# Quaternion-Based Orientation Estimation Using an IMU

Report Author
Joar Edlund
joared@kth.se

Group Partner
Kamil Kaminski
kamilk@kth.se

***Abstract*—In this work, quaternion-based orientation estimation is investigated using gyro and accelerometer measurements from an Inertial Measurement Unit. The measurements are filtered in an Error-state Kalman Filter, where the gyro measurements are used in the prediction step, and the accelerometer measurements are used in the update step. Sensor error sources such as bias and misalignment are ignored, and the algorithm performance is investigated with and without the update step. The algorithm is evaluated on a publicly available dataset, as well as a custom-made dataset with ground truth orientation. The noise related covariances are tuned manually and the filter algorithm shows satisfactory results. Issues regarding the custom dataset are discussed, and future improvements are suggested.**

## I. Introduction

Inertial measurement units (IMUs) are referred to devices that use a combination of a gyro, accelerometer, and in some cases, a magnetometer. IMUs are typically used to determine the orientation (attitude) of a rotating body, and are commonly utilized for navigation within the areas of robotics and aviation.

IMUs provides high-frequency and high precision data, and orientation estimation is usually performed by fusing gyro and accelerometer data. Even though the angular velocity measurements from the gyro provides accurate short-term orientation estimates, achieved by integration, the estimated orientation will experience a continuous drift, which makes the estimation unreliable long-term. By fusing the gyro measurements with the accelerometer measurements, the drift can be compensated (constrained to 2 degrees of freedom). Furthermore, the appropriate choice of orientation representation and fusion algorithm depends on the application. While orientation representation using Euler angles are intuitive, issues such as gimbal lock has to be taken into account. Regarding the choice of fusion algorithm, complex dynamic modelling plays an important role, and can be circumvented using indirect approaches.

### A. Contribution

This work investigates the problem of estimating the orientation of a rotating body using the Indirect/Error-state Kalman Filter (ESKF). The orientation is represented by a quaternion, and the estimation is performed by fusing the gyro and accelerometer measurements from an IMU. The gyro measurements are used in the prediction step, while the accelerometer measurements are used in the update step to correct the predicted estimates. Sensor errors such as bias and misalignment are ignored. We perform an ablation study to evaluate the performance of the ESKF algorithm, with and without the update step. The evaluation is performed on a publicly available dataset, as well as a custom-made dataset, created using a motion capture system.

## II. Related work

The use of quaternion-based Kalman filtering for orientation estimation is a well studied subject and has been subjected to many improvements over the years. One of the earlier works, investigates the use of IMU sensor data to form the Kalman filter state equations, where the orientation is represented by a quaternion [2]. Euler angles suffers from singularity issues (gimbal lock), which was circumvented with the quaternion representation. The authors used a line-of-sight sensor for the update step and investigated methods for handling the singular covariance matrix of the quaternion representation.

The advantage of the ESKF, versus the Extended Kalman Filter (EKF) is outlined in [3], where the main benefits being lesser need of precise covariance tuning and complex dynamic modelling.

The ESKF remains among the current state-of-the-art Kalman filter algorithms for orientation estimation. An exhaustive revision of the relevant quaternion algebra, and its application to the ESKF for orientation estimation is given in [4]. The error-state is parameterized using the rotation vector and thereby represented minimally (same number of parameters as degrees of freedom). Furthermore, the error-state kinematics are approximately linear, meaning it is more suitable for Kalman filtering. This work is highly inspired by [4], which contains the major background for proposed the method. Furthermore, the *sun sensor*, described in [7], will be used in the update step to incorporate the accelerometer measurements.

## III. Background

In this section, we give the basic background for representation, as well as manipulation of orientations represented by quaternions. We also describe the relation between the rotation vector, rotation matrix and quaternion. The theory presented in this section is found in [4].

### A. Quaternion

First, we define the quaternion convention we use in this work. Then we describe the quaternion product, and how unit quaternions represents orientations, which can be composed using the quaternion product.

*1) Definition:* A quaternion $\mathbf{q}$ is defined by 4 parameters according to

$$\mathbf{q} = \begin{bmatrix} q_x \\ q_y \\ q_z \\ q_w \end{bmatrix} = \begin{bmatrix} \mathbf{q}_v \\ q_w \end{bmatrix}$$

where $q_w$ is the *scalar* part and $\mathbf{q}_v$ is the *vector* part. The quaternion represents a complex number according to

$$\mathbf{q} = q_x\mathbf{i} + q_y\mathbf{j} + q_z\mathbf{k} + q_w$$
$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$$

The multiplicative rule shown in the above equation, is the *Hamilton* convention, and is what we will use in this work.

*2) Product:* The product of two quaternions, $\mathbf{q}$ and $\mathbf{p}$, is given by

$$\mathbf{q} \otimes \mathbf{p} = [\mathbf{q}]_L \mathbf{p} = \begin{bmatrix} q_w & -q_z & q_y & q_x \\ q_z & q_w & -q_x & q_y \\ -q_y & q_x & q_w & q_z \\ -q_x & -q_y & -q_z & q_w \end{bmatrix} \mathbf{p} \tag{1}$$

where $\otimes$ denotes the quaternion product. As we see from the above equation, we have conveniently expressed the product as a matrix multiplication using $[\mathbf{q}]_L$.

