Adaptive Indoor Positioning System based on Locating Globally Deployed WiFi Signal Sources

Firas Alsehly

PhD – The University of Edinburgh – 2019

Lay Summary

Over the past decade, social and commercial applications show high interest in indoor positioning to boost their data attribute and enable location based services. Such applications aim to utilise location information on smartphones either in real time, such as navigation services, or streamed through backend processing modules, such as data analytics. However, with GPS being the most mature source of location data on smartphones, obtaining location attribution indoors remains a problem. Nevertheless, GPS has the advantage of being globally available and capable of providing location data with reasonable accuracy wherever line of sight connection to enough number of satellites is feasible. Indoor positioning systems should provide location context without the dependency on sky visibility. Therefore, this research evaluates various methods and algorithms to enable location awareness indoors for popular smartphone applications.

This thesis demonstrates an implementation of indoor positioning solution that utilises the existing infrastructure of *WiFi* signal sources around the world. Using an ordinary smartphone, without any alteration or customisation, to automatically record signal strength indicators for WiFi transmitters deployed around the area, should be enough to estimate the smartphone position. To be able to utilise the majority of globally deployed infrastructure, the proposed solution sets the limit to operate without even previous knowledge of WiFi infrastructure deployment. Hence, this research adopts only algorithms that is suitable for auto-discovery to empower the acquisition of meaningful data and support the construction of global database.

Furthermore, the thesis also proposes segmenting the earth into a grid to minimise local effects on estimating *WiFi* signal strength and to reduce the impact of walls and other obstacles that cannot be identified on global scale. With that in place, the acquired data streamed through set of algorithms to populate the grid with estimated signal strength values. Therefore, each area in the grid would accurately describe the layout of WiFi transmitters and map them into virtually connected graph.

All the above have empowered this research to autonomously locate signal sources, *WiFi* access points, within the designed global grid. Finally, this thesis evaluates the quality of proposed algorithms using large dataset collected in various countries. It also suggests error estimation model and mitigation for multi-dimensional floor ambiguity.

Table of Contents

List of Figures

List of Abbreviations:

Abbreviation	***Description***
3D	Three Dimensions
AoA	Angle of Arrival
DoA	Direction of Arrival
Geoindex	An Index for Geospatial Geographical Area
GNSS	Global Navigation Satellite System
GP	Gaussian Processes
GPS	Global Navigation System
HoR	Hand Over Ratio
KNN	K Nearest Neighbours
Labelled Observations	The data collected when performing "*Scanning*" while location data is available or can be obtained by position estimation
LBS	Location Based Services
LESS	Locating Electromagnetic Signal Sources
LoS	Line of Sight
LSVM	Least Square Vector Machines
MDS	Multi-Dimensional Scaling
NLoS	None Line of Sight
PDF	Probability Distribution Function
PDR	Pedestrian Dead Reckoning
RSSI	Received Signal Strength Indicator
RTT	Round Trip Time

Scanning	The process of collecting observations for signal strength from the surrounding signal sources associated with location data if available
TOF	Time Of Flight
UWB	Ultra-Wide Band
WAP	WiFi Access Point
War-Driving	The process of driving a car around an area, or region, and perform scanning simultaneously
WiFi	abbreviation for wireless fidelity
WLAN	Wide Local Area Network

List of Symbols:

SYMBOL	DESCRIPTION
CH	Cluster Head as the one record represent fused measurements in selected cluster
$D(i,j)$	The estimated distance between observation i and observation j
D_{ERR}	The difference between estimated distance and actual distance
d_{geo}	Geographical Distance Between two points
$DS\mu$	The dependency score from given measurement compared to the mean measurement in the cluster head
G	Geoindex area that contain the set of measurements
$L(T)$	Set of location data for an observation collected, or estimated, on time T
$M\{\}$	Set of signal sources *MAC* addresses grouped in one observation
m_i	One MAC address in set of signal sources $M\{\}$
$O\{\}$	An observation that enclose set of signal sources associated with location
P	One position data point in a set of location data L
$Q(T)$	The covariance measure for set of location data collected, or estimated, on time T
r_j	the RSSI measurement of the *MAC* address m_i
w_i	Weightage measure assigned to the location measure l_i based on the various scoring, covariance and signal strength
Δ_I	The score of confidence in *MAC* address mi compared to all signal sources in the set M
μ	The mean of set of measurements, here used for *RSSI*
$\lambda_i(O_i)$	The density function of observation i
σ	The standard deviation for set of measurements, here used for *RSSI*

Declaration:

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or processional qualification except as specified.

Acknowledgements

This research was partly funded by sensewhere Limited, previously known as stasis Ltd. The Author would like to credit sensewhere for the continuous support and the knowledge shared. The contribution of sensewhere in this research has enriched the contents with valuable real-world implementation examples as well as significantly contributed to the scalability of this research.

All crowd-sourcing data and data collected via sensewhere mobile software shall remain the property of sensewhere Ltd.

Abstract

Recent trends in data driven applications have encouraged expanding location awareness to indoors. Various attributes driven by location data indoors require large scale deployment that could expand beyond specific venue to a city, country or even global coverage. Social media, assets or personnel tracking, marketing or advertising are examples of applications that heavily utilise location attributes. Various solutions suggest triangulation between *WiFi* access points to obtain location attribution indoors imitating the *GPS* accurate estimation through satellites constellations. However, locating signal sources deep indoors introduces various challenges that cannot be addressed via the traditional war-driving or war-walking methods.

This research sets out to address the problem of locating *WiFi* signal sources deep indoors in unsupervised deployment, without previous training or calibration. To achieve this, we developed a grid approach to mitigate for none line of site (*NLoS)* conditions by clustering signal readings into multi-hypothesis Gaussians distributions. We have also employed hypothesis testing classification to estimate signal attenuation through unknown layouts to remove dependencies on indoor maps availability. Furthermore, we introduced novel methods for locating signal sources deep indoors and presented the concept of WiFi access point (*WAP)* temporal profiles as an adaptive radio-map with global coverage. Nevertheless, the primary contribution of this research appears in utilisation of data streaming, creation and maintenance of self-organising networks of WAPs through an adaptive deployment of mass-spring relaxation algorithm. In addition, complementary database utilisation components such as error estimation, position estimation and expanding to 3D have been discussed. To justify the outcome of this research, we present results for testing the proposed system on large scale dataset covering various indoor environments in different parts of the world.

Finally, we propose scalable indoor positioning system based on received signal strength (*RSSI*) measurements of WiFi access points to resolve the indoor positioning challenge. To enable the adoption of the proposed solution to global scale, we deployed a piece of software on multitude of smartphone devices to collect data occasionally without the context of venue, environment or custom hardware. To conclude, this thesis provides learning for novel adaptive crowd-sourcing system that automatically deals with tolerance of imprecise data when locating signal sources.

1 Introduction

1.1 Background

Over the past decade, many social and commercial applications demonstrated high interest in indoor positioning as an extra data attribute added to their location-based services (*LBS*). Such systems aimed to utilise location information on smart phones either on real time, such as navigation services, or streamed through backend processing modules, such as data analytics. To date, the most mature source of location data on smart phones is still limited to Global Navigation Satellite Systems (*GNSS*). Such systems have the advantages of being globally available and capable of providing location data with reasonable accuracy when there is a line of sight at least to three satellites. However, even with the wider coverage of *GNSS* systems (GPS, GLONASS, Beidou, Galileo), it fails to provide reliable location data indoors or in hyper local areas with limited sky visibility. While using *GNSS* to provide the location context seems practical outdoors, researchers have been trying different methods to enable similar positioning performance indoors. Breaking the indoor barrier and maintaining global availability similar to *GPS* soon became well recognised as the indoor positioning challenge.

Among various methods and technologies proposed to enable indoor positioning on smart phone, *WiFi* based solutions were the one widely adopted by native operating systems, such as Android and iOS. The reason behind such adoption were solely due to its potential to operate autonomously on a global scale. alternative technologies were limited by the amount of customization it would need before operating. Other widely present solutions require hardware customization, setup or periodical recalibrations. Examples of such solutions include ultra-wide band (*UWB)*, radio frequency (*RF-tags)* and Bluetooth Beacons [1], [2], [3]. On the other hand, magnetic field fingerprints, 3D imaging based systems [4], [5] or systems that utilise inertial phone sensors, such as pedestrian dead-reckoning (*PDR*) [6], are examples of active research topics that consider infrastructure free solutions.

Further utilisation of existing infrastructure continue to appear in advanced research projects in what is known as fingerprinting technique [1], [7], [8]. However, such systems require pre-calibration or pre-knowledge of infrastructure distribution and environmental parameters or building layout prior to providing reasonable indoor positioning. Furthermore, global spread, or venue-based modelling, remains the primary limitation for such systems [8]. Therefore, the concept of autonomous indoor positioning systems still suffers from various challenges that hold its realisation. To be recognised as autonomous system, an indoor positioning solution should be able to operate without pre-calibration and require minimal efforts for maintenance. Such systems have the potential to overcome the availability challenge and to become the solution for global indoor positioning.

Amongst all indoor positioning technologies mentioned above, *wireless* and *PDR* solutions are the strongest candidates to empower crowd-sourcing based solution. The high availability and low cost of these systems have strengthened its popularity. However, as inertial sensors only provide relative motions, it cannot operate completely on its own without continuous stream of anchor positions along the walking path. Furthermore, *PDR* shows significant sensitivity to smart phones gesture or relative orientation compared to the user body frame. Also, it often requires recalibration to eliminate any bias in phone hardware before every indoor walk. In addition, PDR to date still suffer from accumulated drifts on long walks [9]. Therefore, such solution has become primary source for additional location attributes, such as steps heading and travelled distance, fused with other technologies in a hybrid system rather than being an indoor positioning system on its own. Thus, *WLAN* solutions, based on crowd-sourcing of *WiFi* signal sources, remain the potentially substitute for *GNSS* indoors.

In this thesis we demonstrate our work on research and implementation of an adaptive indoor positioning solution based on crowd-sourcing *WiFi* signal sources location and signal attributes. By crowd-sourcing we refer to fully unsupervised solution that anonymously collects labelled and unlabelled data samples while it performs signal profiling and locates signal sources. The goal of this work is to use ordinary smart phones, without any modifications, to automatically acquire observations, such as received signal strength indicators (*RSSI*) from pre-existing *WiFi* transmitters and estimate its position indoor. As previous knowledge of transmitters' locations, its environmental characteristics or signals attributes are not available for globally deployed crowd-sourcing solution, we only targeted algorithms that support self-calibration with unsupervised learning approach.

Firstly, we proposed a method for modelling *RSSI* observations into sectors of dynamic global grid to eliminate the environmental effects on radio propagation. By utilising a grid approach, we reduced the impact of unknown obstacles such as walls, furniture and human traffic that can only be estimated if map data are available. Secondly, we employed data classification and clustering algorithm to populate the grid with estimated RSSI values for neighbouring WiFi transmitters gradually. As more smart phone users approach each grid location from various sectors our grid became more accurate in recognising popular clusters of data. In the end, our research primarily focusses on studying the feasibility of locating signal sources (*WAPs*) on reference to the proposed global grid. To enable this, we utilised a mass-spring relaxation algorithm to emulate the behaviour of self-organising networks into the designated indoor positioning problem. Finally, our research also examines the feasibility of implementing such framework to employ the proposed algorithms in global scale.

The above demonstrates the system ability to use crowd-sourced data to estimate smart phone location. The proposed system also provides brief discussion for *3D* modelling of *WAPs* database as an attempt to estimate smart phones location in multi-story buildings. To validate the proposed system usability, we perform sanity test experiments on database covering three different countries as a test for global distribution. To populate such database, we streamed unlabelled data collected by anonymous smartphone users as part of their daily routines over long period of time with random distribution in time and location. Such data collection is more realistic than any other procedure as it demonstrates the system ability to deal with modest quality data. For evaluation purposes, we utilised a separated set of labelled data, with ground truth position. Finally, we compared both estimate smart phone location data and the estimated signal sources data in the grid with ground truth data to validate the performance of our crowd-sourcing algorithms.

1.2 Motivation

The recent development in mobile devices, embedded systems and advanced sensors has extended the opportunity for *LBS* to the indoor territories. Today's smartphones are armed with powerful embedded "sensors", such as *GPS* receiver, *WiFi* receiver, accelerometer, gyroscope and digital compass. Such development has enabled new generation of context aware applications, such as safety personal trackers, fitness and health monitoring, geolocation advertising and geofencing. Such applications became widely available for smart phone users through cloud based mobile applications stores where they could download any application instantly. This revolution in application enabled smartphones enables limitless capability of such devices as developers and majority of phone users are keen to adopt to cutting edge technologies. In addition, the latest development on big data and data science inspired such applications to invest resources to acquire more data attributes.

Considering the important role location data play in most data driven application, the demand continues to inspire more research in this topic. Anyone takes a closer look into such data will soon recognise that *GPS* is still by far the most popular source of location for mobile personal devices. As humans tend to spend most of their time indoor, where localization through *GPS* in not feasible, the demands for indoor positioning flamed up. These demands have encouraged us to push the limits of all technologies trying to develop a simple, cost effective and accurate indoor positioning system that satisfy the requirements of the emerging applications.

Furthermore, the absence of reasonably accurate location context in more than 50% of the global occupied landscape forms a huge barrier for such life emerging technologies. This also suggests that solutions with limited coverage either require customization to hardware or calibration on venue basis would not be able to break through this barrier without significant efforts. Such efforts could involve forcing new infrastructure policies globally or utilisation of drones to continuously calibrate wider coverage solutions. On the other hand, WiFi transmitters are very popular in urban and suburban environments, especially in public services buildings such as malls, hospitals, libraries and museums. Previous research [10], [11] has also suggested that knowledge of WiFi signals radio map, or knowledge of transmitter's locations, would provide the granularity for seamless positioning experience indoors. Additionally, as smartphones move into always connected services, power consumption becomes more crucial. Therefore, hybrid localization systems have been adapted by majority of mobile operating systems to replace GPS location acquisition with WiFi even for outdoors when appropriate.

In addition to indoor positioning, sourcing radio map information through smartphone users would provide useful analytical data for network administrators, retail managers, and security officers in assessing service availability, quality and network dead zones. Therefore, autonomous crowd-sourcing of WiFi signals in public areas would provide also provide continuous feedback about its own wireless networks saving all efforts to assess service quality.

All the above has motivated our research to focus on crowd-sourcing WiFi radio maps and transmitters locations as a method for indoor positioning.

1.3 Problem statement

This Thesis is set out to examine the suggestion that global crowd-sourcing of pre-installed WiFi infrastructure can be used to enable indoor positioning everywhere without pre-calibration or additional hardware installation. The boundaries of crowd-sourcing are usually defined on case by case based on the availability of other data. For example, indoor maps availability is still limited to few providers and to date still lack global coverage. In another example, advanced trajectory detection and PDR data that power location filters suffer from limited availability as smart phone users are keen to save power and switch the location services on and off on demand. Therefore, some algorithms and techniques could be more feasible for *LBS* application that is offering navigation guidance to plurality of users only where indoor maps are available. However, the proposed research problem assumes wider coverage that is not limited to maps availability or specific navigation solution.

To set the boundaries of the proposed research problem, we defined the limitations of crowd-sourcing as per the following list:

- The surrounding environment is anonymous. This implies that no map data are utilised by radio maps learning algorithms. Therefore, map matching solutions or regression models that utilise map data are not within the scope of this research.

- No infrastructure modifications should be required in a form of hardware installation or customised network infrastructures. This means that designated positioning infrastructure such as Bluetooth beacons or the latest *RTT* protocols are not considered until they are available globally.

- All observations are submitted by off the shelf smart phones without any user or device identification. This will imply that observations are treated individually as discrete points rather than trajectories. Such limitation affects *PDR* or positioning filters that relay on modelling user's motion.
- Pre-calibration or semi-supervised learning that require prescribed collection of observations labelled with high confidence by humans or robots are not classified as part of our crowd-sourcing.

The thesis addresses these limitations by offering the solution tested with blindfolded data collection via software app running on off the shelf smart phones. The users of these smart phones have anonymously contributed to this program of research by providing unlabelled observation without any identification or human interaction with the software. Data contributors crowd-source anonymous territories without any supervision or prescribed routes. The defined crowd-sourcing method includes collection of signal propagation parameters, namely *RSSI*, passive collection of *GNSS* location references when available, unsupervised learning and adapting the radio map for the anonymous environment while simultaneously using the learned radio map to provide an estimation of smart phones location indoors.

1.4 Research contribution

The main contribution of this research over prior art is visible as it provides novel methods for locating *WiFi* signal sources deep indoors via crowd-sourcing providing global scale coverage with moderate accuracy, enough to provide location attributes data. The acquisition of such database complements the global coverage of *GNSS*, by further providing similar coverage inside venues or rooms. Compared to prior art, where hardware customisation or survey is required, this research propose solution that significantly reduces efforts required to locate smartphone user or device, in large complex territories to specific room without sacrifice on global coverage.

Further details about the research contributions can be summarised by the following:

- The thesis suggests novel solution to employ *NLoS* mitigation of signals propagation model with global deployment providing feasibility for indoor positioning on large scale.
- The thesis proposes utilisation of statistical Gaussian distributions during data clustering and classification to identify common signal anchor points indoors.

- The thesis introduces novel algorithm for creating and maintaining adaptive radio-maps or local and global grid nodes representing anonymous indoor territories.
- This is the first research that study indoor positioning solutions with global coverage providing comprehensive testing through significantly large dataset compared to any data used in prior art.

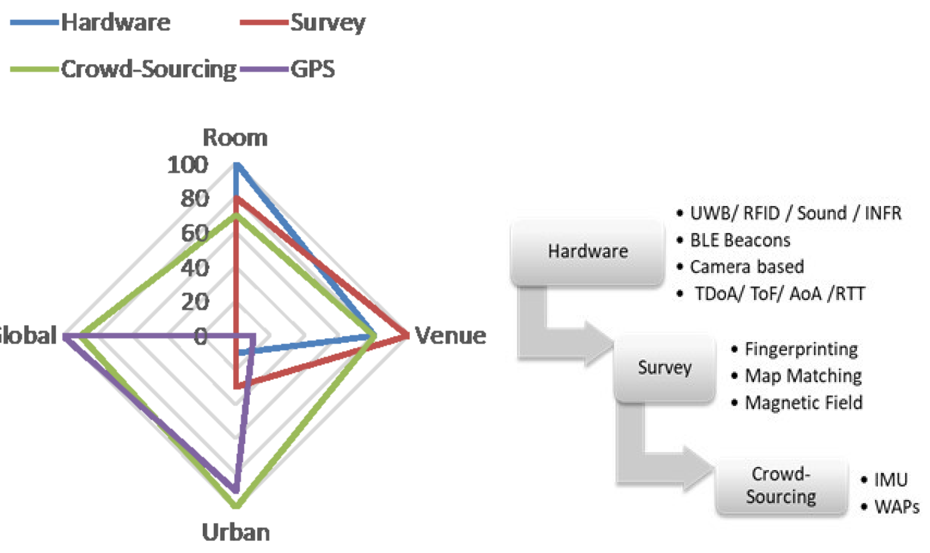


Figure 1.1: Comparison of various categories of indoor positioning solutions with *GPS* in terms of their effectiveness in room, venue, urban and global scale.

To illustrate the contribution of crowd-sourcing solutions compared to other categories in indoor positioning, Figure 1.1 provides metric to measure the effectiveness of each category. The figure suggests percentage measure of effectiveness in four different scales room, venue, urban and global. As clearly visible, while it is well known that *GPS* is not effective on room or venue scale, we argue that any hardware customisation would yield very effective in room setup and slightly less effective on venue scale. On the other hand, survey-based solutions are more effective on venue scale setup and perform good enough on room setup. However, both categories, would fail to scale to urban or global coverage due to adaptability limitation and cost of deployment and maintenance.

To conclude, Figure 1.1 demonstrates that using crowd-sourcing would enable indoor positioning solution to provide reasonably good coverage on the four scales. It also provides estimation of effectiveness in comparison to hardware and survey categories as well as GPS.

1.5 Thesis structure

This thesis addresses the indoor location problem by adopting crowd-sourcing approach to learn a *WiFi* signals radio map for anonymous territory and use it to estimate the position simultaneously. To achieve completely autonomous solution, we adopted data collection through an application deployed on smart phones to report received signal strength from surrounding *WAPs* periodically associating it with location data, driven by *GNSS* receivers on the phone, whenever it's available. In this document we refer to such process as *scanning* and the generated data as *observations*. Without confusing it with ground truth labelled observations, we often refer to observations that is associated with location data as *labelled observations*.

Figure 1.2 shows the basis of generic indoor position system utilising reference database of *WiFi* radio maps, or signal sources. In summary, mobile devices scan surrounding *WiFi* signals and consult a central database by sending "*where am I*" position enquiry. In return the central database and the associated processing units, usually known as positioning servers, would process the data to calculate the mobile device position. In our approach we programmed the software hosted on data contributor's smart phones to always consult the central database to obtain position, even if *GPS* location were available. This has enabled a solution that continuously updates the references database as it utilises crowd-sourcing as backend system setting behind positioning server.

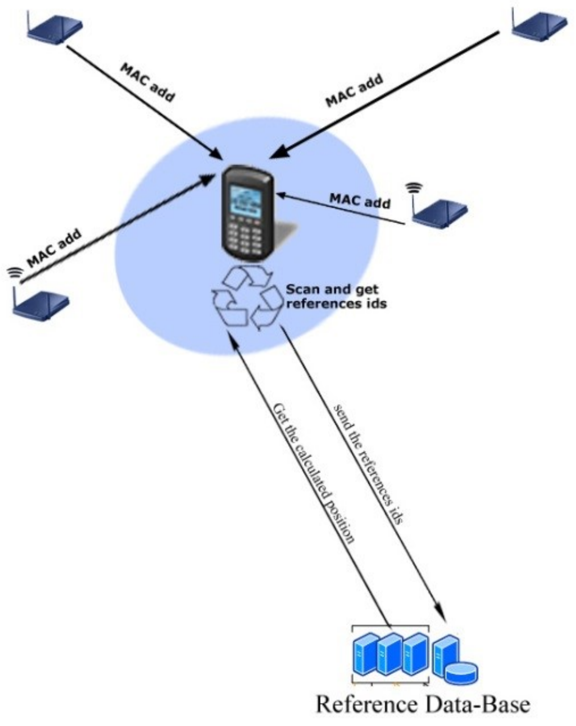


Figure 1.2: An example of *WiFi* positioning system showing smartphone recording observations of WiFi signal strength and making position request to central system with reference database to obtain its estimated position.

In this chapter (Introduction), we set the scope for this research and introduced the crowd-sourcing concept, motivation, problem statement and research contributions. It scans the literature to exploit the concept of radio map creation and WiFi signals propagation from prior art perspectives. In addition, it provides literature to justify excluding fingerprinting-based solutions and introducing previous work related to crowd-sourcing of *WiFi* signals sources or adaptive radio-maps.

In the second chapter (Methodology and Data Analytics) we describe the dataset used throughout this research and present some analytical findings. It also presents the selected test beds, quality metrics and sanity testing and analysis methods adopted in our research. Finally, chapter two also study the implementation and platform architecture used to enable global coverage.

In chapter three (Locating Signal Sources) we present our approach and algorithms in detail. It also introduces the proposed data classification and data clustering based on radio-map similarity metrics to assign reference clusters on local and global grid. In addition, the third chapter presents the first set of test results in form of sanity tests performed on the data sets introduced in chapter two.

Finally, chapter four (Utilisation of *WAPs* Database) introduces an early effort we made on grid construction, alternative to the process described in chapter two. It also presents our testing algorithms used to perform smartphone position estimations. In addition, we attempted to model error estimation, inaccuracies in radio map construction and how it could impact position estimation. Finally, the same chapter describes the concept of floor determination in multi-storey buildings as a potential expansion of *2D* grid construction. It solely focusses on our attempts to identify which floor the user is on based on probability model given the knowledge of *WAPs* distributions among multiple floors.

1.6 Literature

Since the early days of *GPS* utilisation for public, *LBS* service shown high motivation and potential to contribute toward improving quality of life. Such development has encouraged researchers to bring this experience and quality indoors by pushing the limits on various technologies. As *GPS* still to date shows major limitations on its indoors capability, the search for alternative sustainable and calibration free solutions has increasingly appeared in literature. The first classification of proposed solutions distinctly put them in two categories, hardware-based solutions and software-based solutions. Hardware solutions usually accommodate requirements demanding limited implementation scale in order to be affordable. Such systems are not within the scope of this research as we focus on autonomous systems that can work unsupervised globally. However, as smart phones manufacturers and operating systems start to adopt some hardware changes, such as round trip time (*RTT)*, new technologies may become feasible in near future.

On the other hand, software only solutions, were employed by many research projects which can be categorized as ranging techniques, such as trilateration, and none ranging techniques, such as fingerprinting. While ranging solutions try to work out a mobile node location by measuring the distance from signal sources with known location, none ranging solutions only concern finding the best match for radio fingerprint of the received signals from a database of training samples. Nevertheless, both techniques require some knowledge about WiFi signal sources or signal profiles before it can perform any estimation. Hence, crowd-sourcing options has been identified as the only approach that could provide wider availability and global coverage.

1.6.1 Using WiFi for indoor positioning

Few years ago, major smart phone frameworks, such as Apple iOS or Google android, utilised their own global database of *WiFi* signal attributes to provide an estimation of a mobile phone location even inside buildings. A study made by Zandbergen [12] compared such positioning performance on android and iOS on large scale deployment. Zandbergen discovered that accuracy would vary between different areas and different times demonstrating the early adaptation of crowd-sourcing. On the other hand, the rising interest in *WAPs* data supports the utilisation of *WiFi* based indoor positioning as the most feasible solution so far. As people spend most of their time indoors, WiFi based positioning became a valuable complementary to *GPS* which is used to navigate outdoors.

Therefore, many hybrid systems aim for seamless positioning everywhere by combining multiple position technologies, such as *GPS* outdoor and *WiFi* indoors [13]–[15]. However, positioning quality requirements vary based on each use case. The expected accuracy of crowd-sourcing solutions would be affected by many elements such as algorithmic calculations, data availability, *WAPs* quantities and signal characteristics. In this section, we try to cover a wide range of technologies that utilised WAPs in reasonably large-scale deployment. Nevertheless, the most recent state of the art that could match the global coverage criteria doesn't appear in publicly available literature.

1.6.2 Hardware customization for time base solutions

As the defined target for solving the indoor positioning problem somehow overlap with *GNSS*, many researches tried to emulate *GPS* accurate time-based calculations by modifying transmitters and mobile nodes hardware. Particularly, this approach requires the radio signal to be timed very accurately on both ends, transmitter and receiver [16]. Therefore, *WiFi* protocol specifications and hardware modification were required to realise such solutions.

Such option promised an alternative to *RSSI* measurements by offering much accurate distance estimation by exchanging customised time packets. The most common adopted techniques are: Time of flight (*TOF*) and Round-Trip-Time (*RTT*). Both techniques record timestamps at mobile node (laptop or smartphone) and an access point (router or modem) during handshaking. The timestamps recorded at either end can then be utilised to calculate the differential time consumed by signal to travel through the air. This can be used to determine the position of a mobile device between known *WAP* locations using trilateration. It is most suitable for *WiFi* since an acknowledgement signal is already available when a request is made. Unlike the models using Received Signal Strength (*RSSI*), errors in ranging distances estimated using *RTT* are constant [17], therefore the measurements behave better indoors.

On the other hand, *RSSI* ranging errors in the estimated distance are subject to shadowing, multipath and interference. However, *RTT* measurements also suffer from errors and require additional or newer hardware to operate. Unlike *RSSI* measurements, error models for *RTT* measurements cannot be assumed Gaussian because errors will always increase with quantity of measurements. This can be handled by using other error models (Rician, Rayleigh, etc.) [18].

To overcome the hardware requirements, single side *RTT* technique is developed to measure the delay between "*request to send"* command executed by a node on wireless network and *"clear to send"* message received back from the controller, *WAP* in this case. This time delay is usually the sum of *RTT* and processing time on transmitter side. Although *RTT* should be almost constant, processing time is subject to significant variations. To reduce the error produced by processing time variations, statistical models can be adopted [19]. However, such models require significant amount of measurements before computing ranging distance.

With the introduction of timing in newer versions of the 802.11mc standards [20], the use of *RTT* for positioning has become feasible as manufacturers started to support the new standards this year. According to this new standards network advertising, packets exchanged between nodes would be able to measure two-way *RTT* with high precision even from very few measurements. Furthermore, in crowd-sourced scenarios, it increases precisions of the estimated positions of mobile devices hence it improves WAPs profiles database.

Another example of custom hardware appears in the utilisation of special antennas to obtain signals direction of arrival (*DoA)* as an additional parameter when collecting observations from modified mobile units [21]. With such unites possibly being smartphones or other form of handheld device, *DoA* availability provides extra data to aid position estimation and improve accuracy. Authors of [21] have utilised triangulation approach to locate signal sources very precisely, in the range of sub meter. However, as currently deployed antennas on the large population of smart handheld devices does not support angle of arrival (*AoA)* or direction of arrival *(DoA)* detection, the technology would be rendered impossible to implement for crowd-sourcing solution.

To conclude, in near future indoor positioning solutions would combine signal strength and time to substitute for noises and to enforce or discard measurements. This will eventually yield significant improvements to ranging techniques once the adoption of new hardware is common enough to replace three billion *WAPs* currently deployed worldwide. Until then, *RSSI* based distance estimation is the only feasible solution to enable trilateration positioning based on current majority of smart phones and WAPs infrastructure.

1.6.3 Trilateration vs fingerprinting

The trilateration approach is based on fundamentals of least square solving to find an intersection point that satisfies all distances between mobile node and multiple signal transmitters [16], [22]. Similar to *GPS*, *WiFi* trilateration requires at least three transmitters with known locations and a reasonably accurate estimation of their distance to the mobile node in order to get a fix. Unlike *GPS*, *WiFi* protocols have not been designed for location purposes. Therefore, converting signal strength to distance by modelling the propagation of *WiFi* signals indoor remains challenging. Equation (1.1) shows an example of such propagation model, being the most popular for trilateration [23].

$$P_{re} = P_{tr} + 10\, n.log\left[\frac{d}{d_0}\right] + \propto \qquad (1.1)$$

P_{re} : is the received *RSSI* measurement

P_{tr}: is the transmitter power at reference distance of d_0.

n: is a pathloss exponent parameter reflecting the environment and obstacles.

$\propto$: is a parameter representing noise, estimated as Gaussian white noise.

Improving position estimation accuracy in trilateration approach has been mainly limited to improve ranging techniques by adopting various signal propagation models tailored for indoors. Nevertheless, authors of [24] proposed to use trilateration in room range to improve fingerprinting-based solution. This model reduces errors in signal propagation modelling while it uses very few fingerprints as an initial fix to a room. Similarly, other researchers considered hybrid solutions to overcome the propagation errors by adapting linear models on short distances [25]. These solutions have contributed toward fully autonomous indoor positioning system aiming for unsupervised or semi-supervised solutions [26], [27].

