International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 08, August - 2017

Indoor Navigation System using Fingerprinting

Habib M., Abraham . K, Abel D., Amanuel W.,
Gebremikael T., and Michaele G.
College of Electrical and Mechanical Engineering,
Department of Electrical and Electronics Engineering,
Addis Ababa Science and Technology University (AASTU),
Addis Ababa, Ethiopia

***Abstract*—The increasing demand for accurate and affordable indoor positioning has caused the increase in research to achieve it. Although Global Satellite Positioning Systems have acceptable accuracy outdoors, they do not have acceptable accuracy indoors. This led to researchers exploring different systems for indoor positioning. This paper implements an indoor navigation system using fingerprinting. It compares the accuracy of the deterministic kNN algorithm and probabilistic Naïve Bayes algorithm for positioning. We also develop a database table that can be used to give directions. A variant of the Dijkstra's shortest path algorithm is used to find nodes on the shortest path from a source node to a destination node on a server. The source node is determined by using the positioning system developed and the destination node is inputted by the user.**

Keywords—Fingerprint, KNN algorithm, Navigation Systems, Positioning techniques.

I. INTRODUCTION

Nowadays, most mobile applications are location aware. Location aware applications have been used for navigation, improving search results and geo-fencing. Most of the location aware applications use Global Positioning System (GPS). GPS is a navigation system that uses 24 US satellites scattered all over space to provide worldwide coverage. A smartphone equipped with a GPS receiver uses data from 3 of GPS satellites to determine its longitude and latitude. Nonetheless, GPS is well not suited for indoor use due mainly to the high degradation of signals from satellites when indoors. Thus, indoor positioning systems based on Bluetooth, Zigbee and WiFi have been proposed by different researchers [1] [2]. In this paper, we build a WiFi based indoor positioning system using a technique called fingerprinting [3], [4].

Bin Hu [5] developed a smartphone WiFi based indoor positioning system using smart phones. He developed the client side application on Android software platform, and the server side application using Apache Tomcat web server and Microsoft SQL Server database. The smartphone application he developed collects RSSI data and sends it to the Apache Tomcat server, Apache in turn saves the data to the SQL database. Later when a user requests location the database entries are used to estimate the user location. The estimated location is sent to the smartphone via the Apache Tomcat server. He used a custom made algorithm to estimate location.

Landu Jiang [6] developed a Wi-Fi based indoor localization technique based on fingerprinting. In his approach, he uses temporal data in addition to spatial data to estimate location. That is, the algorithm he developed not only uses RSSI values at this instant but also use values in the near past. Therefore, his algorithm does not consider mere points rather it considers tracking lines which contain a prescribed number of points. He evaluated the system using 10 tracking lines and achieved a 5% average increase in accuracy compared to all other RSSI based indoor positioning systems.

Abdul Quyum [7] developed a Wi-Fi based indoor positioning. To mitigate Wi-Fi RSSI fluctuations he used particle filters. During his investigation, he also discovered that collecting calibration data in all directions can bring about better performance.

II. SYSTEM DESIGN

In fingerprinting, a radio map of the building is built by surveying the area for which indoor positioning is desired first. Then, the radio map is used to train a pattern recognition model. The trained pattern recognition model is saved and later used to estimate the position of target devices. The stage in which radio map is built is called offline (calibration) phase. After the pattern recognition model is built, the target device sends the RSSI values it sees to a server which uses the trained classifier to predict the position of the target device. The access points used were one TP-Link TL-WA901N and two TP-Link TL-WA801N. The network shown in Figure 1 was setup. Then, we used an RSSI data collection Android application we developed to collect RSSI data. The Android application collects RSSI data automatically and prompts the data collecting personnel for a label of the current location. It then forwards the RSSI data and label to a data collection server running on the Raspberry Pi. The data collection server receives the RSSI values and the labels and adds it as row to a file. We collected data at 51 reference points the second floor of Block 21 in Addis Ababa Science and Technology University. The app does not just send the set of RSSI values of WiFi signals it detected. It sends which RSSI value was detected from which access point. The wireless access points are identified by their MAC address. The selected reference points are shown in Figure 2. The data collection server is given a list the MAC addresses of the wireless access points that will be used for positioning. Using this list, the location server filters out entries for access points that will not be used for positioning. When data has been collected for all reference points, the server is stopped and it saves all the data it collected in a Numpy compressed array format (.npz). Numpy is a popular Python library for scientific computing. Seventy rounds of data collection were performed thus giving 70 RSSI vectors for each reference point.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 08, August - 2017

