Indoor Navigation Final Report Submission

by Aman Shaikh

Submission date: 25-May-2018 04:49PM (UTC+0400)
Submission ID: 968211155
File name: Final_Submission.pdf (1.49M)
Word count: 7573
Character count: 44335

A REPORT

ON

INDOOR NAVIGATION USING WIFI FINGERPRINTING

Prepared in Fulfilment of the requirements of Design Project

217180166 - Indoor Navigation - CS F376

BY

AMAN SHAIKH
2015A7PS0039U
CS

Under the supervision of

NAND KUMAR

Professor

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

DUBAI CAMPUS, DUBAI UAE

JANUARY to MAY - 2018

ACKNOWLEDGEMENTS

Firstly, I would like to express my heartfelt gratitude to my Supervisor Dr Nand Kumar, Professor, Computer Science Department, BITS Pilani, Dubai campus, United Arab Emirates, for his valuable guidance and encouragement during the course of this project. I am extremely grateful to his for his able guidance, valuable technical unpts and useful suggestions. Secondly I would like to thank Prof. R. N. Saha, Director of BPDC who has given us an opportunity to apply and understand our engineering concepts in a practical atmosphere in form of this project.

I would like to express my deepest appreciation to all those who have helped me to complete this Project. A special gratitude I give to my friends, whose contribution in stimulating suggestions and encouragement, helped me to coordinate my project especially in such short time and enormous work.

Furthermore I would also like to acknowledge with much appreciation the crucial role of the IT staff of BITS Pilani Dubai Campus, who gave the permission to use all required equipment and the necessary material to complete the project. A special thanks goes to my friend, Akansha Das, who helped me to assemble the data and gave valuable suggestion about the project. I have to appreciate the guidance given by other supervisor as well as the staff especially in the project and presentation.

I would like to thank Prof. Santosh Kumar, Dean – Computer Science for giving me this opportunity to apply my knowledge in the technical field and gain first-hand experience.

SIGNATURE OF THE STUDENT

DATE: 24-MAY-2018

LIST OF FIGURES

LIST OF ABBREVIATIONS

(SHOULD BE IN ALPHABETICAL ORDER)

❖ AP	-	Access point
❖ API	-	Application programming interface
❖ AOA	-	Angle of arrival
❖ BLE	-	Bluetooth Low Energy
❖ BVI	-	Blind and visually impaired
❖ CDF	-	Cumulative distribution function
❖ PDF	-	Probability density function
❖ PDR	-	Pedestrian dead reckoning
❖ RSSI	-	Received-signal-strength-indicator
❖ UI	-	User interface
❖ TOA	-	Time of arrival
❖ TDOA	-	Time difference of arrival
❖ WHO	-	World Health Organization
❖ KNN	-	K Nearest Neighbour

CONTENTS

Bibliography

www.arxiv.org

www.people.kth.se

www.docplayer.net

www.diva-portal.org

www.mdpi.com

1. OVERVIEW OF INDOOR NAVIGATION

1.1. INTRODUCTION

It is often a daunting task to navigate within a large complex structure even while having a map at your disposal. With the boom of augmented reality and Internet of things it is a great utility to be able to know your location within such complex structures. For example Navigation within shopping malls, knowing which shop has special offers or even games like scavenger hunt where you need to visit certain check points to avail special offers; large campuses, an indoor navigation system will help new students or visitors to navigate within the campus; even airports. Some other good use of an indoor positioning system will be to help aid navigation of artificially intelligent drones. User and device localization also have wide-scale applications in health sector, industry, and disaster management, building management, surveillance and a number of various other sectors. It can also benefit many novel systems such as Internet of Things (IoT), smart architectures (such as smart cities, smart building, smart grids) and Machine Type Communication.

While Global Positioning System (GPS) and other similar global navigation satellite systems (GNSS) provided good quality for outdoor positioning, robust indoor positioning is still an open problem. The GPS and similar localization networks do not work indoors as they need direct Line-of-Sight (LOS) between the satellites and user which is not a case indoors. Because of the early development of Wi-Fi networks, Wi-Fi access points can be seen everywhere in indoor environments and almost all mobile devices have a built-in Wi-Fi receiving module. As a result, Wi-Fi indoor positioning has become an attractive research topic in developing indoor positioning.

An indoor positioning System can be developed to achieve any of the following reasons:

• Device based localization (DBL): The process in which the user device uses some Reference Nodes (RN) or anchor nodes to obtain its relative location. DBL is primarily used for navigation, where the user needs assistance in navigating around any space.

• Monitor based localization (MBL): The process in which a set of anchor nodes or RNs passively obtains the position of the user or entity connected to the reference node. MBL is primarily used for tracking the user and then accordingly providing different services.

• Proximity Detection: The process of estimating the distance between a user and a Point of Interest (PoI).

1.2. LOCALIZATION TECHNIQUES

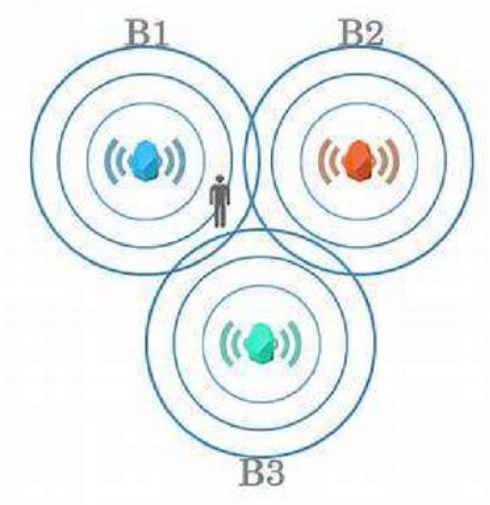
The main categories of this classification by measurement techniques are based on the measurement of distance, angle, fingerprint RF-localization pattern and fingerprint or any combination of these categories.

In indoor environments, the mobile station is surrounded for objects which can distort the signal reception. Moreover, the distance between transmitter and receiver is usually shorter than the time resolution that can be measured by the system.

