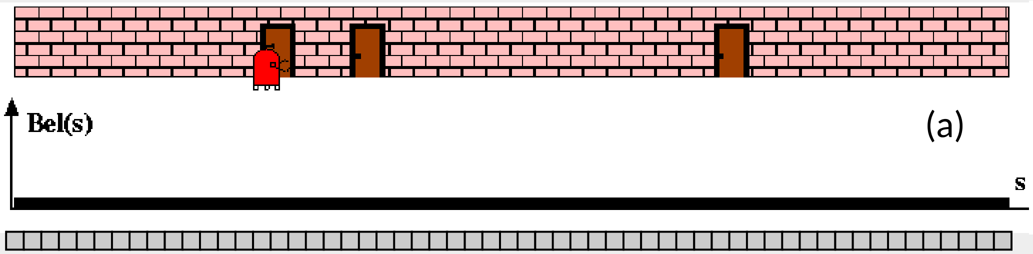
Advanced Automation

15

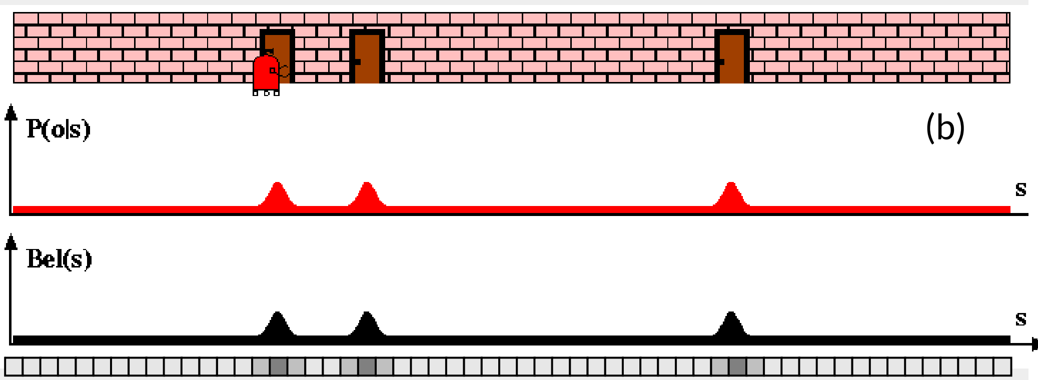
Dorit Borrmann, M.Sc.
Robotics and Telematics
borrmann@informatik.uni-wuerzburg.de

Last Lecture: Markov Localization: 1D Example

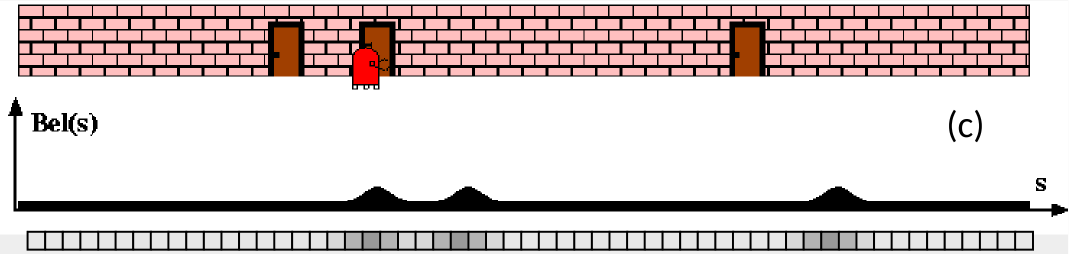
State space *S* : *z*-coordinate in a corridor



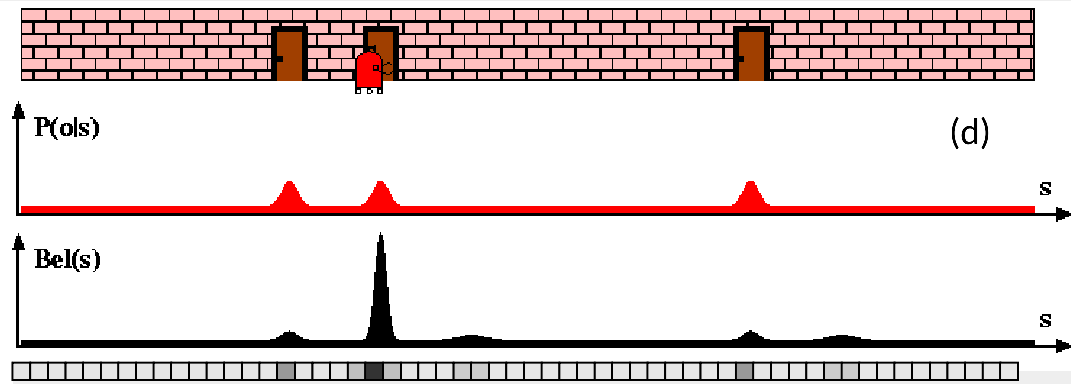
Start: No information, i.e., uniform distribution of possible poses *U*[*S*](*s*)

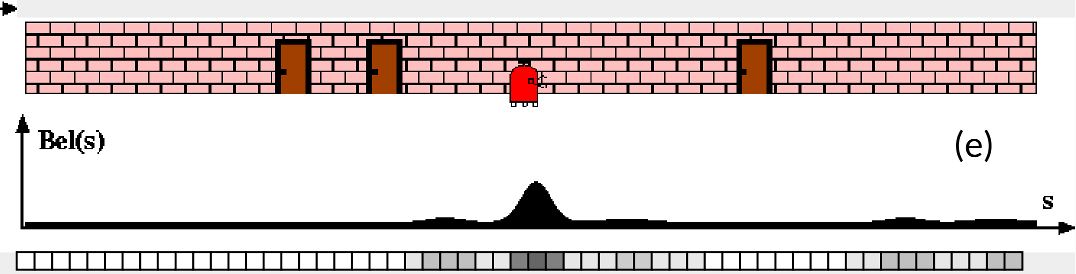


Observation: Door detected! Distribution accumulates at door



Motion: Distributions „proceed" according to motion model



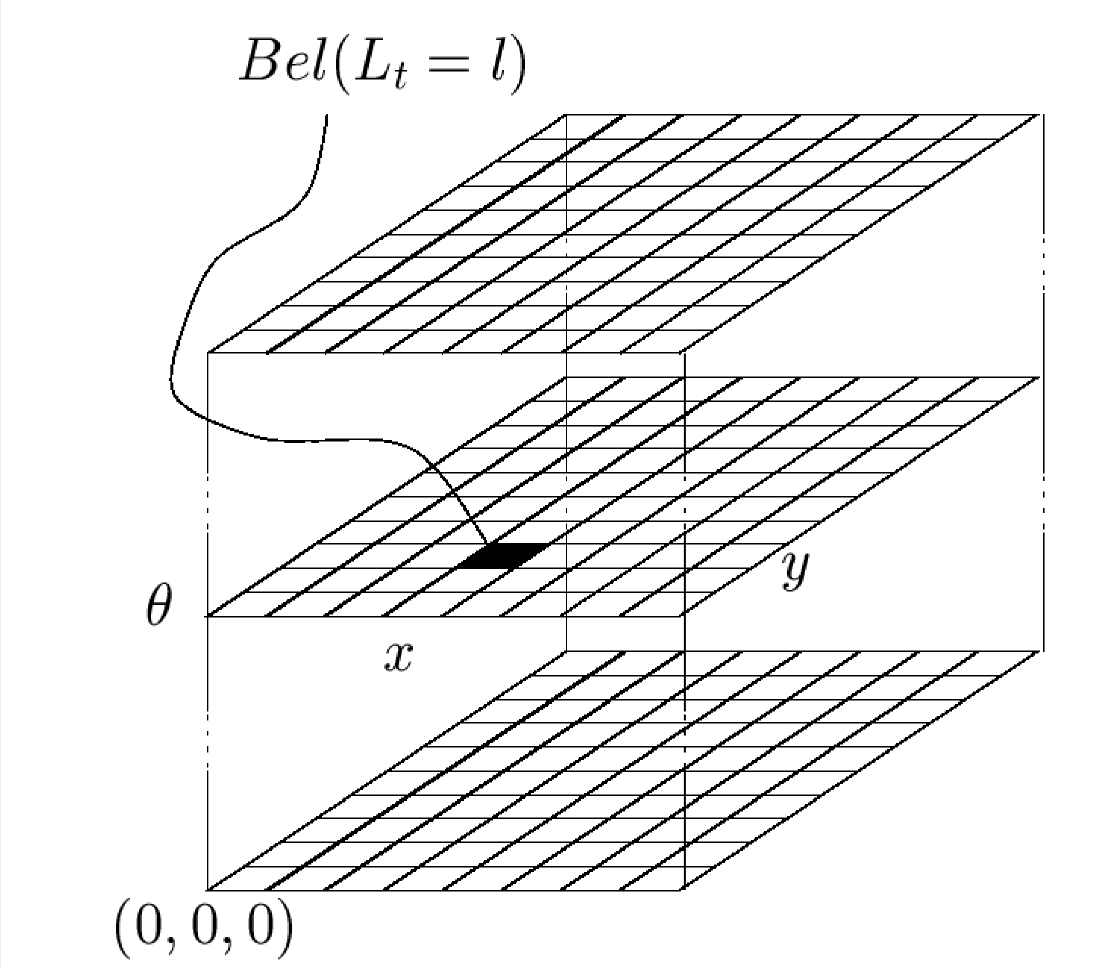


Last Lecture: Markov Localization in grid maps

Iterate for the whole state space the cycle of

- Action
- a-priori pose estimation
- Measurement
- a-posteriori pose estimation

- **Size of the state space** (grid cells) in buildings, e.g., 2000cm x 5000cm / (10cm)^2 x 100 (angle resolution) = 10^7



Last Lecture: Markov Localization in grid maps

- States sparsely interconnected
- However not processable in real time
  - Too many multiplications
  - Marginal probability mass per pose
- For small discrete state spaces (only for small!) applicable

Markov Localization – Sensor and Motion Model

- Motion model
  - Mixture of two independent, zero-centered Gaussian distributions;
    The variances of these distributions are proportional to the length of the measured motion.

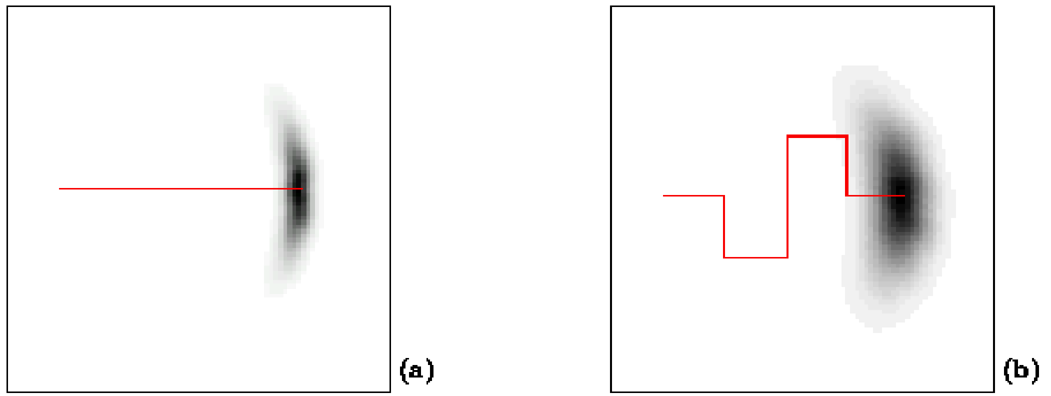
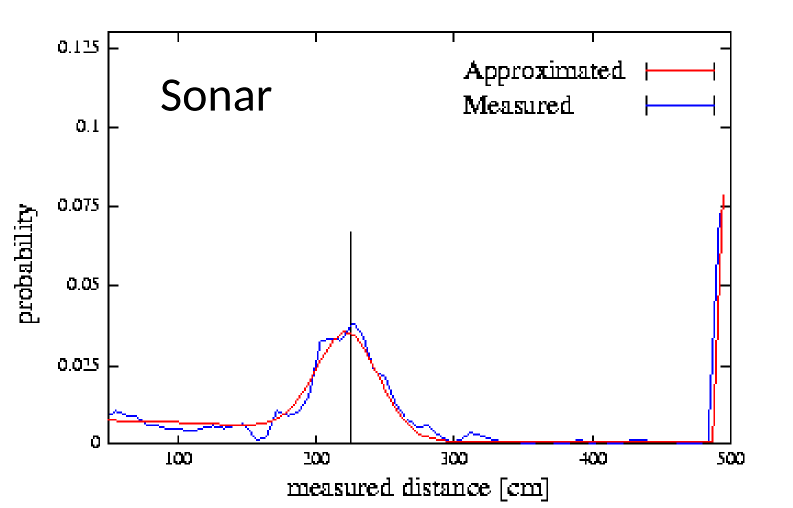
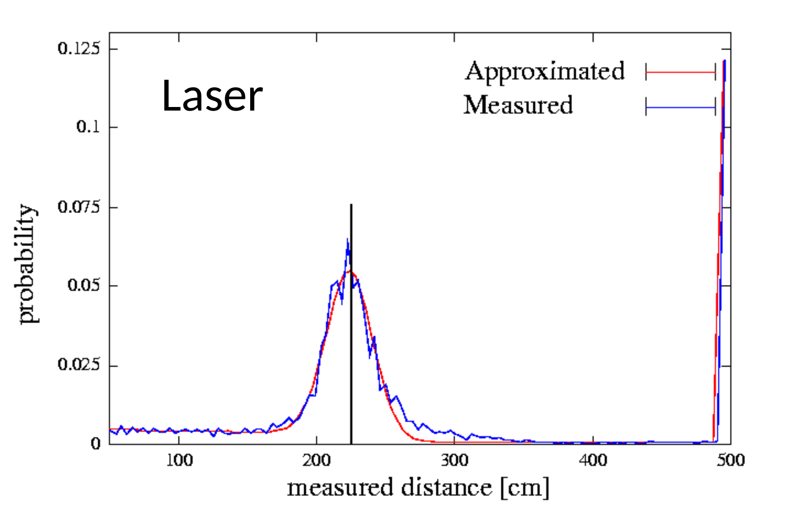


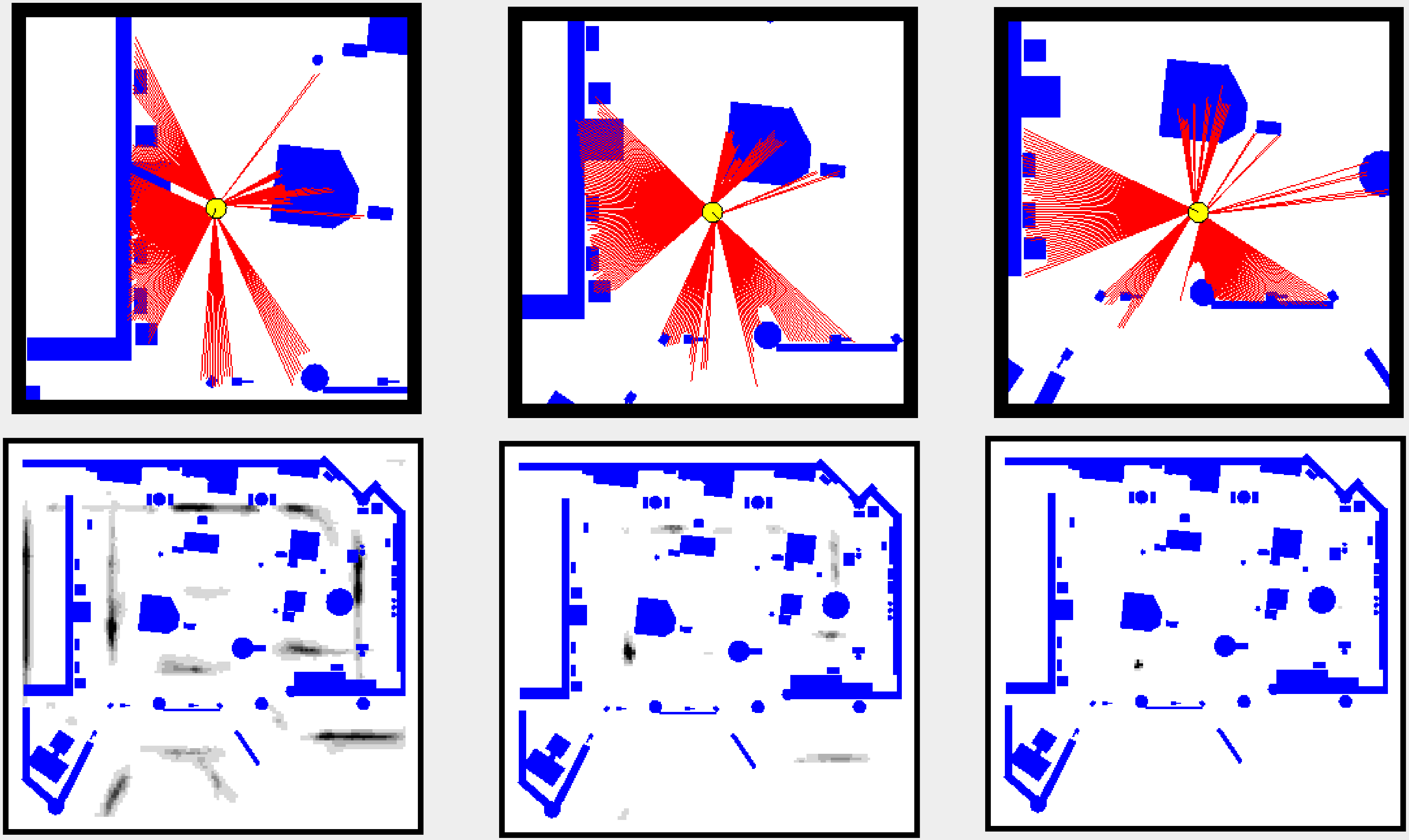
Fig. 3. Typical "banana-shaped" distributions resulting from different motion actions.

- Sensor Model

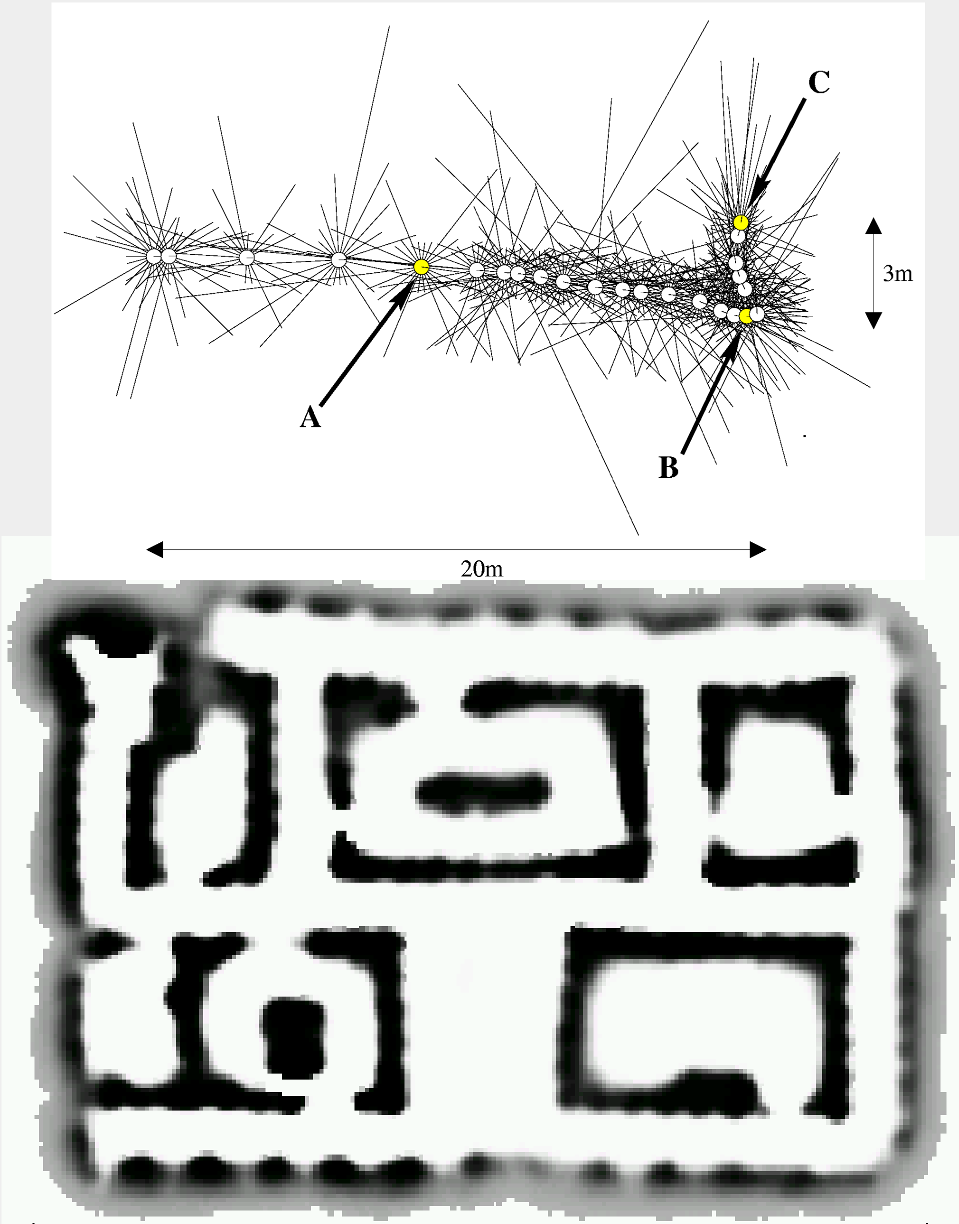


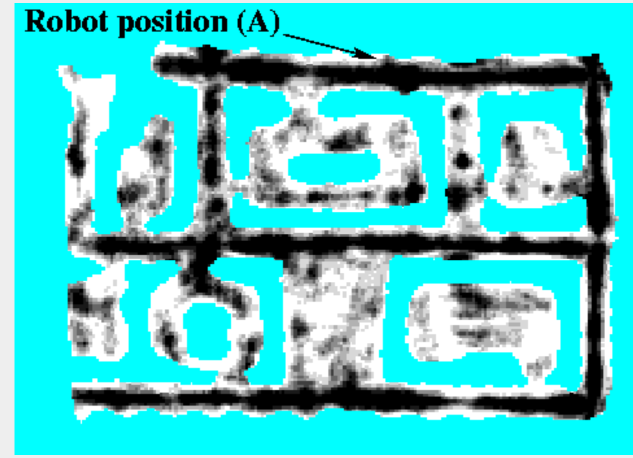


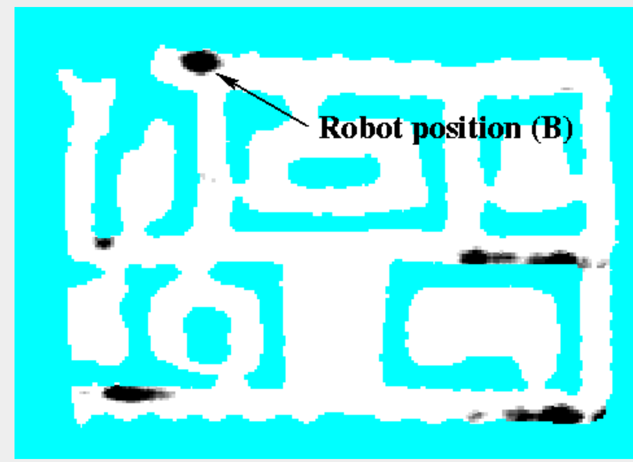
Markov Localization – Example

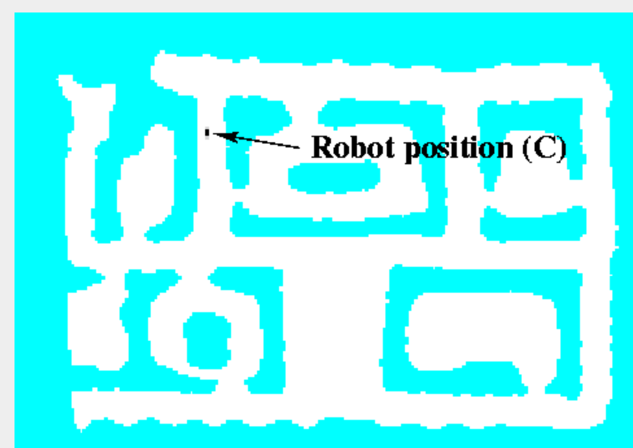


Sonars and Occupancy Grid Map









Markov Localization – Implementation (1)

- To update the belief upon sensory input and to carry out the normalization one has to iterate over all cells of the grid.
- Especially when the belief is peaked (which is generally the case during position tracking), one wants to avoid updating irrelevant aspects of the state space.
- One approach is not to update entire sub-spaces of the state space.
- This, however, requires to monitor whether the robot is de-localized or not.
- To achieve this, one can consider the likelihood of the observations given the active components of the state space.

