A Comparative Analysis of Sensor Fusion Algorithms for Miniature IMU Measurements

Kristel Çoçoli and Leonardo Badia
Dept. of Information Engineering (DEI), University of Padova, Italy
email: {kristel.cocoli@studenti. , leonardo.badia@} unipd.it

***Abstract*—Inertial measurement units, typically consisting of tri-axis gyroscopes and accelerometers, are very important for a plethora of applications in the upcoming Tactile Internet. Yet, especially for miniature devices relying on cheap electronics, their measurements are often inaccurate and subject to gyroscope drift, which implies the necessity for sensor fusion algorithms. These are critical for estimating the orientation of an object by combining multiple measurements, and also require fast computation to be useful in practice. This paper presents a comparative analysis of a standard trigonometry computation, shown to be ineffective, with popular candidate algorithms, namely, Kalman, Mahony, and Madgwick, with a specific focus on their suitability for small embedded systems. The algorithms were evaluated on experimental data based on their accuracy and computational efficiency.**

***Index Terms*—Sensor fusion; Kalman filter; Madgwick filter; Mahony filter; Embedded systems; Tactile Internet.**

I. Introduction

Software representations designed to accurately reflect physical reality, better known as digital twins, are rapidly progressing in many fields. The so-called Tactile Internet, pairing ultra low latency with high reliability, can revolutionize many applications like industrial robotics, biomedical instrumentation, virtual reality, autonomous cars and aerial vehicles [1]–[6].

In these contexts, it may be required to estimate the orientation and position of objects in 3D space. To this end, inertial measurement units (IMUs) are widely used, typically consisting of accelerometers and gyroscopes, which provide measurements of linear and angular motion, respectively. In consumer electronics, these measurements are subject to errors and noise, which can significantly affect the accuracy and reliability of the orientation estimation. Low-cost sensors have inherent drawbacks, including temperature gradients, vibrations, nonlinearity, shocks [7]. To obtain reliable results, miniature IMU sensor measurements can be combined together [8], [9], using sensor fusion algorithms based on techniques such as Kalman, Madgwick, and Mahony filters.

The Kalman filter is a widely used approach for estimating the state of a dynamic system based on noisy measurements. It uses a probabilistic model to estimate the true state of the system, based on previous measurements and knowledge of the system dynamics. The Madgwick and Mahony filters are other popular open-source algorithms for orientation estimation using IMUs. They use quaternion-based representations of orientation and utilize different techniques for fusing accelerometer and gyroscope measurements, including gradient descent and complementary filtering.

While these algorithms have been studied in the literature, there is still ongoing research to improve their performance and assess their suitability for different applications. Sensor fusion can become a challenging problem due to limited computational power of miniature devices, which rely on cheap electronics, and make it difficult to perform complex sensor fusion algorithms in real time. Also, inexpensive sensors used in miniature devices may have lower accuracy and higher noise. Thus, accurate measurements are often unavailable, and error propagation is possible [10]–[12].

In this paper, we present a comprehensive comparison of the Kalman, Madgwick, and Mahony filters for orientation estimation using miniature IMUs. We have used the no-filter method as baseline, to emphasize the importance of sensor fusion algorithms. We evaluate the performance of these filters using a dataset of 3-axis accelerometer and gyroscope measurements, collected in both static and dynamic conditions. We analyze the accuracy, stability, and computational complexity of each filter and compare their performance in different scenarios. We seek to compare possible solutions to this task, with specific focus on low quality sensors, implemented in embedded systems, and requiring little control data, so as to allow for an efficient connectivity and a fresh information exchange. Specifically, we consider an MPU6050 sensor, which combines a MEMS gyroscope and accelerometer, where a 6-DOF IMU for the pitch and roll angles is implemented.

Existing papers [13], [14] perform partial comparisons of these filtering techniques. We analyze and compare all of them together, with the same data. Also, we consider an easily, affordable, and reproducible setup, i.e., a combination of MPU6050 sensor and Arduino Nano. In addition, we discuss computational complexity and suitability for real-time operation of the proposed techniques, which is key for a realistic implementation in the Tactile Internet [15].

The rest of the paper is organized as follows. In Section II, we introduce the mathematical preliminaries. In Section III, we describe the setup and the different kinds of filters tested. Section IV details the comparative experiments performed and the numerical results. Finally, we conclude in Section V.

II. Mathematical model

Roll, pitch, and yaw are the three Euler angles that describe the orientation of an object in three-dimensional space. In fusion algorithms, measurements of these angles from multiple sensors are combined to estimate the orientation in real-time.

