CSE646 Final Report

Si Chen

Junfei Wang

May 10, 2013

A project of 50% of final grade.

Supervised by Dr. Murat Demirbas.

Contents

Acknowledgement of Sources

For all ideas taken from other sources (books, articles, internet), the source of the ideas is mentioned in the main text and fully referenced at the end of the report.

All material which is quoted essentially word-for-word from other sources is given in quotation marks and referenced.

Pictures and diagrams copied from the internet or other sources are labelled with a reference to the web page or book, article etc.

Signed .... Date ....

1 Introduction

Smartphones are becoming more and more popular platform for almost every aspect of people's daily life. There are currently a lot of researches on location-based applications on smartphones[1] [2] [3]. For instance, Surround Sense [1] is a service that provides logical location, based on ambience fingerprints, such as sound, light and color. Among all current researches, indoor localization is a hot topic in this area, there are a lot of papers related to this topic[2] [3]. In [2], the authors introduced a new way for indoor localization, that is, location-based signature. In [3], authors designed LiFS, an indoor localization system based on on-the-shelf WiFi infrastructure and mobile phones.

This project is aiming at tracking people both indoor and outdoor and obtain a more detailed user behavior data for further research. To achieve this goal, we implement current outdoor GPS positioning system and designing a robust indoor localization system. For indoor localization, we have utilize the WiFi signal signature, the gyroscopes, accelerometer, magnetic sensor which embedded in smartphones, and proper algorithms. The objective is to realize a meter-level indoor and outdoor localization application with no other additional device.

2 System Architecture

Our system has two major parts: client side and server side. Client side is in charge of sensor data collection and priliminary data processing, then send these processed data to remote server. Remote server will performs machine learning based algorithms and shows the relative indoor location on a map. Currently, our client side is developed on an Android system, and the server is a computer running *NIX system.

2.1 Client

Our client is set on an Android smartphone. For outdoor localization, our client will periodically acquire GPS information by using Baidu Location Service API. For indoor localization, a client will periodically listen to gyroscope, accelerometer, magnetic sensor and uses WiFi card to sense WiFi signal strength. After pre-processing, we send our sensor data to server by HTTP POST when requested over the network.

2.1.1 Outdoor Localization

To track user outdoor or before perform precise indoor location algorithm, we need to know user's last known GPS location. In our project, we adopts Baidu Location SDK to achieve more accurate localization result (this is due to the reason that Google Android Maps API V3 requires Google Play service which does not support by PhoneLab's Phone). Baidu provides a Geo-location API to help developer using its service. We need first develop our own listener (*MyLocationListener*) which implement a interface called

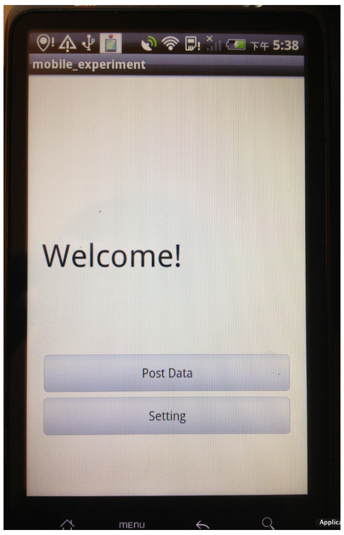


Figure 1: Android App interface

BDLocationListener. This listener works like a callback function, it will return the localization result from the server.

When app starts, it will start a service called *GPSService* and this service will help to set all the parameters and finish initialization step. This service will also open user's GPS to let it be able to acquire GPS information. In every 20 minutes, *GPSService* will post a localization request to Baidu service (if the network is available). The result returned from *MyLocationListener* will contains the latitude, longitude, altitude, time, location type. To obtain the best localization accuracy, Baidu SDK will maintain a "current best estimates" of location by filtering out new, but less accurate location data. After comparing the location from GPS, WiFi and Cellular network, it will choose the one with best estimates, replied the result and marked with different location type.

