SSID Positioning Approach in Pervasive Positioning Standard

By WANG Zekai
Supervised by S.-H. Gary Chan

Table of Content

Abstract

The use of SSID collected from Wifi fingerprint for indoor positioning is a feasible solution, especially in Wifi-dense areas such as shopping malls. This can be achieved through a trained machine learning model that only requires SSID and signal strength as input for prediction, resulting in the output location. This approach reduces human effort and is easily scalable for larger areas. As a result, we decided to include SSID as an additional approach in the "Pervasive Positioning Standard for Fingerprint-based and Proximity-based Systems" to serve clients when they request positioning services. This report discusses our current approach and its limitations, followed by an examination of the "SSID Localization Approach", including the necessary preparations in the pre-collection stage and the integration into our existing mobile application system.

Current Approach

In the current version of our application, we only support the campus area with two types of positioning approaches, cloud and edge.

The normal procedure:

1. We conduct the site survey of our campus and store all the data packages onto the server.
2. When the user enters the campus area, the mobile device will upload the data to our server for identification, such as BuildingID, rough latitude, longitude etc.
3. The server will tell the device which approach should be used according to the information provided.
4. The device receives the order, and starts to collect necessary data (e.g. Wifi fingerprint) used. If the site supports cloud localization, the device uploads the data to the server once completed. If the site supports edge localization, the device will calculate itself using the data.
5. The server will return the exact latitude and longitude to the device, the device displays the current location of the user inside campus on the map.

However, there is an obvious issue with this approach. In order to return a precise location, it is necessary to conduct site surveys to collect a large amount of data from various sensors. However, collecting site surveys is a time-consuming task that involves many sensors, making it difficult to quickly expand the supported area of our localization service. Additionally, due to the complex data structure of site surveys, errors in measurement of the sensors during the pre-collection stage are inevitable.

SSID Approach

To address the issues mentioned earlier, using the SSID location positioning approach is a good solution to consider. This approach is relatively easier to implement:

1. First, collect POI information from publicly available websites such as Google Maps and Apple Maps and feed it to the machine learning model as a filter.
2. Then, when the device receives the order to use the SSID approach for positioning from the server, it only needs to collect the Wifi fingerprint that includes the surrounding Wifi SSID and RSSI.
3. After that, upload the data to the server with a trained machine learning model, and the model will return a precise indoor location and send it back to the device.

Although using SSID to provide positioning service does not require conducting in-person site surveys, there is still a data pre-collection stage. Additionally, we need to figure out how to integrate this approach into our current application.

Experiment: Data Package Creation

The experiment is being carried out at Harbour City in Tsim Sha Tsui, a six-floor shopping complex consisting of multiple parts: Ocean Terminal, Ocean Centre, Gateway Arcade, and Marco Polo Hotel. We chose this location due to its high wifi and population density, which our machine learning model supports. In this data pre-collection stage, we need to build a data package following the structure below (note that in the SSID approach, there is no need to include the data colored in pink):

