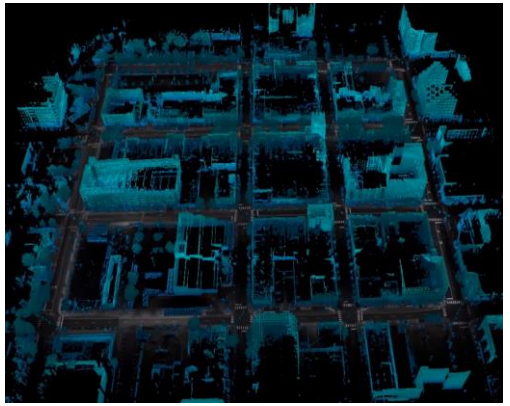
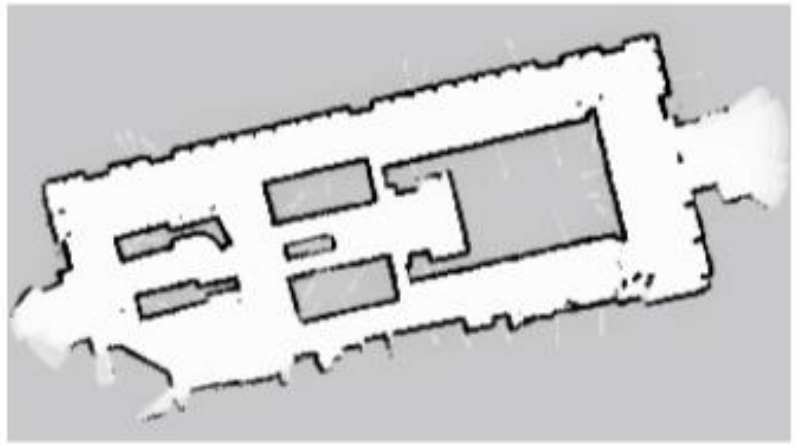
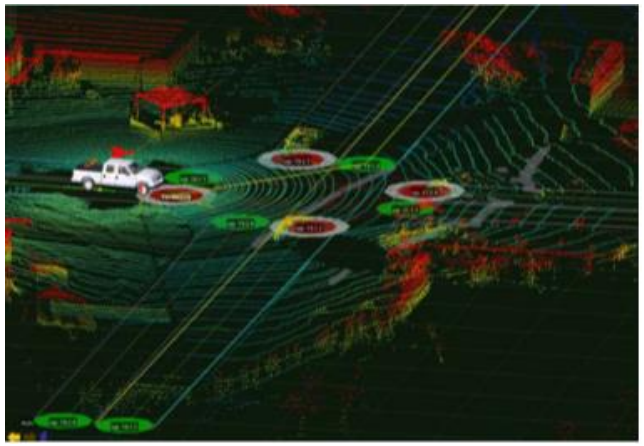
BYU Electrical & Computer Engineering





PARTICLE FILTER LOCALIZATION

ECEN 633: Robotic Localization and Mapping

Some Slides courtesy of Ryan Eustice.

Particle Filter Algorithm (Sub-optimal BUT EFFICIENT version)

Algorithm **particle_filter**(S_{t-1}, $\mathbf{u}_{t-1}$, $\mathbf{z}_t$):

1. $S_t = \varnothing, \quad \eta = 0$
2. **For** $i = 1 \ldots n$ — ***Generate new samples***
3. Sample $\mathbf{x}_t^i$ from $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_{t-1})$ using $\mathbf{x}_{t-1}^i$ and $\mathbf{u}_{t-1}$
4. $w_t^i = p(\mathbf{z}_t \mid \mathbf{x}_t^i)$ — ***Compute importance weight***
5. $\eta = \eta + w_t^i$ — ***Update normalization factor***
6. **For** $i = 1 \ldots N$
7. $w_t^i = w_t^i / \eta$ — ***Normalize weights***
8. Resample $\mathbf{x}_t^{i^*}$ from the discrete distribution given by w_t
9. ***Return*** $S_t = \{\mathbf{x}_t^{i^*}\}$

Particle Filter Algorithm (Sub-optimal BUT EFFICIENT version)

$$Bel(\mathbf{x}_t) = \eta\, p(\mathbf{z}_t \mid \mathbf{x}_t) \int p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_{t-1})\, Bel(\mathbf{x}_{t-1})\, d\mathbf{x}_{t-1}$$

draw $\mathbf{x}^i_{t-1}$ from $Bel(\mathbf{x}_{t-1})$

draw $\mathbf{x}^i_t$ from $p(\mathbf{x}_t \mid \mathbf{x}^i_{t-1}, \mathbf{u}_{t-1})$

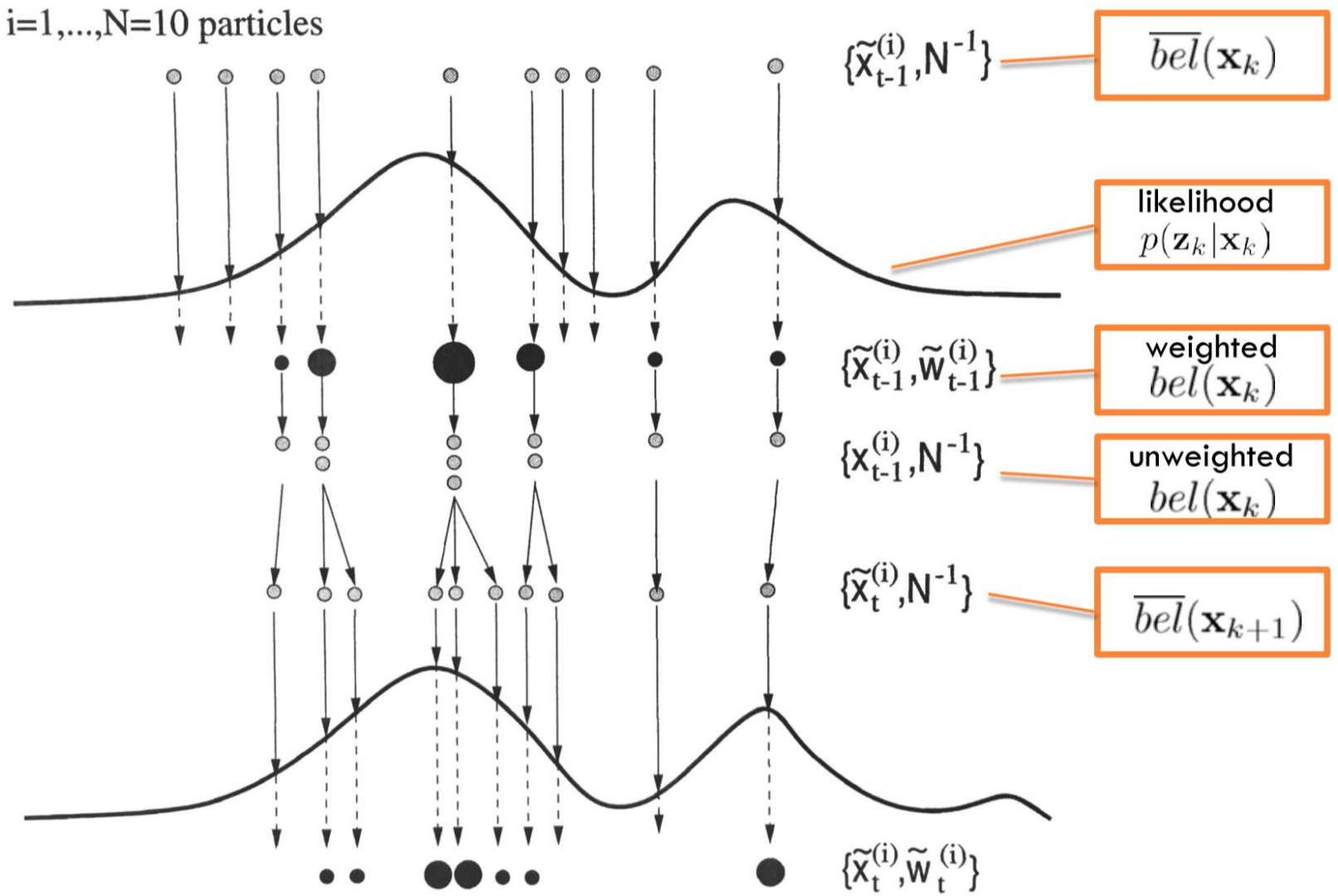
Importance factor for $\mathbf{x}^i_t$:

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

$$= \frac{\eta\, p(\mathbf{z}_t \mid \mathbf{x}_t)\, p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_{t-1})\, Bel(\mathbf{x}_{t-1})}{p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_{t-1})\, Bel(\mathbf{x}_{t-1})}$$

$$\propto p(\mathbf{z}_t \mid \mathbf{x}_t)$$

One Iteration of SIR PF



Monte Carlo Localization

- Each particle is a pose (trajectory) hypothesis
- Proposal is the motion model

$$\mathbf{x}_t^{[j]} \sim p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t)$$

- Correction via the observation model

$$w_t^{[j]} = \frac{\text{target}}{\text{proposal}} \propto p(\mathbf{z}_t \mid \mathbf{x}_t, \mathbf{m})$$

Courtesy: Cyrill Stachniss

Particle Filter Localization

Particle_filter$(\mathcal{X}_{t-1}, \mathbf{u}_t, \mathbf{z}_t)$**:**

1: $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
2: *for* $j = 1$ *to* J *do*
3: *sample* $\mathbf{x}_t^{[j]} \sim p(\mathbf{x}_t \mid \mathbf{u}_t, \mathbf{x}_{t-1}^{[j]})$
4: $w_t^{[j]} = p(\mathbf{z}_t \mid \mathbf{x}_t^{[j]})$
5: $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle \mathbf{x}_t^{[j]}, w_t^{[j]} \rangle$
6: *endfor*
7: *for* $j = 1$ *to* J *do*
8: *draw* $i \in 1, \ldots, J$ *with probability* $\propto w_t^{[i]}$
9: *add* $\mathbf{x}_t^{[i]}$ *to* $\mathcal{X}_t$
10: *endfor*
11: *return* $\mathcal{X}_t$

Courtesy: Cyrill Stachniss

How do we actually model this?

Odometry Motion Model

- How do we model the motion of the robot so that we can sample from it?

Odometry Motion Model – Prob. Robotics

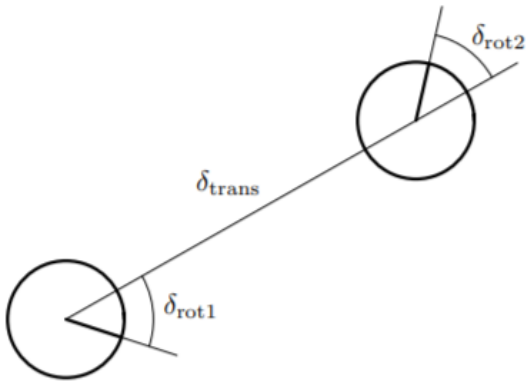


Figure 5.7 Odometry model: The robot motion in the time interval $(t-1, t]$ is approximated by a rotation δ_{rot1}, followed by a translation δ_{trans} and a second rotation δ_{rot2}. The turns and translations are noisy.

1: **Algorithm sample_motion_model_odometry**(u_t, x_{t-1})**:**

2: $\delta_{\text{rot1}} = \text{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$

3: $\delta_{\text{trans}} = \sqrt{(\bar{x} - \bar{x}')^2 + (\bar{y} - \bar{y}')^2}$

4: $\delta_{\text{rot2}} = \bar{\theta}' - \bar{\theta} - \delta_{\text{rot1}}$

5: $\hat{\delta}_{\text{rot1}} = \delta_{\text{rot1}} - \textbf{sample}(\alpha_1 \delta^2_{\text{rot1}} + \alpha_2 \delta^2_{\text{trans}})$

6: $\hat{\delta}_{\text{trans}} = \delta_{\text{trans}} - \textbf{sample}(\alpha_3 \delta^2_{\text{trans}} + \alpha_4\ \delta^2_{\text{rot1}} + \alpha_4\ \delta^2_{\text{rot2}})$

7: $\hat{\delta}_{\text{rot2}} = \delta_{\text{rot2}} - \textbf{sample}(\alpha_1 \delta^2_{\text{rot2}} + \alpha_2 \delta^2_{\text{trans}})$

8: $x' = x + \hat{\delta}_{\text{trans}}\ \cos(\theta + \hat{\delta}_{\text{rot1}})$

9: $y' = y + \hat{\delta}_{\text{trans}}\ \sin(\theta + \hat{\delta}_{\text{rot1}})$

10: $\theta' = \theta + \hat{\delta}_{\text{rot1}} + \hat{\delta}_{\text{rot2}}$

11: *return* $x_t = (x', y', \theta')^T$

Table 5.6 Algorithm for sampling from $p(x_t \mid u_t, x_{t-1})$ based on odometry information. Here the pose at time t is represented by $x_{t-1} = (x\ \ y\ \ \theta)^T$. The control is a differentiable set of two pose estimates obtained by the robot's odometer, $u_t = (\bar{x}_{t-1}\ \ \bar{x}_t)^T$, with $\bar{x}_{t-1} = (\bar{x}\ \ \bar{y}\ \ \bar{\theta})$ and $\bar{x}_t = (\bar{x}'\ \ \bar{y}'\ \ \bar{\theta}')$.

