Data Filtering and Sensor Fusion for Orientation Computing

Caus Danu[1]

***Abstract*— This paper focuses on presenting various general methods of sensor data filtering/fusion used in Attitude and Heading Reference Systems (AHRS). These are systems whose purpose is to estimate the orientation of a rigid body with respect to a reference frame. The paper attempts to compare a standard/-general solution of an Extended Kalman Filter with more domain restricted, "ad-hoc" solutions such as: the gradient descent based Madgwick Filter and the Nonlinear Complementary Mahony Filter. To make things more concrete, the final implementation and comparison results are based on the use of all three types of data filtering and fusion techniques on a KUKA robot and ARM-Cortex M4 evaluation board. The paper argues that the Extended Kalman Filter is a more flexible and powerful sensor fusion method, requiring however more computational resources as compared to the Madgvick and Mahony algorithms, which are therefore more suitable for less powerful hardware modules.**

I. Introduction

Nowadays, one of the most important things when it comes to the design and manufacturing of human-machine interfaces, mobile robots and mobile devices in general is to have an accurate and reliable way of estimating orientation. It is generally not feasible from a commercial and financial standpoint to use very expensive sensors in order to perform very reliable measurements and hence, manufacturers generally resort to low-cost chips. To compensate for the inaccuracies in measurements, all the sensors need to be fused together. The most important aspect of this fusion operation is actually data filtering. That is why these chips that contain all the necessary sensors, called MEMS chips (Micro-Electro-Mechanical Systems) also contain hardware accelerators dedicated specifically for the purpose of filtering sensory information. Typically, they contain a variation of the Extended Kalman Filter or, if the clock rate is not particularly high: either the Mahony or Madgwick Filters to save computational time.

Daily life applications of MEMS chips typically include:

- Human-machine interface in smartphones
- Human-machine interface in smartwatches
- Human-machine interface in game consoles

Other more "professional" applications include:

- Limb movement analysis for rehabilitation purposes [1]
- Performance evaluation of old people [2]
- Real-time systems for capturing body posture [3]

Lastly, more advanced applications in robotics are:

- Orientation calculation for Unmanned Airborne Vehicles (UAV) [4]
- Orientation calculation for Unmanned Underwater Vehicles (UUV) [5]
- Wheel slip estimation of industrial skid steered mobile robots [6]
- Attitude estimation for Micro-Aerial Vehicles (MAV) [7]

[1]7caus@informatik.uni-hamburg.de

II. Theoretical Knowledge

All the algorithms in this paper were implemented using data from only three types of sensors:

- **Accelerometer**
- **Magnetometer**
- **Gyroscope**

This is because the **Madgwick** and **Mahony** filters were analytically designed to have this data as inputs. The **Extended Kalman Filter (EKF)** however does not have this limitation. One can always add new data from other types of sensors and make the orientation more exact. To compare the algorithms on equal grounds, the EKF was also designed to work with only these three types of sensors.

The **Accelerometer** is a sensor that measures acceleration. If it has three axes, it will measure acceleration in each spatial component: x, y and z. The algorithms however make use of the **gravity** component, that is: the component along the negative z axis. The accelerometer outputs noisy signal and if integrated over time, this leads to very large accumulation of errors.

The **Magnetometer** is the sensor that measures magnetic field. It suffers from large amounts of noise coming especially from the fluctuating electrical fields that cause magnetic distortion (see [8]), embedded hard magnets in the device itself or soft magnets, i.e. magnetised components within the device. The magnetometer is used to calculate the **heading** vector.

The **Gyroscope** is the third type of sensor in the setup used to measure angular velocities along all three axes. Unlike the other two sensors, it does not suffer from noise as much since the noise gets integrated. However, the integration process leads to another problem: drift (see figure 1).

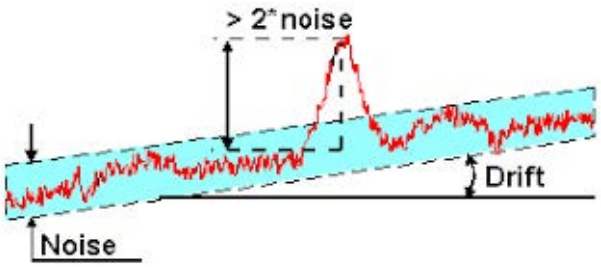

Fig. 1. Drift is a deviation from a true initial position, which is **NOT** the **base level** around which the noise signal oscillates. Courtesy of [9]

In general, the gyroscope is very good at dynamically measuring angular velocity, but there must be some mechanism to compensate for the bias and drift. This is where the other sensors come into play. By computing the gravity and heading vectors and fusing them together with the data coming from the gyroscope, the "true" estimate of angular velocity is calculated.

