MOTION SENSOR FUSION FOR PHONE LOCALIZATION

Vidi Valianto Shaweddy

1697512

November 3, 2017

Markus Wagner Yuval Yarom

Master of Computer Science

Faculty of Engineering, Computer and Mathematical Sciences

University of Adelaide

Contents

List of Figures

Abstract

Motion sensor is an important part of Android smart phone used for localization. The mostly used iteration of Android, Android 6.0 introduces new functions to improve the accuracy of motion sensors. However, the utilisation of motion sensors does not need user's permission. The lack of permission can create security issues as user personal information can be revealed through location. Several researches using motion sensors have been conducted on previous iteration of Android. However, all of them suffer from low accuracy. Therefore, another experiment is conducted to measure the localization accuracy and the security issue level on Android 6.0.

Experiment is conducted on user normal movement activity. An Android application is created to measure the orientation and distance using motion sensors. A comparison is made on sensors, GPS and real path to measure the user tracking accuracy. Additionally, controlled experiment using robot is conducted to measure new sensors' accuracy on Android 6.0. Both accuracy results are used to identify the security issue level on localization.

The experiment result shows that orientation sensor manages to provide good accuracy but linear acceleration suffers from high noise level. It is difficult to utilise linear acceleration for distance calculation. Therefore, there is an accuracy increase on orientation and chance to improve linear acceleration accuracy however, it is not enough to create a security issue on localization.

Keywords: Motion Sensor, Localization, Security

Chapter 1

Introduction

Technology has been evolving rapidly over the last couple years, from the development of small size personal computer into the innovation on the mobile devices. The affordable price of the technologies increases the total amount of the new technologies demanded by the public. One of the most used new technologies in the last couple years is the smart phone. Several operating systems have been introduced to provide a better user experience such as Android by Google.

Over the last couple years, Android operating system (OS) has been updated by the Google couple times to provide new functionality. Along with the upgrade of the operating system, the hardware is also upgraded in several areas, such as memory and display. One of the most improved areas of the smart phone's hardware is sensors. In order to provide new functionalities and accessibilities for users, new sensors are embedded on the smart phone. These new hardware and software combinations help smart phone to solve more complex computation and increase the functionality of the smart phone.

However, the new features from the software and hardware may create problems on the smart phone using Android OS. Several new security issues arise in the last several years especially in the area of personal information exploitation by unauthorized application [7]. The exploitation happened because several sensors or services on the Android phone can be used by an application without any requirements to ask for user's permission such as motion sensors. Also, Android OS may lack several necessary security measures as Android OS is an open source based operating system [8] which allows users to fully customize the operating system based on their needs.

As the smart phone has become an important tool used by masses, more users store important and personal information inside the smart phone and also use it to share personal information with others. These behaviours can create a security breach by an unauthorized user who intends to extract valuable information such as personal identification or password.

One valuable piece of personal information of the phone user is their current location. Users usually share location to the user when the permission is given. As the access to the location of the user can let others learn about user's personal information such as user's workplace or user's daily activity.

Several experiments have been made to find the location of the Android phone user. Tobias [7] found out that signaling system on the phone can inform the attacker about the user location. By using the signal, Tobias created a spy program that pretends to be a network operator. The spy program is used to send a request to the network operator where the phone is connected at the moment. It is possible as the current signaling system does not have any authorization methods to filter the network operator, thus allowing everyone to access the information regarding the user location. However, the spy program can only access the city where the user is located as the information regarding the closest cell tower can only be accessed by the network operator where the phone is currently connected.

In addition, Shala and Rodriguez [9] developed an application to detect user location using the accelerometer, which is a form of motion sensor. As the signal strength often drops inside the building especially below the ground level, Shala and Rodriguez used the accelerometer to calculate the location. However, the application can only be used in a close distance which is inside the building because of the low accuracy of the calculation of the position and displacement and the direction of the movement.

Ayllon et al. [10] also developed a way to detect user location using radio frequency and microphone to capture the ripple of the sound. By using the speaker to send a low-frequency sound, and microphone to capture the sound that bounces back from the wall, the application can measure the distance of the phone from the wall, thus pinpoint the user location. However, this application cannot be used in the outdoor situation as the sound has a distance limit meaning the microphone cannot capture the ripple. Therefore, the experiment can only be done on an indoor location and the user is in proximity to a wall.

Additionally, Han et al. [11] developed an application to predict the user location in an outdoor situation using the accelerometer. The application collects data on the user and use it to predict the movement of the user to the final location without the initial location. However, the application needs to collect a big amount of data to increase the accuracy of the prediction and the prediction will get noisy over time and will be inaccurate after 200 meters.

The latest attempt was created by Michalevsky et al. [12] to track user location using battery consumption in the city. By identifying the spike in battery consumption, the application can detect that there is a hand-over between signal tower. This event can be extracted after the collection of data to distinguish the hand-over event over other battery consumption events. However, the noise produced by the data is high, thus decreasing the accuracy of the result.

The previous experiments conducted by the papers share similar flaws in terms of the location as the attack usually can only be done inside a particular area such as indoor or within a predetermined range. The accuracy of the result has also become one of the major problems of the previous experiments as the level of noise captured during the experiment is quite high. Therefore, the result of the previous experiment cannot be considered as a major security issue or threat to user's privacy. In addition, most of the experiments were conducted on older version of Android OS and the current iteration has developed several improvements on the security and permission area that may render some of the experiments unusable.

Therefore, a new research should be conducted by improving the current state of the research by utilizing the sensors located in the phone that can be accessed without user's permission. The experiment will utilise the Android OS 6.0 as the most used iteration/version of Android on the market [13] and all users may be exposed to a security issue. The experiment will be conducted by using the fusion of the motion sensors to reduce need to store a big amount of data for path prediction as the addition of the new sensors and API on the phone can increase the accuracy of prediction.

An experiment will be conducted to measure the performance of motion sensors on Android 6.0. In order to collect data, an Android application will be made using sensors fusion inside Android phone. There will be 2 main sensors created using sensor fusion, which are, orientation and linear acceleration sensor. Data from 2 sensors will be added to log file alongside Global Positioning System (GPS) latitude and longitude value, step counter, time and other related value needed to calculate the data on the next step.

The data extraction will be conducted on user's normal activity, such as, walking, cycling and driving. Direction and acceleration data will be the two main data sources extracted on the experiment to determine user's final position based on known initial location. The data from motion sensors will be compared with GPS and the real path taken during experiment to compare the performance of motion sensors in terms of direction and distance.

In addition, a controlled experiment will also be conducted to see the performance of newly added sensor, linear acceleration. Different terrains will be used to analyse the linear acceleration's performance on different situation with external noises. The results from controlled experiment will be used to determine the current accuracy with the sensors. In addition, the results can be used to identify the noise level and related issues on current linear acceleration sensor implementation on Android 6.0.

The results of the experiment will be used to determine the quality of motion sensors on Android 6.0 and the improvement that can be made on the next iteration of Android. In addition, the experiment results can also be used to identify the level of security issues in the area of privacy and the next recommendation can be created to prevent information leakage.

Chapter 2

Background

This section will explain the information regarding the sensors, the calculation and other relevant information that are used in the experiment using Android 6.0 on Nexus 6.

2.1 Motion (Inertia) Sensor

On the Android 6.0, there are several motion sensors where three main sensors, accelerometer, gyroscope and magnetometer are hardware-based sensors and other additional sensors which are, step counter and orientation sensor, are software-based sensors. There are three axes on each sensor, which are x, y and z-axis. Each axis in the motion sensors is pointing towards a different direction, x-axis is pointing to the right side of the phone, y-axis is pointing to the top of the phone and z-axis is pointing to the front face of the screen. The axis will remain the same even though the orientation of the phone changes. The figure of axis on the phone can be seen on Figure 2.1.

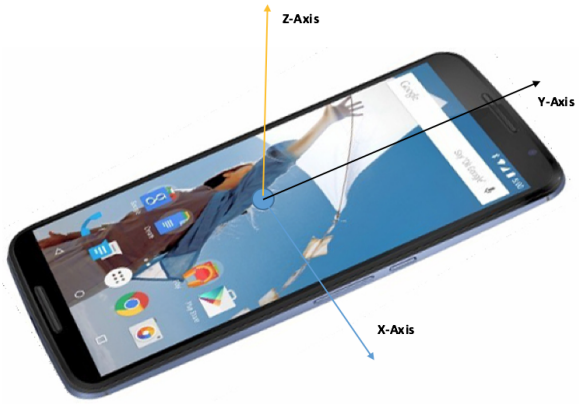

Figure 2.1: Axis Position on Phone

2.1.1 Accelerometer

Accelerometer is a motion sensor used by the phone to measure the change in the velocity caused by force applied to the sensor. Accelerometer is susceptible to the gravity force making the calculation of the acceleration is the addition of the motion acceleration and gravity acceleration [9]. The figure of the acceleration calculation can be seen in Figure 2.2.

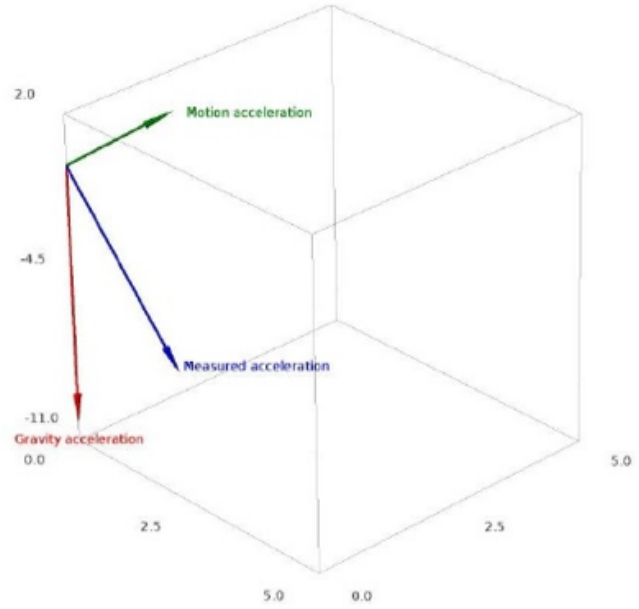

Figure 2.2: Acceleration Calculation of Accelerometer [1]

In order to find the value of the acceleration applied to the device, the equation is,

$a_d = -\sum \frac{F}{m} - g$

where,

F = Force (N)

m = Mass (kg)

g = Gravity (m/s2)

a_d = Acceleration on the device (m/s^2)

as $\sum \frac{F}{m}$ is recorded as square root of –x2,-y2,-z2 on device therefore,

$a_d = \left(-\left(-\sqrt{x^2 + y^2 + z^2}\right)\right) - g$

$a_d = \sqrt{x^2 + y^2 + z^2} - g$

2.1.2 Gyroscope

Gyroscope is a sensor to measure the acceleration of the phone using the rotation of the sensor inside the phone. The phone calculates x, y, and z-axis using the position of the sensor inside the phone where the centre position counts as 0 for x, y and z-axis [1]. The figure of the acceleration calculation can be seen in Figure 2.3.

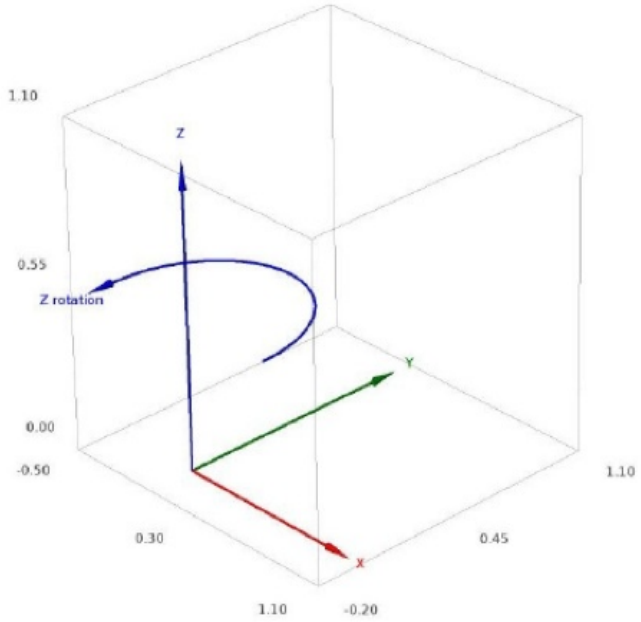

Figure 2.3: Acceleration Calculation of Gyroscope [1]

In order to find the value of the acceleration applied to the device, the equation is,

$a_g = \sum_{i=0}^{n} (\omega_i - \Delta t)$

where,

θ = angular velocity (rad/s2)

t = time (s)

a_g = acceleration in gyroscope (m/s^2)

2.1.3 Magnetometer

Magnetometer is a sensor in the Android smart phone to detect the ambient magnetic field. Magnetometer works similar to the conventional compass using magnetized needle. Magnetometer used earth's magnetic field to find the position of the north compass. However, magnetometer is vulnerable to the noise caused by strong magnetic field near the magnetometer.

The magnetometer sensor on the smart phone or tablet uses a modern solid state technology to build

a small Hall-effect sensor that detects magnetic field using 3 axes X, Y and Z [1]. Hall-effect sensor will produce a voltage that is proportional to the polarity and strength of the magnetic field for each axis [1]. The sensed voltage will be converted to the digital signal that represents the intensity of the magnetic field. There are several other types of magnetometer such as magneto resistive devices that change the measured resistance based on the change in the magnetic field around the device. In addition, the magnetometer can also help detecting and calculating the relative orientation of the device using the strength of earth's magnetic north on each axis.

2.1.4 Step Counter

A sensor of the smart phone using the accelerometer to detect a step taken by the user. The sensor will add one value based on the event generated by each step. The step counter can return value based on the time stamp or for each step as soon as it is taken. The step counter works with step detector function which returns value one if the sensor detects a step.

2.1.5 Orientation Sensor

Orientation sensor on the smartphone is used to calculate the rotation matrix based on the X, Y and Z axis to measure the orientation of the phone. 3 axis represents different rotation on the phone which are azimuth, pitch and roll.

Azimuth is the angle of rotation of the z-axis. The value represents the angle between the magnetic north pole and the y-axis of the device. When the phone is facing towards the north, the value will be zero and when the phone is facing towards the south, the value will be π [14]. The value will be $-\pi/2$ when the phone is facing the west side and $\pi/2$ when the phone is facing the east side [14]. The range of the value is from $-\pi$ to π.

Pitch is the rotation happened on x-axis. The value represents the angle between a plane parallel to the device screen and the back of the phone parallel to the ground. With an assumption that the bottom edge of the device is facing toward the user and the screen is face-up, by tilting the top edge of the phone towards the ground will create a positive pitch angle [14]. The range of the value is from $-\pi$ to π.

Roll is the angle of rotation on the y-axis. The value represents the angle between a plane perpendicular to the ground with the plane perpendicular to the device's screen. When the bottom edge of the device is facing the user and the screen is face-up, by tilting the left edge of the phone towards the ground will create a positive roll angle [14]. The range of the value is from $-\pi$ to π.

The activity of the azimuth, pitch and roll will be used to indicate the roll movement of the phone. The position of each roll activity can be seen in Figure 2.4.

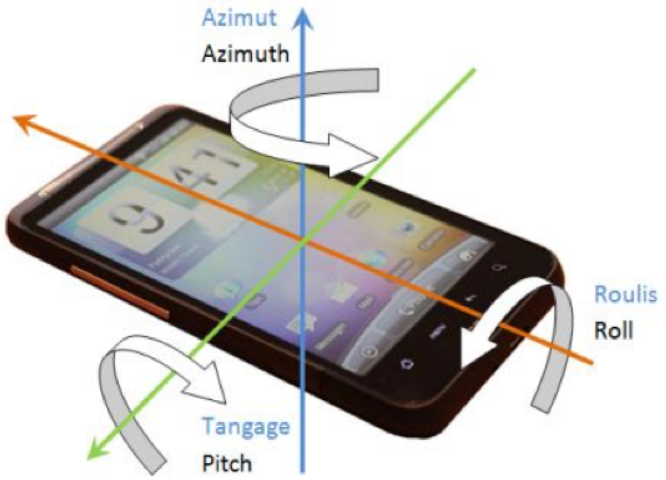

Figure 2.4: Phone Orientation [2]

2.2 Network Based Phone Localization

Phone localization is a term used to ascertain the position or location of the phone whether the phone is in stationary or moving state. Localization of the phone can be done through several types of measurement and different technology usage. Each type of the localization has their strength and weakness based on several areas such as, the battery consumption and location. There are three main types of localization which are network based, Global Positioning System (GPS) based and Wi-Fi based [15].

The network based localization utilizes service providers' network infrastructure. The network based localization has been developed many years prior to the introduction of GPS. The network based uses triangulation technique with the cell base stations. However, the accuracy of the localization based on the concentration of the cell phones around the area of the phone, therefore the highest possible accuracy usually can be achieved at urban areas.

The GPS based localization utilizes the GPS sensor inside the phone to communicate with the satellite. The GPS will communicate with the phone over time to update the location of the phone. There is usually more than one GPS satellite used to communicate with one device to provide a high accuracy location detection. However, as the phone needs to communicate with the GPS satellites every couple minutes, the battery used by the phone will increase significantly and reduce the efficiency of the GPS usage over time.

The Wi-Fi based localization is using the crowd-sourced Wi-Fi data to find the location of the phone. The Wi-Fi based localization usually used for the indoor situation in order to compensate the poor quality of GPS-based localization inside the building. However, the phone must be connected to the Wi-Fi in order for the localization to work properly.

The current phone localization usually combines Wi-Fi, GSM and GPS based localization to pin-

point the phone location. The combination will increase the accuracy of the localization both in the indoor or outdoor situation. However, there are still some areas where the 3 types of the localization methods cannot perform well, such as, inside the tunnel or subway station. Also, the battery consumption is quite high when the combination of the localization methods is used, reducing the efficiency of the usage for long term movement, such as walking. Thus, a new way of localization should be created to accommodate the certain needs of users. One of the solutions proposed in the recent years is the utilization of the motion sensors.

2.3 Sensor Fusion

Sensor fusion is the combination of several sensors or data derived from different sources that can improve the accuracy and reliability of the data that cannot be achieved when the sensor was used individually. The increase of accuracy and reliability on the data is usually known as uncertainty reduction. The combination of the sensors usually comes from two similar sensors however, there are some cases where the fusion comes from two different type of sensors. The combination of heterogeneous sensors is called direct fusion, and the combination of two different type of sensors is called indirect sensors [16].

There are two main types of sensor fusion in terms on how the data fused, which are, centralized and decentralized. Centralized fusion happens after the user collects all relevant data and an entity or function will combine it at the central location. Decentralized fusion occurs when the user will be fully responsible on fusing the data or manual data fusion. In the Android smart phone, the fusion usually uses centralized fusion where the features inside the Android OS where the library containing several functions will combine the data and return the final result.

As one of the most used application of the sensor fusion is phone localization using the combination of GPS and motion sensors, the Android OS provides many different methods to combine the sensors. The Android OS also adds several filters inside the library to provide a higher accuracy data with less noise. Thus, creating a reliable information regarding the phone localization.

2.4 Linear Acceleration

Linear acceleration is the acceleration caused by the movement of the phone with uniform acceleration in a straight line [17]. Linear acceleration will measure the change in velocity in the same direction to the motion where angular acceleration will measure the change in velocity where the change is perpendicular to the direction of the motion, and the difference can be seen in Figure 2.5.

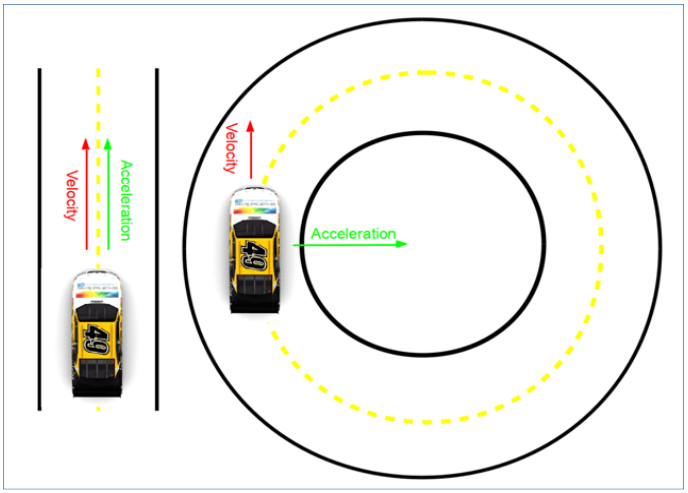

Figure 2.5: Left: Linear Acceleration, Right: Angular Acceleration

Inside the Android OS based smart phone, accelerometer is used to calculate the acceleration. However, the accelerometer is vulnerable to the gravitational force; thus the accelerometer adds the gravitational value to the measurement. In order to calculate the linear acceleration, the gravitational value must be deducted from the measurement of the accelerometer using the equation,

$$Linear\ Acceleration = Measured\ Acceleration - Gravity \text{ [18]}$$

However, the problem arises as the orientation of the phone changes and making it hard to determine which part of the value is gravity.

In the older device, the Android OS combines accelerometer and the magnetometer to calculate the orientation and measure the linear acceleration based on the rotational force on the phone. However, there are some issues with both sensors, accelerometer adds gravitational value to the sensor, increasing the error rate produced by the yaw movement as the gravity is parallel to the yaw. On the other hand, magnetometer uses earth's magnetic field which is parallel to the pitch movement, increasing the error rate of the pitch. Thus, the combination of the roll, pitch and yaw on the combination of accelerometer and magnetometer will be erroneous.

Thus, a new sensor is added to newer version of the device which is the gyroscope. Gyroscope is one of the hardware-based sensor on the smartphone used to measure rotation with a pair of vibrating arms to measure the Coriolis effect [17]. Coriolis effect is caused by the Earth's rotation and the change in direction is measured by the vibrating arms. However, the gyroscope also has errors and issues on rotation measurement known as gyroscope drift. The drift happens when the sensor cannot identify the centre of the phone and start drifting to the side of the phone, changing the point of the centre and adding error value to the measurement.

In order to increase the accuracy of the linear acceleration measurement, the value of the gyroscope should be added to the combination of the accelerometer and magnetometer sensor, as the gyroscope

can deduct the gravity and earth's magnetic value from both sensors. Therefore, the accuracy of the measurement of the linear acceleration can be increased and the value of the calculation has higher reliability than the one on the fusion of accelerometer and magnetometer. The graph of the sensor fusion can be seen in Figure 2.6.

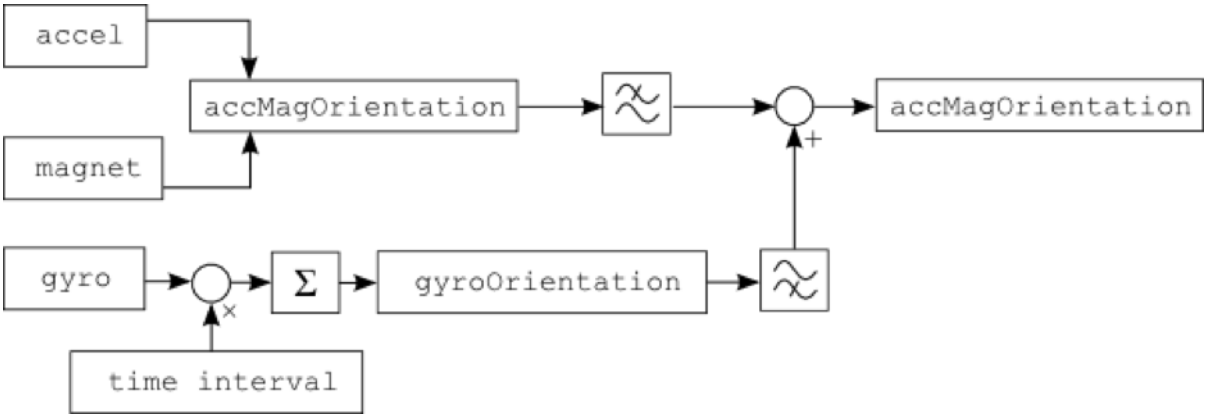

Figure 2.6: Linear Acceleration Sensor Fusion [3]

Azimuth is an angular measurement used in a spherical based coordinate system. The vectors from the centre of sphere or observer to the destination or point of interest are projected perpendicularly onto a plane [19]. The angle produced between the projected vector to the position of the north is called the azimuth.