On the other hand, fingerprinting techniques are based on fully supervised learning based on pretrained models. Usually the system should have two phases, offline and online. The offline, or training phase, various classification and clustering techniques could be used to build a fingerprint database, known as radio map. The process of generating such radio map includes measuring the received signal strength (*RSSI*) of all the *WiFi* access points that can be detected when the device is in known sampling reference point and storing labelled data of *WiFi* signals characteristics in database. Pre-processing can then generate the radio-map or the area covered by this labelled data. Once radio map is populated, mobile devices submit *RSSI* of all the *WiFi* access points (*WAPs*) that can be heard from any unknown location, within the sampling coverage area, so the system can estimate their location. To perform an accurate positioning of mobile devices, set of *RSSI* measurements are compared with the radio maps in the pre-trained database using multi-dimensional similarity measures to find best match [28].

RADAR [29], Horus [30] and Placelab [31] were among the very first solutions to set the grounds for WiFi positioning. Generally, for all these systems, increasing the number of samples during training phase would improve its accuracy as the probability for finding better similarity during online phase increases. Later, more advance efforts have then been made to improve accuracy and reduce location errors. For example, Bayesian Filters were utilised in fingerprinting solutions aiming to reduce sampling efforts without compromising accuracy [32]. Furthermore, neural-network-based methods have also been employed to further reduce calibration overheads [27].

To summarise, all the above have suggested that neither fingerprinting nor the conventional trilateration would provide sustainable solution for indoor positioning problem. Firstly, sampling each public indoor building all over the globe is a severe and worthless effort as infrastructure and layout constantly change. Therefore, until future research proof that an effective and sustainable fingerprinting solution can work autonomously, such solutions will only be adopted on site by site bases. Secondly, obtaining very accurate trilateration without adaptive signal propagation modelling is not yet feasible. Hence, zero calibration systems that satisfy the unsupervised learning of *WiFi* infrastructure and combine both fingerprinting and trilateration is most promising candidate for solving the prescribed problem.

1.6.4 Zero calibration solutions:

As the last few years witnessed many location aware services moving indoors, lead organizations, such as Apple, Google, Microsoft, Here and Mozilla, adapted one form or another of *WiFi* based localization. This trend increased the demands for location attributes data to be available everywhere even if that would compromise accuracy. Therefore, autonomous *WiFi* positioning systems, also known as zero calibration solutions, appeared in research literature to answer for industry demands. In general, an autonomous system can be any positioning system, regardless of the technology behind it, which satisfies the following:

- Its ability to utilize existing hardware without modifications. For example, smart phones and *WiFi* transmitters.
- Its ability to expand beyond initial boundaries automatically by performing fully unsupervised learning of territories as layout or infrastructure change.
- Its ability to provide location attributes, room level accuracy continues to be the most demanding requirement made by commercial *LBS*.

One example of an autonomous system, presented by Cheng et al [33], uses an adaptive approach. In his work Cheng modifies the usual KNN algorithm to use clusters instead of samples. Then he employed self-healing algorithms to update fingerprinting clusters using every location estimation as new reference point to retrain radio maps. His results were also able to satisfy the last condition for accuracy requirement but still require initial site survey, hence expanding beyond initial boundaries were not presented.

Another comprehensive research on utilisation of Multi-Dimensional Scaling (MDS) to locate WiFi access points in unsupervised system is presented by Koo et al. [34], [25], [26]. The proposed solution promises to locate WAPs to the accuracy of 20m. With minimal contribution of GPS, Koo method align the constructed map of signal transmitters with any global location. It then attempts to estimate dissimilarity between all pairs of WAPs by processing time observations O(t) :{RSS_0, RSS_1,…. RSS_n} from all input data. The described method then generates a graph where WAPs are nodes and dissimilarity represent edges. MDS is then employed to find the best fit for all detected WAPs that satisfy all dissimilarities.

More recent work attempts to simultaneously recalculating smart phone positions while locating WiFi Access Points in post processing approach [35]. Their approach does heavily rely on static signal propagation model to estimate distance between Access Point and each observation points submitted by smart phone. The optimisation is made through solving set of least square equations, as per (1.2), defining the relationship between WAPs location and observation points location.

$$\check{F}: F_{pow} + \lambda_1 F_{gps} + \lambda_2 F_{acc} + \lambda_3 F_{\Delta} \qquad (1.2)$$

$\lambda_1, \lambda_2, \lambda_3$: are scaling factors.

F_{pow}, F_{gps} , F_{acc}, F_{Δ}: are functions of *RSS, GPS* positions, Position acceleration and elevation change in sequence.

Another example of optimisation algorithm, in form of error minimisation, we identified mass spring relaxation [36] as alternative approach to *MDS* and least squares. Although such algorithm is not frequently visible under indoor positioning discussions, we propose to model each *WAP* as the centre of gravity between set of springs connecting it to set of observations. In other words, mass-spring relaxation model all *WAPs* and observations into single graph $G{:}[P_0 , L , D]$, where P_0 is initial position estimation of given *WAP*, *L* is set of labelled observations represented by set of location measurements $\{l_1, l_2, \ldots l_k\}$ and *D* represent graph edges d_{ij} as optimum distance between *WAP* position P_i and the corresponding observation position l_j. To locate an optimal solution for *WAP* position P_i, mass-spring relaxation continue to shift the central node P_0 of graph *G* until it satisfies all edges in *D*. However, such optimal position usually is difficult to find. Hence, minimisation of differential errors in distance aggregated as stress measure on all graph edges are used to indicate to local or global minima for each *WAP*.

One major limitation of mass-spring relaxation appears in its inability to locate global minima. Authors of [36] suggest that including some anchors with wrong distance estimation to the central node, *WAP* in this case, would defer the ability of mass-spring to converge. This level of sensitivity suggests that mass-spring only converge to local minima. However, it is possible that the deployment of mass-spring as suggested by [36] would not converge to global minima in most cases. Nevertheless, on our deployment of mass-spring, discussed in chapter three, we expand the graph to include extra springs to anchors even if there are no direct observation to the *WAP* we are locating. Such modification has enabled us to locate global minima by minimising the error in all springs across the graph, rather than just the springs surrounding the *WAP* we are locating.

1.7 Comparison of crowd-sourcing compatible algorithms

As we are set to test the applicability of crowd-sourcing approach for mass-market deployment, we focus on the scalability of each algorithm before we choose one to accommodate our research target. Compared to *MDS* and iterative least squares, mass-spring provides the ground for scalable solution as it offers the flexibility of graph modelling. One can argue that *MDS* also offers graph deployment. However, *MDS* enforces the same weight on all edges of the graph. Hence, it is essential for *MDS* to identify finite boundaries when rendering the graph before starting the optimisation [26]. This is a huge limitation on the absence of maps. On the other hand, mass-spring offers the ability to optimise, scale and provide custom weight for each link regardless of how complex is the graph.

Furthermore, *MDS* require retest of all nodes in the graph on every iteration, while mass-spring can propagate the stress on weaker anchors limiting the integration required to converge. Hence, when applying *MDS* in metropolitan scale, errors in graph can grow exponentially rendering resolution impractical. Furthermore, it is a requirement of MDS to be able to measure the distance between each pair of WAPs accurately rather than elastic or fuzzy range. Koo proposed to do this by ensuring that his dataset covers all possible combination of RSSI measures [34]. Such assumption cannot be guaranteed in crowd-sourcing as it is very common that WAPs are installed off the public routes and region of RSSI measures would not be observed injecting wrong links into MDS.

Similarly, testing an iterative approach of least squares [35] is clearly restricted by number of equations that would need to be solved in large scale deployment as it is derived from all possible combinations of observations and *WAPs*. Therefore, local minima can only be sought after solving for all variables, coordinates and scaling factors, based on initial positions and static pathloss model. Therefore, due to the amount of observations in our global scale, our attempt to get such solution to converge was only successful after limiting the data to local areas rather than full dataset. This limitation renders iterative least squares impractical to be implemented for autonomous crowd-sourcing on global scale.

1.8 Modelling signal propagation for NLoS indoors

As commonly known, none line of sight (*NLoS*) conditions are major source of noises for locating signal sources or mobile devices through ranging techniques. In this section we survey existing methods proposed for mobile navigation application to deal with such conditions. The most common approach exploits the availability of time series measurements when estimating mobile object location indoors. Therefore, various filters used to smooth range measurements to mitigate the *NLoS* errors. Examples of such methods utilises Kalman filtering, particle filtering or combination of both [37], [38]. However, such filters heavily depend on time series measurements of single mobile unit Therefore, it is not applicable for this research, where we process data randomly received from crowd of mobile units.

Another research [39] employs various statistical parameters, or features, to classify and mitigate *NLoS* errors. The authors examined three different models *least squares vector machine (LSVM)*, *Gaussian processes (GP)* and *hypothesis testing*. The first two, *LSVM* and *GP*, are based on modelling the expected behaviour of *NLoS RSSI* set in time space using pre-training data sets. As per our previous argument time-based models are not examined in this research due to applicability. However, Xiao's third method, Hypothesis Testing, falls very well within our use case as it allows us to compare statistical features extracted from mass geo-located *RSSI* measurements to pre-determine statistical distributions for *LoS* and *NLoS*. Such implementation assumes prior knowledge of *NLoS* probability distribution, Gaussian distribution as an example. However, as the solution account for variation of mean and standard deviation it is still valid in most cases as it reserves a model for each WAP within the observation area.

1.9 Conclusion

To summarise, in this chapter we have introduced the hypothesis of autonomous crowd-sourcing to create global database of signal sources attributions. Also, we have demonstrated how this approach contributes toward solving the indoor positioning problem. In addition, we have also defined the scope of this research, focusing on our problem statement and motivations. We then provided more insights on prior art to justify the significance of adaptive radio-map from literature.

Furthermore, we have presented our justification on utilisation of crowd-sourcing versa customisation of hardware or venue survey presenting the concept of zero calibration as definition of crowd-sourcing. At the end, we have located three compatible algorithms *MDS*, *iterative least squares* and mass-spring. With initial justification on the scalability and adaptability to errors in observations, we concluded on using mass-spring for our research. In addition, we identified another limitation in crowd-sourcing approach related to *NLoS* conditions. Considering the scale of our proposed deployment, we scanned the literature and identified *NLoS* mitigation approach [39] that can satisfy our problem statement

2 Methodology and Data Analytics

In this chapter, we would describe how the proposed system is deployed and tested globally. Most of test data used in this thesis to verify the success of global deployment is the courtesy of sensewhere limited "*www.sensewhere.com*" as they have licenced and deployed the system since *2011*. Therefore, the very first part of this chapter will focus on the feasibility of global deployment as per the problem statement in chapter one. In addition, we will present the data set and methods of evaluation used to assess the quality of the obtained *WAP* database.

2.1 Global deployment of proposed algorithms

To analyse the crowd-sourcing success, we took a sample of 25,217,492 observations randomly distributed across the countries within the coverage area. We then used this comprehensive data set to show case the algorithm capability over a global grid. We are not aware of any data set on similar scale used in research literature. Most of the algorithms we came across during literature review were running local and building specific experiments that can easily overfit the algorithm to that specific building or layout. Using global data set is a key part of such methodology.

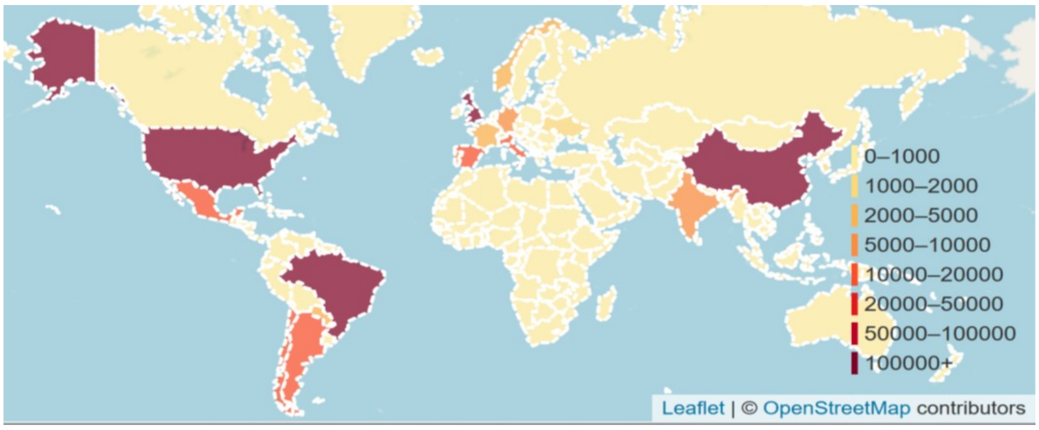


Figure 2.1: The distribution of our test dataset across multiple countries showing global coverage heat map based on observations count.

Figure 2.1 shows how the data set is distributed globally. It clearly demonstrates that the most of our data is based in four counties: China, UK, USA and Brazil. This can be explained by either deployment sensewhere made with customers, such as China and Brazil, or mass data collection exercises conducted in some major cities, such as UK and USA. Data collection process involved each smart phone performing both active and passive scans based on configurations supplied by hosting application and operating systems restrictions. As phone deployment is made by third party, we were unable to accurately measure or obtain details of such configurations or retrieve this from the data. Hence, phone deployment is not going to be covered within the scope of our research. Nevertheless, it is still possible to shed some light on what data attributes we had access to through sensewhere data sources.

Before we start any further analysis, let's have a closer look into the data attributes of raw observations as they are stored in the database before processing. As can be seen from the data attributes, we only deal with labelled data, However, data entries tagged with 'W' as source of location data, is considered unlabelled data due to uncertainty in location accuracy provided. However, in this research we attempt to mitigate for noises generated through such uncertain measurements as expected during large scale crowd-sourcing.

Table 2.1: Data attributes for raw observations received from mobile devices during data collection.

Attribute	Description	Sample
Timestamp	The time when this observation collected in *GMT*	*'2016-12-08 11:51:14'*
Geoindex	The geographical ID of global grid	*129960296450*
Latitude	Angular northing coordinates as per *WGS84*	*116.45020997095*
Longitude	Angular easting coordinates as per *WGS84*	*32.066326530612*
Altitude	An estimated absolute height from sea level.	*102*
Level	numerical indication of floor number relative to ground floor.	*0*
EstimatedError	Specified in meters as an expected error in location.	*12.82*
LocationTag	A label to describe the technology used to obtain this location. *G*: stand for *GNSS* and *W*: stand for *WiFi*	*'G'*
Detected Signals	An array of all MAC addresses heard from the reported location, along with its signal strengths.	*[{'0006C63165AA',-80},{'000CE6020CCC',-67}]*

As mentioned above, this dataset has been collected by software development kit offered to smartphone application developers to embed within their code revoking the native location libraries on Android operating system. By embedding such kit, end-users of such smartphone software application would record signal observation to obtain estimation of location data. Simultaneously all signal observation will anonymously be recorded and stored by central in isolation of any user data.

As a first analytical look into the collected data, we measured the percentage of unlabelled, or uncertain, data within the sample data set. To further show the low dependency on labelled observation, we plotted the proportion of labelled verses unlabelled data identified by *WiFi* and *GNSS* tag entries across our dataset. Figure 2.2 demonstrate an average percentage of labelled data as low as *28.2%* over the course of *17* month. It only considers China's entries as it is the largest and the most randomly distributed data across the country.

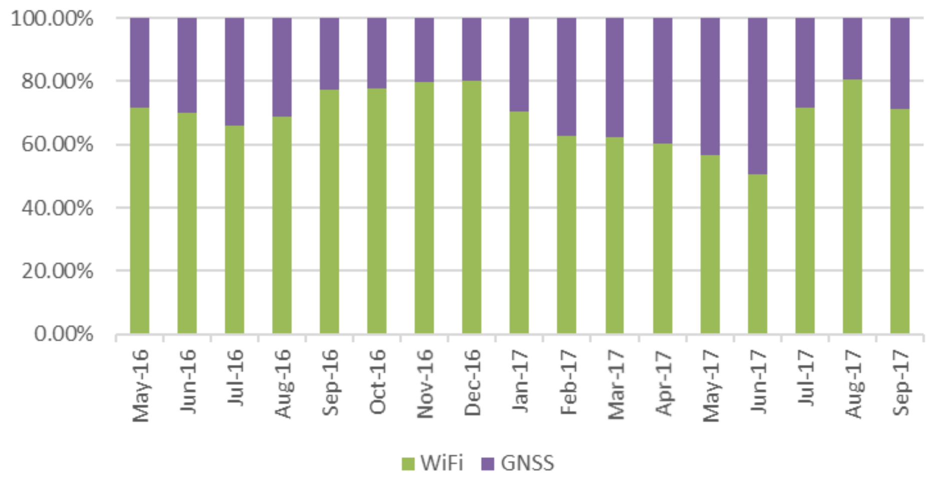


Figure 2.2: Percentage of labelled data *'GNSS'* and unlabelled data *'WiFi'* randomly collected by software development kit deployed in selected test areas in China.

Further look at the data shows that it covers time span of over two years. This guarantee that our data capture the changes over time as some infrastructure relocate to other parts of the city or even another country. Also new or retiring infrastructure would be present. However, we are more interested in proofing the feasibility rather than studying facts about *WiFi* routers life cycle. Hence, we are only going to present statistical measures to describe the quality of the data set used. Starting by number of contributing devices, we render each unique session as contributing device, as we do not hold user identification data. It is true that two different application sessions could be from one device, but for us they equivalent to two devices as the user started brand new request for location service. This of course doesn’t favour the development of algorithms, but it matches the real-world deployment for most cases. For example, Figure 2.3 shows the number of active users per month over the total 30 months of data collection.

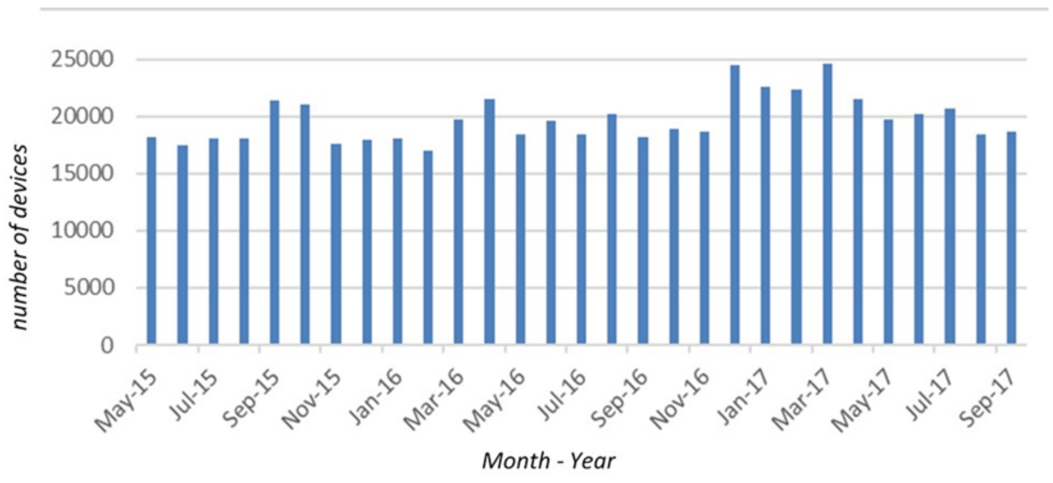


Figure 2.3: Monthly distribution of contributing users based on number of unique devices submitting data per month.

Another important factor related to the accuracy of the obtained database for *WiFi* transmitters is the classification of stationary or mobile signal sources. This is especially important as the research expands beyond a building or city to cover locations across the globe. Furthermore, considering the latest trend of *WiFi* connections available in public transport and personal hot spots, the impact of these mobile *WAPs* could accumulate to become major source of errors when using the database for any *LBS*.

Therefore, we performed some data cleansing before we run the data through the algorithm to remove any *WAPs* that obviously classified as mobile. For this initial classification, a simple voting system is deployed based on optimum WiFi propagation distance. As we are mainly interested in WiFi signal sources that offer wide coverage extending from indoor to the boundary of outdoors, where our labelled data are collected, we set to identify an optimum propagation distance for our cleansing. However, we are also concerned that too wide coverage of WiFi coverage by single WAP would offer more noise into the algorithm than helping it out.

Hence, we set the value to 300 meter as maximum allowed coverage area of *WAPs*. As we scan the data sequentially, based on time, we compare if the same *WAP* appear in observations more than 300m apart. In such case the *WAP* would score one relocation incident. Once a given *WAP* scored 3 or more relocations were classified as mobile. Table 2.2 shows the results of this initial classification for set of *WAPs* appeared in the sample dataset.

Table 2.2: numerical analysis of sample dataset showing *WAPs* grouped by their mobility tag.

Country	*Total*	*Stationed*	*Relocated*	*Mobile*
China	323576	285526	19514	18536
UK	47875	43011	2741	2123
USA	106152	94596	5428	6128
Total	477603	423133	27683	26787

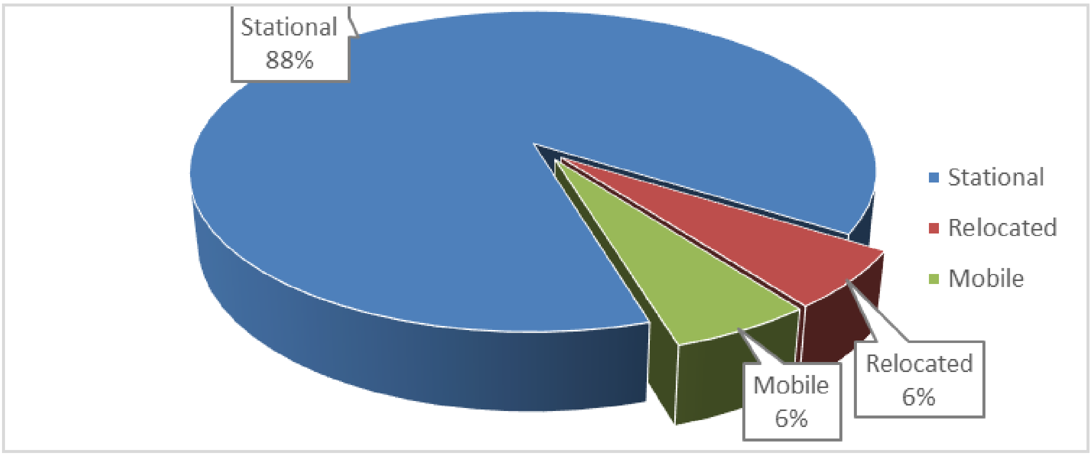


Figure 2.4: Percentage analysis of *WAPs* mobility flag as they are extracted from sample data set. The figure shows combined numbers for three countries based their score for violating maximum WiFi propagation distance. Stationed *WAPs* scored 0, Relocated WAPs scored<3, Mobile *WAPs* scored ≥3.

2.2 Measuring accuracy of WAPs database

The next part of our methodology is assessing the quality of database. As per the introduction, the proposed solution employs adaptive crowd-sourcing and self-learning algorithms to maintain, or automatically create, database of *WAPs* profiles. Justifying that our approach and the proposed algorithms produce an accurate estimation of signal sources locations or their signal profiles is not a straightforward process in the absence of ground truth data. Basically, full database justification is not possible without knowing *WAPs* actual location globally as we do not control areas which data collection app users visit. Therefore, we adopted sanity tests evaluation based on selected samples of the database.

However, to make sure that we do not end up overfitting the data to specific venue, we continue to use the full data set when running data through the system. This would guarantee that any impact of neighbouring data or stretch of *WIFI* signal propagation is considered when producing results. Nevertheless, we found that even on selected data set, it would be easier to evaluate database quality by locating smart phones rather than *WAPs*. Hence, we are going to perform two levels of evaluation: known *WAPs* test and known smart phones test.

Before we start describing each test in detail, it is important to mention that all data collection performed for these tests were performed using more than one phone. As we do not have record of what phones are used by end users while crowd-sourcing, we can’t compare or match the performance per phone model or hardware specification. Still the tests are valid indication of data quality as our ground truth data include measurements from different brands and models. To name some, we have used:

- Samsung S3 - S5.
- LG Nexus4.
- LG Nexus5.
- Motorola Nexus6.
- Huawei Nexus6P

2.2.1 Known *WAPs* tests

To overcome the challenge of *WAPs* with unknown location, we adopted methodology of associating each mac address to an estimated ground truth coverage area instead of specific location. For obvious reasons, asking retailers and public buildings owners about their infrastructure installation plan was rolled out as an option immediately after our first attempt. Therefore, we started an initiative to visit selected test beds and record *WiFi* observations along with best guess as ground truth location manually. Recording Ground truth data is done by using finger touch event on an indoor map overlaid inside custom built smartphone application.

Figure 2.5 demonstrates the process in few simple steps. The collected measurements are then processed to identify set of *WAPs* candidates valid for data evaluation. The processing includes identifying geographical area were the *WAP* has been detected with signal strength greater than -75dbm. Area size and number of observations are then compared with thresholds to realise if the *WAP* is valid candidate for sanity test or not.

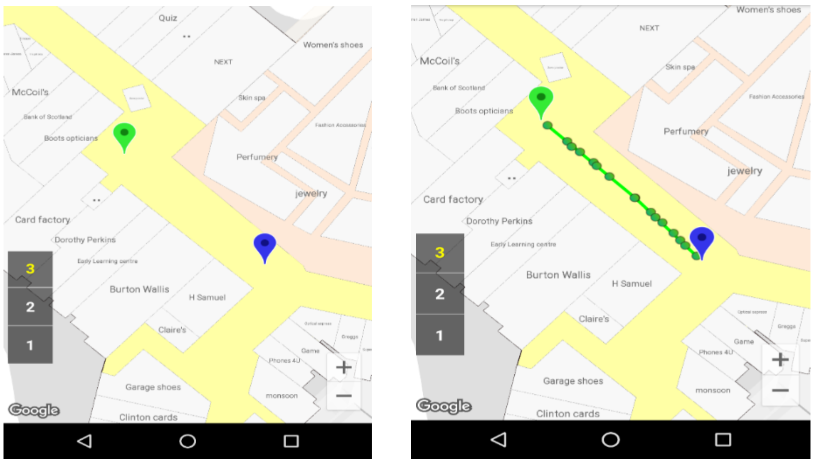


Figure 2.5: Collecting an estimation of ground truth measurements in one of our selected test beds (St.James Edinburgh). The figure shows selecting start and end points (right), then recording measurements as the user walk between them (left)

Based on test we run within university buildings, Kings building campus in Edinburgh, 95% of mac addresses were successfully located within the *-75dbm* coverage area. The 5% can be explained by edge cases where *WiFi* transmitters are located on the peripheral parts of the building so the collected observations didn't capture a qualifying area according to the cut off criteria. In the example below, see Figure 2.6, we marked the -75dbm coverage area in red for one WAP detected in one of our test beds. We also show the histogram for this *RSSI* distribution for the same *WAP*.

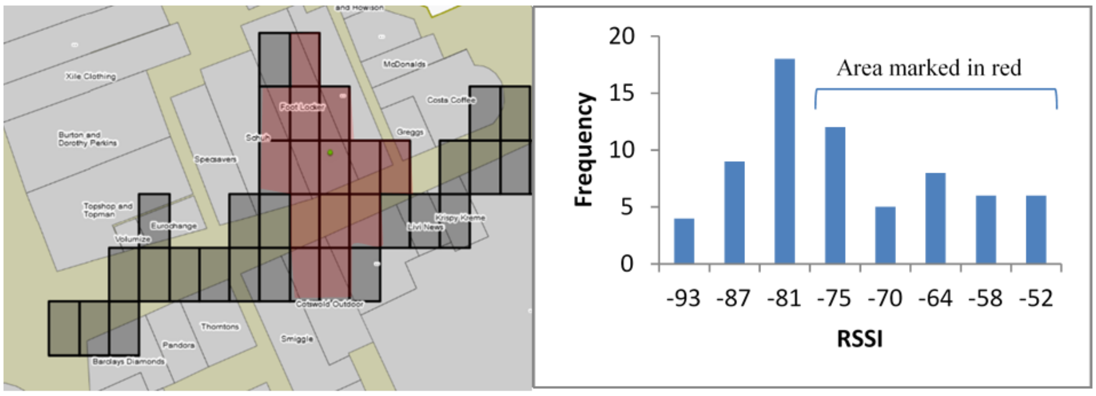


Figure 2.6: Ground truth geometry estimation for *WAP* -75dbm coverage area recorded in one of our testbeds (The Centre, Edinburgh). Map coverage (right). Histogram (left)

To finalise, sanity tests can then be executed by comparing the crowd-sourced location of each candidate *WAP* with its ground truth known area. Errors in the database can then be derived by calculating the shortest distance between the estimated *WAP* location and the perimeter of its -75dbm area from ground truth database. To perform this, we have utilised *MySQL* spatial function *ST_DISTANCE-* [1] for calculating distance between two geometries, where one of them is constructed as point using longitude, latitude as coordinates. and the other is the pre-calculated geometry. The figure below presents an example of distance calculation between point coordinates and polygon geometry.

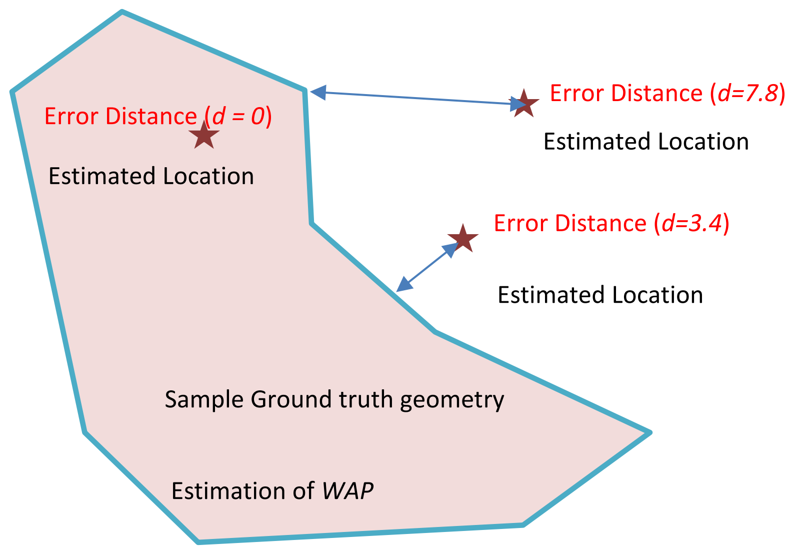


Figure 2.7: An illustration of how we calculate errors in location *WAPs* compared to reference coverage area of -75dbm

[1] https://dev.mysql.com/doc/refman/5.7/en/spatial-relation-functions-object-shapes.html#function_st-distance

2.2.2 Known smart phone tests

In these tests, we collected WiFi observations for one minute in each test point within every test bed building. This would usually provide us with an average of 10 observations per test point, depend on the phone hardware specifications *WiFi* scanning time may vary. At the same time, the ground truth location L_0 of each test point is recorded by pointing on an indoor map using mobile app specifically developed to do so. However, we still estimate the user errors of such process by circle with 5m diameter around the map extracted ground truth location to substitute for any minor map errors or fat finger user errors. The extracted ground truth data files are then stored to perform smart phone positioning sanity tests. Figure 2.8 show the final layout of ground truth points in St.James shopping centre in Edinburgh.