Since all the algorithms presented in this paper try to compute the orientation of a rigid body using the **quaternion** as an output, it is important to have at least an intuition of what this mathematical structure is and why it is so useful.

A quaternion is a 4-dimensional vector of the form:

$$\begin{aligned} {}^{A}_{B}q &= \begin{bmatrix} q1 & q2 & q3 & q4 \end{bmatrix} \\ &= \begin{bmatrix} cos\frac{\theta}{2} & -r_x sin\frac{\theta}{2} & -r_y sin\frac{\theta}{2} & -r_z sin\frac{\theta}{2} \end{bmatrix} \end{aligned} \tag{1}$$

Quaternions are also thought off as 4-dimensional complex numbers, where the q1 component is the real part (encoding the angle θ) and $\begin{bmatrix} q2 & q3 & q4 \end{bmatrix}$ is the imaginary vector component (encoding the axis of rotation). Hence, it is natural to think of quaternion multiplication operations as performing axis-angle rotations (see figure 2)

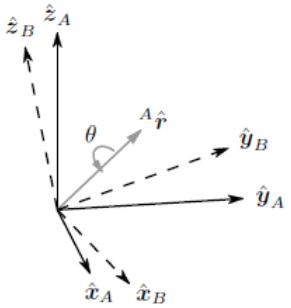

Fig. 2. Frame B is obtained from frame A by a quaternion rotation of angle θ around axis ${}^{A}\hat{r}$. Courtesy of [11]

One of the most useful things that they solve is the so called **Gimbal Lock** problem that for instance Euler Angles have. This problem is suggestively presented by figure 3, where 2 degrees of freedom are lost because of the alignment of all three gimbals.

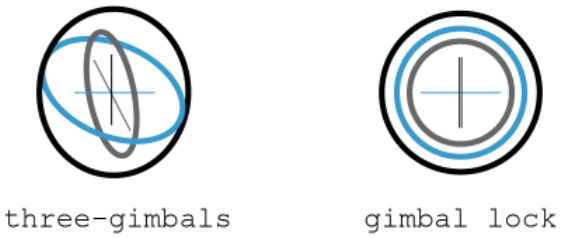

Fig. 3. Gimbal Lock resulting from the alignment of all 3 axes. This results in the loss of 2 degrees of freedom and hence, quaternions are used to overcome this problem. Courtesy of [10]

Note that there can also be a scenario where only 2 gimbals are aligned, resulting in the loss of one degree of freedom.

The three types of algorithms whose performances are compared in this paper are:

- A standard version of the **Extended Kalman Filter** (EKF)
- The gradient descent based **Madgwick Filter**
- The **Mahony nonlinear complementary filter**

To make things easier to comprehend, for the purposes of this paper, all algorithms are presented visually with the help of block diagrams and an intuition of the general behaviour they exhibit. For exact equations, this paper gives references to the exact resources the high level explanations are based upon.

The structure of the standard **Extended Kalman Filter** is presented in figure 4. Just like any Kalman Filter, the EKF has two important steps:

- **Prediction Step** that uses the process noise parameter $\boldsymbol{W}$ in order to make a prediction about the (augmented) state of the system $x_k = \begin{bmatrix} Q_k \\ \omega_k^v \end{bmatrix}$, where Q_k is the quaternion expressing the estimated orientation and ω_k^v is the angular velocity calculated (in the body frame v) at the k-th step.
- **Update Step** that updates the state of the system based on the **Kalman Gain** calculated prior to this step. This particular process makes use of the noise covariance matrix $\boldsymbol{V}$.

Kalman Gain ($\boldsymbol{K}$) parameter computed prior to the update step is very important since it determines what values will the system put more emphasis on: either the measured values from the sensors, or the predicted "belief" values from the prediction step. If $\boldsymbol{K}$ is closer to 0, it means that the noise variance encoded by $\boldsymbol{V}$ is large and therefore, the system will give a higher weight to the predicted values when calculating the new state. When $\boldsymbol{K}$ is closer to 1, it means that the noise is small and hence the measurements are accurate, and the algorithm will consider the measurement values a lot more when updating the state.

