

Cognitive Robotics

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

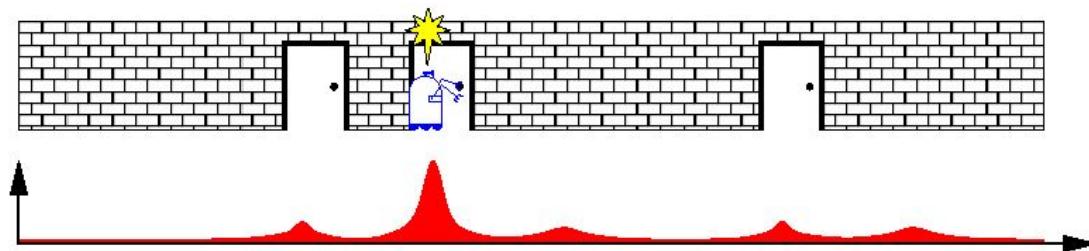
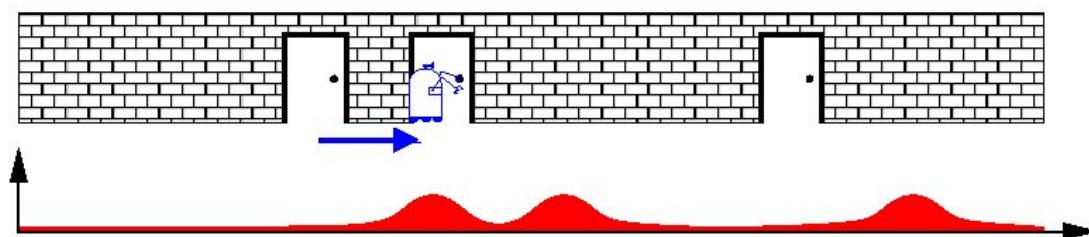
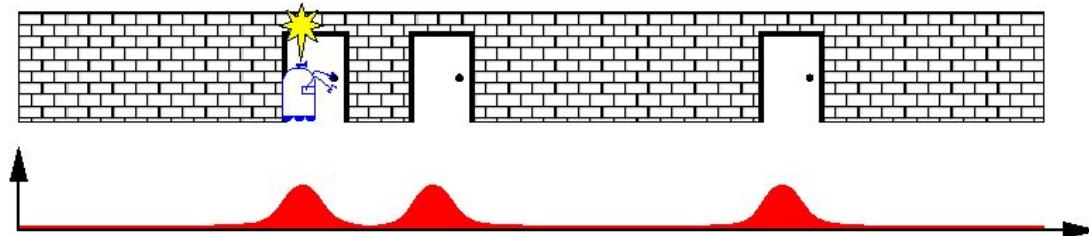
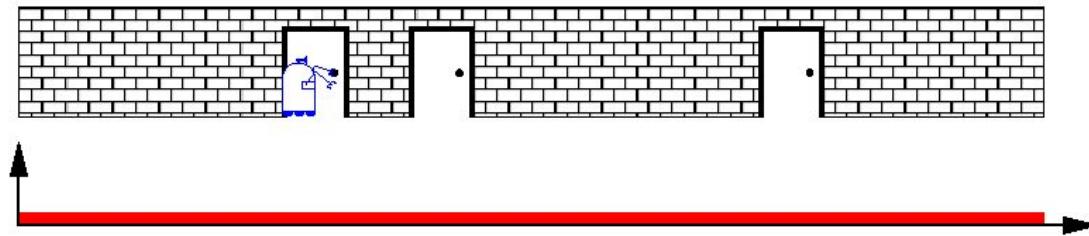
AbdElMoniem Bayoumi, PhD

Fall 2022

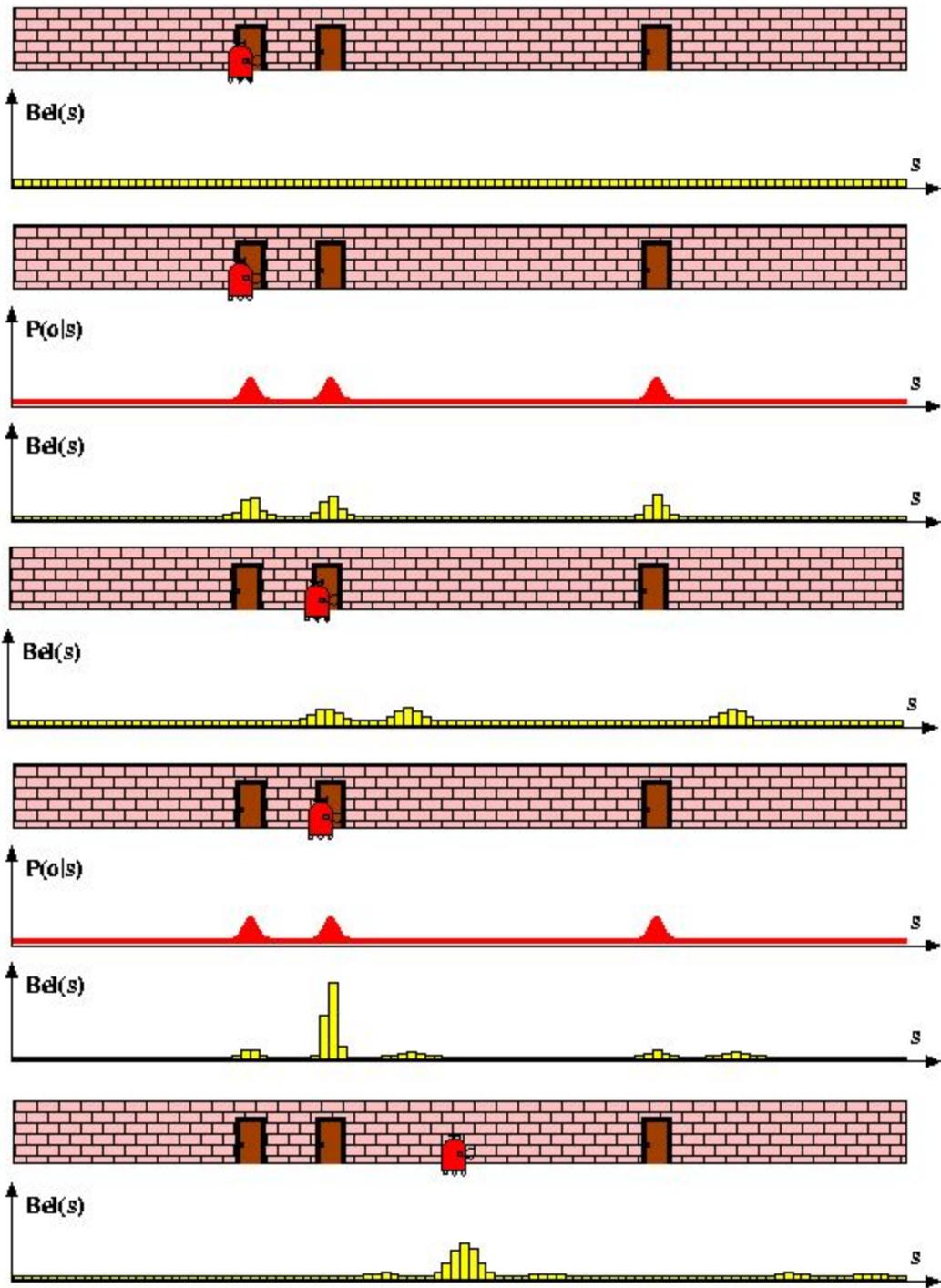
Acknowledgment

- 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

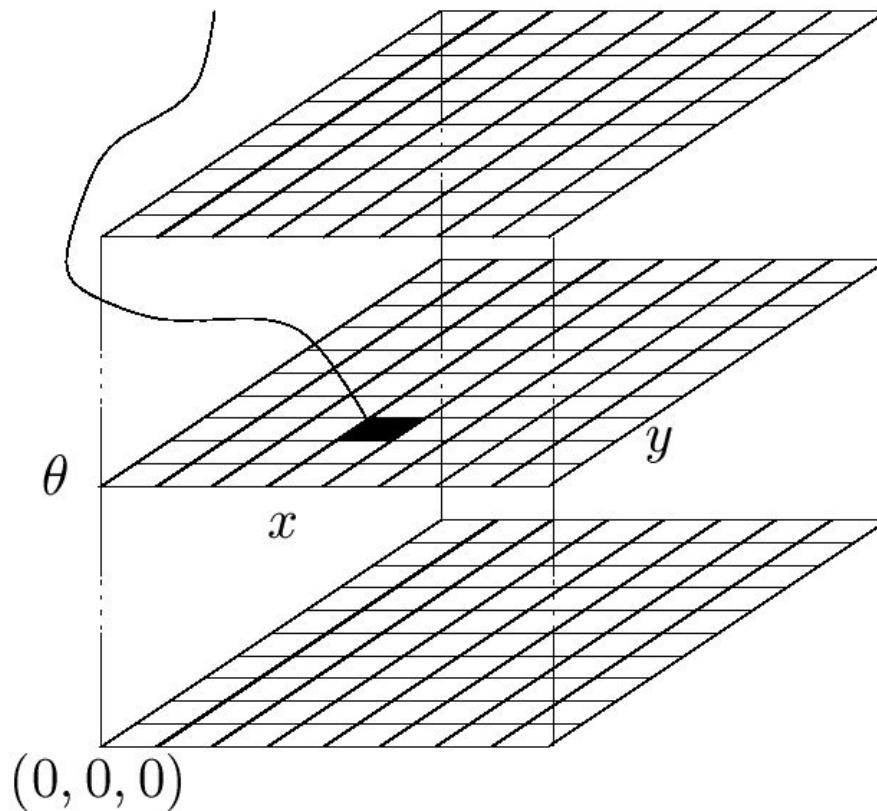
1. Algorithm **Bayes_filter**($Bel(x), d$):
2. $\eta = 0$
3. If d is a **perceptual** data item z then
 4. For all x do
 5. $Bel'(x) = P(z | x)Bel(x)$
 6. $\eta = \eta + Bel'(x)$
 7. For all x do
 8. $Bel'(x) = \eta^{-1}Bel'(x)$
 9. Else if d is an **action** data item u then
 10. For all x do
 11. $Bel'(x)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

sum over all
discrete states

motion model, Ch. 4

Piecewise Constant Representation

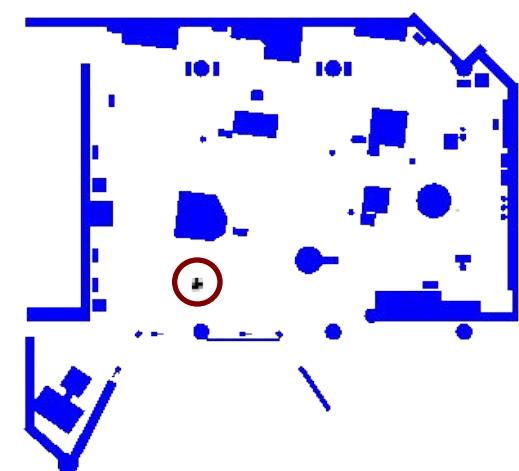
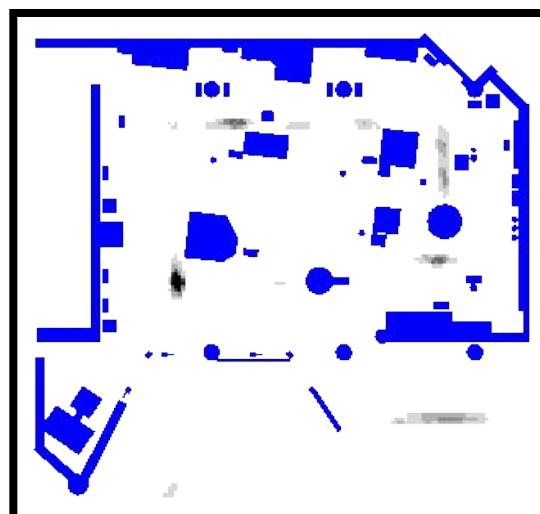
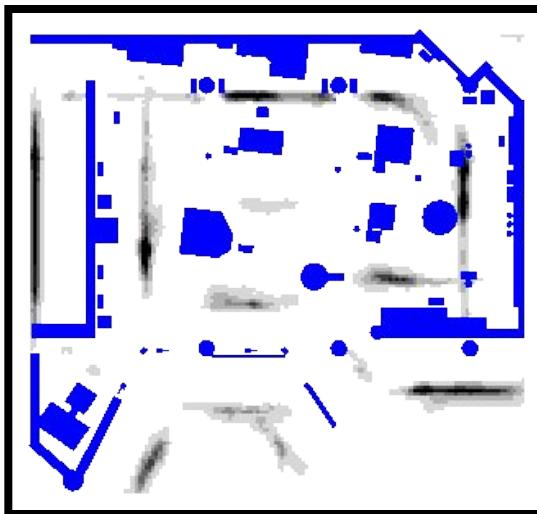
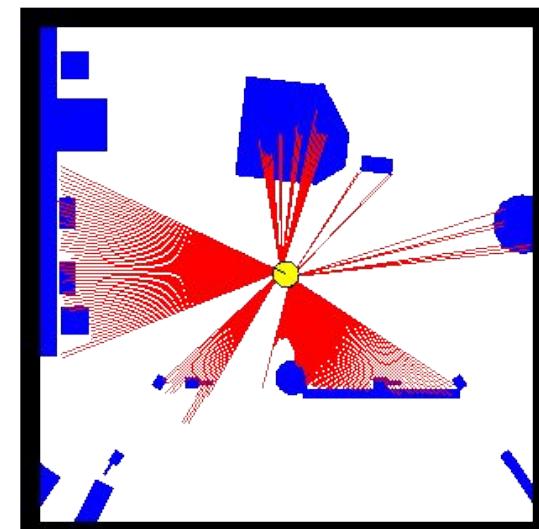
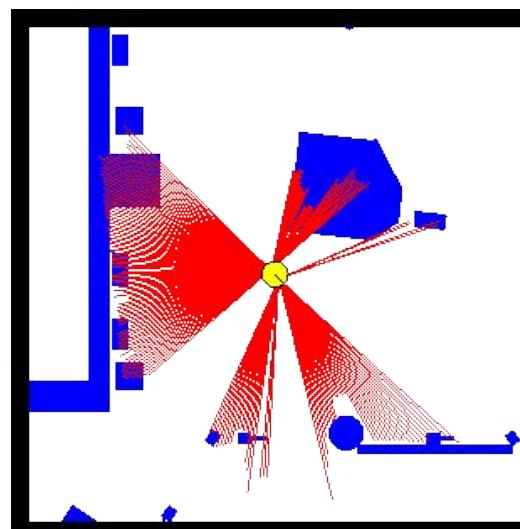
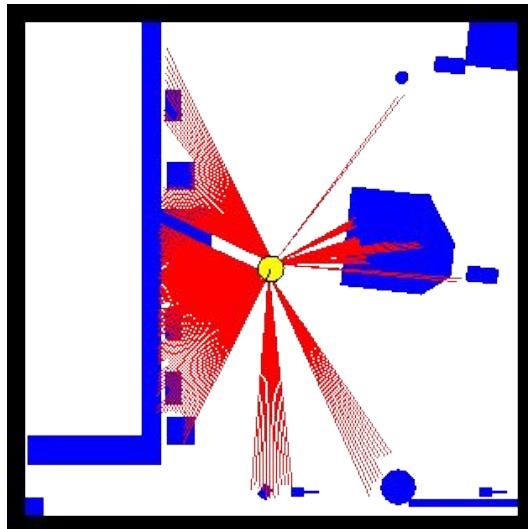
$Bel(x_t = \langle x, y, \theta \rangle)$



