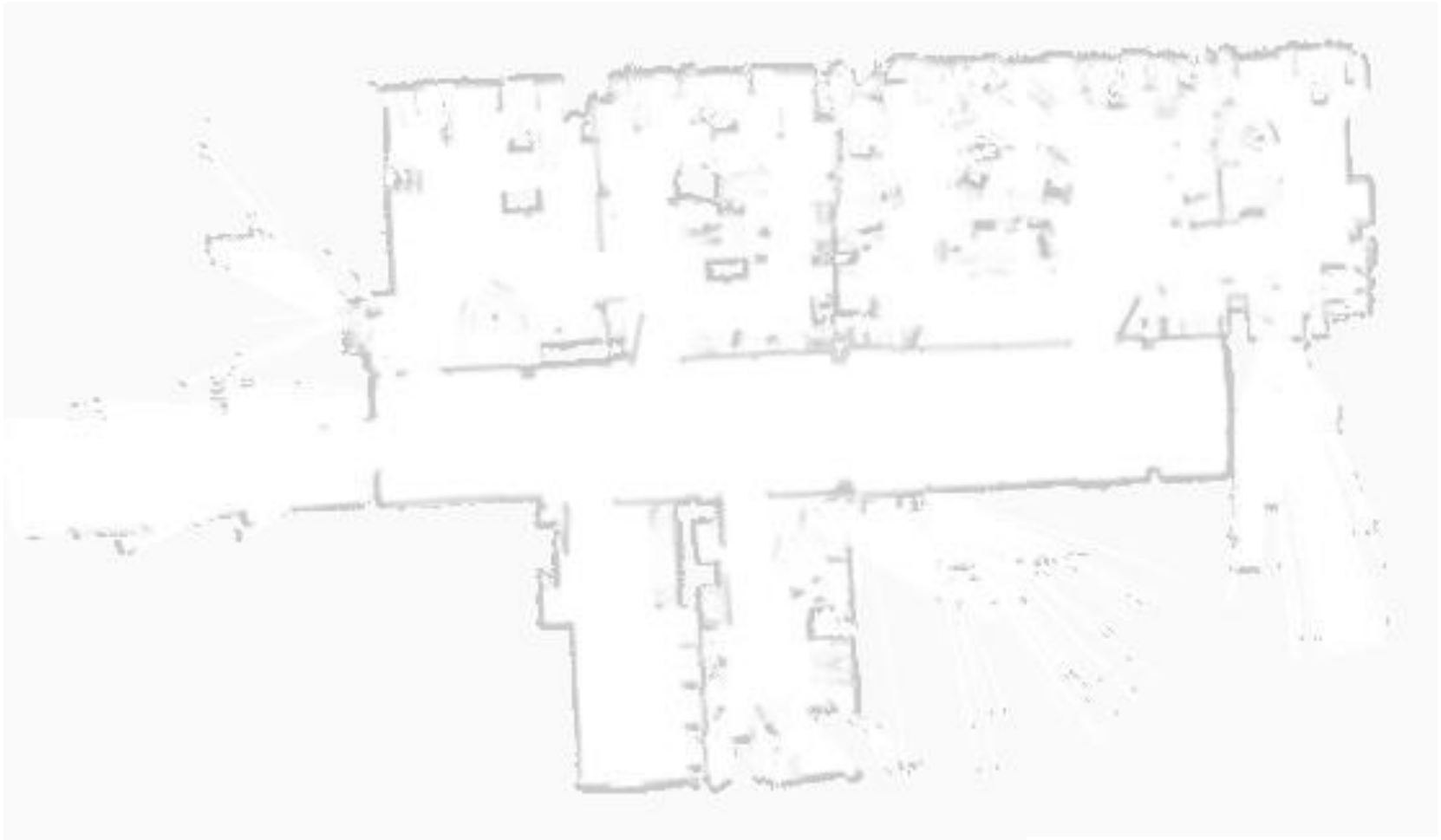
Particle Filter and Monte Carlo Localization

Nived Chebrolu

Slides adapted from Cyrill Stachniss at Uni. Bonn

“Where Am I?”



“Where Am I?”

Gaussian Filters

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

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

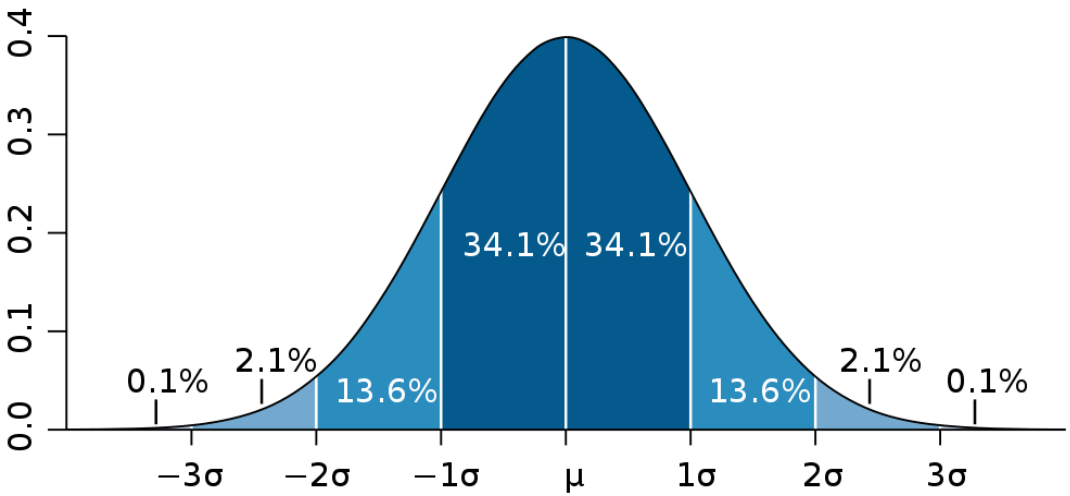
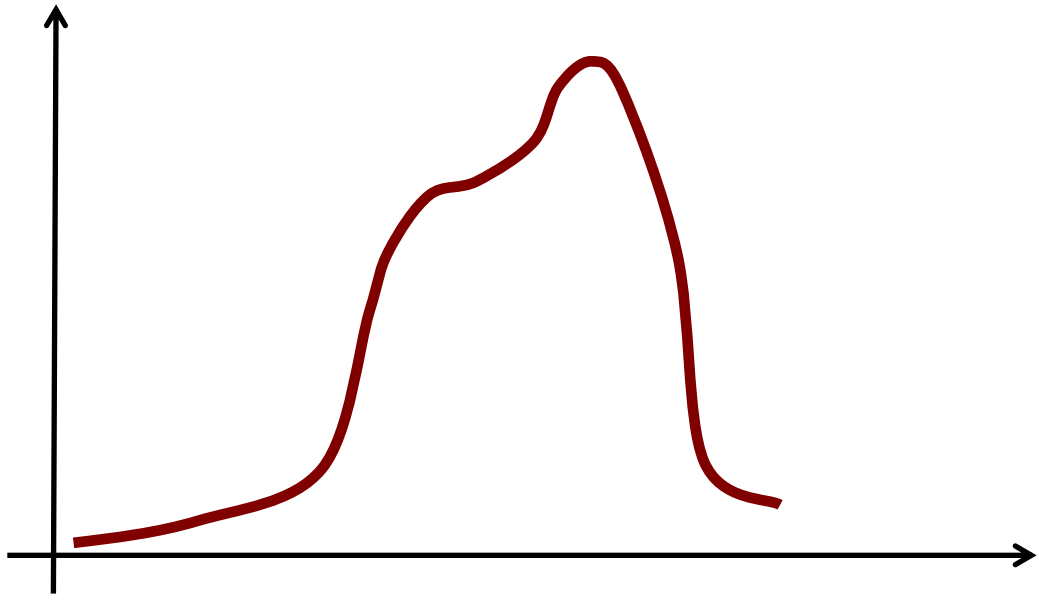


