Data fusion, estimation and sensor calibration

Spring 2015 – Lecture #13
Multisensor systems

- Sensor 1
- Sensor 2
- Sensor n

Data processing algorithm (Estimator)

States
- Position
- Velocity
- Acceleration
- Attitude/orientation

Parameters
- Sensor errors, such as bias/offset

Can be implemented in real-time on an embedded system, or as part of post-processing of sensor data on a PC
Two-sensor data fusion example

- Both sensors take a measurement \( z \) of a constant but unknown parameter \( x \), in the presence of noise \( v \) with standard deviation \( \sigma \)
  - \( z_1 = x + v_1 \) and \( z_2 = x + v_2 \)

Question: How to combine the two measurements to produce an optimal estimate of \( \hat{x} \) of the unknown parameter \( x \)?

Answer:

- \( \hat{x} = k_1 z_1 + (1 - k_1) z_2 \) (the estimate is a linear combination of the measurements)

- \( \hat{x} = \left( \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2} \right) z_1 + \left( \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \right) z_2 \)

- Check: What happens if \( \sigma_1^2 = \sigma_2^2 \), or if \( \sigma_1 \) or \( \sigma_2 \) is equal to zero?
Example: Tilt-compensated compass

The Horizontal component of the Earth’s magnetic field is used for compassing, since it points to North.

1) Rotate measurements to a common phone frame (NED):

\[ G_p = R_x(\phi)R_y(\theta)R_z(\psi)G_r \quad B_p = R_x(\phi)R_y(\theta)R_z(\psi)B_r \]

2) Calculate the roll and pitch angles \( \varphi \) and \( \theta \) from the accelerometer.

3) De-rotate magnetometer readings (level the sensor) to correct for phone orientation:

\[ B_{\text{corrected}} = R_y(-\theta)R_x(-\phi)B_p \]
GNC: Unmanned Aircraft System (UAS)

GNC: Guidance, Navigation and Control

OBC: Onboard computer

Illustration 5: Waypoints for flight over Seattle
Examples of coordinated measurements

4DSpace
Sounding rocket with daughter payloads

Coordinated multi-point measurements
Temporal and spatial information

4 Cluster satellites
Estimation

- In dynamic systems (systems which vary with time) the variables are called **states**.

- **Parameters** are variables that do not vary with time.

- Sometimes the states/parameters of a system are not or cannot be measured directly.

- Any measurements are corrupted by noise and other sensor errors.

- In addition to finding and estimate of the unknown parameters we also want to estimate the uncertainty in our estimate.
Estimator

• An estimator is a data processing algorithm.

• Often a need to combine measurements from different sensors with different data sample rates and accuracies.

• An optimal estimator combines all the available information (about the system and sensors).

• The estimator is optimal when it minimizes the estimation error in a well-defined statistical sense, based on a criterion of optimality.
Standard deviation

\[ \sigma = s = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})^2} \]

The standard deviation is the amount of variation from the mean.
Covariance

- Covariance provides a measure of the strength of the correlation between two or more sets of random variables
- For uncorrelated variables: $\text{cov}(X,Y) = 0$
- $\text{cov}(X,X) = \text{var}(X) = \sigma^2$
- Covariance matrix:
  - A covariance matrix is a matrix whose element in the $i, j$ position is the covariance between the $i^{\text{th}}$ and $j^{\text{th}}$ elements of a random vector
  - The diagonal elements of the covariance matrix is the variances
  - The off-diagonal elements represent the covariances
Multi-sensor fusion

• Gives reduced uncertainty!
• Makes the system more robust!
Reference frame for multi-sensor fusion

- Data measured in different coordinate frames S1 and S3
- Before data fusion all vector measurements must be transformed into a common coordinate system M
An inertial reference frame is a coordinate system that does not rotate, and is either fixed in three-dimensional space or moves in a straight line at constant velocity (with zero acceleration).

Newton's laws are valid only in an inertial reference frame; remember $F = ma$. 