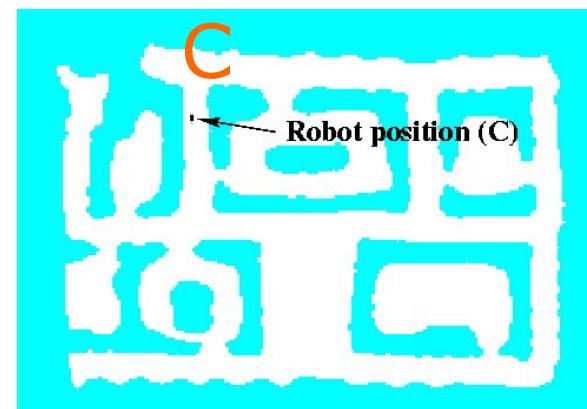
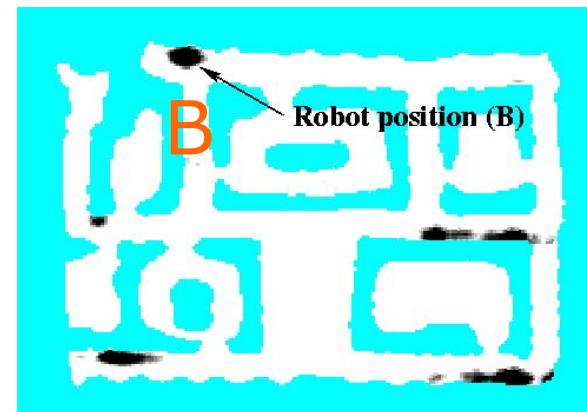
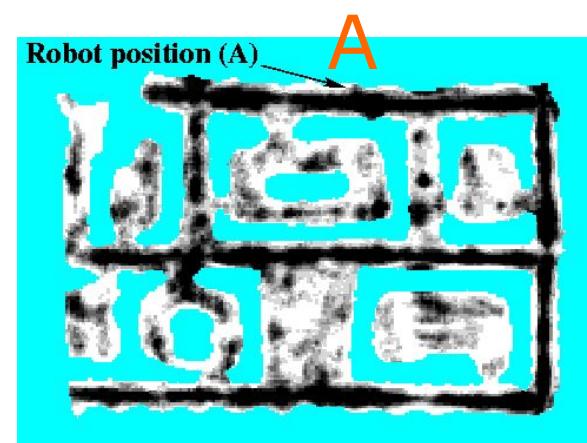
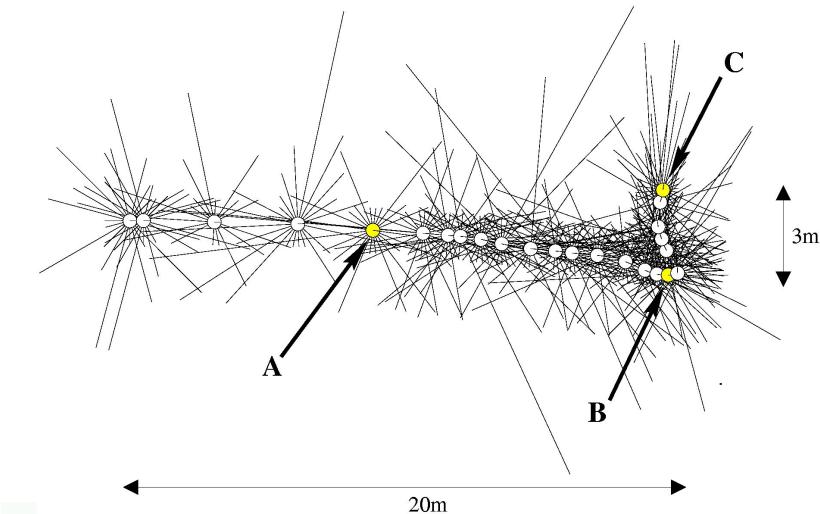
Implementation

- 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

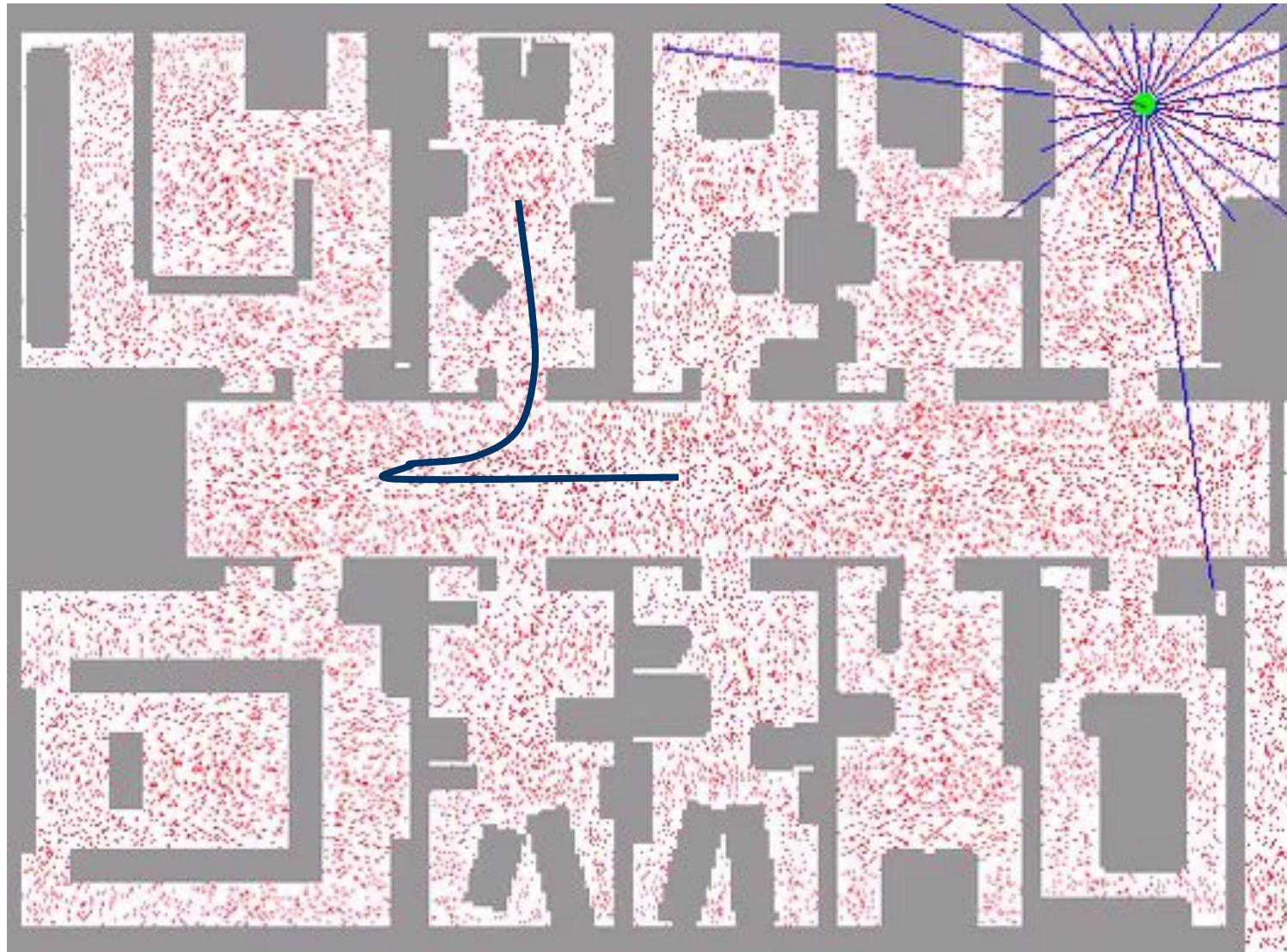
- Discrete filters are an alternative way for implementing the Bayes Filter
- Histograms for representing the density
- Can represent multi-modal beliefs and recover from localization errors
- Huge memory and processing requirements
- Accuracy depends on the resolution of the grid
- In practice: approximations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

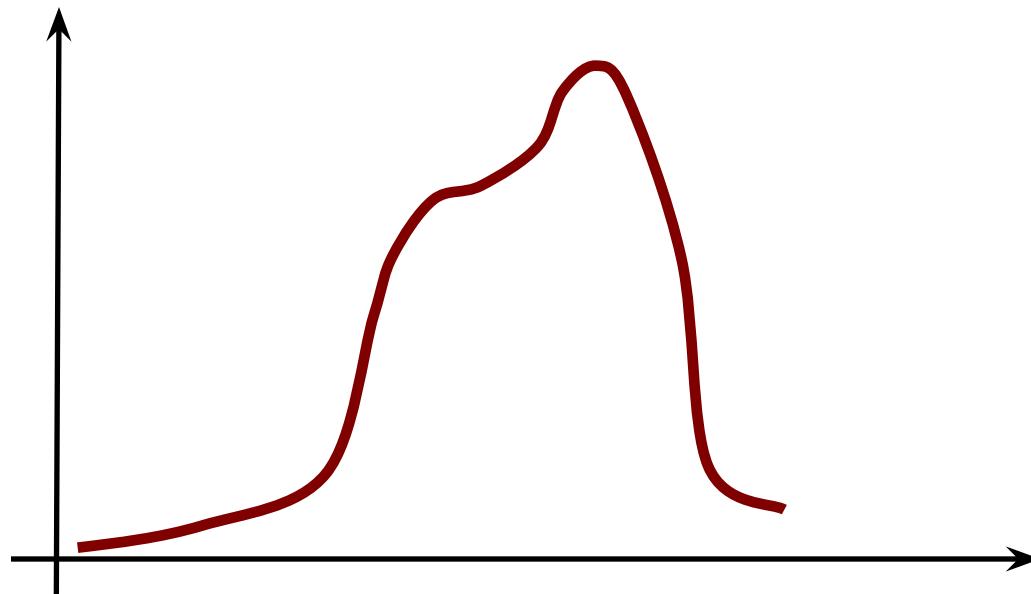
fl awl bnfrd en el robot mwgod fe kol el amakn el fl denya, w b3den ana kol el a3rfo 3n el robot, el motion wl sensor bs, fa lama by7rk le omad 3 5twat msln, b7rk kol el particles 3 khtwat le odam, w elly by2ro y7afzo 3la el khtwat de, bykmlo m3aya, wl mby2drosh, by7slhom vanishing.

Example: Sample-Based Localization (Sonar)



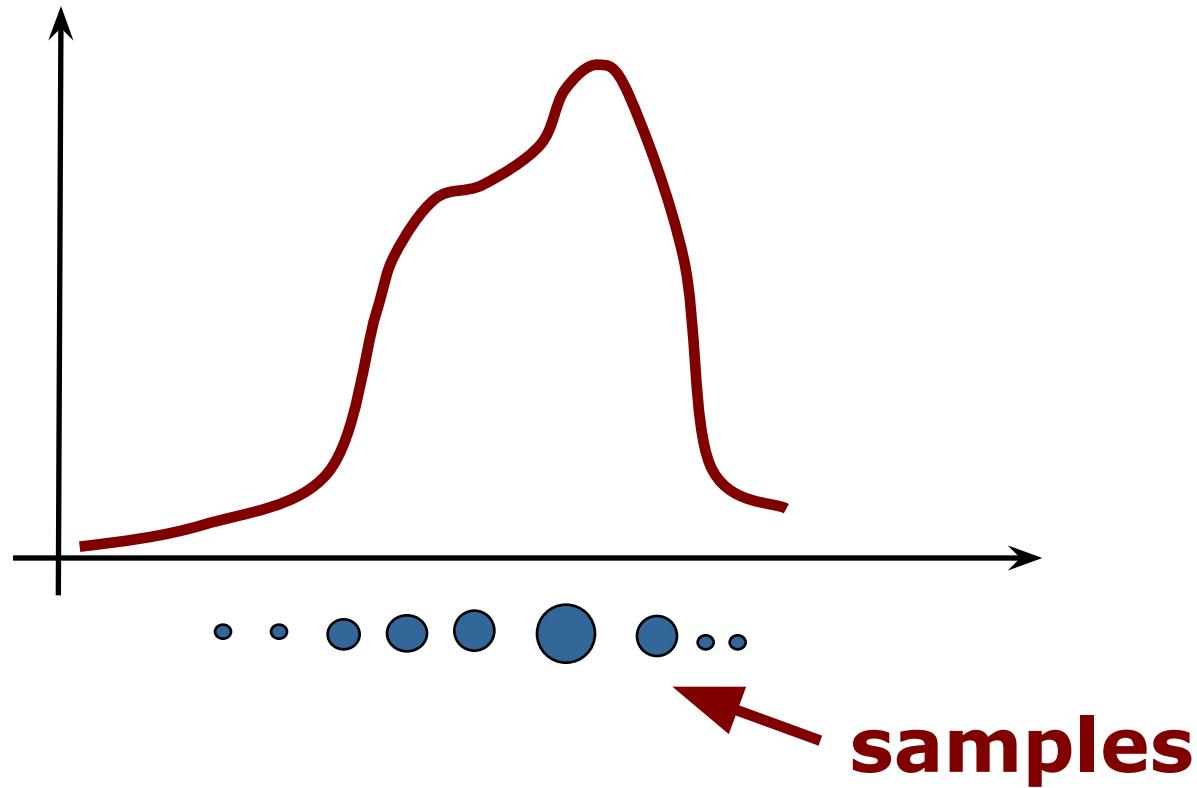
Motivation

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



Key Idea: Samples

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



Particle Set

- Set of weighted samples

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

state hypothesis **importance weight**

- The samples represent the posterior

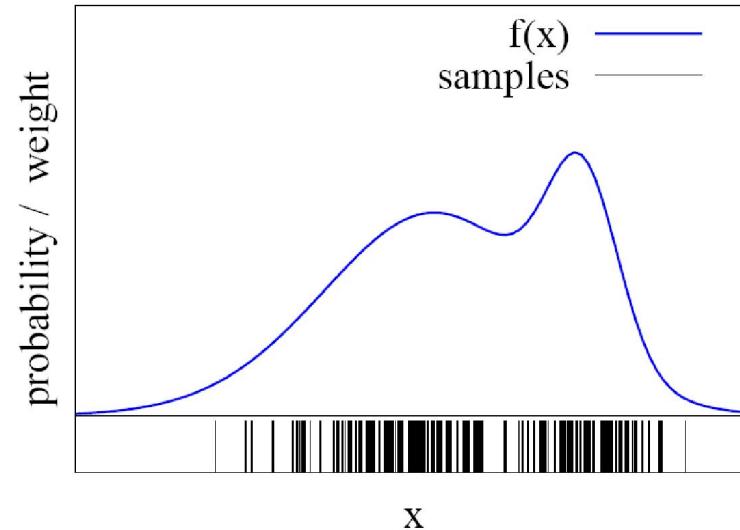
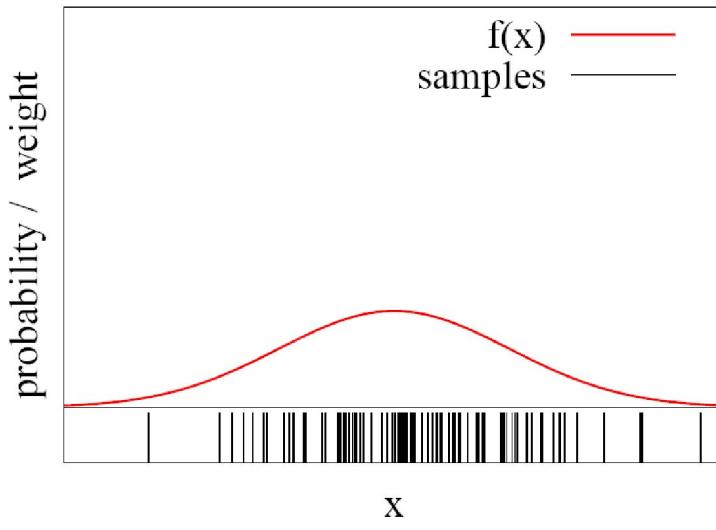
$$p(x) = \sum_{j=1}^J w^{[j]} \delta_{x^{[j]}}(x)$$

delta function, 1 3nd x mo3yna, w 0 otherwise.

Iw 3auzen n3ml tolerance momken na5ud el 3nd x w shuya mn elly 7waleh.

Particles for Approximation

- Particles for function approximation

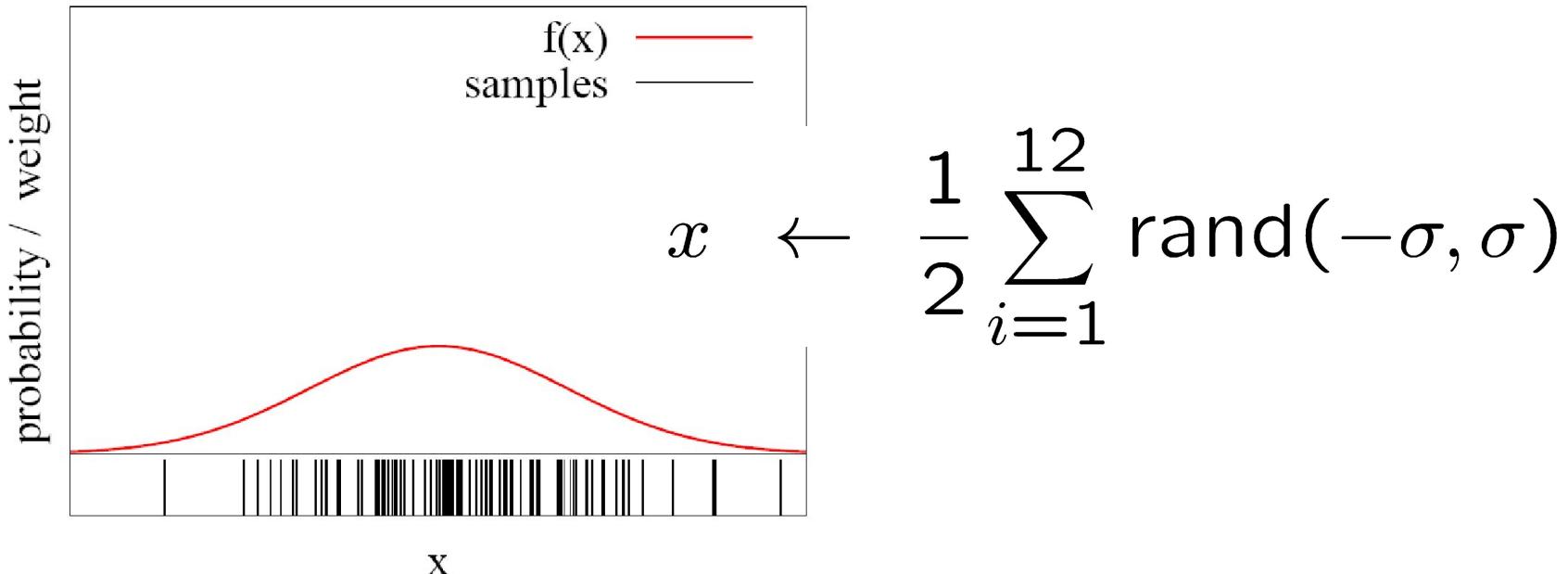


- The more particles fall into a region, the higher the probability of the region

How to obtain such samples?