The azimuth is often being used on the navigation with denotation alpha. Azimuth is defined as the horizontal angle measured clockwise from the true north direction line. The figure represents the position of azimuth can be seen below,

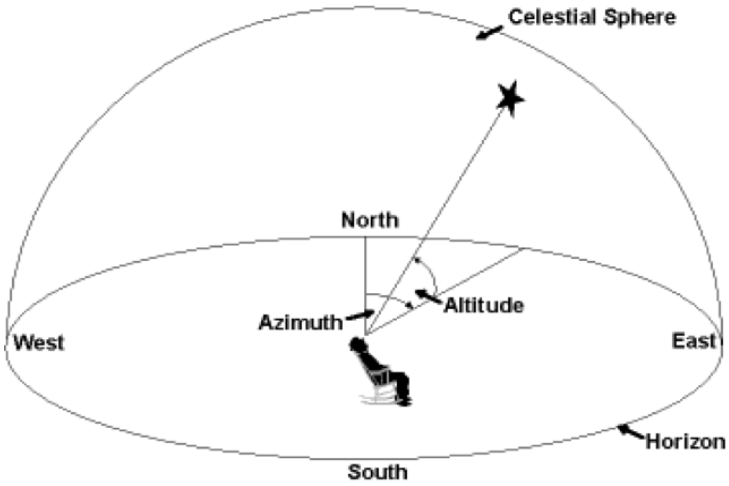

Figure 2.7: The position of Azimuth in Spherical Coordinate System. [4]

In order to find the azimuth value inside the Android OS smartphone, the usage of the orientation sensor is needed as the orientation sensor produces three types of rotation which are azimuth, pitch and roll [14]. The orientation value comes from the combination of accelerometer and magnetometer to produce rotation matrix. The magnetometer will measure the reference line which is earth's magnetic north and the accelerometer will measure the direction of the movement, thus producing the azimuth value. The fusion of accelerometer and magnetometer can be seen in Figure 2.8.

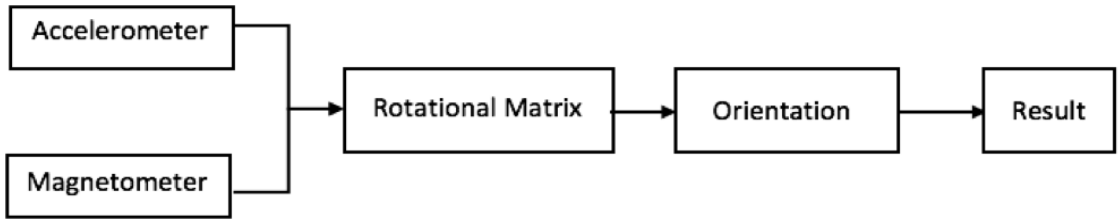

Figure 2.8: Sensor Fusion for Orientation Measurement

2.4.1 Displacement

In order to predict the movement of the phone over time. A calculation to measure the distance of the movement is needed to accurately predict the movement path. There are two types of distance calculation which are, distance and displacement. Both calculations have a different type of usage based on the type of movement needs to be calculated.

Distance is the scalar quantity that calculates the amount of ground that the object moves over a set of time during the motion [20]. In order to calculate the distance, the observer must measure the acceleration at the reference time and at the current time. Therefore, the observer can calculate the change in the speed and measure the distance based on the change.

Displacement is the vector quantity of the movement [20]. It calculates the amount of change in the location during a period of time. Therefore, the value will be zero if the user moves in the circle and come back to the original location within a period of time. However, it helps to measure the object's overall change in position. The difference between distance and displacement can be seen in Figure 2.9.

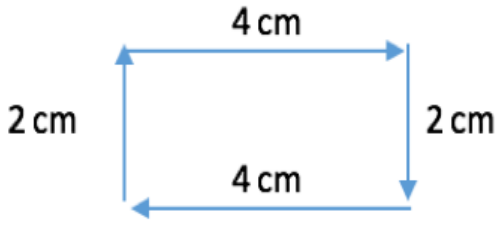

Figure 2.9: Distance and Displacement Comparison Path

In the explanation, if the user moves within a period the time with the path shown by the picture above from lower left and back to the starting point, the distance will be 12 cm and the displacement will be 0. The equation of the displacement can be seen below,

$$a = \frac{dv}{dt}$$

$$dv = a\ dt$$

$$\int_{v_0}^{v} dv = \int_{0}^{\Delta t} a\ dt$$

$$v - v_0 = a\ \Delta t$$

$$v = v_0 + a\ \Delta t$$

$$\text{as, } v = \frac{dx}{dt}$$

$$dx = v\ dt = (v_0 + at)\ dt$$

$$\int_{x_0}^{x} dx = \int_{0}^{\Delta t} (v_0 + at)\, dt$$

$$x - x_0 = v_0 t + \frac{1}{2}at^2$$

$$x = x_0 + v_0 t + \frac{1}{2}at^2$$

where,

x = displacement (m)

v = velocity (m/s)

a = acceleration (m/s^2)

t = time (s)

2.4.2 Clustering

Clustering is a method used in data mining to create partition several different observations into different clusters such as k-means [11]. Several iterations will be made in order to create a convergence. There are 2 main steps of clustering which are,

Assignment step: an observer was created on each intended cluster and add data to the data space which located inside the cluster radius with observer as the centre point [21].

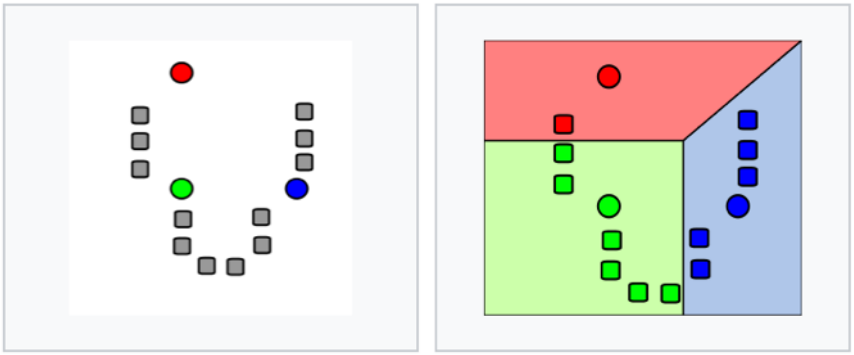

Figure 2.10: Creation of Clusters [5]

Update Step: the observer was moved into one of data point and the data will be put on the intended clusters that are adjusted based on the observers [21].

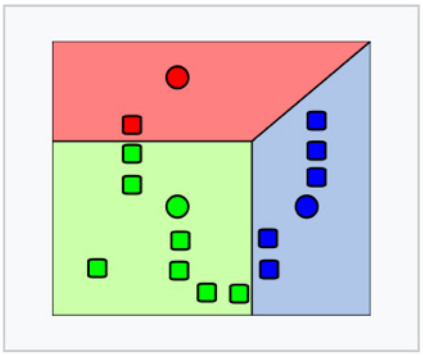

Figure 2.11: Clusters Update

2.4.3 Mean Filter

Mean filtering is an intuitive and simple method of smoothing data by reducing the amount of intensity variation between one data to the next one. The mean filtering is a part of spatial-based filter that often used to reduce noise in the data. Mean filtering replaces the data value inside the dataset with the average (mean) value of its neighbours with the inclusion of itself [22]. The filter will create an effect of eliminating the data that are unrepresentative of the surroundings. Mean filter uses the same approach as a convolutional filter by using a kernel which represents the size and shape of the neighbourhood [22]. The kernel is the sample data used to calculate the mean. The mean filter works the same as low-frequency filter in general and thus, reducing the spatial intensity derivatives present in the data thus creating a smoothing effect of the data. The example of how the mean filter works using 3x3 kernel is,

5	3	6
2	1	9
8	4	7

$$5 + 3 + 6 + 2 + 1 + 9 + 8 + 4 + 7 = 45$$

$$45 / 9 = 5, \text{ thus}$$

5	3	6
2	**5**	9
8	4	7

The centre value changes from 1 to 5 based on the mean of the neighbours around the centre value.

Inside the motion sensors, the mean filter is used to smoothen the data by comparing the value of x, y and z to reduce the noise on the measurement which can cause a spike in value in one of the axes on the sensor. As the filter is needed to reduce the noise carried from the original sensor to the result of fusion between sensors. Thus, it is important to add a filter to increase the quality and reliability of the data.

Chapter 3

Literature Review

Several experiments have been conducted on how to find the user location without using GPS whether to compensate the problem with the lack of GPS signal on the indoor situation or to find an exploit on the phone which can be used to locate the user without user's permission. Several solutions were proposed during the last several years, such as using cell phone network system, motion sensors, battery consumption, and low-frequency signal produced by the phone.

3.1 Localization using Cell Phone Network System

Cell phone network system utilizes an architecture and several network protocols to communicate with the phone and other network operators. One of the protocol used to by cell phone network system to obtain the cell phone information of the user is by using signaling system. One of the protocols used by the signaling system is signaling system 7. Signaling System 7 is an international telecommunications standard that describe the activity of network elements in a public switched telephone network (PSTN) during information exchange on a digital signaling network.

The experiment was conducted by the Tobias [7] to utilize signaling system 7 to obtain the user location. By using the flaws in the signaling system authentication method which allows everyone to access the user's information as long as the request is made by network operators. The network operators can access the user's cell phone number and the user current location on the scale of a city.

The method created by Tobias can be used everywhere as the signaling system is used by all network operators. However, the information obtained by using the signaling system will only provide the city where the user is currently at. In addition, in order to access the user's information, the spy tool created by Tobias must know the cell phone number used by the user. Also, the introduction of VoIP (Voice over Internet Protocol) as part of network operators creates a change in the signaling system implementation by adding several new security protocols to reduce a chance of security breach from an unauthorized user.

3.2 Localization using motion Sensors

One of the available sensors on the smart phone nowadays is motion sensor. The motion sensors are often used as an addition for GPS to increase the accuracy of phone localization. In the last couple years, several experiments were conducted to find out the quality of phone localization using motion sensors and other additional parts of the phone to measure the location of the phone without using the GPS.

An experiment was conducted by Shala and Rodriguez [9] to measure the location of the user using accelerometer during an indoor situation. By calculating the acceleration and distance of movement to the map of the building, the application created by Shala and Rodriguez can measure the location of the user. However, the experiment is limited to an indoor situation where the map of the building is known by the observer and the spy tool. In addition, the noise produced by the accelerometer can reduce the quality of the information, for example, the accelerometer can produce a value based on the vibration caused by the movement of the body even though the user is not moving. Also, the gravity value is added to the value of the accelerometer that can cause an error in measurement especially when the orientation of the phone changes.

The last attempt was made by Han et.al. [11] to measure the user location by utilizing accelerometer. The application made by Han et al. will collect a big amount of data to be used as pre-collected information to predict the movement path. In order to find out the type of movement and reduce the noise, the application used k-means to create clusters of data where the data will be put on the cluster based on the acceleration of x, y and z-axis and the time. After the clustering phase, the acceleration data will be calculated to measure the displacement. Han et.al. proposed displacement by using the trajectory probability and find out which one has the closest similarities to the real life situation which it the map of the city. However, in order to increase the accuracy, the amount of pre-collected data is important, as the calculation of the probability needs to use the pre-collected data to reduce a number of candidates needed to find the trajectory and computation time to calculate the candidate and select the one with the highest probability score.

3.3 Localization using Battery Consumption

A new technique was developed in order to find the phone's location without utilizing the phone network system and motion sensors. The experiment conducted by Michalevsky et al [12] utilizes the battery consumption to measure the movement of the user. By measuring the spike in battery consumption during the hand-off from one cell tower to another tower, the application made by Michalevsky et al. will measure the trajectory. In addition, the battery level can also inform the spy tool that the signal strength is decreasing due to the distance between the phone and the cell tower or the location of the phone which surrounded by many tall buildings.

However, the quality of the measurement is based on the amount of pre-collected data and the

amount of knowledge of the path that the user often uses on the daily basis. Also, there are several spikes caused by other battery usages, such as phone call or display, that can introduce noise to the measurement, especially when the spike on the battery consumption is similar to the hand-off process.

3.4 Localization using Low Frequency Signal

In order to compensate the problem with the indoor experiment using accelerometer which has a low accuracy on determining the user location when the user is not moving, another experiment was conducted by Ayllon et.al. [10] to measure the location of the phone by by using speaker to produce a low frequency signal and microphone to capture the ripple of the signal after bouncing from the wall. By calculating the difference between the signal strength from the sender (speaker) and receiver (microphone), Ayllon et.al. can measure the location of the phone against the wall. Ayllon also utilizes the secondary microphone which acts as noise canceling microphone to measure the orientation of the phone and the difference in signal strength on the main microphone and the secondary microphone. However, as the spy tool relies on the signal captured by the phone, the distance between the wall and the phone is important, as the farther the distance between the phone and the wall, the signal will get weaker and can possibly be lost because the microphone cannot capture the ripple. In addition, there are many external noises around the phone that can get picked up and reduce the quality of the measurement.

3.5 Research Gap

Many previous attempts were made in order to find the user location without using GPS or other functions that need user's permission to be accessed. Four different approaches were used to find the best accuracy on the trajectory prediction. in addition, similar approach can be used for different type of situation such as the location and movement type.

However, the previous attempts have similar flaws in the terms of accuracy and data precision. The phone localization using signaling system has a low precision in terms of the phone localization that can only locate the city where the user is currently at. On the other hand, the phone localization using the motion sensors can only be used on indoor situation using pre-collected data. However, the accuracy of the localization depends on the user movement and the data readings from the sensor during the data collection.

Another approach using the battery consumption also suffers the problem on the accuracy of the data. The accuracy of the phone localization is based on the amount of the pre-collected data and the amount of knowledge that the spy tool has on the path that the user taken during the data collection. Additionally, the phone localization using the low frequency signal produced by the speaker depends on the distance between the phone and the object where the signal will bounce off and the amount of noise produced by the room during the data collection.

Another problem from the previous attempts is the lack of sensor fusion to increase the quality of the data. All approaches used basic sensor without adding another sensor as a new variable to increase the reliability of the measurement. As many sensors embedded on the smart phone can be accessed without user's permission [14]. Thus a new experiment should be conducted to find the impact on the sensor fusion on the measurement accuracy and how the spy tool performs on a different type of situation, such as the type of the movement or the location of the user.

Chapter 4

Method

An Android application will be developed to extract raw data from motion sensors on 2 main scenarios, controlled and uncontrolled environment. The uncontrolled environment is the test environment that has several random internal and external variables that will affect the quality of data and cannot be measured or controlled by examiner.

The examples of uncontrolled environments are when the target is cycling, driving and walking. The controlled environment is the environment where the examiner can control the path, the time and the speed of the movement. The example of controlled experiment is by using robot in particular type of terrain chosen by examiner. The motion sensors used are accelerometer, gyroscope, magnetometer and combination of them. The data will be collected per 50 milliseconds during the movement.

In addition, the data from Global Positioning System (GPS) will be taken for comparison with the data from sensors and the real path taken during experiment to determine the accuracy of the data from motion sensors.

4.1 Data Collection

Data from motion sensors will be combined based on the advantages of each sensor using the formula provided by the Android developer site [14] which is Linear Acceleration calculation using gyroscope as a complementary filter. Accelerometer and magnetometer data will be combined to create a rotational matrix and measure the orientation value and the gyroscope will be added to deduct the gravity value when the orientation changes. The gyroscope will be added in short time intervals as the gyroscope will drift and produce more noise over time. The magnetometer and accelerometer data will be used for a long period after the adjustments from the gyroscope.

In addition, the accelerometer and magnetometer data will be used to calculate the azimuth orientation that will be used to calculate the direction of the movement using Android based on the formula

given by the Android Developer site [23]. As the azimuth value is on the horizontal plane, the error produced by the gravity will not be added to the value of the azimuth.

4.2 Data Processing

Data taken from the sensors contains a different type of noises that can cause the data to suddenly increase during a period of time. Accelerometer adds gravitational value on the measurement, the magnetometer adds strong magnetic field around the sensor to the measurement in addition to the earth's magnetic field value, the gyroscope drifts over time and increase the errors in the measurement. In order to eliminate the noise, the usage of a filter is needed. Mean filter is one type of filter that utilize the usage of the neighbourhood data to eliminate the sudden spike of data or outliers inside the measurement. Mean filter is also one type of low-frequency filter that usually used on motion sensors such as accelerometer and magnetometer.

The mean filter will be used during the sensor fusion to eliminate the noise being added to the result of the combination of sensors. The linear acceleration and orientation value from the sensor fusion will utilize the mean filter before the combination.

In addition, after the data collection, some of data will be presented in graph for data interpretation. In order to smoothen the data and remove the rest of outliers from data, a moving average trend line function will be used. The moving average trend line function used in the experiment will utilise 1 second or equal to 20 neighbours (period) to find average value, in order to reduce the outliers but also prevent over-smoothen on the data at the same time.

A manual clustering will also be made for orientation graph which has range from 0 to 360 degrees to differentiate the direction of the movement using 'compass based clustering'. The value from moving average trend line will be added to the nearest compass direction. The position of each point of compass in degree can be seen below

4.3 Data Filtering

Data taken from the sensors contains a different type of noises that can cause the data to suddenly increase during a period of time. Accelerometer adds gravitational value on the measurement, the magnetometer adds strong magnetic field around the sensor to the measurement in addition to the earth's magnetic field value, the gyroscope drifts over time and increase the errors in the measurement. In order to eliminate the noise, the usage of a filter is needed. Mean filter is one type of filter that utilize the usage of the neighbourhood data to eliminate the sudden spike of data or outliers inside the measurement. Mean filter is also one type of low-frequency filter that usually used on motion sensors such as accelerometer and magnetometer.

The mean filter will be used during the sensor fusion to eliminate the noise being added to the result of the combination of sensors. The linear acceleration and orientation value from the sensor fusion will utilize the mean filter before the combination.

In addition, after the data collection, some of data will be presented in graph for data interpretation. In order to smoothen the data and remove the rest of outliers from data, a moving average trend line function will be used. The moving average trend line function used in the experiment will utilise 1 second or equal to 20 neighbours (period) to find average value, in order to reduce the outliers but also prevent over-smoothen on the data at the same time.

A manual clustering will also be made for orientation graph which has range from 0 to 360 degrees to differentiate the direction of the movement using 'compass based clustering'. The value from moving average trend line will be added to the nearest compass direction. The position of each point of compass in degree can be seen below

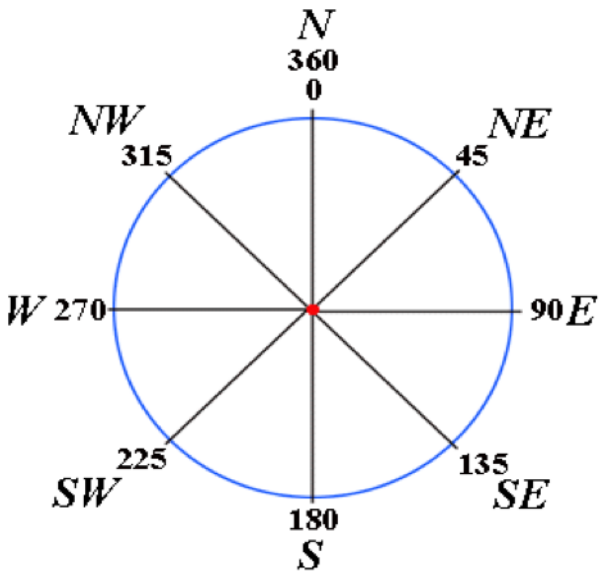

Figure 4.1: Point of Compass Azimuth Value

4.4 Distance Calculation

In order to accurately track the user, two main parts must be available for movement mapping, which are direction and distance. After adding the direction to each cluster, the movement mapping must add the distance of movement during straight path. To find the accurate distance, the step counter will be used for walking movement and the displacement calculation based on the linear acceleration value will be used for cycling, driving and robot movement. For walking experiment, the calculation of the distance is,

$$Distance = (Final\ step\ count - Starting\ step\ count) * 0.7\ meter$$

The 0.7 meter is the average step length of human [24], which is used as global step length value.

In addition, for the other experiments, the distance calculation will utilise displacement formula as there is no dedicated sensor to read the moving activity data other than step. The displacement formula used for experiment is,

$$Distance = x_0 + v_0 t + \frac{1}{2}at^2$$

4.5 Data Analysis

After the data processing for direction and distance, a manual movement pattern mapping will be created in order to compare the result of the motion/motion sensors, GPS and the real path taken during experiment. The comparison will be used to analyse the accuracy of motion sensors and GPS, the performance of motion sensors in terms of user tracking and the security issues that can happen from the utilisation of motion sensors for user tracking. The result will be used to create the suggestion for Android and information for future works.

Chapter 5

Experiment

In order to be able to run experiment, several guidelines and tools must be made. By creating the guidelines and tools, the data collected can be extracted based on needs for further analysis.

5.1 Android Code

The application will be made to extract data from the motion sensors inside the phone based on Android Developers code on motion sensors [14] [18] [3] [23]. The usage of motion sensors API from Android OS usage is essential to access the information from the sensors. In order to extract a higher accuracy and more reliable information, the motion sensors will be combined using the sensor fusion. There are 2 main objectives needed on data extraction, the fusion of sensors to find the linear acceleration and the azimuth orientation. The linear acceleration will be used to measure the displacement in order to find the amount of distance that the user taken over a period of time.

5.1.1 Developing the Linear Acceleration Sensor Fusion

The development of the linear acceleration starts by calling 4 sensors, which are accelerometer, gyroscope, magnetometer and gravity. The gravity is a virtual sensor that returns the value of gravity from each of accelerometer axis value. The accelerometer, gyroscope and magnetometer are physical sensors embedded inside the smart phone. In addition, a filter class will be called and initialized for future use.