Image courtesy: K. Arras

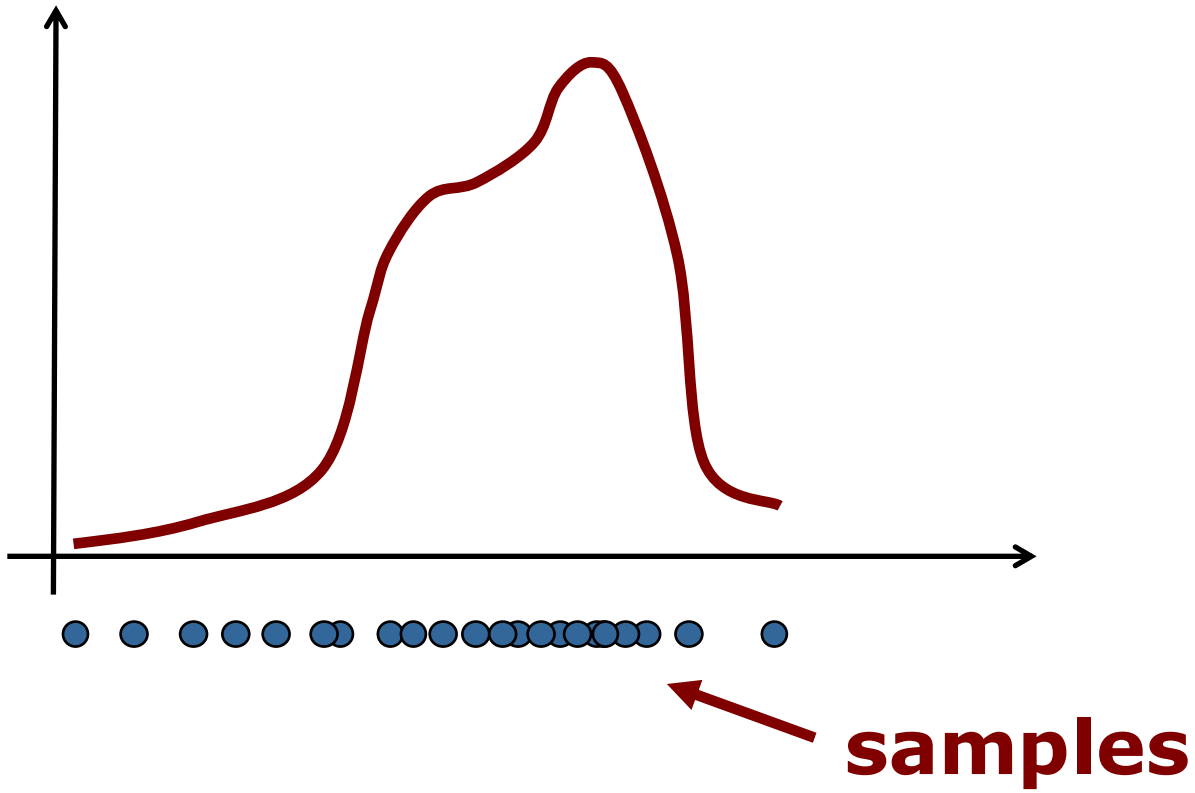
Flexible Function Approximation

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



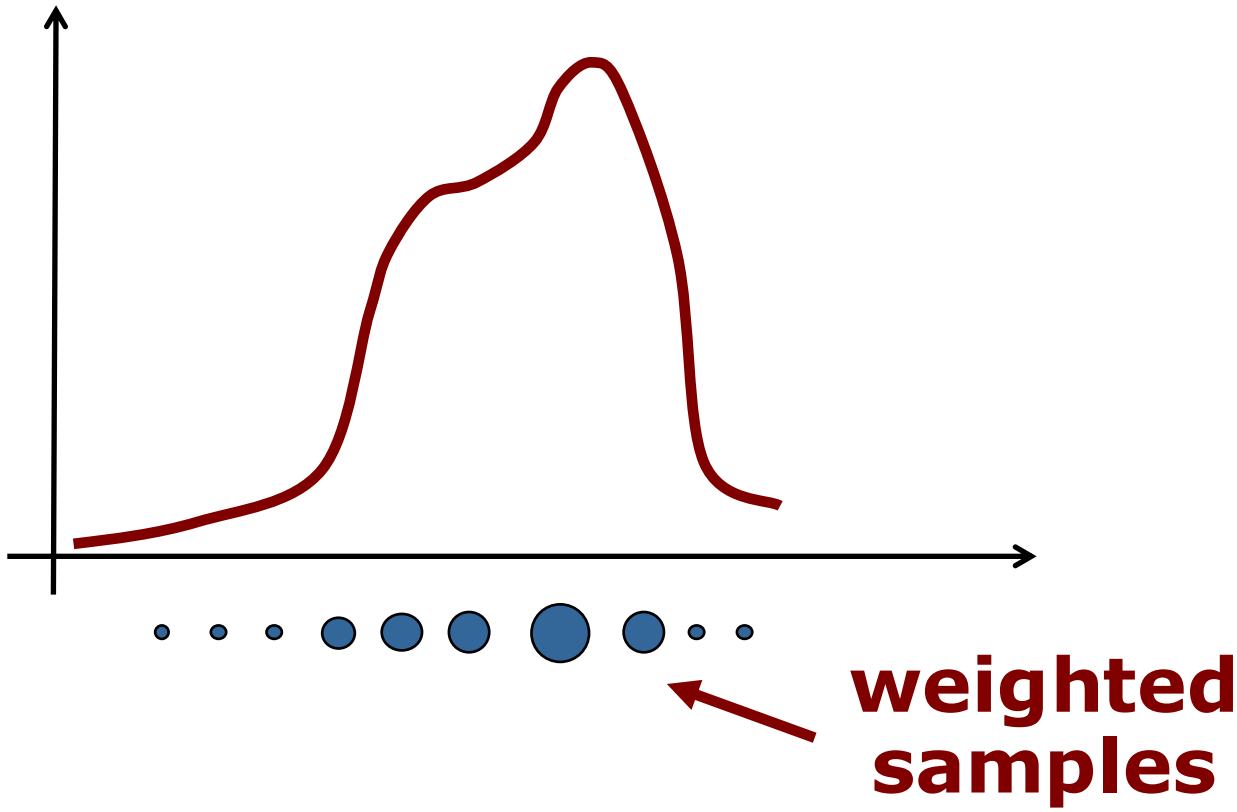
Key Idea: Samples

Multiple samples to represent arbitrary distributions



Key Idea: Weighted Samples

Multiple **weighted 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

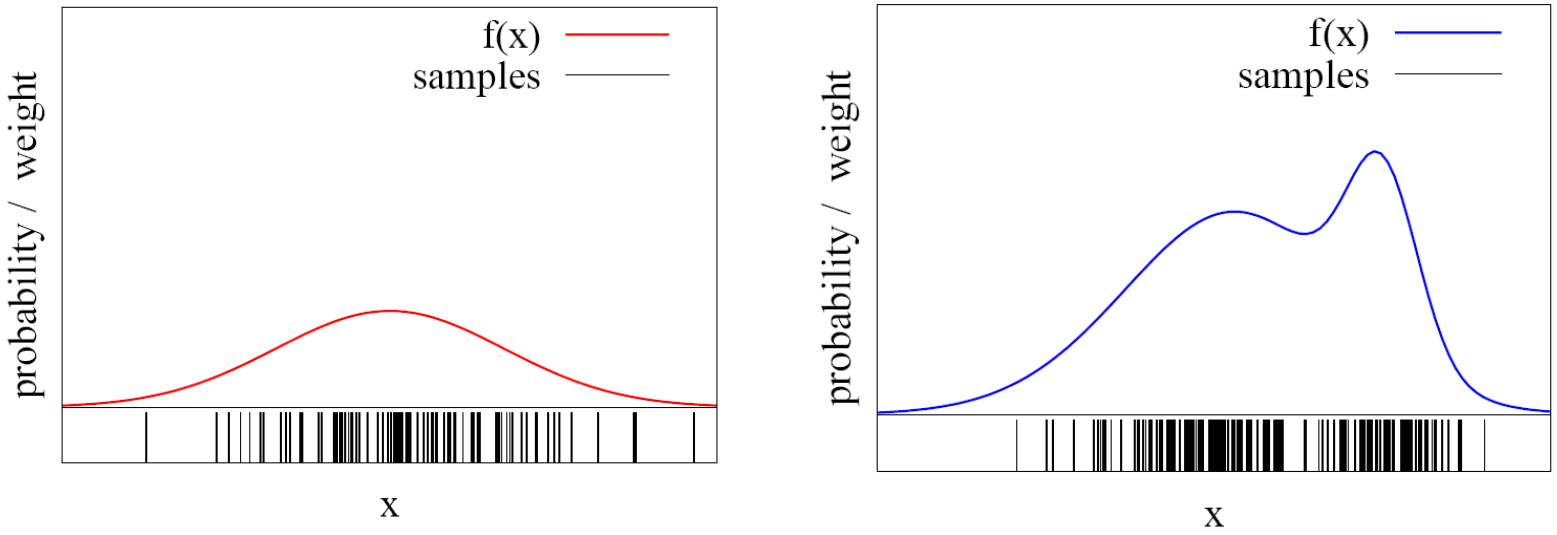
importance weight

- The samples represent the posterior

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

Particles for Approximation

- Particles for function approximation

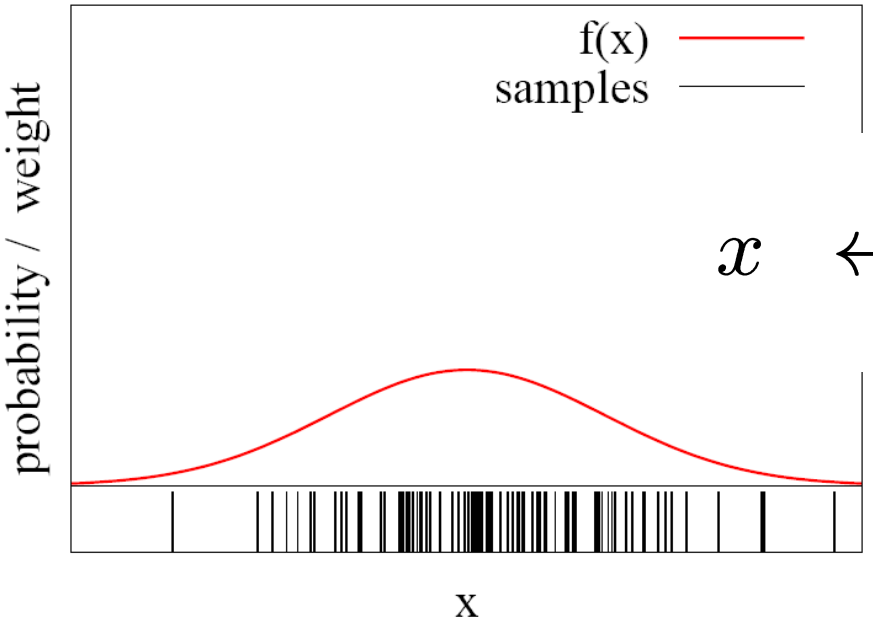


- The more particles fall into a region, the higher the probability of the region

How to obtain such samples?

Closed Form Sampling is Only Possible for a Few Distributions

- Example: Gaussian



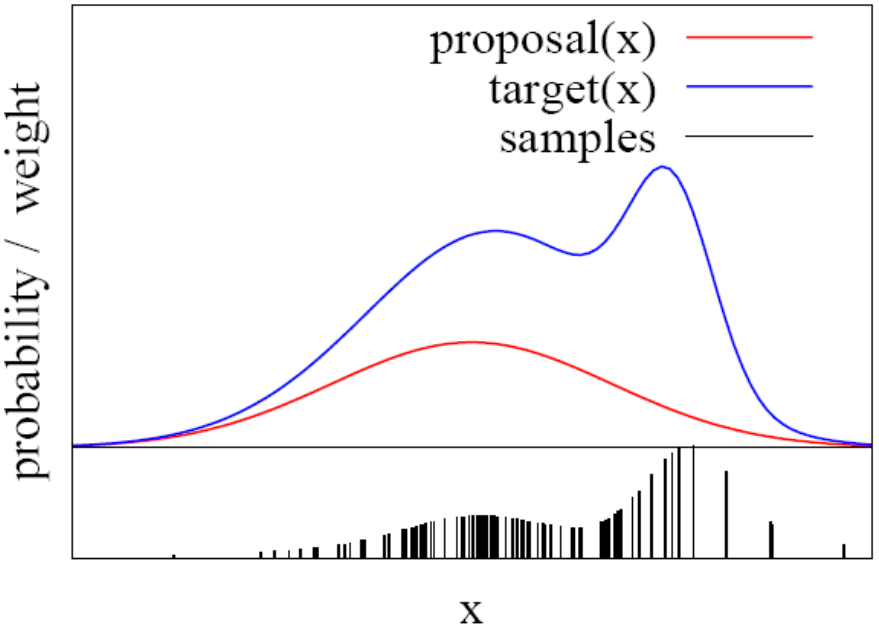
$$x \leftarrow \frac{1}{2}\sum_{i=1}^{12} \text{rand}(-\sigma, \sigma)$$

How to sample from **other** distributions?

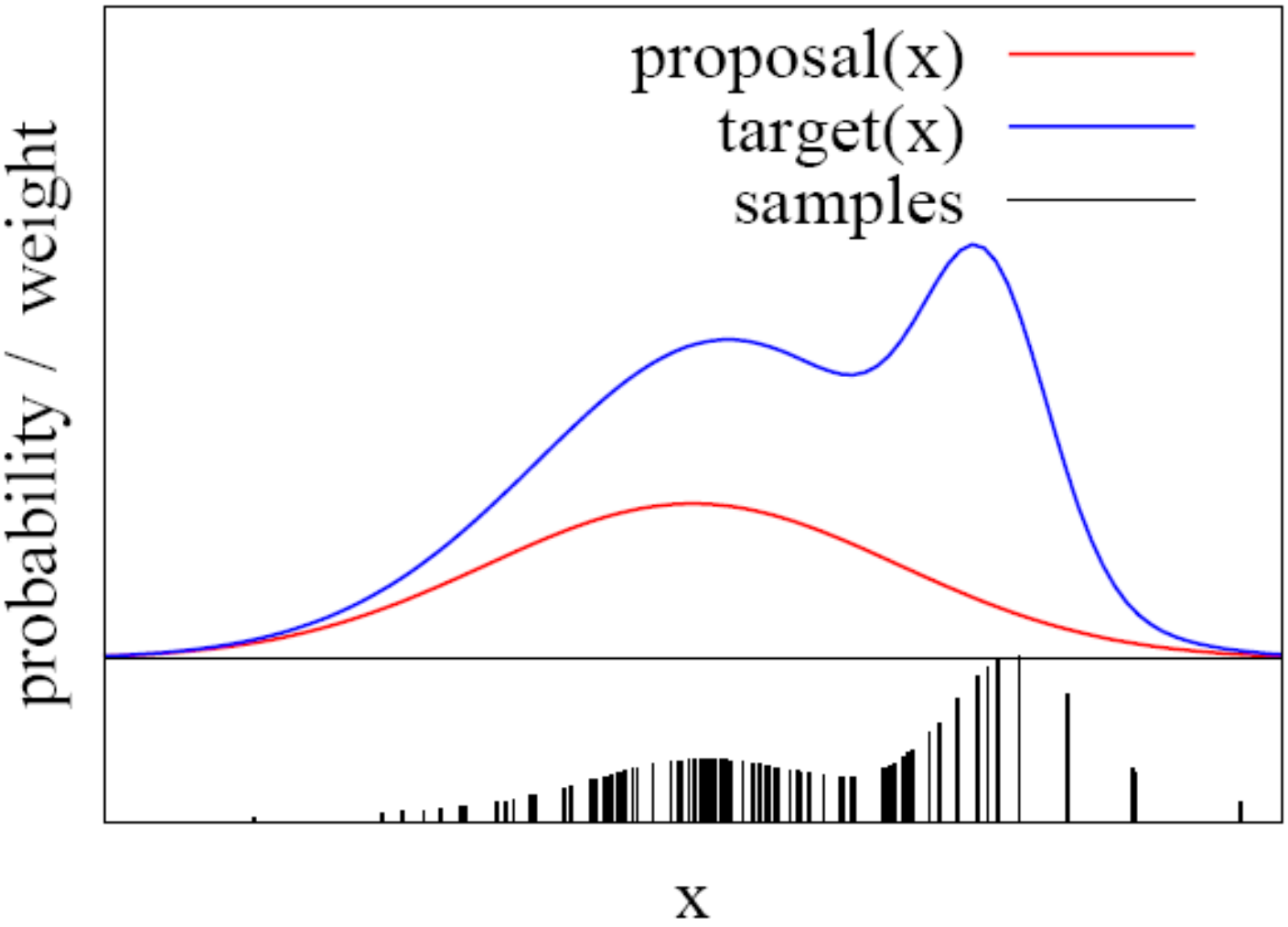
Importance Sampling Principle

- We can use a different distribution π to generate samples from f
- Account for the "differences between π and f" using a weight $\omega = f(x)/\pi(x)$
- target f
- proposal π
- Pre-condition:

$$f(x) > 0 \rightarrow \pi(x) > 0$$



Importance Sampling Principle



Particle Filter for Dynamic State Estimation Problems

- Recursive Bayes filter
- Non-parametric approach
- Models the distribution by samples
- **Prediction:** draw 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

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

2. Compute the importance weights

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

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

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

Monte Carlo Localization: Solve "Where Am I?" Using Particles

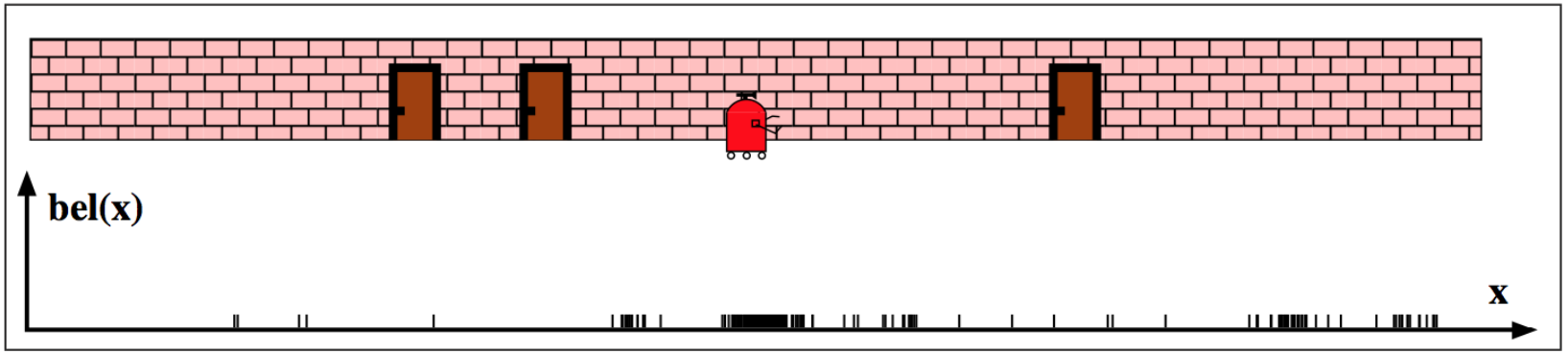
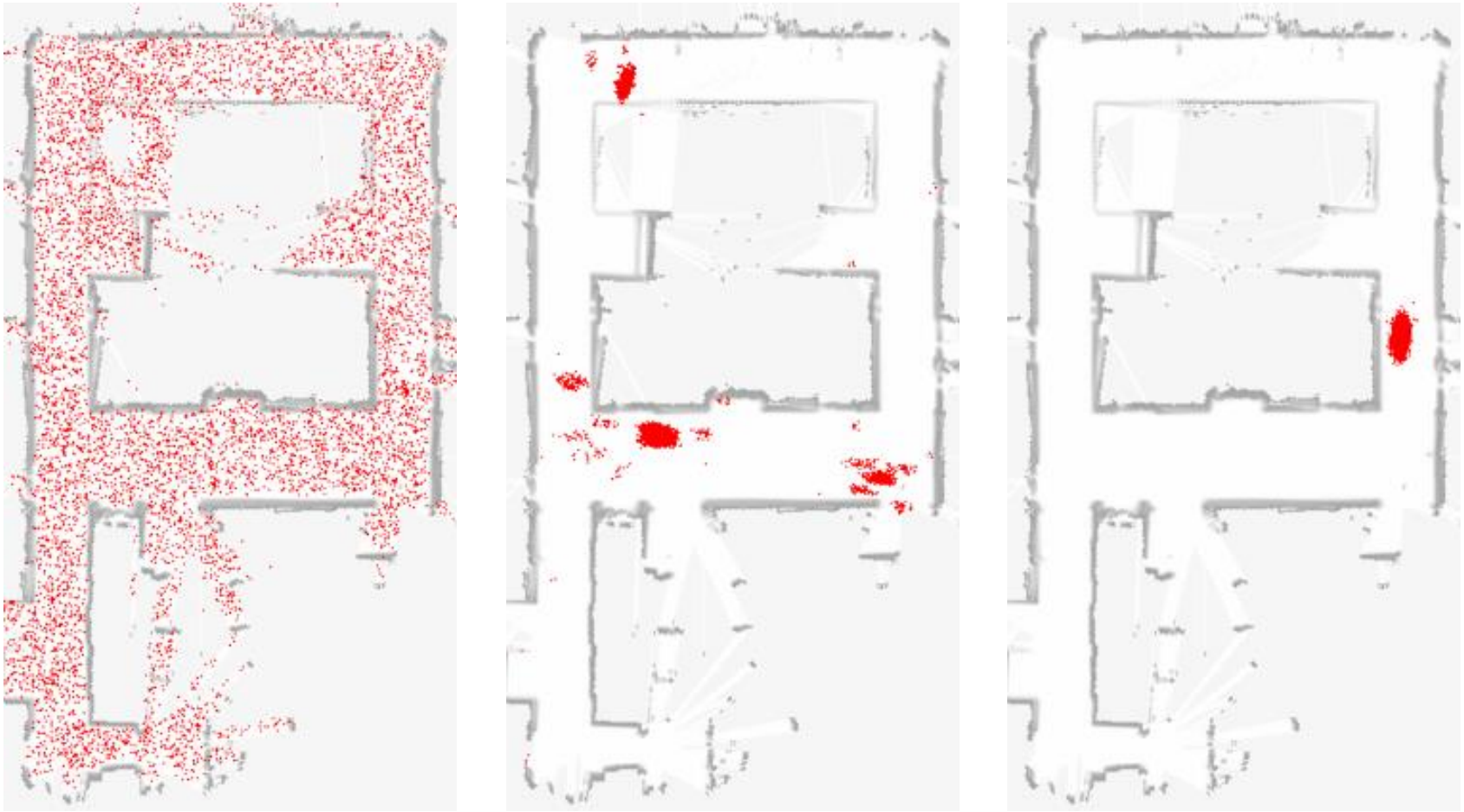