Roll ϕ is the angle of rotation around the longitudinal (or x) axis, which is also perpendicular to the horizontal plane. A positive roll angle indicates that the object is tilted to the right, with the right side of the object pointing downwards and the left side pointing upwards. A negative roll angle indicates the opposite, with the left side of the object pointing downwards and the right side pointing upwards.

Pitch θ is the angle of rotation around the lateral (or y) axis, perpendicular to the horizontal plane. A positive pitch angle indicates that the object is tilted upwards, with the front pointing upwards and the back pointing downwards. A negative pitch angle indicates the opposite.

Yaw ψ refers to the rotational motion of an object around its vertical axis. Specifically, yaw is the angular displacement of an object with respect to a fixed reference frame or coordinate system, measured in degrees or radians.

In this paper, we measure roll and pitch angles only, because of the limitations presented by the lack of a magnetometer or GPS. Accelerometers can measure the tilt of an object relative to gravity, which can be used to estimate the pitch and roll angles. However, accelerometers cannot directly measure yaw [16]. Instead, yaw can be estimated by combining the accelerometer readings with readings from other sensors, such as magnetometers or GPS, using fusion algorithms.

The most straightforward way to compute the Euler angles from the acceleration and angular rate readout is the traditional trigonometric computation, detailed next.

A. Rotational matrix and accelerometer

From the accelerometer sensor, we read the linear acceleration of the object. If we use the North-East-Down (NED) convention for the Euler angles, we expect that a perfectly flat sensor reads an acceleration vector as $(a_x, a_y, a_z) = (0, 0, -g)$, where the negative sign accounts for the direction of the positive z-axis in this convention. Thus, we expect to measure the Earth acceleration for that component, while the others are 0. Every possible rotation to this reference frame can be described with a linear transformation of the basis vectors. These linear transformations will produce distinct rotation matrices, one per each Euler angle. The resulting expression for an arbitrary rotation of the sensor is

$$\begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} = \mathbf{R} \begin{pmatrix} 0 \\ 0 \\ -g \end{pmatrix} \tag{1}$$

Denoting $\cos(\alpha) = c_\alpha$ and $\sin(\alpha) = s_\alpha$, the rotation matrix $\mathbf{R}$ in terms of the Euler angles is

$$\mathbf{R} = \begin{bmatrix} c_\psi c_\theta & s_\psi c_\theta & -s_\theta \\ c_\psi s_\theta s_\phi - s_\psi c_\phi & s_\psi s_\theta s_\phi + c_\psi c_\phi & c_\theta s_\phi \\ c_\psi s_\theta c_\phi + s_\psi s_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi & c_\theta c_\phi \end{bmatrix} \tag{2}$$

Due to the inability to measure the yaw [16], we set $\psi = 0$ and it follows that $s_\psi = 0$ and $c_\psi = 1$, which implies

$$\begin{aligned} a_x &= g \sin(\theta) \\ a_y &= -g \cos(\theta) \sin(\phi) \\ a_z &= -g \cos(\theta) \cos(\phi) \end{aligned} \tag{3}$$

Assuming that (a_x, a_y, a_z) is read from the accelerometer, pitch and roll angles are found through algebraic manipulations as

$$\begin{aligned} \theta &= \arcsin\left(\frac{a_x}{g}\right) \\ \phi &= \arctan\left(\frac{a_y}{a_z}\right). \end{aligned} \tag{4}$$

B. Gyroscope

The gyroscope measures the angular velocity for the rotation of the rigid body along its three axes. This can be used to assess new angular values, if its initial state is known [17]. We can denote this measurement as $\vec{\omega} = (p, q, r)$. These values are measured in the sensor's frame, after the rotation. We want to translate this information into the initial frame of reference, which would give us the Euler angles rate of change. By omitting the rate of change of the yaw angle, we obtain the angular speeds $\dot{\phi}$ and $\dot{\theta}$ for roll and pitch, respectively.

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} 1 & \sin\phi \tan\theta & \cos\phi \tan\theta \\ 0 & \cos\phi & -\sin\phi \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} \tag{5}$$

By calculating the rate of change of the Euler angles, and through proper integration over the time step, which in our case is the time difference between two samples acquired by the sensor, and finally summing the change with the previous value of the angle, we get the values for pitch and roll angles.

