Kalman Filter Theory: A Review

May 25, 2022

1 Kalman Filter Theory

1.1 Notations

Time update (prediction step)

A posteriori mean

$$\hat{x}_{k-1}^{+}$$

A posteriori covariance

$$\hat{P}_{k-1}^{-}$$

Though prediction with linear system model

A priori mean

$$\hat{x}_{k}^{-}$$

A priori covariance

$$\hat{P}_{k}$$

Measurement update (correction step)

A priori mean

$$\hat{x}_{k}^{-}$$

A priori covariance

$$\hat{P}_{k}$$

Through update with linear measurement model

- Measurement z_k
- Measurement covariance R_k

A posteriori mean

$$\hat{x}_{k}^{+}$$

A posteriori covariance

$$\hat{P}_{k}^{+}$$

2 Discrete-time linear system model

Process model:

$$x_k = F_{k-1}x_{k-1} + G_{k-1}\mu_{k-1} + L_{k-1}w_{k-1}$$

Measurement model:

$$z_k = H_k x_k + M_k v_k$$

Note:
x_k: state vector
μ_k: control input vector
w_k: process model noise vector
v_k: measurement noise vector
F_k: state transition matrix
G_k: control input matrix
H_k: measurement matrix
L_k: process model noise sensitivity matrix
M_k: measurement model noise sensitivity matrix
Also note:
Gaussian distribution with zero mean and given covariance matrix:
Process model noise vector:

$$w_k \sim N(0, Q_k)$$

Measurement noise vector:

$$v_k \sim N(0, R_k)$$

Noise is not correlated with time:

$$E(w_k w_j^T) = Q_k \delta_{k-j}$$

$$E(v_k v_j^t) = R_k \delta_{k-j}$$

Process noise and measurement noise are independent:

$$E(w_k v_j^T) = 0$$

2.1 Prediction step (State)

We use the process model of the system that we want to estimate to predict the estimated state (mean of the Gaussian probability density function) forward in time.

$$\hat{X}_k^- = F_{k-1}\hat{X}_{k-1}^+ + G_{k-1}\mu_{k-1}$$

2.2 Prediction step (Covariance)

$$P_0^+ = E[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^T]$$

So given P_0^+, we can propagate this uncertainty with time using the linear transformation:

$$P_1^- = F_0 P_0^+ F_0^T$$

Predicted (a priori) state estimate:

$$\hat{X}_k^- = F_{k-1}\hat{X}_{k-1}^+ + G_{k-1}\mu_{k-1}$$

Predicted (a priori) covariance estimate:

$$P_k^- = F_{k-1}P_{k-1}^+ F_{k-1}^T + Q_{k-1}$$

2.3 Update step (State)

Measurement model:

$$z_k = H_k x_k + M_k v_k$$

Innovation:

$$\tilde{y_k} = z_k - H_k \hat{x}_k^-$$

Innovation covariance:

$$S_k = H_k P_k^- H_k^T + R_k$$

Note:
z_k: measurement vector
$\hat{z_k}$: predicted measurement vector
$\tilde{y}_k$: measurement innovation (error) vector
S_k: innovation covariance matrix
The state is updated based on the size of innovation using Kalman Gain matrix K_k.

State update:

$$\hat{X}_k^+ = \hat{X}_k^- + K_k \tilde{y}_k$$

Kalman Gain (Recursive Least Square estimate):

$$K_k = P_k^- H_k^T S_k^{-1}$$

2.4 Update step (Covariance)

The update process reduces uncertainty in the estimates, and so covariance must change.

Covariance update:

$$P_k^+ = (I - K_k H_k) P_k^-$$

3 Discrete-time non-linear system model

Process model:

$$x_k = f(x_{k-1}, \mu_k, w_k)$$

Measurement model:

$$z_k = h(x_k, v_k)$$

Notations and assumptions apply as in linear model.

