UPPSALA
UNIVERSITET

UPTEC F 15067
Examensarbete 30 hp
November 2015

Investigation of Inertial Navigation for Localization in Underground Mines

John Svensson

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Investigation of Inertial Navigation for Localization in Underground Mines

John Svensson

This thesis project considers the potential use of inertial navigation on a consumer grade tablet mounted in a vehicle in an underground mine. The goal is to identify which sensors and techniques are useful and to design a navigation algorithm based on those results. The navigation algorithm is intended to work alongside the current received signal strength indication (RSSI) positioning system. Testing of the gyroscope, accelerometer and magnetometer sensors suggest that, while dead reckoning is likely not precise enough, an orientation filter can be designed that can be used for navigation. A complementary orientation filter using the gyroscope and accelerometer is then designed that shows better results than the default sensor fusion solutions available in Android. The filter is expandable and can come to include magnetometer data in the future. Based on the outputs of this filter, a navigation algorithm based on angle matching with map information is proposed.

Precise positioning in an underground mine can be crucial to employee safety, and may also bring production benefits.

Handledare: Fredrik Ekenstedt
Ämnesgranskare: Mikael Sternad
Examinator: Tomas Nyberg
ISSN: 1401-5757, UPTEC F15 067

POPULÄRVETENSKAPLIG SAMMANFATTNING

Rörelsesensorerna i en kommersiell Android-surfplatta testas och utvärderas för syftet att förbättra positioneringssystemet i en underjordisk gruva. Utifrån de sensorer som verkar lovande byggs ett komplementärt rotationsfilter upp. Rotationen från filtret används sen som grund för en navigeringsalgoritm som bygger på jämförelser mellan de rotationer som ett fordon gör och möjliga vägar synliga i en karta över gruvan. Algoritmen samarbetar med nuvarande positioneringssystem som bygger på signalstyrkemätningar i gruvans trådlösa nätverk för att approximera en uppkopplad enhets position.

Samtliga sensorer (accelerometer, gyroskop och magnetometer) visar sig vara användbara för navigering. Tyvärr så ger dödräkning av accelerometerdata typiskt sett dålig noggrannhet oberoende av sensorkvalitet, på grund av att man tvingas att numeriskt integrera data två gånger för att nå den önskade storheten. Detta gör att minsta lilla statiska fel eller brus kan ge upphov till snabbt växande fel i den slutliga approximationen. Gyroskopet är desto mer precist, och används med framgång för att följa plattans rotation. Magnetometern lider av störningar från omkringliggande metall, vilket är speciellt tydligt då enheten placeras i en bil. Ett sätt att isolera jordens magnetfält från störningarna föreslås och indikationer på att det är möjligt visas. Tyvärr så visar flera av sensorerna upp icke-konsekventa mätvärden, vilket begränsar noggrannheten.

Det komplementära rotationsfilter som byggs upp visar sig prestera bättre än motsvarigheten som följer med Android, och uppvisar ett fel i storleksordningen några grader per minut. Filtret använder accelerometern som referens för vertikalriktningen och gyroskopet för att följa enhetens rotation, men kan även expanderas till att använda sig av magnetometerdata om det blir tillgängligt.

Navigationsalgoritmen som föreslås bygger på ett partikelfilter, där flera virtuella fordon traverserar gruvan utifrån de rotationer som registreras. När de flesta virtuella fordon eliminerats, då de försökt ta omöjliga svängar eller kommit för långt från aktuell accesspunkt, kommer vi närmare den sanna positionen.

TABLE OF CONTENTS

1 INTRODUCTION

1.1 BACKGROUND

1.1.1 Underground Mines

An underground mine is a dangerous and complex environment. At any one time, a large number of machines, trucks, and operators are working in different parts of the mine, which often consists of a large sprawling network of tunnels. In the event of a fire or other life-threatening situations, it is crucial to have a reliable and accurate way of finding the positions of everyone located underground. Knowing the position of people in potential danger allows for targeted rescue missions, and knowing when everyone is out of the way of danger avoids putting rescue personnel through unnecessary risks. In normal day-to-day operations having an accurate positioning system can allow for increased production efficiency, with less time spent figuring out where operators and machines are located. Going forward, it may also pave the way for traffic control systems that could help avoid unnecessary hiccups in production. One such example is a fully loaded truck having to stop in the middle of a steep ramp due to unexpected oncoming traffic.

Since GPS cannot penetrate hundreds of meters of rock, several of Boliden's underground mines currently use a positioning solution based on *received signal strength indication* (RSSI) from access points in the mines' WLAN networks. This provides a position estimate with an error commonly less than 100m, but depending on the WLAN coverage in the area it may be as high as 200-250m, which is the maximum range of the access points used. This rough position estimate is currently used to automatically control the ventilation systems in some of Boliden's mines [1].

1.1.2 Inertial Navigation

Inertial navigation, in general, can be said to be the usage of motion and rotation sensors to track the movement of an object via some numeric calculations. Typically the motion sensor is an accelerometer, measuring linear acceleration, and the rotation sensor is a gyroscope, measuring angular momentum. Common to all *inertial navigation systems* (INS) is that they suffer from integration drift as none of the sensors directly measure the desired quantity (position or orientation) [2]. This is especially true for accelerometer data as it needs to be integrated twice, often causing rapidly increasing errors. The amount of drift depends on the quality of sensors and the algorithms used to process the raw data. To eliminate the drift, all inertial navigation systems need to synchronize with external reference data at startup and regularly during operation.

An example of this is a car's navigation system which uses GPS as an external reference, while inertial navigation and drivetrain information (such as wheel speed) are used to calculate positions on shorter timescales or when GPS coverage is unavailable.

While not inertial per se, many INS also use a magnetometer that can measure the earth's magnetic field. Combined with the direction of gravity measurable by the accelerometer, this provides an external reference of orientation in all degrees of freedom.

1.1.3 Alternate Solutions

Apart from inertial navigation systems (with some external reference), there are a number of other possible solutions that could work on their own or alongside an INS. Here, we discuss some of the more noteworthy ones and why they may or may not be suitable for an underground mine environment.

RSSI data from a WLAN network can be used to approximate the position of a connected device in a number of ways. First, one can approximate the position of the device to be the same as the position of the wireless access point with highest signal strength (or the associated one, depending on the implementation). To increase the accuracy further, one may employ a *trilateration* algorithm that approximates the position based on the signal strengths of all visible access points. For the highest accuracy, one can make use of *fingerprinting*, which means comparing the current RSSI data with previously sampled data. While theoretically the most precise method, RSSI fingerprinting has a number of downsides compared with trilateration. Conventional fingerprinting requires a large amount of work prior to operation, work which may also have to be repeated in select areas following changes to the infrastructure (such as access points being moved) [3]. Also, the RSSI landscape may appear different depending on which vehicle the connected device is placed in. Currently several of Boliden's mines use RSSI positioning with highest strength approximation [1].

Another possibility is to use *radio frequency identification* (RFID) tags placed around the mine, which when read by passing vehicles indicate its position. However RFID readers that can read passive tags from a reasonable range can be large and cumbersome, and covering a significant part of the mine with high precision requires a very large amount of tags. Instead, RFID tags are more suitable to be used at certain choke points (which can help positioning or enable *gating* functions), or otherwise more important areas. RFID tags can also be placed on mobile equipment, allowing these to be located by passing vehicles. RFID is currently used in production for some applications, and is being tested for several more [1].

Time on arrival (ToA) is a technique used in several positioning systems (such as GPS), which uses the difference in arrival times of signals from several reference points to determine the position of the receiver. While very accurate, this requires precise synchronization of the senders, and can suffer from *multipath* propagation issues where the signal bounces off walls or other surfaces in the surroundings. The main downside of ToA approaches is that they require large amounts of additional hardware to be placed throughout the mine.

Angle of arrival (AoA) data enables triangulation of a signal source by looking at phase differences of a received signal in an antenna array. Reflections and multipath propagation are obviously a problem, and AoA is mostly useful in open areas. However, knowing the direction of a signal source could give important qualitative data when navigating in an underground mine. For example, receivers placed in intersections could determine from which path the signal originated. Some AoA information is possible to extract from a WLAN system, using bi-directional access points.

One or more of these technologies may be used by themselves or in conjunction with inertial measurements to find and track the position of a device.

1.2 PROJECT DESCRIPTION

The purpose of this project is to investigate the possibility of using inertial navigation techniques on a consumer grade Android tablet (Samsung Tab S) to improve the accuracy of the current RSSI-based positioning system. Tablets are currently used in the mines for other purposes, so no new hardware needs to be introduced. This is especially important in the mine environment, where installation and maintenance work (such as battery changes) is time consuming and expensive. In particular, infrastructure-dependent positioning solutions are undesirable due to the large environment. Since vehicle standards vary between brands, the project is limited exclusively to the hardware available in the tablet. This means no information from the vehicle itself, such as wheel speed or odometer data, is available for use. As external reference, an RSSI position estimate is acquired through a HTTP request to a web server.

The project is divided into three parts. First, the tablet’s sensors will be investigated to determine the quality of the sensor data. This includes an evaluation of the sensor fusion algorithms available by default in the Android operating system, and programming of an Android application for data collection. The goal of this part is to determine which sensors provide useful data for the purpose of navigating in a mine.

The second part is an investigation of various filter techniques to process the raw sensor data. A suitable orientation filter is implemented and tested for drift, responsiveness and sensitivity to noise of various frequencies. It is important that the filtering method chosen can handle the heavy vibrations from driving on the rough roads underground. The goal here is to find a filter that handles the sensor data sufficiently well, and to determine what useful output data can be extracted. Also, since the tablet will run several other applications at the same time, it is worth taking computational performance into account at this stage. The precision and performance of the filter is tested with an Android application running the filter in real time.

Lastly, the available filter outputs are used to design a navigation algorithm for navigating in an underground mine. To be considered for further development, the

algorithm should be able to provide an accurate position estimate that is more precise and smoother than the current RSSI positioning. It should be able to detect its own mistakes, and recover from them as quickly as possible. Since the algorithm will be running on a tablet also using other software, good performance is desirable. The proposed algorithm is presented and discussed.

All Android applications are written in Java and compiled using *Android Studio* unless otherwise stated. There is an option to access lower level functions of Android by using the *Native Development Kit* and programming in a lower level language. For this project, the regular *Software Development Kit* and Java is used. Data analysis and presentation is done with Matlab.

2 ANDROID TABLET SENSORS

This chapter includes information on the sensors available on the Samsung Tab S. First, there is a listing of the available relevant sensors and why they may or may not be suited for navigating in an underground mine. Then follows a description of experiments made to determine the quality and viability of the various sensors, as well as some results from these experiments. Lastly, there is a conclusion detailing which sensors were chosen to be used for this project and why. Visual inspection and on-the-go testing of sensors is done using the Android application "Physics Toolbox Suite" by Chrystian Vieyra [4].

On the software level, each sensor reading comes in a package called a "sensor event", which contains the raw sensor data, a timestamp and an indication of which sensor produced the event. The sensor event is passed to a "sensor event listener" which uses the information in the way desired [5].

2.1 AVAILABLE HARDWARE SENSORS

This section is a listing of the relevant hardware sensors available on the tablet and how they may be used for a navigation application. They are referred to as hardware sensors even though there is some layer of abstraction and calibration between the actual physical sensor and the readings obtained. The distinction is made to differentiate them from the software sensors described in 2.2, which use sensor fusion of more than one hardware sensor [6]. All sensors give their measurements in the device frame of reference [5].

For the Samsung Tab S in particular, the accelerometer and gyroscope are found on the same chip [7].

2.1.1 Accelerometer (BMI055)

The accelerometer measures linear acceleration along each axis of the device. In theory, it can be used to calculate a good approximation of any translatory motion by integrating twice over time. Unfortunately, any small error in the measurement is also integrated twice, producing an error of quadratic growth in the result [6]. Also, error in the starting velocity will cause a linearly growing error over time. Considering that measurements from the sensor come at a limited rate and that the orientation must be taken into account, we must always expect some error to be present, even from a perfectly calibrated accelerometer.

However, the accelerometer provides a direct measurement of the direction of gravity that is not subject to any integration side-effects. This can be used as a reference of the attitude of the device, preventing any rotational integration drift except that around the vertical axis [8].

Looking at raw data from the accelerometer can also give us qualitative information about the situation. If the absolute value of the measurement vector is close to g, we are likely to be still or in non-zero linear motion. One could

possibly detect when the vehicle is likely to be standing completely still by looking at vibrations and thus pause the navigation algorithm. A problem with that is that it may be hard to tell road vibrations from the engine idling or a machine working on something in the mine, such as drilling or scaling (removal of loose chunks of rock from the walls and ceiling).

2.1.2 Gyroscope (BMI055)

The gyroscope measures angular velocity around each axis of the device. As shown in appendix 6.2, this is in theory enough to calculate the orientation of the device relative to some starting orientation. However, in practice every real world gyroscope suffers from small errors, resulting in drift of orientation [2]. Since the measured quantity is only the first derivative of the desired result, this drift is usually quite small compared with that resulting from accelerometer errors, even with cheap hardware. In a perfectly calibrated gyroscope, the resulting drift is noise-driven, but in many cases there will be some locally linear drift from a slowly varying gyroscope bias.

In Android, the gyroscope sensor has some built-in drift and compensation that is not visible to the user; however, a version without drift compensation is also available [5]. For the purpose of this project, the drift-compensated gyroscope is chosen.

