B31YS Robotics Systems Science

Week 4

Particle Filter and Monte Carlo Localization

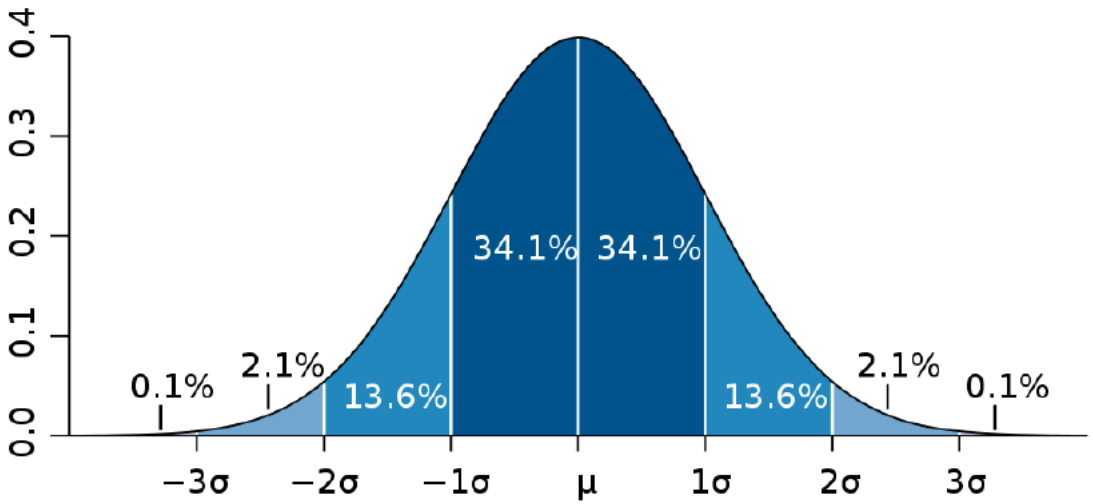
Dr. Sen Wang

s.wang@hw.ac.uk

Gaussian Filters

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

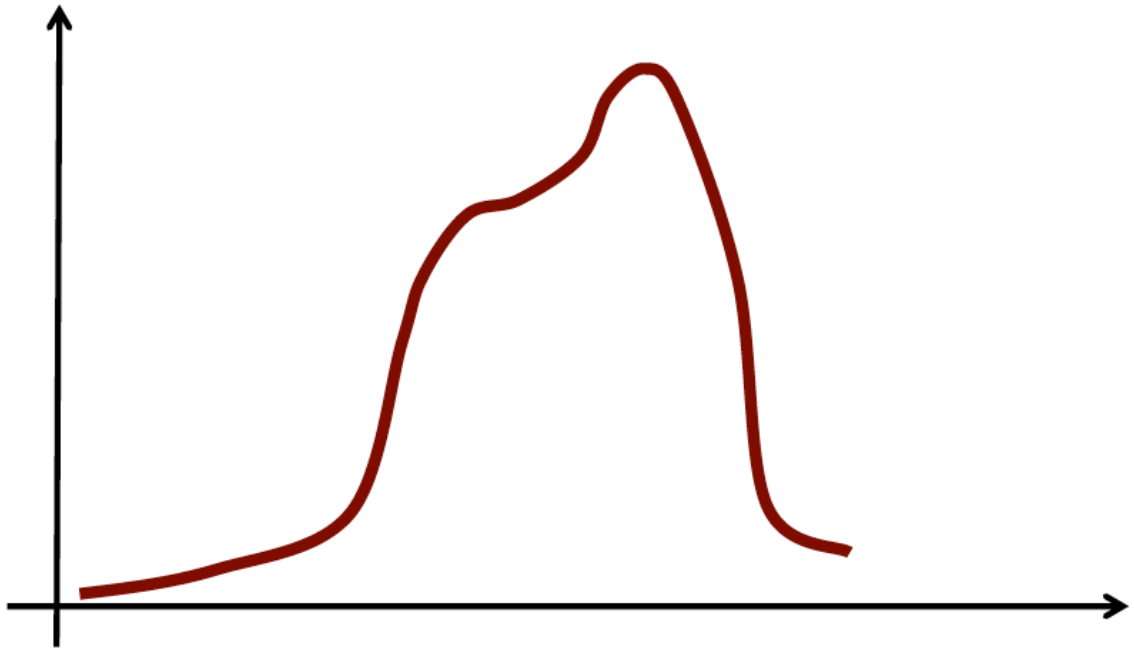
$$p(x) = \det(2\pi\Sigma)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$



Courtesy: K. Arras

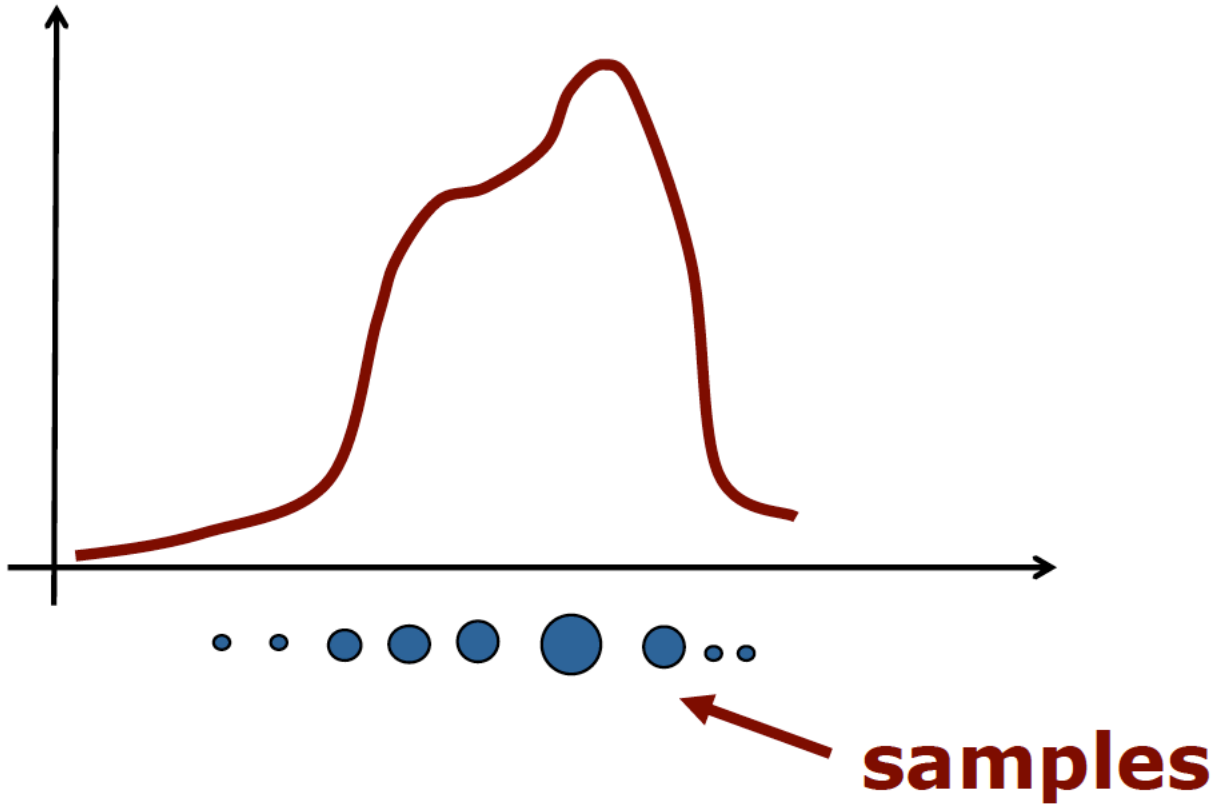
Motivation

- Goal: approach for dealing with arbitrary distributions



Key Idea: Samples

- Use multiple samples to represent arbitrary distributions



Particle Set

- Set of weighted samples

$$\mathcal{X} = \left\{ \left\langle x^{[j]}, w^{[j]} \right\rangle \right\}_{j=1,\ldots,J}$$

state hypothesis

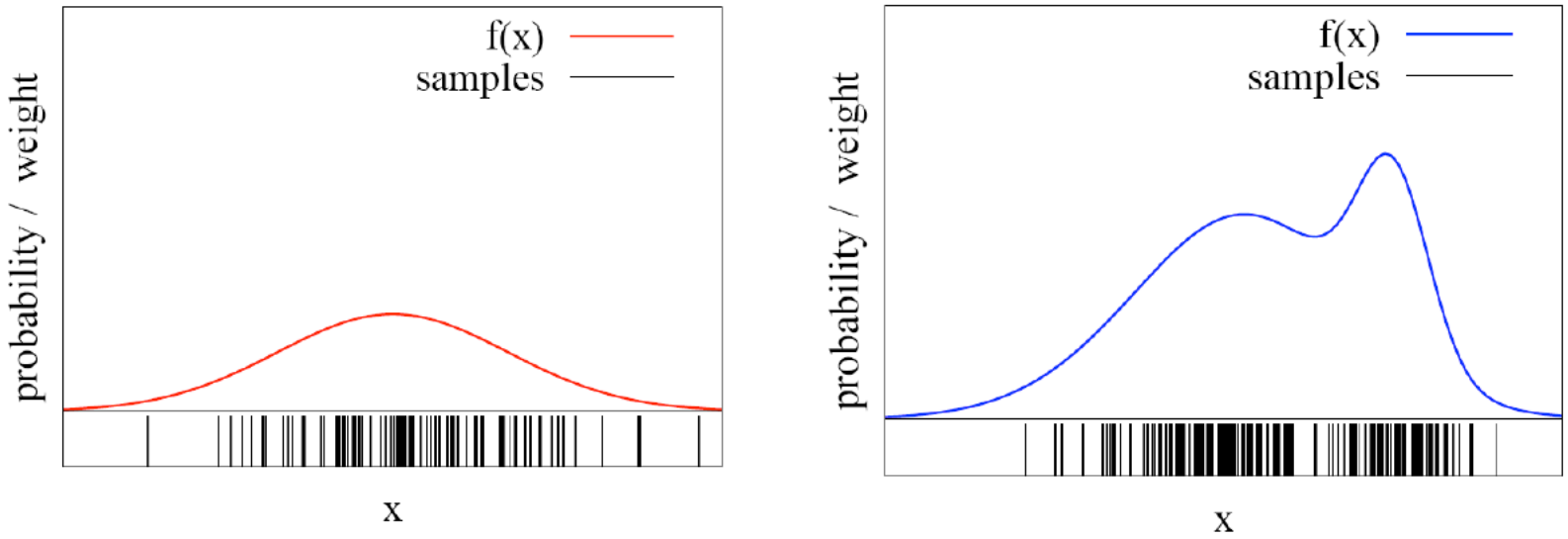
importance weight

- The samples represent the posterior

$$p(x) = \sum_{j=1}^{J} w^{[j]} \, \delta_{x^{[j]}}(x)$$

Particles for Approximation

- Particles for function approximation

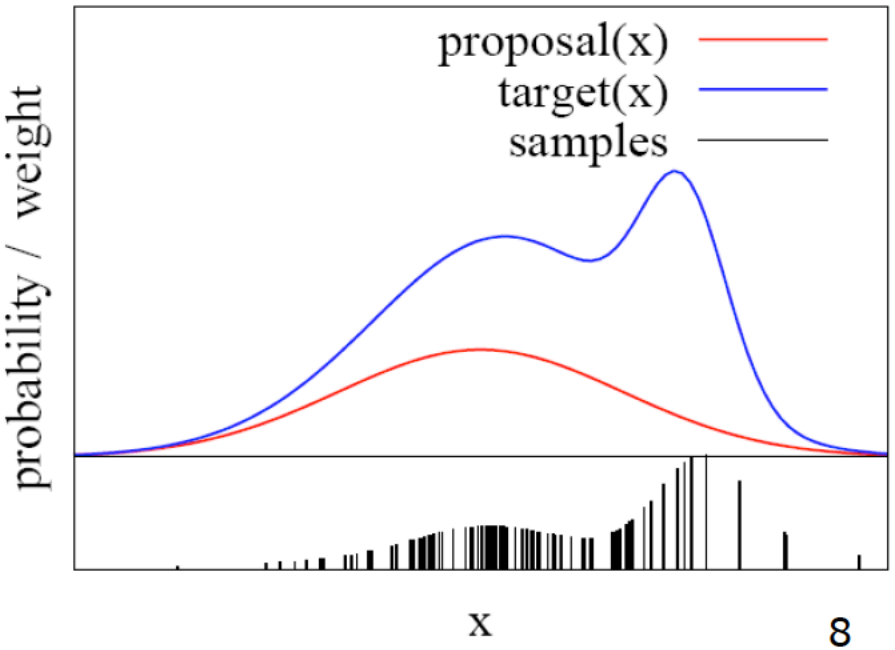


- The more particles fall into a region, the higher the probability of the region

• How to obtain such samples?

Importance Sampling Principle

- We can use a different distribution g 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:
  $f(x)>0 \rightarrow g(x)>0$



Particle Filters

- Idea: Represent belief/distribution by random samples
- Particle filters are a way to efficiently represent non-Gaussian distribution and nonlinear processes
- Recursive Bayes filter
- Non-parametric approach
  - Prediction: draw samples from the proposal
  - Correction: weighting by the ratio of target and proposal

The more samples we use, the better is the estimate!

