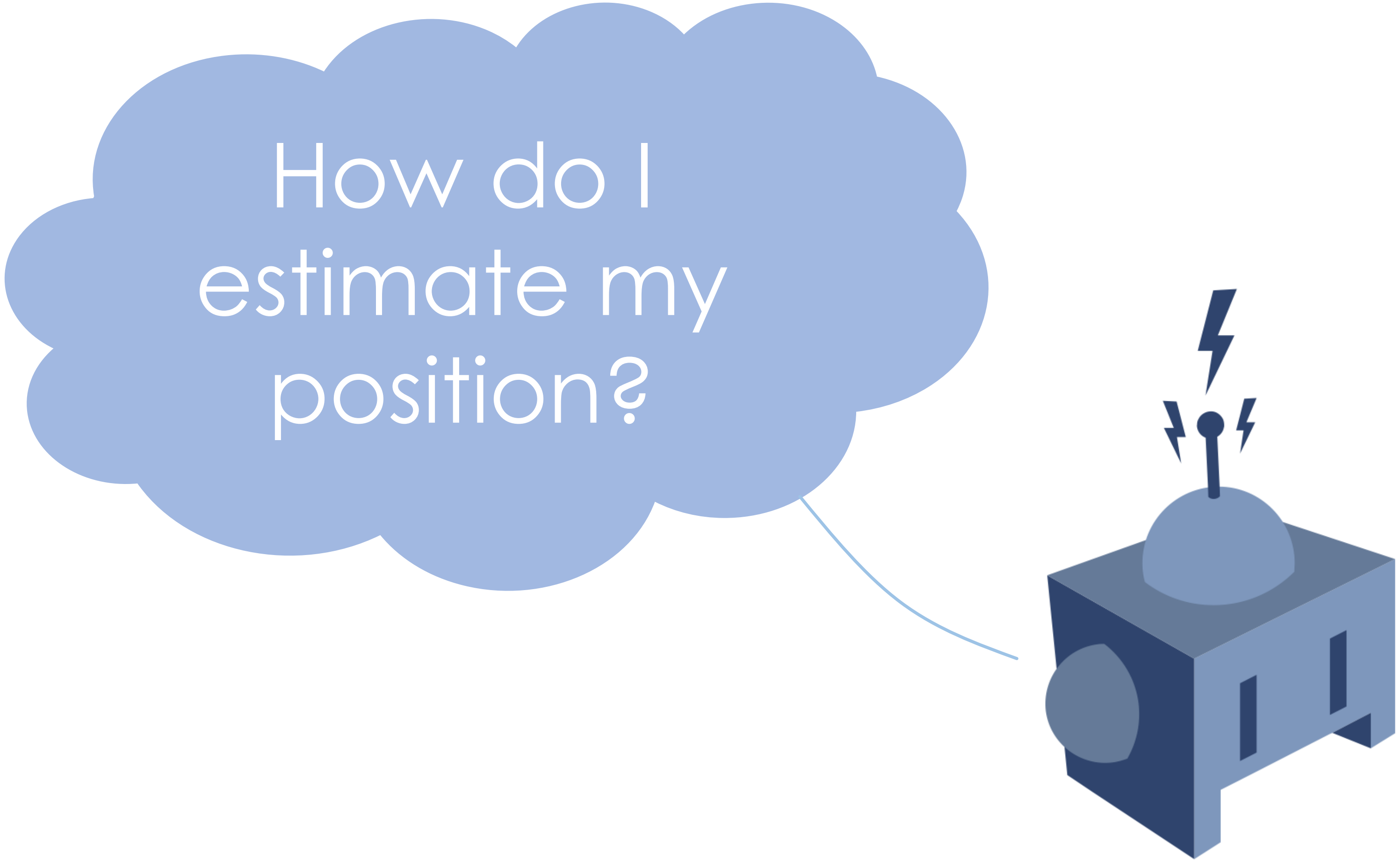
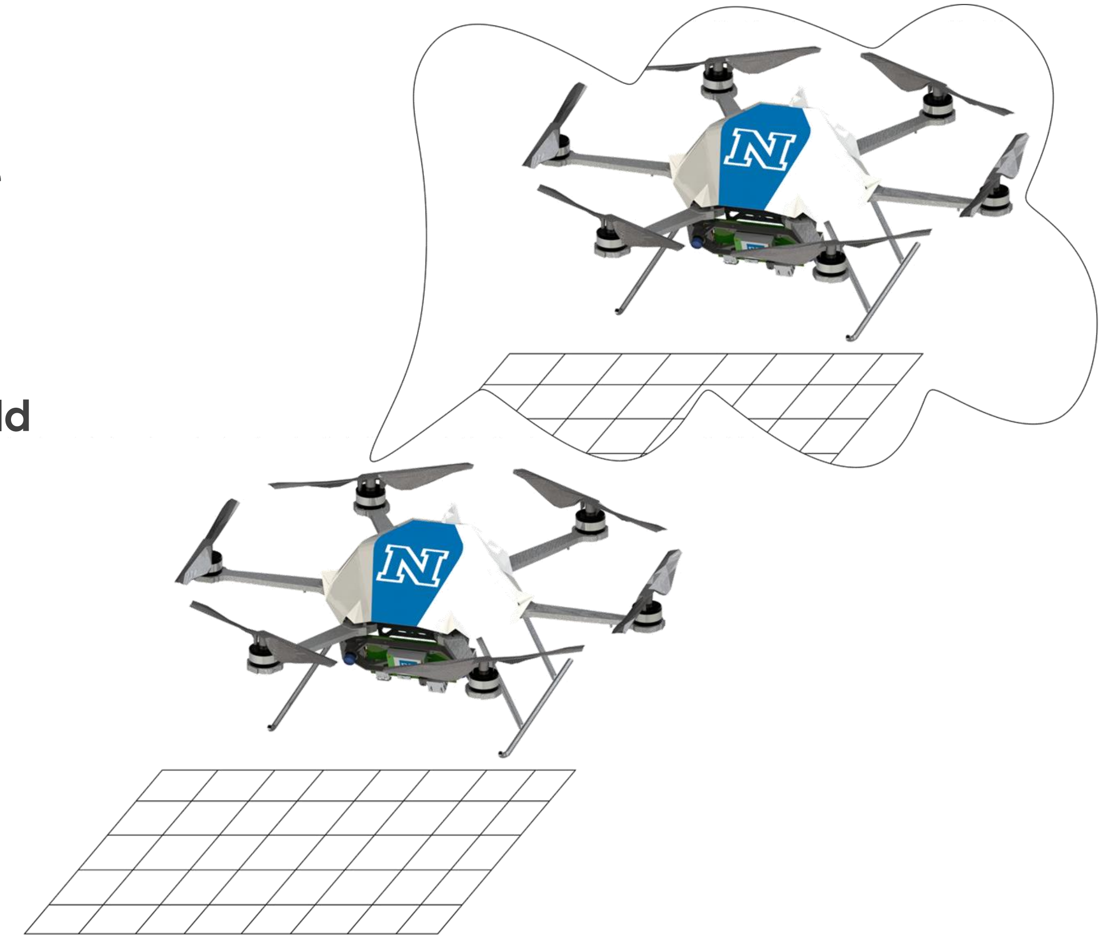
BadgerWorks Lectures

Topic: State Estimation


How do I estimate my position?

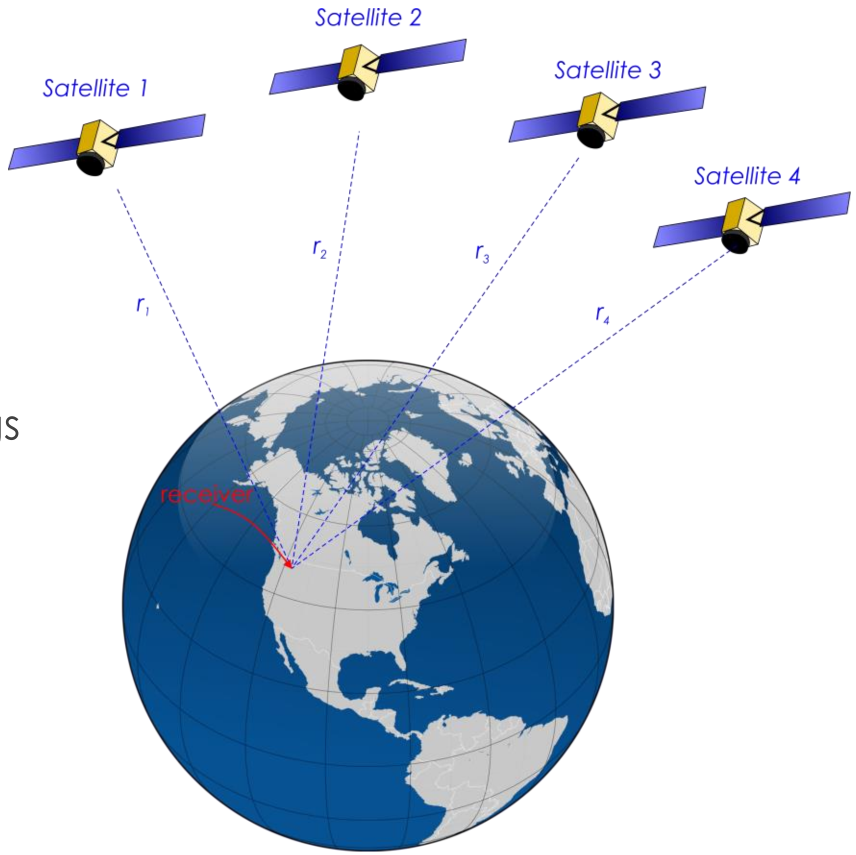
World state (or system state)

- *Belief state:*
  - **Our belief/estimate of the world state**
- *World state:*
  - **Real state of the robot in the real world**



State Estimation

- **What parts of the world state are (most) relevant for a flying robot?**
  - **Position**
  - **Velocity**
  - **Orientation**
  - **Attitude rate**
  - Obstacles
  - Map
  - Positions and intentions of other robots/human beings
  - …



State Estimation

- **Cannot observe world state directly – no sensor tells us where we really are but rather some measurements related to that (and noisy!)**
- **Need to estimate the world state**
  - **But How?**
    - Infer world state from sensor data
    - Infer world state from executed motions/actions

Sensor Model

- Robot perceives the environment through its sensors:

$$\mathbf{z} = h(\mathbf{x})$$

  - Where $\mathbf{z}$ is **the sensor reading**, $\mathbf{h}$ is the **world state**.

- **Goal:** Infer the state of the world from sensor readings.

$$\mathbf{x} = h^{-1}(\mathbf{z})$$

Motion Model

- Robot executes an action (or control) $\mathbf{u}$
  - e.g: move forward at 1m/s

- Update **belief state** according to the **motion model**:

$$\mathbf{x}' = g(\mathbf{x}, \mathbf{u})$$

  - Where $\mathbf{x}'$ is the current state and $\mathbf{x}$ is the previous state.

Probabilistic Robotics

- Sensor observations are noisy, partial, potentially missing.
- All models are partially wrong and incomplete.
- Usually we have prior knowledge.

Probabilistic Robotics

- Probabilistic sensor models: $\mathbf{p}(\mathbf{z}|\mathbf{x})$
- Probabilistic motion models: $\mathbf{p}(\mathbf{x}'|\mathbf{x}, \mathbf{u})$
- Fuse data between multiple sensors (multi-modal):

$$\mathbf{p}(\mathbf{x}|\mathbf{z}_{GPS}, \mathbf{z}_{BARO}, \mathbf{z}_{IMU})$$

- Fuse data over time (filtering):

$$\mathbf{p}(\mathbf{x}|\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_t)$$

$$\mathbf{p}(\mathbf{x}|\mathbf{z}_1, \mathbf{u}_1, \mathbf{z}_2, \mathbf{u}_2, \dots, \mathbf{z}_t, \mathbf{u}_5)$$

BadgerWorks Lectures

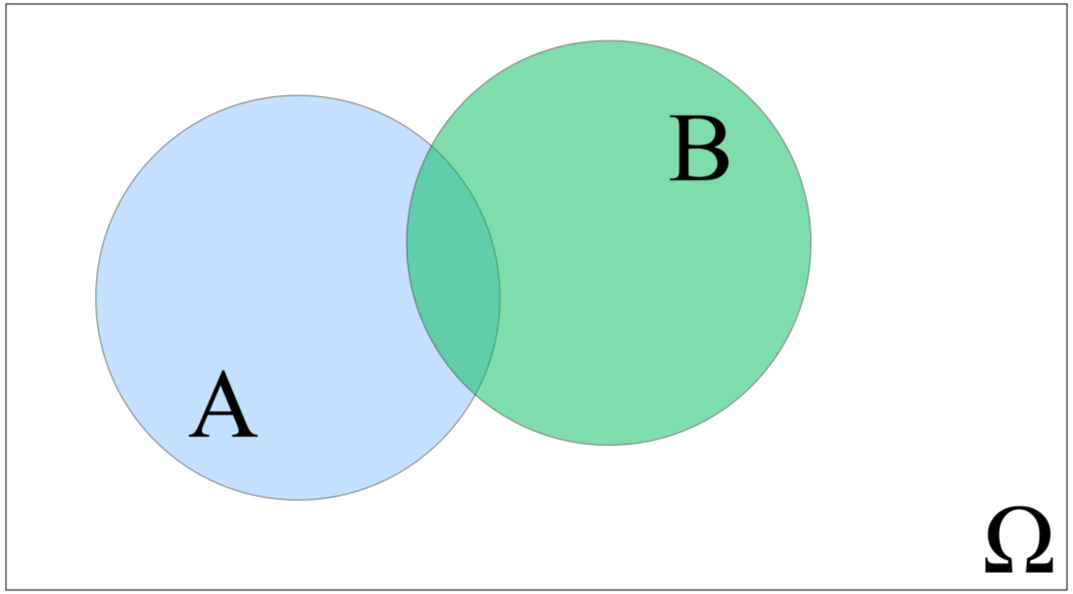
Topic: State Estimation – Recap on Probabilities

