Intelligent Robotics Systems

Armin Biess

Ben-Gurion University of the Negev

Probabilistic State Estimation I

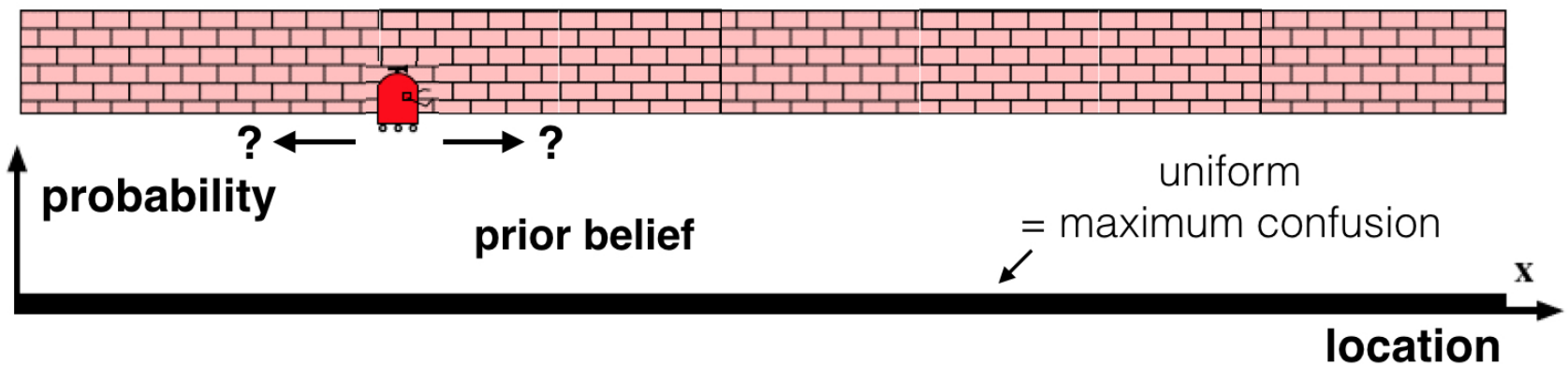
Bayes and Gaussian Filters

Mobile robot localization problem

Localization is the ability for a machine to locate itself in space.
GPS: accuracy: ~ 5m
Localization algorithm: accuracy: $\sim 5 - 10$cm

Problem: Mobile robot in a one-dim world

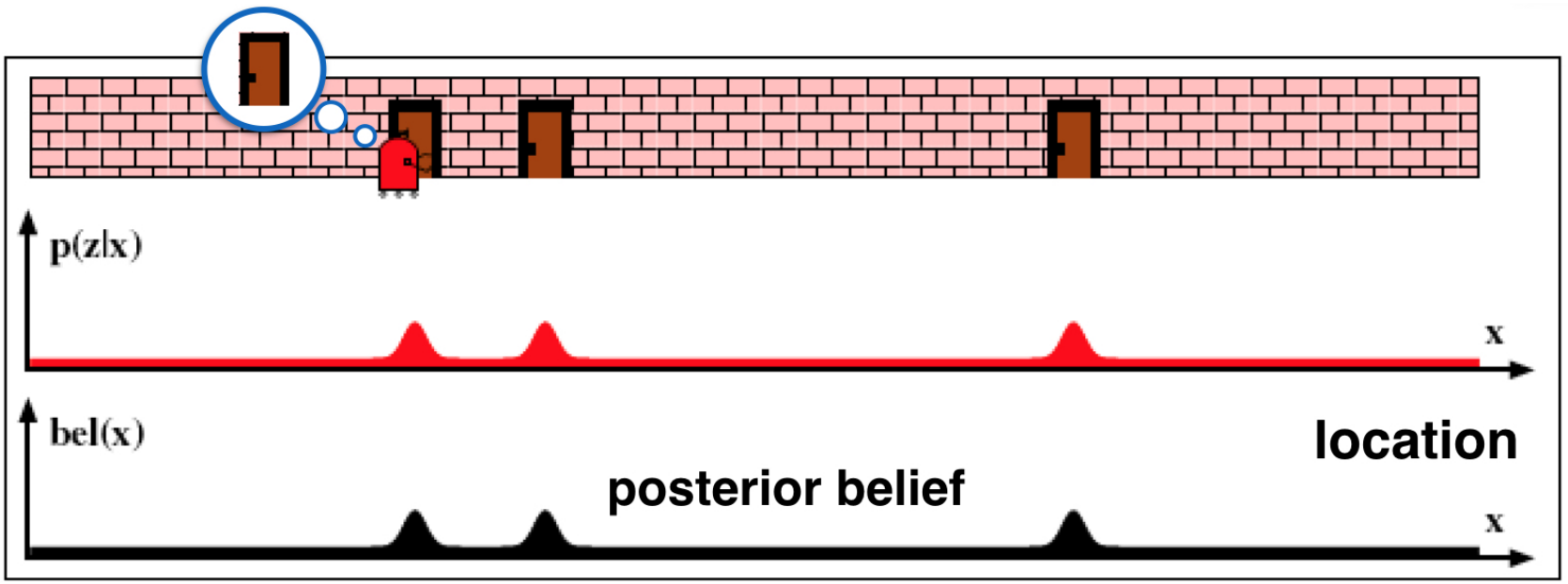
Step 1: **Initial belief** - robot does not know where it is, it can move right or left



prior belief = initial belief = uniform probability function

Localization problem

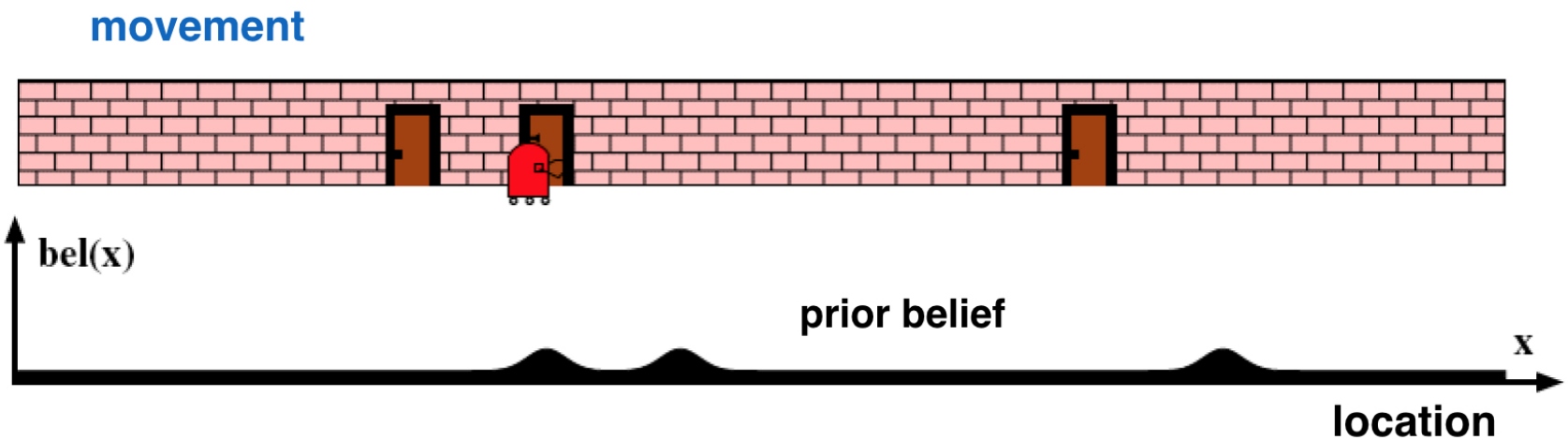
Step 2: **1. Measurement** - assume three identical landmarks (doors) are given and the robot senses a door.
How does it affect the robot's belief?



The robot assigns greater belief to the door locations, whereas all of the other locations have less belief $\rightarrow$ posterior belief

Localization Problem

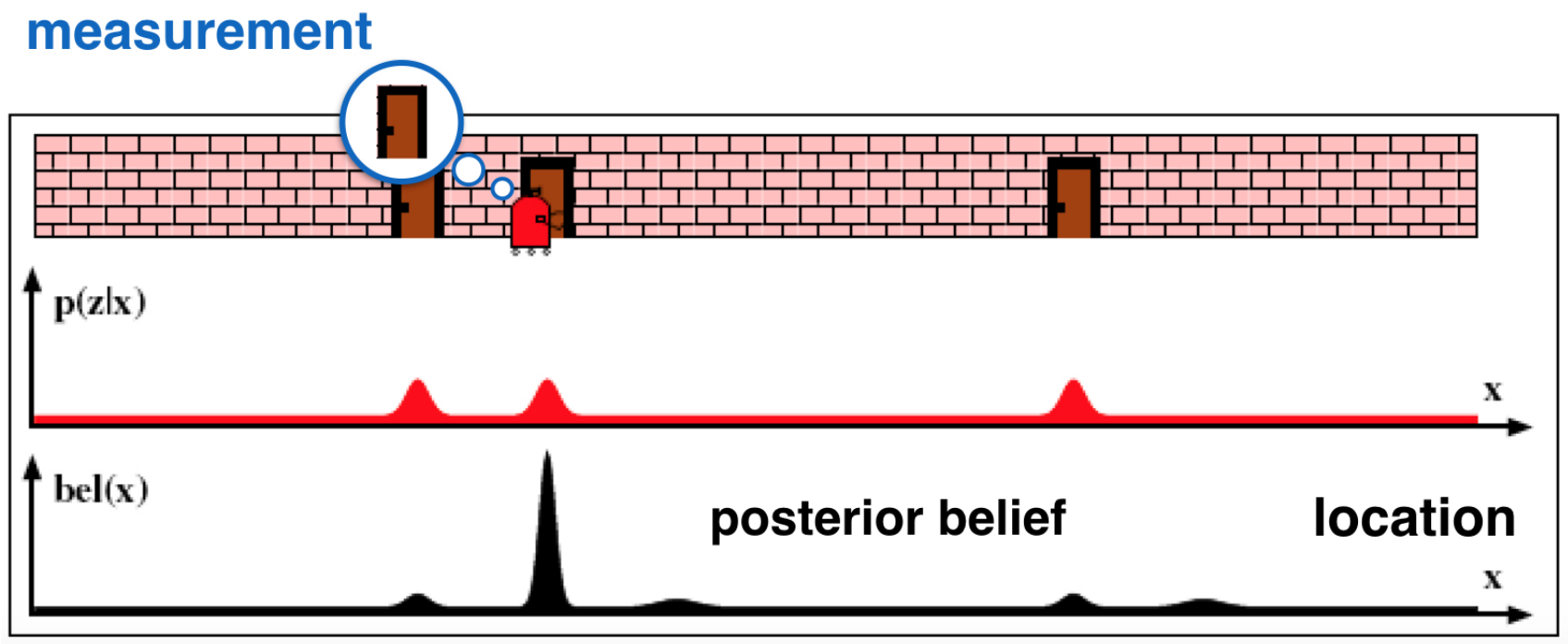
Step 3: **Movement** - if the robot moves to the right a certain distance, the belief is shifted according to the motion. .



Due to uncertainty in robot motion the bumps do not shift perfectly, but get flattened. This is described mathematically by a convolution.

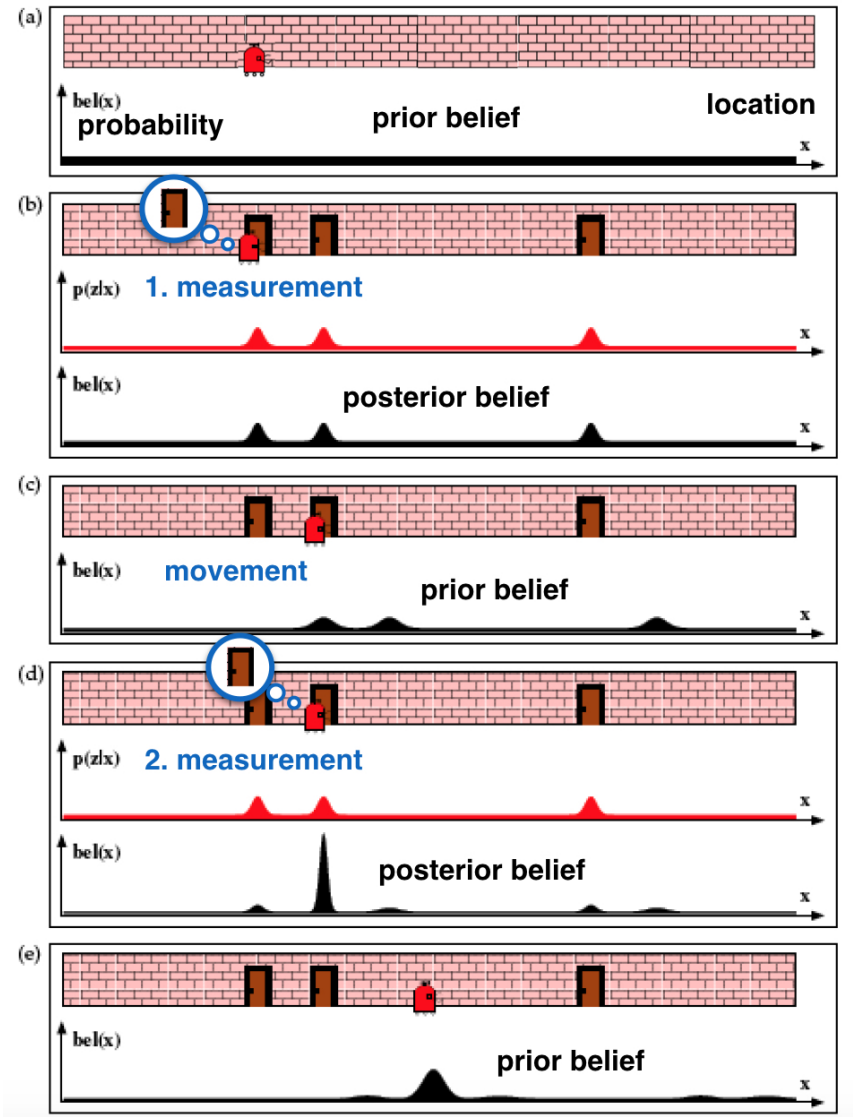
Localization Problem

Step 4: **2. Measurement**: robot senses another door .



A sharp peak at the second door appears and the robot is quite confident of having localized itself.

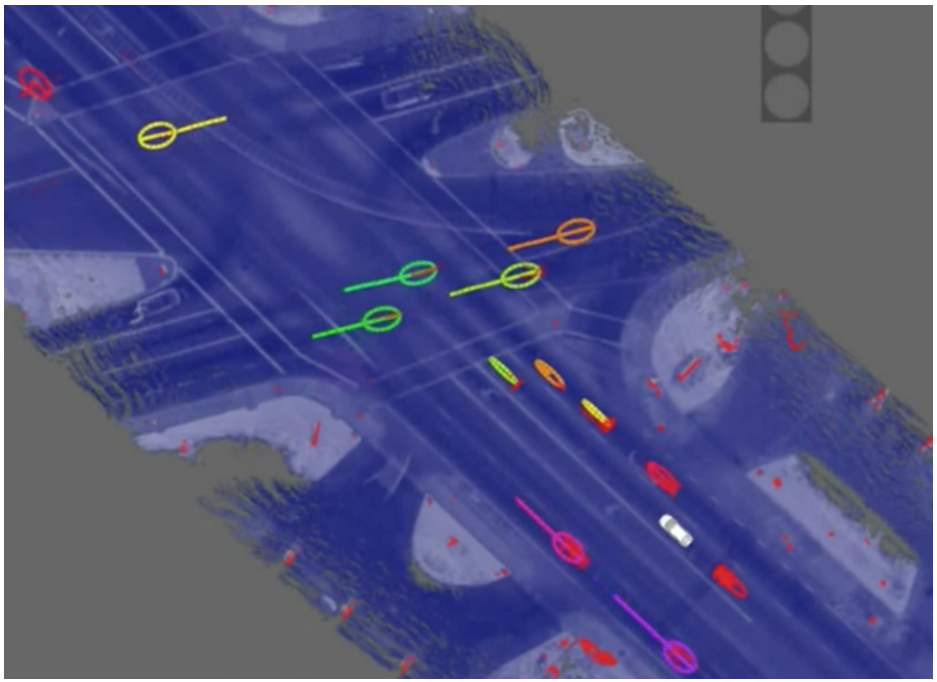
Localization Problem



mobile robot localization is an example of **probabilistic state estimation**

Tracking Problem

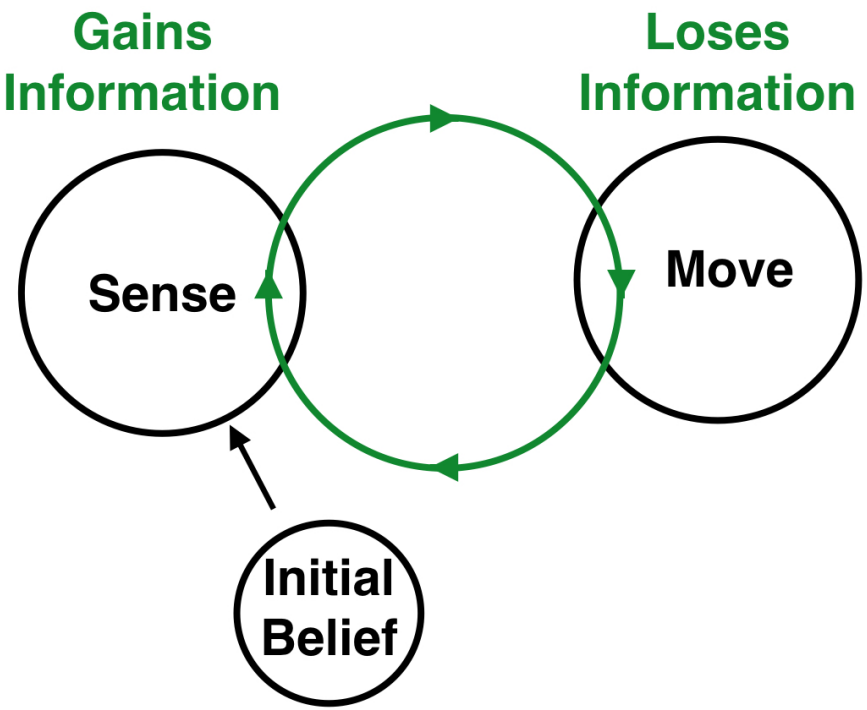
Tracking is the problem of state estimation (position,velocity, etc) of robots, cars, rockets, objects, etc.



