Cognitive Robotics

06. Non-Parametric Filters: Discrete Filter, Particle Filter, Monte Carlo Localization

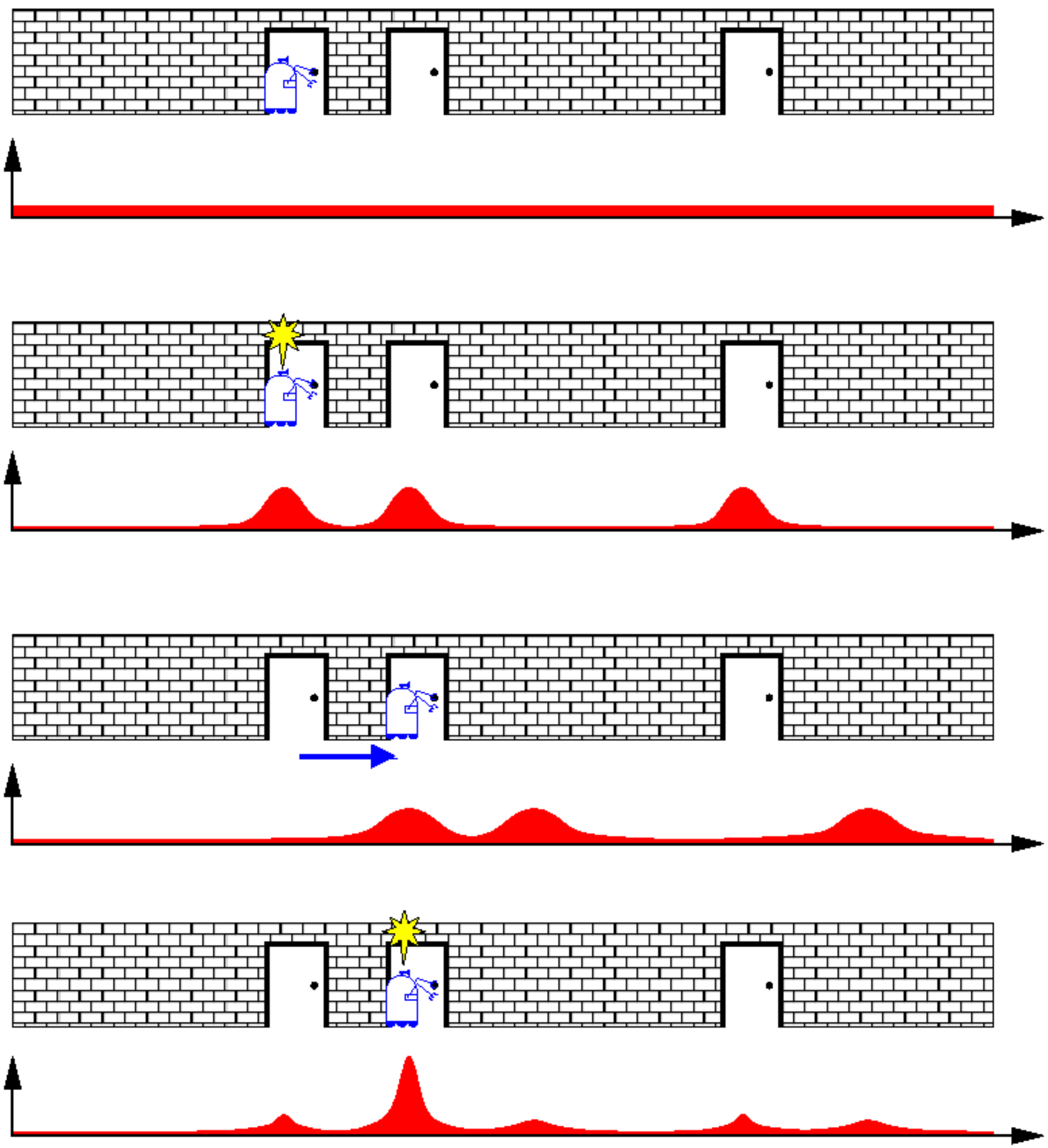
AbdElMoniem Bayoumi, PhD

Spring 2022

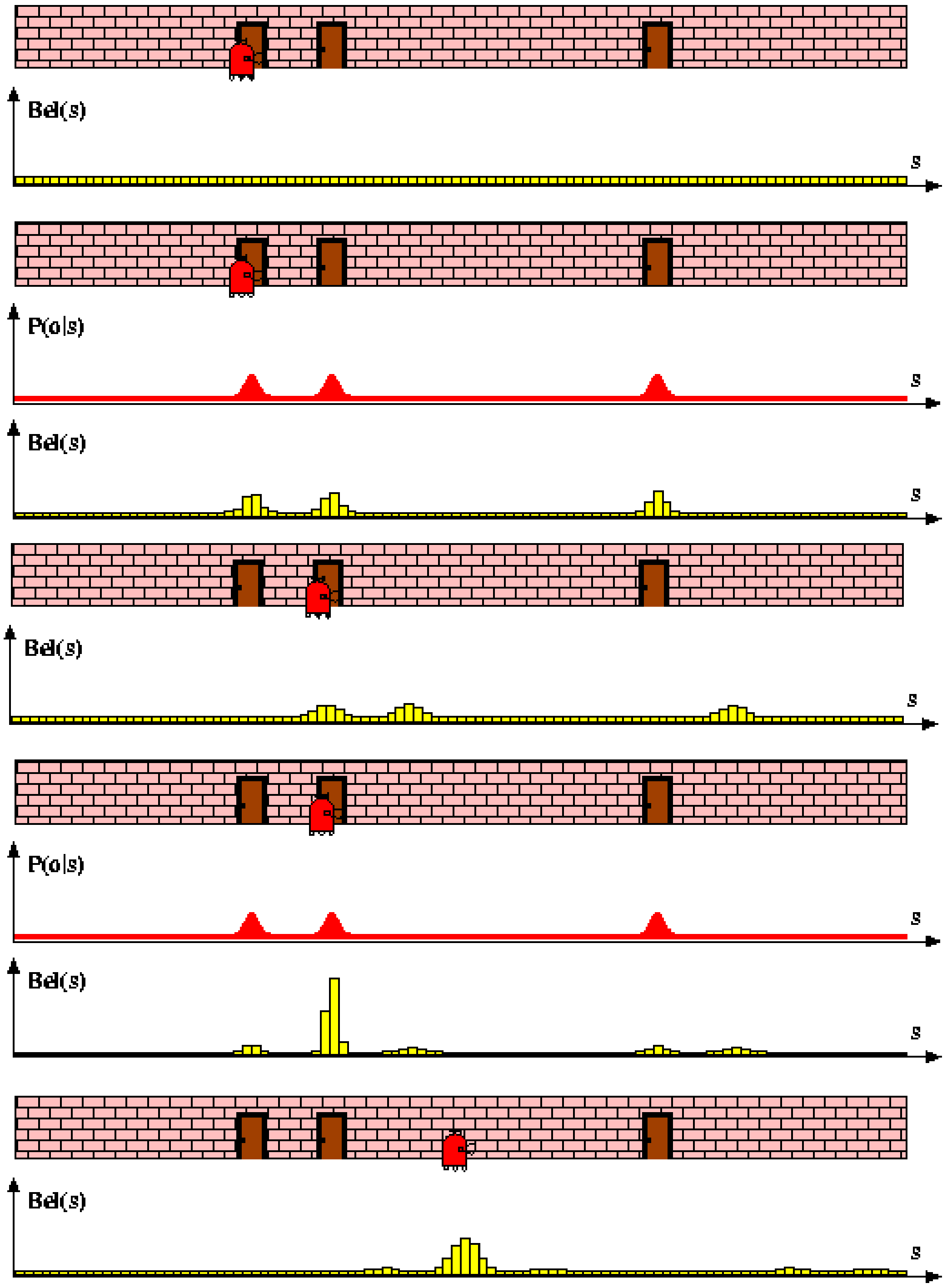
Acknowledgment

- These slides have been created by Wolfram Burgard, Dieter Fox, Cyrill Stachniss and Maren Bennewitz

$$Bel(x \mid z, u) \;=\; \alpha\, p(z \mid x) \int_{x'} p(x \mid u, x')\; Bel(x')dx'$$



Discrete Filter: Piecewise Constant



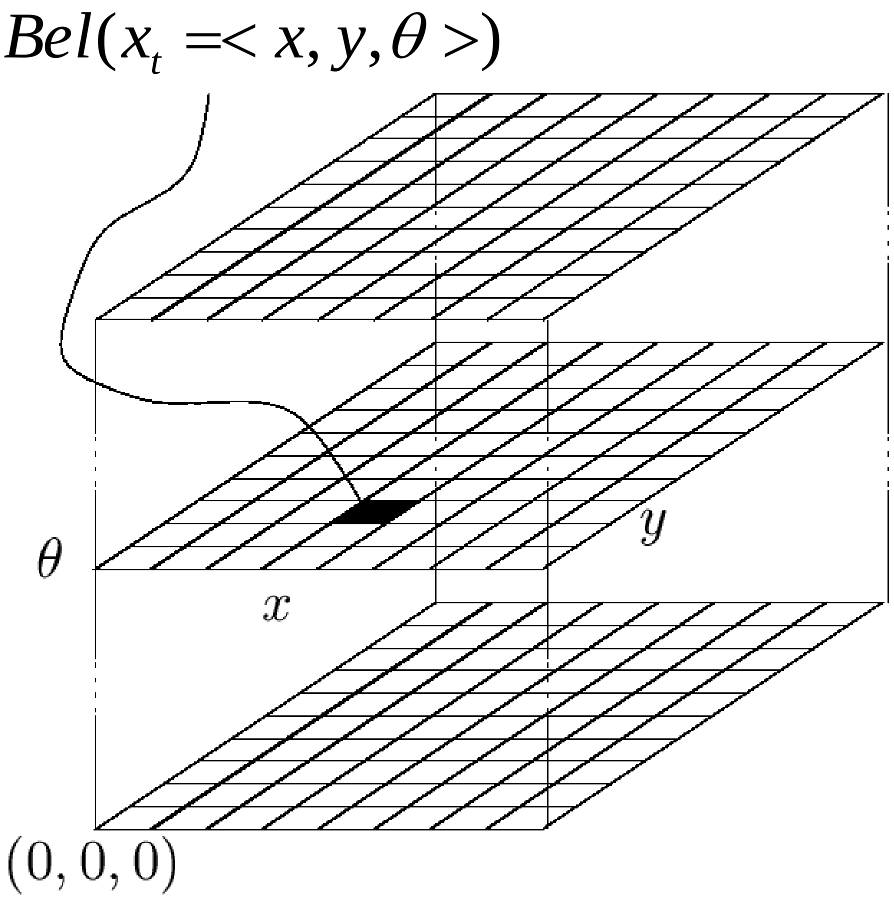
Bayes Filter Algorithm

1. Algorithm **Bayes_filter**(*Bel(x),d*):
2. $\eta = 0$
3. If *d* is a **perceptual** data item *z* then
4. For all *x* do
5. $Bel'(x) = P(z \mid x) Bel(x)$
6. $\eta = \eta + Bel'(x)$
7. For all *x* do
8. $Bel'(x) = \eta^{-1} Bel'(x)$
9. Else if *d* is an **action** data item *u* then
10. For all *x* do
11. $Bel'(x) = \sum_{x'} P(x \mid u, x')\, Bel(x')$
12. Return *Bel'(x)*

sum over all discrete states

motion model, Ch. 4

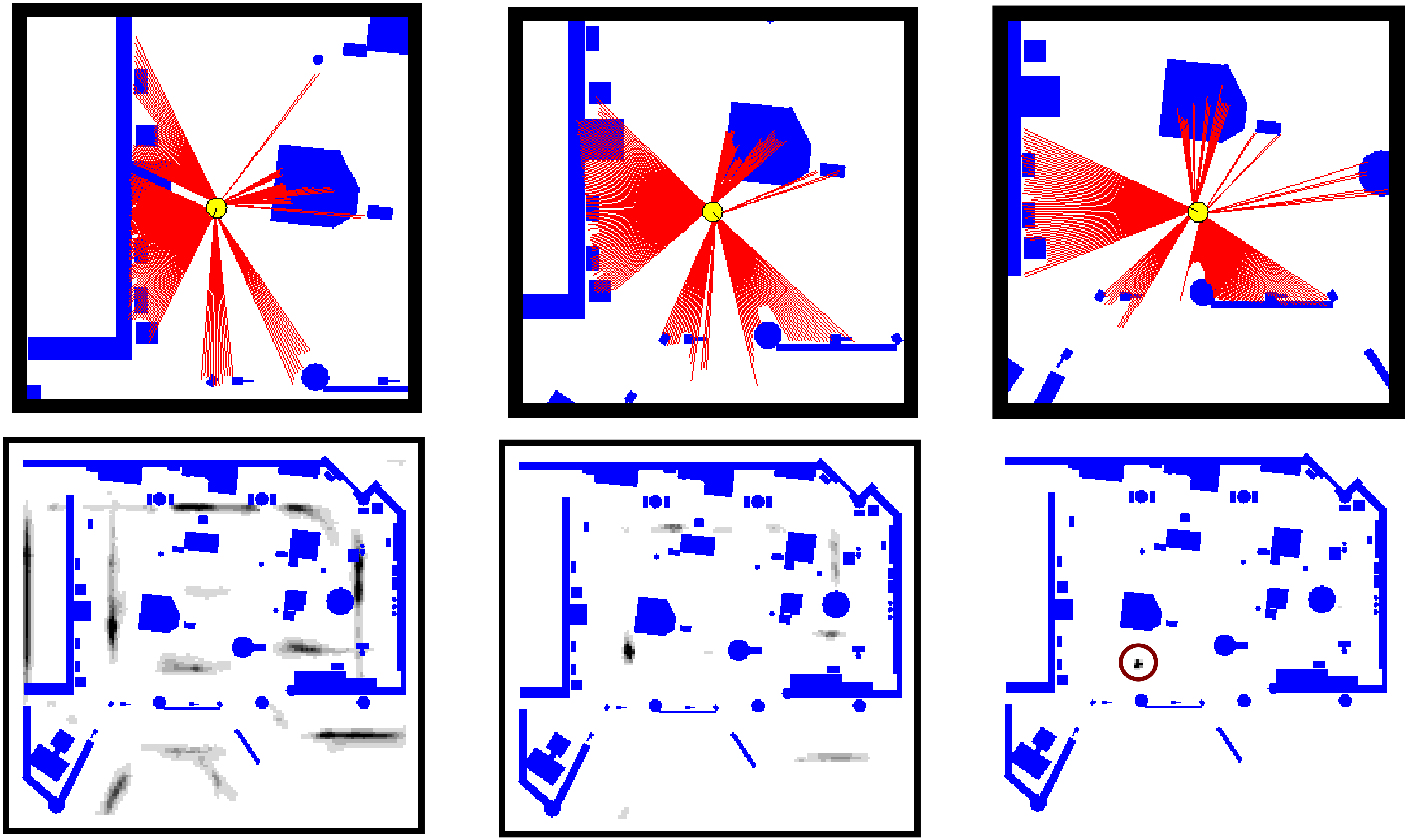
Piecewise Constant Representation



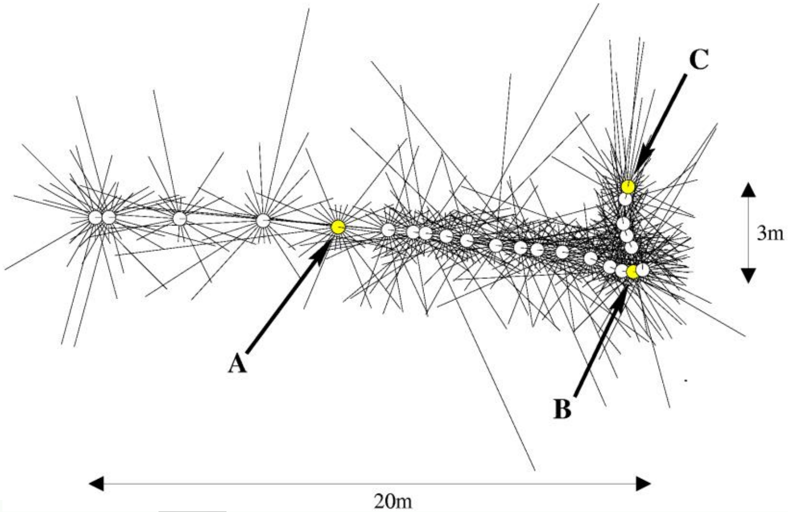
Implementation

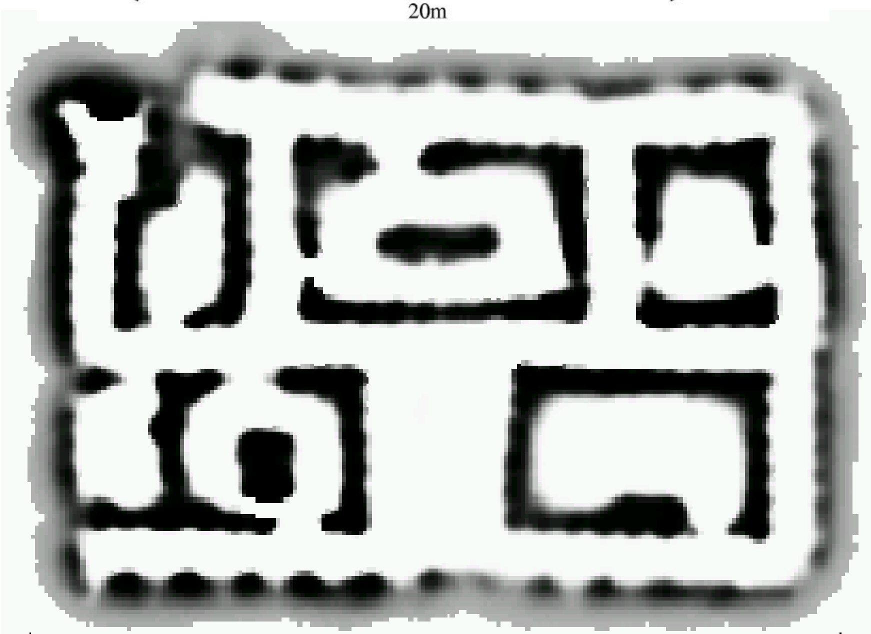
- To update the belief, one has to iterate over all cells of the grid
- When the belief is peaked, one wants to avoid updating irrelevant aspects of the state space
- Monitor whether the robot is de-localized or not
- Consider the likelihood of the observation in the relevant components of the state space
- Assume a bounded Gaussian for the motion uncertainty

