTM access points (APs). The APs are fixed in known locations, and the phone performs range measurements with each of them. For this thesis, two Android applications utilizing the Wi-Fi RTT API were developed: Firstly, an existing multi-purpose data logging app was complemented with RTT features. Secondly, a completely new application for real-time positioning and map visualization was developed. In this chapter, the hardware and software used in the positioning system are introduced.

Figure 3.1: Google Pixel 2 XL and Compulab FTM Responder

3.1 Access Points

At the time of writing, commercial FTM-enabled WLAN APs were difficult to find. One of the very first commercially available devices, Compulab FTM Responder [33], was used in this work. It is essentially a compact PC equipped with an Intel AC8260 Wi-Fi card and custom firmware that implements the FTM protocol. It runs a Yocto Project BSP-based embedded Linux distribution that is preconfigured to act as a Wi-Fi AP responding to FTM requests. The same device can also run the OS image of Compulab WILD (Wireless Indoor Location Device), a similar but more finalized product. The OS is a more advanced GNU/Linux Debian with KDE desktop environment. With the WILD software, the device can easily be configured to act either as an FTM responder or initiator.

The access points were set to operate in the 5 GHz Wi-Fi band and with 80 MHz channel bandwidth. The AP configuration is defined in the `hostapd.conf` file. FTM related settings are configured as follows:

Listing 3.1: FTM-enabled AP configuration

```
# Advertise FTM responder capability in the Extended
# Capabilities element
ftm_responder=1

# Set operation mode to a = IEEE 802.11a (5 GHz)
hw_mode=a

# Enable IEEE 802.11n (HT)
ieee80211n=1

# Enable IEEE 802.11ac (VHT)
ieee80211ac=1
```

```
# Set channel width to 1 = 80 MHz
vht_oper_chwidth=1
```

The APs broadcast the same service set identifier (SSID), although the FTM protocol does not require the client STA to be associated with a network. Each AP has its own basic service set identifier (BSSID), which is used in positioning when combining a range measurement with the coordinates of the AP in question.

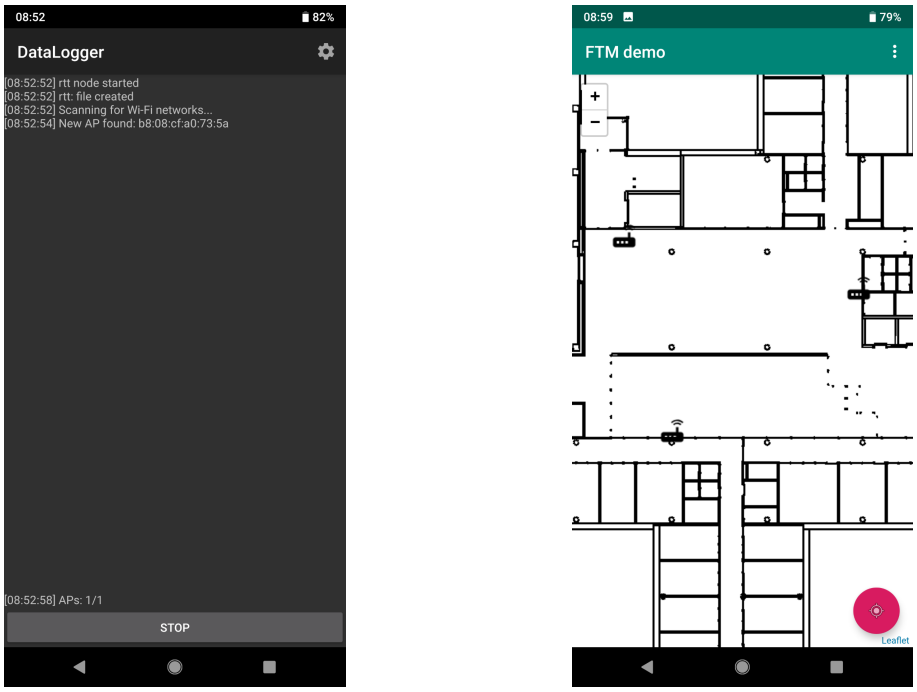
3.2 Mobile Device

In this work, a Google Pixel 2 XL was used as the mobile device to be positioned. It is equipped with Qualcomm Snapdragon 835 mobile platform that is capable of fine timing measurements. The phone has also good software support by Google and is among the first phones to receive updates for the latest version of Android. The phone currently runs Android 9 Pie, which is the first version to have Wi-Fi RTT measurements in the public API.

As mentioned, the RTT functionality was tested with two applications. Datalogger is an Android app developed at VTT for network and sensor measurement purposes. The app can be used to collect data from multiple sources simultaneously, for example GNSS location and LTE signal information. Other data sources include e.g. Wi-Fi, 2G, 3G, Bluetooth, camera, magnetometer, and accelerometer, depending on the phone model. As part of this thesis work, Wi-Fi RTT range measurement logging was added. The data is written to a file and optionally sent to a remote server as user datagram protocol (UDP) messages. Measurement files are used in extensive offline analysis, whereas the UDP transmissions enable real-time monitoring and remote positioning.

For real-time self-positioning on the device, an FTM Demo App was developed. It allows the user to see their location indoors on a provided floor layout. The RTT ranging

procedure is the same as in Datalogger, and the application calculates the position estimates with a Java version of the NLS algorithm described in section 3.3. The next two subsections describe the RTT ranging process used by both applications, as well as the location visualization elements of the FTM Demo App.



(a) Datalogger (b) FTM Demo App

Figure 3.2: Android applications using Wi-Fi RTT

3.2.1 Range Measurements

For RTT ranging, two classes of the Android API are mainly used: `WifiManager` and `WifiRttManager`. Closely related to them are the `ScanResult` and `RangingResult` classes, respectively. The ranging process (Figure 3.3) starts with a regular Wi-Fi scan. In Android, the scan is initiated by acquiring a `WifiManager` object and calling its `startScan()` method. A list of scan results is received asynchronously. The FTM-enabled APs are extracted from the results by calling `is80211mcResponder()` on each `ScanResult` object. They are added to a maintained list of FTM responders,

which holds every unique FTM-enabled AP detected during a measurement run.

Next, a `RangingRequest` is built, and the list of FTM responders is added to the request. The ranging operation begins by calling the `WifiRttManager.startRanging()` method. The results are again received asynchronously. Each `RangingResult` object holds information about one ranging procedure with an AP, see Table 2.2. The information is either appended to a log file, or input to the positioning algorithm. The next ranging operation is scheduled to begin after 500 ms in both apps.

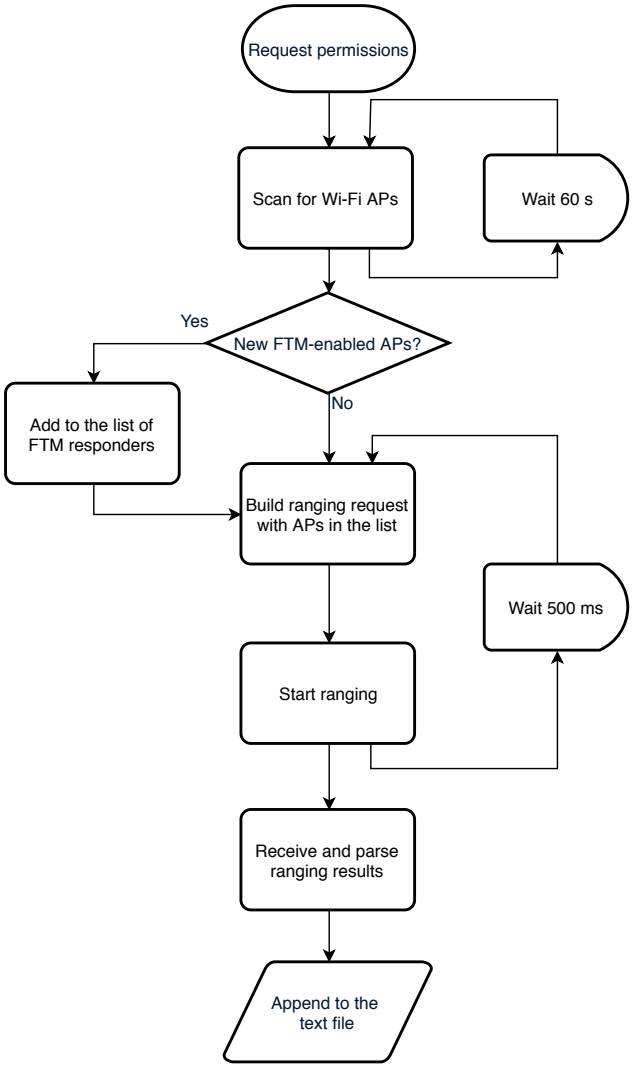


Figure 3.3: The ranging process

3.2.2 Location Visualization

For visualizing the estimated position in FTM Demo App, a `WebView` is used to display the floor layout. The `WebView` object displays an HTML page, which in turn loads a JavaScript (JS) file that initializes and maintains the map. The map view is implemented with Leaflet.js [34], an open-source JS library widely used for interactive maps. The app includes several floor layout images that can be selected as the map. Each map uses local, Cartesian coordinates, which are defined based on the size and scale of the source image.

On Android, interaction between native Java/Kotlin code and JavaScript is possible via a JavaScript interface. Functions in the JS file can be used from Android by calling the `WebView.loadUrl()` method. Native methods can be exposed to JS by using the `@JavascriptInterface` notation. Figure 3.4 describes the interaction between the app components.

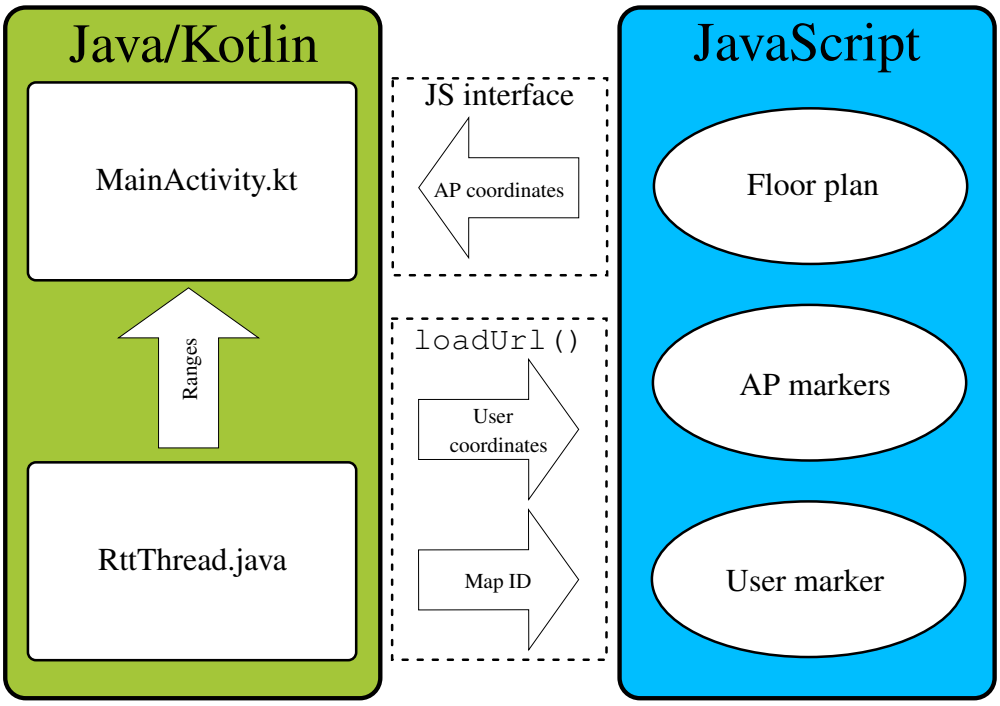


Figure 3.4: Native to JS interaction in FTM Demo App

Interaction from JS to app is needed in the placement of access points: the user can place AP markers on the Leaflet map, and once completed, their coordinates are passed

to the native side of the app. This placement phase is useful in ad-hoc demo setups, when the APs are not configured to broadcast their coordinates with the LCI. Interaction from app to JS is needed when the positioning algorithm outputs the position estimate: the coordinates are passed to JS and the position of the circle marker is updated on the Leaflet map. The map selection is also done in a native dialog menu, and the selected floor layout ID is passed to JS.