Probability theory

- **Random experiment** that can produce a number of outcomes, e.g. a rolling dice.
- Sample space, e.g.: $\{1,2,3,4,5,6\}$
- Event A is subset of outcomes, e.g. $\{1,3,5\}$
- Probability $P(A)$, e.g. $P(A)=0.5$

Axioms of Probability theory

- $0 \leq P(A) \leq 1$
- $P(\Omega) = 1,\ P(\emptyset) = 0$
- $P(A \cup B) = P(A) + P(B) - P(A \cap B)$



Discrete Random Variables

- X denotes a random variable
- X can take on a countable number of values in $\{x_1, x_2, \dots, x_n\}$
- $P(X=x_i)$ is the probability that the random variable X takes on value x_i
- $P(.)$ is called the probability mass function

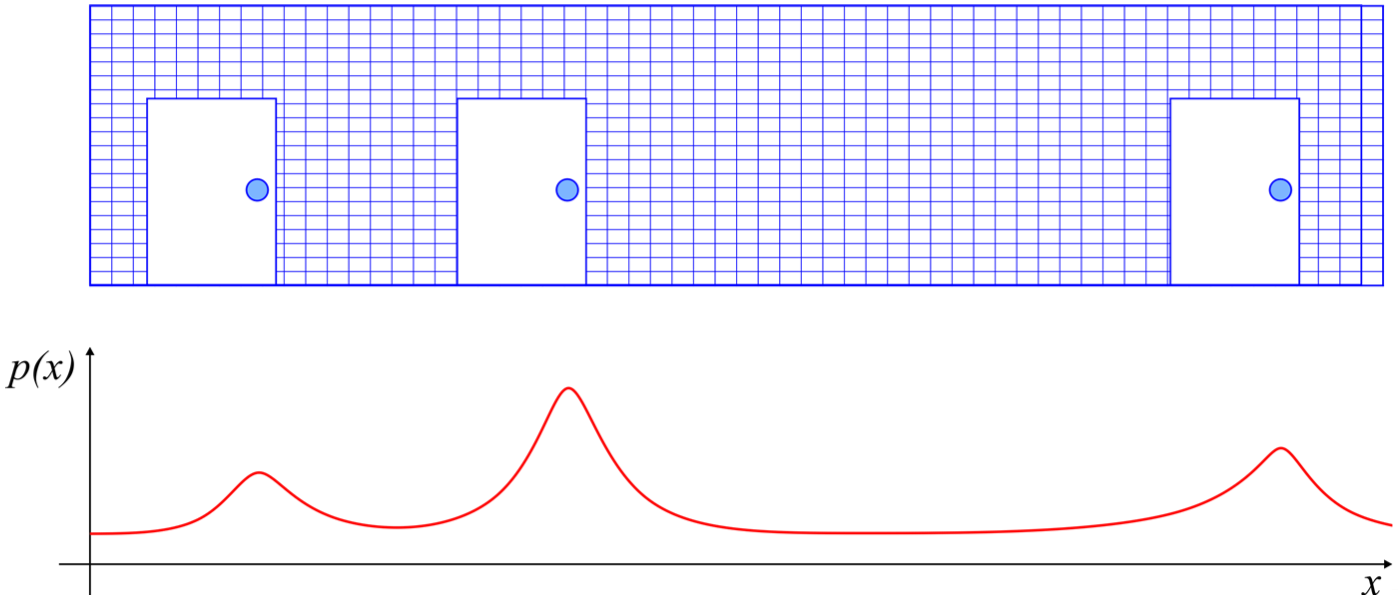
- Example: P(Room)=<0.6,0.3,0.06,0.03>, Room one of the office, corridor, lab, kitchen

Continuous Random Variables

- X takes on continuous values.
- P(X=x) or P(x) is called the **probability density function (PDF)**.

Thrun, Burgard, Fox, "Probabilistic Robotics", MIT Press, 2005

- Example:



Proper Distributions Sum To One

- Discrete Case
$$\sum_x P(x) = 1$$
- Continuous Case
$$\int p(x)dx = 1$$

Joint and Conditional Probabilities

- $p(X = x, \text{ and } Y = y) = P(x, y)$

- If X and Y are **independent** then:

$$P(x, y) = P(x)P(y)$$

- Is the probability of **x given y**

$$P(x|y)P(y) = P(x, y)$$

- If X and Y are independent then:

$$P(x|y) = P(x)$$

Conditional Independence

- Definition of conditional independence:

$$P(x, y|z) = P(x|z)P(y|z)$$

- Equivalent to:

$$P(x|z) = P(x|y, z)$$
$$P(y|z) = P(y|x, z)$$

- Note: this does not necessarily mean that:

$$P(x, y) = P(x)P(y)$$

Marginalization

- **Discrete case:** $$P(x) = \sum_y P(x, y)$$
- **Continuous case:** $$p(x) = \int p(x, y)dy$$

Marginalization example

P(X,Y)	x1	x1	x1	x1	P(Y) ↓
y1	1/8	1/16	1/32	1/32	1/4
y1	1/16	1/8	1/32	1/32	1/4
y1	1/16	1/16	1/16	1/16	1/4
y1	1/4	0	0	0	1/4
P(X) →	1/2	1/4	1/8	1/8	1

Expected value of a Random Variable

- **Discrete case:** $$E[X] = \sum_i x_i P(x_i)$$

- **Continuous case:** $$E[X] = \int x P(X = x) dx$$

- The expected value is the weighted average of all values a random variable can take on.
- Expectation is a linear operator:

$$E[aX + b] = aE[X] + b$$

Covariance of a Random Variable

- Measures the **square expected deviation from the mean**:

$$Cov[X] = E[X - E[X]]^2 = E[X^2] - E[X]^2$$

Estimation from Data

- Observations: $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathcal{R}^d$
- Sample Mean: $\mu = \frac{1}{n} \sum_i \mathbf{x}_i$
- Sample Covariance:

$$\Sigma = \frac{1}{n-1} \sum_i (\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)$$

BadgerWorks Lectures

Topic: State Estimation – Reasoning with Bayes Law

The State Estimation problem

- We want to estimate the world state x from:
  - Sensor measurements z and
  - Controls u

- We need to model the relationship between these random variables, i.e:

$$p(\mathbf{x}|\mathbf{z}) \qquad p(\mathbf{x}'|\mathbf{x}, \mathbf{u})$$

Causal vs. Diagnostic Reasoning

$P(\mathbf{x}|\mathbf{z})$ Is diagnostic

$P(\mathbf{z}|\mathbf{x})$ Is causal

- Diagnostic reasoning is typically what we need.
- Often causal knowledge is easier to obtain.
- Bayes rule allows us to use causal knowledge in diagnostic reasoning.

Bayes rule

- Definition of **conditional probability**:

$$P(x, z) = P(x|z)P(z) = P(z|x)P(x)$$

- **Bayes rule:**

Observation likelihood

Prior on world state

$$P(x|z) = \frac{P(z|x)P(x)}{P(z)}$$

Prior on sensor observations

Normalization

- Direct computation of $P(\mathbf{z})$ can be difficult.
- Idea: compute improper distribution, normalize afterwards.

- **STEP 1:** $L(x|z) = P(z|x)P(x)$

- **STEP 2:** $P(z) = \sum_x P(z,x) = \sum_x P(z|x)P(x) = \sum_x L(x|z)$

- **STEP 3:** $P(x|z) = L(x|z)/P(z)$

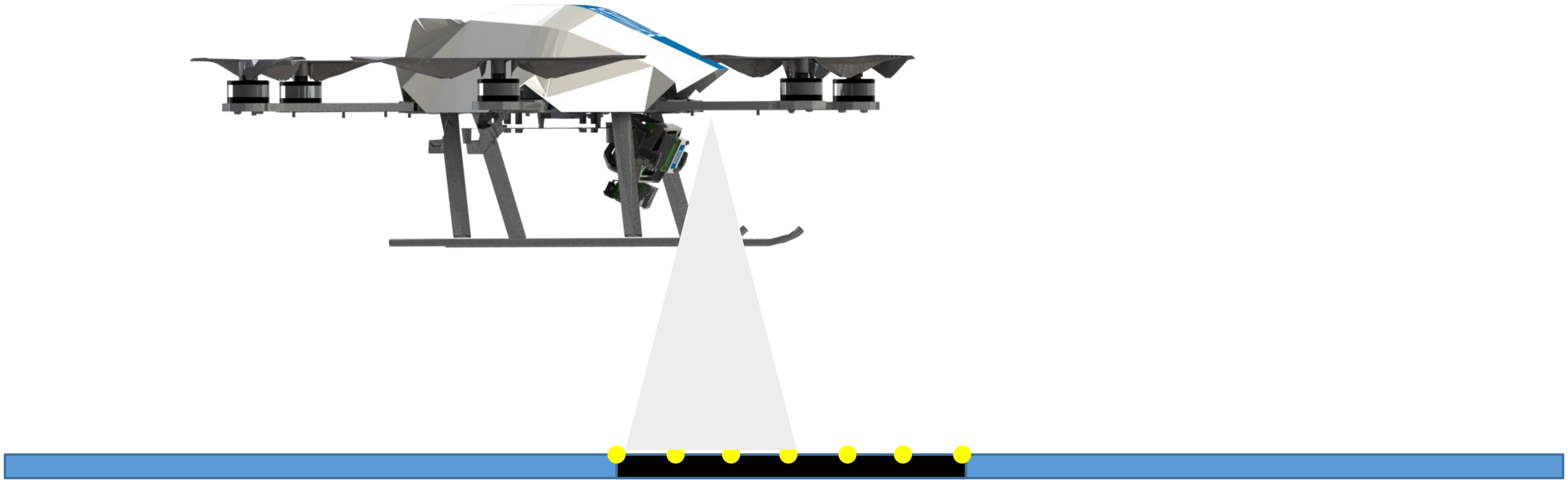
Normalization

- Direct computation of $P(\mathbf{z})$ can be difficult.
- Idea: compute improper distribution, normalize afterwards.

- STEP 1: $$L(x|z) = P(z|x)P(x)$$
- STEP 2: $$P(z) = \sum_x P(z,x) = \sum_x P(z|x)P(x) = \sum_x L(x|z)$$
- STEP 3: $$P(x|z) = L(x|z)/P(z)$$

Example: Sensor Measurement

- Quadrotor seeks the Landing Zone
- The landing zone is marked with many bright lamps
- The quadrotor has a light sensor.



Example: Sensor Measurement

- Binary sensor $$Z \in \{bright, \overline{bright}\}$$
- Binary world state $$X \in \{home, \overline{home}\}$$
- Sensor model $$P(Z = bright | X = home) = 0.6$$ $$P(Z = bright | X = \overline{home}) = 0.3$$
- Prior on world state $$P(X = home) = 0.5$$
- Assume: robot observes light, i.e. $$Z = bright$$
- What is the probability $P(X = home | Z = bright)$ that the robot is above the landing zone.

Example: Sensor Measurement

- Sensor model:
$$P(Z = bright|X = home) = 0.6$$
$$P(Z = bright|X = \bar{home}) = 0.3$$
- Prior on world state: $P(X = home) = 0.5$
- Probability after observation (using Bayes):

$$P(X = home|Z = bright) =$$
$$\frac{P(bright|home)P(home)}{P(bright|home)P(home) + P(bright|\bar{home})P(\bar{home})} =$$
$$\frac{0.6 \cdot 0.5}{0.6 \cdot 0.5 + 0.3 \cdot 0.5} = 0.67$$

Actions (Motions)

- Often the world is dynamic since
  - Actions are carried out by the robot
  - Actions are carried out by other agents
  - Or simply because time is passing and the world changes
- How can we incorporate actions?

Example actions

- MAV accelerates by changing the speed of its motors.
- The ground robot moves due to it being on an inclined terrain.