C. Quaternions

A quaternion $\widehat{\boldsymbol{q}}$ can be used to rotate a three dimensional vector $\boldsymbol{v}_0$ into $\boldsymbol{v}_1$ using the relationship

$$\boldsymbol{v}_1 = \widehat{\boldsymbol{q}} \times \boldsymbol{v}_0 \times \widehat{\boldsymbol{q}^*} \tag{6}$$

The orientation described by $\widehat{\boldsymbol{q}}$ can be represented as the rotation matrix $\mathbf{Q}$.

$$\mathbf{Q} = \begin{bmatrix} 2q_1^2{-}1{+}2q_2^2 & 2(q_2q_3{-}q_1q_4) & 2(q_2q_4{-}q_1q_3) \\ 2(q_2q_3{-}q_1q_4) & 2q_1^2{-}1{+}2q_3^2 & 2(q_3q_4{+}q_1q_2) \\ 2(q_2q_4{+}q_1q_3) & 2(q_3q_4{-}q_2q_2) & 2q_1^2{-}1{+}2q_4^2 \end{bmatrix} \tag{7}$$

The Euler angle representation of $\widehat{\boldsymbol{q}}$ [18] is defined by the following equations, where ψ, θ, ϕ indicate the orientation of the frame. We use atan2 [19] because the calculated angle can be between $-\pi$ and π.

$$\begin{aligned} \psi &= \operatorname{atan2}(2q_2q_3 - 2q_1q_4, 2q_1^2 + 2q_2^2 - 1) \\ \theta &= -\sin^{-1}(2q_2q_4 + 2q_1q_3) \\ \phi &= \operatorname{atan2}(2q_3q_4 - 2q_1q_2, 2q_1^2 + 2q_4^2 - 1) \end{aligned} \tag{8}$$

D. Problems

The above mentioned methods to compute the Euler angles work only in an ideal setup, but in a real system and especially with low quality electronics, we suffer from high frequency noise and time varying biases in the measurements. The bias is even more destructive in the gyroscope readout, since to retrieve the angle values we must integrate for each sampling time. Thus, the bias error accumulates and may grow in value over time [12], causing the so-called *gyroscope drift*. Ideally, we want to have a low pass filtering action to remove the high frequency components from the read signal and also a high pass filtering action with cutoff at very low frequencies to remove any low frequency bias without losing useful information from the signal itself. This prompts us to use the filtering schemes described in the next section.

III. Experiments and Methods

We used a MPU6050 microprocessor unit [20] for data acquisition, which combines a MEMS gyroscope and accelerometer. The MPU6050 is a popular sensor for orientation estimation and motion tracking applications due to its low cost, small size, and acceptable accuracy. The filtering is implemented using an Arduino Nano [21] board with an ATmega328P microcontroller, which is a widely used platform for prototyping and developing embedded systems.

To ensure a fair comparison between the filters, the same input data and integration periods are provided to each filter. The integration period includes the duration of the dead time between sensor readouts, during which a timer starts at the initial data readout and stops at the next readout operation. This time interval is utilized for processing, and is taken into account in the filter calculations. Despite the sequential execution of the filters, the same sensor readout data is held and sampled until all the filters have processed the input to ensure an equitable comparison of their outputs.

Two thousand data entries have been gathered for each filter, regarding the pitch, roll, and time. We considered two scenarios: (i) when the sensor is stationary and (ii) in movement, displacing the sensor. By comparing the outputs of the filters in the same input data and integration period, the performance of each filter can be evaluated and compared. We provide insights into the behavior and performance of the Mahony filter, Kalman filter, and Madgwick filter when used for orientation estimation with the MPU6050 sensor, so as to identify guidelines to choose the most appropriate filter, depending on the specific application.

We compared four different sensor fusion techniques to determine the most effective approach for our specific application. The first technique we experimented with is the traditional trigonometry method discussed in the previous section, which involves combining the sensor data without any additional filtering. This method provides a baseline for comparison with the other methods. We are interested in understanding the impact of using a filter on sensor fusion accuracy and comparing the effectiveness of the different filters we have implemented. By incorporating a filter, we aim to reduce the noise and errors in the sensor data, resulting in more accurate and reliable information.

A. Kalman filter

The Kalman filter can be used to estimate the status of linear systems [22]. This filter algorithm consists of two stages: prediction and update, sometimes [23] also referred to as "propagation" and "correction." A synthetic version of Kalman Filter Algorithm is reported in Table I.

