Basics

Discrete-Time State-Space Systems:

$$x[k] = \begin{bmatrix} x_1[k] \\ \vdots \\ x_n[k] \end{bmatrix} \in \mathbb{R}^n \quad u[k] = \begin{bmatrix} u_1[k] \\ \vdots \\ u_m[k] \end{bmatrix} \in \mathbb{R}^m \quad y[k] = \begin{bmatrix} y_1[k] \\ \vdots \\ y_p[k] \end{bmatrix} \in \mathbb{R}^p$$

state, input (control), measurement

Difference equations:

$$x[k+1] = F\,x[k] + G\,u[k] \qquad F \in \mathbb{R}^{n\times n}$$
$$y[k] = H\,x[k] \qquad G \in \mathbb{R}^{n\times m}, \; H \in \mathbb{R}^{p\times n}$$

Gaussian distribution:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \triangleq \mathcal{N}(\mu, \sigma^2)$$

Multivariate Gaussian distribution:

$$p(x) = \frac{1}{\sqrt{(2\pi)^n \det(\Sigma)}} \exp\left[-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right] \triangleq \mathcal{N}(\mu, \Sigma)$$

Expectation or mean:

$$E(x) = \int x\,p(x)\,dx = \mu$$

Mean vector:

$$E(x) = \mu \in \mathbb{R}^n$$

Variance:

$$Var(x) = E((x-\mu)^2) = \sigma^2$$

Covariance matrix:

$$Cov(x) = E((x-\mu)(x-\mu)^T) = \Sigma \in \mathbb{R}^{n\times n}$$

Σ is symmetric and positive definite

$$\Sigma = \Sigma^T > 0$$

Suppose n=2

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}$$

$$\Sigma = E((x-\mu)(x-\mu)^T) = E\left(\begin{bmatrix} x_1-\mu_1 \\ x_2-\mu_2 \end{bmatrix}\begin{bmatrix} x_1-\mu_1 & x_2-\mu_2 \end{bmatrix}\right) = E\begin{bmatrix} (x_1-\mu_1)^2 & (x_1-\mu_1)(x_2-\mu_2) \\ (x_2-\mu_2)(x_1-\mu_1) & (x_2-\mu_2)^2 \end{bmatrix}$$

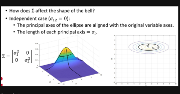
$$= \begin{bmatrix} E((x_1-\mu_1)^2) & E((x_1-\mu_1)(x_2-\mu_2)) \\ E((x_2-\mu_2)(x_1-\mu_1)) & E((x_2-\mu_2)^2) \end{bmatrix}$$

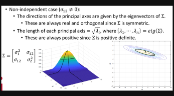
Diagonal terms are just variances of individual components. σ_{ij} is the covariance between x_i and x_j

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{bmatrix}$$

$\sigma_{ij} = 0$ if x_i and x_j are statistically independent

$\sigma_{ij} = \sigma_{ji}$ (symmetric)





Adding a Gaussian Random Variable with a deterministic vector:

$$x \sim \mathcal{N}(\mu, \Sigma), \quad y \in \mathbb{R}^n$$
$$(x+y) \sim \mathcal{N}(\mu+y, \Sigma)$$

$$E(x+y) = E(x) + E(y) = \mu + y$$
$$Cov(x+y) = E((x+y-E(x+y))(x+y-E(x+y))^T) = E((x+y-\mu-y)(x+y-\mu-y)^T) = E((x-\mu)(x-\mu)^T) = \Sigma$$

Adding independent Gaussian Random Variables:

$$x \sim \mathcal{N}(\mu_x, \Sigma_x), \quad y \sim \mathcal{N}(\mu_y, \Sigma_y)$$
$$(x+y) \sim \mathcal{N}(\mu_x+\mu_y, \Sigma_x+\Sigma_y)$$

$$E(x+y) = E(x) + E(y) = \mu_x + \mu_y$$
$$Cov(x+y) = E((x+y-E(x+y))(x+y-E(x+y))^T) = E((x+y-\mu_x-\mu_y)(x+y-\mu_x-\mu_y)^T)$$
$$= E((x-\mu_x)(x-\mu_x)^T) + E((x-\mu_x)(y-\mu_y)^T) + E((y-\mu_y)(x-\mu_x)^T) + E((y-\mu_y)(y-\mu_y)^T)$$
$$= \Sigma_x + 0 + 0 + \Sigma_y = \Sigma_x + \Sigma_y$$

