CAS 765 Final Project Report: Indoor Localization

Yue Sun, Ye Li, Wade Genders

Dr Rong Zheng

9 Dec 2013

This report outlines the final project of CAS 765 by Yue Sun, Ye Li and Wade Genders. It details the indoor localization app developed for the Android OS to track a user's location inside the 1st floor of the ITB building at McMaster University. Our group designed a step counter and stride length (SL) algorithm that attempted to use a magnetometer for heading orientation. We encountered difficulty and poor accuracy using this method so we proceeded to develop a primitive wifi fingerprinting algorithm which achieved better results.

Algorithm – Step Count + Stride Length + Orientation

Building from lessons learned in previous assignments, our group built functionality for the final project app which calculated when a user took a step, the step distance and in which direction the step was taken. For simplicity, even though it encompasses more than just step counting, this paper will refer to the indoor localization that doesn't use wifi fingerprinting as the *step algorithm* and the wifi fingerprinting functionality as the *wifi algorithm.*

The architecture of the step algorithm can be seen below in Figure 1.

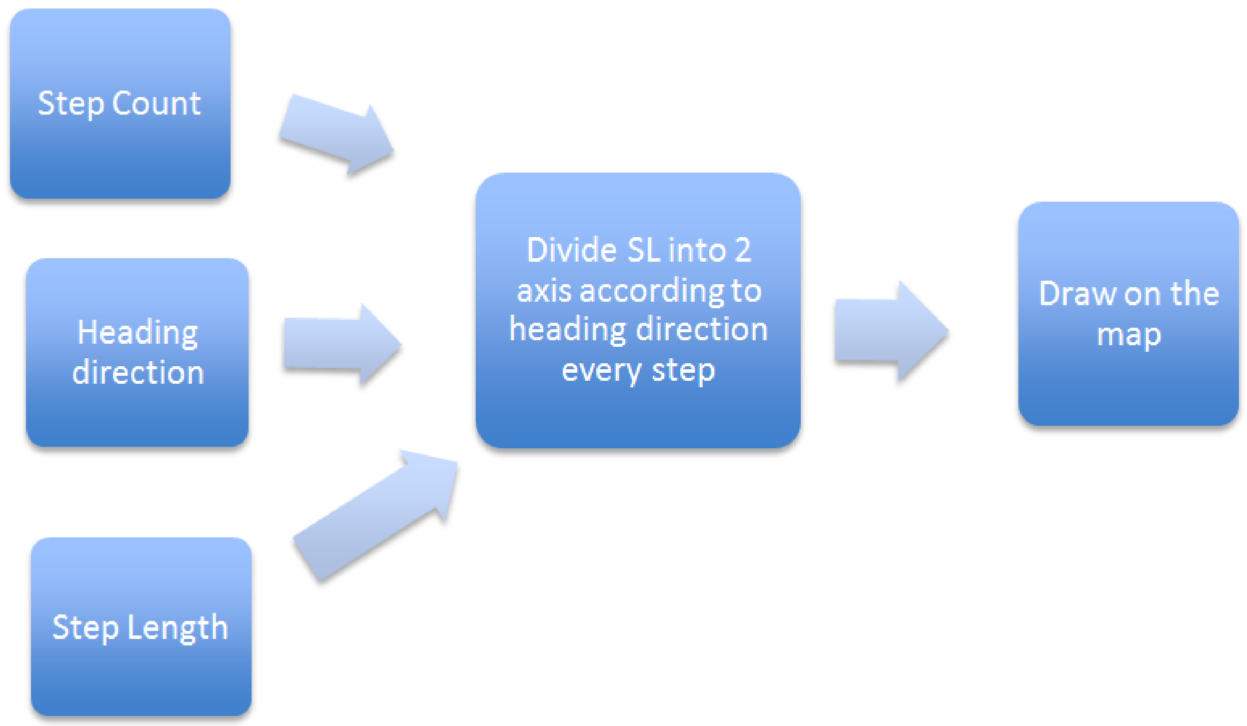

Step Count
Heading direction
Step Length
Divide SL into 2 axis according to heading direction every step
Draw on the map

Figure 1: Block diagram architecture of step algorithm.

