Implementing WiFi-based Localization App

CSE465 – Mobile Computing assignment

2025.04.30

Student name : 홍대의
Student No. : 20201325

INDEX

0. Github Repository
1. Introduction
2. Development Environment
3. App Features & UI
   3.1. Upload/Delete Image
   3.2. Wardriving mode
   3.3. Real-time localization mode
   3.4. Saving data in .csv format and export by email
4. Localization Algorithm
5. Experiment Details
   5.1. Wardriving data collecting method
   5.2. Static/Moving Localization
   5.3. Make ground truth point
6. Experiment Result Analysis
   6.1. Wardriving collected data analysis
   6.2. Static/Moving localization performance comparison
7. Conclusion

0. Github Repository

All related files are also located in github repository
: https://github.com/HongDay/simple_localization_app
: source code, collected data, experimented video, captured screen, etc.

1. Introduction

A large number of wireless signal data exist within a building. In indoor and localized environments such as buildings, signal-based localization can sometimes outperform GPS-based localization in terms of accuracy. The unique types and strengths of signals received at each location within such environments are referred to as the "radio fingerprint," indicating that the signal information is distinct for each position.

This project aims to develop a mobile application that localizes a device's position based on collected Wi-Fi signal data (RSSI, BSSID) from the 1st and 2nd floors of Engineering building 106, UNIST.

This report provides a detailed description of each function of the application, the implementation approach, the operation of key features, analysis of the collected data, and an evaluation of the localization accuracy achieved by the developed application.

2. Development Environment

- Used language : Java
- Android Studio : Meerkat – 2024.3.1 Patch2
- Test device :
- The app requests the following additional permissions :
  - ACCESS_FINE_LOCATION, ACCESS_COARSE_LOCATION (for Wi-Fi-based location tracking)
  - ACCESS_WIFI_STATE, CHANGE_WIFI_STATE (to access and control Wi-Fi state)
  - INTERNET, ACCESS_NETWORK_STATE (to use network-related functionalities)
  - READ_EXTERNAL_STORAGE, WRITE_EXTERNAL_STORAGE (to access external storage)
- Note:
  Starting from Android 6.0 (API 23), location permission is required for Wi-Fi scans.
  Starting from Android 9.0, Wi-Fi scan frequency is limited to 4 times per minute.
  From Android 10 (API 29) and above, storage permission policies have been changed.

3. App Features & UI

In this section, I introduce the 4 main functions of my app and its implementation and working details. And I provide screenshots in sequence to illustrate the flow of the app's GUI during the execution of each feature.

3.1. Upload/Delete Image

Button 'Upload map' is implemented. It needs 'EXTERNEL_CONTENT_URI' permission. If user press this button, user can access to its photo gallery memory and choose one photo. After photo is uploaded, user can check the photo in the screen. 'Upload map' button turns into 'Delete map' button and 'Wardriving' button come out.

Button 'Delete map' terminate all the wardriving, localization process and delete all the scanned data stored in the app, and finally delete the images and dots on the image. It makes the 'Upload map' button come out again.