Our app will firstly store those outdoor location information into SQLite database inside the phone and initialize a flag with 0. Then for a certain time period, we post those location information which flag is 0 to our server and once it returns HTTP 200 OK, we marked this signal flag with 1.

The server will maintain a list of user outdoor tracking result, we will use these result for further indoor tracking.

2.1.2 Indoor Localization

To track user indoor, we couldn't rely on GPS information. In our app, we use several sensors to perform indoor localization.

The first sensor we used is the inbuilt wireless card. With the help of wireless card, we are able to scan the WiFi signals and also obtained the wireless signal strength (BSSID) value. In fact, we can also obtain the MAC address for each AP. To perform WiFi kNN indoor localization, our app provides two mode: scan mode and location mode. In scan mode, our app will collect all the AP MAC address and its signal strength and post those result to sever. Due to time limit, we are not adopt a unsupervised method to achieve auto-anchoring. In stead, we set several WiFi anchor points in Davis 3rd floor. We turn our app into scan mode in those anchor locations and acquire the data. For Davis 3rd floor, we set 10 anchor points, the distance between each of them is around 10m. We use Google Map combined with Davis 3rd floor plan to mark each point by using latitude and longitude. Our app set to scan 20 times for each anchor position. In location mode, the app will not scan one position 10 times. Instead, it will only perform one time scan and post the scan result to the server. The server, runs a kNN algorithm, will return the clustering result back to our client.

Since WiFi indoor localization can only provides an estimated indoor location which is not accurate enough to exactly locate a user, and its highly relate to the topology of the floor (WiFi signal strength uniformed along corridor but may encounter a sharp change at a corner), therefore we need some other methods in addition to this. In our project, we make use of the other sensors: we use accelerometer to detect user current state (walking, standing or running) and calculate user steps and use gyroscopes to detect the relative angle for each step, magnetic sensor are used for detect the direction. Those sensor information will provide a continuous location update for the user.

2.2 Server

The server is an always-on service that will collect all data sent from clients, and compute based on them, finally get the current indoor location for each client. Several machine learning based algorithms are used in server side, such as kNN algorithm.

2.2.1 kNN algorithm

kNN algorithm is the main machine learning based algorithm to determine user's current indoor location based on WiFi signal strength.

kNN algorithm is short for "K-nearest neighbors algorithm", it is is a non-parametric method for classifying objects based on closest training examples in the feature space. With a proper amount of learning approach on a certain floor of a certain building, we can get some clusters based on WiFi signal signatures. From this, the system will get the WiFi signal strength and calculate a estimated location of the user.

In our project, we use Android app to perform WiFi signal scanning at each anchor point inside the building. The result will save as a JSON format and post to the server. Server receive those "$MAC - address : BSSID$" pairs and convert them into a big parameter matrix. Each row of this matrix represent a test for an anchor point and each column of this matrix represent a AP MAC address. If in some places the scanned result does not contain certain MAC address, we just fill that blank with 0, otherwise fill the blank with wifi signal strength data.

We use Python combined with numPy library to implement kNN algorithm.

After set up the model, client posted WiFi signal strength JSON data will be clustered into one of the anchor point group. The server will then give the client a feedback.

2.2.2 WebPy and Online GUI

We want to show user tracking path directly, therefore, we choose to use WebPy. WebPy is a light-weight web server written in Python. WebPy is very simple, it can easily handle HTTP GET and HTTP POST request in few lines. With the help of WebPy, we can not only handle user post request and reply the result but also show the total progress on a website at the same time.

For storage the data, we choose to use MongoDB. MongoDB is a 'NOSQL' database, it support many useful features. One of which is the 'Document-

Oriented Storage', it provides JSON-style documents with dynamic schemas offer simplicity when update database structure for iterative development.