Image Courtesy: S. Thrun, W. Burgard, D. Fox

Monte Carlo Localization: Solve "Where Am I?" Using Particles

Monte Carlo Localization: Solve "Where Am I?" Using Particles



Monte Carlo Localization

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

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

- Correction via the observation model

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

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* $\mathcal{X}_t$

MCL – Correct

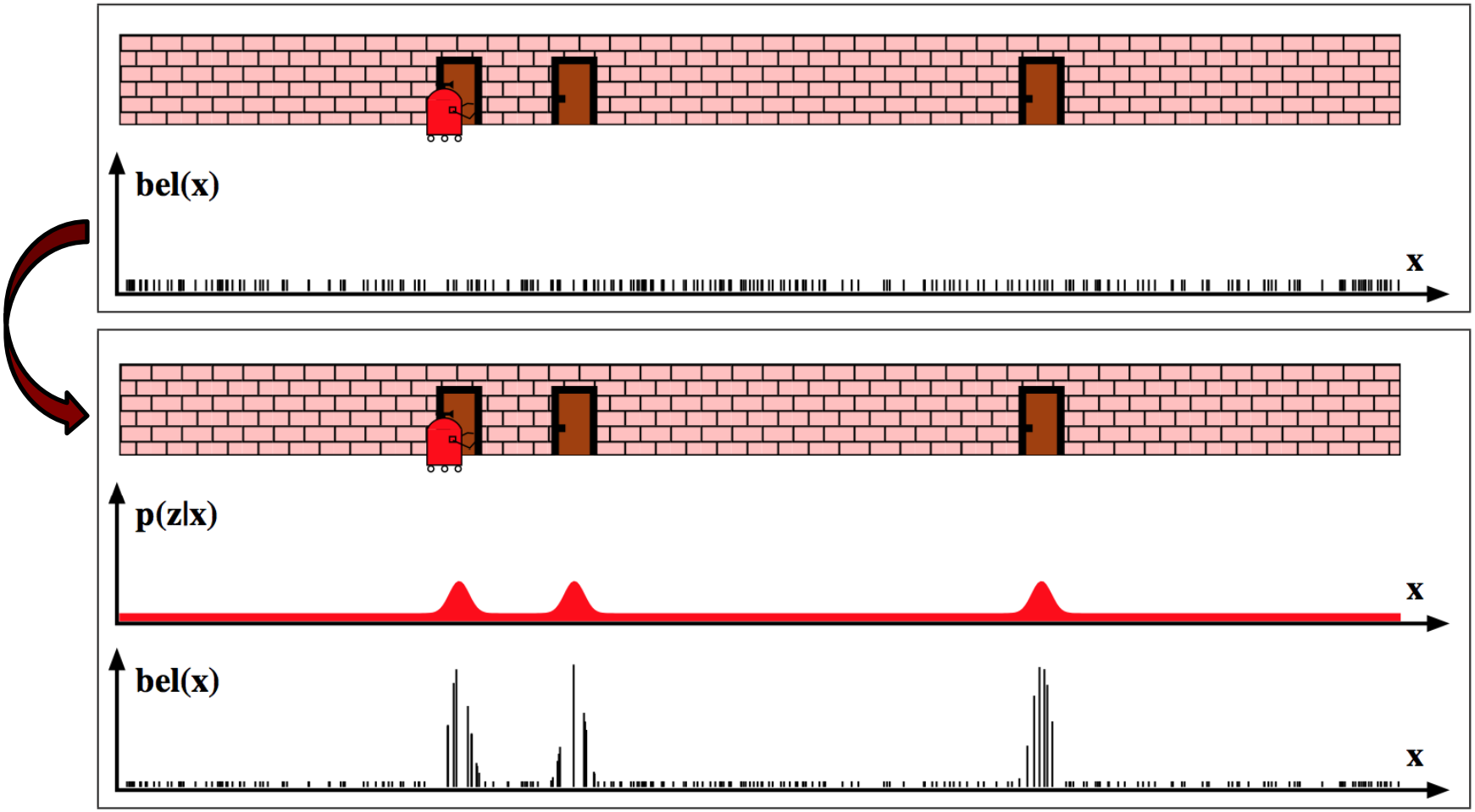


Image Courtesy: S. Thrun, W. Burgard, D. Fox

MCL – Resample & Predict

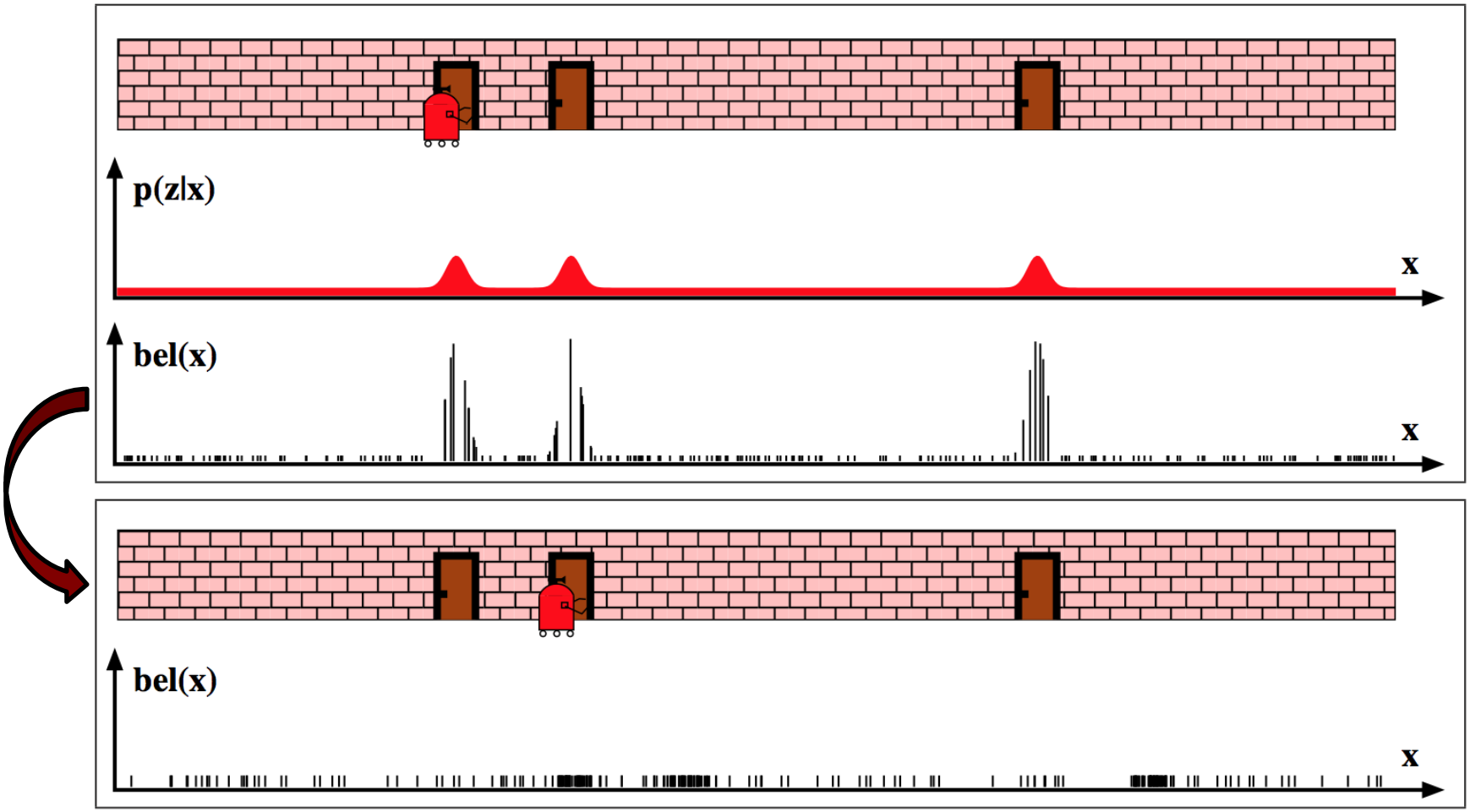


Image Courtesy: S. Thrun, W. Burgard, D. Fox

MCL – Correct

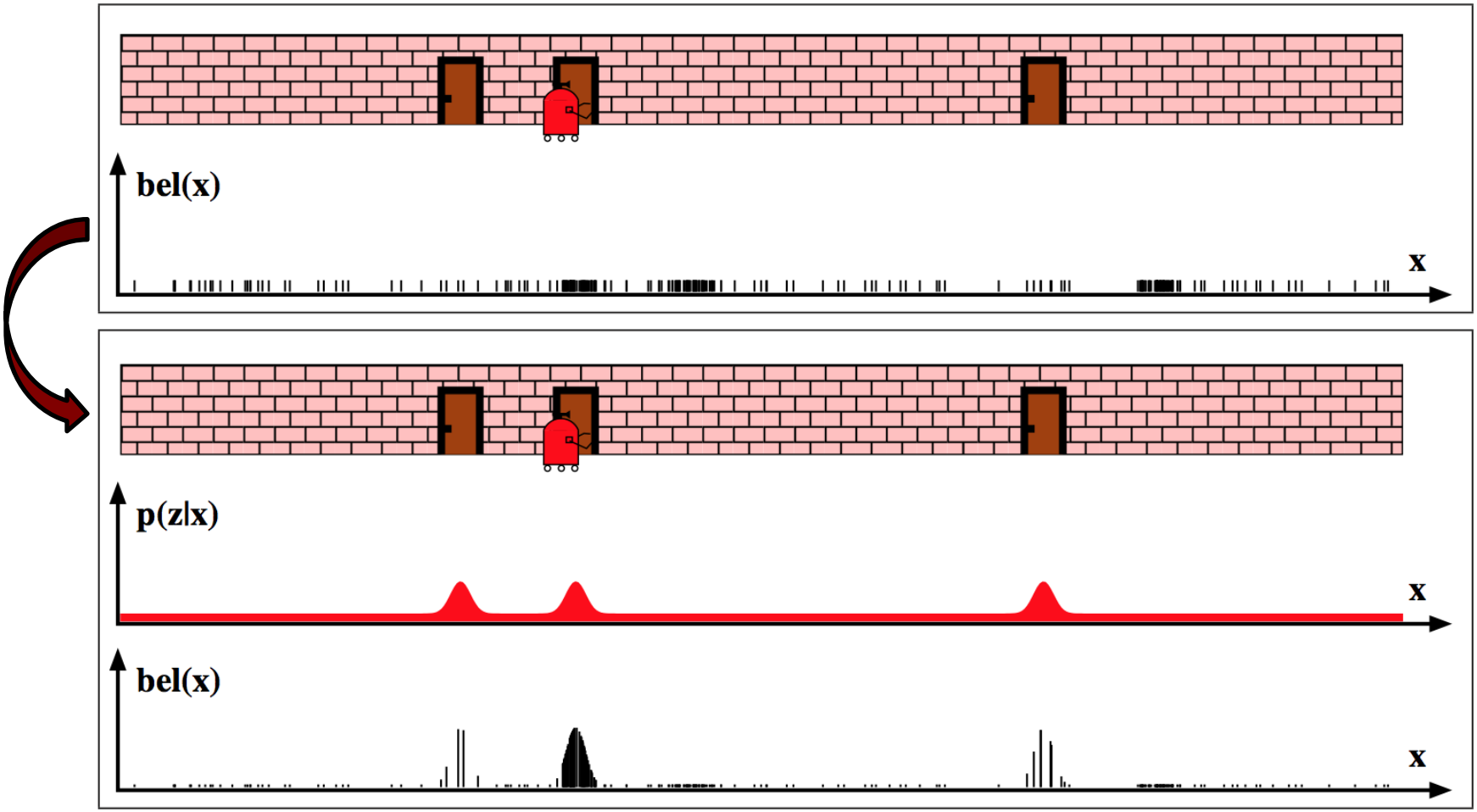


Image Courtesy: S. Thrun, W. Burgard, D. Fox

MCL – Resample & Predict

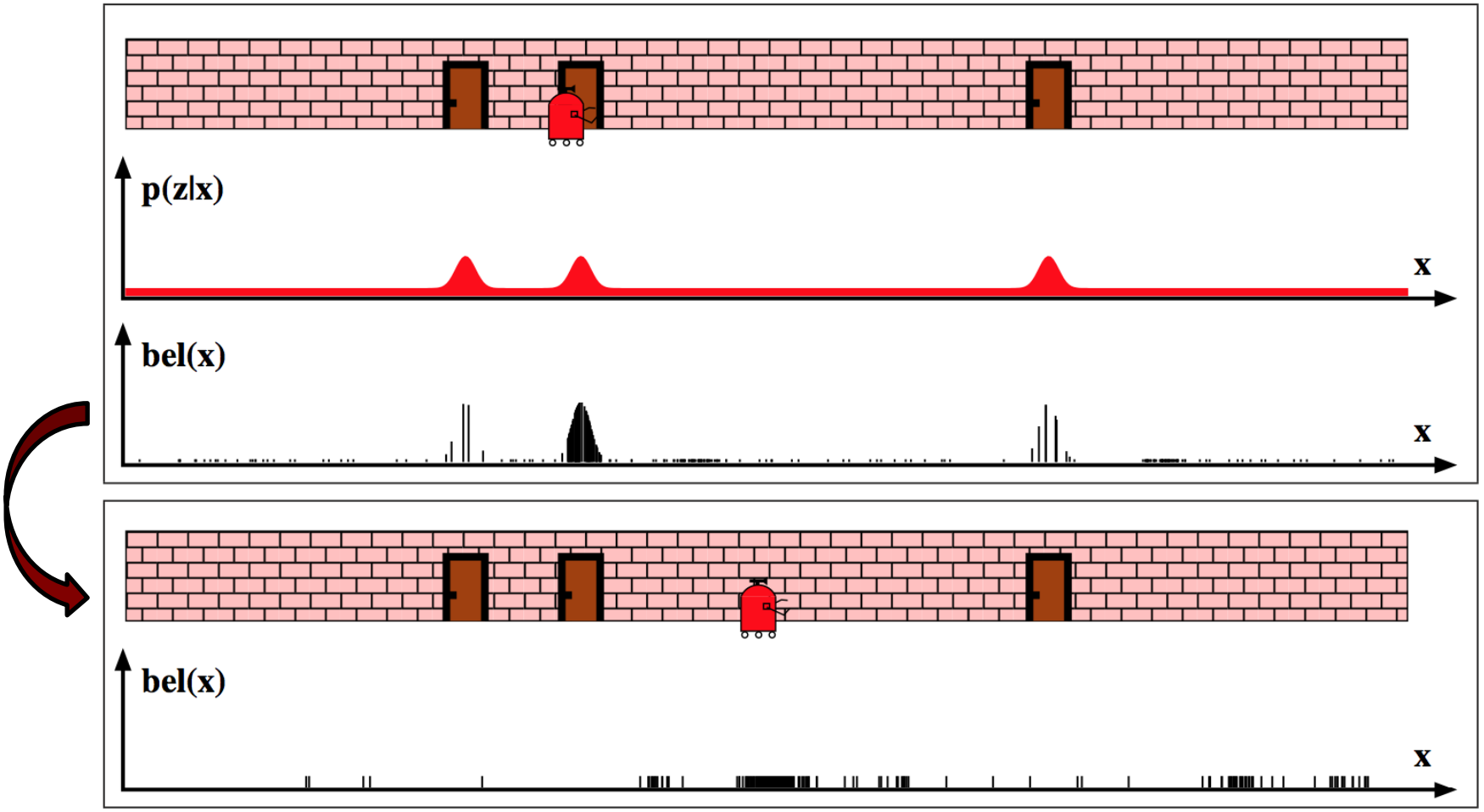
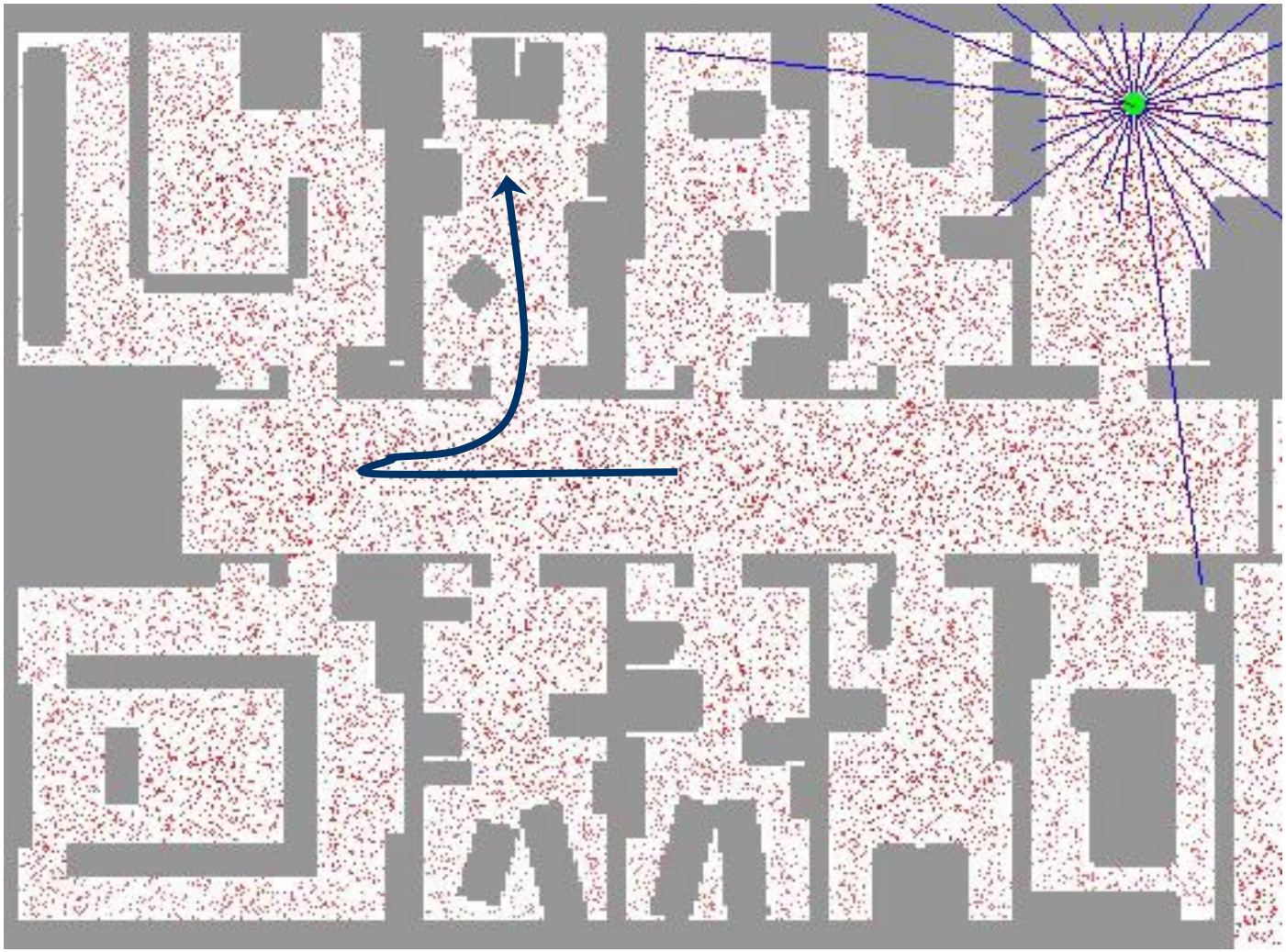


Image Courtesy: S. Thrun, W. Burgard, D. Fox

Application: Particle Filter for Localization in a Known Map

