



4 – Particle filter

Advanced Methods for Mapping and Self-localization in Robotics
MPC-MAP

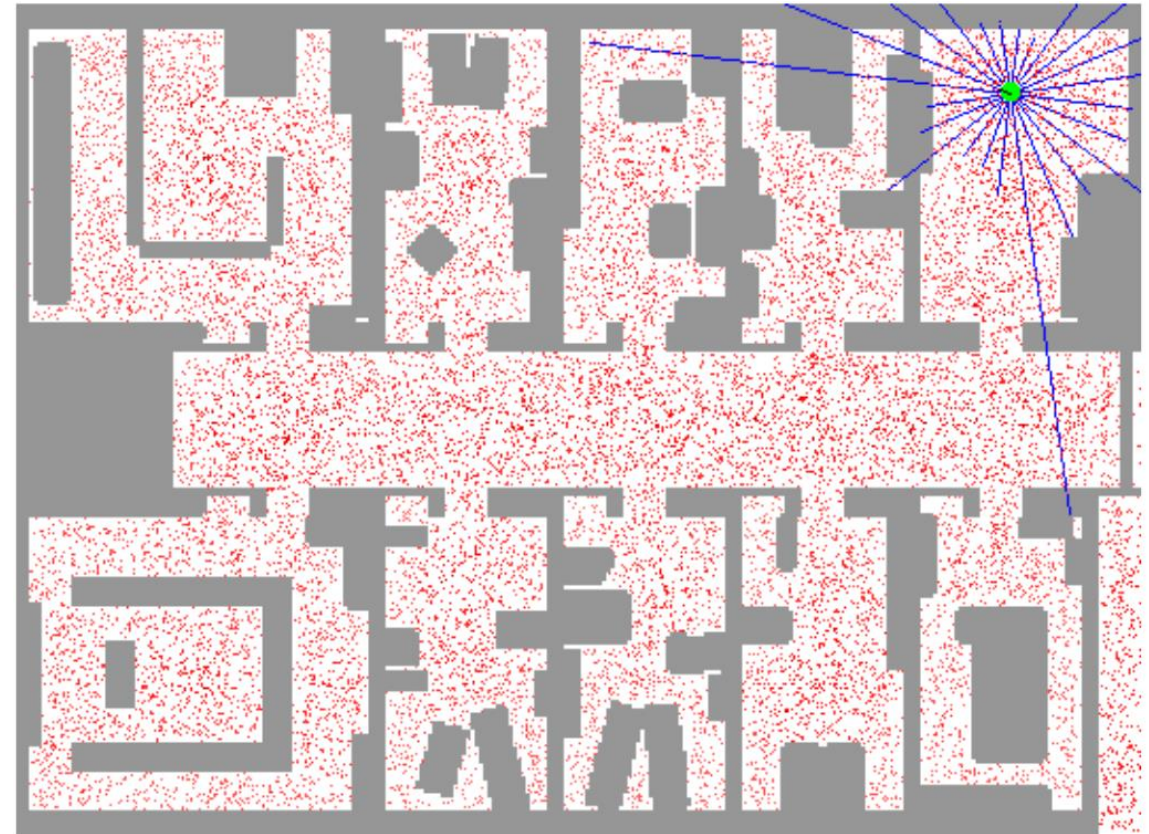
Tomas Lazna
Brno University of Technology
2024



What can a convenient localization algorithm offer?

- Multimodality
- Continuity
- Intuitiveness
- Efficiency
- Scalability

PDF = Probability Density Function



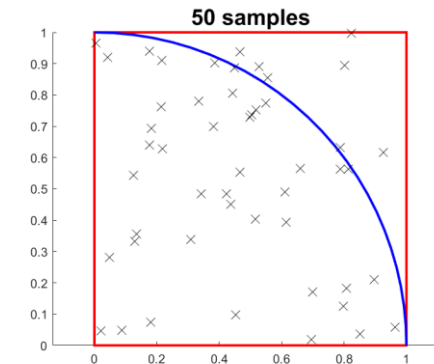
[1]



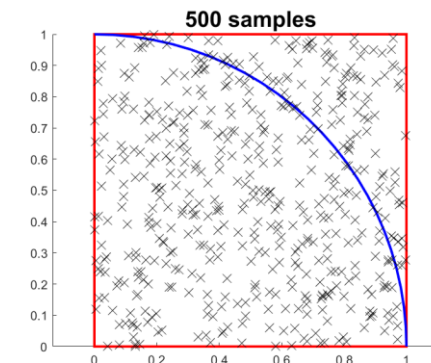
- Numerical methods based on random sampling
- Optimization, numerical integration, drawing from PDFs, modeling
- Law of large numbers
- Estimating the π value (Buffon's needle)
- Particle filter = Sequential MC method



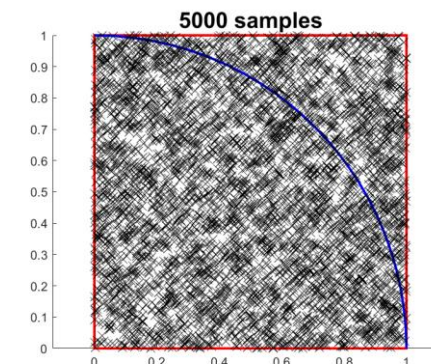
[1]



$$\pi \approx 3.36$$



$$\pi \approx 3.10$$



$$\pi \approx 3.13$$



- Particle representation of a PDF

$$\mathcal{X} = \{[x^{(i)}, w^{(i)}]\}_{i=1, \dots, N}$$

State hypothesis

Belief (weight)

$$p(x) = \sum_{i=1}^N w^{(i)} \delta_{x^{(i)}}(x)$$

Dirac delta function

- Bayes' theorem

$$P(H|E) = \frac{P(E|H) \cdot P(H)}{P(E)}$$

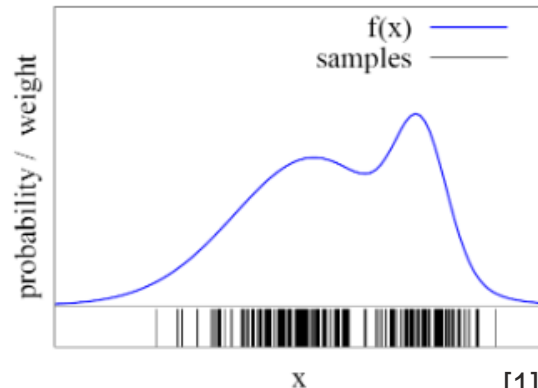
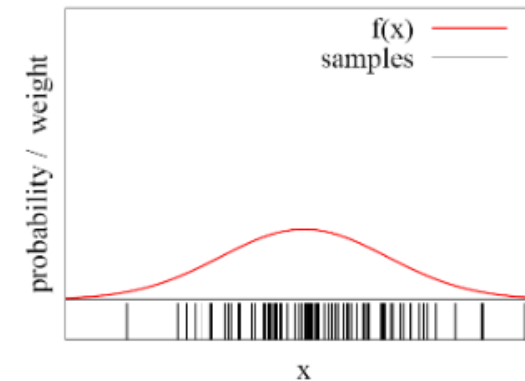
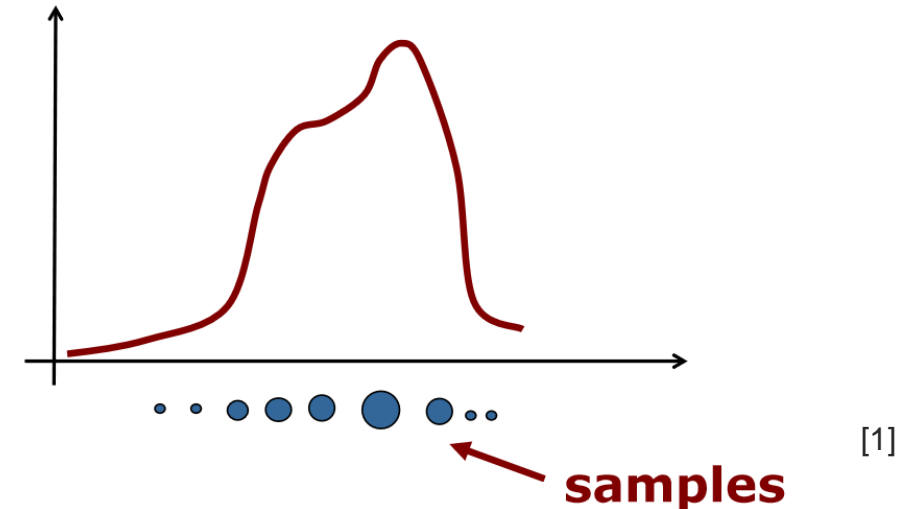
$H \dots$ hypothesis

$E \dots$ evidence

- Particle filter applications:

- Robot localization
- Object tracking, computer vision
- General estimation in nonlinear systems

- Original article (referred to as 'bootstrap filter') [2]





- Uniform distribution

- Any random number generator
- Usually a pseudorandom series
- Can be utilized for other PDFs

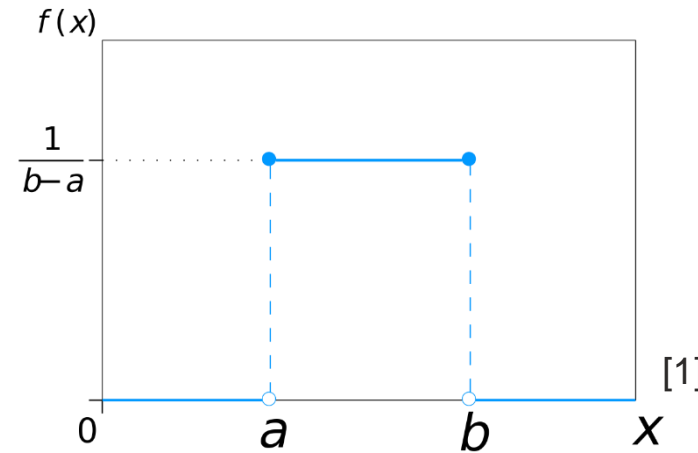
- Normal distribution

- Parametric function
(mean μ , variance σ^2)
- Approximation by UD:

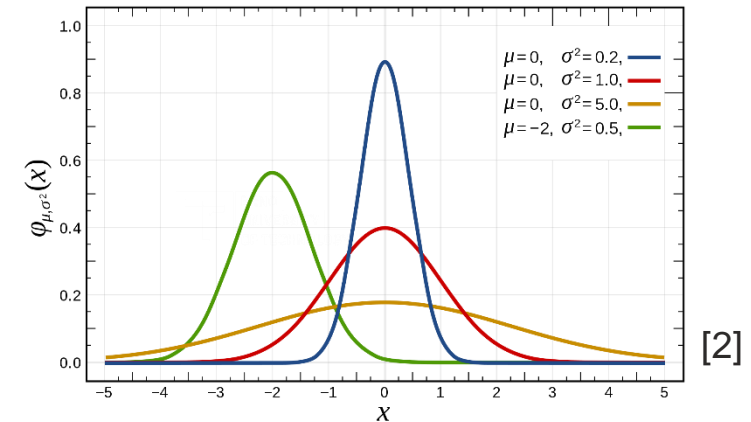
$$x \leftarrow \sum_{i=1}^{12} \text{rand}(-0.5\sigma, 0.5\sigma)$$