Closed Form Sampling is Only Possible for Few Distributions

- Example: Gaussian

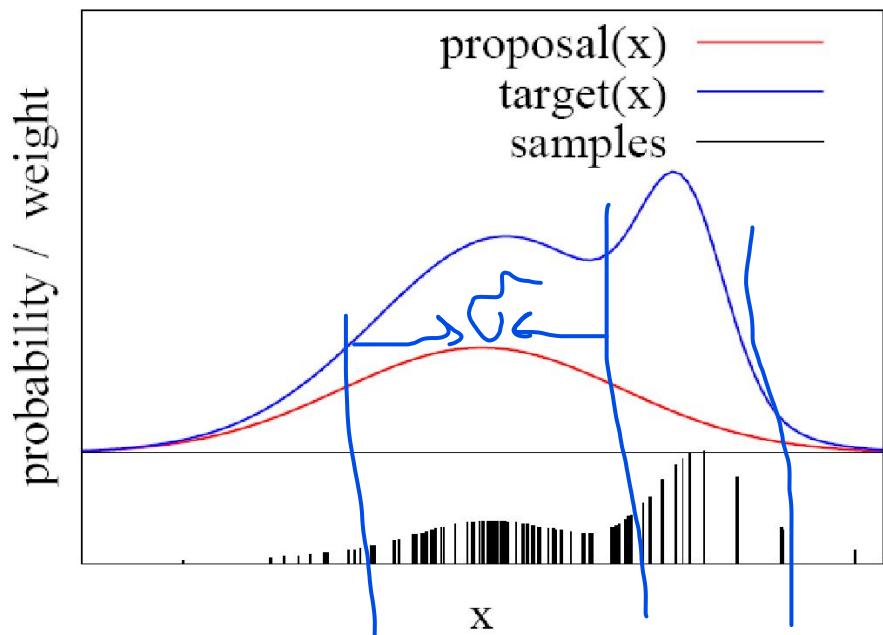


- How to sample from **other** distributions?

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

el sampling da aknk btgeb el x
w b3d keda btgeb el x de t3wd
beha fl functions 34an tgeb
el y fe ay function mnhom
baa.

$$x_t^{[j]} \sim proposal(x_t | \dots)$$

2. Compute the importance weights

3ndna 3 5twat,
1. bn3ml sampling mn el porposal
distribution.
2. bn7sb el weights 3n tre2 enna
ne2sm el value mn el target
distribution bon2an 3la el x elly
3mltlha sampling, 3la el value
mn el proposal($x_t^{[j]}$).
w b3d keda bn3ml resampling, b
probability $wt[i]$

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

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

kol ma tktr 3dd el particles, kol ma tzwd el computations.

I particle n2dr nnfzo 3la el 7agat el low speed, zy el case bta3tna.

Iw mashy fl shar3 w 3auz 7aga bt3ml rapid response, msh hynf3 tst5dmo.

Monte Carlo Localization

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

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

ezay wslt l hena ml fo2ha, aw mn el slide elly fatet.

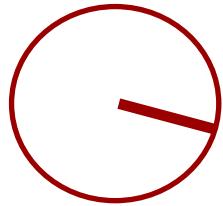
eh 3lakt da bl slide elly fatet -> do it yourself, 34an momken ytl3 fl emt7an.

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

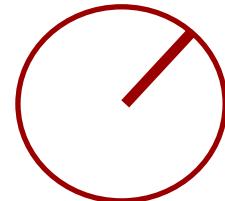
el so2al ezay wsltna lel special case elly fl slide de, mn el general case

Reminder: Odometry Motion Model

start pose

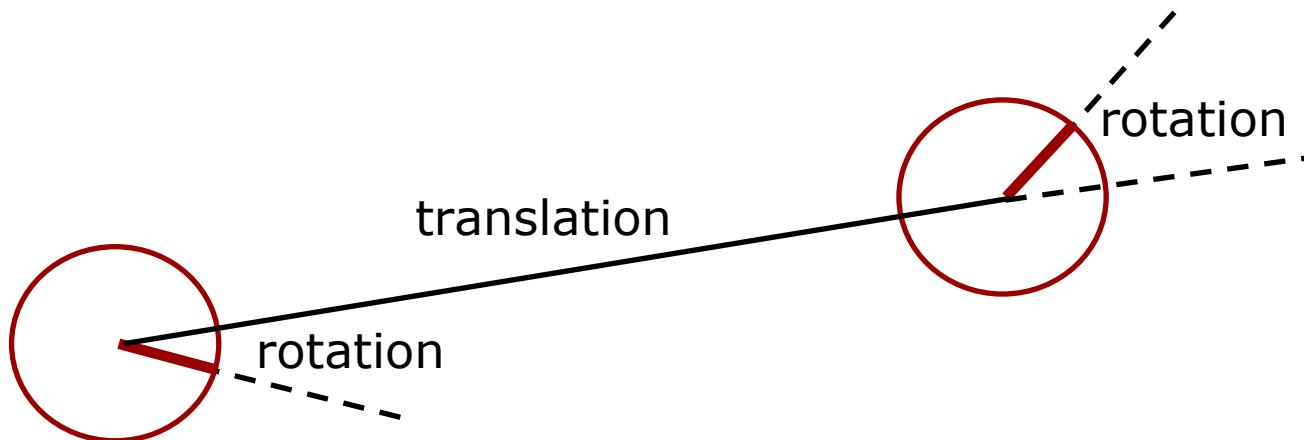


end pose



According to the estimated motion

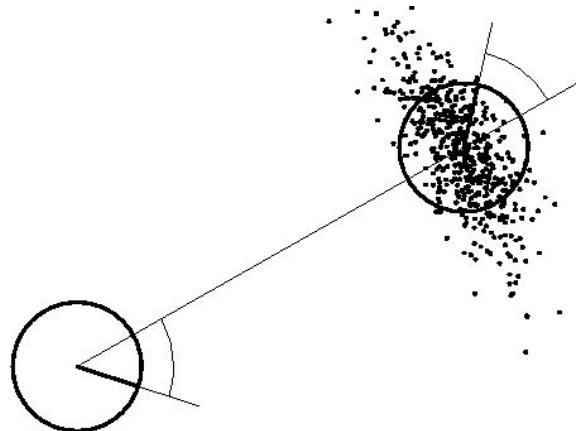
Reminder: Odometry Motion Model



Decompose the motion into

- Traveled distance
- Start rotation
- End rotation

Reminder: Odometry Motion Model

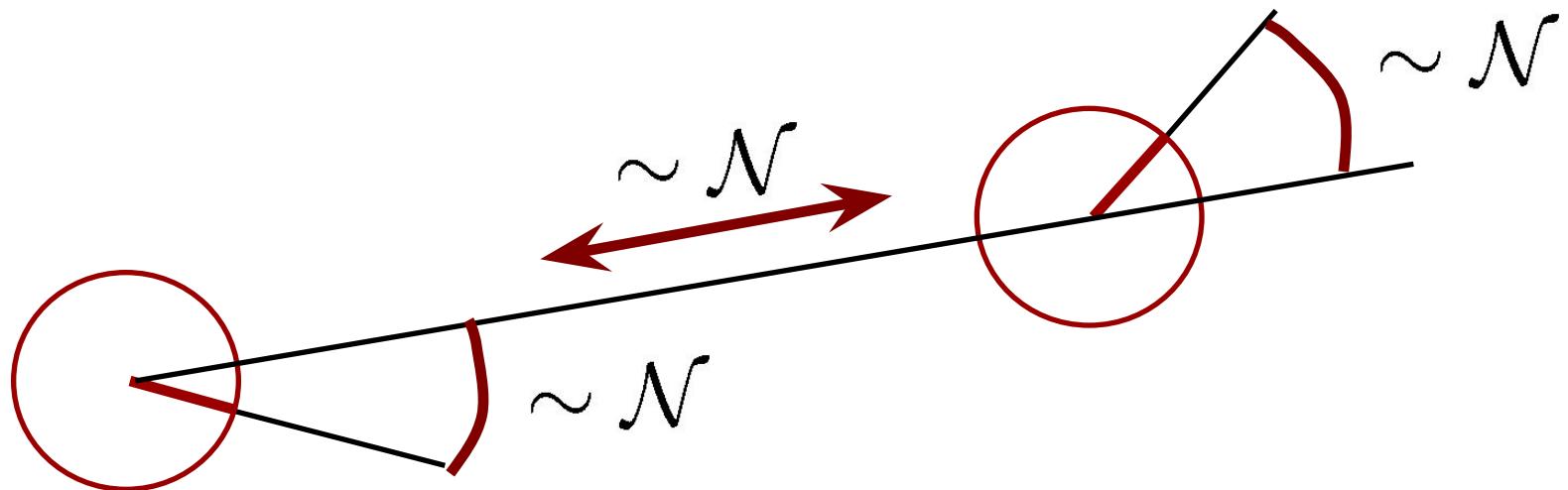


- Uncertainty in the translation of the robot:
Gaussian over the traveled distance
- Uncertainty in the rotation of the robot:
Gaussians over start and end rotation
- **For each particle, draw a new pose by sampling from three normal distributions**