Figure 1. Upload map screen

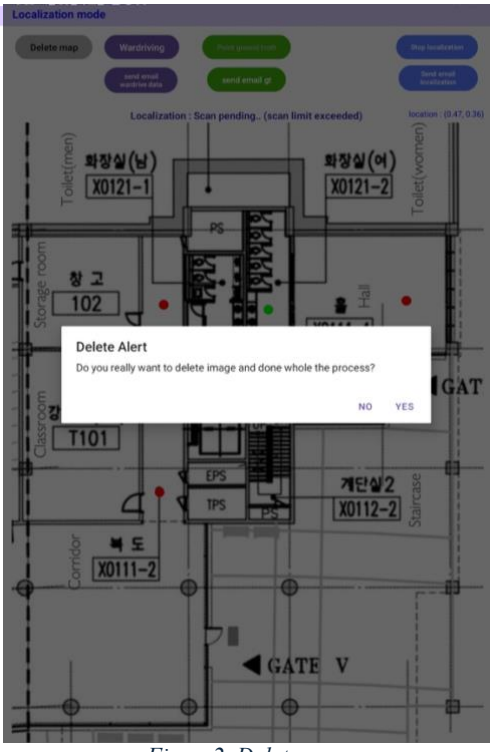

Figure2. Delete map screen

3.2. Wardriving mode

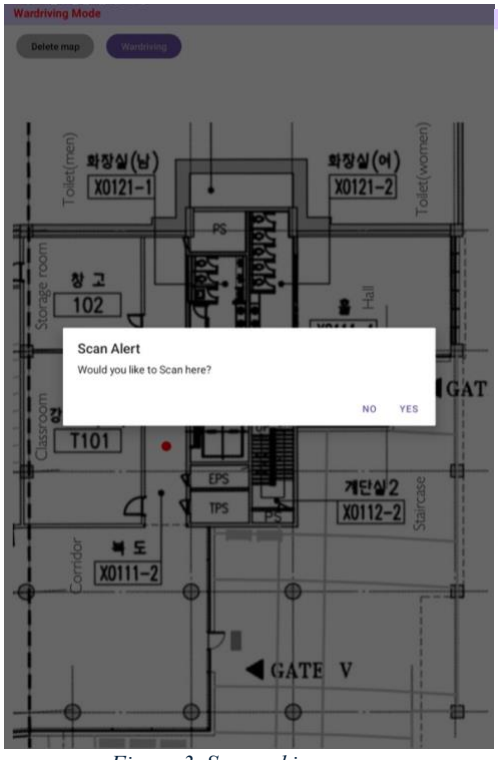

Figure 3. Scan asking screen

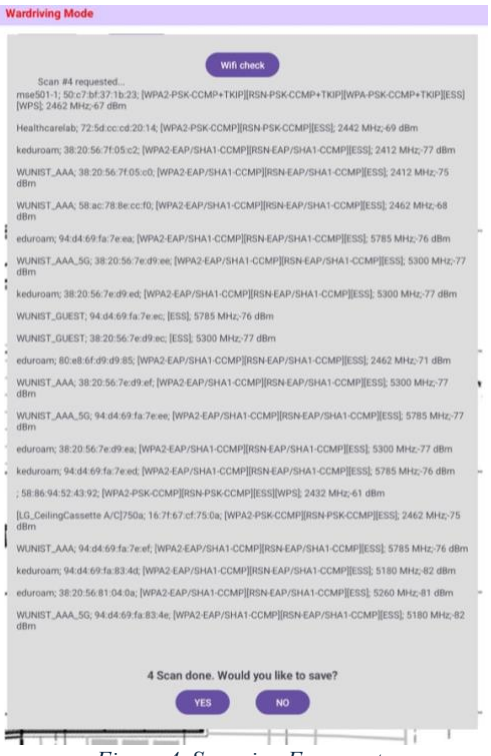

Figure 4. Scanning Fragment

When the 'Wardriving' button is pressed, the wardriving mode is initiated. In this mode, tapping a location on the map displays a red dot and triggers a scan alert popup asking whether to begin scanning. If 'Yes' is selected, a scanning fragment is launched; if 'No' is selected, the red dot is removed. In the scanning fragment, pressing the 'Wi-Fi Check' button starts the scan, displaying a list of detected Wi-Fi BSSID and RSSI values. Scan can be done only 4times in 1minute because of

the android policy, so I set the 6 second gap for each scan. If the user selects 'Yes' under the "Would you like to save?" prompt, the red dot's *(x, y) map coordinates* along with *timestamp, BSSID, and RSSI* data are saved. Selecting 'No' discards both the red dot and the collected data.

Figure 5. alert for existing point

Figure 6. Adding new scan data

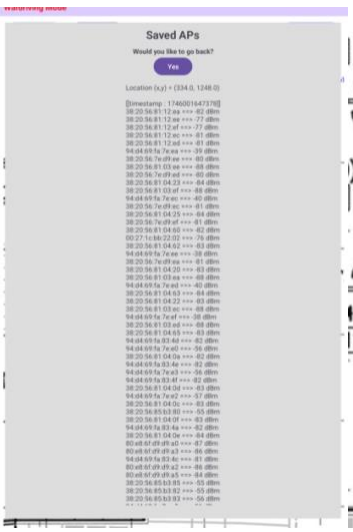

Figure 7. See existing data

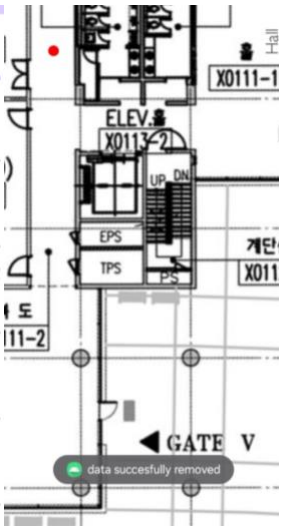

Figure 8. Deleted existing data

Additional features are provided for points that already have scan history. When a red dot (with existing scan data) on the map is tapped again, a popup appears with three action options: *Add New Scan Data*, *Delete the Data*, and *See the Data*.

- Selecting *Add New Scan Data* appends a new timestamp and the Wi-Fi scan data collected at that moment to the existing point.
- Selecting *Delete the Data* removes both the coordinate and all associated Wi-Fi data.
- Selecting *See the Data* allows the user to view the stored (x, y) coordinates, timestamps, and Wi-Fi scan data within the app.

3.3. Real-time localization mode

When at least one wardriving point is created, the 'Point Ground Truth' and 'Localization' buttons become available. Upon clicking the 'Localization' button, real-time Wi-Fi scanning begins, and localization is performed based on the implemented algorithm (detailed in the next section). The estimated location is displayed as a blue dot on the map, and each time a point is shown, the estimated (x, y) coordinates, timestamp, and the Wi-Fi data used for localization at that moment are stored in the app.

Although real-time localization is required, Android policy limits Wi-Fi scanning to four times per minute. Therefore, localization is performed up to four times at 6-second intervals, followed by a 40-second pending period to avoid failed scans. (Forcibly triggering a scan during this time returns only the most recent result.)

