Class Project
Using Inertial Sensors for Position and Orientation Estimation – HMOG Cell Phone Data

J. Kath – Math 178 Summer 2019

Topics

- General discussion of navigation, body frames and quaternions
- IMU sensors: accelerometer, gyroscope and vector magnetometer and errors
- General discussion of Kalman filter (algorithms 2 – 5 from paper)
- Intro to optimization / smoothing estimate math from paper (algorithm 1)
- Intro to EKF math from paper (algorithm 3)
- Results from running HMOG user accelerometer, gyroscope and magnetometer (3d) dataset using EKF algorithm implementation for orientation estimate with error, bias

Coordinate frames - navigation frame n, body frame b

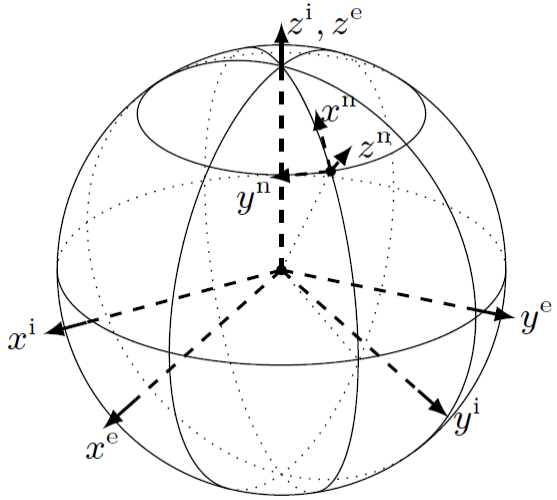
The body frame b is the coordinate frame of the moving IMU. Its origin is located in the center of the accelerometer triad and it is aligned to the casing. All the inertial measurements are resolved in this frame.

The navigation frame n is a local geographic frame in which we want to navigate. In other words, we are interested in the position and orientation of the b-frame with respect to this frame. For most applications it is defined stationary with respect to the earth. However, in cases when the sensor is expected to move over large distances, it is customary to move and rotate the n-frame along the surface of the earth. The first definition is used throughout this tutorial, unless mentioned explicitly.

The inertial frame i is a stationary frame. The IMU measures linear acceleration and angular velocity with respect to this frame. Its origin is located at the center of the earth and its axes are aligned with respect to the stars.

The earth frame e coincides with the i-frame, but rotates with the earth. That is, it has its origin at the center of the earth and axes which are fixed with respect to the earth.

Coordinate frames – rotation matrix



An illustration of three of the coordinate frames: the n-frame at a certain location on the earth, the e-frame rotating with the earth and the i-frame

The vector x expressed in the body frame b and expressed in the navigation frame n

$$x^n = R^{nb}x^b, \qquad x^b = (R^{nb})^{\mathsf{T}}x^n = R^{bn}x^n.$$

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*

Angular Velocity

$$\omega_{\mathrm{ib}}^{\mathrm{b}} = R^{\mathrm{bn}}\left(\omega_{\mathrm{ie}}^{\mathrm{n}} + \omega_{\mathrm{en}}^{\mathrm{n}}\right) + \omega_{\mathrm{nb}}^{\mathrm{b}},$$

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*

Specific Force

$$f^{\mathrm{b}} = R^{\mathrm{bn}}(a_{\mathrm{ii}}^{\mathrm{n}} - g^{\mathrm{n}}),$$

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

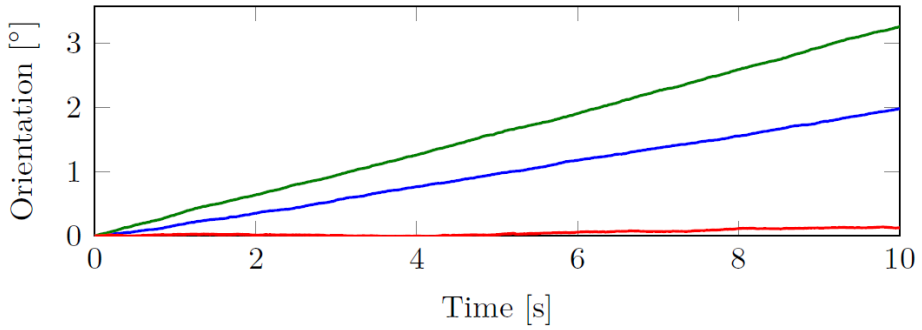
IMU sensor errors

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

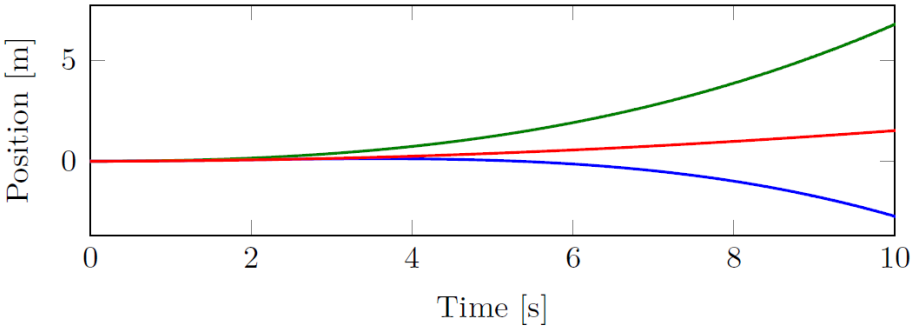
The process of integrating the measurements from inertial sensors to obtain position and orientation information, is called *deadreckoning*

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*

IMU sensor errors

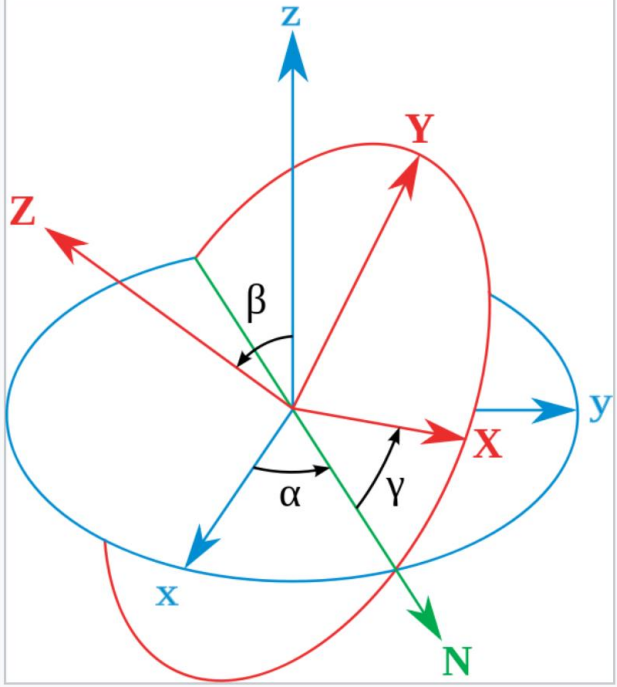


(a) Integrated orientation for the position in x- (blue), y- (green) and z-direction (red).



