Walking trajectory analysis using smartphone sensors

Li Yang 430452848

A thesis submitted in fulfillment of
the requirements of the degree of
Bachelor of Mechatronics Engineering

School of Aerospace, Mechanical and Mechatronic Engineering
The University of Sydney

Submitted June 2015

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the University or other institute of higher learning, except where due acknowledgement has been made in the text.

Li Yang

30 May 2015

Abstract

Over the last few years, various sensors are embedded into the smartphone that allows the device to accurately track the users' location and motion. Since the smart is our most essential and primary device to carry around, the build in sensors provide a solution to quantify our daily activities and provide health related information. Generally, the sensors that correspond to motion and location detection are accelerometers, gyroscope, magnetometer and GPS receiver, which are very common in this generation of mobile devices.

The aim of this thesis is to use the smartphone sensors to analysis human walking trajectory in an outdoor environment using the accelerometer, gyroscope, magnetometer and GPS receiver and plot onto the map.

Keywords: mems sensors, application development, data filtering, calibration, conversion, pedometer, trajectory, map.

Acknowledgements

This thesis would not have been possible without the support of a great many of my supervisor, family and friends …

Special thanks to all the people that inspired me …

After 5 years of oversea studies, from singapre all the way to sydney. I am truly blessed by all the wonderful people that I have meet and the amazing experience I have been through. Finally, I want to thanks my parent for supporting my decisions and always stands on my side.

Contents

Chapter 1

Introduction

1.1 Purpose and Motivation

Mems sensor and GPS receiver are largely embedded in current generation of smart phone devices that allows the application to obtain motion related information. Recently, more and more developers start to release various of health applications that are able to monitor users' daily activities and provide health suggestions. Other products like fitbit and jawbone based on similar principle are able to track human activities more accurately. The potential of smart sensors are truely humongous.
The aim of this thesis is to develop a pedometer using the smartphone's sensors and able to plot the walking trajectory to the map.

1.2 Scope

Here are the restrictions made in order to achieve optimized result: Accelerometers, magnetometers, gyroscope sensors and GPS receiver were the only sensors used in this work. The position of the mobile phone was restricted to the only placed leveled on the test subject's hand, and the experiment is always conducted in an outdoor environment with flat surfaces. The device used for testing is an iPhone5s and the coordinate system is defined as the following diagram (fig.1.2.1):

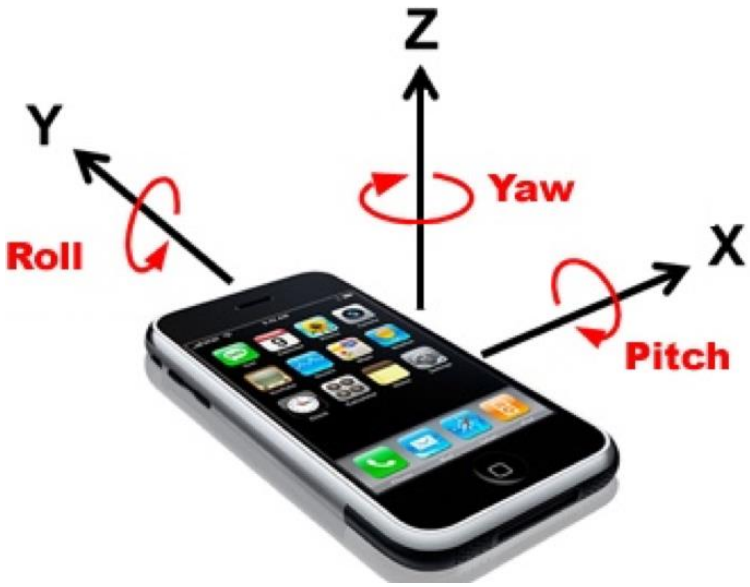


Figure 1.2.1 Assumption of the device coordinate system

1.3 Method

The accelerometer provides device orientation information as well as step detection and speed estimation while the gyroscope and magnetometer this in this application provides the heading information (yaw angle). Finally, the GPS updates position information. Complementary filter is used as a low pass filter to remove high frequency noises and a pedometer algorithm is developed to estimate the basic walking trajectory. At the end, data fusion is applied and as the result, a much more accurate trajectory is obtained as shown in the following diagram (fig. 1.3.1).

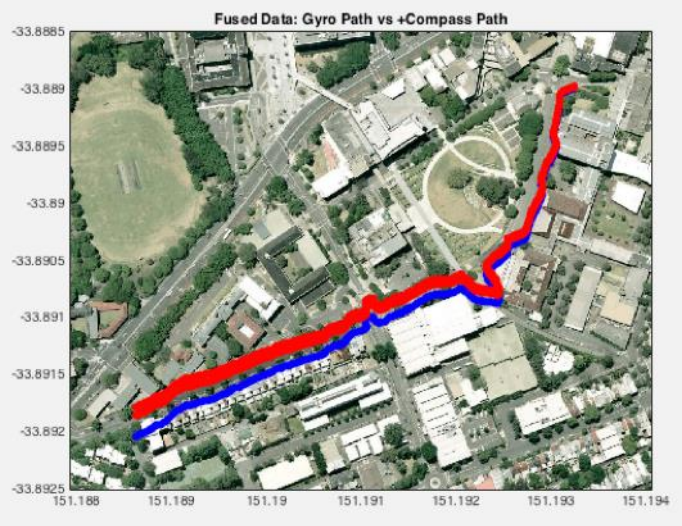


Figure 1.3.1 Example of the walking trajectory plotted on the map

Chapter 2

Background

This chapter reviews the working principle of each sensors and looks into their characteristics in details.

2.1 Working principle of sensors

2.2.1 MEMS sensor

The micro-eletroMechanical sensors (also called mems) are the new generation of sensors that manufactured using micro-electrical and micro-mechanical technologies. They are very compacted in size and have relatively low power consumption level. Therefore, it has been widely used in smart phone industry. In general, the most common mems sensors are accelerometer, gyroscope and magnetometer. These three basic sensors are considered as a motion hub of the device that provide relatively accurate device orientation information. Recently, more mems sensors are embedded into the smart phone, for instance, temperature sensor, altitude sensor and pressure sensors. Therefore, we have the reason to believe that the smart phone will becomes more interactive with the surrounding environments in the near future(IvenSense 2011).

2.2.1.1 Accelerometer

Accelerometer measures the combined linear acceleration of the device. The measurements of the accelerometer consists of static gravity acceleration (g) and dynamic acceleration generated by linear movement or vibration of the device.

To understand the working principle of the accelerometer, imaging a metal ball is sitting in the middle of a box with pressure sensitive walls (as showed in Fig 2.2.1.1.1). If there is no gravity, then the metal ball will simply float in the middle of the box. However, in reality the movement of the box results in location shifting of the metal ball and eventually it will hit the wall. For instance, in Fig 2.2.1.1.2 the ball touches the x- surface then the accelerometer output -1 g (9.8m/s²) on x axis. That is a very close model of the real accelerometer.

More commonly, the acceleration has been measured in x, y and z axis (Fig 2.2.1.1.3) and then it is better to considered the measurements in a 3D coordinate system as shown in the Fig 2.2.1.1.4. Based on the pythagorean theorem in 3D, the vector R is the overall acceleration measured by the sensor and Rx, Ry, Rz are projection of the R vector on the X,Y,Z axes. The relation between them is as follow:

Equation 2.2.1.1.1

$$R^2 = {R_x}^2 + {R_y}^2 + {R_z}^2$$

Lastly, accelerometer has two major the characteristics: provide accurate device orientation angle and sensitive to vibration of the device. On the downside, it suffers from high frequency noise, thus filtering is needed for raw data.

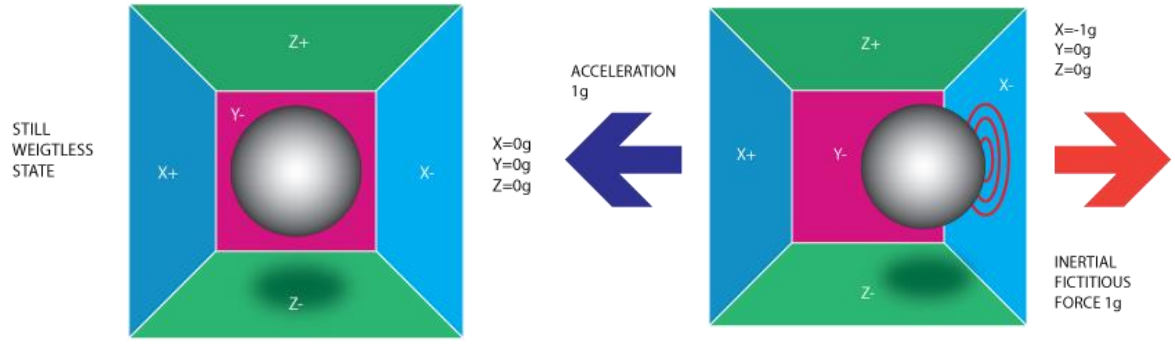


Figure 2.2.1.1.1 Accelerometer model:0 gravity (left)
Figure 2.2.1.1.2 Accelerometer model:1g acceleration on x axis(right)

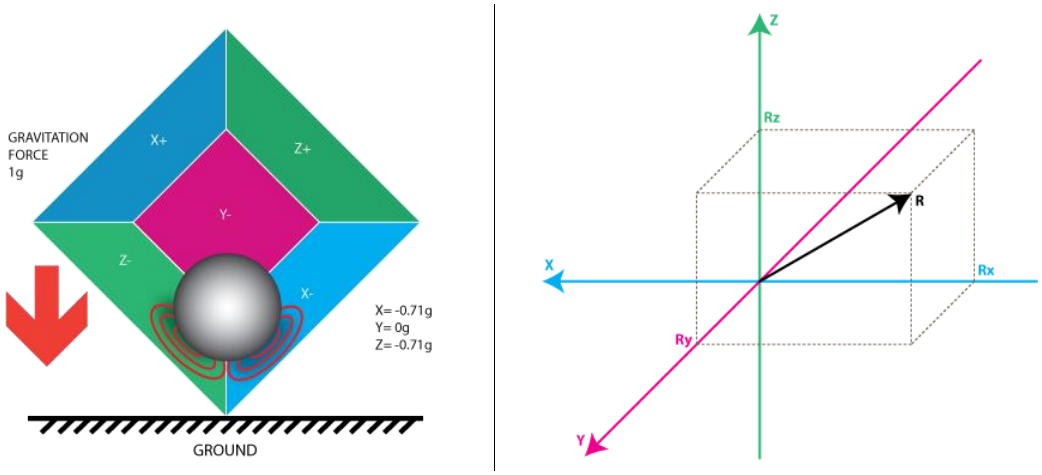


Figure 2.2.1.1.3 Accelerometer model: realistic situation (left)
Figure 2.2.1.1.4 Vector based accelerometer calculation(right)

2.2.1.2 Gyroscope

Gyroscope measures the angular speed around each axis. Compare to the conventional rotation gyroscope (Figure 2.2.1.2.1), the mems gyroscope is based on Coriolis effects, the physical principle is that a vibrating object tends to continue vibrating in the same plane as its support rotates. In the engineering literature, this type of device is also known as a Coriolis vibratory gyro (Figure 2.2.1.2.2) because as the plane of oscillation is rotated, the response detected by the transducer results from the Coriolis term in its equations of motion (also known as "Coriolis force").