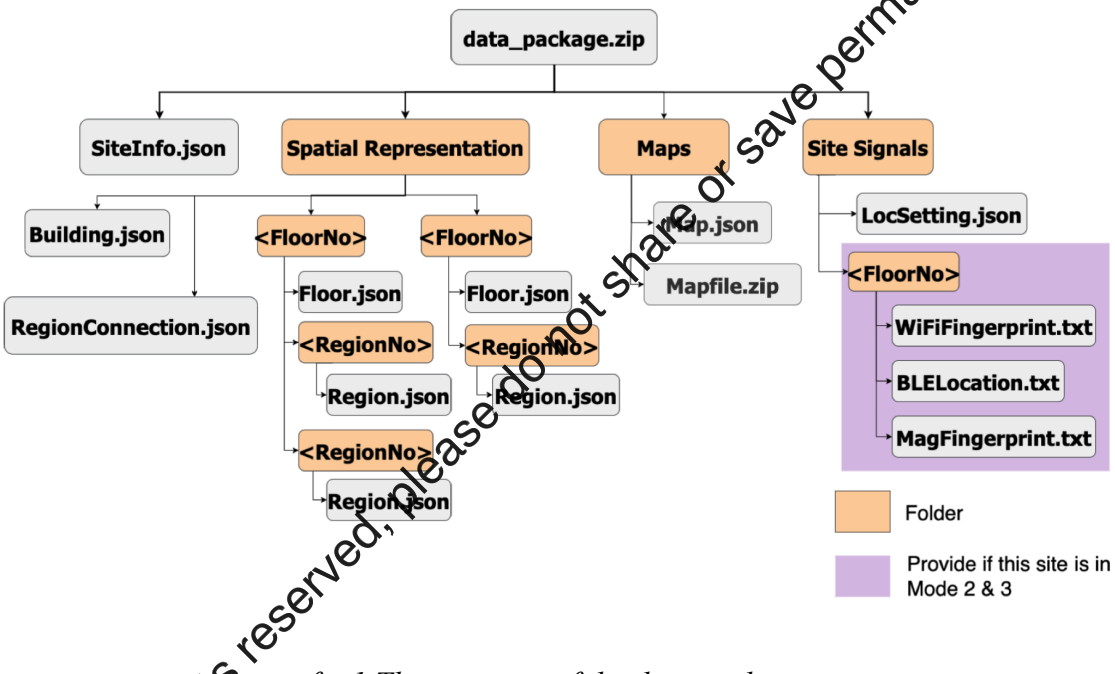

fig.1 The structure of the data package

Next, I will discuss each part of this package to explain what information should be included and how to obtain such information.

SiteInfo.json:

It acts like a cover page of this package, including the address of the site and its owner. Most importantly, it indicates whether this site is indoor or outdoor. In the case of Harbour City, it is an indoor site.

Spatial Representation:

This folder contains data representing the structure of the site, such as floors, connectors (escalators, elevators, stairs, entries to outside), etc. The backend server can precisely understand the site building with this data.

- Building.json: This file indicates the floor list of the site. As mentioned, Harbour City is a six-floor building, so we arrange the floor id from "00" to "05", and the default floor id is "00".
- FloorNo + Floor.json: For each FloorNo, we created a folder, and the Floor.json file contains the region list of this floor. The "region" here means different parts of this floor, which are decided and divided manually. However, for the experiment, we decided to make each floor a single region.
- RegionNo: Therefore, under each FloorNo folder, there is only a single RegionNo folder, which represents the whole floor.
- Region.json: It includes all connectors that connect between different floors (note that the original definition of connector is to connect between different regions, but in this case, it is the same to replace regions with floors). For connectors, there are three pieces of information that need to be included: name, tag, and geometry (latitude + longitude, correct to 6 decimal places). An additional attention needs to be paid is that, for a connector that is an escalator, we do not assume its bidirectional property. In other words, we explicitly indicate which floor this escalator goes to from the current floor. If the escalator is bidirectional (some cases are not), we will add another connector in the destination floor, indicating that this connector can go to the original floor.

This leads to my naming convention of connectors, the type of the connector is at first, then the destination floor, and finally with its id. Below is a sample of a connector:


```
"Name": "Escalator_to_4F_1",
"Tag": "Escalator",
"Geometry": [
    [
        114.165391, 22.294706
    ],
    [
        114.165446, 22.294694
    ]
]
```


fig.2 Data of a connector

- RegionConnectors.json: This folder contains all the connectors and their relationships with different floors. Since there are a large number of connectors in Harbour City, and the same connector may have different names on different floors, it is important to record the name of each connector on the map. Here is an example of my personal record:

fig.3 One of My handwriting draft of the connectors' record

Maps:

This folder contains the maps of each floor, along with their positional information.

- Mapfile.zip: This compressed file contains the map of each floor in .jpg format, which was captured from Apple Maps. In this task, I divided the map of each floor into three parts to preserve a rectangular shape that is convenient for positioning.
- Map.json: For each map image, I located its geodetic points, including the XY coordinate on the map image and the lat-long of the image in the real world. The points are mainly selected from the edges. Although the shape of the building is not perfectly a regular shape, I tried to use as few points as possible to descript the whole building, so using edges is a possible solution. This file also includes the gridding data of each image map. A sample of one image map is shown below:


```
{
    "Filename":"2F_part2.jpg",
    "MapFormat":"jpg",
    "GeodeticPoints": [
        {"X": 42, "Y": 169, "Lon": 114.168666, "Lat": 22.294422},
        {"X": 588, "Y": 59, "Lon": 114.168468, "Lat": 22.294868},
        {"X": 3647, "Y": 94, "Lon": 114.169664, "Lat": 22.297416},
        {"X": 3665, "Y": 1159, "Lon": 114.168832, "Lat": 22.297631},
        {"X": 572, "Y": 1137, "Lon": 114.169432, "Lat": 22.295046},
        {"X": 39, "Y": 1141, "Lon": 114.169531, "Lat": 22.294579}
    ],
    "Grid": ["2008568270457643","2008568280457644","2008568290457644","2008568300457644",
    "2008568270457645","2008568280457645","2008568290457645","2008568300457645","2008568260457640",
    "2008568270457640","2008568280457640","2008568290457640",
    "2008568260457641","2008568270457641","2008568280457641","2008568290457641",
    "2008568270457642","2008568280457642","2008568290457642",
    "2008568270457643","2008568280457643","2008568290457643",
    "2008568260457636","2008568260457636","2008568270457636","2008568280457636",
    "2008568260457637","2008568270457637","2008568280457637",
    "2008568260457638","2008568270457638","2008568280457638",
    "2008568260457639","2008568270457639","2008568280457639",
    "2008568270457640","2008568280457640","2008568290457640","2008568290457640"],
    "AttachedPrimalSpaceID": ["3535817370T2005043002"],
    "Validation": true
},
```


fig.4 A part of a map image data

Site Signal + LocSetting.json:

As mentioned, only LocSetting.json is needed in the Site Signal folder. In LocSetting.json, it mainly contains the gridding data of the whole site using the same gridding scheme. It also contains a flag indicating the mode of this site, which in this project is "ssid." The backend server will determine the positioning approach of a site by this flag.

