Quaternion Based Kalman Filter for Open Loop Attitude Estimation

Moraru Andrei
Department of Automation And Applied Informatics
Faculty of Automation and Computer Science
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
Moraru.gh.andrei@student.utcluj.ro

***Abstract*—Sensors, more specifically Inertial Measurement Units, or IMUs, are used in various applications ranging from mobile phones, the automotive or space industry. The understanding of their data and mathematical models is therefore a pivotal point in the ever growing industries of autonomous driving, IoT, and virtual reality This paper thus proposes an open loop real time angle computation algorithm designed in Matlab®. Its efficacy was tested using an Arduino Nano® coupled with the MPU-6050 as the IMU. The paper also provides a comparison to already existing and widely used methods.**

Keywords—sensor fusion, quaternion, Kalman filter, gyroscope, accelerometer, attitude estimation, MPU-6050, IMU

I. INTRODUCTION

Combining multiple data sources is a way to provide a better understanding of the surrounding world. When the data come from sensors, this process is known as sensor fusion. As the data sources themselves have correlated strengths and weaknesses [1], fusing multiple measurements usually provides a representation that is closer to reality [1]. The most extensively used algorithms in sensor fusion are in general some variation of the Kalman filter. For instance, [2] proposes an unscented Kalman filter for orientation tracking, while [3] elaborates a linear version to fuse magnetic and inertial data for heading estimation. In this paper, a Kalman based sensor fusion algorithm on accelerometer and gyroscope data will be elaborated, and the results will be compared to the already implemented *imufilter* Matlab® function [4], whose return is the orientation of an object with the given inputs of the accelerometer and gyroscope readings.

The paper is structured in five sections. After this short introductory section, section II describes the used model. Section III presents the proposed Kalman filter, with the obtained results in section IV. The work ends with section V as the concluding remarks.

II. ATTITTUDE REPRESENTATION

A. The different types of representation

Attitude, also called orientation, is a way to describe an object's position in space. Its mathematic representation through either Euler angles, direction cosine matrices (DCMs), or quaternions is used to describe a body's reference frame, usually with respect to the world frame. As such, changes in attitude are represented as quantities in the first frame, expressed in the second frame. Out of the three main sets of coordinates enumerated earlier, quaternions have been proven to be the most independent, immune to gimbal lock [4] and an easily interpolated [5] way of representing orientation. Even more, their vector form compactness is able to reduce computation time when compared to the 3×3, often packed with redundancy, matrix alternative [6].

B. Changes in attitude

Changes with time in the orientation (i.e. attitude kinematics) can be conveniently expressed using quaternions. As such, the attitude rate is a function of the current attitude and the measured angular rates [2].

$$\dot{q} = f(q, \vec{\omega}) \tag{1}$$

where q is the atttitude representation in quaternion form, $\vec{\omega}$ is the angular rates vector (also called body) and $\dot{q}$ is the change in attitude (also called attitude rate).

C. Quaternion Algebra

Invented as an extension of complex number system in the three dimensional space [2], quaternions consist of a real component, which denotes the rotation angle, and three imaginary components, representing each axis of rotation. Therefore, their representation uses the Gauss plane [7]:

$$q = q_0 + q_1 i + q_2 j + q_3 k \tag{2}$$

where the rules of the extended complex plane apply as in (3). Due to computation demand, in the remainder of the paper, as well as in the implementation, we will however represent quaternions in vector form [5], as seen in (4).

$$i^2 = j^2 = k^2 = ijk = -1 \tag{3}$$

$$q = [q_0, q_1, q_2, q_3] \tag{4}$$

D. Quaternion norm

The normalization of the quaternion vector is a required operation when transitioning from one rotation to another [6]. When referring to the norm of a quaternion, the Euclidean norm is considered as in (5):

$$\|q\| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \quad (5)$$

When representing a rotation, $\|q\| = 1$ is expected to hold true. In this regard, a quaternion with a norm equal to one is referred to as a unit quaternion [7].