After the initialization and listener registration for real-time update, the event of the sensors will be made, which called onSensorChanged. The data will be added to an array that will contain the value from x, y and z-axis. After the addition of the data into the array, the mean filter function will be called to filter the data before the data fused later on. The code can be seen below,

```
@Override
public void onAccelerationSensorChanged(float[] acceleration, long timeStamp)
{
```

```
    // Get a local copy of the raw magnetic values from the device sensor.
    System.arraycopy(acceleration, 0, this.acceleration, 0,
            acceleration.length);
    this.acceleration = meanFilterAcceleration.filterFloat(this.acceleration);
@Override
public void onMagneticSensorChanged(float[] magnetic, long timeStamp)
{
    // Get a local copy of the raw magnetic values from the device sensor.
    System.arraycopy(magnetic, 0, this.magnetic, 0, magnetic.length);

    this.magnetic = meanFilterMagnetic.filterFloat(this.magnetic);
}
@Override
public void onGravitySensorChanged(float[] gravity, long timeStamp)
{
    // Get a local copy of the raw magnetic values from the device sensor.
    System.arraycopy(gravity, 0, this.gravity, 0, gravity.length);

    this.gravity = meanFilterGravity.filterFloat(this.gravity);

    calculateOrientation(); //to calculate the orientation of gravity
}
```

After the initialisation of the onSensorChanged event, the first step of fusion happens with the combination of magnetometer and gravity sensor. Both sensors will be added to the getOrientation function to find the orientation value which contains the value of the gravity and where it is located on the x, y and z-axis based on the orientation of the phone. The code can be seen below,

```
private void calculateOrientation()
{
    if (SensorManager.getRotationMatrix(rotationMatrix, null, gravity,
            magnetic)) //create Rotation Matrix
    {
        //create orientation array consists of 3 types of rotation
        SensorManager.getOrientation(rotationMatrix, orientation);
        hasOrientation = true;
    }
}
```

After the combination of the magnetometer and gravity, the code for gyroscope is added. In order to find the orientation after the data from the sensor is taken, the rotation matrix must be created, however, Android OS does not provide a built-in function to create a rotation matrix for the gyroscope, thus a set of functions must be created in order to create the rotation matrix for the gyroscope. In order to create rotation matrix, 2 steps must be taken, the first one is to create the gyroscope matrix using the multiplication of identity matrix with the rotation matrix from the combination of the gravity and magnetometer.

The second step is to create the rotation vector must be calculated. The rotation vector is calculated using the angular speed multiply by the current time to create half angle theta which will be needed to create the delta vector, the code can be seen below,

```
// Calculate the angular speed of the sample
omegaMagnitude = (float) Math.sqrt(Math.pow(gyroscope[0], 2)
      + Math.pow(gyroscope\cite{1}, 2) + Math.pow(gyroscope\cite{2}, 2));

// Normalize the rotation vector if it's big enough to get the axis
if (omegaMagnitude > EPSILON)
{
    gyroscope[0] /= omegaMagnitude;
    gyroscope\cite{1} /= omegaMagnitude;
    gyroscope\cite{2} /= omegaMagnitude;
}

thetaOverTwo = omegaMagnitude * timeFactor;
sinThetaOverTwo = (float) Math.sin(thetaOverTwo);
cosThetaOverTwo = (float) Math.cos(thetaOverTwo);

deltaVector[0] = sinThetaOverTwo * gyroscope[0];
deltaVector\cite{1} = sinThetaOverTwo * gyroscope\cite{1};
deltaVector\cite{2} = sinThetaOverTwo * gyroscope\cite{2};
deltaVector\cite{3} = cosThetaOverTwo;
```

After the creation of the vector, the rotation matrix can be calculated by using the built-in function from sensor manager to calculate the rotation matrix from the vector. Therefore, 2 matrices have been created, which are gyroscope matrix and vector rotation matrix, thus both vector will be multiplied in order to combine the value from gravity, magnetometer and gyroscope into one vector. After the multiplication, the function from sensor manager will be called to create the orientation from the sensor fusion. The code can be seen below,

```
SensorManager.getRotationMatrixFromVector(deltaMatrix, deltaVector);
gyroMatrix = matrixMultiplication(gyroMatrix, deltaMatrix);
SensorManager.getOrientation(gyroMatrix, gyroOrientation);
```

The next step is to add a complimentary filter to stabilize the problem with the negative and positive radian transition value when the value from the gyroscope is positive but the value from the combination of gravity and magnetometer is negative due to the limit of the radian. By adding $2 * \pi$ to the negative value before the fusion of gyroscope and orientation from the gravity and magnetometer, and remove the $2 * \pi$ if the final value is higher than 180 degrees can help preventing the error from the transition. The code can be seen below,

```
// azimuth
if (gyroOrientation[0] < -0.5 * Math.PI && orientation[0] > 0.0)
{
    fusedOrientation[0] = (float) (FILTER_COEFFICIENT
            * (gyroOrientation[0] + 2.0 * Math.PI) + oneMinusCoeff
            * orientation[0]);
    fusedOrientation[0] -= (fusedOrientation[0] > Math.PI) ? 2.0 * Math.PI
            : 0;
}
else if (orientation[0] < -0.5 * Math.PI && gyroOrientation[0] > 0.0)
{
    fusedOrientation[0] = (float) (FILTER_COEFFICIENT
            * gyroOrientation[0] + oneMinusCoeff
            * (orientation[0] + 2.0 * Math.PI));
    fusedOrientation[0] -= (fusedOrientation[0] > Math.PI) ? 2.0 * Math.PI
            : 0;
}
else
{
    fusedOrientation[0] = FILTER_COEFFICIENT * gyroOrientation[0]
            + oneMinusCoeff * orientation[0];
}
// pitch
if (gyroOrientation\cite{1} < -0.5 * Math.PI && orientation\cite{1} > 0.0)
{
    fusedOrientation\cite{1} = (float) (FILTER_COEFFICIENT
            * (gyroOrientation\cite{1} + 2.0 * Math.PI) + oneMinusCoeff
```

```
            * orientation\cite{1});
    fusedOrientation\cite{1} -= (fusedOrientation\cite{1} > Math.PI) ? 2.0 * Math.PI
            : 0;
}
else if (orientation\cite{1} < -0.5 * Math.PI && gyroOrientation\cite{1} > 0.0)
{
    fusedOrientation\cite{1} = (float) (FILTER_COEFFICIENT
            * gyroOrientation\cite{1} + oneMinusCoeff
            * (orientation\cite{1} + 2.0 * Math.PI));
    fusedOrientation\cite{1} -= (fusedOrientation\cite{1} > Math.PI) ? 2.0 * Math.PI
            : 0;
}
else
{
    fusedOrientation\cite{1} = FILTER_COEFFICIENT * gyroOrientation\cite{1}
            + oneMinusCoeff * orientation\cite{1};
}
// roll
if (gyroOrientation\cite{2} < -0.5 * Math.PI && orientation\cite{2} > 0.0)
{
    fusedOrientation\cite{2} = (float) (FILTER_COEFFICIENT
            * (gyroOrientation\cite{2} + 2.0 * Math.PI) + oneMinusCoeff
            * orientation\cite{2});
    fusedOrientation\cite{2} -= (fusedOrientation\cite{2} > Math.PI) ? 2.0 * Math.PI
            : 0;
}
else if (orientation\cite{2} < -0.5 * Math.PI && gyroOrientation\cite{2} > 0.0)
{
    fusedOrientation\cite{2} = (float) (FILTER_COEFFICIENT
            * gyroOrientation\cite{2} + oneMinusCoeff
            * (orientation\cite{2} + 2.0 * Math.PI));
    fusedOrientation\cite{2} -= (fusedOrientation\cite{2} > Math.PI) ? 2.0 * Math.PI
            : 0;
}
else
{
    fusedOrientation\cite{2} = FILTER_COEFFICIENT * gyroOrientation\cite{2}
```

```
            + oneMinusCoeff * orientation\cite{2};
}
```

After the final filtering, the gyroscope matrix will be updated from the value of the fusedOrientation to compensate the gyro drift for the next usage. Next, the value of the fusedOrientation will be calculated to find the gravity value for each axis based on trigonometry, linear algebra and rotation matrices. For x-axis, the equation is,

$$X = Earth\ Gravity * \ -cos(pitch) * sin(roll)$$

For y axis, the equation is,

$$Y = Earth\ Gravity * \ -sin(pitch)$$

For z axis, the equation is,

$$Z = Earth\ Gravity * \ -cos(pitch) * cos(roll)$$

After each component is calculated, the value of the gravity will be subtracted from accelerometer data based on each axis. In addition, a filter will be added to the result of subtraction to smoothen the data. The code can be seen below,

```
// values[0]: azimuth, rotation around the Z axis.
// values\cite{1}: pitch, rotation around the X axis.
// values\cite{2}: roll, rotation around the Y axis.
components[0] = (float) (SensorManager.GRAVITY_EARTH
        * -Math.cos(fusedOrientation\cite{1}) * Math
        .sin(fusedOrientation\cite{2}));

// Find the gravity component of the Y-axis
// = g*-sin(pitch);
components\cite{1} = (float) (SensorManager.GRAVITY_EARTH * -Math
        .sin(fusedOrientation\cite{1}));

// Find the gravity component of the Z-axis
// = g*cos(pitch)*cos(roll);
components\cite{2} = (float) (SensorManager.GRAVITY_EARTH
        * Math.cos(fusedOrientation\cite{1}) * Math
        .cos(fusedOrientation\cite{2}));

// Subtract the gravity component of the signal
```

```
// from the input acceleration signal to get the
// tilt compensated output.
linearAcceleration[0] = (this.acceleration[0] - components[0]);
linearAcceleration\cite{1} = (this.acceleration\cite{1} - components\cite{1});
linearAcceleration\cite{2} = (this.acceleration\cite{2} - components\cite{2});
this.linearAcceleration = meanFilterLinearAcceleration.filterFloat(this.linearAccel
```

The full graph describing the steps taken to develop a linear acceleration value can be seen in Figure 5.1.,

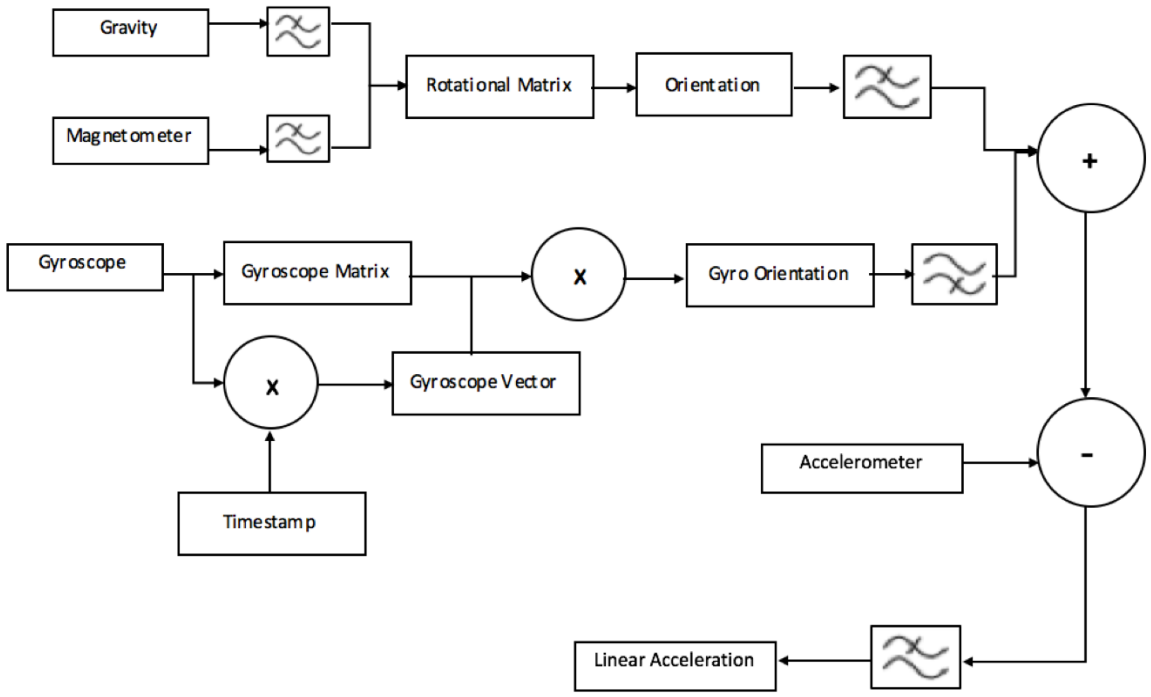

Figure 5.1: Flowchart of Linear Acceleration Calculation

5.2 Azimuth Calculation

In order to find the direction of the movement using the smart phone, the use of motion sensors is important. Magnetometer and Accelerometer are two of motion sensors that can measure the movement and the direction of the user. Accelerometer uses the motion on the X, Y and Z axis on the phone to measure the direction of the movement. On the other hand, the magnetometer uses the earth's magnetic field and measure the strength of the magnetic field on the X, Y and Z axis to find out the direction of the movement compared to the earth's magnetic north.

However, accelerometer and magnetometer have problems on the measurement of the direction. The accelerometer adds gravitational field value to the measurement and the magnetometer adds earth's magnetic north and other magnetic interference to the measurement. Thus, the direction measurement is erroneous.

In order to increase the accuracy of the direction movement, a combination of accelerometer and magnetometer is needed. The Android OS provides methods to combine both sensors. The combination of both sensors on the Android OS will produce a rotation matrix which consists of 3 types of rotation, azimuth, pitch and roll. The Android OS also adds low pass filter inside the combination to reduce the noise from accelerometer and magnetometer.

After the combination of the accelerometer and magnetometer, the value of orientation based on 3 types of rotation is produced. However, in order to find the direction of the user, only the azimuth orientation is needed as it is located on the horizontal plane and the value comes from the difference between earth's magnetic north and the direction of the movement. The code of the sensor fusion can be seen below,

```
System.arraycopy(Acceleration, 0, this.Acceleration, 0,
                Acceleration.length);
        this.Acceleration = meanFilterAcceleration.filterFloat(this.Acceleration);
System.arraycopy(magnetometer, 0, this.magnetometer, 0,
        magnetometer.length);
        this.magnetometer = meanFilterAcceleration.filterFloat(this.magnetometer);
        if(SensorManager.getRotationMatrix(mR, mI, Acceleration, magnetometer))
        {
            float[] mR2 = new float\cite{9};
            SensorManager.remapCoordinateSystem(mR,
                    SensorManager.AXIS_X, SensorManager.AXIS_Z,
                    mR2);
            SensorManager.getOrientation(mR2, mOrientation);
            float azimuthInRadians = mOrientation[0];
            azimuthInDegress = (float)
            (Math.toDegrees(azimuthInRadians) + 360) % 360;
        }
```

The figure representing the flowchart of the sensor fusion of the azimuth orientation can be seen in Figure 5.2.

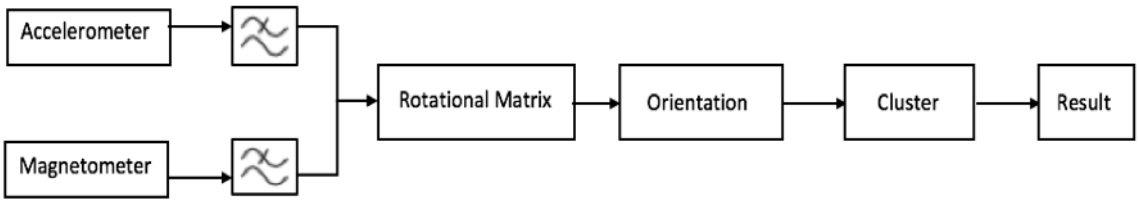

Figure 5.2: The Flowchart of Azimuth Orientation Sensor Fusion

5.3 Type of Movement

In order to determine the level of security issue created by the lack of necessary permission to access motion sensors, several test environments are created based on the real life situation to measure the accuracy level of user's tracking capability using motion sensors. However, an additional controlled environment is established in order to find out how the sensors work and discover the overall performance of the current state and issues with the sensors for future recommendation.

The uncontrolled environment consists of 3 main types of movement, which are walking, cycling and driving condition. The controlled environment will utilize Lego Mindstorms EV3 robot to control the movement variables, such as time and terrain. There will 3 main terrain used for the controlled environment, which are, ceramic tile floor, carpet floor and asphalt road to see how the density and texture of the terrain can add noise or reduce the quality of the sensors' output.

5.3.1 Uncontrolled Environment

The uncontrolled environment is the environment where the variables of the experiment cannot be changed or added. Random noises can appear during the experiment due to internal or external causes, such as bump in the road or a change in phone position due to car's movement or user's activity with the phone.

Walking

The walking experiment will be conducted by re-enacting real condition where the user will hold the phone on his/her hand during data retrieval. In order to simplify the data comparison, the user will take either 90 or 180 degrees turn to help identifying the change in the direction inside the data during analysis. The user will walk in normal pace which means that user can either walk or run during the experiment.

The distance of walking will be calculated using step counter sensor available on Android 6.0 from the initial location or the end of turning event until the next turn event or final location. The step will be multiplied by average step length of human to measure the overall distance.

Then, a comparison will be made on motion sensors value, GPS and real path in order to calculate the accuracy of the data. The difference of the data should not be more than 11.1m which is the fourth decimal place precision of latitude and longitude or lower than the typical accuracy value of commercially used GPS units [25]. Then, the data will be split into failed and acceptable region. The result acceptance will be divided into several regions which are,

Percentage	Status
< 50 %	Fail
50 % - 74 %	Acceptable
≥ 75 %	Accurate

The value of the acceptance percentage is obtained using,

$$\% = \frac{Acceptable}{Acceptable + Failed}$$

The result of the acceptance will be used to determine the recommendation and future works for walking based tracking.

Cycling

The cycling experiment will be conducted by following the normal cycling activity where the user will put the phone inside the pocket during data retrieval. In order to simplify the data analysis, the user will take either 90 or 180 degrees turn in order to easily identify the change in the direction during analysis.

As the step counter sensor cannot work for cycling, the distance will be calculated using linear acceleration data by fusing the motion sensors. The acceleration value will be processed using displacement formula in order to find the distance of the movement from the initial location or the end of turning event until the next turn event or final location.

Then, a comparison will be made on motion sensors value, GPS and real path in order to calculate the accuracy of the data. As there is no dedicated sensor available to calculate the distance, it is difficult to provide a decent accuracy for user tracking. Therefore, the result acceptance will be divided into 2 regions which are,

Percentage	Status
< 50 %	Fail
≥ 50 %	Acceptable

The analysis result will be provided based on the accuracy of the experiment and future recommendation will be added based the analysis.

Driving

The driving experiment will be conducted using the normal driving activity pattern where the user will put the phone inside the car during data retrieval. In order to easily identify the turn event, the user will take either 90 or 180 degrees turn.

As the step counter sensor also cannot work for driving, the distance will be measured using linear acceleration data based on motion sensors fusion. The linear acceleration value will be calculated using displacement formula to find the overall distance of the movement from the initial location or the end of turning event until the next turn event or final location.

Next, a comparison will be made on motion sensors value, GPS and real path in order to calculate the data accuracy. Because of the lack of dedicated sensor for distance measurement, it is difficult to have a good accuracy for user tracking. Thus, the result acceptance will be divided into 2 regions which are,

Percentage	Status
< 50 %	Fail
≥ 50 %	Acceptable

The discussion and recommendation will be provided based on the experiment result.

5.3.2 Controlled Experiment

The controlled environment is the environment where the variables of the experiment controlled or added. Noises can be identified during the experiment as the condition of the environment is set based on the experiment' needs, such as the movement time and the condition of the terrain.

Lego Mindstorms EV3

The experiment will utilise Lego Mindstorms EV3 robot in order to create driving situation within a controlled environment. As the amount of external noises from uncontrolled environment are quite high, a controlled environment test is needed to identify the noise that comes from different sources, such as, bump and vibration of the machine.

There will 3 types of terrain used to determine 3 major sources of noise which are bump, vibration and uneven terrain. The 3 types of terrain used for determining sources of noise are ceramic tile floor , carpet floor and asphalt road for uneven terrain.

The result of each terrain will be compared to the result of experiment on smooth condition which is on the top of wooden table. In order to make an accurate comparison, all experiment will be conducted within 20 seconds of movement. The robot will start at 0 and make an immediate stop at 20.

The result of the experiment will be used to determine the quality of the linear acceleration sensor to handle noise and whether the result from the sensor can become a security issue.

5.4 Data Extraction

In order to extract the data, several steps are taken in order to collect necessary data. The collection of data will be made by 1 device with stock Android OS, Motorola Nexus 6 with android 6.0. The application will be made to retrieve the value from sensors and add them inside a log file with .csv format. The log file will contain the value of the linear acceleration, step counter, azimuth orientation from fusion code, latitude and longitude from GPS, additional comment made during the experiment and current time. The example of the log file can be seen on Figure 5.3.

