Smartphone-based Indoor Localization

Weixiang Zheng
Department of Computer Science
Nanyang Technological University)
Singapore, Singapore
wzheng014@e.ntu.edu.sg

Haoyu Ren
Department of Computer Science
Nanyang Technological University)
Singapore, Singapore
renh0007@e.ntu.edu.sg

Mingyu Zheng
Department of Computer Science
Nanyang Technological University)
Singapore, Singapore
mzhang040@e.ntu.edu.sg

***Abstract*—This Report introduces how we visualize the Kaggle Indoor Location Competition dataset. This work a group work project done by Weixiang Zheng, Mingyu Zhang, Haoyu Ren.**

Keywords—Urban computing,Visualization, Data Analysis Data Visualization, Kaggle Indoor Location Competition, Waypoint Position, Magnetic Strength, 3-WIFI AP,Geospatial Analysis.

I. Introduction

In 21st century, people use Navigation apps such as Google Maps a lot. Although navigation to remote places can be convenient and improved by better routing algorithms, the indoor navigation remains an ongoing problem. The main cause is the imprecision of indoor location, and that makes the improvement of indoor navigation difficult. The outdoor location is normally determined by satellite timing services but indoor reception of satellite signal could be a big problem. To circumvent this issue, researchers have developed different kinds of algorithms for determining indoor locations, including but not limited to carrying external object with users and installing detecting sensors in the indoon environment.However, the advancement of technology now makes it easier and more versatile to track indoor location, one of them being using smart phones. We know smart phones have sensors, sensors generate a lot of data, and if those data could be analyzed using correct algorithms, it could be used to infer user locations and obtain even more information about the user. For this paper, we are introducing three different data visualization methods using real-world data obtained from Microsoft's Indoor Location Competition 2.0 on Kaggle.

In the next few pages, we will first introduce the data set in Section 2, from its size, its dimension, to its format, after you gain a basic knowledge of what data we are working on, we will show our work in Section 3. We are mainly using Python Programming Language and its plotting library Matplotlib to do the visualization work. The first part Section 3.1 is visualization of waypoints, in this section, we generate a trace map based on the position data in the data set. The second part Section 3.2 is visualization of geomagnetic heap map, in this section we show where the densest population is based on the way points position data. And the third part Section 3.3 is visualization of RSS heap maps of 3 WIFI Aps, in this section,we are using WIFI Aps to triangulate the dense population instead of using position data. Section 4 is the bonus tasks. Section 5 is the detailed group member contribution. Last but not least, we will attach the source code for our visualization in the Appendix page.

II. Dataset Overview

The sample dataset contains data from two different buildings. The data is collected by an Android smart phone held by a site-surveyor. When he walks from one point to another, sensor signals including accelerometer, geomagnetic field, gyroscope, rotation vector, WiFi, Bluetooth iBeacon and waypoint locations are recorded by a sensor data recording app.

Each of the site folder is organized by floor folders. Each floor folder contains indoor traces recorded in this floor in .txt format and a floor image. Approximately 110 to 160 traces are recorded on each floor in site1 and about 30 to 100 traces are recorded on each floor in site2. The total number of trace files in site1 is 642 and the number in site2 is 453. The description of a trace file format (*.txt) is as follows.

The first column is the time when each piece of data was recorded. The second column is the sensor data types which are shown below. The values of each row vary depending on data types.

S/N	Data Type	Values
1	TYPE_WAYPOINT	Px, Py
2	TYPE_ACCELEROMETER	X, Y, Z, accuracy
3	TYPE_GYROSCOPE	X, Y, Z, accuracy
4	TYPE_MAGNETIC_FIELD	X, Y, Z, accuracy
5	TYPE_ROTATION_VECTOR	X, Y, Z, accuracy
6	TYPE_ACCELEROMETER	X_b, Y_b, Z_b, X_a, Y_a, Z_a, accuracy
7	TYPE_GYROSCOPE_ UNCALIBRATED	X_b, Y_b, Z_b, X_a, Y_a, Z_a, accuracy
8	TYPE_MAGNETIC_FIELD_ UNCALIBRATED	X_b, Y_b, Z_b, X_a, Y_a, Z_a, accuracy
9	TYPE_WIFI	ssid, bbsid, RSSI, frequency, lastseen
10	TYPE_BEACON	UUID, MajorID, MinorID, Tx Power, RSSI, Distance, MAC, Unix Time