2.1.3 Magnetometer (AK09911C)

The magnetometer measures magnetic field strength along each axis of the device. Apart from the light sensor, this is the only raw sensor available that directly measures an external reference – the magnetic field of Earth. In perfect conditions, this means one can easily eliminate all rotational drift around the vertical world axis. However, the magnetic field measured by the magnetometer is a superposition of the geomagnetic field and any local magnetic fields present, such as from magnetized metal objects or running motors. In addition to this, the geomagnetic field is distorted nearby ferromagnetic objects, such as steel beams or vehicles' structural elements (magnetic deviation) [2]. Thus it may be difficult, depending on the location and situation, to isolate the geomagnetic field. This is especially true in a vehicle, where the detected field may be completely dominated by fields local to the body and engine of the vehicle. Assuming this local field is static with respect to the device (as in, the device is fixed within the moving magnetic field of the vehicle), it could be isolated and eliminated as the vehicle turns. This is further investigated in section 2.4.3.

In an underground mine, one would also expect some errors to be introduced by mine infrastructure and the presence of ferromagnetic minerals in the rock. In particular, iron oxides such as magnetite will make magnetic fields very unpredictable as an external reference where they are common. That said, none of Boliden's current underground mines have high content levels of ferromagnetic minerals [1].

As with the gyroscope, the Android magnetometer sensor comes in two forms. The default, and most commonly used, magnetometer has a calibration algorithm that automatically calibrates when rotating the device [5]. This is mainly to eliminate the parts of the magnetic field that come from the metal parts and vibration motor of the device itself. There is also a non-calibrated magnetometer available, which, according to specification, does none of those things. For this project, the default magnetometer is used.

2.2 AVAILABLE SOFTWARE SENSORS

This section lists potentially relevant *software sensors* available on the tablet and how they may be used in a navigation application. They are called software sensors as they use sensor fusion of several hardware sensors to produce their output [6]. Again, all sensors give their readings in the device frame of reference.

2.2.1 Linear Acceleration Sensor

The linear acceleration sensor outputs an estimate of the linear acceleration along each device axis. The main difference from the accelerometer is that the output does not include gravity, which makes it useful for dead reckoning applications.

2.2.2 Gravity Sensor

The gravity sensor is similar to the linear acceleration sensor, but returns only the gravity component of the acceleration. The direction of gravity is the main indication of the attitude of the device and can eliminate rotational drift around two of the three axes.

2.2.3 Rotation Vector Sensor

The rotation vector sensor outputs the orientation of the device in the form of a quaternion representing the rotation from some default orientation to the current orientation. This sensor uses the magnetometer to prevent drift around the vertical world axis when available.

2.2.4 Game Rotation Vector Sensor

Otherwise identical to the rotation vector sensor, the game rotation vector sensor does not, according to specification, rely on the magnetometer for heading. It cannot prevent drift around the vertical world axis, but is not affected by variations in local magnetic fields.

2.3 ADDITIONAL SENSORS

Lastly, this section contains a short description of Android sensors that are not available on the Samsung Tab S but still deserve mention when working with navigation.

2.3.1 Pressure Sensor

A pressure sensor measures air pressure around the device. Visual analysis on another device suggests it is fairly sensitive and could estimate altitude (or depth in the case of a mine) quite well. However, the active ventilation systems down in the mine may cause local pressure differences and as such, the viability of pressure as a depth indicator would have to be investigated further before use.

2.4 SENSOR EXPERIMENTS

This section contains descriptions and results of experiments and research done to determine the quality and usefulness of the different sensors. Each subsection corresponds to a specific potential usage of the sensors, rather than a specific sensor, and includes a conclusion specific to that usage. The final conclusion on which sensors are used is given in section 2.5.

2.4.1 Polling rates of Hardware Sensors

In Android, there are a number of pre-defined sensor polling rates that are suited to various types of applications. The available rates are: “Normal”, “UI”, “Game” and “Fastest” [5]. For inertial navigation, having faster sensor polling rates is generally better, as this allows more accurate sampling of high frequency vibrations, reducing white noise [2].

To determine the upper limit of sampling rates for the hardware sensors, while at the same time investigating the ability to use several sensors simultaneously, all three hardware sensors as well as the game rotation vector were all set to gather data at the “Fastest” rate. The analysis was then done by looking at the periods between contiguous time stamps for each sensor separately. Results from a 40 second test run are shown in Figure 2.4-1.

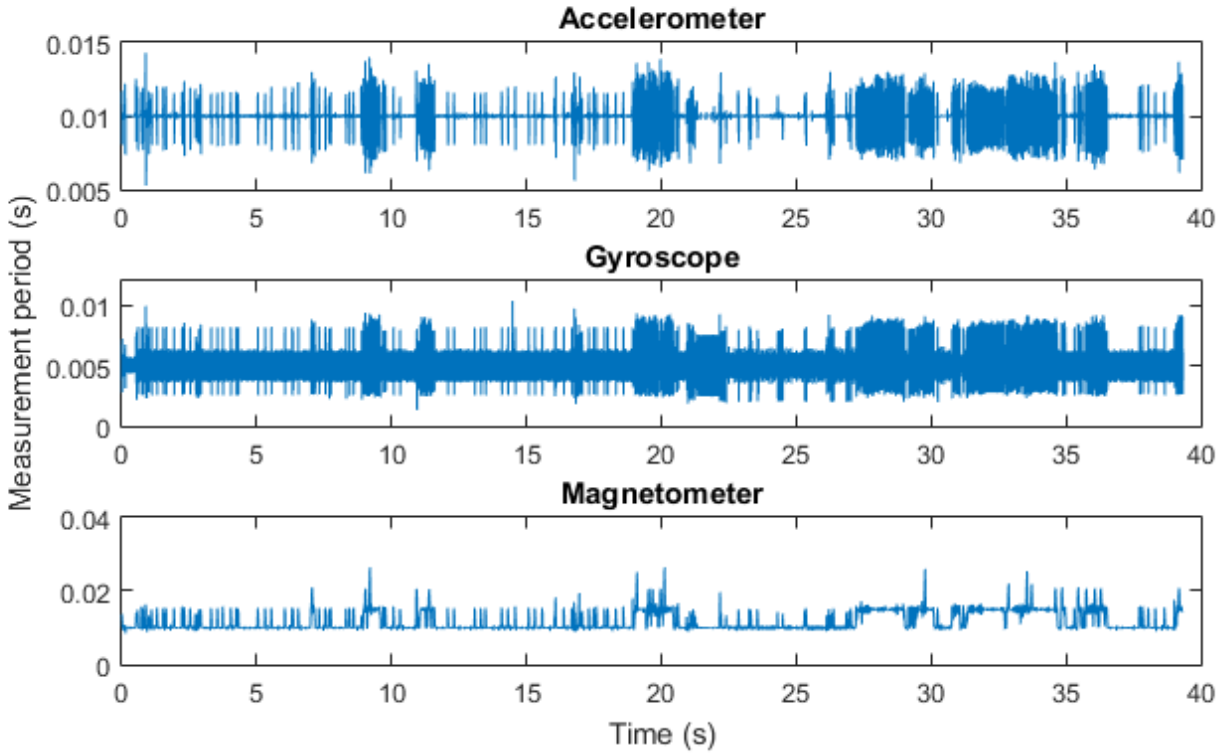

Figure 2.4-1. The figure shows the time between measurements of the hardware sensors measuring at the "Fastest" speed. Data was gathered with all three sensors and the game rotation vector sensor running simultaneously over 40 seconds. All sensors show fairly stable rates, and disturbances correlate between all three, suggesting external influence.

Both the accelerometer and the magnetometer seem to work with a period of 10 ms, or a rate of 100 Hz, while the gyroscope is faster with its period at 5 ms, a rate of 200 Hz. All three sensors show variations between individual measurements, but only the accelerometer and the gyroscope have periods shorter than their apparent target rates, suggesting they measure on some fixed external clock. The magnetometer, however, does not and thus effectively has a slower rate. Software sensors, like the game rotation vector sensor, are synced to one of their hardware sensors.

There are clear similarities between all three graphs. Larger variations in the accelerometer and gyroscope data seem to correlate with longer delays in the magnetometer data, suggesting these variations are not limitations of the individual sensors themselves, but rather in some layer of software. This brings to question whether or not to use the difference between sensor reading timestamps or the expected period between readings in calculations based on the accelerometer and gyroscope. This would depend on where the difference appears. For this project it was decided to use the timestamps, as this is in accordance with official code examples [5].

It would seem as if all sensors provide readings at a reasonably stable rate, with the longest periods being about twice as long as the expected ones. It is also worth noting that the magnetometer is less sensitive to slower or varying rates, as it measures the desired quantity directly, and no integration is required.

2.4.2 Accelerometer Calibration and Dead Reckoning

Looking at the accelerometer readings when the device is in a fixed position reveals that it is not quite calibrated. Values often exceed 10 m/s^2, which is a clear deviation from the expected 9.81 m/s^2. Upon further investigation, it was discovered that the values tended to change over time, even with the device completely fixed. The most probable cause for this appears to be a temperature dependency of the accelerometer, since the values often stabilized if the device was doing the same thing for an extended period of time. Naturally, this makes finding a good calibration difficult, as the device will be operating in an environment where both ambient temperature as well as internal temperature from using the device will vary largely.

Ideally, one would want an on-line calibration algorithm that reduces bias without distorting the data. While some such algorithms exist [9], it was decided to use a simple error model for calibration to try to remove the large errors that appeared consistently through most runs.

The error model used is a description of the accelerometer output $a^{¤}$ along a single axis as

$$a^{¤} = k \cdot a + b_a, \tag{2.4:1}$$

where k is the scaling error and b is the bias.

The calibration process was done by reading the displayed a^* when the axis was aligned with the direction of gravity, and the device held still ($a \approx g$). This was done in both the positive and negative direction of the axis, which produced a system of two equations, from which the estimates k^* and b^* of k and b were obtained. For the tablet used, the values found were:

$$\boldsymbol{k}^* = (1.0032,\ 1.0010,\ 1.0010),$$
$$\boldsymbol{b}_a^* = (-0.069,\ 0.052,\ 0.130).$$

The scaling error is quite small on all three axes, but the bias error is considerable, especially on the z axis. Approximating $a \approx a^*$, where

$$a^* = \frac{a^{¤} - b_a^*}{k^*}$$

for each axis, we obtain the calibrated output. Use of the calibration method described above improves the accelerometer accuracy, but it still remains unstable

due to the apparent temperature dependency and some cross-axis dependencies. For the project, it was decided that no further calibration would be done, as any gains would soon be lost in the large uncertainties. These uncertainties also mean that the accelerometer included in the test device is not suitable for calculating position changes through dead reckoning, as per reasons given in section 2.1.1. However, the accelerometer remains useful as a way to detect the direction of gravity, and dead reckoning is most likely to be used for smoothing transitions between more accurate position estimates obtained in some other fashion.

2.4.3 Heading and Magnetometer in a Vehicle

Since the tablet is intended to be mounted inside a moving vehicle, knowing how the magnetometer is affected by the vehicle itself is important if it is to be used as part of a navigation solution. Looking at the output values when sitting in a vehicle immediately reveals that the magnetometer is heavily affected by the environment, as the values clearly differ from those read outside.

To see how the magnetometer readings varied in a moving vehicle, data was logged while driving around a parking lot with the device fixed flat on the passenger seat. During the test sequence, the vehicle made a number of turns, resulting in a total rotation of about 360° in a counter-clockwise direction. The collected data can be seen in Figure 2.4-2.

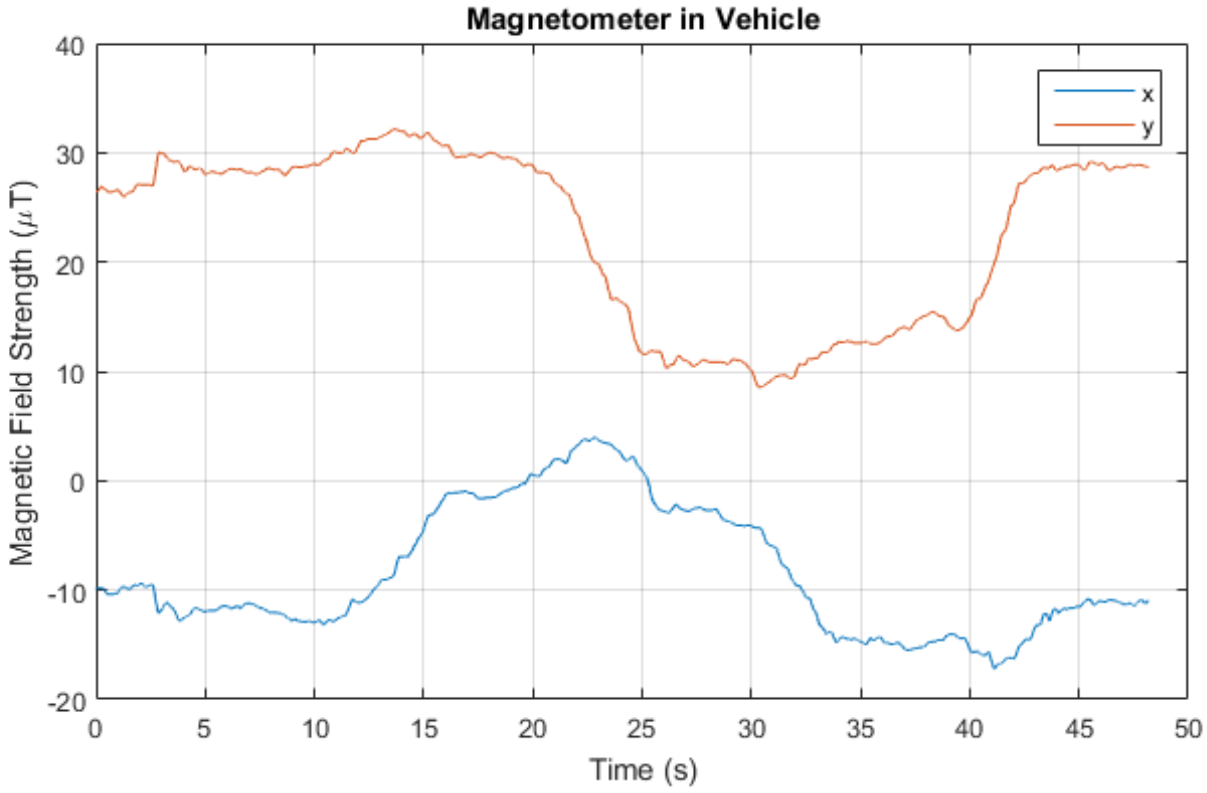

Figure 2.4-2. The graph shows magnetometer readings from within a vehicle making a 360° counter-clockwise rotation. In this test, the y axis points in the forward direction of the vehicle and the x axis points through the door on the right side of the vehicle. Both sets show a distinct bump around the 3-second mark, which is when the engine was turned on, and show wave-like features. The large separation between the two sets suggests the presence of a strong magnetic field turning with the vehicle.

