|  |
| --- |
| **History Flow Analysis in WiFi-based Environments** |
| --EE 5003 project |
| **Fei Yang** |
| **(A0169096N)** |

**Department of Electrical & Computer Engineering, NUS**

In partial fulfillment of the requirements for the Degree of Master of Engineering

National University of Singapore

**12 April 2018**

# Abstract

In this report it proposes a history flow analysis method and designs a friendly application which can automaticly download, process data and show result. By employing Python3.6, MySQL5.7, Tableau, the application shows good performance for NUS WiFi history flow analysis.

This report mainly analyzes the historical flow of NUS buildings from user dewll time and flow rate. It has addressed the download inconvenience from datacommons website and extract problem from a week's compressed file. In order to improve the read and write speed, the application uses MySQL to store data. To improve result presentation, it applys Tableau to display line chart of flow density, pie chart of user type. In addition, it also utilizes the python tkinter GUI to make this application more friendly.

It has provided a detailed method of python, including designing UI, processing csv file, achieving analysis algorithm and operating MySQL. It also demonstrated basic operation of MySQL and Tableau. This detailed process serves as a guide for researchers to continue optimizes our application. Last but not least, it will get better performance with more analysis algorithms and more optimized data structure. And this more accuracy result can be used to estimate building occupancy and utilization to do management for saving energy.

**DECLARATION**

I hereby declare that this report is my original work and it has been written by me in its entirety. I have duly

acknowledged all the sources of information which have been used in the report.

This report has also not been submitted for any degree in any university previously.

Fei Yang

13 April 2018

## TABLE OF CONTENT

[CHAPTER 1 INTRODUCTION 1](#_bookmark0)

* 1. [Background 1](#_bookmark1)
  2. [Motivation and Challenges 2](#_bookmark2)
  3. [Data structure and places 3](#_bookmark3)

[CHAPTER 2 HEAT MAP 8](#_bookmark6)

* 1. [Introduction of heat map 8](#_bookmark7)
     1. [History of heat map 8](#_bookmark8)
     2. [How to generate density heat map 10](#_bookmark9)

[CHAPTER 3 MONTE CARLO METHOD 18](#_bookmark18)

* 1. [Received Signal Strength (RSS) 18](#_bookmark20)
  2. [Error distribution 20](#_bookmark22)
  3. [Monte Carlo Method 26](#_bookmark29)

[CHAPTER 4 AVERAGE PREPROCESSING 38](#_bookmark42)

* 1. [Stationary device judgment 38](#_bookmark43)
  2. [Average preprocessing theory 40](#_bookmark45)
  3. [Implement of Average preprocessing 42](#_bookmark46)
  4. [Application of Average preprocessing 42](#_bookmark47)

[CHAPTER 5 CONCLUSION 46](#_bookmark52)

[Reference 47](#_bookmark53)

[Annex 49](#_bookmark54)

**LIST OF FIGURES**

[Figure 1.1The floor plan of E1-06(Top), E4-03(Mid), E5-03(Bottom) with devices and Access](#_bookmark5) [Points 7](#_bookmark5)

[Figure 2.1.The flowchart of generating heat map 10](#_bookmark10)

[Figure 2.2 The frequency map(left) and the heat map(right) 12](#_bookmark11)

[Figure 2.3 Heat map with different area size for same frequency matrix. Too small size (Top),](#_bookmark12) [suitable size (Mid) and too large size(Bottom) 13](#_bookmark12)

[Figure 2.4 Nearest Neighbor Interpolation 15](#_bookmark13)

[Figure 2.5 Linear Interpolation 15](#_bookmark14)

[Figure 2.6 Natural Neighbor Interpolation 16](#_bookmark15)

[Figure 2.7. cubic Interpolation 17](#_bookmark16)

[Figure 3.1 Original data points (Top) and heat map (Bottom) 18](#_bookmark19)

[Figure 3.2 RSS localization method 20](#_bookmark21)

[Figure 3.3 The error distance between real position and localization position in E1-06. Top](#_bookmark23) [one is the longitude and bottom one is latitude in meters 22](#_bookmark23)

[Figure 3.4 he error distance between real position and localization position in E4-03. Top one](#_bookmark24) [is the longitude and bottom one is latitude in meters 23](#_bookmark24)

[Figure 3.5 The error distance between real position and localization position in E5-03. Top](#_bookmark25) [one is the longitude and bottom one is latitude in meters 23](#_bookmark25)

[Figure 3.6 lillietest result for E4-03 latitude 25](#_bookmark27)

[Figure 3.7 Normplot of E4-03 latitude 25](#_bookmark28)

[Figure 3.8 Flow-chart of Monte Carlo Method 26](#_bookmark30)

[Figure 3.9 Floor division 27](#_bookmark31)

[Figure 3.10 Real position situation with Gaussian distribution 27](#_bookmark32)

[Figure 3.11 Method to get real position probability 28](#_bookmark33)

[Figure 3.12 Flow chart of Monte Carlo method 29](#_bookmark34)

[Figure 3.13 Single boundary linear program 31](#_bookmark35)

[Figure 3.14 multiple boundary linear program 32](#_bookmark36)

[Figure 3.15 multiple dimension multiple boundary linear program 33](#_bookmark37)

[Figure 3.16 The data point map(top two) and heat map(bottom two) of E1-06 before Monte](#_bookmark38) [Carlo method(left two) and after Monte Carlo method(right two) 35](#_bookmark38)

[Figure 3.17 The data point map(top two) and heat map(bottom two) of E4-03 before Monte](#_bookmark39) [Carlo method(left two) and after Monte Carlo method(right two) 36](#_bookmark39)

[Figure 3.18 The data point map(top two) and heat map(bottom two) of E5-03 before Monte](#_bookmark40) [Carlo method(left two) and after Monte Carlo method(right two) 36](#_bookmark40)

[Figure 3.19 Bad situation for Monte Carlo method 37](#_bookmark41)

[Figure 4.1 Flow-chart to justify stationary device 39](#_bookmark44)

[Figure 4.2 Average preprocessing effort. Top two are original points and heat map, next two](#_bookmark48) [are with Average preprocessing points and heat map, then are without Average](#_bookmark48) [preprocessing Monte Carlo points and heat map and Bottom two are after Average](#_bookmark48) [preprocessing points and heat map 43](#_bookmark48)

[Figure 4.3 Final output of E4-03 in 14:30 Feb/15 44](#_bookmark50)

[Figure 4.4 Final output of E5-03 in 14:30 Feb/15 45](#_bookmark51)

# LIST OF TABLES

[Table 1.1.data structure of localization records 3](#_bookmark4)

[Table 2.1 contrast of four kinds of interpolation method 17](#_bookmark17)

[Table 3.1 error distance of different places (meters) 24](#_bookmark26)

[Table 4.1 contrast between with and without Average preprocessing 44](#_bookmark49)

# LIST OF ABBREVIATIONS

|  |  |
| --- | --- |
| **Abbreviation** | **Description** |
| WLAN | Wireless local area network |
| GPS | Global positioning system |
| RSS | Received signal strength |
| LOS | Line of sight |
| GUI | Graphical user interface |

## CHAPTER 1 INTRODUCTION

### Background

There is energy waste in many buildings today. For example, after class there is a few students in the classroom, but the lights and air conditioning are already over-supply, it is possible to save energy by modification if we know the number of students. Manual counting is the traditional way to count the number of people. However, it requires high labor costs and there is a problem that accuracy is decreased under a crowded environment. *Therefore, there have proposed a various methods for counting people or estimating the degree of congestion automatically by using cameras, sensors, and devices attached on people such as RFID and smart phones*.

On the other hand, with the popularity of smart phones, more and more buildings have WiFi. For example, shopping malls, office buildings, airports, subway stations, and campuses all have WiFi coverage. There are more and more WIFI Access point and served for more and more clients in recent years. In NUS campus, it has constructed more than 4,000 Access point in campus to provide Internet connection.

With this development the ability to estimate the level of utilization in a particular environment is available based on WIFI infrastructure. If this can be achieved accurately and cost-effectively, such metrics can be very useful for a variety of buildings to save energy.

### Motivation and Challenges

This project aims to develop a flow analysis system that operates in a WIFI-based environment using access point monitoring number of devices like phone, laptop. we have considered to use the localization information by CISCO system which are on received signal strength indication (RSSI) at devices from an WIFI access point for estimating the number of people in an area. The information on RSSI provided by CISCO WIFI access point will provide the localization of devices by longitude and latitude as well as some extra information like localization time etc. And in this research, we assume that every people take one device that can be localization by access point. We use this existing WIFI networks for estimating the number of people. Therefore, low-cost people counting can be implemented.

In this report, as the first step of our research, I design a friendly UI and propose methods to automaticly download and extract csv file from datacommons in chapter 2. Then to preprocess data and analyze data to get user dwell time and flow rate in chapter 3. Finally, in chapter 4, to write data into MySQL and use Tableau connecting MySQL to show the result.

### Data structure

A rough introduction of data structure and situations are as following:

1. Data structure

For this project I use history data (which means devices records in previous times of the same places) from NUS datacommens CISCO localization system. These data are as following:

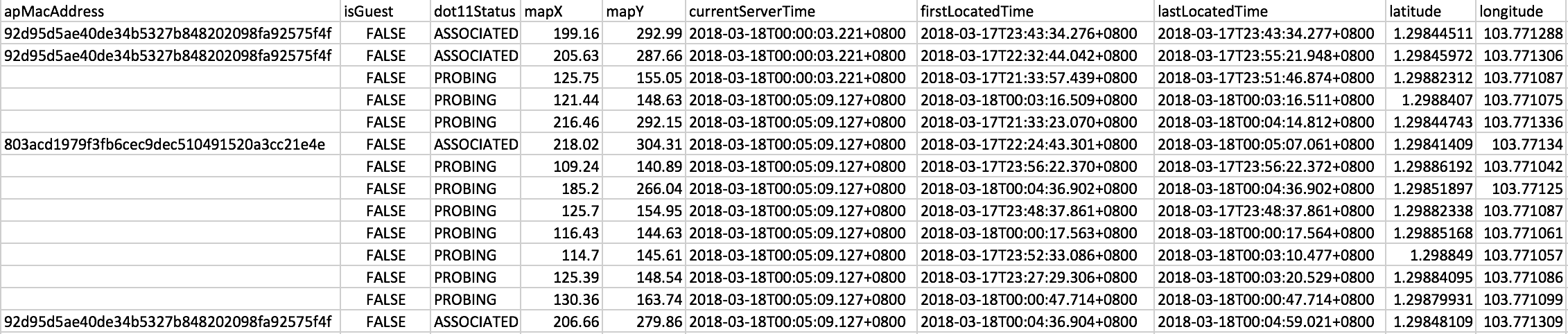


Table 1.1.data structure of localization records

First, we need to get knowledge of this data columns meaning. Refer to the CISCO Community the columns meaning is as following:

* 1. The first column ‘time’ is the time that the server records the data. This column depends on the server time stamp and isn’t totally same as current time. That means when a device is localized by CISCO WIFI system, the information of this device may cannot be updated immediately in system space ;
  2. The second column ‘macaddress\_hash’ is the hashed mac address of device by SHA encryption. For privacy of devices owner, we use SHA encryption to encode the MacAddress of device which is impossible to decode directly;
  3. The third column ‘isTracked’ is the state whether the device can be tracked by server.
  4. The fourth column ‘confidence’ is generate by Cisco system. With every calculated location (say x1, y1), a confidence factor (err\_ft) is returned. This is a floating point scalar used to calculate 95% confidence square. The device is estimated to be inside the square centered at (x1, y1) with sides 2 x (err\_ft) with 95% confidence;
  5. The fifth column ‘ip’ is the IP4 address of devices;
  6. The sixth column ‘username’ is the account number of device;
  7. The seventh column ‘ssid’ is the identity of device, containing: NUS, NUS\_2-4GHz, NUS\_STU, NUS\_STU\_2-4GHz, NUSOPEN, NUS\_Guest, eduroam and blank. The blank means CISCO system fails to identity the network by too short communication time;
  8. The eighth column ‘band’ is used in CISCO system now;
  9. The nineth column ‘apMacAddress’ is the Access Point that localized the device. Its value is the nearest AP mac address from device and hashed with SHA encryption same as device mac address.
  10. The tenth column ‘isGuest’ is whether its identity is guest;
  11. The 11st column ‘dot11Status’ indicates connection status between NUS WIFI and devices. It contains 3 states: UNKNOWN, PROBING, which means the device can’t communicate by NUS WIFI, but it is connected to NUS WIFI. ASSOCIATED, which means the device can communicate by NUS WiFi. Only in ASSOCIATED status, can CISCO system get ip, username, ssid, apMacAddress of devices.
  12. The 12st ‘mapX’ is the offset of image in longitude in inches, which is calculated by device longitude and image longitude;
  13. The 13st ‘mapY’ is the offset of image in latitude in inches, which is calculated by device latitude and image latitude;
  14. The 14st column is the ‘currentServerTime’ which shows the time when the record package update to server;
  15. The 15st column is the ‘firstLocatedTime’ which shows the time when this record generated or when the device is first associated with CISCO WIFI system. It will be regenerated if a device lose connection with WIFI for one hours;
  16. The 16st column is the ‘lastLocatedTime’ which shows the time when the record last updated. The column will change in several situations as following:
      1. The device move beyond the threshold of WIFI system;
      2. The device move from the zone of an AP to another;
      3. The localization record interval between currentServerTime and lastLocatedTime time exceeds 15 minutes;
  17. The 17st is the localization data in latitude.
  18. The 18st is the localization data in longitude.

### 1.4 Tools introduction

1. Python

For programming beginner, Python is a good choice because of its friendly, concise, easy-to-learn features. Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.

Figure1.1 Python icon

1. MySQL
2. Tableau

## CHAPTER 2 HEAT MAP

Before we learn how to apply some algorithms to solve problem above, we need to design a better tool to help us to visualize the crowd density in the floor by data and floor plan above and show the effect of our algorithms. In this report we select heat map to show these things above.

### Introduction of heat map

*A heat map (or heatmap) is a graphical representation of data where the individual values contained in a matrix are represented as colors. The term 'heat map' was originally coined and trademarked by software designer Cormac Kinney in 1991, to describe a 2D display depicting financial market information*[1] *though similar plots such as shading matrices have existed for*

*over a century* [2]. The reason why we want to use heat map to show the crowd density is that it is intuitive and easy to understanding. We can immediately know the high density area with hot color (red etc.) and low density area with dark color (dark blue etc.).

### History of heat map

*The company that acquired Kinney's invention in 2003 unintentionally allowed the trademark to lapse*.[3]

*Heat maps originated in 2D displays of the values in a data matrix. Larger values were*

*represented by small dark gray or black squares (pixels) and smaller values by lighter squares.*

*Loua (1873) used a shading matrix to visualize social statistics across the districts of Paris.*[2] *Sneath (1957) displayed the results of a cluster analysis by permuting the rows and the columns*

*of a matrix to place similar values near each other according to the clustering. Jacques Bertin*

*used a similar representation to display data that conformed to a Guttman scale. The idea for joining cluster trees to the rows and columns of the data matrix originated with Robert Ling in 1973. Ling used overstruck printer characters to represent different shades of gray, one character-width per pixel. Leland Wilkinson developed the first computer program in 1994 (SYSTAT) to produce cluster heat maps with high-resolution color graphics. The Eisen et al. display shown in the figure is a replication of the earlier SYSTAT design[4]*

There are several kinds of heat maps:

* Web Heat Maps are usually used for displaying areas of a Web page that is most frequently clicked by visitors. Web Heat Map is often used with other forms of web analytics and session replay tools.
* Biology Heat Maps for genes across a number of different samples (e.g. cells in different states, samples from different patients) are typically used in molecular biology to show the level of expression.
* The tree map is a 2D hierarchical partitioning of data that visually resembles a heat map.
* A mosaic plot is a tiled heat map for representing a two or more ways table of data.

As with tree maps, the rectangular areas in a mosaic plot are hierarchically organized, which means that the regions are rectangles instead of squares. Friendly (1994) first use this graph.

* A density function visualization is a heat map for representing the density of dots in

a map. It enables one to perceive density of points independently of the zoom factor. *Perrot et al (2015) proposed a way to use density function to visualize billions of dots*

*using big data infrastructure with Spark and Hadoop*.[5]

For the characteristics above we select density function visualization heat map to show crowd density.

### How to generate density heat map

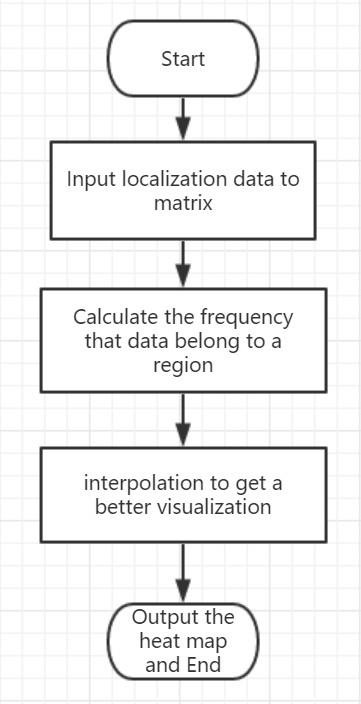
To create a density heat map we use MATLAB, which is a powerful software to do both mathematic calculation and map drawing. There are serval steps to create this heat map by MATLAB as following:

Figure 2.1.The flowchart of generating heat map

1. Input localization data to matrix

Before draw heat map, we need to know what kinds of data we can use to draw it. There are two pairs of data we can use to draw it :< longitude, latitude> and

<mapX, mapY>. In this report we will use < longitude, latitude> because it is independent with original point while <mapX, mapY> is highly dependent with original point.

Then we need to take time into consideration. There are four kinds of time stamps

we can use: Time, Current Time, First Located Time, Last Located Time. For the definition of crowd density we should select data within same current time. These data are localized at same time interval.

MATLAB can draw heat map by matrix, but the data format of devices is .csv file for EXCEL. We need to use command ‘xlsread’ in MATLAB to load .csv files into

MATLAB workspace. The entire command is as following:

[N R T]=xlsread('e5.xlsx');

xx is the name of .csv document. By xlsread function MALAB can load 3 kinds of data. N is data in double precision. MALAB will try to transform all cells of .csv file into double precision. But it cannot transform string cell into double precision data. It will be transform into NA instead. R is same as N, but it is integer format. And T is string format.

But it longitude and latitude is a large unit for floor plan. One degree of longitude or latitude is hundreds kilometers. We have to convert them into meters which are a proper unit to describe floor plan.

The method is simple:

meters(longitude) = longitude × 110km meters(latitude) = latitude × 112km

The ratio of longitude and latitude is based on the position of NUS.

1. Calculate the frequency that data belong to a region

After we get the input data, the next step is to divide the entire floor into small areas to calculate the frequency that data belong to it and draw heat map by this frequency. Because the heat map in MATLAB is still a matrix, we have to transform the data into

frequency matrix. We use an example to show this process as following:

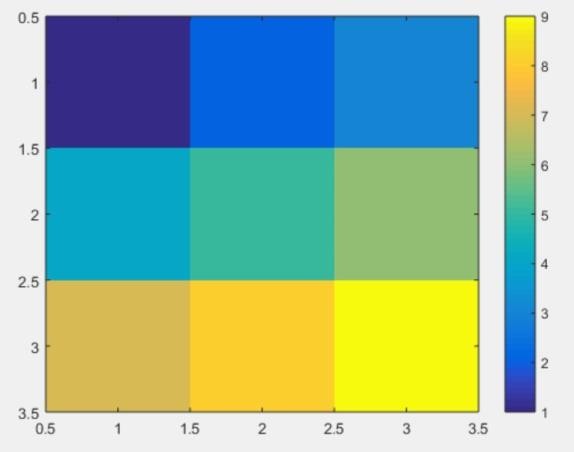
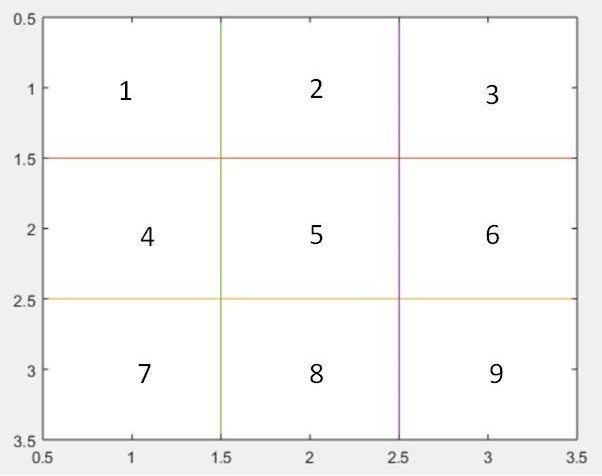
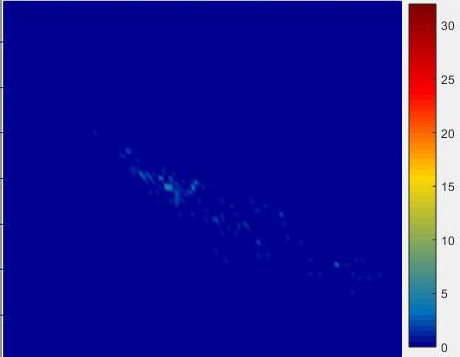
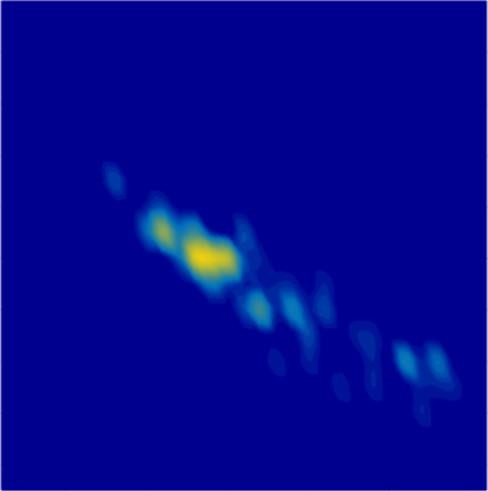


Figure 2.2 The frequency map(left) and the heat map(right)*.*

MATLAB provides function to transform frequency map in left to heat map in right. We can easily find that high value area (which represent high crowd density area) will be assigned lighter color (right bottom) while the area with low value (which means low crowd density area) will be given darker color (top left). However, the frequency map in left has to be in suitable area size. The heat map with different area size are as shown Figure 2.3:



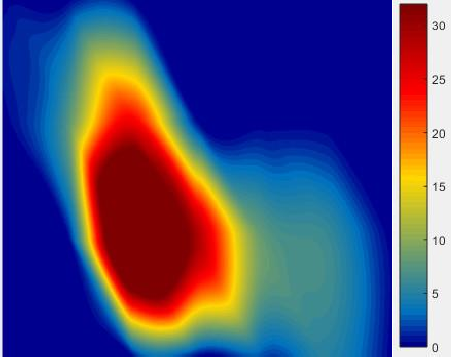


Figure 2.3 Heat map with different area size for same frequency matrix. Too small size (Top), suitable size (Mid) and too large size(Bottom)

We can find that heat map with too small area size cannot show any density information while heat map with too large area size will show a really rough description of crowd density. Only heat map with suitable area size would show crowd density map in a correct way.