3.3 Positioning Algorithms

Two algorithms are used in this work to produce position estimates out of FTM range measurements: an Unscented Kalman Filter (UKF) and a Nonlinear Least Squares (NLS) estimator. The NLS algorithm is used to evaluate the positioning accuracy that can be achieved using only raw range measurements, whereas the UKF algorithm uses a kinematic model to make predictions while reducing the effects of outlier range measurements. Both algorithms are written in Python as parts of scripts that process recorded range measurement data, and the NLS algorithm is also implemented in Java as part of the FTM Demo App. Descriptions of the two algorithms and their Python implementations are given in the following.

3.3.1 NLS

As previously discussed in section 2.4.1, the NLS algorithm is based on minimizing a function that returns the sum of squared differences in estimated and measured ranges. In the post processing script, a measurement file in .csv format is loaded into a `pandas` [35] DataFrame. The measurements are input to the algorithm one group of simultaneous ranging results at a time. Each group has a common timestamp, which is used as the key for `pandas.DataFrame.groupby`. The NLS function is then applied to each group:

Listing 3.2: Applying a function to each group

```
import pandas as pd

# load file to a DataFrame and make a copy
measurements = pd.read_csv(path, sep=";")
result_df = measurements.copy(deep=True)

# apply nls to each timestamp group
result_df = result_df.groupby('timestamp',
        sort=False).apply(nls)
```

The minimized function `squared_differences` takes two arguments: the current estimate for coordinates and a list of AP coordinates and measured ranges to them. The difference between the Euclidean distance between the estimate and the AP and the measured range is calculated, squared, and added to the total sum.

Listing 3.3: Error function to be minimized

```
def squared_difference(estimate, ap_values):
        diffs = 0.0
        for row in ap_values:
                difference = np.linalg.norm(row[:2] -
                estimate) - row[3]
                diffs += difference*difference
        return diffs
```

The `nls` function takes a DataFrame as an argument, makes modifications, and returns it. In this case, the function receives a group of measurements, calculates a position estimate from them, and writes the estimated coordinates to their respective columns in the data frame. Provided the measurement file also has the ground truth coordinates, the

positioning error is also calculated and written to a dedicated column in the DataFrame.

Listing 3.4: The NLS function

```
import numpy as np
import scipy.optimize

def nls(group):
        global start_point
        ground_truth = group.iloc[0, 8:10].values  # x,y

        ap_values = []
        for row in group.itertuples():
                ap_values.append([
                getattr(row, 'bts_loc_x'),
                getattr(row, 'bts_loc_y'),
                getattr(row, 'est_distance')/1000.0)

        # use only 4 closest APs
        if len(ap_values) > 4:
                ap_values.sort(key=get_range)
                ap_values = ap_values[:4]
                ap_values = np.array(ap_values)

        # choose the closest AP as starting point
        # in the first round
        if len(start_point) == 0:
                start_point = ap_values[0, 0:2]
```

```
result = scipy.optimize.minimize(
        squared_difference,  # The error function
        start_point,  # The initial estimate
        args=(ap_values,),  # Additional
        parameters for error function
        method='L-BFGS-B',  # The optimisation
        algorithm
        bounds=((0.0,41.0),(0.0,23.0)),  #
        Coordinate constraints
        options={
                'ftol': 1e-4,  # Tolerance
                'maxiter': 1e+7  # Maximum
                iterations
        })
error = np.linalg.norm(result.x - ground_truth)
start_point = result.x
group['est_loc_x'] = result.x[0]
group['est_loc_y'] = result.x[1]
group['positioning_error'] = error
return group
```

The NLS optimization for finding the position estimate is done with `scipy.optimize.minimize`. The function is provided with an initial estimate for the coordinates, a group of measurement values, and the function to be minimized. The minimization method can be chosen from several optimization algorithms, such as Nelder-Mead, Powell, CG, BFGS, L-BFGS-B or COBYLA. L-BFGS-B was chosen for its possibility to utilize box constraints on the variables. This can be used to limit the possible area of position with lower and upper limits for xy-coordinates. The optimizer converges to a

local minima of the function, and the returned xy-coordinate values are the estimated user position. The estimate is saved to the global variable `start_point`, to be used as the initial estimate for the next group of measurements. This is the only case where this algorithm uses information from previous measurements, and it mainly helps the optimizer to converge quicker to the correct local minima. Range measurement noise or overestimated ranges due to NLOS conditions cause the output trajectory to be noisy.

3.3.2 UKF

To produce more stable and accurate trajectory, an Unscented Kalman Filter was implemented. The Python script uses `FilterPy`, an open-source library by Roger Labbe [36]. The library has tools for creating and running a Kalman filter, such as maintaining the states and covariance matrices and calculating the matrix operations in the prediction- and update phases. In the case of UKF, the library also provides tools for generating the sigma points and doing the unscented transform. The following code generates the sigma points and initializes the filter:

Listing 3.5: Initializing the UKF with sigma points

```
from filterpy.kalman import MerweScaledSigmaPoints
from filterpy.kalman import UnscentedKalmanFilter as UKF


sigmas = MerweScaledSigmaPoints(n=4, alpha=.8, beta=2.,
kappa=-1.)
num_aps = 5  # length of the measurement vector
dt = 0.5  # initial time step between measurements
global ukf
ukf = UKF(4, num_aps, dt, h_rtt, f_rtt, sigmas)
```

For this thesis, a UKF was designed to estimate the position of a smartphone that lies on top of a wheel-based robot. The movement of such robot can be assumed as rather steady with top speed no more than walking speed. During the measurements, the robot is piloted manually along a predetermined route, so there are very few sudden accelerations to directions along or perpendicular to the route. This is a somewhat easier scenario for the filter compared to e.g. a walking person. These characteristics of the robot movement define some of the design choices of the filter.

Firstly, a constant velocity state model was chosen. This means that the filter tracks position and velocity in two or three dimensions, and considers acceleration as noise. By using Newton's dot notation for differentiation, the derivate of position x is marked with $\dot{x}$. That represents the velocity state, since the change in position over a period of time Δt is velocity. The state vector $\mathbf{x}$ of the filter in 2-D positioning is

$$\mathbf{x} = \begin{bmatrix} x & \dot{x} & y & \dot{y} \end{bmatrix}^\top \tag{3.1}$$

When the filter is initialized, the first set of measurements is used to estimate the initial states: the coordinates of the nearest measured AP are used as values for x and y. The velocity states $\dot{x}$ and $\dot{y}$ are set to zero. The error covariance matrix $\mathbf{P}$ consists of the variances of the states in the diagonal elements, and their covariances in the off-diagonal elements. The nearest AP coordinates are used as the initial x and y states, and its range as the basis for the variance of those states. The variances of the velocity states are derived from the maximum speed of the robot. The filter updates the matrix at each iteration, but the initial values should not be too small, to prevent the filter being overconfident. The filter will also calculate the covariance between the states, so they can be initially be set to zero. The error covariance matrix is initialized as

$$\mathbf{P} = \begin{bmatrix} \sigma_x^2 & 0 & 0 & 0 \\ 0 & \sigma_{\dot{x}}^2 & 0 & 0 \\ 0 & 0 & \sigma_y^2 & 0 \\ 0 & 0 & 0 & \sigma_{\dot{y}}^2 \end{bmatrix} \tag{3.2}$$

where σ^2 means the variance of each state.

The prediction phase of a Kalman filter includes state propagation, or transition. The system model describes how the states change with time. The system model of this kind of constant velocity filter is based on Newton's equations of motion. With velocity $\dot{x}$ and previous position x_0, the new value for position x after time t is

$$x = x_0 + \dot{x}\Delta t \tag{3.3}$$

Using the same equation for y and keeping the velocity states constant, the filter predicts the new states by multiplying the state vector by the following state transition matrix $\mathbf{F}$:

$$\mathbf{F} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

The state transition would also include the effects of external inputs to the system, such as gravity or wheel control inputs of a robot, with control vector $\mathbf{u}$. Since neither is the robot itself modelled nor acceleration measured, $\mathbf{u}$ can be discarded completely. The system model is linear, but since UKF supports also nonlinear system models, the `filterpy` implementation uses a function instead of a matrix for the state transition. In this case, the function just performs the multiplication $\mathbf{Fx}$:

Listing 3.6: The state transition function

```
def f_rtt(x, dt):
        """ state transition function """
        F = np.array([[1, dt, 0,  0],
                      [0,  1, 0,  0],
                      [0,  0, 1, dt],
                      [0,  0, 0,  1]], dtype=float)
        return np.dot(F, x)
```

The steady movement of the robot affects also the design of the process noise matrix $\mathbf{Q}$. Process noise accounts for all the unknown changes in the system states that occur over time. Essentially, it is white noise that is added to the error covariance matrix $\mathbf{P}$ during state propagation, so that the uncertainty in the states increases with time in the absence of new information from measurements. In the case of robot movement, the process noise can be considered relatively small. The amount of process noise has an effect on how much the filter will trust its prediction compared to the measurements. $\mathbf{Q}$ is initialized with white noise in `filterpy` as follows:

Listing 3.7: Setting the process noise matrix

```
from filterpy.common import Q_discrete_white_noise
q_var = 0.002  # process noise variance
q = Q_discrete_white_noise(2, dt=dt, var=q_var)
ukf.Q = block_diag(q, q)
```

The function `Q_discrete_white_noise()` models the white noise based on the variance of the highest order term of each state variable. In the above code, the function returns a 2x2 matrix calculated with time step of 0.5 seconds and noise variance that is approximately the maximum change in one-dimensional velocity that can happen during that time step. However, the variable `q_var` is usually chosen empirically [36]. Its value

is also later increased to make the filter adapt to sudden changes in velocities. The returned matrix is used twice in the actual process noise matrix $\mathbf{Q}$, once per each dimension.

