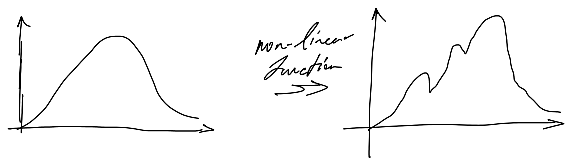
EKF Review

Wednesday, July 14, 2021 9:31 AM

< source: https://www.youtube.com/watch?v=0M8R0IVdLOI>



KF : $\hat{X}_k = F_k X_{k-1} + B_k \vec{U}_k$

↳ linear transform

↳ and Gaussian distro

* non-linear : usually where there is alot of sine & cosine

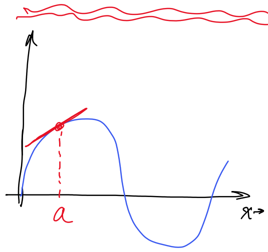
EKF: $\hat{X}_k = g(X_{k-1}, U_k)$

↳ non-linear function (transform function)

* EKF vs. KF :

in EKF (non-linear case), we Linearize our non-linear transition function about the current state ⇒ lin(g( ))

Linearization: First Order Taylor Series



$f(x) \simeq f(a) + \frac{f'(a)}{1!}(x-a)^1$

* first:

↳ linearize the transition function

$$g(X_{k-1}, U_k) = \underbrace{\frac{dg(\mu_k, U_k)}{dX_{k-1}}} \cdot (X_{k-1} - \mu_{k-1}) + g(\mu_k, U_k)$$

↳ * This becomes the Jacobian Matrix G_k

↳ Where Jacobian is defined as ≜

$$J = \begin{bmatrix} \frac{df_1}{dx_1} & \cdots & \frac{df_1}{dx_n} \end{bmatrix}$$

$$J = \begin{bmatrix} \vdots & & \vdots \\ \frac{d f_m}{d x_1} & \cdots & \frac{d f_m}{d x_n} \end{bmatrix} \quad \text{where } f\ R^n \rightarrow R^m$$

* Derivative of Transform function w/ respect to current state.

* Second:

↳ We Linearize the Measurement Function - $\underline{\underline{C}}$

$$C(X_k) \simeq \underbrace{\frac{d\,C(\mu_k)}{d\,X_k}}_{} (X_k - \mu_k) + C(\mu_k)$$

↳ $= J(C(\mu_k)) = C_k$ (or C_k)

Third: Plug everything in:

↳ Prediction Step:

$$\begin{cases} \hat{X}_k = g(X_{k-1}, U_k) \\ \hat{P}_k = G_k P_{k-1} G_k^T + Q_k \end{cases}$$

g (or sometimes f) non-lin state transform

state prediction (a mean estimation — Note Gaussian constraint)

previous state

motion or input command

Covariance matrix (previous)

noise (from process or system)

G (or F) is the linearized state transition matrix.

New estimated Covariance

Update Step:

$$\begin{cases} K_k = \hat{P}_k C_k^T (C_k \hat{P}_k C_k^T + R_k)^{-1} \\ \hat{X}'_k = \hat{X}_k + K(Z_k - C(\hat{X}_k)) \\ P'_k = P_k - K C_k P_k \end{cases}$$

K: Kalman Gain (computed)

$\hat{X}'_k$: new state estimate

P'_k: New state Covariance.

Instance Variables <source: https://filterpy.readthedocs.io/en/latest/kalman/KalmanFilter.html >

You will have to assign reasonable values to all of these before running the filter. All must have dtype of float.

x : *ndarray (dim_x, 1), default = [0,0,0…0]*
 filter state estimate

P : *ndarray (dim_x, dim_x), default eye(dim_x)*
 covariance matrix

Q : *ndarray (dim_x, dim_x), default eye(dim_x)*
 Process uncertainty/noise

R : *ndarray (dim_z, dim_z), default eye(dim_x)*
 measurement uncertainty/noise

H : *ndarray (dim_z, dim_x)*
 measurement function

F : *ndarray (dim_x, dim_x)*
 state transition matrix

B : *ndarray (dim_x, dim_u), default 0*
 control transition matrix

Optional Instance Variables

alpha : float

Assign a value > 1.0 to turn this into a fading memory filter.

Read-only Instance Variables

K : *ndarray*
    Kalman gain that was used in the most recent update() call.

y : *ndarray*
    Residual calculated in the most recent update() call. I.e., the different between the measurement and the current estimated state projected into measurement space (z - Hx)

S : *ndarray*
    System uncertainty projected into measurement space. I.e., HPH' + R. Probably not very useful, but it is here if you want it.

likelihood : *float*
    Likelihood of last measurement update.

log_likelihood : *float*
    Log likelihood of last measurement update.