- Actions are never carried out with absolute certainty: leave a quadrotor hover and see it drifting!
- In contrast to measurements, actions generally increase the uncertainty of the state estimate

Action Models

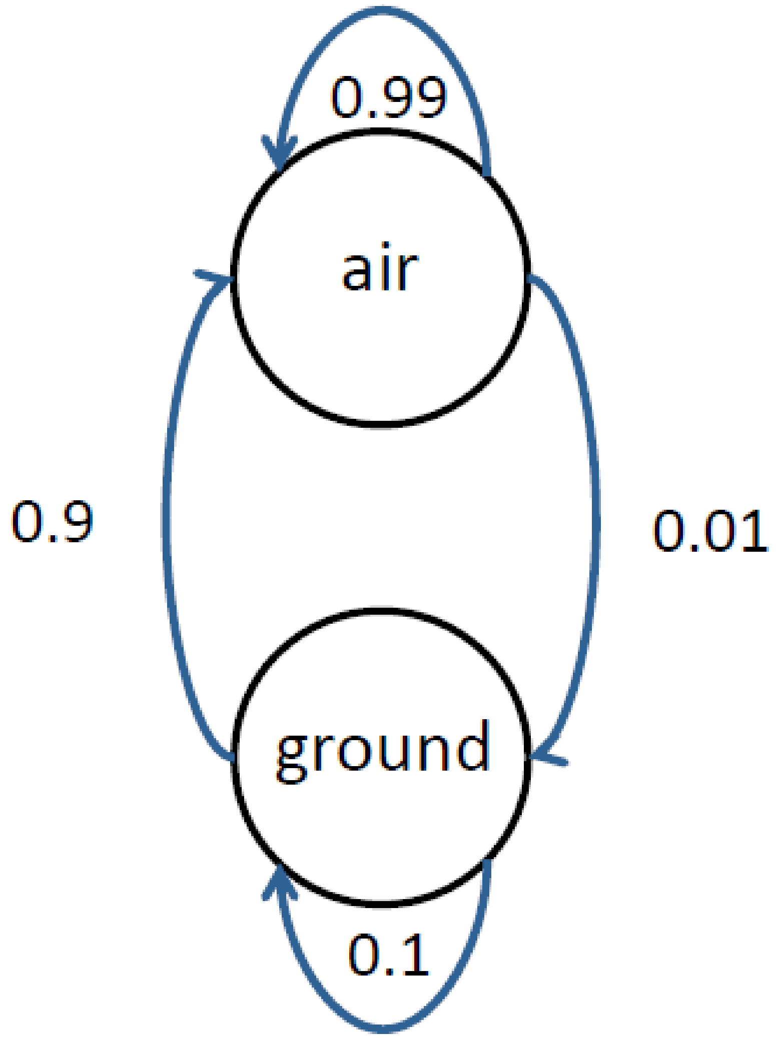
- To incorporate the outcome of an action u into the current estimate ("belief"), we use the conditional pdf

$$p(x' \mid u, x)$$

- This term specifies the probability that executing the action u in state x will lead to state x'

Example: Take-Off

- Action: $u \in \{\text{takeoff}\}$
- World state: $x \in \{\text{ground}, \text{air}\}$



Integrating the Outcome of Actions

- Discrete case:

$$P(x' \mid u) = \sum_x P(x' \mid u, x) P(x)$$

- Continuous case:

$$p(x' \mid u) = \int p(x' \mid u, x) p(x) \mathrm{d}x$$

Example: Take-Off

- Prior belief on robot state: $P(x = \text{ground}) = 1.0$
- Robot executes "take-off" action
- What is the robot's belief after one time step?

$$
\begin{aligned}
P(x' = \text{ground}) = &\sum_x P(x' = \text{ground} \mid u, x) P(x) \\
= &P(x' = \text{ground} \mid u, x = \text{ground}) P(x = \text{ground}) \\
+ &P(x' = \text{ground} \mid u, x = \text{air}) P(x = \text{air}) \\
= &0.1 \cdot 1.0 + 0.01 \cdot 0.0 = 0.1
\end{aligned}
$$

BadgerWorks Lectures

Topic: State Estimation – Bayes Filter

Markov Assumption

- Observations depend only on current state

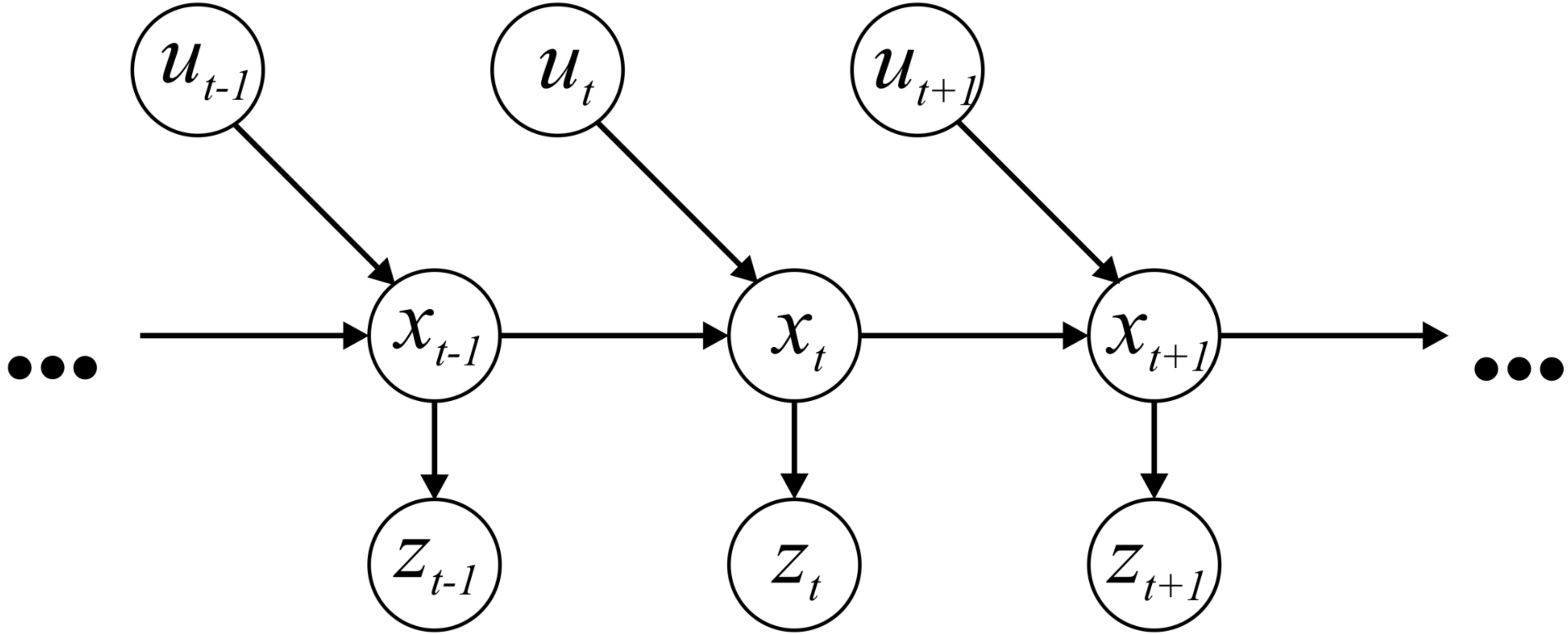
$$P(z_t|x_{0:t}, z_{1:t-1}, u_{1:t}) = P(z_t|x_t)$$

- Current state depends only on previous state and current action

$$P(x_t|x_{0:t}, z_{1:t}, u_{1:t}) = P(x_t|x_{t-1}, u_t)$$

Markov Chain

- A Markov Chain is a stochastic process where, given the present state, the past and the future states are independent.



Underlying Assumptions

- Static world
- Independent noise
- Perfect model, no approximation errors

Bayes Filter

- **Given**
  - Sequence of observations and actions: z_t, u_t
  - Sensor model: $P(z|x)$
  - Action model: $P(x'|x, u)$
  - Prior probability of the system state: $P(x)$
- **Desired**
  - Estimate of the state of the dynamic system: x
  - Posterior of the state is also called belief:

$$Bel(x_t) = P(x_t|u_1, z_1, \ldots, u_t, z_t)$$

Bayes Filter Algorithm

- **For each time step, do:**
  - Apply motion model:

$$\overline{Bel}(x_t) = \sum_{x_t - 1} P(x_t|x_{t-1}, u_t) Bel(x_{t-1})$$

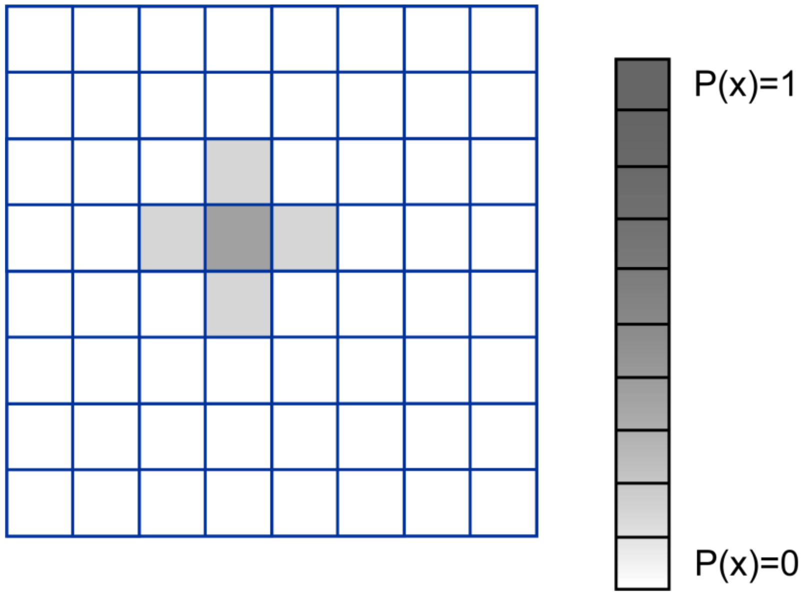
  - Apply sensor model:

$$Bel(x_t) = \eta P(z_t|x_t) \overline{Bel}(x_t)$$

  - η is a normalization factor to ensure that the probability is maximum 1.

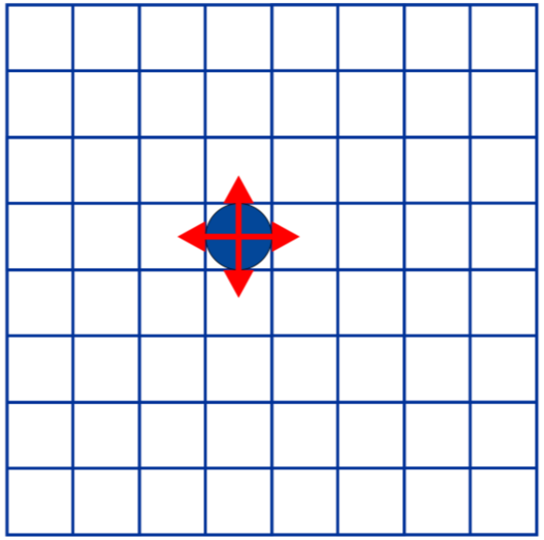
Example: Localization

- Discrete state: $x \in \{1, 2, ..., w\} \times \{1, 2, ..., h\}$
- Belief distribution can be represented as a grid
- This is also called a **historigram filter**



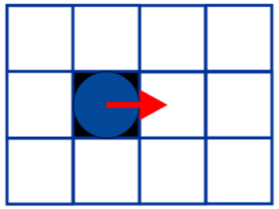
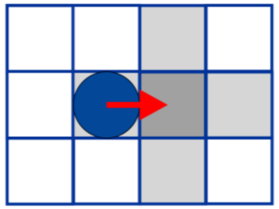
Example: Localization

- Action: $u \in \{north, east, south, west\}$
- Robot can move one cell in each time step
- Actions are not perfectly executed



Example: Localization

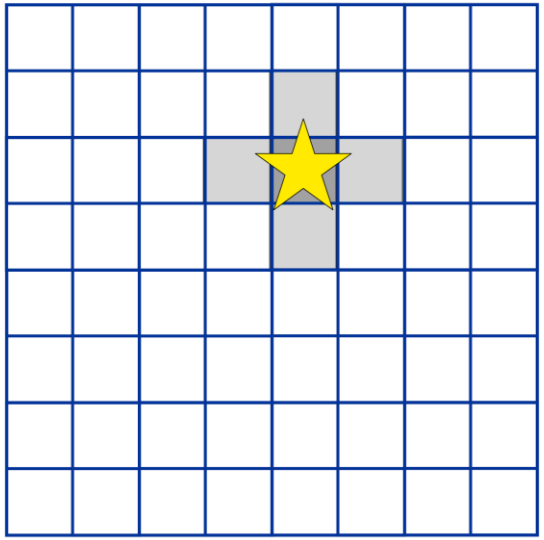
- Action
- Robot can move one cell in each time step
- Actions are not perfectly executed
- Example: move east

$x_{t-1} =$  $u = east$ 

- 60% success rate, 10% to stay/move too far/ move one up/ move one down

Example: Localization

- Binary observation: $z \in \{marker, \bar{marker}\}$
- One (special) location has a marker
- Marker is sometimes also detected in neighboring cells