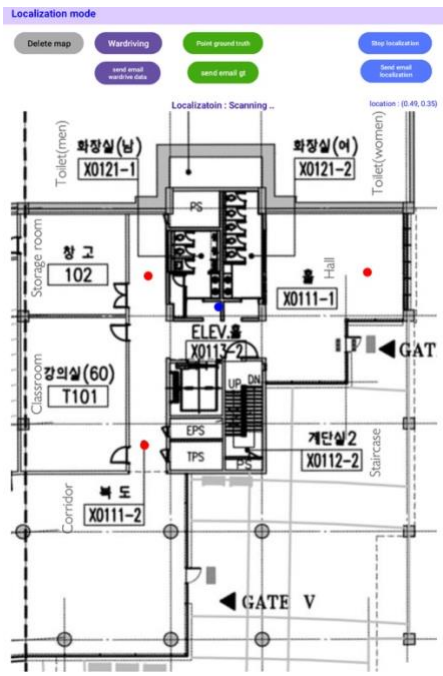

Figure 9. Localization : scanning

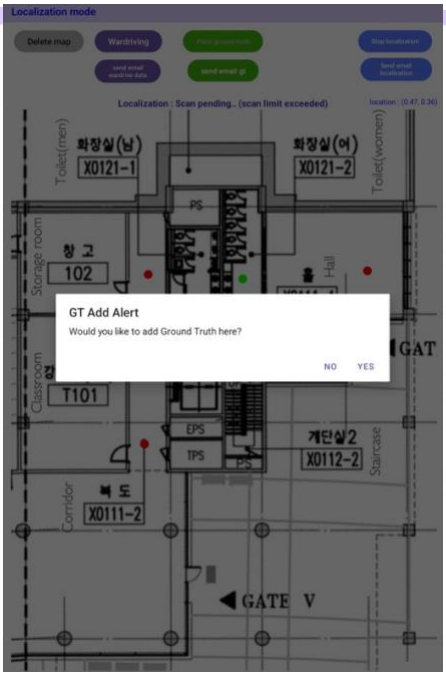

Figure 10. Make ground truth point

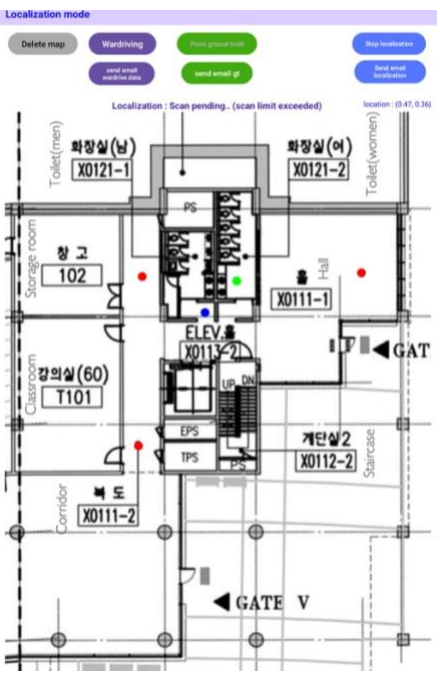

Figure 11. Localization : pending

To support future performance evaluation and data analysis, a feature to mark ground truth points was also implemented. When the 'Point Ground Truth' button is pressed, the user can place a green dot on the map, which is saved along with its actual (x, y) coordinates and timestamp. This allows users to mark their actual position just before starting localization.

3.4. Saving data in .csv format and Exporting by email

Figure 12. Pressed email button

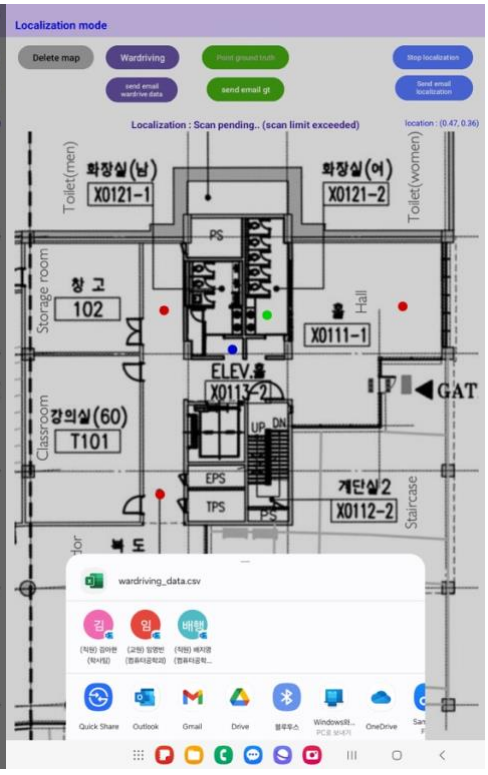

Figure 13. Email sending popup

If at least one data point exists for wardriving, ground truth, or localization, the corresponding buttons *Send Email Wardriving Data*, *Send Email GT Data*, and *Send Email Localization Data* are activated. When each button is pressed, the app first saves the relevant data as a `.csv` file.

Wardriving data, localization data, and ground truth data are internally managed using custom data class structures and stored in ViewModels within the app (detailed in Section 6.1.1). These stored data are then converted into CSV format and saved. Once the export is complete, an email sharing popup is triggered, allowing the user to send the data via email.