There are two ways: empirical and experimental.

For E1-06, E4-03, E5-03 the width and length of these three floors are tens of meters. That means we can select 5 meters or 10 meters to be the suitable area size by empirical assumption.

Then we do experiment to find the best size. We select from area size from 1 meter to 10 meters. The best heat map area size is 4 meters.

1. interpolation to get a better visualization

By method above we already can draw an acceptable heat map. However it would be strange for other people to find that the heat map is a lot of rectangle with discontinuous color, which is not same as the heat map we usually see. That is because the data of heat map is not continuous. We can use interpolation function provide by MATLAB to create a continuous heat map.

The function is as following:

griddata(x,y,z,X,Y,'cubic');

Where x is the longitude in meters of floor plan. Low-case y is the latitude in meters of floor plan. Low-case z is the crowd density of each area. Capital X the interpolation size in x axes. Capital Y the interpolation size in y axes. And ‘cubic’ is the interpolation method we select.

There are 4 kinds of interpolation method we can select: a)Nearest Neighbor

When the interpolation method option is ‘nearest’ then the interpolation method would become Nearest Neighbor method. Its interpolation result is as following: Eg1. Nearest Neighbor

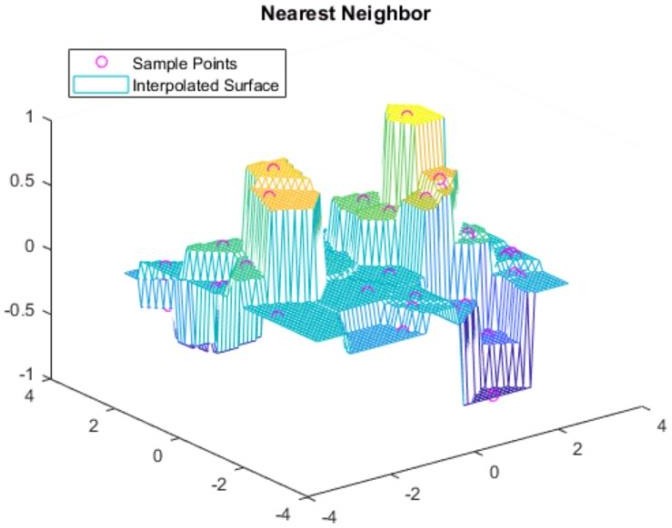


Figure 2.4 Nearest Neighbor Interpolation

We can find that it although it do some interpolation, it still discrete. However, this method is the fastest method to interpolation.

* 1. Liner

When the interpolation method option is ‘linear’ then the interpolation method would become linear method. Its interpolation result is as following:

Eg2.Linear

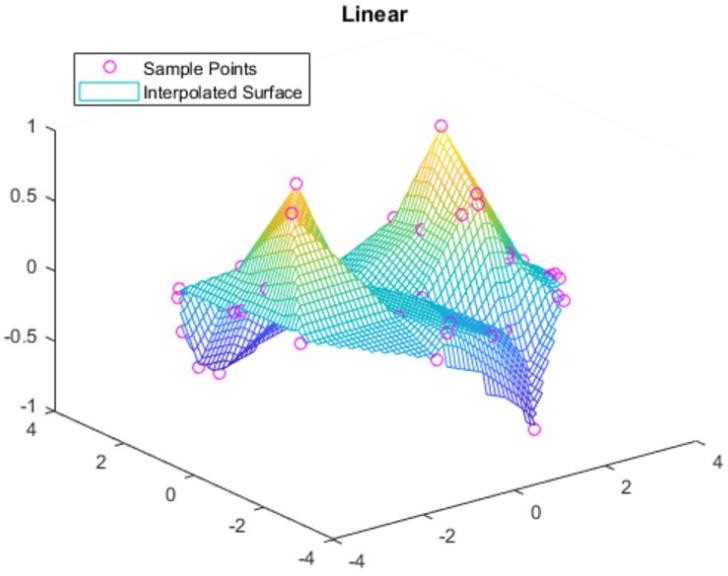


Figure 2.5 Linear Interpolation

We can find that it is much better that Nearest Neighbor. And it still fast.

* 1. Nature Neighbor

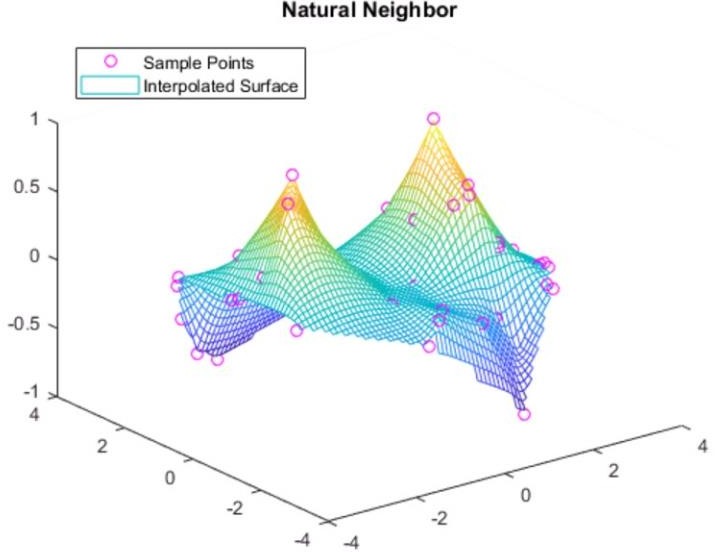
When the interpolation method option is ‘natural’ then the interpolation method would become linear method. Its interpolation result is as following: Eg3.Natural Neighbor

Figure 2.6 Natural Neighbor Interpolation

Natural Neighbor will not just assign interpolation value as nearest neighbor. Instead it will give the interpolate points with a balance value between two points. It is in middle speed but it performance is better.

* 1. Cubic

When the interpolation method option is ‘cubic’ then the interpolation method would become linear method. Its interpolation result is as following:

Eg4.Cubic

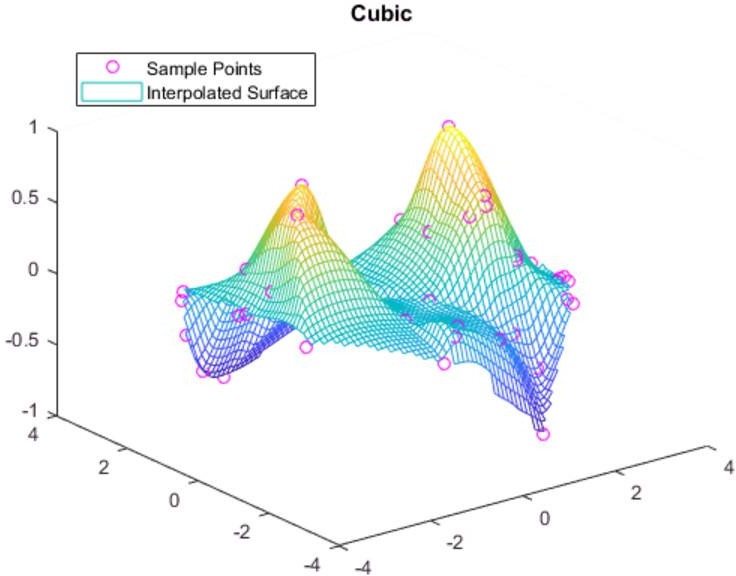


Figure 2.7. cubic Interpolation

We can find that it is almost same as natural neighbor. But Cubic will use polynomial function to balance the difference between points while natural neighbor will use linear balance. The speed of cubic is a little slow than natural neighbor.

To get a good performance of heat map interpolation and speed we select cubic method to do interpolation. The speed to generate a heat map for this method is 5 seconds and it provides a great performance for heat map.

|  |  |  |
| --- | --- | --- |
| **Method Contribution** | **Speed** | **performance** |
| **Nearest Neighbor** | Fastest | Bad |
| **Linear** | Fast | Middle |
| **Natural Neighbor** | Middle | Good |
| **Cubic** | A little slow | Perfect |

Table 2.1 contrast of four kinds of interpolation method

## CHAPTER 3 MONTE CARLO METHOD

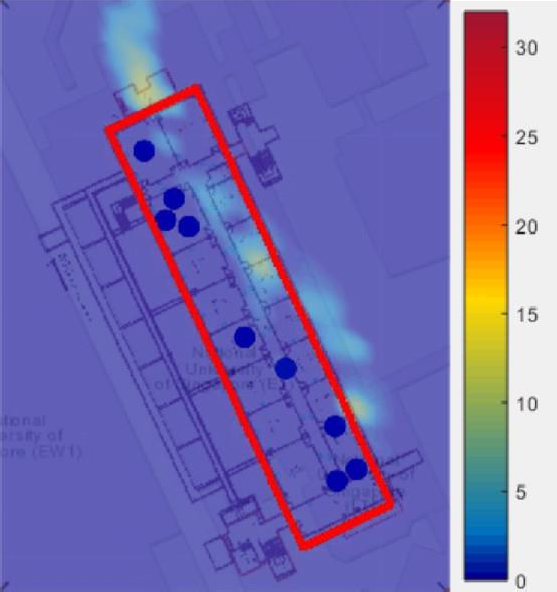
Before introduce the Monte Carlo Method we show the original heat map by method above for all three places as following Figure 3.1:

Figure 3.1 Original data points (Top) and heat map (Bottom)

The main problem of both original data points map and heat map is that a lot of points exceed the border of building which is unacceptable. Our mission is to correct them into building while keep the features of localization points.

To finish this mission we need to know the localization method.

### Received Signal Strength (RSS)

RSS technology has been widely implemented in the wireless-communication network. Basically, the RSS technique generate localization data by the fact that the radio frequency (RF) signal attenuate during propagation. *In the free space situation, the RSS has been proved to be linearly proportional to the inverse square of*

*distance between transmitter and receiver as shown in Friis equation* [7] below:

𝜆 2

𝑃𝑟(𝑑) = (4𝜋𝑑)

𝑃𝑟 𝐺𝑡𝐺𝑟

Where: P 𝑟 (d) = received power at distance d away from transmitter

𝐺𝑡= transmitter antenna gain

𝐺𝑟= receiver antenna gain

𝑃𝑟 = transmitted power

λ = wavelength of the transmitted signal in meter

However, in practice volatile environment dependent factors such as reflection, refraction, and multipath effect affect the signal propagation process and lead to the RSS precision. Usually these effects are modelled by a log-distance path-loss where received signal strength decreases logarithmically with the distance extension.

𝑃𝑟(𝑑) = 𝑃0 − 10𝛽𝑙𝑜𝑔10(𝑑)

Where: 𝑃𝑟(𝑑)= received power at distance d apart from transmitter

𝑃0= signal power at 1 meter apart from transmitter

β = path-loss exponent (range from 2 – 4 in typical indoor environment [3]) d = distance between transmitter and receiver

These above are the basic algorithm for localization based on WIFI system. Then based on these information we can do indoor localization by the unsupervised training phase and online testing phase.

