Nonlinear Data Analytics Final Project

John Kath
john.kath@cgu.edu

Contents

1 Project Overview

1.1 Description

Project Type (1)

1. Application of existing algorithm to a new problem and potentially new data.

1.2 Requirements

- **Partner:** Work independently.
- **Dataset:** HMOG cell phone data from 9 users / 24 sessions-activities each. Applied data analytics and ML methods using accelerometer, gyroscope and magnetometer data (3-axis each).
- **Format:** LaTeX NIPS
- **Code Style:** Used suggested code style guidelines (cookiecutter data science) with MIT open-source license.
- **Programming Tools & Hardware:** Python/Jupyter Notebook with Scikit Kinematics library, Matlab with Sensor Fusion and Tracking toolbox.

1.3 Project goals

This project uses probabilistic models and data analytics methods introduced in the paper [1] *Using Inertial Sensors for Position and Orientation Estimation* to estimate the position and orientation (pose) of a users cell phone during an activity. Kalman filtering fusion algorithms are used in Matlab and Madgwick's fusion algorithm is used in Scikit Kinematics to predict orientation of users cell phone during activities. Cell phone 3D accelerometer, gyroscope and magnetometer data is obtained for analysis from the HMOG dataset associated with the article [2] *HMOG: New Behavioral Biometric Features for Continuous Authentication of Smartphone Users*.

The techniques for data analytics used in the project as discussed in [1] *Using Inertial Sensors for Pose Estimation* are:

- **(Ch2) Inertial Sensors:** Coordinate frames, angular velocity, specific force, sensor error.
- **(Ch3) Probabilistic Models:** Parameterizing/probabilistic orientation modeling (Euler angles, Unit quaternions), measurement/probabilistic models for pose estimation.
- **(Ch4) Estimating Position and Orientation:** Smoothing in optimized frame (Gauss-Newton estimation, uncertainty), smoothing estimation of orientation using optimization, filtering estimate of orientation using optimization, filtering estimate in optimization framework, extended Kalman filtering / complementary filtering.

2 Math Models for Pose/Attitude Estimation

NOTE: The following sections restate material from reference [1].

This section gives the background material and mathematical models used to estimate the orientation of a device given IMU data. These techniques are applied in both the Matlab Sensor Fusion and Tracking toolbox and Python Scikit Kinematics library algorithms. These algorithms will be discussed with analysis results in the next section.

2.1 HMOG cell phone data

3D accelerometer, gyroscope and magnetometer data (3 axis each) [2] HMOG cell phone data for a given user/activity is used as an input to our pose algorithms. The algorithm output is an estimation of orientation (and possibly position). Three algorithms for estimating position and orientation will be introduced in the following sections.

2.1.1 Measurement models

Accelerometer measurement model

The accelerometer measures the specific force f_t^{b} at each time instance t. Our simplified measurement model is

$$y_{\mathrm{a},t} = R_t^{\mathrm{bn}}(a_{\mathrm{nn}}^{\mathrm{n}} - g^{\mathrm{n}}) + \delta_{\mathrm{a},t}^{\mathrm{b}} + e_{\mathrm{a},t}^{\mathrm{b}}. \tag{1}$$

Gyroscope measurement model

The gyroscope measures the angular velocity $\omega_{\text{ib}}^{\text{b}}$ at each time instance t. Our simplified measurement model is

$$y_{\omega,t} = \omega_{\text{nb},t}^{\text{b}} + \delta_{\omega,t}^{\text{b}} + e_{\omega,t}^{\text{b}}. \tag{2}$$

Magnetometer measurement model

Magnetometers measure the local magnetic field, consisting of both the earth magnetic field and the magnetic field due to the presence of magnetic material. Assuming that the magnetometer only measures the local magnetic field, its measurements $y_{\text{m},t}$ can be modeled as

$$y_{\text{m},t} = R_t^{\text{bn}} m^{\text{n}} + e_{\text{m},t}, \tag{3}$$

2.2 Probabilistic models

Most complexity in pose estimation lies in the nonlinear nature of the orientation and the fact that orientation can be parametrized in different ways. When all the measurement data is used in our model this is know as *smoothing*. This is compuation intensive and assumes all measurement data is available for our estimate. The other class of estimation we will model is know as *filtering*. In filtering we estimate x_t using all measurements up to and including time t.

2.3 Pose estimation

For pose estimation, we model the accelerometer and gyroscope measurements as inputs to the dynamics. Hence, the state vector consists of the position p_t^{n}, the velocity v_t^{n} and a parametrization of the orientation. In summary, this leads to the following state space model for pose estimation

$$\begin{pmatrix} p_{t+1}^{\text{n}} \\ v_{t+1}^{\text{n}} \\ q_{t+1}^{\text{nb}} \end{pmatrix} = \begin{pmatrix} p_t^{\text{n}} + T v_t^{\text{n}} + \frac{T^2}{2}\left(R_t^{\text{nb}}(y_{\text{a},t} - \delta_{\text{a},t}) + g^{\text{n}} + e_{\text{p,a},t}\right) \\ v_t^{\text{n}} + T\left(R_t^{\text{nb}}(y_{\text{a},t} - \delta_{\text{a},t}) + g^{\text{n}} + e_{\text{v,a},t}\right) \\ q_t^{\text{nb}} \odot \exp_{\mathsf{q}}\left(\frac{T}{2}(y_{\omega,t} - \delta_{\omega,t} - e_{\omega,t})\right) \end{pmatrix}, \tag{4a}$$

$$y_{\text{p},t} = p_t^{\text{n}} + e_{\text{p},t}, \tag{4b}$$

where

$$e_{\text{p,a},t} \sim \mathcal{N}(0, \Sigma_{\text{a}}), \qquad e_{\text{v,a},t} \sim \mathcal{N}(0, \Sigma_{\text{a}}), \tag{4c}$$

$$e_{\text{p},t} \sim \mathcal{N}(0, \Sigma_{\text{p}}), \qquad e_{\omega,t} \sim \mathcal{N}(0, \Sigma_{\omega}), \tag{4d}$$

with $\Sigma_{\text{a}} = \sigma_{\text{a}}^2 \mathcal{I}_3$ and $\Sigma_\omega = \sigma_\omega^2 \mathcal{I}_3$.