Markov Localization – Implementation (2)

- To efficiently update the belief upon robot motions, one typically assumes a bounded Gaussian model for the motion uncertainty.
- This reduces the update cost from *O(n²)* to *O(n)*, where *n* is the number of states.
- The update can also be realized by shifting the data in the grid according to the measured motion.
- In a second step, the grid is then convolved using a separable Gaussian Kernel.
- Two-dimensional example:

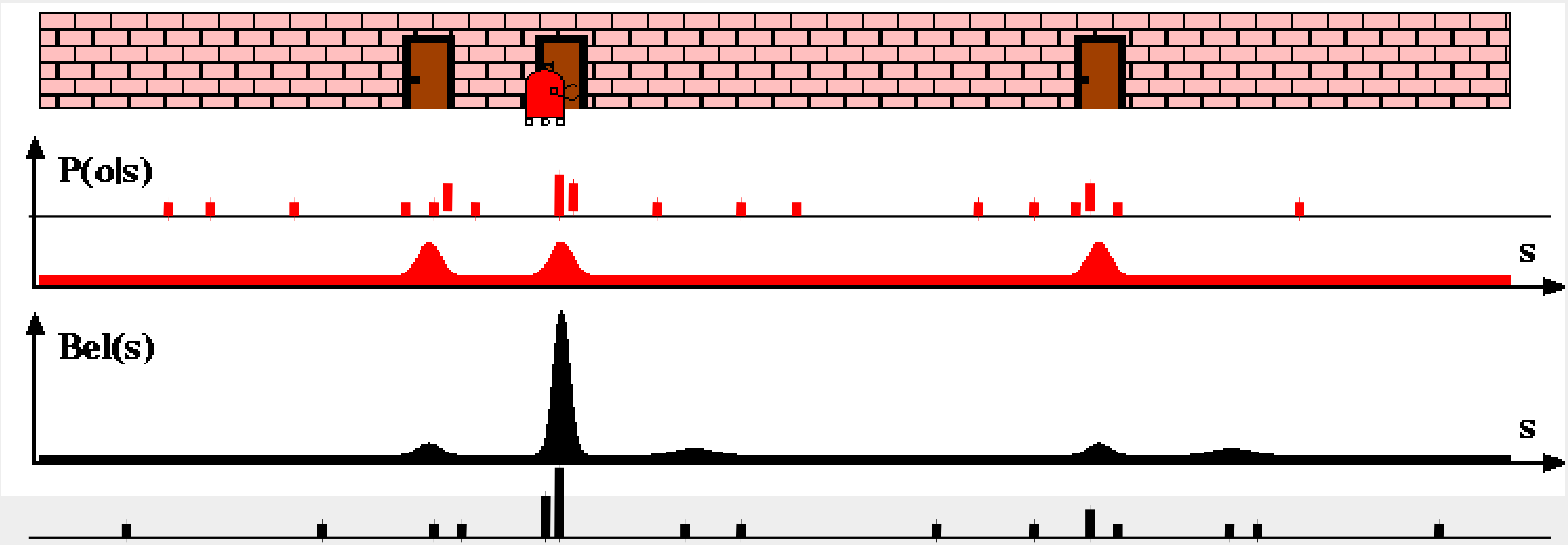
$$\begin{array}{|c|c|c|}\hline 1/16 & 1/8 & 1/16 \\ \hline 1/8 & 1/4 & 1/8 \\ \hline 1/16 & 1/8 & 1/16 \\ \hline \end{array} \cong \begin{array}{|c|}\hline 1/4 \\ \hline 1/2 \\ \hline 1/4 \\ \hline \end{array} + \begin{array}{|c|c|c|}\hline 1/4 & 1/2 & 1/4 \\ \hline \end{array}$$

- Fewer arithmetic operations
- Easier to implement

Approximation through Sampling

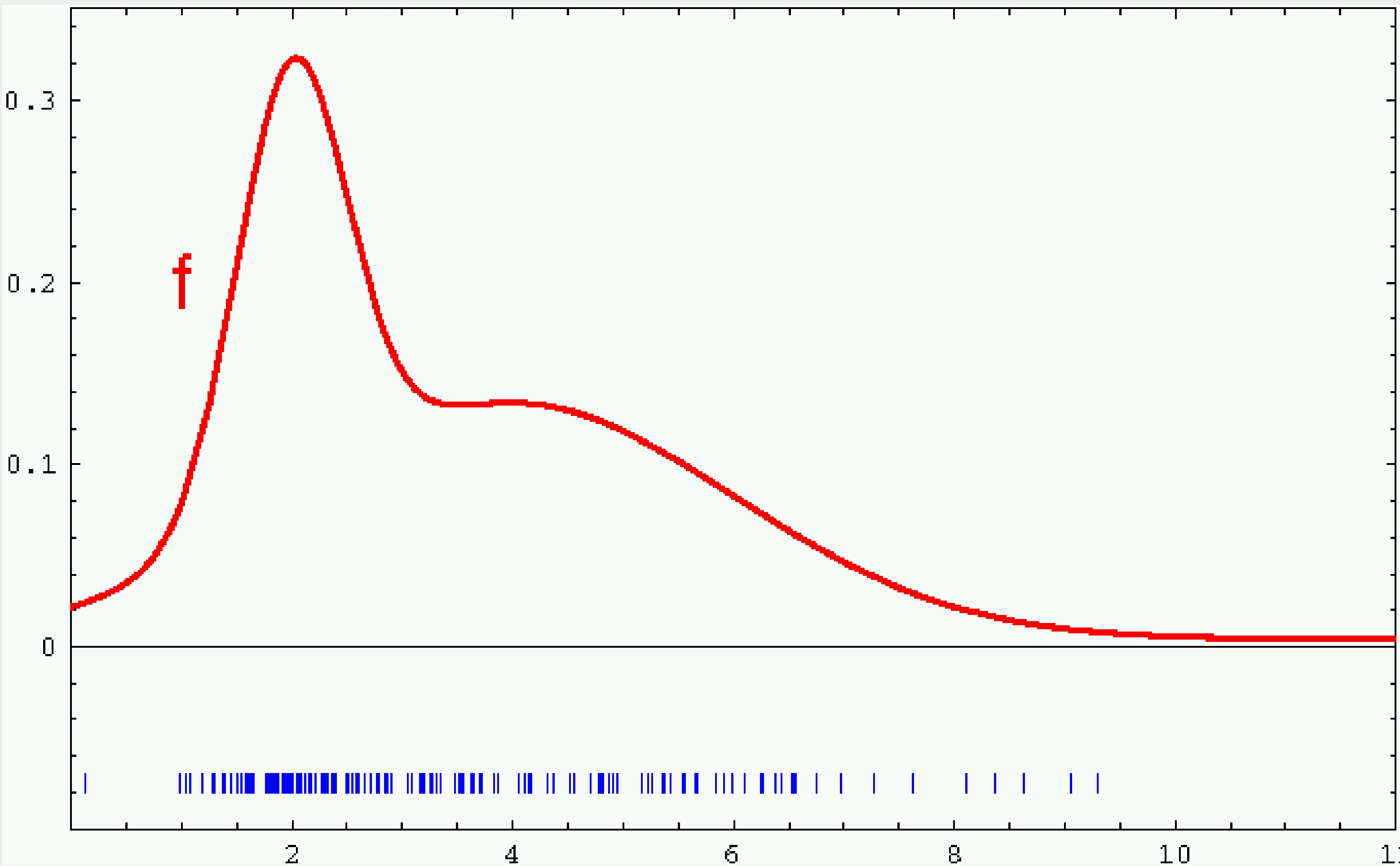
- There is the general scheme of stochastic approximation in computer science („Monte Carlo Algorithms“, „*Sampling*“, „Particle filter“):

Approximate behavior of a random variable through (a relatively small number of) samples!



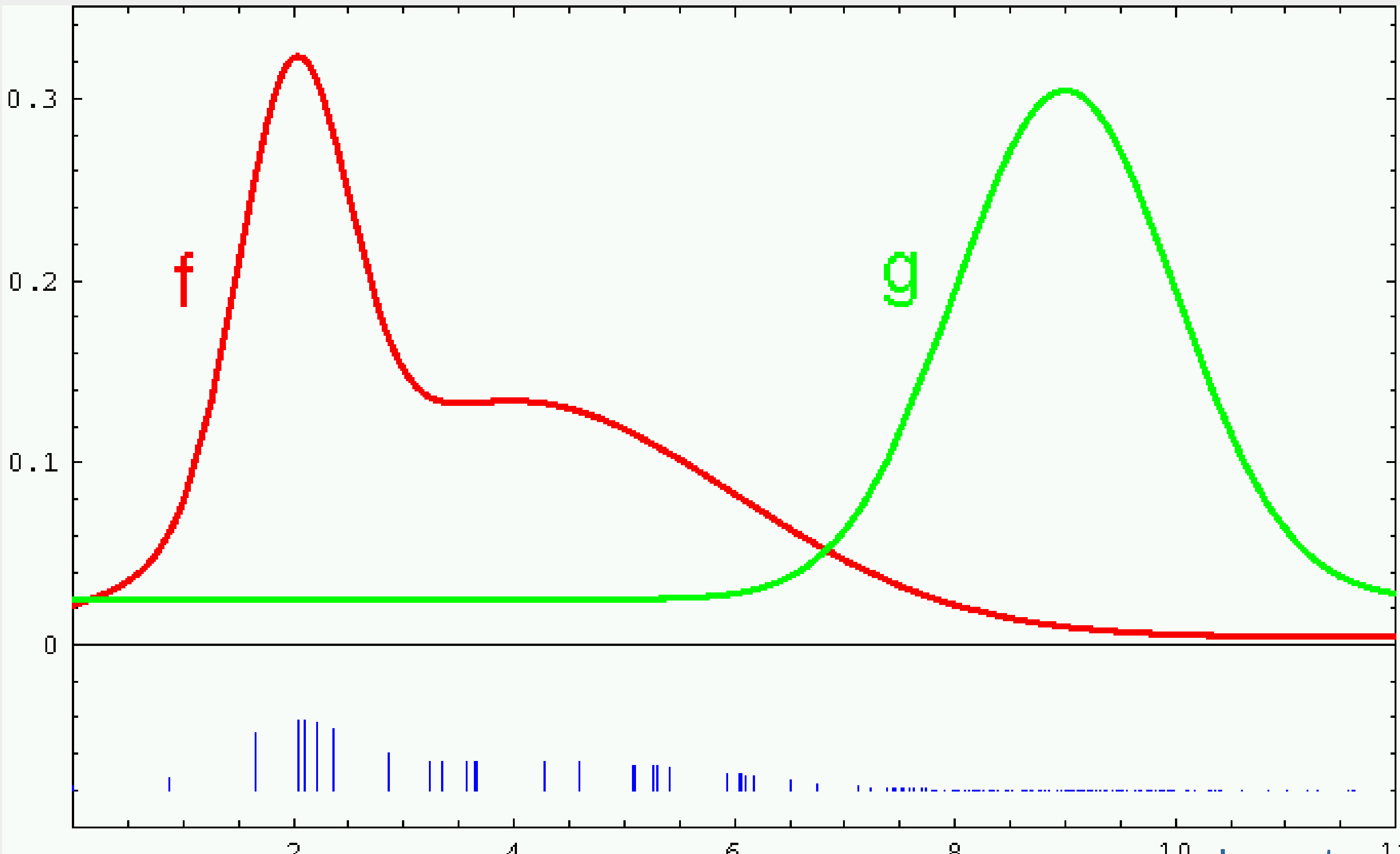
Particle Filter (1)

- Represent probability distributions through samples
- One can represent also non-Gaussian distributions



Particle Filter (2)

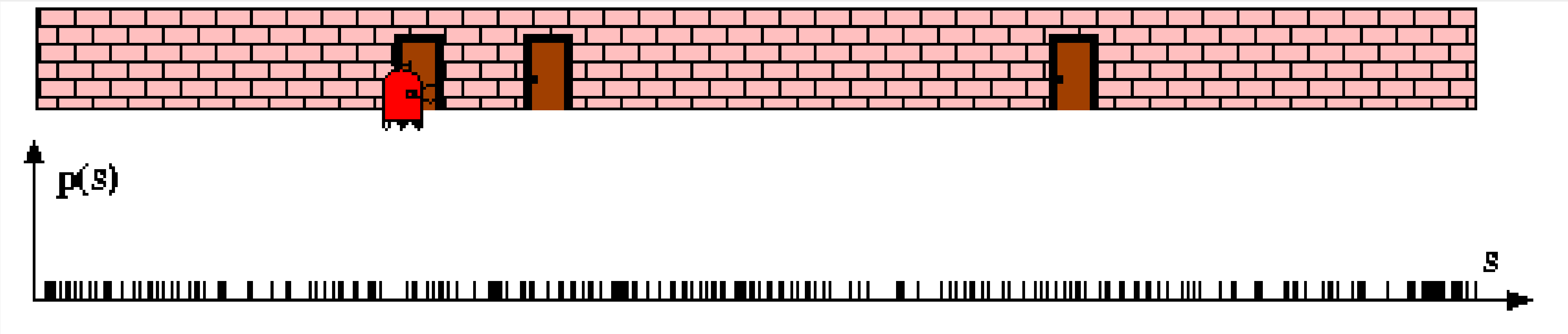
- Represent probability distributions through samples
- One can represent also non-Gaussian distributions



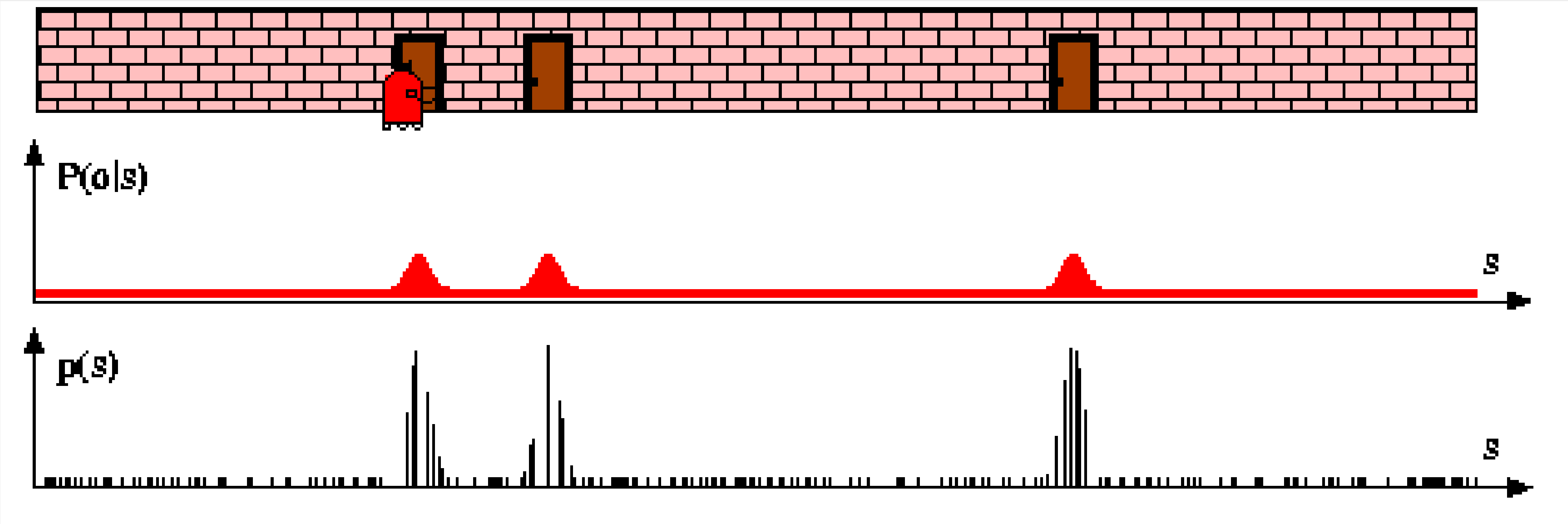
Importance Sampling

Monte-Carlo Localization (1)

- Global Localization



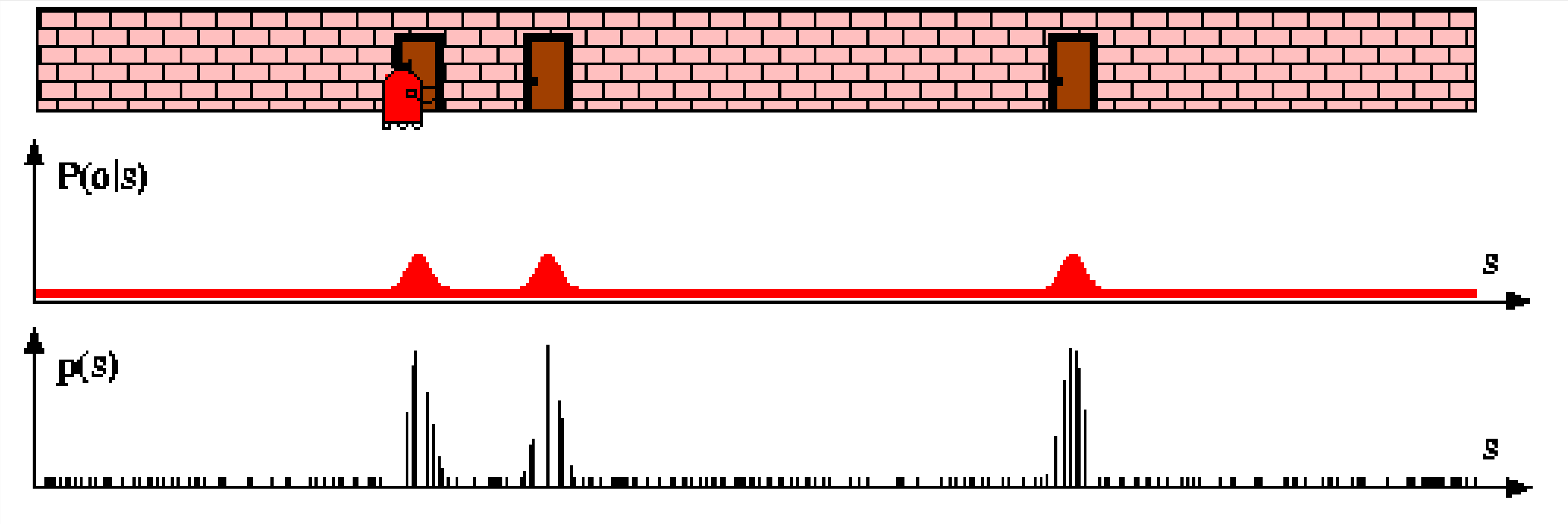
- Update with sensor values



Monte-Carlo Localization (2)

- Update of the weights

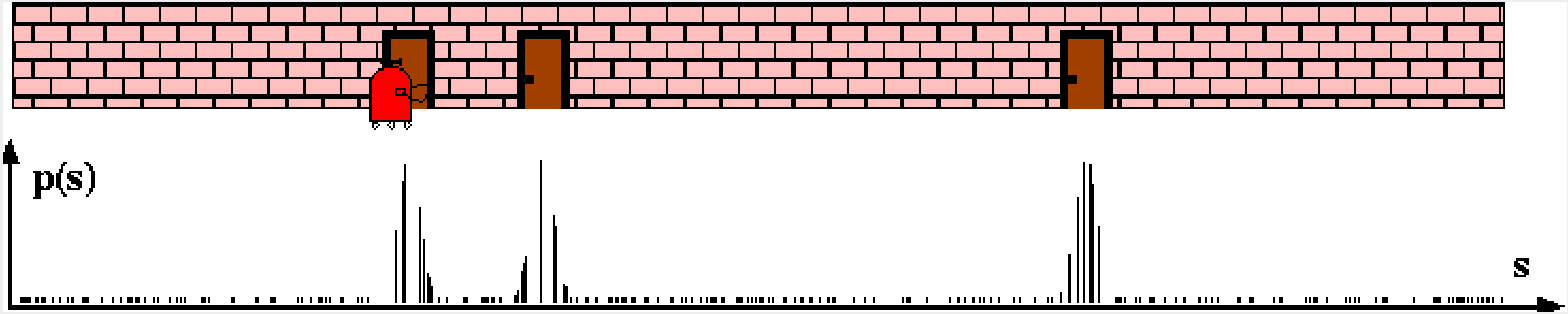
$$
\begin{aligned}
Bel(x) &\leftarrow \alpha\ p(z \mid x)\, Bel^{-}(x) \\
w &\leftarrow \frac{\alpha\ p(z \mid x)\, Bel^{-}(x)}{Bel^{-}(x)} = \alpha\ p(z \mid x)
\end{aligned}
$$



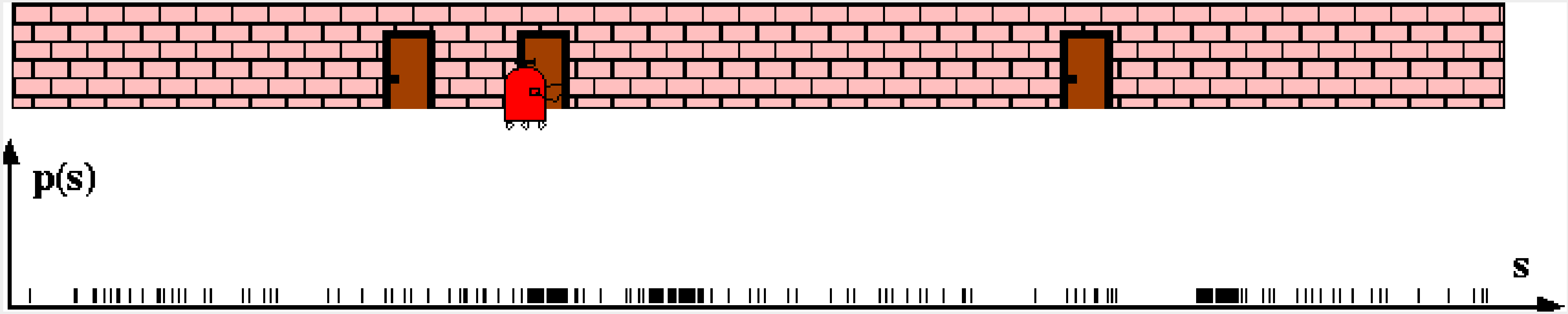
Monte-Carlo Localization (3)

- Update of the robot pose with motion model

$$Bel^{-}(x) \leftarrow \int p(x \mid u, x')\, Bel(x')\, \mathrm{d}x'$$



- Resampling

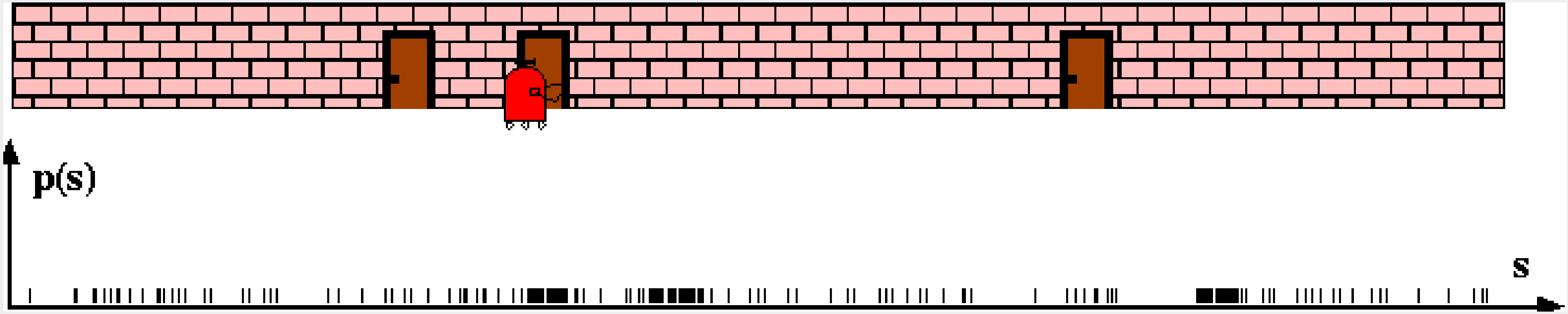


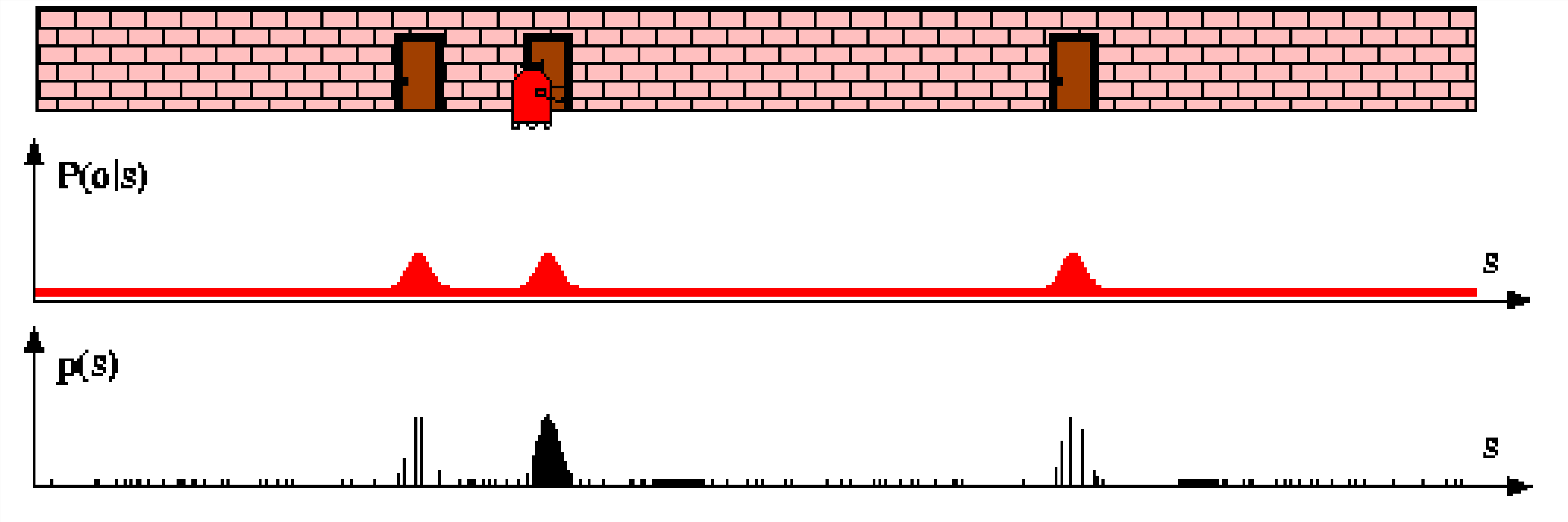
Monte-Carlo Localization (4)

- Update with sensor values
  update of the weights

$$Bel(x) \leftarrow \alpha\ p(z\,|\,x)\ Bel^{-}(x)$$

$$w \leftarrow \frac{\alpha\ p(z\,|\,x)\ Bel^{-}(x)}{Bel^{-}(x)} = \alpha\ p(z\,|\,x)$$

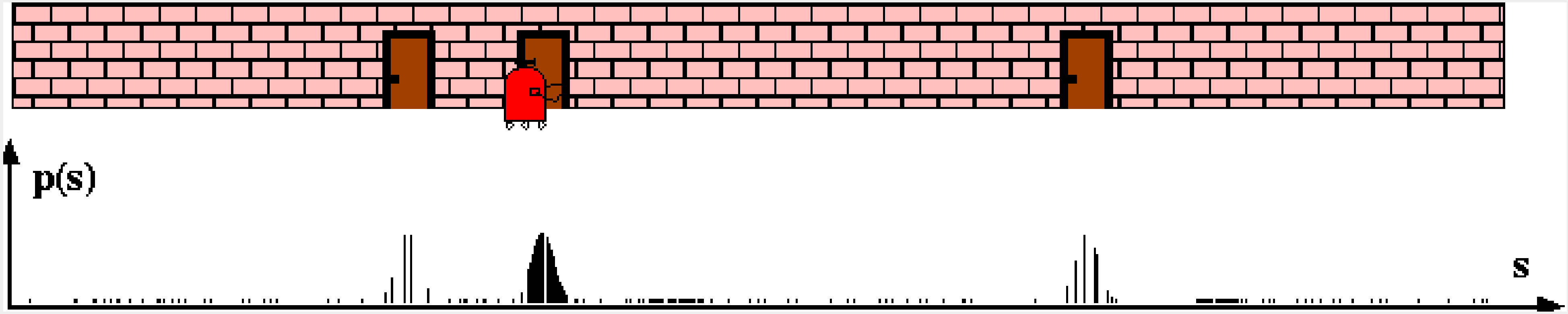




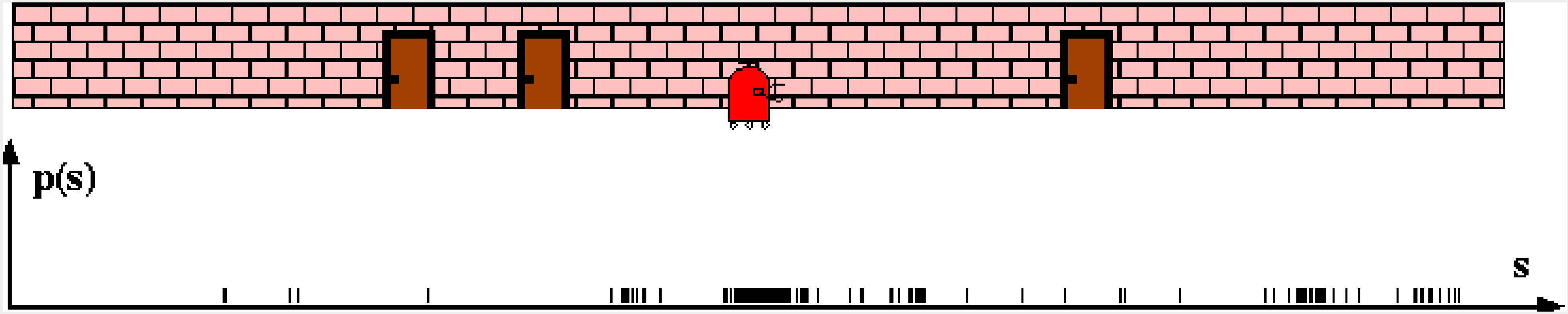
Monte-Carlo Localization (5)