The main steps are as following:

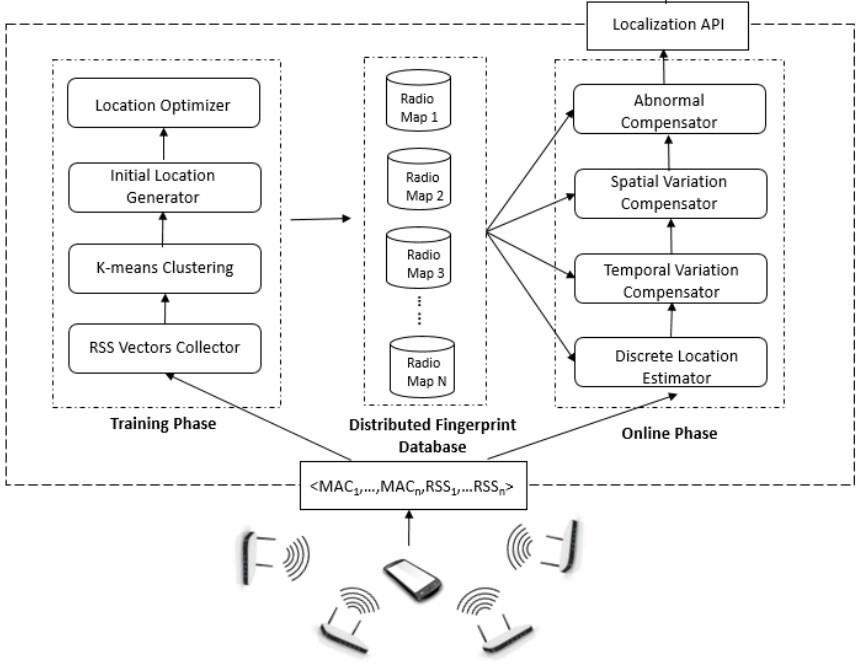


Figure 3.2 RSS localization method

We will not go into details of above method. But these RSS localization method will provide us with some idea of how to correct points into building while keep its location features.

By method above we can find that the localization of a device is generated by RSS Strength between device and WIFI Access point. That means the reason why a lot of localization points exceed the building border is the error of system.

### Error distribution

Next we do analysis for the system. Consider a normal system with error. The error usually contains two parts:

1. System error, which works as offset for system output. It is same for all output;

𝑦𝑜𝑢𝑡𝑝𝑢𝑡 = 𝑦𝑟𝑒𝑎𝑙 + 𝑒𝑠𝑦𝑠𝑡𝑒𝑚

Where 𝑒𝑠𝑦𝑠𝑡𝑒𝑚 is a constant number. In RSS localization system it is because of

some fading or obstruction which is almost same for all output in a small zone.

1. Casual error, which generally is Gaussian distribution for different output;

𝑦𝑜𝑢𝑡𝑝𝑢𝑡 = 𝑦𝑟𝑒𝑎𝑙 + 𝑒𝑐𝑎𝑠𝑢𝑎𝑙

Where 𝑒𝑐𝑎𝑠𝑢𝑎𝑙~𝒩(μ, σ) Gaussian distribution with mean μ.and variance σ. In the system it is because system essential error.

However we do not know whether this model fits the localization system. So we do experiment in E1-06, E4-03 and E5-03 to learn the pattern of error distribution of NUS.

*The experiment design is as following: Experiment*

*Time: 2017/Feb*

*Duration: 2 hours respectively Location: E1-06, E4-03, E5-03*

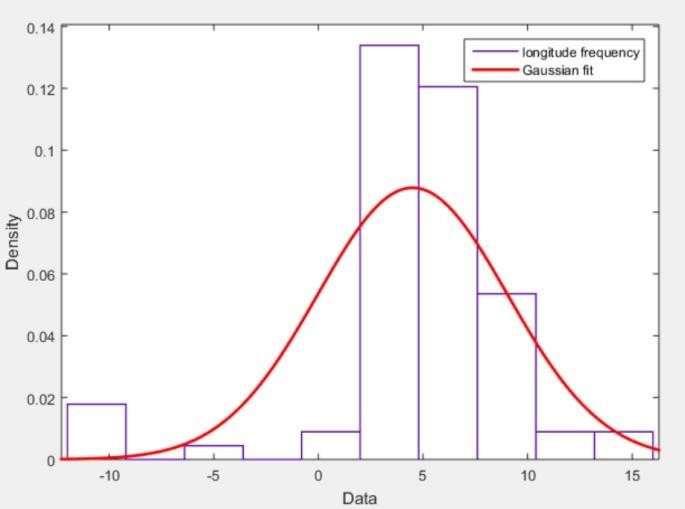
*Experiment description:*

*Stay at E1-06, E4-03, E5-03 for two hours respectively, shut down the WIFI and restart of device WIFI to collect ground truth data and repeat.*

*Purpose:*

*To test if a localization data generated by a stationary device obeys Gaussian distribution.*

The Experiment result is as following:



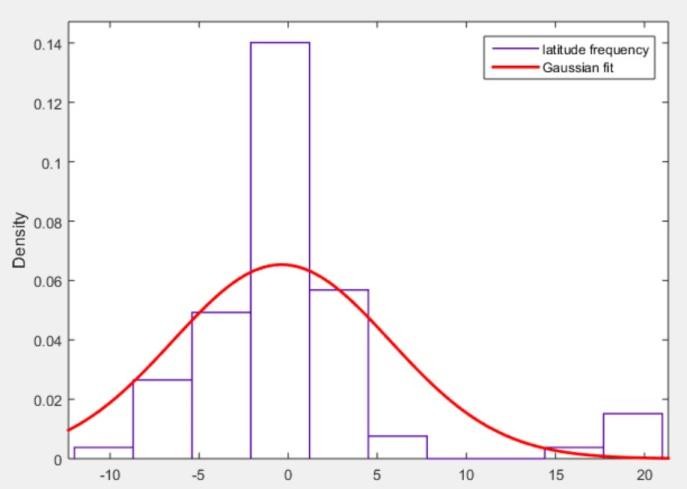
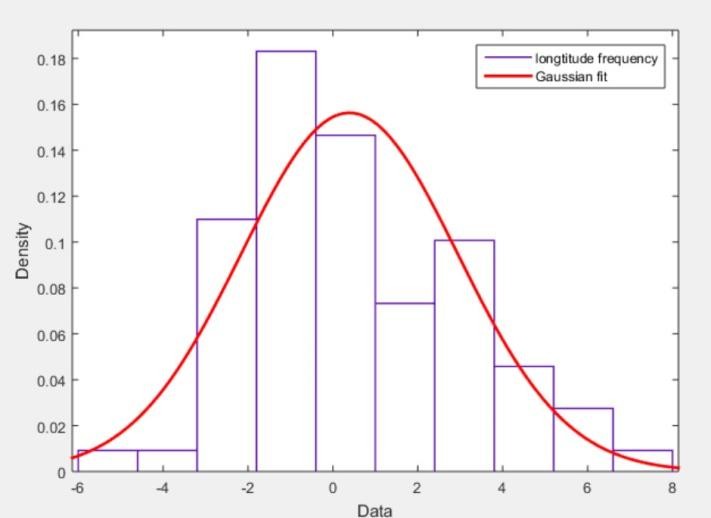
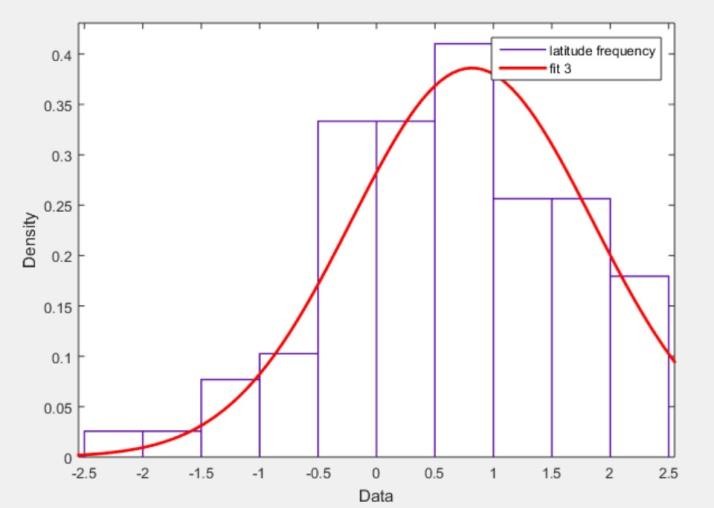
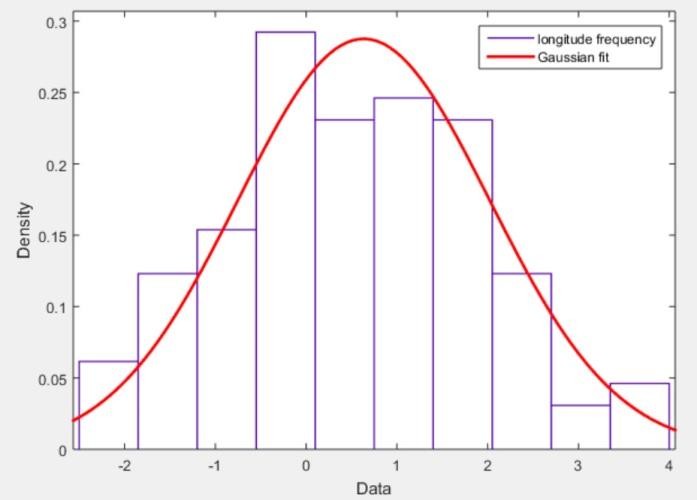


Figure 3.3 The error distance between real position and localization position in E1-06. Top one is the longitude and bottom one is latitude in meters.

It shows that the error distribution in E1-06 is Gaussian distribution with offset, which is fit for normal system. Similarly the result in E4-03, E5-03 is also Gaussian distribution:



Figure 3.4 he error distance between real position and localization position in E4-03. Top one is the longitude and bottom one is latitude in meters

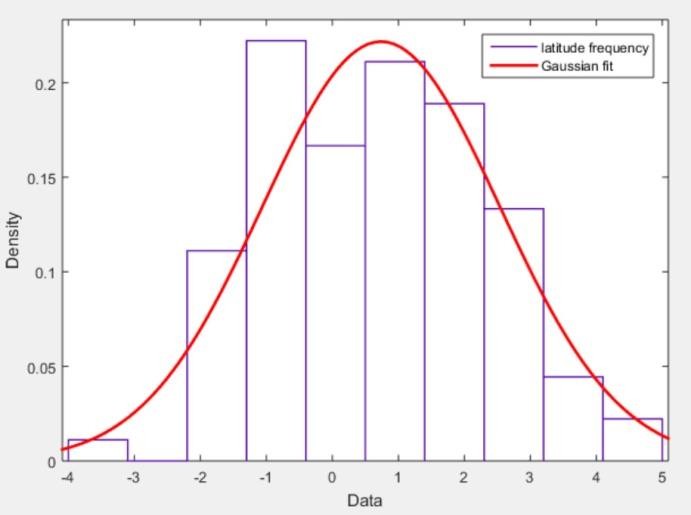


Figure 3.5 The error distance between real position and localization position in E5-03. Top one is the longitude and bottom one is latitude in meters