The step count and SL components of the step algorithm were built using the knowledge gained from previous assignments in CAS 765 along with referencing material from the Weiberg algorithm presented in lecture [1]. To detect a step, the acceleration sensors were used, first by isolating the force of gravity on earth with a low pass filter and then removing it with a high pass filter. Then the signal was normalized with a filter and finally the acceleration readings were fed into the Weiberg algorithm to detect the step length (seen in Figure 2).

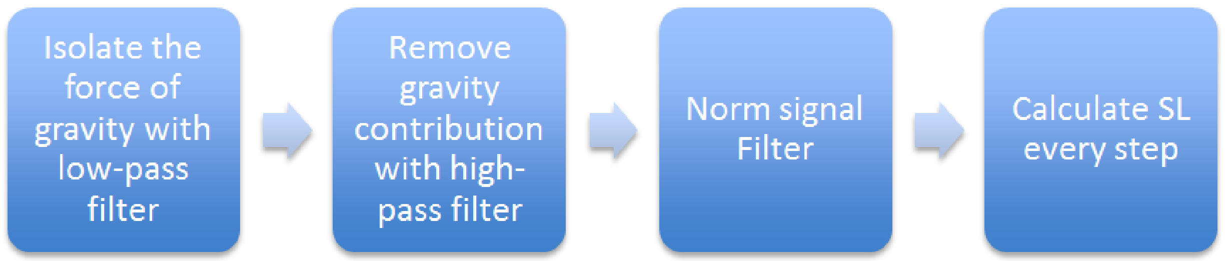

Isolate the force of gravity with low-pass filter
Remove gravity contribution with high-pass filter
Norm signal Filter
Calculate SL every step

Figure 2: Step detection and SL using accelerometers and Weiberg.

For the SL, we broke the step vector into its x & y components, using the angle of the heading orientation, and used this information to draw the user's path on the app map of ITB. A positive x component was to the east and a positive y component was to the north. The heading orientation is obtained by using a moving 16 step window and filtering the compass azimuth input to achieve the direction of each step (seen in Figure 3).

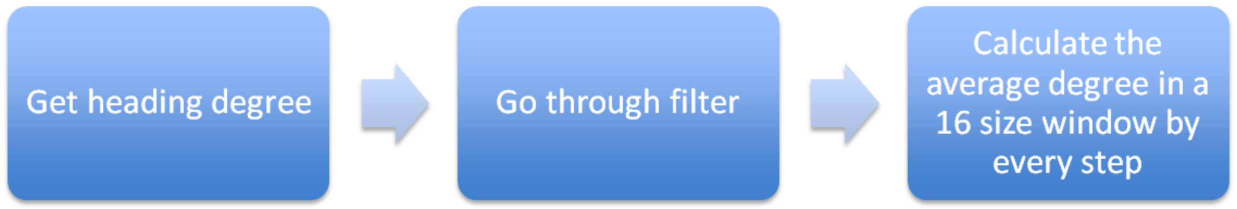

Get heading degree
Go through filter
Calculate the average degree in a 16 size window by every step

Figure 3: Steps to determine heading orientation using compass.

With a direction, x and y distance components, the final step was the draw the user's location on the map. The step algorithm required the user to begin using the app in a pre-determined location, and then every time a step was detected a new point was drawn on the map, in the position and direction determined by the heading, x and y components.

Algorithm – Wifi Fingerprinting

A second indoor localization algorithm was developed, using the unique properties of wireless access points (AP). Each AP has a unique identifier, BSSID, along with a received signal strength (RSS) from a given location. Obviously since wireless internet is provided using radio waves with limited signal power, they do not propagate over an infinite distance. However, on the order of 10's of metres, different signal strengths can be observed of any given AP broadcasting. The RSS of an AP provides an additional metric by which to uniquely identify ones location relative to an AP; the higher the RSS, the closer the AP. If many APs can be detected within a given area, it is possible to use the BSSIDs and RSS to create a wireless fingerprint from the mentioned metrics. If at location A the user can see AP1 (BSSID:111, RSS:-90db) & AP2 (BSSID:222, RSS:-85db) and at location B the user observes AP2 (BSSID:111, RSS:-80db) & AP3 (BSSID:333, RSS:-70db), then the user has enough information to distinguish their location from A or B. Wireless signals propagate through the atmosphere and can encounter interference from many sources, reflecting and diffracting off objects along with sources of electromagnetic interference. Interference changing the RSS forced us to develop wireless fingerprints which detected an AP's BSSID and RSS in relation to a range of RSS. Given a RSS, our algorithm determined if it was within a pre-determined ranged of acceptable signal strengths which were collected prior to the deployment of the app.

