# ASHESI UNIVERSITY

## IMPROVING ACCURACY OF GPS SIGNALS FOR USE IN LAND SURVEYING APPLICATION

**CAPSTONE PROJECCT**

**B.Sc. Computer Engineering**

**Oswald Gyabaah**

**2020**

**ASHESI UNIVERSITY**

# IMPROVING ACCURACY OF GPS SIGNALS FOR USE IN LAND SURVEYING APPLICATION

**CAPSTONE PROJECT**

Capstone Project submitted to the Department of Engineering, Ashesi University in partial fulfilment of the requirements for the award of a Bachelor of Science degree in Computer Engineering

Oswald Gyabaah

2020

# DECLARATION

I hereby declare that this capstone is the result of my own original work and that no part of it has been submitted for another degree in this University or elsewhere.

Candidate's Signature:

……...……

Candidate's Name:

……...……

Date:

……...

I hereby declare that the preparation and presentation of this capstone were supervised in accordance with the guidelines on supervision of capstone laid down by Ashesi University.

Supervisor's Signature:

……

Supervisor's Name:

……

# Acknowledgments

To all who have been of great support to me on this journey, I am grateful.

## Abstract

Location based services applications are very common in our lives today. Services such as uber rely on them to deliver their core products. If a service does not require a very strict accuracy level, it is generally considered enough to use google maps with little or no improvement at all. However, to apply this in finding accurate locations under a  meter, we need to do some improvement on the GPS data we receive to be able to pinpoint the location as needed.

This project examines the deficiencies of google maps and come up with algorithm that combines various sensors in mobile devices to stabilise and improve the accuracy of the coordinates received. These include the accelerometer, gyroscope, and compass. The combination allows us to make an independent computation of direction and distance to decide if the location provided by location services is accurate.

We develop a mobile application to try out the algorithm against traditional google maps. The algorithm was able to provide great consistence where google maps locations were fluctuating. It also able to trace a path straighter than google maps could.

# 1 Introduction

## 1.1 Background

Many applications of **Location-based services** have gained popularity with burst of mobile devices. So far, the most reliable system for Location-based services is the Global Positioning System (GPS) which operates on a set of satellites orbiting the earth to provide approximate locations for bodies on the surface of the earth. This location is provided as a pair of longitude – latitude pairs. The locations received may be accurate to a few meters for outdoors applications with directly of sight to the satellites but may be off by as much as fifty meters in the presence obstructions like tall building or forests [1]. While this provides enough information to locate outdoors targets and with ease, such applications are usually not required to be accurate to a meter or better. For instance, when using GPS to locate a building, the coordinates could point to any location in or close to the building without affecting the ones ability to find it. This is because people use their intuition and senses to make up for any inaccuracies. Similarly, applications that provide routing information do are not require to be accurate to the a better if the route is in the same direction as the nearest road and runs parallel, people are smart enough to find the intended road.

However, this is not the case when one needs to pinpoint a location to the nearest meter or even better accuracy. For applications like demarcation of land boarders, while an error margin of 1 meter or less can be tolerated, anything worse than that would make the borders ambiguous and result in disputes therefore, for such applications one needs to come up with accuracies in the acceptable range of 1 meter of better. For this reason, mobile platform Location-based services applications have not yet emerged that allows the individual to

demarcate land borders on their own. According to [1], the GPS chips used in mobile devices are not powerful enough to extract locations at better accuracies than 5 meters.

## 1.2 Problem Definition

GPS signals received in open spaces could be as accurate as 5 meters or off by a slight margin. Because this accuracy is still far from that needed to pinpoint the specific locations, we haven't been able to develop Location-based services mobile applications that can be installed on mobile device and used as standalone surveying applications. Since the GPS signals themselves are not going to improve significantly soon, we need another way of correcting those signals once received on a mobile device. When this improvement is significant enough, we will then be able to build Location-based services applications for surveying.

