Kalman Filter

Core Concept: A recursive state estimation algorithm that provides the mathematically optimal estimate of a system's state by fusing a linear mathematical model (Prediction) with noisy sensor data (Correction). It assumes all probability "clouds" are Gaussians (Bell Curves), allowing it to replace complex integrals in Bayes Filter with efficient matrix algebra.

1. The Big Picture - Gaussian Fusion:

- Recall that the Bayes Filter is the theoretical "gold standard" but involves difficult integrals to handle any shape of uncertainty.
- The Kalman Filter simplifies by assuming world is Unimodal + Symmetric (Gaussian).
- By restricting the shape of our belief to a Gaussian, we only need to keep track of two things in our Kalman Filter:
  1. Mean (μ): our best guess of the state
  2. Covariance (P): a matrix of uncertainties and correlations between different state variables (e.g. position & velocity).

2. The Variables - Vectors & Matrices:

We assume a multidimensional system, so move from scalars to Linear Algebra:

↳ x_t (state vector): parameters we are tracking (ex. position + velocity). for example in 2D case, $x_t = [p_x, p_y, v_x, v_y]^T$.

↳ P_t (covar matrix): a covariance matrix representing uncertainties of each state along the diagonal and correlation between states in other nondiagonal entries. $P_t[i,j]$ represents covariance between states $x_t[i]$ and $x_t[j]$.

↳ u_t (control input): the vector of inputs passed for the prediction step.

↳ z_t (measurement): correction data from sensor(s), such as GPS coords.

↳ F (state transition matrix): the physics model describing how the state x_t evolves on each time step.

↳ B (control matrix): describes how control input u_t changes the state x_t.

↳ Q (process noise matrix): represents the uncertainty in our physics model, like control input noise or wind.

↳ R (measurement noise matrix): represents the inherent noise/error of the sensors used in the correction step.

↳ H (observation matrix): maps our internal state (x_t) space to the sensor's measurement space, defining how the sensor observes the system relative to the state vector.

3. The Recursive Estimation Algorithm:

- Step A: Prediction [The Prior - $\overline{Bel}(x_t)$]:

  i) State Prediction: $x_t = Fx_{t-1} + Bu_t$ ①

  - Here, we are moving our mean forwards based on the motion model.
  - Recall F represents a physical transformation on $\vec{x}_{t-1}$ to get to $\vec{x}_t$ and B represents a transformation on $\vec{u}_t$ to get effect on $\vec{x}_t$.

  ii) Uncertainty Propagation: $P_t = FP_{t-1}F^T + Q$ ②

  - Here, we propagate the uncertainty to a new covariance matrix.
  - The logic behind this formula is simply to understand that by definition, $P_t = cov(x_t)$. We use x_t from equation ①. See appendix proof #3 for more details!

- Step B: Correction (The Posterior):

  i) Kalman Gain: $K_t = P_t H^T (HP_tH^T + R)^{-1}$ ③

  - The Kalman Gain represents the relative trust between your prediction and the sensor. Is used for weighted state update.
  - Mathematically optimized solution to minimize state covariance, can be seen in Proof #4.
  - Intuitively can be understood as follows. We know H maps from state to sensor space. So, H^T is the opposite: P_tH^T maps measurement residuals to state corrections. Note $(HPH^T + R)$ is just $cov(Hx_t + r_t)$ where r_t is sensor noise. So, it is total uncertainty in measurement space.

- So, $P_t H^T \cdot (HPH^T + R)^{-1}$ is ratio between the state uncertainty and the total measurement uncertainty! In essence, it tells us how much to trust sensor vs. prediction in state update.

ii) State Update: $x_t = \hat{x}_t + K_t(z_t - H\hat{x}_t)$ ④ ($\hat{x}_t$ prior, x_t posterior)

- This updates our state, or mean, based on Kalmain Gain.
- Note that Hx_t transforms $\hat{x}_t$ to sensor space. So, $H\hat{x}_t$ is $z_{predicted}$, i.e. what our sensor should see based on current state $\hat{x}_t$.
- So, $(z_t - H\hat{x}_t)$ is the innovation, the difference between the actual sensor reading and the expected one.
- We multiply the innovation by the Kalman Gain to correct $\hat{x}_t$.

