Kumar Sasmit
SBU ID - 110308698
CSE 570 - Wireless and Mobile Networks
Fall 2015

Project Report

Fingerprint based localization of mobile devices

Summary

The goal of this Project is to implement a simple fingerprint-based localization technique based on real data the student (preferably) collects. The exercise has three parts.

Requirement Details

The project has three parts.

Part 1 - Data Collection
Choose a mobile device that has an independent means of localization so that we know the ground truth location. Knowing the ground truth location is important as this helps us determine the *location error* in the fingerprinting technique. A smartphone with a GPS will work well. If you use a laptop you will need to connect it to an external GPS.

1. Find a mechanism to record the i) GPS location and ii) signal strengths of WiFi APs in the vicinity. Basically, record <latitude,longitude, AP1, RSS1, AP2, RSS2, …>. Here, RSS stands for *received signal strength*. This represents one data point in the fingerprint database. You can program your device to do this.
2. Generally speaking, you will need to know the APIs i) to get location and ii) perform WiFi scans and then record the results. There are many freeware programs that do this already for various applications. So you do not really need to write your program. For example, check Wigle. If you are using an Android device, we can give you a simple program that a graduate student has developed. You can use this directly. See Appendix 1. [I am told that this sort of scan is hard to do on iOS and no app is available. But I did not verify this.]
3. Consider an outdoor area *at least* a few hundred meters across. Larger area is fine and would provide you with more realistic experience.[1] Record N fingerprints roughly well distributed in this area. Larger N will help you in later analysis. One easy way to do this

[1] Indoor works well also. In fact, indoor loca

is simply walk, bike or drive (slowly) around this area while your app is recording fingerprints. With the right app, you can collect thousands of points very quickly.

4. If you really do not have the right device or cannot get a right app to work, we will give you data that others have collected. See Appendix 2. But it is best to make a good attempt to collect your own data. You will learn something just by attempting to do this.

Part 2 - Initial Data Processing

1. You now have N data points of the format <latitude,longitude, AP1, RSS1, AP2, RSS2, …>. Note that not all points have all APs. In other words, if there are a total of M APs that are ever seen, each point likely records a small subset of these M APs. This is simply because not APs could be heard everywhere.
2. Imputation step: Here we supply this missing data. For each data point, populate all missing APs for that data point with a signal strength below noise floor (e.g., -120 dBm). Thus, after this step, all data points have a record for all APs. For example, suppose we have seen a total of 3 APs. A data point that looks like

   <lat,long,AP1,-62dBm,AP3,-58dBm>

   will become

   <lat,long,AP1,-62dBm,AP2,-120dBm,AP3,-58dBm>

   after this imputation step.
3. We now have a fingerprint database of the following form. The example assumes a total of 3 data points and 5 APs.

Latitude	*Longitude*	*AP1*	*AP2*	*AP3*	*AP4*	*AP5*
lat1	long1	-58	-72	-48	-120	-120
lat2	long2	-35	-60	-120	-78	-91
lat3	long3	-89	-120	-120	-120	-78

4. Set aside a part of the data for testing purposes. The best way to do this is to pick a small fraction (e.g., 10%) of data *randomly* and set them aside for testing. The idea is to use these data as test cases to localize and then use the already known <lat,long> location to determine the *error*. For example, assume that we are using the 3rd data point above as test data. Then we will use the signal strengths from the 5 APs as below to determine its location.

-89	-120	-120	-120	-78

   Assume that the location is estimated as <est_lat3, est_long3>. The physical distance between the *estimated* location <est_lat3, est_long3> and the *actual* location <lat3, long3> is the *error*.

The remaining data are called *training* data (i.e., the remaining 90% if you set aside 10% of testing).

Part 3: Localization

3. Estimate the location of *each* of the test cases that are set aside in the step before. The location estimation uses a technique called *k-nearest neighbor*. The idea is to first determine *k* closest '*neighbors'* to this test point in the training data set. The idea of neighborhood here is *not* in terms of physical distance (we are acting as if we do not even know the location of the test point anyway), but in signal space.
4. To do this, assume that an M-dimensional coordinate system has been formed using the signal strength values from M APs. We determine Euclidean distances in this coordinate system to determine who the k closest neighbors are. As an example, assume again that the 3rd data point is a test point in the above example. We can determine the distance of this point from the 2nd point (e.g.) as the distance of

-89	-120	-120	-120	-78

from

-35	-60	-120	-78	-91

This is determined as

$$\sqrt{(-89-(-35))^2+(-120-(-60))^2+(-120-(-120))^2+(-120-(-78))^2+(-78-(-91))^2}$$

