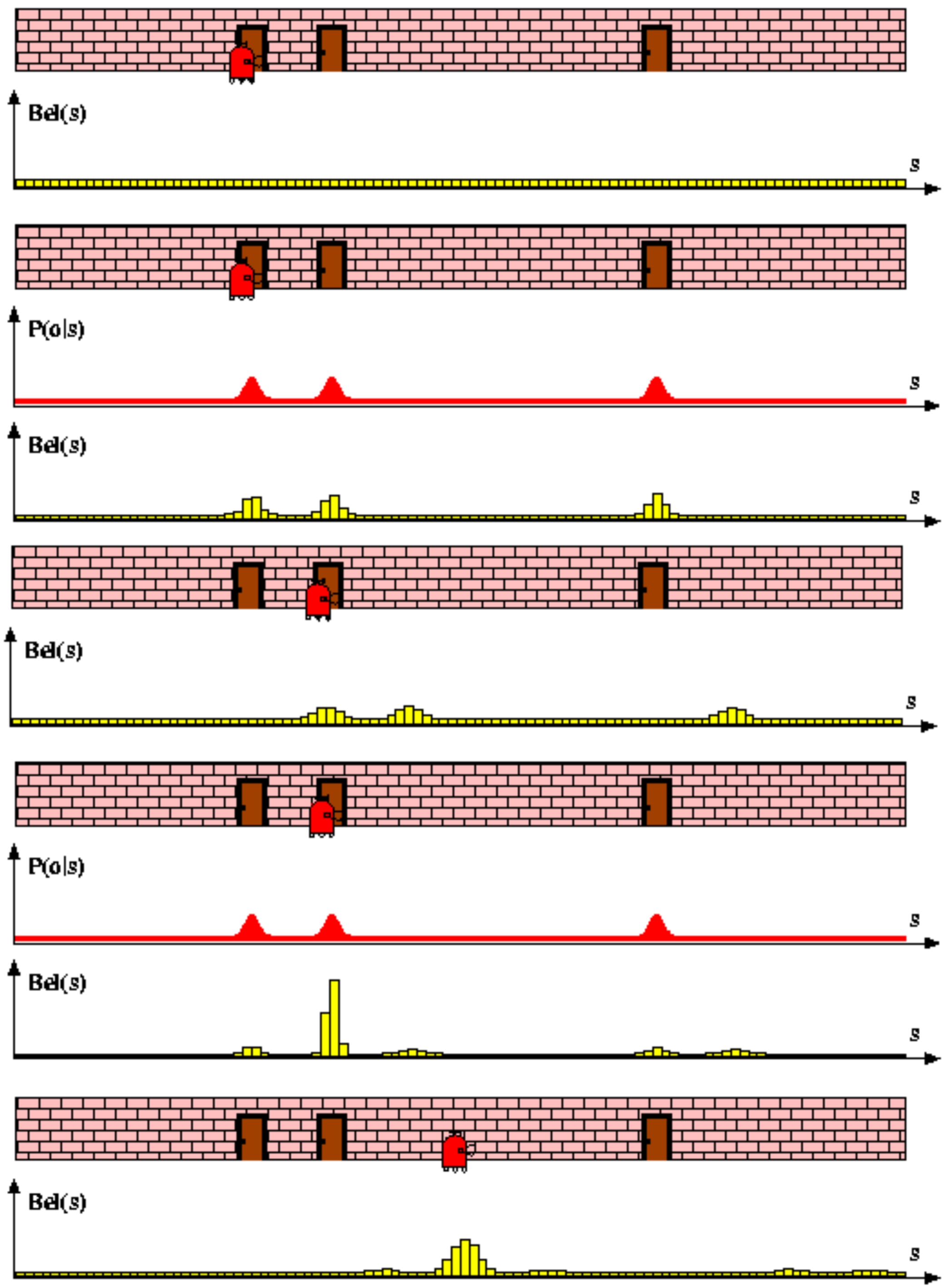
Probabilistic Robotics

Bayes Filter Implementations

Discrete filters, Particle filters

Piecewise Constant

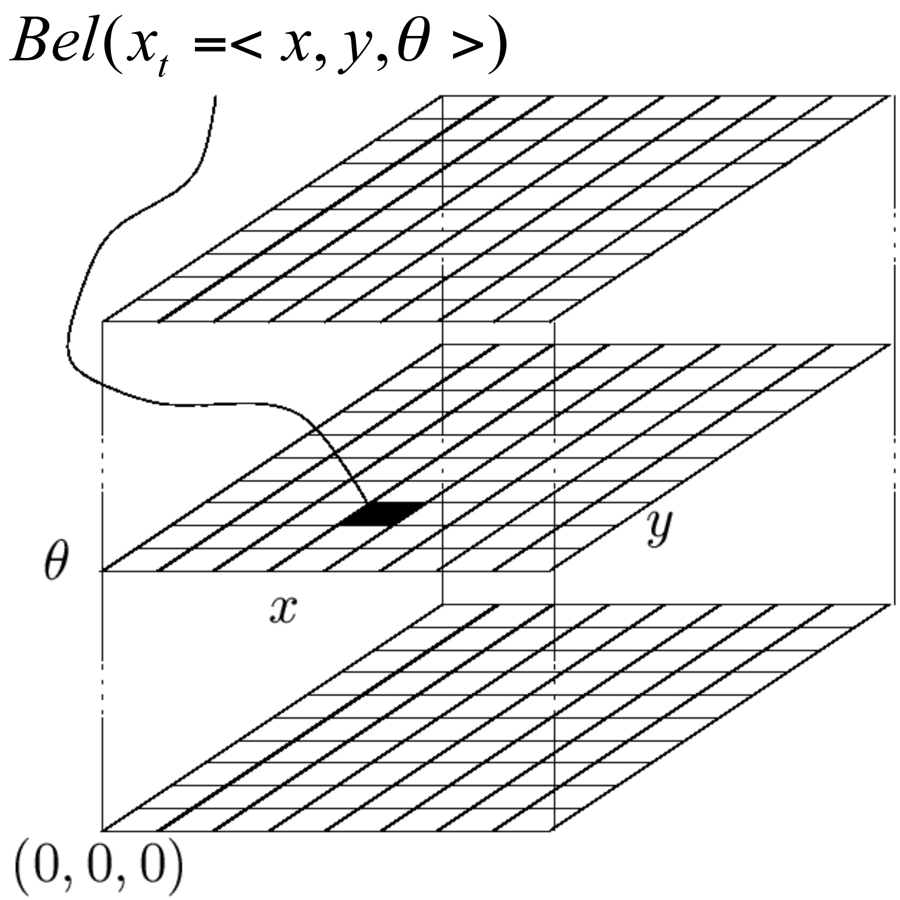
- Representation of belief



Discrete Bayes Filter Algorithm

1. Algorithm **Discrete_Bayes_filter**(*Bel(x),d*):
2. η=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)*

Piecewise Constant Representation



Implementation (1)

- To update the belief upon sensory input and to carry out the normalization one has to iterate over all cells of the grid.
- Especially when the belief is peaked (which is generally the case during position tracking), one wants to avoid updating irrelevant aspects of the state space.
- One approach is not to update entire sub-spaces of the state space.
- This, however, requires to monitor whether the robot is de-localized or not.
- To achieve this, one can consider the likelihood of the observations given the active components of the state space.

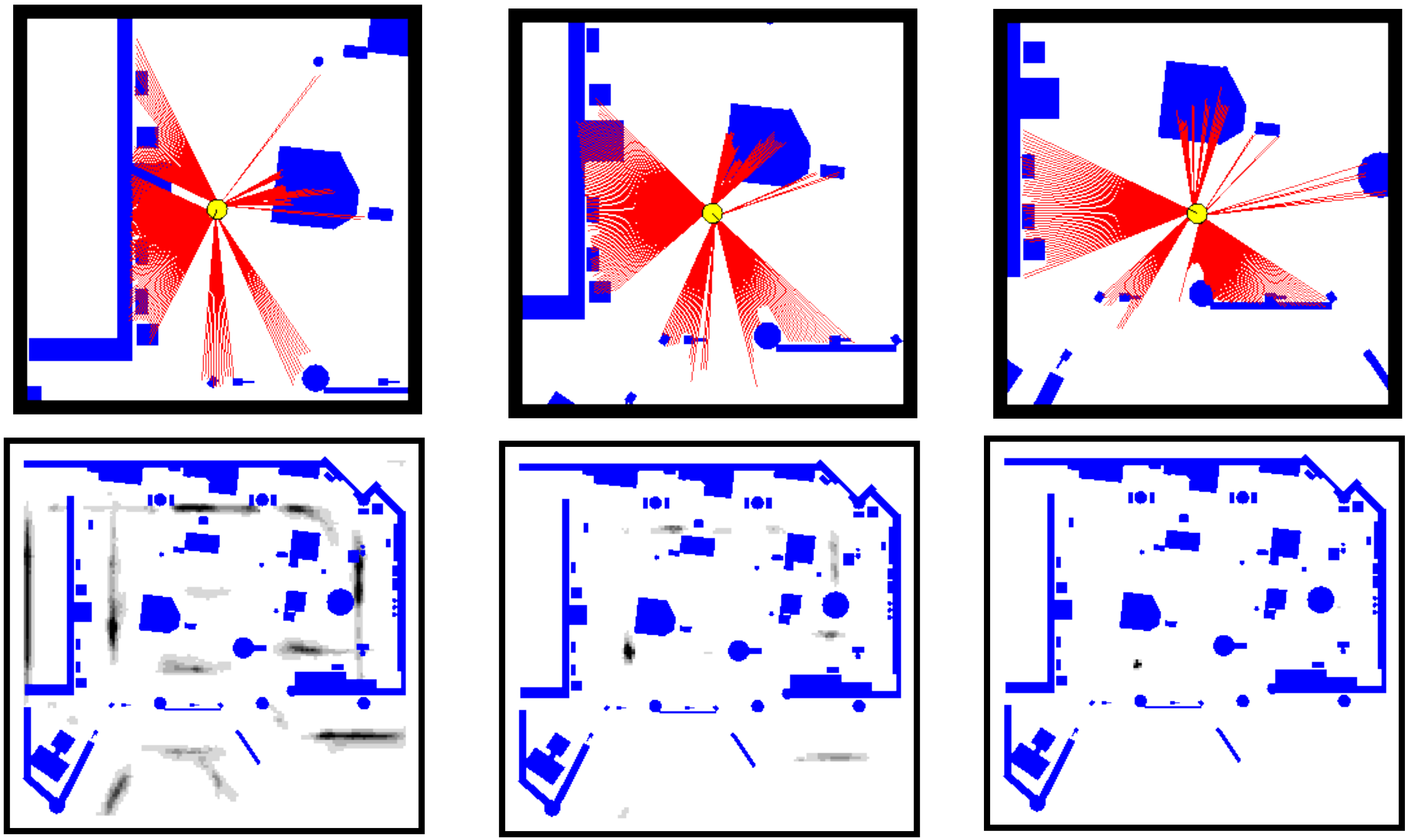
Implementation (2)

- To efficiently update the belief upon robot motions, one typically assumes a bounded Gaussian model for the motion uncertainty.
- This reduces the update cost from $O(n^2)$ to $O(n)$, where n is the number of states.
- The update can also be realized by shifting the data in the grid according to the measured motion.
- In a second step, the grid is then convolved using a separable Gaussian Kernel.
- Two-dimensional example:

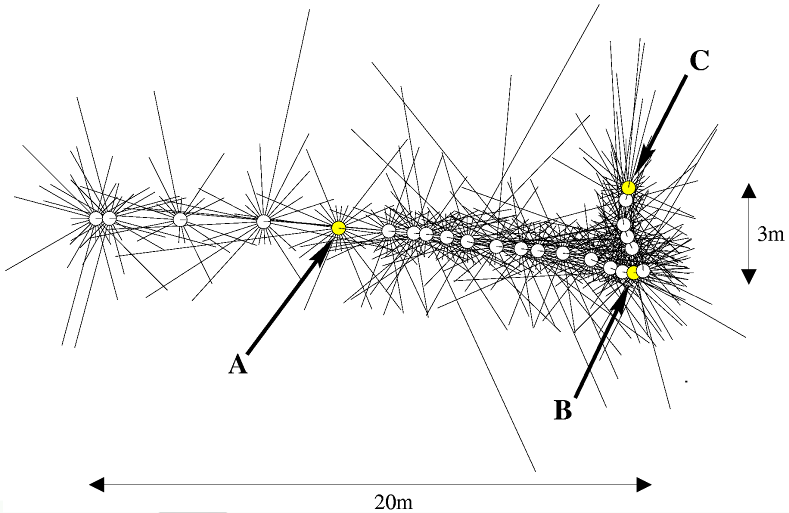
$$\begin{array}{|c|c|c|}\hline 1/16 & 1/8 & 1/16 \\ \hline 1/8 & 1/4 & 1/8 \\ \hline 1/16 & 1/8 & 1/16 \\ \hline \end{array} \cong \begin{array}{|c|}\hline 1/4 \\ \hline 1/2 \\ \hline 1/4 \\ \hline \end{array} + \begin{array}{|c|c|c|}\hline 1/4 & 1/2 & 1/4 \\ \hline \end{array}$$

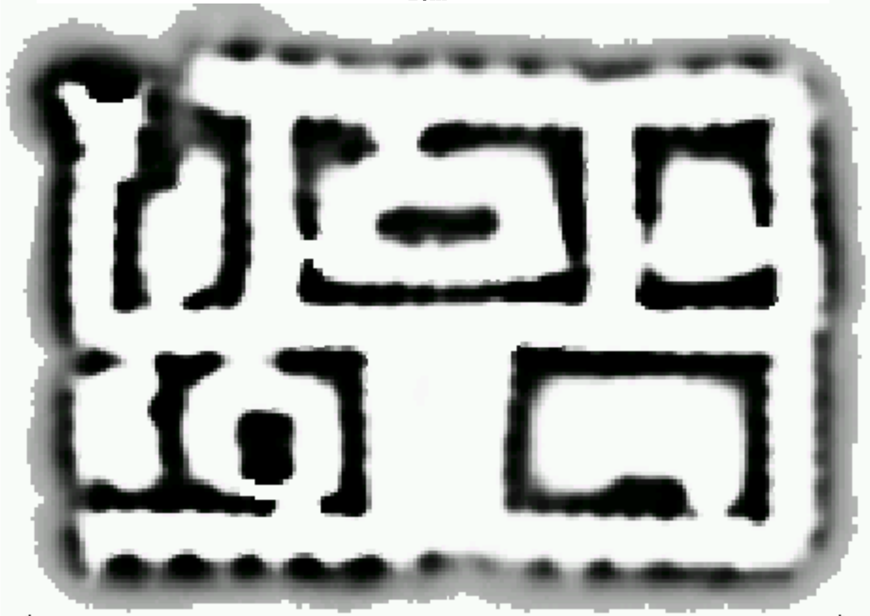
  - Fewer arithmetic operations
  - Easier to implement