Robot - sense and move

As we observe in the mobile robot localization and tracking problem there are two fundamental types of interactions between a robot and its environment:

1. **sensor measurements** (**observations**, **percepts**): robot obtains information about the state of its environment
2. **control actions**: robot asserts actively forces on its environment



next task: model sense, move and initial belief mathematically

Actions

- the robot turns its wheels to move
- the robots uses its manipulator to grasp an object
- actions that are carried out by other agents
- plants grow over time (robots in agriculture)

- actions are never carried out with absolute certainty
- in contrast to measurements, actions generally increase the uncertainty

How to incorporate such actions?

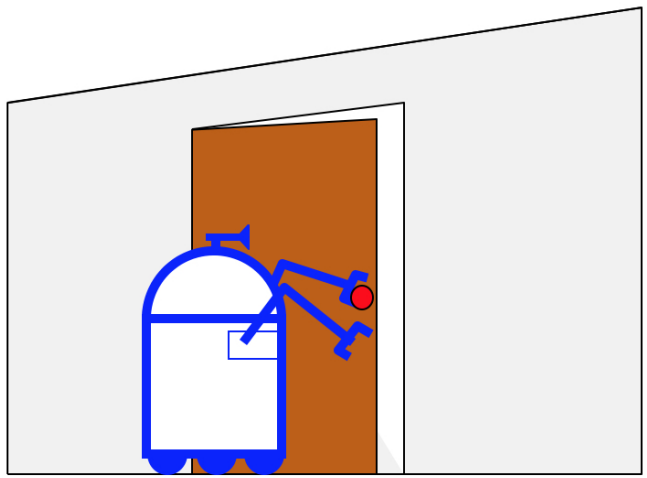
Modeling actions

- to incorporate the outcome of an action u into the current estimate (or belief), we use a **state transition pdf**

$$P(x|u, x')$$

- $P(x|u, x')$ specifies the pdf that executing u changes the state from x' to x

Example: State estimation for action = closing the door



Given:
discrete states: $x = \{\text{door open, door closed}\}$
chosen action: $u = \{\text{close door}\}$
state transition probability: $P(x|u, x')$
prior: $P(x)$

Wanted:
state estimate or belief, $P(x|u)$, i.e., probability to be in state x given action u

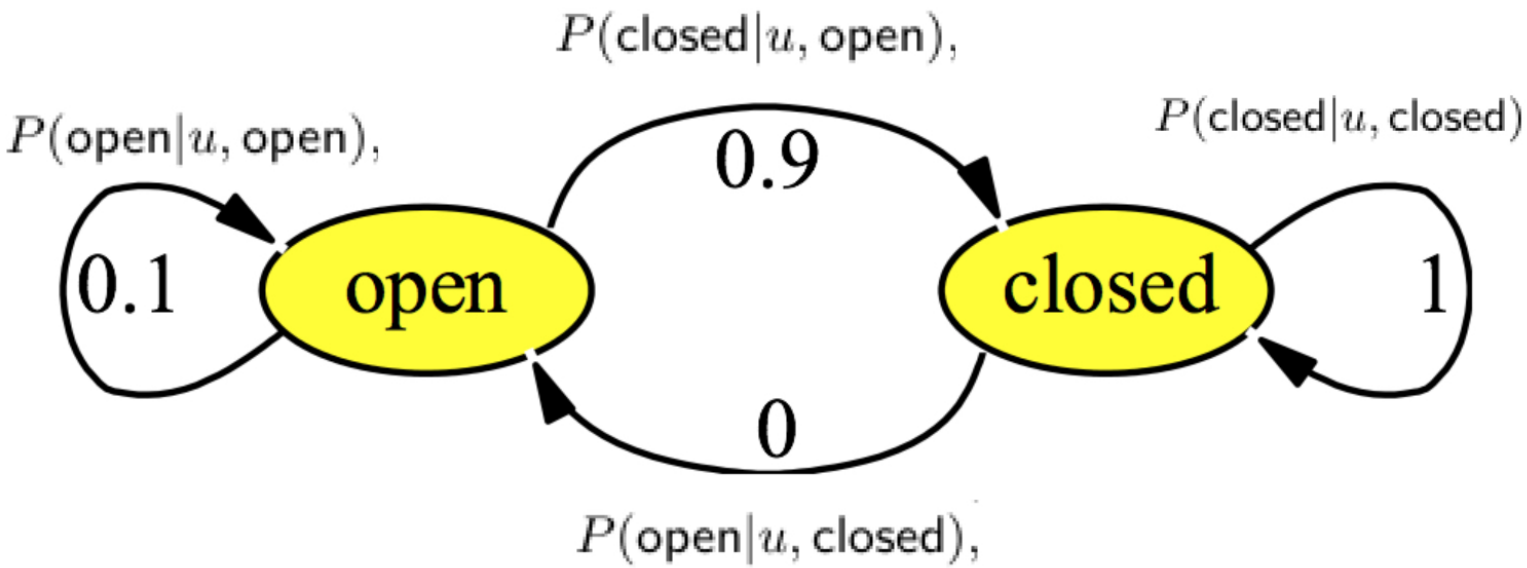
State transitions

Define $P(x|u, x')$ for:

- action $u = \{\text{close door}\}$
- and the two states $x_1 = \{\text{door open}\}$ and $x_2 = \{\text{door closed}\}$

We need to define four state transition probabilities:

$$P(x|u, x') = \begin{pmatrix} P(\text{open}|u, \text{open}) & P(\text{closed}|u, \text{open}) \\ P(\text{open}|u, \text{closed}) & P(\text{closed}|u, \text{closed}) \end{pmatrix} = \begin{pmatrix} 0.1 & 0.9 \\ 0 & 1 \end{pmatrix}$$



Integrating the outcome of actions

The state estimate or belief, $P(x|u)$, is obtained as follows:

- continuous case:

$$P(x|u) = \int P(x|u,x')P(x')dx'$$

- discrete case:

$$P(x|u) = \sum_{x'} P(x|u,x')P(x')$$

What is $P(x)$ in our example?
We can set $P(\text{open}) = P(\text{closed}) = 0.5$

State estimate for action = closing the door

action: $u = \{\text{close door}\}$

state transition:

$$P(x|u,x') = \begin{pmatrix} P(\text{open}|u,\text{open}) & P(\text{open}|u,\text{closed}) \\ P(\text{closed}|u,\text{open}) & P(\text{closed}|u,\text{closed}) \end{pmatrix} = \begin{pmatrix} 0.1 & 0 \\ 0.9 & 1 \end{pmatrix}$$

prior: $P(\text{open}) = P(\text{closed}) = 0.5$

$$\begin{aligned} P(\text{closed}|u) &= \sum P(x|u,x')P(x') \\ &= P(\text{closed}|u,\text{open})P(\text{open}) + P(\text{closed}|u,\text{closed})P(\text{closed}) \\ &= \frac{9}{10}\cdot\frac{1}{2} + 1\cdot\frac{1}{2} = \frac{19}{20} \end{aligned}$$

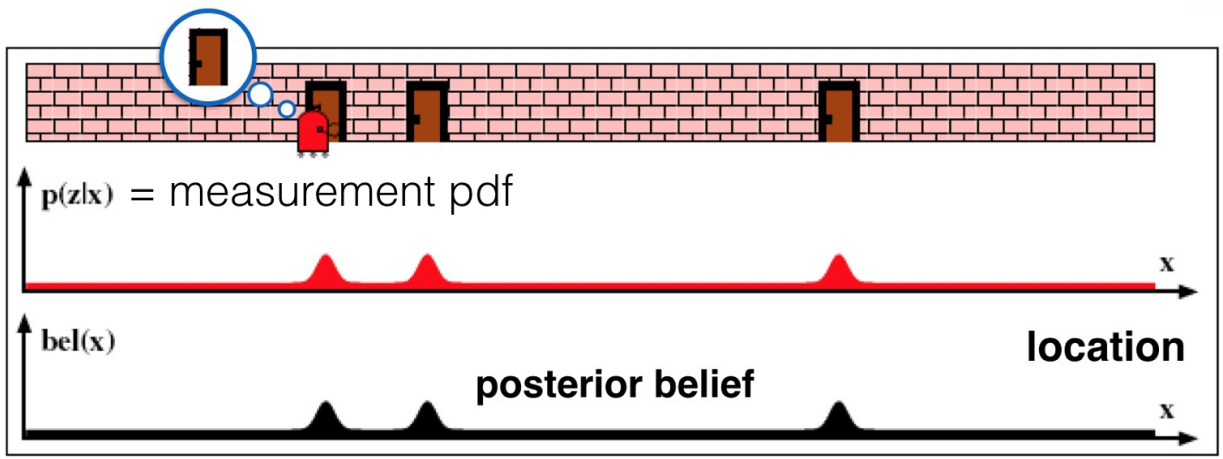
$$\begin{aligned} P(\text{open}|u) &= \sum P(x|u,x')P(x') \\ &= P(\text{open}|u,\text{open})P(\text{open}) + P(\text{open}|u,\text{closed})P(\text{closed}) \\ &= \frac{1}{10}\cdot\frac{1}{2} + 0\cdot\frac{1}{2} = \frac{1}{20} \\ &= 1 - P(\text{closed}|u) \end{aligned}$$

Modeling measurements

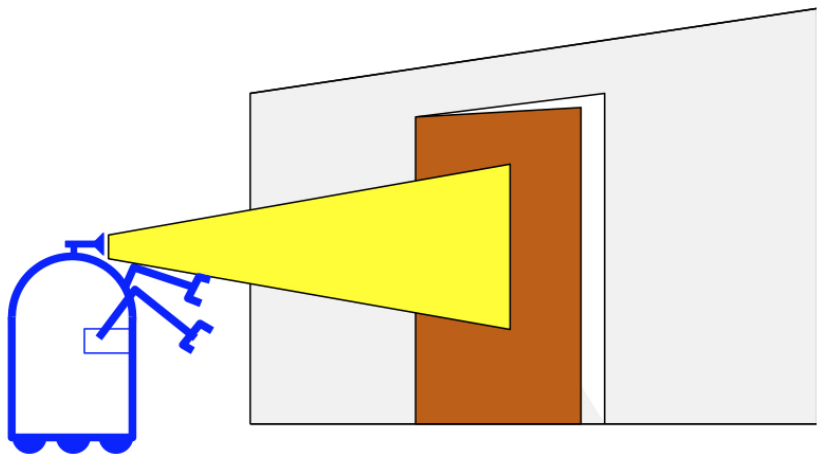
- to incorporate the outcome of a measurement or observation z into the current estimate (or belief), we use the **measurement pdf**

$$P(z|x)$$

- $P(z|x)$ specifies the probability to make measurement (observation) z in state x
- remember the localization problem:
  z = door
  x = position



Example: State estimation for measurement = state of door



Given:

1. discrete states: $x = \{\text{door open, door closed}\}$
2. robot makes a measurement z of state of door
   z could be a measurement of brightness
   measurement probability: $P(z|x)$
3. prior $P(x)$

Wanted:

- what is the state estimate or belief $P(x|z)$?
  (compare this to the state estimate or belief, $P(x|u)$, for action u)

Example: State estimation for measurement = state of door

- Bayes Law:

$$P(\text{open}|z) = \frac{P(z|\text{open})P(\text{open})}{P(z)}$$
$$= \frac{P(z|\text{open})P(\text{open})}{P(z|\text{open})P(\text{open}) + P(z|\text{closed})P(\text{closed})}$$

  and similar for $P(\text{closed}|z)$.

- What is the advantage of Bayes law?

  - measurement probability $P(z|\text{open})$ is easier to obtain - count frequencies:

  $z = \{\text{bright}\}$ $\quad P(z|\text{open}) = \lim_{N\to\infty} n_{\text{bright}}/N$

  N: total number of measurements

  - $P(z|\text{open})$ is **causal** and $P(\text{closed}|z)$ is **diagnostic**

- Bayes law allows to use causal knowledge

Example: State estimation for measurement = state of door

given:

measurement probability: $P(z|\text{open}) = 0.6$ and $P(z|\text{closed}) = 0.3$

prior: $P(\text{open}) = P(\text{closed}) = 0.5$

wanted: $P(x|z)$

$$\begin{aligned} P(\text{open}|z) &= \frac{P(z|\text{open})P(\text{open})}{P(z|\text{open})P(\text{open}) + P(z|\text{closed})P(\text{closed})} \\ &= \frac{0.6 \cdot 0.5}{0.6 \cdot 0.5 + 0.3 \cdot 0.5} = \frac{2}{3} \\ P(\text{closed}|z) &= 1 - P(\text{open}|z)) = \frac{1}{3} \end{aligned}$$

$\Rightarrow$ the measurement z raises the probability that the door is open (from 0.5 to 0.67)

Combining action and measurement

Wanted: state estimate or belief $P(x|z,u)$

- **Bayes Law**: $P(x|z) = \frac{P(z|x)P(x)}{P(z)}$
  condition Bayes Law on the additional variable $U = u$:

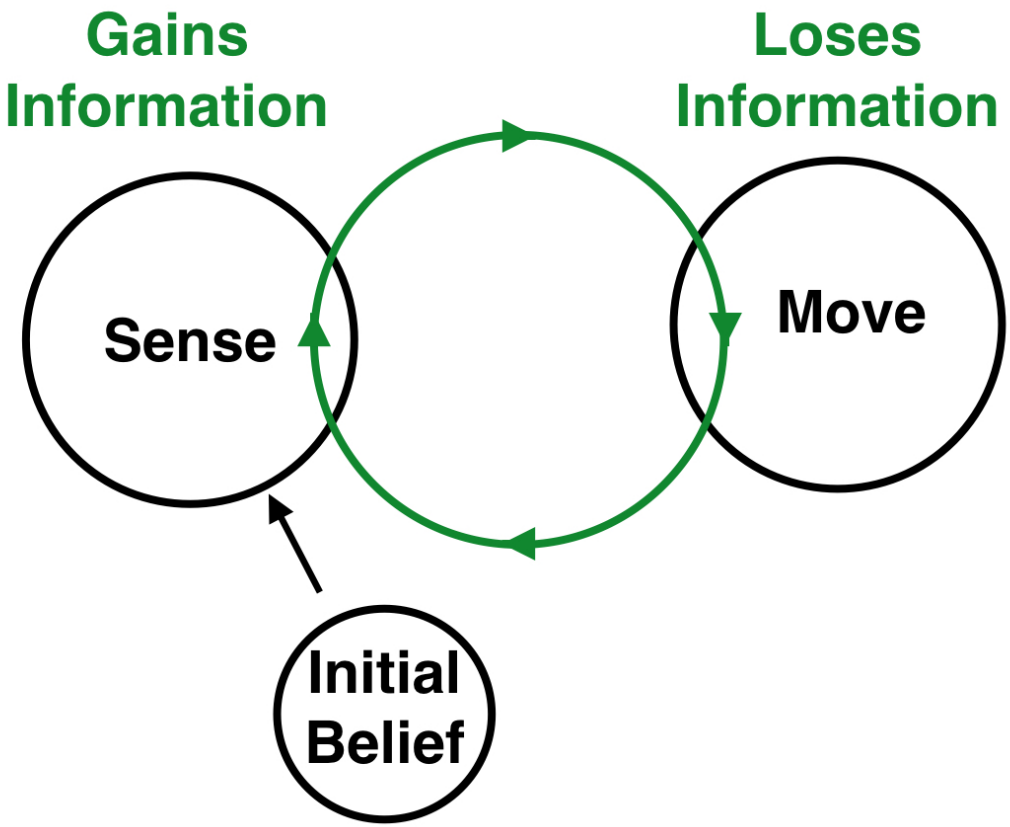
$$P(x|z,u) = \frac{P(z|x,u)P(x|u)}{P(z|u)}$$

- assume **Markov property**: $P(z|x,u) = P(z|x)$
  measurement probability depends only on present state x and not on action u that leads to state x

$$\underbrace{P(x|z,u)}_{\text{belief after measurement}} = \frac{P(z|x)P(x|u)}{P(z|u)} = \eta \cdot \underbrace{P(z|x)}_{\text{measurement prob.}} \cdot \underbrace{P(x|u)}_{\text{belief before measurement}},$$

$$\text{normalizer}: \quad \eta = \frac{1}{P(z|u)} = \frac{1}{\sum_x P(z|x)P(x|u)}$$

Generalization to a sequence of actions and a sequence of measurements



$$u_1, z_1, u_2, z_2, \ldots u_t, z_t$$

States, measurement and actions

- **measurements** x_t
  state at time t: x_t
  sequence of states from time t_1 to t_2, $(t_1 \leq t_2)$:

  $$x_{t_1:t_2} = x_{t_1}, x_{t_1+1}, x_{t_1+2}, \ldots, x_{t_2}$$

- **measurements** z_t
  measurement data at time t: z_t
  sequence of measurement data from time t_1 to t_2, $(t_1 \leq t_2)$:

  $$z_{t_1:t_2} = z_{t_1}, z_{t_1+1}, z_{t_1+2}, \ldots, z_{t_2}$$

- **actions** or **controls** u_t
  action at time t: u_t
  (change of state in the interval $(t-1, t]$)
  sequence of actions from time t_1 to t_2, $(t_1 \leq t_2)$:

  $$u_{t_1:t_2} = u_{t_1}, u_{t_1+1}, u_{t_1+2}, \ldots, u_{t_2}$$

State transition probability and measurement probability

1. move : **state transition probability**

   state x_t may be conditioned on all past states, measurements and controls, thus, the probability distribution has the form

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

   assume **Markov property**: the current state's probability distribution depends only on the previous state and present controls, i.e., x_t depends only on x_{t-1} and u_t.