iii) Uncertainty Update: $P_t = (I - K_t H)\hat{P}_t$ ⑤

- This updates the uncertainty that the measurement explained.
- Derivation can be seen in proof #5 in appendix.
- Intuititively, more data means less uncertainty.
- $K_t H$ represents the part of the state uncertainty that was corrected via observation from the sensor-based update. Subtracting it from I means: keep only the uncertainty the measurement was NOT able to reduce.
- The uncertainty thus shrinks only in spaces the sensor sees. If a state component is strongly observable via the sensor (high gain) its variance drops a lot. If it's weakly observable it barely changes.

4. Concrete Example - 4-State KF (2D Position + 2D Velocity)

Scenario: We want to construct a Kalman Filter to track a robot's position and velocity in 2D. We use an accelerometer as the control input and a 2D GPS for correction. We assume the GPS gives only 2D position p_x, p_y.

1. Define Vectors ($\vec{x}, \vec{u}, \vec{z}$): $\vec{x} = \begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}$; $\vec{u} = \begin{bmatrix} a_x \\ a_y \end{bmatrix}$ (accel) ; $z = \begin{bmatrix} gps_x \\ gps_y \end{bmatrix}$ (gps)

2. Derive Physical Models (F, B, H):

i) We define F using kinematics. We want $\vec{x}_t$ based on $\vec{x}_{t-1}$. Observe, $p_{x,t} = p_{x,t-1} + v_{x,t-1}\Delta t$ and $p_{y,t} = p_{y,t-1} + v_{y,t-1}\Delta t$. Also, acceleration comes from our control input so for F we simply take $v_{x,t} = v_{x,t-1}$ and $v_{y,t} = v_{y,t-1}$.

$$\Rightarrow F = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ii) We define B via more kinematics. The B matrix below multiplies with $\vec{u}$ such that $B\vec{u}$ maps to update for $\vec{x} = [p_x, p_y, v_x, v_y]^T$, via $p_i = \frac{1}{2}a_i\Delta t^2$ and $v_i = a_i\Delta t$, for $i \in \{x, y\}$.

$$\Rightarrow B = \begin{bmatrix} 0.5\Delta t^2 & 0 \\ 0 & 0.5\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} ; \vec{u} = \begin{bmatrix} a_x \\ a_y \end{bmatrix}$$

iii) The observation matrix H is quite simple to define. For simplicity we assume position space to be common between $\vec{x}$ and $\vec{z}$. Then, H such that $\vec{z}_{expected} = H\vec{x}$ is simply:

$$\Rightarrow H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$ (strips away v_x, v_y).

3. Derive Noise Models (Q, R):

i) For Q, we take accelerometer error as our main source of noise. Suppose we have a known scalar a_{noise} (i.e. variance) from datasheet.

$$\Rightarrow Q = a_{noise} \cdot \begin{bmatrix} 0.25\Delta t^4 & 0 & 0.5\Delta t^3 & 0 \\ 0 & 0.25\Delta t^4 & 0 & 0.5\Delta t^3 \\ 0.5\Delta t^3 & 0 & \Delta t^2 & 0 \\ 0 & 0.5\Delta t^3 & 0 & \Delta t^2 \end{bmatrix}$$

The matrix a_{noise} is scaling is known as the 'Piecewise White Noise Model.' Google it to learn more, derivation not in scope of this note.

ii) To formulate R, we assume a simplistic noise model for the GPS where the x and y error are independant. Assuming we have some scalar GPS_{noise} (i.e. variance from datasheet):

$$\Rightarrow R = GPS_{noise} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

4. Example Step:

i) Initial State: We set $\vec{x}_0 = [0\ 0\ 0\ 0]^T$. We also initialise P_0 as the identity matrix I_4. This is a normal way to init P since we assume each of the 4 variables have unit variance and are initially uncorrelated to start.

