Autonomous Mobile Robot Design

Topic: State Estimation

Dr. Kostas Alexis (CSE)

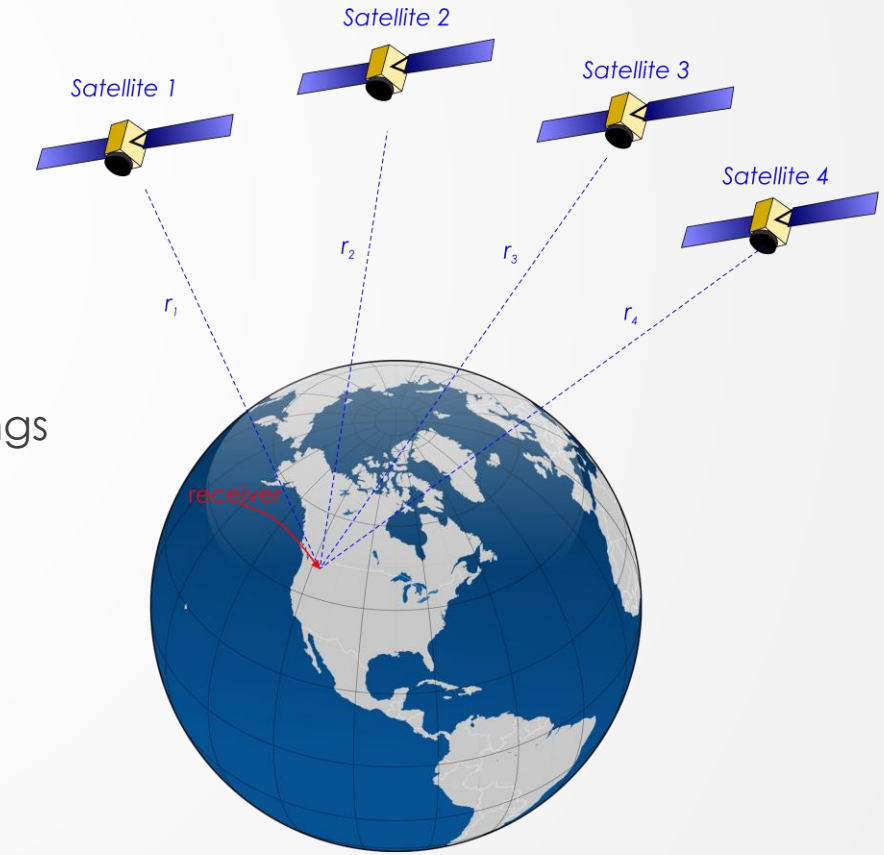
World state (or system state)

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

Parts of this talk are inspired from the edX lecture "Autonomous Navigation for Flying Robots" from TUM

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**
- **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):

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

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

Autonomous Mobile Robot Design

Topic: State Estimation – Recap on Probabilities

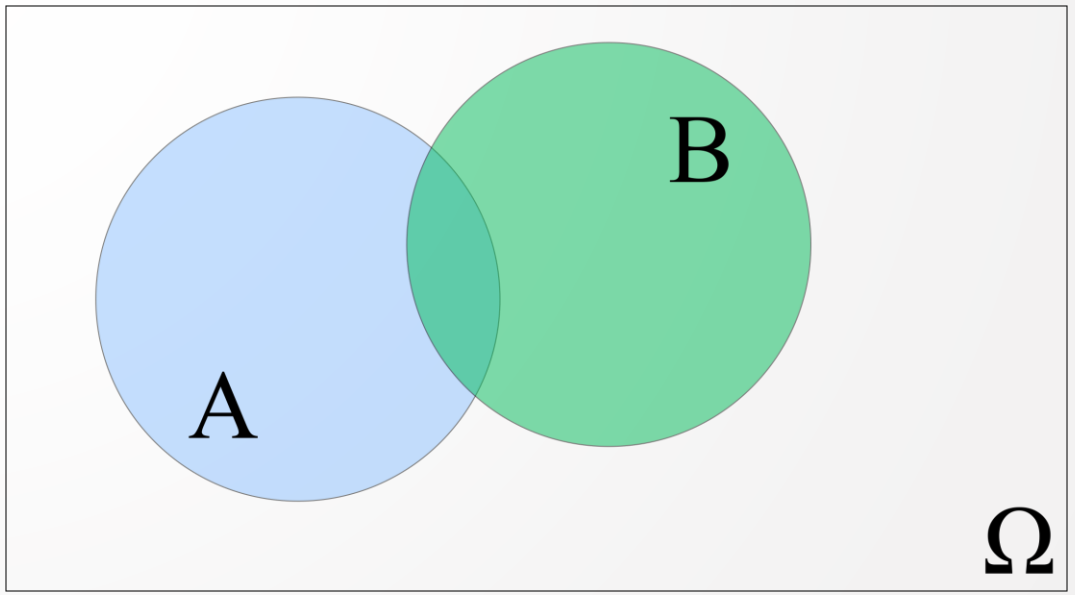
Dr. Kostas Alexis (CSE)

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 $\{x_1, x_2, \ldots, x_n\}$
- $P(X=x_i)$ is the probability that the random variable X takes on value x_i
- $P(.)$ is called the probability mass function