Odometry Motion Model – Prob. Robotics

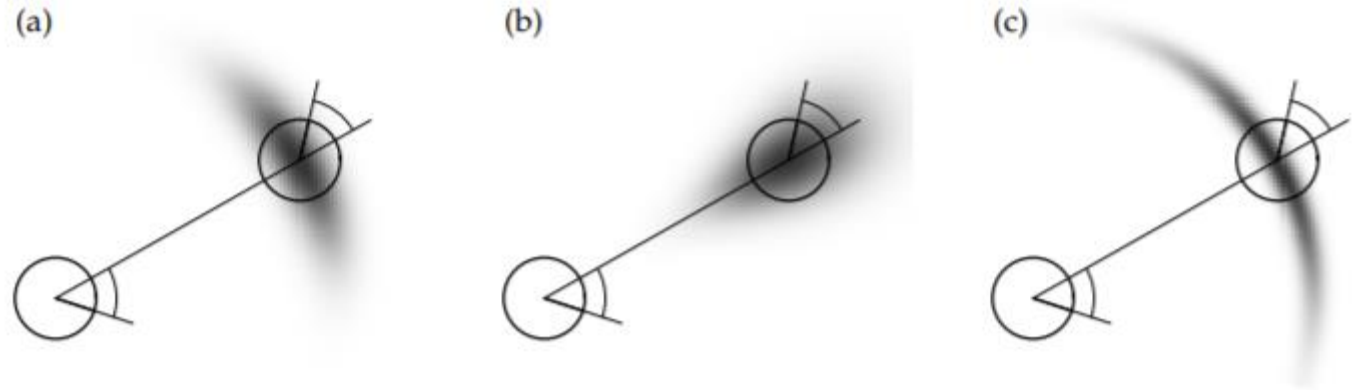


Figure 5.8 The odometry motion model, for different noise parameter settings.

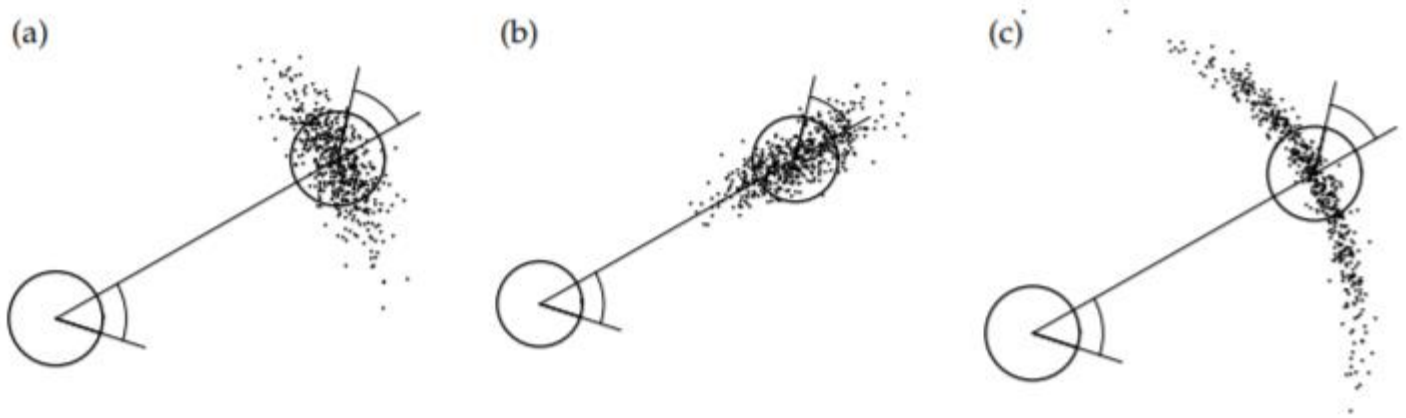


Figure 5.9 Sampling from the odometry motion model, using the same parameters as in Figure 5.8. Each diagram shows 500 samples.

Another Motion Model – Pose Composition (SSC)

- Smith, Self, and Cheeseman (SSC)
  - Head-to-Tail

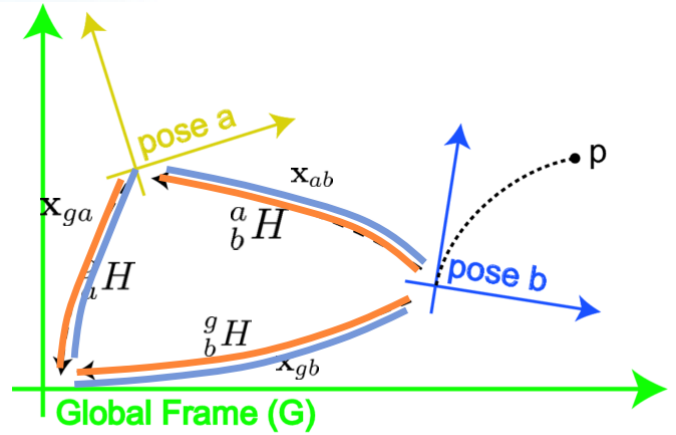
$$\mathbf{x}_{gb} = \mathbf{x}_{ga} \oplus \mathbf{x}_{ab} \quad \Leftrightarrow \quad {}^{g}_{b}H = {}^{g}_{a}H \, {}^{a}_{b}H$$

  - Inverse

$$\mathbf{x}_{ag} = \ominus \mathbf{x}_{ga} \quad \Leftrightarrow \quad {}^{a}_{g}H = {}^{g}_{a}H^{-1}$$

  - Relative Pose

$$\mathbf{x}_{ab} = \ominus \mathbf{x}_{ga} \oplus \mathbf{x}_{gb} \quad \Leftrightarrow \quad {}^{a}_{b}H = {}^{g}_{a}H^{-1} \, {}^{g}_{b}H$$



Which one would be used to calculate an updated pose due to robot motion?

SSC: Head-to-tail

- We can compose two rigid-body transformations such that:

$$ {}^i_k H = {}^i_j H \, {}^j_k H \quad \Leftrightarrow \quad \mathbf{x}_{ik} = \mathbf{x}_{ij} \oplus \mathbf{x}_{jk} $$

- Can we keep it in parameter space?
- How to derive?
  - Multiply out matrices, manipulate until parameters can be read out.

$$ \mathbf{x}_{ik} = \begin{bmatrix} x_{ik} \\ y_{ik} \\ \theta_{ik} \end{bmatrix} = \begin{bmatrix} x_{jk}\cos\theta_{ij} - y_{jk}\sin\theta_{ij} + x_{ij} \\ x_{jk}\sin\theta_{ij} + y_{jk}\cos\theta_{ij} + y_{ij} \\ \theta_{ij} + \theta_{jk} \end{bmatrix} $$

Can you compute $\sum_{\mathbf{x}_{ik}}$ from $\sum_{\mathbf{x}_{ij}}$ and $\sum_{\mathbf{x}_{jk}}$????

Also need $\sum_{\mathbf{x}_{ij}\mathbf{x}_{jk}}$

$$\mathbf{x}_{ik} = \begin{bmatrix} x_{ik} \\ y_{ik} \\ \theta_{ik} \end{bmatrix} = \begin{bmatrix} x_{jk}\cos\theta_{ij} - y_{jk}\sin\theta_{ij} + x_{ij} \\ x_{jk}\sin\theta_{ij} + y_{jk}\cos\theta_{ij} + y_{ij} \\ \theta_{ij} + \theta_{jk} \end{bmatrix}$$

$$J_{\oplus} = \frac{\partial \mathbf{x}_{ik}}{\partial(\mathbf{x}_{ij}, \mathbf{x}_{jk})}$$

$$= \begin{bmatrix} 1 & 0 & (-x_{jk}\sin\theta_{ij} - y_{jk}\cos\theta_{ij}) & \cos\theta_{ij} & -\sin\theta_{ij} & 0 \\ 0 & 1 & (x_{jk}\cos\theta_{ij} - y_{jk}\sin\theta_{ij}) & \sin\theta_{ij} & \cos\theta_{ij} & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$= \underbrace{\begin{bmatrix} 1 & 0 & (-x_{jk}\sin\theta_{ij} - y_{jk}\cos\theta_{ij}) \\ 0 & 1 & (x_{jk}\cos\theta_{ij} - y_{jk}\sin\theta_{ij}) \\ 0 & 0 & 1 \end{bmatrix}}_{J_{\oplus 1}} \underbrace{\begin{bmatrix} \cos\theta_{ij} & -\sin\theta_{ij} & 0 \\ \sin\theta_{ij} & \cos\theta_{ij} & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{J_{\oplus 2}}$$

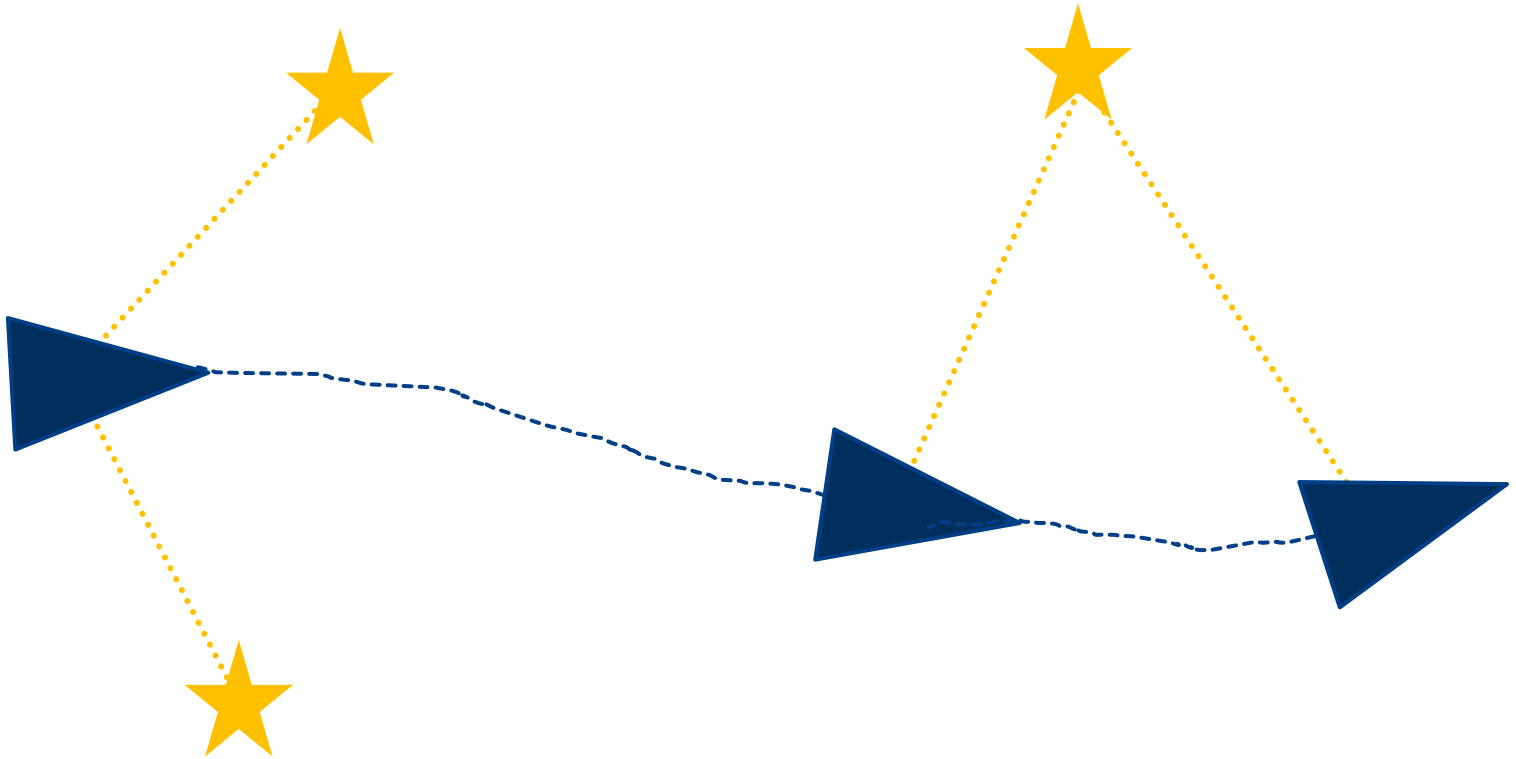
$$\Sigma_{\mathbf{x}_{ik}} = J_{\oplus}|_{\mu_{ij},\mu_{jk}} \begin{bmatrix} \Sigma_{\mathbf{x}_{ij}} & \Sigma_{\mathbf{x}_{ij}\mathbf{x}_{jk}} \\ \Sigma^{\top}_{\mathbf{x}_{ij}\mathbf{x}_{jk}} & \Sigma_{\mathbf{x}_{jk}} \end{bmatrix} J^{\top}_{\oplus}|_{\mu_{ij},\mu_{jk}}$$

Then we can sample from this new distribution!