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

2. sense: **measurement probability**

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

   assume **Markov property**: the current measurement probability distribution depends only on the present state, i.e., z_t depends only on x_t:

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

Belief distributions

- A **belief** reflects the robot's internal knowledge about the state x_t of the environment
- **belief distributions**: belief distributions are posterior probabilities over the state variables conditioned on the available data (measurements and controls)
- **definitions**: we denote the belief distributions as

  $$bel(x_t) = p(x_t|z_{1:t}, u_{1:t})$$

  $$\bar{bel}(x_t) = p(x_t|z_{1:t-1}, u_{1:t})$$

  The first is the belief *after* incorporating the measurement z_t, the second is the belief *before* incorporating the measurement z_t.

task: calculate the beliefs from the initial beliefs, state-transition probability and measurement probability. The most general algorithm for calculating beliefs is given by the **Bayes filter algorithm** ...

Bayes filter algorithm

1: **Algorithm Bayes_filter**($bel(x_{t-1}), u_t, z_t$):
2: *for all* x_t *do*
3: $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\ bel(x_{t-1})\ dx_{t-1}$
4: $bel(x_t) = \eta\ p(z_t \mid x_t)\ \overline{bel}(x_t)$
5: *endfor*
6: *return* $bel(x_t)$

- recursive algorithm: belief at time t is calculated from the belief at time $t-1$ and the most recent control u_t and measurement z_t
- step 3 is called **prediction** (motion update):

$$\bar{bel}(x_t) \quad = \quad \int p(x_t|u_t, x_{t-1})bel(x_{t-1})dx_{t-1}$$

- step 4 is called **measurement update** (correction):

$$bel(x_t) = \eta p(z_t|x_t)\bar{bel}(x_t) \quad (\eta : \text{ normalizer})$$

Bayes filter algorithm - proof

$$bel(x_t) = \eta\, p(z_t|x_t)\, \bar{bel}(x_t) = \eta p(z_t|x_t) \int p(x_t|u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}$$

Proof:

$$\begin{aligned}
bel(x_t) &\overset{:=}{} p(x_t|u_1, z_1, \ldots, u_t, z_t) \\
&\overset{\text{Bayes}}{=} \eta\, p(z_t|, x_t, u_1, z_1, \ldots, u_t)\, p(x_t|u_1, z_1, \ldots, u_t) \\
&\overset{\text{Markov}}{=} \eta\, p(z_t|, x_t)\, p(x_t|u_1, z_1, \ldots, u_t) \\
&\overset{\text{Tot. Prob.}}{=} \eta\, p(z_t|, x_t) \int p(x_t|u_1, z_1, \ldots, u_t, x_{t-1}) \\
&\qquad\qquad p(x_{t-1}|u_1, z_1, \ldots, u_t) dx_{t-1} \\
&\overset{\text{Markov}}{=} \eta\, p(z_t|, x_t) \int p(x_t|, u_t, x_{t-1}) p(x_{t-1}|u_1, z_1, \ldots, u_t) dx_{t-1} \\
&\overset{\text{Markov}}{=} \eta\, p(z_t|, x_t) \int p(x_t|, u_t, x_{t-1}) p(x_{t-1}|u_1, z_1, \ldots, z_{t-1}) dx_{t-1} \\
&\overset{\text{Markov}}{=} \eta\, p(z_t|, x_t) \int p(x_t|, u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}
\end{aligned}$$

Bayes filter algorithm

$bel(x_0)$

1: **Algorithm Bayes_filter**($bel(x_{t-1}), u_t, z_t$):
2: for all x_t do
3: $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$
4: $bel(x_t) = \eta\, p(z_t \mid x_t)\, \overline{bel}(x_t)$
5: endfor
6: return $bel(x_t)$

Given:

1. stream of observations z and actions u:
   data $=(u_1, z_1, \ldots, u_t, z_t)$
2. **prior = initial belief**, $bel(x_0)$
3. **action model = state transition probability:** $p(x_t|x_{t-1}, u_t)$
4. **sensor model = measurement probability:** $p(z_t|x_t)$

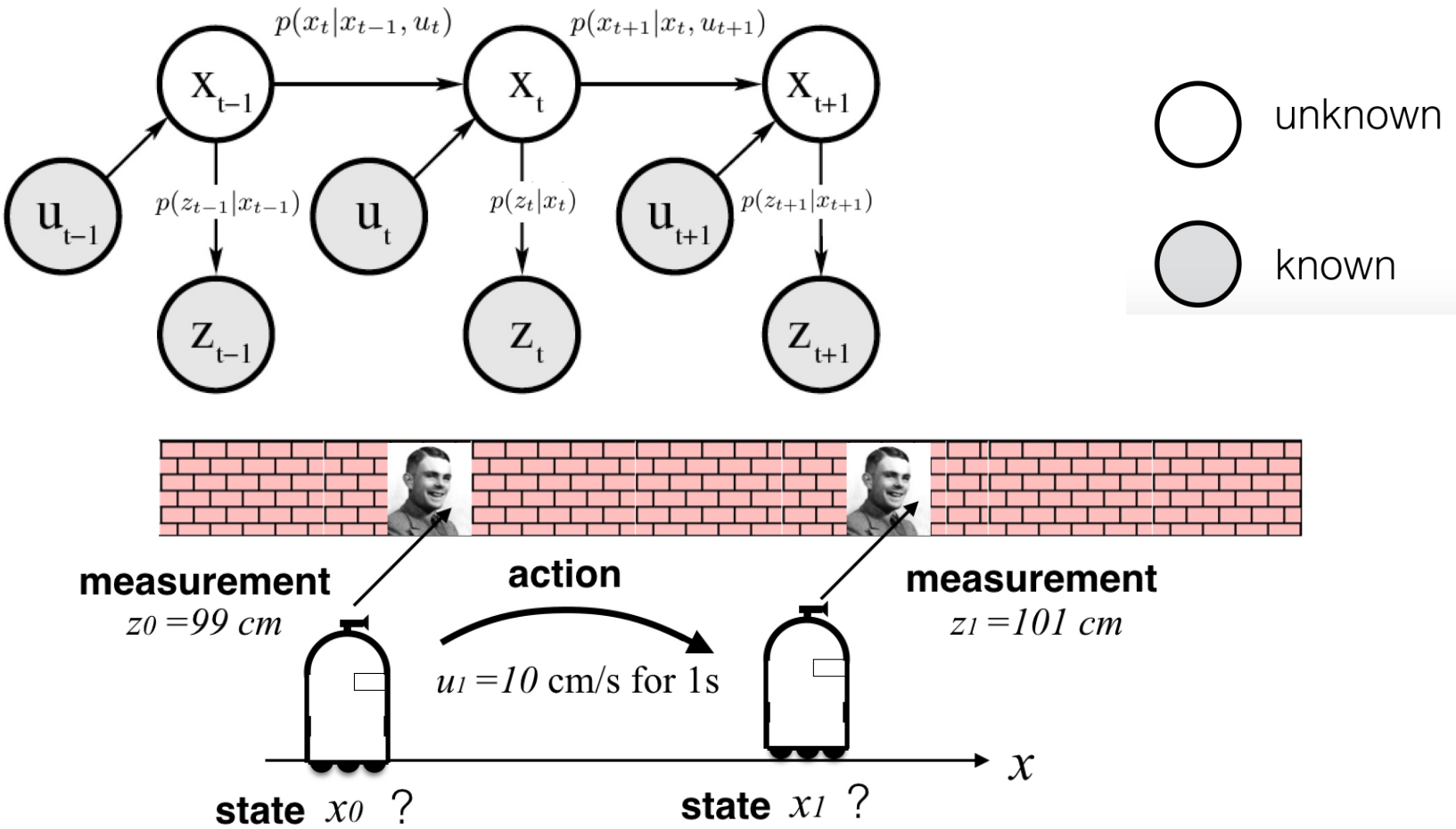
Wanted:

- estimate of the state x_t of a (probabilistic) dynamical system, i.e., the posterior of the state = belief

$$bel(x_t) = p(x_t|u_1, z_1, \ldots u_t, z_t) = p(x_t|data)$$

Evolution of controls, states and measurements

- The state transition probability, $p(x_t|x_{t-1}, u_t)$, and measurement probability, $p(z_t|x_t)$ define the dynamical stochastic system.
- The dynamical stochastic system can be graphically represented by a probabilistic graphical model:



Bayes filters are familiar

- Gaussian filter (Kalman filter)
- Particle filter
- Hidden Markov model (HMM)
- Dynamic Bayesian networks
- Partially Observable Markov Decision Processes (POMDPs)

Gaussian filters

- special Bayes filter for continuous spaces
- beliefs are represented by Gaussian (unimodal) distributions

special Gaussian filters:

- **Kalman filter (KF)**: filter for *Gaussian* systems with *linear dynamics* and *linear measurement functions*
- **Extended Kalman filter (EKF)** - extension of Kalman filter to *nonlinear* dynamics and *nonlinear* measurement functions
- **Unscented Kalman filter (UKF)** - a different *nonlinear* Kalman filter ("unscented" means "without smell")

The Kalman filter (KF)

The Kalman filter was developed by Peter Swerling (1958) and Rudolf E. Kalman (1960)

Peter Swerling
(1929-2000)

Rudolf Kalman
(1930-2016)

R. E. Kalman, *A new approach to linear filtering and prediction problems*, Journal of Fluids Engineering, vol. 82, no. 1, pp. 35 - 45,1960.

The Kalman filter - main uses

1. remove noise from data - filter
2. estimate of state (even for state variables that are **not** measured)
3. sensor fusion

The Kalman filter - applications

The applications of a Kalman filter in science are numerous

- Tracking objects (e.g., satellites, cars, missiles, faces, heads, hands)
- Navigation (all navigation tasks involve localization, i.e., determination of position and direction compared to known patterns e.g., ship with respect to the stars)
  land, marine, aeronautic and space navigation
- Computer vision (pattern recognition, image segmentation, feature and cluster tracking)
- Robotics: senor fusion
- Finance: prediction
- and many more

The Kalman filter is one of the most important discoveries in Engineering

Gaussians

- probability density function (pdf):

$$p(\boldsymbol{x}) = \det(2\pi\Sigma)^{-\frac{1}{2}} \exp\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\} \sim \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu},\Sigma)$$

with $\int p(\boldsymbol{x})d\boldsymbol{x} = 1$

- pdf is defined by the mean $\boldsymbol{\mu}$ and covariance matrix $\Sigma = (\sigma_{ij})$:

$$\boldsymbol{\mu} = E[\boldsymbol{x}] = \int \boldsymbol{x}p(\boldsymbol{x})d\boldsymbol{x}$$

$$\Sigma = E[(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^T], \text{ or in components}$$

$$\sigma_{ij} = E[(x_i-\mu_i)(x_j-\mu_j)] = \int (x_i-\mu_i)(x_j-\mu_j)p(\boldsymbol{x})d\boldsymbol{x}\,,\ i,j=1,2,\ldots n$$

$$\Sigma = (\sigma_{ij}) = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \ldots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & \ldots & \sigma_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \ldots & \sigma_{nn} \end{pmatrix},$$

σ_{ii}: variances of x_i

σ_{ij}: covariances of x_i and x_j, $i \neq j$

$\Sigma_{ij} = \Sigma_{ji}$

the covariance matrix has $\frac{1}{2}n(n+1)$ independent components

Gaussians

- **correlation matrix**:

  $\rho = (\rho_{ij}) = \frac{\Sigma_{ij}}{\sqrt{\sigma_{ii} \cdot \sigma_{jj}}}\,, \quad -1 \leq \rho_{ij} \leq 1\,, \quad \rho_{ij} = \rho_{ji}$

  advantage: components of the correlation matrix are dimensionless

  example: $X =$ length of fish [in cm], $Y =$ weight of fish [in kg]

  $\rho_{xy} = E[(\frac{x-\mu_x}{\sigma_x})(\frac{y-\mu_y}{\sigma_y})]$

  $\rho_{xy} > 0$ means if $(x - \mu_x) \lesseqgtr 0$ then $(y - \mu_y) \lesseqgtr 0$

- **precision matrix**: the inverse of the covariance matrix: Σ^{-1}

Gaussian in 1D

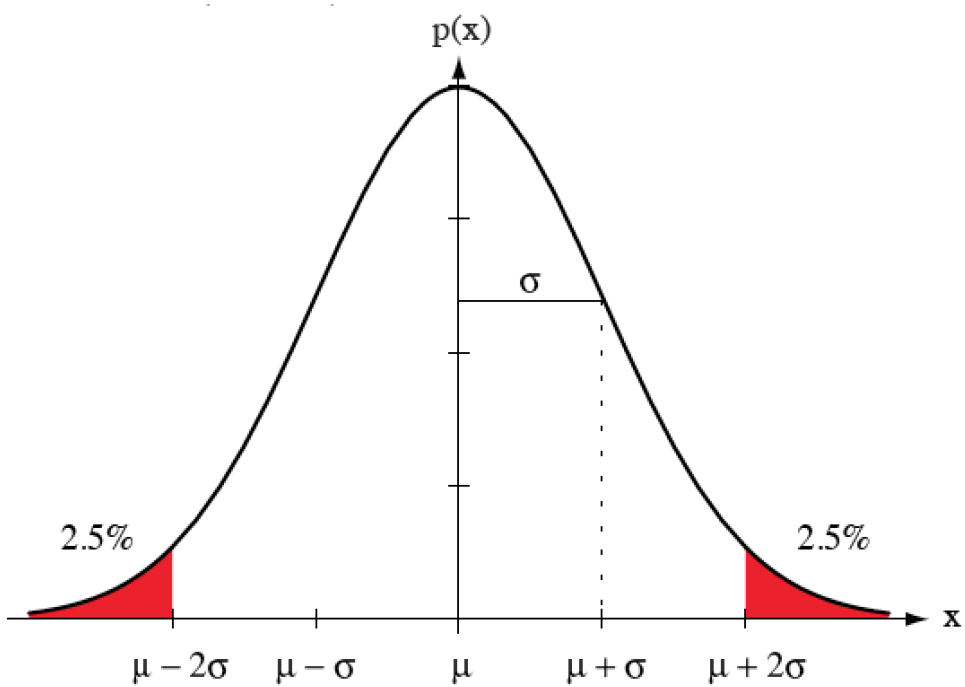


Figure 2.7: A univariate normal distribution has roughly 95% of its area in the range $|x - \mu| \leq 2\sigma$, as shown. The peak of the distribution has value $p(\mu) = 1/\sqrt{2\pi}\sigma$.

