Computer Vision Group
Prof. Daniel Cremers

Technische Universität München

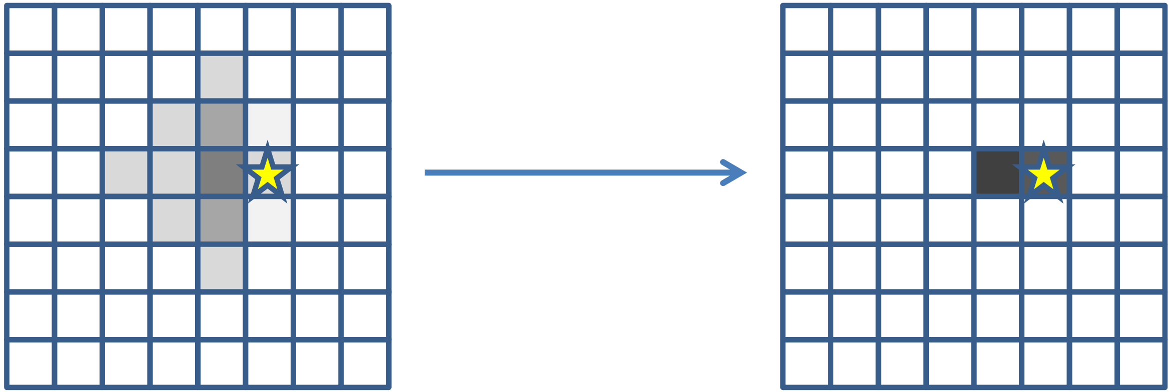
# Autonomous Navigation for Flying Robots

## 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

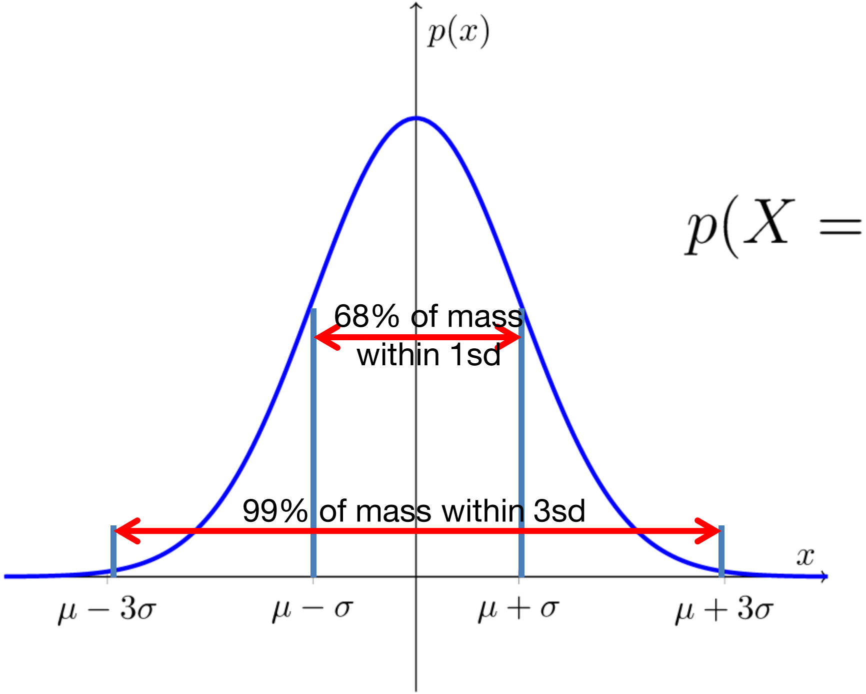
- Bayes filter with continuous states
- State represented with a normal distribution
- Developed in the late 1950's
- Kalman filter is very efficient (only requires a few matrix operations per time step)
- Applications range from economics, weather forecasting, satellite navigation to robotics and many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

# Normal Distribution

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

mean

variance (squared std dev)