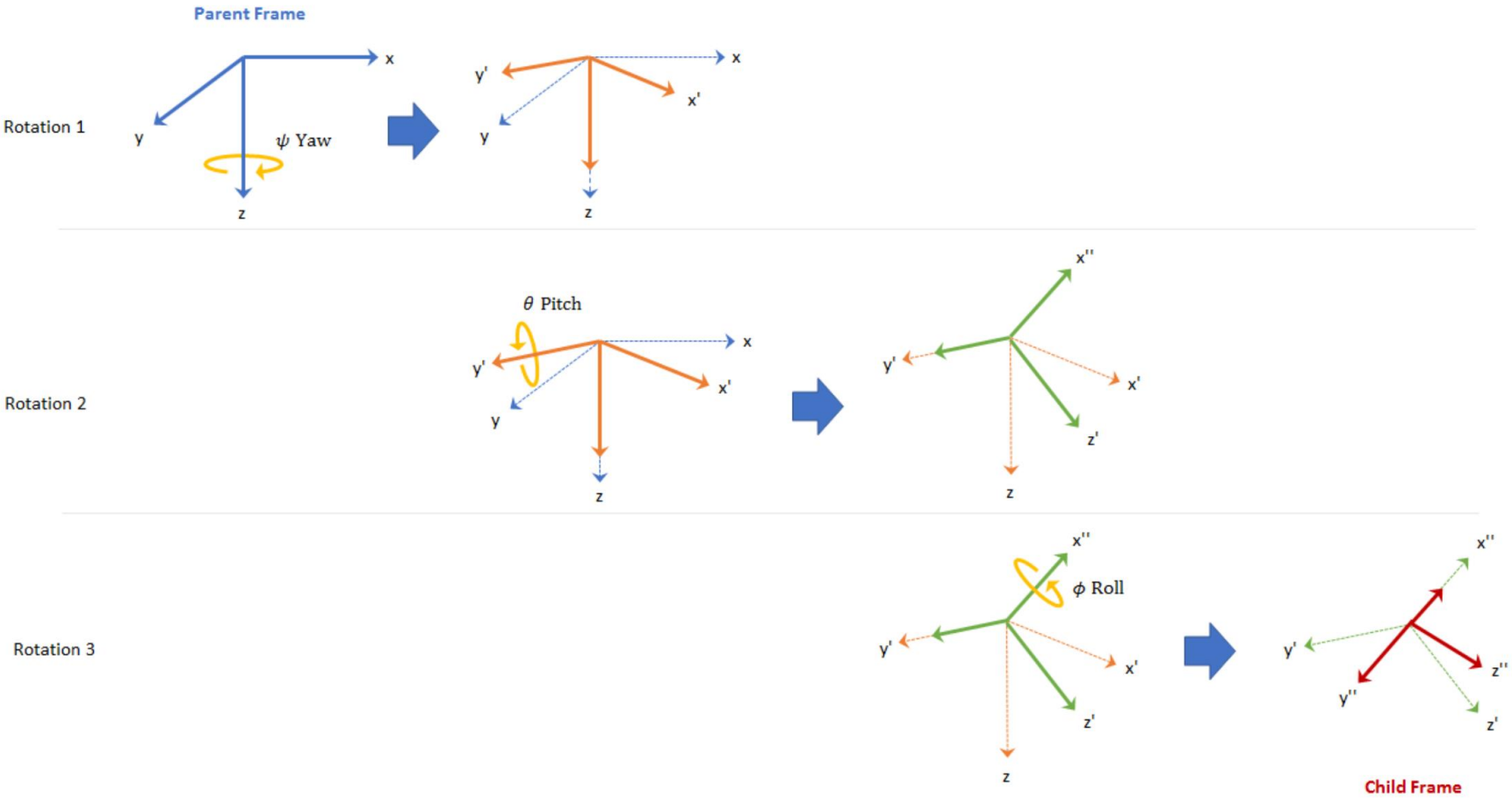
(b) Integrated position for rotation around the x-axis (blue), the y-axis (green) and the z-axis (red).

Orientation – Euler angles



Proper Euler angles geometrical definition. The *xyz* (fixed) system is shown in blue, the *XYZ* (rotated) system is shown in red. The line of nodes (*N*) is shown in green

Orientation – frame rotation



Orientation – rotation matrix

Order of rotations The standard ZYX order is used - first we rotate counter-clockwise around the *z axis*, then around the new *y axis* and finally around the new *x axis*

$$\begin{bmatrix} x' \\ y' \\ z \end{bmatrix} = \mathrm{R}_z(\psi) \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad \begin{bmatrix} x'' \\ y' \\ z' \end{bmatrix} = \mathrm{R}_y(\theta) \begin{bmatrix} x' \\ y' \\ z \end{bmatrix} \qquad \begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix} = \mathrm{R}_x(\phi) \begin{bmatrix} x'' \\ y' \\ z' \end{bmatrix}$$

$$\mathrm{R}(\psi,\theta,\phi) = \mathrm{R}_x(\psi)\mathrm{R}_y(\theta)\mathrm{R}_z(\phi) = \begin{bmatrix} \cos\psi\cos\theta & \sin\psi\cos\theta & -\sin\theta \\ \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \cos\theta\sin\phi \\ \cos\psi\sin\theta\cos\phi + \sin\psi\cos\phi & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \cos\theta\cos\phi \end{bmatrix}$$