ii) Predict: Suppose we sense $a_x = 2.0\ m/s^2$ and $a_y = 1.0\ m/s^2$ from accel for $\Delta t = 1.0$. $\vec{x}_t = F\vec{x}_0 + B\vec{u}_t$ gives $\vec{x}_t = [1.0, 0.5, 2.0, 1.0]^T$. We also propagate $P_t = FP_0F^T + Q$ for the new covar matrix. Note in reality time steps would be much smaller and we would run many prediction steps between corrections, due to accel sample rate being much higher than GPS. Ex. if we sample accel @ 100 Hz and GPS @5Hz we would run predict ~20 times between each correction update.

iii) Correction: Suppose GPS returns position $\vec{z}_t = [1.2, 0.4]^T$. We would calculate $K_t = P_tH^T(HP_tH^T + R)^{-1}$ and apply the updates $x_t = \hat{x}_t + K_t(z_t - H\hat{x}_t)$ and $P_t = (I - K_tH)\hat{P}_t$.

iv) Repeat: We then go back to (ii) and keep recursively solving!

5. Looking Ahead - The Jump to Non-Linearity:

- Linear Limitation: Standard KF requires system to be linear due to modelling it via a state transition matrix F.
- EKF Solution: The Extended Kalman Filter (EKF) bypasses this using Taylor Series to perform Jacobian Linearization. This creates localised linear approximations of curved trajectories.

Appendix

Proof #1: $Cov(x+y) = Cov(x) + Cov(y)$, where x, y independant

By def'n: $Cov(x) = E[(x-\mu_x)(x-\mu_x)^T]$, where $\mu_x = E[x]$.

Note $\mu_{x+y} = E[x+y] = E[x] + E[y] = \mu_x + \mu_y$.

Now, $Cov(x+y) = E[(x+y-\mu_x-\mu_y)(x+y-\mu_x-\mu_y)^T]$

$= E[((x-\mu_x)+(y-\mu_y))((x-\mu_x)+(y-\mu_y))^T]$

$= E[\underbrace{(x-\mu_x)(x-\mu_x)^T}_{Cov(x)} + \underbrace{(x-\mu_x)(y-\mu_y)^T}_{Cov(x,y)} + \underbrace{(y-\mu_y)(x-\mu_x)^T}_{Cov(y,x)} + \underbrace{(y-\mu_y)(y-\mu_y)^T}_{Cov(y)}]$

$= Cov(x) + Cov(y) + \cancel{Cov(x,y)}^{\,0} + \cancel{Cov(y,x)}^{\,0}$ [by x, y indep.]

$Cov(x+y) = Cov(x) + Cov(y)$, QED

Proof #2: $Cov(Ax) = A\,Cov(x)\,A^T$

Let $y = Ax$. Then, $\mu_y = E[Ax] = AE[x] = A\mu_x$.

Now, $Cov(y) = E[(Ax - A\mu_x)(Ax - A\mu_x)^T]$

$= E[A(x-\mu_x)(x-\mu_x)^T A^T]$

$= A\underbrace{E[(x-\mu_x)(x-\mu_x)^T]}_{Cov(x)}A^T$

$Cov(Ax) = A\,Cov(x)\,A^T$, QED

Proof #3: $P_t = FP_{t-1}F^T + Q$ [uses pf. 1, 2]

Note in $x_t = Fx_{t-1} + Bu_t$, we assume u_t has no variance and add a $+q_t$ term for process noise, s.t. $x_t = Fx_{t-1} + Bu_t + q_t$, s.t. $Cov(q_t) = Q$.

Then, $P_t = Cov(x_t) = Cov(Fx_{t-1} + Bu_t + q_t)$

$= Cov(Fx_{t-1}) + Cov(Bu_t) + Cov(q_t)$

$= F\underbrace{Cov(x_{t-1})}_{=P_{t-1}}F^T + \underbrace{B\,\cancel{Cov(u_t)}\,B^T}_{=\vec{0}} + \underbrace{Cov(q_t)}_{=Q}$.