Column 3-5 of TYPE_ACCELEROMETER 、TYPE_MAGNETIC_FIELD 、TYPE_GYROSCOPE 、TYPE_ROTATION_VECTOR are X-axis, Y-axis and Z-axis coordinates of the sensor data. Column 6 is sensor accuracy. Additional dataset information is listed on the competition Github page.[1]

III. Essential Tasks

3.1 Visualization of Waypoints

In this section, our task is to visualize waypoints on a trace. The waypoint is defined by “a stoping place on a journey”.[1] Therefore, we can always assume there exist a trace to which that particular waypoint is belonging. Let take a look at our waypoint data format:

Timestamp	Waypoint_x	Waypoint_y
1573713056859.000000	46.302425	56.807760
1573713071518.000000	34.116898	47.213530

We can see from the example that an entry of the waypoints data consists of three parts: timestamp, the x coordinate of the waypoint, the y coordinate of the waypoint. Timestamp could be used to clarify the starting point and the end point, and the order of the rest points could be tracked by following the route from start position to end position. By knowing the x and y coordinates, our task is basically to draw some line plots. Here we use matplotlib to initialize a white grid board, use standard line plot procedure to generate the graph, and calibrate the range of x axis and y axis to the input arguments width_meter and height_meter. Finally we annotate the start position and the end position of the trace. Here's how the result look like:

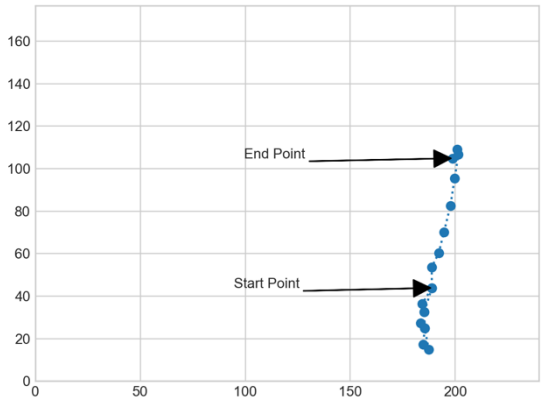

Fig. 1. Trace of 5dda5a99c5b77e0006b1770b, Site 1, F2

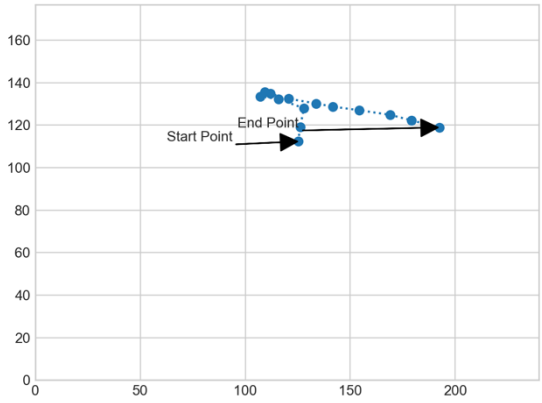

Fig. 2. Trace of 5dda402fc5b77e0006b176c1.txt, Site 1, F2

Next, we woud like to superimpose our trace onto the floor image. The important thing is that we need to calibrate our plot with the floor plan image, i.e. stretch them to the same width and the same height so that the trace does not drift from the original position, we could do that by setting up the xlim and ylim attribute of the image and the plot. Specifically, when doing the imshow to add the floor plan image, we specify the dimensions of the image to be loaded to the ax; when ploting the trace, we also specify the dimension of the plot. The final result should looks like this:

