



Tracking With Wi-Fi

Final Report

ECM3401 Individual Literature Review and Project

Artiom Nistean

Abstract

The strong battery drain that GPS immolates onto the modern day smartphone indicates that a Wi-Fi based approach should be considered in urban areas. Additionally, the accuracy flaws that GPS brings to urban areas invokes the need to use the wealth of WiFi information available in urban environments to develop a WiFi based geolocation algorithm. This paper details research the effectiveness of obtaining a geolocation using WiFi signals. It covers the surrounding literature in this technical frontier and discusses a means to gather the WiFi information. Crowd sourcing with the combination of smartphones can be used to gather the relevant WiFi data in urban areas quickly and effectively. This paper shows an Android application that is able to gather WiFi information in an autonomous fashion. The application closely works with a remote server that records and collects the information accordingly. Once the information is gathered relevant data structures can be generated to best represent and store the information. This paper discusses a Delaunay triangulation approach to finding the intersection of an access points footprint. It also shows a method of increasing an the possible spatial resolution of an area to obtain a more accurate estimated location area. The WiFi based geolocation algorithm is tested and illustrated on a set of WiFi data showing promising results.

I certify that all material in this dissertation which is not my own work has been identified.

May 2016

Table of Contents

1. Introduction	1
1.1 - Project Motivation	1
1.2 - Aims & Objectives	1
1.3 - Report Structure	1
1.4 - Complexity.....	2
2. Summary of Literature & Specification and Design	2
2.1 - Literature Review.....	2
2.1.1 - Current WiFi geolocation technology	2
2.1.2 - WiFi battery consumption compared to GPS.....	2
2.1.3 - Smartphone popularity & The Android API	3
2.2 - Specification & Design.....	4
2.2.1 - Requirements & Design	4
2.2.2 - Android API	4
2.2.3 - Geofence.....	4
2.2.4 - Location & Tracking Algorithm	5
2.2.5 - Testing & Evaluation.....	5
2.2.6 - Time Plan	6
3. Software Engineering.....	6
3.1 - Methodology	6
3.2 - Specification & Design.....	7
3.3 - Implementation	9
3.4 - Testing.....	12
3.5 - Results and Evaluation	13
4. Data Analysis	14
4.1 - Server Design & Implementation	14
4.2 - Data Manipulation	15
4.2.1 - JSON data	15
4.2.2 - Visualisation of a data file	15
4.2.3 - Data Structures	16
5. Research & Experiments.....	17
5.1 - Methodology	17
5.2 - Design	18
5.3 - Implementation	19

5.4 - Testing.....	21
5.5 - Results and Evaluation	22
6. Critical Assessment of The Project as a Whole	23
6.1 - Positives	23
6.2 - Negatives	24
7. Future work	24
8. Conclusion	25
9. Acknowledgements	25
References	26
Appendix	28

1. Introduction

1.1 - Project Motivation

The current approach to obtaining a geolocation in an urban environment consumes too much energy, and is often inaccurate. Urban environments are rich with useful WiFi information that can be used to obtain a geolocation. Using this WiFi information is key to computing the geolocation in an energy efficient manner. Currently, smartphones act as the primary device to obtaining an individuals geolocation, the reason for this being that smartphones are widely used with approximately half of the population owning a smartphone [The Economist, 2015]. These devices possess sufficient geolocation computation capabilities through their built in GPS systems. However, using these GPS systems for a prolonged period of time, can be too costly on the smartphones battery because GPS drastically drains battery and is thus inoperable for lengthy periods on mobile devices [Sharkey, 2009]. As a result, this makes constant tracking infeasible. A WiFi connection, on the other hand, can be active for the most part of the day and night. In urban areas, WiFi access points are densely populated, which provides plenty of data and resources for finding a geolocation using WiFi information. Another motive for this project is the proven inaccuracy of GPS in an urban environment by Wing, Eklund, and Kellogg (2005) in the Journal of Forestry. This shows that an alternative solution, which use WiFi signals, should be explored and used in urban environments and areas where GPS struggles to accurately determine a location. As a result geolocation tracking that uses WiFi signals appears to be a more viable and elegant solution on mobile devices.

1.2 - Aims & Objectives

The aim of this research project is to investigate the effectiveness of a solution in finding a geolocation using WiFi signals. The research aims to uncover the possible techniques and theory behind finding a WiFi based geolocation from a set of underlying WiFi data. The literature has already revealed that there must be an underlying knowledge of the area to be able to find a geolocation using WiFi [Sapiezynski et al., 2015]. Therefore this research project will require a means to effectively and autonomously gather WiFi data. This data can be used to develop an effective WiFi based geolocation algorithm. This project should aim to avoid gathering inaccurate results, because these inaccuracies will ultimately affect the accuracy of the WiFi based geolocation. The objective of this research project is to explore solutions to obtaining a geolocation in an urban environment using WiFi signals. The solution should incorporate a more battery efficient method that yields the same, if not more accurate results than current smartphone GPS services. Overall, this project aims to deliver an effective data gathering method and research into manipulating the gathered data to obtain a geolocation.

1.3 - Report Structure

This report will begin by summarising the literature surrounding this project, outlining current advancements and discoveries the literature has revealed. Additionally, a specification and design will be summarised to outline the requirements and designs for this project. Due to the nature of this project, this report can further be sectioned into two parts. The first part will cover the software engineering aspect of this project, that is, the data gathering which will give detail to the data gathering application of this project as well as cover any data analysis on the collected data.

The second part of the report will cover the research and experiments on the collected data. This will shed light to the WiFi based geolocation algorithm used to obtain a geolocation estimate. A reflection on the project work and any future work will be detailed towards the end of the report.

1.4 - Complexity

The complexity of this project exists in the fact that this project is amongst the first of its kind. This is because the similar projects do not approach the task of finding a geolocation using WiFi signals in the same method. Additionally, this project uses an autonomous smartphone application to gather a large quantity of data that is then processed to create efficient and useful data structures. The scale of this project ensures it is an ambitious undertaking, and as this report will reveal, the project contains multiple computer science aspects.

2. Summary of Literature & Specification and Design

This section will aim to summarise the literature which surrounds this project. This section will discuss the current WiFi geolocation technologies and highlight the high energy impact GPS has on smartphone batteries. Additionally, this section will outline the specification of the project, commenting on the design as appropriate.

2.1 - Literature Review

2.1.1 - *Current WiFi geolocation technology*

The literature reveals that there is no clear knowledge of how WiFi networks can be used to aid in computing a geolocation. It is best assumed that this information is proprietary to large companies that use it to aid in finding a GPS geolocation. The exact utility and mechanics of how geolocation is found using WiFi is therefore, largely unknown, however recent projects have described their efforts to find a WiFi geolocation. There have been narrow and non-systematic studies which have reported on various suggestions and methods on finding the geolocation using WiFi signatures [Kawaguchi, 2009]. For instance, there exists a Bayesian approach to detect pedestrian destinations using WiFi [Danalet et al., 2014]. This is accomplished by using network traces that are supported by a knowledge of an underlying pedestrian map. The different literatures prove that they all require knowledge of the map behind the technical infrastructure to be able to achieve accurate location results.

2.1.2 - *WiFi battery consumption compared to GPS*

The literature proves that WiFi consumes much less battery power than GPS. This is mainly due to the better battery performance that WiFi inherits on smartphones compared to GPS systems. The power consumption related to WiFi depends on the strength of the receiving signal. The strength of a WiFi signal is dependent of the physical distance from the access point as well the objects such as walls and glass in-between the access point [Chen and Kobayashi, 2002]. Additionally, receiving the signal through a large cluster of other broadcasting signals would require more power compared to connecting to a single broadcasting access point. On the other hand, GPS operates by receiving and decoding time signals sent from orbiting satellites. The decoding process requires a considerable amount of computational power due to the fact that the

processor needs to pick out the weak signals from the background noise and compare them with each other [Limonard, 2013]. When traveling, this process is done repeatedly. The faster the movement, the more energy is consumed when processing the correct geolocation [Paek et al., 2010]. Figure 1 shows the GPS power consumption on a N95 smartphone. GPS is repeatedly turned on for 30 seconds and turned off for 90 seconds. GPS consumes more than five times the power when activated which proves to be a roadblock on the way to all-day smartphone usage [Paek et al., 2010].

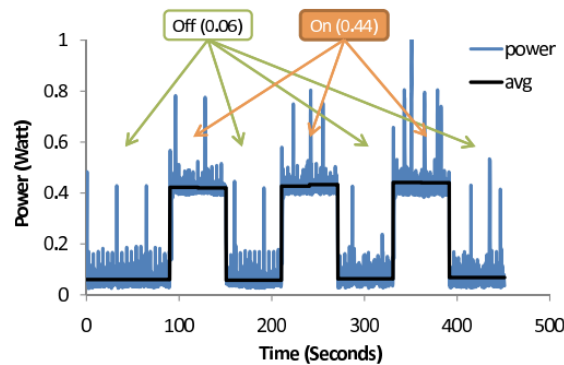


Figure 1. Power Consumption of GPS [Paek et al., 2010].

2.1.3 - Smartphone popularity & The Android API

As mentioned, smartphones are increasingly in popularity and usage according to The Economist (2015). Smartphones have access to sufficient GPS technology that enable them to calculate a geolocation and act as navigation aids [Webster, and Webster 2013]. However, as mentioned GPS is not accurate enough in urban areas. Location-based applications will have to deal with the level of error that GPS presents in urban areas by using application-specific methods, such as map-matching or map-snapping. Map-matching is the process of matching a sequence of real world coordinates to a digital map. This is typically done using a Geographic Information System, however, this can sometimes be computationally expensive [Kay and Schuessler, 2009]. This can hinder the speed at which a location can be calculated and delivered to the end user. Paek, Kim and Govindan (2010) suggest to trade off some position accuracy for reduced GPS energy usage by duty-cycling the GPS usage, but this creates problems in geolocation accuracy. This reveals the need to be cautious with GPS usage when using smartphones to gather the required data.

In 2015, Android dominated the smartphone market with a market share of over 80%. [IDC, 2015]. This popularity dictates the need to investigate the Android API. The literature discovers unique Android permission named *ACCESS_WIFI_STATE*, that allows an Android application to request and receive a list of all visible access points as well as their respective MAC address identifiers. [Android API, (n.d)] This is done after each WiFi scan ordered by any application that is installed on the smartphone. This enables a truly autonomous process of data gathering because, the android application can operate in the background whenever the initial WiFi scans results are collected.

2.2 - Specification & Design

2.2.1 - Requirements & Design

The literature dictates the need to develop a data gathering application. Due to the popularity of the Android operating system and the utility of the Android API, it naturally makes sense to develop a data gathering application on the Android platform. This would maximise the number of users that may use the application to contribute to gathering data. This crowd sourced approach can allow quick and autonomous data collection.