## 1.3 Objectives

To enable easy building of Location-based services applications for pinpointing locations and for border demarcation, we need to first focus on reducing the error margins of GPS signals. The purpose of this project is to find a means of improving the accuracy of said signals in way that allows developers to build surveying and pinpointing Location-based services applications. Because this is targeted at individuals, the goal is to achieve this accuracy preferably without any external peripherals and without introducing additional cost to the individual. We will have to investigate mobile devices for inbuilt sensors that can be leveraged for this purpose.

## 1.4 Expected Outcome

At the end of this project, a set of inbuilt sensors in mobile devices that can be leveraged will be identified and characterised according to their contributions to our goal of improving GPS accuracy. An algorithm that will leverage the outputs of such sensors to provide the

correction will also be developed. A mobile application implementing this algorithm will also be developed as a prototype.

## 1.5 Scope

The scope of the is project includes the design, development and implementation of a prototype of a mobile application with inbuilt built ability to reduce the error margins of normal GPS signals. It serves as proof concept and a guide for anyone who may want to develop an application based on the accuracies that we will come up with. It does not include extended applications such as application in pinpointing or surveying but serves to prove that my approach can be used to develop such an application.

## 1.6 Motivation

The original intent of this project was to provide a digital, commercial mapping and surveying solutions to the individual. However, it quickly came to light that this was not possible until an acceptable margin of error for the GPS signal receive could be achieved. The project was therefore put on hold to find a solution to the problem of GPS before moving on to build the commercial product.

# 2 Literature Review and Related work

In this chapter, scholarly works in this field, based on which a design and prototype of this system can be developed. This includes any models or relations they may point to.

## 2.1 Related Work

Huang and Yu in [1] chronicle the wide use of GPS in the development of Location-based services applications. They attribute this to the reduction in the size of the GPS receivers and the integration of GPS receiving chips into mobile handsets by all manufacturers. This the has enabled the development of Location-based services applications for use on mobile devices. They, however, admit that the accuracy of the GPS information received on cell phones are fraught with wide margins owing the low-cost options for the GPS chips used in the devices. They then propose an algorithm for localisation improvements using data requested from inbuilt gyroscope and compass sensors. The algorithm works on a simple principle, since the GPS information contains directions, altitude and other parameters that help locate the device, and these are all accessible via sensors built into the device, the application collects sensor readings from these sensors and compute the actual direction in which the device is moving. The distance moved is then computed from the GPS which is expected to be larger than the actual distance moved. To estimate the actual distance moved, knowing the actual direction of the movement, they estimate the path by using a trigonometric function as shown in the image below[1]:

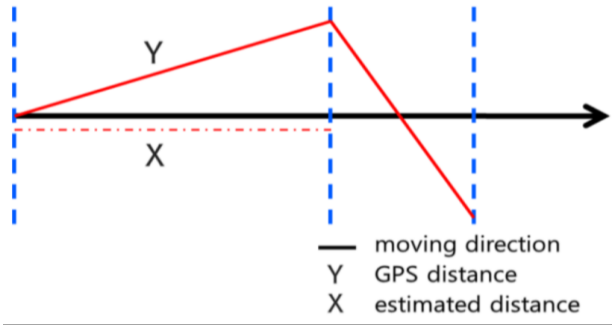



Figure 2.1Trigonometric approximation of estimated distance moved

To estimate the heading of the device, they note that the compass depends on the ambient magnetic field and hence not entirely accurate. To deal with that, they use recurrence strategy which allowed them to collect current data and combine with previous readings to make the new value to be used.

$$x_0 = a_0$$
$$x_1 = (x_0 + a_1)/2$$
$$x_n = (x_{n-1} + a_n)/(n+1)$$
$$x_i : accumulated\ heading\ value$$
$$a_i : a\ new\ compass\ value$$

----------(2)

In the above expression the initial direction is captured and stored, and the average the next ***n*** directions captured are computed in a running sum. As the user moves the compass direction of the user is unstable as the keep changing directions even though equation (1) above in stabilising the compass. To overcome this instability, we use another recurrence relation considering the distance moved so far. This distance is recorded from the inbuilt gyroscope on the device. Except for the distance factor, the new relation would the same and (1) above.