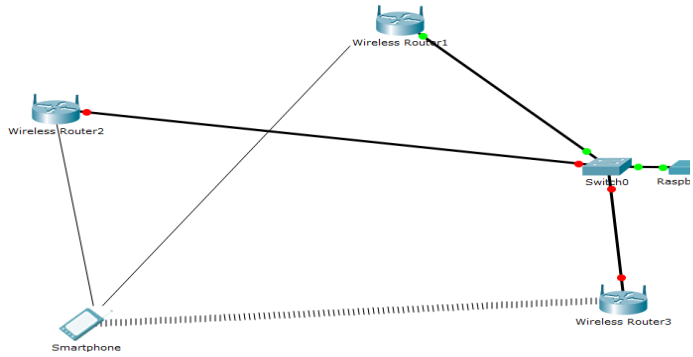

Figure 1Architecutre of the proposed positioning system

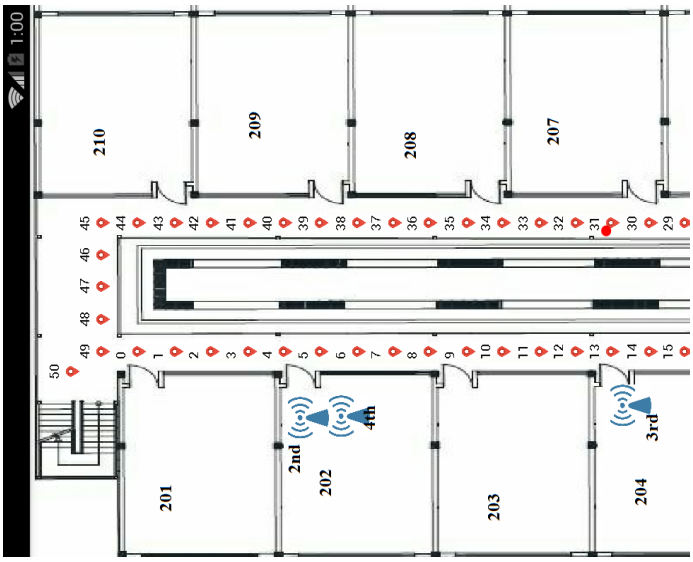

Figure 2: Locations of selected reference points and access points

The data in npz file which will be used to train a classifier. The classifier will then be used to estimate the position of target devices during the positioning phase. The Python library sklearn (sci-kit learn) was used to train the pattern recognition model. First, a k-Nearest Neighbors classifier was trained. Then, the classifier was stored to a file using a persistence model. A module called joblib was used for model persistence. The classifier used was the k-Nearest Neighbors (KNN) algorithm with k equal to 3. A Gaussian Naïve Bayes classifier was also trained and tested on the system. This completes the calibration phase. In the positioning phase, the target device scans for WiFi networks, collects RSSI and MAC address data, sends it to a location server and waits for the server's response. The server receives the data and loads the trained classifier. The classifier is then used to estimate the position. The classifier is fed the RSSI value for each access point used for positioning and outputs a reference point number. The reference point numbers and their corresponding coordinates are stored in a MySQL database. The location server queries the database to determine the coordinates of the predicted reference point on the floor. The coordinates are then converted to their corresponding pixel locations in the floor map image. The pixel coordinates of the estimated location in the original floor map is then sent to the target device. The target device receives this data and sends a request for the floor map to the location server. The location server receives this request and then transfers the location server to the target device. When all of the image is received the target device displays the floor map on its screen. The target device, however, does not display the floor map image in its original size. It scales the image to fit its screen. As it does so it saves the scale factor for its width and height. The scale factors are used to scale the pixel coordinates received from the location server which were the locations in the original image. The other thing that our system can do in the positioning phase is give directions to a desired location from the current location. A lot of the steps on the target device's side to get directions are similar to the steps used to know its location. Similar to the location estimation case, the Android application first sends the RSSI and MAC address data to the location server. The location server first sees the type of the request and determines that it is a request for directions to a room. Then, it parses important information such as the destination room, RSSI and MAC address data. It then feeds the RSSI and MAC address data to the trained classifier. The trained classifier returns the reference point number of the estimated location. The location server queries the MySQL database for the coordinates of the reference point number. The MySQL database does not just have the rows with reference point number and their corresponding coordinates. Rather, it also contains two columns: the first one contains a list of the reference point numbers of immediate neighbors for each reference point and the second one contains distance to each neighbor. These two columns are used to build a graph representation of the reference points in the graph. A graph representation needs to be built from the database entries. We developed a Python module to extract the entries of the database and develop an adjacency list representation of the graph. The module also has an extended implementation of the Dijkstra's shortest path algorithm. After the graph is built, Dijkstra's algorithm is run on the graph which returns the shortest path to the destination from the estimated position. The Dijkstra's algorithm we implemented returns the set of consecutive reference points the shortest path must pass through. The used variant of Dijkstra's algorithm is described below. The location server queries the MySQL database to determine the floor coordinates of each reference point in the shortest path, converts the floor coordinates to pixel coordinates and forwards it to the target device. Then, the location server sends the floor map image to the target device. Upon receiving all this information, the target device displays the floor map and draws a line between every consecutive point in the data it received.

```
Dijkstra (Graph, source, destination):
Set the distance attribute of all nodes to ∞
Set the parent attribute of all nodes to NIL
Set the distance attribute of the source node
to 0
Add all the nodes of the graph to a queue Q
while Q is not empty:
       u = pop a node from Q
       for each vertex v adjacent to node u:
              if  v.distance  >  u.distance  +
weight of edge from u to v:
                     v.distance = u.distance +
weight of edge from u to v
                     v.parent = u.parent
u = destination
path = empty list
while the parent of u is not NIL:
       add u to the path list
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 08, August - 2017