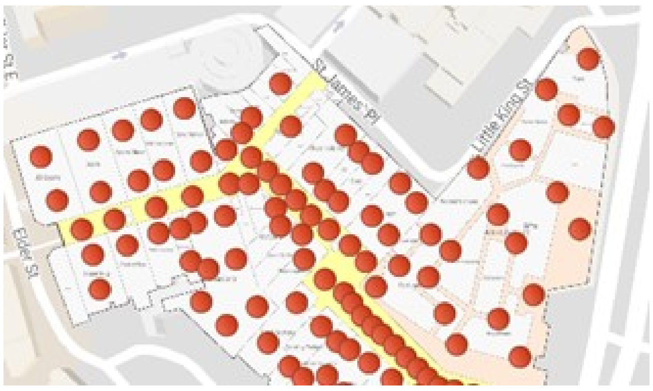


Figure 2.8: Recording ground truth on an indoor map layout over mobile screen.

When executing such sanity tests, we used snapshot of fully crowd-sourced *WAPs* database to query for smart phone best fit position L_t using individual observation from the extracted ground truth data files. The two positions are then compared to calculate positioning errors D_{err} as per the equation (2.1):

$$D_{err} = MAX\left(\left(d_{geo} - 5\right), 0\right) \tag{2.1}$$

where d_{geo} is calculated as geometry distance between two points.

2.2.3 Selecting test beds for evaluation

As stated above, our evaluation process evolved around streaming geographically selected sets of observation covering test areas around various parts of the world. The cluster of servers used to process and crunch these numbers are covered in next section. This section only describes the produced dataset and the tests performed to verify data quality. Once all data is processed, we obtained an estimation of where each *WAP* is located globally. The output is then used to perform sanity tests as prescribed in previous section. Below we have provided illustration of *WAPs* distribution in selected cities or areas that we will be using throughout these tests.

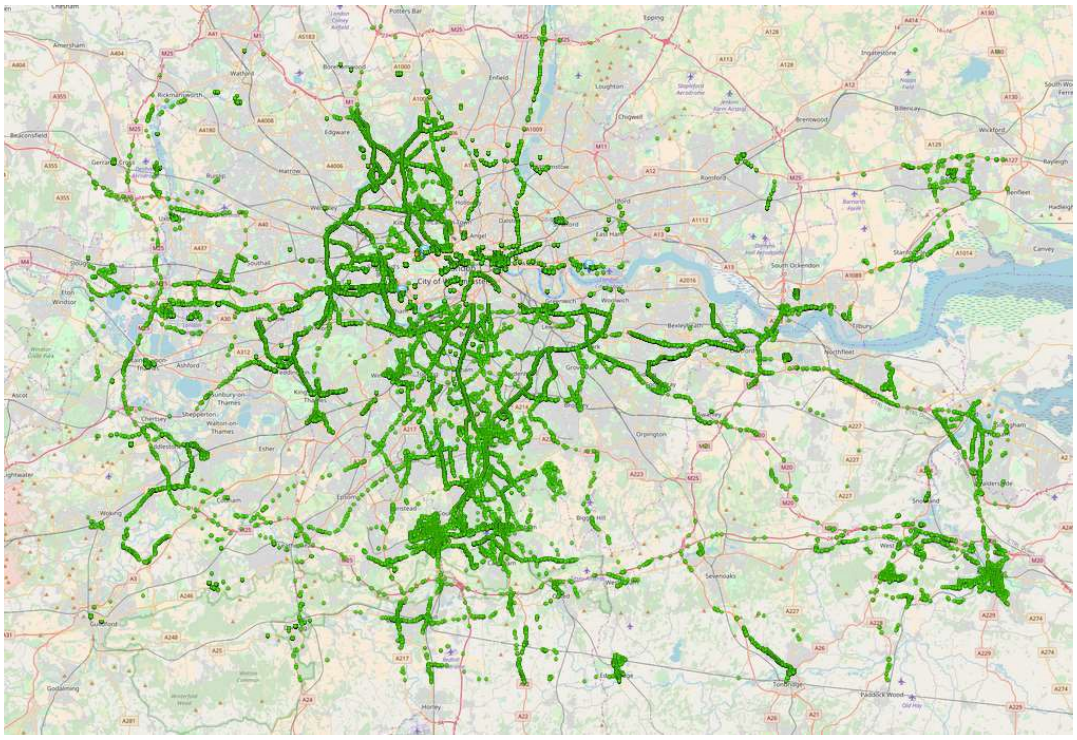


Figure 2.9: ***WAPs*** **recorded in the Greater City of London showing the coverage and distribution**

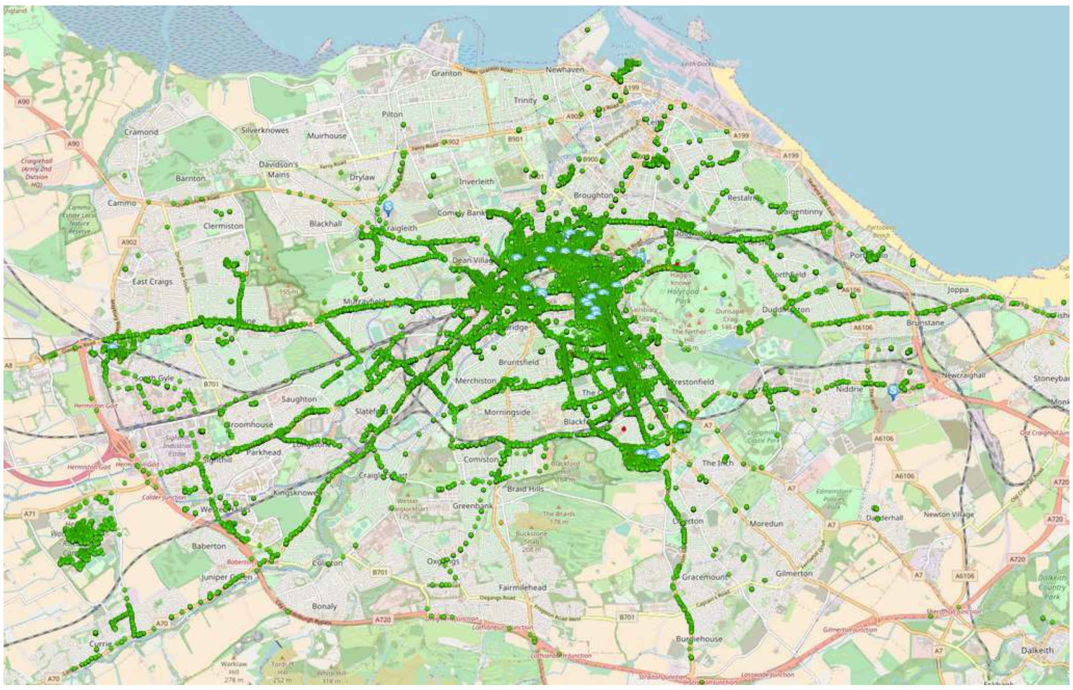


Figure 2.10: ***WAPs*** **recorded in the City of Edinburgh showing coverage and distribution**

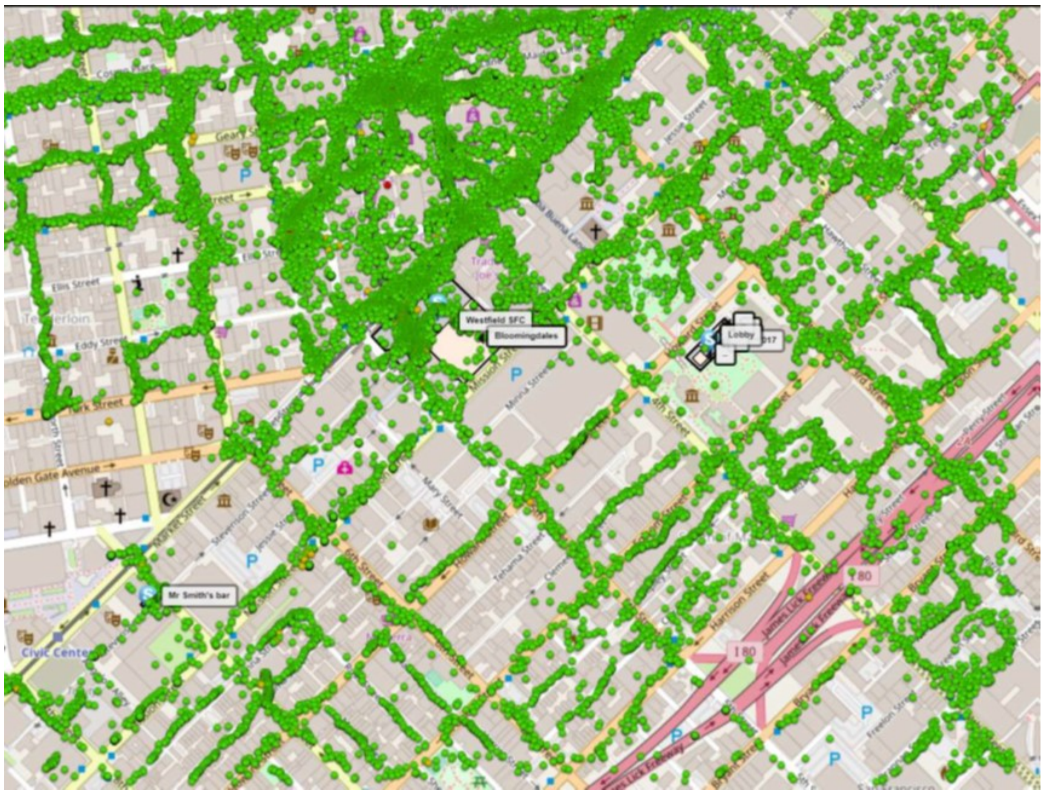


Figure 2.11: ***WAPs*** **distribution across San.Francisco downtown, California, USA**

These three selected areas show major densities outside China. As can be observed from the figures, most of the distribution correlates with main roads and sectors in each city, clearly visible in San Francisco. This can be explained by the uncontrollable collection as these data came from specific use cases assigned by the host application. For example, a map or navigation mobile app users would only use it when driving or walking through complex territory such as shopping mall, University or airport. However, the special case of China was more of social networking app that is used everywhere including public areas, restaurants, work offices or even homes. With large set of data around Beijing city, we were unable to plot each *WAP* to show details distribution. Therefore, we have represented the distribution by plotting grid density map for Beijing and surrounding areas in China.

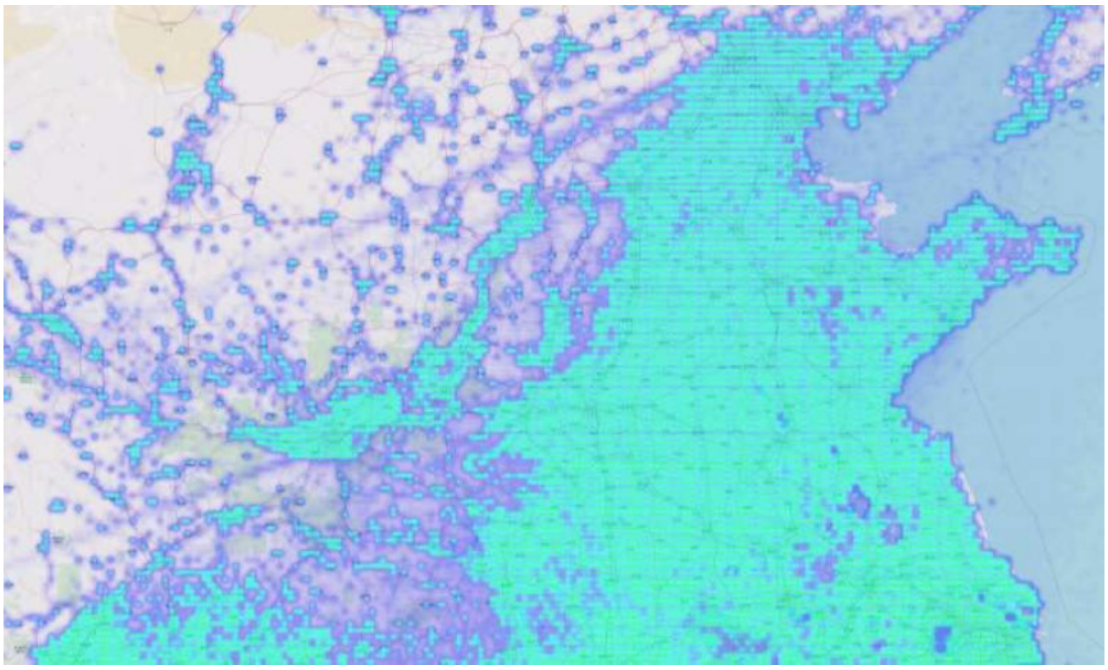


Figure 2.12: *WAPs* density around Beijing shown in heat map style based on grid.

As we have now identified four major areas globally for performing sanity tests, we then nominated specific buildings in these selected cities based on ground truth data availability, building dimensions, *WAPs* density and accessibility. We attempted to cover various types of areas with variations of *WAPs* density, but to be noted Edinburgh got our maximum attention due to the ease of access to collect ground truth. Table 2.3 shows our list of selected test beds.

Table 2.3: List of venues used to perform sanity tests and verify results.

CITY	VENUE	WAPS	SIZE
EDINBURGH, UK	UoE – Alrick Building	*112*	*56m x 17m*
EDINBURGH, UK	UoE – Hudson Bear Building	*89*	*38m x 38m*
EDINBURGH, UK	UoE - Sanderson Building	*49*	*62m x 49m*
EDINBURGH, UK	St.James Shopping Centre	*913*	*238m x 212m*
EDINBURGH, UK	TheCentre Shopping Centre	*547*	*687m x 334m*
LONDON, UK	Westfield Shopping Centre	*2346*	*384m x 384m*
SAN.FRANCISCO, USA	Westfield Shopping Centre	*3488*	*174m x 167m*
BEIJING, CHINA	ECMAll Shopping Centre	*1249*	*159m x 109m*
BEIJING, CHINA	DreamPort Shopping Centre	*3653*	*393m x 116m*
BEIJING, CHINA	Aegean Shopping Centre	*2816*	*172m x 96m*

For each of these identified venues we obtained test points labelled data to identify set of *WAPs* that satisfy the known *WAPs* test criteria stated in 0. Then reference area for each candidate *WAP* is extracted using test points reported *RSSI* values. Below we provide summary of each individual venue to demonstrate its suitability.

Table 2.4: Statistical numbers showing coverage of sanity tests qualified measurements in each selected test bed.

VENUE	TOTOAL WAPS	TEST POINTS	QUALIFIED WAPS	%
ALRICK BUILDING, UK	*112*	*18*	*24*	***21%***
HUDSON BEARE BUILDING, UK	*89*	*23*	*32*	***36%***
SANDERSON BUILDING, UK	*49*	*27*	*17*	***35%***
ST.JAMES, UK	*913*	*59*	*238*	***26%***
THECENTRE, UK	*547*	*86*	*298*	***54%***
WESTFIELD, UK	*2346*	*56*	*911*	***39%***
WESTFIELD, USA	*3488*	*43*	*1587*	***45%***
ECMALL, CHINA	*1249*	*28*	*836*	***67%***
DREAMPORT, CHINA	*3653*	*98*	*1765*	***48%***
AEGEAN,CHINA	*2816*	*34*	*1912*	***68%***

2.3 Creating scalable framework

One major challenge of any algorithm that adopt global deployment approach is that it requires extensive maintenance efforts to deploy in larger scale. This includes two major challenges: scalability of data processors and searchable data storage. As shown in literature, all prior art research suggests storing radio-map of an area permanently [3]. This radio-map varies in size and structure, but in general they share the same scalability limitation when more than one venue is included in the data storage. In our approach there is no static radio-map as it changes every time new observations are received which help the system to overcome the recalibration overhead in maintaining accurate database. However, such flexibility amplifies the challenge of searchable data storage and data processors scalability as data constantly changes. Therefore, it was in our interest to present scalable data structure as part of the research methodology justifying its validity for universal coverage.

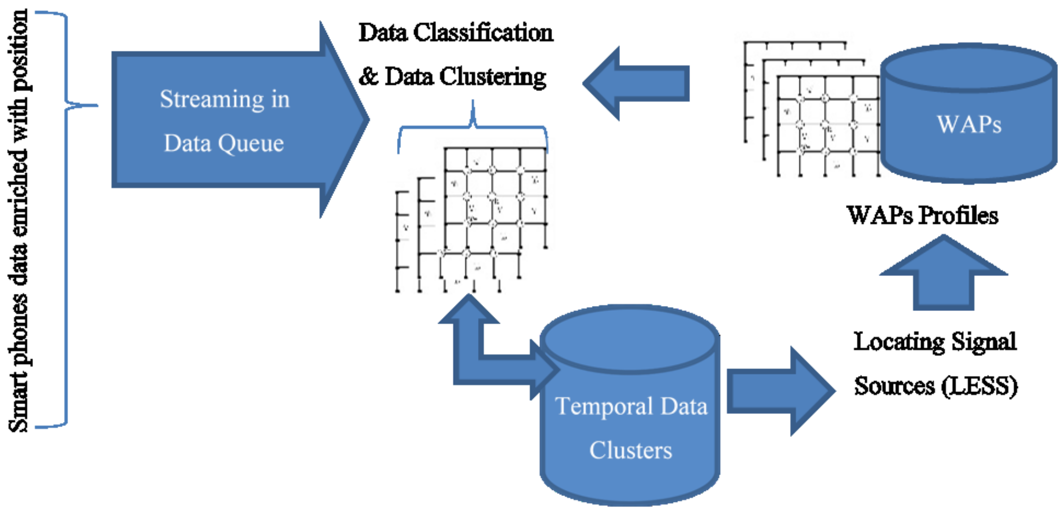

Smart phones data enriched with position
Streaming in Data Queue
Data Classification & Data Clustering
WAPs
WAPs Profiles
Locating Signal Sources (LESS)
Temporal Data Clusters

Figure 2.13: the proposed data processing framework for crowd-sourcing showing multi-stage processing algorithms to generate temporal profiles and execute *LESS* in an adaptive loop.

Figure 2.13 shows the design of our data processing framework stating its high-level components that we are going to cover its detail in next chapter. As can be seen from the graph, we assume that smart phones are responsible for obtaining observations. At the same time, location data can either be obtained from native mobile sensors, such as *GPS*, or by performing position estimation, based on snapshot of relevant data from the "WAPs Data Storage". Hence, we assume that all observations streamed into the proposed algorithms would already have some level of position associated with it. Therefore, any utilised data queue can easily be configured to perform the initial categorisation of data based on geographical location index and MAC address. The geographical location index would then be used to obtain reference clusters from global grid to update the temporal state of these clusters ensuring they contain all valid data streamed in. In addition to reference clusters, MAC addresses are used to obtain location data for each *WAP* in relation to the identified cluster to enable the optimisation performed in *LESS* [40].

2.3.1 Data structure:

Considering the scale of our dataset, accessing and locating data is designed to perform seamlessly when all queries to database are performed using the correct index. It is even more trivial that the key has some representation of data distribution, so data segmentation can be performed efficiently between various data regions with an interconnected cluster of servers. Hence both our data indexes, MAC address and geoindex, are carefully selected to make sure that any query to database is made by one of these two keys. The tables below describe the proposed data storage structure for WAPs, Table 2.5, and temporal Data Clusters, Table 2.6.

Table 2.5: Data structure for WAPs describing access through data keys (vertical) or data fields (horizontal)

Key \ Fields	**Geoindex1**	**Geoindex2**	**Geoindex3**	**Geoindex4**	**Geoindex5**
MAC1	{Binary Data}		{Binary Data}	{Binary Data}	
MAC2		{Binary Data}		{Binary Data}	{Binary Data}

Table 2.6: Data structure for Temporal Data Clusters describing data keys (vertical) or data fields (horizontal)

Key \ Fields	**MAC1**	**MAC2**	**MAC3**	**MAC4**	**MAC5**
Geoindex1-Cluster1	{Binary Data}		{Binary Data}	{Binary Data}	
Geoindex1-Cluster2		{Binary Data}		{Binary Data}	{Binary Data}
Geoindex2-Cluster3	{Binary Data}	{Binary Data}	{Binary Data}	{Binary Data}	{Binary Data}
Geoindex2-Cluster4	{Binary Data}	{Binary Data}		{Binary Data}	{Binary Data}

Using such data structure has provided us the flexibility to access almost instantly, less than 10ms, any record in our data storages using the designated data-keys and data-fields. For example, updating the probability distribution of MAC m in Cluster x in geoindex g, can use the data key g-x and data field m to write or read only the required binary data. This is now very common practice of big data and supported by all big data frameworks [41], [42], [43], [44], [45], [46].

To conclude, our system design relays on adaptive loop continuity to process incoming data and optimise the estimated position of access point simultaneously. This has enabled us to overcome the requirement for processing vast amount of data at the same time. Hence global scalability of our data processor can be achieved by adding any number of processors required as data keys can be used to prevent racing conditions or duplicate processing. However, in this research we only executed one data processor.

2.3.2 Grid and sub-grid hierarchy

In this section we aim describe how our global grid [47] are built for data classification and clustering. Designated Grid filters has previously utilised for tracking smart phones indoors [48]. However, such grids were more of local representative of cells that are defined and stored in pixilate array of shapes, either squares or octagons [49].

Utilisation of the grid as data classifier is just the reverse. Each grid-cell present the belief of set of rules governance what observations could be submitted by a smart phone within the area of such cell. Therefore, we propose that each grid-cell would host an approximation of probability distribution concerning pairs of *RSSI* and *WAPs*. Such beliefs are continuously updated as we receive more observations located inside each grid-cell. An obvious implementation would consider fitting an environmental model to each *PDF* or to customise grid size [50]. However, in the final shape of our proposed framework, we were keen to remove any dependencies on pre-knowledge of the surroundings and end up utilising evenly distributed grid. Therefore, we chose to define the grid as a function of location based on geo-indexing as shown in the figure below.

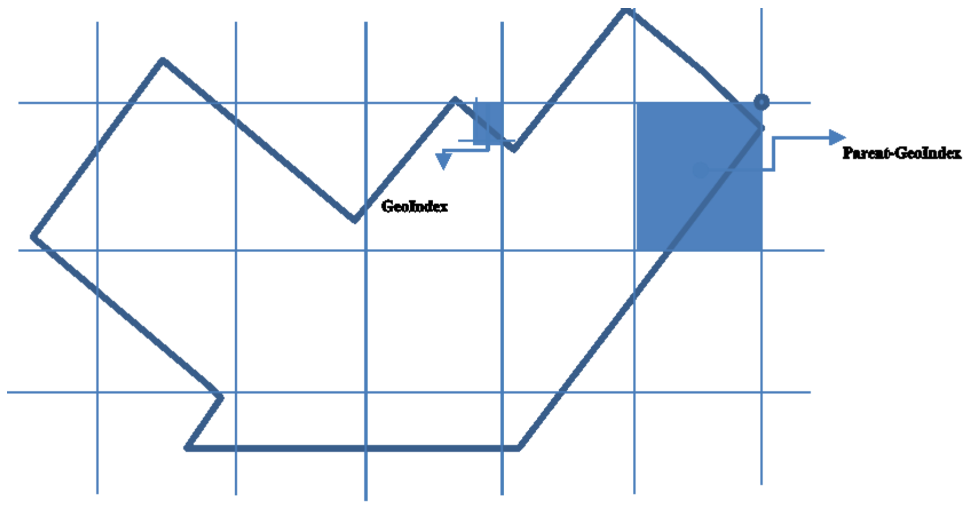


Figure 2.14: The proposed global geo-indexing in square grid supporting multiple level in hierarchy structure where larger geoindex contain set of smaller geoindex areas.

As per any Grid-based filters, our approach aims to represent the globe as small cells of enough size to drive positioning accuracy. Therefore, we chose to render each set of latitude and longitude up to 4 decimals as an indication of 10m to 20m length area. This made our grid operating without previous knowledge of the building shape, location or boundaries.

2.4 Conclusion

In this chapter, we have defined the source of the dataset used in our research. Our definition went on to describe sanity tests and methods used for accuracy justification. We have also provided modular overview of the framework, data structure and georeferenced grid for processing such large dataset. We believe that this chapter has provided justification for the validity of our methods and dataset to locate WAPs on global scale. The rest of the chapter also provides brief description for platform realisation and big data integration as it adds data storage indexing and multi-segment key to enable large scale deployment.

3 Locating WiFi Signal Sources

As demonstrated in the discussion of chapter one, scalability and applicability of crowd-sourcing based algorithms was a major limitation when studying prior art solutions. Instead of attempting to permanently store and interactively locate unlabelled observation to frequently optimise *WAPs* location [26], [35], the proposed solution attempts to label every observation in real time. However, it is still possible to perform initial localisation for labelling observations periodically every 1 to 10 minutes. To enable such labelling in real-time, our approach generates an adaptive descriptor of each *WAP* denoted here as "*WAPs temporal profile*".

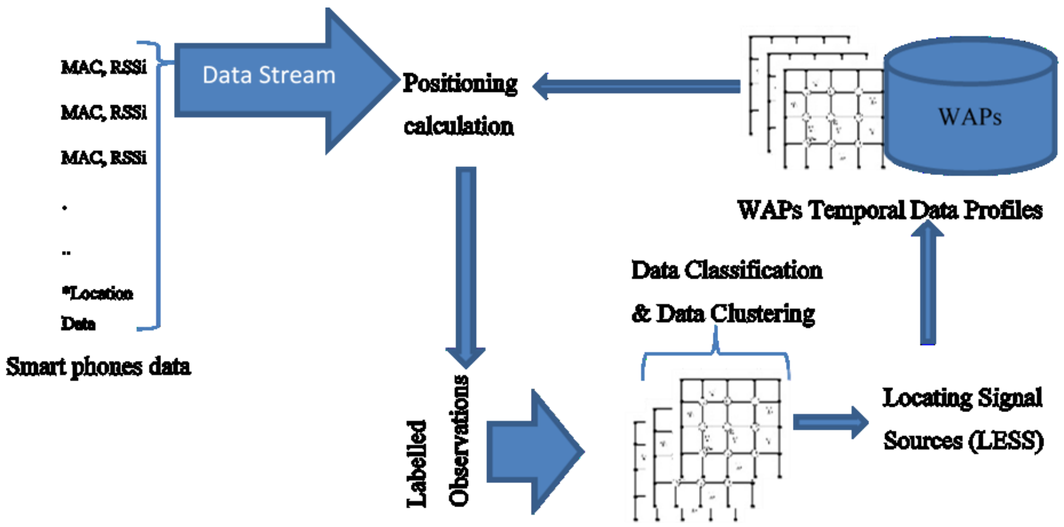


Figure 3.1: Overview of adaptive system architecture based on continuous signal observations streamed and converted into temporal signal profiles

In order to reduce the computational power and possible influence of multi-floor signal distribution errors, we adopted an approach to separate horizontal and vertical attribute of *WAPs* location. By introducing such segregation of (x,y) from the z during the phase of *WAPs* position estimation, we were able to solve one problem at a time. In addition, this approach were essential to enable deployment on large scale by solving for global minima that satisfy (x,y) separately from the global minima of z. Furthermore, we argue that the representation of z in meter is not valuable on the mass scale as humans can't easily make sense of z values without pre-knowledge of building vertical layout measurements. Hence, we propose to solve for (x,y) only to locate *WAPs* while we add floor level as a tag in *WAP* temporal profile instead of z value in meters. However, as discussed in the introduction, we will introduce algorithms to solve for floor estimation as substitute of z in chapter four of this thesis.

As demonstrated in Figure 3.1, the major contribution of our proposed architecture is its complexity reduction saving majority of computational costs compared to the frameworks suggested in literature [15]. This is mainly due to the temporal profiles and reference clustering that allow us to process each observation only once making sure observations are consumable by data streaming. This chapter takes you on the journey of transforming labelled observations into temporal *WAPs* profiles. For the rest of this chapter, we denote our input as set of discrete and independent observation

$O = \{O_1, O_2, \ldots O_n\}$

where each observation O_i is defined as:

- $M_i = \{m_1, m_2 \ldots m_m\}$ set of *WAPs*
- $RSS_i = \{r_1, r_2 \ldots r_m\}$ corresponding set of received signal strength
- $L(t)$ location estimation in time *t* with covariance $Q(t)$.

Having location estimation L(t) defined as temporal entity makes it only valid from time t till time $t+\varepsilon$ with an error vector defined in covariance $Q(t)$. The problem is then contained in calculating the location for *WAPs* $\{M\}$ based on their appearance in $\{O\}$ and their temporal profiles P_m. Each observation, O_i presents state of WiFi signals detected by mobile unit at location in space defined as L_i. Obviously, values represented by $L_i(t)$ is either estimated by an external measurement, such as *GNSS* sensors on the mobile unit, or generated by the same system based on previously generated *WiFi* access points profiles P_m. Therefore, we also define DS_i as a parameter of $L_i(t)$ to flag the dependency of $L(t)$ on the temporal state of P_m.

Furthermore, in this research, we also expand the definition of location of any observation, cluster of observations or *WAPs* to be function of time as all parameters of crowd-sourced data is subject to change beyond the defined uncertainties.

3.1 Data Classification & Data Clustering

As new entries, or observations, are received and streamed into data classification and clustering algorithms, each observation would be placed correctly in the global grid. Similar to the usage of radio-maps for fingerprinting database, signals similarity is a key feature for data classification into grid locations. In our proposed classifier, each geoindex G_x in the grid hosts multiple hypothesis of what *WiFi* signals should be observed by devices visited G_x. Each hypothetical cluster is represented by *PDF* of signal strength *RSSI* measures modelled as a function of Gaussian Distribution (M_i,μ_i,σ_i). Where M_i is the MAC address of *WAP* belong to this cluster, μ_i is the mean *RSSI* and σ_i is the standard deviation of all observations fused in the cluster head CH_i.

The utilisation, or selection, of normal distribution is justified by the fact that signal distraction and multipath would randomly affect measurements taken within limited area. With the proposed clustering and classification of *RSSI* measurements into 10mx10m grid, the noise can be assumed random fading out of the mean value. Furthermore, as experimentally proven the condition of having random variate of RSSI measurements is satisfied as at least 65% of values fall within one σ if the mean [16], [23], [51]. All of this has encouraged us to force Gaussian distribution into the measurements as we stream them to be located into given geoindex of our global grid. Finally, with enough measurements accumulated into each geoindex, the representation of mean *RSSI* as Gaussian gets more accurate.

Finally, our proposed solution employs an adaptive density k-mean clustering [52] on all observation classified to belong to geoindex G_x. The rest of this section describes clustering algorithm and geoindex classification in more details.

3.1.1 Implementation of k-mean for geoindex clustering

K-mean algorithm is commonly used for classification and clustering of large amount of data. The basic clustering process appears in many data mining and data analytics research. The algorithms often depend on one parameter to measure distance between elements and what is defined as cluster heads. Therefore, it has been utilised by fingerprinting based indoor positioning to reduce calibration points by clustering them into k clusters based on pre-defined k. As the primary distance calculation present in *WiFi* indoor positioning solutions is based on *RSSI*, uncertainty should be considered. In our research we employed density k-mean clustering [53] [52] to deal with uncertainty of distance estimation.

The first step on k-means is to determine number of clusters k, which could be randomly set to choose any k objects as initial cluster heads. However, as we deal with continuous stream of data and limited distribution, only within one geoindex, we allow k to change as the system process further patches of observations. Such adaptive implementation of k-mean clustering is simple to implement recursively in continuously changing environment.