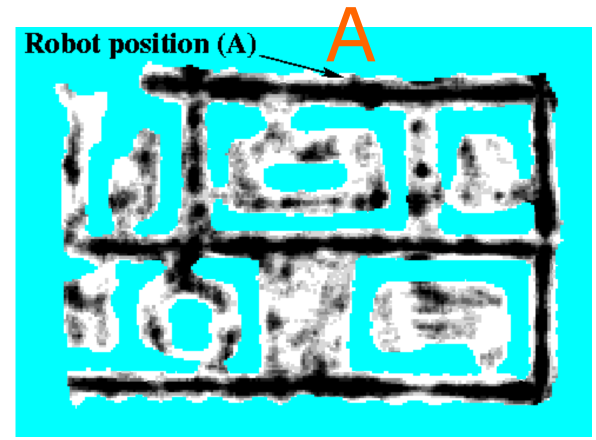
Grid-Based Localization

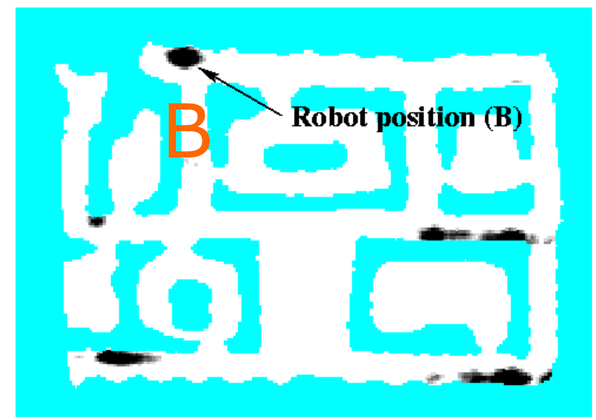


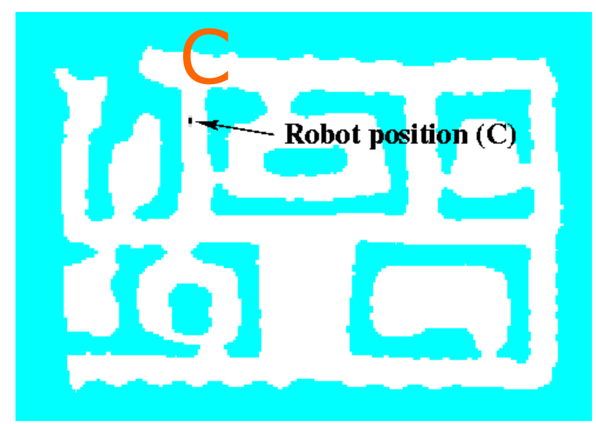
Sonars and Occupancy Grid Map











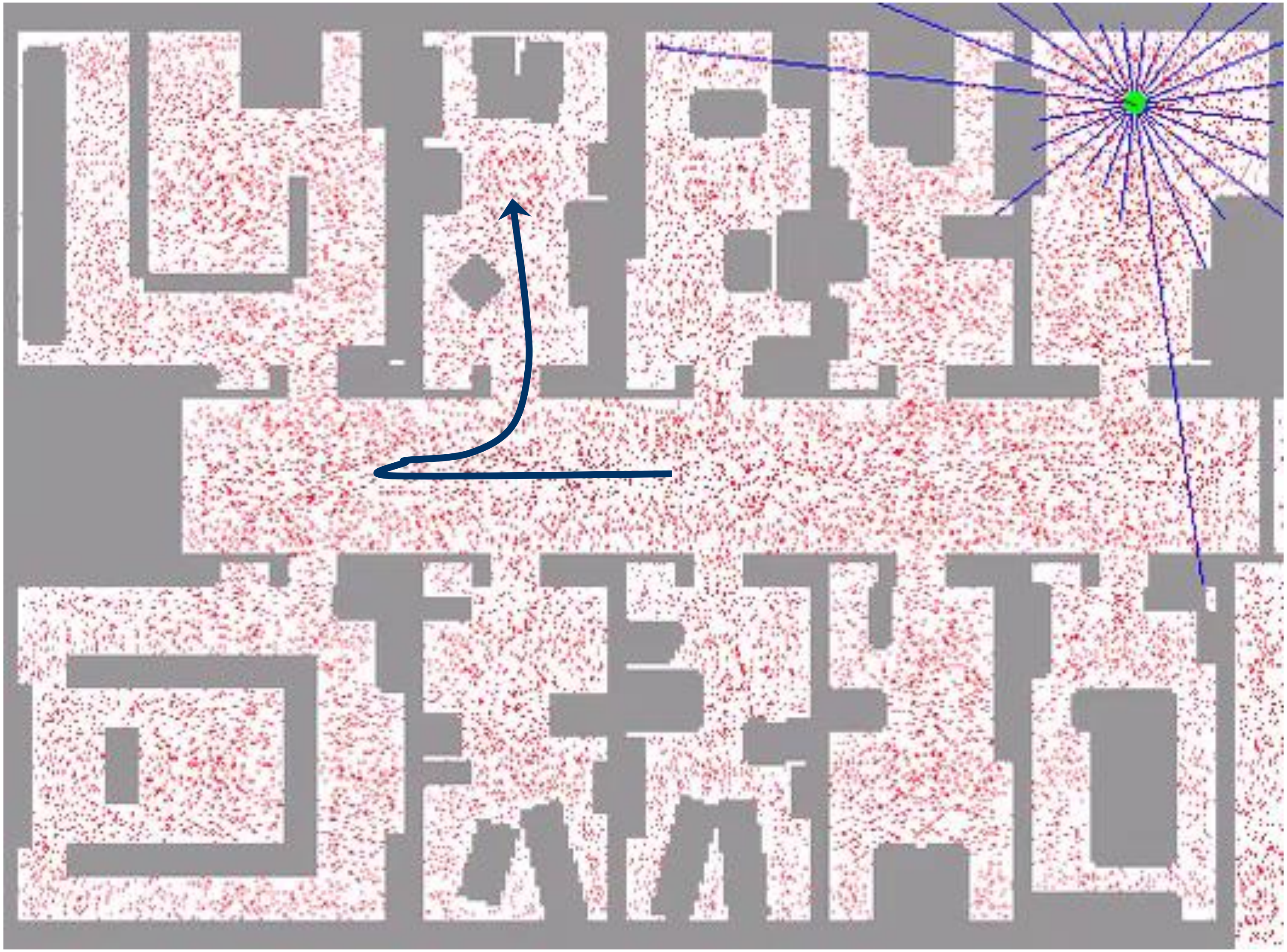
Summary: Discrete Filters

- Discrete filters are an alternative way for implementing the Bayes Filter
- Histograms for representing the density
- Can represent multi-modal beliefs and recover from localization errors
- Huge memory and processing requirements
- Accuracy depends on the resolution of the grid
- In practice: approximations needed

Motivation: Particle Filter

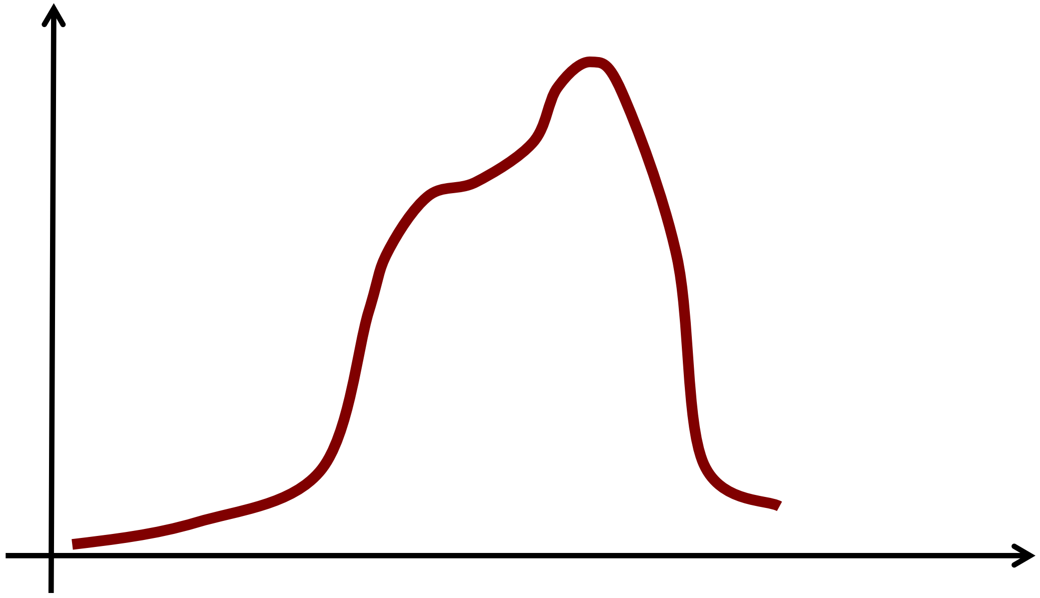
- Discrete filter
  - High memory complexity
  - In general: fixed resolution
- Particle filters are a way to **efficiently** represent **non-Gaussian distributions**
- Basic principle
  - Set of state hypotheses (“particles”)
  - Survival-of-the-fittest

Example:
Sample-Based Localization (Sonar)



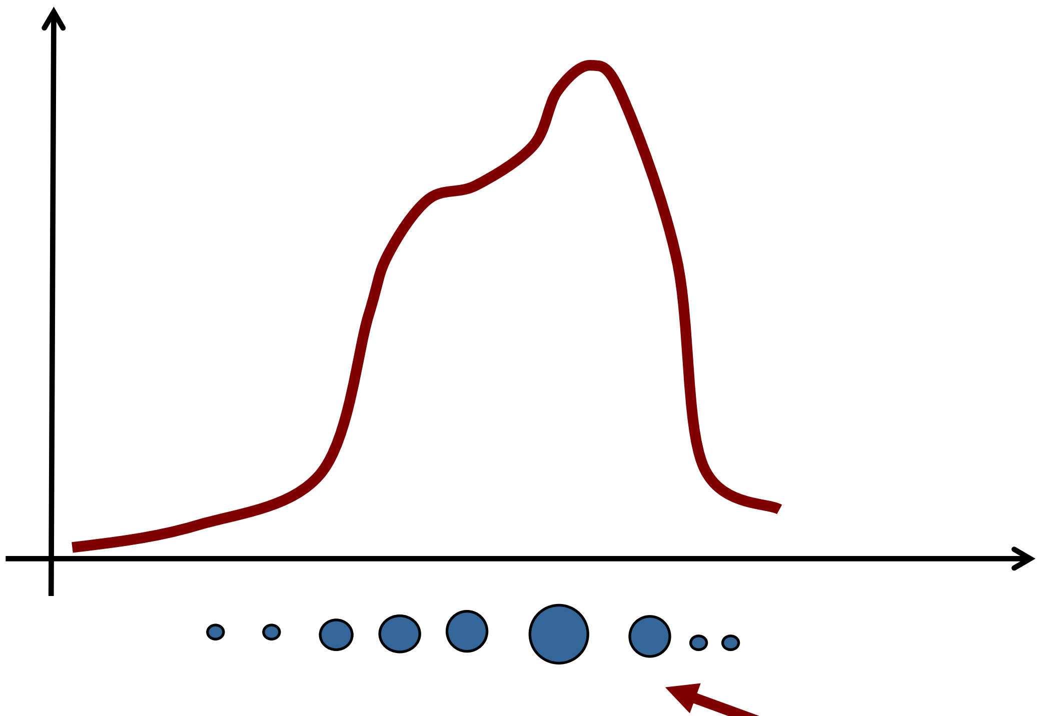
Motivation

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



Key Idea: Samples

Use **a set of weighted samples** to represent arbitrary distributions



samples

Particle Set

- Set of weighted samples

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

state hypothesis

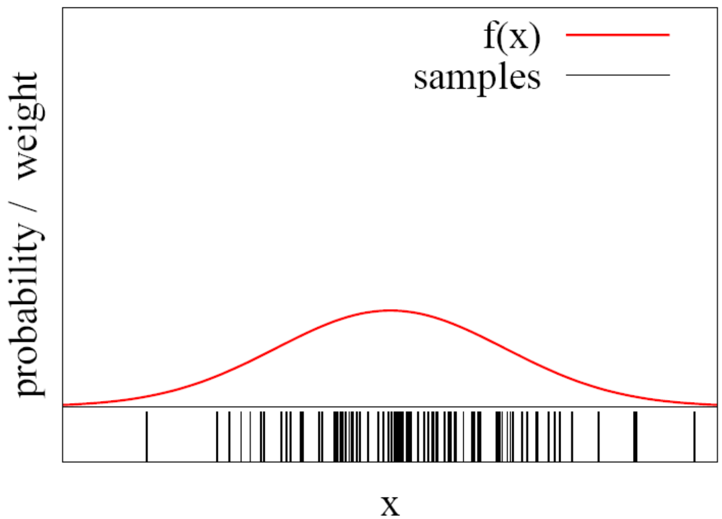
importance weight

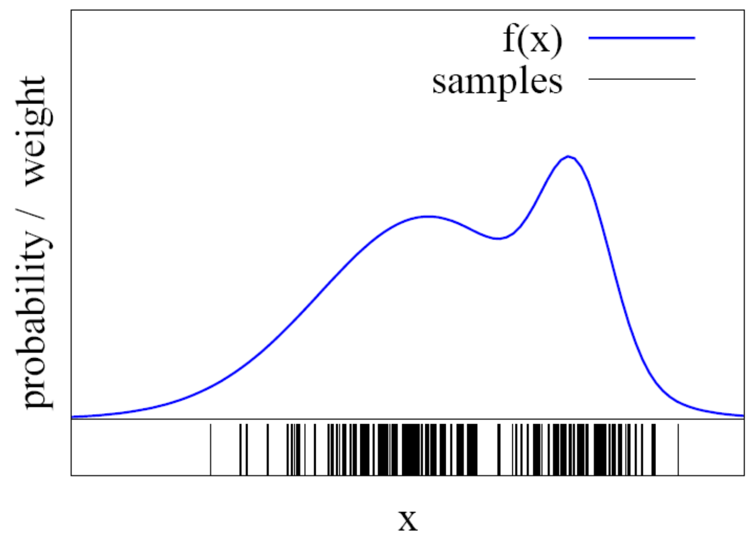
- The samples represent the posterior

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

Particles for Approximation

- Particles for function approximation



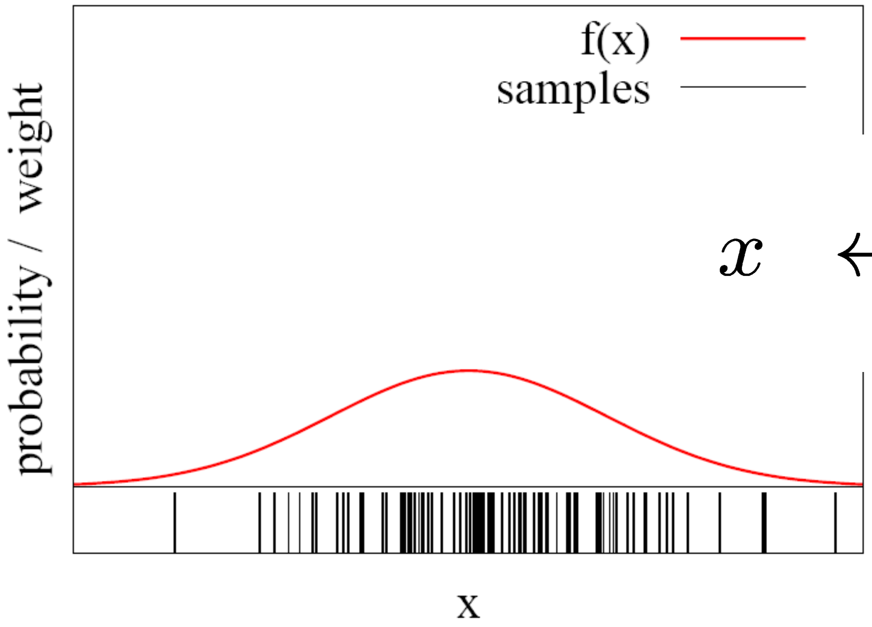


- The more particles fall into a region, the higher the probability of the region

How to obtain such samples?