- Other distributions

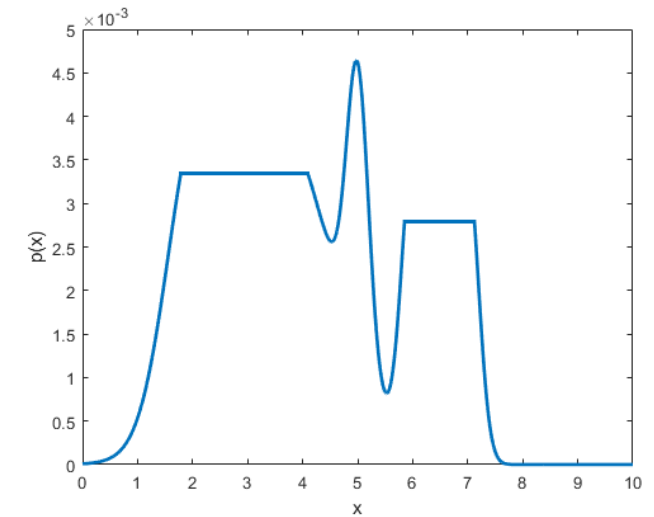
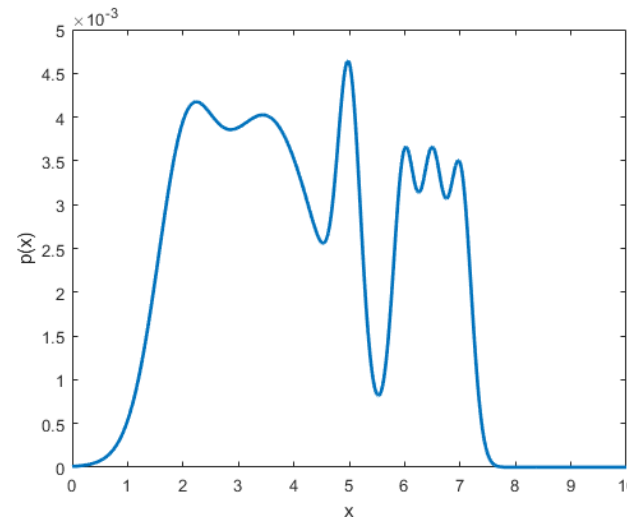
- Parametric
- Non-parametric



[1]

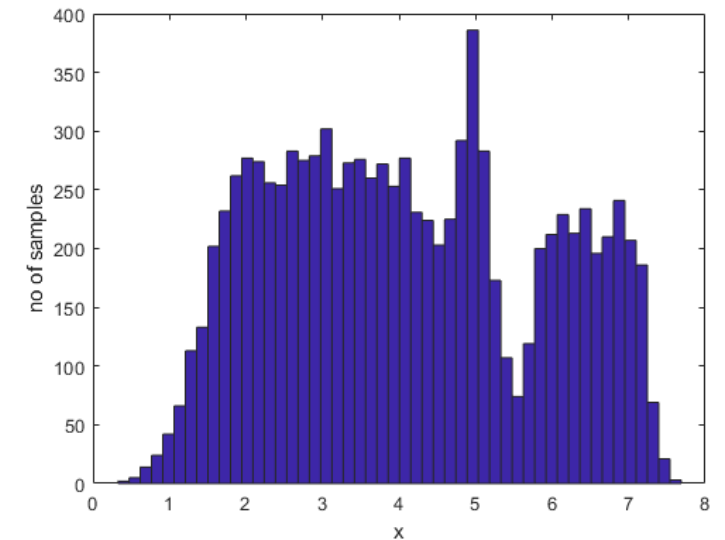
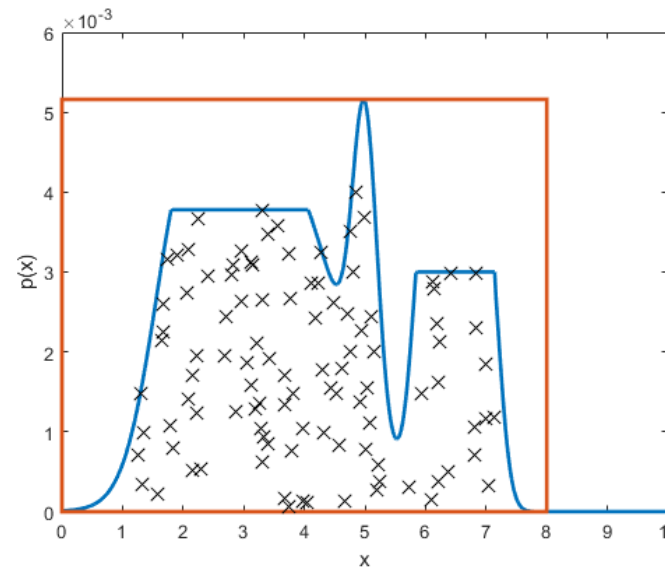
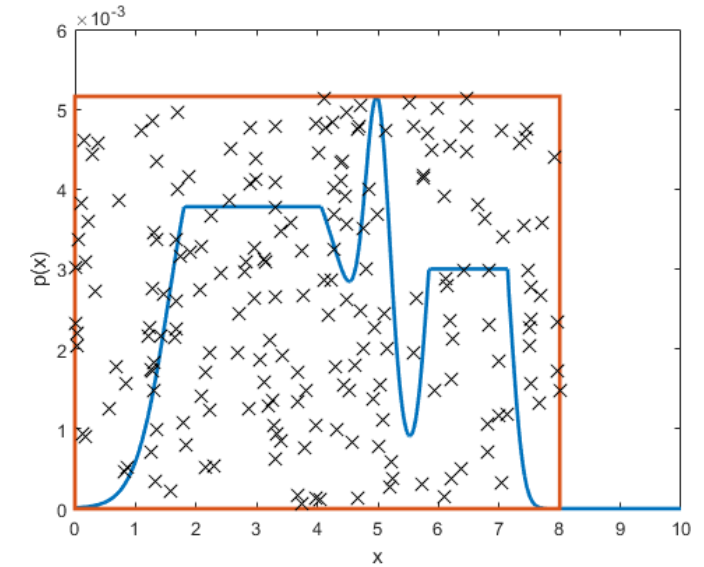
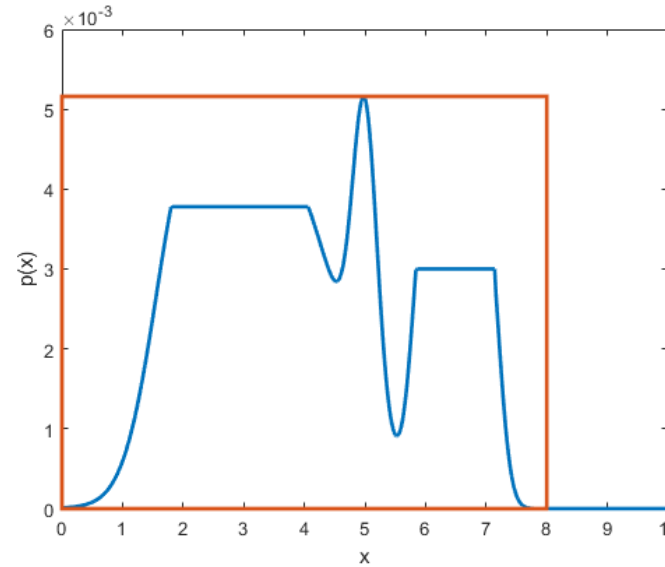


[2]



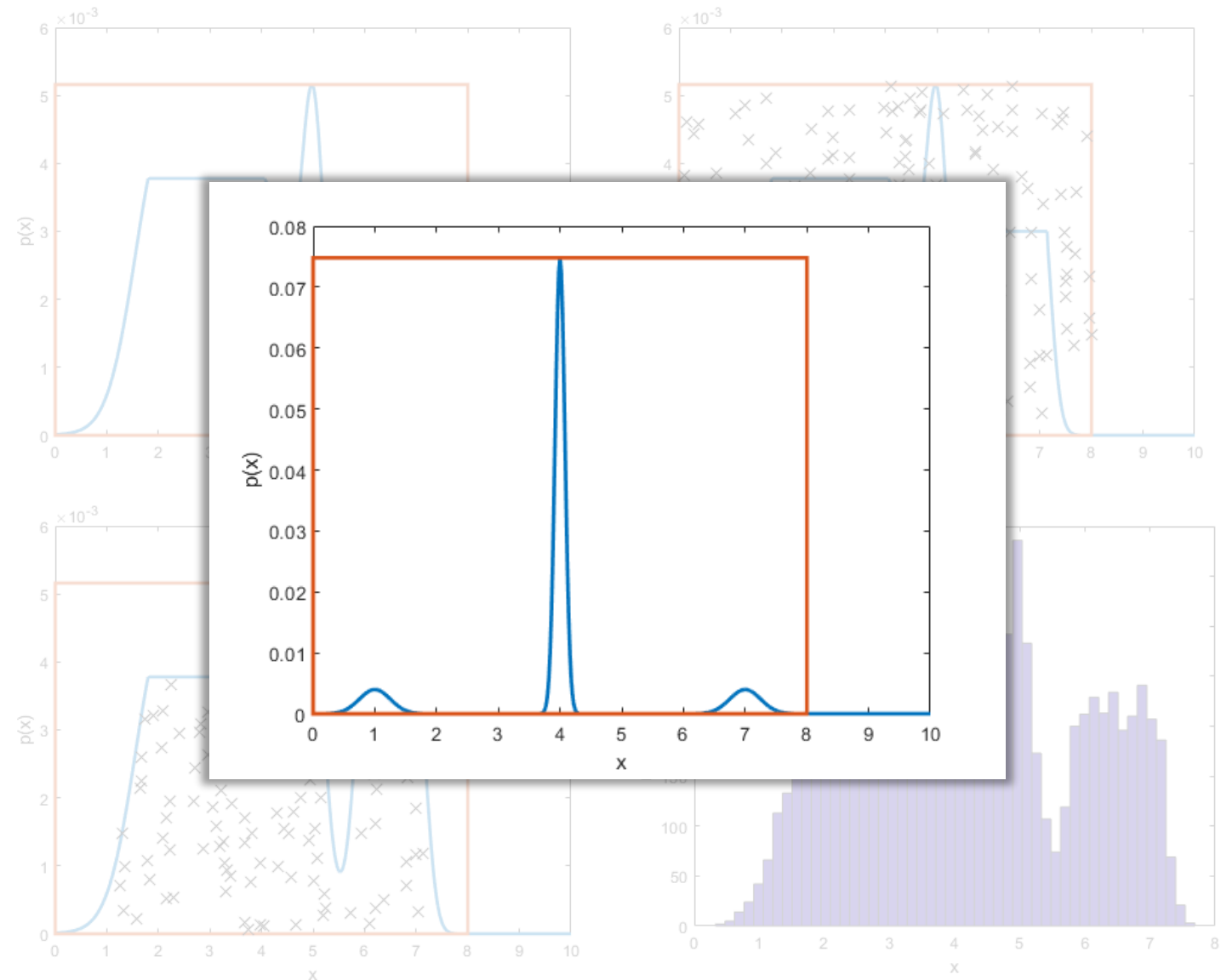


- Generate random samples $[y, f(y)]$ in range:
 - $y \in \langle x_{\min}; x_{\max} \rangle$
 - $f(y) \in (0; \max(p(x)))$
- Reject sample if:
 - $f(y) > p(y)$
- The random variable Y is now distributed according to $p(x)$