For example, let geoindex G_x be an area where the system process observations for the first time. Starting with $k=1$ all observations in the first patch will then form an initial cluster where the cluster head is determined based on density function λ_i representing the density of observations around an observation O_i. The steps below illustrate how we perform an adaptive clustering:

1- Let m be the number of observation in the patch the system is processing, and $d(i,j)$ be the Euclidian distance between O_i and O_j.

$$Q = \begin{bmatrix} d(1,1) & \cdots & d(1,m) \\ \vdots & \ddots & \vdots \\ d(m,1) & \cdots & d(m,m) \end{bmatrix}$$

2- Define average distance D_{avg} as per (3.1).

$$D_{avg} = \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} d(i,j) \tag{3.1}$$

3- Define the density function λ_i as per (3.2).

$$\lambda_i\ (O_i) = \sum_{j=1}^{m} \hat{X}\big(d(i,j) - D_{avg}\big) \tag{3.2}$$

$$\hat{X}(c)\begin{cases} = 0 \ \ when\ c > 0 \\ = 1\ when\ c < 0 \end{cases}$$

4- Find O_i with maximum density and use it as initial cluster head for the first cluster CH_1

5- Define a Boolean exit flag as per (3.3)

$$\text{Exit } = \hat{X}\left(\max_{Q}\big(d(i,j)\big)\ -\ \varepsilon\right) \tag{3.3}$$

Where: ε is the maximum Euclidean distance allowed in a geoindex.

6- If $Exit = 0$, remove all observations where $d(O_i\ , CH_k)\ \leq\ \varepsilon$, else stop the process and return list of CH_1 .... CH_k

7- Repeat to locate CH_{k+1}

Once set of cluster heads, Ch_1 ...CH_k, is finalised, the usual K-mean is applied to classify each observation into the nearest cluster. Then all observations belonging to each cluster is fused to recalculate new cluster head. However, the fusion of multiple observations is performed assuming that *RSSI* measures follow normal distribution, which is valid in a limited space such as our defined geoindex. Therefore, we proposed to separate each observation into set of WAPs before fusion to allow modelling *RSSI* measurements into set of probability distribution functions $PDF_i\,(M_i, G_x)$.

To show our proposed fusion algorithm in more detail, we could summarise it in symbolic collection of mathematical structures, arrays. Let $[O_1\text{-}O_k]$ be set of observations clustered for fusion to create CH_x. The first step in the fusion process is creating list of combined *WAPs* $\vec{M} = \bigcup_{i=1}^{K} m_i$, where each unique *WAP* $m_i\ \in\ \vec{M}$ is described by the following structure:

- ✓ K_i: number of times m_i appeared in the set of observations.
- ✓ $RSSI_i\,[r_{i1} - r_{ki}]$: an array of all *RSSI* measures for *WAP* m_i.

- ✓ *Δ_i*: The score of m_i compared to all *WAPs* in $\vec{M}$, calculated by equation (3.4).

$$\Delta_i = \frac{\max\limits_{j\,:\,1\to K_i}(r_{ij})}{2 \times \max\limits_{i:\,1\to K}\left(\max\limits_{j\,:\,1\to K_i}(r_{ij})\right)} + \frac{K_i}{2 \times K} \qquad (3.4)$$

Where:

K_i is the number of *RSSI* readings for the *WAP* m_i

r_j is the *RSSI* reading *j* of *WAP* m_i

K is the total number of readings in the cluster

From the above equation we intend to represent the quality of given *WAP* in a form of score Δ_i. In the literature, it appears that there are two methods of calculating the robustness if *WAP* appearance in an area as fraction $\frac{K_i}{K}$. Such metric only introduces how many times it appears compared to the total number of observations within the cluster head *CH*. However, we wanted to introduce an influence factor added as fraction $\frac{Maximum\ RSSI\ for\ WAP\ m_i}{Maximum\ RSSI\ for\ all\ WAPs}$. Hence, we assigned a weightage of 0.5 to robustness and influence metrics and combined them in one score metric.

Once the score is calculated per *WAP*, the next step is to filter out any *WAP* that is not scoring high enough compared observations in batch we are processing. In our research, we used threshold of $\Delta_i > 0.2$ to remove unreliable *WAPs* from the temporal profile we are creating to describe each Cluster head. Finally, all *RSSI* measurements, associated with each *WAP* passing the score filter, fed into normal distribution function to extract (k_i, μ_i, σ_i) to compose P_μ.

$$\forall m_i \in \vec{M} :: Norm(RSSI_i) \xrightarrow{yields} \begin{pmatrix} \mu_i \\ \sigma_i \end{pmatrix}$$

As *RSSI* normal distribution function produces temporal profile for the cluster CH_x, location data of all observations would also be fused together to compose location l_i of the cluster. To obtain this location we utilised an improved version of weighted-centroid [25], [54]. However, in our implementation of weighted centroid, we didn't only consider *RSSI* when computing weight per reference point. Instead, we have included the uncertainty in observations localisation errors. Hence, each observation O_i is assigned with weight w_i based on *RSSI* differential values to the fused signal profile P_μ. Equations (3.5) and (3.6) demonstrate how the fusion is done based on m observations and n *WAPs* in P_μ. *WAPs* not part of P_μ are ignored as they do not contribute to the final output of the fusion function.

$$L_\mu = \frac{\sum_{i=1}^{m}(w_i . l_i)}{\sum_{i=1}^{m} w_i} \tag{3.5}$$

$$w_i = \frac{n}{\sum_{j=1}^{n}\left(10^{\frac{|r_{ij} - \bar{r}_j|}{10}}\right)} + \frac{1}{det(q_i). ds_i} \tag{3.6}$$

r_{ij} represents *RSSI* measure for *WAP* with index *j* in observation O_i.

$\bar{r}_j$ represents *RSSI* measure for *WAP* with index *j* in the cluster head signal profile P_μ.

q_i represents the covariance matrix for location l_i in observation O_i.

ds_i represents the dependency score for location data l_i in observation O_i.

Before we complete the reference cluster data structure, we also include covariance matrix Q_μ. In another words, our fusion function estimates the spread of location data among the cluster in a form simple *(x,y)* covariance matrix Q_μ, as shown in (3.7). This value would play an important role when we start consuming the location data to optimise *WAPs* and reference clusters relationship. In addition, another measure of quality proposed in this research is the fused dependency score assigned to each estimated location L_μ. This score is computed by the equation (3.8).

$$Q_{\mu}(\mathrm{x,y}) = \frac{\sum_{i=1}^{m}\left(L_i(x) - L_{\mu}(x)\right).\left(L_i(y) - L_{\mu}(y)\right)}{m} \quad (3.7)$$

$$\mathrm{DS\mu} = \min\left(\min_{i=1:\, i\leq m}(ds_i) + 1, \frac{\sum_{i=1}^{m}(w_i.\, ds_i)}{\sum_{i=1}^{m} w_i}\right) \quad (3.8)$$

Figure 3.2 provides visual description to how the fusion is done in a form of block diagram. The example shown in Figure 3.2 is fusion of k observation generating the total of n qualified *WAPs* on a set of clusters.

In addition to clustering and fusion, the dependency on signals similarity classification to assign observation to specific geoindex forms the second part of this module. Nevertheless, it is more common to calculate the dissimilarity, or *WiFi* distance, as a reverse indication to similarity. In this research we used two similarity calculation algorithms, Euclidian distance and PDF based similarity. Additionally, as our grid allow us to retrieve all neighbouring geoindices for any observation O_i, similarity is calculated for the geoindex that claim the observation, based on location, as well as all neighbouring geoindices.

To estimate the complexity of our proposed classification and clustering algorithm, we set to measure number of operations a machine would take to perform the proposed solution on a set of N observation with M *WAPs* in each observation. For such case, taking into account the maximum possible execution, we estimated the complexity as per the following list:

Weighted Centroid	$O(2.N.M)$
Combining WAPs and Filter based on Δ_i	$O(N.M)$
Calculating Normal Distribution Parameters	$O(2.N.M)$
Calculating Dependency score	$O(N.M)$

As can be seen from the broken down complexity figures above, total number of operations in the proposed algorithm can be estimated to $O(6.N.M)$. However, as we scale the number of observations N and number of *WAPs* M to large numbers, the sequential execution of 6 components in the algorithm is fixed. Hence the overall complexity can be represented by the total operation in the range of: $O(N.M)$.

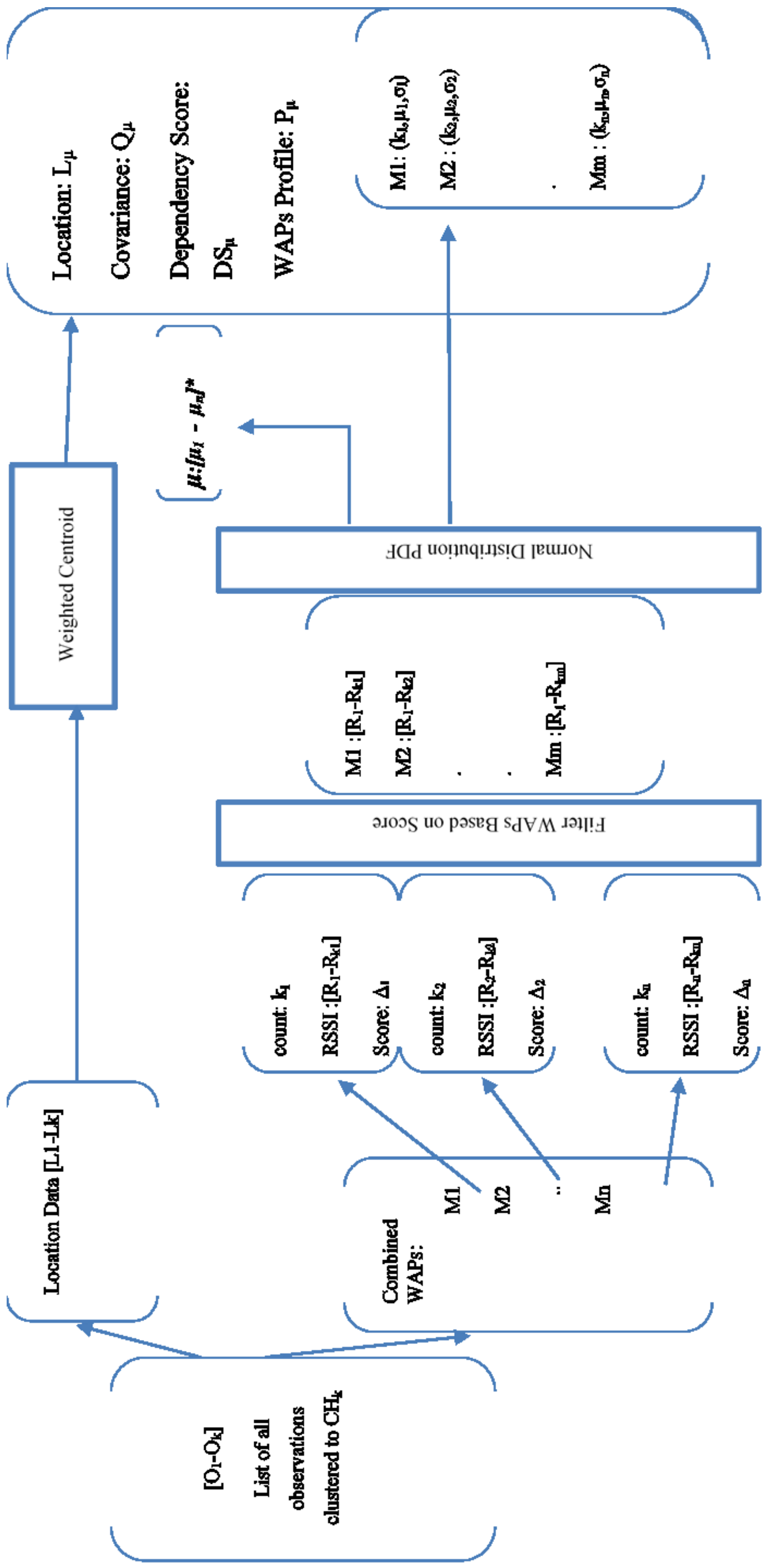


Figure 3.2: Block diagram of fusion algorithm generating temporal profile for set of observations classified into one cluster head. The diagram shows how the input data is formatted or transformed at

3.1.2 Euclidian distance

Euclidean distance is commonly used in fingerprinting positioning algorithms to measure signals similarity between set of calibration points and an online observation with unknown position. Therefore, it is now very well studied to us *RSSI* of several *WAPs* as a measure of Euclidian distance [26] [51]. To achieve this, *RSSI* readings of *n WAPs* could be presented as signals *FP* set $\{rss_{w1A}, rss_{w2A} \ldots rss_{wnA}\}$ for point *A* and $\{rss_{w1B}, rss_{w2B} \ldots rss_{wnB}\}$ for point *B*. The indexes and values in the two vectors should be respecting the order and the availability of *RSSI* measures from all WAPs in the two sets. Then the two sets are used to determining the dissimilarity between signals in point *A* and signals in point B. The normalised distance *d*, the smaller *d* the more similar *A* and *B*, are given as per the Equation (3.9).

$$d(A,B) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(RSS_{w_i} - RSS^{*}_{w_i}\right)^2} \qquad (3.9)$$

n: number of *WAPs* in both sets, both share the same list of *WAPs*.

RSS_{w_i}: is the received signal strength for access point w_i at point A.

$RSS^{*}_{w_i}$: is the received signal strength for access point w_i at point B.

An obvious limitation of the above equation is that the set of *WAPs* in any given two points are very likely to be different. This cover various possibilities from dynamically changing environments, where *WAPs* could change, unstable *WAPs* or observations that are far enough to encounter variation in *WAPs*. To overcome this limitation, the authors of [55] defined thresholds to filter the list of *WAPs* before it is used in distance calculation. However, a simple solution could be used by setting default *RSSI* value to missing *WAPs* from any given observation. In our research we used the same thresholds as per [55] with default minimum *RSSI* set to -90 dbm.

Furthermore, considering the classification problem of crowd-sourced data for dynamically changing environment, we only have one vector for observations as it streams into the classification algorithm. On contrast, our grid is constantly changing as the set of clusters, or temporal profiles, are not set of calibration points. Hence, to measure the similarity of any observation to specific geoindex, we recreate the calibration point *RSSI* set in equation (12) from set of *RSSI* mean values $[\mu_1 \dots \mu_n]$ for each cluster head in geoindex G_x. The distance between an observation *O* and G_x are then estimated as the local minimum of the function $d(O, P_{\mu j})$ where $P_{\mu j}$ is the temporal profile of cluster head *j* in geoindex G_x.

3.1.3 PDF based similarity

Similar to Euclidian distance, this method utilises temporal profiles P_μ in a given geoindex G_x to calculate final probability of an observation O_i belonging to G_x. The concept of this probability calculation assumes that all *WAPs* in the same temporal profile have independent probability distribution. Therefore, the probability of an observation O_i belonging to cluster CH_x with k *WAPs*, is given by equations (3.10) (3.11). To consider the justification and the proof of such probability calculation, readers can refer to [16] for more details.

$$\mathrm{P}(O_i,\ CH_x) \ = \ \prod_{j=1}^{K} PDF\left(r_{ij}\ ,\ \mu_{xj}\ ,\ \sigma_{xj}\right) \tag{3.10}$$

$$\mathrm{PDF}(\mathrm{r}, \mu, \sigma) \ = \ \frac{1}{\sqrt{2\pi}.\ \sigma} . e^{-\frac{1}{2\sigma^2}.(r-\mu)^2} \tag{3.11}$$

r_{ij}: is the *RSSI* reading of *WAP j* in the observation i.

μ_{xj}: is the mean of the *RSSI* distribution of *WAP j* in the profile of CH_x.

σ_{xj}: is the standard deviation of the *RSSI* distribution of *WAP j* in the profile of CH_x.

On the other hand, *PDF* based similarity does provide probability instead of distance. Therefore, local maximum of the probability function (3.10) is direct indication to the probability of O_i to belong to geoindex G_x. Nevertheless, for the consistency of one measurement used in clustering algorithm, distance can also be derived from the maximum probability by the following:

$$D(O_i, CH_x) = (1 - Normalised\,(P(O_i,\, CH_x))) \times MaxD \tag{3.12}$$

$MaxD$: is a configuration parameter represent the maximum allowed distance for an observation to belong to a geoindex. In this research, we use $MaxD = 50m$.

In the normalisation phase of the probability function, as appeared on equation 3.12, we aim to produce linearly distributed values for the function $P(O_i,\, CH_x)$ so that probability numbers cover all range between [0 - 1] when observations get more similar to the given geoindex signal profile. However, as this can’t be guaranteed on all cases, we have limited the use of this distance model to comparison between neighbouring geoindices to avoid the requirement of converting probability numbers to distance.

3.2 Estimating Location of Signal Sources

After we completed the classification and clustering of observation into temporal cluster heads, represented by each Gaussian *PDF* on set of geoindices, we set to examine an iterative trilateration approach to solve for optimal locations for all *WAPs* appeared in our signal observations clusters. Therefore, we developed our novel approach based on mass-spring relaxation. Our proposed implementation of mass-spring consumes reference observations in form of *RSSI*, covariance and location data derived from temporal profiles generated during the classification and clustering stage. The reader should refer to Figure 3.2 for visual dependency of these two processes. Nevertheless, our novel implementation also accommodates for outlier detection before it executes the proposed optimisation algorithm. Therefore, before we describe our algorithm in more details, let's clarify outliers detection and mitigation.

Considering the stream of temporal profiles processed in the patch of time t, denoted here as $TP(t)$, it is very likely that some *WAPs* shows higher level of noise compared to others. Therefore, before we attempt to locate any *WAP*, we compare the set of cluster heads $TP(t)$ with what the algorithm previously processed $TP(t\text{-}1)$ to identify any possible outlier and merge any clusters that show high similarity. The following list of behaviours are monitored by the proposed outlier detection module:

- WAP no longer appear in any observation in the area are removed from $TP(t\text{-}1)$.
- WAPs appear in only one cluster across the processed area are removed from $TP(t)$.
- Matching clusters are fused together. Matching clusters are identified based on condition: $D(CHi\ , CHj) < \varepsilon_1$
  Where ε_1 represents dissimilarity or distance threshold for 10m distance.
- Conflicting score is calculated based on simple voting system as each cluster CH_i adds conflict score of 1 to any neighbouring cluster CH_j if "*Vote Conflicted*" function, as per (3.13), returns *True*.

$$Vote\ Conflicted :: D(\ CH_i\ , CH_j\ ) > \varepsilon_2\ .\ max(CH_j(DS_\mu)\ , CH_i(DS_\mu)\ \ ) \quad (3.13)$$

ε_2: is the maximum dissimilarity/distance allowed between two neighbouring geoindices.
$CH_j\ (DS_\mu)$: is the dependency score for the cluster head CH_j.

Once all clusters are finalised and outliers are mitigated, we are now ready to compose input vector per *WAP* and estimate initial location for each signal source. The sole purpose for the initial estimation is to allow extracting signal propagation parameters to estimate distance to source (*DtS*) value for each *WAP* in each cluster. In addition, an initial estimation also is required for an iterative approach. Therefore, we construct selected set of observation references vector per *WAP*, for example $\overrightarrow{M_\iota}$ combine k selected labelled observations for *WAP* M_i in vector such as:

$$\overrightarrow{M_\iota} \; :: \; \bigcup_{j=1}^{K} \left(L_j \, , \, w_{ij}\right)$$

L_{ij}: is the location of cluster *j* which contain an observation of *WAP* m_i.

W_{ij}: is the weightage assigned to L_j based on P_j -> $(\mu_i \, , \, \sigma_i \,)$ for *WAP* m_i and other quality parameters such as Q_j, DS_j. We compute W_{ij} as shown in Equation (3.14):

$$w_{ij} = \frac{1}{\left(10^{\frac{|\,\mu_i| - \sigma_i - Tr_i}{10}}\right)} + \frac{1}{det(Q_i).DS_j} \tag{3.14}$$

Tr_i is the transmitted power set as 30dbm in this research.

Q_j is the covariance matrix for *WAP* m_i

DS_i is the dependency score for cluster ch_j

Once each *WAP* is allocated with references vector, the process of computing its initial location is simply made using weighted centroid as per equation (3.5). Furthermore, Each *WAP* location is then compared with each entry in references vector to produce list of all geoindices between the signal source and the measurement point. In return this list is used to pull from pretrained database an estimation for each pair if they have line of sight relation (*LoS*) or none line of sight (*NLoS*). The process of obtaining these *LoS* or *NLoS* probability is here referred to as $LoS\ (G_{x1},\ G_{x2})$ function. This function follows the same hypothesis testing classification approach described by [39]. However, in this research we propose an adaptive mitigation instead of static model for either LoS or $NLoS$, we modelled the estimated attenuation between the transmitter and receiver when using log-distance pathloss as per the following:

$$Power(d) = r_0 - 10nlog(d) - \sum_{i=1}^{g} LoS\,(G_i\,,\,G_{i-1}).W \qquad (3.15)$$

r_0: is the reference *RSSI* at 1m distance from the transmitter.

n: pathloss exponent

g: number of geoindices between the reference point and the estimated transmitter location.

W: is an average signal attenuation per wall [23].

σ: is the standard deviation of *RSSI* reported at distance d.

However, as stated above, we are interested in calculating distance to source, rather than $Power(d)$. This can be simply done by considering that $Power(d) = \mu$, where μ represent the mean *RSSI* measurement for the reference point in question. We can then write the same equation as:

$$DtS = 10^{\frac{Pr_0 - \mu - \sum_{i=1}^{g} LoS\,(G_i\,,G_{i-1}).W}{10n}} \qquad (3.16)$$

Finally, we put everything together in Figure 3.3 to demonstrate modular design for the proposed data flow. The proposed flow reads input data as set of temporal profiles for current and previous estimation $CH(t)$, $CH(t\text{-}1)$ and output is the estimated location of each signal source $[M_1\text{-}\ M_n]$. As mentioned before, the same optimisation algorithm also updates reference clusters and output new set of estimated temporal profiles denoted as $CH(t+1)$.

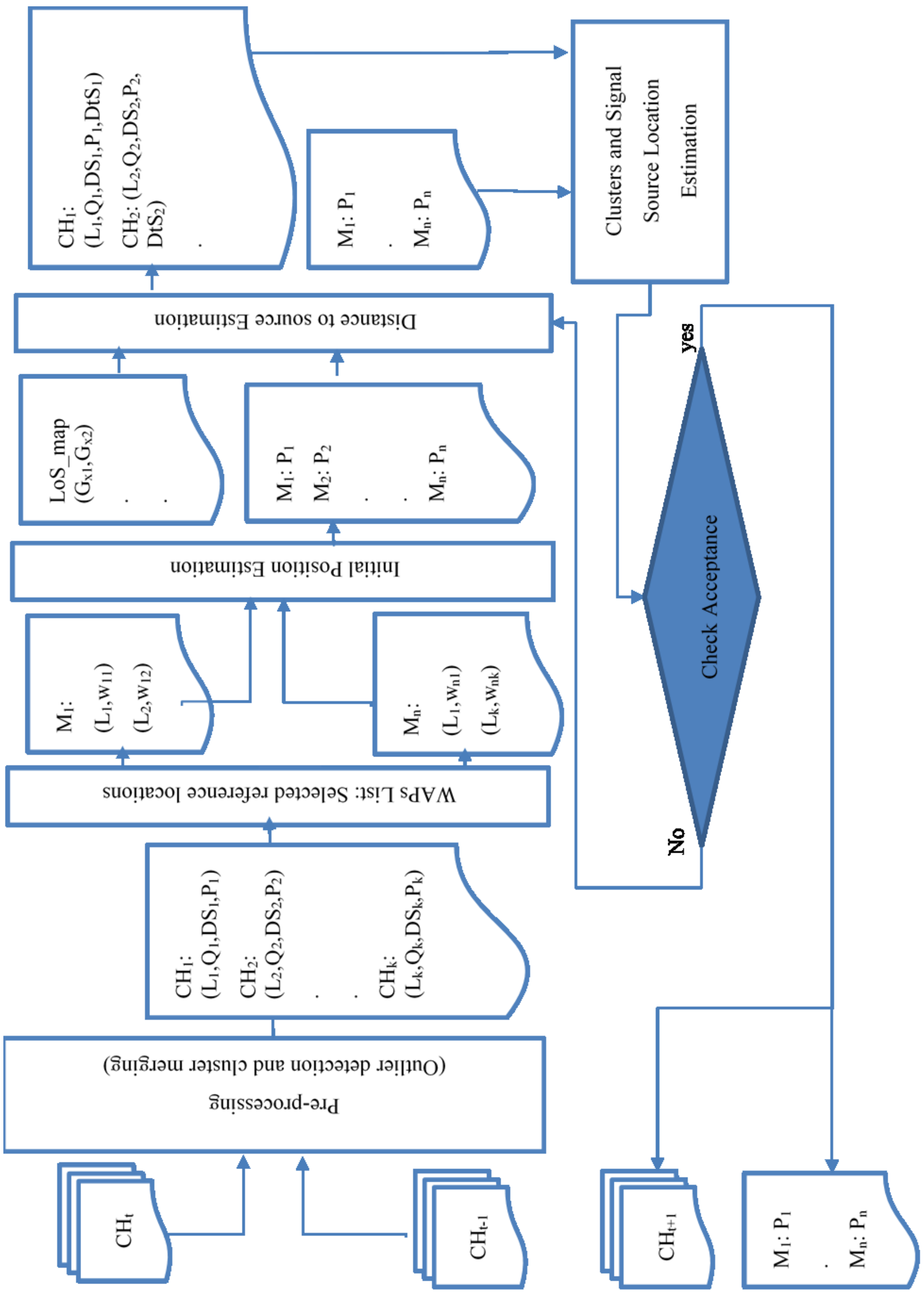

CH_t
CH_{t-1}
Pre-processing
(Outlier detection and cluster merging)
CH_1: (L_1,Q_1,DS_1,P_1)
CH_2: (L_2,Q_2,DS_2,P_2)
CH_k: (L_k,Q_k,DS_k,P_k)
WAPs List: Selected reference locations
M_1: (L_1,w_{11}) (L_2,w_{12})
M_n: (L_1,w_{n1}) (L_k,w_{nk})
Initial Position Estimation
LoS_map (G_{x1},G_{x2})
M_1: P_1 M_2: P_2 M_n: P_n
Distance to source Estimation
CH_1: (L_1,Q_1,DS_1,P_1,DtS_1)
CH_2: (L_2,Q_2,DS_2,P_2,DtS_2)
M_1: P_1 M_n: P_n
Clusters and Signal Source Location Estimation
Check Acceptance
yes
No
CH_{t+1}
M_1: P_1 M_n: P_n

Figure 3.3: Block diagram showing signal source location estimation algorithm. The figure

3.2.1 iterative trilateration

This approach is basically an iterative reduction of distance mean square error cost function. However as shown in literature [5] [35], this is very commonly classified as mutual localisation problem. Therefore, the proposed algorithm should not only estimate the location of signal sources, it also optimises the predetermined location of each reference cluster CH_i. To achieve this we define a local cost function $E(CH_x,P_i)$ as the root square mean error of distance between CH_x, as set of reference clusters $<CH_1,CH_2..CH_k>$, and the estimated position P_i for the *WAP* M_i.

$$E(L,P) = \sqrt{\frac{1}{k}\sum_{i=1}^{k}\left(\sqrt{\left(X_{l_i} - X_P\right)^2 + \left(Y_{l_i} - Y_P\right)^2} - DtS_i\right)^2} \qquad (3.17)$$

Thus, an initial stage is proposed to solve this optimisation problem by finding a value for position P_i of *WAP (i)* that minimise local cost E in (3.17). To identify the local minima, we adopt mass spring relaxation approach [36] by modelling each *WAP* as the centre of gravity between set of springs representing its relation to all reference clusters.

To further describe our mass-spring relaxation modelling based on our predefined terms, let's denote the graph $G:[P_0 , L , D]$ as shown in Figure 3.4. Where P_0 is initial position estimation, L is anchor positions set $<l_1\ l_2 \dots l_k>$ denoted as the positions assigned to each reference cluster and *D* represent graph edges d_{ij} as optimum distance between a given central node position P_i and the corresponding anchor node position l_j. To obtain a stable state of mass-spring relaxation we would place the central node *P* of graph *G* in the optimum position that satisfy all edges in *D* without relocating anchor nodes *L*.

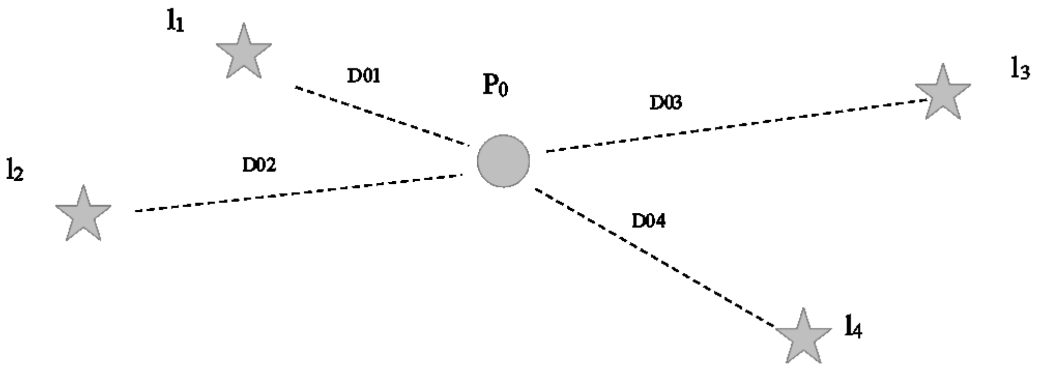


Figure 3.4: Example of mass-spring relaxation shows central node position P_0 and set of anchor references l_1-l_4 in a graph.

To generalise, let’s try to present any given wirelessly connected network as graph components defined as per the following list:

Anchor node (l_j): is a node that we were able to estimate its position with covariance, or error, less than threshold. However, first phase of such iterative relaxation always treats all reference clusters as anchor nodes.

Central node (P): is a node that we are optimising its position or its position is still unknown. For this phase we are representing each WAP as central node. However, each reference cluster could also be central node in a graph, if WAPs were presented as anchor nodes.

Relaxed Edge Distance (d_i): is the distance estimated between two nodes based on the reported signal loss in any communication between them. In our case this is denoted by the estimation of distance to source DtS_i as graph edge between each WAP and reference cluster.

Graph Edge Distance ($\widetilde{D}_i$): is the distance estimated between two nodes based on their estimated positions.

Connectivity Degree (c): is a numerical measurement of how many hops are required to estimate "*Edge Distance*" between two nodes. The example shown in graph *G*, demonstrate connectivity degree of 1. However, in this research we would utilise connectivity degree only up to 2, where second level of connectivity is used to estimate the edge distance between two WAPs.

Graph Tension ($\tau = \sum \tau_i$): is the sum of persistence errors between estimated positions of all graph nodes and their relaxed edge distances. In other words, this represents the sum of tensions caused by anchors nodes place too close or too far from central node.