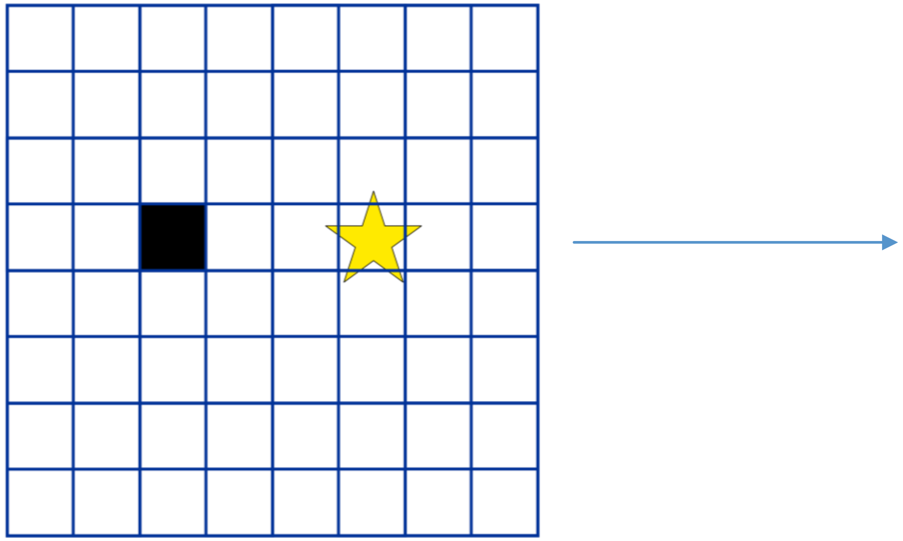


Example: Localization

- Let’s start a simulation run…

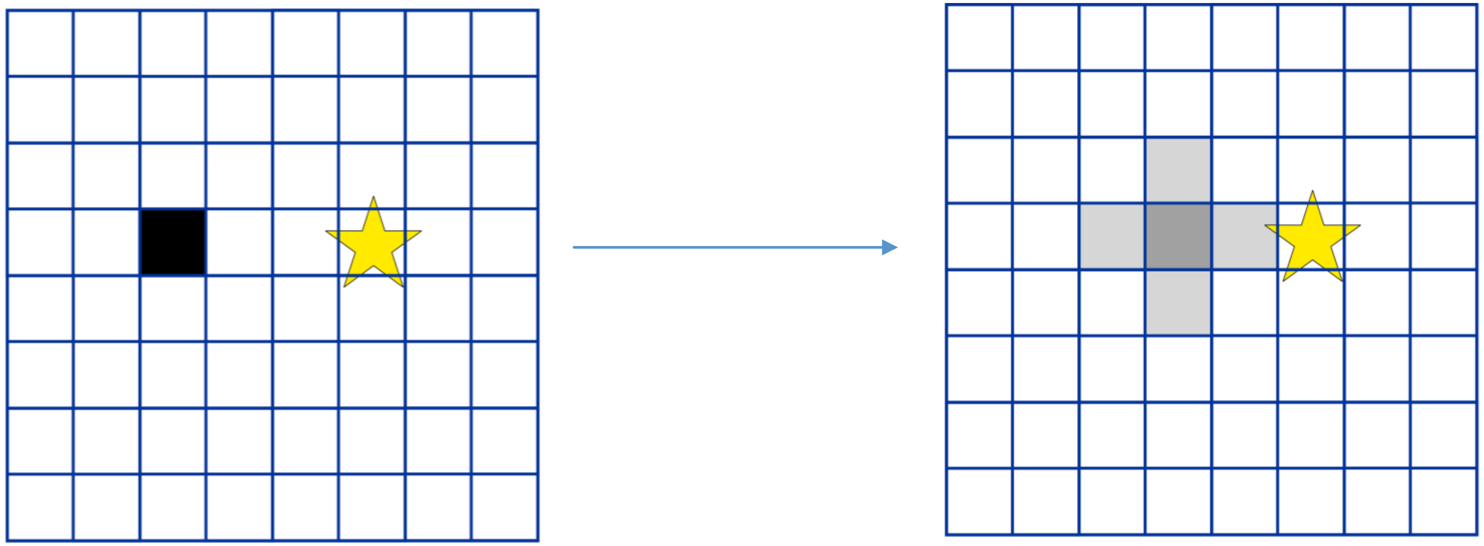
Example: Localization

- t=0
- Prior distribution (initial belief)
- Assume that we know the initial location (if not, we could initialize with a uniform prior)



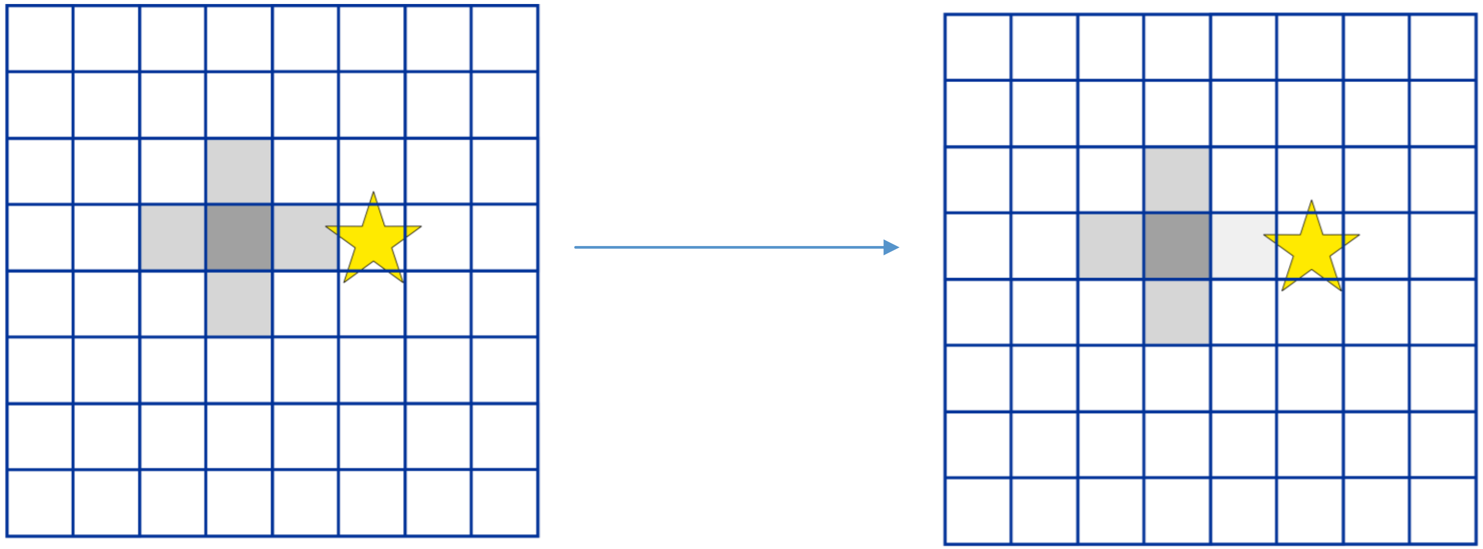
Example: Localization

- t=1, u =east, z=no-marker
- Bayes filter step 1: Apply motion model



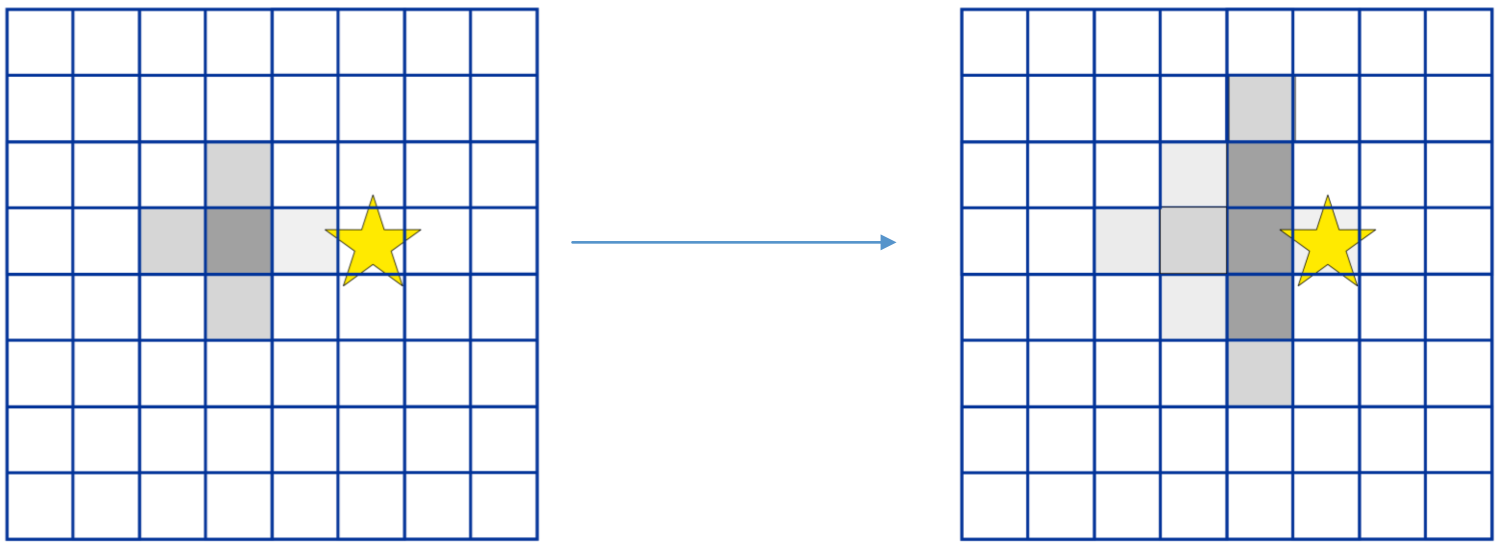
Example: Localization

- t=1, u =east, z=no-marker
- Bayes filter step 2: Apply observation model



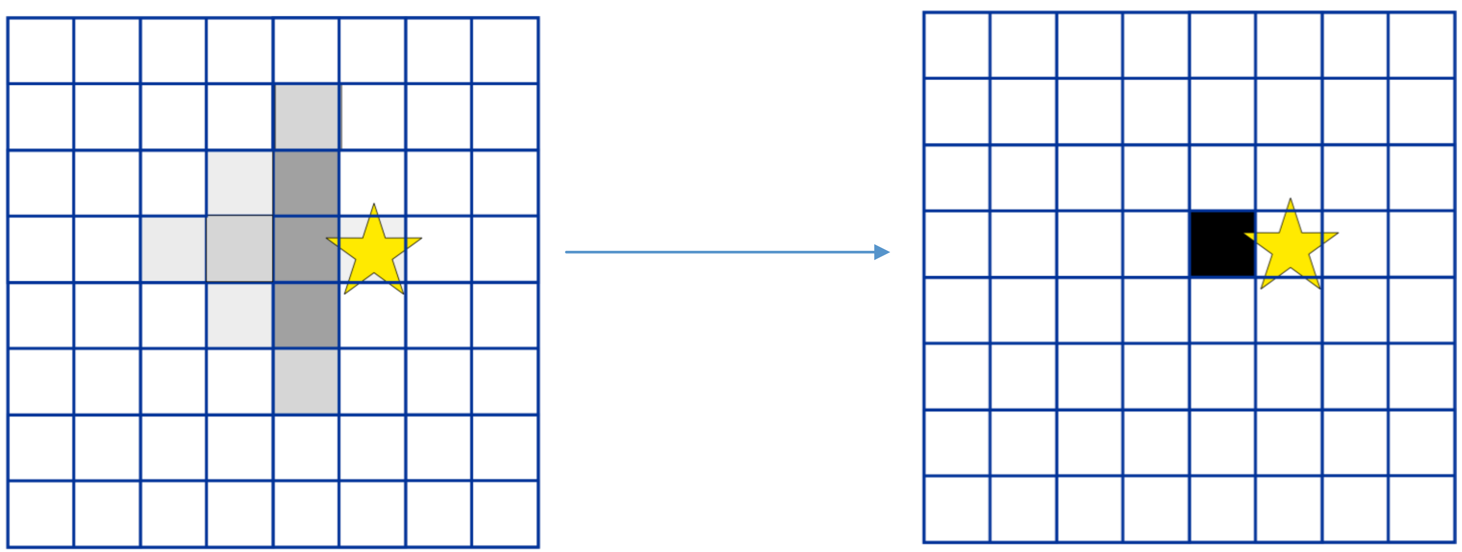
Example: Localization

- t=2, u =east, z=marker
- Bayes filter step 1: Apply motion model



Example: Localization

- t=2, u =east, z=marker
- Bayes filter step 2: Apply observation model
- Question: where is the robot?