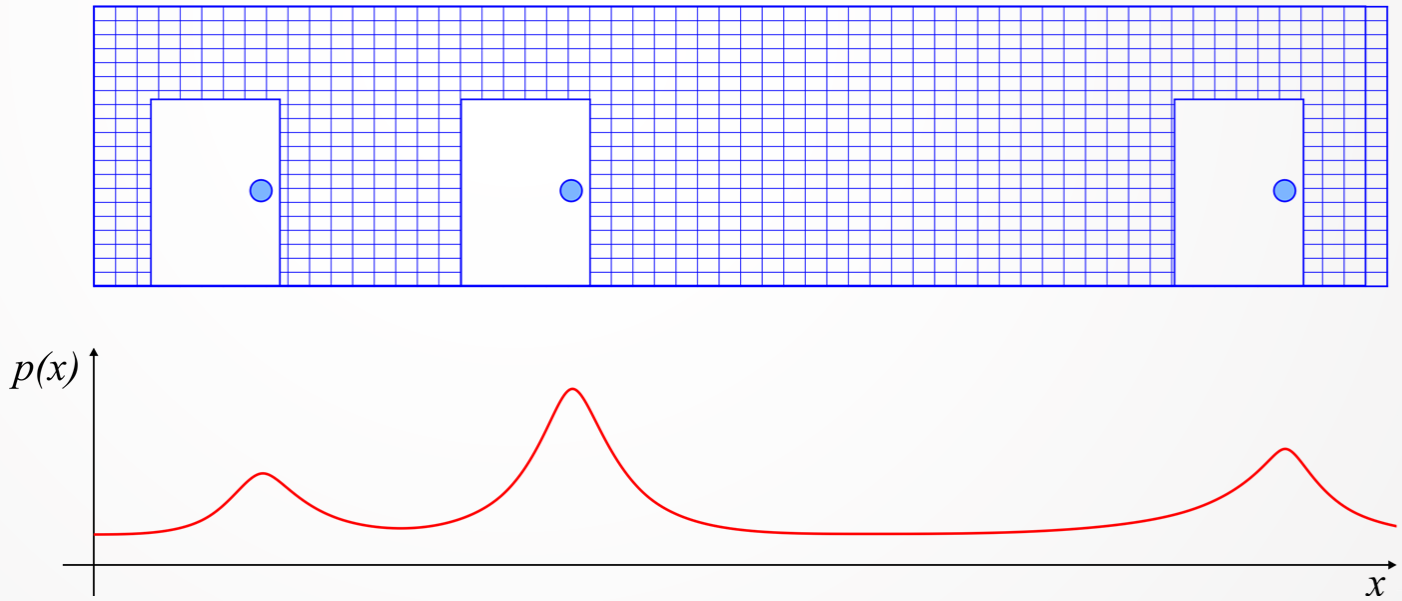
- Example: P(Room)=<0.6,0.3,0.06,0.03>, Room one of the office, corridor, lab, kitchen

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:

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

Autonomous Mobile Robot Design

Topic: State Estimation – Reasoning with Bayes Law

Dr. Kostas Alexis (CSE)

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 $\mathbf{P(z)}$ 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:** $P(x|z) = L(x|z)/P(z)$

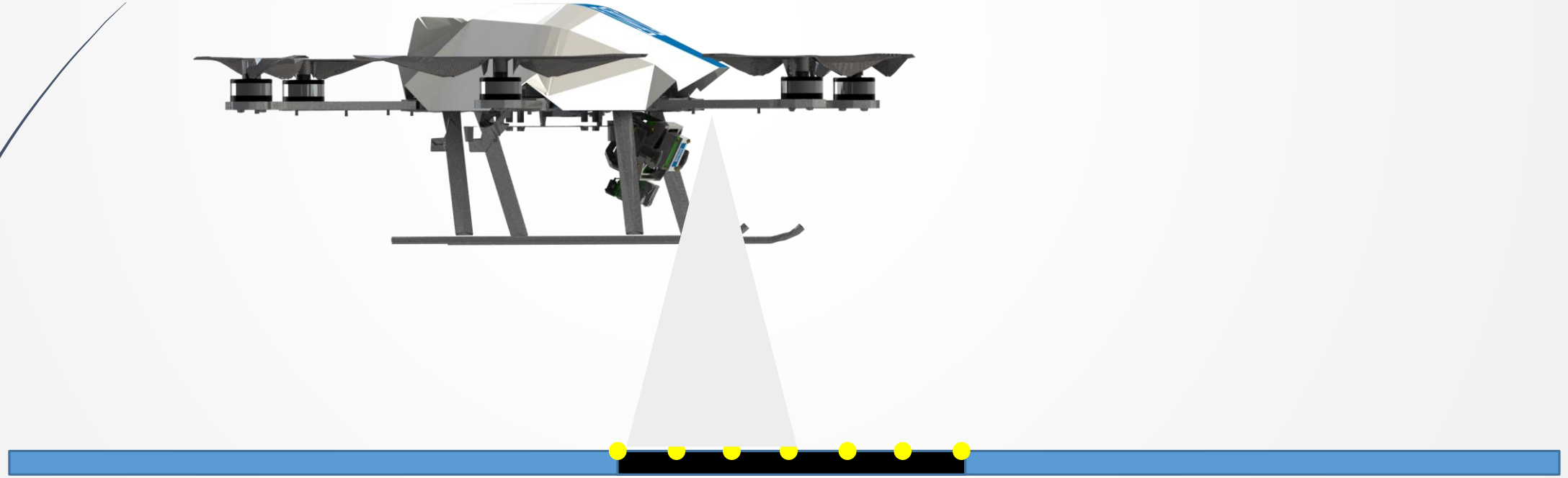
Normalization

- Direct computation of $\mathbf{P(z)}$ 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 $Z \in \{bright, \bar{bright}\}$
- Binary world state $X \in \{home, \bar{home}\}$
- Sensor model $P(Z = bright | X = home) = 0.6$
  $P(Z = bright | X = \bar{home}) = 0.3$
- Prior on world state $P(X = home) = 0.5$
- 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):

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

Autonomous Mobile Robot Design

Topic: State Estimation – Bayes Filter

Dr. Kostas Alexis (CSE)