In this research we estimate the tension between two nodes as the absolute difference between relaxed edge distance and graph edge distance for the edge between them.

$$\tau_i = \widetilde{D}_i - d_i \tag{3.18}$$

To avoid the usual computational overhead of managing multidimensional graphs of complex network, an iterative optimisation approach is adopted. In practice, we aim to achieve an optimised state of each network of reference clusters and *WAPs*. The optimisation process can be formulated as minimisation of graph tension τ over vector points $\{P, <l1,l2,...l_k>\}$. These tensions then enforce change in the positioning of *P* based on the following calculated force:

$$\vec{F} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\tau_i.(\vec{P} - \vec{l_i})}{\widetilde{D}i}\right) \tag{3.19}$$

Hence, on each iteration we attempt to minimise the total force F by moving the estimated position of central node P in the same direction as total force F. The figure below illustrates the modelling on mass-spring during an optimisation of central node position.

3.2.2 Single Connectivity Mass-Spring

Based on single connectivity problem, we only attempt to optimise each node in the graph, based on its distance to plurality of nodes those share an edge with it. Moving back to mass-spring algorithm, the algorithm modelled nodes as masses and edges as springs spread between them. The natural length of each spring in its relaxed state is optimum relaxed distance d_i. Nevertheless, as springs can be compressed or stretched to allow a given graph distance r_i, it develops tension τ_i and force f_i trying to go back to relaxed state. The basic gravity law applies to the relationship between these masses and springs governing positions, tensions, and forces on the network at any time. General network optimisation of mass-spring model resolves by finding optimum positions of all masses.

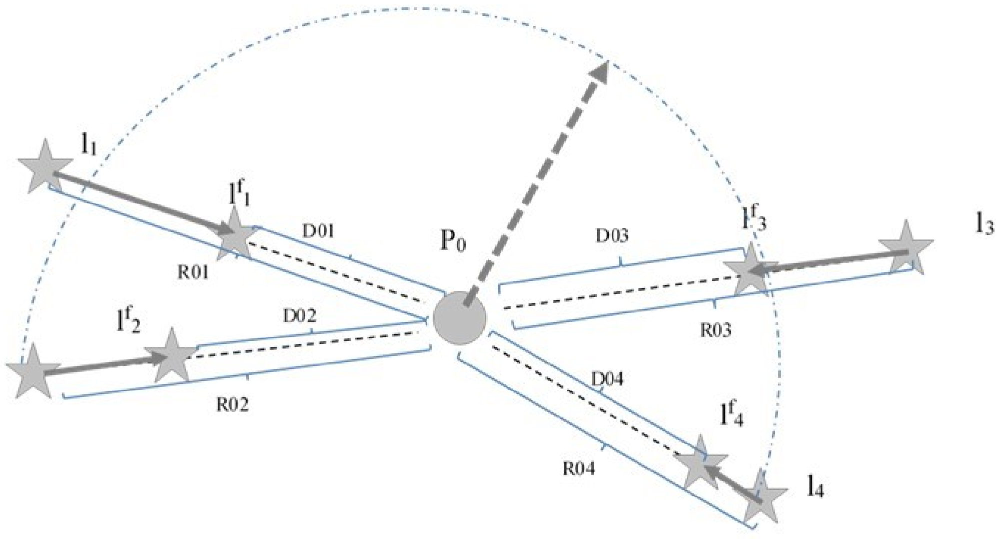


Figure 3.5: Modelling forces in mass-spring as self-organising network connecting each *WAP* to reference observations clusters.

For example, let the relaxed estimated distance of d_{ij} separate an access point w_i and cluster c_j. Based on current positions of both nodes the spring connecting them develop force F that either pushes nodes apart or pull them closer based on their graph distance r_{ij} being greater or smaller d_{ij}. However, this force, does not apply to both nodes equally. Instead, each node takes proportion of the force based on its own flexibility. In most implementations, anchor nodes reaction to such force is zero. However, in our implementation of mass-spring we allow anchor nodes to share a small proportion of these forces as long as it remains within the location covariance area calculated during clustering phase. Then, each step of our iterative network localisation process moves w_i and c_j in the direction of the force F. Finally, the process stops when total force acting on every node is small enough.

As demonstrated in Algorithm 3.1, we implemented mass-spring algorithm using single connectivity measure between *WAPs* and reference clusters. To make sure that locate the optimum point quickly and efficiently, we initialised the algorithm to use only *0.3* of force proportion when moving the nodes before recalculating stress. We also selected relatively the upper limit for force termination as *0.5*, to make sure we do not go into resource consuming loops chasing very fine accuracy, which is unlikely for indoor environment. Finally, we choose to terminate if the optimisation passes *10* attempts to minimise the tension and failed to bring the force absolute value down. Hence, we initiated incident limit to *10*.

Algorithm 3.1: ***Single Connectivity Mass-Spring Algorithm***

```
Get list of reference clusters CH
Get list of WAPs to optimise M
Sort M by tension descending
Initialize Clusters Impact as empty array

Foreach mi in M
    Initialize force proportion to 0.3
    Initialize force termination to 0.5
    Initialize incident limit to 10

    Initialize total force to NULL
Initialize incidents to 0
Initialize Positions as empty array
If Pi exists
    Set central_position to  Pi
    Set central_wieght to wi
Else
    Pi = Weighted Centroid (mi, CH)
    Set cenral_position to  Pi

While LENGTH(total force) > force termination OR total force is NULL
    Initialize Current Force as 0
    Initialize Anchors Count as 0
    Initialize Anchors Impact as empty array
    Foreach chj in CH with connection to mi
        Calculate relaxed distance DtSij from µi in cluster chj
        Calculate current distance rij between central_position and Pj
        Compute tension τij as DtSij – rij
        Compute direction vector Aij as (central_position – Pj) /rij
        Compute force vector fij as τij . Aij
        Set Central_Score as wi/(wi+wj) //where wi is the weightage assigned to mi
        Set Anchor_Score as wj/(wi+wj) //where wj is the weightage assigned to cluster chj
        Add Central_Score. fij to Current Force
        Add Anchor_Score.(-1). fij to Anchors Impact[j]
        Set Anchors Count  to Anchors Count+1
    END Foreach

    If total force is NOT NULL AND Current Force > total force
        Set incidents to incidents+1
    Set total force to Current Force/Anchor Count
    Set Central Position to Central Position + (force proportion . total force)
    Set force to LENGTH(total force)
    Add force, Central Position, Anchors Impact to Positions Array
```

```
            If incidents> incident limit
                Break
        END While

        Sort Positions by force incrementing
        Set Pi to POP (Positions -> Central Position)
        Merge POP (Positions -> Anchors Impact) to Clusters Impact
        Set Qi to COVARIANCE(Positions)
        Set Wi to DET(Qi).DSi //where DSi is the dependency score of mi
        //Start Anchors Optimisation
        Initialise AnchorsForces to empty array
        Initialise AnchorsCount to empty array

        Foreach j in Clusters Impact
            Set AnchorsForces[j]  as SUM of AnchorImpact[j]
            Set AnchorsCount[j] as COUNT of AnchorsCount[j]
            Set Pj to Pj + ( force proportion . AnchorsForces[j] / AnchorsCount[j] )
        END Foreach

END Foreach
Output CH
Output M
```

The most common implementation of mass-spring [36] treats anchor nodes with hard positions and central node with soft position. This imply that only central nodes can move according to forces applied during optimisation. We found this limitation is not suited for crowd-sourcing as errors in reference clusters would permanently limit WAPs localisation. Therefore, as per the algorithm described above, we devised the force on each spring on both directions. However, not both sides of the spring will have the same share of tension. Instead, we derived each side share through weighting. To be specific we used the same weighting described in section 3.2.1 for weighted centroid algorithm.

During the testing of mass-spring algorithm, we observed that edge cases could generate forces pushing some *WAPs* outside the boundaries. These extreme forces mostly appear due to errors in reference clusters, such as lack of coverage on one side of the building. In the following example, we set up controlled environment populated with 5 signal sources in same building and try to measure errors in locating these *WAPs* using Mass-Spring. This sample data represented the case of dispersed observations introducing errors from the far north east of the building.

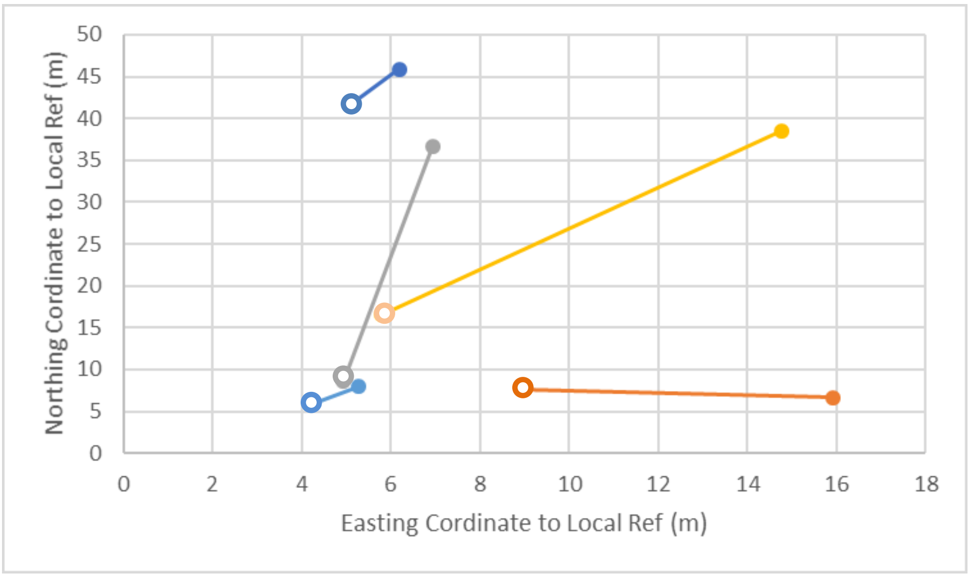


Figure 3.6: Errors in positioning signal sources in controlled environment simulating the edge case of measurements only distributed at the edge of the building. Each colour code represents one signal source with the line connecting ground truth location (hollow) to the estimated location (solid).

As can be seen from Figure 3.6, some signal sources were positioned very close to their ground truth location. In contrast, due to shifted measurements for three of the signal sources, errors in distance estimation has pushed the position out of the centroid area. To address such limitation, we have set our target to try limiting the freedom of mass-spring forces to limit the impact of outliers.

3.2.3 Single Connectivity Mass-Spring with Limited Freedom

To resolve the edge case problem that we have observed with mass-spring algorithm, we set to examine possible ways of customising mass-spring algorithm to add limited freedom on nodes. Therefore, based on each node positioning confidence, an area of freedom is defined.

To set a limit on how far each node can move when applying optimisation forces, we defined the limited freedom area for each node based on its covariance amplified by its dependency score. Hence, when a node reaches the end of its area of freedom, it becomes fixed and start enforcing most of the mass force, or tension, to the other side of the spring. To be more specific, the covariance matrix represents the confidence in node position while dependency score is directly derived from the presence of labelled data, such as *GPS*. Combing both measures guarantees that reference clusters or *WAPs* located at the edge of the building, where *GNSS* are very likely to be present, are limited in freedom and moves only around their central mean position. On the other hand, *WAPs* and reference clusters deep indoors would enjoy more freedom as both covariance and dependency score enlarged on the absence of labelled data. Therefore, this version of mass-spring is expected to guard hard references outdoors or under skylights, while produce more optimised Nodes elsewhere.

To compute the area of freedom, we simply multiply the covariance matrix Q_i with the dependency score value DS_i. The result is an amplified covariance matrix Q^f_i each of its element can be described as $Q^f(i,j) = DS.Q(i,j)$

Once we have the new Q^f_i matrix, we can then use it as positioning boundaries each time we have total force applied to node N_i. This can be presented number of standard deviations in the force vector. Therefore, an allowable force of up to twice the length of standard vector is allowed. Given all of that the decision of crossing borders is formed by equation (3.20).

$$Allowable\ force\ = \sqrt{\left(F.Q^{f^{-1}}\right).\left(F.Q^{f^{-1}}\right)^{T}}\ <\ 2 \qquad (3.20)$$

This modification is then applied to the algorithm by recalculating the force *F* every time it drives the position outside the allowable area. In order to respect the spring direction and only influence the force strength, we defined the factor of *0.7* as our scaling down factor. Hence any force vector fails the above criteria is then scaled down by multiplying its dimensions by the scaling factor. The graph below illustrates a sample case for force vector violating the freedom area boundaries rules.

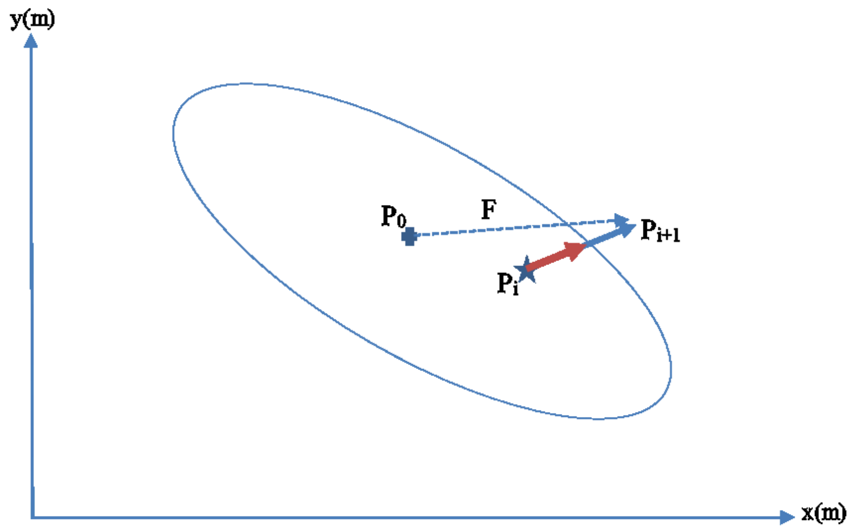


Figure 3.7: An iterative position optimisation through mass-spring. The original force vector F_i shown in Blue. The scaled down F_i to be within freedom area is shown in red.

As can be seen for the graph, force vector F is derived from the central node position before we start optimisation. Depend on how many rounds of optimisation we have already done; the node position would be affected. Therefore, we compute the vector F as $P_0 - P_{i+1}$. To clarify it further, we have modified the pseudo code to reflect this change in the algorithm. The new version is denoted as mass-spring with limited freedom (*MSLF*).

Algorithm 3.2: Single Connectivity Mass-Spring Algorithm with Limited Freedom

```
Get list of reference clusters CH
Get list of WAPs to optimise M
Sort M by tension descending
Initialize Clusters Impact as empty array

Foreach mi in M
    Initialize force proportion to 0.3
    Initialize force termination to 0.5
    Initialize incident limit to 10
    Initialize total force to NULL
    Initialize incidents to 0
    Initialize Positions as empty array
    If Pi exists
        Set central_position to  Pi
        Set central_wieght to wi
    Else
        Pi = Weighted Centroid (mi, CH)
        Set central_position to  Pi

    While LENGTH(total force) > force termination OR total force is NULL
        Initialize Current Force as 0
        Initialize Anchors Count as 0
        Initialize Anchors Impact as empty array
        Foreach chj in CH with connection to mi
            Calculate relaxed distance DtSij from µi in cluster chj
            Calculate current distance rij between central_position and Pj
            Compute tension τij as DtSij – rij
            Compute direction vector Aij as (central_position – Pj) /rij
            Compute force vector fij as τij . Aij
            Set Central_Score as wi/(wi+wj) //where wi is the weightage assigned to mi
```

```
                Set Anchor_Score as wj/(wi+wj) //where wj is the weightage assigned to cluster chj
                Add Central_Score. fij to Current Force
                Add Anchor_Score.(-1). fij to Anchors Impact[j]
                Set Anchors Count  to Anchors Count+1
            END Foreach

            If total force is NOT NULL AND Current Force > total force
                Set incidents to incidents+1
            Set total force to Current Force/Anchor Count
            Set AllowableForce to False
            While Not AllowableForce
                Set Central Position to Central Position + (force proportion . total force)
                Set F to Pi - Central Position
                Set $Q^f$ to DSi.Q
                Calculate AllowableForce as per (16)
                If AllowableForce
                    Break
                Else
                    Set total force to 0.7 Scale
                    Set Anchors Impact to 1.3 Scale
            END While
            Set force to LENGTH(total force)
            Add force, Central Position, Anchors Impact to Positions Array
            If incidents> incident limit
                Break
        END While
        Sort Positions by force incrementing
        Set Pi to POP (Positions -> Central Position)
        Merge POP (Positions -> Anchors Impact) to Clusters Impact
        Set Qi to COVARIANCE(Positions)
        Set Wi to DET(Qi).DSi //where DSi is the dependency score of mi
        //Start Anchors Optimisation
```

```
    Initialise AnchorsForces to empty array
    Initialise AnchorsCount to empty array
    Foreach j in Clusters Impact
        Set AnchorsForces[j]  as SUM of AnchorImpact[j]
        Set AnchorsCount[j] as COUNT of AnchorsCount[j]
        Set AllowableForce to False
        While Not AllowableForce
            Set $P_{j+1}$ to $P_j$ + ( force proportion . AnchorsForces[j] / AnchorsCount[j] )
            Set F to $P_j - P_{j+1}$
            Set $Q^f$ to $DS_j.Q_j$
            Calculate AllowableForce as per (16)
            If AllowableForce
                Break
            Else
                Set AnchorsForces[j] to 0.7 Scale
        END While
        Set $P_j$ to $P_{j+1}$
    END Foreach
END Foreach
Output CH
Output M
```

Based on the implementation we described above, we repeated the same test in the controlled area with 5 signal sources. The results this time, as can be seen from Figure 3.8, shows that even with dispersed and distributed measurements on the edge of the building territory, we are able to limit the freedom area of an estimated location of each signal source and bring it very close to ground truth location.

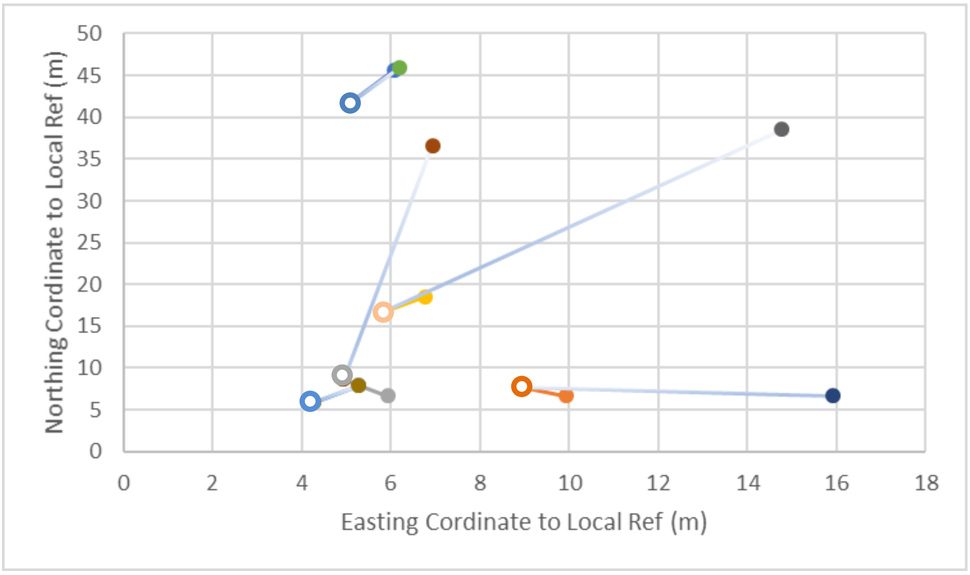


Figure 3.8: Reduction in errors while locating signal sources in controlled environment using mass-spring with limited freedom algorithm. Gradient lines represent the vanilla results. Solid lines represent new results. Coloured circles represent *WAPs*, ground truth location (hollow) and estimated location (solid).

3.2.4 Second Level Connectivity Mass-Spring:

As we continue to look for more innovative ways to improve the algorithm, we examined the possibility of adding extra springs between nodes that is not directly connected. This approach was first examined in the work presented in [56]. The thinking behind these extra springs is to help the algorithm reach more accurate global minima as total error between estimated ranging distance vector and the obtained location distance vector is usually very difficult to match. Therefore, this concept presumes that adding additional restrains between all nodes will reduce global errors measure.

For example, if node *A* connects to node *B* but not to node *C*, while node *B* connects to node *C*, an extra spring is created to estimate the connection between *A* and *C* through node *B*. This will prevent any racing condition where both nodes *A* and *C* try to influence the position of node *B* to their local minima. Adding this extra spring will affect both nodes local minima to account for this indirect relation, hence node *B* will be positioned more accurately. However, in our implementation this implies that estimation of ranging distances between nodes should go beyond the signal propagation model we used.

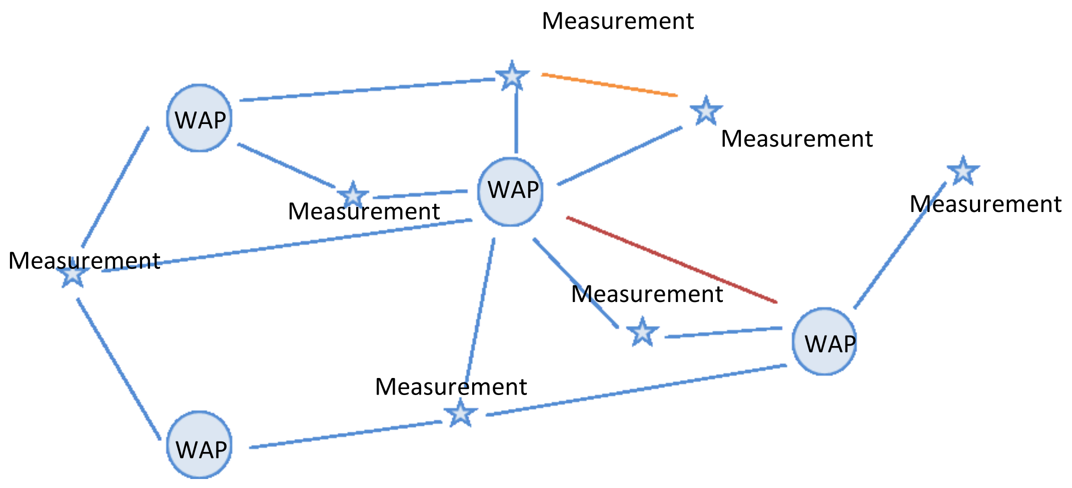
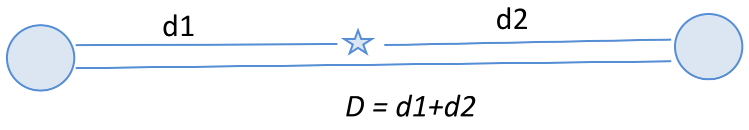


Figure 3.9: Modelling 2nd level of connectivity between *WAPs* or reference clusters as mass-spring graph by leveraging estimated and measure distances between location data and visible WAPs as links.

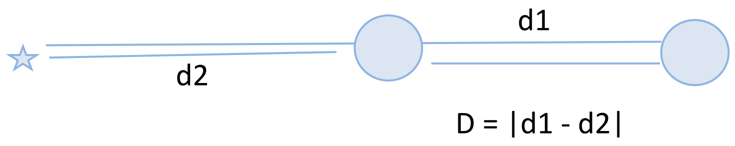
To realise the required connectivity measures between *WAPs* and reference clusters, we define two new problems: estimating *RSSI* ranging distance between two *WAPs* and estimating *RSSI* ranging distance between two clusters. However, both problems could potentially share one solution. In other words, the solution of both cases would utilise multitude of measurements to estimate the ranging distance by consuming all possible routes via the connected graph. Therefore, before we estimate the distance between pair of *WAPs* or pair of reference clusters, our proposed algorithm identifies all possible connections that reach both nodes through another node. Such connections create entries for ranging distance that is categorised as second level connectivity links.

In Figure 3.9, we show the first level connection in blue, one sample second level connection between *WAPs* in red and one sample second level connection between reference clusters in orange. To estimate the ranging distance in these two cases, we compute the upper limit and lower limit by considering the possibility of alignments of nodes. The following scenarios are evaluated of how the measuring node is located relative to the two unconnected nodes:

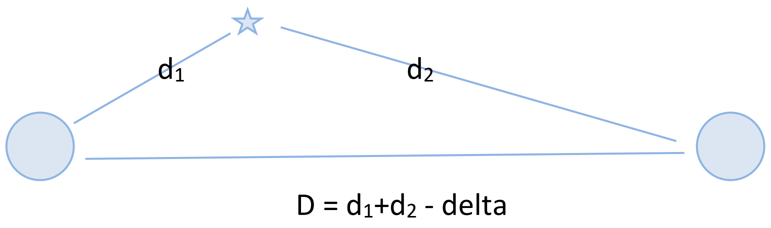
- The measuring node placed anywhere between two unconnected nodes, aligned with them.



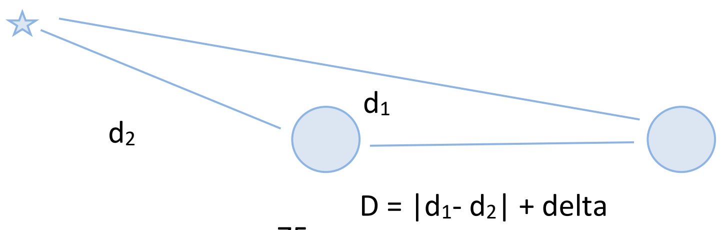
- The measuring node placed outside the bounds of two unconnected nodes, aligned with them.



- The measuring node placed anywhere between two unconnected nodes, but unaligned with them.



- The measuring node placed outside the bounds of two unconnected nodes also unaligned with them.



Considering any random mix of these four cases, we noted that distance *D* is always ranging between $|d_1 - d_2|$ and d_1+d_2. Based on this theory, we have modelled the second level connection between unconnected nodes as a range rather than value. This is more of fuzzy implementation rather than crisp input. However, it is still valid for the mass-spring algorithm. The formula below describes the range of distance *D* estimated from *N* measurement pairs of (d_1,d_2).

$$\max_{i:\,1\rightarrow N} | d_{2i} - d_{1i}| \leq \mathrm{D} \leq \min_{i:\,1\rightarrow N} (d_{2i} + d_{1i}) \tag{3.21}$$

Furthermore, as the main purpose of second connectivity estimated distance *D* is to model forces resulted from compressing or extending the springs between any given two nodes, we modified the algorithm to keep zero tension if graph distance is within the range allocated to estimated distance *D*. The updated pseudo code below provides detail learning of how we implemented such range.

Algorithm 3.3: Second Level Connectivity Mass-Spring Algorithm with Limited Freedom

```
Get list of reference clusters CH
Get list of WAPs to optimise M
Sort M by tension descending
Initialize Clusters Impact as empty array

Foreach mi in M
    Initialize force proportion to 0.3
    Initialize force termination to 0.5
    Initialize incident limit to 10
    Initialize total force to NULL
    Initialize incidents to 0
    Initialize Positions as empty array

    If Pi exists
        Set central_position to  Pi
        Set central_wieght to wi
    Else
        Pi = Weighted Centroid (mi, CH)
        Set cenral_position to  Pi

    While LENGTH(total force) > force termination OR total force is NULL
        Initialize Current Force as 0
        Initialize Anchors Count as 0
        Initialize Anchors Impact as empty array

        Foreach chj in CH with connection to mi
            Calculate relaxed distance DtSij from µi in cluster chj
            Calculate current distance rij between central_position and Pj
            Compute tension τij as DtSij – rij
            Compute direction vector Aij as (central_position – Pj) /rij
            Compute force vector fij as τij . Aij
            Set Central_Score as wi/(wi+wj) //where wi is the weightage assigned to mi
            Set Anchor_Score as wj/(wi+wj) //where wj is the weightage assigned to cluster chj
            Add Central_Score. fij to Current Force
            Add Anchor_Score.(-1). fij to Anchors Impact[j]
            Set Anchors Count to Anchors Count+1
        END Foreach

        Foreach $m_{k \neq i}$ in M  with at least one 2nd level connection to $m_i$
            Get All possible routs d1,d2 pairs as relaxed distances between $m_k$ and $m_i$
            Calculate maximum relaxed distance $D_{max}$ as MIN(d1+d2)
            Calculate minimum relaxed distance $D_{min}$ as MAX|d2-d1|
            Calculate current distance $r_{ik}$ between central_position and $m_k$

            IF ($r_{ik} \geq D_{max}$)
                Compute tension $\tau_{ik}$ as $D_{max} – r_{ik}$
                Compute direction vector $A_{ik}$ as (central_position – $P_k$) /$r_{ik}$
            ELSE IF ($r_{ik} \le D_{min}$)
                Compute tension $\tau_{ik}$ as $D_{min} - r_{ik}$
                Compute direction vector $A_{ik}$ as (central_position – $P_k$) /$r_{ik}$
            ELSE
                Set $\tau_{ik}$ to 0

            IF $\tau_{ik} \ne 0$
                Compute force vector $f_{ik}$ as $\tau_{ik} . A_{ik}$
                Set Central_Score as wi/(wi+wk) //where wi is the weightage assigned to mi
                Add Central_Score. $f_{ik}$ to Current Force
                Set Anchors Count  to Anchors Count+1
        END Foreach

        If total force is NOT NULL AND Current Force > total force
            Set incidents to incidents+1
        Set total force to Current Force/Anchor Count
        Set AllowableForce to False

        While Not AllowableForce
            Set Central Position to Central Position + (force proportion . total force)
            Set F to Pi - Central Position
            Set $Q^f$ to DSi.Qi
            Calculate AllowableForce as per (16)
            If AllowableForce
                Break
            Else
                Set total force to 0.7 Scale
                Set Anchors Impact to 1.3 Scale
        END While

        Set force to LENGTH(total force)
        Add force, Central Position, Anchors Impact to Positions Array
        If incidents> incident limit
            Break
END While

Sort Positions by force incrementing
Set Pi to POP (Positions -> Central Position)
Merge POP (Positions -> Anchors Impact) to Clusters Impact
Set Qi to COVARIANCE(Positions)
Set Wi to DET(Qi).DSi //where DSi is the dependency score of mi
//Start Anchors Optimisation
Initialise AnchorsForces to empty array
Initialise AnchorsCount to empty array
Foreach j in Clusters Impact
    Foreach $ch_{k \neq j}$ in CH  with at least one 2nd level connection to $ch_j$
        Get All possible routs d1,d2 pairs as relaxed distances between $ch_k$ and $ch_j$
        Calculate maximum relaxed distance $D_{max}$ as MIN(d1+d2)
        Calculate minimum relaxed distance $D_{min}$ as MAX|d2-d1|
        Calculate current distance $r_{jk}$ between $ch_j$ and $ch_k$
        IF ($r_{jk} \geq D_{max}$)
            Compute tension $\tau_{jk}$ as $D_{max} - r_{jk}$
        ELSE IF ($r_{jk} \leq D_{min}$)
            Compute tension $\tau_{jk}$ as $D_{min} - r_{jk}$
        ELSE
            Set $\tau_{jk}$ to 0
    IF $\tau_{jk} \neq 0$
        Compute direction vector $A_{jk}$ as $(P_j - P_k) / r_{jk}$
        Compute force vector $f_{jk}$ as $\tau_{jk} . A_{jk}$
        Set Force Score as $w_j/(w_j+w_k)$ //where $w_j$ is the weightage assigned to $ch_j$
        Add Force Score. $f_{jk}$ to AnchorImpact[j]
    Set AnchorsForces[j] as SUM of AnchorImpact[j]
    Set AnchorsCount[j] as COUNT of AnchorImpact[j]
```

```
            Set AllowableForce to False
            While Not AllowableForce
                Set $P_{j+1}$ to $P_j$ + ( force proportion . AnchorsForces[j] / AnchorsCount[j] )
                Set F to $P_j - P_{j+1}$
                Set $Q^f$ to $DS_j.Q_j$
                Calculate AllowableForce as per (16)
                If AllowableForce
                    Break
                Else
                    Set AnchorsForces[j] to 0.7 Scale
            END While
            Set $P_j$ to $P_{j+1}$
        END Foreach
END Foreach
Output CH
Output M
```

3.3 Results and discussion:

To evaluate the performance of our adaptive self-organising *WAPs* mapping based on modified mass-spring relaxation algorithm, we used the test scenarios described in chapter two of our methodology. To be specific, we performed our evaluation based on how far the mapped access points are located from the focused coverage area identified through ground truth seeds collected in each test site. This data set combine 7620 *WAPs* distributed between 10 venues. As stated before, we only had an average of 44% of detected *WAPs* qualify for such tests. The relatively low percentage can be explained as the full data set is collected over long period of time while ground truth data was just snapshot of one day.