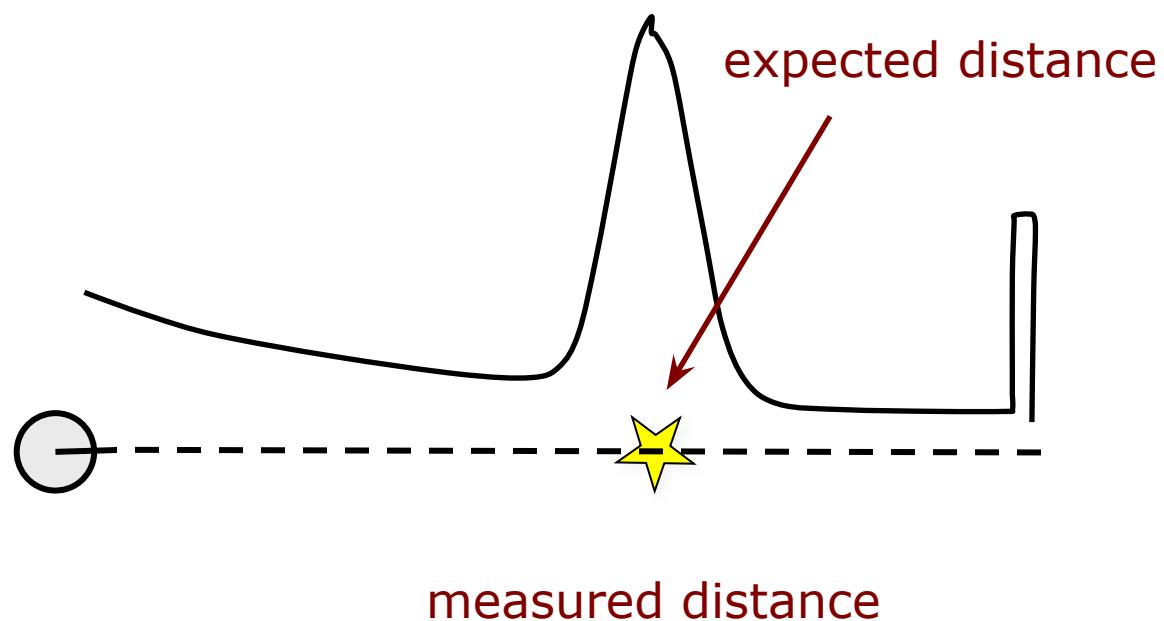
Reminder: Odometry Motion Model

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

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



Reminder: Proximity Sensor Model

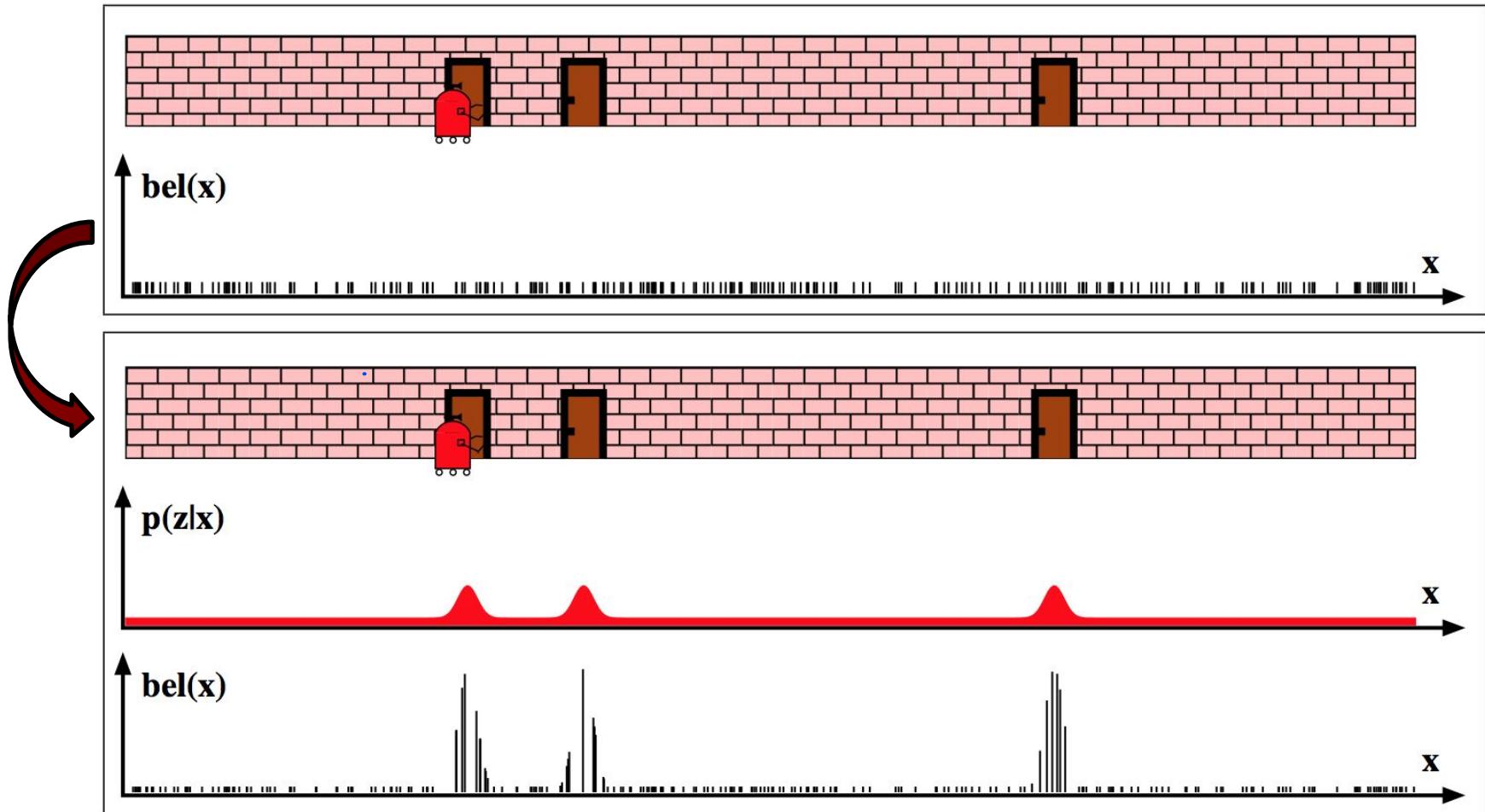


Particle Filter for Localization

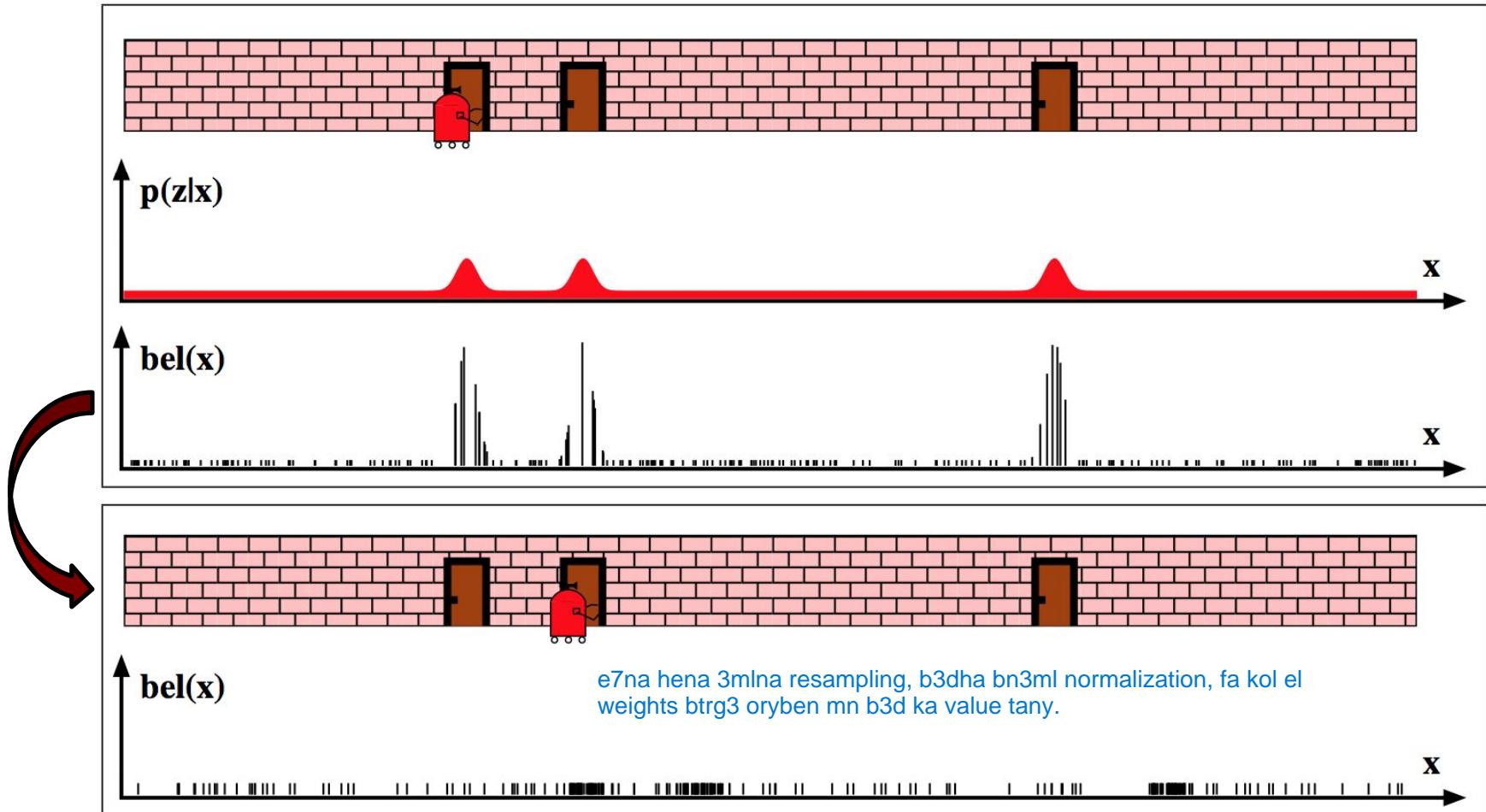
Particle_filter($\mathcal{X}_{t-1}, u_t, z_t$):

```
1:    $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$ 
2:   for  $j = 1$  to  $J$  do
3:     sample  $x_t^{[j]} \sim p(x_t \mid u_t, x_{t-1}^{[j]})$  bnfz 3leha el action  
bta3 el model bta3y/
4:      $w_t^{[j]} = p(z_t \mid x_t^{[j]})$ 
5:      $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[j]}, w_t^{[j]} \rangle$  bnzwdha 3la el set bt3ty;
6:   endfor
7:   for  $j = 1$  to  $J$  do
8:     draw  $i \in 1, \dots, J$  with probability  $\propto w_t^{[i]}$ 
9:     add  $x_t^{[i]}$  to  $\mathcal{X}_t$ 
10:    endfor
11:    return  $\mathcal{X}_t$ 
```

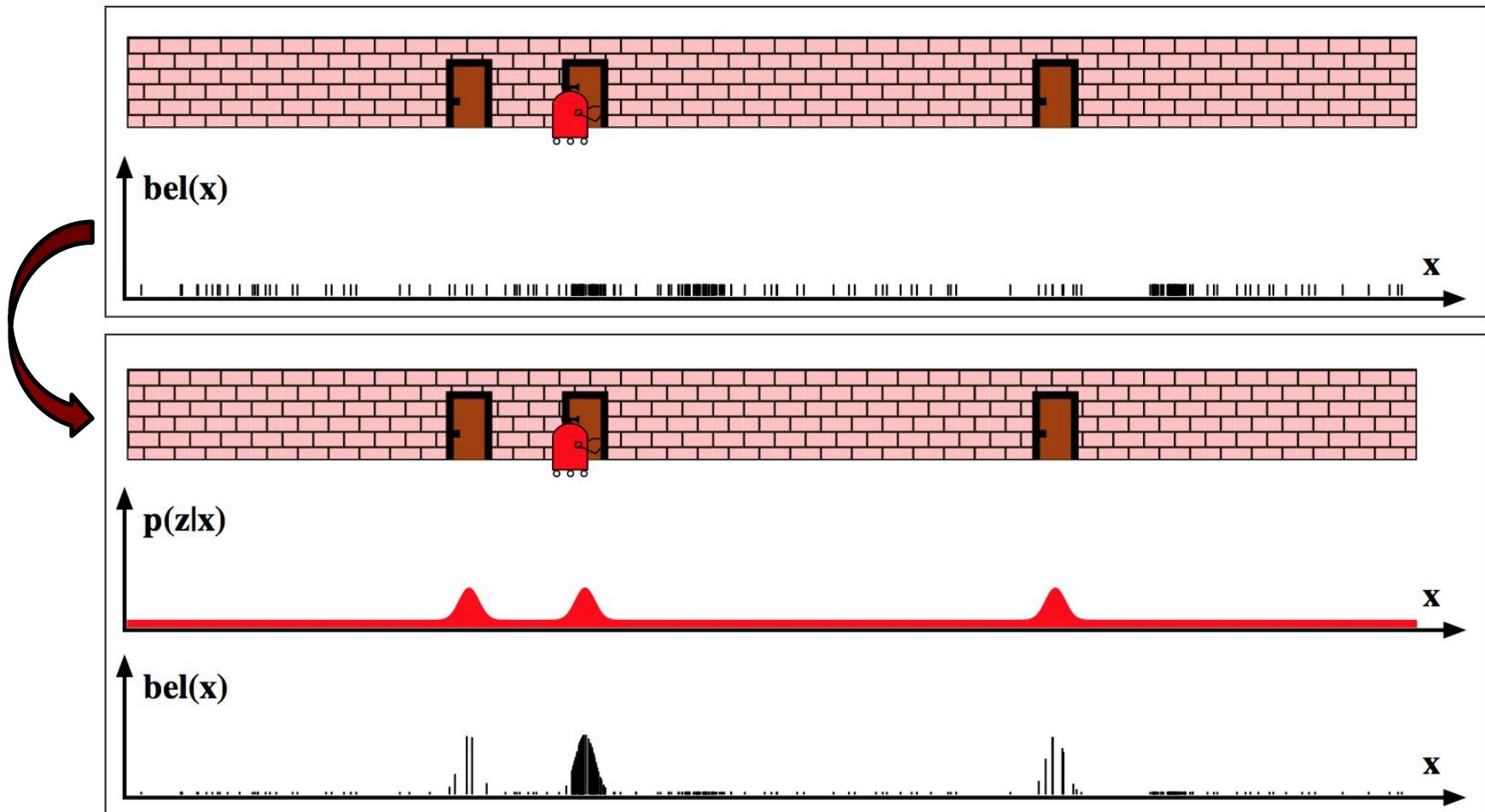
MCL – Correction Step



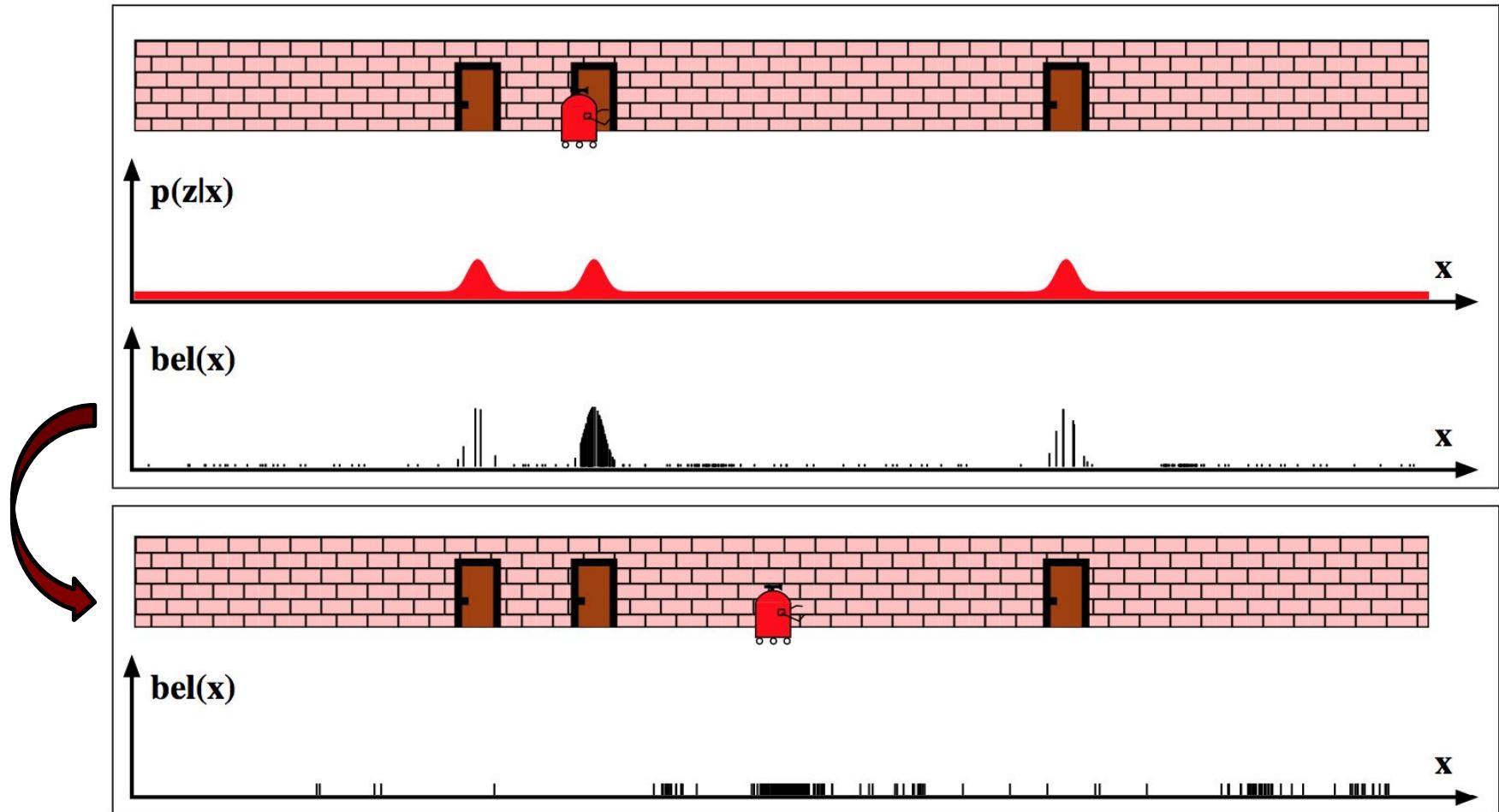
MCL – Prediction Step



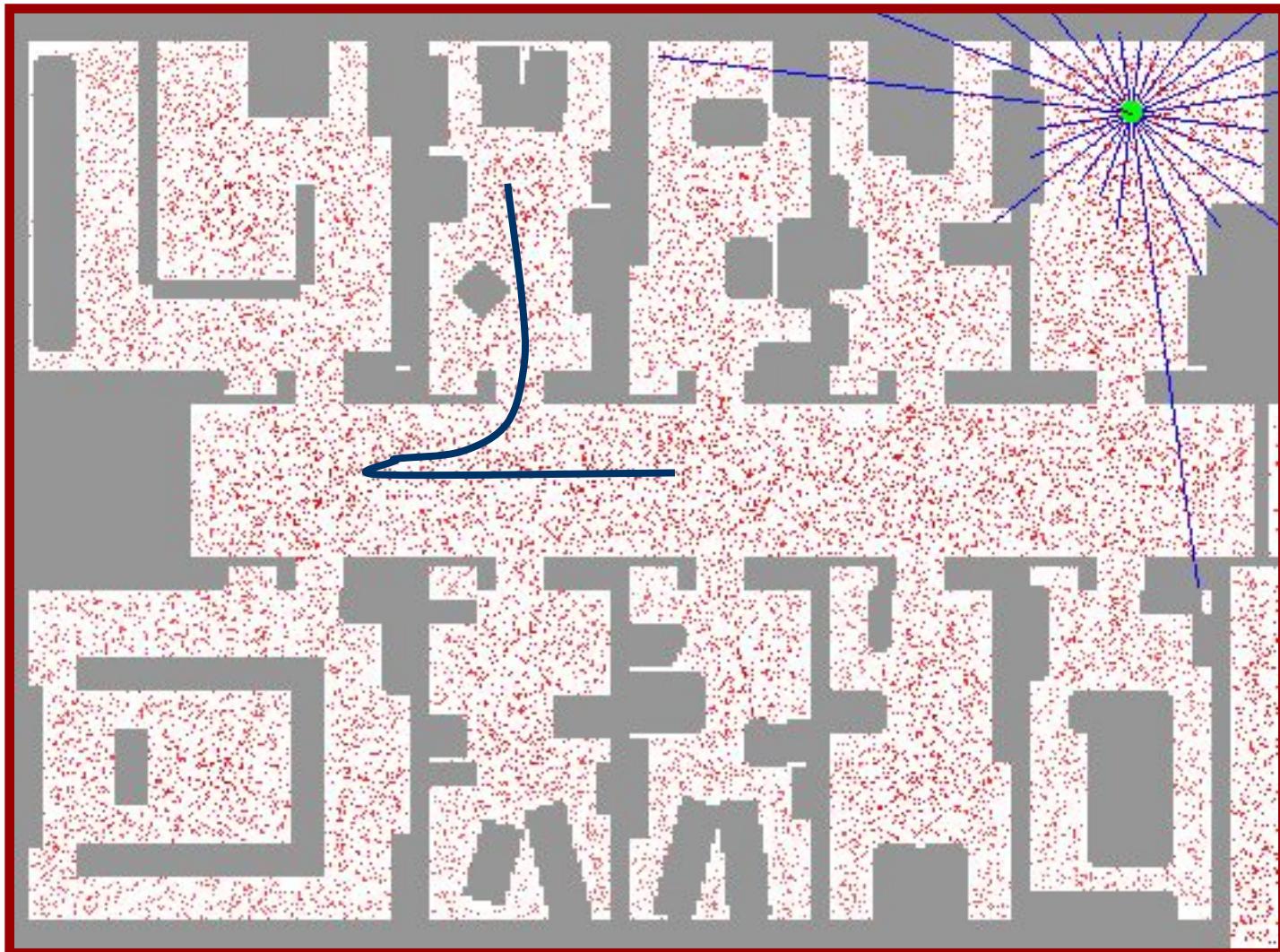
MCL – Correction Step



MCL – Prediction Step



Example: Sample-Based Localization (Sonar)