(4e)

2.4 Orientation estimation

For orientation estimation, the state vector only consists of a parametrization of the orientation. We use the inertial sensors in combination with the magnetometer measurements to estimate the orientation. This leads to the following state space model for orientation estimation,

$$q_{t+1}^{\text{nb}} = q_t^{\text{nb}} \odot \exp_{\mathsf{q}}\left(\frac{T}{2}(y_{\omega,t} - \delta_\omega - e_{\omega,t})\right), \tag{5a}$$

$$y_{\text{a},t} = -R_t^{\text{bn}} g^{\text{n}} + e_{\text{a},t}, \tag{5b}$$

$$y_{\text{m},t} = R_t^{\text{bn}} m^{\text{n}} + e_{\text{m},t}, \tag{5c}$$

where (5a) describes the dynamics while (5b) and (5c) describe the measurement models and

$$e_{\omega,t} \sim \mathcal{N}(0, \Sigma_\omega), \qquad e_{\text{a},t} \sim \mathcal{N}(0, \Sigma_{\text{a}}), \qquad e_{\text{m},t} \sim \mathcal{N}(0, \Sigma_{\text{m}}), \tag{5d}$$

with $\Sigma_\omega = \sigma_\omega^2 \mathcal{I}_3$ and $\Sigma_{\text{a}} = \sigma_{\text{a}}^2 \mathcal{I}_3$. The initial orientation is given by the QUEST algorithm.

2.5 Estimating position and orientation

We will focus on position and orientation estimation using the probabilistic models developed. Algorithm 1 is a orientation estimate based on a smoothing. Algorithms 2 and 3 use filtering to estimate orientation. Our focus will be orientation estimation problems since they illustrate the most important parts of the pose estimation. Most complexities lie in the parametrization of the orientation and in the nonlinear nature of the orientation.

2.6 Gauss-Newton optimization

To obtain a smoothing estimate of the position and orientation using optimization, we first recognize that for our models (4) and (5), all probability distributions are Gaussian. Let us therefore consider a slightly more general problem where the objective function consists of the product of Gaussian probability functions $p(e_i(x_{1:N}))$, $i = 1, \ldots, M$. Hence, the optimization problem can be written as

$$\hat{x}_{1:N} = \arg\min_{x_{1:N}} - \sum_{i=1}^{M} \log p\left(e_i(x_{1:N})\right). \tag{6}$$

The probability distribution of $e_i(x)$ is given by

$$p\left(e_i(x_{1:N})\right) = \frac{1}{\sqrt{(2\pi)^{n_e} \det \Sigma_i}} \exp\left(-\tfrac{1}{2} e_i^{\mathsf{T}}(x_{1:N}) \Sigma_i^{-1} e_i(x_{1:N})\right). \tag{7}$$

Omitting the terms independent of $x_{1:N}$, the optimization problem (6) reduces to

$$\hat{x}_{1:N} = \arg\min_{x_{1:N}} \tfrac{1}{2} \sum_{i=1}^{M} \|e_i(x_{1:N})\|^2_{\Sigma_i^{-1}}, \tag{8}$$

with $\|e_i(x_{1:N})\|^2_{\Sigma_i^{-1}} = e_i^{\mathsf{T}}(x_{1:N}) \Sigma_i^{-1} e_i(x_{1:N})$. The function that is being minimized in optimization problems, is often referred to as the *objective function*.

The solution to (8) can be found by studying the shape of the objective function as a function of $x_{1:N}$. This can be characterized in terms of the *gradient* $\mathcal{G}(x_{1:N})$ and *Hessian* $\mathcal{H}(x_{1:N})$, which provide information about the slope and curvature of the function, respectively. Defining

$$e_i^{\mathsf{T}}(x_{1:N}) \Sigma_i^{-1} e_i(x_{1:N}) = \varepsilon_i^{\mathsf{T}} \varepsilon_i, \qquad \varepsilon_i = \Sigma_i^{-1/2} e_i(x_{1:N}),$$

and the stacked variables

$$\varepsilon = \begin{pmatrix} \varepsilon_1^{\mathsf{T}} & \cdots & \varepsilon_M^{\mathsf{T}} \end{pmatrix}^{\mathsf{T}},$$

the gradient and the Hessian are given by

$$\mathcal{G}(x_{1:N}) = \sum_{i=1}^{M} \left(\tfrac{\partial \varepsilon_i}{\partial x_{1:N}}\right)^{\mathsf{T}} \varepsilon_i = \mathcal{J}^{\mathsf{T}}(x_{1:N}) \varepsilon, \tag{9a}$$

$$\begin{aligned} \mathcal{H}(x_{1:N}) &= \sum_{i=1}^{M} \left(\left(\tfrac{\partial \varepsilon_i}{\partial x_{1:N}}\right)^{\mathsf{T}} \tfrac{\partial \varepsilon_i}{\partial x_{1:N}} + \varepsilon_i^{\mathsf{T}} \tfrac{\partial^2 \varepsilon_i}{\partial x_{1:N}^2} \right) \\ &= \mathcal{J}^{\mathsf{T}}(x_{1:N}) \mathcal{J}(x_{1:N}) + \sum_{i=1}^{M} \varepsilon_i^{\mathsf{T}} \tfrac{\partial^2 \varepsilon_i}{\partial x_{1:N}^2}. \end{aligned} \tag{9b}$$

Note that for notational convenience, we have omitted the explicit dependence of ε on $x_{1:N}$. In (9), we introduced the notation $\mathcal{J}(x_{1:N})$, which is the *Jacobian* of the vector ε with respect to $x_{1:N}$ as

$$\mathcal{J}(x_{1:N}) = \begin{pmatrix} \frac{\partial \varepsilon_1}{\partial x_1} & \cdots & \frac{\partial \varepsilon_1}{\partial x_N} \\ \vdots & & \vdots \\ \frac{\partial \varepsilon_{M n_\varepsilon}}{\partial x_1} & \cdots & \frac{\partial \varepsilon_{M n_\varepsilon}}{\partial x_N} \end{pmatrix}, \tag{10}$$

where n_ε is the length of the vector ε_i. Instead of computing the true Hessian (9b), we compute an approximation of it given by

$$\hat{\mathcal{H}}(x_{1:N}) = \mathcal{J}^{\mathsf{T}}(x_{1:N}) \mathcal{J}(x_{1:N}). \tag{11}$$

This has the benefit of not having to compute second derivatives, at the same time as it guarantees that the Hessian is positive semidefinite. The downside of using (11) is that it introduces an approximation.

The gradient and the (approximate) Hessian can be used to find the minimum of the objective function. For our models (4) and (5), in which the functions $e_i(x_{1:N})$ are nonlinear, an estimate $\hat{x}_{1:N}$ can iteratively be computed as

$$\hat{x}_{1:N}^{(k+1)} = \hat{x}_{1:N}^{(k)} - \beta^{(k)} \left(\hat{\mathcal{H}}(\hat{x}_{1:N}^{(k)})\right)^{-1} \mathcal{G}(\hat{x}_{1:N}^{(k)}), \tag{12}$$

where k denotes the iteration number. The *step length* $\beta^{(k)}$ is computed for instance using a backtracking line search. The *search direction* is computed as $\left(\hat{\mathcal{H}}(\hat{x}_{1:N}^{(k)})\right)^{-1} \mathcal{G}(\hat{x}_{1:N}^{(k)})$. Note that an initial point $\hat{x}_{1:N}^{(0)}$ needs to be chosen close enough to the desired minimum to ensure convergence to this minimum.