$$x_0 = a_0$$
$$x_1 = (x_0 + d_1 + a_1)/2$$
$$x_n = (x_{n-1} + d_n + a_n)/(n+1) \quad \text{---------- (1)}$$
$$x_i : accumulated\ heading\ value$$
$$a_i : a\ new\ compass\ value$$
$$d_i : difference\ of\ heading$$

# 3 Design and Methodology

In this chapter, the requirements of the is discussed. An examination of the functional (high level) requirements that the system should meet together with the technical requirements imposed by the functional requirements are discussed. Afterwards, a description of the layout of the mobile application and the actual user interface will follow. Then, small code snippets showing how the algorithm described in section chapter 2 is implemented in dart code. And the resulting user interface.

## 3.1 Design Objectives

The main objectives of the design of the product is to introduce an application running algorithm that allows for more accurate GPS data collection applicable for use in surveying applications. Since the individual is the target, our objective is to eliminate complexity and allow for seamless integration with existing mobile devices.

## 3.2 Requirements

The end-product of this project is a mobile application with the ability provide GPS coordinates with better accuracy to than the raw GPS data accessible to mobile devices. To achieve that goal, here are a few requirements the app should meet:

1. The application should be intuitive and easy to use.
2. The app should be compatible with both android and iOS platforms.
3. The complete system should have minimal to no external peripherals.
4. The application should be responsive and fast to execute with no lags

5. The application should store data recorded at each stage in a local storage on the phone to improves speed of saving data.

6. The application should be to provide a path by using polylines to visualises the connection between points.

### 3.3 Tools, Components and Resources

Table 3.1 Table of resources

| # | Tool/Resource | Purpose |
|---|---|---|
| **1** | Google Map API | Obtaining map information base on Google maps |
| **2** | Dart Programming language | Writing instructions to process the data and drive the interfaces |
| **3** | Flutter mobile SDK | Building cross platform mobile interfaces |
| **4** | flutter_google_maps | A flutter UI package for displaying google maps |
| **5** | Android/iOS-based smartphone (android used) | For running and testing the prototype |

### 3.4 Google Maps: A closer look

At the base of the design is google maps and the google map APIs provided by the Google Cloud Platform (GCP). Google map provides the mapping interface that allows us to visualise landmarks and attain approximate GPS coordinates of the location we desire. This allows us to determine the initial target location to start from. The map also acts as a basic canvas on which to plot the points obtained from the surveying exercise. We get a set of polylines from google maps to connect various points representing longitude, latitude pairs on the map to produce a visual path. While Google maps comes with a level of precision good enough for open navigation, this is not enough for our application, as we need a more accurate solution. Google maps by default cannot pinpoint the location with centimetre level accuracy.

However, to be able to produce a proper benchmark for determining boundaries, we need and accuracy level better than a meter, preferable not exceeding 20cm. To tackle this problem and reduce its impact on our product, we turn to the remaining components as discussed below.

### 3.5 Design

The system is designed to be a mobile application connected to the internet via 3G/4G/WIFI connection. It receives regular positioning information via radio waves from the telecom provider. To interpret this as a google map, device sends the data to google maps API backend for decoding and encoding as map information. This transaction happens across the internet. The map information is returned to the device to be displayed as a map. The flow of this transaction is as shown in figure 3.1:

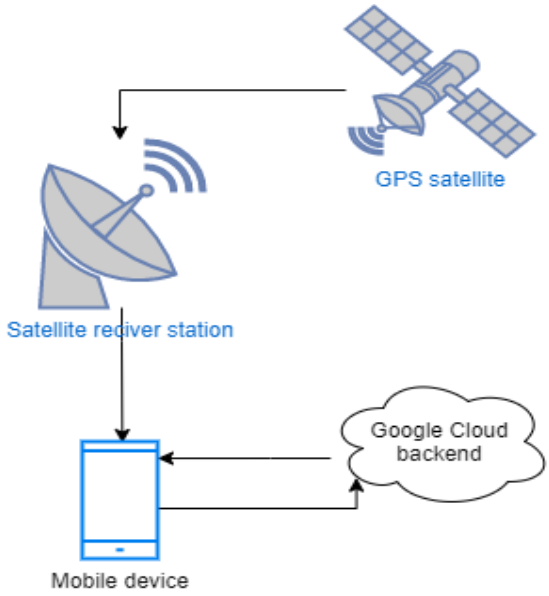



Figure 3.1 Location data exchange

The resulting map would then be displayed to the user with the error margins described earlier. It also comes with a red marker indicating the user's estimated position. As discussed earlier, this reported location is expected to be off by a margin anywhere between 5 and 50 meters depending on the surroundings as discussed. Figure 3.2 shows a sample raw map.


Figure 3.2: Snapshot of a raw map

### 3.6 System Operations

In the previous section we discussed the default operation of a regular google map and how it is used to locate targets. Now we discussed the operation of our enhanced system using the proposed algorithm. This relies on the inbuilt gyroscope, accelerometer and compass to provide a better accuracy for our purpose.

#### 3.6.1 Compass

By default, heading information can be extracted from the google map API. However, due to the high level of inaccuracy associated with the map, direction information obtained from the APIs may be misleading and may not properly respond to small changers in direction

of the subject. Luckily, mobile devices are built with compasses in them for navigation purposes. These can be used to determine the bearing of the device more accurately independent of google maps. To help google maps produce a more accurate reporting, we can use the Flutter SKD and "flutter_compass", a package for obtaining readings of the direction of the host device to get the direction reported by the device. We can then choose which is more accurate. If it the data from "flutter_compass" is determined to be more accurate we will simply replace the direction information obtained from google maps with those obtained independently and adjust the point appropriately.

### 3.6.2 Accelerometer

Google maps also attempts to calculate the speed and acceleration of the device. However, for the same reasons as stated earlier, these could be inaccurate. Thankfully mobile devices have inbuilt accelerometers that track their speed and acceleration and change in orientation of the device. Relying on this, we can obtain the speed of the device as reported by the device itself. Based on this report we can verify the newest coordinate reported by google maps by using the previous coordinates, the amount of time passed, and the speed reported by the internal accelerometer to compute the distance moved in in the given time and speed. We can the compute the distance the between the previous coordinated and the currently reported one. If the distance deviate from form that obtained using the accelerometer, we can then choses which is likely to more accurate. This is illustrated below:

*Let* $\boldsymbol{D}$ *be the distance between two GPS cordinates*

*Let* $\boldsymbol{d}$ *be the distance travelled as computed from accellerometer*

*Let* $\boldsymbol{t}$ *be the time passed between the the previous and current GPS cordinate*

*Let* $\boldsymbol{s}$ *be the average speed of device*

$$distancein\ x\ time\ (d) = \frac{average\ speed\ from\ acceleration\ (s)}{time\ passed\ since\ last\ cordinate(t)} \text{ ---------- (3)}$$

*Let* $\emptyset_1, \boldsymbol{\lambda_1}$ *and* $\emptyset_2, \boldsymbol{\lambda_2}$ *be the longitude and latituds of two GPS cordinates* 1 *and* 2

*Let's also denote their absolute differences as* $\Delta\emptyset$ *and* $\Delta\boldsymbol{\lambda}$

Then we can compute their central angle as $\Delta\sigma$ using the spherical law of cosines as follows:

$$\Delta\sigma = \arctan \frac{\sqrt{(\cos\phi_2 \sin(\Delta\lambda))^2 + (\cos\phi_1 \sin\phi_2 - \sin\phi_1 \cos\phi_2 \cos(\Delta\lambda))^2}}{\sin\phi_1 \sin\phi_2 + \cos\phi_1 \cos\phi_2 \cos(\Delta\lambda)}$$

We can then compute the actual arclength as $\boldsymbol{D = r} \times \Delta\sigma$

After the above two computations, the two distances can easily be compare to determine which one is likely to be accurate based on factors like the accuracy of the previous value, the next and the actual pace of the user's movement. Using this technique, we can attempt to reduce the inaccuracy introduced by the google maps.