- Generate random samples $[y, f(y)]$ in range:
 - $y \in \langle x_{\min}; x_{\max} \rangle$
 - $f(y) \in (0; \max(p(x)))$
- Reject sample if:
 - $f(y) > p(y)$
- The random variable Y is now distributed according to $p(x)$
- Tends to be inefficient



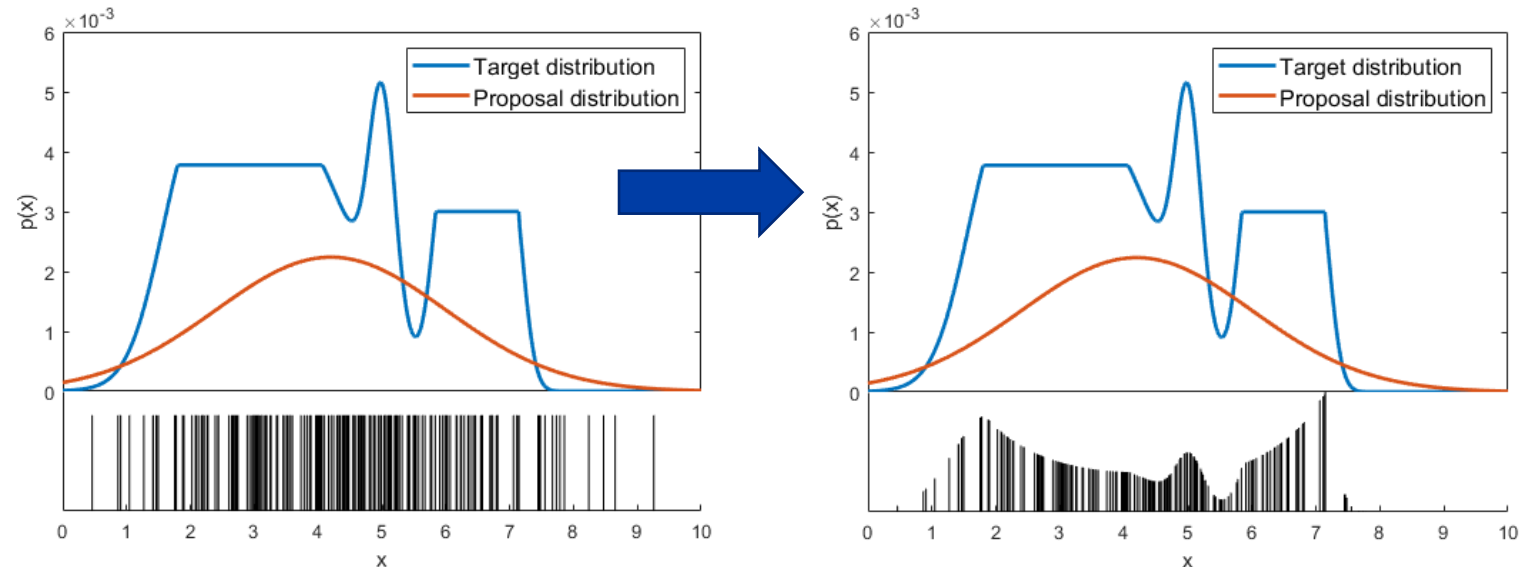


- Use other distribution π that is simple to draw from
- Correct the difference between the target distribution f and proposal π by assigning 'weights' to random samples:

$$w(x) = \frac{f(x)}{\pi(x)}$$

- To ensure samples are drawn from the whole target distribution, following condition needs to be met:

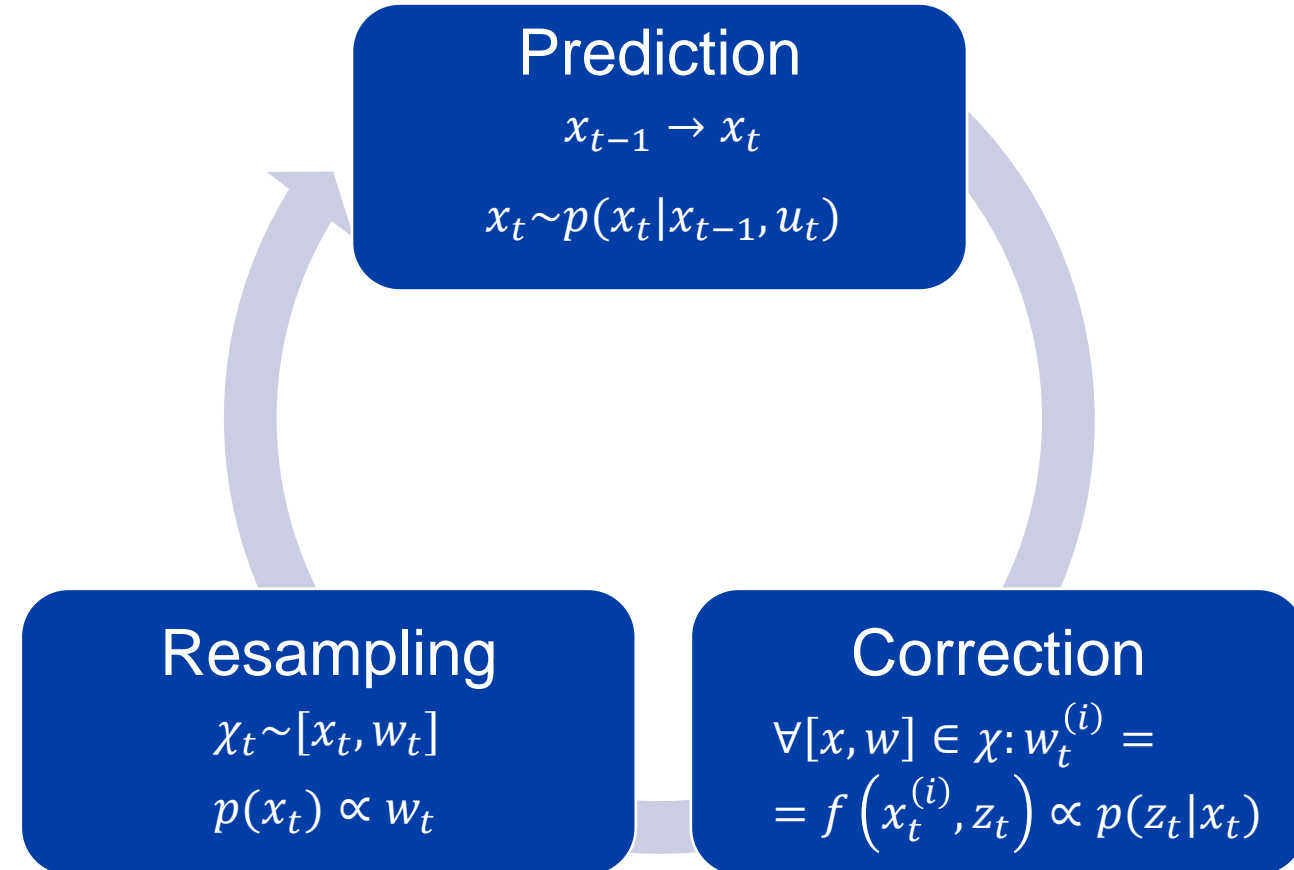
$$\forall x \in \mathbb{R}: f(x) > 0 \Rightarrow \pi(x) > 0$$



- Other methods for drawing random samples
 - Adaptive rejection sampling [1]
 - Markov chain Monte Carlo (MCMC) methods, e.g., Metropolis-Hastings algorithm [2]

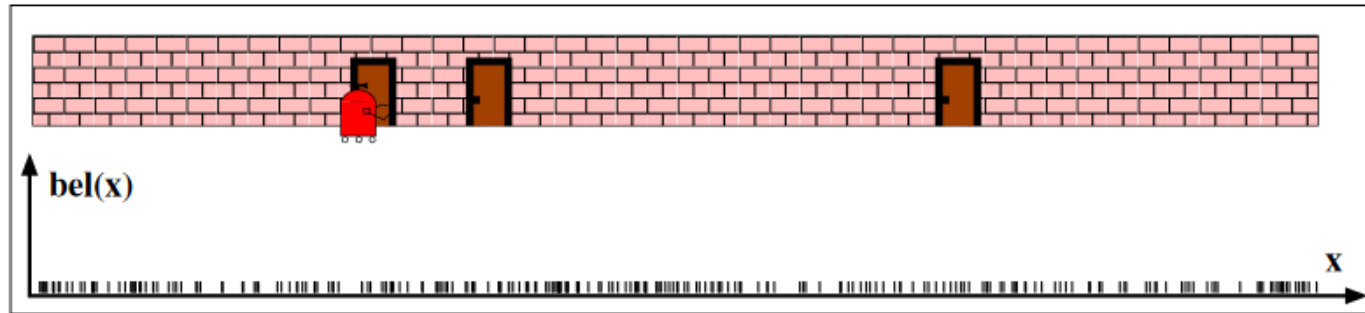


- Prediction step
 - State of all particles is updated in accordance with starting state and the input vector
 - Goal: Estimate the state transition
- Correction step
 - Particles' hypotheses are compared with actual measurement
 - Goal: Find out which particles are the fittest
- Resampling step
 - Draw random samples from the previous particle set with the probability given by weights
 - Goal: Increase the particle density in more probable parts of state space