<table>
<thead>
<tr>
<th>Frame symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I$</td>
<td>Inertial</td>
</tr>
<tr>
<td>$E$</td>
<td>Earth-fixed</td>
</tr>
<tr>
<td>$B$</td>
<td>Body-fixed</td>
</tr>
<tr>
<td>$N$</td>
<td>North-East-Down (local level)</td>
</tr>
</tbody>
</table>

Figure: Gade (2008)
Alignment

- Mounting errors (internal and external) are present when multiple sensors are installed in a vehicle.
- Sensors can also be mounted in different directions with purpose.
- A calibration to re-align the sensors in software can be required
Synchronization

- Combining data (data fusion) from several sensors/instruments usually give a lower measurement uncertainty (and increased system robustness). However, such a data fusion requires the data to be **synchronized in time**.
- If simultaneous sampling of the sensors is not possible, which is often the case when multiples standalone instruments are used, time stamping of the data can be used.
- If every data set or data sample are tagged with an accurate time the different data samples can be aligned in time, for instance using interpolation.
- The GPS system provides easy access to a very accurate time base globally. GPS time is the atomic time scale implemented by the atomic clocks in the GPS ground control stations and the GPS satellites themselves.
System of linear equations I

- A general system of \( m \) linear equations with \( n \) unknowns \( x_i \) can be written as:

\[
\begin{align*}
  a_{11} x_1 + a_{12} x_2 + \cdots + a_{1n} x_n &= b_1 \\
  a_{21} x_1 + a_{22} x_2 + \cdots + a_{2n} x_n &= b_2 \\
  \vdots & \quad \vdots \quad \vdots \\
  a_{m1} x_1 + a_{m2} x_2 + \cdots + a_{mn} x_n &= b_m.
\end{align*}
\]

- This can be written as a matrix equation of the form:

\[
A x = b
\]

where

\[
A = \begin{bmatrix}
  a_{11} & a_{12} & \cdots & a_{1n} \\
  a_{21} & a_{22} & \cdots & a_{2n} \\
  \vdots & \vdots & \ddots & \vdots \\
  a_{m1} & a_{m2} & \cdots & a_{mn}
\end{bmatrix}, \quad
x = \begin{bmatrix}
  x_1 \\
  x_2 \\
  \vdots \\
  x_n
\end{bmatrix}, \quad
b = \begin{bmatrix}
  b_1 \\
  b_2 \\
  \vdots \\
  b_m
\end{bmatrix}
\]
System of linear equations II

- If the number of measurements (number of equations) is equal to the number of unknowns \( x_i \) \((m = n)\), the unknowns can be found from the inverse solution:
  \[ x = A^{-1}b \]  
  (In Matlab: \( x = A\backslash b \), or \( x = \text{inv}(A)*b \))

- In the more common case, there are more measurements (equations) than unknown \((m > n)\). This is called an over determined systems. Then, a **Least squares** method can be used to estimate the unknown parameters.
Batch vs. recursive estimator

- Batch processing:
  - All available measurements are processed at one time

- Recursive processing:
  - Measurements are processed as they become available
    - Required computer storage is kept at a minimum

\[ z(\omega) = a \cdot \omega + b \]
Linear Least-Squares (LS) Estimation

Model (static system) - The Measurement equation:

\[ z_k = H_k x_k + v_k \]

LS Cost function to be minimized:

\[ J = \frac{1}{2} \sum_{k=0}^{N} (z_k - H\hat{x}_k)^T R^{-1} (z_k - H\hat{x}_k) \]

\[ x = [x_1 \ x_2 \ldots \ x_n]^T \]

Explicit Solution for the optimal estimate:

\[ \hat{x} = (H^T R^{-1} H)^{-1} H^T R^{-1} z \]

\[ P = (H^T R^{-1} H)^{-1} \]

This is a batch estimator - all the measurements are processed at the same time. This means that the dimension of the estimation problem increases for each new measurement sample.