Particle Filter Algorithm

1. Sample the particles using the proposal distribution

$$x_t^{[j]} \sim \pi(x_t \mid \ldots)$$

2. Compute the importance weights

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

- Resampling: Draw sample i with probability $w_t^{[i]}$ and repeat J times

Work well in low-dimensional spaces

Particle Filter Algorithm

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

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

2: *for* $j = 1$ *to* J *do*

3: *sample* $x_t^{[j]} \sim \pi(x_t)$

4: $w_t^{[j]} = \frac{p(x_t^{[j]})}{\pi(x_t^{[j]})}$

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

6: *endfor*

7: *for* $j = 1$ *to* J *do*

8: *draw* $i \in 1, \ldots, J$ *with probability* $\propto w_t^{[i]}$

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

10: *endfor*

11: *return* $\mathcal{X}_t$

Monte Carlo Localization (MCL)

- Each particle is a pose hypothesis
- Proposal is the motion model

prediction step $x_t^{[j]} \sim p(x_t \mid x_{t-1}, u_t)$

- Correction via the observation model

correction step $w_t^{[j]} = \frac{target}{proposal} \propto p(z_t \mid x_t, m)$

observation model to compute the importance weight

Monte Carlo Localization (MCL)

The set of weighted particles approximates the posterior belief about the robot's pose

- Particles are drawn from the motion model (proposal distribution)
- Particles are weighted according to the observation model (sensor model)
- Particles are resampled according to the particle weights

Particle Filter for Localization

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

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

2: *for* $j = 1$ *to* J *do*

3: *sample* $x_t^{[j]} \sim p(x_t \mid u_t, x_{t-1}^{[j]})$

4: $w_t^{[j]} = p(z_t \mid x_t^{[j]})$

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

6: *endfor*

7: *for* $j = 1$ *to* J *do*

8: *draw* $i \in 1, \ldots, J$ *with probability* $\propto w_t^{[i]}$

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

10: *endfor*

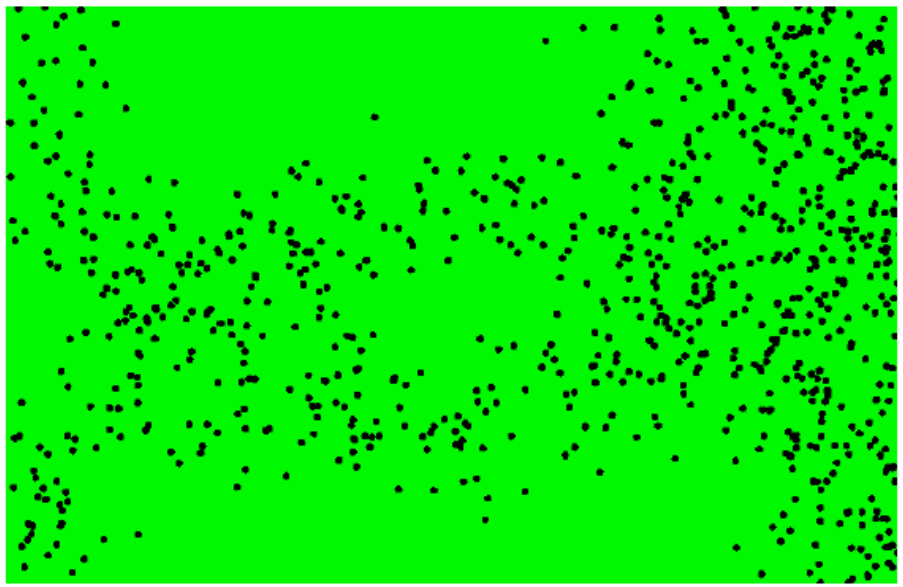
11: *return* $\mathcal{X}_t$

Resampling

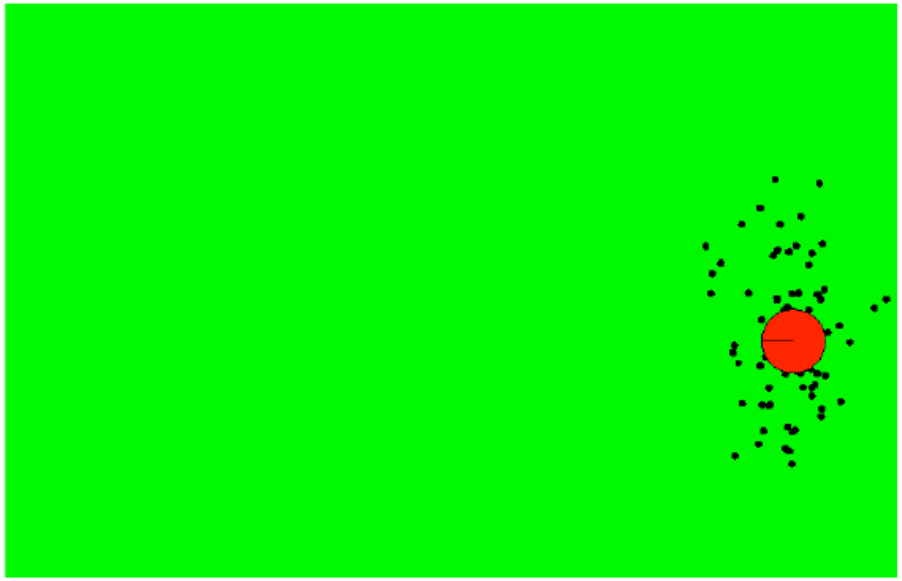
- Draw sample i with probability $w_t^{[i]}$. Repeat J times.
- Informally: "Replace unlikely samples by more likely ones"
- Survival of the fittest
- "Trick" to avoid that many samples cover unlikely states
- Needed as we have a limited number of samples