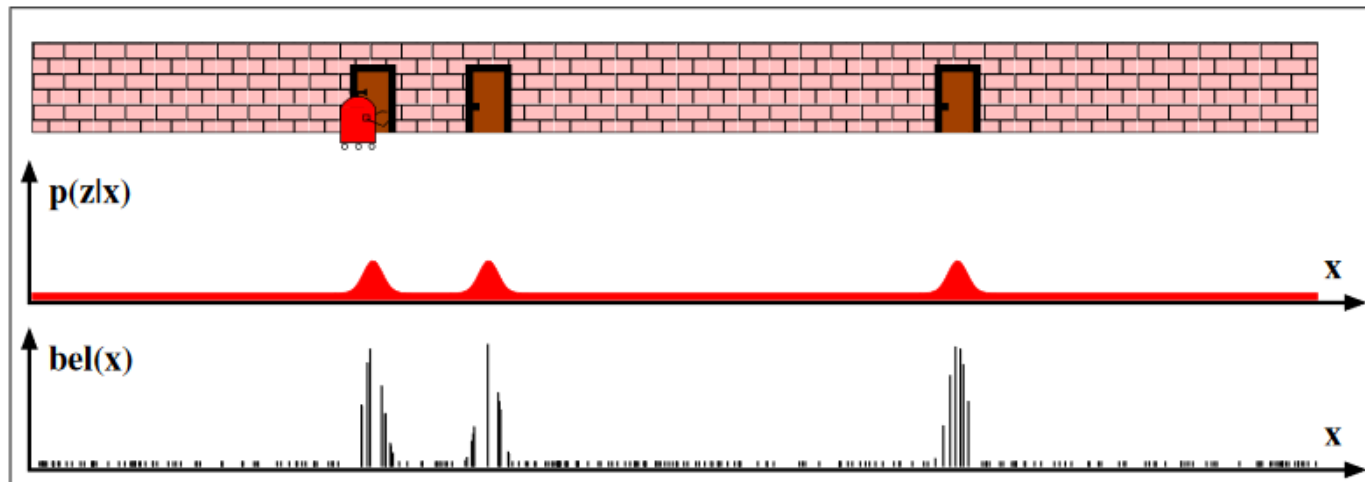




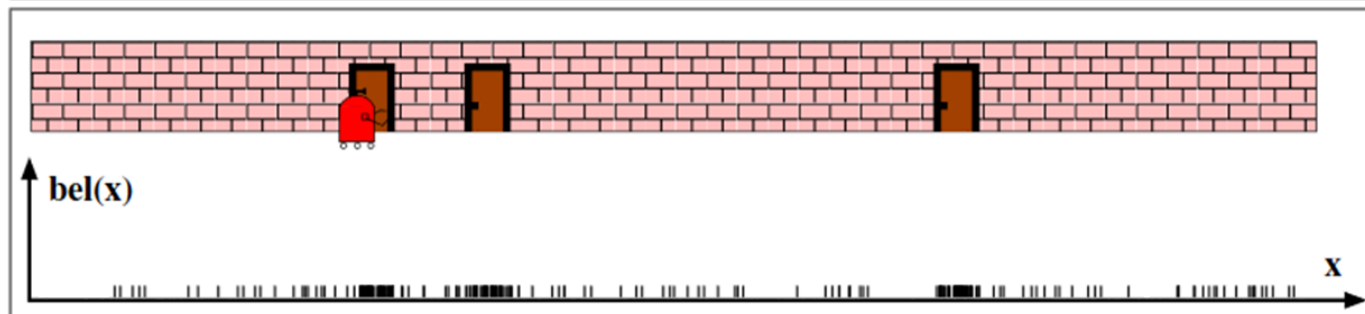
1. Initialization



2. Correction



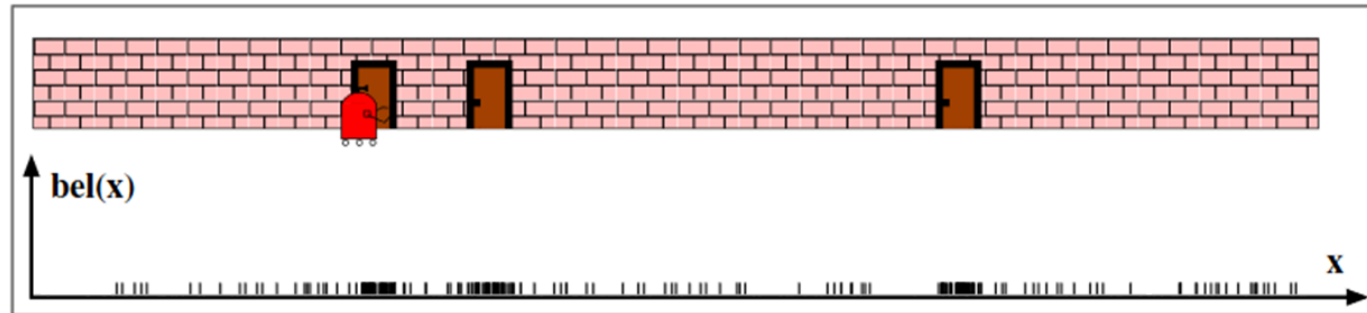
3. Resampling



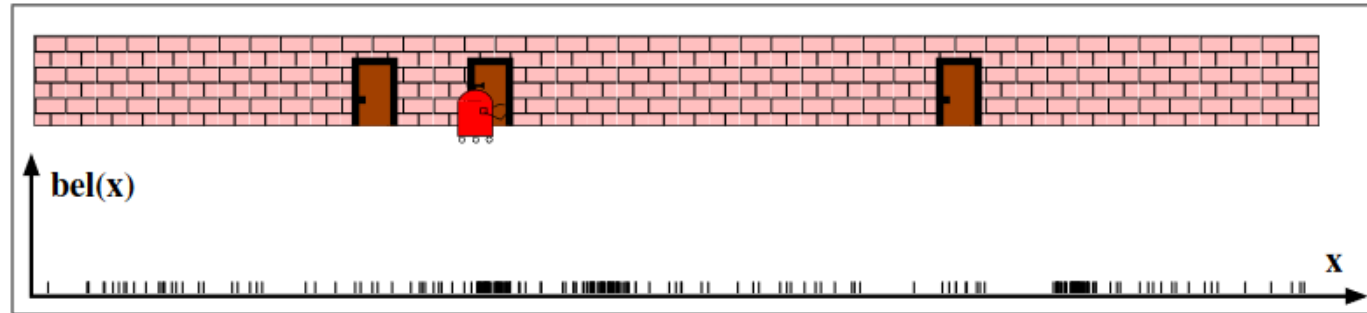
[1]



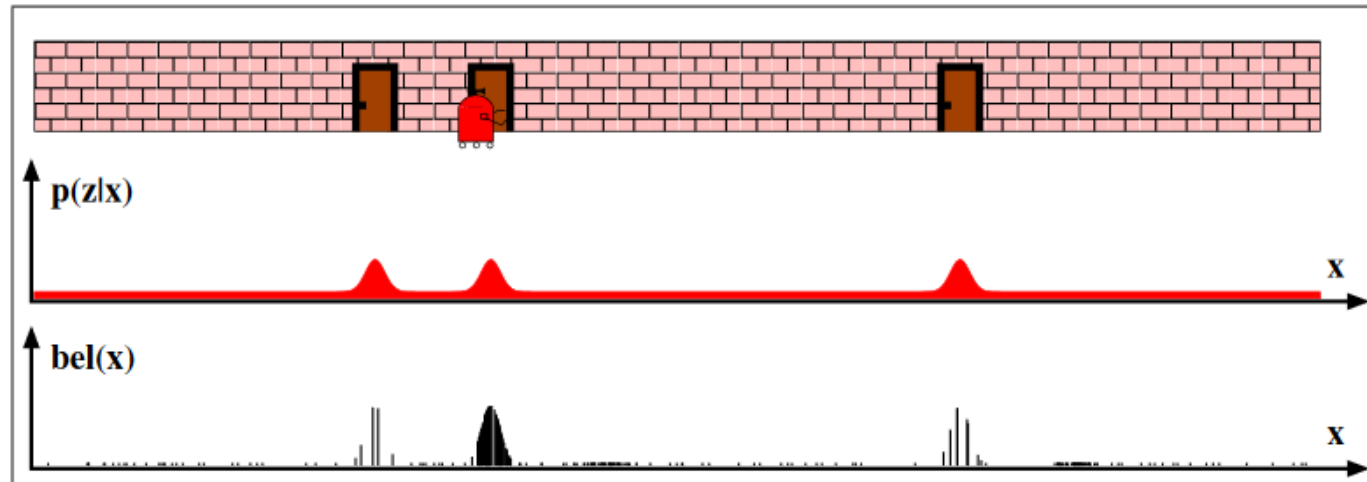
3. Resampling



4. Prediction



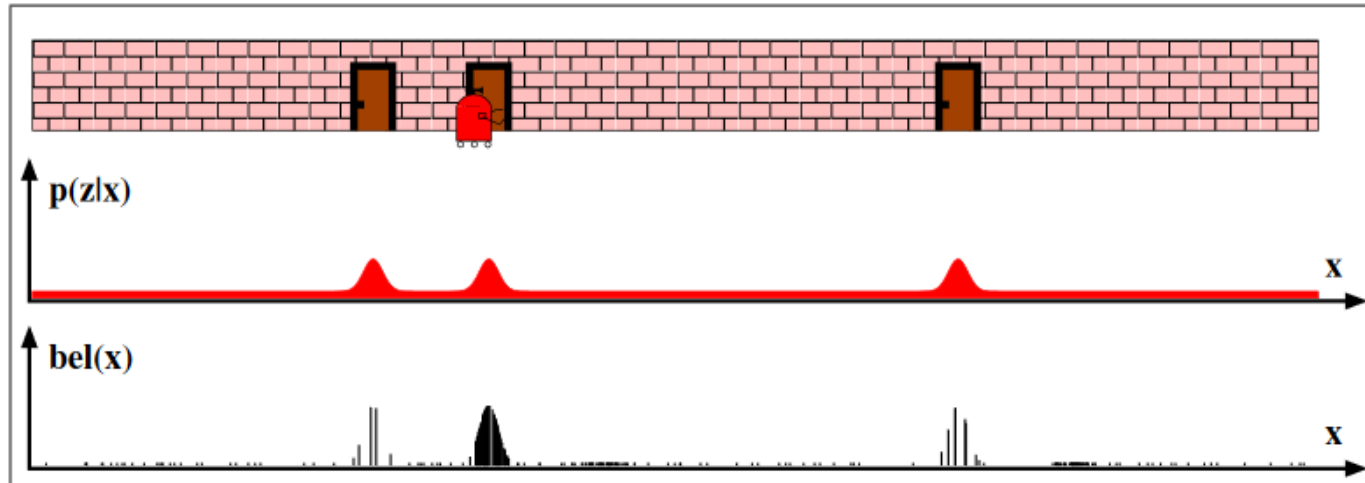
5. Correction



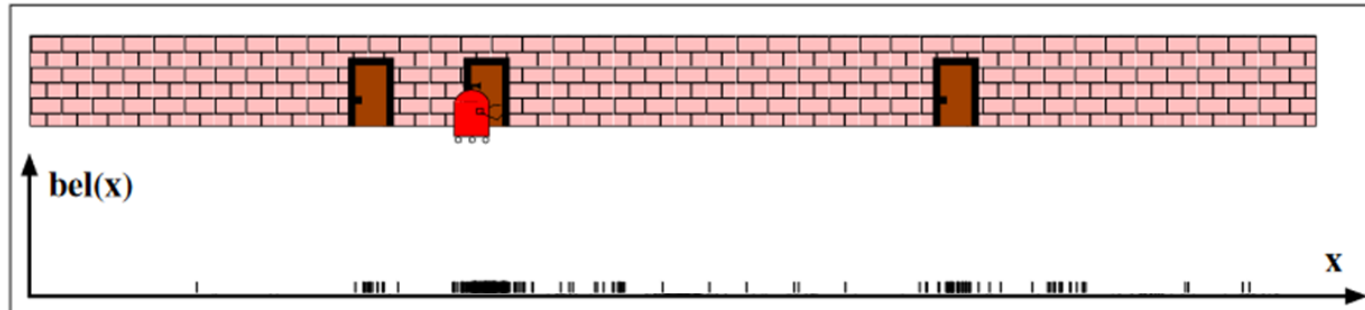
[1]