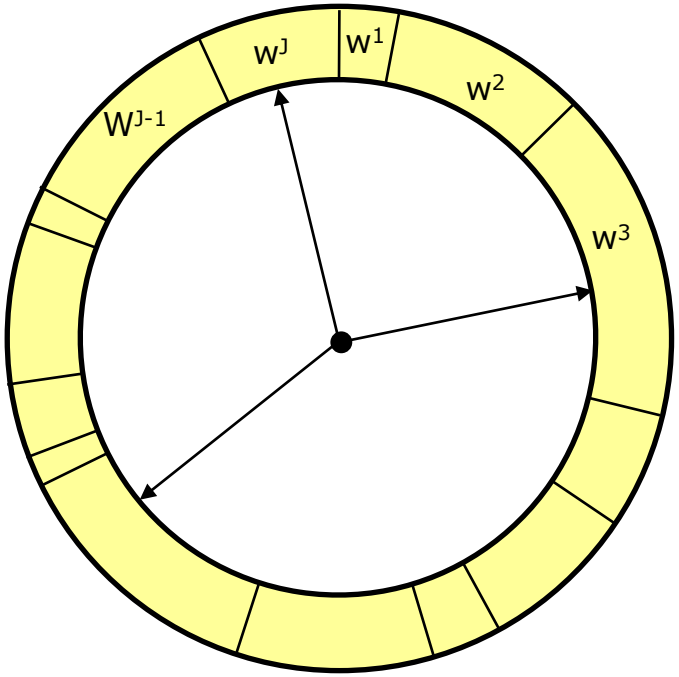


Courtesy: D. Fox

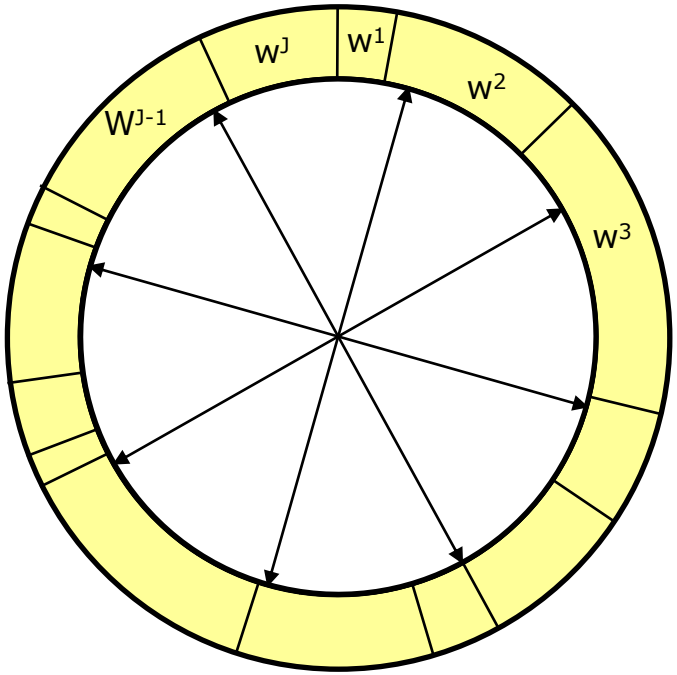
Resampling

- 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



- Roulette wheel
- Binary search
- O(J log J)



- Stochastic universal sampling
- Low variance
- O(J)

Resampling

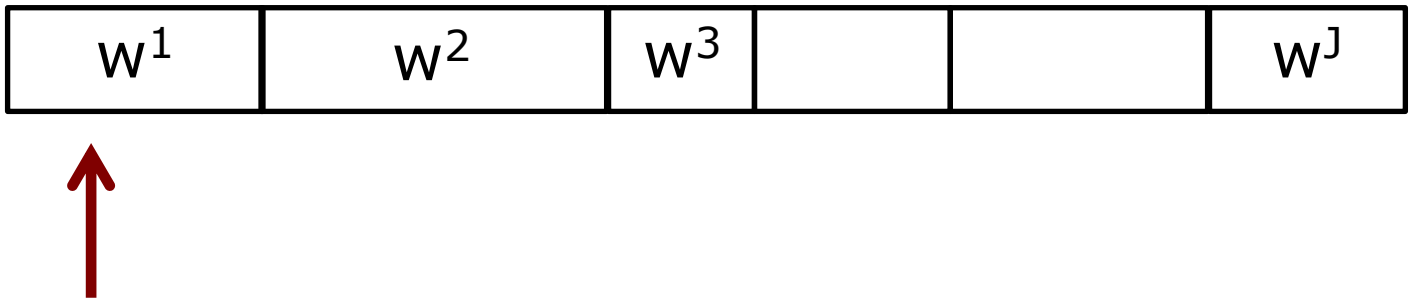
- Sampling with replacement
- Roulette wheel resampling is easy to understand
- ... but it is suboptimal in practice

Resampling

- Sampling with replacement
- Roulette wheel resampling is easy to understand
- … but it is suboptimal in practice

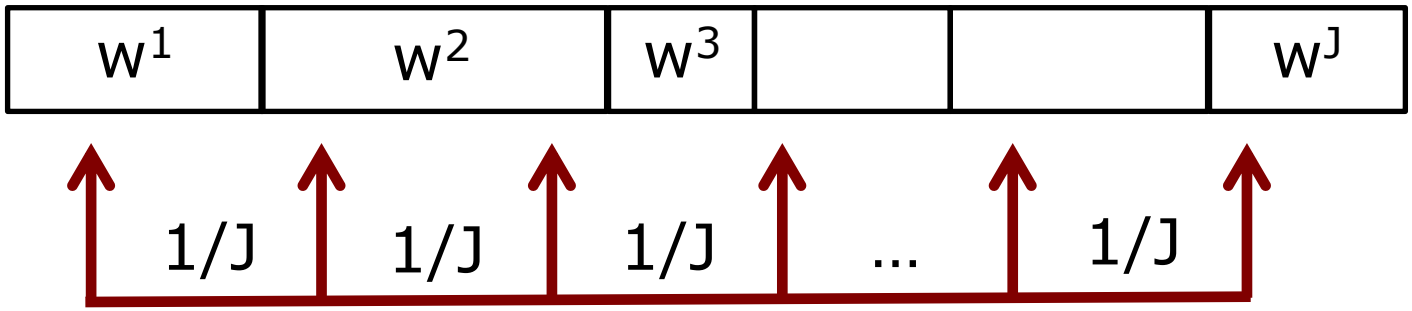
What happens if all sample have the same weight?

Low Variance Resampling Idea



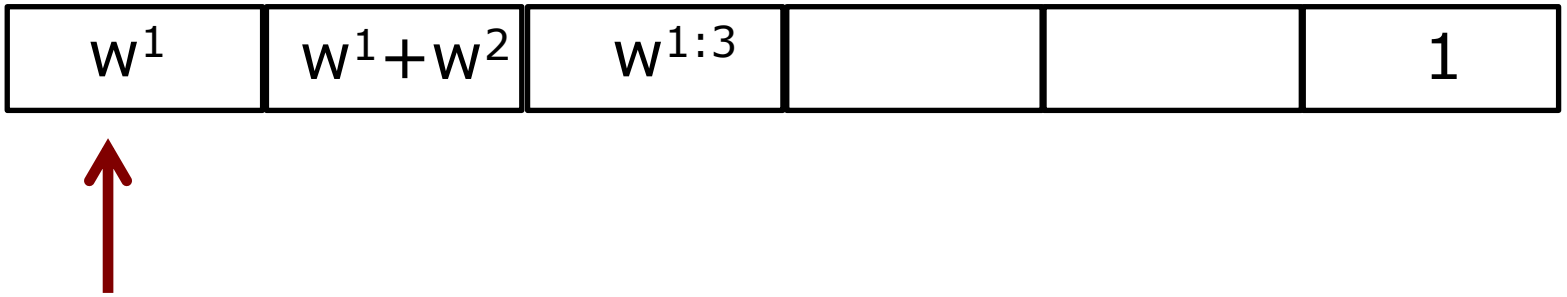
1. draw random number between 0 and 1/J

Low Variance Resampling Idea



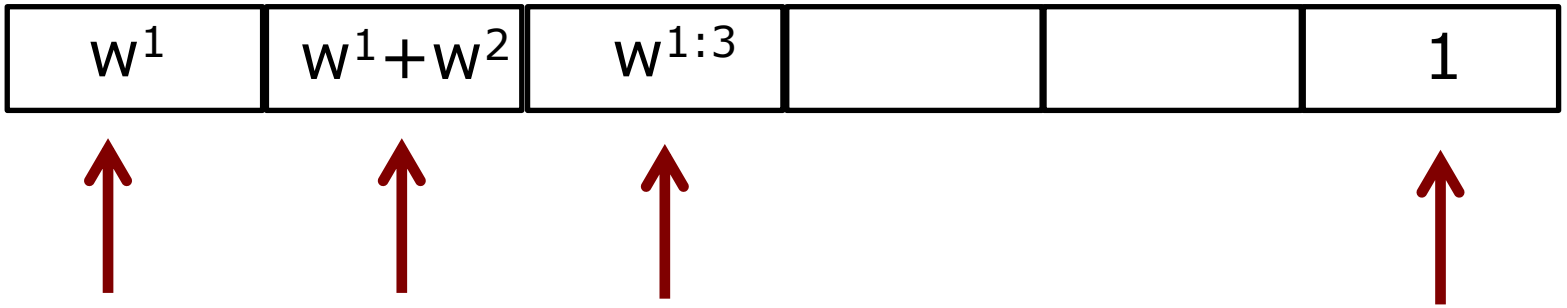
1. draw random number between 0 and 1/J
2. pick J-1 particles in 1/J steps
   (based on the cumulative weight)

