AN ABSTRACT OF THE THESIS OF

Shengkai Zhang for the degree of Master of Science in Electrical and Computer Engineering presented on June 12, 2019.

Title: Exploring IMU Attitude and Position Estimation for Improved Location in Indoor Environments

Abstract approved: ______________________________

Huaping Liu

Wearable sensors with an inertial measurement unit (IMU) are popular for indoor positioning and activity pattern detection. The IMUs can be connected to a wireless transmission module, allowing users to monitor and process motion-related parameters remotely. Because of the complexity and uncertainty of signals in indoor environments, a radio frequency (RF) positioning system alone is often insufficient to provide the position accuracy and stability required for many applications, for example, position-guided indoor mobile robots. Our idea is to fuse two or more sources of data to generate highly accurate positioning information. Specifically, we have developed an IMU-aided RF positioning system, aiming to improve the accuracy of the system in indoor environments for mobile robotics. This approach combines the measurements of the accelerometers, gyroscopes, and magnetometer from an IMU via a complementary filter. The work includes a development of a calibration algorithm, which reduces the IMU drift and error. With the calibrated data, the trapezoidal integration method can now better use the accelerometer data to estimate the velocity and displacement of the mobile robot. In order to transform the data in the coordinate system of the IMU mounted on the mobile robot body to the ground positioning coordinate that RF positioning uses, we implement a quaternion rotation algorithm. This enables the fusion of the IMU and RF positioning estimates to accurately determine the moving trajectory of the mobile robot and guide its moving directions.

Exploring IMU Attitude and Position Estimation for Improved Location in Indoor Environments

by

Shengkai Zhang

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented June 12, 2019
Commencement June 2019

ACKNOWLEDGEMENTS

I would like to thank my advisor Prof. Huaping Liu for his excellent guidance, not only as an advisor but also a mentor of life. Without his plenty of time and energy he invested in guiding me, I cannot possibly have finished this thesis. I also want to thank all the committee members for their inspiring insights and suggestions. Finally, I want to thank my parents for their love, understanding and consistent support and encouragement.

TABLE OF CONTENTS

Page

1 Introduction 1

2 Background 3

2.1 IMU Location 3

2.1.1 How IMU Works 4

2.1.2 Accelerometer 5

2.1.3 Gyroscope 5

2.1.4 Magnetometer 6

2.2 Technical Challenges with IMU Positioning 7

3 Proposed Solutions to the Technical Challenges 9

3.1 IMU Assisted Position System Description 9

3.2 Calibration 12

3.2.1 IMU Offset 12

3.2.2 Result 15

3.3 Complementary Filter 17

3.3.1 Implementation 18

3.3.2 Result 21

3.3.3 Discussion 24

3.4 Position and Direction Calculation 24

3.4.1 Coordinate System Transformation 25

3.4.2 Quaternion 25

3.4.3 Acceleration to Displacement 28

3.4.4 Discussion 32

4 Experimental results 34

4.1 System Structure 34

4.2 Test Environment Setting 34

4.3 Hardware Components Used in Experiment 34

4.3.1 The IMU 34

4.3.2 Communications Module 38

4.4 Trajectory Test Result 40

4.5 Discussion 41

TABLE OF CONTENTS (Continued)

Page

5 Discussion and Conclusions 43

5.1 Discussion 43

5.2 Conclusions 45

Bibliography 45

Appendices 50

A Positioning Diagram 51

B Source Code 52

LIST OF FIGURES

Figure Page

3.1 Hardware development. 10

3.2 Software development. 11

3.3 The ideal situation of the magnetic field shape. 13

3.4 The ideal situation of hard iron distortion shape. 14

3.5 The ideal situation of soft iron distortion shape. 15

3.6 Calibration data with gyroscope. 16

3.7 Calibration data with gyroscope. 17

3.8 Calibration data with magnetometer. 18

3.9 Yaw, roll, and pitch 19

3.10 The effect of complementary filter coefficients. 21

3.11 Testing of yaw rotation. 22

3.12 Testing of roll rotation. 23

3.13 Testing of pitch rotation. 23

3.14 The relationship between ground coordinate system and IMU body coordinate system. 25

3.15 Gravity changed in rotation. 28

3.16 The general integral method and trapezoidal integral. 30

3.17 x-axis testing of the accelerometer. 31

3.18 y-axis testing of the accelerometer 32

4.1 Overall system structure. 35

4.2 Overall system structure. 36

4.3 The overview of the testing environment. 36

4.4 The setting of IMU on the mobile robot. 37

LIST OF FIGURES (Continued)

Figure Page

4.5 The IMU (BNO055). 37

4.6 System architecture of BNO055. 38

4.7 I^2C protocol. 39

4.8 The a circular trajectory of the IMU. 41

4.9 The straight-line trajectory of IMU. 42

LIST OF TABLES

Table Page

3.1 The Average Error of Rotation 24

4.1 The Pin Set of BNO055 40

LIST OF ALGORITHMS

Algorithm Page

LIST OF APPENDIX FIGURES

Figure | Page

A.1 Positioning Diagram 51

Chapter 1: Introduction

The technique of indoor navigation is universally applied, attracting researchers around the world. Still, there remains a considerable challenge to improve the performance of such systems.

The navigation system falls into two categories, indoor and outdoor navigation. The most common use of navigation system is about global position system (GPS) [5]. Yet, it does not work well on some occasions, such as underground, urban areas, tunnels, and indoor environments. The indoor navigation system aims to provides navigation within buildings, under the ground, and some places where the GPS malfunctions. That is where an indoor positioning system (IPS) comes in.

Indoor navigation [2–4, 29] relies either on beacon-based or beacon-free methods. The beacon-based system requires some infrastructure such as radio frequency (RF), for example, ultra-wideband signals [7–16], and infrared ray techniques. For the RF-based system, Wi-Fi signal is the most widely used type due to their extensive coverage. With known anchor points and transmitter point, trilateration is used to obtain position data based on the RF signal information such as time-of-arrival (TOA) [17, 19], or time-different-of-arrival (TDOA) [18]. Although these devices can provide accurate location information, due to the high cost of the device, deployment is delicate, readily affected by environments and rarely useful on a large scale.

As a contrast, beacon-free system is based on the inertial measurement unit (IMU). The IMU is an inertial sensor based on the latest micro electro mechanical systems technology(MEMS) [1]. It includes accelerometers, gyroscopes, and magnetometers. Yet, Magnetometers are sensitive to environmental factors, including environmental metals and magnetic field disturbances, making magnetometers not used alone in indoor navigation systems. In spite of that, the magnetometer can be combined with the accelerometer and gyroscope to produce a piece of stable positional information.

Furthermore, accelerometers measure acceleration in three dimensions. First, the gyroscope can measure the angular velocity of a three-dimensional space. The dead reckoning is based on fusing the acceleration and gyroscope and heading direction during time step to determine displacement and direction of the user has moved from the last known position. In this process, the influence gravity needs to be considered. By combining accelerometers, gyroscopes, and magnetometers, the effect of gravity on each axis can be eliminated. However, accelerometers are susceptible to motion and do not necessarily come from the motion of the target carrying the IMU. It is also possible to come from the vibration of the target. Overall, IMU can deliver accurate tracking performance within a relatively small displacement, and the errors add up as the sensors drift over time. Still, the accuracy can be enhanced up to a higher level with filter algorithms, decreasing the drift error.

In some indoor environments, due to the uncertainty of surroundings, it is challenging to offer accurate position information by relying solely on a separate navigation system. At this time, incorporating the data from different navigation systems and obtaining a useful precision position information through fusion algorithm is key. Both the RF-based and IMU-based systems have their strengths and weakness. The combined system can further improve the reliability and accuracy of the positioning system. In this thesis, we propose a combined indoor navigation and position (CINP) system for combining the advantages of both the RF and IMU system. Meanwhile, the combiner system using a microcontroller equipped with an inertial measurement unit (IMU) is used to collect data. Finally, data fusion is used to obtain more accurate locations.

Chapter 2: Background

2.1 IMU Location

Chapter 2 starts by introducing IMU from a broader perspective as well as the essential workings of IMU. To be specific, sensor refers to an electronic device that can be used in conjunction with other electronic devices to detect changes in some factors in certain environment, such as light, sound, temperature, humidity, magnetic field, wind speed, air pressure or motion. The sensor then passes this information back to the connected microprocessor, which processes and then produces valid data that ultimately matches the actual situation.

IMU is a particular type of sensor that measures angular velocity, force, and also measures magnetic fields. The general IMU consists of a three-axis accelerometer and a three-axis gyroscope, which can be called a six-axis IMU. Sometimes they can also include an additional 3-axis magnetometer, which will be called a 9-axis IMU. It is an abbreviation for inertial measurement components, which is usually adopted in conjunction with sensor fusion algorithms to combine data from multiple sensors to provide direction and heading and trajectory measurements. In this case, IMU can be a combination of sensor and sensor fusion algorithms, which is also called the attitude heading reference system (AHRS) in the field of attitude solving.

In the field of IMU development, low-precision IMUs are commonly used in GPS navigation systems, in monitoring vibration and direction in consumer electronics such as cell phones, wireless mouse and keyboard, game remotes, pedometers, or in augmented reality. The user's limb movement is detected in the augmented reality (AR) and virtual reality (VR) systems. IMU's motion and direction functions are also suitable for maintaining the balance of the drone in the air or balancing the balance of the car on the ground; IMU combines with the laser sensor to improve the direction of the sweeping robot, as well as other IoT and home connectivity appliances.

In addition, the medium-precision IMU is mainly used in industrial and automotive grade products, such as automotive electronic stability systems (ESP or ESC) [36] and GPS-assisted navigation systems. It can help locate antennas or other equipment, measure the vehicle attitudes, and control the robots and autonomous vehicle.

Furthermore, high-precision MEMS inertial sensors are taken as military and aerospace grade products such as communication satellite radio, aircraft flight control, attitude control, and the like.

For example, some companies have added IMU to the remote control of the entertainment system. In addition to the touch function, it also increases the accessibility of the system as a whole. This method is used by the LG Smart TV Remote Control [25], which allows the user to control the TV's user interface through intuitive clicks instead of the ubiquitous direction button controls.

Through these discussions, it figures out the possibility of IMU for future applications, and it will be tightly integrated with more sensors, such as radio frequency (RF), laser radar (light detection and ranging) technologies. These emerging technologies make it possible to locate people, vehicles, and equipment both indoors and outdoors accurately. This article is to discuss the possibility of IMU in indoor positioning centered on IMU.

2.1.1 How IMU Works

The IMU studied in this paper provides nine degrees of freedom, which refers to the number of different ways an object can move in three-dimensional space. The nine degrees of freedom include a 3-degree translational motion along each axis (front/rear, right/left, up/down) on a plane, and a 3-degree rotational motion on each of the x, y, and z-axes, and the 3-degree magnetic field strength of the x, y, and z-axes.

The raw data collected from the IMU provides some information about the world around it, and it can be processed for more messages. Sensor fusion is a (mathematical) art that combines the data of each sensor in an IMU to create a more complete device orientation and pattern. Take an instance, when viewing the motion information of the rotation and acceleration of the gyroscope, the gravitational component of the accelerometer can be removed to create an inertial reference frame. Information about the Earth's magnetic field can also be added to align the entire sensor with the Earth's ground reference frame.

2.1.2 Accelerometer

The most commonly used motion sensor is the accelerometer. The acceleration measured by the accelerometer is the rate of change of the speed of a device relative to time. Its unit is (m/s^2), and it measures the change in acceleration on one axis, just like stepping on a brake or suddenly dropping a phone. The accelerometer sensor follows the sensor's inertial coordinate system. Put it another way, when the device is in free-fall motion, the acceleration in the descending direction is $0m/s^2$. When a device is lying flat on a table, the acceleration in the upward direction is equal to the Earth gravity, i.e. $g = 9.8m/s^2$, and it measures the acceleration of the device pushing up.

The accelerometer itself is of little use because the noise generated by the raw data containing gravity can have an indeterminate effect on each axis. Therefore, accelerometers often participate in the work of other sensors. Still, the use of accelerometers alone does have some merits, such as recording vibration.

Linear acceleration is of attention as well, which refers to the acceleration applied to the device where the sensor is located. Without gravity, gravity removal is called gravity compensation.

For accelerometers, we usually care about the value of the more significant change in acceleration. To avoid the effects of noise, such as gravity, some filters are usually needed to process the data. For example, high-pass filters are often adopted to help isolate linear acceleration, and low-pass filters are used to isolate gravity. Saying that though, this approach introduces a certain amount of delay into the data, resulting in a certain amount of offset in the data output. Afterwards, we will gradually discuss how to remove the effects of gravity.

2.1.3 Gyroscope

Accelerometers measure acceleration but cannot evaluate torsional or rotational motion. However, the gyroscope can sense the angular velocity, relative to itself; so, it measures its rotation, using the inertial force called the Coriolis effect. The gyroscope can measure the angular acceleration of three axes: pitch (x-axis), roll (y-axis), and yaw (z-axis). The gyroscope oscillates at a relatively high frequency to measure angular velocity changes during motion and is, therefore, one of the most power-hungry motion sensors. This also means that they are susceptible to other vibrations, such as vibrating motors or speakers on the same device.