Our group first needed to collect and build a set of wifi fingerprints which corresponded to locations within ITB. This was accomplished with slight modifications to the data collector app developed in a previous assignment. Our procedure for collecting fingerprints was to walk in a 1m square and collect wifi data for 10-20 seconds. The logic to build the data from the collected data can be seen below in Figure 4.

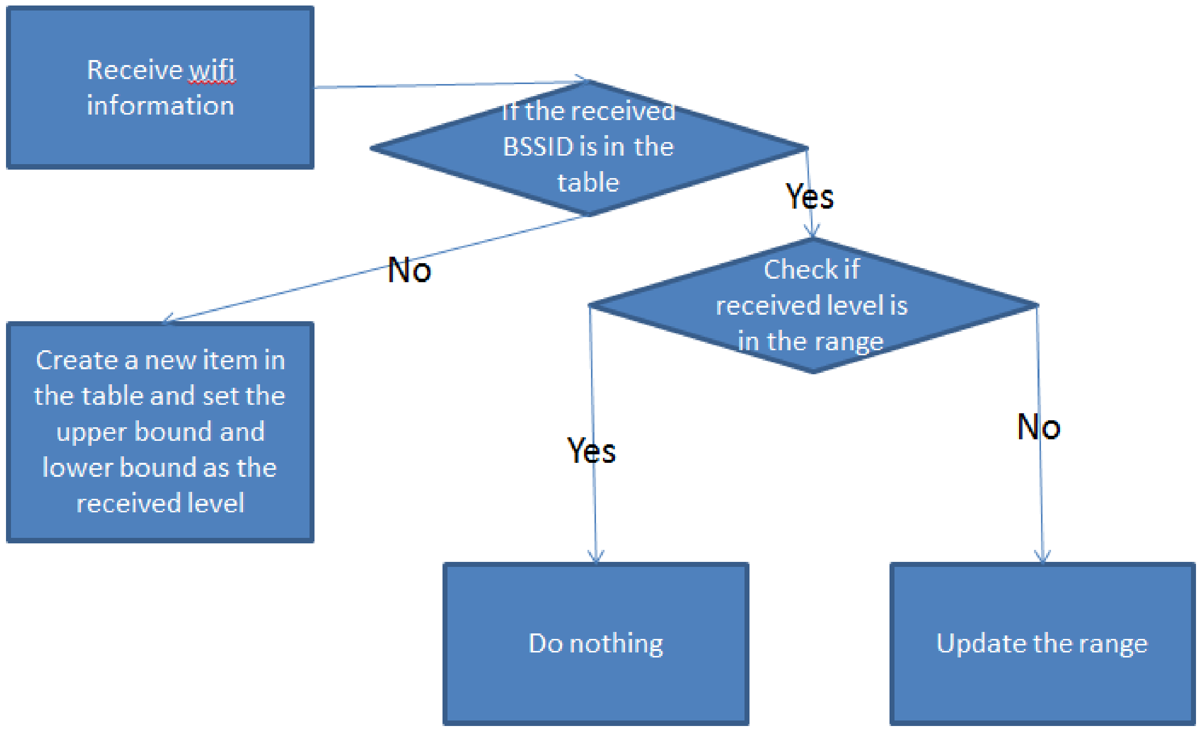

Receive wifi information
If the received BSSID is in the table
Yes
No
Check if received level is in the range
Create a new item in the table and set the upper bound and lower bound as the received level
Yes
No
Do nothing
Update the range

Figure 4; Logic for building wifi fingerprint table.

Whenever data was collected, we had to record the location within the building and connect the wifi data collected to this physical location. We collected data at 44 points within ITB, each approximately 2-4m apart. The locations of each wifi fingerprint can be seen below in Figure 5.