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

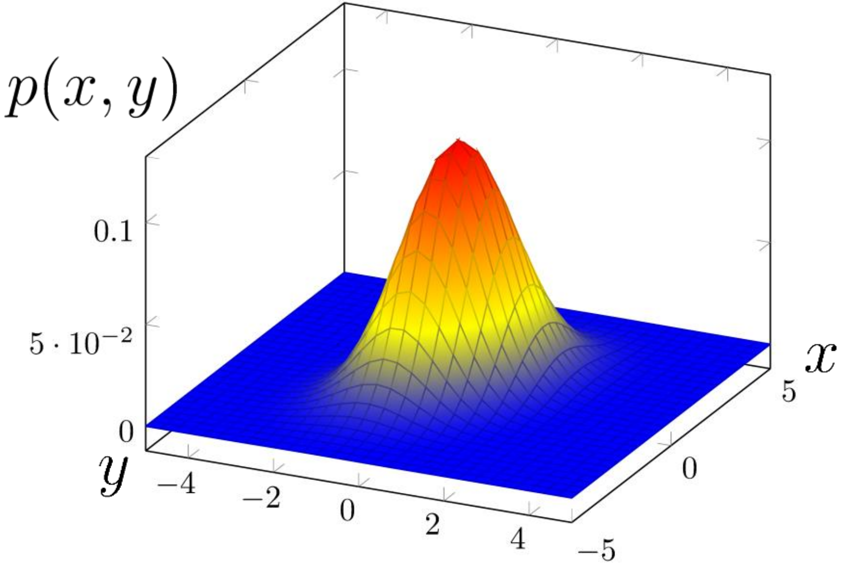
# Normal Distribution

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

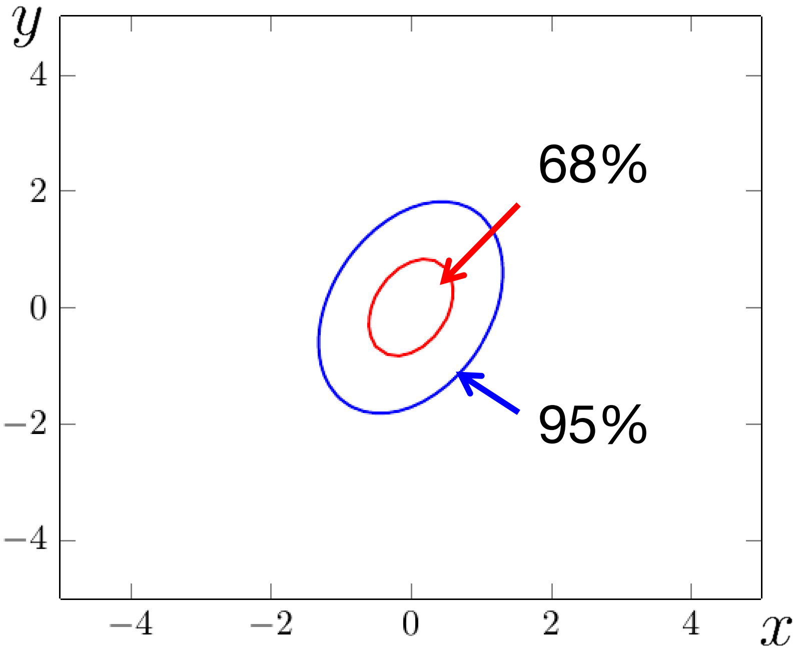
$$
\begin{aligned}
p(\mathbf{X} = \mathbf{x}) &= \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \\
&= \frac{1}{(2\pi)^{n/2} \, |\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right)
\end{aligned}
$$

# 2D Example

Probability density function (pdf)



Isolines (contour plot)



# Properties of Normal Distributions

- Linear transformation → remains Gaussian

$$\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \mathbf{Y} \sim \mathbf{A}\mathbf{X} + \mathbf{B}$$

$$\Rightarrow \mathbf{Y} \sim \mathcal{N}(\mathbf{A}\boldsymbol{\mu} + \mathbf{B}, \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^\top)$$