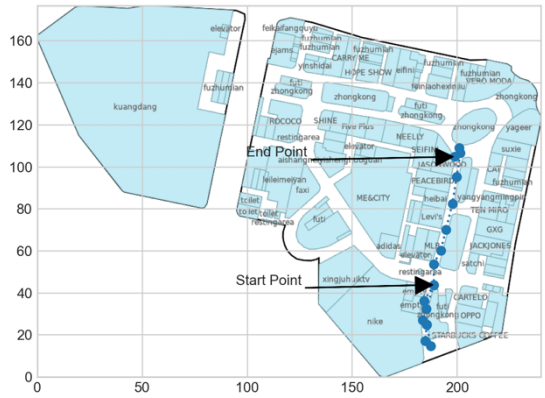

Fig. 3. After superimposing, 5dda5a99c5b77e0006b1770b, Site 1, F2

Fig. 4. After superimposing, 5dda402fc5b77e0006b176c1, Site 1, F2

The detailed implementation could be referred at the Appendix. Basically the visualization of waypoints consists of four steps:

1. Draw a line plot of the points,

2. Annotate the start point and the end point,

3. Calibriate both thte plot dimension and the dimension of loaded floor plan image to the same [xlim, ylim] range

4. Superimpose the plot onto the floor plan image.

3.2 Visualization of geomagnetic heat map

Magnetic Field: Any point in the building is affected by a unique magnetism. The visualization map of each floor is 2-dimensional. With adding magnetic magnitude, it will form a 3-dimensional magnetic heatmap of the floor. The magnetic magnitude of any point in space can be measured by reading the x, y, and z magnetic vectors of the point. Therefore, when the user moves, the phone detects the fluctuation of the magnetic field.

Mu Tesla is a derived unit of the magnetic field strength

MAGNETIC STRENGTH

Position_x	Position_y	Magnetic_strength
129.83163616	110.13035805	51.54282147
132.32097549	128.03953121	44.25965085

There are many issues that need to be paid attention to in the visualization process. First of all, we must choose the appropriate drawing tool library and related *api*. The heatmap is naturally thought to use the heatmap *api* to display, but through experiments, I found that it was not appropriate, because heatmap tends to fit the object that each point has a value. If using heatmap, it's need to set the point without magnetic field strength to 0. Such a large drawing space will take up a lot of memory, so I chose to use the scatter api in matplotlib in the end.

Secondly, we have to display the result in a proper picture ratio. If we used default value, the result will be too full of the whole picture. Therefore, after multiple comparisons, I set xlim and ylim in corporate with background and find a suitable figure size to display the result. What's more, setting the high resolution of the picture would help us to see the details. The upper operations help us to 1:1 present the magnetic field strength in the floor space.

Another problem is how to display the magnetic field intensity corresponding to each data point. I introduced the colorbar as the reference. The distributions of different colors reflect different magnetic field strengths. But how to choose the appropriate sidebar color mode to clearly reflect the intensity difference of each point? I tried different color modes such as *hsv*, *rainbow*, *CMRmap*. The final performance of *rainbow* is the most obvious and intuitive. In order to restore the sense of position, the introduction of coordinate grid lines is indispensable, but in order not to affect the visual effect, the transparency must be adjusted. Last but not least, it's necessary to introduce a title to indicate the object of the visualization. The above is the whole visualization steps.

The following four pictures show the magnetic strength of B1, F1, F2, F3 in site1.