Since both sensor axes along which measurements were done lie in the horizontal plane for the entire experiment, one would expect the values from the respective sets to intersect at some points if only the geomagnetic field is present. Since they both travel the same trajectory through the same plane, measuring the same quantity, one would expect them to show the same values, albeit with a phase delay. This is not the case, and the only difference between when the y axis points, say, north and when the x axis points north is the orientation of the vehicle and the device body. From this we can conclude that the vehicle and the device body have strong magnetic fields that interfere with the magnetometer readings.

This is of course problematic when wishing to use the magnetometer for a heading reference. However, if the magnetic fields produced by the vehicle and device body are somewhat static with respect to the moving reference frame of the device, they would appear as a static bias in the readings, which could then be filtered out. If only the geomagnetic field is present, plotting the x and y readings of a magnetometer rotating around the vertical would draw a circle around the

origin, with the strength of the horizontal component of the field being the radius of the circle. Such a plot for the data presented in Figure 2.4-2 is shown in Figure 2.4-3.

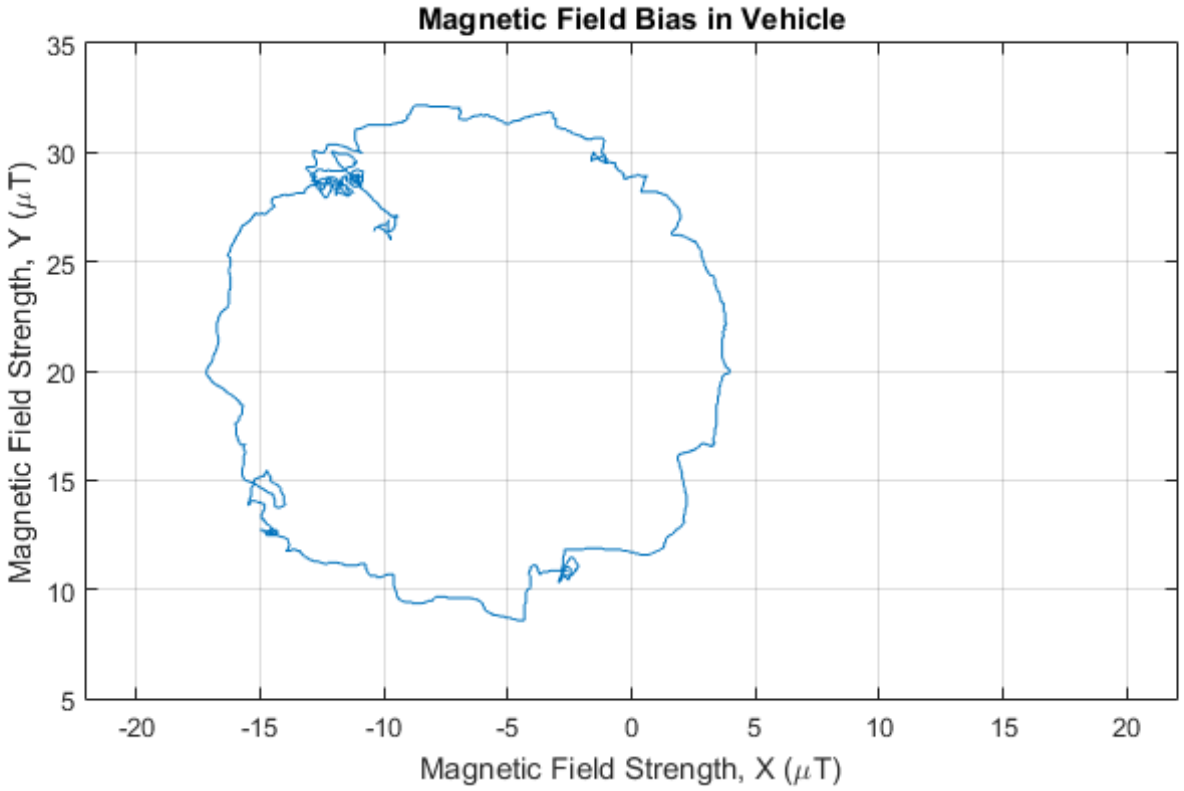

Figure 2.4-3. The figure shows the x and y readings of a magnetometer turning with a vehicle plotted against each other. The paired values form a circular shape around a point near (-6, 20) with an approximate radius of about 10 μT. The circular shape suggests that the magnetic fields from the vehicle and the device body can be approximated to be static in the reference frame of the device.

The path drawn in the plot clearly resembles a circle, which suggests that the magnetic fields from the vehicle and the device body are well approximated as static in the reference frame of the device. Removing the error could then be done by finding the circle which best fits the set of collected points, for example using a least-square algorithm [10]. The geomagnetic field data could then be used in conjunction with a gyroscope to provide a smooth and accurate heading [2].

Eliminating the bias in this fashion requires the vehicle to move around for a while before a reliable circle fitting can be done. Depending on what part of the mine the vehicle is operating in, this may not happen right away. Also, local magnetic disturbances from infrastructure, other vehicles, or ferromagnetic ores may further complicate the search.

To determine the usefulness of the magnetometer for this project, we have to consider other options that fill the same role – an external reference of heading.

One such option is that of *map matching*, which in an inertial navigation context means to use map information to exclude impossibilities, like driving inside a wall. This is especially interesting in the underground mine case, where tunnels are often just wide enough for a single vehicle, and the map is very sparse in the sense that the domain of possible positions and headings is very small (there is a lot of solid rock per unit area of road). For this reason, map matching is a strong alternative to magnetometers for heading references, given that a sufficiently accurate sensor is available to track rotation on a shorter time scale. That said, magnetic heading and map matching are not exclusive and combined could simplify the positioning task. An ideal case of this would be to obtain the heading from the magnetometer, and then look at map segments where a vehicle is likely to have this heading.

2.4.4 Orientation Tracking

Whether one uses navigation using dead reckoning, map matching, or both, knowing the orientation of the device is very important. As discussed in section 2.1.1, a dead reckoning algorithm will quickly diverge if there is any error in the incoming data. This does not only apply to the acceleration measurements, but also to the rotation measurements that determine which world frame direction the accelerations apply to. In fact, even the slightest errors in an orientation estimate will quickly create large position errors [6]. For this reason, having an accurate estimate of the orientation is important (even without dead reckoning, the orientation can be important for map matching). A brief summary of different ways to represent rotations numerically is found in appendix 6.1.

The rotation vector sensor provides immediate access to the orientation of the device [5], and for this reason it is easy to pick up and use without implementing any filter algorithm. The downside is that it is only available as-is, and cannot be modified.

Since the rotation vector sensor uses magnetometer data, it can provide a complete description of the device orientation. To get a first look at how it is affected by local magnetic fields, a test app that visually represents the measured orientation was used. Immediately, it was clear that the sensitivity and reliance on the magnetometer data was too high, as simply moving it around the office environment while keeping it in a relatively stable orientation showed deviations of up to 90º. While it seemed on average to show a correct orientation, the readings were very inconsistent, and were very choppy when moving. When the device was placed in a car with its motor running, it completely lost track of north, even after extended running. It would seem the built-in filtering techniques are unable to eliminate the static field of the vehicle.

As the rotation vector sensor cannot be modified, it was deemed unsuitable for this project. However, the game rotation vector sensor, similar to the rotation vector except for not relying on the magnetometer [5], could still be used for orientation measurements with some external heading reference to prevent

horizontal drift. To investigate the accuracy of the game rotation vector sensor, sensor output was logged to a file and processed in Matlab. Since no hardware to externally measure the rotation was available, it was carefully ensured that the orientation of the device at the beginning of a test was the same as that at the end, by alignment of the device to a predetermined position. For comparison, gyroscope data was also collected and integrated. The method of converting gyroscope data to a rotation quaternion is described in detail in appendix 6.2.

To quantify the rotations, we look at the *angle of displacement* of the orientation quaternion in every time step, which corresponds to the angle θ in the quaternion, defined as in equation (6.1:10). The initial position corresponds to a zero rotation quaternion, defined in equation ((6.1:14). A graph showing the angle displacements from zero rotation over time when the device is picked up, spun randomly in the air, and then placed at its initial position is shown in Figure 2.4-4. The game rotation vector sensor does take any deviation from a completely horizontal position (based on accelerometer data) into account, and may as such show a non-zero value from the start, but in this particular case, the device can be considered to be completely flat. A video representation of the changing rotations, as well as the raw gyroscope data, is also available [11].

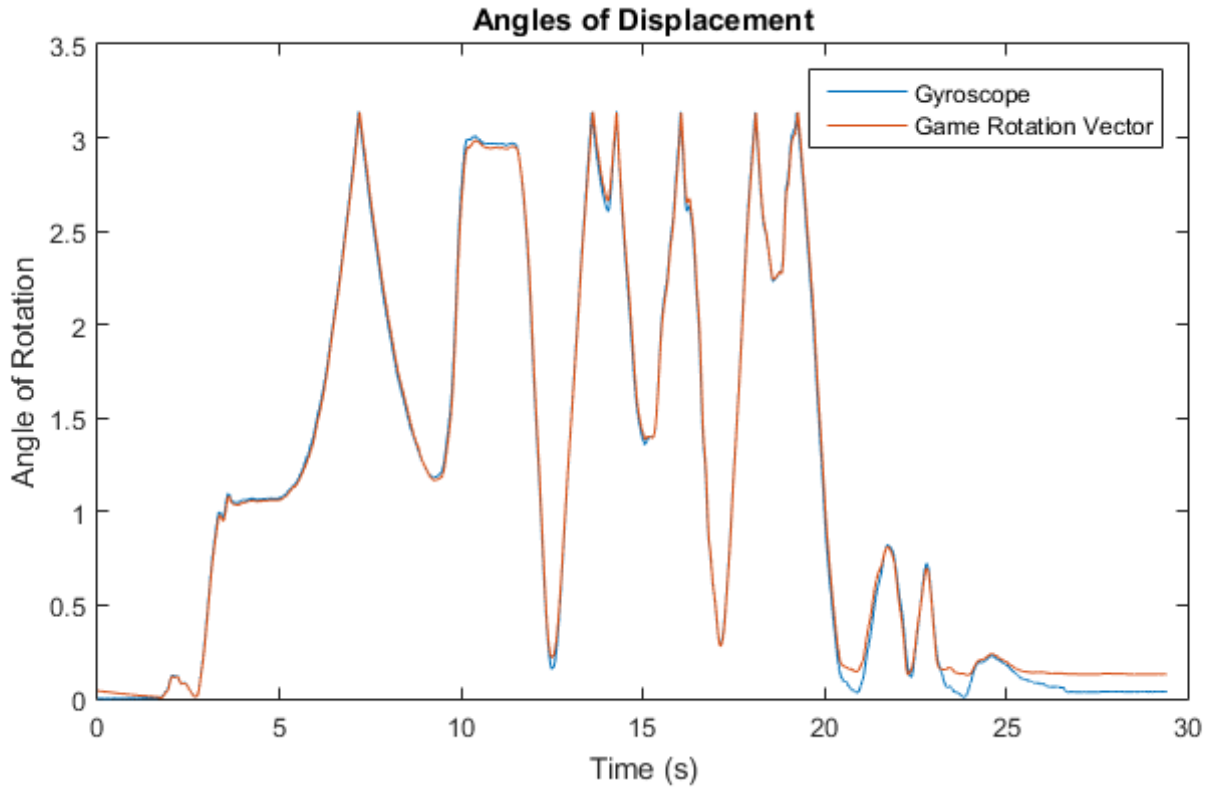

Figure 2.4-4. The graph shows the angle part of the axis-angle representation of the device's orientation over time. The device was returned to its original position, meaning near-zero end values indicate better accuracy or less drift. Both sensors perform quite similarly, but the gyroscope seems to perform slightly better towards the end of the experiment.

We can see that both sensors perform similarly for most of the test, but the gyroscope ends up having a slightly better end result (being closer to zero). This is commonly seen in many of the test runs, and it would appear that the game rotation vector's underlying filter causes some side effects when motion stops. In general, the gyroscope and game rotation vector seem to perform about equally well, with one or the other performing better in individual tests. The gyroscope integration's disadvantage of not being able to counteract *vertical drift* (in this context defined as any drift not in the horizontal plane) becomes very apparent during some tests.