4. Localization Algorithm

In real-time localization, the app estimates the user's position by comparing the wardriving data with the Wi-Fi scan data being collected in real time. This section explains what algorithm in this app use for localization. The main goal is to estimate the current (x,y) coordinate on the map image view. Based on the previously collected Wi-Fi BSSID and RSSI information for each (x, y) coordinate, the app compares it with the current BSSID and RSSI values from real-time scans.

```
Private Map<String, Double> getAverageRssi(MeasurePoint point){}
```

For every (x,y) coordinate in wardriving data, it has several timestamp and each timestamp contains (BSSID->RSSI) map. The function iterates over all these timestamps and collects RSSI values for each BSSID. Then, for each BSSID, it calculates the average RSSI across all timestamps.
The result is a map where each BSSID is associated with its average RSSI at that locatçion

```
Private double computeDistance(Map<String, Integer> curScan, MeasurePoint point)
```

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}.$$

Euclidean distance

This function calculates the RSSI-based Euclidean distance between the current scan (represented as a BSSID-to-RSSI map) and a given wardriving point. For each BSSID in the current scan, if the same BSSID exists in the wardriving point, the difference in RSSI values is included in the Euclidean distance calculation:

$$distance += (rssi_{current} - rssi_{wardriving})^2$$

If a BSSID in the current scan does not exist in the wardriving point, That point might be far from the current location. So I thought just ignoring that RSSI is not proper to compute the distance. Therefore I compute the distance assuming the wardriving RSSI is 0 :

$$distance += (rssi_{current} - 0)^2$$

This approach reflects the idea that missing BSSID signals at a wardriving point may indicate dissimilarity, and therefore should still contribute to the overall distance.

```
Public float[] estsimateLocation(Map<String, Integer> curScan, List<MeasuerPoint> poitnList)
```

This is a localization function that takes the current (BSSID → RSSI) map collected in real time and the list of all stored wardriving points as input, and estimates the user's current (x, y) coordinate.

1. First, it iterates through all wardriving points and computes the Euclidean distance between each point and the current scan using the computeDistance() function. The distances are stored in a list.

2. The list is then sorted in ascending order, and the three closest points are selected.

3. A weighted k-Nearest Neighbors (kNN) algorithm is applied with k = 3 :
   The final (x, y) coordinate is calculated by applying weights based on the inverse of the Euclidean distances for each of the three points.

Weighted kNN regresssion Formula for Localization

- $k = 3$ (number of nearest wardriving point)
- d_i : the Euclidean distance between current scan and i^{th}nearest wardriving point
- (x_i, y_i) be the coordinate of the i^{th} wardriving point

- Weight Calculation

$$w_i = \frac{1}{d_i^2}$$

- Final Estimated Coordinate

$$x = \frac{\sum_{i=1}^{k} w_i \cdot x_i}{\sum_{i=1}^{k} w_i} \quad , \quad y = \frac{\sum_{i=1}^{k} w_i \cdot y_i}{\sum_{i=1}^{k} w_i}$$

5. Experiment Details

This section describes the methods attempted for collecting wardriving data, the experimental procedures for localization in both static and moving scenarios, the number of measurements conducted, and the collection process of localization data along with the corresponding ground truth points used to evaluate localization performance.

5.1. Wardriving data collecting method

I collected 3 kinds of wardriving data
- 2nd Floor ver1. wardriving
- 2nd Floor ver2. wardriving
- 1st Floor ver3. wardriving

Ver1 was the first wardriving method I implemented, and its characteristics are as follows:
1) Wardriving was conducted mainly along the edges of the hallway.
2) Signal data was collected at a total of 30 points.
3) At each point, four scans were performed at 5-second intervals.
My initial reasoning was that since I planned to use a weighted k-Nearest Neighbors (kNN) algorithm for localization, collecting wardriving points mainly along the edges would lead to higher localization accuracy when the device was positioned inside the hallway. I believed that having more reference points would increase accuracy, so I collected data at as many as 30 points.
However, due to Android's security policy, the WifiManager limits Wi-Fi scans to a maximum of four times per minute. Therefore, for each point, I performed four scans at 5-second intervals and saved the scan data accordingly.

Figure 14. Floor2 Ver1 wardriving point

Figure 15. Floor2 Ver2 wardriving point

Despite this, the performance of Ver1 was not as good as expected (specific performance issues will be analyzed in detail with data in Section 6), so I decided to conduct a new experiment with Ver2. The key features of **Ver2** are as follows:

1) Wardriving was performed mainly along the center of the hallway.
2) Signal data was collected at a total of 20 points.
3) Two scans were performed at each point at 5-second intervals,
and after completing scans for all points, two additional scans were performed at each point again.

Considering that users are more likely to be located in the center of the hallway rather than at the edges, I expected that having wardriving points that more closely resemble the actual user position would improve localization performance. Also, I reduced the number of points to 20, since too many wardriving points could increase the chance of incorrect predictions due to real-time signal noise during localization. Additionally, since Wi-Fi signals can slightly change over time, I did not conduct all four scans at once; instead, I split them into two rounds of two scans each.

I also applied **Ver3 to the 1st floor** for further analysis, I tried to scan with Ver2, but because of the time constraints (since only four scans are allowed per minute, time takes a lot), I collected only two scan results per point without time difference for the 1st floor. The only difference between Ver2 and Ver3 is : *3) scan only first 2 scan for each point at 5-second intervals.*

5.2. Static/Moving Localization

For 3 kinds of wardriving data, I conducted the localization experiment using the corresponding wardriving data for each case. Using the Localization algorithm introduced in section4, I experiment for static state and moving state analysis for each wardriving data.

Since the maximum wifi scanning is only 4 times in 1 minute, I set the 6 sec delay between each scan. After 4 scan is done, wait about 40 sec for next localization. Therefore, continuous real-time localization was able to tested with only 4 scan.