It should be mentioned that this application needs to perform these tasks in an autonomous manner. This means performing the gathering actions every time a predetermined amount of time has elapsed. The amount of time can be altered in the application settings or tweaked in testing to achieve results as desired. Additionally, the application must periodically check the users location to check if they are inside the geofence, and thus, determine whether to proceed with gathering the information.

The geolocation algorithm will use the information that has been collected from the data gathering application as the underlying knowledge of the area. The algorithm should be initially developed separate to the server, which is to communicate with the tracking algorithm. This gives focus on the core functionality and technique of the algorithm. As a result, its suggested that the location algorithm be in a simpler programming language with emphasis on visualising the data on a map or diagram.

Once an algorithm has been developed and tested, it can be integrated into the server, to be enable communication between the user and the server. This will be the basis for the tracking algorithm as a list of access points that are audible and their respective strengths can be sent to the server for processing. Thereafter, the algorithm can be used to determine the geolocation information and relay the latitude and longitude coordinates of the estimated geolocation back to the user.

2.2.2 - Android API

As the literature reveals Android permissions are ideal for gathering WiFi information. The latest API level of Android asks permissions at runtime which allows for a more intuitive experience when using the an application. This suggests that the relevant permissions should be prompted at the correct time, such as to maximise the autonomous functionality of this application. It is important to focus on automation and efficient battery usage produced by GPS when gathering the relevant location data associated with the WiFi information. Gathering data in an autonomous manner will give minimum interaction with the user, which allows for a quicker and easier gathering of crowd sourced data.

2.2.3 - Geofence

Obtaining a location and utilising the geofence requires the use of Android location services. As a consequence, the application will require sufficient location permission from the user. The geofence should be initialised to function in the background of the application to further promote the automation of the application. The University of Exeter streatham campus is chosen as the

area in which to find a WiFi geolocation for this project. As a result, the geofence for this project can be set up by using the known GPS coordinates of the centre of the campus as well as a radius in meters, which defines the area of the geofence. Figure 2 shows the area of the geofence to be used by the data gathering application for this project.

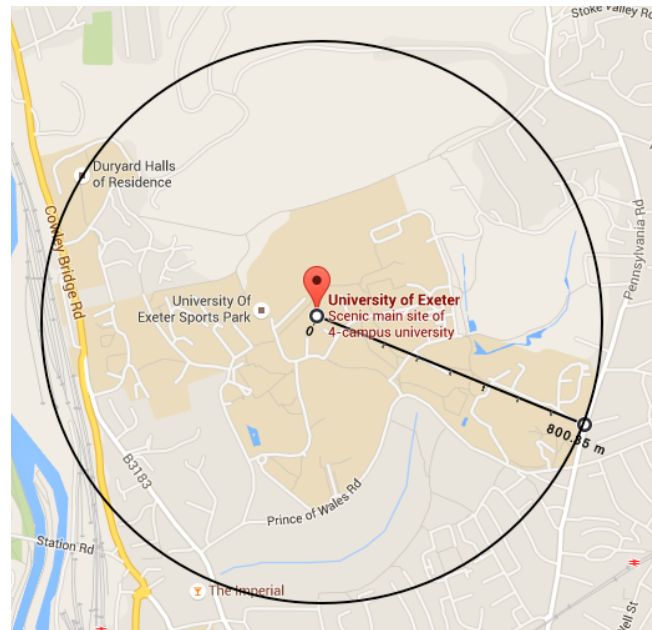


Figure 2. Geofence area for this project. Source: Google Maps.

2.2.4 - Location & Tracking Algorithm

When developing the location algorithm there should be focus on the core functionality, that is obtaining the geolocation from a known set of points which the data gathering application provides. For this reason, the location algorithm should be developed separate to the data gathering application. There is a need to display and visualise the algorithms output, for testing and a better comprehension. This can naturally be plotting points or regions on a map of the area.

Developing the location algorithm can be done in three experimenting parts. By completing each stage, the desired outcome should increase the accuracy of the algorithm. Firstly, the basis of the location algorithm will revolve around what access points the user can hear. By using information of where access points were heard, a location can be estimated by observing which access points are currently being heard. Thereafter, the signal strength of the access points can be interpolated to further increase the accuracy. Lastly, the implementation of well established tracking techniques, such as Kalman Filters, can be used to track a geolocation and further improve accuracies in the WiFi based geolocation.

2.2.5 - Testing & Evaluation

This project calls for the testing to be broken down into two parts. Firstly, testing the data gathering application aims to ensure the results that are collected are accurate and reliable. Inaccurate results pose a risk of inaccuracies to the WiFi based geolocation algorithm. Secondly, testing the location algorithm experiments would aim to evaluate the accuracy of the location prediction as well as the efficiency of the technique.

Initial tests of the data gathering application should determine the ability of gathering the correct information, being the MAC addresses and Signal Strength. By using built in Android API WiFi scanning methods, it is safe to assume accurate and desired results will be produced. Other tests aim to observe the location accuracy and geofence functionality of the data gathering application. The geofence should be crudely tested by moving in and out of the desired area and observing the applications functionality. The tests will compare known coordinates against the coordinates the application is able to produce. The accuracy of these coordinates should be calculated using Androids location client. This is because is able to return an accuracy value, in meters, which represents the radius accuracy of the location¹. Several tests should be performed to obtain a fair indication of the accuracy of the locations gathered.

It is required that the remote server, which handles the collected data, be tested to ensure the data is stored as needed. The test would aim to ensure that the server receives the gathered information and creates the necessary files.

The aim of testing the location algorithm is to see if the particular stage experiment improves the accuracy and efficiency of the technique that finds the predicted location. The method of testing these experiments would be to compare the known coordinates location against the location the algorithm predicts. If there is enough time, taking the difference between the two geolocation would reveal the accuracy error. This should be performed multiple times in addition to a number of locations to obtain an overall accuracy error. This can be done by using the root mean squared error technique. The root mean squared error is used to measure the difference between the location values predicted to the location values measured [Tateishi and Wen, 1994].

2.2.6 - Time Plan

The time plan specifies that data gathering should commence as soon as possible ignorer to dedicate as much time to experimenting the location algorithm. However, it is also necessary that the data is analysed and any conflicts are resolved and corrected. Dealing with these uncertainties and resolving conflicts in the gathered data can hinder the amount of time available to research the WiFi based location algorithm.

3. Software Engineering

This section will outline the software engineering aspect of this research project, that is the data gathering application. This section will cover the methods used for development as well as the specification of the application. Additionally the implementation of the application will be detailed, showing necessary testing and results.

3.1 - Methodology

When undertaking the software engineering aspect of this project, it is important to approach the task with the correct methodology. As with developing any application, the data gathering application is suited to be developed with a modern agile approach. However, the application is developed in a water fall fashion. The motivation behind this methodology is to deploy a

¹ Note: After all, this is just the accuracy of the smartphones GPS.

functional data gathering application and proceed to gather data as quick as possible. This would give more attention to developing a location algorithm and processing the collected data. Nonetheless, appropriate source control is vital when developing any application, therefore, GitHub is used for version and source control. GitHub uses GIT software, which is a widely used source code management system that is primarily used for software development. It will also act as a back-up for the data gathering application, and can prove to be useful during development.

3.2 - Specification & Design

The Android platform represents more than 80% of the smartphone market [IDC, 2015] and holds unique permissions for WiFi information as discussed earlier. For this reason, the data gathering application is developed for Android OS, allowing for most people to contribute in sourcing the required data. Android API level 23 represents Androids current and latest² operating system, named Android 6.0 (Marshmallow). When developing an Android application it is important to specify a target API level as well as a minimum API level to achieve maximum compatibility across all Android devices. The data gathering application should have a target API level of 23 and a minimum of 16 which represents versions of Android 4.1 and above. This allows for over 80% of all Android devices to be able to use the data gathering application [Developer.android.com, 2016].

The data gathering application needs to have an autonomous aspect. This means the application should be able to collect, process and send the necessary data without any user interaction. This minimal user interaction approach allows for a simpler and faster means to gathering the required data. Nevertheless, the application can not be fully autonomous and requires user interaction that specifies the intervals at which to gather and send the data. It is worth mentioning, for testing purposes, the application should have a degree of interaction to allow for the collection and sending of data when desired. This will prove useful for testing the application, and even gathering the data at a desired location or sending the data at a convenient time. The applications needs to perform the following tasks:

- Record the broadcasting WiFi access points MAC addresses.
- Record the respective signal strengths of the WiFi access points.
- Record accurate GPS longitude and latitude coordinates of the users current location.
- Record a time-stamp of when the information is collected.
- Record only when inside the University of Exeter campus geofence.
- Send information to a remote server.
- Autonomously collect and send the recorded data.

The application should only gather data once inside the geofence, however, sending the data should not be location specific. Instead, sending the data should depend on the network connectivity of the users device and the users settings. Ideally, the data should be sent over a WiFi connection as the data can be of a considerable size. This would prevent unwanted data usage on the users device which can be costly for the user depending on the size of the data being sent.

Once the user is inside the geofence, they should be notified and the data gathering can commence. It is important that the recorded GPS location is accurate. However, simply running

² At the time of the completion of this report (May 2016)

the GPS system for as long as needed to determine an accurate location is infeasible. This is because GPS incurs a heavy battery drain and a travelling user can record an inaccurate location. The latter is dependant on how far the user travels in the time it takes to obtain an accurate enough location. Nevertheless, a location still needs to be recorded. This problem can be solved by setting an accuracy criteria and a time threshold when obtaining the location. The application must only record a location which meets the accuracy criteria. For example, obtaining locations with an accuracy lower than 10 meters. If this criteria can not be met due to the surrounding environment, or simply the limitation of the users device, the application must record a GPS location after a certain amount of time has passed. The application should record the most accurate location it was able to obtain. This process is detailed in the flowchart shown in figure 3.

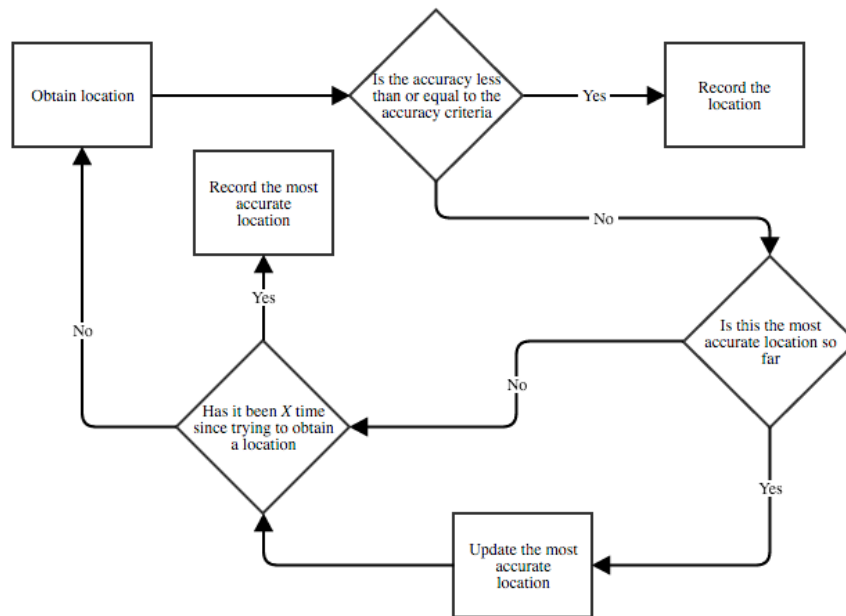


Figure 3. Flowchart for recording a location that meets accuracy criteria within a time threshold.