Markov Assumption

- Observations depend only on current state

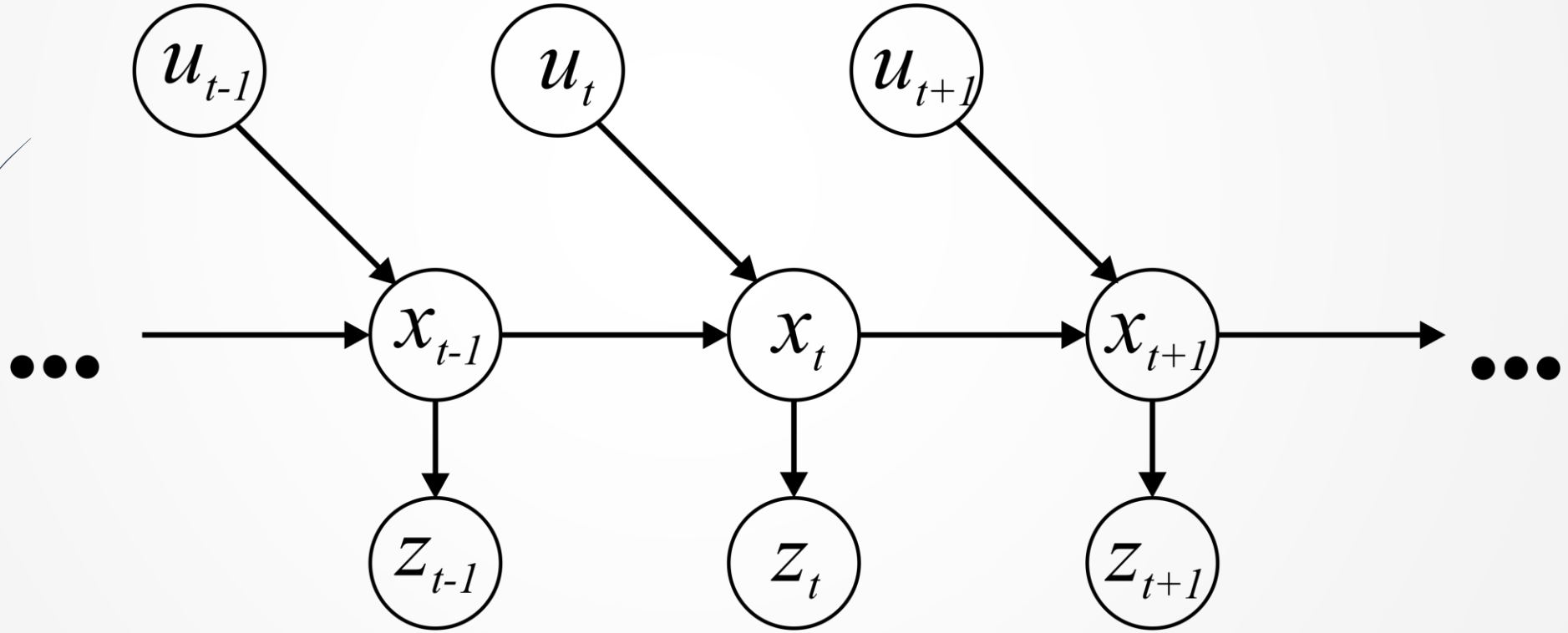
$$P(z_t|x_{0:t}, z_{1:t-1}, u_{1:t}) = P(z_t|x_t)$$

- 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:

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

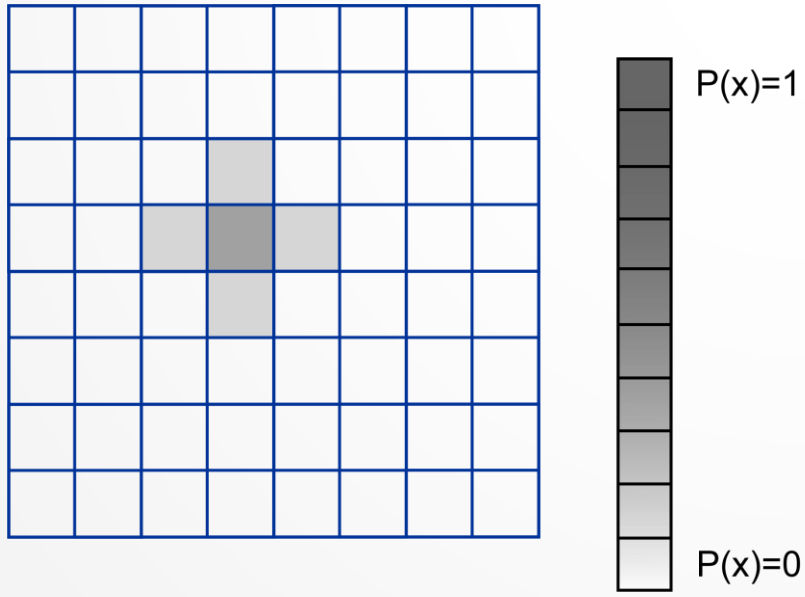
  - η is a normalization factor to ensure that the probability is maximum 1.

Notes

- Bayes filters also work on continuous state spaces (replace sum by integral).
- Bayes filter also works when actions and observations are asynchronous.

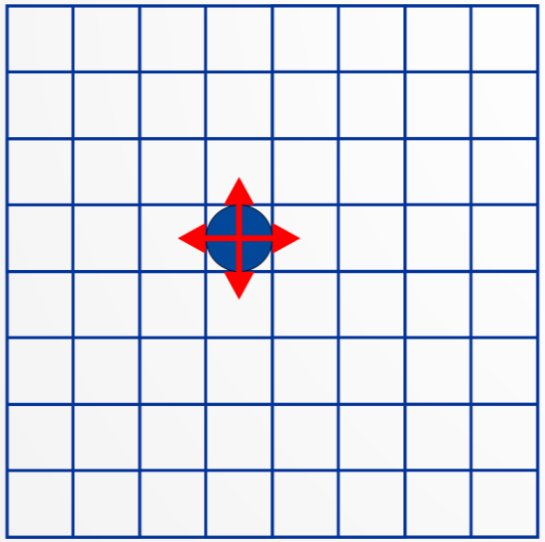
Example: Localization

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



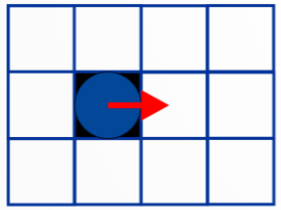
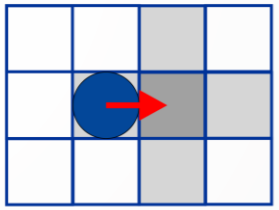
Example: Localization