Hence approaches AOA (angle of arrival) and TDOA (time difference of arrival) are impractical for indoor environments. Subsequently fingerprinting technique has gained importance due to its comparison with those of the first two indoor positioning simpler systems which are more expensive and needs and specialized AP hardware.

1.3. MAIN METHODS TO LOCATE BASED ON WI-FI SYSTEMS

1.3.1. Triangulation of signal Power



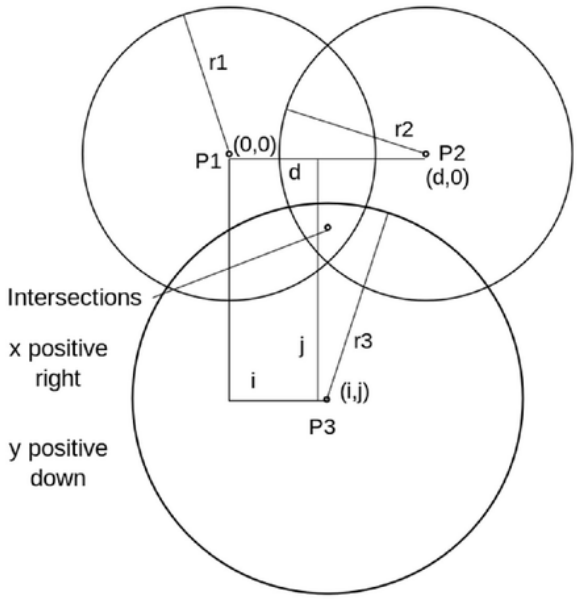
Triangulation is a term originally used in the navigation circles to obtain multiple reference point and locate an unknown position. The triangulation algorithm helps to estimate the location of the target place based on geometric properties of triangles. When the mobile device at the target place receives the Wi-Fi signals from one or more Wi-Fi access points, the time of arrival (TOA), the angle of arrival (AOA) and the received signal strength (RSS) of Wi-Fi signals will be used to calculate the distances between the target place and Wi-Fi access points. With the locations of three or more Wi-Fi access points, the target place can be estimated by triangulation.

The triangulation is based on the following phases:

- With the powers of three access points that receive the user, is created an equations system, representing three circles.
- The system is solved to obtain a set of points, called triangulation points. - Each triangulation point is considered a vertex of a triangle.
- All possible triangles are formed and the areas are calculated to compare them.

The centre of the triangle with the smallest area is taken as the estimate location of the user. However this method isn't widely used due to poor accuracy.

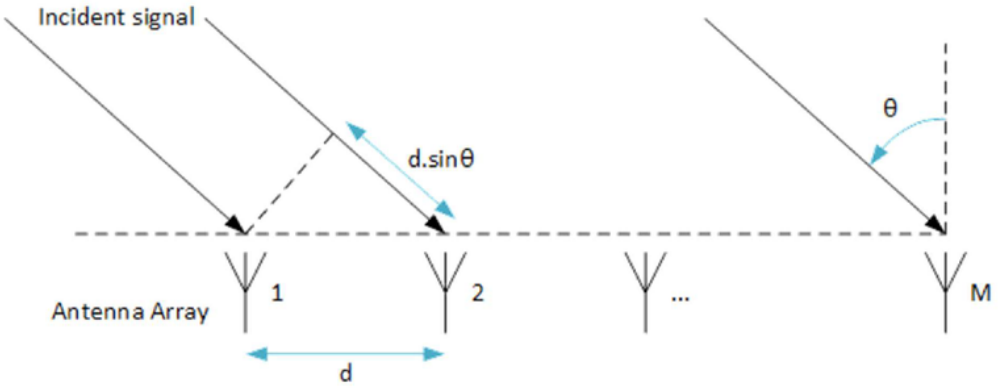
1.3.2. Trilateration of signal Power



This approach is very similar to Triangulation but it does not involve use of the time of arrival (TOA), the angle of arrival (AOA). Trilateration determines the user location using pure geometry. First, RSSI values from several beacons are measured. Each RSSI value is translated into a distance using a propagation model. A propagation model is an equation that describes the relationship between distance and RSSI.A standard indoor path loss function which requires knowledge of a number of variables in order to determine the path loss. However it faces many challenges like:

- Uncertainty: Due to factors such as multipath fading, reflection, the three circles used in trilateration often do no intersect at a common point. Thus there is no single point which meets all the distance constraints.
- No consistency: when there are more than 3 available beacons each subgroup three beacons is able to give a trilateration result. These result often vary and it is difficult to determine which result is the most accurate.
- Propagation model error: A propagation model is need to describe a relation between the RSSI value and the distance. The propagation model needs to be specifically curated based on the propagation environment.

1.3.3. Angle of Arrival (AoA)

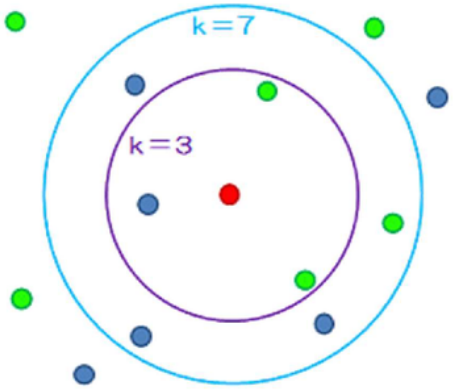


Angle of Arrival (AoA) based approaches use antennae arrays (at the receiver side) to estimate the angle at which the transmitted signal impinges on the receiver by exploiting and calculating the time difference of arrival at individual elements of the antennae array. The main advantage of AoA is that the device/user location can be estimated with as low as two monitors in a 2D environment, or three monitors in a 3D environment respectively. Although AoA can provide accurate estimation when the transmitter-receiver distance is small, it requires more complex hardware and careful calibration compared to RSS techniques, while its accuracy deteriorates with increase in the transmitter-receiver distance where a slight error in the angle of arrival calculation is translated into a huge error in the actual location estimation. Moreover, due to multipath effects in indoor environments the AoA in terms of line of sight (LOS) is often hard to obtain.