To investigate the drift characteristics of the two methods, the device was placed in a fixed position for extended periods of time while collecting sensor data. To evaluate how much the two methods drift, the same angle of displacement used in the previous test is logged. Results from a 70-minute test are shown in Figure 2.4-4, and a video showing the rotations is available [12].

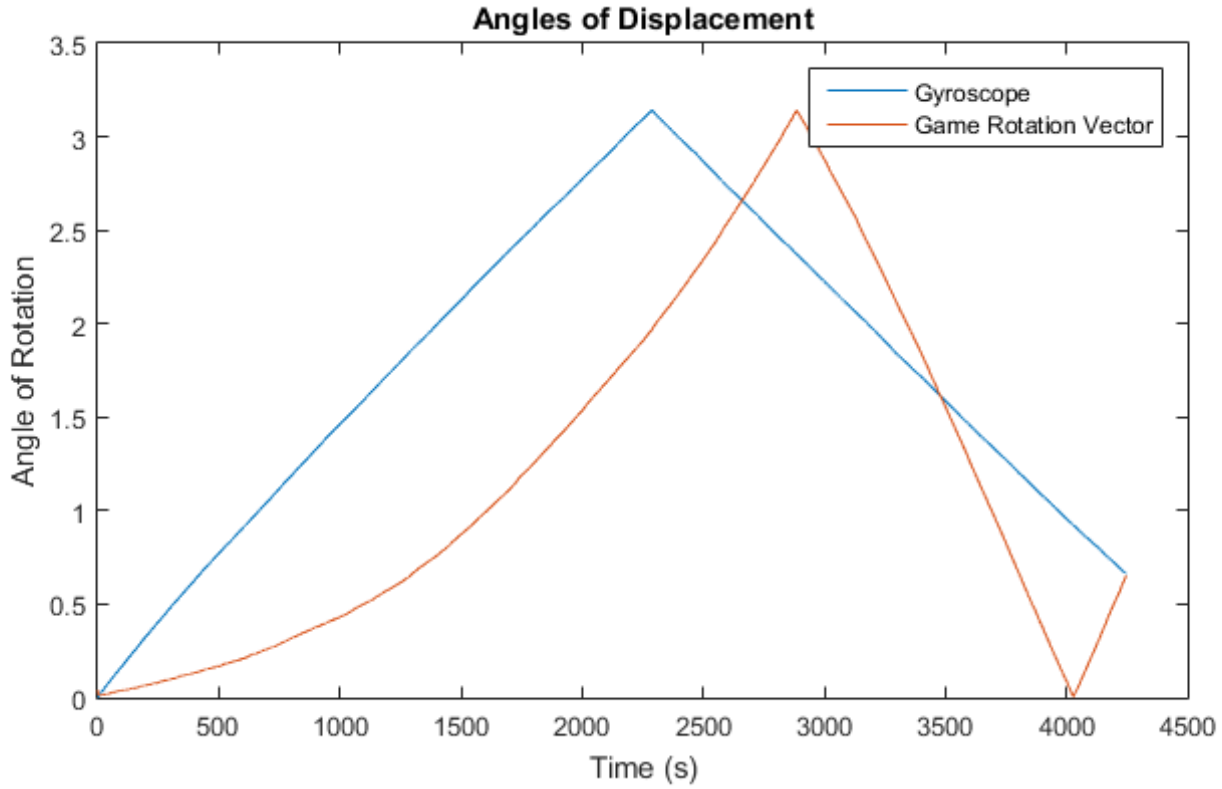

Figure 2.4-5. Angles of rotation for the gyroscope and game rotation vector sensor over a period of 70 minutes. Both methods drift significantly. The gyroscope drifts with a fairly constant rate, while the game rotation vector accelerates over time. That both methods end at the same angle is a coincidence and is inconsequential.

From this figure, we see that the two methods drift in quite different ways. The first thing to notice is that the gyroscope integration drifts with a constant rate while the game rotation vector sensor's drift accelerates over time. The game rotation vector sensor drifts less than the gyroscope integration during the first

1500 s, seems about equal at 2000 s, after which it becomes significantly worse and eventually catches up at the 3500 s mark. While accelerating drift is not good, it might be able to be circumvented by restarting the sensor every 10 minutes. To further investigate the drift, the rate of change in the displacement angle was plotted, seen in Figure 2.4-6. The data displayed is the absolute value of the rate, passed through a low-pass filter.

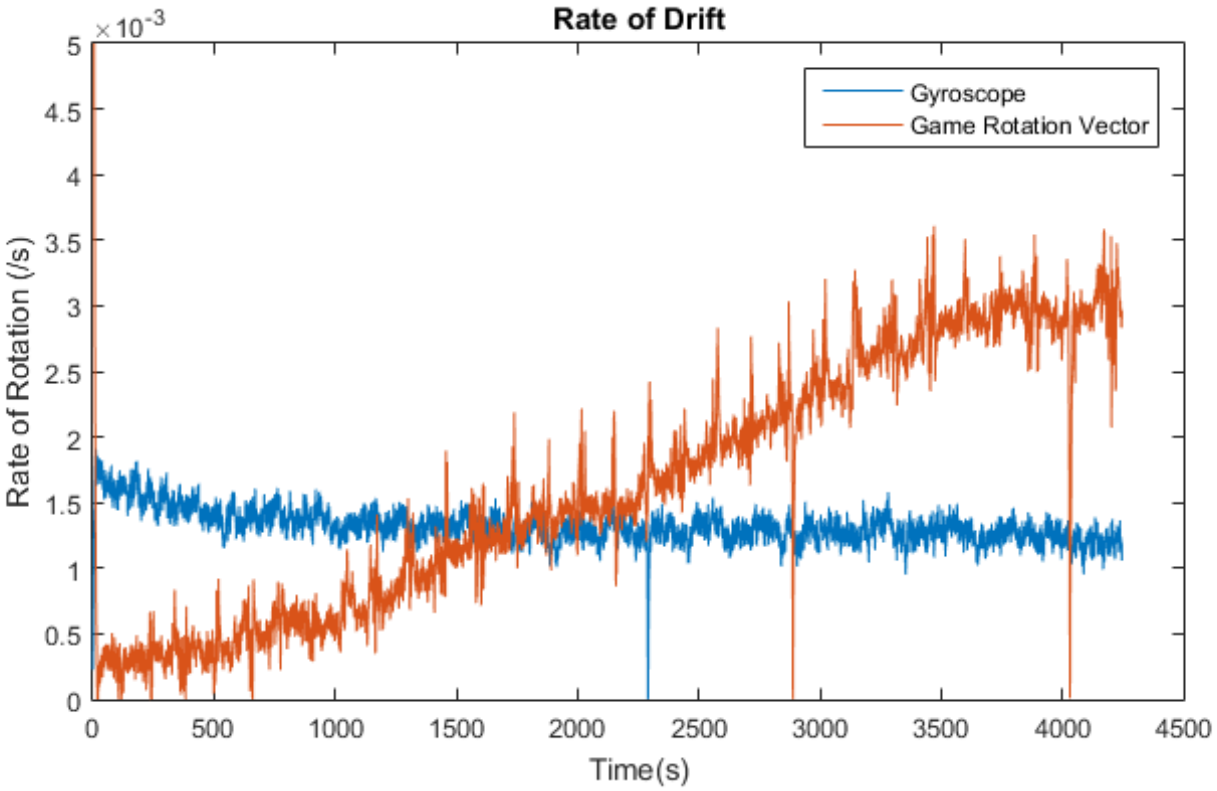

Figure 2.4-6. The graph shows rate of change in the displacement angles of the gyroscope and game rotation vector sensors during a stationary run of 70 minutes. The three vertical spikes are a filtering artifact that appears when the angle of displacement becomes zero or π. We see that the gyroscope drifts at a fairly constant rate (the small reduction likely comes from the deviation from a circular path), while the game rotation vector sensor drifts at a growing rate for most of the duration. We can also note that the game rotation vector has a much more uneven drift rate, with about 30 small spikes showing up at a rate of approximately one every 140 s.

We can immediately notice a general trend, where the gyroscope integration drifts at a more or less constant rate, while the game rotation vector sensor drifts with an ever increasing rate as well as showing large local variations. Looking at the video, we can see that much of the drift happening to the gyroscope integrated rotation is non-horizontal, and thus could be eliminated through sensor fusion with accelerometer data, possibly bringing the gyroscope down to the initial levels of the game rotation vector sensor.

We also notice regularly recurring noise spikes in the game rotation vector sensor's drift rate that are spaced out evenly with an approximate period of 140 s.

The drift rate of the gyroscope integration appears to be in the region of 5º per minute. Due to the time-varying nature of the game rotation vector sensor's drift, we can only conclude that it seems to be better than just the gyroscope over short time scales, but gets progressively worse as time goes on.

Since the intended application involves the device placed in a moving vehicle, the same two methods were applied to data gathered from the test described in 2.4.3, where the device was placed on the passenger seat of a car making a full rotation with a number of turns. A video showing the rotations is available [13].

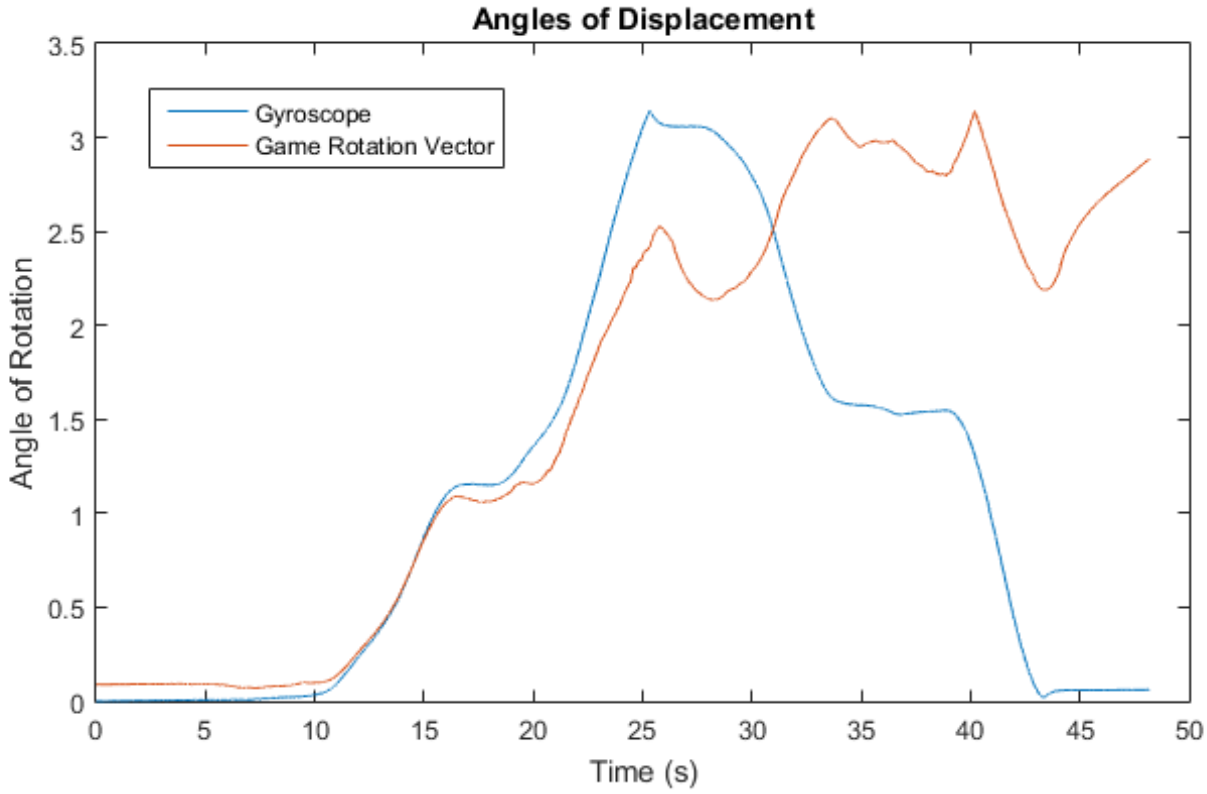

Figure 2.4-7. The graph shows the angles of displacement for the gyroscope integration and the game rotation vector sensor during a full 360º turn, made as a series of about 90º turns. The final orientation was the same as the starting one, within reasonable accuracy of parking a car. We see that the game rotation vector sensor falls behind quickly, and appears to attempt to counteract the rotation, clearly seen at times 25-27 s, 35-40 s, and 43-48 s.

The gyroscope seems to be performing quite well, with the error within the margin of error of parking a car. However, the game rotation vector sensor does not appear to be able to keep up at all, ending with an error of nearly 180º. While it is able to follow the rotations to a somewhat reasonable accuracy, it appears erroneously to detect some motion in the opposite direction immediately following an actual rotation. While an accurate pinpointing of the source of this problem would require further investigation, it appears likely that it is either a filter algorithm problem, which causes the filter to misinterpret accelerometer readings from linear acceleration of the car as changes in attitude, or an indirect

magnetometer dependency. As the official documentation of the game rotation vector sensor states that *it does not use the geomagnetic field* [5], this would suggest that the problem lies in filter misinterpretation of accelerometer readings.

Considering how both of the rotation vector sensors seem to have trouble handling vehicular motion, be it from static magnetic fields or from accelerometer misinterpretations, using the gyroscope with a custom filtering solution seems like the most promising alternative.

