Intelligent Robotics Systems

Armin Biess

Probabilistic State Estimation II

Particle Filters

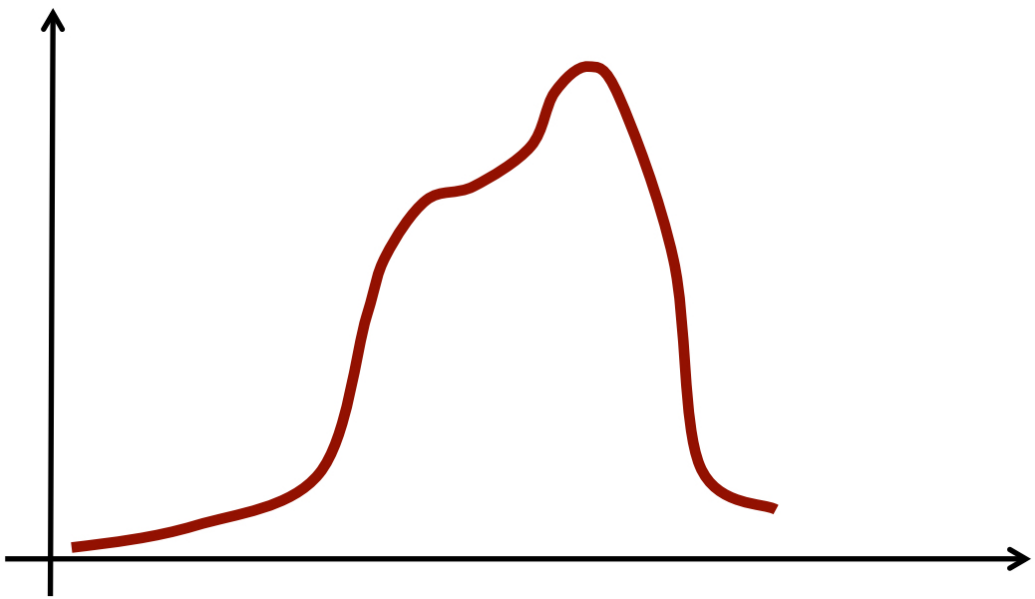
Gaussian Filters

- The Kalman filter and its variants can only model **Gaussian distributions**

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

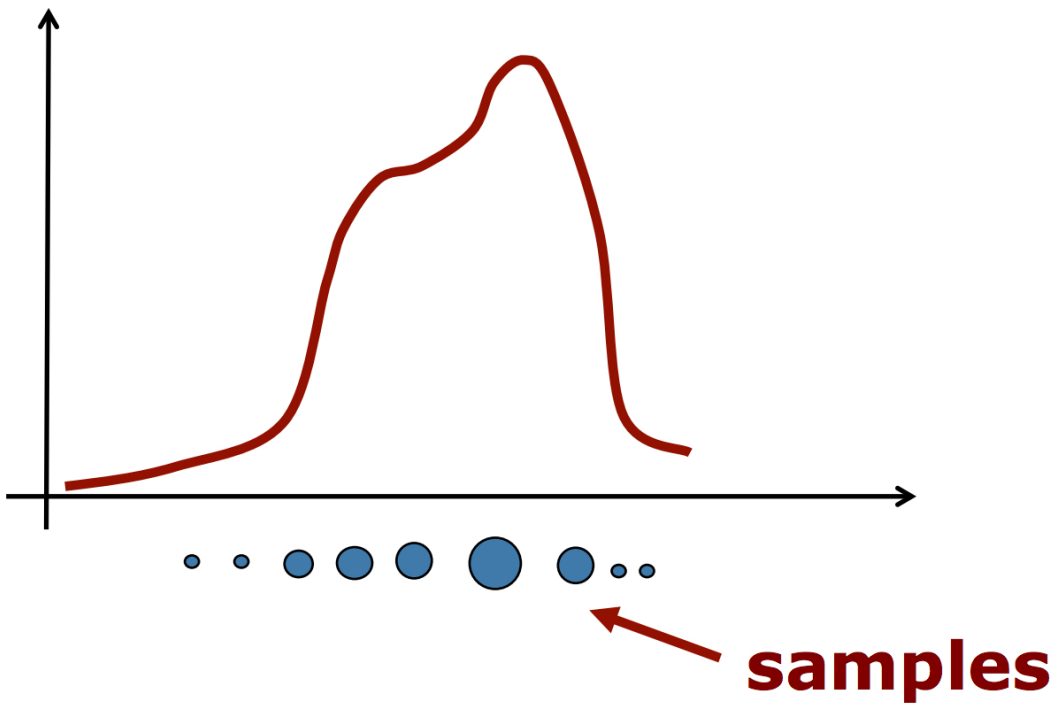
Motivation

- Goal: develop approach for dealing with **arbitrary distributions**



Key idea: use samples

- Use **multiple samples** to represent arbitrary distributions



Particle set

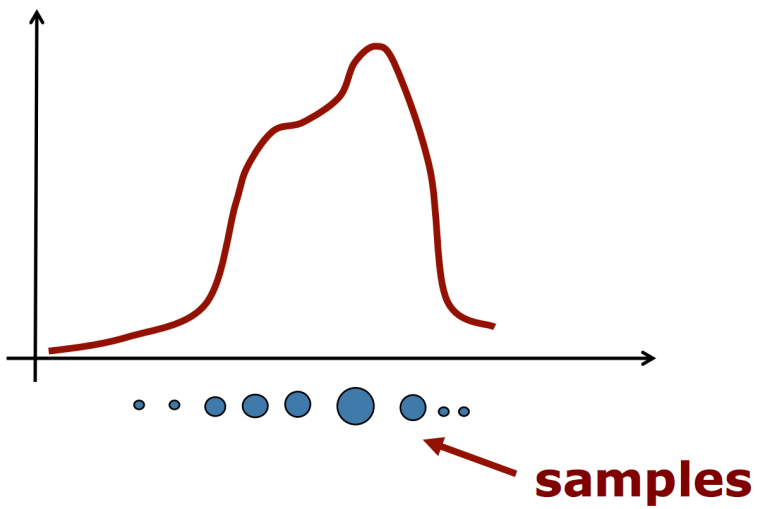
- Set of weighted samples

$$\begin{aligned}\mathcal{X} &= \left\{\langle x^{[m]}, w^{[m]}\rangle\right\}_{m=1,\ldots,M} \\ &= \left\{\langle x^{[1]}, w^{[1]}\rangle, \langle x^{[2]}, w^{[2]}\rangle, \ldots, \langle x^{[M]}, w^{[M]}\rangle\right\}\end{aligned}$$

$x^{[m]}$: **state hypothesis**
$w^{[m]}$: **importance weight**

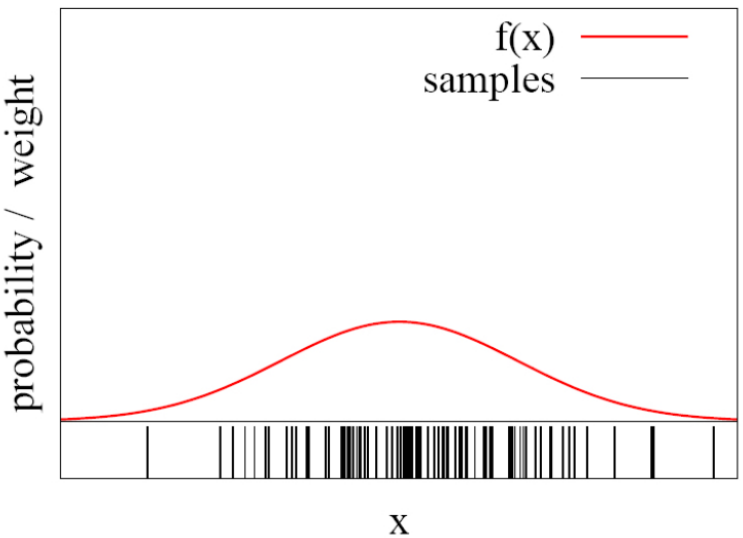
- The samples represent the posterior distributions

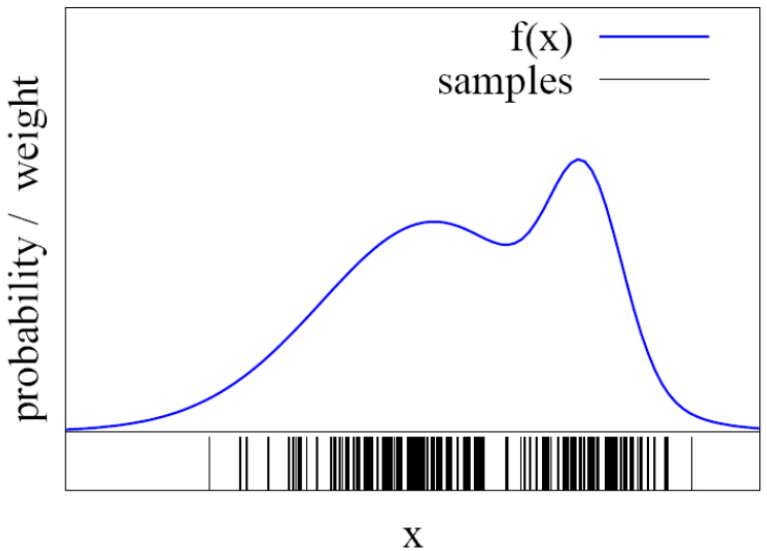
$$p(x) = \sum_{m=1,\ldots,M} w^{[m]} \cdot \delta(x - x^{[m]}), \quad \delta(x) : \text{Dirac-Delta function}$$



Particles for approximation

- Particles for function approximation

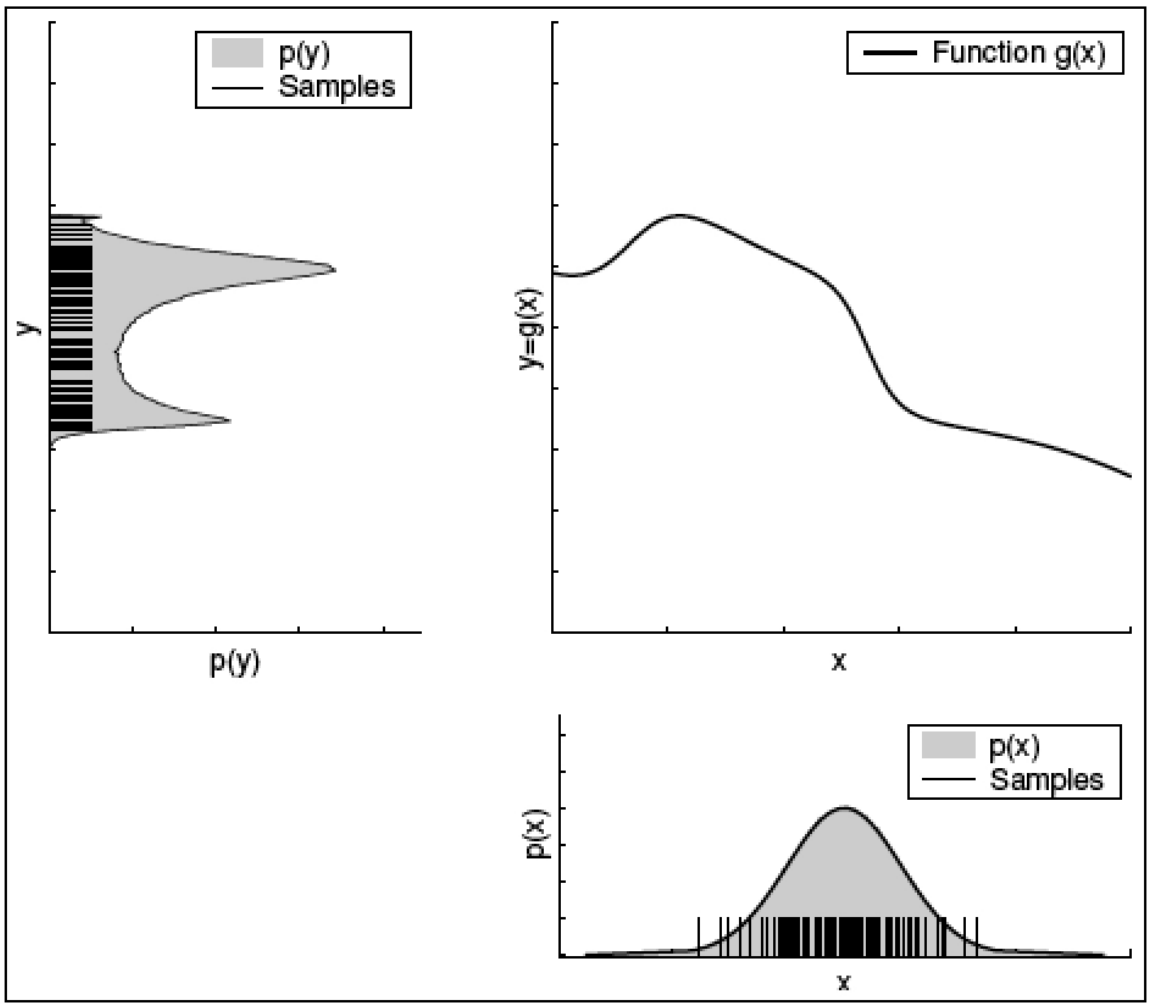




- The more particles fall into an interval, the higher its probability density

Particles for approximation

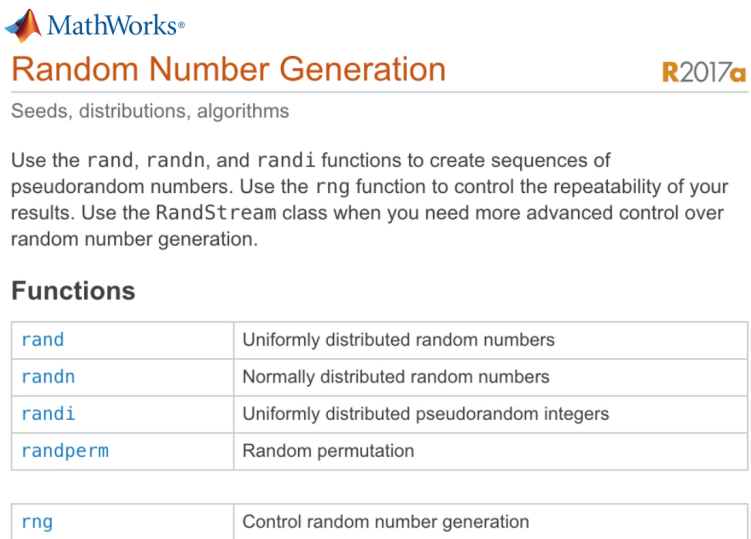
particle representations of pdfs enable an easy implementation of nonlinear transformations of random variables



How to get these samples?

- Computer programs can generate usually random numbers for two distributions:
  uniform distribution
  normal (Gaussian) distribution

  for example, Matlab:



MathWorks®

Random Number Generation R2017a

Seeds, distributions, algorithms

Use the `rand`, `randn`, and `randi` functions to create sequences of pseudorandom numbers. Use the `rng` function to control the repeatability of your results. Use the `RandStream` class when you need more advanced control over random number generation.

Functions

`rand`	Uniformly distributed random numbers
`randn`	Normally distributed random numbers
`randi`	Uniformly distributed pseudorandom integers
`randperm`	Random permutation

`rng`	Control random number generation

  $\Rightarrow$ we can get samples easily for the uniform and Gaussian distributions

- what about other distributions from which we do not know how to sample?

