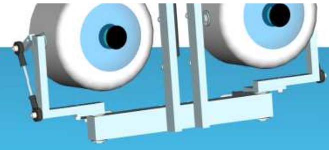


6. Online Estimation: The Kalman Filter

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Autonomous Systems Lab
ETH Zentrum
Tannenstrasse 3, CLA
8092 Zürich, Switzerland

The Overall Bayes Filter (Rep.)

$$\boxed{\mathrm{Bel}(x_t)} = p(x_t \mid u_1, z_1, \ldots, u_t, z_t)$$

(Bayes) $$= \eta \; p(z_t \mid x_t, u_1, z_1, \ldots, u_t) p(x_t \mid u_1, z_1, \ldots, u_t)$$

(Markov) $$= \eta \; p(z_t \mid x_t) p(x_t \mid u_1, z_1, \ldots, u_t)$$

(Tot. prob.) $$= \eta \; p(z_t \mid x_t) \int p(x_t \mid u_1, z_1, \ldots, u_t, x_{t-1}) \, p(x_{t-1} \mid u_1, z_1, \ldots, u_t) dx_{t-1}$$

(Markov) $$= \eta \; p(z_t \mid x_t) \int p(x_t \mid u_t, x_{t-1}) p(x_{t-1} \mid u_1, z_1, \ldots, u_t) dx_{t-1}$$

(Markov) $$= \eta \; p(z_t \mid x_t) \int p(x_t \mid u_t, x_{t-1}) p(x_{t-1} \mid u_1, z_1, \ldots, z_{t-1}) dx_{t-1}$$

$$\boxed{= \eta \; p(z_t \mid x_t) \int p(x_t \mid u_t, x_{t-1}) \mathrm{Bel}(x_{t-1}) dx_{t-1}}$$

Rudolf Kalman receives National Medal of Science on Oct 7 2009
(see http://www.ethlife.ethz.ch/archive_articles/091008_kalman_per/index_EN)

The Kalman Filter - Principle

1. The state at time t depends linearly on the previous state and on the current action (motion):

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

Previous state | Current action | Gaussian noise term

2. The sensor measurement depends linearly on the current state:

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

Current state | Gaussian noise term

Note: Notation is slightly different than in Bishop.

The Kalman Filter - Principle

3. The initial belief is a Gaussian:

$$\mathrm{Bel}(\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_0; \boldsymbol{\mu}_0, V_0)$$

Initial state

Initial state covariance

Using all these, we have:

$$p(\mathbf{x}_t \mid \mathbf{u}_t, \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; A\mathbf{x}_{t-1} + B\mathbf{u}_t, \Gamma)$$

$$p(\mathbf{z}_t \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{z}_t; C\mathbf{x}_t, \Sigma)$$

Sensor noise

Action uncertainty

The Kalman Filter Algorithm

Algorithm *Kalman_filter* $(\boldsymbol{\mu}_{t-1}, V_{t-1}, \mathbf{u}_t, \mathbf{z}_t)$:

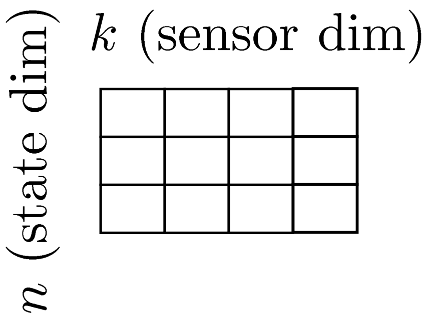
1. $\bar{\boldsymbol{\mu}}_{t-1} \leftarrow A\boldsymbol{\mu}_{t-1} + B\mathbf{u}_t$
2. $P_{t-1} \leftarrow AV_{t-1}A^T + \Gamma$
3. $K_t \leftarrow P_{t-1}C^T(CP_{t-1}C^T + \Sigma)^{-1}$
4. $\boldsymbol{\mu}_t \leftarrow \bar{\boldsymbol{\mu}}_{t-1} + K_t(\underline{\mathbf{z}_t - C\bar{\boldsymbol{\mu}}_{t-1}})$
5. $V_t \leftarrow (I - K_tC)P_{t-1}$
6. return $(\boldsymbol{\mu}_t, V_t)$

Prediction Step

“Kalman gain”

Correction Step

“Innovation“ :
difference between expected and measured sensor input

$K_t =$ 

Properties of the Kalman Filter

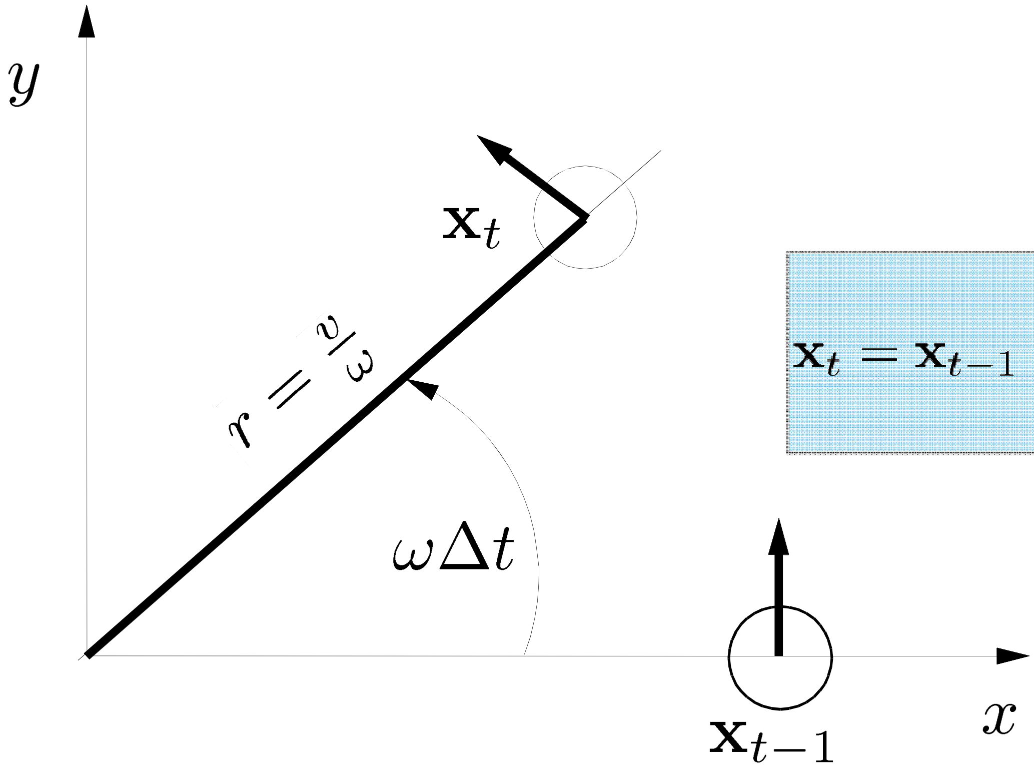
Complexity:

- Involves matrix inversion in line 3:
  $\Rightarrow$ complexity is $O(k^{2.4})$, i.e. approximately cubical
- Does a matrix multiplication in line 5:
  $\Rightarrow$ complexity is at least $O(n^2)$ if $K_t C$ is sparse
- In many applications (e.g. mapping) n is much bigger than k

Linearity:

- The Kalman filter requires a linear mapping from x_{t-1} to x_t and from x_t to z_t
- This is often not correct, e.g. for robots that move and rotate

Problem with the Kalman Filter



Robot state: $\mathbf{x} = (x, y, \theta)$

Control input: $\mathbf{u} = (v, \omega)$

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \begin{pmatrix} -\frac{v}{\omega}\sin(\theta) + \frac{v}{\omega}\sin(\theta + \omega\Delta t) \\ \frac{v}{\omega}\cos(\theta) - \frac{v}{\omega}\cos(\theta + \omega\Delta t) \\ \omega\Delta t \end{pmatrix}$$

The mapping from control inputs to the new state is non-linear!

Variants of the Kalman Filter

To address the non-linearity problem other approaches exist:

- Extended Kalman Filter (linearization using Taylor exp.):
  - State transition: $\mathbf{x}_t = g(\mathbf{x}_{t-1}, \mathbf{u}_t) + \epsilon$
  - Sensor model: $\mathbf{z}_t = h(\mathbf{x}_t) + \delta$

g, h: Non-linear functions

- Unscented Kalman Filter:
  - Extracts sigma-points from the Gaussian
  - Performs linear regression on the sigma-points
- ...

A more efficient version of the Kalman filter provides:

- Information Filter
- Extended Information Filter

The Extended Kalman Filter Algorithm

Algorithm *Extended_Kalman_filter* $(\boldsymbol{\mu}_{t-1}, V_{t-1}, \mathbf{u}_t, \mathbf{z}_t)$

1. $\bar{\boldsymbol{\mu}}_{t-1} \leftarrow g(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t)$
2. $P_{t-1} \leftarrow G V_{t-1} G^T + \Gamma$
3. $K_t \leftarrow P_{t-1} H^T (H P_{t-1} H^T + \Sigma)^{-1}$
4. $\boldsymbol{\mu}_t \leftarrow \bar{\boldsymbol{\mu}}_{t-1} + K_t(\mathbf{z}_t - h(\bar{\boldsymbol{\mu}}_{t-1}))$
5. $V_t \leftarrow (I - K_t H) P_{t-1}$
6. return $(\boldsymbol{\mu}_t, V_t)$

G: Jacobian of g

"Kalman gain"

H: Jacobian of h

This is the same as before, only with the Jacobians G and H (first derivatives) instead of the transition matrices A, B, and C

The Unscented Kalman Filter

There are other ways to do the linearization:

- The Unscented Kalman Filter uses the unscented transform
- It extracts sigma points from the Gaussian belief and passes them through g and h
- The sigma-points are located at the mean and symetrically along the main axes of the covariance matrix
- The state estimate of UKF is more accurate than that of EKF
- The complexity of UKF is the same as for EKF, but in practice it is slightly slower (constant overhead)
- Advantage of UKF: no derivative needs to be computed

Algorithm *Information_filter* $(\boldsymbol{\xi}_{t-1}, \Omega_{t-1}, \mathbf{u}_t, \mathbf{z}_t)$

1. $\bar{\Omega}_t \leftarrow (A\Omega_{t-1}^{-1}A^T + \Gamma)^{-1}$
2. $\bar{\boldsymbol{\xi}}_t \leftarrow \bar{\Omega}_t(A\Omega_{t-1}^{-1}\boldsymbol{\xi}_{t-1} + B\mathbf{u}_t)$
3. $\Omega_t \leftarrow C^T\Gamma C + \bar{\Omega}_t$
4. $\boldsymbol{\xi}_t \leftarrow C^T\Gamma^{-1}\mathbf{z}_t + \bar{\boldsymbol{\xi}}_t$
5. return $(\boldsymbol{\xi}_t, \Omega_t)$