When the gyroscope is integrated with other sensors, the gyroscope can be used to determine the orientation of the object in three dimensions. Although the gyroscope does not have an initial frame of reference (such as an inertial gravity frame of reference), it can combine its own data with the accelerometer data to measure the angular position. In order to obtain the rotation (angle) from the gyroscope, that is, to process the data of the gyroscope, it is necessary to perform one integration on the original data.

$$\int cos\left(2\pi\right) = \left(\frac{1}{2\pi * f}\right) * sin\left(2\pi * ft\right). \tag{2.1}$$

However, in the process of integration, the noise will also be integrated into a drift. As shown in the above formula, the integral gets $1/f$ outside, which means that the high frequency (f) noise will disappear with the integration, i.e. the frequency noise will drop by 100 times. However, the shallow frequency will be amplified, suggesting that the gyroscope will drift over time. Put it another way, if only a gyroscope is used, the data it outputs will drift more and more over time. Some drift compensation calibrates most gyro sensors on the hardware known low frequencies caused by adjacent hardware on the device. For example, the gyroscope previously introduced is combined with an accelerometer for calibration.

2.1.4 Magnetometer

Magnetometers, as the name suggests, measure magnetic fields. It can detect the fluctuation of the Earth's magnetic field by measuring the magnetic flux density at the point of the space where the sensor is located in the air. This indicates that if there is no strong magnetic field nearby, it can sense the Earth's magnetic field through these fluctuations. The Earth's magnetic field is, more or less, pointing to the north, but not the exact north. This can be combined with accelerometer and gyroscope data for identifying absolute heading.

As just mentioned, the magnetometer is very sensitive to the surrounding environment, just like anything on the table that is slightly magnetized. It can even be affected by other things inside the device, although the device manufacturer can make compensation. However, in practice, these sensors work very well in most natural environments. As long as there is nothing magnetized around it, the magnetometer reading is stable enough to isolate the gravity mentioned in subsection of Accelerometer.

The magnetometer is a three-axis sensor, which means it gives a three-dimensional vector pointing to the most active magnetic field. In the meantime, it suggests that it does not force a specific direction to work. However, to know how the device is being held, a gravity vector is needed. In the case of filtering, the gravity vector requires at least one accelerometer. If a more accurate reading is needed, then a gyroscope is required. This is called tilt compensation. The most common example is the use of a magnetometer sensor as part of the IMU data fusion to generate a geostationary-based directional sensor, or a compass, which can be directed to the true north by modulating the deviation with other sensors.

As has been mentioned, the IMU is an inertial sensor that integrates an accelerometer, a gyroscope, and a magnetometer. The IMU is used to measure acceleration, angular velocity, and magnetic field. When combined with sensor fusion algorithms, they can be used to determine motion, direction, and heading. Below we will continue to discuss how to handle the data, and data integration.

2.2 Technical Challenges with IMU Positioning

When the IMU appears as an assisted positioning system in an indoor environment, there are three technical challenges to be addressed.

a. The sensor drift problems. Given that the consumer-level IMU itself has a certain degree of system drift, the accelerometer and gyroscope data are not zero and drift with time when the IMU is stationary.

b. Reduce unnecessary error accumulation. Because the IMU data is relatively independent, the IMU cannot identify the error and the actual valid data during the IMU movement. As a result, some invalid data appears in the positioning system.

c. IMU assisted positioning systems to produce stable data (e.g., attitude angle, positioning information), and then combine with other indoor positioning systems. In other words, the IMU system can generate independent and stable positioning trajectory data.

Chapter 3: Proposed Solutions to the Technical Challenges

3.1 IMU Assisted Position System Description

As described in the introduction, it is inefficient for the indoor environments to solely rely on GPS as the primary navigation tool in that GPS signal inside buildings is too weak to penetrate the building structures. In the indoor environments, there are multiple ways to determine the location of a target object , although each of them are not technically perfect. It generally falls into two categories including absolute and relative position. To determine the absolute position, the observer is required to provide observed position data, such as a radio frequency signal, which provides an absolute rough position. However, for the complexity of indoor positioning environments, RF signals are often interfered by uncertain factors. This will affect the outcome of the final positioning. The relative position is suitable for indoor positioning and is less expensive. To measure the relative position, it is only necessary to detect the change in position of the target object from the starting point.

One possible solution for relative position detection of indoor positioning is based on the IMU. The IMU, as an auxiliary indoor positioning system, improves the accuracy of the indoor positioning system. The IMU is an independent, low-power, the highly miniaturized sensor that collects data from internal motion without any external reference. By using an IMU system with accelerometers, gyroscopes, and magnetometers, the relative position and orientation of objects in all three dimensions can be measured. By combining these data with the RF positioning system, a relatively stable target location is obtained. This paper, in the next few sections, will mainly discuss the feasibility of IMU as an auxiliary positioning system in indoor environments.

a. Calibrating the sensor is a critical step in the IMU's startup. It can reduce the initial sensor drift. Based on this, this thesis proposes a 6-side calibration method, rotate every axis to $+z$-axis(gravity) direction to figure out the offset, in this case, calibration can reducing the error of the IMU itself.

b. Detecting motion is key to reducing unnecessary errors during the IMU's motion. By detecting the motion state of the IMU, it can effectively reduce the error caused by the self-jitter of the IMU, the IMU movement path bump, and the IMU instantaneous inertia start and stop.

c. The stable data output of the IMU is important to the IMU's ability to be an assisted positioning system. Using the appropriate integration method to calculate the relative displacement and direction of the object, the IMU can generate a referenced and accurate movement trajectory, thereby achieving the purpose of assisting other positioning systems (RF).

The development of the IMU assistant positioning system includes hardware development and software algorithm.

Hardware development consists of inertial measurement unit (IMU), micro-controller of Arduino UNO and Raspberry Pi, as shown in figure 3.1

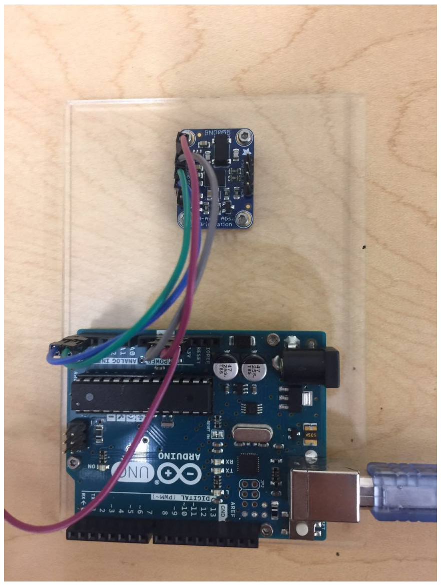

Figure 3.1: Hardware development.

Arduino reads the real-time data of BNO055 through I^2C bus, which is transmitted to Raspberry Pi through a serial port. Finally, Raspberry Pi transmitted the IMU data to the server through TCP wireless network. This hardware configuration provides nine degrees of freedom of real-time data to calculate the position and trajectory of the current object in two-dimensional space.

IMU software development consists of coordinate system conversion module and positioning algorithm module. The coordinate transformation module USES, the direction rotation matrix algorithm and sensor compensation algorithm work together to generate original stable sensor data. The localisation algorithm module includes the object motion detection algorithm and integral displacement algorithm, which can help to determine the moving trajectory of the object. The development of IMU software needs to consider gyro drift correction, accelerometer gravity elimination and yaw Angle magnetic course correction.

The IMU auxiliary positioning system module calculates the current position and direction based on the accelerometer, gyroscope, and magnetometer. First, the linear displacement is calculated by using a complementary filter to remove the acceleration data affected by gravity. Moreover, integrating the acceleration by trapezoid method to compute the velocity and the displacement. The accelerometer, gyroscope, and magnetometer data are combined with a quaternion algorithm to calculate the course of the current position. Finally, the trajectory of the object is calculated by using the Kalman Filter. Figure 3-2 shows the block diagram of the soft development module.

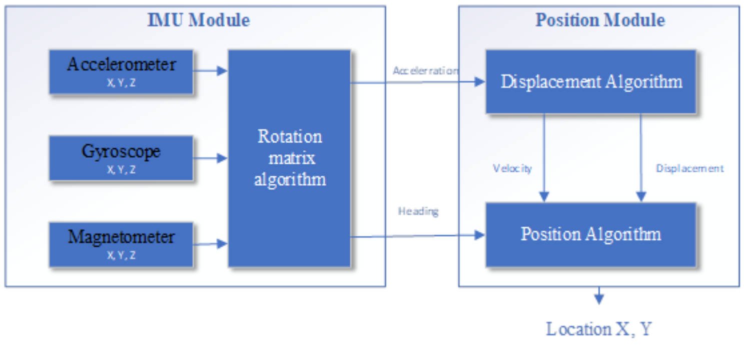

Figure 3.2: Software development.

The next section will describe these solutions one by one.

3.2 Calibration

This subsection will explain what an accelerometer, gyroscope, and magnetometer drafts and errors is. Also, how to set the offset to reduce these kinds of errors will be included as well.

MEMS IMU devices are microdevices, which are easily affected by external factors (as described in the Background). Because of the system drift of the devices, they are inaccurate, which tend to be the feature of inertial sensors. This characteristic means that even when the device is stable, the accelerometer, gyroscope, and magnetometer will still output data other than zero. Similarly, each sensor has its reasons for such drifts or errors. In this case, therefore, different offset methods and correction methods are designed according to the different reasons for their errors, which are named sensor calibration.

3.2.1 IMU Offset

Accelerometer There are two causes explaining the accelerometer's measuring a three-axis physical acceleration with a large offset. First, the acceleration itself is very sensitive, while the other is that the Earth has a gravitational field. Gravity is known ($9.81m/s^2$) when the object is stationary and horizontal, and the direction of gravity is the same as the z-axis of the accelerometer. So, the effect of gravity on the component of gravity is easily removed. However, when the object moves, since the car does not stay horizontal all the time, this component of gravity will affect the x, y, z three axes at the same time. So, the accelerometer does not measure linear acceleration. Rather, it is a composite acceleration. The following formula can express this combined acceleration.

$$A(t) = A_l - g + B_a(t) + N_a(t), \tag{3.1}$$

where $A(t)$ represents the composite acceleration, A_l represents the linear acceleration, $B_a(t)$ is the accelerometer bias, $N_a(t)$ is the noise.

According to (3.1), it is easily found that if a stable linear acceleration is to be obtained, the sensor data must minus the gravity component on the three-axis through the attitude of the sensor. On the other hand, the velocity and displacement of the object will become more significant with the integration of the acceleration data. Due to the integration of the fundamental error, the speed and displacement of each update of the object will be proportional to the time of the initial position. This error is the well-known integral drift problem of the inertial navigation system. The average data output of the sensor is read 500 times by the 6-sided calibration method, thereby the accelerometer's offset can be obtained.

Magnetometer The offset calibration of the magnetometer guarantees the estimated performance of the sensor heading. On the flip side, unlike accelerometers and gyroscopes, magnetometers need to sense the Earth's magnetic field strength to determine the direction of motion of the object. When measuring the Earth's magnetic field strength, the central disturbances can be divided into two groups, hard iron, and soft iron distortion [24].

Assuming that the magnetic field, which is not affected by the distortion of hard and soft iron, is a circle which is centred in the axis. See figure 3.3.

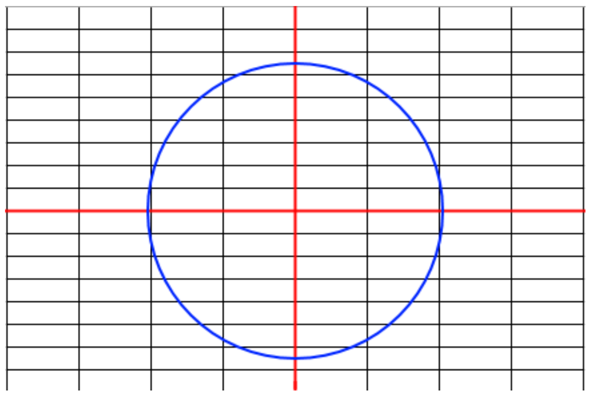

Figure 3.3: The ideal situation of the magnetic field shape.

Hard iron distortion Objects that generate magnetic fields produce hard iron distortion. To be specific, hard iron distortion is easy to calibrate as they are constant regardless of the orientation and position of the device. A simple way to determine this offset is to slowly rotate the sensor in three dimensions and then read the average of the maximum and minimum values for each axis of the 500 data of the magnetometer. The new value in this is the offset of the hard iron deformation of the magnetometer.

It should be noted that hard iron distortion only results in permanent deviations in the output. Put simply, the hard iron distortion will only cause the center of the magnetic field to deviate from the origin. See figure 3.4.

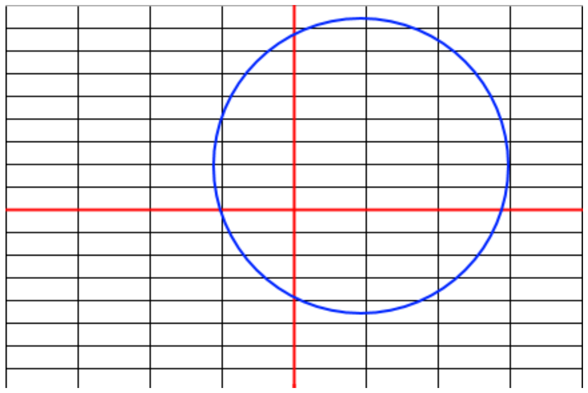

Figure 3.4: The ideal situation of hard iron distortion shape.

Soft iron distortion Compared with the effect of hard iron distortion, the effect of soft iron distortion is difficult to detect and correct. Since it is believed to be the deviation and variation in the existing magnetic field. Usually, some metals will either strengthen or block the magnetic field, and they will change with the direction of change. However, the geomagnetic field will not change, causing different magnetic field deviations in diverse positions. So, this change stretches or distorts the geomagnetic field itself. See figure 3.5.

The method of measuring the soft iron error is to rotate the magnetometer 36 times in a two-dimensional space, 10°each time, and record the magnetometer output data. In the case of soft iron distortion, the magnetic field scales of the x, y, and z-axes are different, and some direction produces a stronger or weaker magnetic field. This makes the distribution of the magnetic field appear an elliptical shape. By calibrating the effects of hard iron distortion, elliptical dots can return to the origin of the coordinates. The next step is to calculate the difference in the ratio between the ellipse and the perfect circle, that is figure out the scaling of each magnetic axis relative to the coordinate axis. Finally, the raw data multiply the scale factor and rotation matrix to obtain the distribution of the magnetic field, which approximates the circle.