Efficient Implementation



1. draw random number r between 0 and 1/J

Efficient Implementation



1. draw random number r between 0 and 1/J

2. for (j=1 .. J)
   U = r + j*1/J
   while (U > cum[i]) i++;
   pick_particle i

Low Variance Resampling

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

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

$r = \text{rand}(0; J^{-1})$

$c = w_t^{[1]}$

$i = 1$

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

  $U = r + (j-1)J^{-1}$

  while $U > c$

    $i = i + 1$

    $c = c + w_t^{[i]}$

  endwhile

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

endfor

return $\bar{\mathcal{X}}_t$

Low Variance Resampling

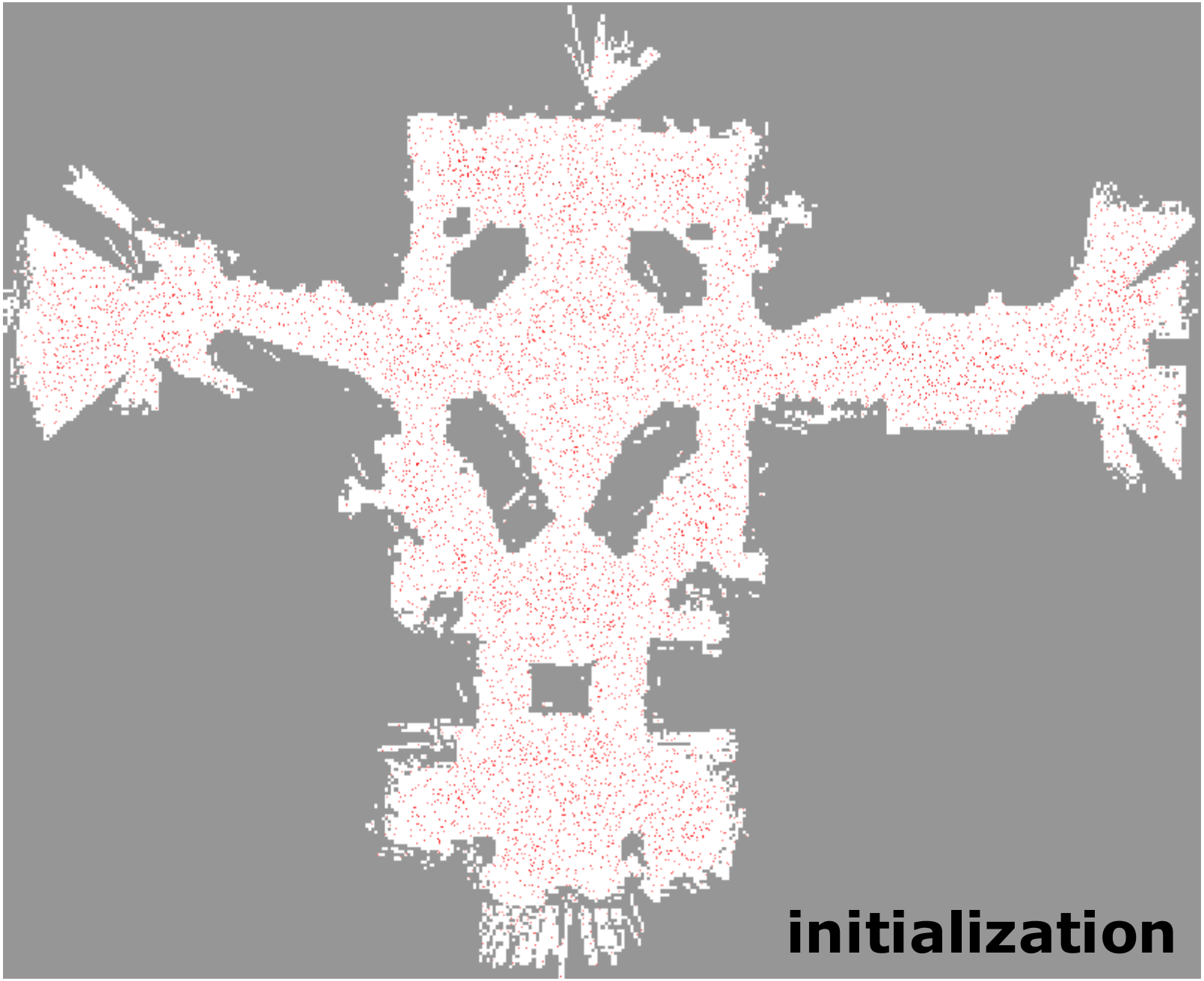
- Performs resampling that keeps the samples in case of same weights
- Is faster than roulette wheel resampling: O(J) vs. O(J log J)

Always use low variance resampling!

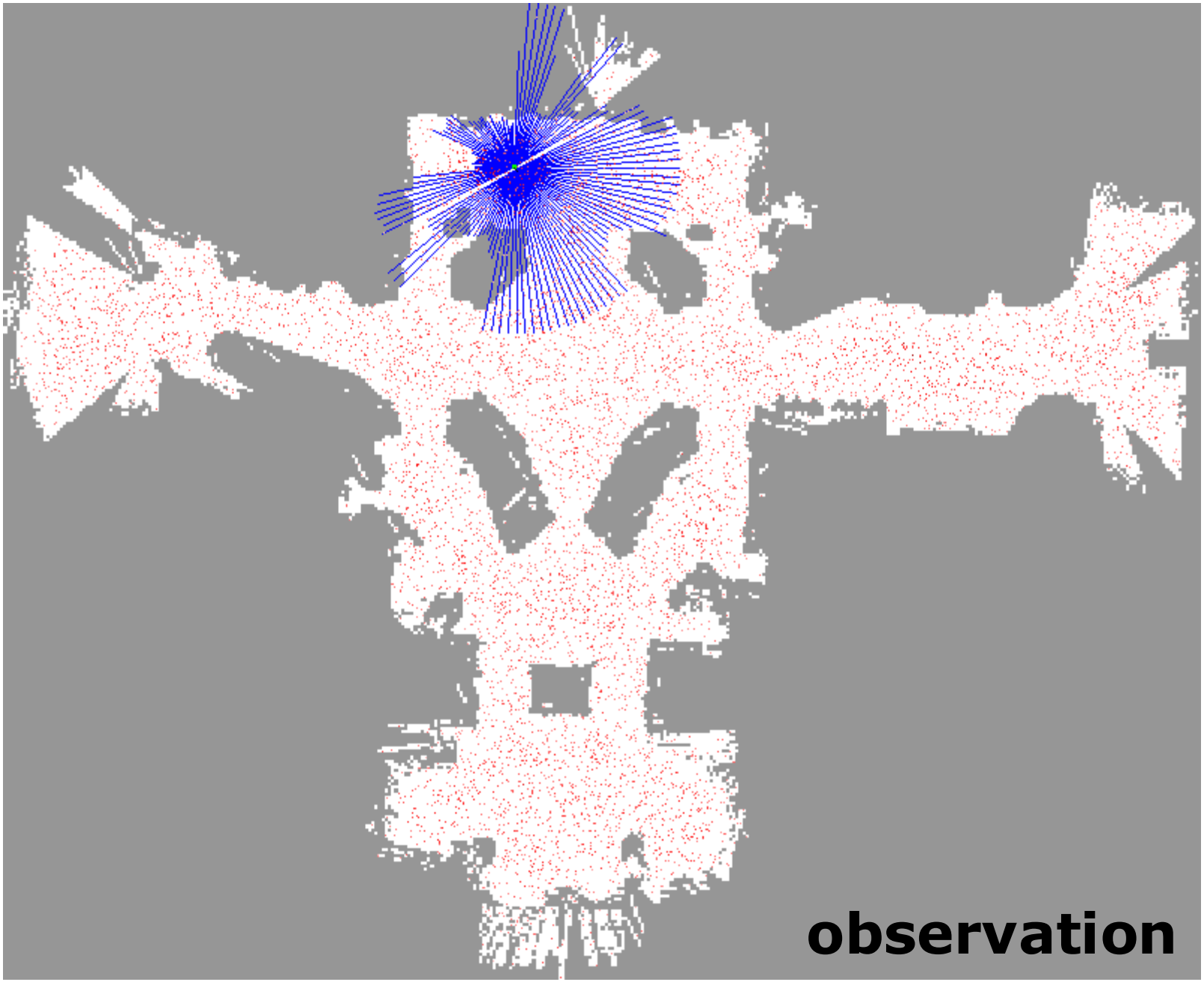
MCL in Action

Rhino, Minerva, Albert (~1998)