Resampling ensures that particles stay in the meaningful area of the state space

Resampling

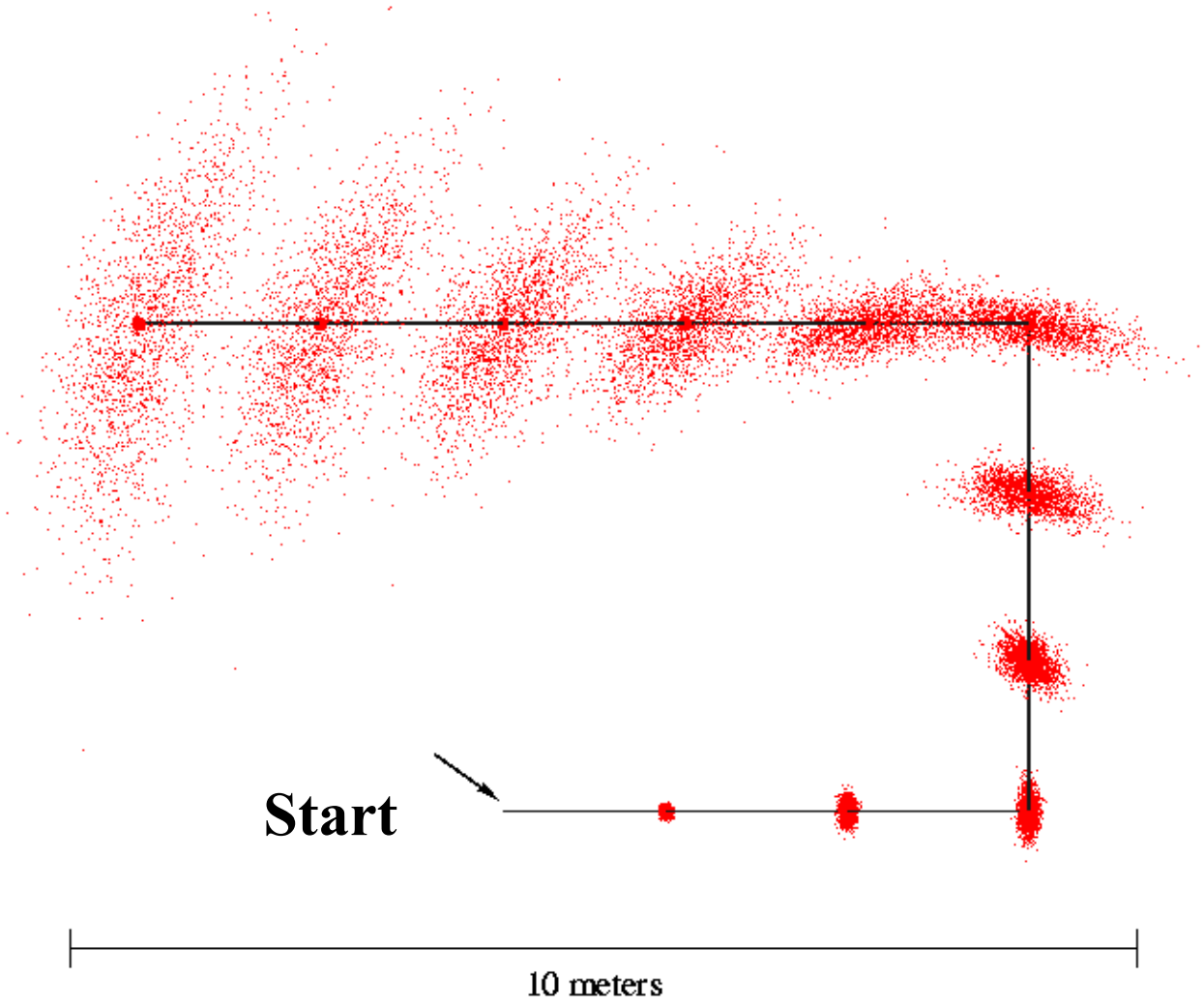


Weighted samples

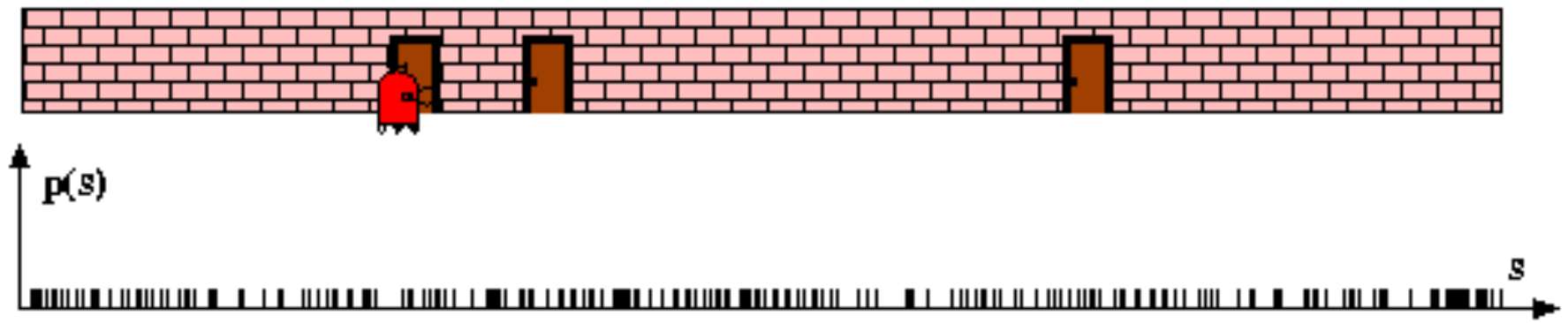


After resampling

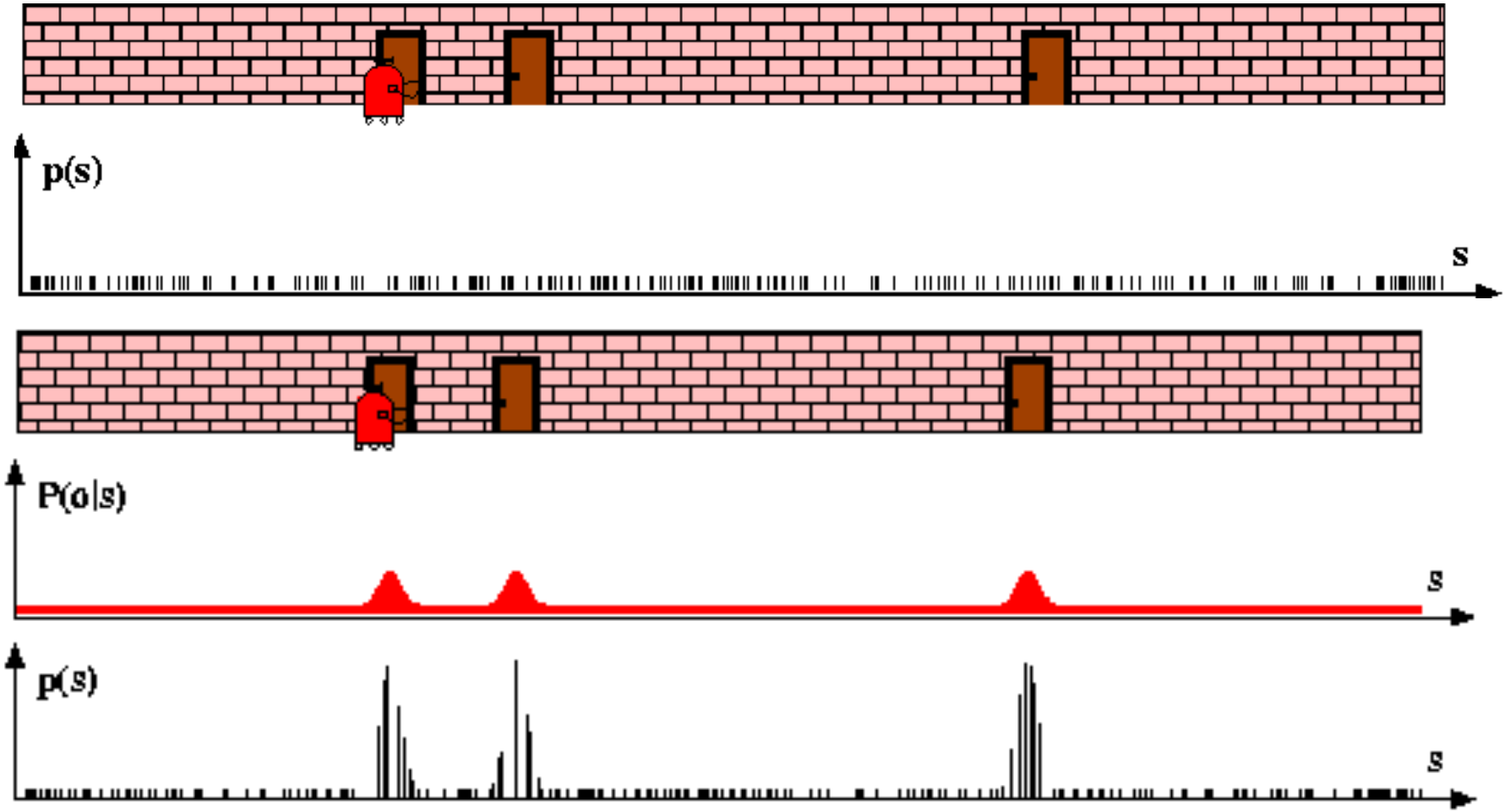
Motion Model Reminder



Particle Filters



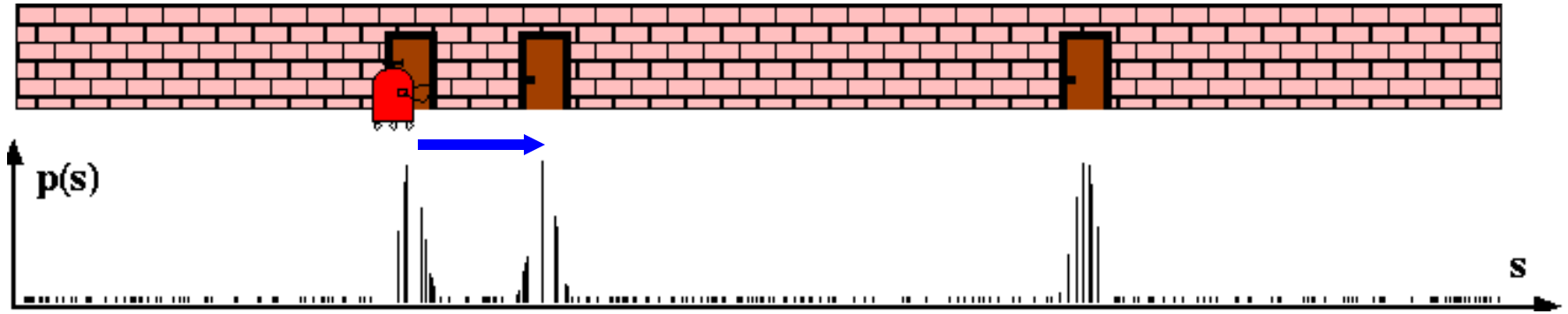
Sensor Information: Importance Sampling

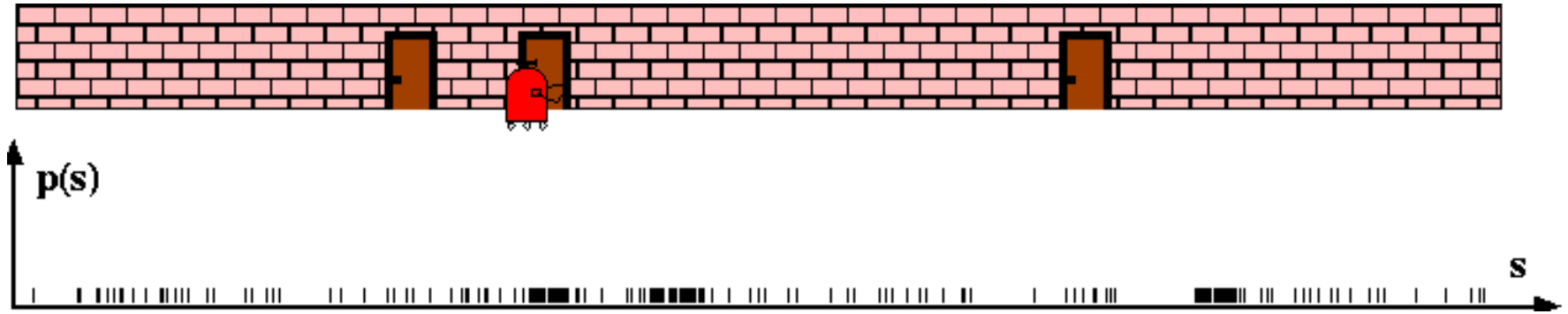


$$
\begin{aligned}
Bel(x) &\leftarrow \alpha\ p(z \mid x)\, Bel^{-}(x) \\
w &\leftarrow \frac{\alpha\ p(z \mid x)\, Bel^{-}(x)}{Bel^{-}(x)} = \alpha\ p(z \mid x)
\end{aligned}
$$

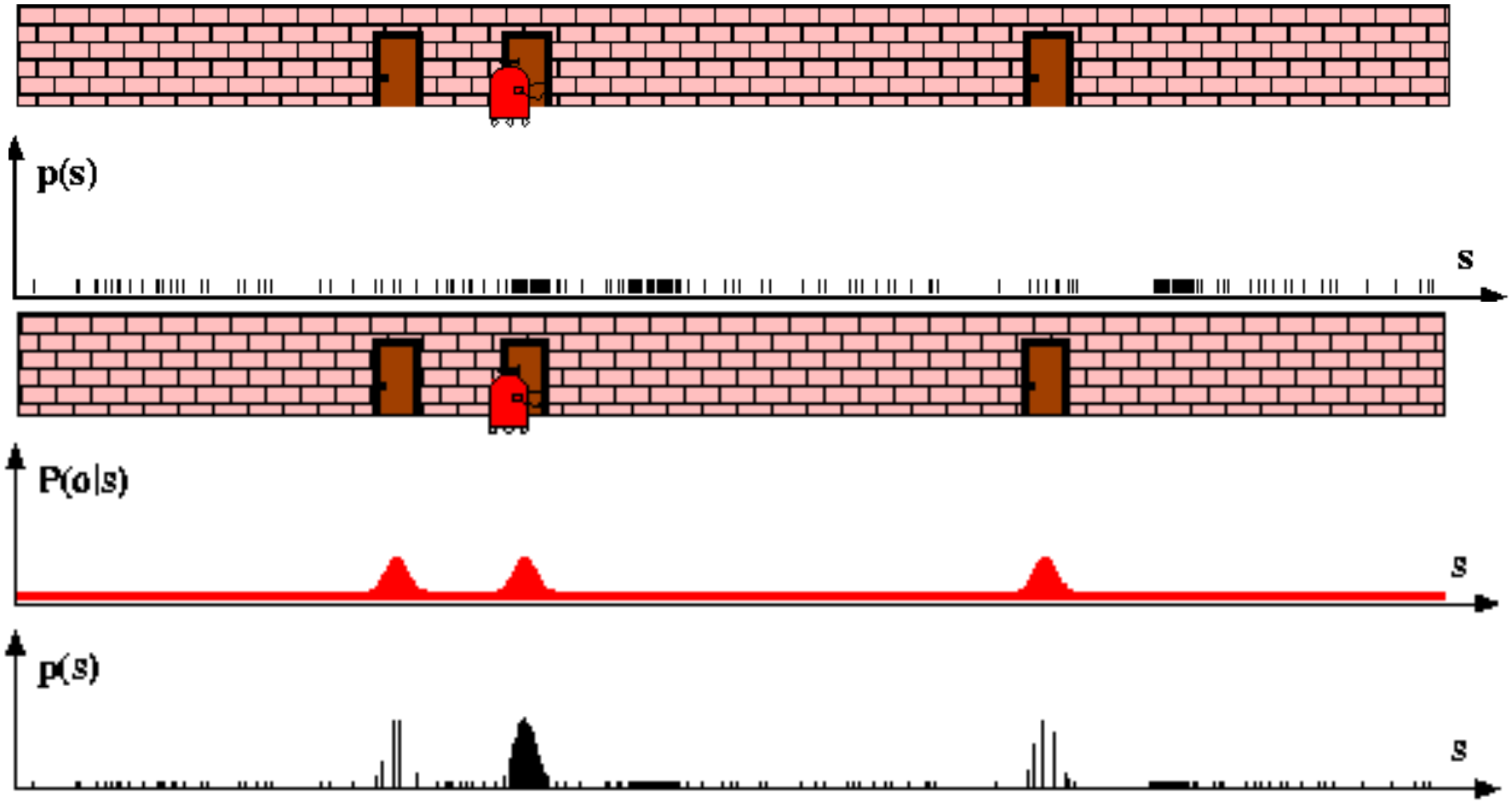
Robot Motion

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



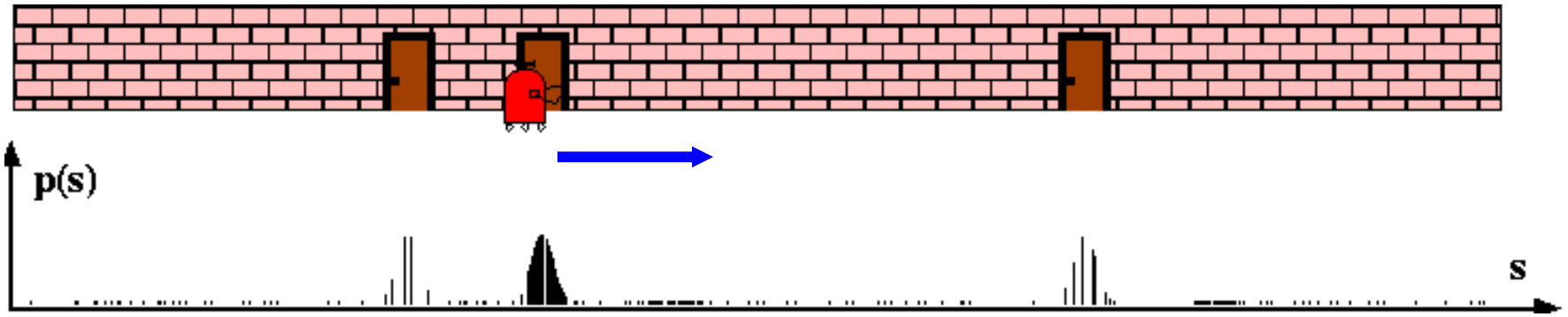


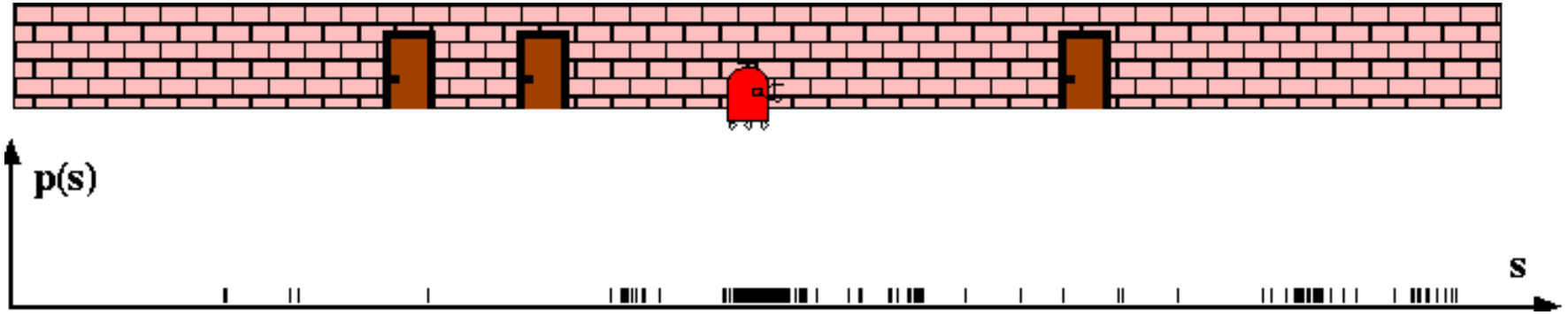
Sensor Information: Importance Sampling



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

Robot Motion





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

Particle Filter Algorithm

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

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

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

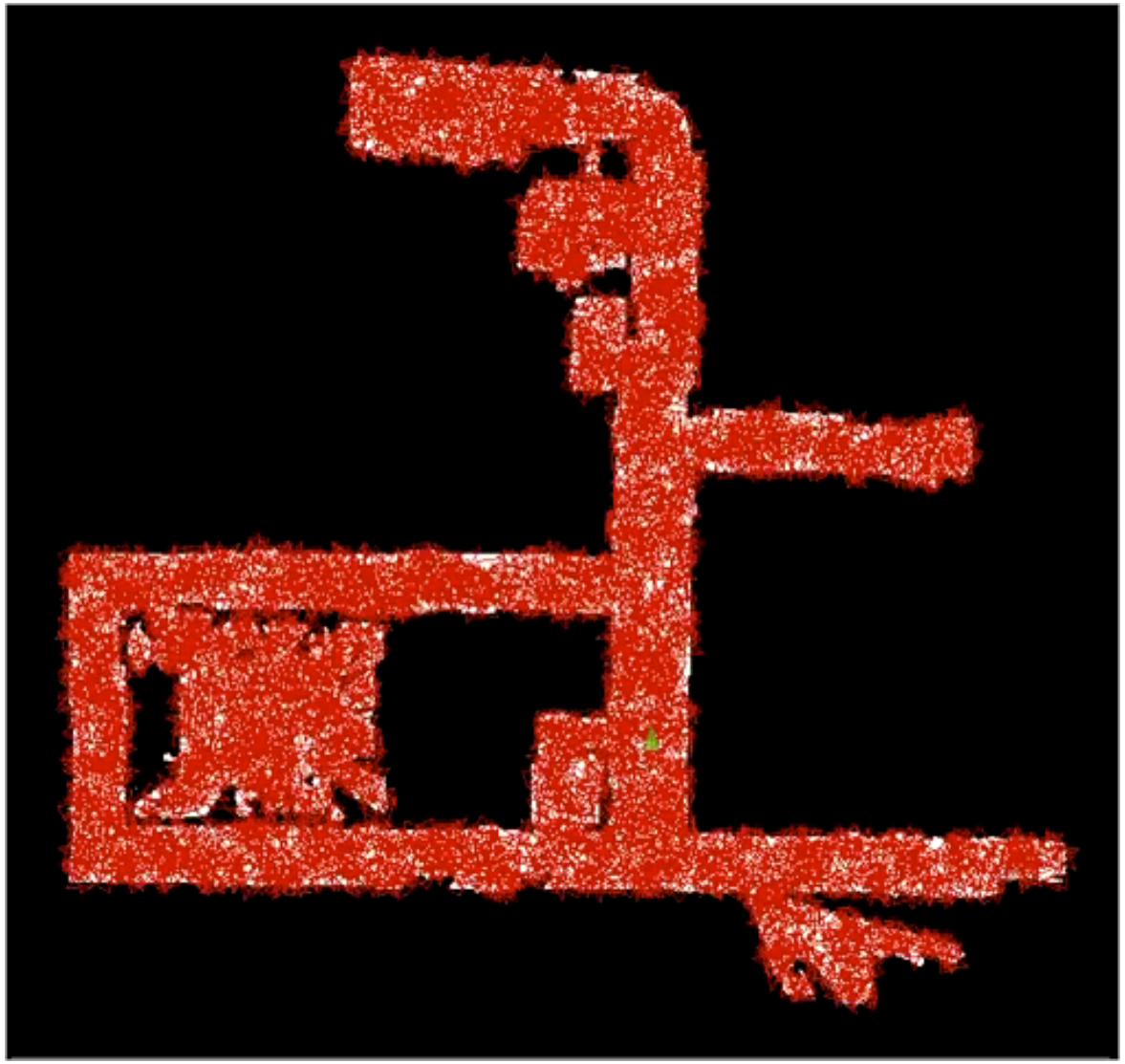
Importance factor for x^i_t:

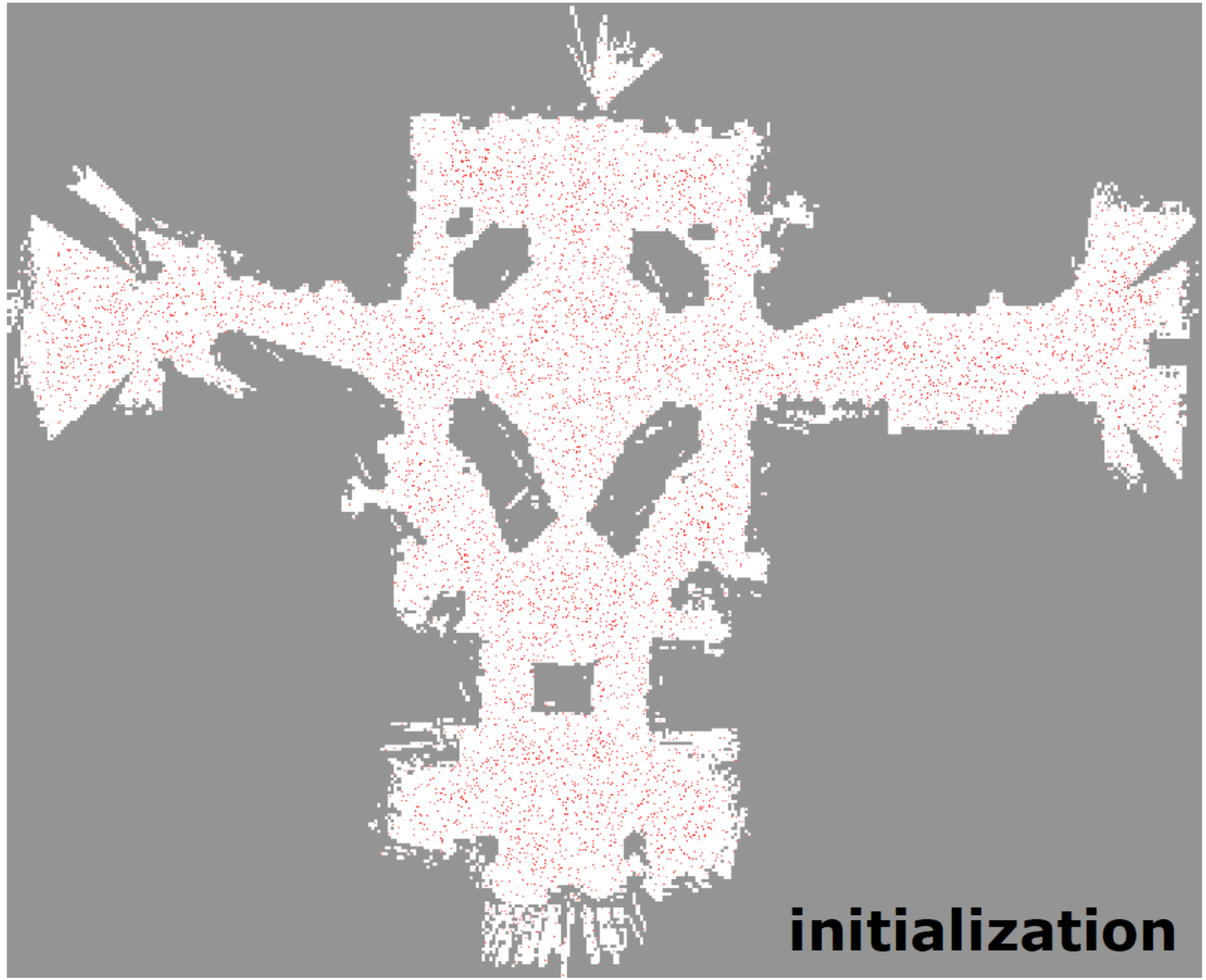
$$w^i_t = \frac{\text{target distribution}}{\text{proposal distribution}}$$

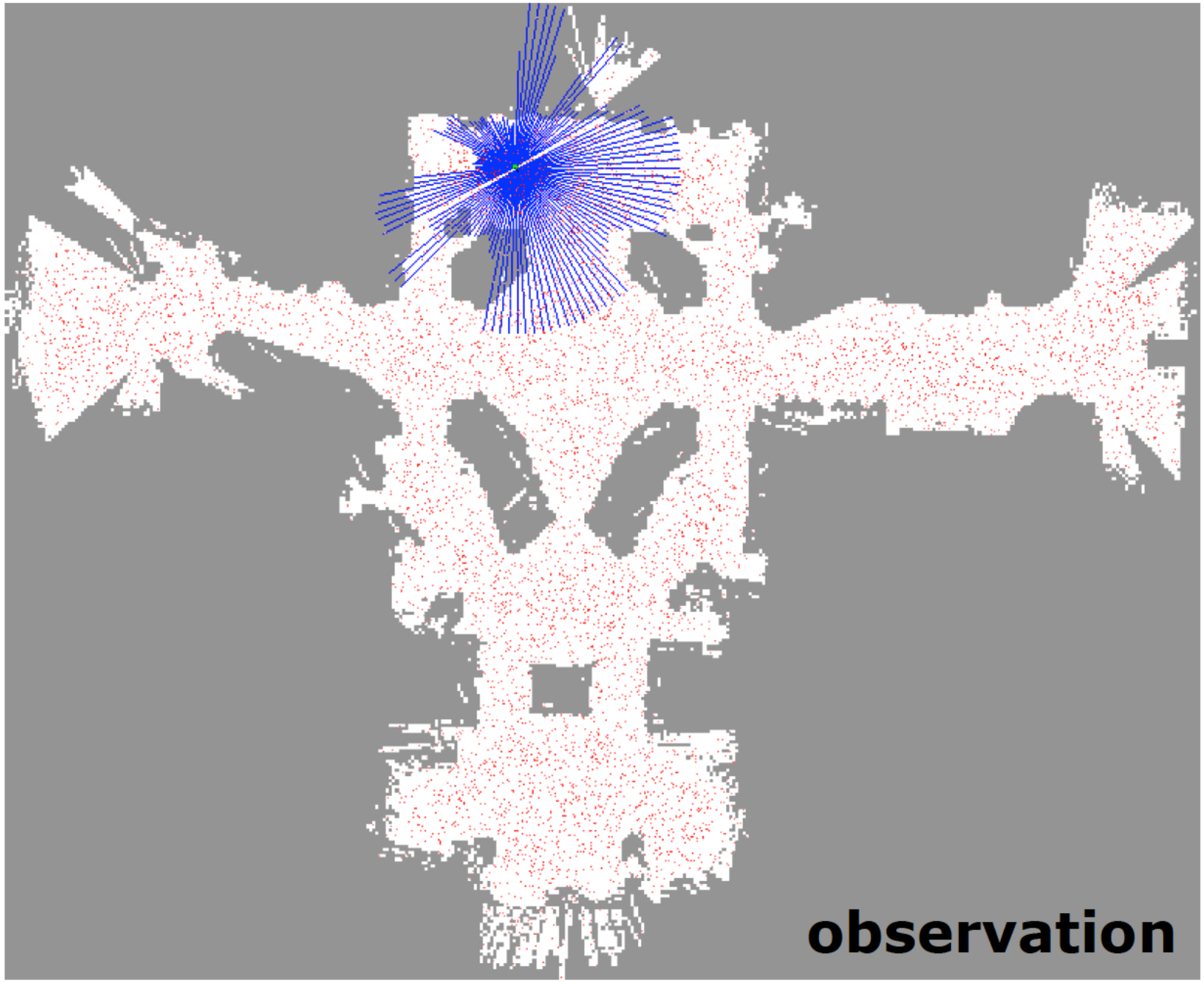
$$= \frac{\eta\; p(z_t \mid x_t)\; p(x_t \mid x_{t-1}, u_{t-1})\; Bel(x_{t-1})}{p(x_t \mid x_{t-1}, u_{t-1})\; Bel(x_{t-1})}$$