At each Gauss-Newton iteration (12), we estimate the state vector $\eta_{1:N}^{\text{n}}$. Before starting the next iteration, the linearization points $\tilde{q}_{1:N}^{\text{nb}}$ are updated and the state vector $\eta_{1:N}^{\text{n}}$ is reset to zero.

2.7 Computing the uncertainty

We are not only interested in obtaining point estimates of the position and orientation, but also in estimating their uncertainty. We can therefore approximate the covariance of our estimates as

$$\operatorname{cov}(\hat{x}_{1:N}) = \left(\mathcal{J}^{\mathsf{T}}(\hat{x}_{1:N})\mathcal{J}(\hat{x}_{1:N})\right)^{-1}. \tag{13}$$

2.8 Smoothing estimation algorithms

Algorithm 1 details the steps necessary to apply the Gauss-Newton optimization method to obtain a smoothing estimate of orientation with associated error covariance.

Algorithm 1 Smoothing estimates of the orientation using optimization

INPUTS: An initial estimate of the orientation $\tilde{q}_{1:N}^{\text{nb},(0)}$, inertial data $\{y_{\text{a},t}, y_{\omega,t}\}_{t=1}^{N}$, magnetometer data $\{y_{\text{m},t}\}_{t=1}^{N}$ and covariance matrices Σ_ω, Σ_a and Σ_m.
OUTPUTS: An estimate of the orientation $\hat{q}_{1:N}^{\text{nb}}$ and optionally its covariance $\operatorname{cov}(\hat{\eta}_{1:N}^{\text{n}})$.

```
1. Set $\hat{\eta}_t^{\text{n},(0)} = 0_{3\times 1}$ for $t = 1, \ldots, N$, set $k = 0$ and compute $\breve{q}_1^{\text{nb}}$ and $\Sigma_{\eta,\text{i}}$.
2. while termination condition is not satisfied do
    (a) Compute the gradient (9a) and the approximate Hessian (11) of the orientation smoothing
        problem using the expressions for the different parts of the cost function and their Jacobians.
    (b) Apply the update (12) to obtain $\hat{\eta}_{1:N}^{\text{n},(k+1)}$.
    (c) Update the linearization point as
            $\tilde{q}_t^{\text{nb},(k+1)} = \exp_{\text{q}}\left(\frac{\hat{\eta}_t^{\text{n},(k+1)}}{2}\right) \odot \tilde{q}_t^{\text{nb},(k)}$,     (14)
        and set $\hat{\eta}_t^{\text{n},(k+1)} = 0_{3\times 1}$ for $t = 1, \ldots, N$.
    (d) Set $k = k + 1$.
   end while
3. Set $\hat{q}_{1:N}^{\text{nb}} = \tilde{q}_{1:N}^{\text{nb},(k)}$.
4. Optionally compute
            $\operatorname{cov}(\hat{\eta}_{1:N}^{\text{n}}) = \left(\mathcal{J}^{\mathsf{T}}(\hat{\eta}_{1:N}^{\text{n}})\mathcal{J}(\hat{\eta}_{1:N}^{\text{n}})\right)^{-1}$.     (15)
```

2.9 Filtering estimation algorithms

Algorithm 2 (this is Algorithm 3 in referenced article [1]) details the steps necessary to obtain the Kalman filtering estimate of orientation with associated error covariance.

Algorithm 2 Orientation estimation using an EKF with quaternion states

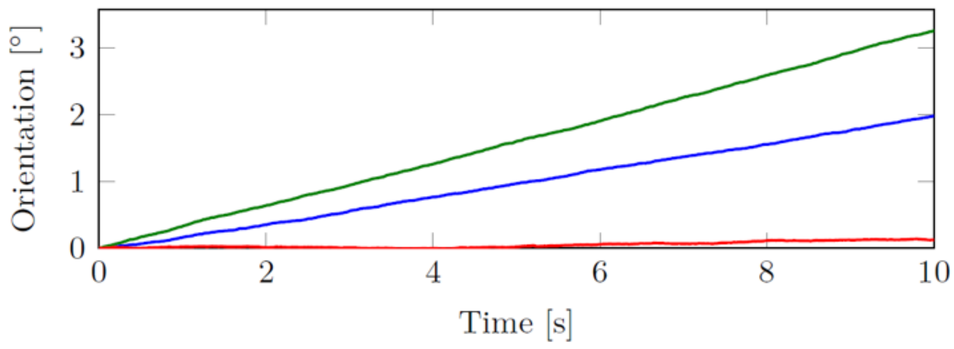
INPUTS: Inertial data $\{y_{\mathrm{a},t}, y_{\omega,t}\}_{t=1}^{N}$, magnetometer data $\{y_{\mathrm{m},t}\}_{t=1}^{N}$ and covariance matrices Σ_ω, Σ_a and Σ_m.
OUTPUTS: An estimate of the orientation $\hat{q}^{\mathrm{nb}}_{t|t}$ and its covariance $P_{t|t}$ for $t = 1, \dots N$.

```
1. Compute $\breve{q}^{\mathrm{nb}}_1$ and $\Sigma_\mathrm{i}$ as described in §?? and set $\hat{q}^{\mathrm{nb}}_{1|1} = \breve{q}^{\mathrm{nb}}_1$ and $P_{1|1} = \Sigma_{\mathrm{q,i}}$.
2. For $t = 2, \dots, N$ do
   (a) Time update
       $$\hat{q}^{\mathrm{nb}}_{t|t-1} = \hat{q}^{\mathrm{nb}}_{t-1|t-1} \odot \exp_\mathsf{q}\left(\tfrac{T}{2} y_{\omega,t-1}\right), \quad (16a)$$
       $$P_{t|t-1} = F_{t-1} P_{t-1|t-1} F_{t-1}^\mathsf{T} + G_{t-1} Q G_{t-1}^\mathsf{T}, \quad (16b)$$
       with $Q = \Sigma_\omega$ and
       $$F_{t-1} = \left(\exp_\mathsf{q}(\tfrac{T}{2} y_{\omega,t-1})\right)^\mathsf{R}, \; G_{t-1} = -\tfrac{T}{2}\left(\hat{q}^{\mathrm{nb}}_{t-1|t-1}\right)^\mathsf{L} \tfrac{\partial \exp_\mathsf{q}(e_{\omega,t-1})}{\partial e_{\omega,t-1}}.$$
   (b) Measurement update
       $$\tilde{q}^{\mathrm{nb}}_{t|t} = \hat{q}^{\mathrm{nb}}_{t|t-1} + K_t \varepsilon_t, \quad (17a)$$
       $$\tilde{P}_{t|t} = P_{t|t-1} - K_t S_t K_t^\mathsf{T}, \quad (17b)$$
       with $\varepsilon_t$, $K_t$ and $S_t$ defined in (??) and
       $$y_t = \begin{pmatrix} y_{\mathrm{a},t} \\ y_{\mathrm{m},t} \end{pmatrix}, \qquad \hat{y}_{t|t-1} = \begin{pmatrix} -\hat{R}^{\mathrm{bn}}_{t|t-1} g^\mathrm{n} \\ \hat{R}^{\mathrm{bn}}_{t|t-1} m^\mathrm{n} \end{pmatrix},$$
       $$H_t = \begin{pmatrix} -\left.\frac{\partial R^{\mathrm{bn}}_{t|t-1}}{\partial q^{\mathrm{nb}}_{t|t-1}}\right|_{q^{\mathrm{nb}}_{t|t-1} = \hat{q}^{\mathrm{nb}}_{t|t-1}} g^\mathrm{n} \\ \left.\frac{\partial R^{\mathrm{bn}}_{t|t-1}}{\partial q^{\mathrm{nb}}_{t|t-1}}\right|_{q^{\mathrm{nb}}_{t|t-1} = \hat{q}^{\mathrm{nb}}_{t|t-1}} m^\mathrm{n} \end{pmatrix}, \qquad R = \begin{pmatrix} \Sigma_\mathrm{a} & 0 \\ 0 & \Sigma_\mathrm{m} \end{pmatrix}.$$
   (c) Renormalize the quaternion and its covariance as
       $$\hat{q}^{\mathrm{nb}}_{t|t} = \frac{\tilde{q}^{\mathrm{nb}}_{t|t}}{\|\tilde{q}^{\mathrm{nb}}_{t|t}\|_2}, \qquad P_{t|t} = J_t \tilde{P}_{t|t} J_t^\mathsf{T}, \quad (18)$$
       with $J_t = \frac{1}{\|\tilde{q}^{\mathrm{nb}}_{t|t}\|_2^3} \tilde{q}^{\mathrm{nb}}_{t|t} \left(\tilde{q}^{\mathrm{nb}}_{t|t}\right)^\mathsf{T}$.
   end for
```