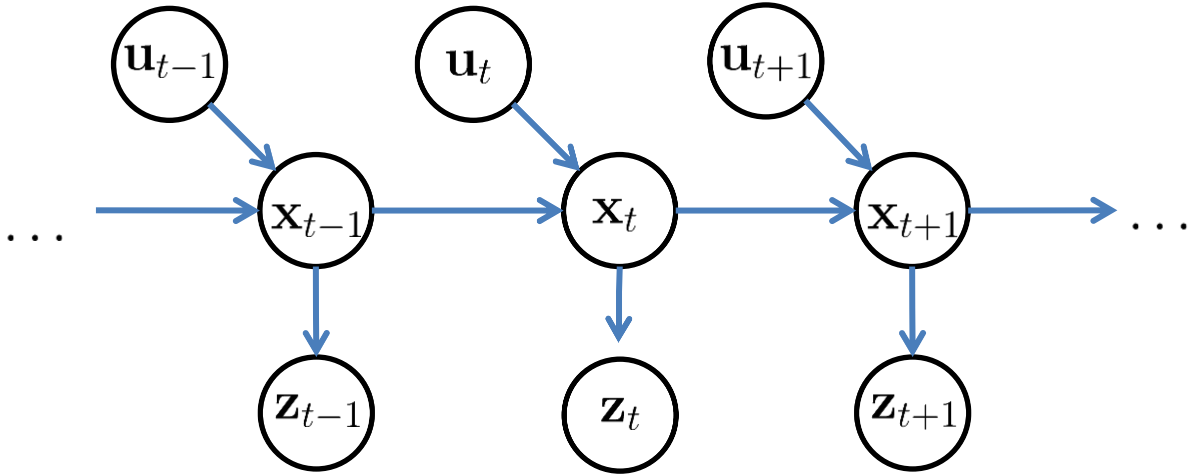
- Intersection of two Gaussians → remains Gaussian

$$\mathbf{X}_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1), \mathbf{X}_2 \sim \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$$

$$\Rightarrow p(\mathbf{X}_1)p(\mathbf{X}_2) = \mathcal{N}\left(\frac{\boldsymbol{\Sigma}_2}{\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2}\boldsymbol{\mu}_1 + \frac{\boldsymbol{\Sigma}_1}{\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2}\boldsymbol{\mu}_2, \frac{1}{\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1}}\right)$$

# 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

$$\mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$$

# Linear Process Model

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

$$\mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$$

- Assume that the system evolves linearly over time, then

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1}$$

# Linear Process Model

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

$$\mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$$

- Assume that the system evolves linearly over time and depends linearly on the controls

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{u}_t$$

# Linear Process Model

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

$$\mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$$

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

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{u}_t + \boldsymbol{\epsilon}_t$$

with $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$

# Linear Observations

- Further, assume we make observations that depend linearly on the state

$$\mathbf{z}_t = \mathbf{C}\mathbf{x}_t$$

# Linear Observations

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

$$\mathbf{z}_t = \mathbf{C}\mathbf{x}_t + \boldsymbol{\delta}_t$$

with $\boldsymbol{\delta}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$

# Kalman Filter

Estimates the state $\mathbf{x}_t$ of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{u}_t + \boldsymbol{\epsilon}_t$$

and (linear) measurements of the state

$$\mathbf{z}_t = \mathbf{C}\mathbf{x}_t + \boldsymbol{\delta}_t$$

with $\boldsymbol{\delta}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ and $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$

# Variables and Dimensions

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

$$\mathbf{x}_t = \underbrace{\mathbf{A}}_{n \times n} \mathbf{x}_{t-1} + \underbrace{\mathbf{B}}_{n \times l} \mathbf{u}_t + \boldsymbol{\epsilon}_t$$