1.3.4. Time of Flight (ToF)

Time of Flight (ToF) or Time of Arrival (ToA) exploits the signal propagation time to calculate the distance between the transmitter Tx and the receiver Rx. The ToF value multiplied by the speed of light c = 3×108 m/sec provides the physical distance between Tx and Rx. Basic geometry can be used to calculate the location of the device with respect to the access points. ToF requires strict synchronization between transmitters and receivers and, in many cases, timestamps to be transmitted with the signal (depending on the underlying communication protocol). The key factors that affect ToF estimation accuracy are the signal bandwidth and the sampling rate. Low sampling rate (in time) reduces the ToF resolution since the signal may arrive between the sampled intervals. Frequency domain super resolution techniques are commonly used to obtain the ToF with high resolution from the channel frequency response. In multipath indoor environments, the larger the bandwidth, the higher the resolution of ToF estimation. Although large bandwidth and super-resolution techniques can improve the performance of ToF, still they cannot eliminate significant localization errors when the direct line of sight path between the transmitter and receiver is not available.

1.3.5. K-nearest neighbour's algorithm.



The location model using Fingerprinting is based on finding the power of received signal for each access point in a certain location. Those powers will be treated as the metric distance, and the k-nearest neighbour's algorithm will be applied. This algorithm searches the database, created in the Fingerprinting process, and selects the points which fits the best compared to the currently received power signals. To select the best fitting points, the smallest Euclidean distance of powers is used, using the powers as metric distances. Selecting the k (where k means the number of points) points with the smallest distance to the point where the user is, the barycentre is calculated, and this result is the estimated location of the user.

As a clarification, if the point where the calculation of the distance is being done does not contain any value of power for one or more of the access points that have been located in our fingerprinting database, a maximum distance will be returned, so when choosing the minimums, that distance will be large, thus ignored, because that point can lead to bad estimations.

Moreover, it has to be clarified that, during the fingerprinting process, when taking power values in one point, several measurements were taken in the same point, and then averaged, before storing the result in the fingerprinting database. This makes the algorithm more efficient. This algorithm is simple enough for being efficient on any machine.

1.3.6. RSSI fingerprint method

The Fingerprinting location model is based on the power of the received signal of the different access points on a certain position, and can then use those values in a series of algorithms that allows it to calculate the user's position. It is called Fingerprinting, because the main part of the method is to create a knowledge base with the power data taken in different environments where the system will be used. A matrix is created with the received power of each access point in a series of known locations.

It is therefore one of the simplest methods to implement indoor positioning match the RSSI values of signals currently being received from a set of Reference points with a list of locations with tuples of signal identifiers and RSSI values.

We consider the set of tuples of Reference point (BSSID) and RSSI values to be a fingerprint for a location. The assumptions are that such a fingerprint characterizes a single location, that fingerprints are stable over time, that the beacons always transmit at the same emitted power, and that the difference in RSSI values measured at a given location by two different devices have only small differences (less than the differences between the RSSI values at different locations).

Mechanism:

Technically, a fingerprint is a vector of RSSI values observed at a location. The AP's signal information is collected when training the devices are stored in a database (vector where each cell saves the power that receive the user from each AP to its position). This method has consisted of three phases: two offline phase and the online phase.

- In the first phase the fingerprints are collected at certain places to build a radio map, we store the AP's information like signal power, bit rate, coverage.
- The second is the training phase, where the database is constructed with the device, so that the powers are kept each AP for each map position. Different location estimation methods use different characteristics of fingerprints, such as their mean and variance is used to often reduce the dataset size.
- In the third step, is obtained for each position a signal power vector, received from each AP. This vector is compared with the database and that, estimates the position where the signal power are closer. The location of the reference point with the “closest fingerprint” (according to some norm) is reported to be the estimated location. The norms to define “closest fingerprint” can be deterministic or probabilistic. These two types of norms are discussed in the following paragraphs.

There are a number of deterministic methods. However, here we introduce only the method utilized in this project. This method was chosen due to its simplicity and popularity. At each reference point, the mean of RSSI values for each access point (AP) is calculated, and then this averaged fingerprint is used to represent the reference point. Next, the Euclidean distances between the current fingerprint and averaged fingerprints of all the reference points are calculated.

The reference point whose fingerprint has the smallest Euclidean distance is the best estimate of the user’s location. To help deal with the problems of individual fingerprints, the K reference points with the smallest Euclidean distances are determined.

To estimate the actual position of the user, a weighted average of the locations represented by these K nearest reference points is used. This process is the so-called the "K-Nearest-Neighbour" algorithm.

Filters

At this point, the problem of estimating of the position is analysed. For this problem it is necessary to define a model that incorporates mobile movement restriction which positions are more feasible given the position in the previous instant. To solve the problem with the fluctuation of the signal, and be sure that the user tracking is correct and with a good precision, some kind of filter can be used:

- Bayesian filter.
- Kalman filter.

1.4. HEURISTICS

Heuristic methods may be used by themselves or as improvement to two other methods, discarding positions of the database to reduce the span of possible locations.

1.4.1. Proximity Heuristics

The proximity algorithm helps to estimate the location of the target place using the proximity relationship between the target place and Wi-Fi access points. When the mobile device at the target place receives Wi-Fi signals from different Wi-Fi access points, the location of Wi-Fi access point with the strongest signal will be regarded as the location of the target place. The accuracy of this algorithm is determined by the distribution density and signal range of Wi-Fi access points. This method is based on the nearest AP to the terminal to determine its position. Depending on to the power received for the user from each AP, the minimum power values are discarded and it is said that the greater power signal received corresponds to the nearest AP. To use this method, the client must be in the position of this access point (the nearest AP).

1.4.2. K-nearest neighbours

The method of the k-nearest neighbours is part of a family of learning techniques known as learning-by-example (instance-based learning). Learning in these algorithms consist simply on memorize the training examples presented. When a new position is presented to the learning system, a set of similar examples is retrieved from the memory for classify the new position. Therefore, an approximate location of user can be calculated. The drawback of this technique is that it requires a large number of learning points in order to make good estimations.