$\Omega = V^{-1}$: Inverse Covariance
„Information Matrix“

$\boldsymbol{\xi} = V^{-1}\boldsymbol{\mu}$

- Representation is done using the canonical parameterization

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \Sigma) = \eta e^{-\frac{1}{2}\mathbf{x}^T\Omega\mathbf{x} - \mathbf{x}^T\boldsymbol{\xi}}$$

- Again, linearity is assumed for the state transition and the sensor model

Algorithm *Extended_Information_filter* $(\boldsymbol{\xi}_{t-1}, \Omega_{t-1}, \mathbf{u}_t, \mathbf{z}_t)$

1. $\boldsymbol{\mu}_{t-1} \leftarrow \Omega_{t-1}^{-1}\boldsymbol{\xi}_{t-1}$
2. $\bar{\Omega}_t \leftarrow (G\Omega_{t-1}^{-1}G^T + \Gamma)^{-1}$
3. $\bar{\boldsymbol{\xi}}_t \leftarrow \bar{\Omega}_t g(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t)$
4. $\bar{\boldsymbol{\mu}}_t \leftarrow g(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t)$
5. $\bar{\Omega}_t \leftarrow H^T\Gamma^{-1}H + \bar{\Omega}_t$
6. $\boldsymbol{\xi}_t \leftarrow H^T\Gamma^{-1}(\mathbf{z}_t - h(\bar{\boldsymbol{\mu}}_t) + H\bar{\boldsymbol{\mu}}_t) + \bar{\boldsymbol{\xi}}_t$
7. return $(\boldsymbol{\xi}_t, \Omega_t)$

Discussion

Kalman Filter:

- Complexity $O(k^{2.4})$ or $O(n^2)$

Information Filter:

- Complexity

6a. Application of KF to Localization and SLAM

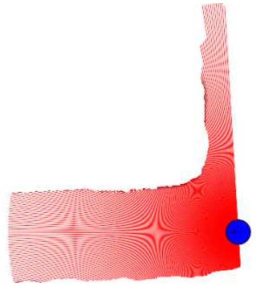
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Autonomous Systems Lab
ETH Zentrum
Tannenstrasse 3, CLA
8092 Zürich, Switzerland

The Localization Problem

Given:

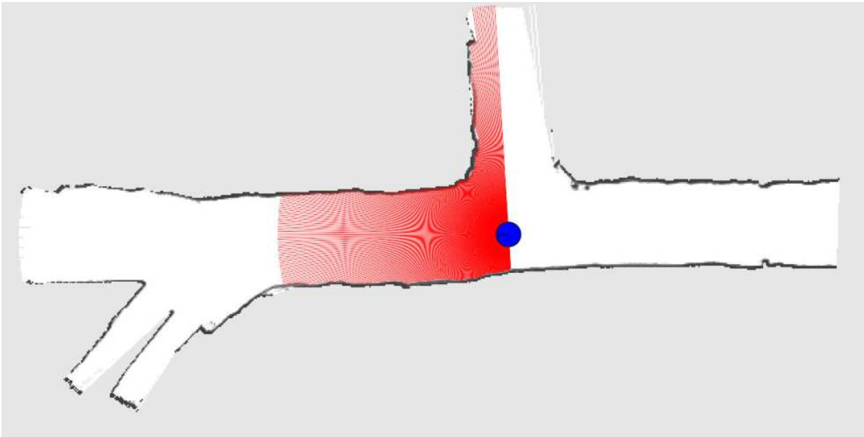
A map of the environment



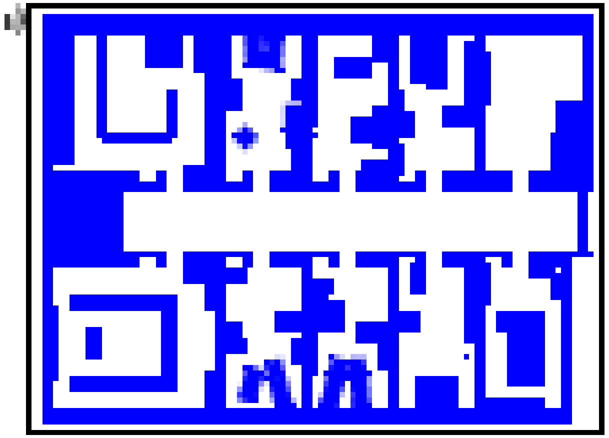
Sensor measurements

Wanted:

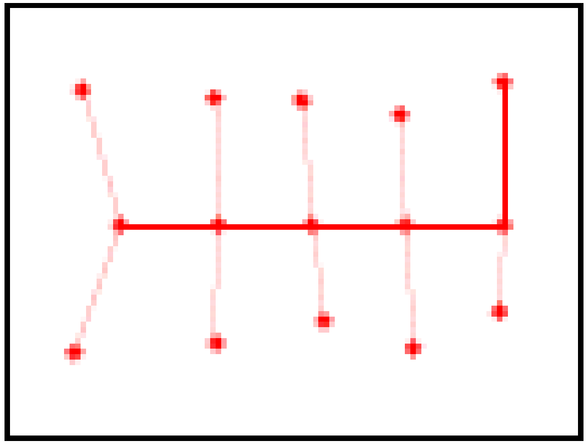
- Global coordinates of the robot (position)



Possible Map Representations



Manually constructed



Topological map

Occupancy grid map

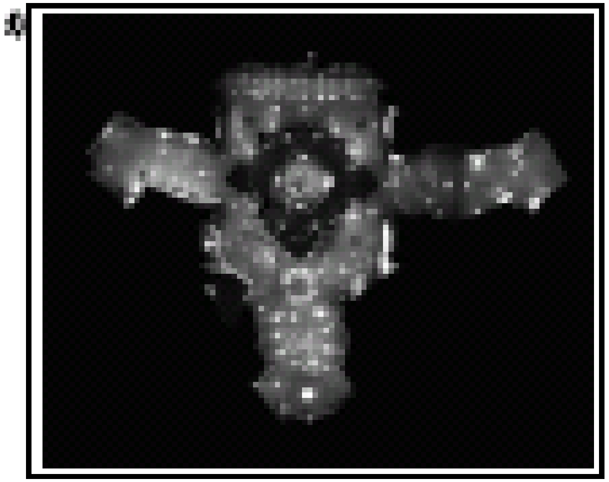


Image mosaic of a ceiling

Taxonomy of the Localization Problem

- Global vs local Localization:
  Is the initial robot pose known?
- Static vs Dynamic Environment:
  Are there moving objects in the environment?
- Passive vs Active Localization:
  Is the robot actively controlled by the localization?
- Single vs Multi-Robot Localization:
  How many robots are involved in the localization prozess?

Online Localization

- The localization problem needs to be solved in each time step. It would be inefficient to perform a full search on the whole state space everytime.

Idea: Use the information of the previous time step to estimate the current position.

- To obtain more information we also include the odometry measurements after traveling between time steps.

Localization is motion estimation (prediction) and measurement update (correction).

Bayes Filter Principle

- We can consider the motion as the robot's action

Localization can be implemented with a Bayes Filter:

- The belief is the probability of being at the current position x_t
- The action update computes a new belief after each robot motion.
- The sensor update computes a new belief after each sensor measurement.

Markov Localization

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

1. $\eta = 0$
2. for all x_t do
3. $\overline{\mathrm{Bel}}(x_t) \leftarrow p(z_t \mid x_t, m)\mathrm{Bel}(x_{t-1})$
4. $\eta \leftarrow \eta + \mathrm{Bel}(x_t)$
5. $\mathrm{Bel}(x_t) \leftarrow \int p(x_t \mid u_t, x_{t-1}, m)\overline{\mathrm{Bel}}(x_t)dx$
6. for all x_t do

   $\mathrm{Bel}(x_t) \leftarrow \eta^{-1}\mathrm{Bel}(x_t)$
7. return $\mathrm{Bel}(x_t)$

Global vs Local Localization

The question remains how to define the initial belief.

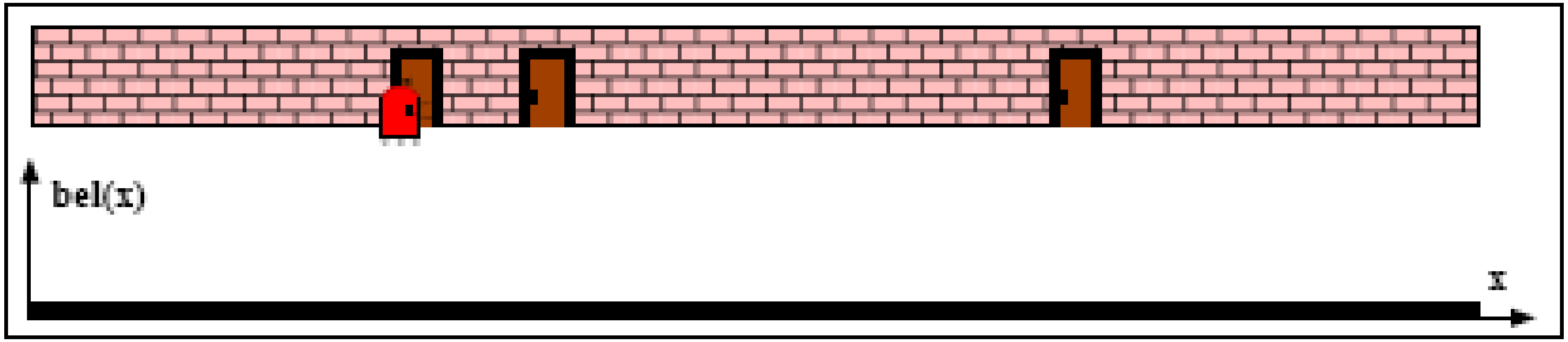
For local localization we are given an (exact) first position $\bar{x}_0$. The initial belief is then

$$\text{Bel}(x_0) = \begin{cases} 1 & \text{if } x_0 = \bar{x}_0 \\ 0 & \text{otherwise} \end{cases} \quad \text{or} \quad \text{Bel}(x_0) = \mathcal{N}(x_0; \bar{x}_o, \Sigma)$$