*3) As orientation:* The orientation of a coordinate frame $B$, with respect to a coordinate frame $A$, is represented by the unit quaternion $\mathbf{q} = {}^A_B\mathbf{q}$ with

$$||\,{}^A_B\mathbf{q}|| = \sqrt{q_x^2 + q_y^2 + q_z^2 + q_w^2} = 1$$

Compositions of orientations, represented by unit quaternions, is performed using the quaternion product

$$\begin{aligned} {}^A_C\mathbf{q} &= {}^A_B\mathbf{q} \otimes {}^B_C\mathbf{q} \\ &= [{}^A_B\mathbf{q}]_L \, {}^B_C\mathbf{q} \end{aligned}$$

The inverse of a unit quaternion $\mathbf{q} = {}^A_B\mathbf{q}$ is calculated as its conjugate, i.e.

$${}^A_B\mathbf{q}^{-1} = \mathbf{q}^* = \begin{bmatrix} -\mathbf{q}_v \\ q_w \end{bmatrix} = {}^B_A\mathbf{q} \tag{2}$$

### B. Rotation vector, Rotation matrix and Quaternion

Any orientation or rotation can be described by an angle $\theta$ around an axis of rotation $\hat{\boldsymbol{\theta}}$. This form is called *axis-angle* representation. More concisely, we can simply represent the orientation using the vector $\boldsymbol{\theta}$, where

$$\boldsymbol{\theta} = ||\boldsymbol{\theta}||\hat{\boldsymbol{\theta}} = \theta\hat{\boldsymbol{\theta}}$$

which is known as the *rotation vector*. Given the rotation vector $\boldsymbol{\theta} = \theta\hat{\boldsymbol{\theta}}$, we obtain the corresponding rotation matrix $\mathbf{R}$, and unit quaternion $\mathbf{q}$ as

$$\mathbf{R}\{\boldsymbol{\theta}\} = \exp([\boldsymbol{\theta}]_\times) = \mathbf{I} + \sin(\theta)[\hat{\boldsymbol{\theta}}]_\times + (1 - \cos(\theta))[\hat{\boldsymbol{\theta}}]_\times^2 \tag{3}$$

$$\mathbf{q}\{\boldsymbol{\theta}\} = \exp(\boldsymbol{\theta}/2) = \begin{bmatrix} \sin(\theta/2)\hat{\boldsymbol{\theta}} \\ \cos(\theta/2) \end{bmatrix} \tag{4}$$

where $[\hat{\boldsymbol{\theta}}]_\times$ denotes the skew-symmetric matrix. For a vector $\boldsymbol{\omega}$, the skew-symmetric matrix is given by

$$[\boldsymbol{\omega}]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{5}$$

Equation 3 is commonly known as *Rodrigues' formula* and Equation 4 is an extension to the *Euler formula*, applied to the quaternion exponential. The reader is encouraged to visit [5] for in-depth understanding about the *exponential/logarithmic map* and *Lie-groups*.

Given a quaternion $\mathbf{q}$, we obtain the corresponding rotation matrix $\mathbf{R}$ as

$$\mathbf{R}\{\mathbf{q}\} = (q_w - \mathbf{q}_v^T\mathbf{q}_v)\mathbf{I} + 2\mathbf{q}_v\mathbf{q}_v^T + 2q_w[\mathbf{q}_v]_\times \tag{6}$$

## IV. Method

In this section, we describe the proposed ESKF, which is highly inspired by [4]. First, we describe the gyro and accelerometer sensor models. Then, we describe the ESKF system kinematics, and finally, we describe the ESKF algorithm in detail. The implementation of the ESKF algorithm was made in Python3 and the source code can be found on GitHub[1].

### A. Sensor models

The gyro and accelerometer provides noisy readings of the true angular velocity, $\boldsymbol{\omega}_t$, and the acceleration $\mathbf{a}_t$, respectively. The measurements are expressed in the body frame, and we will exclude modelling of bias terms. The measurements are assumed to be corrupted by Gaussian white noise according to

$$\boldsymbol{\omega}_m = \boldsymbol{\omega}_t + \boldsymbol{\omega}_n \qquad \boldsymbol{\omega}_n \sim \mathcal{N}(\mathbf{0},\ \sigma_{\boldsymbol{\omega}}^2\mathbf{I}) \tag{7}$$

$$\mathbf{a}_m = \mathbf{R}_t^T(\mathbf{a}_t - \mathbf{g}) + \mathbf{a}_n \qquad \mathbf{a}_n \sim \mathcal{N}(\mathbf{0},\ \sigma_{\mathbf{a}}^2\mathbf{I}) \tag{8}$$
$$\mathbf{a}_t \approx 0$$

where $\mathbf{R}_t \triangleq \mathbf{R}\{\mathbf{q}_t\}$ is the rotation matrix given by Equation 6, and the gravity vector $\mathbf{g}$ is assumed to be known. We will assume that $\mathbf{a}_t \approx 0$, meaning, the acceleration due to linear displacement of the sensor body is considered as disturbance and handled by appropriate tuning of the noise parameter $\sigma_{\mathbf{a}}$.

### B. System kinematics

In this section, we describe the kinematics of the system. First, we introduce the different state variables used in the ESKF, and then we describe the continous- and discrete-time system equations.

[1] https://github.com/joared/quaternion_eskf

*1) True-, nominal- and error-state:* In the ESKF formulation, we consider true-, nominal- and error-state variables. The true-state $\mathbf{x}_t$, is represented by a suitable composition of the nominal-state $\mathbf{x}$, and the error-state $\delta\mathbf{x}$, i.e.

$$\mathbf{x}_t = \mathbf{x} \oplus \delta\mathbf{x} \tag{9}$$

where $\oplus$ denotes the generic composition of the nominal- and error-state. In our case, the composition is performed using the quaternion product

$$\overset{I}{B}\mathbf{q}_t = \overset{I}{\hat{B}}\mathbf{q}_t \otimes \overset{\hat{B}}{B}\delta\mathbf{q} \tag{10}$$

where $I$ is the inertial frame, $B$ is the body frame, and $\hat{B}$ is the estimated body frame. From now on, we will exclude the prescripts for clarity. The error-state represent a (local) perturbation of the nominal state that "brings" us to the true state. The idea is to formulate a linear expression for the error-state and estimate the errors using the ESKF. The nominal state is integrated (non-linearly), without taking noise terms and other modelling errors into account, while the error-state predicts the error and its covariance.