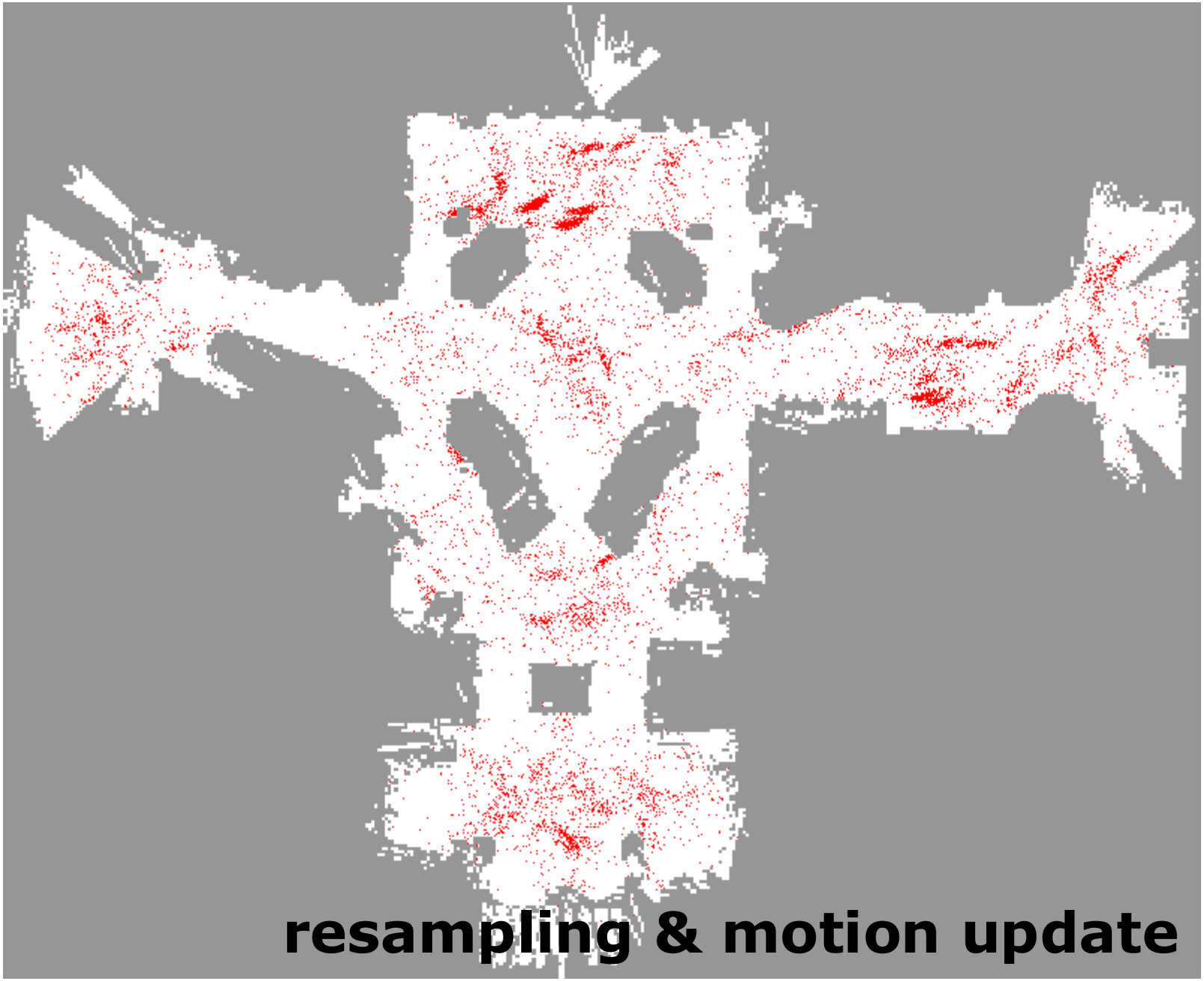
Courtesy: Thrun, Burgard, Fox



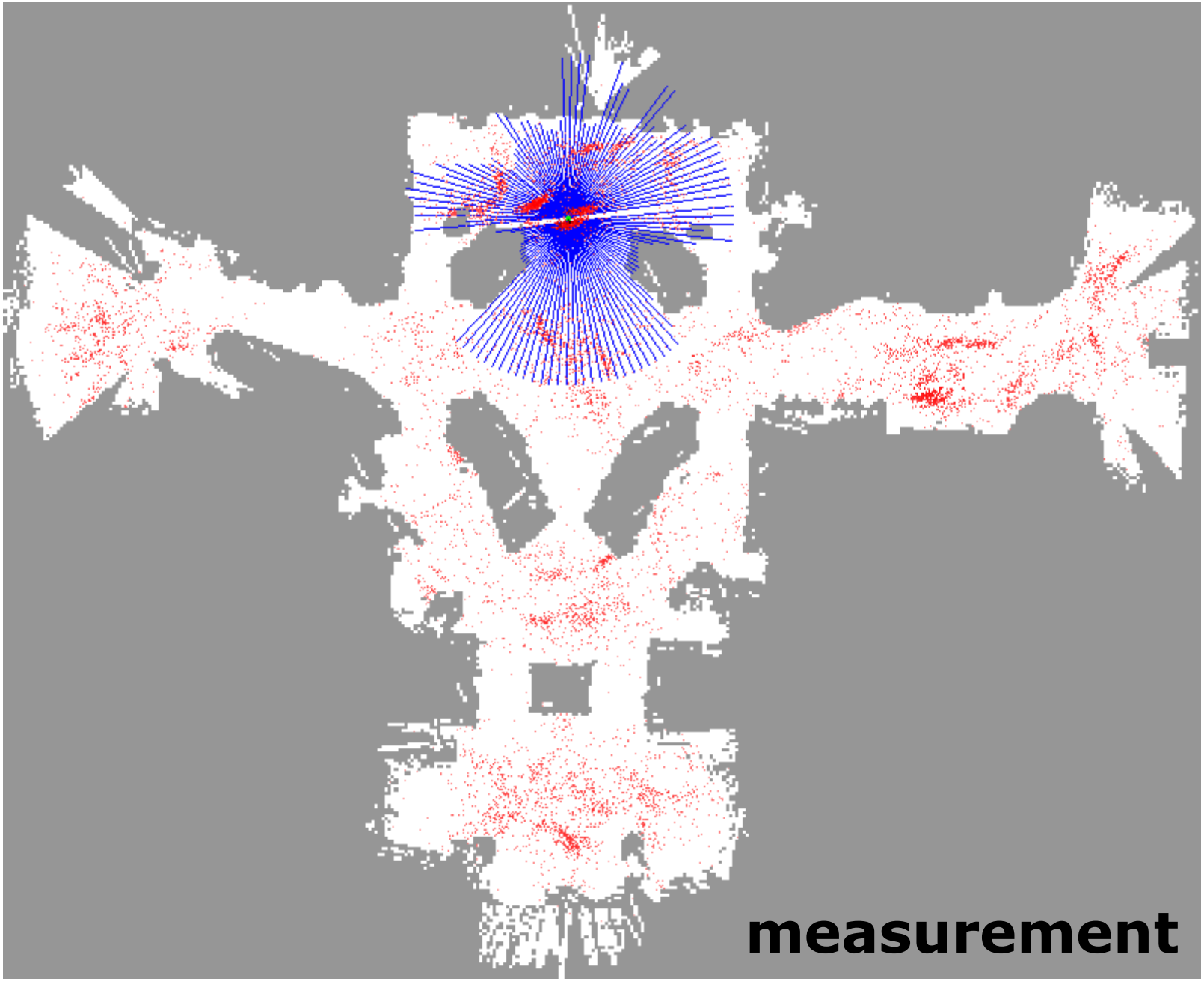
Courtesy: Thrun, Burgard, Fox



Courtesy: Thrun, Burgard, Fox

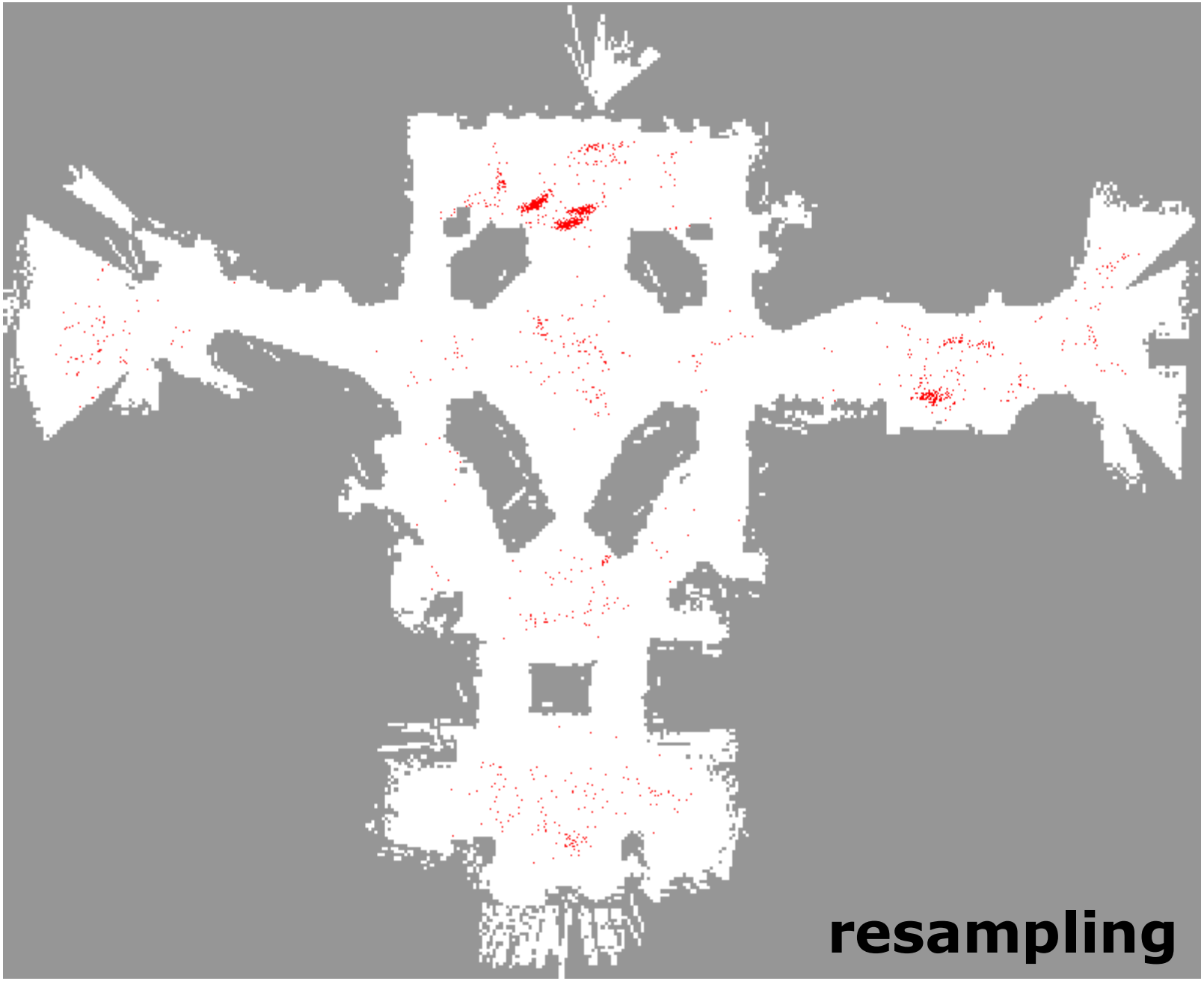


Courtesy: Thrun, Burgard, Fox

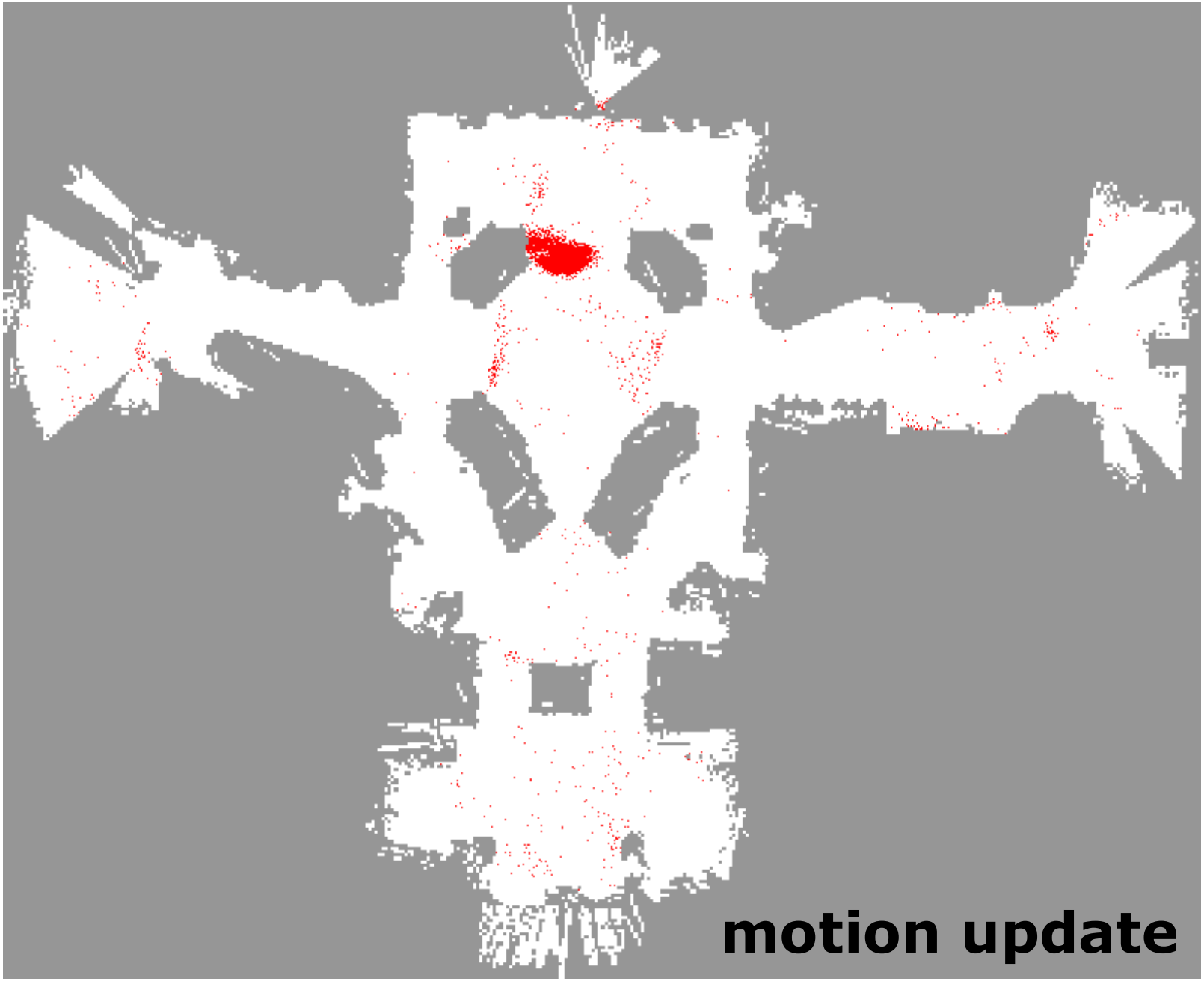


Courtesy: Thrun, Burgard, Fox

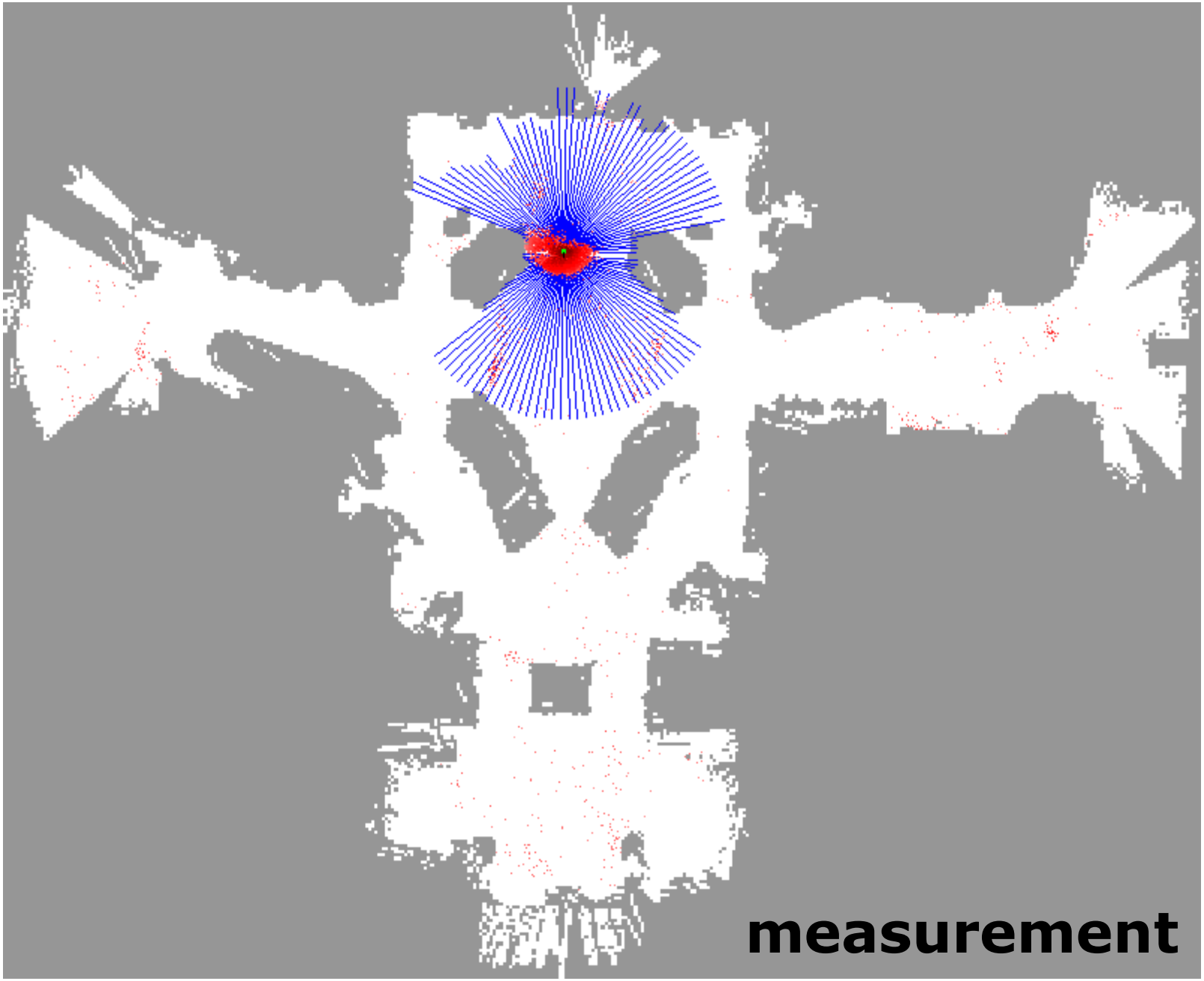
Courtesy: Thrun, Burgard, Fox



Courtesy: Thrun, Burgard, Fox

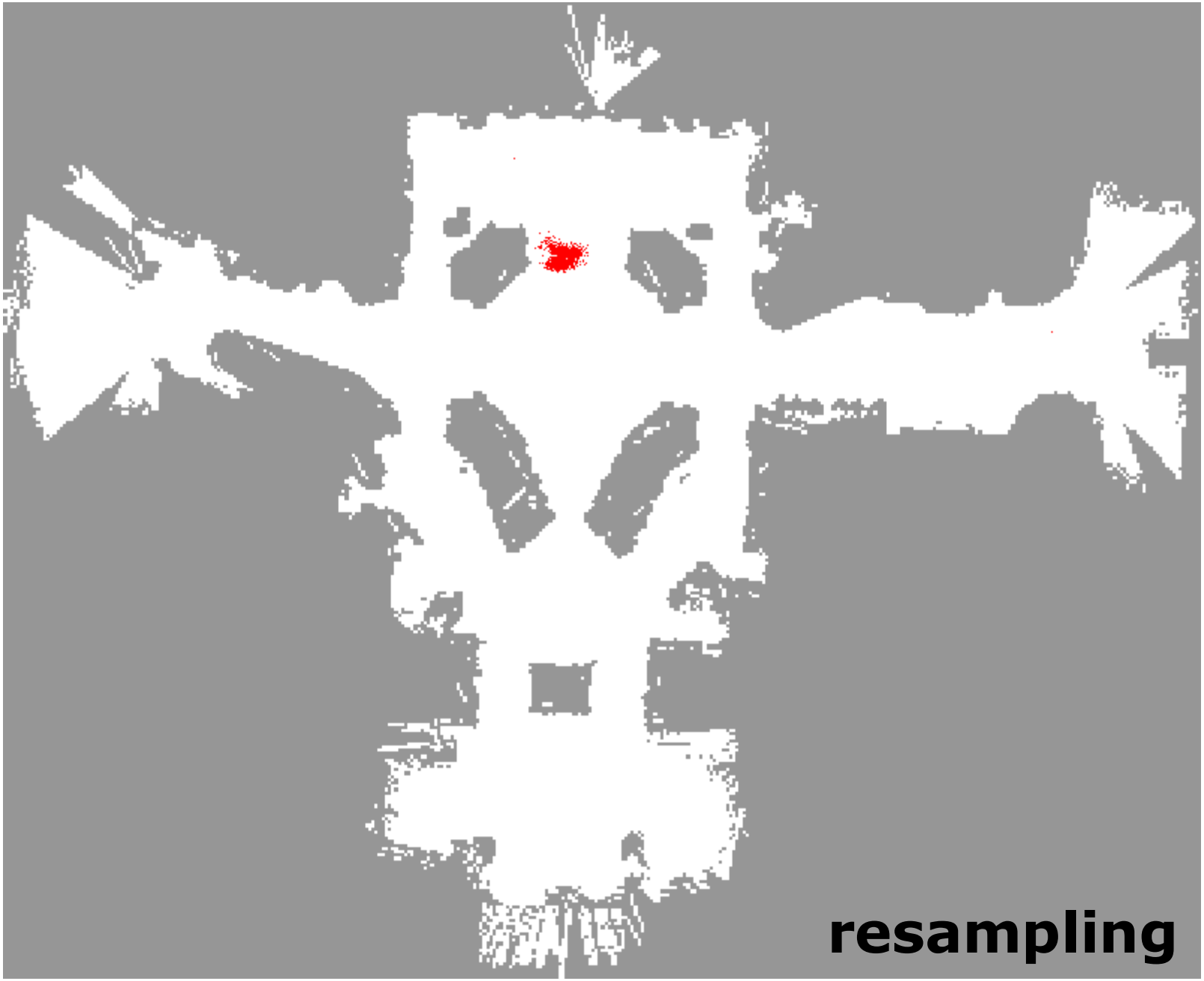


Courtesy: Thrun, Burgard, Fox

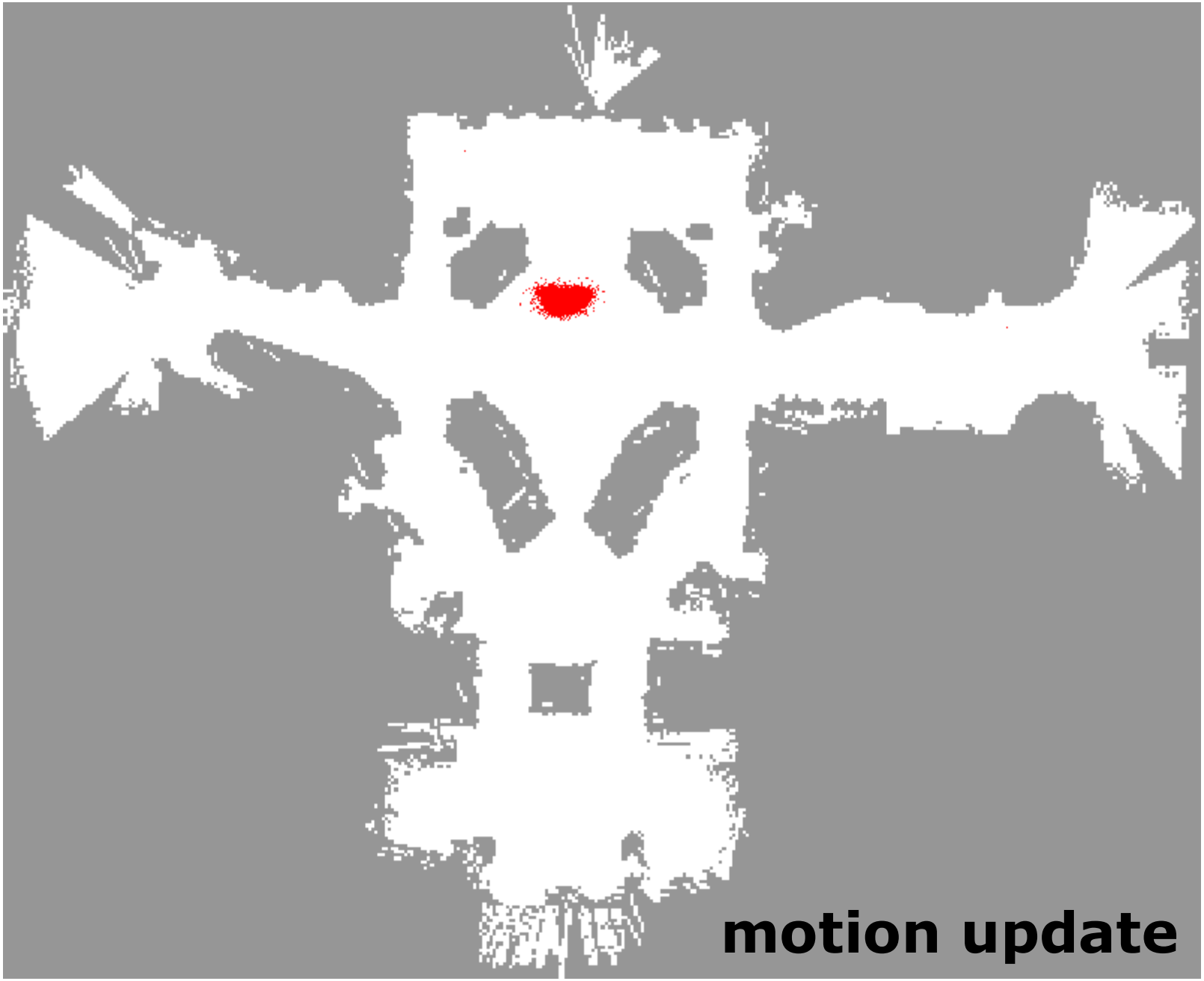


Courtesy: Thrun, Burgard, Fox

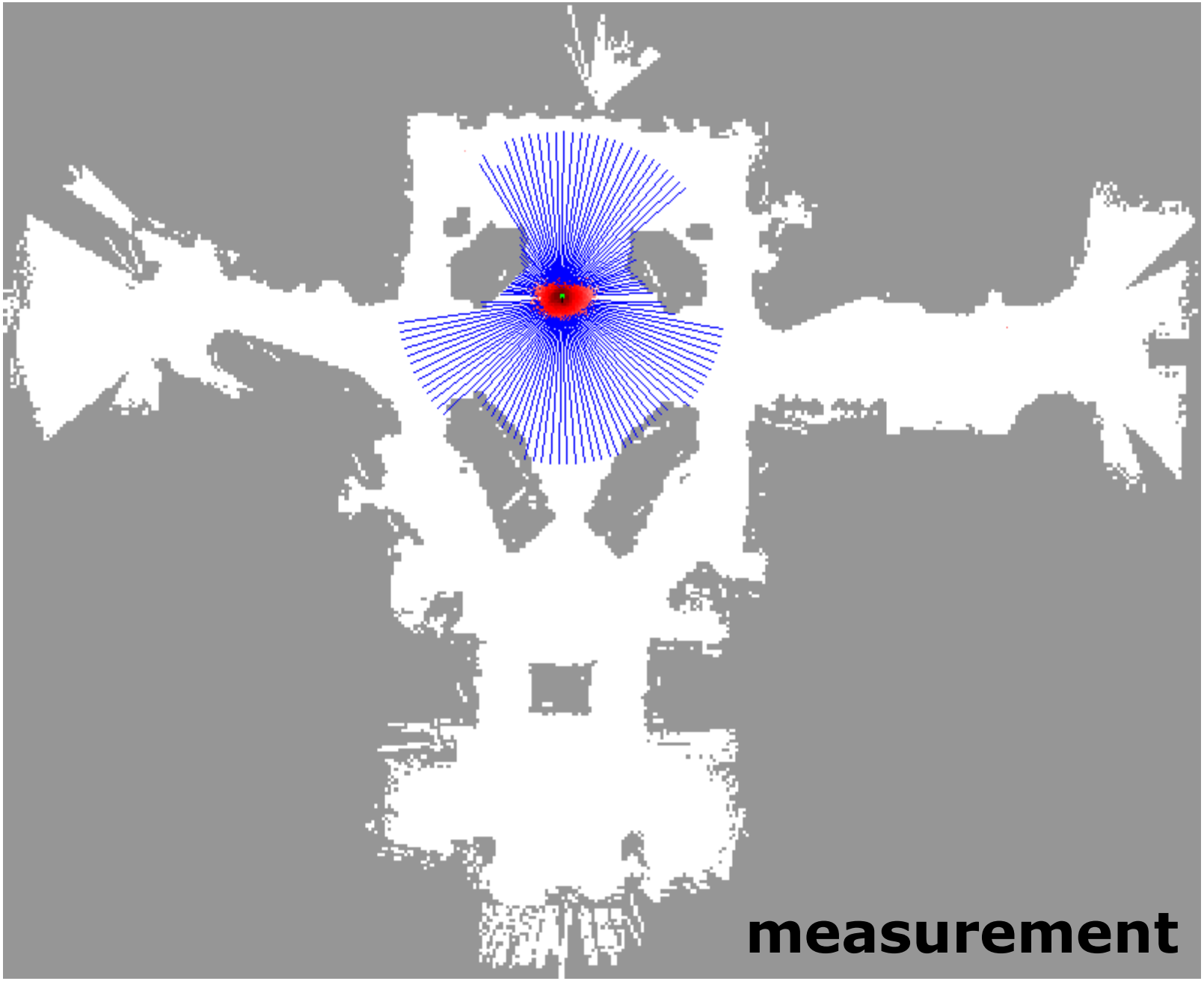
Courtesy: Thrun, Burgard, Fox



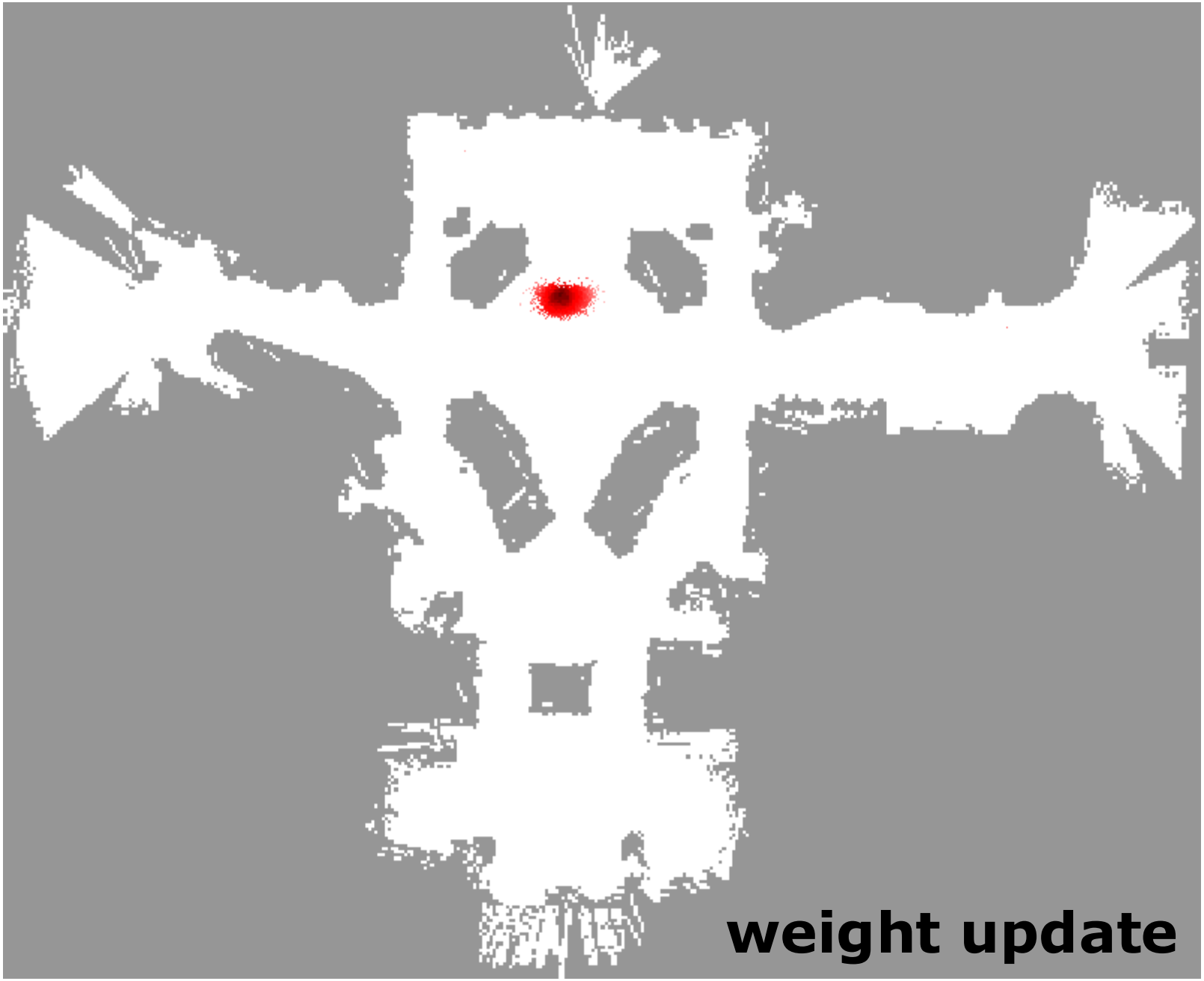
Courtesy: Thrun, Burgard, Fox



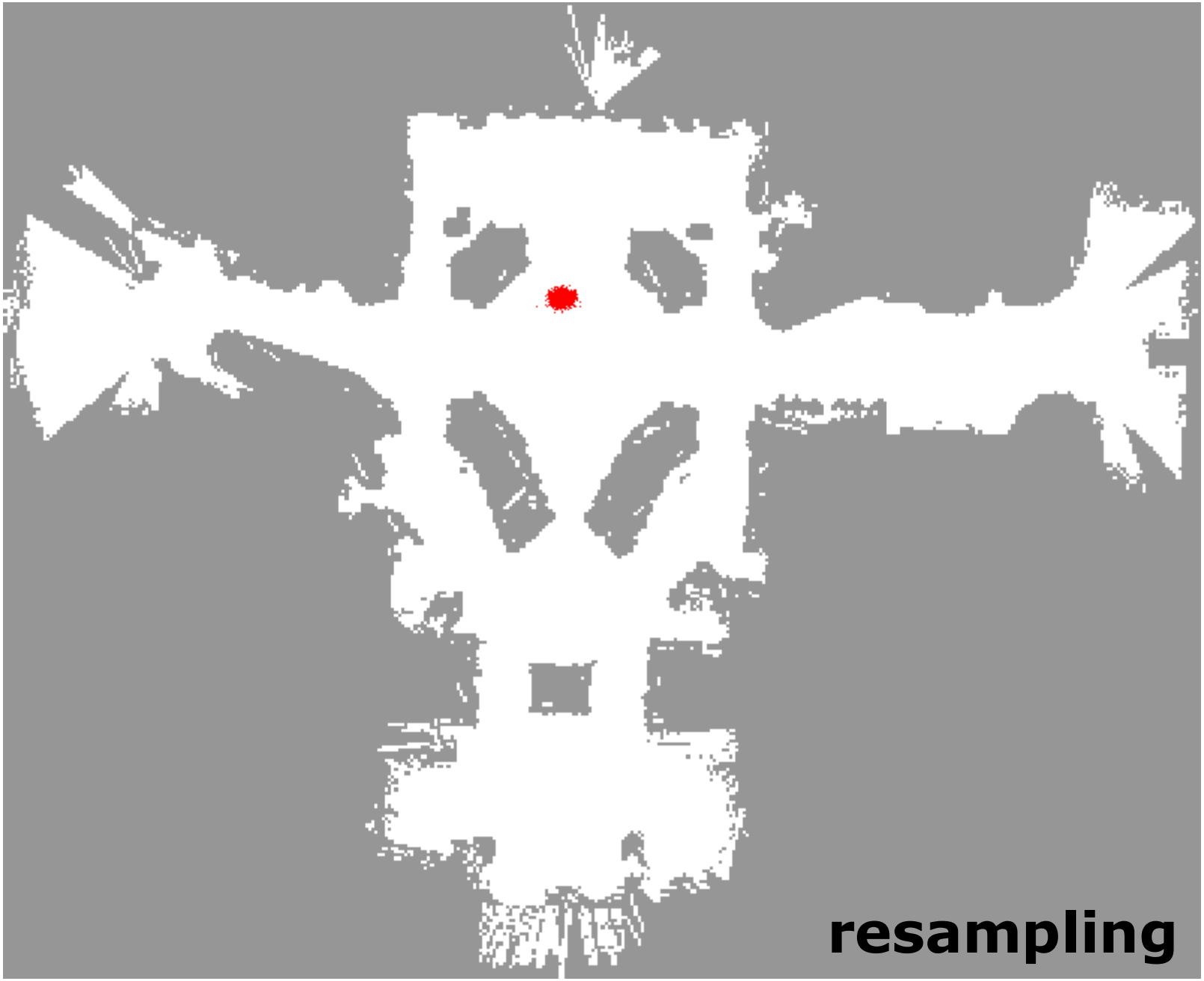
Courtesy: Thrun, Burgard, Fox



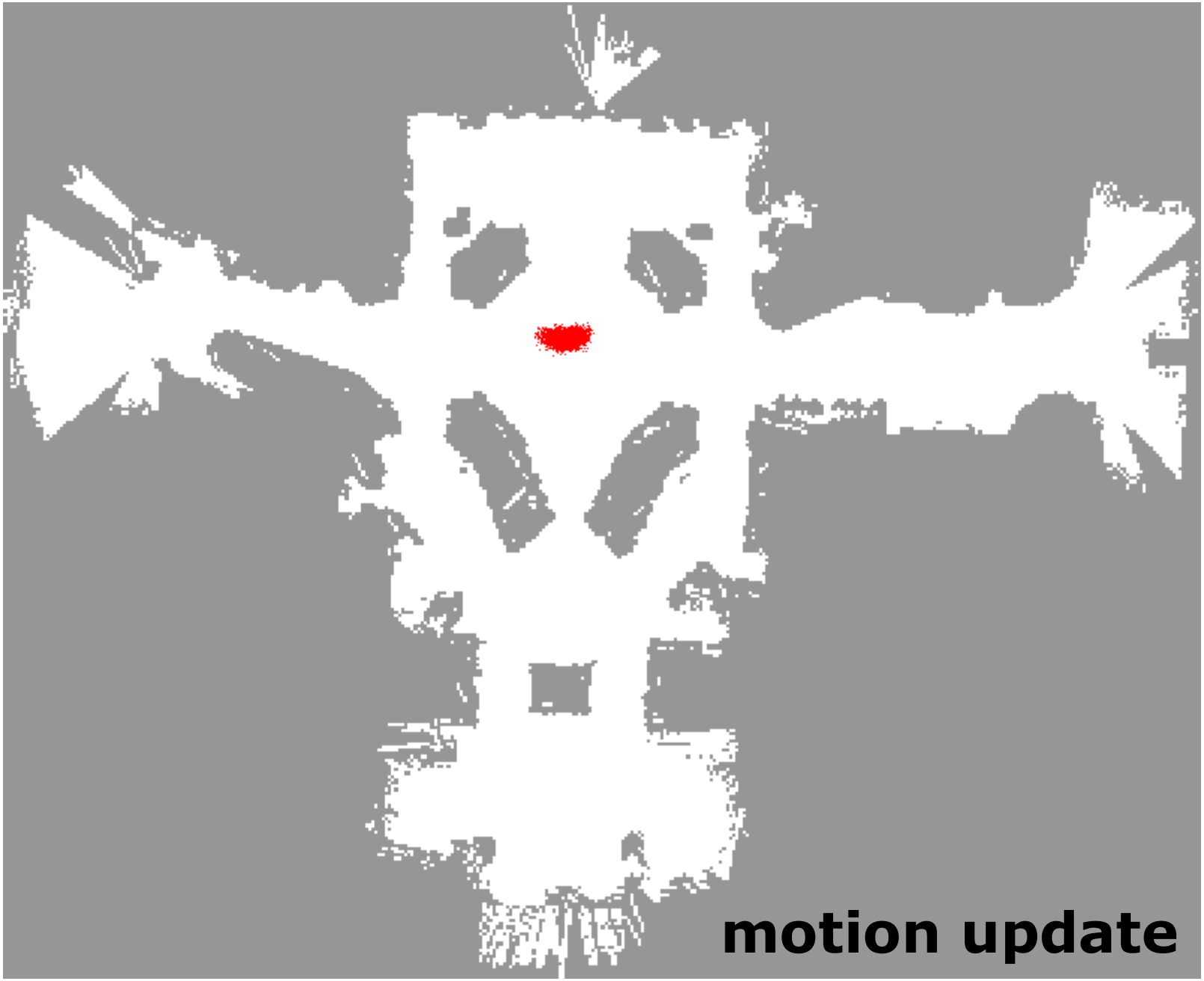
Courtesy: Thrun, Burgard, Fox