For global localization we use a uniform distribution:

$$\text{Bel}(x_0) = \frac{1}{|X|}$$

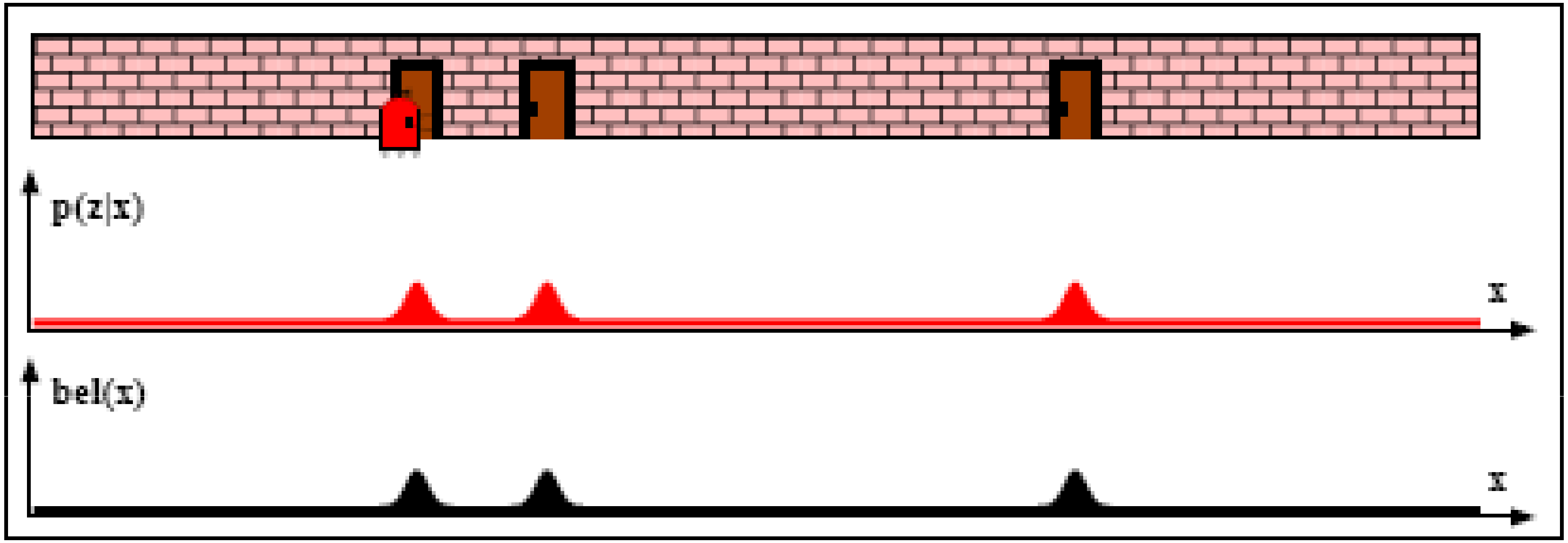
A Simple Example



Assume we have a

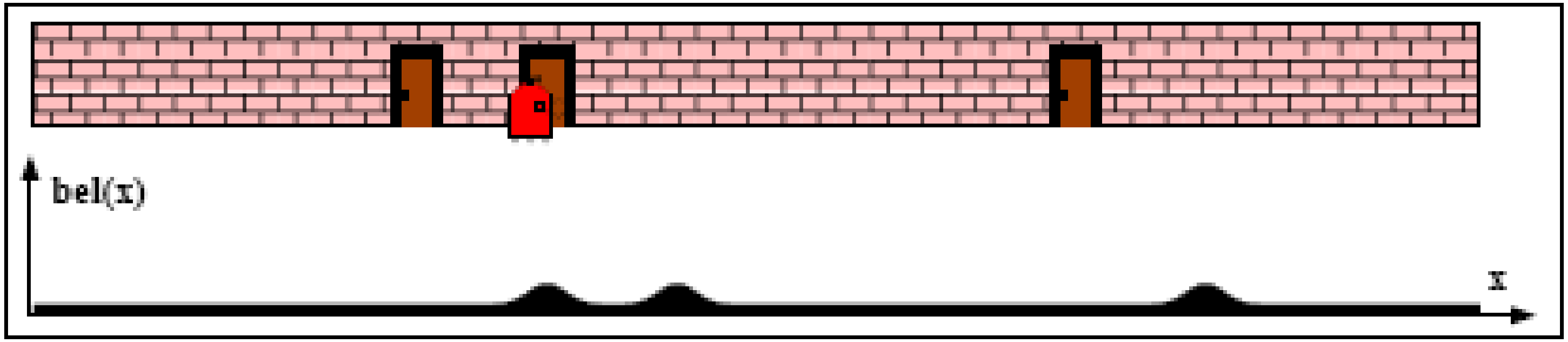
- One-dimensional environment (corridor)
- Sensor for detecting doors
- Uniform initial distribution for the belief (global localization)

A Simple Example



After the first sensor measurement we multiply the old belief $\text{Bel}(x_{t-1})$ with the sensor model $p(z \mid x)$ and obtain a new belief $\text{Bel}(x_t)$.

A Simple Example

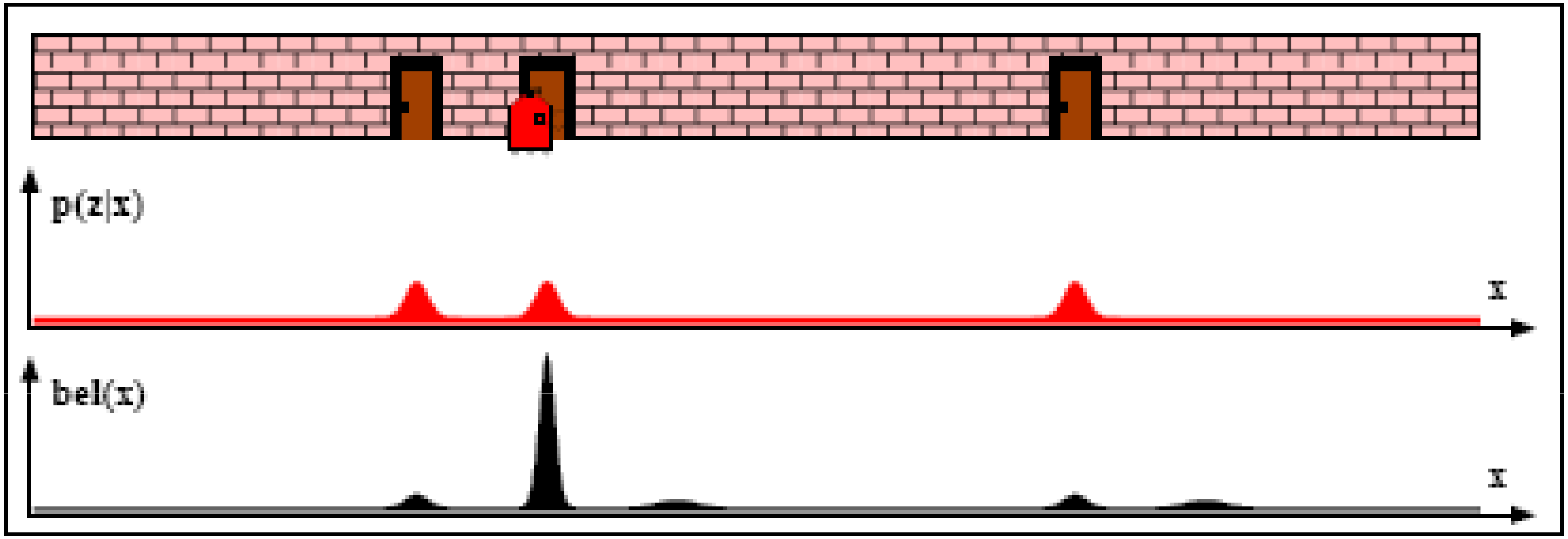


Now the robot moves. We use the motion model $p(x_t \mid u_t, x_{t-1}, m)$ to compute for all x_t:

$$\mathrm{Bel}(x_t) \leftarrow \int p(x_t \mid u_t, x_{t-1}, m)\mathrm{Bel}(x_{t-1})dx$$

This corresponds to a convolution of the motion model with the old belief. The new belief is smoother than the old one.

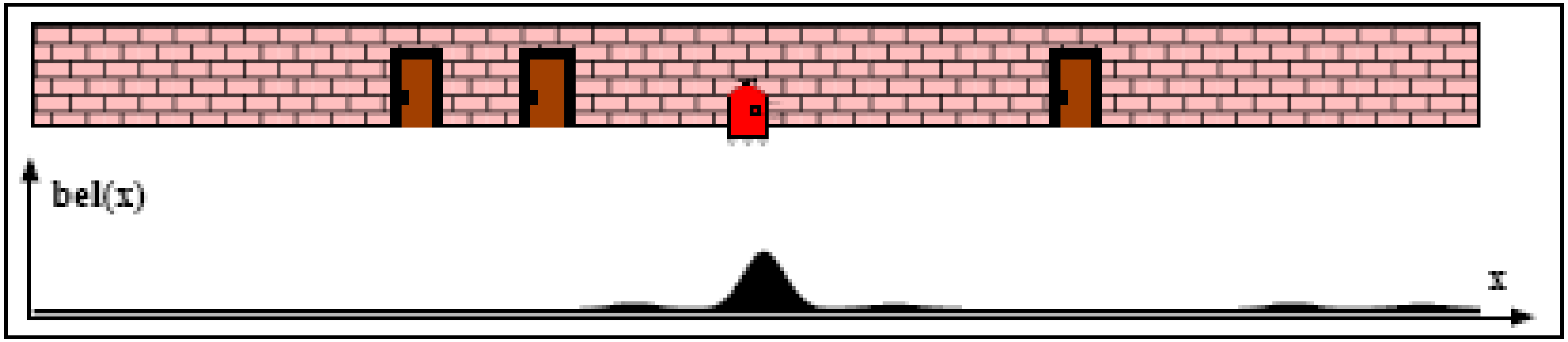
A Simple Example



Next sensor measurement: again we multiply the old belief with the sensor model.

The new belief has one big peak: The robot is localized.

A Simple Example



The robot moves again. After convolution of the motion model with the old belief the localization has more uncertainty, but there is only one peak.

The robot stays localized.

Implementations

Markov localization can be implemented in different ways:

- Kalman filter (EKF, UKF, etc.)
- Discrete filter (histogram)
- Particle filter
- ...