Table 3.1: Analytical results of the 10 test venues comparing *WAPs* positioning errors for implementation of: weighted centroid (WC), mass-spring (MS) and mass-spring with limited freedom (MSLF). Also showing the effect of connectivity factor c=1 vs c=2.

Venue	**WC** [1]		**MS C=1** [36]		**MS C=2 [proposed]**		**MSLF C=1 [proposed]**		**MSLF C=2 [proposed]**	
	μ	**σ**	**μ**	**σ**	**μ**	**σ**	**μ**	**σ**	**μ**	**σ**
Alrick	16.84	4.27	14.22	3.35	10.52	1.82	9.99	1.69	9.28	1.62
HudsonBeare	10.37	2.36	9.41	2.09	8.14	1.90	8.41	1.93	8.03	1.91
Sanderson	12.63	3.05	11.05	2.58	11.07	2.51	8.64	1.86	8.68	1.86
St.James	15.62	6.32	11.50	4.18	11.23	4.04	9.16	3.16	9.70	3.42
TheCentre	11.28	3.96	10.34	3.60	9.62	3.32	8.65	3.02	7.71	2.61
Westfield,UK	13.19	5.08	11.74	4.31	9.67	3.30	9.22	3.20	8.12	2.80
Westfield,US	13.50	5.53	12.01	4.40	11.25	4.04	10.32	3.41	8.67	3.05
ECMAll	11.59	4.18	10.82	3.70	9.47	3.44	11.29	4.08	9.25	3.15
Dreamport	16.57	6.61	14.66	5.81	13.90	5.61	11.20	3.94	12.95	4.96
Aegean	13.20	4.95	11.65	4.29	9.86	3.59	9.61	3.34	8.72	3.03

On each venue separately, we compared reference trilateration implementation utilising only weighted centroid (*WC*) to estimate the position of each *WAP* and reference cluster, our implementation of mass-spring (*MS*) and the proposed modified version of mass spring with limited freedom (*MSLF*) as shown on previous section. We also tested both mass-spring implementations with connectivity factor c=1 and c=2, the second represent the latest version of mass-spring we implemented in section 3.2. Table 3.1 summarises results per venue in a form of mean error μ and the standard deviation of all errors σ.

In summary, the utilisation of multi-iterations approach using any version of mass-spring shows significant improvements compared to the common weighted centroid implementation. In addition, adding the limitation of freedom in *MSLF* version of the algorithm reduced the standard deviation indicating to more robust localisation of WAPs throughout the various iterations. Although the impact of *MSLF* on the mean seams insignificant, it is expected to reduce mobile unit positioning errors as we will demonstrate in chapter four. Finally, Table 3.1 also proves that using second level of connectivity improves the performance of mass-spring in larger venues where labelled data can't reference the majority of *WAPs*.

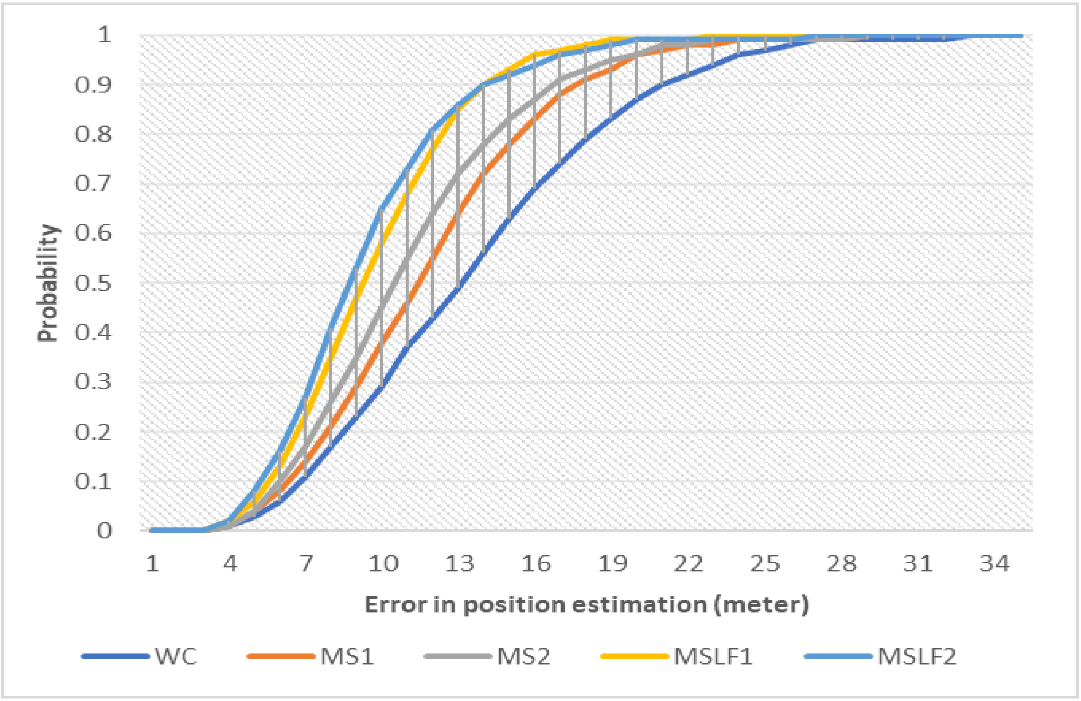


Figure 3.10: Cumulative errors probability comparison between weighted centroid (*WC*), mass-spring with *c=1* (*MS1*), mass-spring with *c=2* (*MS2*), mass-spring with limited freedom *c=1* (*MSLF1*) and mass-spring with limited freedom *c=2* (*MSLF2*)

To present the same results in another form, we combined all venues in one data set and plotted the cumulative probability distribution of *WAPs* positioning errors. Figure 3.10 confirms the results of our previous analysis and suggests that *MSLF* with $C=2$ is performing better than most other implementations. It also suggests that our research has managed to locate *WAPs* globally with an overall mean accuracy of 12.5m. On the other hand, the figure also highlights close overlap between the performance gain we are getting due to adding extra level of connectivity. This clearly suggests that restricting mass-spring to area of freedom has provided more uplifting in performance compared to adding the second level of connectivity.

3.3.1 Correlation measures:

To further analyse the correlation between number of access points, venue size and *WAPs* positioning errors, we attempted fitting linear relationship between these measures. Figure 3.11 and Figure 3.12 show the same results in a form of correlation coefficient analysis per algorithm.

In summary, we noted that by improving the performance of accurately positioning *WAPs* in each venue, the correlation factor decreases. In other words, mass-spring with first level of connectivity shows noticeable increase in errors as venue size or *WAPs* count increase. This is an expected behaviour of most iterative algorithms. Basically, the more nodes introduced in the graph the less optimised the graph would be. In the mass-spring case particularly, these extra nodes, or *WAPs*, would generate stress on attached springs and introduce more marginal errors when calculating total force. Hence it will affect the overall quality of locating *WAPs*. The same concept applies to large venues, as the larger the venue to more *WAPs* it will have. However, it is also expected that errors in meter is proportional representation of edges length in the graph.

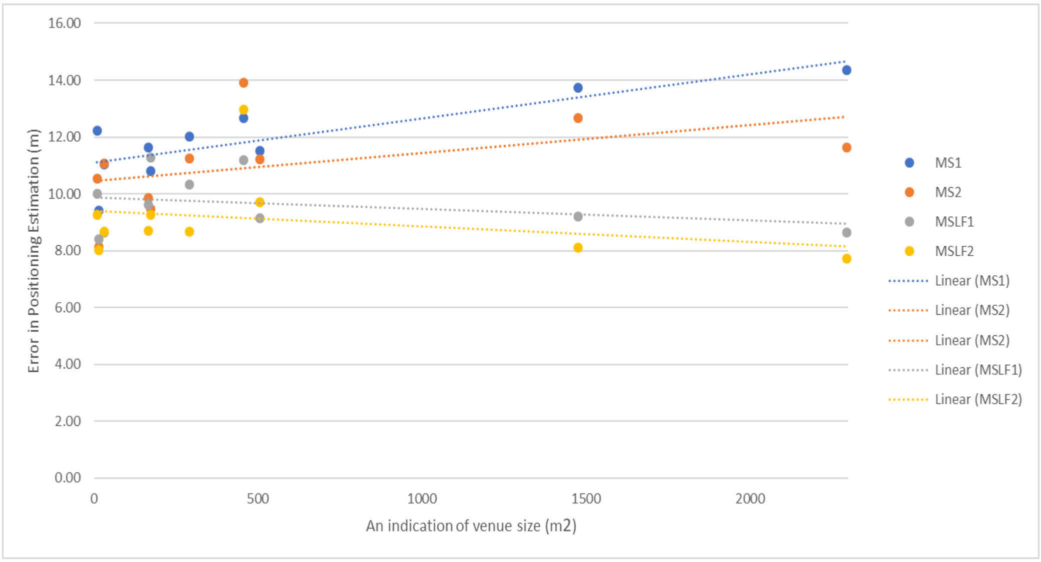


Figure 3.11: An indicative linear fitting represents correlation between number of *WiFi* access points in each venue and errors in positioning of signal sources when executing the proposed algorithms.

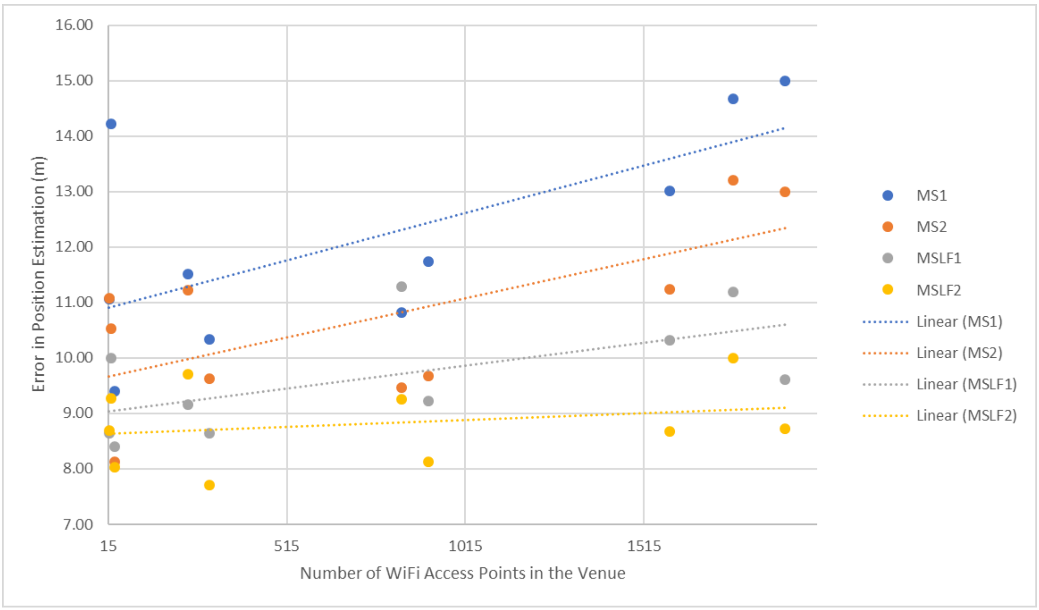


Figure 3.12: An indicative linear fitting represents correlation between number of *WiFi* access points in each venue and errors in positioning of signal sources when executing the proposed algorithms

In contrast, mass-spring with limited freedom start to get more robust showing almost the same performance everywhere by being less sensitive to venue size or

WAP count. This was one of the major drivers behind developing the limited freedom area. It certainly limits the movement of *WAPs* during iterative optimisation, keeping edges over stressed to obtain better graph fitting as global minima. Therefore, we are expecting better positioning reliability from the database as it provides better spread of *WAPs*.

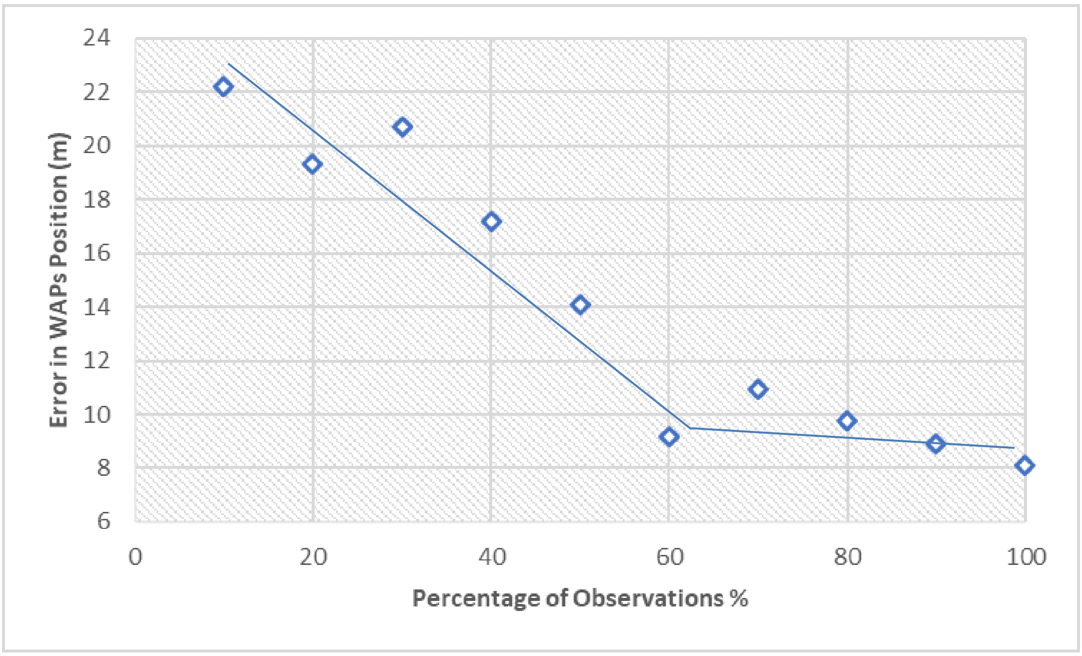


Figure 3.13: An experimental result showing the relationship between errors in *WAPs* positions and number of observations processed as percentage.

Furthermore, we set an experiment to test the effect of number of observations on locating *WAPs* accurately in our designated test venues. However, using the full data set, we measured Pearson's correlation between number of observations per *WAP* verses the estimated error for the lot of *7620 WAPs*. The results of *-0.12* correlation coefficient suggests that there is a very weak correlation between the two measures. Nevertheless, when we tried to measure the effect of reducing the amount of data streamed through the algorithm, we got different results. This time we divided all observations into tiles of *10%* each based on time. We then streamed the data in a deployment of *MSLF*, with connectivity *c=2*, and measured *WAPs* mapping accuracy after each tile. Figure 3.13 demonstrates correlation plot between growth in observations and improvements of accuracy.

However, until the tile containing over *60%* of observation, strong correlation was clearly present. After that the correlation suggests that more data would do very minor improvements. This behaviour of the algorithm should be expected. Furthermore, it is also noticeable that with more data there is always level of noise that made accuracy numbers change around specific range of errors.

3.3.2 Measuring computational overheads:

Compared to weighted centroid, someone could simply argue that mass-spring would be more computational heavy and hence costs far more when it comes to deployment. In chapter two, we justified the proposed implementation of the system as we demonstrated data structure and data flow. However, in this chapter we only focus on improving accuracy of mass-spring with different implementations. To support the implementation justification with some results, we performed an experiment to measure the computational overhead by running *20%* of all dataset into single virtual cloud server machine. The server virtual hardware specifications are shown in Table 3.2.

Table 3.2: Specifications used for server virtual machine to test the algorithm implementation.

CPU	*4 cores*
Allocated Memory	*8GB*
Storage Volume	*500GB SSD*
Connection	*1 Gbps*
Operating System	*Centos7*

To measure the efficiency of each algorithm, we estimated the time consumed by this *VM* to process the data in batches. To make our recording of time more accurate, we configured the *VM* to only allow one backend processor at any time. Each processor then writes to log file the number of *WAPs* updated and the time consumed in milliseconds. We believe this measure is the most accurate estimation of overheads as it combines *CPU*, memory and system level overheads in one metric based on time.

Furthermore, the impact of time is significant as it defines how many processors or *VMs* is required on a scale of larger deployment. Therefore, our selected metric proofs to impact the cost directly. Additionally, we made none analytical observation of system load average during the run and recorded much higher values when we run any mass-spring algorithm compared to weighted centroid. However, the variations between different versions of mass-spring implementations seams irrelevant through system load average, hence we have not included it in this study.

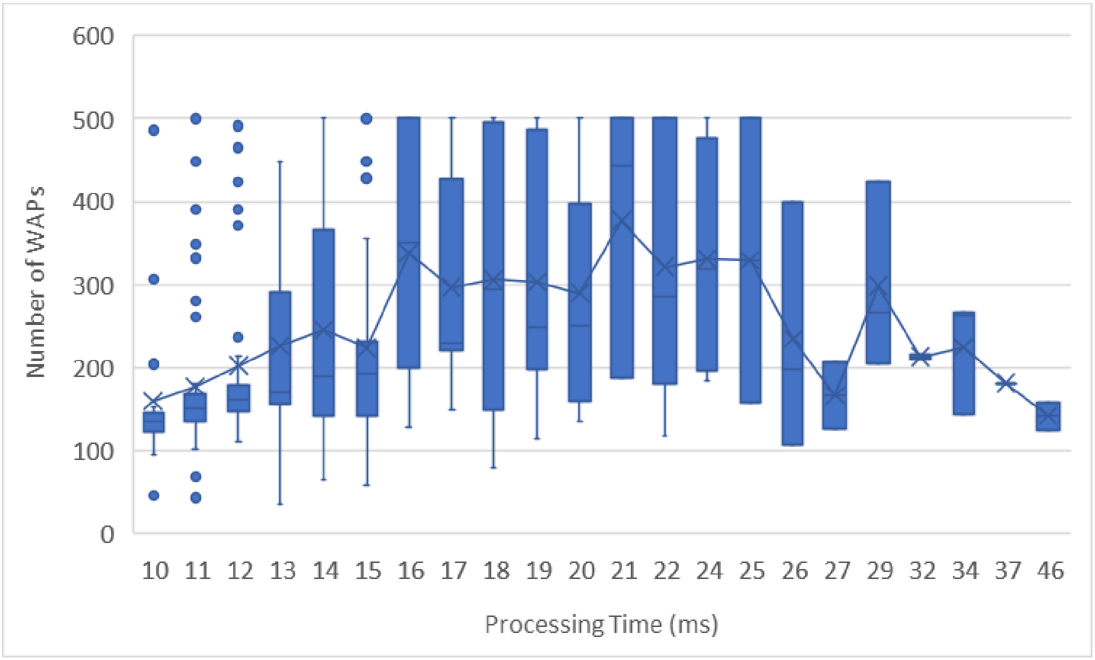


Figure 3.14: Processing speed illustrated as number of *WAPs* one virtual machine server was able to process in each batch relative to processing time in milli-seconds

The variation in Figure 3.14 suggests that an iterative optimisation might take more time even if fewer number of *WAPs* were involved. However, it also demonstrates the applicability of mass-spring algorithm implementation proving that in few milli-seconds one processor was able to optimise a batch affecting over *500 MAC* addresses. Some of the trends or outliers in the graph can be explained as a result of log generating. Basically, a marginal error of *2-3* milliseconds is expected as writing the log data to storage files is also unpredictable. Hence, we can point out to some points falling off the trend such as the entries of 27 milliseconds mark in Figure 3.14. However, the overall results in this figure clearly demonstrates that the overall trend of processing batches of *500 WAPs* only cost *15-25* milliseconds processing time of our standard testbed *VM*.

Finally, we concluded on the validity of proposed methods to expand wireless network localization beyond the roads where *GNSS* references are available. However, we also realize that the larger our network or graph of *WAPs* grows, the more computational time it will take. As a rough estimation we concluded on a model that estimates the relationship between the graph size, without *GNSS* measurements, and the computational time. In our estimation we use the factor of $10^{(n-1)}$ to render the exponential growth in processing time when the graph cover more than one parent geoindex. By a parent geoindex we refer to only two decimals of latitude and longitude coordinates, on average it covers *1km* by *1km* area. For example, if we assume that a graph covers only 1 geoindex can be optimized in T seconds, expanding the graph over n geoindices would result in $T*10^{(n-1)}$ seconds consumed in optimization. Similar model can be derived for accuracy as it degrades with the graph expanding throughout multiple parent geoindices. However, we have not attempted to estimate the base of such degradation and only considered $10*n$ for such case.

3.3.3 Prior art comparison:

As we scanned the literature, we have only identified few research projects on similar scope with published results [31], [33], [57]. These projects used smaller data set but utilized unsupervised crowd-sourcing of smart phone observations. However, the level of performance reported on their publications, still in the range *40m* error, which is mainly utilized through war-driving without much details on indoor coverage of radio-maps. Hence, these systems are not within the competitive range of accuracy for comparison. On the other hand, more recent research in this area [26], [35] only utilised limited dataset in implementation. However, we have attempted to implement and test both algorithms, adaptive least squares and multi-dimensional scaling algorithms, using only subset of our data on venue by venue basis. Starting from the University buildings, we notice that our implementation did not match the results published by the authors.

Table 3.3: Comparison with literature suggested implementation of least squares and multi-dimensional scaling algorithms

Venue	**multi-dimensional scaling algorithm [24]**		**iterative least squares [33]**	
	μ	σ	μ	σ
Alrick	*26.65*	*12.45*	*23.63*	*6.89*
Hudson Beare	*19.48*	*9.63*	*18.78*	*5.32*
Sanderson	*23.65*	*11.58*	*19.92*	*8.49*

With a further consideration of both publications, we realised that as our data is collected randomly and over long period of time. Therefore, it is not matching the data quality the authors of [26], [35] have used. Hence, we didn't progress further with the comparison.

Another unpublished system that could be more applicable, is the adaptation of *FLP* on Andriod phones that recently evolved as data collection and crowd-sourcing. Although Google did not provide any details justifying their implementation, it is still relatively the most common platform utilizing crowd-sourcing on global scale. Therefore, we included the native position measurements when we collected ground truth data in the selected test venues. Nevertheless, these native positions represent the phone position rather than the position of *WAPs*. Hence, in this comparison below, we can include reporting errors to ground truth location as there is no public access to Google *WAPs* database.

Table 3.4: An illustration of errors in native positioning system on Android phones

Venue	**Native positioning accuracy**	
	μ	σ
Alrick	*12.36*	*5.98*
Hudson Beare	*15.63*	*7.21*
Sanderson	*11.31*	*4.67*
St.James	*45.69*	*15.12*
TheCentre	*9.87*	*3.40*
Westfield,UK	*14.15*	*4.49*
Westfield,US	*11.68*	*3.97*

Table 3.4 shed some lights on how current Android phones perform. It is noticeable that some venues perform significantly better than others. Hence, we couldn't verify through these set of results that Google utilize crowd-sourcing, nevertheless, the variation in performance still suggests that two different algorithms is used based on unknown condition. Therefore, it might be suggesting that fingerprinting is utilized in some venues by owners or by Google maps team.

3.4 Conclusion

This chapter provided full study of our proposed crowd-sourcing solution to address the indoor positioning challenge in global scale. We described the details of our global grid geoindex classification and demonstrated our density-based clustering algorithm. We proposed to use Gaussian to model *RSSI* into set of probability distribution functions located per geoindex. By doing so, we achieved better representation of signal propagation on horizontal frame and explained the value of obtaining fused reference measurement of *RSSI.* Each reference measurement has been associated with location attribution derived from data clustering.

Later, we described in detail our journey through improving mass-spring relaxation algorithm and provided insights on the results and implementation of the limited freedom area feature. We have also compared the first degree of connectivity with second degree of connectivity in mass-spring algorithm. Finally, we compared the results of various scenarios and attempted to find a competitive study for benchmarking.

4 Utilisation of WAPs Database

As demonstrated in the motivation of this research, the wide spread of *WiFi* signals globally has driven most of indoor positioning solutions to utilise *WiFi* signals during position estimation. Accuracy, errors and biases of the obtained positions in any given area heavily depend on the quality of pre-trained database. Despite the technology or algorithms used to create such database, the most important measure is always driven by how accurate the system can locate mobile units or users indoors. To get an accurate position, researchers have been utilising various probabilistic or deterministic algorithms [3], [58]. However, the selection of one approach over another always depend on what kind of data attributes associated with WiFi radio map or signal sources database.

Depending on the type of data collected and mapped during database training phase, a mobile unit could use a subset of positioning algorithms. For example, a probabilistic approach would employ fingerprinting similarity algorithms to locate the best match of observed RSSI fingerprint of unknown location to specific reference point. Various similarity measures and pattern matching algorithms continue to appear in literature [28]. However, it is obvious that such techniques can’t accommodate the uncertainty in crowd-sourcing as the data has been collected from variation of devices making it subject to various noises and errors. Therefore, we rolled out this option when selecting an algorithm for evaluating our database.

Similar to an access point, any mobile unit equipped with *WiFi* receiver can also be located by trilateration. This deterministic approach estimates the mobile unit position with respect to at least three *WAPs* whose locations are known to the system through crowd-sourcing. The most simplistic deployment of trilateration used the geo-spatial centre and places the mobile node’s position in the centre of polygon its edges defined by all *WAPs* observed [59]. Another very similar solution utilised positioning centroid calculates the node’s position by averaging the locations of selected set of *WAPs* based on qualifying criteria [54]. The main disadvantage of both methods is the assumption that *WAPs* distribution is unified. In a practical scenario, it is now well known that *RSSI* measures of each *WAP* is valid indication that an access point is within proximity to the mobile device. Thus, its position should be skewed more towards access points with strong *RSSI* compared to weaker ones. To cater for outliers and variations in *RSSI* distribution that may exist among nearby access points, various weighting techniques can be added to centroid algorithm to improve positioning accuracy [25].

In previous chapter we have utilised distance-based weighting centroid algorithm as reference *WiFi* positioning system. The same algorithm was also used to estimate the initial location of a reference point or *WAP*. For such approach to work, we have described an adaptive signal propagation model based on log-distance pathloss model [23]. In this chapter we will introduce algorithms and methods we developed to estimate mobile node position utilising our radio-map of *WAPs* for the selected test areas. Furthermore, we also present an innovative way of estimating errors in such adaptive system where radio-maps constantly changing. This quality measure of positioning accuracy sets the right expectation for end user, or data consumer, when utilising location attributes. Hence it is essential for any indoor positioning system to provide an estimation of error associated with each position estimation. Finally, we present our brief experiment to add an elevation estimation where *WAPs* are located on different floors in multi-storey building.

4.1 WiFi Handover positioning algorithm

The Handover algorithm is based on dividing the mapped area into grids depending on two factors. The first factor is the initial position P_0 which can be calculated using any reference *WiFi* positioning system. Here we used the weighted centroid system as mentioned earlier. The initial position P_0 creates central node as the first node in the grid. The second factor will be the grid granularity which is presented as a cell length (*CL*). Depend on how small *CL* is, position determination will have better chance to allocate various weighting to each *WAP*.

The main aim of developing the handover algorithm is to improve the location estimation in cases were *WAPs* data base suffer from inconsistency or high level of noise. This also include biases and uneven distribution of *WAPs*. This will be achieved by determining the consistency of multiple *RSSI* measurements and amend the weighting of *WAPs* accordingly. In addition, eliminating the conflicting measurements from set of *RSSI* measurements, before calculating final position, could be considered as one benefit of this algorithm. Complimenting any *WiFi* based positioning, the Handover algorithm restrict position estimation to the cells around the central node minimising biases. In an iterative way the algorithm estimates *HoR* (Handover Ratio) as weightage to be assigned to each node on the grid.

The Handover algorithm adopt deterministic approach to detect the next central node that corresponds to changes in *RSSI* measurements, emulating the motion direction of the device without *PDR*. This determination assumes that the algorithm holds multiple readings of the surrounding *WiFi* access points estimated *RSSI*. Alternatively, we could presume that implementing this algorithm in the phone will allow continuous access to *RSSI* measurements. By measuring the delta between every two consequence *RSSI* readings in each grid cell, the Handover algorithm weights each node with *HoR* estimation. To provide a smooth and reliable location in real time, the Handover algorithm works in two stages: grid initiating and position tracking.

4.1.1 Grid initialization

In this model the initial position P_0, calculated by applying an improved version of weighted centroid [54], will be used to query list of *records* from *WAPs* database. The purpose for this list is to populate an initial grid. Hence, we require an input of three parameters to calculate our query lookup boundaries to avoid overfitting or underfitting the grid. Therefore, considering all received observations, more than one is preferred, minimum *RSSI*, denoted as *MinRSS*, and maximum *RSSI*, denoted as *MaxRSS*, regardless of mac address is recorded. As *WAPs* database entries are commonly stored as a set of *WAPs* spatial geo-tagged with location, *MinRSS* and *MaxRSS* parameters should be converted into distance to perform database query. In this section we have not utilised the adaptive pathloss model presented in Chapter3. Instead, we have used equations (4.1) and (4.2) to convert *RSSI* to distance.

$$Path\,Loss = P_T - RSS = PL(d_0) + 10n\,log_{10}\left(\frac{d}{d_0}\right) \tag{4.1}$$

$$d = d_0 \cdot 10^{\left(\frac{P_T - RSS - PL(d_0)}{10n}\right)} \tag{4.2}$$

d: transmitter-receiver separation distance in *m*

d_0: reference distance, typically *1m*

$PL(d_0)$: reference path loss at close distance to transmitter in *dBm*

PT: transmit power i.e. *-20dBm* for most *WAPs*

n: path loss exponent

RSS: received signal strength in dBm

To complete the grid size calculation, a maximum distance will be calculated from *MinRSS*. When this distance is added as a buffer around P_0, P_{max} and P_{min} would be obtained. Then by querying all *WAPs* in the database between P_{max} and P_{min}, we obtained our *WAPs* set within the grid area. Such list of *WAPs* should cover enough area surrounding the initial location to improve it further. Depending on the number of *WAPs* in the list (N) and the distribution of these *WAPs* with respect to P_{min} and

P_{max}, grid parameters can be calculated. To calculate these parameters, we defined the following parameters:

> D_{min} *as the minimum distance estimated by applying (4.2) on MinRSS.*
>
> N_{min} *as the number of WAPs in circular area limited with diameter of* D_{min}.
>
> CL *is the cell length of the proposed grid calculated as* $\frac{2 . D_{min}}{N_{min}}$
>
> D_{max} *as the maximum distance in the buffer around* P_0 *and*
>
> M *number grid cells per row calculated as* $2 . Ceil\left(\frac{D_{max}}{CL}\right)$

Once we have calculated all parameters of the grid, we could then allocate each *WAP* from the list the grid cell that corresponds to its location relative to central node located in P_0.