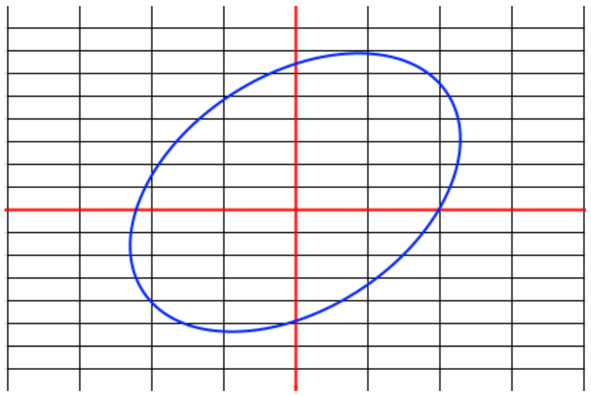

Figure 3.5: The ideal situation of soft iron distortion shape.

Gyroscope The gyroscope compensation is the simplest among the three calibration methods, and this can be accounted for by the fact that it does not need to calibrate the scale error but to calibrate the deviation of the zero motion. Putting the device on the stable horizontal planet and make sure the device does not move. Under such circumstances, the gyroscope's data output should be the zero angular velocity. If the raw data does have a non-zero component, it is the offset of the gyroscope. The correct offset will be calculated from the average of 500 outputs raw gyroscope data. Also, subtract these non-zero values from the raw data whenever the data are collected. Eventually, more efficient data is generated corresponding to the gyroscope motion.

3.2.2 Result

Gyroscope The result of the gyroscope offset calibration is shown in Figure 3.6. The blue waveforms in the Figure 3.6 give un-calibrated values, and these values' center is not zero. The red waveform in the Figure 3.6 indicates the calibrated value, shows the data near the zero value, and the calibration is complete.

Accelerometer The result of accelerometer offset calibration is shown in Figure 3.7. The blue waveforms in the Figure 3.7 give un-calibrated values. The red waveform in the Figure 3.7 gives the calibrated value, as same as the gyroscope. The Accelerometer calibration is complete.

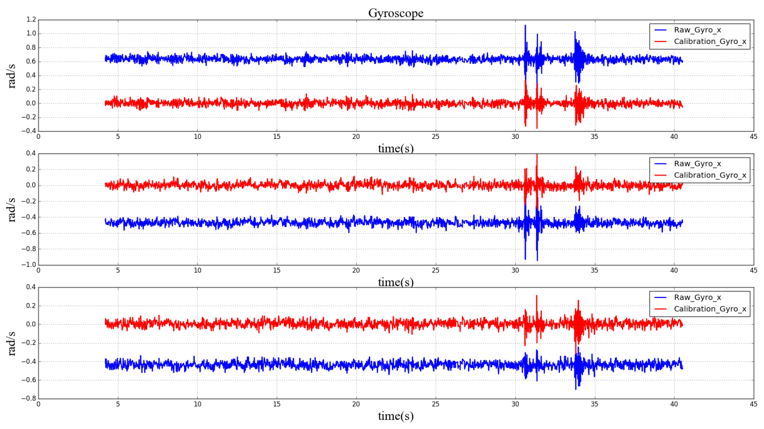

Figure 3.6: Calibration data with gyroscope.

Magnetometer As described in the method of calibrating the magnetometer, the original magnetometer data, when the device rotates 360°, is shown in Figures 3.8(1) and 3.8(2). It can be seen that the circle does not rotate around the center of the plane because of the error of hard iron distortion. It can also be seen that these circles are slightly elliptical, and this can be attributed to the error of soft iron distortion. The calibrated data is shown in Figure 3.8(3). Figure 3.8(4) shows the magnetic field distribution of the magnetometer under ideal conditions.

Because the experimental environment removes metals which might be produced soft iron, the effects of soft iron distortion in the Figure 3.8(1) and Figure 3.8(2) are not apparent. However, reports suggest that the soft iron distortion still affects the magnetometer in the environment. It is worth notice that the magnetic field influence is the difference in the real application environment. Moreover, some objects, which produce soft iron distortion, cannot be removed. Therefore, the magnetometer must be recalibrated whenever the experimental environment changes, and magnetometer calibration is complete.

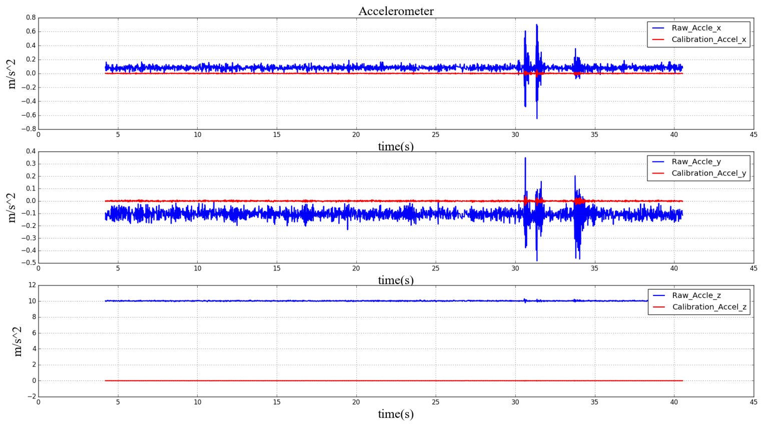

Figure 3.7: Calibration data with gyroscope.

3.3 Complementary Filter

In order to calculate the direction and position of the mobile robot, the IMU data must be combined. In order to make combined data more effective, some filters are commonly used, such as Kalman filter [22] and COMPLEMENTARY filter [27].

The Kalman filter is useful but also very complex [26] in the meanwhile. There are two challenges: First, it is difficult to determine the noise parameters; Second, it is tricky to realize some microcontrollers.

As the noise in indoor environments is complex and changeable, these noise parameters and Kalman filter need much calculation, which will slow down the operation of the microcontroller and thus causing an inevitable delay to real-time data. Much computing is not easy to do on microprocessors. Compared with the Kalman Filter, the complementary filter is simpler to implement and does not need to calculate a large number of noise parameters. Moreover, the complementary filter gives the same result [31].

The value of the gyroscope will drift over time, and that makes it a short-term sensor. The accelerometer data is only reliable for the long-term ones. The complementary filter gives the best of both worlds. In the short term, the data of the gyroscope is used because it is very accurate and is not affected by external forces. In the long term, as a contrast, the data from acceleration is mainly adopted for it does not drift. Put simply, the complementary filter filters out short-term fluctuations through a low-pass filter,

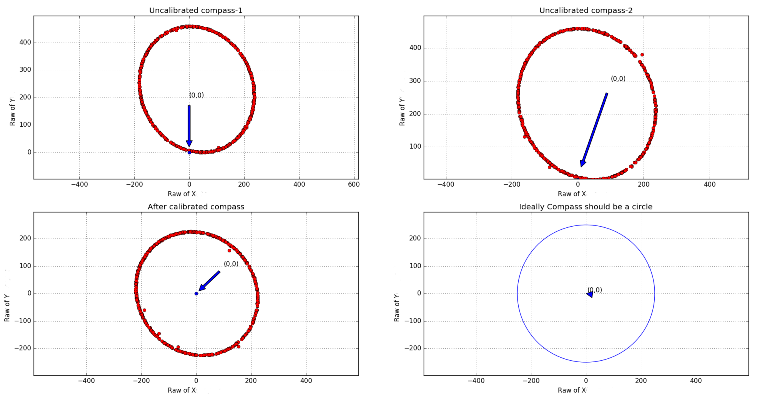

Figure 3.8: Calibration data with magnetometer.

allowing long-term changes to be preserved, i.e., averaging over time. The integrator obtains the relative direction of the sensor by integrating the angle. The high-pass filter filters out long-term changes and keeps transient changes that can cause the gyroscope to filter out drift.

3.3.1 Implementation

Roll, pitch, and yaw can determine the direction of the object. The distribution of the three angles is shown in Figure 3.9. [23]Yaw indicates that the object rotates around the z-axis, pitch indicates that the object rotates around the y-axis, and roll indicates that it rotates around the x-axis. Roll, pitch, and yaw are also direct output values of the gyroscope.

The filter formula is shown below.

$$\begin{aligned} Angle\left(t\right) =& K * \left(Angle\left(t-1\right) + GyroscopeData\left(t-1\right) * dt\right) \\ &+ \left(1-K\right) * accAngle\left(t-1\right). \end{aligned} \tag{3.2}$$

Angle represents the relative angle of the current object, K represents the filter coefficient of the complementary filter between 0 and 1, GyroscopeData represents the

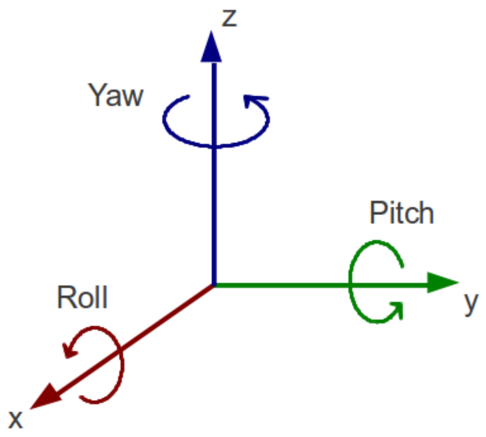

Figure 3.9: Yaw, roll, and pitch

gyroscope data, dt represents the sampling time between two iterations, and accAngle represents the angle value after converting the current acceleration. A trigonometric function can represent the angle of the current accelerometer. Because the acceleration is calculated in radians, therefore, it must be converted to an angle, the process of which is shown as following.

$$accRoll = arctan\left(\frac{ay}{\sqrt{ax^2 + az^2}}\right) * \frac{180}{pi}. \tag{3.3}$$

$$accPitch = arctan\left(\frac{-ax}{\sqrt{ay^2 + az^2}}\right) * \frac{180}{pi}. \tag{3.4}$$

The Roll and Pitch of the gyroscope after complementary filtering can be calculated according to the following formula.

$$\begin{aligned} Roll(t) =& K * (Roll(t-1) + Gyroscope_x(t-1) \\ & * dt) + (1-K) * accRoll(t-1). \end{aligned} \tag{3.5}$$

$$\begin{aligned} Pitch(t) =& K * (Pitch(t-1) + Gyroscope_y(t-1) \\ & * dt) + (1-K) * accPitch(t-1). \end{aligned} \tag{3.6}$$

Because the z-axis of the accelerometer measures gravity, which remains constant, the yaw axis cannot be compensated by the accelerometer. A magnetometer measures the geomagnetic field perpendicular to gravity and thus determines the rotation in the horizontal plane. So yaw can be compensated with a magnetometer[10]. The formula is shown below.

$$a = -y_{mag} * \cos(Roll) + z_{mag} * \sin(Roll), \tag{3.7}$$

$$b = x_{mag} * \cos(Pitch), \tag{3.8}$$

$$c = y_{mag} * \sin(Pitch) * \sin(Roll), \tag{3.9}$$

$$d = z_{mag} * \sin(Pitch) * \cos(Roll), \tag{3.10}$$

$$yawMag = arctan(\frac{a}{b+c+d}) * \frac{180}{pi}. \tag{3.11}$$

The Yaw of the gyroscope after complementary filtering can be calculated according to the following formula.

$$\begin{aligned} Yaw(t) =& K * (Yaw(t-1) + Gyroscope_z(t-1) * dt) + \\ & (1-K) * yawMag(t-1). \end{aligned} \tag{3.12}$$

The following formula determines the coefficients of the complementary filter.

$$K = \frac{tau}{tau + dt}. \tag{3.13}$$

tau is the desired time constant how fast the IMU reading to response, dt is the sampling time, which is the same with the complementary filter. However, the specific K value still needs to be selected in the experiment. In the project, K was set as 0.02. This is based on the results of different K values.

3.3.2 Result

According to the selection of the complementary filter coefficients, the IMU is placed at the horizontal table for 42 seconds to observe the drift of the gyroscope. As shown in Figure 3.10, the gyroscope drift compensates for different K values. It can be found that when the K value becomes small and K-1 becomes large, the drift of the gyroscope can be effectively reduced.

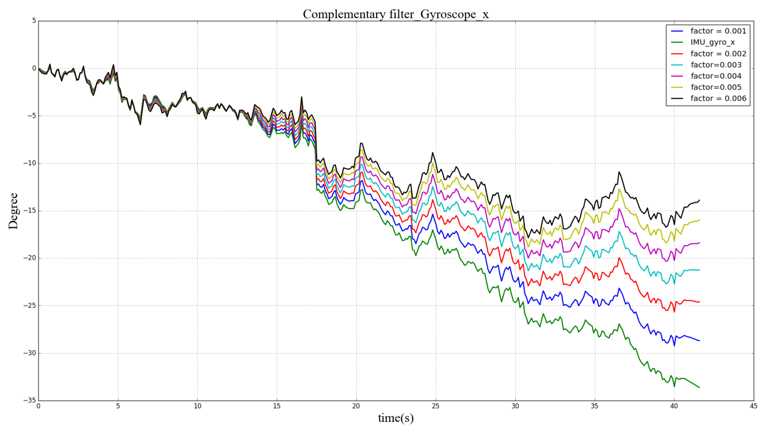

Figure 3.10: The effect of complementary filter coefficients.