TABLE I
Steps of the Kalman Filter

Description	Equation
Kalman Gain	$K_k = \mathbf{P}'_k\mathbf{H}^T(\mathbf{H}\mathbf{P}'_k\mathbf{H}^T + \mathbf{R})^{-1}$
Update Estimate	$\hat{\mathbf{x}}_k = \hat{\mathbf{x}'}_k + K_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}'}_k)$
Update Covariance	$\mathbf{P}_k = (\mathbf{I} - K_k\mathbf{H})\mathbf{P}'_k$
Project into k+1	$\hat{\mathbf{x}'}_{k+1} = \mathbf{\Phi}\hat{\mathbf{x}}_k$
	$\mathbf{P}_{k+1} = \mathbf{\Phi}\mathbf{P}_k\mathbf{\Phi}^T + \mathbf{Q}$

The first equation represents the expression for the Kalman gain in a Kalman filter, where K_k is the Kalman gain at time step k, $\mathbf{P}'_k$ is the predicted error covariance matrix, $\mathbf{H}$ is the measurement matrix, and $\mathbf{R}$ is the measurement noise covariance matrix [24]. The second equation represents the update step in a Kalman filter, where $\mathbf{x}_k$ is the updated state estimate at time step k, $\mathbf{x}'_k$ is the predicted state estimate, $\mathbf{z}_k$ is the measurement at time step k. The third equation represents the update step for the error covariance matrix in a Kalman filter, where $\mathbf{P}_k$ is the updated error covariance matrix at time step k. The matrix subtraction $(\mathbf{I} - K_k\mathbf{H})$ is often referred to as the Kalman gain update.

For the projection, the first equation, $\hat{\mathbf{x}'}_{k+1} = \mathbf{\Phi}\hat{\mathbf{x}}_k$, predicts the state of the system at the next time step based on the current state. Here, $\hat{\mathbf{x}}_k$ is the estimated state of the system at time step k, and $\mathbf{\Phi}$ is the state transition matrix, which describes how the state of the system evolves over time. Multiplying $\hat{\mathbf{x}}_k$ by $\mathbf{\Phi}$ gives the predicted state of the system at time step $k + 1$, denoted by $\hat{\mathbf{x}'}_{k+1}$.

The second equation, $\mathbf{P}_{k+1} = \mathbf{\Phi}\mathbf{P}_k\mathbf{\Phi}^T + \mathbf{Q}$, predicts the error covariance matrix at the next step. Matrix $\mathbf{\Phi}\mathbf{P}_k\mathbf{\Phi}^T$ represents the propagation of the uncertainty in the estimated state to the next step. Matrix $\mathbf{Q}$ is the process noise covariance matrix, which represents the uncertainty in the dynamics of the system that is not accounted for by the state transition matrix. The sum of $\mathbf{\Phi}\mathbf{P}_k\mathbf{\Phi}^T$ and $\mathbf{Q}$ gives the predicted error covariance matrix at time step k+1, denoted by $\mathbf{P}_{k+1}$.

Together, these two equations allow us to predict the state of the system at the next time step, and estimate the uncertainty in the state prediction. The prediction step is followed by the correction step, which updates the state estimate based on new measurements. The combination of the prediction and correction steps allows the Kalman filter to estimate the state of a system with high accuracy, even in the presence of noise and uncertainty.

B. Mahony filter

The Mahony filter [25] tries to improve the estimates from low-quality measurements through a quaternion-based approach, to represent the orientation of the device in 3D space. It uses a proportional-integral-derivative (PID) control algorithm to adjust the estimated orientation based on the difference between the measured and estimated sensor readings. The algorithm is designed to minimize errors over time by adjusting the gain parameters of the filter.

This filter is known for its ability to provide accurate and stable estimates of orientation even in the presence of external disturbances such as vibration or magnetic interference. It is widely used in a variety of applications, including robotics, aerospace, and virtual reality [2], [13]. However, it requires careful tuning of its gain parameters to achieve the best performance. A pseudocode of the Mahony filter, as was implemented in the sensor is reported in Algorithm 1.