Closed Form Sampling is Only Possible for Few Distributions

- Example: Gaussian



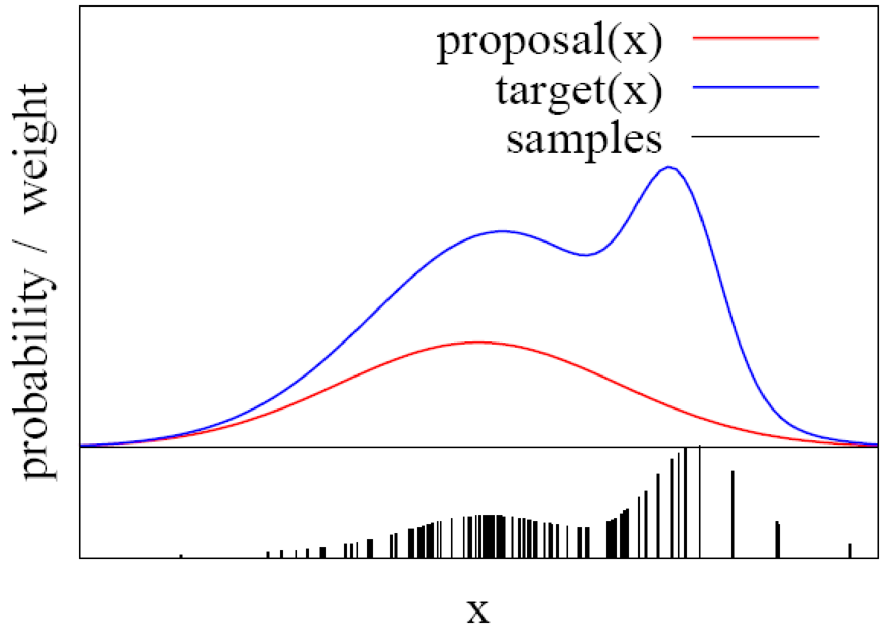
$$x \leftarrow \frac{1}{2}\sum_{i=1}^{12} \mathrm{rand}(-\sigma, \sigma)$$

- 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:

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



Particle Filter

- Recursive Bayes filter
- Non-parametric approach
- Models the distribution by weighted samples
- **Prediction:** draw from the proposal
- **Correction:** weigh particles 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

Monte Carlo Localization

- **Each particle is a pose hypothesis**
- **Prediction**: For each particle, sample a new pose from the the motion model

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

- **Correction**: Weigh samples according to the observation model

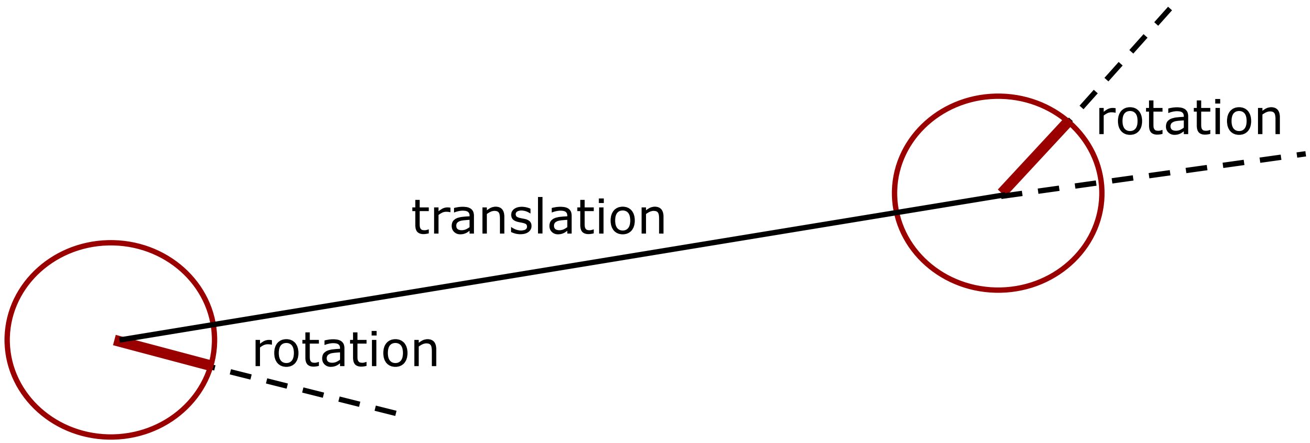
$$w_t^{[j]} \propto p(z_t \mid x_t^{[j]})$$

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

Reminder: Odometry Motion Model

According to the estimated motion

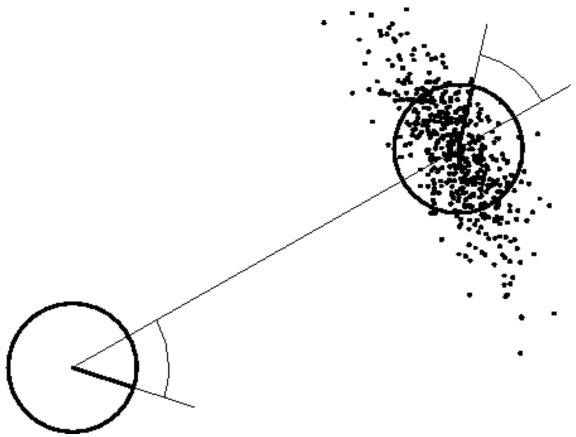
Reminder: Odometry Motion Model



Decompose the motion into

- Traveled distance
- Start rotation
- End rotation

Reminder: Odometry Motion Model

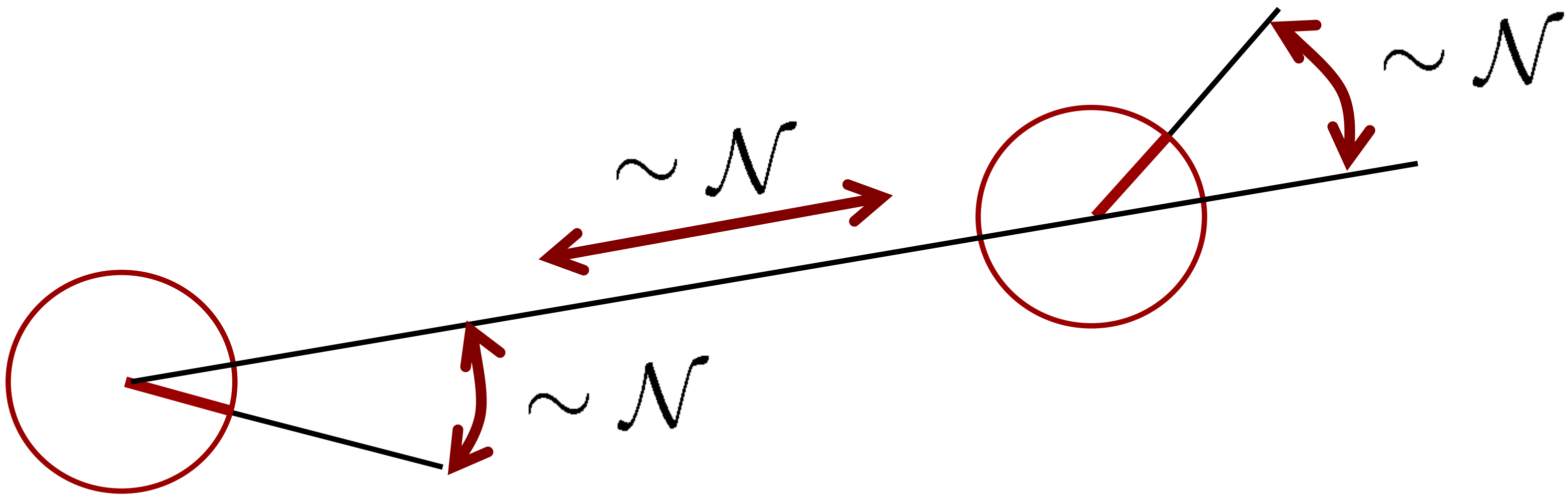


- Uncertainty in the translation of the robot: Gaussian over the traveled distance
- Uncertainty in the rotation of the robot: Gaussians over start and end rotation
- **For each particle, draw a new pose by sampling from three normal distributions**

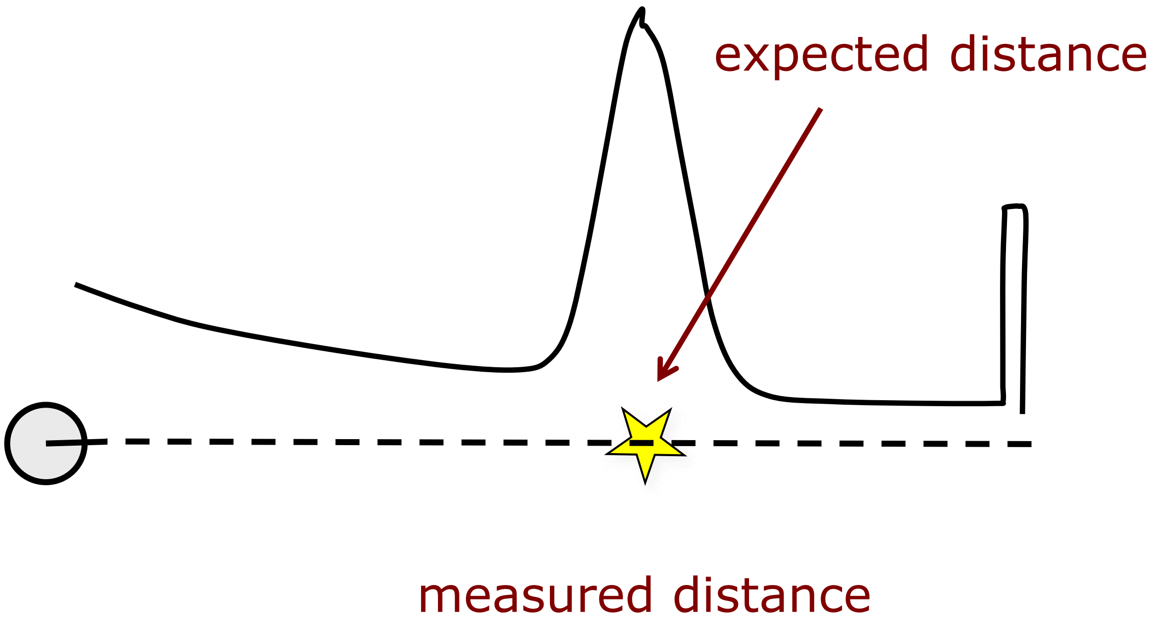
Reminder: Odometry Motion Model

- Noise in odometry $u = (\delta_{rot1}, \delta_{trans}, \delta_{rot2})$
- Example: Gaussian noise

$$u \sim \mathcal{N}(0, \Sigma)$$



Reminder: Proximity Sensor Model



Particle Filter for Localization

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

$$
\begin{array}{l}
\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset \\
\textit{for } j = 1 \textit{ to } J \textit{ do} \\
\qquad \textit{sample } \underline{x_t^{[j]} \sim p(x_t \mid u_t, x_{t-1}^{[j]})} \\
\qquad w_t^{[j]} = \underline{p(z_t \mid x_t^{[j]})} \\
\qquad \bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[j]}, w_t^{[j]} \rangle \\
\textit{endfor} \\
\textit{for } j = 1 \textit{ to } J \textit{ do} \\
\qquad \textit{draw } i \in 1, \ldots, J \textit{ with probability} \propto w_t^{[i]} \\
\qquad \textit{add } x_t^{[i]} \textit{ to } \mathcal{X}_t \\
\textit{endfor} \\
\textit{return } \mathcal{X}_t
\end{array}
$$