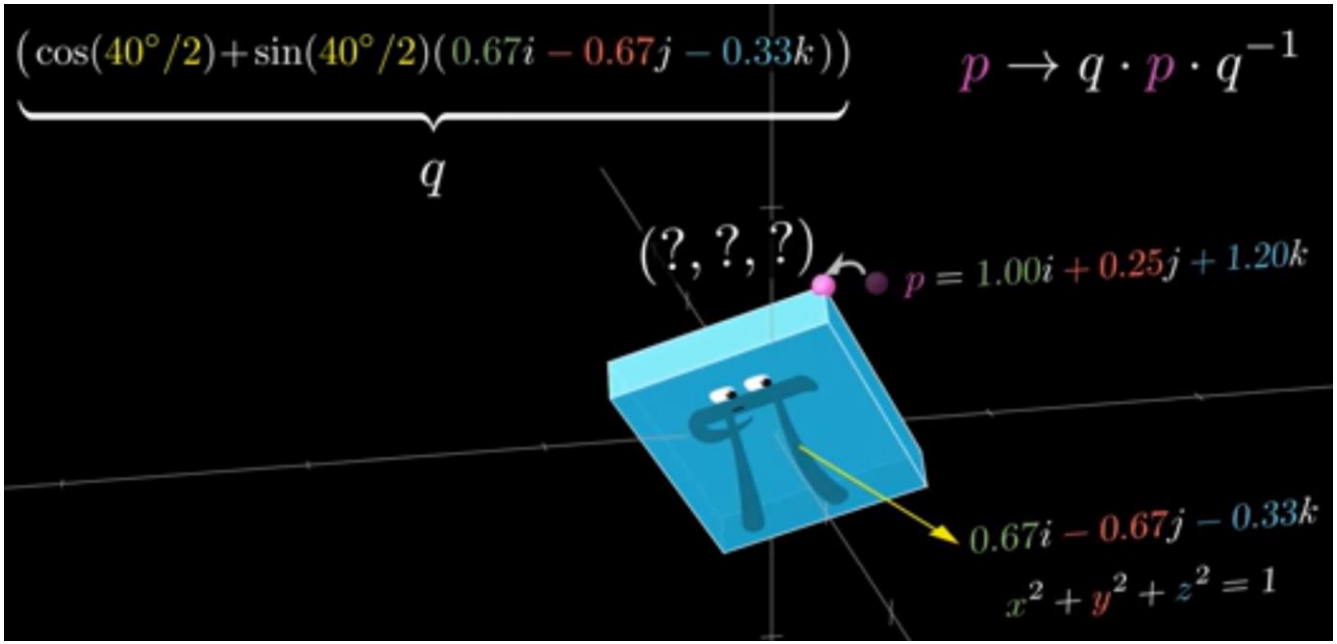
Attitude determination Our aim is to find the attitude of the device, equipped with sensors relative to some other pre-defined frame. We call the first frame “body” frame and latter “navigation” frame. We can define the attitude of the body *using two non-collinear vectors attached to it.*

Orientation – quaternions

Quaternions are generally represented in the form

$$a + b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k}$$

Point rotation of *p* about *axis* by 40 degrees

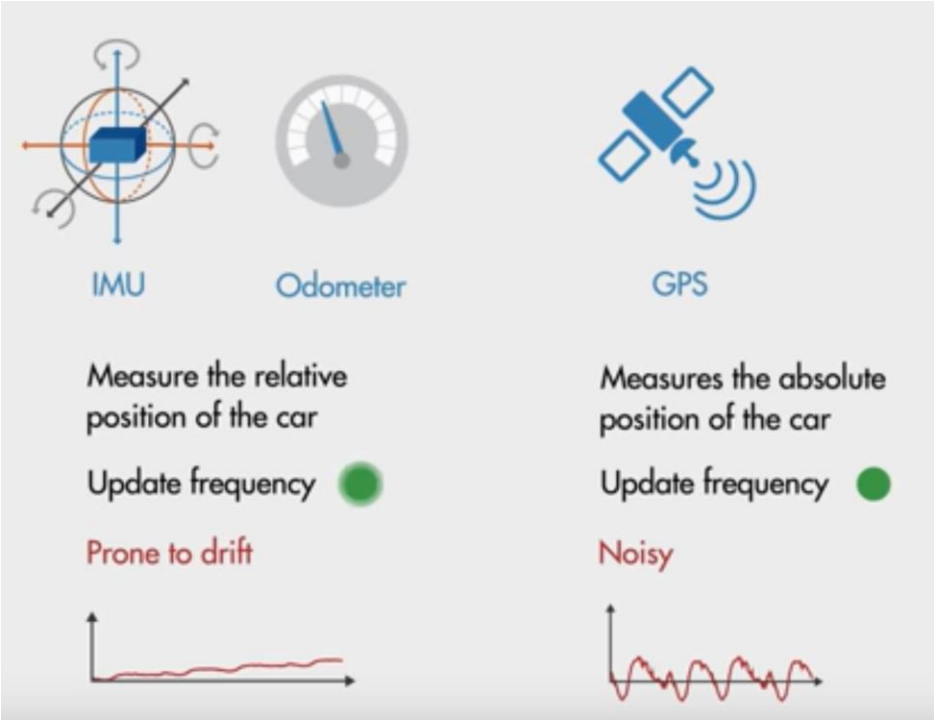


Quaternion multiplication

×	1	*i*	*j*	*k*
1	1	*i*	*j*	*k*
i	*i*	−1	*k*	−*j*
j	*j*	−*k*	−1	*i*
k	*k*	*j*	−*i*	−1

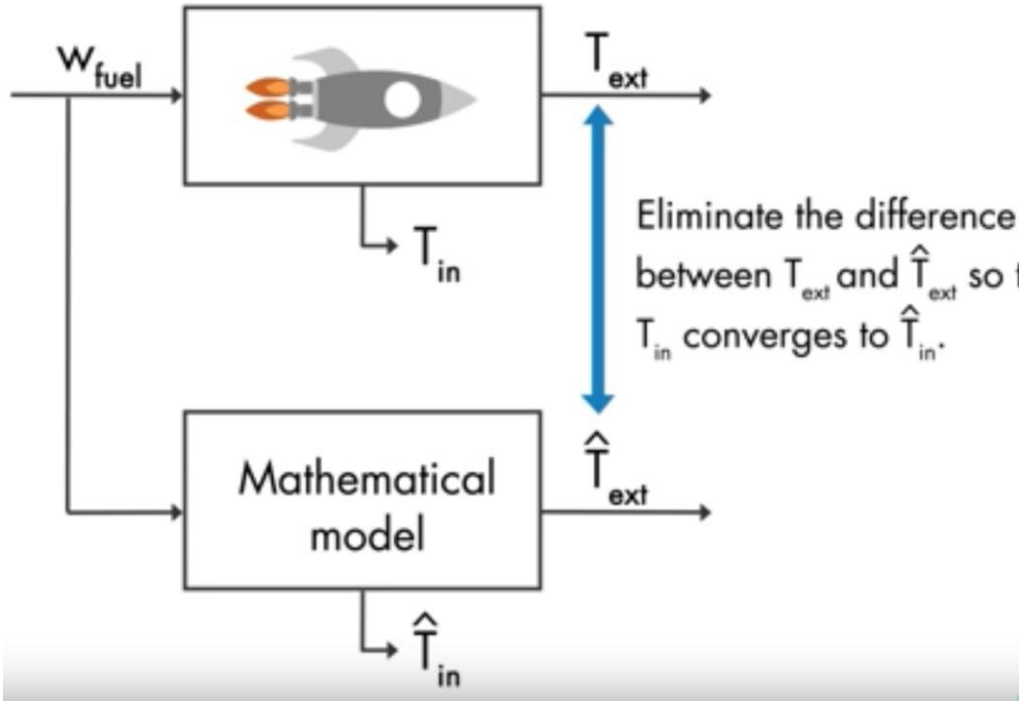
$$ij = -ji = k$$
$$jk = -kj = i$$
$$ki = -ik = j,$$