# 4 Implementation.

This chapter will be providing clear details of the implementation process. It should be noted that this chapter is short and concise as everything here is a mere implementation of the mathematical models provided in chapter 3. The chapter including screenshots and code snippets to illustrate the inner workings of the mobile app. It all begins with a brief justification for the choice of programming language and framework.

## 4.1 Choosing Flutter

Flutter is a framework for developing mobile user interfaces and implementing application logic using the Dart programming language. It was designed by google to solve one of the most pertinent challenges mobile developers have faced over the years – cross platform compatibility. When an application is developed for the android platform, traditional it is written in either Java or Kotlin. However, even the simplest application developed this way cannot be used on the iOS platform because of incompatibility. To get the benefits of the same mobile application on iOS, one needs to undergo a full process of developing the same application from scratch using objective C or swift.

The introduction of flutter made it possible for one to write the interfaces and logic in of an application in a single language and codebase. Then then they can compile into both android and iOS compatible applications. This makes it extremely easy to develop an application with no concerns about platform limitations.

In this project flutter is used for two main reasons as follows:

1. Cross platform compatibility: due to the cross-platform compatibility nature of flutter it is easy to test this same code on iOS and android.
2. Ease of use: Flutter provides a set of already made widgets and libraries for faster manipulation of data. For instance, to retrieve data from device sensors, there is no need to write native code. There exist libraries to help with that.

## 4.2 User Interface and Basic Setup

To get started, an API key from google cloud platform is obtained. Then the google_maps_flutter library which provided the graphical view of google maps in flutter using the API key is installed. With additional setup and tweaks, the maps interface was up and running showing the Ashesi campus where the project was carried out. Figure 4.1 is a screenshot of the application at this stage.

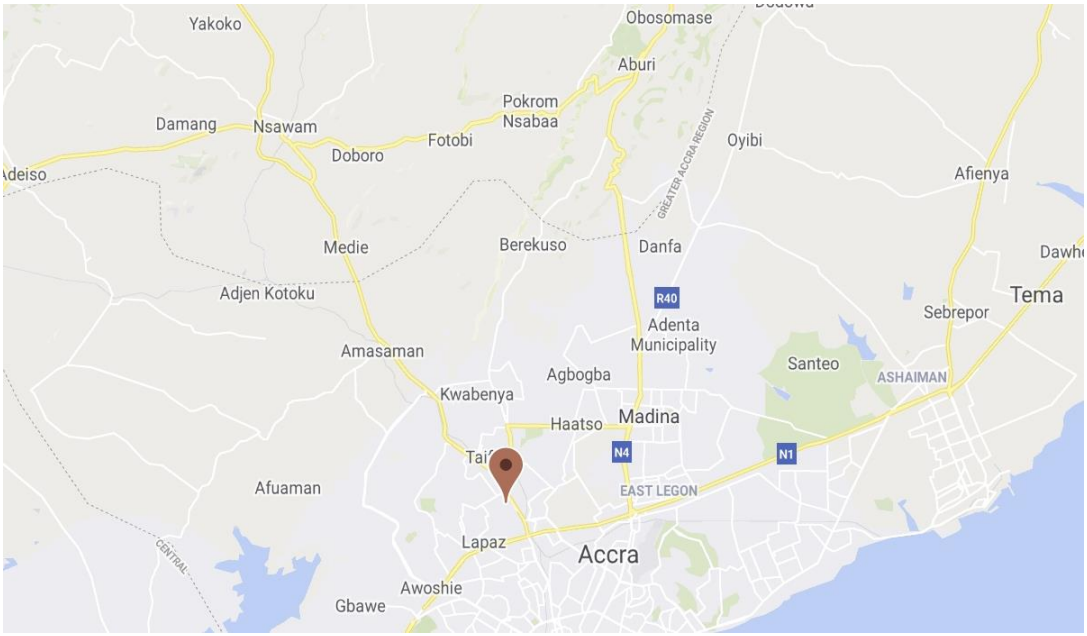



Figure 4.1: Google maps initial appearance

Once the UI is set up with an initial map view, we're ready to include functionalities such as tracking to make sure that map updates its focus region from the home to the user' new location should they move. Figure 4.2 is a snippet of a controller that would be helping to update the focus to the user's current location.

```
void _onCameraMove(CameraPosition position) {
  _lastMapPosition = position.target;
}

void _onMapCreated(GoogleMapController controller) {
  _controller.complete(controller);
}
```