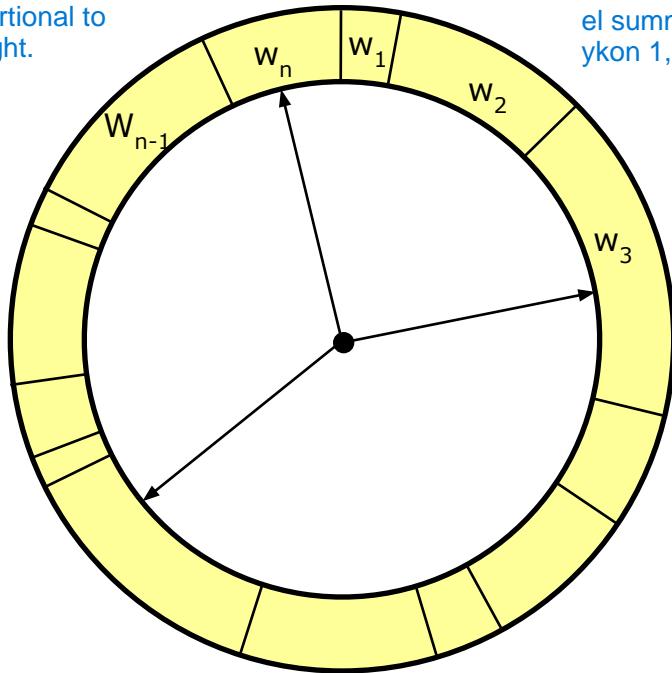
Resampling

- 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

Resampling

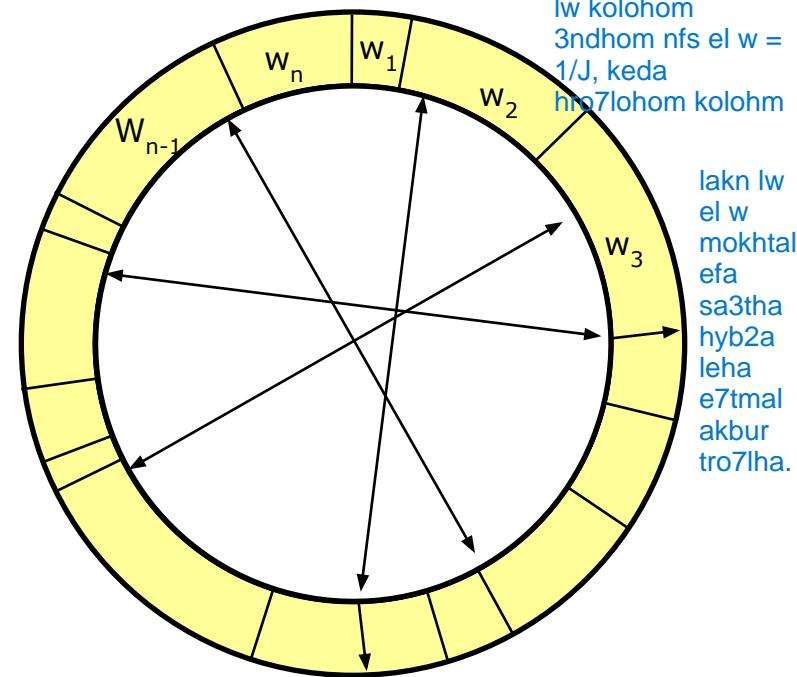
“roulette wheel”

el pocket size
is proportional to
the weight.



el summation el mfrd
ykon 1, le kol el weights.

initial value between 0 and $1/J$
step size = $1/J$



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

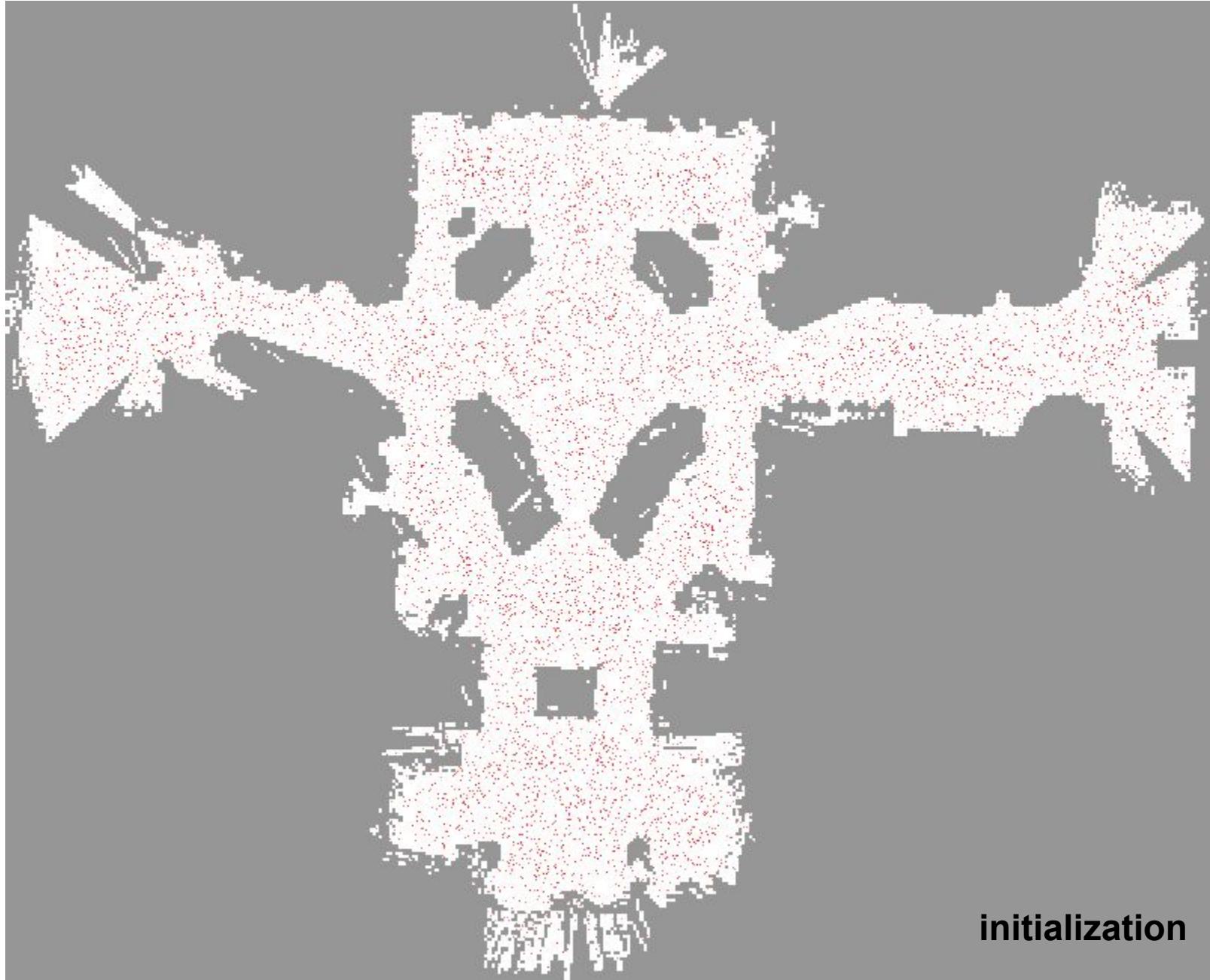
BS 34an homa sorted.

- Systematic resampling
- Low variance resampl.
- $O(J)$
- Also called “stochastic universal resampling”

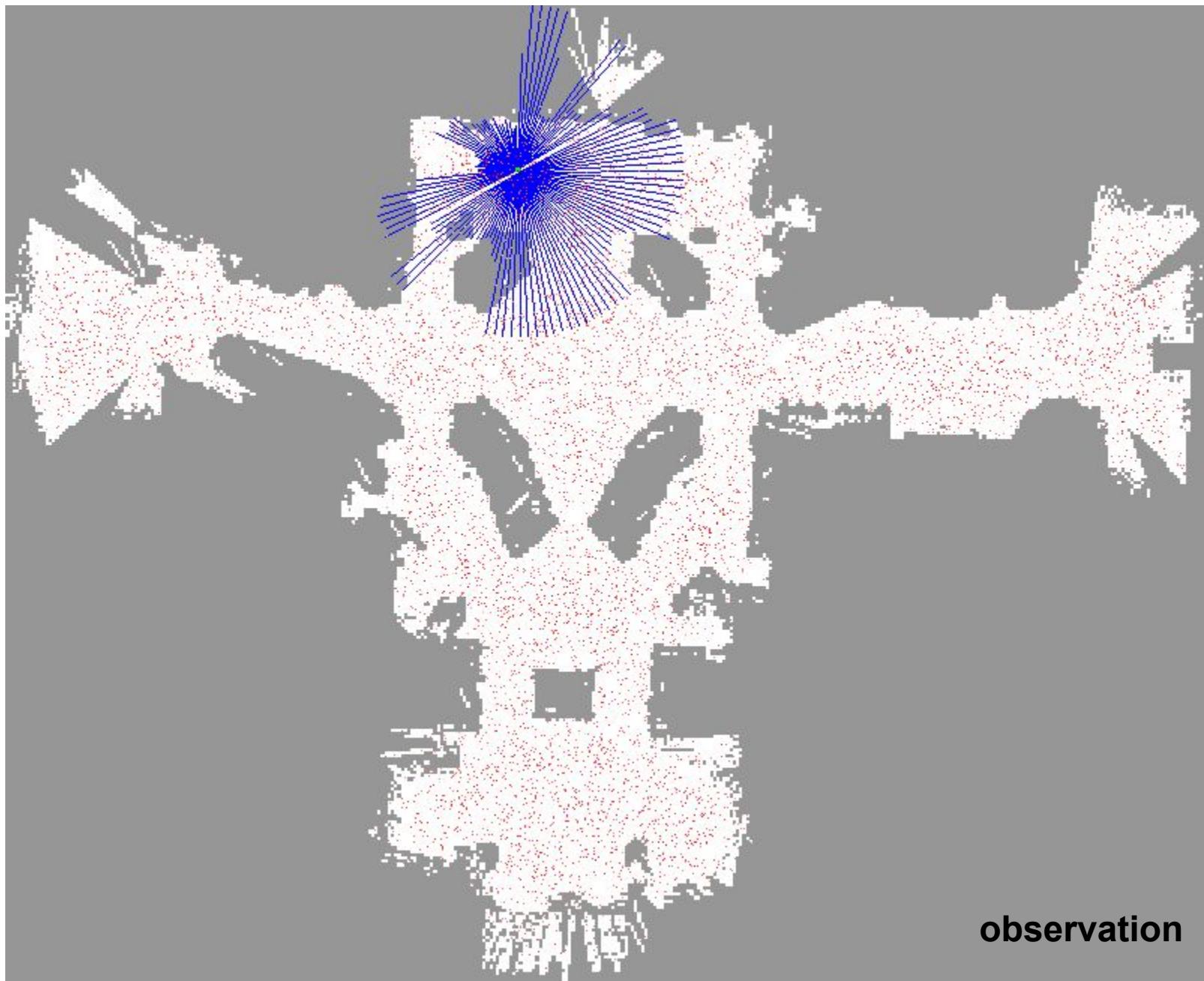
Low Variance Resampling

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

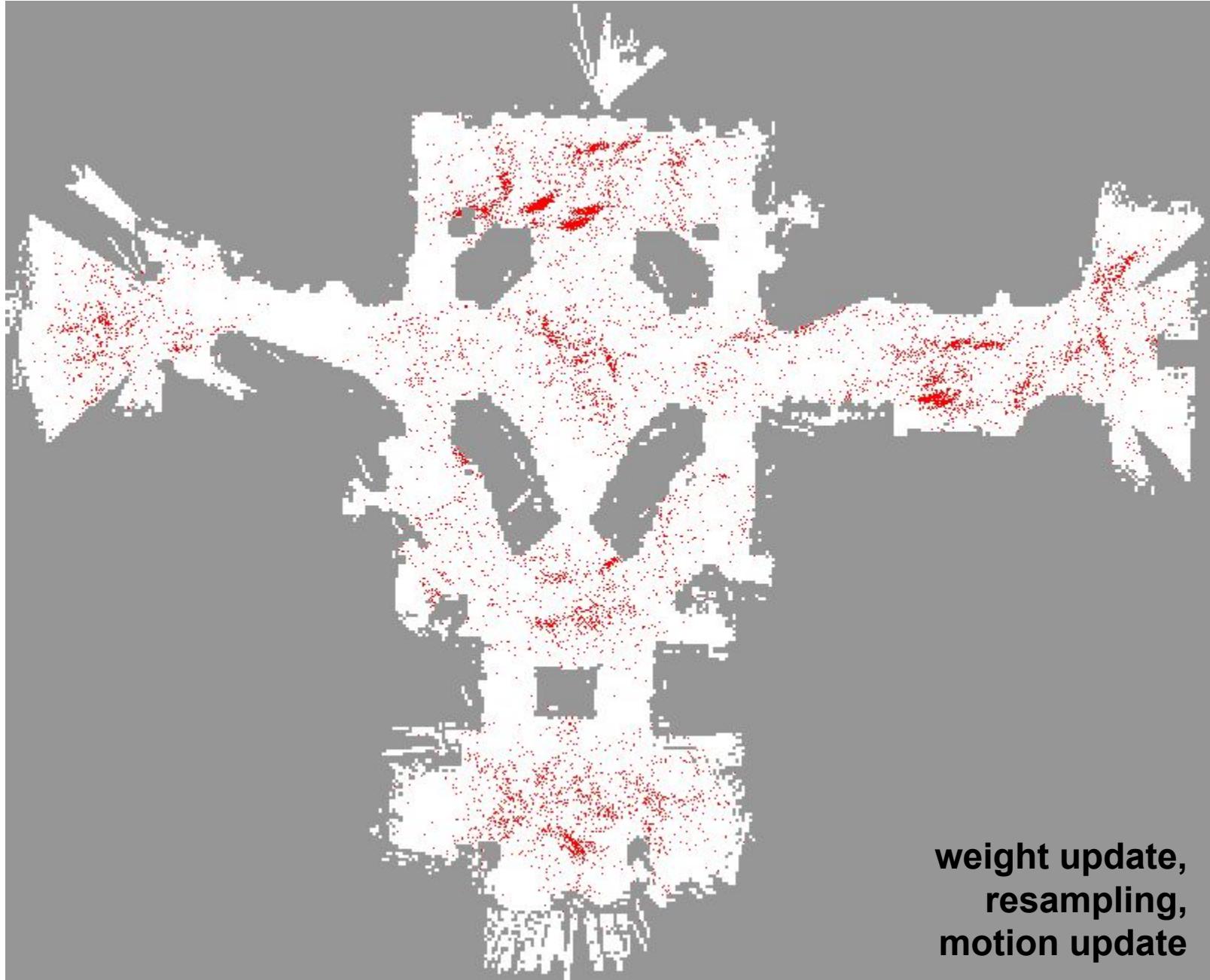
```
1:       $\bar{\mathcal{X}}_t = \emptyset$ 
2:       $r = \text{rand}(0; J^{-1})$  initialization
3:       $c = w_t^{[1]}$  cumulative sum of weights
4:       $i = 1$ 
5:      for  $j = 1$  to  $J$  do            $J = \#particles$ 
6:           $U = r + (j - 1)J^{-1}$  step size =  $1/J$ 
7:          while  $U > c$    decide whether or not
8:               $i = i + 1$     to take particle  $i$ 
9:               $c = c + w_t^{[i]}$ 
10:         endwhile
11:         add  $x_t^{[i]}$  to  $\bar{\mathcal{X}}_t$ 
12:     endfor
13:     return  $\bar{\mathcal{X}}_t$ 
```