The function in the code implements the complementary filter according to the formula in this section. Firstly, the calibration is carried out to ensure that the data of three axes of the gyroscope is approximately 0. The experimental rotation direction is shown in Figure 3.9. Because the chip is not completely aligned with the gravity axis, there is some drift in the rotation of the IMU. The first thing is rotating the YAW axis. This project sets the rotation range of yaw as 0°-360° because it can better match the direction of the mobile robot in certain context. As mentioned above, yaw is compensated by complementary filtering of the magnetometer so that the initial value will indicate the direction of an absolute value of the Earth's magnetic field. At the beginning of the test, ensure that the output Angle of the yaw axis is 0, rotate 90° every 30 sampling points, and the sampling time is 0.01s. See figure 3-11.

The rotation of the yaw axis is shown by sampling points 900 to 1010. The IMU was initially stationary at about 270°, then rotated counterclockwise 90° for sample interval

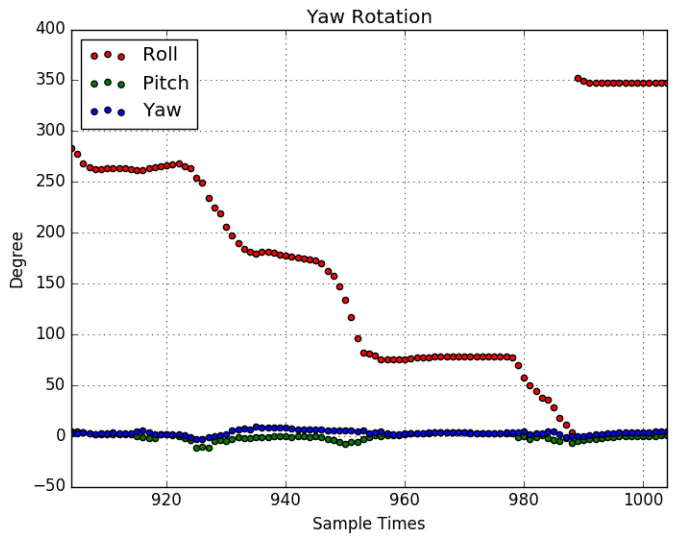

Figure 3.11: Testing of yaw rotation.

930 to 950, and then continued to rotate counterclockwise90° to 90°for sample interval 950 to 980. Finally, rotate 90° to 360°, sample interval 970 to 1010.

The second one is rotating the Roll axis. Remove the Yaw display for a more intuitive view of Roll's Angle change, as shown in figure 3.12.

As with the Yaw rotation test, it is time to ensure that the initial Roll axis output is 0, and select sample points 650 to 760 for display. According to the actual situation of the roll axis, the rotation range is from −180° to +180°.Rotate 90° for every 15 sample points.

The third point is to rotate the pitch axis. Remove the Yaw and Roll display for a more intuitive view of Roll's Angle change, as shown in Figure 3.13.

As with the Roll rotation test, ensure the initial Pitch axis output is 0, and select sample points 980 to 1080 for display. According to the actual situation of the roll axis, the rotation range is from −90° to+90°. Rotate 90° for every 20 sample points.

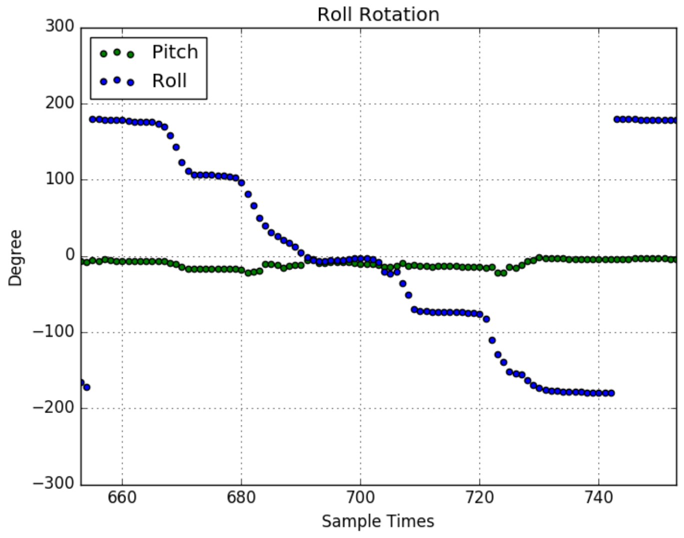

Figure 3.12: Testing of roll rotation.

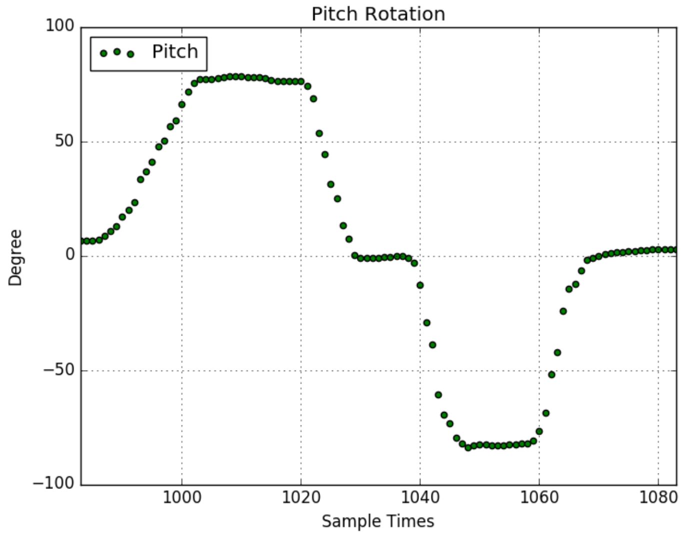

Figure 3.13: Testing of pitch rotation.

Table 3.1: The Average Error of Rotation

Axis Name	The Error of Rotate from 0° to 360°
Yaw	5° - 20°
Roll	2° - 6°
Pitch	5° - 10°

3.3.3 Discussion

According to the experimental results of rotation, the average error is calculated, as shown in table 2. It can be seen from table 2 that the complementary filter has a good compensation effect for Roll and Pitch, because the accelerometer compensation is stable. However, the Yaw axis has a large error in some directions, which is caused by the uneven distribution of the magnetic field, and certain electromagnetic interference exists in some angles of the indoor environments. Therefore, each time the complementary filter is used to correct Yaw, multiple magnetic field calibration must be performed on the test site in order to reduce the interference of the current environment to IMU.

3.4 Position and Direction Calculation

In the lead up section, the attitude of IMU was known by calculating Roll, Pitch, and Yaw. However, this kind of attitude is only for IMU's body coordinate system. In order to obtain the accurate attitude and displacement of the mobile robot, the reference coordinate system needs to convert from the body coordinate system to the ground coordinate system. The transformation of the coordinate system requires the model of a quaternion rotation matrix. Moreover, the quaternion rotation matrix can remove the influence of gravity on the accelerometer and calculate the linear acceleration. With this linear accelerometer and quaternion, the displacement and direction of mobile robots can be calculated. Eventually, the displacement and direction data are sent to the server and displayed. This section mainly introduces the quaternion rotation matrix and the displacement calculate algorithm.

3.4.1 Coordinate System Transformation

IMU is created in their body spaces, and that means the IMU output their data based on the Body Coordinate System. This is in that the indoor positioning system is built based on the ground coordinate system. Through the direction and displacement relative to the ground, the robot can be accurately positioned. So that is why the IMU Body coordinate system needs to convert to the Ground Coordinate, shown as the Figure 3.14.

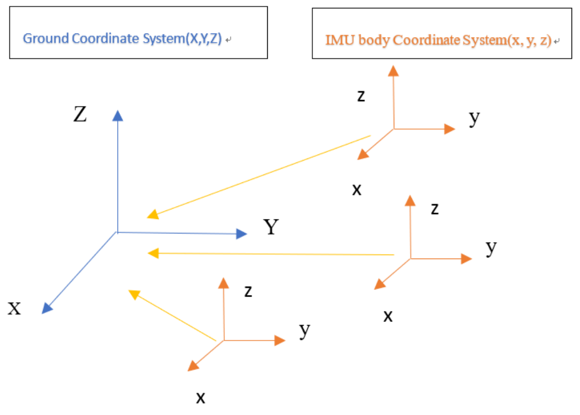
Ground Coordinate System(X,Y,Z)
IMU body Coordinate System(x, y, z)
Z
Y
X
z
y
x
z
y
x
z
y
x

Figure 3.14: The relationship between ground coordinate system and IMU body coordinate system.

The blue axes represent the ground coordinate system, the three orange axes represent the different positions of the mobile robot, and the yellow arrows represent the position of the robot relative to the ground coordinate system. That is why the rotation representation is important.

3.4.2 Quaternion

The quaternion [21, 33, 39] is the number system that extends the complex numbers. The Irish mathematician William Rowan Hamilton is the first person who described in 1843 and applied to mechanics in three-dimensional space. The quaternion provides a measurement that is unaffected by the gimbal lock. A quaternion is a quaternion vector that can be used to encode any rotation in a coordinate system. Its form is similar to a complex number. The quaternion axes i, j, and k are defined with the following relationship:

$$i^2 = j^2 = k^2 = ijk = -1. \tag{3.14}$$

The quaternion q is denoted as

$$q = w_0 + xi + yj + zk. \tag{3.15}$$

The q from BNO055 is defined as the unit vector from the body frame of the sensor to the ground frame, where w is the rotation around its axis in radians relative to the reference frame, and x, y, z are familiar coordinates used to indicate the three-dimensional vector. Rotation (w) is important for tracking the direction of the object. In most mechanical models, the rotation of the vector is tracked by the amount of rotation around the reference frame axis, so x rotation is the rotation of the vector around the x-axis, y rotation is the rotation of the vector around the y-axis, and z-rotation is the vector around the z-axis Rotate. Three rotations around the axis can then give a complete three-dimensional rotation. If using roll, pitch, and yaw to calculate rotation directly, there will be Gimbal-Lock problems [38]. The quaternion does not have this problem. This makes the quaternion very suitable for the calculation of this project because it will eliminate unnecessary rotation errors and problems. Moreover, it can help mobile robots track their movement direction under different movement conditions.

Then, there are the basic quaternion operations needed to figure out the direction. First, construct an expression that quaternions can use from the angles which is calculated roll, pitch, yaw by the last section.

$$w_0 = \cos(\frac{pitch}{2})\cos(\frac{yaw}{2})\cos(\frac{roll}{2}) + \sin(\frac{pitch}{2})\sin(\frac{yaw}{2})\sin(\frac{roll}{2}), \tag{3.16}$$

$$x = \sin(\frac{pitch}{2})\cos(\frac{yaw}{2})\cos(\frac{roll}{2}) - \cos(\frac{pitch}{2})\sin(\frac{yaw}{2})\sin(\frac{roll}{2}), \tag{3.17}$$

$$y = \cos(\frac{pitch}{2}) sin(\frac{yaw}{2})\cos(\frac{roll}{2}) + \sin(\frac{pitch}{2})\cos(\frac{yaw}{2})\sin(\frac{roll}{2}), \tag{3.18}$$

$$z = \cos(\frac{pitch}{2})\cos(\frac{yaw}{2})\sin(\frac{roll}{2}) - \sin(\frac{pitch}{2})\sin(\frac{yaw}{2})\cos(\frac{roll}{2}). \tag{3.19}$$

moreover, if the vector$v_b = (v_x, v_y, v_z)^T$ is defined as the axis of the rotation of the sensor, using the quaternion rotation the vector to the ground frame can be expressed as the following relationship.

$$v_i = q * \begin{pmatrix} 0 \\ v_b \end{pmatrix} * q^{-1}. \tag{3.20}$$

Wherein v_i is the output vector after quaternion transformation. q^{-1} is the conjugate of quaternion q.

$$q^{-1} = [w, -x, -y, -z]\,. \tag{3.21}$$

Based on this equation, IMU can obtain sensor data relative to the ground coordinate system. Quaternion multiplication has its definition. Shown below.

There have q_1 and q_2, $q_1 = (w_1 + x_1 + y_1 + z_1)$ and $q_2 = (w_2 + x_2 + y_2 + z_2)$. Then the quaternion product q_1q_2 is given by:

$$q_1q_2 = \begin{pmatrix} w_1w_2 - x_1x_2 - y_1y_2 - z_1z_2 \\ w_1x_2 + x_1w_2 + y_1z_2 - z_1y_2 \\ w_1y_2 - x_1z_2 + y_1y_2 + z_1x_2 \\ w_1z_2 + x_1y_2 - y_1x_2 + z_1w_2. \end{pmatrix} \tag{3.22}$$

Since the 3D vector can be regarded as a quaternion, where the first element is equal to zero, it is possible to define a certain degree of a 3D vector around the axis by multiplication of the quaternion. Quaternion is an excellent and clean representation of the rotation between the two coordinates, or the direction of the vector. It is accessible to develop in coding, and the quaternion is not affected by any error and mistakes from Roll, Pitch and Yaw. Therefore, select the quaternion as the coordinate transformation tool in this project. Therefore, IMU sensor data rotate from body frame to the ground frame.

3.4.3 Acceleration to Displacement

To determine the movement of a mobile robot, the change in the position must be known. In the other word, the displacement must be calculated through acceleration. Use the quaternion to find the acceleration relative to the ground. The data output by the accelerometer is in g ($9.8m/s^2$), so the acceleration data must first be converted to data in meter. This subsection describes the use of quaternion to find linear acceleration, uses the trapezoidal integral method to find the displacement [32], and finally adopts motion detection to distinguish the static state of the mobile robot from the motion state.

3.4.3.1 Linear acceleration

Acceleration measurements include gravity component, as described in chapter 3 and background. If to compute linear acceleration, the gravitational component must first be removed from the accelerometer. The change in gravity component is shown in the figure below.