Memory hungry!
1D model example – curve fitting

- Measured gyro rotation rate $z$ (in voltage) as a function of applied rate table rotation rate $\omega$ (in deg/sec).

1D Model:

$$z(\omega) = a \cdot \omega + b$$

Model as a matrix equation:

$$z = H \mathbf{x}$$

$$H = \begin{bmatrix} \omega & 1 \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} a \\ b \end{bmatrix}$$

Simplest case: $R = I$ (identity matrix) → $R$ falls out of the LS-equations
Weighted Least Squares

• The traditional least squares solution places equal emphasis on each measurement.

• However, measurements are often made with unequal precision (e.g. due to different sensor accuracies). Therefore, we want to add a weight such that the more precise measurements are given more importance.

\[
\hat{x} = (H^T R^{-1} H)^{-1} H^T R^{-1} z
\]

\[
P = (H^T R^{-1} H)^{-1}
\]

• The R matrix (weighting matrix) can be selected as a diagonal matrix with the variance of the sensor measurements on the diagonal.

• If the R matrix is selected to be an identity matrix (similar to not include the R matrix in the equations), all measurements are weighted equally.

• R is the measurement error covariance matrix.
The recursive LS estimator

- Batch algorithms are not suitable for realtime applications.
- A recursive estimator, on the other hand, processes the measurements sequentially as they become available, and stores only the value of the last state (“Parameter” is used for time-invariant (or quasi-static) variables, while "state" is used for time-varying (dynamic) variables)
- The recursive least squares estimator:

\[
\hat{x}(k+1) = \hat{x}(k) + W(k+1)[z(k+1) - H(k+1)\hat{x}(k)]
\]

The new estimate \(\hat{x}\) at time \((k+1)\) is equal to the estimate at time \((k)\) plus a correction term. The correction term consists of a gain \(W(k+1)\) that multiplies the difference between the measurement \(z(k+1)\) and the predicted value of this measurement given by \(H(k+1)\hat{x}(k)\)
Filtering

• In estimation the term **filtering** refers to estimating parameters (the state vector) describing the system at the current time, based on all past measurements. So, filtering in estimation theory includes more than just filtering noise, such as a low pass filter.

• Offline (non RT) processing makes it possible to obtain more accurate estimates. A **smoother** produces improved estimates by making use of data both before and after any given time point of interest.
Kalman filter (KF) I

- The Kalman filter, introduced by Kalman (1960), has been one of the most widely used estimation algorithms.
- The Kalman filter is a recursive estimator.
- The Kalman filter is the optimal minimum mean square error (MMSE) estimator for linear, Gaussian systems.
- MMSE one possible (and very often used) optimization criteria.
- “Gives a best fit (to observed measurements) in a statistical sense.”
Kalman filter (KF) II

- Two sorts of information are utilized:
  - *Measurements* from relevant sensors
  - *Mathematical model* of the system (describing how the different states depend on each other, and how the measurements depend on the states)

- In addition the *accuracy* of the measurements and the model must be specified.
Kalman filter III

- The Kalman filter produces estimates of the true values of measurements by predicting a value, estimating the uncertainty of the predicted value, and computing a weighted average of the predicted value and the measured value. The most weight is given to the value with the least uncertainty. The estimates produced by the method tend to be closer to the true values than the original measurements because the weighted average has a better estimated uncertainty than either of the values that went into the weighted average. From Wikipedia
The discrete state space representation of a linear system is given by:

\[ x_{k+1} = \Phi_k x_k + \Lambda_k u_k + \Gamma_k w_k \quad (6.1) \]
\[ z_k = H_k x_k + v_k \quad (6.2) \]

Equation (6.1) is called the system dynamics equation or process equation. \( \Phi_k \) is called the state transition matrix, and relates the state at the previous time step \( k \) to the state at the current step \( k + 1 \). \( \Lambda \) is the control input matrix, \( u \) is a deterministic control input, \( \Gamma \) is the process noise matrix and \( w \) is the process noise, assumed to be zero-mean, white (uncorrelated), and with a Gaussian probability distribution. The index on the matrices \( \Phi \), \( \Lambda \), \( \Gamma \) and \( H \) indicates that they might be time-variant.