Kalman filter



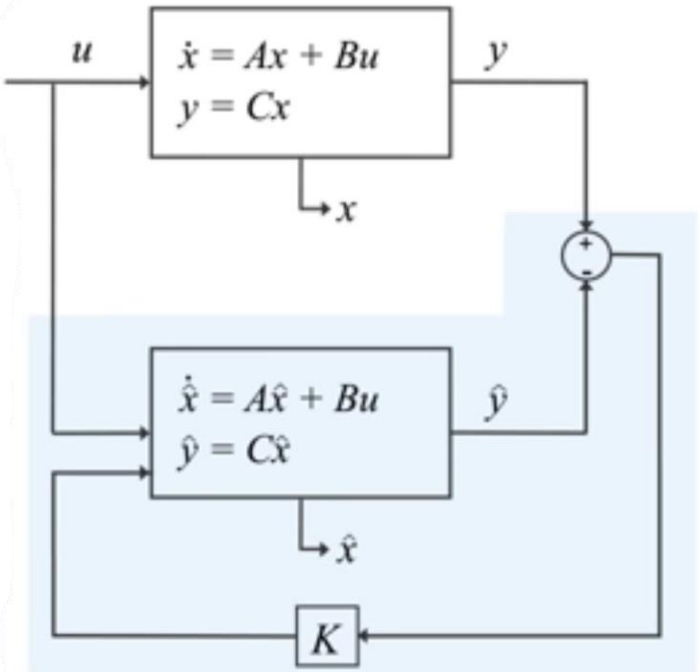
Kalman filter can be used to fuse these three measurements to find the optimal position of the car

Kalman filter – state observers



We are trying to minimize the difference between the estimated and measured external temperatures

Kalman filter – state observers



$$e_{obs} = x - \hat{x}$$

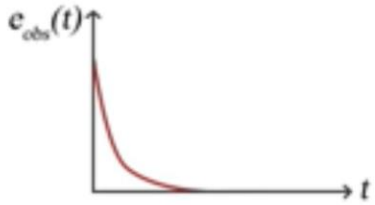
$$\dot{x} = Ax + Bu \qquad y = Cx$$

$$\dot{\hat{x}} = A\hat{x} + Bu + K(y - \hat{y}) \qquad \hat{y} = C\hat{x}$$

$$\dot{e}_{obs} = (A - KC)e_{obs} \qquad y - \hat{y} = Ce_{obs}$$

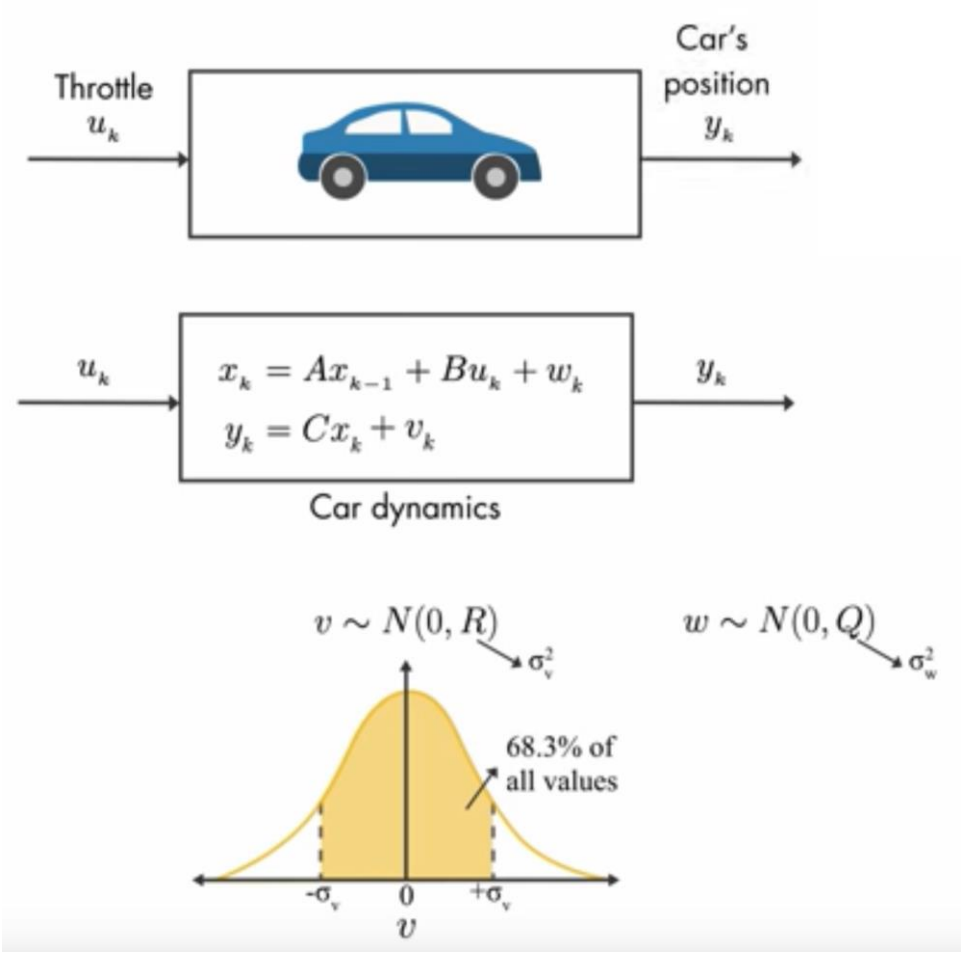
$$\hookrightarrow e_{obs}(t) = e^{(A-KC)t}e_{obs}(0)$$

If $(A\text{-}KC) < 0$, then $e_{obs} \to 0$ as $t \to \infty$. So, $\hat{x} \to x$.