Mapping

World map of 1665

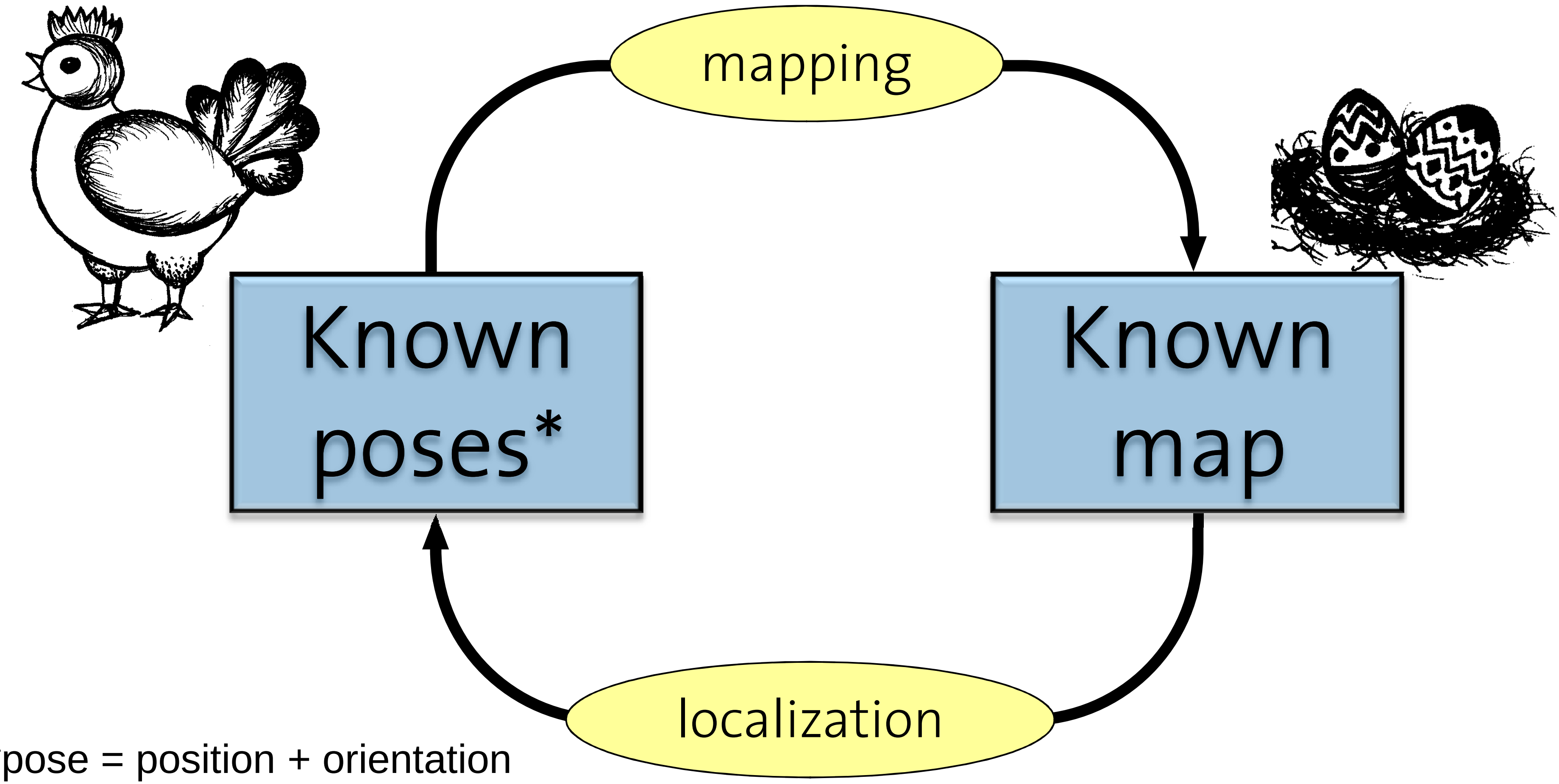
Modern world map

- For a good map we need a good localization
- To localize we need a map

Simultaneous Localization and Mapping

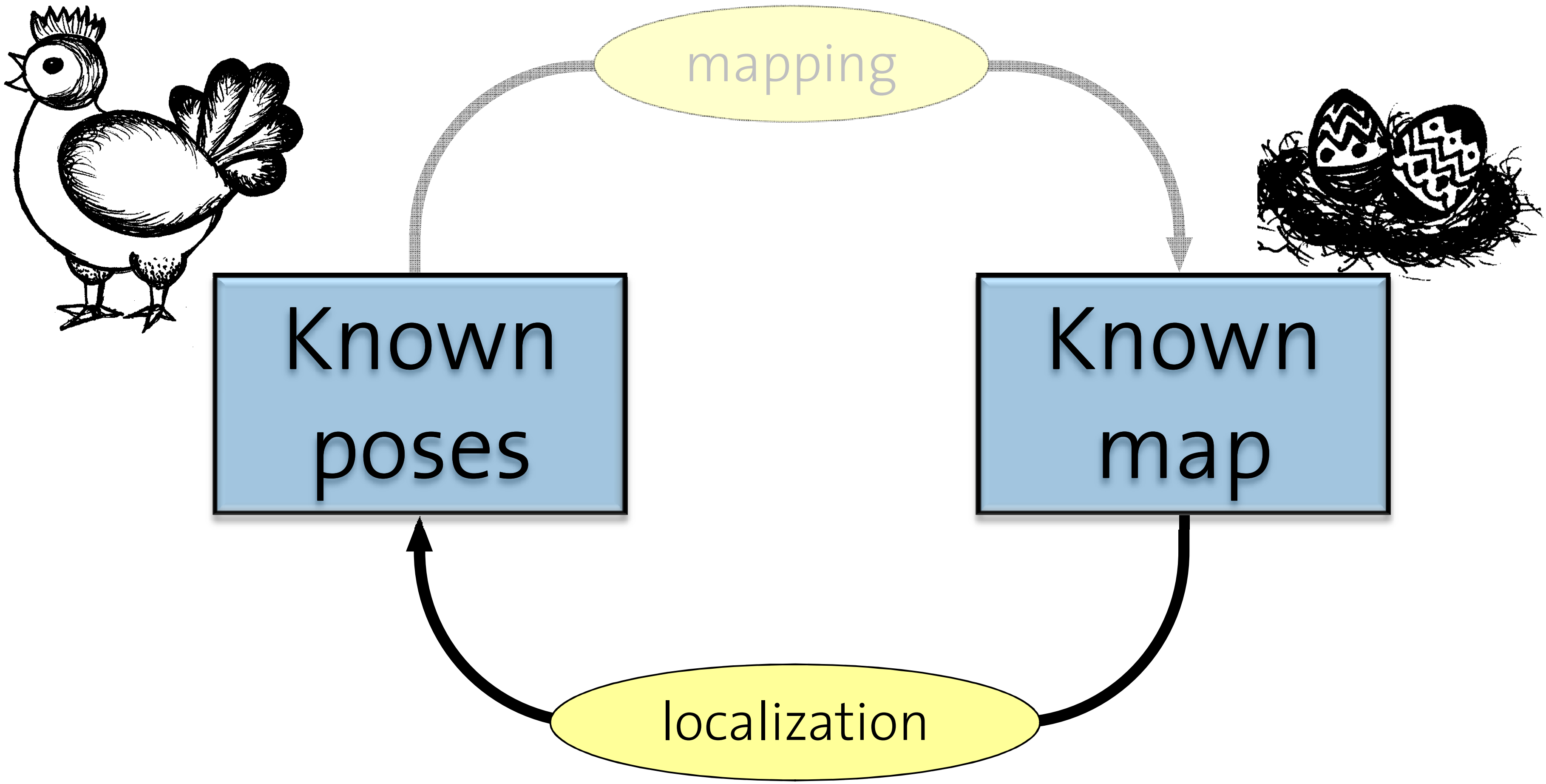
The SLAM Problem

“Chicken-and-Egg-Problem”



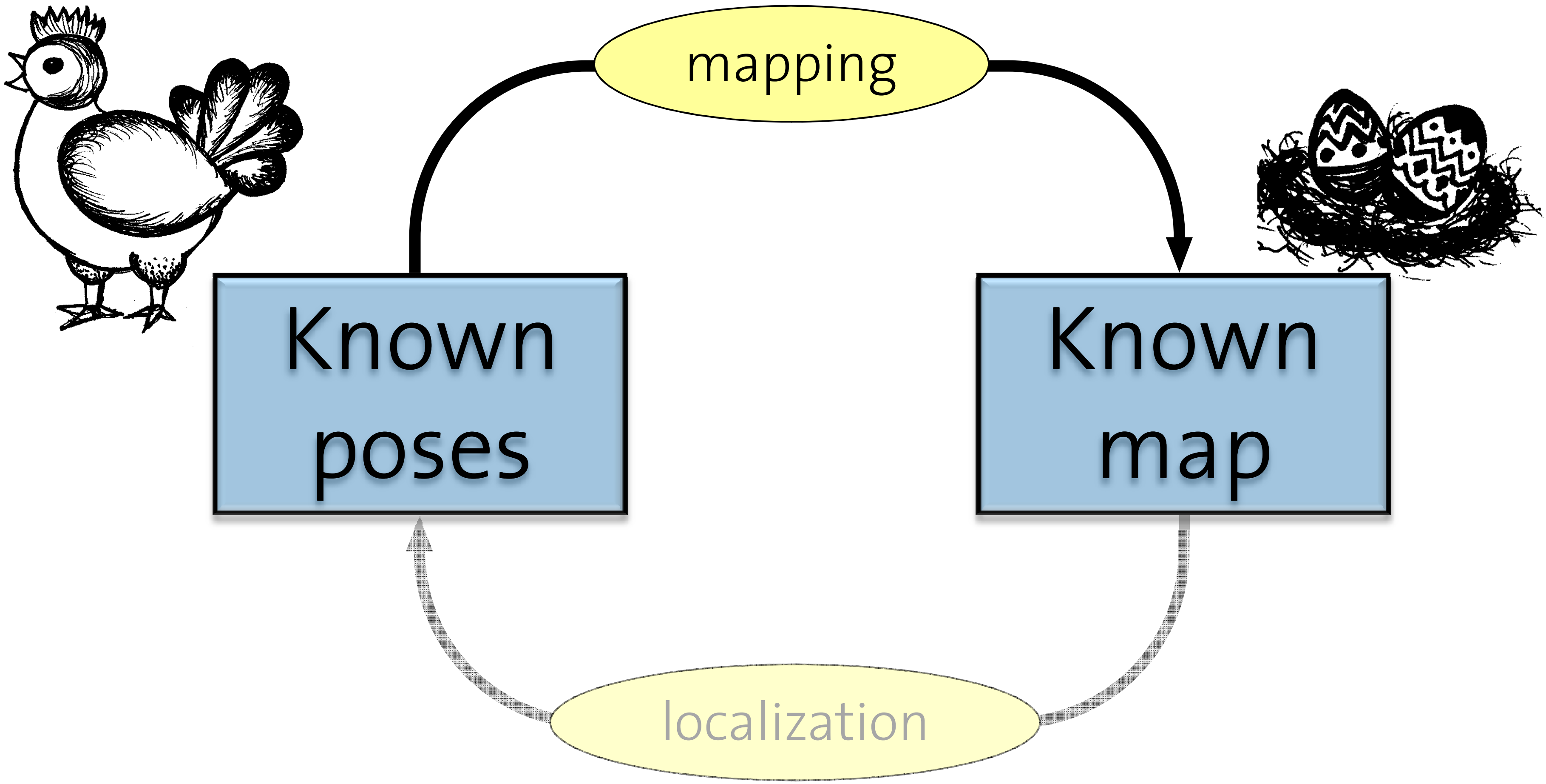
*pose = position + orientation

Localization given a map



The SLAM Problem

Mapping with known poses



Challenges in Mapping

- Size:

  Big environments (wrt. the range of the sensors) are more difficult to map.

- Noise in the sensors and actuators:

  If the sensor is very noisy, mapping is difficult.