The data gathering application needs sufficient permissions to perform the required tasks. With the introduction of runtime permissions in Android 6.0 (API level 23) the application will need prompt access to each permission when it is required. The permissions that the application requires are outlined below in Table 1.

Permission	Purpose
Access Fine Location	Locating the user for the geofence and for recording their coordinates
Access WiFi State	Using the Android device's WiFi scanner for WiFi information
Change WiFi State	Enabling the WiFi scanner
Storage	Storing the gathered data as necessary.
Internet Access	Sending the information to a remote server.
Network State	Determining if the user is connected to WiFi or has an internet connection.

Table 1. Permissions are their purpose for the Android data gathering application.

3.3 - Implementation

The Android application was developed using Android Studio as the IDE and platform for development. It took approximately one and a half months to develop a working prototype that is able to gather the required data in an autonomous fashion.

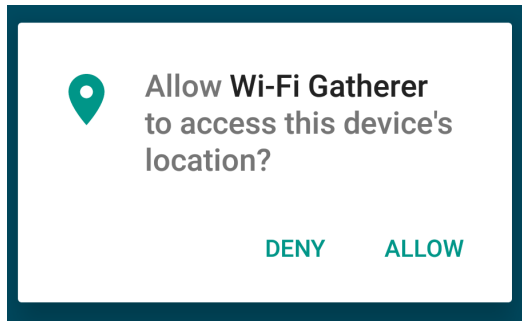


Figure 4. The Android application asking for location permission on runtime.

Starting the application for the first time allows the application to initialise the geofence. To do this, the users permission is required. Therefore a runtime permission is request when first launching the application. This is shown in figure 4 on the left. At lower Android API levels, this permission is requested when installing the application. The geofence is only set up if the user allows the permission for their location. The geofence is set up to use a *GoogleAPIClient*, which handles all location services for this application.

The *GeofenceTransitionsIntentService* class, handles the geofence transitions and holds responsibility for the functionality of the geofence. This class is set up once the *GoogleAPIClient* has established a connection to the location services provided by Google. This is set up in the *MainActivity* of the application, and is set to function in the background as long as there is an instance of the application. The application is able to send an *intent* to this class. The *intent* is sent when the user transitions in and out of the geofence. The *GeofenceTransitionsIntentService* class computes the necessary transition details from the intent. After this the class updates the global variable *insideGeofence* accordingly and sends a notification to the user. These notifications are shown in figure 5.1 and figure 5.2.

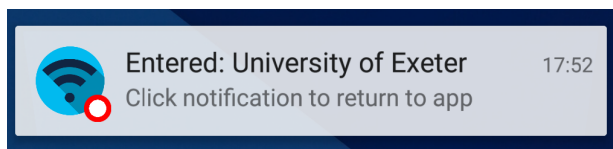


Figure 5.1. Android notification for entering the geofence.

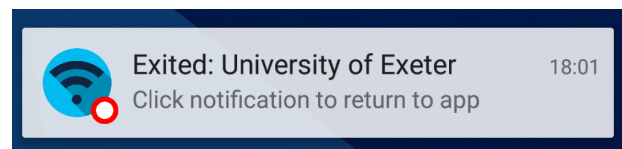


Figure 5.2. Android notification for leaving the geofence.

The *Constants* class contains all the constants values for this application. The geofence utilises this class to specify the geofence expiration, the geofence radius in meters, and the central coordinates for the geofence. The geofence expiration defines the amount of time after which the location services stop tracking the geofence. The application sets the expiration of the geofence to 48 hours. This means if the user has not used the application for two days, the application will stop checking the users location. This allows the application to deal with the scenario in which the user decides to stop gathering data. The geofence central coordinates used in this application are 50.7365 latitude and -3.5344 longitude. The radius of the geofence from the central point is set to 800 meters. This allows the application to include the whole of the University of Exeter streatham campus inside the geofence.

The user can configure different scan and send intervals for the data gathering application. These settings allow the user to specify how often application will scan for data, and how often the data is sent to the remote server. This allows the application to be robust and and configured as

needed. These settings are implemented using Androids *Preferences* which allows for state persistent preferences. In addition to the scan and send interval, the user can toggle automation of the application as well as network specific collection of data. Figure 6.1 shows the settings activity of the data gathering application. Figure 6.2 and figure 6.3 show the different intervals available to the user for scanning and sending intervals respectively.

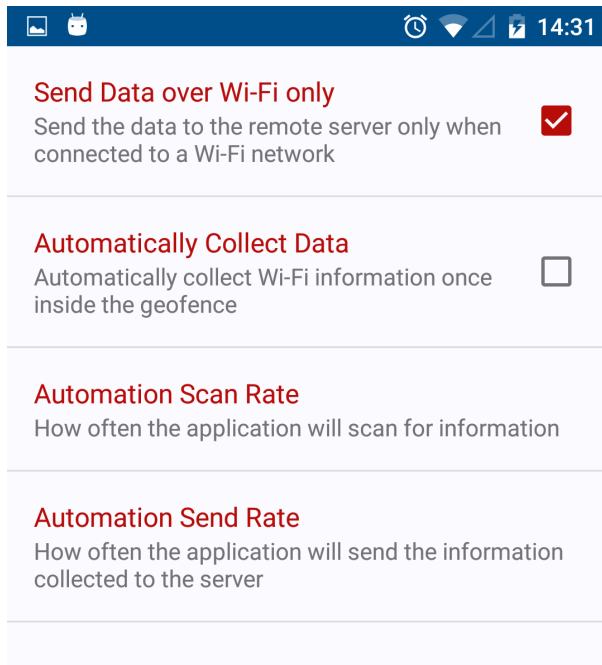


Figure 6.1. The Android applications settings screen.

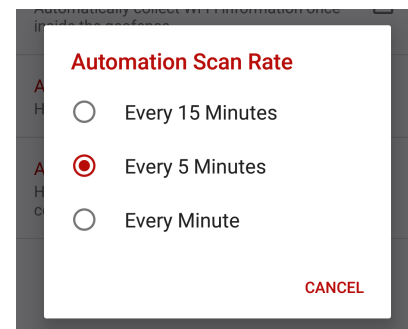


Figure 6.2. Scanning interval options

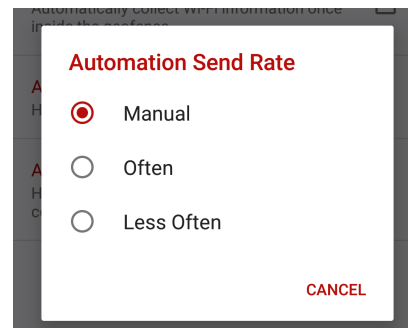


Figure 6.3. Sending interval options.
Often is every 5 Minutes.
Less Often is every 2 hours.

Automation is achieved by using services coupled with a managing class to allow for background functionality. Typically, Android applications use activity classes to specify functionality in the foreground of the application. When a task is occurring in the background, it means the application the task belongs to does not have to be open, however an instance must be created. If a task is occurring in the foreground, the application must be open and thus, have an instance already created.

When first opening the data gathering application, an instance is created. Additionally, the users settings and preferences are inspected to see if automation is enabled and to determine the intervals to autonomously scan and send. If automation is enabled, a scan and send *Intent* are created with the respective managing class. The *WiFiAutomantionManager* and *SendAutomationManager* both extend the *BroadcastReceiver* class. This allows them to manage the functionality of the applications *Intents* even when the application is in the background. This is done by building the necessary *PendingIntents* using the managing class and the current *Intent*. These are sent to the managing class at user specified intervals. The sending of these *PendingIntents* is managed by Androids *AlarmManager* class. Naturally, two instances of the *AlarmManager* class are constructed, one for scanning of the WiFi data, and the other for the sending of the collected data.

Once the managing classes receive a *PendingIntent*, the classes are responsible for executing the relevant functions and methods. The *WiFiAutomantionManager* class relies on the help from the

WiFiScanningService class for WiFi scanning methods and GPS location services. WiFi information is obtained using Androids built in *WifiManager* class. This class synergies well with an inner class, *WiFiScanReceiver*, that extends the *BroadcastReceiver* class to handle the scan results. The users location is obtained using a *GoogleAPIClient*, which has an *onLocationChanged(Location location)* method to obtain a new instance of the users location. Inside this method, the accurate, but time efficient location gathering algorithm is implemented as specified in §3.2. The application uses a accuracy criteria of five meters and a time threshold of 30 seconds when finding a location. These values are implemented as constants in the *Constants* class. Figure 7 shows the pseudocode for this algorithm, it is assumed the *GoogleAPIClient* will constantly be providing location updates until they are stopped by the algorithm. For completeness, the source code for the *onLocationChanged(Location location)* method, and the methods it incorporates are attached in the Appendix.

Algorithm 1: Accurate & Quick Location Recording

```

Input: newLocation
Result: Records an accurate location within a given time limit
Start location updates;
Time = System.CurrentTime;
if newLocation.accuracy < Accuracy Criteria then
    Record newLocation;
    Record a Time-stamp;
    Stop location updates;
else
    if (System.CurrentTime - Time) > Time Theshold then
        Record bestLocation;
        Record a Time-stamp;
        Stop location updates;
    else
        if bestLocation is not null then
            if bestLocation.Accuracy > newLocation.Accuracy then
                L bestLocation = newLocation
            else
                L bestLocation = newLocation

```

Figure 7. Pseudocode for the location gathering algorithm.

Both managing classes create instances of the *WiFiData* class, which is used to represent the collected data. The class possess the ability to encode the information to JavaScript Object Notation (JSON), such that it can be decoded by the remote sever. When encoding to JSON, a key is added to the JSON String. This key is secret, only known to the application and the remote server. This is done to introduce a degree of security when sending data to the remote server. The method the server takes to process and authenticate the information is described in §4.1

The *SendAutomationManager* class checks the users preferences before sending the collected data. Before sending the data, the network of the device is checked for an active internet connection. This is done using the *NetworkChecker* class which is able to detect if there is an active WiFi connection as well as if internet can be used. If the user has specified to be able to send data over any network, WiFi or Mobile, then only an active internet connection is tested. The data is encoded into JavaScript Object Notation (JSON), which is a a lightweight data-interchange format that is language independent and self describing. This allows the data to be easy to understand, and thus, easy to develop and use. Once encoded and there is a legal active internet connection, the data is sent to a remote server via a HTTP POST request. This is performed by using the *SendJsonDataToServer* class. Androids API level 16 insists that any network operations should be performed in parallel to the main operations of the application. Android API level 16 is the minimum target API for this application, thus the *SendJsonDataToServer* class complies with this development standard and performs an asynchronous, independent task.