MCL – Correction Step

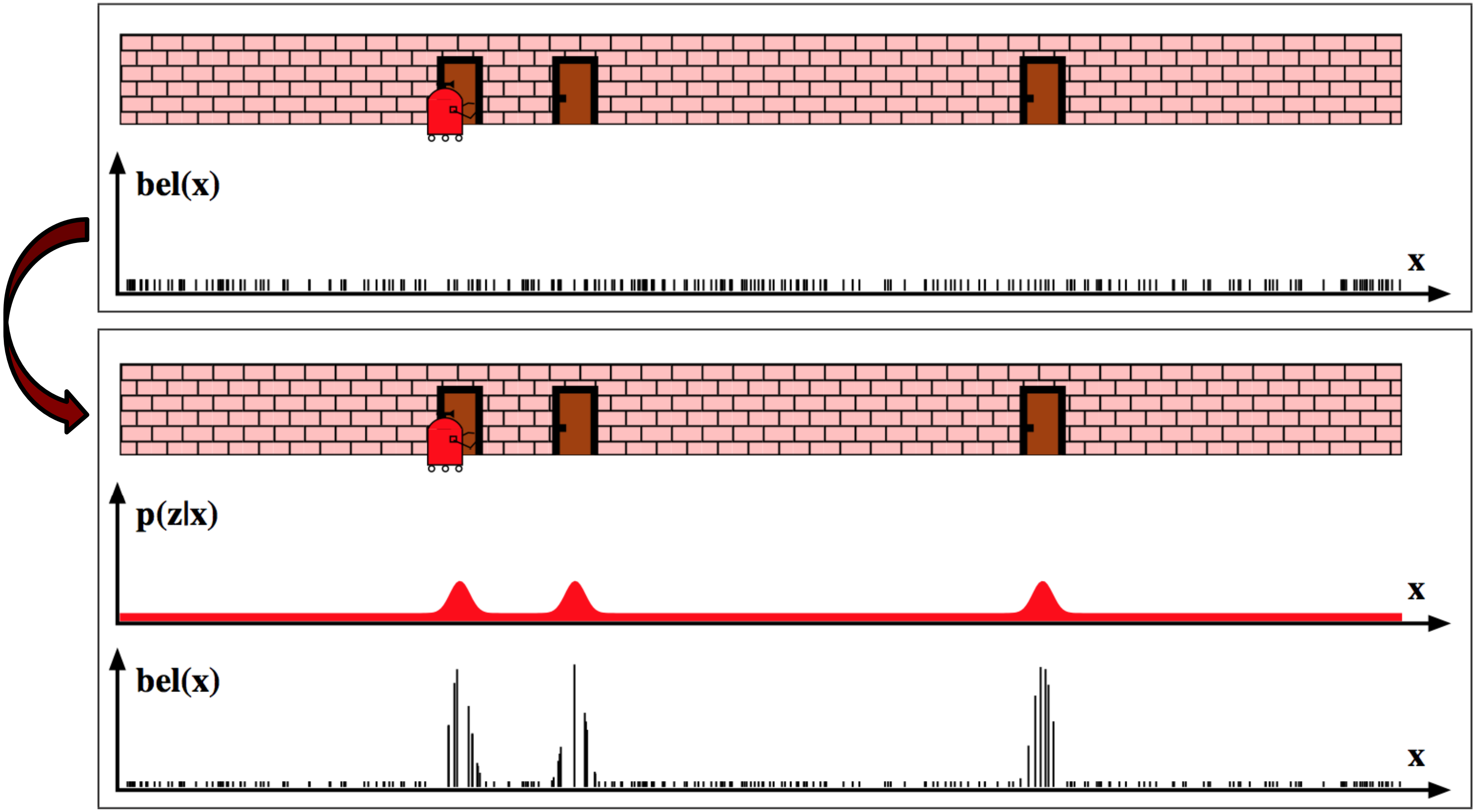


Image Courtesy: S. Thrun, W. Burgard, D. Fox

MCL – Prediction Step

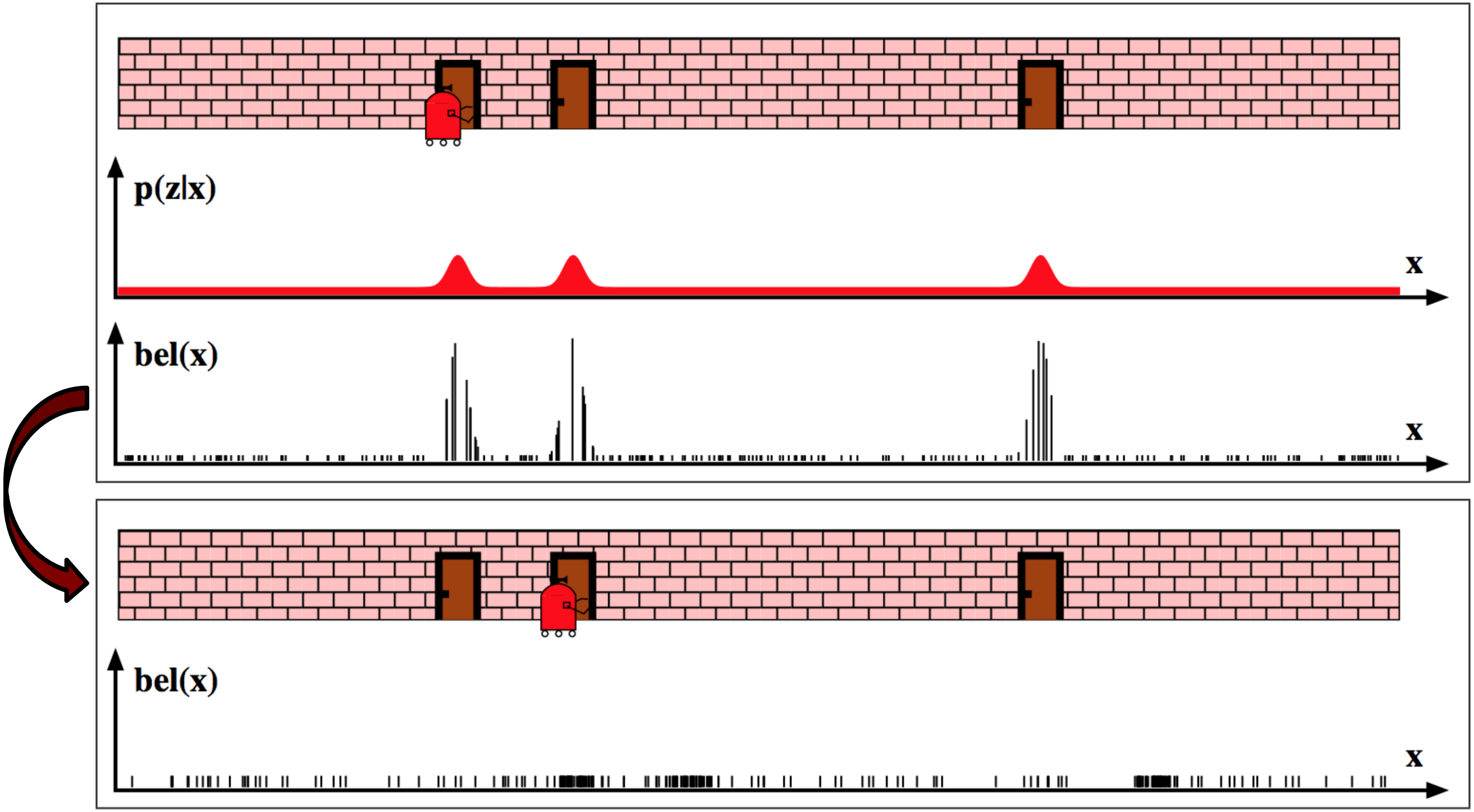


Image Courtesy: S. Thrun, W. Burgard, D. Fox

MCL – Correction Step

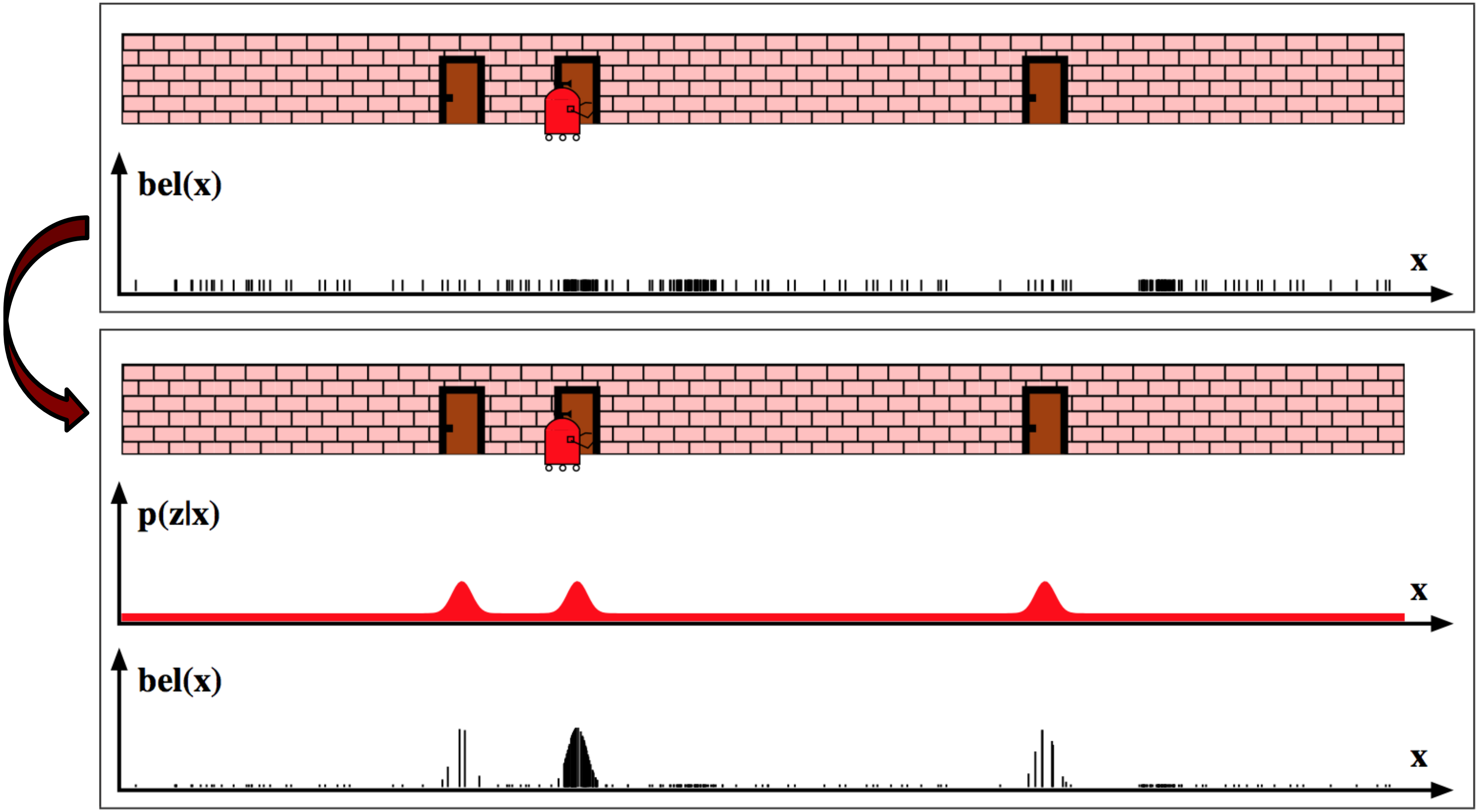


Image Courtesy: S. Thrun, W. Burgard, D. Fox

MCL – Prediction Step

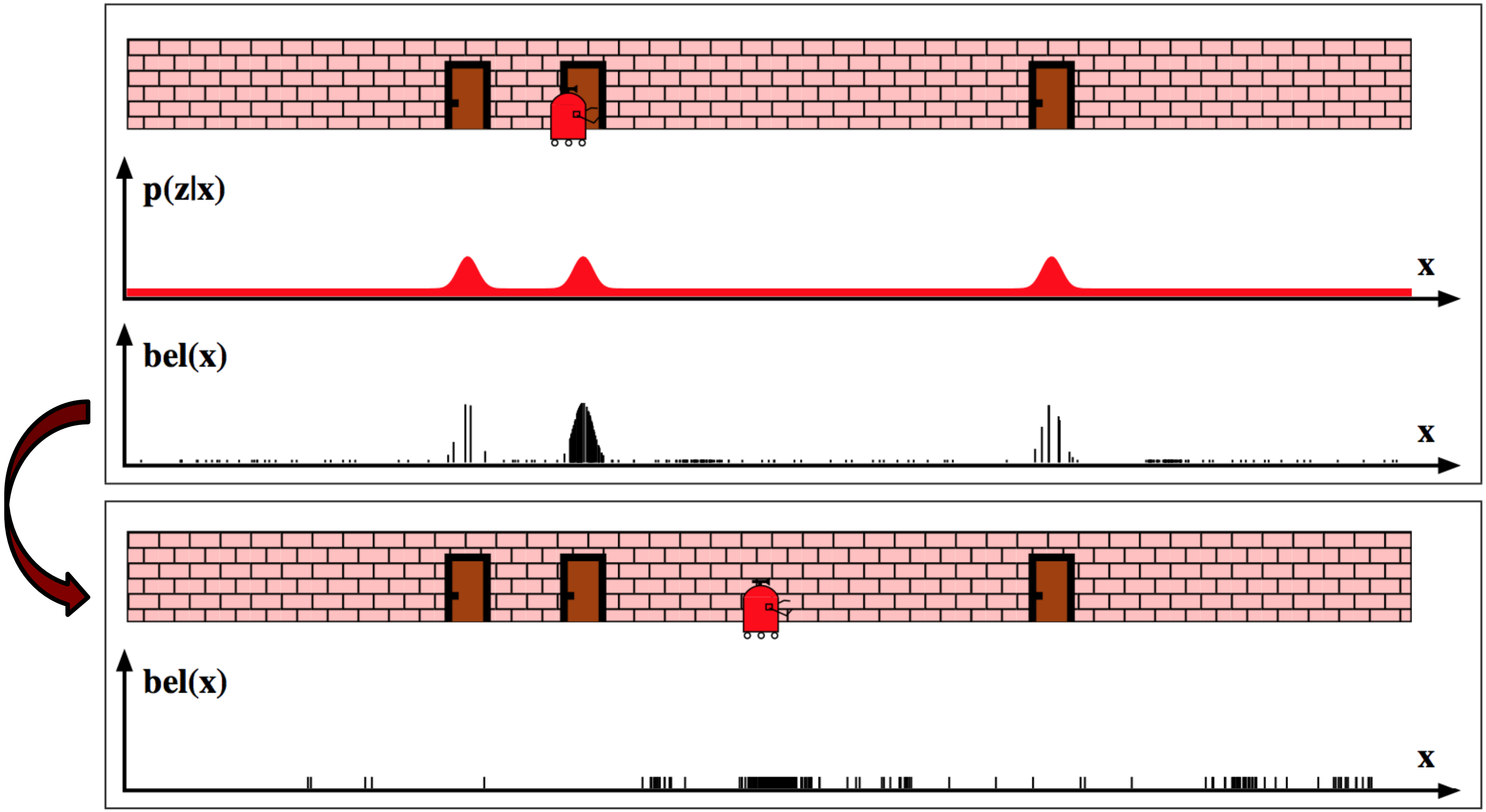
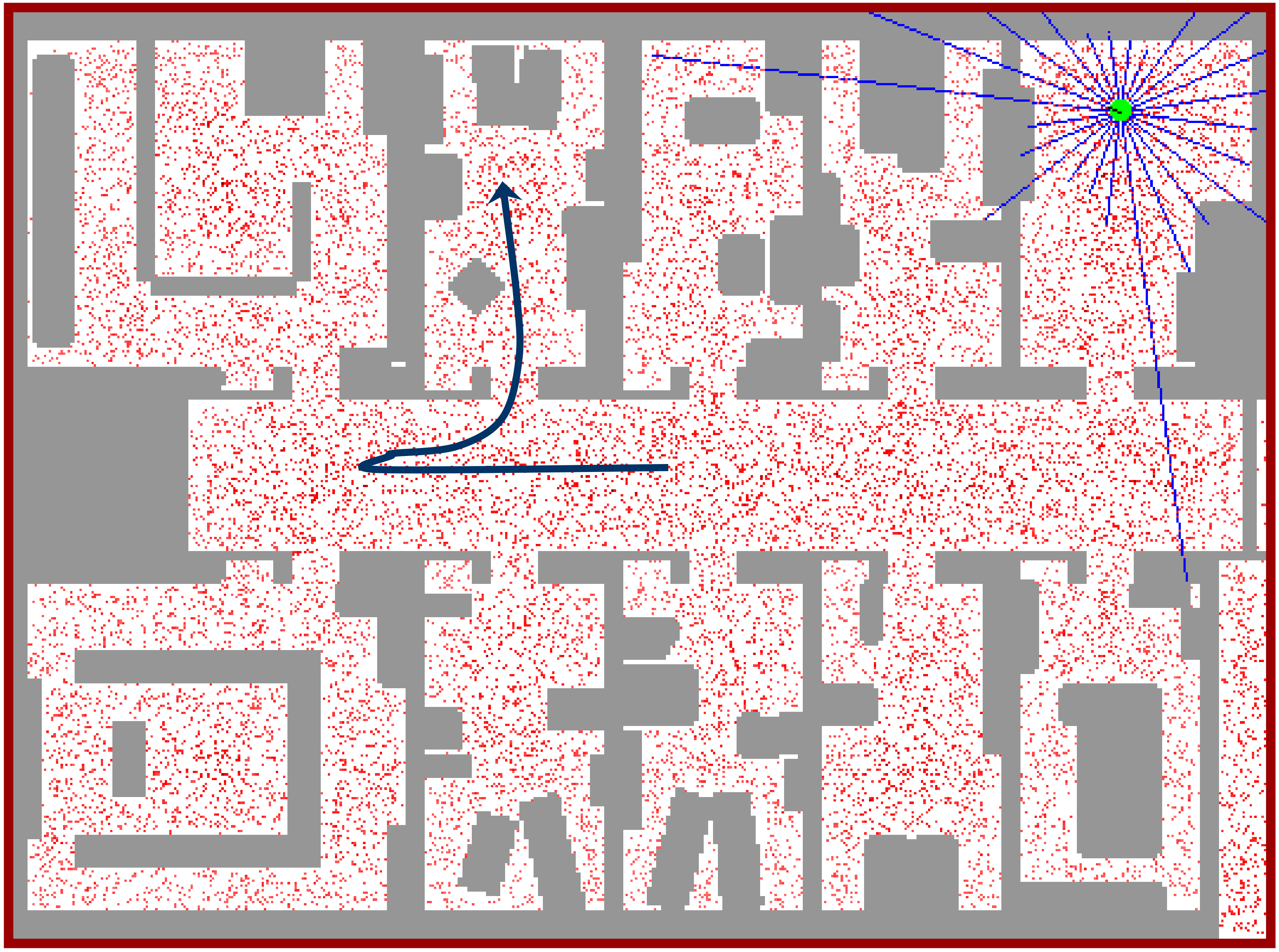


Image Courtesy: S. Thrun, W. Burgard, D. Fox

Example: Sample-Based Localization (Sonar)