Measurement Model

- How do we model measurements so we can evaluate their probability?

Beacon Measurement Model



Beacon Measurement Model

Algorithm landmark_model_known_correspondence(f_t^i, c_t^i, x_t, m)**:**

$j = c_t^i$

$\hat{r} = \sqrt{(m_{j,x} - x)^2 + (m_{j,y} - y)^2}$

$\hat{\phi} = \text{atan2}(m_{j,y} - y, m_{j,x} - x) - \theta$

$q = \textbf{prob}(r_t^i - \hat{r}, \sigma_r) \cdot \textbf{prob}(\phi_t^i - \hat{\phi}, \sigma_\phi) \cdot \textbf{prob}(s_t^i - s_j, \sigma_s)$

return q

IGNORE

Table 6.4 Algorithm for computing the likelihood of a landmark measurement. The algorithm requires as input an observed feature $f_t^i = (r_t^i \;\; \phi_t^i \;\; s_t^i)^T$, and the true identity of the feature c_t^i, the robot pose $x_t = (x \;\; y \;\; \theta)^T$, and the map m. Its output is the numerical probability $p(f_t^i \mid c_t^i, m, x_t)$.

Laser Range Measurement Model

Thought Experiment

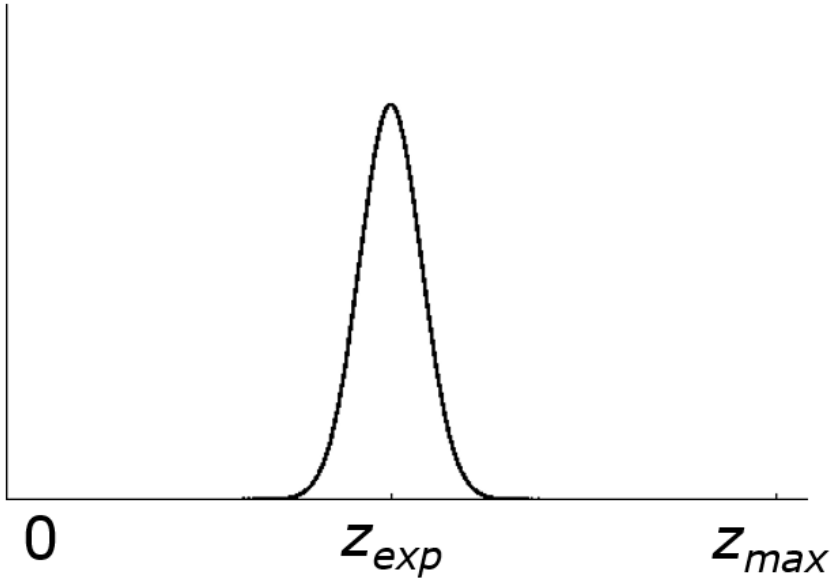
Given a robotic vehicle with:

- A previously estimated occupancy grid map
- An array of laser range finders (like we used in the lab)

How would you evaluate your measurements and update particle weights?

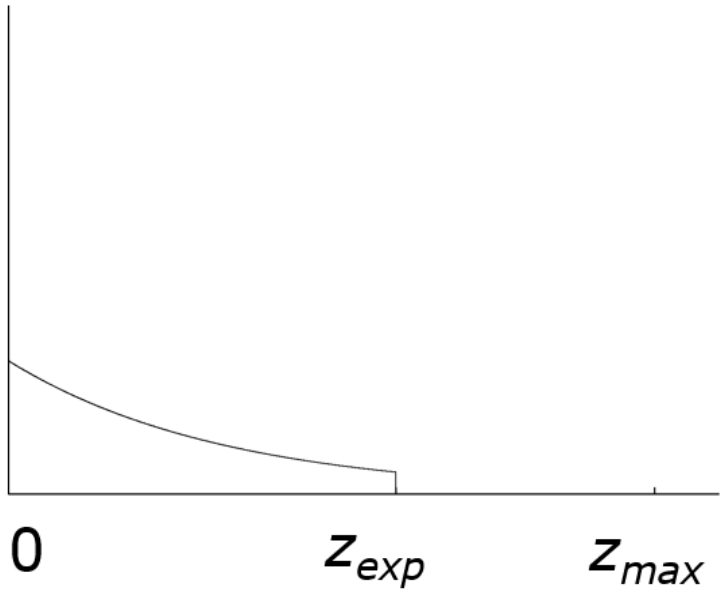
Laser Range Measurement Model

Measurement noise



$$P_{hit}(z \mid x, m) = \eta \frac{1}{\sqrt{2\pi b}} e^{-\frac{1}{2}\frac{(z - z_{\exp})^2}{b}}$$

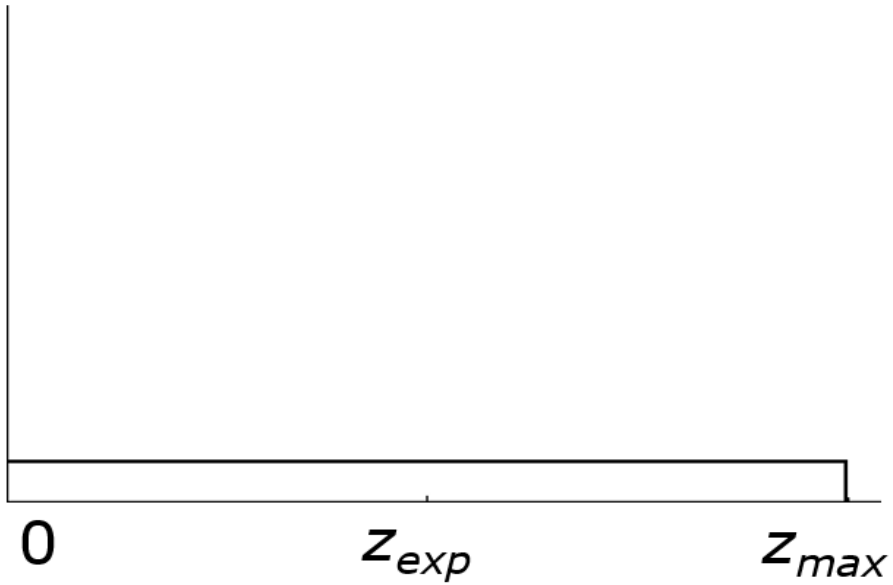
Unexpected obstacles



$$P_{\text{unexp}}(z \mid x, m) = \begin{cases} \eta\, \lambda\, \mathrm{e}^{-\lambda z} & z < z_{\exp} \\ 0 & otherwise \end{cases}$$

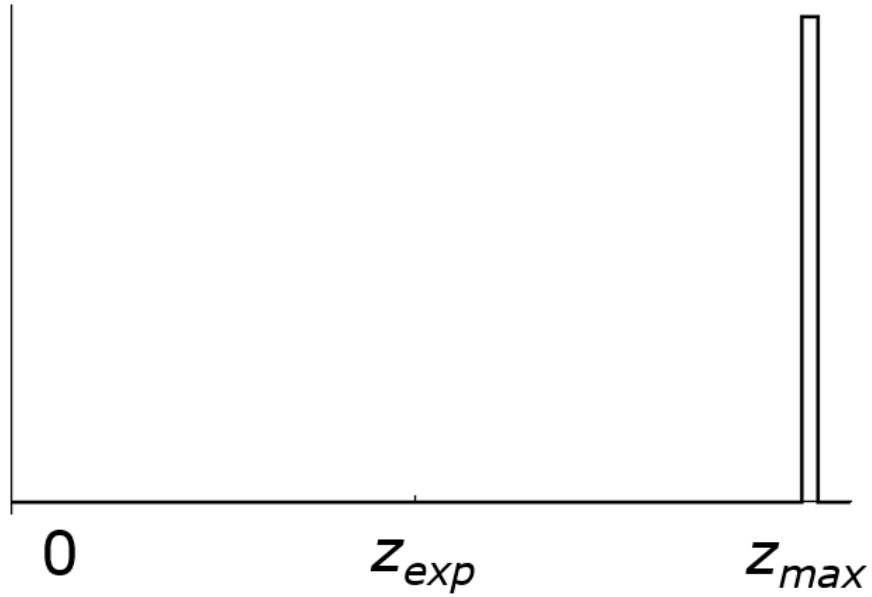
Laser Range Measurement Model

Random measurement



$$P_{rand}(z \mid x, m) = \eta \frac{1}{z_{\max}}$$

Max range



$$P_{\max}(z \mid x, m) = \eta \frac{1}{z_{small}}$$

Laser Range Measurement Model

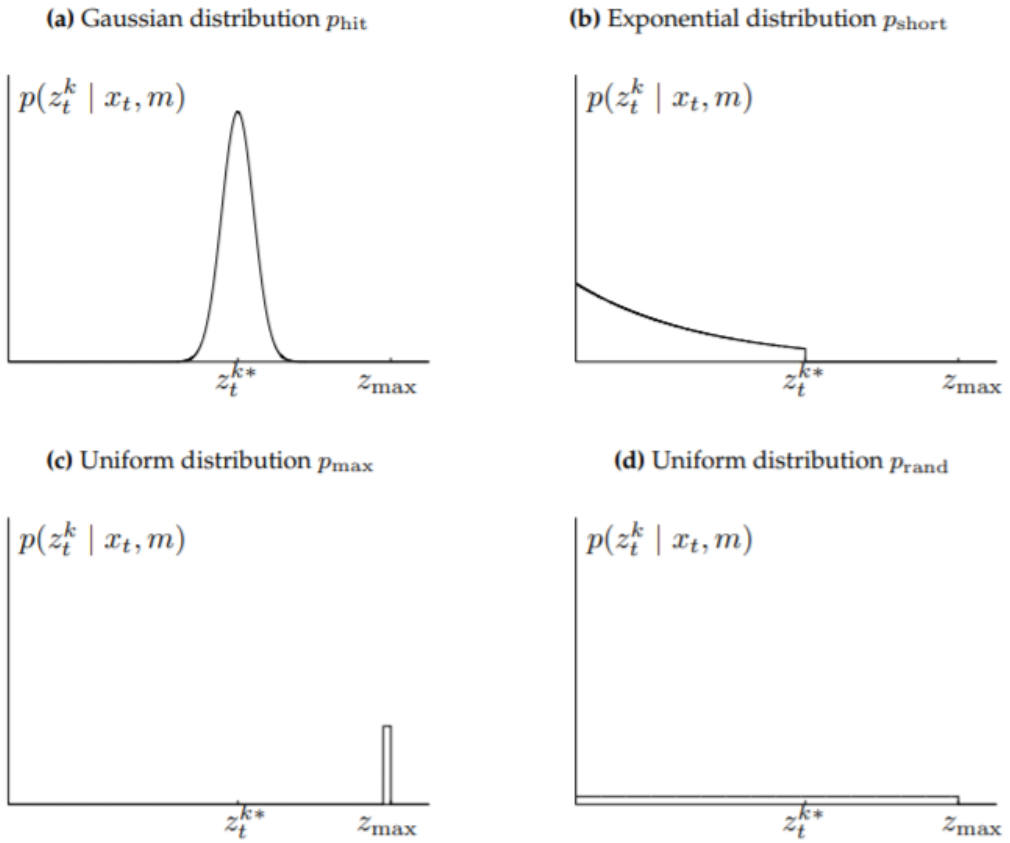


Figure 6.3 Components of the range finder sensor model. In each diagram the horizontal axis corresponds to the measurement z_t^k, the vertical to the likelihood.