2.4.5 Gyroscope Calibration

Judging by the relatively constant rate of drift shown by the gyroscope seen in Figure 2.4-5 and Figure 2.4-6, there seems to be a fairly stable bias in the measurements. Visual inspection, by writing out the values directly to the display, however, shows that the bias varies over time around some more static value. To investigate this time-varying bias, the device was placed in a fixed upright orientation for several hours. The gyroscope measurements from this test are displayed in Figure 2.4-8, with each axis displayed separately and passed through a low-pass filter for readability.

We can see that the bias varies slightly over time, but with no clear pattern. The cause of this variation is unknown, but temperature dependencies seem likely.

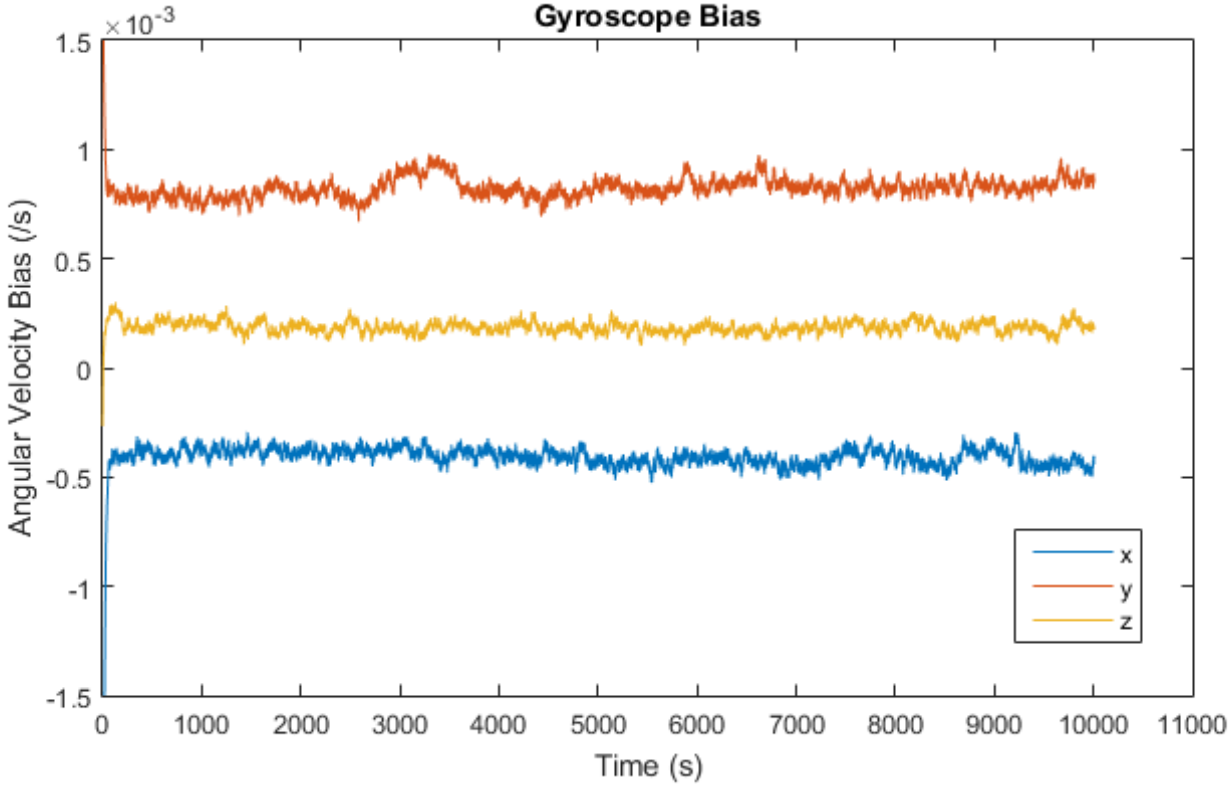

Figure 2.4-8. The graph shows low-pass filtered gyroscope readings with the device fixed in an upright position over about three hours. All three axes show readings that vary with time, but no clear pattern appears.

Using a model of the measured angular velocity for a single axis, $\omega^{¤}$, as:

$$\omega^{¤} = \omega + b_c, \tag{2.4:2}$$

where ω is the real angular velocity and b_c is the bias, we obtain the following approximation, ω^*, of ω:

$$\omega^* = \omega^{¤} - b_c. \tag{2.4:3}$$

From the experiment above, we obtain the following gyroscope bias:

$$\boldsymbol{b}_c \approx (-0.4057,\ \ 0.8227,\ \ 0.1864) \cdot 10^{-3}. \tag{2.4:4}$$

This, of course, only applies to the particular device used for testing in this project. Applying the calibration reduced the drift to near-zero levels, with typical drift rates of 0.1-0.4 degrees per minute. For comparison Earth spins at 0.25 degrees per minute.

For any larger-scale adaption one would most certainly want to use some on-line calibration algorithm, which also could help deal with the small bias changes that occur over time. A simple way to implement this is to add a slow high-pass filter to the gyroscope outputs, but in order to avoid filtering out relevant data it would have to be very slow, and thus not very responsive.

The need for on-line calibration became evident when applying the calibration to older data sets, collected weeks before the calibration was done. In most of the old data sets, the calibration did little or nothing to improve the accuracy. Examining the gyroscope data from these old tests suggests a completely different bias was present at the time. The later readings, on which the calibration is based, have been accurate since, but since there is no clear cause of why the bias changed, it is wise to assume it will happen again.

2.5 SENSOR CONCLUSIONS

After considering the results from experimenting with the sensors, using the gyroscope for rotation tracking with the accelerometer as a reference of the direction of gravity seemed like the solution most likely to bring good results within the time frame of the project.

As the magnetometer would require on-line calibration and possibly depend on the vehicle operators to do the initial self-calibration routine at startup, it was decided to only use map matching as an external reference of heading. However, the magnetometer remains promising and will likely be one focus of a future continuation of the project.

As dead reckoning suffers from large errors due to the double integration of accelerometer data, one would require additional hardware to obtain a reference of

speed or distance travelled. While the project is limited to only using the tablet, any such hardware would likely prove very beneficial in later continuations. However, due to the very sparse nature of an underground mine map, one may be able to track the position of a vehicle by comparing the tracked heading to that of adjacent map segments.

In general, the hardware present in the Samsung Tab S seems to provide fairly precise results that are unfortunately somewhat inconsistent over time. This is the case with the gyroscope bias investigated in 2.4.5 and accelerometer's apparent sensitivity to temperature, discussed in 2.4.2. These variations make it difficult to improve the gathered sensor information further, and on-line calibration algorithms may be necessary depending on the required level of precision.

3 FILTERING

This chapter covers a number of different possible filtering solutions for sensor fusion that are applicable to the orientation tracking problem. First, there is a listing of a few common approaches as well as their advantages and disadvantages. Second, there is a detailed description of the chosen algorithm, followed by results and comparisons with the qualities of the raw sensor data examined in 2.4. Ideally, the filtering algorithm should preserve the responsiveness of the gyroscope, while drifting less and always presenting an attitude that is as close to the real one as possible.

For completeness and future-proofing, magnetometer data will also be considered when comparing different algorithms.

3.1 ATTITUDE FILTERS

This section covers a couple of the most common ways to filter and fuse data from two or more sensors into an estimation of the attitude of a device. This is intended to provide a general overview of available algorithms without going into too much detail. The chosen algorithm is presented in detail in section 3.2.

The common goal of all sensor fusion algorithms is to take several different measurements of a quantity and combine them in a way that preserves the advantages of the individual measurements, and eliminates the disadvantages [6]. In the orientation case, we have three commonly available sensors, each of which possesses very different qualities. Both the accelerometer and the magnetometer *directly* measure the sought quantity. Neither of them can fully determine the orientation of the device, as they lack one degree of freedom - a magnetometer cannot detect the “roll” around the field lines, while an accelerometer cannot detect the “yaw” around the direction of gravity. Excluding the magnetic poles, the two can be used to uniquely determine the orientation of a device.

However, the accelerometer is very noise-sensitive by nature, as it not only measures gravity, but also the second derivative of the position. As such, any slight change in position, like car engine vibrations, will distort the gravity reading and throw the orientation off. The magnetometer, on the other hand, may pick up electromagnetic noise or be statically biased by local magnetic fields in the surroundings.

3.1.1 Kalman Filters

Kalman filtering, also known as *linear quadratic estimation*, is commonly used in many fields, and is one of the most reliable methods of combining several noisy measurements into a single estimate of the desired quantity. To produce an accurate estimation, Kalman filters take a lot of knowledge about the system into account, such as control-input, models of how the system is observed, knowledge about the covariance of noise in the different measurements, as well as how relevant physical laws apply to the system. This knowledge is encoded in a

number of matrices that are used in the calculation [14].

Kalman filters come in many variants, are generally technically quite complex and may require a good amount of knowledge about the system beforehand. As such, simpler solutions are often used even though a Kalman filter may give superior results.

3.1.2 Complementary Filters

A *complementary filter* does not, contrary to the Kalman filters, consider statistical descriptions for the noise in the measured signals, and as such is not expected to reach the same levels of accuracy in the general case [15]. However, their relative simplicity compared with the Kalman filters make them an attractive choice when pursuing the optimal accuracy is not necessary, as they are often much easier to implement and tweak. Thus, it can be a good idea to use a complementary filter in situations where the accuracy produced is *good enough*.

There does not appear to be much of a rigid definition of what a complementary filter is. However, the general idea is that two or more different measurements of a quantity are filtered independently and added together to form a better approximation of the quantity than any of the individual measurements. For example, a low pass filter on a signal with high frequency noise, and a high pass filter on a signal with a varying bias, can form a complementary filter so that the combined output better approximates the measured quantity than any of the individual measurements do.

3.2 THE MAHONY COMPLEMENTARY FILTER

For this project, a modified version of a complementary filter developed by Mahony et al. in 2008 [16], is used to track the orientation of the device. This section contains a brief overview of the original Mahony filter, followed by a note on some additions and modifications that were made, and finally some results and comparisons to the tests made in 2.4.

The filter was chosen as it was made for a similar application to that of this project, is fairly easy to implement, and immediately upon testing showed promising results. It also allows for easy future inclusion of magnetometer data.

Out of the three variations of his filter described by Mahony, the *Explicit Complementary Filter* is the one best suited for this project, as it is formulated directly in terms of the types of sensor readings gotten from inertial motion sensors. Using a simple *Riemann sum* for integration, the update equation of the current estimated orientation q becomes:

$$q_{n+1} = q_n + \frac{1}{2} q_n \cdot p(\boldsymbol{\omega}') \cdot \Delta t, \tag{3.2:1}$$

where $p(\boldsymbol{\omega}')$ is the *pure imaginary quaternion* by zero extension of the three-element vector $\boldsymbol{\omega}'$, the *filtered* angular velocity. $\boldsymbol{\omega}'$ is defined by:

$$\boldsymbol{\omega}' = \boldsymbol{\omega} + K_p \boldsymbol{e} - \boldsymbol{b}, \tag{3.2:2}$$

where K_p is a proportionality constant on the error $\boldsymbol{e}$, and $\boldsymbol{b}$ is an approximation of the gyroscope bias. The bias approximation is updated as:

$$\boldsymbol{b}_{n+1} = \boldsymbol{b}_n - K_i \boldsymbol{e} \cdot \Delta t, \tag{3.2:3}$$

where K_i is a constant describing the rate of integration of the error term $\boldsymbol{e}$. The filter thus has a built-in *PI regulator* driven by $\boldsymbol{e}$, defined by K_p and K_i. The error term $\boldsymbol{e}$ is based on a number of known *reference vectors* $\boldsymbol{v}_i$ in the world frame and the measurements of these vectors, $\hat{\boldsymbol{v}}_i$, according to:

$$\boldsymbol{e} = \sum_{\boldsymbol{i}} \frac{k_i}{2} (\boldsymbol{v}_i \times \hat{\boldsymbol{v}}_i), \tag{3.2:4}$$

where k_i is a weight parameter that allows tweaking of the correction speed for individual reference vectors. The value of k_i will depend on the rate of drift around the relevant axes, as well as the accuracy of the $\hat{\boldsymbol{v}}_i$ measurement. For example, in the case of a magnetometer reference, the error should be weighted high enough, so that horizontal drift is eliminated, but otherwise as low as possible to reduce the sensitivity to noise and local magnetic fields.

While, in theory, any number of reference vectors can be used [16], the most easily available ones are gravity and the geomagnetic field. The general formulation of the filter in terms of generic reference vectors make it easily expandable if any additional information becomes available, as no implementation details have to be changed to accommodate the new information. However, the usage of the filter output may have to be changed, as it contains more information (for example, the filter output would tend to align to the correct heading if a magnetometer reference vector is added).

If only a single reference vector is used, the output orientation will drift in one degree of freedom. If that reference vector is gravity, then the drift occurs in the horizontal plane, similar to the game rotation vector sensor, discussed in 2.2.4 and 2.4.4. This is the case with the implementation of the filter used for this project, and the characteristics of the drift are investigated further in section 3.2.3.

3.2.1 Mahony Filter Constants and Accelerometer Tolerance

To find decent values of the K_i and K_p parameters, a few different sets of data were passed through the filter for a number of values of the parameters. Having either value too high causes the filter to show similar effects as the game rotation vector sensor when significant linear accelerations are present, as the filter

misinterprets the horizontal accelerometer readings as the device having been tilted to some side. Any rotation measured while this artificial tilt is in place will not be correctly applied, and the final orientation often largely differs from the real one (due to the non-commutativity of rotations).