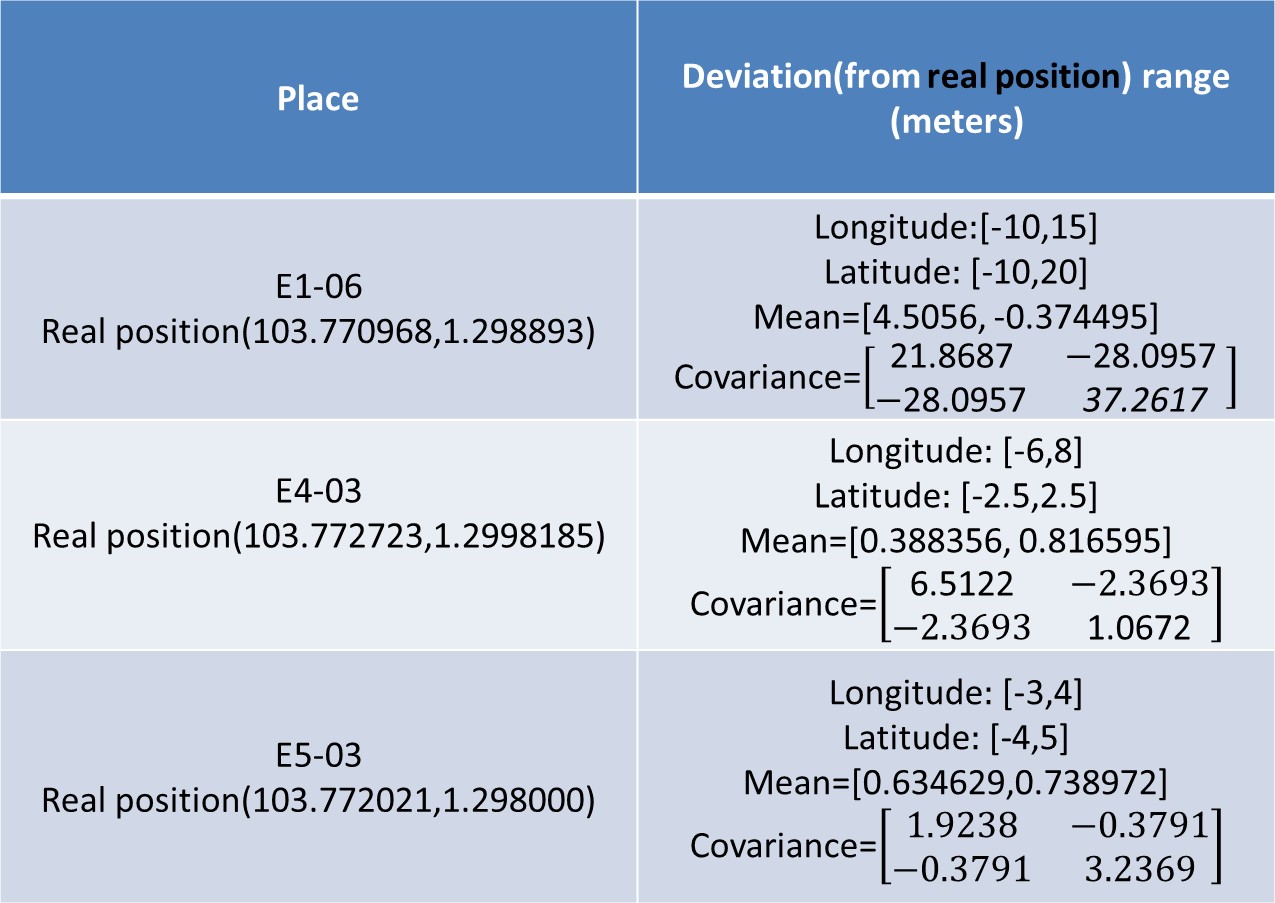


Table 3.1 error distance of different places (meters)

To confirm the distribution of error, we cannot just use observation. We have to use algorithm to test if the distribution is Gaussian distribution. MATLAB provide test function called lillietest. We use command to do lillietest:

[H,P, KSTAT, CRITVAL]=lillietest(x,alpha)

Where h is whether the assumption that the examples are Gaussian distribution. When H=0 it means the assumption is acceptable;

P is the probability that the sample belong to Gaussian distribution. The max probability is 0.5;

KSTAT is test statistic KSTAT= max|S(x) - CDF|;

CRITVAL is the he critical value CRITVAL for the test. if KSTAT > CRITVAL, the null hypothesis can be rejected at a significance level of ALPHA

One of the result is as following:

Eg. Lillietest in E4-03 latitude

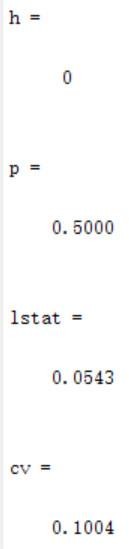


Figure 3.6 lillietest result for E4-03 latitude

We can easily find that it is Gaussian distribution by *normplot* function in MATLAB as following:

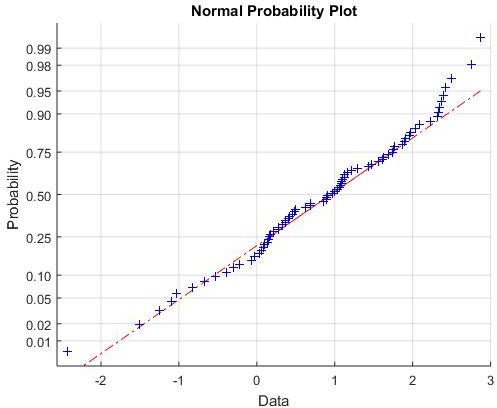


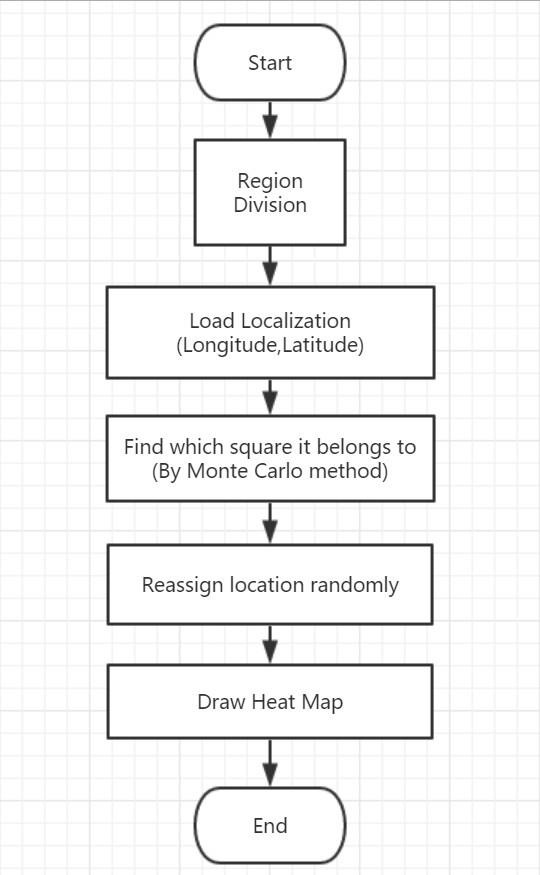
Figure 3.7 Normplot of E4-03 latitude*.*

The red line is the standard Gaussian distribution data while the blue points are points in E4-03. We can find that the points are almost same as standard Gaussian distribution which means that data points is highly possible belong to Gaussian

distribution.

By above experiment we can be confident that the localization error for a stationary device is Gaussian distribution. We can use this distribution to push the data points into building while keep its position feature.

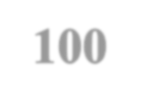
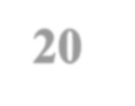
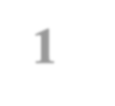
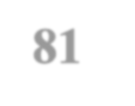
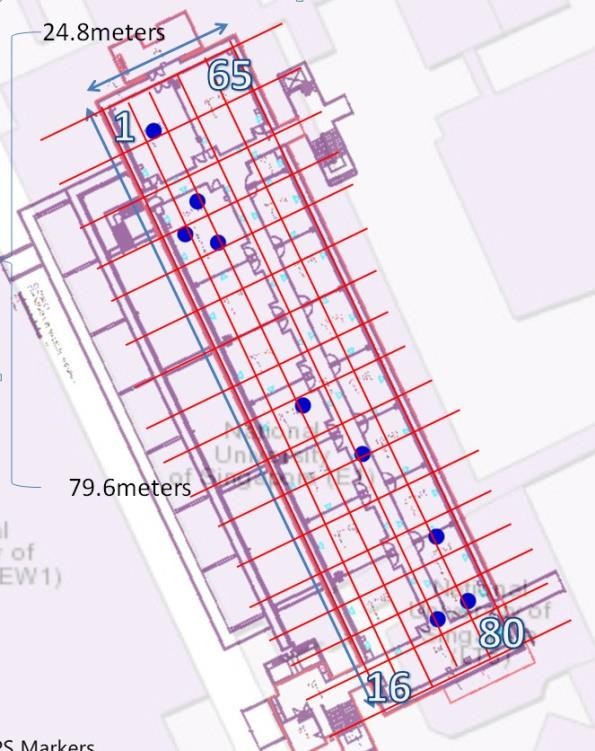
### Monte Carlo Method

From above we already know that the error distribution is 2-D Gaussian distribution. We can use algorithm to push the points into border. The steps are as following:

1. Region Division

Figure 3.8 Flow-chart of Monte Carlo Method

Because the best heat map area size is 4 meters mentioned by chapter 2 “heat map”, we divide the entire building space into 4 × 4 square meters small area. This division is as following:



1. Load Localization

Figure 3.9 Floor division

We first combine error distribution with the data points:

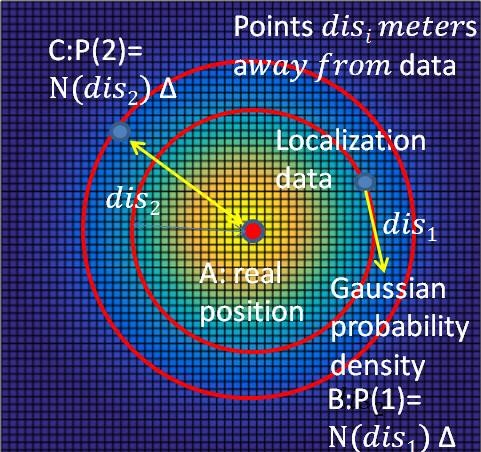
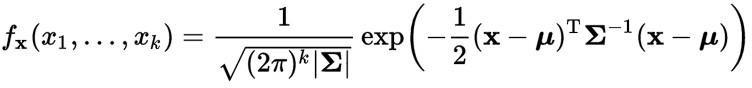
Suppose we have a device whose real position is at A, the error distribution is 2-D Gaussian distribution e~𝒩(x, Σ):

Figure 3.10 Real position situation with Gaussian distribution

Where 𝒩(x, Σ) is 2-D Gaussian distribution with 2-D mean x vector and

2 × 2 covariance matrix Σ. The multi-dimensional Gaussian distribution is:

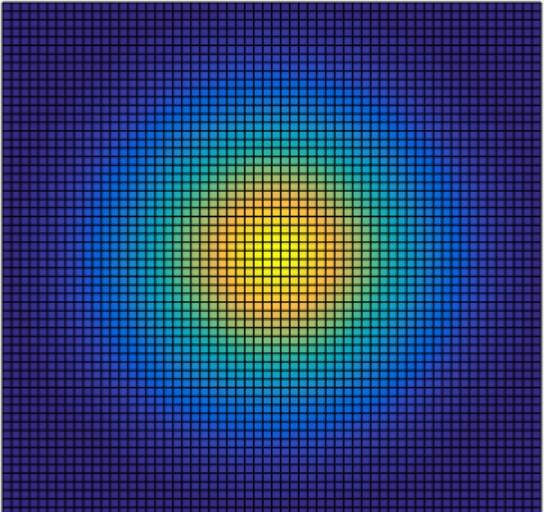


dis means the localization error from real position to localization points;

∆ is a very small distance, because we use Gaussian probability density;

P is the probability that the localization points appears at the place (B and C). We take this theory into consideration inversely:

Suppose we have a localization points at A, then the real position is 2-D Gaussian distribution as well because it have same probability density as above. So if we want to know the probability that the real position of localization point belong to area P, we just need to do integral of this Gaussian distribution in the area P as following:



Area P

Figure 3.11 Method to get real position probability

However, integral is slow process especially when the probability density function is complexed. Therefore we use Monte Carlo method instead:

1. Monte Carlo method

Monte Carlo methods (or Monte Carlo experiments) are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. Their essential idea is using randomness to solve problems that might be

deterministic in principle. They are often used in physical and mathematical problems and are most useful when it is difficult or impossible to use other approaches. *Monte Carlo methods are mainly used in three distinct problem classes: optimization, numerical*

*integration, and generating draws from a probability distribution* [8].

