Sensor shield and fusion algorithm evaluation of DIY Attitude and Heading Reference Systems (AHRS)

Audio engineering project

Paul Berghold

Supervisor: Dipl.-Ing. BSc Daniel Rudrich

Graz, September 9, 2020

institut für elektronische musik und akustik

Abstract

Head orientation estimation is an important task for plausible perception of virtual acoustics. To gain a reliable 3D acoustic scene via headphone playback, tracking of the listener's head movements is a necessary feature. Micro Electro Mechanical Systems (MEMS) offer a cheap and sufficient possibility to track such movements, but lack in accuracy. This work evaluates state of the art sensor shields (BNO055, NXP 9DOF and MPU9250) and fusion algorithms for the design of an attitude and heading reference system (AHRS) on the Arduino platform. The orientation estimation of the sensors is compared to an optical tracking system (Optitrack Flex 13). It is shown that all sensors can achieve sufficient orientation estimation with the right fusion algorithm and proper calibration. Their performance differs in the robustness against magnetic distortions and gyroscope bias drift. Nevertheless the BNO sensor outperforms the others with its ease of use as it has an onboard sensor fusion.

Contents

1 Introduction

In 2016 binaural syntheses and 3D Audio applications hit the consumer market as companies like *Google* and *Facebook* enabled the possibility for spatial audio playback on platforms like *YouTube* and *Facebook360*. To gain a convincing 3D audio reproduction via headphone playback, tracking of the listener's head movements is necessary. Micro Electro Mechanical Systems (MEMS) offer a cheap and sufficient possibility to track such movements. Due to their low cost and small size MEMS can be assembled in nearly any device such as smart- and headphones. The disadvantage of MEMS is that they lack in accuracy and have high noise levels. To overcome these downsides different sensors and clever processing of their signals can be used. This is called an Attitude and Heading Reference System (AHRS). There are two typical setups used for this purpose, an Inertial Measurement Unit (IMU) and a Magnetic, Angular Rate and Gravity sensor (MARG). An IMU consists of a gyroscope and accelerometer, whereas a MARG uses an additional magnetometer. Each MEMS sensor represents a different three dimensional vector pointing either towards the earth's magnetic north pole, the direction of gravity or the derivation of an angular rotation. This results in a system with six degrees of freedom (6DOF) for an IMU and a system with nine degrees of freedom (9DOF) for a MARG sensor. A fusion algorithm is then used to combine the overdetermined system of three dimensional vectors to get an optimal estimation for the true attitude of the whole sensor shield.

The Extended Kalman filter [Kal60] became the standard choice for sensor fusion algorithms, but in 2011 Madgwick [MHV11] showed that for an AHRS the quaternion based complementary filter can compare with the performance of an orientation estimation by a Kalman filter. The quaternion based filter also works at a much lower computational complexity compared to the Kalman filter. Therefore it can be used with a low cost micro controller like the Adafruit Feather MO Bluefruit LE [Ada20] used in this work. All the processing can be done on the MO+ core of the Adafruit Feather controller. It also comes with a built in Bluetooh LE shield and a USB battery charger, enabling to build a low cost standalone wireless head tracker for orientation estimation. The new headtracker is based on the MrHeadTracker [RBR+17] presented in 2017.

This report evaluates three different Arduino based sensor shields (BNO055, NXP 9-DOF and MPU9250) [Sen16] [Sem17] [Sem15] [Inv16] and two quaternion based sensor fusion algorithms [MHV11], [VDX15] in six and nine degrees of freedom for an AHRS. The attitude estimations are compared with each other and with an optical tracking system as reference (Optitrack Flex 13). All derivations of the used fusion algorithms are quaternion based. This ensures a unique representation of a three dimensional rotation trough its interpretation in a four dimensional hyperplane and avoids ambiguities which in the euler representation often causes problems.

One of the most important tasks in an AHRS system is sensor calibration. Especially magnetometers suffer from strong errors due to influences of unwanted magnetic flux e.g., headphone magnets or ferromagnetic materials such as cables, housing, or even metallic parts in nearby furniture. Originally these algorithms [VDX15] were not designed for head tracking, but come from Micro Aerial Vehicles (MAVs). Orientation estimation using low cost sensors is basically the same task for head tracking and MAVs, but in air travel there is simple fewer unwanted magnetic flux. This difference has to be regarded separately.

2 Background theory

Section 2 is a short introduction into the principles of quaternion based mathematics and sensor calibration. The experienced reader can skip this section.

2.1 Quaternion Representation

Quaternions are a four dimensional number system which can be used for the representation of three-dimensional rotations. The advantages to other representations is that quaternions are unambiguous to the means of rotations. They are mathematically represented in the form:

$$q = a + bi + cj + dk \tag{1}$$

with,

$$i \times i = j \times j = k \times k = i \times j \times k = -1 \tag{2}$$

and

$$\begin{aligned} i \times j = k, &\quad \textit{and} \quad j \times i &= -k \\ j \times k = i, &\quad \textit{and} \quad k \times j &= -i \\ k \times i = j, &\quad \textit{and} \quad i \times k &= -j. \end{aligned} \tag{3}$$

Where *a, b, c, d* are real numbers and *i, j, k* are fundamental quaternion units pointing towards the three spatial axes. This means that the set of quaternions span a 4 dimensional vector space over the real numbers, with *1, i, j, k* as a basis. For simpler notation we choose a quaternion representation which only shows the real number parts of a quaternion as a normalised unit quaternion q with $||q|| = 1$.

An arbitrary rotation of a rigid body in 3D space from observation frame A to frame B is therefore denoted as:

$$^B_A q = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T = \begin{bmatrix} cos\frac{\alpha}{2} & \vec{e_x} sin\frac{\alpha}{2} & \vec{e_y} sin\frac{\alpha}{2} & \vec{e_z} sin\frac{\alpha}{2} \end{bmatrix}^T \tag{4}$$

Quaternions express a rotation as a rotation angle α about a rotation axis $\vec{e}$. One can think of rotating the whole coordinate system fixed to the rigid body in observation frame A (before the rotation) around the axis $\vec{e}$, resulting in a new orientation in observation frame B. The now rotated coordinate systems in frame A and the observation of frame B become the same. Such a rotation is shown in Figure 1.

The inverse of a unit quaternion, for the rotation from frame B to frame A is defined by,

$$(^B_A q)^{-1} = {}^A_B q = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \end{bmatrix}^T \tag{5}$$

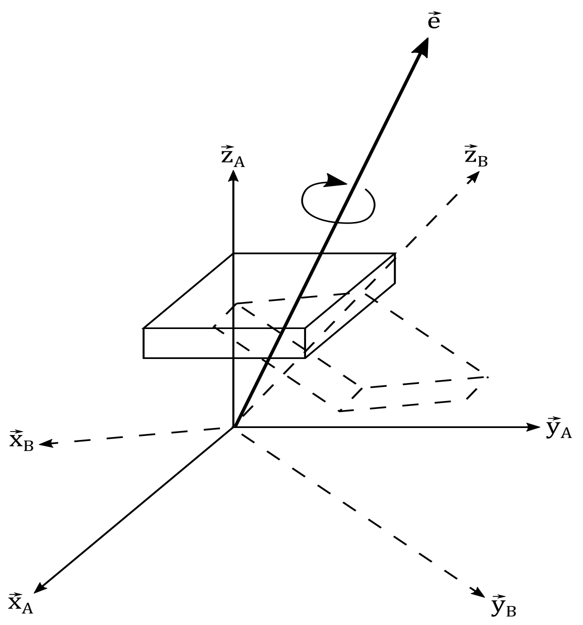


Figure 1 – Quaternion rotation ${}^B_A q$ of a rigid body (sensor) from frame A to B

and represents the rotation in inverse direction around the inverse rotation axis $-\vec{e}$. With the quaternion representation a sequence of rotations can be described by simply multiplying the quaternions from frame A to B and B to C.

$${}^C_A q = {}^C_B q \otimes {}^B_A q \tag{6}$$

2.1.1 Multiplication

If p represents the rotation from frame B to C and q represents the rotation from frame A to B, their multiplication is defined by:

$${}^C_B p \otimes {}^B_A q = p \otimes q = \begin{bmatrix} p_0q_0 - p_1q_1 - p_2q_2 - p_3q_3 \\ p_0q_1 + p_1q_0 + p_2q_3 - p_3q_2 \\ p_0q_2 - p_1q_3 + p_2q_0 + p_3q_1 \\ p_0q_3 + p_1q_2 - p_2q_1 + p_3q_0 \end{bmatrix} \tag{7}$$

It is important to mention that quaternion multiplication is a non-commutative operation, therefore the order of multiplication changes the orientation.

$${}^C_B q \otimes {}^B_A q \neq {}^B_A q \otimes {}^C_B q. \tag{8}$$

2.1.2 Matrix notation

Another useful notation is the quaternion rotation ${}^B_A q$ in $3x3$ matrix form $R({}^B_A q)$ defined as:

$$R({}^B_A q) = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 - q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{9}$$

2.1.3 Vectors in quaternion representation

As mentioned in section 1 all sensors generate a three dimensional vector as output, but the fusion algorithm works in quaternion domain. Therefore we need a vector $\vec{v}$ in quaternion form which can be expressed as a unit quaternion v_q like:

$$v_q = \begin{bmatrix} 0 & \vec{v} \end{bmatrix}^T = \begin{bmatrix} 0 & v_x & v_y & v_z \end{bmatrix}^T \tag{10}$$

With this representation unit quaternions can be applied to operate rotations for a 3D vector from the observations in frame A to frame B,

$${}^{B}\vec{v}_q = {}^{B}_{A}q \otimes {}^{A}\vec{v}_q \otimes {}^{B}_{A}q^{-1} \tag{11}$$

where the superscripts correspond to the reference frame of the vectors and $\otimes$ is the quaternion multiplication.

2.2 IMU and MARG

As mentioned in the introduction, an internal measurement unit (IMU) as well as a Magnetic, Angular Rate and Gravity sensor (MARG) consist of multiple sensors which measure the orientation of a rigid body. They comprise magnetometers, accelerometers and gyroscopes. Magnetometers measure the local magnetic field of the earth and output a vector $\boldsymbol{m}$ which points to the magnetic north as a reference. Similarly accelerometers measure the force of gravity and use it as a reference. The output data is a vector $\boldsymbol{a}$ which points towards the normal of the surface of earth. Lastly gyroscopes measure the angular velocity $\boldsymbol{\omega}$ of a rigid body which can be integrated to get the change of orientation over time. In theory one could only use a gyroscope to calculate the attitude of the sensor[1], but due to requirements of cheap and small sensors, like the MEMS, they suffer from high noise levels and insufficient calibration. The solution to this problem is combining different sensors and use a sensor fusion algorithm. For this there are filters based on statistical models like the Kalman filter or simple deterministic ones like the complementary filter and the gradient descent algorithm, which are explained later in section 2.3.

