**Project Report on**

# Scalable Indoor Positioning System Driven By User Feedback

*by*

**M. Sri Krishna Shanmukh**

(Roll No: 127236)

**G. Kranthi Kiran**

(Roll No: 127223)

**L. Midhun Sai Reddy**

(Roll No: 127237)

*Under the esteemed guidance of*

**Dr. D. V. L. N. Somayajulu**

Professor


DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY, WARANGAL

2012-2016

# Abstract

Location aware assistance is the backbone of hyper local marketing. As in the digital world marketers would like to target the right customers when they are at the right location. With growing penetration of smartphones, geo based marketing is being preferred by both consumers and retailers. The most important problem of location based marketing is to accurately know the user's position to a few meters. GPS is very inaccurate when it comes to tracking behavior within a retail store. Much work has been done in the past on Wi-Fi based indoor positioning. But this comes with a huge upfront cost of accurately mapping a location and is very time consuming. In this project we will be exploring large scale indoor positioning that can get down cost of mapping a location and driven by incentivized user behavior.

# Table of Contents

# Table of Figures

# Introduction

After over a decade of research and development, location-aware services have gradually penetrated into real life. They assist human activities in a wide range of applications such as fleet management, travel aids, location identification, emergency services and vehicle navigation. With the technology for location – aware services in outdoor environment being well established, their applicability for indoor environments is being explored. The applications that can be developed, once a well indoor positioning system is developed, are endless. They extend from local – hyper marketing to route assistance in case of an emergency.

Many outdoor positioning systems are based out of Global Positioning System (GPS). However, it cannot be used for indoor positioning, as GPS signals cannot reach indoors. The inquisitive nature of human, in finding an alternative for indoor positioning has led to several innovative technologies which use infrared, laser and/or ultrasonic range finders. The disadvantages of such an approach are its size, complexity, and cost, which render it infeasible for mobile devices.

For indoor position determination, Wi-Fi infrastructure has become quite popular now. Location can be determined using fingerprinting or trilateration. Wi-Fi trilateration is based on the principle that Received Signal Strength Indication (RSSI) of a Wi-Fi Access Point (AP) decreases as distance increases. This approach, when used alone, is quite ineffective due to the complex building geometries, random movement of people etc. and involves lot of math and positioning of Wi-Fi AP's to be exactly determined in advance. Wi-Fi fingerprinting uses RSSI values of each Wi-Fi AP obtained at a point to determine the location. This approach has two phases – training and positioning. In training phase, RSSI values at each location are obtained and stored in the database. In positioning phase, the current obtained RSSI values are fed to the trained dataset and the most probable location of the user is calculated.

Wi-Fi fingerprinting involves significant costs in terms of initial configuration and ongoing maintenance in order to continuously adapt to environmental changes and Wi-Fi infrastructure

alterations. To reduce such costs and enhance the accuracy of the system, we can integrate User Feedback to the fingerprinting system.

# Literature Review

[1] *Analysis of three indoor localization technologies to support facility management field activities*, presented a comprehensive overview of comparison between WLAN, RFID and IMU technologies usage for indoor localization. It was shown that of all the three different technologies, WLAN (Wi-Fi) technology reported the least error, hence highest accuracy. Fingerprinting technique was used in determining location using WLAN, using kNN algorithm for location determination, given training dataset.

[2] *Enhancing Wi-Fi fingerprinting for indoor positioning using human-centric collaborative feedback,* forms a baseline for our project. It presented usage of user feedback to the indoor positioning system, so as to reduce maintenance costs and increase efficiency of system. It uses Gaussian function for determining most likelihood location of the user.

[3] *Location determination using Wi-Fi fingerprinting versus Wi-Fi trilateration,* presented an overview of Wi-Fi fingerprinting and Wi-Fi trilateration approaches. Means of RSSI's are stored after the training phase and euclidean distance is used in finding the most appropriate location of the user.

[4] *Indoor Wi-Fi Positioning System for Android-based Smartphone,* presented usage of android smartphone for display of user location. It used manhattan distance in Wi-Fi fingerprinting for user location determination.

## Problem Statement

While GPS signals are excellent for navigational and tracking purposes outdoors, they are weak and unreliable indoors. Tracking positions indoors can allow users to infer which areas are occupied during the course of a day from a commercial and security perspective. The InPoSy (Indoor Positioning System) provides a comprehensive solution for indoor tracking and personal navigation using an Android powered mobile application and a web-based remote monitoring application.