Generation	Timestamp	GoogleLinearX	GoogleLinearY	GoogleLinearZ	AccelX	AccelY	AccelZ	Steps	Degrees	Degrees Fusi	Latitude	Longitude	Comment	Time	Timemilis
0	1	-1.2416103	0.8510566	1.007802	-1.2416103	0.8510566	1.007802	943	175.03162	175.70258	-34.918278	138.60155		3:04:27	1.5061E+12
1	243	0.40736857	-0.8907151	-1.9002023	0.5330641	-0.9254308	-0.9916024	943	33.111877	169.70123	-34.918278	138.60155		3:04:27	1.5061E+12
2	315	-0.28097597	-0.60756636	-1.8042388	-0.1935876	-0.2831521	-1.7563548	943	171.22516	168.78064	-34.918278	138.60155		3:04:28	1.5061E+12
3	373	0.2761537	0.13549566	-1.0370636	0.22826967	0.1271162	-1.1747303	943	175.20868	175.69275	-34.918278	138.60155		3:04:28	1.5061E+12
4	423	0.15375735	0.63668346	1.462019	0.19086748	0.6570339	1.5362387	943	174.4018	174.99701	-34.918278	138.60155		3:04:28	1.5061E+12
5	473	0.7034673	0.7331672	0.9417329	0.8459223	0.66972065	1.1871381	944	171.50427	170.56427	-34.918278	138.60155		3:04:28	1.5061E+12
6	523	0.6278888	0.54105854	3.6132722	0.6231004	0.5638032	3.4684224	944	166.61133	164.55359	-34.918278	138.60155		3:04:28	1.5061E+12
7	573	1.0343621	0.51010275	1.4877968	1.0343621	0.51010275	1.4877968	944	161.34906	159.30847	-34.918278	138.60155		3:04:28	1.5061E+12
8	632	-0.6656691	-1.2748377	3.1962519	-0.6656691	-1.2748377	3.1962519	944	156.54858	153.70813	-34.918278	138.60155		3:04:28	1.5061E+12
9	695	-0.39456648	0.05334711	-1.5998964	-0.0629696	0.22333527	-0.4949722	944	153.38757	154.1153	-34.918278	138.60155		3:04:28	1.5061E+12
10	746	-0.19380015	-0.9022677	-0.6447153	-0.1650697	-0.9262097	-0.4567709	944	157.06793	156.44928	-34.918278	138.60155		3:04:28	1.5061E+12
11	796	-0.15760499	-0.10185671	-2.6184177	-0.157605	-0.1018567	-2.6184177	944	160.6651	160.75043	-34.918278	138.60155		3:04:28	1.5061E+12
12	846	-0.60749304	-0.1586852	-4.628112	-0.6613627	-0.0700998	-4.643674	944	163.36688	165.03632	-34.918278	138.60155		3:04:28	1.5061E+12
13	896	0.11716549	-0.31090927	-1.4917097	0.11716549	-0.3109093	-1.4917097	944	169.96173	171.54272	-34.918278	138.60155		3:04:28	1.5061E+12
14	947	-0.21767977	0.09236336	-0.16254091	-0.2176798	0.09236336	-0.1625409	944	173.08862	173.73523	-34.918278	138.60155		3:04:28	1.5061E+12
15	997	-0.4002182	0.1282053	1.9040031	-0.4002182	0.1282053	1.9040031	944	174.48615	174.81812	-34.918278	138.60155		3:04:28	1.5061E+12
16	1052	-0.7458943	0.82865334	4.8550797	-0.8296913	0.8178797	4.500738	945	175.8852	176.55048	-34.918278	138.60155		3:04:28	1.5061E+12
17	1125	-0.60438496	0.34850168	1.7337952	-0.604385	0.34850168	1.7337952	945	71.6926	180.3179	-34.918278	138.60155		3:04:28	1.5061E+12
18	1176	-0.29234147	-0.5049567	1.5830908	-0.2923415	-0.5049567	1.5830908	945	324.3312	180.45273	-34.918278	138.60155		3:04:28	1.5061E+12
19	1229	1.4478811	0.9306526	-1.7651052	1.6334317	0.959383	-1.663352	945	181.14601	181.04489	-34.918278	138.60155		3:04:28	1.5061E+12
20	1279	0.5726276	-1.2510762	-0.1472168	0.4672827	-1.2881863	-0.5841589	945	174.53198	174.45111	-34.918278	138.60155		3:04:28	1.5061E+12
21	1329	-0.10875982	-1.0176871	-3.805728	-0.1087598	-1.0176871	-3.805728	945	167.79785	166.57861	-34.918278	138.60155		3:04:29	1.5061E+12
22	1379	-1.0625418	-1.1736975	-2.1706538	-1.0625418	-1.1736975	-2.1706538	945	167.08441	167.88342	-34.918278	138.60155		3:04:29	1.5061E+12
23	1445	-0.33474118	-0.64200306	-2.6704698	-0.261718	0.3408165	-1.8755946	945	174.25122	174.69318	-34.918278	138.60155		3:04:29	1.5061E+12
24	1508	-0.034513056	0.5589843	1.0943046	0.2599738	0.3806162	1.5982842	945	143.57169	179.78424	-34.918278	138.60155		3:04:29	1.5061E+12
25	1573	-0.4714493	0.47710943	3.8592696	-0.3780754	0.556118	3.7120261	946	177.14429	175.49933	-34.918278	138.60155		3:04:29	1.5061E+12
26	1635	2.1223736	1.771029	1.5026083	2.1223736	1.771029	1.5026083	946	173.5824	170.29126	-34.918278	138.60155		3:04:29	1.5061E+12
27	1685	-0.03949675	-0.6391487	2.3269558	-0.5853747	-1.1730556	1.9007874	946	167.1073	163.72772	-34.918278	138.60155		3:04:29	1.5061E+12
28	1736	0.49230042	0.3464198	-1.2669053	0.49230042	0.3464198	-1.2669053	946	163.84106	163.78363	-34.918236	138.60153		3:04:29	1.5061E+12
29	1787	-0.12761796	-1.1779859	-1.249639	0.06272107	-1.2881191	-0.7959375	946	164.193	164.02661	-34.918236	138.60153		3:04:29	1.5061E+12
30	1848	0.0432083	-0.6818874	-1.470366	0.0432083	-0.6818874	-1.470366	946	163.69482	162.57092	-34.918236	138.60153		3:04:29	1.5061E+12

Linear Acceleration

Accelerometer minus Gravity

Azimuth

GPS

Time

Figure 5.3: The Log File Example on Nexus 6

After the creation of the log file, the data is further processed into different graph representing acceleration and direction of the movement. The linear acceleration graph represents the value of the acceleration after the gravity value is deducted from the accelerometer value. The difference between the linear acceleration and acceleration can be seen on Figure 5.4.

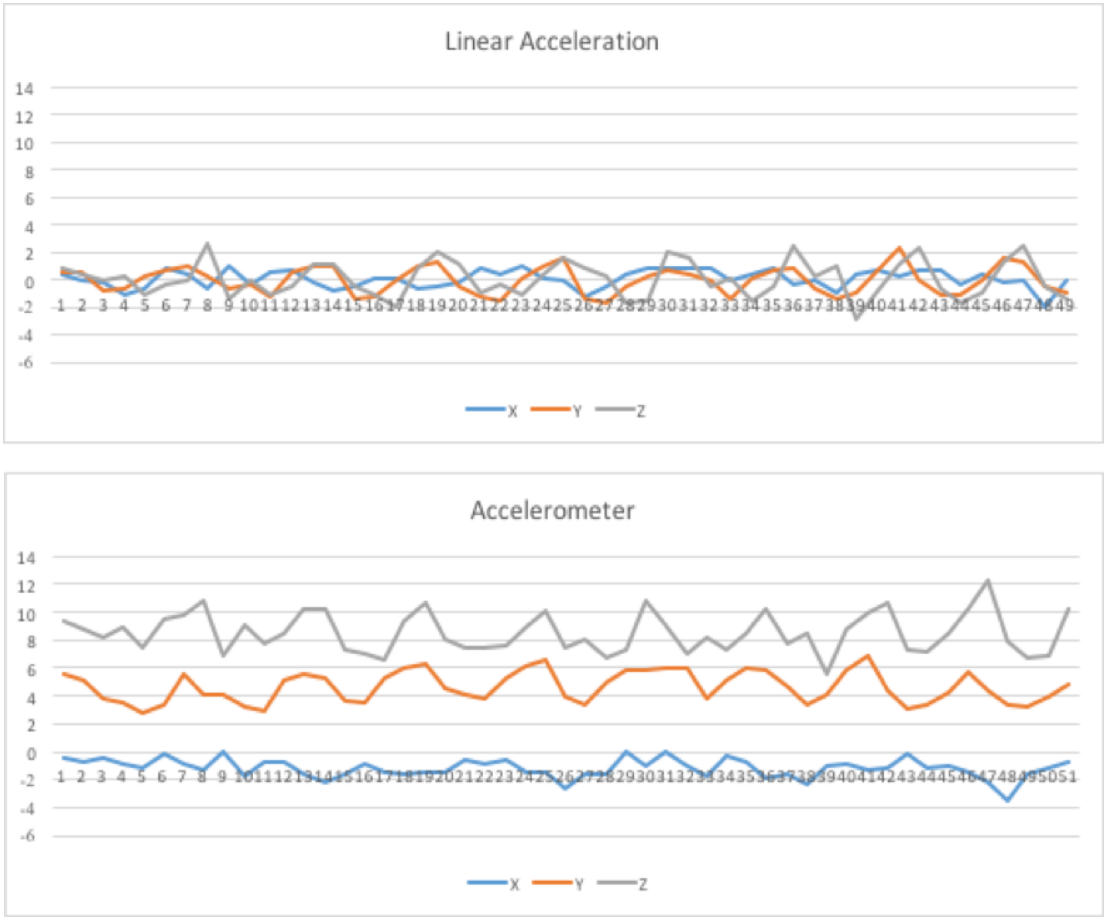

Figure 5.4: The Difference of Accelerometer and Linear Acceleration on Nexus 6 on Driving Movement

As the Figure 5.4 shows, the linear acceleration removes the gravity part from the accelerometer using the combination of magnetometer and gyroscope to determine the gravity value inside the accelerometer. The linear acceleration value will be used to determine the distance for cycling and driving activity and step counter for walking activity.

In addition, the orientation value will also be processed into a graph to determine the direction of the movement. In order to remove the outliers from the data causes by nearby magnetic interference, a mean filter is added to smoothen the data. The difference between unfiltered and filtered orientation value can be seen on Figure 5.5.

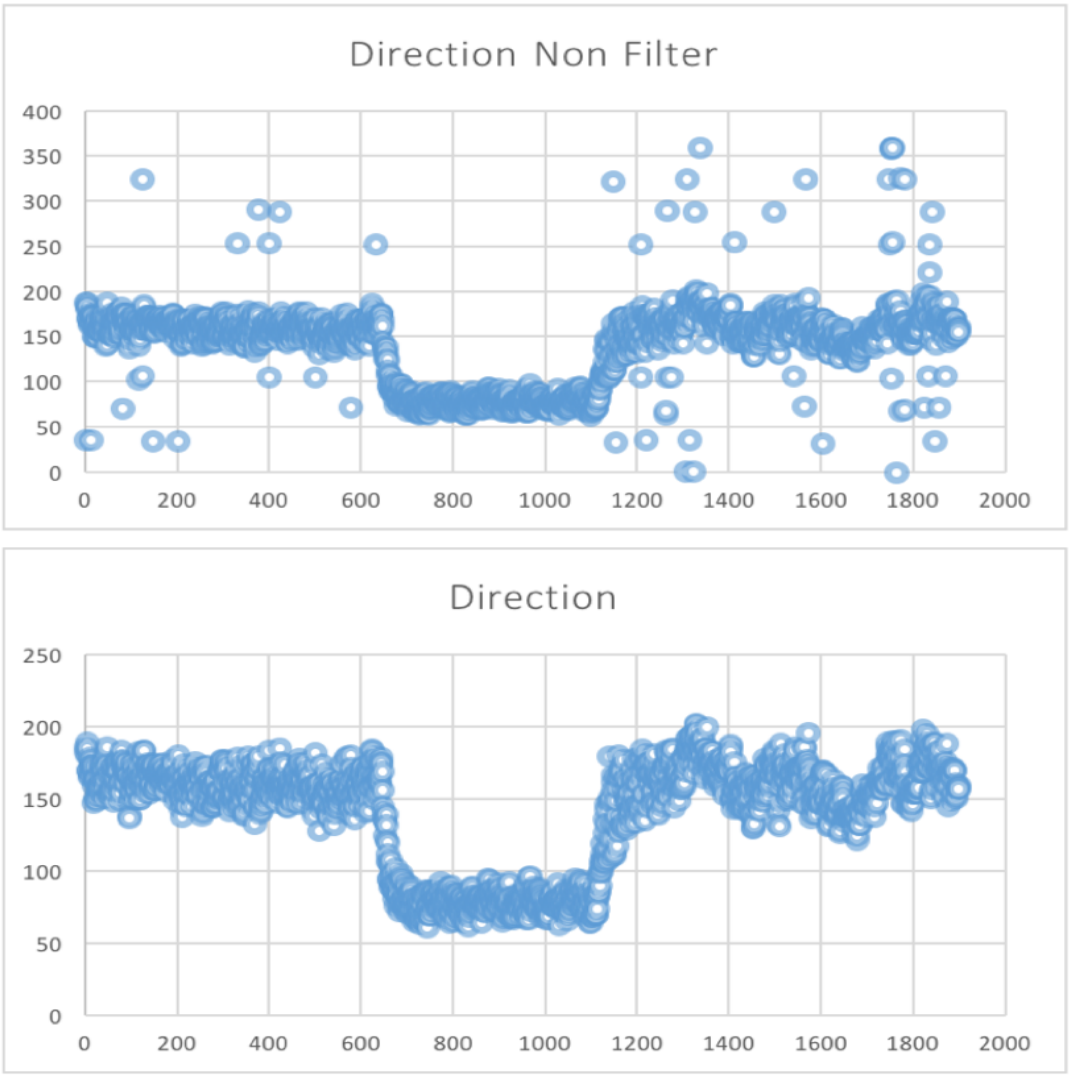

Figure 5.5: The Difference of Unfiltered and Filtered Orientation Value

The addition of the filter removes most of the outliers that are added into the graph during the experiment which can increase the quality of data for analysis. The orientation value will be used to determine the direction of the movement for all types of movement.

In order to analyse the quality of the motion sensors on tracking capability, the GPS value will also be added into the file for comparison. The GPS value will be converted into markers on Google maps to visualize the path taken during the experiment using darrinward.com Lat/Long tool [26]. The example of the GPS visualisation can be seen on Figure 5.6.

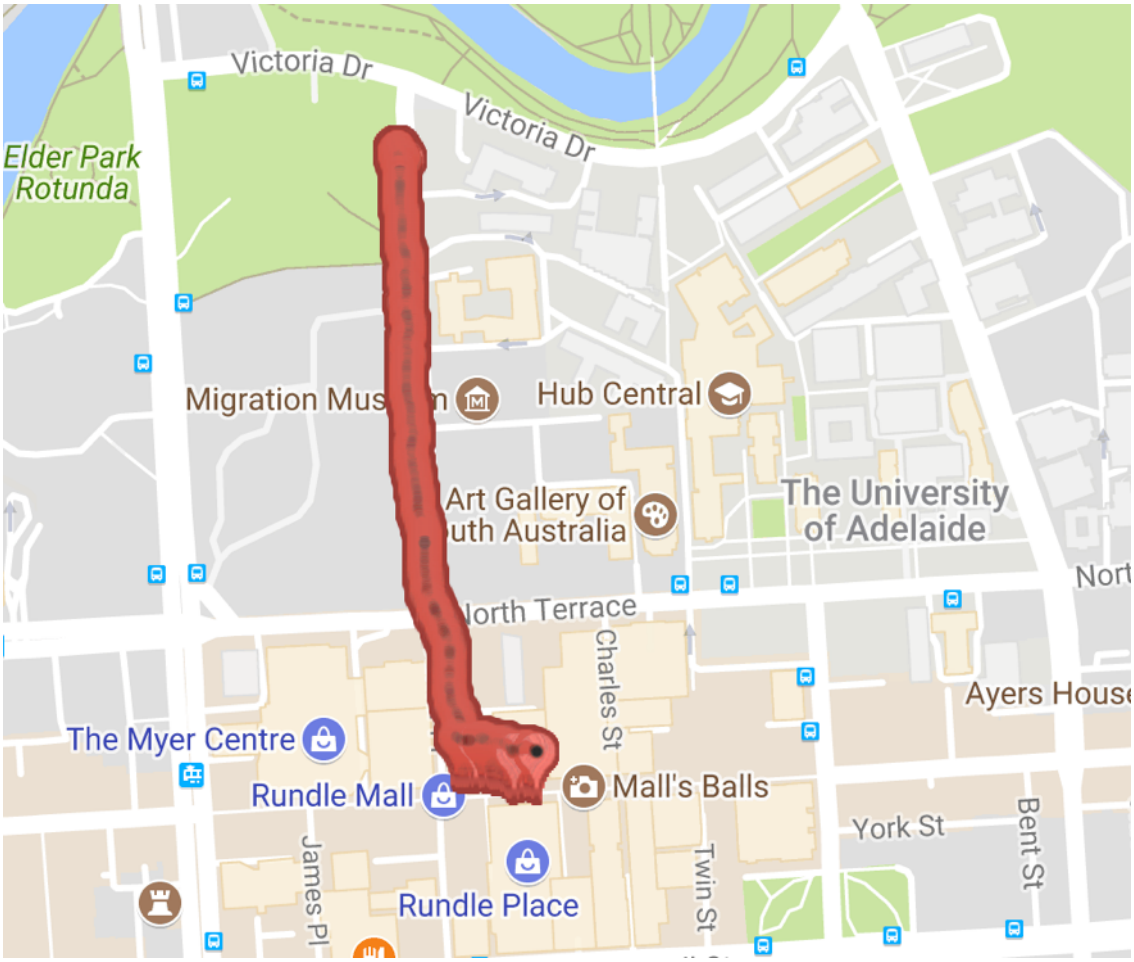

Figure 5.6: The GPS Visualisation on Google Maps

Next, the motion sensors and GPS result will be compared to the real path taken during the experiment and the accuracy of data will be calculated based on how the data performs in both direction and distance. The analysis result and recommendation will be made for each type of movement for future works.

Chapter 6

Analysis

Several experiments are conducted for uncontrolled and controlled experiments. The analysis for each experiment will be divided into several sections.

6.1 Uncontrolled Environment

Experiments are conducted to determine the quality of phone tracking with motion sensors. The motion sensors result and GPS sensors will be compared with the real path taken during the experiment to identify which result is closer to the real path

6.1.1 Walking

The walking experiments are divided into 2 parts, without and with obstruction to see the quality of motion sensors compared to GPS in different type of scenarios.

No Obstruction

The experiment was conducted using Android 6.0 on outdoor from hub central towards Nexus Building at Pulteney St. before data collection begun, the GPS must be turned on and the magnetometer was calibrated. The experiment used normal walking pace with phone located on hand (looking at phone).

The result of orientation from experiment will be divided into 2 main parts, the graph for direction and the graphical interpretation of GPS. Further, a simple calculation will be added to measure the accuracy of distance calculation between the log data from motion sensors and real distance taken during experiment using mapdevelopers.com [27] distance finder function. The graph for direction can be seen on Figure 6.1.

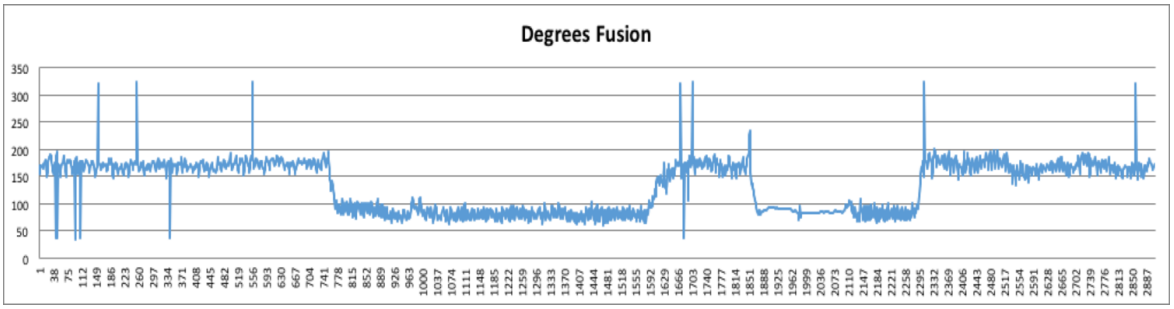

Figure 6.1: Orientation Graph

Figure 6.1 shows some changes during the experiment as the orientation value utilizes 360 degrees based value where 180 degree represents south and 360 or 0 degree represents north. In order to further reduce noises and outliers during the movement, a trend calculation using moving average based calculation will be made to smoothen the data. The result of the moving average calculation can be seen on black line on Figure 6.2.

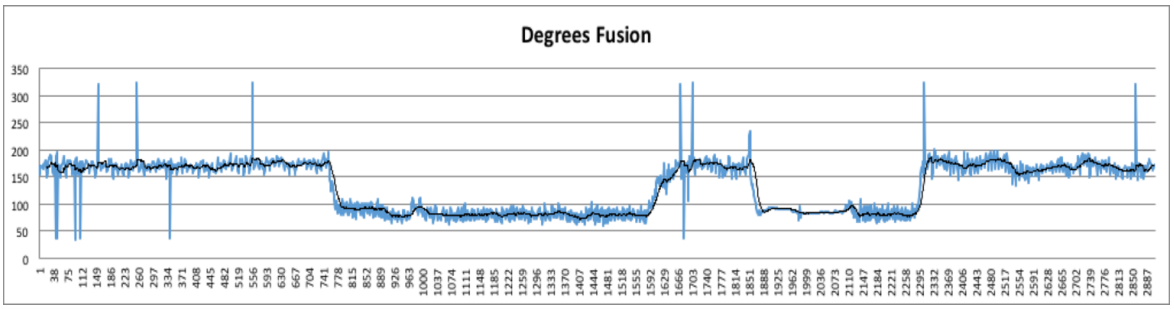

Figure 6.2: Orientation Graph with Trend Line

In addition, the experiment also takes longitude and latitude information from GPS for comparison. The graphical interpretation can be seen on Figure 6.3.

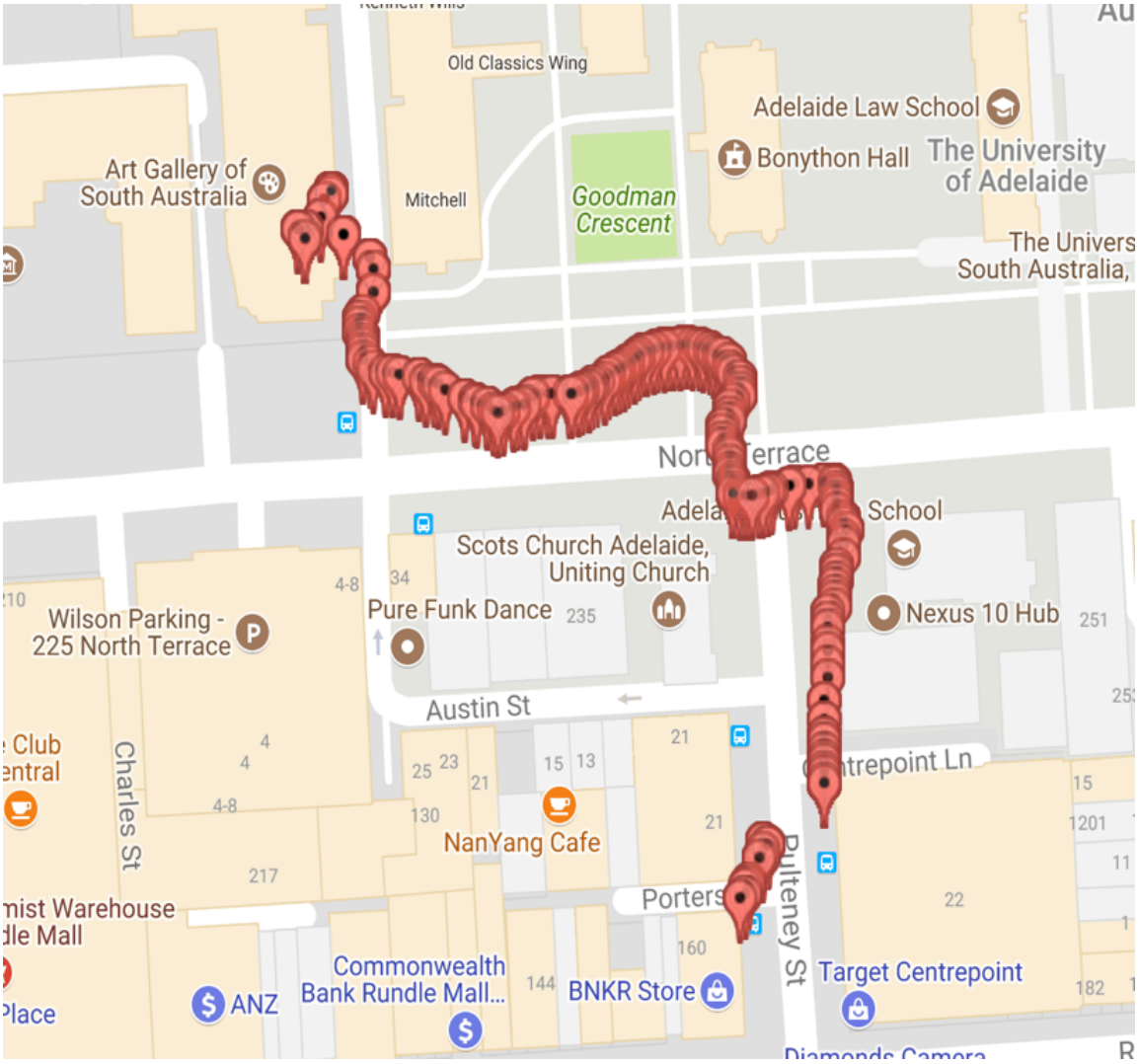

Figure 6.3: Graphical Interpretation of GPS

Based on Figure 6.3, the data shows some weird jumps on the location along the way. This phenomenon happens because GPS is trying to refresh and it takes time to find the exact location of the phone.

Based on the data from motion sensors and GPS, a comparison should be made to see the accuracy of data especially on the turning event. By adding the real path map taken during experiment into the comparison, the accuracy level of both results can be checked. The comparison can be seen on Figure 6.4.