- Measurement equation

$$\mathbf{z}_t = \underbrace{\mathbf{C}}_{n \times k} \mathbf{x}_t + \boldsymbol{\delta}_t$$

# Kalman Filter

- Initial belief is Gaussian

$$\mathrm{Bel}(\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_0; \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$$

- Next state is also Gaussian (linear transformation)

$$\mathbf{x}_t \sim \mathcal{N}(\mathbf{A}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{u}_t, \mathbf{Q})$$

- Observations are also Gaussian

$$\mathbf{z}_t \sim \mathcal{N}(\mathbf{C}\mathbf{x}_t, \mathbf{R})$$

# Remember: Bayes Filter Algorithm

For each time step, do

1. Apply motion model

$$\overline{\mathrm{Bel}}(\mathbf{x}_t) = \int p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t)\mathrm{Bel}(\mathbf{x}_{t-1})\mathrm{d}\mathbf{x}_{t-1}$$

2. Apply sensor model

$$\mathrm{Bel}(\mathbf{x}_t) = \eta p(\mathbf{z}_t \mid \mathbf{x}_t)\overline{\mathrm{Bel}}(\mathbf{x}_t)$$

# From Bayes Filter to Kalman Filter

For each time step, do

1. Apply motion model

$$\overline{\mathrm{Bel}}(\mathbf{x}_t) = \int \underbrace{p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t)}_{\mathcal{N}(\mathbf{x}_t;\mathbf{A}\mathbf{x}_{t-1}+\mathbf{B}\mathbf{u}_t,\mathbf{Q})} \underbrace{\mathrm{Bel}(\mathbf{x}_{t-1})}_{\mathcal{N}(\mathbf{x}_{t-1};\boldsymbol{\mu}_{t-1},\boldsymbol{\Sigma}_{t-1})} \, d\mathbf{x}_{t-1}$$

# From Bayes Filter to Kalman Filter

For each time step, do

1. Apply motion model

$$\overline{\text{Bel}}(\mathbf{x}_t) = \int \underbrace{p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t)}_{\mathcal{N}(\mathbf{x}_t;\mathbf{A}\mathbf{x}_{t-1}+\mathbf{B}\mathbf{u}_t,\mathbf{Q})} \underbrace{\text{Bel}(\mathbf{x}_{t-1})}_{\mathcal{N}(\mathbf{x}_{t-1};\boldsymbol{\mu}_{t-1},\boldsymbol{\Sigma}_{t-1})} \, d\mathbf{x}_{t-1}$$

$$= \mathcal{N}(\mathbf{x}_t; \mathbf{A}\boldsymbol{\mu}_{t-1} + \mathbf{B}\mathbf{u}_t, \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^\top + \mathbf{Q})$$

$$= \mathcal{N}(\mathbf{x}_t; \bar{\boldsymbol{\mu}}_t, \bar{\boldsymbol{\Sigma}}_t)$$

# From Bayes Filter to Kalman Filter

For each time step, do

2. Apply sensor model

$$\mathrm{Bel}(\mathbf{x}_t) = \eta\ \underbrace{p(\mathbf{z}_t \mid \mathbf{x}_t)}_{\mathcal{N}(\mathbf{z}_t;\mathbf{C}\mathbf{x}_t,\mathbf{R})}\ \underbrace{\overline{\mathrm{Bel}}(\mathbf{x}_t)}_{\mathcal{N}(\mathbf{x}_t;\bar{\boldsymbol{\mu}}_t,\bar{\mathbf{\Sigma}}_t)}$$

$$= \mathcal{N}\left(\mathbf{x}_t; \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t(\mathbf{z}_t - \mathbf{C}\bar{\boldsymbol{\mu}}), (\mathbf{I} - \mathbf{K}_t\mathbf{C})\bar{\mathbf{\Sigma}}\right)$$

$$= \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_t, \mathbf{\Sigma}_t)$$

with $\mathbf{K}_t = \bar{\mathbf{\Sigma}}_t\mathbf{C}^\top(\mathbf{C}\bar{\mathbf{\Sigma}}_t\mathbf{C}^\top + \mathbf{R})^{-1}$ (Kalman gain)