The measurements are incorporated to the filter in the update phase. Every half seconds, the Datalogger app produces RTT range measurements of every available AP. The ranges and their respective AP coordinates are given to the filter in a measurement vector $\mathbf{z}$. A measurement model defines how the current state is transformed into a set of measurements. The model is nonlinear, since it includes square root in the calculation of Euclidean distance between two points. The filter has the following measurement function $\mathbf{H}$:

Listing 3.8: The measurement function

```
def h_rtt(x, aps):
        """ measurement function for converting 2-D
        constant velocity states
        into range measurements"""
        pos = np.array((x[0], x[2]))
        ranges = []
        for ap in aps:
                range_ = np.linalg.norm(ap - pos)
                ranges.append(range_)
        return ranges
```

The measurements have noise, and to model that, the filter uses the measurement noise covariance matrix $\mathbf{R}$. The noise in measurements to each AP is independent, so the off-diagonal elements in the matrix can be set to zero. The diagonal elements equal the variance in each range measurement. From the Android RTT API, the standard deviation of an RTT measurement burst can be used for this. The standard deviation of a long-term calibration measurement could also be used. As the process noise affects on how the uncertainty of the states increases with time, the measurement noise defines how reliable the

measurements are. They both affect the calculation of the Kalman gain, which essentially assigns proper weights to the prediction and the measurements.

The UKF script handles measurement data in the same way as the NLS script: the data is loaded into a `pandas` DataFrame, and the predict-update cycle of the filter is applied to every timestamp group. After the prediction, the new set of measurements is checked for unwanted values: if the measured range differs from the predicted range more than a set value (e.g. 8 meters), the measurement is deemed invalid due to NLOS conditions or other anomaly. The measurement noise covariance matrix $\mathbf{R}$ is recalculated in correct size and possibly with the measurement specific variances. After the call to `update()`, the estimated coordinates from the state vector are saved to the DataFrame.

The updated estimate is compared to the predicted estimate by calculating the normalized residual. A large value means that the new measurements indicate different behaviour than what was predicted, such as a sudden change of direction. In such event, the process noise is gradually increased to make the filter more responsive to measurements. Once the normalized residual decreases under a limit value, the process noise is gradually decreased. This kind of continuous process noise adjustment improves the performance of a constant velocity filter [36].

Chapter 4

Measurement Setup

To gather data for developing and testing the positioning system, various indoor measurements were conducted. Firstly, calibration measurements were done to study the ranging accuracy of the device combination in use. Afterwards, measurements were performed in two different indoor scenarios to find out if the promises of meter-level positioning accuracy hold true. In this chapter, the measurement tools and methods are described.

4.1 Reference Location

A useful tool for indoor positioning research at VTT is a remote controlled robot (Figure 4.1). The robot is used for the laborious task of gathering data in known locations. The robot uses a Lidar sensor and a simultaneous localization and mapping (SLAM) algorithm to generate a map of its surroundings. By matching the map with a floor layout, the position of the robot is known in a local coordinate system with accuracy of approximately 5–10 centimeters. This position is considered as the ground truth for measurement samples. The robot position data and the RTT measurement data from the smartphone are merged based on timestamps; the clocks of the on board control laptop and the smartphone are synchronized before every measurement run. With the ground truth position available, it is possible to evaluate the positioning accuracy of the system under development.

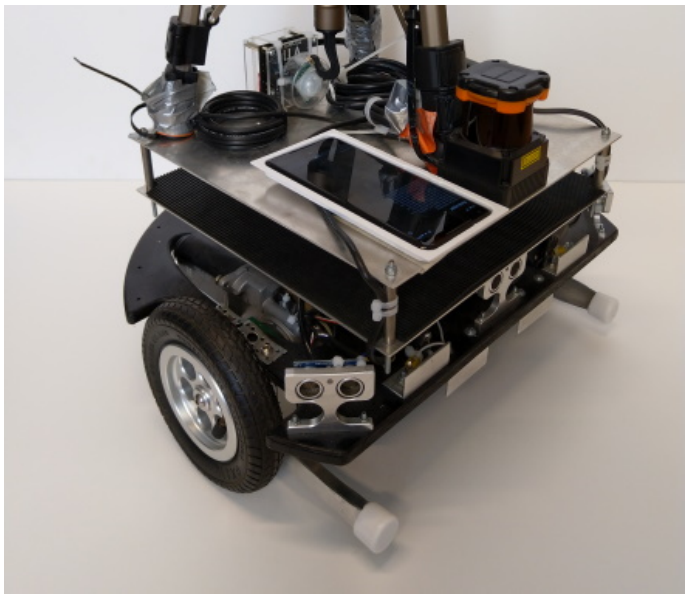


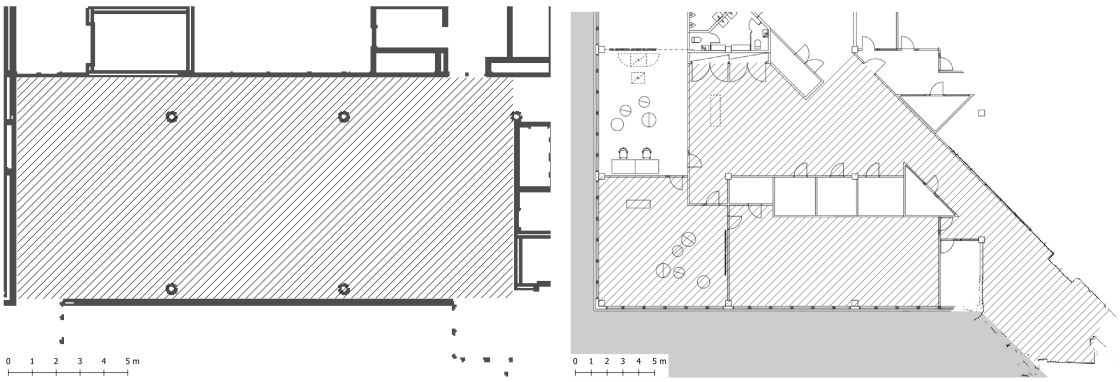
Figure 4.1: Measurement robot

4.2 Visual Analysis

For visualizing the estimated path and accuracy, a Python plotting library `matplotlib` [37] was used. Also, to draw the path on top of a floor layout and to compare it to the paths of other positioning techniques, QGIS [38], an open-source geographic information system, was used. The measurements were performed in two buildings from which digital 3-D models were available. Measurement files can be input to the models via a data interface, so that the estimated path and accuracy are drawn in the model.

4.3 Sites

The measurements were done at two sites. **Site A** is an open room of 190 m^2 in the VTT Micronova building. It is surrounded by three full height walls while the fourth side is open. The room is furnished with tables and chairs. **Site B** spans multiple rooms and a hallway in an office building at Nokia Campus. The combined area of the rooms visited during the measurements is 310 m^2. Figure 4.2 shows the layouts of the two measurement sites.



(a) Site A (b) Site B

Figure 4.2: Test sites and their measurement areas

4.4 Calibration Measurements

The need for a calibration measurement was discovered soon after the first tests with the FTM Responder devices. The reported distance seemed to be several meters too short especially in close range. Negative ranging values in the close proximity of the AP were expected to happen, but this offset of approximately 5 meters resembles the behaviour that was also discovered in [28] with the same Intel AC8260 Wi-Fi chip. This problem was later fixed in a firmware update but before that, a constant value had to be added to the reported distances.

A stationary measurement was done at site A. Five FTM Responders were installed around the room, and the Pixel 2 XL phone was placed on a table where it had line-of-sight (LOS) to all of the APs (Figure 4.3). The Datalogger app was set to record RTT range measurements for five minutes. The averaged estimated distances to each AP were subtracted from the real distances. The average of those offsets was later applied as a common offset value for every AP in site A measurements. The measurement results are shown in table 4.1.

At site B, a calibration measurement was done using the robot. The setup was similar to what is proposed in the calibration guide of the Android open source project (AOSP)

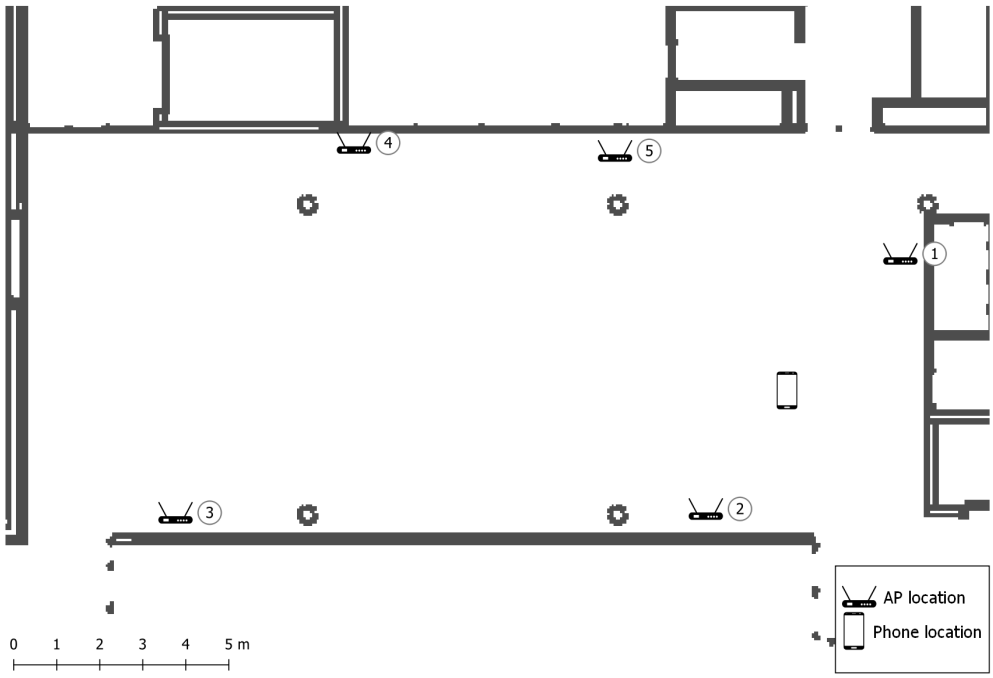


Figure 4.3: Device arrangement in the stationary measurement

Table 4.1: Ranging offsets in the stationary measurement

AP id	Distance (m)		Offset
	Real	Measured	
1	4.21	−0.90	5.11
2	3.22	−1.35	4.57
3	15.00	10.66	4.34
4	11.75	7.87	3.88
5	7.03	1.96	5.07
		Mean	4.59

[39]. The guide suggests a large open space or a corridor without metallic objects and a LOS path of 25 meters between two FTM devices mounted 20 centimeters above the floor. In this setup, two APs were fixed at both ends of a hallway 60 cm above the floor, and the robot carried the smartphone 30 cm above the floor along a straight line between them. A round-trip between the APs was repeated six times, changing the robot orientation by 180 degrees after the first three rounds. The APs had the updated firmware installed, so no constant offset of several meters in the reported ranges was expected. The goal of the calibration measurement was to find out the ranging accuracy and to see if it varies with distance. Also, the difference in ranging accuracy between LOS and NLOS conditions was of interest, as a third access point was partially in line-of-sight during the measurement. Figure 4.4 shows the AP arrangement and the measurement path. The ranging accuracy results are discussed in section 5.1.