- Update of the robot pose with motion model

$$Bel^{-}(x) \quad \leftarrow \quad \int p(x \mid u, x')\, Bel(x')\;\mathrm{d}x'$$



- Resampling



Particle Filter Algorithm

1. Algorithm **particle_filter**(S_{t-1}, u_{t-1} z_t):
2. $S_t = \emptyset, \quad \eta = 0$
3. **For** $i = 1 \ldots n$ — *Generate new samples*
4. Sample index $j(i)$ from the discrete distribution given by w_{t-1}
5. Sample x_t^i from $p(x_t \mid x_{t-1}, u_{t-1})$ using $x_{t-1}^{j(i)}$ and u_{t-1}
6. $w_t^i = p(z_t \mid x_t^i)$ — *Compute importance weight*
7. $\eta = \eta + w_t^i$ — *Update normalization factor*
8. $S_t = S_t \cup \{< x_t^i, w_t^i >\}$ — *Insert*
9. **For** $i = 1 \ldots n$
10. *Normalize weights*
11. $w_t^i = w_t^i / \eta$

Monte Carlo Localization is a Bayes Filter

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

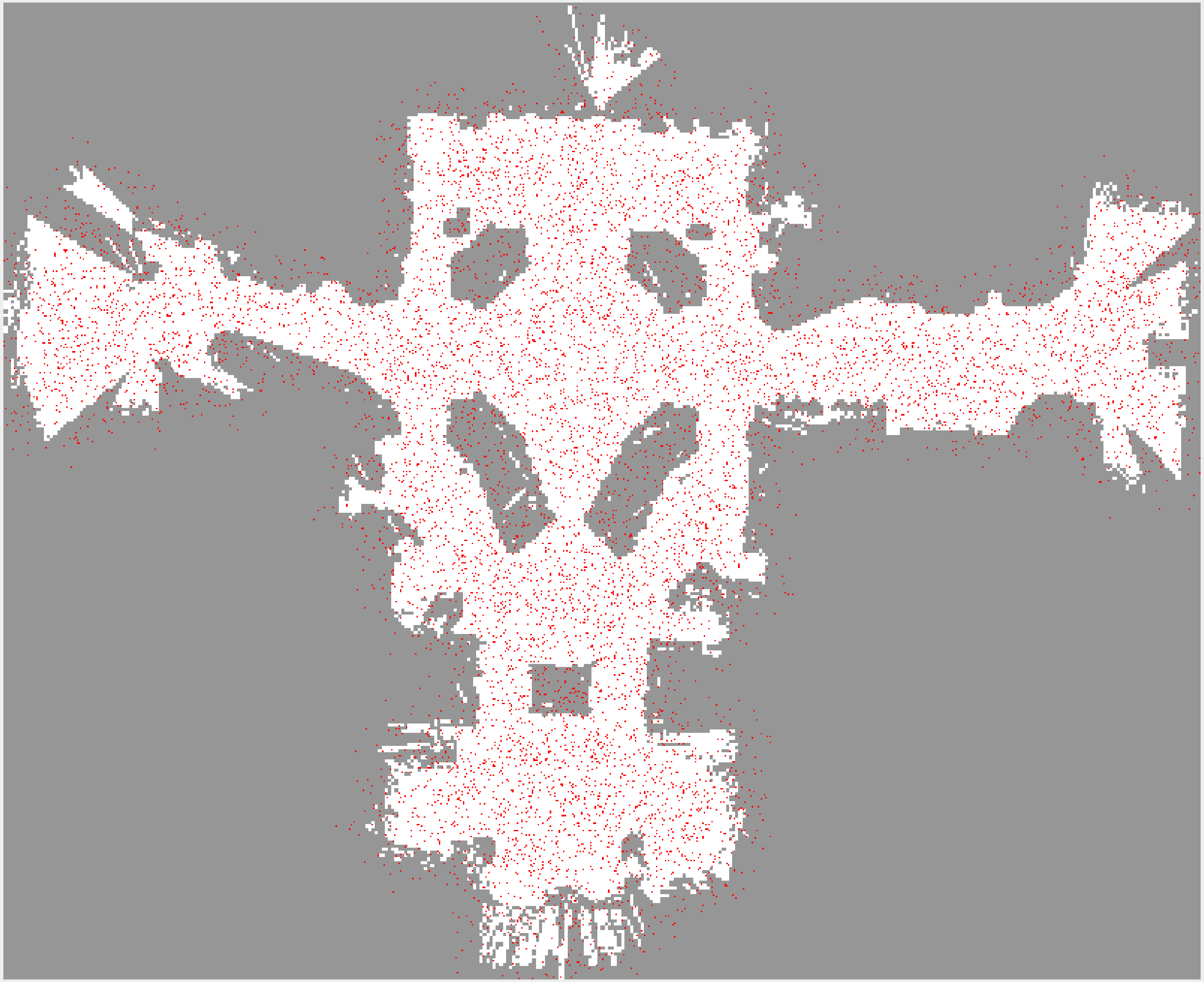
draw x^i_{t-1} from $Bel(x_{t-1})$

draw x^i_t from $p(x_t \mid x^i_{t-1}, u_{t-1})$

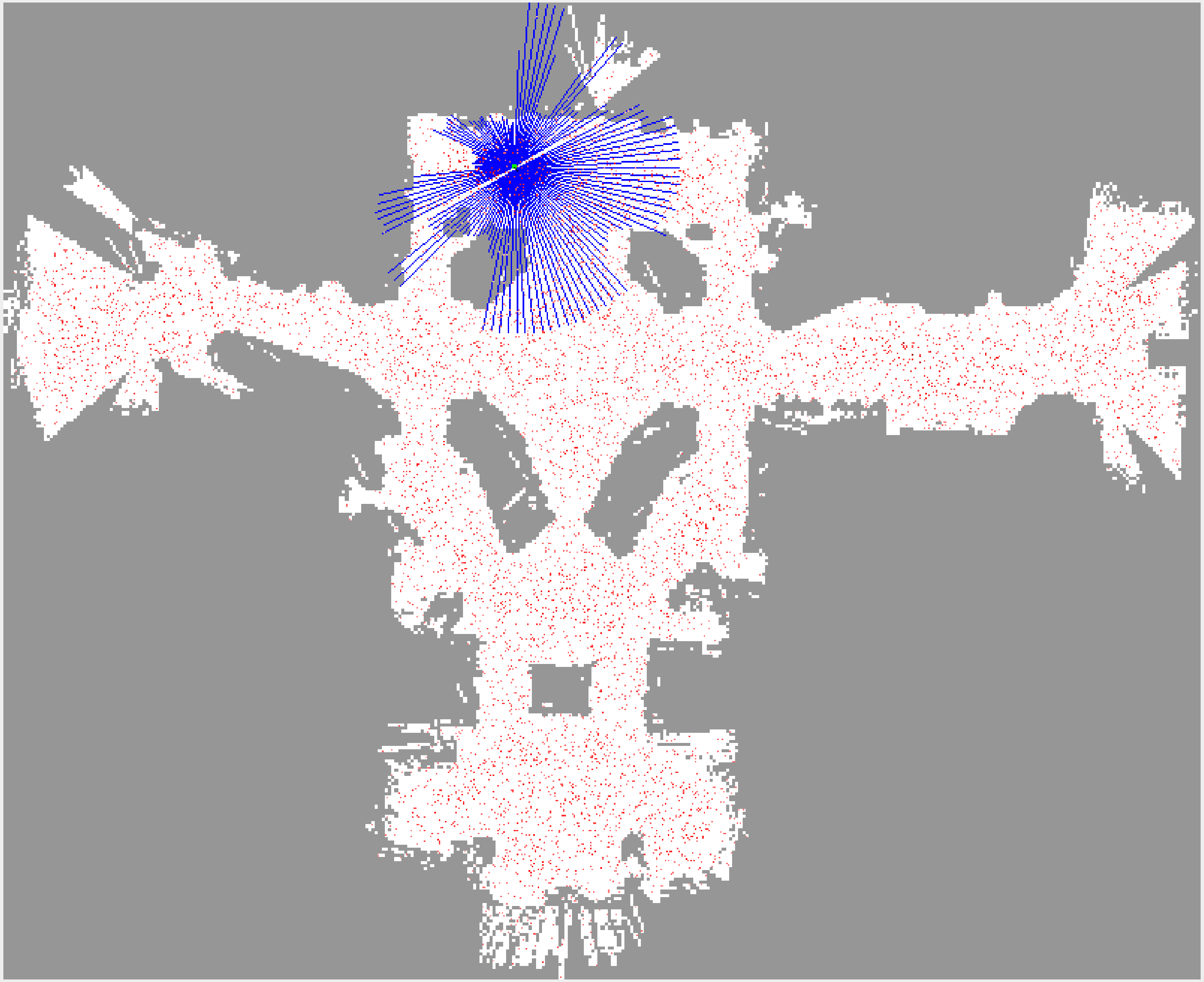
Importance factor for x^i_t:

$$\begin{aligned} w^i_t &= \frac{\text{target distribution}}{\text{proposal distribution}} \\ &= \frac{\eta\ p(z_t \mid x_t)\ p(x_t \mid x_{t-1}, u_{t-1})\ Bel(x_{t-1})}{p(x_t \mid x_{t-1}, u_{t-1})\ Bel(x_{t-1})} \\ &\propto p(z_t \mid x_t) \end{aligned}$$

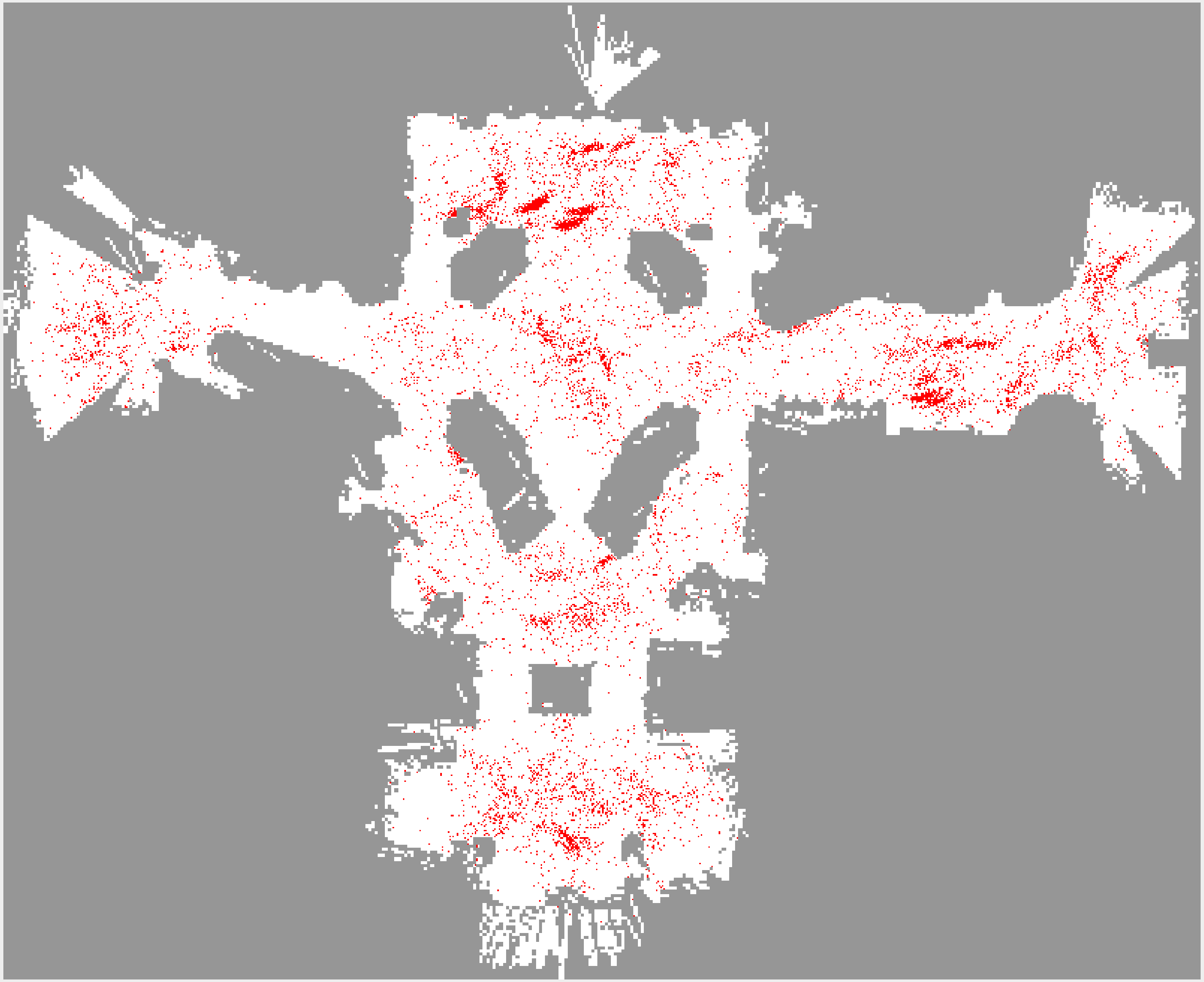
Particle Filter – Example (1)



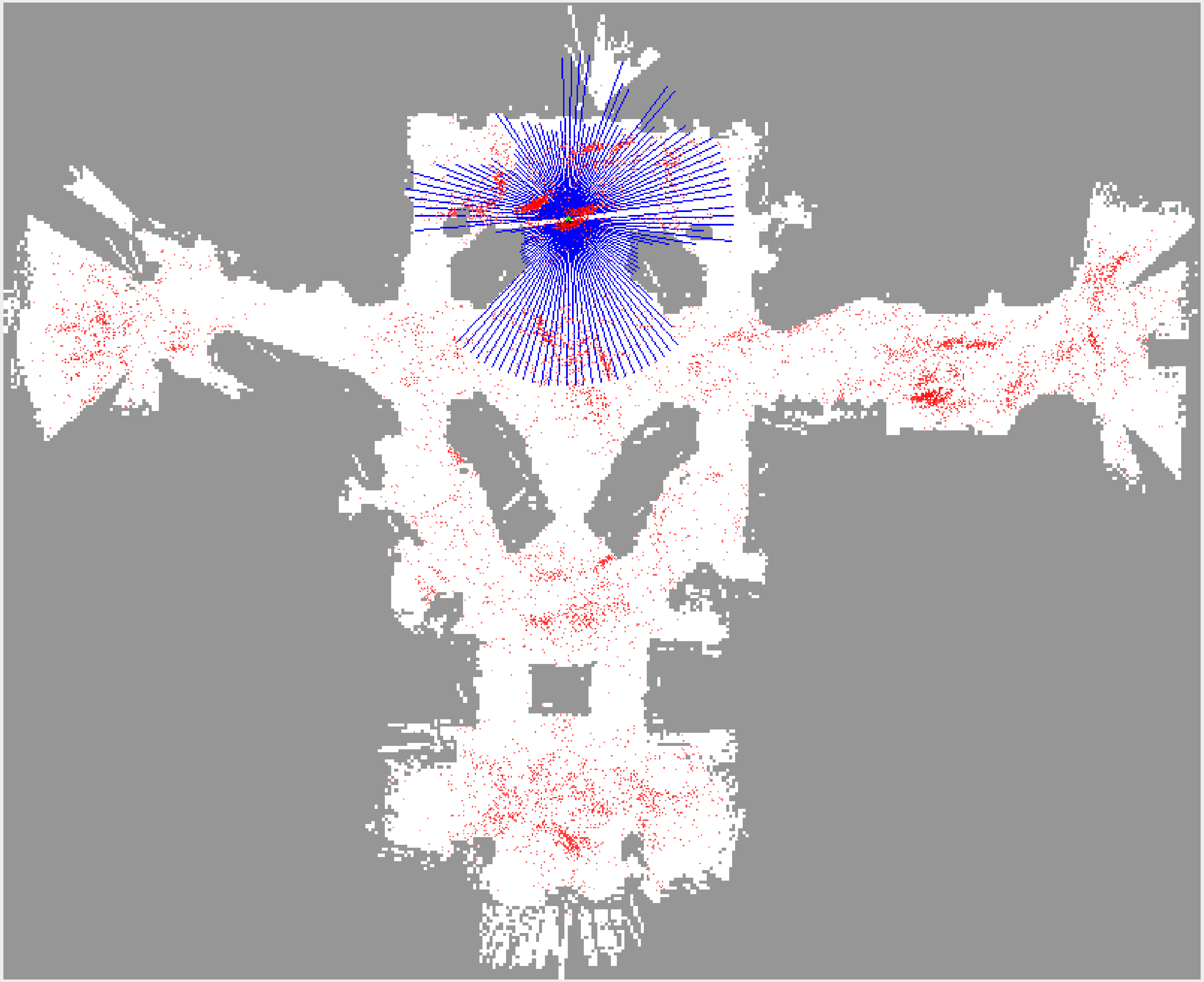
Particle Filter – Example (2)



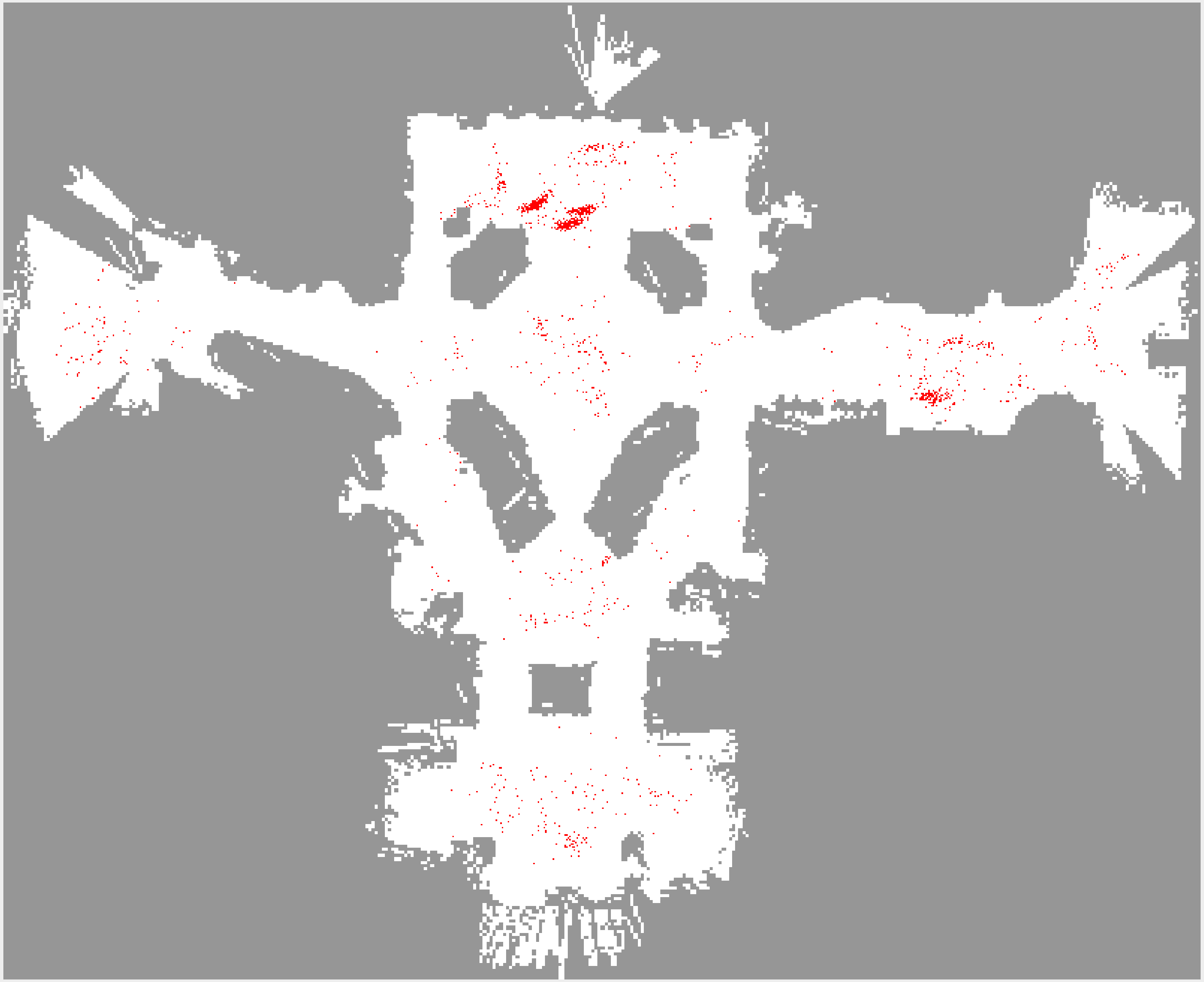
Particle Filter – Example (3)



Particle Filter – Example (4)

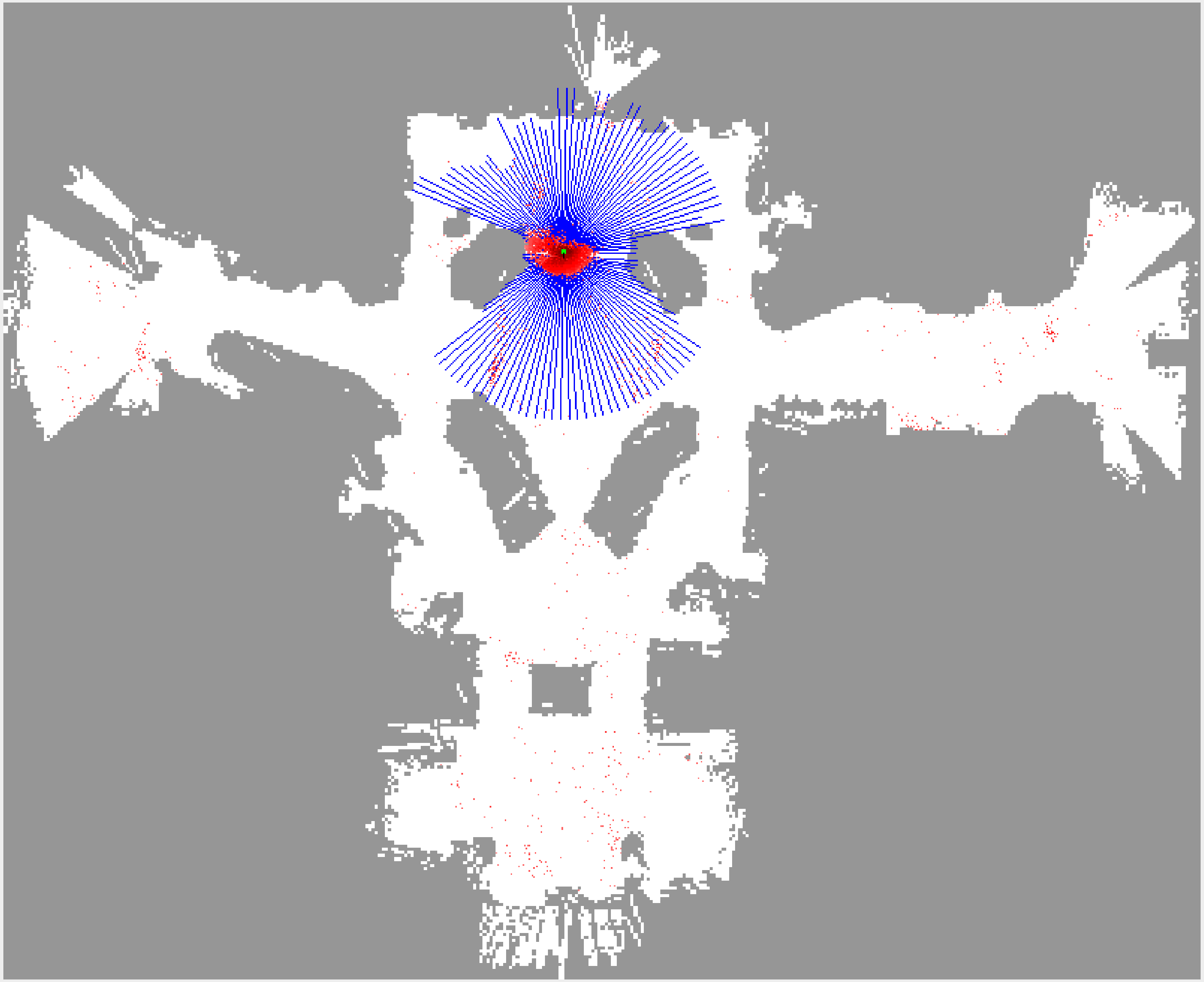


Particle Filter – Example (5)

