Autonomous Navigation for Flying Robots

Lecture 6.2: Kalman Filter

Jürgen Sturm
Technische Universität München
Motivation

- Bayes filter is a useful tool for state estimation
- Histogram filter with grid representation is not very efficient
- How can we represent the state more efficiently?
Kalman Filter

- Bayes filter with continuous states
- State represented with a normal distribution
- Developed in the late 1950’s
- Kalman filter is very efficient (only requires a few matrix operations per time step)
- Applications range from economics, weather forecasting, satellite navigation to robotics and many more
Normal Distribution

- **Univariate normal distribution** \( X \sim \mathcal{N}(\mu, \sigma^2) \)

\[
p(X = x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{1}{2} \frac{(x - \mu)^2}{\sigma^2}\right)
\]

- 68% of mass within 1sd
- 99% of mass within 3sd

Jürgen Sturm  
Autonomous Navigation for Flying Robots
Normal Distribution

- Multivariate normal distribution \( X \sim \mathcal{N}(\mu, \Sigma) \)
- Mean \( \mu \in \mathbb{R}^n \)
- Covariance \( \Sigma \in \mathbb{R}^{n \times n} \)
- Probability density function

\[
p(X = x) = \mathcal{N}(x; \mu, \Sigma)
= \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)^\top \Sigma^{-1}(x - \mu)\right)
\]
2D Example

Probability density function (pdf)

Isolines (contour plot)

$\mathbf{p}(\mathbf{x}, \mathbf{y})$

68%

95%
Properties of Normal Distributions

- Linear transformation → remains Gaussian
  \[ X \sim \mathcal{N}(\mu, \Sigma), \ Y \sim AX + B \]
  \[ \Rightarrow Y \sim \mathcal{N}(A\mu + B, A\Sigma A^\top) \]

- Intersection of two Gaussians → remains Gaussian
  \[ X_1 \sim \mathcal{N}(\mu_1, \Sigma_1), \ X_2 \sim \mathcal{N}(\mu_2, \Sigma_2) \]
  \[ \Rightarrow p(X_1)p(X_2) = \mathcal{N}\left(\frac{\Sigma_2}{\Sigma_1 + \Sigma_2} \mu_1 + \frac{\Sigma_1}{\Sigma_1 + \Sigma_2} \mu_2, \frac{1}{\Sigma_1^{-1} + \Sigma_2^{-1}}\right) \]
Linear Process Model

- Consider a time-discrete stochastic process (Markov chain)
Linear Process Model

- Consider a time-discrete stochastic process
- Represent the estimated state (belief) by a Gaussian

\[ x_t \sim \mathcal{N}(\mu_t, \Sigma_t) \]
Linear Process Model

- Consider a time-discrete stochastic process
- Represent the estimated state (belief) by a Gaussian
  \[ x_t \sim \mathcal{N}(\mu_t, \Sigma_t) \]
- Assume that the system evolves linearly over time, then
  \[ x_t = Ax_{t-1} \]
Linear Process Model

- Consider a time-discrete stochastic process
- Represent the estimated state (belief) by a Gaussian
  \[ x_t \sim \mathcal{N}(\mu_t, \Sigma_t) \]
- Assume that the system evolves linearly over time and depends linearly on the controls
  \[ x_t = Ax_{t-1} + Bu_t \]
Linear Process Model

- Consider a time-discrete stochastic process
- Represent the estimated state (belief) by a Gaussian

\[ x_t \sim \mathcal{N}(\mu_t, \Sigma_t) \]

- Assume that the system evolves linearly over time, depends linearly on the controls, and has zero-mean, normally distributed process noise

\[ x_t = Ax_{t-1} + Bu_t + \epsilon_t \]

with \[ \epsilon_t \sim \mathcal{N}(0, Q) \]
Further, assume we make observations that depend linearly on the state

\[ z_t = C x_t \]
Linear Observations

- Further, assume we make observations that depend linearly on the state and that are perturbed by zero-mean, normally distributed observation noise

\[ z_t = Cx_t + \delta_t \]

with \( \delta_t \sim \mathcal{N}(0, \mathbf{R}) \)
Kalman Filter

Estimates the state $x_t$ of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$x_t = Ax_{t-1} + Bu_t + \epsilon_t$$

and (linear) measurements of the state

$$z_t = Cx_t + \delta_t$$

with $\delta_t \sim \mathcal{N}(0, R)$ and $\epsilon_t \sim \mathcal{N}(0, Q)$
Variables and Dimensions

- State $x \in \mathbb{R}^n$
- Controls $u \in \mathbb{R}^l$
- Observations $z \in \mathbb{R}^k$
- Process equation

$$x_t = A x_{t-1} + B u_t + \epsilon_t$$

- Measurement equation

$$z_t = C x_t + \delta_t$$
Kalman Filter

- Initial belief is Gaussian
  \[ \text{Bel}(x_0) = \mathcal{N}(x_0; \mu_0, \Sigma_0) \]

- Next state is also Gaussian (linear transformation)
  \[ x_t \sim \mathcal{N}(Ax_{t-1} + Bu_t, Q) \]

- Observations are also Gaussian
  \[ z_t \sim \mathcal{N}(Cx_t, R) \]
Remember: Bayes Filter Algorithm