Vibrating structure gyroscopes are simpler and cheaper than conventional rotating gyroscopes of similar accuracy. Therefore, it is largely used in this generation of smart phones.

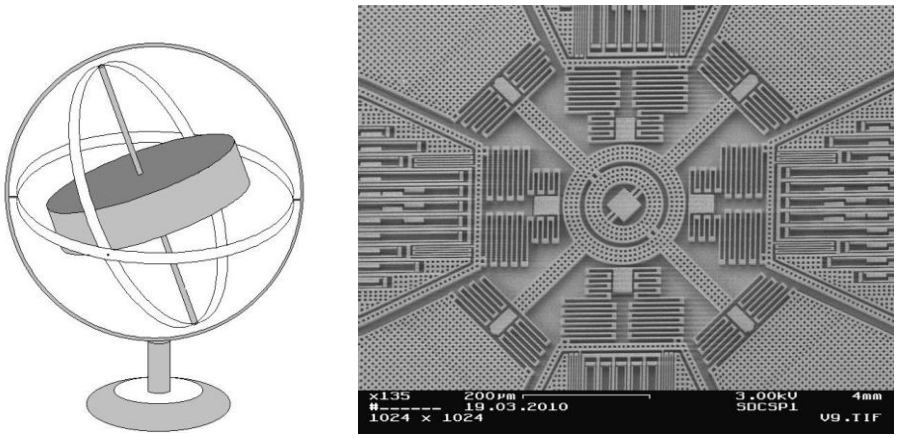


Figure 2.2.1.2.1 Conventional gyroscope model (left)
Figure 2.2.1.2.2 Mechanical structure of mems gyroscope(right)

The mems gyroscope suffers from a relatively constant bias and has to be removed before conducting any further calculation. Meanwhile, it is affected by surrounding electrical devices and suffers from random walk and white noises which may cause the value of the bias changes slightly. Other than that, the gyroscope is affected by temperature.

2.2.1.3 Magnetometer

The magnetometer is able to tell the absolute heading respect to the north. It measures ambient magnetic strength in unit od micro-Tesla.
A MEMS-based magnetic field sensor is small in size, and so it can be placed close to the measurement location and thereby achieves higher spatial resolution. Additionally, constructing a MEMS magnetic field sensor does not involve the micro fabrication of magnetic material. Therefore, the cost of the sensor can be greatly reduced.
On the other hand, the magnetometer is affected by the local magnetic field (hard iron distortion) and the surrounding electrical devices and circuits(soft iron distortion) which dramatically decrease the accuracy of the measurements. Therefore, calibration process has be applied before using.

2.2.2 Assisted GPS navigation system

The technology of used for navigation by this generation of smart phone is called assisted GPS navigation system. It uses GPS signal and the cellular network signals or even wifi. As shown in the following diagram(Fig 2.2.2.1), the basic working principle is that the GPS receiver receives GPS signal from multiple satellites, however it takes a long time to initialize. To improve that, the phone actually gets the initial location information from the closest cellular network station where the location information is saved locally. This combination accelerate the navigation speed and improve the accuracy to about 5-10 meters. In addition, the map information is downloaded to the phone through either wifi or the cellular network.

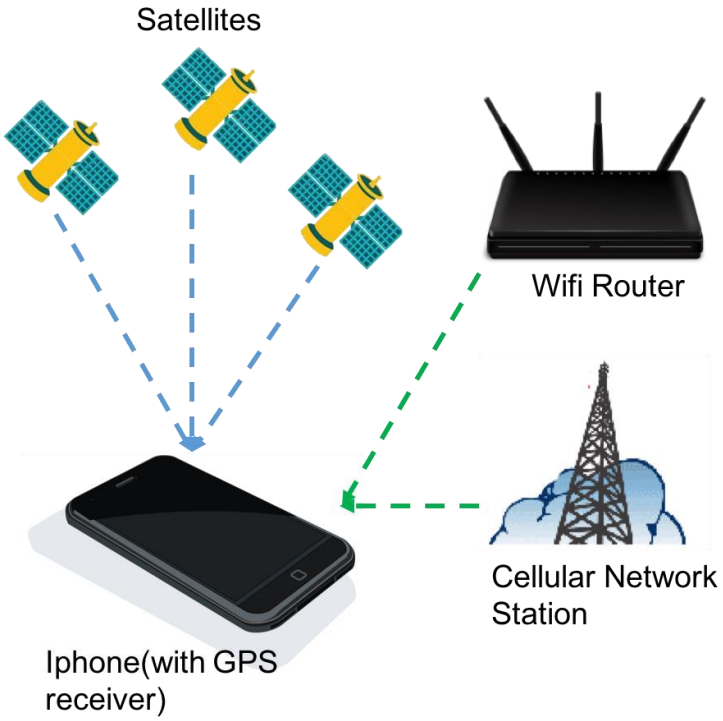


Figure 2.2.2.1 Assisted GPS system working principle

Chapter 3

Data acquisition

Sensors data are accessible through certain mobile phone applications. In this thesis, an application has developed through Qt platform at the early stage for initial testing purpose. During the later stage of outdoor experiment, another application called 'Sensor Log' is used to efficiently gather all sensors' data and send back to the laptop wirelessly. The following section will discuss both of them in details.

3.1 Develop application using Qt: SensorLogFile

3.1.1 What is Qt

Qt is a cross-platform software development environment (Rischpater 2014) based on c++ that allows the same source code to be implanted into various hardware and software platforms. In this thesis, Qt is used to develop an ios application for the iPhone.

The reason to use Qt is as following: fisrt of all, it has an easy-to-use interactive GUI[1] design interface and secondly, the program developed for ios platform could easily replant for the Android devices.

The version of the Qt used to develop this application is Qt 5.4.0(Clang 6.0(Apple),64 bit).

[1] GUI: stands for graphical user interface

3.1.2 Application development

The main purpose of the application is to gain access to the iPhone sensors, then log the real time sensors data into files. Finally, send back to PC for further analysis. Overall, there are three part of the program: GUI, back-end data handing and creating the log file. The relationship between these three parts of the program is shown in the following state transition diagram. It is noticeable that the application is only capable of logging data from accelerometer, gyroscope and magnetometer. The GPS data requires higher authorization from apple, therefore, it is not available in this application.

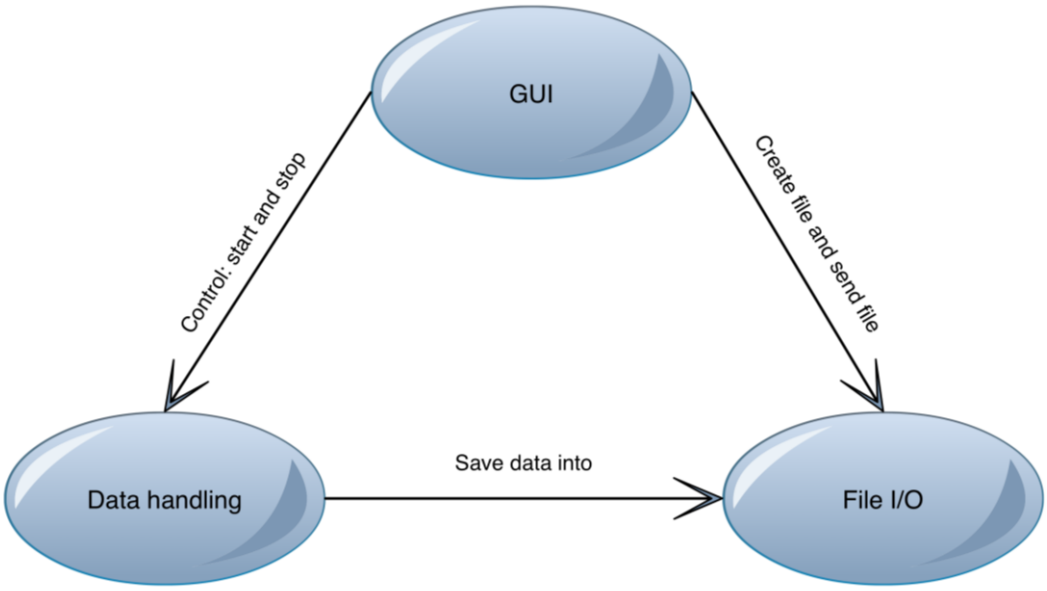


Figure 3.1.2.1 Application state transition diagram

3.1.2.1 GUI Interface

The GUI interface let the user control the data acquisition process and monitor real-time sensor data on the screen. Furthermore, it allows the user to save the current sensor logging file into the memory. The figure 3.1.2.2 below demonstrates the functionalities of the GUI.

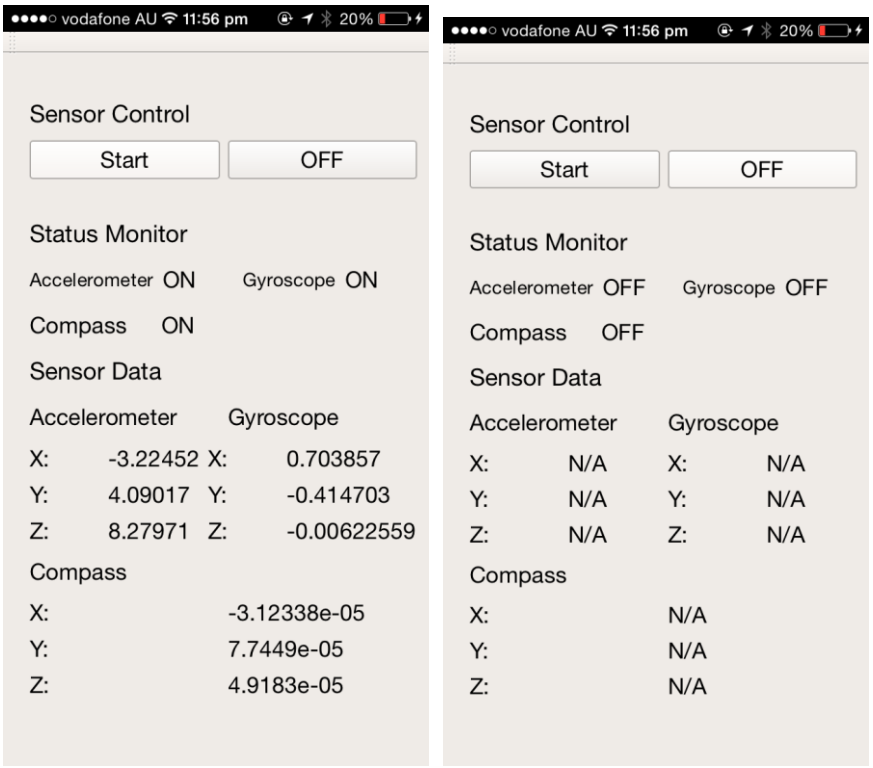


Figure 3.1.2.2 Application GUI interface

3.1.2.2 Sensor data handling

After the user presses the start button on the screen, the back-end program activates the sensors and start to send sensors' readings at a specific frequency. Meanwhile, it constantly updates sensors' status (on or off). The sensors' data are sent to GUI, log file and serial port. Therefore, the user is able to monitor sensors' data in various methods.