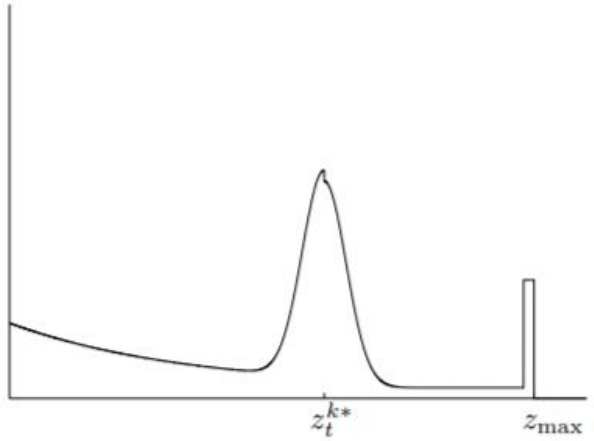
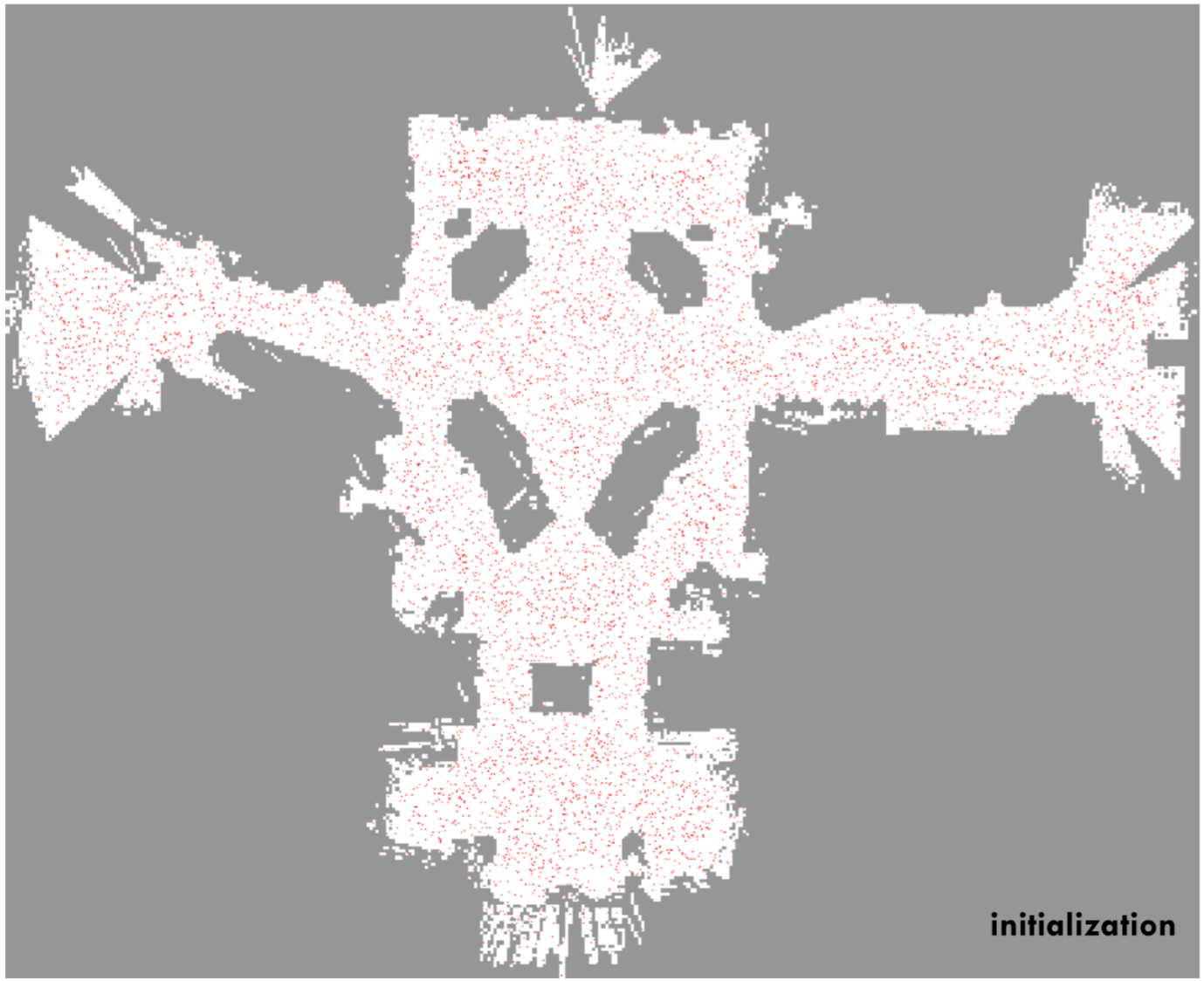


Figure 6.4 "Pseudo-density" of a typical mixture distribution $p(z_t^k \mid x_t, m)$.

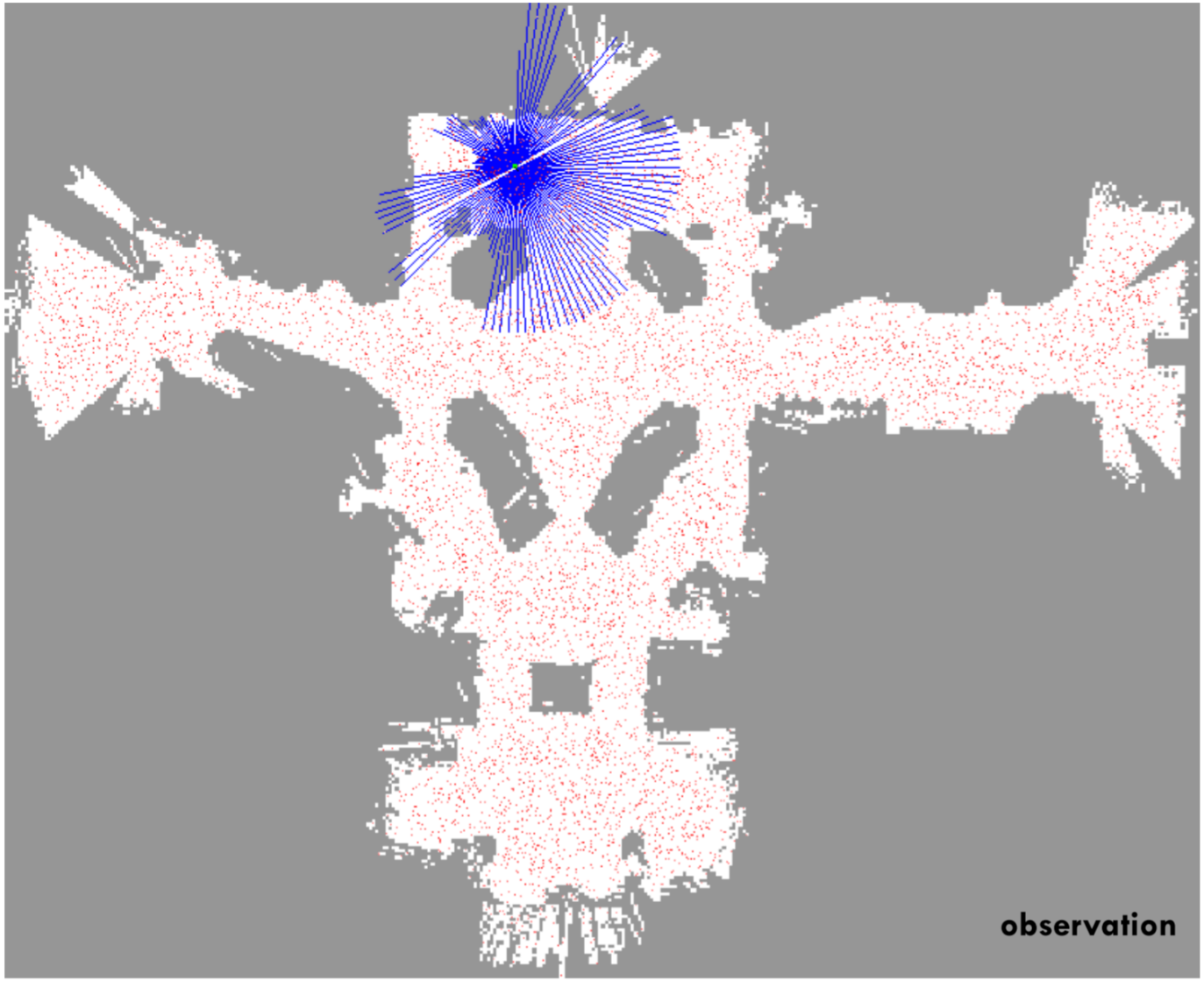
$$P(z \mid x, m) = \begin{pmatrix} \alpha_{\text{hit}} \\ \alpha_{\text{unexp}} \\ \alpha_{\text{max}} \\ \alpha_{\text{rand}} \end{pmatrix}^T \cdot \begin{pmatrix} P_{\text{hit}}(z \mid x, m) \\ P_{\text{unexp}}(z \mid x, m) \\ P_{\text{max}}(z \mid x, m) \\ P_{\text{rand}}(z \mid x, m) \end{pmatrix}$$