Courtesy: Thrun, Burgard, Fox 36

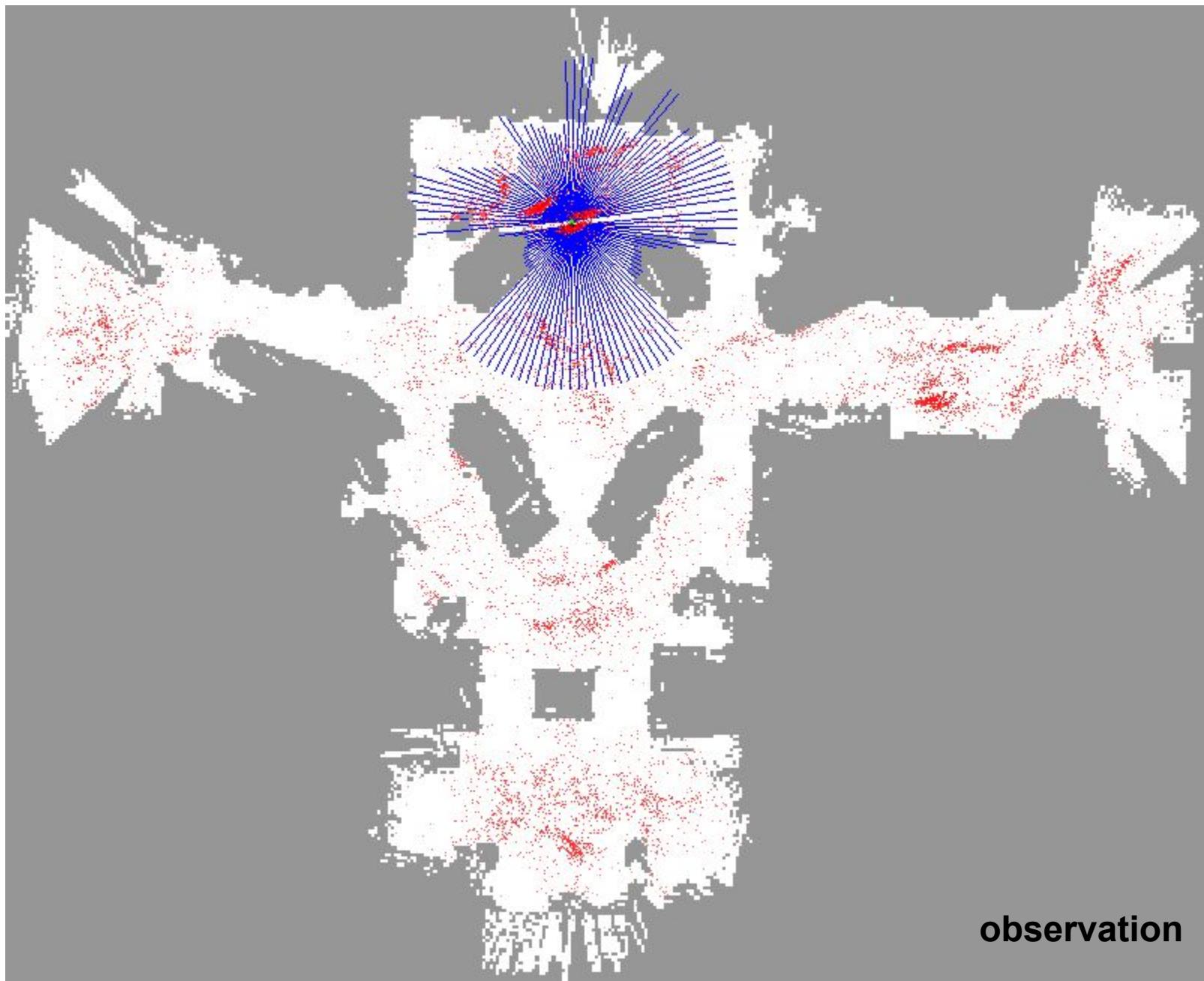


Courtesy: Thrun, Burgard, Fox 37



**weight update,
resampling,
motion update**

Courtesy: Thrun, Burgard, Fox 38

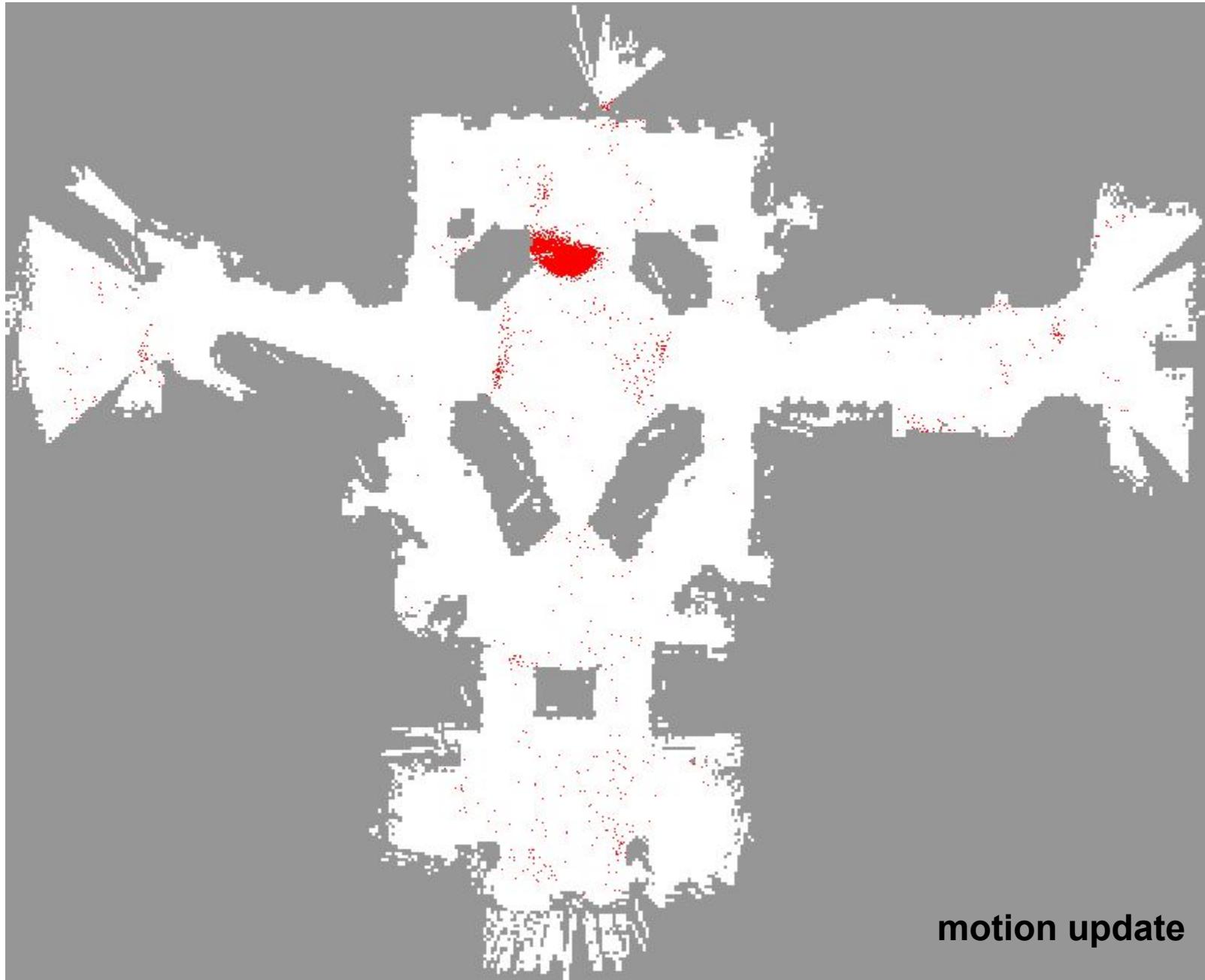


Courtesy: Thrun, Burgard, Fox 39

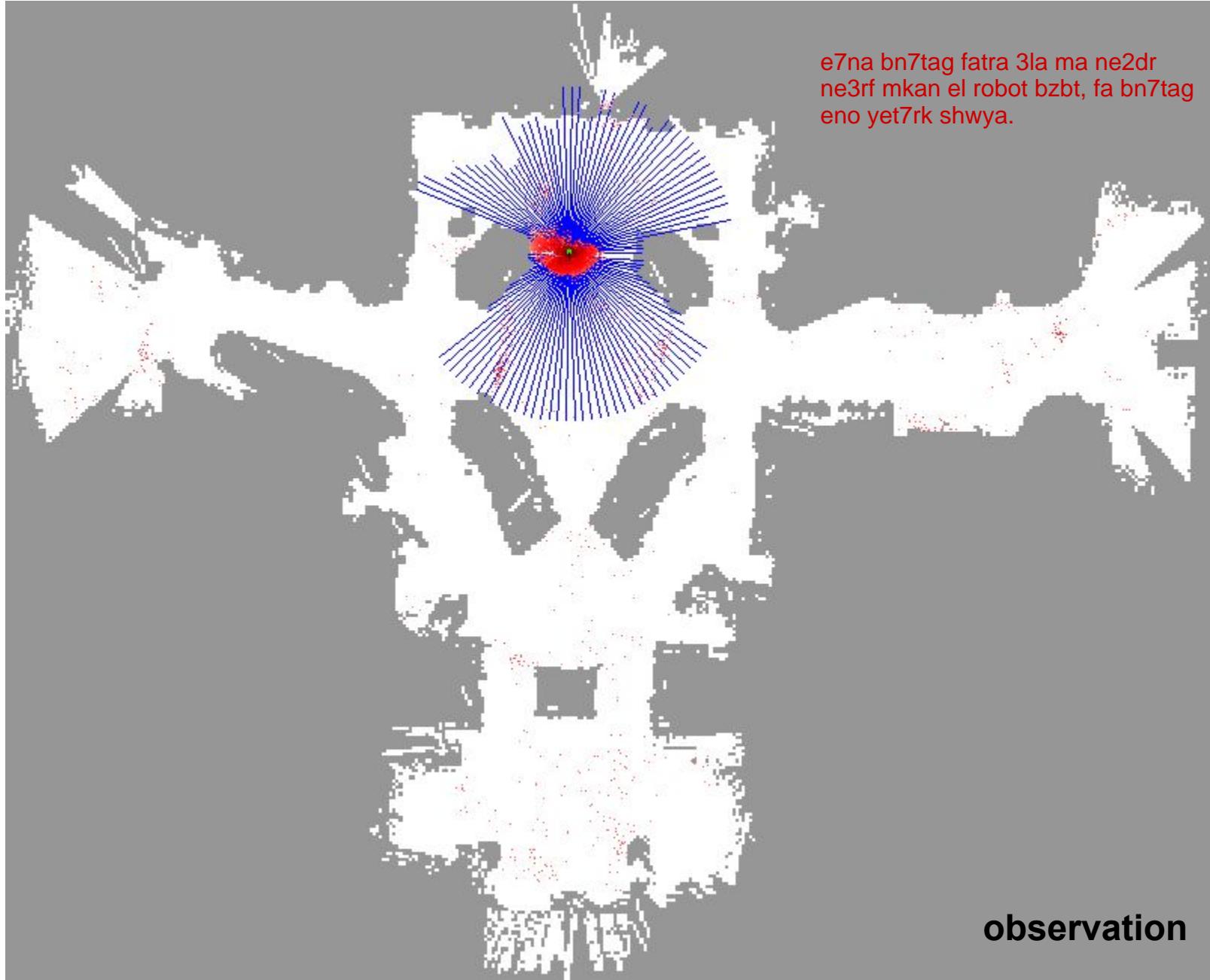


**weight update,
resampling**

Courtesy: Thrun, Burgard, Fox 40



Courtesy: Thrun, Burgard, Fox 41



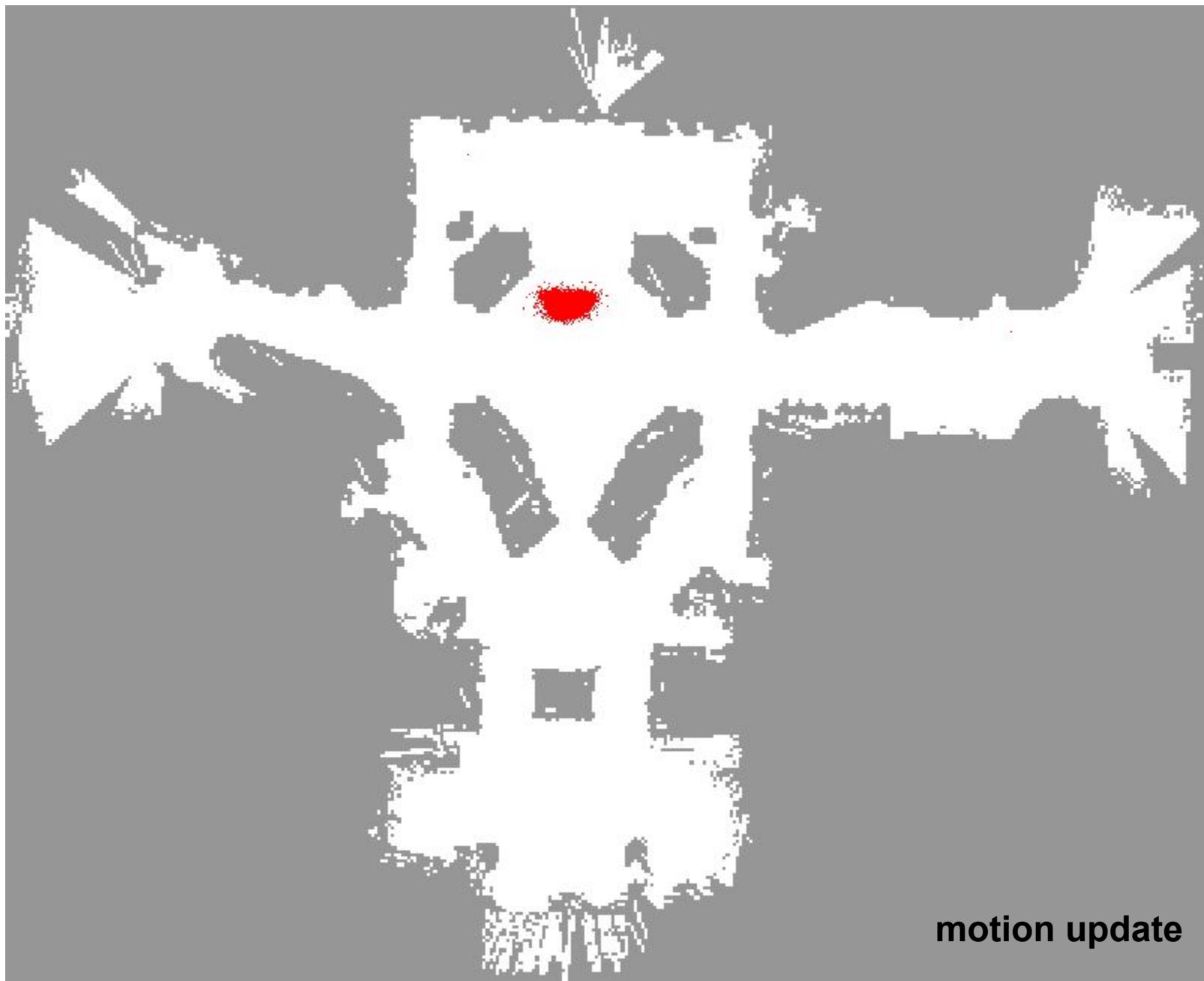
e7na bn7tag fatra 3la ma ne2dr
ne3rf mkan el robot bzbt, fa bn7tag
eno yet7rk shwya.