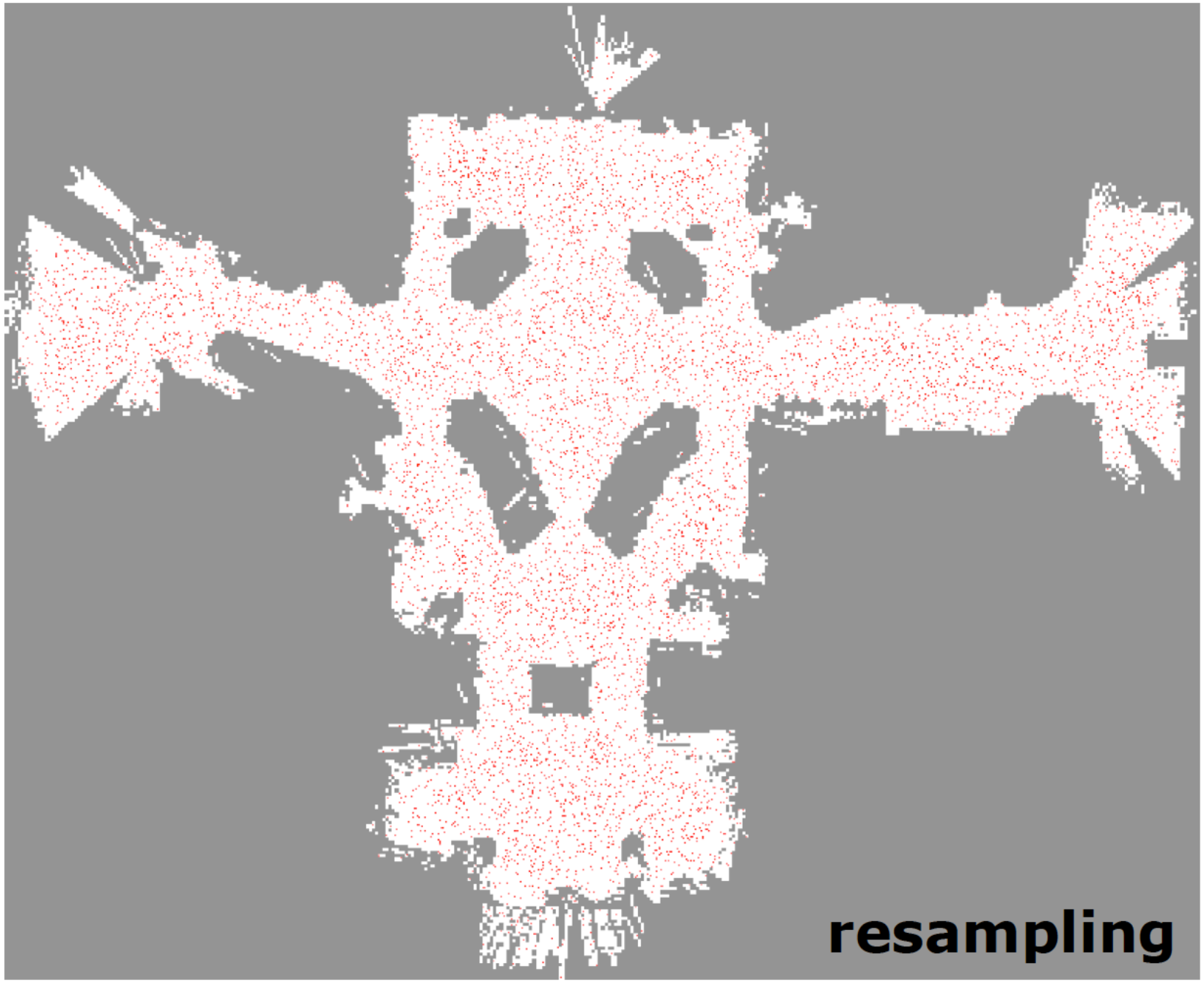
$$\propto p(z_t \mid x_t)$$

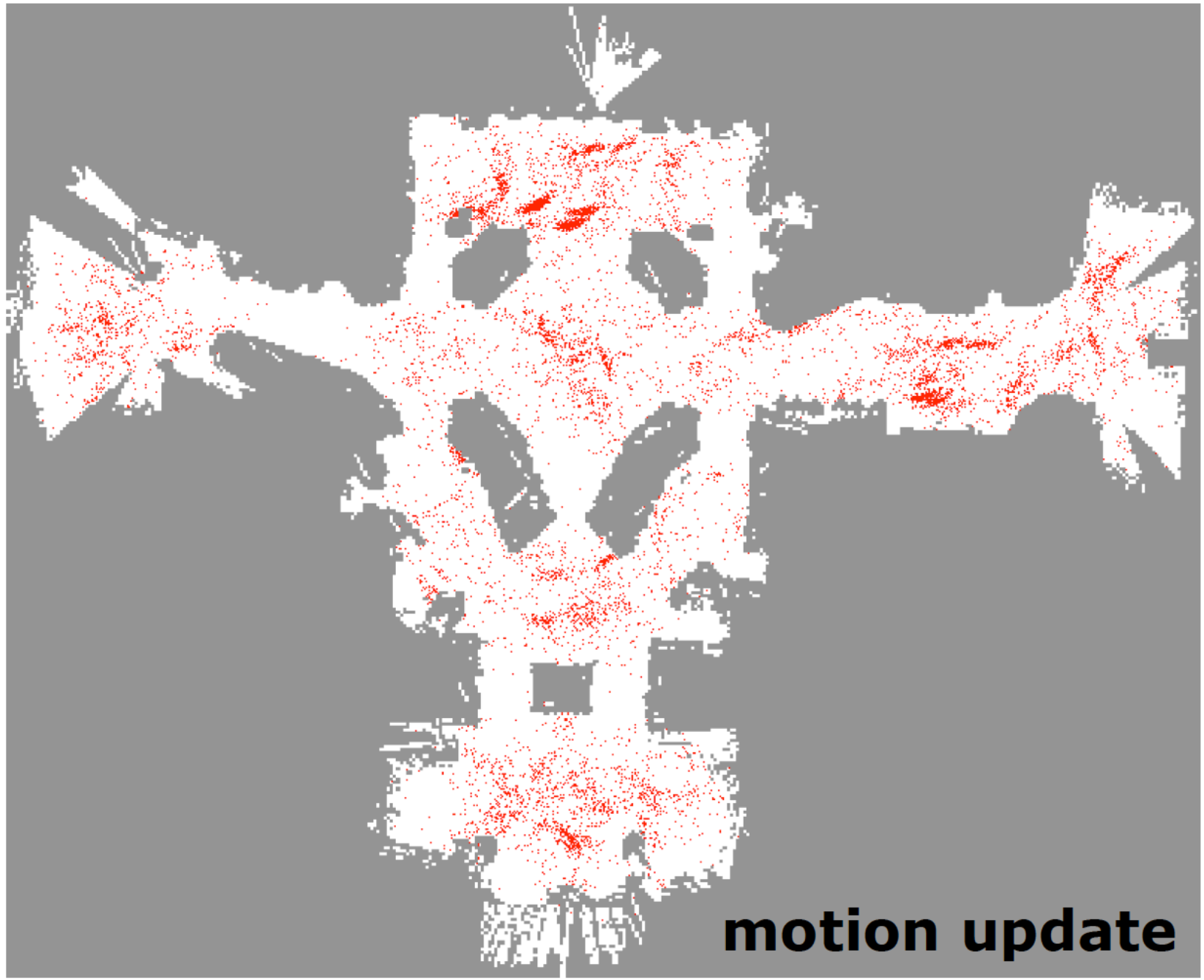
Monte Carlo Localization

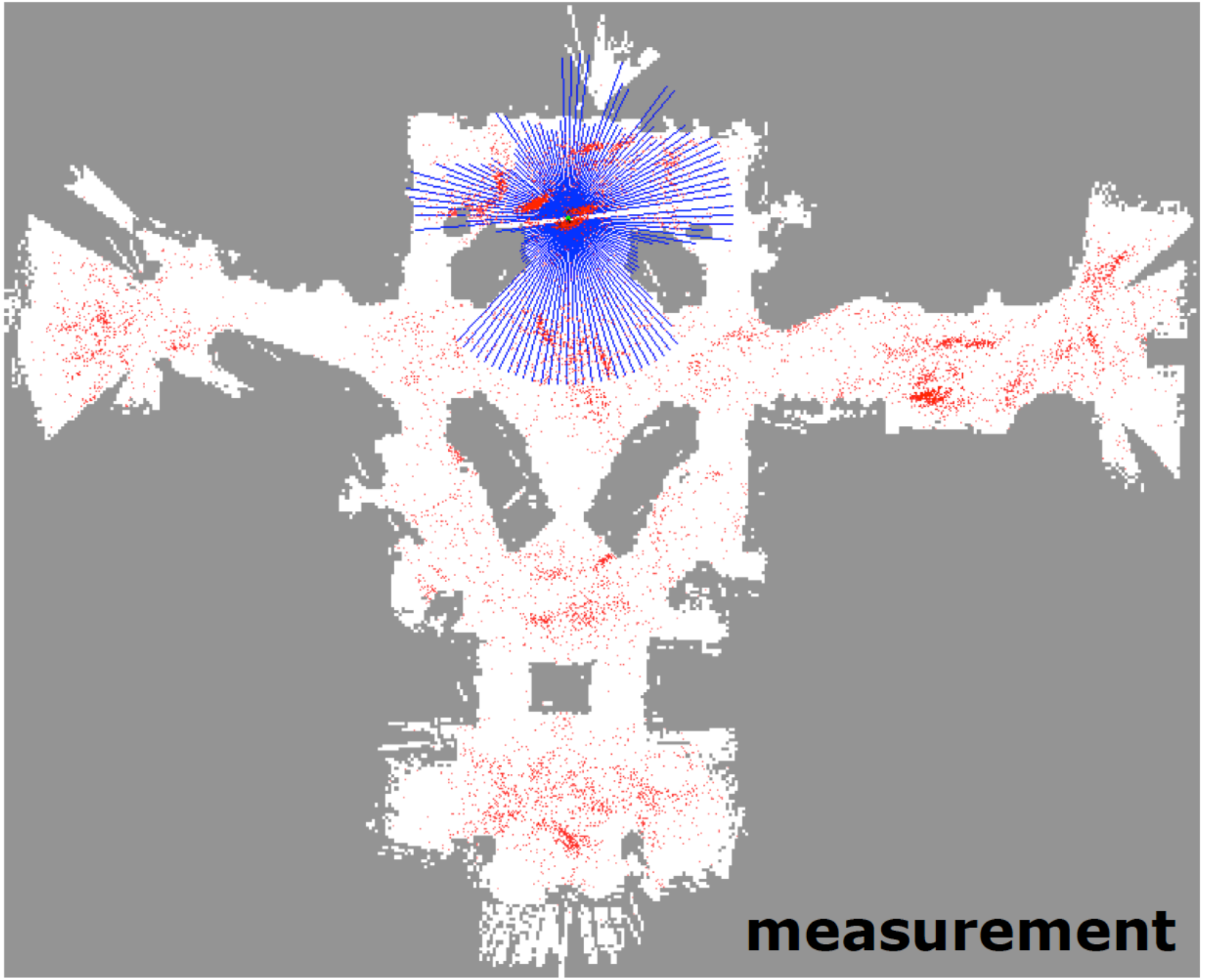



initialization

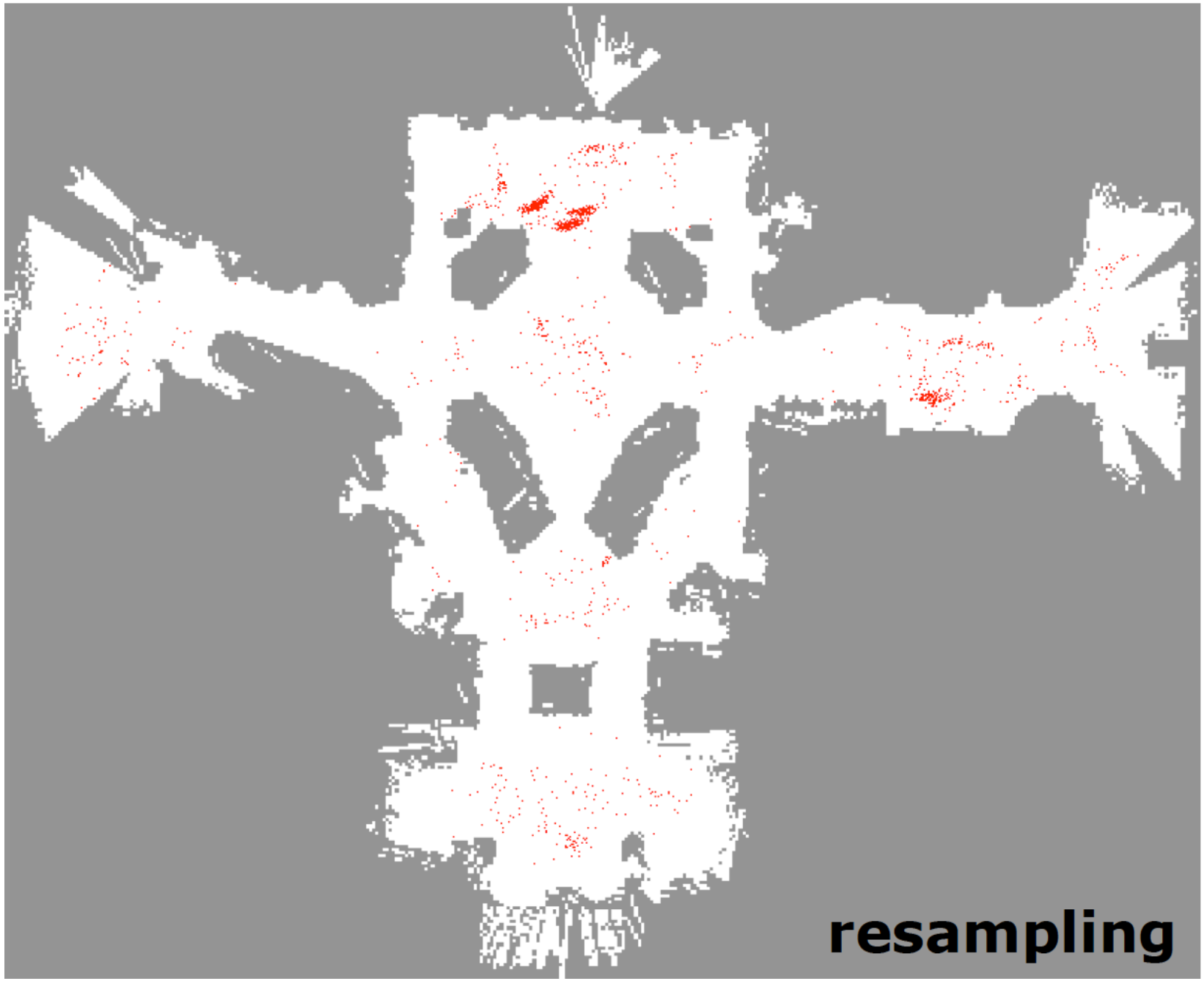

observation

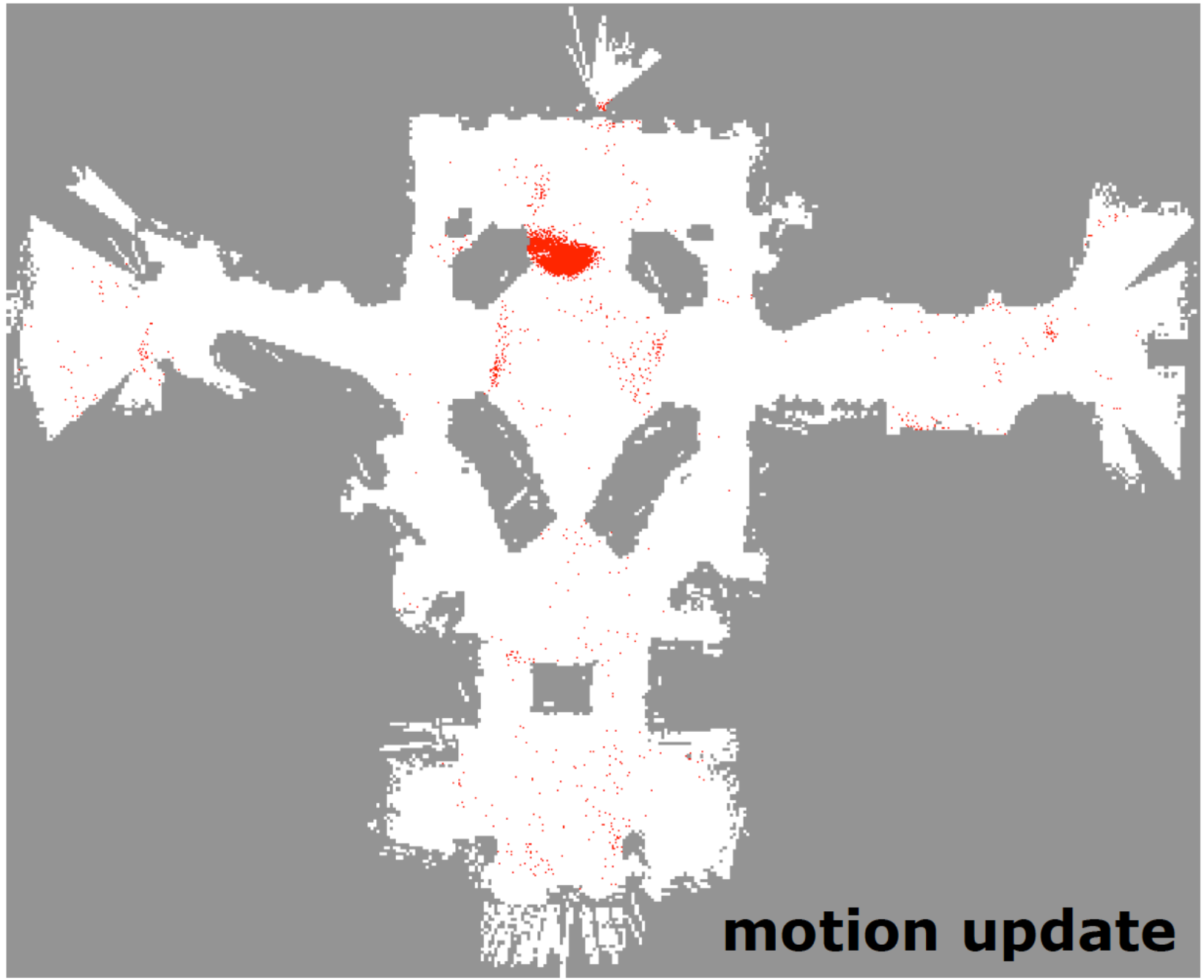

resampling

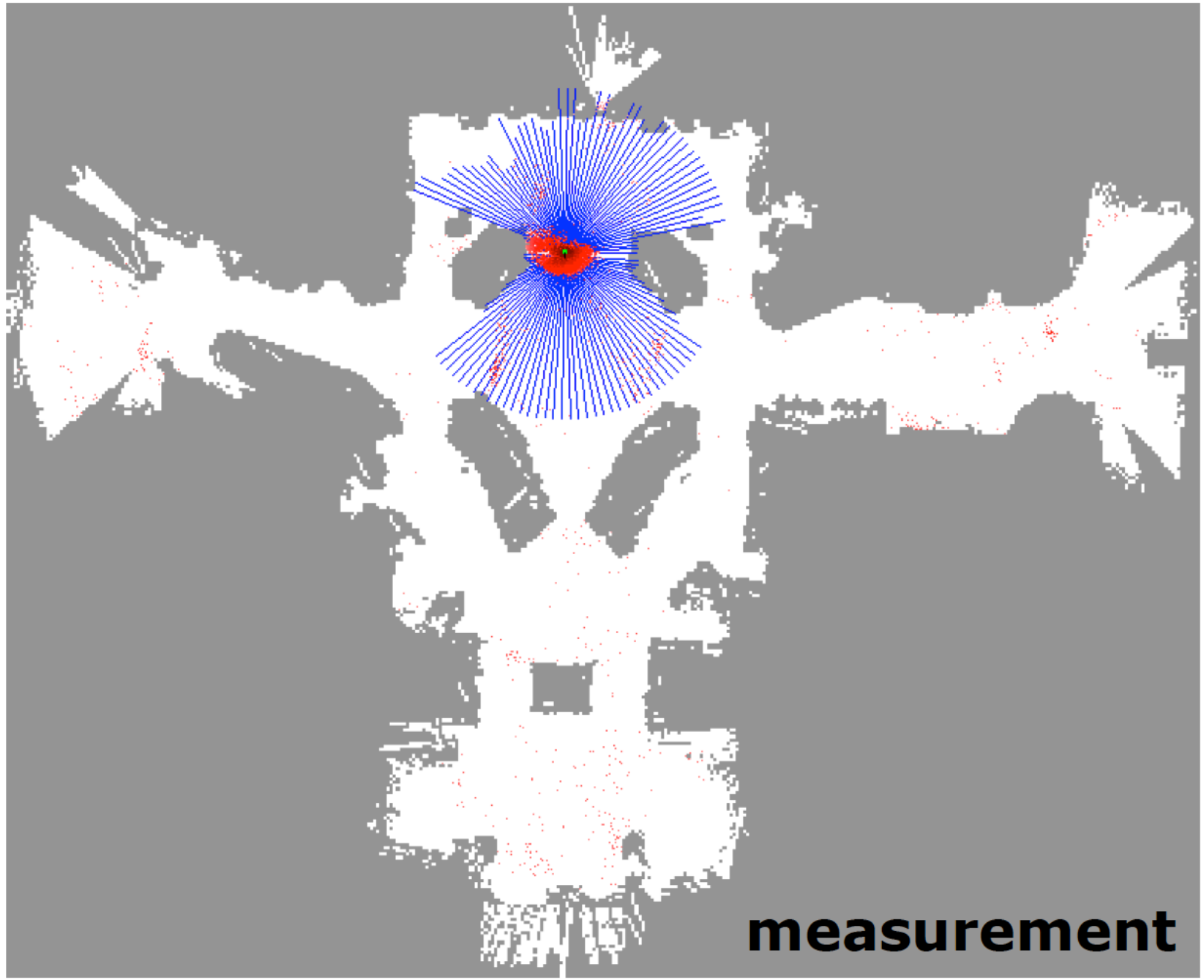

motion update


measurement

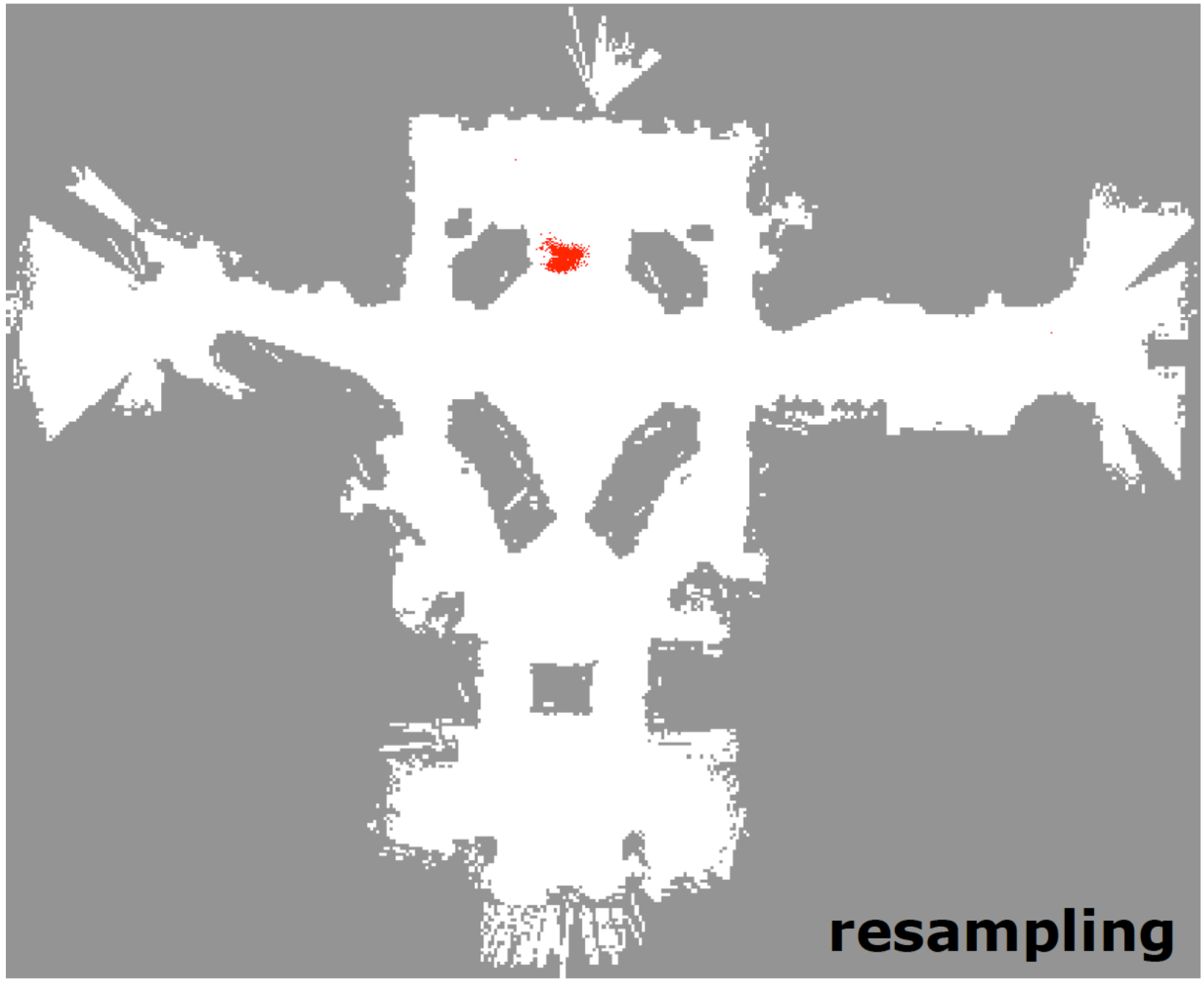

weight update

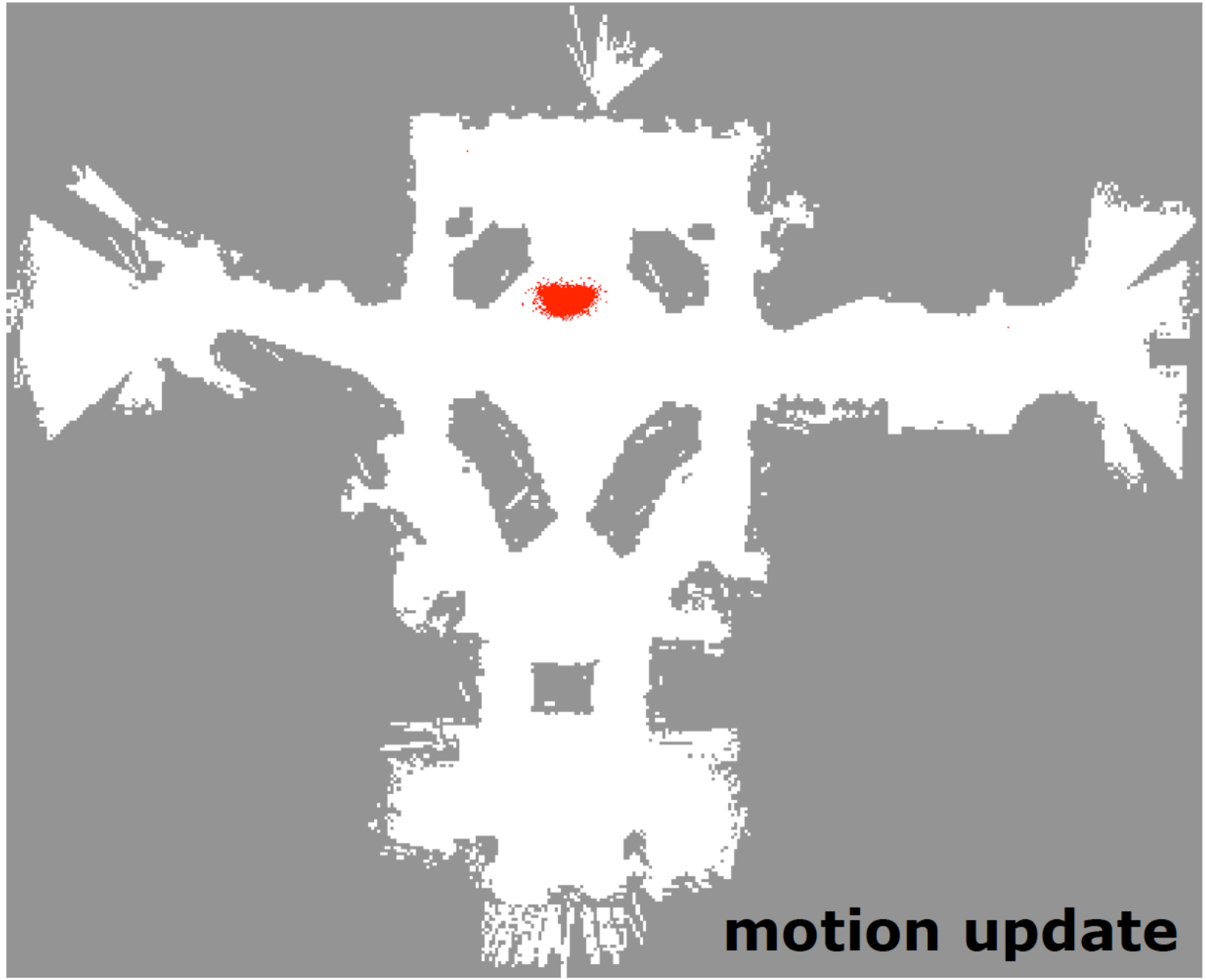

resampling

