Minet Magnetic Indoor Localization

Michael Drake

Minet: Magnetic Based Indoor Localization

by
Michael Drake

A thesis submitted to the faculty of The University of Mississippi in partial fulfillment of the requirements of the Sally McDonnell Barksdale Honors College.

Oxford
May 2020

Approved by

Advisor: Dr. Feng Wang

Reader: Dr. Yixin Chen

Reader: Dr. Dawn Wilkins

ABSTRACT

Indoor localization is a modern problem of computer science that has no unified solution, as there are significant trade-offs involved with every technique. Magnetic localization, though less popular than WiFi signal based localization, is a sub-field that is rooted in infrastructure-free design, which can allow universal setup. Magnetic localization is also often paired with probabilistic programming, which provides a powerful method of estimation, given a limited understanding of the environment. This thesis presents Minet, which is a particle filter based localization system using the Earth's geomagnetic field. It explores the novel idea of state space limitation as a method of optimizing a particle filter, by limiting the scope of possibilities the filter has to predict. Minet is also built as a distributed model, which can be easily modified to integrate new technologies. Minet showed promising results, but ultimately fell short of its accuracy goal. Minet had some inconsistencies that led to these accuracy issues, but these issues have been diagnosed and can be fixed in future updates. Finally, potential improvements of Minet's base components are discussed, along with how different technologies such as a Deep Learning model can be implemented to improve performance.

DEDICATION

In loving Memory of Mark Drake (3/14/61 - 3/26/2019). Thank you not only for all the years of parental guidance but also the great friendship.

For my dear sister, Nicole, future brother-in-law, Taylor, and wonderful Mother, Mom, thank you for always being the loving, supportive, and fun people you've always been, and I've always needed.

ACKNOWLEDGEMENTS

My sincerest gratitude to Dr. Feng Wang, who guided me through the thesis process.

A special thanks to my friends Gabe and Jared for being the funniest guys I know. Along with Gary, Ivan, Jarad, Kyle, Antoine, Wesley, and Jeremiah for fun times in the Discord.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

While outdoor navigation has been practically solved by GPS since around 2005 [9], indoor localization is a modern problem without a universal solution. There have been many feasible and robust efforts, but each solution has its drawbacks. With any given solution, there is always consideration of cost of infrastructure, difficulty of implementation, and performance.

There are many applications for indoor localization. One natural possibility is to fill in the gaps that are not served by GPS, such as interiors of buildings or places where there is no clear pathway to positioning satellites, such as airports or in dense urban areas. There are many other possibilities in the realm of public safety.

Indoor localization can provide a passive means of protecting senior citizens [5]. With a rapidly aging population, there are more elders, who need specialized care, which involves monitoring in case of accidents. Localization can provide a passive means of monitoring, where perhaps by a learning model, a computer could detect an irregular behavior that could be a sign of an accident and alert staff.

Another application is when an emergency occurs (e.g., a building catches fire); having an effective route out of a building is of utmost importance. With proper deployment an application could potentially help people escape unfamiliar buildings, or possibly ease the panic of evacuation by showing a clear, quick route out of a building. Indoor localization could also potentially aid in rescue efforts, by signalling where trapped civilians are, who may be unable to communicate [2].

The form of indoor localization presented in this thesis is a Geomagnetic based model using a smartphone and a particle filter. I chose this method because it allows a map that can be created using only a smartphone, which brings potential for crowdfunded indoor mapping, and passive improvement using soft computing techniques (e.g., clustering [1] or deep learning[8]). Particle Filters have proven to be a fairly robust and stable form of localization [11].

Minet is named for the first two letters in 'Michael' and the last three letters in 'magnet.' While particle filtering can be quite accurate, it is computationally expensive, so Minet mitigates the impact on the phone, by taking the computation off the device and connecting it to a server, which has also been demonstrated to be promising in the Maloc system [10]. While taking computation to a server may introduce elements of latency and/or external dependency, the distributed system is more extensible. Since the device does not have to do computation, the only concern is data transmission, which is already a staple of other localization techniques. Relieving the device of computation allows the back-end localization to be modified, without impacting the device. For example, different estimation systems such as Deep Learning models can be substituted, as the device only expects a coordinate, and has no knowledge of the technology behind the scenes. As will be discussed in the Further Discussion section, this also opens up a lot of possibilities for improvements both to the magnetic mappings and accuracy. The Minet system creates an extensible system, which can easily be adapted to new technologies.

Another improvement Minet intends to introduce is to limit the state space of the problem. The state space represents all of the possibilities for a state, or position, in a given environment. By limiting this, the particle filter will face less computation, which will allow it to compute and return a response faster. This speed can allow a device to send data in real-time. Minet transmits data on each step. This is different from Maloc, which sent a transmission every 2-3 steps and thus had to perform step estimation to cover localization in the time between transmissions [10]. While having real-time transmission, could potentially provide a more responsive navigation experience, it sacrifices battery efficiency. Limiting the state space is also a rather heavy optimization that emphasizes speed, which hinders accuracy, as discussed in the Evaluation section. The state-space optimization is the most novel idea of Minet, and has important trade-offs, but shows potential for an optimized system, which can operate in real-time.

In this thesis, Minet explores some of the challenges, trade-offs, and benefits of indoor localization using geomagnetic data and a particle filter. In Chapter 2 - Background, different technologies and concepts involved in this thesis are detailed and explained. Chapter 3 - Infrastructure provides a general overview of the components of the Minet system and how they interact with each other. Chapter 4 - System gives an in-depth description of the most important features of Minet and how they are implemented. In Chapter 5 - Evaluation, the performance of Minet is evaluated and discussed. Chapter 6 - Further Improvement discusses implementation fixes and new technologies that can improve Minet. Chapter 7 - Conclusion and Future Works concludes this thesis and summarizes possible explorations of improvements that can be made to the less successful aspects of Minet.