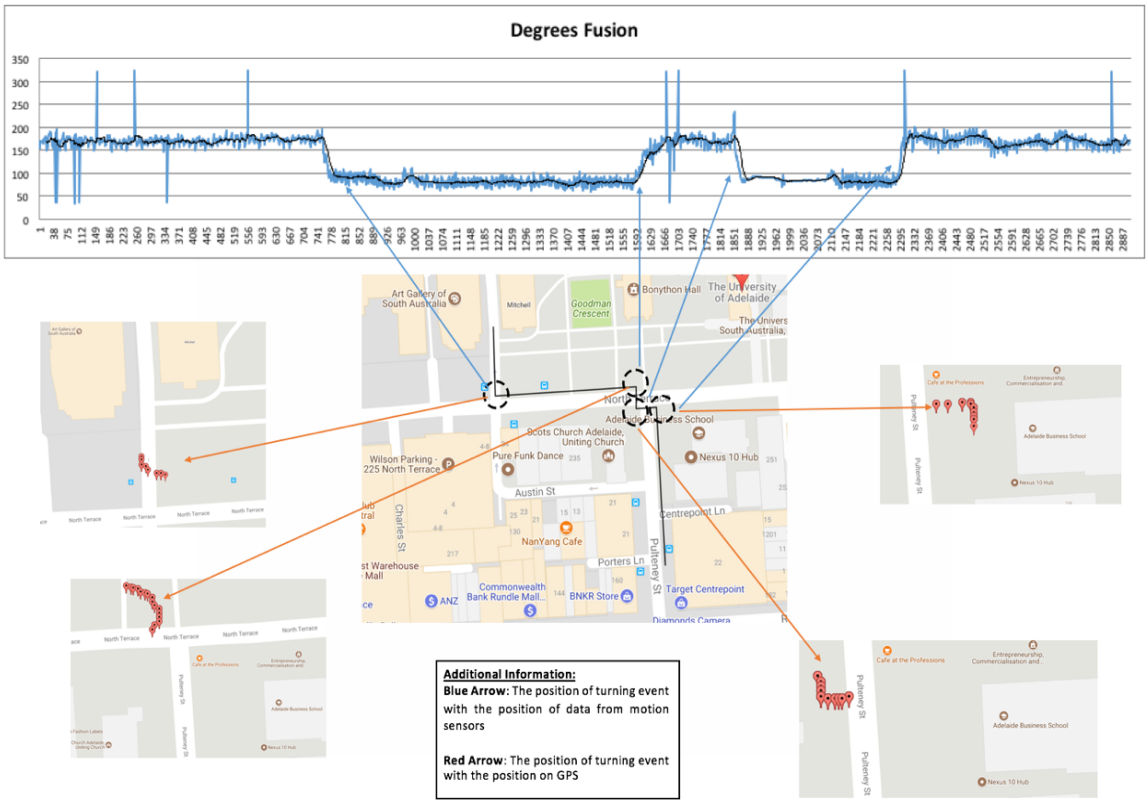

Figure 6.4: Comparison of the results

Figure 6.4 shows that both data from motion sensors and GPS shows good accuracy as both show the right turning event based on the real path taken during experiment. However, compared with the GPS that can pin-point the user location, motion sensors cannot tell the user location. with an assumption that the initial location is known, the motion sensors need to calculate the distance between the initial event and each turning event to estimate user's final location. Thus a comparison is made using the step formula and the distance taken using mapdevelopers.com based on real path. the result of comparison can be seen below,

Total Distance Calculation

Total Distance based on mapdevelopers.com: 305m

Total Steps taken based on log data: 442 steps

Distance taken based on average human stride length: 0.7m x 442 = 309m

The difference in value: 309 – 305 = 4m (Acceptable)

Initial Position to first turning

Total Distance based on mapdevelopers.com: 70.6m

Total Steps taken based on log data: 107 steps

Distance taken based on average human stride length: 0.7m x 107 = 74.9m

The difference in value: 74.9 – 70.6 = 4.3m (Acceptable)

First turning to second turning

Total Distance based on mapdevelopers.com: 101m

Total Steps taken based on log data: 114 steps

Distance taken based on average human stride length: 0.7m x 114 = 79.8m

The difference in value: 101 – 79.8 = 21.2m (Failed)

Second turning to third turning

Total Distance based on mapdevelopers.com: 23.1m

Total Steps taken based on log data: 49 steps

Distance taken based on average human stride length: 0.7m x 49 = 34m

The difference in value: 34 – 23.1 = 10.9m (Acceptable)

Third turning to fourth turning

Total Distance based on mapdevelopers.com: 19.1m

Total Steps taken based on log data: 33 steps

Distance taken based on average human stride length: 0.7m x 33 = 23m

The difference in value: 23 – 19.1 = 3.9m (Acceptable)

Fourth turning to final location

Total Distance based on mapdevelopers.com: 91.9m

Total Steps taken based on log data: 139 steps

Distance taken based on average human stride length: 0.7m x 139 = 97.3m

The difference in value: 97.3 – 91.9 = 5.4m (Acceptable)

Based on the distance calculation and the difference of distance between motion sensors and Google Maps distance tools provided by mapdevelopers.com, 5 out of 6 regions are within acceptable areas or have better accuracy compared to the range of commercially used GPS without interference. Furthermore, the regions with the acceptable result will be processed and changed into percentage value. The calculation of the acceptable result is,

$$\frac{5}{6}x100\% = 83.3\%$$

Based on the result acceptance table, the result of the experiment can be counted as accurate, as it is within the range of 75 % - 100 %. As the accuracy result of the motion sensors is better than the

accuracy of GPS, it can be concluded that motion sensors fusion is capable of tracking the walking movement without obstruction.

With Obstruction

Another experiment was conducted using Android 6.0 on outdoor from Commonwealth Bank at Rundle mall until in front the asphalt road next to art gallery. Before data collection begun, the GPS has been turned on and the magnetometer has been calibrated. The experiment used fast walking pace with phone located on hand (looking at phone). In the middle of the experiment, there will a tunnel as an obstruction to GPS. The obstruction will be used to check how the motion sensors compensate the issue and increase the accuracy of user tracking.

The result of orientation experiment will be the comparison of direction graph, graphical interpretation of GPS with the map of real path taken during experiment. Next, a simple calculation will be added to measure the accuracy of distance calculation between the step counter data from motion sensors and real distance taken during experiment using mapdevelopers.com distance finder function. The graph for direction can be seen on Figure 6.5.

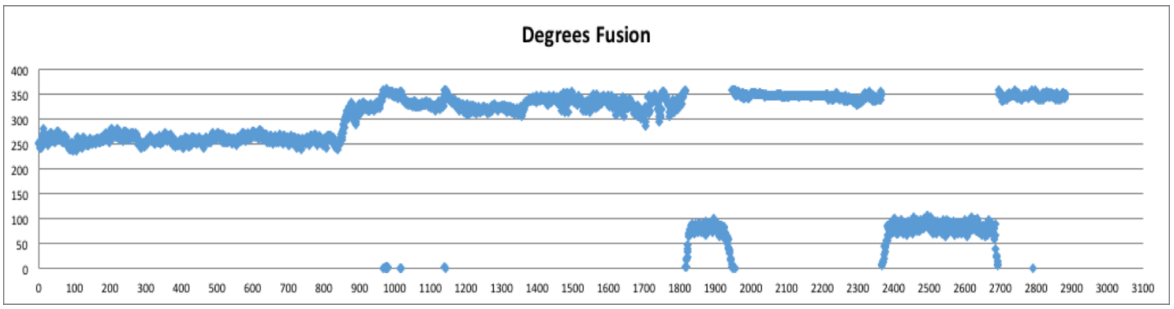

Figure 6.5: Orientation Graph

The figure 6.5 shows some changes during the experiment as the orientation value utilizes 360 degrees based value where 180 degree represents south and 360 or 0 degree represents north. in order to reduce the noise during the movement. A trend calculation using moving average based calculation will be made to smoothen the data. The result of the moving average calculation can be seen on black line on Figure 6.6.

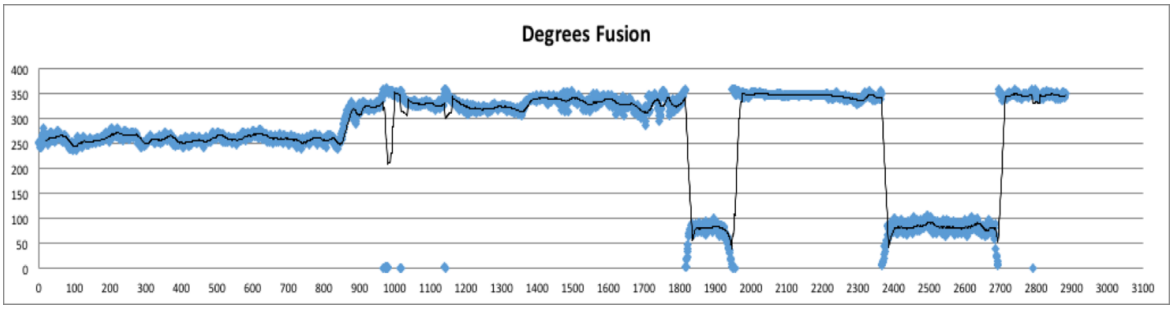

Figure 6.6: Orientation Graph with Trend Line

On the other hand, the experiment also takes longitude and latitude data from GPS for comparison. The graphical interpretation can be seen on Figure 6.7.

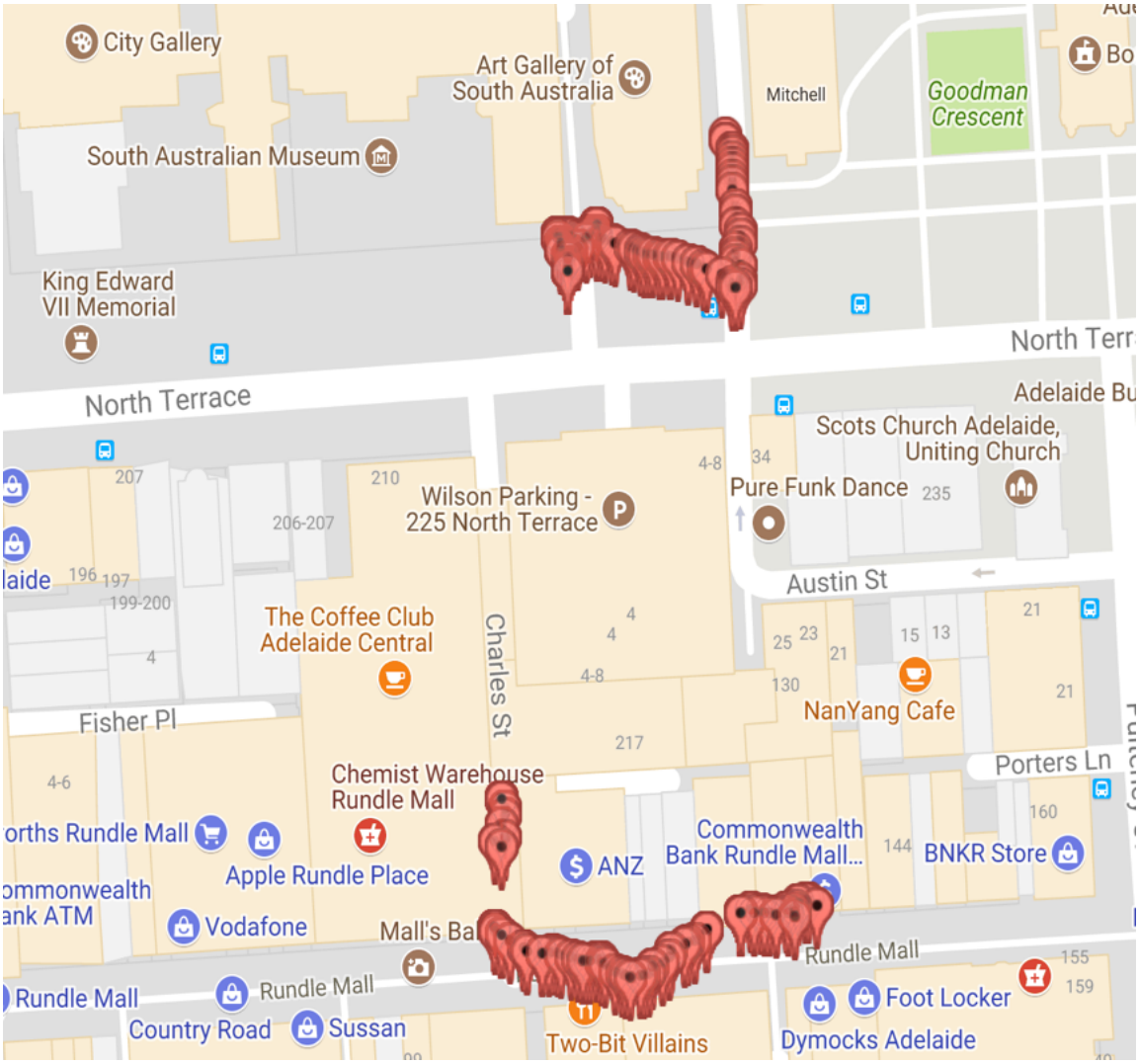

Figure 6.7: Graphical Interpretation of GPS

The Figure 6.7 shows some weird jump on the location along the way and the GPS even lose signal during the tunnel section between Rundle Mall and North Terrace and tried to recover after a while which resulted in missing parts in some areas.

Based on the data from motion sensors and GPS, the accuracy of data especially on the turning event must be checked. Thus, a comparison is made with the real path map taken during experiment to measure the accuracy level of both results. The comparison can be seen on Figure 6.8.

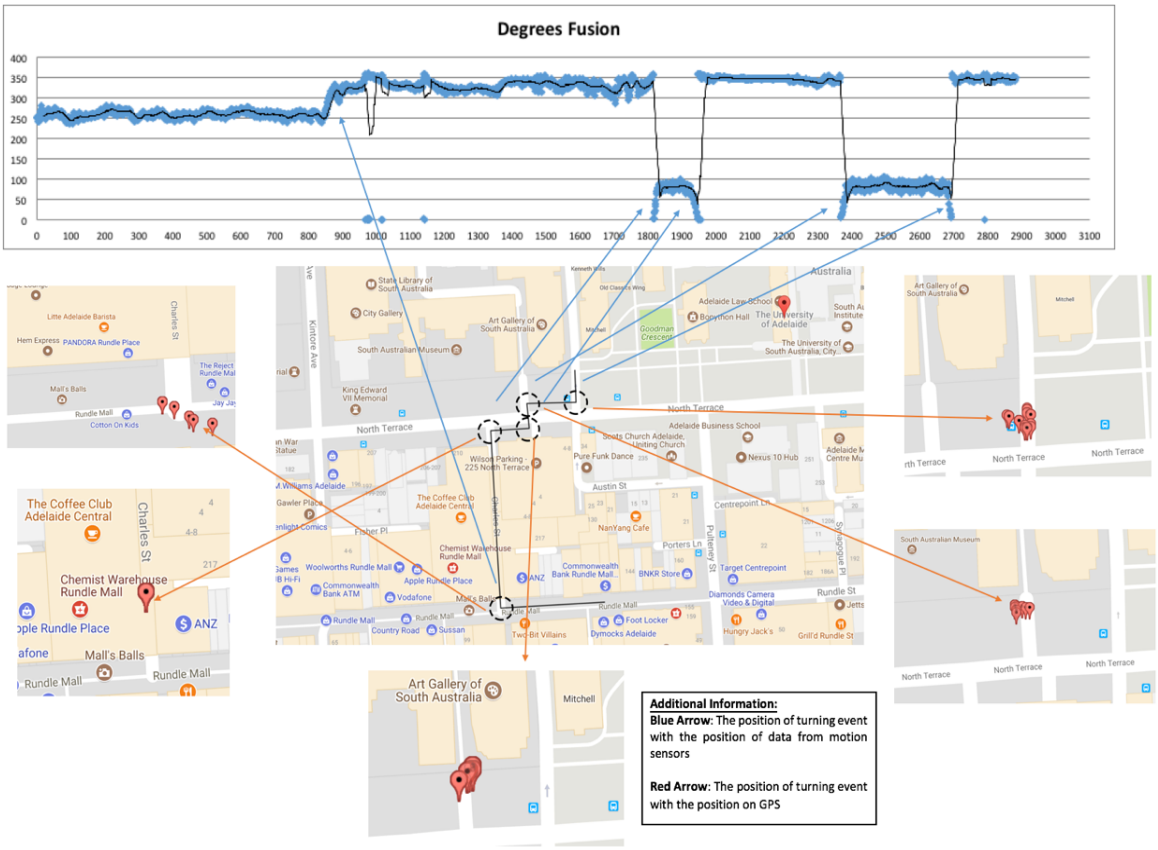

Figure 6.8: Comparison of the results

Figure 6.8 shows that data from motion sensors and shows better accuracy than GPS as the motion sensors manages to show the right turning event based on the real path taken during experiment. However, compared with the GPS that can pin-point the user location, motion sensors cannot tell the user location. With an assumption that the initial location is known, the motion sensors need to calculate the distance between the initial event and each turning event to estimate user's final location. Thus, a comparison is made using the step counter sensor and the distance taken using mapdevelopers.com based on real path. The result of comparison can be seen below,

Total Distance Calculation

Total Distance based on mapdevelopers.com: 349m

Total Steps taken based on log data: 486 steps

Distance taken based on average human stride length: 0.7m x 486 = 340.2m

The difference in value: 349 – 340.2 = 8.8m (Acceptable)

Initial Position to first turning

Total Distance based on mapdevelopers.com: 89.2m

Total Steps taken based on log data: 124 steps

Distance taken based on average human stride length: 0.7m x 124 = 86.2m

The difference in value: 89.2 – 86.2 = 3m (Acceptable)

First turning to second turning

Total Distance based on mapdevelopers.com: 147m

Total Steps taken based on log data: 170 steps

Distance taken based on average human stride length: 0.7m x 170 = 119.7m

The difference in value: 147 – 119.7 = 27.3m (Failed)

Second turning to third turning

Total Distance based on mapdevelopers.com: 23.4m

Total Steps taken based on log data: 24 steps

Distance taken based on average human stride length: 0.7m x 24 = 16.8m

The difference in value: 23.4 – 16.8 = 6.6m (Acceptable)

Third turning to fourth turning

Total Distance based on mapdevelopers.com: 20.7m

Total Steps taken based on log data: 43 steps

Distance taken based on average human stride length: 0.7m x 43 = 30.1m

The difference in value: 30.1 – 20.7 = 9.4m (Acceptable)

Fourth turning to fifth turning

Total Distance based on mapdevelopers.com: 42.1m

Total Steps taken based on log data: 73 steps

Distance taken based on average human stride length: 0.7m x 73 = 51.1m

The difference in value: 51.1 – 42.1 = 9m (Acceptable)

Fifth turning to final location

Total Distance based on mapdevelopers.com: 26.6m

Total Steps taken based on log data: 51 steps

Distance taken based on average human stride length: 0.7m x 51 = 35.7m

The difference in value: 35.7 – 26.6 = 9.1m (Acceptable)

Based on the distance calculation and the difference of distance between motion sensors and Google Maps distance tools provided by mapdevelopers.com, 6 out of 7 regions are within acceptable areas or have better accuracy compared to the range of commercially used GPS without interference. Furthermore, the regions with the acceptable result will be processed and changed into percentage value. The calculation of the acceptable result is,

$$\frac{6}{7}x100\% = 85.7\%$$

Based on the result acceptance table, the result of the experiment can be counted as accurate, as it is within the range of 75 % - 100 %. As the accuracy result of the motion sensors is better than the accuracy of GPS, it can be concluded that motion sensors fusion is capable of tracking the walking movement with obstruction and has better quality over GPS.

6.1.2 Cycling

The experiment was conducted using Android 6.0 at night without traffic issue from Angas St. to Fullarton Rd. Before data collection begun, the GPS must be turned on and the magnetometer was calibrated. The experiment used normal cycling pace with phone located inside the front pocket of jacket or the direction of the movement is located on Z-plane.

The result of orientation will be divided into 2 major parts, the graph for direction based on sensor fusion code and the GPS graphical interpretation. Also, an analysis will be made to measure the distance based on linear acceleration value between the log data from motion sensors and real distance taken during experiment using mapdevelopers.com distance finder function. The graph for direction can be seen on Figure 6.9.

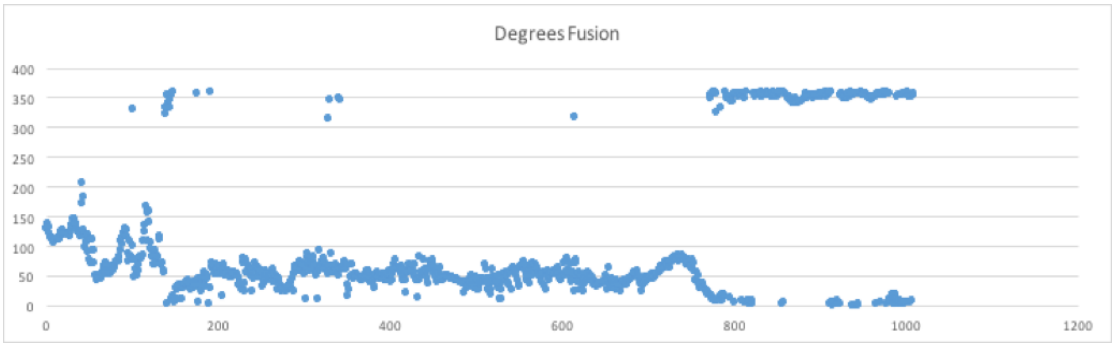

Figure 6.9: Orientation Graph

Figure 6.9 shows some changes during the experiment as the orientation value utilizes 360 degrees based value where 180 degree represents south and 360 or 0 degree represents north. In order to further reduce noises and outliers during the movement, a trend calculation using moving average based calculation will be made to smoothen the data. In addition, a simple clustering based on the real movement during experiment will be made to group the data into each cluster. The result of the moving average calculation and clustering can be seen on Figure 6.10 and 6.11.