2.2.1 Magnetometer

As mentioned above, magnetometers measure the magnetic field of the earth and point to the direction of north. In comparison to a normal compass magnetometers incorporate a third dimension in the measurement of magnetic flux. The local magnetic flux ${}^{S}\boldsymbol{m}$ at the sensor frame and the true earth magnetic field ${}^{E}\boldsymbol{h}$ are defined in the same way Valenti,Dryanovski and Xiao did in their work [VDX15]. A detailed explanation on how their algorithm works can be found later in section 3:

$${}^{S}\boldsymbol{m} = [m_x m_y m_z]^T \text{, with } ||\boldsymbol{m}|| = 1 \tag{12}$$

1. If all initial conditions of the sensor are known.

$$^{E}\boldsymbol{h} = [h_x h_y h_z]^T \text{ , with } ||\boldsymbol{h}|| = 1 \tag{13}$$

For the later derivation of the fusion algorithms all vectors are normalised to unit length. The measured quantity is micro tesla (mT). One gross disadvantage of magnetometers is theneed for proper calibration, since magnetic flux can be distorted by any metallic object, cable wires or even the power current used by the sensor shield. More on how calibration works can be found in section 2.4.

2.2.2 Accelerometer

Next, accelerometers measure the earth gravitation in $\frac{m}{s^2}$ as the proper acceleration to its resting frame. The schematic setup of an accelerometer is a mass, attached to springs in all three dimensions, mounted in a closed reference case. Yet it is to keep in mind that the actual sensor is a MEMS. Deducted from the schematics, an accelerometer will measure $9.81\frac{m}{s^2} = 1g$ if positioned on the surface of earth. In free fall it will measure zero acceleration. Similarly to the magnetometer we define the measured acceleration in the sensor frame $^{S}\boldsymbol{a}$ as a unit vector

$$^{S}\boldsymbol{a} = [a_x a_y a_z]^T \text{ , with } ||\boldsymbol{a}|| = 1 \tag{14}$$

and the true earth gravitational acceleration $^{E}\boldsymbol{g}$ like:

$$^{E}\boldsymbol{g} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \tag{15}$$

Accelerometers offer a solid reference for the orientation of the sensor frame. In the most fusion algorithms they are used as a first step to rotate the sensor frame to a known orientation in one dimension, parallel to the earth's surface. The output signal tends to be corrupted by high-frequency noise, but if needed it can be smoothed with a low-pass filter.

2.2.3 Gyroscope

Finally, gyroscopes measure the change of angular velocity $^{S}\omega$ around the three sensor frame axis in rad/s. Then the output is integrated to get the orientation of the sensor. The underlying physical principle is that a vibrating object tends to continue vibrating in the same plane even if its support rotates. The Coriolis effect causes the object to exert a force on its support, and by measuring this force the rate of rotation can be determined. The angular velocity is also defined as:

$$^{S}\omega = [\omega_x \omega_y \omega_z]^T \tag{16}$$

Although a single gyroscope sensor would be able to measure the orientation of the sensor frame, given it's initial attitude, gyroscopes produce large errors due to the MEMS technology. Further errors propagate through the integration, resulting in a drift over time. Consequently a gyroscope only shows sufficient approximation of orientation over very short time but suffer a drift in the heading direction over long time, resulting in an insufficient orientation estimation. This is the main reason for the need of sensor fusion algorithms.

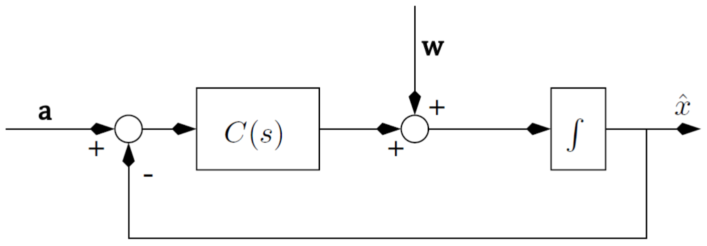


Figure 2 – Block diagram of a classical complimentary filter adapted from [MHP05]

2.3 Sensor fusion

Sensor fusion utilizes the idea of combining different sensors and therefore minimizing the particular disadvantages of each.

2.3.1 Complementary filter

The simplest approach is a complementarity filter which uses an analysis of the different sensor signals in the frequency domain and later combining them to obtain a better estimation of a particular quantity. In the case of an AHRS the output data of a gyroscope and an accelerometer are used for fusion. The measured gyroscope data has a growing low frequency drift due to high noise levels and the necessary integration process, which additional adds up the error over time. A high pass filter (HP) can improve the signal quality and therefore the whole estimation. Accelerometers are sensitive against linear acceleration and the output can be improved by using a low pass filter (LP) to smoothen the signal. Finally both signals get fused together forming an all pass filter with a constant magnitude,

$$LP(s) + HP(s) = 1 \tag{17}$$

ensuring that the cross over frequency of the LP matches the cross over frequency of the HP filter. A simple complimentary filter can be designed by classical control system techniques like a proportional (P) and or proportional integrating (PI) controller. Figure2 shows a blockdiagram of such a filter. The corresponding system equation is given by

$$\hat{\boldsymbol{x}}(s) = \frac{C(s)}{s + C(s)}\boldsymbol{a}(s) + \frac{s}{C(s) + s}\frac{\boldsymbol{\omega}(s)}{s} \tag{18}$$

with $\hat{\boldsymbol{x}}$ being the estimation of the orientation, $C(s) = k$ for a proportional gain and $C(s) = k + \frac{1}{\lambda}$ for a PI controller. The proportional gain k can be tuned depending on the signals' frequency characteristics.

2.3.2 Kalman filter

Another approach is a filter based on a statistical model of the input signal, the Kalman filter (KF). It has been shown in many publications [Suh10], [MXB$^+$01], [VDX15] that the

Kalman filter is the benchmark for orientation estimation and sensor fusion algorithms. The following fusion algorithm example of an AHRS was done in the work from Feng, Li and Liu in [FLX$^+$17], but the mathematical theory behind is the same for every Kalman filter. Therefore, it should give a proper idea of what an KF could look like and show the complexity behind it. The first step for a Kalman filter design is to define the process model, in particular the state vector and state equation.

$$\boldsymbol{X}_k = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}^T \tag{19}$$

$$\dot{\boldsymbol{q}} = \boldsymbol{\Omega}(\omega)\boldsymbol{q} \tag{20}$$

where

$$\boldsymbol{\Omega}(\omega) = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \tag{21}$$

Next the observation model needs to be defined:

$$\boldsymbol{Z}_k = \begin{bmatrix} {}^c q_0 & {}^c q_1 & {}^c q_2 & {}^c q_3 \end{bmatrix}^T \tag{22}$$

The superscript ^{c} denotes a calculated quaternion from the accelerometer and magnetometer data. Additionally the process noise covariance matrix $\boldsymbol{Q}_k$ and the measurement noise covariance matrix $\boldsymbol{R}_k$ need to be calculated. Finally, the steps of the Kalman filter fusion can be calculated.

The computation starts with some initial conditions:

$$\hat{\boldsymbol{X}}_0 = \mathbb{E}(\boldsymbol{X}_0) \tag{23}$$

$$\boldsymbol{P}_0 = \mathbb{E}((\boldsymbol{X}_0 - \hat{\boldsymbol{X}}_0)(\boldsymbol{X}_0 - \hat{\boldsymbol{X}}_0)^T) \tag{24}$$

The next step is to project the state and covariance estimates from time step $k-1$ to step k:

$$\hat{\boldsymbol{X}}_{k+1}^- = \mathbb{E}(\boldsymbol{\Omega}_k \Delta T)\boldsymbol{X}_k \tag{25}$$

$$\boldsymbol{P}_{k+1}^- = \mathbb{E}(\boldsymbol{\Omega}_k \Delta T)\boldsymbol{P}_k \mathbb{E}(\boldsymbol{\Omega}_k \Delta T)^T + \boldsymbol{Q}_k \tag{26}$$

with the superscript $^-$ denoting the projection and ΔT indicating the discrete time. Then, the Kalman gain is calculated as:

$$K_{k+1} = \boldsymbol{P}_{k+1}^-(\boldsymbol{P}_{k+1}^- + \boldsymbol{R}_{k+1})^{-1} \tag{27}$$

The final step is to obtain the posterior error covariance estimate:

$$\hat{\boldsymbol{X}}_{k+1} = \hat{\boldsymbol{X}}_{k+1}^- + K_{k+1}(\boldsymbol{Z}_{k+1} - \hat{\boldsymbol{X}}_{k+1}^-) \tag{28}$$

$$\boldsymbol{P}_{k+1} = (\boldsymbol{I} - K_{k+1})\boldsymbol{P}_{k+1}^- \tag{29}$$

where $\boldsymbol{Z}_{k+1}$ is the computed quaternion estimation like in equation 22. From the process of the Kalman filter mentioned above, we can obtain the optimal estimated quaternion and finally calculate the 3D attitude of the body. Despite all advantages of the Kalman filter, it is very computationally complex and still susceptible to bad magnetometer calibration like the other fusion algorithms. Moreover it was shown in [VDX15] that fusion algorithms based on deterministic approaches match up to the performance of Kalman filters.

2.3.3 Gradient descent

Many optimization algorithms exist but the gradient descent algorithm is one of the simplest to implement and compute. It is a first order iterative optimization algorithm for finding the local minimum of an error function $\boldsymbol{f}({}^{E}\boldsymbol{q}_S - {}^{E}\hat{\boldsymbol{q}}_S)$ and then taking a step towards the negative error function curvature, finding an optimal solution. Despite a random starting orientation, the algorithm converges after a few iterations towards the minimum and remains a sufficient estimation. Therefore final implementations of the fusion algorithms on the M0+ chip are done as a gradient descent algorithm as shown from Madgwick in his internal report related to [MHV11] and [VDX15].

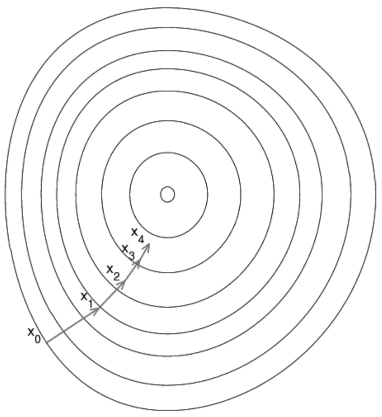


Figure 3 – Illustration of gradient descent on a series of level sets, graphic from [?]

2.4 Magnetic calibration

Magnetometers need to be calibrated after final mounting for their planned application, because until then, wide variations in accuracy can occur. In particular even mounting the sensor e.g., on a pc board can add physical stresses that can easily result in a shift of calibration. Investigations into the effect of magnetic distortions on an orientation sensor's performance have shown that substantial errors may be introduced by sources including electrical appliances, metal furniture and metal structures within a buildings construction [?]. Also declination errors, those in the horizontal plane relative to the earth's surface can become a problem and cannot be corrected without an additional reference of heading. Inclination errors, those in the vertical plane relative to the earth's surface, may be compensated for as the accelerometer provides an additional measurement of the sensor's attitude.