```
        u = the parent of u
reverse the path list
return path
```

Algorithm 1: Dijkstra's algorithm [8]

III. EXPERIMENTAL RESULTS AND DISCUSSIONS

Three access points were fixed in Block 21 of at AASTU. The access points were placed in the second floor, third floor and the fourth floor. All the access points were connected to a switch. Also connected to the switch was a Raspberry Pi. The data collection server and the location server were installed in the Raspberry Pi. Then, fifty-one points were selected on the corridor of the second floor of Block 21. Each of these reference points were assigned a label (or a reference point number). Consecutive reference points were placed at a distance of 1.5m from each other. Initially, the access points were placed on the corridors of the 2nd, 3rd and 4th floors, three rounds of data collection were carried out with the access points located at these locations. However, upon examining the collected data we noted that up to 10 consecutive reference points had the same set of RSSI values. This is undesirable since the classifier will have a hard time in distinguishing between the consecutive reference points that have the same set of RSSI values. We suspected that the cause of the similarity in RSSI values was the availability of a line-of-sight path from all the access points to the target device. The availability of line-of-sight path means that the set of RSSI values received mainly depend on the distance from the access points. And a significant change in RSSI value cannot be expected in points separated by 1.5 m when LOS path is available. Thus, we placed the three access point inside rooms. The first was placed in room 202 in the second floor, the second was placed in room 304 in the third floor and the last one placed in room 402 of the fourth floor. By placing the access points in the rooms we made the RSSI from each access point not only dependent on the distance from access point but also on the number of walls the WiFi signal has to pass through. The placement of the access points indoors significantly improved the distinguishability of the RSSI vectors at consecutive reference points. All subsequent experiments were carried out with the access points placed at these locations. The locations of selected reference points and their and their labels are shown in Figure 2. The locations of the access points are also shown in the map. The labels 2nd, 3rd and 4th next to the access point symbols in the map show the floor the access point is at. The training data was collected with a Huawei SCL-U31 phone which runs Android OS. The RSSI Collector application was installed in the device. At each reference point, seventy sets of RSSI values were collected. To test the indoor positioning system, 52 points were selected. The Huawei phone was placed at some location. The phone holder notes which grid point it should be classified as. This is done by identifying the grid point that is closest to the