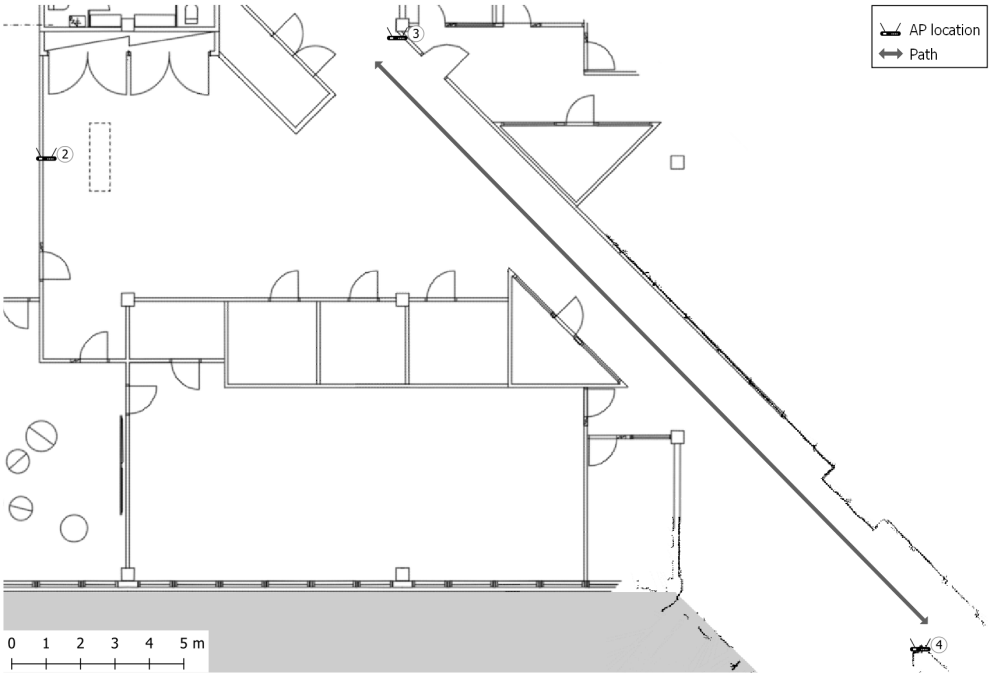


Figure 4.4: Calibration measurement arrangement

4.5 Mobile Measurements

The main portion of the data used for algorithm development came from mobile measurements. Measurement routes around sites A and B were followed with the robot. At site A, the AP arrangement was the same as in the previously discussed calibration measurement. From most locations in the measurement area, the phone had line-of-sight to every AP. The measurement route (Figure 4.5) first wound in between groups of tables, and then ran through the room in a straight line.

At site B, the five APs were farther away from each other with walls in between, resulting in more frequent NLOS conditions. They were installed on desks, stands and walls to heights between 60 and 140 centimeters above floor level. Here, the phone had line of sight to three APs at best, inside a very limited area. The access point locations and the measurement route at site B are shown in figure 4.6.

The routes were driven from start to finish multiple times to ensure good amount of data. For the sake of simplicity of analysis and visualization, single rounds were extracted from the files for algorithm development and performance evaluation. The rest of the data remained as backup and for additional tests.

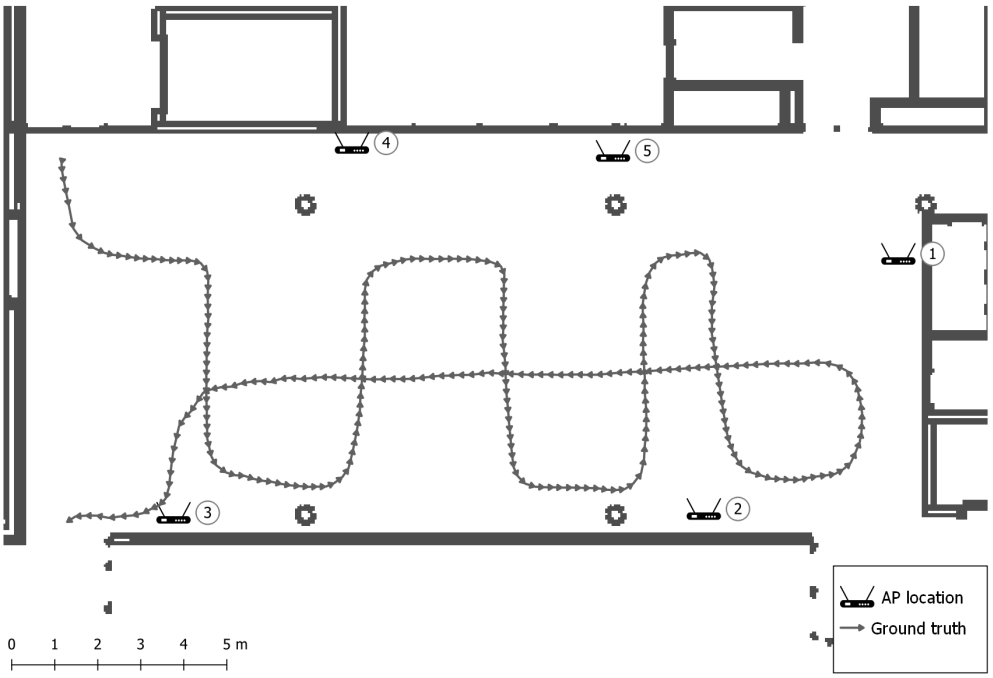


Figure 4.5: Measurement route at site A

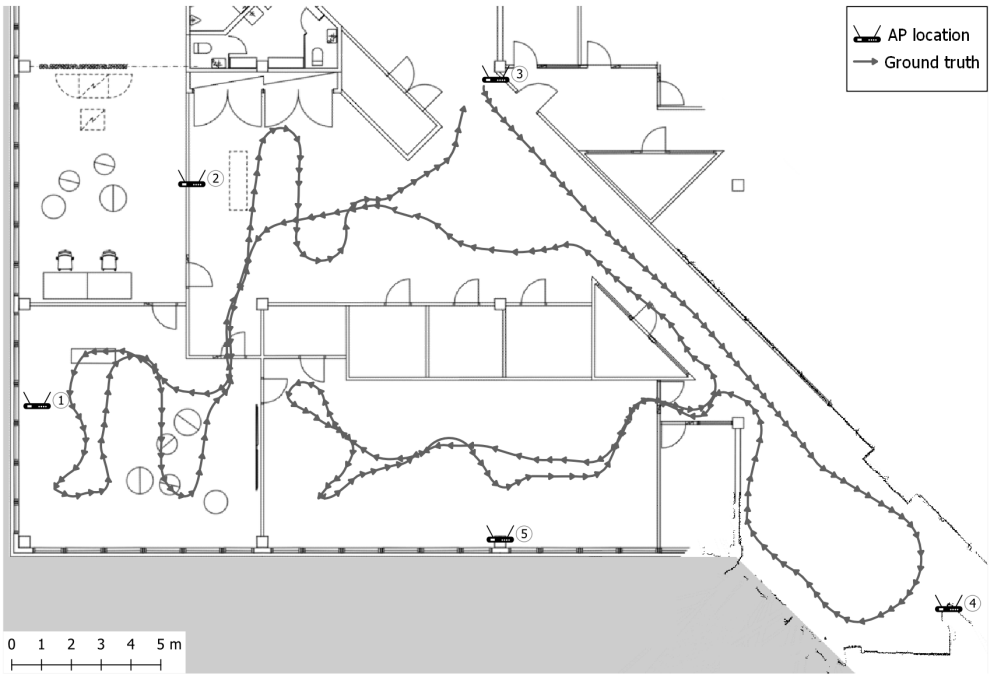


Figure 4.6: Measurement route at site B

Chapter 5

Results and Discussion

After gathering FTM data with the Datalogger app, data processing and statistical analysis were performed. The FTM samples and ground truth locations were combined to same files, which were then run through the positioning scripts. In this chapter, ranging accuracy in LOS and NLOS conditions and the overall positioning accuracy are analyzed.

5.1 Ranging Accuracy

The calibration measurement at site B provided data for evaluating the ranging accuracy of FTM. The AOSP calibration guide [39] was used as a basis for the evaluation. According to the guide, 90% of range estimates of an FTM-enabled Android device should have ranging error of 2 meters or less (the 90th percentile error). The 2-meter tolerance is for the 80 MHz channel bandwidth that was used in all measurements, while the tolerances for 40 and 20 MHz bandwidths are 4 and 8 meters, respectively. The guide also states that a regression line fitted to a chart with true distance on the x-axis and estimated distance on the y-axis should ideally have gradient of 1.0 and offset of 0.0 meters [39].

5.1.1 Line-of-Sight

The data consisted of LOS measurements to access points 3 and 4 (Figure 4.4), with the robot first facing AP 4 and then AP 3. The smartphone was placed horizontally on the front edge of the robot, so some differences in the ranging values were expected between the two robot orientations. These orientations are later referred to as *facing away* and *facing towards* relative to the access point in question. The estimated distance versus the ground truth distance to both APs in both orientations is plotted in figure 5.1.