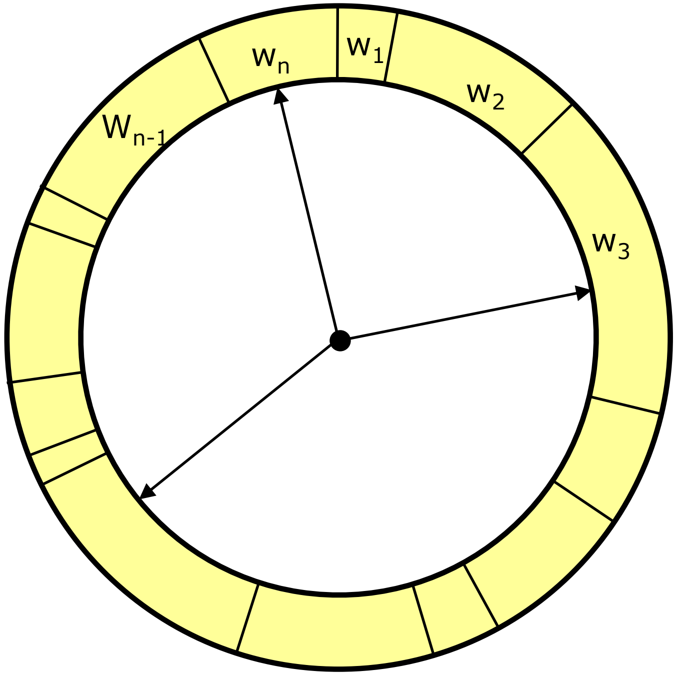
Resampling

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

Assumption: normalized weights

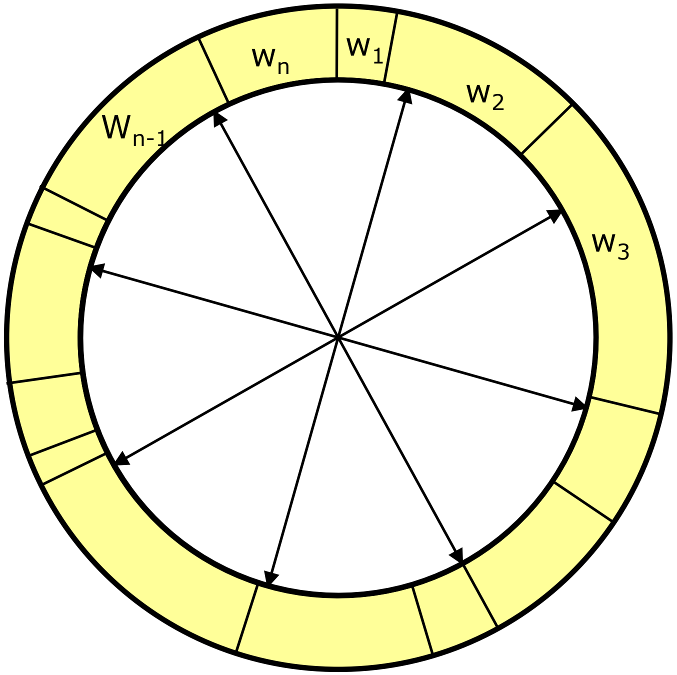
Resampling

"roulette wheel"



initial value between 0 and 1/J

step size = *1/J*



- Draw randomly between 0 and 1
- Binary search
- Repeat *J* times
- O(*J* log *J*)

- Systematic resampling
- Low variance resampl.
- O(*J*)
- Also called "stochastic universal resampling"

Low Variance Resampling

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

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

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

$c = w_t^{[1]}$ cumulative sum of weights

$i = 1$

for $j = 1$ *to* J *do* — *J* = #particles

  $U = r + (j-1)J^{-1}$ — *step size = 1/J*

  while $U > c$ — decide whether or not to take particle *i*

    $i = i + 1$

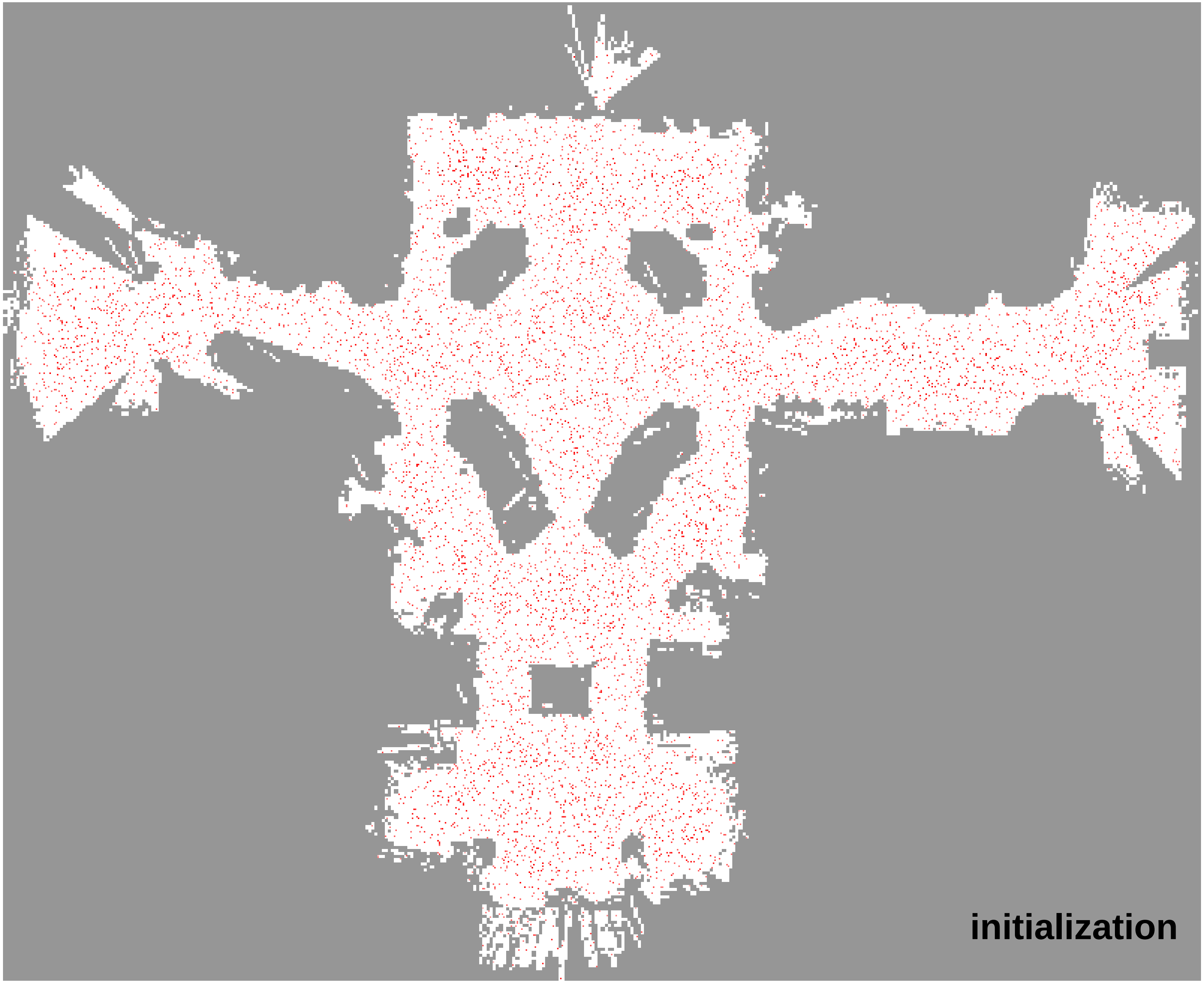
    $c = c + w_t^{[i]}$

  endwhile

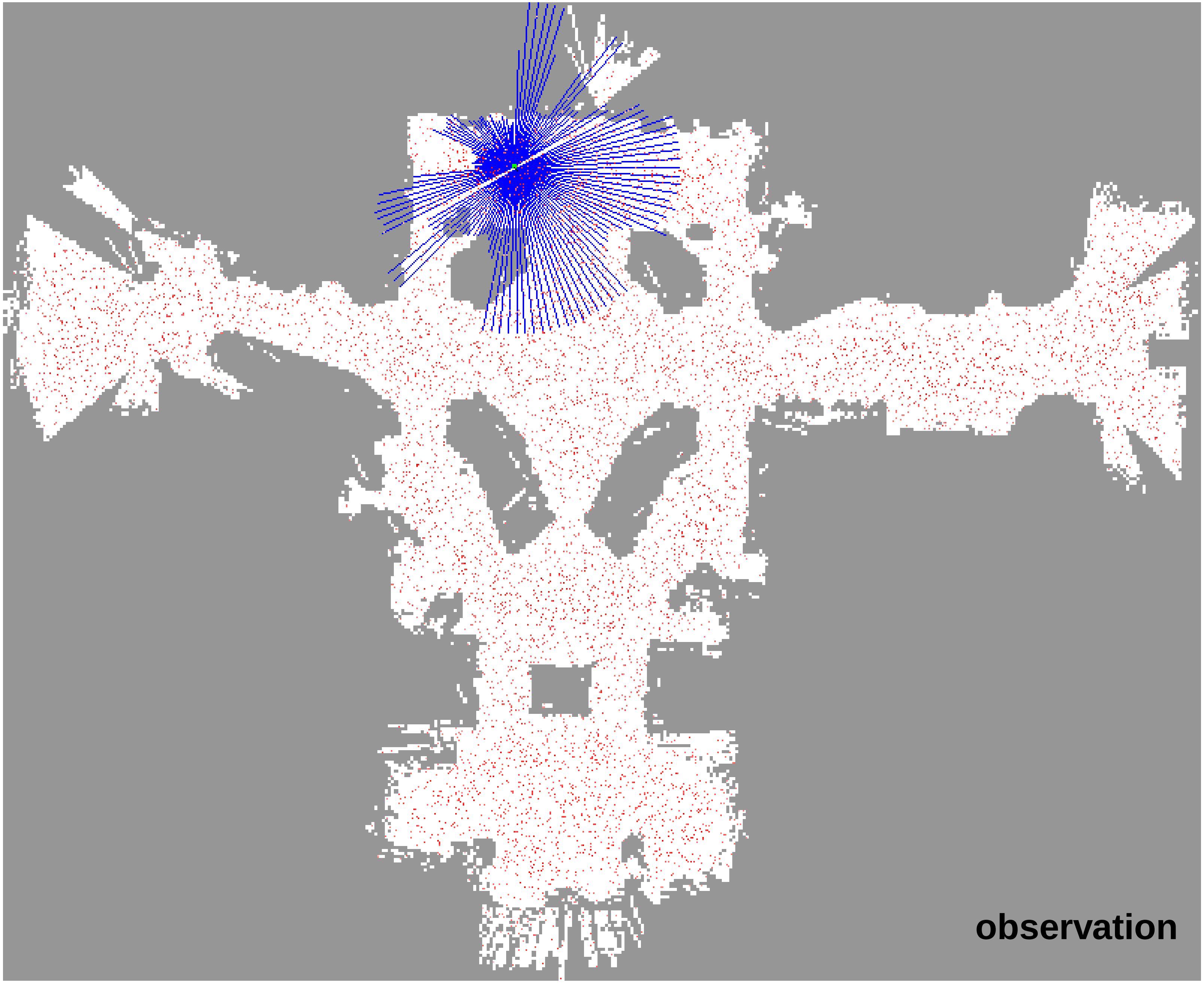
  add $x_t^{[i]}$ *to* $\bar{\mathcal{X}}_t$