Static state localization:

1) Test for 7 ground truth points
2) For each point, conduct 4 times of scan
3) Save the predicted (x,y) coordinate, timestamp, and point the ground truth

Moving state localization:

1) Test for 4 path (through the hallway)
: One path, there is 4 ground truth point and 4 localization estimated point.

2) For each **path**, conduct 4 times of scan
3) Save the predicted (x,y) coordinate, timestamp, and point the ground truth

5.3. Make ground truth point

Right before scanning, I pointed the ground truth on the map imageview so that I can compare the ground truth point with localization point and use in analysis

Static state ground truth:
Since it 4 times localization was conducted per 1 point,
1 ground truth point has 4 localization estimated point.

Moving state ground truth:
Since 4 scan localization was conducted per 1 path, which has 4 point.
1 ground truth point has 1 localization estimated point in a path.
I pointed ground truth right before each localization scanning is done.

6. Experiment Result Analysis

This section analyzes the Wi-Fi data collected through wardriving and evaluates the localization performance using the localization data and ground truth data

6.1. Wardriving collected data analysis

6.1.1. Data structure

Wardriving View Model
- Measure point 1
    - (x,y) coordinate
    - Timestamp1
        - BSSID : RSSI
        - BSSID : RSSI
        - …
    - Timestamp2
        - BSSID : RSSI
        - BSSID : RSSI
        - …
    - …
- Measure point 2

  …

Wardriving view model has several measure point.
Each measure point has 1 coordinate and Timestamp list.
Each Timestamp list contains 1 timestamp and (BSSID->RSSI) map

6.1.2. Analysis Metric

I evaluated the signal stability at each location by measuring how much the types of BSSIDs vary within the same MeasurePoint and how much the RSSI values fluctuate for the same BSSID.
To assess the signal uniqueness of each location, we also analyzed how many unique BSSIDs exist between different MeasurePoints and, in cases where the same BSSID appears across points, how much their RSSI values differ.

1) Average Standared Deviation of BSSID Count (Same x,y)

To assess the stability of the number of Wi-Fi access points observed at the same location over time, we calculated the standard deviation of unique BSSID counts per timestamp at each (x, y) coordinate.

- $B_{xy}^{(t)}$: *number of unique BSSIDs detected at coordinate* (x, y) *in timestemp* t

$$\sigma_{BSSID}^{(x,y)} = \mathrm{std}\left(\left\{B_{xy}^{(t)}\right\}_{t=1}^{T}\right) \qquad \frac{1}{N}\sum_{(x,y)}\sigma_{BSSID}^{(x,y)}$$

2) Average RSSI Standared Deviation (Same x,y)

To measure the fluctuation of signal strength at a fixed location, the standard deviation of RSSI values was computed for each BSSID across timestamps, and then averaged over all BSSIDs at that point.

$$\sigma_{RSSI}^{(x,y,b)} = \mathrm{std}\left(\left\{RSSI_{xy,b}^{(t)}\right\}_{t=1}^{T}\right) \qquad \mu_{RSSI}^{(x,y)} = \frac{1}{|B|}\sum_{b\in B}\sigma_{RSSI}^{(x,y,b)} \qquad \frac{1}{N}\sum_{(x,y)}\mu_{RSSI}^{(x,y)}$$

3) Average BSSID Jaccard Similarity (different x,y)

To evaluate how similar the sets of detected BSSIDs are between two different coordinates, the **Jaccard Similarity** is used:

$$J(c_1, c_2) = \frac{|B_{c_1} \cap B_{c_2}|}{|B_{c_1} \cup B_{c_2}|}$$

- B_c : *set of BSSIDs observed at coordinate* c

The final metric is the average Jaccard index over all unique pairs (c1, c2)

4) Average RSSI Difference on Common BSSID (Between different x,y)

For each pair of coordinates (c1,c2)(c1,c2), we extract common BSSIDs and compute the mean absolute difference in average RSSI:

$$\text{RSSI diff}_b = |RSSI_{c_1,b} - RSSI_{c_2,b}| \qquad \text{Average RSSI diff}_{c_1,c_2} = \frac{1}{|B_{common}|}\sum_{b\in B_{common}} \text{RSSI diff}_b$$

The final metric is the average of $Average\ RSSI\ diff_{c1,c2}$ over all unique pairs (c1, c2)

6.1.3. Data Analysis

Metric 1 : Average Standard Deviation of BSSID Count (Same x,y)
Metric 2 : Average RSSI Standard Deviation (Same x,y)
Metric 3 : Average BSSID Jaccard similarity (different x,y)
Metric 4 : Average RSSI Difference on Common BSSID (different x,y)

	Metric 1	Metric 2	Metric 3	Metric 4
Floor2 – Ver1	6.196	2.396	0.370	12.256
Floor2 – Ver2	10.174	3.230	0.407	11.660
Floor1 – Ver3	6.029	2.555	0.403	9.473