Algorithm 1 Mahony Filter Algorithm for IMU (Accelerometer and Gyroscope Only)

```
1. Set algorithm coefficients $K_i, K_p$ and initialize quaternion $q_1 = 1,\ q_2 = q_3 = q_4 = 0$
   while: sensor data is available
2. Read accelerometer measurements $a_x, a_y, a_z$ and gyroscope measurements $g_x, g_y, g_z$
3. Compute orientation error from accelerometer data, where $e_{i,t}$ represents the integral error of the measurements at time $t$.
   $$\mathbf{e}_{t+1} = \begin{bmatrix} a_{x,t} \\ a_{y,t} \\ a_{z,t} \end{bmatrix} \times \begin{bmatrix} 2(q_2q_4 - q_1q_3) \\ 2(q_1q_2 + q_3q_4) \\ (q_1^2 - q_2^2 - q_3^2 + q_4^2) \end{bmatrix} \quad (9)$$
   $$\mathbf{e}_{i,t+1} = \mathbf{e}_{i,t} + \mathbf{e}_{t+1}\Delta t \quad (10)$$
4. Update angular velocity computed from gyroscope with the $K_i$ and $K_p$ terms using feedback (fusion)
   $$\omega_{t+1} = \omega_t + K_p e_{t+1} + K_i e_{i,t+1} \quad (11)$$
5. Compute orientation increment from gyroscope measurements
   $$\dot{q}_{\omega,t+1} = \frac{1}{2}\hat{q}_t \otimes \left[0, \omega_{t+1}\right]^T \quad (12)$$
6. Numerical integration
   $$q_{t+1} = \hat{q}_t + \Delta t \dot{q}_{\omega,t+1} \quad (13)$$
   endwhile
```

C. Madgwick filter

The Madgwick filter also uses quaternions [26] and evaluates the orientation, represented as a quaternion, by fusing/combining estimates, i.e., integrating gyroscope movements with the direction obtained by the accelerometer.

The gyroscope's estimates are relied upon for short time intervals and rapid movements, as they provide accurate depictions. On the other hand, the accelerometer estimates are utilized to provide accurate directions for compensating long-term drift of the gyroscope.

The Madgwick filter used in this paper is relatively simple and can be implemented in most programming languages. It involves calculating the error between the predicted and measured quaternion values and then using that error to update the quaternion estimate. The filter performance can be improved by fine-tuning the algorithm parameters and adjusting the sensor fusion weights to suit specific application requirements. Overall, the Madgwick filter is robust and efficient for attitude estimation and can provide accurate orientation estimates in real-time. The pseudocode of the Madgwick filter, as implemented in the experiments, is shown in Algorithm 2.

Algorithm 2 Madgwick Filter Algorithm for IMU (Accelerometer and Gyroscope Only)

```
1. Set algorithm gain $\beta$ and initialize quaternion $q_1 = 1,\ q_2 = q_3 = q_4 = 0$
   while: sensor data is available
2. Read accelerometer measurements $a_x, a_y, a_z$ and gyroscope measurements $g_x, g_y, g_z$
3. Normalize the accelerometer measurements
4. Calculate the Jacobian matrix and compute the gradient of the cost function
   $$\nabla f = J^T f \quad (14)$$
   $$f = \begin{bmatrix} 2(q_2q_4 - q_1q_3) - a_x \\ 2(q_1q_2 + q_3q_4) - a_y \\ 2\left(\frac{1}{2} - q_2^2 - q_3^2\right) - a_z \end{bmatrix} \quad (15)$$
   $$J = \begin{bmatrix} -2q_3 & 2q_4 & -2q_1 & 2q_2 \\ 2q_2 & 2q_1 & 2q_4 & 2q_3 \\ 0 & -4q_2 & -4q_3 & 0 \end{bmatrix} \quad (16)$$
5. Update the quaternion using the gradient descent algorithm
   $$q_{\nabla,t+1} = -\beta \frac{\nabla f}{||f||} \quad (17)$$
6. Compute orientation increment from gyroscope measurements
   $$\dot{q}_{\omega,t+1} = \frac{1}{2}\hat{q}_t \otimes \left[0, \omega_{t+1}\right]^T \quad (18)$$
7. Fuse measurements to obtain the estimated attitude
   $$q_{t+1} = \hat{q}_t + \Delta t(\dot{q}_{\omega,t+1} + q_{\nabla,t+1}) \quad (19)$$
   endwhile
```

IV. Experimental results

We performed data collection in two scenarios, referred to as 'stationary' and 'in movement,' where the setup is static and corresponding to being in a moving vehicle, respectively [27]. In both setups, we compare the no-filter trigonometric computation (referred to as the "Matrix" method in the plots) and the three presented filtering methods. We evaluate pitch and roll angles, and the computational complexity of the procedure. In our setup, for the Madgwick filter implementation, we have chosen β = 0.05. A low value of β, such as 0.05, results in a smoother output by allowing slower convergence. This is advantageous in our scenario where the measured data

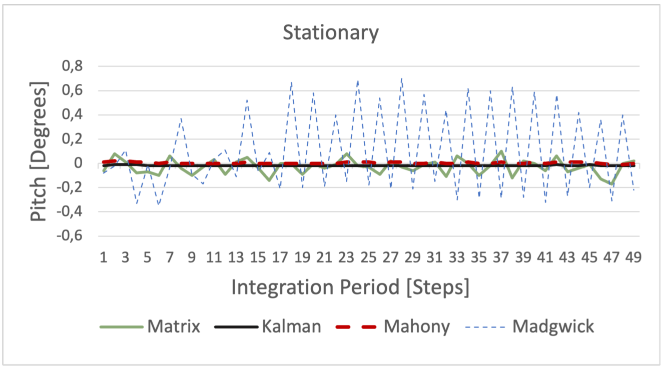

Fig. 1. Pitch angle comparison of the filters for a stationary IMU.