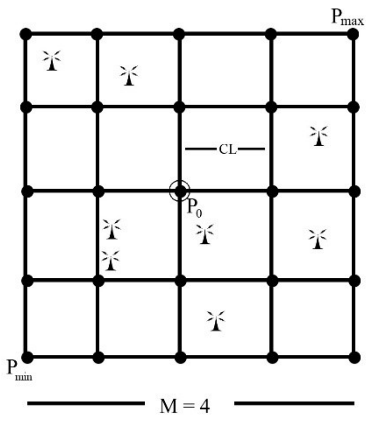


Figure 4.1: Sample grid of *4x4* showing the distribution of *WAPs* around central node

4.1.2 Position estimation

During position estimation phase, the proposed algorithm compute new estimation of mobile unit position based on scoring assigned to *WAPs* within each cell of the grid. We aim for this model to recognize reference cells that correspond better to the range of *RSSI* submitted by smartphones. We expect position estimation to be improved by handing over process to one central node each iteration. This process is repeated until P_0 remain in the same node for two iterations. The process can be described by the following steps:

- ✓ Calculate the *HoR* (Hand over Ratio) for each node in the grid.
- ✓ Compute the normalised *HoR* by adding $1+|min(HoR)|$ to all nodes.
- ✓ Weight each grid cell as per equation (4.3)

  $$wi = \frac{normalised\ (HoR_i)}{D_i} \tag{4.3}$$

  Where:

  D_i is the distance estimated as per equation (4.2) from the mean *RSSI* measurement for all *WAPs* in the grid cell number *"i"*.
- ✓ Calculate the position using weighted centroid between grid cells surrounding the node with Maximum *HoR*.

As noted from the position calculation steps above, *HoR* provide an additional weighting factor to the conventional weighted centroid algorithm. In other words, it is an indication to direction of arrival of *WiFi* signals without the requirement of smart antenna, as these are not yet available in majority of off the shelf smart phones. Therefore, we integrated *HoR* in our weightage calculation to indicate which node should take over the position calculation. The procedure to calculate the *HoR* for each node is shown in Figure 4.2. As a result of the previous *HoR* calculation for each node, we now have an index to point to the node that will be the next step Handover node. This will be used to calculate the new user position using this node information. We simply used the node with the maximum *HoR* as a Handover Node.

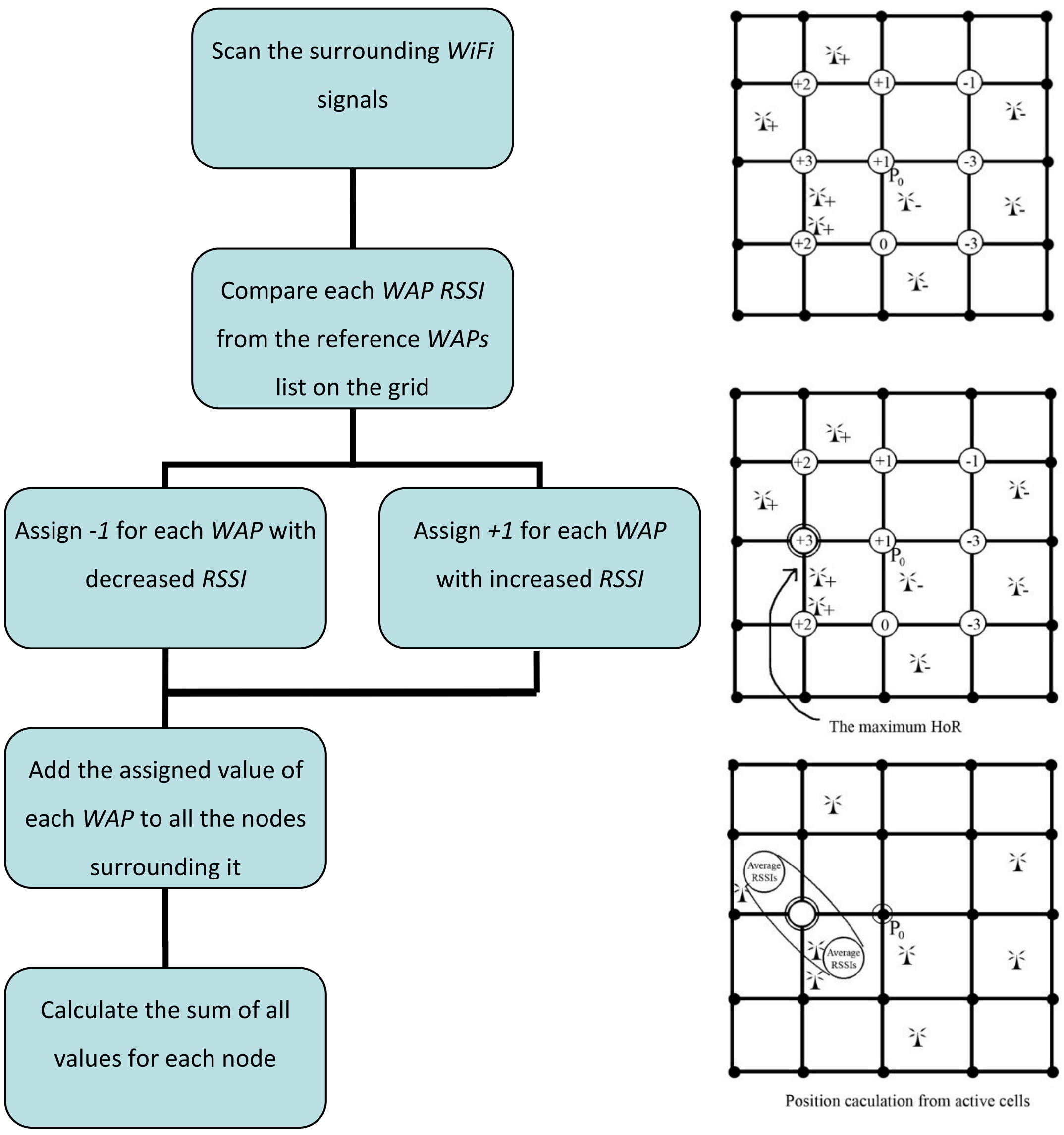

Scan the surrounding WiFi signals
Compare each WAP RSSI from the reference WAPs list on the grid
Assign -1 for each WAP with decreased RSSI
Assign +1 for each WAP with increased RSSI
Add the assigned value of each WAP to all the nodes surrounding it
Calculate the sum of all values for each node
P0
The maximum HoR
Average RSSIs
Average RSSIs
Position caculation from active cells

Figure 4.2: Modular design and sample grid demonstrating positioning estimation of mobile unit via tracking sequence

Finally, new estimation of mobile unit position is calculated by selecting only *WAPs* associated with the grid cells around the voted handover node. However, as each cell may contain many *WAPs*, we chose one per cell. Various selection criteria have been evaluated, but the most effective one was to select the one that is closer to the mean *RSSI* measurement. Once we constructed our reference *WAPs* with associated mean *RSSI* per each grid cell, we then estimate the position by applying weighted centroid. This new position is then used to recreate the grid for next iteration.

4.1.3 Experimental results and discussion

The main purpose of testing and developing positioning algorithms was to test the validity of our crowd-sourcing system and make sure that *WAPs* are reasonably located. However, as the work on Handover algorithm started before we develop mass-spring algorithm, we thought it will be good to consider both algorithms. Hence, in this section we are going to show the results comparing three means to estimate positions from the same set of test data. The first is the basic implementation of weighted centroid. Secondly, the proposed combination of weighted centroid with weightage assigned by Handover algorithm. Finally, we estimated mobile unit position using single connectivity mass-spring algorithm as described in chapter three.

Table 4.1: The compiled results for testing positioning performance in the selected test venues.

Venue	*Weighted Centroid [1]*		*Handover [Proposed]*		*mass-Spring [Proposed]*	
	μ	*σ*	*μ*	*σ*	*μ*	*σ*
Alrick	*19.23*	*6.06*	*9.53*	*0.62*	*13.19*	*2.93*
Hudson Beare	*14.51*	*4.14*	*13.96*	*3.81*	*12.92*	*3.49*
Sanderson	*10.61*	*2.27*	*14.28*	*3.91*	*10.55*	*2.42*
St.James	*21.28*	*9.21*	*14.08*	*5.61*	*11.71*	*4.28*
TheCentre	*18.29*	*7.57*	*14.93*	*6.05*	*12.62*	*4.6*
Westfield,UK	*20.28*	*8.62*	*16.38*	*6.47*	*13.18*	*5.08*
Westfield,US	*17.03*	*6.76*	*14.15*	*5.49*	*14.82*	*5.95*
ECMAll	*15.86*	*6.25*	*14.47*	*5.84*	*13.51*	*5.12*
Dreamport	*28.41*	*13.07*	*21.11*	*9.31*	*18.35*	*7.78*
Aegean	*15.86*	*6.32*	*12.68*	*4.82*	*11.15*	*4.29*

To analyse the performance of these algorithms, we use the method described in chapter two performing known smart phone tests using the same test venues. We then compile the results in a form of mean error and standard deviation to identify the position quality as well as the robustness. In general, we notice that the results vary between venues and that can be explained due to the differences in crowd-sourcing, size, WAP count and layout. However, as the main purpose was to verify the quality of WAPs database, we believe that this analysis is satisfactory. The results are demonstrated in Table 4.1.

As can be seen from the highlighted best results obtained per venue, mass-spring seems to outperform the proposed Handover algorithm in most cases. It is also clear that the work we did on handover had improved the results over the implementation of traditional weighted centroid without handover. Nevertheless, there is still a possibility to test the scenario where we combine handover with mass-spring to test if it can drive the results closer to the demanded 10m error that usually is targeted in crowd-sourcing systems.

4.2 Modelling estimating positioning error

The ability to assess the quality of estimated positions would benefit any system regardless of the technology behind it. This will assure that correct expectation is set to end user. Nevertheless, it is very crucial to assess such information accurately when it is proposed to be used as feedback to the same system. Previous research employs analytical models to assess positioning errors for *WiFi* positioning systems [60], [61]. Such models are very valuable for measuring or understanding the expected position errors for a specific positioning system or method. Hence, researchers frequently refer to them while analysing the general impact of signal transmitters distribution, density of sampling, grid size in fingerprinting, number of detected WAPs, signal characteristics and signal propagation.

On the other hand, these models suffer from two major limitations. Firstly, it expects an input from training data to create the reference analytical model. However, autonomous systems should be able to estimate its errors without previous training and preloaded models. The second limitation is that these models didn’t count for dynamic changes and the validity of radio-maps. As radio-maps might reach maturity with autonomous system deployment to be used as training data, it will only be valid for specific period of time as more data continue to merge in and dynamically change radio-maps. Therefore, static analytical models do not fit within the purpose of this research.

More advanced positioning algorithms appeared in literature have focused on modelling errors during position estimation in real time rather than limiting error estimation to radio-maps. Such models estimate uncertainty in positioning based on correlation measure between sections of data used in online phase. However, the main difference compared to analytical models is that the first try to understand the relationship between system parameters and positioning quality while the second only uses information from position determination algorithm. In one example, [27] presented a method that uses a regression technique based on signal strength and compare it with ground truth model created from training data. However, as the training data cannot be used as ground truth for an autonomous system, such model cannot serve the purpose of this research.

More *WiFi* specific models were also presented in [62]. The authors presented four different error estimation models to estimate the quality of fingerprinting samples and measure its effect on positioning experience. Their first model “Fingerprint clustering” measures the similarity among the samples and cluster all neighbouring fingerprints with high similarity into one reference. It then measures the estimated error by the cluster size. The second model is “*Leave out Fingerprint*”. It works to create a static error map by recalculating location of each fingerprint sample using all other fingerprints in the building. The third one is “*Best candidate set*”. In this model the author proposed to use *k* best matching fingerprints and compute distance between each pair then return the maximum distance as an estimation of error. The last model was “*Signal Strength variance*”. This model calculates the variance in signal strength per *WAP* then estimate the error as an average of variance of all *WAPs* in dataset.

Although the above models promise an accurate estimation of errors, they heavily rely on the quality of calibration data in all calculations. This requirement is a major limitation for crowd-sourcing systems as radio map are constructed with variation in quality of input data the system may receive from mobile devices. Also, as the radio map changes constantly, such models will not be able to measure aging factor. For comparison purposes, we have implemented the clustering and best candidate set models into our adaptive system, as they are expected to provide best results according to the author [62].

4.2.1 Modelling accumulative errors

Our proposed model is more suited for the way our radio-maps and *WAPs* database are constructed in an autonomous system. As per the details in previous chapters, crowd-sourcing systems rely on data streams from different quality and use cases to estimate signal characteristics in each geoindex of the radio-map. To be specific, *GPS* data proofed to be the main source of location when crowd-sourcing radio-maps in new territories. However, trusting that we always have reasonable quality *GPS* when we deploy crowd-sourcing systems on large scale is not realistic. Even outdoors, *GPS* performance could deteriorate as the users move closer to buildings and start losing line of sight with the sky. As demonstrated, adaptive optimisation algorithms were employed to extend radio-maps recursively to cover as much of indoor territories as possible. Nevertheless, quality measures of initial *GPS* location data and the consistency of all *GPS* enriched reference clusters are key parameters in our proposed model.

Taking a step back to describe our data models, we denote each radio-map as number of nodes each represents an area roughly close to *10x10* meters. Let's assume a radio map with K nodes populated by signal measurements from M different *WiFi* transmitters (*WAPs*). Each node N_k represented by collection of Gaussian probability distribution functions $PDF(\mu_i,\sigma_i)$ modelling all observations of each *WAP* W_m observed within this node. Each time new set of observations for W_m reported within the area covered by N_k, the probability distribution function of transmitter W_m in node N_k would be updated to $PDF(\mu_{i+1},\sigma_{i+1})$. Obviously, the error ε_m concerning the uncertainty of W_m location on reference to K nodes, each with *PDF*, can be calculated as in (4.4).

$$\varepsilon_m = \frac{1}{K} * \sqrt[q]{\sum_k \left| D(N_i, W_m) - \ddot{D}(\mu_i, \sigma_i) \right|^q} \qquad (4.4)$$

$D(N_i, W_m)$: is the function that computes distance between node i and *WAP* m.

$\ddot{D}(\mu_i, \sigma_i)$: is the function the calculates log-distance based on *RSSI* in node i .

In addition to the uncertainty *WAP* location data, each node also associated with quality metric $Q_k(t)$ represents the uncertainty in location data from all observations fused in node k until time t. Similar to quality assigned to each node when we estimated weightage for position calculation, $Q_k(t)$ can be estimated as the determent of covariance matrix multiplied by dependency score of node k.

During the positioning phase, if the mobile detected *L WAPs*, *L* radio maps will be fetched from the database. But only n matching radio maps are used to calculate user location. Then positioning error (*Err*) would be estimated as in (4.5).

$$Err(t) = \frac{1}{n} \sum_{k=1}^{n} [\varepsilon_k(t).\ [\sum_{i=1}^{K} Q_i(t). P_i(t)] + V_k] \qquad (4.5)$$

$P_i(t)$: is the probability of observing *RSSI* of *WAP* k on node i at time t, where $\sum_{i=1}^{K} P_i(t) = 1$

V_k: is average error of the radio map initiating vector, mean value of *GPS* reported errors in radio map k.

4.2.2 Data analysis and test results:

To evaluate the feasibility of our proposed model, we run tests in three different shopping centres around the city: *Site1(250m x 235m), Site2(287m x 110m)* and *Site3(198m x 107m)*. Then we recorded the ground truth location P_0 every *5* meters and collected estimation of position $P`$ from the Handover algorithm associated with estimated error calculated as per equation (4.5). The absolute measure of difference between ground truth error, calculate as line of sight distance between $P_0$ and $P`$, and the estimated error Err for every location $\vec{l}(x, y)$ in test points L is then used as quality metric as per equation (4.6).

$$Dif(l) = \left| \sqrt{\left(\dot{P}_l(x) - P`_l(x)\right)^2 + \left(\dot{P}_l(y) - P`_l(y)\right)^2} - Err(l) \right| \qquad (4.6)$$

$\dot{P}$: is the ground truth position.

$P`$: is the estimated position using the proposed handover algorithm.

Err: is the estimated error as per equation (4.5)

Table 4.2: Test results showing accumulative errors for the proposed error estimation algorithm

Dif stats	*Site1 (1650 WAP reported) (220 Test Point)*	*Site2 (580 WAP reported) (130 test Point)*	*Site3 (221 WAP reported) (80 test Point)*
Average	*8.91*	*6.54*	*5.82*
Maximum	*26.12*	*21.05*	*19.41*
Minimum	*0.92*	*0.41*	*0.21*

As can be seen from Table 4.2, the results show an average fit of error estimation compared to ground truth. In some cases, we have managed to closely match the errors as the positioning algorithm acquire more data building better estimation of

WAPs locations. However, it is also visible that outliers still exist as the maximum errors on proportional bases to the venue size or total number of access points.

On the other hand, running the same test set through the models presented in [62], showed a rise in average differentials values due to inaccurate references treated on the same as any grid point during the test. However, we noted that minimum differentials values are much better. This was due to our model over estimating errors in reasonably accurate radio maps in some areas. Table 4.3 and Table 4.4 show the results for the clustering and best candidate models on sequence.

Table 4.3: Test results showing accumulative errors for the *Clustering Model* algorithm [62]

Dif stats	***Site1***	***Site2***	***Site3***
Average	*31.57*	*18.94*	*24.36*
Maximum	*92.34*	*36.51*	*51.79*
Minimum	*2.21*	*3.68*	*1.46*

Table 4.4: Test results showing accumulative errors for *Best Candidate Model* algorithm [62]

Dif stats	***Site1***	***Site2***	***Site3***
Average	*36.14*	*16.45*	*23.81*
Maximum	*87.98*	*28.96*	*59.36*
Minimum	*2.02*	*1.14*	*2.34*

4.3 Floor determination in multi-story buildings

While indoor positioning seems to be developing very quickly, floor determination in 3D frame is still challenging topic in research. In multi-story buildings the two-dimensional position is only one portion of location data attributes. Also, many location-based services, such as safety and emergency call location tagging, have assigned floor determination more attention due to the efforts required to search multiple floors when any incident is reported. Furthermore, the recent interest in measuring human traffic in commercial buildings, is not feasible without some level of floor determination. All the above reasons have encouraged us to investigate the possibility of using database of wireless signal sources to fill this gap.

To date indoor positioning solutions for floor determination have been mainly based on either Fingerprinting [61], [62], [65] or installed beacons [66]. However, the authors in [65] have shown the feasibility of using *WiFi RSSI* values for floor determination. In their research they have developed a *WiFi* fingerprinting system to work with multi-story building. Compared to other fingerprinting systems the mentioned system requires less intensive sampling points in the calibration phase as it utilise linear regression. On the other hand, all above solutions still suffer from the common fingerprinting limitations. The first one is that the solution will not be able to accommodate any changes in the *WiFi* infrastructure and will require a complete recalibration. Secondly, the solution is dependent on the quality of the calibration. Therefore, an intensive training phase will essential for the system to work.

In this section we present the research we conducted to develop *WiFi* based indoor positioning algorithms utilising reference database of signal sources, instead of fingerprints. We belief that such deployment would save time and cost. Furthermore, in this research we have only tested using available *WiFi* signals for floor determination. We argued that floor determination is a standalone process that should be conducted separately from the usual positioning for it to provide better accuracy. Overall, we aim to enable an indoor positioning system to work with off-the-shelf components. Hence, we did not consider any additional requirements other than access to *WAPs* database and mobile devices equipped with *WiFi* receiver. Moreover, unlike fingerprinting we have designed the proposed algorithms particularly to minimise calibration using reference database parameters.

4.3.1 Research contribution

In this research we have designed two different models for using labelled *WiFi* signal sources to determine the floor number in multi-story building. The first model is "*The Nearest Floor Algorithm*" which is a simplified solution of *KNN* used in commonly fingerprinting [67]. The second one is our novel statistical model "The Group Variance Algorithm". This new model groups the detected *WAPs* per floor based on their label and compare statistical features of each group to find the best match floor number. Each model assumes that a reference *WAPs* database, associating every *WiFi* Access point with its floor number is available. Such database could be obtained from venue owners or *IT* team if the solution is deployed on limited scale. However, we argue that with the recent development on smart phone sensors, our *WAPs* database will be able to accommodate crowd-sourcing of this extra label in near future.

4.3.2 The nearest floor algorithm:

This algorithm has been developed to simplify the well-known *KNN* fingerprinting algorithm "*K Nearest Neighbour*". *KNN* algorithm is usually implemented as a supervised classification method where positioning is obtained based on finding the nearest *k* neighbours in pre-trained references database. The main part of this algorithm is the training samples. The training samples should be collected intensively during the calibration process of the area of interest. Usually, each record of the training samples will contain a reference ground truth position along with *WiFi* observations. Such data are then stored either individually or in clusters to enable *KNN* in online phase. For examples, authors in [68] generates clusters per area, including floors clusters. They have also evaluated various algorithms to perform clustering and obtain more distinct fingerprints per area.

Given a new observation reported from the unknown position during online phase, *KNN* works to identify the best *k* candidate clusters in training data for position estimation. The selection of best candidates is based on distance function that employs *WiFi* similarity between the online measurements and the collected training data [28]. Nevertheless, the main factor in tuning *KNN* performance is usually selecting value for *k*. It is very common that *k* is related to density of training database and therefore, could be a system variable on its own.

4.3.3 How we approach KNN in this research?

As we mentioned earlier the nearest floor algorithm has been designed to select the nearest reference *WAPs* to decide the floor number. The key difference between our designed algorithm and *KNN* is that we do not require training data or clusters on fingerprinting. While *KNN* works after an intensive calibration, our algorithm only maintains *WAPs* as references. We consider this as research contribution for few reasons. Firstly, the effect of minor infrastructure changes on *WAPs* as references is very minimal, while fingerprinting clusters could suffer from significant inaccuracies for the same level of changes. Secondly, data bandwidth and local storage required to transfer or store the reference data is very light compared to fingerprinting.

To maintain compact and searchable data structure, with ability to extend to global coverage, we have designed *WAPs* reference data structure specifically for floor estimation. The proposed data structure holds only one entry for each *WAP*, so mac address can be used as database key for distributing and searching the data at any scale. Also, we assume that only *WAPs* with enough accuracy and maturity, or any equivalent quality indicators, will be added to floors data structure. This is the main reason for keeping this structure separated from the main global database. Figure 4.3 shows the structure of the database which has been designed for floor determination only.

MAC Address	***Weight***	***Floor Number***	***MaxRSS***

Figure 4.3: Reference WAPs database structure used to enable floor estimation

Even for none fingerprinting system, maintaining the reference *WAPs* data is essential to support a dynamic and adaptive solution. Therefore, we have implemented the system to simultaneously update *WAPs* records in the database to keep track of maximum reported *RSSI* for each *WAP*, recorded in *MaxRSS* field in the database. This field is then used during any floor estimation to eliminate the effect of variation in transmitters power levels and various receiver's manufacturers. Hence, we employed a procedure for selecting the best reference *WAPs* based on the difference between *MaxRSS* and the reported *RSSI* during the online phase. Similar to *KNN* algorithm we then pick up *WAPs* that have the smallest *RSSI* distance compared to each observation in online phase.

In this research we have used $k=3$ as a guide for matching the floor number with the reference database. The selection of this value was due to the nature of floor selection. Basically, to resolve any conflict when the selected *WAPs* disagree on best matching floor, we would need more than two *WAPs*. Choosing only one *WAP*, as per $k=1$, would present risk of selecting the wrong floor based on outlier. However, any number over three would also start to propagate an effect of weak signals into the decision-making process.

Below we present the pseudo-code for the proposed nearest floor algorithm implementation.

Algorithm 4.1: Detailed implementation of nearest floor algorithm in a form of pseudo-code

```
Input WAPs list from observation data
Set k to 3
SET available_waps to empty list
Query reference database to get Ref_WAPs
Foreach $m_i$ in WAPs list
    IF  Ref_WAPs[$m_i$] exists
            Set  $m_i$[rank] to Ref_WAPs[$m_i$] [maxrssi] - $m_i$[rssi]
            Set $m_i$[floor] to Ref_WAPs[$m_i$] [floor]
            Add  $m_i$ to available_waps
      END IF
END FOR
SORT available_waps by rank value ascending
SET Floors to empty list
SET i to 0
FOR i=1 to k
     Set top_record to POP(available_waps)
     Set floor_estimate to top_record[floor]
     IF Floors [floor_estimate ] exists
          Increment Floors [floor_estimate ] by 1
     ELSE
          Set Floors [floor_estimate ] to 1
    END IF
Sort Floors by count Descending
Set floor_keys to KEYS(Floors)
Set Estimate1 to POP(floors)
Set Estimate2 to POP(floors)
IF (Estimate1 equals Estimate2) AND |floor_keys[1] – floor_keys[2]|>1
         RETURN Ceiling((Estimate1+Estimate2)/2)
ELSE
        RETURN Estimate1
```

Someone might see *KNN* implementation limited to vertical distribution of the *WiFi* access points. Our initial tests in University buildings were subject to distribution of access points in a vertically aligned uniform. However, in further tests, shown in results section, we did verify that the algorithm does not require any vertical alignment, but it will benefit from more spread distributing the access points horizontally in each floor to maintain a strong *WiFi* coverage.

4.3.4 Group Variance Algorithm

Some of the observations that we had while testing the nearest floor algorithm have brought to our attention the need for additional statistical parameters. For example, during our walk near an elevation transition point, we notice signal strength variation between floors became randomly favouring one floor over another. In addition, areas where all *WiFi RSSI* values are weak enough, finding a distinction between floors only based on strongest signal is not utilising all potential information made available to the phone. Even more, we have also recorded variation of *RSSI* from one side of the building to another and between rooms. Therefore, we concluded that more statistical parameters are required to assist or replace *MaxRSS*.

Looking into various statistical parameters, we have selected the range, the variance and the availability. Those three parameters will use the *WiFi RSSI* readings to provide an indication of the floor number which the user is in. The novelty of the group variance algorithm comes by considering the distribution of the *RSSI* values in each floor rather than the usual distance measurement as discussed earlier.

4.3.5 How does the group variance algorithm work?

We configured the mobile device to make request for floor determination by collecting observation over time window of *10* seconds. The system will then use the reference *WAPs* database to assign floor number to each mac address reported. The algorithm starts by grouping, or clustering, list of mac addresses by the assigned floor number, to apply the selected statistical models on each floor separately. As we mentioned earlier this model consist of three parameters: range, variance and availability. The variance S^2, shown in (4.8), is representative of variation of *RSSI* values in each floor. The range R and availability $A^{\%}$ are shown in equations (4.9) and (4.10) respectively.

$$\overline{RSS}(x) = \frac{1}{N} \cdot \sum_{i=1}^{N} RSSI_i \tag{4.7}$$

$$S^2(x) = \frac{1}{N-1} \cdot \sum_{i=1}^{N} \{(RSSI\}_i - \overline{RSS}) \tag{4.8}$$

$$R(x) = \max_{i<N}(RSSI_i) - \min_{i<N}(RSSI_i) \tag{4.9}$$

$$A^{\%}(x) = \frac{N}{M} \cdot 100 \tag{4.10}$$

N is the number of distinct mac addresses seen for floor x.

M is the number of mac addresses recorded in the reference *WAPs* database for floor *x*.

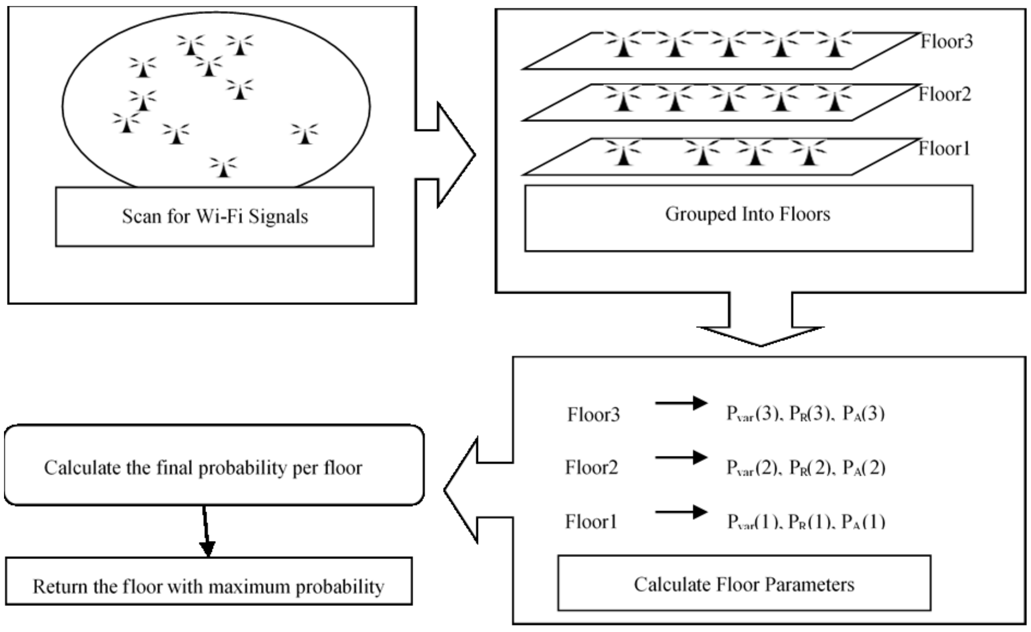

Scan for Wi-Fi Signals
Floor3
Floor2
Floor1
Grouped Into Floors
Floor3 → Pvar(3), PR(3), PA(3)
Floor2 → Pvar(2), PR(2), PA(2)
Floor1 → Pvar(1), PR(1), PA(1)
Calculate Floor Parameters
Calculate the final probability per floor
Return the floor with maximum probability

Figure 4.4: Group Variance Algorithm Explained in Step by Step Block Diagram

The values of these three parameters will present *WiFi* signals distribution pattern for each floor. This should enable us to estimate which floor we are on. Depending on the structure and the building materials used in multi-story buildings, *WiFi* signals will never spread equally in all directions. Therefore, in such buildings the horizontal and vertical signals distributions will certainly be different. This encouraged us to examine floor determination based on the selected *RSSI* statistical features. The estimation of floor number works by selecting the floor that maximise values for variance, range and availability. However, we have realised that the three parameters do not always indicate to the same floor. Therefore, we proposed adding normalisation stage to convert each parameter estimation into probability ranging between *0* and *1*.