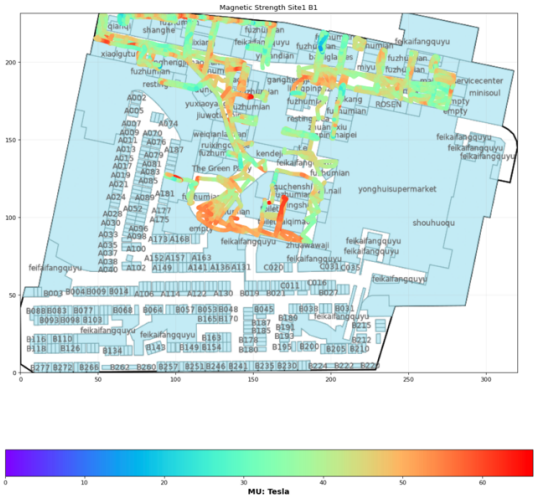

Fig. 5. Magnetic Strength in Site1 B1

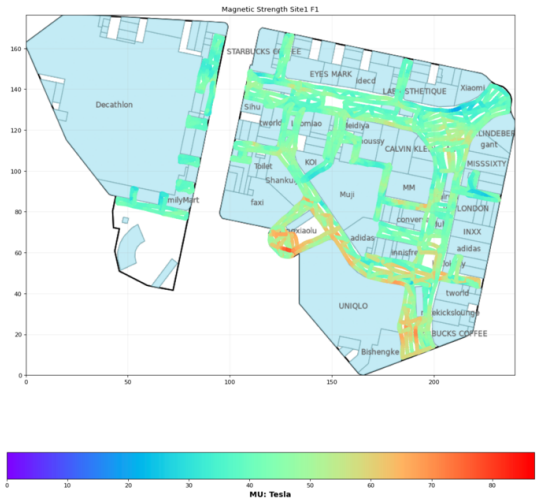

Fig. 6. Magnetic Strength in Site1 F1

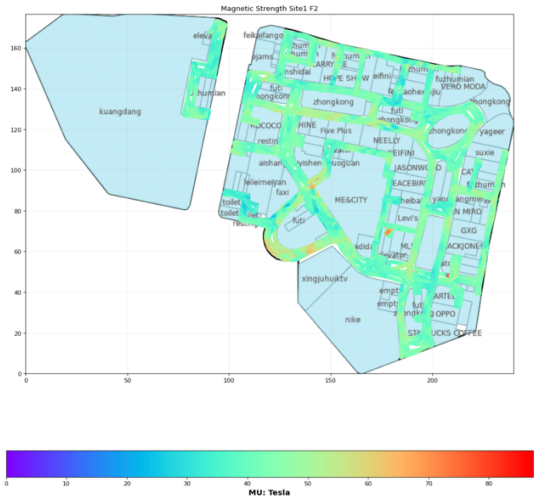

Fig. 7. Magnetic Strength in Site1 F2

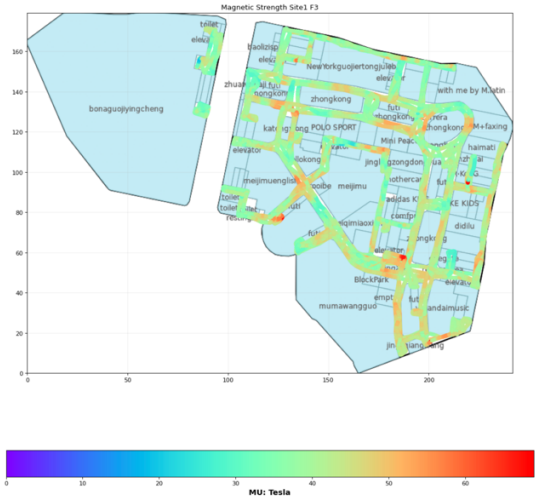

Fig. 8. Magnetic Strength in Site1 F3

The earth itself is a huge magnet, which forms a basic magnetic field between the north and south poles of the geography. But this kind of earth's magnetic field will be disturbed by metal objects, especially when passing through reinforced concrete structures, the original magnetic field is disturbed and distorted by the building materials (metal

structures), making each building have a unique magnetic field, which means that a regular "indoor magnetic field" is formed indoors.[2]

Moreover, if the building does not undergo structural changes, the characteristics of the indoor magnetic field will remain unchanged. The indoor geomagnetic positioning formally captures the regular characteristics of this "indoor geomagnetic field". The indoor magnetic field data is collected through the universally integrated geomagnetic sensor on the mobile phone to identify the difference in magnetic field signal strength at different locations in the indoor environment, so as to localize relative position in space.