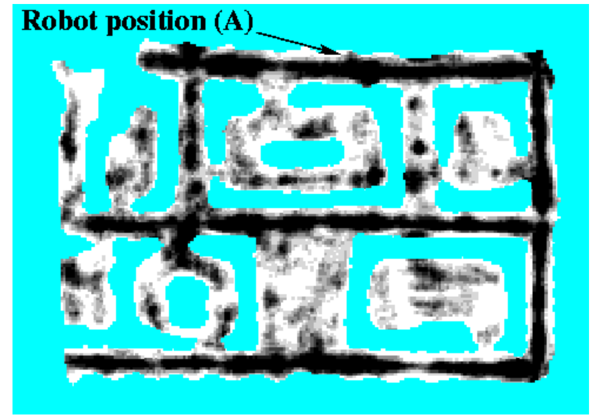
Grid-based Localization

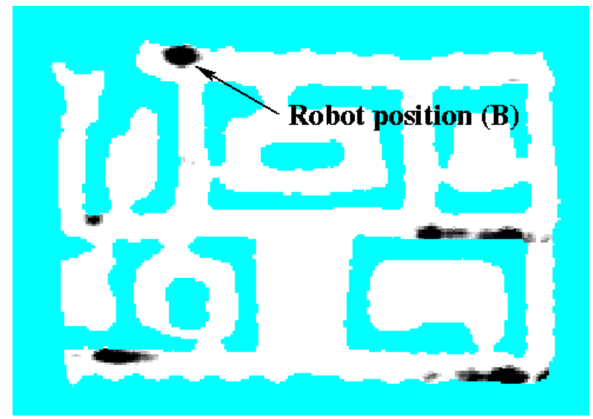


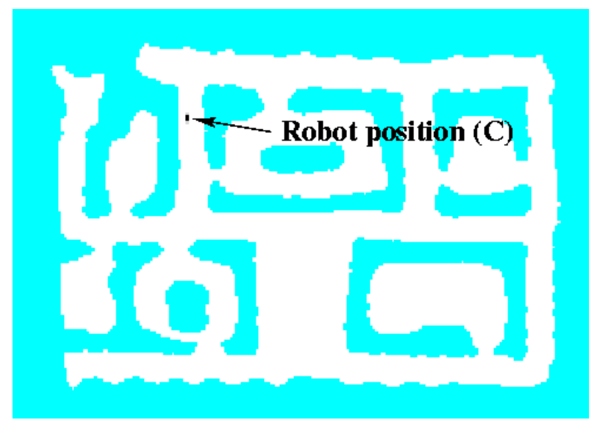
Sonars and Occupancy Grid Map





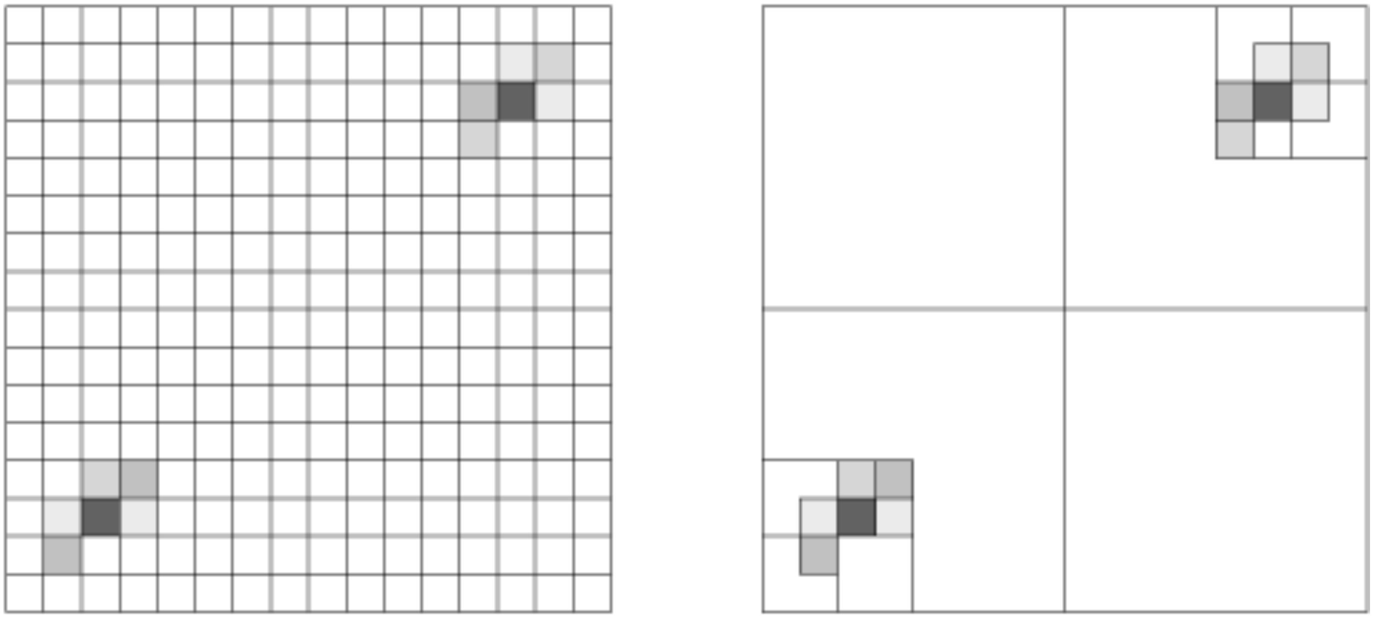






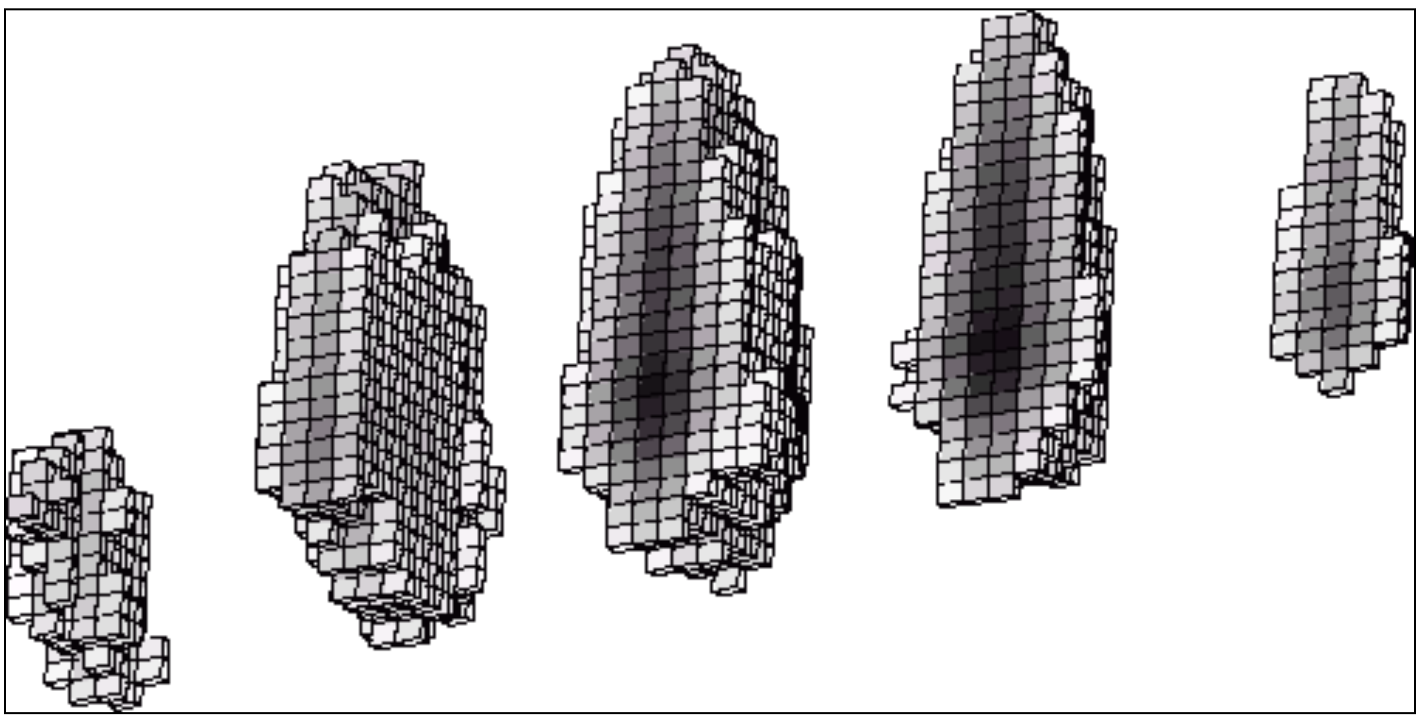
Tree-based Representation

Idea: Represent density using a variant of octrees

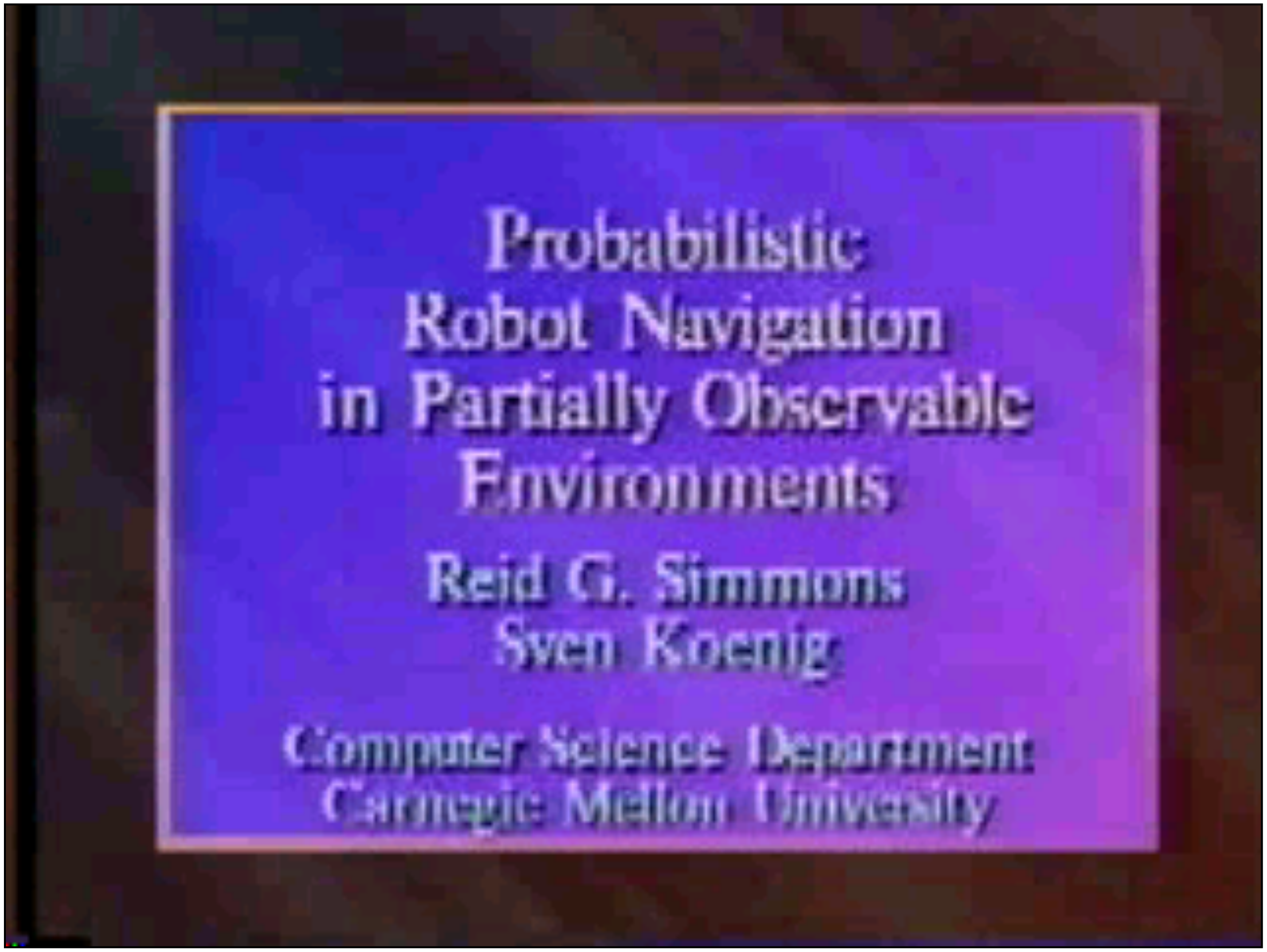


Tree-based Representations

- Efficient in space and time
- Multi-resolution

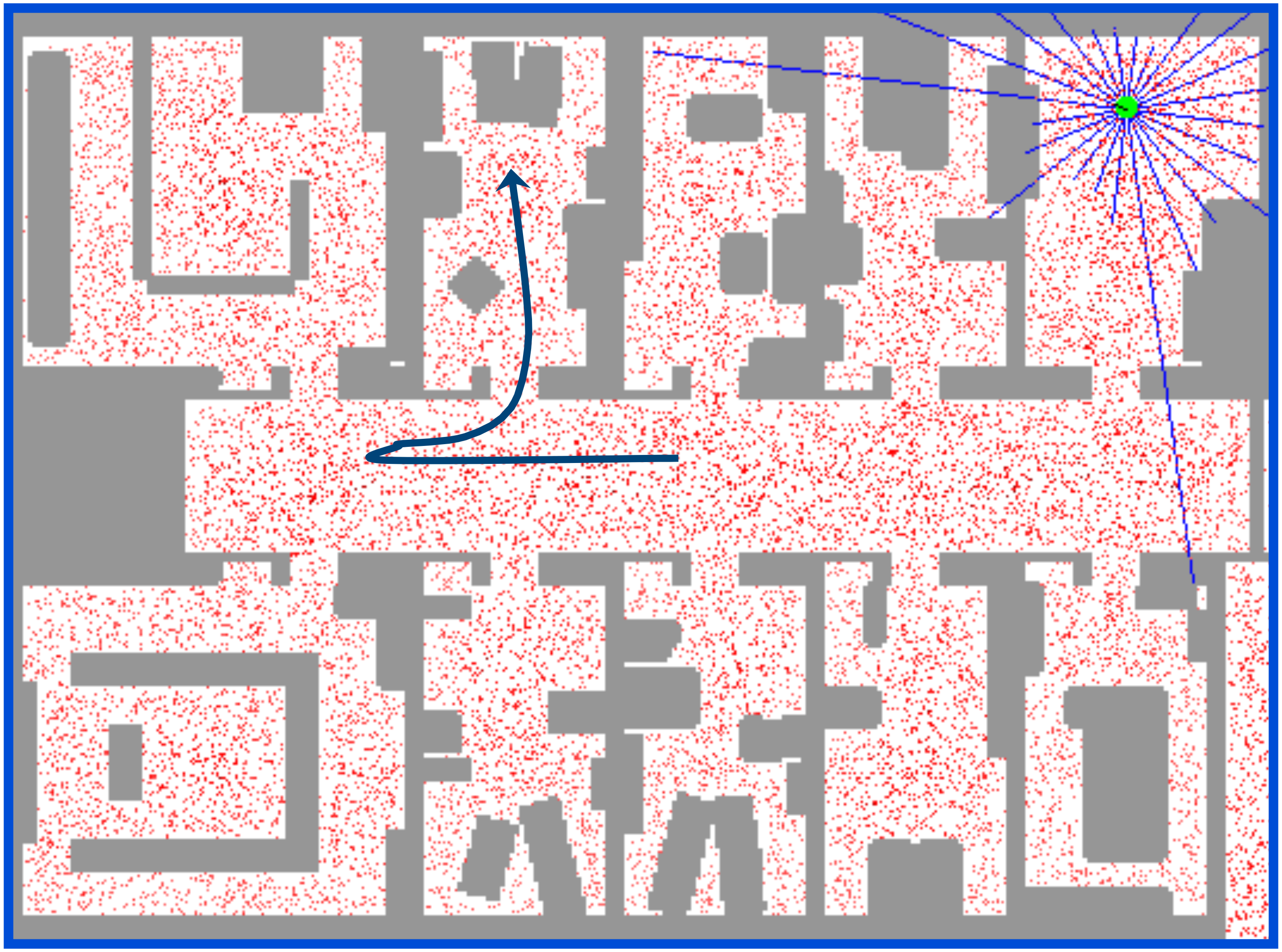


Xavier:
Localization in a Topological Map



[Courtesy of Reid Simmons]

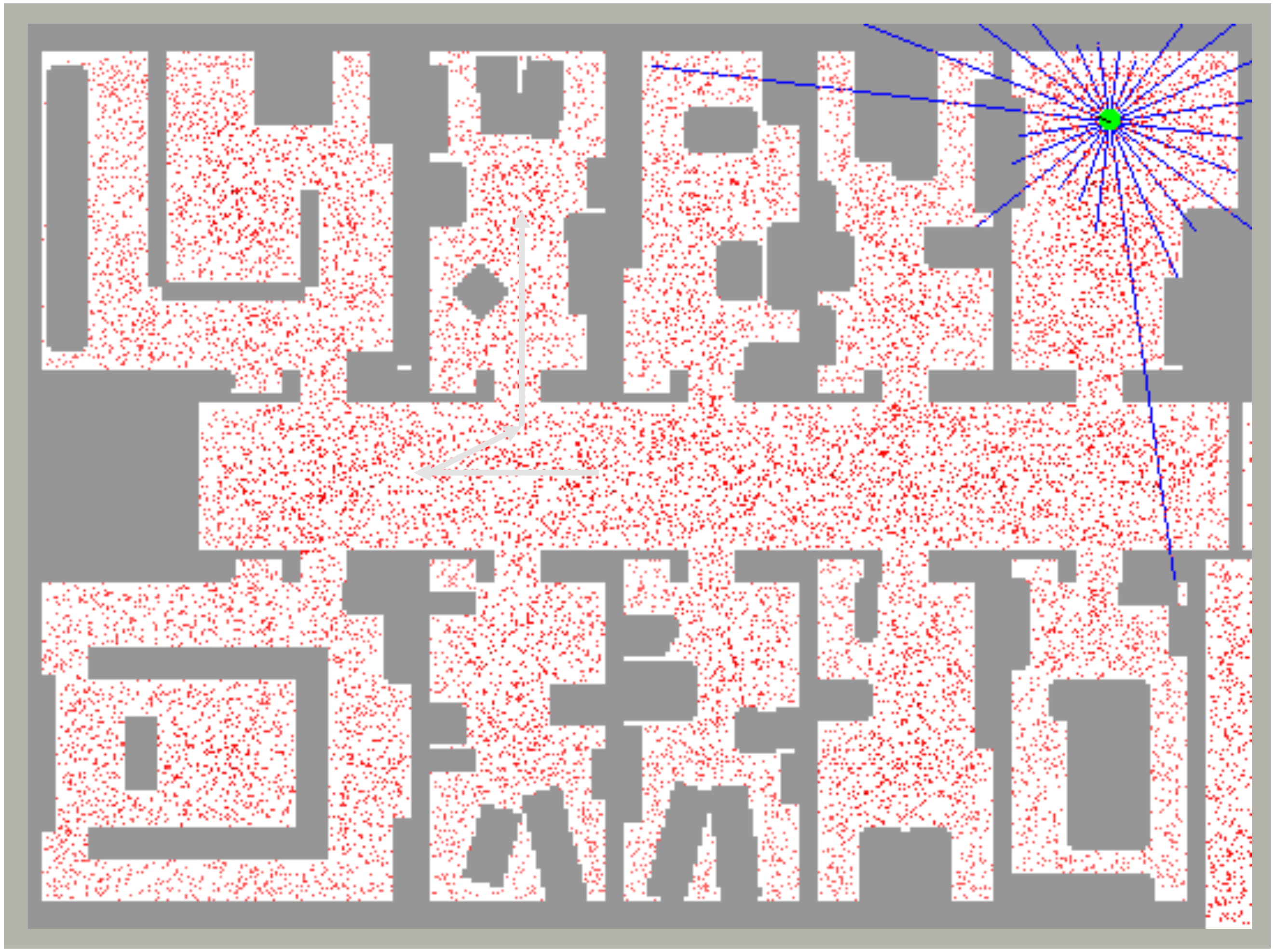
Sample-based Localization (sonar)



Markov ⇔ Kalman Filter Localization

- Markov localization
  - localization starting from any unknown position
  - recovers from ambiguous situation.
  - However, to update the probability of all positions within the whole state space at any time requires a discrete representation of the space (grid). The required memory and calculation power can thus become very important if a fine grid is used.
- Kalman filter localization
  - tracks the robot and is inherently very precise and efficient.
  - However, if the uncertainty of the robot becomes to large (e.g. collision with an object) the Kalman filter will fail and the position is definitively lost.

Monte Carlo Localization



Particle Filters

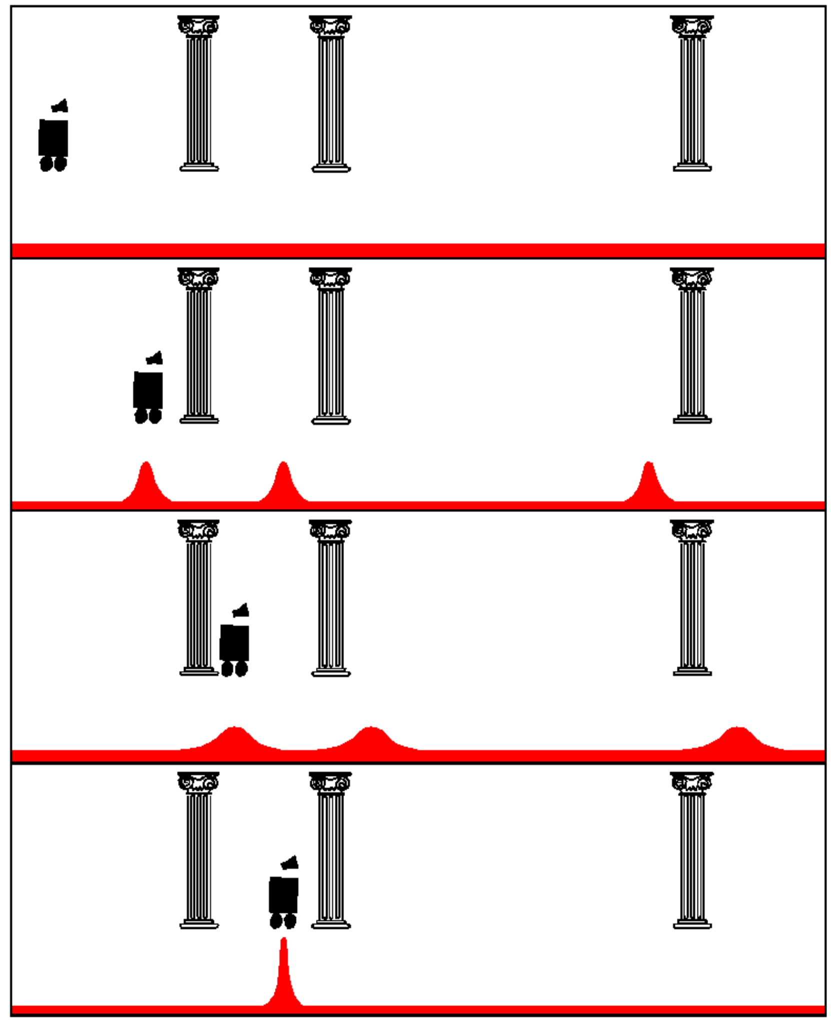
- Represent belief by random samples
- Estimation of non-Gaussian, nonlinear processes

- Monte Carlo filter, Survival of the fittest, Condensation, Bootstrap filter, Particle filter

- Filtering: [Rubin, 88], [Gordon et al., 93], [Kitagawa 96]
- Computer vision: [Isard and Blake 96, 98]
- Dynamic Bayesian Networks: [Kanazawa et al., 95]d

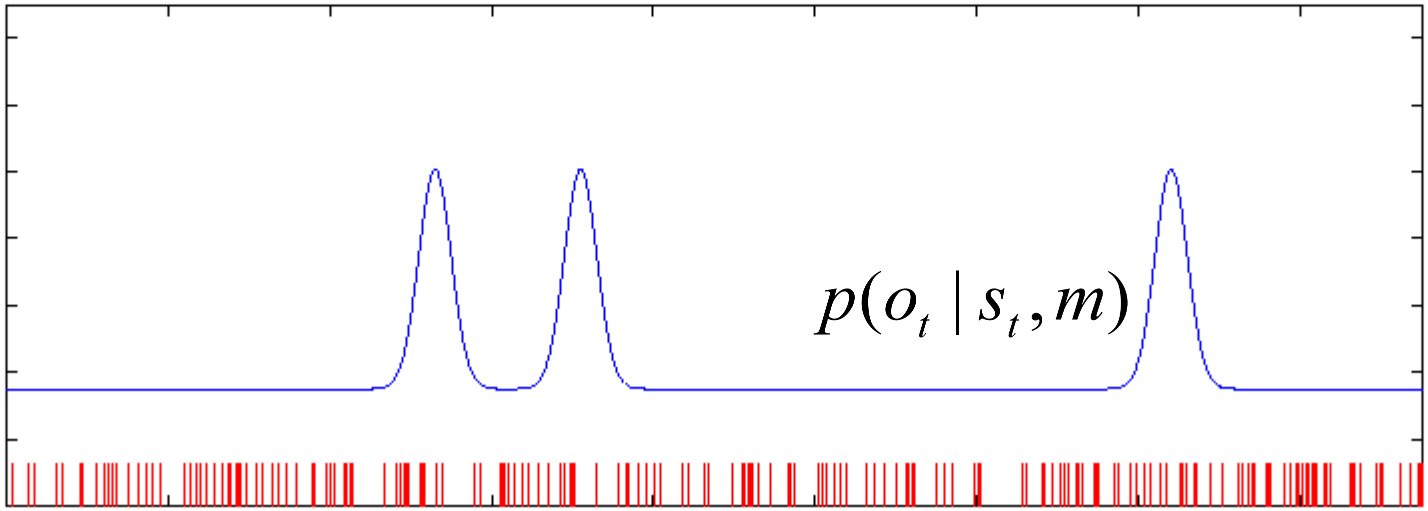
Markov Localization: Case Study 2 – Grid Map (3)

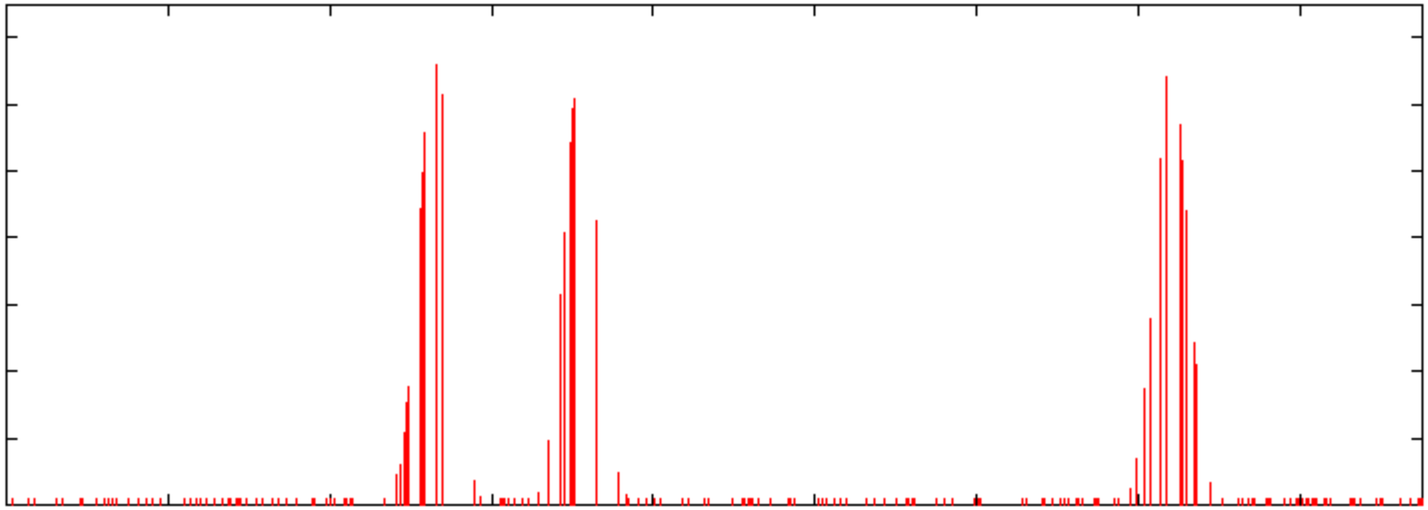
- The 1D case
  1. Start
     - No knowledge at start, thus we have an uniform probability distribution.
  2. Robot perceives first pillar
     - Seeing only one pillar, the probability being at pillar 1, 2 or 3 is equal.
  3. Robot moves
     - Action model enables to estimate the new probability distribution based on the previous one and the motion.
  4. Robot perceives second pillar
     - Base on all prior knowledge the probability being at pillar 2 becomes dominant

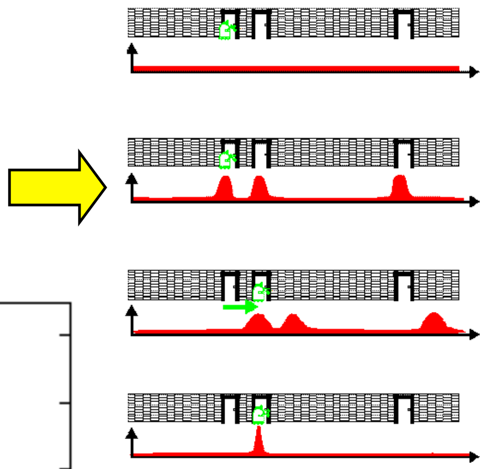


MCL: Importance Sampling

$$b(s_t) \leftarrow \eta\, p(o_t \mid s_t, m)\, b(s_t)$$

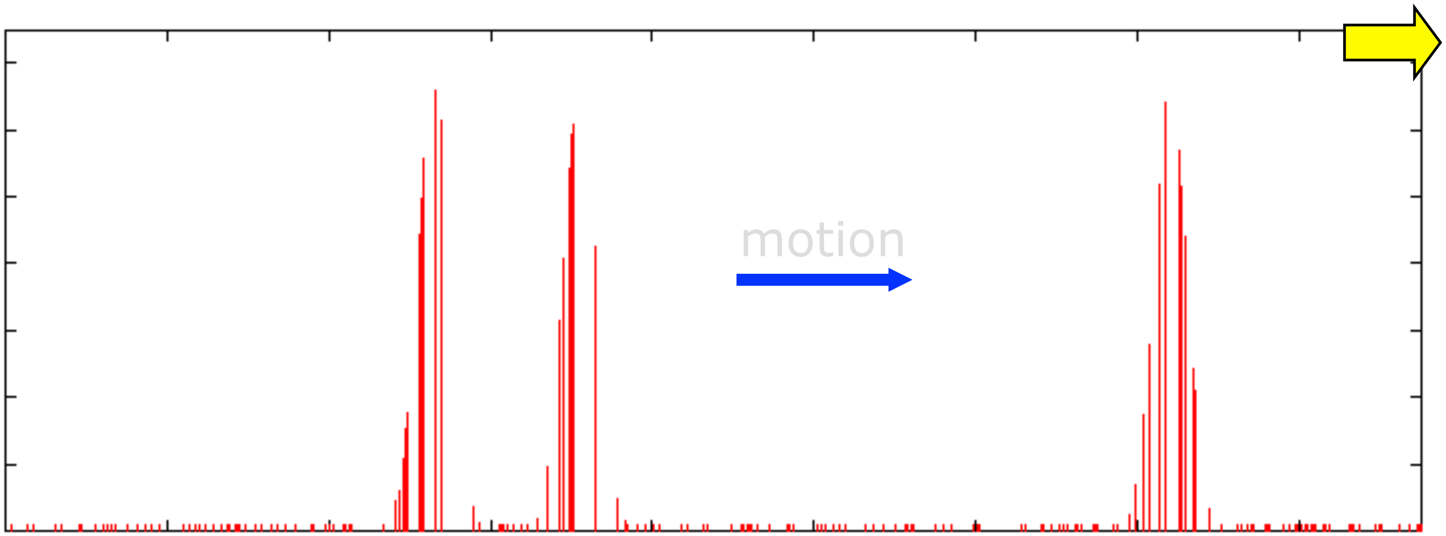


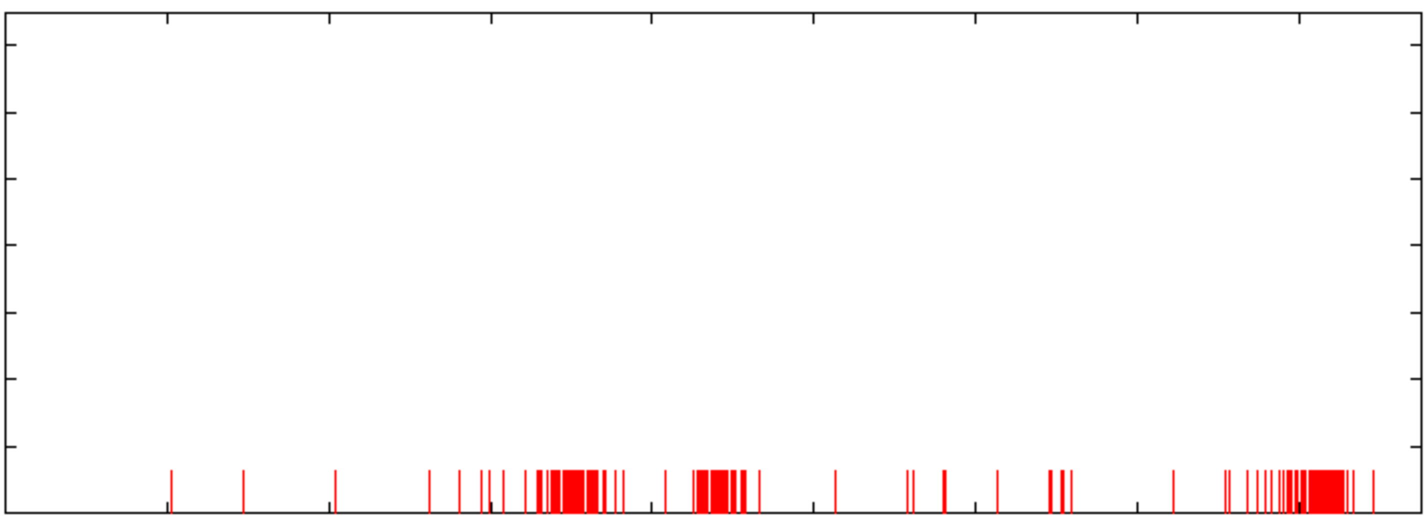


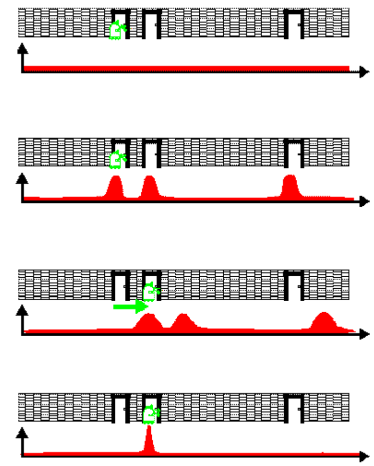


MCL: Robot Motion

$$b(s_{t+1}) \leftarrow \int p(s_{t+1} \mid a_t, s_t, m)\, b(s_t) \; \mathrm{d}s_t$$