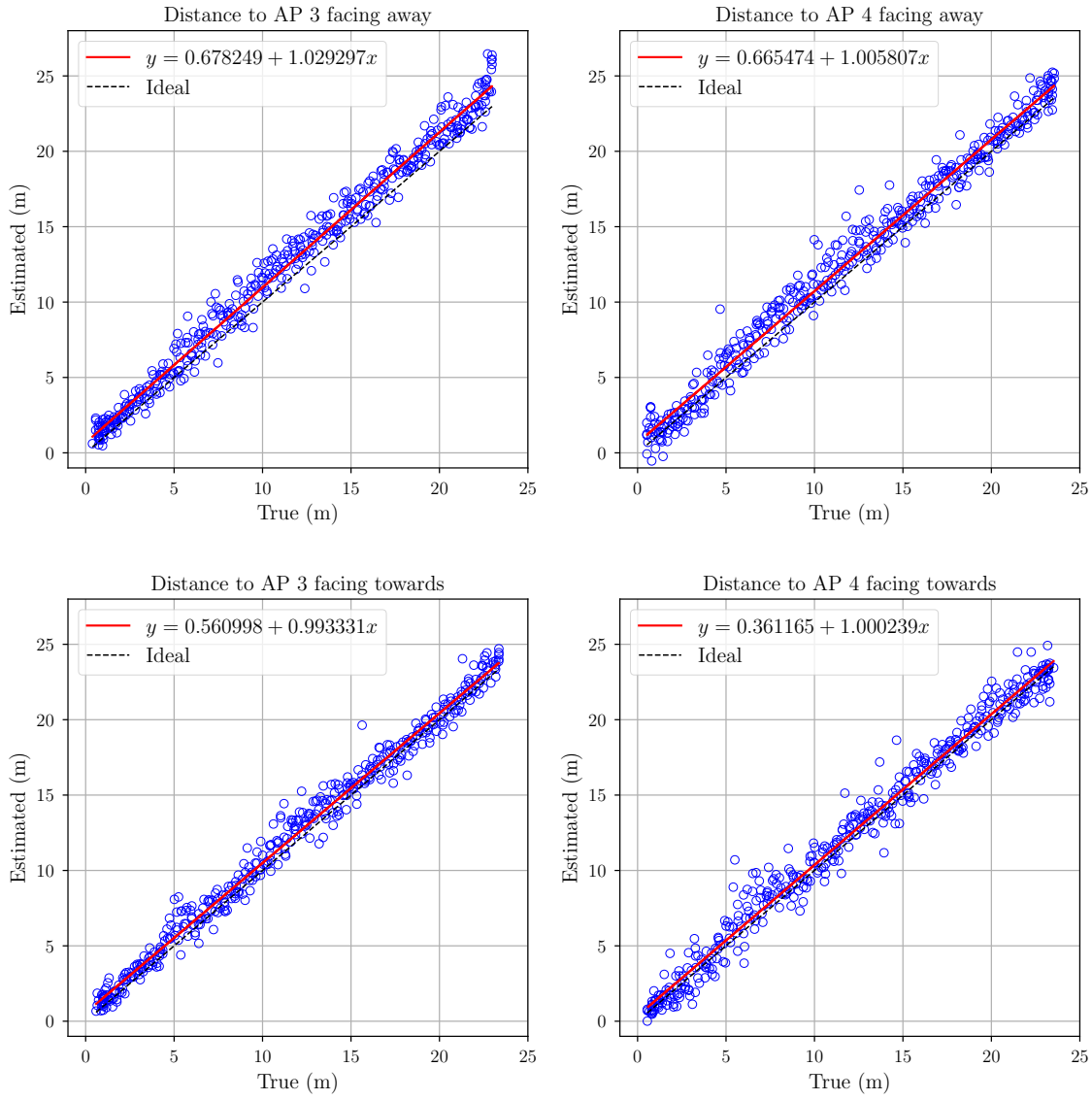


Figure 5.1: Calibration range measurements to two access points

The equations of the regression lines indicate that the ranging system performed closer to ideal when the robot was facing the access point: with both APs, the gradient is closer to 1 and the offset is smaller compared to when the robot was facing away from the AP. In figure 5.2, the ranging error is plotted against true distance; there is no sign of the ranging error increasing with distance.

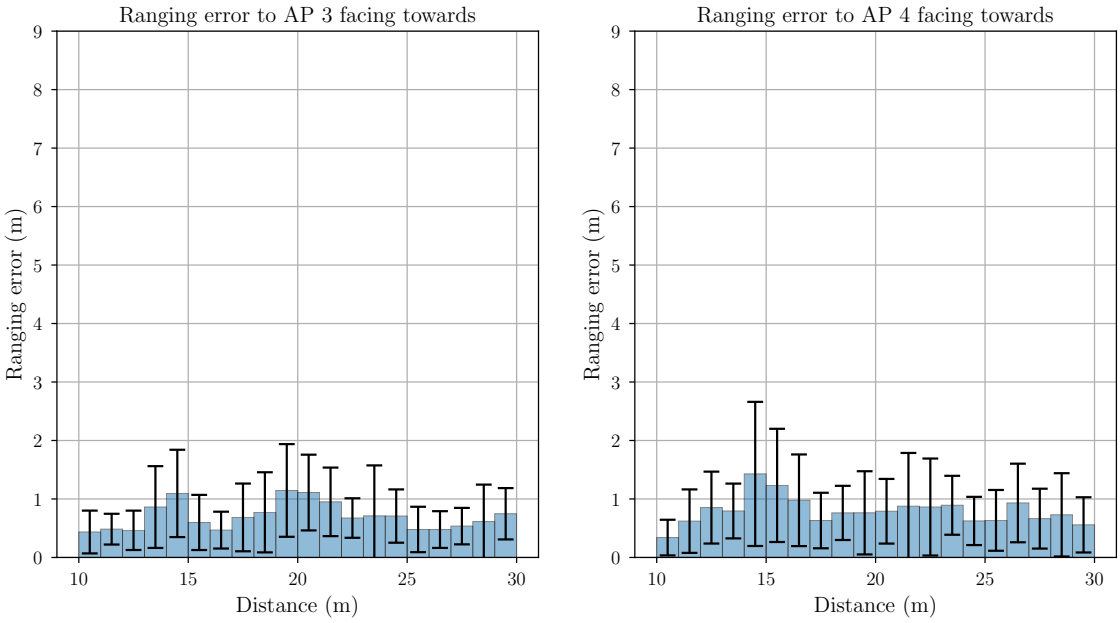


Figure 5.2: Absolute ranging error in LOS conditions (error bars represent ± 1 standard deviation)

The AOSP calibration guide allows deviations from the ideal gradient and offset values if the 90th percentile error stays under 2 meters. Figure 5.3 shows the cumulative distribution function (CDF) of absolute ranging error in the same four cases. The chart shows that when the smartphone was facing away from AP 3, the 90th percentile error exceeded 2 meters (2.17 m), while in other cases the ranging error stayed within the 2-meter tolerance. The statistics of the LOS calibration measurement are shown in table 5.1.

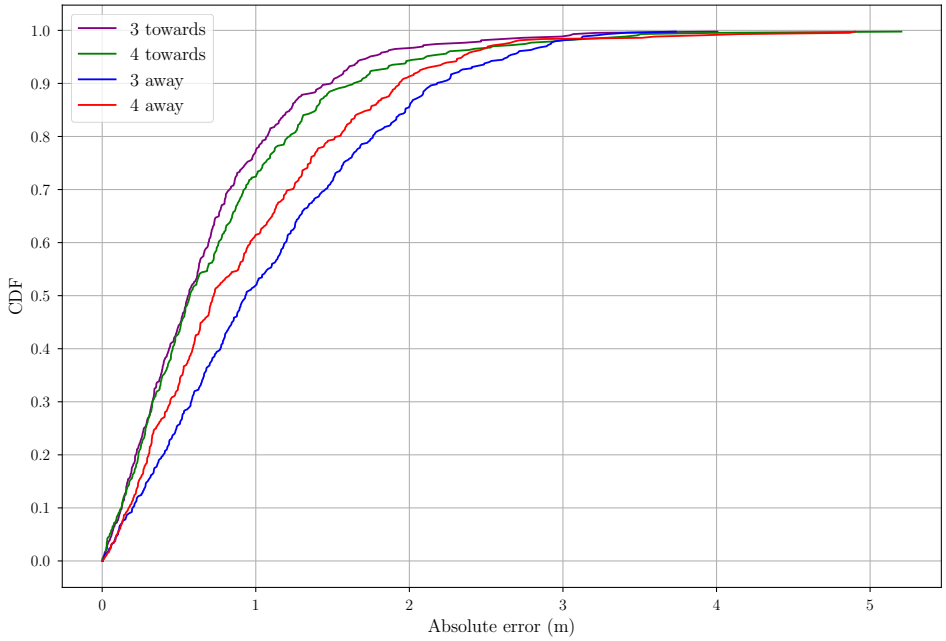


Figure 5.3: CDF of absolute ranging error

Table 5.1: Ranging error statistics in the calibration measurement

AP id	Orientation	Absolute ranging error (m)		
		Mean	Median	90th percentile
3	Towards	0.69	0.56	1.45
	Away	1.10	0.93	2.17
4	Towards	0.77	0.57	1.63
	Away	0.94	0.72	1.93

5.1.2 Non-Line-of-Sight

NLOS data was available from the same measurement as the LOS data by inspecting range estimates to access point 2 (Figure 4.4). Along the 23-meter measurement path there was a section of 7 meters with LOS to AP 2, while the rest of the path was in NLOS. Within this LOS section, the true distance to AP 2 was approximately between 11 and 16 meters. At true distances of 10–11 and 16–29 meters the robot was in NLOS with the AP. In figure 5.4, the samples from the LOS section can be seen to be closer to the ideal reference line. The linear regression lines diverge from the ideal significantly compared to the LOS-only measurement.

Figure 5.5 shows the distribution and variability of the ranging error along the measurement path. The height of the bars in the plot represent the mean absolute ranging error, and the error bars span 1 standard deviation above and below the mean. Especially in the case where the phone was facing towards the AP, the bars between 12 and 16 meters (the LOS section) are clearly lower and the standard deviations smaller. In the other direction, when the phone was facing away from the AP, this distinguishable section with less ranging error spans from 12 to 19 meters. In contrast to the LOS measurement, the ranging error increases with distance in NLOS conditions.

5.2 Positioning Performance

The mobile measurements at sites A and B provided data for testing the two positioning algorithms, NLS and UKF. Single rounds of both measurement routes were selected as input data for both algorithms, and the positioning accuracy of the output esimates was analyzed. The goal was to reach accuracy sufficient for indoor navigation and find out, if meter-level accuracy is possible with FTM as claimed.