Another function of the back-end data handling program is to log time information. When the sensors activates, the program is able to log the current time and updates when new batch of sensors' data flush in.

3.1.2.3 Creating logging file

The sensors data is saved to the applications document directory by using Qt Qfile class. For iPhone, there is an extra process to gain access to this document directory. Once the application is successfully deploy to the device, it is necessary to go back to the deploy folder on the computer and change settings in a file called “info.plist”. Basically, this file controls the properties of the application. Therefore, in order to access the file in the laptop when the phone connects to PC through USB cable, a property (Key) called “Application supports iTunes file sharing” with value “YES” need to be added to the key list. Then the same application need to be deployed again to update the settings.

Furthermore, the iPhone document directory is dynamic, hence it is not possible to save to a fixed location. Instead, the program gets the dynamic document directory every time by calling following code:

```
QStandardPaths::strandardLocations(QstrandardPaths::DocumentLocation);
```

After that, the program checks if this directory is available, if yes, then the program create new file inside the directory and starts to log all available sensors data. For this application, all the sensors readings are logged into a single file with related time information. In the appendix B, there is a sample log file.

To access the log file on the laptop, the user need to connect the phone to the laptop through the USB cable. Then open iTunes>devices>Apps>File Sharing>Click on the deployed application(“SensorSendFile”), the log file should appears in the right hand side documents list. In appendix A, there is code for the application.

3.1.2.4 Comments

The application has the ability to log all the necessary information, however, the sensor logging frequency is controlled in the back-end program which is not accessible through the GUI. Other than that, the process of saving the log file to the laptop is not automated, not to mention the trouble of using iTunes.
Overall, this application is a good approach to understand how to access sensors' information from iPhone.

3.2 Sensor Log

Sensor Log is an ios platform application that used for logging sensor data. It is capable of sampling data up to 100Hz and able to send sensor data via email as a csv file or stream it via tip/ip. (Alan, Arnrich et al. 2014)The application has a nice designed GUI that allows user to monitor real time data intuitively as a time based logging diagram(Fig 3.2.1).
For iPhone5s, the following information are available:

- GPS location: Longitude and latitude
- Accelerometer
- Gyroscope
- Magnetometer
- Pedometer(count steps)

The following diagrams shows the interface of the application. The user is able to choose which sensors to activate and modify the sample frequency from 0 to 100 Hz. Furthermore, it allows the user to change between different sensors' interfaces to monitor the real time data. In the appendix B, there is a sample log file.

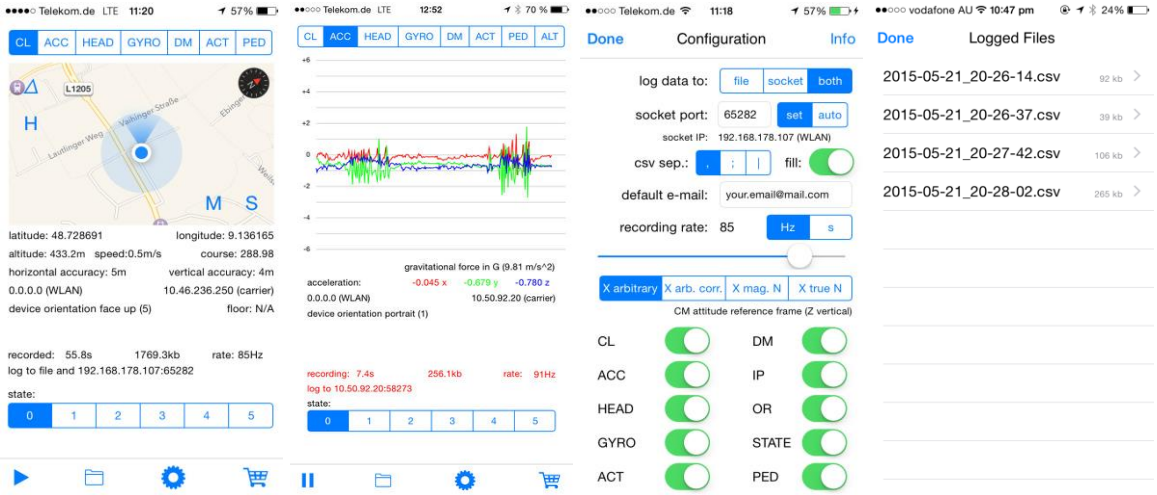


Figure 3.2.1 Sensor log GUI interface

3.3 Experimental data collection process

Just to clarify the data collection process: During the experiment, three set of data will be collected. The 1st set is used for gyroscope calibration, the 2nd set is used for magnetometer calibration, and finally, the 3rd set of data collects sensors' information when the test subject walks around with the phone lays flat on his/her hand. In the following article, I will describe these three sets of data as set 1, set 2 and set 3 respectively.

3.4 Data processing model

In this thesis, the iPhone is charged of collecting sensors' raw data and save into .csv files. Next sensor data files send to the PC through email. For the purpose of further analysis, the .csv files are imported to matlab. As demonstrated in the following diagram (Fig 3.4.1).

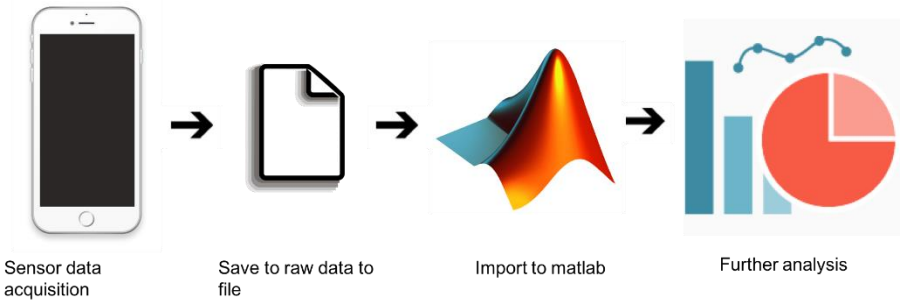


Figure 3.4.1 Workflow of the project

3.5 Summary

The Qt application SensorLogFile:

The application has the ability to log all the necessary information, however, the sensor logging frequency is controlled in the back-end program which is not accessible through the GUI. Other than that, the process of saving the log file to the laptop is not automated, not to mention the trouble of using iTunes.

Overall, this application is a good approach to understand how to access sensors' information from iPhone.

Senor Log:

This application is fully capable of logging sensor data and most importantly, it is able to send file wirelessly through email. However, the application set the same sample frequency for all sensors, which is not exactly correct. As mentioned in the chapter 2, different type of sensor can only update fast enough to certain frequency. For instance, the iPhone GPS receiver update frequency is only up to 1Hz. Therefore, the application itself must have some kind of data processing and filling ability. To solve this problem, later in the data fusion process, the GPS and the compass data is resampled(lower the sample frrquency) to match their real performance.

Chapter 4

Data filtering and conversion

The raw data we received from the sensors need further filtering and conversion in order to be used in the later data fusion process. Some sensors even need calibration. As mentioned above, all the data have to convert into either position or heading angle information. In the following sections, I will discuss the sensor filtering, calibration and conversion in details.

4.1 Accelerometer

The accelerometer measures the combined (gravity and device linear movement) linear acceleration of the device.(Unit: m/s^2). As mentioned above, it is sensitive to vibration and able to provide an accurate device orientation information. The aim is to convert the accelerometer data into pitch and roll angle that able to represent device orientation.

4.1.1 Data filtering: Complementary filter

The accelerometer suffers from high frequency noise due to fact that it is a low cost MEMS sensor. To demonstrate, in the following figure 4.1.1.1(left), the phone is placed on a flat surface for about 9 seconds. From the raw data plotting, it is obvious to see that the accelerometer do have quite amount of noises.
The common way to get rid of the noise is to use a low pass filter. The most popular

low pass filter is complementary filter and kalman filter. In this thesis, complementary filter is used because it is easy to understand and to program. Furthermore, based on the feedback from the other researchers, in terms of filtering accelerometer, both of them have similar filtering effect(Georgy, Iqbal et al. 2009).
The first order complementary filter as shown in the following formula (Equation 4.1.1.1) combine the current data with the previous data and by adjusting the weight(alpha) value (between 0 to 1) to change the filtering effect.