Example



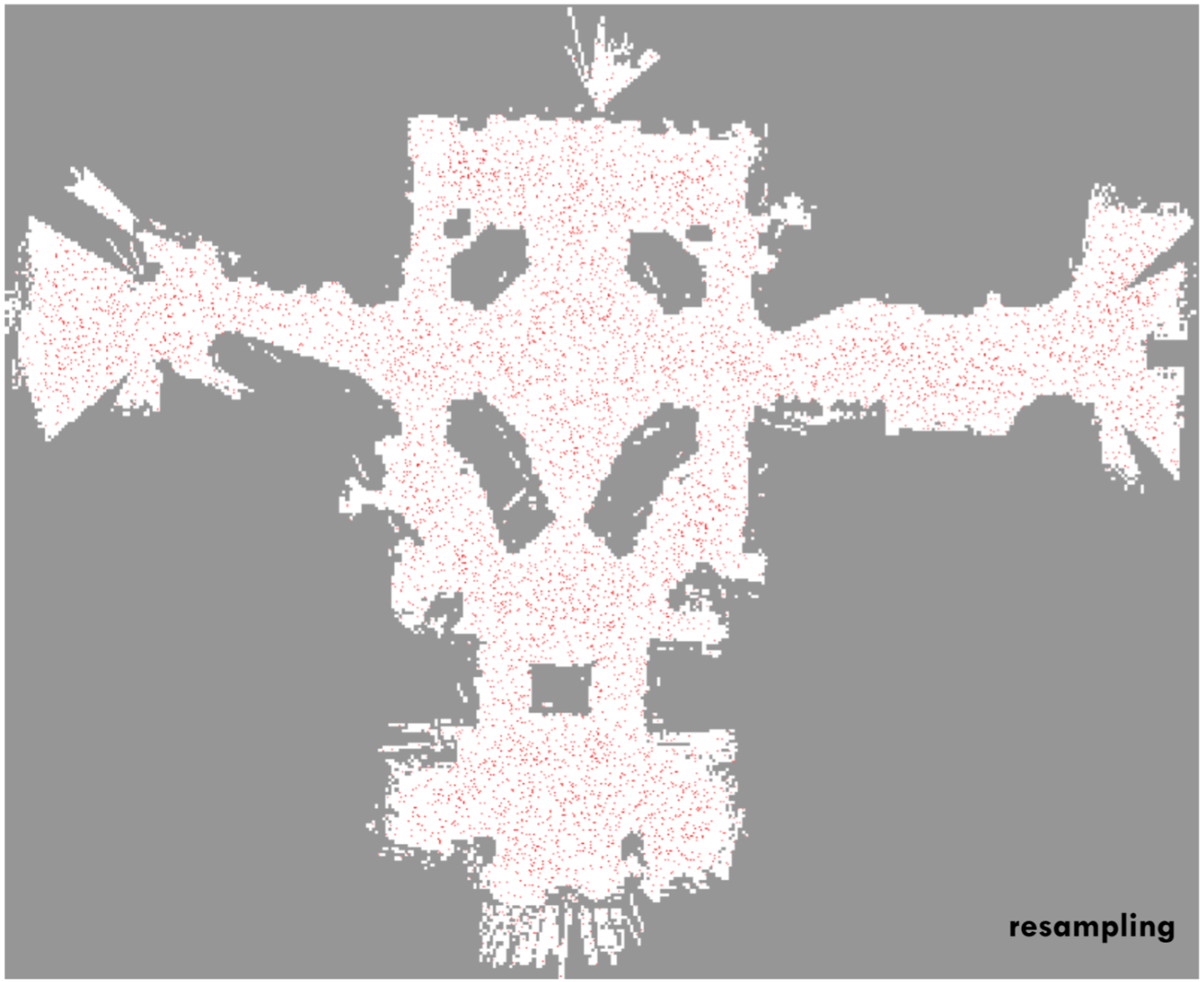
Courtesy: Thrun, Burgard, Fox

Example



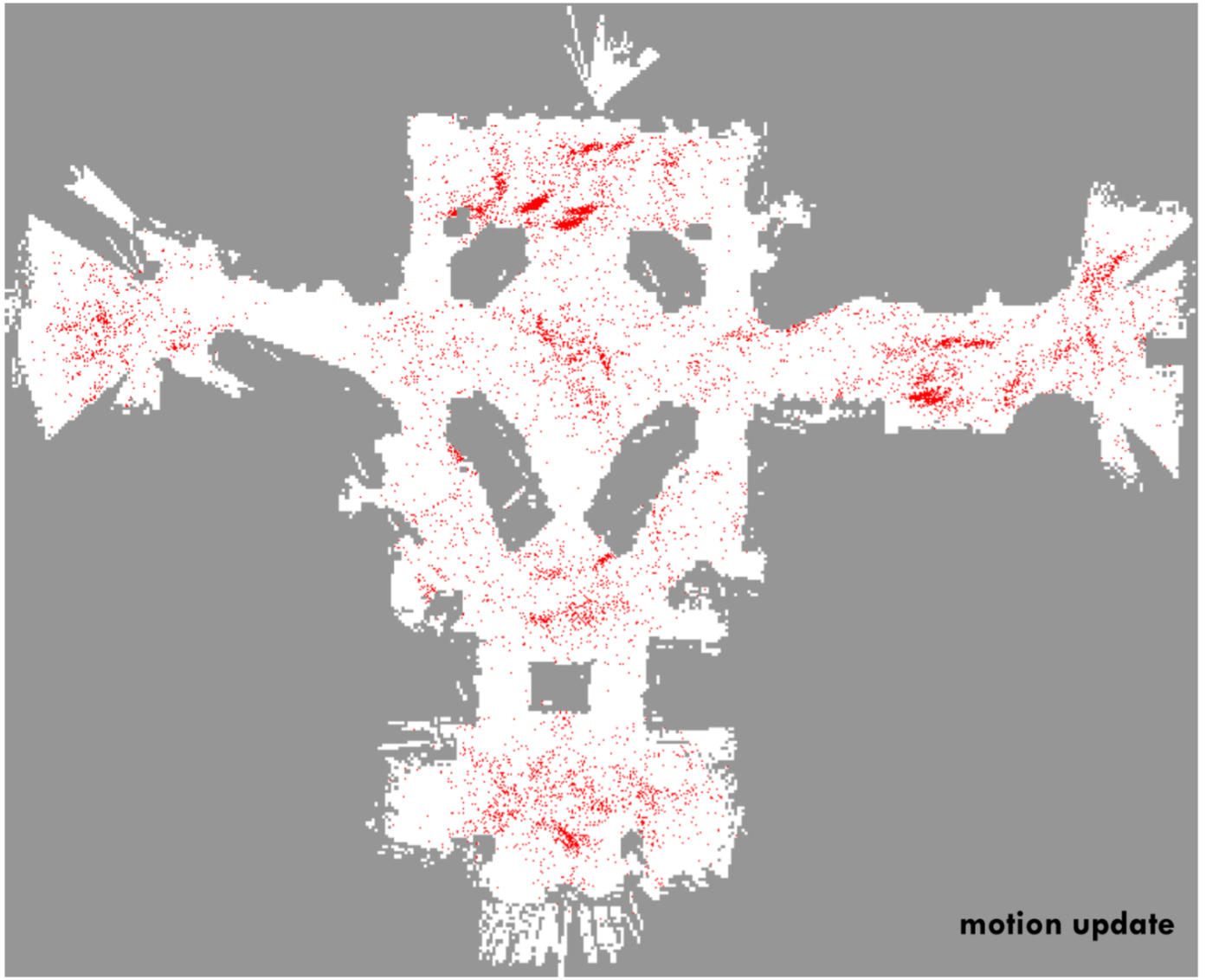
Courtesy: Thrun, Burgard, Fox

Example



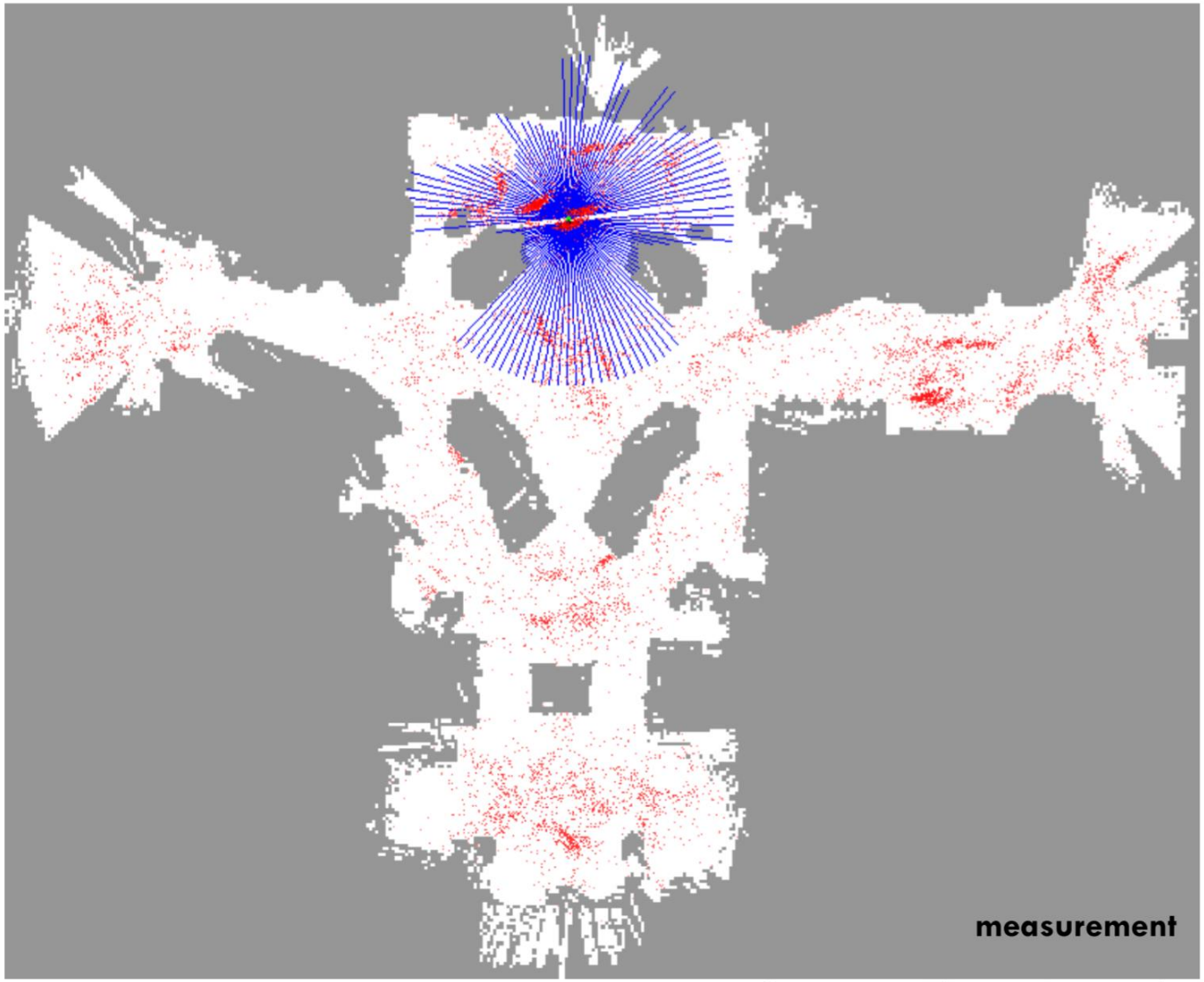
Courtesy: Thrun, Burgard, Fox

Example



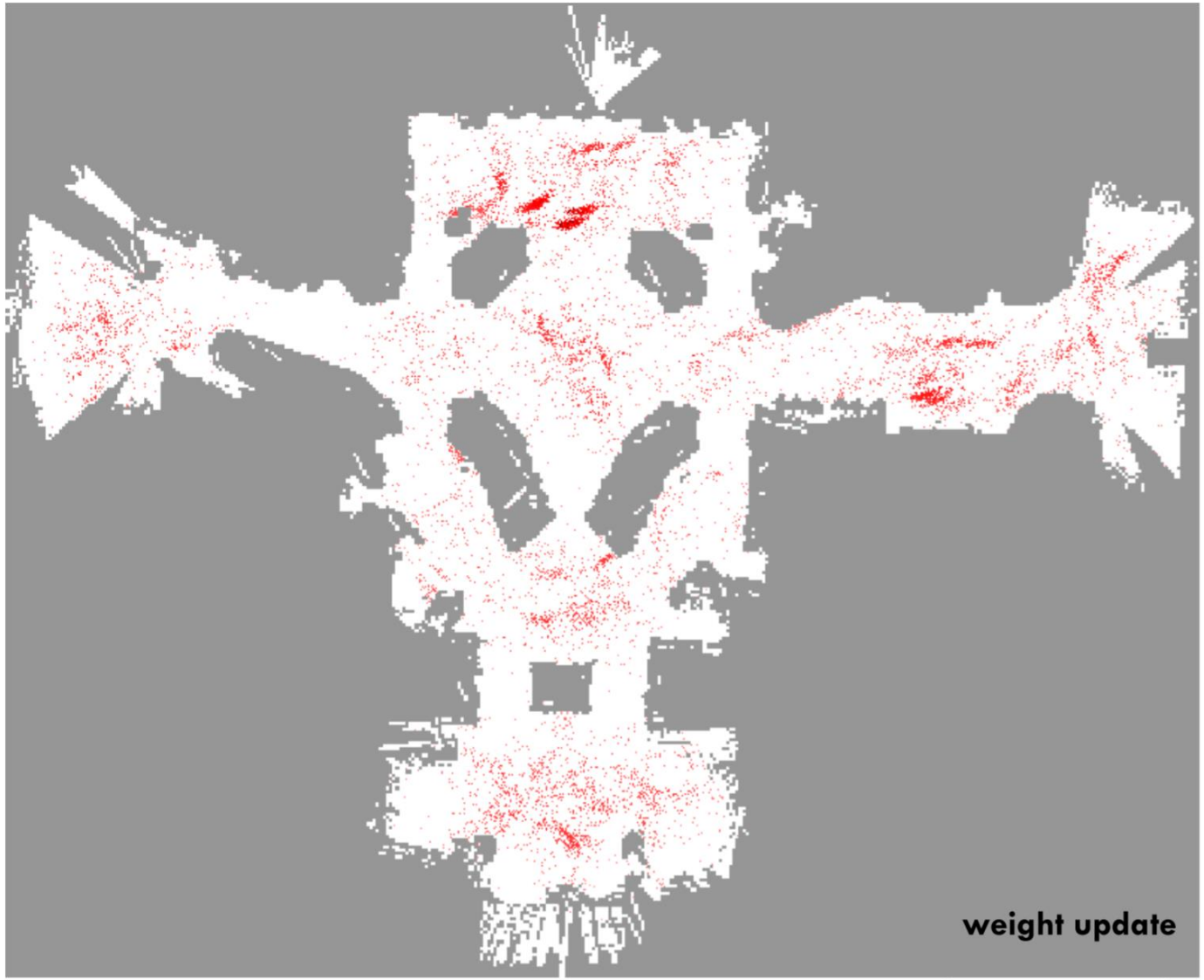
Courtesy: Thrun, Burgard, Fox

Example



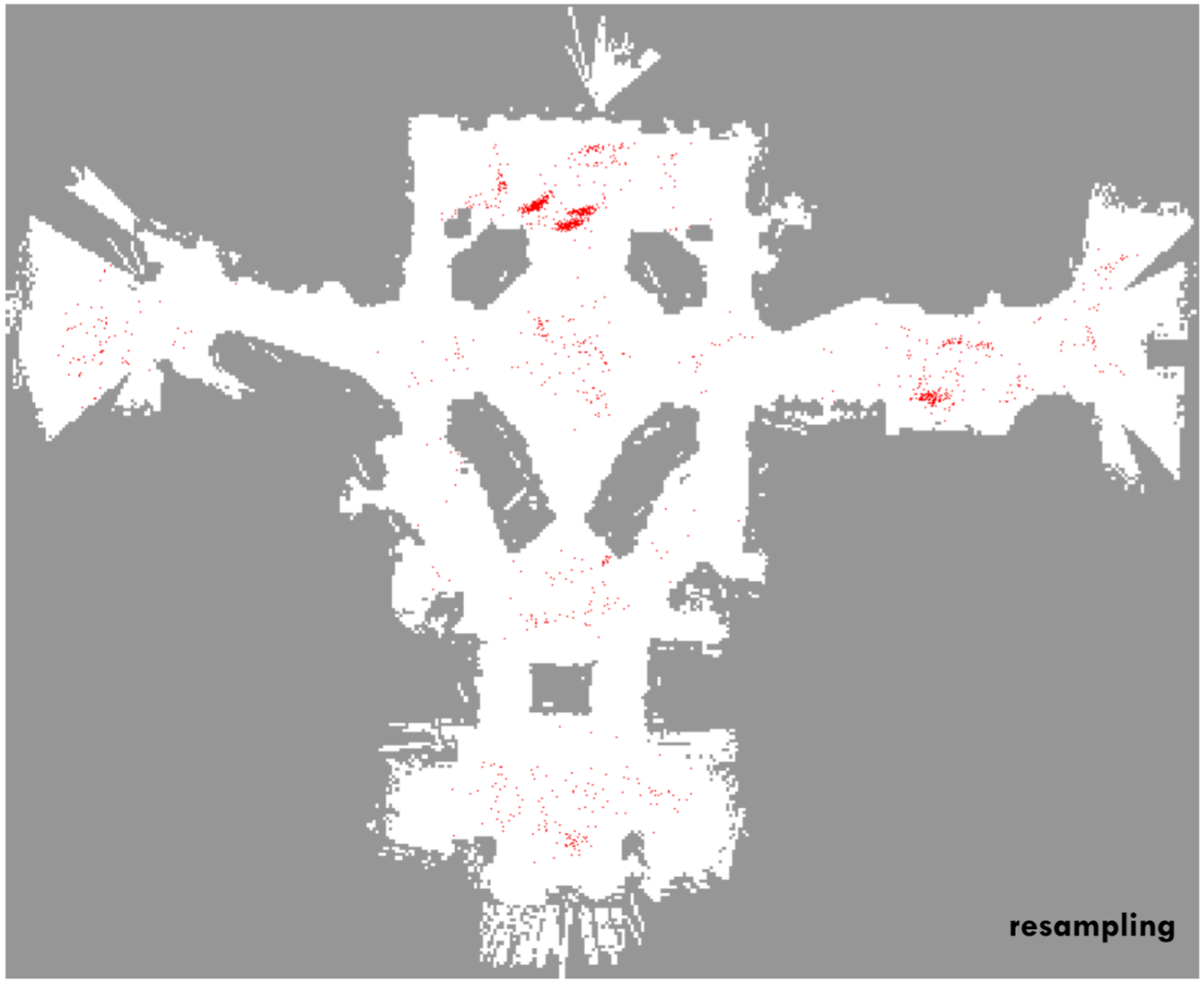