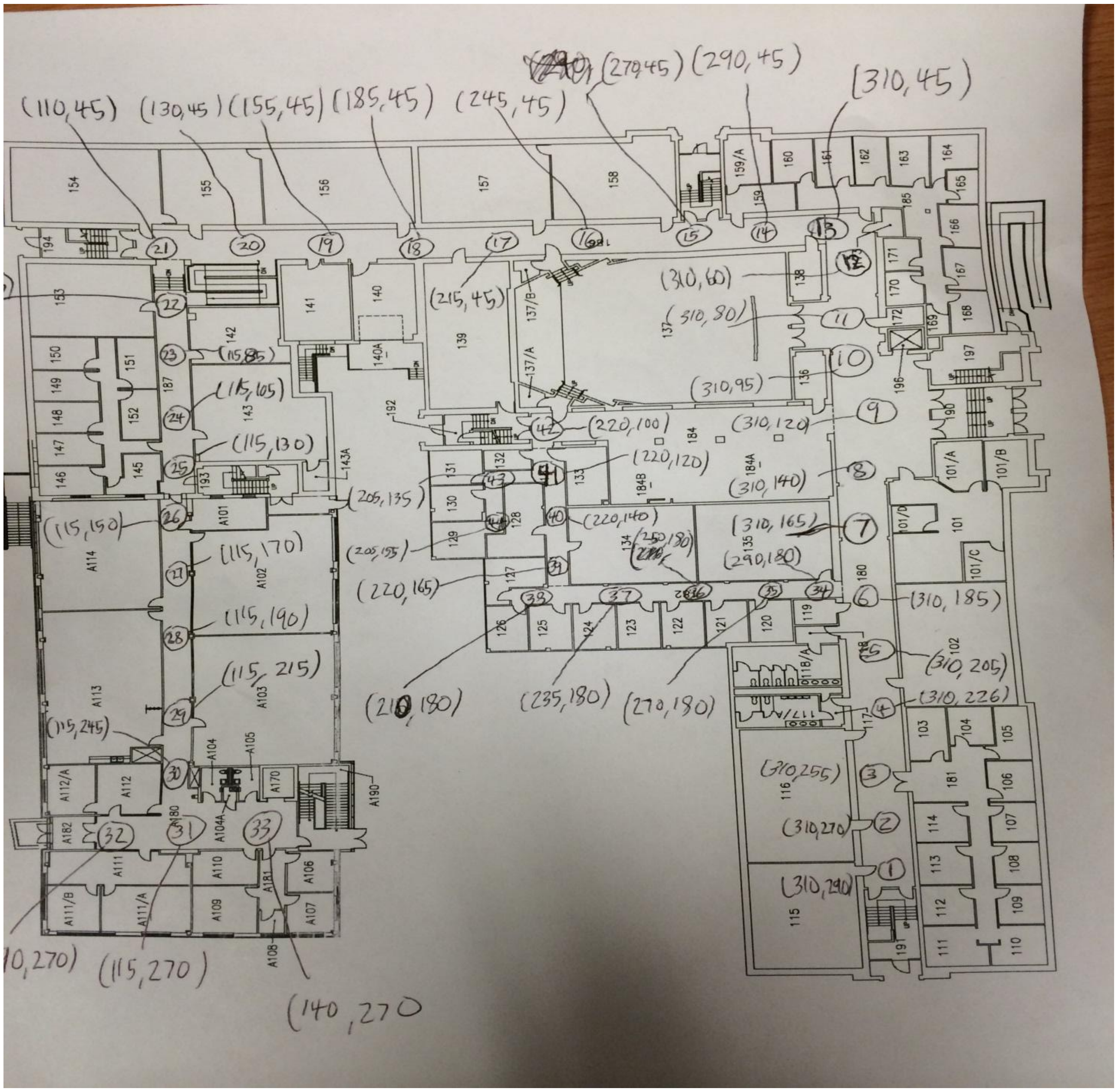