The benefit of building up a Online GUI is the ability to track and show the result in multiple place and cross-platform. People didn't need to install special software on his/her current operating system, we only require an internet connection and an internet browser. This design also give us great facility for testing the client app, because all the result can be viewed on both the computer and the iPad or iPhone directly, it is good for system parameter correction and system debugging.

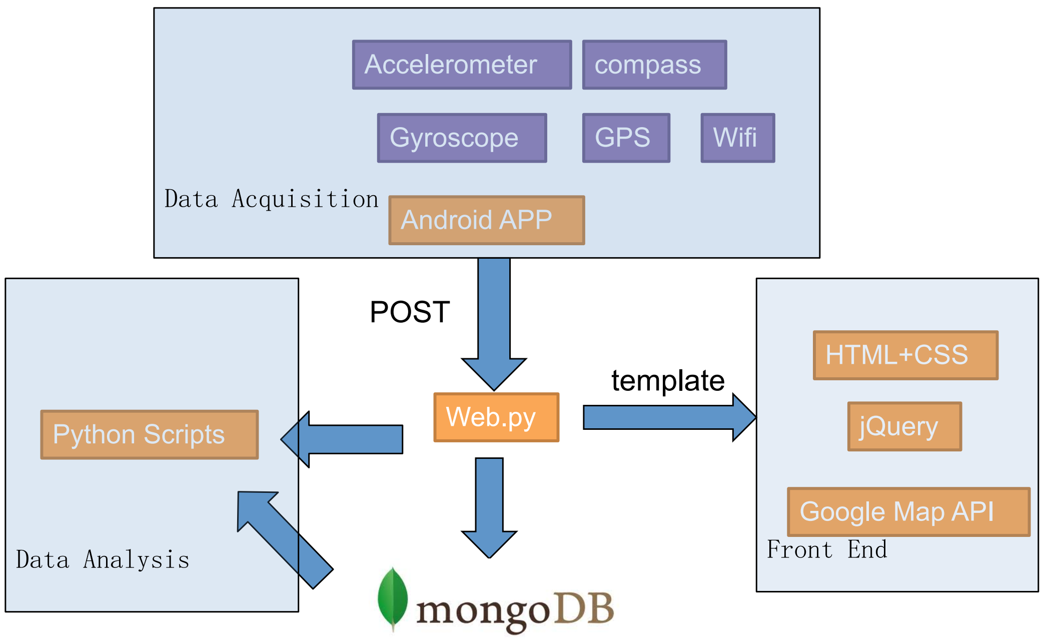


Figure 2: System Architecture

3 Experiment Setup

3.1 Step Detection

We use accelerometer readings to do step detection. As the figure shown below, this is a typical accelerometer reading, blue hollow dots are detected step beginnings, and red solid dots are detected step ends. In our project, we use a block size ($n = 8$) based algorithm to determine step. First, we use $getMagnitudeOfAccel$ to obtain the reading of the 3-axis accelerometer. The second step is to use $getLocalMeanAccel$ with a sliding window (window size $W = 15$) to calculate a list of local mean accelerate. The third step is to use $getAverageLocalMeanAccel$ to calculate the mean of the list of local mean accelerate. After that, we use $getAccelVariance$ function to calculate the variance between 3-axis accelerate and the mean of the list of local mean accelerate. Finally, we set a threshold, we called it sensitivity, this threshold is compared with the variance for each of the mean of the list of local mean accelerate and if it is larger than the threshold, it will returns 1 otherwise returns 0.

Then we use a function $checkForStep$ to count the number of 1s and number of 0s for a given list, if the continue 1s and 0s are both larger than the block size ($n = 8$), we can be sure that we detect a signal step. We will set all the count to zero and re-run the whole program to detect the following steps.