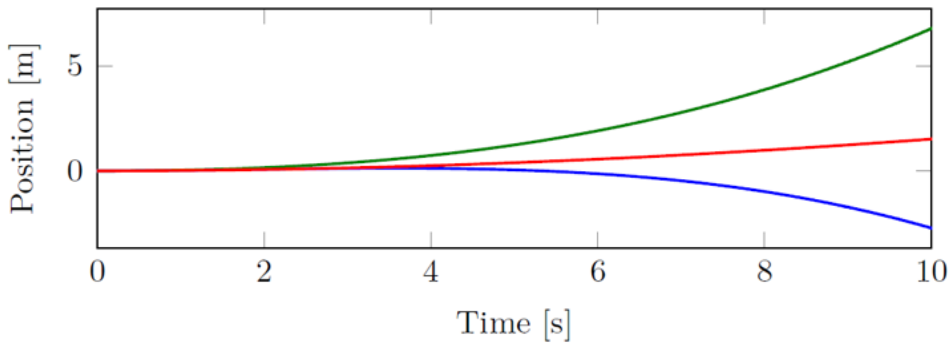
3 Analysis Results

We will be comparing HMOG user cell phone orientation estimates for two kinds of attitude and heading reference system (AHRS) algorithms: Kalman filters and Madgwick & Mahony filtering. These algorithms are know as *fusion algorithms* because they fuse accelerometer, gyroscope and magnetometer measurements to produce the optimal estimate of a devices orientation. The goal of these filters is to remove error introduced by IMU sensor measurements in the form of bias, drift and noise.

The gyroscope measures the angular rate of a moving object, the accelerometer measures the acceleration of a certain object, and the magnetometer measures the magnetic field. However, low-cost sensors have inherent drawbacks such as nonlinearity, random walk, temperature drift, etc. In order to obtain a reliable attitude solution, magnetic and inertial measurement unit (MIMU) sensor measurements have to be fused together using optimal sensor fusion algorithms. There are mainly two different fusion approaches. One category includes the *complementary filters* and the other relates to *Kalman filtering*.



(a) Integrated orientation for the position in x- (blue), y- (green) and z-direction (red).



(b) Integrated position for rotation around the x-axis (blue), the y-axis (green) and the z-axis (red).

Figure 1: Integration drift.

3.1 IMU sensor measurements

Axes definitions Device is equipped with three-axis accelerometer, gyroscope and vector magnetometer with mutually orthogonal axes: x axis is looking “forward”, z axis is looking “downwards”, y axis completes the right orthogonal system of axes.

Gyroscope measures positive angular rates being rotated counter-clockwise (if one watches from the end of the corresponding axis) in rad/sec.

Accelerometer measures the gravity force (it actually doesn’t - it can only measure active forces, and when it sits on the table it measures the force with which the table acts on the accelerometer body preventing it from falling right through the lid of the table) in m/sec^2.

Magnetometer measures the Earth’s magnetic field vector in $micro - tesla$.

3.2 IMU sensor errors

Integration of the gyroscope measurements provides information about the orientation of the sensor. After subtraction of the earth’s gravity, double integration of the accelerometer measurements provides information about the sensor’s position. To be able to subtract the earth’s gravity, the orientation of the sensor needs to be known. Hence, estimation of the sensor’s position and orientation are inherently linked when it comes to inertial sensors.

The process of integrating the measurements from inertial sensors to obtain position and orientation information, is called *deadreckoning*.

Inertial measurements are noisy and biased. Because of this, the integration steps from angular velocity to rotation and from acceleration to position introduce *integration drift*.

Accelerometers can itself measure the inclination, they have a slow response. Gyros measure angle rate change, fast response but the problems for gyros is the zero drift and it has to be compensated for any usable application.

While at steady state, a filter is used such that accelerometers track and eliminate the gyro drift in the vertical plane, where the gravity vector is used for inclination (roll, pitch), but in horizontal plane heading/yaw there is not such possibility, therefore magnetometer can be used. A magnetometer is mainly used for speed detection and with fusion algorithm it can eliminate the gyro offset.

3.3 Matlab Sensor Fusion and Tracking Toolbox

The Matlab Ahrsfilter and Imufilter algorithms were used to estimate orientation for 9 HMOG cell phone users 24 sessions each. Result images for 3 users (single session each) are shown in section four.

Filter Descriptions

FUSE = imufilter returns an indirect Kalman filter System object, FUSE, for fusion of accelerometer and gyroscope data to estimate device orientation. The filter uses a nine-element state vector to track error in the orientation estimate, the gyroscope bias estimate, and the linear acceleration estimate.

FUSE = ahrsfilter returns an indirect Kalman filter System object, FUSE, for sensor fusion of accelerometer, gyroscope, and magnetometer data to estimate device orientation and angular velocity. The filter uses a 12-element state vector to track the estimation error for the orientation, the gyroscope bias, the linear acceleration, and the magnetic disturbance.