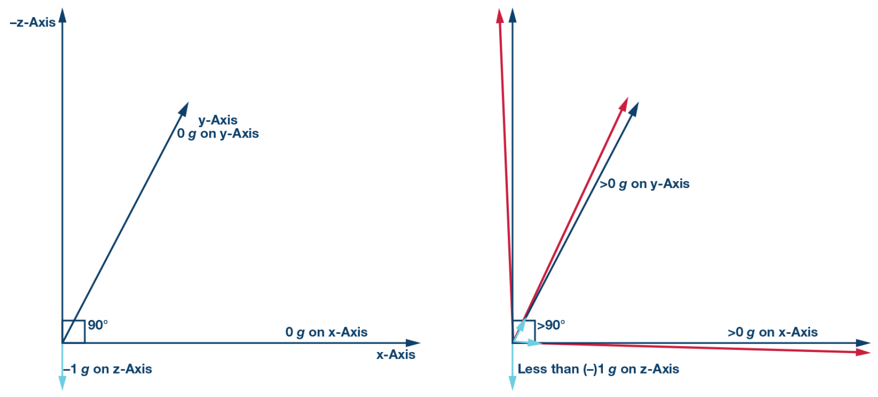

Figure 3.15: Gravity changed in rotation.

Figure 3.15 suggests that if the mobile robot moves on a completely horizontal plane, its gravity component is shown on the left of Figure 3.15. However, in normal environments, it is almost impossible for the mobile robot to make it, and its gravity component is shown on the right in Figure 3.15. Put it another word, the gravity component affects the measurement data of the accelerometer, explaining why the gravity component needs to be removed from the accelerometer.

The component of gravity can be expressed as $G = [0, 0, g]$.

Each gravity component is estimated from the attitude value of the current quaternion. It is used to compare the gravity components actually measured by the accelerometer to achieve the correction of the four-axis attitude [42].

$$\begin{pmatrix} g_x \\ g_y \\ g_z \end{pmatrix} = \begin{pmatrix} q_1 * q_3 - q_0 * q_2 \\ q_0 * q_1 + q_2 * q_3 \\ q_0 * q_0 - q_1 * q_1 - q_2 * q_2 + q_3 * q_3 \end{pmatrix}. \tag{3.23}$$

Linear acceleration can be expressed as

$$LinearAcc = \begin{pmatrix} a_x - g_x \\ a_y - g_y \\ a_z - g_z \end{pmatrix}, \tag{3.24}$$

where a_x, a_y, and a_z is the raw data from the accelerometer.

3.4.3.2 Displacement

The acceleration is integrated as a velocity vector, tracking the velocity throughout the process. The velocity vector is then integrated again to give the offset in one iteration. The following equation does this:

$$d_v = d_t * a, \tag{3.25}$$

$$d_s = d_t * d_v, \tag{3.26}$$

where d_v is the velocity vector, a is the acceleration, d_s is the displacement vector, d_t is sample time between two acceleration changed. However, mobile robots require real-time displacement updates and use discrete input values. Therefore, the trapezoidal integration method is used to calculate velocity and displacement. The following methods can determine the trapezoidal integral:

The First integration:

$$velocity(1) = velocity(0) + \frac{(acceleration(1) - acceleration(0)}{2} * d_t. \tag{3.27}$$

The second integration:

$$position(1) = position(0) + \frac{(velocity(1) - velocity(0)}{2} * d_t. \qquad (3.28)$$

In this equation velocity[1] and position[1] are the integrated output, velocity[0] and position[0] are the previous input, dt is the sample time like the previous description. Figure 3.16 shows the integrated different between general integral method and trapezoidal integral method [34].

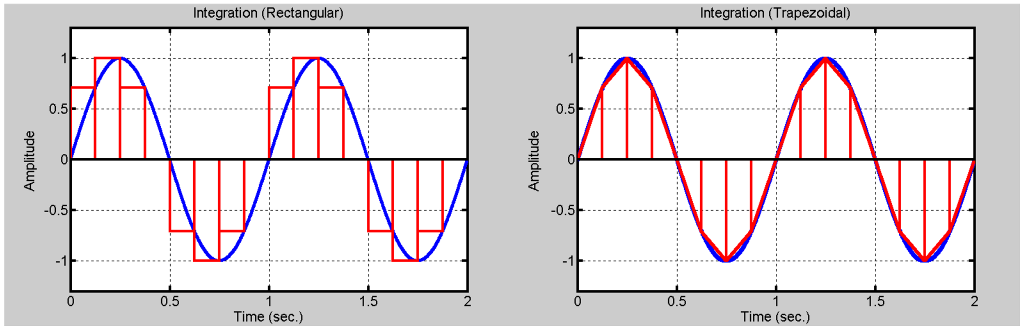

Figure 3.16: The general integral method and trapezoidal integral.

3.4.3.3 Motion detection

It is optimal to use the magnitude of the accelerometer to judge the motion state of the IMU. The following equation shows the relationship.

$$magnitude = \sqrt{a_x^2 + a_y^2 + a_z^2}. \qquad (3.29)$$

If the IMU is not moving, then the only acceleration it will feel will be due to the Earth's gravity. However, the calibrated IMU data ideally has a triaxial acceleration of 0. Therefore, the magnitude is also 0. In actual cases, the data generated by the three axes of the accelerometer will have slight fluctuations, and at this time, IMU can use the magnitude to calculate a threshold. The magnitude infers the motion state (moving or stationary) of the IMU by comparison with this threshold. If the magnitude is larger than the threshold, it means the mobile robot is in the moving situation. Whereas when the magnitude is smaller than the threshold, this shows the mobile robot stays in a stable situation. For the movement end check, the magnitude of the accelerometer determines the starting condition, and the value of the accelerometer determines the end of motion detecting. The movement end check is accomplished by continually reading the acceleration and comparing it to zero. If this is the case during a certain number of samples, then the speed returns to zero. Motion detection can effectively suppress the origin drift in the static state and has a simple function of filtering data.

3.4.3.4 Result

To test the displacement algorithm, the IMU is placed on a horizontal table, and the displacement sum can be used to figure out the distance of the IMU moves during the test. Then the test distance is compared with the actual distance to get the accuracy and error of the algorithm. This test has been combined with the motion detection of IMU. When the motion detection of IMU determines the IMU activity, then the server displays the movement change relationship of IMU.

The first testing is the IMU moving on the X-axis. Figure 3.17 shows the relationship between acceleration, velocity, and displacement of the x-axis.

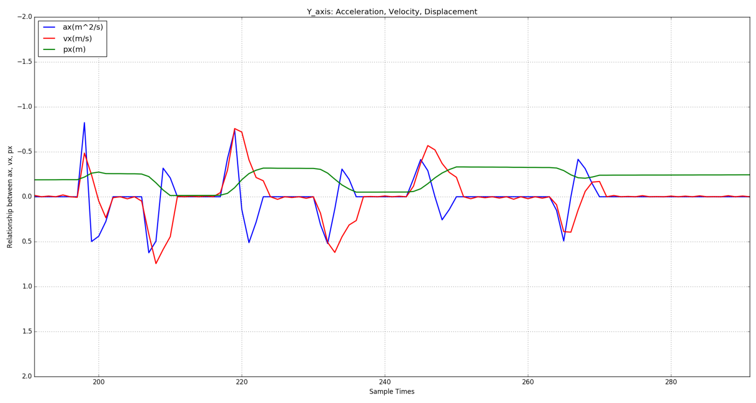

Figure 3.17: x-axis testing of the accelerometer.

The IMU moved 25 centimeters 15 times at x-axis direction across the horizontal table. In Figure 3.17, the blue line is the acceleration, the red line is the velocity, and the green line is the displacement of the IMU at x-axis between sampling times 180 and 300.

The second testing is the IMU moving on the y-axis. Figure 3.18 shows the relationship between acceleration, velocity, and displacement of the y-axis.

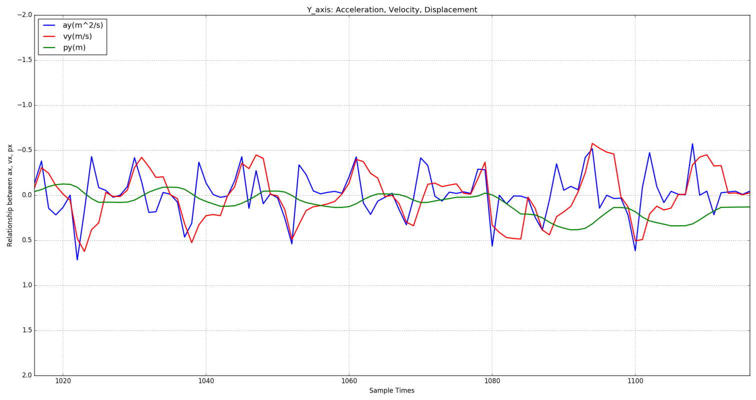

Figure 3.18: y-axis testing of the accelerometer

The IMU moved 10 centimeters 15 times at y-axis direction across the horizontal table. In Figure 3.18, the blue line is the acceleration, the red line is the velocity, and the green line is the displacement of the IMU at x-axis between sampling points 1000 and 1200.

3.4.4 Discussion

Through the experimental results, it can be found that the trapezoidal integral method and the quaternion can track the displacement of the IMU very well. However, it will also be found that after a certain number of movements, the IMU displacement cannot be tracked, that means, the displacement will drift. For example, the sampling range from 260 to 280 in Figure 3.17. And the sampling range from 1070 to 1080 in Figure 3.18

There are two reasons for drift. First, because the distribution of the magnetic field is not uniform, the direction of the yaw axis changes, so that the quaternion rotation matrix changes, resulting in the drift of the IMU. Second, the delay of IMU motion detection is mainly reflected in the fact that when the acceleration is zero, the speed cannot be zeroed in time, and additional displacement is generated from the time to cause the IMU to drift.The improvement is to improve the accuracy of motion detection. Avoid motion detection delays. Alternatively, add other filters to filter the data.

Chapter 4: Experimental results

4.1 System Structure

In previous chapters, complementary filtering, quaternion and displacement integral algorithms are introduced. This chapter mainly introduces the indoor test results of IMU positioning system of a mobile robot combined with all data processing methods. Using Arduino Uno to read out the data from BNO055 through serial communication, and using Raspberry Pi to send data to the server to display the trajectory by TCP [40] communication. The overall system structure diagram is shown in Figure 4.1.

4.2 Test Environment Setting

The indoor positioning of the mobile robot was tested in an indoor laboratory at Oregon State University. The testing ground is in a laboratory with an area around 6 meters by 6 meters with a grid layout on the floor exclusively for positioning tests. figure 4.2 shows the setup at the testing ground.

The IMU is considered as an add-on to the RF positioning system. Some experimental tests are completed to evaluate the IMU performance. The IMU connects to a micro controller that processes the sensor measurements and sends the data to the server through the wireless network. Then, the server back-end code estimates the position based on the sensor measurement to recover the target moving trajectory. The Figure 4.3 shows the overview of the mobile robot. Figure 4.4 reveals the setting of IMU on the mobile robot.

4.3 Hardware Components Used in Experiment

4.3.1 The IMU

In this thesis, the BNO055 [35] will be used (Figure 2.1) that includes accelerometers, gyroscopes, and magnetometers.

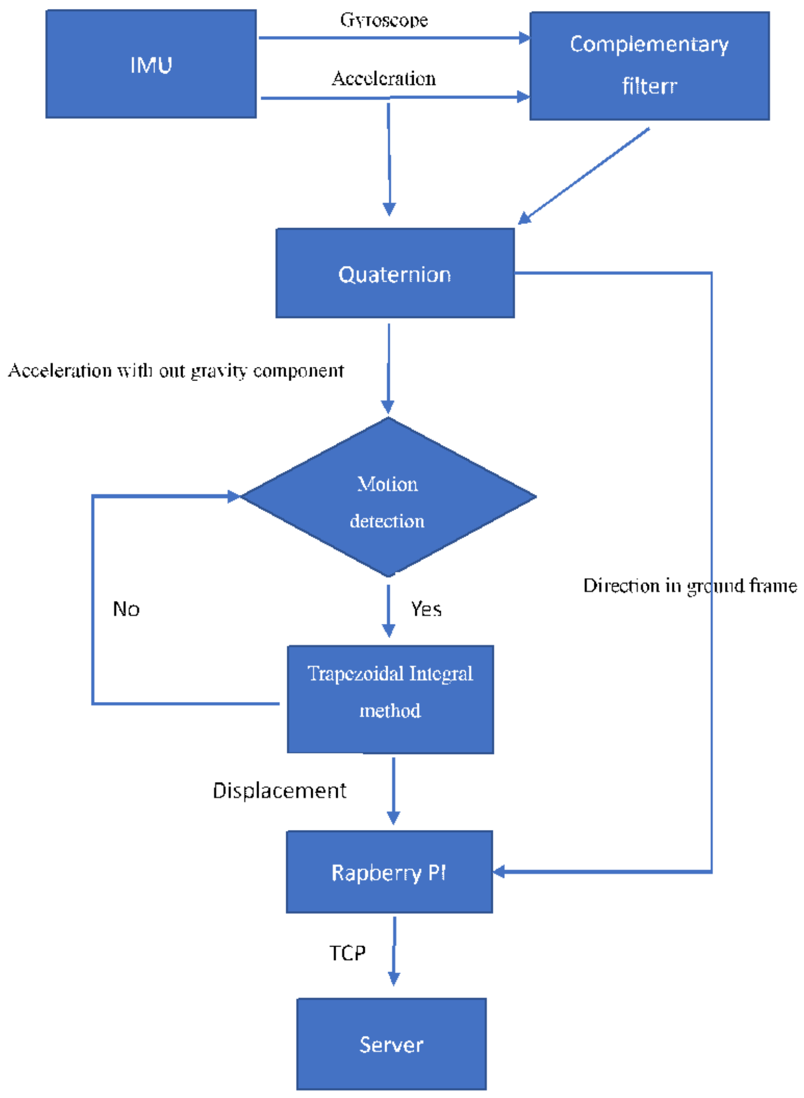

Figure 4.1: Overall system structure.