Courtesy: Thrun, Burgard, Fox

Example



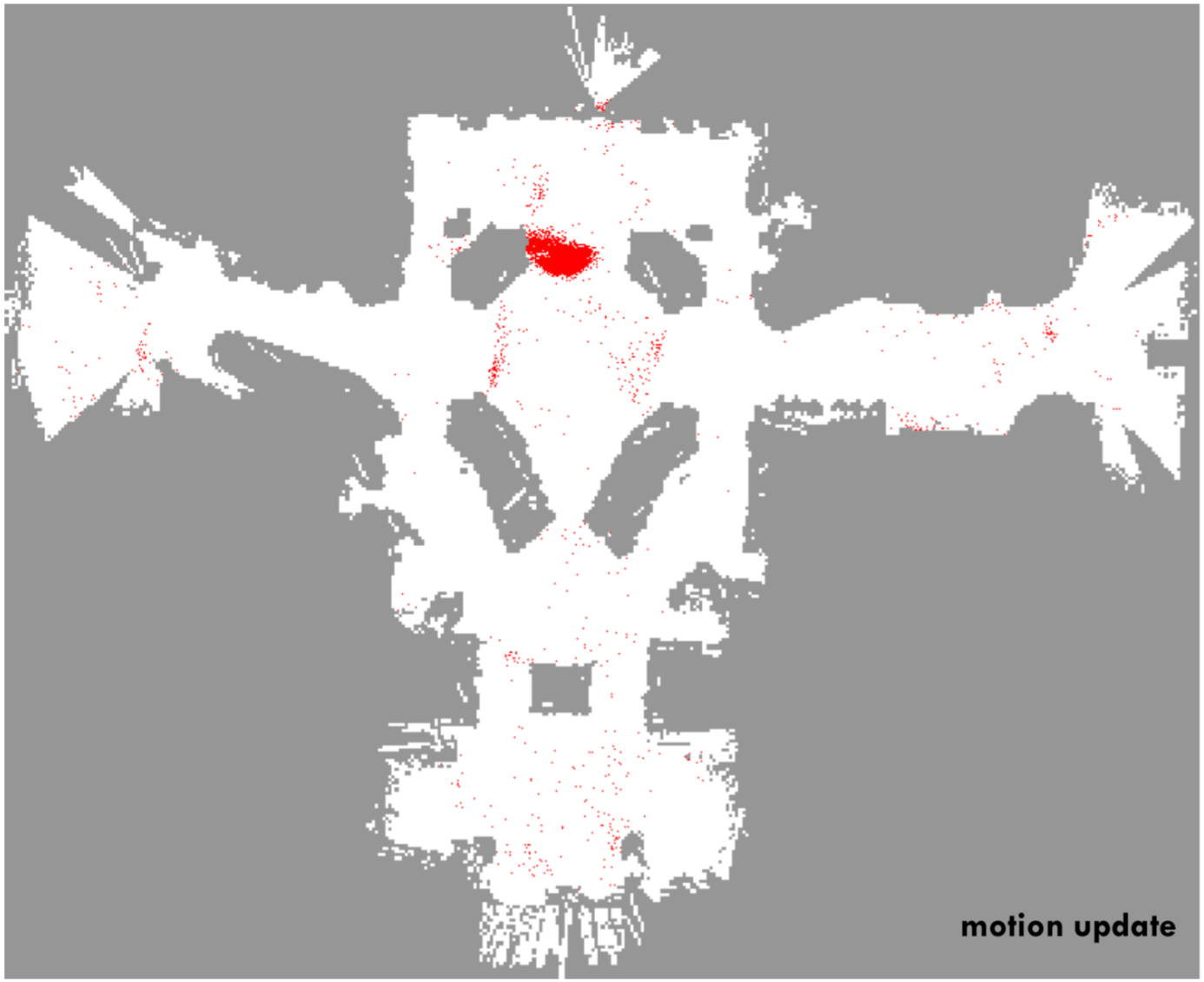
Courtesy: Thrun, Burgard, Fox

Example



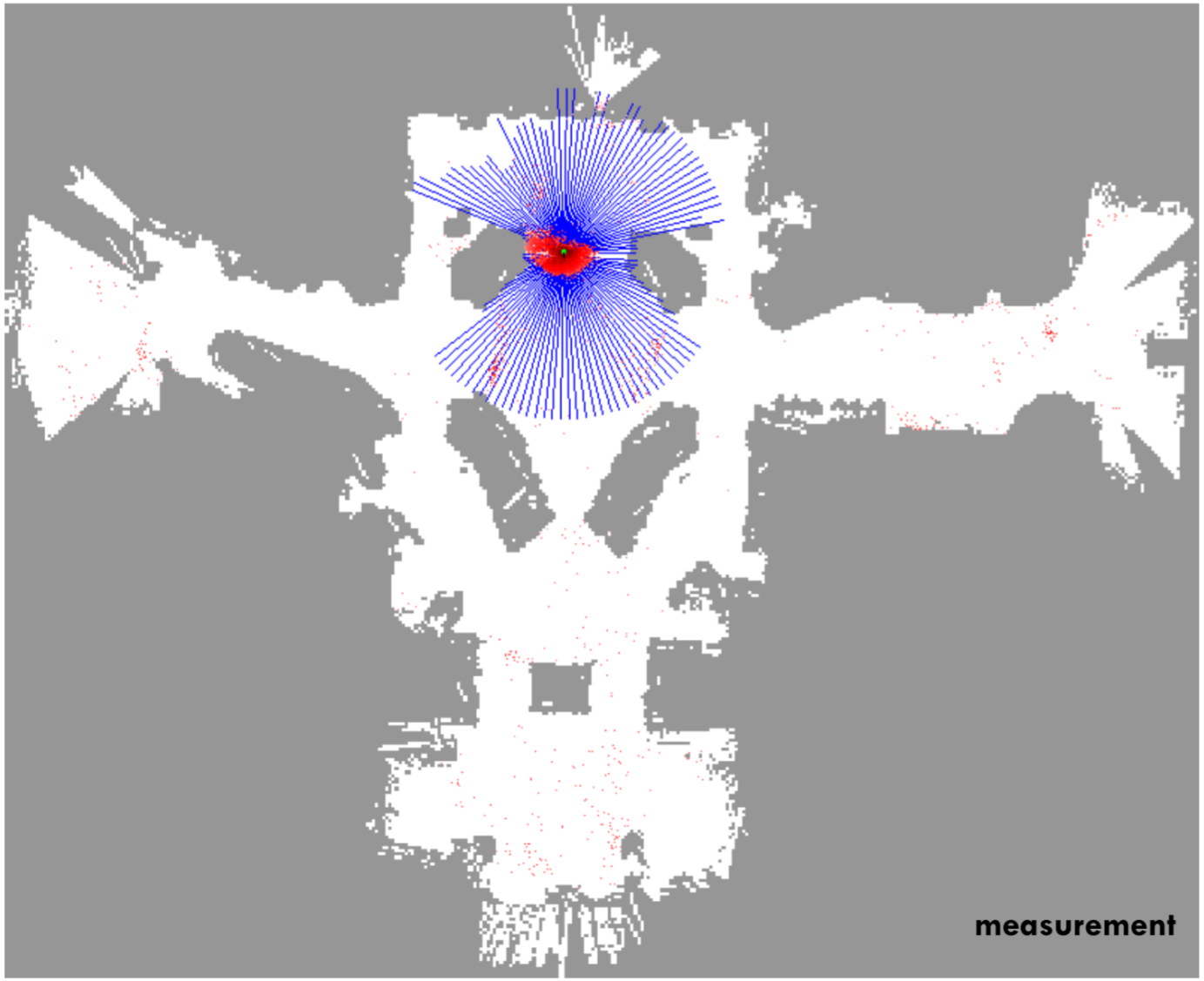
Courtesy: Thrun, Burgard, Fox

Example



Courtesy: Thrun, Burgard, Fox

Example

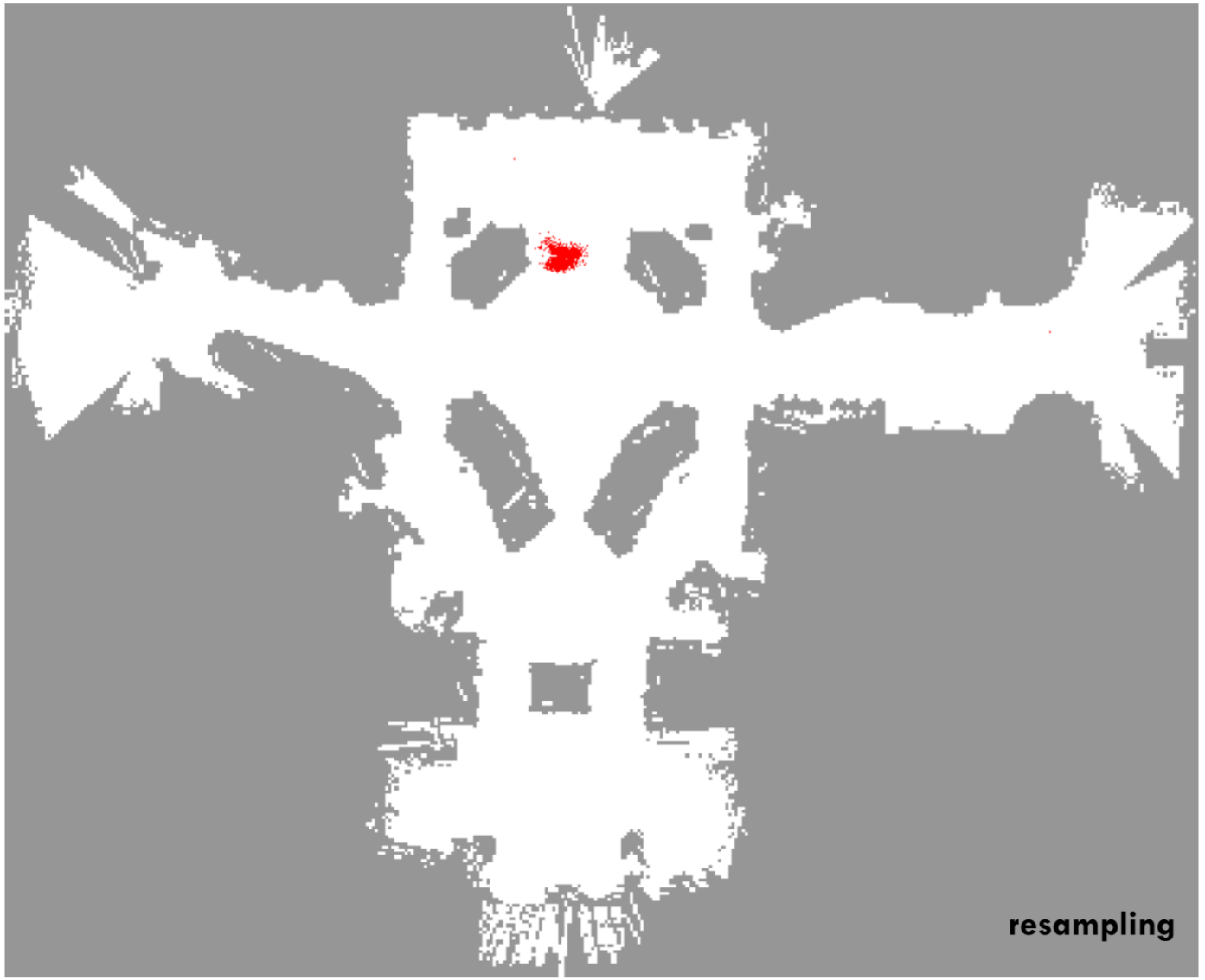


Courtesy: Thrun, Burgard, Fox

Example

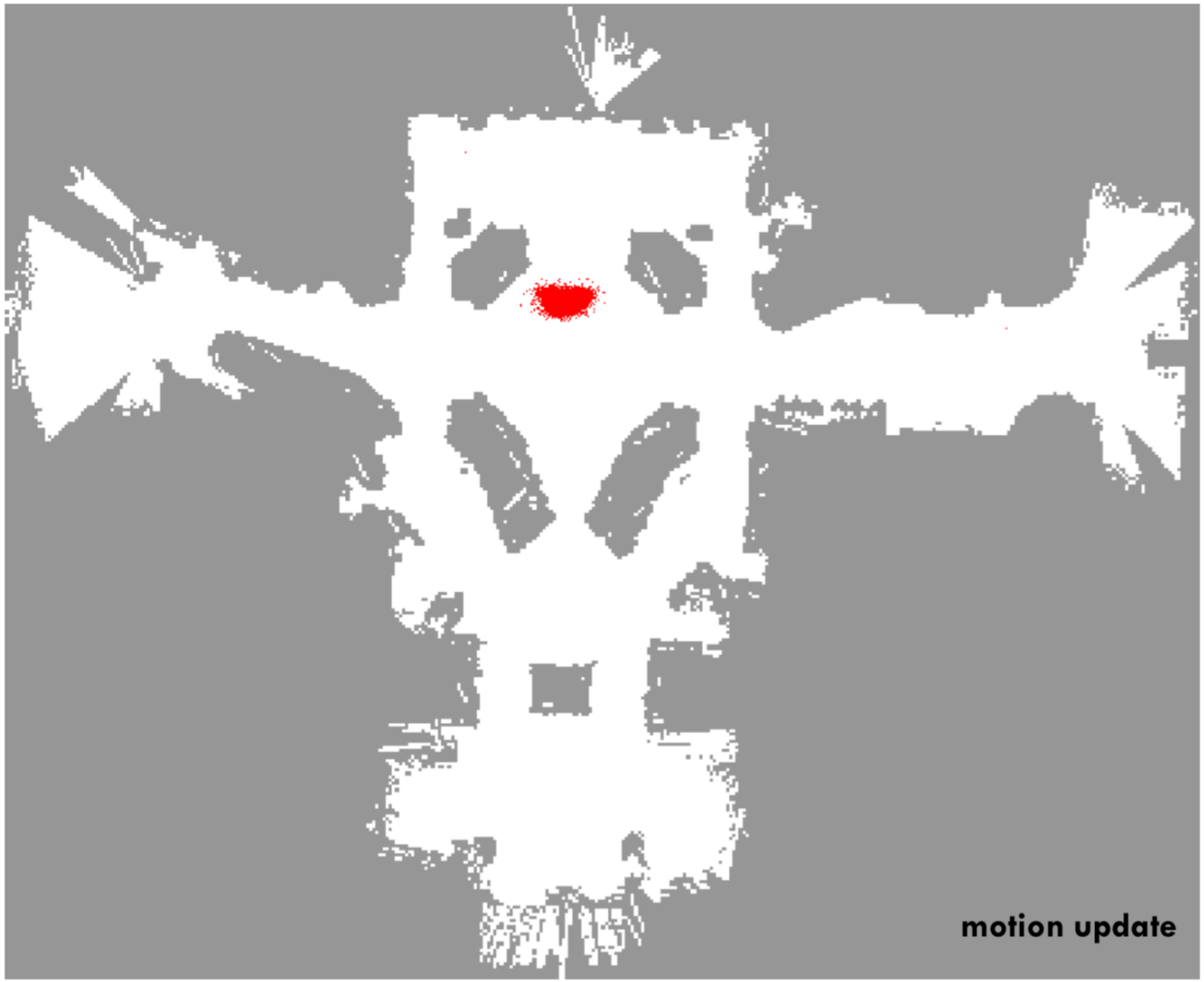