The Ahrsfilter algorithnm provides a better estimate of orientation because the addition of magnetometer measurements allow for correcting gyroscope bias/drift. Generally, the Ahrsfilter orientation z-axis exhibits less variance and appears to be stabilized vs Imufilter orientation z-axis. On the other hand, the orientation y-axis and x-axis appear to be very close generally for both filters.

3.4 Python Scikit Kinematics Library

The scikit kinematics Kalman, Madgwick and Mahony filtering algorithms were used to estimate orientation for 3 HMOG cell phone users single session each. Result images for 3 users (single session each) are shown in section four.

Filter Descriptions

NOTE: all three filters use accelerometer, gyroscope and magnetometer measurements to estimate orientation.

Kalman Calculate the orientation from IMU data.

Madgwick Madgwick's gradient descent filter.

Mayhony Madgwick's implementation of Mayhony's AHRS algorithm.

The Madgwick and Mayhony algorithms give similar results and generally give better orientation estamates than the kalman filtering algorithm. The former fuse magnetometer measurements to correct for gyro error resulting in a better attitude estimation.

3.5 Conclusion and further work

The Matlab SF&T and scikit kinematics filtering algorithms did a nice job of orientation estimation for cell phone users.

For future work, I would like to implementing a basic 3D Kalman filtering algorithm and explore the error covariance matrix in more detail.

References

[1] Manon Kok, Jeroen D. Hol and Thomas B. Schon (2017), Using Inertial Sensors for Position and Orientation Estimation, *Foundations and Trends in Signal Processing: Vol. 11: No. 1-2*, pp 1-153.
`http://dx.doi.org/10.1561/2000000094`

[2] SITOVÁ, Zdeňka, Jaroslav ŠEDĚNKA, Qing YANG, Ge PENG, Gang ZHOU, Paolo GASTI and Kiran BALAGANI. HMOG: New Behavioral Biometric Features for Continuous Authentication of Smartphone Users. *IEEE Transactions on Information Forensics and Security, 2016, Vol. 11, No. 5*, p. 877 - 892. ISSN 1556-6013.
`http://dx.doi.org/10.1109/TIFS.2015.2506542`

[3] MATLAB Sensor Fustion and Tracking Toolbox.
`https://www.mathworks.com/help/fusion/`

[4] Scikit Kinematics Library.
`http://work.thaslwanter.at/skinematics/html`

[5] Open Source Sensor Fusion.
`https://github.com/memsindustrygroup/Open-Source-Sensor-Fusion/tree/master/docs`

[6] "Design, Implementation, and Experimental Results of a Quaternion-Based Kalman Filter for Human Body Motion Tracking" Yun, X. and Bachman, E.R., IEEE TRANSACTIONS ON ROBOTICS, VOL. 22, 1216-1227 (2006)

4 Analysis Images

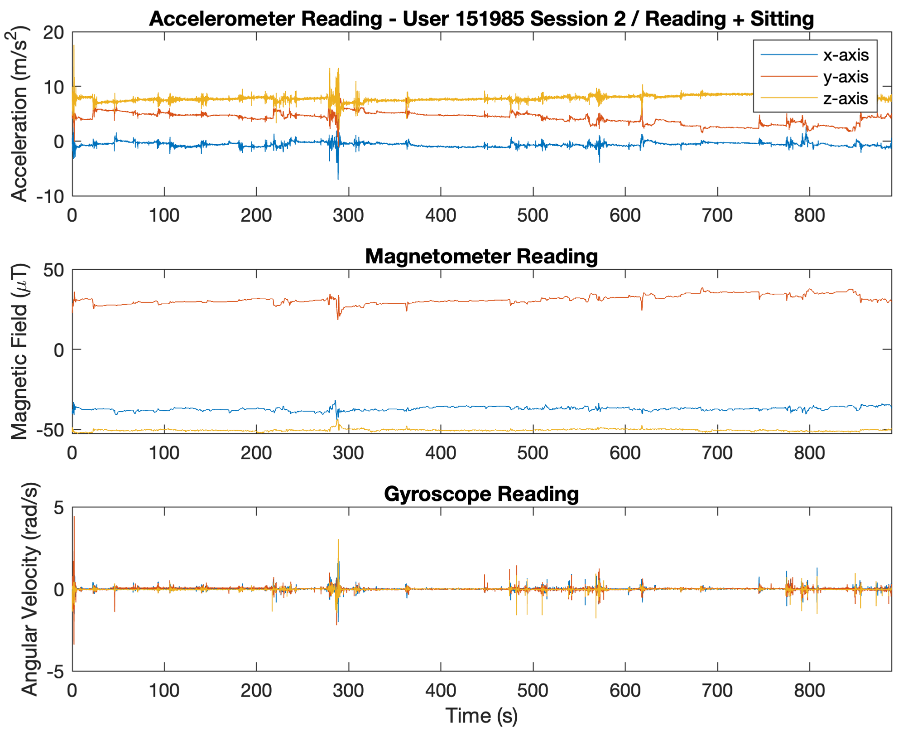


Figure 2: HMOG cell phone IMU data.

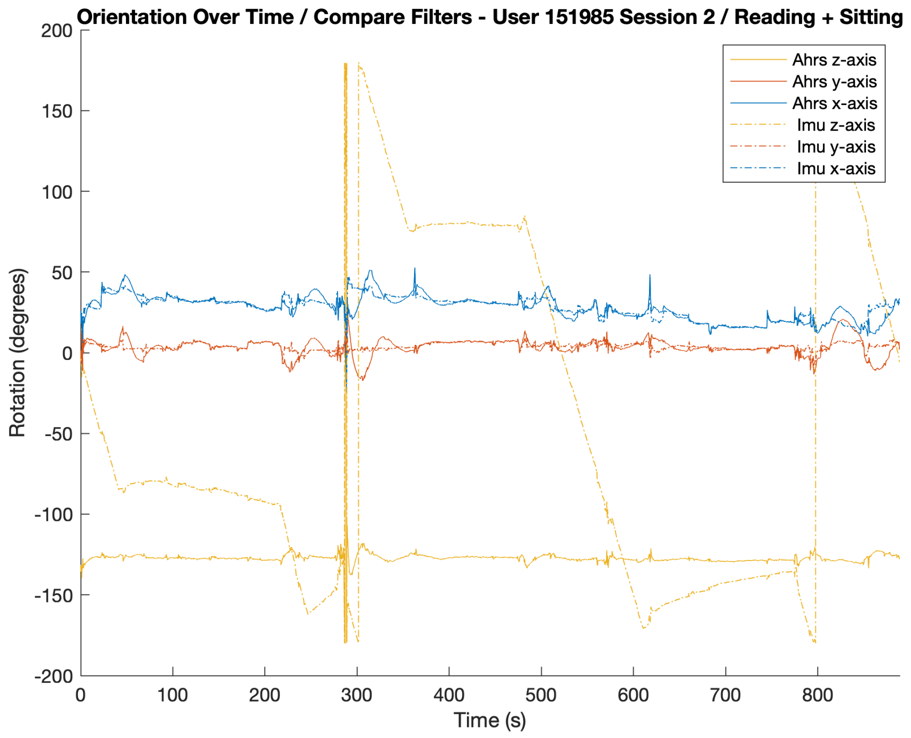


Figure 3: Comparison Matlab AHRS and IMU filtering algorithms.