Multiplying a Random Variable by a matrix:

$$x \sim \mathcal{N}(\mu, \Sigma), \quad A \in \mathbb{R}^{m\times n}$$
$$Ax \sim \mathcal{N}(A\mu, A\Sigma A^T)$$

$$E(Ax) = A\,E(x) = A\mu$$
$$Cov(Ax) = E((Ax-A\mu)(Ax-A\mu)^T) = A\,E((x-\mu)(x-\mu)^T)A^T = A\Sigma A^T$$

Kalman Filter

Assumes discrete-time linear system with noises of Gaussian distributions

$$x[k+1] = F\,x[k] + G\,u[k] + v[k] \quad \text{motion model}$$
$$y[k] = H\,x[k] + w[k] \quad \text{sensor model}$$

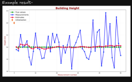
- $v[k] \sim \mathcal{N}(0, V[k])$: Process noise
- $w[k] \sim \mathcal{N}(0, W[k])$: Measurement noise

Kalman Filter represents state estimation with a Gaussian distribution

$$\hat{x}[k|j] \sim \mathcal{N}(\hat{x}[k|j], \hat{\Sigma}[k|j])$$

best guess of x at time k given measurement up to time j

Example result:



1. Prediction Step:

$$\hat{x}[k|k] \rightarrow \hat{x}[k+1|k]$$

Dynamic model -

$$\hat{x}[k+1|k] = F\,\hat{x}[k|k] + G\,u[k] + v[k]$$
$$\hat{x}[k|k] \sim \mathcal{N}(\hat{x}[k|k], \hat{\Sigma}[k|k]) \qquad v[k] \sim \mathcal{N}(0, V[k])$$

Mean - $\hat{x}[k+1|k] = E(F\hat{x}[k|k] + G u[k] + v[k]) = F\,E(\hat{x}[k|k]) + G u[k] + E(v[k]) = F\hat{x}[k|k] + G u[k]$ — State extrapolation

Covariance - $\hat{\Sigma}[k+1|k] = Cov(F\hat{x}[k|k] + G u[k] + v[k]) = F\,Cov(\hat{x}[k|k])F^T + Cov(v[k]) = F\hat{\Sigma}[k|k]F^T + V[k]$ — Covariance extrapolation

$$\hat{x}[k+1|k] \sim \mathcal{N}(\hat{x}[k+1|k], \hat{\Sigma}[k+1|k])$$

2. Update Step:

$$\hat{x}[k+1|k] \rightarrow \hat{x}[k+1|k+1]$$

$$y[k+1] = H\,x[k+1] + w[k+1] \qquad \hat{x}[k+1|k] \sim \mathcal{N}(\hat{x}[k+1|k], \hat{\Sigma}[k+1|k])$$
$$w[k+1] \sim \mathcal{N}(0, W[k+1])$$

$$\hat{x}[k+1|k+1] = K\,y[k+1] + (I - KH)\,\hat{x}[k+1|k]$$

Mean - $E(\hat{x}[k+1|k+1]) = K\,y[k+1] + (I-KH)\,\hat{x}[k+1|k]$

Covariance - $Cov(\hat{x}[k+1|k+1]) = K W K^T + (I-KH)\hat{\Sigma}[k+1|k](I-KH)^T$

Kalman Filter picks K in a way that the covariance is minimized (highest confidence)

$$\frac{d\,Cov(\hat{x}[k+1|k+1])}{dK} = 0$$

$$K = \hat{\Sigma}[k+1|k]H^T\left[H\hat{\Sigma}[k+1|k]H^T + W[k+1]\right]^{-1} \triangleq K[k]$$

K is the Kalman gain

$$\hat{x}[k+1|k+1] = K\,y[k+1] + (I-KH)\,\hat{x}[k+1|k] = \hat{x}[k+1|k] + K(y[k+1] - H\hat{x}[k+1|k])$$