Equation (6.2) is called the measurement model. \( z \) is the measurement vector, \( H \) is the measurement matrix and \( v \) is the measurement noise, assumed to be zero-mean, Gaussian white noise. The process noise and the measurement noise are also assumed to be independent/uncorrelated.
Kalman filter

Model (dynamic system):

\[
\begin{align*}
x_{k+1} &= \Phi_k x_k + \Lambda_k u_k + \Gamma_k w_k \quad \text{Eq. 1} \\
z_k &= H_k x_k + v_k \quad \text{Eq. 2}
\end{align*}
\]

- Eq. 1: a mathematical model of how the states \( x \) varies with time; called the Process Model
- Eq. 2: Measurement model
- Only simple statistical information needed: mean and standard deviation.
- LS vs. Kalman filter
  - LS used for nonrandom (deterministic), time-invariant parameters.
  - The Kalman filter is used for random parameters (which can be time varying).

KF equations (in discrete time):

\[
\begin{align*}
\bar{x}_{k+1} &= \Phi_k \hat{x}_k + \Lambda_k u_k \\
\bar{P}_{k+1} &= \Phi_k \hat{P}_k \Phi_k^T + \Gamma_k Q_k \Gamma_k^T \\
K_k &= \bar{P}_k H_k^T (H_k \bar{P}_k H_k^T + R_k)^{-1} \\
\hat{x}_k &= \bar{x}_k + K_k (z_k - H_k \bar{x}_k) \\
\hat{P}_k &= (I - K_k H_k) \bar{P}_k
\end{align*}
\]

- \( Q \) is the process noise covariance matrix that represent the uncertainty in the process model in Eq. 1
- \( \bar{x} \) is predicted value
- \( \hat{x} \) is updated value based on measurements
- \( R \) is the measurement covariance matrix ("accuracy of the measurements"
- \( P \) is the covariance matrix of the estimate ("accuracy of the estimate")
KF equation – basic concepts

- The ratio between Q and R is important!
- Large R → small gain K
- Small R and large Q → large gain K
- Large gain K means a rapid response to the measurements
- Small gain K means a slower response to the measurements
- \((z – Hx)\) should be zero mean and white noise (if not, the model is wrong)

\[
\begin{align*}
\bar{x}_{k+1} &= \Phi_k \hat{x}_k + \Lambda_k u_k \\
\bar{P}_{k+1} &= \Phi_k \bar{P}_k \Phi_k^T + \Gamma_k Q_k \Gamma_k^T \\
K_k &= \bar{P}_k H_k^T (H_k \bar{P}_k H_k^T + R_k)^{-1} \\
\hat{x}_k &= \bar{x}_k + K_k (z_k - H_k \bar{x}_k) \\
\hat{P}_k &= (I - K_k H_k) \bar{P}_k
\end{align*}
\]

Need initial values \(x_0\) and \(P_0\)

You should understand these basic concepts of the Kalman filter!
Estimation error vs. time for rate gyro (simulation examples)

Error increases; parameter not observable (only rotation about x-axis)

Rate gyro calibration using KF; misalignment

KF spacecraft attitude estimator
KF equations - implementation

- The equations become very simple in the case of only one time-invariant unknown.
  - E.g. a single sensor axis
  - All matrices (P, H, K etc.) becomes scalar values
  - Simple to implement on an embedded processor.