endfor

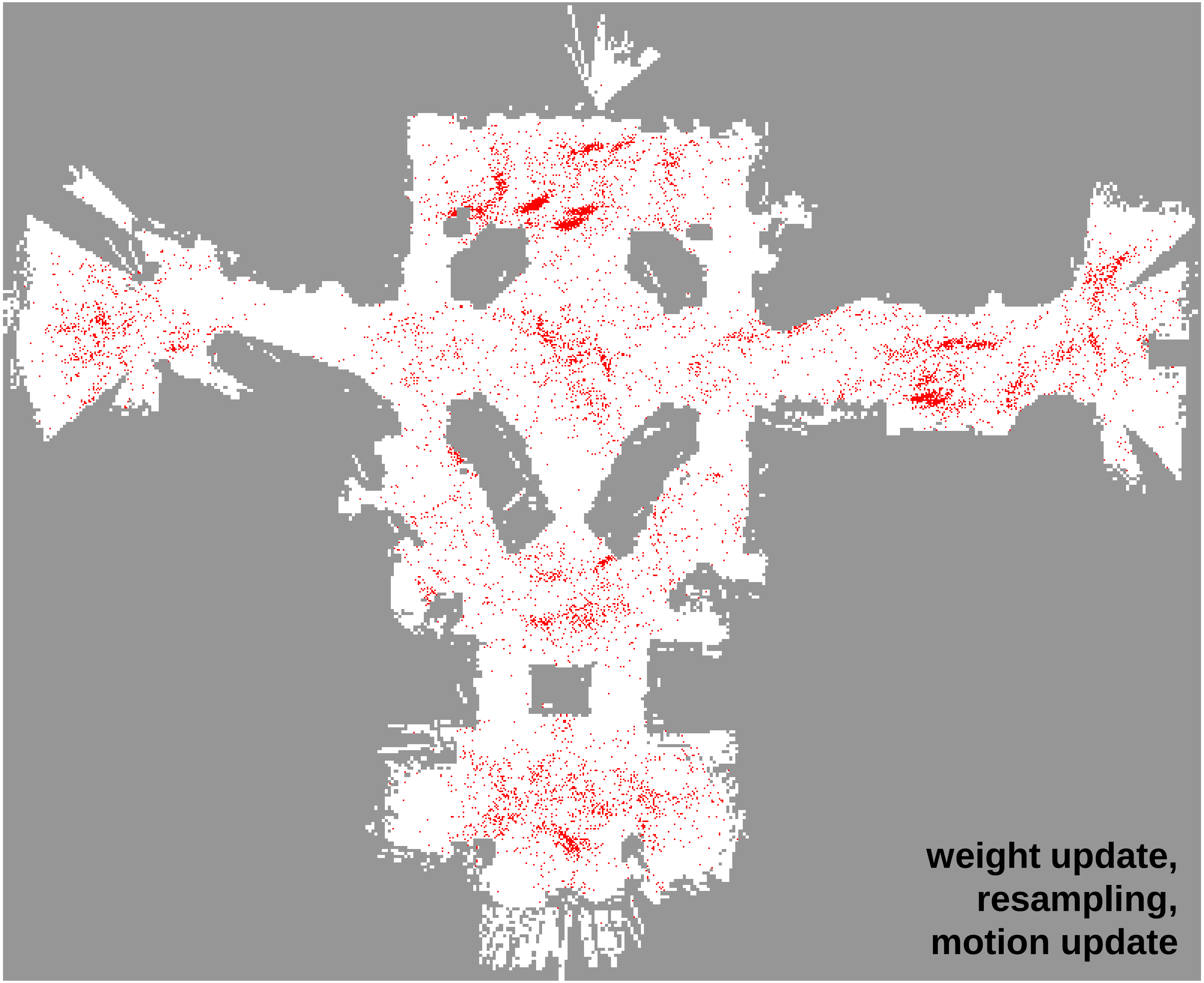
return $\bar{\mathcal{X}}_t$



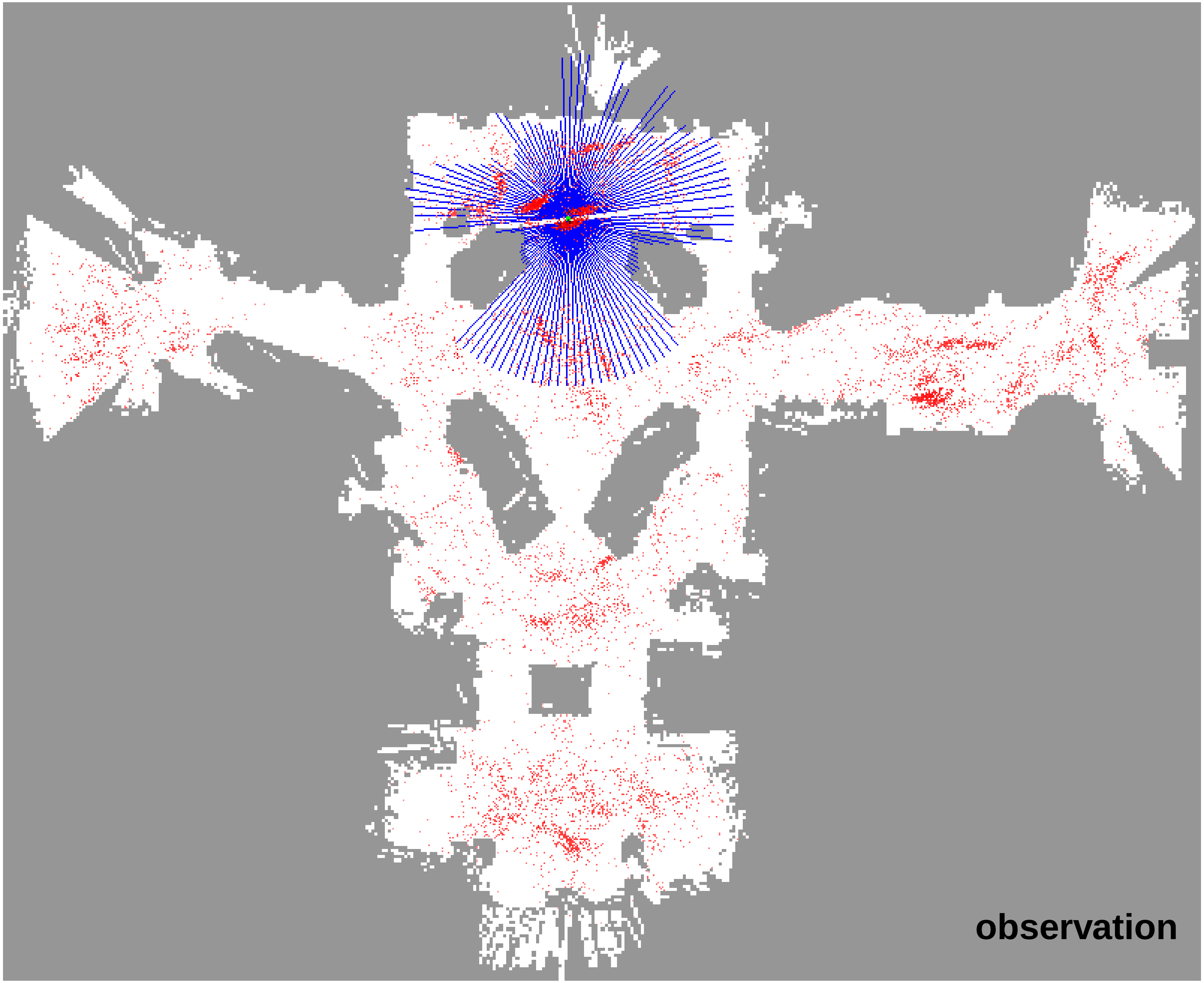
Courtesy: Thrun, Burgard, Fox



Courtesy: Thrun, Burgard, Fox

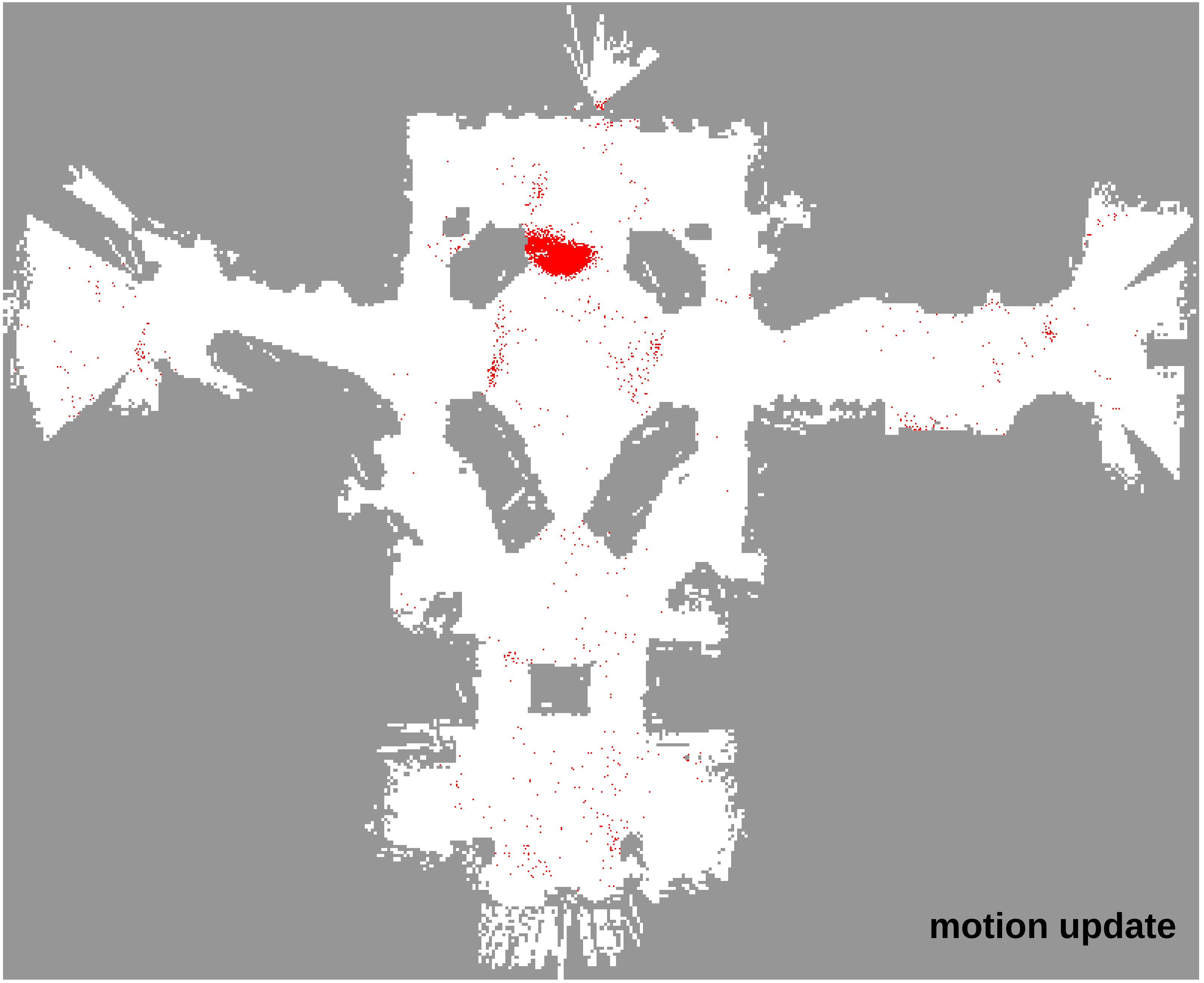


Courtesy: Thrun, Burgard, Fox

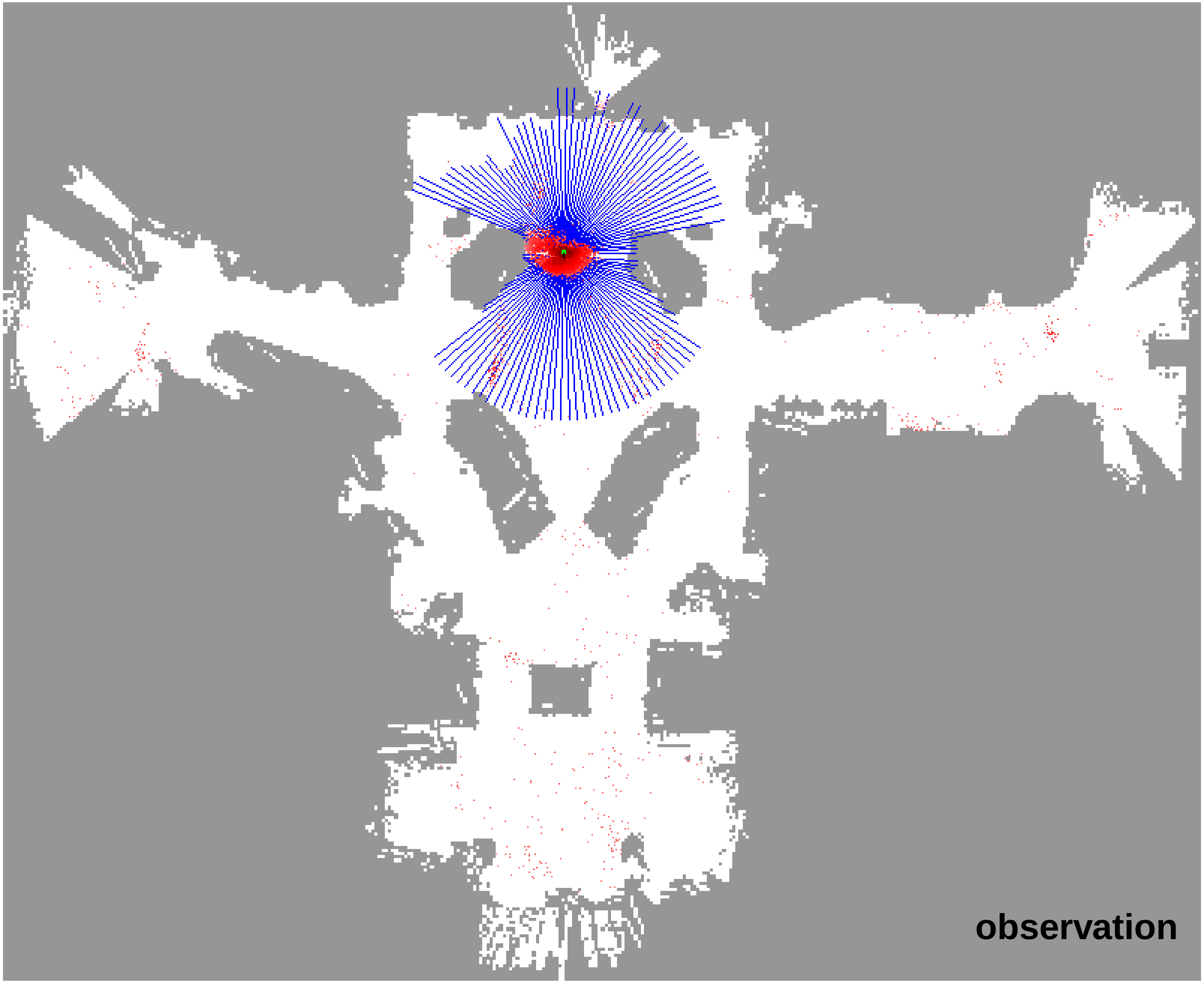


Courtesy: Thrun, Burgard, Fox

Courtesy: Thrun, Burgard, Fox

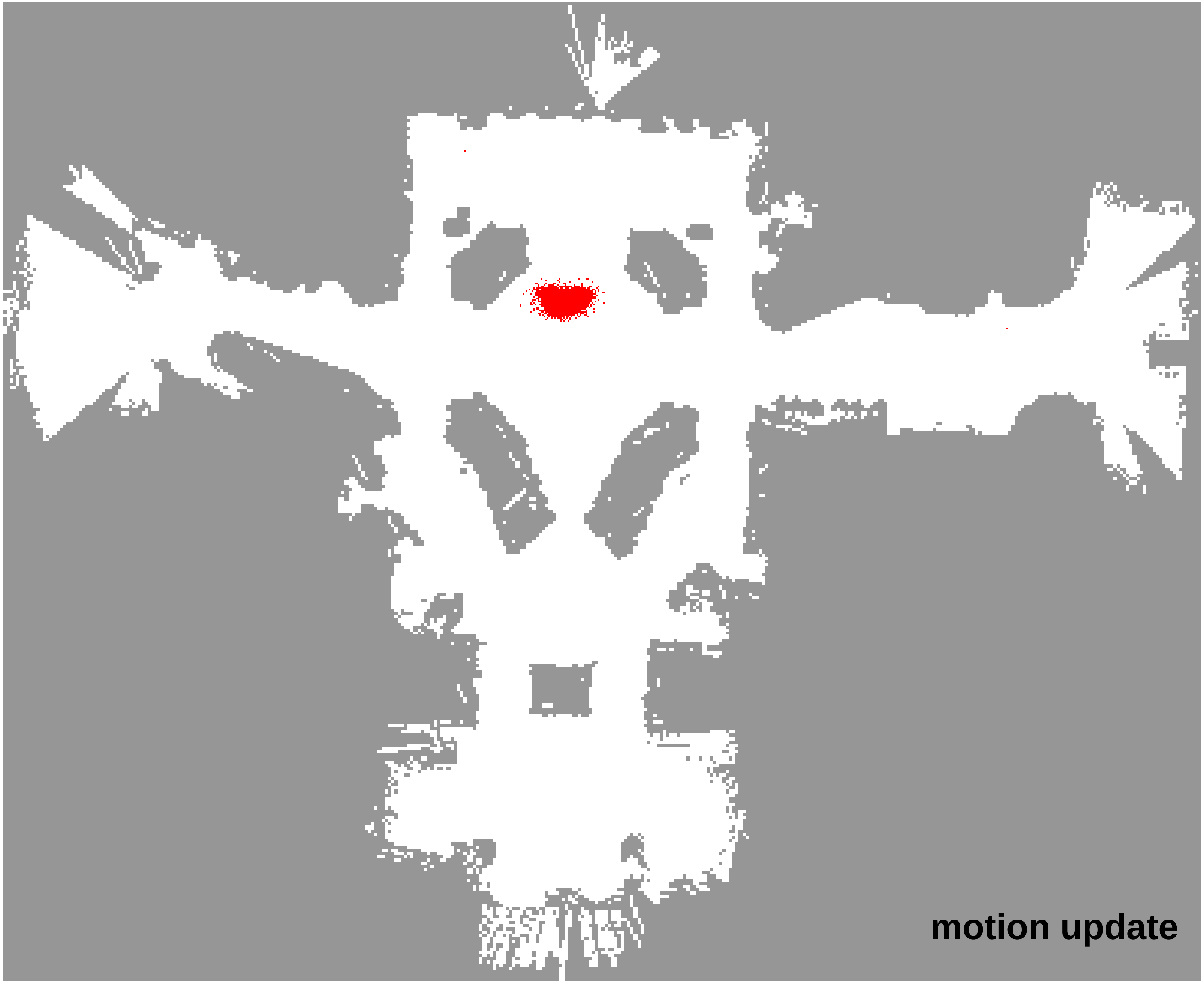


Courtesy: Thrun, Burgard, Fox

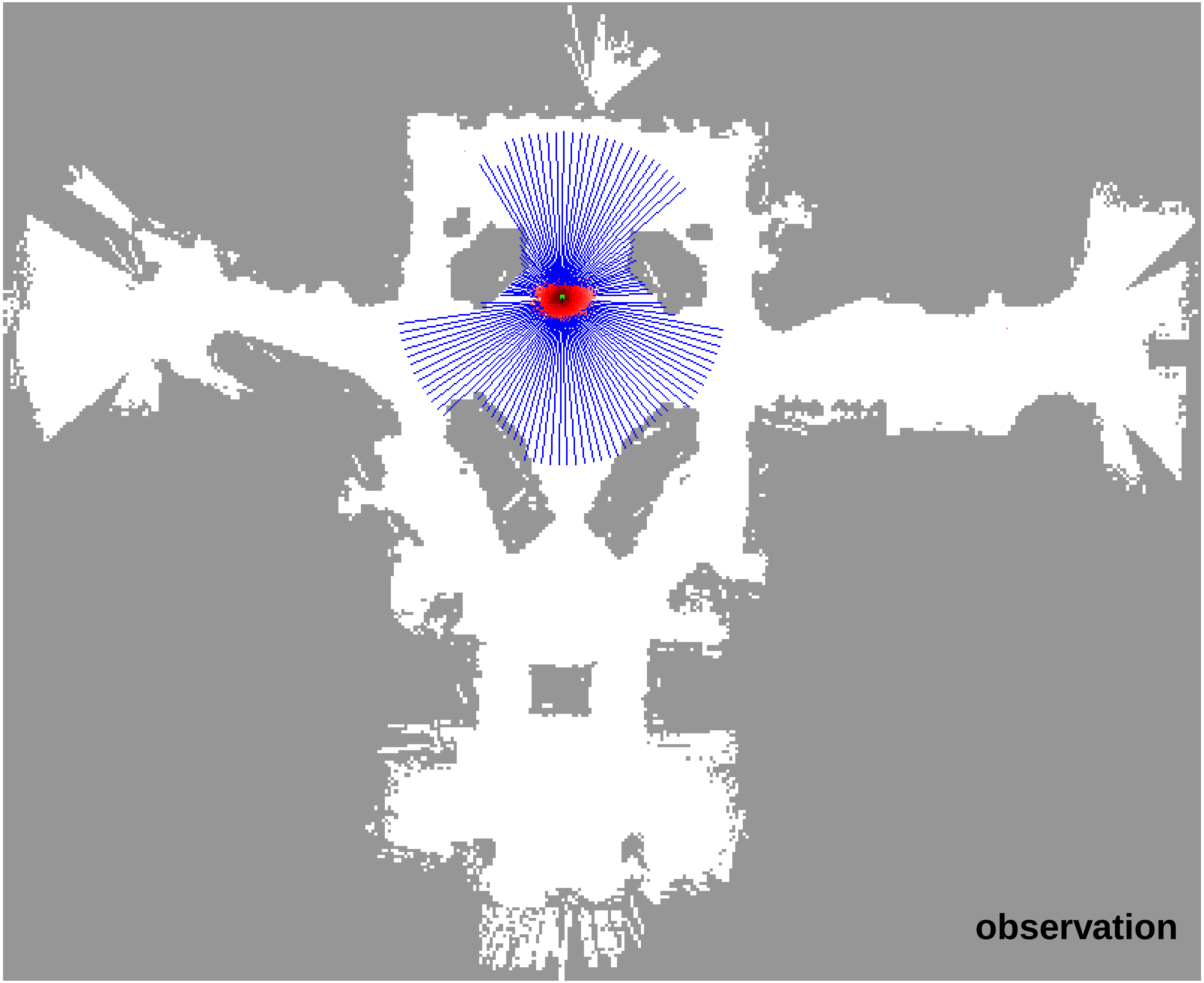


Courtesy: Thrun, Burgard, Fox