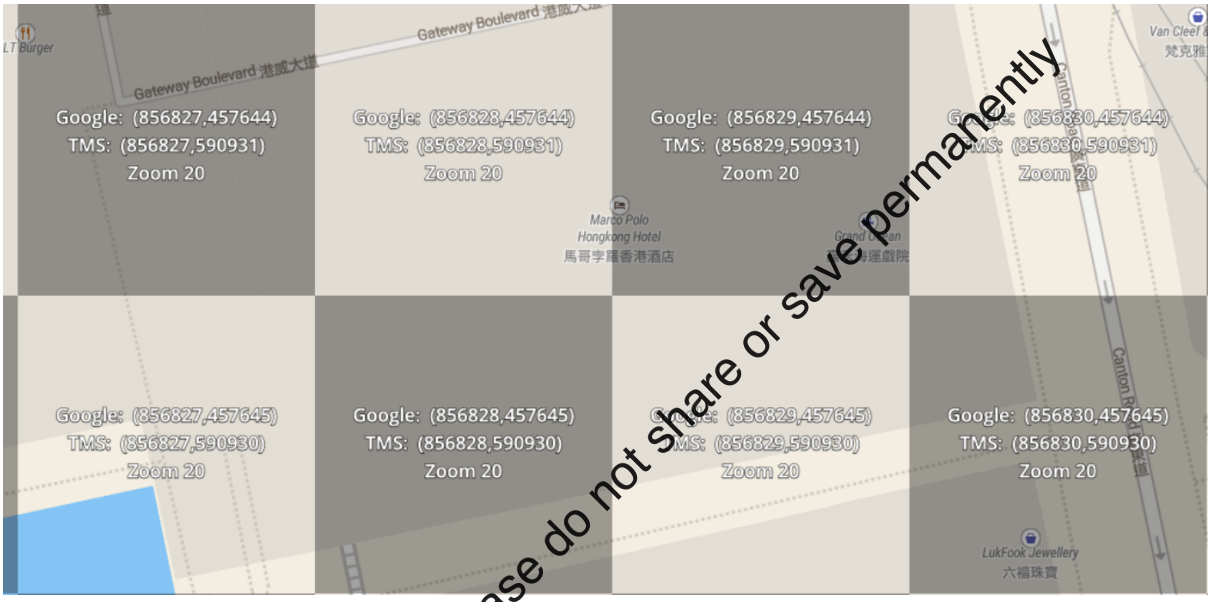

fig.5 The Marco Polo Hotel under the grid system Zoom 20

After the package was created, we uploaded it to our server and tested the API connection. The server is able to return the location setting if the device is located in Harbour City. The next step is to integrate this new SSID approach into the application.

Experiment: Integration

The original code logic flow of the application is as follows:

1. Mobile devices determine a rough lat-long and upload it to the server.
2. The server searches for the buildingID and positioning approach of the lat-long pair and returns it to the device.
3. The device starts collecting data according to the approach.

To integrate the new approach, we only need to create a branch for data collection using the SSID approach. Since the server-side is completed, we can use our own android SDK to collect Wifi signals in this branch. The device will scan its surroundings and collect all the detected Wifi signals, storing the "ssid" and "rssi" of each access point. After scanning, the application will send an API request with the Wifi signals to the backend machine learning model. The model will return the precise location of the current device. Finally, the application will display the current position on the floor map obtained from the data package in the server.

Conclusion

Up until now, we have not attempted to locate the device using the SSID approach, but we are very close to performing the testing. Additionally, we are going to conduct an in-person site survey in Tsim Sha Tsui for experimental purposes to increase the accuracy of this approach. Throughout this project, I gained a comprehensive understanding of localization and practical implementations of a localization standard. Furthermore, I honed my programming skills by modifying the localization application. Finally, I am excited to see the success of this approach and scale up the supporting regions of our localization service.

Reference

All of my work is following this standard:

Pervasive Positioning Standard for Fingerprint-based and Proximity-based Systems

Prepared by Gary W.-H. Cheung, Peter W.-L. Tsui, Mengyun Liu

Supervised by S.-H. Gary Chan