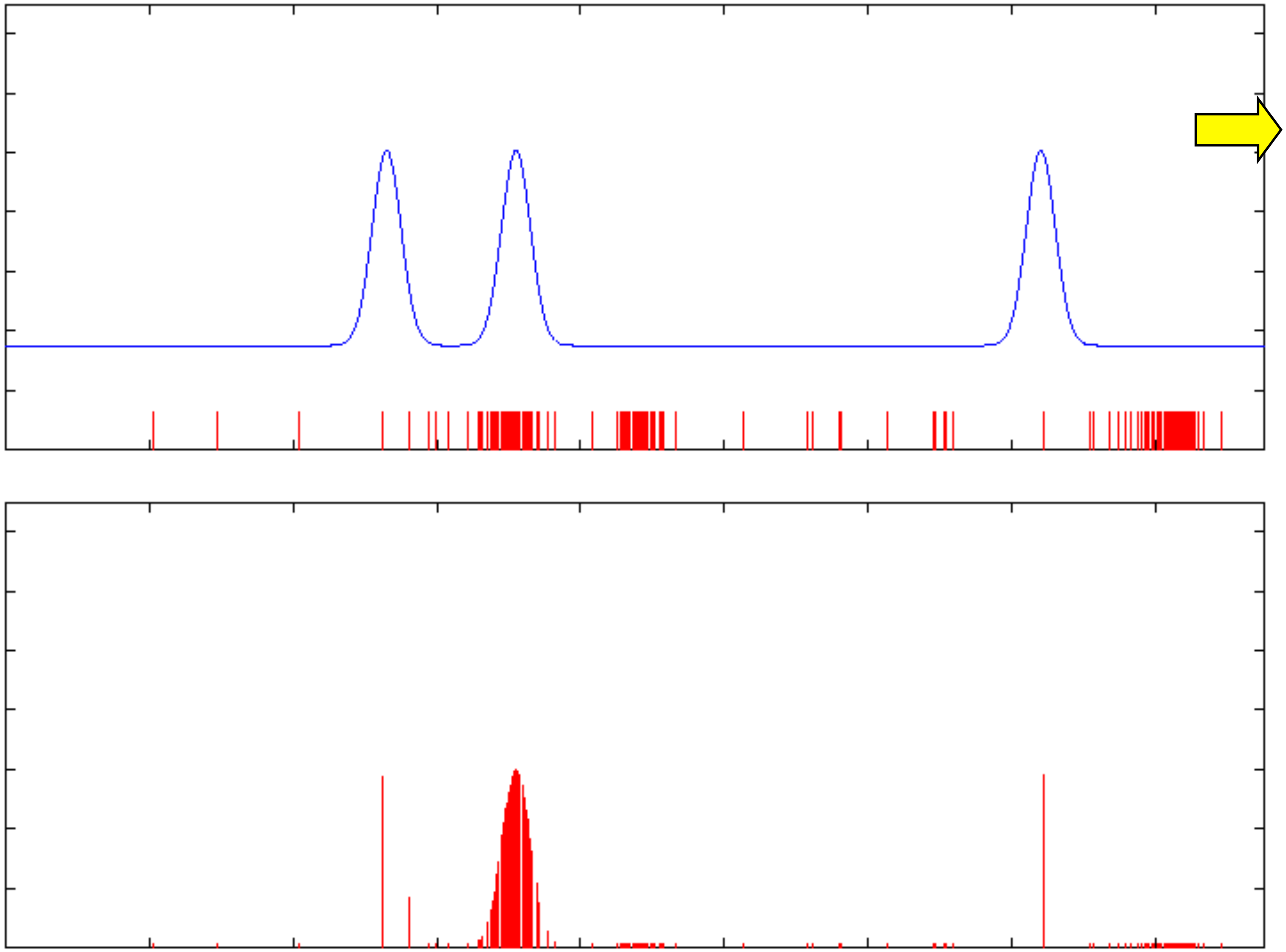


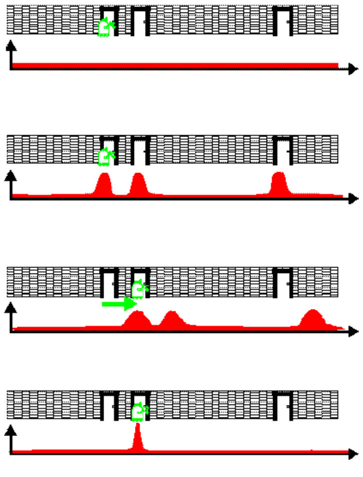




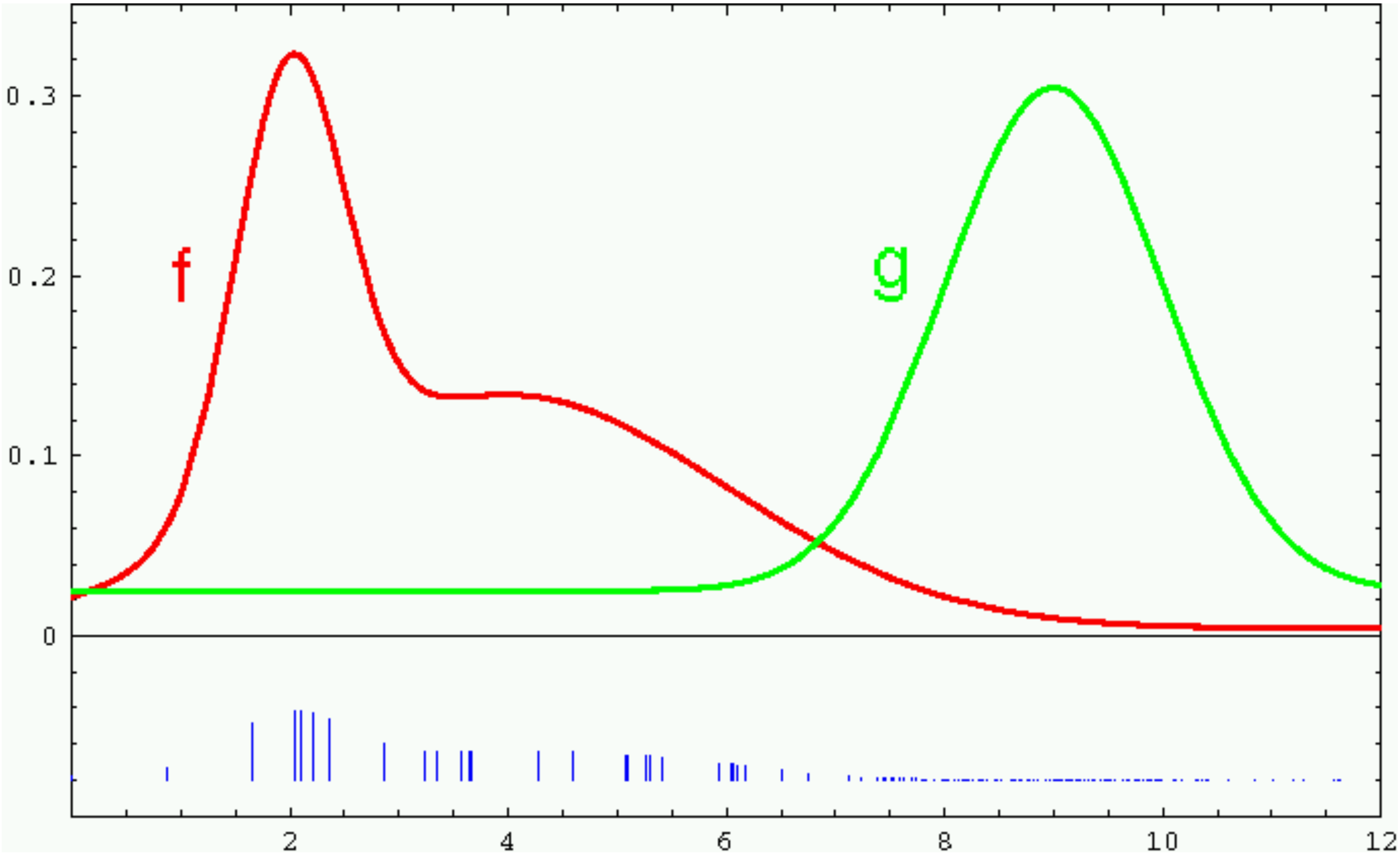
MCL: Importance Sampling

$$b(s_{t+1}) \leftarrow \eta \, p(o_{t+1} \mid s_{t+1}, m) \, b(s_{t+1})$$





Importance Sampling



Weight samples: $w = f / g$

In particle filters f corresponds to Bel(x(t)) and g to predicted belief x(t)

Importance Sampling with Resampling: Landmark Detection Example

Probabilistic Model

1. Algorithm **landmark_detection_model**(z,x,m):
   $z = \langle i, d, \alpha \rangle, x = \langle x, y, \theta \rangle$
2. $\hat{d} = \sqrt{(m_x(i) - x)^2 + (m_y(i) - y)^2}$
3. $\hat{a} = \mathrm{atan2}(m_y(i) - y, m_x(i) - x) - \theta$
4. $p_{\mathrm{det}} = \mathrm{prob}(\hat{d} - d, \varepsilon_d) \cdot \mathrm{prob}(\hat{\alpha} - \alpha, \varepsilon_\alpha)$
5. Return $z_{\mathrm{det}} p_{\mathrm{det}} + z_{\mathrm{fp}} P_{\mathrm{uniform}}(z \mid x, m)$

Computing likelihood of landmark measurement

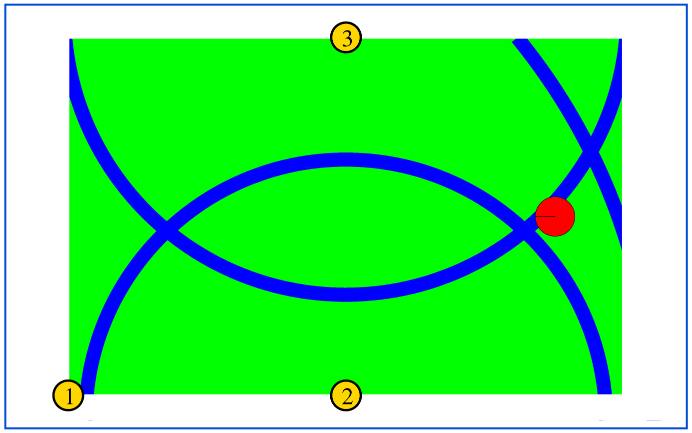
Landmark – distance, bearing and signature

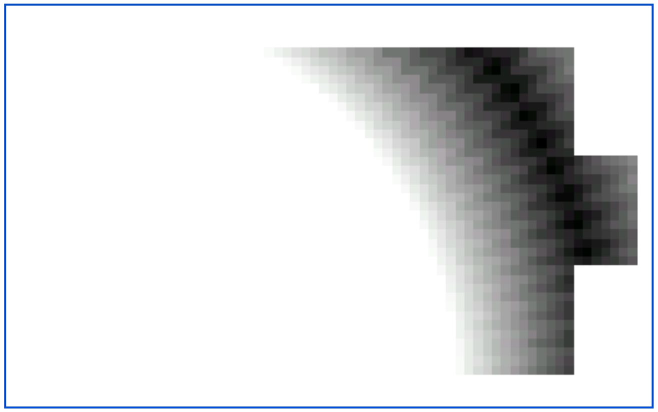
Landmark model

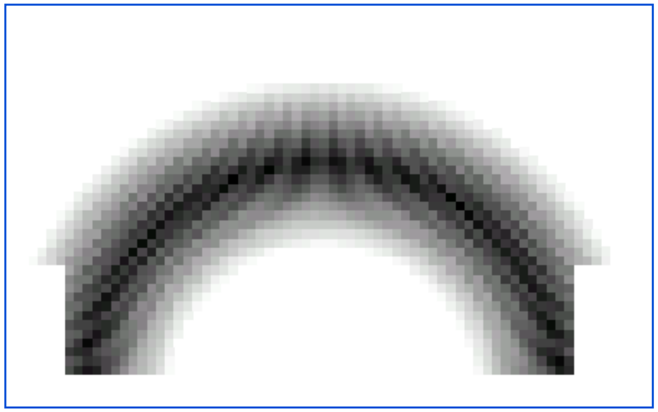
- How to sample likely poses, given the landmark measurement ?
- Given a landmark measurement, robot can lie on a circle. Generate samples along the circle.

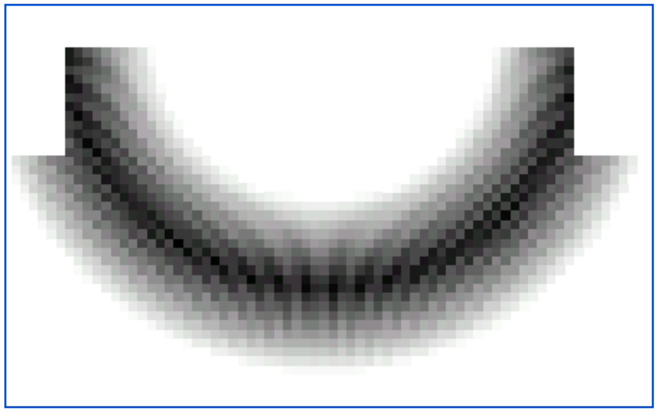
- Idea: use the inverse sensor model. Given distance and bearing (possibly noisy), generate samples along circle.
- Do it for every landmark

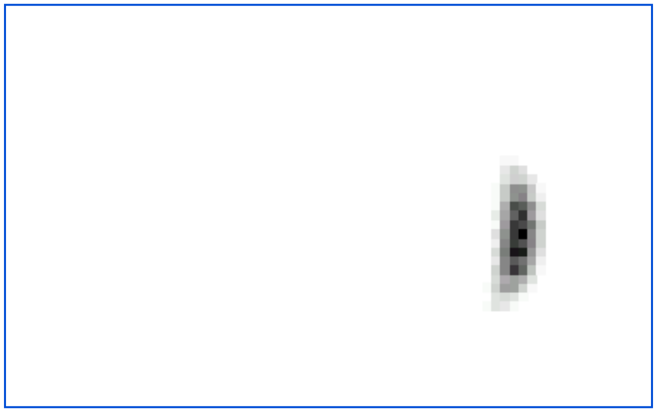
Distributions



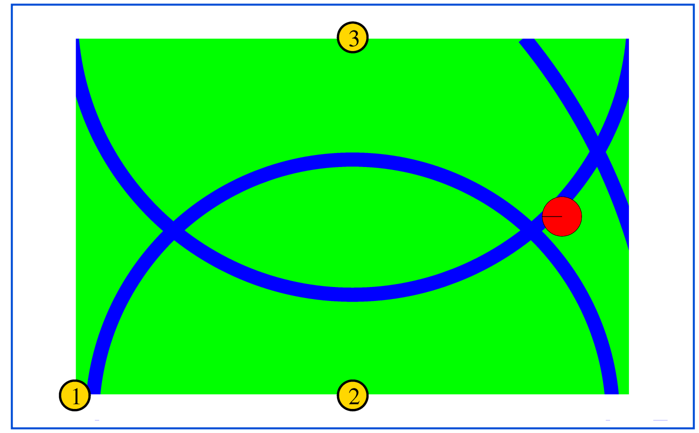


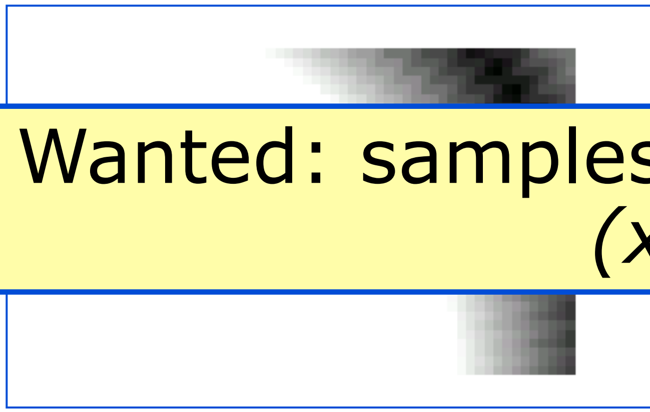


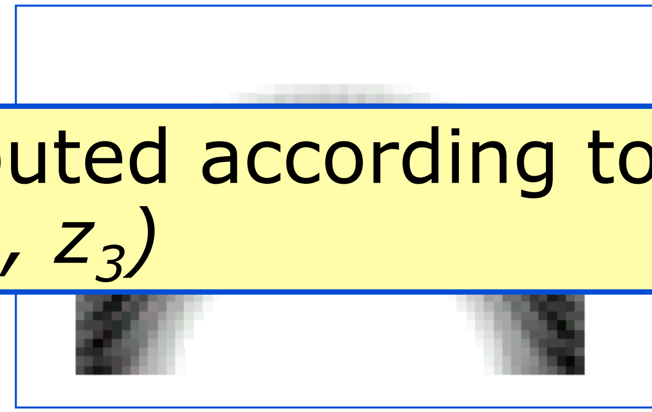




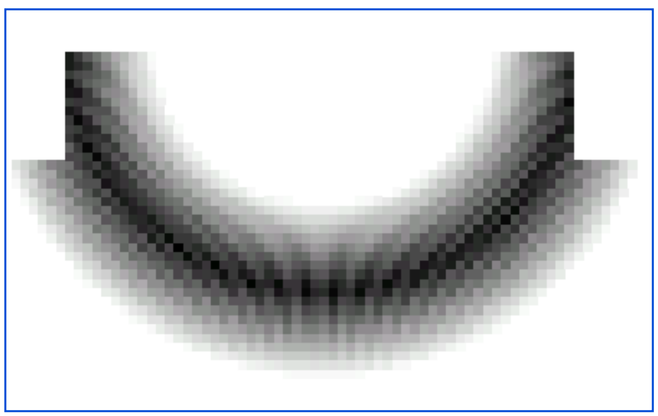
Distributions





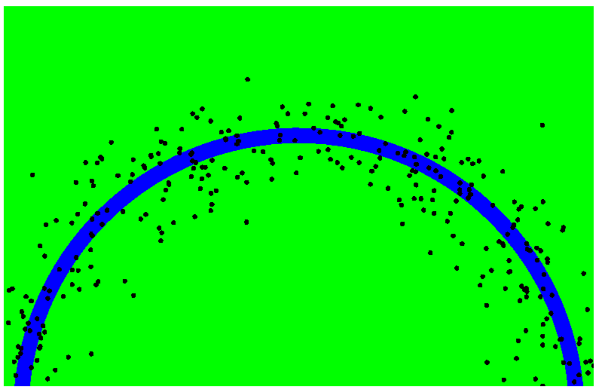


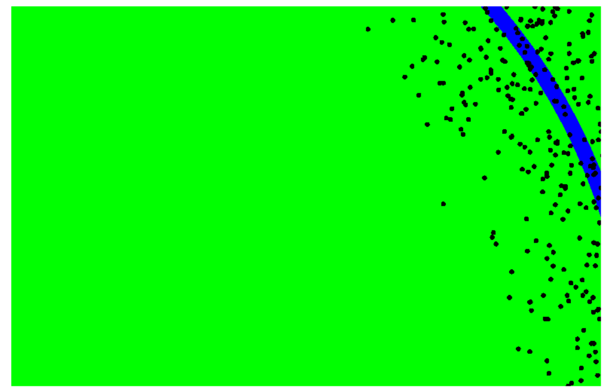
Wanted: samples distributed according to $p(x \mid z_1, z_2, z_3)$

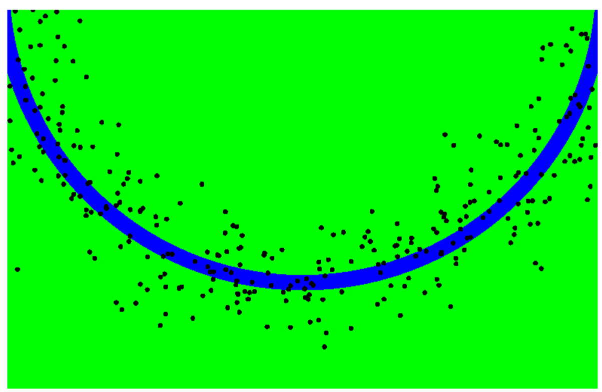


This is Easy!

We can draw samples from $p(x|z_l)$ by adding noise to the detection parameters.







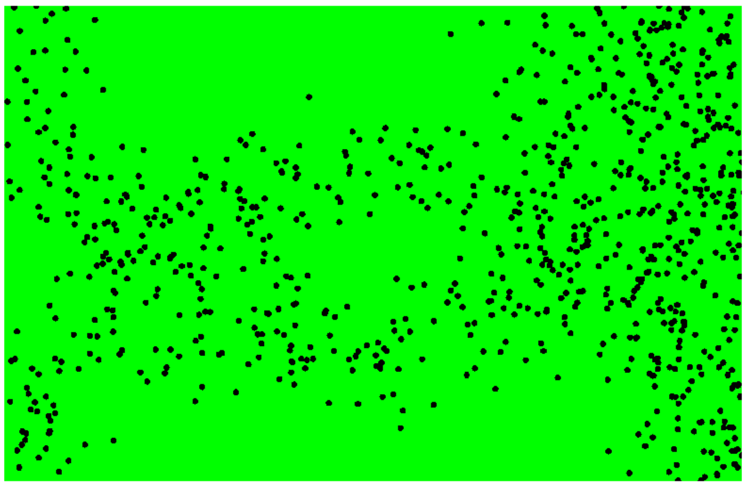
Importance Sampling with Resampling

$$\text{Target distribution f} : p(x \mid z_1, z_2, \ldots, z_n) = \frac{\prod_k p(z_k \mid x) \quad p(x)}{p(z_1, z_2, \ldots, z_n)}$$

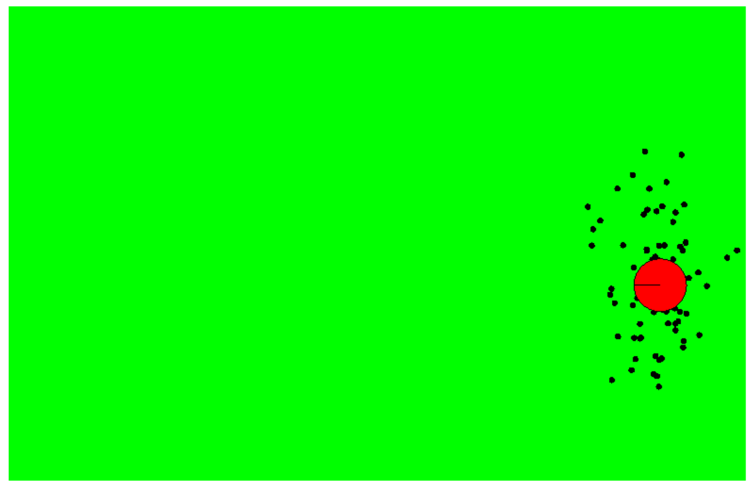
$$\text{Sampling distribution g} : p(x \mid z_l) = \frac{p(z_l \mid x) p(x)}{p(z_l)}$$

$$\text{Importance weights w} : \frac{f}{g} = \frac{p(x \mid z_1, z_2, \ldots, z_n)}{p(x \mid z_l)} = \frac{p(z_l) \prod_{k \neq l} p(z_k \mid x)}{p(z_1, z_2, \ldots, z_n)}$$

Importance Sampling with Resampling

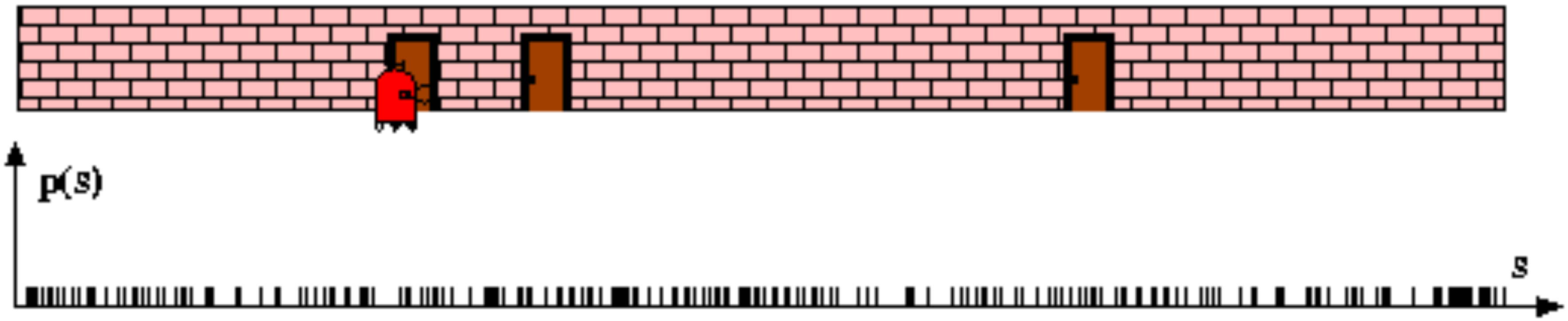


Weighted samples



After resampling

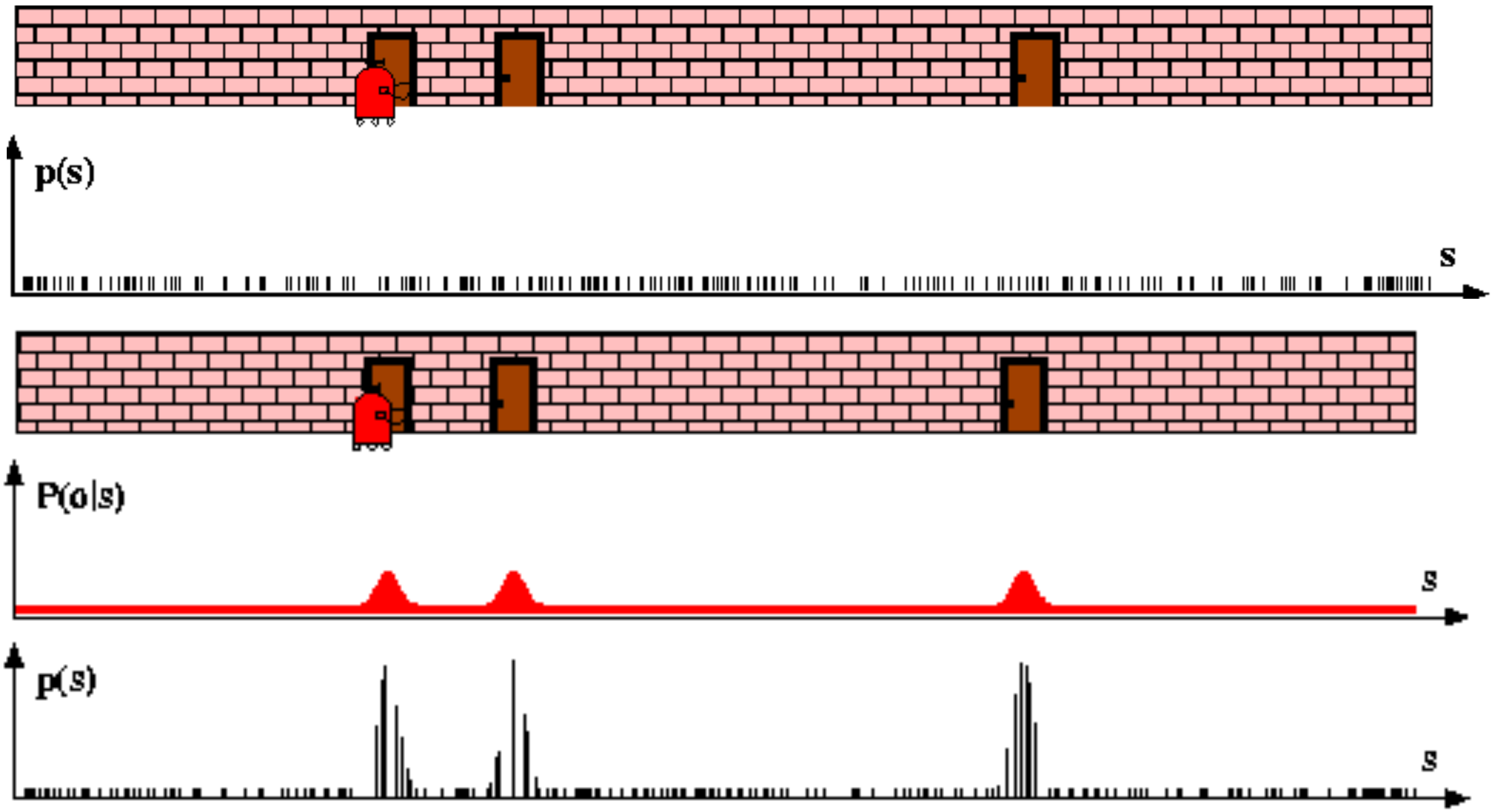
Particle Filters



Sensor Information: Importance Sampling

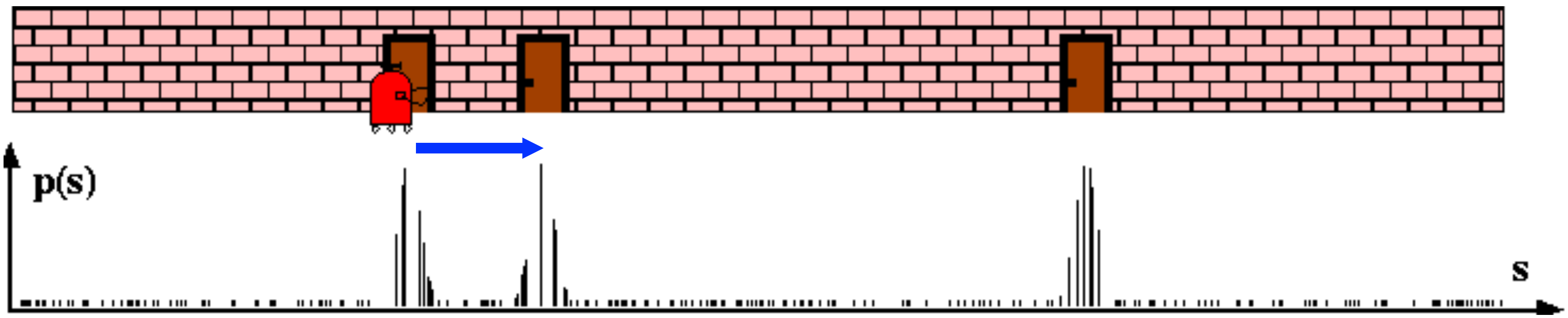
$$Bel(x) \quad \leftarrow \quad \alpha\ p(z\,|\,x)\,Bel^{-}(x)$$