Based on the above analysis, the greater the difference in magnetic field strength, the more helpful it is to accurately confirm the location of the user. It can be seen from the upper figures that the difference in magnetic strength of B1 and F3 is the largest, so when determining the location of these two floors, the magnetic strength localization method will be very effective, and F2 will be the smallest, and the effect of the magnetic strength localization method will be weakened.

3.3 Visualization of RSS heat map of three WiFi Aps

In this task, we decide to process data of floor 1 in site1. The first thing we need to do is to set the global variable 'floor_data_dir' to the file path of floor 1. Each file in the 'F1/path_data_files' folder is a trace recorded at floor 1.

Then we need to read all the data line by line and integrate different data into categories based on the type of data by using the function below:

def calibrate_magnetic_wifi_ibeacon_to_position(path_file_list)

The input variable should be a list of all file names. This function reads all the data by calling the function def read_data_file(path_filename) and convert the timestamp value in every WiFi data to a spatial coordinate.

We also define a function def wifi_heatmap(target_wifi) to visualize RSS heat map of a WiFi Ap. The input variable of this function is the bssid of a WiFi data. To visualize the heat map, we use matplotlib package. When running the program, entering three different WiFi bssid in the terminal will generate the corresponding WiFi heat map on the floor 1 of site1. In this task, the three WiFi bssids we choose are

- 0e:74:9c:2b:13:8f
- 0a:74:9c:2b:62:7b
- 74:ee:2a:cd:eb:43

The results are shown below.

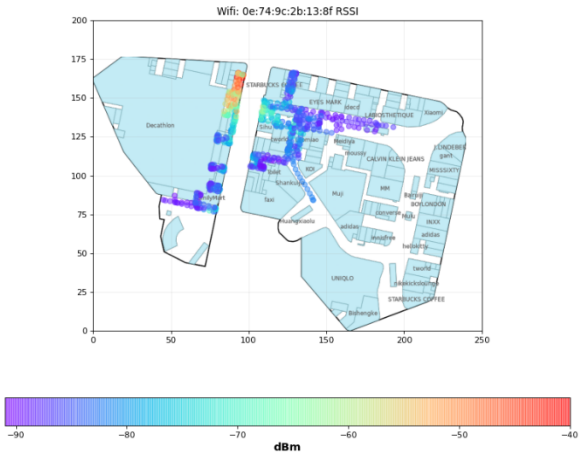

Fig. 9. WiFi 0e:74:9c:2b:13:8f heat map

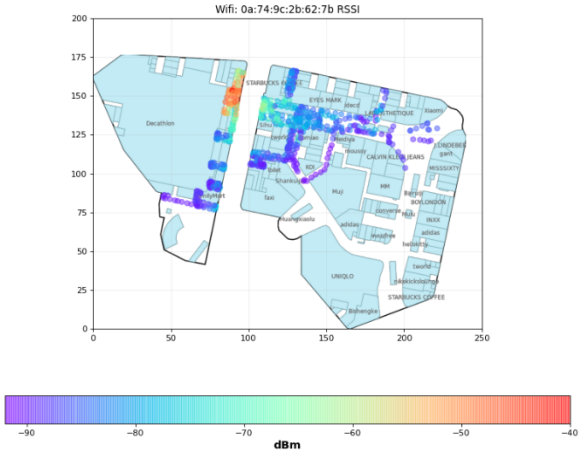

Fig. 10. WiFi 0a:74:9c:2b:62:7b heat map

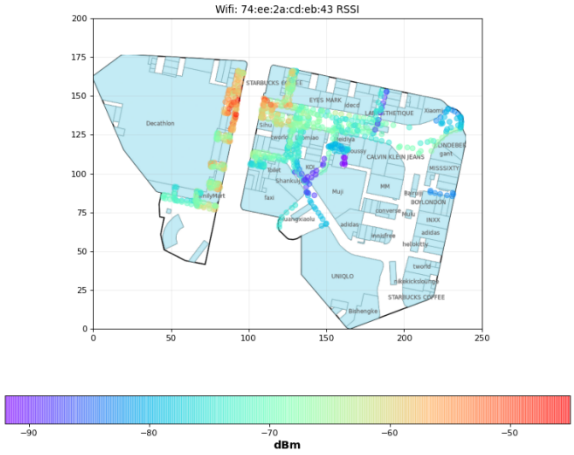

Fig. 11. WiFi 74:ee:2a:cd:eb:43 heat map

Points on each of the above figure mean that the WiFi signal has been recorded in these locations and the color of the point represents the absolute value of the WiFi power at that point.