- However, most real-world problems include several unknown parameters/states to be estimated, and several of them are usually time-varying.

\[
\begin{align*}
\bar{x}_{k+1} & = \Phi_k \bar{x}_k + \Lambda_k u_k \\
\bar{P}_{k+1} & = \Phi_k \bar{P}_k \Phi_k^T + \Gamma_k Q_k \Gamma_k^T \\
K_k & = \bar{P}_k H_k^T (H_k \bar{P}_k H_k^T + R_k)^{-1} \\
\hat{x}_k & = \bar{x}_k + K_k (z_k - H_k \bar{x}_k) \\
\bar{P}_k & = (I - K_k H_k) \bar{P}_k
\end{align*}
\]

For a time-invariant system (constant H, \(\Phi\), Q, \(\Gamma\) and R) the covariance P and the gain K will reach a steady-state value quickly. A Steady state solution of KF equations is easier to compute in real-time
- P and K fixed, and pre-computed.
- This is a sub-optimal solution
KF equations – implementation II

• The computational requirements are (usually) proportional to $n^3$, where $n$ is the number of states in the filter
  – compare $[p_x, p_y, p_z, v_x, v_y, v_z]^T$ vs. $[p_x, p_y, p_z, v_x, v_y, v_z, a_x, a_y, a_z, b_x, b_y, b_z]^T$

• Can get numerical issues on less powerful hardware (less than 32 bits) using fixed point.

• A powerful floating point processor/DSP is one way to reduce the impact of computational round off errors
  – Remember that your PC use 64 bits (double precision) floating point

KalmanFilterforDummies
Example: KF for 1D simple linear gyro model

- Process model – unknown constants:
  \[ x_{k+1} = x_k \quad \quad \varphi = I \]

- Measurement model: \( z(\omega) = a \cdot \omega + b \) \quad \text{matrix equation} \quad z = H \mathbf{x}
  \[ H = \begin{bmatrix} \omega & 1 \end{bmatrix} \mathbf{x} = \begin{bmatrix} a \\ b \end{bmatrix} \]

- Assume a perfect model; no process noise: \( Q = \sigma^2 = 0 \)

- Measurement covariance matrix becomes a scalar value in 1D: \( R = \sigma_v^2 \)

- Typically we do not use \( Q = 0 \); we always add a small amount of noise since our model is «never» perfect.
Estimation in nonlinear systems

\[ x_{k+1} = f_k(x_k, u_k) + \Gamma_k w_k \]
\[ z_k = h_k(x_k) + v_k \]

- Based on **linearization** (taylor series expansion) of the non-linear equations
- Requires an initial estimate of the parameters close to the true parameter values, in order to ensure that the data processing algorithm converges to the true solution
- This makes non-linear (in the unknown parameters/states) problems much more complicated!
- Most real-world problems are non-linear!
Example: GPS position calculation

- The measured pseudorange $\tilde{P}^k$ from a satellite $k$ can be expressed as
  (since we can assume no clock error in the satellite):
  $$\tilde{P}^k = \sqrt{(X^k - x)^2 + (Y^k - y)^2 + (Z^k - z)^2 + d + v} = \rho^k + d + v$$

- $c\tau = d$ is the position error due to receiver clock error, $(X^k, Y^k, Z^k)$ is the
  known position of satellite $k$, $(x, y, c)$ is the true receiver position, and $v$ is
  zero mean Gaussian white noise with variance $\sigma^2$

- This is a **nonlinear problem** on the form: $z_k = h_k(x_k) + v_k$

where

$$h(x) = \sqrt{(X^k - x)^2 + (Y^k - y)^2 + (Z^k - z)^2 + d}$$

- $x = [x, y, z, d]^T$ are the unknown parameters to be estimated.
Calibration

- Calibration is the process of comparing instrument outputs with known reference information and determining coefficients that force the output to agree with the reference information over a range of output values.

- An accurate calibration can turn low accuracy sensors into a sensor suitable for higher accuracy measurements.

