Final Year Project

Indoor Positioning with Smartphones

Eric Whye

Student ID: 19336881

A thesis submitted in part fulfilment of the degree of

BSc. (Hons.) in Computer Science

Supervisor: Professor Chris Bleakley and Associate Profesor Neil Hurley

Project Repository

UCD School of Computer Science
University College Dublin

May 20, 2023

Table of Contents

Chapter 1: Project Specification

The objective of this project is to develop a proof of concept indoor positioning system for smartphones, using Wi-Fi and inertial sensor data. The project will involve a literature survey to identify the best previous work on building such a system; collection of an indoor positioning dataset for the UCD Science Centre; development of software for estimation of position using the data; evaluation of the accuracy of the software; and exploration for various alternative algorithms for positioning.

Currently, smartphones cannot accurately estimate their position indoors, due to attenuating of the GPS satellite signal by the building fabric. This data can be useful for future applications such as acquiring indoor building traffic, or locating oneself in an airport.

The system will be evaluated based on the accuracy of static location and mobile tracking tests, while fulfilling other criterion as well.

The scope of the project will reside in the O'Brien Centre for Science at University College Dublin. A dataset relating to indoor positioning will be formed and be used to conduct location estimate and mobile tracking tests. The tests will consist of gathering data onsite while standing still, walking etc. The system will perform localisation in non-real-time i.e. an unlimited amount of post-processing time will be allowed. This is to prove the performance of the algorithms and techniques employed, rather than the computational capabilities of the hardware. Implementation and evaluation of other alternative techniques will be explored as well.

1.1 Core Objectives

1. Develop an indoor positioning system that takes estimate Wi-Fi location data as input, compares it to collected Wi-Fi data about the test environment, and outputs an estimate location.
2. Further develop the system to incorporate tracking features.
3. Introduce data fusion technology in the form of inertial sensor data, that can be found and generated from smartphones to improve accuracy of static location and tracking tests.
4. Introduce other testing variables such as spinning on the spot, running, walking backwards etc. to measure how much the human body interferes with Wi-Fi signal propagation and detection
5. Evaluation of the system
   (a) Achieve a mean accuracy of < 2.0m.
   (b) Achieve a similar accuracy after 150m of continuous tracking.
   (c) Test the robustness and reliability of the system, for example, moving along the edge of the map, walking in a quick zig-zag pattern, sidestepping and changing direction many times.

All data will be collected onsite and evaluated later.

1.2 Advanced Objectives

1. Incorporate multiple floors in the building environment dataset, using a barometer sensor to aid in calculating floor level for locating/tracking.
2. Develop an orientation-specific RSS grid for the offline phase described in Section 3.3.3
3. Widen the environment dataset to incorporate most of the UCD Science building.
4. Develop an orientation model, so that heading data can be acquired in any position.
5. Develop a personalisation model that adheres to the user's walking footprint for better step detection and step length accuracy.
6. Explore the change in performance when using methods accounting for uncalibrated sensors.

1.3 Datasets

Collection of an RSS fingerprinting map of the Science Centre will be conducted. The density of the radio map will be 2m-2.5m between each grid point depending on the size of the test area.

Chapter 2: **Introduction**

Farid *et al.* [1] describe how an indoor positioning system can be defined as any system that can provide a precise location of a user in a closed structure, be it an office building, an airport, a school etc. The reason why indoor and outdoor positioning are separately categorised is because they come with different challenges, and solving those challenges require different techniques.

This project will focus on wireless indoor positioning i.e. using external information about the environment to better determine location. It will also focus on the challenges that are associated with the technology, and the variety of different techniques studied that attempt to combat these problems.

This will be done by discussing the following points:

- The different positioning techniques and why Wi-Fi is the most suitable technology for this project.
- An in-depth explanation of how Wi-Fi indoor positioning is carried out.
- The usage of smartphone sensors to attain more data and enhance accuracy.
- The methods used to combine different sources of data (such as the above) into a valid output.
- The implementation I plan to do based based on the research of the aforementioned points.

Chapter 3: Background Research

This section details the research conducted in relation to indoor positioning relevant to this project. There have been many studies covering different types of positioning techniques including GPS, RFID etc. The main focus of this report will be on Wi-Fi based indoor positioning; its history, the current state-of-the-art and most effective techniques, and why it is much more effective for this project's specification as opposed to GPS, RFID, etc.

The challenges that come with indoor positioning are a lack of Line-Of-Sight (LOS), many obstacles, doors, human beings etc. that can attenuate, reflect and otherwise dissipate electromagnetic signals. Acquiring a precision of 50 metres is also useless as a viable commercial product [2].

Although the project specification constrains implementation to use of Wi-Fi RSS fingerprinting, it is useful to compare the advantages and disadvantages of other positioning techniques, which shall be mentioned briefly. Based on the research of [1, 3], I have compiled metrics for evaluating different positioning techniques.

- **Accuracy**
  The accuracy, or distance error between the estimated location and the actual location. The higher the accuracy, the better the system.
- **Scalability**
  Scalability is a necessity to prove a system's viability in a real-life application. The system must be able to be applied to a wide variety of different environments, and be able to scale up to a larger coverage area without a performance hit, or a drastic increase in complexity.
- **Reliability**
  The system must reliably calculate a good estimated location in the environment under most circumstances, for example an emergency such as a firefighting scenario where vision is greatly impaired, and smoke may disrupt certain signals.
- **Cost**
  A system without large upfront cost and maintenance would be desirable, which would enable it to be easily and feasibly adopted to different applications.

3.1 Positioning Techniques

Here are compiled a few popular positioning techniques that have been studied and some developed into commercial products that are in use today. They shall be discussed with the above metrics in mind to evaluate their strengths and weaknesses.

3.1.1 GPS

Global-Positioning-System (GPS) is the most popular navigation system to find locations and positions of objects [1]. The problem with GPS as a consideration for indoor environments, is

the increased attenuation through the building fabric. It is possible through the use of expensive state-of-the-art GPS receivers, though the ones in smartphones are not adequate [4].

3.1.2 RFID

Radio frequency identification (RFID) uses radio signals to locate deployed active or passive RFID tags on objects or the surrounding environment [3]. RFID technology is used in a wide range of applications including locating people, automobile assembly industry, warehouse management, supply chain network, and assets without the need of LOS [1]. However, RFID requires setting up infrastructure (though inexpensive) and deploying RFID tags on smartphones, which is not feasible.

Diallo *et al.* [5] outline their approach and implementation (which is easy to set up) with passive tags installed in the environment, though still not as simple as Wi-Fi.

3.1.3 Bluetooth

Bluetooth is a technology present in all smartphones today, but Bluetooth Low Energy (BLE) beacons must be installed in the environment [6]. One major drawback to Bluetooth is the device discovery procedure, which significantly increases latency (10-30s) [1]. This makes Bluetooth impractical for real-time positioning applications. However, Apple Airtags [7] have recently been introduced. They use Bluetooth signals to detect and update the location of an "Airtag" on the cloud, and use Ultrawideband technology for short-range detection and localisation.

3.1.4 Infrared Radiation

Infrared Radiation (IR) is a common wireless positioning system [1]. It requires LOS and is accurate at detecting and tracking IR tags that can be put on mobile devices. The tags emit an IR signal at regular intervals, and receivers in the room can accurately detect and locate the tag. However IR signals can be easily disrupted by sunlight and cannot penetrate walls and obstacles [3], meaning the coverage is quite narrow so many IR receivers must be placed in the environment, driving up cost greatly.

3.1.5 Ultrawideband

Ultrawideband (UWB) is a short range, high-bandwidth technology, capable of resisting signal attenuation [1]. The technology is promising, with accuracy achieving 20cm-30cm. However it is known to be very costly as the hardware is expensive, making it difficult to scale up. Infsoft [8] is a company that has developed a UWB indoor positioning system using multilateration techniques (See Section 3.2.3) but, requiring LOS to work.

3.1.6 Ultrasonic

Ultrasonic positioning uses ultrasound signals to estimate the position of an emitter tag from receivers [3]. Ultrasonic positioning systems typically use multilateration (See Section 3.2.3) and

Time of Arrival (ToA) to extract the difference from signal delays between each signal and extrapolate the position. Navigine [9] and Sonitor [10] are companies that have developed ultrasonic indoor positioning technologies. They claim a very high accuracy of 3cm and up to a couple of inches respectively, and other very desirable characteristics such as real-time efficiency and relatively low cost and maintenance.

3.1.7 Wi-Fi

Currently, Wi-Fi-based technology is the most widespread and most attractive method to achieve an indoor positioning system [3]. It is highly compatible with mobile devices and does not necessitate installing additional hardware, making it the most appropriate for a student project as the cost is near zero.

Zegeye *et al.* [11] derive two main categories of indoor positioning: infrastructure-free and infrastructure-based. Technologies like Wi-Fi, Bluetooth, geo-magnetic and sound signals are infrastructure-free, as they rely on existing signals and does not require implementing additional hardware into the environment or on mobile devices. Technologies like RFID, IR, ultrasound require placing expensive infrastructure in the environment like beacons, transmitters, receivers etc. The infrastructure-based technologies inherently demand higher cost on the environment host and the user. Such high costs are unfeasible for a growing technology like indoor positioning to be adopted for smartphone applications.

Table 3.1 summarises the advantages and disadvantages of the positioning techniques mentioned above.

Table 3.1: Comparison of common position systems used for localisation [1].

System	Accuracy	Coverage	Cost
GPS	6m-10m	Good Outdoor Poor Indoor	High
RFID	1m-2m	Indoor	Low
Bluetooth	2m-5m	Indoor	High
Infrared	1m-2m	Good Indoor	Medium
Ultrasonic	3cm-1m	Indoor	Medium
Wi-Fi	1m-5m	Building Level (outdoor/indoor)	Low

3.2 Wi-Fi-Based indoor positioning

Zegeye *et al.* [11] discusses the two general categories of Wi-Fi indoor positioning techniques which are:

- Received Signal Strength (RSS) fingerprinting in the form of building a radio map of labelled data, indicating the RSS values and the corresponding access points (APs).

- The other method involves using triangulation together with the known location of the environment's APs to determine the user's location.

In both cases, data about the environment must be collected beforehand.

3.2.1 RSS Fingerprinting

RSS represents the power and quality of the received signal and is measured in dBm [3]. An RSS value effectively represents the strength of the signal between the AP and the communicating device.

RSS fingerprinting is conducted in two phases. An offline phase and an online phase. During the offline phase, a radio map of labelled data in the indoor environment is constructed. The data consists of RSS values and the corresponding AP MAC addresses. Many papers [6, 11–13] have simplified the radio map down to a grid-like layout of the environment, with each grid-point recording the RSS values between AP addresses and the corresponding points on the map. A conceptual example of which as shown in Figure 3.1 [13].

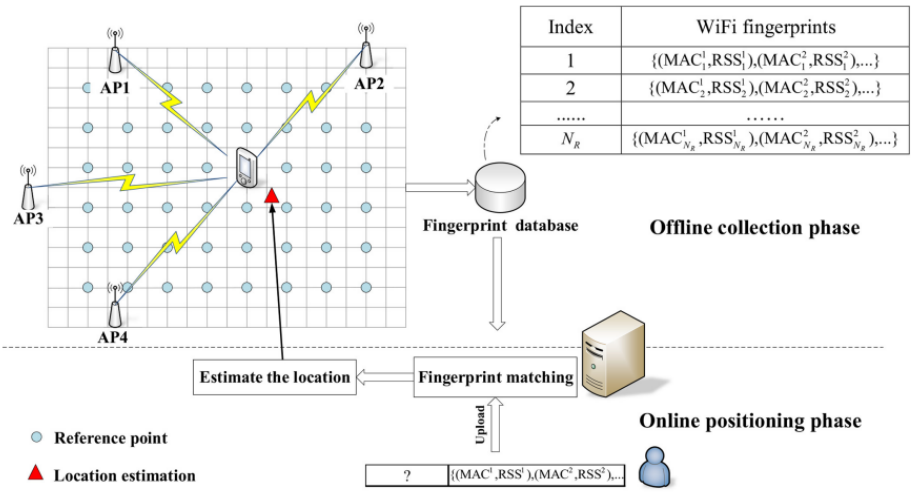


Figure 3.1: Wi-Fi-based indoor fingerprint positioning system [13].

Theoretically and in practice, this mitigates the problem of signal attenuation from floors, walls, doors etc. An RSS reading in the environment from a user should also yield similar values on the radio map.

The user attempting to locate their position is called the online phase, where it considers the radio map an optimisation problem and uses a reverse function to try and estimate the user's position [11]. The reverse function is called a positioning/localising algorithm however there are problems to consider.