CHAPTER 2

BACKGROUND AND RELATED WORK

Magnetic field strength is measured in its intensity in direction along x, y, and z axes. On a smartphone, these axes align: x is the width of the device, y is the height of the device, and z is the distance of the device from the ground (if the device is lying flat). Given the dependency of the measurements on the orientation of the device, the device must be localized in the same orientation as that of when the magnetic field was mapped. This limitation can be addressed by increasing the map coverage or predicting device movement; however in this thesis, the device is assumed to be flat.

One concern with the viability of magnetic localization is the stability of the interference. The composition of buildings gives enough unique disturbance to the geomagnetic field of the Earth to be used for localization [6]. These disturbances come from ferromagnetic materials in buildings. The primary contributor to this disturbance is steel, which is used in some facet in almost all modern buildings. Studies into the stability of this disturbance have been performed and shown promising results. The authors of [8] found that while sensor data may not always have the same readings over time, the overall magnetic pattern does not change. They also found in small area testing that the geomagnetic field could be used for precise positioning. In another study, the authors recorded magnetic samples in 10 different locations at 5 different times, and concluded their findings confirmed the reliability of magnetic fingerprinting for indoor localization. They also found that disturbances are distinct enough for localization.

Localization of a smartphone inherently brings a lot of uncertainty. The phone does not control its own movement and only has access to noisy sensor measurements to localize itself. Because of these factors, a probabilistic approach is necessary to provide accuracy. This approach does not demand a perfect environmental model (i.e knowledge of possible state, and the states of other entities such as pedestrians), but still provides accuracy [7]. The idea of this probabilistic approach is to infer a state cannot be observed through measuring sensors, such as an arbitrary coordinate on a map. The state includes information about the environment and the device: the pose, velocity, and landmarks. The pose is all positional vectors of the device including (x, y, z) coordinates, and orientation. The velocity represents any changes in the pose, and the landmarks are any static member of the environment. The state changes through a control action, which in this case is a human carrying the device, U_t. The control action changes the state, and the observation data depends on the state. Due to Bayes rule, we can inverse this relationship of observation data and state and infer the hidden X_t from the observable Y_t. Since this state, X_t cannot be measured directly, belief distributions, $bel(Y_t)$ are used. The belief distribution is a conditional probability distribution of X_t given the posterior distribution. $Bel(Y_t)$ represents the probability of seeing Y_t, given the prior distribution. In a Bayes filter, this belief distribution is then update with a control action and observable data. This is the foundation of probabilistic state estimation. The particle filter is an extension of the Bayes filter that provides greater accuracy in non-linear state spaces.

The advantage of the particle filter is accuracy at the trade-off of computational complexity, where the limitation is often in computational power. A particle is a estimation of the true state, X_t, and a particle filter works with hundreds to thousands of particles, which cluster around the most probable state. The procedure for a particle filter is similar to a Bayes Filter. The input to a particle filter is the particle set from the previous time step, with the last U_t movement and Z_t measurement. U_t is the motion model, observed by the entity that is being localized, in this case, a smartphone. Z_t is a measurement of some observable information from the environment, such as magnetic field readings. The particle filter is a hidden Markov model, where the future state depends only on the current state, and not any past ones. However, this only means that all the information is contained in one state, as a state in the particle filter is created using data from all the previous state. This idea comes from Bayesian statistics, where beliefs are updated with incoming information. Another aspect of hidden Markov systems is that there is an internal state that cannot be measured, so it has to be inferred from observable information. Particle filters work by reducing the error of this inference. The initial state is populated with random guesses, or some guesses derived from a process. However, through the mechanisms of the particle filter these guesses are narrowed around the most probable state. The power of the particle filter comes from the weighing and resampling functions [7]. The weighting function determines how close the guess is to the true state, through some sort of calculation on the observed data. The resampling function determines how to read the weights from the weighting function to determine the distribution of new particles. A good resampling function should not generate the least probable particles, only the ones that appear to be closest to the true state. This probabilistic approach is a much more feasible way of determining a state than to attempt to program for all features in a given environment, although Deep Learning models find success using a similar pattern.

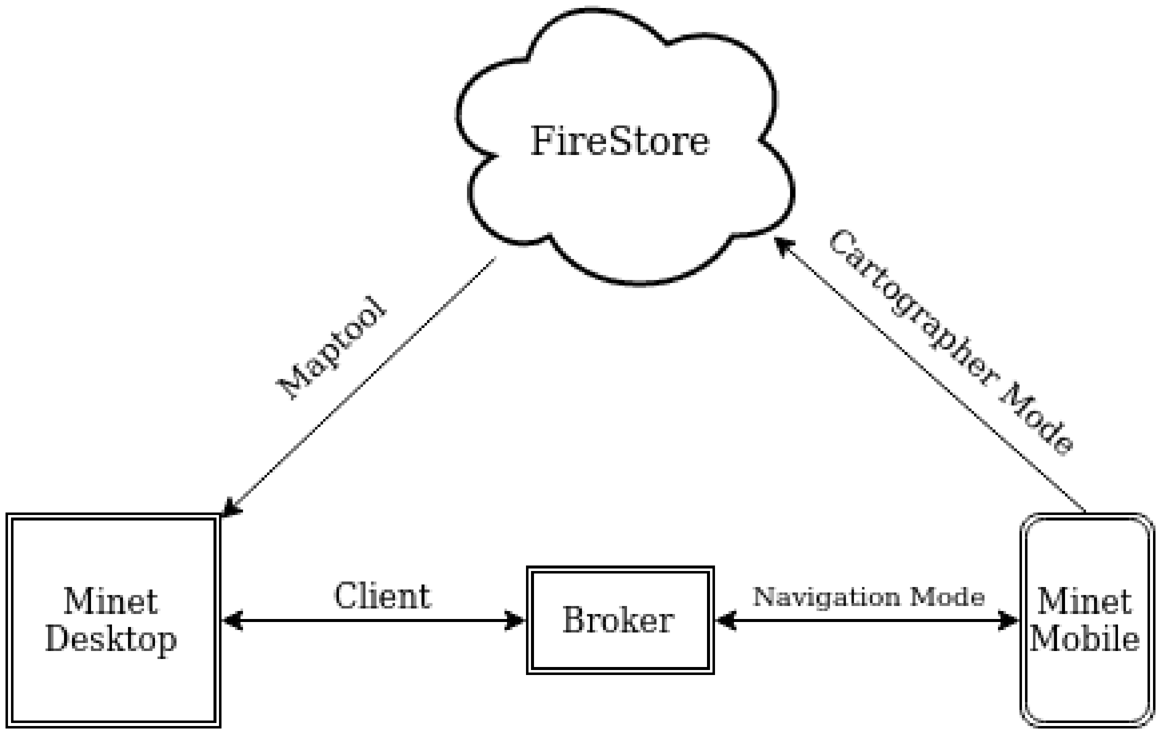
MQTT, or Message Queuing Telemetry Transport Protocol, was chosen for this project, for its lightweight communications among machines. MQTT was designed in 1999 as a simple publish and subscribe protocol. Clients publish and subscribe to a broker based on topics. Data is published with a topic, which allows the broker to send it to the devices that are subscribed to that topic. MQTT allows three levels of reliability, or Quality of Service (QoS); level 0 is used here, as the transmission of sensor data needs to occur as quickly as possible. MQTT level 0 is a Fire-and-Forget model, where the data is transmitted to the broker, with no other confirmation. MQTT messages have an extremely small header at 2 bytes, and is built on top of TCP. MQTT is popular for Machine to Machine (M2M) because of its low latency, lightweight transmissions.