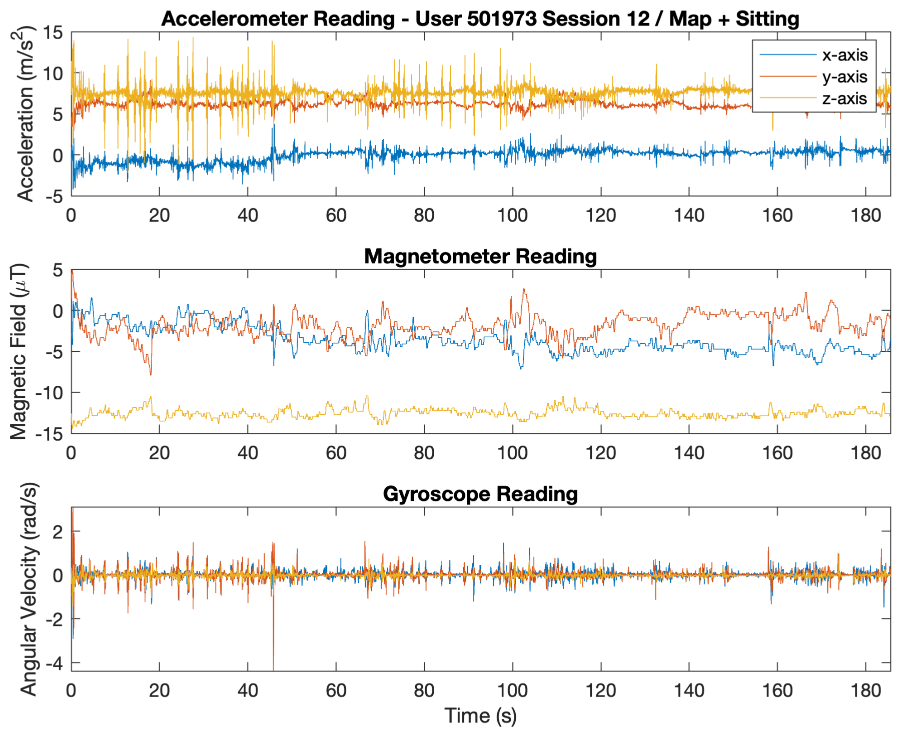


Figure 4: HMOG cell phone IMU data.

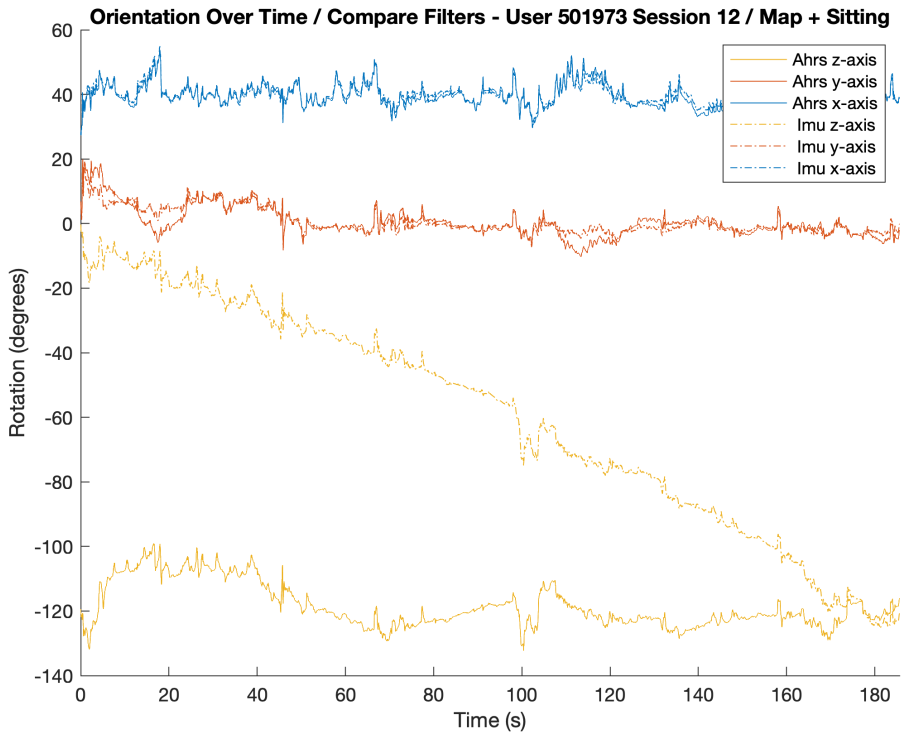


Figure 5: Comparison Matlab AHRS and IMU filtering algorithms.

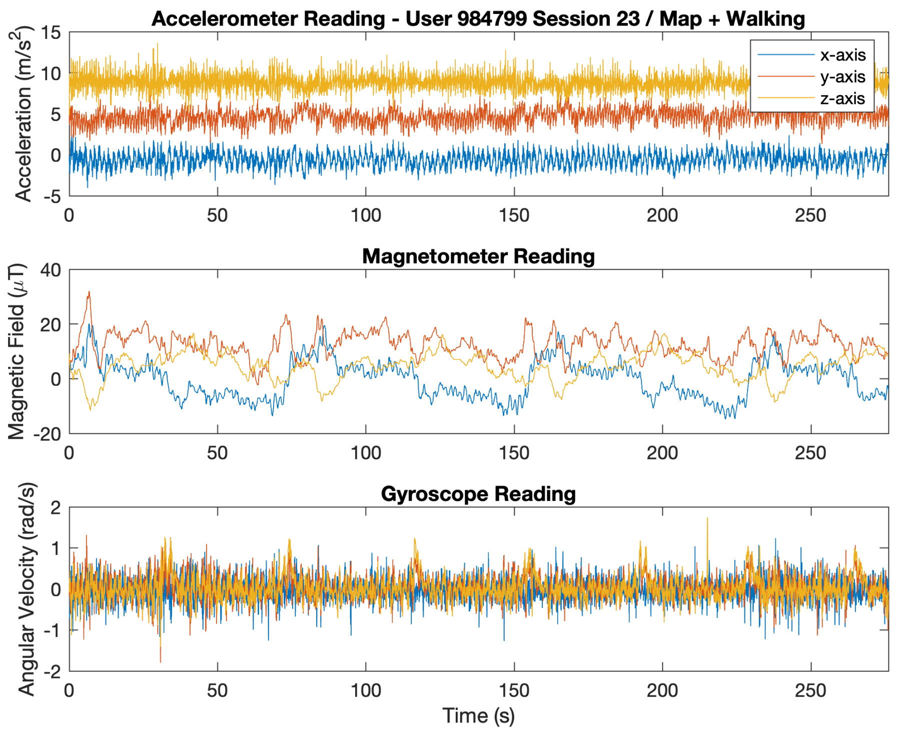


Figure 6: HMOG cell phone IMU data.