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



Example: Localization

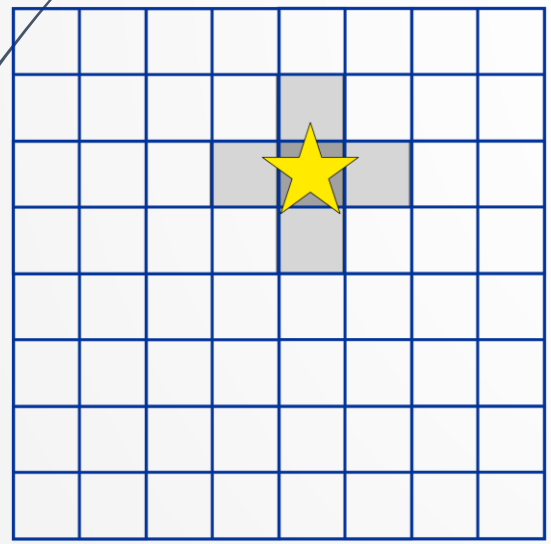
- Action
- Robot can move one cell in each time step
- Actions are not perfectly executed
- Example: move east

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

- 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

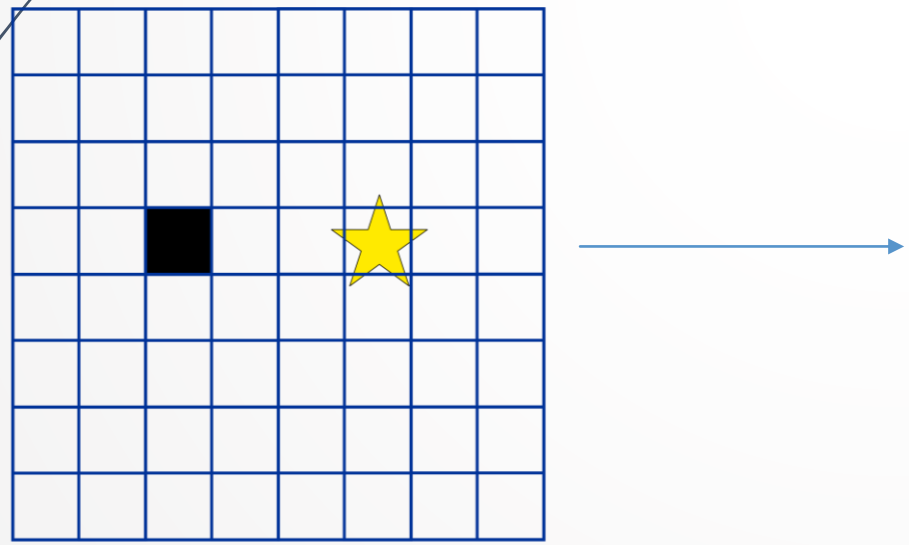


Example: Localization