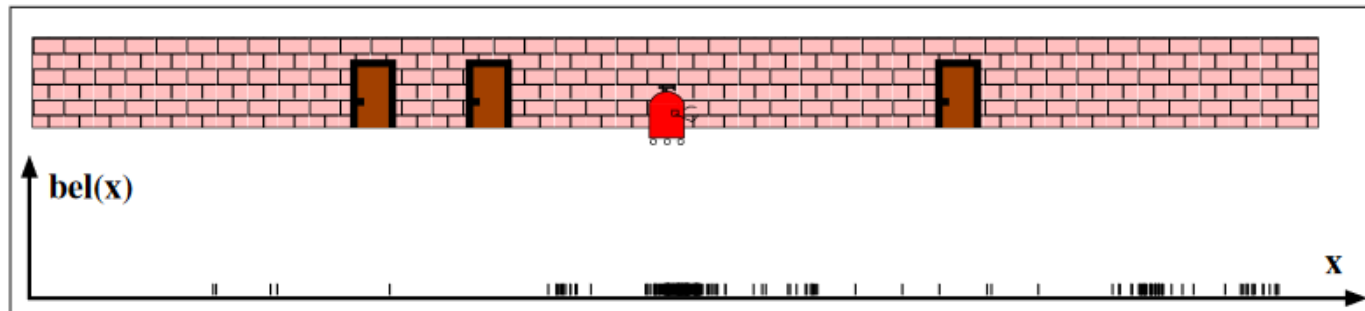
5. Correction



6. Resampling



7. Prediction



[1]



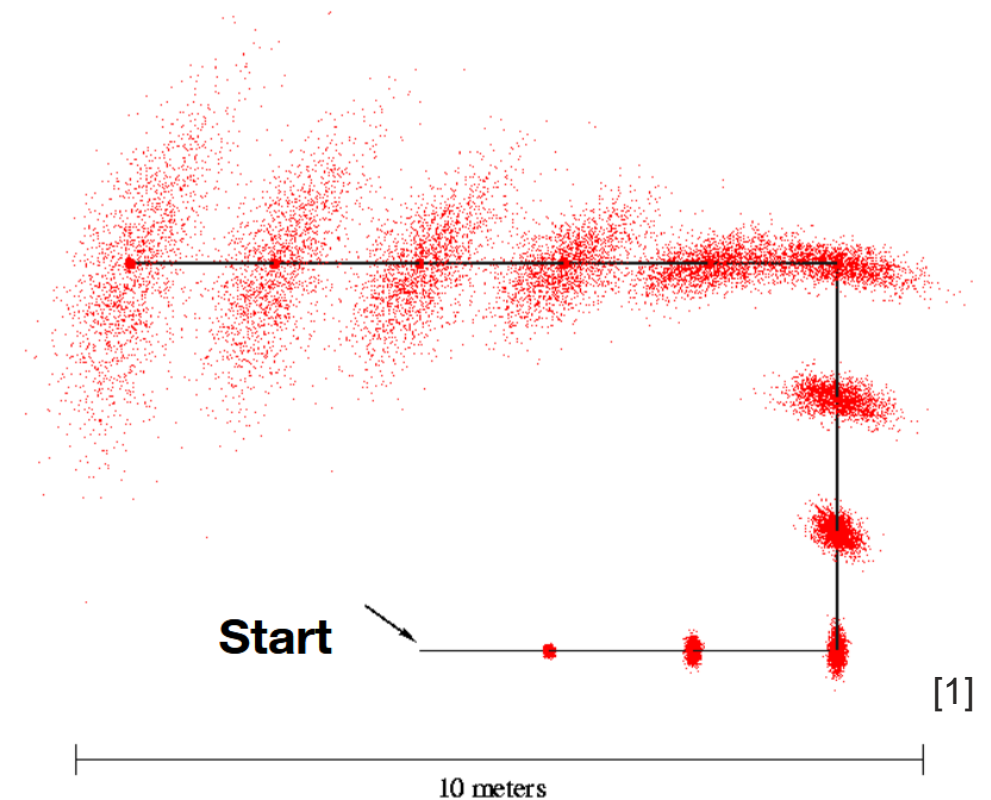
- Predict state of the system after a control vector is applied
- Increase variance of particles
- Let us assume that the state transition depends on the previous state only (first order Markov Process)
- The noise can have arbitrary distribution

$$\mathbf{x}_t^{(i)} = f(\mathbf{x}_{t-1}^{(i)}, \mathbf{u}_t, \mathbf{Q})$$

Predicted state $\rightarrow \mathbf{x}_t^{(i)}$
 Original state $\rightarrow \mathbf{x}_{t-1}^{(i)}$
 Control vector $\rightarrow \mathbf{u}_t$
 Random variable – noise $\rightarrow \mathbf{Q}$

- Example – omnidirectional drive

$$\begin{pmatrix} x_t \\ y_t \\ \theta_t \end{pmatrix} = \begin{pmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{pmatrix} + \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} \Delta t + \mathbf{Q}$$



Prediction

$$x_{t-1} \rightarrow x_t$$

$$x_t \sim p(x_t | x_{t-1}, u_t)$$



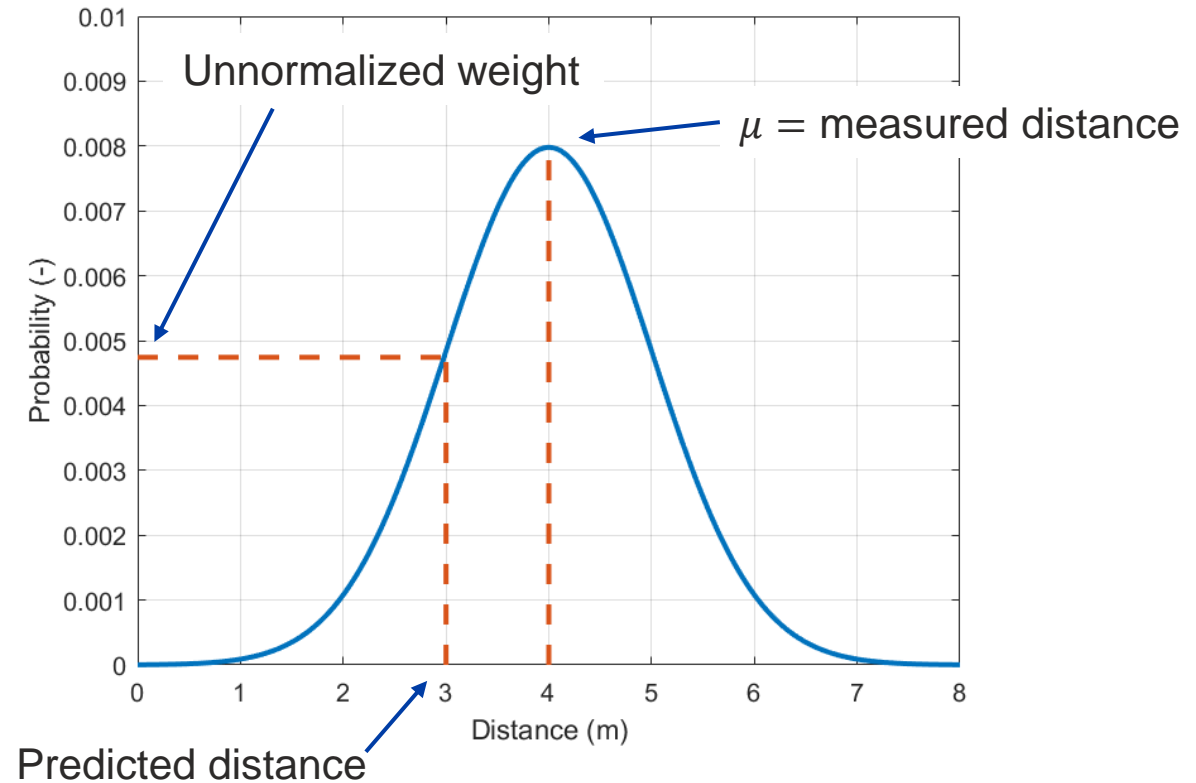
- Application of Bayes' rule
 - Prior: all particles have the same weight $1 / N$
 - Posterior: proportional to the measurement model
- Weights should be normalized
- Examples for rangefinders:
 - Using normal distribution (σ)

$$w \propto \prod_{m=1}^M e^{-\frac{1}{2} \left(\frac{d_m - p_m}{\sigma} \right)^2}$$

- Using Euclidean distance

$$w \propto \frac{1}{\sqrt{\sum_{m=1}^M (d_m - p_m)^2}}$$

d ... measured distance
 p ... predicted distance
 M ... number of measurements



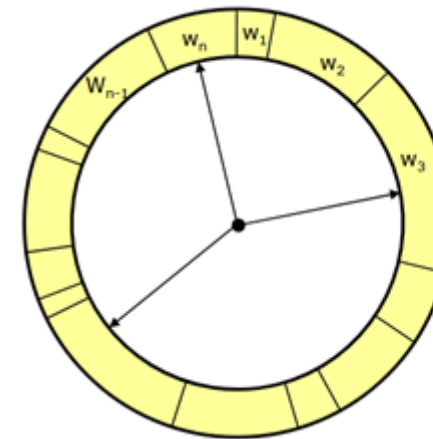
Correction

$$\forall [x, w] \in \chi: w_t^{(i)} = f(x_t^{(i)}, z_t) \propto p(z_t | x_t)$$

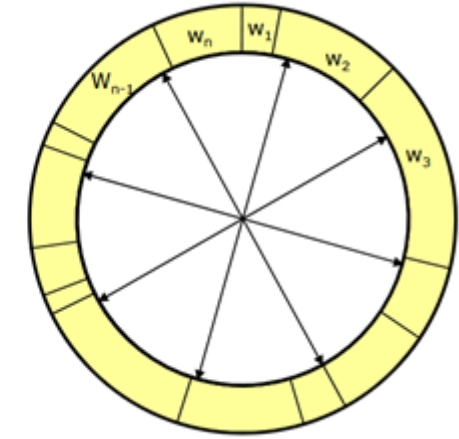


- Increase density of particles in regions of high posterior probability and vice versa
- Needed in case of limited number of samples
- Draw N particles with the probability given by weights of original set
- General algorithm:
 1. Generate sorted set of N random numbers u_k in range $(0, 1)$
 2. Compute cumulative sum of weights
 3. For each u_k pick particle x_i according to condition:

$$u_k \in \left(\sum_{s=1}^{i-1} w_s, \sum_{s=1}^i w_s \right)$$



Roulette wheel



Systematic

[1]