<source: Kalman Filter Explained With Python Code >

* Bayes Filter (Brief)

$$\underbrace{Bel(x_t)}_{\text{Current state est.}} \triangleq \underbrace{\eta P(Z_t \mid x_t)}_{\text{Correction or Update (Gaussian)}} \underbrace{\int P(x_t \mid u_{t-1}, x_{t-1}) \cdot Bel(x_{t-1})\, d_{t-1}}_{\text{Prediction or Estimation (previous state est.) (Gaussian)}}$$

so this is Measurement

& this is prediction

* Note look up:
↳ Robot localization
↳ & Markov localization

* Bellman's equation multiplies two Gaussian hypothesis' (new measurement & state estimation) which results in another Gaussian.

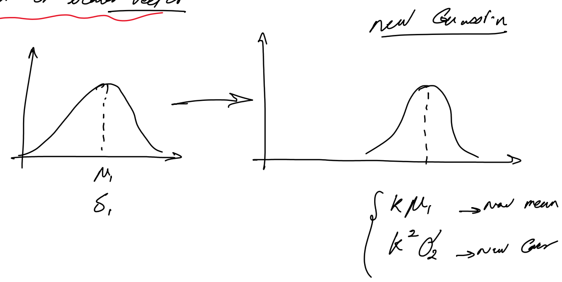
* Operations on Gaussians

* ↳ Product of a Gaussian & a scalar or scalar vector

$Cov(x) = \Sigma$ → Normal Covar

$Cov(Ax) = A \Sigma A^T$ → same old Covar

→ This is how we multiply a Gaussian w/ a vector



* ↳ Product of two Gaussians

assume:

Some hypothesis (not important here; other than it's a Gaussian distro)

$$N(x, \mu_0, \sigma_0) \cdot N(x, \mu_1, \sigma_1) = N(x, \mu', \sigma')$$

↳ New mean & Covar

* Note: ↳ same x

Then we have:

$$\begin{cases} \mu' = \mu_0 + \dfrac{\sigma_0^2(\mu_1 - \mu_0)}{\sigma_0^2 + \sigma_1^2} \\ \sigma'^2 = \sigma_0^2 - \dfrac{\sigma_0^4}{\sigma_0^2 - \sigma_1^2} \end{cases}$$

⇒ We also have

$$K = \frac{\sigma_0^2}{\sigma_0^2 + \sigma_1^2}$$

so we rewrite:

$$\begin{cases} \mu' = \mu_0 + k(\mu_1 - \mu_0) \\ \sigma'^2 = \sigma_0^2 - k\sigma_0^2 \end{cases}$$

---

so for sensor data fusion (all vectors & matrices; ndim data)

$$\begin{cases} K = \Sigma_0 (\Sigma_0 + \Sigma_1)^{-1} \\ \vec{\mu}' = \vec{\mu}_0 + k(\vec{\mu}_1 - \vec{\mu}_0) \\ \Sigma' = \Sigma_0 - K\Sigma_0 \end{cases}$$

---

* Kalman Filter

position

$$\begin{cases} P_k = P_{k-1} + V_{k-1}\Delta t + \frac{1}{2} a \Delta t^2 \\ V_k = V_{k-1} + a \Delta t \end{cases}$$

velocity

where $X \triangleq \begin{bmatrix} P_k \end{bmatrix}$

velocity $\quad \hat{X}_k = \begin{bmatrix} \\ V_k \end{bmatrix}$

$$\Rightarrow \hat{X}_k = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \hat{X}_{k-1} + \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} . a$$

$$\hat{X}_k = F \hat{X}_{k-1} + B\vec{u}$$

$\hat{X}_k$: state prior

F: prediction or transition matrix (new mean for the next covariance)

B: Control matrix (for input mapping)

$\vec{u}$: Control (input) vector

Now we calculate the new P_k (Prior Error Covar - for current state)

Prior Error Covar ($Cov(Ax) = A \Sigma A^T + \text{Noise}$)

$$* \; P_k = F_k \cdot P_{k-1} F_k^T + Q_k$$

Q_k: Process or system Noise

This is also our new Covariance

* ↳ $\hat{X}_k$ (calculated in previous step) is the new mean (estimated)

↳ Current State mean estimation

Current state mean & Covar Prediction (hypothesis) Which is based previous hypothesis

* Next

* Correction or Update Step

$$* \; \mu_k = H_k \hat{X}_k$$

H_k: ↳ measurement matrix scaling, unit conversion

* may add identity matrix here or scaling or

Current state mean & Covar

based current state measurements

New Mean (actual) * takes into account new measurements

unit conversion

$$\underbrace{\Sigma_k}_{\text{New Covariance}} = H_k P_k H_k^T + \underbrace{R_k}_{\text{measurment noise}}$$

New Covariance

New Covar for current state that is based on current state measurements

$\longrightarrow$

* Now w