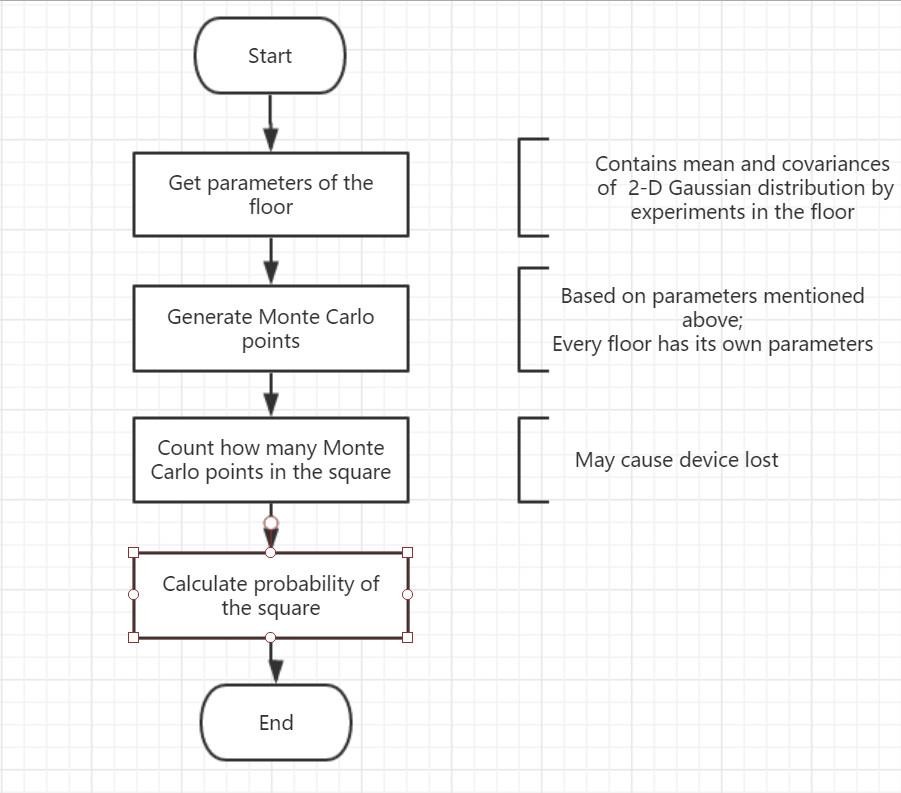
Monte Carlo method provides a powerful way to do integral for complexed probability density function. We implement this method to our model by the following steps:

Figure 3.12 Flow chart of Monte Carlo method

1. Get the parameters of the floor.

We use 2-D Gaussian distribution as Monte Carlo basic function. There are two parameters that need to be confirmed: Mean vector x and covariance matrix for longitude and latitude.

These parameters can get from curve fitting by the error distribution in chapter 3.2.

The characteristic that need to be mentioned here is that this mean vector x and covariance matrix Σ is unique for every floor plan because it has its own physical features. The parameters are shown at table 3.1 in chapter 3.2.

1. Generate Monte Carlo points

Since we already get the probability distribution by last step, we can generate Monte Carlo points by this distribution. MATLAB provide function to generate 2-D Gaussian distribution points with mean vector x and covariance matrix Σ as

following:

rd=mvnrnd(mu,sig,number)

where rd is the number × 2 matrix of Monte Carlo points; mu is the mean vector x;

sig is the covariance matrix Σ

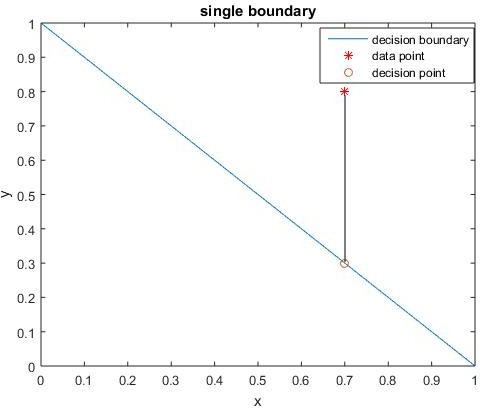
number is how many Monte Carlo points to generate. The large this number is , the accuracy Monte Carlo method is. But the computation cost is higher as well. To balance the accuracy and computation cost of Monte Carlo method we select number=100.

1. Calculate how many Monte Carlo points in the square

It is easy to calculate the number of points in an area or square manually. But since there are a lot of localization points and each one contains 100 Monte Carlo points, we have to design a program to do calculation automatically.

We can use Linear Programming method to do so:

Consider the simplest situation first to justify the area the point belongs to:



Area2

Area1

Figure 3.13 Single boundary linear program

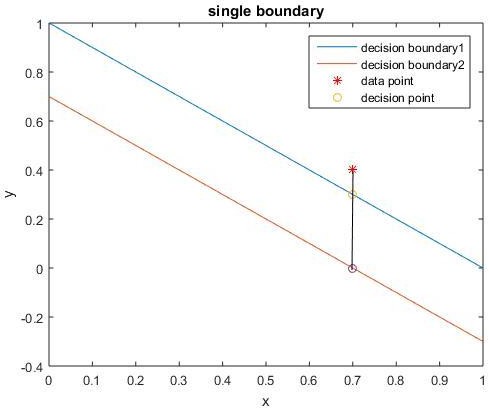
The red \* point is the point that we want to know which area it belong to;

The orange circle o is the decision point on decision boundary line. We will do judgement based on this point;

The blue line is the decision boundary which divide the entire space into 2 parts: Area1 and Area 2;

By linear program we can find that if 𝑦𝑝𝑜𝑖𝑛𝑡 > 𝑦𝑑𝑒𝑐𝑖𝑠𝑖𝑜𝑛 then data point belong to top area1, we give it a label +1; Otherwise if 𝑦𝑝𝑜𝑖𝑛𝑡 ≤ 𝑦𝑑𝑒𝑐𝑖𝑠𝑖𝑜𝑛 then data point belong to bottom area2, we similarly give it a label -1;

For multiple decision boundary is same:



Area3

Area2

Area1

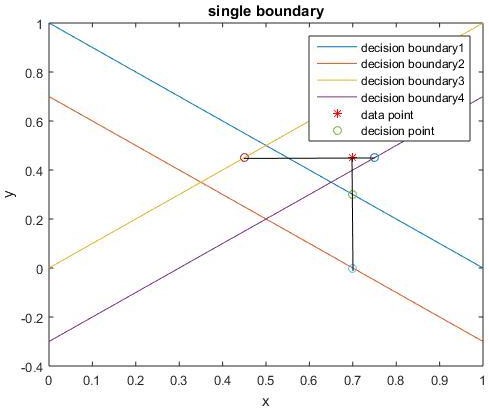
Figure 3.14 multiple boundary linear program

This time there are 3 areas. We can justify which area the point belong to by label sequence as following:

For decision boundary1 we have label +1 and for decision boundary2 we have label

+1 by processing as single boundary. That mean the point is at the top half surface of both decision boundary1 and decision boundary2. Therefore we know that it belong to Area3. Similarly if the point belong to Area2, we can get label sequence as [+1,-1] and for Area1 it is [-1,-1];

Then for boundary with different normal vector we can repeat this process. The difference is just the label sequence should be 2-dimension.



Area5

Area4

Area6

Area3

Area9

Area7

Area2

Area8

Area1

Figure 3.15 multiple dimension multiple boundary linear program

We repeat the process above for 2-D label sequence. The result is as following:

Pointsequence = (+1 +1

)

−1 +1

This label sequence means ( 𝑡𝑜𝑝 𝑜𝑓 𝑏𝑜𝑢𝑛𝑑𝑎𝑟𝑦1 𝑡𝑜𝑝 𝑜𝑓 𝑏𝑜𝑢𝑛𝑑𝑎𝑟𝑦2) which

𝑏𝑜𝑡𝑡𝑜𝑚 𝑜𝑓 𝑏𝑜𝑢𝑛𝑑𝑎𝑟𝑦3 𝑡𝑜𝑝 𝑜𝑓 𝑏𝑜𝑢𝑛𝑑𝑎𝑟𝑦4

is Area 6. Then we can give it a final label as <datapoint, 6> which means <the

𝑖𝑡ℎ𝑑𝑎𝑡𝑎 𝑝𝑜𝑖𝑛𝑡,label 6>.

And we can repeat this process for more boundaries.

Finally, after determine all labels of Monte Carlo points for a single data point, we can know the probability that point I belong to square (area) j as 𝑃𝑖,𝑗. This 𝑃𝑖,𝑗 will be use to assign new location for data points.

1. Reassign location randomly

Suppose we have a data point i, and its probability 𝑃𝑖,𝑗 for data point i belong to square j, we cannot say that the crowd density in square j is 𝑃𝑖,𝑗 which is not a integral number because it has no physical meaning. What we need to do is reassign a most possible place for this point.

The most common method is that select the square(area) with highest probability to

be the area the data point belong to and reassign a location randomly to this data point to avoid overlap. We can show this procedure by an example as following:

Eg. Suppose we have a data point i, and two square (area) j,k that:

𝑃𝑖,𝑗 = 0.4, 𝑃𝑖,𝑘 = 0.6

Then we select k to be the square that device i belong to and reassign a location randomly in square k to be the new location for data point i.

In MATLAB it is easy to do: First, we use maximum function:

[M,I] = max(P)

Where M is the maximum value of probability vector; I is the Index of vector;

P is the probability vector for data point i.

Then we load the information of the square (center and width) to generate new location by function:

Newlocation = center(k) + width × rand(2,1) Where center (k) is a 2 × 1 vector for the center of square k; Width is 2 meters which is half of square width;

Function rand (2,1) will generate a 2 × 1 vector which belong to (0,1); By above we finish the reassignment of data point location.