Resampling

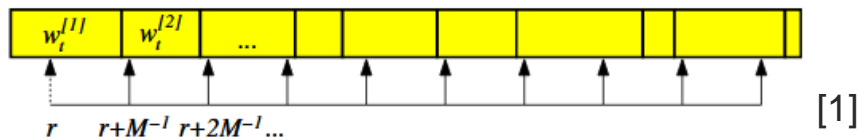
$$\chi_t \sim [x_t, w_t]$$

$$p(x_t) \propto w_t$$



- Low variance systematic resampling
 - Only one iteration through the weights set
 - Keeps particles of even weights alive
 - Generate random number \tilde{u} in range $(0, \frac{1}{N})$

$$u_k = \frac{(k-1) + \tilde{u}}{N}$$



- Thrun's heuristic algorithm
 - Higher degree of randomness
 - Easy implementation

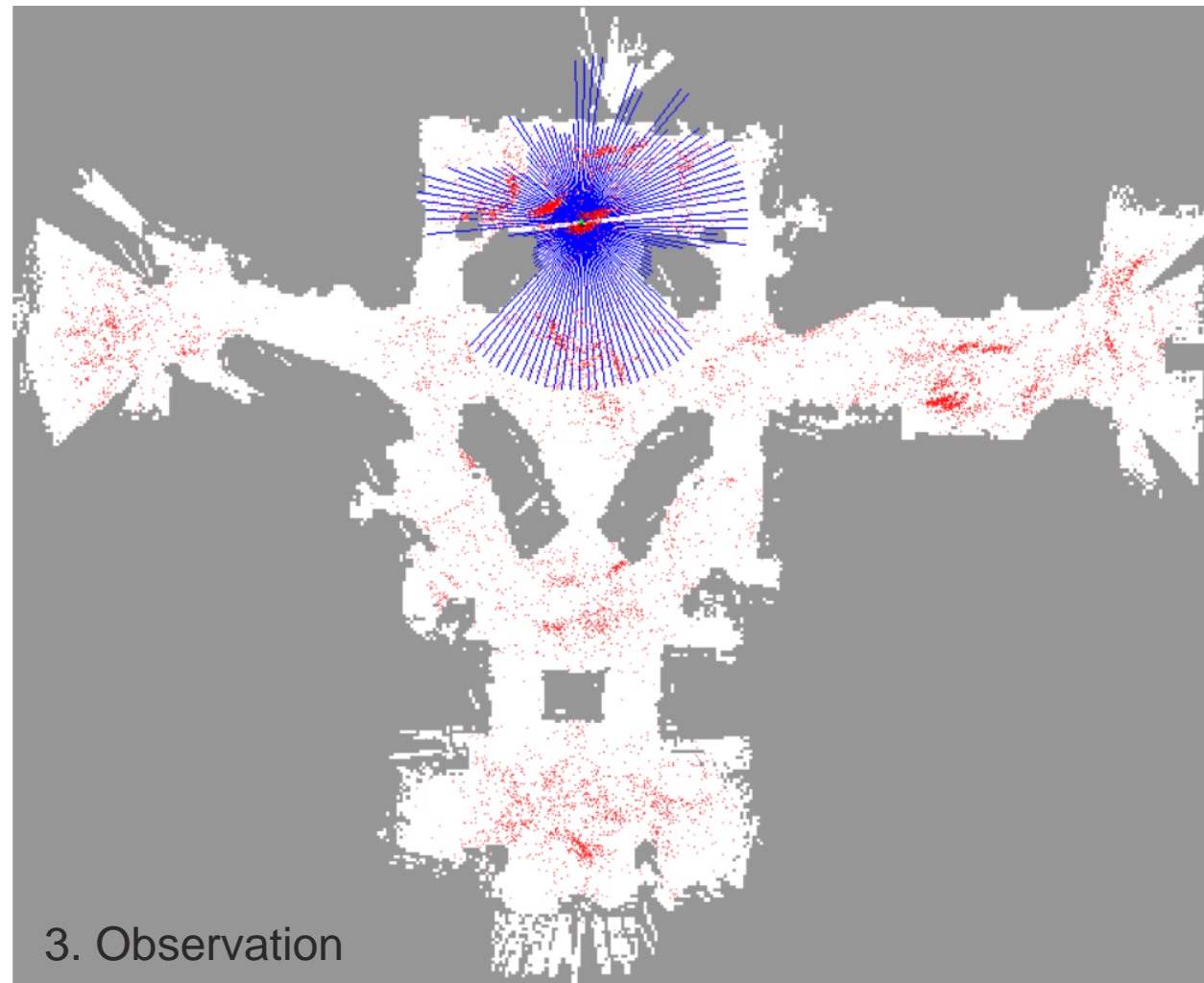
1	<i>index</i> = rand(0, <i>N</i> -1)
2	for <i>i</i> = 1 to <i>N</i> do
3	<i>beta</i> = rand(0, 2 <i>w</i> _{max})
4	while <i>w</i> [<i>index</i>] < <i>beta</i>
5	<i>beta</i> = <i>beta</i> – <i>w</i> [<i>index</i>]
6	<i>index</i> ++
7	if <i>index</i> > <i>N</i>
8	<i>index</i> = 1
9	<i>new_particles</i> [<i>i</i>] = <i>particles</i> [<i>index</i>]



1. Initialization

2. Prediction

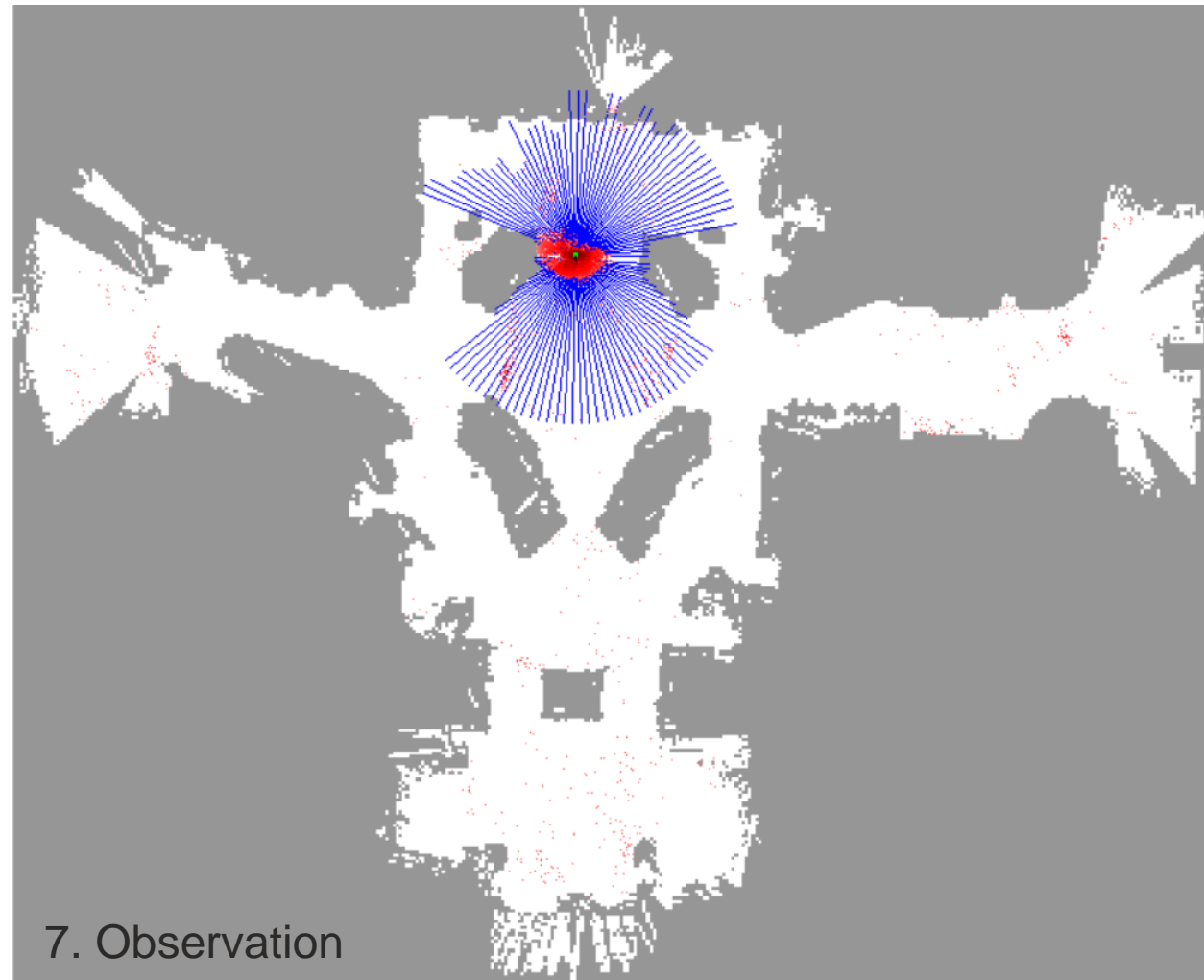
[1]