Importance Sampling Principle

- we can use a different distribution g, from which we know how to draw samples, to generate samples from f
- account for the "differences between g and f" using a **weight** $w = f/g$
- target f
- proposal g
- pre-condition: $f > 0 \Rightarrow g > 0$
  (the condition guarantees that samples from f can be generated from samples from g for all state where $f > 0$)

Particle filter

- Recursive Bayes filter
- Non-parametric filter
- Model the distribution by samples
- Prediction: draw samples from the proposal distribution
- Measurement update/correction: weighting by the ratio of target and proposal

Note: The more sample we use, the better is the estimate ($N \geq 100$)

Particle filter for pdf approximation

three steps:

1. Sample the particles using the proposal distribution

$$x_t^{[m]} \sim proposal(x_t)$$

2. Compute the importance weights

$$w_t^{[m]} = \frac{target(x_t)}{proposal(x_t)}$$

3. Resampling: "Replace unlikely samples by more likely ones"

Particle filter - Algorithm: version 1

Algorithm Particle_filter $(\mathcal{X}_{t-1}, proposal,\ target)$:

$\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$

for $m = 1$ to M do

  sample $x_t^{[m]} \sim proposal(x_t^{[m]})$ — 1. Proposal

  $w_t^{[m]} = target(x_t^{[m]}) / proposal(x_t^{[m]})$ — 2. Importance weight

  $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$

endfor

for $m = 1$ to M do — 3. Resampling

  draw i with probability $\propto w_t^{[i]}$

  add $x_t^{[i]}$ to $\mathcal{X}_t$

endfor

return $\mathcal{X}_t$

Particle filter - Example

Given:

$$\begin{aligned} proposal(x) &= const\,,\ x \in (-5, 15)\,, \quad \text{(uniform)} \\ target(x) &= p \cdot \mathcal{N}(x|4, 2) + (1-p) \cdot \mathcal{N}(x|9, 1) \quad \text{with } p = 0.5 \\ &\quad \text{(Gaussian Mixture Model (GMM))} \end{aligned}$$

Wanted: samples of $target(x)$

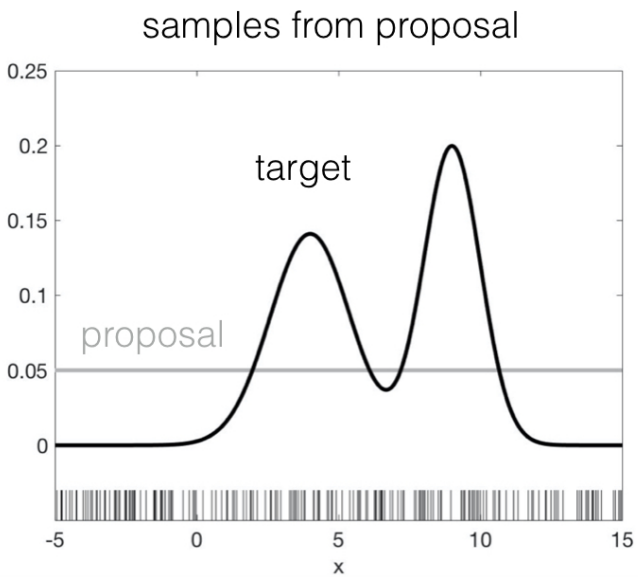
remark: Gaussian Mixture Model = superposition of Gaussian distributions

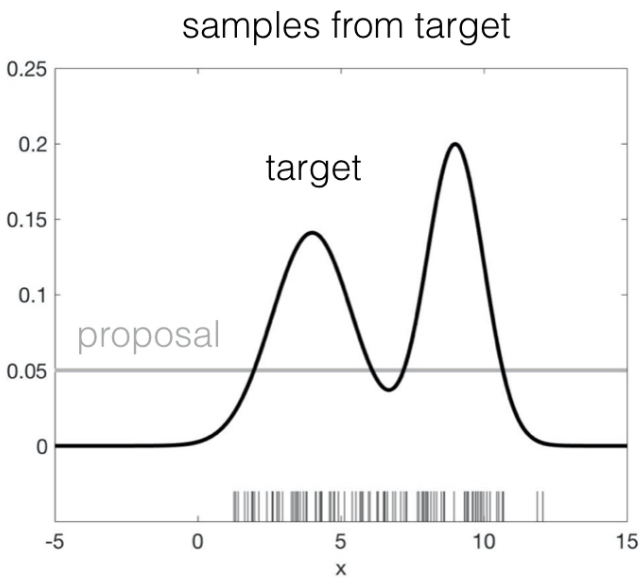
$$p(x) = \sum_{i=1}^{N} p_i \cdot \mathcal{N}(x|\mu_i, \sigma_i^2)\,, \quad p_i \text{ : mixing parameters}$$

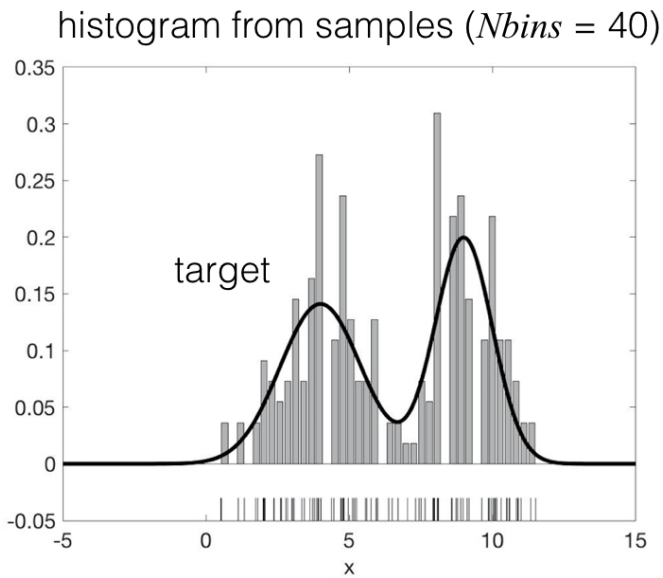
since $\int_{-\infty}^{\infty} \mathcal{N}(x|\mu_i, \sigma_i^2)dx = 1$ it is $\sum_{i=1}^{N} p_i = 1$

Particle filter - Example

$N = 200$ particles

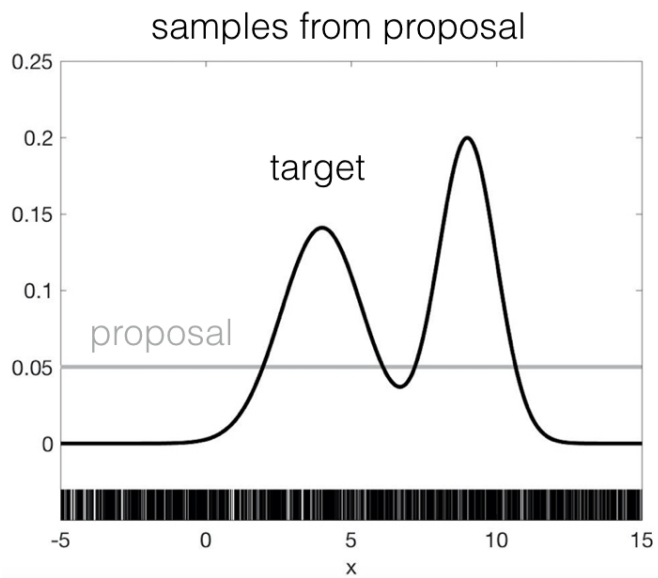


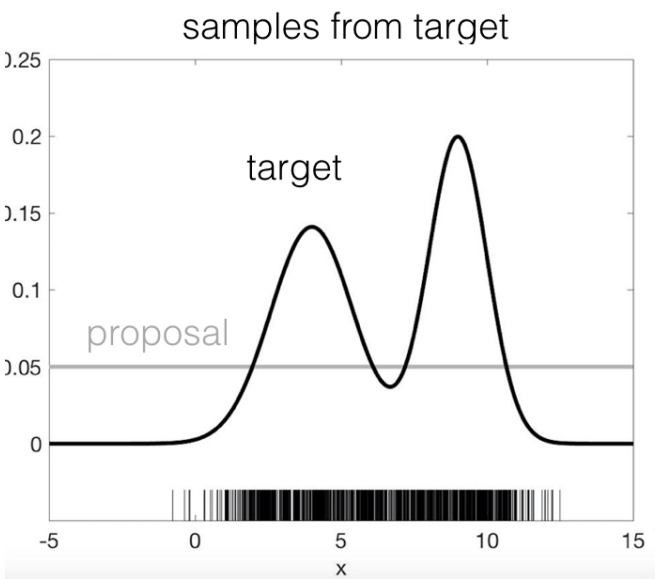


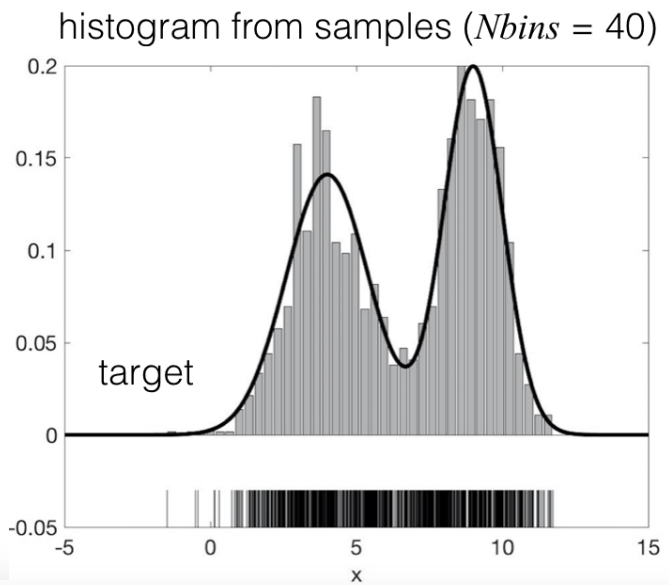


Particle filter - Example

$N = 2000$ particles

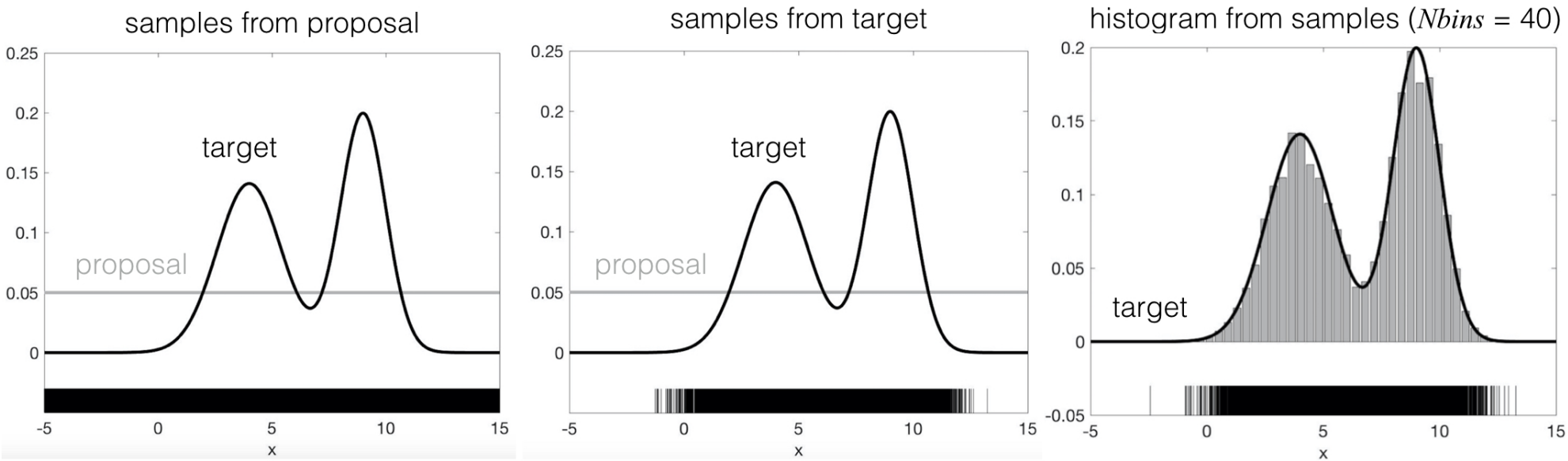






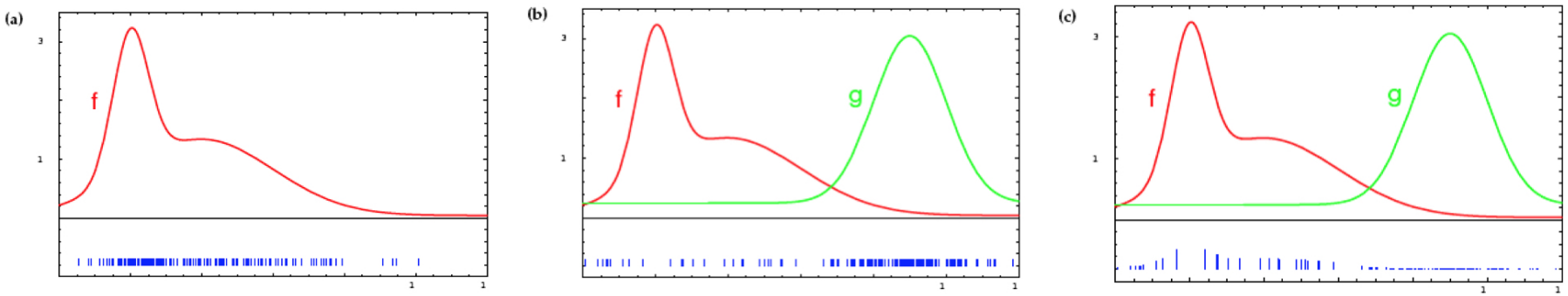
Particle filter - Example

$N = 20000$ particles



remark: the problem of extracting a continuous density from samples is called **density estimation**