- Let's start a simulation run…

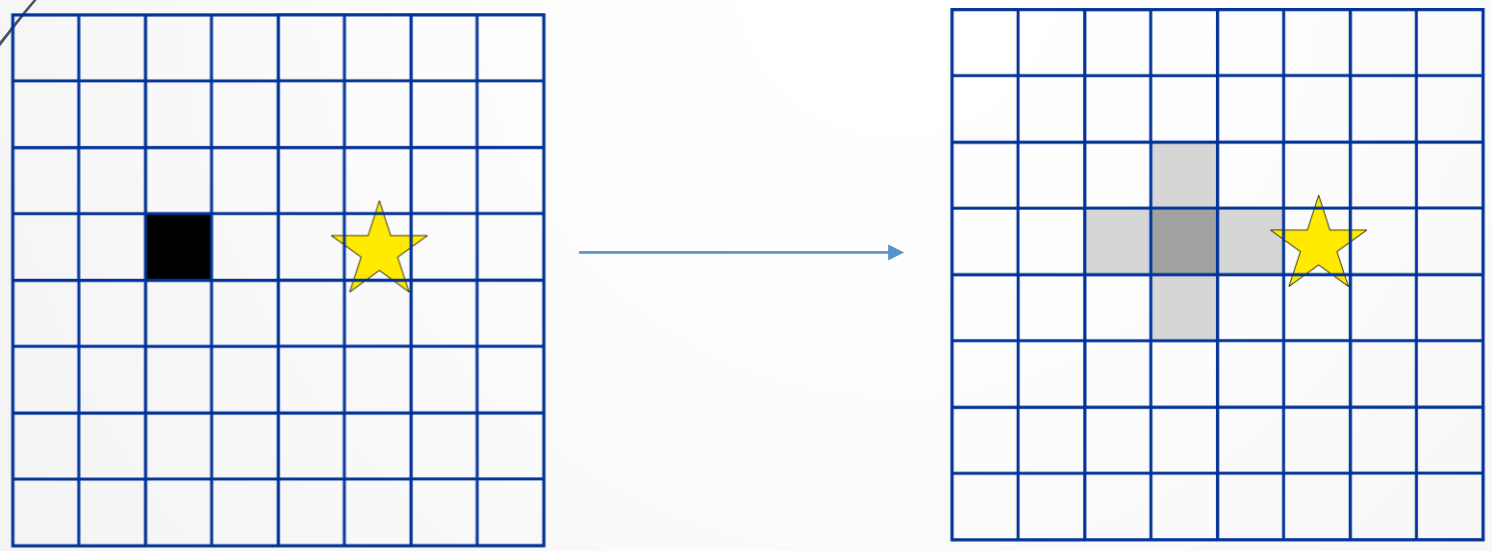
Example: Localization

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



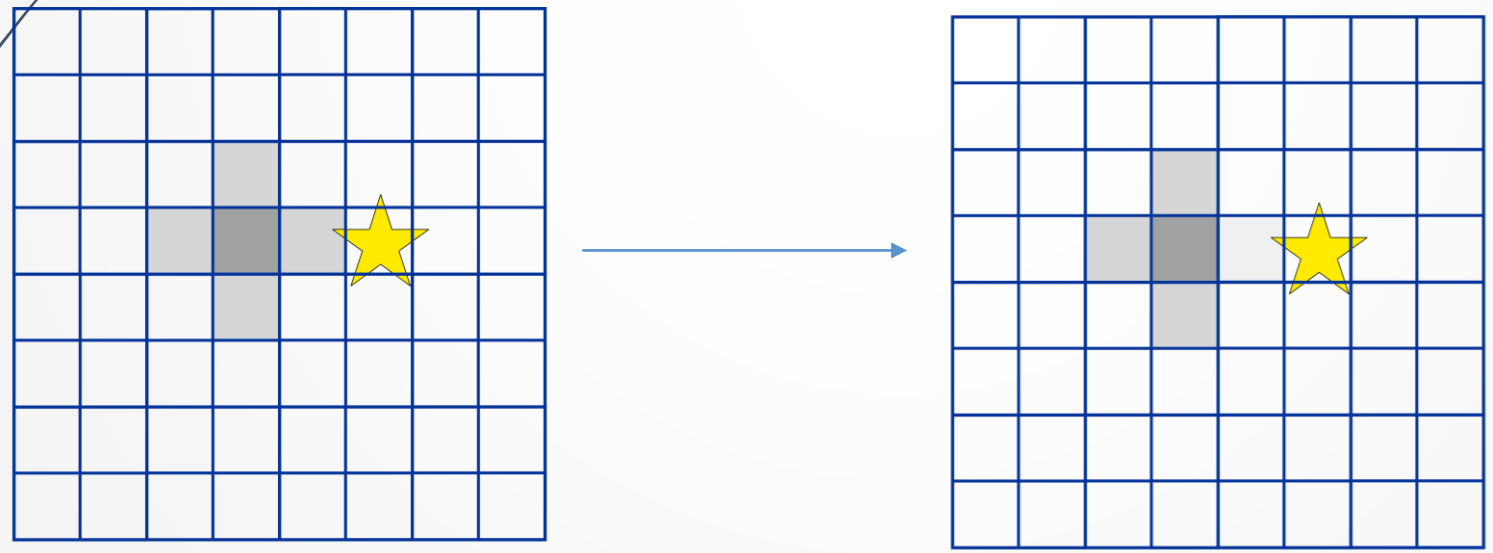
Example: Localization

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



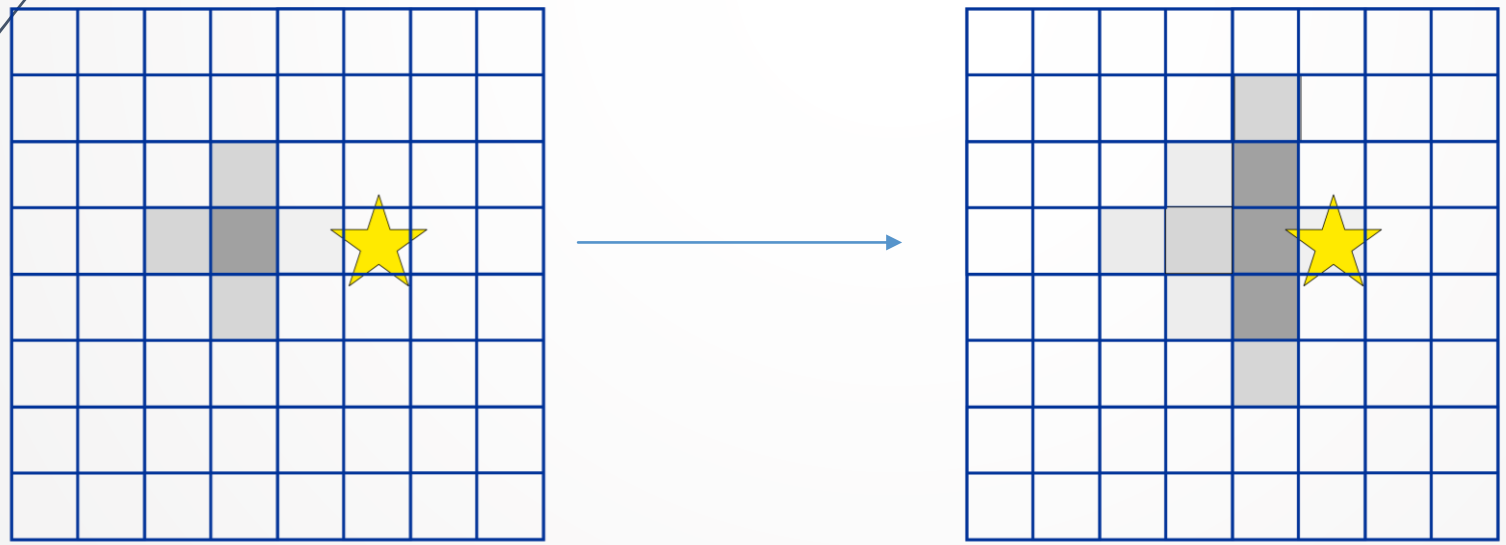
Example: Localization

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



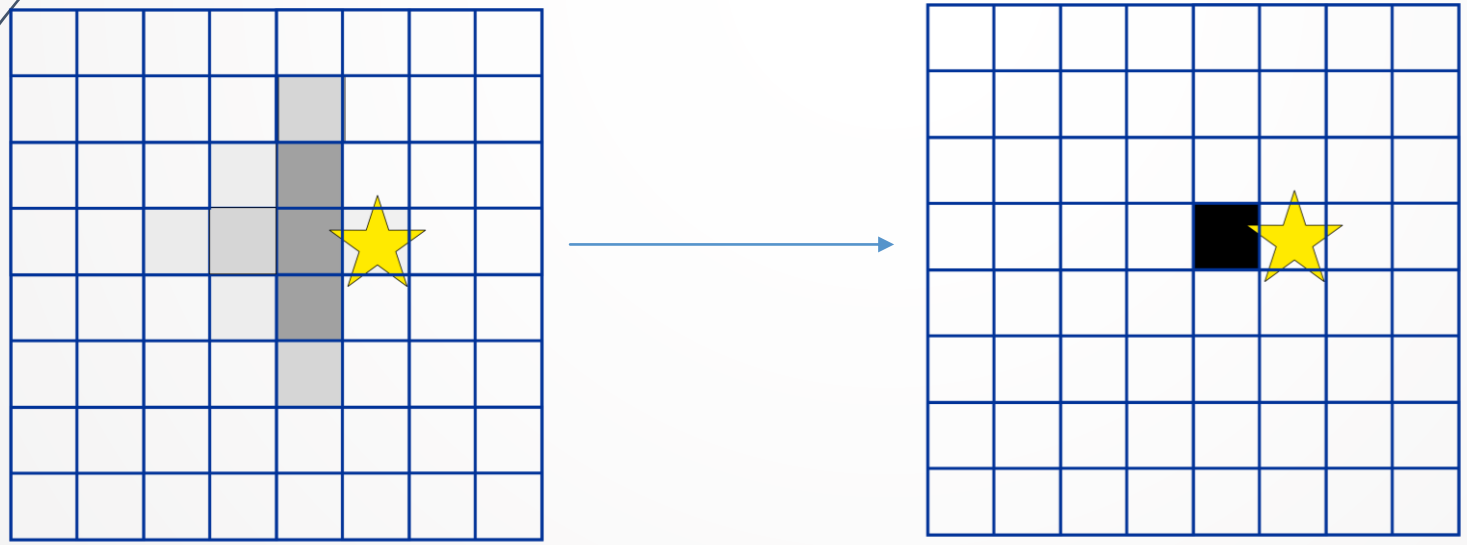