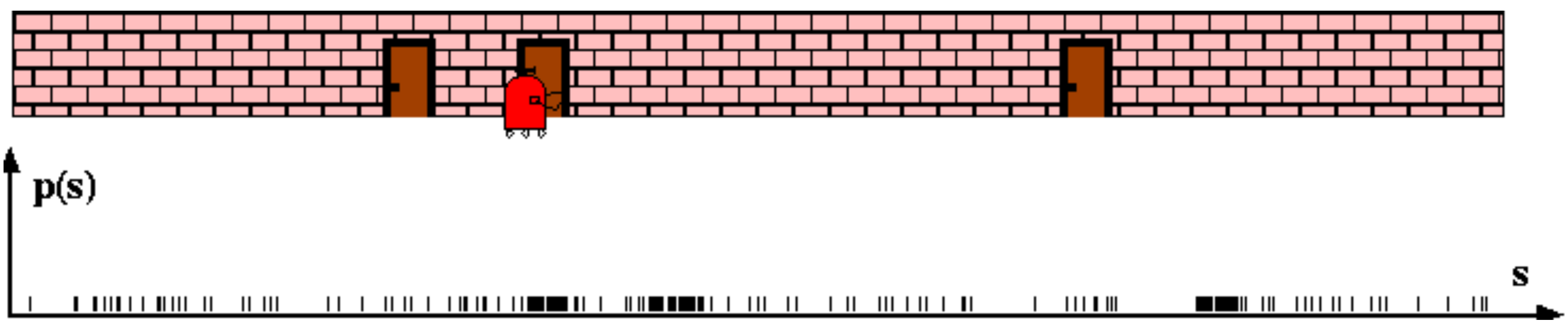
$$w \quad \leftarrow \quad \frac{\alpha\ p(z\,|\,x)\,Bel^{-}(x)}{Bel^{-}(x)} \quad = \quad \alpha\ p(z\,|\,x)$$



Robot Motion

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

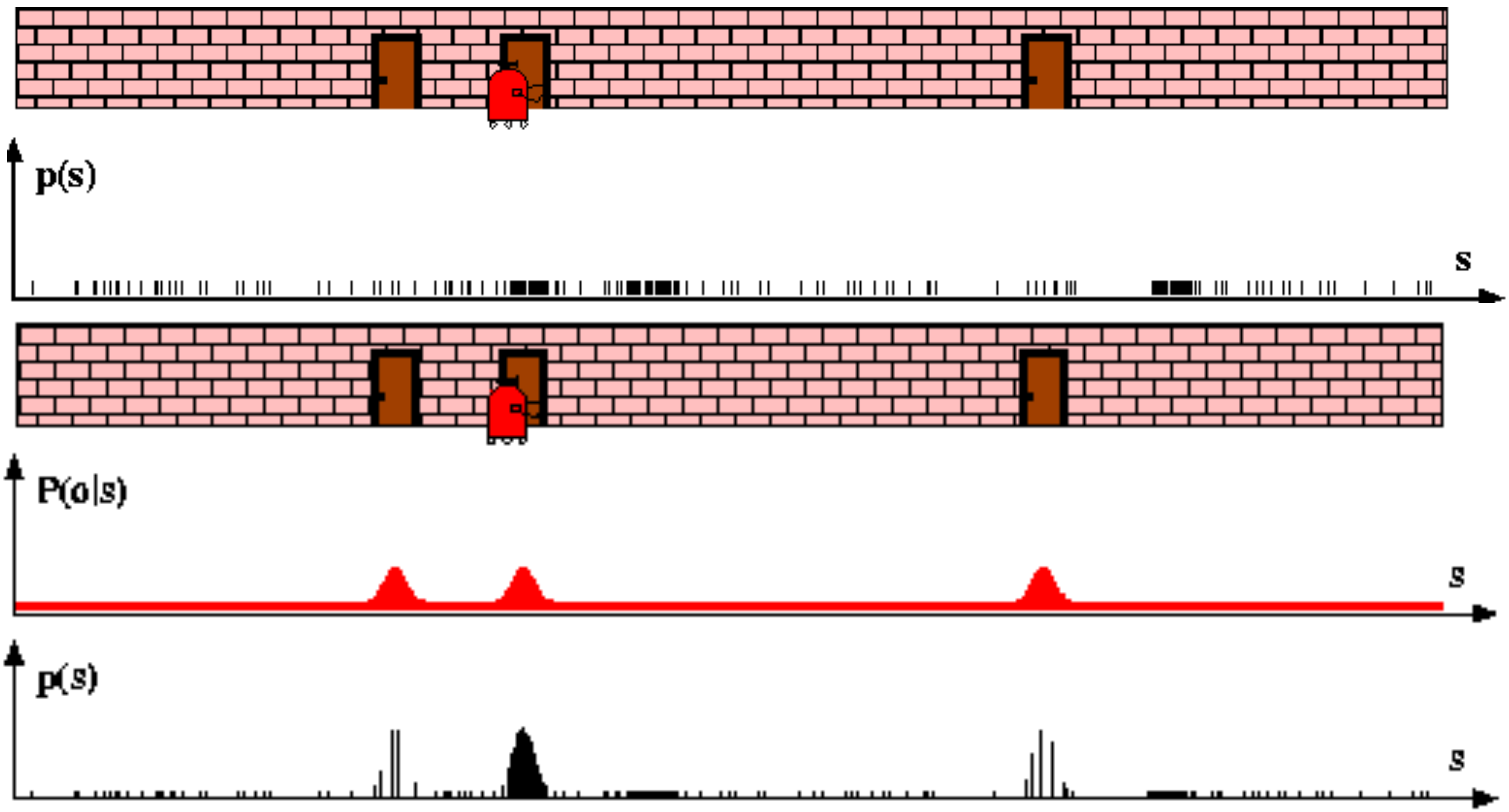




Sensor Information: Importance Sampling

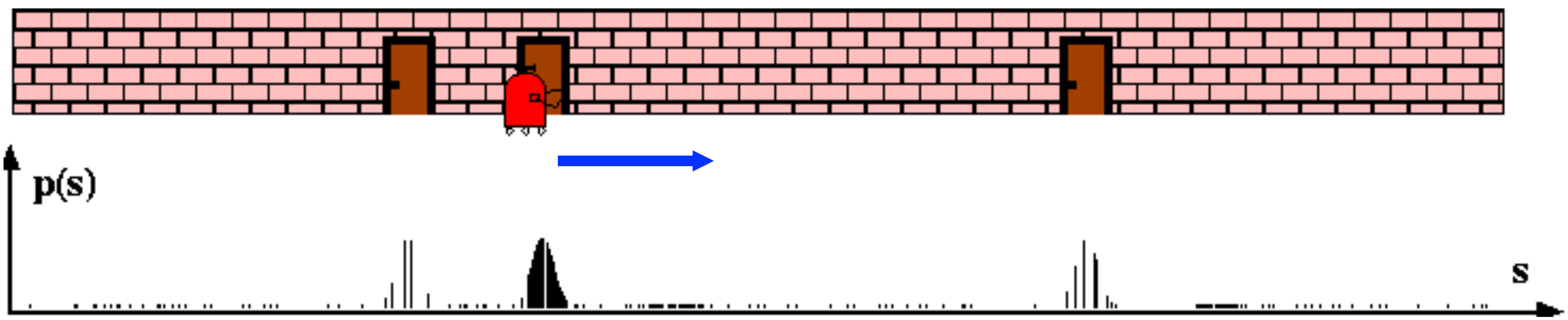
$$Bel(x) \leftarrow \alpha\ p(z \mid x)\ Bel^{-}(x)$$

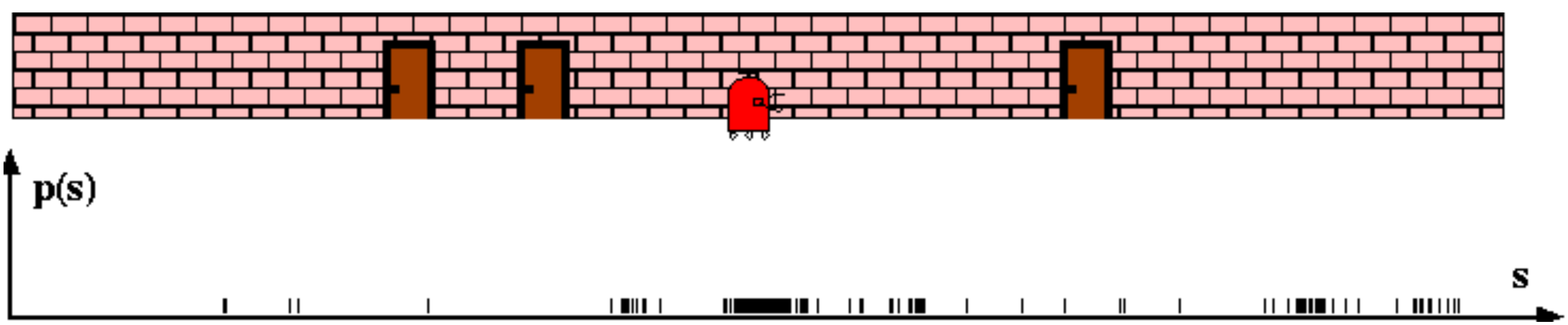
$$w \leftarrow \frac{\alpha\ p(z \mid x)\ Bel^{-}(x)}{Bel^{-}(x)} = \alpha\ p(z \mid x)$$



Robot Motion

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





Particle Filter Algorithm

Algorithm **particle_filter**(S_{t-1}, u_{t-1} z_t):

$S_t = \emptyset, \quad \eta = 0$

For $i = 1 \dots n$ ***Generate new samples***

Sample index *j(i)* from the discrete distribution given by w_{t-1}

Sample x_t^i from $p(x_t \mid x_{t-1}, u_{t-1})$ using $x_{t-1}^{j(i)}$ and u_{t-1}

$w_t^i = p(z_t \mid x_t^i)$ ***Compute importance weight***

$\eta = \eta + w_t^i$ ***Update normalization factor***

$S_t = S_t \cup \{< x_t^i, w_t^i >\}$ ***Insert***

For $i = 1 \dots n$

$w_t^i = w_t^i / \eta$ ***Normalize weights***

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:

$$\begin{aligned} w^i_t &= \frac{\text{target distribution}}{\text{proposal distribution}} \\ &= \frac{\eta\, p(z_t \mid x_t)\, p(x_t \mid x_{t-1}, u_{t-1})\, Bel(x_{t-1})}{p(x_t \mid x_{t-1}, u_{t-1})\, Bel(x_{t-1})} \\ &\propto p(z_t \mid x_t) \end{aligned}$$

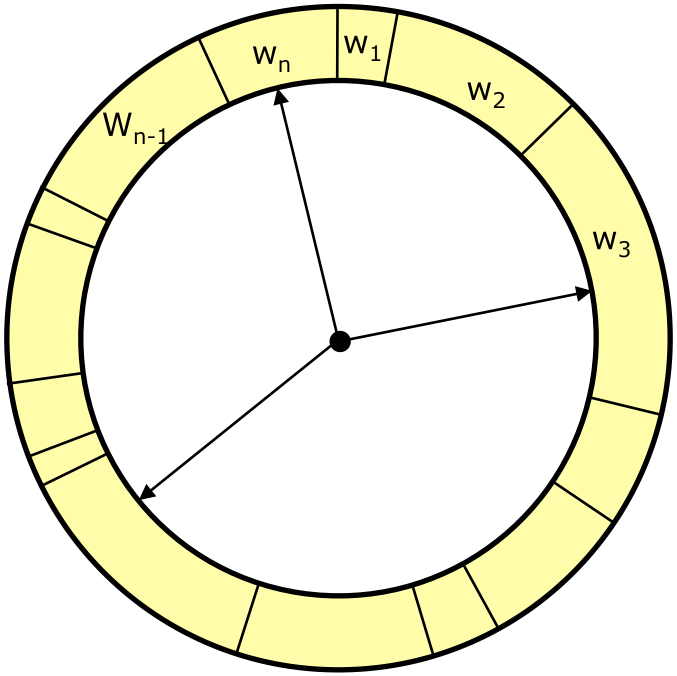
Resampling

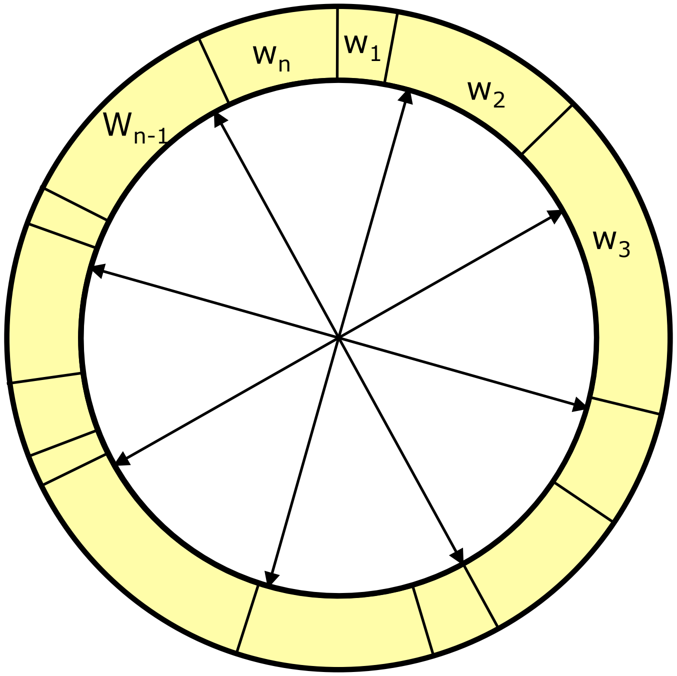
- **Given**: Set S of weighted samples.

- **Wanted** : Random sample, where the probability of drawing x_i is given by w_i.

- Typically done n times with replacement to generate new sample set S'.

Resampling





- Roulette wheel
- Binary search, n log n

- Stochastic universal sampling
- Systematic resampling
- Linear time complexity
- Easy to implement, low variance

Resampling Algorithm

1. Algorithm **systematic_resampling**(S,n):

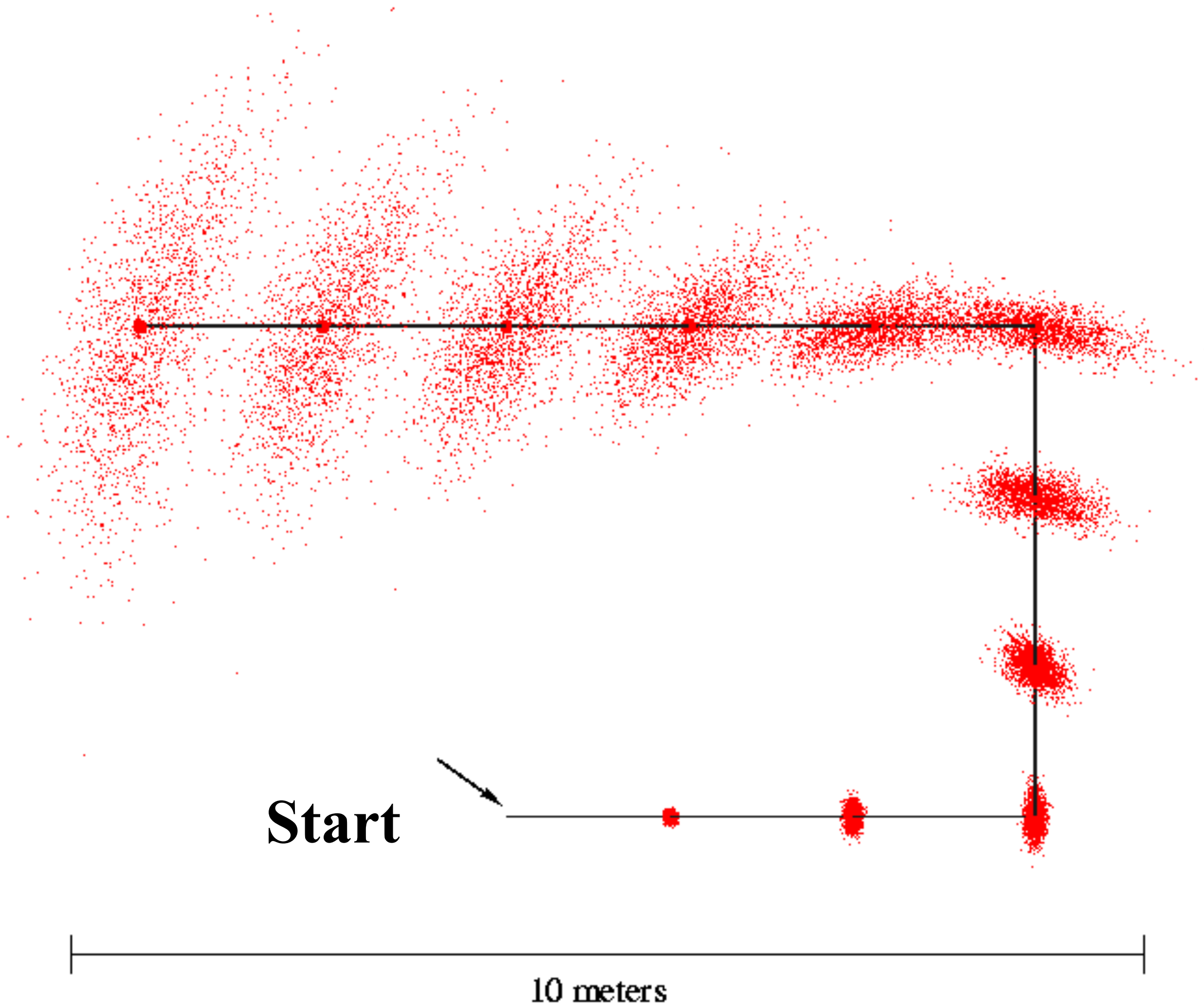
2. $S' = \emptyset, c_1 = w^1$
3. **For** $i = 2 \ldots n$ ***Generate cdf***
4. $c_i = c_{i-1} + w^i$
5. $u_1 \sim U]0, n^{-1}], i = 1$ ***Initialize threshold***

6. **For** $j = 1 \ldots n$ ***Draw samples …***
7. **While** ($u_j > c_i$) ***Skip until next threshold reached***
8. $i = i + 1$
9. $S' = S' \cup \left\{ < x^i, n^{-1} > \right\}$ ***Insert***
10. $u_{j+1} = u_j + n^{-1}$ ***Increment threshold***

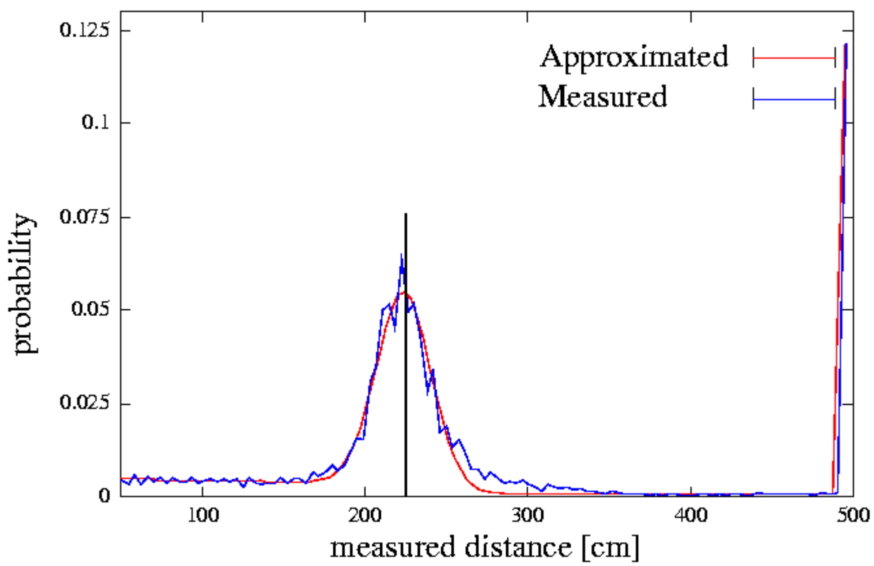
11. **Return** S'