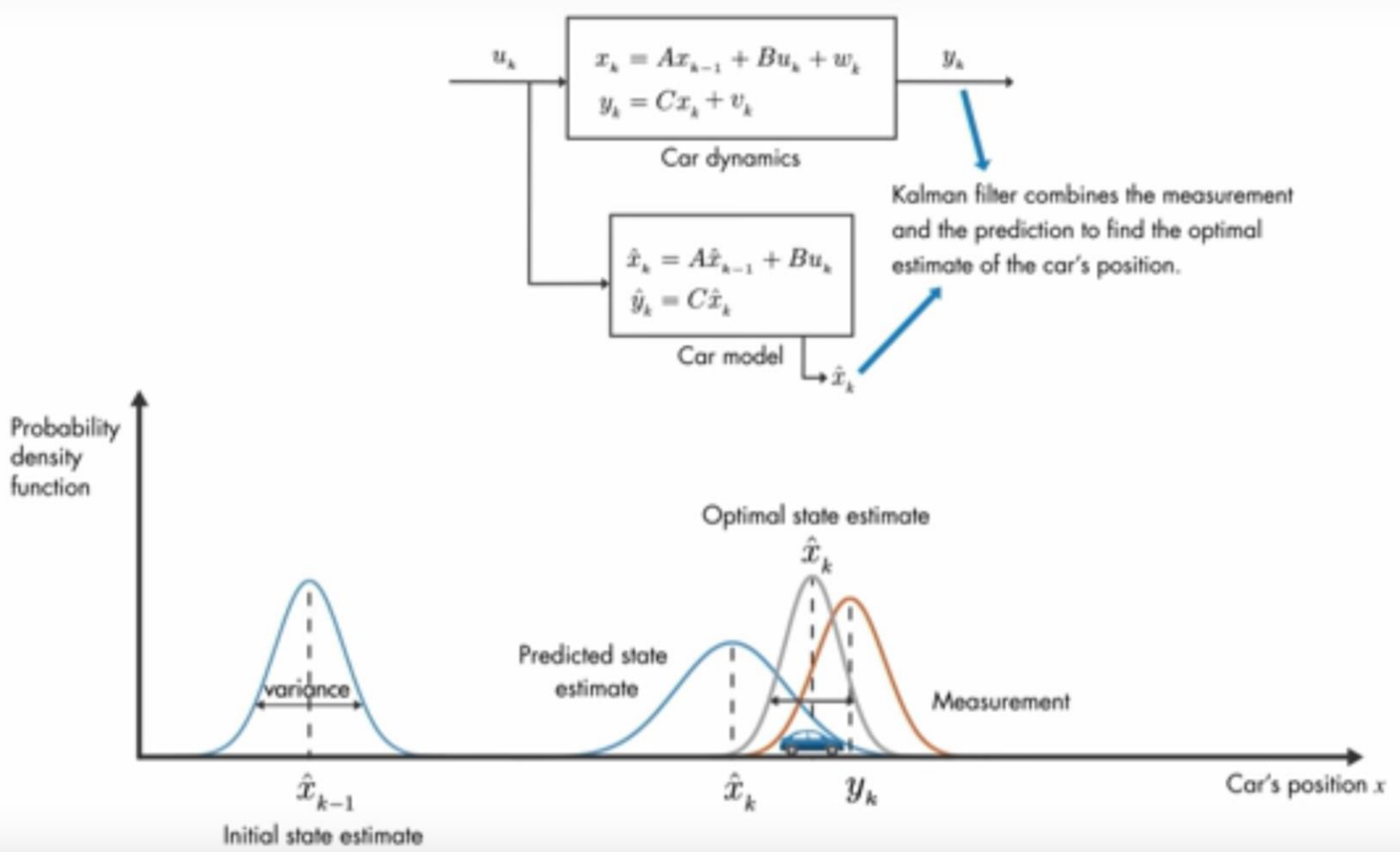


State observer a feedback control system with controller gain *K*

Kalman filter – optimal state estimator



Kalman filter – optimal state estimator



Kalman filter – optimal state estimator algorithm

State observer $\hat{x}_{k+1} = A\hat{x}_k + Bu_k + K(y_k - C\hat{x}_k)$ Deterministic system

Kalman filter $\hat{x}_k = \underbrace{A\hat{x}_{k-1} + Bu_k}_{\hat{x}_k^-} + K_k(y_k - C(A\hat{x}_{k-1} + Bu_k))$ Stochastic system

$\hat{x}_k^-$: A Priori Estimate

A Posteriori Estimate

$\hat{\hat{x}}_k = \underbrace{\hat{x}_k^-}_{\text{Predict}} + \underbrace{K_k(y_k - C\hat{x}_k^-)}_{\text{Update}}$

Prediction

$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$

$P_k^- = AP_{k-1}A^T + Q$

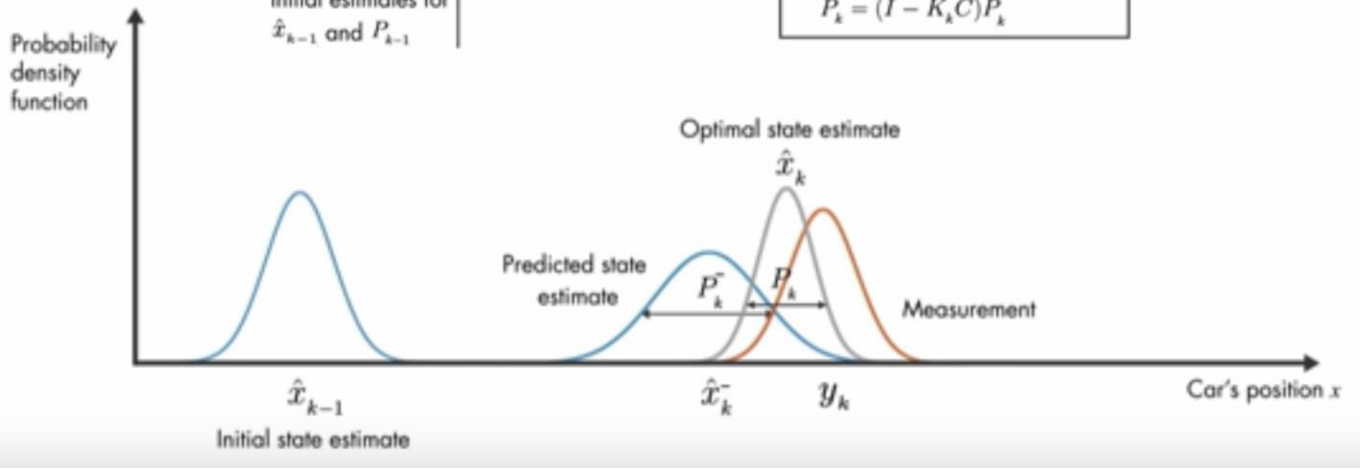
Initial estimates for $\hat{x}_{k-1}$ and P_{k-1}

Update

$K_k = \frac{P_k^- C^T}{CP_k^- C^T + R}$

$\hat{x}_k = \hat{x}_k^- + K_k(y_k - C\hat{x}_k^-)$

$P_k = (I - K_k C)P_k^-$



Probabilistic models – measurement

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 $\omega_{\mathrm{ib}}^{\mathrm{b}}$ at each time instance t. Our simplified measurement model is

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

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 $y_{\mathrm{m},t}$ can be modeled as

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

Probabilistic models – smoothing vs filtering

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 computation intensive and assumes all measurement data is available for our estimate. The other class of estimation we will model is known as *filtering*. In filtering we estimate x using all measurements up to and including time t

Probabilistic models – estimation

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,

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

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

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

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

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

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

Gauss-Newton optimization

Smoothing in an optimization framework

Perhaps the most intuitive way to solve the smoothing problem is by posing it as an optimization problem, where a *maximum a posteriori (MAP)* estimate is obtained as

$$\begin{aligned}\hat{x}_{1:N} &= \arg\max_{x_{1:N}} p(x_{1:N} \mid y_{1:N}) \\ &= \arg\max_{x_{1:N}} p(x_1) \prod_{t=2}^{N} p(x_t \mid x_{t-1}) p(y_t \mid x_t). \end{aligned} \tag{4.1}$$

Here, we use the notation in terms of probability distributions as introduced in §3.1 and model the measurements and the state dynamics as described in §3.4 and §3.5, respectively. Furthermore, we assume that a prior on the initial state is obtained using the measurements at $t = 1$ as described in §3.6. Because of this, the measurement model $p(y_1 \mid x_1)$ from (3.3a) is explicitly omitted in (4.1). Note that in practice, we typically minimize $-\log p(x_{1:N} \mid y_{1:N})$ instead of maximizing $p(x_{1:N} \mid y_{1:N})$ itself, resulting in the optimization problem

$$\arg\min_{x_{1:N}} -\log p(x_1) - \sum_{t=2}^{N} \log p(x_t \mid x_{t-1}) - \sum_{t=2}^{N} \log p(y_t \mid x_t). \tag{4.2}$$

Gauss-Newton optimization

Smoothing in an optimization framework

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*.