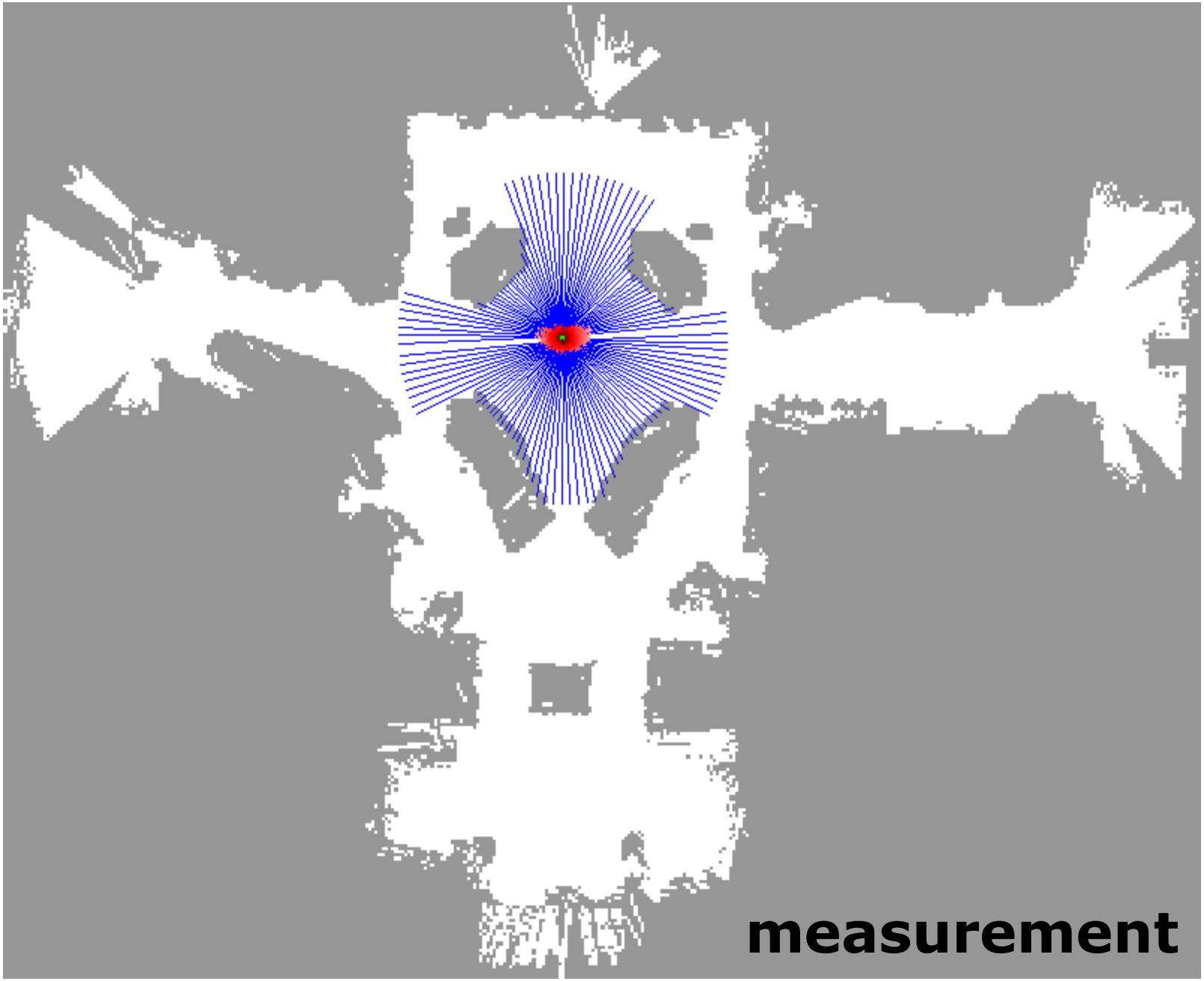
Courtesy: Thrun, Burgard, Fox



Courtesy: Thrun, Burgard, Fox

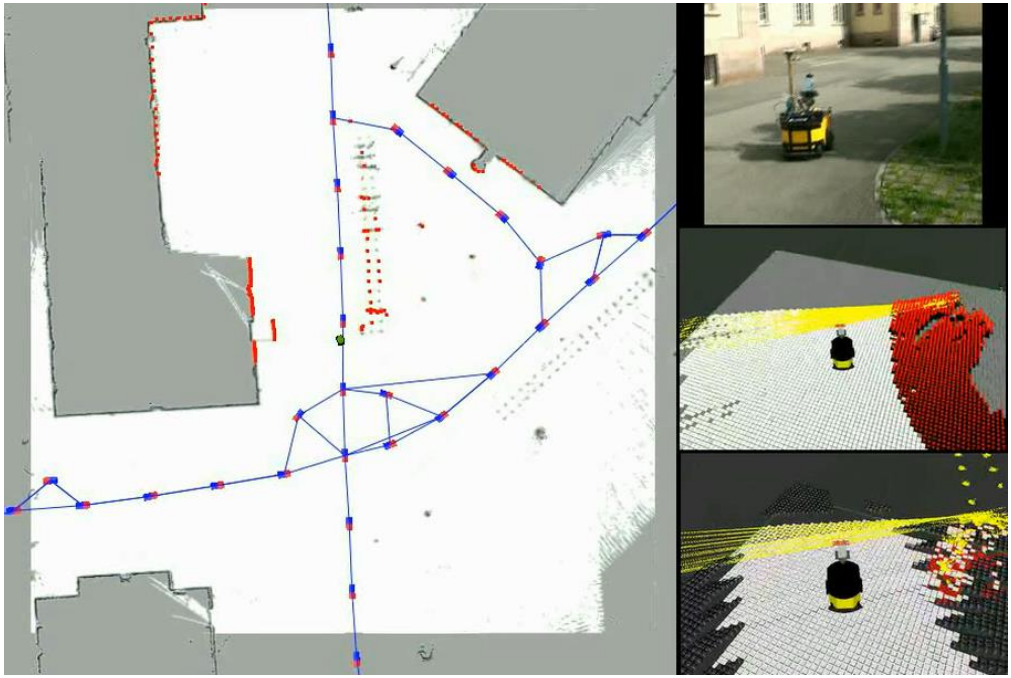


Courtesy: Thrun, Burgard, Fox



Courtesy: Thrun, Burgard, Fox

MCL: Two Examples



Cons

- Problematic in high-dimensional spaces
- Problematic in situations with high uncertainty
- Particle depletion problem

Pros

- Handles non-Gaussian distributions
- Works well in low-dimensional spaces
- Can elegantly handle data association ambiguities
- Can easily incorporate different sensing modalities
- Comparably robust
- Easy to implement

Variants

There exists several variants of the particle filter algorithm

- Real-Time particle filters
  for dealing with sensor data that is obtained at different frame rates
- Delayed state particle filters
  for dealing with substantial delays in sensor data streams
- Rao-Blackwellized particle filters
  for dealing with high-dimensional state spaces
- …

Summary – Particle Filters

- Particle filters are non-parametric, recursive Bayes filters
- Posterior is represented by a set of weighted samples
- Proposal to draw the samples for t+1
- Weight to account for the differences between the proposal and the target
- **The art is to design appropriate motion and sensor models**

Summary – PF Localization

- Particles are propagated according to the motion model
- They are weighted according to the likelihood of the observation
- Called: Monte-Carlo localization (MCL)
- MCL is the gold standard for indoor mobile robot localization today

Literature

On the particle filter

- Thrun et al. “Probabilistic Robotics”, Chapter 3

On Monte Carlo Localization

- Thrun et al. “Probabilistic Robotics”, Chapter 8.3

Rehearsal:

On motion and observation models

- Thrun et al. “Probabilistic Robotics”, Chapters 5 & 6