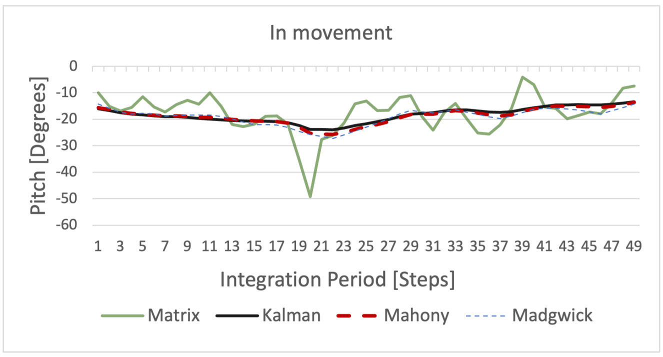

Fig. 3. Pitch angle comparison of the filters with moving IMU.

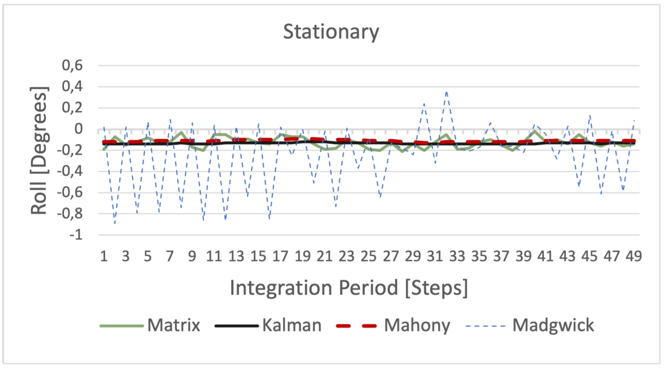

Fig. 2. Roll angle comparison of the filters with stationary IMU.

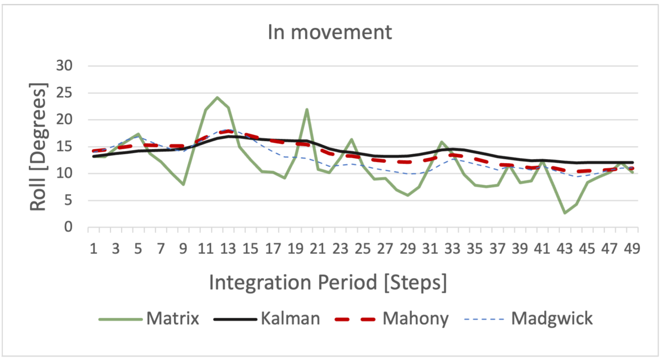

Fig. 4. Roll angle comparison of the filters with moving IMU.

contains noise or rapid fluctuations, and a stable and less jittery output is desired. Mahony filter parameters K_p and K_i are 10 and 0, respectively. We noticed during the experiment that in our scenario, K_i did not significantly affect the quality of the measurement, therefore, we set it to 0 for simplicity. This also avoids issues associated with integration, such as overshoot, instability, or slow convergence, especially in scenarios with negligible steady-state errors and gyroscope bias [10].

A. Stationary results

Figs. 1 and 2 present the results for the four compared methods in the static scenario, where the sensor is in a resting state, showing the pitch and roll angles, respectively. They show the ability of all techniques to converge to a stable and consistent estimate of the system state over a period of time when the inputs to the system are not changing, yet with different overall performance.

Since the device is static without any tilt, we expect the pitch and roll values to be near 0. In Fig. 1, Madgwick's behavior appears to have wider oscillations, possibly caused by the acquisition of more noise. This depends on the coefficient β, which tunes the memory from the previous state of the quaternion, as described in (17). This implies that β is implicitly related to the low-pass pole of the system, and there exists a trade-off. If we want a high bandwidth of the filter, to respond to fast-changing angles, the cost is the presence of more noise, acquired by the filter itself. This value of β was chosen because it allowed the Madgwick filter to respond as fast as the other filters, but as we can see, we do acquire more noise compared to Kalman and Mahony filters.