Once the three probabilities are calculated a weightage value is assigned to each parameter to combine the three values in simplistic floor voting system. The equations below demonstrate the process of obtaining the final probability *P(f),* where *f* is a given floor number, by combining the probabilities of these three parameters. This is of course only possible under the assumption that these three parameters are independent in their probability distribution.

$$P_{total}(f_i) = \propto . P_{var}(f_i) + \beta .\ P_R(f_i) + \gamma .\ P_C(f_i) \tag{4.11}$$

$$P_{parameter}(f_i) = \frac{\sum_{F=f_i} parameter}{\sum_{All\ Floors} parameter} \tag{4.12}$$

To select a weightage for each parameter, we randomly picked *20%* of our labelled reference data, keeping the rest for testing, aiming to measure the significance of each parameter. Using each floor data, we then calculated the coefficient factor for correct floor estimation of each parameter separately. Then the weight value for each parameter was calculated as the percentage of the sum of values with correct estimations to total values. Basically, if any parameter would always indicate to the correct floor, it will get the weightage of one. After normalisation of three weightage values, we updated the total probability equation as per the following:

$$P_{total}(f_i) = 0.41 . P_{var}(f_i) + 0.36 .\ P_R(f_i) + 0.23 .\ P_C(f_i) \tag{4.13}$$

These figures clearly give priority to the variance if the three parameters estimate three different floors. However, it is still favouring any two parameters when they agree on given floor.

4.3.6 Combined solution:

Based on our initial results [69] we produce for Arlick building in the university, further research was required to examine the possibility of combining the two algorithms into one floor estimation solution. Therefore, we set to test such scenario by retraining the model in equation (4.13) to include probability measure based on the percentage of *WAPs* with strongest *RSSI* each floor claim. To start we determined an *RSSI* cut off value that offer the maximum probability to correspond to the correct floor. Therefore, we used the same training data set and plotted the probability distribution of *RSSI* to floor estimation. The graph below demonstrates this analysis.

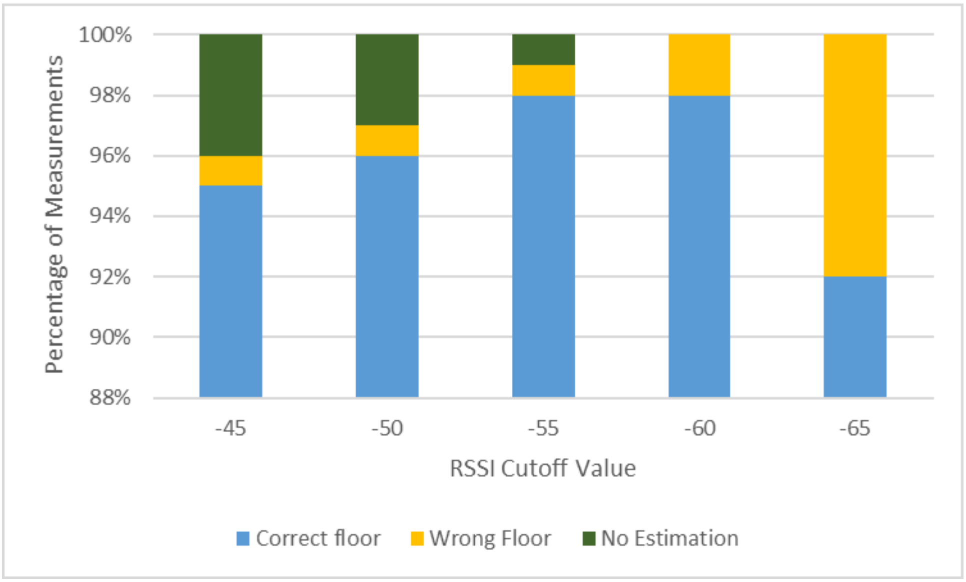


Figure 4.5: Analysis of correct and incorrect floor estimation based on *RSSI* cut off number.

The Figure 4.5 shows that using *WAPs* with *RSSI* stronger than *-55* or *-60 dbm* provides the best percentage of correct estimation using this modified version of *KNN*. Considering that the no estimation is better than wrong estimation, we chose *-55dbm* to select input data for *KNN*. This means only *WAPs* that has been detected with *RSSI* ≥ *-55* will be used to compute P_{knn} as a function of each candidate floor as per equation (4.14). To conclude, our new trained model for the combined solution is set as per the details in equation (4.15).

$$P_{knn}(f_i) = 1 - \frac{\sum_{F=f_i}(RSSI_{max} - RSSI_{reported})}{\sum_{All\ Floors}(RSSI_{max} - RSSI_{reported})} \qquad (4.14)$$

$$P_{total}(f_i) = 0.31.P_{var}(f_i) + 0.15.\ P_R(f_i) + 0.09.\ P_C(f_i) + 0.46\,.P_{knn}(f_i) \qquad (4.15)$$

Furthermore, to reduce false floor determination is edge cases where two floors show very close probability, we added a validation check before confirming to floor f_i. The proposed validation criteria measure the significance of top two candidate floors as per the following:

$$\text{Significance}(f_i) \quad = \quad \frac{P_{total}(fi)}{\sum_{Top\ two}\big(P_{total}(f)\big)} \tag{4.16}$$

If the floor didn't score $Significance > 0.6$, we do not confirm the device to any floor.

4.3.7 Test results and discussion

Unlike all previous sections, in this case we didn't use *WAPs* database constructed by the crowd-sourcing observations due to the absence of *3D* attributes in it. Instead, we created another copy of reference *WAPs* database as per the structure in Figure 4.3. To obtain floor number assigned to each *WAP*, we just reversed the *KNN* algorithm to run per *WAP* rather than per observation. The reversed KNN elected the floor number that claim higher percentage of observations with *RSSI>-55dbm* considering all observations assigned to any given *WAP*. such arrangements are temporary as with the latest generations of smart phones equipped with pressure sensors, the crowd-sourcing algorithms in Chapter3 would be able to generate *3D* labelled *WAPs*. In the meantime, we justified that reversed *KNN* provides accurate labelled *WAPs* data, when its input is labelled to the correct floor, by comparing the results of *24* mac addresses located in university buildings that we could verify from *IT* team. Hence, we used the reverse *KNN* on all test venues considering that any error in labelled *WAPs* is neglectable.

The table below provides an overview of test venues we used to compare floor determination algorithm proposed in this section. All buildings are in Edinburgh, UK and were accessible to us during data collection. The data is collected using variations of smart phones with different brands (Nokia, HTC, LG and Motorola). *WiFi* observations are recorded by running "*WiFi Stumbler*" application for dedicated time slot per floor while manually recording floor number and time in separate file. Tests are then performed by running log files through code scripts on server *VM* to generate labelled *WAPs* initially then output floor estimation per observations. Floor labels in each observation are then used to test each estimation and add line per observation in results log file. Each line provides the comparison result indicating if the estimated floor were correct, false or not available.

Table 4.5: An overview of test venues used to validate the proposed algorithms for floor determination algorithms

VENUE	FLOORS TESTED	LABELLED OBSERVATIONS	TOTOAL FLOORS	LABELLED WAPS
ALRICK BUILDING, UK	2	186	6	**12**
FARADY BUILDING, UK	3	154	4	**12**
ST.JAMES, UK	5	547	6	**62**
OCEAN TERMINAL, UK	3	432	3	**167**

To compare the performance of proposed algorithms, we performed three separate runs of same test data and recorded results for *KNN* algorithm, Group Variance algorithm and the combined algorithm. The percentage comparisons of these results are compiled in Table 4.6.

Table 4.6: Compiled test results comparing *KNN* algorithm with Group Variance algorithms and the Combined Probability algorithm

VENUE	*KNN*			*GROUP VARIANCE*			*COMBINED*		
	True	*False*	*NA*	*True*	*False*	*NA*	*True*	*False*	*NA*
ALRICK	*86%*	*14%*	*0%*	*72%*	*28%*	*0%*	*92%*	*3%*	*5%*
FARADY	*89%*	*11%*	*0%*	*78%*	*22%*	*0%*	*98%*	*1%*	*1%*
ST.JAMES	*78%*	*17%*	*5%*	*76%*	*19%*	*5%*	*86%*	*6%*	*8%*
OCEAN TERMINAL	*72%*	*25%*	*3%*	*74%*	*23%*	*3%*	*89%*	*4%*	*7%*

Table 4.6 clearly shows that selecting nearest floor based on *KNN* algorithm mostly performed better in all venues, except Ocean Terminal. This can be explained due to openings between floors that is only present in this venue. On the other hand, it also proofs that Group variance probability method are more robust and less sensitive to building structure and layout. The results show significant improvements achieved through the new combined algorithm. Nevertheless, this success is dependent on the training of weightage parameters and might not be transferable globally without further research. In addition, looking into the percentage of "*NA*", both *KNN* and Group Variance tend to always provide floor estimation, unless WAPs in observations are not mapped in reference *WAPs* database. On the other hand, the combined solution produces slightly higher percentage of "*NA"* due to the validation of probability we have added.

4.4 Conclusion

In this chapter we set various models to utilise our generated database of *WAPs*. We focused on three different implementations of such database. The first was estimating mobile units position using Handover algorithm. In the second part of the chapter, we examined the ability of estimating errors in database and positioning by modelling various quality metrics. Finally, we set to try mitigating floor ambiguity as separated algorithm from 2D position estimation, as we earlier discussed in chapter three.

In terms of positioning estimation of mobile units, we reported on different results between venues and explained the variation as due to the differences in crowd-sourcing, size, *WAP* count and layout. However, as the main purpose was to verify the quality of *WAPs* database, we believe that the results are satisfactory without expanding more advanced analysis. We also highlighted that mass-spring would be better fit for performing indoor positioning when utilising *WAPs* database derived from crowd-sourcing as it outperforms the Handover algorithm proposed in this chapter. However, it was also clear that the work we did on Handover had improved the results over the implementation of traditional weighted centroid. Finally, we explained that a better result could be expected by combining handover with mass-spring.

In terms of modelling errors in radio-maps and position estimation, we presented comprehensive study and designed custom model tailored for crowd-sourcing deployment. However, the main contribution of our model was presenting error estimation as self-evaluation problem that was hardly covered in literature. We strongly believe that similar models should be adopted by active research groups working on indoor positioning. More importantly, any positioning framework or hybrid deployment proposal, should be able to provide quality metrics as well as positioning data. Finally, the model we presented only deal with errors in circular format and do not present multi-dimensional errors. To provide more realistic error estimation more work would be required to fit our proposed model into multi-dimensional frame.

Finally, solving floor ambiguity has been confronted in this chapter by testing two different theories. Our implementation and experimental results would enable global deployment of multi-dimensional indoor positioning system once reference floor can be assigned to enough *WAPs* in crowd-sourced database. Further work on this topic would focus on utilization of barometer readings into crowd-sourcing frame. Recent research [35], [70] have already started considering barometer measure. However, the main challenge remains in the absence of continuity when it comes to collecting user's data during crowd-sourcing. This limitation will certainly affect barometer models that require ground floor reference on the same device to detect floor change events or work out how many floors are in the building. Furthermore, the absence of unified ground floor definition makes the indexing of floors even more complicated and run into problems of usability of these indexes. In some countries, ground floors are labelled to start from *0*, while other countries start labelling from *1*. Therefore, some input from maps are expected to play important part in the utilization of multidimensional crowd-sourcing in the coming years.

5 Conclusion and Future Research

During our research, we focused on testing the feasibility of crowd-sourcing *WiFi* signal sources in a global scale. Our research has provided evidence to justify the validity of such concept. We also addressed the research problem by employing unsupervised learning techniques to estimate *WAPs* signal attributes and locations. Our learning is performed using large scale data with global distribution and independent from maps layout or any user specified environmental parameters. In summary, we study the effectiveness of managing local signal attributes by adopting grid approach. Along with the large-scale data available to us, the employed data clustering and classification algorithms enabled local optimization of each area on its own. Furthermore, we presented the details of our framework implementation including grid formation and data structure. To validate results, we proposed using sanity tests to accommodate the absence of ground truth data outside the university lab. Our described sanity tests provide enough confidence of emulated ground truth reference coverage area close to signal sources physical presence. Compared to the state of the art, our research is the first to address this problem with implementation justification and large-scale test data.

To further study the optimization of signal sources estimated positions, we presented a method that involve modeling signal sources and reference clusters as graph. We then proved that using mass-spring relaxation algorithm to optimize the graph is applicable to this research target. Our research then went on testing mass-spring with limited freedom and different levels of graph connectivity. Particularly, we have proofed that restraining mass-spring to limited area of freedom of each node in the graph would provide the best performance. Finally, we presented our results for running the proposed algorithms on dataset covering different *WAPs* density and building sizes. To justify signal sources positioning quality, we compared the position of each mac address to the extracted coverage area from ground truth data. One can argue that our results could be affected by the flexibility of coverage area estimation.

However, as a sanity tests, the presented results supported our theory to enable large scale crowd-sourcing of signal sources.

In the last chapter, we have also examined three different ways to utilize *WAPs* database. Firstly, we presented our Handover algorithm that proved to provide better weighting over traditional weighted centroid. However, to further test Handover algorithm accuracy we also compared it to mass-spring. The presented results identified mass-spring as better fit for crowd-sourcing. We then presented our own model for estimating positioning errors based on quality measures of crowd-sourcing. We compared our model with two models appeared in literature but previously used for fingerprinting. We then provided justification that the proposed model is better tailored for radio-maps created via crowd-sourcing.

Finally, we tested the possibility of providing floor estimation to complement location attributes. As we search for valid solution, we compared *KNN* implementation, tailored to *WAPs* database utilization, with our own statistical model. Initial results showed that *KNN* outperform the proposed statistical model. However, after further modification to *KNN* to provide probability estimation per floor, a combined solution was proposed. As a conclusion, our results show that combining floor probabilities from all the features we examined, *KNN*, variance, range and availability, provides best performance. To conclude, the brief discussion of floor estimation was inspired by the importance of the topic. Nevertheless, we still think that to support critical emergency services, less than *1%* error in floor estimation is required. Hence, further research would be required to enable an efficient floor identification.

5.1 Future research

Given the opportunity for conducting future research, we would examine more advanced data clustering and classification algorithm. Particularly, the recent development of deep learning techniques as functional descriptive, or unsupervised learning, should motivate researchers to try artificial neural networks as a way to describe signal propagation in unknown territories. The primary target of such functions would be to generate multi-layers network of neurons that would identify valid clusters from large set of signal observations collocated in limited area. This could save segmentation efforts and substitute for fine global grid. Hence such research would also need to be coupled with rough geocaching system that utilize roads and open GIS data to divide the earth into linked sections. In a way, the computational requirements in realization of such solution would be much more than proposed solution. However, it might allow implementing priorities when processing incoming data to create batches with different frequencies.

Further research concept can also test the validity of coupling mass-spring relaxation with fuzzy frame. The main thinking behind that is due to the amount of noise present in crowd-sourced input data. Considering the constructed graph for mass-spring, edges in a form of fuzzy data can add flexibility to reach global minima compared to crisp data. This research has demonstrated that allowing level of freedom to each node would also improve results. Hence, transforming all input data into fuzzy values before executing mass-spring would produce more distributed input across the venue. In addition to mass-spring, fuzzy frame can also play a part in improving data classification and clustering. Estimating the belongingness of any signal measurement to a given geoindex can be easily translated into fuzzy input. With set of defined ownership degrees, clustered data in each geoindex can claim new signal measurements as they are streamed in.

Finally, more advanced 3D algorithms utilizing barometer readings in large scale deployment is certainly the most urgent research topic that should be conducted. We would argue that efforts in crowd-sourcing floor transition features in unknown topology or layout is valid topic for future research. Such methods will help assigning relative floors to WAPs without the need for user inputs or detail layouts from maps.

6 Bibliography

[1] S. Subedi and J.-Y. Pyun, “Practical Fingerprinting Localization for Indoor Positioning System by Using Beacons,” *J. Sensors*, vol. 2017, pp. 1–16, 2017.

[2] J. Bardwell, “Converting Signal Strength Percentage to dBm Values,” *WildPackets, Inc*, no. November, pp. 1–12, 2002.

[3] P. Davidson and R. Piché, “A Survey of Selected Indoor Positioning Methods for Smartphones,” *IEEE Communications Surveys and Tutorials*, vol. 19, no. 2. pp. 1347–1370, 2017.

[4] Y. Du, T. Arslan, and A. Juri, “Camera-aided region-based magnetic field indoor positioning,” in *2016 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2016*, 2016.

[5] A. Franchi, G. Oriolo, and P. Stegagno, “Probabilistic mutual localization in multi-agent systems from anonymous position measures,” *Proc. IEEE Conf. Decis. Control*, pp. 6534–6540, 2010.

[6] R. Hostettler and S. Särkkä, “IMU and magnetometer modeling for smartphone-based PDR,” in *2016 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2016*, 2016.

[7] W. Liu, X. Fu, and Z. Deng, “Coordinate-based clustering method for indoor fingerprinting localization in dense cluttered environments,” *Sensors (Switzerland)*, vol. 16, no. 12, 2016.

[8] E. Lohan, J. Torres-Sospedra, H. Leppäkoski, P. Richter, Z. Peng, and J. Huerta, “Wi-Fi Crowdsourced Fingerprinting Dataset for Indoor Positioning,” *Data*, vol. 2, no. 4, p. 32, 2017.

[9] K. Abdulrahim, C. Hide, T. Moore, and C. Hill, “Increased Error Observability of an Inertial Pedestrian Navigation System by Rotating IMU,” *J. Eng. Technol. Sci.*, vol. 46, no. 2, pp. 211–225, 2014.

[10] J. Cheng, Y. Cai, Q. Zhang, J. Cheng, and C. Yan, “A new three-dimensional indoor positioning mechanism based on wireless LAN,” *Math. Probl. Eng.*, vol. 2014, 2014.

[11] M. Socha, W. Górka, and I. Kostorz, “Fuzzy logic in indoor position determination system,” *Theor. Appl. Informatics*, vol. 27, no. 2, pp. 1–15, 2016.

[12] P. A. Zandbergen, “Accuracy of iPhone locations: A comparison of assisted GPS, WiFi and cellular positioning,” *Trans. GIS*, vol. 13, no. SUPPL. 1, pp. 5–25, 2009.

[13] R. F. Brena, J. P. García-Vázquez, C. E. Galván-Tejada, D. Muñoz-Rodriguez, C. Vargas-Rosales, and J. Fangmeyer, “Evolution of Indoor Positioning Technologies: A Survey,” *J. Sensors*, vol. 2017, pp. 1–21, 2017.

[14] A. Yassin, Y. Nasser, M. Awad, and A. Al-Dubai, “Recent Advances in Indoor Localization: A Survey on Theoretical Approaches and Applications,” *Surv. Tutorials*, pp. 1–21, 2016.

[15] H. Nurminen, M. Dashti, and R. Piché, “A Survey on Wireless Transmitter Localization Using Signal Strength Measurements,” *Wirel. Commun. Mob. Comput.*, vol. 2017, pp. 1–12, 2017.

[16] G. De Angelis, A. Moschitta, and P. Carbone, “Positioning techniques in indoor environments based on stochastic modeling of UWB round-trip-time measurements,” *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 8, pp. 2272–

2281, 2016.

[17] S. Bartelmaos, K. Abed-Meraim, and R. Leyman, “General selection criteria to mitigate the impact of NLoS errors in RTT measurements for mobile positioning,” in *IEEE International Conference on Communications*, 2007, pp. 4674–4679.

[18] A. Bahillo, S. Mazuelas, J. Prieto, P. Fernández, R. M. Lorenzo, and E. J. Abril, “Hybrid RSS-RTT localization scheme for wireless networks,” in *2010 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2010 - Conference Proceedings*, 2010.

[19] C. Hoene and G. Andre, “Measuring Round Trip Times to Determine the Distance between WLAN Nodes,” *TKN Tech. Rep.*, vol. TKN-04-16, no. December, pp. 20–23, 2004.

[20] E. Au, “The Latest Progress on IEEE 802.11mc and IEEE 802.11ai [Standards],” *IEEE Veh. Technol. Mag.*, vol. 11, no. 3, pp. 19–21, 2016.

[21] Z. Wei, Y. Zhao, X. Liu, and Z. Feng, “DoA-LF: A Location Fingerprint Positioning Algorithm with Millimeter-Wave,” *IEEE Access*, 2017.

[22] S. Shrestha, J. Talvitie, and E. S. Lohan, “Deconvolution-based indoor localization with WLAN signals and unknown access point locations,” in *2013 International Conference on Localization and GNSS (ICL-GNSS)*, 2013, pp. 1–6.

[23] D. Faria, “Modeling signal attenuation in ieee 802.11,” *Wirel. Networks*, vol. 1, pp. 1–11, 2005.

[24] G. Retscher, “FUSION of LOCATION FINGERPRINTING and TRILATERATION BASED on the EXAMPLE of DIFFERENTIAL WI-FI POSITIONING,” in *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information*

Sciences, 2017, vol. 4, no. 2W4, pp. 377–384.

[25] H. Choi, Y. Koo, S. Lee, and S. Park, "Indoor positioning system based on an improved weighted-trilateration algorithm with fingerprinting technique," *Int. J. Multimed. Ubiquitous Eng.*, vol. 11, no. 9, pp. 165–176, 2016.

[26] J. Koo and H. Cha, "Unsupervised locating of WiFi access points using smartphones," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 42, no. 6, pp. 1341–1353, 2012.

[27] R. Battiti, T. Le Nhat, and A. Villani, "LOCATION-AWARE COMPUTING: A NEURAL NETWORK MODEL FOR DETERMINING LOCATION," *Wirel. LANs*, p. http://www.dit.unitn.it, 2002.

[28] G. Minaev and A. Visa, "Comprehensive Survey of Similarity Measures for Ranked Based Location Fingerprinting Algorithm," in *International Conference on Indoor Positioning and Indoor Navigation, IPIN 2017*, 2017, pp. 1–4.

[29] P. Bahl and V. N. Padmanabhan, "RADAR: an in-building RF-based user location and tracking system," *Proc. IEEE INFOCOM 2000. Conf. Comput. Commun. Ninet. Annu. Jt. Conf. IEEE Comput. Commun. Soc. (Cat. No.00CH37064)*, vol. 2, pp. 775–784.

[30] M. Youssef and A. Agrawala, "The Horus WLAN location determination system," *Proc. 3rd Int. Conf. Mob. Syst. Appl. Serv. - MobiSys '05*, p. 205, 2005.

[31] H. J, L. A, and S. I, "Practical lessons from PlaceLab," *IEEE Pervasive Comput.*, vol. 5, no. 3, pp. 32–39, 2006.

[32] H. Nurminen *et al.*, "Statistical path loss parameter estimation and positioning using RSS measurements in indoor wireless networks," in *2012*

International Conference on Indoor Positioning and Indoor Navigation (IPIN), 2012, pp. 1–9.

[33] H. Cheng, F. Wang, R. Tao, H. Luo, and F. Zhao, “Clustering algorithms research for device-clustering localization,” in *2012 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2012 - Conference Proceedings*, 2012.

[34] J. Koo and H. Cha, “Localizing WiFi access points using signal strength,” *IEEE Commun. Lett.*, vol. 15, no. 2, pp. 187–189, 2011.

[35] S. Burgess, K. Åström, M. Högström, B. Lindquist, and R. Ljungberg, “Smartphone positioning in multi-floor environments without calibration or added infrastructure,” *2016 Int. Conf. Indoor Position. Indoor Navig. IPIN 2016*, pp. 4–7, 2016.

[36] J. Eckert, F. Villanueva, R. German, and F. Dressler, “Distributed Mass-Spring-Relaxation for Anchor-Free Self-Localization in Sensor and Actor Networks,” in *2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*, 2011, no. September, pp. 1–8.

[37] C. Liang and R. Piché, “Mobile tracking and parameter learning in unknown non-line-of-sight conditions,” in *13th Conference on Information Fusion, Fusion 2010*, 2010, no. July, pp. 1–6.

[38] S. Schwarz, I. Safiulin, T. Philosof, and M. Rupp, “Gaussian Modeling of Spatially Correlated LOS/NLOS Maps for Mobile Communications,” in *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*, 2016, pp. 1–5.

[39] Z. Xiao, H. Wen, A. Markham, N. Trigoni, P. Blunsom, and J. Frolik, “Non-Line-of-Sight Identification and Mitigation Using Received Signal Strength,” *IEEE*

Trans. Wirel. Commun., vol. 14, no. 3, pp. 1689–1702, 2015.

[40] F. Alsehly, T. Arslan, and Z. Sevak, "LOCATING ELECTROMAGNETIC SIGNAL SOURCES," EP3098620 (A1), 2009.

[41] Apache, "HBase," *The Apache Software Foundation*, 2017. [Online]. Available: http://hbase.apache.org/.

[42] The Apache Software Foundation, "Apache Cassandra Database," *Cassandra*, 2015.

[43] Amazon, "Amazon DynamoDB - NoSQL Cloud Database Service," *Amazon Web Services*, 2017. [Online]. Available: https://aws.amazon.com/dynamodb/.

[44] D. Hutchison, J. Kepner, V. Gadepally, and B. Howe, "From NoSQL Accumulo to NewSQL Graphulo: Design and utility of graph algorithms inside a BigTable database," in *2016 IEEE High Performance Extreme Computing Conference (HPEC)*, 2016, pp. 1–9.

[45] S. Brunozzi, "Big Data and NoSQL with Amazon DynamoDB," in *Proceedings of the 2012 workshop on Management of big data systems - MBDS '12*, 2012, p. 41.

[46] S. Kalid, A. Syed, A. Mohammad, and M. N. Halgamuge, "Big-data NoSQL databases: A comparison and analysis of 'Big-Table', 'DynamoDB', and 'Cassandra,'" in *2017 IEEE 2nd International Conference on Big Data Analysis, ICBDA 2017*, 2017, pp. 89–93.

[47] A. Fox, C. Eichelberger, J. Hughes, and S. Lyon, "Spatio-temporal indexing in non-relational distributed databases," in *Proceedings - 2013 IEEE International Conference on Big Data, Big Data 2013*, 2013, pp. 291–299.

[48] F. Galcik and M. Opiela, “Grid-based indoor localization using smartphones,” in *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2016, no. October, pp. 1–8.

[49] W. Muttitanon, N. K. Tripathi, and M. Souris, “An Indoor Positioning System (IPS) Using Grid Model,” *J. Comput. Sci.*, vol. 3, no. 12, pp. 907–913, 2007.

[50] F. Alsehly, R. M. Sabri, Z. Sevak, and T. Arslan, “Dynamic Indoor Positioning with the Handover Algorithm,” in *Proceedings of the 23rd International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2010)*, 2010, pp. 1233–1242.

[51] J. Luo and X. Zhan, “Characterization of Smart Phone Received Signal Strength Indication for WLAN Indoor Positioning Accuracy Improvement,” *J. Networks*, vol. 9, no. 3, pp. 739–746, Mar. 2014.

[52] H. P. Kriegel, P. Kröger, J. Sander, and A. Zimek, “Density-based clustering,” *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 1, no. 3, pp. 231–240, 2011.

[53] K. M. Kumar and A. R. M. Reddy, “An efficient k-means clustering filtering algorithm using density based initial cluster centers,” *Inf. Sci. (Ny).*, vol. 418–419, pp. 286–301, 2017.

[54] I. N. Kosović and T. Jagušt, “Enhanced Weighted Centroid Localization Algorithm for Indoor Environments,” *Int. Conf. Commun. Networks Appl. (ICCNA 2014)*, vol. 8, no. 7, pp. 1184–1188, 2014.

[55] S. Gansemer, U. Großmann, and S. Hakobyan, “RSSI-based Euclidean distance algorithm for Indoor Positioning adapted for the use in dynamically changing WLAN environments and multi-level buildings,” *2010 Int. Conf. Indoor*

Position. Indoor Navig. IPIN 2010 - Conf. Proc., no. September, pp. 15–17, 2010.

[56] X. Cui and A. Technology, “Distributed localization for anchor-free sensor networks * * This project was partially supported by the National Natural Scie ....,” no. May, 2016.

[57] A. LaMarca *et al.*, “Place Lab: Device Positioning Using Radio Beacons in the Wild,” 2005, pp. 116–133.

[58] R. Mautz, “Indoor Positioning Technologies,” *Inst. Geod. Photogramm. Dep. Civil, Environ. Geomat. Eng. ETH Zurich*, no. February 2012, p. 127, 2012.

[59] X. Wang, M. Jiang, Z. Guo, N. Hu, Z. Sun, and J. Liu, “An Indoor Positioning Method for Smartphones Using Landmarks and PDR,” *Sensors*, vol. 16, no. 12, p. 2135, Dec. 2016.

[60] M. Wallbaum, “A priori error estimates for wireless local area network positioning systems,” *Pervasive Mob. Comput.*, vol. 3, no. 5, pp. 560–580, 2007.

[61] A. S. Krishnakumar and P. Krishnan, “On the accuracy of signal strength-based location estimation techniques,” in *Proceedings - IEEE INFOCOM*, 2005, vol. 1, pp. 642–650.

[62] H. Lemelson, M. B. Kjærgaard, R. Hansen, and T. King, “Error estimation for indoor 802.11 location fingerprinting,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2009, vol. 5561 LNCS, pp. 138–155.

[63] G. Caso and L. De Nardis, “On the applicability of multi-wall multi-floor propagation models to WiFi fingerprinting indoor positioning,” in *Lecture*

Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, 2015, vol. 159, pp. 166–172.

[64] G. Caso and L. De Nardis, "Virtual and Oriented WiFi Fingerprinting Indoor Positioning based on Multi-Wall Multi-Floor Propagation Models," *Mob. Networks Appl.*, vol. 22, no. 5, pp. 825–833, 2017.

[65] a. S. M. Al-Ahmadi, a. I. a. Omer, M. R. B. Kamarudin, and T. B. a. Rahman, "Multi-Floor Indoor Positioning System Using Bayesian Graphical Models," *Prog. Electromagn. Res.*, vol. 25, no. September, pp. 241–259, 2010.

[66] V. Chandel, N. Ahmed, S. Arora, and A. Ghose, "InLoc: An end-to-end robust indoor localization and routing solution using mobile phones and BLE beacons," in *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2016, pp. 1–8.

[67] O. Sutton, "Introduction to k Nearest Neighbour Classification and Condensed Nearest Neighbour Data Reduction," *Introd. to k Nearest Neighb. Classif.*, pp. 1–10, 2012.

[68] R. S. Campos, L. Lovisolo, and M. L. R. De Campos, "Wi-Fi multi-floor indoor positioning considering architectural aspects and controlled computational complexity," *Expert Syst. Appl.*, vol. 41, no. 14, pp. 6211–6223, 2014.

[69] F. Alsehly, T. Arslan, and Z. Sevak, "Indoor positioning with floor determination in multi story buildings," in *2011 International Conference on Indoor Positioning and Indoor Navigation*, 2011, pp. 1–7.

[70] H. Ye, T. Gu, X. Tao, and J. Lu, "Scalable floor localization using barometer on smartphone," *Wirel. Commun. Mob. Comput.*, vol. 16, no. 16, pp. 2557–2571, 2016.