CHAPTER 3

INFRASTRUCTURE

The main infrastructure of the proposed testbed system includes, as shown in Fig. 3.1. an Android app and a desktop server. The app primarily collects and transfers data, while the desktop does processing and returns a coordinate. The communication between devices is performed using a publish and subscribe model, and two real-time databases. This testbed provides a means of experimenting with a distributed particle filter with an optimized state space, which attempts to estimate a step level accuracy using magnetic information in real time.

Figure 3.1: Minet Architecture



3.1 Minet App

The Minet app is relatively simple with two main activities: Cartographer and Navigation. The Cartographer mode is used for creating a magnetic fingerprint map. I will go into greater detail of the individual components in a later section, but functionally, the purpose of this mode is to bind the observed state (x, y, z) to a true state (x, y, θ). The app takes sequential readings and sends them to a FireStore database. The navigation mode sends the current magnetic reading to the Minet desktop, and listens for a coordinate response, displaying it on receipt.

3.2 Minet Desktop

The Minet Desktop has two main functions: map-making and localization. The map-making takes data from the data stored in FireStore. Building the map using the direction to determine the map shape. The localization is done with a customized particle filter that estimates, based on the landmarks created in the map-maker, with the intent of producing step level accuracy. The desktop also has a visual representation of the particle filter and a tool to generate a heat map for a given magnetic fingerprint map.

CHAPTER 4

SYSTEM

4.1 Minet App

4.1.1 Cartographer Mode

Cartographer Mode is the data recording feature of the Minet app. The purpose of the Cartographer Mode is to create a sequence of data that can map a magnetic space to coordinates representing physical spaces. In the approach for this application, a map can be created simply by taking steps. While this is a simple approach, there are a few drawbacks; the user has to be careful when taking measurements, so they do not get an unintended or incorrect shape. Another is that, since the device must be held in a steady orientation, the native step tracking in android is not effective. Thus, the user must manually mark each step. Data collection was a challenge, that changed many times over development. The reason, a step-based approach was chosen, rather than an interpolated state space, was to limit the state space of nearly infinitely many readings, to just those in the path, thus reducing the amount of possibilities for the Particle Filter. However, this also means there are bigger assumptions being made, which could lead to more severe drift. One way to improve this would be using another source of truth such as a LSTM model, which will be elaborated in Chapter 6.

Cartographer Mode makes heavy use of Sensors Fragment, which is a fragment wrapper of a Android SensorManager implementation, which allows it to be easily reused between activities. The Sensors Fragment makes synchronized readings of geomagnetic and accelerometer data, to create Data and MetaData nodes. A Data Node is the observable of magnetic data (x, y, z), and MetaData is (direction, roomId, and stepId), where the roomId can be used by the classifier for room recognition. A MetaData node is created on a step event, along with a Data node, but Data nodes will also be created and mapped to the same MetaData node, if the direction changes without the step changing. These nodes are stored in an HashMap of MetaData to Data and then uploaded to FireStore. The Cartographer Mode displays the output of the Sensor Fragment to the user, so they can see the direction and magnetic magnitudes. It provides access to FireStore, and an input field to send data to a specified location. It also has functionality to provide roomIds; before starting mapping, the user enters the different rooms that will be charted, which creates radio buttons. Then the user can start mapping, and switch between rooms when taking readings, as quick way to populate that MetaData field.

4.1.2 Navigation Mode

The Navigation Mode is the live data mode of the Minet app. This mode connects the broker, which allows bidirectional communication with Minet desktop app. The mobile device sends a request to the desktop, asking for the processed map, which on receipt, renders on the screen. After that, the device can send its live coordinates, through a similar manual step process as that of the Cartographer Mode. Unfortunately, this is the most accurate way, although it not the most user friendly. This is drawback of magnetic localization's reliance on device orientation. After the map has rendered the device is ready for live navigation. It can send a reading on step, which will be interpreted by the desktop. The device will then receive an updated coordinate. On this subscription callback, the device will render the coordinate on the screen. This is a basic rendering of a coordinate map, but could be expanded to show a bitmap representation of the building in the future.

The Navigation Mode uses the Connection fragment to establish a connection to the Eclipse Mosquitto broker, which is hosted on the same box as the Minet desktop application. The connection fragment retrieves the broker address from Firebase, which is only used to store connection strings. Once the fragment has successfully retrieved a connection string, the connect button is enabled, which initializes an MQTT client, imported from the Eclipse Paho library. When a connection is established, the app sends a publish which fires an event on the desktop, that then, returns a list of the points that represent the map. These are then rendered, according to a scalar. The user can then create step events to send live instances of data as JSON through the client's publish method. It then subscribes to the returned coordinate, with a callback to render the coordinate in red. The rendering is done with a simple custom view class with Android Paint.