The user has the ability to gather and send WiFi data manually. This is done by interacting with the two buttons presented when they launch the application. This screen the users sees when launching the application is shown in figure 8.1. This shows the two *Send* and *Scan* buttons available to the user. Pressing the *Scan* button initiates the WiFi scan and the results are displayed onto the screen as shown in figure 8.2. The WiFi information appears in a list where the user is able to scroll though by swiping up and down. The users location and the timestamp is displayed above the list of WiFi information.



Figure 8.1. Android application initial screen.

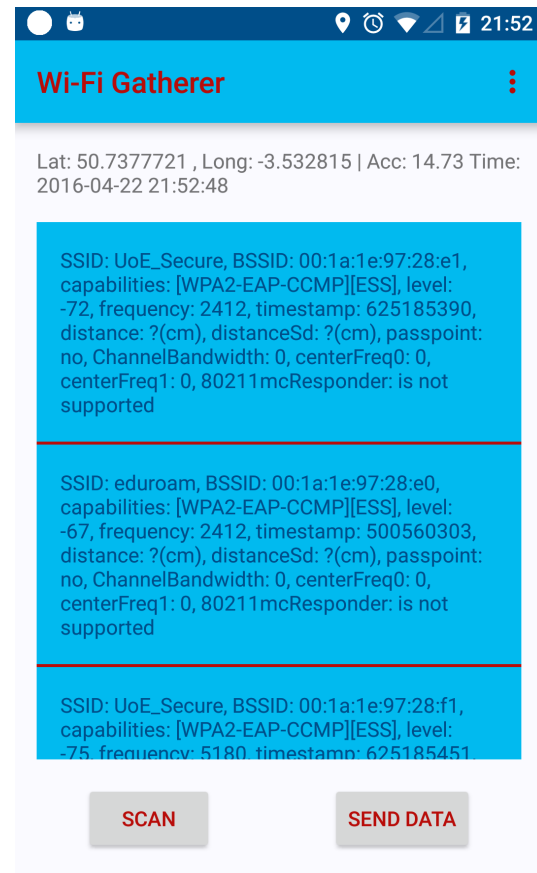


Figure 8.2. Android application WiFi data and location information.

This is implemented by calling the appropriate functionality that is used by the respective managing class whenever a button click is detected. For example, pressing the *Scan* button will register a scan button click. This as a result will execute the *scanClick(View view)* method which uses the *WiFiScanningService* class to obtain the WiFi information. The information is displayed using a *CustomListAdapter* class to achieve a uniform and organised appearance. Once the WiFi information has been obtained, the location and time-stamp is displayed using the *WiFiScanningService* classes *onLocationChanged(Location location)* method. As discussed above, the method will only update the text with increasingly accurate locations.

3.4 - Testing

Testing the application should aim to test the overall functionality and ability to gather accurate data. The application is tested on a Nexus 5 smartphone running Android (6.0 Marshmallow). The application should be able to gather a MAC address and a signal strength of an access point. The application used Androids built in WiFi scanning service, which allowed the application to be able to collect the relevant and necessary information about an access point.

Sending the JSON encoded data to the server is tested. This test aims to check how well the application handles a different network status when an autonomous or manual send is initiated. Table 2 shows the different network conditions and the results when a send is initiated³. The application is able to receive a response from the server. This confirms the application is able to send the gathered data to the server successfully when required.

WiFi	Mobile Network	Internet	Result
Yes	Yes	Yes	Send Successful
Yes	Yes	No	Unable to Send
No	Yes	Yes	Unable to Send
No	No	No	Unable to Send

Table 2. Android application network and sending testing.

The accuracy of a recorded location is obtained using Androids location services. The location service is able to return an estimated accuracy of the location the GPS calculates, which is later recorded. The data gathering application was able to gather information about 81 locations over several days. When gathering the data in an open area, such as a car park, the location criteria of five meters was reached within around 10 seconds. However, when obtaining a location in an enclosed area, such as a building, the average accuracy was around 21 meters. This is calculated as the average accuracy of all the points that were unable to meet the accuracy criteria. There are 49 locations with an accuracy of under than 10 meters, 43 of these locations have an accuracy of less than 5 meters. Overall the average accuracy of all the locations is 12 meters, with 53% of the locations meeting the accuracy criteria.

The geofence was tested with a set of known coordinates. The aim was to check if the application would recognise the users location and adjust functionality accordingly. For example, if the user was outside the geofence, the application should disable data gathering. Table 3 shows the set of locations, the expected behaviour from the application, and the observed result.

Latitude	Longitude	Inside Geofence?	Expected	Result
50.736221	-3.539412	Yes	All Functionality	As Expected
50.723482	-3.476557	No	Sending Only	As Expected
50.734443	-3.524619	Yes(On the Boarder)	All Functionality	As Expected
50.7365	-3.5344	Yes(Centre of geofence)	Al Functionality	As Expected

Table 3. Android application geofence testing.

3.5 - Results and Evaluation

The various testing performed on the data gathering application has revealed promising and expected results. Relevant WiFi information was able to be collected in addition to accurate location data. Most of the points gathered were able to meet the accuracy threshold within the specified time. Testing revealed the average accuracy of the collected points is 12 meters. This

³ Tests assume the user has preferred to only be able to send over a WiFi connection.

result is expected since the literature reveals GPS is able to achieve an accuracy of 10 meters in urban environments. However, around 53% of the locations were able to achieve an accuracy better than the expected GPS accuracy in urban areas. The location algorithm developed and implemented allows the application to be conservative with battery usage whilst being able to obtain accurate locations.

The overall developed Android application is able to perform the intended task of gathering relevant WiFi information at given locations. This can be done in an autonomous fashion or manually if desired. This application can be deployed to a handful of users to effectively build an underlying WiFi knowledge of an area. This knowledge plays a vital role in finding the WiFi geolocation of the area. The intuitive design of the application allows for easy and minimal interaction from the user.

4. Data Analysis

This section will aim describe the structure of the data gathered by the Android application as well as the design and implementation of the remote server. Additionally, this section will reveal how the gathered data can be converted to represent a data structure that is relevant and useful for processing and research.

4.1 - Server Design & Implementation

The remote server is responsible for collecting the data from the data gathering application. The servers functionality is written in PHP, a sever side programming language. The server is hosted on webspace provided by GoDaddy, which is a publicly traded internet domain registrar and web hosting company. The server is located on a linux based server accessible via the following url: <http://artiomnist.com/server/collector.php>.

The primary objective of the remote server is to record the data gathered from the Android application. The server is designed to constantly listen for a HTTP POST request. This allows the data gathering application to send data at any time with the confidence that the server will always be listening and recoding the data. Once the server receives a POST request, it attempts to decode the JSON data within the request. Once the data has been decoded, the server checks for the agreed secret key and only processes the information if the secret key is present and correct. This prevents the server processing data sent by any arbitrary POST request and ensure a basic level of security.

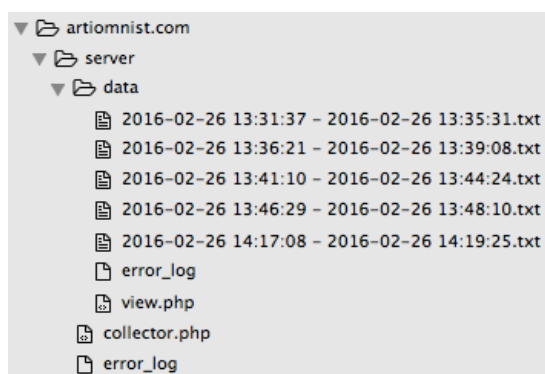


Figure 9. Remote server directory.

The data is recorded by writing the raw JSON string to a text file. Figure 9, on the left, shows the directory structure for the remote server. The file size of a data recording is dependent on the number of access points that were recorded. Each file will contain a minimum of one data location, and a maximum of 120 data locations. A total of 21 data files are used for this project. The average size of a data file that contains five data points is 13.23KB. This calculates to roughly 2.65KB per data point. This is an

acceptable size considering several points are required for the WiFi location algorithm. Once the data has been written, the file is moved to another directory and renamed to the timestamp provided by the data. This prevents any data race issues when attempting to access files that are still being written.

4.2 - Data Manipulation

4.2.1 - JSON data

The data files store the raw JSON data which contains multiple name and value pairs to represent WiFi information heard at specific geolocations. A secret key is also provided in this data for security purposes that are discussed in §4.1. The JSON data contains an array of access points which contains a list of MAC addresses and signal strengths for each access point that is audible at the location. The JSON data will also contain location information, which is represented as an array containing the latitude, longitude, and accuracy of the location. Additionally, the JSON data contains a timestamp of when this information was recorded, which helps in determining the reliability of the data. Figure 10 shows the structure for the JSON data that is stored on the server. For completeness, the appendix holds a better comprehension and visualisation of the JSON object this data provides.

```
{
  "Key": "5f4dcc3b5aa765d61d8327deb882cf99",
  "Access Points" : [
    {"MAC Address" : "00:1a:1e:97:28:e1", "Signal Strength" : -57},
    {"MAC Address" : "00:1a:1e:97:28:e2", "Signal Strength" : -10},
    {"MAC Address" : "00:1a:1e:97:28:e3", "Signal Strength" : -89},
    {"MAC Address" : "00:1a:1e:97:28:e4", "Signal Strength" : -43}
  ],
  "Location" : [{"Lat" : 50.737721, "Lng" : -3.5329606, "Acc" : 5}],
  "Time" : "2016-02-26 14:17:43"
}
```

Figure 10. Example JSON data structure.

4.2.2 - Visualisation of a data file

The encoded JSON data can be easy to understand, but when dealing with a large quantity of data other methods should be employed. For this reason, a data viewing webpage was developed to display an organised collection of the gathered data. The webpage assists when debugging the data gathering application by creating an easier way to inspect the gathered data. The webpage is populated with the filenames of the gathered data. This creates a data file listing. This is achieved by using PHP to display the contents of the server's data directory. HTML, CSS, and Bootstrap⁴ is used to create a elegant and responsive layout. By creating a responsive webpage, the data can be checked on all devices, meaning the webpage is rendered correctly and efficiently on mobile and tablet devices. This is effective when testing the data gathering application, as the data can be gathered and inspected on the same device. A section of the webpage is shown in figure 11.1. The section illustrates a row of the data file listing. A render of the webpage is also included in the Appendix.