This effect can be minimized by reducing the values of K_i and K_p, but this also slows the filter's responsiveness to other errors. To find a better compromise, the value of the weighting factor k_a of the accelerometer reference vector is made to depend on the absolute value of the accelerometer according to:

$$k_a = \begin{cases} 1, & \big||\boldsymbol{a}| - g\big| < \alpha \\ 0, & \big||\boldsymbol{a}| - g\big| \geq \alpha \end{cases}, \tag{3.2:5}$$

where g is the net gravitational force on an object, 9.82, and α is the *tolerance range* of the accelerometer. This tolerance range allows filtering out of all extreme values of $\boldsymbol{a}$, and reduces the rotation error following artificial tilt from linear accelerations. While the average quality of the accelerometer reference measurements improve with lower α, lowering it too much makes the filter sensitive to small time-varying biases. For the device used in the project, $\alpha = 0.2$ is a good value using the calibration method detailed in 2.4.2. This typically lets more than half of the measured values through, and even in tests with a lot of linear acceleration about a fourth to a third of values pass through. While this may seem like a small amount, the acceleration information contained in the discarded readings essentially creates more error than it can be used to compensate for. That said, setting α too small can also be detrimental. For example, one should ensure α is large enough that pure gravitational readings are not pushed out of the tolerance range by noise or time-varying bias.

Introducing the tolerance criteria allows increasing the filter without introducing the kind of artificial tilt seen earlier. However, while a small increase may improve the responsiveness of the filter, increasing the filter constants far above what is required to eliminate drift may cause unexpected behavior in some rare edge cases, and is not advisable. $K_p = 1.00$ and $K_i = 0.01$ give good results with the device used in the project, and should be a good starting point for a similar setup.

3.2.2 Mahony Filter Responsiveness

To verify that the filter works as intended, it is compared against integrated gyroscope data, using the method presented in appendix 6.2, for a number of different data sets. First, the responsiveness is investigated to determine if the filter can keep up with rapid movements. For this, the *displacement angles*, defined as the angle θ in the orientation quaternion, as defined in equation ((6.1:10), are compared between the two methods. This comparison can be seen in Figure 3.2-1.

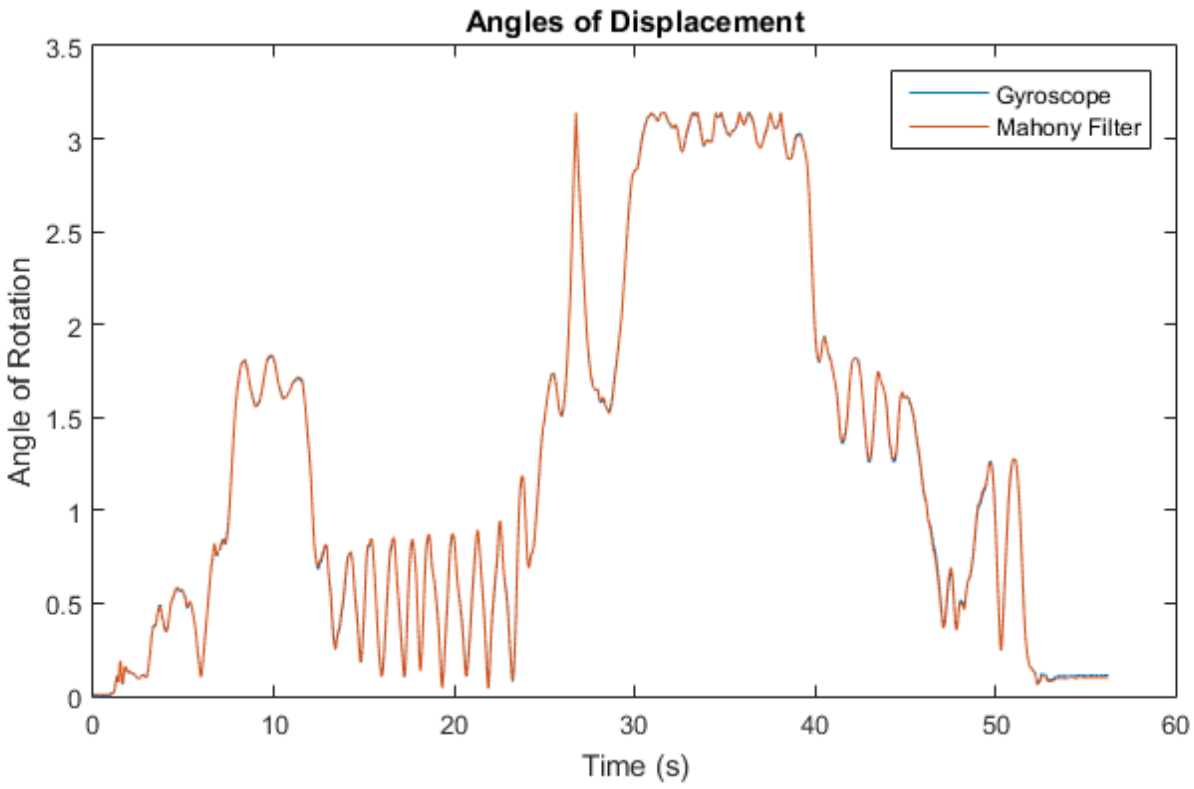

Figure 3.2-1. Displacement angles of the Mahony filter and regular gyroscope integration during fast motion. The two methods are mostly indistinguishable, and the final error is about 6º for both methods. The similarity shows that the filter has no problem tracking fast motion.

We can see that the integrated gyroscope output and the output of the Mahony filter are nearly identical, which suggests the filter has no problems with rapid motion. Note that both methods show a final error of approximately 6º. This is not ordinary drift (we will see further on that the Mahony filter drifts noticeably less than gyroscope integrated rotations), but rather some inaccuracies that appear during rapid movements. This error does *not* appear to be reliably reduced by introducing a high-pass filter on the gyroscope readings $\boldsymbol{\omega}$.

This type of error appears to emerge in a couple of different ways: First, whenever the gyroscope peaks at its maximum value which, for the test device, is about 8.6 rad/s. Rates of rotation exceeding this value can easily be reached while rotating the device by hand, but are non-existent or rare during normal operating conditions. If they do happen regularly, large errors are introduced, but the Mahony filter is able to correct the attitude, limiting the resulting error to one degree of freedom (heading). Secondly, the accuracy of the gyroscope seems to get slightly worse with higher rates of rotation. The exact reason for this is unknown, but it shows that the operational accuracy must be expected to be worse than shown during drift tests.

3.2.3 Mahony Filter Drift

Applying the Mahony filter to a number of drift tests shows that it is significantly better than the gyroscope by itself. This shows that the attitude correction by accelerometer measurements works as intended. Displacement angles for the Mahony filter applied to the same test used in Figure 2.4-5 is shown in Figure 3.2-2. A high pass filtered on $\boldsymbol{\omega}$ was added to try to prevent the heading drift, and it was made as responsive as possible without distorting short-term variations. The high-pass filter $\boldsymbol{hp}(\boldsymbol{\omega})$ at event n is described by:

$$\begin{gathered} \boldsymbol{hp}(\boldsymbol{\omega}) = \boldsymbol{\omega} - \boldsymbol{lp}(\boldsymbol{\omega}) \\ \boldsymbol{lp}_n(\boldsymbol{\omega}) = \alpha \cdot \boldsymbol{\omega} - (1-\alpha) \cdot \boldsymbol{lp}_{n-1}(\boldsymbol{\omega}) \\ \alpha = \frac{\Delta t}{RC + \Delta t} \end{gathered}, \qquad (3.2{:}6)$$

where $\boldsymbol{lp}(\boldsymbol{\omega})$ is a low-pass filter with a *smoothing factor* α, calculated with the *time constant* RC, which determines the responsiveness and inertia of the filter. This filter is very simple and not particularly effective, but it shows that high-pass filters can be used to improve the data.

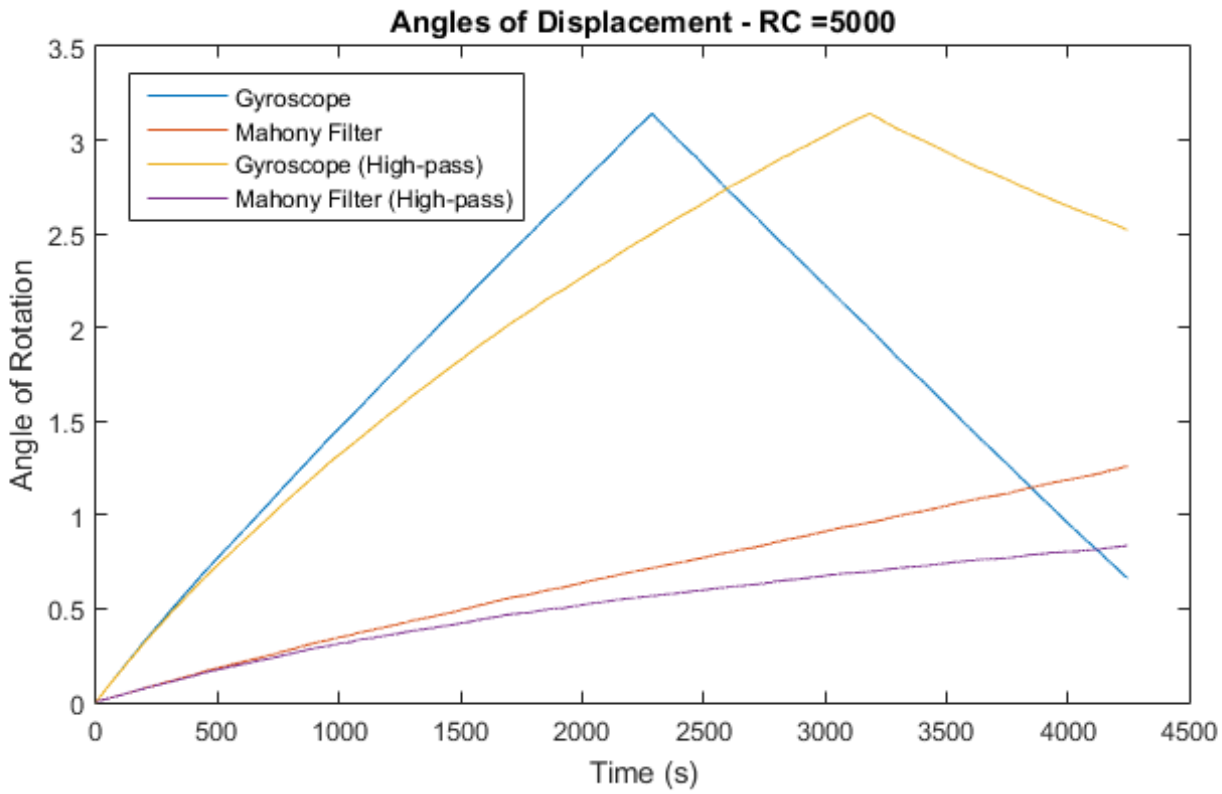

Figure 3.2-2. The graph shows displacement angles of gyroscope integration and the Mahony filter from a completely still device, with and without a high-pass filter on the gyroscope readings. The Mahony filter drifts far less, and the high-pass filter reduces the drift in both cases.

We can see that the Mahony filter drifts considerably less than the gyroscope on its own. This is not surprising, since the drift is limited to one of three degrees of freedom, but it is vastly better than the game rotation vector sensor, which uses a similar sensor fusion algorithm that causes accelerating drift, shown in Figure 2.4-5.

We can also see that the high-pass filter reduces the drift rate over time, but that the value of RC used in the test makes the high-pass filter very slow to respond. Decreasing RC improves this, but the filter may then begin to affect actual motion data. In fact, the filter may have to be slowed down even further to handle cases where a vehicle moves slowly along a curved path (an extreme case being a loaded truck driving up a spiral ramp for several minutes). This may be partially offset by using a filter with a more sharply defined cut-off frequency, which could allow for faster drift elimination. If available, magnetometer data is a better option as it allows for fast, responsive drift elimination without the downsides of high-pass filtering.

The average drift rate displayed by the Mahony filter in these tests is about 1º per minute, with the high-pass filter reducing it even further during prolonged use. This is close enough to zero to conclude that linear drift will not be the main contributor to the heading error, with it being an order of magnitude smaller than the more unpredictable errors, discussed in section 3.2.2, that arise during motion.

3.3 FILTER CONCLUSIONS

The Mahony complementary filter performs well in all of the tests, especially after introducing the accelerometer tolerance parameter, detailed in section 3.2.1. The filter is shown to keep up with the responsive gyroscope, while also being able to maintain a correct attitude in cases where gyroscope integration does not. The remaining drift is shown to be reduced by adding a bias-eliminating high-pass filter on the gyroscope readings. However, since the frequency range of vehicle motion may be very wide, the high-pass filter will likely have to be very slow, and magnetometer data would likely be strictly better for drift elimination. The Mahony filter appears to be strictly better than the game rotation vector sensor.

Without a deeper understanding of the system, it seems unlikely that a Kalman filter would perform much better than the Mahony filter. Still, Kalman filters should not be dismissed from further consideration.

The filter output is a quaternion describing the relative orientation of the device, but depending on the navigation algorithm, less information may be necessary. For example, one may want only to have the pitch and yaw angles. The yaw angle is easily obtained by setting the pitch and roll components of quaternion axis to zero and renormalizing the quaternion. Extracting θ from the normalized quaternion then gives the yaw angle.