4.2 Minet Desktop

Minet Desktop contains the most important functions of the project. While the mobile app is mostly used as a sensor, the desktop application does the computation, map processing, broker setup, visualization, and experimental features such as the classifier. The primary functions are map processing, initializing the broker, and running the particle filter. The program's entry point is through client.py, and while broker.py is also an integral part, its functionality is separated from client.py, so that the broker can be hosted on a different machine, if desired.

4.2.1 Broker

Broker.py is the first step in setting up the desktop client. Using the ip module defined in the Helpers section below, gets the machine's current IP address, and then sets it in Firebase. Broker then launches the mosquitto client, using python's sub process module. This process currently only supports Windows, as it has not been needed in a Linux environment that has Mosquitto installed. Linux is often the recommended operating system for a machine that operates as a broker, and through testing has performed with no problems.

4.2.2 Client

Client.py uses the Eclipse Paho python library to build a custom client that subscribes to two topics: "connect" and "realtime." The "connect" topic is published when a mobile device establishes a connection to a broker. On this event, the client publishes “map,” which is a JSON representation of the coordinate map. The “realtime” topic is published by the mobile device on a step event. On receiving this message, the client passes the data to particle filter, which is initialized at class startup. After the particle filter returns a result, the client publishes “result,” which is the hypothesized coordinate. When client.py is run, it retrieves the connection string of the broker from Firebase, the same way as the mobile app. Then it instantiates its class, and runs in a loop waiting for subscriptions.

4.2.3 MapFilter

Algorithm 1: Minet Desktop: Particle Filter

```
Result: A coordinate (x, y)
initialize StateManager ;
initialize n random particles ;
for step received do
    store the heading and observable;
    movement = getMovementFromReceivedDirection(heading);
    update motion model with dynamics_fn;
    use observe_fn to create new hypothesis;
    call weight_fn to re-weight hypotheses;
    call resample_fn to create next generation of particles;
    return most popular hypothesis;
end
display visualization;
```

The dynamics_fn is responsible for updating the motion model. The motion model of this system is a single step in any given direction in the x-y plane. Using the movement, received from the heading, to move each particle by the movement (x, y) coordinate.

The observe_fn translates the received observable (x, y, z) into an (x, y) coordinate using the most similar match in the fingerprint map.

The weight_fn uses a squared error formula

$$mse = (\frac{1}{n})\sum_{i=1}^{n}(y_i - x_i)^2$$

The resample_fn uses the SciPy Cookbook re-sampling function, which returns n particles created at random from their distribution of weights.

The Particle Filter's implementation is based on the pfilter library, but with modified source code, which fixed a bug that was overwriting the initialized particles to values of 0, after they had already been assigned a valid value. The pfilter library is a class based implementation of a particle filter, which accepts functions as arguments in its constructer to make a custom particle filter quickly. However, the particle filter implementation that is used by the client, is actually MapFilter.py, which is a model that instantiates the pfilter class with values from the StateManager class. StateManager.py is another model that wraps mapper.py and presents information about states that is necessary for the MapFilter. MapFilter contains the functions mentioned in Algorithm 1, and also visualization functionality. The visualization is handled by the openCV library which provides a graphics library, and is a simple way of drawing coordinates. The idea behind these to models was to abstract away the back-end logic from the client, allowing it to only have the responsibility of handing connections.

Mapper.py contains all of the methods necessary to do transformations on maps. This has been one of the files that has changed the most over the project as data collection has changed frequently. The primary function is to take a series of magnetic measurements and turn them into a list of coordinates that can be used by the MapFilter and displayed. These measurements can be gathered directly from Firebase or read from a JSON file. The map is created off of heading and steps of the sequence. A coordinate is created with (x, y, θ), where theta is the heading and (x,y) is the current index. When the stepId in the MetaData of the sequence changes, the index moves in the direction of the heading. The movement is defined by 8 increments of 45 degrees, where each is a combination of any movement (positive, negative, or none) of x and/or any movement of y. These states (x, y, θ) are mapped to their Data node observable (x, y, z) and stored in a python dictionary. Mapper can also create visualization data from the coordinates, by first transforming their values relative to zero, so that a 2d array can be initialized. This can then be used by the figure class which creates heatmaps, or by the MapFilter via the StateManager, which creates the map with a scalar.

4.2.4 Helpers

Database.py - Initializes the FireStore client with the account settings file. Defines a root for the collection, which can be specified, but by default is empty. The database.py object can also be initialized with a new root. The primary functionality of this class is to get the collection of fingerprints and write them to a JSON file. This can be done in a few different ways. GetCollection takes a collection path and a limit. The optional path is appended to the root, and the limit parameter limits the amount of results that are returned. This method is more often used to sample the data, while GetCollectionAndWriteDocumentsToFile grabs the whole collection and writes it to a JSON file.

Ip.py - actually has some overlap with database.py in that it accesses a database. However, ip.py uses Firebase, instead of FireStore, because of the simple nature of the data. While it could have been accomplished using database.py, it was simpler just to use a Firebase client directly in this one isolated place. Ip.py initializes this client, and uses subprocess to call Window's "ipconfig" method. This also works with "ifconfig" on Linux. The ip is parsed out and returned. The ip can either be IPv4 or IPv6 as the broker Mosquitto works for both.

Knn.py - is a simple nearest neighbor calculator. It takes in a current state (x, y, θ) and compares it a passed in list of landmarks (from the map). It either finds the nearest neighbor or averages the k nearest neighbors depending on the parameter k.