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

Note: the higher the peak, the smaller the standard deviation ($p(\mu) \sim \frac{1}{\sigma}$)

Gaussian in 2D

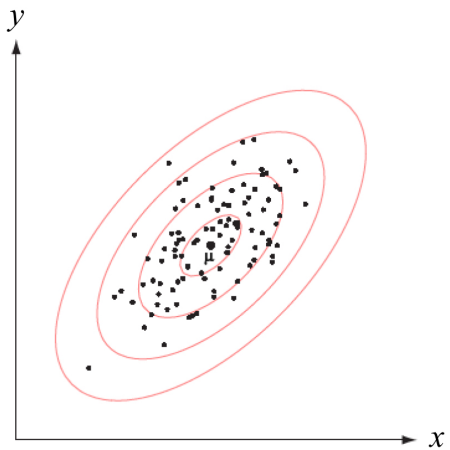


Figure 2.9: Samples drawn from a two-dimensional Gaussian lie in a cloud centered on the mean μ. The red ellipses show lines of equal probability density of the Gaussian.

$$\Sigma = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{yx} & \sigma_y^2 \end{pmatrix} = \begin{pmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{pmatrix}$$

$$\Sigma^{-1} = \frac{1}{\sigma_x^2\sigma_y^2(1-\rho^2)} \begin{pmatrix} \sigma_y^2 & -\rho\sigma_x\sigma_y \\ -\rho\sigma_x\sigma_y & \sigma_x^2 \end{pmatrix}$$

$\rho \in [-1, 1]$: correlation coefficient

$$p(x,y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \times \exp\{-\frac{1}{2(1-\rho^2)}\left[\left(\frac{x-\mu_x}{\sigma_x}\right)^2 + \left(\frac{y-\mu_y}{\sigma_y}\right)^2 - 2\rho\left(\frac{x-\mu_x}{\sigma_x}\right)\left(\frac{y-\mu_y}{\sigma_y}\right)\right]\}$$

Properties of Gaussians: Marginalization and Conditioning

- given Gaussian $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \Sigma)$, where

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{x}_a \\ \boldsymbol{x}_b \end{pmatrix}, \mu = \begin{pmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{pmatrix}, \Sigma = \begin{pmatrix} \Sigma_a & \Sigma_c \\ \Sigma_c^T & \Sigma_b \end{pmatrix}$$

- the marginals are Gaussians

$$p(\boldsymbol{x}_a) = \mathcal{N}(\boldsymbol{x}_a|\boldsymbol{\mu}_a, \Sigma_a), \quad p(\boldsymbol{x}_b) = \mathcal{N}(\boldsymbol{x}_b|\boldsymbol{\mu}_b, \Sigma_b)$$

- the conditionals are Gaussians

$$\begin{aligned} p(\boldsymbol{x}_a|\boldsymbol{x}_b) &= \mathcal{N}(\boldsymbol{x}_a|\hat{\boldsymbol{\mu}}_a, \hat{\Sigma}_a) \\ \hat{\boldsymbol{\mu}}_a &= \boldsymbol{\mu}_a + \Sigma_c \Sigma_b^{-1}(\boldsymbol{x}_b - \boldsymbol{\mu}_b) \\ \hat{\Sigma}_a &= \Sigma_a - \Sigma_c \Sigma_b^{-1} \Sigma_c^T \end{aligned}$$

$$\begin{aligned} p(\boldsymbol{x}_b|\boldsymbol{x}_a) &= \mathcal{N}(\boldsymbol{x}_b|\hat{\boldsymbol{\mu}}_b, \hat{\Sigma}_b) \\ \hat{\boldsymbol{\mu}}_b &= \boldsymbol{\mu}_b + \Sigma_c \Sigma_a^{-1}(\boldsymbol{x}_a - \boldsymbol{\mu}_a) \\ \hat{\Sigma}_b &= \Sigma_b - \Sigma_c \Sigma_a^{-1} \Sigma_c^T \end{aligned}$$

The Kalman filter - transition and measurement function

- The Kalman filter assumes **linear** transition and measurement models
- zero mean **Gaussian noise**
- initial **Gaussian belief**

1. state transition function

$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t \quad \text{with} \quad \epsilon_t \sim \mathcal{N}(0, R_t)$$

2. measurement function:

$$z_t = C_t x_t + \delta_t \quad \text{with} \quad \delta_t \sim \mathcal{N}(0, Q_t)$$

3. initial belief must be normally distributed

$$bel(x_0) \sim \mathcal{N}(\mu_0, \Sigma_0)$$

The Kalman filter - components of the KF filter

- A_t - state transition matrix:
  Matrix $(n \times n)$ that describes how the state evolves from $t-1$ to t without controls or noise.

- B_t - control input matrix:
  Matrix $(n \times m)$ that describes how the control u_t changes the state from $t-1$ to t.

- R_t - covariance matrix of process noise:
  Matrix $(n \times n)$ that describes the noise of the motion.

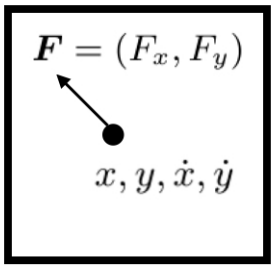
- C_t - measurement matrix:
  Matrix $(k \times n)$ that describes how to map the state x_t to an observation z_t.

- Q_t - covariance matrix of measurement noise:
  Matrix $(k \times k)$ that describes the noise of the measurement

- ϵ_t, δ_t - Random variables representing the process and measurement noise that are assumed to be independent and normally distributed with covariance R_t and Q_t, respectively.

Example: linear transition model

- state: $\boldsymbol{x}_t = (x_t, y_x, \dot{x}_t, \dot{y}_t)^T$
- from physics: Newton's equation: $\boldsymbol{F} = m\boldsymbol{a} = m\ddot{\boldsymbol{x}}$ (ignore bouncing on wall)



$$
\begin{aligned}
x_{t+1} &= x_t + \dot{x}_t \Delta t + \frac{1}{2}\frac{F_{x,t}}{m}\Delta t^2 \\
y_{t+1} &= y_t + \dot{y}_t \Delta t + \frac{1}{2}\frac{F_{y,t}}{m}\Delta t^2 \\
\dot{x}_{t+1} &= \dot{x}_t + \frac{F_{x,t}}{m}\Delta t \\
\dot{y}_{t+1} &= \dot{y}_t + \frac{F_{y,t}}{m}\Delta t\,,
\end{aligned}
$$

which has the form $\boldsymbol{x}_{t+1} = A\boldsymbol{x}_t + B\boldsymbol{u}_t$:

$$
\underbrace{\begin{pmatrix} x_{t+1} \\ y_{t+1} \\ \dot{x}_{t+1} \\ \dot{y}_{t+1} \end{pmatrix}}_{\boldsymbol{x}_{t+1}} = \underbrace{\begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{A_t} \cdot \underbrace{\begin{pmatrix} x_t \\ y_t \\ \dot{x}_t \\ \dot{y}_t \end{pmatrix}}_{\boldsymbol{x}_t} + \underbrace{\begin{pmatrix} \frac{1}{2}\frac{\Delta t^2}{m} & 0 \\ 0 & \frac{1}{2}\frac{\Delta t^2}{m} \\ \frac{\Delta t}{m} & 0 \\ 0 & \frac{\Delta t}{m} \end{pmatrix}}_{B_t} \cdot \underbrace{\begin{pmatrix} F_{x,t} \\ F_{y,t} \end{pmatrix}}_{\boldsymbol{u}_t}
$$

Example: linear measurement model

let us assume we can measure the position (x, y) of the object i.e., we have a measurement vector: $z_t = (x_t, y_t)^T$. Then

$$\underbrace{\begin{pmatrix} x_t \\ y_t \end{pmatrix}}_{\boldsymbol{z}_t} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}}_{C_t} \cdot \underbrace{\begin{pmatrix} x_t \\ y_t \\ \dot{x}_t \\ \dot{y}_t \end{pmatrix}}_{\boldsymbol{x}_t}$$

i.e., we obtain a linear measurement model

$$\boldsymbol{z}_t = C_t \boldsymbol{x}_t$$

Example: noise

In the previous example we have derived **deterministic** equations.
By adding noise vectors we obtain **stochastic** equations:

- transition model

$$\boldsymbol{x}_{t+1} = A\boldsymbol{x}_t + B\boldsymbol{u}_t + \boldsymbol{\epsilon}_t \quad \text{with} \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(0, R_t)$$

- measurement model

$$\boldsymbol{z}_t = C_t\boldsymbol{x}_t + \boldsymbol{\delta}_t \quad \text{with} \quad \boldsymbol{\delta}_t \sim \mathcal{N}(0, Q_t)$$

The amount of noise that we add must come from an understanding of the dynamic and measurement process:
example: assume our position sensor has a precision in x and y of $\sigma_x = \pm 0.5$m and $\sigma_y = \pm 0.8$, respectively, then we set

$$Q_t = \begin{pmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{pmatrix} = \begin{pmatrix} 0.25 & 0 \\ 0 & 0.64 \end{pmatrix} \ [\text{m}^2]$$

estimates of unknown forces and disturbances (in our example, rough surface, friction) in the dynamic process determine

$$R_t = \begin{pmatrix} r_x^2 & 0 & 0 & 0 \\ 0 & r_y^2 & 0 & 0 \\ 0 & 0 & r_{\dot{x}^2} & 0 \\ 0 & 0 & 0 & r_{\dot{y}^2} \end{pmatrix}$$

The Kalman filter (KF) - probability distributions

1. **state transition probability density**

   motion under Gaussian noise leads to

$$p(x_t|x_{t-1},u_t) = \det(2\pi R_t)^{-\frac{1}{2}} \times \exp\{-\frac{1}{2}(x_t - \underbrace{A_t x_{t-1} - B_t u_t}_{\text{mean}})^T R_t^{-1}(x_t - \underbrace{A_t x_{t-1} - B_t u_t}_{\text{mean}})\}$$

   covariance $R_t = E[\epsilon_t \cdot \epsilon_t^T]$ describes noise of motion

2. **measurement probability density**

   measurement under Gaussian noise leads to

$$p(z_t|x_t) = \det(2\pi Q_t)^{-\frac{1}{2}} \exp\{-\frac{1}{2}(z_t - \underbrace{C_t x_t}_{\text{mean}})^T Q_t^{-1}(z_t - \underbrace{C_t x_t}_{\text{mean}})\}$$

   covariance $Q_t = E[\delta_t \cdot \delta_t^T]$ describes noise of measurement

3. **initial belief**

$$bel(x_0) = \det(2\pi\Sigma_0)^{-\frac{1}{2}} \exp\{-\frac{1}{2}(x_0 - \mu_0)^T \Sigma_0^{-1}(x_0 - \mu_0)\}\,.$$

The Kalman filter - everything stays Gaussian

- using the Bayes filter algorithm we can compute the beliefs $\bar{bel}(x_t)$ and $bel(x_t)$ for every time step t
- given an initial Gaussian belief, the belief is always Gaussian - **everything stays Gaussian**

$$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\ bel(x_{t-1})\ dx_{t-1}$$

$$bel(x_t) = \eta\, p(z_t \mid x_t)\ \overline{bel}(x_t)$$

Proof: non-trivial (see Probabilistic Robotics, Chapter 3)

- Gaussian distributions are described by means and covariance matrices for every time step t:

$$\bar{bel}(x_t) \iff \bar{\mu}_t, \bar{\Sigma}_t$$
$$bel(x_t) \iff \mu_t, \Sigma_t$$

which are computed by a Kalman filter ...

The Kalman filter - algorithm

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

2: $\bar{\mu}_t = A_t\ \mu_{t-1} + B_t\ u_t$ } prediction (move)

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

4: $K_t = \bar{\Sigma}_t\ C_t^T (C_t\ \bar{\Sigma}_t\ C_t^T + Q_t)^{-1}$ Kalman gain

5: $\mu_t = \bar{\mu}_t + K_t(z_t - C_t\ \bar{\mu}_t)$ } measurement update (sense)

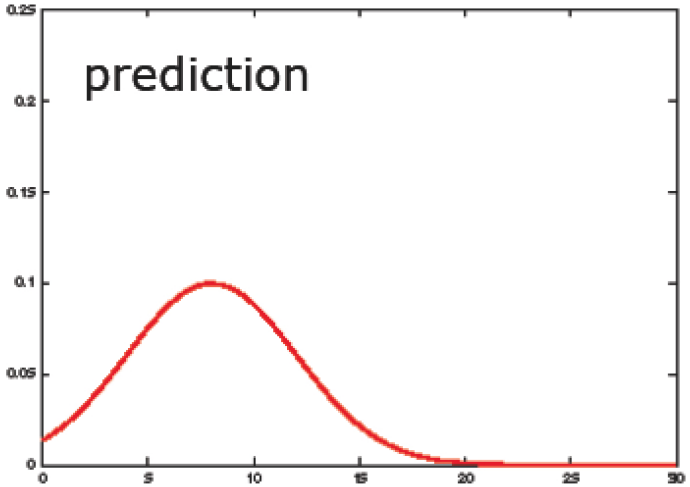
6: $\Sigma_t = (I - K_t\ C_t)\ \bar{\Sigma}_t$ }

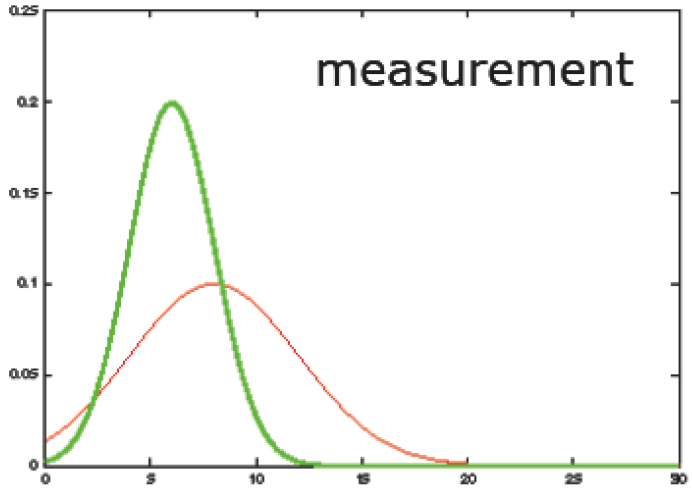
7: return μ_t, Σ_t

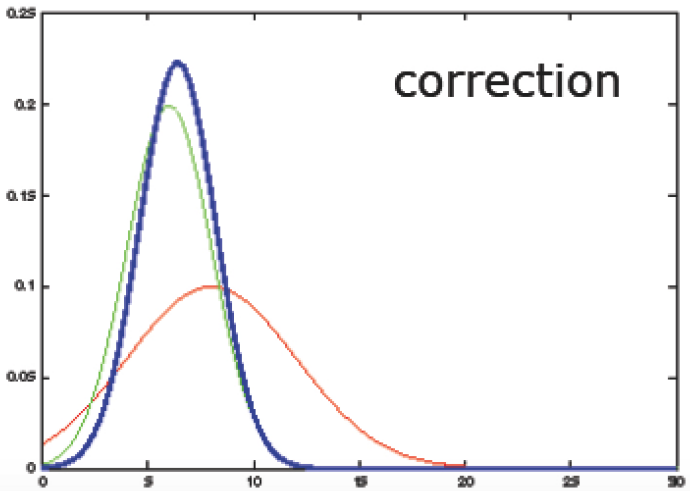
note:

- if measurement noise Q_t is large then **Kalman gain** K_t is small, leading to $\mu_t \approx \bar{\mu}_t$ and $\Sigma_t \approx \bar{\Sigma}_t$, i.e., the less informative is the measurement. The Kalman gain is a measure of information.
- $z_t - C_t\bar{\mu}_t$ is the difference between the actual measurement, z_t, and the measurement prediction, $\bar{z}_t = C_t\bar{\mu}_t$, according to the measurement model. $z_t - \bar{z}_t$ is called **innovation**.

The Kalman filter - 1D example

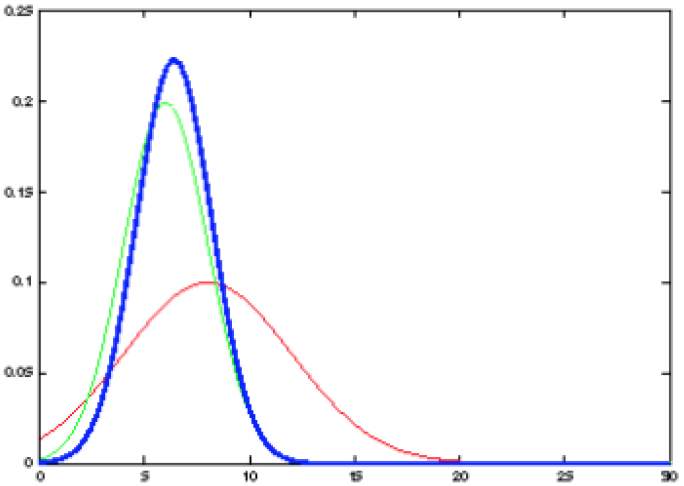


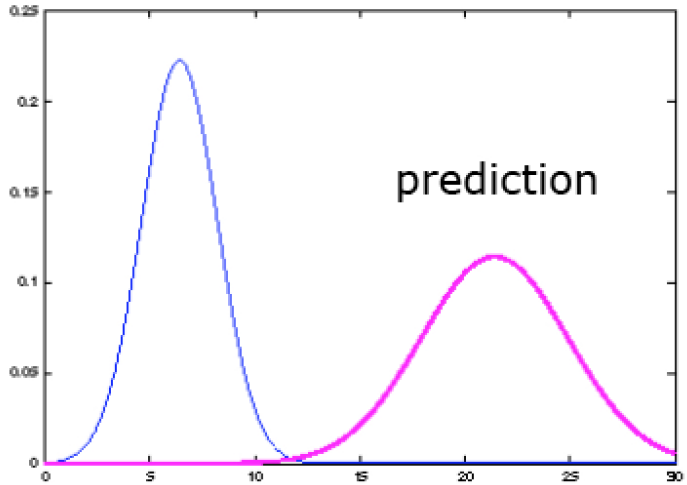


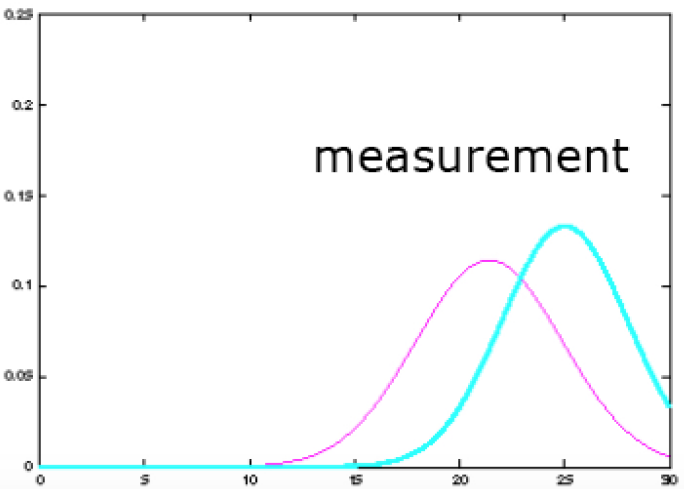


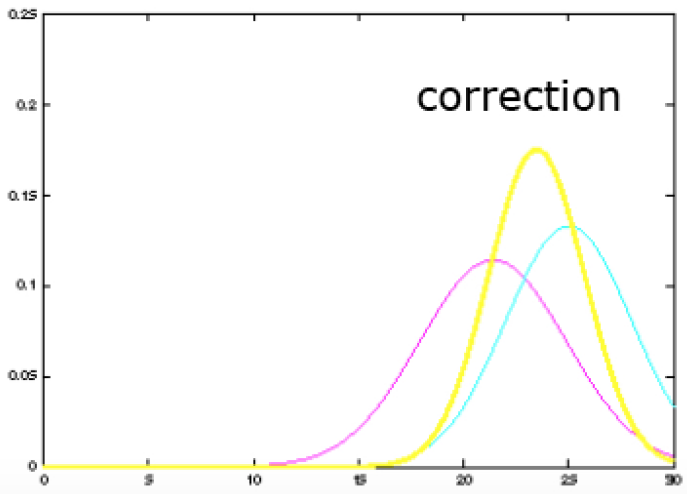
It's a weighted mean!