As the last step in the algorithm, the **quaternion normalization** is done. This has 2 functions:

- For showing attitude, magnitude has no relevance

and therefore it makes sense for mathematical simplification to remove it.
- To avoid numerical overflow issues on the processors on which the algorithm is running, because the algorithm is iterative in nature and it can lead to very high accumulation of the quaternion magnitude over time resulting in an overflow at some point.

For exact equations related to this particular version of the EKF see [14]

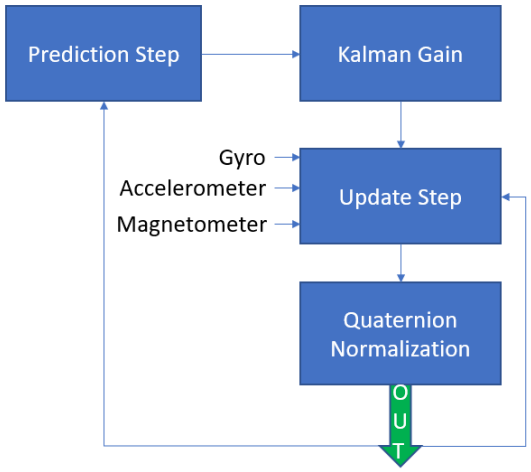

Fig. 4. Extended Kalman Filter Algorithm. Refactored block diagram with ideas taken from [12]. The algorithm has two important steps: **prediction step** to compute the belief of current attitude and **update step** to correct the belief by incorporating concrete measurements. Normalization is done to avoid numerical overflow, which is very likely to happen if the system has low bit resolution.

The **Madgwick Algorithm** fuses data from MARG (Magnetic, Angular Rate, and Gravity) sensor arrays. MARG chips are basically an extension of IMUs (Inertial Measurement Units) to also include magnetometers (since an IMU has by default only accelerometers and gyroscopes). This is an important point, since the version of this algorithm that does not make use of magnetic data is performing very poorly because of the inability to find a unique solution for the final attitude quaternion. Moreover, the algorithm is rigid in a sense, since unlike the EKF, it cannot make use of data from other types of sensors (for instance: altitude and vertical velocity derived from pressure sensors or GPS). The algorithm has two main steps (see figure 5):

- The magnetometer and accelerometer measurements are fused together using a tunable parameter β to come up with a quaternion estimate of the attitude. The gradient descent algorithm is used to minimize the error of this quaternion estimate.
- The gyroscope drift and bias errors are corrected using a tunable parameter ζ and the quaternion coming from the output of the gradient descent algorithm

Essentially, the general idea is to compute the **heading vector** using the magnetometer data and the **gravity vector** using the accelerometer data and then by fusing them together, a unique plane is determined that allows the algorithm to correct the drift of the gyroscope. The gradient descent algorithm is essentially used to minimize as much as possible the error between the true unique plane and the estimated one. The gyroscope drift is corrected by using the quaternion derivative. For exact equations of each step see [11]. As mentioned, the β and ζ parameters are tunable, however [11] suggests some optimal values, namely: 0.041 for β and 0.001 for ζ

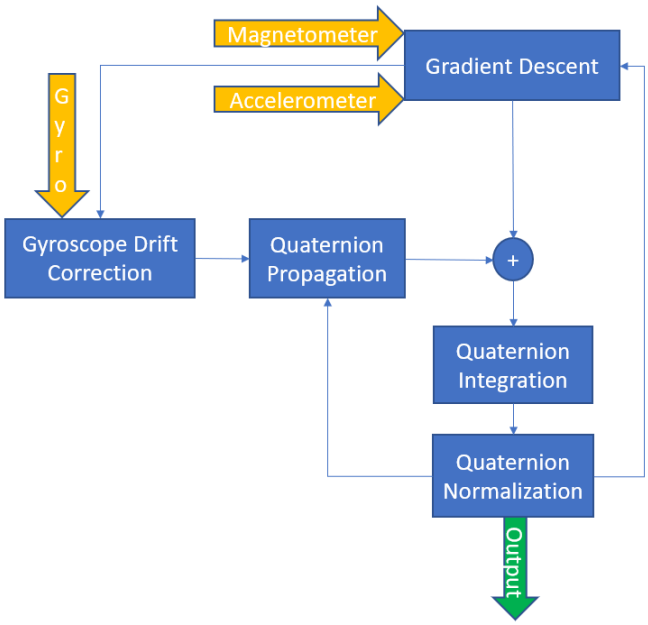

Fig. 5. Madgwick Algorithm. Refactored block diagram with ideas taken from [12]. The magnetometer and accelerometer data is filtered using a gradient descent algorithm and fused in order to correct the gyroscope drift. At each time step a quaternion increment/derivative is computed and fed into an integrator block to calculate the final attitude value. Normalization is done to avoid numerical overflow, which is very likely to happen on the less expensive and less powerful chips for which the Madgwick algorithm is intended.