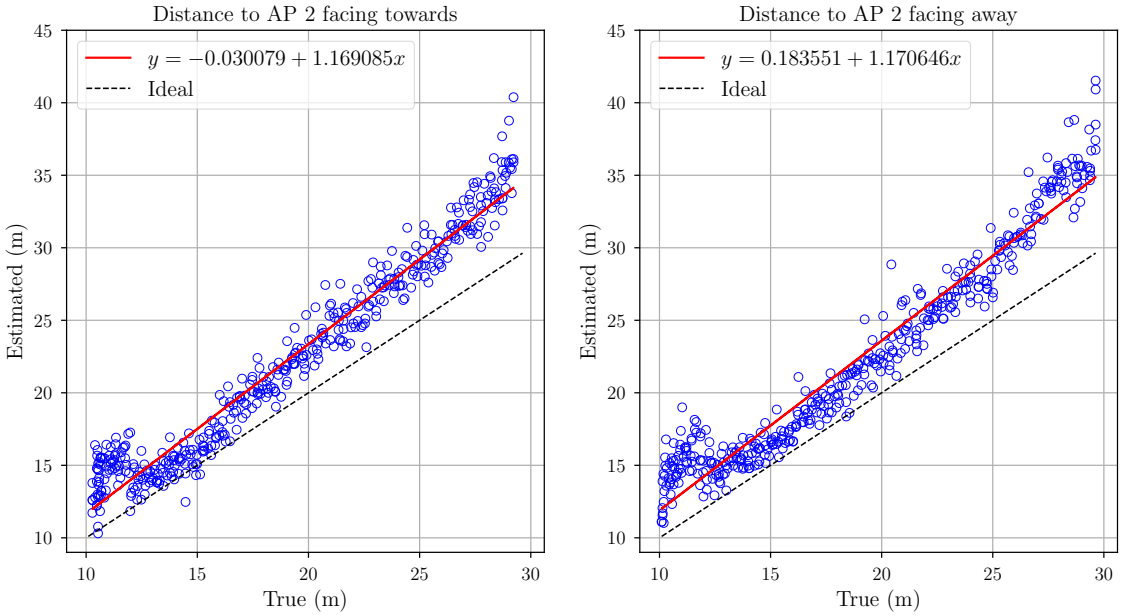


Figure 5.4: Range measurements in partial NLOS conditions

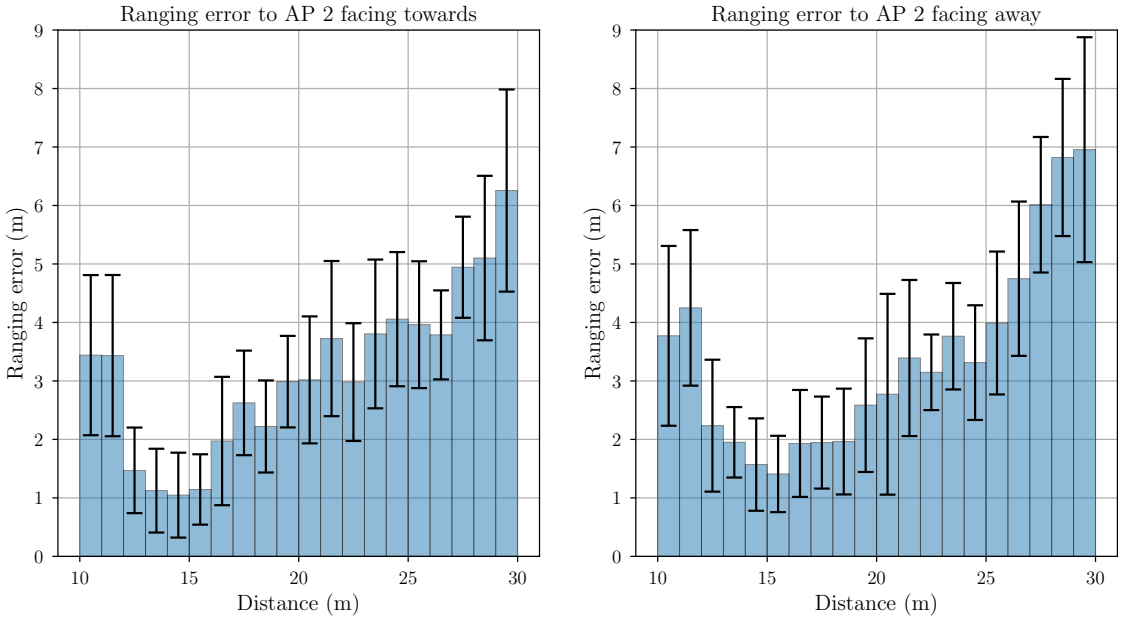
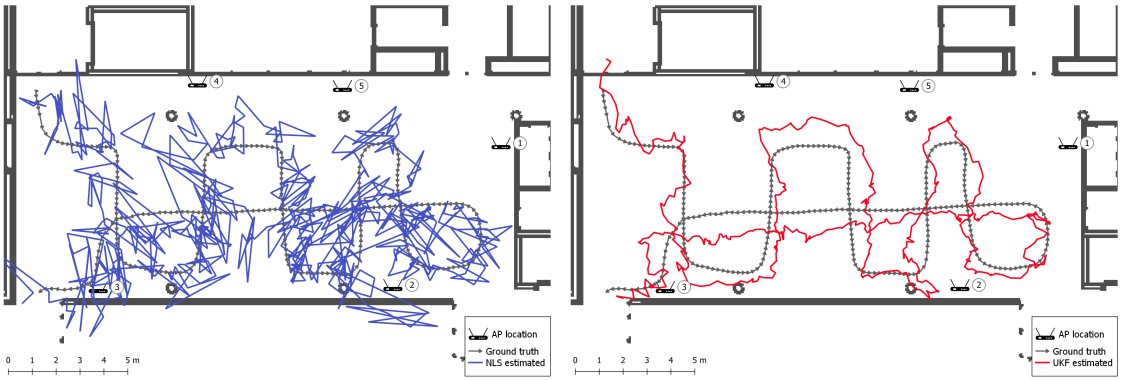


Figure 5.5: Absolute ranging error in partial NLOS conditions (error bars represent ± 1 standard deviation)

5.2.1 Site A: Favourable Environment with Plenty of LOS Conditions

A large, open room such as the one at site A provides good conditions for radio wave-based ranging and positioning due to ubiquitous line-of-sight to multiple APs surrounding the room. After acquiring the ranging offset correction from the stationary measurement discussed in section 4.4, the corrected range measurements along the mobile measurement route were input to the positioning algorithms. The resulting estimated trajectories produced by both algorithms are shown in figure 5.6.



(a) NLS (b) UKF

Figure 5.6: Position estimates at site A

The NLS algorithm produces a jagged trajectory, since the position estimate is calculated based on solely the most recent set of range measurements. The noise in the measurements can cause the estimate to be farther away from the previous estimate than the real travelled distance would suggest. However, since the previous estimate is used as a starting point for the optimization algorithm, there is a risk of divergence due to the algorithm converging to a local minima far away from the ground truth. In this case, there is no major divergence, and the estimate follows the ground truth path fairly well. The 90th percentile and mean positioning errors were 1.89 and 1.01 meters, respectively.

The UKF utilizes a constant velocity kinematic model to predict the position, and

updates the prediction with new range measurements. This results in a smoother trajectory estimate, since the noisy measurements are not defining the estimate alone. The filter properties must be configured properly in order to achieve a good balance between the predictions and measurements. The UKF produced more accurate output estimates than the NLS algorithm, with 1.16 m 90th percentile and 0.71 m mean positioning errors. Figure 5.7 shows the CDF curve comparison of the accuracies of the algorithms.

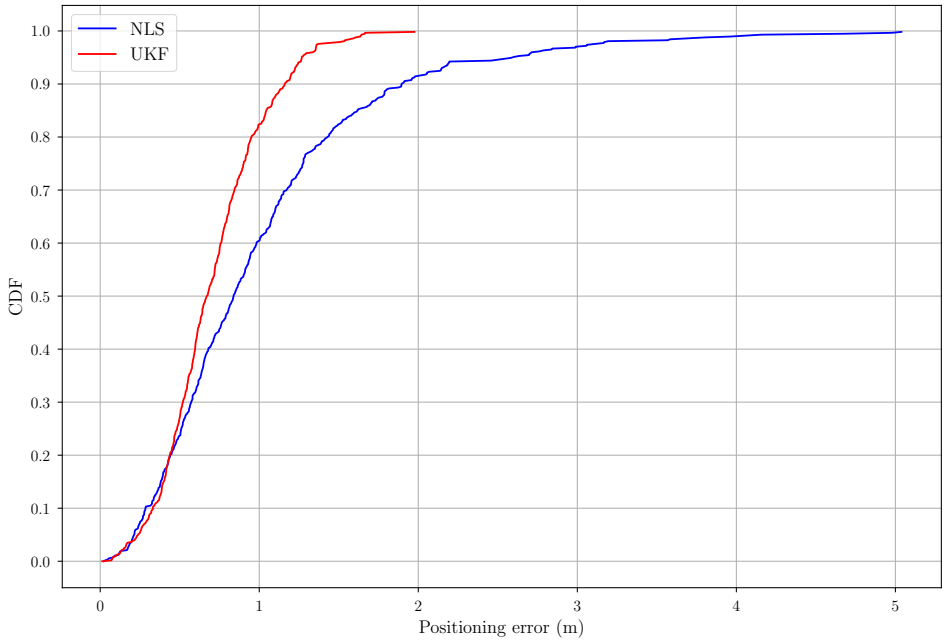


Figure 5.7: CDF of positioning error at site A

These results confirm that meter-level positioning accuracy is achievable with FTM. This specific scenario, however, is not representing a feasible indoor positioning solution that utilizes existing Wi-Fi infrastructure. The five access points surrounding one room provide good conditions for positioning, but they are clearly more than enough for wireless communications. The next tests were done at site B with a more realistic Wi-Fi AP constellation for an office environment.

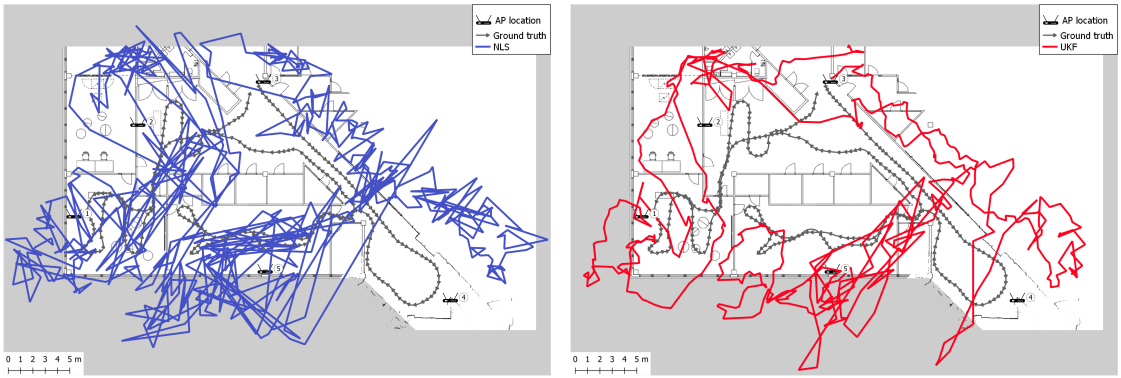
5.2.2 Site B: Challenging Environment with Mostly NLOS Conditions

At site B, the access point arrangement was not as dense as it was at site A. Instead, the scenario was closer to real world use, as the APs were placed in multiple rooms around the office floor. The environment provided a good opportunity to evaluate the effect of non-line-of-sight signal propagation on FTM-based positioning. As discussed in section 5.1.2, NLOS propagation causes ranging error that increases with distance. The measurement data from site B was used to study the positioning accuracy with the two algorithms, and with or without range corrections obtained from the calibration measurement.

The estimated trajectories in figure 5.8 indicate that by using only raw FTM range estimates, the positioning result is affected negatively by the NLOS conditions. Since the distances to APs in NLOS are usually overestimated, the position estimates tend to concentrate towards the edges of the AP constellation. The effect can be clearly seen in the corridor that was used in the calibration: the estimated trajectory goes along the corridor, but on the wrong side of the wall. In that corridor, APs 1, 2 and 5 are in NLOS, and as the ranges to those APs get overestimated, the positioning solution converges farther away from them.