observation

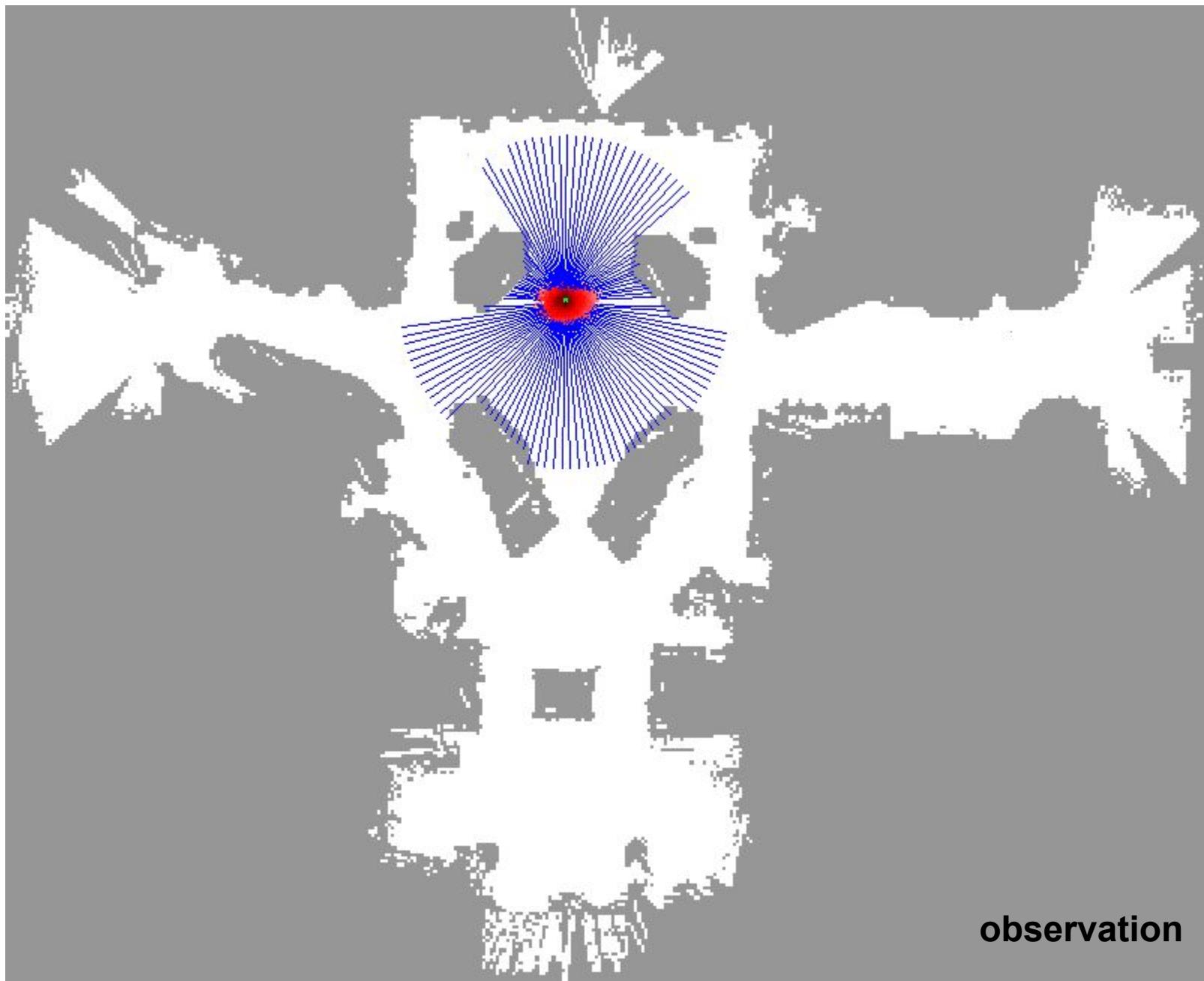
Courtesy: Thrun, Burgard, Fox 42



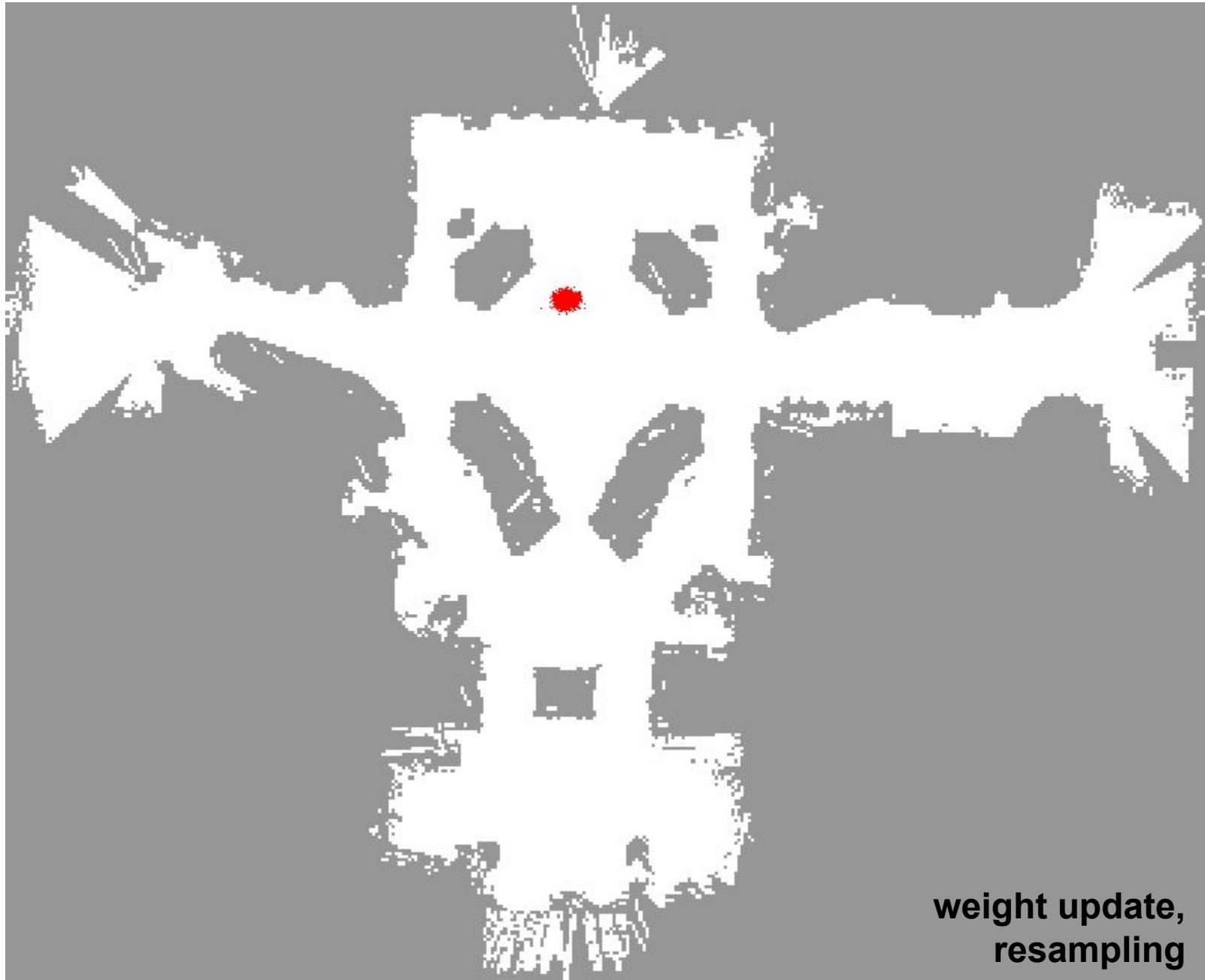
Courtesy: Thrun, Burgard, Fox 43



Courtesy: Thrun, Burgard, Fox 44

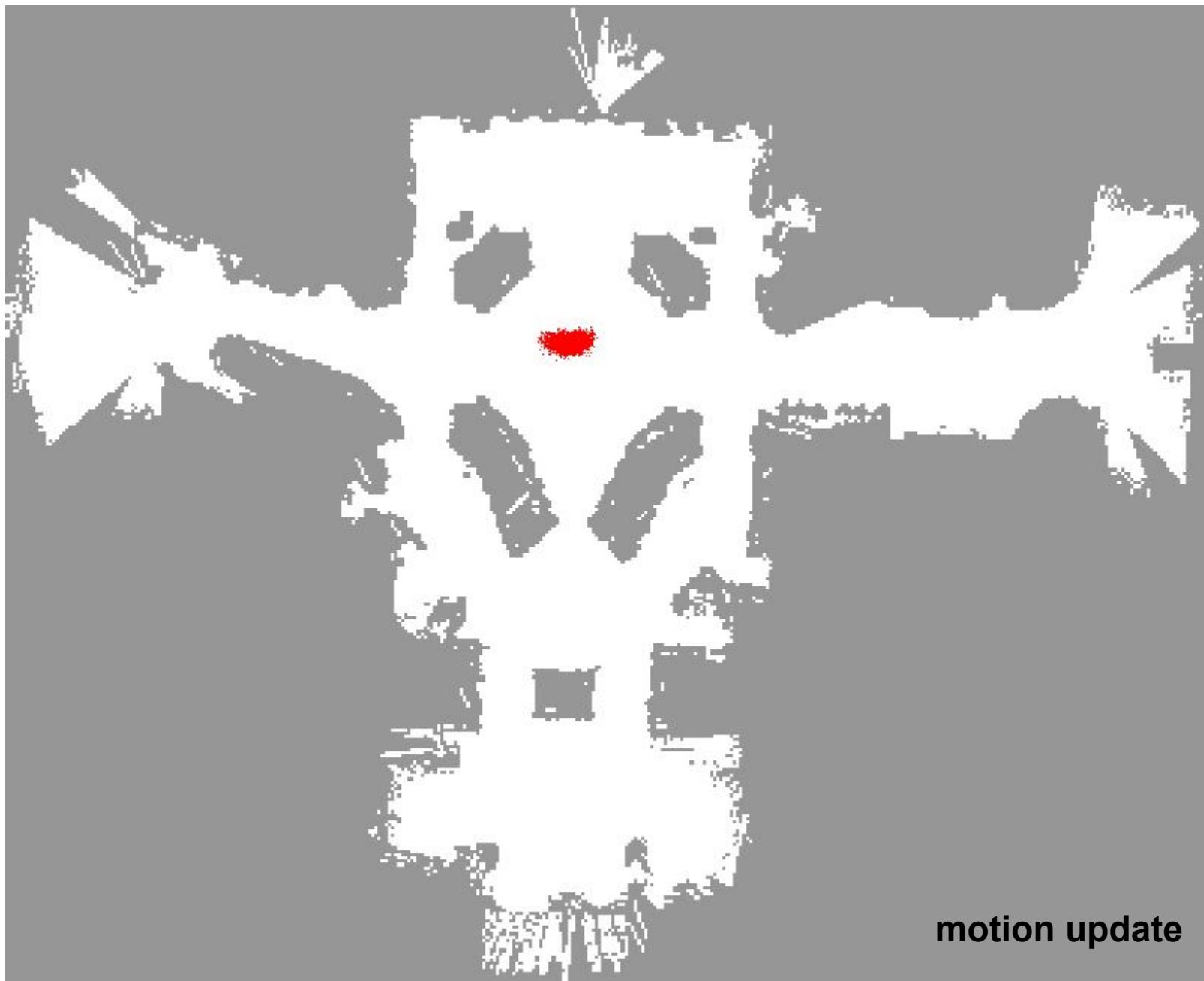


Courtesy: Thrun, Burgard, Fox 45



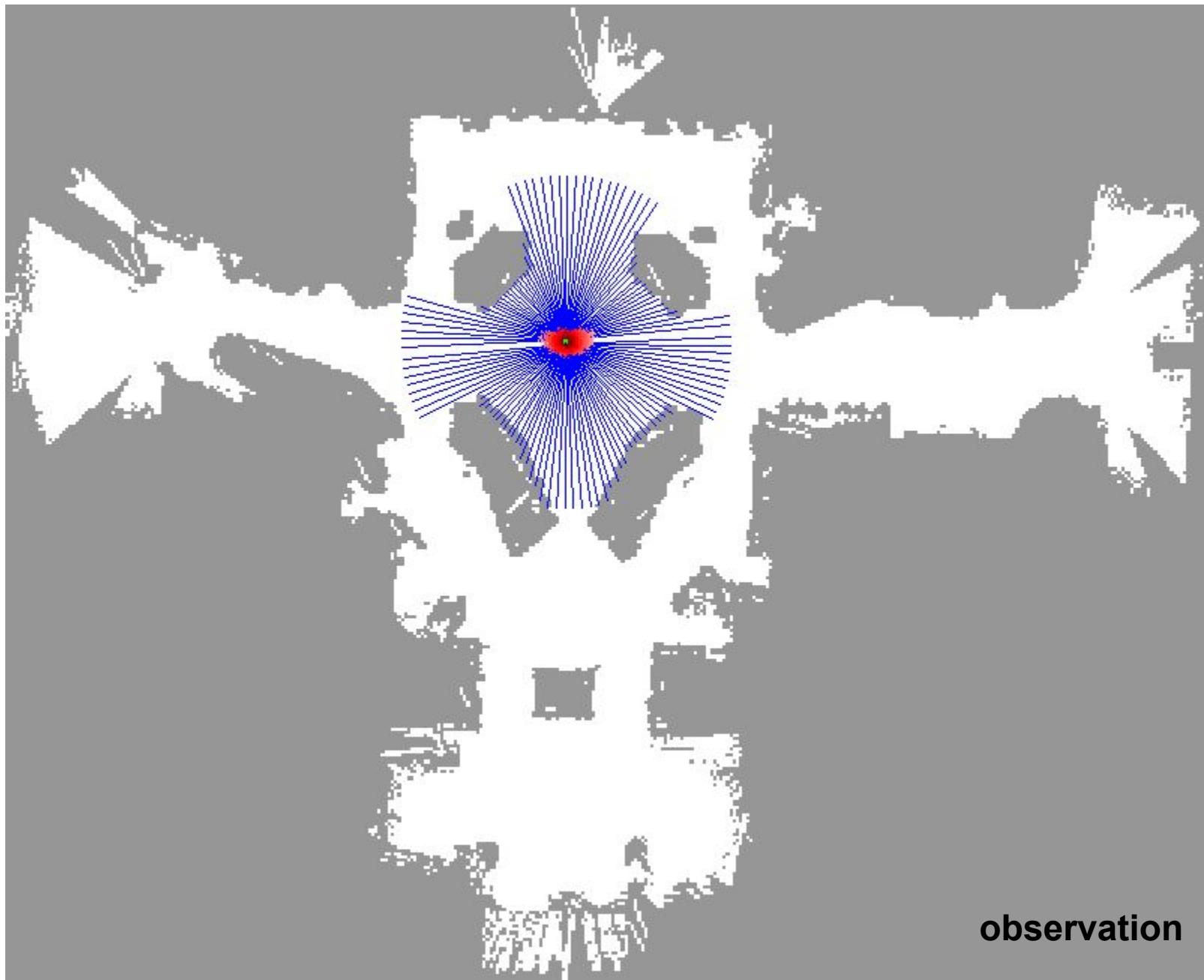
**weight update,
resampling**

Courtesy: Thrun, Burgard, Fox 46



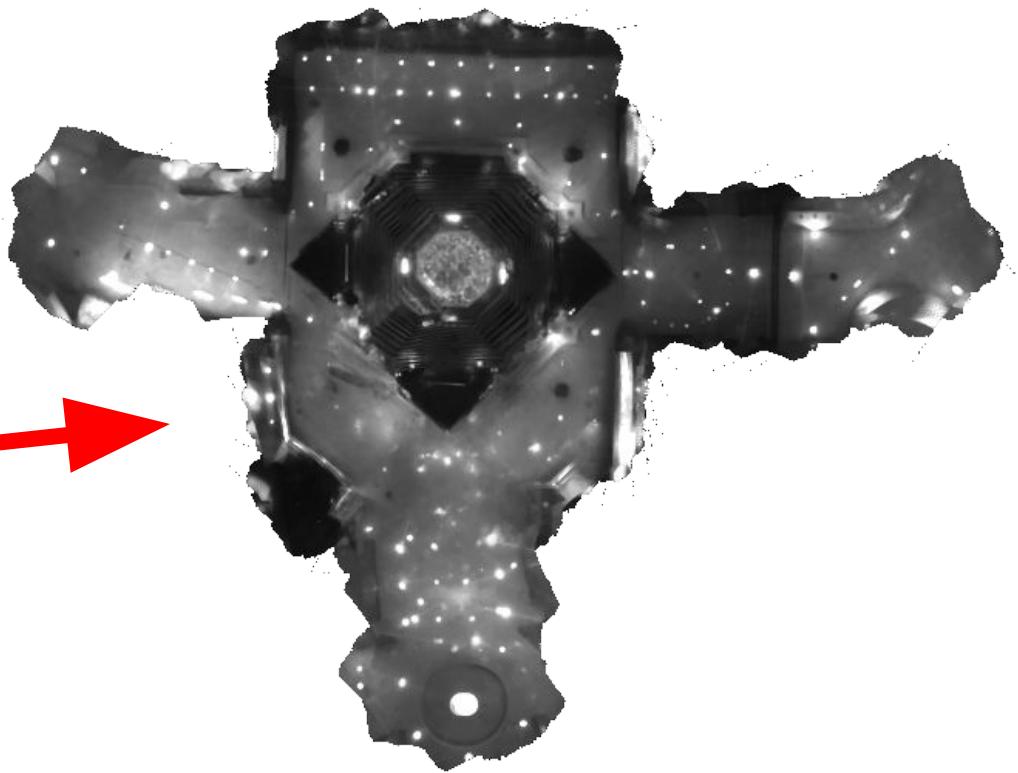
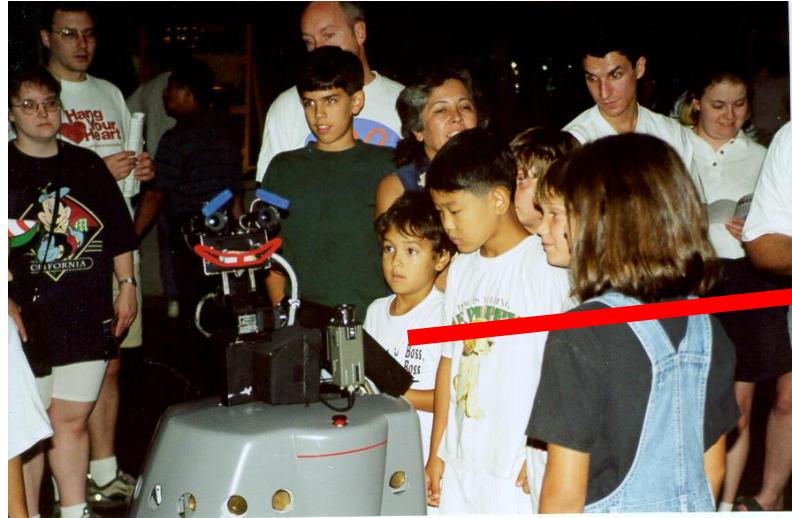
motion update