The Kalman filter - 1D example



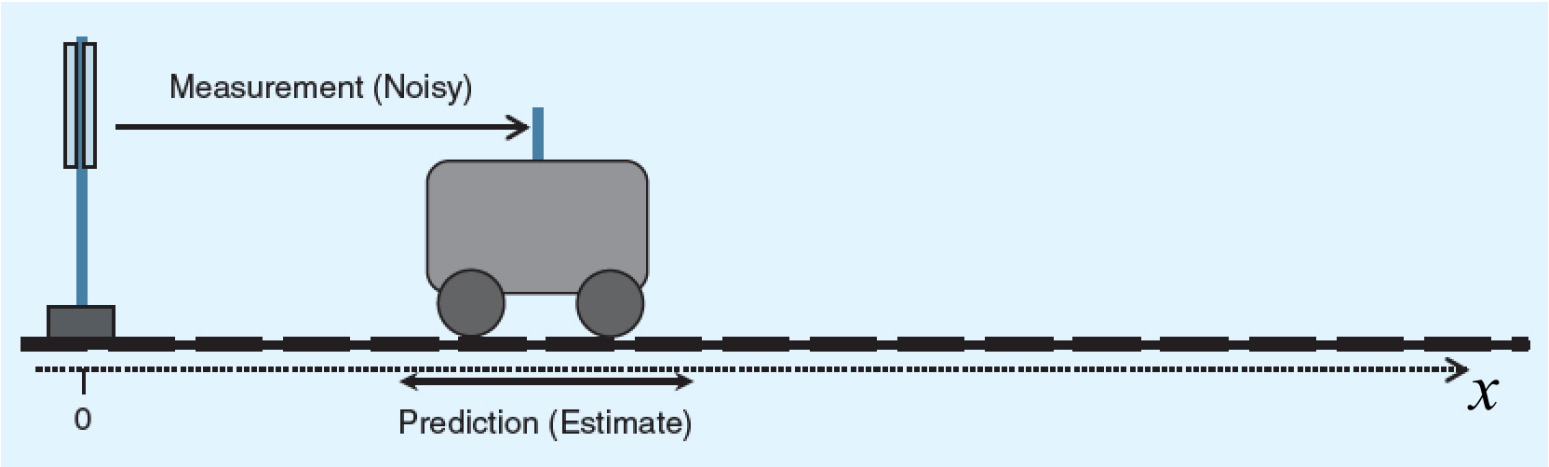






The Kalman filter - 1D example, expanded

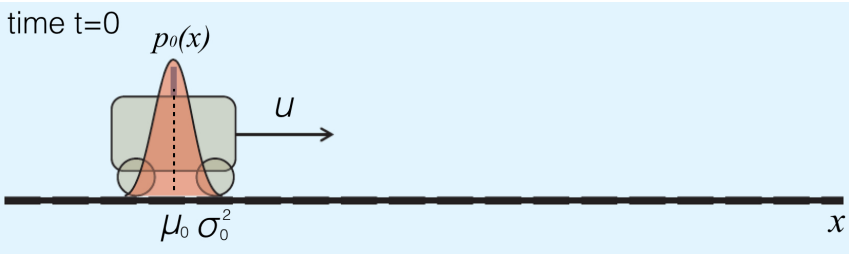
Tracking of a car (e.g., self-driving car that tracks other cars)

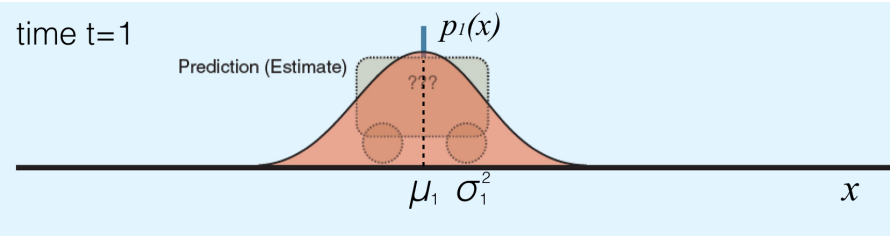


initial belief:

The Kalman filter - 1D example, expanded

Prediction step: $\bar{bel}(x_t) = \int p(x_t|u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}$





$$p_0(x) = \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left[-\frac{(x-\mu_0)^2}{2\sigma_0^2}\right] \equiv bel(x)\,,$$

$$p(x'|u,x) = \frac{1}{\sqrt{2\pi r^2}} \exp\left[-\frac{(x'-(x+u\Delta t))^2}{2r^2}\right],$$

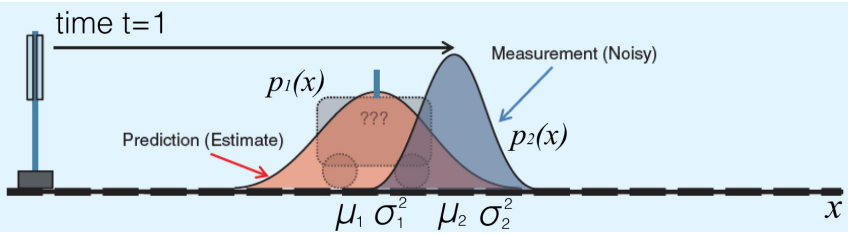
$$p_1(x') = \int_{-\infty}^{\infty} p(x'|u,x) \cdot p_0(x) dx = \ldots = \frac{1}{\sqrt{2\pi(\sigma_0^2+r^2)}} \exp\left[-\frac{(x'-(\mu_0+u\Delta t))^2}{2(\sigma_0^2+r^2)}\right]$$

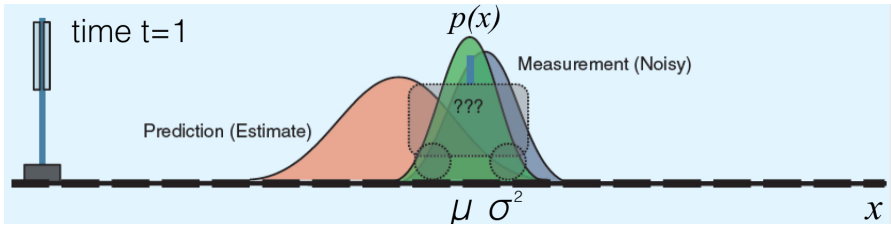
$\Rightarrow$ *mean*: $\mu_1 = \mu_0 + u\Delta t$
$\Rightarrow$ *variance*: $\sigma_1^2 = \sigma_0^2 + r^2 \geq \sigma_0^2$

Conclusion: prediction step (move) loses information

The Kalman filter - 1D example, expanded

Measurement update: $bel(x_t) = \eta p(z_t|x_t)\bar{bel}(x_t)$





$$p_1(x) = \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left[-\frac{(x-\mu_1)^2}{2\sigma_1^2}\right] \equiv \bar{bel}(x)$$

$$p_2(x) = \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left[-\frac{(x-\mu_2)^2}{2\sigma_2^2}\right] \equiv p(z|x) \text{ with } z = \mu_2$$

$$p(x) = p_1(x) \cdot p_2(x) \Rightarrow \text{exponent: } \alpha(x) = -\left[\frac{(x-\mu_1)^2}{2\sigma_1^2} + \frac{(x-\mu_2)^2}{2\sigma_2^2}\right]$$

mean: $\alpha'(x)_{|x=\mu} = 0 \Rightarrow \mu = \frac{\sigma_2^2\mu_1 + \sigma_1^2\mu_2}{\sigma_1^2+\sigma_2^2} = \mu_1 + \frac{\sigma_1^2(\mu_2-\mu_1)}{\sigma_1^2+\sigma_2^2}, \; (\mu_1 \leq \mu \leq \mu_2)$

variance: $1/\sigma^2 = -\alpha''(x) \Rightarrow \sigma^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}, \quad (< \sigma_1^2, \sigma_2^2)$

Conclusion: measurement update (sense) gains information

The Kalman filter - 1D example, expanded

Results of the KF-algorithm:

intial belief :

$p(x_0) \sim \mathcal{N}(0, \sigma_0^2)$

dynamics :

$$x_t = x_{t-1} + u\Delta t + \epsilon_t\,, \quad \epsilon_t \sim N(0, r^2)$$

measurement process :

$$z_t = x_t + \delta_t\,, \quad \delta_t \sim N(0, \sigma_2^2)$$

$$\Rightarrow A_t = 1, B_t = \Delta t, u_t = u, R_t = r^2$$
$$\Rightarrow C_t = 1, z_t = \mu_2, Q_t = \sigma_2^2$$

1. Prediction: $\mu_1 = \mu_0 + u\Delta t,$ $\sigma_1^2 = \sigma_0^2 + r^2$
2. Kalman gain: $K = \sigma_1^2/(\sigma_1^2 + \sigma_2^2)$
3. Measurement update: $\mu_2 = \mu_1 + \frac{\sigma_1^2}{(\sigma_1^2+\sigma_2^2)}(\mu_2 - \mu_1)$ $\sigma_2 = (1 - \frac{\sigma_1^2}{(\sigma_1^2+\sigma_2^2)})\sigma_1^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}$

i.e., the results that we obtained by multiplication of Gaussians

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

$\bar{\mu}_t = A_t\,\mu_{t-1} + B_t\,u_t$
$\bar{\Sigma}_t = A_t\,\Sigma_{t-1}\,A_t^T + R_t$ } prediction (move)

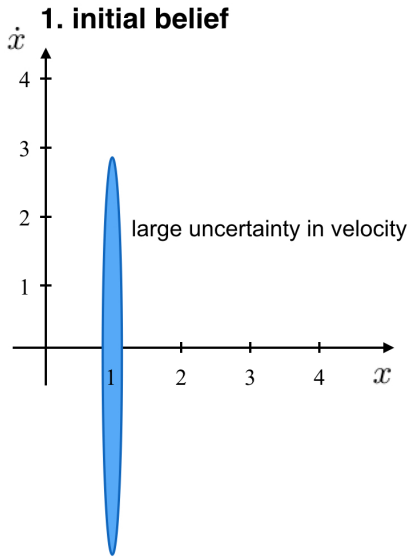
$K_t = \bar{\Sigma}_t\,C_t^T(C_t\,\bar{\Sigma}_t\,C_t^T + Q_t)^{-1}$ Kalman gain

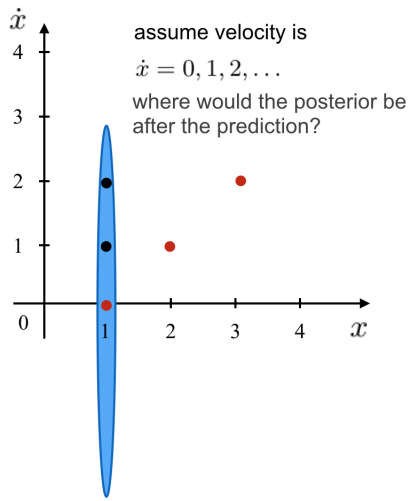
$\mu_t = \bar{\mu}_t + K_t(z_t - C_t\,\bar{\mu}_t)$
$\Sigma_t = (I - K_t\,C_t)\,\bar{\Sigma}_t$ } measurement update (sense)

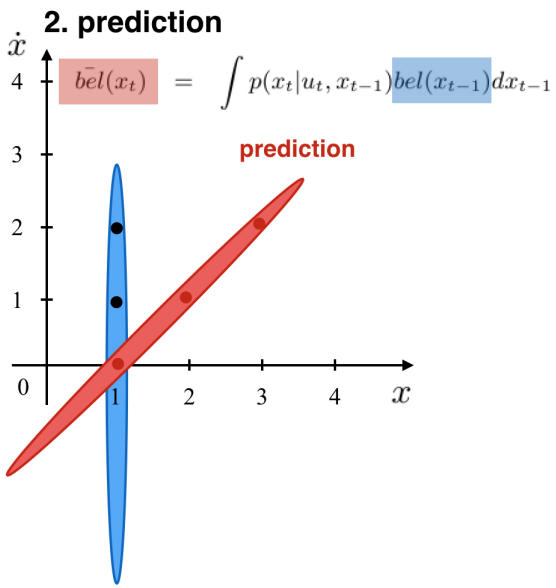
return μ_t, Σ_t

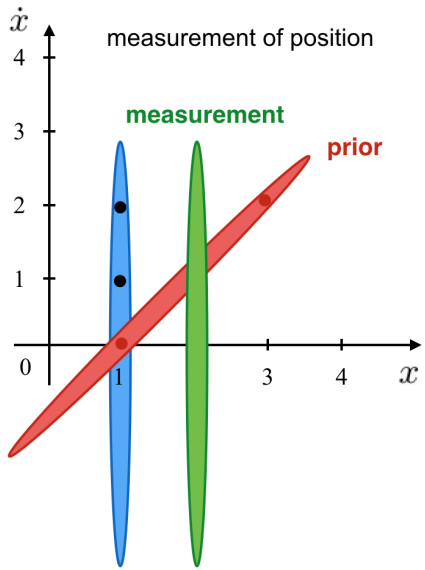
The Kalman filter - higher dimensions

consider a 2D systems with state vector: $\boldsymbol{x} = (x, \dot{x})$

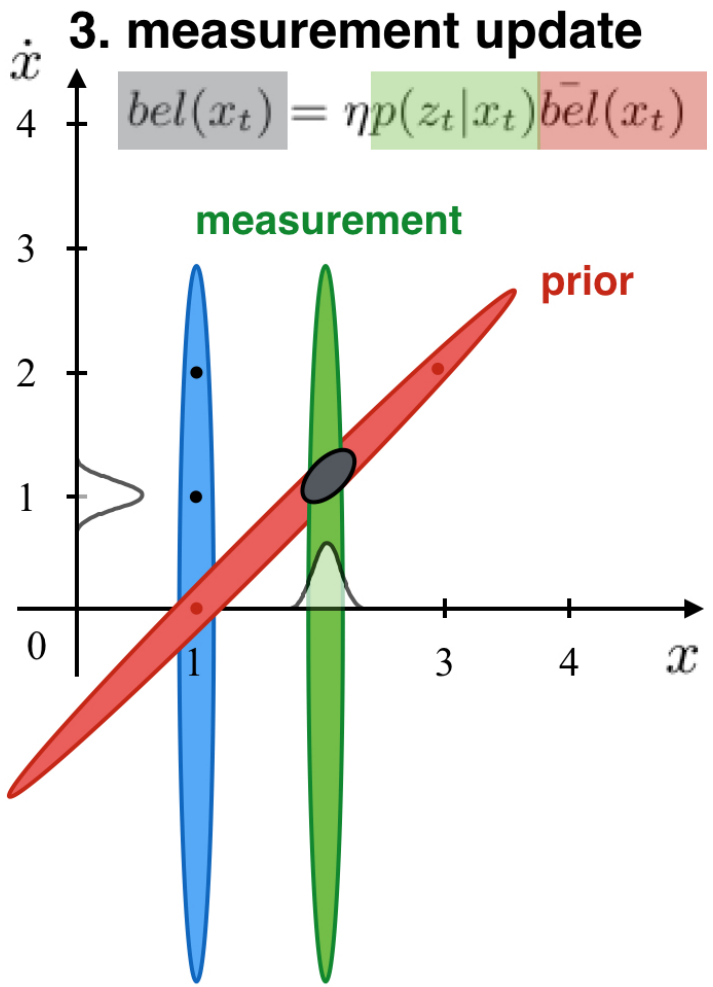






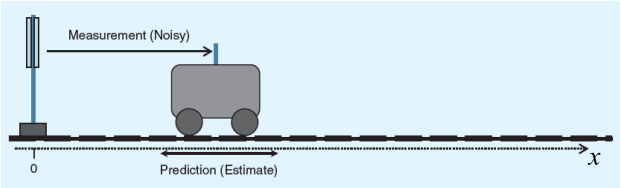


The Kalman filter - higher dimensions



- observation of the position provides information of the variable velocity that is **not** measured
- position and velocity are now estimated with high certainty

The Kalman filter - example



A car is moving with constant acceleration a perturbed by noise with zero mean and variance σ_a^2 along a straight road. Noisy measurements (with variance σ^2) of the position x are taken. Estimate the location and velocity of the car.