test point. Then, the user opens the app and sends a WAI request to the server to which the server replies with the estimated grid point. The estimated grid point is noted and compared with the grid point it should be classified as. All of the selected test points but two were closer to different reference points. Upon completion of the test in sixty-six percent of the test points were correctly labeled, nine percent were labelled to be at the next reference point and 5.5 percent were labelled to be two grid points away by the KNN classifier. The Bayesian classifier correctly predicted for 18% of the test points only. The whole summary of the results is shown in figure 3and figure 4. At each of the selected test point, we run the find direction feature to different destinations. The system was 100% accurate in finding the shortest path. However, because it had to estimate the current position and that was about 66% accurate some mistakes in the starting position (source node) were seen.

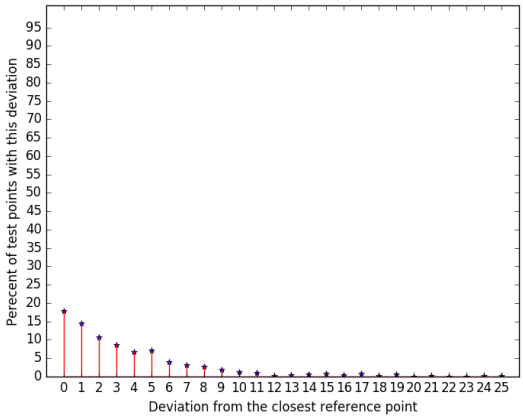

Figure 3: Accuracy of the Bayesian classifier

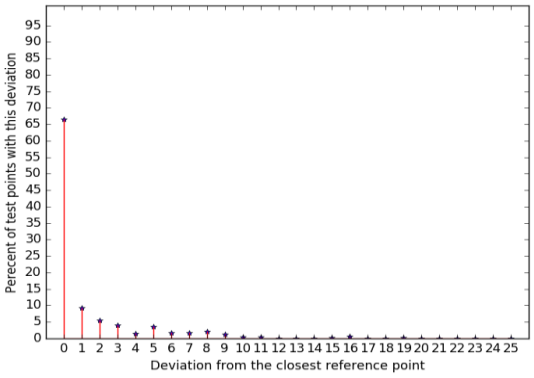

Figure 4: Accuracy of the kNN classifier

IV. CONCLUSION

We proposed a method for positioning and navigation that can be applied indoors. Results show that kNN classifier achieves an accuracy of 0.75 m (position predicted as the closest reference point with a reference point separation of 1.5 m) 66% of the time and an accuracy of 1.5 m (position predicted as the second closest reference point) 9% of the time. The kNN classifier outperformed the Gaussian naïve Bayes which achieved an accuracy of 0.75m only 17% of the time. It was also noted that, reception of strong signals over wide areas causes the degradation of performance by yielding indistinguishable RSSI vectors at consecutive reference points.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 08, August - 2017

REFERENCES

[1] M. A. Al-Ammarz, S. Alhadhrami, A. Al-Salman, A. Alarifiy, H. S. Al-Khalifa, A. Alnafessahy and M. Alsalehy, "Comparative Survey of Indoor Positioning Technologies, Techniques, and Algorithms," in *International Conference on Cyberworlds*, Santander, 2014.

[2] D. R. Mautz, "Indoor Positioning Technologies," 2012.

[3] R. O. Duda, P. E. Hart and D. G. Stork, "Maximum likelihood and Bayesian estimation," in *Pattern Classification*, New York, Wiley - Interscience, 2001, pp. 84 - 140.

[4] S. He and S.-H. G. Chan, "Wi-Fi Fingerprint-based Indoor Positioning: Recent Advances and Comparisons," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 466 - 490, 2015.

[5] B. Hu, "Wi-Fi Based Indoor Positioning System Using Smartphones," 2013.

[6] L. Jiang, "A WLAN Fingerprinting Based Indoor Localization Technique," Lincoln, USA, 2012.

[7] M. A. Quyum, "Guidelines for Indoor Positioning," 2013.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, "Dijkstra's Algorithm," in *Introduction to Algorithms*, Massachusetts, The MIT Press, 2009, pp. 658-662.