BadgerWorks Lectures

Topic: State Estimation – Kalman Filter

Kalman Filter

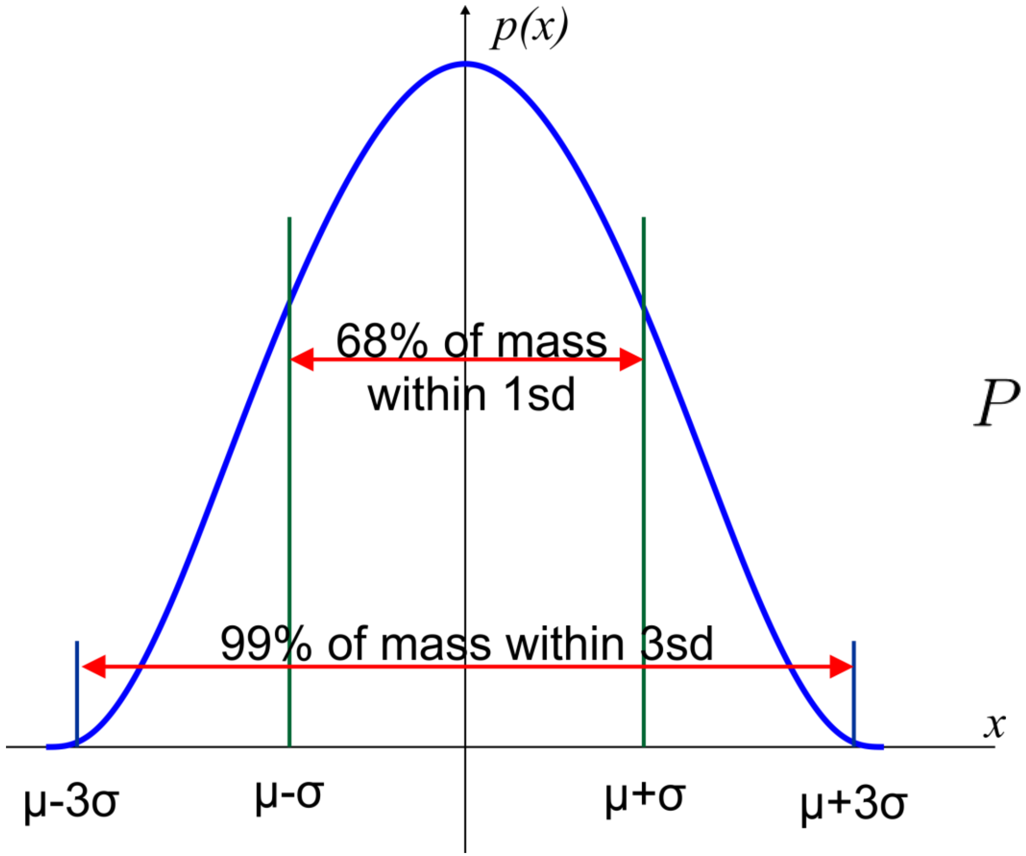
- Bayes filter is a useful tool for state estimation.
- Histogram filter with grid representation is not very efficient.
- How can we represent the state more efficiently?

Kalman Filter

- Bayes filter with continuous states
- State represented with a normal distribution
- Developed in the late 1950's. A cornerstone. Designed and first application: estimate the trajectory of the Apollo missiles.
- 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.

Kalman Filter

- Univariate distribution



$$X \sim \mathcal{N}(\mu, \sigma^2)$$

mean

Variance (squared standard deviation)

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

Kalman Filter

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

$$p(\mathbf{X} = \mathbf{x}) = \mathcal{N}(\mathbf{x}; \mu, \mathbf{\Sigma}) = \frac{1}{(2\pi)^{n/2}|\mathbf{\Sigma}|^{1/2}} \exp(-\frac{1}{2}(\mathbf{x} - \mu)^T \mathbf{\Sigma}^{-1}(\mathbf{x} - \mu))$$

Properties of Normal Distributions

- Linear transformation – remains Gaussian

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

- Intersection of two Gaussians – remains Gaussian

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

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

Properties of Normal Distributions

- Linear transformation – remains Gaussian

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

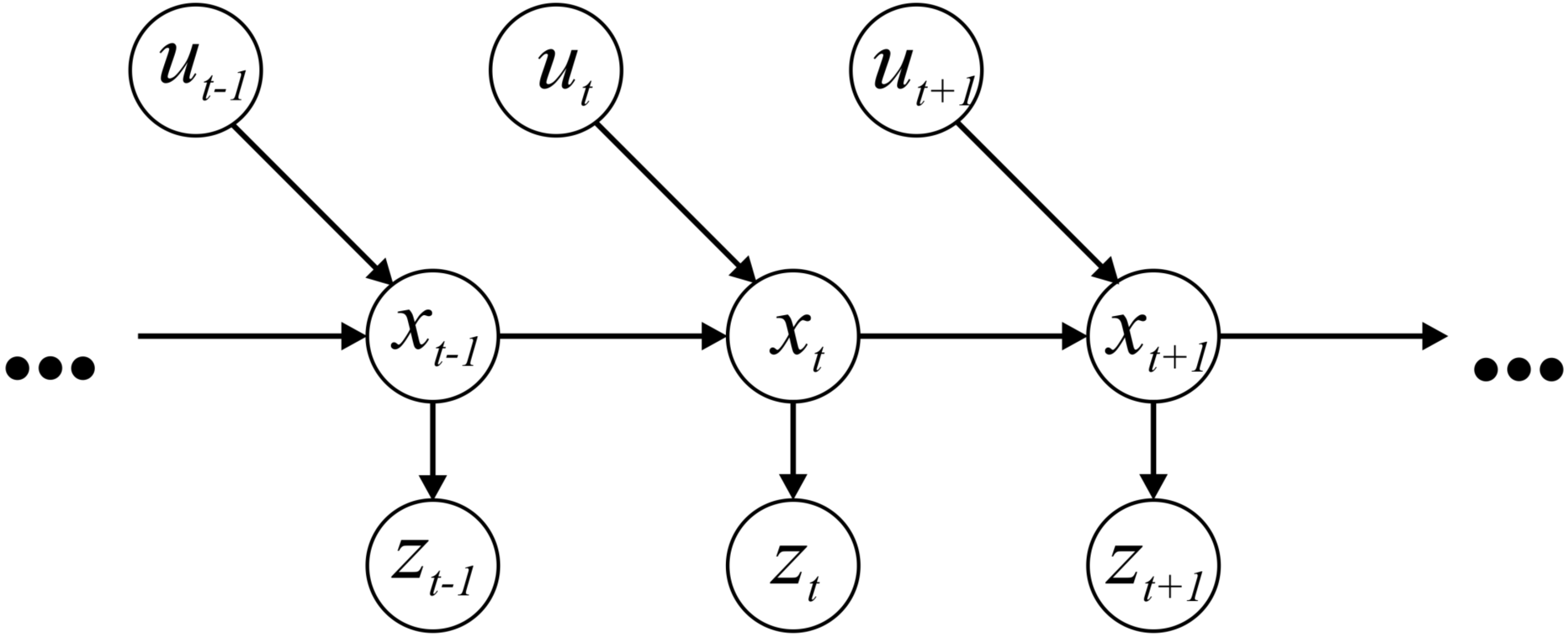
- Intersection of two Gaussians – remains Gaussian

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

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

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) with a Gaussian

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

Linear Process Model

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

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

- 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) with a Gaussian

$$\mathbf{x}_t \sim \mathcal{N}(\mu_t, \Sigma_t)$$

- Assume that the system evolves linearly over time, then 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) with a Gaussian

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

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

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

- With $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$

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 zero-mean, normally distributed observation noise

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

- With $\delta_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$

Kalman Filter

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

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

- And (linear) measurements of the state

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

- With $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ and $\delta_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$

Kalman Filter

- State $\mathbf{x} \in \mathbb{R}^n$
- Controls $\mathbf{u} \in \mathbb{R}^l$
- Observations $\mathbf{z} \in \mathbb{R}^k$
- Process equation $\mathbf{x}_t = \underbrace{\mathbf{A}}_{nxn}\mathbf{x}_{t-1} + \underbrace{\mathbf{B}}_{nxl}\mathbf{u}_t + \epsilon_t$
- Measurement equation $\mathbf{z}_t = \underbrace{\mathbf{C}}_{nxk}\mathbf{x}_t + \delta_t$

Kalman Filter

- Initial belief is Gaussian

$$Bel(x_0) = \mathcal{N}(\mathbf{x}_0; \mu_0, \mathbf{\Sigma}_0)$$

- Next state is also Gaussian (linear transformation)

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

- Observations are also Gaussian

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

Recall: Bayes Filter Algorithm

- For each step, do:
  - Apply motion model

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

  - Apply sensor model

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

From Bayes Filter to Kalman Filter

- For each step, do:
  - Apply motion model

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

From Bayes Filter to Kalman Filter

- For each step, do:
  - Apply motion model

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

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

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

From Bayes Filter to Kalman Filter

- For each step, do:
  - Apply sensor model

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

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

$$= \mathcal{N}(x_t; \mu_t, \mathbf{\Sigma}_t)$$

- With $\mathbf{K}_t = \bar{\mathbf{\Sigma}}_t\mathbf{C}^T(\mathbf{C}\bar{\mathbf{\Sigma}}_t\mathbf{C}^T + \mathbf{R})^{-1}$ (**Kalman Gain**)

From Bayes Filter to Kalman Filter

Blends between our previous estimate $\bar{\mu}_t$ and the discrepancy between our sensor observations and our predictions.

The degree to which we believe in our sensor observations is the Kalman Gain. And this depends on a formula based on the errors of sensing etc. In fact it depends on the ratio between our uncertainty Σ and the uncertainty of our sensor observations R.

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

old mean

Kalman Gain

From Bayes Filter to Kalman Filter

- For each step, do:
  - Apply sensor model

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

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

$$= \mathcal{N}(x_t; \mu_t, \mathbf{\Sigma}_t)$$

  - With $\mathbf{K}_t = \bar{\mathbf{\Sigma}}_t \mathbf{C}^T (\mathbf{C}\bar{\mathbf{\Sigma}}_t \mathbf{C}^T + \mathbf{R})^{-1}$ (**Kalman Gain**)

Kalman Filter Algorithm

- For each step, do:
  - Apply motion model (prediction step)

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

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

  - Apply sensor model (correction step)

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

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

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

Kalman Filter Algorithm

Prediction & Correction steps can happen in any order.

- For each step, do:
  - Apply motion model (**prediction step**)

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

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

  - Apply sensor model (**correction step**)

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

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

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

Kalman Filter Algorithm

Prediction & Correction steps can happen in any order.

Prediction

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

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

Correction

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

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

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

Complexity

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

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

- Optimal for linear Gaussian systems
  - But most robots are nonlinear! This is why in practice we use Extended Kalman Filters and other approaches.

BadgerWorks Lectures

Topic: Extended Kalman Filter

Dr. Kostas Alexis (CSE)

These slides relied on the lectures from C. Stachniss, J. Sturm and the book “Probabilistic Robotics” from Thurn et al. as well as the courses of C. Stachiniss

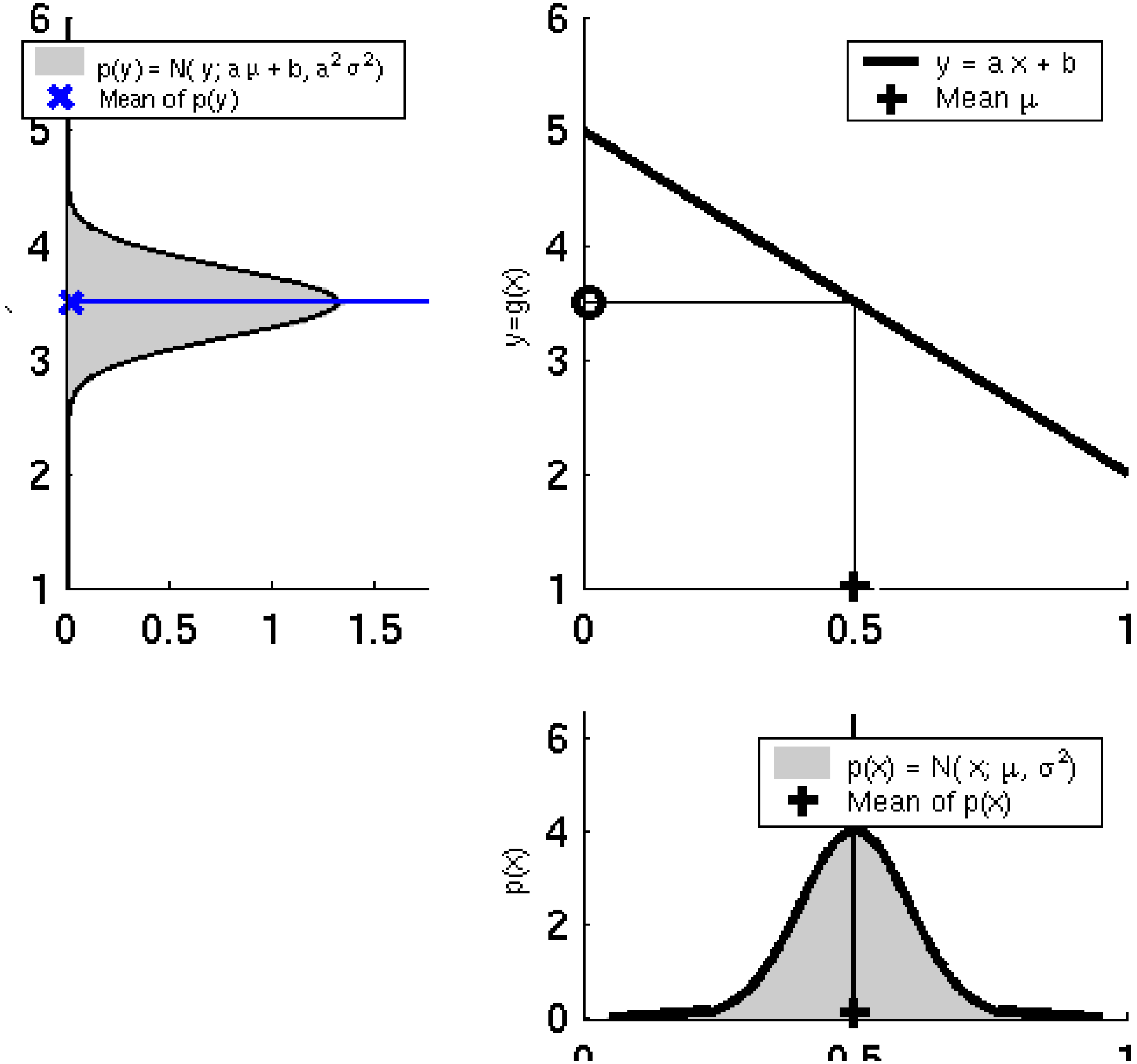
Kalman Filter Assumptions

- Gaussian distributions and noise
- Linear motion and observation model
  - **What if this is not the case?**