Noise is a major factor. Surrounding people walking around can block Wi-Fi signals momentarily leading to inconsistent data, and even the body of the data gatherer and/or end user may block certain AP signals which can lead to faulty readings. There are many different methods to combat these issues such as using different positioning algorithms like K-Nearest Neighbour (KNN, used by RADAR [14]) or a probabilistic method or fusion technology, or a complex Artificial Neural Network (ANN), all have advantages and disadvantages in reducing noise depending on the algorithm characteristics, application and implementation. Other methods such as using compass heading is discussed in Section 3.3.3.

Noise can also be mitigated by increasing the density of the grid points [12]. However, this is only true up until a certain point.

3.2.2 Positioning Algorithms.

After the initial offline phase of fingerprinting is conducted, the online phase or localisation phase will happen, which involves actually estimating user location. This is where the user will go onto the site location, stand somewhere, and record RSS values. These values can then be compared with the radio map to determine location. Basri and El Khadimi [3] describe the main two different approaches possible:

- **Deterministic Approaches**

  Deterministic Approaches are generally the most simplistic, are easy to implement, and are very useful for real-time applications.

  The best example of a deterministic approach is K-Nearest Neighbour (KNN) and is one of the most basic classification algorithms [3]. The idea is to determine the user position based on the measurement similarity between k-nearest neighbours on the map. It was first use by RADAR. RADAR is an early positioning system developed by Microsoft Research that utilises indoor positioning Wi-Fi technology [14]. RADAR is based on empirical signal strength measurements combined with a theoretical propagation model. The empirical signal strength measurements taken are akin to RSS fingerprinting with KNN as the positioning algorithm. These keywords are not used, perhaps because the terms had not been coined yet.

  The RADAR was published in 2000 and the system achieved a median resolution accuracy of 2 to 3 metres. The size of a typical office room.

  SMARTPOS [15] was another early indoor positioning that used Wi-Fi RSS fingerprinting with a deterministic Weighted KNN (WKNN) positioning algorithm, and a digital compass to filter out RSS values based on user orientation. This method was inspired by an earlier system COMPASS [15] that used a probabilistic positioning algorithm instead (See Section 3.3.3 for how the COMPASS method worked).

  COMPASS was published in 2006 and achieved an average error distance of less than 1.65m.

  SMARTPOS was published in 2011 and achieved a mean positioning error of 1.16m with a variance of 0.66m.

- **Probabilistic Approaches**

  Probabilistic algorithms come with high computational complexity but provide better accuracy, and are robust to noise [3]. Liu *et al.* [16] describe how the model first calculates an RSS probability distributive function (PDF) for each AP received at a grid point from the fingerprinting map, then combines the distributive functions to create a joint distributive function for the grid point. This contrasts greatly with the deterministic approach where the raw RSS data is stored for each grid point. Then during the localisation phase when the user receives RSS values, the algorithm will search for the most probable location(s).

Other more involved methods include Support Vector Regression (SVR), Artifical Neural Networks (ANN), Extreme Learning Machine (ELM), Deep Neural Networks (DNN) to name a few [6]. These machine learning methods have shown promise but increases the complexity of the system greatly. The indoor positioning research space is only growing, and it is expected that even more development will be invested in the topic.

3.2.3 Lateration and Angulation

Lateration/Trilateration/Multilateration all refer to distance-based measurements from the indivdual APs and the device location [1]. Derivations of this are also possible; for example, ToA (Time of Arrival) where all devices are synchronised beforehand and calculate the delay between signal transmission and reception of the 3 (or multiple) APs and the user's device. See Figure 3.2 [1]. This method, much like measuring RSS, calculates distance, though is very susceptible to signal attenuation and is very sensitive as radio signals travel at the speed of light, so calculating these time delays requires very sensitive hardware.

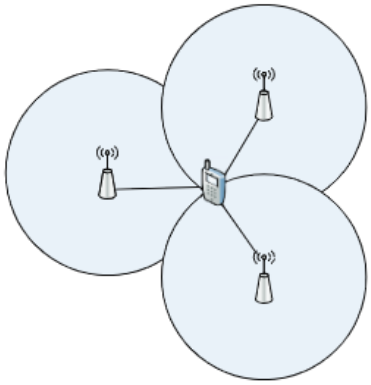


Figure 3.2: Positioning based on ToA measurements [1].

For example, Chan and Sohn [17] use trilateration to compute distance with a path loss function on RSS values and AP coordinates and produced a local (x, y, z) coordinate.

AoA (Angle of Arrival)/Angulation/Angle-based approaches are similar (See Figure 3.3), whereby the angle from the APs is measured and the intersection can be calculated to find the position. As Farid *et al.* [1] has outlined however, AoA has limitations in the fact that typical hardware cannot measure angular information, increasing cost drastically. An AoA system is also affected by signal attenuation from indoor objects and is therefore ineffective for indoor applications.

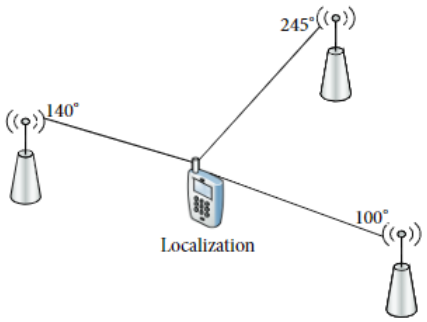


Figure 3.3: Angle-of-arrival positioning method [1].

Evaluation of WLAN-Based indoor positioning. Lateration and Angulation is given as a brief explanation but it is already ruled out as an applicable approach for a student project. It is much more feasible and accurate to use RSS Fingerprinting and is the current state-of-the-art.

3.3 Smartphone Sensors

The first part of this project discussed the challenges with indoor positioning, Wi-Fi-based and other indoor positioning techniques. The following sections will focus on smartphone sensor data and combining the two together.

As previously mentioned, a large advantage of smartphones are the sensors housed within them. These sensors can be exploited to gather more data and further increase accuracy of tracking a subject through a building. Key sensors in smartphones are the compass to give heading information and inertial sensors, where we can extrapolate steps taken by the user to measure odometry. Many papers [12, 18–21] use accelerometers, gyroscopes and/or compasses and is a widely adopted approach [1].

3.3.1 Challenges with using smartphone sensors

- A smartphone can be held in the hand or the pocket and can face many different ways. Using raw compass heading with the phone facing downwards is impossible, but solvable [19].
- Smartphone sensors are generally small and low quality and are subject to noise and false readings.
- Usage of the smartphone is not always consistent. The position and orientation of the phone changes rapidly. i.e. a phone is regularly held in the hand, put in the pocket, spun around, all in the space of a few seconds. This is detrimental to data collected if not accounted for.
- Smartphones are used by a wide range of different people with different physical characteristics. i.e. some people walk faster than others, are taller than others, walk with heavier steps than others etc.

A system that cannot adapt to these irregularities cannot reliably use smartphone sensor data.

3.3.2 Inertial Sensors

The most popular candidate for fusion technology are inertial sensors [22]. Many papers [23–25] use IMUs (inertial measurement units) to further improve localisation accuracy. These IMUs can commonly be found in smartphones. The motivation behind using inertial sensors is to detect steps and step length to measure distance and speed. This approach is referred to as a pedestrian navigation system (PNS) [19, 25] and it is an instance of the dead reckoning approach.

Pedestrian Dead Reckoning (PDR) is a very popular approach with inertial sensor localisation technology [22]. The idea is that accelerometer data can detect foot impacts with the ground as the user is walking. Using accelerometer data to detect steps, and with an estimation of step length along with heading and a known start position, velocity and distance can be calculated. The heading can be computed with gyroscope and magnetometer outputs. This is not a wireless positioning technique as mentioned before as it requires only internal information but it can be very helpful when combined with a real world anchor such as RSS fingerprinting.

The challenge with PDR in smartphones are the low sensitivity and high drift IMUs, as well as the challenges mentioned above (Section 3.3.1). Though there are some methods to combat these problems.

Samuel *et al.* [26] developed an indoor positioning system using an RFID-based positioning and an inertial tracking device attached to the foot. This is to take advantage of the Zero Velocity Update algorithm (ZUPT). ZUPT is a drift-cancelling method that relies on the IMU to be attached to the foot. When the foot hits the floor, the velocity data can be reset to zero, thereby reducing sensor drift. The foot impact is used as a reference, as the foot is stationary which allows most of the drift to be mitigated. Although, this method cannot be used for smartphones. Error rates proportional to the distance travelled still exist which is why a positioning technique such as RFID was used. Samuel *et al.* [26] achieved an average path error of 0.47m and a maximum error of 1.77m.

Zhang *et al.* [23] implemented an indoor positioning system using only PDR with inertial sensors in a smartphone. Their step detection model takes in vertical acceleration data, filters it, and decides steps based on positive-going, zero crossing of the acceleration. i.e. when the signal goes from negative to positive, the system then registers this as a step. However even small vibrations can misdetect a step and give a false positive. To combat this, the two thresholds method is used whereby if the absolute value of the acceleration signal is within the two thresholds, no step is detected. Figure 3.4 shows this clearly.

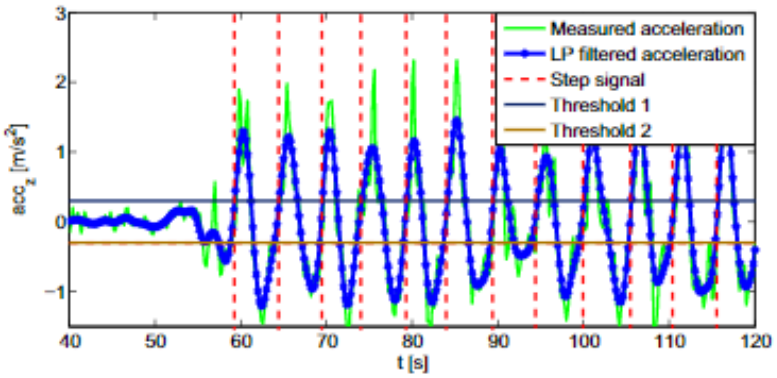


Figure 3.4: Step detection using two thresholds [23].

Figure 3.5: Estimated path vs. ground truth without (top) or with (bottom) personalisation[19].

Zhang *et al.* [23] also used a step length determination model using well described algorithms in literature. It uses several equations that simulate a biomechanic model of a kneeless biped. This mathematical model along with the step detection system was incorporated into their Kalman filter (See Section 3.4). As it was an experimental result, no accuracy error value was stated.

Fan *et al.* [19] also developed an indoor positioning using only smartphone sensors. They also use a two thresholds method to detect steps. However their step length estimation model is adaptive and attempts to personalise to the user. This is to address the challenges mentioned above (Section 3.3.1).

Figure 3.5 shows the results of tracking with and without personalisation. Every lap and the location was reset by the user, if the system failed to track correctly to eliminate drift in the collected data. The personalisation model greatly reduced tracking failure from 17 failures to 3 failures.

Fan *et al.* [19] also developed a model to infer heading direction from a phone orientation model. This orientation model was also applied to step detection and step length so the position and orientation of the phone was fully accounted for, and results from a wide variety of extensive experiments (50 subjects walking an aggregate distance of 40km) conducted showed this well.

A mean accuracy of 1.5m was achieved for the in-hand case and 2m for the in-pocket case. This is a very impressive result for an indoor positioning system that doesn't rely on external infrastructure.

3.3.3 Compass Heading

King *et al.* [27] developed a system called COMPASS. One of the key ideas was that during the offline phase, the orientation of the person taking the RSS readings was recorded as well, with perhaps multiple readings with different orientations for each reference point. This information can be useful during the online/localisation phase; The user's heading would be taken into account to use only RSS readings that were recorded with a similar heading orientation (See Figure 3.6).