The team decided to make a mobile application and web-based remote monitoring application to solve the problem of accurate indoor tracking and navigation. The mobile application finds an accurate location fix on a smartphone in real time using Wi-Fi access points, and plots these coordinates on a digital map. Furthermore, it simultaneously transmits the coordinates to a remote monitoring application, which will collect and display data from multiple smartphone users. The team chose the Android platform because of its widespread use, availability of test phones, and collective experience with programming in Java. As for the remote monitoring application, initially the team had planned to implement a mobile based solution, however a web-based application allowed for greater platform independence.

The InPoSy had specific design parameters for each of its subsystems. For the mobile application, these included an accurate position fix, frequent position updates, and a friendly user interface. The position fix needed to be accurate to within a three meter radius and updated five times per second, accounting for a running user. The vast majority of users generally do not move faster than five meters per second. (By computing position five times per second, the system will track user movement with a resolution of better than three meters.) Additionally, Wi-Fi fingerprinting, which uses signal strength to pinpoint device location, will need to function with 80% accuracy even when there is interference in the Wi-Fi spectrum. This would result in four out of five locations determined from fingerprinting to be accurate within a three meter radius. Additionally, the user interface need to adhere to Android Human Interface guidelines and provide easy-to-use buttons and menus for navigation.

For the remote monitoring application, the team should ensure that the application captures data from multiple users simultaneously. The team also should ensure a one to one correspondence between the position fix shown on a mobile device and the corresponding position on the remote monitoring application.

The InPoSy system must operate in a standard mobile phone environment with enough Wi-Fi access points to provide network connectivity. The physical environmental requirements of the mobile system are the same as those of any Android phone. The environment will need to contain sufficient Wi-Fi coverage for the InPoSy to perform successful fingerprinting and transmission of location data to the remote monitoring application. The Wi-Fi fingerprinting technique becomes more accurate when more access points are in range of the smartphone. Since 802.11G access points have a typical range of 38m, system administrators will need to install enough access points to cover the area to be monitored. These access points do not need to be on the same network, but they must remain stationary. When Wi-Fi connectivity is reestablished, the new position data can be relayed to the remote monitoring application. The mobile device running the application may access the Internet through these Wi-Fi access points or through a mobile carrier.

# Solution offered

## Wi-Fi Fingerprinting Approach

### Overview

The design consists of two phases – Training and Positioning. In offline phase, there are further two subsystems, first is an application on an Android phone, and the second is the Remote Monitoring Application that runs on a web server. The smartphone application sends real-time position information to the Remote Monitoring Application. The mobile application reads Wi-Fi RSSI values and sends it to training dataset through Remote Monitoring Application. In positioning phase, this dataset will be used as input for Location determination Algorithm.

### Training phase

The system training is conducted for each survey point in a two-step process. The first step is to collect multiple Wi-Fi scans in order to stabilize the average of RSS readings. The following step utilizes the information collected by these Wi-Fi scans to generate an RSS fingerprint for each survey position.

#### *Collect Raw Wi-Fi RSS data*

At each survey position, system administrators use trainer application which is described below to scan for nearby Wi-Fi APs (Access Point). In each Wi-Fi scan, beacon frames from different APs are received and converted to a tuple, which contains the MAC address of an AP, the RSS in dBm, ScanID and indexID (unique ID associated to a point). Note that a single scan may not be able to capture beacon frames from all nearby APs due to the different beacon frame broadcasting periods or severe signal fading. Also, the collected RSS values have a natural variation when indoors, which is unavoidable. To compensate the RSS fluctuation and obtain complete AP information, a sufficiently large number of scans is needed to create an RSS fingerprint. As a result, in a given period of sampling, the device logs a time series of RSS vectors. These vectors are then used to construct the Wi-Fi RSS fingerprints for each measured location in the training grid.