For calibration, the ideal response surface for a three-axis magnetometer is a sphere centered at the 3D origin. The response surface is the projection of the measured magnetometer data points to the three dimensional room. Insufficient calibration means that sensor data projected to either the xy-, xz- or yz-plane results in a point cloud which has either an offset to the origin or the edge of the cloud is not a perfect circle, but it is skewed. Calibration can be split in two steps, the hard and soft iron error compensation. Note that only distortion sources which are fixed relative to the rigid body can be compensated. This is the same for soft and hard iron errors.

2.4.1 Hard iron error

Hard iron errors represent magnetic field sources, which add or subtract to the earth's magnetic field measurements in the sensor. Examples of this type of error are with permanent magnets or power supply currents. Like shown in Figure 4, using the polar plot representation of the xy plane, hard iron errors that remain fixed to the sensor result in a shift to the origin of the response surface. The hard iron errors can easily be fixed by adding a constant offset to the respective axis.

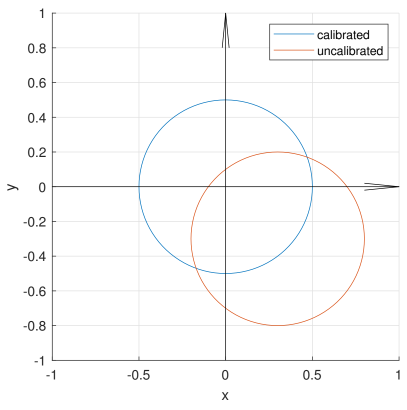


Figure 4 – x-y magnetometer plot with hard-iron errors

2.4.2 Soft iron error

Soft iron errors represent the magnitude and direction change that the earth's magnetic field experiences when near ferromagnetic objects. Further soft iron distortion is the result of material that influences a magnetic field but doesn't generate one itself. Figure 5 shows a polar plot representation of an soft iron error, which results in a skew, elliptic response surface (edge) in the xy-plane. The distortion produced by soft-iron materials is dependent upon the orientation of the material relative to the sensor and the magnetic field. Thus, soft-iron distortion cannot be compensated with a simple constant. Instead a more complicated procedure is required.

2.4.3 Error compensation

In summation, magnetic error compensation for an AHRS breaks down to calculating a rotation matrix $\boldsymbol{S}$ shown in equation 30, scaling it along the main diagonal and adding an offset vector $\boldsymbol{H}$ to the measured magnetic field $\boldsymbol{M}$. It is important to note that if

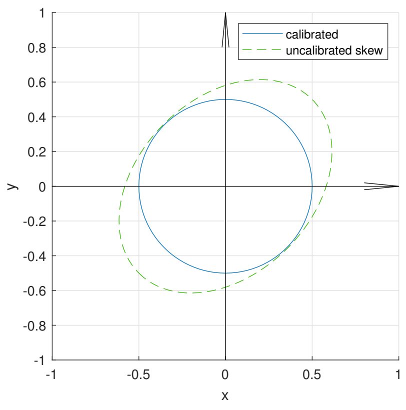


Figure 5 – x-y magnetometer plot with soft-iron errors

hard-iron effects are present, compensation for these distortions must be applied prior to correcting for soft-iron distortions. The calculation of the rotation matrix in 3D space is not scope of this work. The software used for this is $MotionCalApplication$ designed by Paul Stoffregen and can be downloaded here[2]. Further information about rotation matrix calculation in the 3D space can be found in Kupiers book [?].

$$\begin{bmatrix} \hat{M}_x \\ \hat{M}_y \\ \hat{M}_z \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} & S_{13} \\ S_{21} & S_{22} & S_{23} \\ S_{31} & S_{32} & S_{33} \end{bmatrix} \times \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} + \begin{bmatrix} H_x \\ H_y \\ H_z \end{bmatrix} \tag{30}$$

The $\hat{}$ symbol in equation 30 indicates the updated and error compensated magnetic sensor data.

2.5 Orientation

In general there are six degrees of freedom for a rigid body moving in three-dimensional space. Three of them describe the relative position of the moving object, left-right, up-down, back and front. The other three describe the rotation. The result of the head-tracker performance will be split into three rotation components around their respective axis. These 3 rotations are normally defined as yaw, pitch and roll like shown in Figure 6.

3 Fusion algorithms

As mentioned above, sensor fusion utilizes the idea of combining different sensors, and therefore, minimizing the particular disadvantages of each. This work compares the sensor fusion algorithm developed from Sebastian Madgwick [MHV11], which is based

2. `https://github.com/PaulStoffregen/MotionCal.git`

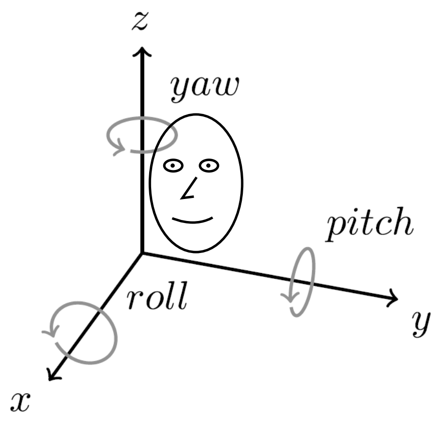


Figure 6 – Head orientation in yaw, pitch and roll

on the complimentary filter from Robert Mahony [MHP05], with the fusion algorithm designed by Valenti, Dryanovski and Xiao [VDX15]. The first approach is based on a first order gradient decent algorithm, the second is a closed form solution and both are implemented in 6DOF (without magnetometer data) and 9DOF (with magnetometer data). They address the same problem, the gyroscope drift, but differ in the processing of the measurement data for the correction step.

3.1 Madgwick fusion algorithm

This section points out the most important derivation steps for Madgwicks fusion algorithm. In this work from now on it is referred to as the complementary filter. For a detailed explanation how the algorithm works one should read section 3. of Madgwicks internal report[3] or [MHV11].

The complementary filter consists of two main parts, first the orientation from angular rates measured from gyroscope data and second the orientation from vectors measured from the accelerometer and magnetometer. To get an unambiguous orientation estimation solution those two parts need to be combined via sensor fusion.

3.1.1 Orientation from angular rate (gyroscope)

First the orientation of the earth frame relative to the sensor frame at time t, ${}^{S}_{E}\boldsymbol{q}_{\omega,t}$ is computed by numerically integrating the quaternion derivative ${}^{S}_{E}\dot{\boldsymbol{q}}_{\omega,t}$. This is done by multiplication with the quaternion. The gyroscope measures the angular rate of change ${}^{S}\boldsymbol{\omega}_t$ in the sensor frame which is then multiplied with the quaternion estimate of the previous orientation ${}^{S}_{E}\hat{\boldsymbol{q}}_{est,t-1}$ and the sampling period.

3. `https://x-io.co.uk/open-source-imu-and-ahrs-algorithms` (accessed: 24.06.2020)

$$
\begin{aligned}
{}^S_E\boldsymbol{q}_{\omega,t} &= {}^S_E\hat{\boldsymbol{q}}_{est,t-1} + {}^S_E\dot{\boldsymbol{q}}_{\omega,t}\Delta t && (31)\\
&= {}^S_E\hat{\boldsymbol{q}}_{est,t-1} + (\frac{1}{2}{}^S_E\hat{\boldsymbol{q}}_{est,t-1} \otimes {}^S\boldsymbol{\omega}_t)\Delta t && (32)
\end{aligned}
$$

In this equations Δt is the sampling period, $\otimes$ denotes the quaternion multiplication as shown in equation (7), ${}^S\boldsymbol{\omega}_t$ is the measured angular rate by the gyroscope and the subscript ω denotes the orientation calculation from angular rates.

3.1.2 Orientation from vectors (accelerometer and magnetometer)

In order to find an orientation estimation from accelerometer and magnetometer readings an optimisation problem needs to be solved. The main idea is to find an objective function $\boldsymbol{f}({}^S_E\hat{\boldsymbol{q}}, {}^E\hat{\boldsymbol{d}}, {}^S\hat{\boldsymbol{s}})$ which is minimised to find an optimum solution.

$$
\underset{{}^S_E\hat{\boldsymbol{q}} \in \mathbb{R}^4}{min} \boldsymbol{f}({}^S_E\hat{\boldsymbol{q}}, {}^E\hat{\boldsymbol{d}}, {}^S\hat{\boldsymbol{s}}) \tag{33}
$$

With ${}^S_E\hat{\boldsymbol{q}}$ is the orientaion of the sensor, ${}^E\hat{\boldsymbol{d}}$ is a predefined reference direction of the field in the earth frame and ${}^S\hat{\boldsymbol{s}}$ is the measured direction of the field in the sensor frame.

Finding an optimum solution in linear algebra is to find the minimum through it's Jacobian:

$$
\nabla\boldsymbol{f}({}^S_E\hat{\boldsymbol{q}}, {}^E\hat{\boldsymbol{d}}, {}^S\hat{\boldsymbol{s}}) = \boldsymbol{J}^T({}^S_E\hat{\boldsymbol{q}}, {}^E\hat{\boldsymbol{d}})\boldsymbol{f}({}^S_E\hat{\boldsymbol{q}}, {}^E\hat{\boldsymbol{d}}, {}^S\hat{\boldsymbol{s}}) \tag{34}
$$

Many optimisation algorithms exist but the gradient descent algorithm is one of the simplest to both implement and compute, which leads to the following equation:

$$
{}^S_E\boldsymbol{q}_{k+1} = {}^S_E\hat{\boldsymbol{q}}_k - \mu\frac{\nabla\boldsymbol{f}({}^S_E\hat{\boldsymbol{q}}_k, {}^E\hat{\boldsymbol{d}}, {}^S\hat{\boldsymbol{s}})}{||\nabla\boldsymbol{f}({}^S_E\hat{\boldsymbol{q}}_k, {}^E\hat{\boldsymbol{d}}, {}^S\hat{\boldsymbol{s}})||} \tag{35}
$$

||...|| denotes the quadratic norm $\mathbb{L}^2$, k denotes discrete time steps, Δt are the time steps and μ is the step size. The step size and time step parameter define how fast and if the algorithm converges to its optimum at all, so they need to be chosen with caution.