⁴ Bootstrap is an open-source HTML, CSS, and Javascript front-end library for developing responsive projects on the web.

[Download](#)

23-02-2016 At: 15:52 Until: 15:56

Figure 11.1. A row from the the data file listing as shown on the data viewing webpage.

The webpage offers the ability to download a specific data file. This is performed by clicking the *download* link, as shown in figure 11.1 above. This is useful when inspecting specific files and gives means to obtain files for offline usage. Clicking on a data file name will display a modal which contains all the data for that data file as shown in figure 11.2. A modal is a section which displays content that temporarily blocks interactions with the main view of a website.

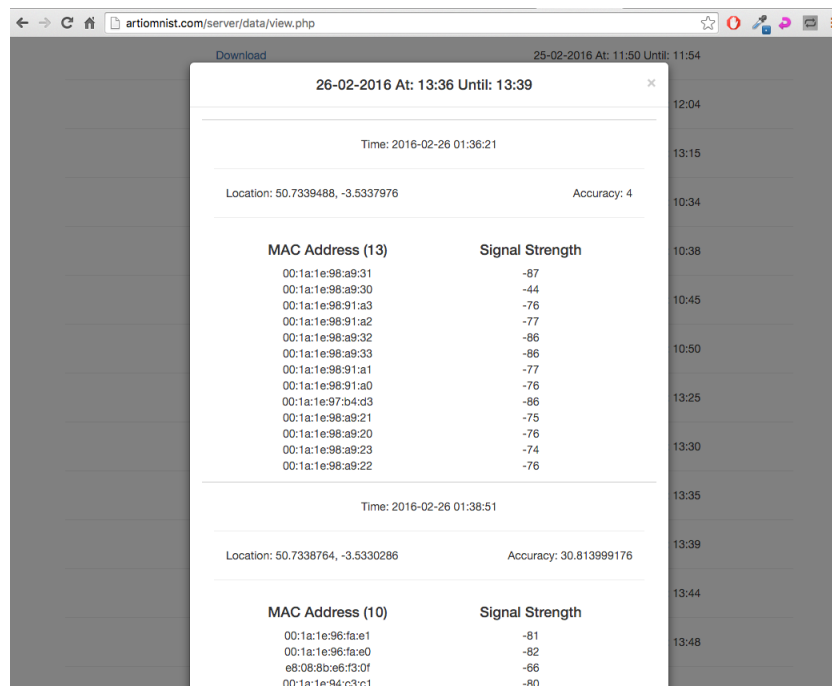


Figure 11.2. Modal showing the gathered data from a data file.

4.2.3 - Data Structures

There lies the challenging task of converting the raw gathered data into meaningful data structures that can be used in the development of the WiFi based geolocation algorithm. The first step in tackling this problem is obtaining and accessing the data files stored on the server. A trivial solution would be to download all the files either from the server or from the developed webpage. However, there exists the problem of having to re-download the files every time a new data file is written. Therefore, the files would need to be access through the internet via their respective url. The data retrieval is implemented using python, a programming language. This is done to synergise with the methodology of the research conducted in this project as described in §5.1.

The data is retrieved by crawling the developed data viewing webpage for hyperlinks. The only hyperlinks on the webpage are the download links to individual data files. The list of all links would therefore represent the url locations of all the data files. Iterating this list would enable the information stored in each data file to be retrieved. This information can be appended to an array to be further manipulated into a meaningful data structure. This approach relies on the ability to access the data viewing webpage. Therefore, it can be said that the developed webpage aids in accessing the data for the research conducted in this project.

There exist two meaningful ways the collected data can be represented; locations to what access points are audible at the locations, and at which locations a certain access point is audible. These data structures can be represented as python dictionaries. The location dictionary should assign a locations latitude and longitude coordinates as the key, and an array of access points as the values. The array should contain the MAC address and signal strength of each access point heard at that locations coordinates. The access point dictionary should assign the MAC address of the access point as the key, and an array of location coordinates as the values. The access point dictionary becomes the inverse dictionary of the locations dictionary.

Creating the locations dictionary should be an arbitrary task. This is because the structure of the JSON data is similar to the anticipated structure of the dictionary. Creating the location dictionary is achieved by extracting the location coordinates from each data entry. This represents the key for the dictionary. The access point array is extracted from the data entry and assigned as the values for the key along with the accuracy of the location and the timestamp. If a location key already exists in the dictionary the time stamp is inspected. If the timestamp is more recent, the access point array is overwritten. This ensures the locations dictionary is created with the most recent data entries.

The access points dictionary can be created by restructuring and manipulating the locations dictionary. In doing this, the access points dictionary will be consistent and up to date with the information represented in the locations dictionary. Creating the dictionary is achieved by iterating over the locations dictionary values. The list of locations is created by implementing a function which searches through the locations dictionary and returns an array of keys, which are the locations, given a MAC address. Figure 12 shows a snippet of the access points dictionary.

```
'00:1a:1e:98:82:b2' : [(-3.5333006, 50.7375589), (-3.5329606, 50.737721),
                      (-3.5329682, 50.7377185), (-3.5329566, 50.7377182),
                      (-3.5329701, 50.7377211)],
'00:1a:1e:98:8f:50' : [(-3.5341267, 50.7368377), (-3.5350102, 50.7359788),
                      (-3.5351271, 50.736039)],
'18:64:72:f9:c0:c1' : [(-3.5335885, 50.7352204), (-3.5336738, 50.7352756)],
'00:1a:1e:97:a5:33' : [(-3.5336761, 50.7341304)], ...
```

Figure 12. Code snippet from the access points dictionary.

5. Research & Experiments

This section will aim explain the experiments and research conducted on the collected data. With the help of the literature for this project, this section will describe the approach of using triangulation to obtain a WiFi based geolocation algorithm.

5.1 - Methodology

The research and experiments conducted in the project are kept separate from the development of the data gathering application. This gives more focus on the core functionality of the WiFi based geolocation algorithm. For this reason, Python and appropriate libraries are used when developing

and implementing experiments. IPython notebook is used to create an interactive computation environment for interpreting and displaying the research that are conducted on the collected data. Using IPython notebook allows the combination of code execution, rich text, mathematics, plots and rich media which provides a good platform to visualise the data and results.

5.2 - Design

The idea behind the research and experiments is to utilise the collected data to visualise the information collected by the data gathering application and develop a WiFi based tracking algorithm. This should be achieved by firstly developing a WiFi based geolocation algorithm. Thereafter, the accuracy of the algorithm should be improved with each experiment. Finally, the implementing a Kalman filter to achieve tracking. This is summarised as the specification and design in §2.2.4. Due to time constraints, only research on the WiFi based geolocation could be performed. This was due to the requirement of developing, testing and deploying an effective data gathering application. Additionally, gathering the data took several days. Once enough data was collected, time was dedicated to process and analyse the data to ensure the data was accurate and practical before starting the experiments and research.

Each key in the access points dictionary represents an access point and the values represents the locations at which the access point is audible. These location points represent a footprint of the access point. Therefore, the access points dictionary holds the list of all footprints for where all access points are audible. A footprint represents an area where an access point is audible. The set of location points should form a footprint that is a convex hull. This footprint can be computed using a Delaunay triangulation approach. The Delaunay triangulation of a set of points P is distinguished by having no point in the set that lies in the interior of any triangles circumscribing disk [de Berg et al., 2008]. This ensure the triangulation is created with triangles that are as equilateral as possible. Galton and Duckham (2006) discuss the Delaunay triangulation approach to computing the footprint of a set of points. Their method aims to trim the boundary edges in decreasing order of length such to produce a final shape that contains all inputs and is both regular and has a simple closed curve, also known as a Jordan curve.

To represent a region that is occupied by the points at a certain granularity, it is natural to suppose that each point has an area of effect. This region can be represented by the aggregation of all the areas of effect of the points in the set [Galton and Duckham, 2006]. Therefore the points themselves would only lie inside the region, meaning no one point is on the perimeter. This region can be called an extended footprint and in this project it can represent the accuracy of the recorded location. Galton and Duckham (2006) also discuss the method of generating an extended footprint, and pose the question of how far the footprint is to be extended into its outer space. To represent the accuracy for a gathered location, the footprints should be extended proportionately to the accuracy of the location. This means if a gathered locations accuracy is 20 meters, it's footprint should be extended much further than if it had an accuracy of just four meters. This would represent the extended footprint as closely as possible to the accuracy of the data the WiFi based geolocation depends upon to produce a location area estimate.

By computing the intersection of the convex hulls that represent an access point's footprint, an estimated location area can be found, given the access points that are audible. This can be done by using a Delaunay triangulation method to create the footprints and use the simplices and vertices to compute the intersecting simplex. The simplex can be made of multiple simplices, thus

forming a polygon. If an intersecting simplex cannot be found, the footprint of the common vertices should be inspected. A possible approach would be to increase the resolution of the triangulation by adding new points to the total set of points within the footprint of the intersecting vertices. This will increase the number of points inside the triangulation which can result in an intersecting simplex. The estimated location should be displayed on a map image in order to understand and visualise the result.

5.3 - Implementation

The data structures created from the collected data as discussed in §4.2.3 should be used. This is implemented by importing the python file which is able to create the data structures. The file contains necessary functions for obtaining the information and creating the data structures. This creates a platform for future work and developments. The gathered data is plotted onto a map image to visualise the data points. This is implemented using the *matplotlib* library as well as a *mpl_toolkit basemap* object. These libraries allow for the plotting of geolocation coordinates onto map projections. Figure 13 shows an area of the plotted locations that were collected by the data gathering application. The full map image which contains 81 location points can be found in the Appendix. The map image for plotting the location information is obtained from OpenStreetMap, which holds open source mapping data.

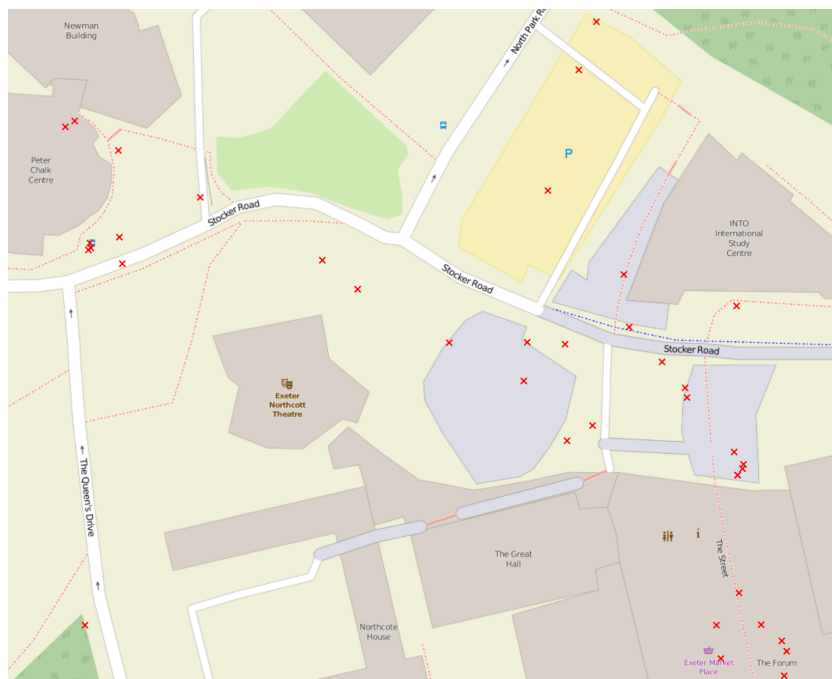


Figure 13. Locations collected by the data gathering application.