Figure 4.2: Tracking initialisation blocks of code

The first block in the snippet changes the centre of the map to the new position of the user as soon as movement is detected and a change in coordinates is recorded. This ensures that the map "follows" the user if they move. The second block is called any time the app is opened it had been closed. It initialises the maps using the API keys to ensure the correct information is exchanged with the backend and the app.

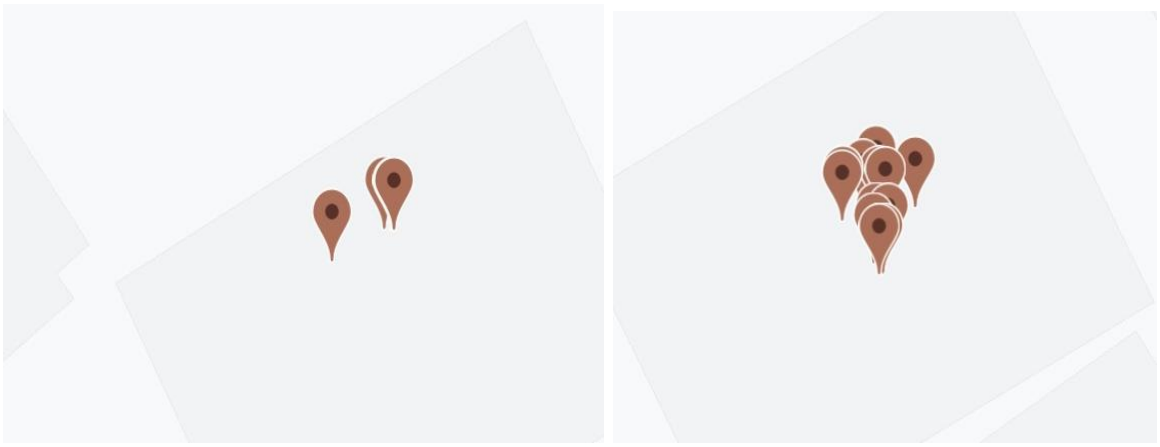
### 4.3 Filtering and Correcting Signals

Once the map is capable of tracking movement, the next agenda is to combine and apply the filtering and correction algorithms developed to the data we recorded. This once the algorithms a designed, applying them is as simple as calling a function, retrieve the return values and use plotting them on the graph. This will produce a new point that has been filtered can corrected to be closer to the real location point that the raw GPS data would.

# 5 Results and Analysis

This chapter shows some test results for using normal maps my device to find a path a using the application developed to trace the same path. It also lays side by side the polylines for each image and show how they differ or are similar. It will then proceeds to discuss the observations.

## 5.1 Staying Still



(b) Data from raw maps (b) Data from raw maps 30secs later

Figure 5.1 Comparing data

In the first test, google map and my custom map (hereafter referred to as *survemap*) were tested in a stable environment with the phone lying down in one spot. Google maps was programmatically induced to drop a red pin whenever location changes. However, the device was kept still and stable. In the first five seconds, it had dropped three pins as can be seen on figure a. 30 seconds later, it had dropped many more pins around the location as can be seen in Figure 5.1.For the same experiment, survemap was placed next to the google maps for the entire duration of the experiment. It was also programmed to drop a pin anytime the location

changed to mark new location received just like google maps. However, after a minute of waiting, survemap had dropped three pins but all pins were so close together that they be rounded off to the same value as seen in figure 5.3