Table 1. Wardriving data analysis with 4 metrics

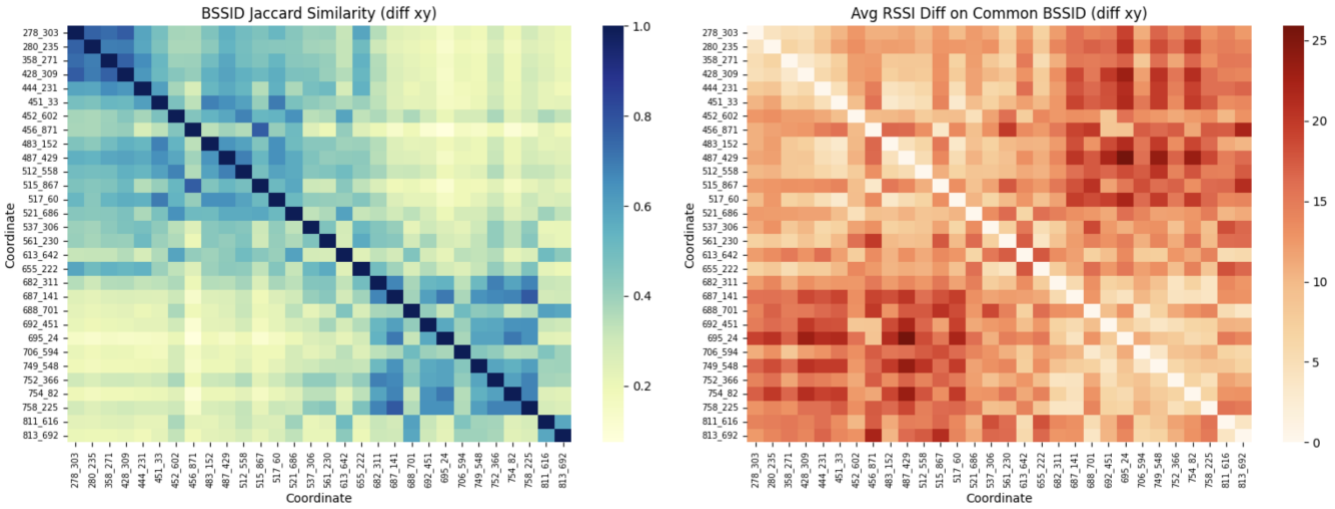

Figure 16. Heatmap for BSSID Jaccard similarity and RSSI Diff – Floor2 Ver1.

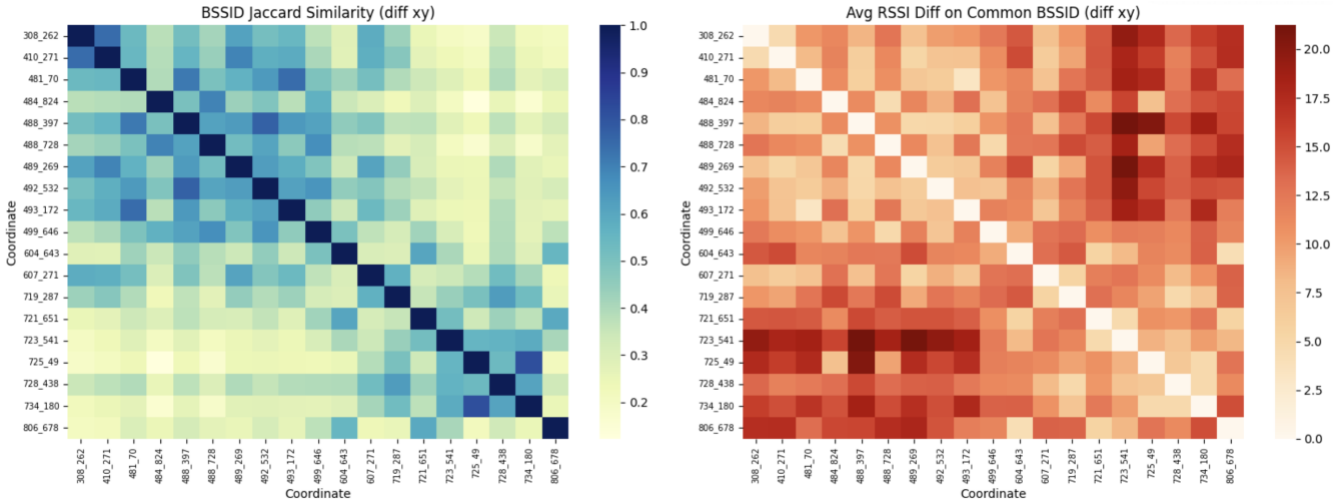

Figure 17. Heatmap for BSSID Jaccard similarity and RSSI Diff – Floor2 Ver2.