Also called **stochastic universal sampling**

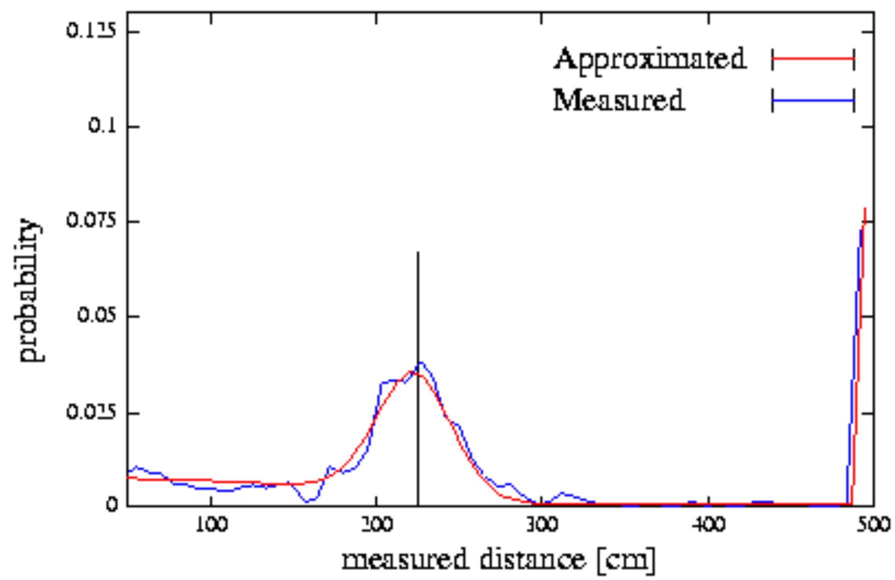
Motion Model Reminder



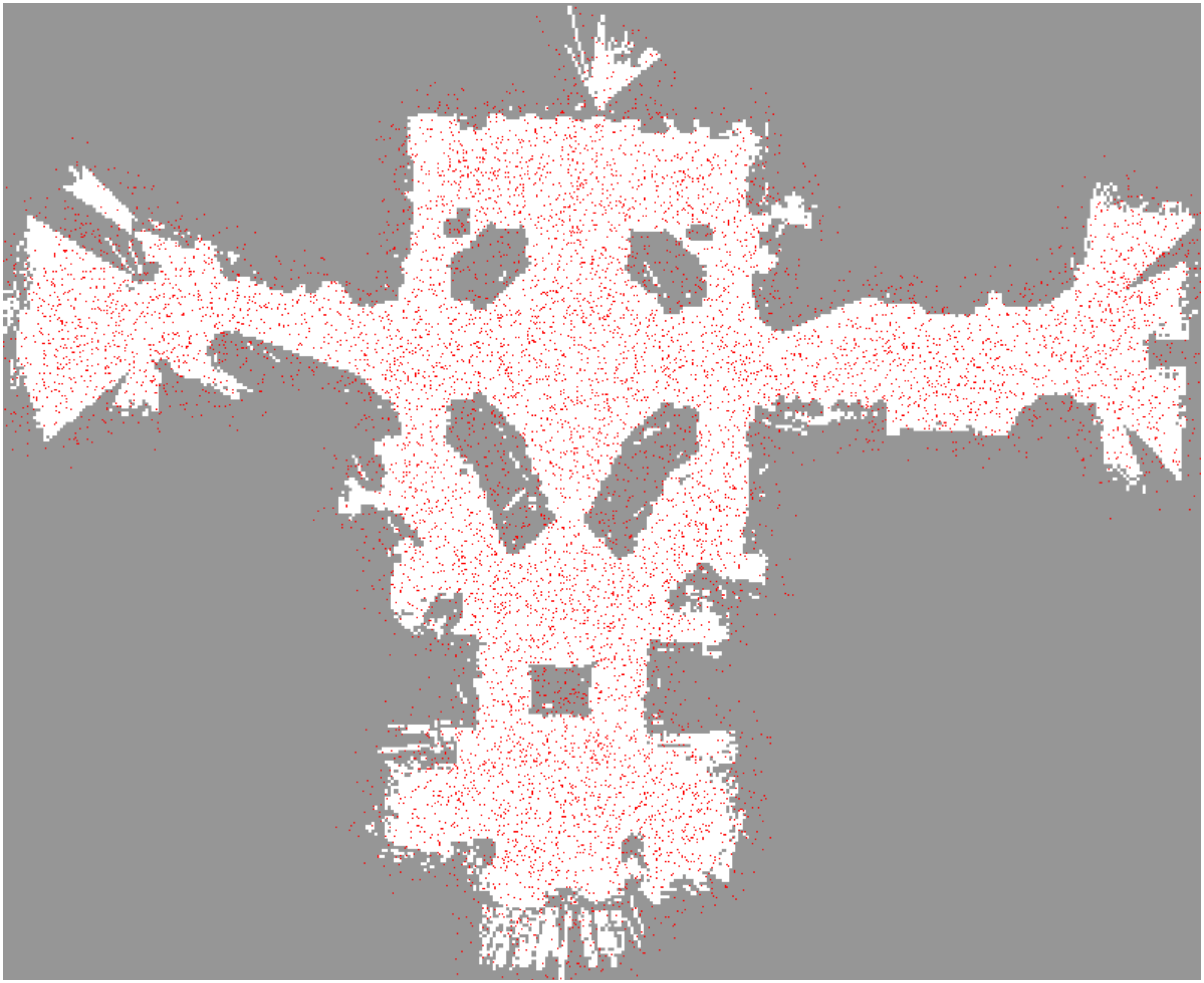
Proximity Sensor Model Reminder

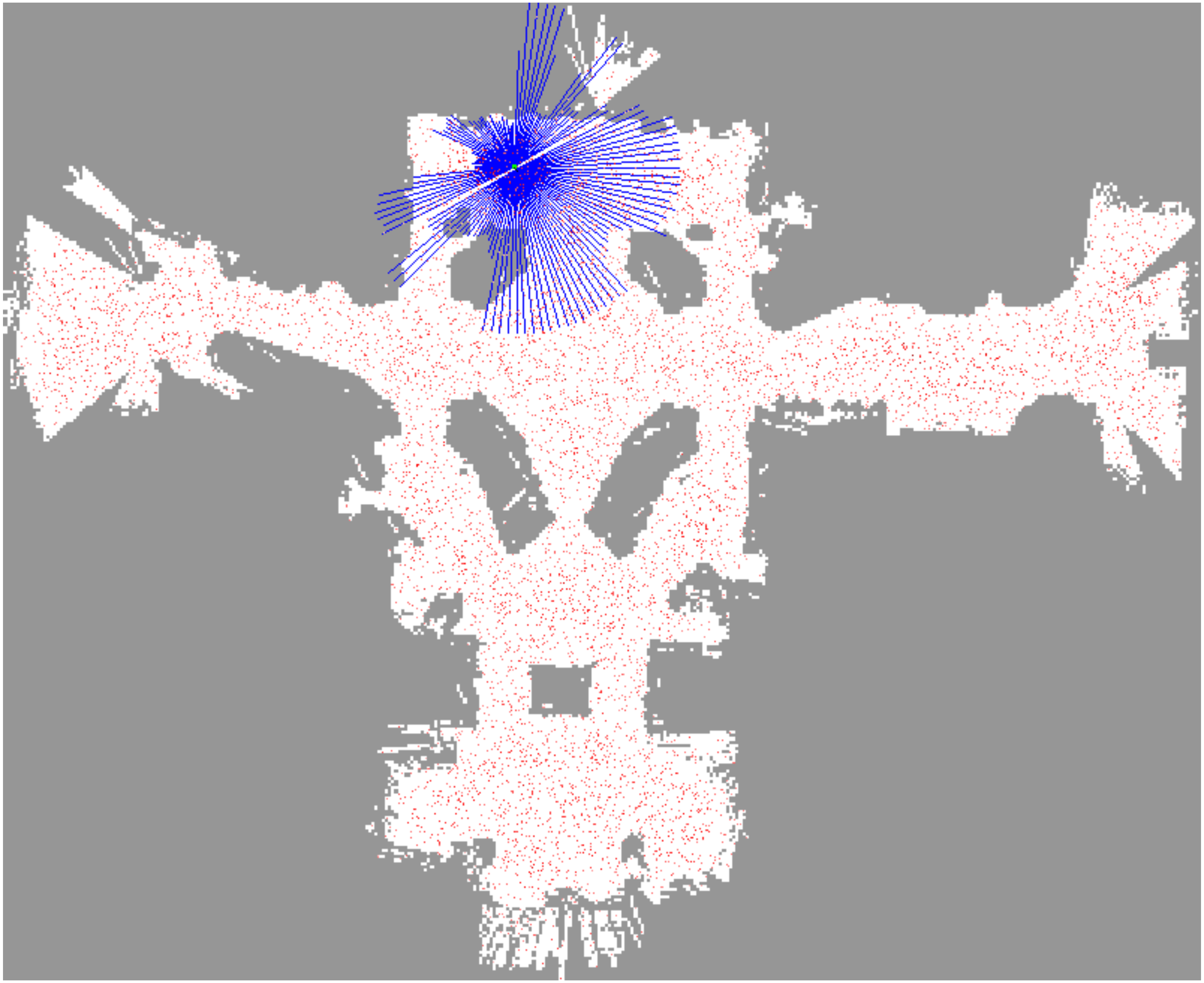


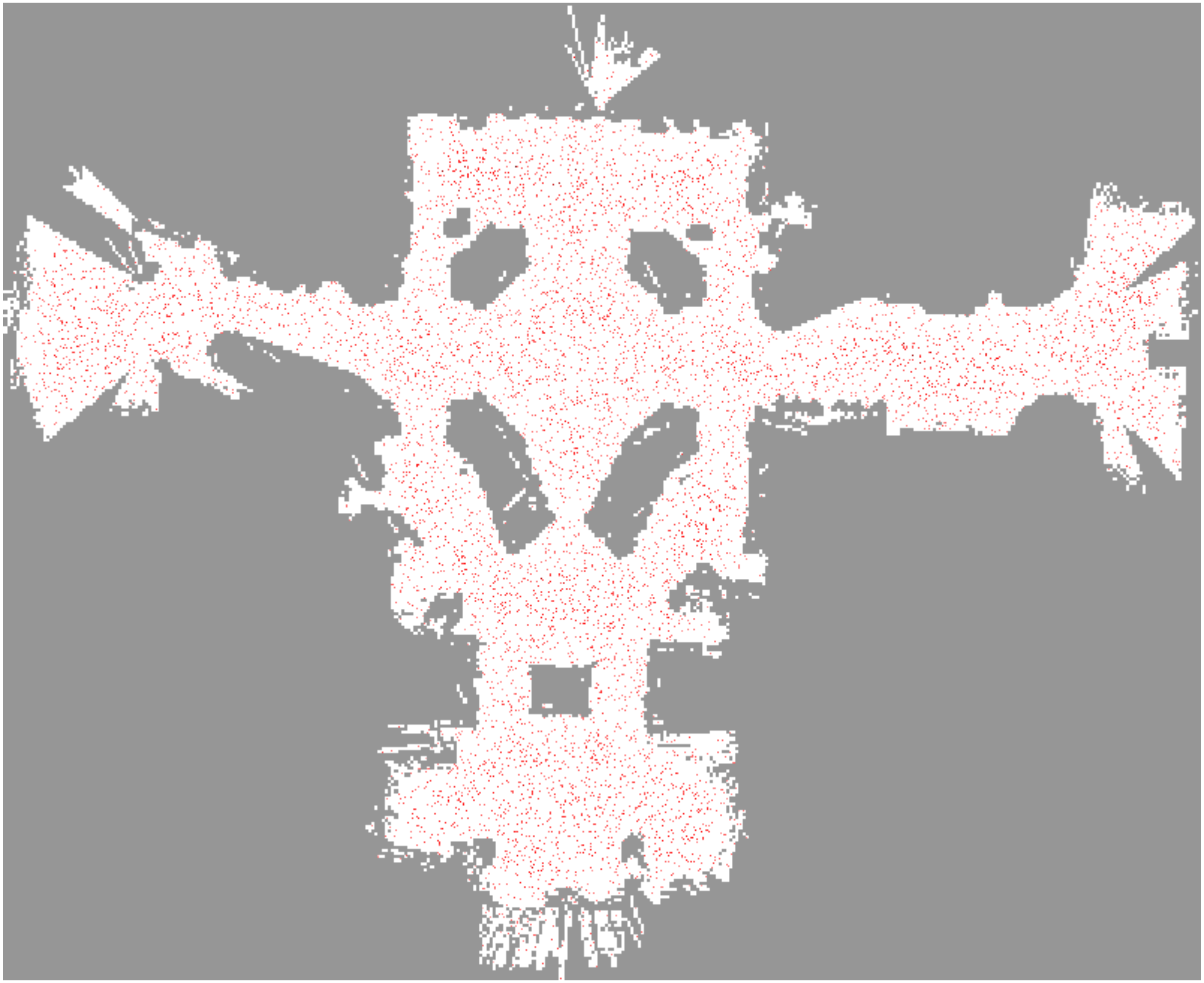
Laser sensor

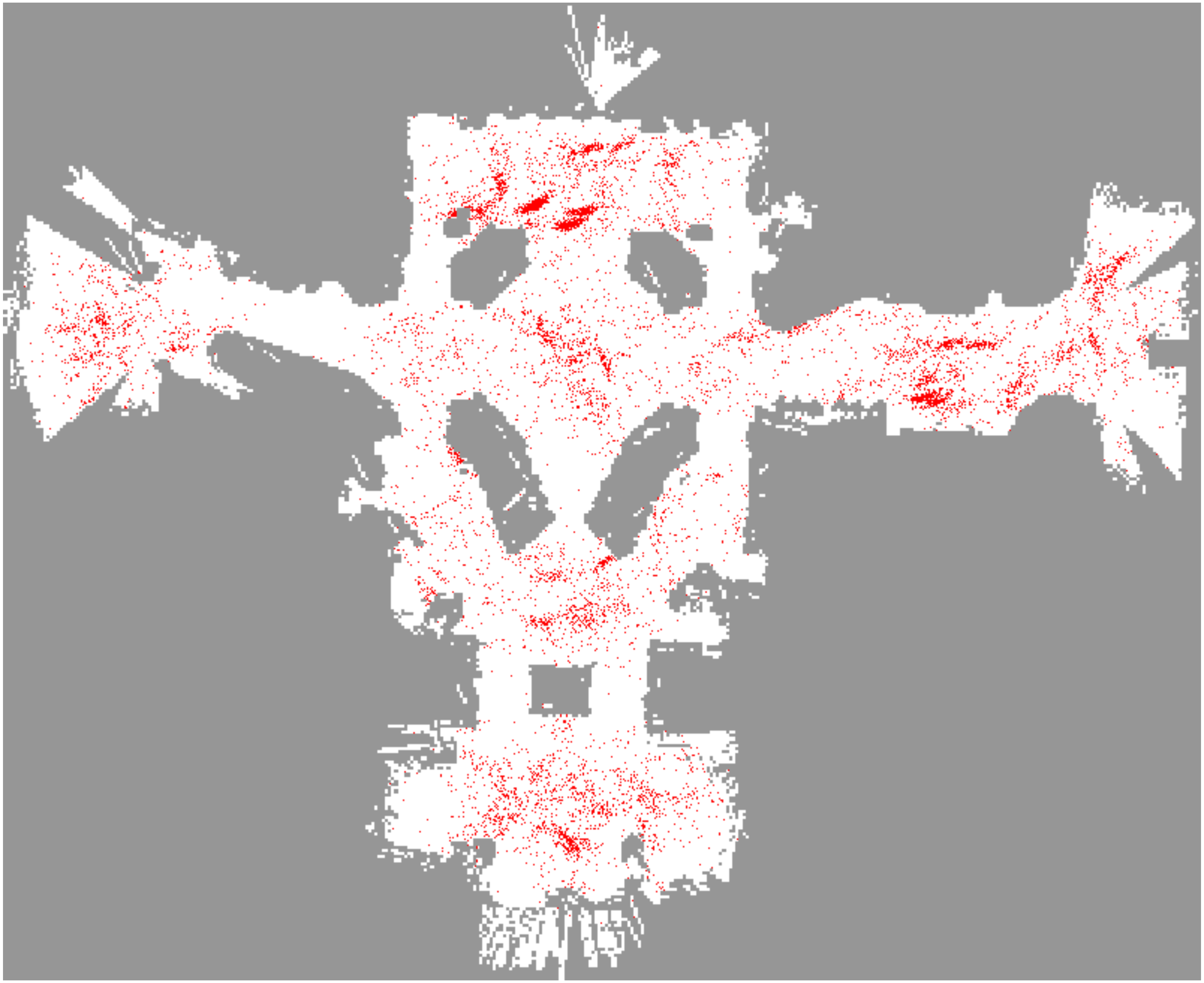


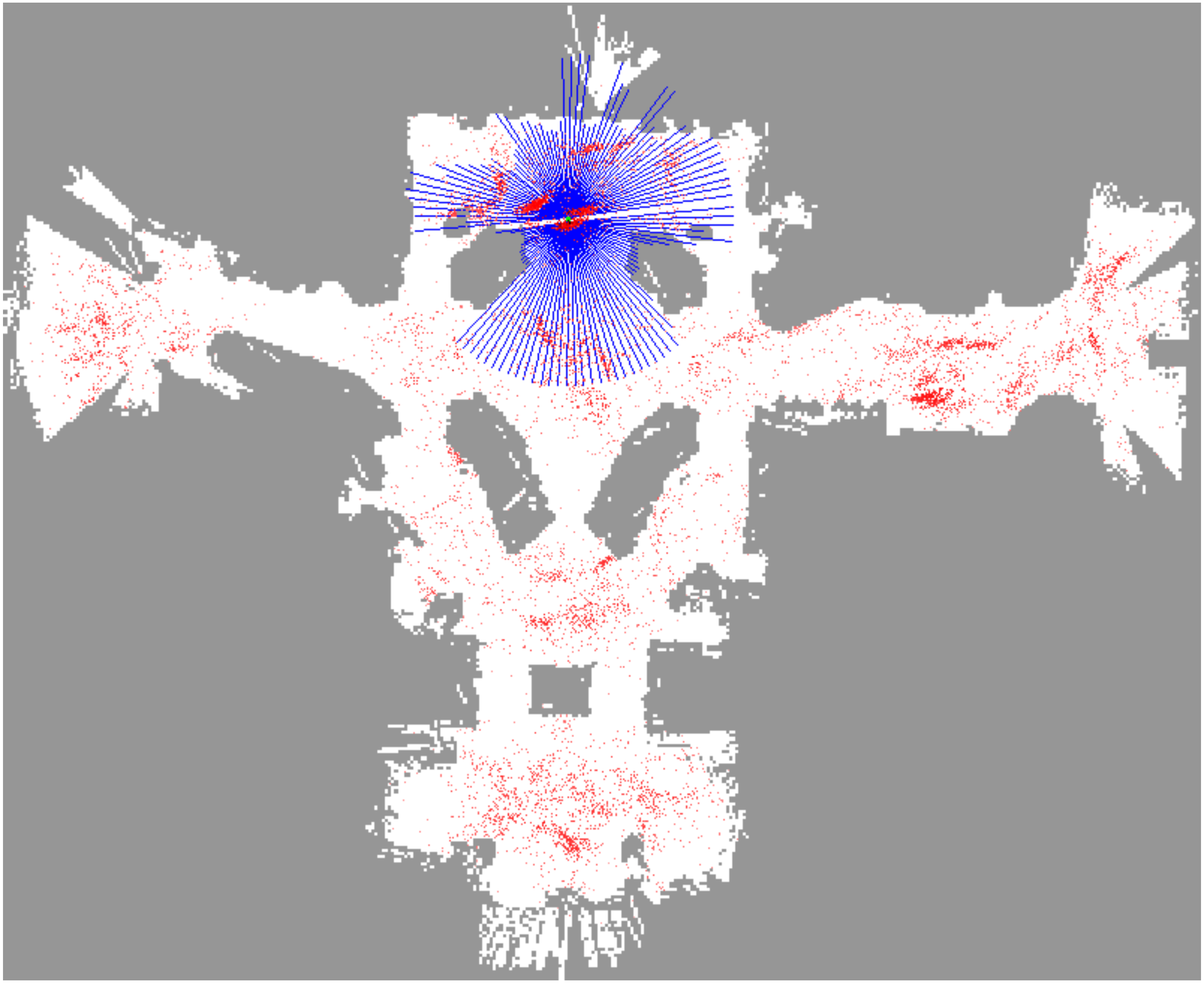
Sonar sensor