*2) Continuous time kinematics:* Isolating the true angular velocity in Equation 7, the true system kinematics is formulated as a quaternion multiplication according to

$$\dot{\mathbf{q}}_t = \frac{1}{2}\mathbf{q}_t \otimes \boldsymbol{\omega}_t = \frac{1}{2}\mathbf{q}_t \otimes (\boldsymbol{\omega}_m - \boldsymbol{\omega}_n) \tag{11}$$

Note that $\boldsymbol{\omega}_t$ is interpreted as a pure quaternion i.e. $\boldsymbol{\omega}_t = \begin{bmatrix}\boldsymbol{\omega}_t \\ 0\end{bmatrix} = \begin{bmatrix}\boldsymbol{\omega}_m - \boldsymbol{\omega}_n \\ 0\end{bmatrix}$. From the true-state kinematics in Equation 11, we derive the continuous-time kinematics for the nominal- and error-state as follows:

| **Nominal-state** | **Error-state** | |
|---|---|---|
| $\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \otimes \boldsymbol{\omega}_m$ | $\dot{\delta\boldsymbol{\theta}} = -[\boldsymbol{\omega}_m]_\times \delta\boldsymbol{\theta} - \boldsymbol{\omega}_n$ | (12) |

The formulation for the error-state above, is minimally represented by the error vector $\delta\boldsymbol{\theta}$ (rotation vector consisting of three parameters). The derivation is formed from the assumption that the error propagation is small, and we use the *small angle approximation* of a quaternion

$$\delta\mathbf{q} = \begin{bmatrix}\sin(\theta/2)\hat{\boldsymbol{\theta}} \\ \cos(\theta/2)\end{bmatrix} \approx \begin{bmatrix}\frac{1}{2}\delta\boldsymbol{\theta} \\ 1\end{bmatrix} \tag{13}$$

With small errors $\delta\boldsymbol{\theta}$, we do not risk ambiguities of the rotation vector representation which occurs at $\theta \pm \pi$.

*3) Discrete time kinematics:* The discrete-time kinematics are derived by integration of the continuous nominal- and error-state kinematics:

| **Nominal-state** | **Error-state** | |
|---|---|---|
| $\mathbf{q} = \mathbf{q} \otimes \mathbf{q}\{\boldsymbol{\omega}_m \Delta t\}$ | $\delta\boldsymbol{\theta} = \mathbf{R}^T\{\boldsymbol{\omega}_m \Delta t\}\delta\boldsymbol{\theta} + \boldsymbol{\theta}_\mathbf{i}$ | (14) |

where $\Delta t$ is the sampling time, and $\boldsymbol{\theta}_\mathbf{i}$ is the random impulse with covariance $\mathbf{Q} = \sigma^2_{\boldsymbol{\theta}}\mathbf{I}$, obtained by integration of $\boldsymbol{\omega}_n$. $\sigma_{\boldsymbol{\theta}}$ will in fact be dependent on the sampling time, however, will assume that the sampling time is constant and hand tune the noise parameter $\sigma_{\boldsymbol{\theta}}$. The nominal state propagation corresponds to a zeroth-order integration, meaning, we assume that the angular velocity is constant over $\Delta t$. The integration of the error-state results in a closed form solution, following the small angle approximation in Equation 13.

### C. Error-state Kalman Filter algorithm

In this section, we describe the ESKF algorithm in detail. First, we describe the transition model and prediction step. Then, we describe the measurement model for the accelerometer measurement and the update step, which compared to the traditional Kalman Filter, includes a reset operation.

*1) Transition model:* The transition model is obtained from the discrete kinematics of the error-state in Equation 14 as

$$\boldsymbol{\Phi}(t + \Delta t, t) = \mathbf{R}^T\{\boldsymbol{\omega}_m \Delta t\} \qquad \mathbf{Q} = \sigma^2_{\boldsymbol{\theta}}\mathbf{I} \tag{15}$$

In this case, the transition matrix is trivial since it is linear in the error-state $\delta\boldsymbol{\theta}$. However, in the general case, the transition matrix can be obtained by the Jacobian of the error-state kinematics to achieve an Extended error-state Kalman Filter formulation.

*2) Prediction step:* The nominal- and error-state mean and covariance is propagated according to:

| **Nominal-state** | **Error-state** | |
|---|---|---|
| $\hat{\mathbf{q}} = \hat{\mathbf{q}} \otimes \mathbf{q}\{\boldsymbol{\omega}_m \Delta t\}$ | $\hat{\delta\boldsymbol{\theta}} = \boldsymbol{\Phi} \cdot \hat{\delta\boldsymbol{\theta}}$ | (16) |
| | $\mathbf{P} = \boldsymbol{\Phi}\mathbf{P}\boldsymbol{\Phi}^T + \mathbf{Q}$ | (17) |

The equation of the error-state in Equation 16 is merely describing the propagation of the error-state mean, and not implemented in practice since it will always return zero. Equation 17 follows the standard Kalman Filter covariance propagation.