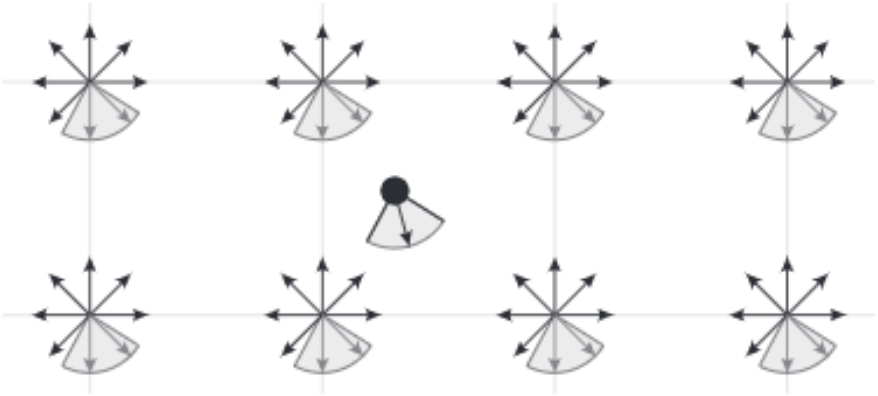
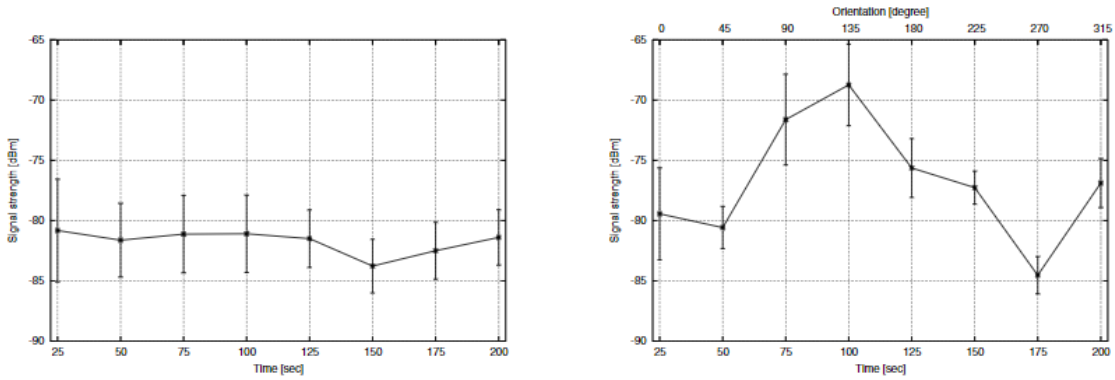


Figure 3.6: Each intersection of the grid lines are a reference point. They each have 8 orientations with RSS readings for each. The black dot represents the user and their orientation. The gray shaded area is the accepted orientation-specific distribution. The localisation algorithm will only take RSS values from reference points with that specific orientation as shown with the same shaded area applied to each reference point [27].

This process was used to mitigate signal attenuation caused by the human body which can be seen in Figure 3.7.



(a) User is motionless.

(b) User is turning around.

Figure 3.7: RSS readings over time based where (a) is the change in RSS when the user is motionless and (b) is the change in RSS when the user is turning on the spot [27].

The main disadvantage to this method is the major increase of labour of the site survey. For example, with a site survey of 20 grid points and 8 orientation-specific readings for each, there will be 160 RSS readings. This is discounting probable extra measurements such as averaging RSS readings over time which will also greatly increase labour cost (though Werner and Kessel [15] mention the state-of-the-art approach is 4 heading directions for each reference point which is still significant).

Liu *et al.* [28] combines compass data, accelerometer data and map constraint data through the use of a particle filter (See Section 3.4.1 for much greater explanation). Compass data was recorded at specific times and correlated with step detection data, to measure turning angle of each step. Each generated particle simulates a possible location for the user. It is applied new data and is given weights based on likelihood of the event. For example, if a particle is given data that makes it take a turn and step through a wall, it is weighted down. The estimated location is then the average of the survived particles after the particles have converged.

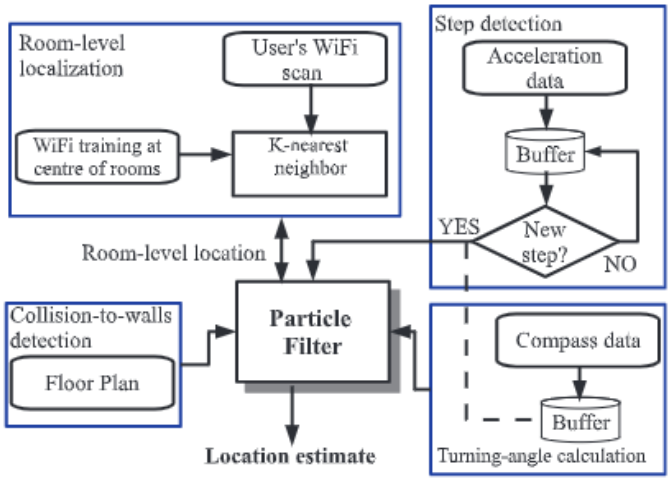


Figure 3.8: Diagram of the system [28].

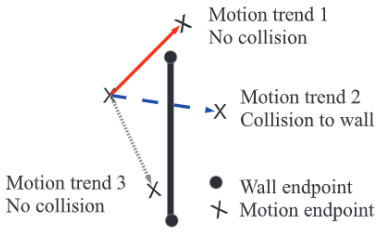


Figure 3.9: A collision is detected in scenario 2 [28].

Figure 3.4.1 shows conceptual diagram of the system developed by Liu *et al.* [28] and Figure 3.9 shows the map constraint data that was applied to the particle filter.

3.4 Fusion Technology

Fusion Technology seeks to combine two or more different types of data to improve accuracy from initial localisation estimates [16]. It normally entails using sensors in smartphones but it can incorporate the meaning of using external sensors like cameras as well. There many fusion schemes available, but ones relating to Wi-Fi indoor positioning using smartphone sensor are generally either particle filters (as mentioned above) or a variation of Kalman filters (KF) [22].

The main disadvantage to using particle filters is the exponential complexity. Though as Potorti *et al.* [22] discuss, Kalman filters are getting more and more unpopular, as the advantages and disadvantages of particle filters are starting to outweigh Kalman filters, especially as computational capability keeps increasing year on year.

MATLAB [29] and Gustafasson [30] explains in great detail how a Kalman filter (and its many variations) may not be optimal for a localisation optimisation problem such as indoor positioning. Reasoning being, Kalman filters tend to fail to follow map constraints, and the inherent noise in the new tracking information (RSS values, heading and odometry information) and the estimated output state may not all follow a gaussian probability distribution, which a Kalman filter heavily relies on [31].

3.4.1 Particle Filters

Particle filters or Sequential Monte Carlo (SMC) methods are a set of simulation-based methods[31] that can calculate posterior distributions of partially observable, controllable Markov chains. Thrun [32] explains that a partially observable, controllable Markov chain is a set of state distributions where each state x_t at time t depends on the previous state x_{t-1} and the control/action u_t (i.e. movement made by the user) according to the probabilistic law $p(x_t|u_t, x_{t-1})$.

Since the true state x_t of the Markov chain is partially observable, we can validate our true state with z_t, the observation data. Though since our observation data is still sensor information, it carries a degree of noise and uncertainty. z_t is measured for likelihood via $p(z_t|x_t)$ for each particle.

This problem of recovering the posterior distributions is solved by Bayes filters, which computes the posterior distribution recursively [32]:

$$p(x_t|z^t, u^t) = const. \times p(z_t|x_t) \int p(x_t|u_t, x_{t-1}) p(x_{t-1}|z^{t-1}, u^{t-1}) dx_{t-1}$$

under the initial condition $p(x_0|z^0, u^0) =$ some distribution $p(x_0)$. The Bayes Filter is the underlying driver behind particle filters and resembles Kalman Filters closely [32]. That is to say, that the math behind both particle filters and Kalman filters are identical if certain parameters are met. However particle filters do not have actual probability distributions, each particle has different weights that simulate those distributions [30], meaning it is not a Gaussian distribution that a Kalman Filter assumes, and can therefore take on more general cases of Markov chains [32].

Particle Filter Algorithm Drawing inspiration from Thrun [32], Gustaffson [30], Udactiy [33] and Shu *et al.* [21], a simple particle filter algorithm is outlined:

Variable details:

- Let $N =$ the number of particles, $i \in [1, N]$ is the particle index.
- The set of all particles at time t: $\{x_t^i\}$ which approximates the posterior set of sample states, where each x_t^i is a state sample of the true state x_t

  $\{x_{t-1}^i\}$ is the set of sample states directly previous of the posterior. The basic idea is to recursively calculate $\{x_t^i\}$ from $\{x_{t-1}^i\}$, which calculates itself from $\{x_{t-2}^i\}$ and so on and so forth.
- w_t^i is the weight associated with particle x_t^i. w_t is the total simulated weight distribution.
- u_t is the control or movement data at time t
- z_t is the sensor measurement/observation data at time t

- **Intialisation:** When $t = 0$, generate N particles with weight $w_i = \frac{1}{N}$. Call this set of particles X_0
- **Recursion:** When $t > 0$:
  For each particle x_{t-1}^i in X_{t-1}:

  1. Sample an index $j(i)$ by drawing from the simulated distribution w_{t-1}.
  2. Generate a particle x_t^i by drawing from $p(x_t|u_t, x_{t-1}^{j(i)})$.
  3. Reweight w_t^i by $w_t^i = p(z_t|x_t^i)$ i.e. Computing the likelihood of the sample particle x_t^i being correct given z_t

  Normalise w_t
  Call this resulting set X_t

As t increases, the set of particles X_t converges uniformly to the desired posterior $p(x_t|z_t, u_t)$.

This is the basic idea of the algorithm. It is a way of statistically simulating an environment where the probability of a current is affected by past events, and applying gathered data compute these probabilities. However in realistic situations, u_t and z_t have noise added to them in form of some independent Gaussian as movement and observation data naturally has inherent noise [32].

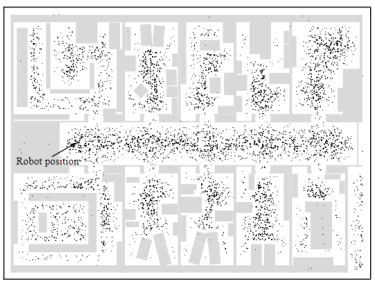


Figure 3.10: Global Uncertainty [32].

Figure 3.11: Partial Certainty [32].

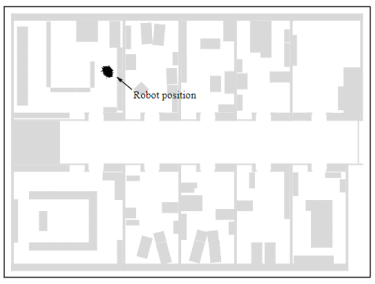


Figure 3.12: Convergence of particles [32].

Figure 3.10, Figure 3.11 and Figure 3.12 visually showing the particle states over time in this robot positioning model. Notice there are 2 main clusters in figure 3.11 where the robot has given

observation data of turning left into a door, however as it enters a unique room in figure 3.12, all particle have correctly clustered around the main point.

Other processes like dead-reckoning can be applied to predict future events [30]. Many more steps may be conducted to improve performance and/or accuracy such as prediction, smoothing and marginals.

Thrun [32] discusses the main disadvantage of particle filters and that is the exponentially higher complexity when posed with a higher dimensional space problem or the "kidnapped robot problem" where the object of interest is in an unknown environment and must map its surroundings and its position in it. Though variations of the particle filter have been developed to solve these issues, it is not relevant to this project.

Table 3.2: Comparison of fusion technology approaches

System	Accuracy	Reference
Wi-Fi+PDR+KF:	4.83m	[16]
Wi-Fi+PDR+PF	4m	[16]
Wi-Fi+Rao Blackwellized PF	1.2m	[34]
Wi-Fi+Map data+Inertial Sensors	1.45m	[12]
Wi-Fi+Map data+Inertial Sensors+PF	1.33m	[28]

Table 3.2 shows some results achieved by different research papers though each one has unique differences that is impossible to express in a simple and short comparison.

Chapter 4: Project Management Plan

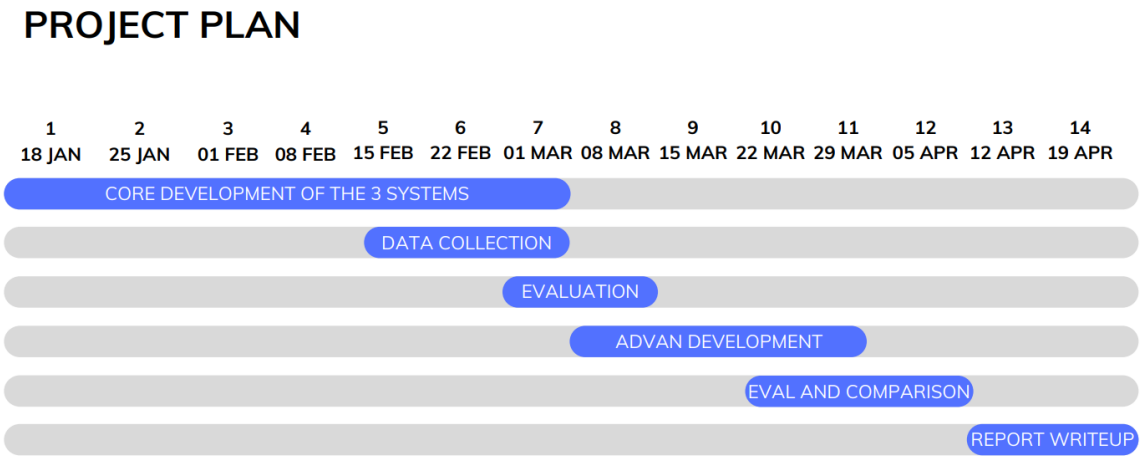


Figure 4.1: Project Workplan timeline

Most of the next semester will be dedicated towards the implementation of the core objectives. A large amount of time will be dedicated towards evaluation, as there will be many variations of each of the systems, particularly the particle filter, and all of these variations will need to be compared for performance.