Particle filter - another example

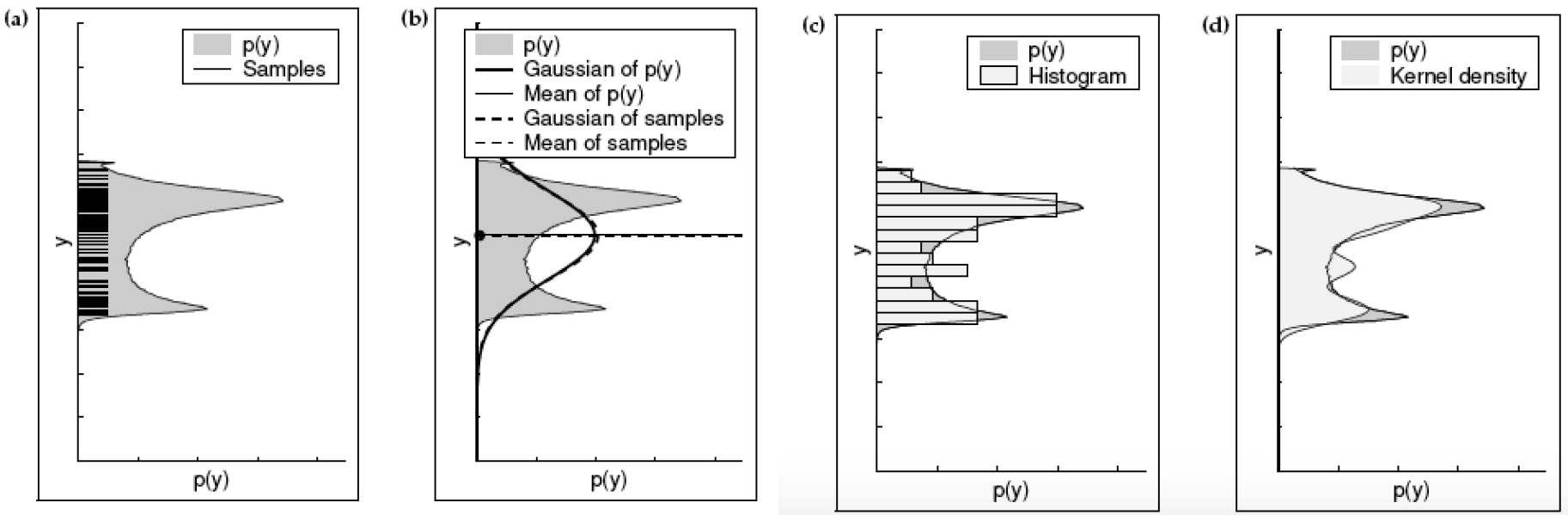


- samples of f are obtained by attaching the weight $w = f(x)/g(x)$ to each sample x drawn from g
- condition: density g must be such that $f(x) > 0$ implies $g(x) > 0$
- f corresponds to the belief $bel(x_t)$ and g to the belief $\bar{bel}(x_t)$

Particle filter - Density estimation from particles

other methods:

- Gaussian
- Histogram
- Kernel methods



Particle filter for state estimation

- proposal is the **motion model** (**state transition model**)

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

- measurement update (correction) via **measurement model**

$$w_t^{[m]} \sim p(z_t|x_t)$$

remark:

$g = p(x_t|x_{t-1}, u_t)bel(x_{t-1}) = \bar{bel}(x_t)$ is the proposal distribution
$f = \eta p(z_t|x_t) \cdot \bar{bel}(x_t) = bel(x_t)$ is the target distribution

Particle filter - Algorithm: version 2

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 $\mathcal{X}_t$

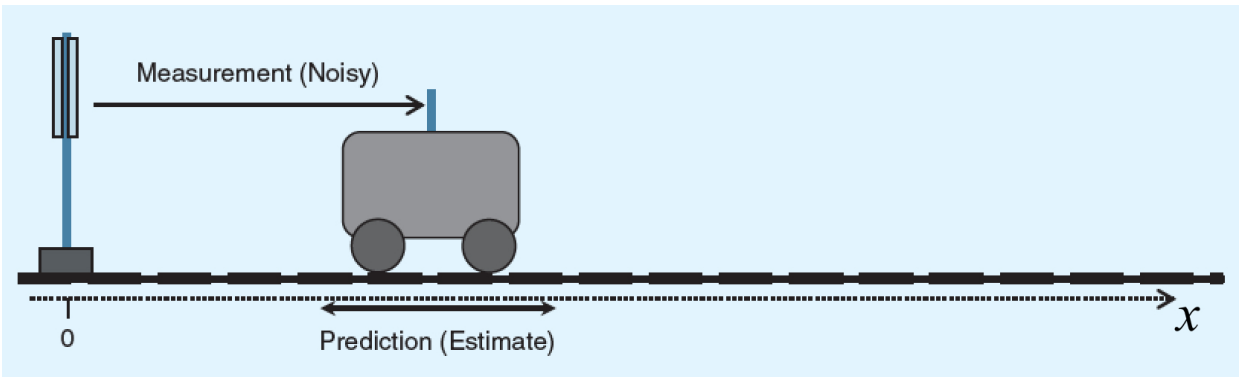
1. prediction (motion update) $= \bar{bel}(x_t)$
2. measurement update (weight update) $= bel(x_t)$
3. Resampling

Particle filter - Example

Given: initial belief $bel(x_0)$
motion model (state transition model): $p(x_t|x_{t-1}, u_t)$
measurement model: $p(z_t|x_t)$
measurements z_t and actions u_t

Wanted: belief $bel(x_t)$

We will implement a particle filter for state estimation of a car for a single time step using the tracking problem that we have considered in lecture 2:



but now we add another spatial dimension y ...

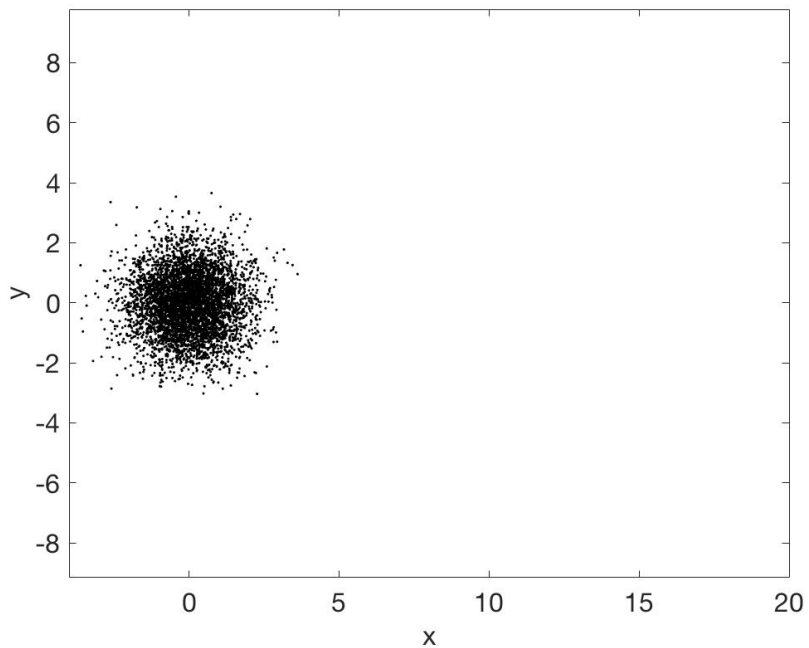
Particle filter - Example

$M = 5000$ particles, state $\boldsymbol{x}_t = (x_t, y_t)$

1. initial belief: $bel(\boldsymbol{x}_0) = \mathcal{N}(x_0|0,1) \cdot \mathcal{N}(y_0|0,1)$

draw M samples from intial belief $bel(\boldsymbol{x}_0)$: $\boldsymbol{x}_0^{[m]} \sim bel(\boldsymbol{x}_0)$, i.e.,

$x_0^{[m]} \sim \mathcal{N}(x_0|0,1)$ and $y_0^{[m]} \sim \mathcal{N}(y_0|0,1)\,,\ m = 1, \ldots, M$



Every particle represents a possible location (hypothesis) of the

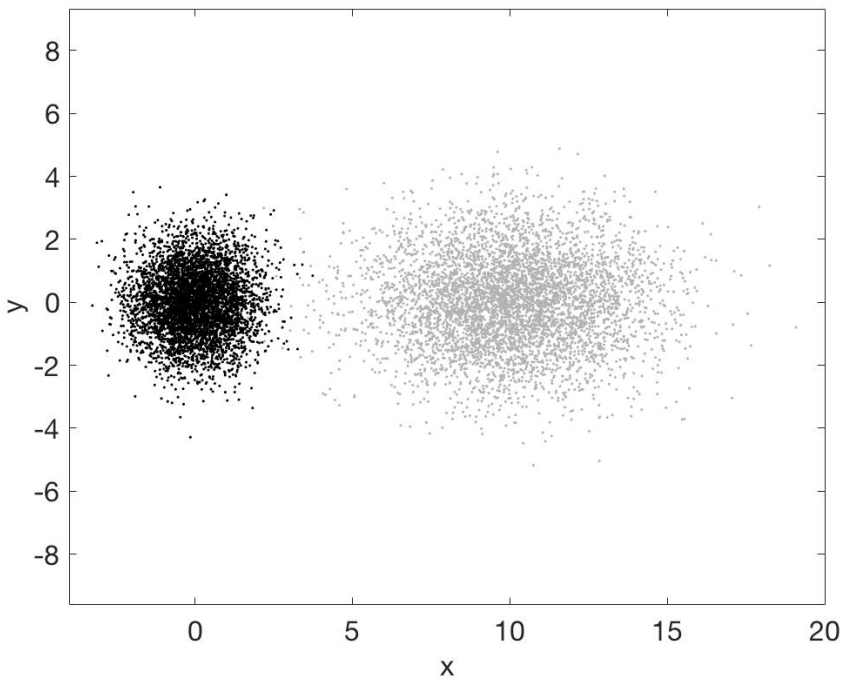
Particle filter - Example

2. prediction step: motion model $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{u}_t)$ and motion command $\boldsymbol{u}_t = (u_t, v_t)$

$$\begin{aligned} p(\boldsymbol{x}_1|\boldsymbol{x}_0, \boldsymbol{u}_1) &= \frac{1}{\sqrt{2\pi r_x^2}} \exp\left[-\frac{(x_1-(x_0+u_1\Delta t))^2}{2r_x^2}\right] \times \frac{1}{\sqrt{2\pi r_y^2}} \exp\left[-\frac{(y_1-(y_0+v_1\Delta t))^2}{2r_y^2}\right] \\ &= \mathcal{N}(x_1|x_0+u_1\Delta t, r_x^2) \cdot \mathcal{N}(y_1|y_0+v_1\Delta t, r_y^2) \end{aligned}$$

motion command : $\boldsymbol{u}_1 = (u_1, v_1) = (10, 0)\,[\text{m/s}],\ \Delta t = 1\,[\text{s}]$

draw M samples from motion model: $\boldsymbol{x}_1^{[m]} \sim p(\boldsymbol{x}_1|\boldsymbol{x}_0^{[m]}, \boldsymbol{u}_1)$



$$r_x^2 = 4$$
$$r_y^2 = 1$$

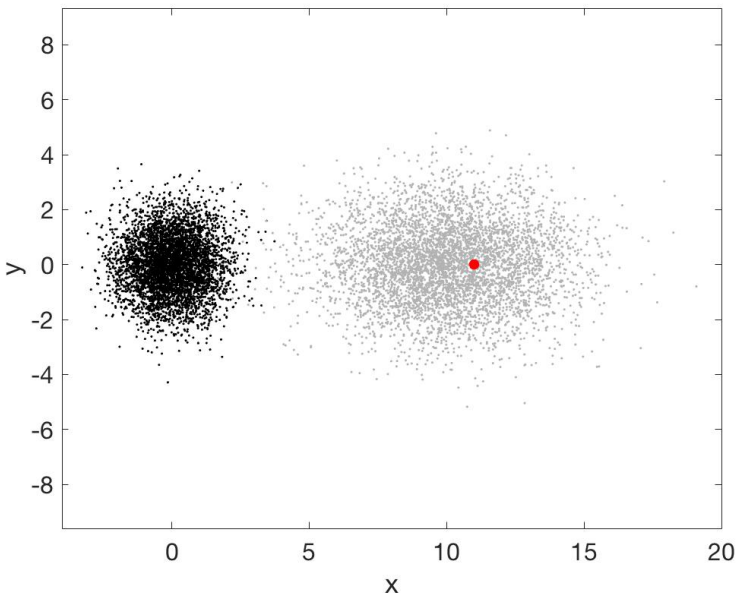
Particle filter - Example

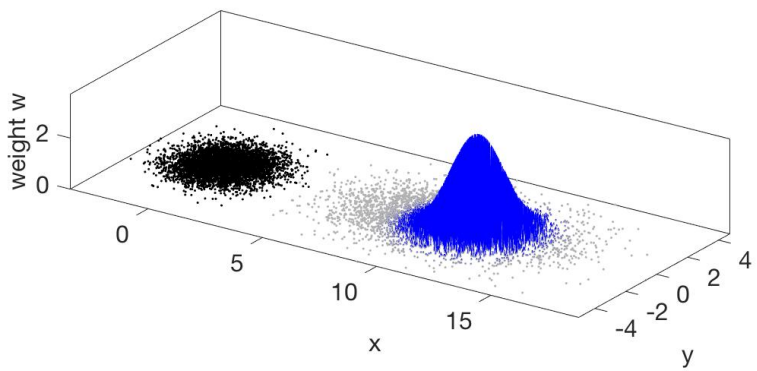
2. measurement update: measurement model $p(\boldsymbol{z}_t|\boldsymbol{x}_t)$ and measurement $\boldsymbol{z}_t = (\alpha_t, \beta_t)$

$$
\begin{aligned}
p(\boldsymbol{z}_1|\boldsymbol{x}_1) &= \frac{1}{\sqrt{2\pi\sigma_x^2}} \exp\left[-\frac{(x_1-\alpha_1)^2}{2\sigma_x^2}\right] \times \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left[-\frac{(y_1-\beta_1)^2}{2\sigma_y^2}\right] \\
&= \mathcal{N}(x_1|\alpha_1, \sigma_x^2) \cdot \mathcal{N}(y_1|\beta_1, \sigma_y^2)
\end{aligned}
$$

measurement : $\boldsymbol{z}_1 = (\alpha_1, \beta_1) = (11, 0)\,[\text{m}]$,

weight update: $\boldsymbol{w}_1^{[m]} = p(\boldsymbol{z}_1|\boldsymbol{x}_1^{[m]})$ (shown in blue as heights)

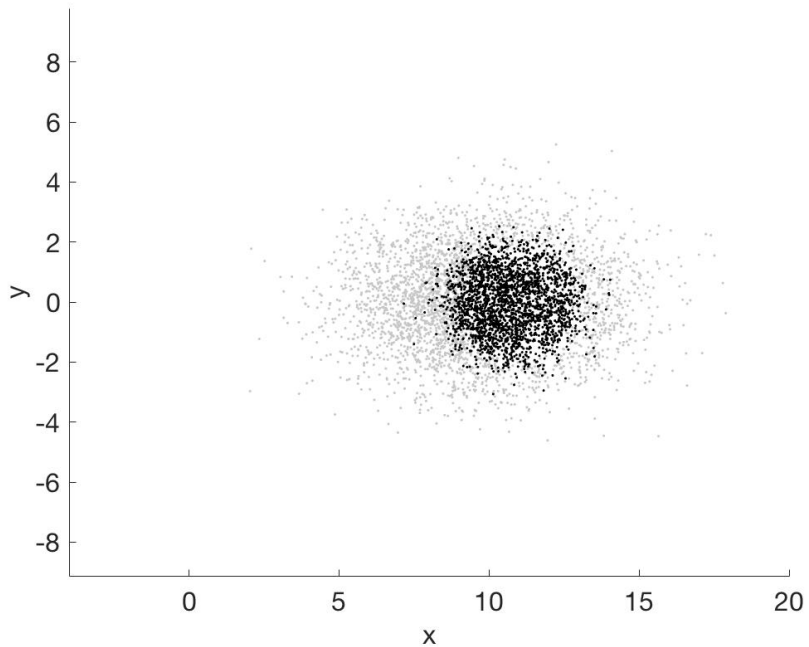




$\sigma_x^2 = 1$
$\sigma_y^2 = 1$

Particle filter - Example

3. resampling:



we implemented one time step $t = 1$ of the particle filter, continue in a similar way for $t = 2, 3, \ldots, T$.