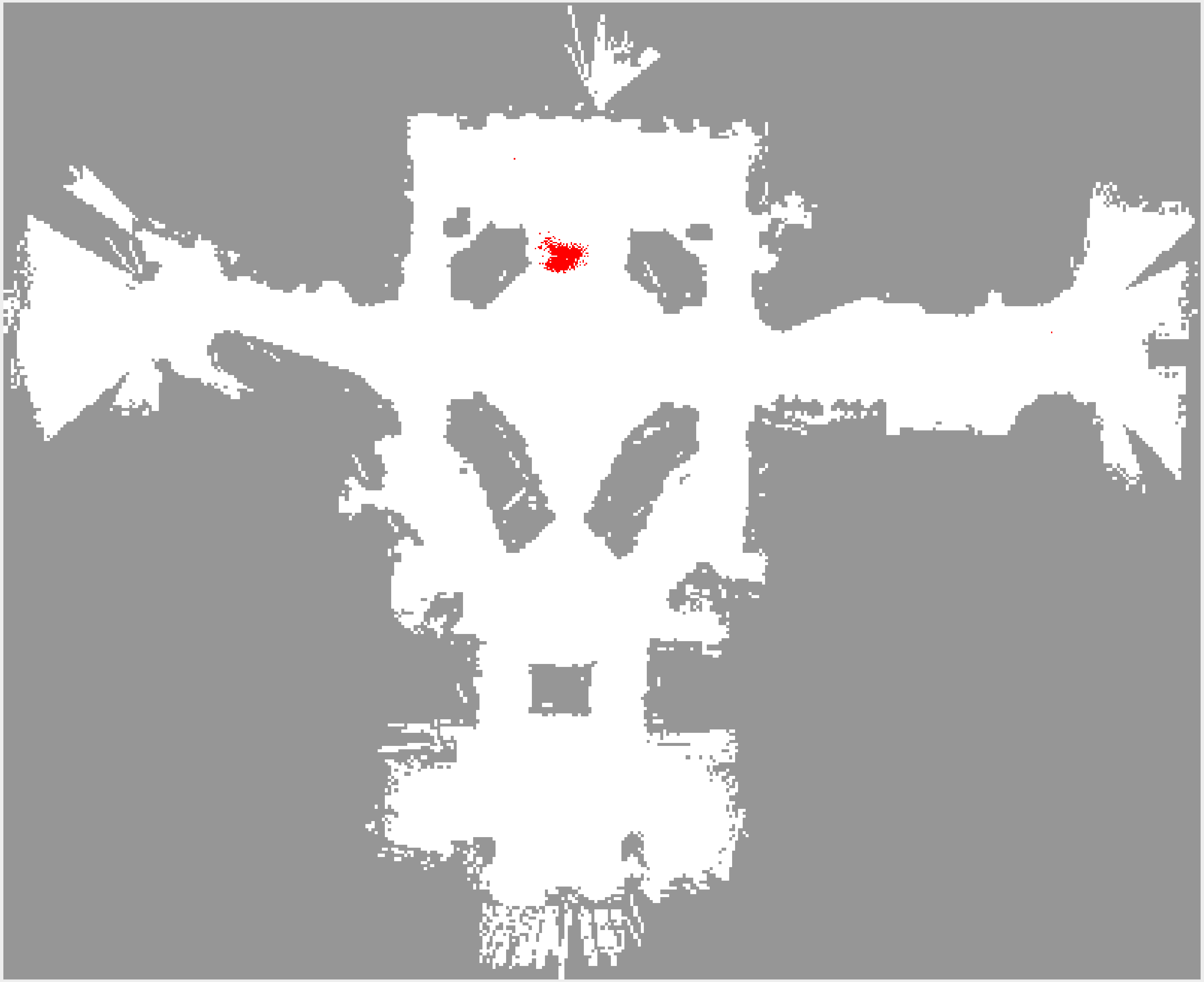


Particle Filter – Example (6)

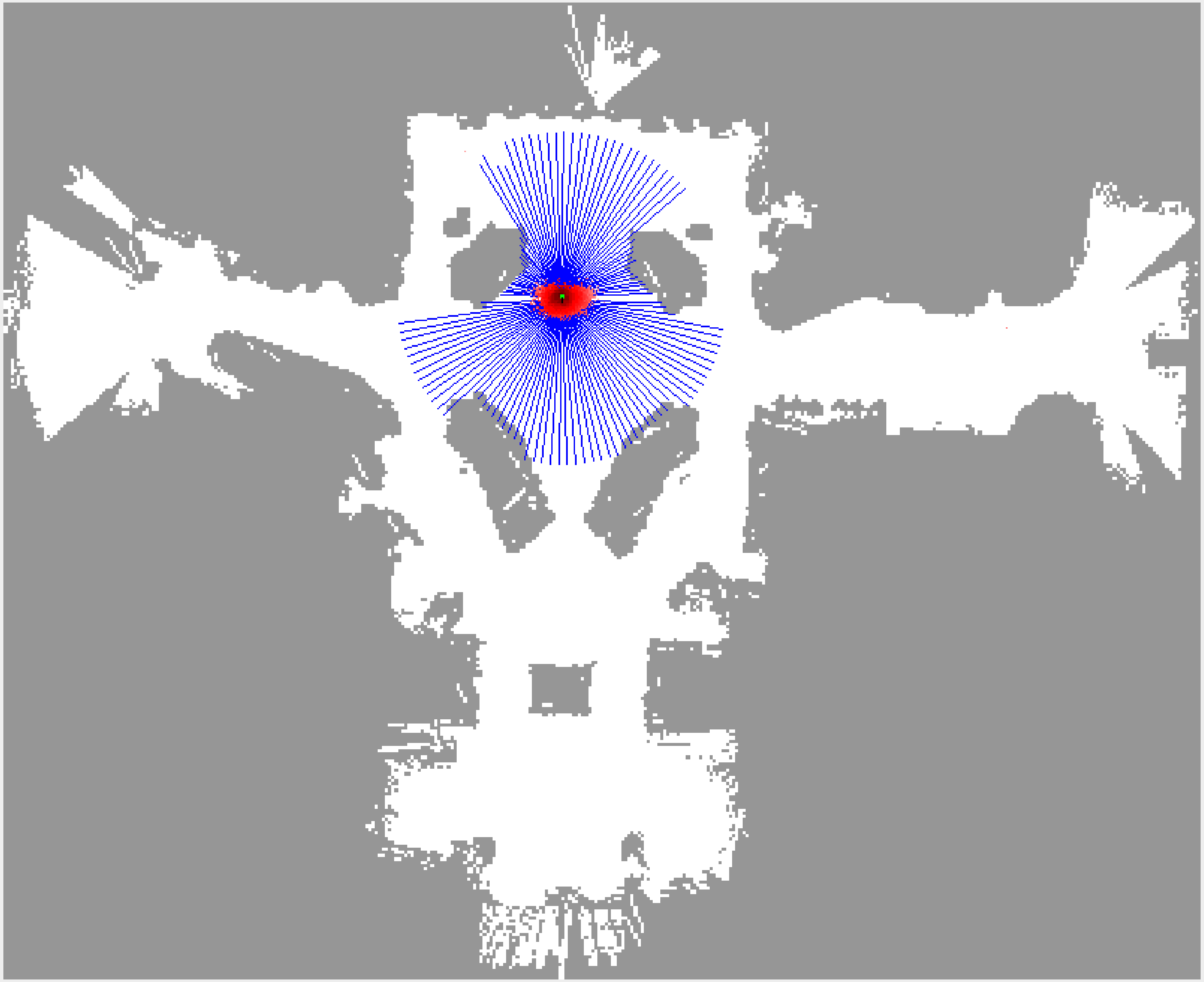
Particle Filter – Example (7)



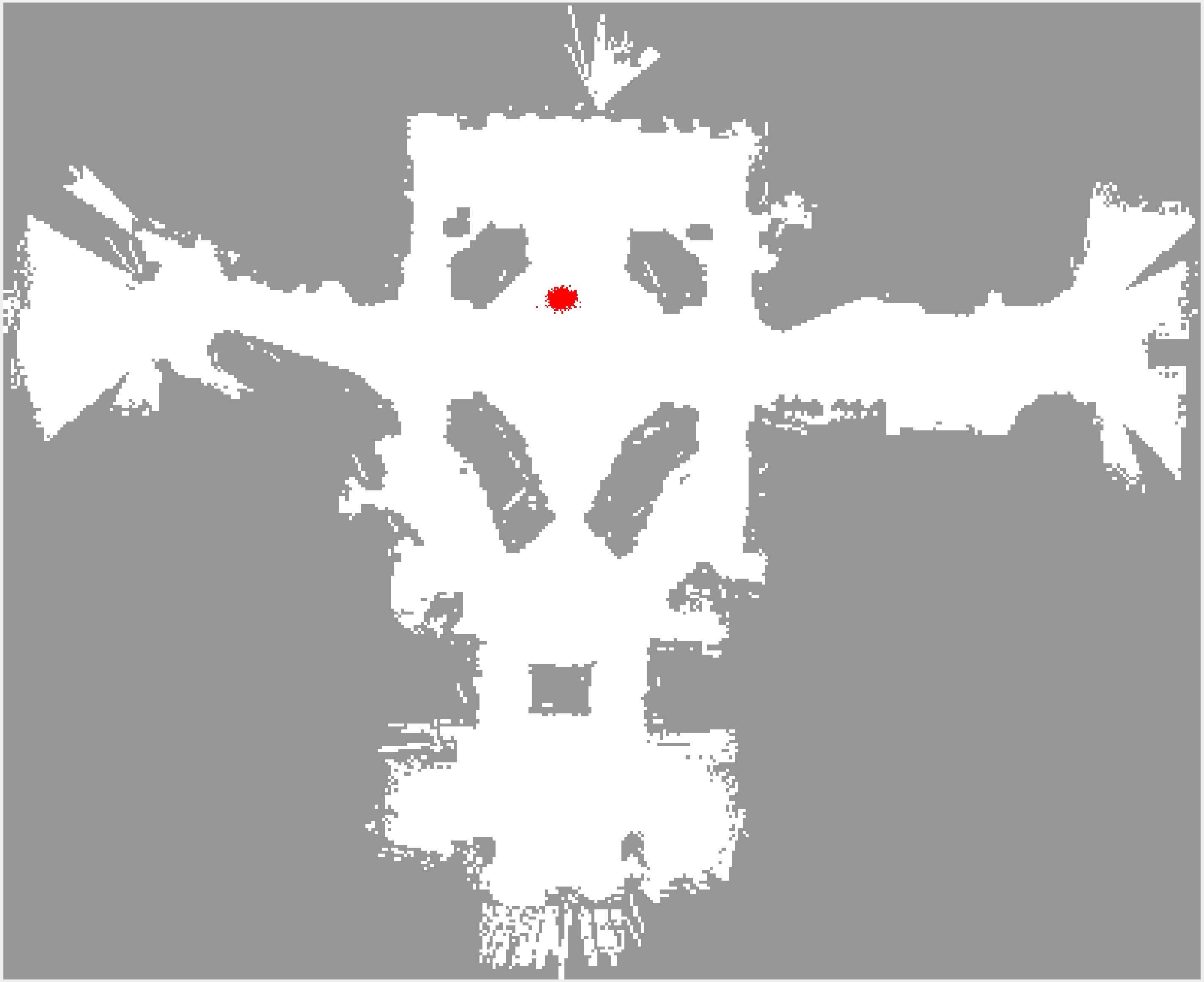
Particle Filter – Example (8)



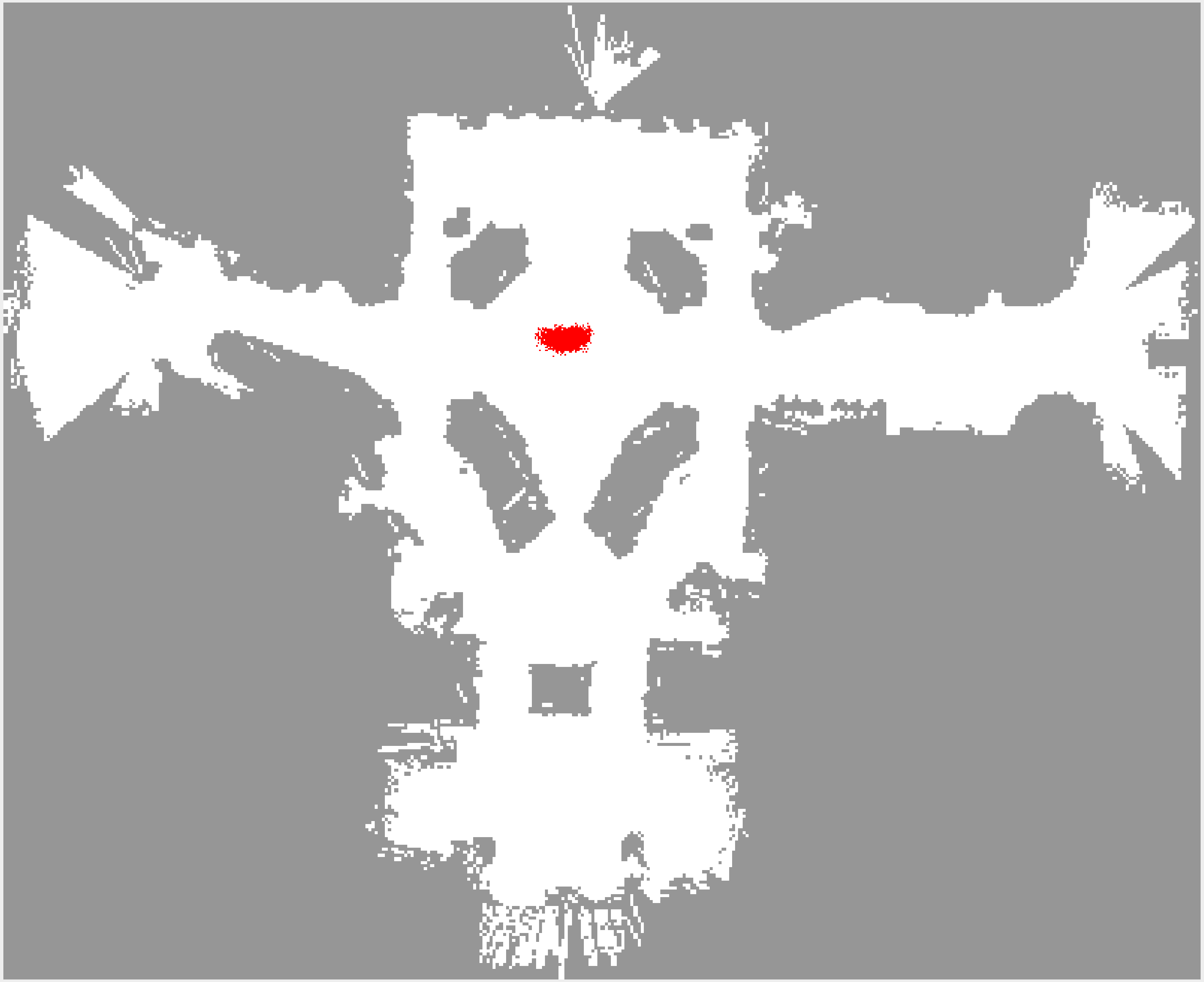
Particle Filter – Example (9)



Particle Filter – Example (10)



Particle Filter – Example (11)



Bayes-Filter

- Monte-Carlo Localization is a Bayes-Filter

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

draw x^i_{t-1} from $Bel(x_{t-1})$

draw x^i_t from $p(x_t \mid x^i_{t-1}, u_{t-1})$

Importance factor for x^i_t:

- Markov Localization is a Bayes-Filter

  Probalilities in discrete state space + computation of the state space

- Kalman-Filter is a Bayes-Filter

  Probabilities are Gaussians with mean and covariance

Drawing samples according to a distribution

- Is a classical problem in statistics
  („… telephone survey with 1001 representatively chosen housewives “)

Occurrence #1 in MCL:

- **static**: S, a vector of size N of samples, initially generated from P(X)

Ê Random sampling of size N from (arbitrary) probability distribution

Basic idea (theoretical):

- Calls of **random(.)** approximate uniform distribution over [0,1]
- Samples $S_{1,\ldots,N}$ uniform distributed [0,1] can be converted to any distribution **P**
  value in **P** of S_i is x_i where:

$$S_i = \int_{-\infty}^{x_i} \mathbf{P}(x)\,dx$$

Drawing samples practically (1)

Approximations T/B/F p.124
or Winkler 1995, Appendix A.4

- Start distribution (e.g. start pose) are usually „good-natured“: non-ambiguous value, or uniform distributed (≈ random), or Gaussian distributed, or combination of these

Changing a uniform distribution into a Gaussian one:

- Box-Mueller-Method (Theorem): If U_1, U_2 uniform distributed random variables over (0,1), then N_1, N_2 are Gaussian distributed variables, where
  $N_1 = (–2 \cdot \ln U_1)1/2 \cdot \cos(2\pi U_2)$ and
  $N_2 = (–2 \cdot \ln U_1)1/2 \cdot \sin(2\pi U_2)$
- N-Approximation: The following function approximates a normal distribution (mean 0, std-deviation b):

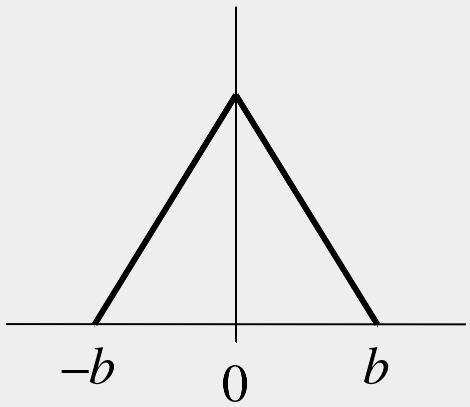
$$\frac{1}{2}\sum_{i=1}^{12} \mathtt{random}(-b,b)$$

Drawing samples practically (2)

Even simpler approximations

- Approximate b-triangle distribution by:

$$\frac{\sqrt{6}}{2}\left[\text{random}(-b,b)+\text{random}(-b,b)\right]$$



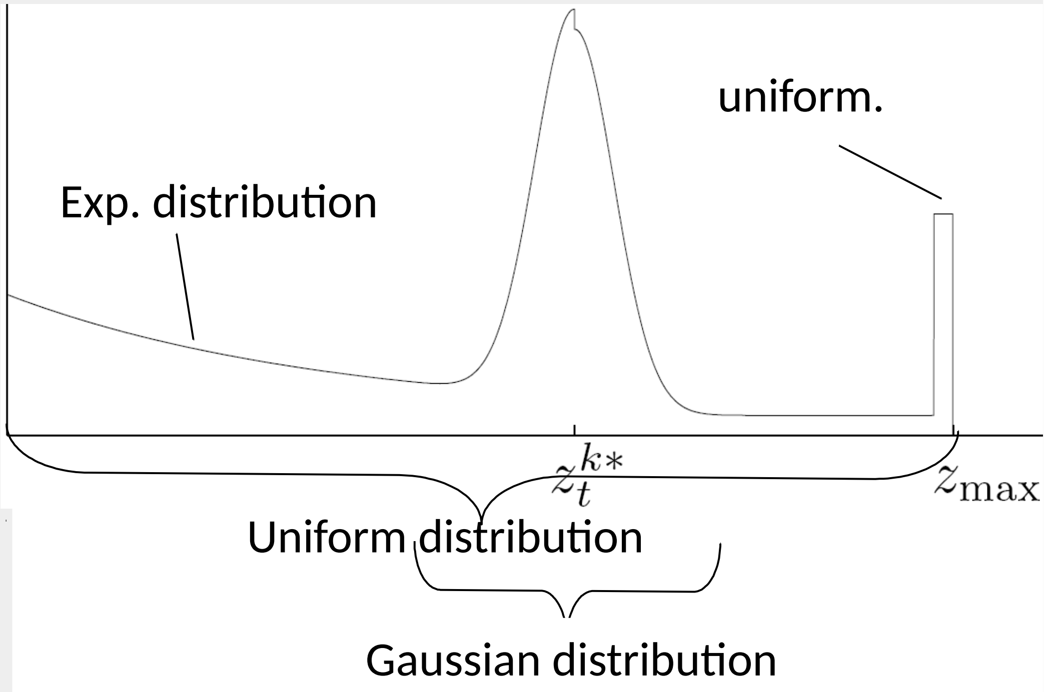
- Use existing libraries (e.g., C++ boost lib)

Mixture distributions

… represent them as superposition of basic distributions (*stratified sampling*)

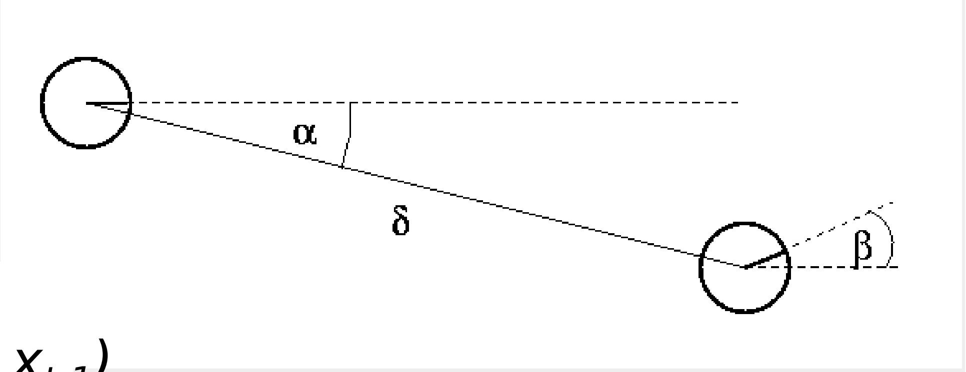
- Use measured distributions, i.e., histograms (cf. next slide)

Example: Error model laser

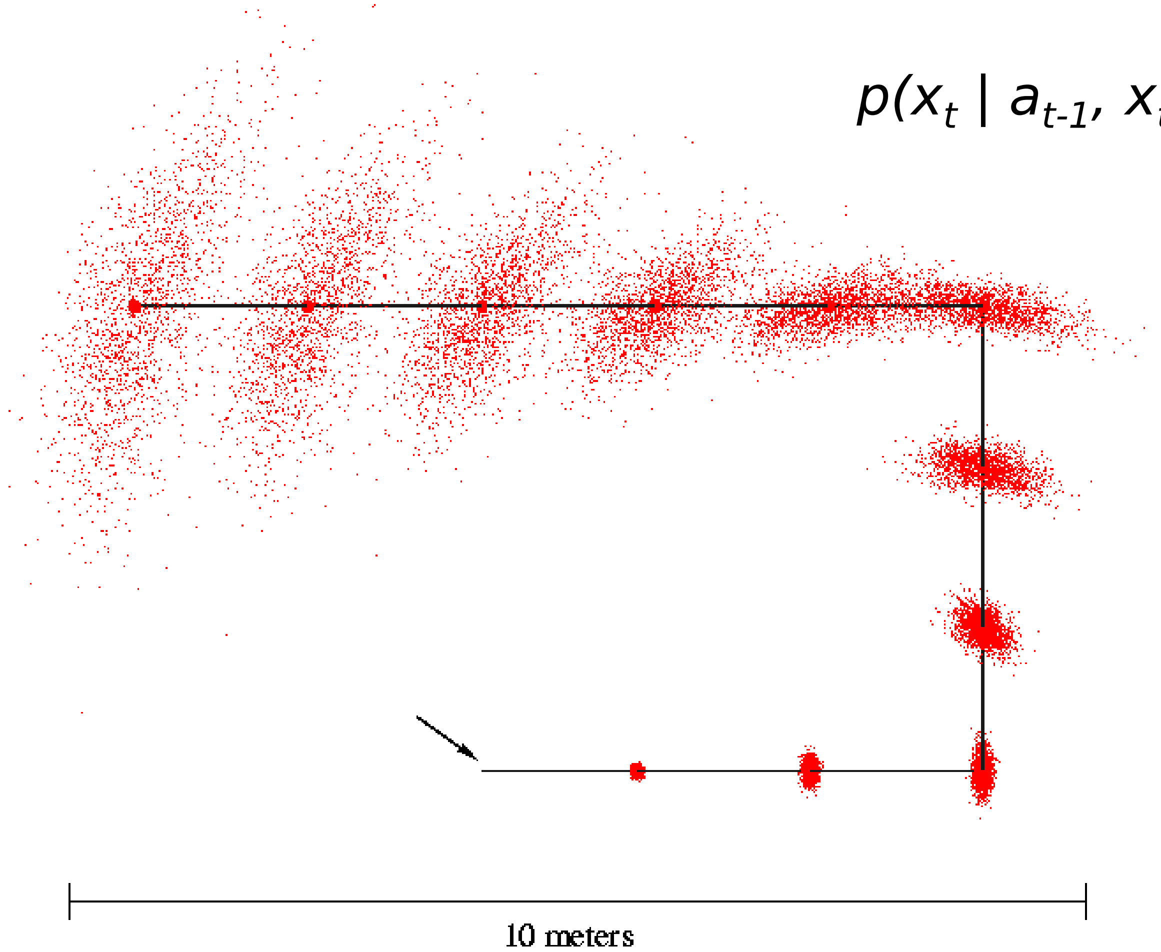


Monte Carlo Localization – Motion model

- Assumption:
  Odometry error with Normal-distribution in α, β and δ



$$p(x_t \mid a_{t-1}, x_{t-1})$$



Markov Localization – Motion model

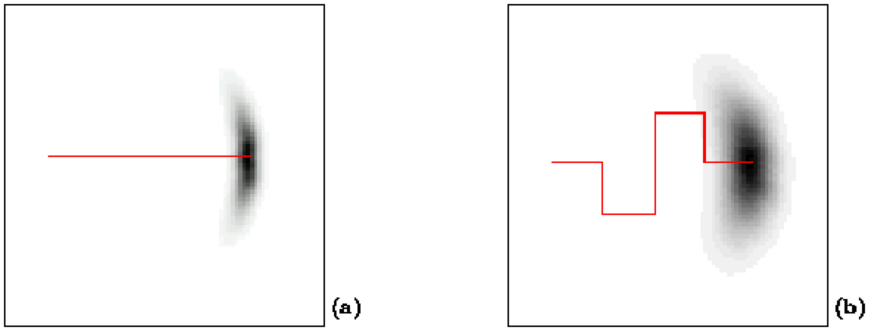


Fig. 3. Typical "banana-shaped" distributions resulting from different motion actions.