Visualising the locations where an access point is audible can be achieved using a similar method. Processing the access points dictionary data structure revealed the information might be difficult to represent and understand. This is because from the 81 gathered locations, a total of 1182 unique access points were heard, 500 of which were audible in three or more locations. These 500 access points are of interest for our triangulation approach to a WiFi based geolocation algorithm. This is because it requires a minimum of three points to form a polygon that can represent a location area. Whilst the number of points to plot onto the map remain the same, the number of colours necessary to represent each access point is equal to the number of unique access points with three or more locations. Here lies the limitation and problem of representing where all the unique access points are audible, but representing a smaller number of access points

is feasible and can allow us to grasp a basic understanding of the data. Figure 14 shows five unique access points and the locations at which they are audible.

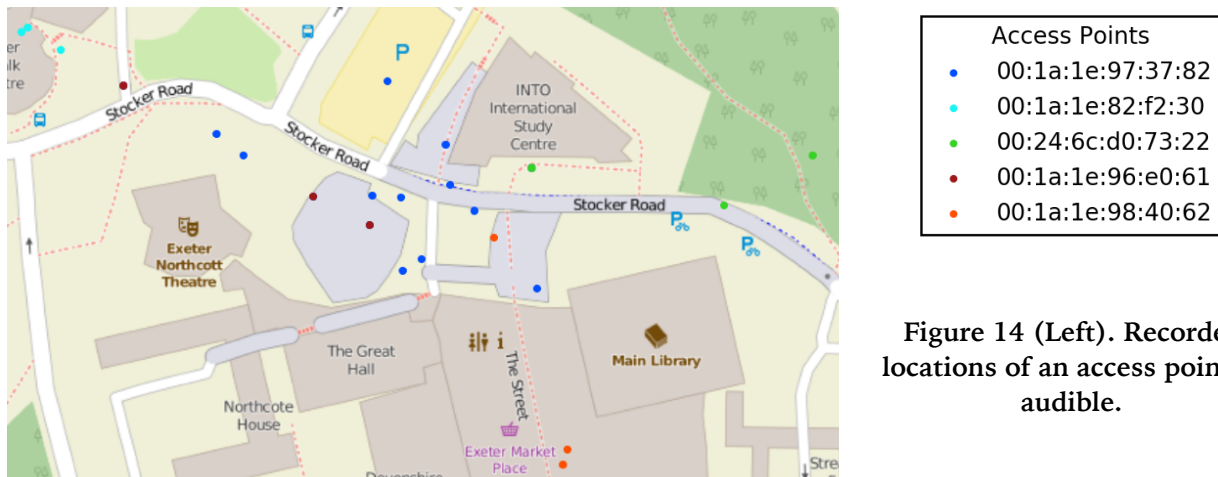


Figure 14 (Left). Recorded locations of an access point is audible.

The *SciPy* library is used to achieve a Delaunay triangulation. *SciPy* provides a collection of open source software for scientific computation in python. The Delaunay method takes a set of points and returns a triangulation object that contains the attributes that detail the triangles (simplices) and points of the Delaunay triangulation. This *SciPy* Delaunay triangulation method is an interface for the *Qhull* library. The triangulation approach to the WiFi based location algorithm is first developed and tested with a known set of points before being applied to the collected data. This creates an simpler approach to developing, debugging and testing the algorithm. Figure 15 shows the triangulation of a set of test points. The green squares represent the locations where access point A is audible, and the red circles represent the locations where access point B is audible. The blue squares represent the locations where both access points are audible.

To find the location area where both access points are audible the algorithm needs to obtain the triangles which contain vertices of the common points. This can be seen as finding the triangles which intersect the two access points. This is first achieved by obtaining the list of common points given which access points are audible. The common points can then be used to obtain the triangles that contain all vertices belonging to the set of common points. Figure 16 shows the location area when both access points are audible.

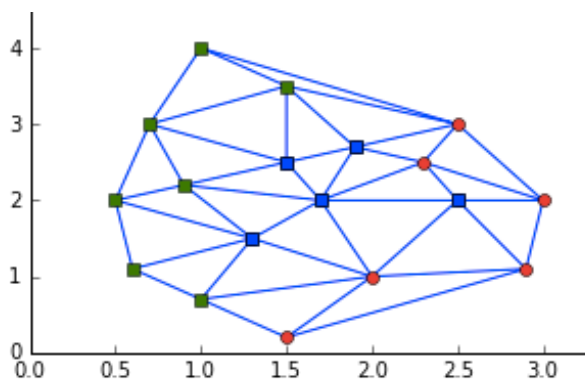


Figure 15. Locations collected by the data gathering application.

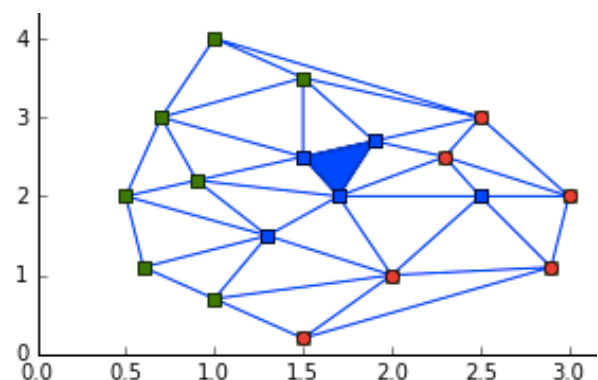


Figure 16. The estimated location area if both access points are audible.

This approach is limited to only finding triangles where all three vertices are common points. Therefore, the accuracy of the algorithm can be improved by increasing the resolution of the triangulation at the common points. This is achieved by introducing new vertices into the

triangulation and re-computing the Delaunay triangulation for the new set of points. The new vertices are only introduced for triangles which contain two or more common points. The vertices are introduced in the middle of each edge. If the two vertices are common points, the new point is marked as a common point. The location of the new point is added to both access points audible location list. However, if only one vertex is a common point, the new point is marked as the access point which the other vertex belongs to. For example, if one vertex, X , is a common point, and the other vertex, Y , represents access point B , the new point is added at the middle of the edge which connects vertex X and Y . This new point is then marked as only representing access point B .

Once the new triangulation is computed the algorithm performs another attempt at finding triangles with vertices which are common points. If the algorithm is unable to obtain an intersecting area, it returns the union area of the audible access points. Figure 17 shows the location area when both access points are audible. It should be mentioned that for this figure a common point was removed from one of the access points.

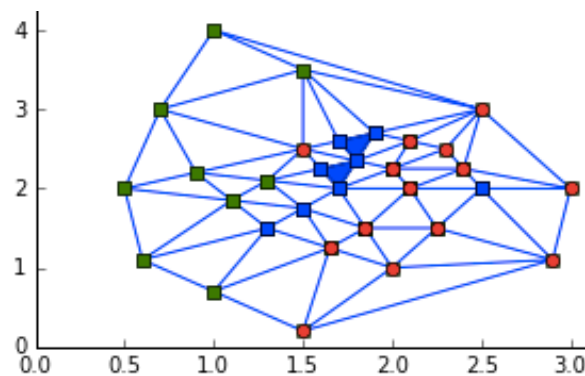


Figure 17. The estimated location area with increased accuracy if both access points are audible.

5.4 - Testing

Testing the triangulation approach to a WiFi based geolocation can be done by applying the algorithm to the collected data and testing various access points. The Delaunay triangulation for all gathered data points is shown in the Appendix. The access points for a certain location can be found by querying the data locations dictionary. Searching the data structure revealed the main access points that are audible in the Harrison building are 00:24:6c:d0:7a:43, 00:1a:1e:98:90:41, 00:1a:1e:98:84:52. Using these access points as the input for the WiFi based geolocation revealed the location area shown in figure 18.1.

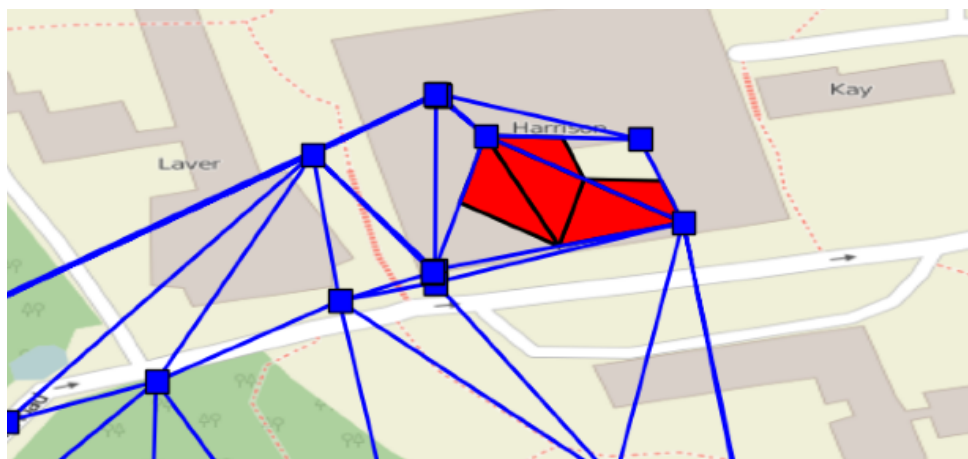


Figure 18.1. The estimated location area when Harrison access points are audible.

Figure 18.2 shows the estimated location area for a single access point at the centre of the University of Exeter. by adding more access points that are audible, the accuracy of the location should improve. The improvement in accuracy should result in a smaller estimated location area. Figure 18.3 shows this area when a two more access points are added to the input.

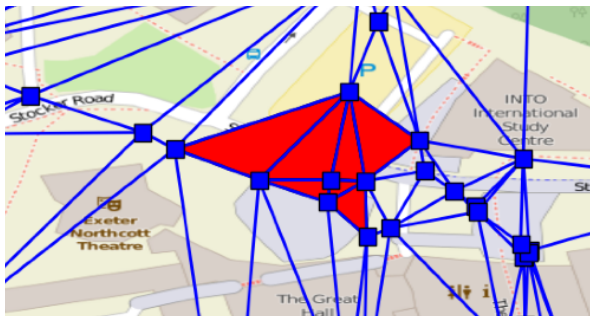


Figure 18.2. The estimated location area when only one central access point is audible.