Equation 4.1.1.1

$$accelFilrted_K = \alpha \times accel_K + (1 - \alpha) \times accel_{K-1}$$

Based on testing, the first order complementary filter is not sufficient enough to filter out most of the high frequency noise, therefore, in this thesis, I added a for loop to this first order complementary filter to enhance the effect.
Overall, to adjust the filtering strength, both the weight (alpha) and the number of loops contributes to the final results.
It needs to be point out that using filter results in certain degree of delay. Therefore, both the alpha and the number of loops has to be carefully selected to minimize the delay effect. The table below (table 4.4.1.1) shows the delay effect of using different combination of values.
The best combination is when selecting alpha as 0.5 and number of loops as 4. This combination sufficient reduce the high frequency noise without too much compromise on time delay. The final filtering result is shown the figure 4.1.1.1(right)
For more accelerometer filtering comparison, please check appendix C.

	alpha	N(no. of loops)	Delay time
Test 1	0.3	15	3.58 sec
Test 2	0.5	4	0.12 sec
Test 3	0.5	6	0.18 sec

Table 4.4.1.1 Workflow of the project

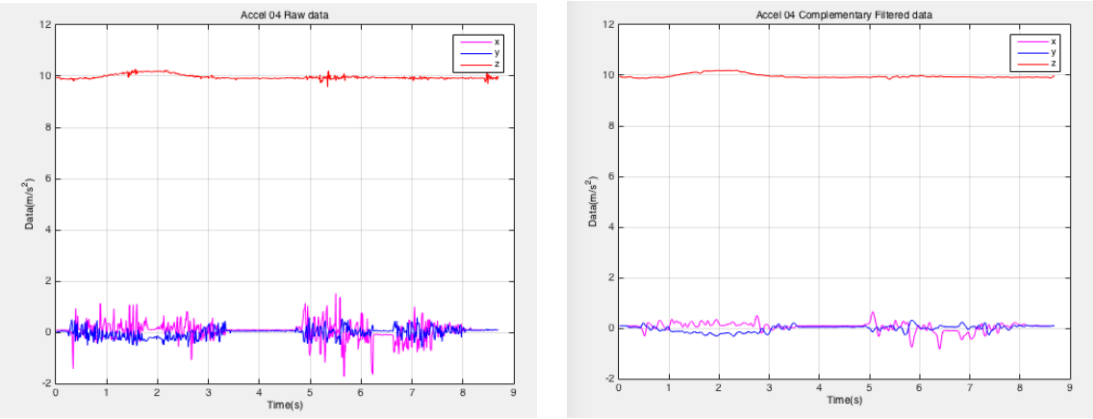


Figure 4.1.1.1 accelerometer filtering

4.1.2 Data conversion

The aim of this section is to convert the accelerometer data into pitch and roll angle. The accelerometer itself measures linear acceleration on x, y and z axis respectively. These three measurements are the component vectors of the device overall acceleration vector R that projects to each axis (as shown in the following figure 4.1.2.1 Rx, Ry and Rz). Essentially, the pitch(tilt angle of y axis) and roll angle(tilt angle of x axis) that represent the device orientation is equal to the Ayz and Axz on the diagram below.

At this point, the conversion from acceleration data to angle data is simply a vector calculation.

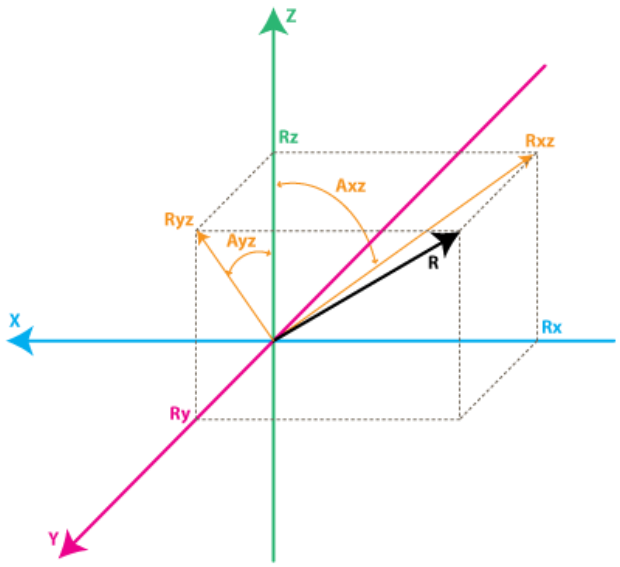


Figure 4.1.2.1 accelerometer space vector model
where Rx,Ry,Rz:linear acceleration of each axis
R: 3 dimensional vector represent overall linear acceleration
Ayz, Axz: Pitch and roll angle

The following two sets of formulas are both used to calculate pitch and roll angles based on the acceleration vectors:

Equation 4.1.2.1

Set 1: use 3 sets of data at the same time:

$$A_{yz} = \tan^{-1}\left(\frac{R_Y}{\sqrt{{R_X}^2 + {R_Z}^2}}\right)$$
$$A_{xz} = \tan^{-1}\left(\frac{R_X}{\sqrt{{R_Y}^2 + {R_Z}^2}}\right)$$

Set 2: use 2 sets of data at the same time:

$$A_{yz} = \tan^{-1}(R_y / R_z)$$
$$A_{xz} = \tan^{-1}(R_x / R_z)$$

The conversion results from both set of the formulas are the same. It is worth mentioning that accelerometer data is not capable of getting yaw angle out of the three component vectors, owing to the fact that accelerometer characteristics of measuring linear acceleration. The following diagram shows the conversion result: tilt the iPhone for 90 degree on both x axis and y axis. For more accelerometer conversion comparison, please check appendix C.

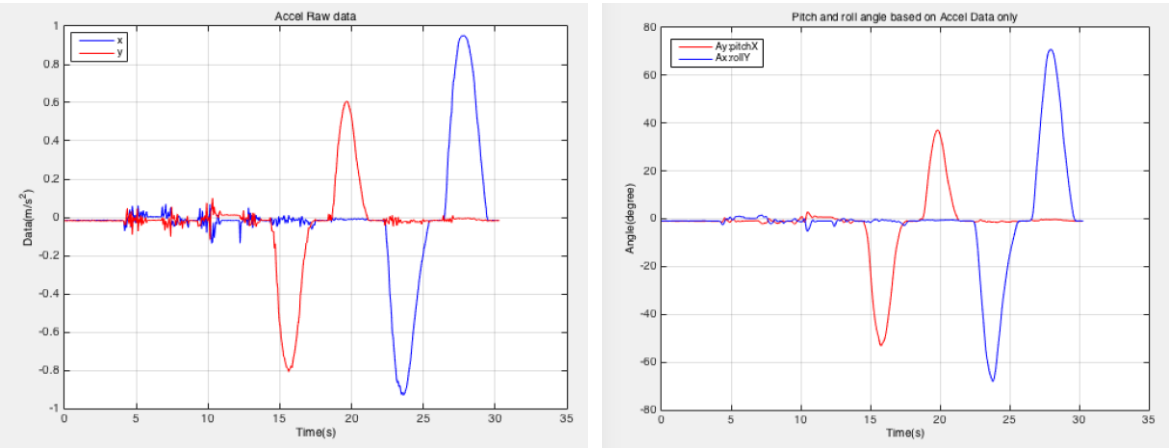


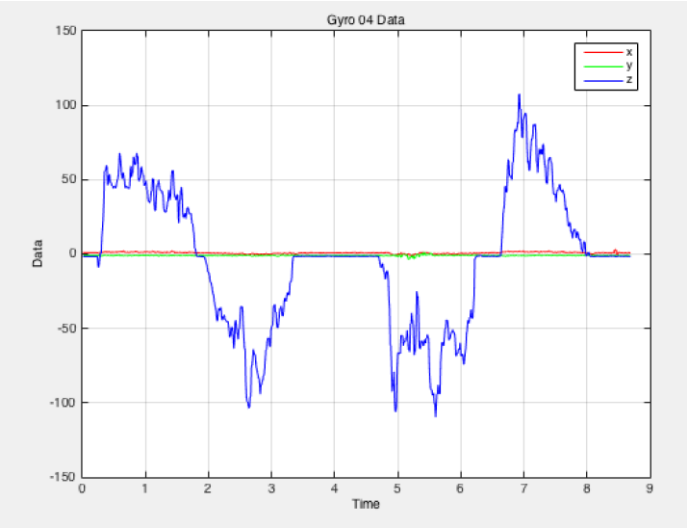
Figure 4.1.2.2 accelerometer conversion result

4.2 Gyroscope

Gyroscope measures the angular speed around each axis. A quick recap from chapter 2, the gyroscope response fast to angle changes but not sensitive to the device vibration. In this thesis, it provides heading (yaw) angles. The sensor is capable of calculating pitch and roll angle as well, but since the experiments are based on the assumption that the phone lies in the hand of the testing subject levelly all the time during experiment, therefore, the yaw angle is only angle from gyroscope that is useful(STMicroelectronics 2010).

4.2.1 Remove bias

Gyroscope suffers from constant bias when the device is sitting still. Since the output of the gyroscope is integrated to find the orientation angle, the constant bias error grow linearly with time. To remove this bias, every time before the experiment, the phone is placed on a flat surface for around 20 second to collect a set of calibration data (Set 1). Then the constant bias value is obtained by taking average of the gyroscope output set 1. Lastly, remove this bias from the real experiment data set 3. The constant bias is shown in the following diagram(Fig 4.2.1.1).



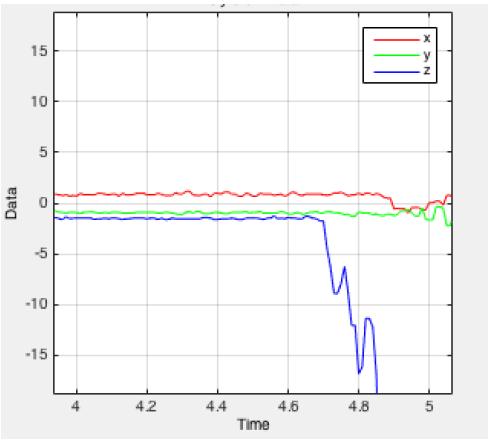


Figure 4.2.1.1 gyroscope bias(left)
gyroscope bias zoom in to details(right)

4.2.2 Data conversion

As mentioned above, the gyroscope measures the angular speed, hence, the orientation angle is obtained through integration. Within the program, the current gyroscope angle is obtained by adding previous angle with the changed angle $\omega\Delta t$, which is the product of angular velocity and time interval in between(as shown in the following formula).

Equation 4.2.2.1

$$Yaw_k = Yaw_{k-1} + \omega\Delta t$$

Meanwhile, as the gyroscope angle keeps accumulating, there is possibilities that the angle might exceeds 360 degrees or goes below 0 degree. Therefore, limitations have to be set to examine the current orientation angles and make correction to fit in the range between 0 and 360 degrees.

Conversion result:

The following diagram (Fig 4.2.1.1) shows comparison between gyroscope angle (yaw) conversion results with and without removing the constant bias. It is clear that after removing the constant bias, the integration angle does not shift away.

