MOBILE ROBOTS 7/8- Uncertainties

Prof. Francesco Mondada

EPFL

Bibliography and Sources

***Elements of Robotics* (ch. 8),** M. Ben-Ari, F. Mondada, Springer, 2017.

***Springer Handbook of Robotics* (ch 5),** B. Siciliano, and O. Khatib (Eds.), 2nd edition, Springer, 2016.

Probabilistic Robotics by Thrun, Burgard, and Fox, MIT Press, 2005

Mobile Robots - EPFL - J.-C. Zufferey, Felix Schill, 2013.

Table of Contents

Motivation

How can we represent a world

- which is perceived with errors,
- on which we do actions that do not correspond exactly to our orders,
- with maps that are uncertain?

Let use probabilistic (random) variables

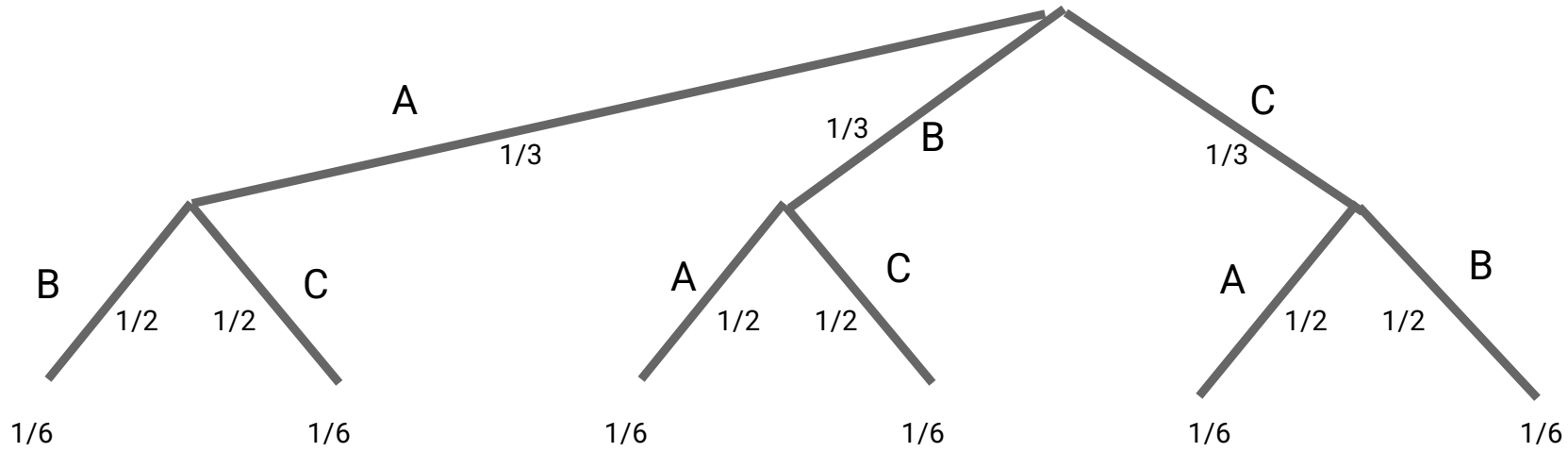
Conditional Probability

Random variables often carry information about other random variables. Suppose we already know that *Y* value is *y*, and we would like to know the probability that *X* value is *x* conditioned on that fact:

$$p(x \mid y) = \frac{p(x, y)}{p(y)}$$

Where *p(x,y)* is the probability of having *x* and *y*.

Conditional Probability



$$p(x \mid y) = \frac{p(x, y)}{p(y)}$$

Bayes Rule

The Bayes rule relates a conditional of the type $p(x \mid y)$ to its "inverse" $p(y \mid x)$:

$$p(x \mid y) = \frac{p(y \mid x) \cdot p(x)}{p(y)}$$

Bayes rule plays a predominant role in probabilistic robotics (and probabilistic inference in general).

Bayes Rule applied to robotics

$$p(x \mid y) = \frac{p(y \mid x) \cdot p(x)}{p(y)}$$

If ***x* is the robot's state** that we would like to infer from ***y* (the sensor data)**:

- $p(x)$ is referred to as *prior probability distribution*, which summarizes the knowledge (or ignorance) we have regarding the robot state X **prior** to incorporating the sensor data y.
- $p(x \mid y)$ is called the *posterior probability distribution* over X meaning the knowledge we have regarding the robot state X **after** incorporating the sensor data y.
- $p(y \mid x)$ is often coined *likelihood* or *generative model*, since it describes how state variables X impacts on sensor measurements Y. This information can typically come from a map that gives information on the environment based on the state of the robot (position).

Bayes Rule and Normalisation

Note that the denominator of the Bayes rule, *p(y)*, the general probability to have a given sensor readings, does not depend on *x*. For this reason, $p(y)^{-1}$ is often seen as a normalizer:

$$p(x \mid y) = \eta \cdot p(y \mid x) \cdot p(x)$$

The posterior integral just needs to be equal to 1.

To sum up, the Bayes rule provides a convenient way to compute a posterior $p(x \mid y)$ using the "inverse" conditional probability $p(y \mid x)$ along with the prior probability $p(x)$.

Uncertainty in Motion & Sensing

Table 8.2 Localization with uncertainty in sensing and motion
sensor=after multiplying by the sensor uncertainty
norm=after normalization
right=after moving right one position

position	0	1	2	3	4	5	6	7
door?	•	•			•	•	•	
initial	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13
sensing → sensor	0.11	0.11	0.01	0.01	0.11	0.11	0.11	0.01
norm	0.19	0.19	0.02	0.02	0.19	0.19	0.19	0.02
moving → right	0.05	0.17	0.17	0.04	0.04	0.17	0.19	0.17
sensing → sensor	0.05	0.17	0.02	0.00	0.03	0.15	0.17	0.02
norm	0.08	0.27	0.03	0.01	0.06	0.25	0.28	0.03
moving → right	0.06	0.12	0.23	0.05	0.01	0.07	0.23	0.25
sensing → sensor	0.05	0.10	0.02	0.01	0.01	0.06	0.21	0.02
norm	0.11	0.21	0.05	0.01	0.02	0.13	0.43	0.05

Table of Contents

State variable

In probabilistic robotics, the *state x* is the collection of all aspects of the robot and its environment (both constituting a single dynamical system) that can impact the future of the robot (defined in the state).

State variables in mobile robotics typically include:

- *robot pose*: location & orientation relative to a global coordinate frame
- configuration of the robot's manipulators
- robot velocity (dynamic state)
- sensor status or parameters (e.g. inertial sensor biases)
- location and properties of surrounding objects in the environment
- location and velocities of moving objects in the environment
- etc. (the list is endless!)

In this chapter, we assume a static world and focus on estimating only the pose of a kinematic mobile robot (without dynamic objects, nor changing sensor parameters).

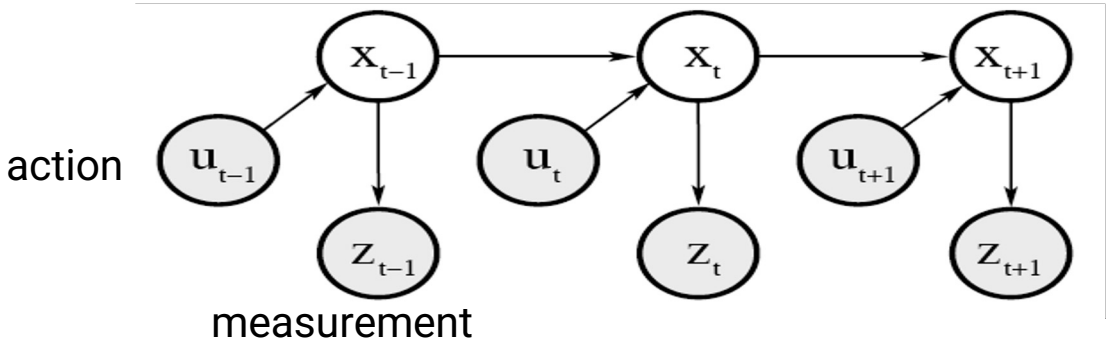
Complete State and Markov Chain

A state x_t is said to be *complete* if it fully captures all the information that could influence its future evolution.