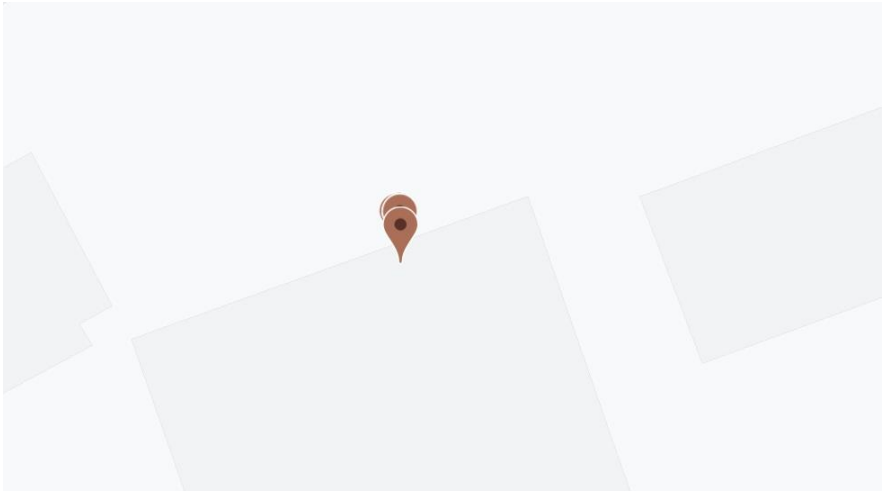


Figure 5.2: Survemap after 1 minute.

In the second experiment, like the first, both were programmed to drop a pin anytime the location changed, this time with a more practical approach – the device was in motion. Since this project is being undertaken to potentially develop a mapping solution based on it, it was required that we put both to the test under similar conditions. Figures a and b represent google maps and survemap respectively. Survemap traced a straighter line than google maps.