CHAPTER 5

EVALUATION

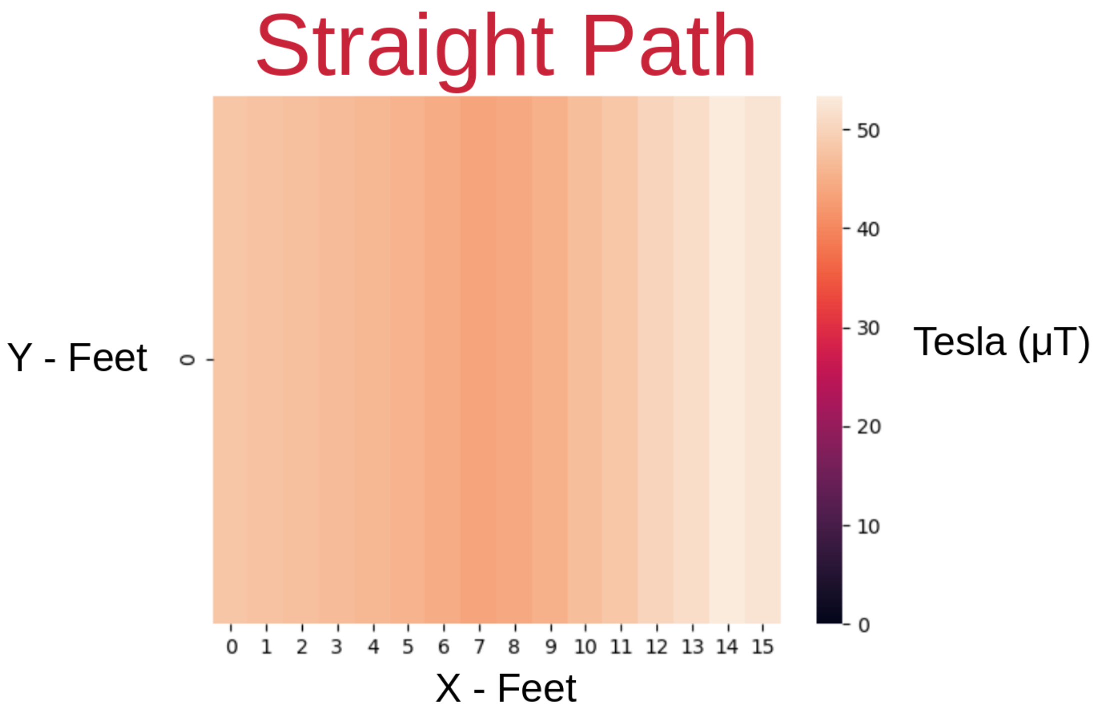
5.1 Experiments

Two experiments were conducted to evaluate the accuracy of Minet. Each experiment took place in a two bedroom apartment, representing two different scenarios: a simple straight path and a complex environment respectively. Markers were placed on the ground to simulate the desired environment, which then using the device, created a map of these markers. Each marker was placed exactly one foot apart with one reading taken at each. Each marker was assigned its matching coordinate in the generated map. The accuracy estimations were calculated using the percent error of a parameterized expected distance and the distance between the true coordinate of the marker and the actual value returned by the particle filter. The the mobile device used is a Motorola Moto g7 Optimo. The experiments were evaluated with a goal of 3 feet of accuracy, because meter level accuracy can be achieved with WiFi localization, and is about the best result of an indoor localization thus far.

5.1.1 Experiment 1 - Straight Path - Setup

In the first experiment 16 markers were placed on the floor in a straight line, with the intent to test a straight path, such as a hallway. With one reading per marker, mapping both directions in the path, produces 32 total fingerprints. The location of this map is against one wall and on its opposite side, a open area. This location is close enough to a wall to have some fingerprint distinctness, as the most difficult scenario for a particle filter is open space. Figure 5.1 shows the average magnetic magnitude (Tesla μT) at each step in the path. The average μT of the steps is 46.84, with a standard deviation of 2.61 μT. The minimum reading is 42.47 μT and the maximum reading is 53.42 μT.

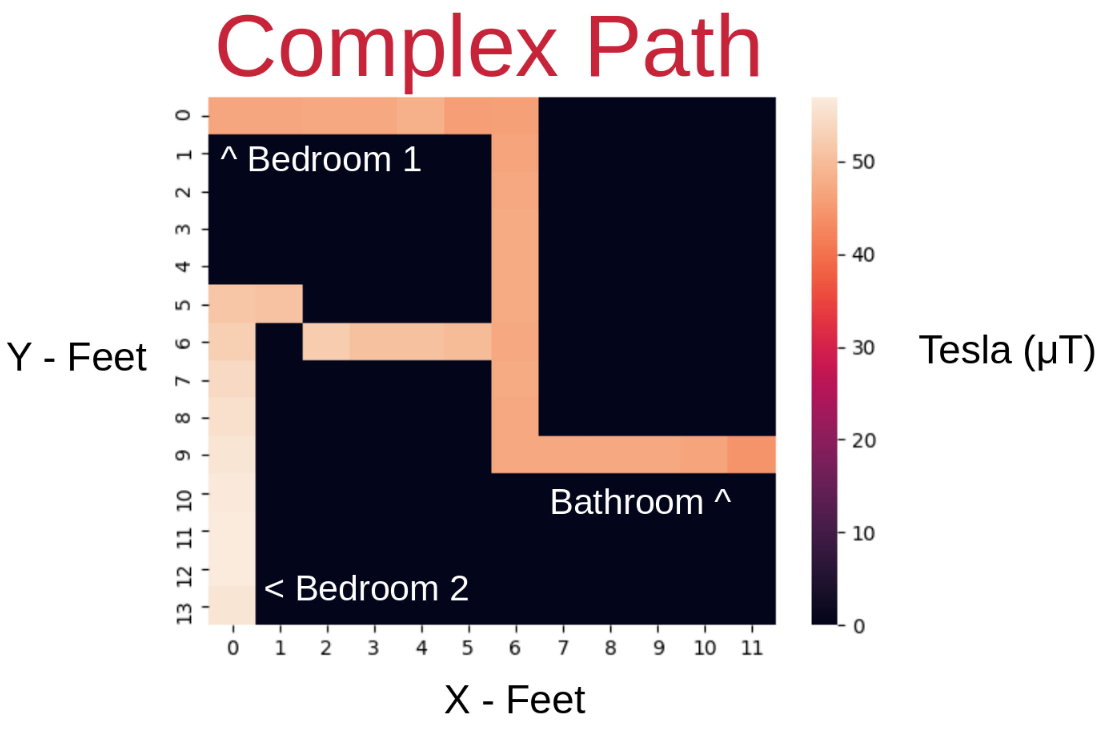
Figure 5.1: Experiment 1 - Magnetic Intensity Heat Map