*3) Measurement model:* We express the measurements $\mathbf{y} = \mathbf{a}_m$ from the accelerometer in Equation 8 as

$$\mathbf{y} = h(\mathbf{q}_t) + \mathbf{a}_n \qquad \mathbf{a}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{V}) \tag{18}$$

with $\mathbf{V} = \sigma^2_{\mathbf{a}}\mathbf{I}$. The function $h(\mathbf{q}_t)$, describes what the measurement true measurement would be, given the true orientation $\mathbf{q}_t$ and is given by

$$h(\mathbf{q}_t) = \mathbf{q}_t^* \otimes (-\mathbf{g}) \otimes \mathbf{q}_t = -\mathbf{R}_t^T\mathbf{g} \tag{19}$$

The measurement matrix is obtained by the Jacobian of $h$ w.r.t. the error-state, evaluated at the best estimate $\hat{\mathbf{x}}_t = \mathbf{x} \oplus \hat{\delta\mathbf{x}}$. Since we at this point have only made predictions, $\hat{\delta\mathbf{x}} = \delta\boldsymbol{\theta} = \mathbf{0}$, and therefore, $\hat{\mathbf{x}}_t = \mathbf{q}$. We get

$$\mathbf{H} = \left.\frac{\partial h}{\partial \delta\mathbf{x}}\right|_{\mathbf{x}} = \left.\frac{\partial h}{\partial \delta\boldsymbol{\theta}}\right|_{\mathbf{q}} = [\mathbf{R}^T\mathbf{g}]_\times \tag{20}$$

A derivation of the above expression is found in [7], and the measurement model describes the integration of the accelerometer as a *sun sensor*.

*4) Update step:* The update step of the error-state is performed by the standard EKF update equations

$$\begin{aligned} \mathbf{K} &= \mathbf{P}\mathbf{H}^T(\mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{V})^{-1} \\ \hat{\delta\boldsymbol{\theta}} &= \mathbf{K}(\mathbf{y} - h(\hat{\mathbf{q}}_t)) \\ \mathbf{P} &= (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P} \end{aligned} \tag{21}$$

The nominal state is then updated using the new error-mean $\hat{\delta\boldsymbol{\theta}}$ according to

$$\mathbf{q} = \mathbf{q} \otimes \mathbf{q}\{\hat{\delta\boldsymbol{\theta}}\} \tag{22}$$

*5) ESKF reset:* Since the error-mean is non-zero after the update step, we perform a reset operation. The reset operation is performed by setting the error-mean back to zero, and adjusting the covariance $\mathbf{P}$ to account for this reset. The reset function is given by

$$\delta\mathbf{q} = g(\delta\mathbf{q}) = \hat{\delta\mathbf{q}}^* \otimes \delta\mathbf{q} \tag{23}$$

The Jacobian is calculated as $\mathbf{G} = \frac{\partial g}{\partial \delta\boldsymbol{\theta}}|_{\hat{\delta\boldsymbol{\theta}}} = \mathbf{I} - [\frac{1}{2}\hat{\delta\boldsymbol{\theta}}]$ and the reset is performed as follows:

$$\begin{aligned} \hat{\delta\boldsymbol{\theta}} &= \mathbf{0} \\ \mathbf{P} &= \mathbf{G}\mathbf{P}\mathbf{G}^T \end{aligned} \tag{24}$$

### D. Evaluation

The ESKF algorithm will be evaluated on two datasets that are introduced in the following section. We will perform an ablation study, investigating the performance of the ESKF algorithm with and without the update step.

*1) Error-metric:* The performance will be quantified by considering the absolute orientation error-angle $|\Delta\theta|$ between the true orientation $\mathbf{q}_t$ and the estimated orientation $\hat{\mathbf{q}}$. $\Delta\theta$ can be obtained by considering the error quaternion

$$\Delta\mathbf{q} = \mathbf{q}_t \otimes \hat{\mathbf{q}}^* = \begin{bmatrix} ||\Delta\mathbf{q}_v||\hat{\Delta\mathbf{q}_v} \\ \Delta q_w \end{bmatrix} = \begin{bmatrix} \sin(\Delta\theta/2)\hat{\Delta\boldsymbol{\theta}} \\ \cos(\Delta\theta/2) \end{bmatrix}$$

From $\Delta\mathbf{q}$ we can obtain $|\Delta\theta|$ as follows:

$$|\Delta\theta| = \left| 2 \cdot \tan^{-1}\left( \frac{||\Delta\mathbf{q}_v||}{\Delta q_w} \right) \right| \tag{25}$$

This formulation is equivalent to the *Geodesic on the Unit Sphere* error-metric in [1], which henceforth will be referred to as **GUS**.

## V. Experiments

To evaluate the performance of the proposed ESKF algorithm, with and without accelerometer updates, we tested the algorithm on two datasets: the RepoIMU dataset, publicly available online, and a custom-made dataset. We present the results using the **GUS** error-metric (explained in the previous section) and with illustrative plots of the estimated and true quaternion components.

### A. RepoIMU

The RepoIMU dataset [6] contains 9DoF IMU sensor data consisting of gyro, accelerometer and magnetometer (not used in our work) measurements. The dataset includes ground truth orientations in quaternion format which has been recorded with a motion capture system. At the time of writing this report, the website from which the dataset was downloaded, was experiencing issues[2].

### B. Custom dataset

A custom-made dataset was created with a motion capture system using a 6DoF IMU (Invense ICM-20948) attached on a stick (see Figure 1), together with the target (OptiTrack Flex 3). Gyro and accelerometer data was recorded during motion around its 3 axes of rotation, and a video illustrating the recording process can be found here[3]. The sensor data (gyro and accelerometer) and ground truth (orientation as quaternion) was recorded separately, and time-synchronization was tuned manually. Raw sensor data scaling was required to obtain the actual angular rate and acceleration data from the gyro and accelerometer, respectively.