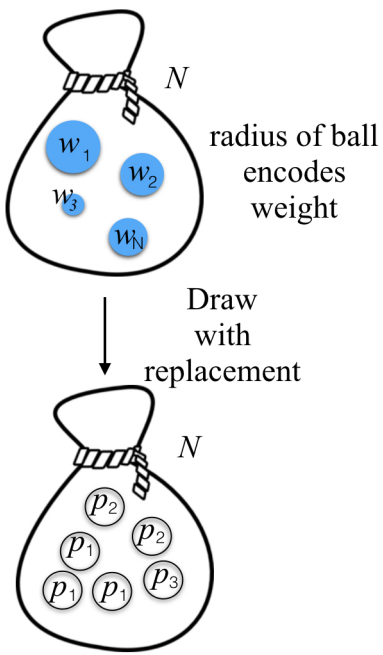
Q: How to implement the resampling step?

Resampling

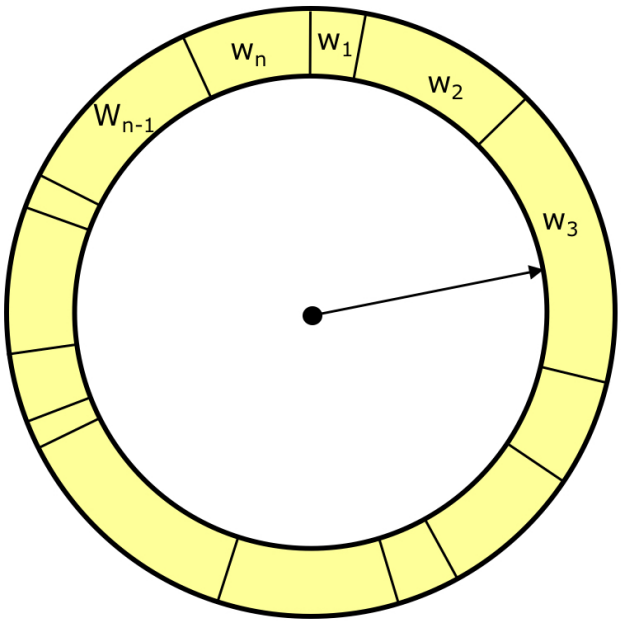
- **Given:** Set S of weighted samples
- **Wanted:** Random sample, where the probability of drawing particle i (or x_i) is given by w_i
- Typically done n times with replacement to generate new sample set S'

$$\tilde{w}^i = p(z|x^i)$$
$$w^i = \frac{\tilde{w}^i}{\sum_i \tilde{w}^i}$$

particle	importance weight	prob.
p_1	$\tilde{w}^1$	w^1
p_2	$\tilde{w}^2$	w^2
$\vdots$	$\vdots$	$\vdots$
p_N	$\tilde{w}^N$	w^N
		$\sum_{i=1}^N w^i = 1$

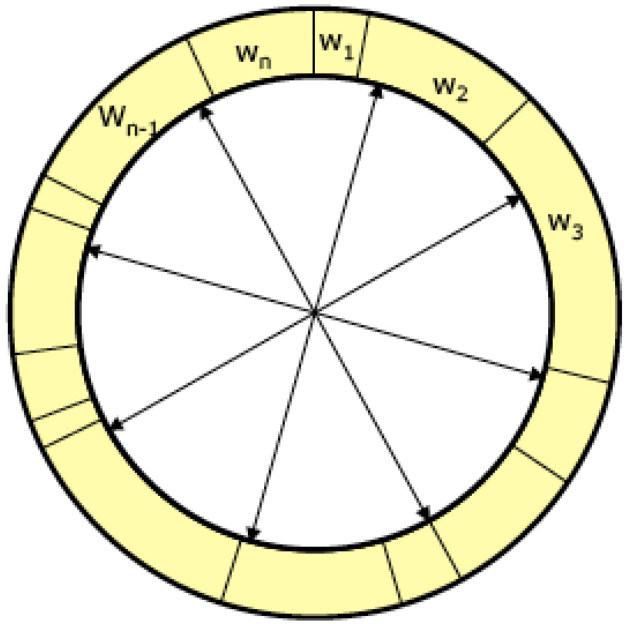


Resampling Algorithm



- roulette wheel
- binary search
- complexity: $O(N \log N)$

  [N random numbers: $O(N)$,
  binary search for each particle for each random number: $O(\log N)$]

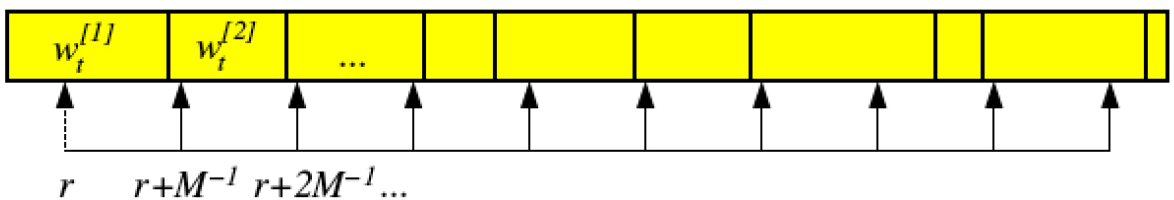


- stochastic universal sampling
- low variance
- complexity: $O(N)$

Low Variance Resampling Algorithm

Low_variance_resampling$(\mathcal{X}_t, \mathcal{W}_t)$**:**

$\bar{\mathcal{X}}_t = \emptyset$
$r = \text{rand}(0; M^{-1})$
$c = w_t^{[1]}$
$i = 1$
for $m = 1$ *to* M *do*
  $U = r + (m-1) \cdot M^{-1}$
  while $U > c$
    $i = i + 1$
    $c = c + w_t^{[i]}$
  endwhile
  add $x_t^{[i]}$ *to* $\bar{\mathcal{X}}_t$
endfor
return $\bar{\mathcal{X}}_t$



choose a random number r and select those particles that correspond to $U = r + (m-1) \cdot M - 1$ where $m = 1, \ldots, M$ and M is the number of samples to be drawn at time t.

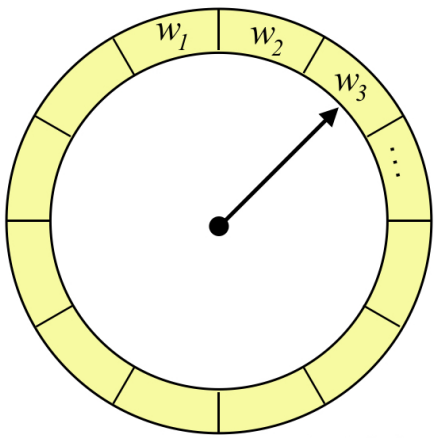
Resampling

- resampling step is a probabilistic implementation of the Darwin idea of **survival of the fittest**: "Replace unlikely samples by more likely ones"
- "trick" to avoid that many samples cover unlikely states
- needed as we have a limited number of samples

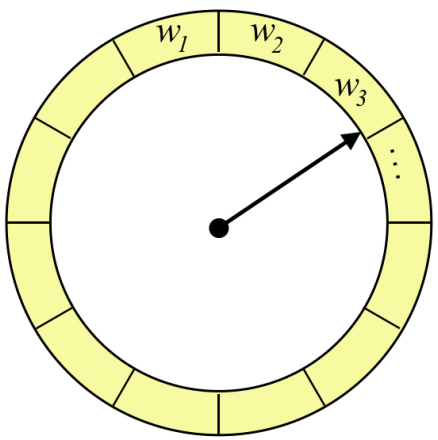
Why is it a low variance resampling algorithm?

assume all weights are equal, $M = 12$ particles, run experiment A $N_A = 1200$ times and experiment B $N_B = 100$ times

A

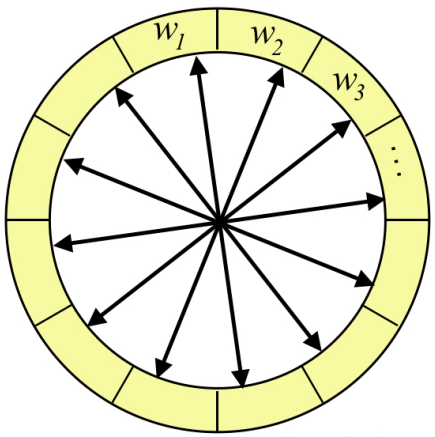


1. trial

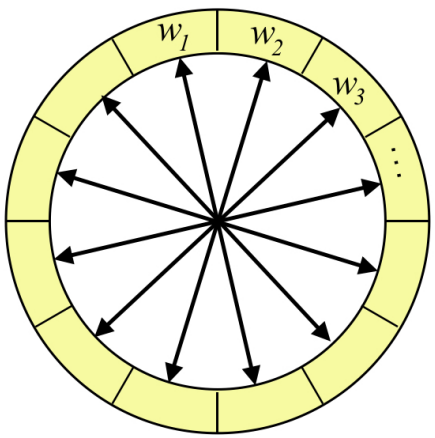


2. trial

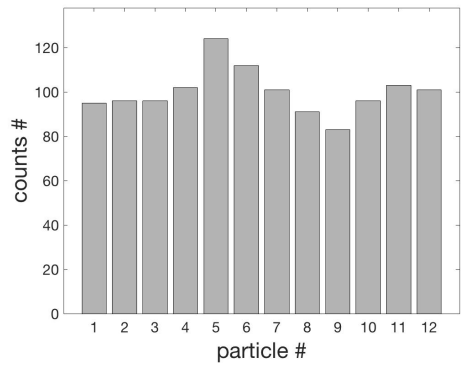
B

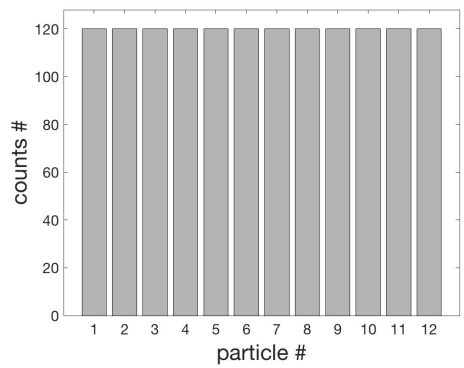


1. trial



2. trial





Conclusion: A has variance > 0, whereas B shows no variance.

Particle filter for localization = Monte Carlo (MC) Localization

- each particle is a pose hypothesis
- proposal is the **motion model** (**state transition model**)

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

- measurement update (correction) via **measurement model**

$$w_t^{[m]} \sim p(z_t|x_t, m)$$

including a map m

remark: sometimes the map is also included into the motion model:
$p(x_t|u_t, x_{t-1}, m)$

Particle filter for localization - Algorithm: version 3

Algorithm MCL$(\mathcal{X}_{t-1}, u_t, z_t, m)$**:**

$\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$

for $m = 1$ to M do

  $x_t^{[m]} = \textbf{sample\_motion\_model}(u_t, x_{t-1}^{[m]})$

  $w_t^{[m]} = \textbf{measurement\_model}(z_t, x_t^{[m]}, m)$

  $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$

endfor

for $m = 1$ to M do

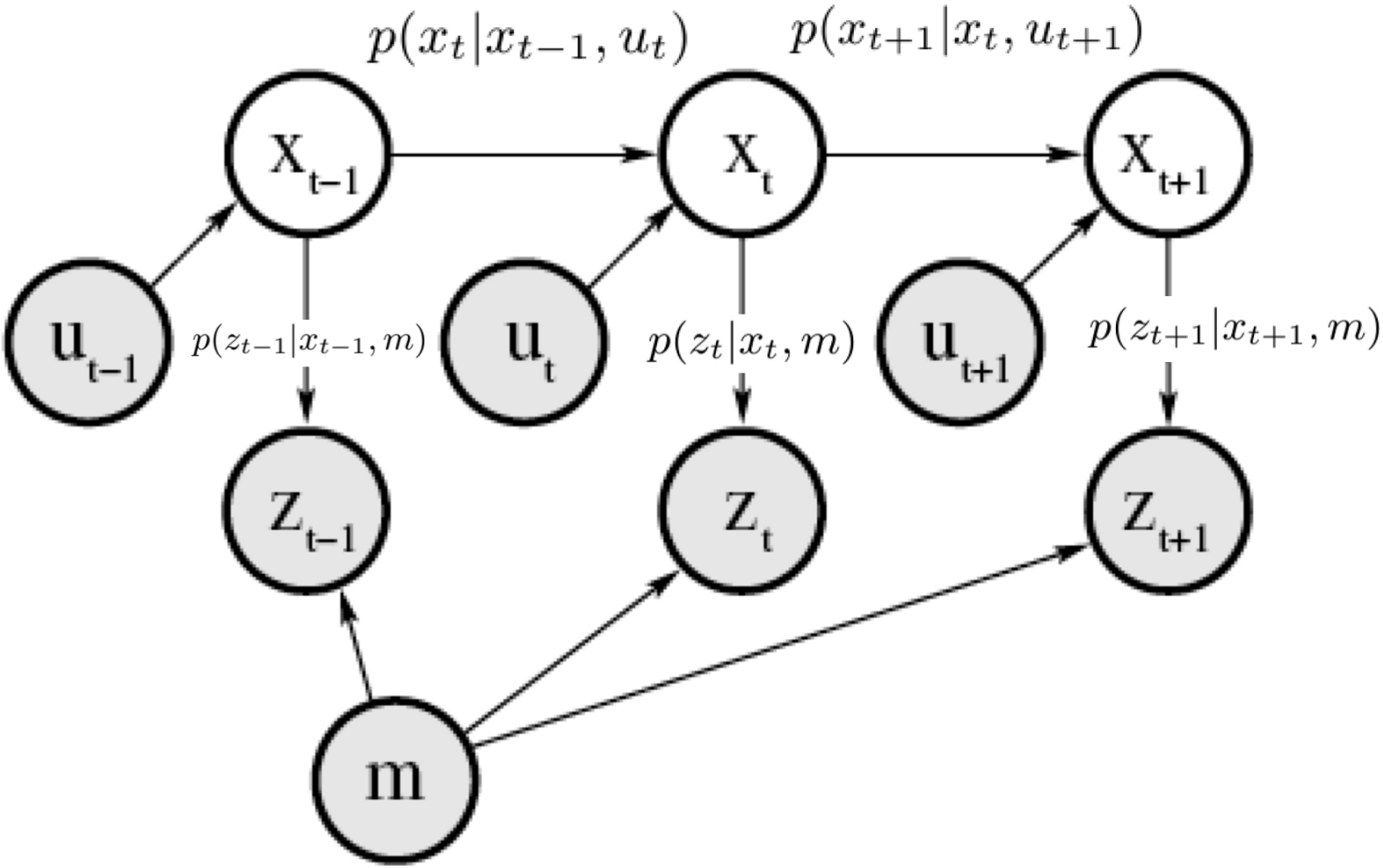
  draw i with probability $\propto w_t^{[i]}$