1.4.3. Heuristic Motion

The movement is an important part of the context of a user in a positioning system. It is possible to classify a user as stationary or moving based on the strength of the Wi-Fi signal. Taking into account that the signal strength of the AP's create more power peaks around the estimated position when the device is in motion. The device has to be trained when moving around an area as well as when it is stationary at certain point.

1.4.4. Bayes theory

Is a probabilistic technique that creates a probability distribution of all the possible positions? Probabilistic techniques achieve a higher precision than deterministic techniques in exchange of bigger computational cost. The Bayesian approach is typically applied in cases where the representation of the environment is in the form of grids. Another alternative for pattern the environment is to use a topological map. In this case the location is based on the fact that the device, identifies automatically that has reached a map node, based on some geometric information of the environment.

1.4.5. Neural Networks

This technology can build systems capable to learn, adapt to varying conditions, even if it has a sufficient collection of data to predict the future state of some models. Training in this case is used to accelerate learning. It makes memorizing characteristics of the points of interest and thus the device recognizes the location area in real-time.

1.4.6. Support Vector Machine (SVM)

Support vector machine is an attractive approach for classifying data as well as regression. SVM is primarily used for machine learning (ML) and statistical analysis and has high accuracy. As highlighted by [16], SVM can also be used for localization using offline and online RSSI measurements. While the RSS based approach is simple and cost efficient, it suffers from poor localization accuracy (especially in nonline-of-sight conditions) due to additional signal attenuation resulting from transmission through walls and other big obstacles and severe RSS fluctuation due to multipath fading and indoor noise. While different filters or averaging mechanisms can be used to mitigate these effects, it is unlikely to obtain high localization accuracy without the use of complex algorithms.

1.5. EVALUATION AND CONCLUSIONS OF THE MAIN METHODS FOR POSITION CALCULATION

The technique of triangulation Power, has no training phase of the devices, this method goes directly from the phase of AP's knowledge to the phase of estimate the user position. Heuristic techniques are used to improve accuracy in systems using methods such as triangulation of power, in which the precision can be increased applying heuristic techniques.

Fingerprinting is a widely used method due to its relatively high accuracy and reliability. Compared to triangulation, where an explicit signal propagation model is required, the fingerprinting method implicitly stores the characteristics of the signal in the radio map, thus complex radio propagation modelling is avoided. However, a severe problem with fingerprinting is its high cost. In the offline phase, much labour and time are required to build the fingerprint database. Moreover, the database has to be updated anytime the environment changes in order to maintain accuracy. To diminish this cost, one solution is to develop efficient and reliable interpolation and approximation methods, so that only a few fingerprints need to be collected and additional fingerprints can be predicted (typically by interpolation). Another solution is to implement an organic system that leaves the fingerprint collection to users.

In the online phase, matching the observed current fingerprint to a large fingerprint database requires a considerable amount of computation. Sanchez et al. designed a system which uses trilateration to determine a starting point for the search and then applies the fingerprinting to reference points close to this starting point.

Besides cost, the accuracy of fingerprinting suffers when there are numerous points (in the training data) with similar fingerprints. This can happen due to factors such as multipath.

2. DATABASE

2.1. FINGERPRINTING DATABASE

BSSID - 1	RSSI -1	BSSID - 2	RSSI - 2	BSSID - 3	RSSI - 3	LABEL
d4:6d:50:ed:ca:81	-56	d4:6d:50:e8:09:e1	-62	a0:ec:f9:65:b7:f1	-69	2v
d4:6d:50:ed:ca:81	-60	d4:6d:50:e8:09:e1	-62	a0:ec:f9:65:b7:f1	-73	2v
d4:6d:50:ed:ca:81	-63	d4:6d:50:e8:09:e1	-63	a0:ec:f9:65:b7:f1	-75	2v
d4:6d:50:ed:ca:81	-58	d4:6d:50:e8:09:e1	-59	a0:ec:f9:65:b7:f1	-74	2v
d4:6d:50:ed:ca:81	-59	d4:6d:50:e8:09:e1	-45	a0:ec:f9:65:b7:f1	-81	2v
d4:6d:50:ed:ca:81	-58	d4:6d:50:e8:09:e1	-50	a0:ec:f9:65:b7:f1	-75	2u
d4:6d:50:ed:ca:81	-63	d4:6d:50:e8:09:e1	-53	a0:ec:f9:65:b7:f1	-81	2u
d4:6d:50:ed:ca:81	-58	d4:6d:50:e8:09:e1	-48	a0:ec:f9:65:b7:f1	-83	2u
d4:6d:50:ed:ca:81	-42	d4:6d:50:e8:09:e1	-67	a0:ec:f9:65:b7:f1	-85	2u

The Soul of any application that uses Data Mining techniques and Machine Learning is the database. So the first step to creating the database is identifying the terms that go into the database. We decide to choose the number 3 as the number of Access Points to take into consideration. Meaning the app records data only if the app detects there are 3 Bits-Students Access Points in close proximity. The app creates an entry which consist of the BSSID or the MAC ID of the Access Points along with the received Signal Strengths of each of the Access Points. So this creates an offline database on the user's phone. Only the AP information of each fingerprint is reserved. Thus, we can avoid the disadvantages of traditional methods that the fingerprint database needs to sample all the APs.

A Simple fingerprint matching algorithm is put forward based on important Access Points, which effectively narrows down the scope of the database retrieval. Therefore, it can reduce the computation and eliminate the interference factors to obtain a higher positioning accuracy.

The algorithm efficiently reduces the complexity of the computations required to perform KNN by the applying a technique called clustering. Meaning we cluster the Access Points based on Floors and thereby Segments. A floor is divided into 4 clusters namely the left atrium, the right atrium, the seating area segment and the professors segment for each floor. Hence on execution of the primary algorithm we classify the users location based on the received BSSID and classify the user to one of the many clusters and then the KNN algorithm is applied to records of only that segment so a Database of 10,000 entries the KNN is applied to only 10,000/ number of clusters. The corresponding fingerprint in the database is extracted accurately so that the accuracy of mobile positioning is improved.