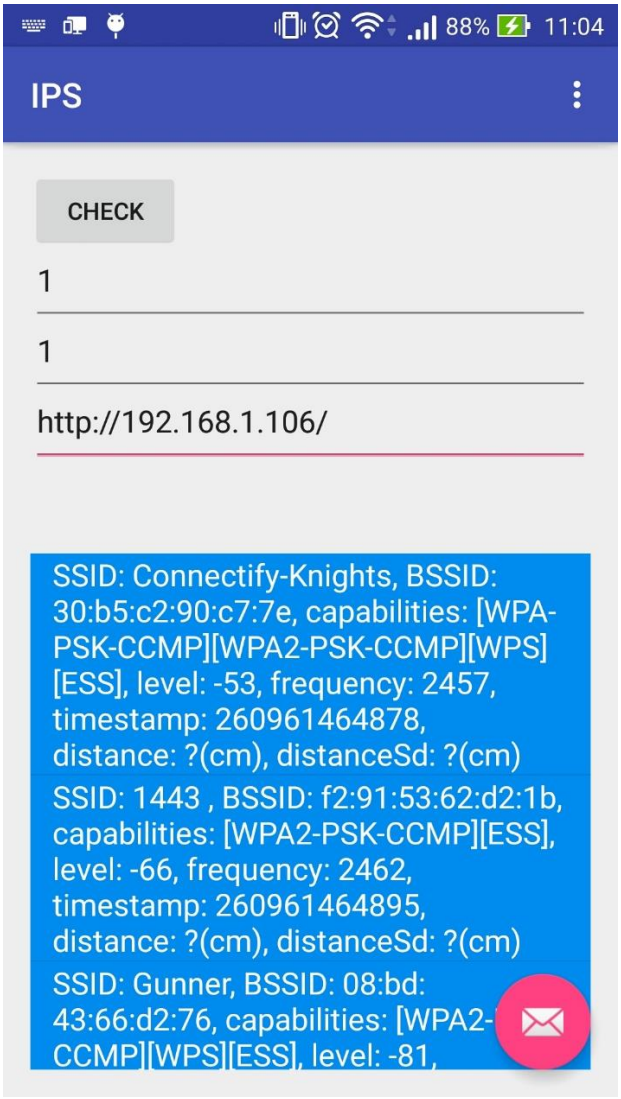


*Figure 1 Trainer Application for Android Mobile*

The first text box in the Trainer Application is used for providing the index ID which represents the location spot chosen by the system administrator. The second text box in the Trainer Application is used for providing the scan ID which as mentioned earlier is used for scanning multiple times at a chosen index ID location to minimize the interference effects and get normalized Wi-Fi RSS readings. The third text box is used for communicating with the server which acts as the Remote Monitoring Application running at a specified location as a web server. As shown in Figure 2, the data captured from the activity mentioned above by the System

administrator using the Trainer application, will be sent as a HTTP Request through the Trainer Application in real time to the Remote Monitoring Application ran as web server.

| | id | indexID | mac_id | rssi_value | ssid | scanID |
|---|---|---|---|---|---|---|
| Edit Copy Delete | 19 | 1 | 30:b5:c2:90:c7:7e | -53 | Connectify-Knights | 1 |
| Edit Copy Delete | 20 | 1 | 08:bd:43:66:d2:76 | -81 | Gunner | 1 |
| Edit Copy Delete | 21 | 1 | f2:91:53:62:d2:1b | -66 | 1443 | 1 |
| Edit Copy Delete | 22 | 1 | 02:1a:11:ff:f5:90 | -28 | BCPV-c2hhbm51 | 1 |

*Figure 2 Training Data captured from Trainer Application*

**Positioning phase**

The positioning phase is the phase where the calculation software periodically receives measurements from one or more mobile devices. This information is compared against the values obtained from the training phase, which yields a calculated position for each device. Once the received measurement has been parsed and found to be correct, it will be used as input for the calculation algorithm described below.

## k-NN algorithm

### *Overview*

The k – Nearest Neighbors algorithm (or k-NN for short) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression.

In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small).

In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors.

***Algorithm***

The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.

In the classification phase, k is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the k training samples nearest to that query point.

Distance from the training vector (x) to the unlabeled vector (y) can be calculated using any of the following methods.

$$Euclidean\ Distance = \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2}$$

$$Manhattan\ Distance = \sum_{i=1}^{k} (x_i - y_i)$$

$$Minkowski\ Distance = \left(\sum_{i=1}^{k} (|x_i - y_{i|})^q\right)^{1/q}$$

From the above formulae for Distance calculation, it is evident that Euclidean Distance is a special case of Minkowski Distance obtained when the value of q is 2. Also, Manhattan Distance is a special case of Minkowski Distance obtained when the value of q is 1.