Courtesy: Thrun, Burgard, Fox

Example



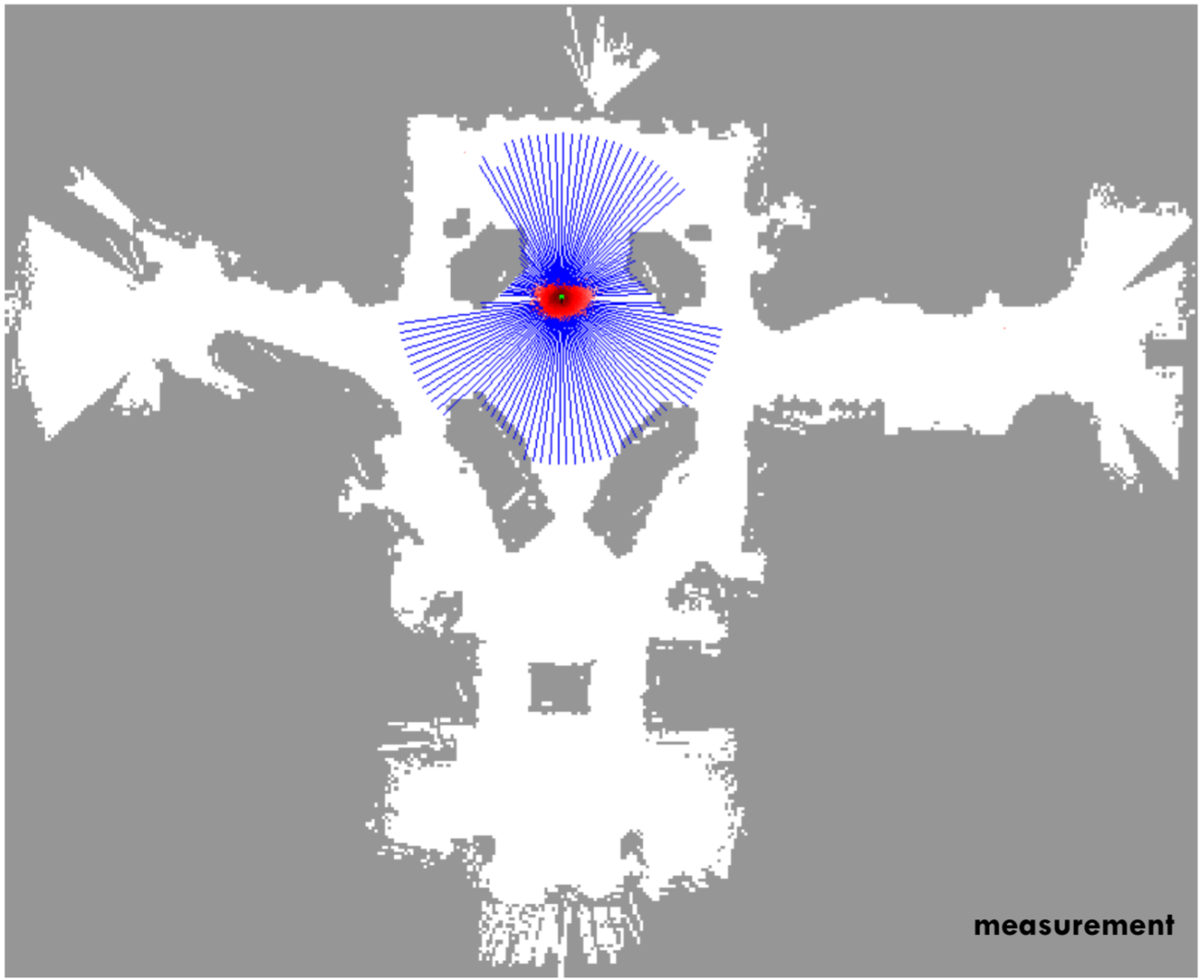
Courtesy: Thrun, Burgard, Fox

Example



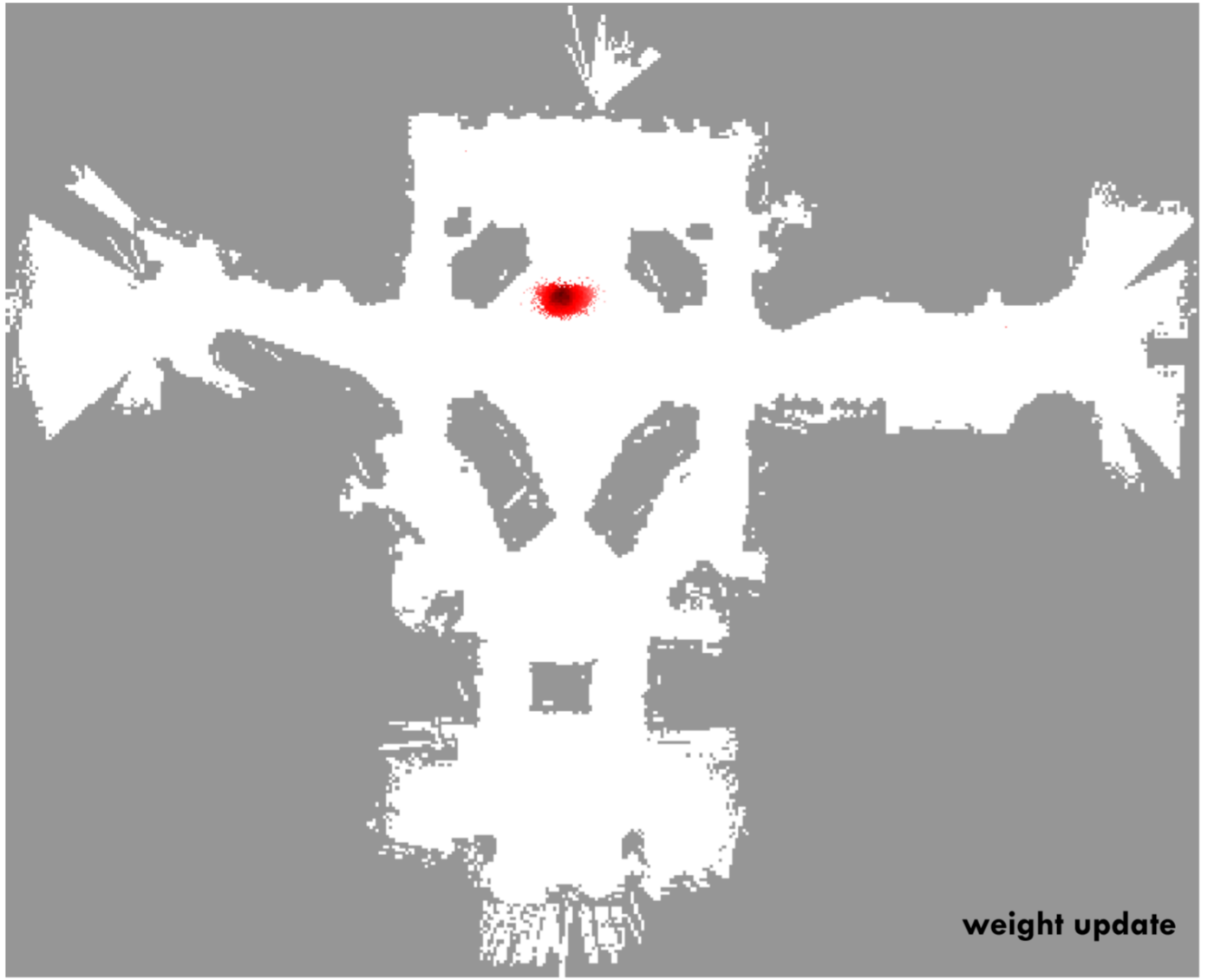
Courtesy: Thrun, Burgard, Fox

Example



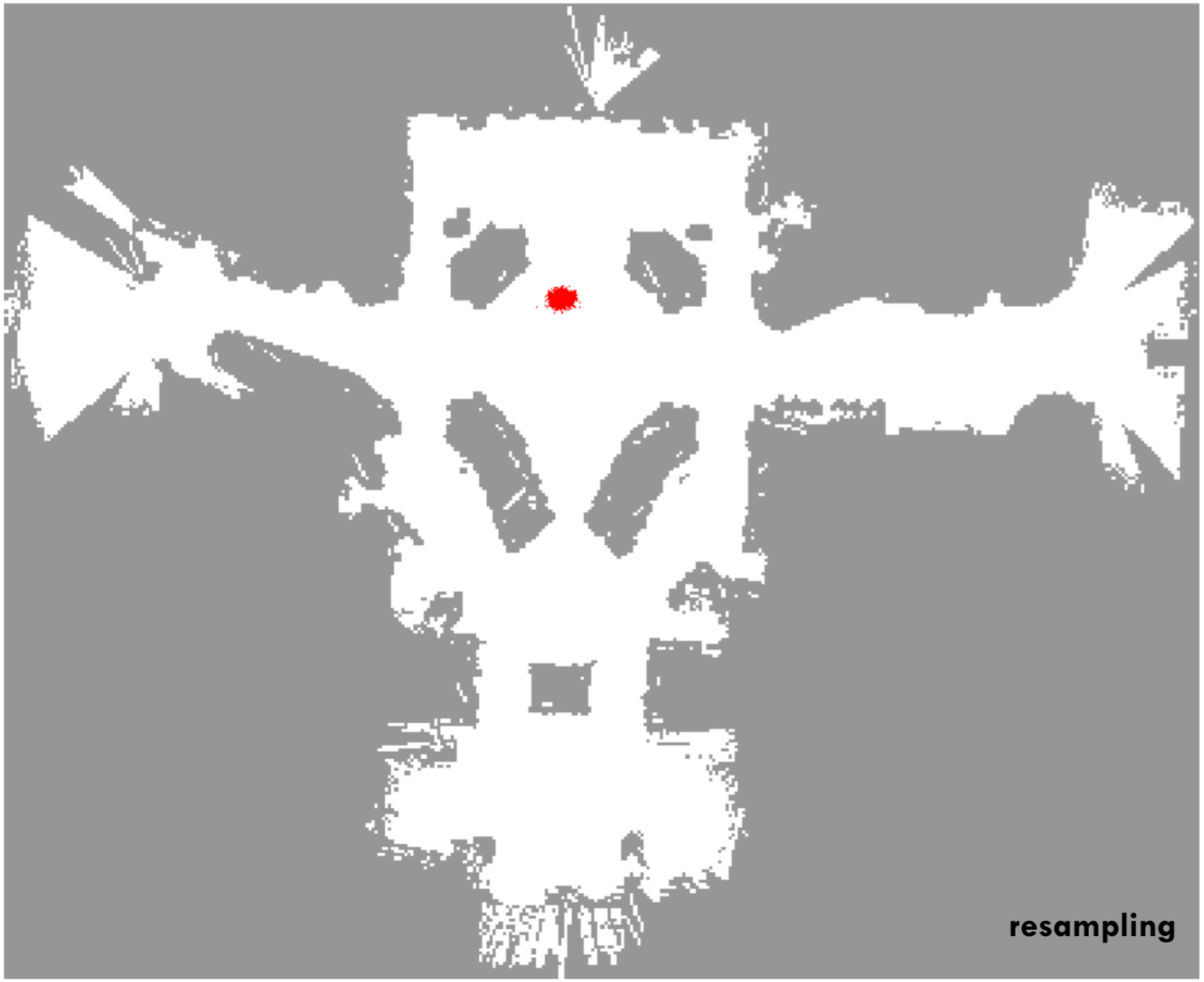
Courtesy: Thrun, Burgard, Fox

Example



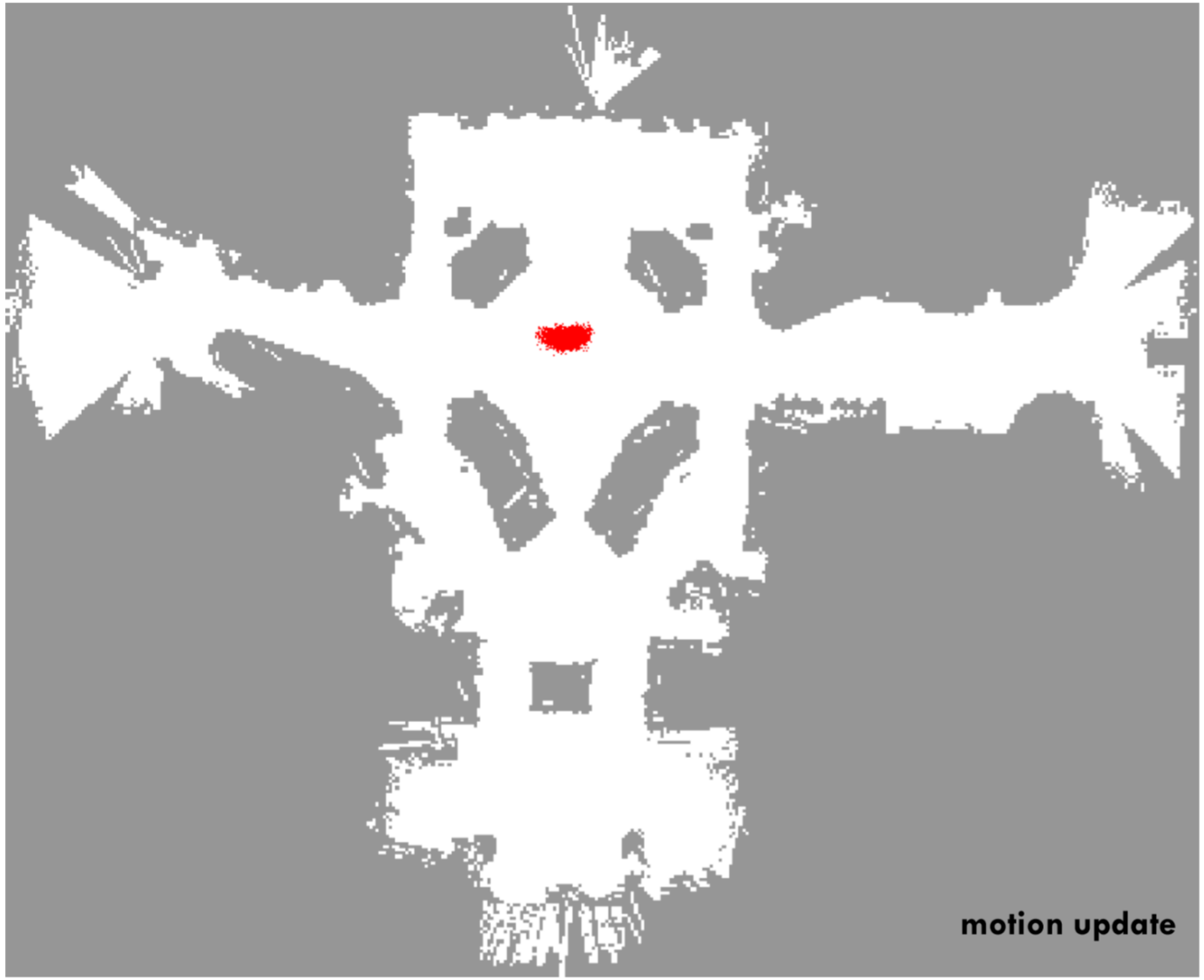
Courtesy: Thrun, Burgard, Fox

Example