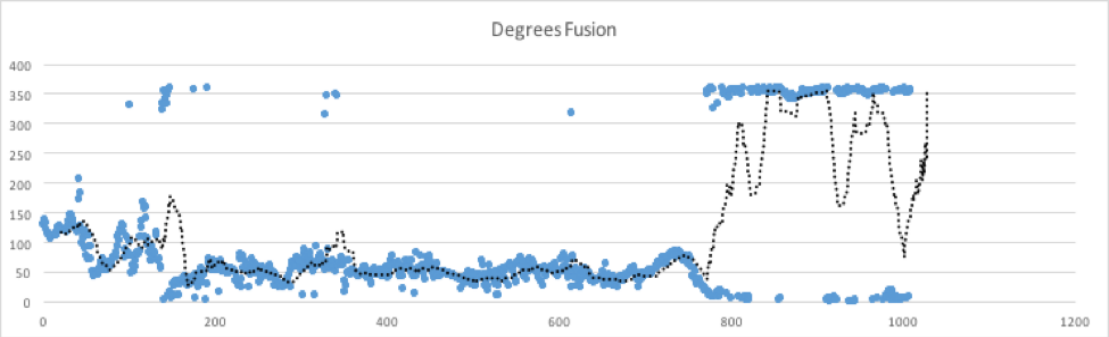

Figure 6.10: Orientation Graph with Trend Line

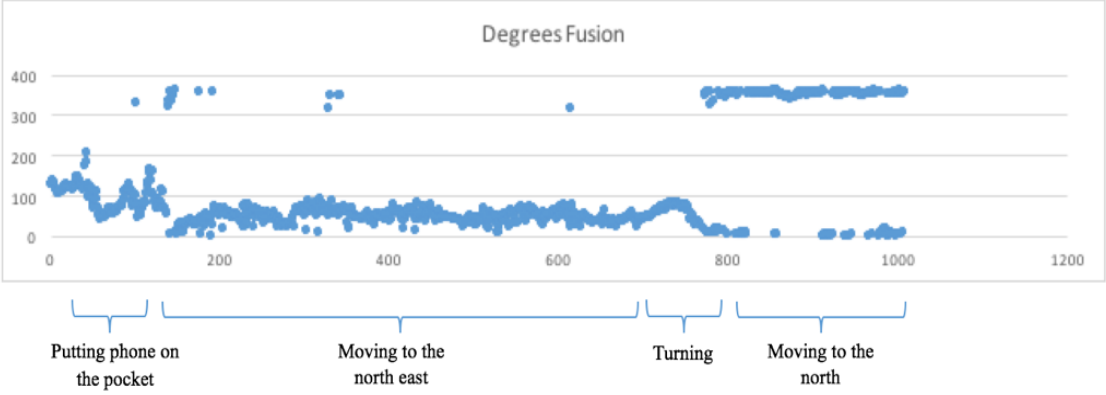

Figure 6.11: Clustered Orientation Graph Based on Movement

There are some issues at the beginning of the orientation graph as the biker is putting the phone inside the pocket and start riding the bike that may cause erratic movement on the graph. It may be caused by the accelerometer value inside the fusion that cannot be handled by magnetometer. However, the orientation becomes stable after the bike moves and the magnetometer manages to handle the movement on accelerometer.

In addition, the experiment also takes longitude and latitude information from GPS for comparison. The graphical interpretation can be seen on Figure 6.12.

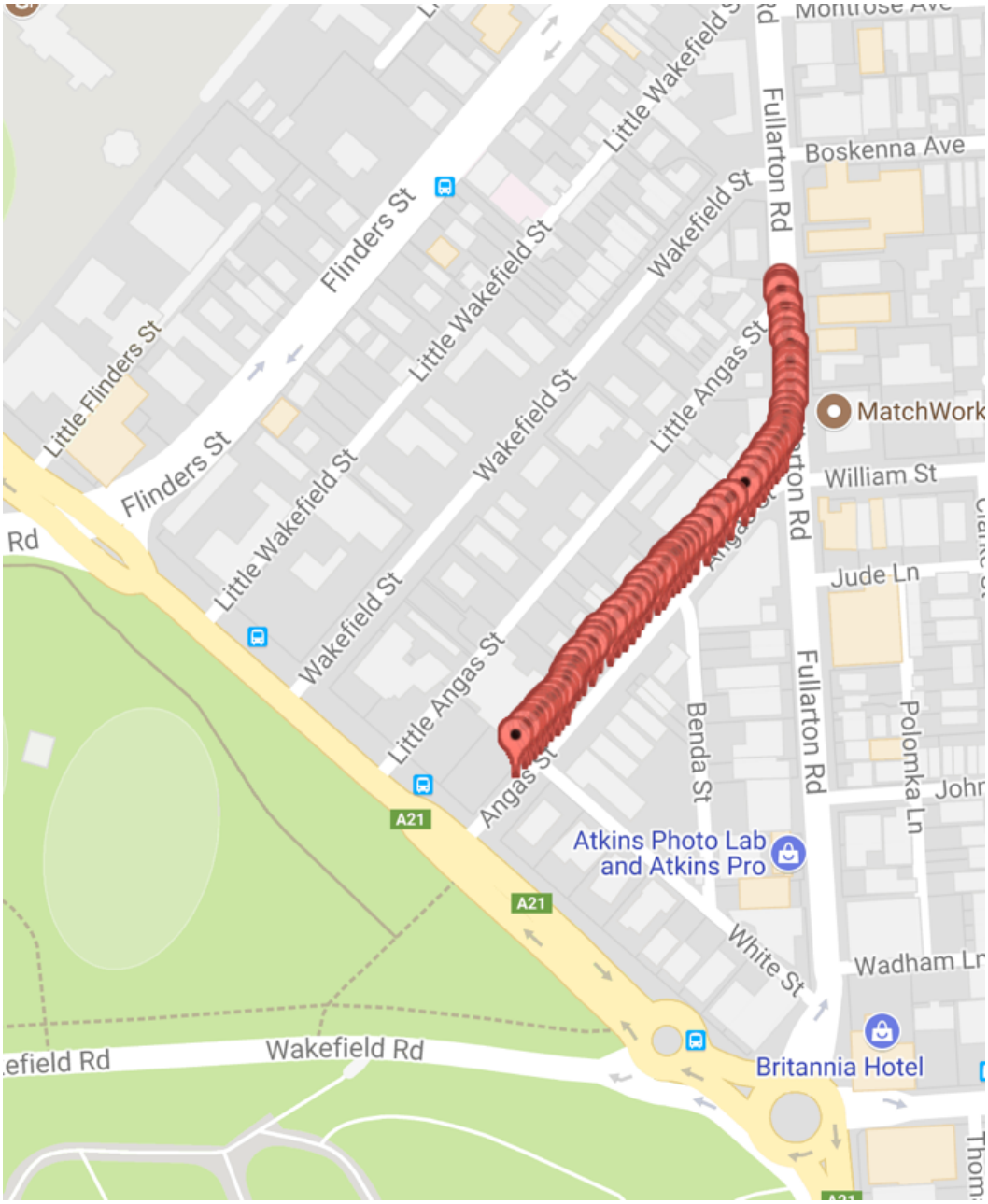

Figure 6.12: Graphical Interpretation of GPS

Based on Figure 6.12, the data shows that GPS can find the location perfectly along the way. This phenomenon justifies the high accuracy of GPS for cycling tracking as it can find the exact location of the phone.

Based on the data from motion sensors and GPS, a comparison should be made to see the accuracy of data especially on the turning event. By adding the real path map taken during experiment into the comparison, the accuracy level of both results can be checked. The comparison can be seen on Figure 6.13.

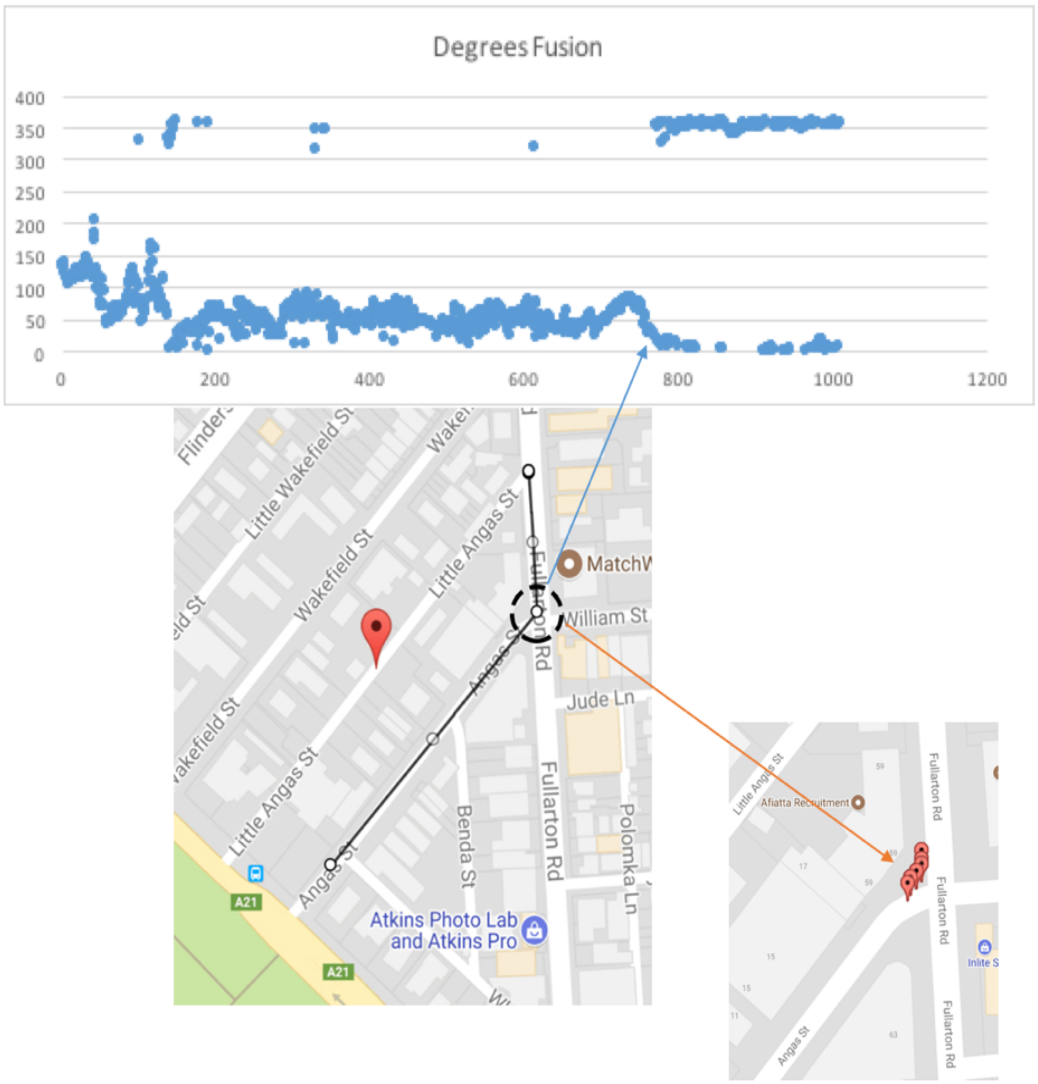

Figure 6.13: The Comparison of Results

Figure 6.13 shows that data from motion sensors and GPS show good accuracy. Both results show the turning event, however, motion sensors add more information regarding the movement of the user and even slight turn to the right before going to the north. However, GPS can pin-point the user location when motion sensors cannot tell the user location. Therefore, linear acceleration sensor will be used to calculate the distance on straight movement until the next turning event or the movement stops. With the assumption that the initial location is known, the motion sensors will use displacement formula to calculate the total distance. Thus a comparison is made using the linear acceleration in m/s^2 and the distance taken using mapdevelopers.com based on real path. the result of comparison can be seen below,

Total Distance:

Linear acceleration towards X plane: 48.09

Linear acceleration towards Y plane: -18.5

Linear acceleration towards Z plane: -166.34

Distance based on mapdevelopers.com: 0.309 Km

Time: 79 seconds

Initial Location to First Turn

Linear acceleration towards X plane: 29.49

Linear acceleration towards Y plane: 0.035

Linear acceleration towards Z plane: -127.97

Distance based on mapdevelopers.com: 0.219 Km

Time: 58 seconds

First Turn to Final Destination

Linear acceleration towards X plane: 18.6

Linear acceleration towards Y plane: -18.46

Linear acceleration towards Z plane: -38.37

Distance based on mapdevelopers.com: 0.09 Km

Time: 21 seconds

Based on the raw data collection, it can be seen that the value of the linear acceleration on Z-plane is lower than 0 which means that the car is decelerating rather than accelerating. Without enough acceleration, it means the bike was not moving at all during the experiment. Therefore, the data from linear acceleration is unusable for distance calculation.

Further, the X plane shows positive acceleration value which can be happened to the user movement when riding the bike as there is a small movement to the left and right which may cause some value get registered as acceleration value. It indicates that the motion sensors prone to the vibration caused by any type of movement and it can cause high level of noise inside the linear acceleration data and render the acceleration value unusable.

Therefore, a modified filter must be created to remove all noises which is really difficult in current stage and need more experiments conducted with controlled parameters to pin-point the cause and the solution for each source of noise. As without the correct filter, it is difficult to calculate the value as the result may be wrong and causes issue to the output of data analysis.

Therefore, it can be concluded that the result of motion sensors cannot be used to track driving movement. The linear acceleration is a new sensor implemented by Android 6.0 and it is difficult to extract good acceleration data as previous researches also suffer the same issue with the quality and accuracy of the data.

6.1.3 Driving

The experiment was conducted using Android 6.0 at night without traffic issue. Before data collection begun, the GPS must be turned on and the magnetometer was calibrated. The experiment used normal driving pace with phone located inside the car.

The result of orientation will be divided into 2 major parts, the graph for direction based on sensor fusion code and the GPS graphical interpretation. Also, an analysis will be made to measure the distance based on linear acceleration value between the log data from motion sensors and real distance taken during experiment using mapdevelopers.com distance finder function. The graph for direction can be seen on Figure 6.14.

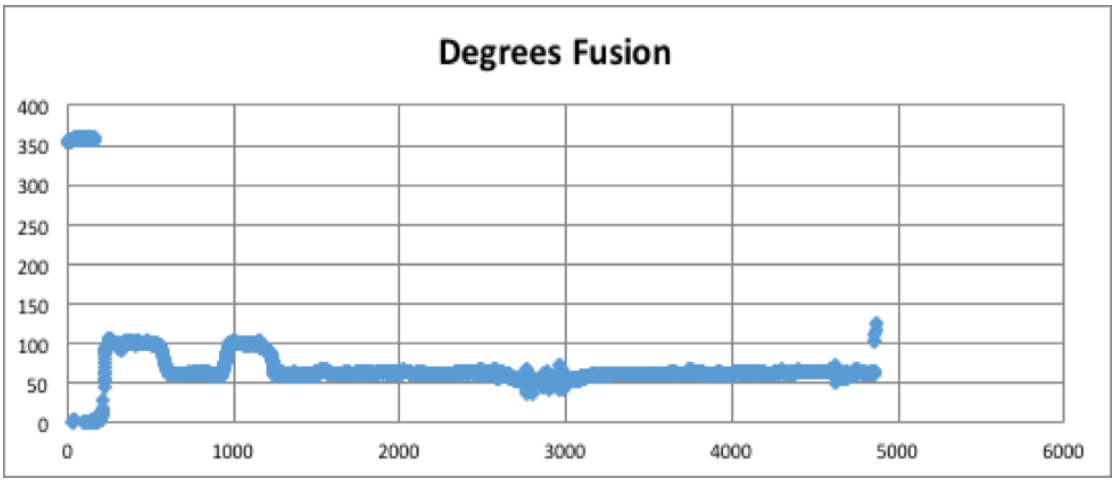

Figure 6.14: Orientation Graph

Figure 6.14 shows some changes during the experiment as the orientation value utilizes 360 degrees based value where 180 degree represents south and 360 or 0 degree represents north. In order to further reduce noises and outliers during the movement, a trend calculation using moving average based calculation will be made to smoothen the data. The result of the moving average calculation can be seen on black line on Figure 6.15.

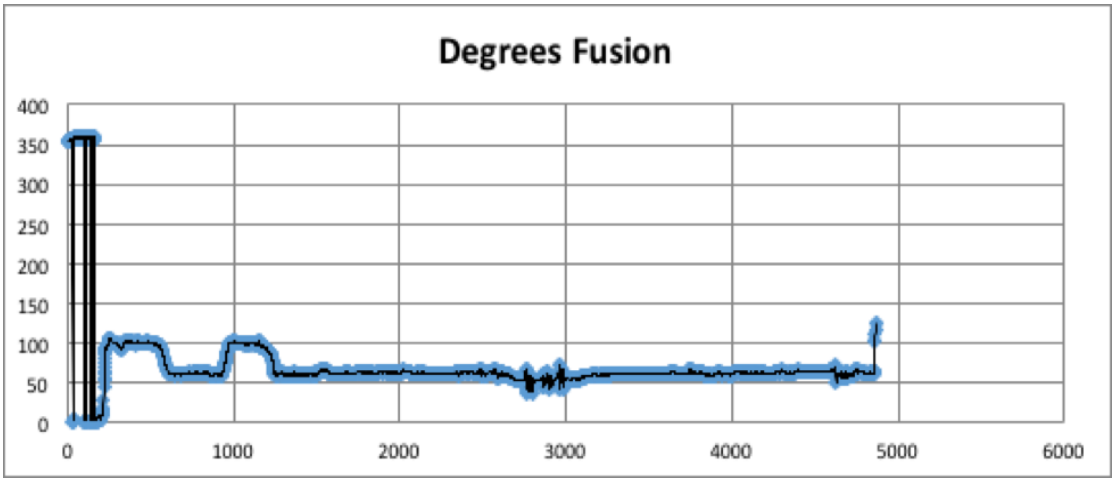

Figure 6.15: Orientation Graph with Trend Line

There are some issues on the orientation graph as the turn is only counted as 45 degrees rather than 90 degrees. It may be caused by the magnetic interference inside the car as many of car's components utilise magnet. In addition, some electromagnetic fields may be created during the movement of motor which may cause noise on the magnetometer and reduce the quality of the interpretation. However, the accelerometer does not get affected by magnet can still measure the turning event but with lower precision. Therefore, the turning event can be captured but the direction of the movement does not precisely follow the magnetic compass value.

In addition, the experiment also takes longitude and latitude information from GPS for comparison. The graphical interpretation can be seen on Figure 6.16.

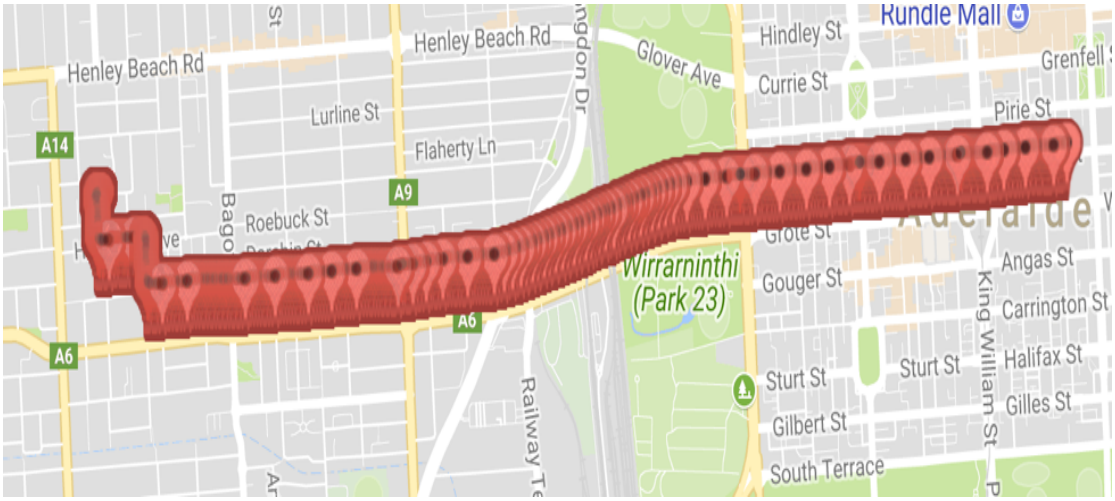

Figure 6.16: Graphical Interpretation of GPS

Based on Figure 6.16, the data shows that GPS can find the location perfectly along the way. This phenomenon justifies the high accuracy of GPS for driving tracking as it can find the exact location of the phone. Based on the data from motion sensors and GPS, a comparison should be made to see the accuracy of data especially on the turning event. By adding the real path map taken during experiment into the comparison, the accuracy level of both results can be checked. The comparison can be seen on Figure 6.17,

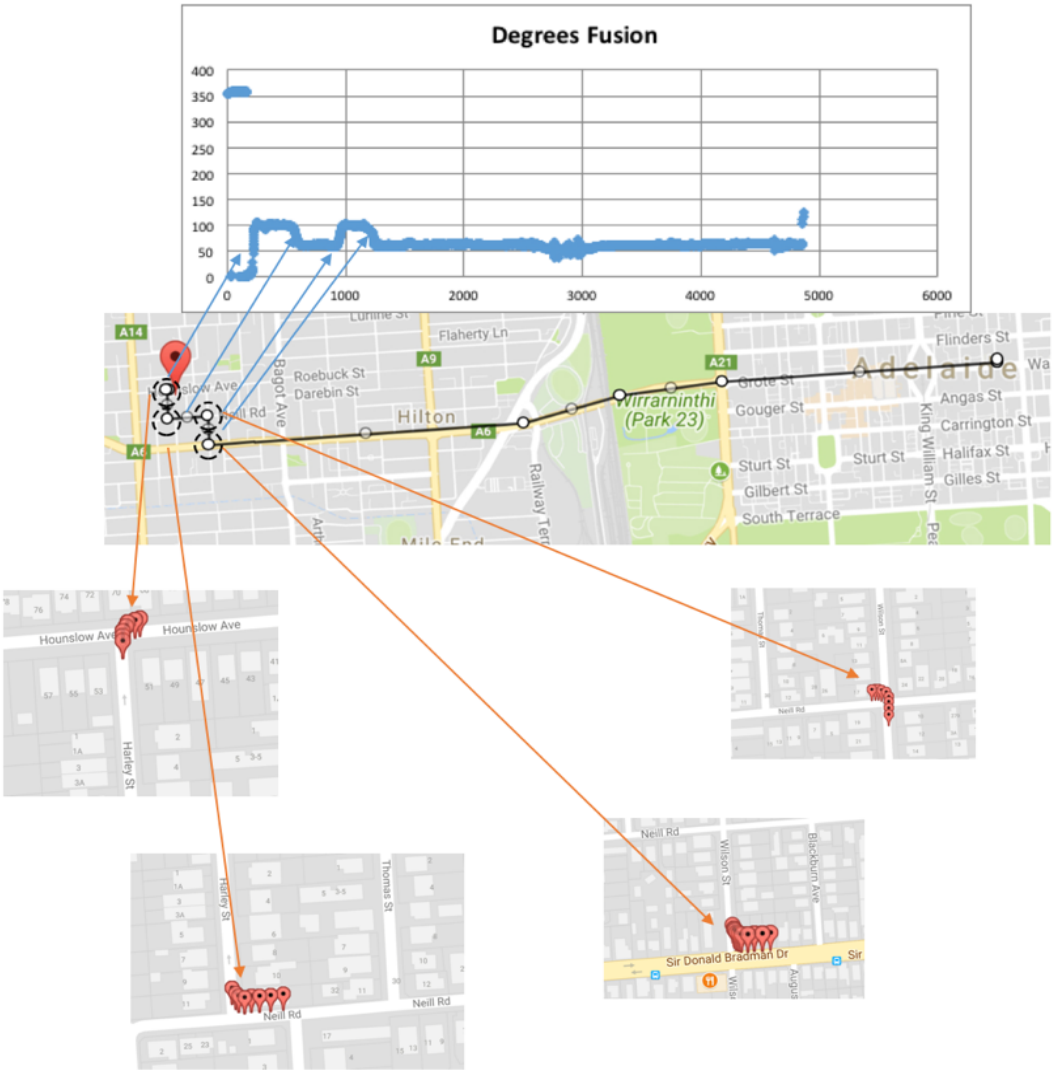

Figure 6.17: The Comparison of Results

Figure 6.17 shows that data from GPS shows better accuracy compared to the motion sensors. Even though, both results show the turning event, however, GPS can show the right turning degrees which is 90 degrees. The 90 degrees is based on the real path taken during experiment. In addition, GPS can pin-point the user location when motion sensors cannot tell the user location. Therefore, linear acceleration

sensor will be used to calculate the distance on straight movement until the next turning event or the movement stops.