- Perceptual ambiguity:

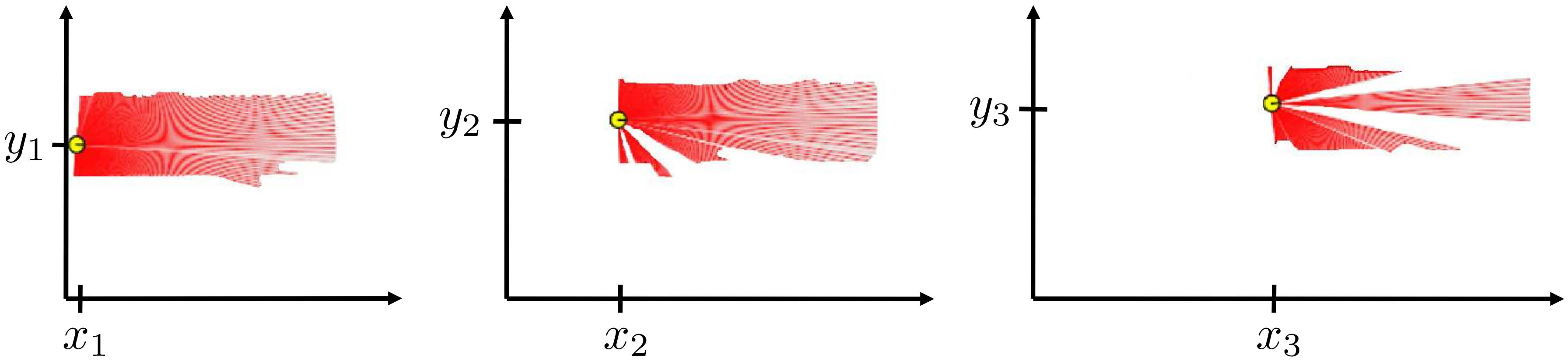
  Self-similar environments are difficult to map.

- Cycles:

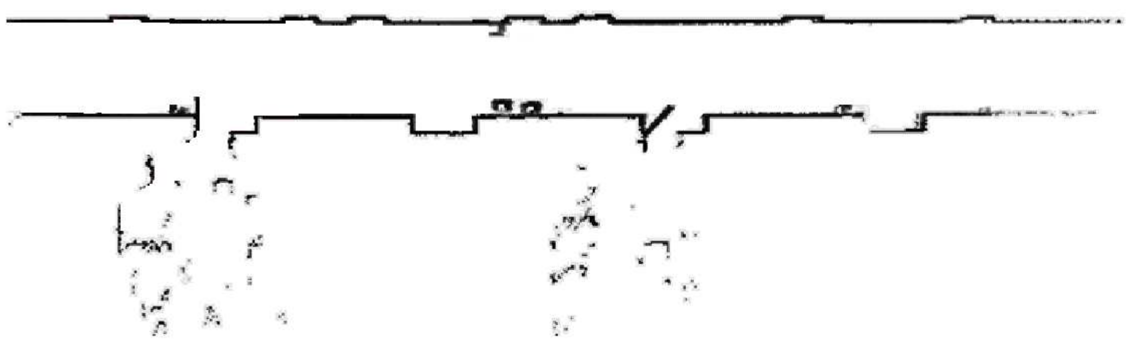
  If the robot returns to its first position from another path, it accumulates its error.

Mapping with Known Poses

Given: Measurements and robot poses



Wanted: map of the environment



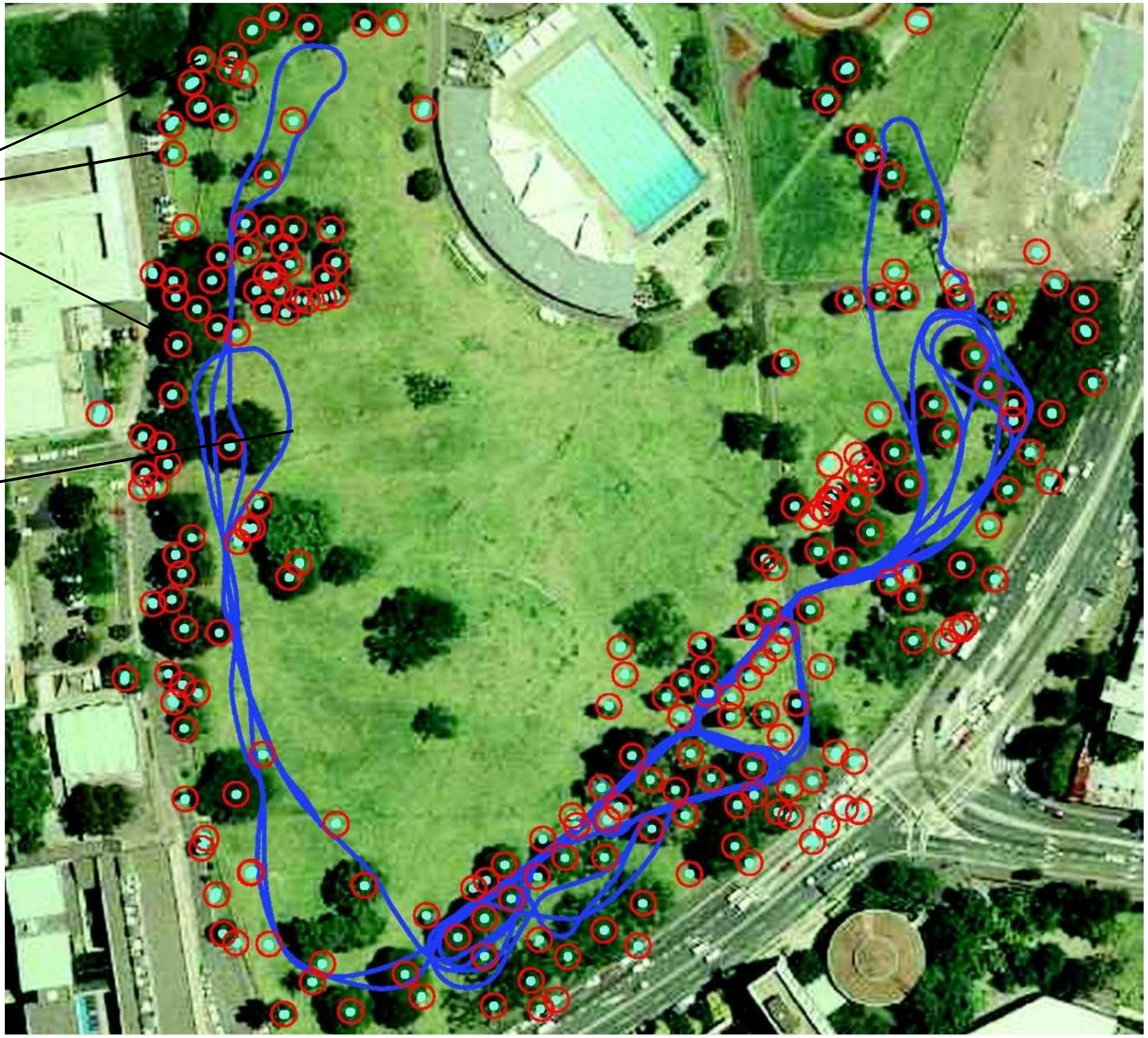
Different Kinds of Maps

- Feature-based maps:
  store features ("landmarks") of the environment (e.g. lines, corners, circles, etc.)
- Occupancy grid maps:
  store at each xy-position the probability that the corresponding cell is occupied
- Topological maps
  store intersections (nodes) and connections (edges) in a graph structure, no geometric information (e.g. distances)
- …

Example: Feature-based Map

trees

vehicle trajectory

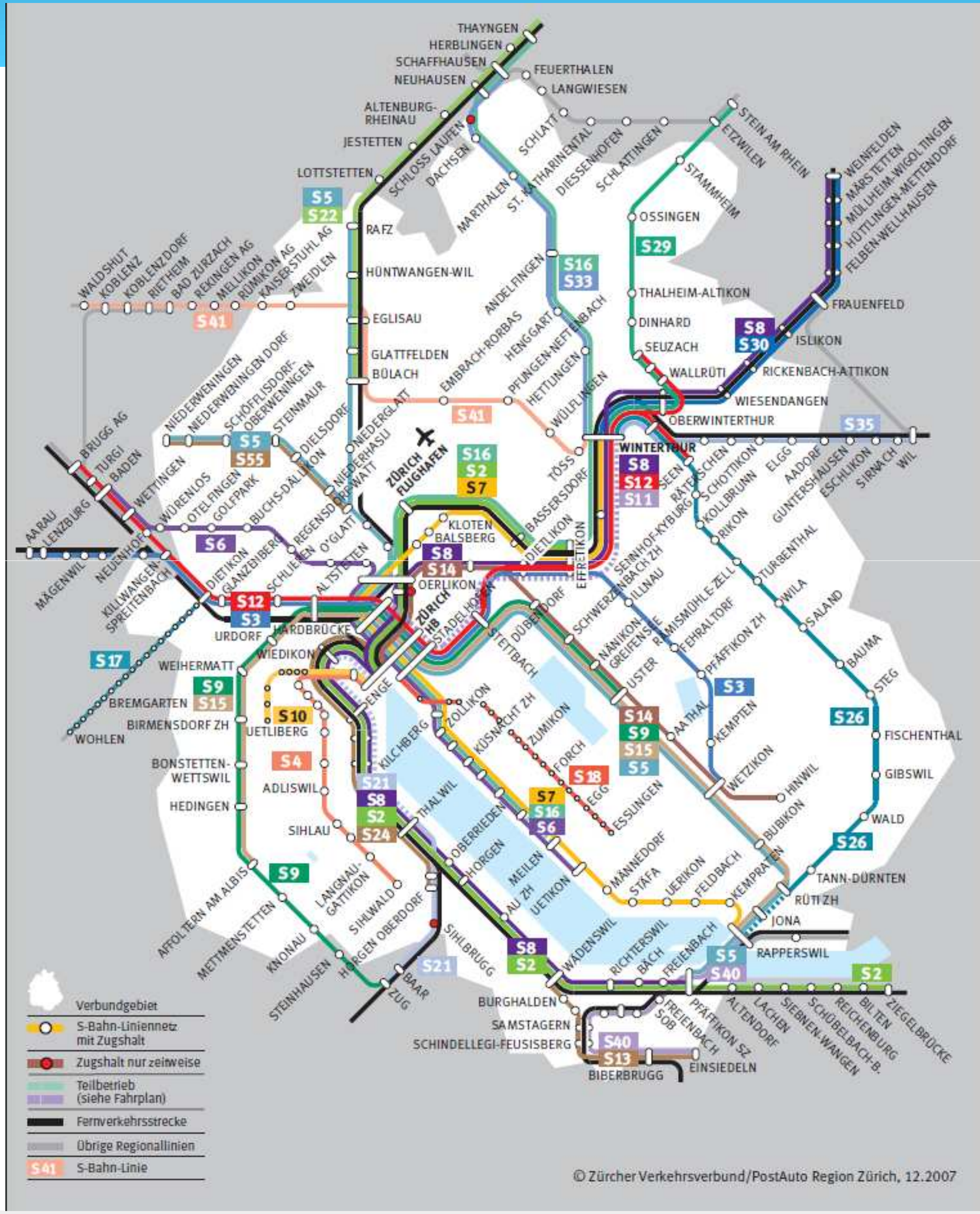


"Victoria Park"

[courtesy by E. Nebot]

Example: Occupancy Grid Map

Example: Topological Map



Occupancy Grid Maps

- Introduced by Moravec and Elfes in 1985
- Represent environment by a grid.
- Estimate the probability that a location is occupied by an obstacle.
- Key assumptions:
  - Occupancy of individual cells (m_{xy}) is independent of neighboring cells
  - Robot positions are known!