Completeness entails that knowledge of past states, measurements, or controls carry no additional information that would help us predict the future more accurately.

Complete State and Markov Chain

A *Markov chain* is a temporal process that meets this condition of state completeness. A Markov chain describes at successive times the (complete) state of a system.



Note: The notion of state completeness is mostly of theoretical importance (i.e. required to derive the Bayes filter). In practice, it is impossible to specify a complete state for any realistic robot system. A complete state includes not just all aspects of the environment that may have an impact on the future, but also the robot itself, the content of its computer memory, the brain dumps of surrounding people, etc.

Types of Robot–Environment Interactions

Environment sensor **measurements** (=observation, percept)

- Process by which the robot uses its sensors to obtain information about the state of its environment.
- Environment *measurement data* will be denoted z_t where

$$z_{t1:t2} = z_{t1}, z_{t1+1}, z_{t1+2}, \ldots, z_{t2}$$

denotes the set of all measurements acquired from time t_1 to t_2.

Types of Robot–Environment Interactions

Control actions (or motion)

- Process by which the robot changes the state of the world by actively asserting forces on the robot's environment.
- *Control data* u_t carry information about the *change of state*. Typical control data include: velocity of robot actuators or odometer data (!)
- As before, a sequence of control data will be denoted:

$$u_{t1:t2} = u_{t1}, u_{t1+1}, u_{t1+2}, \ldots, u_{t2}$$

Types of Robot–Environment Interactions

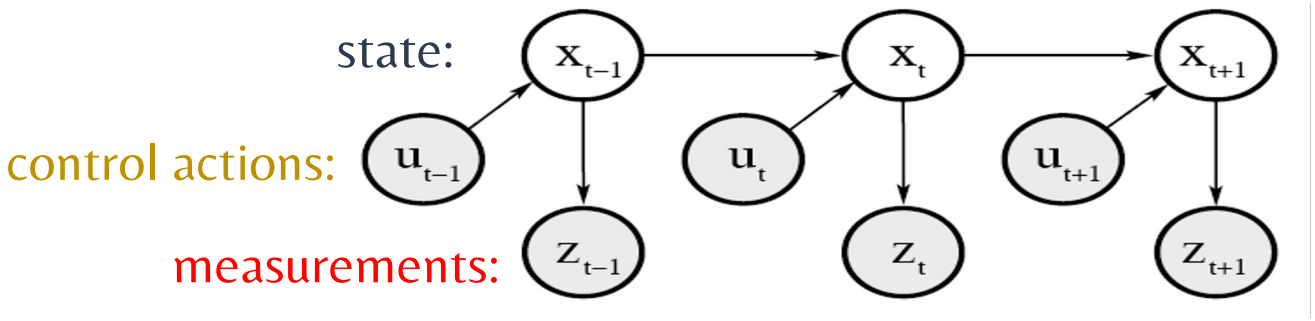
The distinction between measurement and control is a crucial one, as both types of data play fundamentally different roles in the material yet to come.

- **Perception** provides information about the environment's state, hence it **tends to increase the robot's knowledge**.

- **Motion**, on the other hand, **tends to induce a loss of knowledge** due to the inherent noise in robot actuation and the stochasticity of robot environments; (although sometimes a control makes the robot more certain about the state.)

Evolution of State and Measurement

The evolution of state and measurements is governed by probabilistic laws:



The *state transition probability* is the probabilistic law characterizing the evolution of state:

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

The *measurement probability* is the process by which measurements are generated:

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

Such a generative model is also known as dynamic Bayes network (DBN), which belongs to the class of **hidden** Markov model (HMM).

Belief Distribution

A *belief* reflects the robot's internal knowledge about the state, *which cannot be measured directly (hidden state)*.

A belief distribution assigns a probability to each possible hypothesis with respect to the true state ("what is the probability to be somewhere?").

Belief distributions are *posterior probabilities* over state variables conditioned on the available data:

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

If this posterior is calculated before incorporating the latest measurements, it is referred to as *prediction* and denoted:

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

This terminology reflects the fact that $\overline{bel(x_t)}$ predicts the state at time *t* based on the previous state posterior, before incorporating the measurement at time *t*. Calculating $bel(x_t)$ from $\overline{bel(x_t)}$ is called *correction* or *measurement update*.

Table of Contents

Bayes Filter

The most general algorithm for calculating beliefs is the *Bayes filter*. It is a recursive filter and the following pseudo-code depicts a single iteration of it:

Algorithm Bayes_filter$(bel(x_{t-1}), u_t, z_t)$**:**

for all x_t *do*

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

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

endfor

return $bel(x_t)$

prediction

measurement update

action model

measurement model

Two steps:

1. *Control update* or *prediction* (based on the theorem of total probability).
2. *Measurement update* or *correction* (based on the Bayes rule).

This algorithm can only be implemented in the form stated here for very simple estimation problems. One either needs to be able to carry out the integration in step 1 and the multiplication in step 2 in closed form, or one needs to restrict himself to finite state spaces, so that the integral becomes a finite sum.

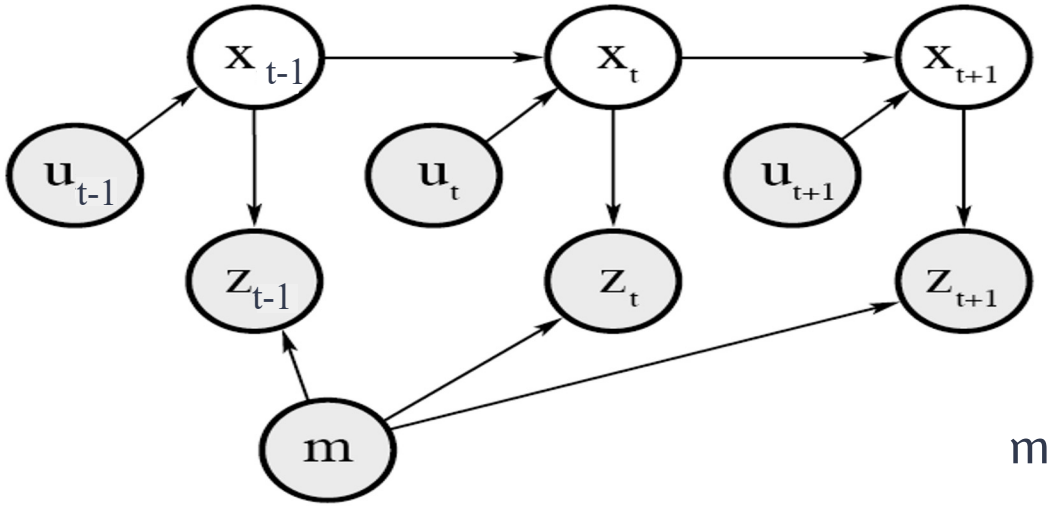
Uncertainty in Motion & Sensing

Table 8.2 Localization with uncertainty in sensing and motion
sensor=after multiplying by the sensor uncertainty
norm=after normalization
right=after moving right one position

	position	0	1	2	3	4	5	6	7
	door?	•	•			•	•	•	
	initial	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13
measurement update	sensor	0.11	0.11	0.01	0.01	0.11	0.11	0.11	0.01
	norm	0.19	0.19	0.02	0.02	0.19	0.19	0.19	0.02
prediction	right	0.05	0.17	0.17	0.04	0.04	0.17	0.19	0.17
measurement update	sensor	0.05	0.17	0.02	0.00	0.03	0.15	0.17	0.02
	norm	0.08	0.27	0.03	0.01	0.06	0.25	0.28	0.03
prediction	right	0.06	0.12	0.23	0.05	0.01	0.07	0.23	0.25
measurement update	sensor	0.05	0.10	0.02	0.01	0.01	0.06	0.21	0.02
	norm	0.11	0.21	0.05	0.01	0.02	0.13	0.43	0.05

Localisation : the Bayesian Perspective

The robot is given a map m of its environment and its goal is to determine its position relative to this map given the perceptions of the environment and its movements.



state ~ robot pose

control actions

measurements

map

In probabilistic robotics, this problem is solved using a variant of the Bayes filter.