With the assumption that the initial location is known, the motion sensors will use displacement formula to calculate the total distance. Thus a comparison is made using the linear acceleration in m/s^2 and the distance taken using mapdevelopers.com based on real path. the result of comparison can be seen below,

Total Distance:

Linear acceleration towards X plane: -28.59

Linear acceleration towards Y plane: -168.16

Linear acceleration towards Z plane: -1020.77

Distance based on mapdevelopers.com: 5.048 Km

Time: 11 minutes 27 seconds (687 seconds)

Initial Location to First Turn

Linear acceleration towards X plane: -0.902

Linear acceleration towards Y plane: -11.69

Linear acceleration towards Z plane: -32.05

Distance based on mapdevelopers.com: 0.014 Km

Time: 15 seconds

First Turn to Second Turn

Linear acceleration towards X plane: -16.9

Linear acceleration towards Y plane: -16.62

Linear acceleration towards Z plane: -81.95

Distance based on mapdevelopers.com: 0.163 Km

Time: 25 seconds

Second Turn to Third Turn

Linear acceleration towards X plane: 0.155

Linear acceleration towards Y plane: -7.36

Linear acceleration towards Z plane: -84.13

Distance based on mapdevelopers.com: 0.219 Km

Time: 28 seconds

Third Turn to Fourth Turn

Linear acceleration towards X plane: -5.25

Linear acceleration towards Y plane: -0.39

Linear acceleration towards Z plane: -54.66

Distance based on mapdevelopers.com: 0.157 Km

Time: 24 seconds

Fourth Turn to Final Location

Linear acceleration towards X plane: -6.29

Linear acceleration towards Y plane: -132.71

Linear acceleration towards Z plane: -769.34

Distance based on mapdevelopers.com: 4.495 Km

Time: 595 seconds

Based on the raw data collection, it can be seen that the value of the linear acceleration is mostly lower than 0 which means that the car is decelerating rather than accelerating. Without enough acceleration, it means the car was not moving at all during the experiment. Therefore, the data from linear acceleration is unusable for distance calculation.

In addition, it also means that there were many sources of noise during the experiment and were captured by the sensors. As the level of noise is really high, especially on Z-plane, a modified filter must be created to remove all noises which is really difficult in current stage and need more experiments conducted with controlled parameters to pin-point the cause and the solution for each source of noise. This experiment also indicates that the linear acceleration sensor is prone to vibration caused by noises and can make the result of the sensor unusable for analysis.

Therefore, it can be concluded that the result of motion sensors cannot be used to track driving movement. The linear acceleration is a new sensor implemented by Android 6.0 and it is difficult to extract good acceleration data as previous researches also suffer the same issue with the quality and accuracy of the data.

6.2 Controlled Experiment

The controlled experiment will be conducted on 3 main types of terrain with 1 additional terrain for final comparison. The 3 main types of terrain are asphalt, ceramic tile and carpet. Next, a full comparison between 3 main types of terrain and 1 smooth terrain (triplex) will be made in order to analyze how the external noises affect the quality of the linear acceleration measurement by the Android phone. Figure 6.18 will show the picture of terrains used for experiment,

Figure 6.18: List of Terrains

A: Asphalt B: Carpet C: Ceramic Tile D: Triplex (Wooden Board)

The experiment was conducted with the robot moving towards X direction. The phone was put on the top of the robot and the robot was moving for 10 seconds. The increase of speed at the beginning of the movement was high as the robot was set to move in full power. However, the robot was gradually reducing speed towards the end of the movement (9 – 10 s).

In order to provide a better comparison, 2 types of linear acceleration code will be used, the first one is the linear acceleration sensor using the fusion of physical based sensors (Accelerometer and magnetometer) provided by Google. The second one is the sensor fusion code created using the combination of accelerometer, magnetometer and gyroscope. The gyroscope is the new sensor added to the new version of phone and Android 6.0 introduces several new features to add gyroscope into the sensor fusion.

The value from 2 different codes will be compared with one another to see how linear acceleration behaves in a particular type of terrain. After the comparison of 2 linear acceleration's codes, the result of the code will be compared with other terrain to identify the noise produced during the experiment on each terrain. This experiment will be used to determine the current quality of the linear acceleration and the improvement that can be made for future works.

6.2.1 Asphalt

Asphalt is the type of terrain that introduce some external sources of noise such as bump and uneven terrain. However, asphalt is important as many of the movement is done on asphalt. Therefore, an experiment is needed to measure the level of the noise and whether the linear acceleration data is usable after the experiment with the addition of noise. There are 2 graphs representing the linear acceleration code to measure the acceleration of the robot during the experiment. Figure 6.19 represents the linear acceleration value graph from the linear acceleration sensor provided by Google.

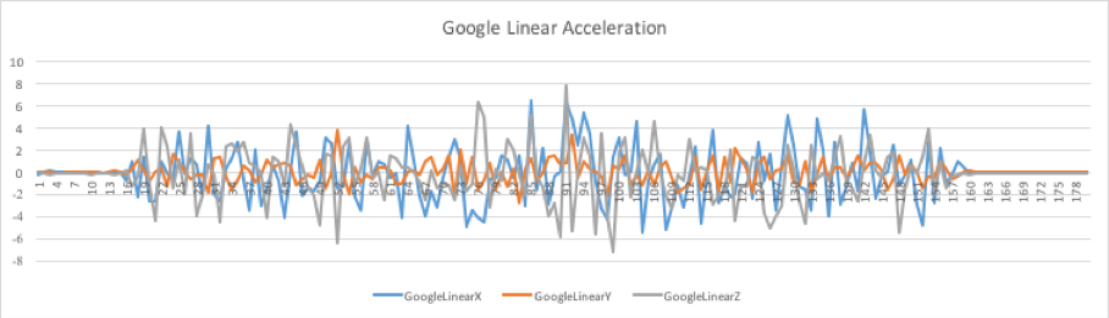

Figure 6.19: Google Linear Acceleration Code

The graph on Figure 6.19 shows some erratic movement of the value for X, Y and Z which happens due to the addition of the noise as the robot only moves towards X during the experiment. As it is difficult to pin-point which part is noise as there are several sources of noise, the linear acceleration can be considered as unusable until a specific type of filter is created to removes most of the noises. For comparison, a new fusion code is added using the combination of accelerometer, magnetometer and gyroscope to see whether the quality of linear acceleration measurement improves or not. The figure can be seen below.

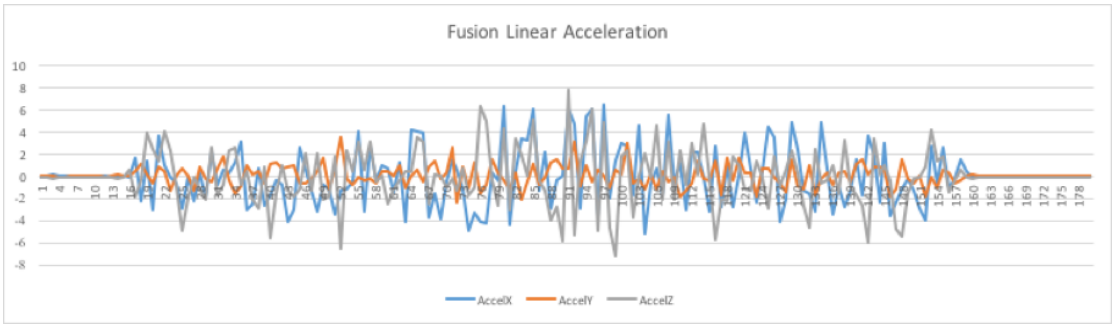

Figure 6.20: Fusion Linear Acceleration Code

As the figure shows, the noise remains inside the linear acceleration graph even after the addition of the gyroscope. It may indicate that the source of noise add value higher than the amount of reduction that gyroscope can compensate. It is difficult to utilize linear acceleration as source of distance measurement at the moment even with the addition of new functionalities by Android 6.0. Therefore, the value of the linear acceleration can be assumed as unusable for movement on asphalt based terrain.

6.2.2 Ceramic Tile

Ceramic tile is used to identify bump during the change from one tile to another as there is a slight gap between each tile. The tile used during the experiment is 30x30cm tile. The figure of the tile can be seen below

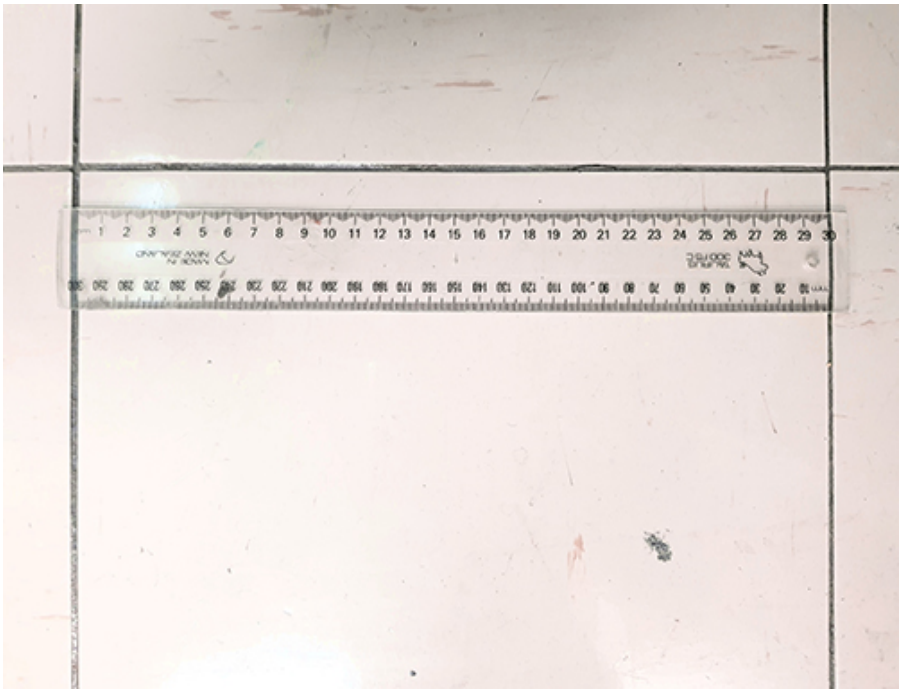

Figure 6.21: Measurement of Tile with Ruler

During the experiment, the robot will move for 10 seconds and the robot manages to stop on the end of twelve-th tiles, which means that it takes,

$$\frac{10s}{12tiles} = 0.83s/tile$$

Thus, an experiment was made to pin point the noise made by small bump during movement. There are 2 graphs representing the linear acceleration code to measure the acceleration of the robot during the experiment. Figure 6.22 represents the linear acceleration value graph from the linear acceleration sensor provided by Google.

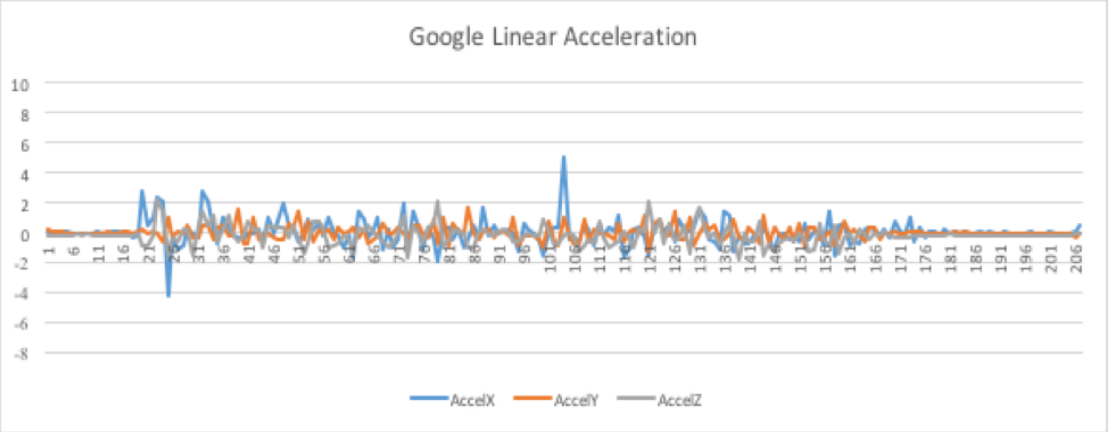

Figure 6.22: Google Linear Acceleration Code

The Figure 6.22 shows some erratic movement of the value for X, Y and Z axes which happens due to the addition of the noise as the robot only moves toward X during the experiment. As the log file takes data around 60-70ms in average, therefore the data should show the bump for every 13 points on the graph based on the graph X-plane. However, there is no significant change on the data caused by bump the graph. It can indicate whether the bump is not significant to cause some changes on the data or there is another source of noise that produce higher level of noise compared to the bump. In order to compare the quality of the code, Figure 6.23 represents the fusion code with the addition of the gyroscope,

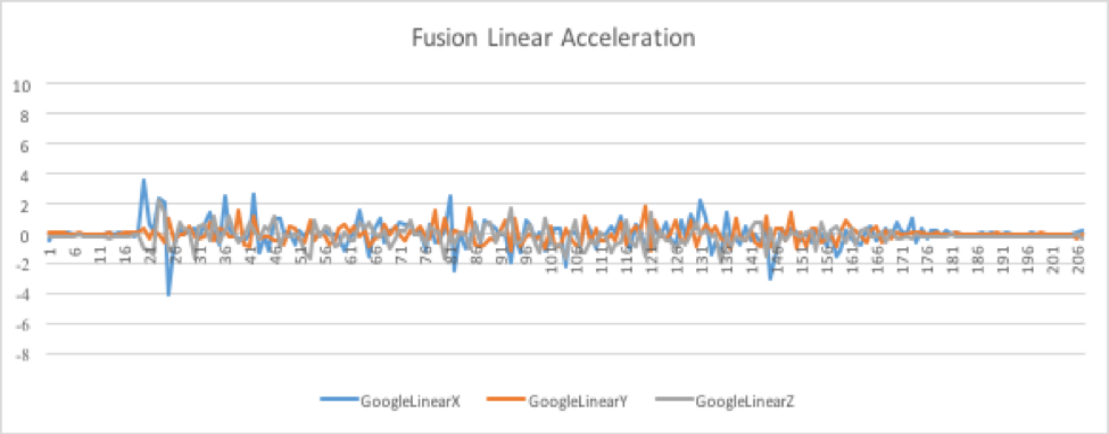

Figure 6.23: Fusion Linear Acceleration Code

As the figure shows, the data is still noisy with some high spikes on the data. It is still difficult to analyze the linear acceleration value as there are some weird movement on Y and Z axes during the movement. It may indicate that there are additional sources of noise which add value to the graph other than bump and it affects all axes. The source of noise may come from external source such as crooked tile or internal source such as the vibration of the motor. Therefore, it is difficult to utilize linear acceleration as source of distance measurement at the moment on the road condition similar to ceramic tile even with the addition of new functionalities by Android 6.0.

6.2.3 Carpet

Carpet is used to identify the movement on the soft terrain to see the impact of the softness to the acceleration value. In addition, soft terrain can also cause uneven terrain as the softness can be different on left and right side. Therefore, an experiment was made to measure whether the soft terrain can cause deceleration during the movement and how the uneven terrain adds noise into the linear acceleration value. The Figure 6.24 will provide the value graph measured by Google linear acceleration sensor. The figure can be seen below,

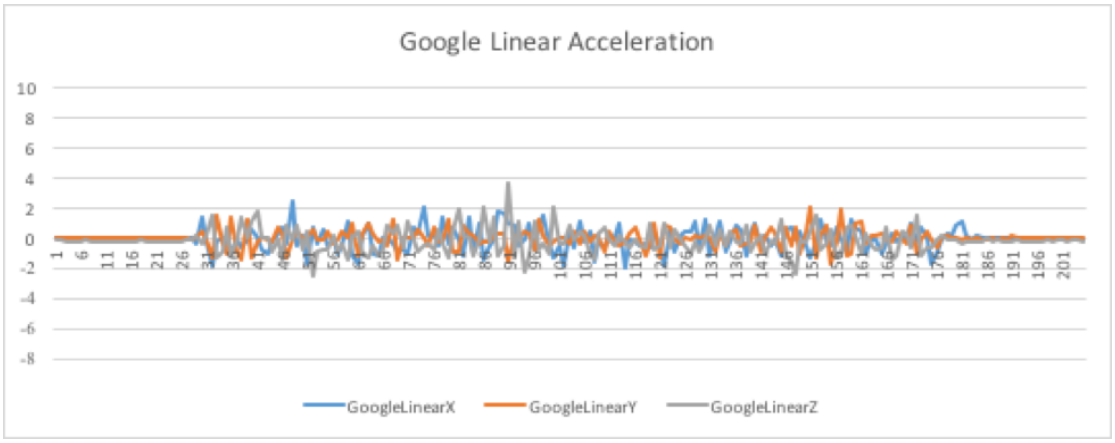

Figure 6.24: Google Linear Acceleration Code

Based on Figure 6.24, There are many unidentified value on X, Y and Z axes especially because the robot moves towards X axes for the experiment. It is difficult to find the noise from uneven terrain or other sources of noise as there is no significant characteristic for each noise. The linear acceleration can be considered as unusable as at the moment, there is no enough information to pin-point the location of the acceleration value and how to remove the noise from the data. For comparison, a graph is made from the linear acceleration value created by the fusion code. The graph is also used to measure the quality of linear acceleration after the addition of gyroscope. The graph of the fusion code can be seen below,

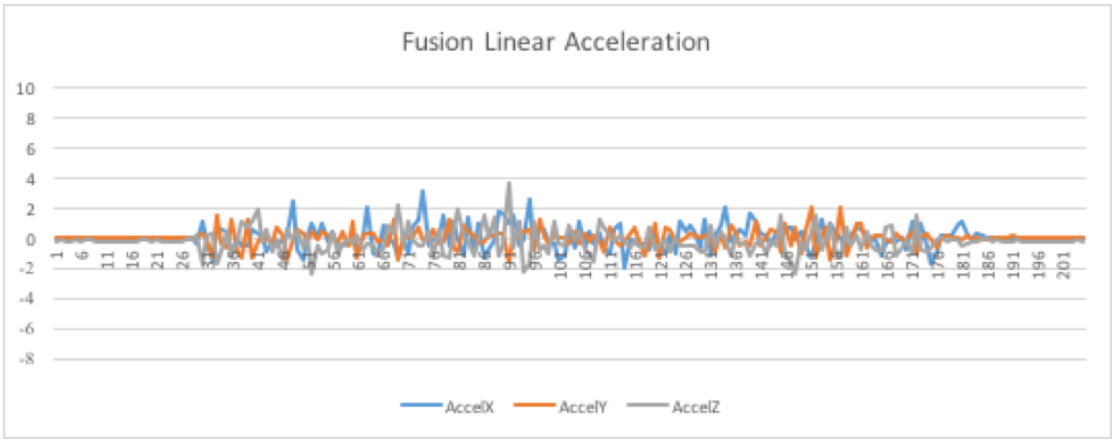

Figure 6.25: Fusion Linear Acceleration Code

Based on the graph, the linear acceleration value shows some small spikes inside the data along the way. It shows that the texture of the carpet may reduce the friction between the tires and the road which lead to the reduction of speed and the effect of the bump as the robot does not move as fast as when it is placed in hard terrain. however, it is still difficult to utilise the data as it is hard to identify the location

and the cause of noise.

In addition, there may be noise caused by the uneven terrain due to the different of thickness in the carpet and it is difficult to identify the exact position without additional measurement. Also, The linear acceleration sensor utilises accelerometer which based on previous researches, is sensitive to vibration and can lead to noisy data. Therefore, the current state of linear acceleration sensor may be not suitable for road condition similar to the carpet.

6.2.4 Comparison of Results

Linear acceleration is the new sensor added to the Android 6.0 and the sensor itself has not been fully utilised by many applications. Therefore, there is no specific research to measure the performance of the linear acceleration value.

In order to analyse the performance of linear acceleration in the real life condition, a controlled experiment using robot is made in different terrains. There are 4 terrains used to test the quality of the data, which are, asphalt, carpet, ceramic tile and on the top of wooden board.

The asphalt represents the real situation as driving is usually performed on asphalt road. The carpet represents the soft terrain with uneven road condition and random bump due to the difference in thickness of the carpet. The ceramic tile represents the terrain with bump to check how the bump affects the quality of the linear acceleration data. The last one is on the top of wooden board or smooth surface. The smooth surface is used to analyse the internal noise caused by the motor by reducing the amount of noise from external sources.

In order to provide a good comparison, data from linear acceleration sensor are taken using Lego Mindstorms EV3. The photo of the robot can be seen below.