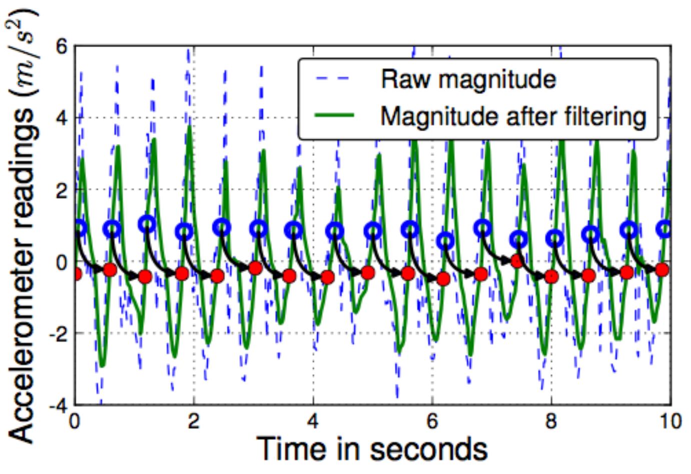


Figure 3: Accelerometer readings. Blue hollow dots are detected step beginnings, and red solid dots are detected step ends.

3.2 Heading Estimation

Upon detection of a stride, it is necessary to estimate the heading of this stride in order to calculate the next displacement of a person. Gyroscopes are useful for perform heading estimation, this is due to the reason that it is much less noisy than accelerometers or magnetometers, and a relatively immune to environmental disturbances. However, gyroscopes only measure angular velocity, which must be numerically integrated to produce heading information. The heading errors will accumulate over time without bound. In order to bound heading errors, J.Borenstein et al. [4] proposed Heuristic Drift Elimination (HDE) algorithm to calibrate gyro readings of the foot-mounted inertial measurement unit. This algorithm uses an interesting feature that corridors and walls are straight and either parallel or orthogonal to each other in a building. When a walker turns or moves on a curved paths, HDE simply suspends calibrating actions.

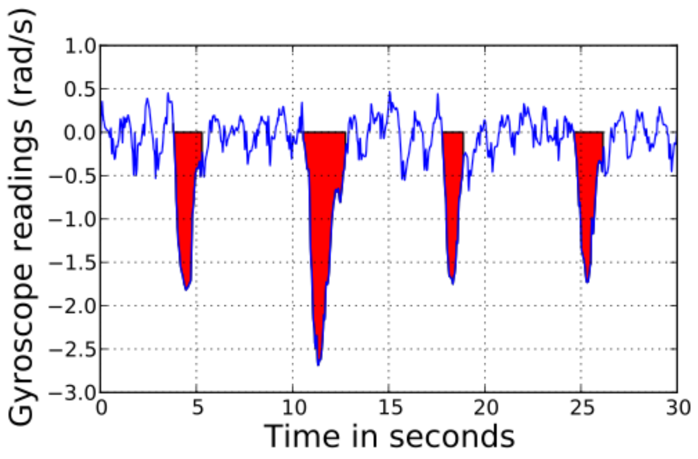
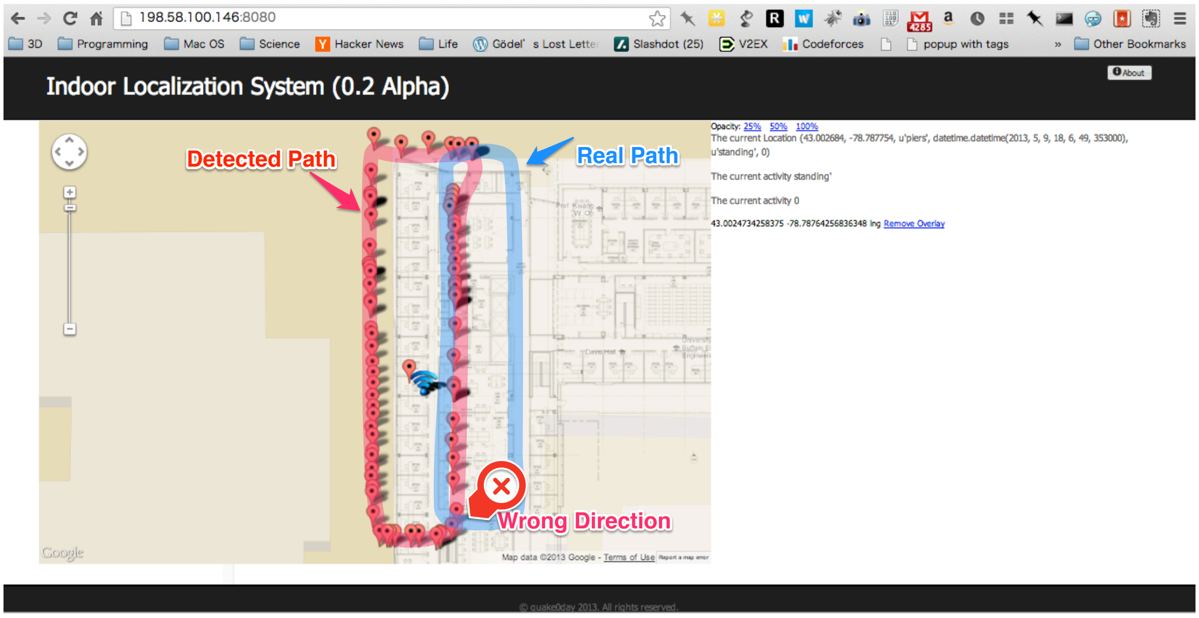