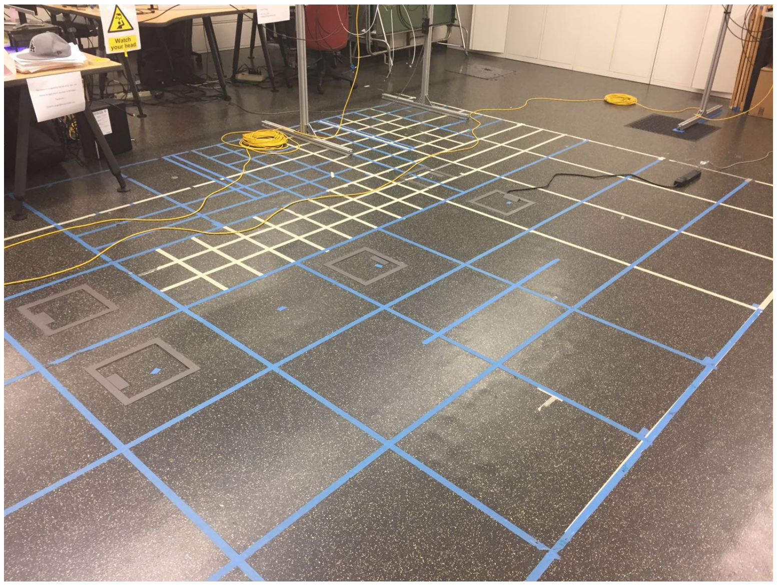

Figure 4.2: Overall system structure.

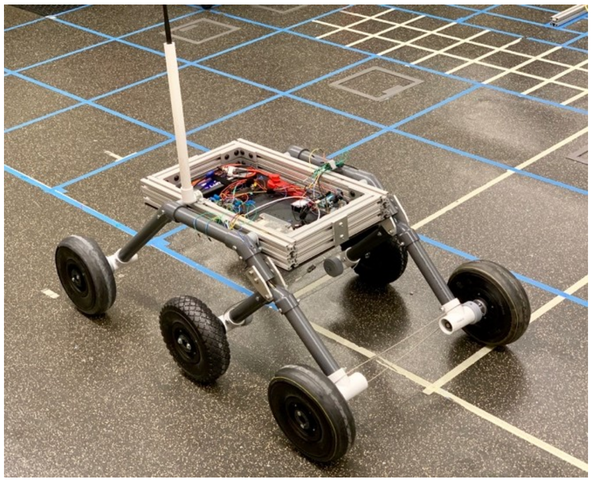

Figure 4.3: The overview of the testing environment.

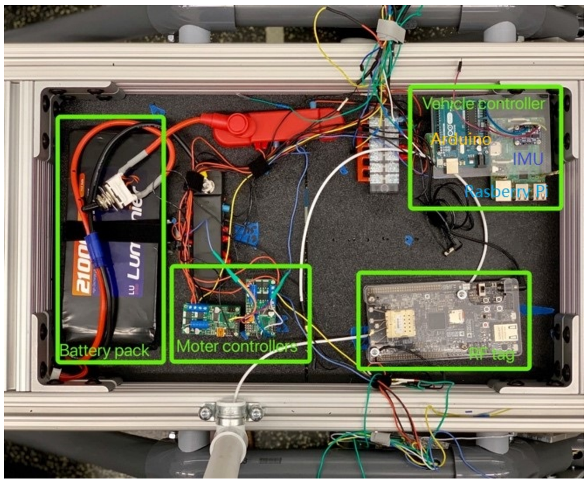

Figure 4.4: The setting of IMU on the mobile robot.

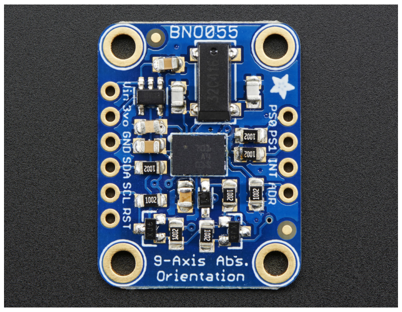

Figure 4.5: The IMU (BNO055).

Bosch Sensortec BNO055 is a 9-Axis Absolute Orientation Sensor. The BNO055 is integrating a three-axis 14-bit accelerometer with a range of ±2g, ±4g, ±8g, and ±16g. A three- axis 16-bit gyroscope with a range switchable from ±125 degrees per second to ±2000 degrees per second, and a three-axis geomagnetic sensor. These chipsets are integrated into a 28-pin LGA 3.8 mm x 5.2 mm x 1.1 mm housing. For improved system integration, the BNO055 is equipped with digital bidirectional I^2C and UART interfaces. BNO055 also includes a 32-bit ARM Cortex M0+ microcontroller for running Bosch Sensortec sensor fusion software. The figure below shows the System Architecture of BNO055 [28].

BNO055's internal processor integrates the accelerometer, gyroscope and magne-

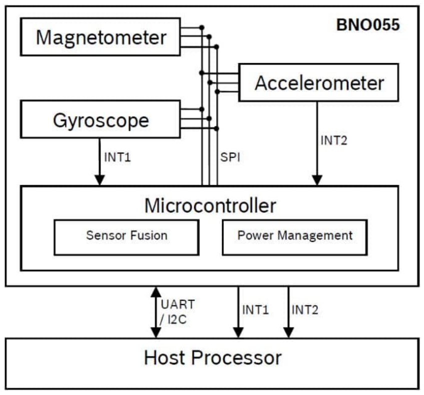

Figure 4.6: System architecture of BNO055.

tometer data, transmits it to the server via I^2C, and finally waits for further processing by the server algorithm.

There is an Arduino Uno library provided by Adafruit to readout the BNO055 with use of the I^2C bus [20]. And there also need I^2C library to drive BNO055. This I^2C library can be downloaded in the Arduino Library Management. The BNO055 library helps to initialize the I^2C bus, and there also have different functions to read out the data from the FIFO buffer. Also, the BNO055 library provides some functions to help with a sensors data reading, some of those data will be used to implement the calculation of the position and orientation of the vehicle, described in section 3.6. More about the I^2C bus background can be found in the next subsection.

The reason to select this IMU over others is because of its comprehensive implementation in combination with the Arduino, meaning that all subsystems can be able to communicate with each other subsystem. And many examples and libraries can be used as a sample. BNO055 data have a better measurement accuracy and stability compared to other IMU; the. On top of that, it is small enough to fit into the robot and takes up minimal space.

4.3.2 Communications Module

I^2C was proposed by Philips and is currently widely used for communication between multiple ICs in a system. I^2C is a bus capable of supporting multiple devices, including a two-wire serial data line SDA and a serial clock line SCL. Each device connected to the bus has a separate address which the host can use to access different devices. The host finds the slave through the SDA line transmission device address (SLAVE-ADDRESS). The SLAVE-ADDRESS can be 7 or 10 bits. A data bit following SLAVE-ADDRESS is used to indicate the data transmission direction, that is, the 8th or 11th bit. When it is 0, it means to write data; whereas when it is 1, it means to read data. The master device does not require an address because it generates a clock signal (via SCL) and processes the individual I^2C slave device.

The data on the SDA line must be stable during the high period of the clock, and the high or low state of the data line can only be changed when the clock signal on the SCL line is low. In other words, when SCL is high, it indicates valid data. SDA is high for "1" and low for "0". When SCL is low, it means invalid data. At this time, SDA will switch state and prepared the data for the next time. Figure 2-3 [41]shows the timing diagram of data validity.

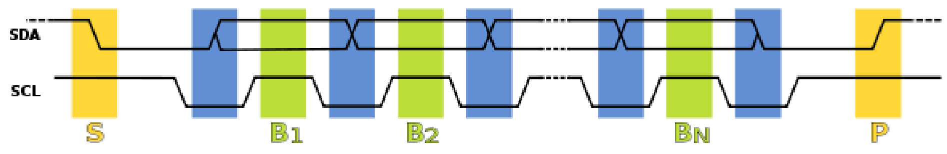

Figure 4.7: I^2C protocol.

Start condition S: When SCL is high, SDA is switched from high level to low level; Stop condition P: SDA transitions from low to high when SCL is high.

The host typically generates "start" and "stop" conditions. The bus is busy after the start condition, followed by the address of the slave (B1). If bit 0 of the address byte is set to 0, the master will write to the slave (B2). Once all bytes (Bn) are read or written, the master generates a Stop condition (P). After some time of the stop condition, the bus is idle again, and another device may use the bus.

The I^2C communication is implemented by using the I^2C dev library and the BNO055 library from Bosch and Adafruit. With using the functions of I^2C writeBytes, I^2C writeWords, I^2C readBytes, and I^2C readWords from the I^2C dev library, the I^2C is initialized and used to read out the BNO055. To collect data from the BNO055, the Arduino first indicates which registers are needed to be Readout. The BNO055 then puts these registers bytes in the FIFO buffer to be accessed by the I^2C Bus. The Arduino then reads out the First in First Out (FIFO) [37]buffer and puts the right bytes in the corresponding variables.

The single read mode is to read the data of each sensor of the BNO055 once, put it into the read register and issue an end signal, and enter the sleep mode again. With the single read mode, Arduino reads the data from the read register and sends data to the server. A small advantage of the single read mode is that the readout frequencies of BNO055 and Arduino are synchronized because Arduino reads the BNO055 data only when BNO055 is ready for new data.

Table 4.1: The Pin Set of BNO055

BNO055	Arduino Uno
SCL	Analog 5
SDA	Analog 4
VDD	3.3V DC
Ground	Common Ground

To test if I^2C communication is working, connect BNO055 to Arduino, according to Table 2.1. Turn on the power and read the values of the accelerometer, gyroscope, and magnetometer, then print them on the serial monitor of the IDE Arduino interface. When reading the data, the BNO055 is turned in all directions according to the x, y, z-axis of Accelerometer, Gyroscope, and magnetometer to check if the readings are as expected. The output of the accelerometer, gyroscope, and magnetometer corresponds to the expected value. Under such circumstances, it is concluded that I^2C communication is effective.

4.4 Trajectory Test Result

In the first experiment, the IMU was fixed on a mobile robot, and the mobile robot made a circular motion with a radius of 1.5 meters. The IMU collects data and performs real-time data transmission, and finally displays the trajectory on the server. The test results are shown in Figure 4.5.

The black point is the estimated trajectory of the mobile robot, and the blue line is the preset trajectory of the mobile robot. According to the experimental results, when

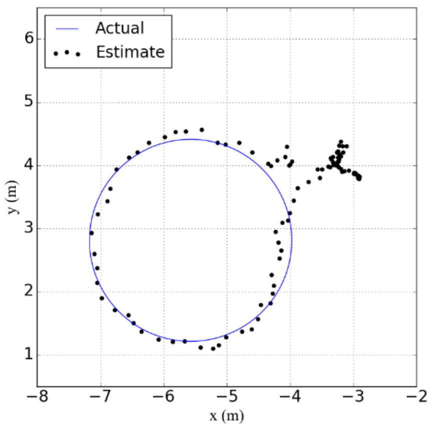

Figure 4.8: The a circular trajectory of the IMU.

the robot moves to some angles, the trajectory will have a great draft. The other angles, the preset trajectories, and the estimated trajectories match well.

In the second experiment, the robot followed a 1.5m by 1.5m square counterclockwise trajectory.See figure 4.6.

The black point is the estimated trajectory of the mobile robot, and the blue line is the preset trajectory of the mobile robot. According to the track results of test 2, it is found that both test 2 and test 1 have the same track drift problem.

4.5 Discussion

According to the experimental results, it is found that the mobile robot's motion estimation trajectory matches the preset trajectory, but the motion estimation trajectory still drifts in some directions. This result is consistent with the yaw angle error analysis in Section 3.5.2.

There have two reasons that cause estimated trajectory drifts. The first reason is that the laboratory still exists soft iron distortion in a certain direction. This magnetic field distortion causes errors in the yaw angle compensated by the magnetometer, which causes the quaternion rotation matrix to occur deviation. Eventually, the estimated trajectory drifted. The second reason is that when the value of the accelerometer becomes

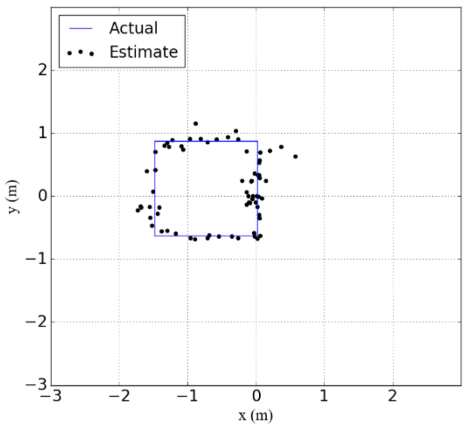

Figure 4.9: The straight-line trajectory of IMU.

0 during the experiment, the speed is not zero in time (about around 1s delay). This 1s delay causes the speed to output additional displacement data. Thereby the estimated trajectory drifts.

According to the test results and analysis, the IMU positioning system has the possibility of becoming an assistant positioning system for RF. The IMU positioning system exchanges data with the RF positioning system. In a short time, the RF system updates and corrects the position of the IMU. The IMU supplements the estimated displacement of the RF. Thereby achieving the purpose of the IMU positioning system to assist RF positioning system.