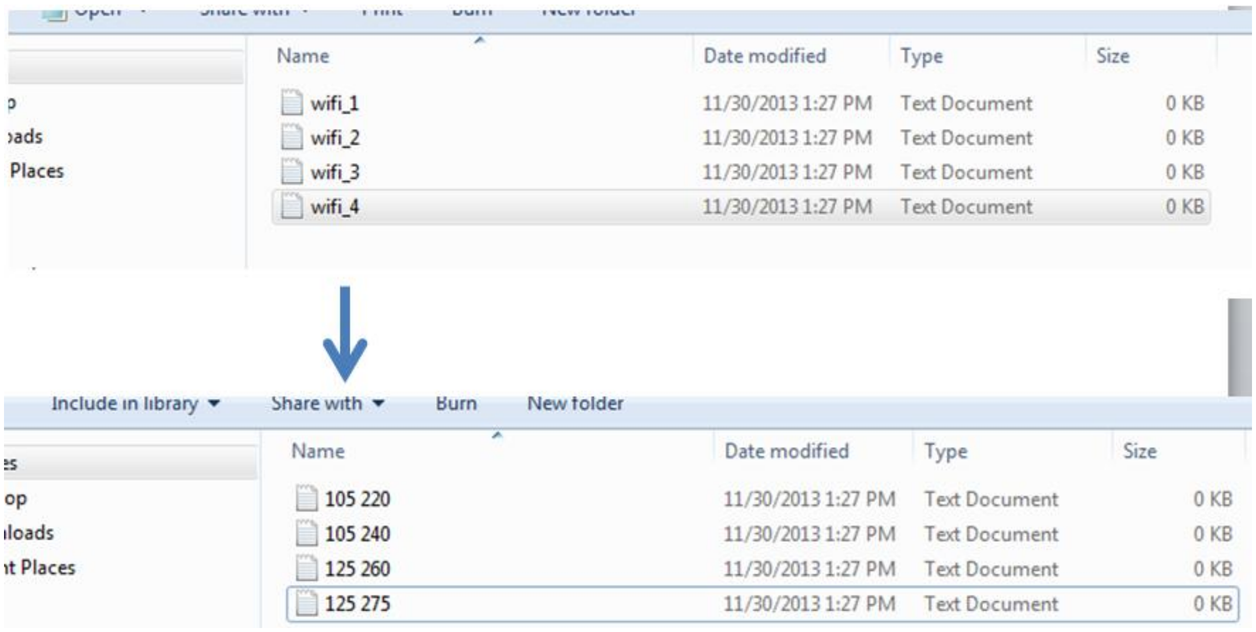