  add $x_t^{[i]}$ to $\mathcal{X}_t$

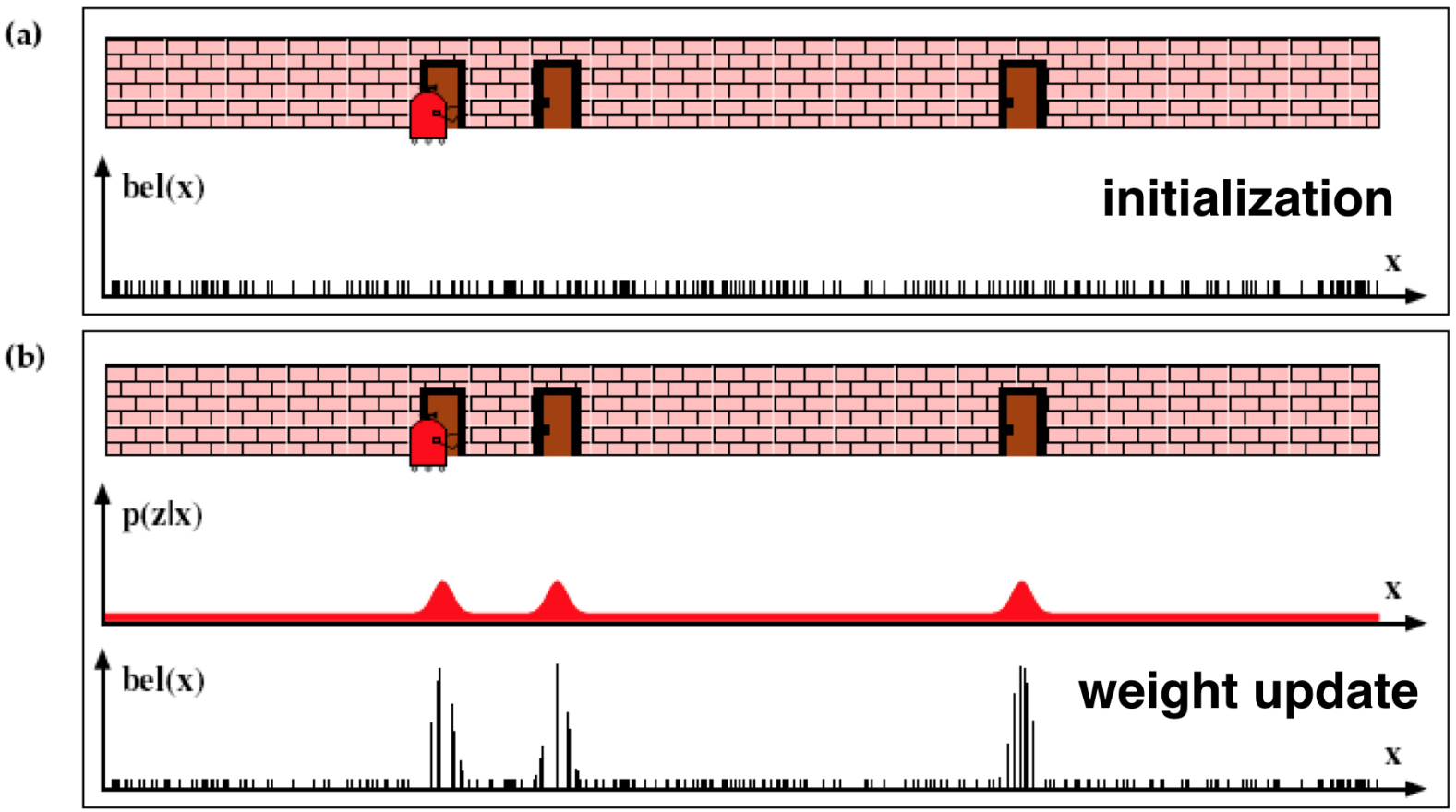
endfor

return $\mathcal{X}_t$

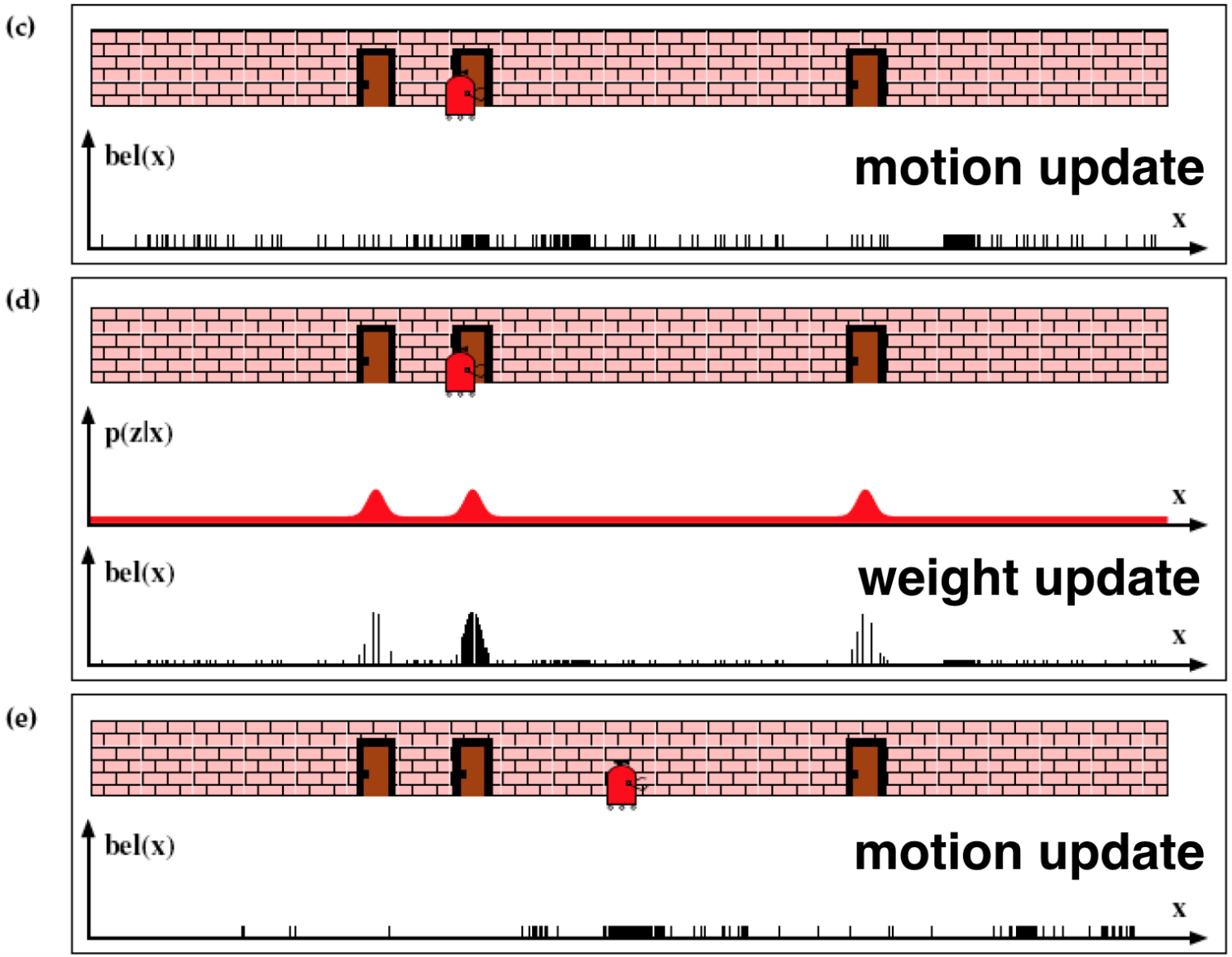
Particle filter for localization - Graphical probabilistic model