Linear vs. Extended Kalman filter

- Linear Kalman Filter
  - Best linear estimator
  - Stable for any initial conditions
  - Stable for any perturbations
- Extended Kalman Filter

  EKF: a recursive estimator that solves the linear quadratic estimation problem using minimum mean squared error (MSSE) method:

  $$\hat{x}_{i|j} = argmin_{\hat{x}_{i|j} \in \mathbf{R}^n} E[(x_i - \hat{x}_{i|j})(x - \hat{x}_{i|j})^T | z_1, ..., z_j]$$

  - Not the best estimator

- No guarantees on stability, can diverge

Note:
True state: x
Estimated state: $\hat{x}$
Estimation error: $\tilde{x} = x - \hat{x}$
a priori state:

$$\hat{x}_k^- = \hat{x}_{k|k-1} = E[x_k|Z^{k-1}]$$

a posteriori state:

$$\hat{x}_k^+ = \hat{x}_{k|k} = E[x_k|Z^k]$$

a priori covariance:

$$P_k^- = P_{k|k-1} = E[\tilde{x}_{k|k-1}\tilde{x}_{k|k-1}^T|Z^{k-1}]$$

a posteriori covariance:

$$P_k^+ = P_{k|k} = E[\tilde{x}_{k|k}\tilde{x}_{k|k}^T|Z^k]$$

Jacobian

Jacobian matrix ∇f_x is a $(m \times n)$ matrix for the function $f = f(x)$ whose elements are the partial derivatives of the m-outputs of the functions with respect to n-inputs.

State and noise Jacobians: ∇f_x is a $(n \times n)$ matrix, whereas ∇f_w is a $(n \times l)$ matrix.

3.1 Prediction step (State)

State prediction step:

$$\hat{X}_k^- = f(\hat{X}_{k-1}^+, u_k, 0)$$

$$(\hat{x}_{k-1}^+ = \hat{x}_{k-1|k-1}) \to f(\cdot) \to (\hat{x}_k^- = \hat{x}_{k|k-1})$$

i.e.
$E[x_{k-1}|Z_{k-1}]$ (a posteriori state) → $f(\cdot)$ → $E[x_k|Z_{k-1}]$ (a priori state)

3.2 Prediction step (Covariance)

We also want to propagate the covariance forward in time as well:

$$P_0 = E[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T]$$

So given P_0, we want to propagate time forward to find P_1^-, using the linear transformation, where the transformation matrices are *a linear approximation of the non-linear model* (just replace by the linear matrices by Jacobians).

In general:

$$P_k^- = \nabla f_x P_{k-1}^+ \nabla f_x^T + \nabla f_w Q_k \nabla f_w^T$$

where the Jacobian matrices are:
(evaluated at the best estimate so far)

$$\nabla f_x = \frac{\partial f}{\partial x}|_{x=\hat{x}_{k-1}^+}$$

$$\nabla f_w = \frac{\partial f}{\partial w}|_{x=\hat{x}_{k-1}^+}$$

Thus,

a posteriori error (covariance for $k-1$) → $\nabla f(\cdot)$ → a priori error (covariance for k)

Proof: Taylor series expansion.

3.3 Update step (State)

The Kalman Filter corrects for state estimation error by feeding back a weighted term based on observed measurement errors.

We can predict the measurement $\hat{z_k}$ based on the current a priori state estimate $\hat{x}_k^-$, using the nonlinear measurement function:

$$\hat{z}_k = h(\hat{x}_k^-, 0)$$

We can then define the innovation vector (measurement error) as:

$$v_k = z_k - h(\hat{x}_k^-, 0)$$

State update:

$$\hat{x}_k^+ = \hat{x}_k^- + K_k v_k$$

Kalman Gain:

$$K_k = P_k^- \nabla h_{x_k}^T S_k^{-1}$$

Recall a priori state for k:

$$\hat{x}_k^- = E[x_k | Z^{k-1}]$$

and, a posteriori state for k:

$$\hat{x}_k^+ = E[x_k | Z^k]$$

3.4 Update step (covariance)

The innovation covariance S_k is defined as:

$$S_k = E[v_k v_k^T]$$

The term can be calculated using Jacobians:

$$S_k = \ \nabla h_x P_k^- \nabla h_x^T + \nabla h_v R_k \nabla h_v^T$$

Jacobians are partials evaluated at $x = \hat{x}_k^-$.