5. In general, the *distance in signal space* between two data points A and B with signal strengths from M APs as

$$RSS_{A,1}, RSS_{A,2}, \ldots, RSS_{A,M}$$

and

$$RSS_{B,1}, RSS_{B,2}, \ldots, RSS_{B,M}$$

is given by

$$\sqrt{\sum_{i=1}^{M}(RSS_{A,i}-RSS_{B,i})^2}$$

8. Revisit step 5. For each test point, determine its *distance in signal space* from *each* of the training points. Thus, determine the *nearest* k training points (i.e., the top k points providing the smallest distance from this test point). The assumption is that these k points should also be the closest in physical space to the test point being considered. k should be small. A value of 3 to 5 should work well.
9. Use the *average* of <lat,long> of these k training points as the estimated location of the test point being considered. Call this the *estimated location* <est_lat, est_long> for the

test point. Note that we know the actual location <lat,long> of the test point already from the measurements.

10. Compute distance between <est_lat, est_long> and <lat,long>. This provides the *localization error.* This distance is physical distance over the earth's surface. You need to look up how to compute the physical distance between two <lat,long> points. This is actually not straightforward. But good news is that the code is widely available. So, your burden is small.
11. So, finally we get the error for each of the test points. Compute the median, 67 percentile and 90 percentile errors. These numbers are your results.
12. Provide a short report on your methodology and present the above 3 numbers as results.

Appendix 1

You can install and run this program on your Android phone. It creates a data file (text format) in `/sdcard/CellularMeasurements` directory. It logs location <lat,long> as well as cellular and WiFi signal strengths that the phone hears. Ignore the cellular part. One sample data record is below. You will need yank out only the red portions using some sort of script. This shows the <lat,long> location and list of WiFi AP MAC addresses and with the RSS in dBm. This is the only information that is relevant for your work. If you move around with this program running, the phone will continuously log data.

```
2315_46_30
[52282,174397530,336]
Loc: 40.90655632 -73.10848849 2315_46_30
GSM RSSI: -103
HSPA Neighboring Cell 88 -112
HSPA Neighboring Cell 336 -105
HSPA Neighboring Cell 88 -109
WiFi: 18:64:72:1d:84:d2 WolfieNet-Secure -80 5240
WiFi: 18:64:72:20:fd:12 WolfieNet-Secure -85 5765
WiFi: 18:64:72:35:df:12 WolfieNet-Secure -85 5805
WiFi: 18:64:72:35:e2:72 WolfieNet-Secure -86 5745
```

Note since this program is not in the app store, you may need to provide appropriate permissions to install/run this program on your device. This program has been extensively used in Nexus 4 and 5 phones. If you experience bugs in other platforms please let me know.

You may also need a file browser application to inspect or to move the data. If you are not already familiar with one, try "ES File Explorer" -- a free app available in the play store.

Appendix 2

For those of you unable or unwilling to collect their own data, a cleaned up data set is made available. The data is already separated in training and test for your ease of use. The data format is csv, with each line is a single data point. The format is <lat, long, AP1, RSS1, AP2, RSS2, …> This data was collected in the Chapin apartment area on east campus.

Implementation Details

Following are the Implementation details:

1. The code has been written in C++ and is a console application which asks the data file name from the user.
2. I used the data from csv data file provided in the appendix 2 due to the unavailability of time and Final exam pressure. Please enter the filenames of already provided data file as mentioned in appendix 2.
3. I have considered a max limit of total 100 APs in the area.It can be changed any time by changing,
   ```
   #define MAX_NO_AP 100
   ```
   in the code.
4. All the location points are read from the csv data file into a structure.With signal strengths of the APs as present in the data file, also the signal strengths of the non reachable APs initialized with default signal strength = -120 dBm.
5. For storing the signal strength of each AP i have mapped the mac address of the AP to an index in an array, Such that each AP has it's unique index in the array.
6. 10% of the data have been chosen and kept as test data and the remaining as training data.
7. This number can be modified any time by changing the value of
   ```
   #define TEST_DATA 10
   ```
   in the code.The randomization has been done with respect to the present time value in at present in seconds such that each time different sets of test data are chosen.
8. I have used K-nearest neighbour estimation for finding the estimate location of each location.The value of k can be varied by changing
   ```
   #define K 3
   ```
   in the code.
9. Rest of the process followed is same as that mentioned in the requirement doc.
10. To get the values of intermediate values in various functions,
   ```
   //#define IFDEBUG 0
   ```
   Uncomment the line above in the code.

Resources

I used some web resources for finding out the distance between two latitude and longitude values.I also visited wiki pages for fetching some mathematical formulae.

Further Improvements

I followed KNN method for finding all the measurements and the final localization error.Some other methods can be used too for the same like Naive Bayes classifier as suggested. But the available time did not allow me to go into any details of it or any other method for working out the same problem.