Markov Localisation

Markov localization is just a different name for the Bayes filter applied to the mobile robot localization problem:

Algorithm Markov_localization$(bel(x_{t-1}), u_t, z_t, m)$**:**

for all x_t *do*

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

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

endfor

return $bel(x_t)$

motion model (not always map dependent)

prediction

measurement update

measurement model (map dependent)

Markov localization can address the global localization problem (initial belief is uniform), the position tracking problem (initial belief is typically a tight Gaussian), and the kidnapped robot problem in static environments.

1D Example

a) Initial belief is uniform over all poses (global localization).

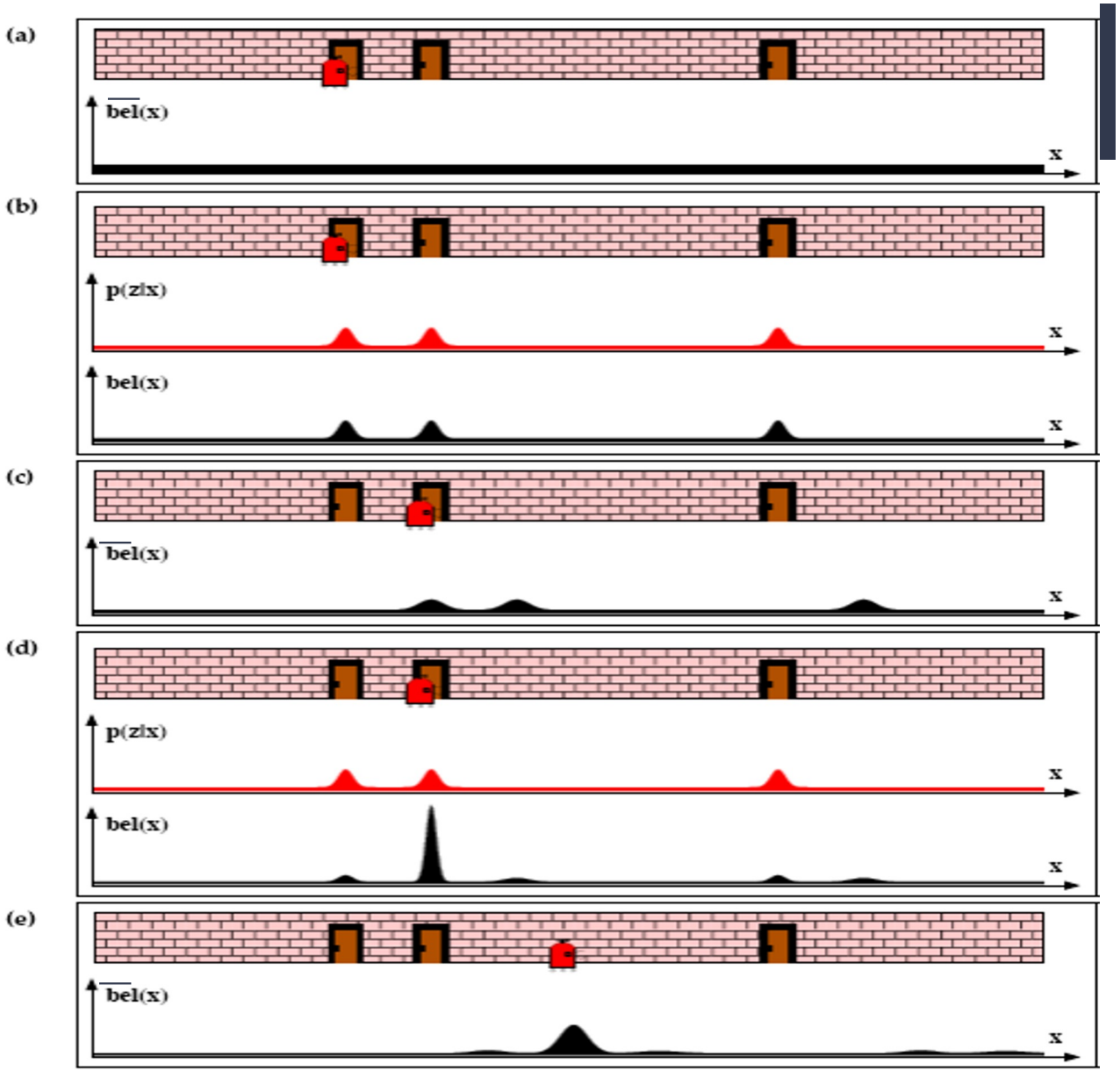
As the robot moves to the right, the 1st step of Bayes filter convolves its belief with the motion model $p(x_t \mid u_t, x_{t-1})$, not indicated here at the beginning.

b) As the robot queries its sensors and notices that it is adjacent to one of the doors, it multiplies its belief by $p(z_t \mid x_t, m)$ according to the 2nd step of the Bayes filter.

c) As the robot moves to the right, the 1st step of Bayes filter is applied again.

d) The 2nd measurement allows to correct the previous prediction. Now the robot is quite confident of having localized itself.

e) Robot belief after having moved further down the hallway (without further measurements).



Tractability Of Bayes Filter For Localisation

Since the Bayes filter is not a practical algorithm (numerical computation is feasible only for very specific cases), probabilistic algorithms for robot localization use approximations.

The nature of approximation has important ramifications on the complexity of the algorithm and the type of localization (e.g. global vs tracking).

Example of widely-used approximations:

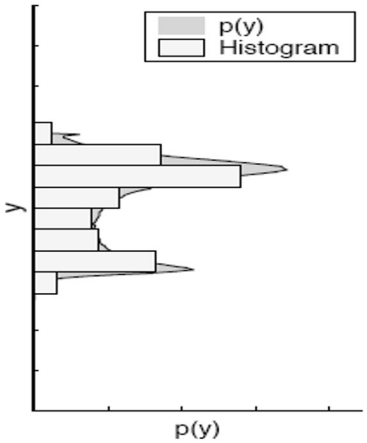
- Discretization of the belief space -> nonparametric, discrete filters
  - *Histogram filter -> grid localization:*
    - belief is discretized into an histogram (finite state spaces)
  - *Particle filter -> Monte Carlo Localization (MCL):*
    - represents the belief by a set of random state samples drawn from the belief
- Linearization and parameterization -> Gaussian filters
  - *Extended Kalman filter -> EKF localization:*
    - belief is represented using Gaussian(s)
    - motion and measurement models are linearized (using the Jacobian matrix)

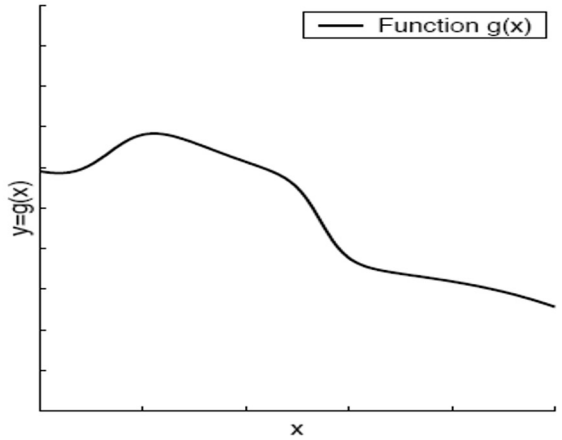
Histogram Filter (Markov Localization)

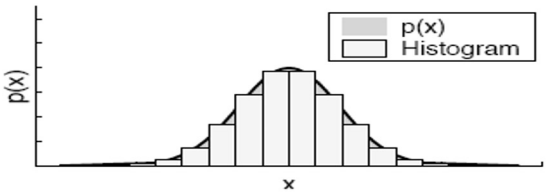
Belief is discretized into a n-dimensional histogram.

Simplest to understand.

Becomes intractable for large or high dimensional state spaces.