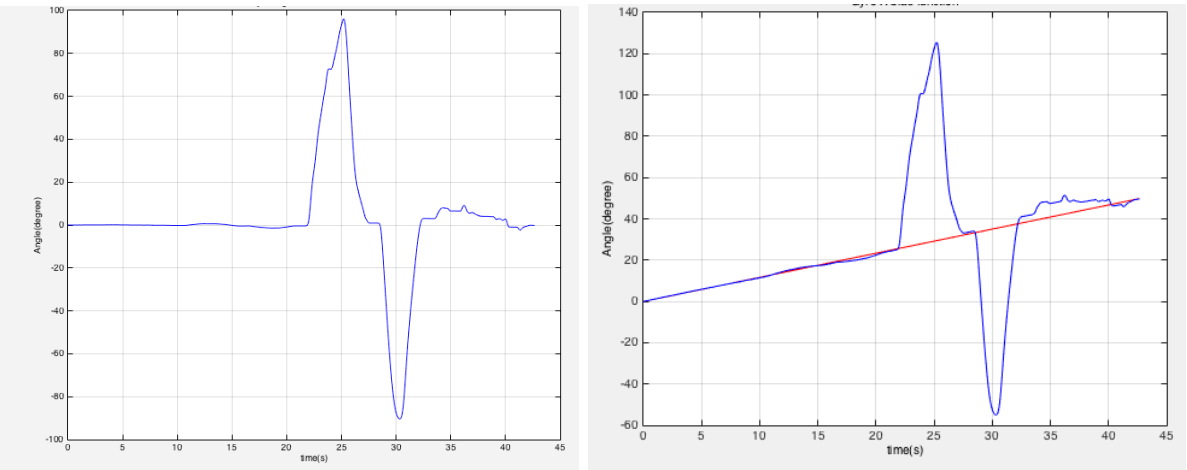


Figure 4.2.1.1 gyroscope conversion result:
Without bias(left), with bias(right)

Additionally, it need to mention that the gyroscope also suffers from random walk and white noises error. More specifically, if the gyroscope remains active for too long, then bias tends to shift slightly which results in dramatically changes in the orientation angle. In this thesis, this problem is solved by adjusting gain factor during the integration process. This solution is not perfect but it is good enough for the current application.

4.3 Magnetometer

Magnetometer also known as E-compass measures the magnetic strength (unit: μT mirco-Tesla) surrounding the device (Wahdan, Georgy et al. 2015)using hall effect as mentioned in chapter 2. Without any kind of calibration, the accuracy is only up to few tens of degree variation. Since the output of the magnetometer is the heading angle (represent walking direction), therefore, it is important to minimize the heading error. In the following sections, several ways of calibration will be discussed.

4.3.1 Magnetometer calibration

4.3.1.1 Calibration data collection methods

If you have noticed the iPhone compass application, when the user first enter the app, there is a calibration process. During this process, the application requires the user to hold the phone steadily in their hand and rotate around respect to a fixed origin point (as shown in the following figure 4.3.1.1.1). The purpose of this process is to get

enough sample points to reduce all kinds of magnetometer errors. For this thesis, similar technics are applied.

As mentioned above, the 2nd set of data is used for magnetometer calibration and the data collection process is really critical to the compass heading accuracy. Thus, special calibration data collection process has to be applied. In details, there are two commonly used methods: surface rotation calibration and 3D "∞" calibration.

Figure 4.3.1.1.1 compass application calibration

Method 1: Surface rotation calibration

Surface calibration is a simplified and effective way to collect sufficient magnetometer calibration data. The data collection procedure is as follow:

1. Place the phone screen face up on a flat surface and rotate the phone for 360 degree(figure 4.3.1.1.2 a) and collects changes in x and y axis
2. Then place the phone vertically as shown in the figure 4.3.1.1.2 b and rotate the phone again for 360 degrees. This step collect changes in x and z axis.
3. Finally, lean the phone on the its longer edge then rotate in this orientation for 360 degrees (figure 4.3.1.1.2 c) and collects changes in y and z axis.

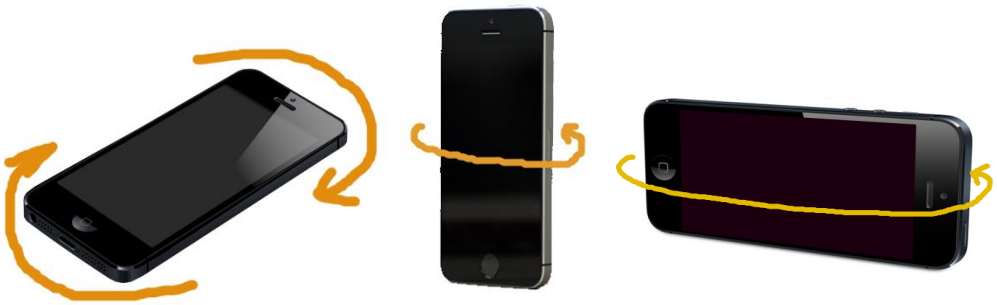


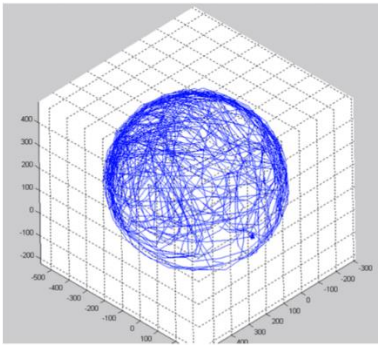
Figure 4.3.1.1.2 compass calibration method 1
A: horizontal orientation
B: vertical orientation stands on shorter edge

C: vertical orientation stands on longer edge

This method collects 3 set of data for the rotation around x,y and z axis, which latter allow the program to find out the offset value for each axis. This will be described latter in 4.3.1.3.

Method 2: 3D"∞" calibration

The basic principle of this method is that if the device held steadily in the users hand and keep rotating as a "∞" shape in space respect to a relatively fixed original point, the output geometrically is supposed be a perfect sphere(as shown in the following figure 4.3.1.1.3). However, due to the hard iron and soft iron errors, the output is generally an ellipsoid with central offset. Therefore, more data on the sphere are collected, more accurate the calibration will be.



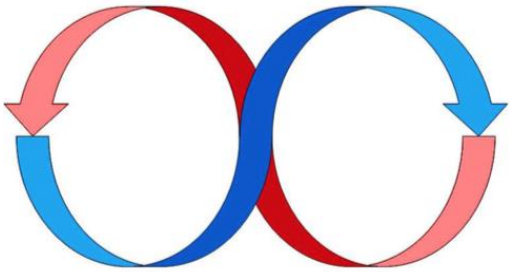


Figure 4.3.1.1.3 compass calibration method 2 outline(left)
Figure 4.3.1.1.4 compass calibration method 2 rotation(right)

This data collection methods requires the user to rotate the device in a "∞" shape (as shown in the figure 4.3.1.1.4) in all direction, in principle, try to cover the data from all orientations.

Comparison between two methods:

The first method has strict requirements on phone orientation, therefore, it is less reliable than the 2nd method. Both of them are not perfect and requires the user paying close attention during the data collection process. For instance, if the phone's origin shift during the process, then this set of data is not useable.

Based on the output(heading) and compare with iPhone's reference data(true heading provided by xcode), method 2 is selected.

4.3.1.2 Ellipsoid fit matlab tool box

As mentioned in chapter 2, the major errors of magnetometer comes from hard iron and soft iron errors. Meanwhile, from previous section, we know that the trajectory of the device using method 2 is supposed to be a sphere. Essentially hard iron and soft iron errors represent the shift of the origin and the ellipsoid shape of the surface (as shown in the following figure).

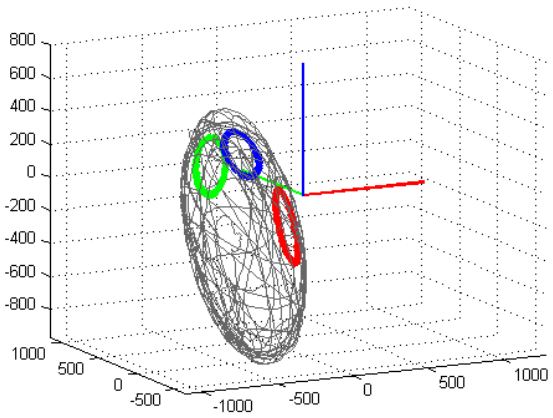


Figure 4.3.1.2.1 Uncalibrated data from magnetometer

To find the origin and fit the ellipsoid shape, a matlab toolbox called "Ellipsoid fit" is used[2]. This toolbox has the functionality to fits an ellipsoid into a 3D set of points and return critical information about the ellipsoid shape(like, centroid location, ellipsoid radius in x,y,z direction, etc). In short, here are the input and output:

Input: magnetometer calibration data

Output: centroid location, ellipsoid radius, eigenvector and algebraic factors to form the ellipsoid equation.

[2] Download link: http://www.mathworks.com/matlabcentral/fileexchange/24693-ellipsoid-fit

4.3.1.3 Remove hard iron error

From the calibration data set and the ellipsoid fit toolbox, the sphere centroid location can be used to remove the hard iron error (Wahdan, Georgy et al. 2015) for set 3 of the experimental data (normal walking data). The method is just simply shift all the magnetometer data back to the origin by using following formula:

Equation 4.3.1.3.1:

$$X_{afterHardIron} = X_{magnet} - ellipsoidCenter_x$$
$$Y_{afterHardIron} = Y_{magnet} - ellipsoidCenter_y$$
$$Z_{afterHardIron} = Z_{magnet} - ellipsoidCenter_z$$

Where $X,Y,Z_{afterHardIron}$: Magnetometer data after removing the bias
X,Y,X_{magnet}: Raw magnetometer data
$ellipsoidCenter_{x,y,z}$: ellipsoid sphere center locations

4.3.1.4 Remove soft iron error

For the calibration data set 2, soft iron error distorts the magnetometer data from perfect sphere into ellipsoid shape. Therefore, in order to remove the soft iron error, the magnetometer data has to be refitted onto the sphere. Here, the eigenvector and algebraic factors from the ellipsoid toolbox helps to recreate the sphere shape. Basically, we use the shortest radius from the ellipsoid and recreate the sphere, then refit all the data on to the sphere. Later for the set 3 experimental data, the same the eigenvector and algebraic factors are applied, so that the calibration is automatically done. The details are in the matlab code named: B1F7_MagnetCalibration, please check the appendix A for more details.

4.3.1.5 Declination Correction

After successfully removed the hard iron and soft iron distortion, the last step is to compensate the magnetic declination. The current location declination data is measured by www.magnetic-declination.com (as shown in the following figure). Since it is the south hemisphere, the declination is positive. Then in the program, directly add this data to the all magnetometer readings.

Figure 4.3.1.5.1NSW magnetic declination is around 12°26'

4.3.1.6 Calibration result

The following diagrams display the calibration process and result. On the left, the blue line represents the raw data and the red line represents calibrated data after removing hard and soft iron distortion. It is obvious to see both the shift of the centroid location and the changes of the outer shape. To examine whether the calibration data is valid, in the figure on the right hand side, the calibrated magnetometer data are projected to the idea sphere. If most of the data points lays on the surface of the sphere, then the calibration process is valid. Up to this point, the

calibration of magnetometer is done.