Drawing samples according to a distribution (2)

Occurrence #2 in MCL:

Sample x_t^i from $p(x_t \mid x_{t-1}, u_{t-1})$ using $x_{t-1}^{j(i)}$ and u_{t-1}

- Examine the 1(!) „most probable“ development of every particle according to the prediction
- Sampling according to distribution **P** as in #1 (e.g. draw 1(!) sample from Gaussian distribution)

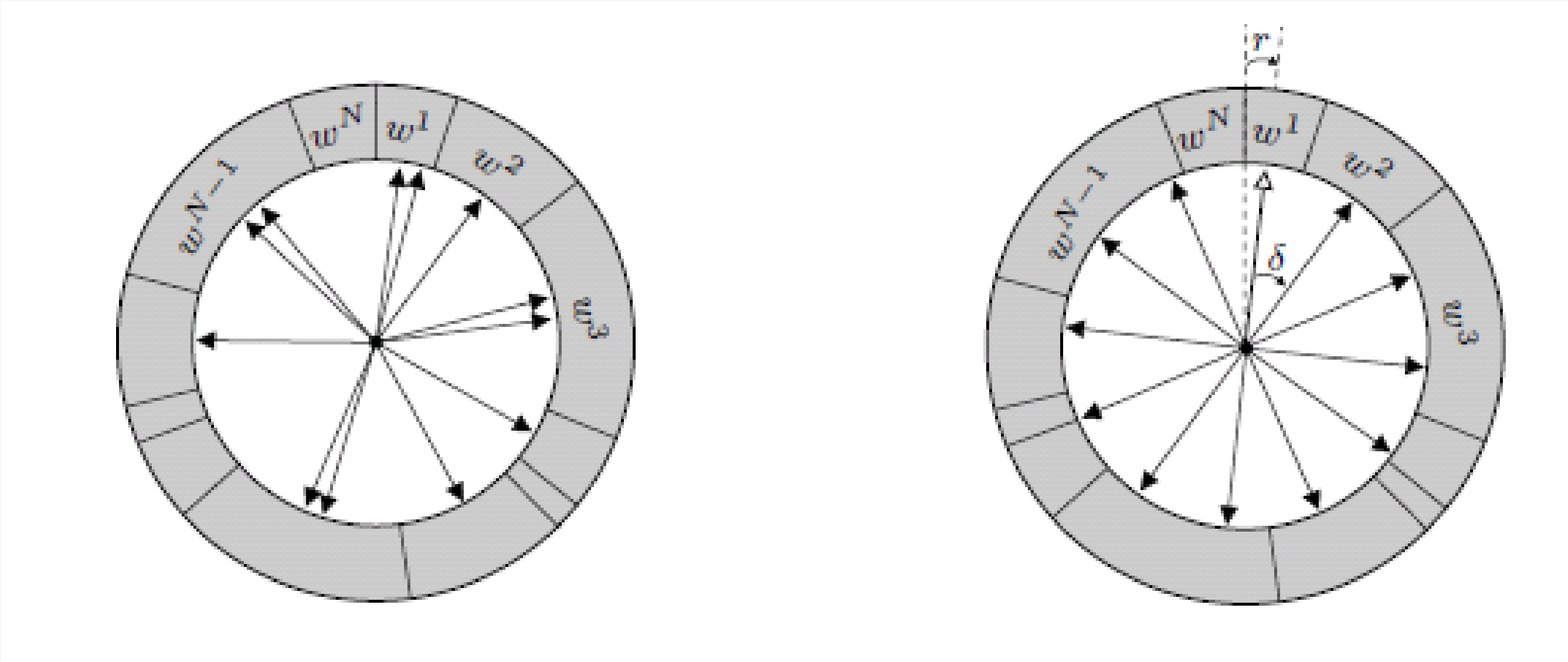
Occurrence #3 in MCL:

Compute importance weight $w_t^i = p(z_t \mid x_t^i)$

Ê extinction of „underweight“ particle
(also „not possible poses“ e.g. poses inside a wall);
removed particle are replaced according to distribution (e.g. close to particle with much weight or random)

Drawing samples practically (3)

- There exists an intuitive methods, i.e., the roulette wheel method.
- Samples are located on a circle their weight is represented by the size of the segment.
- The wheel is turned *n*-times and the resulting particle is chosen.



- The set of chosen particles approximates the distribution of samples.

Drawing samples practically (4)

- Left roulette wheel:

Input: Distribution $\mathcal{X}_t = \{\langle x^i, w^i \rangle\}_{i=1,\ldots,N}$

Output: Distribution $\mathcal{X}_{t+1}$ with the weights w^i

$r_{\max} = \sum_{i=1}^{N} w^i$
for $i = 1 \ldots N$ **do**
    $r = \text{random}(0, r_{\max})$
    $k = 0$
    $j = 0$
    while $k < r$ **do**
        $j = j + 1$
        $k = k + w^i$
    end while
    $\mathcal{X}_{t+1} = \mathcal{X}_{t+1} \cup \{\langle x^j, w^j \rangle\}$
end for
return $\mathcal{X}_{t+1}$

Advanced Automation

16

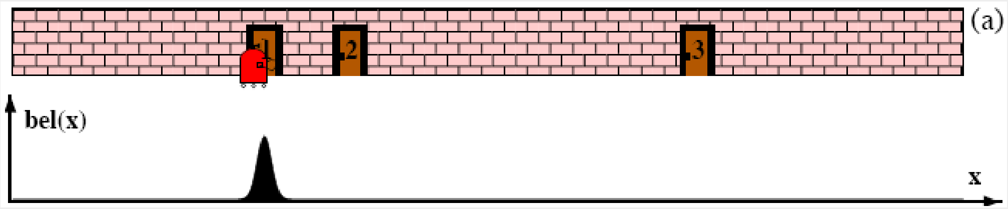
Prof. Dr. Andreas Nüchter
Robotics and Telematics
andreas@nuechti.de

Last Lectures: Mono modal pose distributions, -densities

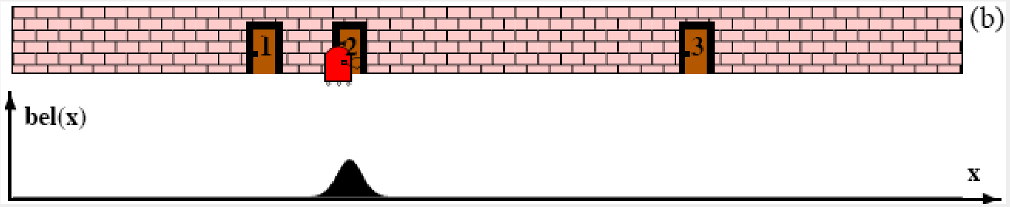
… similar to using an (extended) Kalman filter:

A single pose hypothesis is being tracked (with a variance).

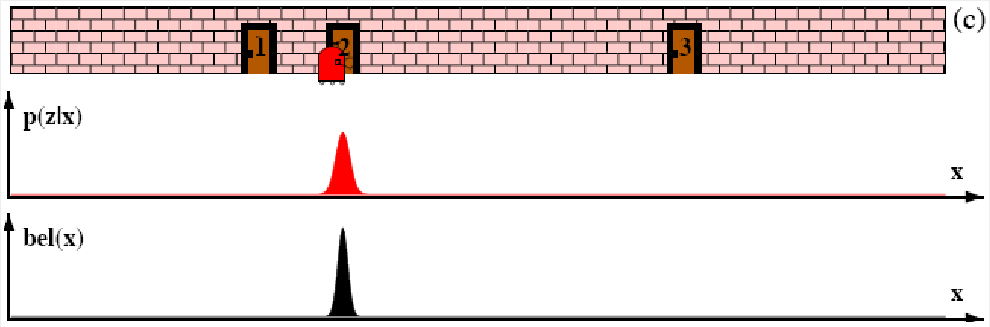
- Same case (*maximum likelihood*) using scan matching



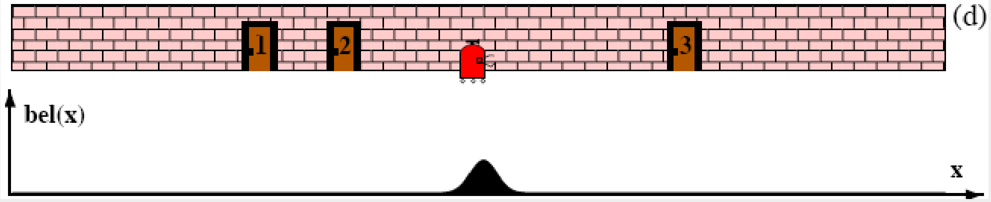
(a) start: Pose Gaussian distributed



(b) priori: motion model



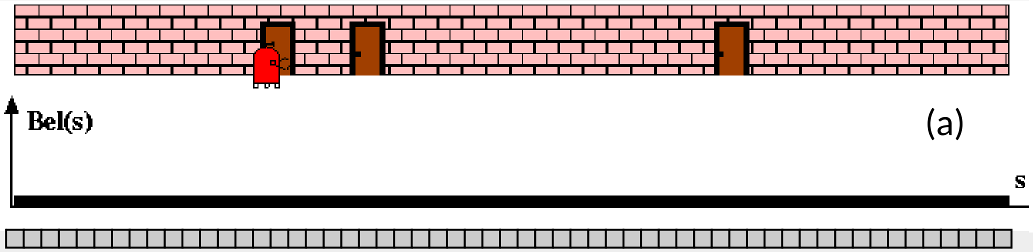
(c) posteriori: sensor model



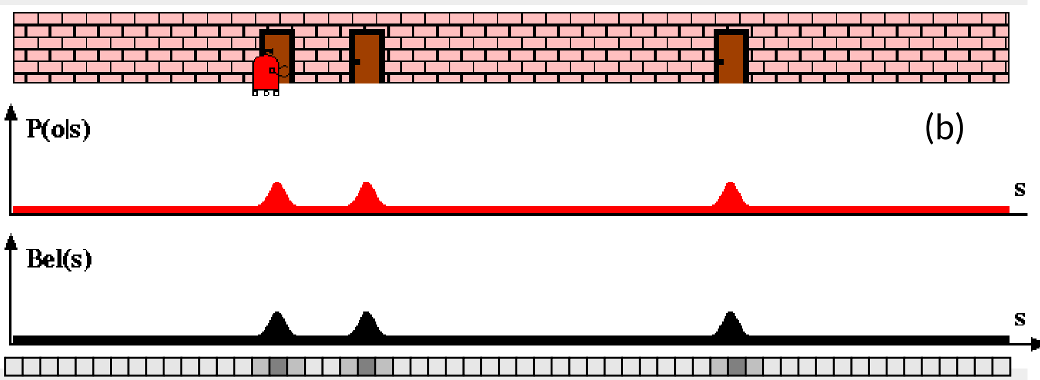
(d) priori: motion model … etc

Last Lecture: Markov Localization

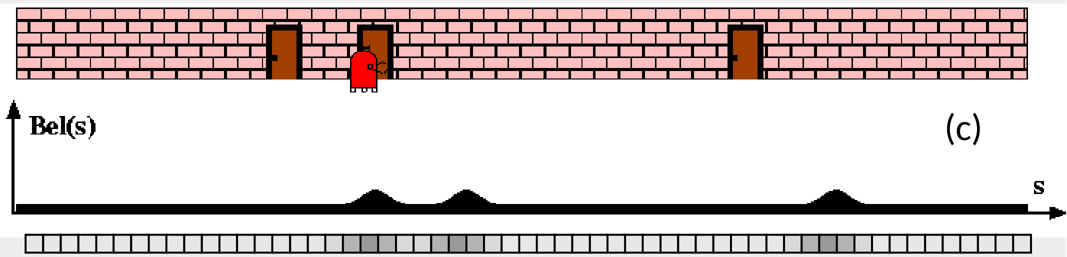
- State space S: 1D vector



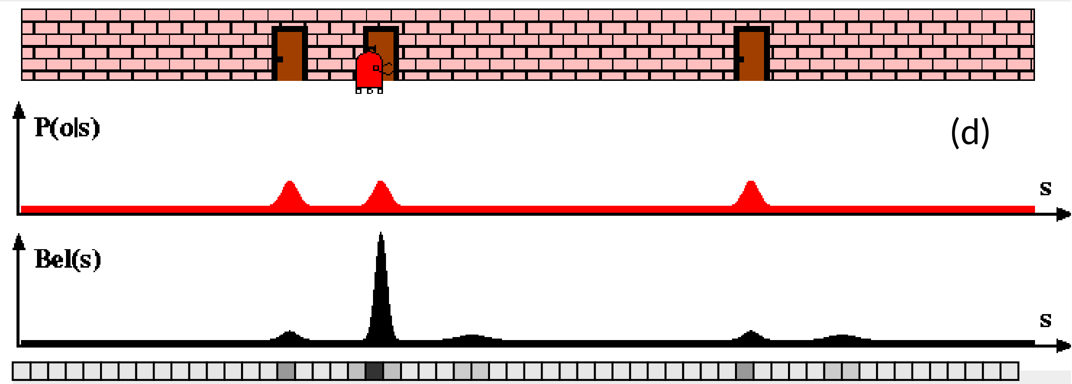
Start: No information, i.e., uniform distribution of possible poses *U*[*S*](*s*)

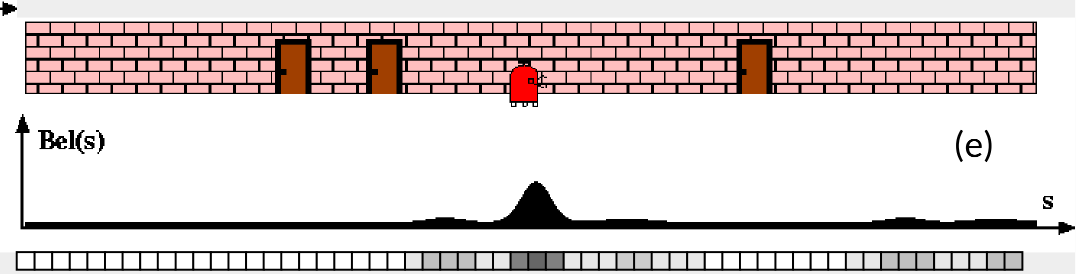


Observation: Door detected! Distribution accumulates at door



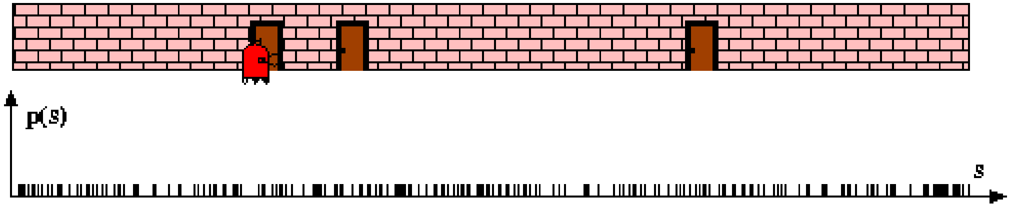
Motion: Distributions „proceed“ according to motion model



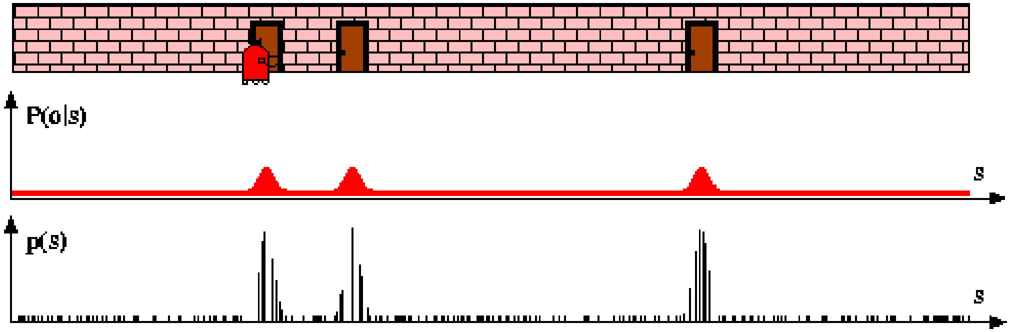


Last Lecture: Monte Carlo Localization

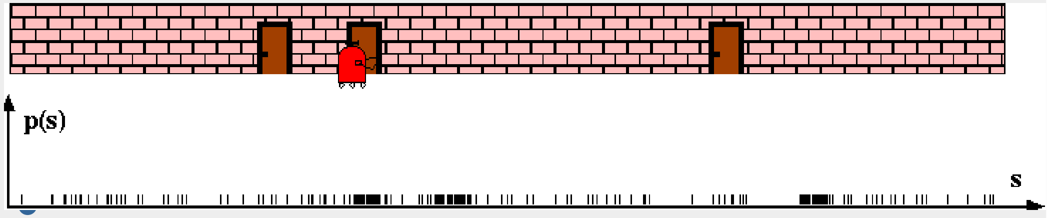
- Initialization:
  uniform distribution of particles representing the pdf



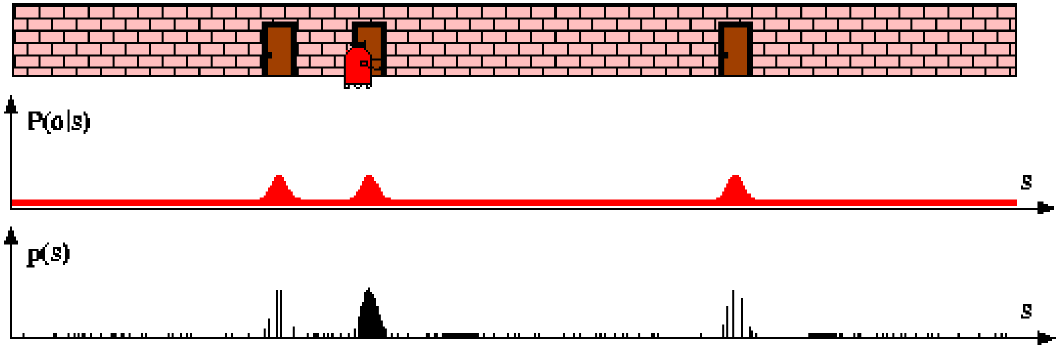
- Update with sensor values:
  new weights for particles



- Update with the robot motion:
  Resample the particles



- Update with sensor values:
  new weights for particles



Quality of pose estimation using MCL

Definition: Entropy of a distribution **P**(*X*) of variable X:

$$H(X) := -\sum_{i=1}^{|V|} P(x_i)\log_2 P(x_i)$$

… has a maximal value for uniform distribution

$$H_{\max}(X) = -\sum_{i=1}^{|X|} \tfrac{1}{|X|}\log_2 \tfrac{1}{|X|} = -\sum \tfrac{1}{|X|}\left(\log_2 1 - \log_2|X|\right) = \log_2|X|$$

(Localization: Poses are equally distributed, Robot has no clue where it is)

… is minimal if information is known, i.e., 0 (because log(1)=0)

$$H(S) := -\sum_s b(s)\frac{\log_2 b(s)}{\log_2|S|} = -\sum_s b(s)\log_{|S|} b(s)$$

… normalized to the interval [0,1] for state *S*

- Quality measure for current belief
- *H*(*S*)=0 ➷ accurate knowledge about the pose
  H(*S*)=1 ➷ absolute lack of knowledge

How many samples are needed

(Video Sonar)

(Video Laser)

More about particle filter
Dieter Fox, U. Washington
www.cs.washington.edu/ai/Mobile_Robotics/mcl/

Depends…

- If the set of possible poses
  - is small → *N* small
  - is large → *N* large
- Number of landmarks in the environment:
  - many → *N* small
  - few → *N* large
- How good the pose is currently knows (variable!):
  - $H(S)$ close to 0 → *N* small
  - $H(S)$ close to 1 → *N* large
- For navigation in buildings N«10.000
- For small area (RoboCup!) N«1.000

- KLD-Sampling adapts *N*

Classic Example of Monte Carlo Localization

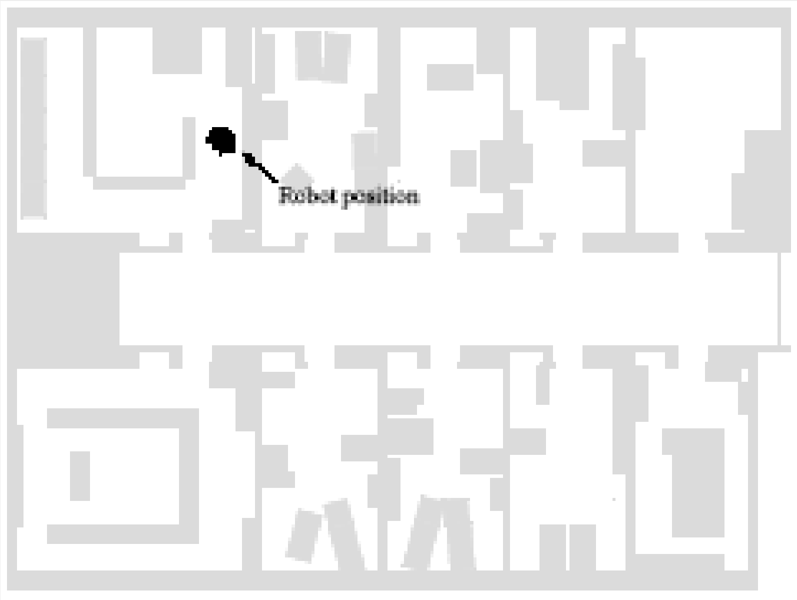
Samples uniform distributed (start)

Rhino data set
Bonn, Römerstr. 164, 1. floor
right part, Univ., Informatik III
[Fox & al., 1999]

~5000 samples

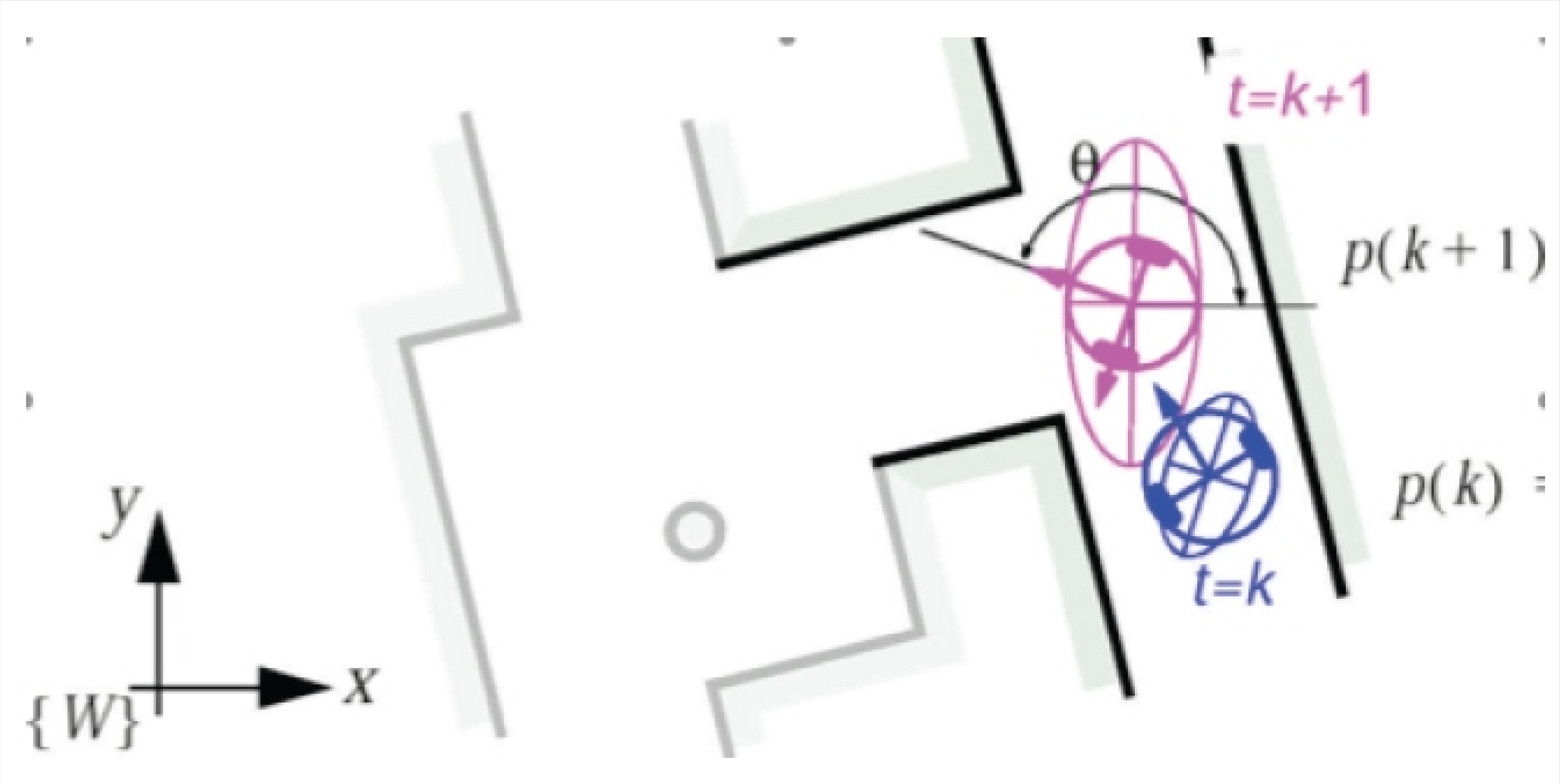
Samples after 1 meter driving

Samples after 3 meter



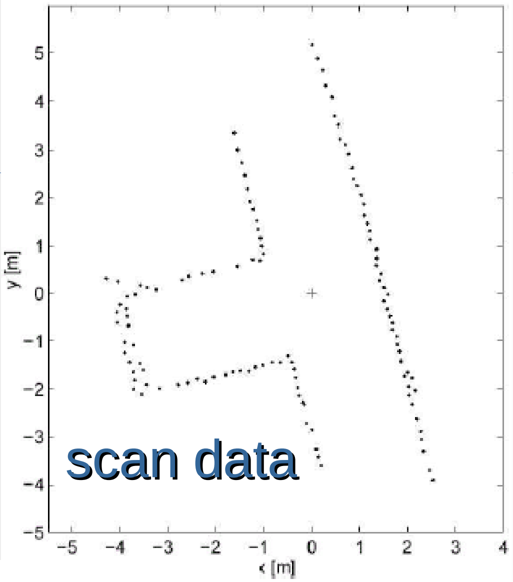
Kalman localization in Maps (1)

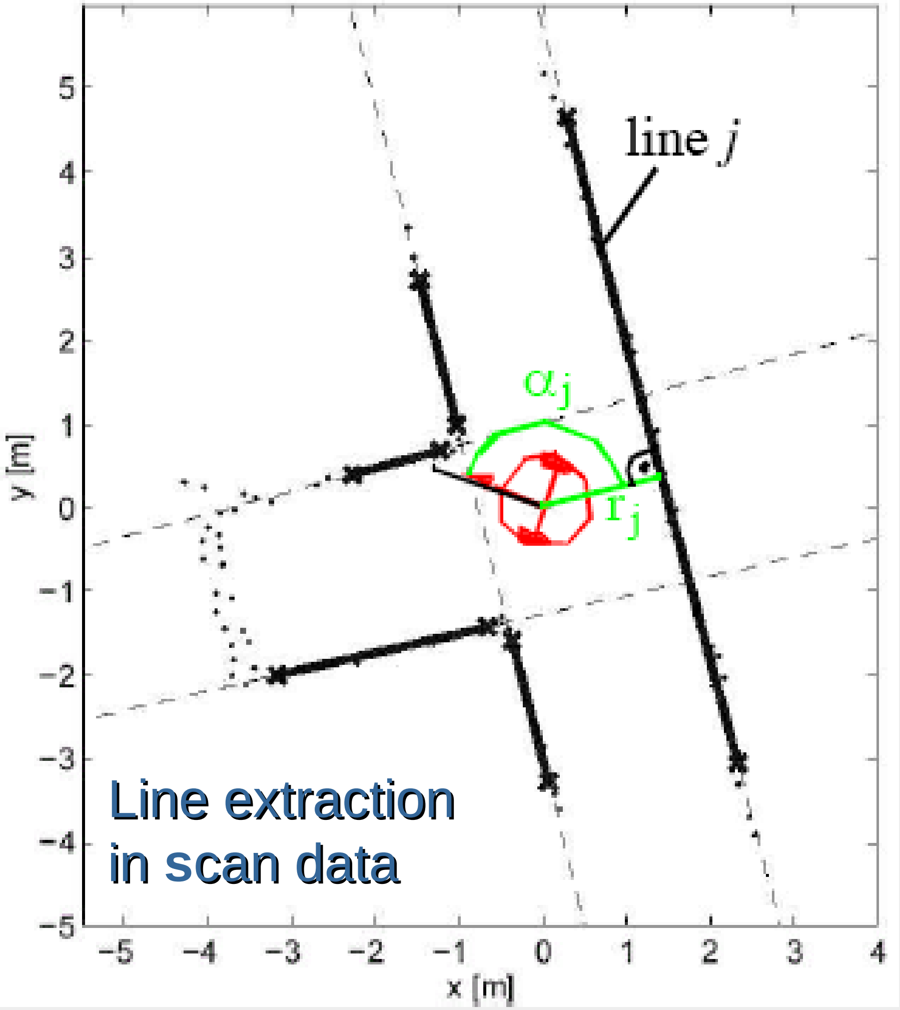
- Pose estimation according to odometry model
- a-priori position at t=k+1