In Fig. 2, we notice oscillations around -0.1 degrees. Ideally, in a stationary state, the roll value would be 0. This could be due to several factors. One possible reason is that the sensor has a small bias or offset in its measurements, which can cause it to read a non-zero value even at rest. Another possibility is that the sensor is mounted on a surface that is not perfectly level. In most cases, a small deviation such as -0.1 degrees of roll angle when the sensor is not moving is not a significant concern, as it falls within the expected range of error for many IMUs [28].

B. Results in movement

It is also important to evaluate the ability of fusion filters to provide accurate and reliable estimates of the system state as it moves. We intend to utilize this IMU in self-driving vehicles [5], [6], onboard of which the orientation of the device can change. However, unless the car is flipping over, the values of pitch and roll angles are expected to be less than 90 degrees, so we moved the device within that extent.

Figs. 3 and 4 show that the recorded pitch and roll angles in the no-filter scenario (Matrix) have more oscillations. It is interesting to note that the expected behaviour of the traditional method would be with the presence of drifts. In these graphs, we do not have drifts. This is possibly because we have graphed data from only 50 integration steps over 2000 acquired integration steps. By applying no-filtering action, it is expected

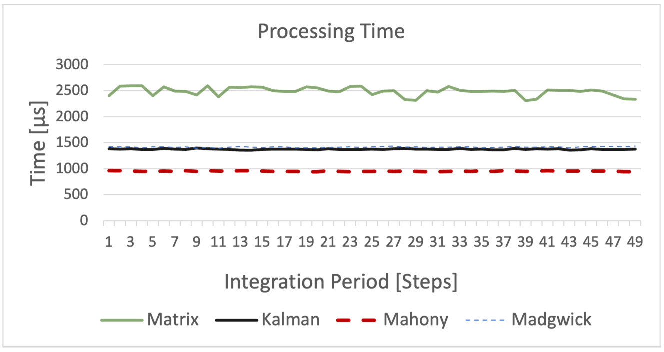

Fig. 5. Comparing the processing time of the filters

TABLE II
AVERAGE PROCESSING TIME

Method	Average processing time
Matrix	2477.4 µs
Kalman	1375.9 µs
Mahony	950.5 µs
Madgwick	1416.8 µs

that in the signal acquisition, the high frequency component of the noise is present. The filters' response is similar. We can see a small difference in Kalman filter's response. Kalman filter exhibits a better behaviour, with less oscillations than Madgwick filter and Mahony filter.

In Fig. 3, the pitch measurement is around the value -15 degrees, which indicates that the sensor is tilted downwards, whereas in Fig. 4, the roll measurement is around the value $+15$ degrees, implying that the sensor is tilted to the right.

C. Complexity

In the moving state, when real-time analysis of the input data is valuable, we computed the processing speed of the filters. Fig. 5 shows that the trigonometry method is the slowest, while the time is significantly reduced with the filter implementations. Kalman and Madgwick exhibit similar behavior. Mahony is the fastest, with the lowest computation time per integration step. Table II reports the average values.

V. CONCLUSIONS

We compared different filters for sensor fusion algorithms and found that the matrix method is proven ineffective for our task without any filtering. Therefore, it is strongly advised to use a filter, the choice of which depends on the specific requirements and constraints of the application, such as the sensor type, noise level, computation power, and desired accuracy and stability. According to our results, the Kalman filter would be the suggested option if accuracy is the primary concern and computational resources and memory are not constraints. If computational efficiency is a critical factor, the Mahony filter would be more suitable due to its computational speed. Overall, the findings of this paper provide valuable insights for selecting an appropriate filtering algorithm for miniature embedded systems.