2.2. ANALYZING THE EFFECT OF SSID CHOSEN

While coming up with the primary algorithm to identify the user location a certain aspect is often ignored, which is the Access Points which as filtered to be entered in the database. So we come up with an idea named "Wi-Fi fingerprint-based localization based on the important access points." Among the fingerprint data collection, the APs' RSS values are collected and sorted in descending order. One or several top ranked APs are considered as important access points (IAP) of Wi-Fi fingerprint. In the procedure of fingerprint matching positioning, in order to reduce computation and positioning time, some fingerprints are screened when they are the same as the fingerprint IAP in the fingerprint database. Thus it was noticed that Access Points other than that chosen showed better results but due to accessibility and availability BITS-Students was chosen. The important access point is one or several Wi-Fi access points which have the strongest signal strength. The obtained Wi-Fi fingerprints which comprise a plurality of signal strength information of the Wi-Fi APs are considered as the important features of the fingerprint.

In indoor environments, various Wi-Fi access points are dispersed and usually arranged in different rooms. The detected signal intensity is seriously influenced by the number of walls where Wi-Fi signal passes through. Therefore, signal strengths detected in different rooms from the same Wi-Fi AP are different. Simultaneously, Wi-Fi APs with the strongest signal strength detected from different rooms at the same time are deferent.

2.3. ANALYZING THE EFFECT OF NUMBER OF ENTRIES PER NODE

It is often considered that more the entries in the database the better results any Data Mining technique gives. But this is not necessarily true as we can also has cases where the same inputs give multiple outputs which reduces the accuracy and moreover in case of applying K nearest algorithm we might have biased datasets as there may be certain nodes which have more entries than others and as a result they show up more frequently while applying the KNN algorithm.

2.4. ALGORITHM INTERACTION WITH THE DATABASE

The algorithm implemented in the prototype is naïve but an enhanced version of KNN. The primary algorithm uses many subtle but effective techniques to cut down computations. Firstly When the Identification process begins the app gathers the user's current Wi-Fi fingerprint then this fingerprint is associated with a cluster of entries then KNN is only applied to those entries.

For example the second floor is to be considered the second floor is divided into 4 clusters namely the left atrium, the right atrium, the professor chambers atrium and the seating area's atrium. Let's assume the total number of entries are 1000 for simplicity then the checking the users Wi-Fi fingerprint we associate the user to second floor which in itself reduces the complexity further his location is associated to more specific clusters in the second floor. Hence the total database of 1000 entries are reduced to about 1000/4 which is 250 entries we can assume the number 200 for simplicity as we know most of the entries are recorded for the ground floor. The 200 entries are further reduced to only about a 50 as 200/4 as we have 4 clusters. As we can see the computations are extremely minimal compared to an extensive all out KNN.

Furthermore the 50 entries are further reduced to specific segments by checking only those entries in the database which match the users WIFI fingerprint MAC ID this will further reduce the computations to 50/4 as on average we have 4 segments per cluster. These segments also only have about a few rooms so the accuracy is practically reduced to a few rooms which corresponds to approximate 5m.

3. IDENTIFICATION

3.1. IDENTIFICATION OF USER LOCATION

The application's first function is to identify the user's location which is done by the algorithm implemented in the prototype, it is naïve but an enhanced version of KNN. The primary algorithm uses many subtle but effective techniques to cut down computations. Firstly When the Identification process begins the app gathers the user's current Wi-Fi fingerprint then this fingerprint is associated with a cluster of entries then KNN is only applied to those entries. As we can see the computations are extremely minimal compared to an extensive all out KNN. The application then decides the maximum occurrence among the "5" nearest neighbours calculated. We have chosen 5 to be the number as each node (Room/Path) has approximately entries. Meaning the room itself would give a saturation as in all 5 possibility would be that room. Further the number 5 is chosen for the next step which is navigation. As with 5 nodes we can identify which room the user is in approximate 3 entries and the remaining 2 possibility can be seen as the next closest room or node.

3.2. HINDERANCE RECORDED

The application struggled with unmarked hidden Access Points or in situations where the room does not have a proper coverage meaning Rooms which only have 1 or 2 close Access Points hence the user cannot be associated to a specific cluster as multiple clusters will have that Access Point as port of their database. This hindrance can easily be overcome by involvement of more users and their input in unexamined locations and thereby completing the database.

4.1. NAVIGATION

The application is then used for the daunting task of navigation. Navigation is handled by the Dijkstra's Algorithm. The user's location is provided as source and the user is allowed to pick the location. The source and the destination is provided to the Dijkstra's algorithm which returns back intermediate nodes. Then the application draws a path on the Map for the user. Just like Identification the navigation process is thought off in great detail. For simplicity and verification of correctness the testing environment is limited to only the second floor which includes 20 rooms and 26 path points.

4.2. Dijkstra's algorithm

The application relies completely on the dijkstra's algorithm for routing and navigation. Dijkstra's algorithm is the most popular and commonly used for finding the shortest paths between nodes in a graph but in our case in between rooms and path nodes. Dijkstra's original variant found the shortest path between two nodes,but a more common variant fixes a single node as the "source" node and finds shortest paths from the source to all other nodes in the graph, producing a shortest-path tree.