- The first step in the calibration process is to make a mathematical model of the sensor.
Common sensor errors

- Sensor bias (offset)
  - If the output signal is not zero when the measured property is zero, the sensor has a bias (offset).
- Sensor scale factor/sensitivity
  - Can be linear \( y = ax + b; \text{ where } a = \text{sensitivity} \)
  - Can be non-linear
- Sensor misalignment
- Sensor Noise

Digital level to physical unit:
\[
\frac{[(DL \times LSB) - \text{offset}]}{(A \times SF)}
\]

DL = digital level, A = gain, SF = scale factor, LSB = \( \frac{V_{\text{ref}}}{2^n} \), n = number of bits
Rate gyro calibration and test facility UIO
Rate gyro sensor modelling

The sensor models include biases, misalignments and scale factor errors.

\[
\mathbf{\omega}^b = S_{\text{g}} \mathbf{\tilde{\omega}} - \beta^b - \eta^b
\]

- \( \mathbf{\omega}^b \): 3x1 vector true rate
- \( S_{\text{g}} \): 3x3 misalignment & scale factor error matrix
- \( \beta^b \): 3x1 vector bias
- \( \eta^b \): 3x1 vector noise

\[
S = I + M = \begin{bmatrix}
1 + \lambda_x & \delta_{xy} & \delta_{xz} \\
\delta_{yx} & 1 + \lambda_y & \delta_{yz} \\
\delta_{zx} & \delta_{zy} & 1 + \lambda_z
\end{bmatrix}
\]
Gyro modelling – 3D model

The true angular rate $\omega^b$ in the body frame is related to the measured angular rate $\tilde{\omega}$ in the gyro frame, by

$$\omega^b = S\tilde{\omega} -(\beta^b - \eta_v) \quad (9.1)$$

where $\beta^b$ is gyro bias (in the body frame) and $\eta_v$ is zero-mean, Gaussian white noise. The measured angular rate (in deg/sec) is given by

$$\tilde{\omega}_i = \frac{(\tilde{E}_i - O_i)}{K_{0i}} \quad (9.2)$$

where $\tilde{E}_i$ is the raw measurement from the i’th gyro in engineering unit (V), $O_i$ is the null offset value for the i’th gyro in engineering unit (V) and $K_{0i}$ is the nominal scale factor of the i’th gyro (found from the sensor data sheet).
3D rate gyro calibration

Calibration manoeuvre applied to the rate table

(Some) calibration parameters can be estimated “outside” the embedded system, such that the correction parameters only are applied directly in the real-time system

\[ \omega^b = S_g \tilde{\omega} - \beta^b \]

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Estimated value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \delta_{xy} )</td>
<td>0.53 deg</td>
</tr>
<tr>
<td>( \delta_{xz} )</td>
<td>0.018 deg</td>
</tr>
<tr>
<td>( \delta_{yx} )</td>
<td>0.29 deg</td>
</tr>
<tr>
<td>( \delta_{yz} )</td>
<td>-0.60 deg</td>
</tr>
<tr>
<td>( \delta_{zx} )</td>
<td>0.32 deg</td>
</tr>
<tr>
<td>( \delta_{zy} )</td>
<td>0.84 deg</td>
</tr>
<tr>
<td>( \lambda_x )</td>
<td>-1.22 x 10^{-2}</td>
</tr>
<tr>
<td>( \lambda_y )</td>
<td>-6.77 x 10^{-3}</td>
</tr>
<tr>
<td>( \lambda_z )</td>
<td>-7.55 x 10^{-3}</td>
</tr>
<tr>
<td>( \hat{\beta}_x )</td>
<td>-0.51 deg/sec</td>
</tr>
<tr>
<td>( \hat{\beta}_y )</td>
<td>-9.2 x 10^{-3} deg/sec</td>
</tr>
<tr>
<td>( \hat{\beta}_z )</td>
<td>-1.25 x 10^{-2} deg/sec</td>
</tr>
</tbody>
</table>
Temperature compensation

- The following linear model is used to model the offset temperature dependence:

\[ O(T) = O_0 + a_T \Delta T \]

- \( a_T \) is the temperature sensitivity coefficient and \( \Delta T \) is the temperature change

The gyro is not rotating in this test (Roll rate should be zero)!
Attitude determination system simulations

Simulated rocket angular rates

“Real-time” filter result

Post processing (non real-time) result

Post-processing using a smoother will give more accurate results than what is possible using a real-time filter, since more information is available!
Digital filter examples

• Moving average filter: \( y[n] = \frac{1}{3}(x[n] + x[n-1] + x[n-2]) \)

• 1. order low pass filter: \( y[n] = (1-a)y[n-1] + a x[n] \)
  – E.g. with \( a = 0.2 \)

• Note: moving average and low pass filtering will result in a delay (lag) in the output!