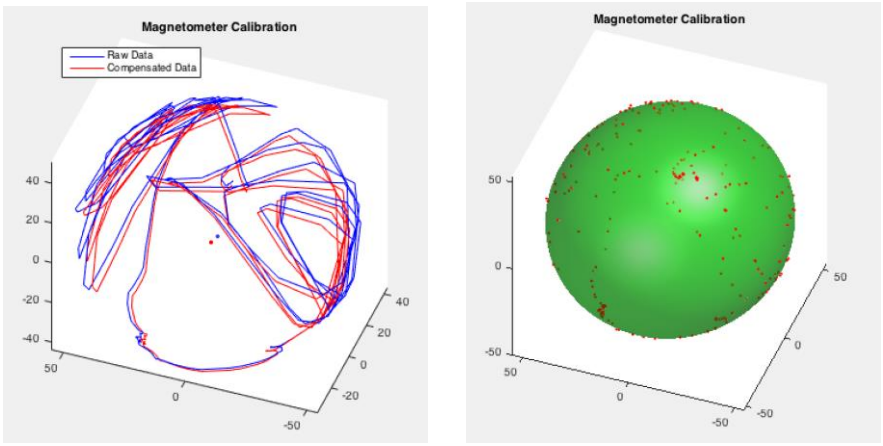


Figure 4.3.1.6.1 Magnetometer calibration

4.3.2 Data conversion based on tilt compensation

Heading Calculation

Even though for compass application, only the heading (yaw angle) is interested, we do have magnetometer readings in x y and z directions. Therefore, the conversion between magnetic strength measurements and heading angle is necessary.

First of all, assume the device is at a leveled position, the device's pitch and roll angle are 0 degree, then the heading is calculated based on following diagram.(STMicroelectronics 2010)

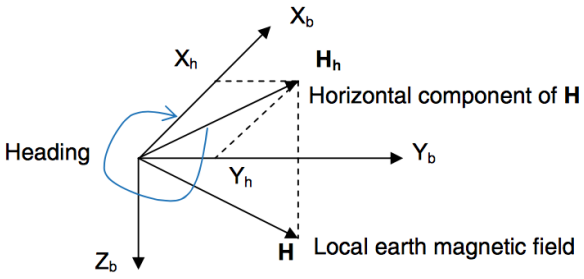


Figure 4.3.2.1 Heading calculation
Where H: Local magnetic field
Hh:Fixed horizontal component
Xb Yb Zb:magnetometer axis
Xh Yh: Component of Hh

Local earth magnetic field H has a fixed component Hh on the horizontal plane pointing to the earth's magnetic north. This component can be measured by the magnetometer then converted to X_h and Y_h. Hence, the heading angle is calculated

as:

Equation 4.3.2.1

$$Heading_{mag} = \tan^{-1}(Y_h / X_h)$$

Tilt Compensation

In practice, the device is not always at the leveled position, owing to the fact that the device is always affected by the movements. In this case, the pitch and roll angles are not equals to 0 degree (demonstrated in the following figure).(STMicroelectronics 2010)

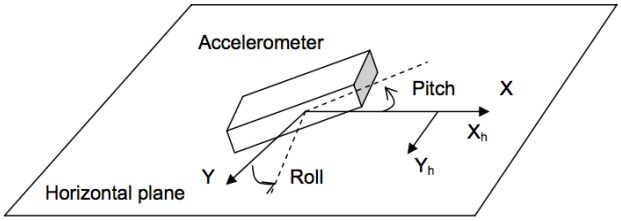


Figure 4.3.2.2 Tilt compensation

To compensate, the pitch and roll angle measured by a 3-axis accelerometer provides the device orientation information. Therefore, the magnetometer measurements X_m Y_m and Z_m need to be compensated to obtain Xh and Yh and then use the previous heading calculation equation to obtain the accurate magnetic heading. The tilt compensation formula is as follow:

Equation 4.3.2.2

$$X_h = X_m \cos(pitch) + Z_m \sin(pitch)$$

$$Y_h = X_m \sin(roll)\sin(pitch) + Y_m \cos(roll) - Z_m \sin(roll)\cos(pitch)$$

$$Heading_{mag} = \tan^{-1}(Y_h / X_h)$$

Where, XM, YM, and ZM are magnetic sensor measurements.

Conversion result

The following diagram shows the magnetic heading after conversion. The result is compared with a reference angle provided by iPhone called “trueHeading” which is the fully calibrated data using apple’s algorithm.

To conclude, the calibration and conversion is successful, even though it is slightly noisy, kalman or complementary filter could applied to reduce the noise.

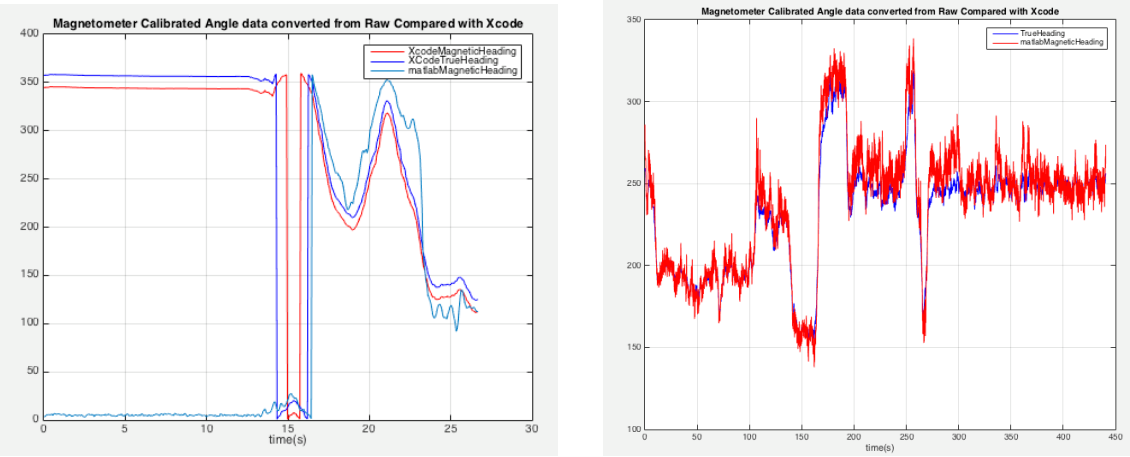


Figure 4.3.2.3 conversion result

4.4 Summary

Up to this point, all the mems sensors’ data have effectively converted to angles which will be used for step detection and data fusion in later process.

Chapter 5

Walking pattern analysis

In this section, the methods of detecting steps and estimating walking speed will be discussed. The aim is to develop a pedometer algorithm that is able to plot basic walking trajectory.

5.1 Assumptions

The data used for the pedometer algorithm are collected during the outdoor experiment and the assumptions are as follow:

- The experiments are always conducted on flat surface in the outdoor environment.
- The walking is continuous and maintained at a relatively constant speed.

5.2 Methods

5.2.1 Step detection

The following diagram demonstrate the basic human walking pattern. It is obvious that when the human walks, the body's central of gravity shift up and down which generates vibration. This vibration eventually passes to the iPhone then affects the sensor's data the test subject collects. As mentioned in chapter 2, the accelerometer is sensitive to vibration, therefore, in this case the pitch and roll angle converted from accelerometer are used to detect the steps.

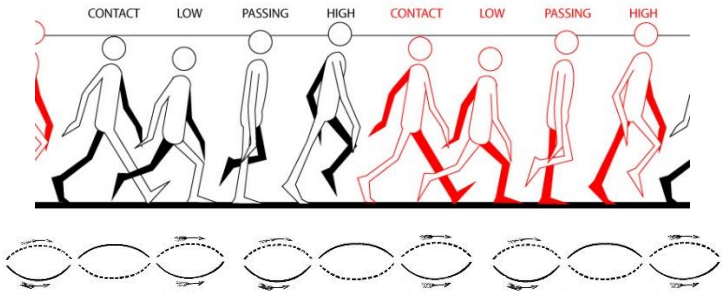
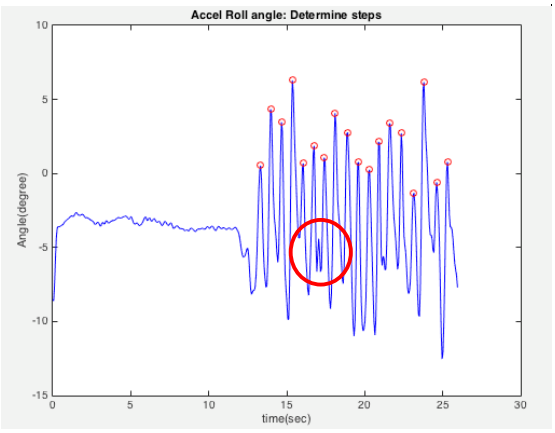


Figure 5.2.1.1 human walking pattern

Essentially, the step detection is to find peaks in the roll angle data as shown in the following diagram on the left. In details, to find the correct peaks that represent each step, two important factors have be considered: the strength of the step and the time interval between each step. For instance, the strength of the steps respond to the amplitude of the peaks. As to the time intervals between steps are respond to time difference between peaks. Thus, based on these two criteria, it is possible to remove unwanted peaks (red circle) and return relatively accurate step information. The figure on the right hand side shows a longer walking experiment for about 4 minutes. The same algorithm picks up the steps really well.



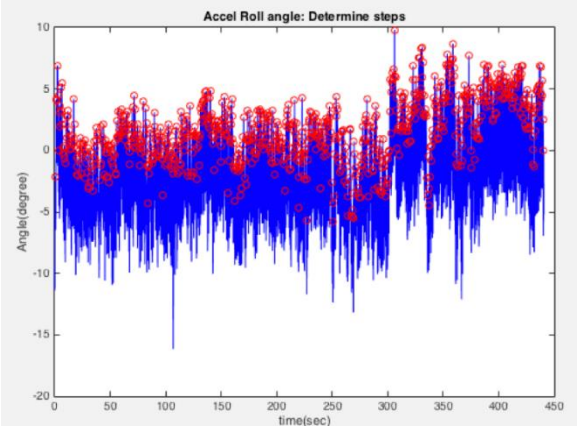


Figure 5.2.1.2: 18 steps in 30s Figure 5.2.1.3: 809 steps in 450s

5.2.2 Speed estimation

Speed can be easily estimated since the step information and the time interval for each step are recorded. It is notable that since I am the only test subject, thus my step size is constant at 65cm and this data is critical for calculating speed. The formula used to calculate speed per step is as follow:

Equation 5.2.2.1

$$speed\ per\ step = \frac{step\ size}{time\ per\ step}$$

To examine the result, the speed is comapred with a reference speed information that provided by GPS(as shown in the following figure). The ralted progam in matlab is B1F9_FindStepsFromAccelRoll, please check appendix A more detials. Overall, the speed estimation is accurate, however, during the experiment, the step size is not always a constant. Therefore this assumption will affects the result slightly.