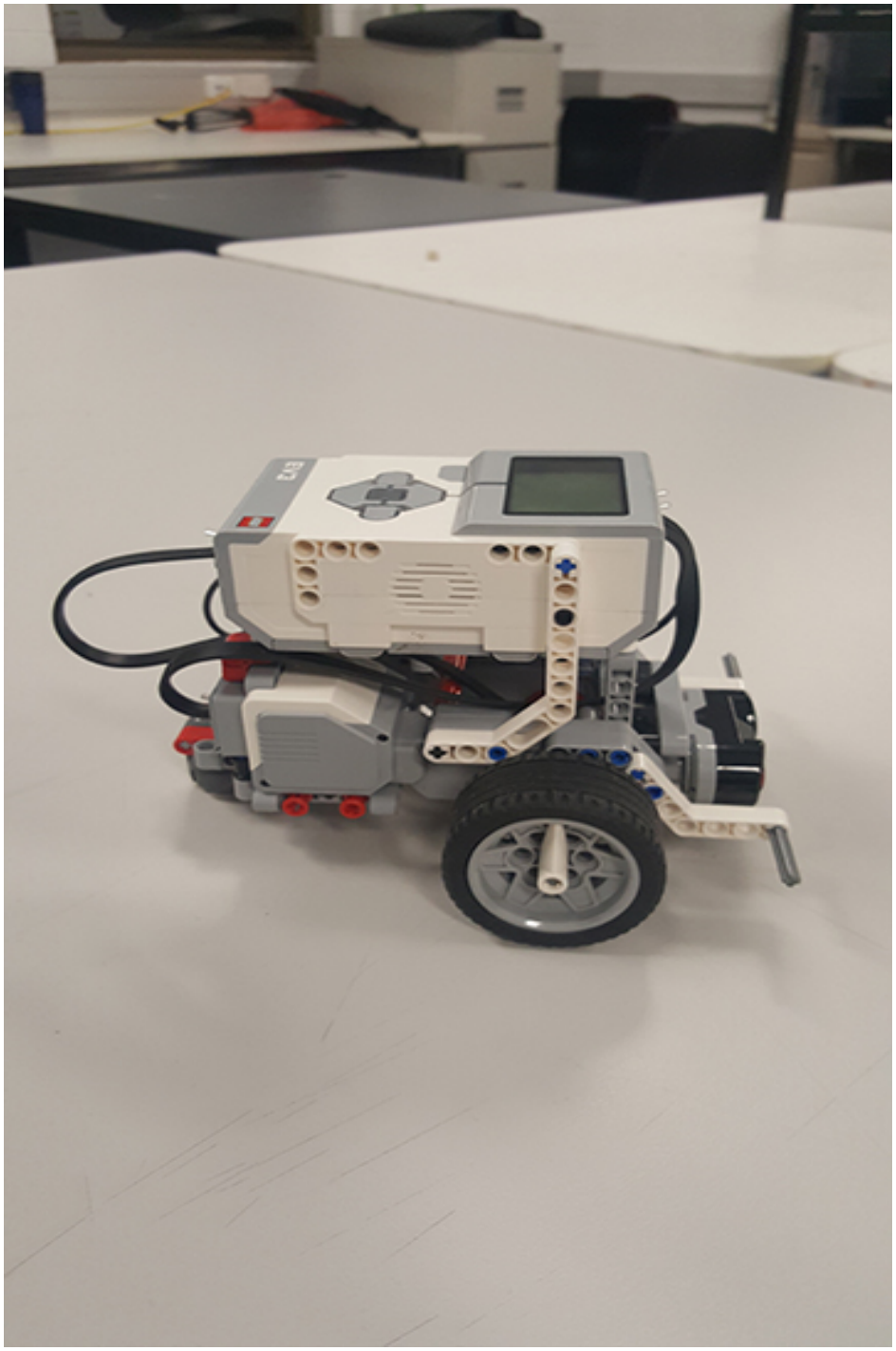

Figure 6.26: Lego Mindstorms EV3

The robot is using full speed and the robot moves for 10 seconds and reduces its speed on normal pace (starts from 8s and stops completely at 10s). The Android phone is placed on the top of the robot and the robot moves toward X-plane direction. The linear acceleration graph of all terrains can be seen below,

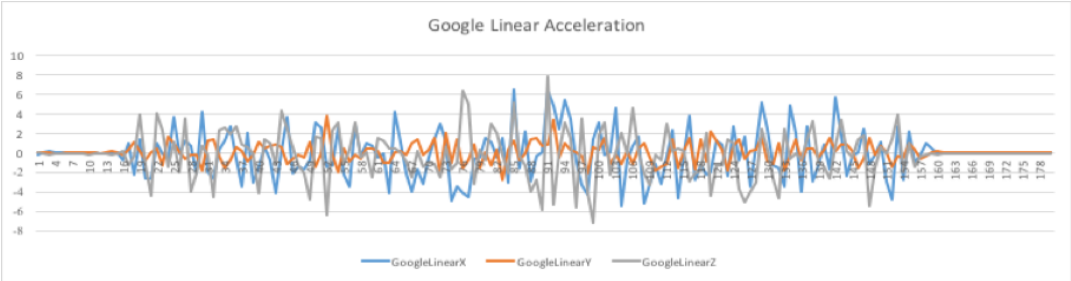

Figure 6.27: Linear Acceleration Value on Asphalt

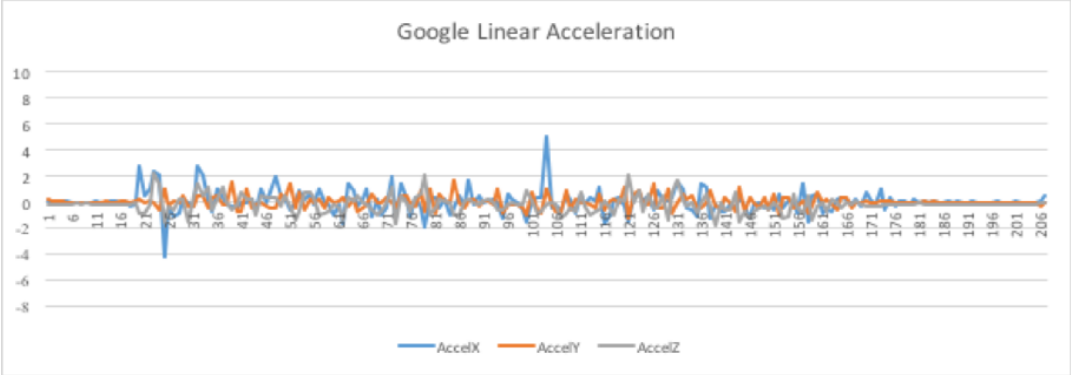

Figure 6.28: Linear Acceleration Value on Ceramic Tile

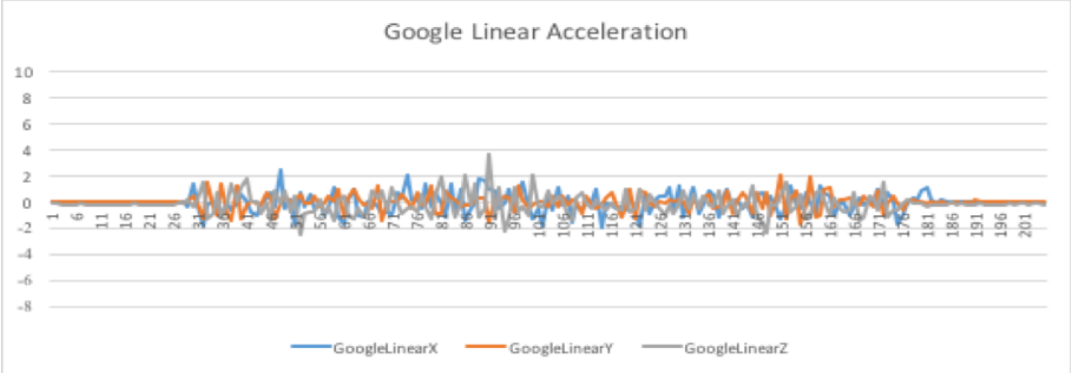

Figure 6.29: Linear Acceleration Value on Carpet

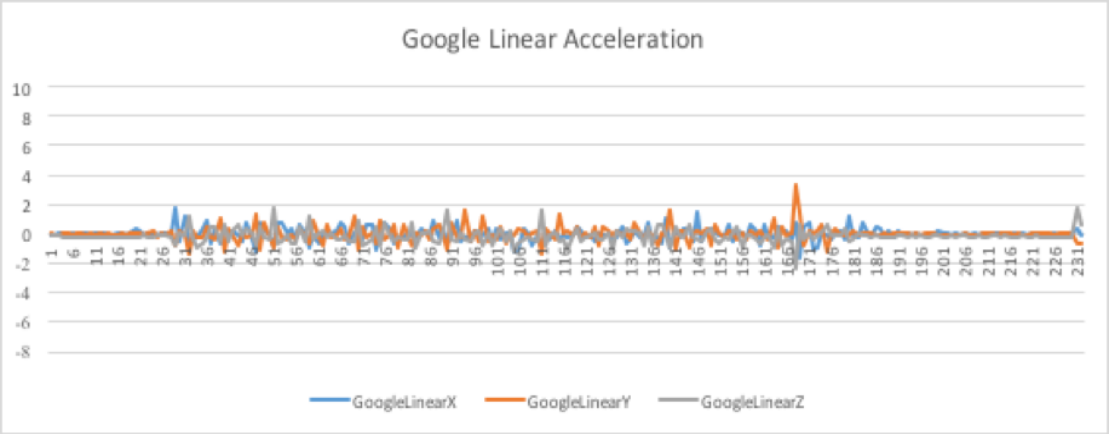

Figure 6.30: Linear Acceleration Value on Smooth Surface

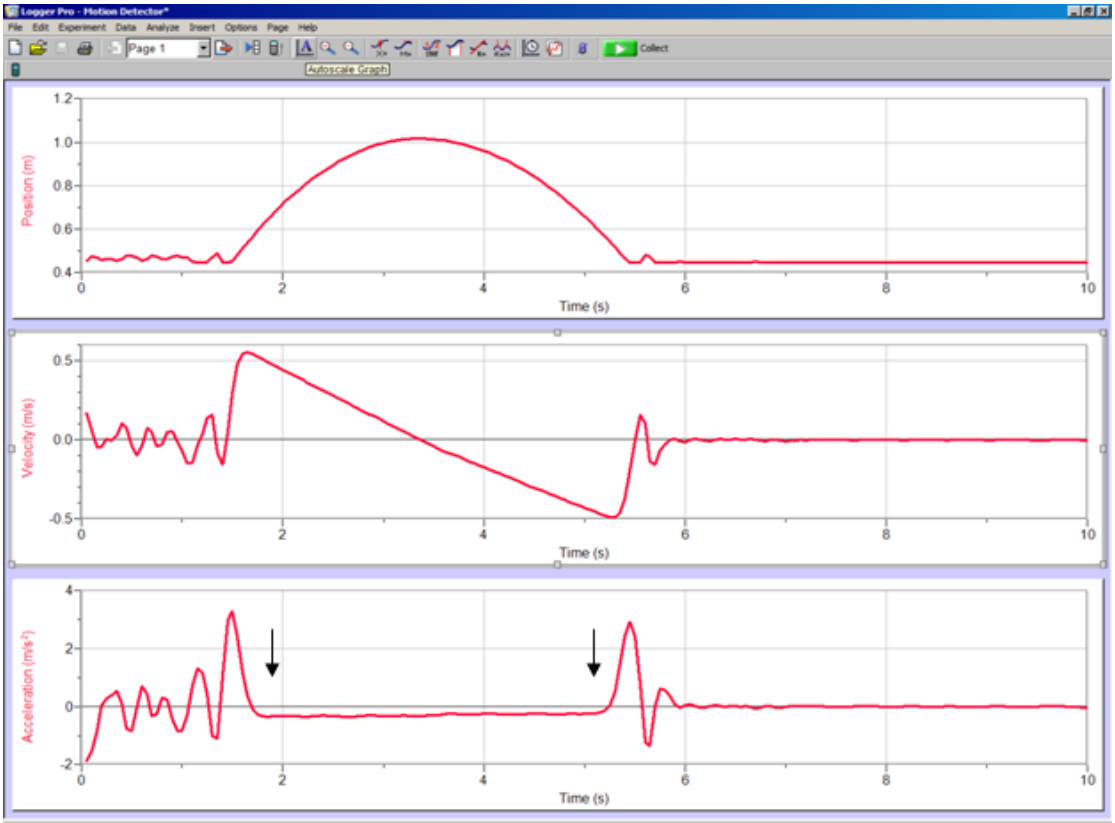

Figure 6.31: Ideal Linear Acceleration Value Graph [6]

In order to provide a valid comparison, Figure 6.31 is used as a base for comparison for all terrains. By using Figure 6.31, the noise or issue from each experiment on different terrain can be identified. Furthermore, an initial conclusion regarding the issue can be achieved after the comparison. Based on the graph on Figure 6.27, asphalt introduces data with the highest value with one highest value in the

middle of experiment which is around $8m/s^2$. As the value is really close to the value of gravity which is $9.8m/s^2$, it can be assumed that the value comes from the addition of noise, as $8m/s^2$ means that the robot moves around 8 meters in 1 second which is unlikely to happen as the experiment on the ceramic tile with 30x30 cm size shows that the robot needs around 1 second to pass 1 tile.

The second graph which is Figure 6.28. is ceramic tile. It can be seen that there are some spikes which may be caused by the gap between the tile that acts like a bump. However, compared to asphalt, the overall value of the acceleration data is around $1 - 2m/s^2$. The data is still noisy but the value is more realistic compared to the asphalt.

The third graph which is Figure 6.29. is carpet. As can be seen on the graph, the data on Y and Z plane shows higher value compared to the ceramic tile. It support the evidence about uneven terrain which may some acceleration value to the left and right side as the robot swings toward one direction when there is an uneven terrain. In addition, carpet is part of soft terrain which means the uneven terrain does not produce noise with the value as high asphalt.

Figure 6.30. shows value from smooth terrain which means that there is almost no external noises . There may be other type of external noises that cannot be identified during the experiment that can reduce the quality of the data, however, the major cause of noise from the environment has been reduced or removed from the experiment on smooth terrain.

As Figure 6.30. shows, the smooth terrain produces the smallest overall linear acceleration value compared to the other terrain. However, there is still some spikes on Y and Z plane which may come from other source of noise. As the experiment is conducted to identify the external noise, there may be some types of noise produced internally. The internal noises may come from the vibration of motor or the wheel friction when moving or braking. Therefore, smooth terrain introduces another type of noise that affect the quality of the linear acceleration even after the removal of several major causes of noise on linear acceleration data.

Next, the figure on each terrain is calculated to find the average and standard deviation of linear acceleration value on each axis to see the roughness of the terrain and how the roughness affects the value of acceleration on the sensor. the result can be seen on the table below

Terrain	Type of Calculation	X Value	Y Value	Z Value
Asphalt	Average Value	2.125654489	0.865800435	2.051847582
	Standard Deviation	1.521714086	0.668423941	1.697928891
Ceramic tile	Average Value	0.662479332	0.484736695	0.570715982
	Standard Deviation	0.660851357	0.374681146	0.457925266
Carpet	Average Value	0.680541252	0.507100691	0.733556248
	Standard Deviation	0.507500029	0.451980781	0.570775981
Wooden Board	Average Value	0.421156111	0.428002651	0.427369091
	Standard Deviation	0.344645598	0.442389089	0.363304543

The table shows that asphalt has the highest roughness and it increases the value of the linear acceleration. On the other hand, the wooden board shows the lowest standard deviation which means that it has the smoothest surface compares to other surfaces and it also gets reflected by the low average value of the linear acceleration value. Therefore, it can be concluded that the roughness of the terrain increases the amount of noise therefore increase the average linear acceleration value measured by the sensors.

Based on the experiment of the linear acceleration using robot on several type of terrain. It can be seen that linear acceleration value is noisy and it is difficult to identify the real acceleration value from the data as the amount of noise is high. Therefore, it can be concluded that linear acceleration sensor on Android 6.0 is still unusable for tracking function and a custom filter must be made in order to extract the real linear acceleration value from the data.

6.3 Discussion of Result

Android 6.0 introduces several new features to enhance the quality of motion sensors to track user's movement. In order to analyse the performance of the features and how it can fix the issues found by previous researches in the similar areas, new experiment was conducted by using the motion sensors and the fusion of sensors. Based on the experiment, the step orientation sensor managed to perform well under different situation. Even though, there are some situations where the quality of sensor decreases due to an exposure to noise sources such as magnetometer with magnetic field, however the sensor still managed to identify the turning event.

As orientation sensor is need to find the direction of the movement, a way to track the distance is also needed. For walking experiment, Android has developed step counter and detector sensor to identify and measure the step. The experiment conducted using step counter showed a good result with the difference between the distance calculated from step counter and the distance measured using Google Maps tool only showed less than 11 metres. 11 metres is the level of accuracy that can be achieved by GPS, therefore step counter provides better accuracy than GPS. However, as step counter can only be used for walking, another way is needed to measure the distance for other type of movement, such as, cycling and driving.

Linear Acceleration is part of new sensors added on Android 6.0 to measure the acceleration of the user in line with the direction of the movement, which means that if the movement is going towards Y-axis, the linear acceleration value will be added into Y-axis too. As the linear acceleration utilises accelerometer as part of the sensor, there may be some noises inside the data due to high sensitivity of accelerometer on gravity and vibration.

Based on the experiment created to utilise the linear acceleration to calculate distance, the value provided by the sensor is highly noisy and unusable. As there is no enough information regarding how to remove the noise part from the data, it is difficult to fully utilise linear acceleration sensor at the

moment.

In order to identify the source of the noise, several experiments were conducted under specific environment where the terrain condition and experiment time can be measured. Based on the experiment conducted under different terrains, linear acceleration value is still noisy, even after the major noise sources, such as, bumps and even terrain have been removed. it may indicate that there are several other sources that add noise into the sensor, for example, the vibration of motor and wheel friction.

In order to utilise the linear acceleration sensor for daily usage, a customized filter needs to be made in order to remove the majority of the noise. In addition, several enhancement needs to be made on the fusion code in order to reduce the sensitivity of the sensor to vibration which can lead to high amount of noise inside the data. Therefore, with the low accuracy of linear acceleration on Android 6.0, it is difficult to track user on daily basis. However, It is possible to increase the accuracy of motion sensors measurement using additional and customized filters. The addition of the correct filter for linear acceleration may cause security issue on localization.

Chapter 7

Conclusion

In order to provide a comprehensive conclusion based on the experiment, the information will be split into 2 sections, which are, discussion and future work. Discussion will explain condition of current experiment and the comparison with previous research. Future work will provide recommendation and suggestion that can be used to improve the current experiment.

7.1 Conclusion

Motion sensor is sensor used to measure movement. Motion sensor utilise vibration to capture the motion value. Android 6.0 introduces several functions to implement motion sensors into the application. To increase the accuracy, Android 6.0 also develops several software based sensor by combining value from motion sensors, such as, accelerometer and magnetometer.

However, the utilisation of the motion sensors can be done without user's permission and it can lead to the security breach. Several researches have been done on previous version of Android to measure the performance of the sensors on user tracking and all of them suffers from low accuracy and quality of data. Therefore, a new experiment is made to measure the increase in quality on the improvement made on Android 6.0.

In order to track user, 2 main parts of data are needed. Orientation sensor is used to track the direction of movement and acceleration sensor is used to track the distance of movement. Walking movement will utilise step counter sensor and other movements will utilise linear acceleration sensor.

The result of the experiment shows that orientation sensor on the Android 6.0 manages to perform well on most condition where the interference is low. However, the quality is reduced when there is interference in the area, such as, magnetic field.

In addition, step counter sensor used to measure walking experiment shows better accuracy than commercially used GPS with most of the distance calculation have only less than 11 metres difference with the real distance.

However, linear acceleration sensor cannot provide a good acceleration measurement. The level of noise inside the data is really high and render it unusable for data calculation. As linear acceleration sensor utilises accelerometer as part of the sensor, it can be seen that the linear acceleration sensor also suffers from the same issue with the current researches that utilise accelerometer for user tracking.

In order to analyse linear acceleration sensor as new sensor available on Android 6.0, a controlled experiment is made to see how the sensor behaves in different terrain and how different type of noise affects the data extraction. It can be concluded that there are many noise sources externally and internally that affect the quality of the data apart from major noise sources, such as, bump and uneven terrain.

The current experiment discovers the current limitation of motion sensor on user tracking. The result is still unreliable and unusable in the real situation. Based on the result, motion sensors does not raise security issue on localization on Android 6.0. However, with several improvements made in the future to fix the low accuracy on linear acceleration sensor, motion sensors may be able to accurately track user and develop security issue. Thus, A security improvement from the next iteration of Android is needed to prevent future issues on localization due to unauthorized usage of motion sensors.

7.2 Future Work

The experiment conducted on Android 6.0 shows positive and negative results on different areas. However, many improvements can be made in order to fully utilise the motion sensors for user tracking. There will be several suggestions made on orientation and distance measurement.

In the area of orientation measurement, the data taken from the experiment will be put into different cluster based on point of compass. However, as the movement will have more variation in real life, the cluster made to identify the direction must be more specific to identify a slight turn. A pattern mapping can also be made to identify the movement pattern of the user.

In the area of distance measurement, step counter has good accuracy as the sensor has been developed by Android longer than other sensors. However, another experiment must be made to identify different walking pace or the position of the phone during tracking to identify whether the step counter sensor can deal with different type of situation.

Also, linear acceleration is used to calculate distance on different type of movement. However, the level of noise is high and make the linear acceleration data unusable for distance calculation. More controllable experiments must be made in order to identify the noise and in order to develop specific filter to remove the noise from data. In addition, the linear acceleration fusion can also be enhanced by adding specific calculation on each axis to reduce the noise addition from accelerometer.

Bibliography

[1] G. Paller, "Better motion control using accelerometer/gyroscope sensor fusion," in *Droidcon Tunis 2012*, 2012.

[2] Roztahib, "Advances in development - last touches." Available at `https://rozsatib.wordpress.com`.

[3] P. Lawitzki, "Android sensor fusion tutorial." Available at `https://www.codeproject.com/Articles/729759/Android-Sensor-Fusion-Tutorial`, 2014.

[4] C. Peat, "Azimuth." Available at `http://www.heavens-above.com/glossary.aspx?term=azimuth`.

[5] P. Weston, "Wikimedia commons - k-means." Available at `https://commons.wikimedia.org/wiki/File:K_Means_Example_Step_2.svg`.

[6] A. I. S. Inc. and T. A. University, "Motion with constant acceleration." Available at `http://www.webassign.net/question_assets/tamucalcphysmechl1/lab_3/manual.html`.

[7] T. Engel, "Locating mobile phones using signalling system 7," in *http://events.ccc.de/congress/2008/Fahrplan/attachments/1262 25c3-locating-mobile-phones.pdf, 25th Chaos Communication Congress (25C3)*, 2008.

[8] Google, "Android open source project." Available at `https://source.android.com`.

[9] H. Xu, Z. Yang, Z. Zhou, L. Shangguan, K. Yi, and Y. Liu, "Indoor localization via multi-modal sensing on smartphones," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '16, (New York, NY, USA), pp. 208–219, ACM, 2016.

[10] D. Ayllón, H. Sánchez-Hevia, R. Gil-Pita, and M. Rosa-Zurera, "Indoor blind localization of smartphones by means of sensor data fusion," in *2015 IEEE Sensors Applications Symposium (SAS)*, pp. 1–6, April 2015.

[11] J. Han, E. Owusu, L. T. Nguyen, A. Perrig, and J. Zhang, "Accomplice: Location inference using accelerometers on smartphones," in *2012 Fourth International Conference on Communication Systems and Networks (COMSNETS 2012)*, pp. 1–9, Jan 2012.

[12] Y. Michalevsky, G. Nakibly, G. A. Veerapandian, D. Boneh, and G. Nakibly, "Powerspy: Location tracking using mobile device power analysis," in *Proceedings of the 24th USENIX Conference on Security Symposium*, SEC'15, (Berkeley, CA, USA), pp. 785–800, USENIX Association, 2015.

[13] Google, "Dashboard." Available at `https://developer.android.com/about/dashboards/index.html`.

[14] Google, "Sensor motion." Available at `https://developer.android.com/guide/topics/sensors/sensors_motion.html`.

[15] S. Wang, J. Min, and B. K. Yi, "Location based services for mobiles: technologies and standards," in *IEEE International Conference on Communications (ICC)*, 2008.

[16] H. Durrant-Whyte and T. C. Henderson, *Multisensor Data Fusion*, pp. 585–610. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[17] B. Litz, "Gyroscopic (spin) drift and coriolis effect," tech. rep., Applied Ballistics, LLC, 2008.

[18] Google, "Sensor event." Available at `https://developer.android.com/reference/android/hardware/SensorEvent.html#values`.

[19] U.S.Army, "Map reading and land navigation," tech. rep., Dept. of the Army, 1990.

[20] P. Classroom, "Lesson 1 - describing motion with words." Available at `www.physicsclassroom.com/class/1DKin/Lesson-1`.

[21] D. J. C. MacKay, *Chapter 20. An Example Inference Task: Clustering*, pp. 284–292. Cambridge University Press, 2003.

[22] R. Fisher, S. Perkins, A. Walker, and E. Wolfart, "Mean filter." Available at `https://homepages.inf.ed.ac.uk/rbf/HIPR2/mean.htm`.

[23] Google, "Rotation matrix." Available at `https://developer.android.com/reference/android/hardware/SensorManager.html#getRotationMatrix(float[],%20float[],%20float[],%20float[])`.

[24] B. Abernethy, V. Kippers, S. Hanrahan, M. Pandy, A. McManus, and L. Mackinnon, *Biomechanics Across the Life Span*, p. 131. Human Kinetics, 2013.

[25] CEACT, "Gps compass solutions, application -vs- accuracy," tech. rep., CEACT Information Systems, 2006.

[26] D. Ward, "Plot lat/long points on map by coordinates." Available at https://www.darrinward.com/lat-long/.

[27] Mapdevelopers, "Distance finder tool." Available at https://www.mapdevelopers.com/distance_finder.php.