Example: Localization

- 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?



Autonomous Mobile Robot Design

Topic: State Estimation – Kalman Filter

Dr. Kostas Alexis (CSE)

Kalman Filter

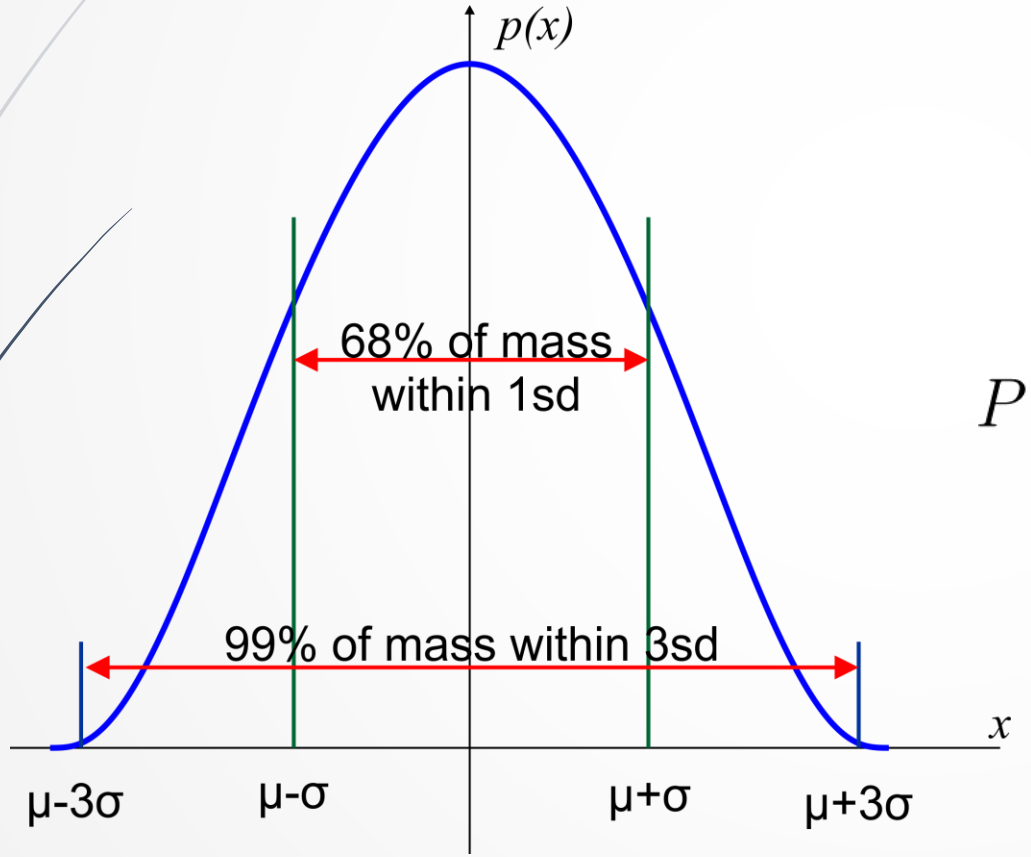
- Bayes filter is a useful tool for state estimation.
- Histogram filter with grid representation is not very efficient.
- How can we represent the state more efficiently?