A WiFi signal with specific bssid can be recorded many times on different traces. Thus, it can be recorded in many different locations on the same floor. Every time the WiFi signal is recorded, the absolute value of the power may not be

the same, so the picture will show different color points distributed in various locations.

IV. Bonus Tasks

4.1 Visualization of iBeacon RSSI

This section we explore the iBeacon data. According to Wikipedia[3]: iBeacon is a protocol developed by Apple in 2013. Various vendors have since made iBeacon-compatible hardware transmitters – typically called beacons – a class of Bluetooth Low Energy (BLE) devices that broadcast their identifier to nearby portable electronic devices. Therefore this is the data generated when doing Bluetooth exchange activities. Let have a look at our data, according to Github indoor-location-competition-20[1] page, one entry of iBeacon data consists of the following[4]:

- UUID: Identifier specific to the app and deployment use case

- MajorID: Further specification for iBeacon and use case

- MinorID: Further specification of region or use case

- TxPower

- RSSI

- Distance: The distance is calculated from TxPower and RSSI

- MAC Address

Next, we re-use the code for visualization of headmap to generate a iBeacon RSSI strength heatmap, the here we choose three different iBeacon IDs:

·616C6970-6179-626F-7869-626561636F6A_49662_37272

·616C6970-6179-626F-7869-626561636F6A_28214_49911

·FDA50693-A4E2-4FB1-AFCF-C6EB07647825_10073_61418

The first two are dataset with fewer data points, and the last one is the dataset with dense data points, and the rendering effect looks like:

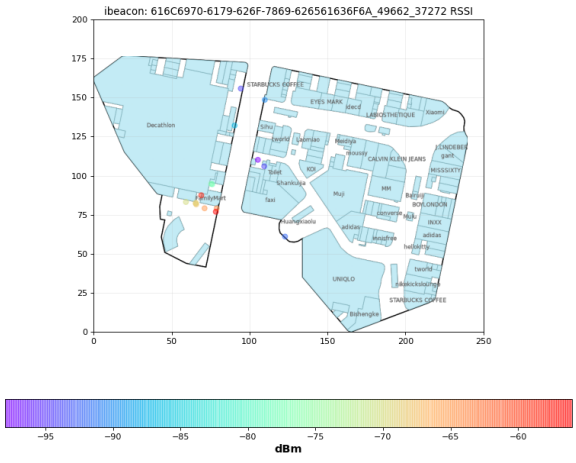

Fig. 12. 616C6970-6179-626F-7869-626561636F6A_49662_37272

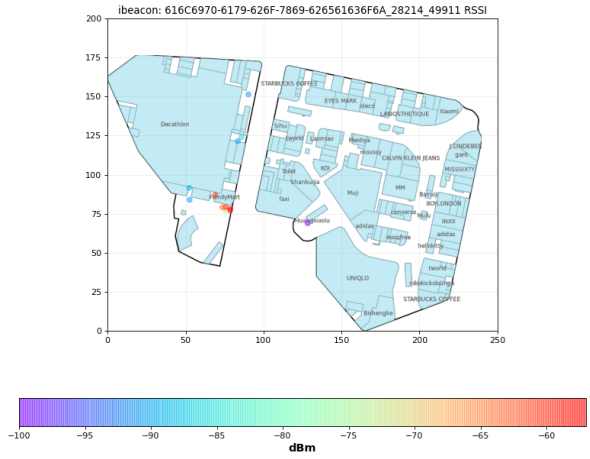

Fig. 13. 616C6970-6179-626F-7869-626561636F6A_28214_49911

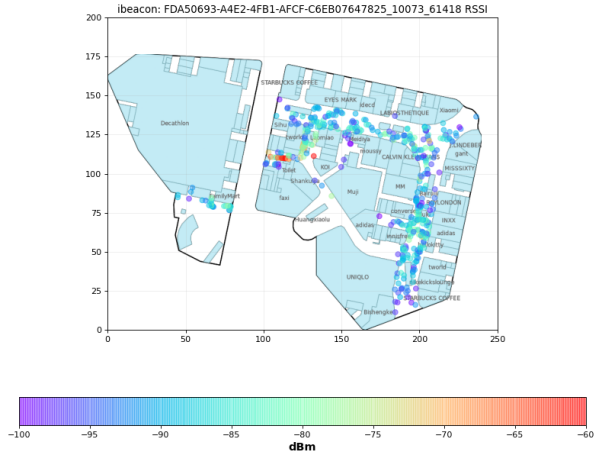

Fig. 14. FDA50693-A4E2-4FB1-AFCF-C6EB07647825_10073_61418

To implement the RSSI strength heatmap, we use a Python list to store all iBeacon positions, i.e., ibeacon_rssi/ [target_ibeacon] keys() and another Python list to store all iBeacon RSSI signal strength, i.e., ibeacon_rssi/ [target_ibeacon].values(), and re-use visualize_heatmap() function we used to generate the RSSI heatmap.

We can see from the plots that for the plot with fewer data points, the points are relatively sparse and strong, whereas for the plot with dense data points, the points are relatively crowded but the power level are relatively small, represented by the relatively purple points cloud shown on the plot. Also we can see that the trajectory of iBeacon data points is almost the same as waypoints data, this is because almost all devices that has a Bluetooth sensor also has a GPS sensor

V. Group Member Contributions

Zheng Weixiang: Introduction part and visualization of wayporints

Zhang Mingyu: Dataset part and visualization of RSS heat map of three WiFi Aps

Ren Haoyu: Visualization of geomagnetic heat map and Integration of the report

Bonus part is evenly contributed by three members

REFERENCES

[1] M. Research, “Indoor location competition 2.0 (sample data and code),” 2020. [Online]. Available: https://github.com/location-competition/indoor-location-competition-20

[2] ZhiHu, ‘Talk about the magical indoor geomagnetic positioning’, [Online]. Available: https://zhuanlan.zhihu.com/p/34176105

[3] Wikipedia, “iBeacon”, 2021, [Online]. Available: https://en.wikipedia.org/wiki/IBeacon

[4] Apple, “Getting Started with iBeacon”, 2021, [Oneline]. Available: https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf

VI. APPENDIX

5.1 Visualization of Waypoints

```python
import matplotlib.pyplot as plt

def visualize_trajectory_matplotlib(trajectory, floor_plan_filename, width_meter, height_meter):
    plt.style.use('seaborn-whitegrid') # background

    fig = plt.figure()
    ax = plt.axes()

    img = plt.imread(floor_plan_filename)
    ax.imshow(img, extent=[0, width_meter, 0, height_meter])

    x = trajectory[:, 0]
    y = trajectory[:, 1]

    plt.plot(x,y, marker='o',linestyle='dotted')

    plt.xlim([0, width_meter])
    plt.ylim([0, height_meter])

    plt.annotate('Start Point', xy=(x[0], y[0]), xytext=(x[0]/2, y[0]),
                 arrowprops=dict(facecolor='black',width=0.1, shrink=0.01))
    plt.annotate('End Point', xy=(x[-1], y[-1]), xytext=(x[-1]/2, y[-1]),
                 arrowprops=dict(facecolor='black',width=0.1, shrink=0.01))

    plt.show()

    return
```