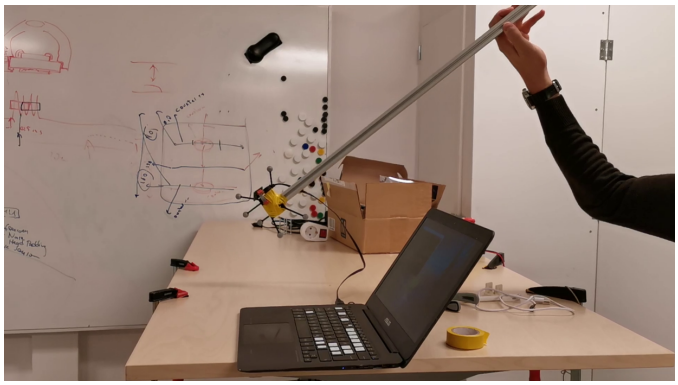


Fig. 1: Custom dataset motion capture setup

### C. Results and discussion

In all experiments, we used the gravity vector and tuning parameters below:

$$\mathbf{g} = \begin{bmatrix} 0 \\ 0 \\ -9.8255 \end{bmatrix} \quad \sigma_{\mathbf{a}} = 0.29 \times ||\mathbf{g}|| \approx 2.64 \quad \sigma_{\boldsymbol{\theta}} = 5.8 \times 10^{-4}$$

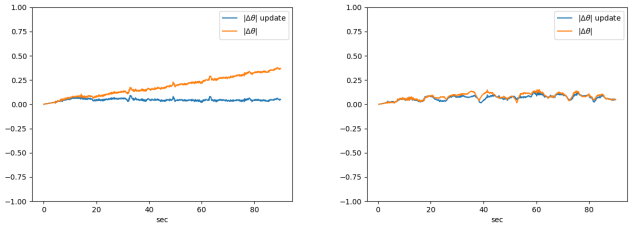
Table I shows the result of the proposed ESKF algorithm using the **GUS** error-metric. The mean and max values are shown for the algorithm both with and without the update step. Sequence 1-6 are recordings from the RepoIMU dataset. Sequence 1 is a static recording, while sequences 4-6 are more dynamic than sequence 2 and 3. The Custom sequence is our custom-made dataset.

[2]http://kgwisc.aei.polsl.pl/index.php/pl/dataset/60-repoimu
[3]https://youtu.be/y7lyQ6JPYD8

| | Update off | | Update on | |
|---|---|---|---|---|
| Sequence | Mean | Max | Mean | Max |
| 1 | 0.397 | 0.809 | 0.097 | 0.174 |
| 2 | 0.176 | 0.360 | 0.043 | 0.081 |
| 3 | 0.184 | 0.378 | 0.045 | 0.091 |
| 4 | 0.063 | 0.121 | 0.053 | 0.117 |
| 5 | 0.061 | 0.142 | 0.055 | 0.125 |
| 6 | 0.079 | 0.151 | 0.066 | 0.128 |
| Custom | 0.249 | 0.558 | 0.189 | 0.557 |

TABLE I: Results showing the mean and max **GUS** error, with and without the update step using accelerometer measurements. Sequences 1-6 is from the RepoIMU dataset, and Custom is the custom-made dataset.

The results show that incorporating the update step, produces more accurate results for all sequences. Considering the less dynamic sequences (1-3), versus the more dynamic sequences (4-6 and Custom), we can see hat the update step has a bigger impact on the sequences with lower dynamics, which is also illustrated in Figure 2. It is likely that this is caused by the accelerometer measurements being disturbed by linear acceleration caused by translational movement of the sensor body, resulting in inaccurate observations of the gravity vector. Another interesting observation is that without the update step (using only gyro predictions), the overall error is lower for the higher dynamic sequences than for the lower dynamic sequences. A hypothesis is that during more dynamic motion, and rotation of the sensor body around all axes, the effect of gyro bias could potentially be "canceled out" at angles of $\pm\pi$ radians. The evolution of the **GUS** error for the remaining sequences are given in Appendix A.




(a) Low dynamics (Sequence 3) (b) High dynamics (Sequence 6)

Fig. 2: Evolution of the **GUS** error for low and high dynamics

Figure 3 and 4 shows the quaternion components for the static sequence (sequence 1), without and with the update step, respectively. It showcases the drift caused by the gyro bias, as well as the issues regarding only using the accelerometer for correction. The accelerometer successfully compensates for the gyro bias, but only in 2 degrees of freedom. The accelerometer can not observe information about orientation around the axis parallel to the gravity vector, hence, the rotation error around the z-axis grows over-time.