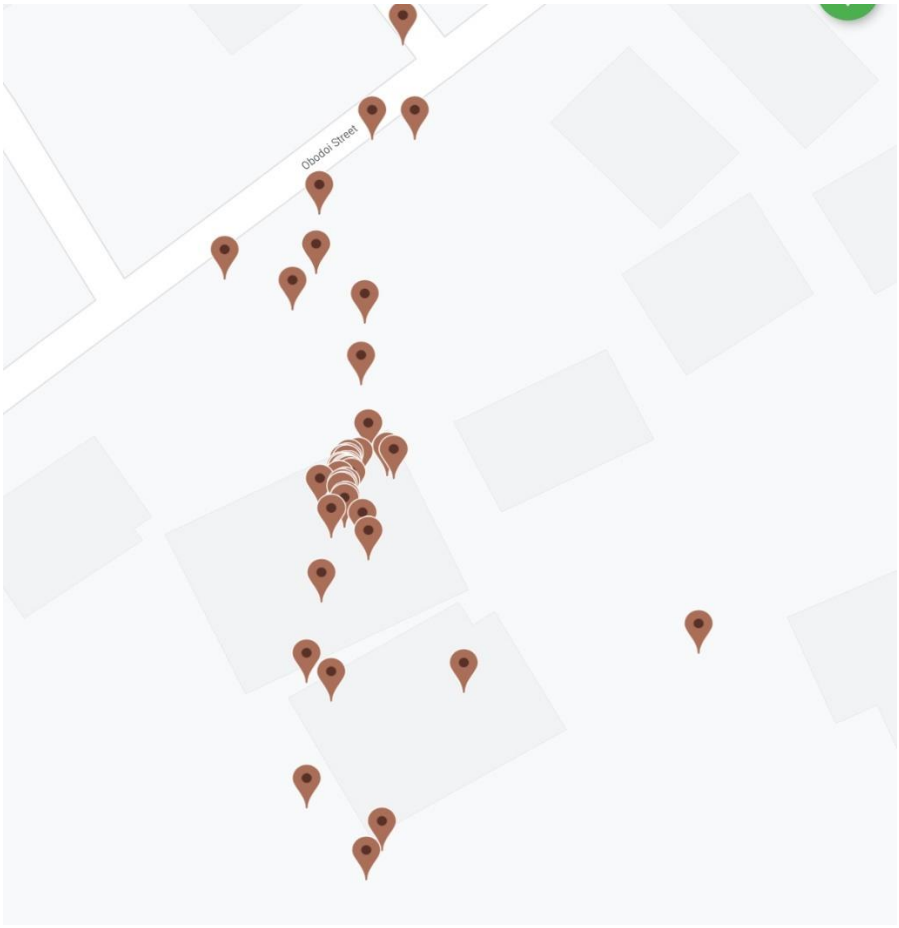



*Figure 5.3a google maps path tracing*

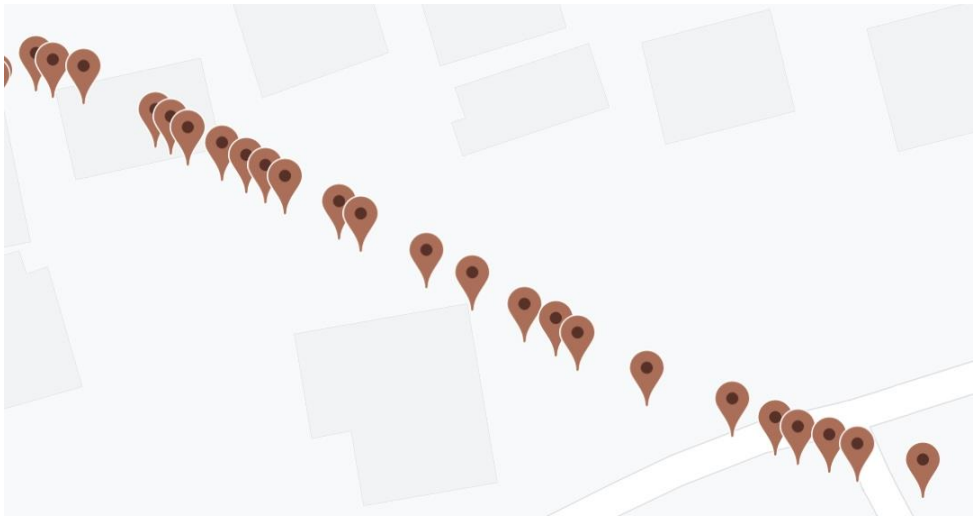


Figure 5.4 survemap path tracing

## 5.2 Discussion

Form the two experiments above, google maps, while good for general navigation purposes, it fails meet accuracy requirements for applications that require very accurate location data. The multiple pins seen in figure 5.1 indicates that with updated location google map receives, it introduces a different error that makes the points inconsistent leading to the littering of pins around the location. Survemap on the other hand using the given algorithms could determine such inconsistencies and discard the values keeping only those values close enough to the previous.

# 6 Conclusion, Limitations and Future Works

## 6.1 Conclusion

To conclude, we should note that while survemap maintained a level of consistency and demonstrated an ability to stick to its initial accuracy, it still approximated the location to accuracies of about 1.5 meters. This means it can be applied to some pinpointing or surveying application where the accuracy is not strictly required to be under 1 meter. More work can be done to improve this for a more reliable application.

## 6.2 Limitations

This project has demonstrated some success but still relies on the internal gyroscope, accelerometer and compasses which in themselves are note very accurate. **Therefore,** while we can achieve consistency it may not provide precise location sometimes.

## 6.3 Future Works

In the future, another way of further improving the system is to use a process known as multilateration. It relies on 4 routers to create intersecting circles around the target to determine its location. This method is known to be more accurate but require the use other peripheral and could be expensive to build. However, if accuracy becomes the topmost priority then, it might be a better solution to use.

# References

[1] S. Hwang and D. Yu, “GPS Localization Improvement of Smartphones Using Built-in Sensors”, International Journal of Smart Home Vol. 6, No. 3, July 2012

[2] J. Chon and H. Cha, "LifeMap: A Smartphone-Based Context Provider for Location-Based Services" in IEEE Pervasive Computing, vol. 10, no. 02, pp. 58-67, 2011.

doi: 10.1109/MPRV.2011.13