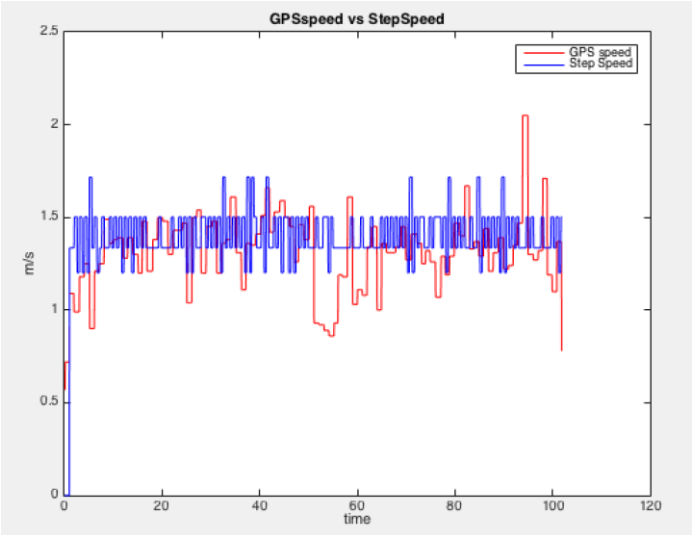


Figure 5.2.2.1: Speed estimation

5.2.3 Pedometer algorithm

The pedometer algorithm is aimed to plot basic walking trajectory using heading angle from gyroscope and the step information from the above.

Initially, the algorithm just use step size and the number of steps to plot a straight line (as shown in the following figure on the left) that represent the absolute distance from the start point to the end point. Then at the end of each step, an arrow represents

the heading angle is plotted in orange. Next, to get the trajectory, the heading angle is added to the calculation using the following formula and the result is shown on the figure on the right. From this pedometer algorithm, the walking trajectory can be estimated.

Equation 5.2.3.1

$$X_n = X_{n-1} + stepSize \times \cos(headingAngle)$$
$$Y_n = Y_{n-1} + stepSize \times \sin(headingAngle)$$

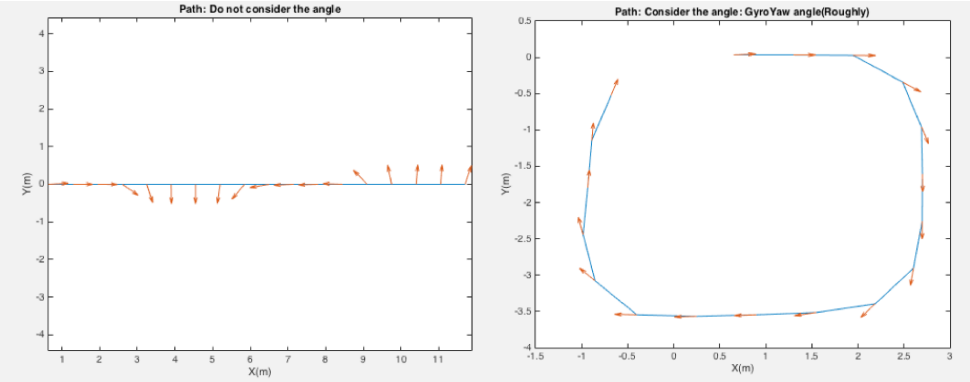


Figure 5.2.3.1: Trajectory estimation

Improvement of the pedometer algorithm using speed information:

A better approach to estimte the trajectory using only gyroscope heading angle and the step information is to estimate using the speed per step. So in this algorithm, the heading angle updates more frequently at the gyroscope sample frquency. Futhermore, the distance information is calculated using speed and time. The formula for this kind of algorithm is:

Equation 5.2.3.2

$$X_n = X_{n-1} + currentSpeed \times \frac{1}{sampleFrequency_{gyro}} \times \cos(headingAngle)$$
$$Y_n = Y_{n-1} + currentSpeed \times \frac{1}{sampleFrequency_{gyro}} \times \sin(headingAngle)$$

The improved trajectory is shown below (Fig 5.2.3.2). Compared to the last trajectory (Fig 5.2.3.1), it is much smoother and preserves more detailed information of the

walking trajectory. This algorithm is served as the base of the data fusion process.

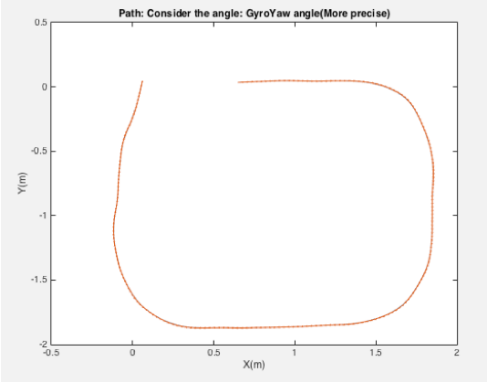


Figure 5.2.3.2: Trajectory estimation improved

5.3 Summary

By looking into the walking pattern and pedometer algorithm, we now understand more about how the sensors interact with the world.

Chapter 6

Data Fusion

6.1 Overview

First of all, to summarize the all the process before the data fusion: all the sensors data have converted into either position related or angle related information (as shown in the figure 6.1.1). Based on the concept introduced in the chapter 2, each sensor serves as an observer of the environment, they send in data at different frequency to update either position or heading information. For instance, in this thesis, the gyroscope and accelerometer runs at 30Hz, the magnetometer runs at 5Hz and finally, the GPS runs at only 0.5Hz. As the time increases, data from each sensor fuses in and update current location and heading as shown in the figure 6.1.2)(Gadeke, Schmid et al. 2012).

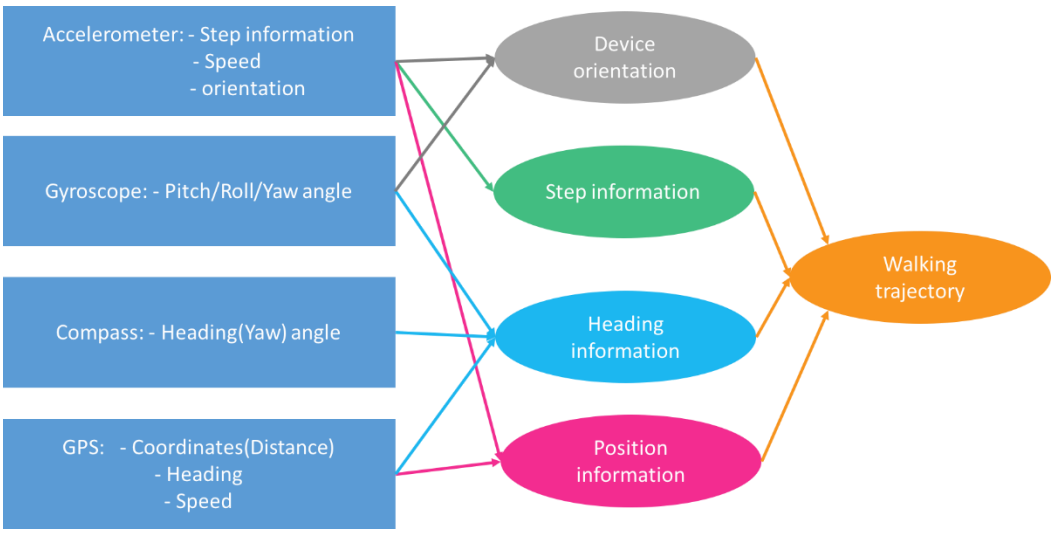


Figure 6.1.1: data fusion overview

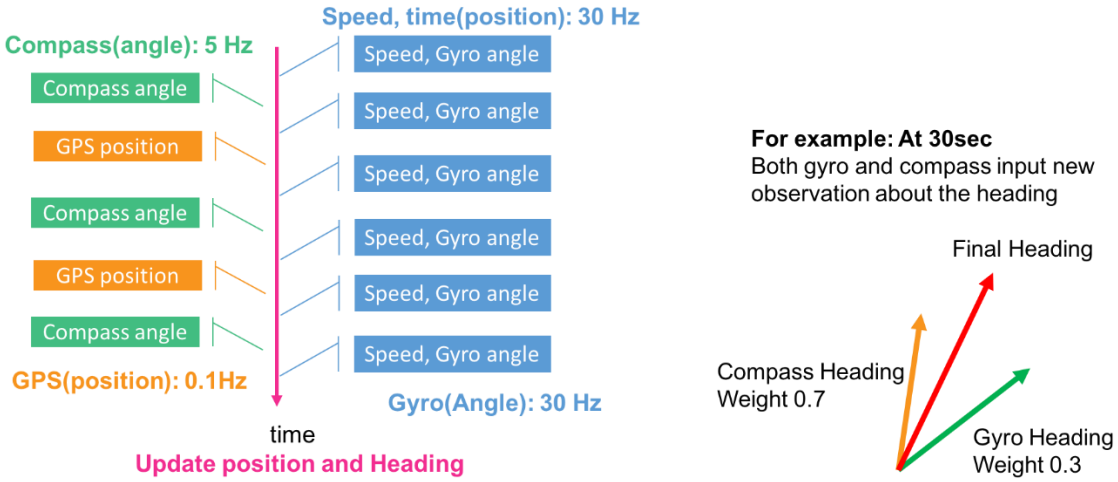


Figure 6.1.2: data fusion model
Figure 6.1.1: data fusion example

6.2 Data fusion model

Update heading information

In this section, the data fusion model will be discussed. As mentioned in chapter 5, the gyroscope heading and the speed information from accelerometer serves as the base of the data fusion to update both position and heading information. The model for update is:

Equation 6.2.1

$$\begin{bmatrix} X_{(k|k-1)} \\ Y_{(k|k-1)} \\ \theta_{(k|k-1)} \end{bmatrix} = \begin{bmatrix} X_{(k-1|k-1)} + \Delta t \times v \times \cos(\theta_{(k-1|k-1)}) \\ Y_{(k-1|k-1)} + \Delta t \times v \times \sin(\theta_{(k-1|k-1)}) \\ \theta_{(k-1|k-1)} + \Delta t \times \dot{\theta}_{(k)} \end{bmatrix}$$

Where $X,Y_{(k|k-1)}$: current X and Y position based on previous data
$\theta_{(k|k-1)}$: current heading angle based on previous data
$X,Y_{(k-1|k-1)}$: previous X and Y position
$\theta_{(k-1|k-1)}$: previous heading angle
$\dot{\theta}$:angular speed measured by gyroscope
Δt:1/sample frequency of gyroscope
v:current speed of the step

Next the compass updates the heading angle at 5Hz. More specifically, when the compass heading updates to a new value, this value fused in with current heading angle by multiplying with weight α_θ (between 0 and 1)as shown in the following formulas. Apparently, the output is effected by the weight α_θ. To obtain the optimised output, multiple experiments have conducted and the best α_θ for this program is 0.8.