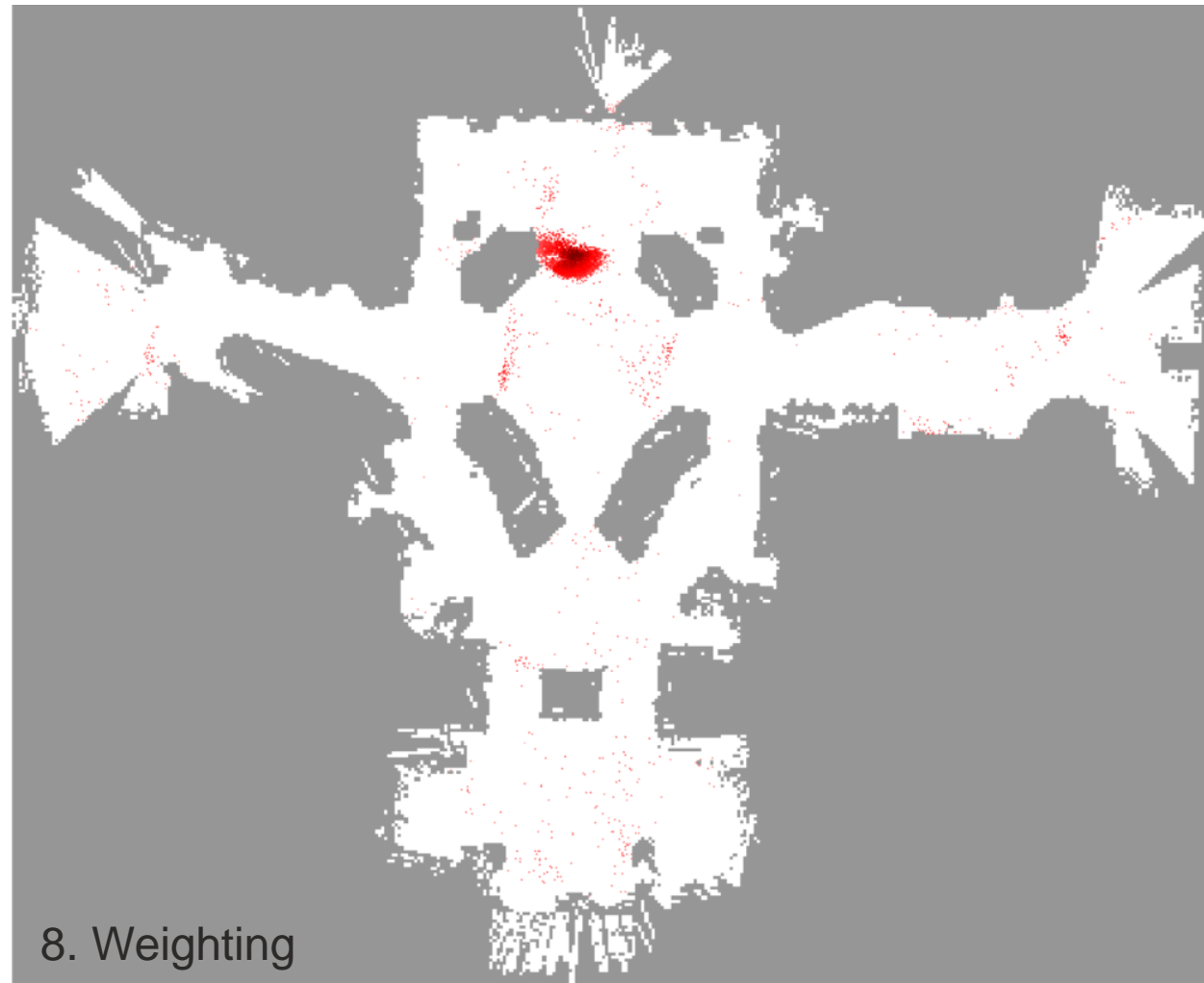


5. Resampling

6. Prediction



7. Observation



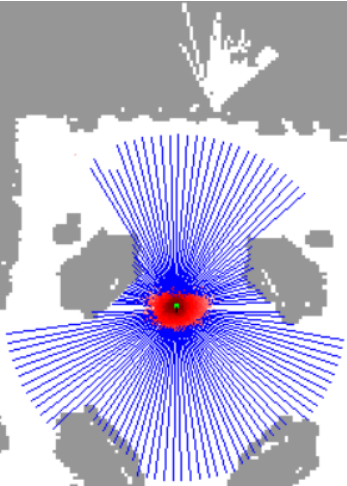
8. Weighting



9. Resampling

10. Prediction

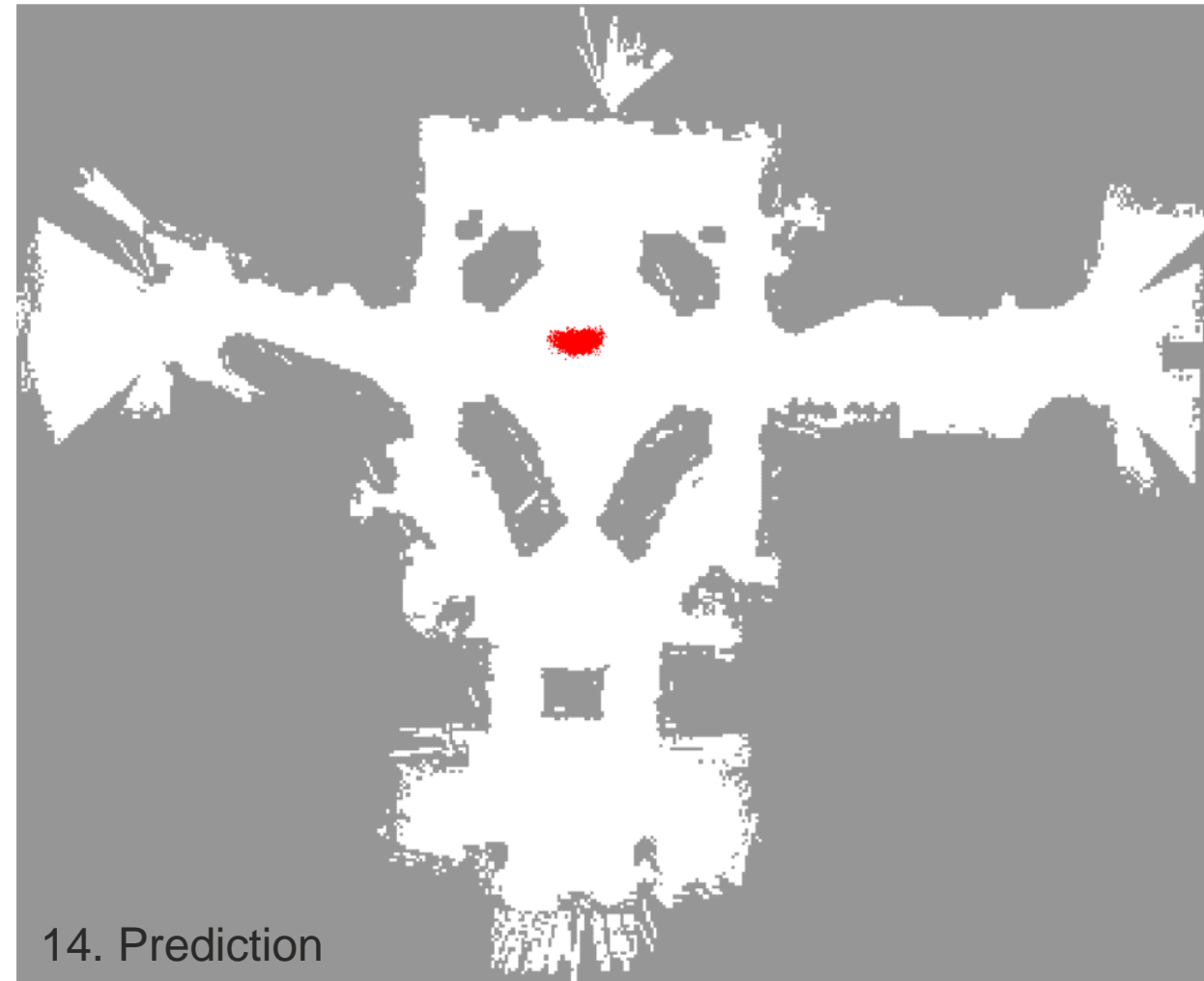
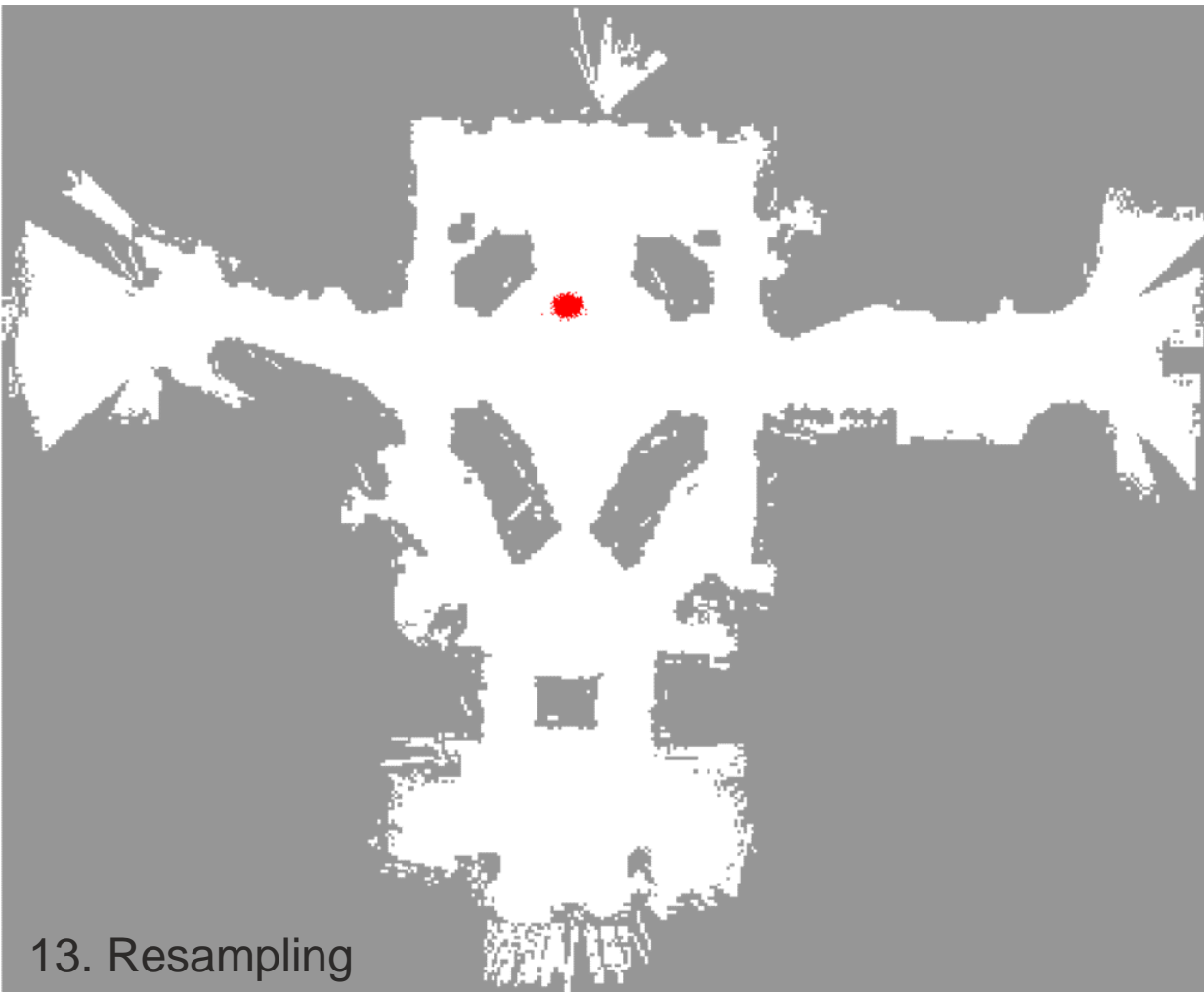
[1]