Core Development will include testing and RSS data collection of the Science Centre. This will include the aforementioned Wi-Fi RSS Fingerprinting, PDR Smartphone sensors and particle filter integration.

Evaluation of the core program will include graphing results, comparison and so on.

Time will be left over to be used for the development of advanced objectives with evaluation, and comparison of all of the different variations that have been implemented.

Finally, the last two weeks will be reserved to write up the final report.

Chapter 5: Implementation

The objective of the implementation is to achieve indoor positioning with a smartphone. To do this, the following objectives must be achieved:

- Develop a Wi-Fi RSS Fingerprinting system (Discussed in Section 3.2.1).
- Develop a PDR system using a smartphone with its embedded inertial sensors and compass sensor (Discussed in Section 3.3).
- Develop a Particle Filter that will use the previous two systems as input data (discussed in Section 3.4.1). The particle filter will need outputs from the PDR system as control data, and outputs from the RSS Fingerprinting system (with map constraint data) as observation data.
- Evaluate all three systems extensively.

The 3 systems will be used for:

- Gathering data for the Wi-Fi RSS Fingerprinting system and the PDR system for both the offline and online phases. This will be done in an area in the UCD Science Building. The offline phase data will be an RSS Fingperinting map (Discussed in Section 3.2.1). The online phase data will be conducted trials of walking around in the mapped area, with the path taken recorded for later reference.
- Following this, the data will be processed into observation and control data for the particle filter. This will be done after the offline and online phase has been conducted, i.e. in non-real-time.
- Finally, the particle filter will be used with the above data to predict the path taken during the online phase. These results are the goal of the project and will largely determine the success of the implementation as a whole.

5.1 Planned Implementation

5.1.1 Wi-Fi RSS Fingerprinting System

The Wi-Fi RSS Fingerprinting system I am planning to implement has several hurdles it needs to overcome:

- It must be able to gather Received Signal Strength Indicator (RSSI, interchangeably used with RSS) data and MAC address data.
- It must transform the data in different ways so they are suitable for both the offline and online phases.

- It must compute and output predicted coordinates for the online phase from the captured data of both phases.

To achieve the first step, a third-party tool is the best solution for both time and cost. It will be used to gather the data during both the offline and online phases. For the offline phase, a designated area of the UCD Science building will be mapped onto a grid and given grid coordinates. The tool will then be used to gather Wi-Fi RSSI data at each grid reference point, so all the grid points will have an X and Y coordinate, as well as one or multiple Wi-Fi scans that are complete with RSSI and MAC address data.

For the second step, the tool during the offline phase must output the raw Wi-Fi data into XML format where each branch corresponds to a grid point. Each grid point will have given X and Y coordinates as well as RSSI and MAC address data, so each branch of the XML will also have this information too. I elaborate more on this method in Section 3.2 where Zegeye *et al.* [11] implemented it with good results.

During the online phase, the tool will gather the same Wi-Fi RSSI data but without grid coordinates. Instead, each branch will have a timestamp associated with it, so we know when each scan occurred during the conducted (walking) trial.

For the final step, a positioning algorithm will be implemented using KNN or WKNN to do Wi-Fi RSS localisation (Discussed in Section 3.2.2). Essentially, the system will compare the data from the offline and online phases to try and predict where each scan, from the online phase, is in the real world, by looking for the most similar subset of scans from the set of scans taken during the offline phase (Explained further in Section 3.2). A potential dilemma may arise from using KNN or WKNN whereby any MAC addresses that are present in only one scan, can either be set to -100dbm or the whole grid reference can be completely ignored as a candidate for a (K) Nearest Neighbour. There is already a precedent to ignore the missing values [11, 15] and I shall attempt both, evaluate them, and use the better option.

The resulting predicted coordinates from the output of this system will be used as observation data (referred to in 3.4.1) for the particle filter system.

Evaluating the system is also very important. This shall be done by testing the system with static location estimates and tracking, with short time intervals between Wi-Fi scans and evaluating each result.

5.1.2 Pedestrian Dead Reckoning System

The implementation of the Smartphone sensors involves a system that gathers data from smartphone sensors and produces data about exactly when steps were taken during the online phase and heading data that is consistently updated throughout the trial. This is essentially a PDR system (Discussed in Section 3.3.2). But that means it comes with some assumptions in order to ease the initial requirements of development:

- The position of the smartphone is held in the hand face-up.
- The orientation of the smartphone is assumed to be flat and the top of the phone facing the direction of travel.

This is to gather to most accurate information as an orientation-free model is not considered for the initial implementation of the system.

Although different systems entirely, the objectives of the PDR system are similar to those of the Wi-Fi RSS Fingerprinting system.

It must gather inertial data from the accelerometer sensors and heading data from the compass sensor during the online phase. This data is inherently of a time-series format A third-party tool will also be used to capture this data as it is the simplest solution. Using the captured data, the system must transform the time-series inertial data to a time-series graph of when steps are predicted to have occurred. To do this, the inertial data is fed through a step detection model using the two thresholds method (Discussed in Section 3.3.2). The thresholds will initially be manually configured and later on, automatic personalisation may be considered.

To aid in the step detection model, a step length will have to be set. Initially, the length of each step will be hard-coded but, a personalisation model that can adapt to any user's step length may be considered (Discussed in Section 3.3.2).

The step detection data is then used as control data (referred to in 3.4.1) for the particle filter system.

Other objectives to consider would be the aforementioned orientation-free model to nullify the above assumptions to increase the versatility of the indoor positioning tool. The implementation of this model is outlined in [18].

5.1.3 Particle Filter System

The particle filter combines data from both systems outlined above, to empirically predict the real location of the user during the online phase. This is the main goal of the entire project, and it relies on the particle filter working well.

The characteristics that the particle filter must have, are that it has to take in the output of the Wi-Fi RSS Fingerprinting system as the observation data and the output of the PDR system as control data. It also has to somehow visually show what it is doing every iteration. The two requirements are achievable through research and coding work.

Some other objectives to consider would be applying map constraint data outlined in Section 3.3.2, decreasing the standard deviation of the added Gaussian noise as the algorithm progresses, and also decreasing the number of particles resampled as the iterations progress.

Evaluation at this stage would play the largest part as the output of the particle filter is essentially the output of the entire project. Things like changing input data, and tweaking variables will be conducted to test the system extensively and attempt to improve its accuracy.

All 3 systems will most likely be implemented in Python using Jupyter Notebook as the tools for handling data are easy to use and plotting graphs to visually show results will be easier than other environments. My own mild familiarity with Jupyter Notebook will also serve to save time in learning the new techniques required to develop the 3 systems from the ground up.

5.2 Achieved Implementation

The work conducted has followed the planned implementation relatively closely. But rather than the 3 separate systems outlined in the planned implementation 5.1, instead, there were 4 stages of

development: data gathering of both the Wi-Fi RSS Fingerprinting and PDR data, data processing of Wi-Fi RSS data, data processing of Wi-Fi inertial and heading data, and for the outputs of the data processing to be used by the particle filter as observation data and control data respectively. This change occurred mostly because of the fact that I had to develop tools to gather data instead of using a third-party tool (Discussed later) and that data processing would be conducted after all the data was collected, i.e. in non-real-time.

5.2.1 Data Gathering

I conducted the data gathering work in mainly two areas shown in Figure 5.1 and Figure 5.2. The data was gathered with a grid point density of 2m in both areas. These locations were chosen because of the low traffic of passersby but also the fact that the floor tiles were aligned parallel with the walls, making measurement of each grid point much easier.

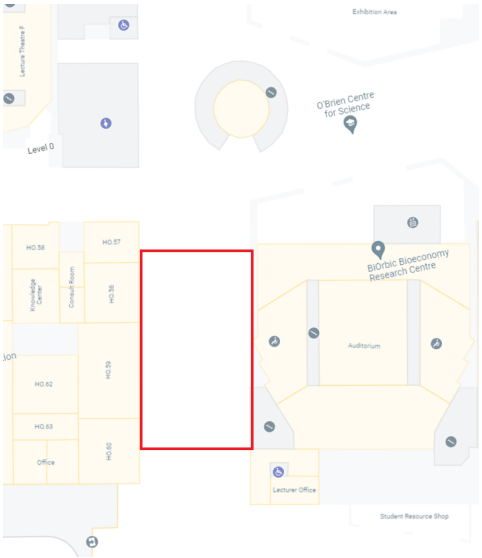


Figure 5.1: The area highlighted in the red box beside the Moore Auditorium
Area: 14 x 8m

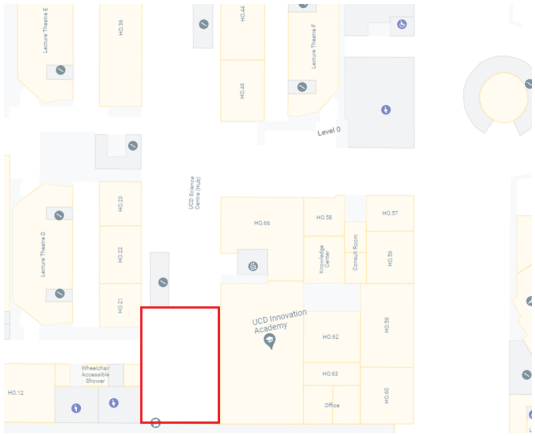


Figure 5.2: The red box beside the door leading to UCD Science Centre South
Area: 12m x 6m

However, one major setback occurred at the start of the implementation. I found it extremely difficult to find a tool to gather Wi-Fi RSS data. Any tools I did find were not available for free. So I was forced to develop my own tools to capture and gather the necessary Wi-Fi RSS data. As I was already going to make tools for the Wi-Fi data, I also incorporated gathering inertial and heading data into the tools as well. These tools were developed for use by my Android phone so Android-specific apps would be needed. In the end, two apps were developed. One for gathering Wi-Fi RSS data during the offline phase (App A), and one that gathers inertial data, heading data, and Wi-Fi RSS data as well during the online phase (App B). These apps were inherently restricted by the Android API, and I found they had some severe limitations associated with them discussed in Section 5.3. A screenshot of the two apps developed is shown in Figure 5.3 and Figure 5.4.

App A takes a grid coordinate as input to be associated with that scan or series of scans. START SCAN will attempt Wi-Fi scans every second, though as per the API restrictions, this doesn't actually result in one scan per second. STOP SCAN will stop the thread that is scanning and store the acquired scans as well as the grid coordinate input in memory as an object, then scanning for the next grid point will be available again. The WRITE button will take the list of stored objects and write them to internal storage as an XML file. An example of which shown in Figure 5.5.

The XML file will contain grid data that consists of a Wi-Fi scan that has a physical address, and corresponding RSSI and Service Set Identifier (SSID) values (SSID is the public display name of a Wi-Fi network). Multiple scans for the same grid point will not be stored in the same branch, instead, each scan will be a separate object and therefore a separate branch. This means grid coordinates in branches may not be unique.

App B constantly reads inertial data from the Android API and presents it visually as a graph at the top of the screen. It also is constantly receiving "local heading" data relative to the orientation of when the app was started up. The app only has a "local heading" because the magnetometer was too unreliable and did not give accurate readings even while outside. This limitation will also be discussed more in Section 5.3. The heading is shown in degrees and operates around the vertical axis. It also rotates the compass image for visual reference. The CALIBRATE button reassigns a new sensor object to the variable, so it just resets where the "local heading" is relative to without having to restart the app. The START button starts the recording of the current acceleration reading, the current heading, the current timestamp, and the current Wi-Fi data if it is available (if not, it is set to null), into memory as objects. When the END button is pressed, the list of objects is then written to file in XML format in a similar process to App A. An example of the output is shown in Figure 5.6. Any objects where Wi-Fi data was set to null is set to an empty <wifi></wifi> header in the output.