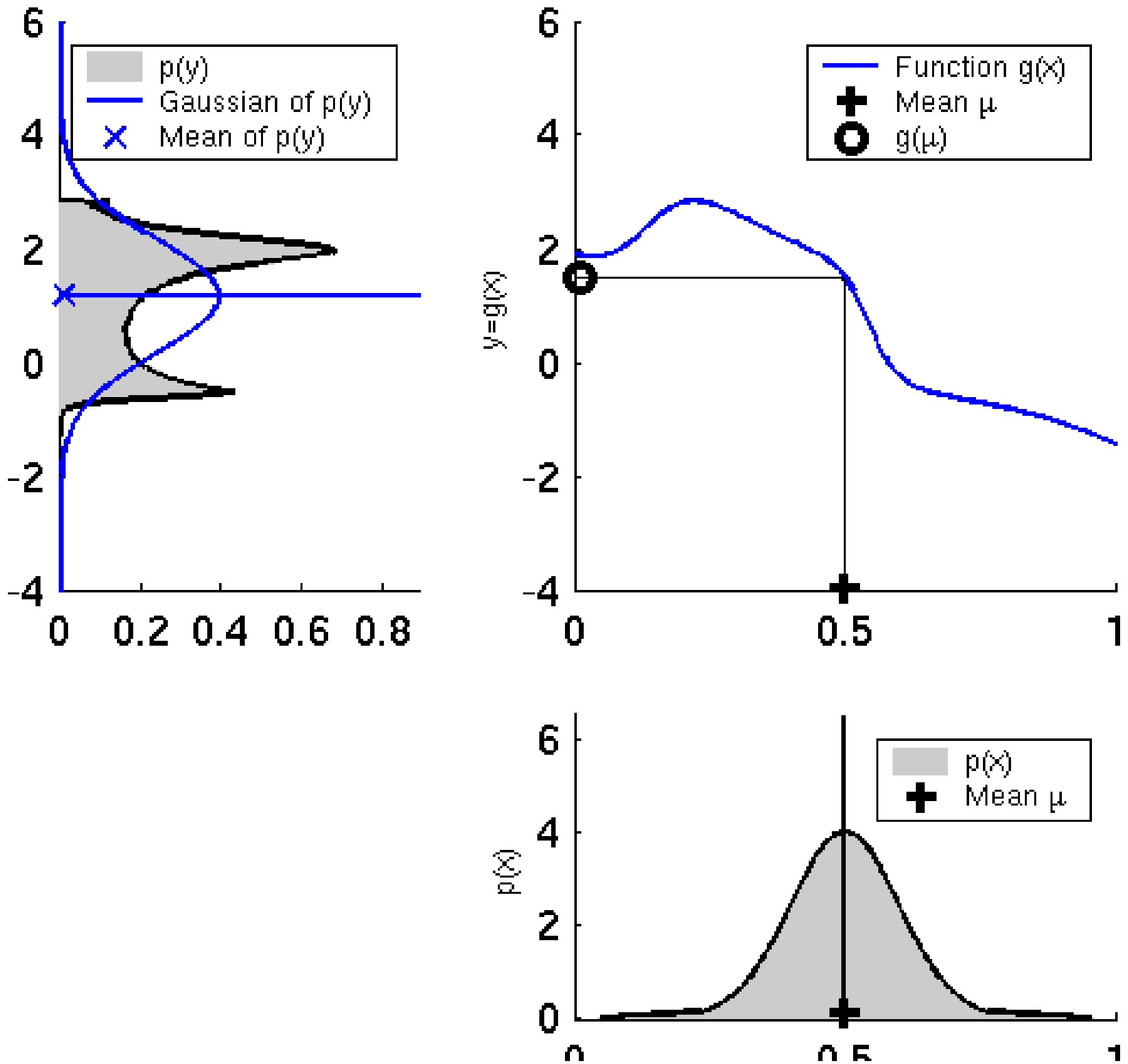
$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t$$

$$z_t = C_t x_t + \delta_t$$

Linearity Assumption Revisited



Nonlinear Function



Nonlinear Dynamical Systems

- Real-life robots are mostly nonlinear systems.
- The **motion equations** are expressed as **nonlinear differential (or difference) equations**:

$$x_t = g(u_t, x_{t-1})$$

- Also leading to a **nonlinear observation function:**

$$z_t = h(x_t)$$

Taylor Expansion

- Solution: approximate via linearization of both functions
- **Motion Function:**

$$g(x_{t-1}, u_t) \approx g(\mu_{t-1}, u_t) + \frac{\partial g(\mu_{t-1}, u_t)}{\partial x_{t-1}}(x_{t-1} - \mu_{t-1})$$
$$= g(\mu_{t-1}, u_t) + G_t(x_{t-1} - \mu_{t-1})$$

- **Observation Function:**

$$h(x_t) \approx h(\bar{\mu}_t) + \frac{\partial h(\bar{\mu}_t)}{\partial x_t}(x_t - \mu_t)$$
$$= h(\bar{\mu}_t) + H_t(x_t - \mu_t)$$

Reminder: Jacobian Matrix

- It is a non-square matrix *mxn* in general
- Given a vector-valued function:

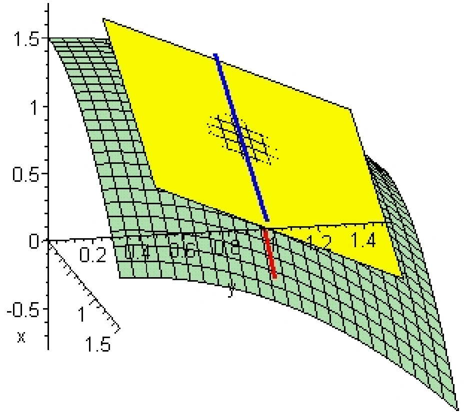
$$g(x) = \begin{pmatrix} g_1(x) \\ g_2(x) \\ \vdots \\ g_m(x) \end{pmatrix}$$

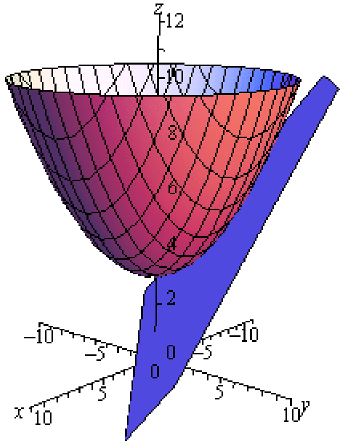
- The **Jacobian matrix** is defined as:

$$G_x = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \cdots & \frac{\partial g_2}{\partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial g_m}{\partial x_1} & \frac{\partial g_m}{\partial x_2} & \cdots & \frac{\partial g_m}{\partial x_n} \end{pmatrix}$$

Reminder: Jacobian Matrix

- It is the orientation of the tangent plane to the vector-valued function at a given point





Courtesy: K. Arras

- Generalizes the gradient of a scaled-valued function.

Extended Kalman Filter

- For each time step, do:
- **Apply Motion Model:**

$$\bar{\mu}_t = g(\mu_{t-1}, u_t)$$

$$\bar{\Sigma}_t = G_t \Sigma G_t^\top + Q \quad \text{with} \quad G_t = \frac{\partial g(\mu_{t-1}, u_t)}{\partial x_{t-1}}$$