REFERENCES

[1] L. Ojeda and J. Borenstein, "FLEXnav: fuzzy logic expert rule-based position estimation for mobile robots on rugged terrain," in *Proc. ICRA*, vol. 1, 2002, pp. 317–322.
[2] B. Barshan and H. Durrant-Whyte, "Inertial navigation systems for mobile robots," *IEEE Trans. Robot. Autom.*, vol. 11, no. 3, pp. 328–342, 1995.
[3] G. Cisotto, A. V. Guglielmi, L. Badia, and A. Zanella, "Classification of grasping tasks based on EEG-EMG coherence," in *Proc. IEEE Healthcom*, 2018.
[4] M. Euston, P. W. Coote, R. E. Mahony, J. Kim, and T. Hamel, "A complementary filter for attitude estimation of a fixed-wing UAV," *Proc. IROS*, pp. 340–345, 2008.
[5] U. Michieli and L. Badia, "Game theoretic analysis of road user safety scenarios involving autonomous vehicles," in *Proc. IEEE PIMRC*, 2018, pp. 1377–1381.
[6] C. Raveena, R. Sravya, R. Kumar, and A. Chavan, "Sensor fusion module using IMU and GPS sensors for autonomous car," in *Proc. INOCON*, 2020.
[7] J. Leclerc, "MEMs for aerospace navigation," *IEEE Aerosp. Electron. Syst. Mag.*, vol. 22, no. 10, pp. 31–36, 2007.
[8] A. Makni, H. Fourati, and A. Y. Kibangou, "Energy-aware adaptive attitude estimation under external acceleration for pedestrian navigation," *IEEE/ASME Trans. Mechatronics*, vol. 21, no. 3, pp. 1366–1375, 2016.
[9] A. Zancanaro, G. Cisotto, and L. Badia, "Modeling value of information in remote sensing from correlated sources," *Proc. MedComNet*, 2022.
[10] S. A. Ludwig, "Optimization of control parameter for filter algorithms for attitude and heading reference systems," in *Proc. IEEE CEC*, 2018.
[11] L. Badia, "On the effect of feedback errors in Markov models for SR ARQ packet delays," in *Proc. IEEE Globecom*, 2009.
[12] M. H. Afzal, V. Renaudin, and G. Lachapelle, "Use of Earth's magnetic field for mitigating gyroscope errors regardless of magnetic perturbation," *Sensors*, vol. 11, no. 12, pp. 11 390–11 414, 2011.
[13] S. A. Ludwig and K. D. Burnham, "Comparison of Euler estimate using extended Kalman filter, Madgwick and Mahony on quadcopter flight data," in *Proc. ICUAS*, 2018, pp. 1236–1241.
[14] S. Yean, B. Lee, C. Yeo, C. Vun, and H. L. Oh, "Smartphone orientation estimation algorithm combining Kalman filter with gradient descent," *IEEE J. Biomed. Health Inform.*, vol. 22, no. 5, pp. 1421–1433, 2018.
[15] N. Caporusso, L. Mkrtchyan, and L. Badia, "A multimodal interface device for online board games designed for sight-impaired people," *IEEE Trans. Inf. Technol. Biomed.*, vol. 14, no. 2, pp. 248–254, 2009.
[16] M. Long Hoang and A. Pietrosanto, "Yaw/heading optimization by machine learning model based on mems magnetometer under harsh conditions," *Measurement*, vol. 193, 2022.
[17] J. E. Bortz, "A new mathematical formulation for strapdown inertial navigation," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 7, no. 1, pp. 61–66, 1971.
[18] S. Madgwick, "An efficient orientation filter for inertial and inertial / magnetic sensor arrays," University of Bristol, Tech. Rep., 2010.
[19] V. Torres, J. Valls, and R. Lyons, "Fast- and low-complexity atan2(a,b) approximation [tips and tricks]," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 164–169, 2017.
[20] InvenSense, Inc., "MPU-6000 and MPU-6050 register map and descriptions revision 4.2," 2015, accessed on March 24, 2023.
[21] Arduino, *Arduino Nano 2.3 (ATmega328P) User Manual*, 2012.
[22] Q. Li, R. Li, K. Ji, and W. Dai, "Kalman filter and its application," in *Proc. ICINIS*, 2015, pp. 74–77.
[23] F. Govaers, *Introduction and Implementations of the Kalman Filter*. IntechOpen, May 2019.
[24] S. Y. Song, Y. Pei, and E. T. Hsiao-Wecksler, "Estimating relative angles using two inertial measurement units without magnetometers," *IEEE Sensors J.*, vol. 22, no. 20, pp. 19 688–19 699, 2022.
[25] R. Mahony, T. Hamel, and J.-M. Pflimlin, "Nonlinear complementary filters on the special orthogonal group," *IEEE Trans. Autom. Control*, vol. 53, no. 5, pp. 1203–1218, 2008.
[26] S. Madgwick, A. J. L. Harrison, and R. Vaidyanathan, "Estimation of IMU and MARG orientation using a gradient descent algorithm," *Proc. ICRR*, 2011.
[27] L. Badia and N. Bui, "A group mobility model based on nodes' attraction for next generation wireless networks," in *Proc. ACM Mobility*, 2006.
[28] L. Ricci, "On the orientation error of imu: Investigating static and dynamic accuracy targeting human motion." *PLoS One*, vol. 11, 2016.