Algorithm Discrete_Bayes_filter$(\{p_{k,t-1}\}, u_t, z_t)$**:**

for all k *do*

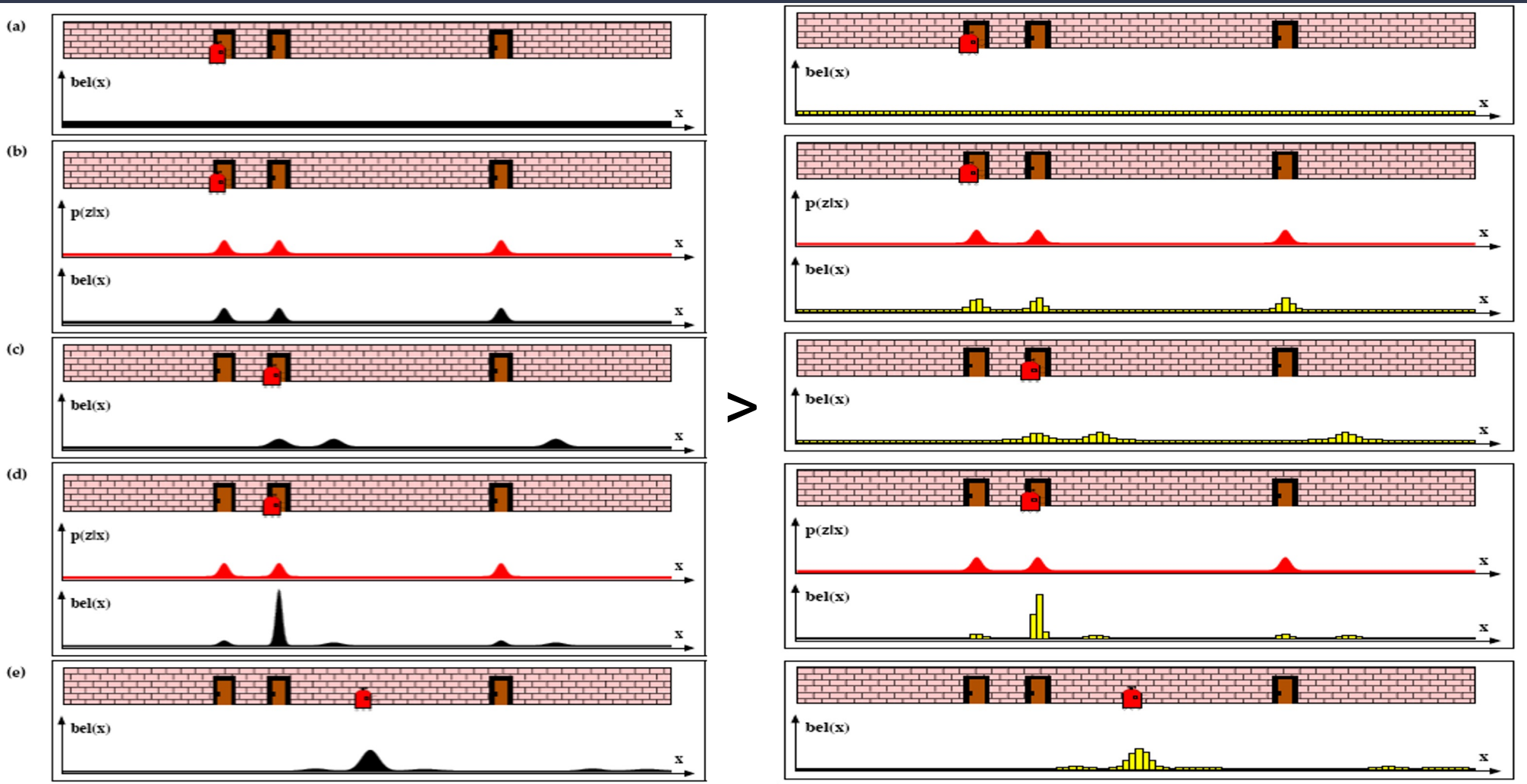
$$\bar{p}_{k,t} = \sum_i p(X_t = x_k \mid u_t, X_{t-1} = x_i)\ p_{i,t-1}$$

$$p_{k,t} = \eta\ p(z_t \mid X_t = x_k)\ \bar{p}_{k,t}$$

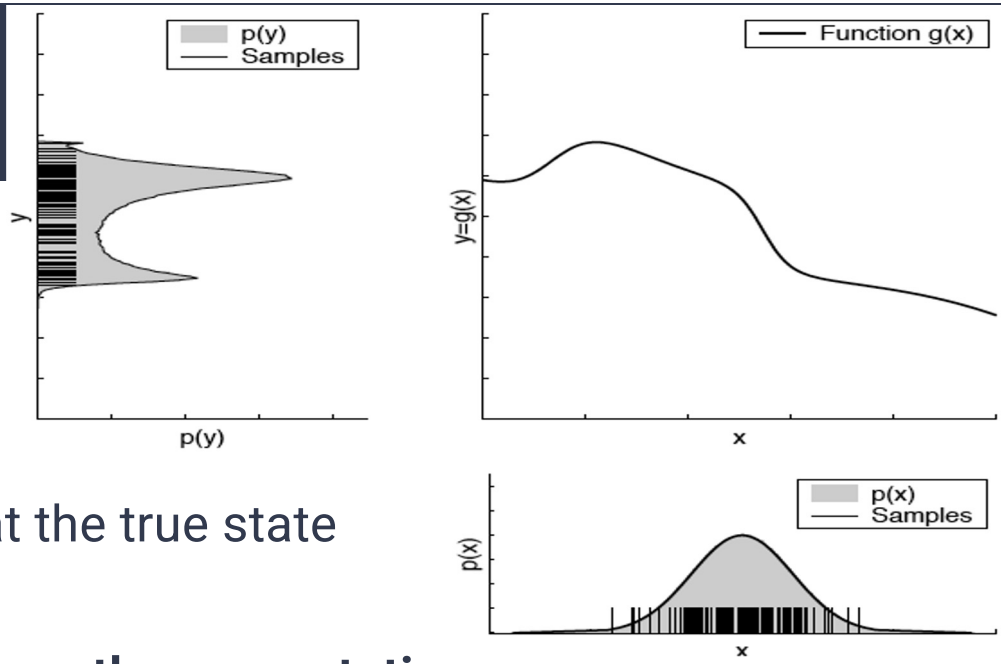
endfor

return $\{p_{k,t}\}$

1D Grid Example



Particle Filter



Particle filters represent a distribution by a set of samples.

The denser a subregion of the state space is populated by samples, the more likely it is that the true state falls into this region.

Such a representation is approximate, but focuses the computation on the most probable location, "forgetting" the locations that have very low probability.

Weights are assigned to particles through the measurement model and resampling allows to redistribute particles approximately according to the posterior $bel(x_t)$.

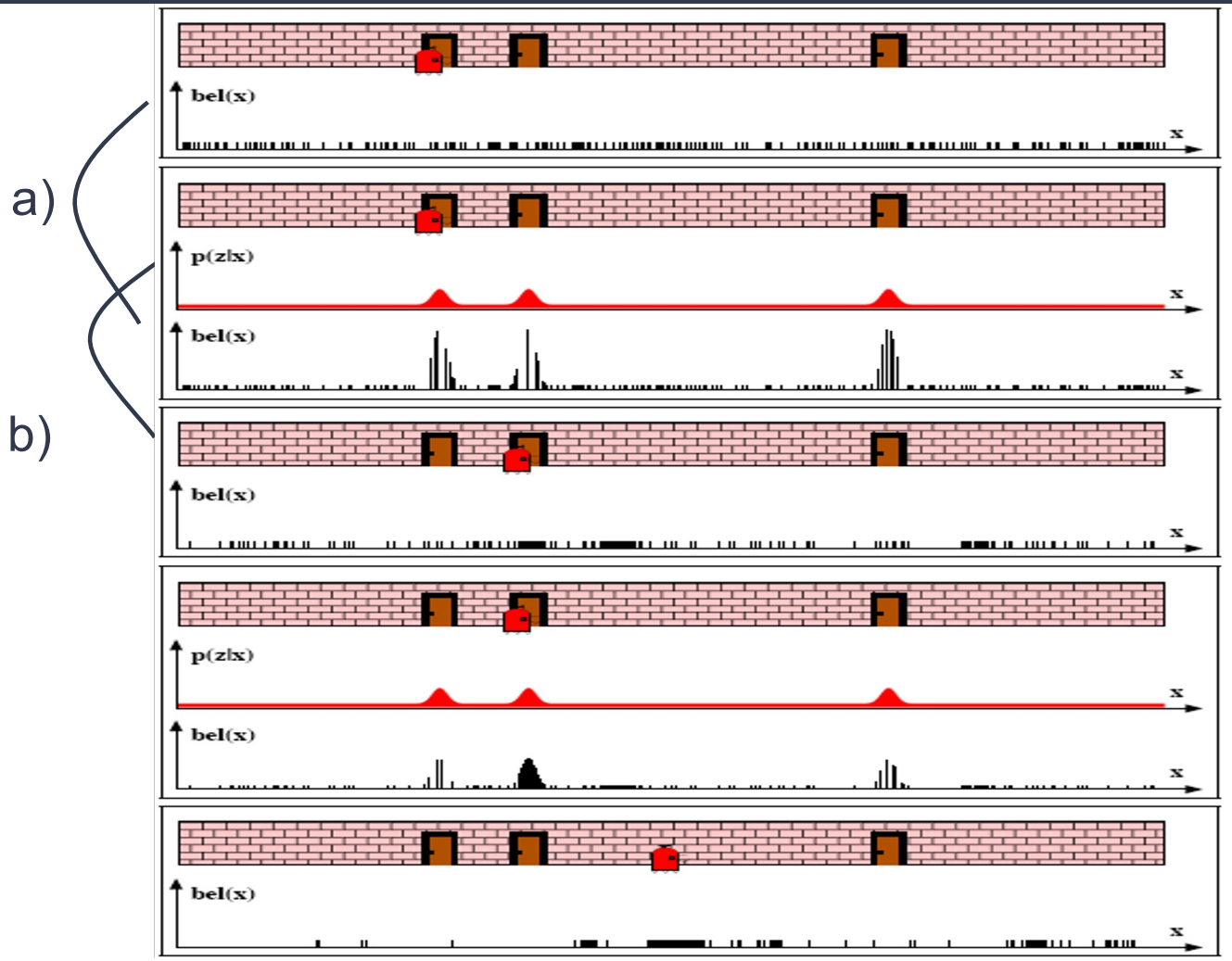
The resampling step is a probabilistic implementation of the Darwinian idea of *survival of the fittest*: it refocuses the particle set to regions in state space with high posterior probability.