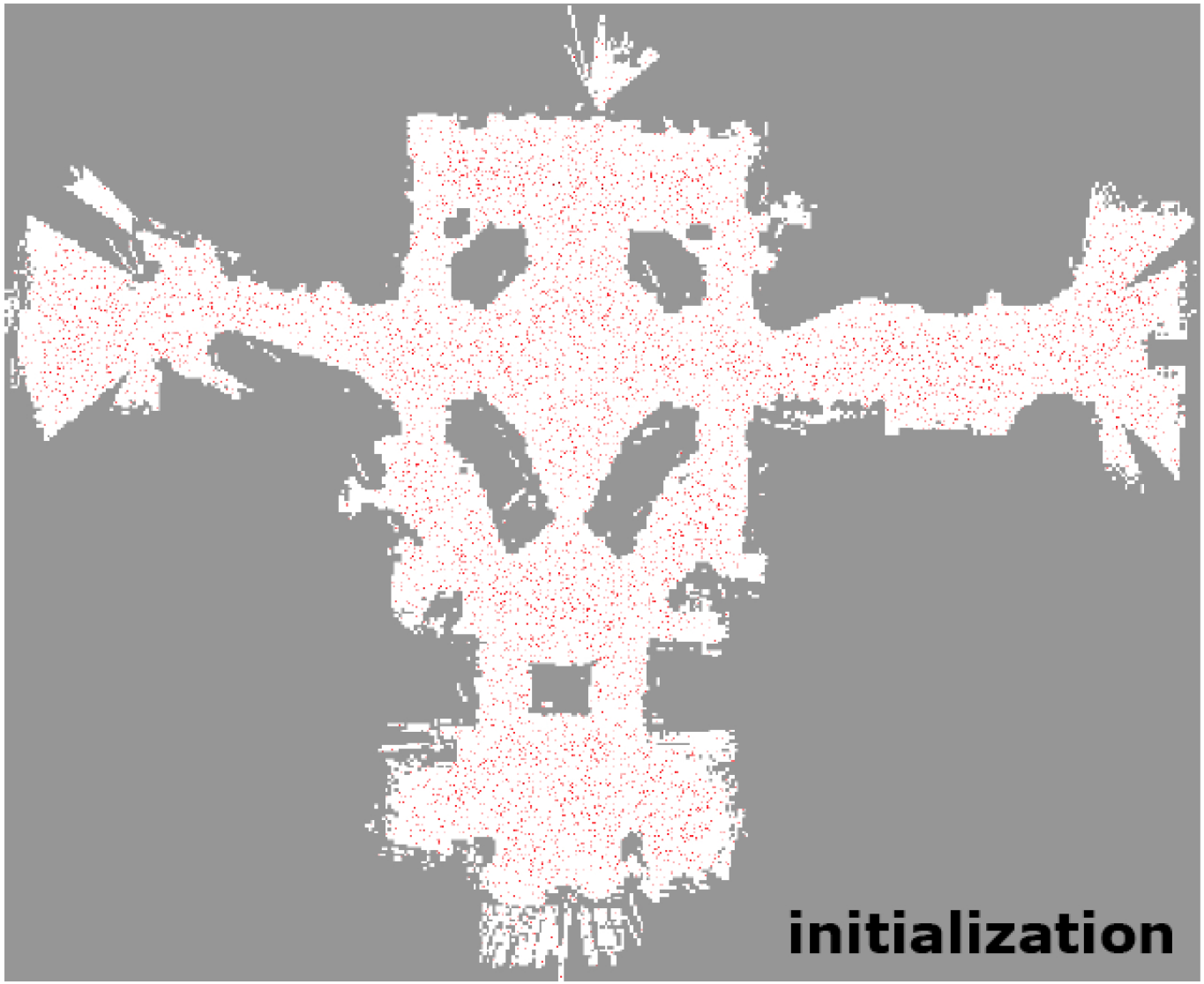
Particle filter for localization



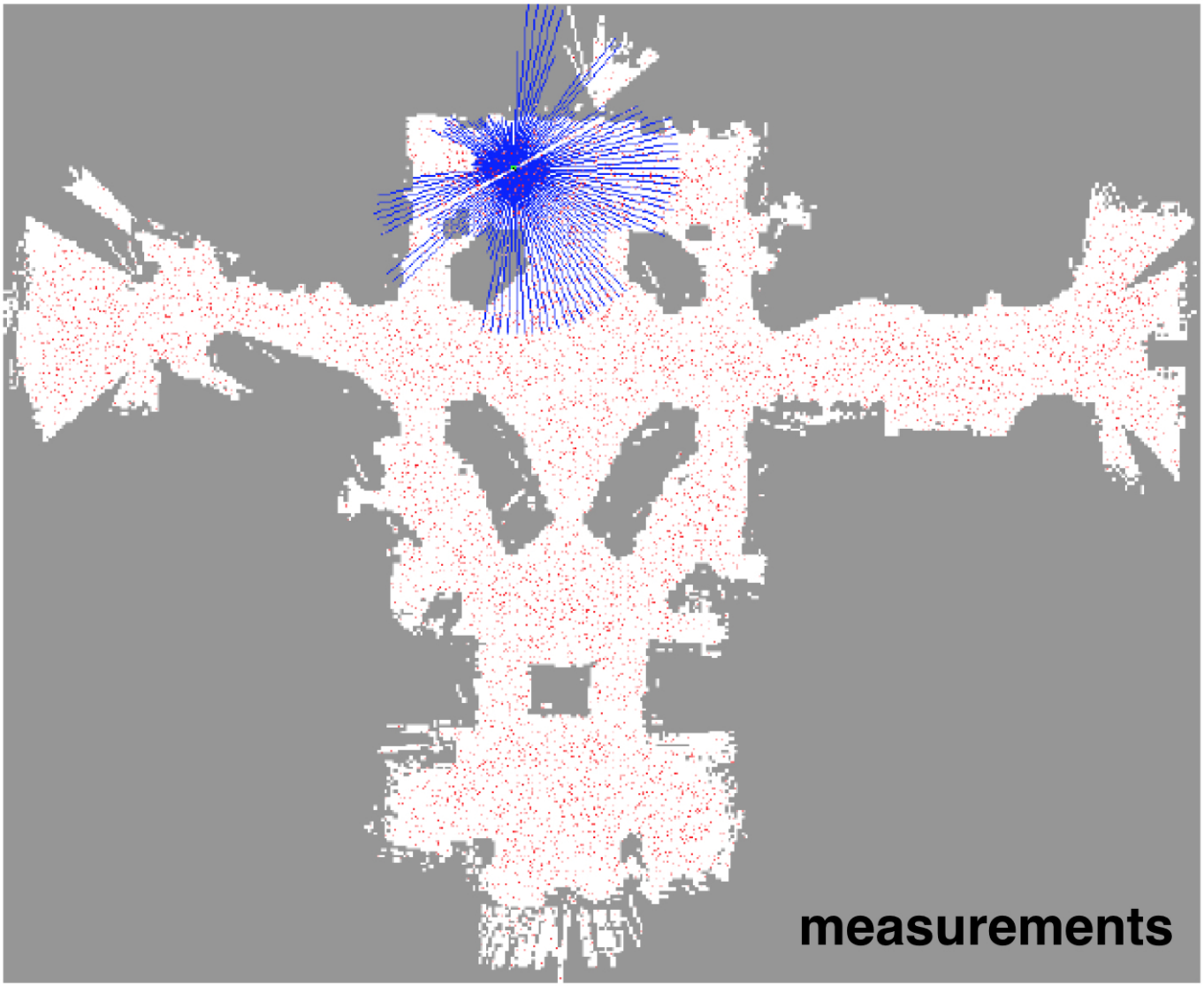
Particle filter for localization



Particle filter for localization - Example

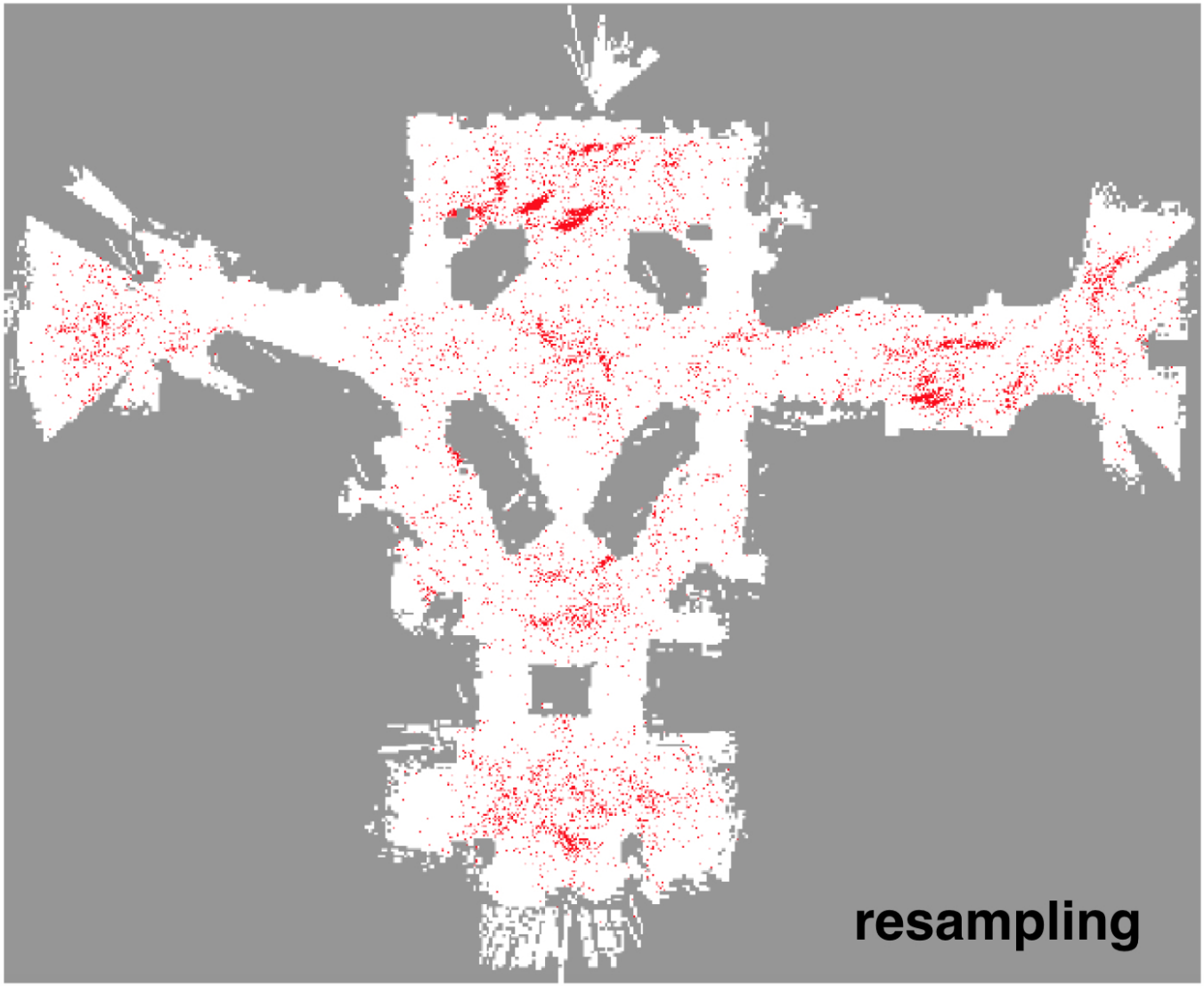


Particle filter for localization - Example

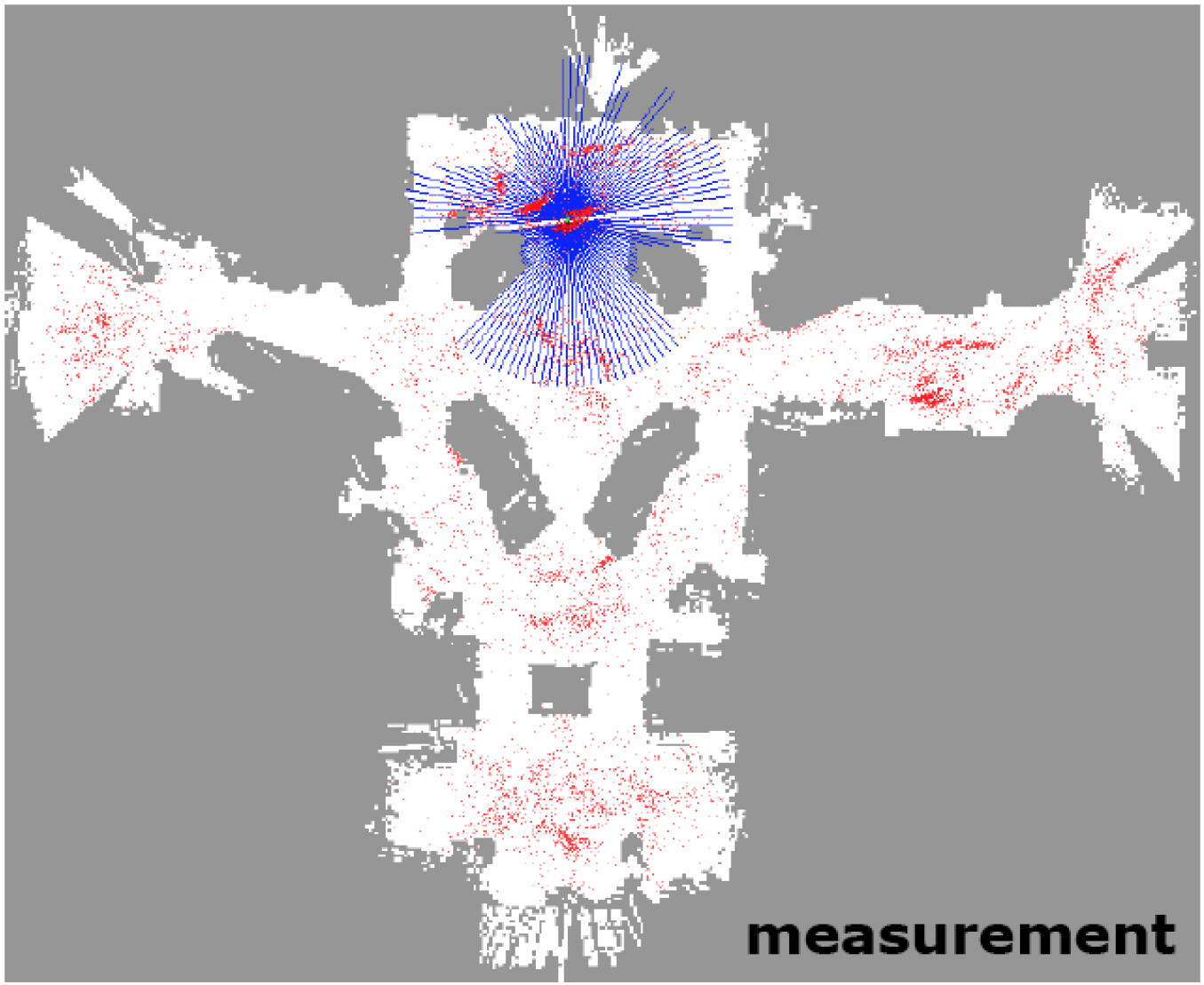


Particle filter for localization - Example

Particle filter for localization - Example

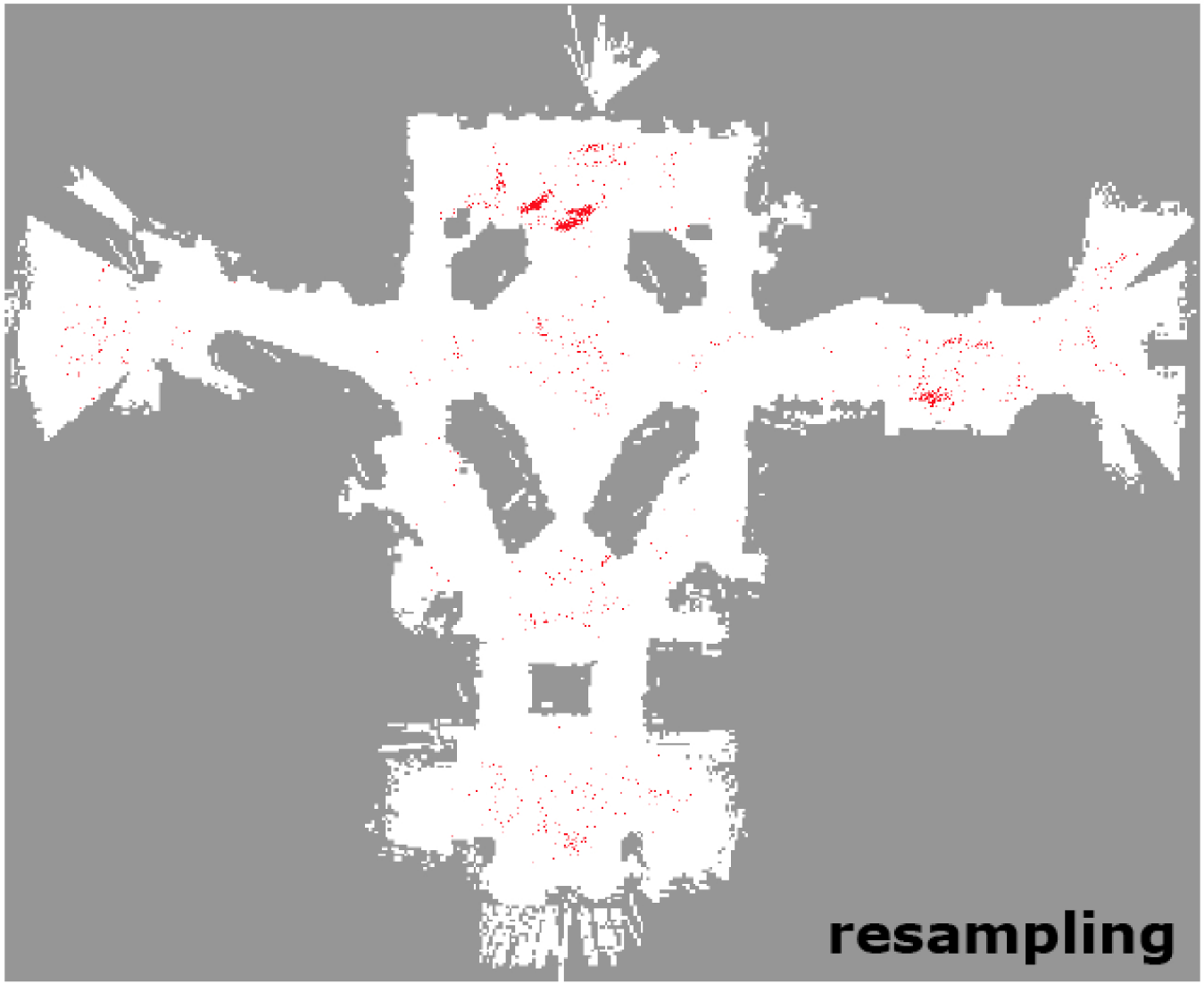


Particle filter for localization - Example