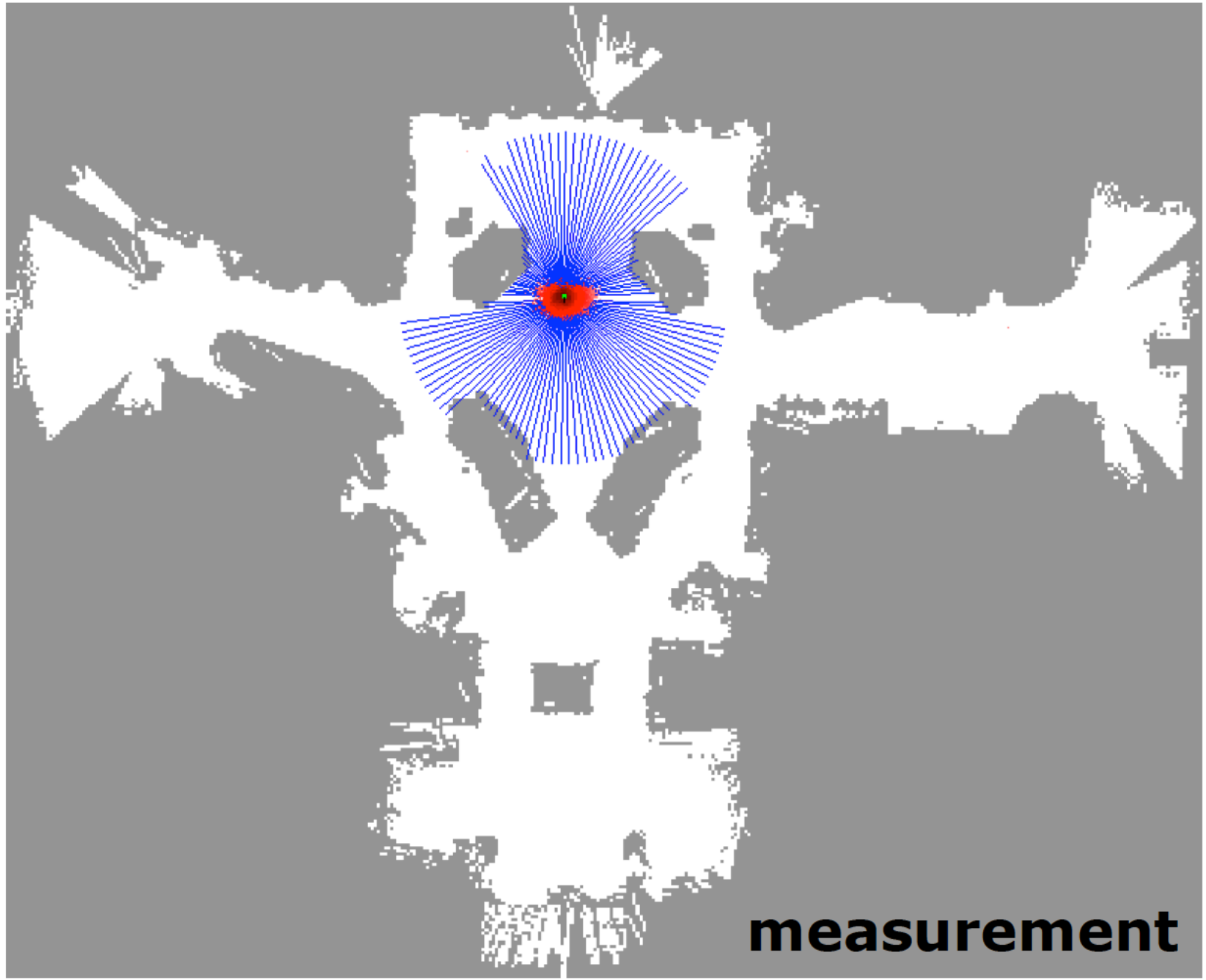

motion update


measurement

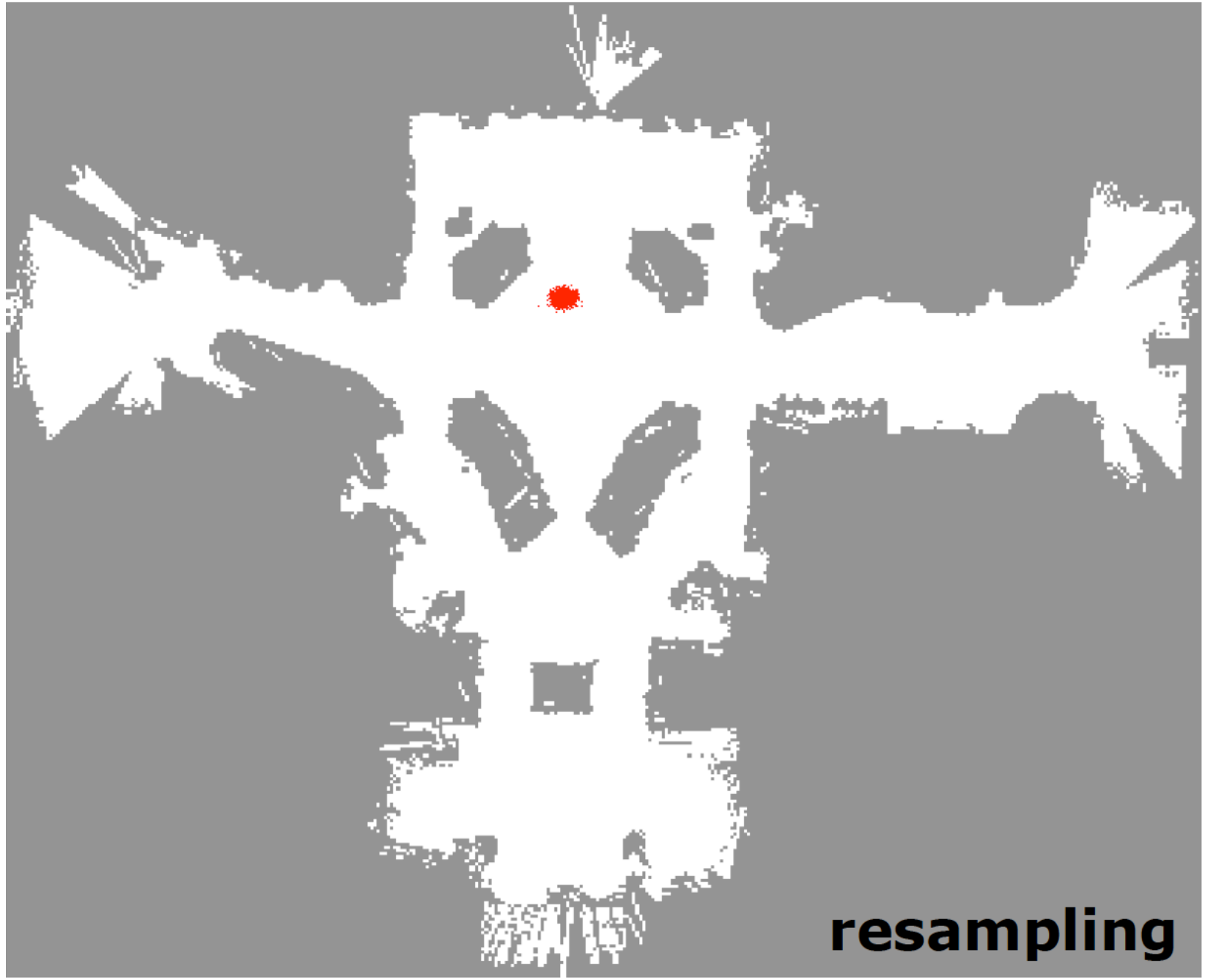

weight update


resampling


motion update


measurement


weight update

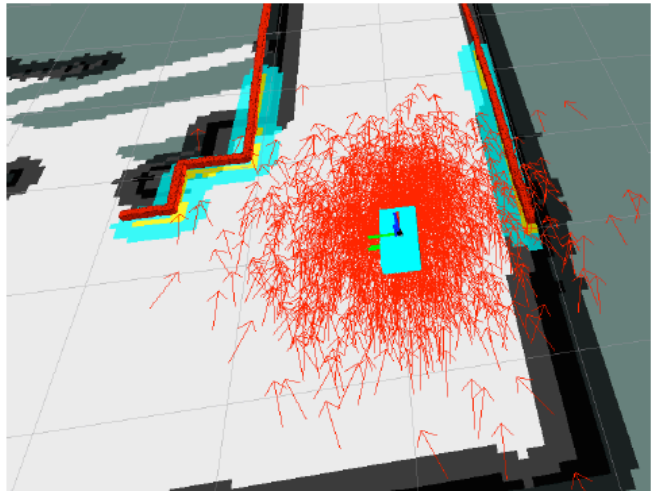

resampling

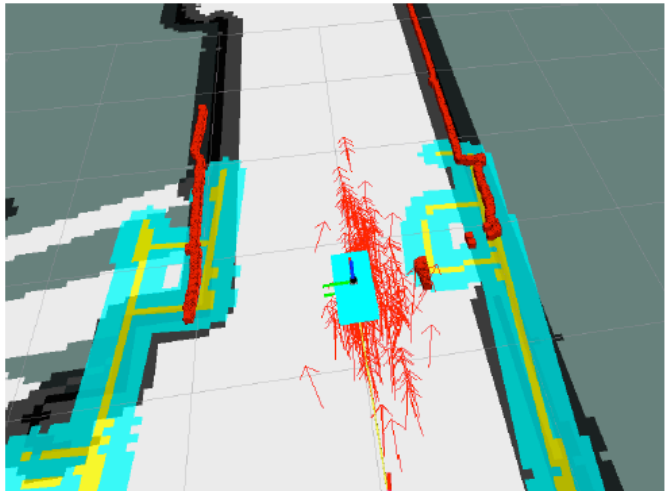
ROS Navigation Stack

AMCL

AMCL: Adaptive Monte Carlo Localization

- amcl is a probabilistic localization system for a robot moving in 2D.