5.1.2 Experiment 2 - Complex Environment - Setup

In this experiment, the tested map is of a complex environment spanning three rooms, with a short hall between them. This environment spanned 40 markers, with 1 foot between them. Backtracking over them, produced 80 total fingerprints. Figure 5.2 shows a heat map of a path between and in 3 rooms: two bedrooms, and a bathroom. The average μT is 49.81 with a standard deviation of 3.80 μT. The minimum reading is 44.37 μT, while the maximum readings is 58.03 μT. Bedroom 1 is at the top left of the figure; the second bedroom is bottom left, and the bathroom is at the far right. Each room was walked through in a straight line before reversing path, and returning in the opposite direction. Bedroom 1 was visited first, before heading down the hall and into the bathroom; the course was then reversed, and on returning to Bedroom 1, branched into Bedroom 2, before returning to the starting point in Bedroom 1.

Figure 5.2: Experiment 2 - Magnetic Intensity Heat Map



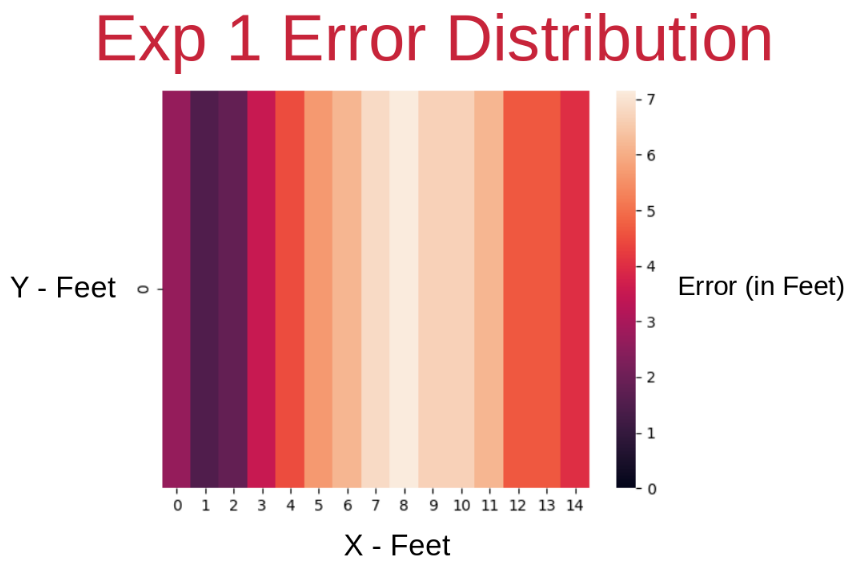
5.2 Results

5.2.1 General Findings

One of the first findings during testing, but also seen sparingly throughout development, was a problem of inconsistent sensor readings. This is a device wide problem, not specific to the Minet app. The problem is with the orientation calculation, which could either be an issue with the accelerometer data or the geomagnetic data, but it is likely an issue with the geomagnetic sensor. The device displays the incorrect compass orientation by as far as an entire cardinal direction. When the device is in such an error state, it also shows inflated geomagnetic readings, by as high as 10 times their average rate. The Moto g7, which while being a fairly recent phone, is inexpensive, and could explain the inconsistent sensors. The g7 is the only device available to test, so the extent of the problem is unknown. The only known fix for this problem is spinning the phone rapidly, in several directions, often with a toss in the air. The problem persists through multiple restarts of the phone, but rapid movement of the device causes some sort re-calibration.