An overview of the algorithm is mentioned below:

- Stamp all rooms/path nodes unvisited. Make a new set of all the unvisited rooms/path nodes called the unvisited set.
- Allocate to each rooms/path nodes the distance between each other it is to be noted that this is a bidirectional graph: set the distance to zero for our underlying starting/source room and to 9999 for every other rooms/path nodes. Set the underlying room/path node as present.
- For the present room/path node, consider all of its unvisited neighbours and calculate their *tentative* distances through the current node. Contrast the recently computed distance with the current computed distance and relegate the smaller one. For instance, if the present hub An is set apart with a separation of 6, and the edge interfacing it with a neighbour B has length 2, at that point the separation to B through A will be 6 + 2 = 8. On the off chance that B was already set apart with a separation more prominent than 8 at that point change it to 8. Something else, keep the present distance.
- When we are finished thinking about the majority of the neighbours of the present node, stamp the present node as visited to and remove it from the unvisited set. A visited to hub will never be checked again.
- Move to the following unvisited room/path node with the minimum conditional distance and repeat the above steps which check neighbours and assign them as visited.
- If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the *unvisited set* is infinity (when planning a complete traversal; occurs when there is no connection between the initial node and remaining unvisited nodes), then stop. The algorithm has finished.
- Select the unvisited node that is set apart with the minimum distance, set it as the new "current node", and backpedal to step 3.

4.2. CONCLUSIONS

The Prototype built on course of this project serves as an evidence that indoor navigation is quite a simple and valuable tool both for the users and the data collectors. The prototype shows that indoor navigation does not require any additional equipment in terms of hardware or software. A simple application can be designed which neither require Wi-Fi or the GPS. The naïve prototype created in due course of this project successfully achieves an accuracy of more than 5m which leaves scope for future improvements by using more complex techniques and heuristics.

Extensive research was done about creation of the fingerprint database as to how any entries are optimal to achieve maximum accuracy and how many nodes are to be consider per node to be entered in the database and to which Wi-Fi SSID is best for the particular application. Another viable option of research is that the time in which the measurements are made as the Wi-Fi dynamics change throughout the day as the density of students connected to Wi-Fi in different location change with time. In addition if we have multiple users on different devices taking measures more date would be available to have higher accuracy. As a result of this enhanced and improved database the prototype's primary KNN algorithm is able to achieve a more accurate location.

Moreover, other location techniques such as GPS applied in outdoor environments have had extensive research by giant corporations and now GPS operates with great accuracy. Many apps that we commonly use like Facebook, Instagram, Snapchat all work in harmony with GPS location but none with Indoor Navigation. By developing this project we change can achieve a trend of applications useful for security, hospitals, military use and so on. With some additional work indoor navigation can easily become more accurate than GPS and easier to implement as most moving nodes such as users and UAV, robots all are capable of picking up Wi-Fi Signals.

Development of the prototype has been a great experience trying to develop a relatively new technology and learnt a lot of concepts and facts like indoor navigation is a very primal application of Data mining and Machine Learning with certain tweaks and by connecting the database to a cloud we can truly call it an application of Machine Learning as the app gradually become more and more accurate with time and users.

This project can be further extended to achieve automated attendance system by using the prototype as a framework. The app can be connected to the web which uploads the user's location as certain timeslots on different days and we can simply run an algorithm on the collected database to collect the id of students which have where present in the classroom as the correct timeslot.

The decision to use android as a platform has been quite successful thanks to the combination of an accessible and well documented API along with system deployed on numerous mobile phones let us foresee that in the near future both the number of applications and the need for developers in this area will increase. Hence the experience gained during the development of this project will be highly valued.

5. PROTOTYPE CREATION

5.1. PROTOTYPE FUNCTIONS

5.1.1. WIFI SCANNING

```
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    onResume();
    lvWifiDetails = (ListView) findViewById(R.id.lvWifiDetails);
    btnRefresh    = (Button) findViewById(R.id.btnRefresh);
    mainWifi      =
(WifiManager)getApplicationContext().getSystemService(Context.WIFI_SERVICE);

    receiverWifi  = new WifiReceiver();

    registerReceiver(receiverWifi, new
IntentFilter(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION));

    scanWifiList();
}
private void scanWifiList()
{
    Toast.makeText(MainActivity.this,"scanWifiList",Toast.LENGTH_LONG).show();
    mainWifi.startScan();
    wifiList = mainWifi.getScanResults();
    setAdapter();
}

private void setAdapter()
{   adapter = new ListAdapter(getApplicationContext(), wifiList);
    lvWifiDetails.setAdapter(adapter);
}


class WifiReceiver extends BroadcastReceiver
{
    public void onReceive(Context c, Intent intent)
    {   if (intent.getAction().equals(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION))
        {    wifiList = mainWifi.getScanResults();  }
    }
}
```

5.2. WIFI FILTERING

```
private void FilterWifiList()
{
    Toast.makeText(MainActivity.this,"FilterWifiList",Toast.LENGTH_LONG).show();
    mainWifi.startScan();
    wifiList = mainWifi.getScanResults();

    for(int i = 0; i < wifiList.size(); i++)
    {
        if(!wifiList.get(i).SSID.equals("BITS-Students"))
        //if(!wifiList.get(i).SSID.equals("Dina"))
        {
            wifiList.remove(i);
            i--;
        }
    }
```

```
    setAdapter();
}
```

5. 1.3. DATABASE CREATION

```
private void RecordFingerPrint() throws IOException
{
    mainWifi.startScan();
    wifiList = mainWifi.getScanResults();

    String toFile ="";

    ArrayList<String> dataToFile = new ArrayList<String>();
    for(int i=0; i < wifiList.size(); i++)
    {   //change to test
        if(wifiList.get(i).SSID.equals("BITS-Students"))
        {
            dataToFile.add(wifiList.get(i).BSSID+" | "+wifiList.get(i).level);
        }
    }

    if(dataToFile.size() >= 3)
    {
        for(int i=0; i < 3; i++)
            toFile += dataToFile.get(i)+" | ";

        EditText text = findViewById(R.id.EditText01);
        String str = text.getText().toString();
        toFile += str+"\n";
        File dir = new File(path);
        dir.mkdirs();

        File file = new File(path + "/savedData.txt");

        FileOutputStream fos = null;
        try
        {   fos = new FileOutputStream(file,true);    }
        catch (FileNotFoundException e) {e.printStackTrace();}
        try
        {
            fos = new FileOutputStream(file,true);
            fos.write(toFile.getBytes());
            Toast.makeText(MainActivity.this,"Data
Added",Toast.LENGTH_LONG).show();
        }
        catch(FileNotFoundException ex2)
        {
            Toast.makeText(MainActivity.this,"Failed to Add
Data",Toast.LENGTH_LONG).show();
        }finally
        {
            fos.close();
        }
    }
}
```