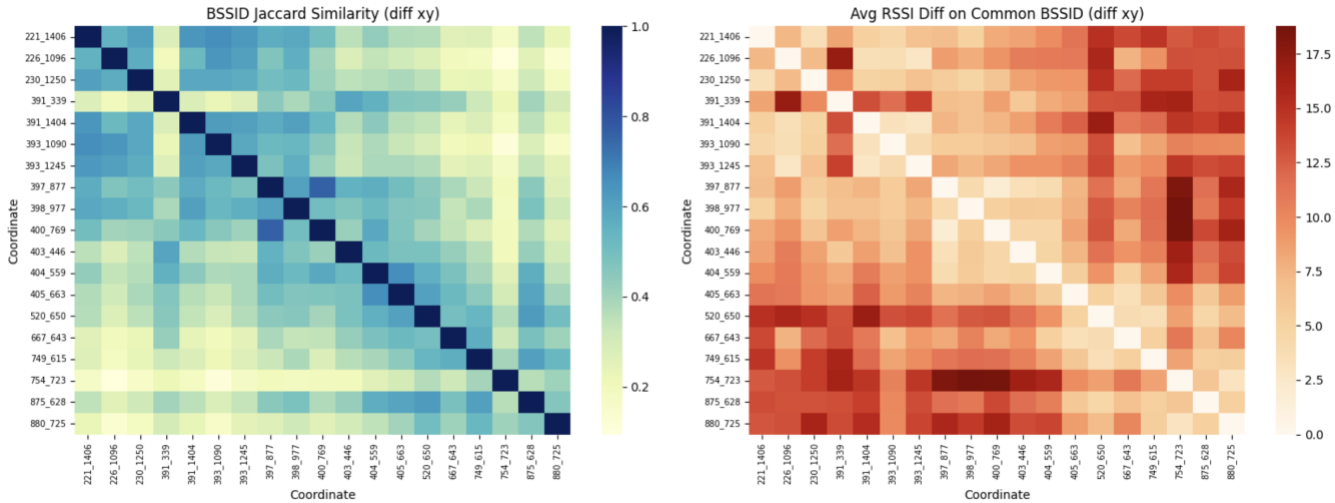

Figure 18. Heatmap for BSSID Jaccard similarity and RSSI Diff – Floor1 Ver3.

For Metric1 and Metric2:

comparing Floor2-ver1 and Floor2-ver2, the number of BSSID counts within the same (x, y) coordinates differs by approximately 4, and the RSSI standard deviation differs by about 0.85.

The key difference between Floor2-ver1 and ver2 lies in the scanning method: ver1 performed four scans at once per point, while ver2 conducted two scans at different times. This difference in measurement timing resulted in variations in both the number of BSSIDs and the RSSI strength, indicating that radio signals can change slightly over time and may become unstable depending on the time of day. Since ver2 captures these time-based variations in signal information, it can be considered to have more generalized data. This may lead to more accurate localization in the future when averaged values are used for localization.

In the case of Floor1, two scans were conducted per point without time differences, showing similar BSSID counts and RSSI deviation as in ver1.

For Metric3 and Metric4:

If the BSSID Jaccard similarity between different (x, y) points is too high, or the average RSSI difference is too low, it suggests that the signal strength or types of signals do not vary significantly across different locations. In such cases, it becomes difficult to identify the correct location using localization.

When comparing Metric 4 with Metric 1, the differences in Floor2-ver1 and Floor2-ver2 are approximately 10 and 8.5, respectively, indicating clear distinguishability between locations. However, in Floor1-ver3, the difference is around 7, which is relatively lower and may lead to reduced localization accuracy.

The heatmaps show similar trends across all three cases: nearby points tend to have higher Jaccard similarity (darker colors), while distant points have lower similarity. In Floor1-ver3, there are more high-similarity areas along the diagonal compared to the others, which may also contribute to lower localization performance.

6.2. Static/Moving localization performance comparison

6.2.1. Data Structure

Localization View Model

- Estimate point 1
    - (x,y) coordinate
    - Timestamp1
        - BSSID : RSSI
        - BSSID : RSSI
        - …
- Estimate point 2

  …

Ground Truth View Model

- Ground truth point 1
    - (x,y) coordinate
    - Timestamp1
- Ground truth point 2

  …

6.2.2. Analysis Metric

For static localization :

There's 4 localization point for 1 ground truth location. For each timestamp, I paired each ground truth point to corresponding 4 localization points. By comparing (x,y) coordinates, the performance can be evaluated by RMSE (Root Mean Squared Error) and Average pairwise distance.

$$\text{RMSE}_i = \sqrt{\frac{1}{4}\sum_{j=1}^{4}\|\mathbf{p}_{ij} - \mathbf{g}_i\|^2} \qquad \text{Mean Dispersion}_i = \frac{1}{6}\sum_{1\le j<k\le 4}\|\mathbf{p}_{ij} - \mathbf{p}_{ik}\|$$

- ‖ ‖ denotes euclidean distance, and 6 is 4C2.

For moving localization :
There's 1 localization point for 1 ground truth location. So I could simply evaluate the localization performance by calculate RMSE for whole points.

$$\text{RMSE}_{\text{moving}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\|\mathbf{p}_i - \mathbf{g}_i\|^2}$$

6.2.3. Data Analysis

All units shown in the table are in dp, representing the (x, y) dimensions of the activity screen.

- Max (x, y) of the 1st floor: (1160dp, 1540dp), with a maximum diagonal distance of 1930dp
- Max (x, y) of the 2nd floor: (970dp, 1540dp), with a maximum diagonal distance of 1820dp

Point #	GT (x, y)	RMSE to GT	Mean Pairwise Distance
1	(404.00, 267.00)	69.509	66.433
2	(486.00, 731.00)	161.153	138.004
3	(611.00, 277.00)	45.748	46.335
4	(717.00, 100.00)	59.371	69.558
5	(752.00, 648.00)	27.194	21.832
6	(728.00, 417.00)	73.704	62.583
7	(477.00, 82.00)	177.105	190.526
Avg		87.683	85.039

Table 2. Static Localization Performance – Floor2 Ver1.

	RMSE to GT
Moving localization point avg	126.418

Table 3. Moving Localization Performance – Floor2 Ver1.