E. Transformations

In order to be able to compare results in a comprehensible manner, transformations from one form of representation to another need to be carried out. As real sensor data is used, first computation will be an Euler sequence of angles, which can be converted to quaternion form [5] using (6) to (9):

$$q_0 = cos\left(\frac{\phi}{2}\right)cos\left(\frac{\theta}{2}\right)cos\left(\frac{\psi}{2}\right) + sin\left(\frac{\phi}{2}\right)sin\left(\frac{\theta}{2}\right)sin\left(\frac{\Psi}{2}\right) \quad (6)$$

$$q_1 = sin\left(\frac{\phi}{2}\right)cos\left(\frac{\theta}{2}\right)cos\left(\frac{\psi}{2}\right) - cos\left(\frac{\phi}{2}\right)sin\left(\frac{\theta}{2}\right)sin\left(\frac{\Psi}{2}\right) \quad (7)$$

$$q_2 = cos\left(\frac{\phi}{2}\right)sin\left(\frac{\theta}{2}\right)cos\left(\frac{\psi}{2}\right) + sin\left(\frac{\phi}{2}\right)cos\left(\frac{\theta}{2}\right)sin\left(\frac{\Psi}{2}\right) \quad (8)$$

$$q_3 = cos\left(\frac{\phi}{2}\right)cos\left(\frac{\theta}{2}\right)sin\left(\frac{\psi}{2}\right) - sin\left(\frac{\phi}{2}\right)sin\left(\frac{\theta}{2}\right)cos\left(\frac{\Psi}{2}\right) \quad (9)$$

where ϕ, θ and Ψ are the X, Y and Z sequence of angles, also called roll, pitch and yaw in the Tait-Bryan convention [6], and $[q_0, q_1, q_2, q_3]$ are the elements of the quaternion vector, as shown in (4).

To recover the sequence of angles from a quaternion vector, an intermediary DCM shall be used [8] as in (10):

$$DCM = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \quad (10)$$

From which each angle can be extracted using trigonometric functions [6] as in (11), (12) and (13):

$$\theta = -\operatorname{asin}(DCM_{31}) \quad (11)$$

$$\phi = \operatorname{atan2}(DCM_{32}, DCM_{33}) \quad (12)$$

$$\psi = \operatorname{atan2}(DCM_{21}, DCM_{11}) \quad (13)$$

where $asin$ is the inverse *sine*, DCM_{31} is the element of the DCM matrix located in the third row and first column and $atan2$ is the four quadrant inverse tangent.

Lastly, to represent the change in rotation, the expanded formula first stated in (1) is carried out as in (14), where the Q is the body frame matrix [2] shown in (15).

$$\dot{q} = \frac{1}{2}Q\vec{\omega} \quad (14)$$

$$Q = \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & q_3 & -q_2 \\ -q_3 & q_0 & q_1 \\ q_2 & -q_1 & q_0 \end{bmatrix} \quad (15)$$

III. The Kalman Filter

Being not only the best linear estimator, as a solver of the least squares method, the Kalman filter is also a widely used state of the art algorithm in sensor fusion. When working with multiple sensors, as the three axis accelerometer and gyroscope present in the MPU-6050, the Kalman gain K represents the proportion of trust given to the different measurements [9], and is computed at each measurement update, given the tuning parameters Q and R, which represent the process and measurement noise [10].

A. Process Model

The process model is a mathematical representation of a dynamic system meant to encapsulate the kinematics of the state space representation, which makes it is a focal part of estimation algorithms. The orientation is constructed as a quaternion vector, whose four components represent the system states, which are then propagated in time, using Euler integration [10] (also known as *dead reckoning* in the gyroscope context), along with a a matrix consisting of the variances of each of the errors present in the states estimation (also called covariance matrix P [2]) . The quaternion vector can be recovered from the angular rates body measured by the gyroscope as shown in (14). This is known as the prediction phase of the filter [9].