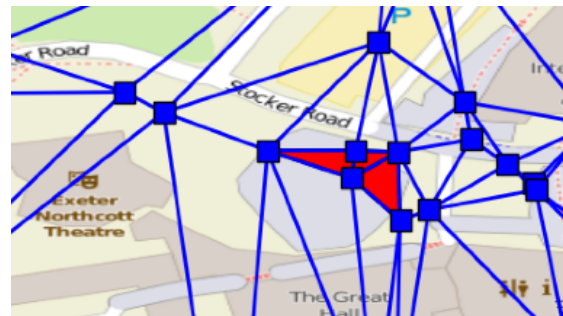


Figure 18.3. The estimated location area when only three central access point are audible.

Testing the accuracy of the experiment can be done by removing one of the location coordinates from the data structures. The access points of the removed location can be entered as the audible access points for the WiFi based geolocation algorithm. The estimated location area can be compared to the known coordinates from the removed location. The location with the latitude and longitude coordinates 50.7372425 and -3.5329765 respectively was removed from the data. The list of access points audible at that location is used as the input for the WiFi based geolocation algorithm. Figure 19.1 shows the estimated location area that the algorithm produced, and figure 19.2 shows a map marker of the location.



Figure 19.1. The estimated location area of the location with coordinates 50.7372425 and -3.5329765. (Latitude and Longitude)

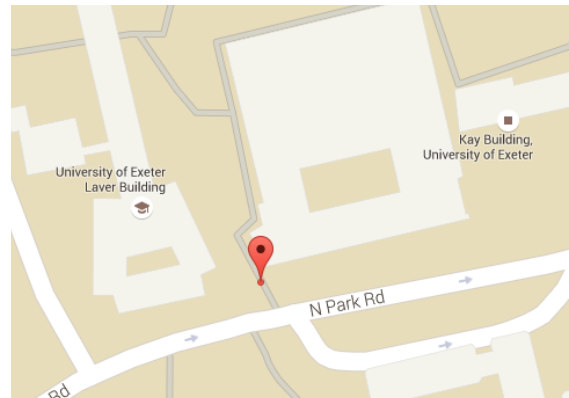


Figure 19.2. The location of coordinates 50.7372425 and -3.5329765. (Latitude and Longitude).

5.5 - Results and Evaluation

The tests revealed the triangulation approach to a WiFi based geolocation is able to produce an estimated location area. The tests revealed that adding more access points that are audible at a location often produced a smaller location area which correlates to an increase in accuracy. When obtaining an estimated geolocation area inside a building the area produced included a section of the building. Whilst this result seems to be inaccurate, it is expected. This is because the accuracy of the recorded data locations that are gathered in enclosing areas, such as buildings, is on average

around 21 meters. The results show that the gathered inaccuracies can propagate to the WiFi based location algorithm as anticipated.

The literature dictates the WiFi based location algorithm relies on underlying knowledge of the area. Therefore, it is correct to predict and assume that more location points the data gathering application is able to collect, the better the WiFi based algorithm will perform in terms of accuracy. However, collecting too much information can pose inaccuracies and contradictions within the information. This creates the need to effectively manage the reliability and freshness of the information that is gathered. This is achieved by comparing the time stamp when creating the data structures as discussed in §4.2.3.

6. Critical Assessment of The Project as a Whole

This research project was a challenging task which seemed daunting at first, however it is the complexity and challenge the project offers which made the project enjoyable and interesting. The scale of the project demanded knowledge in different computing areas. This is apparent from developing an Android application, implementing a server, analysing the data and conducting various experiments on the data. This resulted in the scale of the project to become larger than anticipated, and therefore, time was a key limitation to this project. Combining this with the work requirements of a third year at university made time a very sparse resource.

6.1 - Positives

The literature surrounding this project is just beginning to be published. This is encouraging because this project is utilising modern technologies and methods that are described in recent literatures. Additionally, the literature available gives insight to how others approach the problem. This is good because it inspired solutions as well as possibilities for research.

The development of the Android application was conveniently time with the continuous assessments from the ECM3424, Mobile and ubiquitous computing module. Having no prior android development knowledge the module enabled me to undertake the challenge of developing the data gathering application. The assessments where set during the design of the android application. This enabled me to understand how the application could be implemented, and gave me confidence in developing a working application.

Already owning the webspace for remote server was helpful for this project. Having previous working experience of PHP allowed me to easily and confidently develop the remote server. By owning private webspace, the implementation of the remote server came naturally, and allowed for a simple, debugging, development and testing process.

The many discussions with my supervisor created various ideas for methods to undertake for obtaining a WiFi geolocation. These debates occurred during the weekly project meeting, which were beneficial in keeping this project headed in the right direction. The discussions provided initial grounds for research conducted in this project.

6.2 - Negatives

The definite negative to this project was not having enough time to devote to the research and experiments in this project. The time limitation meant the project was not able achieve a tracking element to this project. This time limitation was unavoidable because of the necessary work required to develop the data gathering application and analysis the collected data.

Another negative to this project was only gathering data with one device. This proved to be tedious when aiming to collection a large amount of data. Deploying the data gathering application for crowd sourced data would be ideal. If this was the case this research project would innately contain a user focus, and the project would become more of a software engineering undertaking. This would also create a stricter time limitation on the overall project by further increasing the scale of the project.

If this project was done again, I would do several things differently. Preferably, this project would be started with an already developed data gathering application as well as a means to covert the data to its appropriate data structures. This would allow me give more focus to the quality and quantity of the research for the project. I would use multiple devices to effectively gather a larger data set for the research. As a result this would give me more time to conduct research into the WiFi based geolocation algorithm and implement interpolation and tracking algorithms to improve the accuracy a WiFi based geolocation.

7. Future work

This project only begins to uncover the possibilities of obtaining a geolocation using WiFi signals. Various techniques and theories exist that can be coupled with the vast amount of WiFi information that is available in urban areas to investigate different techniques and methods that can be used in finding a WiFi based geolocation. Additional research and experiments on increasing the accuracy WiFi based geolocation algorithm is an possible area of future work. This work would aim to use existing theories and techniques to either further increase the accuracy of the WiFi based location, or implement a algorithm which is more efficient in computing the WiFi based geolocation. Such work would include investigating various interpolation methods that incorporate the signal strength data to improve the accuracy of the result. Additionally, experiments that use Kalman filters to implement tracking and further improve the accuracy of the WiFi based geolocation can be considered areas of future work.

Areas of future work do not have to focus on improving the accuracy of the estimated location. Instead, the robustness of the WiFi based geolocation algorithm can investigated. For example, implementing a scoring system which attributes a rank to a triangle in a given triangulation where a certain access point is heard can generate a heat map of the estimated location area. This would make the algorithm robust as it can cope with missing access points and represents the probability of the WiFi based geolocation being in a certain section of the triangulation.

Once enough research and experiments are conducted on the collected data, this project should become a software engineering venture. The project should aim to create a product, possibly in the form of a smartphone application, which replays the access points that it can hear to the

developed WiFi based geolocation algorithm. The algorithm can then return a location estimate based off the data that is gathered for the surrounding area.

8. Conclusion

In conclusion a triangulation approach to finding a WiFi based geolocation is an effective method. The underlying knowledge of an area that is required when attempting to find a WiFi based geolocation can be collected using a smartphone. An Android application was developed to achieve effective data gathering inside a specified geofence. Data is recored and sent to a remote server that is able to store the data for the research and experiments. From the collected data, two main data structures are apparent and relevant for research. These are the locations to which access points are audible and access point to at which locations it is audible. Visualising the collected data can be done by plotting the points into a map, however, plotting the locations at which unique access points are audible can prove to be difficult to represent because of the high number of unique access points in urban areas.

The approach taken to finding a WiFi based geolocation involves representing a unique access point as a footprint of an estimated location. These footprints are convex hulls, and the intersection of the footprints represents the estimated location area of the audible access points. Computing the intersection of the convex hulls can be computationally expensive and difficult, but given the nature of the collected data a triangulation approach can be used to find the intersecting polygon of the convex hulls and ultimately the footprint which they represent. For this reason, it is confident to say that a triangulation approach to a WiFi based geolocation is an effective method to find a location estimate given a set of WiFi data points.

While it is unfortunate that not all of this project could be attempted, there is still research and future work available for this project. This future work should primarily aim to improve upon the triangulation approach to estimating a WiFi based geolocation. The work should demonstrate an improvement in the accuracy of the estimated geolocation, or propose an more efficient approach to the WiFi based geolocation algorithm. Given an accurate enough WiFi base geolocation is developed, this project can become a software engineering project, which aims to produce a product that is capable of tracking a users location given a set of WiFi information points.

9. Acknowledgements

Special thanks must be made to Professor Richard Everson for supervising this project. His guidance and mentorship gave this project confidence to head in the right direction. The valuable discussions and debates about the project went a long way in shaping and tackling the complex challenges posed by the project.

References

- Android API. (n.d.). Manifest permission | Android Developers. [online]. [developer.android.com. Available at http://developer.android.com/reference/android/Manifest.permission.html#ACCESS_WIFI_STATE](http://developer.android.com/reference/android/Manifest.permission.html#ACCESS_WIFI_STATE) [Accessed: 17 Apr. 2016].
- Chen, Y. and Kobayashi, H. (2002). Signal strength based indoor geolocation. 2002 IEEE International Conference on Communications. Conference Proceedings. ICC 2002 (Cat. No. 02CH37333), 1, pp.436-439. doi: 10.1109/ICC.2002.996891
- Danalet, A., Farooq, B. and Bierlaire, M. (2014). A Bayesian approach to detect pedestrian destination-sequences from WiFi signatures. Transportation Research Part C: Emerging Technologies, [online] 44, pp.146-170. doi: 10.1016/j.trc.2014.03.015. Available at: <http://www.sciencedirect.com/science/article/pii/S0968090X14000874> [Accessed 12 Apr. 2016].
- de Berg, M., Chewning, O., van Kreveld, M. and Overmars, M. (2008). Computational geometry: Algorithms and Applications. 3rd ed. Berlin: Springer-Verlag.
- Developer.android.com. (2016). Dashboards | Android Developers. [online] Available at: <http://developer.android.com/about/dashboards/index.html> [Accessed 21 Apr. 2016].
- Galton, A. and Duckham, M. (2006) in *Raubal, M et al(Eds):. Geographic information science*, LNCS 4197. Berlin: Springer. pp. 81-98.
- IDC. (2015). International Data Corporation, (2015). *IDC: Smartphone OS Market Share*. [online] Available at: <http://www.idc.com/prodserv/smartphone-os-market-share.jsp> [Accessed 10 Nov. 2015].
- Kay, A. W. and Schuessler, N. (2009). Map-matching of GPS traces on high-resolution navigation networks using the Multiple Hypothesis Technique (MHT). pp.2-6
- Kawaguchi, N. (2009). WiFi Location Information System for Both Indoors and Outdoors. In: Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living. Springer; 2009. p. 638–645.
- Limonard, C. (2013). What Drains The Smart-Phone Battery?. [online] [Electronicdesign.com. Available at: http://electronicdesign.com/ed-europe/what-drains-smart-phone-battery](http://electronicdesign.com/ed-europe/what-drains-smart-phone-battery) [Accessed 13 Apr. 2016].
- Paek, J., Kim, J. and Govindan, R. (2010). Energy-efficient rate-adaptive GPS-based positioning for smartphones. Proceedings of the 8th international conference on Mobile systems, applications, and services - MobiSys '10. [online] pp.299-314. doi: 10.1145/1814433.1814463. Available at: <http://dl.acm.org/citation.cfm?id=1814463> [Accessed 16 Apr. 2016].
- Sapiezynski P, Stopczynski A, Gatej R, Lehmann S (2015) Tracking Human Mobility Using WiFi Signals. PLoS ONE 10(7): e0130824. doi:10.1371/journal.pone.0130824. pp.1-22.