5. 1.4. IMPLEMENTATION OF KNN

```
private void FindRoomNumber()
{   mainWifi.startScan();
    wifiList = mainWifi.getScanResults();

    TextView RoomNumber = findViewById(R.id.RoomNumberTxt);

    TextView stat1 = findViewById(R.id.stat1);
    TextView stat2 = findViewById(R.id.stat2);
    TextView stat3 = findViewById(R.id.stat3);
    TextView stat4 = findViewById(R.id.stat4);
    TextView stat5 = findViewById(R.id.stat5);


    ArrayList<String> Scanned_BSSID = new ArrayList<String>();
    ArrayList<Integer> Scanned_RSSI = new ArrayList<Integer>();

    for(int i=0; i < wifiList.size(); i++)
    {   //change to test
        if(wifiList.get(i).SSID.equals("BITS-Students"))
        {
            Scanned_BSSID.add(wifiList.get(i).BSSID);
            Scanned_RSSI.add(wifiList.get(i).level);
        }
    }

    String omg = "";

    if(Scanned_BSSID.size() >= 3)
    {
        File file = new File(path + "/savedData.txt");
        FileInputStream fis = null;

        try
        {   fis = new FileInputStream(file);    }
        catch (FileNotFoundException e) {e.printStackTrace();}

        InputStreamReader isr = new InputStreamReader(fis);
        BufferedReader br = new BufferedReader(isr);

        ArrayList<String> inFromFile = new ArrayList<String>();
        String line;
        try
        {   while((line=br.readLine())!=null)
            {   //CHANGE IN FUTURE
                int a = line.indexOf(Scanned_BSSID.get(0));
                int b = line.indexOf(Scanned_BSSID.get(1));
                int c = line.indexOf(Scanned_BSSID.get(2));
                if((a >= 0)&&(b >= 0)&&(c >= 0))
                {   inFromFile.add(line);   }
            }
        }
        catch (IOException e) {e.printStackTrace();}

        //  apply algorithm here

        double minimumDistance = 99999.0;
        String Roomno = "";
        omg += inFromFile.size()+"\n";
```

```
        HashMap<Double, String> hmap = new HashMap<Double, String>();

        for(int i = 0; i < inFromFile.size();i++)
        {   StringTokenizer st = new StringTokenizer(inFromFile.get(i), " | ");
            String BSSID1 = st.nextToken();
            int RSSI1 = Integer.parseInt(st.nextToken());
            String BSSID2 = st.nextToken();
            int RSSI2 = Integer.parseInt(st.nextToken());
            String BSSID3 = st.nextToken();
            int RSSI3 = Integer.parseInt(st.nextToken());
            String rno = st.nextToken();

            int rs1 = Scanned_RSSI.get(Scanned_BSSID.indexOf(BSSID1));
            int rs2 = Scanned_RSSI.get(Scanned_BSSID.indexOf(BSSID2));
            int rs3 = Scanned_RSSI.get(Scanned_BSSID.indexOf(BSSID3));

            double tot = EuclideanDistance(RSSI1,rs1,RSSI2,rs2,RSSI3,rs3);
            if(tot < minimumDistance)
            {
                minimumDistance = tot;
                Roomno = rno;
            }

            hmap.put(tot,rno);
            omg += rno+"\t\t: "+tot+"\n";
        }
        RoomNumber.setText(Roomno);

        Map<Double, String> map = new TreeMap<Double, String>(hmap);
        Set set2 = map.entrySet();
        Iterator iterator2 = set2.iterator();
        Map.Entry me2;

        if(iterator2.hasNext())
        {   me2 = (Map.Entry)iterator2.next();
            stat1.setText(me2.getKey()+" || "+me2.getValue());
        }
        if(iterator2.hasNext())
        {   me2 = (Map.Entry)iterator2.next();
            stat2.setText(me2.getKey()+" || "+me2.getValue());
        }
        if(iterator2.hasNext())
        {   me2 = (Map.Entry)iterator2.next();
            stat3.setText(me2.getKey()+" || "+me2.getValue());
        }
        if(iterator2.hasNext())
        {   me2 = (Map.Entry)iterator2.next();
            stat4.setText(me2.getKey()+" || "+me2.getValue());
        }
        if(iterator2.hasNext())
        {   me2 = (Map.Entry)iterator2.next();
            stat5.setText(me2.getKey()+" || "+me2.getValue());
        }

        if(inFromFile.size() == 0)
        {    RoomNumber.setText("Update Dataset");  }
    }
    else
    {   RoomNumber.setText("Update Dataset");   }
}
public double EuclideanDistance (int RSSI1,int rs1,int RSSI2,int rs2,int RSSI3,int
```

```
rs3)
{
    double a = Math.pow(RSSI1-rs1 ,2);
    double b = Math.pow(RSSI2-rs2 ,2);
    double c = Math.pow(RSSI3-rs3 ,2);
    return Math.sqrt(a+b+c);
}
```