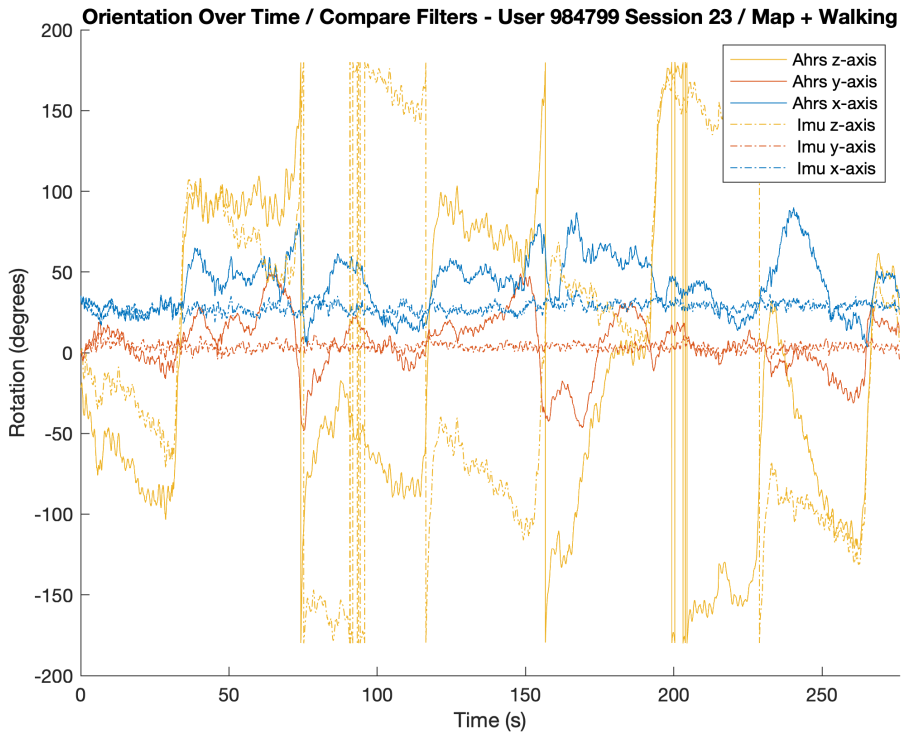


Figure 7: Comparison Matlab AHRS and IMU filtering algorithms.

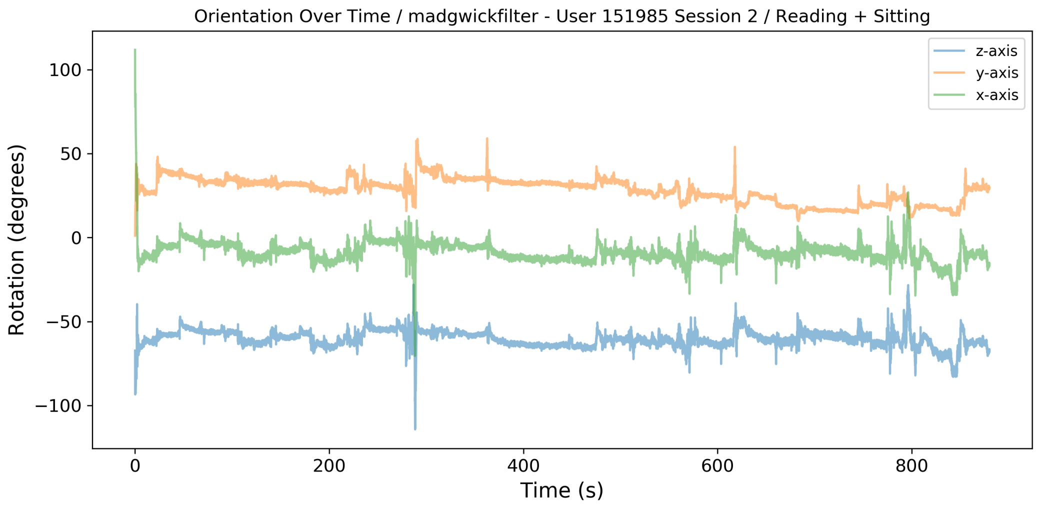


Figure 8: scikit kinematics AHRS Madgwick filter.

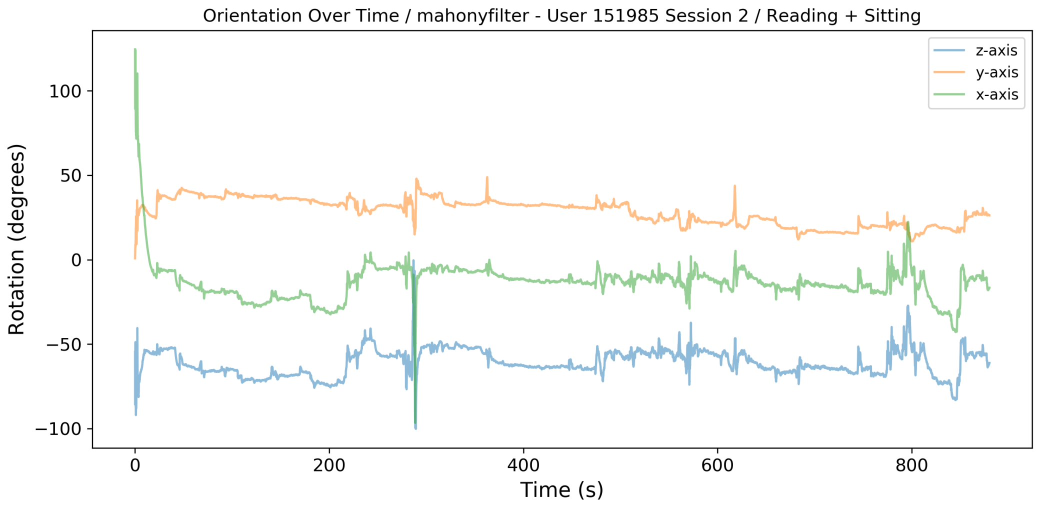


Figure 9: scikit kinematics AHRS Mahony filter.