1D Particle Filter (MCL) Example

MCL = Monte Carlo localization

a) weighing of the particles depending on the measurement model & resampling according to the weights

b) application of the motion model



Particle Filter – Example Algorithm

In particle filters, the samples of a posterior distribution are called *particles* and are denoted:

$$\mathcal{X}_t := x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[M]}$$

1: **Algorithm Particle_filter**($\mathcal{X}_{t-1}, u_t, z_t$):
2: $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3: for $m = 1$ to M do
4: sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$
5: $w_t^{[m]} = p(z_t \mid x_t^{[m]})$
6: $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7: endfor
8: for $m = 1$ to M do
9: draw i with probability $\propto w_t^{[i]}$
10: add $x_t^{[i]}$ to $\mathcal{X}_t$
11: endfor
12: return $\mathcal{X}_t$

Sampling generation of new particles from the old one using the probabilistic motion model (spreading)

Evaluation of the **weight** by incorporating the measurement z_t into the particle set. The weight is thus the probability of the measurement z_t under the particle $x_t^{[m]}$. Particles that have a position that fits well with the measurement get higher weight.

Particle Filter – Example Algorithm

In particle filters In particle filters, the samples of a posterior distribution are called *particles* and are denoted:

$$\mathcal{X}_t := x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[M]}$$

1: **Algorithm Particle_filter**($\mathcal{X}_{t-1}, u_t, z_t$):
2: $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3: for $m = 1$ to M do
4: sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$
5: $w_t^{[m]} = p(z_t \mid x_t^{[m]})$
6: $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7: endfor
8: for $m = 1$ to M do
9: draw i with probability $\propto w_t^{[i]}$
10: add $x_t^{[i]}$ to $\mathcal{X}_t$
11: endfor
12: return $\mathcal{X}_t$

Resampling. The algorithms draws with replacement M particles from the temporary belief $\overline{X_t}$. The drawing probability is proportional to the weight. Whereas before the resampling step, particles were an approximation of $\overline{bel(x_t)}$, after they are an approximation of $bel(x_t)$

2D Localisation on Thymio

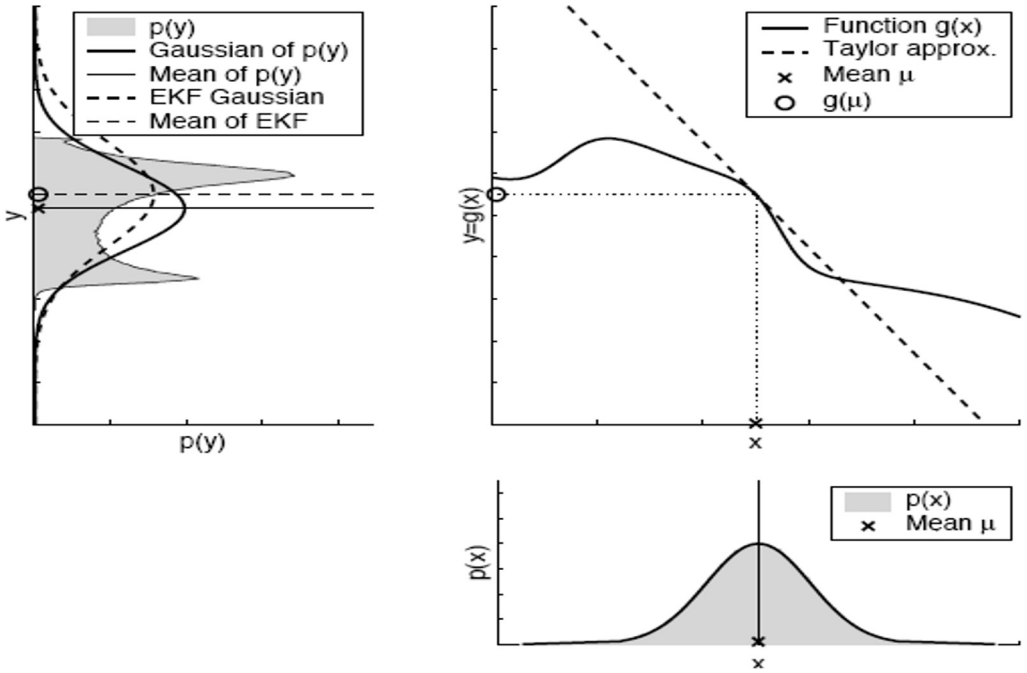
Wang, S., Colas, F., Liu, M., Mondada, F., & Magnenat, S. (2018). Localization of inexpensive robots with low-bandwidth sensors. In *Distributed Autonomous Robotic Systems* (pp. 545-558). Springer, Cham.

What should I remember

- Motivation of using probabilistic representations
- Conditional probability law and Bayes rule
  - meaning and use in a pose estimation use
- State variables
  - what is represented
  - complete state: understanding
  - markov chain: concept
- Interactions with environment and role in the probabilistic state estimation
  - measurements
  - actions
  - belief distribution
- Markov localisation
  - principle and computation
- Markov localisation
  - principle and computation

Kalman Filters

Kalman filters represent the belief by means of multinormals, i.e.mean vector and covariance matrix.



This approach allows to fuse information from several sensors in an “intelligent way”, by combining their statistical properties.

The aim of the Kalman filters is to minimize the a posteriori (after sensor integration) error covariance.

A Simple Scalar Example

Let consider a model of the position of a robot moving in 1D at constant speed. The system model is the following:

$$x_{t+1} = x_t + d + w_t$$

$$y_t = x_t + v_t$$

Where

x_t is the position at time t,
d is the displacement between two steps,
w_t is the displacement noise with variance q,
y_t the measured position,
v_t the measurement noise of variance r.

A Simple Scalar Example

Let compute the **prediction** of the mean, based on the previous mean:

$$\bar{\mu}_t = \mu_{t-1} + d$$

… and compute the **prediction** of the error variance:

$$\bar{\Sigma}_t = \Sigma_{t-1} + q$$

A Simple Scalar Example

If we now take a **measurement** we can have a look to the difference between measure and prediction, called *innovation*:

$$i_t = y_t - \bar{\mu}_t$$

… and compute an optimal gain for the correction (not developed here):

$$K_t = \frac{\bar{\Sigma}_t}{\bar{\Sigma}_t + r}$$

A Simple Scalar Example

Based on the optimal gain, we can generate an estimation *a posteriori* that takes into account the **measurement** through the innovation and the gain:

$$\mu_t = \bar{\mu}_t + K_t i_t$$

… and compute the corresponding variance:

$$\Sigma_t = (1 - K_t)\bar{\Sigma}_t$$

Towards a Generic Kalman

Let consider a model of the state of a robot having two variables, a and b, components of the vector x :

$$x_{t+1} = Ax_t + Bu_t + w_t$$

$$y_t = Cx_t + v_t$$

Where x_k is the state of the system at sample k,
A is the matrix defining how the system evolves,
u is the known input to the system, impacting x through B,
w_k is the state stochastic perturbation with covariance matrix Q,
y_k the measurement, related to the state by matrix C,
v_k the measurement noise with covariance matrix R.