State update equation

$$\hat{\Sigma}[k+1|k+1] = Cov(\hat{x}[k+1|k+1]) = (I - KH)\hat{\Sigma}[k+1|k]$$

Covariance Update Equation

Summary:

Kalman Filter Function (time k):

Prediction step: $\hat{x}[k|k] \rightarrow \hat{x}[k+1|k]$

$$\hat{x}[k+1|k] = F\hat{x}[k|k] + G u[k]$$
$$\hat{\Sigma}[k+1|k] = F\hat{\Sigma}[k|k]F^T + V[k]$$

Update step: $\hat{x}[k+1|k] \rightarrow \hat{x}[k+1|k+1]$

$$K = \hat{\Sigma}[k+1|k]H^T\left[H\hat{\Sigma}[k+1|k]H^T + W[k+1]\right]^{-1}$$
$$\hat{x}[k+1|k+1] = \hat{x}[k+1|k] + K(y[k+1] - H\hat{x}[k+1|k])$$
$$\hat{\Sigma}[k+1|k+1] = (I - KH)\hat{\Sigma}[k+1|k]$$

Algorithm Kalman filter$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$:

1. $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
2. $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$
3. $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$
4. $\mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t)$
5. $\Sigma_t = (I - K_t C_t)\bar{\Sigma}_t$
6. return μ_t, Σ_t

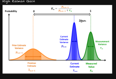
Kalman Gain Intuition:

$$K = \frac{\text{Variance in Estimate}}{\text{Variance in Estimate + Variance in Measurement}}$$

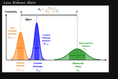
Kalman Gain is close to zero when the measurement uncertainty is high and the estimate uncertainty is low. Hence in the state update equation, we give a significant weight to the estimate and a small weight to the measurement.

Kalman Gain is close to one when the measurement uncertainty is low, and the estimate uncertainty is high. Hence in the state update equation, we give a low weight to the estimate and a significant weight to the measurement.

High Kalman Gain



Low Kalman Gain



Extended Kalman Filter

This is the Kalman filter extended to a non-linear system

$$x[k+1] = f(x[k], u[k]) + v[k]$$
$$y[k] = h(x[k]) + w[k]$$

- $v[k] \sim \mathcal{N}(0, V[k])$: Process noise
- $w[k] \sim \mathcal{N}(0, W[k])$: Measurement noise

1. Prediction Step:

$$\hat{x}[k|k] \rightarrow \hat{x}[k+1|k]$$

Dynamic model -

$$\hat{x}[k+1|k] = f(\hat{x}[k|k], u[k]) + v[k]$$

Linear approximation

$$F = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}[k|k],\,u[k]}$$

$$\hat{x}[k+1|k] = f(\hat{x}[k|k], u[k])$$
$$\hat{\Sigma}[k+1|k] = F\hat{\Sigma}[k|k]F^T + V[k]$$

2. Update Step:

$$\hat{x}[k+1|k] \rightarrow \hat{x}[k+1|k+1]$$
$$y[k+1] = h(x[k+1]) + w[k+1]$$

Linear approximation

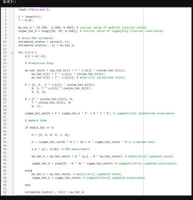
$$H = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}[k+1|k]}$$

$$K = \hat{\Sigma}[k+1|k]H^T\left[H\hat{\Sigma}[k+1|k]H^T + W[k+1]\right]^{-1}$$
$$\hat{x}[k+1|k+1] = \hat{x}[k+1|k] + K(y[k+1] - h(\hat{x}[k+1|k]))$$
$$\hat{\Sigma}[k+1|k+1] = (I - KH)\hat{\Sigma}[k+1|k]$$

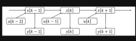
Algorithm Extended Kalman filter$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$:

1. $\bar{\mu}_t = g(u_t, \mu_{t-1})$
2. $\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$
3. $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$
4. $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$
5. $\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$
6. return μ_t, Σ_t

EKF:



Bayes Filter



Markov assumption:

$$p(x[k+1] \mid x_{0:k}, u_{0:k}, y_{1:k}) = p(x[k+1] \mid x[k], u[k]) \quad \text{state transition probability}$$
$$p(y[k] \mid x_{0:k}, u_{0:k-1}, y_{1:k-1}) = p(y[k] \mid x[k]) \quad \text{measurement probability}$$