Courtesy: Thrun, Burgard, Fox 47



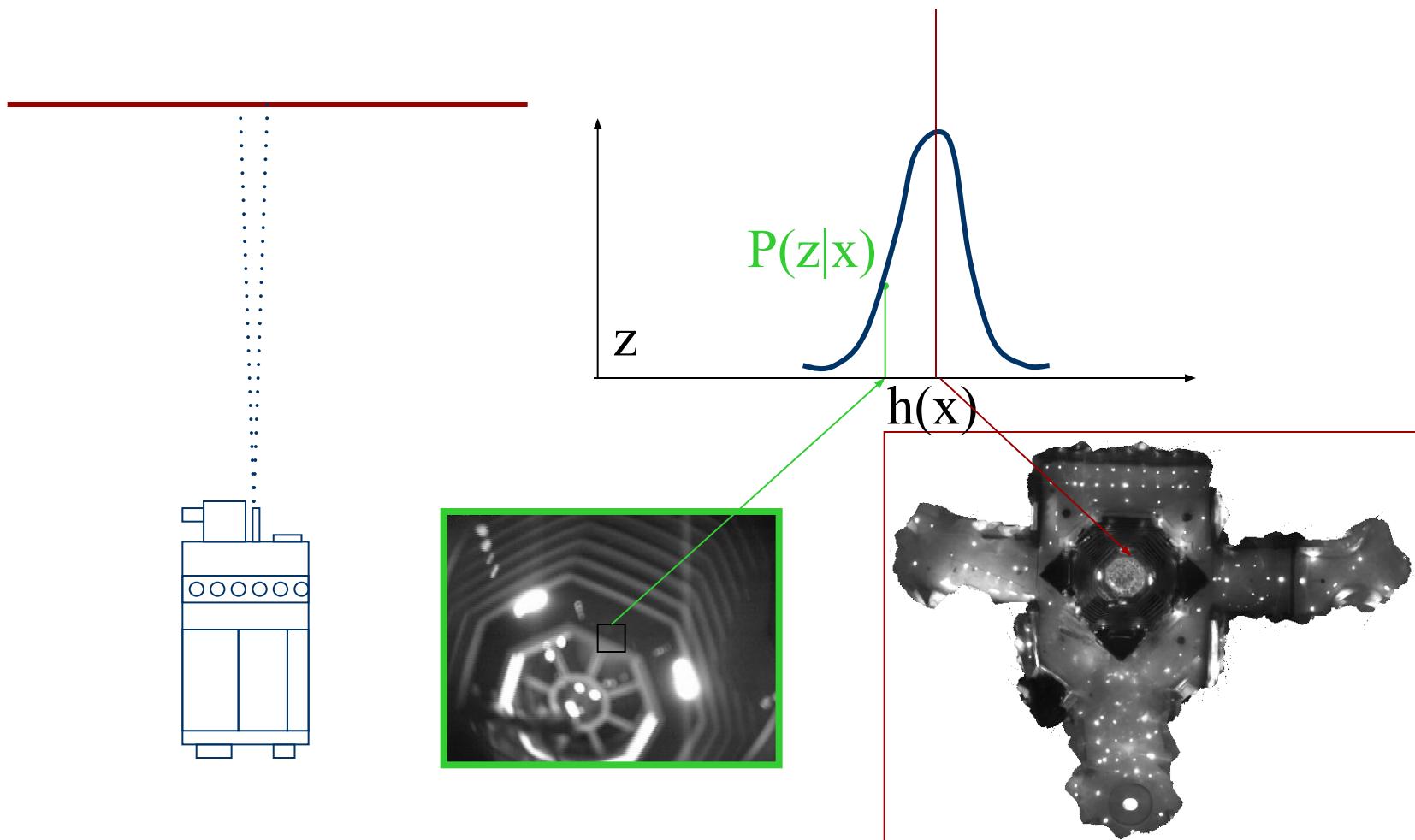
Courtesy: Thrun, Burgard, Fox 48

Using Ceiling Maps for Localization



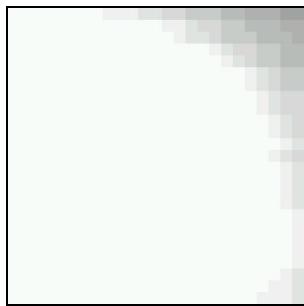
[Dellaert et al. 99]

Vision-Based Localization

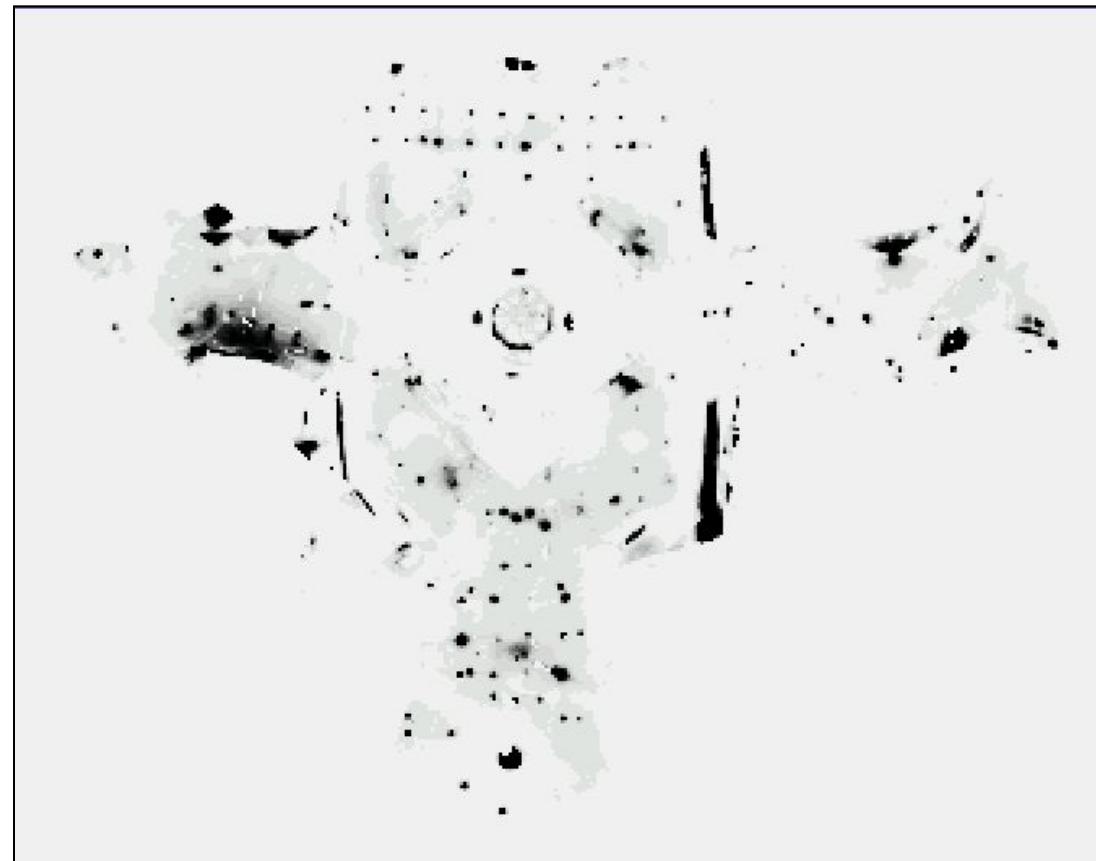


Under a Light

Measurement z:

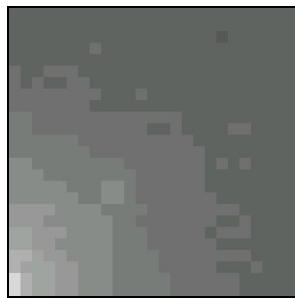


$P(z|x)$:

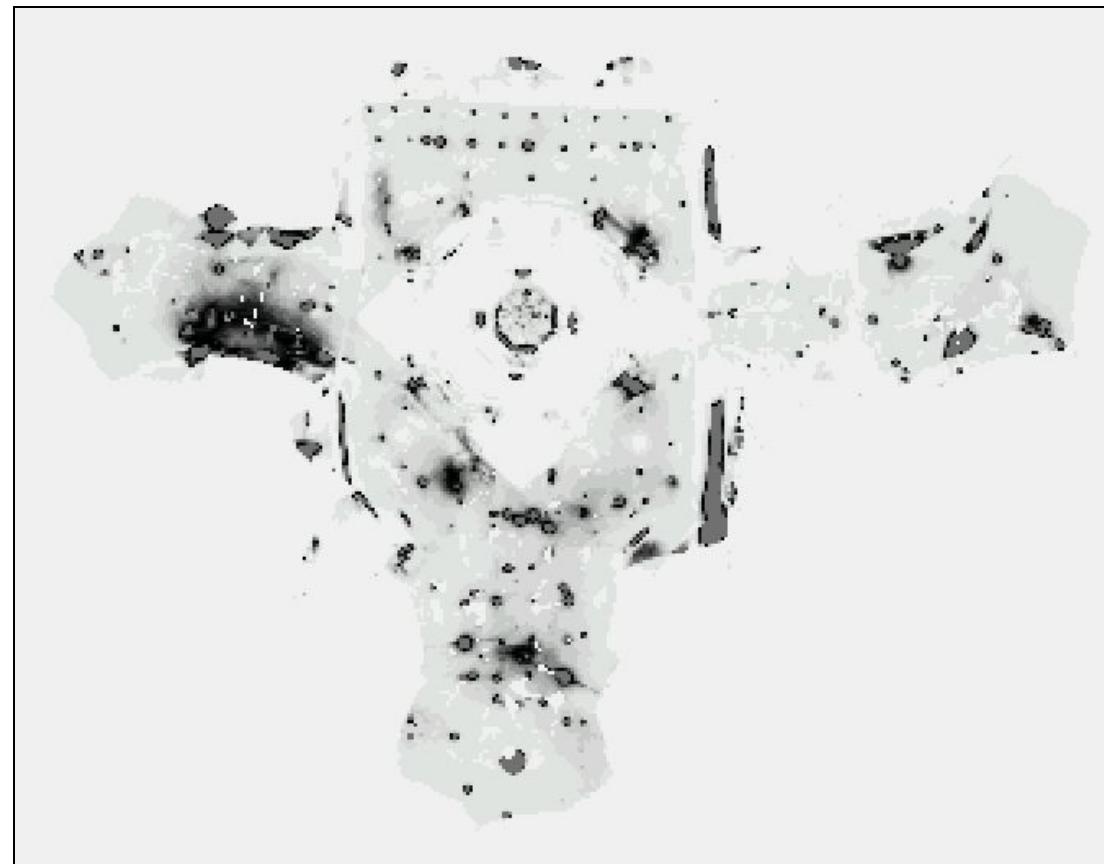


Next to a Light

Measurement z:



$P(z|x)$:

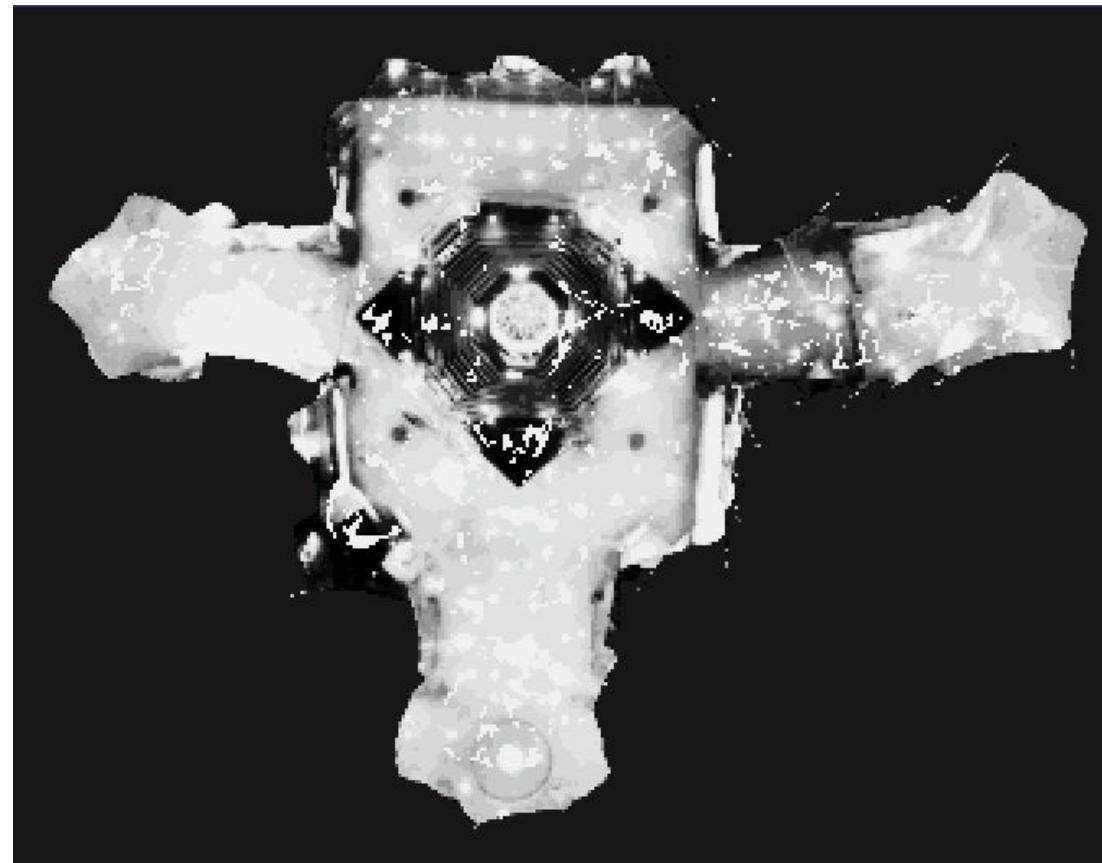


Elsewhere

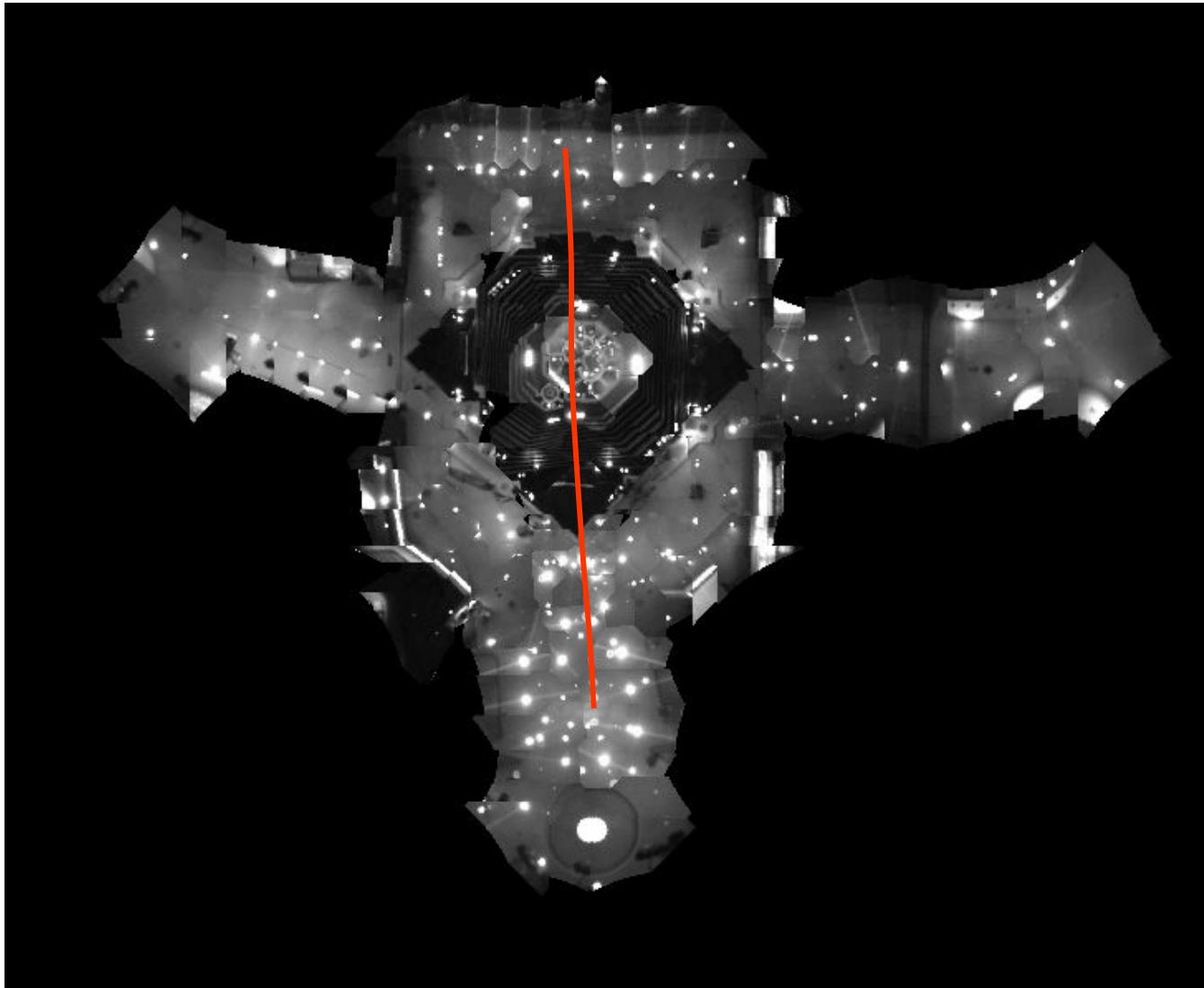
Measurement z:



$P(z|x)$:



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)?

ehna lw 7sl kidnapp lel robot, fa bl tre2a de msh hn2dr ne3ml recover, 34an n7l el moshkela de, bndkhl random number of samles zyada, 34an el average bta3hom ye2dr y2oli el ana fen bzbt.

w kman 34an ady e7tmalya en momken akon fe mkan hwa mknsh momkne ykon feh.

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