Towards a Generic Kalman

As an example we could consider a simple system with no external input:

$$x_{t+1} = Ax_t + w_t$$

$$y_t = Cx_t + v_t$$

Where:

$$A = \begin{bmatrix} 1 & -0.9 \\ 1 & 0 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad Q = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \quad R = \begin{bmatrix} 0.1 \end{bmatrix}$$

Towards a Generic Kalman

Let compute the expectation of the **predicted** state, based on the previous mean:

$$\bar{\mu}_t = A\mu_{t-1} + Bu_t$$

… and compute the covariance matrix of the **predicted** state:

$$\bar{\Sigma}_t = A\Sigma_{t-1}A^T + Q$$

Towards a Generic Kalman

If we now take a **measurement** we can have a look to the difference between measure and prediction, called *innovation* (and its variance):

$$i_t = y_t - C\bar{\mu}_t \qquad S_t = C\bar{\Sigma}_t C^T + R$$

… and compute an optimal gain for the correction (not developed here):

$$K_t = \bar{\Sigma}_t C^T S_t^{-1}$$

Towards a Generic Kalman

Based on the optimal gain, we can generate an estimation *a posteriori* that takes into account the **measurement** through the innovation and the gain:

$$\mu_t = \bar{\mu}_t + K_t i_t$$

… and compute the corresponding the covariance matrix:

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

Extended Kalman Filter (EKF)

The Extended Kalman Filter is a Kalman filter applied to a non-linear

system linearized at each discrete time *k*

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

$$\bar{\mu}_t = g(u_t, \mu_{t-1})$$
$$\bar{\Sigma}_t = G_t \, \Sigma_{t-1} \, G_t^T + R_t$$
$$K_t = \bar{\Sigma}_t \, H_t^T (H_t \, \bar{\Sigma}_t \, H_t^T + Q_t)^{-1}$$
$$\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$$
$$\Sigma_t = (I - K_t \, H_t) \, \bar{\Sigma}_t$$
$$\textbf{return } \mu_t, \Sigma_t$$

prediction: computation of mean and covariance using linearization of the motion model *g*

measurement update: computation of final mean and covariance using Kalman gain K_t

g is the motion model : $x_t = g(u_t, x_{t-1}) + \varepsilon_t$ where $\varepsilon_t \sim N(0, R)$ is a multinormal that models the uncertainty introduced by the state transition.

G is the Jacobian of the motion model *g*.

Extended Kalman Filter (EKF)

The Extended Kalman Filter is a Kalman filter applied to a non-linear

system linearized at each discrete time *k*

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

$$\bar{\mu}_t = g(u_t, \mu_{t-1})$$
$$\bar{\Sigma}_t = G_t \, \Sigma_{t-1} \, G_t^T + R_t$$
$$K_t = \bar{\Sigma}_t \, H_t^T (H_t \, \bar{\Sigma}_t \, H_t^T + Q_t)^{-1}$$
$$\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$$
$$\Sigma_t = (I - K_t \, H_t) \, \bar{\Sigma}_t$$
$$\textbf{return } \mu_t, \Sigma_t$$

prediction: computation of mean and covariance using linearization of the motion model g

measurement update: computation of final mean and covariance using Kalman gain K_t

h is the measurement model: $z_t = h(x_t) + \delta_t$ where $\delta_t \sim N(0, Q)$ is a multinormal describing the measurement noise.

H is the Jacobian of the measurement model *h*.

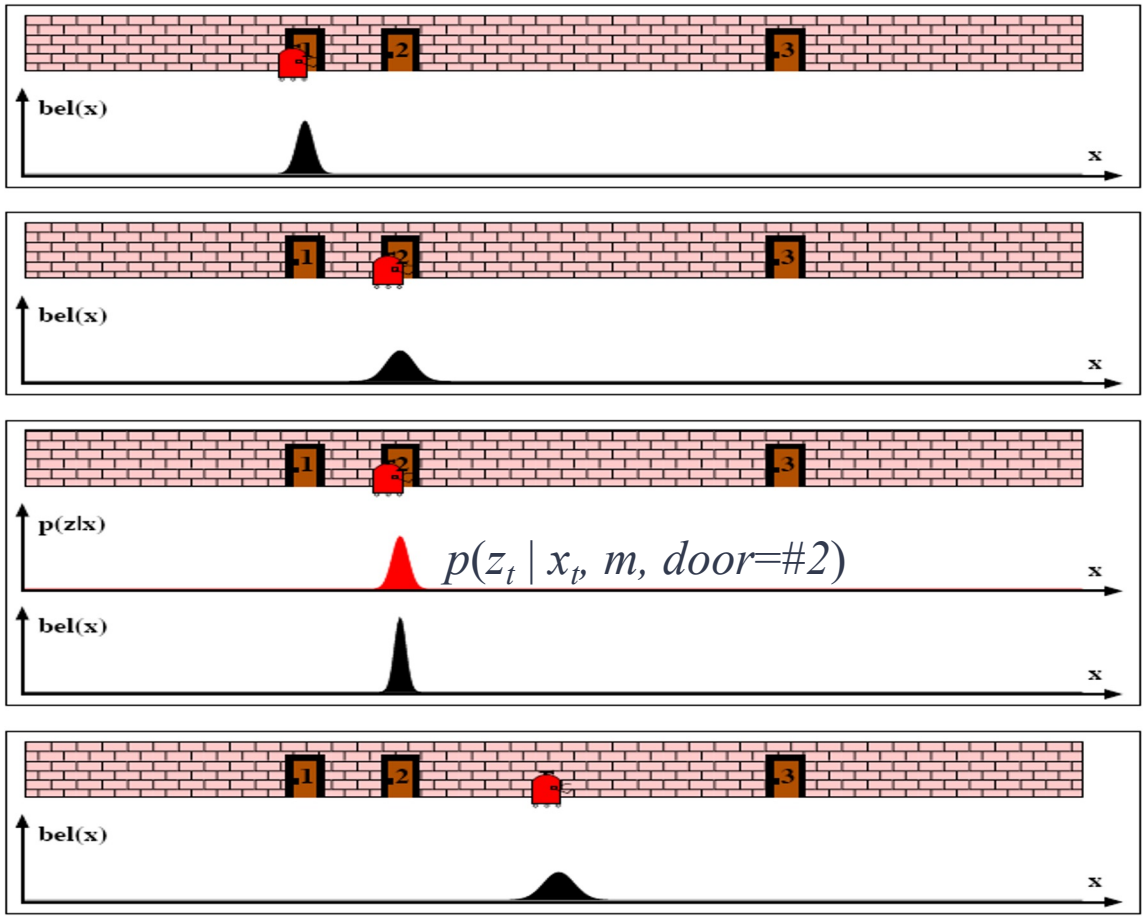
K is the *Kalman gain*. It specifies the degree to which the measurement is incorporated into the new state estimate. It is computed so to minimize the a posteriori error covariance.

1D EKF Example

Initial belief.

Initial belief is convolved with the motion model (prediction step).