The **Mahony noncomplementary filter** seen in figure 6 is a variation of the **Kalman Unscented Filter**. Just like the Madgwick Filter, it also computes an orientation error using the accelerometer and magnetometer measurement data. This error is then fed into a Proportional-Integral (PI) regulator/compensator block to compensate for the drift of the gyroscope, that is: to correct the measured angular velocity. Afterwards, the attitude change is obtained in the form of a quaternion derivative value at a given time step, which is subsequently propagated to the integrator block that accumulates all the discrete derivatives and outputs a quaternion estimate of the orientation. This particular algorithm has 2 tunable parameters: K_p and K_i used in the PI regulator block. The suggested values for them are: $K_p = 2$ and $K_i = 0.6$. This algorithm may suffer from systematic error accumulation when it is subject to rotations close to $\pm\pi$. For exact equations corresponding to each individual step in the block diagram, see [13]

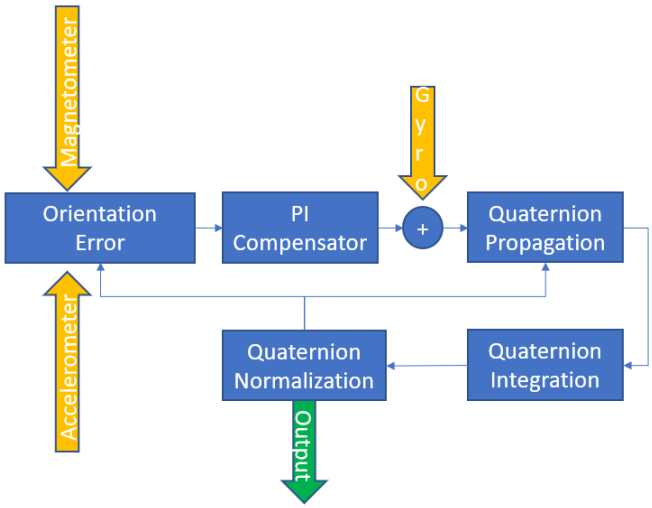

Fig. 6. Mahony Algorithm. Refactored block diagram with ideas taken from [12]. Accelerometer and magnetometer data is fused together to calculate an orientation error, which is subsequently used to correct the gyroscope drift. The correction step takes place via a Proportional-Integral Compensator block. The result at each time step is in the form of a quaternion derivative that subsequently needs to be cumulated by another integration block to compute the full quaternion value. Normalization is important to avoid numerical overflow, which is very likely to happen on the very inexpensive and computationally weak devices for which the Mahony algorithm is intended.

III. Setup

The experiment setup can be seen in figure 7. It consists of a KUKA Youbot robot, with an IMU mechanically mounted to its arm. On the back of the robot, an STM32F3 Discovery board is mounted. This board contains an ARM-Cortex M4 microcontroller (STM32F303VCT6), a 3-axis digital gyroscope (ST L3GD20) and a MEMS system (ST LSM303DLHC) consisting of a tri-axis accelerometer and a tri-axis magnetometer.

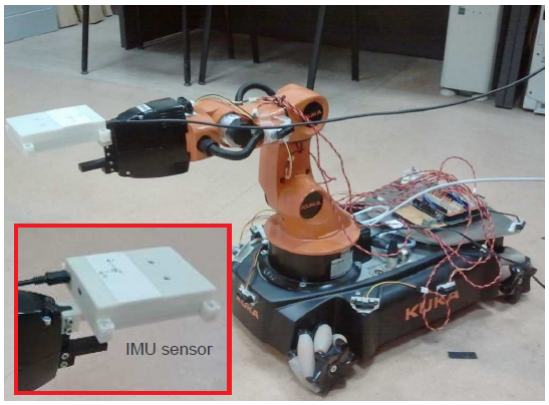

Fig. 7. KUKA Youbot. Courtesy of [14]. The IMU unit will be used to assess the algorithms and the robot arm encoders will act as ground truth calculators for the later comparisons and filter performance evaluations.