- adaptive (Kullback-Leibler Divergence KLD sampling) Monte Carlo localization approach, using a particle filter to track robot pose against a known existing map.
    - Variable Particle size
    - Sample size is proportional to error between odometry position and sample based approximation
    - i.e smaller sample size when particles have converged





AMCL in ROS

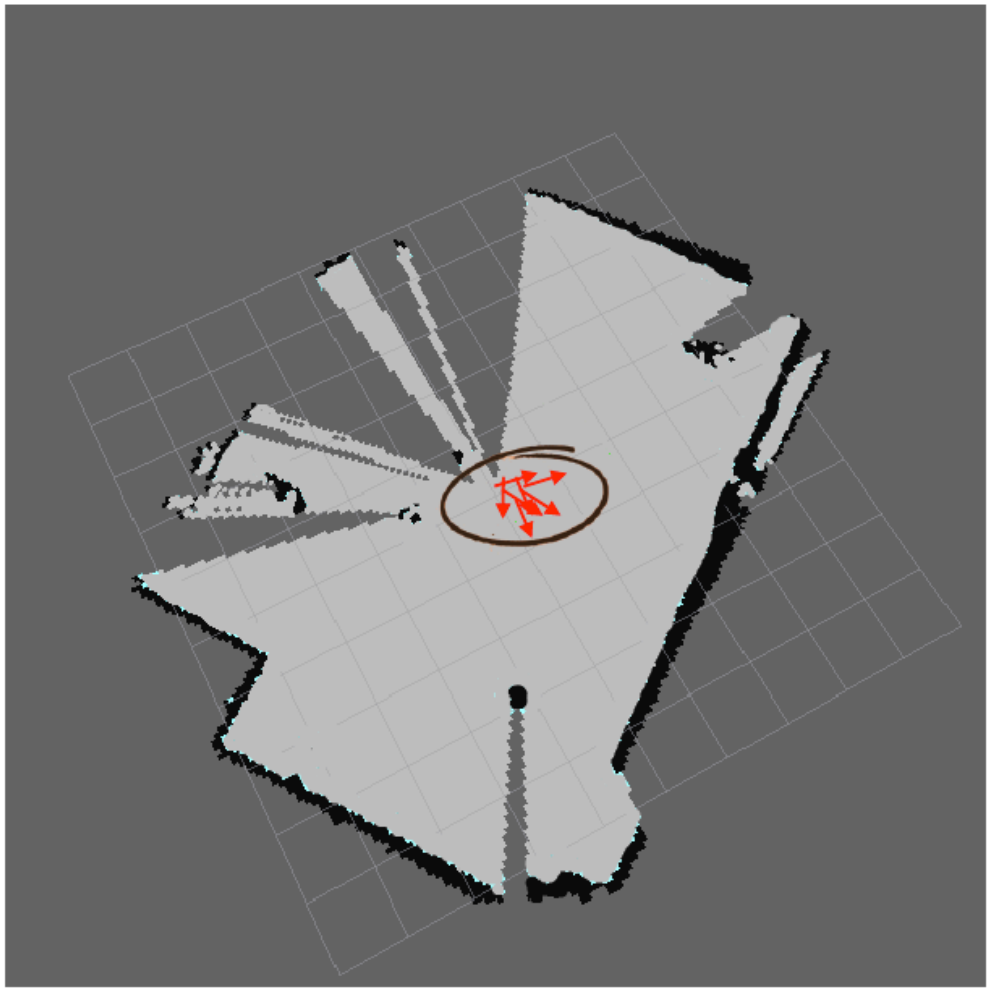
- relies on a “laser” (Kinect)

Input Parameters:

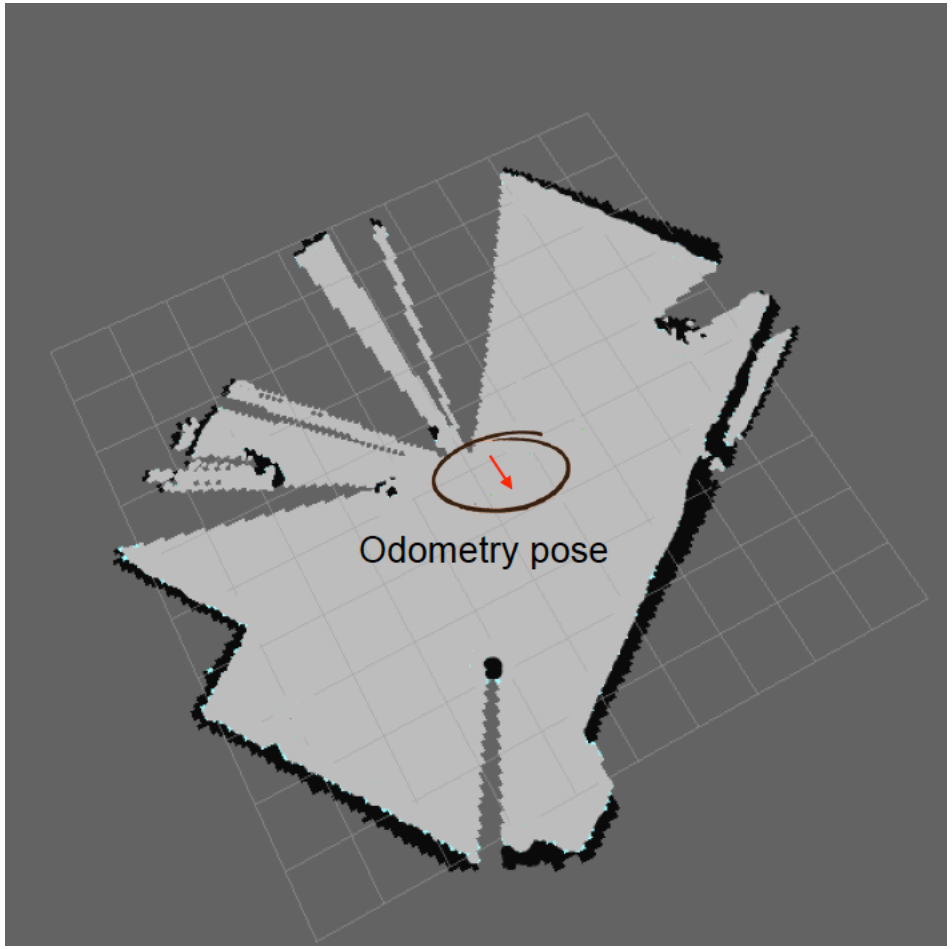
1. Laser Scan
2. Dead Reckoning/Odometry
3. Map