B. Measurement Model

The measurements coming into the filter are the sequence of tilt angles coming from the accelerometer's linear and gravitational accelerations measurements. As the output of a sensor is not linear, the angles cannot be computed through simple integration, and rather need to be transformed using trigonometric functions [10], as shown in (16) and (17). The measurements are then compared to the states computed in the prediction phase and the error (also called innovation) is calculated as the difference between the estimated state and the measured state. The state is then updated based on the Kalman gain. This is known as the update step of the filter [2].

$$\phi_{meas} = \operatorname{atan2}\left(\frac{accel\_y}{accel\_z}\right) \quad (16)$$

$$\theta_{meas} = \operatorname{atan2}\left(\frac{-accel\_x}{\sqrt{accel\_y^2 + accel\_z^2}}\right) \quad (17)$$

C. Algorithm

Although consistent usage of trigonometric functions for angle computation is present, as the angles themselves are introduced directly as measurements, there is no need for linearization in this regard, and so the linear Kalman filter was chosen over the extended Kalman filter, as its convergence is guaranteed [9]. The algorithm is carried out as it is depicted in Fig.1. When it comes to the tuning parameters of the filter, the process model noise matrix Q is considered as a diagonal matrix of the squared standard deviations of each of the quaternion vector components, while the measurement noise matrix R is constructed as a matrix of the same quaternion vector element variance, yet this time as a measurement uncertainty coming from the accelerometer measurement which is transformed in quaternion form. Equations (18) and (19) provide a representation of their choice.

$$Q = \begin{bmatrix} \sigma_{q_0}^2 & 0 & 0 & 0 \\ 0 & \sigma_{q_1}^2 & 0 & 0 \\ 0 & 0 & \sigma_{q_2}^2 & 0 \\ 0 & 0 & 0 & \sigma_{q_3}^2 \end{bmatrix} \quad (18)$$

$$R = \begin{bmatrix} \sigma_{q_{meas0}}^2 & 0 & 0 & 0 \\ 0 & \sigma_{q_{meas1}}^2 & 0 & 0 \\ 0 & 0 & \sigma_{q_{meas2}}^2 & 0 \\ 0 & 0 & 0 & \sigma_{q_{meas3}}^2 \end{bmatrix} \quad (19)$$

As the linear filter is bound to converge regardless of initialization, the covariance matrix *P* can be initialized as a diagonal matrix whose trace are numbers greater than the expected variance of each state. An example of choice is provided in (20), where p is a positive integer chosen empirically. If the first state is known, *p* should be set to zero.

$$P = \begin{bmatrix} p\sigma_{q_0}^2 & 0 & 0 & 0 \\ 0 & p\sigma_{q_1}^2 & 0 & 0 \\ 0 & 0 & p\sigma_{q_2}^2 & 0 \\ 0 & 0 & 0 & p\sigma_{q_3}^2 \end{bmatrix} \quad (20)$$

Figure 1. The proposed algorithm

Initialize filter on (16) and (17)
Initialize Q, R and P on (18), (19) and (20)
Initialize $q_0 = [0,0,0,0]$
Initialize transition and observation matrices $F = H = I_4$
Set state $x_0 = q_0$
do while arduino is connected:
read [gyro_x, gyro_y, gyro_z] from gyroscope
$\vec{\omega}$ = [gyro_x, gyro_y, gyro_z]
$\vec{\omega} \rightarrow \dot{q}$ using (14), (15)
Begin prediction step:
$x_{k+1} = x_k + \dot{q}\Delta t$ (euler integration)
$x_{k+1} = \frac{x_{k+1}}{\|x_{k+1}\|}$ using (5) (quaternion normalization)
$P_{k+1} = FP_kF^T + Q$ (covariance propagation)
Begin update step:
read [accel_x, accel_y, accel_z] from sensor
compute angles using (16) and (17)
$z = measurement = [\phi_{meas}\ \theta_{meas}]$
map: euler → quaternion for z using (6) to (9)
$\hat{z} = Hx_k$ (compute estimation)
$innov = z - \hat{z}$ (compute error)
$S_k = HP_kH^T + R$ (innovation covariance)
$K_k = P_kH^TS_k^{-1}$ (Kalman gain)
$\hat{x}_k^+ = \hat{x}_k^- + K_k \cdot innov$ update state)
$P_k^+ = P_k^- - K_kHP_k^-$ (update covariance)
recover euler sequence using (10) to (13)
Optional:
IIR filter with Butterworth

Considering that the states of the vector are independent of each other, the transition matrix as well as the measurement (also called observation) matrix are chosen as the identity matrix I_4.

Due to only the roll and pitch angles being measured from the accelerometer, because the quaternion vector holds informantion about the yaw angle as the rotation around the Z axis, the measurement vector has to be augmented with the yaw state for the matrix multiplications to be valid, yet the value itself is not considered, nor verified for convergence.