Gauss-Newton optimization

Smoothing in an optimization framework

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

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

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

Gauss-Newton optimization

Smoothing in an optimization framework

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

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

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

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

Algorithm 1 - Smoothing estimates of the orientation using optimization

Algorithm 1 Smoothing estimates of the orientation using optimization

INPUTS: An initial estimate of the orientation $\tilde{q}^{\mathrm{nb},(0)}_{1:N}$, 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 $\hat{q}^{\mathrm{nb}}_{1:N}$ and optionally its covariance $\mathrm{cov}(\hat{\eta}^{\mathrm{n}}_{1:N})$.

1. Set $\hat{\eta}^{\mathrm{n},(0)}_t = 0_{3\times 1}$ for $t = 1, \ldots, N$, set $k = 0$ and compute $\breve{q}^{\mathrm{nb}}_1$ and $\Sigma_{\eta,\mathrm{i}}$.
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 $\hat{\eta}^{\mathrm{n},(k+1)}_{1:N}$.
   (c) Update the linearization point as

$$\tilde{q}^{\mathrm{nb},(k+1)}_t = \exp_{\mathrm{q}}\left(\frac{\hat{\eta}^{\mathrm{n},(k+1)}_t}{2}\right) \odot \tilde{q}^{\mathrm{nb},(k)}_t, \tag{14}$$

   and set $\hat{\eta}^{\mathrm{n},(k+1)}_t = 0_{3\times 1}$ for $t = 1, \ldots, N$.
   (d) Set $k = k + 1$.
   end while
3. Set $\hat{q}^{\mathrm{nb}}_{1:N} = \tilde{q}^{\mathrm{nb},(k)}_{1:N}$.
4. Optionally compute

$$\mathrm{cov}(\hat{\eta}^{\mathrm{n}}_{1:N}) = \left(\mathcal{J}^{\mathsf{T}}(\hat{\eta}^{\mathrm{n}}_{1:N})\mathcal{J}(\hat{\eta}^{\mathrm{n}}_{1:N})\right)^{-1}. \tag{15}$$

Algorithm 3 - Extended Kalman Filtering

Algorithm 3 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 Σ_i as described in §3.6 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

$$\hat{q}^{\mathrm{nb}}_{t|t-1} = \hat{q}^{\mathrm{nb}}_{t-1|t-1} \odot \exp_\mathsf{q}\left(\tfrac{T}{2} y_{\omega,t-1}\right), \tag{4.45a}$$

$$P_{t|t-1} = F_{t-1} P_{t-1|t-1} F_{t-1}^\mathsf{T} + G_{t-1} Q G_{t-1}^\mathsf{T}, \tag{4.45b}$$

with $Q = \Sigma_\omega$ and

$$F_{t-1} = \left(\exp_\mathsf{q}(\tfrac{T}{2} y_{\omega,t-1})\right)^\mathsf{R}, \; G_{t-1} = -\tfrac{T}{2} \left(\hat{q}^{\mathrm{nb}}_{t-1|t-1}\right)^\mathsf{L} \tfrac{\partial \exp_\mathsf{q}(e_{\omega,t-1})}{\partial e_{\omega,t-1}}.$$

Algorithm 3 - Extended Kalman Filtering

Algorithm 3 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 $\hat{q}_{t|t}^{\mathrm{nb}}$ and its covariance $P_{t|t}$ for $t = 1, \dots N$.

2. **For** $t = 2, \dots, N$ **do**

(b) Measurement update

$$\tilde{q}_{t|t}^{\mathrm{nb}} = \hat{q}_{t|t-1}^{\mathrm{nb}} + K_t \varepsilon_t, \tag{4.46a}$$
$$\tilde{P}_{t|t} = P_{t|t-1} - K_t S_t K_t^{\mathsf{T}}, \tag{4.46b}$$

with ε_t, K_t and S_t defined in (4.38) and

$$y_t = \begin{pmatrix} y_{\mathrm{a},t} \\ y_{\mathrm{m},t} \end{pmatrix}, \qquad \hat{y}_{t|t-1} = \begin{pmatrix} -\hat{R}_{t|t-1}^{\mathrm{bn}} g^{\mathrm{n}} \\ \hat{R}_{t|t-1}^{\mathrm{bn}} m^{\mathrm{n}} \end{pmatrix},$$

$$H_t = \begin{pmatrix} -\left.\frac{\partial R_{t|t-1}^{\mathrm{bn}}}{\partial q_{t|t-1}^{\mathrm{nb}}}\right|_{q_{t|t-1}^{\mathrm{nb}} = \hat{q}_{t|t-1}^{\mathrm{nb}}} g^{\mathrm{n}} \\ \left.\frac{\partial R_{t|t-1}^{\mathrm{bn}}}{\partial q_{t|t-1}^{\mathrm{nb}}}\right|_{q_{t|t-1}^{\mathrm{nb}} = \hat{q}_{t|t-1}^{\mathrm{nb}}} m^{\mathrm{n}} \end{pmatrix}, \qquad R = \begin{pmatrix} \Sigma_\mathrm{a} & 0 \\ 0 & \Sigma_\mathrm{m} \end{pmatrix}.$$

(c) Renormalize the quaternion and its covariance as

$$\hat{q}_{t|t}^{\mathrm{nb}} = \frac{\tilde{q}_{t|t}^{\mathrm{nb}}}{\|\tilde{q}_{t|t}^{\mathrm{nb}}\|_2}, \qquad P_{t|t} = J_t \tilde{P}_{t|t} J_t^{\mathsf{T}}, \tag{4.47}$$