Mathematical Formulation

We formulate the mapping problem as a maximum likelihood estimation.

Given:

- Sensor measurements up to time t: $z_{1:t}$
- Robot positions at each time step: $x_{1:t}$

Wanted:

- Map cells $\mathbf{m} = (m_1, \ldots, m_N)$

$$\mathbf{m}^* = \arg\max_{\mathbf{m}} p(\mathbf{m} \mid z_{1:t}, x_{1:t})$$

- Independence assumption:

$$p(\mathbf{m} \mid z_{1:t}, x_{1:t}) = \prod_i p(m_i \mid z_{1:t}, x_{1:t})$$

Binary Bayes Filter

- Each map cell m_i has two possible states: occupied (= 1) or free (= 0).

$$\mathrm{Bel}_t(m_i) := p(m_i \mid z_{1:t}, x_{1:t}) = 1 - p(\neg m_i \mid z_{1:t}, x_{1:t})$$

- The world is static. This means that the map cells are independent of time.
- We use a binary static-state Bayes filter (no action update):

$$\begin{aligned} \mathrm{Bel}_t(m_i) &= \eta\, p(z_t \mid m_i) p(m_i \mid z_{1:t-1}, x_{1:t-1}) \\ &= \eta\, p(z_t \mid m_i) \mathrm{Bel}_{t-1}(m_i) \end{aligned}$$

Log-Odds Ratio

Definition 4.1: For a given binary random variable X the odds ratio is defined as:

$$\frac{p(x)}{p(\neg x)} = \frac{p(x)}{1 - p(x)}$$

Accordingly, we define the log-odds ratio as:

$$l(x) := \log \frac{p(x)}{1 - p(x)}$$

By rearranging this, we obtain:

$$p(x) = 1 - \frac{1}{1 + \exp(l(x))}$$

p can be expressed using l

Bayes Filter with Log-Odds Ratio

With the log-odds ratio the Bayes filter simplifies:

$$\text{Bel}_t(m_i) = \frac{p(z_t \mid m_i)\text{Bel}_{t-1}(m_i)}{p(z_t \mid z_{1:t-1}, x_{1:t-1})}$$

(Bayes Rule)

Similarly:

$$\text{Bel}_t(\neg m_i) = \frac{p(\neg m_i \mid z_t)p(z_t)\text{Bel}_{t-1}(\neg m_i)}{p(\neg m_i)p(z_t \mid z_{1:t-1}, x_{1:t-1})}$$

This means:

$$\frac{\text{Bel}_t(m_i)}{\text{Bel}_t(\neg m_i)} = \frac{p(m_i \mid z_t)}{p(\neg m_i \mid z_t)} \frac{\text{Bel}_{t-1}(m_i)}{\text{Bel}_{t-1}(\neg m_i)} \frac{p(\neg m_i)}{p(m_i)}$$

Bayes Filter with Log-Odds Ratio

$$= \frac{p(m_i \mid z_t)}{1 - p(m_i \mid z_t)} \frac{\text{Bel}_{t-1}(m_i)}{1 - \text{Bel}_{t-1}(m_i)} \frac{1 - p(m_i)}{p(m_i)}$$

The log-odds ratio is then:

$$l_t(m_i) = \log \frac{p(m_i \mid z_t)}{1 - p(m_i \mid z_t)} + l_{t-1}(m_i) - \log \frac{p(m_i)}{1 - p(m_i)}$$

log-odds ratio of inverse sensor model

previous log-odds ratio

log-odds ratio of prior, usually zero

$$=: l_0$$

Occupancy Grid Mapping Algorithm

Algorithm Occ_grid_mapping $(l_{t-1}(\mathbf{m}), x_t, z_t)$:

1. for all cells m_i do
2. $\qquad l_t(m_i) = l_{t-1}(m_i) + \mathrm{inv\_sens\_model}(m_i, x_t, z_t) - l_0$
3. return $l_t(\mathbf{m})$

All computations are done in log-space.

$$\bar{l}(m_i) := \log \frac{p(m_i \mid z_t)}{1 - p(m_i \mid z_t)}$$

“Inverse sensor model”

The Inverse Sensor Model

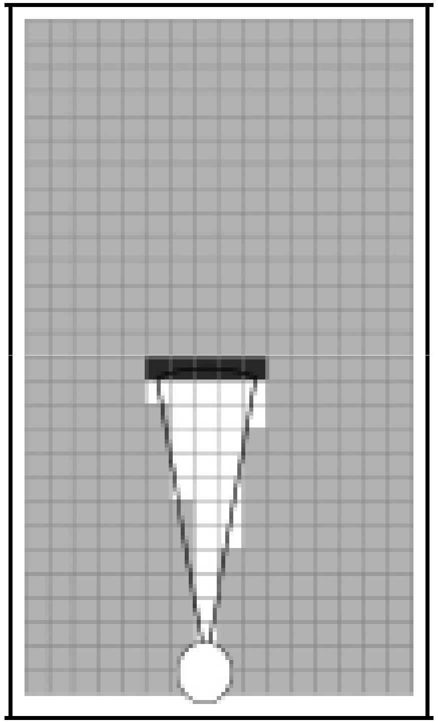
We need to compute: $\bar{l}(m_i) := \log \frac{p(m_i \mid z_t)}{1 - p(m_i \mid z_t)}$

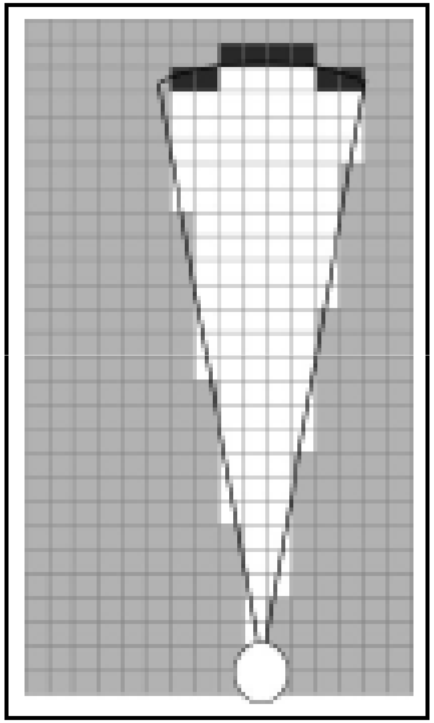
We assume a range sensor (e.g. laser scanner) that measures the distance z_t in meters.

Three possibilities:

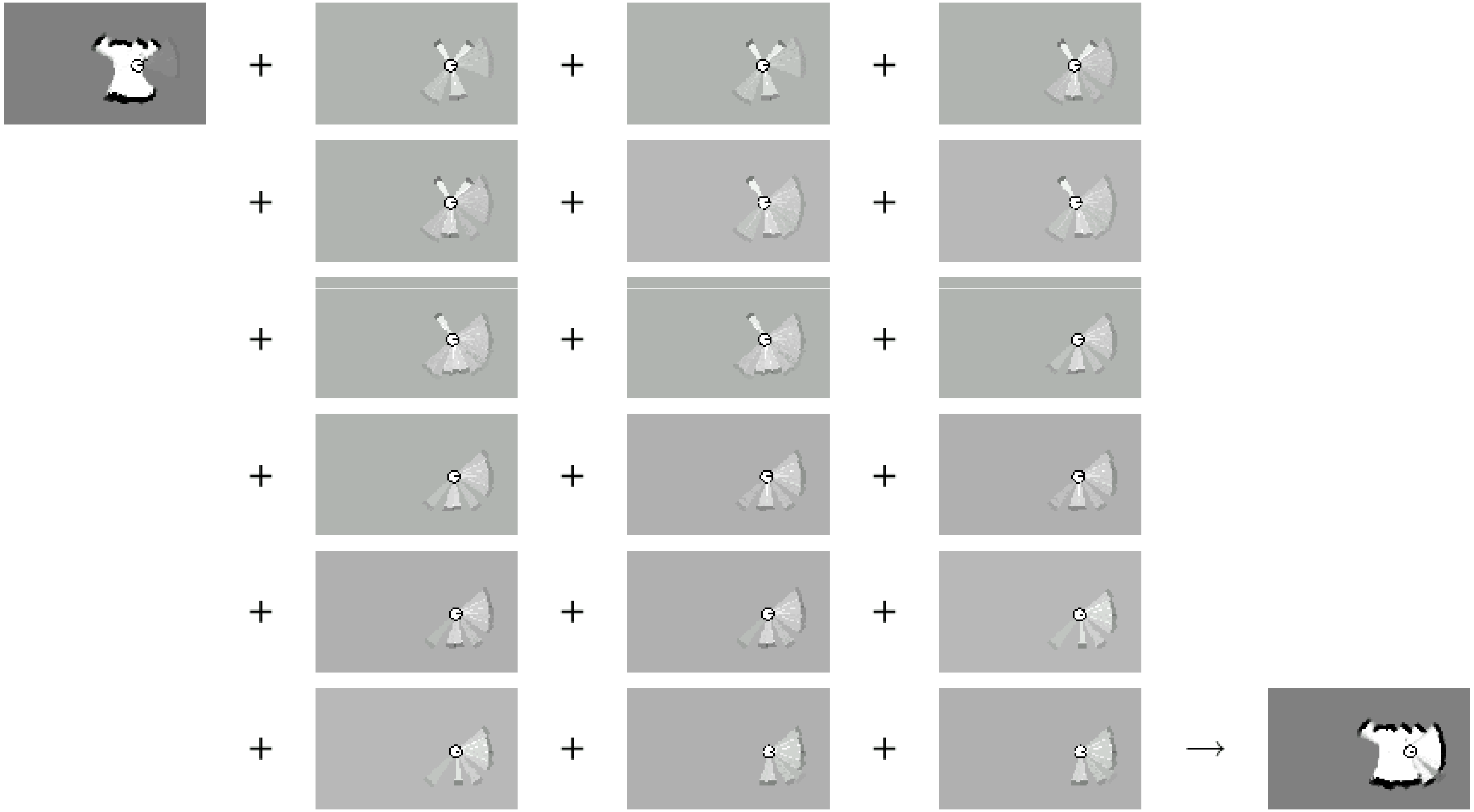
- The beam does not intersect the cell
  $\rightarrow$ No change of the occupancy value: $\bar{l}(m_i) = l_0$
- The beam passes through the cell
  $\rightarrow$ Occupancy value decreases: $\bar{l}(m_i) = l_{\text{free}}$
- The beam ends in the cell
  $\rightarrow$ Occupancy value increases: $\bar{l}(m_i) = l_{\text{occ}}$