define state $\boldsymbol{x}_t = \begin{pmatrix} x_t \\ \dot{x}_t \end{pmatrix}$

1. state transition function (from physics)

$$\begin{aligned} x_t &= x_{t-1} + \dot{x}_{t-1}\Delta t + \frac{1}{2}a_t\Delta t^2 \\ \dot{x}_t &= \dot{x}_{t-1} + a_t\Delta t \quad \text{with} \quad a_t = N(a, \sigma_a^2) \end{aligned}$$

$$\boldsymbol{x}_t = A_t\boldsymbol{x}_{t-1} + B_t u_t + \boldsymbol{\epsilon}_t$$

where $A_t = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix}, B_t = \begin{pmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{pmatrix}, u_t = a, \boldsymbol{\epsilon_t} = \begin{pmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{pmatrix} \sigma_a \cdot \eta$

with $\eta = N(0, 1)$

The Kalman filter - example

covariance matrix of process noise:

$$R_t = E[\boldsymbol{\epsilon_t} \cdot \boldsymbol{\epsilon_t}^T] = \begin{pmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 \end{pmatrix} E[\sigma_a^2 \eta^2] = \begin{pmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 \end{pmatrix} \sigma_a^2 E[\eta^2]$$

$$= \begin{pmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 \end{pmatrix} \sigma_a^2$$

2. measurement function:

$$\boldsymbol{z}_t = C_t \boldsymbol{x}_t + \boldsymbol{\delta_t} \quad \text{with} \quad \boldsymbol{\delta_t} \sim \mathcal{N}(0, Q_t)$$

only position is measured, thus, $\boldsymbol{z}_t = x_t$, and

$$C_t = \begin{pmatrix} 1 & 0 \end{pmatrix}$$

covariance matrix of measurement noise: $Q_t = \sigma^2$

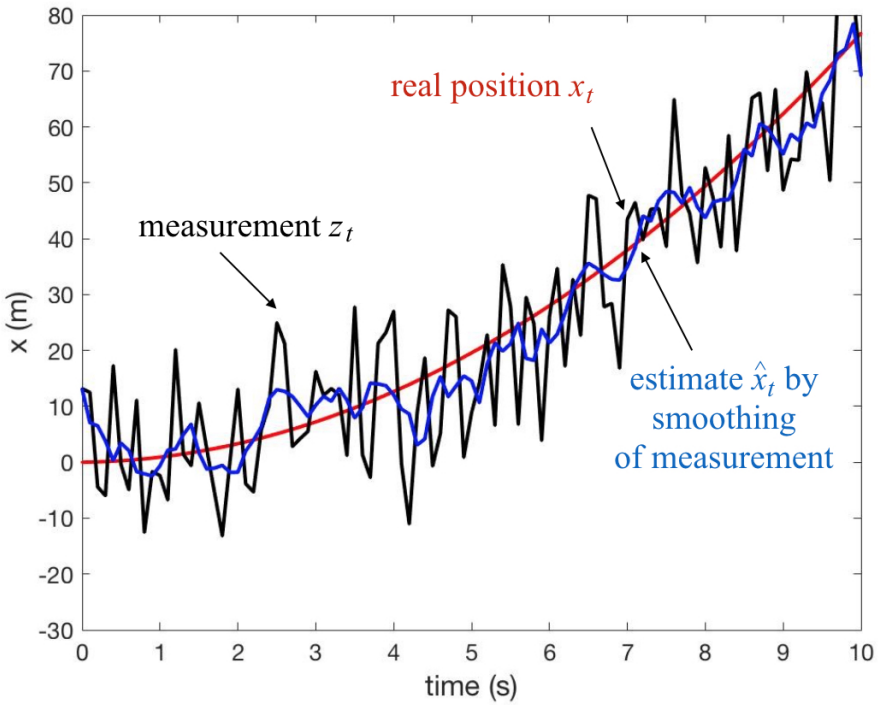
3. deterministic initial state $\boldsymbol{x}_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ and variance $\Sigma_0 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$

The Kalman filter - simulation results of example: position

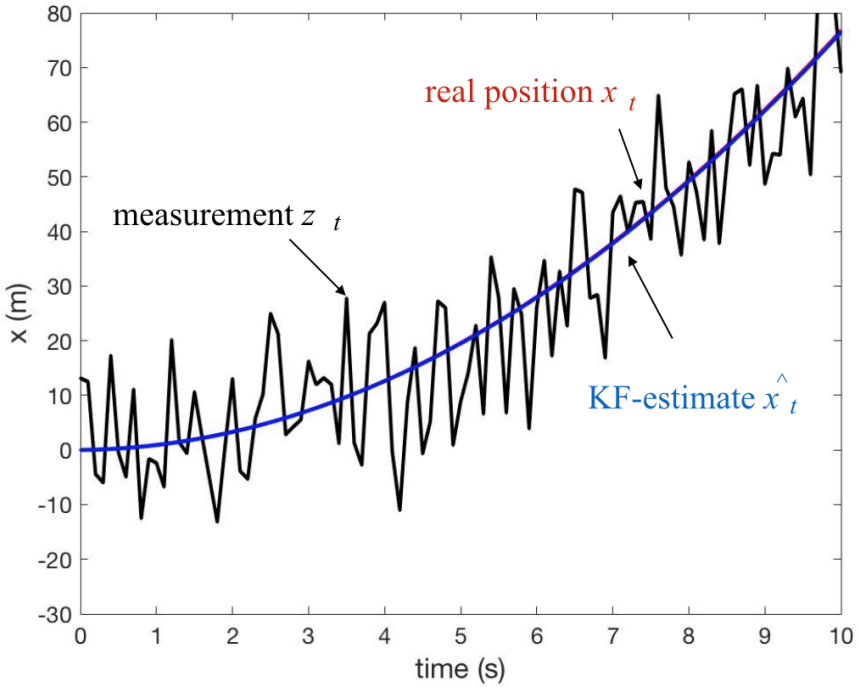
Simulation parameters:

total simulation time: $T = 10$s
time step: $\Delta t = 0.1$s
acceleration: $a = 1.5\text{m/s}^2$
noise (SD): $\sigma_a = 0.05\text{m/s}^2$, $\sigma = 10$m

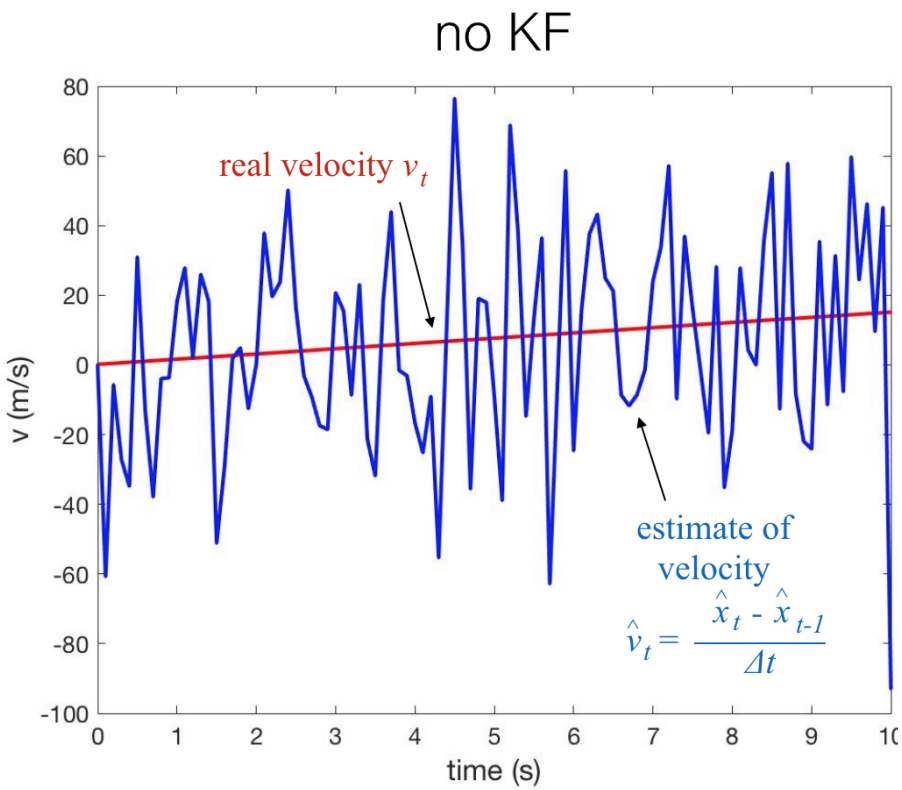
no KF

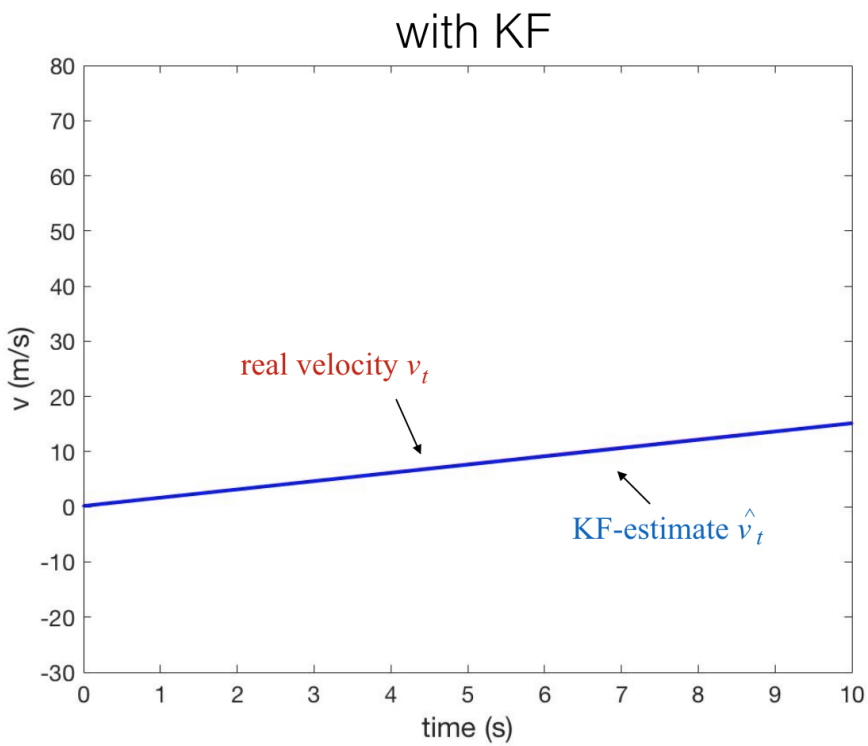


with KF



The Kalman filter - simulation results of example: velocity





Kalman filter assumptions

1. Gaussian distributions and noise
2. Linear motion and measurement model

$$\begin{aligned} x_t &= A_t x_{t-1} + B_t u_t + \epsilon_t \\ z_t &= C_t x_t + \delta_t \end{aligned}$$

What if this is not the case?

Non-linear dynamic systems

- Most realistic problems in robotics involve nonlinear functions

$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t \qquad z_t = C_t x_t + \delta_t$$

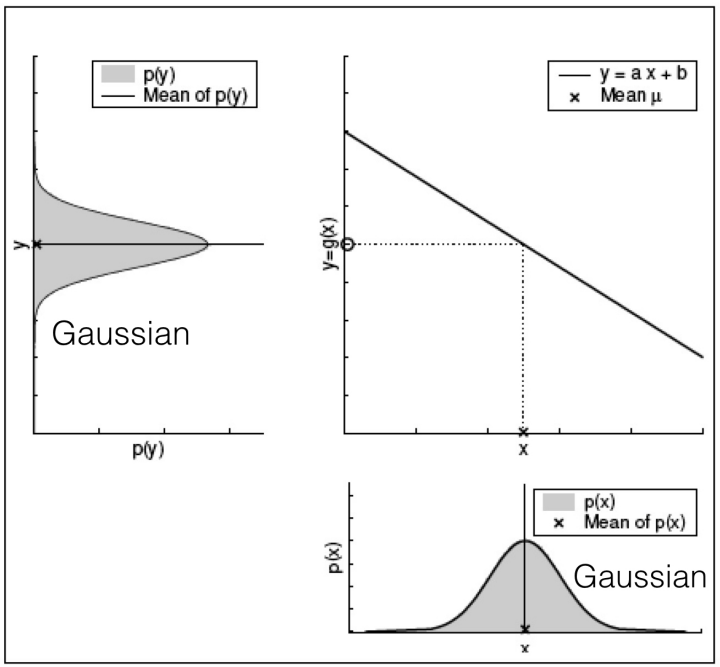
$$x_t = g(u_t, x_{t-1}) + \epsilon_t \qquad z_t = h(x_t) + \delta_t$$

What effect do these nonlinear functions have?

Linear and nonlinear transformation of a Gaussian random variable

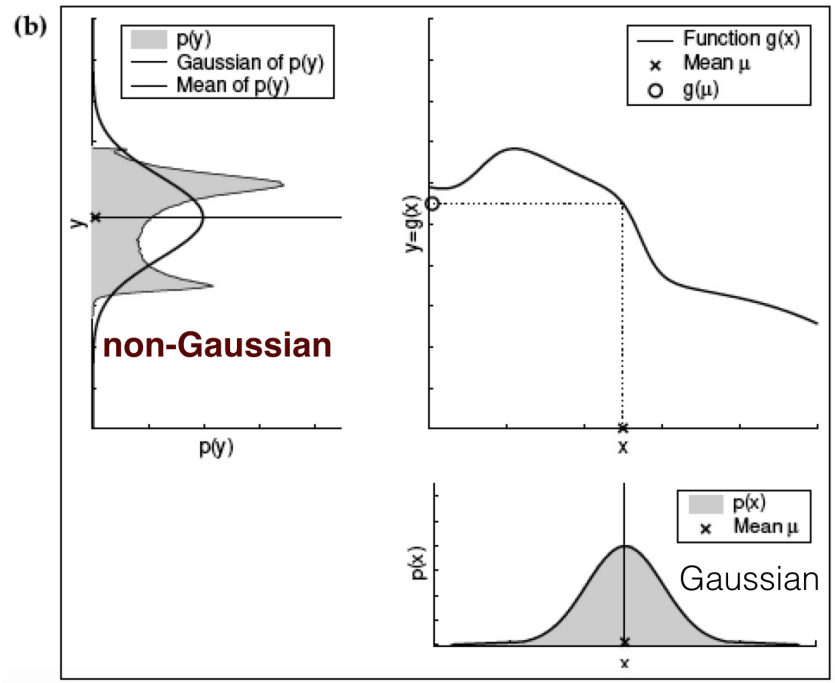
in the Kalman filter setting: $Y = x_t$ and $X = x_{t-1}$

linear transformation of Gaussian random variable X



$$
\begin{aligned}
X &\sim \mathcal{N}(x|\mu, \sigma^2) \\
y &= ax + b \\
Y &\sim \mathcal{N}(y|a\mu + b, a^2\sigma^2)
\end{aligned}
$$

nonlinear transformation of Gaussian random variable X



$$
\begin{aligned}
X &\sim \mathcal{N}(x|\mu, \sigma^2) \\
y &= g(x), \quad g \text{ nonlinear} \\
Y & \quad \text{is not a Gaussian random variable}
\end{aligned}
$$

Non-Gaussian Distributions

- The non-linear functions lead to non-Gaussian distributions
- Kalman filter is not applicable anymore!

What can be done to resolve this?

Local linearization!

Reminder: Taylor expansion around $\boldsymbol{x}_0$ for a vector-valued function $\boldsymbol{f}(\boldsymbol{x})$:

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}_0) + \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}}_{|\boldsymbol{x}=\boldsymbol{x}_0} \cdot (\boldsymbol{x} - \boldsymbol{x}_0) + \cdots ,$$

where

$$\boldsymbol{J}(\boldsymbol{x}) = \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}}$$

is the **Jacobian** of $\boldsymbol{f}$.

Reminder: Jacobian

- Given a vector-valued function with m components

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{pmatrix} f_1(\boldsymbol{x}) \\ f_2(\boldsymbol{x}) \\ \vdots \\ f_m(\boldsymbol{x}) \end{pmatrix},$$

  where $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$.