***Example***

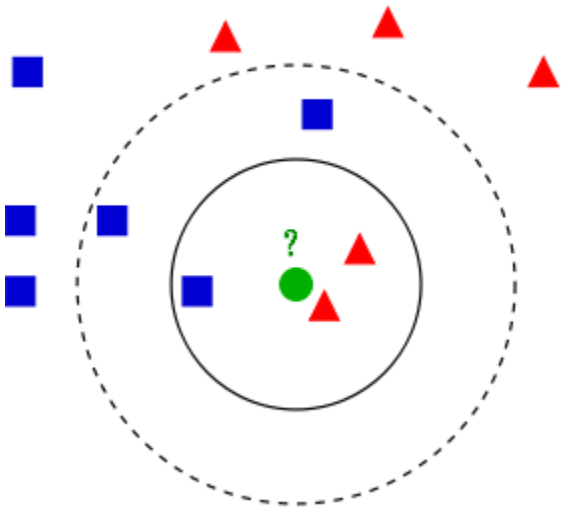


*Figure 3 Dataset for k-NN algorithm*

In the above example, we have taken k = 3. The class of the unlabeled point in green color can be determined by finding its 3 closest neighbors. Here, we have two classes (red color and blue color). The unlabeled point takes red colored class as it has majority number of elements in the closest neighbor set.

## Location Determination Algorithm

### Overview

For our project, we have chosen k – value as 1. So, the problem reduces to finding of the closest neighbor for the given live RSSI tuple. Each index point is taken as a class and the mean of all the scans at that point is determined as the only class point. The distances from the live RSSI tuple to mean RSSI tuple of each class are determined for finding the most probable location of the user.

For determining the distances, we have even assigned weight to each Wi-Fi AP in the mean tuple of the class. The weight is assigned based on the number of appearances of that Wi-Fi AP in the training data of that particular point to the total number of scans performed at that point.

**Algorithm**

---

**Algorithm:** Algorithm for location determination

---

**Input:** Fingerprint Database containing $(P, F)$ – (point, fingerprint) of all points
$R$ – live RSSI tuple at a point

1. Calculate $(B, P')$ – (BSSID, points) tuples from fingerprint database Each BSSID tuple contains all corresponding position id's for that BSSID
2. Take Set $bssid$ – for storing all points which contain any of the BSSID in R
3. **for each** BSSID **in** R **do**
   add all points $(P')$ of $B == BSSID$ to the set $bssid$
   **end for**
4. **for each** point **in** $bssid$
   Compute Euclidean distance between R and RSSI of point
   $Euclidean\ distance = \sqrt{\sum_{i=1}^{n}(point.F_i - R_i)^2}$
   where $point.F_i$ is RSSI value for corresponding MAC (BSSID) in $R$
   **end for**
5. Set probablePoint as point with least Euclidean Distance

Manhattan Distance can also be used for finding probable location

$$Manhattan\ distance = \sum_{i=1}^{n} point.W_i * (point.F_i - R_i)$$

For greater accuracy, we use weights obtained from database in calculation of Euclidean Distance as follows

$$Euclidean\ distance = \sqrt{\sum_{i=1}^{n} point.W_i * (point.F_i - R_i)^2}$$

---

```
// Finding probable point
public String caluclateVarianceEuclid(Hashtable<String,Double> bssid){
    ....
    Set<String> points = new HashSet<String>();
    Set<String> keys = bssid.keySet();
    for (String key: keys)
    {
        points.addAll(bssid_Points.get(key)); // Store points corresponding to a bssid
    }
    for(String ipoint: points)
    {
        tvar = iPoint.get(ipoint).calculateVarianceEuclid(bssid);
        if(tvar < var)
        {
            var = tvar;
            tpoint = ipoint;
        }
    }
    return tpoint;
}

// Calling calculateVarianceEuclid
public void testData(){
        ....
        int count=0,manCount=0,euclidCount=0;
        for(; (line = br.readLine()) != null; )
        {
            // process the line.
            Hashtable<String, Double> bssid = new Hashtable<String,Double>();
            String[] parts =  line.split(",");
            curPoint = parts[0];
            for(int i=1;i<parts.length;i+=2) // Convert Bssid-level to a Hashtable
            {
                bssid.put(parts[i], new Double(Double.valueOf(parts[i+1])));
            }
            if(parts[0].equals(data.caluclateVarianceEuclid(bssid))) // point - BSSID values
            {
                euclidCount++;
            }
        }
    }
    ....
}
```