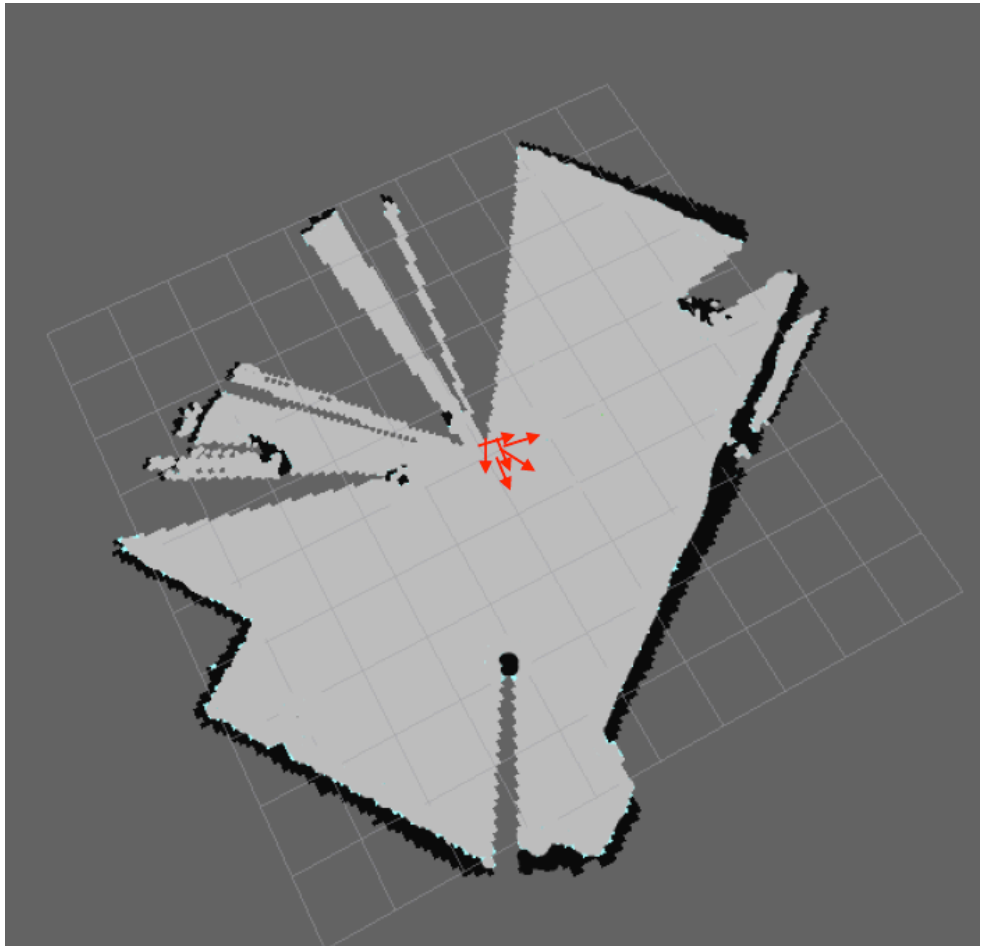
Output Parameters:

1. AMCL pose
2. Particle Cloud

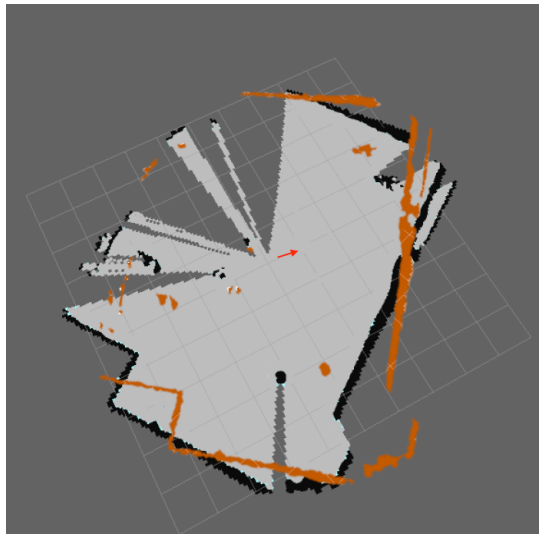
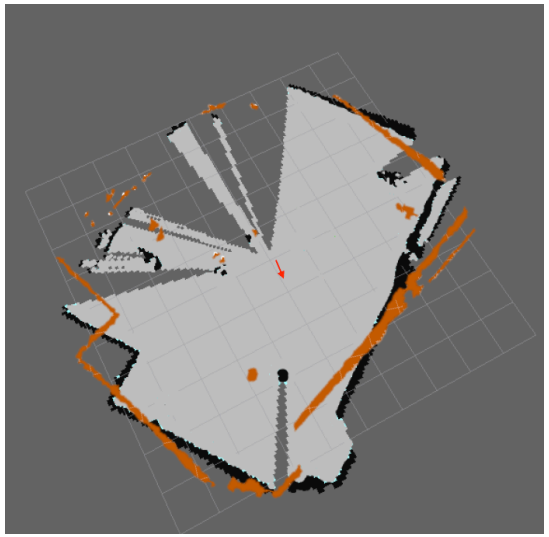
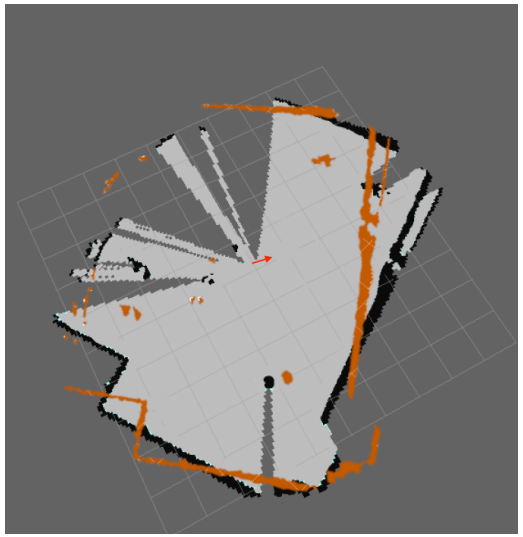
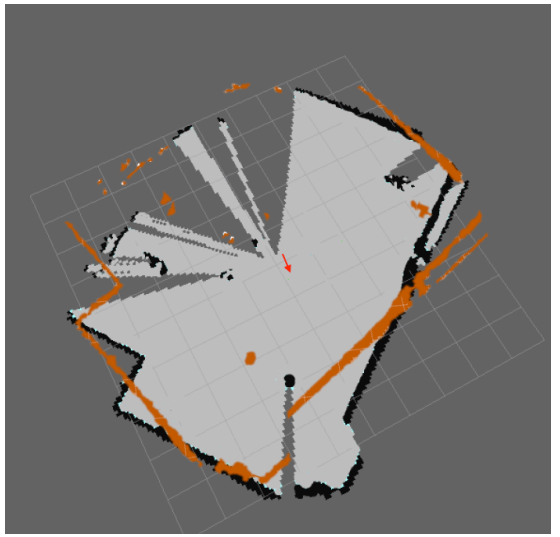
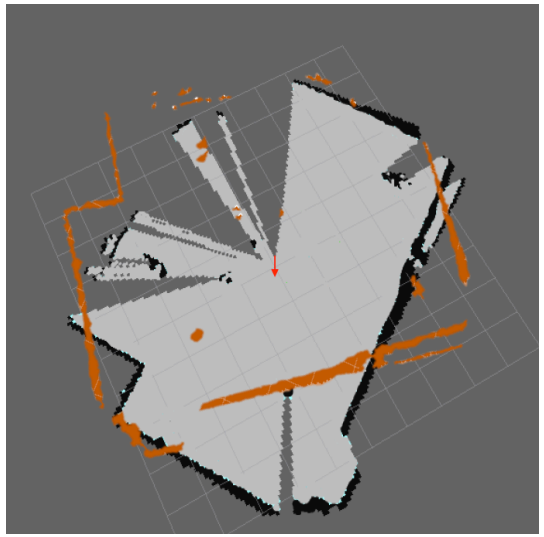
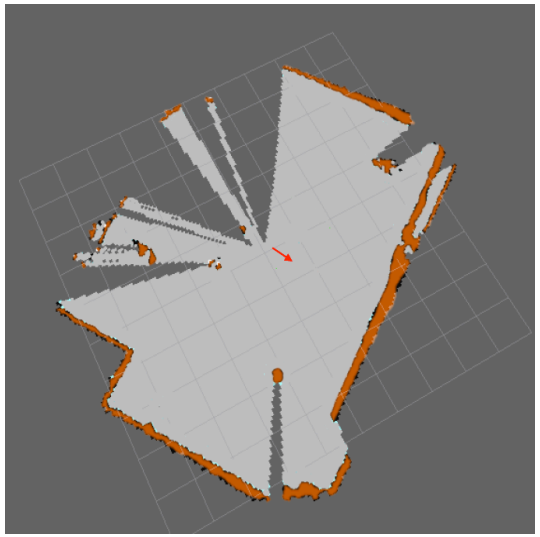


Particle Filter in 2D



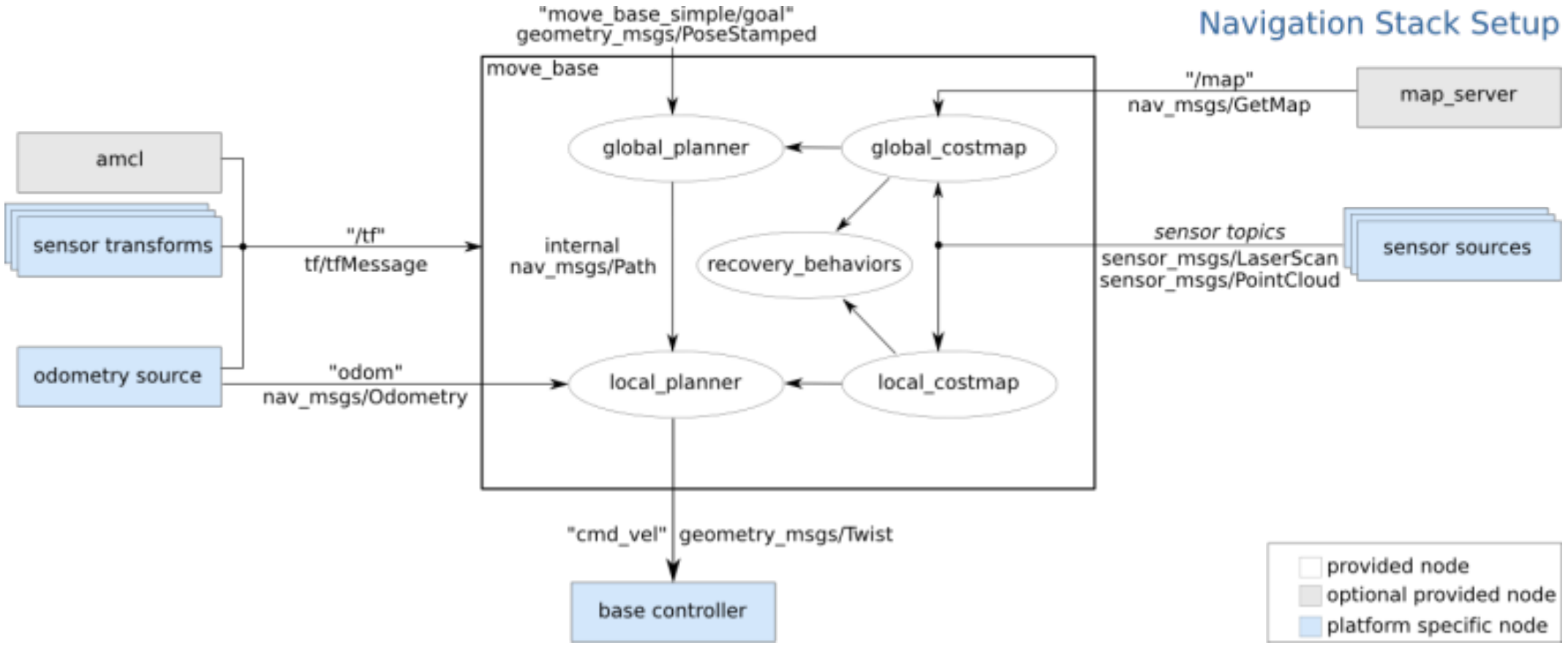


Scan Correction

Particle	Weight
P1	W1
P2	W2
P3	W3
P4	W4
P5	W5
P6	W6

Navigation Stack



Building a Map

- GMapping implements the FastSLAM 2.0 algorithm
- map is an occupancy map and it is represented as an image showing the blueprint of the environment

Navigation Stack in ROS