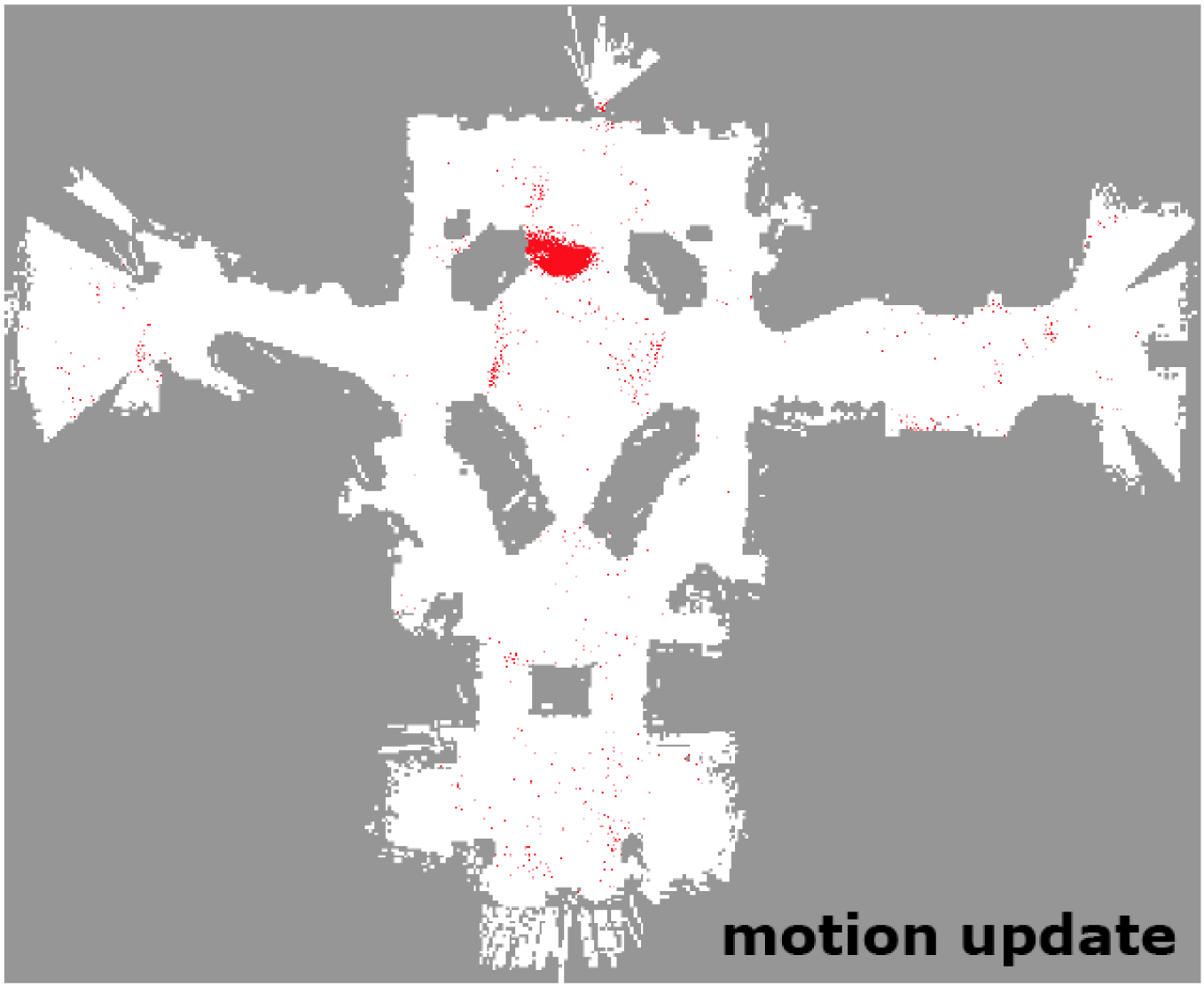


Particle filter for localization - Example

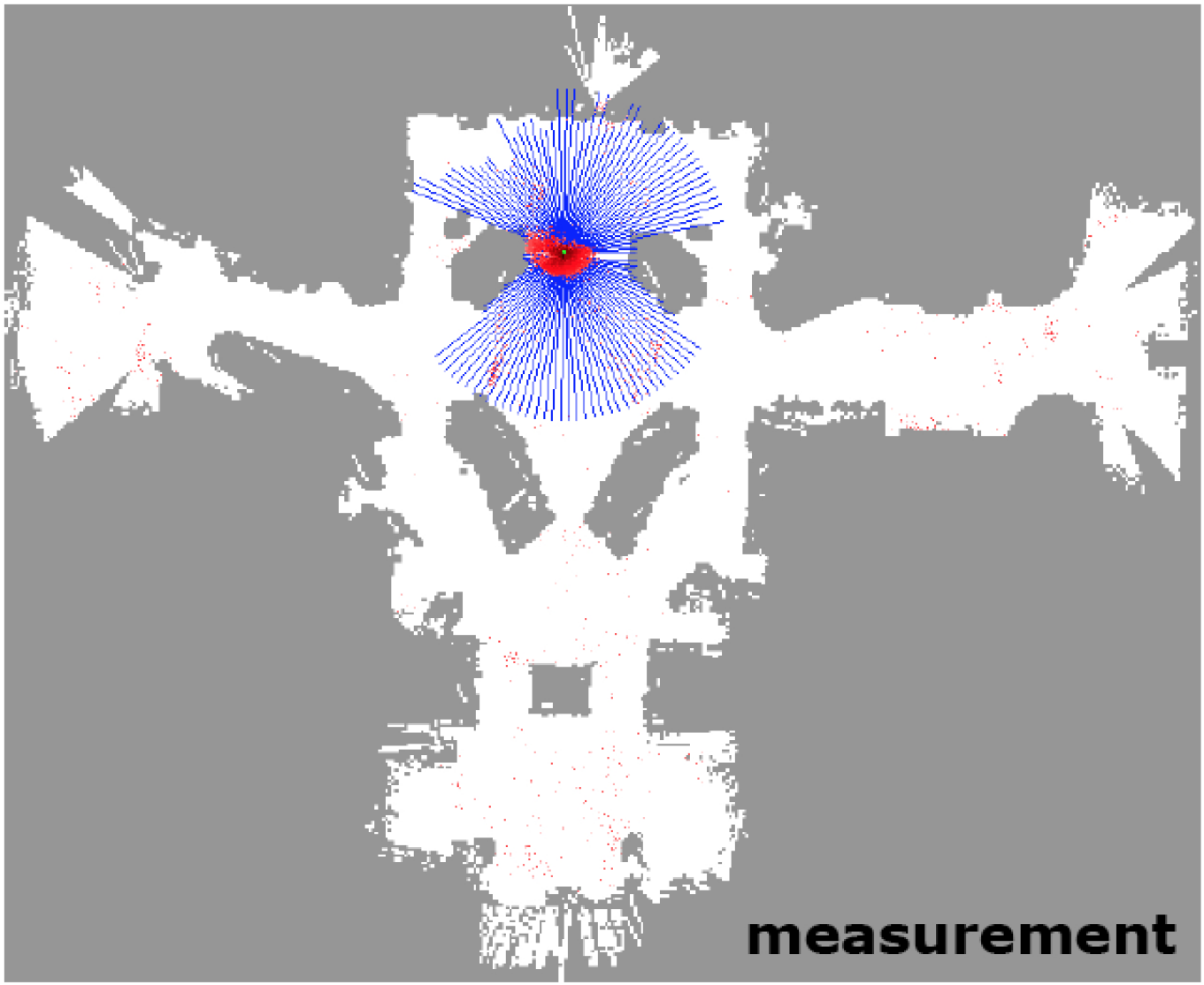
Particle filter for localization - Example



Particle filter for localization - Example

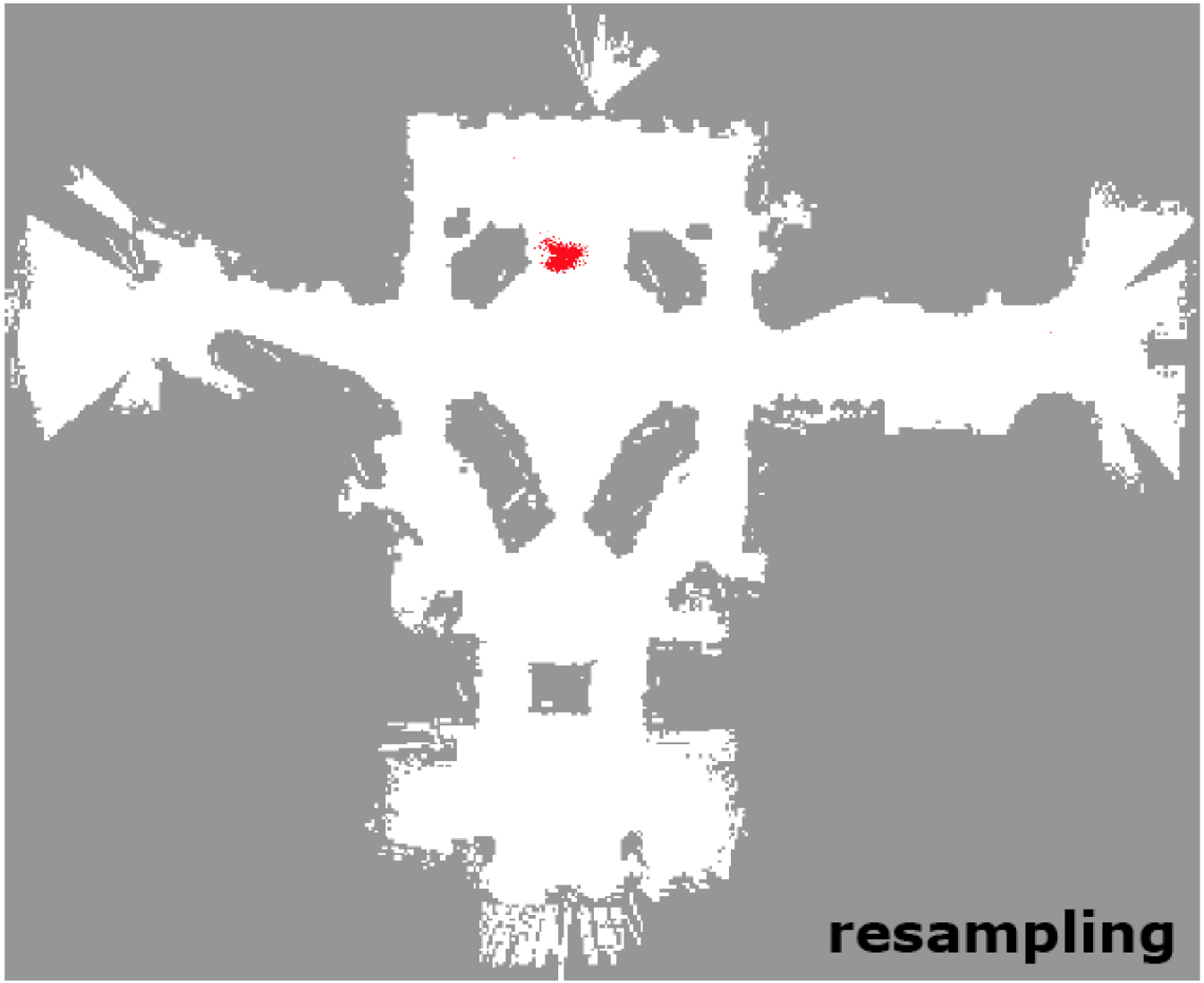


Particle filter for localization - Example

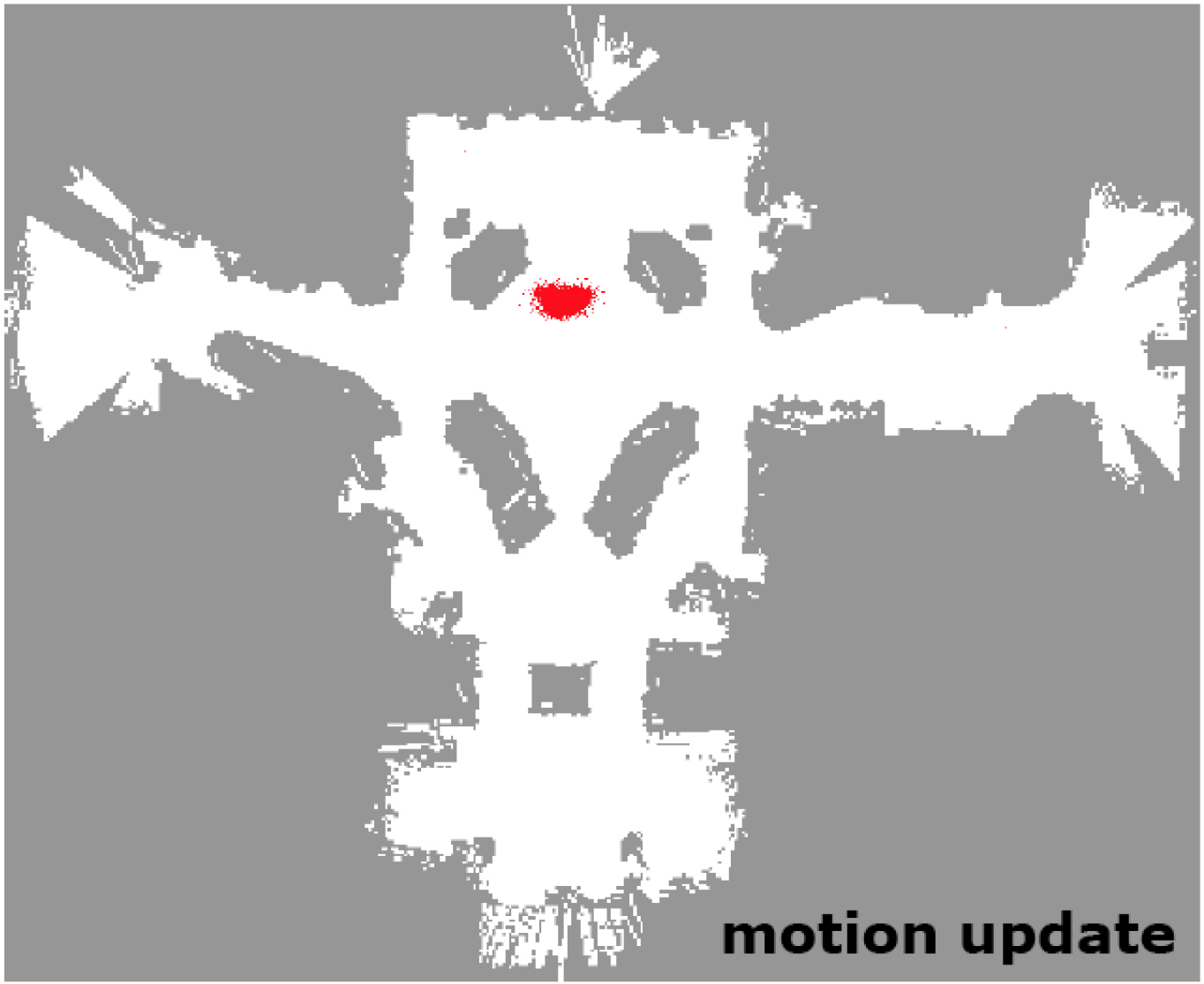


Particle filter for localization - Example

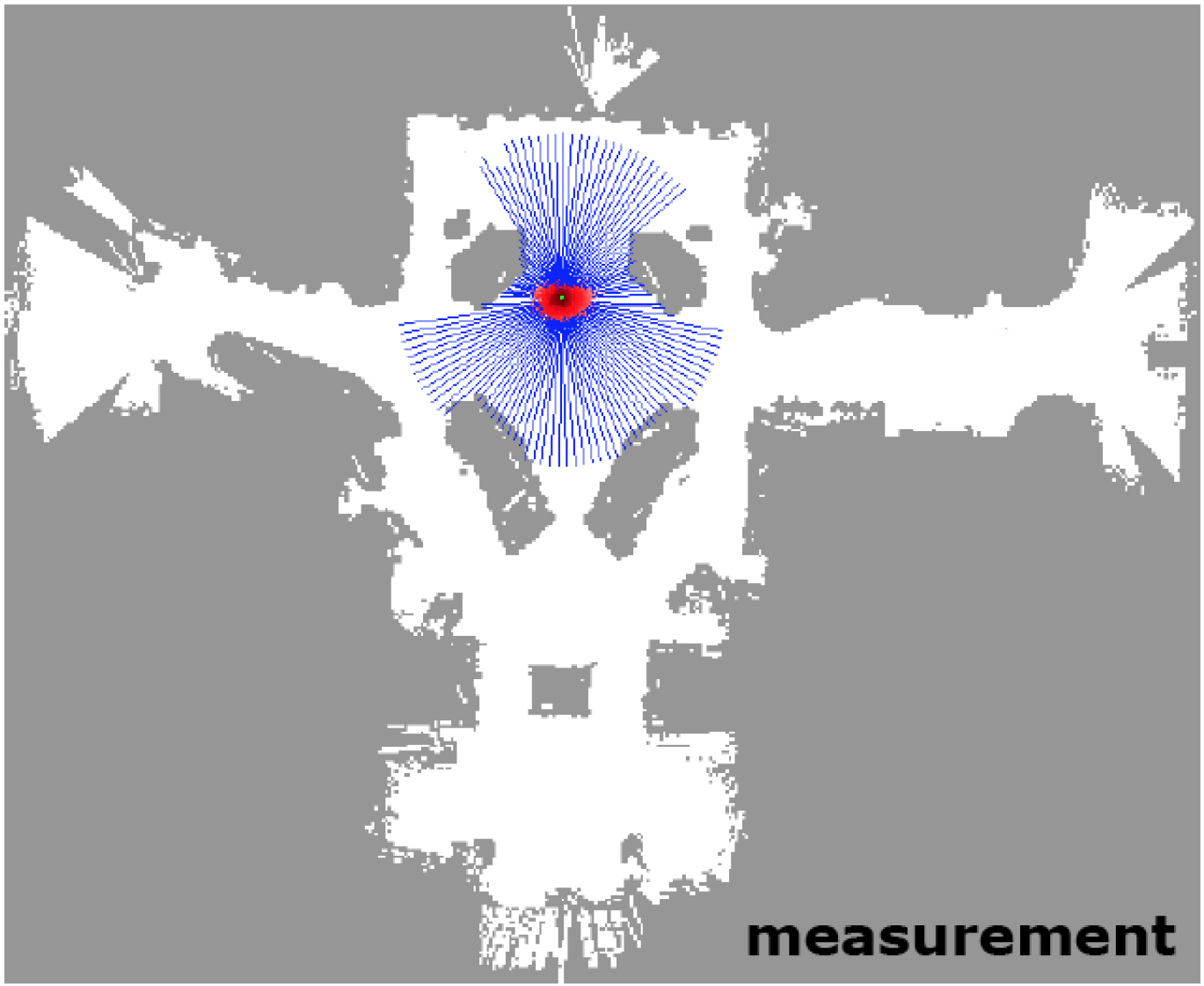
Particle filter for localization - Example