*Figure 4 Code Snippet for Location Determination*

Above code snippet calculates Euclidean Distance for the live RSSI with each point in the database and returns the point with least distance.

```
// Calculting Point with least Euclidean Distance

public Double calculateVarianceEuclid(Hashtable<String,Double> bssid) // Calculate Variance given BSSIDs
{
    ....
    Set<String> keysBssidList = bssidList.keySet();

    for(String eachBssid: keysBssidList) // each bssid for a Point
    {
        values = bssidList.get(eachBssid);
        Double temp;
        temp = bssid.get(eachBssid);
        if(temp == null)
        {
            var += (values[1])*( 100 + values[0])*(100 + values[0])/30;
        }
        else
        {
            var +=  (values[0] - temp) * (values[0] - temp) * values[1]/30;
        }
    }

    keysBssidList = bssid.keySet();

    for(String eachBssid: keysBssidList) // each BSSID in live RSSI tuple
    {
        if(bssidList.get(eachBssid) == null)
        {
            Double temp;
            temp = bssid.get(eachBssid);
            var += (100 + temp)*( 100 + temp);
        }
    }

    return var;
}
```

*Figure 5 Code Snippet for Euclidean Distance Calculation*

Above code snippet determines how Euclidean Distance is calculated for a given live RSSI tuple and point in the database.

## Technologies Involved

For any software to be built, revision or version control is an important aspect, hence we started our project by learning about Git, a version control system to manage a project over time to push, pull, rollback and also collaborate with other team members over a version control host. We have setup our private repositories under the organization named InPoSy on www.gitlab.com and started maintaining our repositories for various modules of the software on gitlab.com. We have also setup our blog as a repository so as to maintain track of our documentation and insight into our modules under InPoSy. The blog has been setup using github.com and a ruby based tool called Jekyll.

In order to test the efficiency of various algorithms designed by us, we have chosen Java as our primary language for implementation purposes since it is simple, robust, highly object oriented and has a lot of user base on the web which will help us in solving any programming based issues. We also use Python to some extent in understanding the raw dataset and also in utilizing small graphical user interface libraries in better understanding of the data. We use Eclipse Java EE(Enterprise Edition) Integrated Development Environment(IDE) to write Java snippets and our core algorithm implementations.

We also have learnt about cross platform mobile application development by trying Xamarin and Phonegap frameworks. Since, native application development gives more freedom in implementation of our algorithm, we chose to develop for one mobile platform at a time and started working on Android Studio IDE with target platform as Android 5.1.1. We are using the in-house material design provided by Google to cater our UI needs for mobile application development. Later, we shall move to implement using cross platform mobile application frameworks which we have learnt previously.

For the remote monitoring application in our software which processes, computes and communicates data to our Trainer and Client Applications, we have chosen django framework to implement the server. We have learnt to implement a server using django since it offers robust scaling, high performance and availability of various plugins & packages. We have made a proxy

based server model, where requests will be called as REST API calls to the django based server configured on an Nginx server.

We have planned to use

- Android Material Design concepts provided by Google to implement the graphical user interface for our mobile Trainer and Client applications.
- AngularJS to provide dynamic user interaction with the server in our remote monitoring application.
- MaterializeCSS and Bootstrap CSS frameworks to provide aesthetic touch to the user interfaces in our software.

## Conclusions

We have developed a Trainer Application in Training phase for capturing training data and in positioning phase we have tested for accuracy of various algorithms over the training data.

Using the Euclidean Distance by combining the weights assigned to each is the most accurate in classifying the data point. This is followed by the Euclidean Distance without weight assignment which has a slightly less accuracy rate. The classification algorithm that uses the Manhattan Distance with weights assigned, showed the worst performance as well as the worst classification rate.

We plan to develop Android Client Application to display the position on a graphical user interface at real time which is returned by the Location determination algorithm.

# References

[1] Saurabh Taneja, Asli Akcamete, Burcu Akinci, James Garrett, Lucio Soibelman, E. William East. *Analysis of three indoor localization technologies to support facility management field activities (2009)*

[2] Yan Luo, Orland Hoeber and Yuanzhu Chen. Luo et al.: *Enhancing Wi-Fi fingerprinting for indoor positioning using human-centric collaborative feedback. Human-centric Computing and Information Sciences 2013 3:2.*

[3] E. Mok & G. Retscher (2007). *Location determination using Wi-Fi fingerprinting versus Wi-Fi trilateration, Journal of Location Based Services, 1:2, 145 159, DOI:10.1080/17489720701781905*

[4] Beom-Ju Shin, Kwang-Won Lee, Sun-Ho Choi, Joo-Yeon Kim, Woo Jin Lee, and Hyung Seok Kim. *Indoor Wi-Fi Positioning System for Android-based Smartphone*