The reason why the yaw angle is not computed is because it cannot be calculated reliably using only an accelerometer and a gyroscope. The explanation for this is straightforward: as the sensor is rotated around the Z axis, the acceleration of the axis stays the same. While the gyroscope integration can still be done, the whole system now lacks a referrence frame and the algorithm is no longer performing sensor fusion. State of the art methods for calculating the yaw angle usually include a magnetometer as a third sensor [3] and are referred to as Attitude and Heading Referrence Systems (AHRS)

The convergence of the filter to the true state is guaranteed by the minimization of the trace of the covariance matrix *P* [9].

At the expense of convergence speed, an extra IIR first or second order butterworth filter may be considered to further reduce the noise when the matrix *R* was chosen and tested. For the implementation of the filter, the built in *butter* Matlab® function [11] is used, with the measurement biases as the initial conditions.

Table 1: Performance Analysis

Method	Mean Squared Error		
	Phase	***Roll Angle***	***Pitch Angle***
Gyroscope Dead Reckoning	Prediction	0.0341	0.0083
Accelerometer Measurement	Update	0.0374	0.0142
Proposed Kalman Filter	Fusion	0.0306	0.0075
MATLAB's *imufilter* (processed to remove delay)	Fusion	0.0056	0.00067
Butterworth + Proposed KF	IIR Filtering	0.0256	0.0020

IV. RESULTS

The overall performance of each implementation and phase was measured using the Mean Squared Error (MSE) as the difference between the measured signal and the ground truth, when the sensor is not moving, so that the ground truth would be zero (0). During the testing phase, it was found that the MPU-6050 has run-to-run biases for the angle measurements, therefore calibration has to be carried out before each online estimation. The biases of the measurements will be subtracted from the actual measurement using the *mean* Matlab® function. For this experiment, bias drift [10] due to increase in temperature was not considered, nor modelled, because its effect was found to be negligible. As seen in Fig.2 and Fig.3, the convergence of the *imufilter* from Matlab® is much slower, and the delay in time is caused by the gyroscope measurement integrating the error in time as well as the angular velocity. While the proposed filter is more affected by noise (as it can be

seen in Table 1), its response to changes in attitude is faster and proves stable for both angles, as shown in Fig.4 and Fig 5. If convergence speed is preferred in favor of less overall noise, one might decide to use the accelerometer measurement by itself. If the gyroscope is the only available sensor, with the correct process model, the angles can be obtained reliably, although delay from integration will be present. If both sensors are present, data fusion through either of the methods is preferred.

V. Conclusions

The present paper proposes a sensor fusion method based on a linear Kalman filter with quaternion transformations as the process model. The algorithm successfully converges to a value closer to the truth than the raw measurements and is able to mitigate the integration error and time delay effects of the *imufilter* implementation.

Acknowledgment

Very great appreciation goes towards my professors, Dr. eng. Eva H. Dulf and Dr. eng. Zsófia Lendek for their patient guidance, useful critiques and encouragement.