The Resulting Grid Map

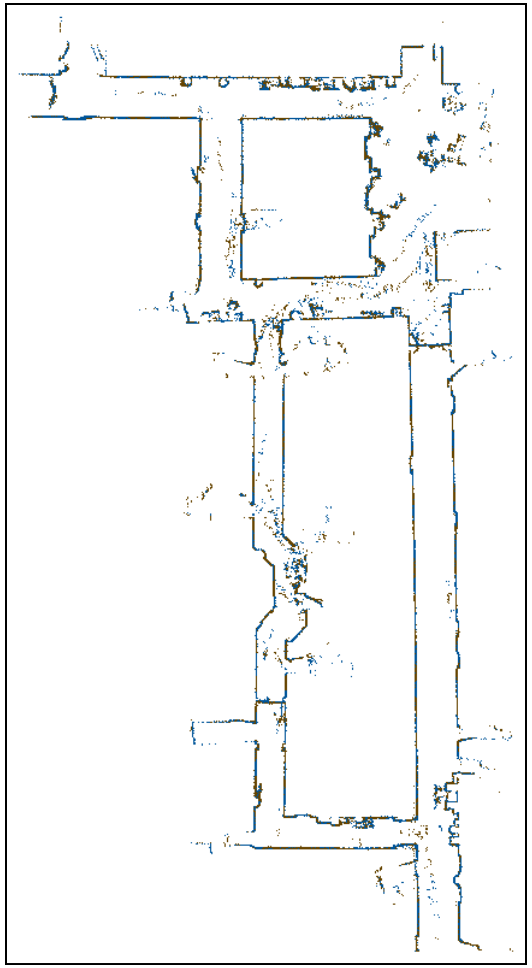




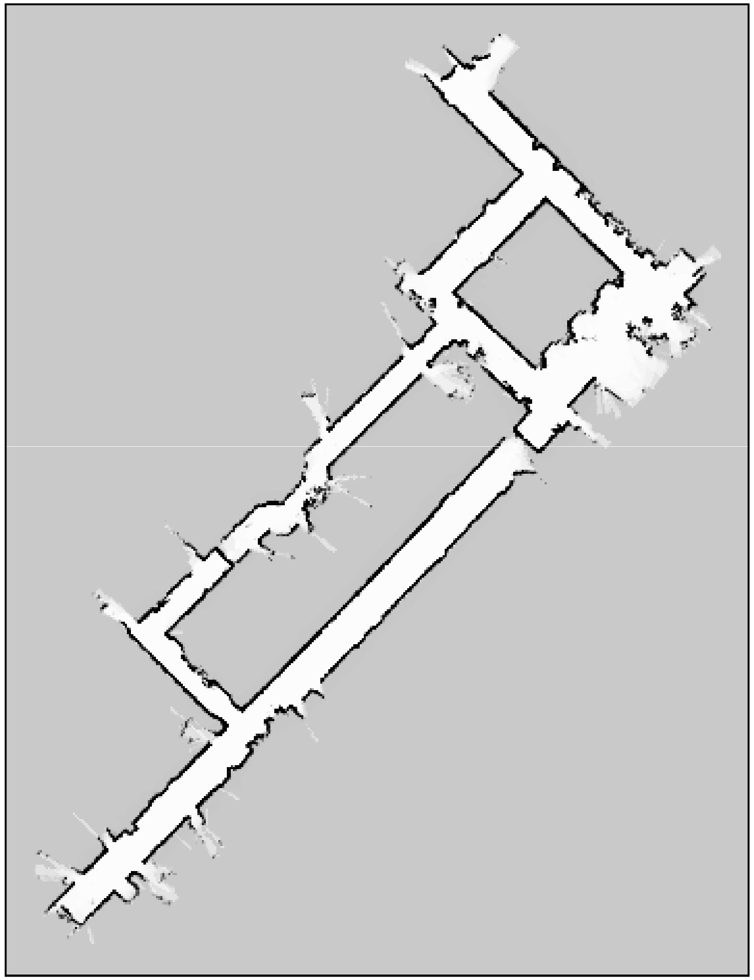
Incremental Updating of the Grid



Occupancy Grids: From Scans to Maps

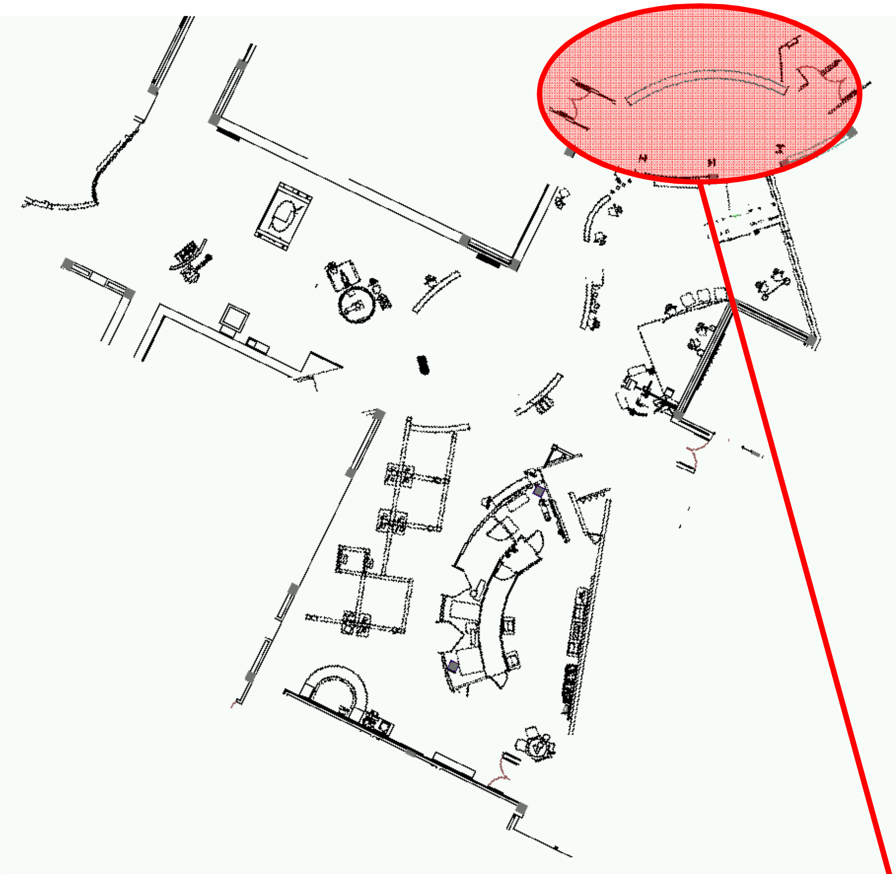


laser scan data

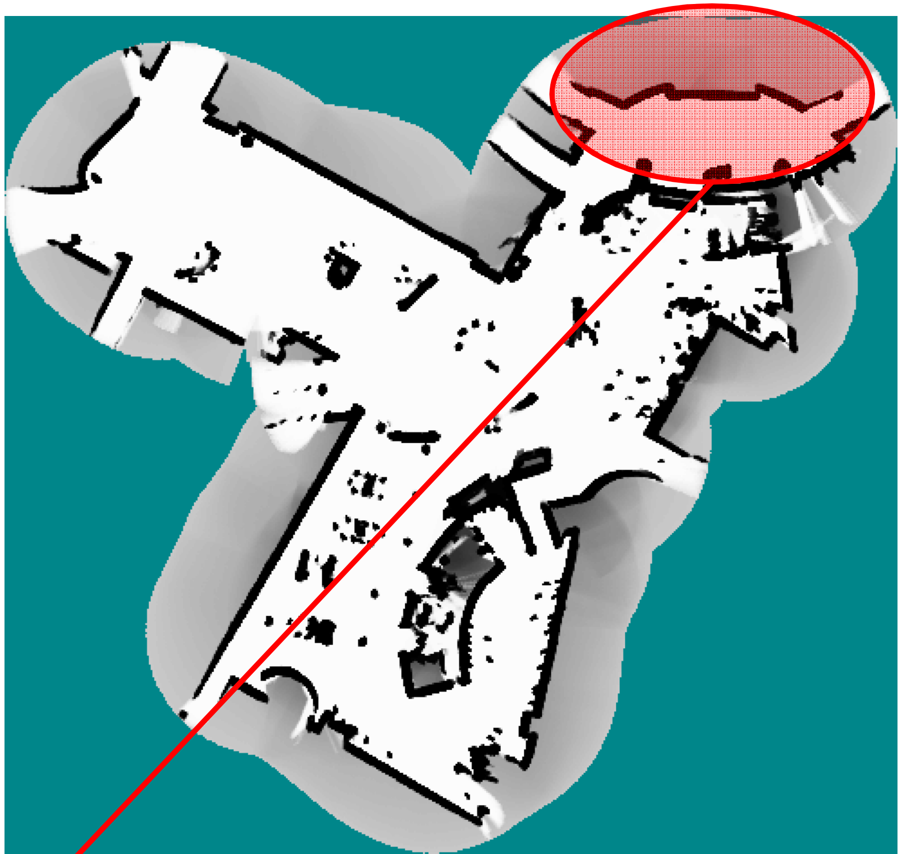


occupancy grid

Tech Museum, San Jose



CAD map



occupancy grid map

The grid map is more accurate!

The SLAM Problem

So far we had:

- Localization: given a map, find the current robot position
- Mapping: given a set of robot positions, find the most likely map

Now we want to solve both at the same time:

- Given a sequence of sensor measurements and motion commands, find the correct robot positions and the most likely map!