It is very important to re-calibrate the accelerometer and the magnetometer after the setup is done because a lot of systematic errors appear from the mounting process itself, even though the sensors are pre-calibrated by the original manufacturer. The calibration is performed using a 3D-ellipsoid fitting algorithm (see [15]) and results in a significant increase in accuracy as it can be seen in figures 8 and 9. The basic recalibration equation is:

$$\hat{y}^s = \lambda(y_f^s - b^s) \tag{2}$$

where $\hat{y}^s$ is the re-calibrated sensor output in the sensor frame, y_f^s is the factory calibrated output in the sensor frame, λ represents the scale factors and b^s represents the offset/bias factors in the sensor frame.

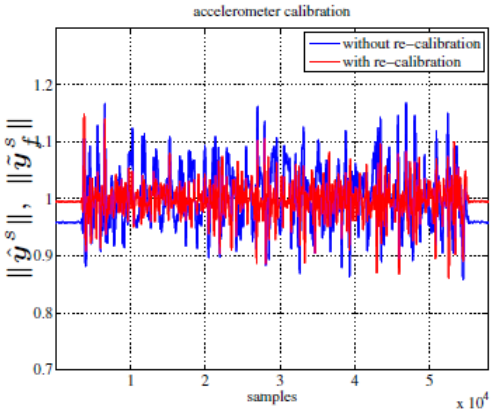

Fig. 8. Accelerometer RE-Calibration. Courtesy of [14]. $\hat{y}^s$ is the re-calibrated sensor output in the sensor frame, y_f^s is the factory calibrated output in the sensor frame. Note that accelerometer data is generally less noisy than magnetometer data (see figure 9)

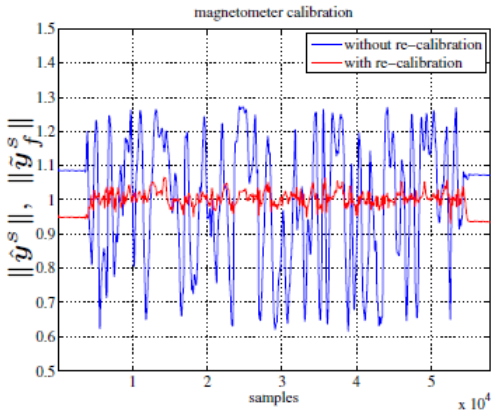

Fig. 9. Magnetometer RE-Calibration. Courtesy of [14]. $\hat{y}^s$ is the re-calibrated sensor output in the sensor frame, y_f^s is the factory calibrated output in the sensor frame. Note that magnetometer data is generally a lot more noisy than accelerometer data (see figure 8)

Recalibration results in standard deviations of 2.8% for the magnetometer, compared to 17.9% before recalibration and 3.1% for the accelerometer, compared to 5.2% prior. As a side note, because the magnetometer is more susceptible to noise and disturbances, as it can be seen from the numbers above, this fact will be later on used to measure the performance of all three algorithms

in a noisy environment created specifically by inducing external magnetic fields (see paragraph VI-.0.b and figure 12).

IV. Approach

The experiment consists of several parts. The first interesting question to answer is which of the three algorithms performs better in a **static scenario**. The premise is that even though the robot manipulator does not move, the three types of sensors always measure a noisy signal oscillating around the base value. Hence, the algorithms should reduce as much as possible these errors and ideally have an error of zero degrees in terms of Euler angles.

The second interesting question is what happens in a dynamic scenario. If the robot manipulator is rotated, what are the root mean square errors observed in terms of Euler angles (i.e. differences from the true Roll, Pitch and Yaw values) ? This scenario is subdivided in two subcases:

- **Slow Trajectory** case, where rotations are performed with an average speed of 18 degrees/second
- **Fast Trajectory** case, where rotations are performed with an average speed of 45 degrees/second

Finally, the last part of the experiment tries to quantify the computational burden of each one of the three algorithms, both in a software environment and a hardware embedded system environment.