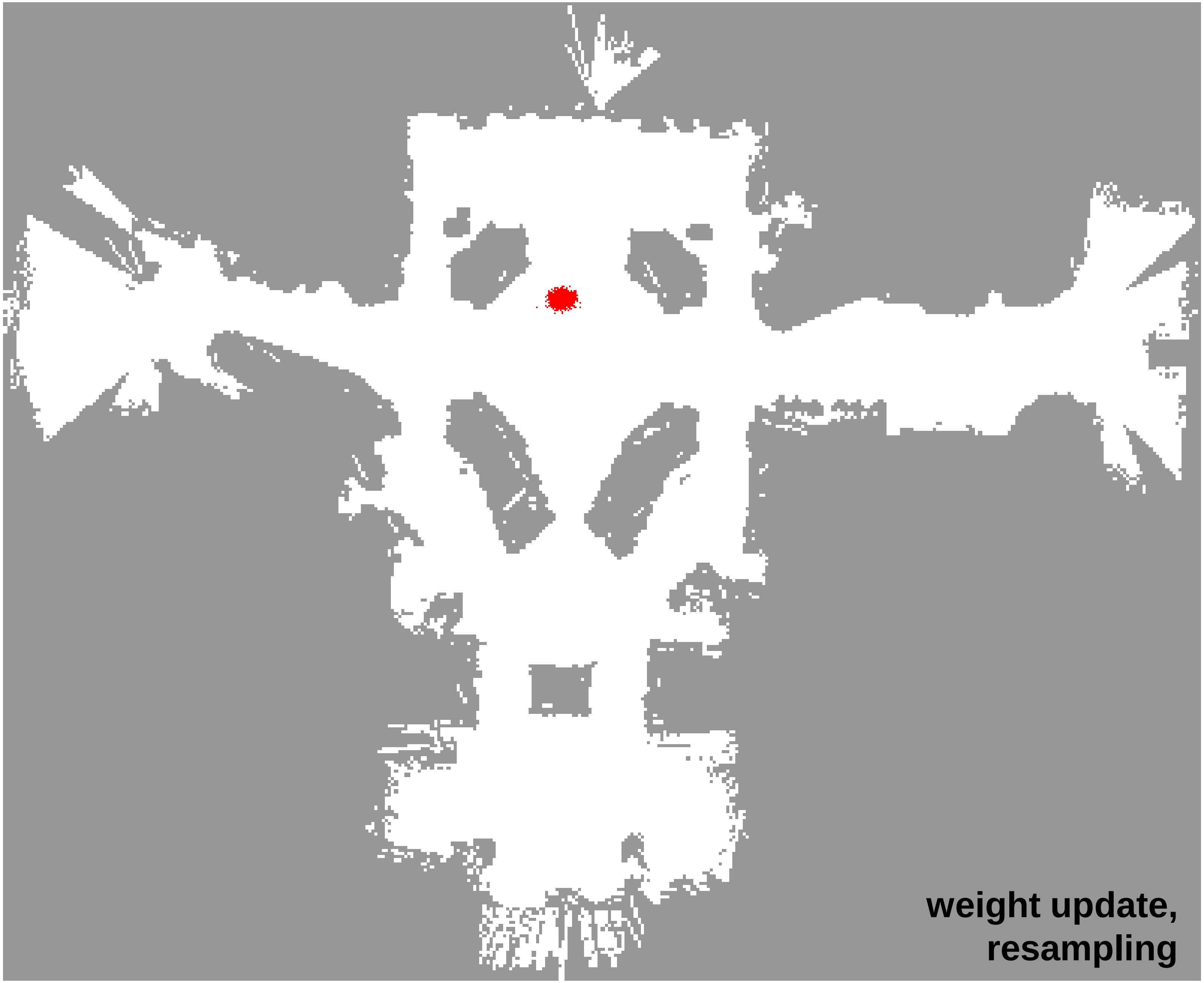
Courtesy: Thrun, Burgard, Fox



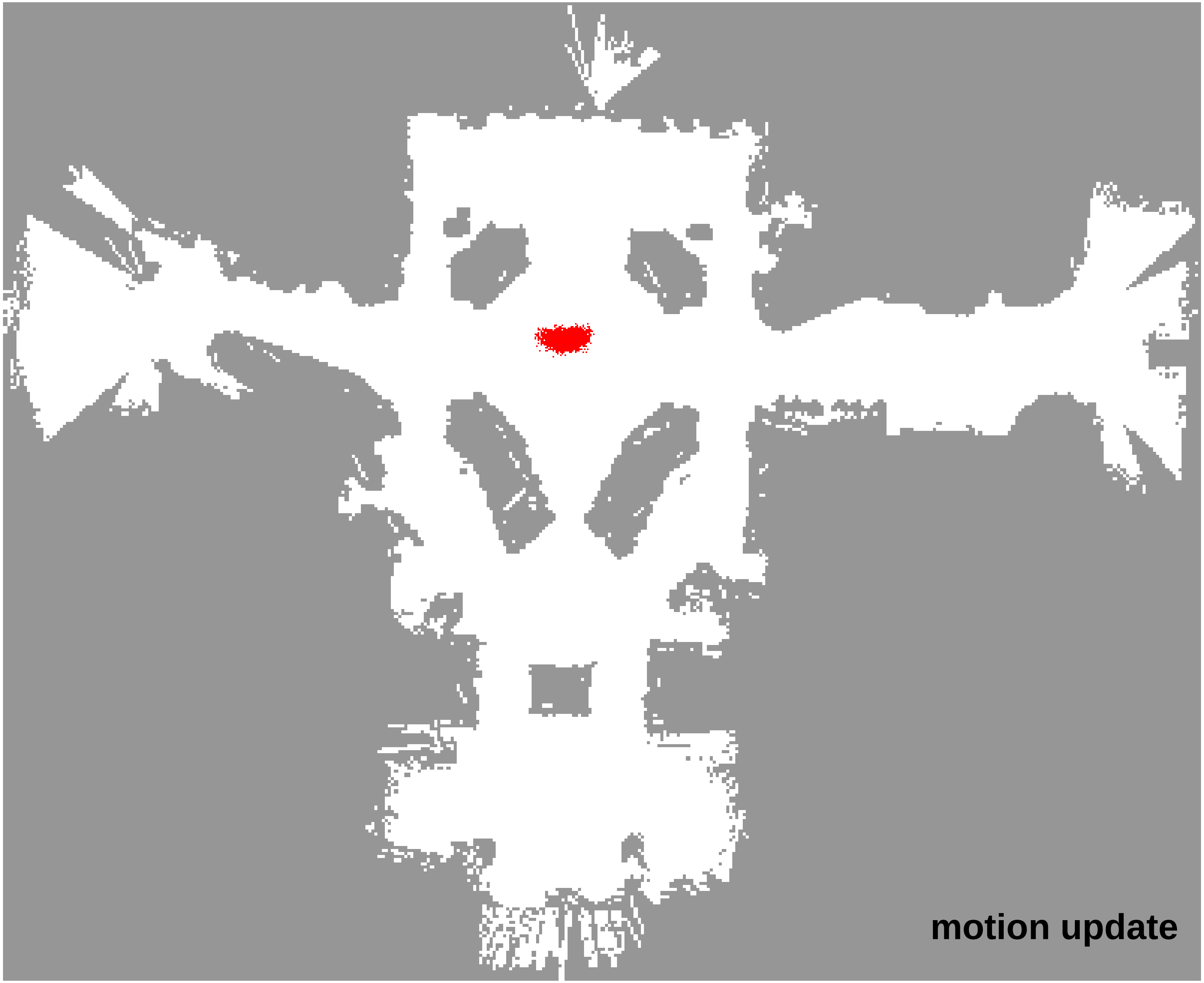
Courtesy: Thrun, Burgard, Fox



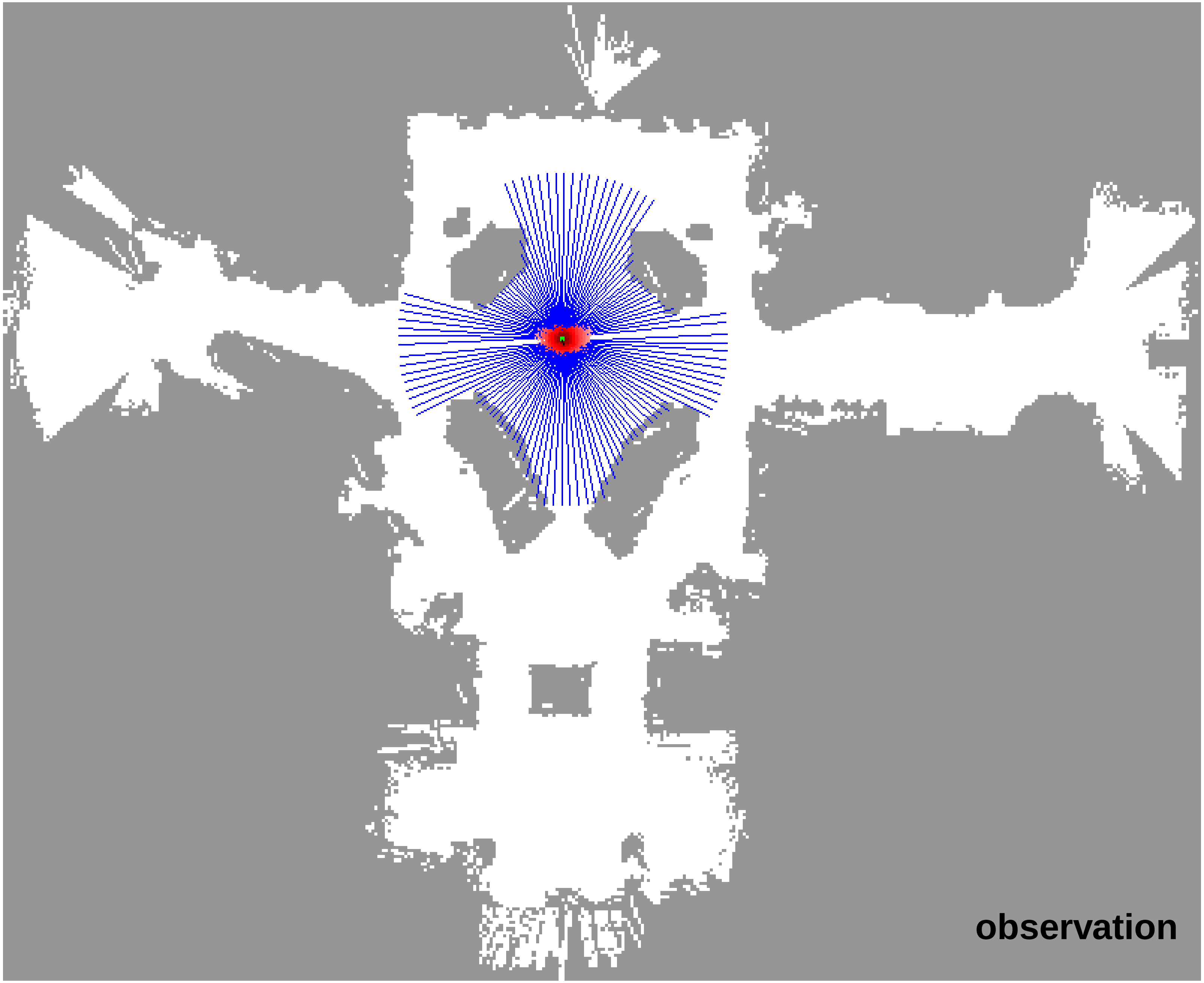
Courtesy: Thrun, Burgard, Fox



Courtesy: Thrun, Burgard, Fox

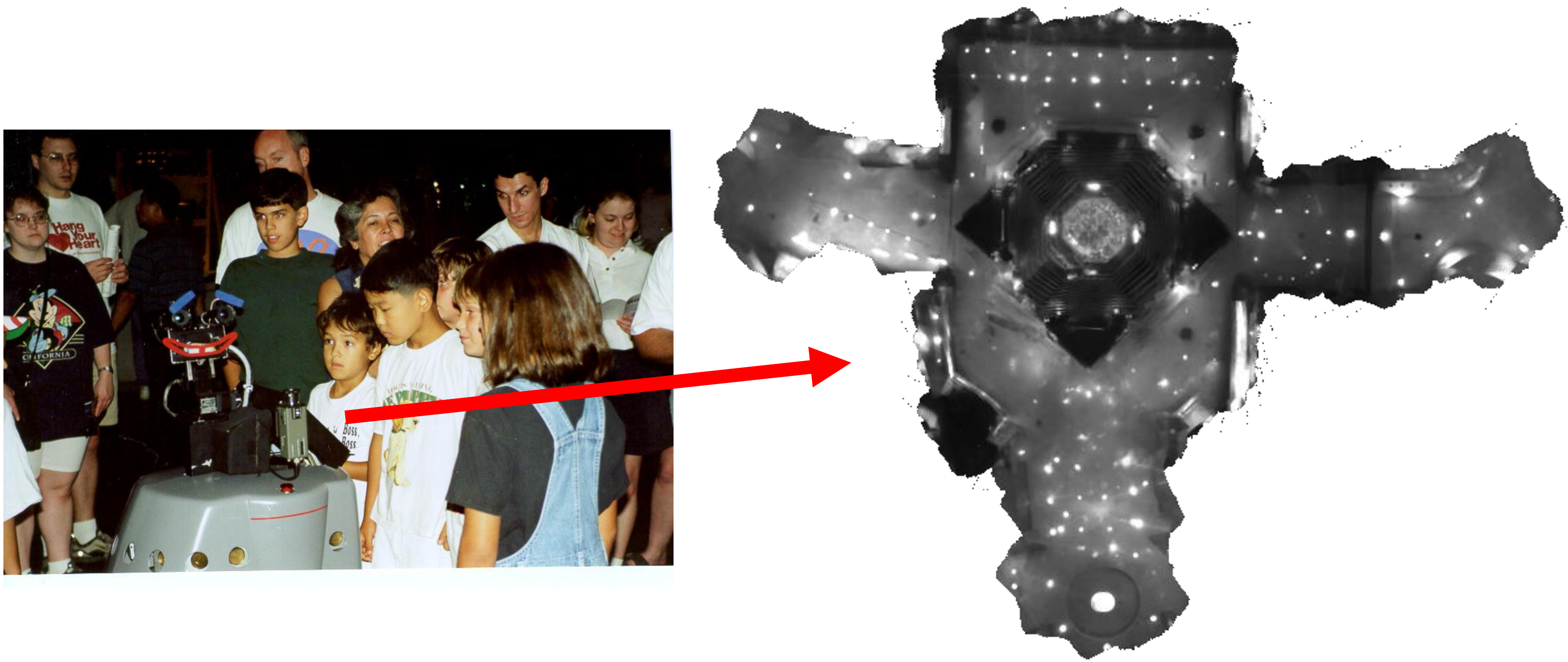


Courtesy: Thrun, Burgard, Fox



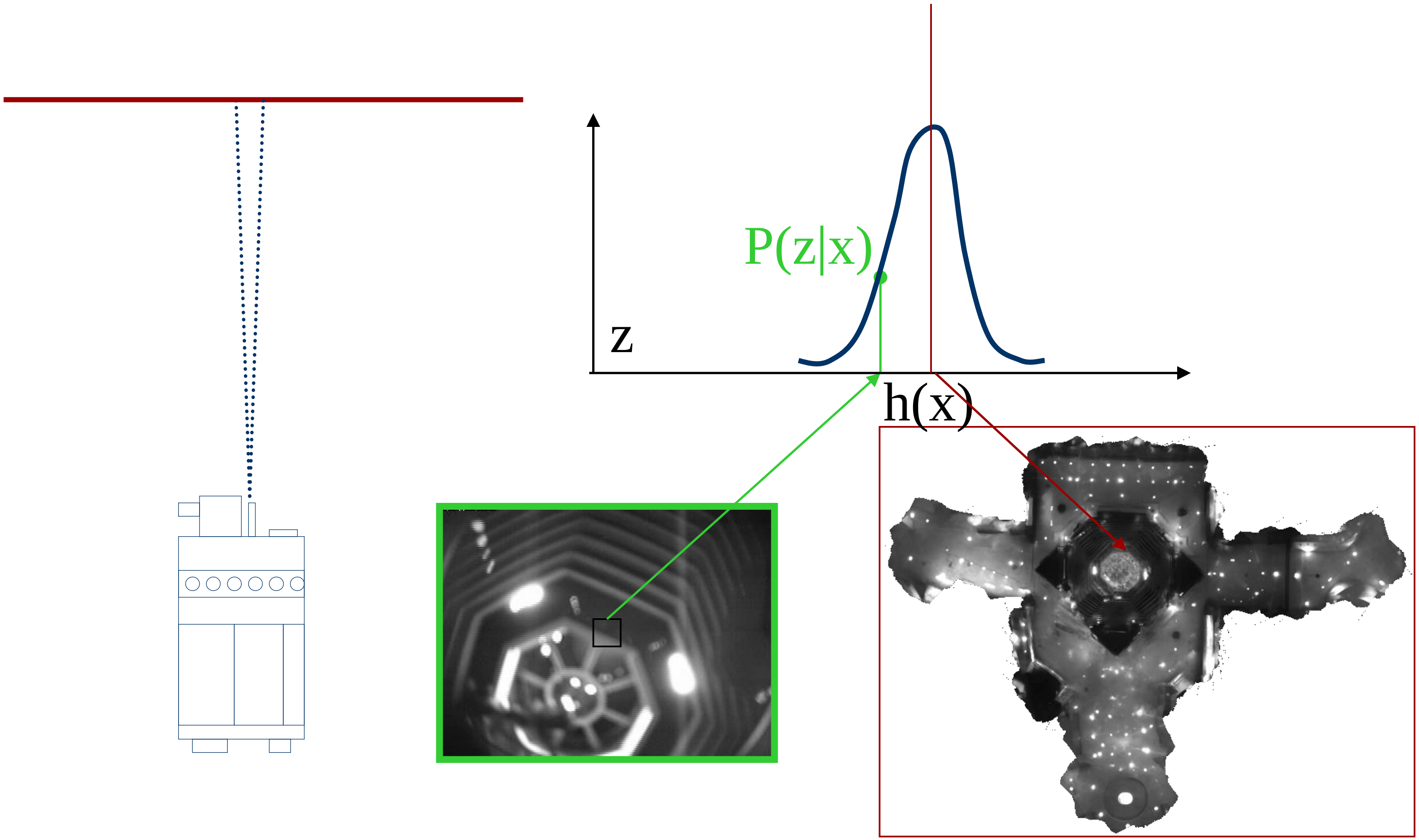
Courtesy: Thrun, Burgard, Fox

Using Ceiling Maps for Localization



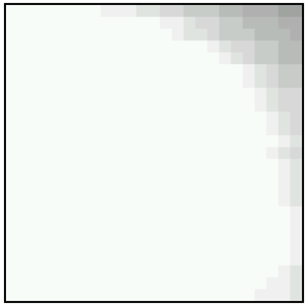
[Dellaert et al. 99]