1. Draw heat map

We already introduce the method to heat map in chapter 2. The output is as following:

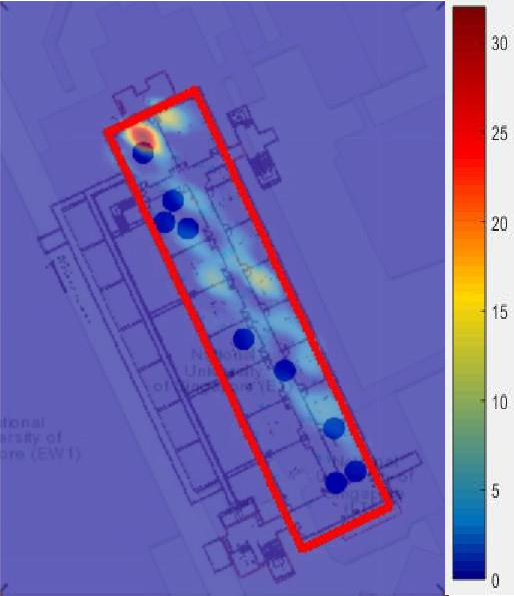
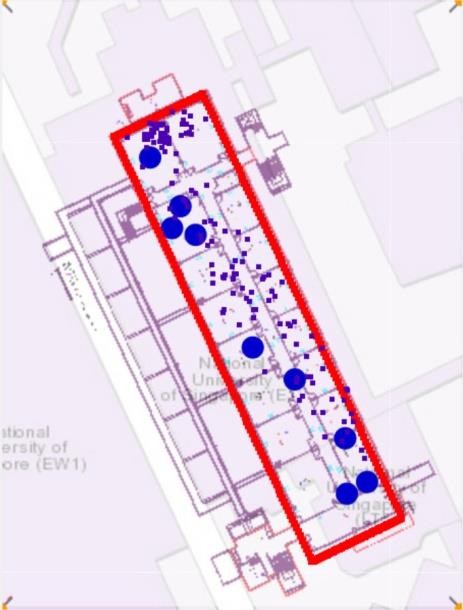
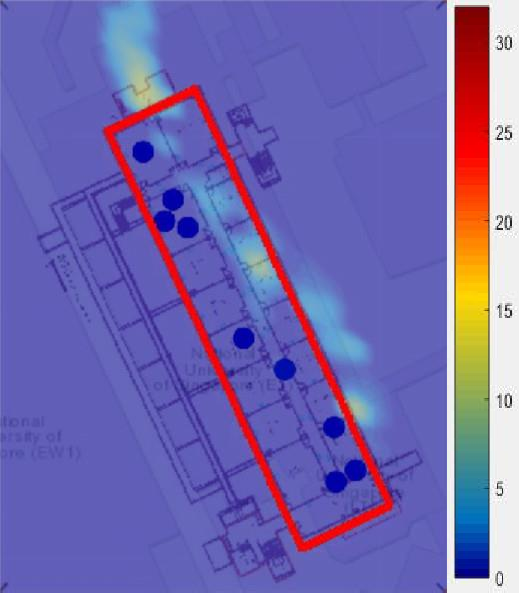


Figure 3.16 The data point map(top two) and heat map(bottom two) of E1-06 before Monte Carlo method(left two) and after Monte Carlo method(right two)

Similarly we repeat this process for E4-03 and E5-03:



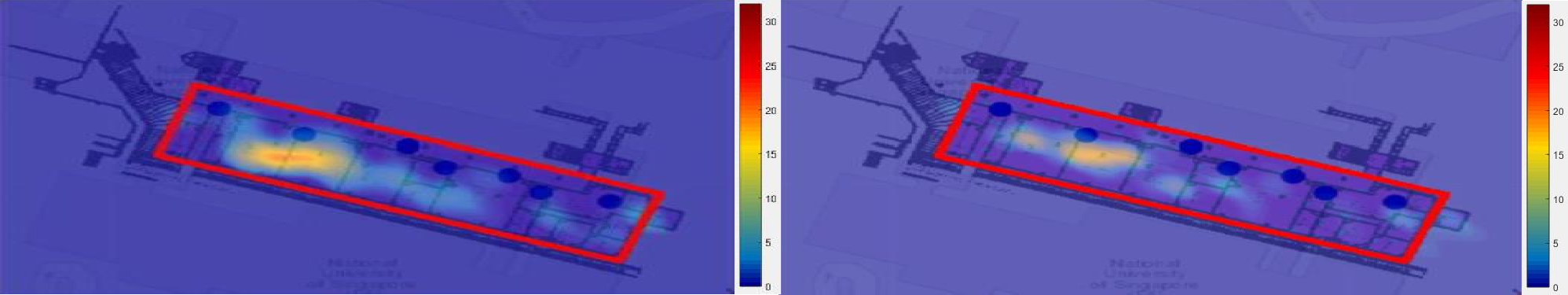


Figure 3.17 The data point map(top two) and heat map(bottom two) of E4-03 before Monte Carlo method(left two) and after Monte Carlo method(right two)

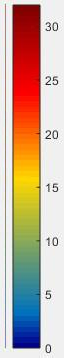
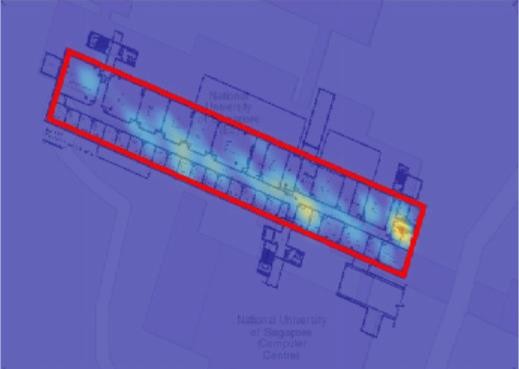


Figure 3.18 The data point map(top two) and heat map(bottom two) of E5-03 before Monte Carlo method(left two) and after Monte Carlo method(right two)

We can find that the effect of algorithm push the data points into building successfully while keep their position feature. By this heat map we already can get a clear crowd density conception in a certain time.

However this algorithm has a problem: “device lost”

Suppose we have a localization point that far away from the building as following:

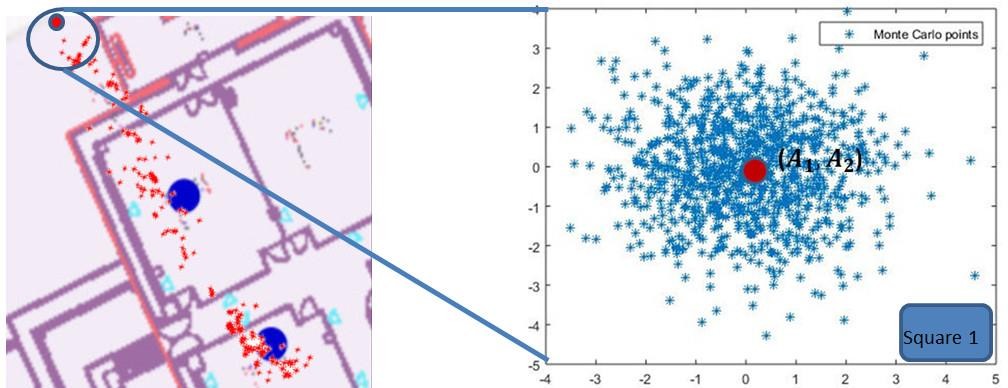


Figure 3.19 Bad situation for Monte Carlo method

By result that the localization of this device could be 0 to 30 meters away from real position, which sometimes will lead to bad performance of probability method.

Eg. In 18:13 there are 159 devices in E1-06, but after Monte Carlo method the total devices in heat map is only 149 devices. Some devices lost because of this method.

How could we improve our model to solve this problem?

Predict real position not only just by live data but also by history data. Although both history data and live data are dirty data, we can use them to clear the current live data as showing in next chapter.

## CHAPTER 4 AVERAGE PREPROCESSING

To solve this problem we first recall the reason why this problem appears. It is because some of the data points are really far sway form the building. That means if we can apply some preprocessing algorithm to push these data points to a place closer to the building and then use Monte Carlo method, the problem can be worked out.

We design new preprocessing algorithm still based on the error distribution. The Gaussian distribution of error distance reminds us with the idea to use more data for the same device to estimate the real position of this device. There are three question need to be solved to design the preprocess algorithm.

First, we need to know what kind of history data we can use to do preprocess.

When a device is moving, it is impossible to clear localization data by history data because the time snap for localization data is 1 minute. This period of time is enough for a person to walk through the localization zone to another.

But stationary device’s localization data can be corrected by history data because it do not move. So the error distance distribution should same as Chapter I, which is Gaussian distribution. Before we use these data to clean current localization data, we have to justify whether a data is from stationary device or moving device.

### Stationary device judgment

In chapter 1 we have already shown that every localization data contains a column called ‘APmacaddress’ which means the Access point that the device are communicating with. We have mentioned that the device will only connect to the nearest AP as well. So if the device connected to another AP in last time snap, the

device must be moving.

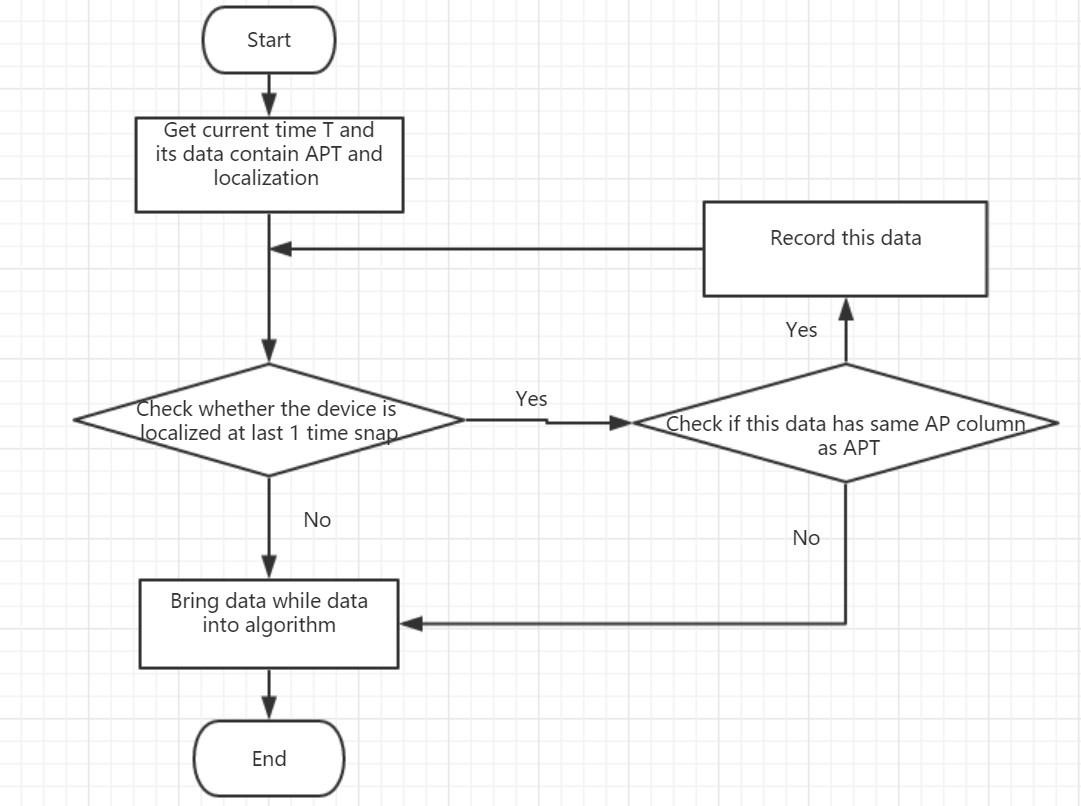
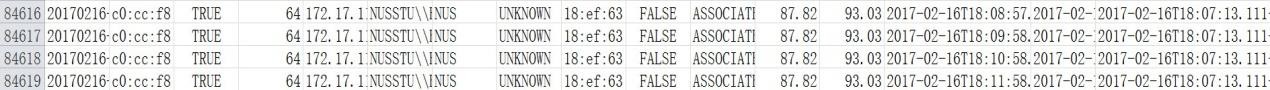
Another situation is if there is not any record in last minute, the device should not be a stationary device because a stationary device communicates with AP in a period time. Based on these assumptions we can justify stationary device by the following method:

Figure 4.1 Flow-chart to justify stationary device

We use 3 examples to show how to apply this method to get history data to clean current localization data.