Figure 5: Locations within ITB where wifi data was collected.

Each wifi fingerprint location has a corresponding x, y coordinate, which represents where to draw the point on the map representing the user's location. Each wifi fingerprint is stored in a separate file and labeled with the appropriate x, y coordinates, so that when the wifi algorithm is running and finds a match, it can read the file name and use those two coordinates as the location to draw the user's location. An example of modifying the collected wifi fingerprints to their physical x, y coordinates is seen in Figure 6.

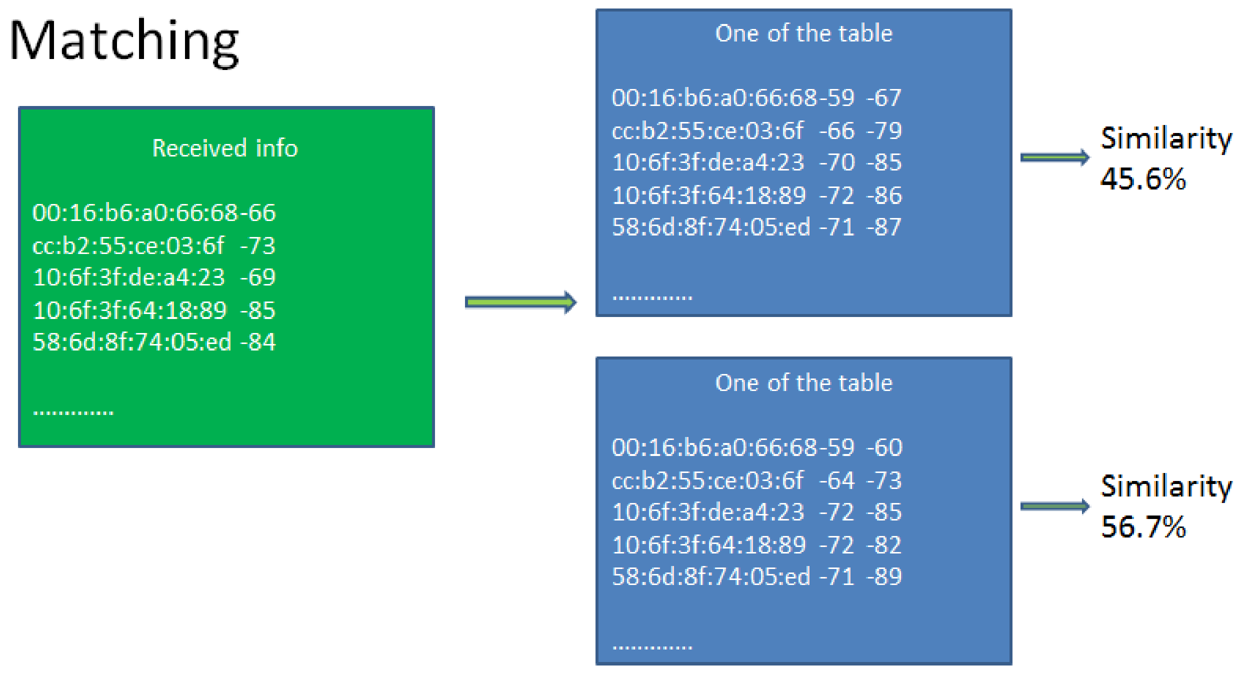


Name	Date modified	Type	Size
wifi_1	11/30/2013 1:27 PM	Text Document	0 KB
wifi_2	11/30/2013 1:27 PM	Text Document	0 KB
wifi_3	11/30/2013 1:27 PM	Text Document	0 KB
wifi_4	11/30/2013 1:27 PM	Text Document	0 KB

Name	Date modified	Type	Size
105 220	11/30/2013 1:27 PM	Text Document	0 KB
105 240	11/30/2013 1:27 PM	Text Document	0 KB
125 260	11/30/2013 1:27 PM	Text Document	0 KB
125 275	11/30/2013 1:27 PM	Text Document	0 KB

Figure 6: Modifying collected wifi fingerprints to the corresponding x, y physical map coordinates.

With all of the wifi fingerprints collected and modified, the wifi localization executes by continuously collecting data and attempting to match the received BSSIDs and RSS to a fingerprint in the table (Figure 7). The fingerprint with the highest number of matched BSSIDs and RSS which fall within the RSS range is considered the current location of the user, and is displayed by a dot on the map.



Received info:

```
00:16:b6:a0:66:68-66
cc:b2:55:ce:03:6f  -73
10:6f:3f:de:a4:23  -69
10:6f:3f:64:18:89  -85
58:6d:8f:74:05:ed -84

............
```

One of the table:

```
00:16:b6:a0:66:68-59  -67
cc:b2:55:ce:03:6f  -66  -79
10:6f:3f:de:a4:23  -70  -85
10:6f:3f:64:18:89  -72  -86
58:6d:8f:74:05:ed -71  -87

............
```

One of the table:

```
00:16:b6:a0:66:68-59  -60
cc:b2:55:ce:03:6f  -64  -73
10:6f:3f:de:a4:23  -72  -85
10:6f:3f:64:18:89  -72  -82
58:6d:8f:74:05:ed -71  -89

............
```

Figure 7: Example of received wifi data and trying to match to fingerprint in table.

Testing

Only the wifi algorithm was tested as the step counter suffered from high inaccuracy, partially attributed to the electromagnetic interference inside ITB. 3 testing runs were conducted and the results can be seen in the table 1 below.

Table 1: Test Results

	Correct Location	Incorrect Location	Accuracy
Slow Walk	29	15	65%
Fast Walk	23	21	52%
Walk and Stop	31	13	70%

Three different test configuration were completed. The first was walking at a slow pace, the second was at a faster pace and third had the tester walk from wifi fingerprint to fingerprint and briefly pause in an attempt to achieve a better signal. The walk and stop test run achieved the highest accuracy, this can be attributed to the wifi signals being able to "settle", compared to the other test runs having the phone moving and the RSS always changing.