Equation 6.2.2

$$\left[\theta_{(k|k)}\right] = (1-\alpha_\theta)\left[\theta_{(k|k-1)}\right] + \alpha_\theta\left[\theta_{observe(k)}\right]$$

Where $\theta_{(k|k)}$: current fused heading angle
$\theta_{(k|k)}$: current heading angle based on previous data
α_θ: compass fusing weight
$\theta_{observe(k)}$: current compass heading

The final step is to update the GPS location into the position data. Before look into the data fusion model, there are few important properties about the GPS that need to be aware of.

GPS accuracy

The unit of longitude and latitude (as shown in the following figure) is in degrees and minutes that represent any locations on earth. However, its accuracy is critical in this application since the position information is measured in meters. By looking at the following table, it is clear that all the calculation related to GPS coordinates has to reach at least 6 digits precision. In the program, all the coordinate related information has to be saved either as double or long.

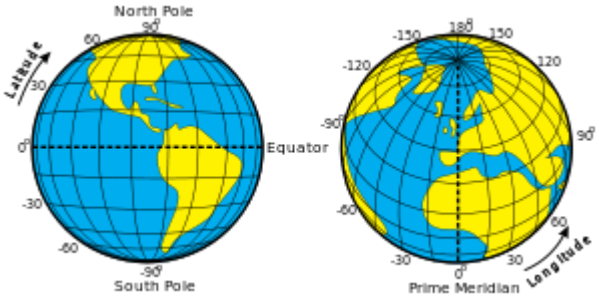


Figure 6.2.1: Earth model

Decimal places	degree	Distance
0	1.0	111 km
1	0.1	11.1 km
2	0.01	1.11 km
3	0.001	111 m
4	0.0001	11.1 m
5	0.00001	1.11 m
6	0.000001	0.111 m
7	0.0000001	1.11 cm

Table 6.2.1:GPS accuracy

Calculate destination point GPS coordinates giving distance and bearing from the start point

The position information from the mems sensors is measured in meters, to fused with GPS data, all the units has be converted into longitudes and latitudes. The formula (as shown below) of calculating destination point GPS coordinates giving the current increment distance and bearing from the start point. Later in the position fusion, this formula helps to convert the position to GPS coordinates.

Equation 6.2.3

$$\varphi_2 = \arcsin(\sin(\varphi_1)\cos(\delta) + \cos(\varphi_1)\sin(\delta)\cos(\theta))$$
$$\lambda_2 = \lambda_1 + \arctan(\frac{\sin(\theta)\sin(\delta)\cos(\varphi_1)}{\cos(\delta) - \sin(\varphi_1)\sin(\varphi_2)})$$

Where φ is latitude, λ is longitude, θ is the bearing (clockwise from north)
δ is the angular distance d/R; d being the distance travelled, R the earth's radius

For instance, the following data is collected on a football court of campus, by giving the start location and the increment bearing and distance, the output is the destination point coordinates[3] as shown in the following diagram.

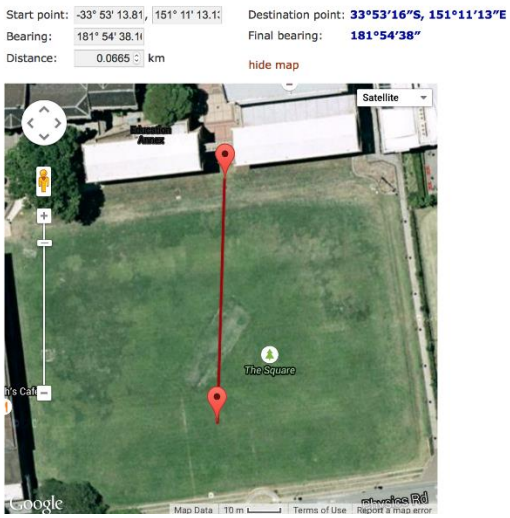


Figure 6.2.2: Position to GPS

Update position information

Finally, we can look into the position update model that combines mems sensors (accelerometer, gyroscope and compass) location with GPS coordinates. It is similar to the heading updates, where the output of the location is affected by fusion weight α_p. Same as the heading update, the optimized α_p value is 0.8. The position update model is:

Equation 6.2.4

$$\begin{bmatrix} X_{(k|k)} \\ Y_{(k|k)} \end{bmatrix} = (1-\alpha_p)\begin{bmatrix} X_{(k|k-1)} \\ Y_{(k|k-1)} \end{bmatrix} + \alpha_p \begin{bmatrix} X_{observe(k)} \\ Y_{observe(k)} \end{bmatrix}$$

Where X,Y(k|k): current fused location
X,Y(k|k): current position based on previous data
ap: position fusing weight
X,Yobserve(k): current GPS coordinates

[3] Test link: http://www.movable-type.co.uk/scripts/latlong.html

6.3 Data fusion result

The following figures demonstrate the data fusion results. In figure 6.3.1, the fused heading sits between the gyroscope heading and compass heading. In figure 6.3.2, the final trajectory of the fused data is labelled in red. It notable that the raw trajectory fused by all sensors' data becomes extremely noisy at the end phase, this is due to the fact that as time increases, the difference between compass heading and gyroscope heading becomes bigger and bigger which brings high frequency noise to the data fusion. To minimize the noise, a simple complementary filter is used again and that red line is actually the final filtered trajectory. Finally, in figure N3, all the position information are converted into GPS coordinates and projected to the google map.

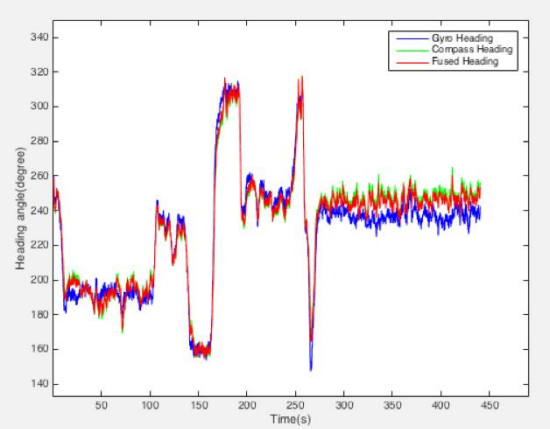


Figure 6.3.1 Heading fusion result

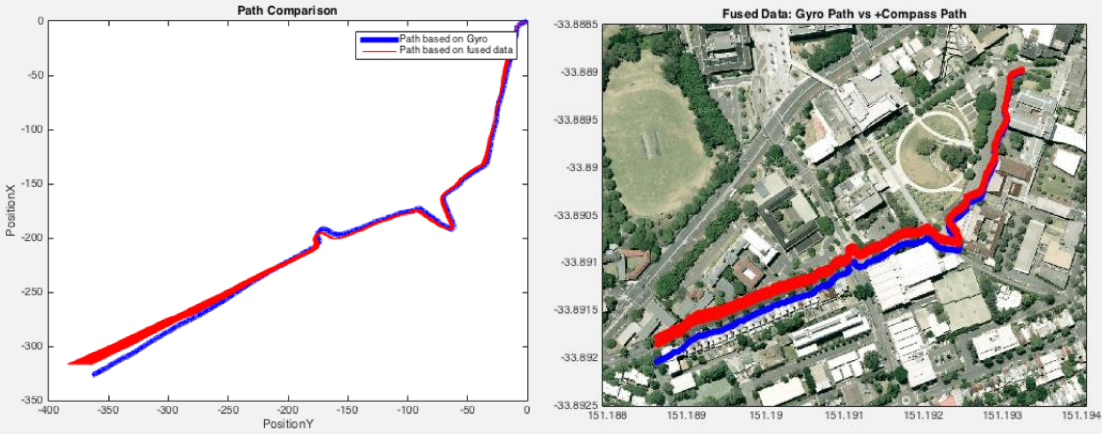


Figure 6.3.2:position fusion result: in meters
Figure 6.3.3:position fusion result: in GPS coordinates
Where Blue: path based on gyroscope
Red: path based on fused data

6.4 Summary

Overall, the initial aim has been reached, now the program is able to fuse different types of sensor and estimate a relatively accurate trajectory. However, the noise filtering of the data fusion is not idea, more filtering methods should be tested in the future to optimize the output.

Chapter 7

Conclusion

The purpose of this thesis is to look into all the related information and algorithm to develop a pedometer using the smartphone's sensors. Firstly, an application is developed for data collection purpose and then various methods of sensor calibration, filtering and conversion are discussed especially the calibration of the magnetometer. Next the pedometer algorithm has been developed to get information related to human walking pattern. Finally, all the sensors data are combined together using the data fusion model to gain the accurate walking trajectory on the map.

To conclude, the initial goal is reached and to myself, this thesis largely improves my problem solving skills and research ability. Most importantly, it changes my definition of mechatronics again and shows the potential of this degree.

6.2 Future Work

The algorithm from data filtering and conversion to step information analysis and data fusion should be able to re-planted from matlab to the smartphone platform (ios or andriod) using Qt.
Furthermore, more realistic experiment model should be applied. Instead of holding the phone in a leveled position, we should test the other walking patterns when the phone is inside the subject's pocket, tied to arm or even lays in the bag. Then compare the differences.

List of References

Alan, H. F., et al. (2014). Sensor Log: A mobile data collection and annotation application. Signal Processing and Communications Applications Conference (SIU), 2014 22nd.

Gadeke, T., et al. (2012). Smartphone pedestrian navigation by foot-IMU sensor fusion. Ubiquitous Positioning, Indoor Navigation, and Location Based Service (UPINLBS), 2012.

Georgy, J., et al. (2009). Quantitative comparison between Kalman filter and Particle filter for low cost INS/GPS integration. Mechatronics and its Applications, 2009. ISMA '09. 6th International Symposium on.

IvenSense (2011). "Motion sensor Introduction."

Rischpater, R. (2014). Application Development with Qt Creator - Second Edition, Packt Publishing.

STMicroelectronics (2010). "AN3192 Application Note."

STMicroelectronics (2010). "MEMS motion sensor: ultra stable 3 axis digital output gyroscope datasheet."

Wahdan, A., et al. (2015). "Three-Dimensional Magnetometer Calibration With Small Space Coverage for Pedestrians." Sensors Journal, IEEE **15**(1): 598-609.

Appendix

Please check the attached CD, all the appendix information are there. Including code and sample data.