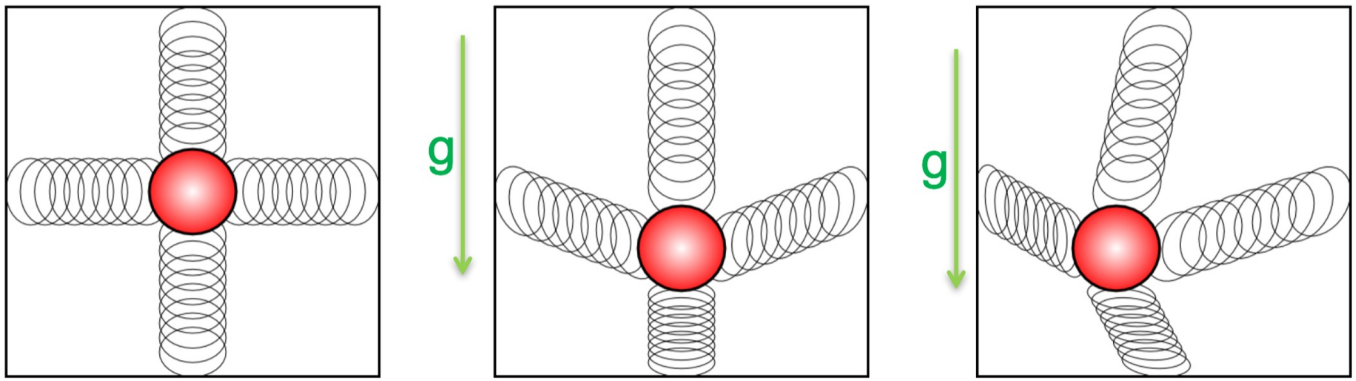
Here we assume that the robot can identify the feature it sees as door #2 (known correspondence).
Resulting belief is tighter than the variances of both the robot's previous belief and the observation density



Concrete example sensor fusion (from Adrien Briod)

Accelerometers

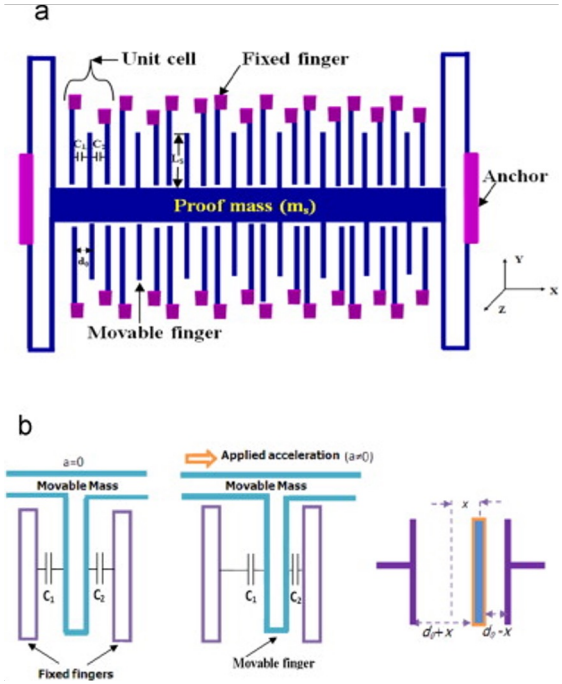
- Measure linear accelerations (including g!)



Free fall

Gravity

Gravity & linear accelerations



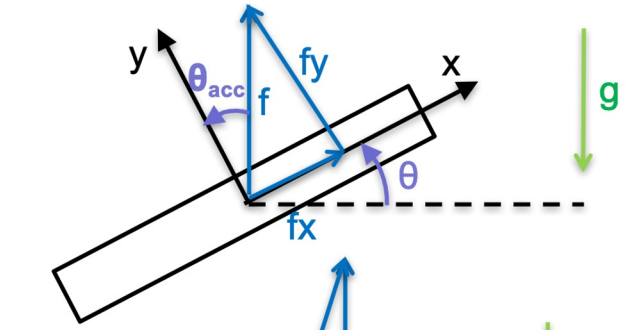
S. Kavitha, R. Joseph Daniel, K. Sumangala, "High performance MEMS accelerometers for concrete SHM applications and comparison with COTS accelerometers", Mechanical Systems and Signal Processing, Volumes 66–67, 2016, Pages 410-424, ISSN 0888-3270, https://doi.org/10.1016/j.ymssp.2015.06.005.

Concrete example sensor fusion (from Adrien Briod)

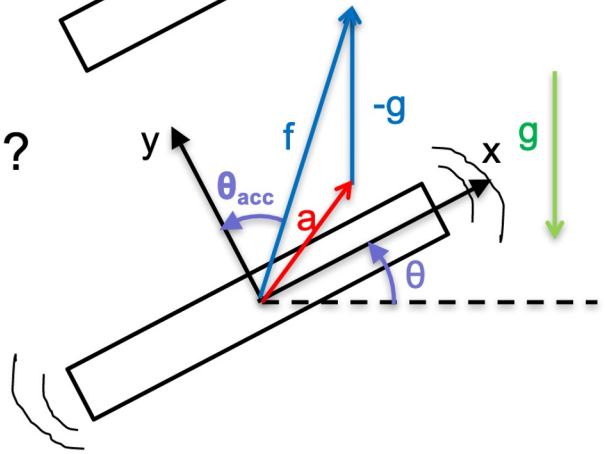
Orientation from accelerometers

- At constant speed, the accelerometers only measure gravity **g**
  → Measurement **f** = **-g**

Θ_{acc} = atan2(**fx**, **fy**)

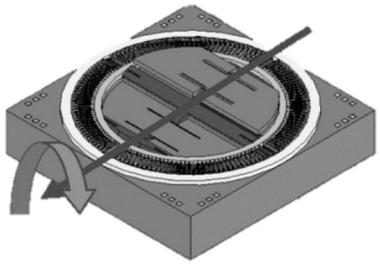


- What if additional accelerations **a** ?
  → Measurement **f** = **-g** + **a**
  → Θ_{acc} is perturbed by **a**

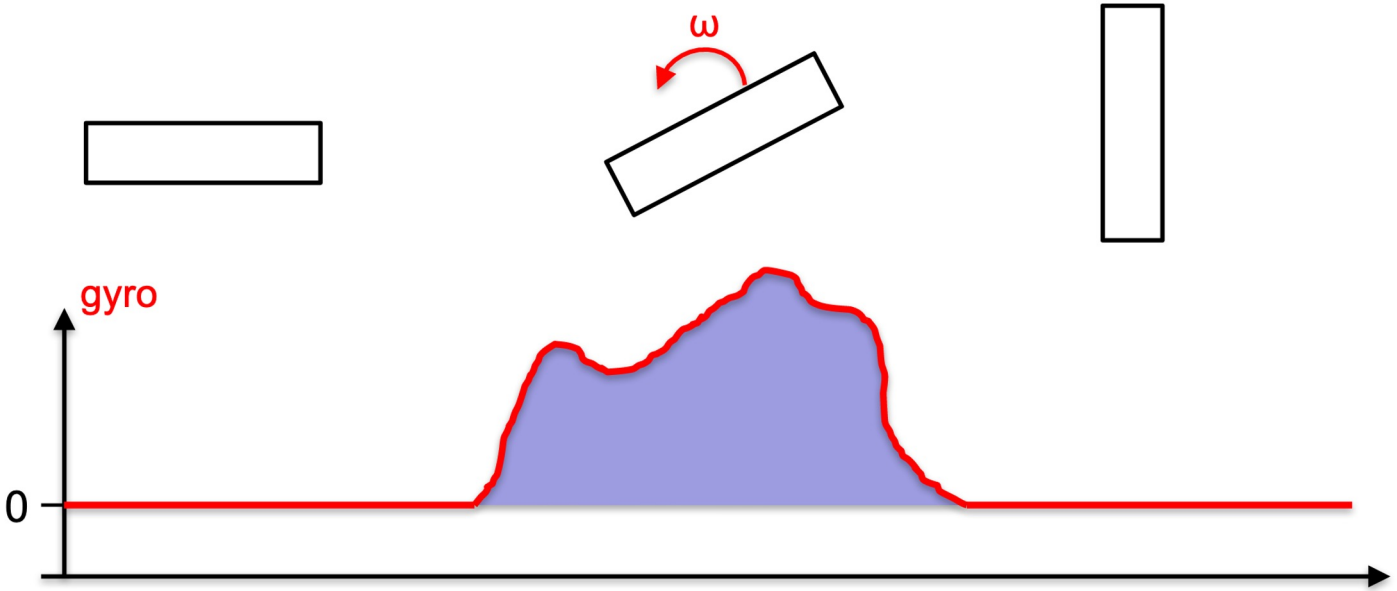


Concrete example sensor fusion (from Adrien Briod)

Rate gyroscopes



- Measure rotation speed (or angular velocity)
  - Pure self-motion. No absolute information.