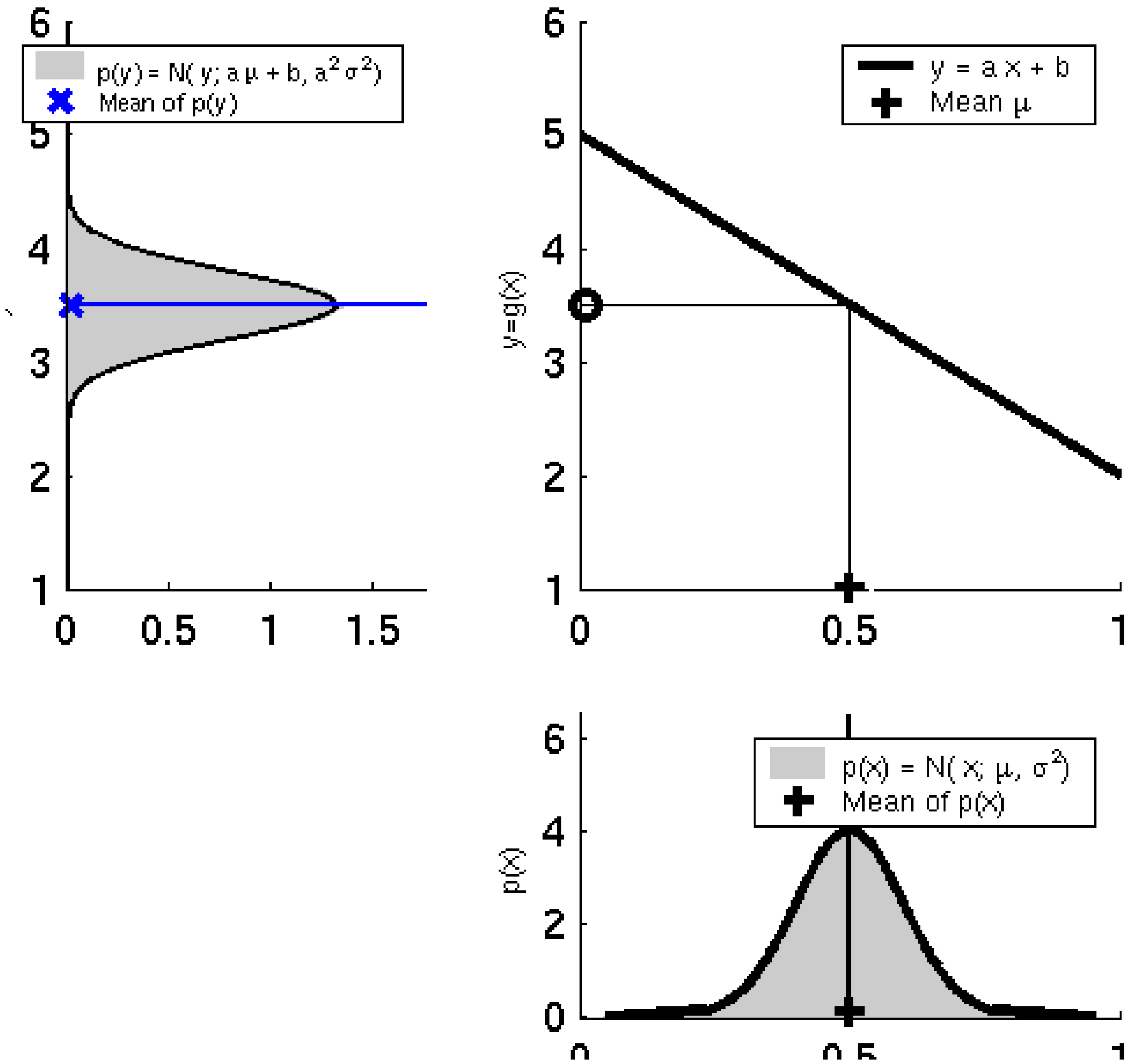
- **Apply Sensor Model:**

$$\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$$

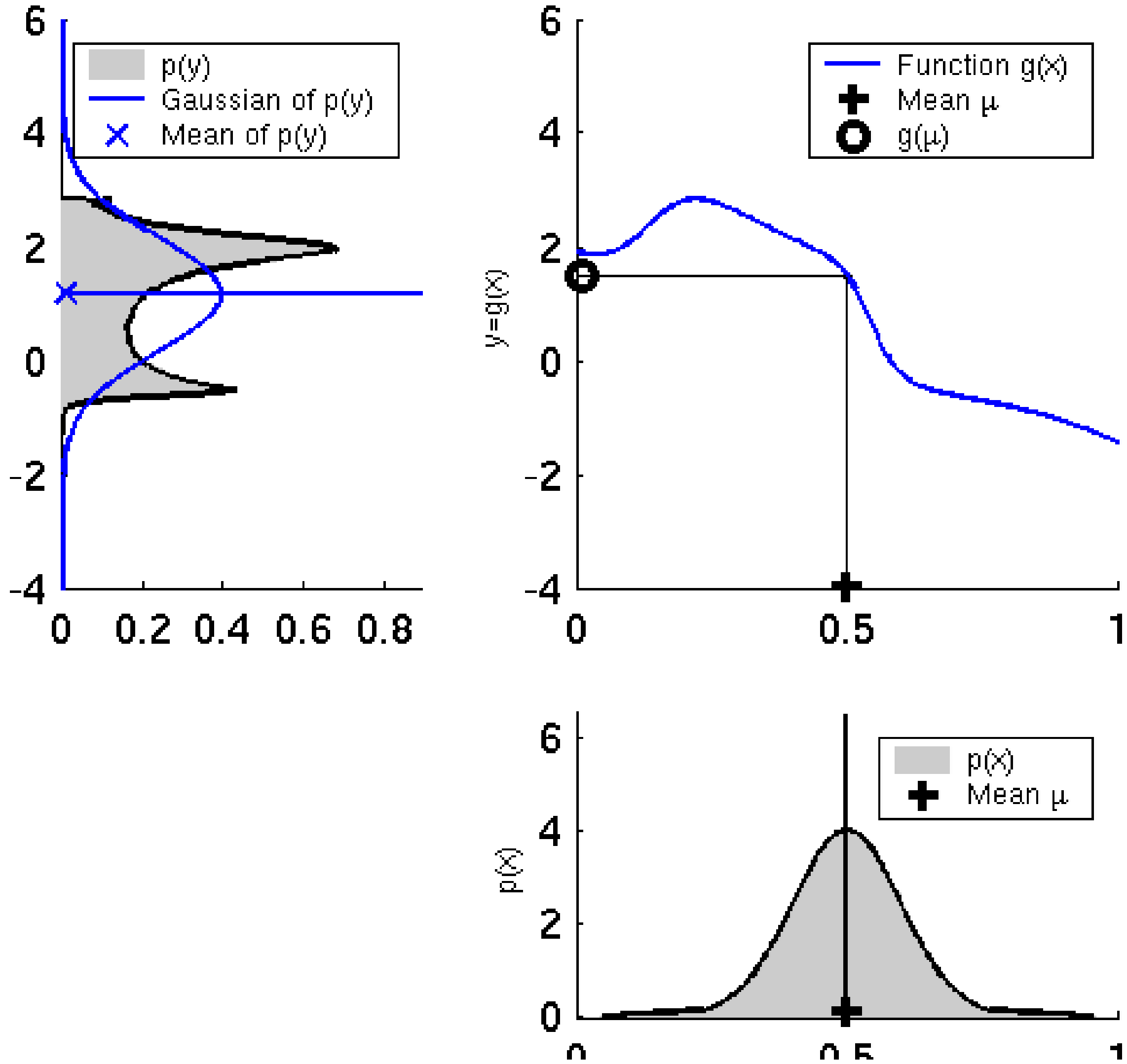
$$\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$$

where $K_t = \bar{\Sigma}_t H_t^\top (H_t \bar{\Sigma}_t H_t^\top + R)^{-1}$ and $H_t = \dfrac{\partial h(\bar{\mu}_t)}{\partial x_t}$