The difference between the algorithms is the same as in site A measurements: the UKF produces a smoother and more accurate trajectory estimate. The negative effects of NLOS ranging are similar between the algorithms. The overestimated measurements have a direct impact on the estimates of the NLS algorithm, and the kinematic model predictions of the UKF cannot overcome the problem. With no corrections applied to the range estimates, the 90th percentile and mean positioning errors were 9.17/4.95 m (NLS) and 7.57/4.65 m (UKF).

In order to mitigate the effects of NLOS ranging, a correction was used to the estimated ranges. As a simple principle, all measured ranges above 10 meters were considered to be subject to NLOS ranging error. Those samples were corrected using a formula



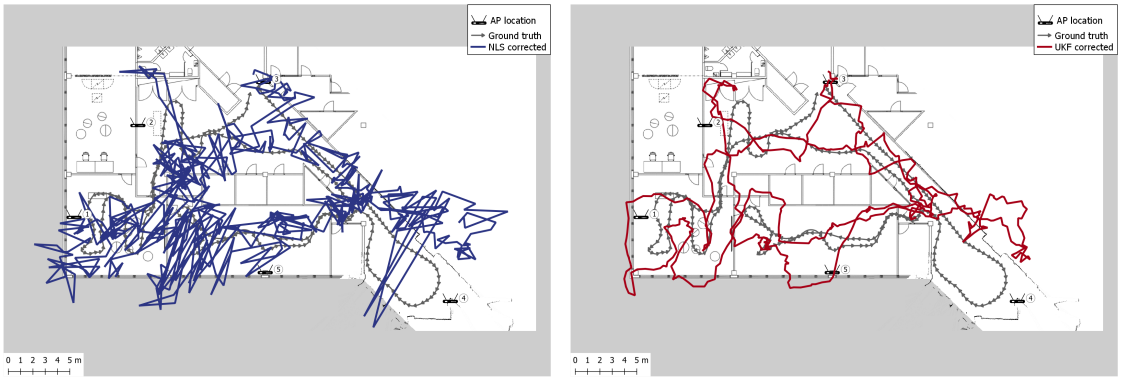
(a) NLS (b) UKF

Figure 5.8: Position estimates at site B using raw range measurements

based on the regression line in the second calibration measurement plot of figure 5.4:

$$d_{true} = \frac{d_{measured} - 0.18}{1.17} \tag{5.1}$$

Figure 5.9 shows the improved position estimates. Positioning accuracy in the previously problematic corridor is now considerably better, and the estimates stay on the correct side of the wall. In general, the estimates of both algorithms are less spread out compared to the previous results.



(a) NLS (b) UKF

Figure 5.9: Position estimates at site B using corrected range measurements

The proximity of AP 4 still remains a challenging area where the loop is incorrectly

estimated. In the UKF estimate, the manoeuvres of the robot are tracked more precisely especially near APs 1 and 2. Overall, applying the correction to range estimates over 10 meters halved the positioning error: the NLS algorithm now had 4.49 m 90th percentile and 2.41 m mean error, while the UKF had errors of 3.98/2.07 m. CDF comparison of the positioning error of both algorithms with raw and corrected range measurements is shown in figure 5.10.

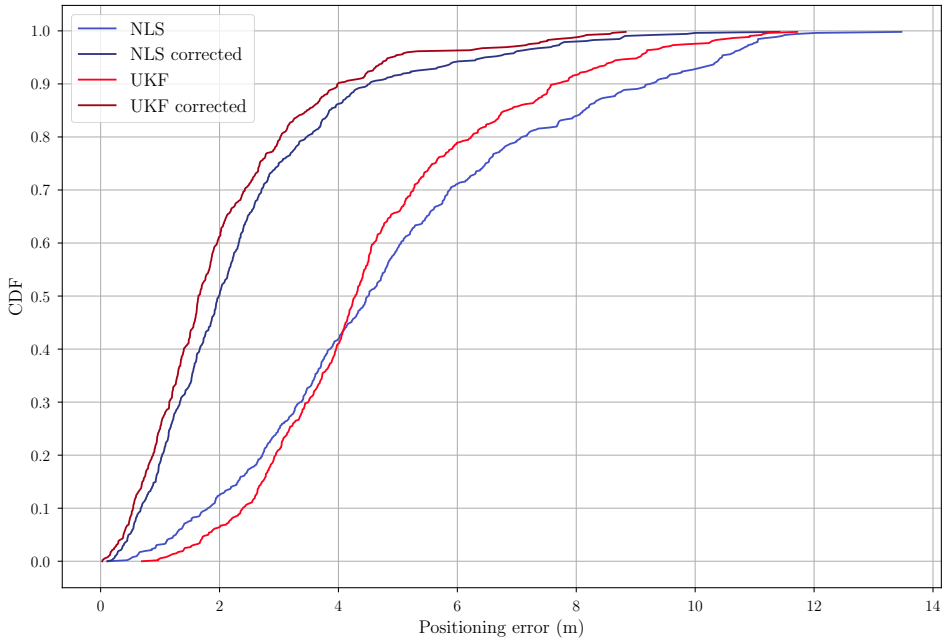


Figure 5.10: CDF of positioning error at site B

Although meter-level positioning accuracy was not achieved at site B, it can be said that FTM-only positioning is suitable for detecting a room or part of a room. With the simple NLOS mitigation method used here in this five access points setting, the positioning solution generally lies within the correct room. In this type of office environment, the UKF estimates are sufficient for indoor navigation i.e. finding a room based on the smartphones location on the map.

5.3 Discussion

In this chapter, the accuracies of FTM-based range estimates and two positioning algorithms were evaluated. The main interests were the possibilities and limitations of the FTM technology. Ranging accuracy in line-of-sight conditions was found to be consistent, since the estimated range only had a small constant offset from the true value. The device combination fulfilled the LOS ranging accuracy requirements of the AOSP calibration guide by having less than 2 meters 90th percentile ranging error.

The results showed that in non-line-of-sight, the ranging error increases with distance. Multipath propagation causes the range to be estimated from a signal that has not travelled through the direct path, making the estimate too large. The variance in the estimates also increases with distance, so they become less reliable. A correction derived from the NLOS calibration measurement was later used to enhance positioning accuracy, by decreasing estimates above 10 meters to account for the ranging error.

The positioning results are summarized in table 5.2. Meter-level positioning accuracy was achieved in the favourable environment at site A. Since the ranging error is constant at various distances in LOS conditions, samples from both near and far APs are useful in positioning. In the absence of NLOS conditions, the variation in the measurements is small, making the range estimates reliable. The claim of meter-level positioning accuracy is plausible at least in this specific environment with a dense AP arrangement.

At site B, the mean positioning error was over 2 meters. The result was achieved by applying a correction to range measurements over 10 meters, in order to account for overestimated ranges caused by multipath propagation. The majority of ranging samples were measured from a NLOS channel due to the realistic placement of the access points (one per room). So, without the correction the mean error was almost five meters, which renders the system unusable for indoor navigation.

WLAN signals are prone to fading and reflections from indoor obstacles, so the FTM ranging performance decreases in NLOS. It is also worth noting that at site B, the most

Table 5.2: Positioning error statistics

Site	Algorithm	Positioning error (m)		
		Mean	Median	90th percentile
A	NLS	1.01	0.84	1.89
	UKF	0.71	0.67	1.16
B	NLS	4.95	4.49	9.17
	NLS corrected	2.41	1.98	4.49
	UKF	4.65	4.28	7.57
	UKF corrected	2.07	1.64	3.98

challenging area was at the most extreme perimeter of the AP constellation. There, the positioning accuracy is greatly affected by dilution of precision (DOP), because four of the five APs are in the same general direction (within a 40°sector when viewed from AP 4). It is therefore important to plan the AP placements carefully in order to get good signal geometry around the site.

A more sophisticated method for detecting and mitigating NLOS ranging error would improve the result. Data from the motion sensors of the smartphone could make the estimation more accurate. The UKF implementation is suitable for sensor fusion by using e.g. step detection data. Also, site-specific information from a floor plan could be utilized for keeping the estimates inside accessible areas. Improvements are justified if the use case requires more accurate positioning, for example in real-time indoor navigation. If only room-level accuracy is needed, then FTM range measurements alone are sufficient.

5.4 Comparison with Other Technologies

The measurements at site B provided data for comparing the FTM positioning system with three other methods: RSSI-based trilateration, WLAN fingerprinting, and ultra-wideband

(UWB) trilateration. The RSSI values used for range estimation were taken from the FTM samples, while the WLAN fingerprinting database (three rounds of route in figure 4.6 reversed) and test data (one round of the measurement route) were gathered with a separate smartphone on the robot. The UWB device setup was similar to the FTM system; five UWB anchors were co-located with the FTM APs, and one UWB tag was installed on the robot. Similarly to FTM, the UWB system provides ToF-based range estimates.

FTM and its time-based ranging provides a clear improvement to signal strength-based ranging, as can be seen from the CDF curves in figure 5.11. Compared to the 3.98 m 90th percentile positioning error of FTM, the RSSI-based trilateration method had 11.19 m 90th percentile error. An empirical path loss model that was based on site-specific calibrations was used, but the fluctuation of the received signal strength caused the range estimates to be unreliable. Close to FTM was WLAN fingerprinting with 4.26 m 90th percentile error. Fingerprinting has the benefit of using existing WLAN infrastructure for good positioning accuracy, but it requires extensive site surveys both before and after deployment. UWB managed the NLOS conditions better than FTM with 0.91 m 90th percentile error. The wider bandwidth of UWB enables better multipath resolution, which results in more accurate range estimates [5]. Although UWB devices are popular in indoor positioning, UWB radios are uncommon in smartphones.