# Kalman Filter Algorithm

For each time step, do

1. Apply motion model (prediction step)

$$\bar{\boldsymbol{\mu}}_t = \mathbf{A}\boldsymbol{\mu}_{t-1} + \mathbf{B}\mathbf{u}_t$$

$$\bar{\Sigma}_t = \mathbf{A}\Sigma\mathbf{A}^\top + \mathbf{Q}$$

2. Apply sensor model (correction step)

$$\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t(\mathbf{z}_t - \mathbf{C}\bar{\boldsymbol{\mu}}_t)$$

$$\Sigma_t = (\mathbf{I} - \mathbf{K}_t\mathbf{C})\bar{\Sigma}_t$$

with $\mathbf{K}_t = \bar{\Sigma}_t\mathbf{C}^\top(\mathbf{C}\bar{\Sigma}_t\mathbf{C}^\top + \mathbf{R})^{-1}$

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 $\mathbf{x}_t = g(\mathbf{u}_t, \mathbf{x}_{t-1})$
- Observation function $\mathbf{z}_t = h(\mathbf{x}_t)$
- Can we **linearize** these functions?

# Taylor Expansion

- **Idea: Linearize both functions**
- Motion function

$$g(\mathbf{x}_{t-1}, \mathbf{u}_t) \approx g(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t) + \left.\frac{\partial g(\mathbf{x}, \mathbf{u}_t)}{\partial \mathbf{x}}\right|_{\mathbf{x}=\boldsymbol{\mu}_{t-1}} (\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})$$

$$= g(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t) + \mathbf{G}_t(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})$$

- Observation function

$$h(\mathbf{x}_t) \approx h(\bar{\boldsymbol{\mu}}_t) + \left.\frac{\partial h(\mathbf{x}, \mathbf{u}_t)}{\partial \mathbf{x}}\right|_{\mathbf{x}=\bar{\boldsymbol{\mu}}_t} (\mathbf{x}_t - \bar{\boldsymbol{\mu}}_t)$$

$$= h(\bar{\boldsymbol{\mu}}_t) + \mathbf{H}_t(\mathbf{x}_t - \bar{\boldsymbol{\mu}}_t)$$

# Extended Kalman Filter (EKF)

For each time step, do

1. Apply motion model (prediction step)

$$\bar{\boldsymbol{\mu}}_t = g(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t)$$

$$\bar{\boldsymbol{\Sigma}}_t = \mathbf{G}_t \boldsymbol{\Sigma} \mathbf{G}_t^\top + \mathbf{Q} \quad \text{with } \mathbf{G}_t = \left.\frac{\partial g(\mathbf{x}, \mathbf{u}_t)}{\partial \mathbf{x}}\right|_{\mathbf{x}=\boldsymbol{\mu}_{t-1}}$$

2. Apply sensor model (correction step)

$$\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t(\mathbf{z}_t - h(\bar{\boldsymbol{\mu}}_t))$$

$$\boldsymbol{\Sigma}_t = (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t)\bar{\boldsymbol{\Sigma}}_t$$

with $\mathbf{K}_t = \bar{\boldsymbol{\Sigma}}_t \mathbf{H}_t^\top (\mathbf{H}_t \bar{\boldsymbol{\Sigma}}_t \mathbf{H}_t^\top + \mathbf{R})^{-1}$ and $\mathbf{H}_t = \left.\frac{\partial h(\mathbf{x}, \mathbf{u}_t)}{\partial \mathbf{x}}\right|_{\mathbf{x}=\bar{\boldsymbol{\mu}}_t}$

# Lessons Learned

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

- Next: Example in 2D