- The Jacobian matrix is defined as

$$\boldsymbol{J}(\boldsymbol{x}) = \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}} = \begin{pmatrix} \frac{\partial f_1(\boldsymbol{x})}{\partial x_1} & \frac{\partial f_1(\boldsymbol{x})}{\partial x_2} & \cdots & \frac{\partial f_1(\boldsymbol{x})}{\partial x_n} \\ \frac{\partial f_2(\boldsymbol{x})}{\partial x_1} & \frac{\partial f_2(\boldsymbol{x})}{\partial x_2} & \cdots & \frac{\partial f_2(\boldsymbol{x})}{\partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_m(\boldsymbol{x})}{\partial x_1} & \frac{\partial f_m(\boldsymbol{x})}{\partial x_2} & \cdots & \frac{\partial f_m(\boldsymbol{x})}{\partial x_n} \end{pmatrix}.$$

- It is a non-square matrix $m \times n$ in general

Extended Kalman filter (EKF) linearization: 1.order Taylor expansion

- **prediction**

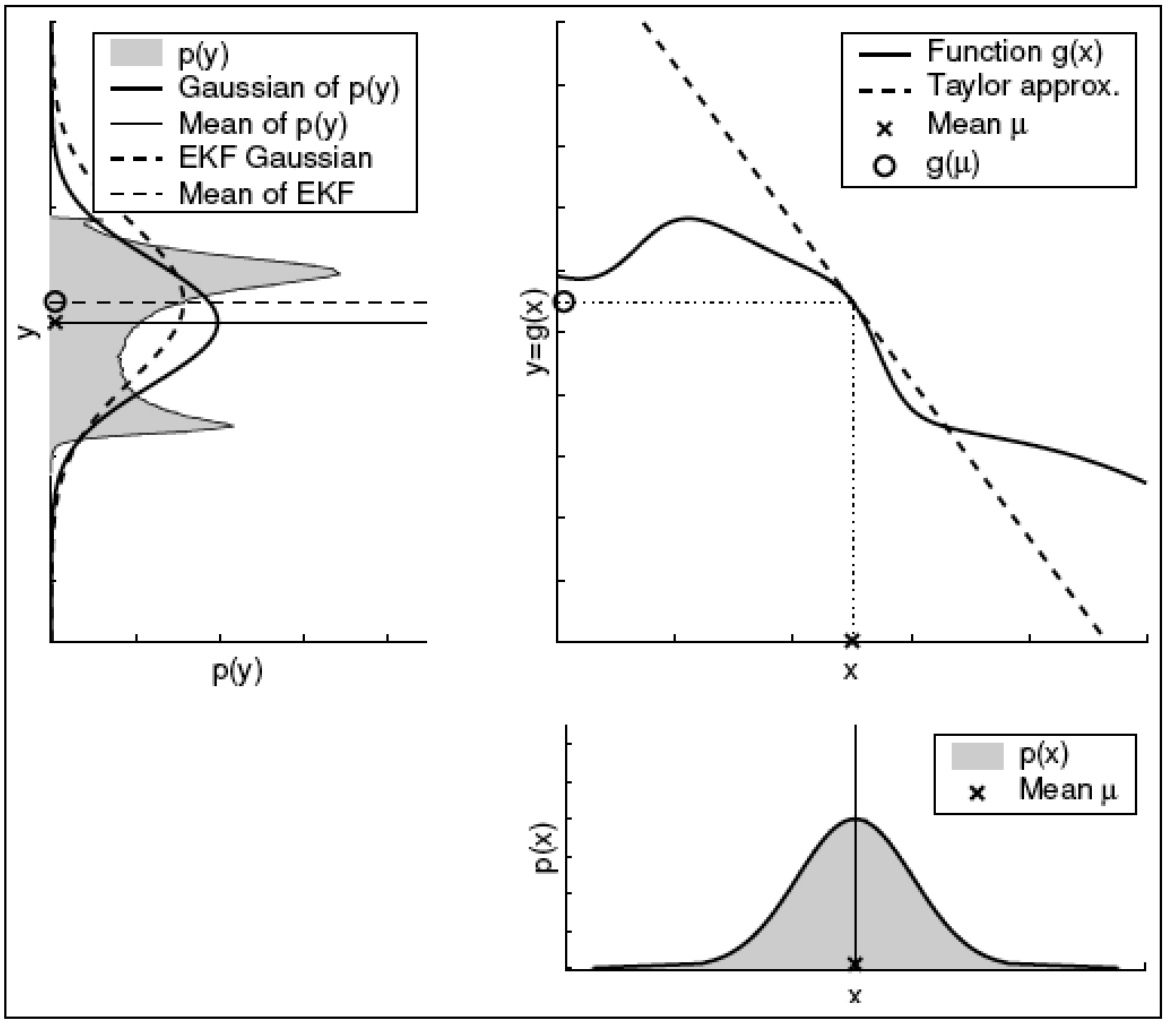
$$\begin{aligned} x_t &= g(u_t, x_{t-1}) + \epsilon_t\,, \quad \epsilon_t \sim N(0, R_t) \\ &\approx g(u_t, \mu_{t-1}) + \underbrace{\frac{\partial g(u_t, x_{t-1})}{\partial x_{t-1}}\Big|_{x_{t-1}=\mu_{t-1}}}_{=:G_t} \cdot (x_{t-1} - \mu_{t-1}) + \epsilon_t \end{aligned}$$

- **measurement update**

$$\begin{aligned} z_t &= h(x_t) + \delta_t\,, \quad \delta_t \sim N(0, Q_t) \\ &\approx h(\bar{\mu}_t) + \underbrace{\frac{\partial h(x_t)}{\partial x_t}\Big|_{x_t=\bar{\mu}_t}}_{=:H_t} \cdot (x_t - \bar{\mu}_t) + \delta_t \end{aligned}$$

G_t and H_t are Jacobian matrices

Extended Kalman filter - Linearization



Extended Kalman filter (EKF) - Algorithm

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

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

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

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

5: $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$ — measurement update (sense)

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

7: return μ_t, Σ_t — measurement update (sense)

comparison KF vs EKF:

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

2: $\bar{\mu}_t = A_t \, \mu_{t-1} + B_t \, u_t$ } prediction (move)

3: $\bar{\Sigma}_t = A_t \, \Sigma_{t-1} \, A_t^T + R_t$ } prediction (move)

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

5: $\mu_t = \bar{\mu}_t + K_t(z_t - C_t \, \bar{\mu}_t)$ } measurement update (sense)

6: $\Sigma_t = (I - K_t \, C_t) \, \bar{\Sigma}_t$ } measurement update (sense)

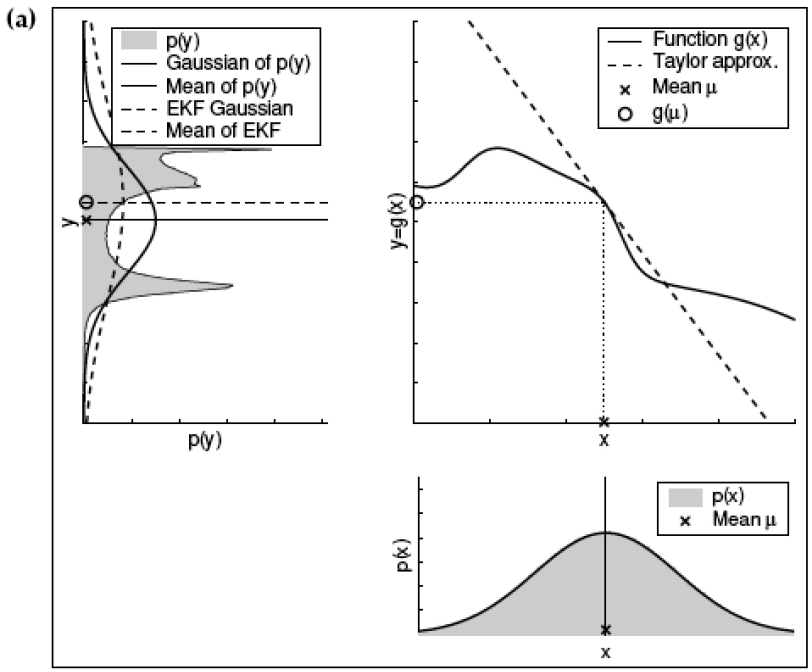
7: return μ_t, Σ_t } measurement update (sense)

	Kalman filter	EKF
Line 2	$A_t \, \mu_{t-1} + B_t \, u_t$	$g(u_t, \mu_{t-1})$
Line 5	$C_t \, \bar{\mu}_t$	$h(\bar{\mu}_t)$
Line 3	A_t	G_t
Line 4,6	C_t	H_t

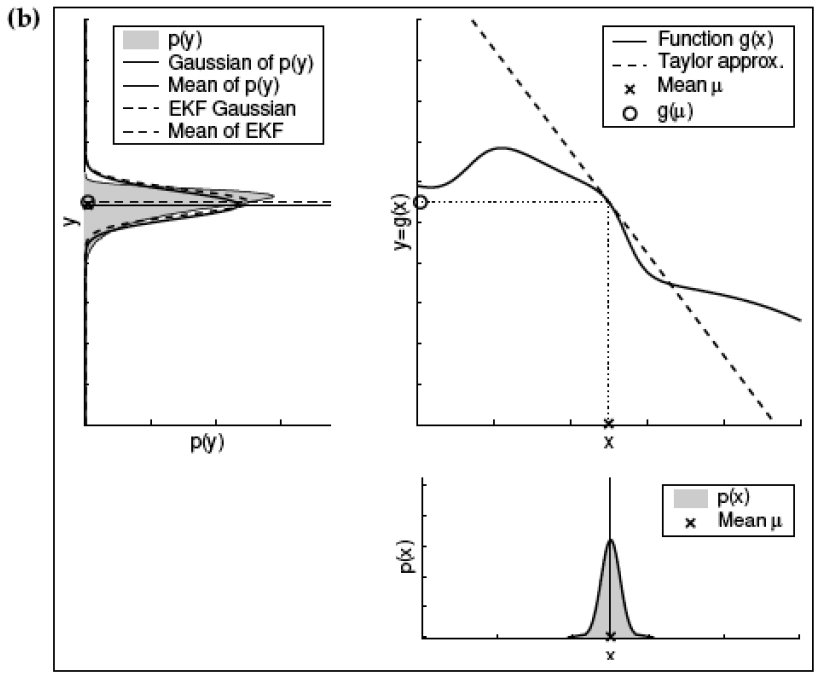
Extended Kalman filter - performance

the performance depends on the **degree of uncertainty**:

high uncertainty



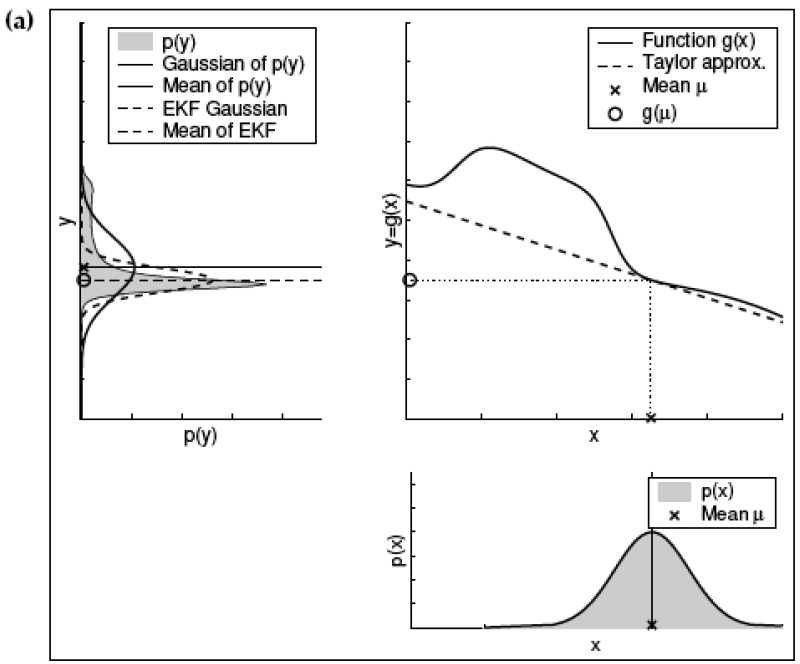
low uncertainty



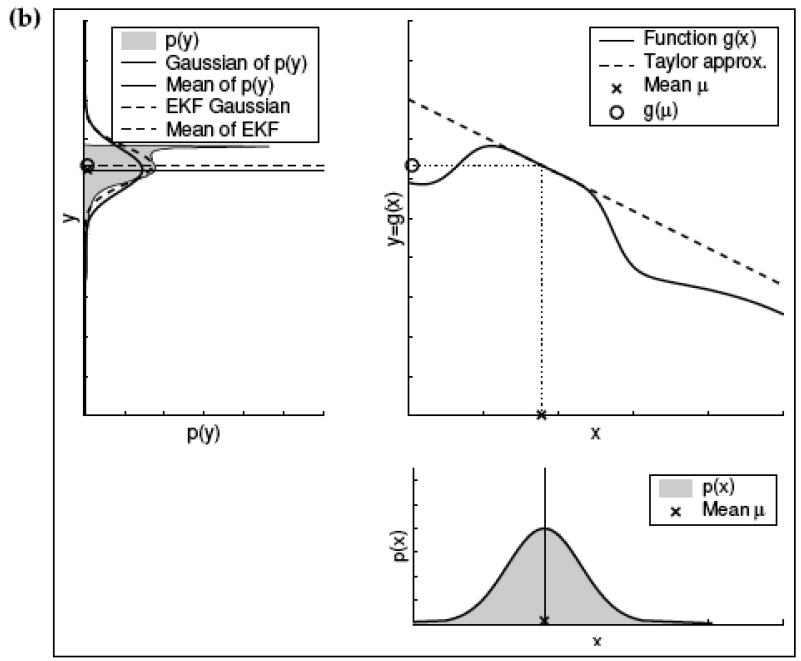
Extended Kalman filter - performance

the performance depends on the **degree of local nonlinearity**:

high degree



low degree



The Extended Kalman filter - probability distributions

1. **state transition probability density**
   motion under Gaussian noise for the linearized model leads to

$$p(x_t|x_{t-1},u_t) \approx \det(2\pi R_t)^{-\frac{1}{2}} \exp\Big\{-\frac{1}{2}[x_t - \underbrace{g(u_t,\mu_{t-1}) - G_t\cdot(x_{t-1}-\mu_{t-1})}_{\text{mean}}]^T R_t^{-1}[x_t - \underbrace{g(u_t,\mu_{t-1}) - G_t\cdot(x_{t-1}-\mu_{t-1})}_{\text{mean}}]\Big\}$$

2. **measurement probability density**
   measurement under Gaussian noise for the linearized model leads to

$$p(z_t|x_t) \approx \det(2\pi Q_t)^{-\frac{1}{2}} \exp\Big\{-\frac{1}{2}[x_t - \underbrace{h(\bar{\mu}_t) - H_t\cdot(x_t-\bar{\mu}_t)}_{\text{mean}}]^T Q_t^{-1}[x_t - \underbrace{h(\bar{\mu}_t) - H_t\cdot(x_t-\bar{\mu}_t)}_{\text{mean}}]\Big\}$$

Extended Kalman filter - summary

- Extension of the Kalman filter
- Ad-hoc solution to handle the nonlinearities
- Performs local linearizations
- Works well in practice for moderate non-linearities
- Complexity: $O(k^{2.4} + n^2)$

Unscented Kalman filter (UKF)

Julier and Uhlmann (1997), "unscented" means "without smell"

- unscented Kalman filter uses a different linearization technique : unscented transform (UT) and sigma-points
- no Jacobians in UKF (derivative-free)

Unscented transform algorithm (UT)

- consider a state variable $\boldsymbol{x} \in \mathbb{R}^n$ that follows a normal distribution

$$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \Sigma_x)$$

- if $\boldsymbol{x}$ is transformed by an arbitrary (nonlinear) function $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$ what is the mean and the covariance of $\boldsymbol{y}$?

Unscented transform algorithm (UT)

1. define sigma points $\boldsymbol{\chi}_i$ and weights W_i for $\boldsymbol{x} \in \mathbb{R}^n$:

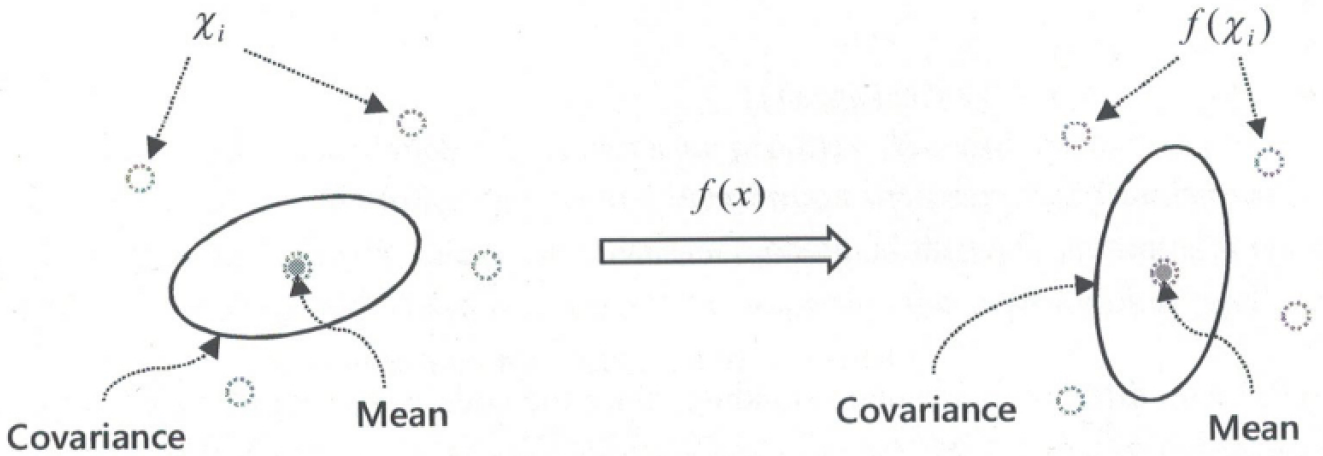
$$\begin{aligned} \boldsymbol{\chi}_1 &= \boldsymbol{\mu} & W_1 &= \frac{\kappa}{n+\kappa} \\ \boldsymbol{\chi}_{i+1} &= \boldsymbol{\mu} + \boldsymbol{u}_i & W_{i+1} &= \frac{1}{2(n+\kappa)} \quad i = 1, 2, \ldots, \\ \boldsymbol{\chi}_{i+n+1} &= \boldsymbol{\mu} - \boldsymbol{u}_i & W_{i+n+1} &= \frac{1}{2(n+\kappa)} \quad i = 1, 2, \ldots, \end{aligned}$$

   where $\boldsymbol{u}_i$ is row vector from the matrix U with $U^T U = (n+\kappa)\Sigma_x$ (Cholesky factorization - a definition of $\sqrt{\Sigma}_x$)
   in Matlab: $U = chol((n+\kappa) * \Sigma)$, chol = Cholesky fact.
   κ is a scaling parameter (often set to zero):