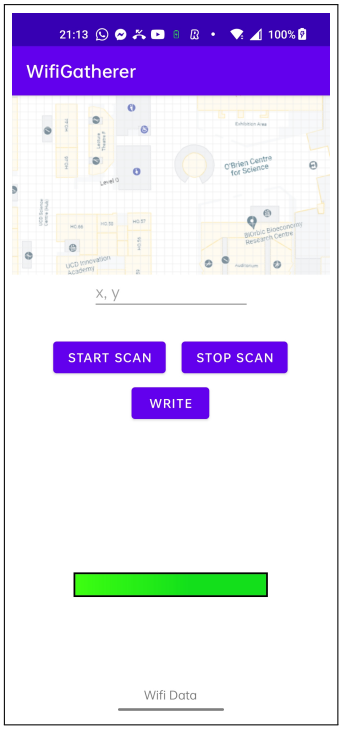


Figure 5.3: Screenshot of the tool used to gather Wi-Fi RSS Fingerprinting data during the offline phase

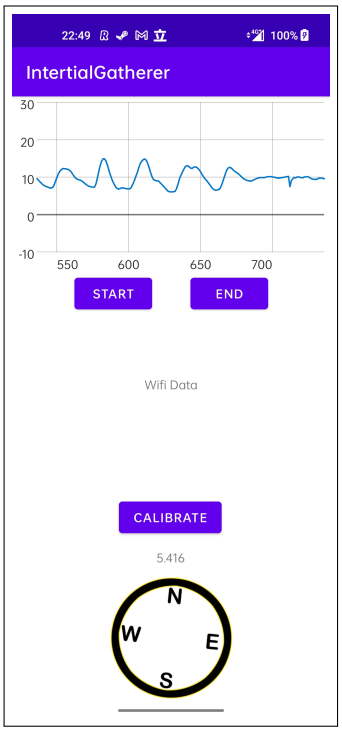


Figure 5.4: Screenshot of the tool used to gather inertial data (shown in the graph), Wi-Fi RSS data, and heading data during the online phase. The compass image is for visual representation and was not used to indicate real-world orientation

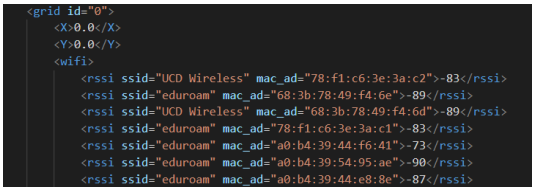


Figure 5.5: Snippet of one block of data output from the Offline tool

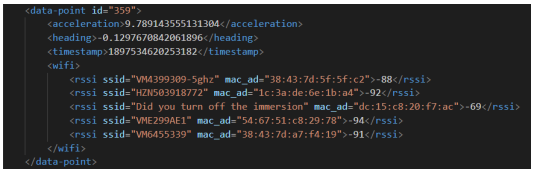


Figure 5.6: Block of data from the Online tool

5.2.2 Wi-Fi RSS Fingerprinting data processing

The Wi-Fi RSS data from the offline phase is used to compute predicted coordinates for the user during the online phase.

This is done using KNN. Each scan during the online phase is compared to each grid point scan recorded in the offline phase. For any matching MAC addresses, the difference in their RSSI values (the error) are taken and cumulatively added up with the other error values in the same scan. The cumulative error between the online scan and all the other grid point scans are stored. The K smallest cumulative errors are taken and the average of their coordinates is designated the predicted coordinate for that (online) scan.

Some simplified Python code for the above algorithm is outlined below in Listing 5.1. Reading some of the code from the repository (up to cell 12) will help to understand the format of the inputs that this algorithm takes.

This algorithm also worked for the two interpretations of the raw data. One was where all the multiple branches for the same coordinate were all turned into one grid coordinate with all the matching RSSI values averaged together (denoted by averagedGridPointsList in the code). The other was where it was left untouched and there are multiple grid points with the same grid coordinate (denoted by gridPointsList in the code).

There are also a lot of variables to tweak in this stage like K, an RSSI cutoff point (to not accept consider RSSI values below a certain value), and the accepted number of matching MAC addresses for averagedGridPointsList (Discussed later in Section 6.2.

```python
wifi_data_df #Dataframe of all the Wi-Fi scans that took place during the online phase
     with the Elapsed Time as well.
grid_points_list #List of all the offline grid points.

def predict_coords_with_KNN(K):
    #Output DataFrame
    predicted_coords_df = pd.DataFrame()

    for i, dataPoint in wifi_data_df.iterrows():
        errors_list = [] #Cumulative error of the current Wi-Fi scan.

        for grid_point in grid_points_list:
            #Converting Wi-Fi data from both phases to key-value pair format
            #MAC address is the key and RSSI is the value
            grid_dict=grid_point.wifiData.set_index("mac_ad").to_dict()["rssi"]
            online_dict=datapoint.wifiData.set_index("mac_ad").to_dict()["rssi"]

            error = 0
            for mac_ad in online_dict.keys():
                if mac_ad in grid_dict:
                    error += abs(
                        int(grid_dict[mac_ad]) - int(online_dict[mac_ad]))
                else:
                    error += 100
            errors.append({"error": error,
                           "X": grid_point.x, "Y": grid_point.y})

        #Sort errors by the ascending error and acquire the least K elements
        sorted_errors_list = sorted(errors, key=lambda k: k["error"])
        K_sorted_errors_list = sorted_errors_list[0:K]

        #Calculate the average distance of the K nearest grid points
        x = 0
        y = 0
        for grid_point in K_sorted_errors_list:
            x += float(grid_point["X"])
            y += float(grid_point["Y"])
        x /= K
        y /= K

        predicted_coords_df["Predicted X"] = x
        predicted_coords_df["Predicted Y"] = y
        predicted_coords_df["Time Elapsed"] = datapoint["Time Elapsed"]
    return predicted_coords_df
```

Listing 5.1: KNN online phase prediction in simplified Python

5.2.3 Inertial and Heading data processing

Step detection is calculated from the inertial data gathered during the online phase. Figure 5.7 shows the inertial data plotted on a graph. This data comes from an XML file similar to Figure 5.6. The only transformations made are that the Acceleration is applied a subtraction of 9.8 to account for gravity and the timestamp is converted to Time Elapsed (since the start of the recording of the trial).

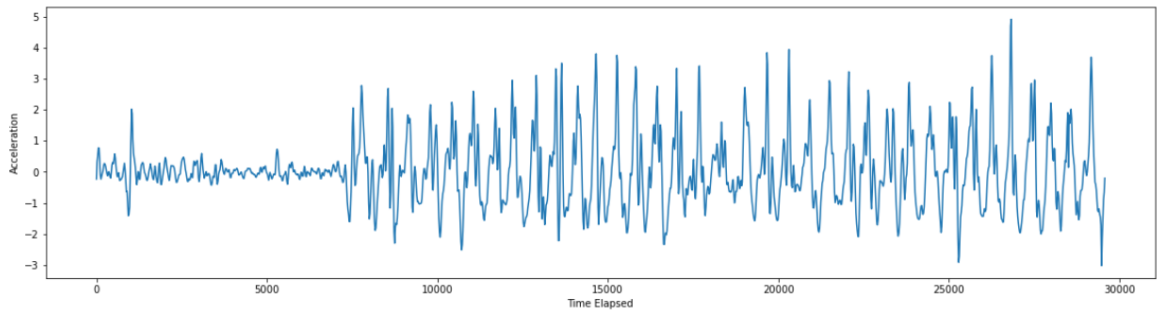


Figure 5.7: Inertial data plot of Acceleration against Time Elapsed (in milliseconds).

To smooth out the curve and remove noise, a rolling mean is applied. Then 2 thresholds are applied to the y-axis (acceleration axis). The purpose of the thresholds is to further reduce noise. This method is outlined in Section 3.3.2. Any crests or troughs (minimum and maximum of each curve/wave) that are within the two thresholds are discounted and not applicable for step detection. A $y = 0$ green dashed line is also added for visual clarity. These processes were applied to the previous graph and are shown in Figure 5.8.

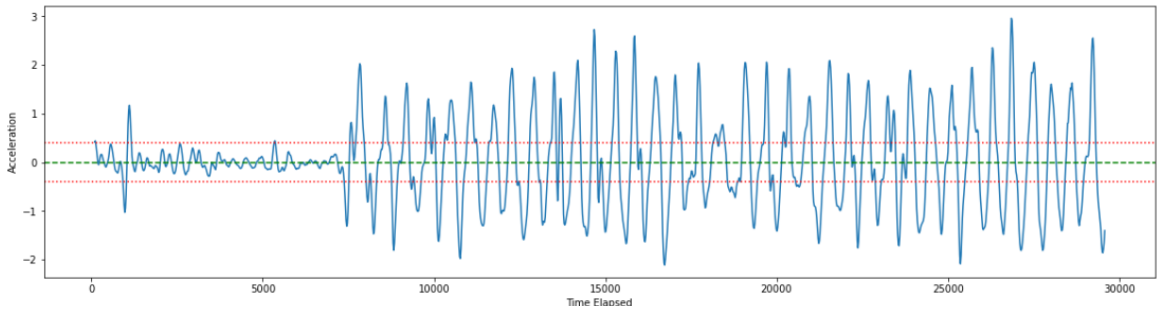


Figure 5.8: Acceleration vs Time Elapsed data that is smoothed out and given the 3 dashed lines for visual clarity.

To use the thresholds, we calculate the positions of all the crests and troughs indicated by the green and red dots respectively in Figure 5.9. Any line that originates from a trough (red dot) that passes the green dashed $y = 0$ line counts as a step, provided that neither the trough nor the crest involved is within the 2 thresholds. Any steps detected are denoted by the vertical purple dashed line and the result of this process is a list of x-axis values (Time elapsed values) that are the predicted occurrences of when steps happened during the online phase. The particle filter will use the list of values for Time Elapsed together with the heading data at that point in time, to use as control data to move the particles.

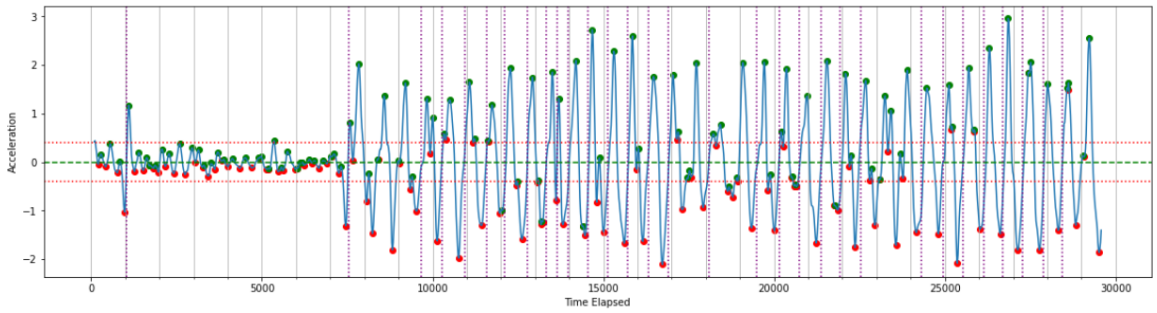


Figure 5.9: Detected Steps are being calculated.
Red dots are troughs. Green dots are crests. Purple dashed vertical lines are when steps were detected.

This method is just for detecting steps but does nothing toward acquiring the length of each step. I took the length of each step to be ∼0.6m as I measured my steps out to be approximately that length.

5.2.4 Particle Filter

The particle filter takes in all the processed data above and attempts to estimate the real location of the user. This is the main goal of the whole project. The data is a form of a time-series graph with detected steps and predicted coordinates at specific points in time that are hopefully in agreement with events in the real world. An example of some input data is in Figure 5.10. The observation data are contained in the first two columns of the data while the control data is in the latter two columns.

	Predicted X	Predicted Y	Step Detected	Heading
0	4.000000	-2.500000	False	0
1	NaN	NaN	True	0
2	NaN	NaN	True	0
3	2.507754	-2.799904	True	-30
4	NaN	NaN	True	-30
5	NaN	NaN	True	-30
6	1.404442	-3.436612	False	-30
7	NaN	NaN	True	-45
8	NaN	NaN	False	-50
9	0.788942	-4.284737	True	-60

Figure 5.10: Snippet of particle filter example input data.

The algorithm of the particle filter is outlined in Section 3.4.1 and the simplified python code of the implementation is in Listing 5.2. In summary, the particle filter algorithm samples a set of N particles uniformly at random from the input space. It then iteratively calculates the distance of each particle from the observation data

There are three main methods to the kernel of the particle filter: `apply_movement()`, `calculate_weights()` and `resample()`.

- `apply_movement()` takes as input an array of particles, a distance (or step length), a heading, and a standard deviation. The array of particles is an array of size $N \times 3$ where N is

the number of particles. Each of the N rows contains three values (x, y, h), where x is the an x-coordinate, y is the y-coordinate and h is the heading of each particle. `apply_movement()` returns a similar array of particles but with the coordinates of each particle updated to represent that it has moved the given distance in the given direction/heading. The standard deviation is used to add some Gaussian noise to both the distance and the heading to account for sensor error. This method is essentially the PDR system (Discussed in Section 5.1.2) in action.