3.1.3 Physical assumptions about the sensors measurements

Next a few assumptions about the physical measurements of the sensors need to be done. Like shown in equation (15) we take the assumption that gravity only has a vertical component, normal to the earth surface in the z-axis for the accelerometer,

$$
{}^E\boldsymbol{g} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \tag{36}
$$

and the measured direction of the field is therefore defined as:

$$^S\hat{\boldsymbol{a}} = \left[\begin{array}{cccc} 0 & a_x & a_y & a_z \end{array}\right] \tag{37}$$

For the magnetometer it can be assumed, that the earth's magnetic field only has components in one horizontal axis and the vertical axis, the vertical component due to the inclination of the field. This follows,

$$^E\boldsymbol{b} = \left[\begin{array}{cccc} 0 & b_x & 0 & b_z \end{array}\right] \tag{38}$$

and the measured direction of the magnetic field is therefore defined as

$$^S\hat{\boldsymbol{m}} = \left[\begin{array}{cccc} 0 & m_x & m_y & m_z \end{array}\right] \tag{39}$$

The measurement of gravity or the earth's magnetic field alone will not provide a unique orientation of the sensor. To do so, the measurements and reference directions of both fields need to be combined. With this assumptions the gradient decent algorithm can be simplified to,

$$^S_E\boldsymbol{q}_{\nabla,t} = {}^S_E\hat{\boldsymbol{q}}_{est,t-1} - \mu\frac{\nabla\boldsymbol{f}}{||\nabla\boldsymbol{f}||} \tag{40}$$

where the sub-script ∇ indicates that the quaternion is calculated using the gradient descent algorithm, t denotes the estimation at a moment in time and ∇f is the combined optimisation of the accelerometer and magnetometer objective function and its Jakobian:

$$\nabla f = \begin{cases} \boldsymbol{J}_g^T({}^S_E\hat{\boldsymbol{q}}_{est,t-1})\nabla\boldsymbol{f}_g({}^S_E\hat{\boldsymbol{q}}_{est,t-1}, {}^S\hat{\boldsymbol{a}}_t) \\ \boldsymbol{J}_g^T({}^S_E\hat{\boldsymbol{q}}_{est,t-1}, {}^E\hat{\boldsymbol{b}}_t)\nabla\boldsymbol{f}_g({}^S_E\hat{\boldsymbol{q}}_{est,t-1}, {}^S\hat{\boldsymbol{a}}_t, {}^E\hat{\boldsymbol{b}}_t, {}^S\hat{\boldsymbol{m}}_t) \end{cases} \tag{41}$$

3.1.4 Fusion: Complimentary filter

The fusion then combines the quaternion calculations ${}^S_E\boldsymbol{q}_{\nabla,t}$ and ${}^S_E\boldsymbol{q}_{\omega,t}$ like,

$${}^S_E\boldsymbol{q}_{est,t} = \gamma_t {}^S_E\boldsymbol{q}_{\nabla,t} + (1-\gamma_t){}^S_E\boldsymbol{q}_{\omega,t}, \text{ with } 0 \leq \gamma_t \leq 1 \tag{42}$$

where γ_t is a weight applied to each orientation. An optimal value of γ_t can be defined as that value, which ensures the weighted divergence of ${}^S_E\boldsymbol{q}_{\omega,t}$ to be equal to the weighted convergence of ${}^S_E\boldsymbol{q}_{\nabla,t}$. This means, the accelerometer and magnetometer compensate the error caused by the gyroscope drift. Moreover in the original paper [MHV11], a single parameter β was introduced as a filter gain, which represents all zero mean gyroscope measurement errors, expressed as the magnitude of a quaternion derivative, and therefore, is the sole weighting factor in the filter.

Finally the system equation can be written as

$${}^S_E\boldsymbol{q}_{est,t} = {}^S_E\hat{\boldsymbol{q}}_{est,t-1} + \left[{}^S_E\dot{\boldsymbol{q}}_{\omega,t} - \beta\frac{\nabla f}{||\nabla f||}\right]\Delta t, \tag{43}$$

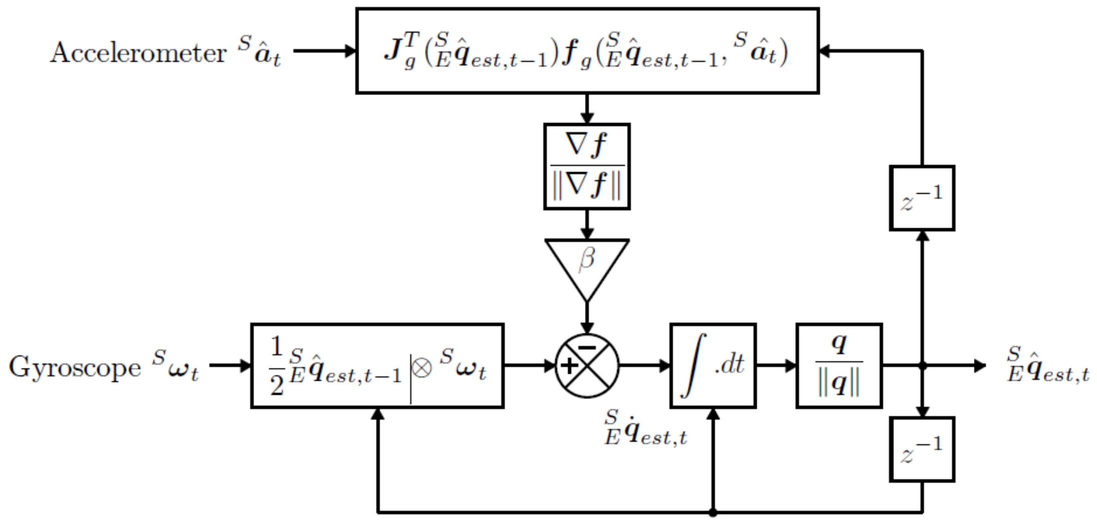

Accelerometer ${}^S\hat{\boldsymbol{a}}_t$
$\boldsymbol{J}_g^T({}^S_E\hat{\boldsymbol{q}}_{est,t-1})\boldsymbol{f}_g({}^S_E\hat{\boldsymbol{q}}_{est,t-1}, {}^S\hat{\boldsymbol{a}}_t)$
$\frac{\nabla \boldsymbol{f}}{\|\nabla \boldsymbol{f}\|}$
β
z^{-1}
Gyroscope ${}^S\boldsymbol{\omega}_t$
$\frac{1}{2}{}^S_E\hat{\boldsymbol{q}}_{est,t-1} \otimes {}^S\boldsymbol{\omega}_t$
$\int .dt$
$\frac{\boldsymbol{q}}{\|\boldsymbol{q}\|}$
${}^S_E\hat{\boldsymbol{q}}_{est,t}$
${}^S_E\dot{\boldsymbol{q}}_{est,t}$
z^{-1}

Figure 7 – Block diagram representation of the complete orientation filter for an IMU(6DOF) implementation, graphic taken from [MHV11]

with

$$ {}^S_E\dot{\boldsymbol{q}}_{\omega,t} = \frac{1}{2}{}^S_E\hat{\boldsymbol{q}}_{est,t-1} \otimes {}^S\boldsymbol{\omega}_t. \tag{44}$$

Figure 7 shows the corresponding block diagram for the complementary filter in 6DOF without the magnetometer implementation. Only the objective function $\nabla \boldsymbol{f}({}^S_E\hat{\boldsymbol{q}}, {}^E\hat{\boldsymbol{d}}, {}^S\hat{\boldsymbol{s}})$ would change with a 9DOF implementation and the magnetometer in use. In his original work Madgwick also designed a magnetic distortion and gyroscope bias drift compensation for a MARG(9DOF) implementation. The factor $\frac{\boldsymbol{q}}{\|\boldsymbol{q}\|}$ stands for the unit quaternion normalisation and z^{-1} is a first order discrete time delay block.

3.2 Good Attitude

This section points out the most important derivation steps for Valenti, Dryanovski and Xiaos fusion algorithm. In this work it is referred to as the *good attitude filter*. For a detailed explanation on how the algorithm works the reader is referred to [VDX15]. The definition of the system is based on the same approach presented in [MHV11] and they use the same sensor measurement data. Also all quaternions are normalised to unit quaternions. In the literature, the quaternion derivative from an angular rate measurement is usually calculated for the forward quaternion ${}^S_E\boldsymbol{q}$, that is, the one representing the orientation of the sensor frame with respect to the earth frame. Valenti, Dryanovski and Xiao used the inverse orientation, but it can be shown that they are the same besides the contrary point of view. They also defined the local frame with the letter L and the global frame with G. For better compatibility to the *complimentary filter* from the previous section, the local and global reference in their derivations is substituted with the sensor frame, indicated by the letter S and the earth frame indicated by the letter E respectively.

Their approach is to align the global coordinate frame E with the magnetic north. Specifically, the global frame's x-axis points in the same direction as the local magnetic field (the z-axis remains vertical). Obviously, this global frame is only "fixed" in the case when the local magnetic field does not change its heading [VDX15]. Furthermore in their derivations they decompose the quaternion rotation ${}^S_E\boldsymbol{q}$ into two separate auxiliary quaternions, which makes the performance of the algorithm more robust to magnetic disturbance and linear acceleration problems. The objective is to find the rotation ${}^S_E\boldsymbol{q}$ that positions the sensor frame parallel to the earth surface, $\boldsymbol{q}_{acc}$, and secondly rotates the sensor only around the z-axis, $\boldsymbol{q}_{mag}$, to point the heading direction to the north of the magnetic field.

$$ {}^S_E\boldsymbol{q} = \boldsymbol{q}_{acc} \otimes \boldsymbol{q}_{mag} \tag{45}$$

As a result, the calculations from the accelerometer only perform a rotation in the pitch and roll angle. The magnetometer only influences the rotation about the yaw angle. Besides, no gradient descent algorithm is necessary. Additionally an adaptive gain calculation is developed, but this is not relevant for human head movements, because it only addresses very fast non-gravitational acceleration. With this sensor fusion algorithm one doesn't need to calibrate the headtracker before using it, because the initial state of the sensor is already calculated from the acceleration and magnetometer readings in one single step. Futher, they adopt a low-pass filter to separate the bias from the actual angular velocity measurements. To avoid filtering useful information, the low-pass filtering is applied only when the sensor is in a steady-state condition that is previously checked.