The a priori state covariance matrix P_k^- can also be updated with the same information to form the a posteriori state error covariance matrix P_k^+ using:

$$P_k^+ = (I - K_k \nabla h_{x_k}) P_k^-$$

Summary: LKF vs. EKF update step

- LKF: update with linear model

  State:

  $$\hat{x}_k^+ = \hat{x}_k^- + K_k v_k$$
  $$K_k = P_k^- H_k^T S_k^{-1}$$

  Covariance:

  $$P_k^+ = (I - K_k H_k) P_k^-$$

- EKF: update with linear **approximation**

  State:

  $$\hat{x}_k^+ = \hat{x}_k^- + K_k v_k$$

$$K_k = P_k^- \nabla h_k^T S_k^{-1}$$

Covariance:

$$P_k^+ = (I - K_k \nabla h_{x_k}) P_k^-$$

Linear uncertainty transformation

$$X \sim N(\bar{X}, \Sigma_X)$$

Suppose

$$Y = AX + B$$

$$Y \sim N(\bar{Y}, \Sigma_Y)$$

can be written as:

$$Y \sim N(A\bar{X} + B, A\Sigma_Y A^T)$$

Remark:

- The Jacobian and linear transformation in general well if the system is approximately linear at the linearization point.
- Linear approximation of uncertainty tends to underestimate error, leading to overconfident estimates.

4 Unscented Kalman Filter

UKF is a variant of the EKF that uses the unscented transform for a better approximation of the nonlinear probability distribution transformations used inside the filter.

Operation: Calculates the non-linear transform of key sample points (Sigma Points), then fits a Gaussian distribution to the transformed points.

Properties of UKF:

- 3rd-order accurate for mean and covariance
- Does not require Jacobian calculation
- Slower calculation speed
- Overall: increased accuracy but slower calculation

Let x be a $(n \times 1)$ vector with mean $\bar{x}$ and covariance P.
Choose $2n$ sigma points $x^{(i)}$ as follows:

$$x^{(i)} = \bar{x} + \Delta x^{(i)}, i = 1, ..., 2n$$

$$\Delta x^{(i)} = (\sqrt{nP})_i, i = 1, ..., n$$

$$\Delta x^{(n+i)} = -(\sqrt{nP})_i, i = 1, ..., n$$

Mean approximation:

$$y^{(i)} = h(x^{(i)}), i = 1, ..., 2n$$

$$\bar{y} = \sum_{i=1}^{2n} W^{(i)} y^{(i)}$$
$$W^{(i)} = \frac{1}{2n}$$

Covariance approximation:

$$P_y = \sum_{i=1}^{2n} W^{(i)} (y^{(i)} - \bar{y})(y^{(i)} - \bar{y})^T$$

General unscented transformation

Idea: allows using different weights, or weighting function.

4.1 Prediction step

Case: Additive noise

$$x_k = f(x_{k-1}, u_k) + w_k$$

Generate Sigma Points for the unscented transform for the prediction step

$$\hat{x}_{k-1}^{(0)} = \hat{x}_{k-1}^{+}$$

$$\hat{x}_{k-1}^{(i)} = \hat{x}_{k-1}^{+} + \Delta x^{(i)}, i = 1, ..., 2n$$

Sigma points:

$$\Delta x^{(i)} = (\sqrt{(n+\kappa)P_{k-1}^{+}})_i$$

$$\Delta x^{(n+i)} = -(\sqrt{(n+\kappa)P_{k-1}^{+}})_i$$

Case: General noise model

$$x_k = f(x_{k-1}, u_k, w_k)$$

State prediction step:

$$\hat{x}_k^{(i)} = f(\hat{x}_{k-1}^{(i)}, u_k, w_k^{(i)})$$

Calculate mean:

$$\hat{x}_k^{-} = \sum_{i=0}^{2n} W^{(i)} \hat{x}_k^{(i)}$$

$$W^{(0)} = \frac{\kappa}{\kappa + n}$$

$$W^{(i)} = \frac{1}{2(n+\kappa)}, i = 1, ..., 2n$$

Calculate covariance:

$$P_k^{-} = \sum_{i=0}^{2n} W^{(i)} (\hat{x}_k^{(i)} - \hat{x}_k^{-})(\hat{x}_k^{(i)} - \hat{x}_k^{-})^T$$

4.2 Update step

Innovation (measurement error):

$$v_k S_k P_{xz}$$

Update with unscented approximation:
State:

$$\hat{x}_k^{+} = \hat{x}_k^{-} + K_k v_k$$

$$K_k = P_{xz} S_k^{-1}$$

Covariance:

$$P_k^{+} = P_k^{-} - K_k S_k K_k^T$$

5 Filtering

5.1 Sensor models and errors

Errors:

- Deterministic

  Bias error: $z = x + b$

  Scale factor error: $z = ax$

  Nonlinearity error

  Asymmetry error

  Dead zone error

  Quantization error

- Stochastic

  White noise

  Bias drift (a function of time plus white noise)