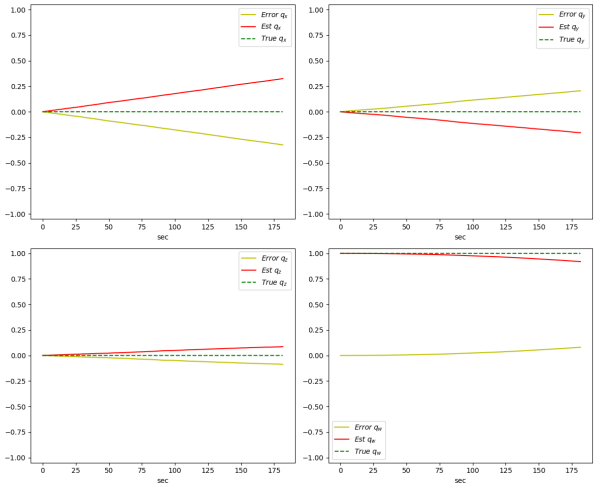



Fig. 3: Quaternion components for Sequence 1

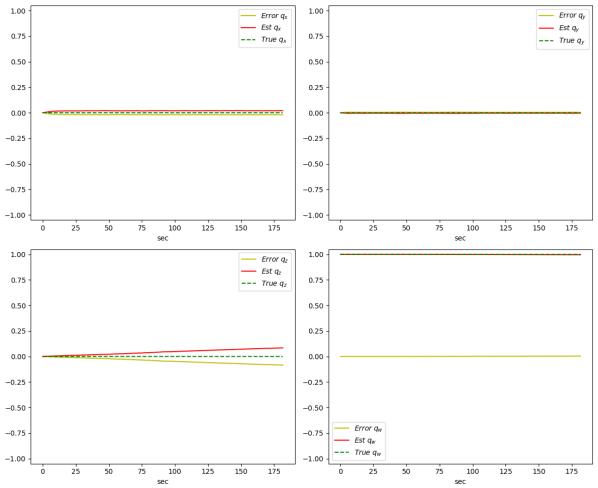



Fig. 4: Quaternion components for Sequence 1 with update

The performance of the algorithm on the custom dataset shows larger errors compared to the errors on the RepoIMU dataset, but still manages to follow the ground truth orientation approximately. The quaternion components are shown without and with the update step in Figure 5 and 6, respectively. Firstly, the quality of the IMU and motion capture system is of lower quality than that of the RepoIMU dataset, which can affect the performance of the estimated orientation. Secondly, improper data scaling and time-synchronization can also be a source of error. Finally, modelling errors (sensor bias and misalignment) are presumably a source of error for all sequences. Plots of the quaternion components for the remaining sequences are shown in Appendix A.

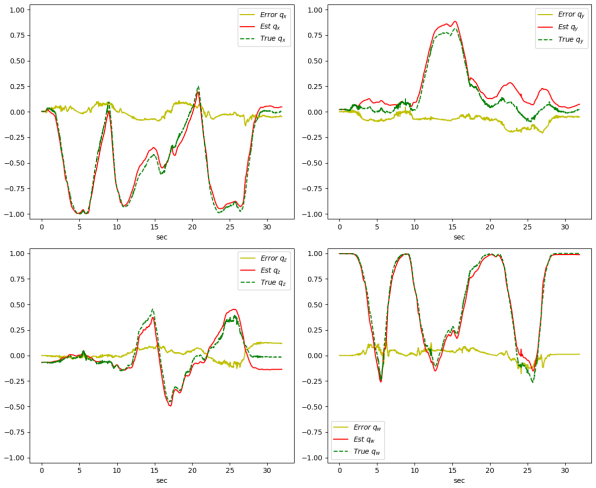



Fig. 5: Quaternion components for Custom sequence

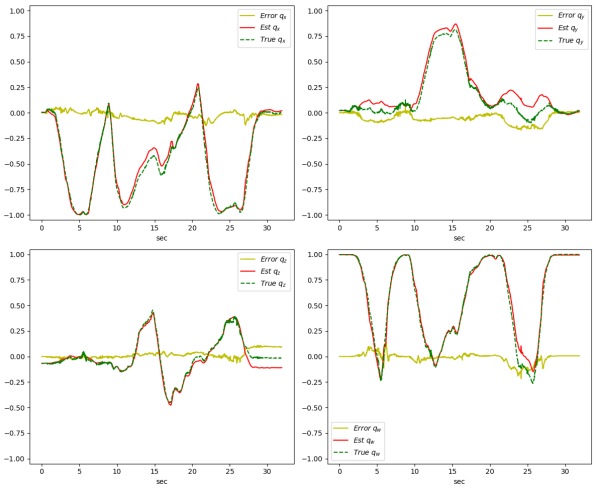



Fig. 6: Quaternion components for Custom sequence with update

## VI. Summary and Conclusions

In this work we have investigated the performance of a quaternion-based ESKF, incorporating gyro and accelerometers measurements, where modelling of bias terms and sensor misalignment have been ignored. We evaluated the proposed ESKF on a publicly available dataset, as well as on a custom-made dataset. Using the gyro for prediction, and accelerometer for correction in the update step, we performed an ablation study, investigating the impact of the acceleration correction. Results show that the accelerometer updates improves the orientation estimation performance in both low and high dynamic scenarios, and is able to partially compensate the gyro drift. Presumably, modelling of sensor bias would reduce the impact of the gyro drift. However, to account for the drift around the gravity vector, 3 DoF orientation measurements would be necessary, and can be achieved by fusion with magnetometer and/or GPS data.

Issues regarding the custom-made dataset have been discussed, but the overall result show that the motion capture procedure for data acquisition is a reasonable proof-of-concept that could be used in future work.

## References

[1] Du Q. Huynh. Metrics for 3d rotations: Comparison and analysis. 35(2):155–164.

[2] E. LEFFERTS, Landis Markley, and M. SHUSTER. *Kalman filtering for spacecraft attitude estimation*.

[3] S.I. Roumeliotis, G.S. Sukhatme, and G.A. Bekey. Circumventing dynamic modeling: evaluation of the error-state kalman filter applied to mobile robot localization. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, volume 2, pages 1656–1663 vol.2. ISSN: 1050-4729.

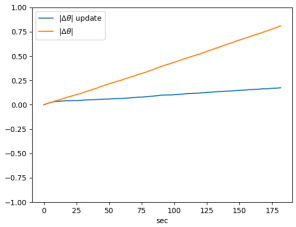
[4] Joan Solà. Quaternion kinematics for the error-state kalman filter.

[5] Joan Solà, Jeremie Deray, and Dinesh Atchuthan. A micro lie theory for state estimation in robotics.

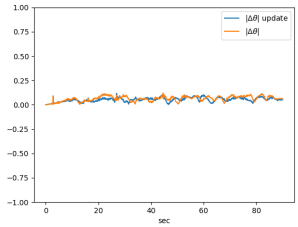
[6] Agnieszka Szczesna, Przemysław Skurowski, Przemyslaw Pruszowski, Damian Peszor, Marcin Paszkuta, and Konrad Wojciechowski. *Reference Data Set for Accuracy Evaluation of Orientation Estimation Algorithms for Inertial Motion Capture Systems*, volume 9972. Pages: 520.

[7] Nikolas Trawny and Stergios I Roumeliotis. Indirect kalman filter for 3d attitude estimation. page 25.