The main objective is defined as,

$$\begin{cases} \boldsymbol{R}^T({}^S_E\boldsymbol{q})^S\boldsymbol{a} = {}^E\boldsymbol{g} \\ \boldsymbol{R}^T({}^S_E\boldsymbol{q})^S\boldsymbol{m} = {}^E\boldsymbol{h} \end{cases} \tag{46}$$

with $\boldsymbol{R}^T({}^S_E\boldsymbol{q})$ being the rotation matrix defined in equation 9, ${}^S\boldsymbol{a}$ is the acceleration data, ${}^S\boldsymbol{m}$ are the magnetometer measurements, ${}^E\boldsymbol{g}$ is the earth's gravitation and ${}^E\boldsymbol{h}$ is the magnetic field of the earth frame. This system, however, is overdetermined - each of the two equations provides two independent constraints on the orientation ${}^S_E\boldsymbol{q}$, whereas the orientation only has three degrees of freedom. The solution is to modify the second equation so it has only one constraint left, to point to the positive earth xz-half-plane ${}^E\boldsymbol{\Pi}_{zx^+}$.

$$\begin{cases} \boldsymbol{R}^T({}^S_E\boldsymbol{q})^S\boldsymbol{a} = {}^E\boldsymbol{g} \\ \boldsymbol{R}^T({}^S_E\boldsymbol{q})^S\boldsymbol{m} \in {}^E\boldsymbol{\Pi}_{zx^+} \end{cases} \tag{47}$$

They further define q_{mag} to have only a single degree of freedom:

$$\boldsymbol{q}_{mag} = \begin{bmatrix} q_{0_{mag}} & 0 & 0 & q_{3_{mag}} \end{bmatrix} \tag{48}$$

3.2.1 Quaternions from accelerometer readings

To find the auxiliary quaternion $\boldsymbol{q}_{acc}$ one has to solve the equation

$$\boldsymbol{R}({}^S_E\boldsymbol{q})^E\boldsymbol{g} = {}^S\boldsymbol{a} \tag{49}$$

resulting in:

$$\boldsymbol{R}(\boldsymbol{q}_{acc})\boldsymbol{R}(\boldsymbol{q}_{mag}) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}. \tag{50}$$

The representation of the gravity vector in the global frame only has a component on the z-axis; therefore, any rotation about this axis does not produce any change on it. [VDX15] So one can rewrite equation 50 and expand the multiplication from the rotation matrix, leading to:

$$\begin{cases} 2(q_{1_{acc}}q_{3_{acc}} + q_{0_{acc}}q_{2_{acc}}) = a_x \\ 2(q_{2_{acc}}q_{3_{acc}} - q_{0_{acc}}q_{1_{acc}}) = a_y \\ q_{0_{acc}}^2 - q_{1_{acc}}^2 - q_{2_{acc}}^2 + q_{3_{acc}}^2 = a_z \end{cases} \tag{51}$$

This system is underdetermined and has therefore infinite solutions. To solve this, one can set $q_2 = 0$ or $q_3 = 0$ zero, following that the solution has a singularity at $a_z = \mp 1$. The final formulation of $\boldsymbol{q}_{acc}$ that avoids the singularity problem can be obtained by combining both solutions, as follows:

$$\boldsymbol{q}_{acc} = \begin{cases} \begin{bmatrix} \sqrt{\frac{a_z+1}{2}} & -\frac{a_y}{\sqrt{2(a_z+1)}} & \frac{a_z}{\sqrt{2(a_z+1)}} & 0 \end{bmatrix}^T, & a_z \geqslant 0 \\ \begin{bmatrix} -\frac{a_y}{\sqrt{2(1-a_z)}} & \sqrt{\frac{1-a_z}{2}} & 0 & \frac{a_x}{\sqrt{2(1-a_z)}} \end{bmatrix}^T, & a_z < 0 \end{cases} \tag{52}$$

Effectively, the singularity problem is solved by having two separate formulations for $\boldsymbol{q}_{acc}$ depending on the hemisphere in which $\boldsymbol{a}$ is pointing. It is important to mention that $\boldsymbol{q}_{acc}$ is not continuous at the point $a_z = 0$, but this will not be an issue due to the formulation of $\boldsymbol{q}_{mag}$.

3.2.2 Quaternions from magnetometer readings

In the first step, the rotation $\boldsymbol{q}_{acc}$ was calculated and can now be applied before the calculation of rotation from the magnetometer readings $\boldsymbol{q}_{mag}$, leading to an intermediate state where the z-axis is now fixed to the normal of the earth's surface, but with unknown yaw angle. This is shown as

$$\boldsymbol{R}^T(\boldsymbol{q}_{acc})^S\boldsymbol{m} = \boldsymbol{l}, \tag{53}$$

with ${}^S\boldsymbol{m}$ being the sensor frame magnetic field vector and $\boldsymbol{l}$ being the rotated magnetic field vector. Next, it is to find the quaternion $\boldsymbol{q}_{mag}$ that rotates the vector $\boldsymbol{l}$ into the vector that lies on the ${}^E\boldsymbol{\Pi}_{zx+}$ plane using the following system equation:

$$\boldsymbol{R}^T(\boldsymbol{q}_{mag}) \begin{bmatrix} l_x \\ l_y \\ l_z \end{bmatrix} = \begin{bmatrix} \sqrt{\Gamma} \\ 0 \\ l_z \end{bmatrix} \tag{54}$$

where

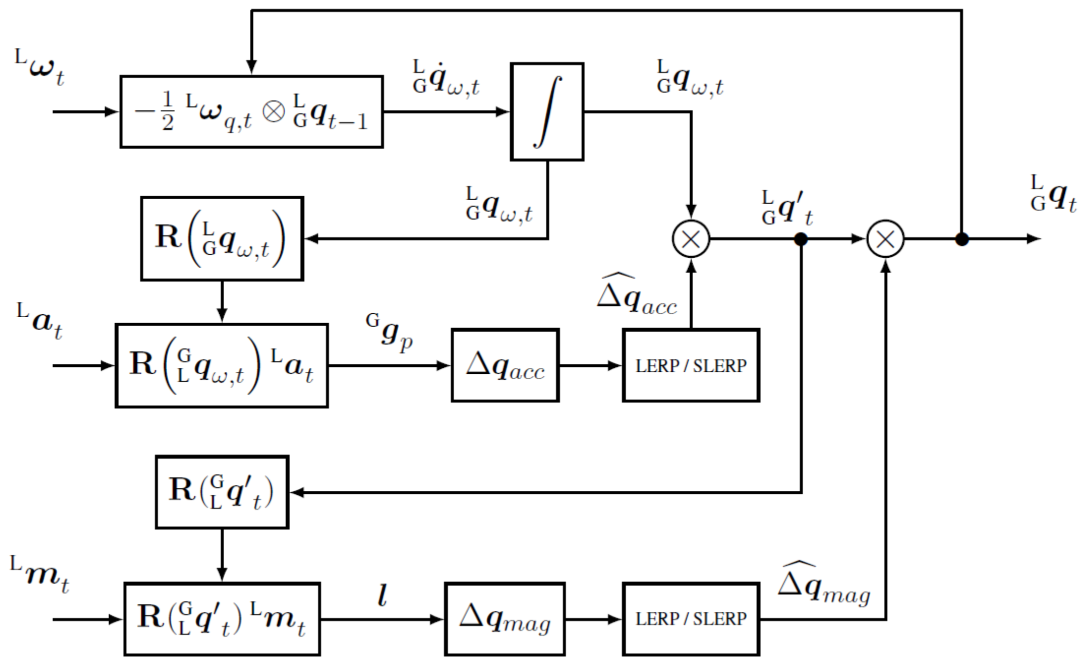
$$\Gamma = l_x^2 + l_y^2 \tag{55}$$

This rotation will only change the heading component of the orientation without affecting the roll and pitch components. Therefore, in presence of magnetic disturbances, their influence is only limited on affecting the heading angle. In the same way like shown in section 3.2.1, one can solve this system by expanding the multiplication from the rotation matrix leeding to the solution:

$$\boldsymbol{q}_{mag} = \begin{bmatrix} \frac{l_y}{\sqrt{2}\sqrt{\Gamma+l_x\sqrt{\Gamma}}} & 0 & 0 & \frac{\sqrt{\Gamma+l_x\sqrt{\Gamma}}}{\sqrt{2\Gamma}} \end{bmatrix}^T \tag{56}$$

It is easy to see, that this solution again has a singularity for $l_x < 0$ and $l_z = 0$. This can be prevented by flipping the vector $\boldsymbol{l}$ about $180°$ when $l_x < 0$. The final solution again has two cases:

$$\boldsymbol{q}_{mag} = \begin{cases} \begin{bmatrix} \frac{\sqrt{\Gamma+l_x\sqrt{\Gamma}}}{\sqrt{2\Gamma}} & 0 & 0 & \frac{l_y}{\sqrt{2}\sqrt{\Gamma+l_x\sqrt{\Gamma}}} \end{bmatrix}^T, & l_x \geqslant 0 \\ \begin{bmatrix} \frac{l_y}{\sqrt{2}\sqrt{\Gamma-l_x\sqrt{\Gamma}}} & 0 & 0 & \frac{\sqrt{\Gamma-l_x\sqrt{\Gamma}}}{\sqrt{2\Gamma}} \end{bmatrix}^T, & lx_{<}0 \end{cases} \tag{57}$$



$^{\mathrm{L}}\boldsymbol{\omega}_t$
$-\frac{1}{2}\,{}^{\mathrm{L}}\boldsymbol{\omega}_{q,t} \otimes {}^{\mathrm{L}}_{\mathrm{G}}\boldsymbol{q}_{t-1}$
${}^{\mathrm{L}}_{\mathrm{G}}\dot{\boldsymbol{q}}_{\omega,t}$
$\int$
${}^{\mathrm{L}}_{\mathrm{G}}\boldsymbol{q}_{\omega,t}$
${}^{\mathrm{L}}_{\mathrm{G}}\boldsymbol{q}_{\omega,t}$
$\mathbf{R}\left({}^{\mathrm{L}}_{\mathrm{G}}\boldsymbol{q}_{\omega,t}\right)$
${}^{\mathrm{L}}_{\mathrm{G}}\boldsymbol{q}'_t$
${}^{\mathrm{L}}_{\mathrm{G}}\boldsymbol{q}_t$
$\widehat{\Delta\boldsymbol{q}}_{acc}$
$^{\mathrm{L}}\boldsymbol{a}_t$
$\mathbf{R}\left({}^{\mathrm{G}}_{\mathrm{L}}\boldsymbol{q}_{\omega,t}\right){}^{\mathrm{L}}\boldsymbol{a}_t$
$^{\mathrm{G}}\boldsymbol{g}_p$
$\Delta\boldsymbol{q}_{acc}$
LERP / SLERP
$\mathbf{R}({}^{\mathrm{G}}_{\mathrm{L}}\boldsymbol{q}'_t)$
$^{\mathrm{L}}\boldsymbol{m}_t$
$\mathbf{R}({}^{\mathrm{G}}_{\mathrm{L}}\boldsymbol{q}'_t)\,{}^{\mathrm{L}}\boldsymbol{m}_t$
$\boldsymbol{l}$
$\Delta\boldsymbol{q}_{mag}$
LERP / SLERP
$\widehat{\Delta\boldsymbol{q}}_{mag}$

Figure 8 – Block diagram of the Good Attitude filter for a MARG implementation (with magnetometer data), graphic from [VDX15]

3.2.3 Fusion: Good Attitude filter

Finally, the delta quaternions $\Delta\boldsymbol{q}_{acc}$ and $\Delta\boldsymbol{q}_{mag}$ can be calculated to correct the rotation estimation from angular readings ${}^{S}_{E}\boldsymbol{q}_{\omega}$. The singularity does not matter because the correction quaternions are applied every time step and therefore only small changes, not in the range of singularity's, appear. The predicted quaternion ${}^{S}_{E}\boldsymbol{q}_{\omega}$ is corrected by the two delta quaternions leading to the final solution:

$$ {}^{S}_{E}\boldsymbol{q} = {}^{S}_{E}\boldsymbol{q}_{\omega} \otimes \Delta\boldsymbol{q}_{acc} \otimes \Delta\boldsymbol{q}_{mag} \tag{58} $$

As the delta quaternions are affected by the accelerometer's high frequency noise and magnetic disturbances, before applying it to the predicted quaternion, they are scaled down by using an interpolation with the identity quaternion $\boldsymbol{q}_I$. Two different scaling interpolations are implemented, the linear interpolation (LERP) and the spherical linear interpolation (SLERP) shown in [VDX15]. LERP is used for little noise levels, if $\Delta q_{0acc} > \epsilon$ with $\epsilon = 0.9$ and SLERP for higher noise levels.