Bayes Filter can be of 2 types:
- Parametric Filters - Gaussian beliefs - only gaussian beliefs
- Non-Parametric Filters - Represent arbitrary probability distributions

1. Prediction Step:

$$p(x[k] \mid u_{k-1}, y_k) \rightarrow p(x[k+1] \mid u_k, y_k) = \overline{bel}(x[k+1])$$

$$p(x[k+1] \mid u_k, y_k) = \int p(x[k+1] \mid x[k], u_k, y_k)\, p(x[k] \mid u_k, y_k)\, dx[k]$$
$$= \int p(x[k+1] \mid x[k], u[k])\, p(x[k] \mid u_{k-1}, y_k)\, dx[k]$$
$$= \int p(x[k+1] \mid x[k], u[k])\, bel(x[k])\, dx[k]$$

where I'm going, where I am

2. Update Step:

$$p(x[k+1] \mid u_k, y_k) \rightarrow p(x[k+1] \mid u_k, y_{k+1}) = bel(x[k+1])$$

$$p(x[k+1] \mid u_k, y_{k+1}) = \frac{p(y[k+1] \mid x[k+1], u_k, y_k)\, p(x[k+1] \mid u_k, y_k)}{p(y[k+1] \mid u_k, y_k)}$$

$$bel(x[k+1]) = \frac{p(y[k+1] \mid x[k+1])\, p(x[k+1] \mid u_k, y_k)}{p(y[k+1] \mid u_k, y_k)}$$

$$\propto p(y[k+1] \mid x[k+1])\, p(x[k+1] \mid u_k, y_k)$$

$$\propto p(y[k+1] \mid x[k+1])\, \overline{bel}(x[k+1])$$

Summary:

Bayes Filter Function (time k):

Prediction step: $p(x[k] \mid u_{k-1}, y_k) \rightarrow p(x[k+1] \mid u_k, y_k) = \overline{bel}(x[k+1])$

$$p(x[k+1] \mid u_k, y_k) = \int p(x[k+1] \mid x[k], u[k])\, bel(x[k])\, dx[k]$$

Update step: $p(x[k+1] \mid u_k, y_k) \rightarrow p(x[k+1] \mid u_k, y_{k+1}) = bel(x[k+1])$

$$p(x[k+1] \mid u_k, y_{k+1}) = \frac{p(y[k+1] \mid x[k+1])\, p(x[k+1] \mid u_k, y_k)}{p(y[k+1] \mid u_k, y_k)}$$

$$\propto p(y[k+1] \mid x[k+1])\, \overline{bel}(x[k+1])$$

Particle Filter

Particles:

$$X[k] = \{\hat{x}^{(1)}[k], \hat{x}^{(2)}[k], \ldots, \hat{x}^{(M)}[k]\}$$

A particle $\hat{x}^{(m)}[k]$, $1 \leq m \leq M$, is a concrete instantiation of the state at time k

The particles represent the state

$$\hat{x}^{(m)}[k] \sim p(x[k] \mid u_{k-1}, y_k)$$

The denser a subregion of the state space is populated by samples, the more likely it is that the true state falls into this region.

Particle_filter (X[k-1], u[k-1], y[k]):

- From m = 1 to M do
  - Sample
    $$\hat{x}^{(m)}[k] \sim p(\hat{x}^{(m)}[k] \mid u[k-1], \hat{x}^{(m)}[k-1])$$
  - Weights
    $$w^{(m)}[k] = p(y[k] \mid \hat{x}^{(m)}[k])$$
- $X[k] = \phi$
- From m = 1 to M do
  - Draw i with probability $\propto w^{(i)}[k]$
  - Add $\hat{x}^{(i)}[k]$ to X[k]
- Return X[k]