$\Rightarrow P_t = FP_{t-1}F^T + Q$, QED

Proof #4: $K_t = P_t H^T (HP_tH^T + R)^{-1}$ ⇒ optimal Kalman Gain

To derive optimal Kalman Gain, we are simply looking for K_t such that we minimise the posterior covar matrix P_t after performing the update $x_t = \hat{x}_t + K_t(z_t - H\hat{x}_t)$. To begin, we define the posterior error as $e_t = \tilde{x}_t - x_t$ where $\tilde{x}_t$ is the actual state. Then,

$$
\begin{aligned}
e_t &= \tilde{x}_t - x_t \quad , \text{sub } x_t = \hat{x}_t + K_t(z_t - H\hat{x}_t) \\
&= \tilde{x}_t - [\hat{x}_t + K_t(z_t - H\hat{x}_t)] \quad , \text{note } z_t = H\tilde{x}_t + r_t \text{ (noise with covar } R) \\
&= (\tilde{x}_t - \hat{x}_t) - K_t(H\tilde{x}_t + r_t - H\hat{x}_t) \\
e_t &= (I - K_tH)(\tilde{x}_t - \hat{x}_t) - K_t r_t .
\end{aligned}
$$

Now, note $P_t = E[e_t e_t^T]$. Assuming prior error $(\tilde{x}_t - \hat{x}_t)$ and noise r_t are uncorrelated, we can apply this to get:

$$
\begin{aligned}
P_t &= (I - K_tH)\hat{P}_t(I - K_tH)^T + K_tRK^T \\
\Rightarrow P_t &= \hat{P}_t - K_tH\hat{P}_t - \hat{P}_tH^TK_t^T + K_t(H\hat{P}_tH^T + R)K_t^T
\end{aligned}
$$

To find the optimal K_t we can minimise the trace of P_t by taking the derivative of $Tr(P_t)$ w.r.t. K_t and setting it to 0. Using the matrix calculus identities below:

$$
\frac{\partial Tr(AB)}{\partial A} = B^T \quad \text{and} \quad \frac{\partial Tr(ACA^T)}{\partial A} = 2AC
$$

We get: $\frac{\partial Tr(P_t)}{\partial K_t} = -2(H\hat{P}_t)^T + 2K_t(H\hat{P}_tH^T + R) = 0$.

Solving for K_t:

$$
\begin{aligned}
2K_t(H\hat{P}_tH^T + R) &= 2(H\hat{P}_t)^T \quad , \text{note } (AB)^T = B^TA^T \\
K_t(H\hat{P}_tH^T + R) &= \hat{P}_t^TH^T \quad , \text{note } \hat{P}_t^T = \hat{P}_t \text{ (symmetric)} \\
K_t(H\hat{P}_tH^T + R) &= \hat{P}_tH^T
\end{aligned}
$$

$$
\Rightarrow K_t = \hat{P}_tH^T(H\hat{P}_tH^T + R), \quad \text{QED}
$$

Proof #5: $P_t = (I - K_t H)\hat{P}_t$

We take the result below from Proof 4:

$$P_t = (I - K_t H)\hat{P}_t(I - K_t H)^T + K_t R K^T$$

This is known as the Joseph Form and is perfectly valid to use. However it is computationally expensive, and can be simplified by plugging in our optimal K_t formula. Recall from Proof 4:

$$P_t = \hat{P}_t - K_t H \hat{P}_t - \hat{P}_t H^T K_t^T + K_t(H\hat{P}_t H^T + R)K_t^T$$

We also recall $K_t(H\hat{P}_t H^T + R) = \hat{P}_t H^T$ from Proof 4. Substitute:

$$P_t = \hat{P}_t - K_t H \hat{P}_t - \hat{P}_t H^T K_t^T + (\hat{P}_t H^T) K_t^T$$

$$= \hat{P}_t - K_t H \hat{P}_t$$

$$P_t = (I - K_t H)\hat{P}_t \text{ , QED}$$