Vision-Based Localization

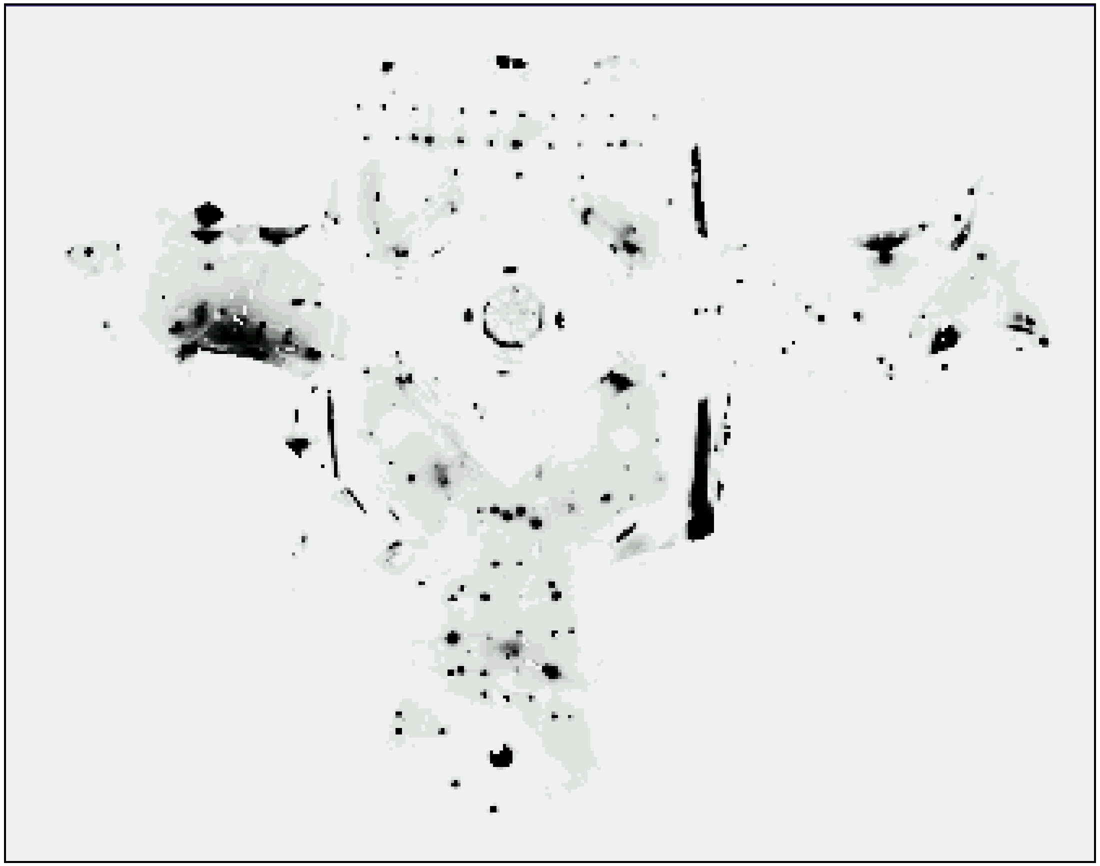


Under a Light

Measurement z:

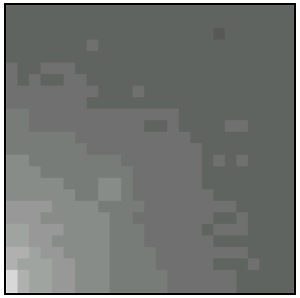


***P(z|x)*:**

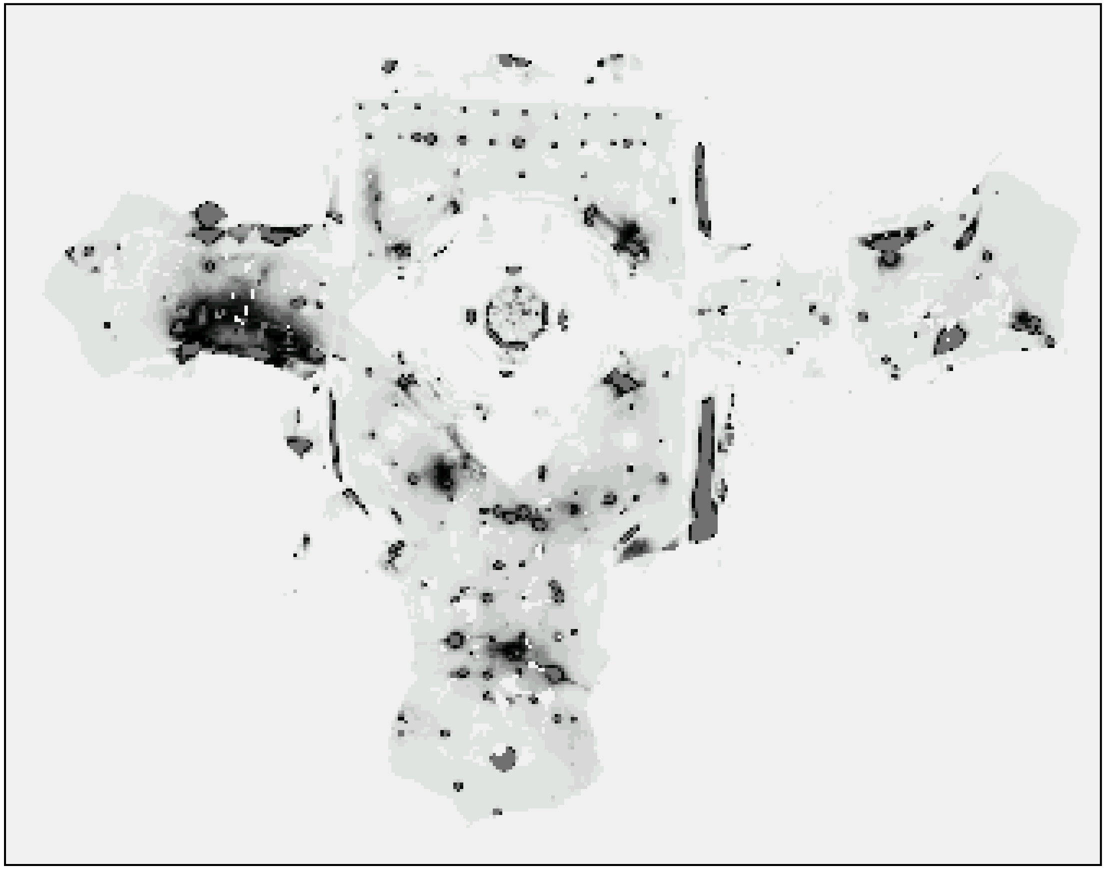


Next to a Light

Measurement z:



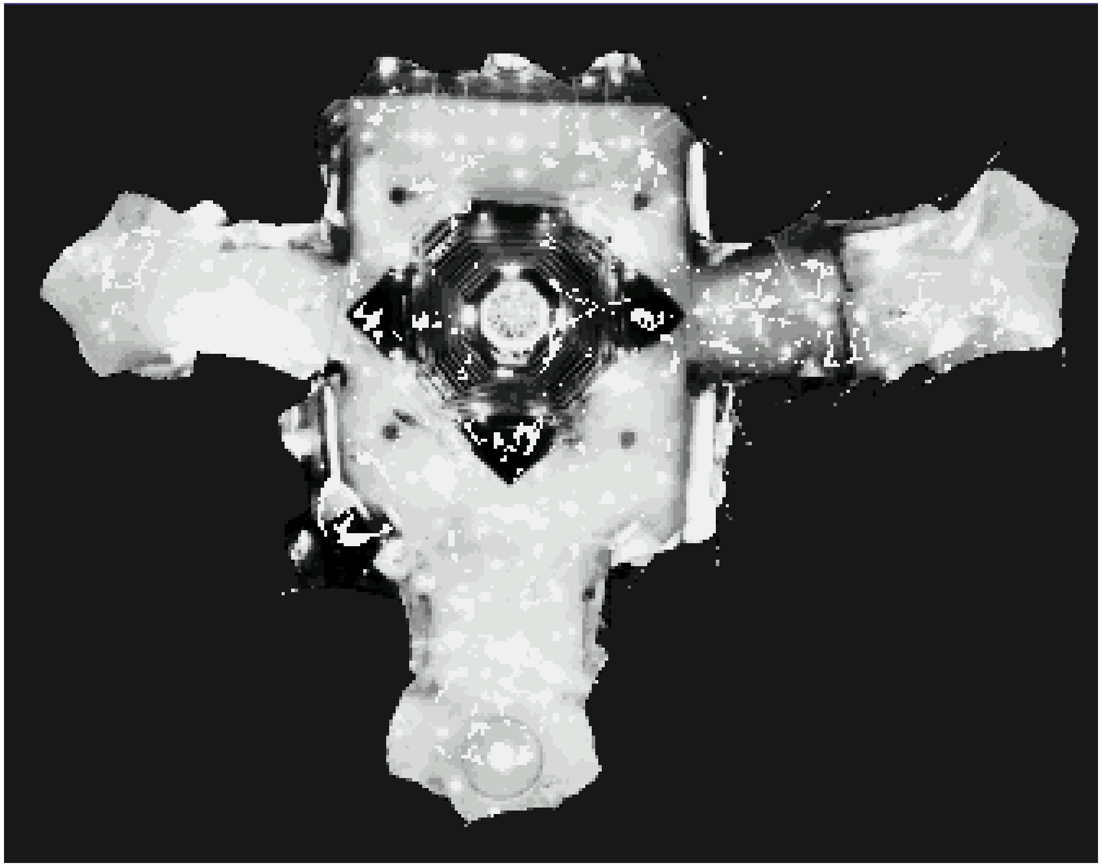
***P(z|x)*:**



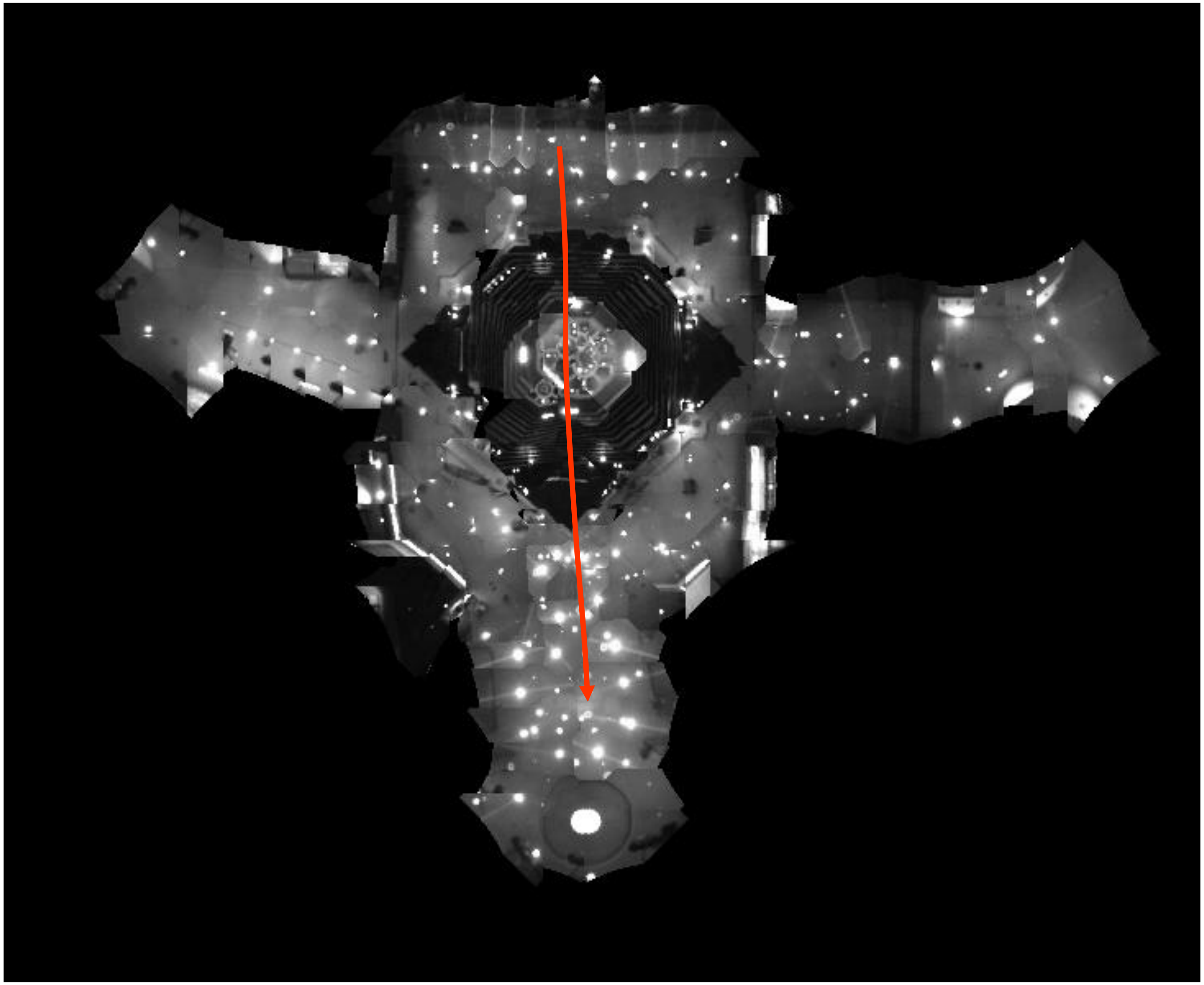
Elsewhere

Measurement z:

***P(z|x)*:**



Global Localization Using Vision



How to deal with localization errors?

- The approach described so far is able
  - to track the pose of a mobile robot and
  - to globally localize the robot

- How can we deal with localization errors (i.e., the kidnapped robot problem)?

Approaches

- At each time step, randomly insert a fixed number of samples
- Alternatively, insert random samples proportional to the average likelihood of the particles

Summary – Particle Filters

- Particle filters are non-parametric Bayes filters
- Belief represented by a set of weighted samples
- Proposal distribution to draw the samples for the next time step
- Particle weight to account for the differences between the proposal and the target
- Re-sampling: Draw new particles with a probability proportional to the weight

Summary – PF Localization

- Particles are propagated according to the motion model
- Particles are weighted according to the likelihood of the observation
- Called: Monte-Carlo localization (MCL)
- Used in many practical localization systems
- **The art is to design appropriate motion and sensor models**

Acknowledgment

- These slides have been created by Wolfram Burgard, Dieter Fox, Cyrill Stachniss and Maren Bennewitz