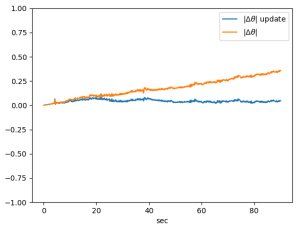
## APPENDIX



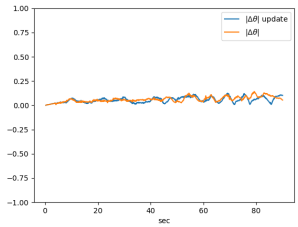

(a) Sequence 1



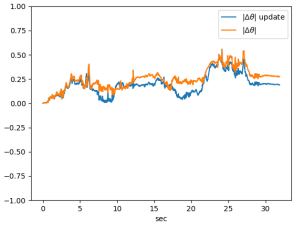

(b) Sequence 4




(c) Sequence 2




(d) Sequence 5




(e) Custom

Fig. 7: Evolution of the **GUS** error.

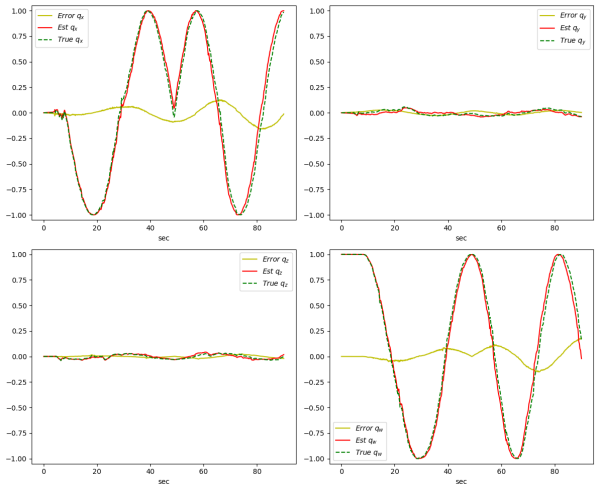



Fig. 8: Quaternion components for Sequence 2

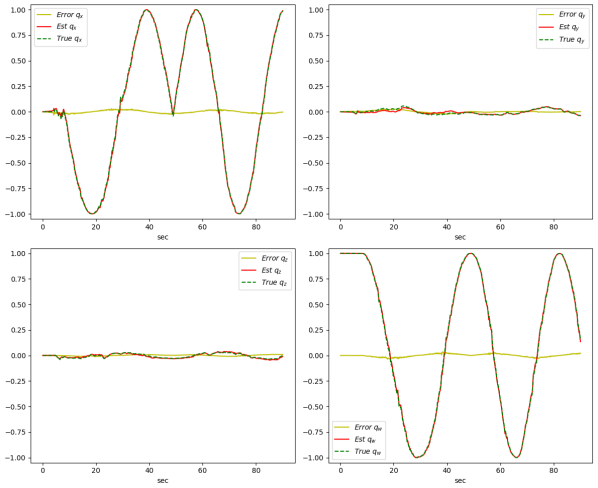



Fig. 9: Quaternion components for Sequence 2 with update

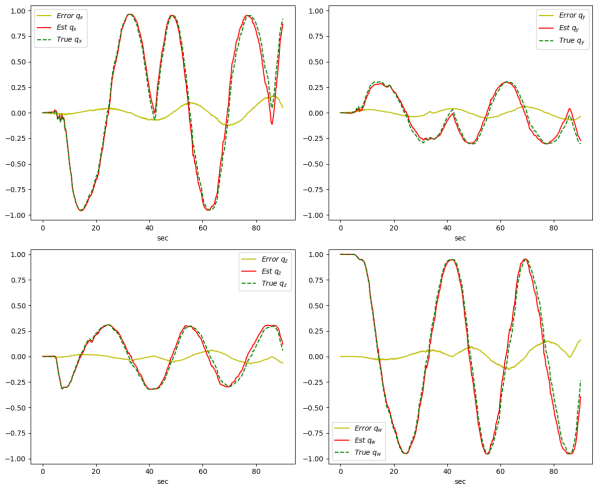



Fig. 10: Quaternion components for Sequence 3

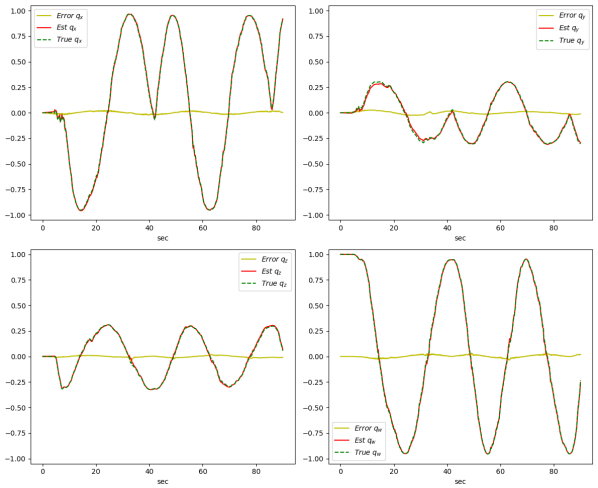