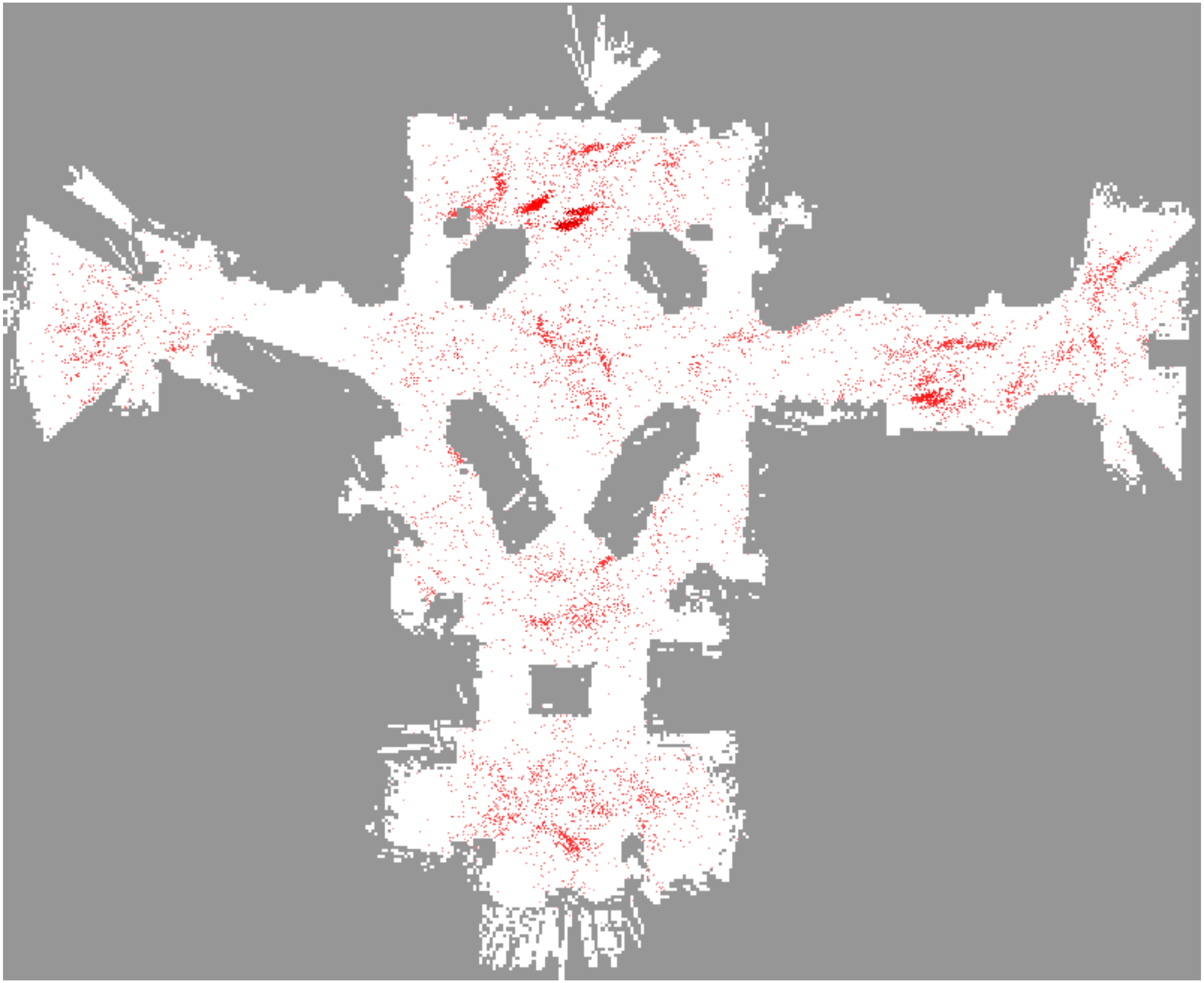


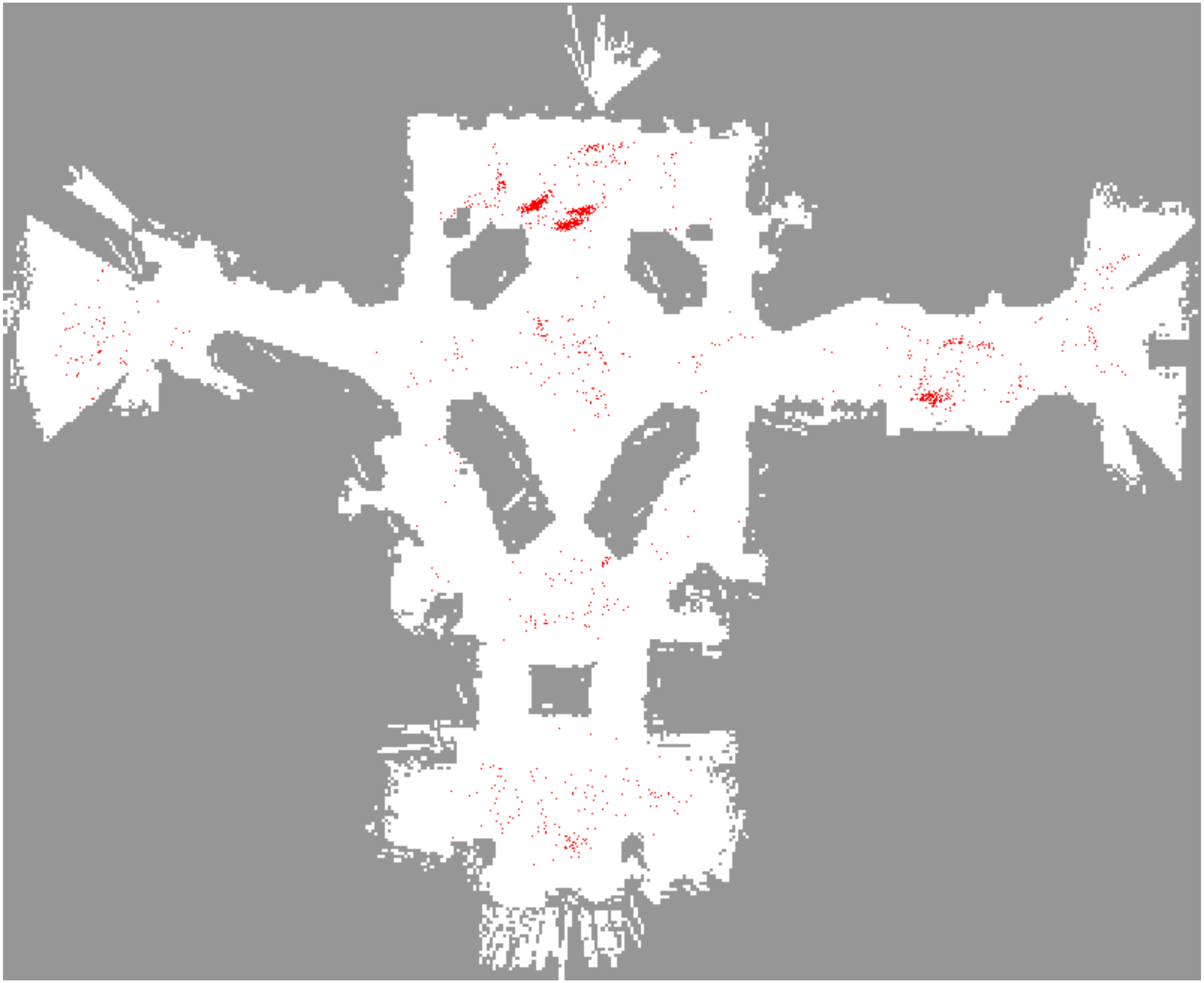


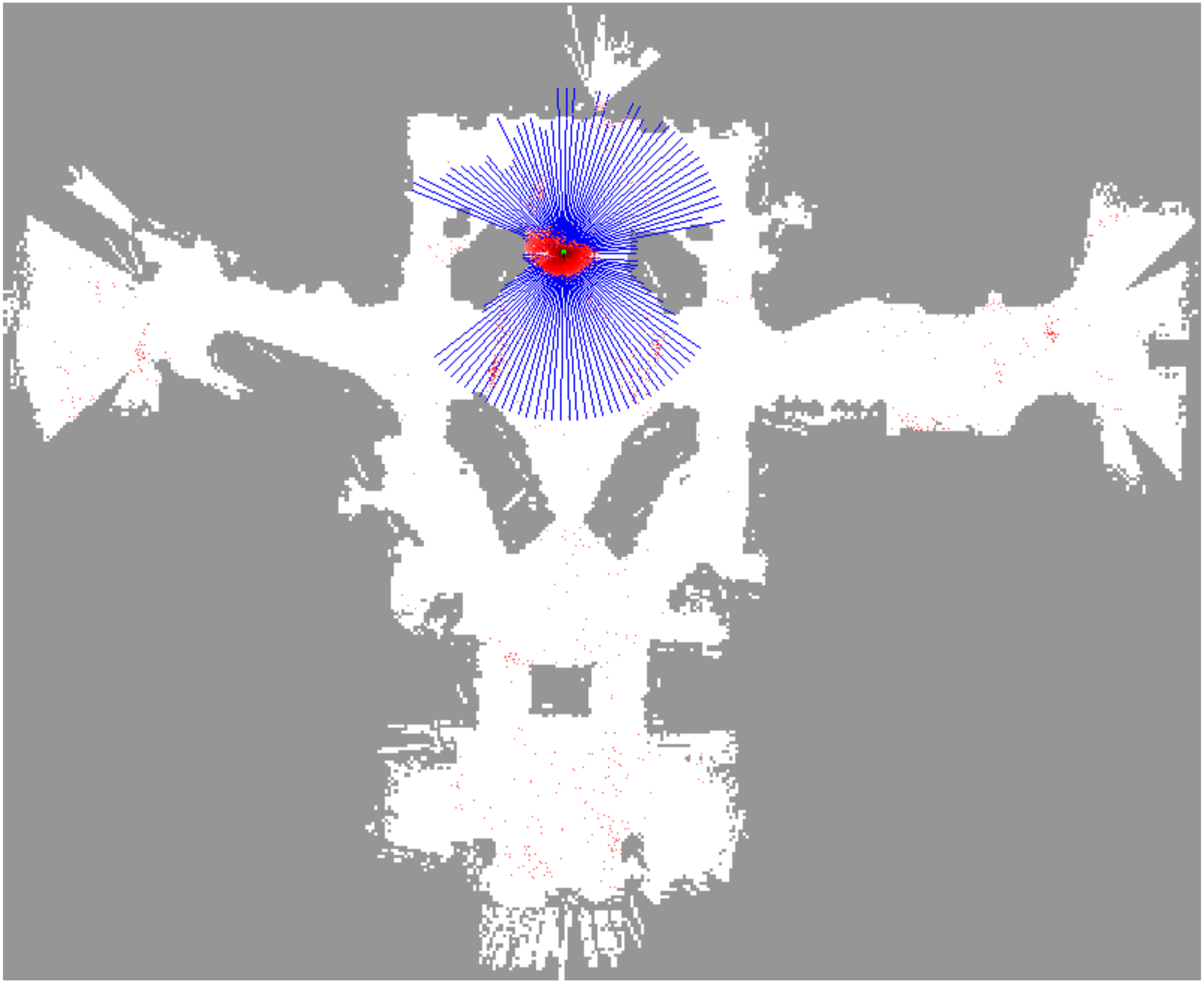


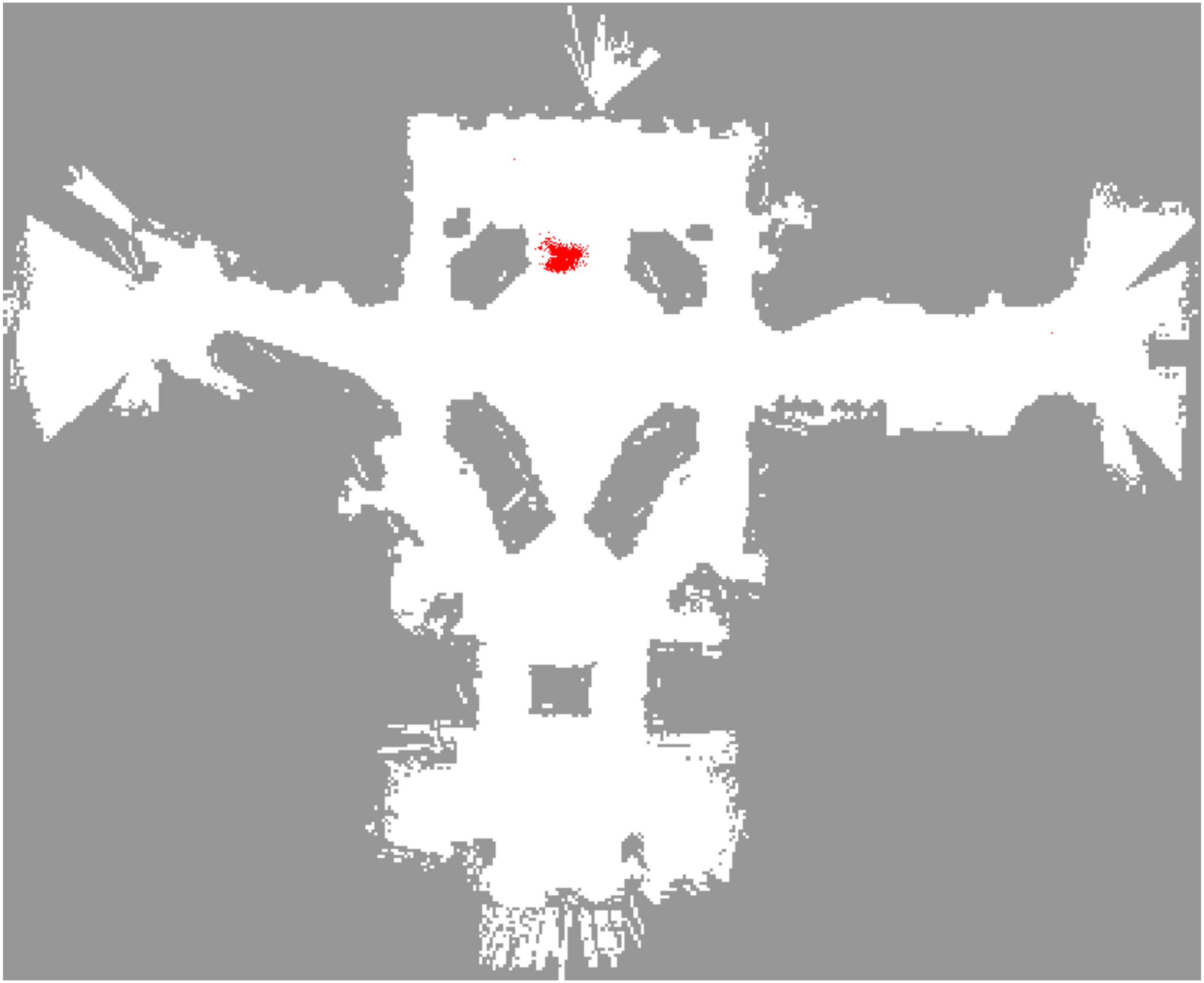


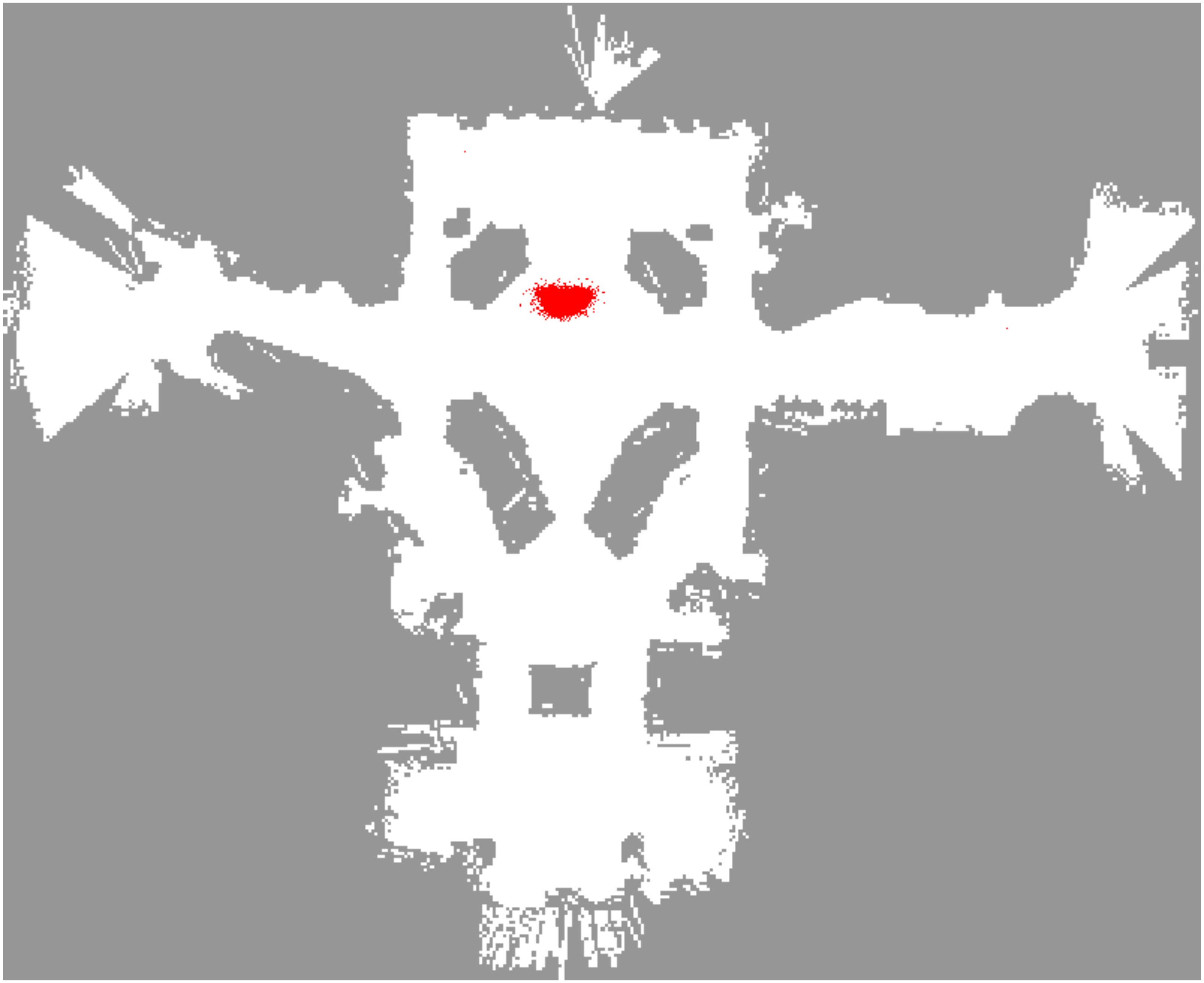


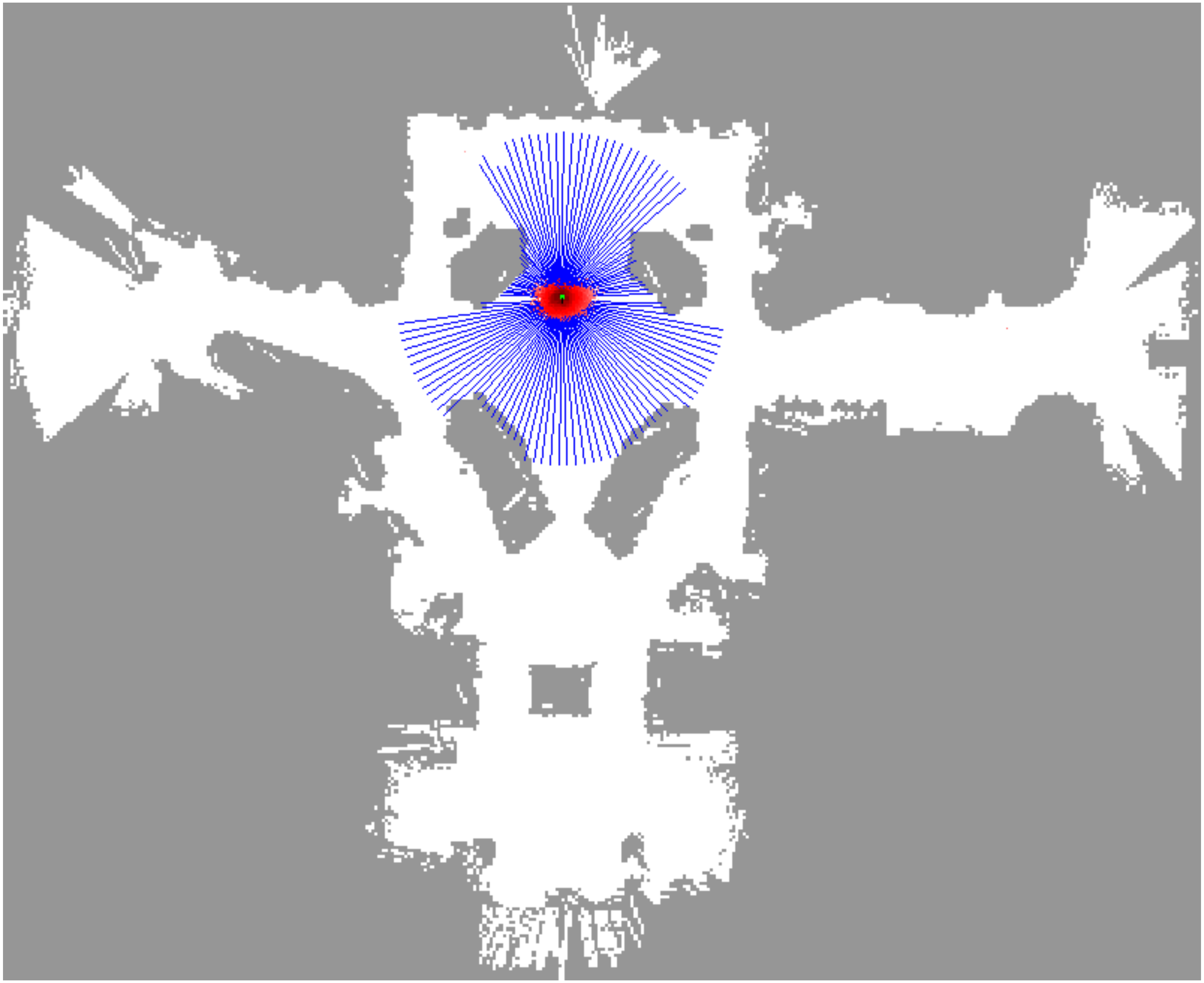


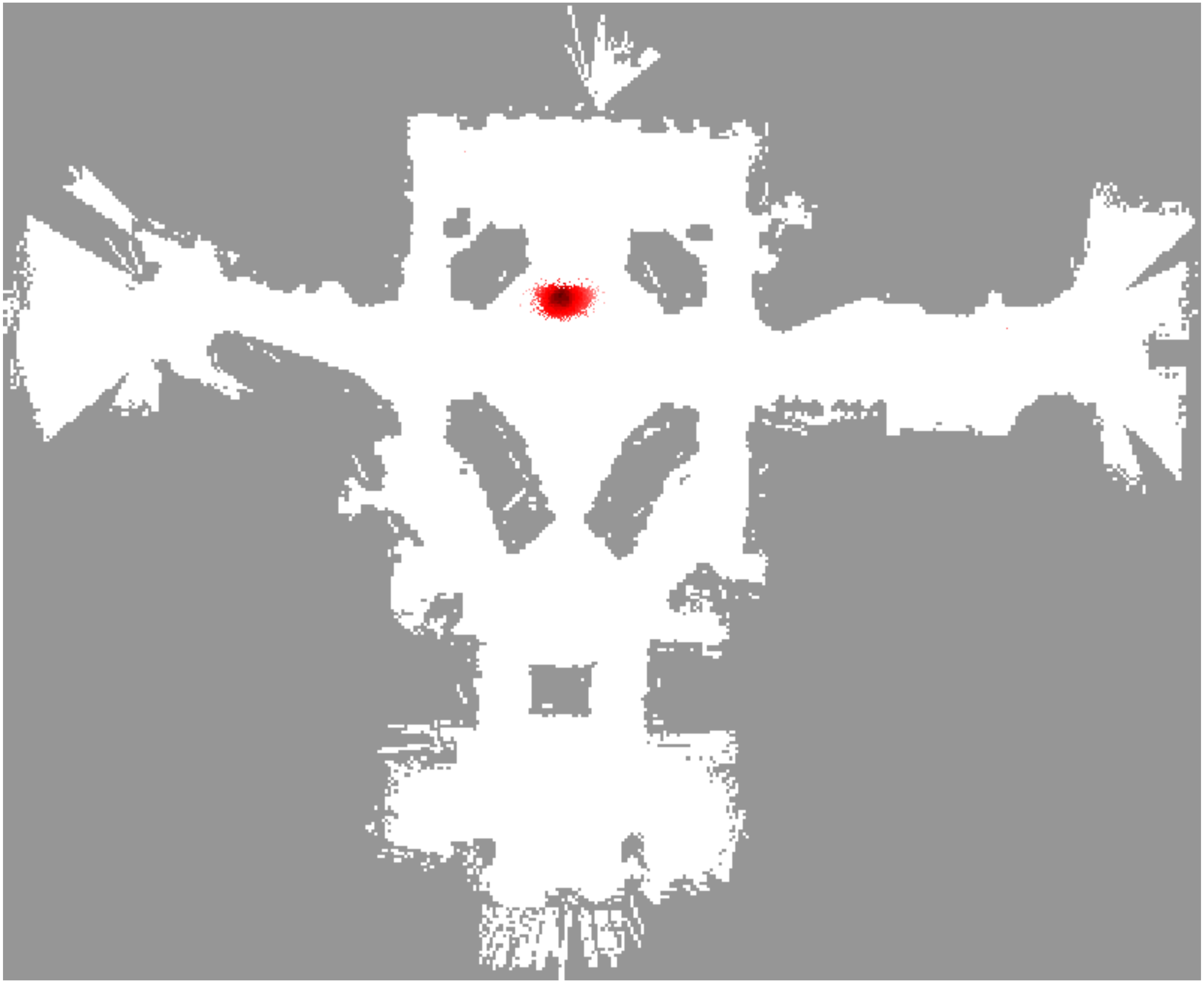


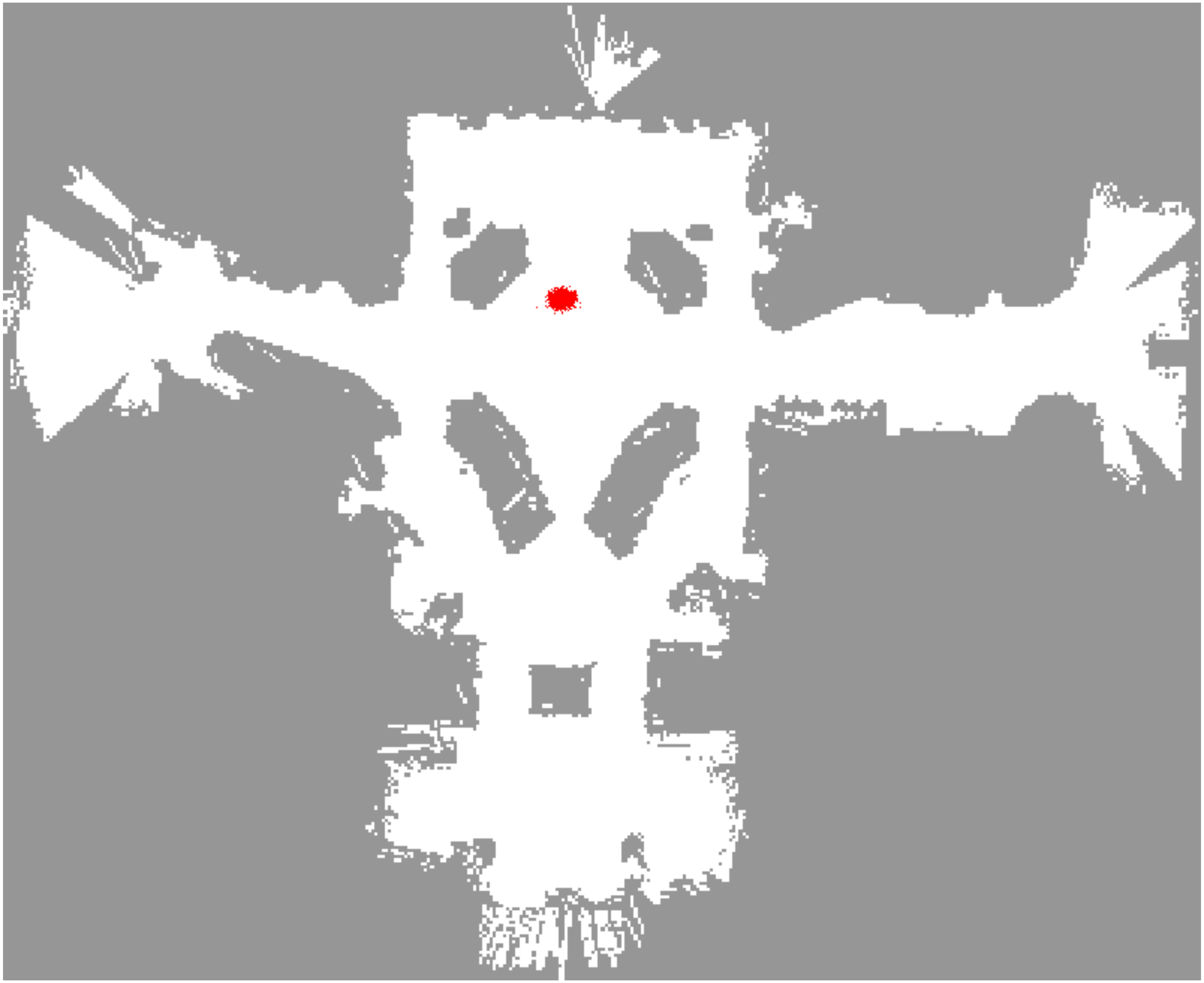


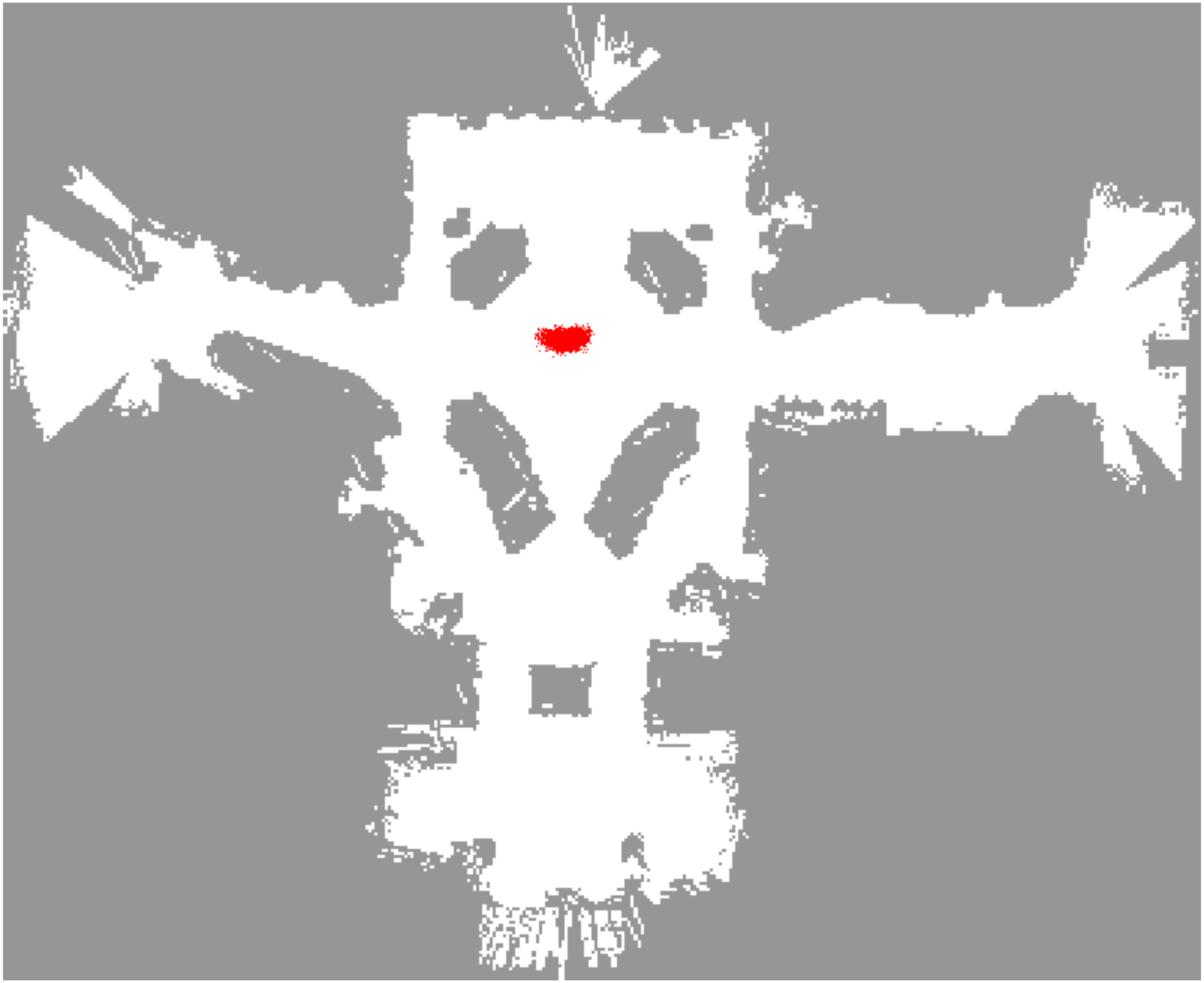