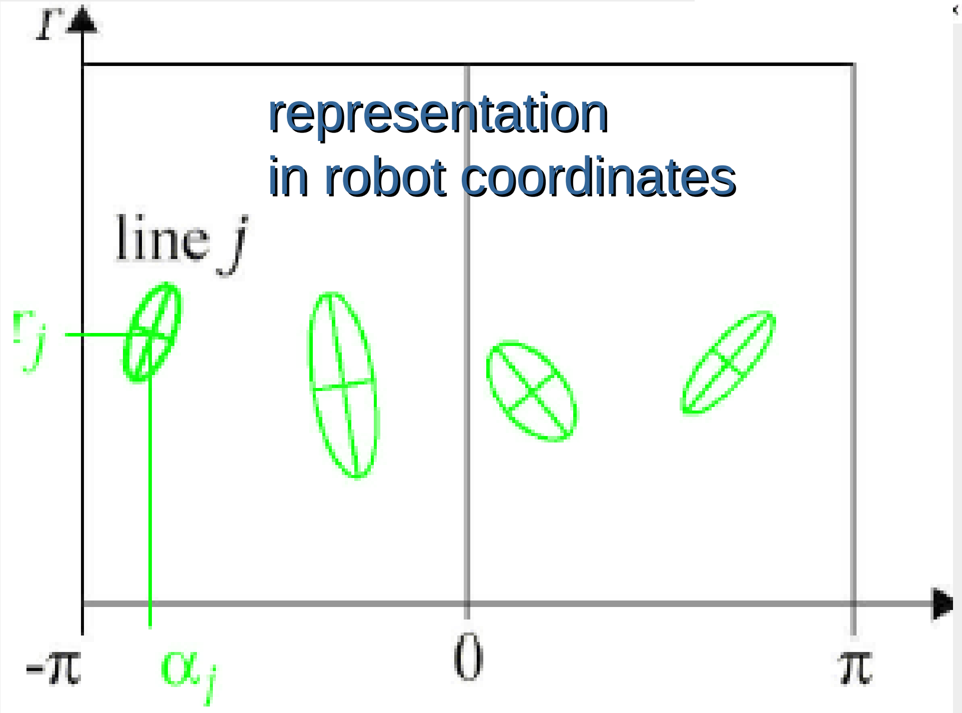


Kalman localization in Maps (2)

Transformation extracted Lines in pose estimate

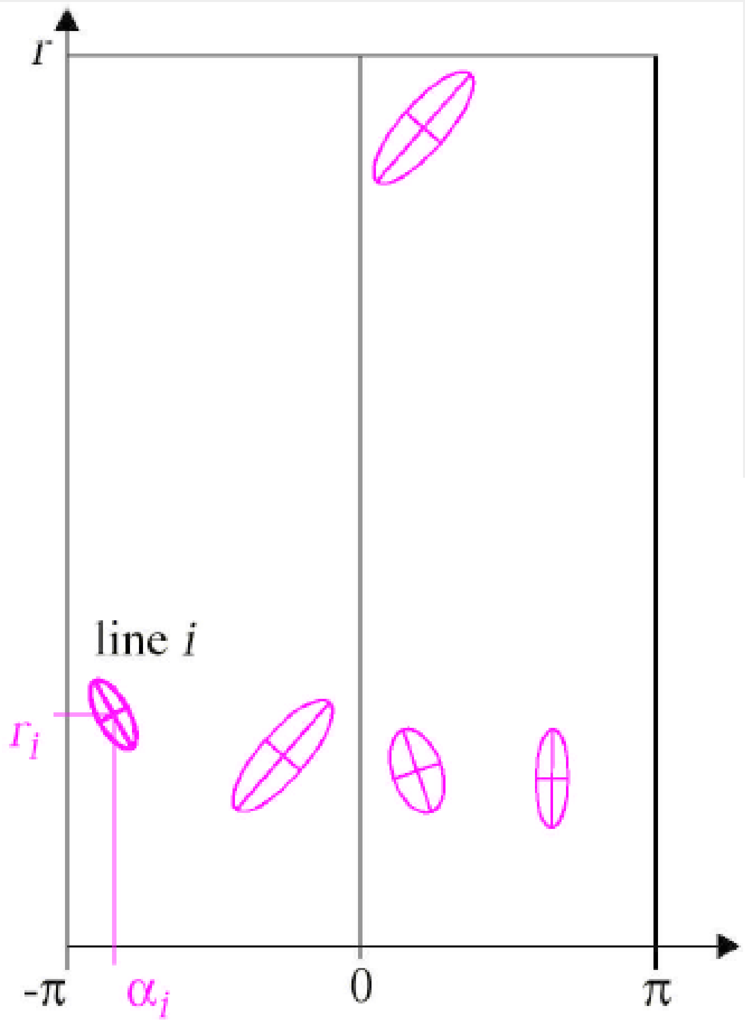




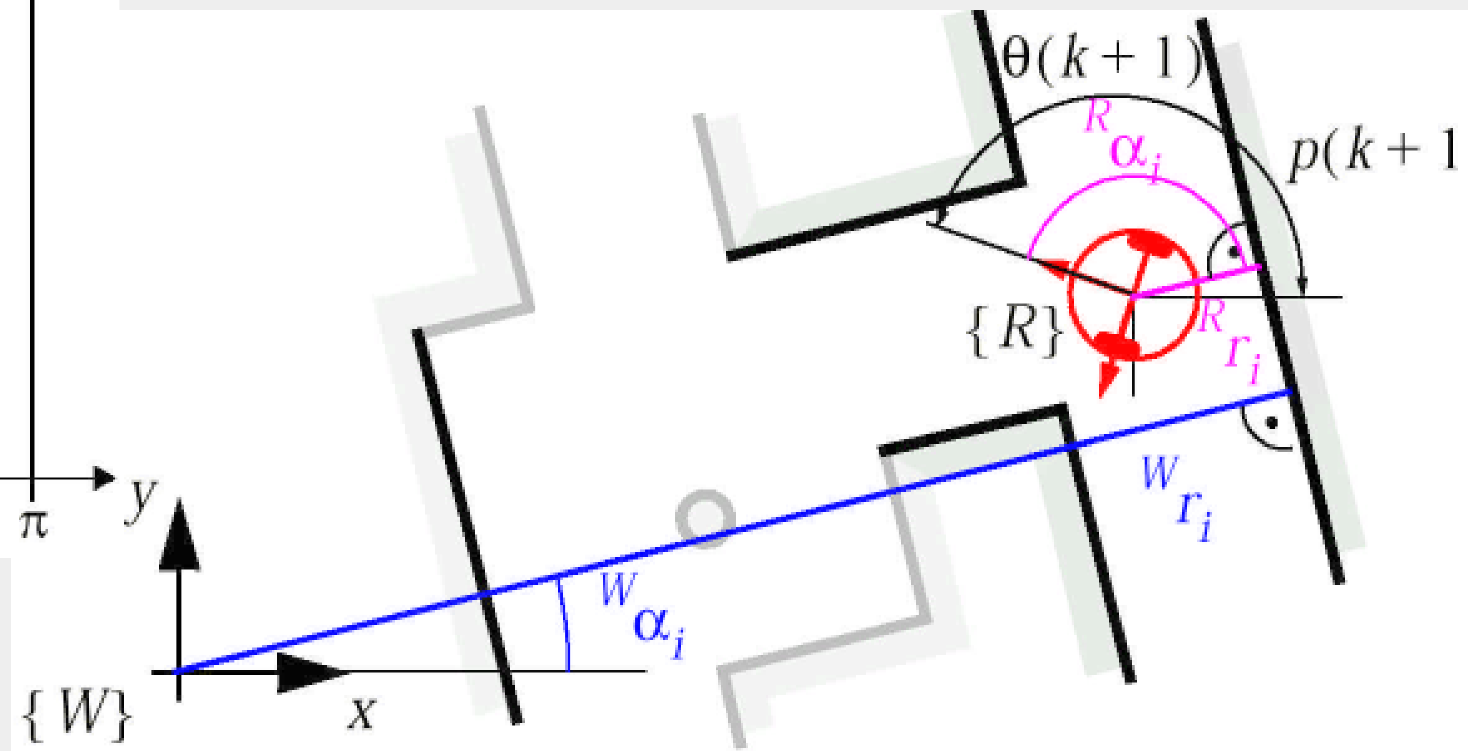


Mean and variance of the norm of the angle and the normal of the robot coordinate.

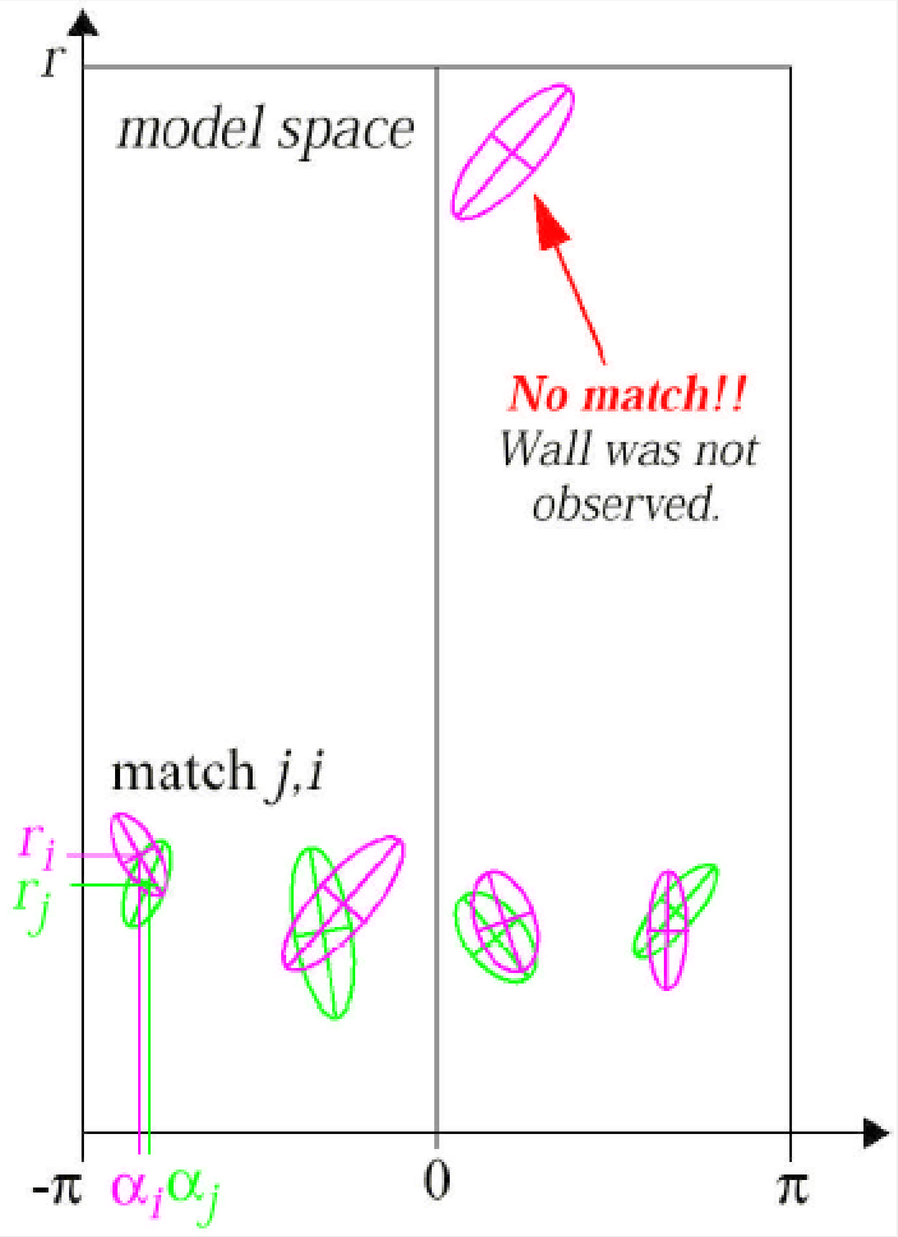
Kalman localization in Maps (3)



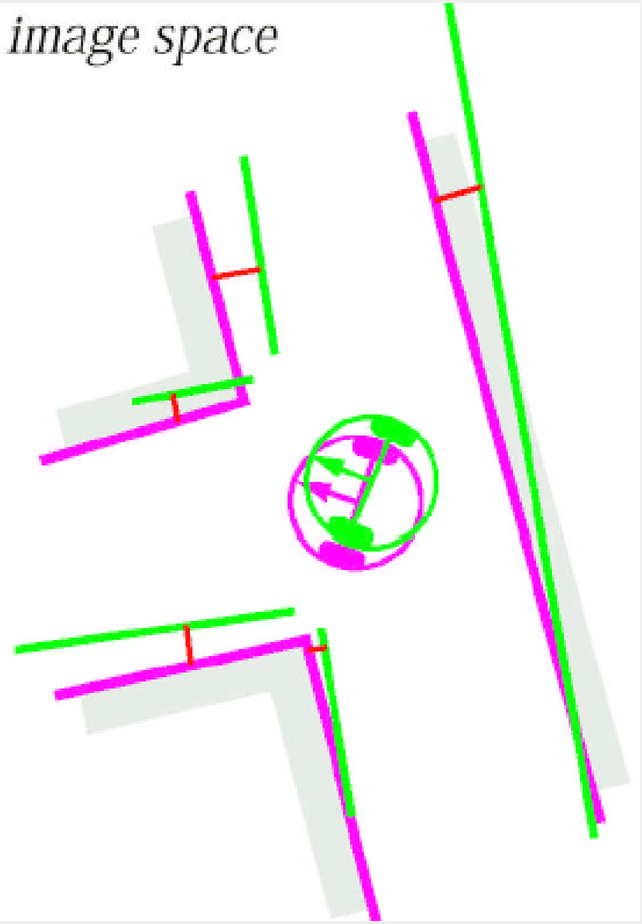
Expected line measurements according to the map



Kalman localization in Maps (4)

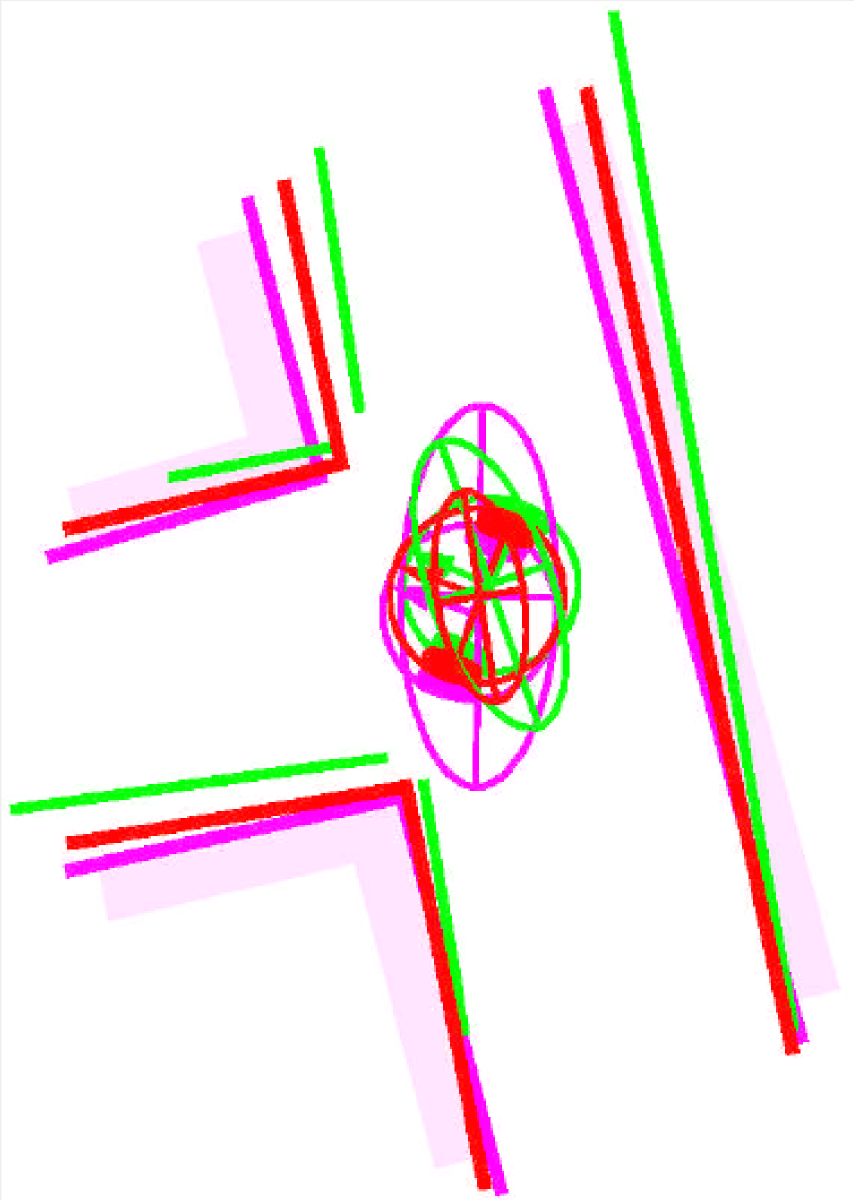


matching



Kalman localization in Maps (5)

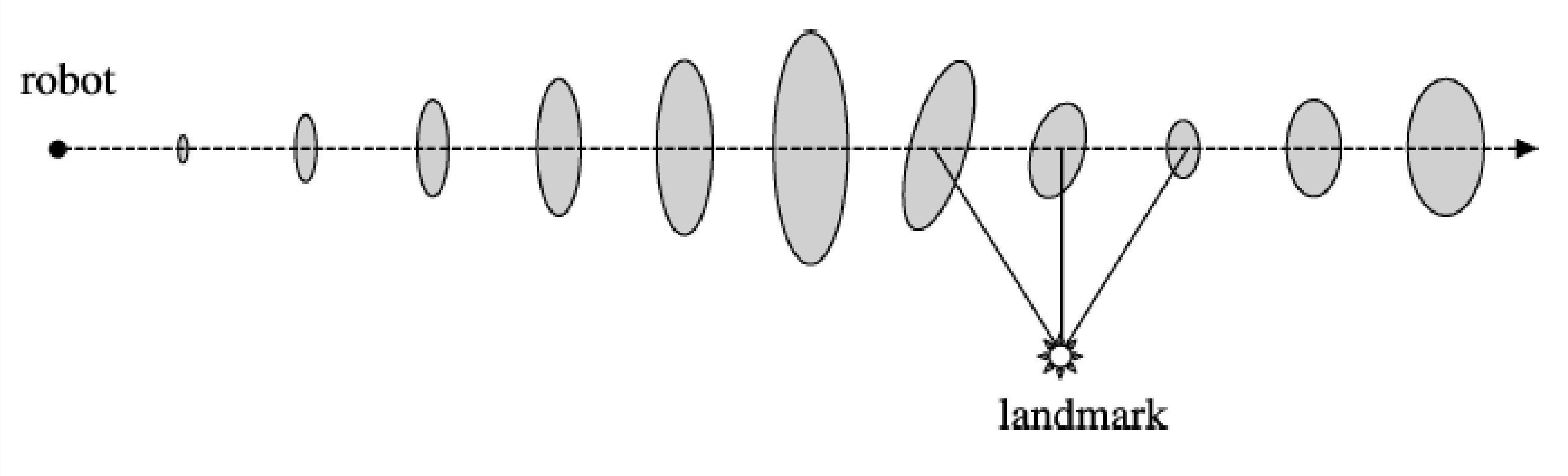
A-posteriori pose estimate



The red „robot ellipse“ shows new pose estimate with (x,z)-mean and variance in the red map

EKF Localization in a Map

- Example of localization using the extended Kalman filter. The robot moves on a straight line. As it progresses, its uncertainty increases gradually, as illustrated by the error ellipses. When it observes a landmark with known position, the uncertainty is reduced.



Disadvantages of single hypothesis localization

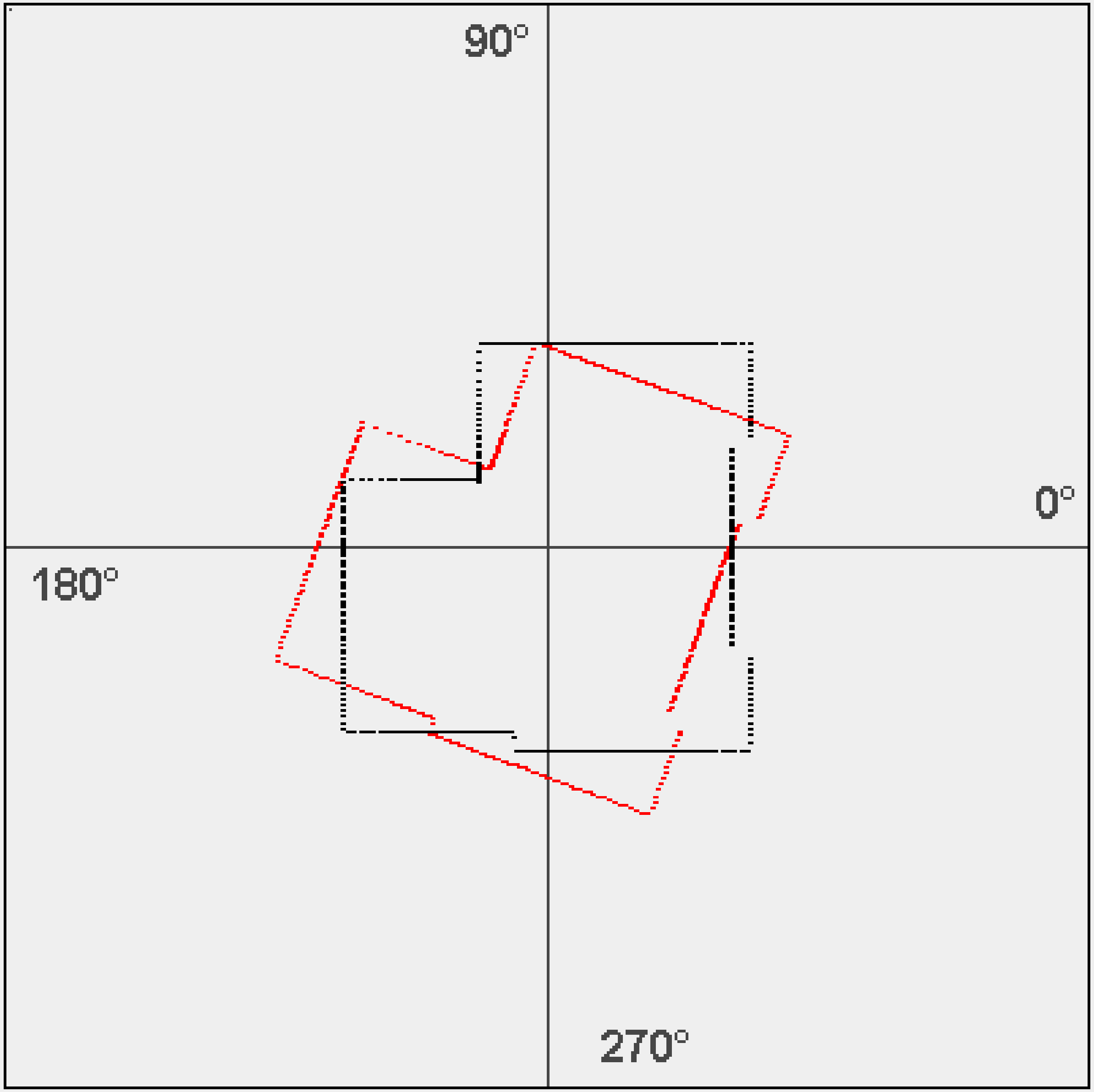
- Starting pose must be known (tracking of **one** pose)
- Acute danger of being lost, if
  - Drive through areas without features („open space“)
  - Local errors in maps, including dynamic effects
  - Pose change through external effects („*kidnapped robot*“)
- There is only a very small chance to recover from a lost robot pose
- Global information could be used to recover the pose

Remember the Minerva-video!