- `calculate_weights()` takes in an array of particles, an x- and y-coordinate for the predicted location (observation data), and a standard deviation. It calculates a distance d_i between the i^{th} particle and the observation data. This distance d_i and is fed into a "likelihood" function that calculates the likelihood/probability that the particle is in the observation data. A Gaussian kernel for this likelihood. In particular, for the i^{th} particle, a weight w_i is computed as

  $$w_i = e^{-\frac{d_i^2}{2\sigma^2}}$$

  Note that as $d_i \to 0$, $w_i \to 1$, such that the likelihood increases as the distance between particle and observed data diminishes. Also as $d_i \to \infty$, the likelihood $w_i \to 0$, so that the likelihood weight is monotonically decreasing with distance from the observation.

  The weights are normalised by dividing by $\sum_{i=1}^{N} w_i$. Now the weights can be used to sample particles such that particles closer to the observation are more likely to be chosen.

- `resample()` takes as input the array of particles, the set of weights for each particle, N, the number of particles to resample, and a standard deviation. In `resample()`, particle indices are chosen at random with replacement, such that probability of choosing particle i is proportional to w_i, in order to form a new population of N particles for the next time step. Particles closer to the observation are more likely to be chosen. Gaussian noise with the given standard deviation is added to each new particle's coordinates, so that the new population of particles is likely to be clustered around those particles in the previous time-step that were closest to the observation.

The three methods are used in every iteration of the algorithm where appropriate. If there is control data, that is, if a step is detected, at a particular time-step, then `apply_movement()` is applied to all particles. If there is observation data, that is, if predicted coordinates are available, then `calculate_weights()` and `resample()` are called to generate a new set of particles, that are likely to be clustered around the observed data.

Conceptually, each particle is a "guess" of the user's position. With each iteration, the "guess" of the user's position becomes more refined as particles mimic the user's movements and more and more particles end up where the observation data predicts. Therefore more particles end up surviving with larger weights in the same place, generating more particles in the predicted area, making the "guess" more refined and so on and so forth. Eventually, the particles converge enough that the centroid of the cluster of particles can be used to predict the location of the user.

Examples of the particle filter operating on test data are shown in Figure 5.11 and Figure 5.12.

```python
import numpy as np
from numpy import random
from numpy import uniform

def apply_movement(particles, distance, heading, distance_std, heading_std):
    new_particles = np.empty(len(particles), 3)
    for i in range(len(particles)):
        particle = particles[i]
        #Adding noise
        noisy_distance = random.normal(distance, distance_std)
        noisy_angle = random.normal(0, heading_std)

        #New heading direction for particle
        direction = particle[2] + heading + noisy_angle
        x = particle[0]+(noisy_distance*math.cos((direction*math.pi)/180))
        y = particle[1]+(noisy_distance*math.sin((direction*math.pi)/180))
        new_particles[i] = [x,y,direction]
    return new_particles

def calculate_weights(particles, predicted_location, std):
    weights = []
    sum_of_probabilities = 0
    for particle in particles:
        d = distance_between(particle, predicted_location)
        p = likelihood(d, std) #Probability that d is the correct distance
        weights.append(p)
        sum_of_probabilities += p
    #Make all the weights sum up to 1
    weights[:] = [w / sum_of_probabilities for w in weights]
    return weights

def resample(particles, weights, N, std):
    new_particles = np.empty(N, 3)
    for i in range(N):
        index = random.choice(np.arange(0,len(weights), 1), p=weights)
        x = random.normal(particles[index][0], std)
        y = random.normal(particles[index][1], std)

        new_particles[i] = [x, y, particles[i][2]]
    return new_particles

def driver_code(list_of_observation_and_control_data, N, step_length_std, heading_std,
 observation_std):
    #Create a starting set of particles with random position and heading
    particles = np.empty((N, 3))
    particles[:, 0] = uniform(x_min, x_max, size=N) # X coord value
    particles[:, 1] = uniform(y_min, y_max, size=N) # Y coord value
    particles[:, 2] = uniform(0, 360, size=N) # Heading value

    for data_point in list_of_observation_and_control_data:
        if data_point["Control_Data"] != None:
            particles = apply_movement(particles, step_length, data_point["heading"],
    step_length_std, heading_std)

        if data_point["Observation_Data"] != None:
            weights = calculate_weights(particles, data_point["Observation_Data"],
    observation_std)
            plot_particles(particles, size=weights)
            particles = resample(particles, weights, N, observation_std)

            #Converge particles over time by minimising std
            if observation_std > 0.2:
                observation_std -= 0.1
```

Listing 5.2: Particle filter algorithm in Python

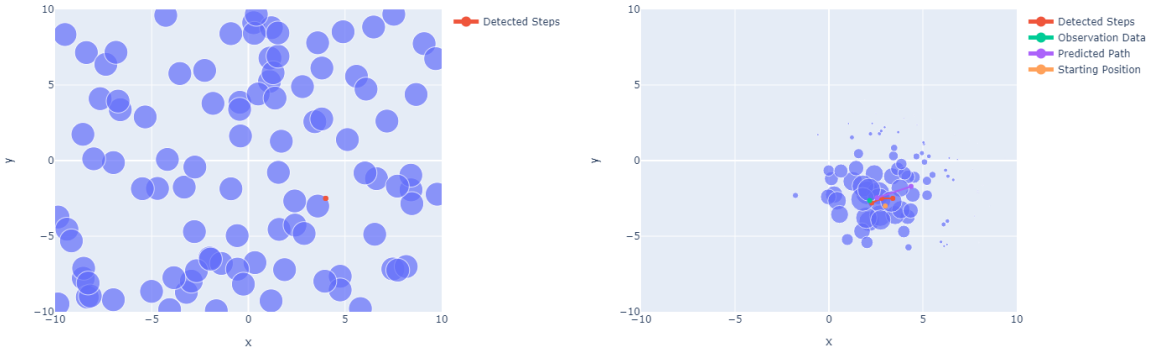


Figure 5.11: The figure on the left show the first generation of particles where positions are random and the weights are uniform (indicated by the size of each particle), followed by resampling around the observation data indicated by the green dot on the right figure

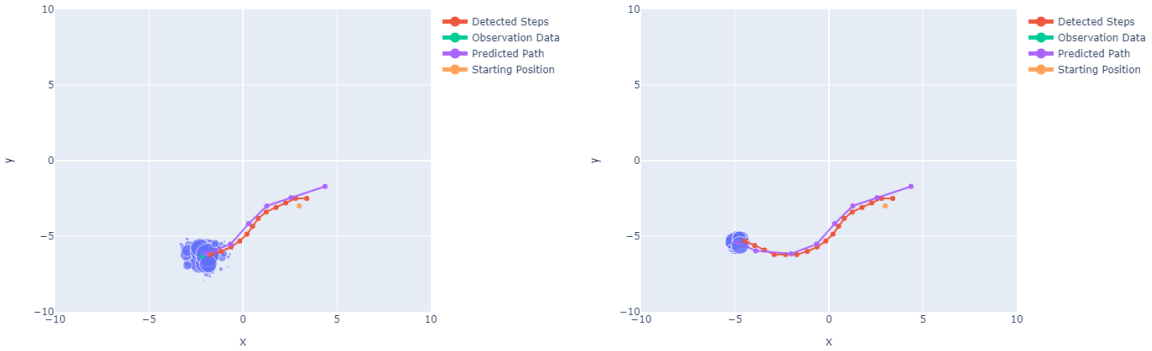


Figure 5.12: As the algorithm progresses, the particles increasingly converge and the predicted position (centroid of the cluster) can be taken even by eye.
The predicted path (calculated centroid) is highlighted in purple at every time their weights were calculated.
Each dot in `Detected Steps` and `Predicted Path` are a time-step where their respective data was available and calculated

Note that the standard deviation used in the calculation of weights and the generation of the new population of particles is reduced as the algorithm proceeds. If the algorithm succeeds, then the effect of this gradual reduction in the standard deviation, is that the particles tend to cluster closer to the observation data. However, in the case that the quantity of observation data is severely lacking in comparison to the control data, the population of particles can drift quite far from the observation by the time it is available to update the particles. In this case, the closest particles to the observation can be more than a single standard deviation away from it. This means the noisy selection of the new particle coordinates is not able to fully correct the drift. This becomes an issue when we apply the particle filter to real data, as shown in Chapter 6.

5.3 Problems Encountered

- The first problem I encountered was the fact I could not find a tool to do any Wi-Fi RSS data gathering. Despite being stated in the project brief that building a smartphone app would not be necessary, I found myself needing to build two. One for the offline phase and one for the online phase. Which leads to the next problem.
- Since the apps were Android apps for my Android smartphone, I called methods for Wi-Fi scanning through the Android API. The API however would restrict the frequency of repeated Wi-Fi scans. "Each foreground app can scan four times in a 2-minute period. This allows for a burst of scans in a short time." [35]. Fortunately, there is a developer setting called "Wi-Fi scan throttling" that can be switched off which increases the frequency of allowed API calls to scan.

  However, in my experience, these more frequent scans are degraded in quality and do not seem to be accurate information but rather data perhaps already stored in memory. I elaborate further on this lack of quality characteristic in Section 6.1.
- Another problem I found is somewhat related to the previous problem. I had plans (stated in the advanced objectives) to implement an orientation model and an orientation-specific RSS grid model for the offline phase but they were deemed impossible as they relied on a compass reading form the smartphone. I found that the magnetometer in my smartphone was highly inadequate and extremely unreliable even when used outdoors where there should have been no interference.

  My only solution was to use the inertial sensors to extrapolate this information. The heading would only be a local heading and would not be grounded in the real world but would have to suffice as heading information, even a local one, would be required for PDR. The fact that it is a local heading should not affect the Core Objectives too much, but implementing the above-mentioned advanced objectives would be impossible.

  As it was a local heading that was initialised at run-time, it would be susceptible to drift. I found it drifted around -1 degrees every 5-15 seconds.

5.4 Project Schedule

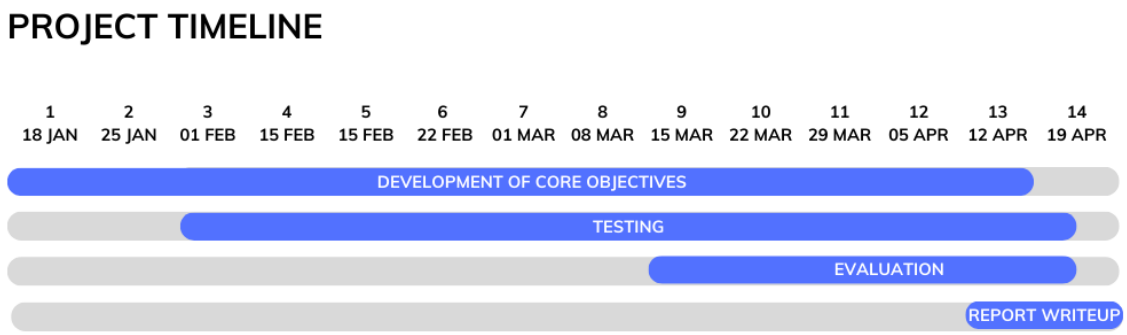


Figure 5.13: Project Timeline

Figure 5.13 depicts the actual timeline of implementation during the project. Most of the time was spent developing the core objectives (getting the system to work) and testing the implementation. As the deadline neared, I had to start evaluating the work I had done so far to prepare to write my thoughts for this report.

The timeline differs from the projected timeline (Figure 4.1) a little but largely remains the same. The data collection is considered a part of the development of the core objectives in the actual timeline.

Chapter 6: **Results**

The Results section will be split up into 4 sections, each highlighting the results garnered in the Achieved Implementation (Section 5.2). There will be a small discussion at the end of each section concluding how the results can be interpreted.

6.1 Data Gathering

The implementation started with gathering data and developing tools to gather that data. Though difficult and laborious, the tools developed were able to gather the data I required. Although in the end, they turned out to be quite complex and difficult to maintain as spending significant time on it was (at the time), not in my best interest. It was not even part of the planned implementation and I was aware it might put me behind schedule because of that. A snippet of some of the raw data I acquired is in Figure 5.5 and Figure 5.6. App B, the tool for the online phase recorded inertial and heading data approximately every $\leq$20 milliseconds and produced quite a significant amount of data for each trial which was around 30 seconds long. The Wi-Fi RSS scans in contrast ammount to around 7 scans in 30 seconds.

6.1.1 Discussion

Acquiring the necessary data for the offline phase for the relatively small 12m x 6m area took $\sim$20 minutes. Scaling this up would take considerable labour effort. This is due in part to the lack of rapid scans. On average, over a 30-second period, I could get about 7 scans. Ideally, I would hope for at least one scan per second for both the offline and online phases.