5.2.2 Experiment 1 - Straight Path - Error

Figure 5.3: Experiment 1 - Error Distribution

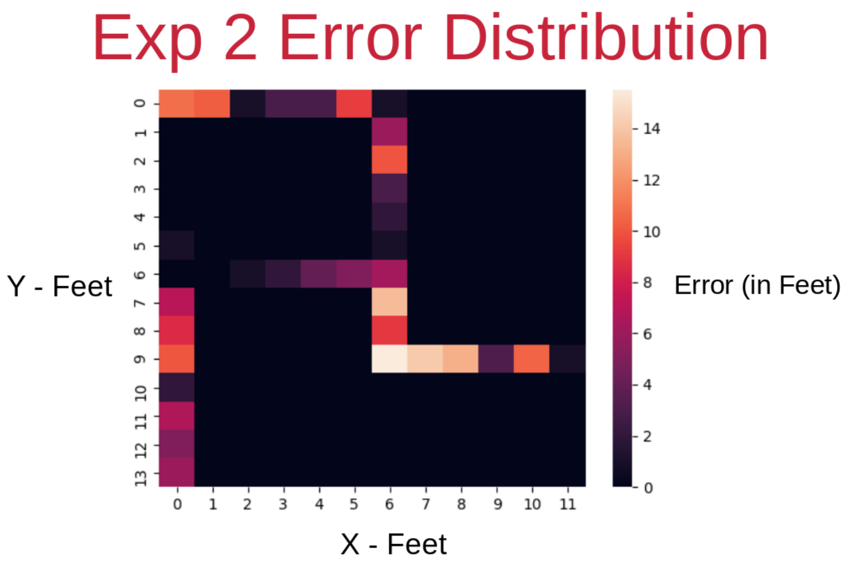


5.2.3 Experiment 1

First, in evaluating the fingerprint map, the distinction is low, producing a difficult to localize environment. While the standard deviation is 2.61 μT, that deviation is over the entire path, but many groups of 2 or 3 fingerprints vary by less than 1 μT. At 1 foot of distance, the particle filter showed an average of 63.63% error on a walk to the end and back of the path, calculated allowing 3 feet of error. On another run of this experiment, going 3 times to the end and 4 times back, resulting in 81.79% error. The percent error was calculated with an expected value of 3 feet, meaning that the error was greater than 3 feet 81.79% of the time, which shows an inability to be used for precise localization. The difference between these two experiments shows that over continued use, the particle filter does not reduce error, and in fact, may do the opposite. This is possibly because of Minet's limited state space, and the use of nearest neighbor to calculate the observable. While designed to reduce the amount of possibilities the filter would have to contend with, snapping to landmarks causes the filter to get stuck in certain pockets of near-identical data such as steps 0 through 5. The filter gained accuracy as it approached the two ends of the path, getting within 3 feet of the landmarks at the ends, and then affixed to a value, before jumping to a more accurate one towards the end of the path. The middle stretch of the path accumulated the most error, with 94.13% error, while the edges had a lower, but still significantly high, value of 61.11% error. While this particle filter, performs worse than some of those in other studies, it shows consistency in that error in open space is significantly higher than the error approaching a source of ferromagnetic material, such as the walls at the ends of the path. Looking at the 81.7% error, in the second run of the experiment, 58.83% error came from the middle steps, while only 22.91% error came from the edges, which supports that a more confined space could provide a better result. The average error was 4.42 feet, or about 1.35 meters, which would be on par with some other experiments, but in the context of this experiment, the estimation is only coming from a space within 16 feet, the filter could only narrow down the location to around 25% of the total area. In conclusion this experiment has shown that Minet has more space to be improved to achieve high accuracy to perform indoor localization.z

5.2.4 Experiment 2 - Complex Environment - Error

Figure 5.4: Experiment 2 - Error Distribution



5.2.5 Experiment 2

The fingerprint map, also showed fairly low distinction, especially in the hall area between Bedroom 1 and the bathroom. The percent error had to be calculated over 3 feet, because most of the readings were over even 4 feet, with 91.5% error. This experiment revealed an issue with the particle filter and outliers. When the particle filter returns an incorrect guess, which is the coordinate with the most particles clustered around it, the distance can be astronomically high, relative to what should be possible. One of the likely causes of theses outliers, is that pockets of similar magnetic intensity are not being properly disposed, due to the motion model and filtering. The outliers in this experiment were as high as 17 feet. Without these outliers of greater than 12 feet, the percent error is down to 68% error, expecting 4 feet. However, the average percent error with an expected value of 3 feet and no outliers is 95%. The average error in this experiment was 5.71 feet; without any outliers, the average error is 4.55 feet. While these averages are decent given the size of the map, the error is inconsistent, with some areas of the map working properly, before the estimation jumps to another location, providing high error, before correcting. With quite a high number of readings that had a distance greater than 12 feet, Minet is not stable enough to provide reliable navigation. One potential solution for limiting these errors could potentially be filtered out by not allowing a hypothesis to be too far from the previous one. The reason the error can vary so highly is impart because of the similarity of fingerprints, and also the Minet particle filter algorithm not effectively handling movement. The particle filter should not be able to jump across the map, and solutions will be discussed in the Conclusion and Future section.

5.2.6 Comparison of Experiments

The results of the two experiments reveal similar conclusions. The key difference is in the scale of the error. In the first experiment, when a hypothesis was incorrect, the distance of the error was relatively small, as all guesses were a short distance from each other. However, in the second experiment, a wrong guess could be as far as 17 feet, whereas in the first experiment the maximum was 12 feet. These outliers show a flaw in the particle filter's ability to distinguish similar fingerprints across the map. When both experiments had the same maximum error of 12 feet, they had similar average errors of 4.42 and 4.55, which shows that aside from the occasional wrong selection, the filter can consistently place a device within 1.5 meters. This was achieved with a minimal amount of readings, and could potentially be improved with a few redesigns discussed in the future section.

5.3 Conclusion

The results show moderate accuracy, but with poor precision. While localization within 1.5 meters, is not bad, the inconsistency of the error, causes Minet to be unreliable for navigation. The data collection, mapping, and networking aspects of Minet work as intended, and the results show potential for an improved version of the system. The accuracy of the system came from a single reading per marker, which can be pushed to further utilize the strengths of the particle filter.The particle filter could have better implemented the motion model to enforce to eliminate the 'teleportation' problem. Lastly, the state space could have been implemented in a more efficient way that not only would make the motion model improvements easier, but also more precisely follow the intent of the design.

CHAPTER 6

POTENTIAL IMPROVEMENTS

6.1 Potential Improvements for Current System

6.1.1 State Space Improvements

One possible exploration to improve the state space implementation is to replace the array-based model with a graph data structure. The graph could enforce that the nodes can only move to the next node in the correct direction, and if that node does not exist, then it the weight would be lowered for that node. This implementation better captures the main idea of limiting the state-space to allow, a more accurate approximation by only allowing moves that are possible. However, this will not be able to handle noise as well as the current implementation. In this implementation (x, y) have noise applied to them, which gives the particle filter, a closer approximation of real movement, and also ensures that not all particles are the same. However, when particles shift towards a coordinate that is not possible to access on the current path, it can possibly snap to it by the nearest neighbor approximation. This point may geographically be the closest on the map, but it is only coincidental, or possibly blocked by a barrier. A potential hybrid implementation can possibly solve both of these problems, by keeping a grid approach, but also giving particles the knowledge of the next node on the path. The idea of a graph was considered during development, but not tested, due to time constraints. This will be a priority in future development of Minet.

6.1.2 Particle Filter Improvements

The filter returns the most popular coordinate, that with the highest number of particles This is the feature that allows the 'teleportation' issue. An alternative would be to use the mean internal state of the particle filter, which would be an averaged guess, which could then calculate the nearest neighbor to stay on the path. This however, could provide less predictable results, as the mean state could be in the center of a zone with no fingerprints. This adjustment could provide more stability, but may lose accuracy. In addition, this idea could be incompatible with the state space model proposed.