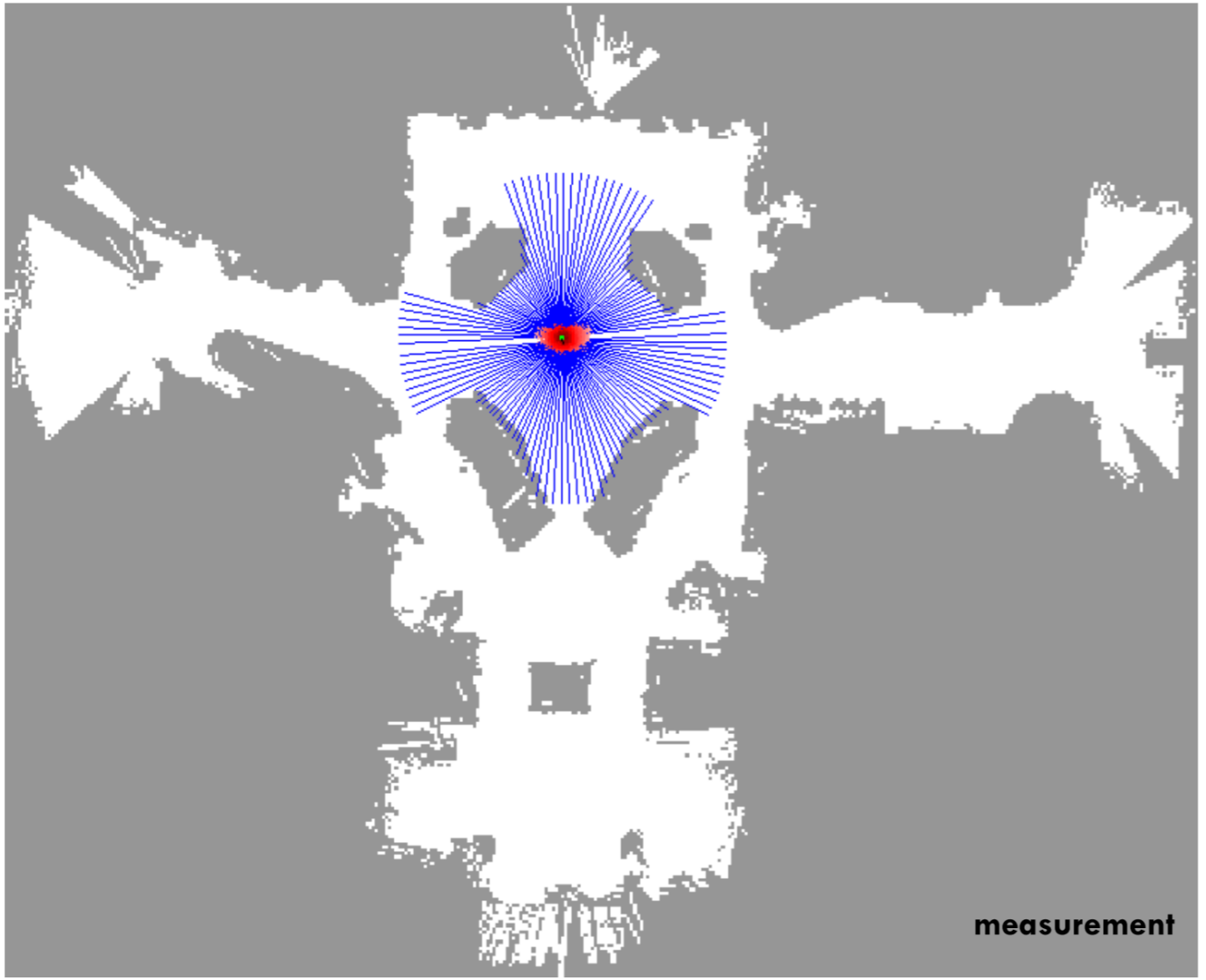
Courtesy: Thrun, Burgard, Fox

Example



Courtesy: Thrun, Burgard, Fox

Example



Courtesy: Thrun, Burgard, Fox

Summary – Particle Filters

- Particle Filters are non-parameteric, 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
- Works well in low-dimensional spaces

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 mobile robot localization today