Point #	GT (x, y)	RMSE to GT	Mean Pairwise Distance
1	(395.00, 272.00)	80.363	8.052
2	(727.00, 654.00)	29.532	5.223
3	(720.00, 155.00)	65.119	3.625
4	(578.00, 268.00)	30.272	34.547

5	(486.00, 130.00)	68.755	28.507
6	(486.00, 492.00)	63.343	88.358
7	(487.00, 790.00)	13.916	7.334
Avg		50.186	25.092

Table 4. Static Localization Performance – Floor2 Ver2.

	RMSE to GT
Moving localization point avg	133.228

Table 5. Moving Localization Performance – Floor2 Ver2.

Point #	GT (x, y)	RMSE to GT	Mean Pairwise Distance
1	(812.00, 664.00)	72.113	84.87
2	(478.00, 654.00)	72.092	47.364
3	(399.00, 466.00)	128.954	113.1
4	(394.00, 901.00)	115.788	111.267
5	(261.00, 1127.00)	115.487	14.406
6	(320.00, 1376.00)	121.696	17.317
Avg		104.355	64.721

Table 6. Static Localization Performance – Floor1 Ver3.

	RMSE to GT
Moving localization point avg	327.88

Table 7. Moving Localization Performance – Floor1 Ver3.

For **Floor 2 - Ver1**, the average RMSE of static localization is approximately **88dp**. Considering that the maximum diagonal length of the 2nd floor map is **1820dp**, this corresponds to roughly **5% of the total map size**, indicating relatively high accuracy.
In contrast, the average RMSE during moving localization is about **126dp**, which is larger than in the static case. This decrease in accuracy can be attributed to the instability of signal reception while user is moving.

For **Floor 2 - Ver2**, the average RMSE for static localization is around **50dp**, showing an improvement of about **38dp** compared to Ver1. This accuracy gain is likely due to the use of more generalized data—wardriving was conducted at multiple time intervals, and the averaged RSSI values were used for localization.
Moreover, since the wardriving points were collected mainly at the center of the hallway—where users are more likely to be located—rather than along the edges, the algorithm likely benefited from a closer match between training and actual user positions.
Additionally, the **Mean Pairwise Distance** (i.e., the average distance between repeated localization estimates for the same ground truth point) was significantly reduced, from **85dp in Ver1** to **25dp in Ver2**, an improvement of **60dp**. This suggests lower positional variance, likely due to the use of more stable and generalized wardriving data.

In the case of **Floor 1 - Ver3**, there was no temporal variation in wardriving, and only two scans were performed per point. Furthermore, the RSSI differences between points were relatively small, indicating

low spatial uniqueness. As a result, the RMSE in static localization increased, and the error in moving localization increased **significantly**, indicating degraded performance.

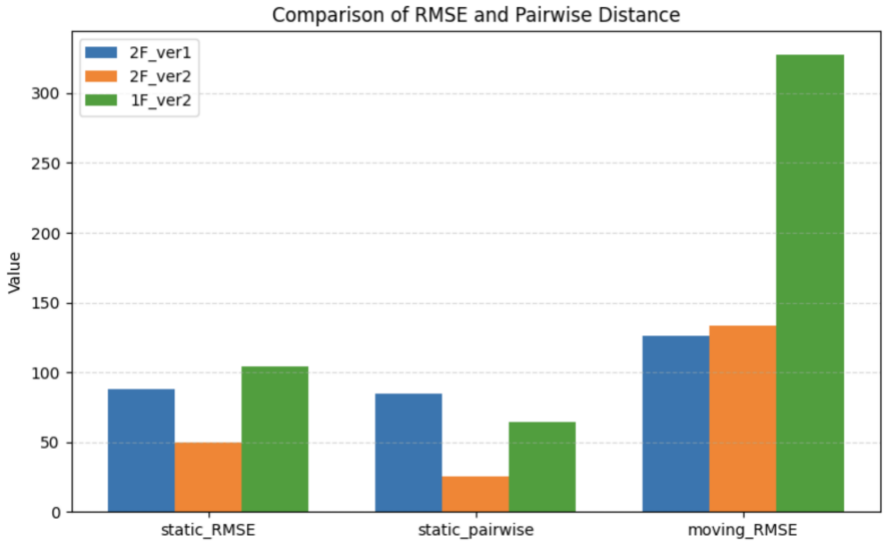

Chart 1. Comparison of RMSE and Pairwise distance

7. Conclusion

- By implementing all four core features—upload/delete, wardriving, real-time localization, and email sharing—I successfully developed a Wi-Fi-based localization app.

- However, due to Android's policy limiting Wi-Fi scans to a maximum of four times per minute, it was difficult to collect continuous data and verify real-time localization performance while the device was in motion.
  Despite this, using ground truth points allowed for effective analysis of the experimental results.

- By averaging wardriving data collected at different timestamps and applying a **weighted kNN regression**-based localization algorithm, a localization function with reasonably high accuracy was achieved.

- Wi-Fi signals within the building tend to fluctuate slightly over time, and the signal information was not entirely stable.

- Due to this variability, collecting a moderate number of wardriving points yielded better performance than collecting too many, as slight signal fluctuations can otherwise lead to misclassification of positions.

- Collecting signal data at a single time point proved less effective than performing scans at different times and averaging them; the latter approach produced significantly better accuracy.

- In areas where RSSI differences between locations were small or where signal reception was unstable, localization performance degraded noticeably. This was especially evident in moving localization, where accuracy dropped substantially.