Particle Filter:

```
% iterate particle filter in this loop
for k = 2:numSteps

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %              put particle filter prediction step here              %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    T = t(k) - t(k-1);

    for m = 1:numParticles
        x_hat = particles(:, m);
        u_k = u(:, k-1);

        v = mvnrnd([0,0],V)';
        v1 = v(1);
        v2 = v(2);

        x_hat_next = [x_hat(1) + T * (u_k(1) + v1) * cos(x_hat(3));
                      x_hat(2) + T * (u_k(1) + v1) * sin(x_hat(3));
                      x_hat(3) + T * (u_k(2) + v2)];

        particles(:, m) = x_hat_next;
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %                put particle filter update step here                %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    y_k = y(:, k);

    for m = 1:numParticles
        expectedMeasurements = [sqrt((particles(1,m) - b1(1))^2 + (particles(2,m) - b1(2))^2);
                                sqrt((particles(1,m) - b2(1))^2 + (particles(2,m) - b2(2))^2)];

        w = y_k - expectedMeasurements;

        weights(m) = exp(-0.5 * (w' / measurementNoiseCov * w)) ./ sqrt((2 * pi) * det(measurementNoiseCov));

        disp(weights(m))

    end

    % weight particles

    weights = weights / sum(weights);

    % resample particles

    indices = randsample(1:numParticles, numParticles, true, weights);
    particles = particles(:, indices);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % plot particle cloud, robot, robot estimate, and robot trajectory here %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Estimate state
    estimated_state = sum(particles .* weights, 2);
```

Differential Drive Robot

$$\text{State } x = \begin{bmatrix} x_r \\ y_r \\ \theta \end{bmatrix} \qquad \text{Input } u = \begin{bmatrix} v_f \\ \omega \end{bmatrix}$$

Motion model

$$x[k+1] = \begin{bmatrix} x_1[k] + (u_1[k] + v_1[k])\cos(x_3[k])T - v_3\sin(x_3[k])T \\ x_2[k] + (u_1[k] + v_1[k])\sin(x_3[k])T + v_3\cos(x_3[k])T \\ x_3[k] + (u_2[k] + v_2[k])T \end{bmatrix}$$

$$v_1 \sim \mathcal{N}(0, \sigma_{v_f}) \qquad v_2 \sim \mathcal{N}(0, \sigma_\omega) \qquad v_3 \sim \mathcal{N}(0, \sigma_{v_\perp})$$

Measurements are distances to beacons at known locations:
- beacon 1 at $(x_{b,1}, y_{b,1})$
- beacon 2 at $(x_{b,2}, y_{b,2})$

Sensor model

$$y[k] = \begin{bmatrix} \sqrt{(x_1[k] - x_{b,1})^2 + (x_2[k] - y_{b,1})^2} + w_1 \\ \sqrt{(x_1[k] - x_{b,2})^2 + (x_2[k] - y_{b,2})^2} + w_2 \end{bmatrix}$$

$$w_1 \sim \mathcal{N}(0, \sigma_1) \qquad w_2 \sim \mathcal{N}(0, \sigma_2)$$

For particle filter:

Prediction:
- For every particle
  - Use random number generator to generate process noises
  - Move the particle using the motion model

Update weights
- For every particle
  - For each measurement $y_i[k+1] = h(x[k+1]) + w_i$

$$y_i[k+1] \sim \mathcal{N}(h(\hat{x}^{(m)}[k+1]), \sigma_i)$$

$$p(y_i[k+1] \mid \hat{x}^{(m)}[k+1]) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left[\frac{(y_i[k+1] - h(\hat{x}^{(m)}[k+1]))^2}{2\sigma_i^2}\right] = \text{weight}(m)$$

  - For multiple measurements:

$$w^{(m)}[k+1] = p(y[k+1] \mid \hat{x}^{(m)}[k+1]) = \prod_{i=1}^{p} p(y_i[k+1] \mid \hat{x}^{(m)}[k+1])$$

Resample particles:
- Normalize $w^{(m)}[k+1]$ with the sum of weights
- Make a cumulative weight vector CW

$$CW^{(m)} = \sum_{i=1}^{m} w^{(i)}[k+1]$$

- $X[k+1] = \phi$
- For m = 1 to M do
  - Use a random number generator to generate a uniform random number $z \in [0,1]$
  - Find the smallest i such that $CW^{(i)} > z$
  - Add $\hat{x}^{(i)}[k+1]$ to X[k+1]