V. Experiment execution

The experiment is executed using the 5-dof (5 degrees-of-freedom) robot arm. The sampling frequency of the robot encoders is 40 Hz. This frequency is sufficient in order to not allow signal aliasing effects even for the fast trajectory of 45 degrees per second (for more on the topic of sampling and aliasing see [16]). The robot arm acts as a **ground truth** calculator, since the joint encoders are very exact at computing/measuring the true rotational angles. In a first phase, the robot arm rotates in such a way so that the Roll, Pitch and Yaw are calculated individually, that is: a rotation is performed along one axis only. In a second phase, the Yaw, Pitch and Roll are calculated simultaneously by executing complex rotations along all three axes at the same time. The first phase is used merely to obtain a reference attitude for each individual axis.

For the last part of the experiment, the computational burden is estimated by implementing the algorithms first in a software environment using Matlab and Simulink. The algorithms are executed on an I7 processor with 1.6 GHz and the tic-toc Matlab functions are used to measure the execution cycle. The algorithms are afterwards implemented in an embedded-system with ARM Cortex M4 processor using Chibi/OS real time operating system and the ARM Cortex Microcontroller Software Interface Standard (CMSIS) DSP library.

VI. Results

a) Normal Noise Environment: After the execution of the experiment, figure 10 compiles the results for the Root Mean Square Error (RMSE) between the true Euler Angles as computed by the robot joint encoders and the angles calculated by the IMU and MARG units using the three types of algorithms. The RMSEs are presented for the static case, the slow and fast trajectory cases for each axis individually, that is: for the roll, pitch and yaw.

Euler angles [°]	EKF	Madgwick	Mahony
Roll (static)	0.04	0.03	0.02
Pitch (static)	0.01	0.05	0.05
Yaw (static)	0.30	1.92	1.85
Roll (slow)	4.71	4.85	5.07
Pitch (slow)	1.91	2.65	2.89
Yaw (slow)	5.19	5.13	5.67
Roll (fast)	6.55	6.51	6.69
Pitch (fast)	2.83	3.34	2.85
Yaw (fast)	6.71	7.07	6.92

Fig. 10. Root Mean Square Errors. Courtesy of [14]. The EKF is generally better in all scenarios: static, slow and fast trajectory movement. When it performs worse than the other two algorithms, the difference is negligible.

b) High Noise Environment: The experiment was also performed with a slight twist in order to see what happens in a very noisy environment. To create such an environment, large magnetic fluctuations were generated in order to induce a large variance in the magnetometer measurements (see figure 11 as opposed to figure 12). As a reminder, the magnetometer data is used to calculate the heading of the robot, which is later on fused with the gravity vector calculated using the accelerometer.

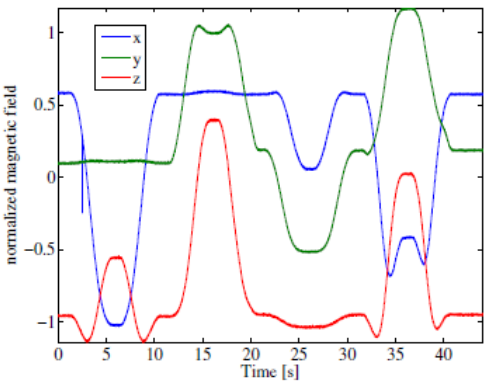

Fig. 11. Low noise magnetic field. Courtesy of [14]. To be compared with figure 12

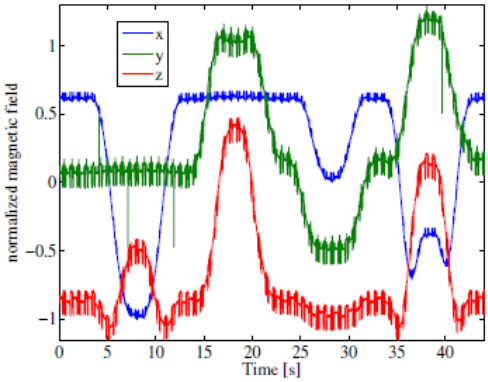

Fig. 12. High cyclic noise magnetic field. Courtesy of [14]. In order to test the robustness of the filtering and fusion algorithms, the magnetic field is distorted because it is the easiest way to induce external noise to signals that will later on be used as measured values by the filters.

Figure 13 shows the calculated roll, pitch and yaw using the three algorithms in this noisy scenario.

Euler angles [°]	EKF	Madgwick	Mahony
Roll	5.05	5.54	5.87
Pitch	3.24	3.93	4.53
Yaw	5.93	6.27	6.66

Fig. 13. Root Mean Square Errors with noisy measurements. Courtesy of [14]. The EKF is slightly better than its counterparts in a noisy environment.

c) Computational Performance: Lastly, figure 14 shows the execution cycle in milliseconds for each individual algorithm, first in a Matlab/Simulink software environment running on a generic I7 processor and then in an embedded system hardware environment.