3.3.1 Possible Future Improvements to the Filter

There are a few areas where improvements may result in improved filter performance. Having access to magnetometer data as a secondary reference vector would be of great help if a suitable on-line calibration, as mentioned in 2.4.3, is found. It could potentially bring the filter rotation from a relative one to an absolute one in the world frame. If sufficiently accurate, it would significantly simplify navigation algorithm further down the line, and it would be very easy to add, as it could be directly added as a reference vector in equation ((3.2:4), without any other modifications to the filter.

If magnetometer data is not available, implementing a more effective high-pass filter to use on the gyroscope data could help reduce horizontal drift. However, given the wide frequency range of vehicle motion, the high-pass filter would be slow to respond.

Ideally, one would like to have some sort of on-line calibration algorithm for the accelerometer data, as the bias tends to vary over time. This would enable a more restrictive tolerance range, possibly together with a noise-cancelling low-pass filter. Deviations in accelerometer readings cause applied rotations from the gyroscope to be misaligned, as the filter assumes the accelerometer reading is parallel to the z-axis in the world frame.

The method for temporal integration used is the simplest possible, and a better one may provide some improvements to the precision. This would be a low priority considering the high and fairly consistent polling rates of the sensors shown in 2.4.1, but may still make a difference.

By combining horizontal accelerometer data with yaw rotation from the gyroscope, one could approximate the speed of the vehicle as well as determine if it is moving forwards or backwards during a turn. This is only a rough estimation, but it could give some usable qualitative information about the vehicle’s motion, which would likely be useful for any navigation algorithm based on the filter output.

4 NAVIGATION

This chapter covers the problem of navigating in an underground mine. First, there is a summary of available in-data, also mentioning additional data sources that could become available in the future. Second, a navigation method given the set of data sources currently available is proposed, with a discussion of the limitations and possible solutions to problems that arise.

A brief introduction to inertial navigation systems is given in section 1.1.2.

4.1 AVAILABLE DATA

This section is a summary of the data sources available for navigation given the constraints set by the project in 1.2, as well as notable future sources of data that may become useful to a navigation solution. For each data source, it is detailed how they are acquired, and in what way they can be used.

In general, data sources are divided into two groups: *external* or *reference* data, and *relative* motion data. External data provides an accurate measure of some quantity, and ideally would be the only data required for a navigation application. However, external data is often not very precise, so relative motion data is used for smoothing and small-scale movements. By combining the two, one can have a precise and accurate approximation.

4.1.1 WLAN RSSI Position

In Boliden's underground mines, the primary external source of position data is through *received signal strength indication* (RSSI) of the WLAN network [1]. In the simplest (and current) case, this means approximating the position by the position of the access point with the strongest signal. This estimate, acquired via a HTTP request to a web server, is an accurate measurement with bad precision, as the range of an access point may exceed 200m (although in most cases the error will be less than 100m). Still, it limits the possible positions to a relatively small area of the mine, and enables the verification or dismissal of a position obtained through inertial navigation. When starting up a device within the mine, this is the only initial indication of position.

The precision of RSSI positioning can be improved by employing a trilateration algorithm to find a position estimate instead of just using the access point with the strongest signal. As of writing, this is under development by the provider of the current RSSI position.

The position estimate is delivered in the form of a *node* ID.

4.1.2 Map Information (Node Graph)

For each of the mines, there is a virtual representation called the *node graph*. This is essentially a simplified map of the mine, consisting of a large number of points,

or *nodes*. Each node consists of a set of 3D-coordinates as well as a list of adjacent nodes. This knowledge essentially reduces the possible positions from a large volume to a relatively small area defined by the inter-node connections. Also, since the underground roads in a mine are relatively narrow, the forward axis of a vehicle can often be approximated to be parallel to some straight line between two interconnected nodes in the graph. This map covers most of the mine area (the exception being actively excavated areas), and can be pre-loaded or obtained through a HTTP request over the wireless network. Apart from limiting the search area, the node graph can also serve as a meta-external reference in situations where the position is well known, by matching the changes in heading from inertial sensors to the angles of nearby node connections.

A useful possibility would be to match the slope of a road to slopes measured in the map, but this is likely to be problematic in certain parts of the mine where the node graph does not contain accurate slope information.

4.1.3 Orientation Filter (Mahony)

The orientation filter described in 3.2 gives us access to the orientation of the device relative to some initial starting rotation. As shown in Figure 3.2-1, the filter is quite accurate over shorter periods of time, which means it can be used for map matching by tracking short-term changes in orientation. It also allows easy access to the slope of the current location, allowing for further matching with the node graph.

The filter, in its current form, is essentially an external source of attitude data (in the sense that it is accurate), and a source of relative heading data. This leads to the heading being the last degree of freedom without any reference measurement. However, if magnetometer data is made available to the filter, it becomes an accurate source of complete orientation data.

4.1.4 Dead Reckoning

While the initial results from dead reckoning of accelerometer data, shown in 2.4.2, were discouraging, the available data could be used to some extent. The problem is a lack of reference data of position or speed, which causes the position estimate from dead reckoning to be able to diverge quickly. However, there is still the possibility of using dead reckoning on even shorter time scales to provide some reliable information to a navigation application.

4.2 PROPOSED METHOD

Considering the environment and the available data, a navigation algorithm was devised. The algorithm is based on matching the heading angle rotations to inter-node connections in the node graph. Since the initial position is only given with a precision of hundreds of meters, a particle filter is introduced. This will also help relieve issues around straight parts and intersections. This algorithm has not been tested in practice, and lacks some implementation details, but it is believed by the

author to be usable in a working application once implemented. The concepts of this algorithm are described below.

4.2.1 Internal Map Representation

The map is represented by the *node graph*, in which a large number of points are represented as *nodes*, detailed in 4.1.2. In the algorithm, the entire map is reworked into a set of *edges*, representing the connections between nodes. The primary properties of an edge are its length and the angle it makes with the y-axis of the map. This brings the map closer to the data sources we have available - the angle is closely related to the output of the Mahony filter, and the length is useful for determining the propagation length of WLAN signals (since the walls are solid rock, the Euclidean distance is not very useful). Nodes are also represented internally, but mostly as a way of accessing the edge network, specifically when a RSSI position estimate is returned as a node.

4.2.2 Vehicular Particle Filter

As the initial position estimate is very rough, the vehicle and device may be located anywhere within a fairly large area. For this reason, a particle filter is used, consisting of a number of virtual "*ghost*" vehicles distributed over the possible area. They are referred to as "ghosts" to differentiate them from actual physical vehicles. A ghost is defined by the edge it stands on and its heading, which is expressed as an offset from the Mahony filter's heading. This allows for all ghosts to take advantage of the Mahony filter, without having to repeat the intense computations of the filter.

When the first RSSI position is obtained, two ghosts are placed on every edge within WLAN range – one facing in each direction. When the filter heading changes, ghosts are moved to adjacent edges that match the new heading, and ghosts that make *impossible turns* are removed. Impossible turns can, for example, be when a left-hand turn is made and a ghost is in a right-hand curve. As the amount of ghosts shrink, we come closer to finding our position. Removal of ghosts can also happen when RSSI data is updated and ghosts are out of range or when ghosts "compress" into the same edge. Eventually we should have one or a few possible positions, which we then approximate our vehicle position by.

This differs from most particle filtering algorithms in that it attempts to converge to a single ghost as quickly as possible, where other algorithms maintain a swarm of particles that are resampled at each time step according to a probability distribution based on current data [17]. With our model, this could translate to placing new ghosts on edges adjacent to other occupied edges when ghosts are removed, or even placing several ghosts on the same edge, with each ghost having a slightly different angle. If a good speed estimate becomes available we can consider using a more conventional particle filter, as we become less tied to the discrete map description.

To detect impossible turns accurately, we will often need to have qualitative knowledge of the speed of the vehicle, as otherwise forward and backward motion can easily be confused (reversing is not uncommon when navigating an underground mine). As mentioned in 3.3.1, this information can be extracted by comparing the direction of rotation and the direction of sideways acceleration when making a turn. While no finished method of obtaining this data has been produced within the project, initial testing has produced encouraging results.

4.3 PROBLEMS AND POSSIBLE SOLUTIONS

As previously stated, lacking knowledge of the direction of motion presents problems when turning. Reversing into an exit on your left looks the same as turning into an exit on your right while going forward, when only considering the heading. This is immediately solved if (signed) speed data is made available, but could also be deducted from inertial measurements if the device placement within the vehicle is known.

Not having speed or distance data causes problems in certain situations, where the matching of angles does not produce reliable results. One such case is whenever a straight road segment is reached. Since all adjacent edges have the same heading in such a segment, it is impossible to determine what part of the road the vehicle is situated on. This is even more problematic if there are several intersections along the straight path, as we cannot discern which turn was taken when one is detected. The proposed solution to this is to create one ghost at every possible turn, whenever there are several indiscernible option, each ghost prohibited from continuing along the straight path. After a turn is made, RSSI data should quickly eliminate the false guesses, as taking a turn significantly changes the signal strengths of nearby access points, due to obstruction of line of sight.

If only map information is available as an external reference of heading, it becomes difficult to correct errors, as the only indication is deviations from the edges that the ghosts are matched to. Since the matching is based on the heading, it becomes circular in many cases, and very difficult to determine how much error has accumulated. This limits true heading references to specific road shapes, such as when the road makes an “S”-shape, where the transition from left-hand to right-hand turn can be used as an accurate position, and thus also heading, reference. While underground roads often move in snaky patterns, this could still cause failure in specific parts of the mine where such patterns are few. This problem further indicates that reliable magnetometer heading data would simplify the navigation task.

4.4 NAVIGATION DISCUSSION

Navigating an underground mine environment using only inertial sensors, map information, and RSSI position estimates is difficult. Not only are there several situations where many possible positions must be evaluated, but it also requires a circular error correction method for the heading. This is likely possible, but it

seems more productive to investigate a way to obtain useful magnetometer readings to produce one additional reference vector for the Mahony filter. This would not require any additional hardware, and could completely remove the need for the most difficult and possibly unstable part of the algorithm.

One thing that will further improve the accuracy of navigation is the addition of a trilateration algorithm tied to the RSSI positioning service. This could improve the worst-case performance of the algorithm, as the searchable area could be reduced. However, the same uncertainties that apply to the current RSSI positioning accuracy apply, such as directional shielding by the vehicle. It is currently not known how much the RSSI landscape varies between vehicle types and device placement within vehicles. This would have to be tested before drawing any conclusions about the viability of high-precision RSSI uses, which also include position syncing when passing below an access point (which in perfect conditions would show up as a RSSI maximum).

Lastly, having any sort of speed indication (perhaps combined with reliable magnetometer data) would open up a whole new range of possibilities. As the currently proposed algorithm is built without taking this additional information into account, access to it would warrant a completely new algorithm to be constructed. Access to speed data can possibly be further improved by filtering it along with accelerometer readings in a simple complementary filter.

Out of the three improvements listed, only the speed measurement requires additional hardware to implement, but is also the one most likely to greatly improve the accuracy and precision of the navigation task. Knowing the speed of the vehicle greatly reduces the amount of guesswork needed in the algorithm, and reduces the reliance on map information as more than a source of error correction. While the speed could theoretically be approximated from road vibrations, the varied road quality and texture in an underground mine make it unlikely to be applicable in practice. Instead, the most likely source of speed data is from the vehicle itself, such as from an *on-board-diagnostics* (OBD) outlet, transferred via Bluetooth to the tablet.

It may be the case that an improved RSSI positioning algorithm may render inertial navigation unnecessary for navigation in the inner parts of the mine, where several WLAN access points are typically visible. However, it can never exactly find the position of a device where many small tunnels branch off, as is often the case in actively excavated areas.

5 CONCLUSIONS

The sensors present in the Samsung Tab S leave some things to be desired. While the gyroscope is reasonably precise over shorter time frames, both the accelerometer and gyroscope show inconsistencies and time-varying bias, as shown in 2.4.2 and 2.4.5. However, both sensors are accurate enough to be useful in a navigation application, especially if the time-varying bias can be eliminated.

Dead reckoning of accelerometer data is not likely to be useful as a way of determining the velocity or position of the device, even if a more accurate accelerometer were to be used. This is an inherent difficulty that comes with double integration of noisy and/or inaccurate data. Dead reckoning is most likely to be useful for smoothing between reference points (for example RFID), and not as an actual position measurement.

The geomagnetic heading is likely to provide significant benefit to the navigation task, if isolated from the vehicle's static field, as discussed in 2.4.3. This requires on-line identification of the static bias, and will require an initial calibration period. Still, this is likely to produce more consistent results, especially when recovering from previous errors.

The Mahony complementary filter, presented in 3.2, tracks orientation well, and is easy to implement. The formulation of the filter in terms of general reference vectors makes it expandable, and the calculations are relatively lightweight. If complemented by on-line bias elimination of the accelerometer, as well as magnetometer readings, it should be able reliably to provide an orientation estimate with good precision.

The navigation algorithm based solely on inertial sensors, map information, and RSSI data could potentially improve the positioning over just having RSSI position estimates. However, one or more additional data sources (vehicle speed, geomagnetic field) are likely required for the navigation algorithm to reach high levels of accuracy and precision. Having a speed measurement will be the biggest game-changer for the navigation task, but the geomagnetic field will bring much needed stability and solve several troubling issues, discussed in 4.3.