Chapter 5: Discussion and Conclusions

5.1 Discussion

It is widely known that wearable sensors with an IMU have become the new trend in the field in positioning technology. As has been introduced at the very beginning, the IMU is able to be linked to a wireless transmission device, through which the users are allowed both to supervise and process the motion-related variables in a distant way. This research is of paramount importance in that it fills the research gap caused by the insufficiency of simply using a radio frequency positioning system, and this can be mainly attributed to the complexity and uncertainty of signals in some cases. Under such circumstances, in order to offer more accurate, reliable and effective positioning function for a good variety of applications in modern society, the IMUs positioning system has to come and play a determinant role.

Before going any further, this thesis does find some difficulties encountered when using IMU positioning, and it turns out these barriers do affect the experimental result afterwards. The IMU is not positioned accurately and correctly mainly due to 2 reasons. First and foremost, the attitude inaccuracy leads to the outcome that the effects of gravity cannot be removed. Under such circumstances, it is no wonder why the errors are accumulated in a constant way. In addition to the attitude, the drift of the device itself cannot be avoided. To this problem, the general processing method is to remove with high-pass filtering, but the high-pass filtering only obtains relative motion relationship, and cannot get accurate position information. Therefore, in actual application, the corresponding processing method can be designed according to the real situation. For example, in gait positioning, motion still judgment can be used [30].

Generally speaking, therefore, this paper aims to explore the possibility of IMU providing assisted location for RF positioning system. After reviewing extensive relevant literature and experiment reports, this thesis develops and then examines a complete, independent, and lightweight indoor IMU-aided RF positioning system by conducting an experiment. To be specific, in the case of the mobile robot, the moving data of the IMU is collected, compared and then analyzed to estimates the trajectory of the mobile robot. This data can be processed and then display the movement trajectory of the mobile robot on the server side. The objective of this experiment is to enhance the accuracy of the system in indoor environments for mobile robotics up to a higher level. This thesis mainly integrates the measurements of the accelerometers, gyroscopes, as well as magnetometer from an IMU through a complementary filter. Furthermore, a calibration algorithm has been developed, which is used to minimize the IMU drift and error, which will be elaborated later. With the calibrated data, the trapezoidal integration method is able to utilize and then process the accelerometer data so as to evaluate the velocity and displacement of the mobile robot. Another major task throughout this experiment is: in order to transform the data in the coordinate system of the IMU mounted on the mobile robot body to the ground positioning coordinate that RF positioning uses, a quaternion rotation algorithm is operated. This is so in that it helps enable the fusion of the IMU and RF positioning estimates to accurately determine the moving trajectory of the mobile robot and guide its moving directions.

In addition, this research not only provides theoretical foundations which are necessary but also carries out experimental verification in a quantitative method. The key findings of this experiment can be demonstrated as following:

First, the complementary filter can make up for the output of the gyroscope heading angle very well. In addition, it has been found that the quaternion algorithm also works effectively for converting IMU body frames to ground frames. As far as the displacement is concerned during the experimental process, the IMU's motion detection accurately identifies the motion and rest of the IMU. The problem is, however, that the overall displacement remains inconsistent with the actual situation from time to time, and it turns out that the IMU will have certain drift on some occasions. What should be noticed is that this drift will increase over time. After analysis and tracking, it has been figured out that the core of the problem is still the complexity of the indoor ambient noise, and it can be amplified by the integration of velocity and displacement, which affects the accuracy of the result in the end. Under such circumstances, therefore, this key finding shows that although the IMU assisted positioning system is working within a short range of motion, it cannot run and operate for a longer period of time. In order to make up for this, it is advisable to re-initialize the data of the IMU within a certain period of time.

In addition, another point to be taken into consideration is that great chances are that the IMU positioning system might fail in complex motion trajectories. To explain, assuming that the IMU cannot find the dominant direction of movement within a long period of time, then it is much likely that the displacement trajectory will be misaligned. Although in theory, an electronic compass can be used as an additional source of data, some instruments or signals can interfere with the magnetic field in an indoor environment, making the electronic compass work abnormally. And this can be attributed to the unavoidable noise uncertainty. Saying that though, the short-range displacement results of the IMU lay a solid foundation for sensor fusion study in the coming research, especially when it comes to the fusion and incorporation between RF and IMU. With the complementary cooperation between RF and IMU, it is expected that the displacement status can be upgraded. In this respect, the possibility, accuracy as well as reliability of the trajectory tracking of mobile robots can be expected in the foreseeable future.

5.2 Conclusions

It is widely known that wearable sensors with an IMU have become the new trend in the field in positioning technology. As has been introduced at the very beginning, the IMUs is able to be linked to a wireless transmission device, through which the users are allowed both to supervise and process the motion-related variables in a distant way. This research is of paramount importance in that it fills the research gap caused by the insufficiency of simply using a radio frequency positioning system, and this can be mainly attributed to the complexity and uncertainty of signals in some cases. Under such circumstances, in order to offer more accurate, reliable and effective positioning function for a good variety of applications in modern society, the IMUs positioning system has to come and play a determinant role.

The short-range displacement results of the IMU lay a solid foundation for sensor fusion study in the coming research, especially when it comes to the fusion and incorporation between RF and IMU. With the complementary cooperation between RF and IMU, it is expected that the displacement status can be upgraded. In this respect, the possibility, accuracy as well as reliability of the trajectory tracking of mobile robots can be expected in the foreseeable future.

Bibliography

[1] James J Allen. Introduction to MEMS (microelectromechanical systems). Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2007.

[2] Y. Zou and Q. Wan, "Moving target localization in noncoherent distributed MIMO Radar system using range and range rate measurements," in *Proc. Asia-Pacific Signal and Information Process. Association(APSIPA)*, 2016, pp. 1–6.

[3] R. Ye, S. Redfield, and H. Liu, "High-precision indoor UWB localization: Technical challenges and method," in *Proc. IEEE ICUWB'10*, Nanjing, China, Sep. 2010.

[4] T. Qiao and H. Liu, "Improved least median of squares localization for non-line-of-sight mitigation," *IEEE Communications Letters*, vol. 18, no. 8, pp. 141–144, Aug. 2014.

[5] Ahmed El-Rabbany. *Introduction to GPS: the global positioning system.* Artech house, 2002.

[6] C. Hu, R. Khanna, J. Nejedlo, K. Hu, H. Liu, and P.Y. Chiang, "A 90 nm-CMOS, 500 Mbps, 3–5 GHz fully-integrated IR-UWB transceiver with multipath equalization using plse injection-locking for receiver phase synchronization," *IEEE Journal on Solid-State Circuits*, vol. 46, no. 5, pp. 1076-1088, May 2011.

[7] H. Liu, "Multicode ultra-wideband scheme using chirp waveforms," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 4, pp. 885-891, Apr. 2006.

[8] H. Liu, "Error performance of a pulse amplitude and position modulated ultra-wideband system over lognormal fading channels," *IEEE Communications Letters*, vol. 7, no. 11, pp. 531-533, Jul. 2003.

[9] H. Liu, R.C. Qiu, and Z. Tian, "Error performance of pulse-based ultra-wideband MIMO systems over indoor wireless channels," *IEEE Transactions on Wireless Communications*, vol. 4, no. 6, pp. 2939-2944, Nov. 2005.

[10] C. Hu, R. Khanna, J. Nejedlo, K. Hu, H. Liu, and P.Y. Chiang, "A 90 nm-CMOS, 500 Mbps, 3–5 GHz fully-integrated IR-UWB transceiver with multipath equalization using plse injection-locking for receiver phase synchronization," *IEEE Journal on Solid-State Circuits*, vol. 46, no. 5, pp. 1076-1088, May 2011.

[11] S. Zhao, H. Liu, and Z. Tian, “Decision directed autocorrelation receivers for pulsed ultra-wideband systems,” *IEEE Transactions on Wireless Communications*, vol. 5, no. 8, pp. 2175-2184, Aug. 2006.

[12] S. Zhao, H. Liu, and Z. Tian, “A decision-feedback autocorrelation receiver for pulsed ultra-wideband systems,” in *Proc. IEEE Radio and Wireless Conf.*, pp. 251-254, Sep. 2004

[13] S. Zhao and H. Liu, “On the optimum linear receiver for impulse radio systems in the presence of pulse overlapping, *IEEE Communications Letters*, vol. 9, no. 4, pp. 340-342, Apr. 2005.

[14] S. Zhao, P. Orlik, A.F. Molisch, H. Liu, and J. Zhang “Hybrid ultrawideband modulations compatible for both coherent and transmit-reference receivers,” *IEEE Transactions on Wireless Communications*, vol. 6, no. 7, pp. 2551-2559, July 2007.

[15] S. Zhao and H. Liu, “Transmitter-side multipath preprocessing for pulsed UWB systems considering pulse overlapping and narrow-band interference,” *IEEE Transactions on Vehicular Technology,* vol. 56, no. 6, pp. 3502-3510, Nov. 2007.

[16] S. Zhao and H. Liu, “Prerake diversity combining for pulsed UWB systems considering realistic channels with pulse overlapping and narrow-band interference,” in *IEEE Global Telecommunications Conference*, Nov. 2005.

[17] S. Hara, D. Anzai, T. Yabu, T. Derham, and R. Zemek, “Analysis on TOD and TDOA location estimation performances in a cellular system,” in *Proc. 2011 IEEE International Conference on Communications (ICC)*, 2011.

[18] T. Qiao, S. Redfield, A. Abbasi, Z. Su, and H. Liu, “Robust coarse position estimation for TDOA localization,” *IEEE Wireless Communications Letters*, vol. 2, no. 6, pp. 623–626, Dec. 2013.

[19] T. Qiao and H. Liu, “An improved method of moments estimator for TOA based localization,” *IEEE Communications Letters*, vol. 17, no. 7, pp. 1321–1324, Jul. 2013.

[20] Hoffmannjan. Adafruit-bno055, 2016.

[21] Berthold KP Horn. Closed-form solution of absolute orientation using unit quaternions. *Josa a*, 4(4):629–642, 1987.

[22] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[23] Kampfkarren. Take out pitch from rotation matrix while preserving yaw and roll, 2018.

[24] Christopher Konvalin. Compensating for tilt, hard-iron, and soft-iron effects, 2009.

[25] LGUniteState. Intuitive control is at hand, 2019.

[26] Gabriele Ligorio and Angelo Sabatini. Extended kalman filter-based methods for pose estimation using visual, inertial and magnetic sensors: comparative analysis and performance evaluation. *Sensors*, 13(2):1919–1941, 2013.

[27] Robert Mahony, Tarek Hamel, and J-M Pflimlin. Complementary filter design on the special orthogonal group so (3). In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 1477–1484. IEEE, 2005.

[28] MouserElectronicsUniteState. Bosch sensortec bno055 intelligent 9-axis absolute orientation sensor, 2019.

[29] NavibeesUniteState. Introduction to indoor navigation systems, 2017.

[30] Abdelmoumen Norrdine, Zakaria Kasmi, and Jörg Blankenbach. Step detection for zupt-aided inertial pedestrian navigation system using foot-mounted permanent magnet. *IEEE Sensors Journal*, 16(17):6766–6773, 2016.

[31] Michał Nowicki, Jan Wietrzykowski, and Piotr Skrzypczyński. Simplicity or flexibility? complementary filter vs. ekf for orientation estimation on mobile devices. In *2015 IEEE 2nd International Conference on Cybernetics (CYBCONF)*, pages 166–171. IEEE, 2015.

[32] Kurt Seifert and Oscar Camacho. Implementing positioning algorithms using accelerometers. *Freescale Semiconductor*, pages 1–13, 2007.

[33] Ken Shoemake. Animating rotation with quaternion curves. In *ACM SIGGRAPH computer graphics*, volume 19, pages 245–254. ACM, 1985.

[34] Lance D Slifka. An accelerometer based approach to measuring displacement of a vehicle body. *Master of Science in Engineering, Department of Electrical and Computer Engineering, University of Michigan–Dearborn*, 2004.

[35] Kevin Townsend. Adafruit bno055 absolute orientation sensor, 2015.

[36] Wikipedia. Electronic stability control, 2019.

[37] Wikipedia. Fifo (computing and electronics), 2019.

[38] Wikipedia. Gimbal lock, 2019.

[39] Wikipedia. Quaternion, 2019.

[40] Wikipedia. Transmission control protocol, 2019.

[41] A Two wire Serial Protocol. I2c info – i2c bus, interface and protocol, 2019.

[42] Xiaoping Yun, Eric R Bachmann, and Robert B McGhee. A simplified quaternion-based algorithm for orientation estimation from earth gravity and magnetic field measurements. Technical report, NAVAL POSTGRADUATE SCHOOL MONTEREY CA DEPT OF ELECTRICAL AND COMPUTER ..., 2008.