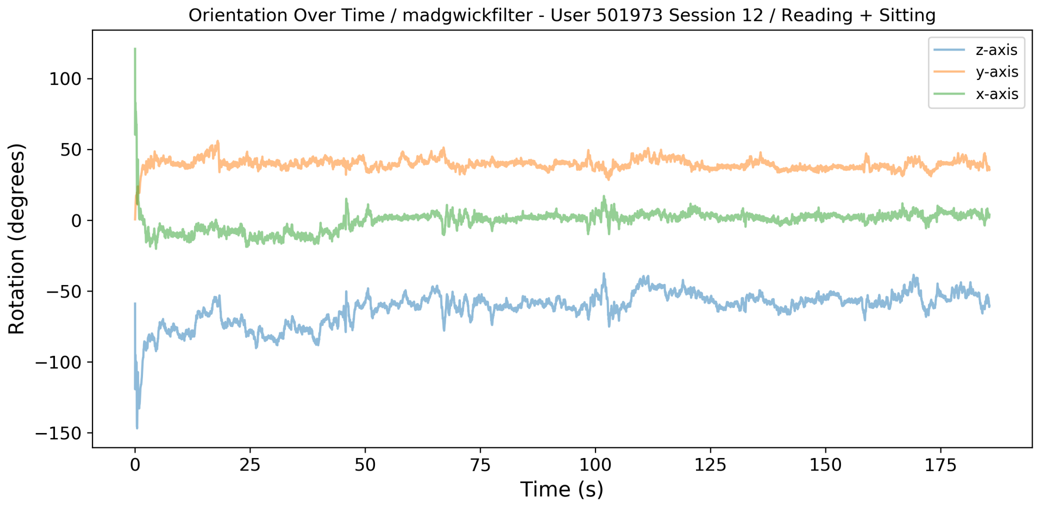


Figure 10: scikit kinematics AHRS Madgwick filter.

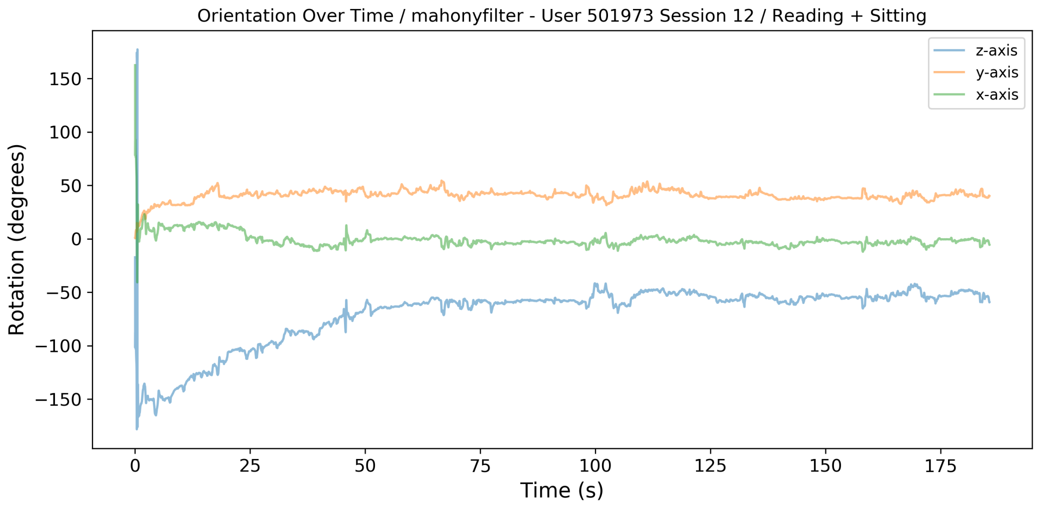


Figure 11: scikit kinematics AHRS Mahony filter.

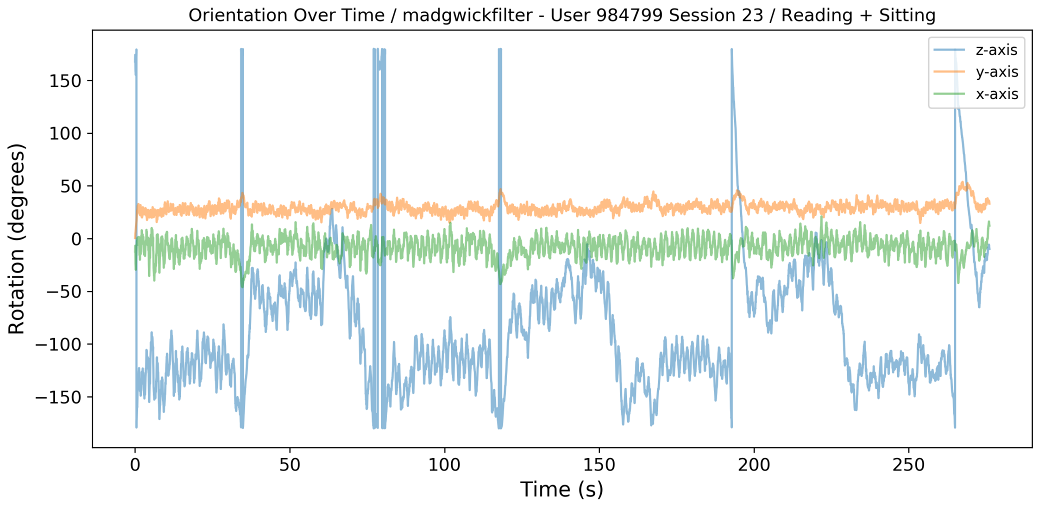


Figure 12: scikit kinematics AHRS Madgwick filter.

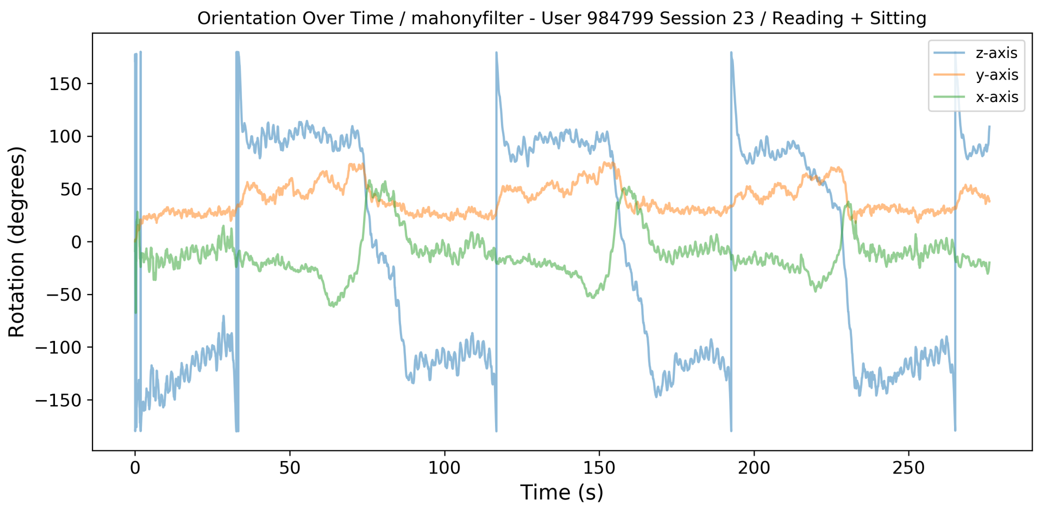


Figure 13: scikit kinematics AHRS Mahony filter.