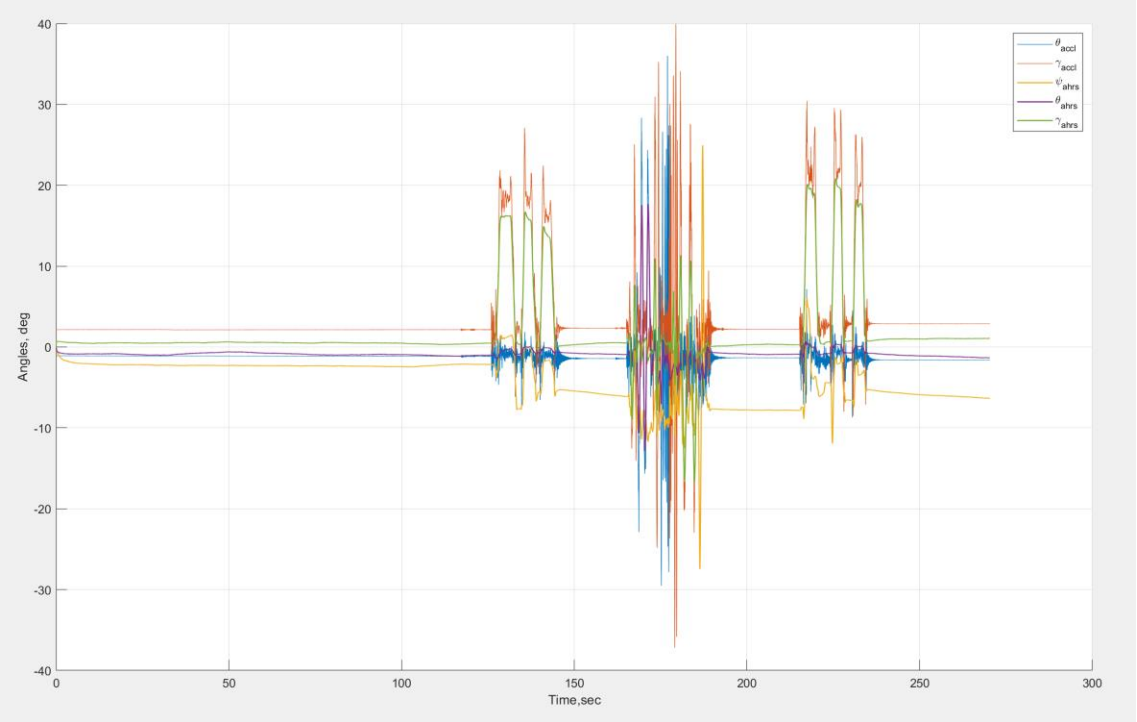
with $J_t = \frac{1}{\|\tilde{q}_{t|t}^{\mathrm{nb}}\|_2^3} \tilde{q}_{t|t}^{\mathrm{nb}} \left(\tilde{q}_{t|t}^{\mathrm{nb}}\right)^{\mathsf{T}}$.

end for

Orientation estimation results using reference data – EKF algorithm

MATLAB *GyroLib* reference data – microelectromechanical system (MEMS) IMU module: Fairchild's FMT 1010