Note: \( x[n] \) is the measurement at time \( n \)
\( y[n] \) is the filter output at time \( n \)
Alpha filter – a first-order approach

• If you have a measurement $\tilde{x}_k$, you can apply a first order filter:

$$\hat{x}_k = (1 - \alpha)\bar{x}_k + \alpha\tilde{x}_k$$

• $\hat{x}_k$ is the updated (from measurements) estimate at time $k$
• $\bar{x}_k$ is the predicted (time propagated) estimate at time $k$, from a model:

$$\bar{x}_k = f(\hat{x}_{k-1})$$

  • Data fusion example: rate gyro measurement used for predicting a rotation angle and an accelerometer used as an inclinometer to measure the absolute angle.

• $\alpha$ is a scalar gain between 0 and 1 (typically constant)
• If no measurements $\tilde{x}_k$ are available, $\alpha$ is set to 0 $\rightarrow$ only prediction
• This approach will filter out noise, but a good $\alpha$ must be found from “trial and error” (possibly with some “guidelines”)
• Not as good as a Kalman filter!
  • Not an optimal solution!
Complimentary filter for data fusion

• Another simpler alternative to the Kalman filter
  – Not an optimal solution for a properly modelled random process.
  – Can be a good solution if the signals are not well-modelled, and/or
    the signal-to-noise ratio in the measurements are high.

• The idea behind the complementary filter is to take slow
  moving signals and fast moving signals and combine them.
  – The filter is based on an analysis in the frequency domain.

• The complementary filter fuses the sensor1 and sensor2 data
  by passing the former through a 1st-order low pass and the
  latter through a 1st-order high pass filter and adding the outputs.

• Possibly easy to implement on a embedded processor.
Complimentary filter architectures

- Assume two sensors that take a measurement $z$ of a constant but unknown parameter $x$, in the presence of noise $v$.
- $z_1 = x + v_1$ and $z_2 = x + v_2$
- Assume that the noise in $z_2$ is mostly high frequency, and the noise in $z_1$ is mostly low frequency.
- Two possible complimentary filter architectures to estimate $x$: 

![Diagram](https://uiopages.com/ComplimentaryFilterArchitectures.png)

[Video YouTube]
Complementary filter - balance robot

Often, there are cases where you have two different measurement sources for estimating one variable and the noise properties of the two measurements are such that one source gives good information only in low frequency region while the other is good only in high frequency region. 

→ You can use a complementary filter!

Example: Tilt angle estimation using accelerometer and rate gyro

\[ \theta \approx \int (\text{angular rate}) \, dt \]
- not good in long term due to integration

\[ \theta \approx \sin^{-1} \left( \frac{\text{accel. output}}{g} \right) \]
- only good in long term
- not proper during fast motion

\[ \hat{\theta}_k = \alpha (\theta_{k-1} + \omega_k \Delta t) + (1 - \alpha) a_k \]

Example: \( \alpha = 0.98 \)
To limit gyro drift
Complimentary INS/GPS integration with KF

GPS
State + GPS error
update rate: 1 – 50 Hz

INS
State + INS error
update rate: ~1 kHz

Kalman filter

INS error estimate

State estimate

A very common INS/GPS integration
Centralized INS/GPS integration

Theoretically optimal integration, but heavy computational burden and poor fault tolerance.
Distributed INS/GPS integration

- Two filters run in parallel
- The integration filter does the combination/data fusion
- Simpler approach than the centralized solution