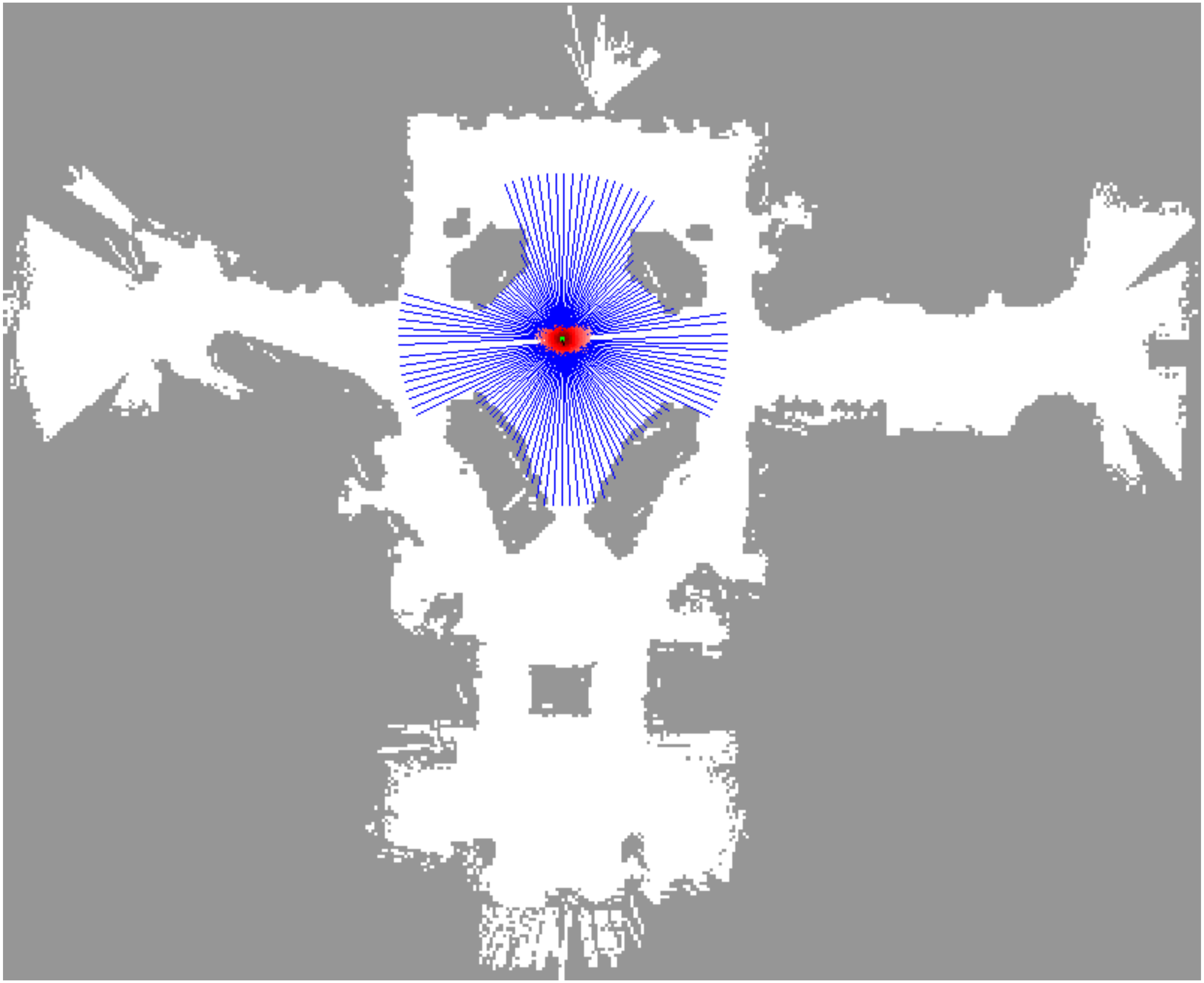


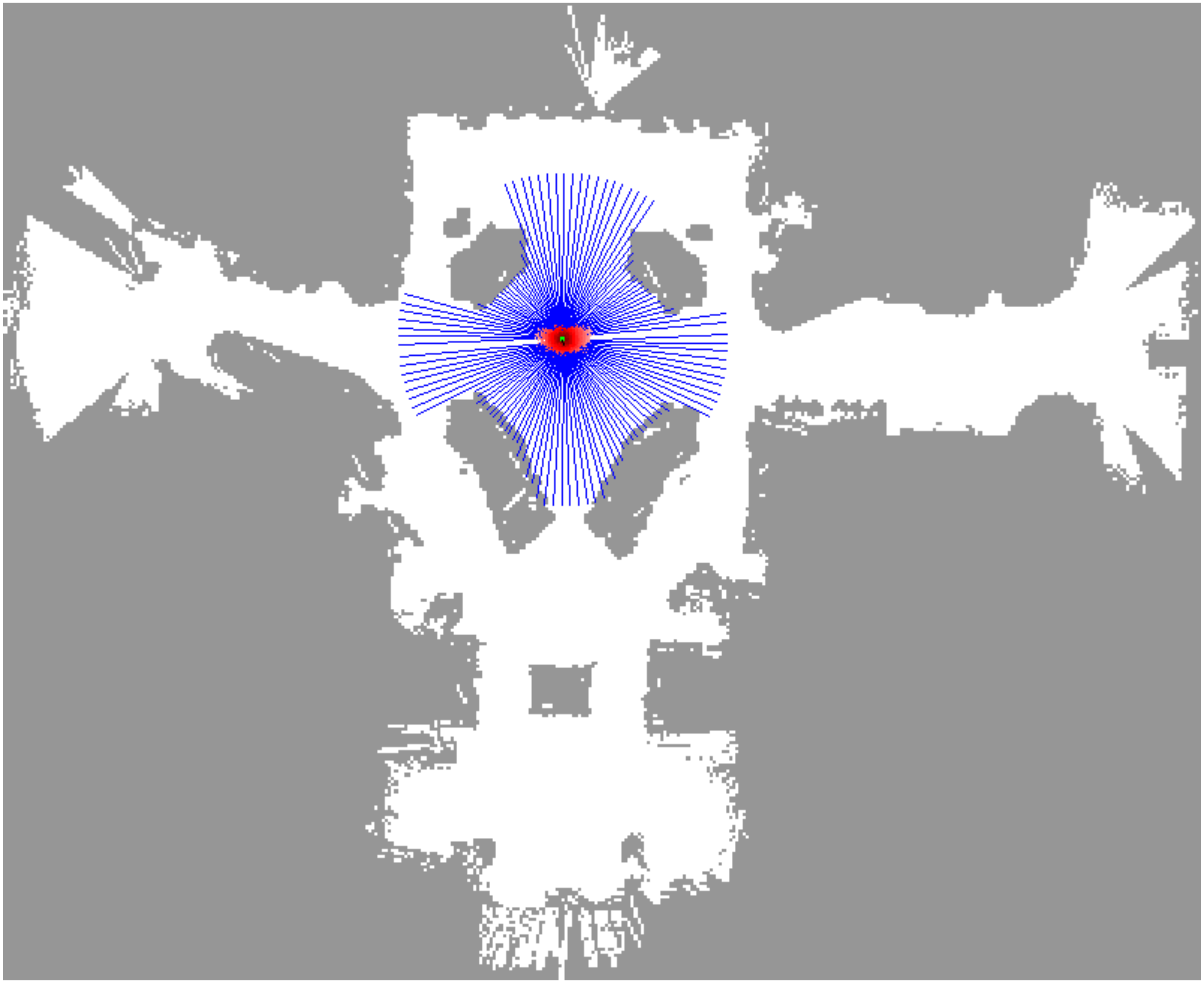




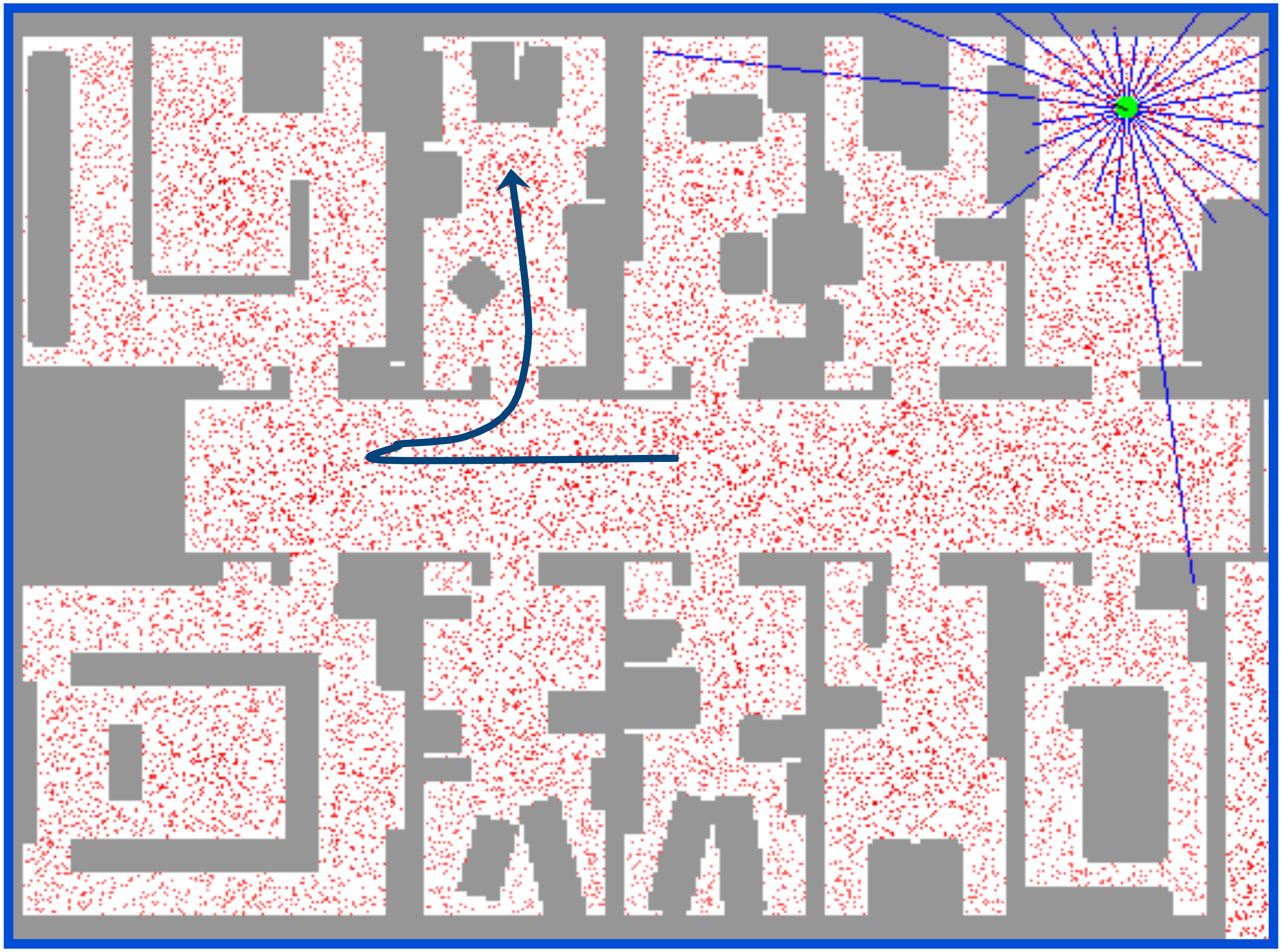




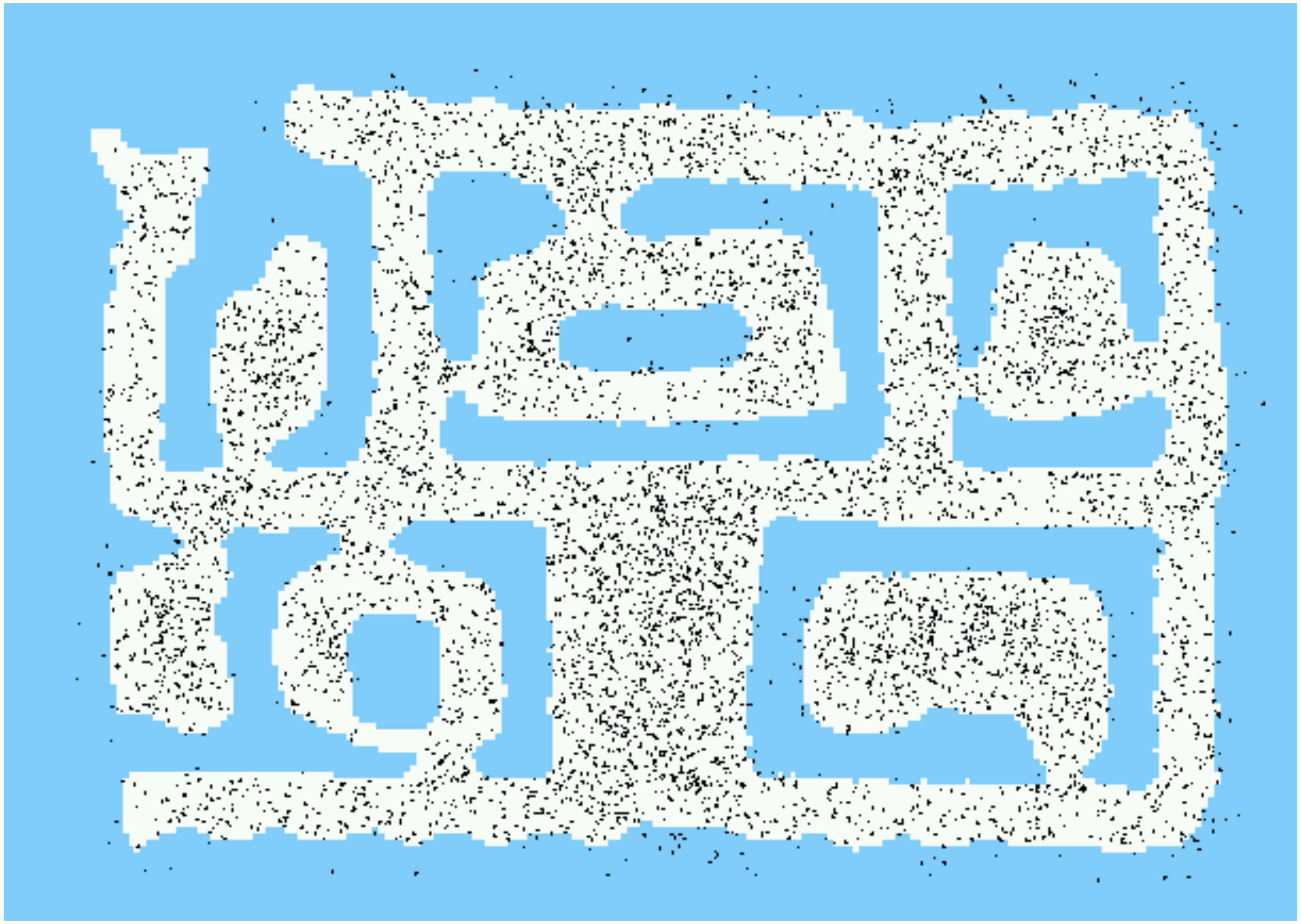




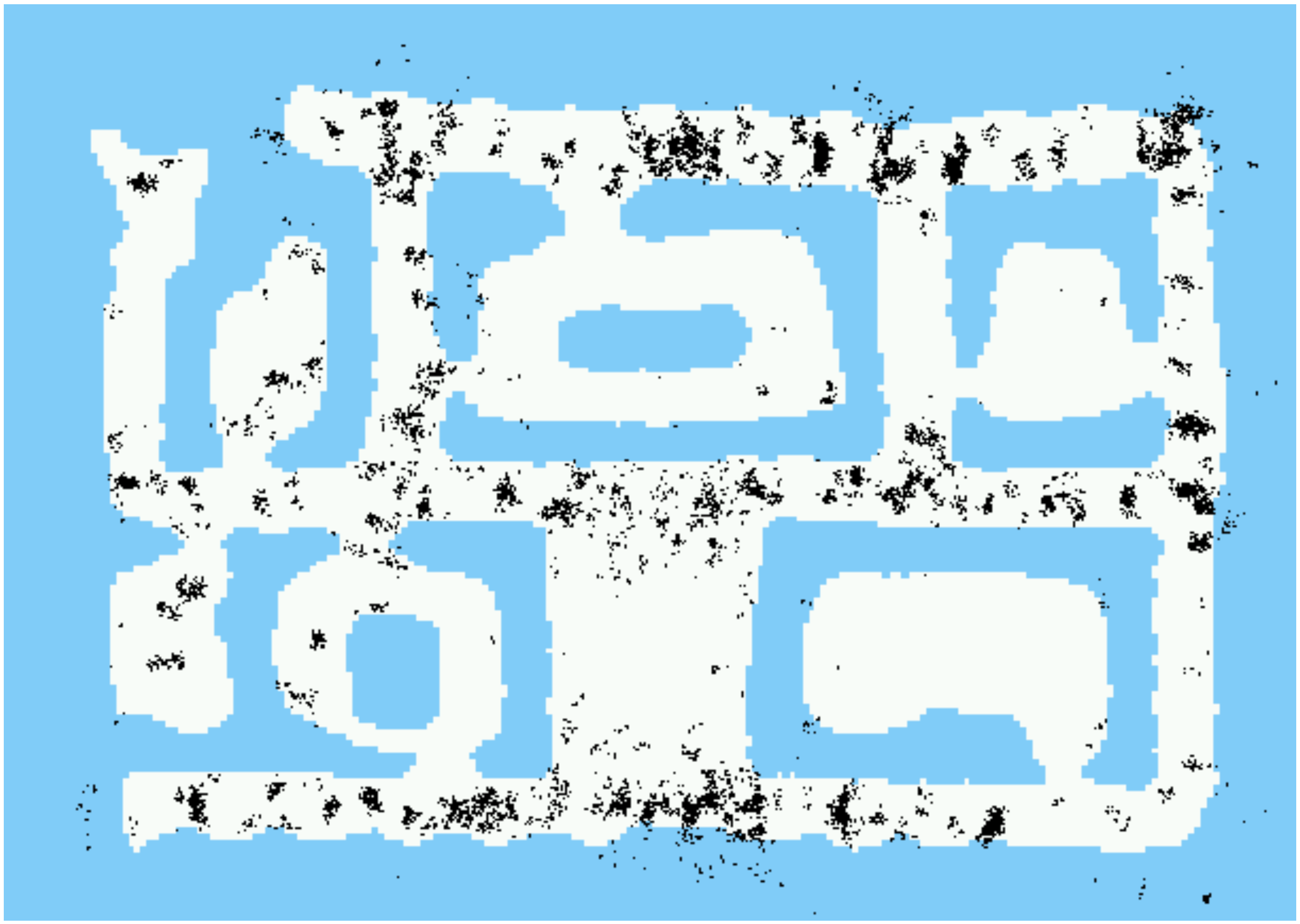
Sample-based Localization (sonar)



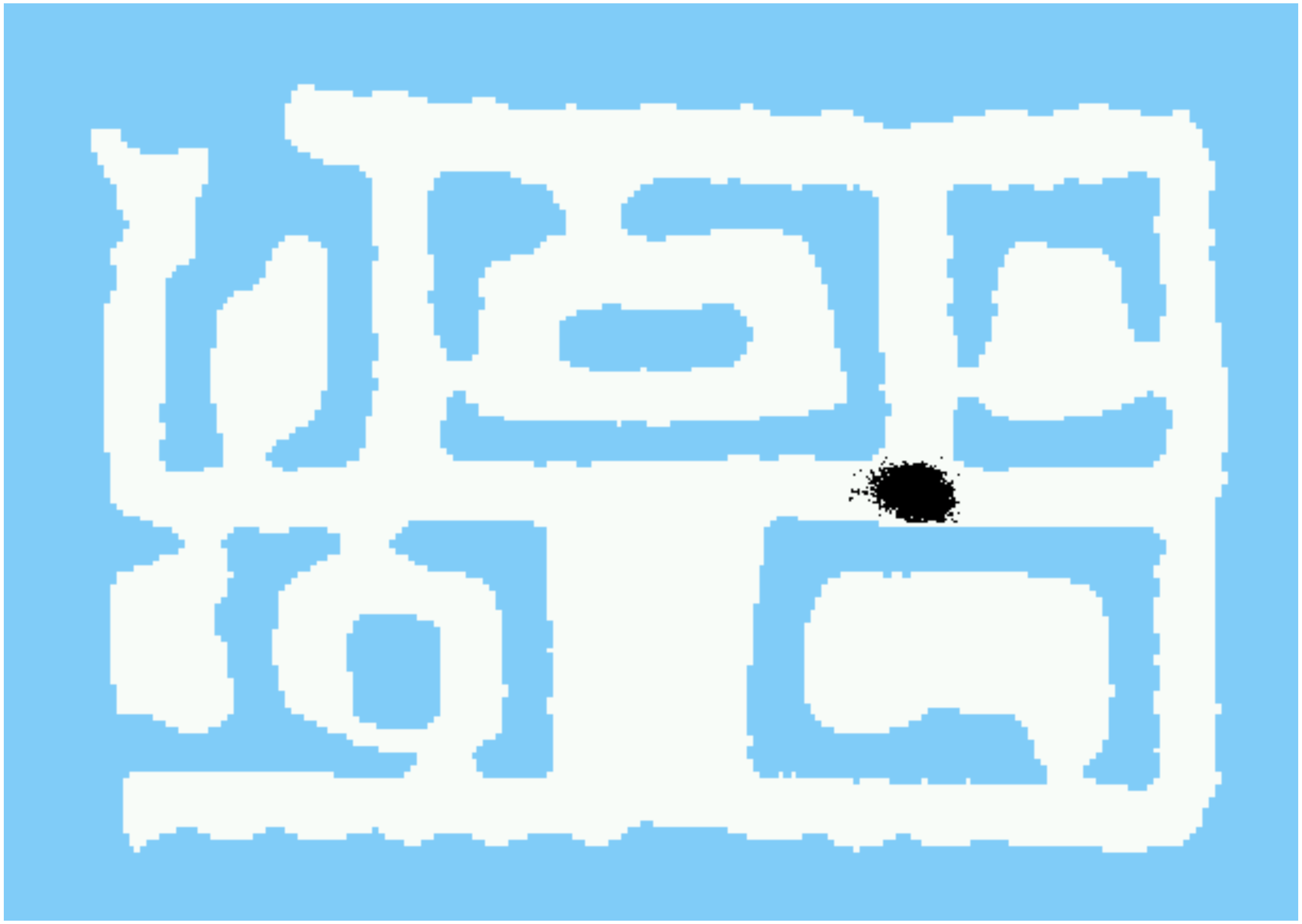
Initial Distribution



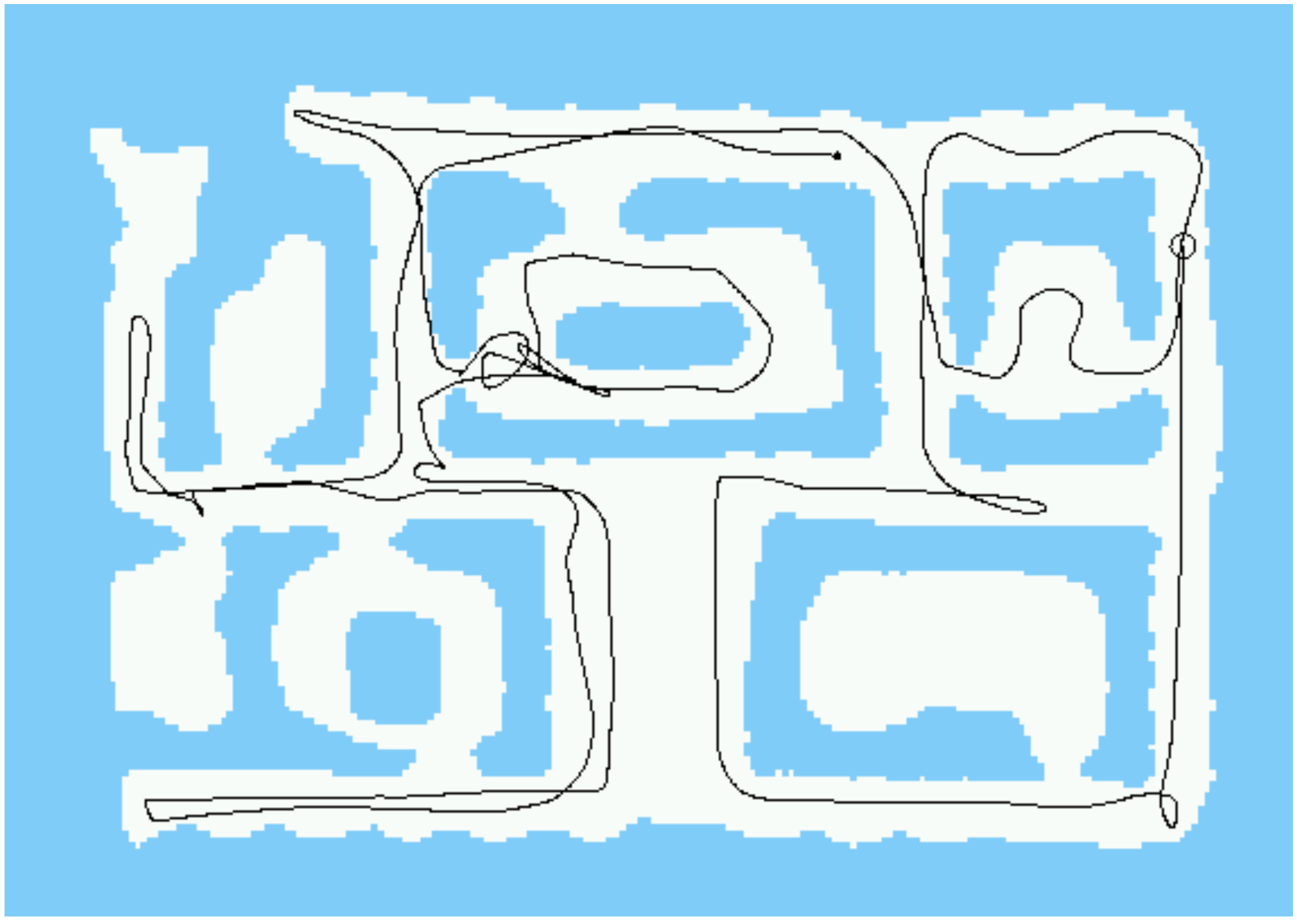
After Incorporating Ten Ultrasound Scans