Sharkey, J. (2009). Presentation, San Fransisco, California Google I|O 2009.

Tateishi, R. and Wen, C. (1994). Relationship between Root Mean Square Error and Probable Error. Journal of the Japan society of photogrammetry and remote sensing, 33(4), pp.15-22.

The Economist. *Planet of the phones*. (2015). The Economist, (Vol. 414. 8927), p.9.

Webster, P. and Webster, H. (2013). The use of GPS devices and smart phones as navigation aids. [online] The Mountaineering Council of Scotland. Available at: <http://www.mcofs.org.uk/navigation-gpsandsmartphones.asp> [Accessed 13 Apr. 2016].

Wing, M., Eklund, A. and Kellogg, L. (2005). Consumer-Grade Global Positioning System (GPS) Accuracy and Reliability. Journal of Forestry, 103(4), pp.169-175.

Appendix

The `onLocationChanged(Location location)` method source for collecting an accurate location within a given time limit is shown below.

```
/**
 * This Method attempts to obtain a location that meets the accuracy criteria within
 * the time limit. If the location is accurate enough or the too much time has elapsed
 * location variable is updated and saved along with a time stamp.
 */
@Override
public void onLocationChanged(Location location) {

    // Check The location accuracy and make sure the new location is different to the
    // previous location.
    if (location.getAccuracy() < Constants.ACCEPTABLE_ACCURACY
        && location != currentLocation
        && location != MainActivity.currentLocation) {

        // Record the new location and a time stamp.
        currentLocation = location;

        timestamp = android.text.format.DateFormat.format("yyyy-MM-dd HH:mm:ss",
            new java.util.Date()).toString();

        // Update what the user can see on screen.
        locationText = updateDisplayMessage(currentLocation, timestamp);

        // Stop location updates to turn off GPS usage.
        stopLocationUpdates();

        // Display a confirmation message to the user.
        Toast.makeText(this, "Completed: Scanning", Toast.LENGTH_SHORT).show();

        // Save/Record the new data.
        saveInfo();

        // Update the global variables as well as reset the current location.
        MainActivity.currentLocation = currentLocation;
        MainActivity.timestamp = timestamp;
        currentLocation = null;
    } else {

        // Location has not met the accuracy criteria. Check if enough time has
        // elapsed.

        if ((System.currentTimeMillis() - time) > Constants.ACCEPTABLE_TIME) {

            // Record a time stamp.
            timestamp = android.text.format.DateFormat.format("yyyy-MM-dd HH:mm:ss",
                new java.util.Date()).toString();

            // Update what the user can see on screen.
            locationText = updateDisplayMessage(currentLocation, timestamp);

            // Stop location updates to turn off GPS usage.
            stopLocationUpdates();

            // Display a confirmation message to the user.
            Toast.makeText(this, "Completed: Scanning", Toast.LENGTH_SHORT).show();
        }
    }
}
```

```

        // Save/Record the new data.
        saveInfo();

        // Update the global variables as well as reset the current location.
        MainActivity.currentLocation = currentLocation;
        MainActivity.timestamp = timestamp;
        currentLocation = null;

    } else {
        // Still within the Time threshold. Update the location if accuracy is better.

        // Check the current best accuracy against the new locations accuracy.
        if (currentLocation != null) {
            if (currentLocation.getAccuracy() > location.getAccuracy()) {
                currentLocation = location;
            } // else there is no need to update.
        } else {
            // The case where the First location the GoogleAPIClient obtains is not
            // accurate enough.
            currentLocation = location;
        }
    }
}
}
}
}

```

Source code for the methods used in the `onLocationChanged(Location location)` method is shown below.

```

/**
 * This Method creates a display message string depending on a location and timestamp
 *
 * @param loc - The Location object containing a Latitude and Longitude.
 * @param timestamp - The String containing
 * @return String showing the coordiantes, their accuracy and the timestamp for that
 * location.
 */
private String updateDisplayMessage(Location loc, String timestamp) {

    String displayCords = "Lat: " + currentLocation.getLatitude() +
        " , Long: " + currentLocation.getLongitude();

    String accuracy = "Acc: " + currentLocation.getAccuracy();

    return displayCords + " | " + accuracy + " Time: " + timestamp;
}

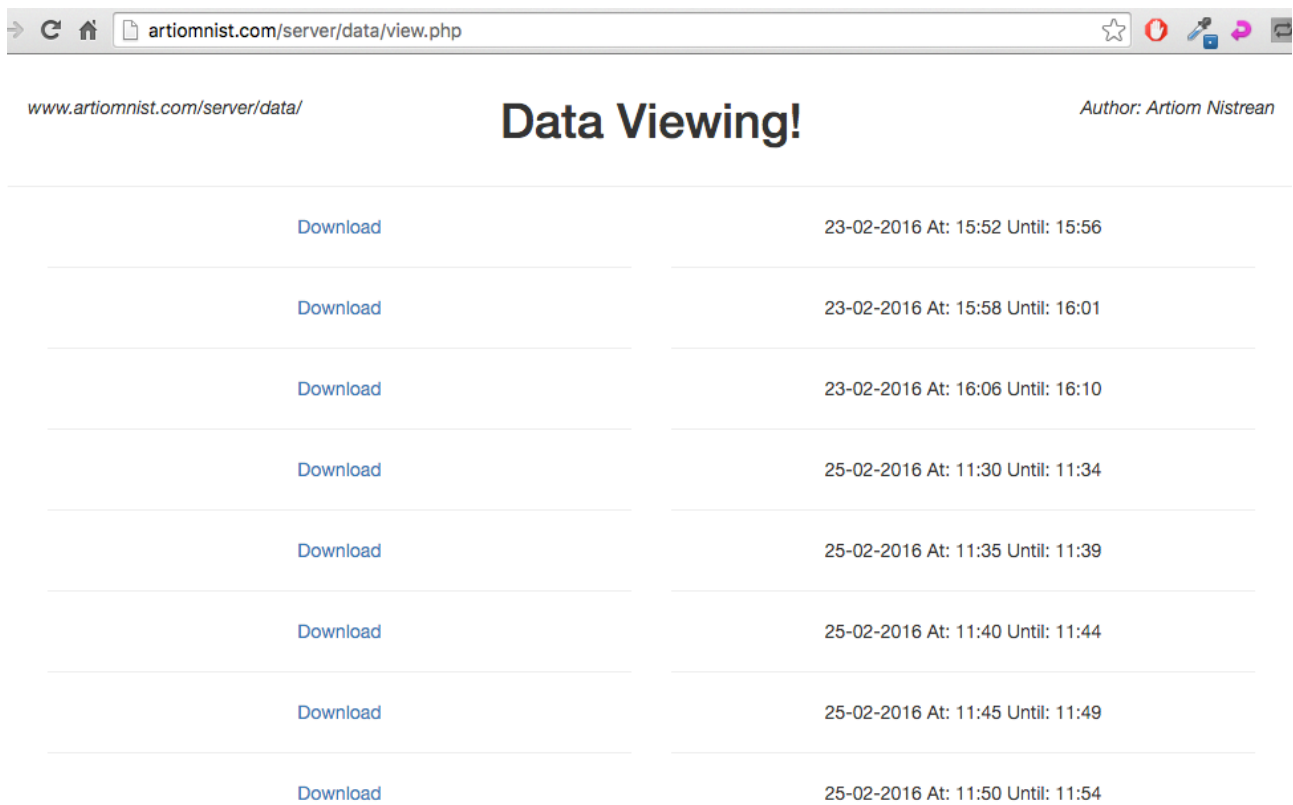
/**
 * This method saves the current recorded information to the global wifis arraylist
 * which holds all the current recorded information.
 *
 */
private void saveInfo() {
    // Make sure the wifis list is not null.
    if (MainActivity.wifis != null) {
        // Make sure there is information to be recorded.
        if (!MainActivity.wifis.isEmpty() && currentLocation != null
            && !timestamp.isEmpty()) {
            WifiData wifiData = new WifiData(MainActivity.wifis, currentLocation,
                timestamp);
            collectedInformation.add(wifiData);
        }
    }
}
}

```

The visualisation of the JSON object provided by the gathered data is shown below.

```
▼ object {4}
  Key : 5f4dcc3b5aa765d61d8327deb882cf99
  ▼ Access Points [4]
    ▼ 0 {2}
      MAC Address : 00:1a:1e:97:28:e1
      Signal Strength : -57
    ▼ 1 {2}
      MAC Address : 00:1a:1e:97:28:e2
      Signal Strength : -10
    ▼ 2 {2}
      MAC Address : 00:1a:1e:97:28:e3
      Signal Strength : -89
    ▼ 3 {2}
      MAC Address : 00:1a:1e:97:28:e4
      Signal Strength : -43
  ▼ Location [1]
    ▼ 0 {3}
      Lat : 50.737721
      Lng : -3.5329606
      Acc : 5
  Time : 2016-02-26 14:17:43
```

The render of the data viewing webpage is shown below.



Download	23-02-2016 At: 15:52 Until: 15:56
Download	23-02-2016 At: 15:58 Until: 16:01
Download	23-02-2016 At: 16:06 Until: 16:10
Download	25-02-2016 At: 11:30 Until: 11:34
Download	25-02-2016 At: 11:35 Until: 11:39
Download	25-02-2016 At: 11:40 Until: 11:44
Download	25-02-2016 At: 11:45 Until: 11:49
Download	25-02-2016 At: 11:50 Until: 11:54

The plot of all the 81 data points collected by the data gathering application is shown below.



The Delaunay triangulation of all the gathered data points is shown below.