Fig. 11: Quaternion components for Sequence 3 with update

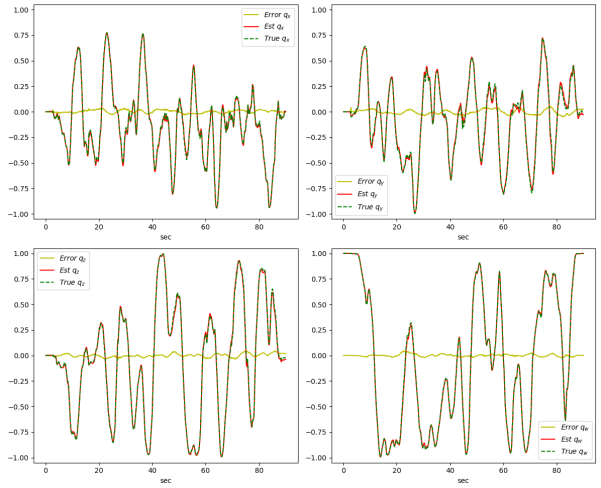


Fig. 12: Quaternion components for Sequence 4

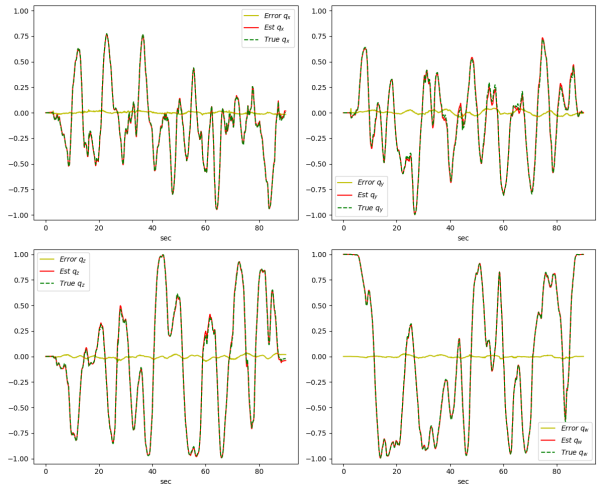


Fig. 13: Quaternion components for Sequence 4 with update

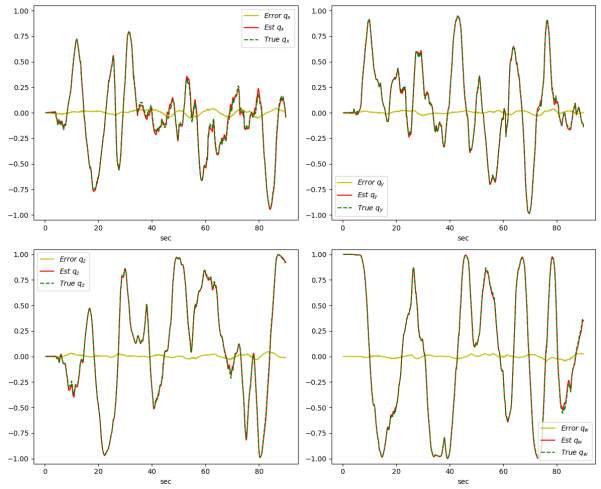


Fig. 14: Quaternion components for Sequence 5

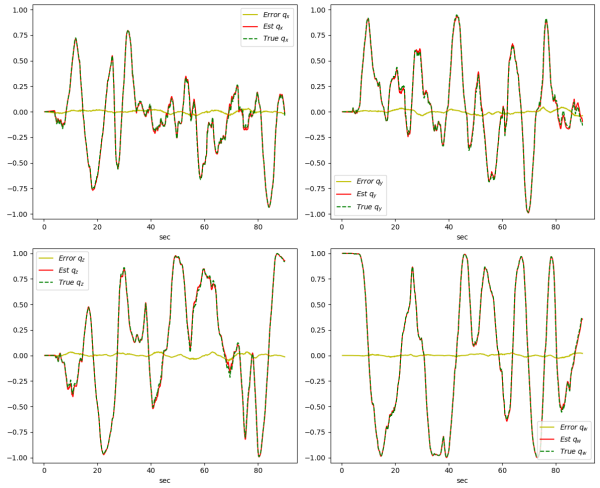


Fig. 15: Quaternion components for Sequence 5 with update

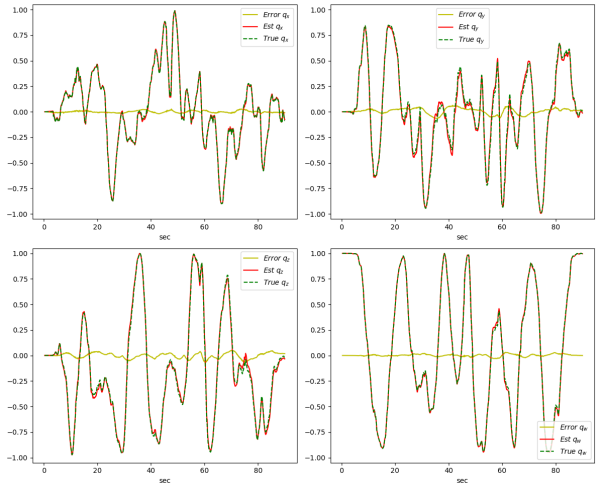


Fig. 16: Quaternion components for Sequence 6

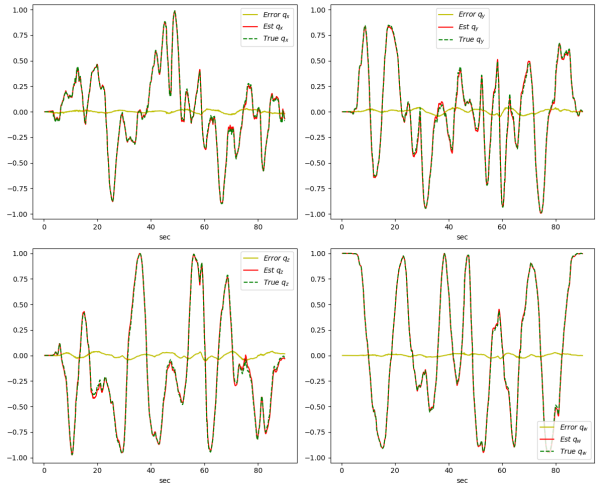


Fig. 17: Quaternion components for Sequence 6 with update