Summery Localization Algorithms (5)

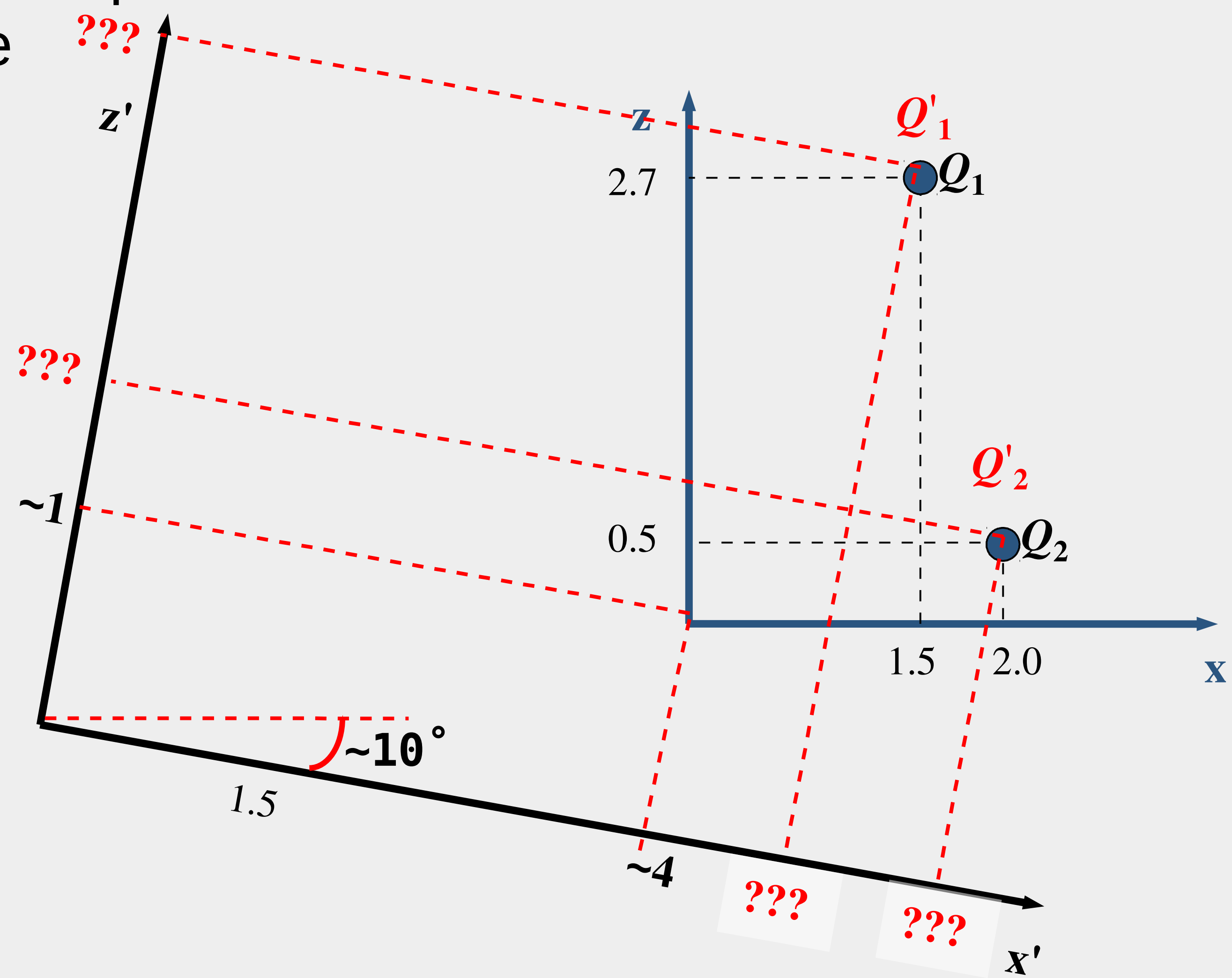
	Kalman Filter	Tracking many Hypothesis – not discussed here	Grid-based	Topological maps – not discussed here	Particle-Filter
Sensors					
Posterior					
Efficiency (memory)					
Efficiency (time)					
Implementation					
Accuracy					
Robustness					
Global localization					

Localization through 2D scan matching



Localization through 2D scan matching

- Example: After a pose change of about $[4, 1, 10^\circ]^T$ (odometry estimation) there are two scanned points Q_1 and Q_2 (local coordinate system).
- If we know from a previous pose the location of the same points as Q'_1, Q'_2, we could improve the pose estimation!



a) Find / guess corresponding points in space from two scans

b) Compute the pose change based on the rotation and translation of these scan points

Repetition: Rotation matrices (1)

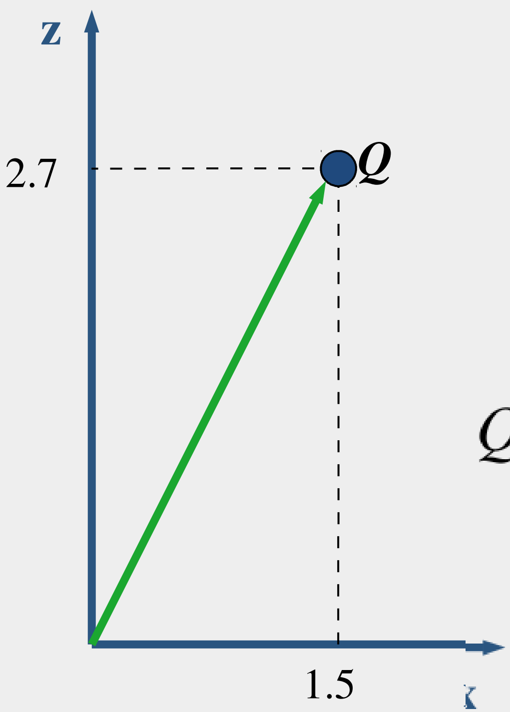
… we know from analytical geometry:

- Rotation in a plane about angle ω:

$$R_\omega = \begin{bmatrix} \cos\omega & -\sin\omega \\ \sin\omega & \cos\omega \end{bmatrix}$$

- $$Q' = R_\omega \cdot Q$$

Rotates vectors about ω counterclockwise, i.e. rotates the reference frame clockwise.

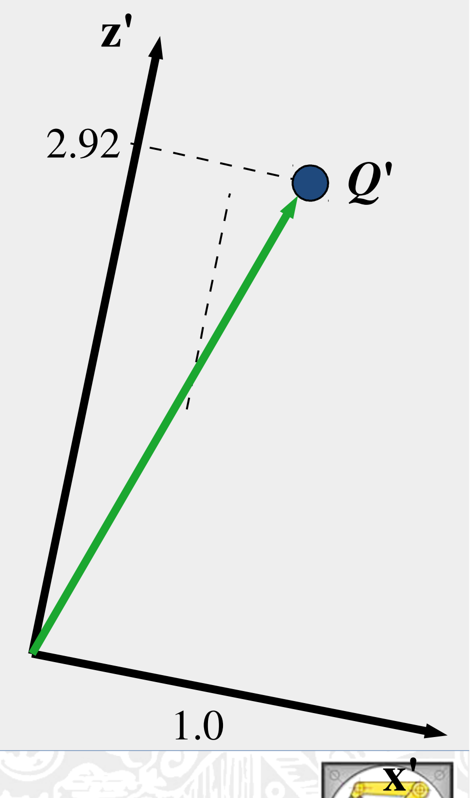


Example:

Rotation of a vector Q of about ω=10°, i.e., the frame of reference of ω=–10°

$$Q' = \begin{bmatrix} \cos 10° & -\sin 10° \\ \sin 10° & \cos 10° \end{bmatrix} \cdot \begin{bmatrix} 1.5 \\ 2.7 \end{bmatrix}$$

$$= \begin{bmatrix} 0.9848 & -0.1736 \\ 0.1736 & 0.9848 \end{bmatrix} \cdot \begin{bmatrix} 1.5 \\ 2.7 \end{bmatrix} = \begin{bmatrix} 1.0084 \\ 2.9195 \end{bmatrix}$$



Repetition: Rotation matrices (2)

- Properties of a rotation matrix

$$\mathbf{RR}^T = I$$

$$\det \mathbf{R} = 1.$$

- **R** is normalized: the squares of the elements in any row or column sum to 1
- **R** is orthogonal: the dot product of any pair of rows or any pair of columns is 0.
- The rows of **R** represent the coordinates in the original space of unit vectors along the coordinate axes of the rotated space.
- The columns of **R** represent the coordinates in the rotated space of unit vectors along the axes of the original space.

$$R_{Xrot} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_{Yrot} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_{Zrot} = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 & 0 \\ \sin\varphi & \cos\varphi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- In 3D a rotation matrix **R** is computed as follows:

$$\mathcal{M}(\alpha,\beta,\gamma) = \begin{bmatrix} \cos\alpha\cos\gamma - \cos\beta\sin\alpha\sin\gamma & -\cos\beta\cos\gamma\sin\alpha - \cos\alpha\sin\gamma & \sin\alpha\sin\beta \\ \cos\gamma\sin\alpha + \cos\alpha\cos\beta\sin\gamma & \cos\alpha\cos\beta\cos\gamma - \sin\alpha\sin\gamma & -\cos\alpha\sin\beta \\ \sin\beta\sin\gamma & \cos\gamma\sin\beta & \cos\beta \end{bmatrix}$$

Scan matching via minimization of error functions

If

- We have *n pairs* (P_i, P_i') corresponding points in space as scanned points and
- all P_i, P_i' are precise scan values,

then

- The minimum (namely the zero point) of the following error function of the pose change $[T_x, T_z, \omega]^T$ gives a precise pose estimate:

$$E_{\text{dist}}(\omega, T) = \sum_{i=1}^{n} \left\| R_\omega \cdot P_i + T - P_i' \right\|^2$$

But …

– Both preconditions can practically not by satisfied!

– Minimum ≠0 of E_{dist} estimates pose change

– Use odometry based pose estimation as start value of an iterative gradient descent over corresponding points.

Scan matching error function minimization for (ω, T)

- Due to Lu/Milios 1993 we know the following theorem:

  There exists a closed form solution in (ω, T) for minimization of $E_{dist}(\omega, T)$ given the point correspondences, namely

$$\omega = \arctan \frac{S_{xz'} - S_{zx'}}{S_{xx'} + S_{zz'}}$$

$$T_x = \bar{x}' - (\bar{x}\cos\omega - \bar{z}\sin\omega)$$

$$T_z = \bar{z}' - (\bar{x}\sin\omega + \bar{z}\cos\omega)$$

Using:

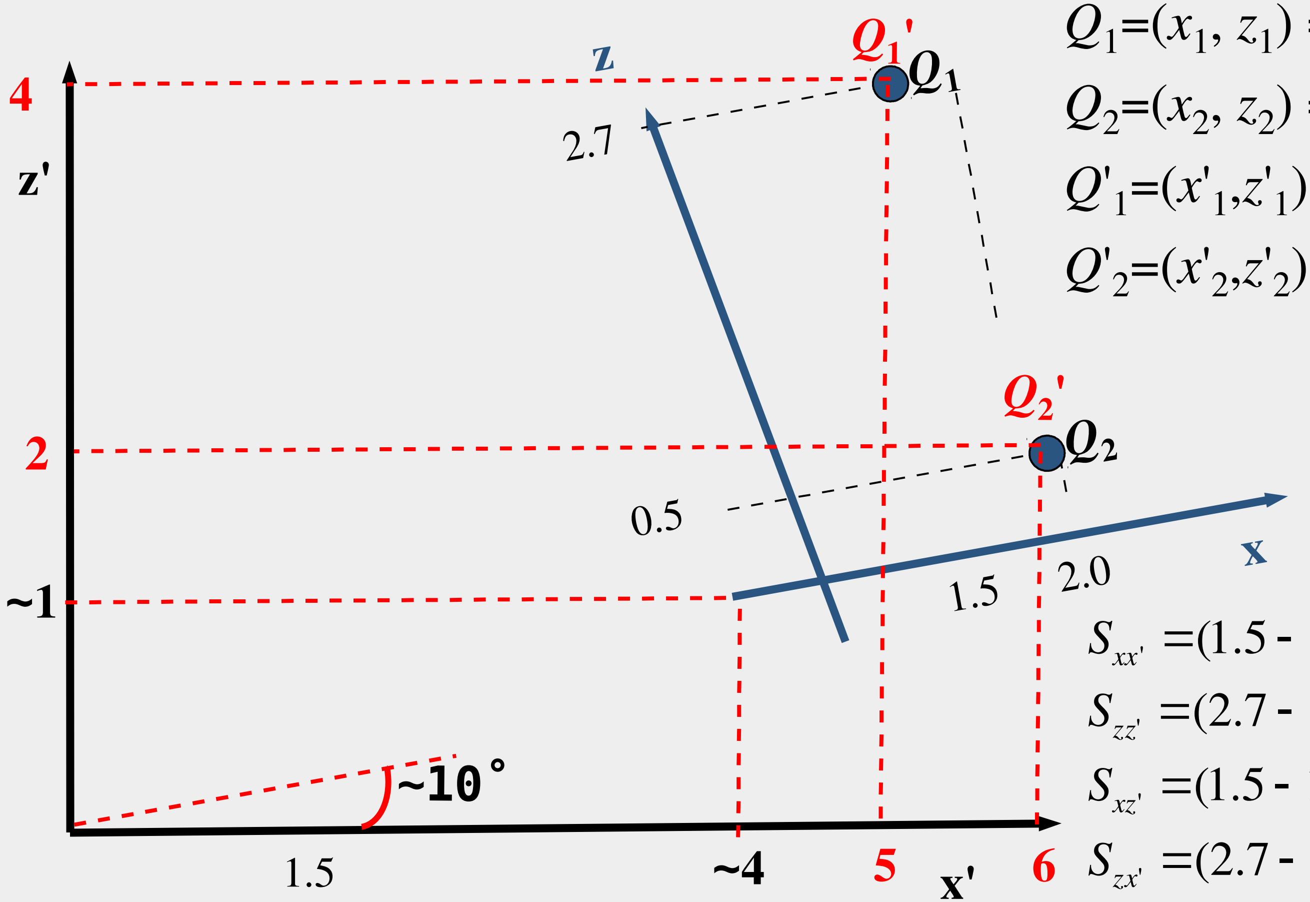
$$\bar{x} = \frac{1}{n}\sum_i x_i \qquad S_{xx'} = \sum_i (x_i - \bar{x})(x'_i - \bar{x}')$$

$$\bar{x}' = \frac{1}{n}\sum_i x'_i \qquad S_{xz'} = \sum_i (x_i - \bar{x})(z'_i - \bar{z}')$$

$$\bar{z} = \frac{1}{n}\sum_i z_i \qquad S_{zx'} = \sum_i (z_i - \bar{z})(x'_i - \bar{x}')$$

$$\bar{z}' = \frac{1}{n}\sum_i z'_i \qquad S_{zz'} = \sum_i (z_i - \bar{z})(z'_i - \bar{z}')$$

Example: Error function minimization for (ω, T)



$Q_1=(x_1, z_1) = (1.5, 2.7)$

$Q_2=(x_2, z_2) = (2, 0.5)$

$Q'_1=(x'_1,z'_1)= (5, 4)$

$Q'_2=(x'_2,z'_2)= (6, 2)$

$$\bar{x} = \tfrac{1}{2}(1.5+2)=1.75$$

$$\bar{z} = 1.6$$

$$\bar{x}' = 5.5$$

$$\bar{z}' = 3$$

$$S_{xx'} = (1.5-1.75)(5-5.5)+(2-1.75)(6-5.5) = 0.25$$

$$S_{zz'} = (2.7-1.6)(4-3)+(0.5-1.6)(2-3) = 2.2$$

$$S_{xz'} = (1.5-1.75)(4-3)+(2-1.75)(2-3) = -0.5$$

$$S_{zx'} = (2.7-1.6)(5-5.5)+(0.5-1.6)(6-5.5) = -1.1$$

$$\omega = \arctan\frac{S_{xz'}-S_{zx'}}{S_{xx'}+S_{zz'}} = \arctan\frac{0.6}{2.45} = 13{,}7608^\circ$$

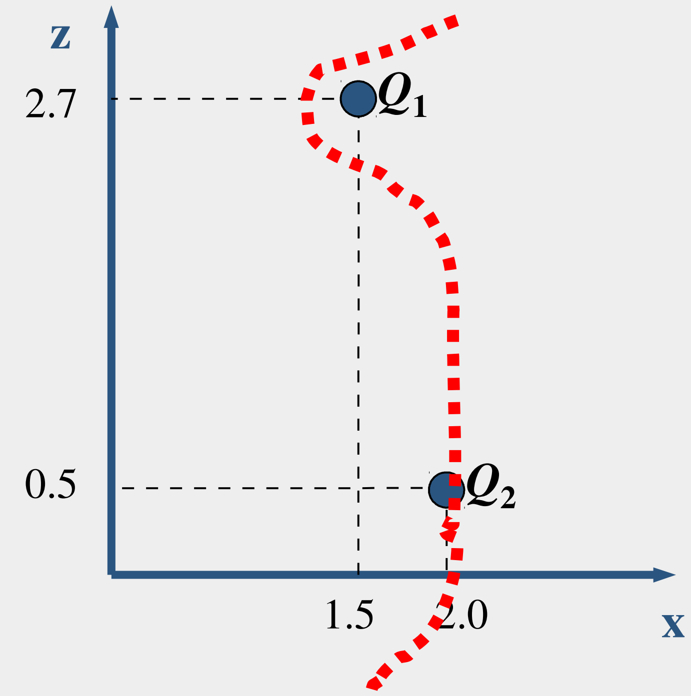
$$T_x = \bar{x}' - (\bar{x}\cos\omega - \bar{z}\sin\omega) = 5.5 - (1.6998 - 0.3806) = 4.1808$$

$$T_z = \bar{z}' - (\bar{x}\sin\omega + \bar{z}\cos\omega) = 3 - (0.4163 + 1.5541) = 1.0396$$

Based on the new pose estimation, revise the correspondecies. (Iterate until pose is stable < ε)

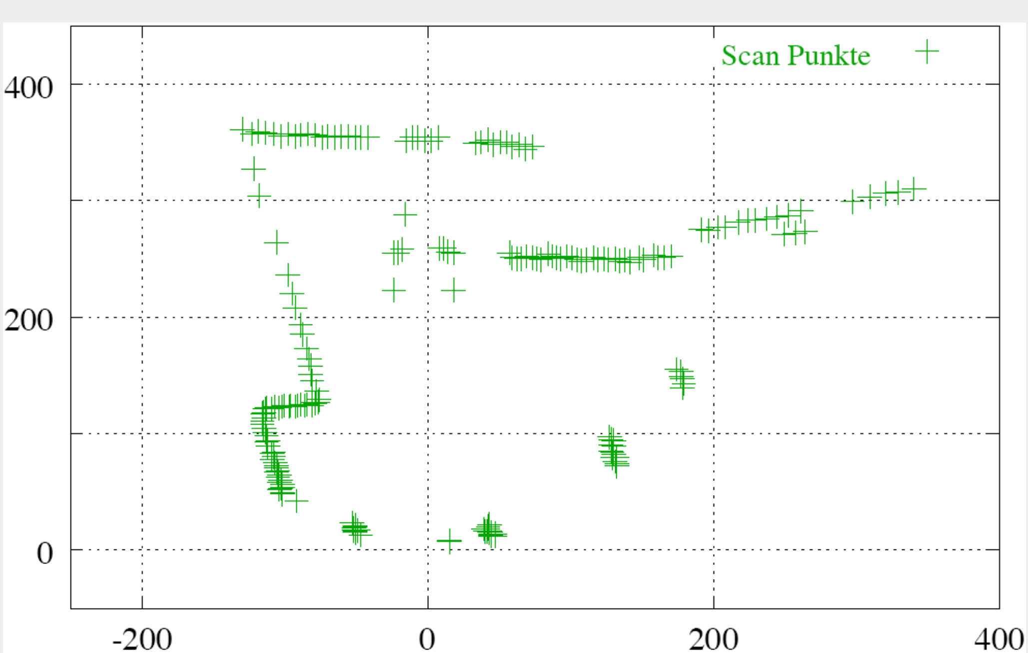
Point Correspondences

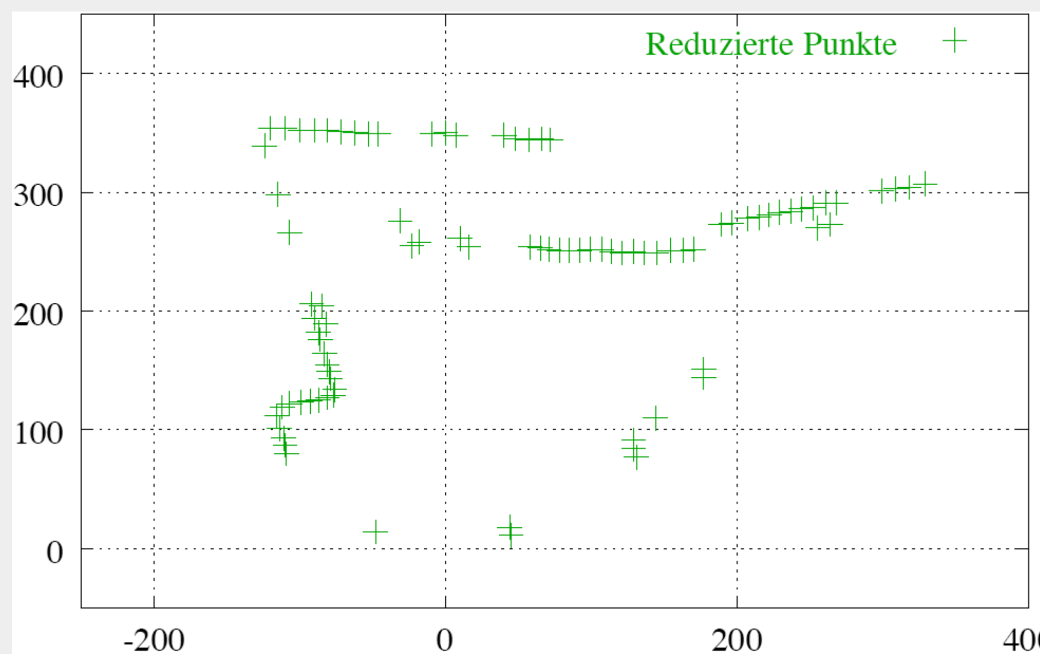
- Given a „Model“ (e.g. Reference scan, continuous curve), which scan points Q_i („data points“) corresponds model points Q_i'?
- How many correspondences exists at all?
- And which?
- Example: Measured points Q_i after pose estimation superimposed of a model set (red)
  - Minimize distances?
  - Maximize overlap?
  - Minimize (ω, T)?
  - Incorporate visibility?



Preparatory work for correspondence estimation

- Not all point of the model and data set have to be taken into account for scan matching
- Example: Reduce data points through replacement of a data point by mean value of the points in an ε-area surrounding it.
  - Linear runtime
  - Linear reduction





Scan Matching: The closest points rule

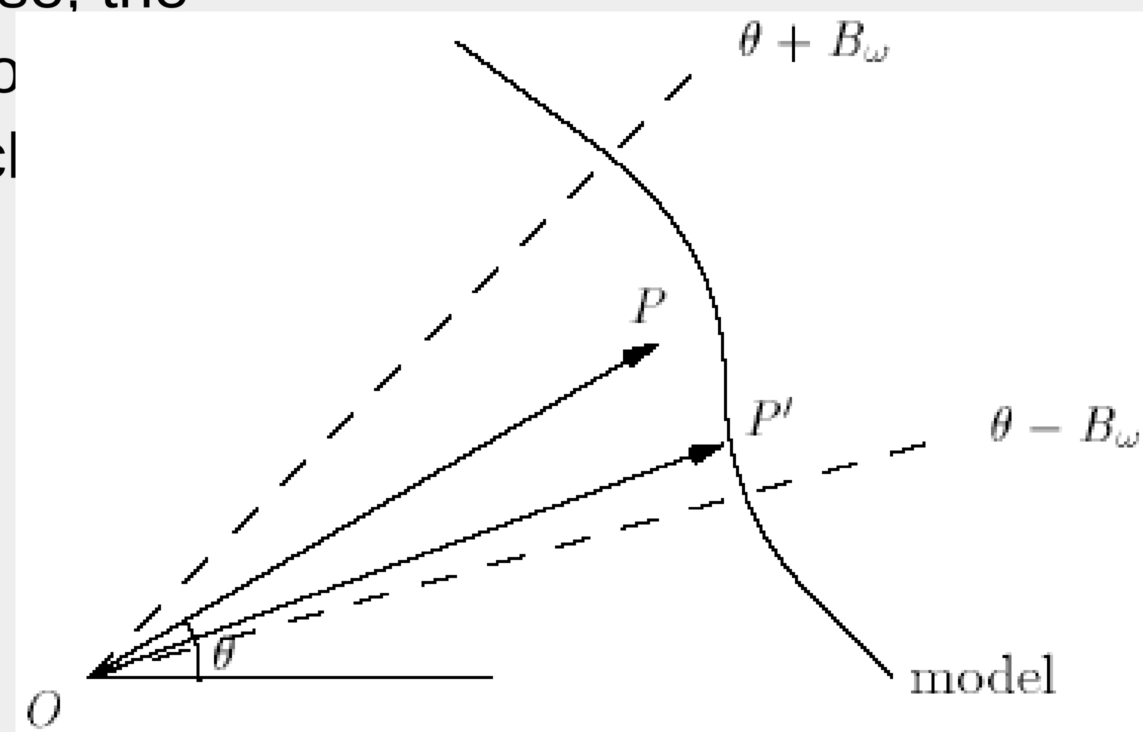
- Iterate through the points (filtered) of the data set
- If there is in the e-area of a data points P_i at least one point in the model set then take the closest point as corresponding point. Define „closest“ via squared distance
  (If there is no such point then P_i has no correspondence)
- Basis of the ICP (*Iterative Closest Points*, Besl/McKay, 1992) (also *Iterative Corresponding Points*) algorithm
- Converges (with the error function) to a local minimum
- Compensates translational errors good, if the rotational errors are small

Scan Matching: Matching range points

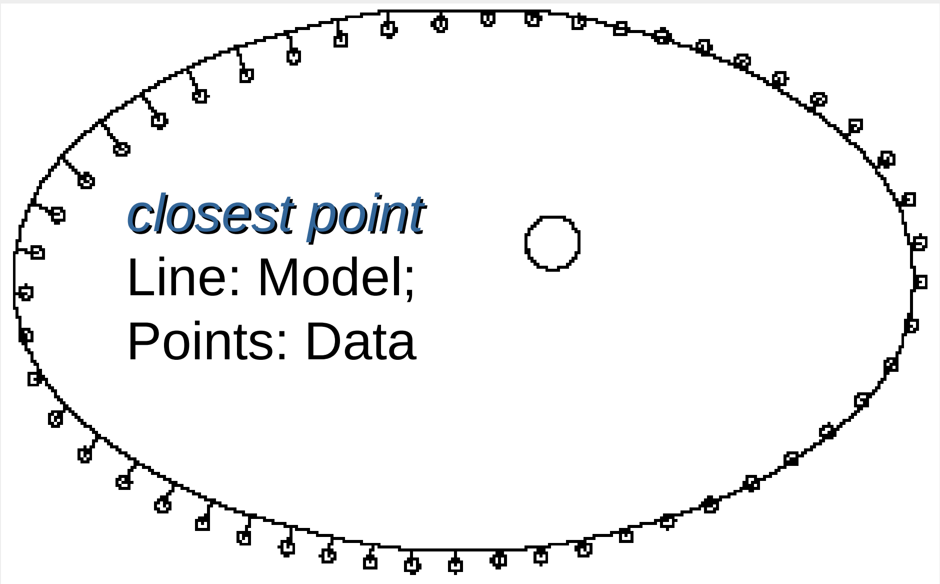
- Use the data points in polar coordinates (r, θ)
  (as they are given by the scanner)
- ω denotes the last estimated rotation estimation.
  (ω usually decreases over the time of the scan process.)
- For a data point P=(r, θ) choose, the
  model points P‘ as correspo
  point from the $\theta \pm B_\omega$ for whicl

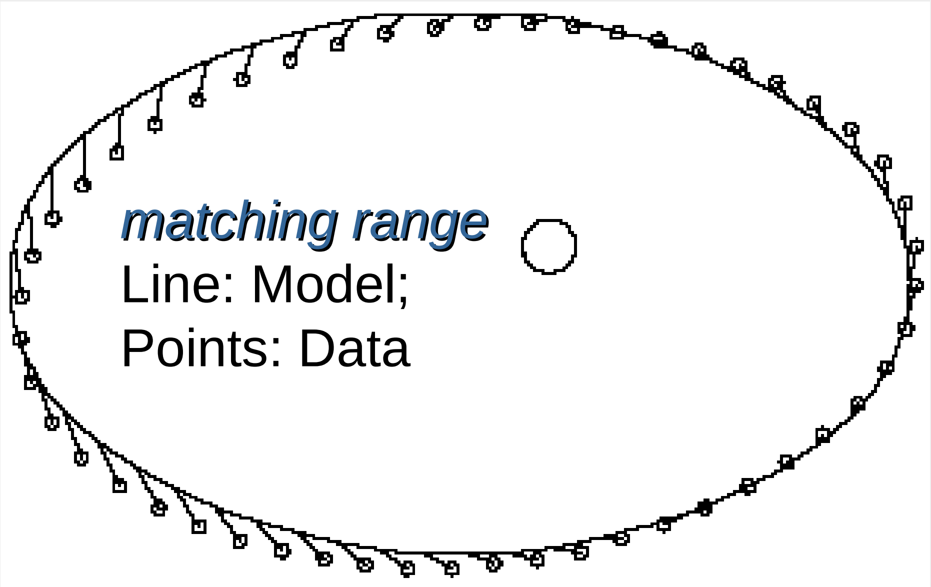
$$\left(\|P'\| - r\right)^2$$

is minimal.