The reader should keep in mind, that the indices for the local and global frame in Figure 8 need to be substituted like, $L = S$ and $G = E$ because the graphic was taken from the original work from [VDX15].

4 Assembly and sensor shields

The whole processing and sensor fusion for the orientation was done on the Adafruit Feather M0+ Bluefruit. It is an all-in-one Cortex M0+ Arduino-compatible mircro controller with Bluetooth Low Energy and built in USB and battery charging. For iOS Bluetooth Low Energy works out of the box, but for windows you need to install third party software (MIDIberry[4] and loopMidi[5]) to use the tracker e.g. with a DAW, like Reaper. This is due to missing drivers for MIDI over bluetooth on the windows platform.

The Feather M0+ uses an ATSAMD21G18 ARM Cortex M0+ processor, clocked at 48 MHz and at 3.3V logic. It has a 256K of FLASH (8x more than the Atmega328 or 32u4) and 32K of RAM (16x more than the Atmega328 or 32u4). Additionally, it comes with built in USB, so it has USB-to-Serial program and debug capability built in with no need for an FTDI-like chip. Figure 9 shows the mircro controller and its pinout. All three sensors were connected to a Feather M0+ controller each. To start tracking only four connections needed to be done - the V_{in} pin and the ground GND - to connect the sensors with the power supply of the micro controller and the SCL (Serial Clock) and SDA(Serial Data) - to enable data transmission over I^2C. Additionally all mirco controllers were connected to a push button to reset the orientation and start tracking at the same time. Figures 10, 11 and 12 show the sensors used in this work.

The sensors in use were all mounted on a pcb board to ensure that the same rotation is performed during the measurements simultaneously. Tracking elements for the optical

4. `https://www.microsoft.com/en-us/p/midiberry/9n39720h2m05?activetab=pivot:overviewtab`
5. `http://www.tobias-erichsen.de/software/loopmidi.html`

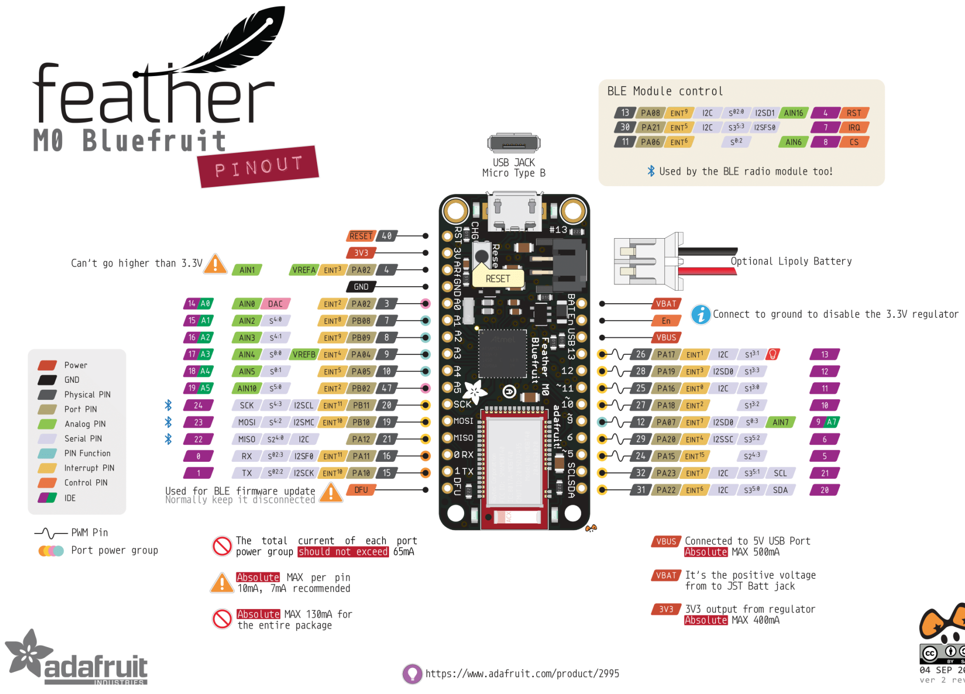


Figure 9 – Adafruit Feather M0+ Bluefruit and pinout

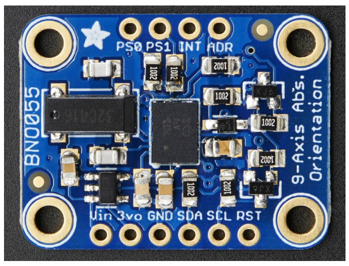


Figure 10 – BNO055 sensor shield from Adafruit

reference system were also mounted on the pcb board. The final test setup is shown in Figure 13. Additionally, it was discovered, that even loose cables on a bread board can cause disturbances for the magnetometer measurements. For specific sensor characteristics and features like zero rates, sensitivities, baud rates and sensor resolutions, the reader is referred to the data sheet of the respective sensor. All sensors were set to the same ranges, $\pm 250 dps$ for the gyroscope, $2G$ for the accelerometer and full-scale, depending on the sensor, for the magnetometer. A detailed analysis on gyroscope zero drift rates and how to interpret details on the manufactures data sheet can be found here[6]. The review points out that the NXP sensor should outperform all other sensors in the zero drift and is therefore the most precise. This assumption will be taken into account for the analysis of the fusion performance.

6. `https://learn.adafruit.com/comparing-gyroscope-datasheets/overview`

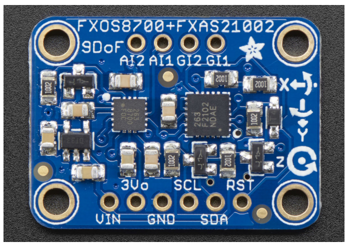


Figure 11 – NXP Precision 9DOF sensor shield from Adafruit

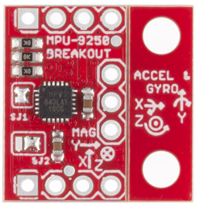


Figure 12 – MPU9250 sensor shield from Sparkfun

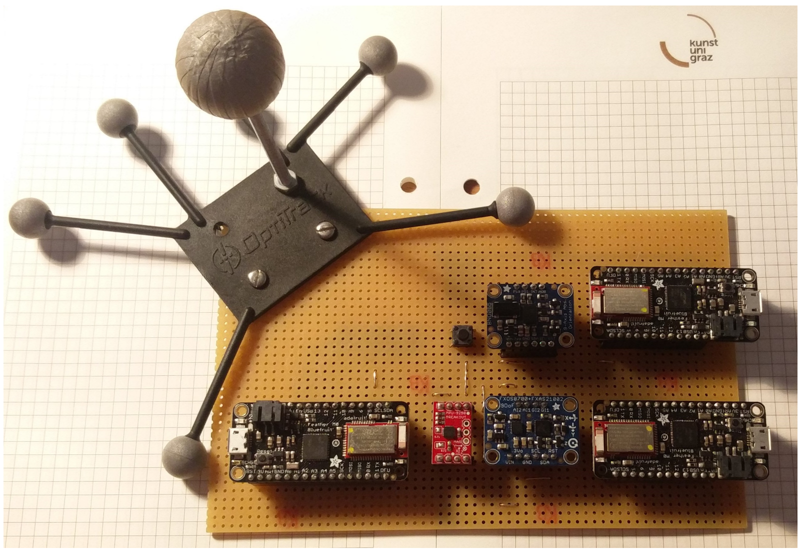


Figure 13 – pcb board with all sensors connected to an Adafruit Feather M0+ each and reference points mounted on the board for optical tracking

Table 1 – Fusion algorithms setups used for the measurements

measurements/sensors:	BNO	MPU	NXP
1-3	Good Attitude	Good Attitude	Good Attitude
4-6	internal	Madgwick MARG	Madgwick MARG
7-9	internal	Madgwick IMU	Madgwick IMU

5 Experiments

The above mentioned fusion algorithms were tested on three different orientation sensors shields, (BNO055, NXP 9-DOF and MPU9250) [Sen16] [Sem17] [Sem15] [Inv16] connected to an Adafruit Feather M0 Bluefruit LE micro processor. Both implementations of Madgwicks *complimentary filter* for IMU (6DOF) and MARG (9DOF) were tested on the MPU9250 and NXP sensor shields, the *Good Attitude Filter* in 9DOF was tested on all sensor shields and the intern sensor fusion of the BNO055 shield in 9DOF was tested too. All attitude estimations are compared with each other and with an optical tracking system as reference (Optitrack Flex 13). The raw sensor data and the orientation in yaw, pitch, roll angles were transmitted via a serial print command of the Arduino boards via USB and then written to a text (.txt) file with a small software application, *OptiAndSerialToFile*, written by Daniel Rudrich. This script opens the serial port connected to the PC or Mac and directly writes the *SerialPrint()* commands of the Arduino to a text file with some delimiter in a (.csv) like form, so it can be imported and analysed in Matlab. Additionally, the measurement data from the optical reference system is also written to file. Upfront, all sensors were calibrated for soft and hard iron errors with the help of Paul Stoffregens MotionCalApplication[7]. Before each measurement the sensors were put in an idle state, then moved by hand in a figure of eight pattern to enable self calibration and bias estimations. To start measurements at the exact same time a button was built on the pcb-board, which when pressed, starts a routine on the Arduino board to reset the orientation to a known state defined by the gravity vector of the accelerometer and prints the name of the sensor, e.g: MPU in the serial bus, which then triggers the *OptiAndSerialToFile* script to start writing to the file.

Three movements were performed for every sensor setup - an isolated rotation into every yaw pitch and roll angle - idle state at the beginning, then slow figure of eight movements, ending in an idle state again - and starting with slow movements getting faster. The test setups can be seen in the following table:

All sensor were initially set to a sampling frequency of $100Hz$ for the data input, but this is not the same as the fusion output data rates. The performance of the sensor fusion drastically relies on the correct time steps Δt. Therefore time stamps needed to be implemented in the arduino code, actually calculating the time steps Δt from one filter update to the next. This is because in both sensor fusion algorithms in the prediction step for the orientation from angular readings the attitude estimation is numerical integrated with the factor Δt, which leads to errors if the integration time is not correct.