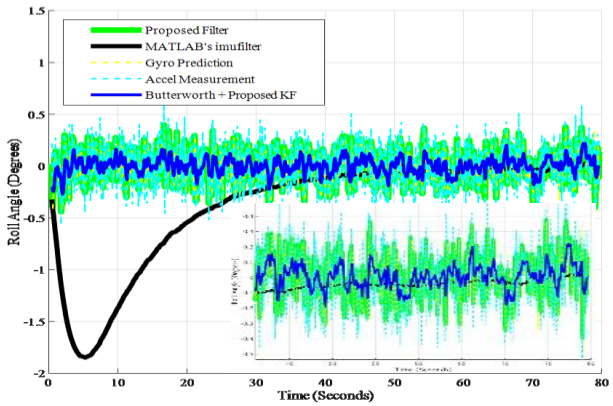

Figure 2: Roll angle measured in stationary position

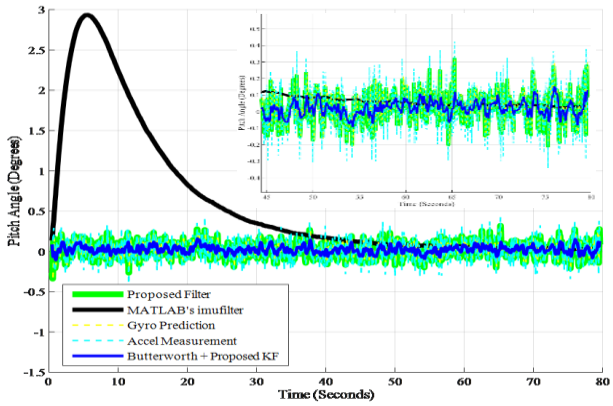

Figure 3: Pitch angle measured in stationary position

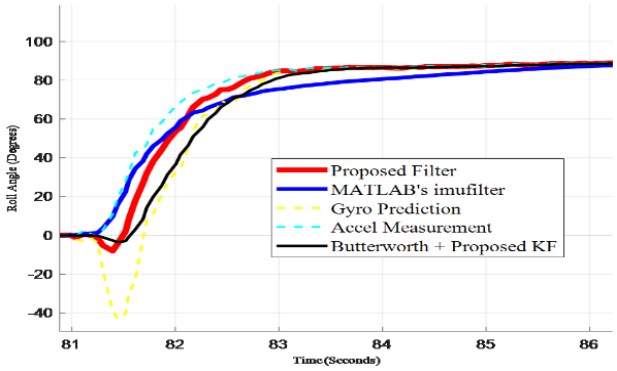

Figure 4: Roll angle response to orthogonal change in position

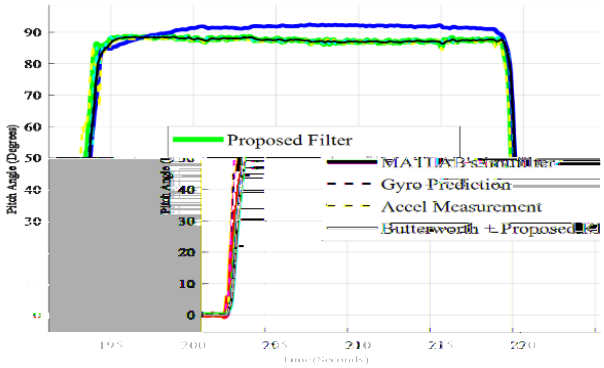

Figure 5: Pitch angle responses to orthogonal changes in position

References

[1] Abyarjoo, Fatemeh, et al. "Implementing a sensor fusion algorithm for 3D orientation detection with inertial/magnetic sensors." *Innovations and advances in computing, informatics, systems sciences, networking and engineering*. Springer, Cham, (2015):305-310.

[2] Kraft, Edgar. "A quaternion-based unscented Kalman filter for orientation tracking." *Proceedings of the sixth international conference of information fusion*. Vol. 1. No. 1. IEEE Cairns, 2003.

[3] Neto, Pedro, Nuno Mendes, and A. Paulo Moreira. "Kalman filter-based yaw angle estimation by fusing inertial and magnetic sensing: A case study using low cost sensors." *Sensor Review* (2015).

[4] ***Matlab, https://de.mathworks.com/help/fusion/ref/imufilter-system-object.html - Accessed April 2022

[5] Perumal, Logah. "Quaternion and its application in rotation using sets of regions." *International Journal of Engineering and Technology Innovation* 1.1 (2011): 35.

[6] Diebel, James. "Representing attitude: Euler angles, unit quaternions, and rotation vectors." *Matrix* 58.15-16 (2006): 1-35.

[7] Jia, Yan-Bin. "Quaternions and rotations." *Com S* 477.577 (2008): 15.

[8] Sarabandi, Soheil, and Federico Thomas. "Accurate computation of quaternions from rotation matrices." *International Symposium on Advances in Robot Kinematics*. Springer, Cham, 2018.

[9] Choukroun, Daniel, Itzhack Y. Bar-Itzhack, and Yaakov Oshman. "Novel quaternion Kalman filter." *IEEE Transactions on Aerospace and Electronic Systems* 42.1 (2006): 174-190.

[10] Wu, Zheming, et al. "Attitude and gyro bias estimation by the rotation of an inertial measurement unit." *Measurement Science and Technology* 26.12 (2015): 125102.

[11] ***Matlab, https://www.mathworks.com/help/signal/ref/butter.html - Accesed April 2022

.