Simultaneous Localization and Mapping (SLAM)

Existing SLAM Techniques

- Extended Kalman filter
  - [Cheeseman and Smith ‘86]
- Sparse Extended Information Filter
  - [Thrun *et al*. ‘04]
- Rao-Blackwellized Particle Filter
  - [Murphy ‘99, Doucet *et al*. ‘00]
- GraphSLAM
  - [Lu and Milios ’97, Gutmann ‘99]

Taxonomy of the SLAM Problem

We distinguish between:

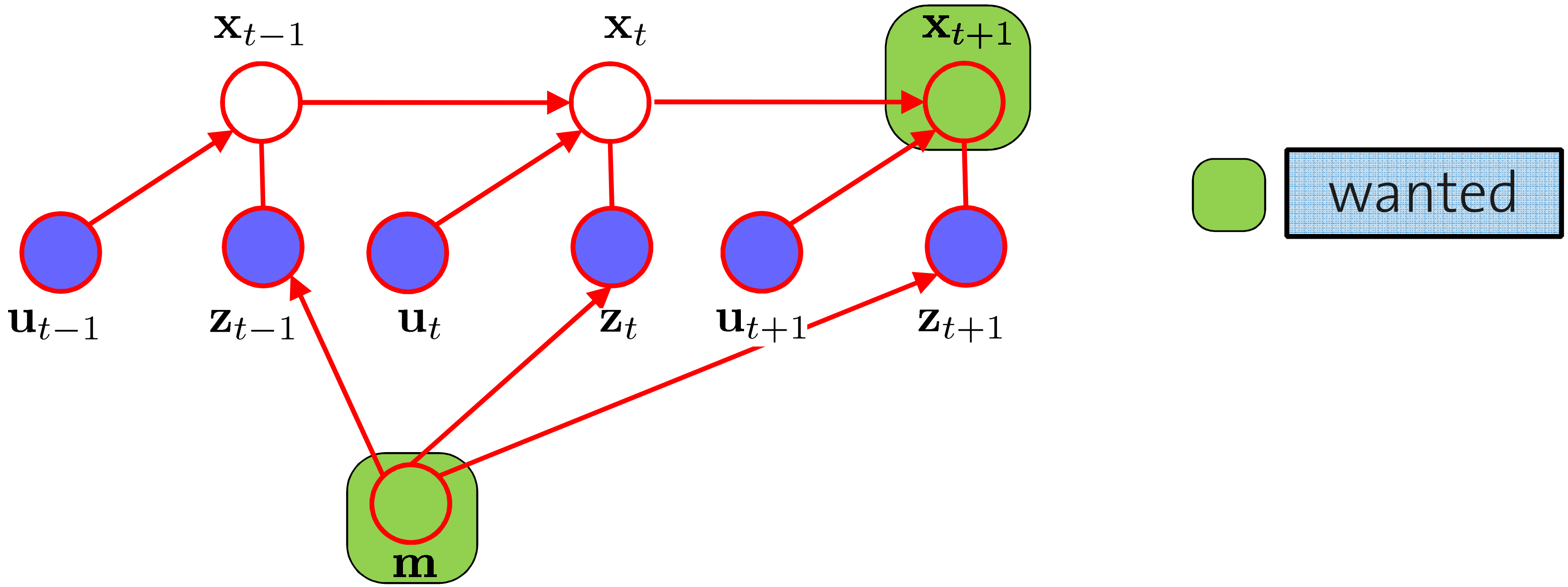
- Online-SLAM: given all measurements and all motion commands up to time t, compute the current pose and the map:

$$p(x_t, \mathbf{m} \mid z_{1:t}, u_{1:t})$$

- Full SLAM: given all measurements and all motion commands up to time t, compute the entire path and the map:

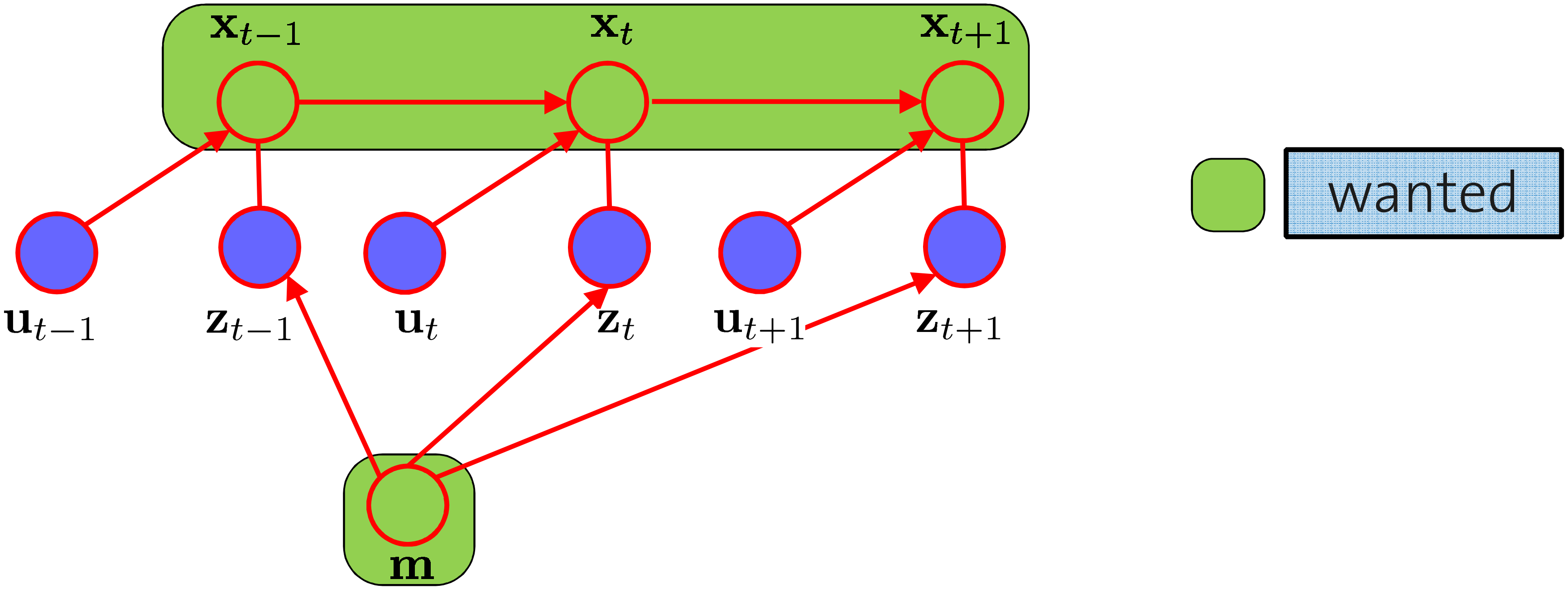
$$p(x_{1:t}, \mathbf{m} \mid z_{1:t}, u_{1:t})$$

Graphical Model of Online-SLAM:



$$p(\mathbf{x}_t, \mathbf{m} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \int \cdots \int p(\mathbf{x}_{1:t}, \mathbf{m} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) d\mathbf{x}_1 \ldots d\mathbf{x}_{t-1}$$

Graphical Model of Full SLAM:



$$p(\mathbf{x}_{1:t}, \mathbf{m} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t})$$

Map Representations

Two different map representations may be used:

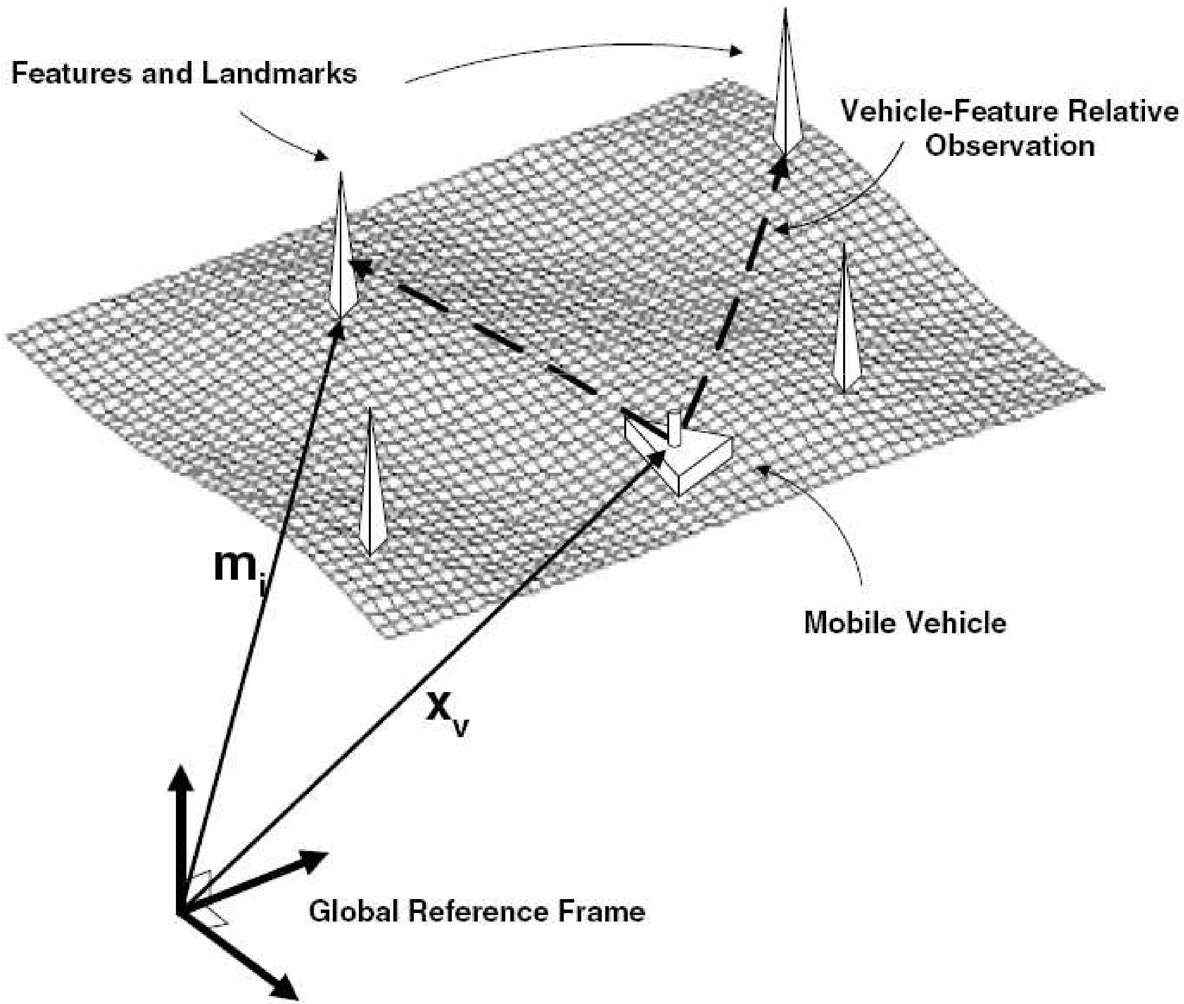
- Landmark-based maps: the map consists of a fixed number of N point landmarks (e.g. corners, pillars)

$$\mathbf{m} = (m_{1,x}, m_{1,y}) \ldots (m_{N,x}, m_{N,y})$$

- Occupancy grid maps: the map consists of a two-dimensional grid with N rows and M columns:

$$\mathbf{m} = \{m_{x,y} \mid x = 1, \ldots, N;\ y = 1, \ldots, M\}$$

The Landmark-based SLAM-Problem



Summary

- Particle filters are an implementation of recursive Bayesian filtering
- They represent the posterior by a set of weighted samples
- They can model non-Gaussian distributions
- Proposal to draw new samples
- Weight to account for the differences between the proposal and the target
- Monte Carlo filter, Survival of the fittest, Condensation, Bootstrap filter