Algorithm	Matlab/Simulink [ms]	Embedded System [ms]
EKF	0.1	2.7
Madgwick	0.017	0.15
Mahony	0.014	0.11

Fig. 14. Execution Time in Hardware and Software. Courtesy of [14]. The EKF takes much more time in both software and hardware environments/ on a generic or more specialized processors.

VII. Evaluation

Figure 10 shows the Euler angles Root Mean Square Error for all three algorithms in the case of a slow trajectory (18 deg/s), fast trajectory (45 deg/s) and static case as well. As it can be observed, the Extended Kalman Filter has a better estimation and lower errors in almost all cases, and when it technically is worse than the other algorithms, it has a very negligible difference in performance.

As it can be seen from figure 13 in the case of a very noisy environment (by deliberately introducing large magnetic fluctuations), all the 3 algorithms have a decrease in performance. None of the algorithms can be labeled as being absolutely better than the others for such a scenario.

So far, one might argue that the Extended Kalman Filter is generally better. Judging purely from a performance point of view, the answer would be affirmative. But of course this performance comes at a price. Figure 14 shows that in a software implementation running on a generic processor (not on an FPGA for instance, which allows for highly configurable hardware resulting in a significant speed-up for any particular algorithm), the EKF is roughly 5 to 7 times slower than the other two algorithms. In a hardware environment implementation, the time difference is not as drastic, but still the EKF is 2 to 3 times slower. All this suggests that there is always a trade-off and in order to have better accuracy, one can use the EKF, but this will definitely increase the computational burden on the processor on which the EKF is running. On the other hand, in some real-time systems, where each iteration counts as an important additional burden or in systems with lower clock rates, the Mahony or Madgwick filter can be an acceptable and preferred solution.

VIII. Conclusion

In conclusion, one can say that all three filters presented in this paper are suitable solutions to implement sensor fusion algorithms on devices that contain IMU (and MARG) units. Depending on the concrete application however, it is better to choose one over the others. The Extended Kalman Filter is generally more accurate and is not limited to only three sensors. It is generally easy to integrate data from new sensors provided that the statistics of the noise is known for those new sensors. However, the Extended Kalman Filter comes at a higher computational cost. During the prediction and update process, a lot of high dimensional matrices (exact dimensionality increases with more sensors employed) need to be added and multiplied together at each time step. This is not the case when using the Mahony Filter or Madgwick Filter. They are analytically derived so that they use as little computation steps as possible. This allows them to run on devices that have a small clock rate and using sensors that have low sampling rate of measurements. The disadvantage trade-off is that they perform slightly worse in terms of errors from the true measured value and don't have the flexibility of introducing new sensory information other than from the accelerometer, magnetometer and gyroscope. Between the Mahony and the Madqwick algorithms, the Mahony noncompementary filter is generally more lightweight and suitable for very low-cost hardware with limited performance. However, the application might be restricted to scenarios where rotations are in the interval of $\pm\pi$. That is because, unlike the Madgwick filter, this algorithm has problems reconstructing the attitude if rotations go beyond $\pm\pi$ and the problems manifest themselves as an accumulation of systematic errors over time. In general, if hardware efficiency is a concern and the available sensors are only the accelerometer, magnetometer and gyroscope, without the need of adding

further sensors, the Madgwick algorithm would be the best choice.

References

[1] I. Prayudi and Kim Doik. Design and implementation of imu-based human arm motion capture system. In Int. Conf. on Mechatronics and Automation (ICMA), pages 670–675, 2012.

[2] M. Zecca, K. Saito, S. Sessa, L. Bartolomeo, Z. Lin, S. Cosentino, H. Ishii, T. Ikai, and A. Takanishi. Use of an ultra-miniaturized imu-based motion capture system for objective evaluation and assessment of walking skills. In 35th Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society (EMBC), pages 4883–4886, 2013.

[3] J. Ziegler, H. Kretzschmar, C. Stachniss, G. Grisetti, and W. Burgard. Accurate human motion capture in large areas by combining imu- and laser-based people tracking. In IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), pages 86–91, 2011.