2. mean and covariance of the function $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$ follow from sigma points and weights

$$\begin{aligned} \boldsymbol{\mu}_y &= \sum_{i=1}^{2n+1} W_i \boldsymbol{f}(\boldsymbol{\chi}_i) \\ \Sigma_y &= \sum_{i=1}^{2n+1} W_i [\boldsymbol{f}(\boldsymbol{\chi}_i) - \boldsymbol{\mu}_y][\boldsymbol{f}(\boldsymbol{\chi}_i) - \boldsymbol{\mu}_y]^T \end{aligned}$$

Unscented transform algorithm (UT)

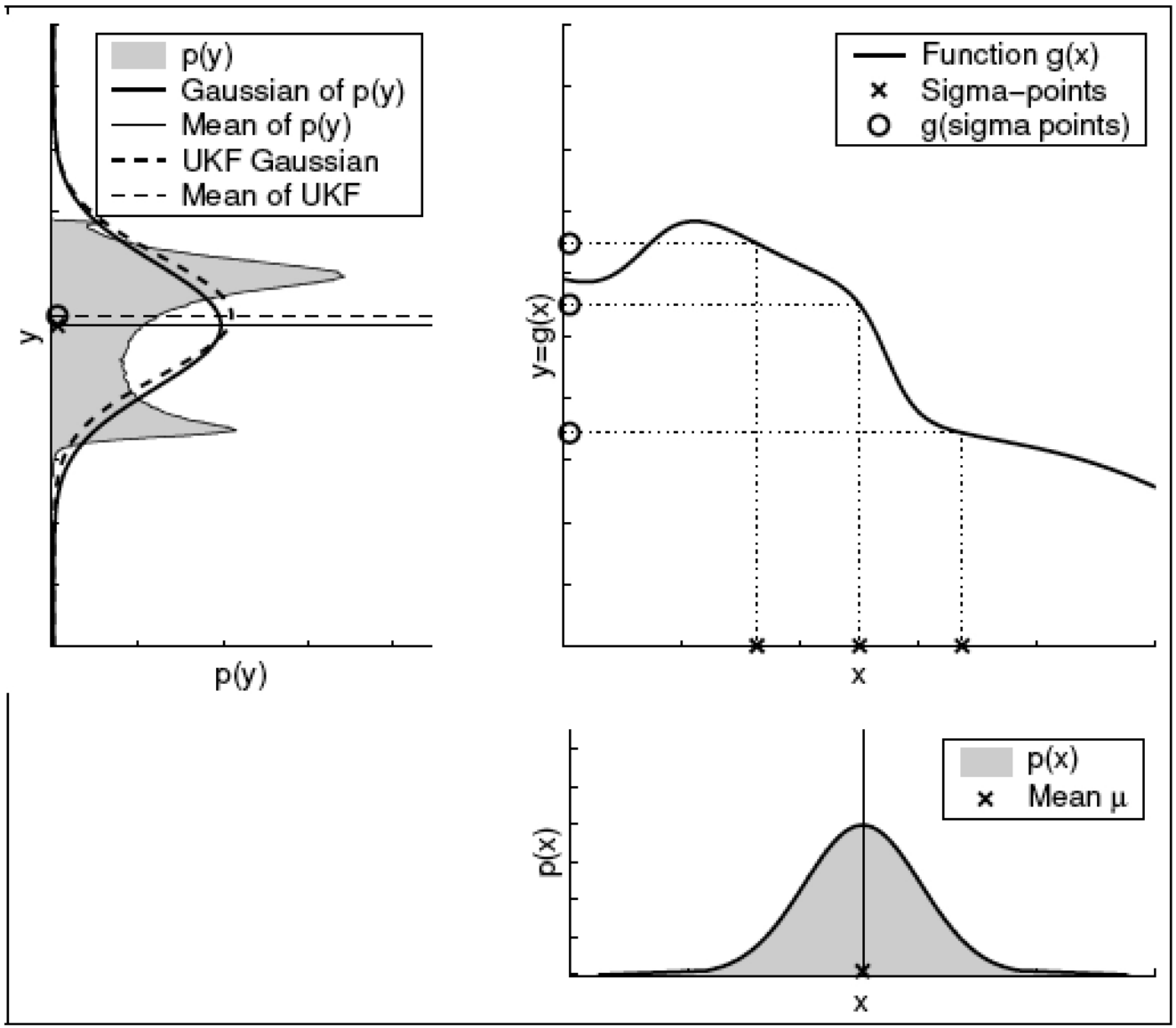


Note that sigma points $\boldsymbol{\chi}_i$ and weights W_i satisfy the following characteristics

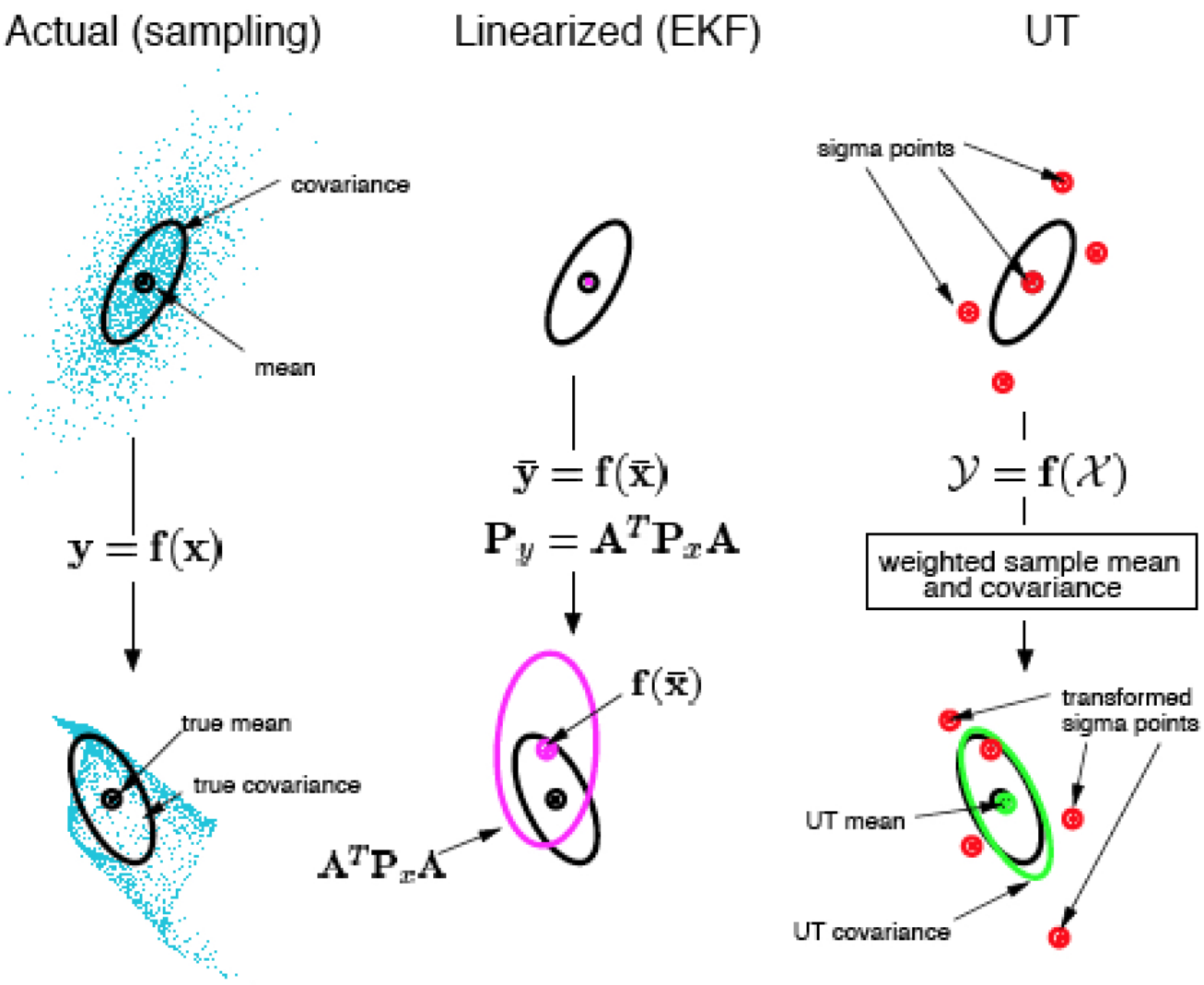
$$\boldsymbol{\mu} = \sum_{i=1}^{2n+1} W_i \boldsymbol{\chi}_i$$

$$\Sigma_x = \sum_{i=1}^{2n+1} W_i [(\boldsymbol{\chi}_i - \boldsymbol{\mu}_y][\boldsymbol{\chi}_i - \boldsymbol{\mu}_y]^T$$

Unscented transform algorithm (UT)



Unscented transform algorithm (UT) - comparison



Unscented Kalman Filter - algorithm

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

2: $\mathcal{X}_{t-1} = (\mu_{t-1} \quad \mu_{t-1} + \gamma\sqrt{\Sigma_{t-1}} \quad \mu_{t-1} - \gamma\sqrt{\Sigma_{t-1}})$

3: $\bar{\mathcal{X}}_t^* = g(u_t, \mathcal{X}_{t-1})$

4: $\bar{\mu}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{X}}_t^{*[i]}$

5: $\bar{\Sigma}_t = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{X}}_t^{*[i]} - \bar{\mu}_t)(\bar{\mathcal{X}}_t^{*[i]} - \bar{\mu}_t)^T + R_t$

6: $\bar{\mathcal{X}}_t = (\bar{\mu}_t \quad \bar{\mu}_t + \gamma\sqrt{\bar{\Sigma}_t} \quad \bar{\mu}_t - \gamma\sqrt{\bar{\Sigma}_t})$

7: $\bar{\mathcal{Z}}_t = h(\bar{\mathcal{X}}_t)$

8: $\hat{z}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{Z}}_t^{[i]}$

9: $S_t = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)^T + Q_t$

10: $\bar{\Sigma}_t^{x,z} = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{X}}_t^{[i]} - \bar{\mu}_t)(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)^T$

11: $K_t = \bar{\Sigma}_t^{x,z} S_t^{-1}$

12: $\mu_t = \bar{\mu}_t + K_t(z_t - \hat{z}_t)$

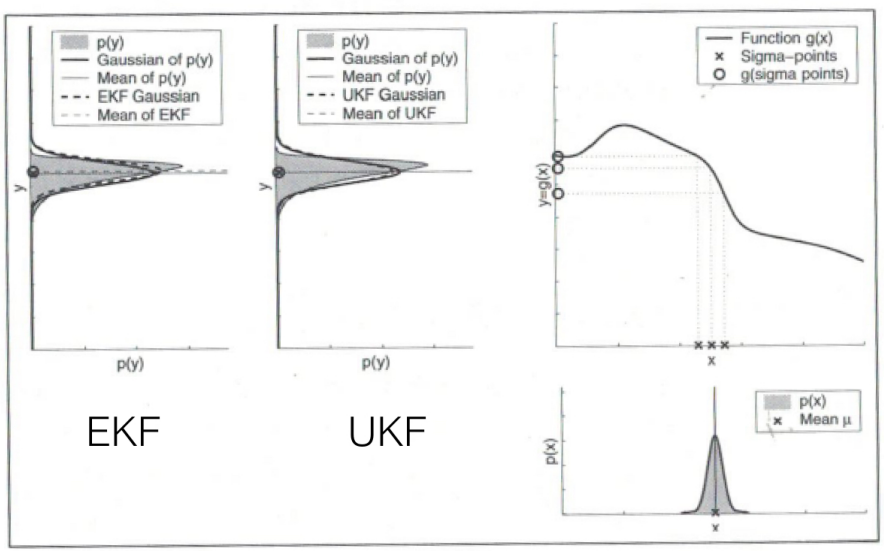
13: $\Sigma_t = \bar{\Sigma}_t - K_t S_t K_t^T$

14: *return* μ_t, Σ_t

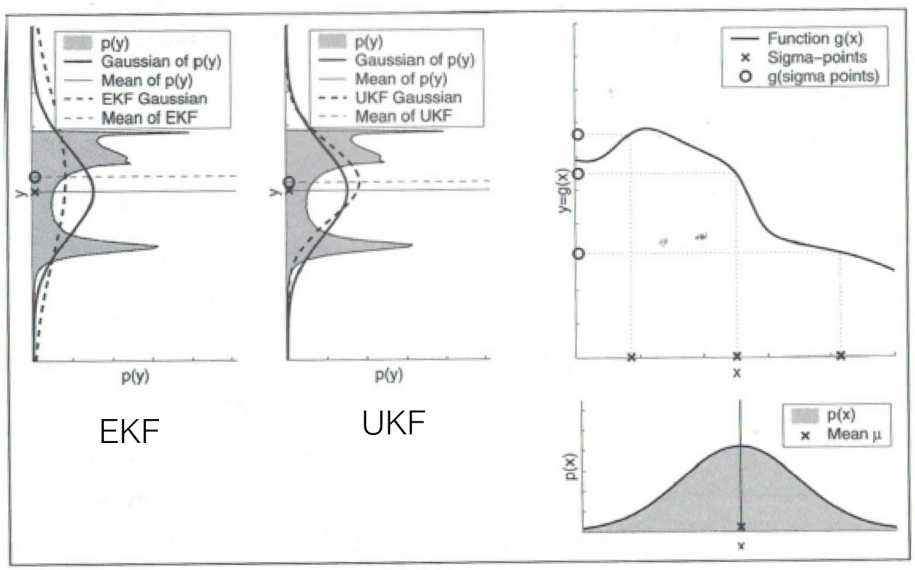
Unscented Kalman Filter - performance

the performance depends on the **degree of uncertainty**:

low uncertainty



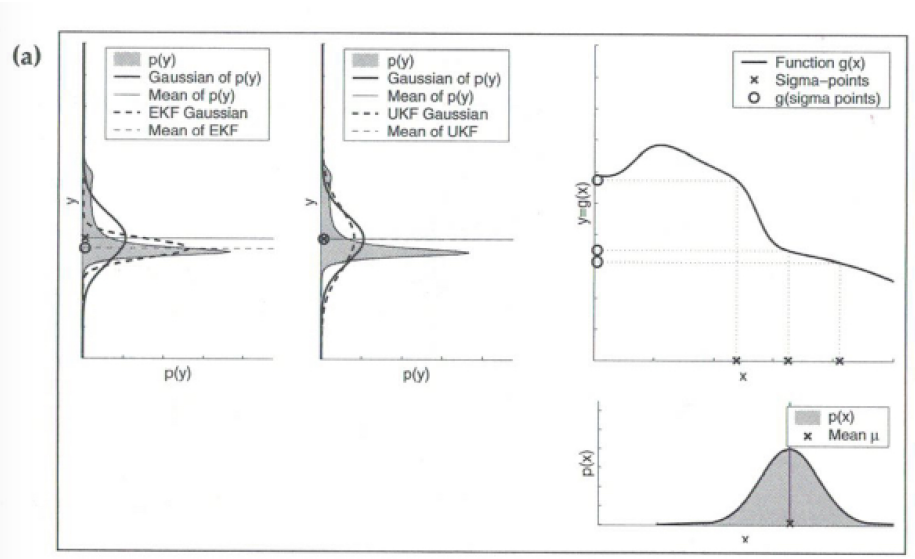
high uncertainty



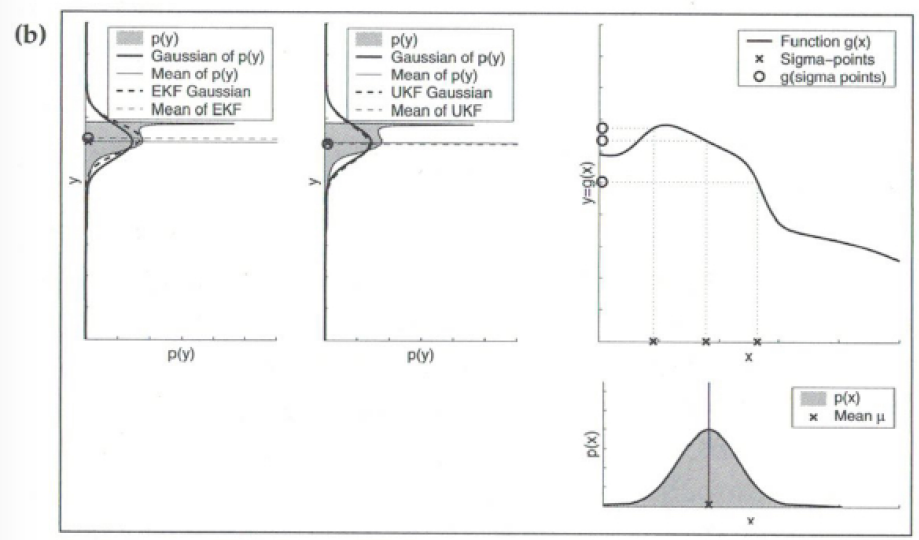
Unscented Kalman Filter - performance

the performance depends on the **degree of local nonlinearity**:

high degree



low degree



Unscented Kalman filter - summary

- **derivative free** filter (no Jacobian is needed)
- overall number of computations are of the same order as for EKF
- performs often better than EKF for problems with high level of uncertainty and high level of local nonlinearity