7. `https://github.com/PaulStoffregen/MotionCal.git`

Table 2 – Measured sampling frequency ranges from timestamps of the different sensor setups

BNO	MPU	NXP	OPTI
15-28 Hz	19-27 Hz	16-21 Hz	109-119 Hz

It was shown in Madgwicks internal report, section 5.4, that the algorithm performance increases with higher sampling frequencies till about $50Hz$. In their work [MHV11] they first recorded the data set and afterwards applied the fusion algorithm with different sampling frequencies ranging from $1-512Hz$. Performing the fusion on a micro controller chip, like the M0+ limits the achievable sampling frequency to about $20Hz$. Perhaps this could be improved with some optimisation of the source code. The averaged sampling frequencies in this experiment were calculated from the measurement time stamps (time passed, from one data entry written to file to the next) for each of the 9 different measurements ranging from:

The optical reference system runs on a sampling frequency of $120Hz$. Although in the measured data it seems like, that when there is very little change in orientation, e.g idle state measurements 2.,5. and 8., data is not written every time.

6 Results

6.1 Raw measurements

To ensure that the raw sensor data from accelerometer, gyroscope and magnetometer readings is correct and all vectors have the same orientation in the x-y-z axis, the measured vectors are compared to true raw sensor data calculated from the quaternion representation of the optical reference system. For the accelerometer this is achieved by rotating the quaternions with the gravity vector $\boldsymbol{g}$ and then transforming the spherical coordinates to cartesian.

$$\boldsymbol{g} = \begin{bmatrix} 0 \\ 0 \\ 9.80665 \end{bmatrix} \tag{59}$$

The same must be done for the gyroscope, but here the quaternion first needs to be derived to get the orientation vector. In quaternion domain this is done by left multiplying the reference quaternion q_{ref_t} with the previous one $q_{ref_{t-1}}$:

$$q_{diff} = q^T_{ref_{t-1}} \otimes q_{ref_t} \tag{60}$$

Calculating the true magnetometer data works the same way like for the accelerometer, only the gravitational vector needs to be exchanged with a vector $\boldsymbol{m}$ that points to the magnetic north. For this a sample from the idle state measurement was extracted. Finally the true magnetic vector needs to be scaled with the local magnetic field constant

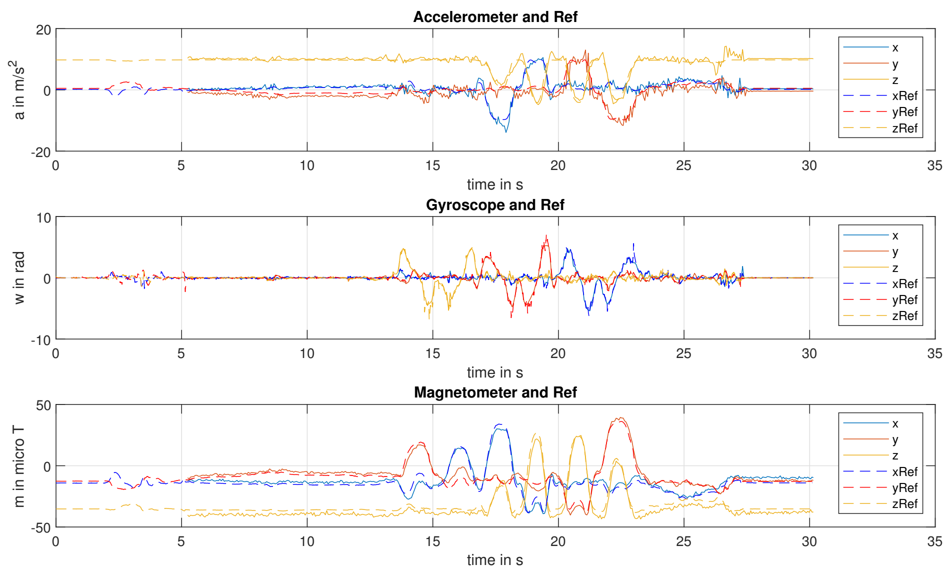


Figure 14 – NXP precision 9DOF sensor data with reference calculated from optical tracking system in cartesian coordinates

Table 3 – Summed and normalised RMS errors of the sensor data compared to the reference signal

BNO			NXP			MPU		
accel	gyro	mag	accel	gyro	mag	accel	gyro	mag
0.0101	0.0051	0.0386	0.0105	0.0055	0.0383	0.0094	0.0043	0.0395

which was assumed to be about $40\mu T$ in Graz, Austria.

$$\boldsymbol{m} = \begin{bmatrix} -17.2584 \\ -12.0579 \\ -44.0009 \end{bmatrix} \tag{61}$$

Figure 14 represents the NXP sensor with *Good Attitude* sensor fusion, which shows a good match of the raw sensor data with the calculated from the optical reference system. The small offset for the magnetometer comes from the assumption of the local magnetic field strength of $40\mu T$. It can be neglected, because this is just a first evaluation step, to see if the measured data of all sensors is in the right orientation. This is done for all 9 measurements and all sensor shields to ensure correct data. The summed and normalised RMS errors from the reference signal to the measured sensor data for all 9 measurements are shown in the following table, but it is to note, that for the magnetometer this is just a rough assumption.

The RMS error for the MPUs gyro and accelerometer is lower than the error of the two others. Furthermore no significant differences between raw sensor data could be detected.

6.2 Euler and Quaternion interpretation problems

During the evaluation process a few problems occurred with plotting the data, both for the Euler and quaternion representation. Although quaternions are unambiguous to rotations, frame alignment can be achieved extrinsic or intrinsic. Depending on the reference system the attitude can be described with two different vectors and rotation angles, facing opposite directions. Following, that the $w-x-y-z$ quaternion trajectories of an orientation estimation from one sensor to another can differ, despite representing the same rotation. In such a case the trajectories are mirrored towards the respective axis and the corresponding quaternions would look like,

$$q_{sensor_1} = \begin{bmatrix} q_w & q_x & q_y & q_z \end{bmatrix}^T \text{ and } q_{sensor_2} = \begin{bmatrix} -q_w & -q_x & -q_y & -q_z \end{bmatrix}^T, \tag{62}$$

both representing the same rotation, just another way around. If one transforms this quaternions to yaw, pitch and roll angles 360° jumps occur in the plots. To ensure that all rotations are represented in the same way, the quaternions were multiplied by -1 if the sign of the w-channel was negative. Meaning that all sensors are aligned extrinsic with the reference system. Figure 15 shows a rotation along the yaw, pitch and roll angles for all three sensors using the Good Attitude algorithm in quaternion representation. All sensors were corrected to the same reference. Although good performance is shown

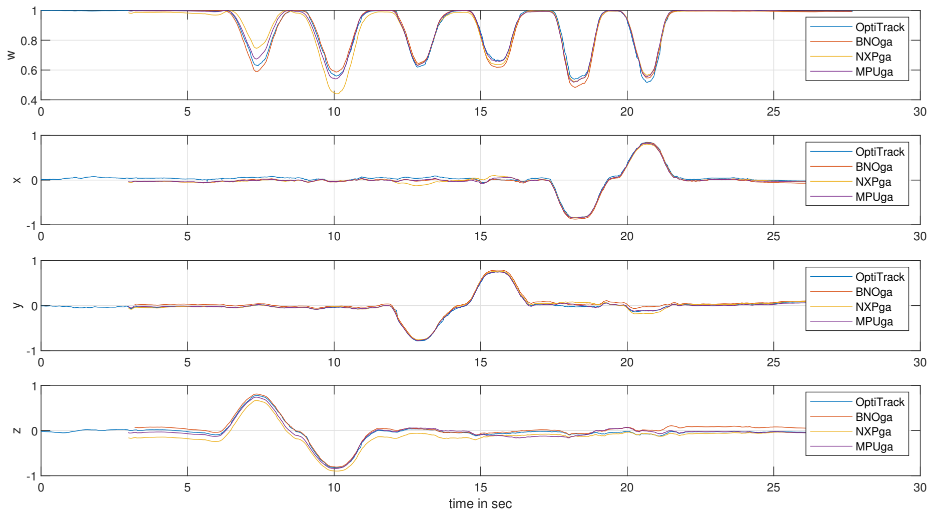


Figure 15 – Measurement 1: Rotation towards the yaw, pitch and roll angles for all three sensors using the Good Attitude algorithm in quaternion representation

for all three sensors, the plot doesn't give straightforward information about the actual rotation. A representation in Euler angles provides more readability of the orientation estimation. As mentioned above, the down side of this representation is that periodic jumps occur when the measured rotations exits the $\pm180°$ range of the $atan2$ function, which is used to transform quaternions to the yaw and roll angle. Such jumps are shown in Figure 16 in the yaw angle for the NXP sensor (yellow line) at about 15 seconds and in the roll angle for the OptiTrack system (blue line) at about 13 seconds. Additionally, ambiguities occur again, meaning that a rotation in Euler representation can be achieved in multiple ways, depending on the rotation order along its respective axis. An example of such an ambiguity is shown in Figure 16 for the reference system compared to all three sensors at about 13 seconds. If one compares Figure 16 and 15 it is not easy to see that both represent the exact same rotation in two different forms. This points out the problem with representing a 3D rotation in a two dimensional space. The Euler

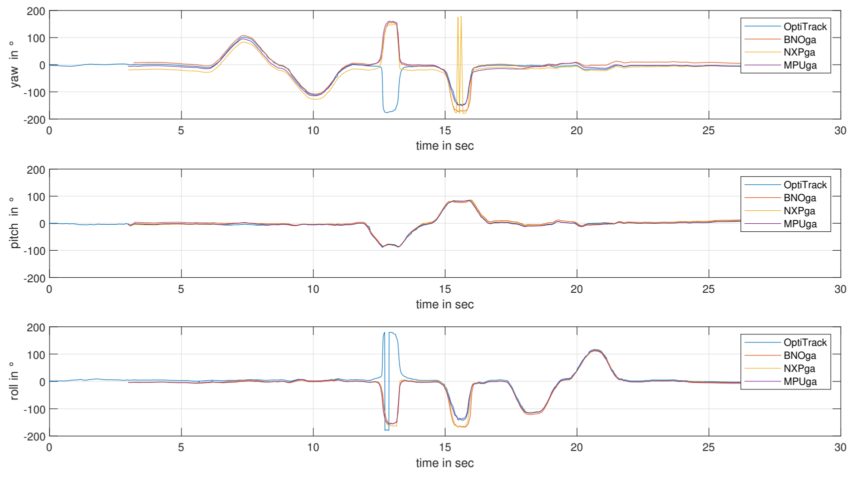


Figure 16 – Measurement 1: Rotation towards the yaw, pitch and roll angles for all three sensors using the Good Attitude algorithm in Euler representation

angle representation has the advantage that the decoupled yaw, pitch, roll angles may be more easily interpreted or visualised, but it fails to describe the coupling between each of the parameters and will subject to large and erratic errors if the Euler angle sequence reaches a singularity. On the other hand, the quaternion representation is a theoretical number system which doesn't show a direct connection between the measurements and the rotation performed in the real world.