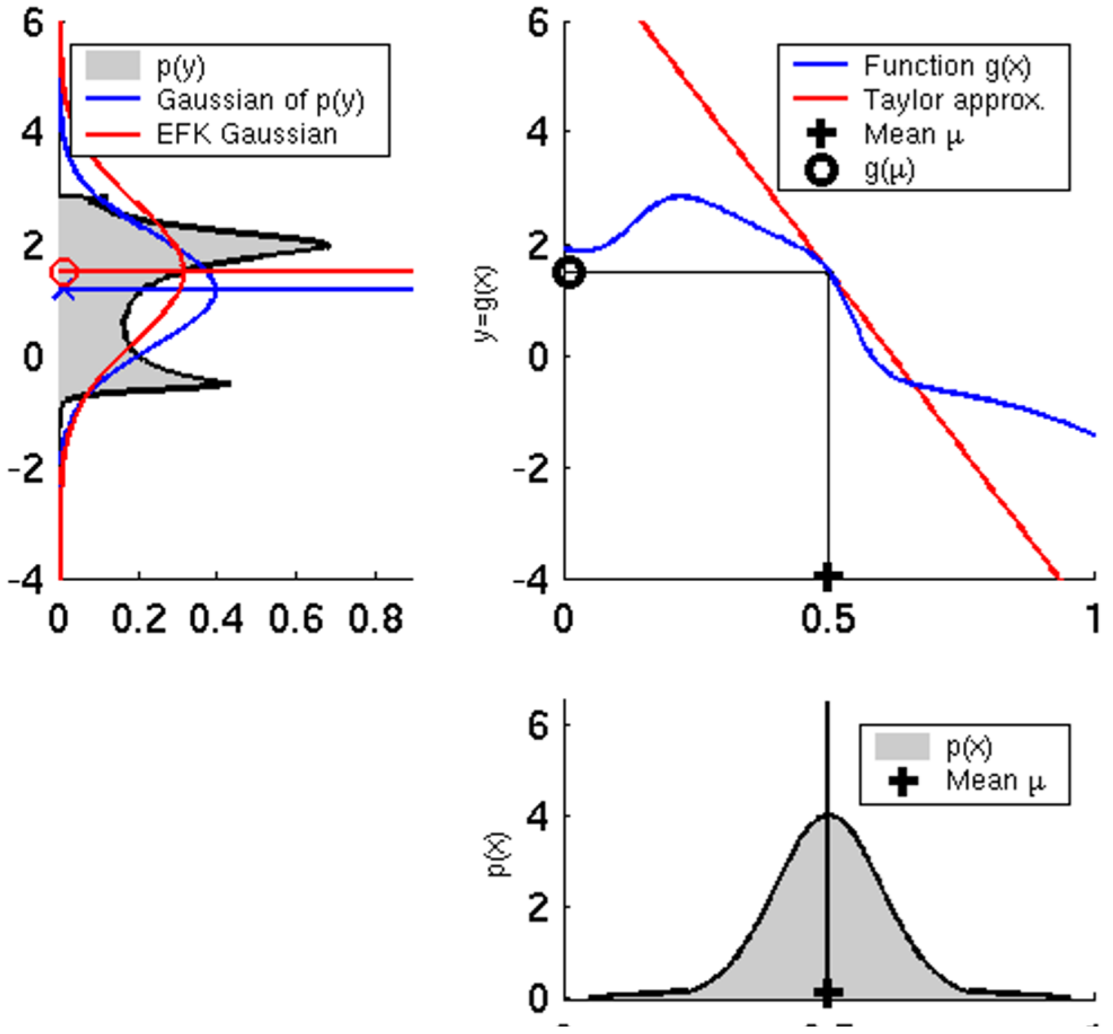
Linearity Assumption Revisited



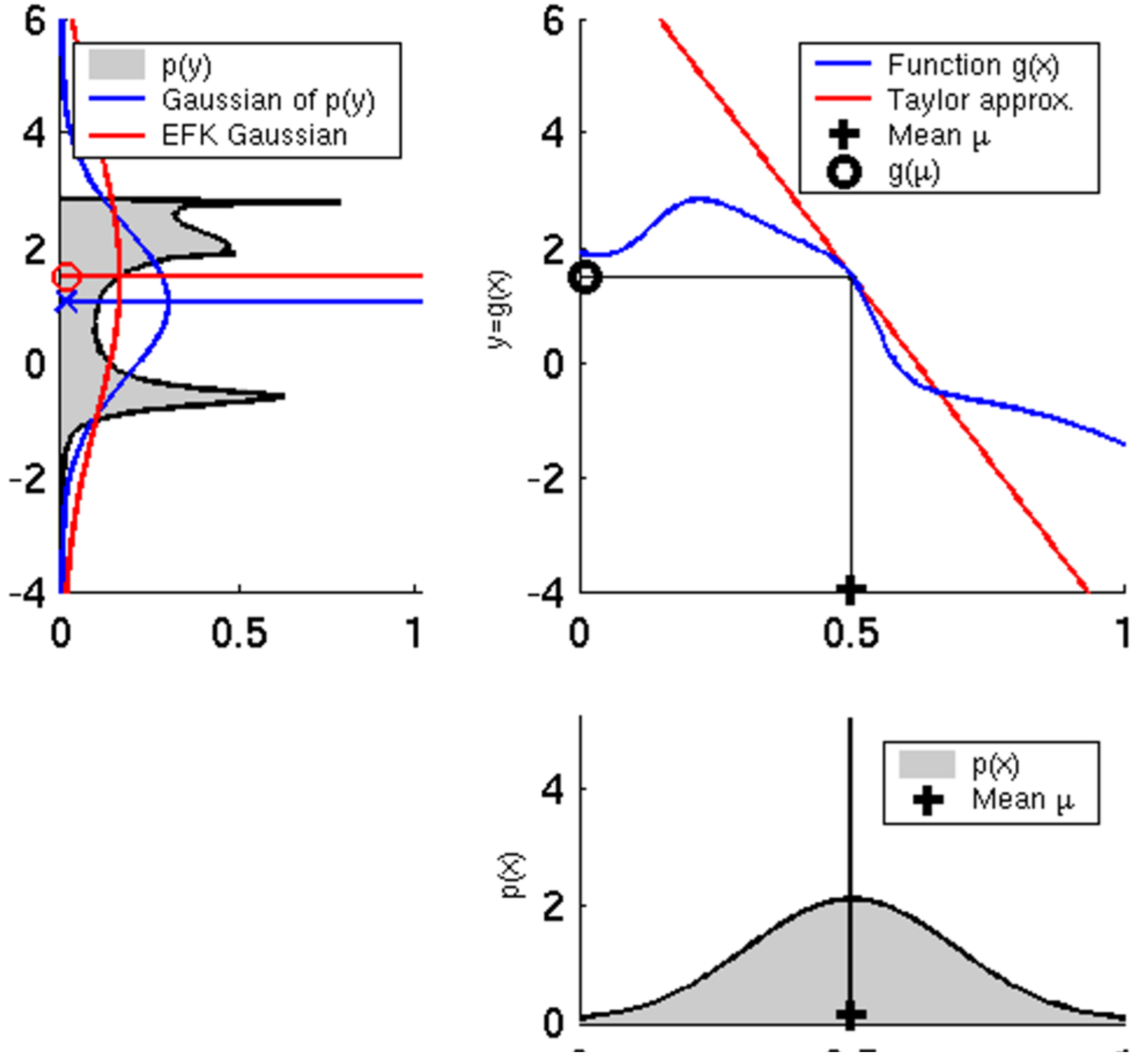
Nonlinear Function



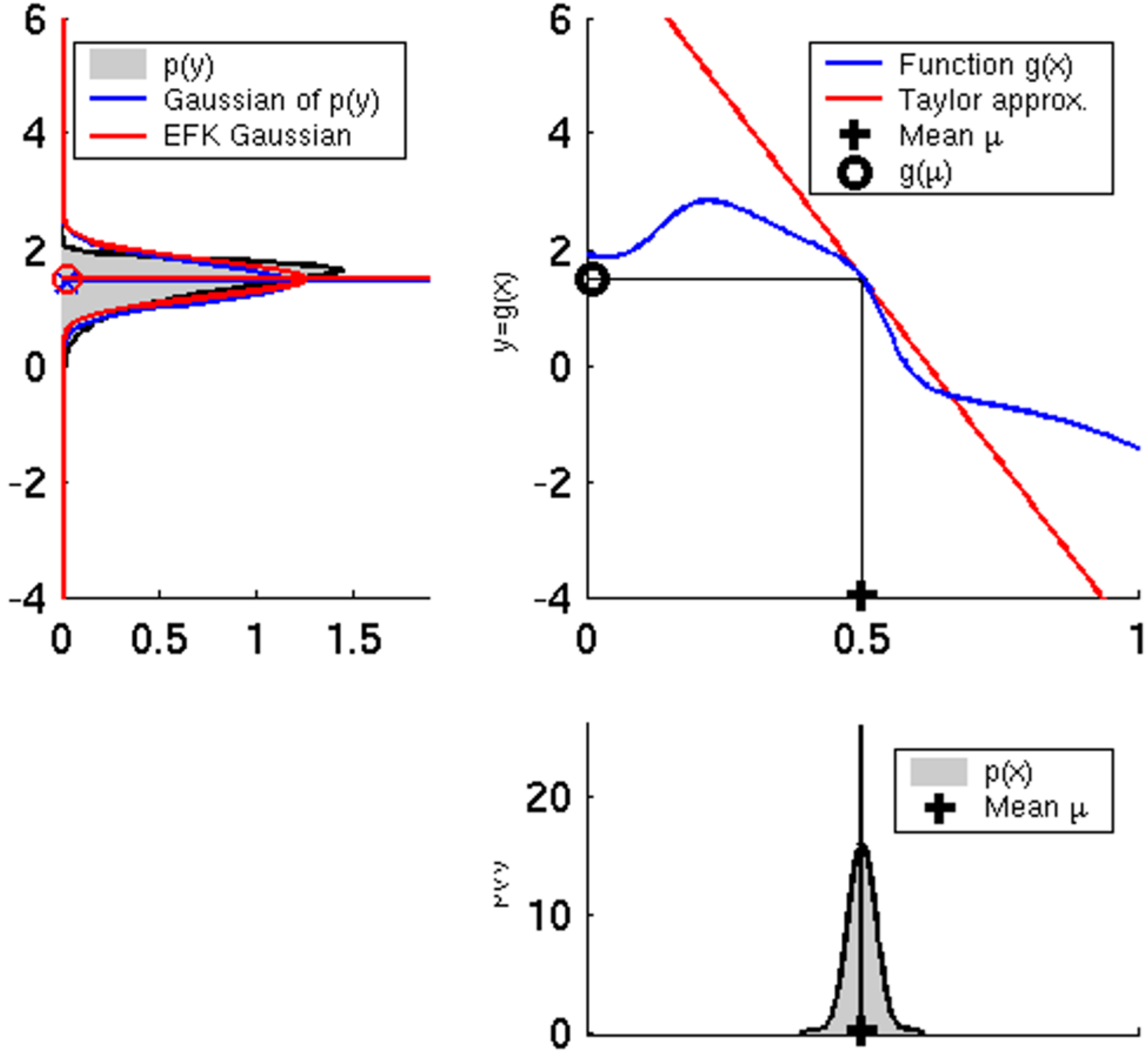
EKF Linearization (1)



EKF Linearization (2)



EKF Linearization (3)



Linearized Motion Model

- The linearized model leads to:

$$p(x_t \mid u_t, x_{t-1}) \approx \det\left(2\pi R_t\right)^{-\frac{1}{2}}$$
$$\exp\Big(-\frac{1}{2}\left(x_t - g(u_t,\mu_{t-1}) - G_t\left(x_{t-1}-\mu_{t-1}\right)\right)^T$$
$$R_t^{-1}\left(x_t - \underbrace{g(u_t,\mu_{t-1}) - G_t\left(x_{t-1}-\mu_{t-1}\right)}_{\text{linearized model}}\right)\Big)$$

- R_t describes the noise of the motion.

Linearized Observation Model

- The linearized model leads to:

$$p(z_t \mid x_t) = \det\left(2\pi Q_t\right)^{-\frac{1}{2}}$$
$$\exp\left(-\frac{1}{2}\left(z_t - h(\bar{\mu}_t) - H_t\left(x_t - \bar{\mu}_t\right)\right)^T\right.$$
$$\left.Q_t^{-1}\left(z_t - \underbrace{h(\bar{\mu}_t) - H_t\left(x_t - \bar{\mu}_t\right)}_{\text{linearized model}}\right)\right)$$

- Q_t describes the noise of the motion.

EKF Algorithm

1: $\textbf{Extended\_Kalman\_filter}(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$**:**

2: $\bar{\mu}_t = \underline{g(u_t, \mu_{t-1})}$

3: $\bar{\Sigma}_t = G_t\, \Sigma_{t-1}\, G_t^T + R_t$

4: $K_t = \bar{\Sigma}_t\, H_t^T (H_t\, \bar{\Sigma}_t\, H_t^T + Q_t)^{-1}$

5: $\mu_t = \bar{\mu}_t + K_t(z_t - \underline{h(\bar{\mu}_t})))$

6: $\Sigma_t = (I - K_t\, H_t)\, \bar{\Sigma}_t$

7: return μ_t, Σ_t

$A_t \leftrightarrow G_t$

$C_t \leftrightarrow H_t$

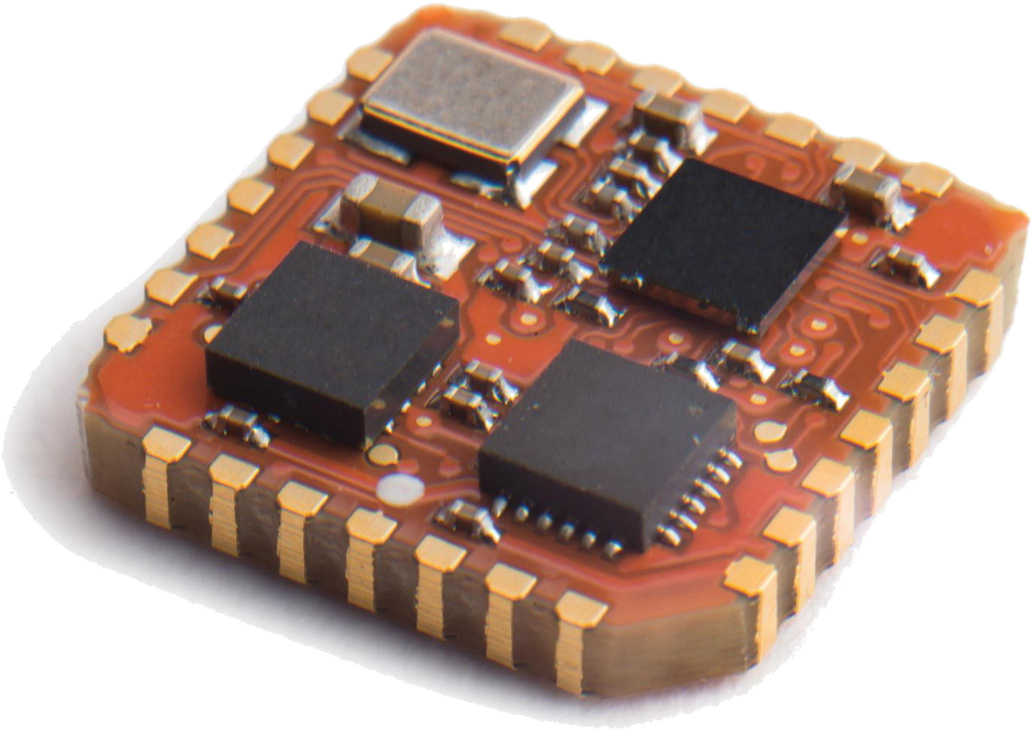
KF vs EKF

EKF Summary

- Extension of the Kalman Filter.
- One way to deal with nonlinearities.
- Performs local linearizations.
- Works well in practice for moderate nonlinearities.
- Large uncertainty leads to increased approximation error.

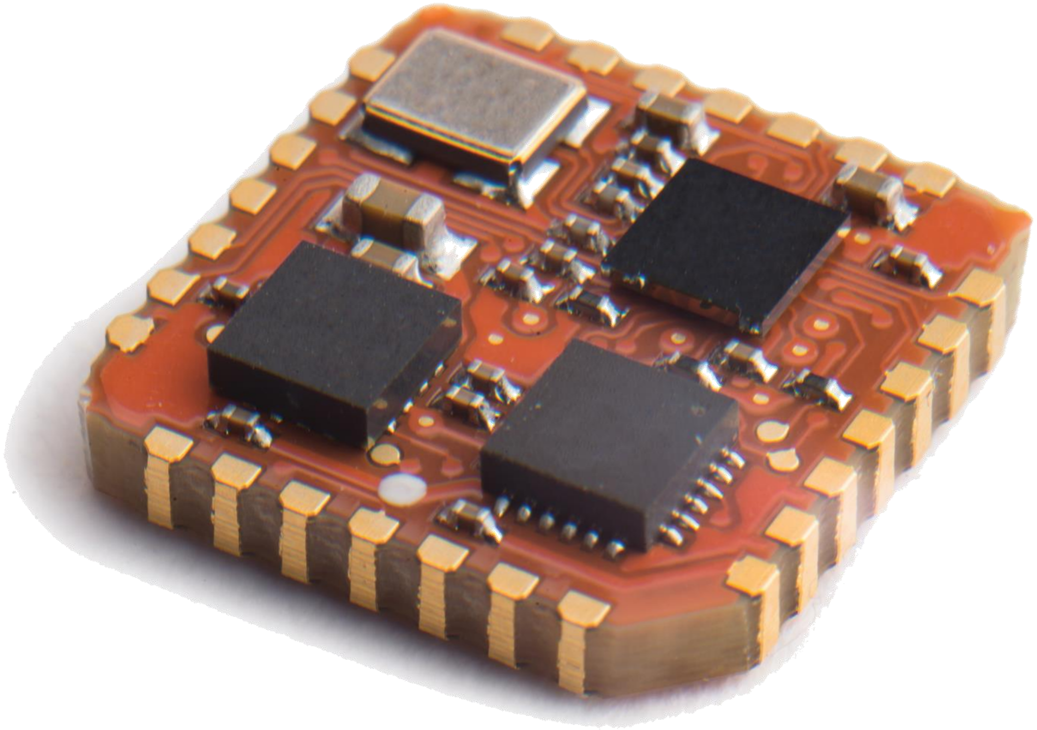
EKF Discussion

IMU



EKF Discussion

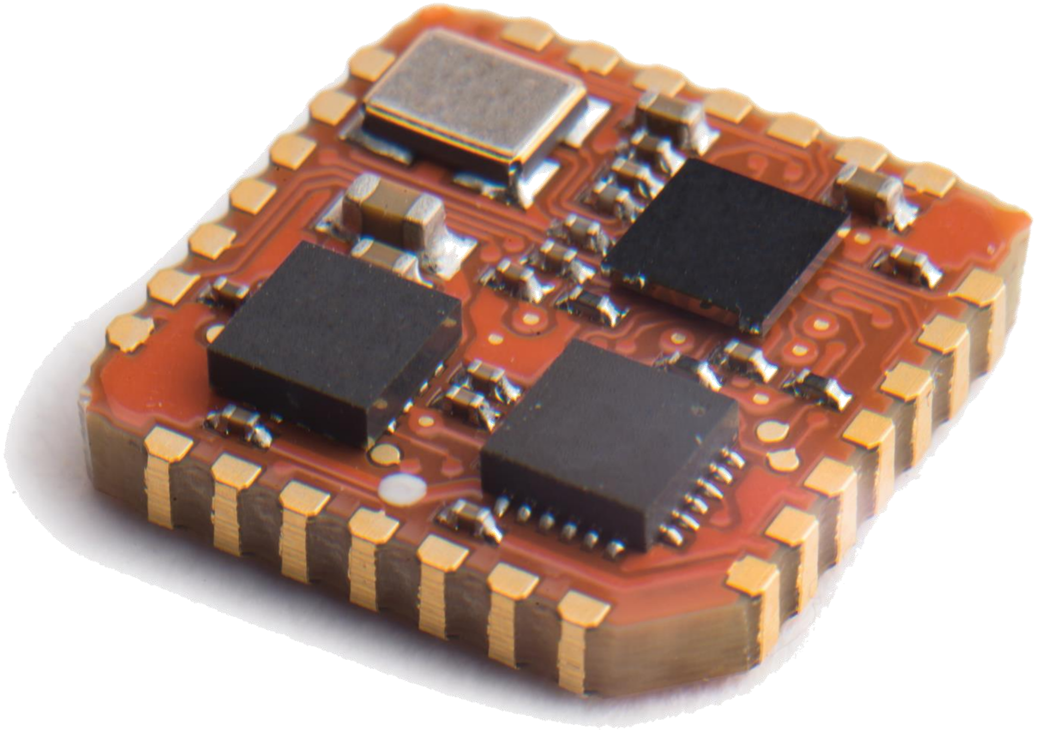
IMU + Compass



GPS

EKF Discussion

IMU + Compass



Camera

Find out more

- Thrun et al.: “Probabilistic Robotics”, Chapter 3
- Schön and Lindsten: “Manipulating the Multivariate Gaussian Density”
- “A New Extension of the Kalman Filter to Nonlinear Systems” by Julier and Uhlmann, 1995
- http://home.wlu.edu/~levys/kalman_tutorial/
- https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python
- http://www.autonomousrobotslab.com/the-kalman-filter.html
- http://aerostudents.com/files/probabilityAndStatistics/probabilityTheoryFullVersion.pdf
- http://www.cs.unc.edu/~welch/kalman/
- http://home.wlu.edu/~levys/kalman_tutorial/
- https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python
- http://www.autonomousrobotslab.com/literature-and-links.html

Code Examples and Tasks

- KF, EKF, UKF
  - Kalman Filter: https://github.com/unr-arl/drones_demystified/tree/master/matlab/state-estimation/kalman-filter

Thank you!

Please ask your question!