Disregarding the infrequent scans, the quality of each and every scan is simply not enough. After delving into the raw data and my own personal experience of gathering all these scans. It is clear that the software/hardware within my smartphone does not prioritise accurate RSSI information. I have tested this theory by leaving my smartphone to scan without moving and acquiring a multitude of scans that differ enormously from each other. At the same time, I have also done multiple scans a few seconds apart while walking and receiving the exact same scan, or a very similar scan, with only one missing MAC address in the latter scan or an RSSI value in the one MAC address that has ± 1 of difference.

These scans are clearly not indicative of what an accurate RSSI scan would look like and does not guarantee that the scans taken during the online phase would be similar to the fingerprinting scans taken during the offline phase.

6.2 Wi-Fi RSS Fingerprinting data processing

The Wi-Fi RSS Fingerprinting system theoretically succeeded. It was able to convert the raw data into predicted coordinates. A snippet of the output is shown in Figure 6.1.

```
[{'error': 13, 'X': '1.0', 'Y': '-1.0'}, {'error': 13, 'X': '0.0', 'Y': '-1.0'}, {'error': 16, 'X': '0.0', 'Y': '-1.0'}]
Predicted Coordinates: 0.3333333333333333,-1.0

[{'error': 4, 'X': '3.0', 'Y': '-1.0'}, {'error': 5, 'X': '3.0', 'Y': '-3.0'}, {'error': 5, 'X': '1.0', 'Y': '0.0'}]
Predicted Coordinates: 2.3333333333333335,-1.3333333333333333

[{'error': 4, 'X': '3.0', 'Y': '-1.0'}, {'error': 5, 'X': '3.0', 'Y': '-3.0'}, {'error': 5, 'X': '1.0', 'Y': '0.0'}]
Predicted Coordinates: 2.3333333333333335,-1.3333333333333333

[{'error': 2, 'X': '0.0', 'Y': '-2.0'}, {'error': 2, 'X': '0.0', 'Y': '-2.0'}, {'error': 3, 'X': '0.0', 'Y': '-2.0'}]
Predicted Coordinates: 0.0,-2.0
```

Figure 6.1: The line before each predicted coordinate are the K grid points of least error. In this case $K = 3$

In an attempt to improve accuracy, I spent a lot of time tweaking variables (to no avail). Variables such as:

- Taking multiple scans for each grid point during the offline phase, and either averaging out matching RSSI values or just having more scans in total to use
- Tweaking with variables like different values for K (in KNN) and RSSI cutoff points e.g. any RSSI value less than -80 would be disregarded (Wi-Fi RSSI values typically range from -50 to -90).
- Tweaking the number accepted number of matching MAC addresses e.g. if the same address doesn't show up more than x times, disregard that address.

6.2.1 Discussion

The predicted coordinates were just too inaccurate and seemingly random.

I found the main problem was due to the source of the data. This weakness is discussed previously in the Data Gathering Results (Section 6.1).

6.3 Inertial and Heading data processing

The inertial and heading data gathered was reliable and would be accurate at detecting steps given a little tweaking of the rolling mean and the thresholds.

The integration of the inertial and heading data into a PDR system also worked well despite the associated inevitable drift of an inaccurate step length measurement and local heading drift.

Figure 6.2 shows the drift associated with the PDR system

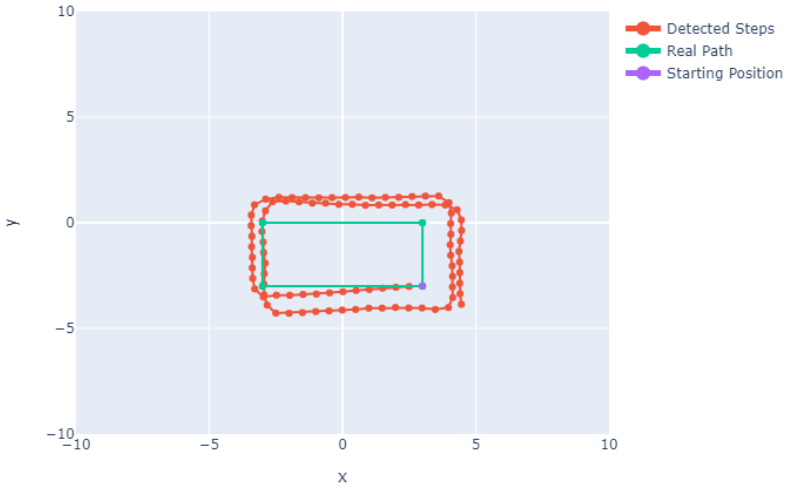


Figure 6.2: The green lines show the real path taken which were 2 clockwise "revolutions" ending back at the starting position.
Time taken: ∼1 minute to walk

6.3.1 Discussion

Despite the minor problems of drift and a hard-coded step length, the PDR system worked well. All the trial runs for the online phase were around 30 seconds to minimise the effect of drift to acquire proper results. I suspect if the duration of the online phase was longer, the minor problem of drift would become quite major. One solution would be to somehow force a calibration by the user every 2 minutes by perhaps standing next to an indoor landmark and facing a particular direction.

6.4 Particle Filter

The particle filter is the main entity to achieve the goal of the project, to achieve indoor positioning on a smartphone.

Figure 6.3 and Figure 6.4 are snippets of the particle filter output given data from a 34-second trial in a 12m x 6m area. (There were only 7 total plots as there were 7 scans of observation data). This trial used real data that was gathered and processed with the previous steps and it was too inaccurate and almost random. As the observation data, in this case, was lacking in quality and quantity, the particle filter was not able to accurately determine the true position at any given moment in time.

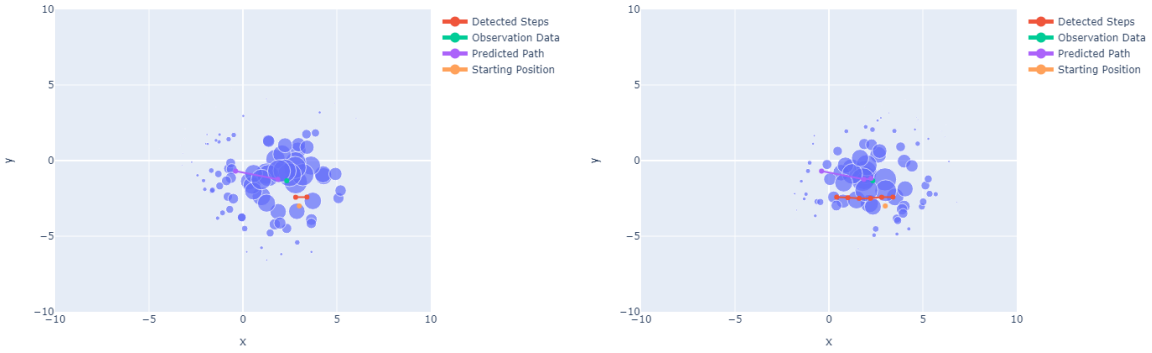


Figure 6.3: Snippets of the particle attempting to follow the real location.
The bounds for the area and particles are (-3, 3) for the x-axis and (0, -3) for the y-axis
These two iterations happen next to each other, showing the lack of available observation data between two significant travel distances.

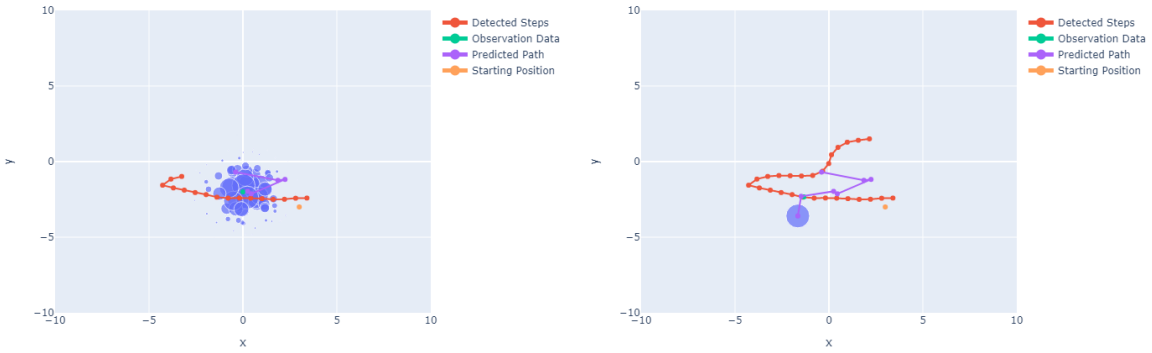


Figure 6.4: As is a characteristic of KNN, the observation data tends towards the middle.
The path taken by the red `Detected Steps` is approximately in line with the real path taken during the trial.
In the right figure, the distance from the centroid of the cluster to the real path is 6.3675m

Due to the nature of the particle filter, as long as there is accurate control and observation data, there doesn't actually have to be a large quantity of observation data. Figure 6.5 below demonstrates this quite well. And for confirmation, the particles only re-sample when there is available observation data and can only re-sample based on the weights of previous existing particles, proving that if the control data and observation data are accurate, the quantity of observation data is not necessary. Though there is still an argument to be made for a larger quantity of scans to make up for the low-quality nature of each scan.

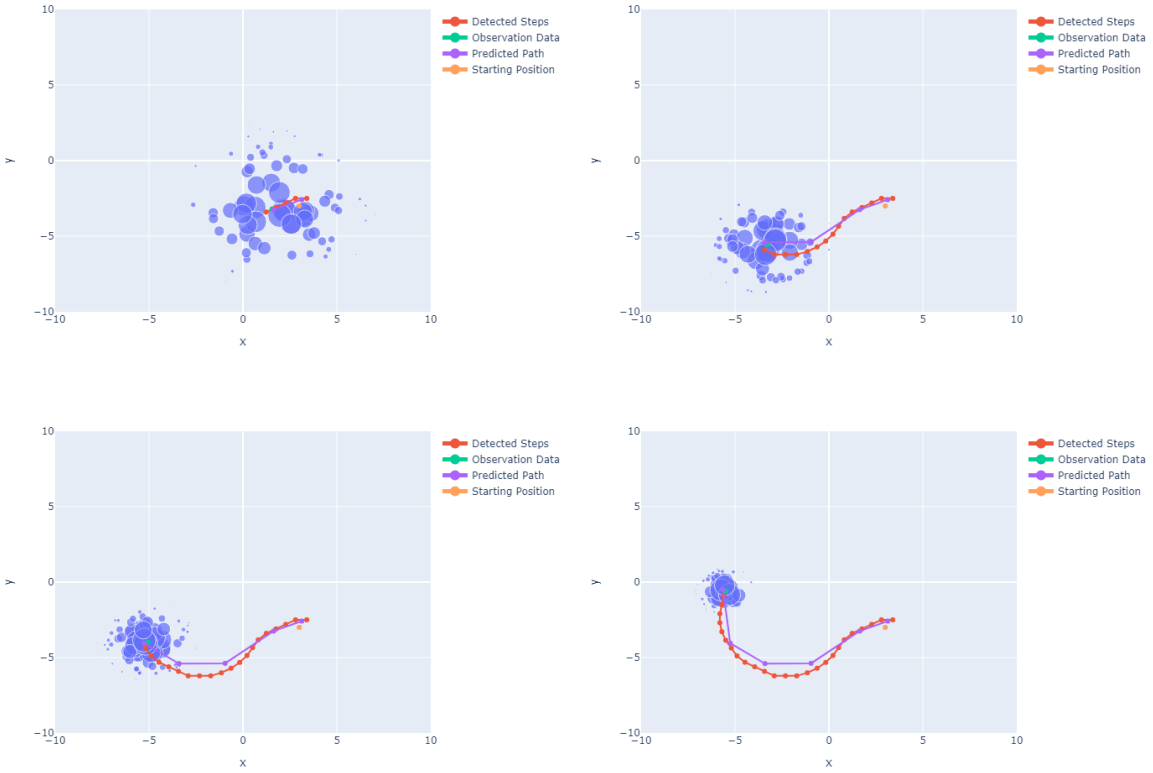


Figure 6.5: The 4 plotted outputs of the particle filter above are consecutive iterations as there are only 5 scans of observation data and 31 total steps.

However, in this case, the real data was both lacking in quantity and quality. The blow to inaccuracy was lightened by the accurate control data, but the quality remains the key factor to the particle filter's success.

6.4.1 Discussion

Despite the inability of the particle filter to locate the position of the smartphone during the conducted trials, the particle filter was a success. Given artificial test data, which consisted of reliable observation and control data. The particles were able to locate the real position quite easily (shown in Figure 5.11 and Figure 5.12). So theoretically, the implementation achieved during the project attained a satisfactory outcome, but practically, it failed to reach the minimum requirement and a satisfactory result using real-world data.

Chapter 7: **Future Work**

There is definitely room for more work, not just in this project but in this area of research as a whole.