In literature, the performance of an orientation estimation typically is given in RMS values of the Euler representation. In this work such a calculation is not trivial. This is because during the measurements the $\pm 180°$ boundaries for unambiguous data was not kept in mind. One could calculate RMS values from the quaternion representation, but due to non-linearities this doesn't provide significant results either. Therefore, the evaluation was done solely qualitatively.

6.3 Sensor performance evaluation

To compare the sensors with each other the Good Attitude fusion algorithm was implemented on each Adafruit board and three measurements were performed, yaw-pitch-roll, idle state and fast movements, like described in section 5. The first measurements from 1-3 are shown in Figure 16, 17,18. One can see, that for a normal use case like in

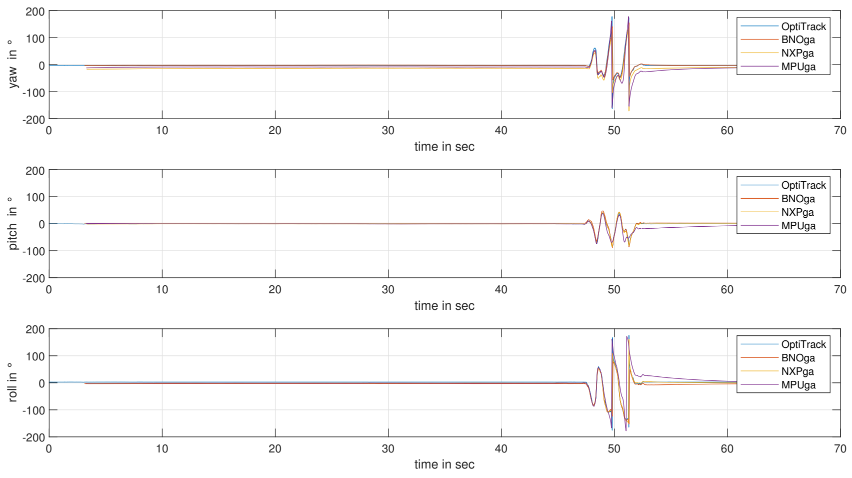


Figure 17 – Idle state measurement for all three sensors using the Good Attitude algorithm in Euler representation

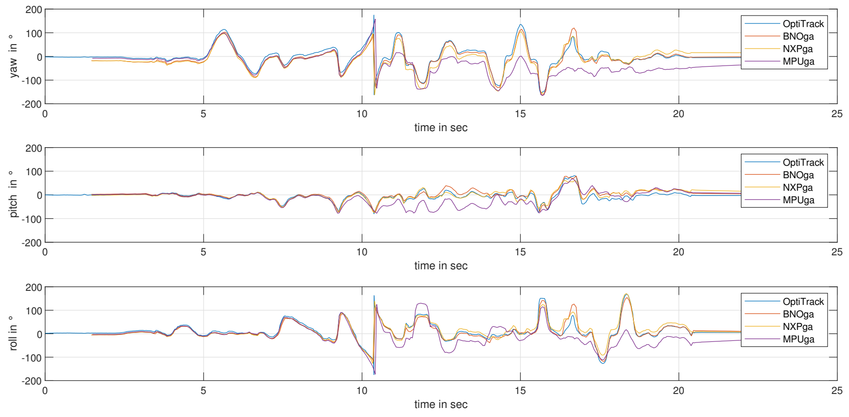


Figure 18 – Fast figure of eight measurement for all three sensors using the Good Attitude algorithm in Euler representation

measurement 1 (Figure 16), all sensors perform sufficiently for a AHRS. The BNO and NXP sensor overestimate the orientation up to $15°$ if there is a high deflection in the roll angle, whereas the MPUs performance is very similar to the reference system. No drift in the yaw angle is detected for the idle state measurement in Figure 17 but the convergence of the MPU sensor is slower than for the others. Finally in measurement 3 it is shown that all sensors struggle when facing very fast attitude changes. In this case the MPUs performance is the worst of the three.

6.4 Fusion algorithm evaluation

All fusion algorithms are implemented with the raw BNO sensor data to compare them with each other and the internal fusion of the BNO shield. Figure 19 shows that the internal sensor fusion of the BNO sensor outperforms all the others. This is the only fusion working out of the box. The Good Attitude fusion algorithm gives a solid orientation estimation for all tested sensors. Due to the decoupled design of magnetometer and accelerometer it is also more robust to magnetic disturbances than the BNO sensor in 9DOF. The Madgwick sensor fusion has a drift in the yaw angle in every implementation varying from $\frac{0.35°}{s}$ for 9DOF to $\frac{1°}{s}$ for 6DOF if calculated on the Adafruit M0+ chip. The Madgwick implementation in Figure 19 is done directly on the PC and shows a drift of $\frac{5°}{s}$ which is the worst performance of the whole measurement set. It is assumed that such large drifts only occur due to some unknown error in the sensor fusion performed on the PC, but the problem couldn't be found. The Madgwick algorithm in 9DOF performed directly onboard the Adafruit Feather M0+ chip provides sufficient orientation estimation for an AHRS.

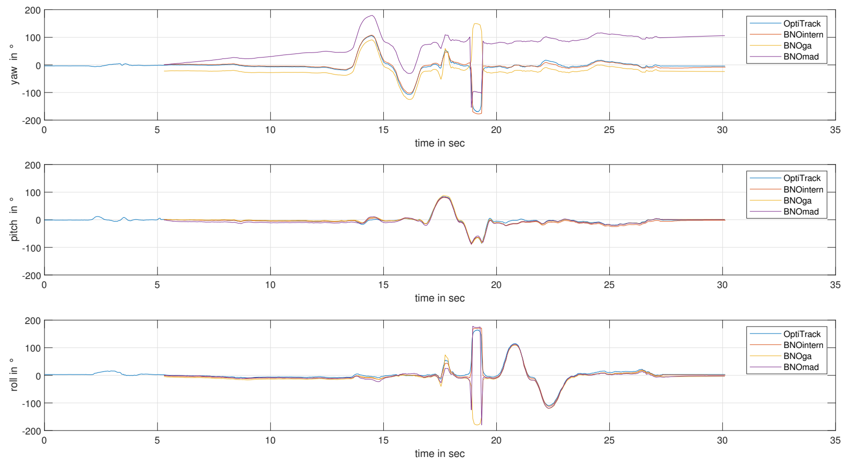


Figure 19 – Comparison between fusion algorithms (BNO intern, Good Attitude and Madgwick), with the BNO raw data, performing a rotation along the yaw, pitch and roll angles in Euler representation

7 Conclusion

In this work two fusion algorithms and three state of the art orientation sensor shields, the BNO055, NXP 9DOF and MPU9250 were tested and compared to each other. Beginning with the raw sensor data from accelerometer, gyroscope and magnetometer readings the differences in measurement performance are neglectable for an AHRS. All three sensors provide sufficient raw sensor data to perform a good orientation estimation. Only the MPU sensor suffers in accuracy if exposed to fast changes in attitude. As a result the orientation estimation mainly depends on the fusion algorithm and additional signal processing steps, like gyroscope bias estimation and filter gain adjustments. Magnetometer calibration needs to be addressed with caution, because it is the main cause for estimation errors in every 9DOF sensor. The Good Attitude fusion algorithm gives a solid orientation estimation for every tested sensor and measurement scenario leading to a good choice for a fusion algorithm. Additional it provides better stability against magnetic disturbances, than the other fusion algorithms due to its decoupled design. Neither the less, the build in fusion algorithm of the BNO055 shield outperforms all other sensors. Its main advantage is that it works out of the box. Particularly the 6DOF mode of the sensor shield, provides drift free orientation estimation without typical magnetometer calibration and magnetic distortion problems, because it doesn't use the magnetometer at all. The only disadvantage of the BNO sensor shield is its price ($\approx 32\$$) which is twice the amount as for the NXP and MPU sensor shields. All this leads to the conclusion, that the choice of the actual sensor is not as important as the fusion algorithm and the out of the box usability e.g., with the BNO sensor. Moreover if possible no magnetometers should be used for attitude estimation of a rigid body. The BNO sensor demonstrates that drift free orientation estimation is possible with only 6DOF. It is assume that this can be achieved with on the fly sensor calibration and smart gyroscope drift compensation, but further investigations need to be done. Perhaps inverse engineering the BNO sensor could lead to new insights.

References

[Ada20] Adafruit. (2020) Adafruit feather m0 bluefruit. [Online]. Available: https://learn.adafruit.com/adafruit-feather-m0-bluefruit-le/overview

[FLX+17] K. Feng, J. Li, Z. Xiaoming, C. Shen, Y. Bi, T. Zheng, and J. Liu, "A new quaternion-based kalman filter for real-time attitude estimation using the two-step geometrically-intuitive correction algorithm," *Sensors*, vol. 17, p. 2146, 09 2017.

[Inv16] InvenSense. (2016) Mpu-9250 product specification. [Online]. Available: https://www.invensense.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf

[Kal60] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 03 1960. [Online]. Available: https://doi.org/10.1115/1.3662552

[MHP05] R. Mahony, T. Hamel, and J. . Pflimlin, "Complementary filter design on the special orthogonal group so(3)," in *Proceedings of the 44th IEEE Conference on Decision and Control*, Dec 2005, pp. 1477–1484.

[MHV11] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan, "Estimation of imu and marg orientation using a gradient descent algorithm," in *2011 IEEE International Conference on Rehabilitation Robotics*, June 2011, pp. 1–7.

[MXB$^+$01] J. L. Marins, Xiaoping Yun, E. R. Bachmann, R. B. McGhee, and M. J. Zyda, "An extended kalman filter for quaternion-based orientation estimation using marg sensors," vol. 4, pp. 2003–2011 vol.4, Oct 2001.

[RBR$^+$17] M. Romanov, P. Berghold, D. Rudrich, M. Zaunschirm, M. Frank, and F. Zotter, "Implementation and evaluation of a low-cost head-tracker for binaural synthesis," 05 2017.

[Sem15] N. Semiconductors. (2015) Data sheet fxas21002c: 3-axis digital angular rategyroscope. [Online]. Available: https://www.nxp.com/docs/en/data-sheet/FXAS21002.pdf

[Sem17] ——. (2017) Data sheet fxos8700cq: 6-axis sensor with integrated linear accelerometer and magnetometer. [Online]. Available: https://www.nxp.com/docs/en/data-sheet/FXOS8700CQ.pdf

[Sen16] B. Sensortech. (2016) Data sheet bno055 intelligent 9-axis absolute orientation sensor. [Online]. Available: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bno055-ds000.pdf

[Suh10] Y. S. Suh, "Orientation estimation using a quaternion-based indirect kalman filter with adaptive estimation of external acceleration," *IEEE Transactions on Instrumentation and Measurement*, vol. 59, no. 12, pp. 3296–3305, Dec 2010.

[VDX15] R. Valenti, I. Dryanovski, and J. Xiao, "Keeping a good attitude: A quaternion-based orientation filter for imus and margs," *Sensors*, vol. 15, no. 8, pp. 19 302–19 330, Aug 2015. [Online]. Available: http://dx.doi.org/10.3390/s150819302