Based on the data gathered at site B, FTM provides positioning accuracy that is comparable to WLAN RSSI fingerprinting. Room-level positioning accuracy is achievable with FTM without the need for a sample database. Time-based ranging works for positioning considerably better than signal strength-based range estimation. FTM is not as resilient to NLOS conditions and multipath propagation as UWB due to narrower signal bandwidth, so NLOS detection and -mitigation are needed to enhance the positioning accuracy. The developed system was also compared with image-based positioning and RSS fingerprinting inside the one-room area at site A in [40]. While FTM with the NLS algorithm did not achieve the sub-meter positioning accuracy of the image-based system,

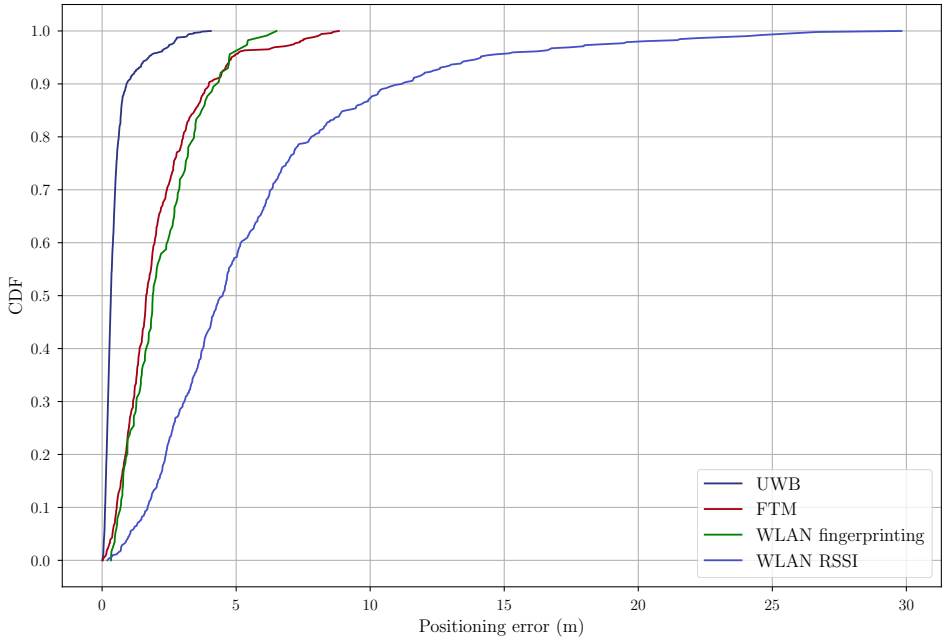


Figure 5.11: FTM positioning performance against other technologies at site B

it had better accuracy than LTE and Wi-Fi fingerprinting.

Chapter 6

Conclusions

This thesis presented an indoor positioning system featuring a smartphone that uses Wi-Fi fine timing measurement (FTM). FTM is an emerging technology, whose properties were discussed in this thesis. FTM allows Wi-Fi devices to measure distance between each other. The distance is derived from precise round-trip time (RTT) measurements in an FTM frame exchange procedure. The new level of precision in these measurements is why FTM is expected to be a better solution for range estimation (with Wi-Fi devices) compared to previously popular signal strength.

The developed system utilizes two alternative Android applications for gathering range measurements to Wi-Fi access points, and two different algorithms that convert the measurements into position estimates. The purpose was to study the positioning accuracy of an FTM-based system and see if meter-level accuracy is possible in practice. This was verified by performing measurements at two different office sites. In the LOS measurement at site A, positioning errors of 0.72 m (mean) and 1.17 m (90th percentile) were achieved in an open space surrounded by five FTM APs, thereby proving meter-level accuracy possible. However, in a larger office space at site B where the APs were installed in different rooms, the positioning errors were 2.07 m (mean) and 3.98 m (90th percentile). The increased positioning error resulted from the prevalence of non-line-of-sight (NLOS) signal propagation conditions. Among the implemented two algorithms,

the Unscented Kalman Filter (UKF) provided the best results. The results proved that FTM provides more reliable range estimates than signal strength, and enables the same level of positioning accuracy as Wi-Fi fingerprinting with less overhead in establishment and maintenance.

6.1 Future Work

To gain more accurate indoor location for a smartphone, the FTM ranges could be used in sensor fusion with other data sources of the device, such as orientation sensor and step detector. Deployment of a larger system across multiple floors of a building also requires research on floor detection. Combining the past research on fingerprinting techniques with FTM, i.e. using a database of ranges instead of RSSI values, is a topic to study in the future. If FTM-enabled APs gain popularity in WLAN infrastructures, their configured coordinates together with e.g. crowdsourced distance databases could enable universal indoor positioning applications.

References

[1] G. Dedes and A. G. Dempster, "Indoor GPS positioning - challenges and opportunities," in *VTC-2005-Fall. 2005 IEEE 62nd Veh. Technol. Conf.*, vol. 1, pp. 412–415, Citeseer, 2005.

[2] "IEEE standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements - part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, 2016.

[3] Wi-Fi Alliance, "Wi-Fi Location." https://www.wi-fi.org/discover-wi-fi/wi-fi-location, 2019. Accessed: 04.01.2019.

[4] Android Open Source Project, "Wi-Fi location: ranging with RTT." https://developer.android.com/guide/topics/connectivity/wifi-rtt, 2018. Accessed: 04.01.2019.

[5] P. D. Groves, *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*. Norwood: Artech House, 2013. ID: 1531533.

[6] "IEEE standard definitions of terms for antennas," *IEEE Std 145-1993*, pp. 1–32, 1993.

[7] R. Mautz, *Indoor positioning technologies*. PhD thesis, ETH Zurich, 2012.

[8] "Propagation data and prediction methods for the planning of indoor radio communication systems and the radio local area networks in the frequency range 900 MHz to 100 GHz," *Rec. ITU-R P.1238-2*, 2001.

[9] J. S. Seybold, *Introduction to RF propagation*. John Wiley & Sons, 2005.

[10] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of wireless indoor positioning techniques and systems," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 37, no. 6, pp. 1067–1080, 2007.

[11] E. G. Bakhoum, "Closed-form solution of hyperbolic geolocation equations," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 42, no. 4, pp. 1396–1404, 2006.

[12] D. Munoz, F. B. Lara, C. Vargas, and R. Enriquez-Caldera, *Position Location Techniques and Applications*. Elsevier Science, 2009.

[13] P. Bahl and V. N. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in *INFOCOM 2000. 19th Annu. Joint Conf. IEEE Comput. and Commun. Societies. Proc.*, vol. 2, pp. 775–784, IEEE, 2000.

[14] S. Gezici, "A survey on wireless position estimation," *Wireless pers. commun.*, vol. 44, no. 3, pp. 263–282, 2008.

[15] W. S. Murphy and W. Hereman, "Determination of a position in three dimensions using trilateration and approximate distances," *Dept. of Math. and Comput. Sci., Colorado School of Mines, Golden, Colorado, MCS-95*, vol. 7, p. 19, 1995.

[16] I. D. Coope, "Reliable computation of the points of intersection of n spheres in R n," *ANZIAM J.*, vol. 42, 2000.

[17] B.-C. Liu, K.-H. Lin, and J.-C. Wu, "Analysis of hyperbolic and circular positioning algorithms using stationary signal-strength-difference measurements in wireless communications," *IEEE Trans. Veh. Technol.*, vol. 55, no. 2, pp. 499–509, 2006.

[18] R. E. Kalman, "A new approach to linear filtering and prediction problems," *J. Basic Eng.*, vol. 82, no. 1, pp. 35–45, 1960.

[19] R. E. Kalman and R. S. Bucy, "New results in linear filtering and prediction theory," *J. Basic Eng.*, vol. 83, no. 1, pp. 95–108, 1961.

[20] M. S. Grewal and A. P. Andrews, "Applications of Kalman filtering in aerospace 1960 to the present [historical perspectives]," *IEEE Control Syst.*, vol. 30, no. 3, pp. 69 – 78, 2010.

[21] A. Shareef and Y. Zhu, "Localization using extended Kalman filters in wireless sensor networks," in *Kalman Filter Recent Advances and Applications*, InTech, 2009.

[22] S. Konatowski and A. T. Pieni, "A comparison of estimation accuracy by the use of KF, EKF & UKF filters," *WIT Trans. Model. Simul.*, vol. 46, 2007.

[23] K. Radnosrati, F. Gunnarsson, and F. Gustafsson, "New trends in radio network positioning," in *2015 18th Int. Conf. Inf. Fusion (Fusion)*, pp. 492–498, IEEE, 2015.

[24] J. Polk, M. Linsner, M. Thomson, and B. Aboba, "Dynamic host configuration protocol options for coordinate-based location configuration information," RFC 6225, RFC Editor, 2011.

[25] H. Schulzrinne, "Dynamic host configuration protocol (DHCPv4 and DHCPv6) option for civic addresses configuration information," RFC 4776, RFC Editor, 2006.

[26] A. Günther and C. Hoene, "Measuring round trip times to determine the distance between WLAN nodes," in *Int. Conf. Res. in Netw.*, pp. 768–779, Springer, 2005.

[27] L. Schauer, F. Dorfmeister, and M. Maier, "Potentials and limitations of WIFI-positioning using time-of-flight," in *Int. Conf. Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–9, 2013.

[28] M. Ibrahim, H. Liu, M. Jawahar, V. Nguyen, M. Gruteser, R. Howard, B. Yu, and F. Bai, "Verification: Accuracy evaluation of WiFi fine time measurements on an open platform," in *Proc. 24th Annu. Int. Conf. Mobile Comput. and Netw.*, pp. 417–427, ACM, 2018.

[29] L. Banin, U. Schatzberg, and Y. Amizur, "WiFi FTM and map information fusion for accurate positioning," in *Int. Conf. Indoor Positioning and Indoor Navigation (IPIN)*, 2016.

[30] IEEE P802.11 – Task Group AZ, "Status of IEEE 802.11az next generation positioning (NGP)." http://www.ieee802.org/11/Reports/tgaz_update.htm, 2019. Accessed: 05.04.2019.

[31] L. Banin, O. Bar-Shalom, N. Dvorecki, and Y. Amizur, "High-accuracy indoor geolocation using collaborative time of arrival (CToA)," *Intel White Paper*, 2017.

[32] I. Pefkianakis and K.-H. Kim, "Accurate 3D localization for 60 GHz networks," in *Proc. 16th ACM Conf. Embedded Networked Sensor Syst.*, SenSys '18, (New York, NY, USA), pp. 120–131, ACM, 2018.

[33] Compulab Ltd, "Yocto indoor positioning FTM responder (FTMR)." https://fit-iot.com/web/product/indoor-positioning-ftm-responder-ftmr/, 2019. Accessed: 17.7.2019.

[34] V. Agafonkin, "Leaflet, an open-source JavaScript library for mobile-friendly interactive maps." https://leafletjs.com/index.html, 2017. Accessed: 20.05.2019.

[35] W. McKinney, "Data structures for statistical computing in Python," in *Proc. 9th Python in Sci. Conf.* (S. van der Walt and J. Millman, eds.), pp. 51 – 56, 2010.

[36] R. Labbe, "Kalman and Bayesian filters in Python." https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python, 2015. Accessed: 12.04.2019.

[37] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 90–95, 2007.

[38] QGIS Development Team (YEAR), "QGIS geographic information system. Open source geospatial foundation project." http://qgis.osgeo.org, 2019. Accessed: 11.06.2019.

[39] Android Open Source Project, "Wi-Fi RTT (IEEE 802.11mc)." https://source.android.com/devices/tech/connect/wifi-rtt#calibration, 2018. Accessed: 04.01.2019.

[40] S. Horsmanheimo, S. Lembo, L. Tuomimaki, S. Huilla, P. Honkamaa, M. Laukkanen, and P. Kemppi, "Indoor positioning platform to support 5G location based services," in *2019 IEEE Int. Conf. on Commun. Workshops (ICC Workshops)*, pp. 1–6, 2019.