For each time step, do

1. Apply motion model

\[
    \overline{Bel}(x_t) = \int p(x_t \mid x_{t-1}, u_t)Bel(x_{t-1})dx_{t-1}
\]

2. Apply sensor model

\[
    Bel(x_t) = \eta p(z_t \mid x_t)\overline{Bel}(x_t)
\]
From Bayes Filter to Kalman Filter

For each time step, do

1. Apply motion model

\[
\overline{\text{Bel}}(x_t) = \int \underbrace{p(x_t \mid x_{t-1}, u_t)}_{\mathcal{N}(x_t; Ax_{t-1}+Bu_t, Q)} \underbrace{\text{Bel}(x_{t-1})}_{\mathcal{N}(x_{t-1}; \mu_{t-1}, \Sigma_{t-1})} \, dx_{t-1}
\]
From Bayes Filter to Kalman Filter

For each time step, do

1. Apply motion model

\[
\text{Bel}(x_t) = \int p(x_t \mid x_{t-1}, u_t) \underbrace{\text{Bel}(x_{t-1})}_{\mathcal{N}(x_{t-1}; \mu_{t-1}, \Sigma_{t-1})} \, dx_{t-1}
\]

\[
= \mathcal{N}(x_t; A\mu_{t-1} + Bu, A\Sigma A^\top + Q)
\]

\[
= \mathcal{N}(x_t; \bar{\mu}_t, \bar{\Sigma}_t)
\]
From Bayes Filter to Kalman Filter

For each time step, do

2. Apply sensor model

\[
\text{Bel}(x_t) = \eta \underbrace{p(z_t \mid x_t)}_{\mathcal{N}(z_t; Cx_t, R)} \underbrace{\text{Bel}(x_t)}_{\mathcal{N}(x_t; \mu_t, \Sigma_t)}
\]

\[
= \mathcal{N}(x_t; \mu_t + K_t(z_t - C\mu), (I - K_tC)\Sigma)
\]

\[
= \mathcal{N}(x_t; \mu_t, \Sigma_t)
\]

with \( K_t = \Sigma_t C^T (C\Sigma_t C^T + R)^{-1} \) (Kalman gain)
Kalman Filter Algorithm

For each time step, do

1. **Apply motion model (prediction step)**
   \[
   \tilde{\mu}_t = A\mu_{t-1} + Bu_t \\
   \tilde{\Sigma}_t = A\Sigma A^\top + Q
   \]

2. **Apply sensor model (correction step)**
   \[
   \mu_t = \tilde{\mu}_t + K_t(z_t - C\tilde{\mu}_t) \\
   \Sigma_t = (I - K_t C)\tilde{\Sigma}_t
   \]
   with \( K_t = \tilde{\Sigma}_t C^\top (C\tilde{\Sigma}_t C^\top + R)^{-1} \)

See Probabilistic Robotics for full derivation (Chapter 3)
Complexity

- Highly efficient: Polynomial in the measurement dimensionality $k$ and state dimensionality $n$:

$$O(k^{2.376} + n^2)$$

- Optimal for linear Gaussian systems!
- Most robotics systems are **nonlinear**!
Nonlinear Dynamical Systems

- Most realistic robotic problems involve nonlinear functions

- Motion function \( x_t = g(u_t, x_{t-1}) \)

- Observation function \( z_t = h(x_t) \)

- Can we **linearize** these functions?
Taylor Expansion

- **Idea: Linearize both functions**

- **Motion function**

\[
g(x_{t-1}, u_t) \approx g(\mu_{t-1}, u_t) + \frac{\partial g(x, u_t)}{\partial x} \bigg|_{x=\mu_{t-1}} (x_{t-1} - \mu_{t-1})
\]

\[
= g(\mu_{t-1}, u_t) + G_t(x_{t-1} - \mu_{t-1})
\]

- **Observation function**

\[
h(x_t) \approx h(\bar{\mu}_t) + \frac{\partial h(x, u_t)}{\partial x} \bigg|_{x=\bar{\mu}_t} (x_t - \bar{\mu}_t)
\]

\[
= h(\bar{\mu}_t) + H_t(x_t - \bar{\mu}_t)
\]
Extended Kalman Filter (EKF)

For each time step, do

1. Apply motion model (prediction step)
   \[
   \tilde{\mu}_t = g(\mu_{t-1}, u_t)
   \]
   \[
   \tilde{\Sigma}_t = G_t \Sigma G_t^T + Q \quad \text{with} \quad G_t = \frac{\partial g(x, u_t)}{\partial x}
   \]

2. Apply sensor model (correction step)
   \[
   \mu_t = \tilde{\mu}_t + K_t (z_t - h(\tilde{\mu}_t))
   \]
   \[
   \Sigma_t = (I - K_t H_t) \tilde{\Sigma}_t
   \]

with \(K_t = \tilde{\Sigma}_t H_t^T (H_t \tilde{\Sigma}_t H_t^T + R)^{-1}\) and \(H_t = \frac{\partial h(x, u_t)}{\partial x}\)
Lessons Learned

- Kalman filter
- Linearization of sensor and motion model
- Extended Kalman filter

- Next: Example in 2D