Weakness/Improvements

The step algorithm's accuracy suffered due to interference encountered in ITB. Some areas of the building have significantly higher amounts of electromagnetic interference and this caused the compass to report inaccurate and erratic readings. Adding information from the gyroscope sensor could be a possible method to mitigate the interference, as you could determine if the user is actually turning the phone and changing their heading, or if no change in the gyroscope readings are detected, you could attribute the change to interference and ignore the compass.

The wifi algorithm worked much better than the step but still has potential for improvement. Currently the wifi algorithm only detects location based on matching BSSIDs and RSS range; if two locations have very similar fingerprints, inaccurate locations can be reports. It would be possible to develop some additional elements to complement to the fingerprint, such as determining the distance between the previous known location and the next fingerprint location, and determining if the user has traversed the required distance, using the step algorithm, to arrive at the next computed location. An example can be seen below in Figure 8.

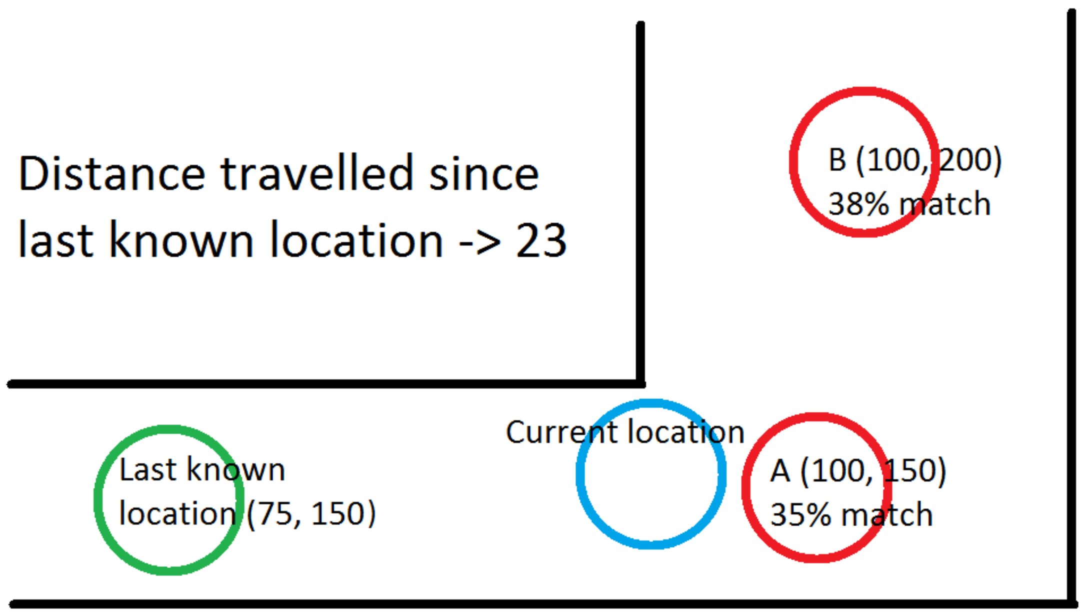

Distance travelled since last known location -> 23
B (100, 200)
38% match
Current location
Last known location (75, 150)
A (100, 150)
35% match

Figure 8: Improvement to wifi fingerprint using last known location and distance travelled from step algorithm.

In Figure 8, an example where the current location is close to a wifi fingerprint with a slightly lower match than a fingerprint further away is shown. As the wifi algorithm currently exists, the app would print on the map that the user is at location B, even though we can clearly see they are much closer to location A. Instead of only considering the wifi fingerprint match, we could use the step algorithm and record the distance travelled since the last known position and compare it to the difference in distance between the highest matched fingerprint locations indicating the next location. In this example, the distance between the last known location and B is 55.9, while the distance between A and the last known location is 25. If step algorithm reports that the user has travelled 23, then we could set some kind of threshold, arbitrarily 5 for this example, and say that if the difference between two fingerprints competing for the next current location, A & B in this example, of the user are within the threshold, then compare the distance travelled with the distance from the last known location and the two competing locations and choose the one with the lowest difference. With this additional distance factor, A would be reported as the true next location even though it has a lower wifi fingerprint match.

References

[1] A. Jim´enez, F. Seco, C. Prieto, and J. Guevara. A comparison of pedestrian dead-reckoning algorithms using a low-cost MEMS IMU. In Intelligent Signal Processing, IEEE International Symposium on, pages 37–42, 2009.