As I've identified, the underlying problem with my implementation is the lack of quality Wi-F i RSS data which may be due to a number of things:

- The hardware/software within *my* smartphone is of low quality and is incapable of reading accurate Wi-Fi RSS data.
- The RSS fingerprinting map recorded may have been non-indicative of quality Wi-Fi RSS fingerprints as orientation-specific RSS fingerprints were unavailable. This can be a problem as the human body can diminish Wi-Fi signals, thus resulting in a different RSS fingerprint.
- The methodology may have been wrong. Perhaps my implementation followed an erroneous path or an error I made in the code perhaps is hampering the true result of the work achieved.

I believe that the real problem is to do with my smartphone's sensors being of low quality. This is because I already know that the magnetometer in my smartphone is highly unreliable, it is not a guarantee that other sensors don't also suffer from the same low-quality characteristic.

Not only that, my methodology of using a particle filter with inertial sensors and Wi-Fi RSS Fingerprinting has been achieved before. Liu et. al[28] implemented a similar system and achieved room-level accuracy (A diagram of their implemented system is shown in Figure3.8). There are some differences notably in our difference in environments. Their test environment consists of one narrow corridor with 12 rooms occupying all sides of the corridor. The characteristic of focusing only on room-level accuracy allows for clever methods to be used like if the majority of the particles were inside a particular room, then all re-sampled particles will be constrained to that room. As well, this characteristic makes fingerprinting easier to achieve as the walls of each room diminish Wi-Fi signals, giving each room a more unique fingerprint. An open floor environment would have RSS fingerprints that resembled each other more closely and may not give each grid point a sufficiently unique fingerprint.

With all this in mind, there is a plethora of ideas for future work:

- The exploration of using different smartphones that range in brand and price.
- Reducing the scope of the goal to perhaps room-level localisation and scaling up from there.
- Changing the test environment to a more physically crowded environment. i.e. an area with more walls and differentiating characteristics to give each RSS grid point a more unique RSS fingerprint.
- Implementing an orientation-specific RSS fingerprint map with less dense grid points to compensate for labour and similar RSS fingerprints.
- Changing more variables in the implementation. There is a plethora of variables that are viable for experimentation and not just simple things like standard deviation. Perhaps the Gaussian noise that I implemented does not fit the model well or that the likelihood function is not an accurate representation of probability for this application.

Chapter 8: Summary and Conclusions

In summary, I conducted background research on the topic of indoor positioning and explored most of the relevant aspects of it in relation to smartphones:

- The different indoor positioning techniques and how they compare to each other.
- How and Why a Wi-Fi-based indoor positioning system is the most applicable to a student project due to cost constraints etc.
- How a Wi-Fi based RSS Fingerprinting system works and why it is more effective than Triangulation.
- Why using smartphone sensors such as inertial sensors and compass sensor is effective for mobile applications, and how they can be applied.
- And finally, how fusion technology is used to combine different sources of data into one output.

I described some achievable objectives based on my research and proposed a plan to implement a solution to achieve those objectives.

I developed some tools to gather the necessary Wi-Fi RSS and smartphone sensor data. Using the data, I processed them into the two inputs needed for my implemented particle filter solution, observation data and control data. Finally, I tested and optimised my solutions in an attempt to improve them and failed to achieve indoor positioning with real-world data. However, it was achieved theoretically with manual test data that showed given accurate data, the system achieved what it set out to do.

The core objectives I set out to achieve and why I did or did not achieve them:

1. Develop an indoor positioning system that takes estimate Wi-Fi location data as input, compares it to collected Wi-Fi data about the test environment, and outputs an estimate location.

   - This was achieved with the Data gathering tools and the Wi-Fi RSS Fingerprinting system (Section 5.2.1 and Section 5.2.2).

2. Further develop the system to incorporate tracking features.

   - The data gathering and the systems to process the data were inherently capable of tracking a real position and not just static positions.

3. Introduce data fusion technology in the form of inertial sensor data, that can be found and generated from smartphones to improve accuracy of static location and tracking tests.

   - The fusion technology used was the particle filter and was implemented successfully (Section 5.2.4).

4. Introduce other testing variables such as spinning on the spot, running, walking backwards etc. to measure how much the human body interferes with Wi-Fi signal propagation and detection.

- This was done initially but when the weaknesses of the data were starting to appear, more attention and time was placed on trying to get the system working accurately with just the most optimal testing situation to at least achieve a satisfactory result. So in the end, restrictions on the testing situation like the above were not introduced.

5. Evaluation of the system

   (a) Achieve a mean accuracy of < 2.0m.

   (b) Achieve a similar accuracy after 150m of continuous tracking.

   (c) Test the robustness and reliability of the system, for example, moving along the edge of the map, walking in a quick zig-zag pattern, sidestepping and changing direction many times.

   - The whole suite of systems was evaluated and tested and this is discussed in Chapter 6. The most useful evaluation was the use of test data to confirm whether or not the systems worked correctly without the variable of erroneous data. The sub-goals were not achieved with due to the fact hat in the end, the indoor positioning system did not work with the real-world data gathered.

The advanced objectives were not achieved or even pursued as the majority of the effort went towards accomplishing the Core Objectives or in another case, they were not possible due to certain limitations that were discovered during implementation.

Although the core objectives were not achieved, the implementation of `Indoor positioning with smartphones` was otherwise a success from a theoretical standpoint. From the work I have conducted, I have seen the limitations but also the potential of this area. There is definitely room for future development, even from within the scope of my project, as I mentioned above in Chapter 7.

Bibliography

1. Farid, Z., Nordin, R. & Ismail, M. Recent Advances in Wireless Indoor Localization Techniques and System. *Journal of Computer Networks and Communications* **2013,** pp. 1–13 (2013).
2. Brena, R. F. *et al.* Evolution of Indoor Positioning Technologies: A Survey. *Journal of Sensors* **2017,** pp. 1–22 (2017).
3. BASRI, C. & El Khadimi, A. *Survey on indoor localization system and recent advances of WIFI fingerprinting technique* in *2016 5th International Conference on Multimedia Computing and Systems (ICMCS)* (2016), pp. 253–259.
4. Kjærgaard, M. B. *et al. Indoor Positioning Using GPS Revisited* in *Pervasive Computing* (2010), pp. 38–56.
5. Diallo, A., Lu, Z. & Zhao, X. Wireless Indoor Localization Using Passive RFID Tags. *Procedia Computer Science* **155,** pp. 210–217 (2019).
6. Chen, Z., Zou, H., Yang, J., Jiang, H. & Xie, L. WiFi Fingerprinting Indoor Localization Using Local Feature-Based Deep LSTM. *IEEE Systems Journal* **14,** pp. 3001–3010 (2020).
7. Apple. *Apple AirTag* Accessed Nov. 29, 2022 [Online]. https://www.apple.com/ie/airtag/.
8. Infsoft. *Ultra-Wideband for indoor positioning – RTLS by infsoft.* Accessed Nov. 29, 2022 [Online]. https://www.infsoft.com/basics/positioning-technologies/ultra-wideband/.
9. Navigine. *Working Principles and Advantages of the Ultrasonic System.* Accessed Nov. 29, 2022 [Online]. https://navigine.com/blog/how-ultrasonic-indoor-positioning-works/#:~:text=The%5C%20working%5C%20principle%5C%20of%5C%20ultrasonic,the%5C%20sensor%5C%20to%5C%20the%5C%20receiver..
10. Sonitor. *Accurate ultrasound positioning, reliable RTLS.* Accessed Nov. 29, 2022 [Online]. https://www.sonitor.com/ultrasound-innovation.
11. Zegeye, W. K., Amsalu, S. B., Astatke, Y. & Moazzami, F. *WiFi RSS fingerprinting indoor localization for mobile devices* in *2016 IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)* (2016), pp. 1–6.
12. Wang, X., Wei, X., Liu, Y., Yang, K. & Du, X. Fingerprint-based Wi-Fi indoor localization using map and inertial sensors. *International Journal of Distributed Sensor Networks* **13,** pp. 1550–1329 (2017).
13. Du, X., Liao, X., Liu, M. & Gao, Z. CRCLoc: A Crowdsourcing-Based Radio Map Construction Method for WiFi Fingerprinting Localization. *IEEE Internet of Things Journal* **9,** pp. 12364–12377 (2022).
14. Bahl, P. & Padmanabhan, V. *RADAR: an in-building RF-based user location and tracking system* in *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)* **2** (2000), pp. 775–784 vol.2.
15. Kessel, M. & Werner, M. *SMARTPOS: Accurate and Precise Indoor Positioning on Mobile Phones* in *Proceedings of the first international conference on mobile services, resources, and users, MOBILITY* (2011), pp. 158–163.
16. Liu, F. *et al.* Survey on WiFi-based indoor positioning techniques. *IET Communications* **14,** pp. 1372–1383 (2020).

17. Chan, S. M. & Sohn, G. Indoor localization using wi-fi based fingerprinting and trilateration techiques for lbs applications. *International Archives of the Photogrammetry* **38,** Chapter 26 (2012).
18. Deng, Z.-A., Hu, Y., Yu, J. & Na, Z. Extended Kalman Filter for Real Time Indoor Localization by Fusing WiFi and Smartphone Inertial Sensorss. *Micromachines* **6,** pp. 523–543 (2015).
19. Li, F. *et al. A Reliable and Accurate Indoor Localization Method Using Phone Inertial Sensors* in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing* (2012), pp. 421–430.
20. Yang, Z. *et al.* Mobility Increases Localizability: A Survey on Wireless Indoor Localization Using Inertial Sensors. *ACM Comput. Surv.* **47,** pp. 1–34. ISSN: 15577341 (Apr. 2015).
21. Shu, Y. *et al.* Gradient-Based Fingerprinting for Indoor Localization and Tracking. *IEEE Transactions on Industrial Electronics* **63,** pp. 2424–2433 (2016).
22. Potortì, F., Crivello, A., Palumbo, F., Girolami, M. & Barsocchi, P. *Trends in smartphone-based indoor localisation* in *2021 International Conference on Indoor Positioning and Indoor Navigation (IPIN)* (2021), pp. 1–7.
23. Zhang, R., Bannoura, A., Höflinger, F., Reindl, L. M. & Schindelhauer, C. *Indoor localization using a smart phone* in *2013 IEEE Sensors Applications Symposium Proceedings* (2013), pp. 38–42. ISBN: 9781467346351.
24. Beauregard, S. & Haas, H. *Pedestrian dead reckoning: A basis for personal positioning* in *Proceedings of the 3rd Workshop on Positioning, Navigation and Communication* (2006), pp. 27–35.
25. Klingbeil, L. & Wark, T. *A Wireless Sensor Network for Real-Time Indoor Localisation and Motion Monitoring* in *2008 International Conference on Information Processing in Sensor Networks (ipsn 2008)* (2008), pp. 39–50.
26. House, S. *et al. Indoor localization using pedestrian dead reckoning updated with RFID-based fiducials* in *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society* (2011), pp. 7598–7601.
27. King, T., Kopf, S., Haenselmann, T., Lubberger, C. & Effelsberg, W. *COMPASS: A Probabilistic Indoor Positioning System Based on 802.11 and Digital Compasses* in *Proceedings of the 1st International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization* (2006), pp. 34–40.
28. Liu, Y., Dashti, M., Abd Rahman, M. A. & Zhang, J. *Indoor localization using smartphone inertial sensors* in *2014 11th Workshop on Positioning, Navigation and Communication (WPNC)* (2014), pp. 1–6.
29. MATLAB. *Understanding the particle filter | | autonomous navigation, part 2.* YouTube. July 2020 [Online]. https://www.youtube.com/watch?v=NrzmH_yerBU.
30. Gustafsson, F. Particle filter theory and practice with positioning applications. *IEEE Aerospace and Electronic Systems Magazine* **25,** pp. 53–81 (2010).
31. Doucet, A., de Freitas, N. & Gordon, N. in *Sequential Monte Carlo Methods in Practice* (eds Doucet, A., de Freitas, N. & Gordon, N.) pp. 3–14 (Springer New York, New York, NY, 2001).
32. Thrun, S. *Particle Filters in Robotics.* in *UAI* **2** (2002), pp. 511–518.
33. Udacity. *Using Particle Filters with Computer Vision.* Udacity. July 2015 [Online]. https://learn.udacity.com/courses/ud810/lessons/3d4efe79-de5a-4177-b3a2-66117da04f30/concepts/d7f9e1bc-5c8b-4a22-84fd-1a40e4021059.
34. Zhu, N., Zhao, H., Feng, W. & Wang, Z. A novel particle filter approach for indoor positioning by fusing WiFi and inertial sensors. *Chinese Journal of Aeronautics* **28,** pp. 1725–1734 (2015).
35. Android. *Android Developers Docs Guide* Accessed March. 10, 2022 [Online]. https://developer.android.com/guide/topics/connectivity/wifi-scan.