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

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

Properties of Normal Distributions

- Linear transformation – remains Gaussian

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

- Intersection of two Gaussians – remains Gaussian

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

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

Properties of Normal Distributions

- Linear transformation – remains Gaussian

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

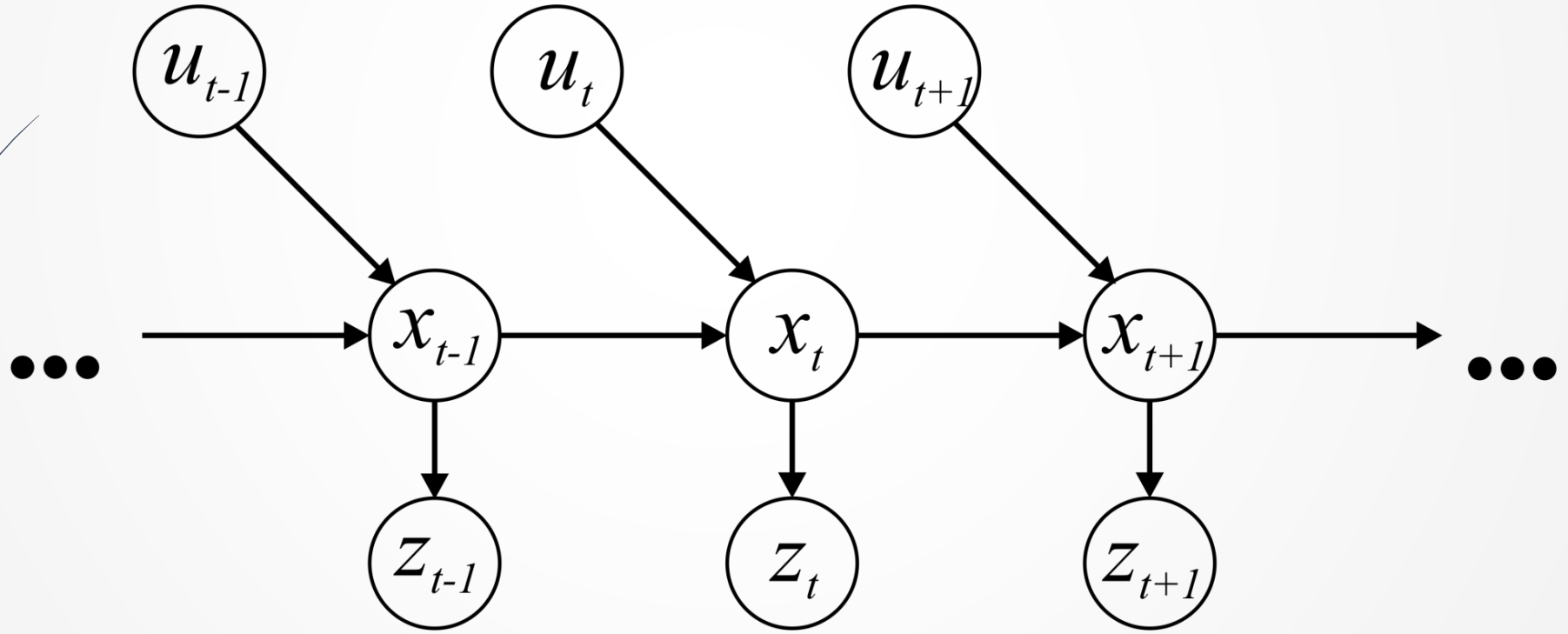
- Intersection of two Gaussians – remains Gaussian

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

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

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) with a Gaussian

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

Linear Process Model

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

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

- 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

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

- 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 $\underset{n\text{x}n}{\mathbf{A}}\mathbf{x}_{t-1} + \underset{n\text{x}l}{\mathbf{B}}\mathbf{u}_t + \epsilon_t$
- Measurement equation $\mathbf{z}_t = \underset{n\text{x}k}{\mathbf{C}}\mathbf{x}_t + \delta_t$

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.

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

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

$$\bar{\mu}_t = \mathrm{A}\mu_{t-1} + \mathrm{B}\mathrm{u}_t$$
$$\bar{\Sigma}_t = \mathrm{A}\Sigma\mathrm{A}^\top + \mathrm{Q}$$

Correction

$$\mu_t = \bar{\mu}_t + \mathrm{K}_t(\mathrm{z}_t - \mathrm{C}\bar{\mu}_t)$$
$$\Sigma_t = (\mathrm{I} - \mathrm{K}_t\mathrm{C})\bar{\Sigma}_t$$
$$\mathrm{K}_t = \bar{\Sigma}_t\mathrm{C}^\top(\mathrm{C}\bar{\Sigma}_t\mathrm{C}^\top + \mathrm{R})^{-1}$$

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.

Python KF Implementation

```
from numpy import *
import numpy as np
from numpy.linalg import inv
from KalmanFilterFunctions import *

# time step of mobile movement
dt = 0.1

# Initialization of state matrices
X = array([[0.0], [0.0], [0.1], [0.1]])
P = diag((0.01, 0.01, 0.01, 0.01))
A = array([[1, 0, dt, 0], [0, 1, 0, dt], [0, 0, 1, 0], [0, 0, 0, 1]])
Q = eye(X.shape[0])
B = eye(X.shape[0])
U = zeros((X.shape[0],1))

# Measurement matrices
Y = array([[X[0,0] + abs(random.randn(1)[0])], [X[1,0] + abs(random.randn(1)[0])]])
H = array([[1, 0, 0 , 0], [0, 1, 0, 0]])
R = eye(Y.shape[0])

# Number of iterations in Kalman Filter
N_iter = 50

# Applying the Kalman Filter
for i in range(0, N_iter):
    (X, P) = kf_predict(X, P, A, Q, B, U)
    (X, P, K, IM, IS, LH)  = kf_update(X, P, Y, H, R)
    Y = array([[X[0,0] + abs(0.1 * random.randn(1)[0])], [X[1, 0] + abs(0.1 * random.randn(1)[0])]])
```

```
from numpy import dot, sum, tile, linalg, log, pi, exp
from numpy.linalg import inv, det

def kf_predict(X, P, A, Q, B, U):
    X = dot(A, X) + dot(B, U)
    P = dot(A, dot(P, A.T)) + Q
    return(X, P)

def gauss_pdf(X, M, S):
    if M.shape[1] == 1:
        DX = X - tile(M, X.shape[1])
        E = 0.5 * sum(DX * (dot(inv(S), DX)), axis=0)
        E - E + 0.5 * M.shape[0] * log(2 * pi) + 0.5 * log(det(S))
        P = exp(-E)
    elif X.shape[1] == 1:
        DX = tile(X, M.shape[1] - M)
        E = 0.5 * sum(DX * (dot(inv(S), DX)), axis =0)
        E = E + 0.5 * M.shape[0] * log(2 * pi) + 0.5 * log(det(S))
        P = exp(-E)
    else:
        DX = X - M
        E = 0.5 * dot(DX.T, dot(inv(S), DX))
        E = E + 0.5 * M.shape[0] * log(2 * pi) + 0.5 * log(det(S))
        P = exp(-E)
    return (P[0],E[0])

def kf_update(X, P, Y, H, R):
    IM = dot(H, X)
    IS = R + dot(H, dot(P, H.T))
    K = dot(P, dot(H.T, inv(IS)))
    X = X + dot(K, (Y-IM))
    P = P - dot(K, dot(IS, K.T))
    LH = gauss_pdf(Y, IM, IS)
    return (X,P,K,IM,IS,LH)
```

http://www.kostasalexis.com/the-kalman-filter.html

Assignment 1: Estimating a Random Constant

- The goal of this task is to estimate a scalar random constant, which may be a voltage level. Let's assume that we have the ability to take measurements of the constant, but that the measurements are corrupted by 0.1 Volt RMS white measurement noise (e.g. our analog to digital converter is not very accurate). In this example, the process is governed by the linear difference equation:

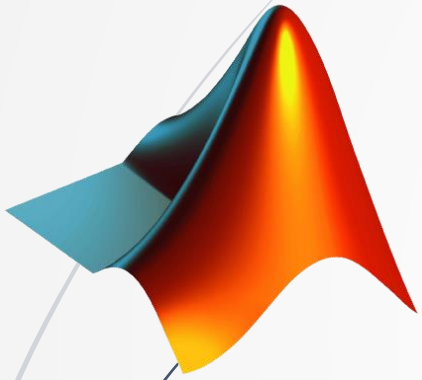
$$x_k = Ax_{k-1} + Bu_{k-1} + w_k, A = 1, B = 0$$

- with a measurement z that is:

$$z_k = Hx_k + v_k, H = 1$$

- The state does not change from step to step, this is why A=1. There is no control input, therefore B =0,u=0. Our noisy measurement, directly measures the state - therefore H=1. Notice that the subscript k was dropped in several places because the respective parameters remain constant in our simple model.
  - **Programming Language:** preferably one **Python, MATLAB, C++, or JAVA**
  - **Form of the Report:** 1. Brief report with the code and the relevant plots indicating the correct estimate of the constant, 2. Comments on what you understood about the filter operation.
  - **Deadline:** March 11, 2016

Code Examples and Tasks



- KF, EKF, UKF
  - Kalman Filter: https://github.com/unr-arl/autonomous_mobile_robot_design_course/tree/master/matlab/state-estimation/kalman-filter
  - Extended Kalman Filter: https://github.com/unr-arl/autonomous_mobile_robot_design_course/tree/master/matlab/state-estimation/extended-kalman-filter
  - Unscented Kalman Filter: https://github.com/unr-arl/autonomous_mobile_robot_design_course/tree/master/matlab/state-estimation/unscented-kalman-filter

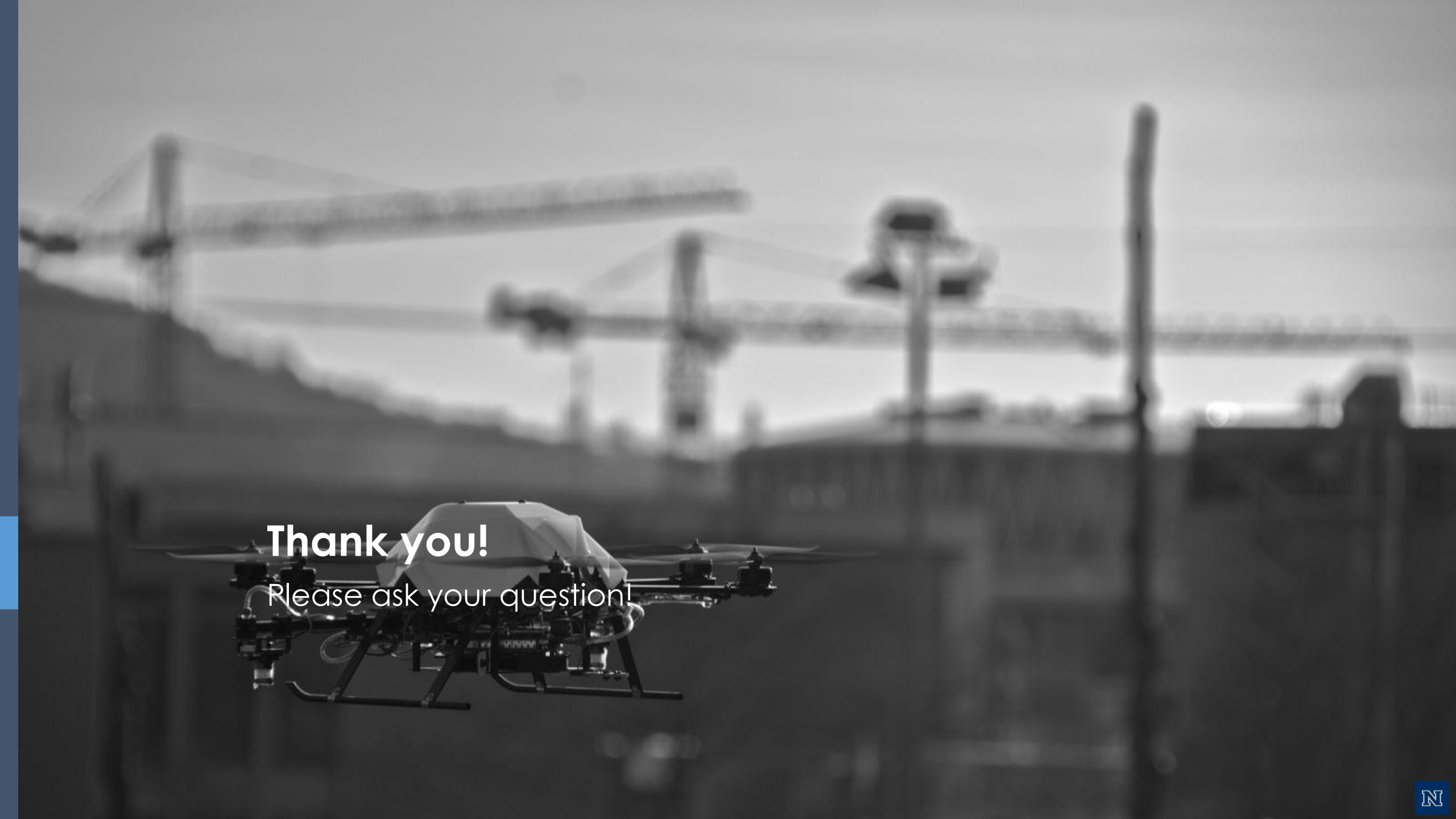
How does this apply to my project?

- State estimation is the way to use robot sensors to infer the robot state. You will use it for estimating your robot pose or its map, to track and object and be able to follow it etc.

Find out more

- http://www.kostasalexis.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.kostasalexis.com/literature-and-links.html



Thank you!

Please ask your question!