[4] Han Joong-hee, J.H. Kwon, Lee Impyeong, and Choi Kyoungah. Position and attitude determination for uav-based gps, imu and at without gcps. In Int. Workshop on Multi-Platform/Multi-Sensor Remote Sensing and Mapping (M2RSM), pages 1–5, 2011.

[5] Ho-Sung Kim, Hyeung-Sik Choi, Jong su Yoon, and P.I. Ro. Study on ahrs sensor for unmanned underwater vehicle. Int. J. of Ocean System Engineering, 1:165–170, 2011.

[6] Yi Jingang, Zhang Junjie, Song Dezhen, and S. Jayasuriya. Imubased localization and slip estimation for skid-steered mobile robots. In IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), pages 2845–2850, 2007.

[7] ClarkN. Taylor. Enabling navigation of mavs through inertial, vision, and air pressure sensor fusion. In Hernsoo Hahn, Hanseok Ko, and Sukhan Lee, editors, Multisensor Fusion and Integration for Intelligent Systems, volume 35 of Lecture Notes in Electrical Engineering, pages 143–158. Springer Berlin Heidelberg, 2009.

[8] Electrodynamics `http://www2.oberlin.edu/physics/dstyer/Electrodynamics/ChangeChange.pdf` Accessed February 1, 2018

[9] Noise and drift `http://hplc.chem.shu.edu/NEW/HPLC_Book/Detectors/det_nise.html` Accessed February 1, 2018

[10] Levison M. The Quaternions, Physics 212 `http://ffden-2.phys.uaf.edu/webproj/212_spring_2014/Michael_Levison/Michael%20Levison/rotation/index.html` Accessed February 1, 2018

[11] Madgwick S. An efficient orientation filter for inertial and inertial/magnetic sensor arrays. Report x-io and University of Bristol (UK), 2010.

[12] Cirillo A., Cirillo P., De Maria G.,Natale C., Pirozzi S. A comparison of multisensor attitude estimation algorithms. Researchgate `https://www.researchgate.net/profile/Pasquale_Cirillo/publication/303738116_A_comparison_of_multisensor_attitude_estimation_algorithms/links/5750181208aeb753e7b4a0c0/A-comparison-of-multisensor-attitude-estimation-algorithms.pdf`, 2016. Accessed February 1, 2018

[13] Mahony, R., Hamel, T. and Pflimlin, J.M. Nonlinear complementary filters on the special orthogonal group. IEEE Transactions on automatic control, 53(5), pp.1203-1218, 2008.

[14] Cavallo, A., Cirillo, A., Cirillo, P., De Maria, G., Falco, P., Natale, C. and Pirozzi, S. Experimental comparison of sensor fusion algorithms for attitude estimation. IFAC Proceedings Volumes, 47(3), pp.7585-7591, 2014.

[15] Malyugina, A., Igudesman, K. and Chickrin, D. Least-squares fitting of a three-dimensional ellipsoid to noisy data. Applied Mathematical Sciences, 8(149), pp.7409-7421, 2014.

[16] Schesser J. Sampling and Aliasing, Biomedical Computing `https://web.njit.edu/~joelsd/Fundamentals/coursework/BME310computingcw6.pdf` Accessed February 1, 2018

[17] Thrun, S., Burgard, W. and Fox, D. Probabilistic robotics. MIT press. pp. 48-53, 2005.

Eidesstattliche Erklärung

Hiermit versichere ich, Caus Danu, an Eides statt, dass ich die vorliegende Seminararbeit mit dem Titel *Data Filtering and Sensor Fusion for Orientation Computing*, sowie die Präsentationsfolien zu dem dazugehörigen mündlichen Vortrag ohne fremde Hilfe angefertigt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Alle Teile, die wörtlich oder sinngemäß einer Veröffentlichung entstammen sind als solche kenntlich gemacht.

Die Arbeit wurde in dieser oder ähnlicher Form noch nicht veröffentlicht, einer anderen Prüfungsbehörde vorgelegt oder als Studien- oder Prüfungsleistung eingereicht.

Declaration of an Oath

Hereby I, Caus Danu, declare that I have authored this thesis, titled *Data Filtering and Sensor Fusion for Orientation Computing*, and the presentation slides for the associated oral presentation independently and unaided. Furthermore, I confirm that I have not used other than the declarared sources / resources.

I have explicitly marked all material which has been quoted either literally or by content from the used sources

This thesis, in same or similar form, has not been publicated, presented to an examination board or submitted as an exam or course achievement.

Hamburg, February 3, 2018

Caus Danu