5.1 AUTHOR'S RECOMMENDATIONS

The author's recommendation to Boliden is to first investigate how much the precision of the RSSI-based positioning system can be improved by introducing a trilateration algorithm, as this is likely to improve the precision of the current system without requiring any new hardware. Inertial navigation is most likely only necessary for precise positioning in active areas, where vehicles are located on the outside of the grid of access points.

For inertial navigation, isolating the geomagnetic field is the recommended next step, as it brings stability and eliminates much guesswork, without requiring additional hardware. In order to bring inertial navigation to its full potential, however, it is likely necessary to extract speed data from the vehicle.

6 APPENDIX

6.1 NUMERICAL ORIENTATION DESCRIPTIONS

There are many ways of describing the rotation or orientation of an object. In this appendix, axis-angle, Euler and Tait-Bryan angles, rotation matrices and rotation quaternions are described, and reasons for why quaternions were chosen for this project. All rotations mentioned here are considered around the origin of the coordinate system they are described in. Orientations are described as a rotation from a known starting orientation, usually defined as the intuitive "forward" direction object pointing along the y-axis of the coordinate system (such as a car or plane parked flat on the ground while facing north).

6.1.1 Axis-Angle

Euler's rotation theorem states that any displacement of a rigid body such that a point on the body remains fixed, is equivalent to a single rotation around some axis (the *Euler axis*) that runs through the fixed point. If we disregard translatory displacement by fixing the origin of our coordinate system to some point in the rigid body, this becomes equivalent to stating that any rotation of the body can be described by one rotation around some vector starting at the origin. Most often comprised of a scalar and a unit vector, this is called an *axis-angle* representation. While not often used as-is, axis-angle representations are closely tied to quaternions, which are briefly explained in 6.1.4.

6.1.2 Euler and Tait-Bryan Angles

When talking about aircraft, the *yaw, pitch and roll* system is often used, since it makes the orientation easy to visualize for a human, as well as being directly measureable from a gimbal. In this context, yaw is the compass heading, pitch is the angle between the horizontal plane, and roll is the angle of rotation of the aircraft around its own forward-pointing axis. Yaw, pitch and roll are a specific set of *Tait-Bryan angles*, which in turn are closely related to *Euler angles*, the difference being that Tait-Bryan angles use all three axes, while Euler angles use the same axis for the first and third rotation. Both representations can be used with either *intrinsic*, meaning the rotations occur about a coordinate system that rotates with each of the rotations, or *extrinsic* rotations, meaning rotations occur about the axes of a fixed coordinate system.

One downside with using Euler or Tait-Bryan angles is that there is a coordinate singularity in some situations. For example, when following the above definition of yaw, pitch and roll, the pitch is exactly 90° - in this case, roll and yaw is the same thing, and there is no unique way to describe the current orientation. This effect is commonly known as *gimbal lock*.

Even with the downsides considered, Euler and Tait-Bryan angles are very common representations of orientation, since they are easy to visualize and are the

least memory-intensive representation commonly in use, requiring only three numbers to describe any orientation.

6.1.3 Rotation Matrices

A *rotation matrix* is a representation of a rotation as a square, orthogonal matrix that has a lot of nice properties. Given a n-by-n rotation matrix R and a column vector v of length n, the rotated vector v' is simply calculated as

$$v' = Rv. \tag{6.1:1}$$

Consider a vector v, rotated first by R^1 and then by R^2. The rotated vector v' is then calculated as

$$v' = R^2(R^1 v) = (R^2 R^1)v = R^t v, \tag{6.1:2}$$

where $R^t = R^2 R^1$. Several rotations can as such be computed before applying them to the vector that is to be rotated, which is very useful in cases where the same set of rotations are to be applied to a large number of vectors. When multiplying matrices, one must regularly ortho-normalize the resulting matrix or risk ending up with improper or inaccurate rotations.

From $R^{-1}R = I$, we see that the inverse of a rotation matrix corresponds to the opposite rotation to that of the original matrix. Since rotation matrices are orthogonal, they satisfy

$$R^T = R^{-1}. \tag{6.1:3}$$

Since transposing is a much cheaper operation than inverting, this can save a lot of performance when a lot of inversions are needed.

Rotating a vector with a rotation matrix uses considerably less computations than Euler angles, but in turn the matrix uses three times more memory.

6.1.4 Unit Quaternions

Quaternions are an extension of the complex numbers to four dimensions that have a few properties that make them well suited to represent rotations. A quaternion q can be written as

$$q = a + bi + cj + dk, \tag{6.1:4}$$

commonly denoted $q = (a, b, c, d)$ with i, j, k defined by

$$i^2 = j^2 = k^2 = ijk = -1. \tag{6.1:5}$$

From this description, we can derive the possible products of i, j, k as

$$\begin{aligned} ij &= k, \quad ji = -k \\ jk &= i, \quad kj = -i \\ ki &= j, \quad ik = -j. \end{aligned} \tag{6.1:6}$$

Multiplication of quaternions is non-commutative and, with $q_1 = (a_1, b_1, c_1, d_1)$ and $q_2 = (a_2, b_2, c_2, d_2)$, defined as

$$\begin{aligned} q_1 \cdot q_2 = (&a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2, \\ &a_1b_2 + b_1a_2 + c_1d_2 - d_1c_2, \\ &a_1c_2 - b_1d_2 + c_1a_2 + d_1b_2, \\ &a_1d_2 + b_1c_2 - c_1b_2 + d_1a_2). \end{aligned} \tag{6.1:7}$$

Quaternions are often viewed as a scalar, a, and a vector, (b, c, d). If $a = 0$, it is a *pure imaginary* quaternion, and reversely, if $b = c = d = 0$, it is a *real* quaternion. The norm, $\|q\|$, of a quaternion is defined as

$$\|q\| = \sqrt{a^2 + b^2 + c^2 + d^2}. \tag{6.1:8}$$

and a *unit quaternion*, q_u, can be obtained by normalization:

$$q_u = \frac{q}{\|q\|}. \tag{6.1:9}$$

The quaternion representation of rotation is closely tied to the axis-angle representation discussed in 6.1.1. If the angle is θ and the axis is that which runs through the unit vector $u = (u_x, u_y, u_z)$, the corresponding rotation quaternion q is:

$$q = \left(\cos\left(\frac{\theta}{2}\right), u_x \sin\left(\frac{\theta}{2}\right), u_y \sin\left(\frac{\theta}{2}\right), u_z \sin\left(\frac{\theta}{2}\right)\right). \tag{6.1:10}$$

Explaining the reasons for why only half the angle is used require delving into hyper dimensional geometry, which is beyond the scope of this appendix. One simpler way to look at it is that the angle used is applied twice during the conjugation operation described in equation (6.1:12), once from the left and once from the right.

Similar to rotation matrices in equation (6.1:3), the inverse, q^{-1} of a unit quaternion $q = (a, b, c, d)$ is its *complex conjugate* q^*, which differs from q only by having the opposite sign on its vector part:

$$q^{-1} = q^* = (a, -b, -c, -d). \tag{6.1:11}$$

A unit quaternion q can be used to rotate a vector p (here represented as a pure imaginary quaternion $p = (0, p_1, p_2, p_3)$) by the *conjugation* operation

$$p^* = qpq^*, \tag{6.1:12}$$

p^* being the rotated vector and q^* being the conjugate of q.

Just like with rotation matrices, any number of rotation quaternions can be multiplied to precompute the resulting rotation. Consider a vector p rotated first by q_1 and then by q_2:

$$p^* = q_2(q_1 p q_1^*)q_2^* = q_2 q_1 p q_1^* q_2^* = (q_2 q_1)p(q_1^* q_2^*). \tag{6.1:13}$$

The rotation is equivalent to that of p by a precomputed quaternion $q_2 q_1$. The result should be normalized as often as performance allows in order to avoid improper or inaccurate rotations.

Extracting the rotation around one specific axis can be done by setting the remaining two to zero, and then normalizing the resulting quaternion.

Finally, a quaternion q_0 representing *no rotation* is defined as:

$$q_0 = (1, 0, 0, 0), \tag{6.1:14}$$

which represents a zero rotation ($\cos(0) = 1$) around the zero vector.

The advantages of quaternions as a representation of rotation are several, with the main downside being that most people are not as used to dealing with them as with matrices or Euler angles. Compared with rotation matrices, quaternions use less memory, are much easier to normalize, and are easy to interpolate, which is often the case in graphics applications where smooth object and camera movements are desirable. Also, multiplying quaternions is faster than multiplying matrices, which makes a big difference when combining a large number of small rotations. However, rotating a vector by a quaternion using the conjugation operation defined in equation (6.1:12) is more computationally expensive than rotation by a rotation matrix, which is a single matrix-vector multiplication. This should be taken into account when deciding on which representation to use.

6.2 GYROSCOPE INTEGRATION

This appendix details how raw angular velocity measurements from a 3-axis gyroscope are used to compute rotation quaternions and, in turn, how those quaternions are used to update a quaternion representing the orientation of the measurement device.

The gyroscope outputs data in the form $\omega = (\omega_x, \omega_y, \omega_z)$, where $\omega_x, \omega_y, \omega_z$ are the rates of rotation in radians around the x, y and z axes of the device. The *magnitude* of the rotation is the norm of ω:

$$\|\omega\| = \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2}. \tag{6.2:1}$$

The *axis of rotation*, u, is calculated by normalizing the gyroscope data ω by $\|\omega\|$:

$$u = \frac{\omega}{\|\omega\|}. \tag{6.2:2}$$

If the measurement was taken over a time period dt, the *angle of rotation*, θ, is approximated as $\theta = \|\omega\| \cdot dt$. The quaternion dq representing the rotation is:

$$dq = \left(\cos\left(\frac{\theta}{2}\right), u_x \sin\left(\frac{\theta}{2}\right), u_y \sin\left(\frac{\theta}{2}\right), u_z \sin\left(\frac{\theta}{2}\right)\right). \tag{6.2:3}$$

Given a quaternion q, representing the current orientation of the object, the new orientation q^* is calculated as:

$$q^* = dq \cdot q, \tag{6.2:4}$$

as per the definition of quaternion multiplication given in equation (6.1:7). If the measurement taken is the first one, the default orientation q_0 is set according to equation (6.1:14).

Note that since this orientation is described in the device's frame of reference, rotation of vectors has to be done by the conjugate, q^{-1}, of q. To obtain q in the world frame, one would rotate the angular velocity data, ω, by q in every time step of the calculation.

7 REFERENCES

[1] P. Burman, Interviewee, [Interview]. September 2015.

[2] O. J. Woodman, ”An introduction to inertial navigation,” 2007.

[3] L. De Nardis and G. Caso, "Overview on RSSI-based Positioning," [Online]. Available: http://www.diag.uniroma1.it/~querzoni/corsi_assets/1314/GreatIdeas/great_ideas_de_nardis_2.pdf. [Accessed 18 September 2015].

[4] Vieyra Software, "Physics Toolbox Sensor Suite," [Online]. Available: https://play.google.com/store/apps/details?id=com.chrystianvieyra.physicstoolboxsuite&hl=en. [Accessed 18 September 2015].

[5] Android, "Sensors Overview," [Online]. Available: https://developer.android.com/guide/topics/sensors/sensors_overview.html. [Accessed 18 September 2015].

[6] Google Tech Talks, "Sensor Fusion on Android Devices: A Revolution in Motion Processing," August 2010. [Online]. Available: https://www.youtube.com/watch?v=C7JQ7Rpwn2k.

[7] BMI055, "Bosch Sensortec," [Online]. Available: http://www.bosch-sensortec.com/de/homepage/products_3/6_axis_sensors_2/inertial_measurement_unit_1/bmi055_1/bmi055. [Accessed 18 September 2015].

[8] M. Pedley, ”Tilt Sensing Using a Three-Axis Accelerometer,” Freescale Semiconductor, Inc., 2013.

[9] T. Beravs, J. Podobnik and M. Munih, "Three-Axial Accelerometer Calibration Using Kalman Filter Covariance Matrix for Online Estimation of Optimal Sensor Orientation," *IEEE Transactions on Instrumentation and Measurement,* vol. 61, no. 9, pp. 2501-2511, 2012.

[10] W. Gander and G. H. S. R. Golub, "Least-Squares Fitting of Circles and Ellipses," *BIT Numerical Mathematics,* vol. 34, no. 4, pp. 558-578, 1994.

[11] "Spin Test," [Online]. Available: https://www.youtube.com/watch?v=A660HVfPpAE.

[12] "Drift Test," [Online]. Available: https://www.youtube.com/watch?v=sO3_JT-z0V4.

[13] "In-Vehicle Test," [Online]. Available: https://www.youtube.com/watch?v=-9C8Xx1uTIo.

[14] M. I. Riberio, ”Kalman and Extended Kalman Filters: Concept, Derivation and Properties,” Istitute for Systems and Robotics, IST, Lisbon, 2004.

[15] W. T. J. Higgins, "A Comparison of Complementary and Kalman Filtering," *IEEE Transactions on Aerospace and Electronic Systems,* vol. 11, no. 3, pp. 321-325, 1974.

[16] R. Mahony, T. Hamel and J.-M. Pflimlin, "Nonlinear Complementary Filters on the Special Orthogonal Group," *IEEE Transactions on Automatic Control,* vol. 53, no. 5, pp. 1203-1218, 2008.

[17] A. Doucet, N. d. Freitas och N. Gordon, ”An Introduction to Sequential Monte Carlo Methods,” i *Sequential Monte Carlo Methods in Practice*, Springer New York, 2001, pp. 3-14.