6.1.3 Throughput Improvements

Minet's throughput should be increased to feed the particle filter more data. One of the foundational ideas of Minet was to take computation to a server where virtually unlimited computational complexity is available. However, in this thesis, this idea was not tested. The readings were limited, to not overload the particle filter, as live readings came in too quickly during development. The reason a timer based system was not implemented was due to unfamiliarity of handling sensor readings in Android. An early attempt at this was to change the sensor speed, but the readings would still output just as quickly but without updated values. However after placing the sensor readings into a fragment, the data can be easily accessed through an interface, then in the Navigation activity, a timer could be implemented that calls for the current reading in the interface on a specified interval. After making this change, a future experiment would be to load test the capacity of the filter. Another would be to verify that more readings improve the accuracy. In theory, more readings should produce better results, so this is a worthy investigation.

6.1.4 Flaws in Experimentation

One flaw in the tests comes from the device's single reading per marking. Since the first reading is completely random, it produced an 11 foot error; the next step was the same, before jumping to the correct value with a 1 foot error. These 3 readings could have been oriented on the first marker, which would have given the test a slightly more accurate result, without inflated error. Another flaw is limited scope of tests; more test runs should have been conducted to get a larger sample of data.

6.2 Further Improvements

6.2.1 Improvements in Map Making

The magnetic map of Minet, while able to quickly and dynamically create a path-based magnetic map, is not as robust as other options out there. Each step in the path only contains one reading, which makes it a limited representation of the space. Soft Computing techniques can create a more featured map [1]. MagicFinger also uses geomagnetic data exclusively, but does much more with the map-making process. Data is pre-processed: transformed into features, filtered, normalized, then clustered and classified [1]. With these steps, the map contains distinct landmarks which can then be used to localized relative areas. Using these as anchors, the map has a better source of truth. These techniques could potentially be applied to Minet, as a separate layer to the path, where the landmarks can be reference to help distinguish one part of the path from another.

6.2.2 Multi-Modal Data

As previously mentioned, Minet was designed to be extensible. Since the computation is on a server, the particle filter has virtually unlimited access to computing power, which can allow particle filter to work with larger data than presented in this thesis. Data could be expanded to include other sensor data from smartphones, such as light [3] or potentially WiFi signals. Particle Filters experience Drift, so the more “sources of truth” there are, the more the filter can correct itself.

6.2.3 Deep Learning Model

Another potential improvement is to use a deep learning model. Long Short-Term Memory or LSTM is a Recurrent Neural Network (RNN), which specializes in time series of data. Since localization can be considered as a series of locations based on time, this is a proper tool. Two major benefits of this system, are in storage and speed. Although training for LSTM can be time-consuming [8], the actual usage of the trained model is quick. In addition, the only storage needed is a trained set of weights, rather than a map of fingerprints for each location. LSTM models can also work with multi-modal data, allowing the use of other sensors to make more distinct landmarks. One study that combined light and geomagnetic data [8] found that their system performed with under 0.5 meters of error 58% of the time, and under 2 meters 82% of the time, so the system is fairly robust, as tested in a complex environment.

6.2.4 Particle Filter LSTM Model

While LSTM models are deterministic, particle filters are probabilistic. As mentioned background section, a probabilistic technique is ideal when all the variables in an environment cannot be realistically calculated. In the case of localization, especially magnetic localization, every reading for every orientation is unrealistic to be calculated considering the noise and random disturbances in any given location. The advantages of probabilistic programming have been explored with deep learning models in a limited capacity. An LSTM embedded particle filter has been used to predict pedestrian walking paths and found that their predictor could produce plausible hypotheses on paths [3]. Indoor Positioning has been attempted with both the usage of LSTM and a particle filter [4]; however, no indoor localization system yet exists using this combination at the time of this research. Combining the advantages of both of these techniques could provide a performance breakthrough in indoor localization.

CHAPTER 7

CONCLUSION AND FUTURE WORK

Indoor localization is an important and complex problem. Minet explores a magnetic based solution that uses a modified particle filter and a distributed model. The novel technique of state space limitation shows potential in improving magnetic indoor localization. The implementation in Minet was not perfect but showed promise in the idea. With some improvements to the current system's implementation, better results could potentially be achieved.

There are also potential expansions for Minet that have not been tested before. Deep learning models have shown strong results. Combining a deep learning model with a particle filter for the problem of indoor localization has not been tested but could provide an even better solution than Minet initially proposed.

Bibliography

[1] Daniel Carrillo et al. “Magicfinger: 3d magnetic fingerprints for indoor location”. In: *Sensors* 15.7 (2015), pp. 17168–17194.

[2] A. Depari et al. “Indoor Localization for Evacuation Management in Emergency Scenarios”. In: *2018 Workshop on Metrology for Industry 4.0 and IoT*. 2018, pp. 146–150.

[3] Ronny Hug et al. “Particle-based pedestrian path prediction using LSTM-MDL models”. In: (Apr. 2018).

[4] Ghulam Hussain et al. “Indoor positioning system: A new approach based on lstm and two stage activity classification”. In: *Electronics* 8.4 (2019), p. 375.

[5] Zhao Jin et al. “Development of Indoor Localization System for Elderly Care Based on Device-Free Passive Method”. In: Aug. 2015, pp. 328–331. DOI: `10.1109/ISDEA.2015.88`.

[6] Binghao Li et al. “How feasible is the use of magnetic field alone for indoor positioning?” In: *2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE. 2012, pp. 1–9.

[7] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. Vol. 1. MIT press Cambridge, 2000.

[8] X. Wang, Z. Yu, and S. Mao. “DeepML: Deep LSTM for Indoor Localization with Smartphone Magnetic and Light Sensors”. In: *2018 IEEE International Conference on Communications (ICC)*. 2018, pp. 1–6.

[9] Michael G Wing, Aaron Eklund, and Loren D Kellogg. “Consumer-grade global positioning system (GPS) accuracy and reliability”. In: *Journal of forestry* 103.4 (2005), pp. 169–173.

[10] H. Xie et al. “A Reliability-Augmented Particle Filter for Magnetic Fingerprinting Based Indoor Localization on Smartphone”. In: *IEEE Transactions on Mobile Computing* 15.8 (2016), pp. 1877–1892.

[11] Hongwei Xie et al. “MaLoc: A practical magnetic fingerprinting approach to indoor localization using smartphones”. In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. 2014, pp. 243–253.