APPENDICES

Appendix A: Positioning Diagram

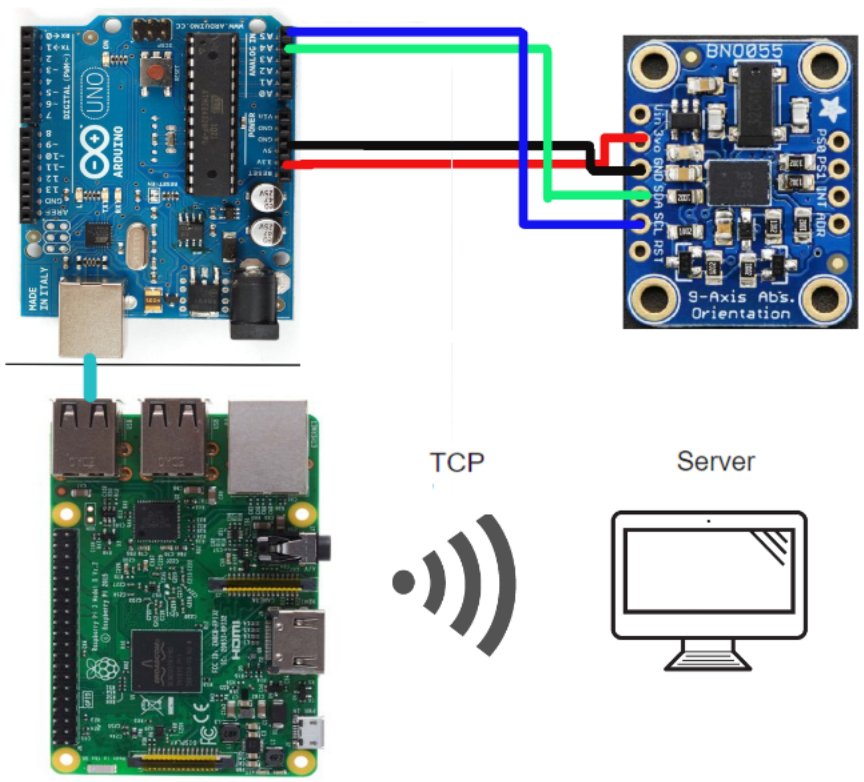

Figure A.1: Positioning Diagram

Appendix B: Source Code

More source code at https://github.com/zhangshengkai520/MasterThesisSourceCode

B.1 The Code of Positioning Information

```
# The relationship between acceleration, velocity , and displacement
import socket, pickle , time
import matplotlib.pyplot as plt
from math import sqrt
import numpy as np
from scipy import signal
font2 = {'family': 'Times New Roman',
'weight': 'normal',
'size': 24,
}
def server_program():
host = '192.168.2.198'
print(host)
port1 = 5656
server_socket1 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket1.bind((host, port1))
server_socket1.listen(10)

print("Server is ready to receive data .....")
conn1, address1 = server_socket1.accept()
print("Connect from : " + str(address1))
lin_accel_old = np.zeros((1, 3))
linVel_old = np.zeros((1, 3))
linVel_new = np.zeros((1, 3))
```

```
linPos_old  = np.zeros((1,  3))
linPos_new = np.zeros((1,  3))
steptime = 0.12
i  = 0
plt.grid()
plt.ion()
P_x = []
P_y = []
#      __________________plotting_____________________
Steptime = []
Time = 0
ax = []
ay = []
az = []
vx = []
vy = []
vz = []
px = []
py = []
pz = []
Euler = []
#       ________________________________________________
samplePeriod = 1/100
ACCELEROMETER_DRIFT_WHEN_STATIONARY = 2.3e-26
countaccX = 0
countaccY = 0

while True:
plt.grid()
start  = time.time()
data1 = conn1.recv(5 * 1024 * 1024)
data_package1 = pickle.loads(data1)
euler,  accel,    lin_acc ,q = read_imu(data_package1)
```

```
if i > 20 :
accMagnitude = sqrt((lin_acc[0] * lin_acc [0]) + (lin_acc [1] * lin_acc [1]) + (lin_acc
    [2] * lin_acc [2]) )
# print(accMagnitude)
# Use a high−pass filter to remove some noise
 filterCutoff = 0.001
butterFilterB, butterFilterA = signal.butter(1, (2 * filterCutoff ) / (1 /
    samplePeriod), 'highpass')
accMagnitudeFiltered = signal. filtfilt (butterFilterB, butterFilterA, [accMagnitude,
    accMagnitude], padlen=1)

# Take the absolute value of the filtered magnitude
accMagnitudeFiltered = abs(accMagnitudeFiltered)
# print(accMagnitudeFiltered)
# Use a low−pass filter to remove some noise
 filterCutoff = 5
butterFilterB, butterFilterA = signal.butter(1, (2 * filterCutoff ) / (1 /
    samplePeriod), 'lowpass')
accMagnitudeFiltered = signal. filtfilt (butterFilterB, butterFilterA,
[accMagnitudeFiltered, accMagnitudeFiltered], padlen=1)

# Are we actually stationary?
stationary = accMagnitudeFiltered <
    ACCELEROMETER_DRIFT_WHEN_STATIONARY
print(stationary, 'stationary ', accMagnitudeFiltered)
# If we are stationary, don't bother doing anything
if stationary.any():

linVel_new =np.zeros((1, 3))
linPos_new += linVel_new * steptime
P_x.append(linPos_new[0, 0])
P_y.append(linPos_new[0, 1])
 lin_accel_old = np.zeros((1, 3))
```

```
linVel_old  = np.zeros((1, 3))
continue
Time += steptime
Steptime.append(Time)
Acc = [lin_acc [0]  * 9.81,  lin_acc [1]  * 9.81,  lin_acc [2]  * 9.81]
acc_new = np.matrix(quaternrotate(Acc, quaternconj(q)))
#  ________________________Trapezoidal      rule___________________
# Current velovity
leakRateAcc = 1
linVel_new = linVel_new * leakRateAcc + ((lin_accel_old + ((acc_new - lin_accel_old)
    /2)) * steptime)
 lin_accel_old  = acc_new
#  ______________________movement   end check window for
         Acceleration_______________________________________

 if  -0.12 < acc_new[0,0] < 0.12:
acc_new[0,0]  = 0
acc_new[0,1]  = 0
 elif  -0.12 < acc_new[0,1] < 0.12:
acc_new[0,1]  = 0
acc_new[0,  0]  = 0
#__________X-axis
 if  acc_new[0,0]  == 0:
countaccX +=1
 else:     #reset counter
countaccX = 0
 if  countaccX > 1:
linVel_new [0,0]  = 0
 linVel_old  [0,0]  = 0
countaccX = 0
#  ________________Y  -axis

 if  acc_new[0,1]  == 0:
```

```
countaccY += 1
else:
countaccY = 0
if countaccY > 1:
linVel_new [0,1] = 0
linVel_old [0,1] = 0
countaccY = 0
print(linVel_new)
#    -------------------------------------
leakRatevel = 1
# linPos_new = linPos_old + ((acc_new + lin_accel_old) / 4) * steptime * steptime +
    linVel_old * steptime
linPos_new = linPos_new * leakRatevel + (linVel_old + ((linVel_new − linVel_old)/2))
    * steptime
linVel_old = linVel_new
# ------------data   ploting-----------
# print(linVel_new, linVel_old , linPos_new, linPos_old , lin_acc )
# print(linVel_new)
Euler.append(euler[0])
ax.append(acc_new[0, 0])
ay.append(acc_new[0, 1])
az.append(acc_new[0, 2])
vx.append(linVel_new[0,0])
vy.append(linVel_new[0,1])
vz.append(linVel_new[0,2])
px.append(linPos_new[0,0])
py.append(linPos_new[0,1])
pz.append(linPos_new[0,2])
a = plt.subplot(1,1,1)
plt.xlim(i − 100, i )
# plt.ylim(−3, 3)
a.plot(px, "b", linewidth=2.0)
a.plot(vx, "r", linewidth=2.0)
```

```
a.plot(ax, "g", linewidth=2.0)
a.plot(Euler, linewidth=2.0)
a.plot(vy, linewidth=2.0)
a.plot(ay, linewidth=2.0)
a.xaxis.grid(True, which='major')
a.yaxis.grid(True, which='major')
a.legend(['px', 'vx', 'ax'], loc='upper left')

# b = plt.subplot(3,1,2)
# b.plot(ay , "b", linewidth=2.0)
# b.plot(vy, "r", linewidth=2.0)
# b.plot(py, "g", linewidth=2.0)
# b.xaxis.grid(True, which='major')
# b.yaxis.grid(True, which='major')
# b.legend(['ay', 'vy', 'py'], loc='upper left')

# c = plt.subplot(3,1,3)
# c.plot(Time,px[-1] , "b", linewidth=2.0)
# c.plot(Time,py[-1], "r", linewidth=2.0)
# c.plot(Time,pz[-1], "g", linewidth=2.0)
# c.xaxis.grid(True, which='major')
# c.yaxis.grid(True, which='major')
# c.legend(['px', 'py', 'pz'], loc='upper right')
#     ----------------------------------------------------
P_x.append(linPos_new[0, 0])
P_y.append(linPos_new[0, 1])
# linVel_old, linPos_old = linVel_new, linPos_new
# print(P_x[-1], P_y[-1])


# if i == 200:
#     linVel_old , lin_accel_old , linPos_old = reset_data()
#     P_x[:] = []
#       P_y[:] = []
#       P_x.append(0)
#       P_y.append(0)
#       i = 0
plt.scatter(P_y, P_x, s=20, label='Trajectory', c='k')
plt.scatter(P_y[0], P_x[0], s=100, label='Start', c='g')
plt.scatter(P_y[-1], P_x[-1], s=100, label='Goal', c='r')
plt.legend(loc='upper left')
plt.axis('equal')
plt.pause(0.00000000000001)
plt.show()
plt.clf()



i += 1
# print(i)
def read_imu(data_package1):
euler_data = [data_package1[0],data_package1[1], data_package1[2]]
accel_data = [data_package1[3]/9.81,data_package1[4]/9.81, data_package1[5]/9.81]
linear_data = [data_package1[6]/9.81, data_package1[7]/9.81, data_package1[8]/9.81]
q = [data_package1[9], data_package1[10], data_package1[11], data_package1[12]]
# q = [ data_package1[10], data_package1[11], data_package1[12],data_package1[9]]

return euler_data, accel_data, linear_data, q
def quaternconj(q):
q = [q[0], -q[1], -q[2], -q[3]]
return q

def quaternprod(a ,b):
ab = [0, 0, 0, 0]
ab[0] = a[0] * b[0] - a[1] * b[1] - a[2] * b[2] - a[3] * b[3]
ab[1] = a[0] * b[1] + a[1] * b[0] + a[2] * b[3] - a[3] * b[2]
```

```
ab[2] = a[0] * b[2] − a[1] * b[3] + a[2] * b[0] + a[3] * b[1]
ab[3] = a[0] * b[3] + a[1] * b[2] − a[2] * b[1] + a[3] * b[0]
return ab

def quaternrotate(acc, q):
x = quaternprod(q, [0, acc [0], acc [1], acc [2]])
y = quaternprod(x , quaternconj(q))
z = np.array([y [1], y [2], y [3]])
return z
def reset_data():
 lin_accel_old = np.zeros((1, 3))
 linVel_old = np.zeros((1, 3))
linPos_old = np.array((−2.4, 0, 0))
return linVel_old , lin_accel_old , linPos_old


 if __name__ == '__main__':
server_program()
```

B.2 Data Record

```
import socket
import pickle
import datetime
import csv


outputFilename1 = "IMU_Values_{0}.csv".format(int(datetime.datetime.now().strftime
    ("%Y%m%d%H%M")))
outputFilename2 = "Compass_Values_{0}.csv".format(int(datetime.datetime.now().
    strftime("%Y%m%d%H%M")))
headers1 = ['DT', 'TP', 'A_x', 'A_y', 'A_z', 'G_x', 'G_y', 'G_z']
```

```
headers2 = ['DT', 'x_out', 'y_out', 'z_out', ' x_digital_out ', ' y_digital_out ', '
    z_digital_out ', ' x_digital_out ', 'Bearing']

with open(outputFilename1, 'w', newline='') as csv_a:
writer_a = csv.DictWriter(csv_a, fieldnames=headers1)
writer_a.writeheader()
csv_a.close()
with open(outputFilename2, 'w', newline='') as csv_c:
writer_c = csv.DictWriter(csv_c, fieldnames=headers2)
writer_c.writeheader()
csv_c.close()

def server_program():
host = '192.168.2.198'
print(host)
port1 = 5656
port2 = 6565
server_socket1 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket2 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket1.bind((host, port1))
server_socket2.bind((host, port2))
server_socket1.listen(10)
server_socket2.listen(10)

print("Server is ready to receive data .....")
conn1, address1 = server_socket1.accept()
conn2, address2 = server_socket2.accept()
print("Connect from : " + str(address1),"Connect from : " + str(address2) )

while True:
global i
i=0
data1 = conn1.recv(5 * 1024 * 1024)
```

```
data2 = conn2.recv(5 * 1024 * 1024)
data_package1 = pickle.loads(data1)
data_package2 = pickle.loads(data2)
print(data_package1, data_package2)
write_csv_imu(data_package1)
write_csv_com(data_package2)
# read_csv()
#plotting(ax)
i = i + 1

def write_csv_imu(data_package1):
Temp=data_package1[0]
Accle_x = data_package1[1]
Accle_y = data_package1[2]
Accle_z = data_package1[3]
Gyro_x= data_package1[4]
Gyro_y = data_package1[5]
Gyro_z = data_package1[6]
Datetime = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
rows = [{'DT': Datetime, 'TP': Temp, 'A_x': Accle_x, 'A_y': Accle_y, 'A_z': Accle_z,
'G_x': Gyro_x, 'G_y': Gyro_y, 'G_z': Gyro_z}]
with open(outputFilename1, 'a+', newline='') as csv_imu:
writer_w = csv.DictWriter(csv_imu, delimiter=",", lineterminator='\n', dialect='
    excel', fieldnames=headers1)
writer_w.writerows(rows)

def write_csv_com(data_package2):
X_out = data_package2[0]
Y_out = data_package2[1]
Z_out = data_package2[2]
X_digital_out = data_package2[3]
Y_digital_out = data_package2[4]
Z_digital_out = data_package2[5]
```

```
Bearing = data_package2[6]
Datetime = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
rows = [{'DT': Datetime, 'x_out': X_out, 'y_out': Y_out, 'z_out': Z_out, '
    x_digital_out ': X_digital_out ,
' y_digital_out ': Y_digital_out , ' z_digital_out ': Z_digital_out , 'Bearing': Bearing}]
with open(outputFilename2, 'a+', newline='') as csv_com:
writer_c = csv.DictWriter(csv_com, delimiter=",", lineterminator='\n', dialect='
    excel', fieldnames=headers2)
writer_c .writerows(rows)

if __name__ == '__main__':
server_program()
```