Concrete example sensor fusion (from Adrien Briod)

Orientation from gyroscopes

- Integration of angular speed (ω)

$$\Theta = \int \omega \, dt$$

→ Θ can be computed incrementally as follows :

$$\Theta_k = \Theta_{k-1} + \omega \, \Delta t$$

- In reality, ω is noisy :

$$\omega = \omega_{true} + r$$

r: gyroscope noise

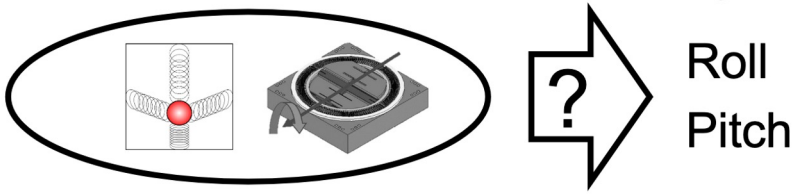
→ The angle error is increasing over time
(e.g: if constant error r=1˚/s → angle error is 60˚ after 1 minute)

Concrete example sensor fusion (from Adrien Briod)

The problem

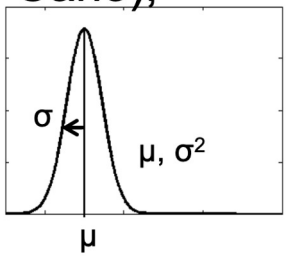
- Accelerometers or rate gyroscopes alone don't provide an accurate orientation

  → How to fuse both sensors to obtain a good estimation ?



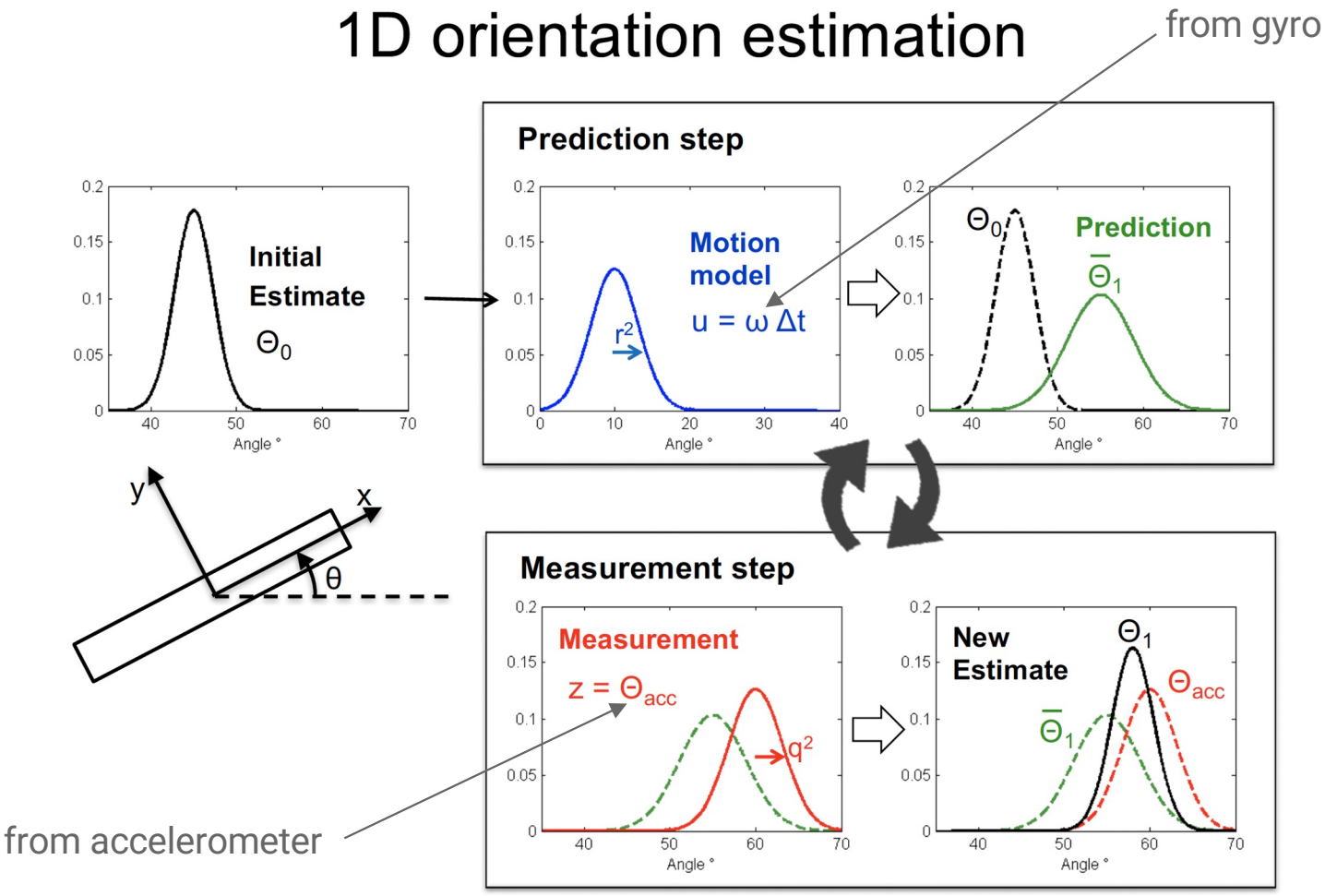
- Answer: Sensor Fusion algorithm
  - Bayes, Histogram (Grid), Particle (Monte-Carlo), Kalman,… which one ?

→ An Extended Kalman filter



- Unimodal probability distribution (Gaussian)
  - → Advantage: Computationally efficient
  - → Drawback: Needs correct initialization, can make only one hypothesis, can diverge

Concrete example sensor fusion (from Adrien Briod)



What should I remember

- Kalman filter
  - basic principle, advantages and disadvantages in respect to other methods
  - computation for simple and more complex approaches
  - adjustment of parameters
    - variances (co-variance matrix) for sensors and actuators
  - Applications

Looking for a semester project?

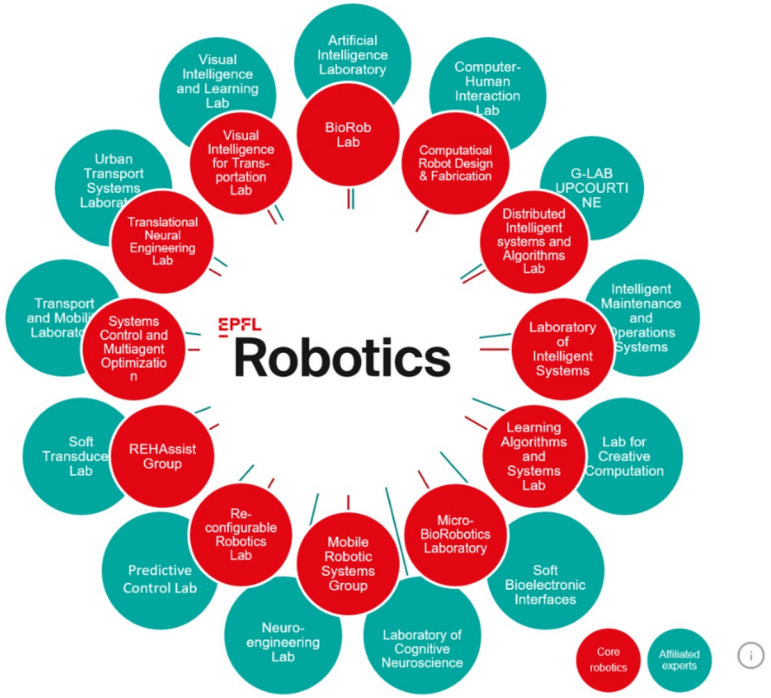
EPFL About Education Research Innovation Schools Campus EN

/ Research / Research Domains / Robotics at EPFL / Our Labs

- EPFL Robotics News
- Our Expertise
- **Our Labs**
  - Core Robotics
  - Affiliated Expertise
- Virtual Lab Tour
- Our Spin-offs
- Centers
- Study Robotics
- Women in Robotics
- Innovation Booster Robotics
- Swiss Robotics Day
- Swiss Robotics Association
- Contact Us & Visit

Our Labs

A showcase of cutting-edge research and development where EPFL's robotics labs drive the future of technology through groundbreaking projects.



Looking for a semester project?