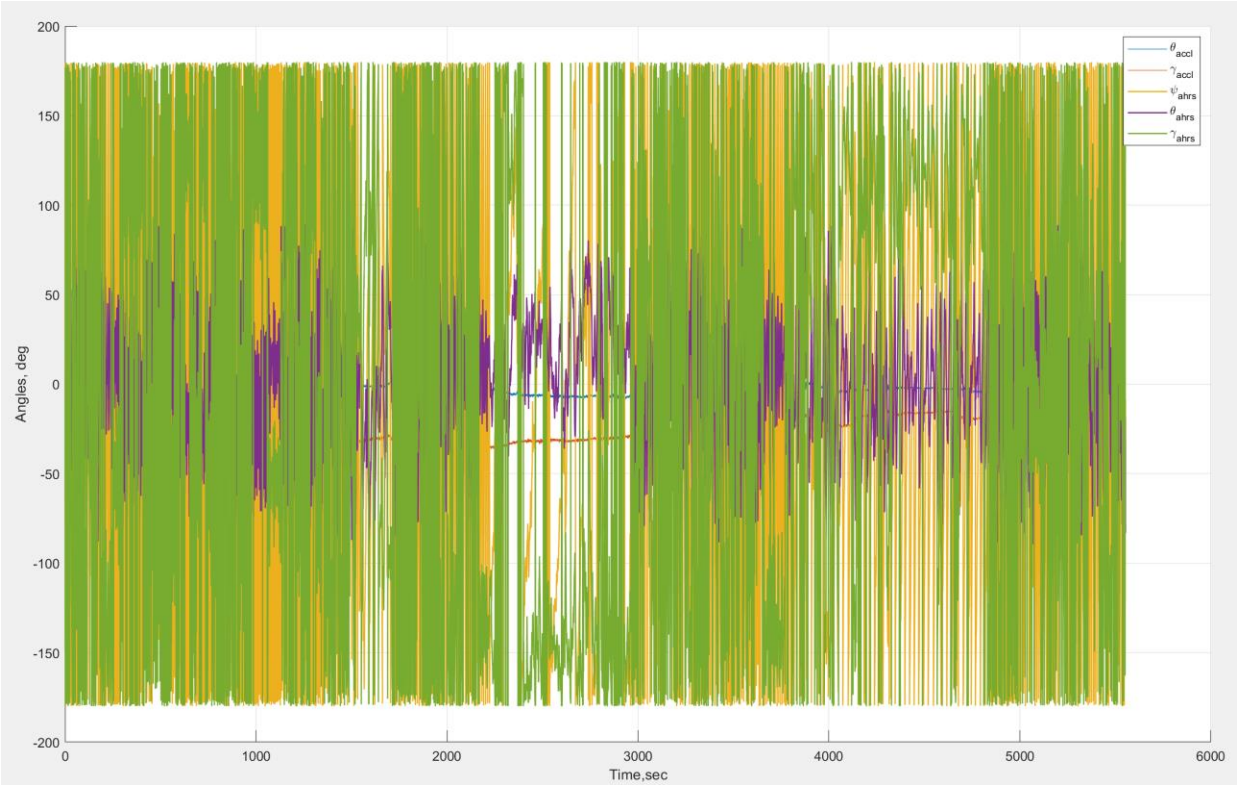
Orientation – Euler angles



Orientation estimation results using HMOG data – EKF algorithm

HMOG user data – user Id 151985 session 2 / Reading + Sitting

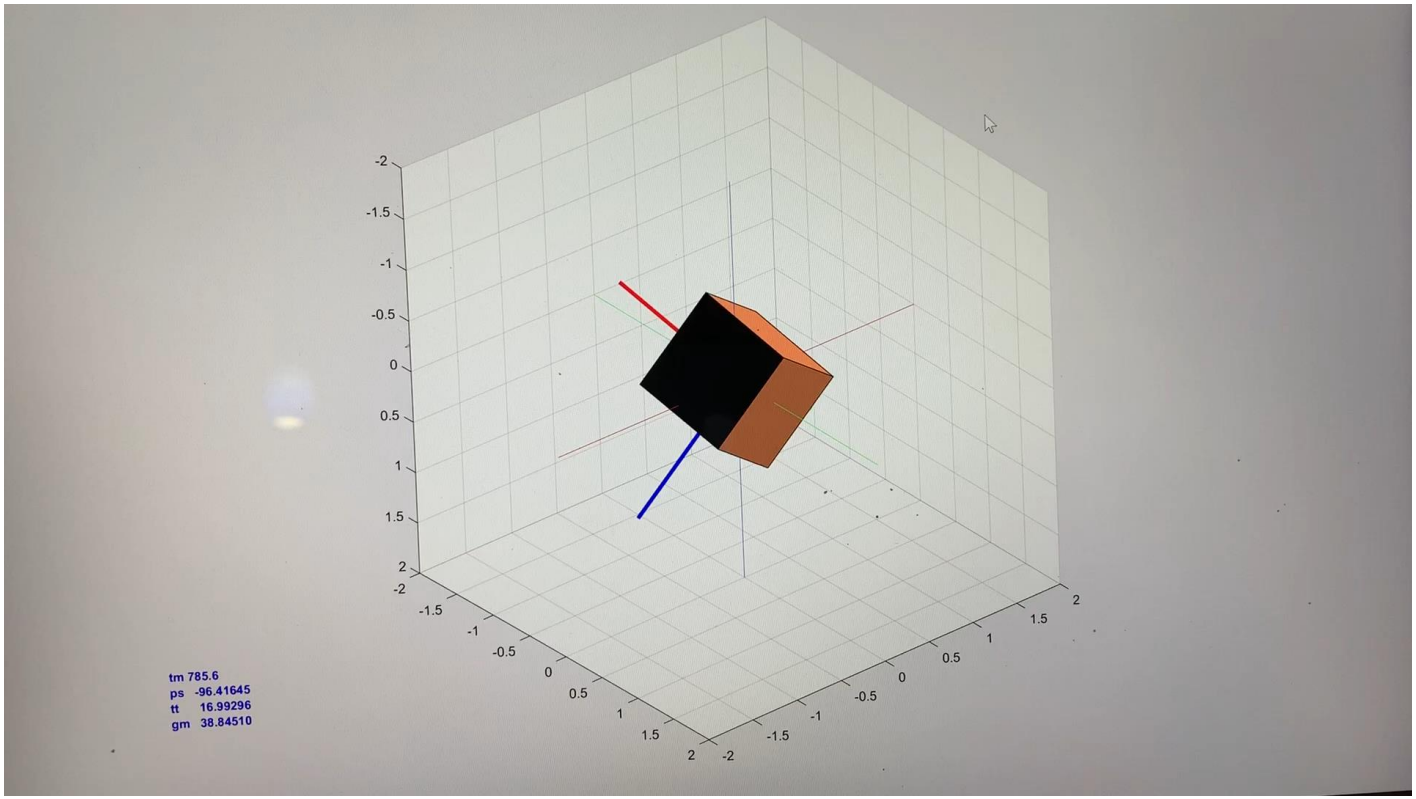
Orientation – Euler angles



Orientation estimation results using HMOG data – EKF algorithm

HMOG user data – user Id 151985 session 2 / Reading + Sitting

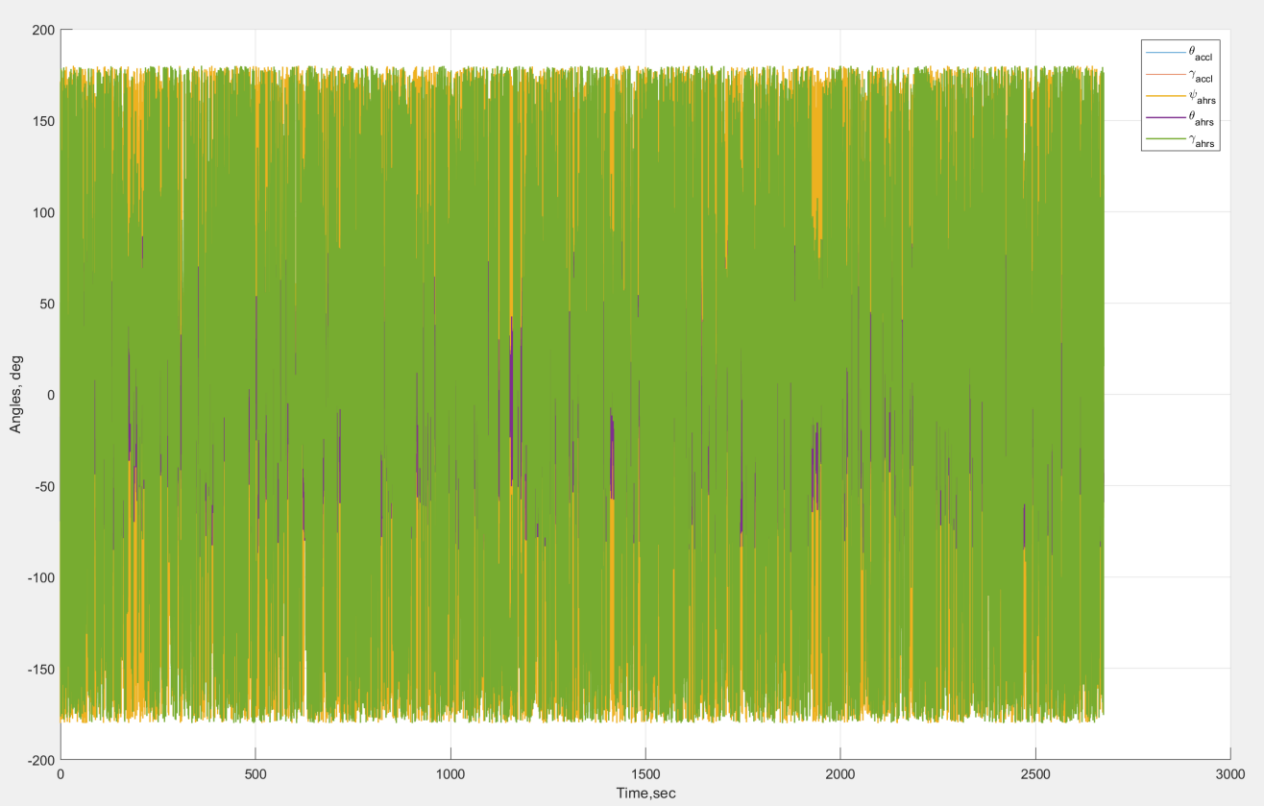
Orientation – Euler angles



Orientation estimation results using HMOG data – EKF algorithm

HMOG user data – user Id 984799 session 16 / Reading + Walking

Orientation – Euler angles



References

1. Manon Kok, Jeroen D. Hol and Thomas B. Schon (2017), Using Inertial Sensors for Position and Orientation Estimation
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
3. Matlab IMU Toolbox Documentation
4. Matlab YouTube Channel – Kalman Filters
5. https://eater.net/quaternions/ - Visualizing Quaternions