  Scale factor instability

  Run to run bias

Dealing with deterministic errors: calibration

- Calculate the relationship difference between sensor measurements and the known truth
- Use the inverse relationship to compensate the measurement to return it into the ideal measurement
- Fuse compensated measurement into Kalman filter

Dealing with stochastic errors: more difficult, can't use prior knowledge

- Noise is dealt with via KF implicitly (Gaussian noise)
- Bias drift, as well as other error sources must be estimated and compensated for as the system runs
- The parameters must be included in the model

5.2 Faulty data

Types of faults:

- Spike: ignore the spike
- Lock-in-place
- Hard-over
- Dead
- Float

Dealing with sensor faults: inspect measurement innovation called **innovation checking**:

- Zero mean
- Innovation covariance (estimated in the update step)

The innovation should be a normally distributed random variable:

$$v \sim N(0, S)$$

Ignore measurement if

$$|v| > 2\sqrt{S}$$

Normalized Innovation Squared (NIS)

For multidimensional case we use Normalized Innovation squared (NIS). Let v be the m-dimensional innovation vector, and let S be the innovation covariance matrix. The NIS can be calculated using:

$$\epsilon = v^T S^{-1} v$$

We can test to see if the innovation belongs to the unit Gaussian distribution for a specified probability level using the Chi-Squared test:

$$\epsilon < \chi^2_m(p)$$

Note: m = Degrees of Freedom, or the number of dimensions; p-value: p = 0.05, 95 % confidence level that the measurement belongs to this distribution.

Rule of thumb: measurement out of range if the NIS is greater than the Chi-Squared threshold.

5.3 Dealing with stochastic noise

Dealing with stochastic errors is hard.

Recall measurement model:

$$z = h(x, v) + b$$

State model:

$$x_a = \begin{bmatrix} x \\ b \end{bmatrix}$$

State process model:

$$x_a = \begin{bmatrix} f(x, u, w) \\ b + w_b \end{bmatrix}$$

State process noise:

$$Q_a = \begin{bmatrix} Q & 0 \\ 0 & \sigma_b^2 \end{bmatrix}$$

We can use a joint estimation process to estimate the bias parameter:

Include the bias parameter in the measurement model and as an estimated state in the state vector to form an augmented state vector.

We can model the dynamics of the bias parameter as a stationary constant, such that $\dot{b} = 0$, and set the process noise to be how quickly we want to allow this parameter estimate to change or drift.

5.4 Dealing with initial conditions

Filter initialization

- Filter state must be initialized close to the truth
- Filter error covariance must be small
  - so that does not include nonlinearity in the error range
  - error covariance need to set to approximately the true uncertainty size
- We should not start the filter until we have a good estimate of the starting state and error

6 Summary

6.1 Linear Kalman Filter Problem

Problem:

$$\hat{x}_{i|j} = argmin \hat{x}_{i|j \in \mathbf{R}^n} E[(x_i - \hat{x}_{i|j})(x_i - \hat{x}_{i|j})^T | z_1, ..., z_j]$$

Subject to system model:

$$x_k = F x_{k-1} + G u_{k-1} + w_{k-1}$$

$$z_k = H x_k + v_k$$

Assumptions:

$$w_k \sim N(0, Q_k)$$
$$v_k \sim N(0, R_k)$$

Predict:

$$\hat{x}_k^- = F_{k-1} \hat{x}_{k-1}^+ + G_{k-1} u_{k-1}$$
$$P_k^- = F_{k-1} P_{k-1}^+ F_{k-1}^T + Q_{k-1}$$

Update:

$$\tilde{y} = z_k - H_k \hat{x}_k^-$$

$$S_k = H_k P_k^- H_k^T + R_k$$

Kalman Gain:

$$K_k = P_k^- H_k^T S_k^{-1}$$

Update the state estimate:

$$\hat{x}_k^+ = \hat{x}_k^- + K_k \tilde{y}_k$$

Update the covariance matrix:

$$P_k^+ = (I - K_k H_k) P_k^-$$

Validation:
Minimize error between Kalman Filter state estimate and the true states:

min RMSE

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\tilde{x}_i^2)}$$

Tuning:

Ensure measurement innovation is close to zero mean:

$$\bar{z} \approx 0$$

Ensure measurement innovation covariance is close to actual innovation covariance:

$$\bar{S} \approx \sigma_z^2$$