Eg1. Stationary example



* + 1. Select one data
    2. Check whether the last 1 minute time snap record exist

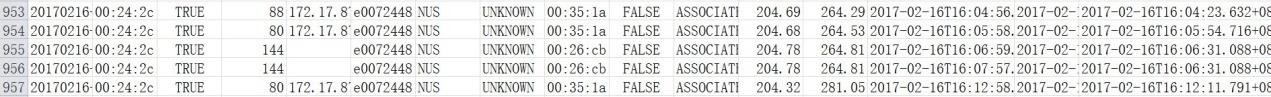
(When exist ,the last record’s ‘current time’ would equal to 18:10 ) That means the device is not a new located device;

1. If is ‘Yes’ ,check Apmacaddress ,whether the device connect to another AP in 1

minute time snap. If ‘Yes’, then the device might be moving; If ‘No’, the device

might be stationary; If the device is stationary in these minutes, we can bring data of these minutes into algorithm

Eg2.Stop situation(Ap change)



Moving Example: 1.Select one data

1. Check whether the last 1 minute time snap record exist(Yes) That means the device is not a new located device;

check Apmacaddress ,whether the device connect to another AP in 1 minute time snap. ( ‘Yes’, then the device might be moving)

Then last minute data cannot bring into algorithm Eg3.Stop situation(No record exist)



1. Select one data(blue rectangle)
2. Check whether the last 1 minute time snap record exist (NO)

Then the device lost, which means this last minute data cannot be bring into algorithm

### Average preprocessing theory

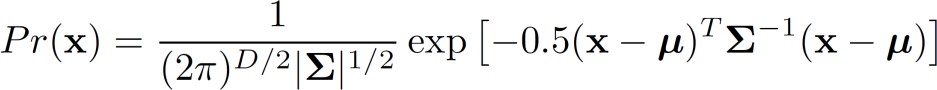
In chapter 3 we have discussed the error distribution of localization data is Gaussian distribution and get parameters of this distribution by experiment.

However, it is obvious that every device has it own localization error distribution

parameters because of different position and different device power. It is hard to

say that the parameter is suitable for all localization data. We use ML algorithm to estimate this unique parameters for every device.

We use ML estimation the parameters for multi-variable Gaussian distribution which is as following:



For short we can write as:



Where mean vector μ ∈ ℝ2 for longitude and latitude;

Symmetric positive-definite 2 × 2 covariance matrix Σ and its precision matrix

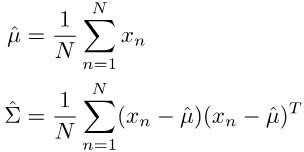
Λ = Σ−1;

ML estimation is:



To get the maximum function value we do log transform first:

f(𝑥1, 𝑥2) = log(p(𝑥1, 𝑥2)|μ, Σ)

We can bring the 2-D Gaussian distribution into function above and do derivation to get parameters which maximize the log function:

The demonstration is at Annex 1.

This result tells us that we can use average of all data points from a single device in continuous time to clean the localization data of this device.

### Implement of Average preprocessing

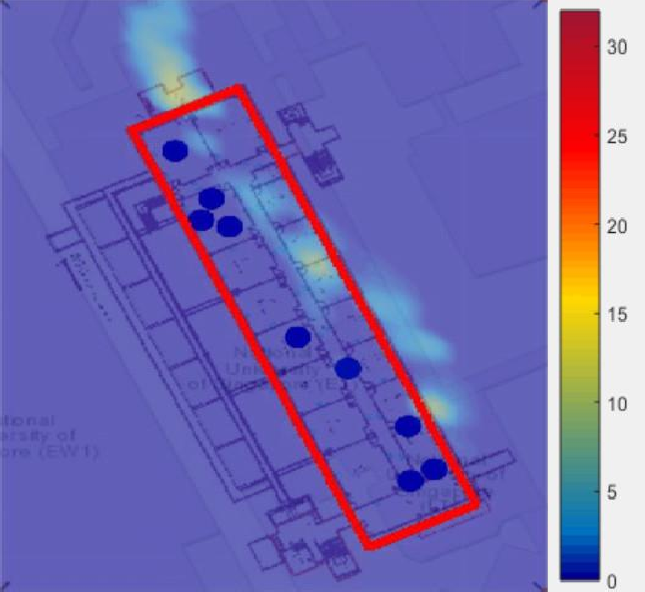
By chapter 4.1 we can get a group of history data for device to help us to clean up the current localization data;

By chapter 4.2 we have shown that we can use average of these data to move the current point to a more proper position.

In conclude we replace the current localization data with the average of history data before Monte Carlo method (which means preprocessing). Then we can draw a better performance heat map by these data.

### Application of Average preprocessing

We take E1-06 as an example:



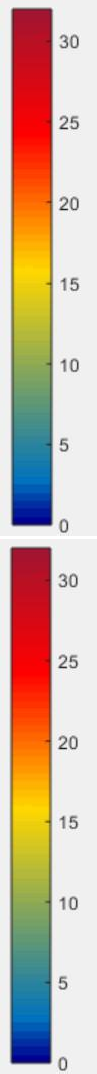
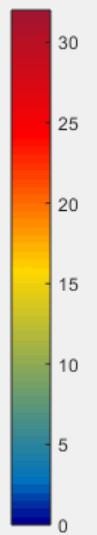
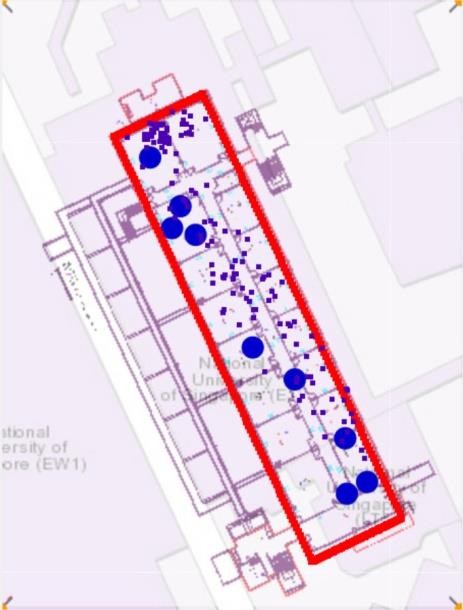


Figure 4.2 Average preprocessing effort. Top two are original points and heat map, next two are with Average preprocessing points and heat map, then are without Average preprocessing Monte

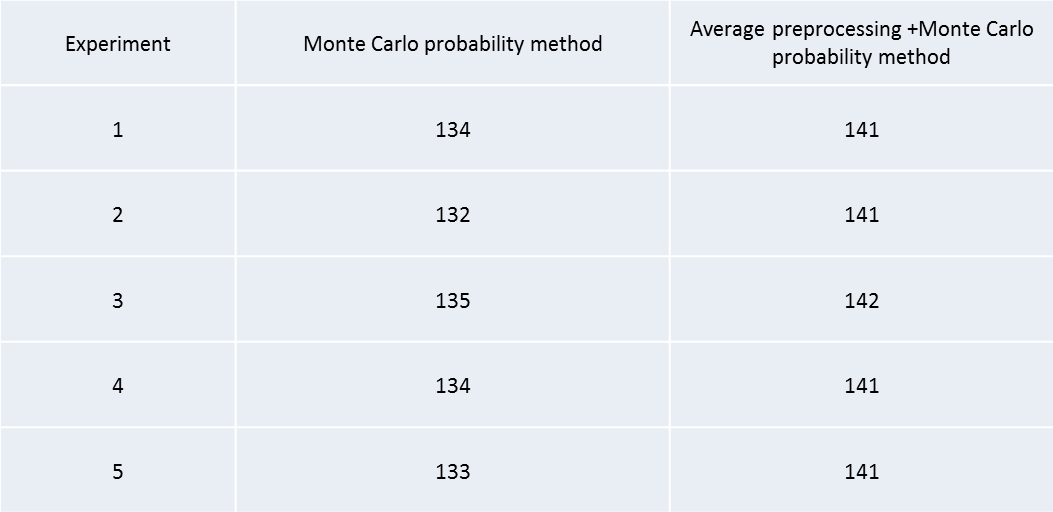
Carlo points and heat map and Bottom two are after Average preprocessing points and heat map We can compare the difference between with average preprocessing and without average preprocessing as following:

Table 4.1 contrast between with and without Average preprocessing

Total number of device in E1-06 at 18:11 is 157

We can find that although this problem still exists, it has been rectified to some extent and less device lost by Monte Carlo probability method with Average preprocessing.

Similarly we can do for E4-03 and E5-03 and get a better output

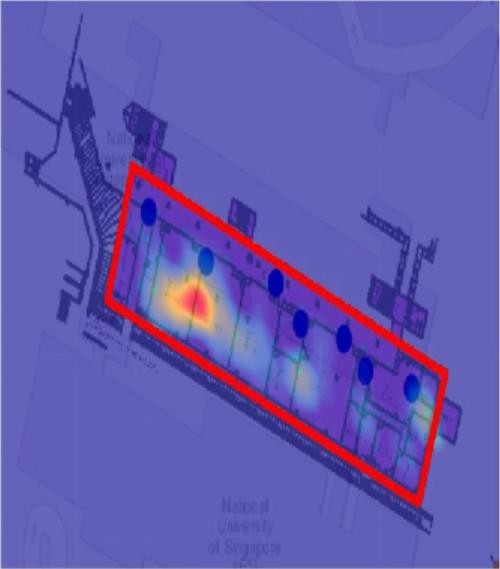


Figure 4.3 Final output of E4-03 in 14:30 Feb/15



Figure 4.4 Final output of E5-03 in 14:30 Feb/15

The Average preprocessing and Monte Carlo method have good performance in such places as well.

## CHAPTER 5 CONCLUSION

In this report it proposes Monte Carlo method and Average preprocessing algorithm which is a cost-effective, accurate, robust, scalable and calibration free localization algorithm. By employing Multivariable Gaussian distribution, Most Likelihood algorithm, Monte Carlo theory, the heat map’s performance will significantly increase.

It has addressed the various causes for localization error containing system error, casual error. To improve this problem it shows experiment in several places to test the error distribution. And by this error distribution we to apply 2-dimensional Gaussian distribution and Monte Carlo probability method to push localization data points into building while keep their position feature. In addition, it has also identified the problem of device WIFI power difference problem and addressed it through Average preprocessing by history data to estimate a more accuracy Gaussian distribution parameters.

It has also provided a detailed analysis of system error which clearly explains where the error may have occurred in our system. This detailed analysis serves as a guide for researchers to continue optimizes our system. Last but not least, it will get better performance with more accuracy Access Point position and localization position. And this more accuracy heat map can be used to estimate crowd density to do management for more convenient life.

# Reference

[1] Takuya Yoshida, Yoshiaki Taniguchi,” Estimating the number of people using existing WiFi access point in indoor environment”, Advances in Computer Science [2]Y. Wang, J. Yang, H. Liu, Y. Chen, M. Gruteser, and R. P. Martin, “Measuring human queues using WiFi signals,” in Proceedings of ACM MobiCom 2013, Sep. 2013, pp. 235–237.

[3]"United States Patent and Trademark Office, registration #75263259". 1993-09-01.

1. Wilkinson, Leland; Friendly, Michael (May 2009). "The History of the Cluster Heat Map". The American Statistician. 63 (2): 179–184. doi:10.1198/tas.2009.0033.
2. "Software Engineering Perspectives and Application in Intelligent Systems: Proceedings of the 5th Computer Science On-line Conference 2016 (CSOC2016), Volume 2, by Radek Silhavy et al".
3. Wikipedia: Heat Map
4. Perrot, A.; Bourqui, R.; Hanusse, N.; Lalanne, F.; Auber, D (2015). "Large interactive visualization of density functions on big data infrastructure". IEEE 5th Symposium on Large Data Analysis and Visualization (LDAV), 2015: 99–106. doi:10.1109/LDAV.2015.7348077.grf
5. Goering, Richard (4 October 2004). "Matlab edges closer to electronic design automation world". EE Times.
6. G. Mao, B. Fidan, B.D.O. Anderson, “Wireless sensor network localization techniques,” Computer Networks, vol. 51, pp. 2529–2553, 2007
7. Anderson, Herbert L. (1986). "Metropolis, Monte Carlo and the MANIAC". Los Alamos Science. 14: 96–108

## Annex

**Demonstrate of ML Algorithm**