11. Observation



12. Weighting

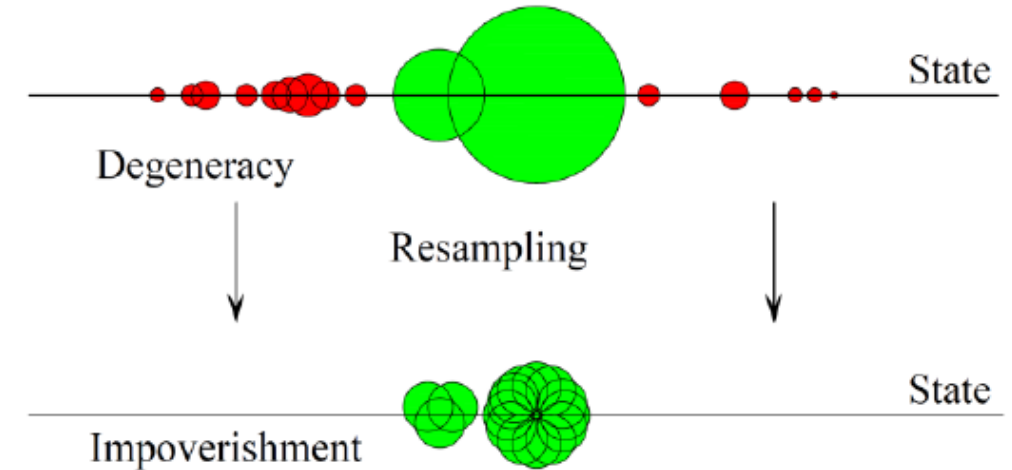




Particle_filter(χ_{t-1}, u_t, z_t):		
1	$\bar{\chi}_t = \chi_t = \emptyset$	
2	for $n = 1$ to N do	
3	sample $x_t^{(n)} \sim p(x_t u_t, x_{t-1}^{(n)})$	prediction step
4	$w_t^{(m)} = p(z_t x_t^{(n)})$	compute weight (correction)
5	$\bar{\chi}_t \leftarrow [x_t^{(n)}, w_t^{(m)}]$	keep list of weighted original particles
6	$r = \text{rand}(0, 1/N)$	low variance resampling algorithm
7	$c = w_t^{(1)}, i = 1$	
8	for $n = 1$ to N do	
9	$u = r + (n - 1)/N$	
10	while $u > c$	
11	$c = c + w_t^{(++i)}$	
12	$\chi_t \leftarrow x_t^{(i)}$	
13	return χ_t	



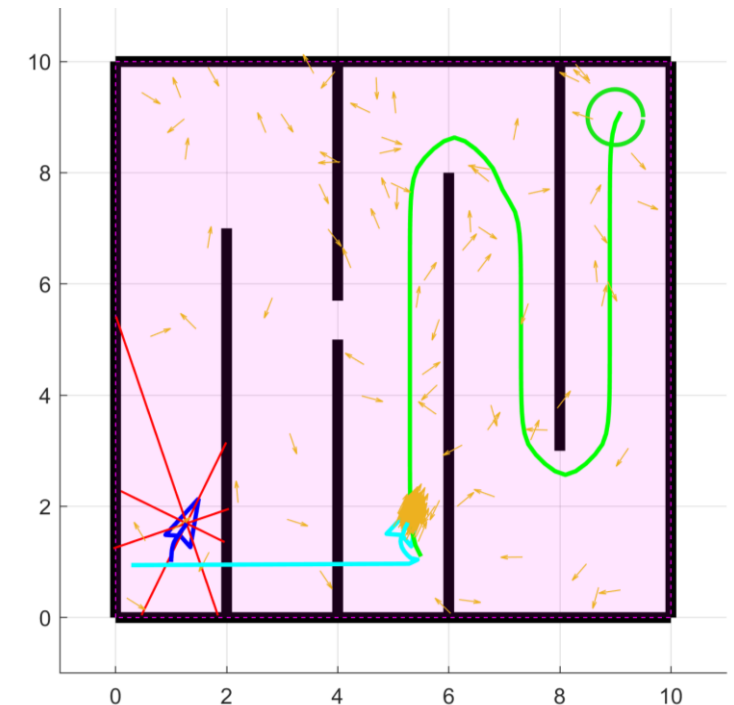
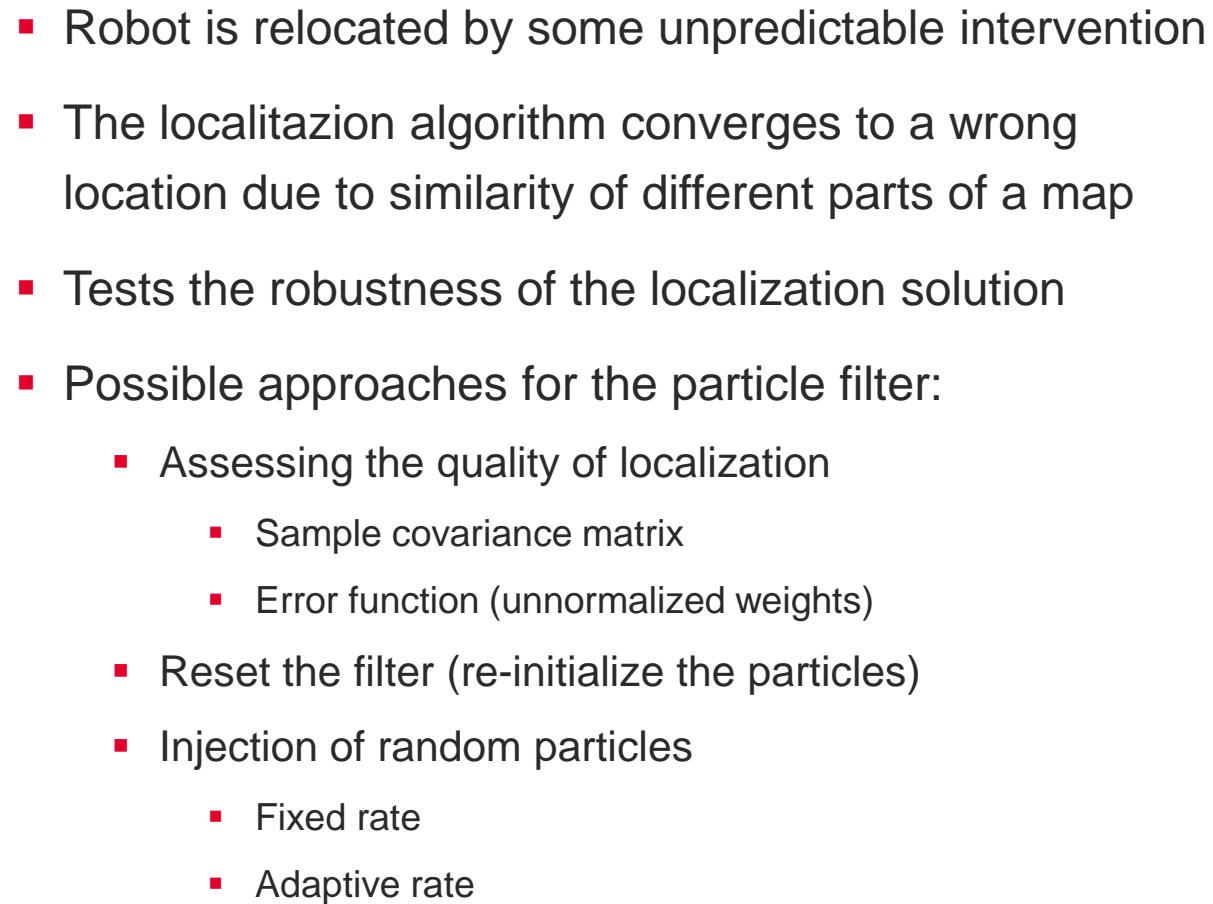
- Each resampling step results in so-called particle degeneracy and impoverishment
- Particles with low probability are eliminated while particles with large weights exist in too many copies
- How to address the degeneracy issue?
 - It is essential to increase the variance of the particle set
 - Intensify noise in the prediction step
 - Do not resample in each iteration
 - Add Gaussian noise in the resampling step (= regularized particle filters [2])



[1]

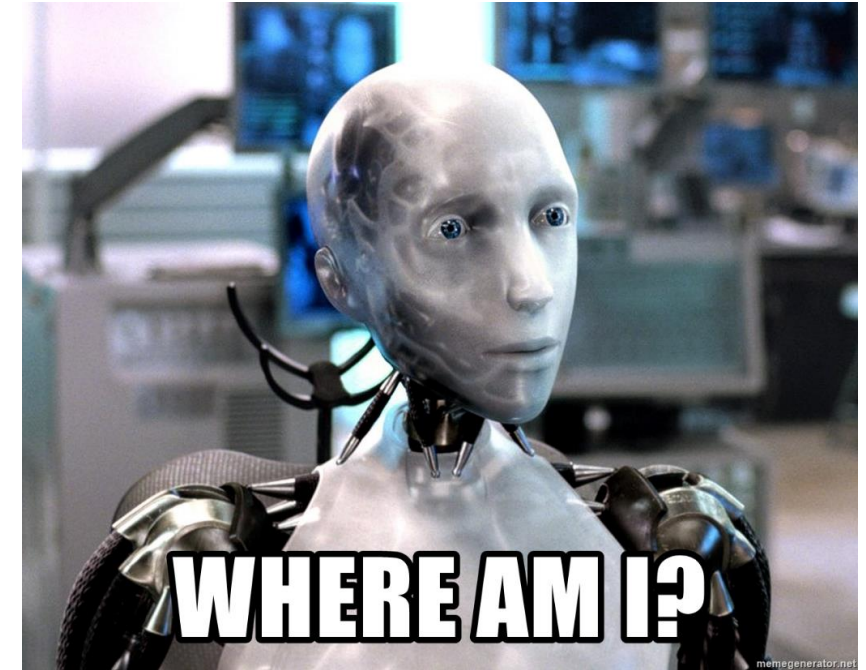
$$x_t^{(i)} = x_t^{(i)} + h\Gamma_t\varepsilon$$

h	... Bandwidth
Γ_t	... Square root of empirical covariance matrix
ε	... Random vector drawn from Gaussian kernel





- Particle filter
 - Non-parametric recursive Bayes filter
 - Approximates the posterior by weighted samples
 - Can handle non-Gaussian PDFs and non-linear transitions
 - Basic principles: Importance sampling and Survival-of-the-fittest
- Monte Carlo localization (MCL)
 - Based on the particle filter
 - Prediction: Applying the motion model to particles
 - Correction: Likelihood of observations
 - Easy implementation
 - Accuracy and robustness depends on the quality of motion and measurement models
 - Standard for mobile robots localization



[1]



Tomas Lazna

tomas.lazna@ceitec.vutbr.cz

Brno University of Technology
Faculty of Electrical Engineering and Communication
Department of Control and Instrumentation



Robotics and AI Research Group