5.1.5. DIJKSTRA ALGORITHM

```
package com.ascot.mxiv.zenith;
import android.util.Log;

import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;

public class DijkstraAlgorithm {

    private final List<Edge> edges;
    private Set<Vertex> settledNodes;
    private Set<Vertex> unSettledNodes;
    private Map<Vertex, Vertex> predecessors;
    private Map<Vertex, Integer> distance;

    public DijkstraAlgorithm(Graph graph) {
        // create a copy of the array so that we can operate on this array
        this.edges = new ArrayList<Edge>(graph.getEdges());
    }

    public void execute(Vertex source, Vertex destination) {
        settledNodes = new HashSet<Vertex>();
        unSettledNodes = new HashSet<Vertex>();
        distance = new HashMap<Vertex, Integer>();
        predecessors = new HashMap<Vertex, Vertex>();
        distance.put(source, 0);
        unSettledNodes.add(source);
        while (unSettledNodes.size() > 0) {
            Vertex node = getMinimum(unSettledNodes);
            settledNodes.add(node);
            unSettledNodes.remove(node);
            findMinimalDistances(node);
        }

    }

    private void findMinimalDistances(Vertex node) {
        List<Vertex> adjacentNodes = getNeighbors(node);
        for (Vertex target : adjacentNodes) {
            if (getShortestDistance(target) > getShortestDistance(node)
                    + getDistance(node, target)) {
                distance.put(target,
                        getShortestDistance(node) + getDistance(node, target));
                predecessors.put(target, node);
                unSettledNodes.add(target);
            }
        }

    }

    private int getDistance(Vertex node, Vertex target) {
        for (Edge edge : edges) {
            if (edge.getSource().equals(node)
```

```
                && edge.getDestination().equals(target)) {
            return edge.getWeight();
        }
    }
    throw new RuntimeException("Should not happen");
}

private List<Vertex> getNeighbors(Vertex node) {
    List<Vertex> neighbors = new ArrayList<Vertex>();

    for (Edge edge : edges) {
        if (edge.getSource().equals(node)
                && !isSettled(edge.getDestination())) {
            neighbors.add(edge.getDestination());
        }
    }
    return neighbors;
}

private Vertex getMinimum(Set<Vertex> vertexes) {
    Vertex minimum = null;
    for (Vertex vertex : vertexes) {
        if (minimum == null) {
            minimum = vertex;
        } else {
            if (getShortestDistance(vertex) < getShortestDistance(minimum)) {
                minimum = vertex;
            }
        }
    }
    return minimum;
}

private boolean isSettled(Vertex vertex) {
    return settledNodes.contains(vertex);
}

//private int getShortestDistance(Vertex destination) {
public int getShortestDistance(Vertex destination) {
    Integer d = distance.get(destination);
    if (d == null) {
        return Integer.MAX_VALUE;
    } else {
        return d;
    }
}

/*
 * This method returns the path from the source to the selected target and
 * NULL if no path exists
 */
public ArrayList<Vertex> getPath(Vertex target) {
    ArrayList<Vertex> path = new ArrayList<Vertex>();
    Vertex step = target;
    // check if a path exists
    if (predecessors.get(step) == null) {
        return null;
    }
    path.add(step);
    while (predecessors.get(step) != null) {
        step = predecessors.get(step);
        path.add(step);
    }
    // Put it into the correct order
    Collections.reverse(path);
    Log.d("omg","inside Dj"+path.size());
    return path;
}
```

```
}
```

5. 1.6. ROUTING

```
    public void findRoute(String src,String des) {
        refreshMap();

        String source = src;
        String destination = des;

        Log.d("hi", "called");
        CalculatePath obj = new CalculatePath(getApplicationContext());
        ArrayList<Vertex> al = new ArrayList<>();
        al = obj.initialize(n, e, source, destination);
        String str = "Shortest Path :\n";

        //Insert BLOCK

        Paint paint = new Paint();
        paint.setStrokeWidth(35);
        paint.setStyle(Paint.Style.STROKE);
        paint.setDither(true);                    // set the dither to true
        paint.setStyle(Paint.Style.STROKE);       // set to STOKE
        paint.setStrokeJoin(Paint.Join.ROUND);    // set the join to round you want
        paint.setStrokeCap(Paint.Cap.ROUND);      // set the paint cap to round too
        paint.setPathEffect(new CornerPathEffect(10));   // set the path effect
when they join.
        paint.setAntiAlias(true);
        // PATH START
        Path p = new Path();
        //p.moveTo(0,0);
        p.moveTo(RoomX.get(al.get(0).getName()), RoomY.get(al.get(0).getName()));
        int i;
        for (i = 0; i < al.size(); i++) {
            str += al.get(i).getName() + "\n";
            p.lineTo(RoomX.get(al.get(i).getName()),
RoomY.get(al.get(i).getName()));
        }

        paint.setColor(0xff800000);
        paint.setShader(new LinearGradient(0, 0, 0, 1700, 0xff3F51B5, 0xffFF4081,
Shader.TileMode.MIRROR));
        canvas.drawPath(p, paint);

        Paint paint2 = new Paint();
        //Starting Dot
        paint2.setColor(Color.WHITE);
        canvas.drawCircle(RoomX.get(al.get(0).getName()),
RoomY.get(al.get(0).getName()), 35, paint2);
        paint2.setColor(Color.parseColor("#40c7dc"));
        canvas.drawCircle(RoomX.get(al.get(0).getName()),
RoomY.get(al.get(0).getName()), 25, paint2);

        //ending Dot
        i--;
        paint2.setColor(Color.WHITE);
        canvas.drawCircle(RoomX.get(al.get(i).getName()),
RoomY.get(al.get(i).getName()), 35, paint2);
        paint2.setColor(Color.parseColor("#5c63f6"));
        canvas.drawCircle(RoomX.get(al.get(i).getName()),
RoomY.get(al.get(i).getName()), 25, paint2);
```

```
        Log.d("omg", str + "this work");
        PathMeasure pathMeasure = new PathMeasure(p, false);
        float pathLength = pathMeasure.getLength();

        Toast.makeText(this, "pathLength: " + pathLength,
Toast.LENGTH_LONG).show();

    }
public void drawRoute()
{   final TextView RoomNumber = findViewById(R.id.RoomNumberTxt);
    final TextView DesRoomNumber = findViewById(R.id.desroomNumber);

    String src = "" + RoomNumber.getText();
    Log.d("sup", src);

    String des = ""+DesRoomNumber.getText();
    Log.d("sup", des);
    if(!src.equals("Update Dataset") && !src.equals("ROOM") &&
!des.equals("DESROOM") && !src.equals(des))
    {   findRoute(src, des);    }
}
```

Indoor Navigation Final Report Submission

ORIGINALITY REPORT

0%	0%	0%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

Exclude quotes	On	Exclude matches	< 10%
Exclude bibliography	On		