Difference between range point and closest point rule



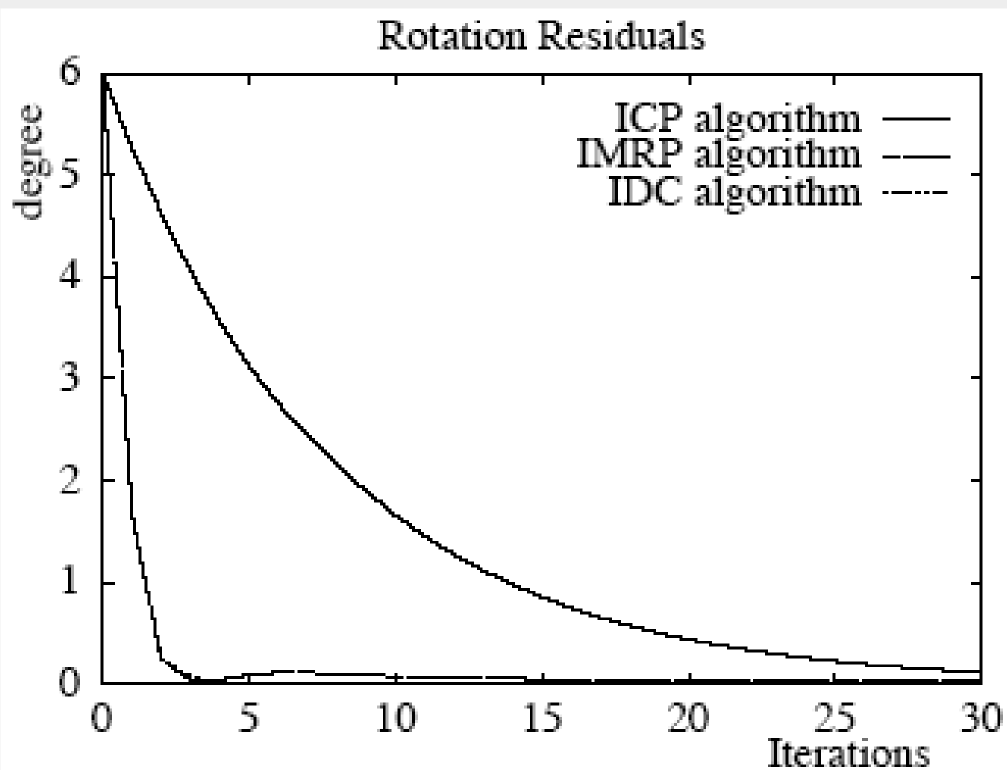


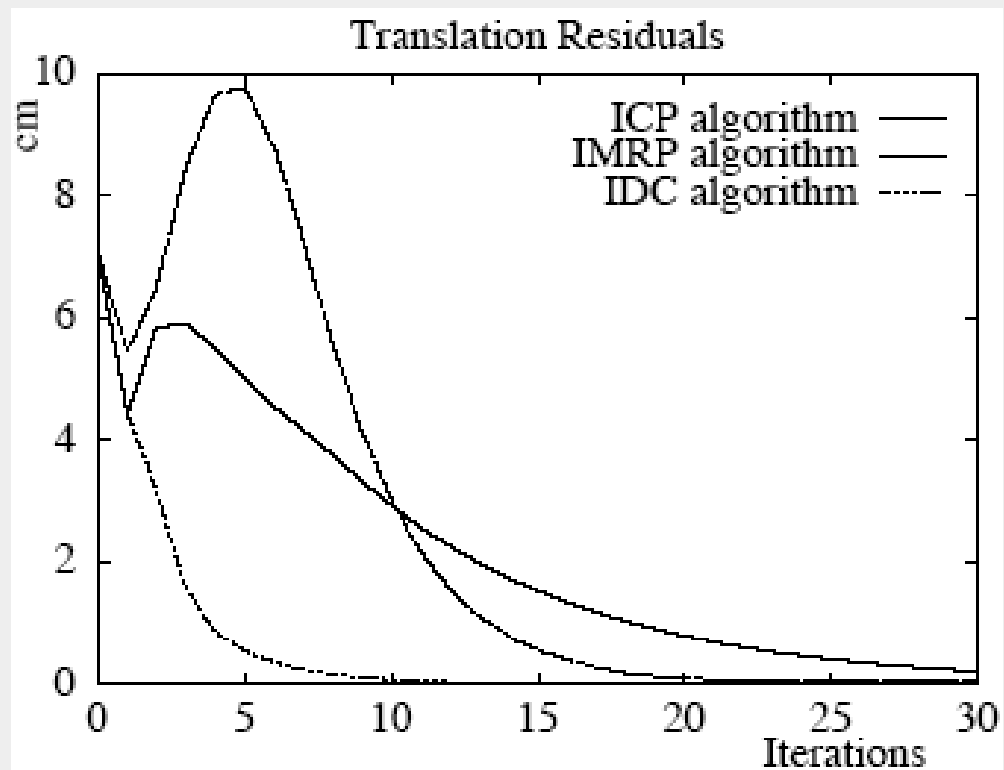
matching-range chooses correspondences with equal rotation

compensates rotational errors good, but has problems with translational ones, as long as B_ω is high.

IDC – Iterative Dual Correspondence

- Compute Correspondences based on *c-p-* and *m-r-p-* rule
- Calculate for both correspondences the optimal pose estimate respectively
- Result: Rotation based on *m-r-p-* and translation from *c-p-* rule

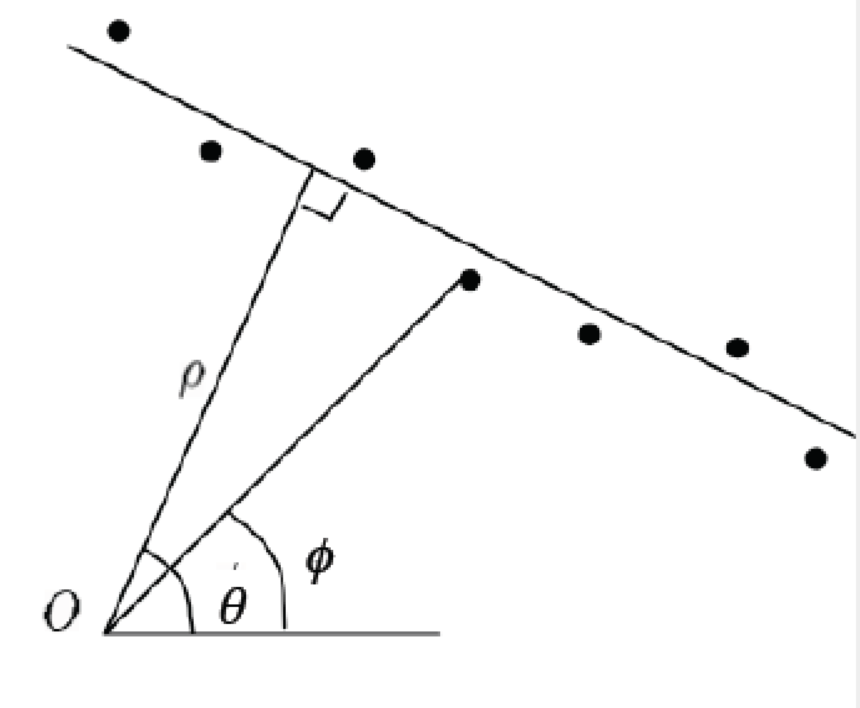




Localization at Lines

- Instead of matching point sets with point sets, match lines with lines for localization
- Makes sense in polygonal environments!
- Model Lines origin from a map or from a previous reference scan
- Question: How to make lines from the laser data?
- Fit a regression line into the set of points (e.g. Hough-Transformation)
- Unclear:
  - How many lines are in a point set?
  - How are outliers treated?
- Method must be real time capable
- Exploit the ordering of the scan points as criterion for neighborhood!

Tangential lines according to Lu/Milios



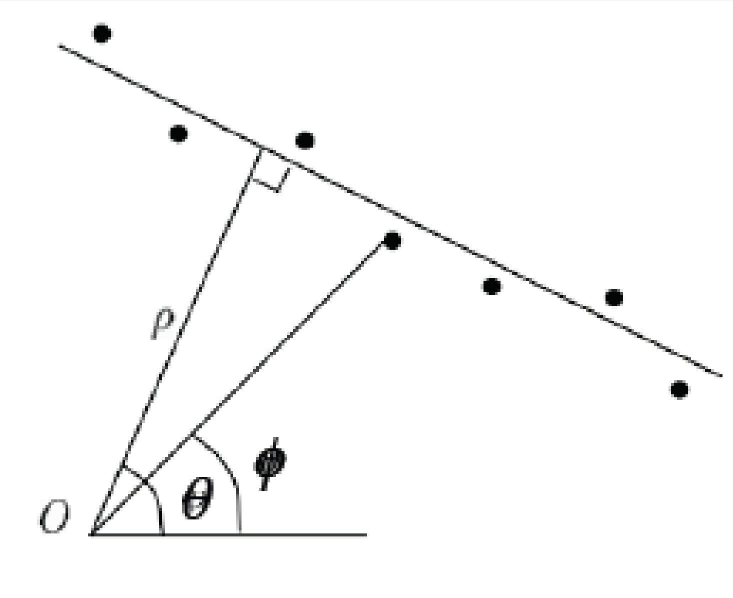
Idea:

- For points at angle ϕ specify a regression line using $(n–1)/2$ adjacent points to the left and right
- Distance via normal is ρ; angle difference is $|\theta-\phi|$ between ρ and normal

Conditions:

- $|\theta-\phi|$ cannot exceed a maximal value
  (if so „small“ scan angle or distance discontinuity in the data)
- Sum of the squared distances between n points cannot exceed a maximal value

Line based solution according Lu/Milios



The wanted line is the one that minimizes the error function

$$E_{\text{fit}} = \sum_{i=1}^{n} \left(x_i \cos\theta + z_i \sin\theta - \rho \right)^2$$

There exists a closed form solution for $\min_{(\rho,\theta)} E_{\text{fit}}$

$$\theta = \frac{1}{2} \arctan \frac{-2S_{xz}}{S_{zz} - S_{xx}}$$

$$\rho = \bar{x} \cos\phi + \bar{z} \sin\phi$$

And it is:

$$\min_{(\rho,\theta)} E_{\text{fit}} = \frac{1}{2} \left[S_{xx} + S_{zz} - \sqrt{4S_{xz}^{\ 2} + \left(S_{zz} - S_{xx} \right)^2} \right]$$

Line based solution – online variant

Idea:

- For points of a (reduced) scan check in order of the points if the points ± a difference are on the line with the preceding points
  - If so, extend the line with the new point
  - If not, finish the line and start a new one
- A line is detected, iff ≥*n* points are on the line (e.g. *n*=3)

If $a_j, \ldots, a_k$ is the currently constructed line …

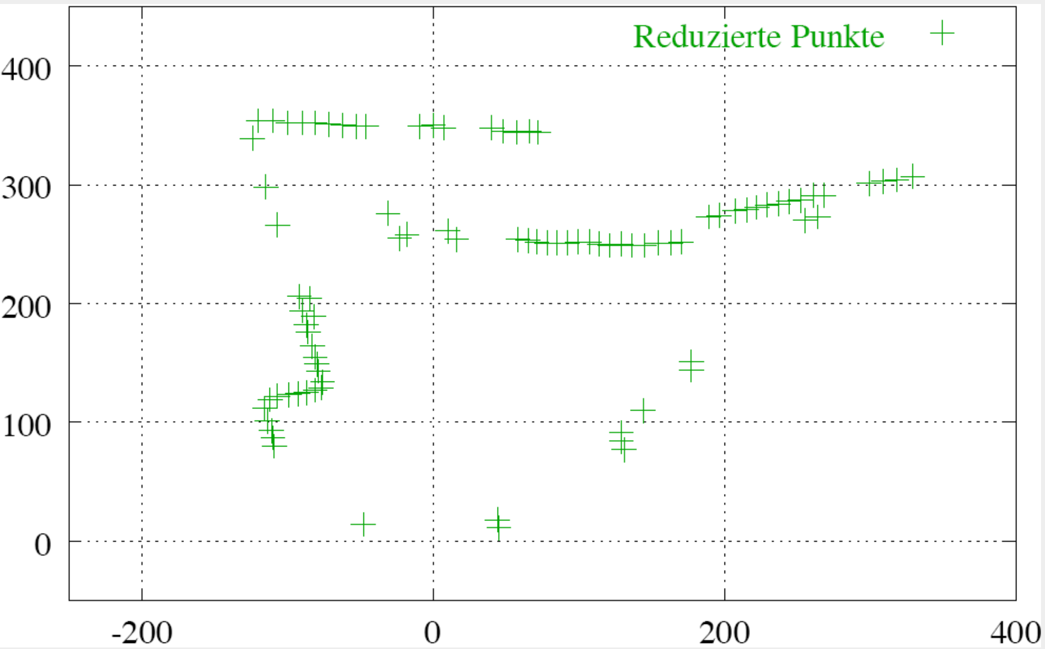
1. Small distance new point to preceeding points (beeline)
2. Local beeline argument typical value $\varepsilon \approx 0.8$
3. Maximum direct distance

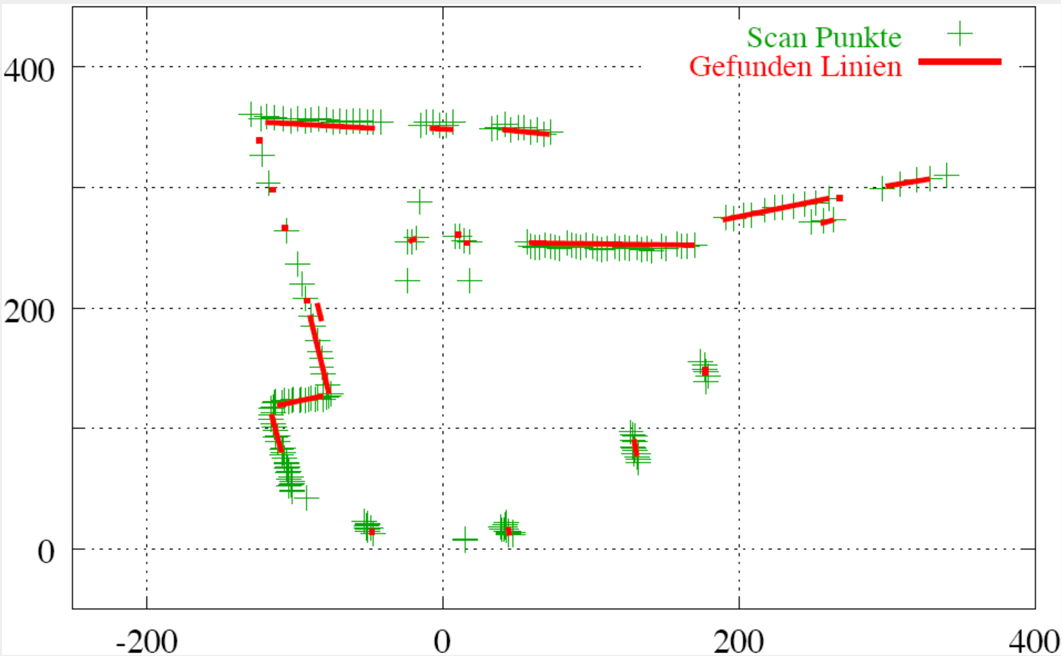
$$[1 \geq] \frac{\|a_j, a_{k+1}\|}{\sum_{i=j}^{k} \|a_i, a_{i+1}\|} \geq \varepsilon(k)$$

$$[1 \geq] \frac{\|a_{k-1}, a_{k+1}\|}{\|a_{k-1}, a_k\| + \|a_k, a_{k+1}\|} \geq \varepsilon$$

Results: Line based solution – online variant

1. Reduce points
2. Find lines





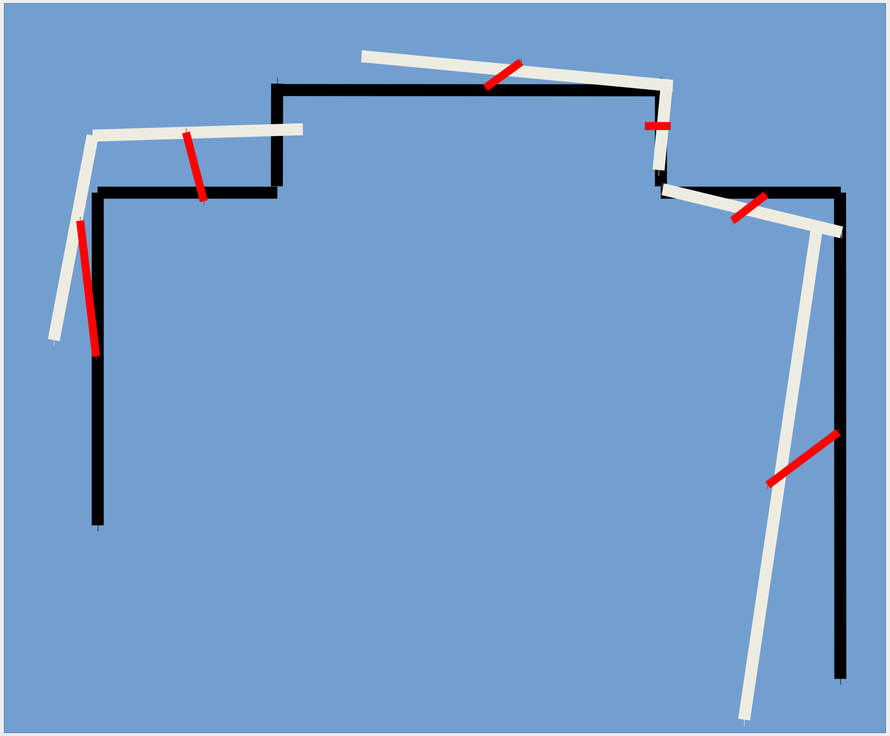
Here: lines found in reduced points with all points superimposed (cf. Slide 222)

What to do with the computed lines?

- Match lines of the current scan against lines in a line map or a previous acquired scan.
  - Use Odometry a starting guess
  - Attention! Not all lines in the map are in present in the scan! Not all scan lines are in the map!
    - Use e.g., line histograms for improving the rotation estimate (weight them with the length of the lines)!
    - Transform until a maximum of overlap is found!

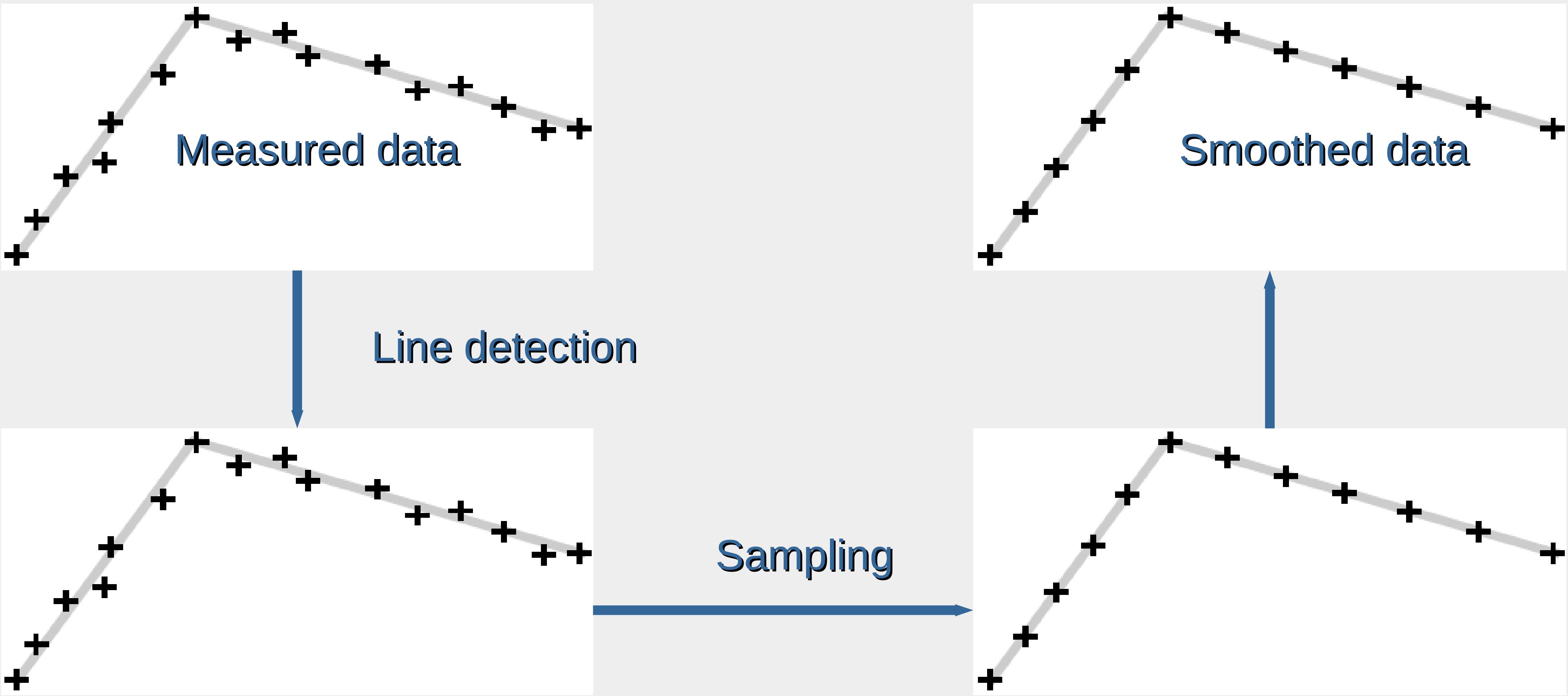
Example:

- Describe a line with center, length, orientation: $L=\langle c, l, \phi\rangle$
- Pair lines using different thresholds for ϕ *and l*
- Match center with ICP / IDC



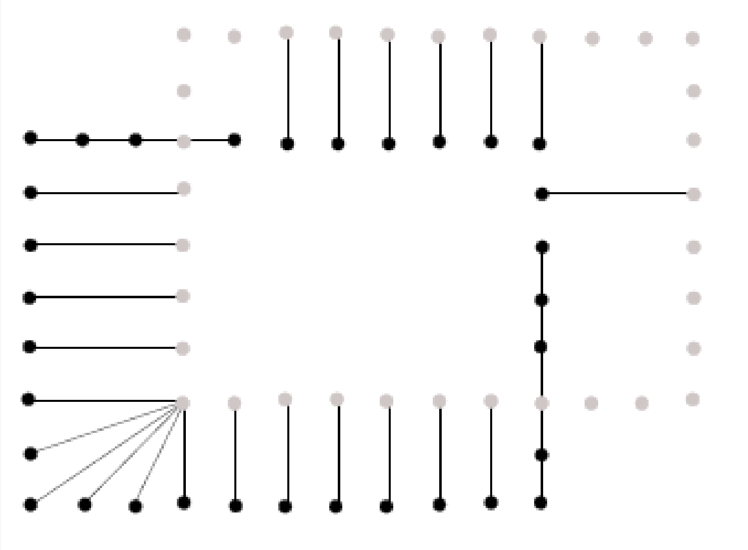
Convert lines back to Points: Sampling

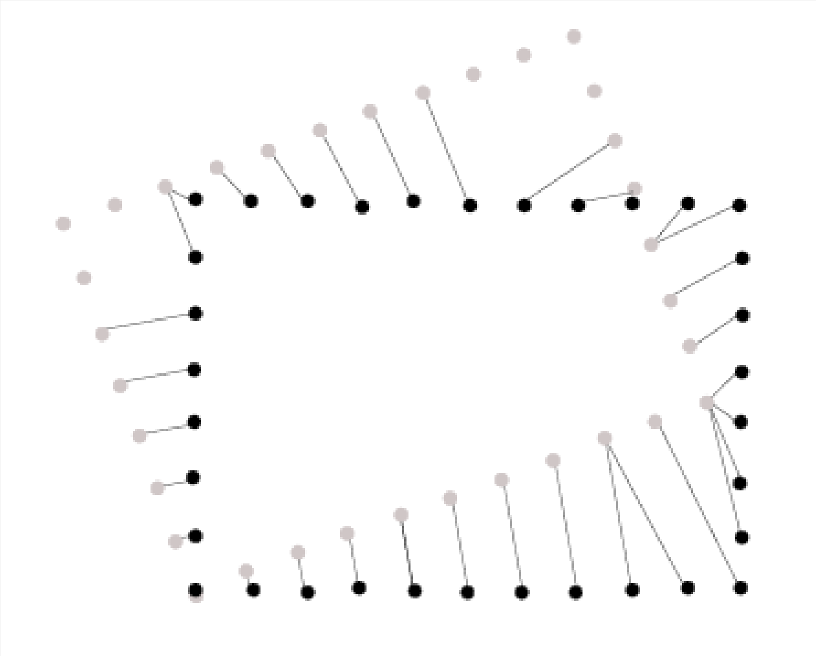
Usually one wants to match points instead of lines ICP/IDC),
but line detection helps smoothing the data!



Difference between range point and closest point rule

- Closest point rule





- Range point rule