5.2 Visualization of RSS heat map of three WiFi Aps

```python
def visualize_heatmap(position, value, floor_plan_filename, width_meter, height_meter, colorbar_title="colorbar", title=None, show=False):
    x=position[:, 0],
    y=position[:, 1],
    color=value

    plt.figure(figsize=(12, 12), dpi=80)
    ax = plt.axes()
    img = plt.imread(floor_plan_filename)
    ax.imshow(img, extent=[0, width_meter, 0, height_meter])
    plt.xlim(0,250)
    plt.ylim(0,200)
    plt.grid(alpha=0.2)
    plt.title(title)

    c = plt.scatter(x,y,c = color, cmap='rainbow', alpha=0.5)
    bar = plt.colorbar(c, orientation='horizontal')
    bar.set_label('dBm', rotation=0, fontsize=13, weight='bold')
    plt.show()

def wifi_heatmap(target_wifi):
  heat_positions = np.array(list(wifi_rssi[target_wifi].keys()))
  heat_values = np.array(list(wifi_rssi[target_wifi].values()))[:, 0]
  visualize_heatmap(heat_positions, heat_values, floor_plan_filename, width_meter, height_meter, colorbar_title='dBm', title=f'Wifi: {target_wifi} RSSI', show=True)
```

5.3 Visualization of geomagnetic heat map

```
import matplotlib.pyplot as plt

magnetic_strength = extract_magnetic_strength(mwi_datas)
heat_positions = np.array(list(magnetic_strength.keys()))
heat_values = np.array(list(magnetic_strength.values()))

x = heat_positions[:,0]
y = heat_positions[:,1]
colors = heat_values

plt.figure(figsize=(16, 16), dpi=80)
ax = plt.axes()
img = plt.imread(floor_plan_filename)
ax.imshow(img, extent=[0, width_meter, 0, height_meter])
plt.xlim(0,width_meter)
plt.ylim(0,height_meter)
plt.grid(alpha=0.2)
plt.title('Magnetic Strength Site1 F3')
c = plt.scatter(x, y, c=colors, vmin=0, vmax=colors.max(), cmap='rainbow')
bar = plt.colorbar(c, orientation='horizontal')
bar.set_label('MU: Tesla', rotation=0, fontsize=13, weight='bold')
plt.show()
```

5.4 Visualization of iBeacon RSSI

```
def ibeacon_heatmap(target_ibeacon):
    heat_positions = np.array(list(ibeacon_rssi[target_ibeacon].keys()))
    heat_values = np.array(list(ibeacon_rssi[target_ibeacon].values()))[:, 0]
    visualize_heatmap(heat_positions, heat_values, floor_plan_filename, width_meter, height_meter, title=f'ibeacon: {target_ibeacon} RSSI')

def visualize_heatmap(position, value, floor_plan_filename, width_meter, height_meter, colorbar_title="colorbar", title=None, show=False):
    x=position[:, 0],
    y=position[:, 1],
    color=value

    plt.figure(figsize=(12, 12), dpi=80)
    ax = plt.axes()
    img = plt.imread(floor_plan_filename)
    ax.imshow(img, extent=[0, width_meter, 0, height_meter])
    plt.xlim(0,250)
    plt.ylim(0,200)
    plt.grid(alpha=0.2)
    plt.title(title)

    c = plt.scatter(x,y,c = color, cmap='rainbow', alpha=0.5)
    bar = plt.colorbar(c, orientation='horizontal')
    bar.set_label('dBm', rotation=0, fontsize=13, weight='bold')
    plt.show()

ibeacon_rssi = extract_ibeacon_rssi(mwi_datas)
print(f'This floor has {len(ibeacon_rssi.keys())} ibeacons')
ten_ibeacon_ummids = list(ibeacon_rssi.keys())[0:10]
print('Example 10 ibeacon UUID_MajorID_MinorIDs:\n')
for ummid in ten_ibeacon_ummids:
    print(ummid)
target_ibeacon1 = input(f"Please input target ibeacon UUID_MajorID_MinorID:\n")
target_ibeacon2 = input(f"Please input target ibeacon UUID_MajorID_MinorID:\n")
target_ibeacon3 = input(f"Please input target ibeacon UUID_MajorID_MinorID:\n")
ibeacon_heatmap(target_ibeacon1)
ibeacon_heatmap(target_ibeacon2)
ibeacon_heatmap(target_ibeacon3)
```