Figure 4: Detecting user turns based on gyroscope readings. The red fills represent detected user turns.

4 Experiment Result

After several tests, the proposed system is able to do both outdoor localization and indoor localization. The result is shown in a browser and we use Google Map API with customer icon to show each user reported location. The WiFi- kNN algorithm have a 90% accuracy due to the reason that many WiFi APs are implement in this floor and the step detection method is performed pretty good. After several test, we can accurate plot the user walking path within 5-meter error. There are several drawbacks for our system. The first one is miscalculate of the user turning direction. It happens 20% of the time. Since the direction is calculate based the combination of gyroscope and magnetometer sensor, it is valuable to electric interference. The demo place is in UB north campus Davis 3rd floor, which is the floor for UB computer science and engineering department, many computers and servers, are running in this floor. The magnetometer is not suitable for such environment. The next issue is there still contains 10% error when perform WiFi kNN localization, it always happens at one certain anchor (which kNN returns a incorrect one), add more BSSID data does not help.



```
The current Location (43.002684, -78.787754, u'plers', datetime.datetime(2013, 5, 9, 18, 6, 49, 353000),
u'standing', 0)

The current activity standing'

The current activity 0

43.0024734258375 -78.78764256836348 lng
```

Figure 5: Error when detecting user turns

5 Future Work

This system is still in α version, it only provides several basic outdoor and indoor localization function. In the future, we will fix the wrong direction problem by applying several rectify algorithms. The aim of this project is to record user daily routine path, therefore, we also need to consider the energy issue when perform tracking.

References

[1] Azizyan, Martin and Constandache, Ionut and Roy Choudhury, Romit, *SurroundSense: mobile phone localization via ambience fingerprinting,*. Addison Wesley, Massachusetts, MobiCom 09, 2009.

[2] Wang, He and Sen, Souvik and Elgohary, Ahmed and Farid, Moustafa and Youssef, Moustafa and Choudhury, Romit Roy, *No need to war-drive: unsupervised indoor localization,*. MobiSys 12, 2012.

[3] Yang, Zheng and Wu, Chenshu and Liu, Yunhao, *Locating in fingerprint space: wireless indoor localization with little human intervention,*. MobiCom12, 2012.

[4] J. Borenstein and L. Ojeda, Heuristic Drift Elimination for Personnel Tracking Systems, Journal of Navigation, vol. 63, pp. 591-606, Sept. 2010.