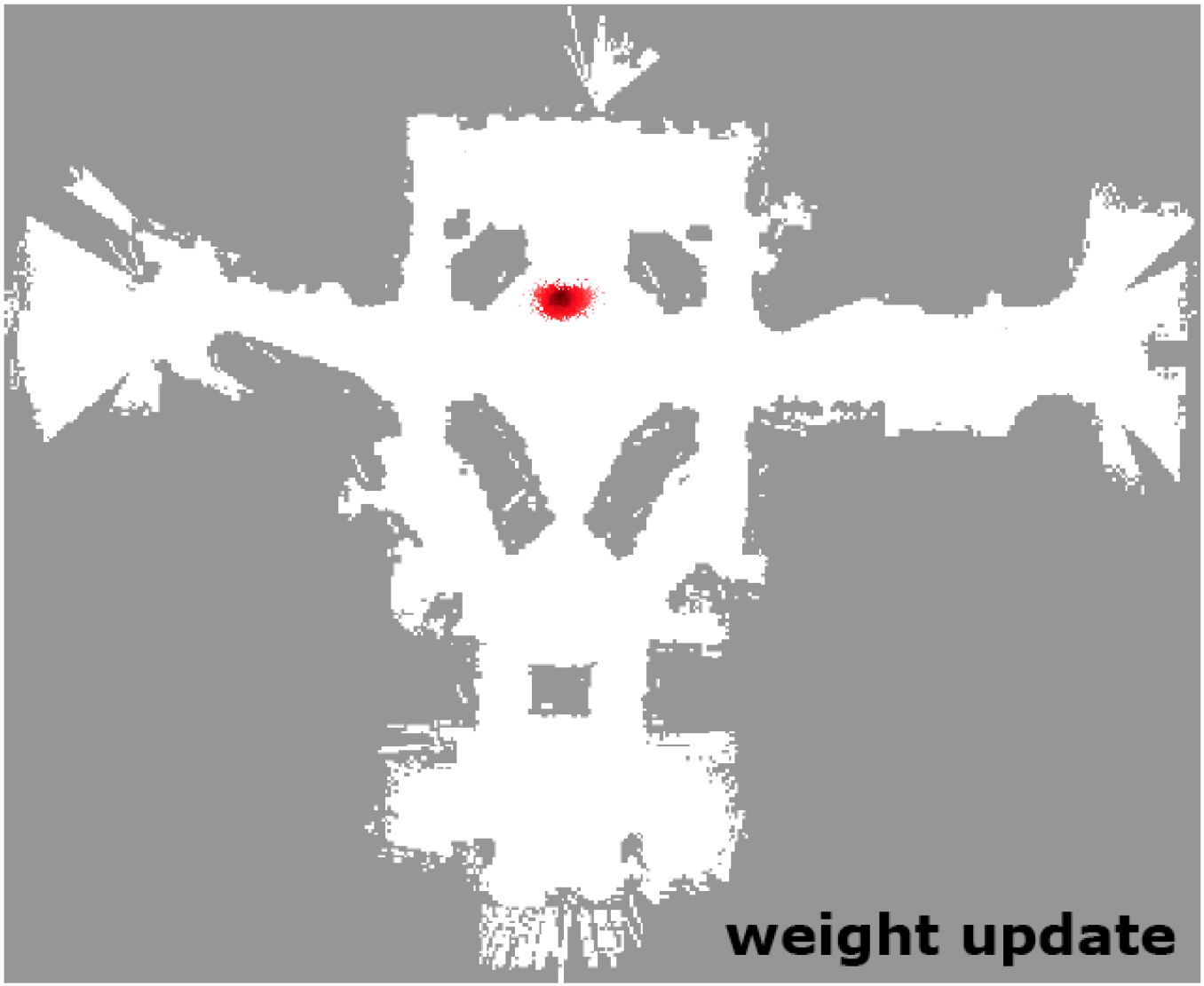
Particle filter for localization - Example



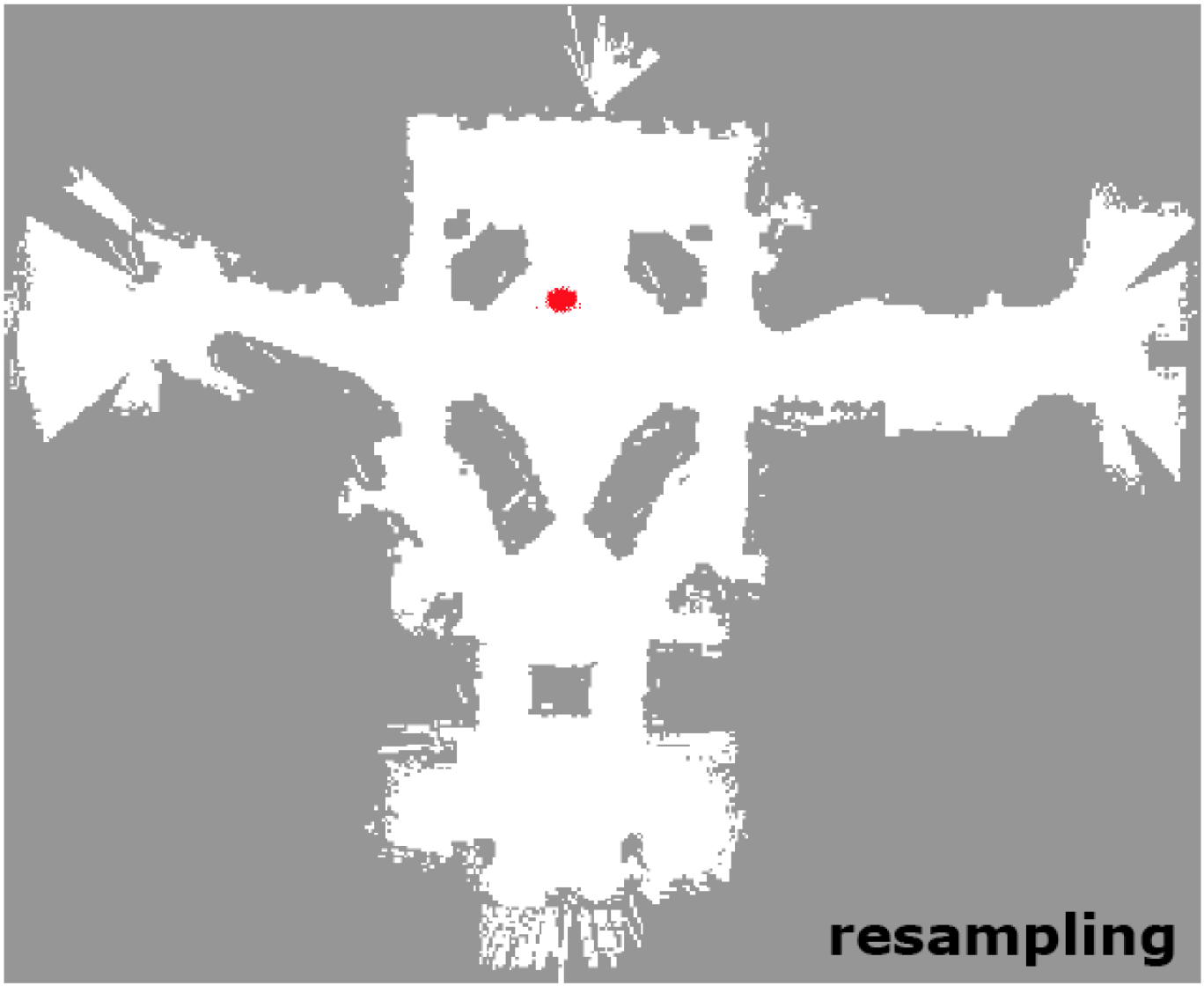
Particle filter for localization - Example



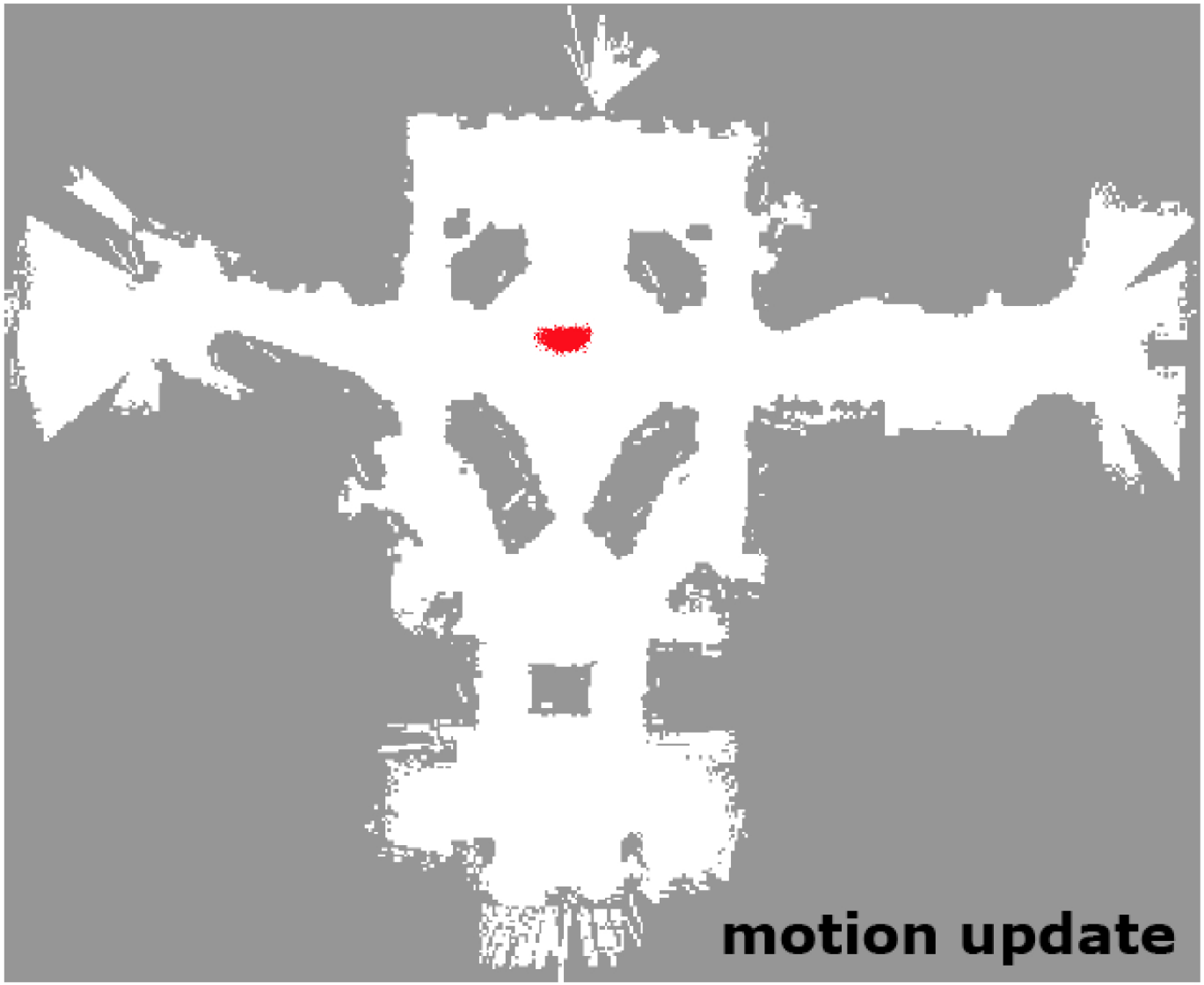
Particle filter for localization - Example



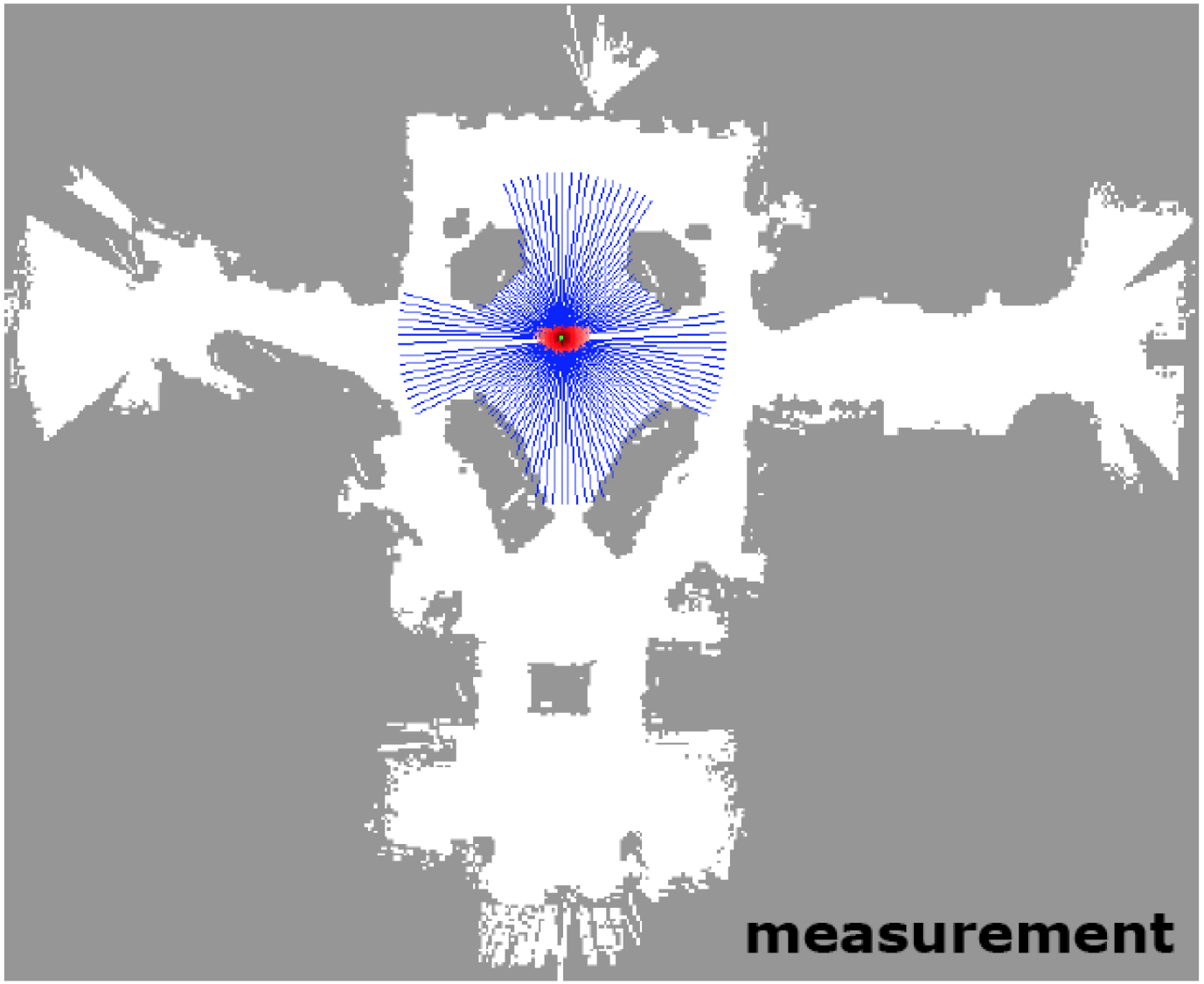
Particle filter for localization - Example



Particle filter for localization - Example



Particle filter for localization - Example



MC localization - summary

- Particles are propagated according to the motion model
- They are weighted according to the likelihood of the measurement model
- MCL is the standard for mobile robot localization today

Particle filter - summary

- PF are non-parametric, recursive Bayes filter
- Posterior is represented by a set of weighted samples
- PF are the most versatile of all Bayes filter algorithms
- PF are easy to implement and very popular in robotics

Acknowledgements - additional resources

- Lecture Notes: *Robot Mapping* by Cyrill Stachniss