After Incorporating 65 Ultrasound Scans



Estimated Path

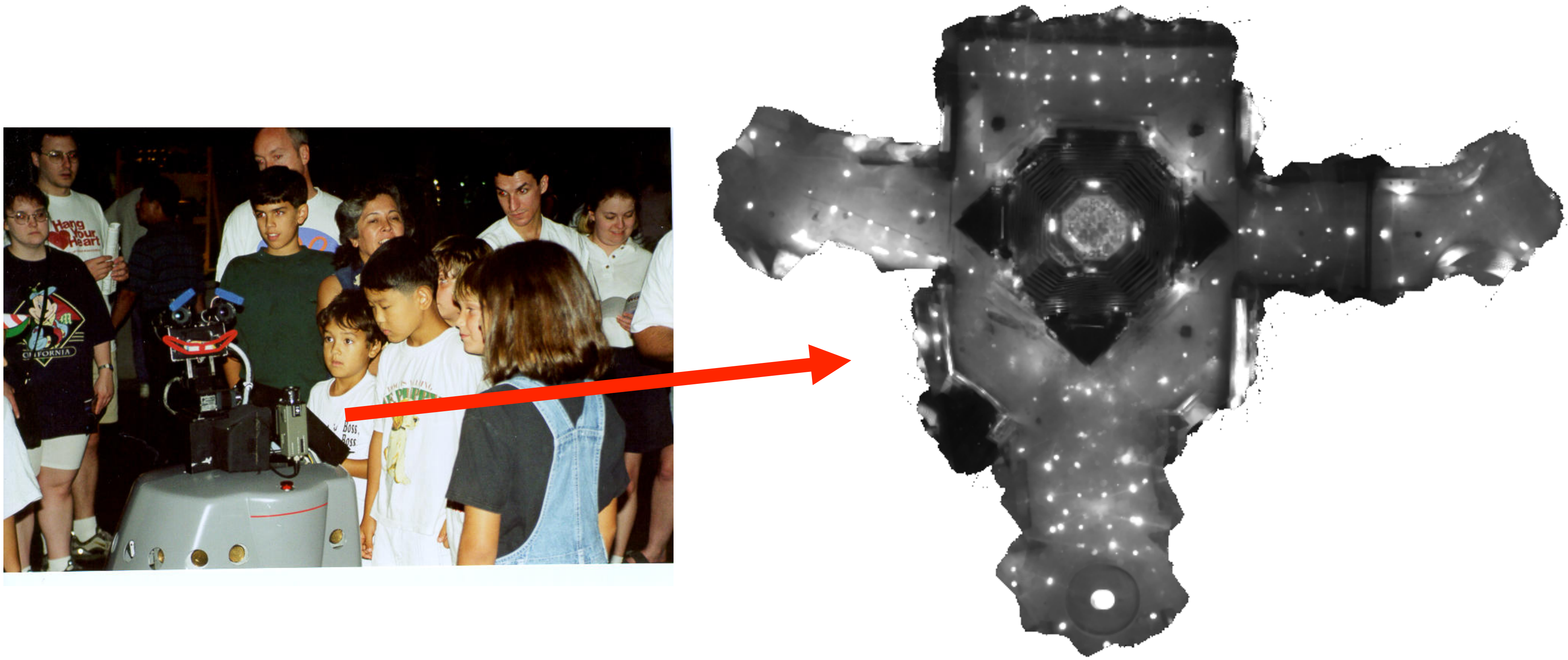


http://www.youtube.com/watch?v=MELYZ5r5V1c

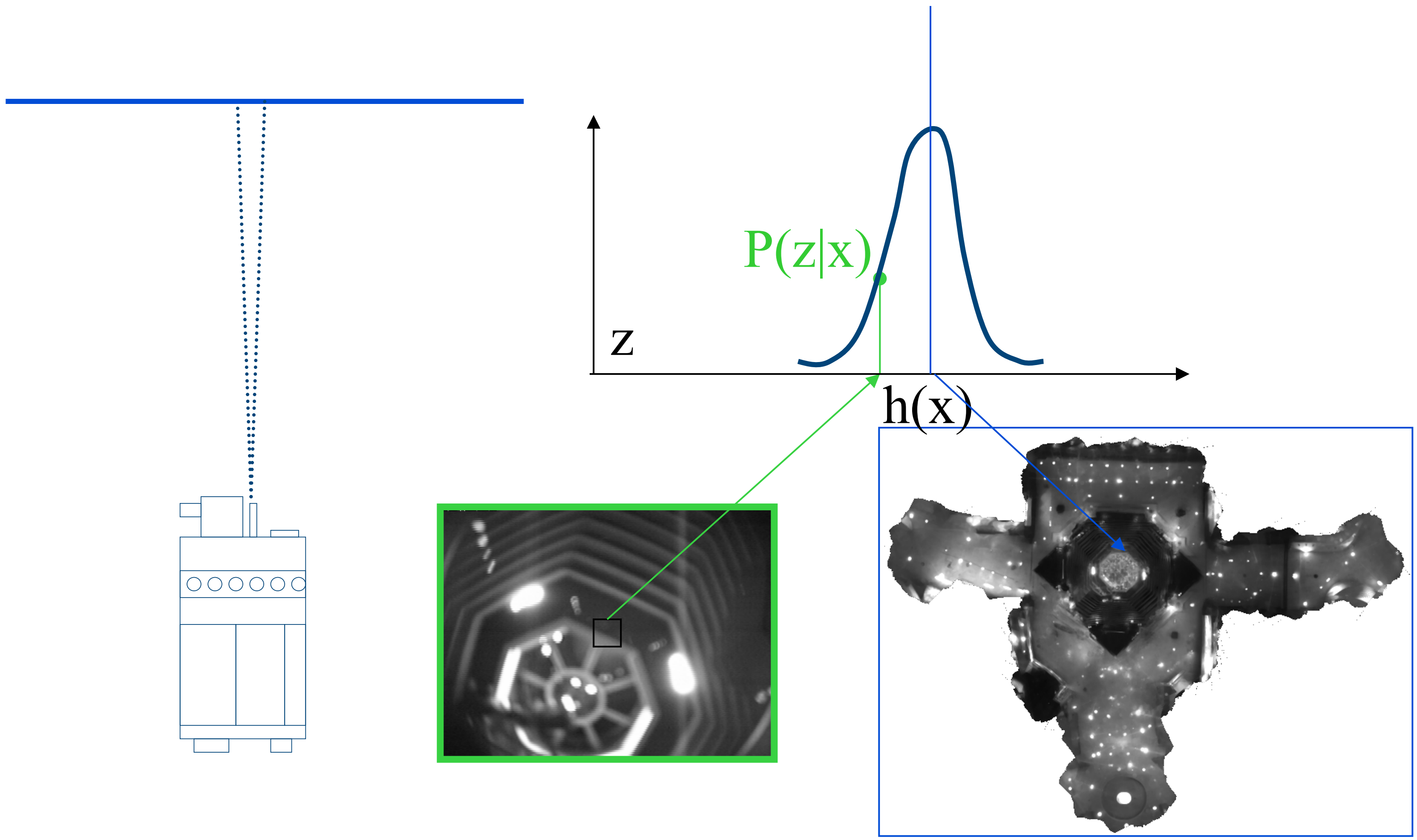
Summary

- Particle filters are an implementation of recursive Bayesian filtering
- They represent the posterior by a set of weighted samples.
- In the context of localization, the particles are propagated according to the motion model.
- They are then weighted according to the likelihood of the observations.
- In a re-sampling step, new particles are drawn with a probability proportional to the likelihood of the observation.

Using Ceiling Maps for Localization

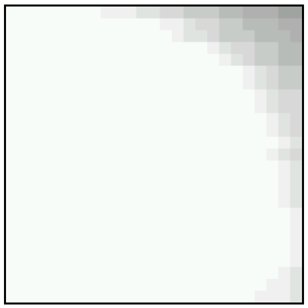


Vision-based Localization

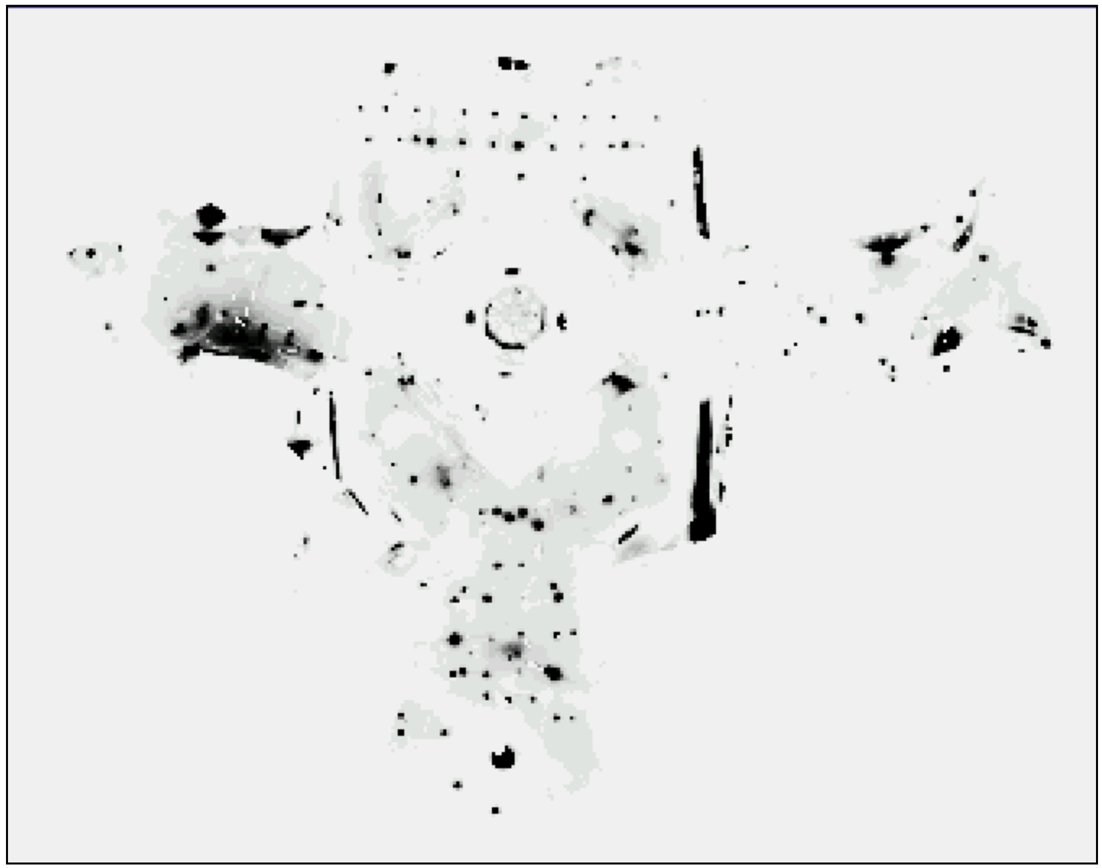


Under a Light

Measurement z:

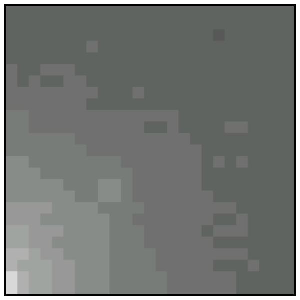


***P(z|x)*:**

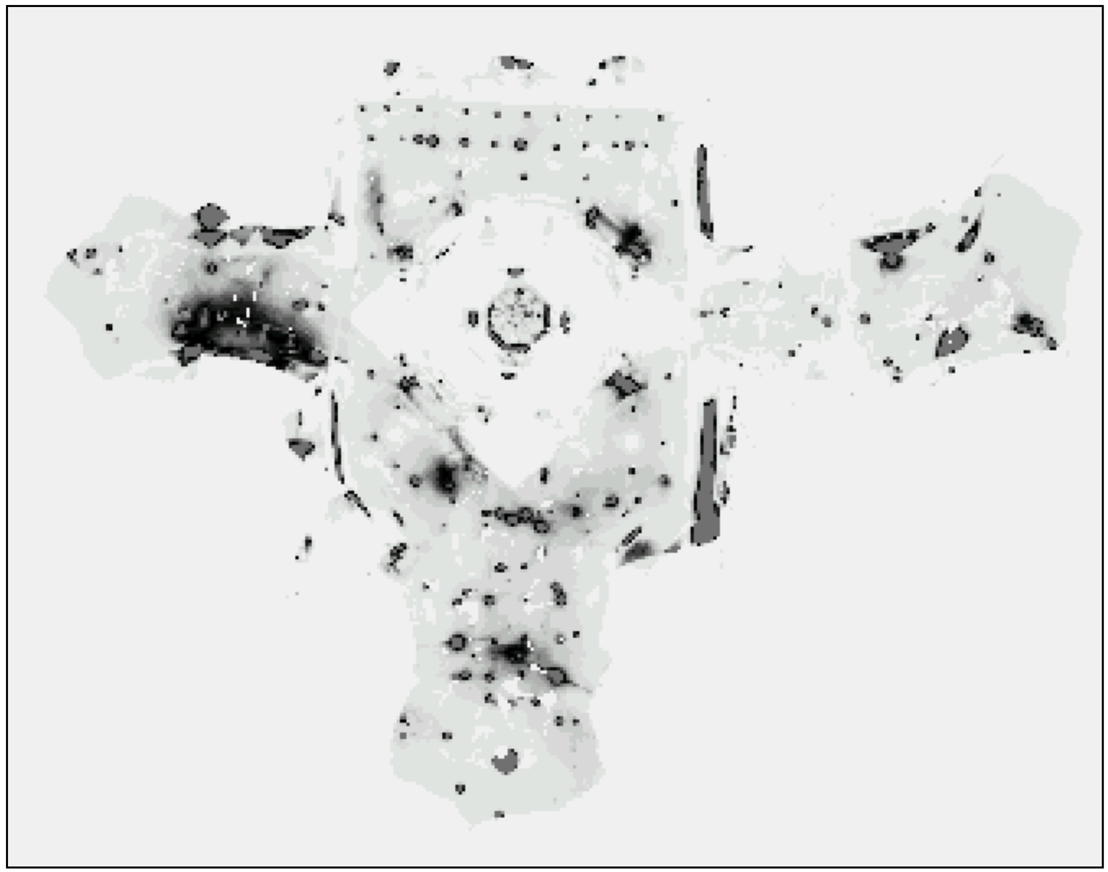


Next to a Light

Measurement z:



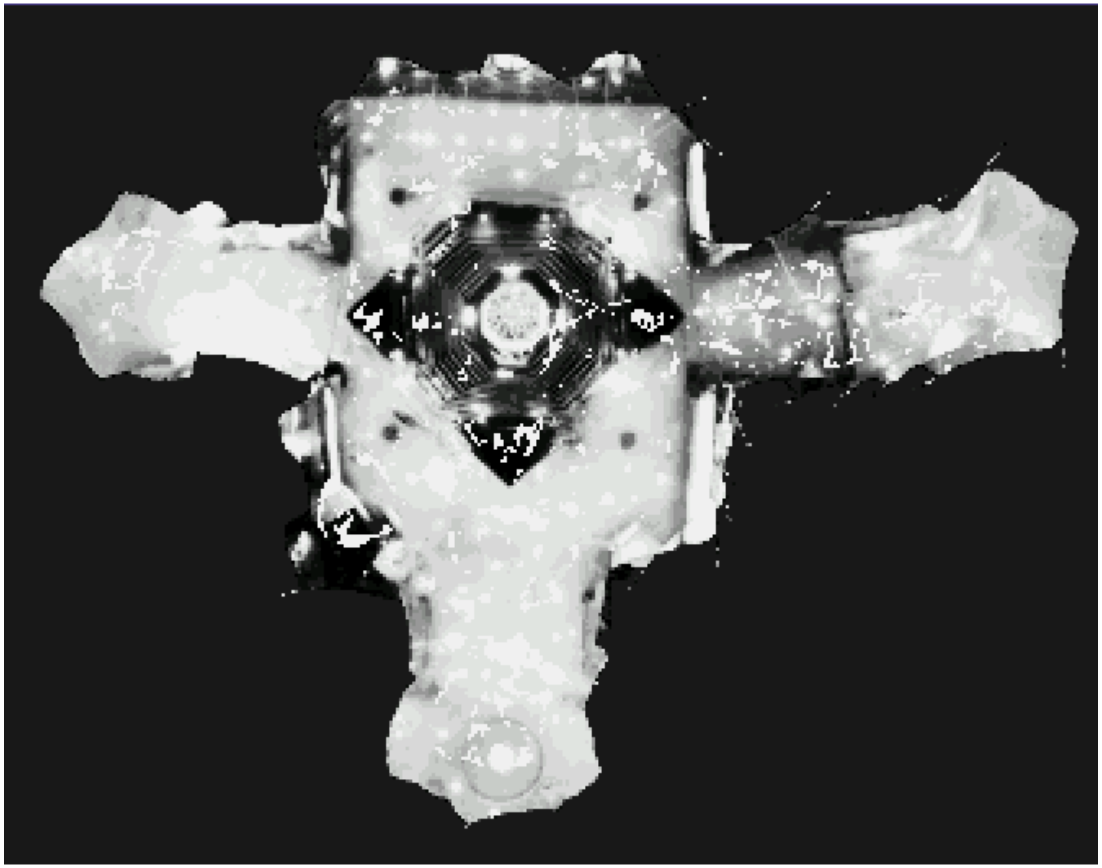
***P(z|x)*:**



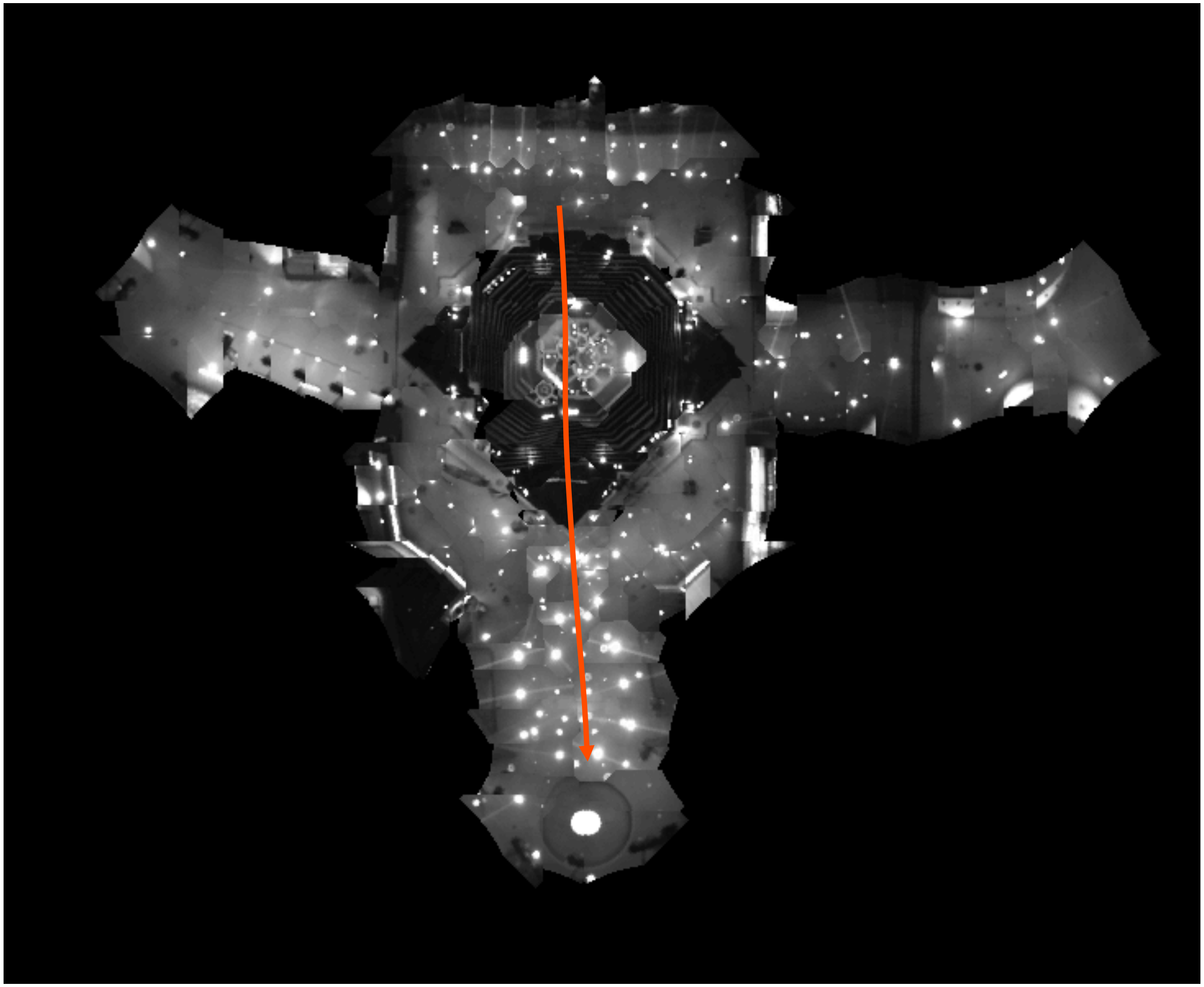
Elsewhere

Measurement z:

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

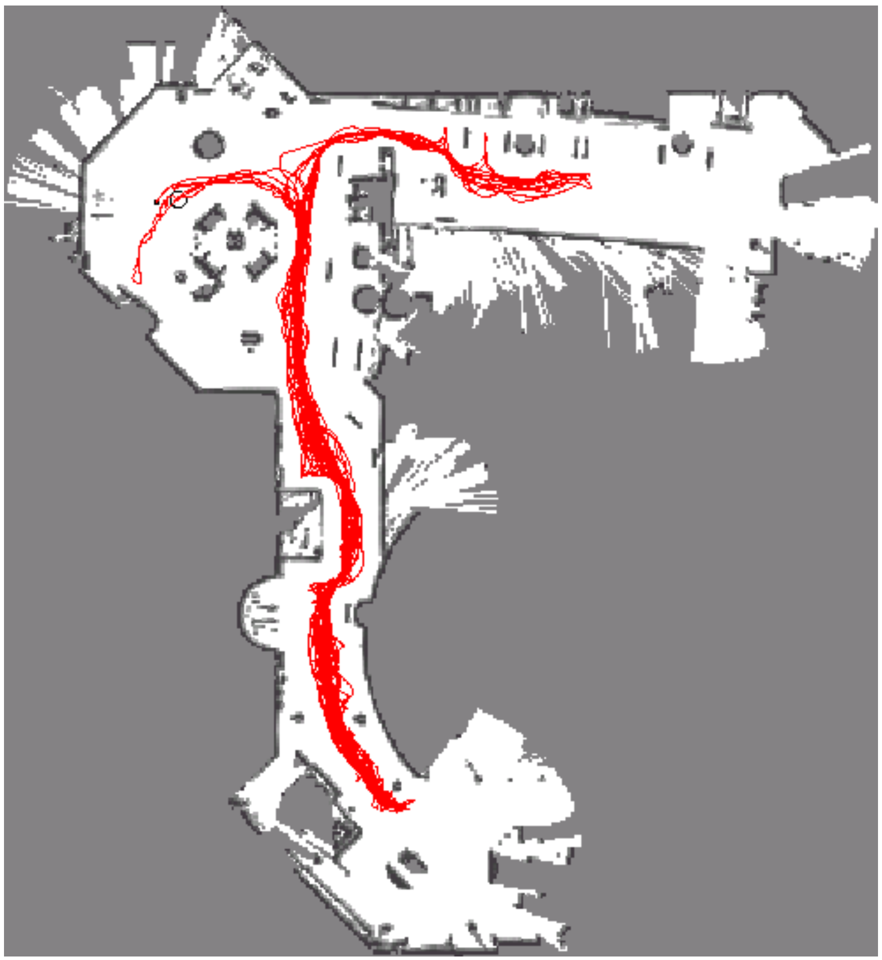


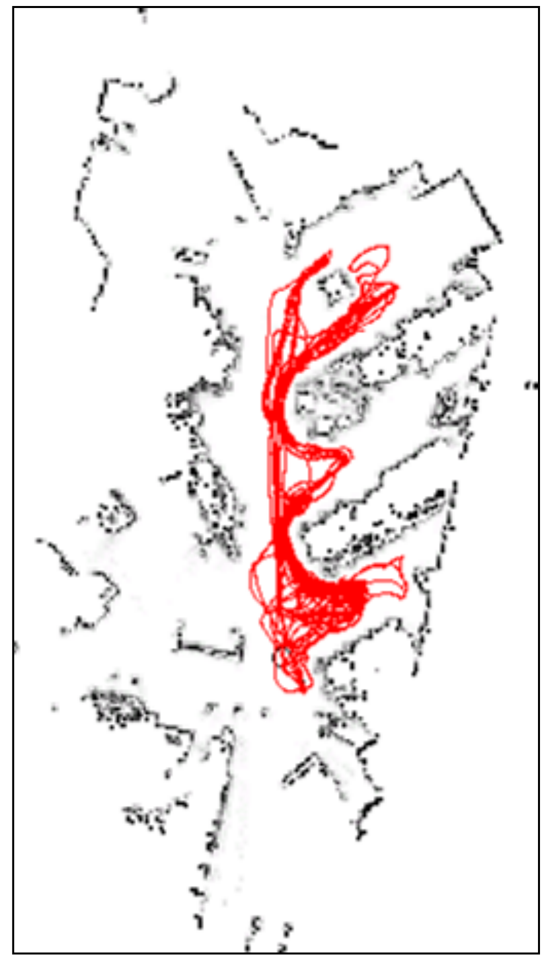
Global Localization Using Vision



Robots in Action: Albert

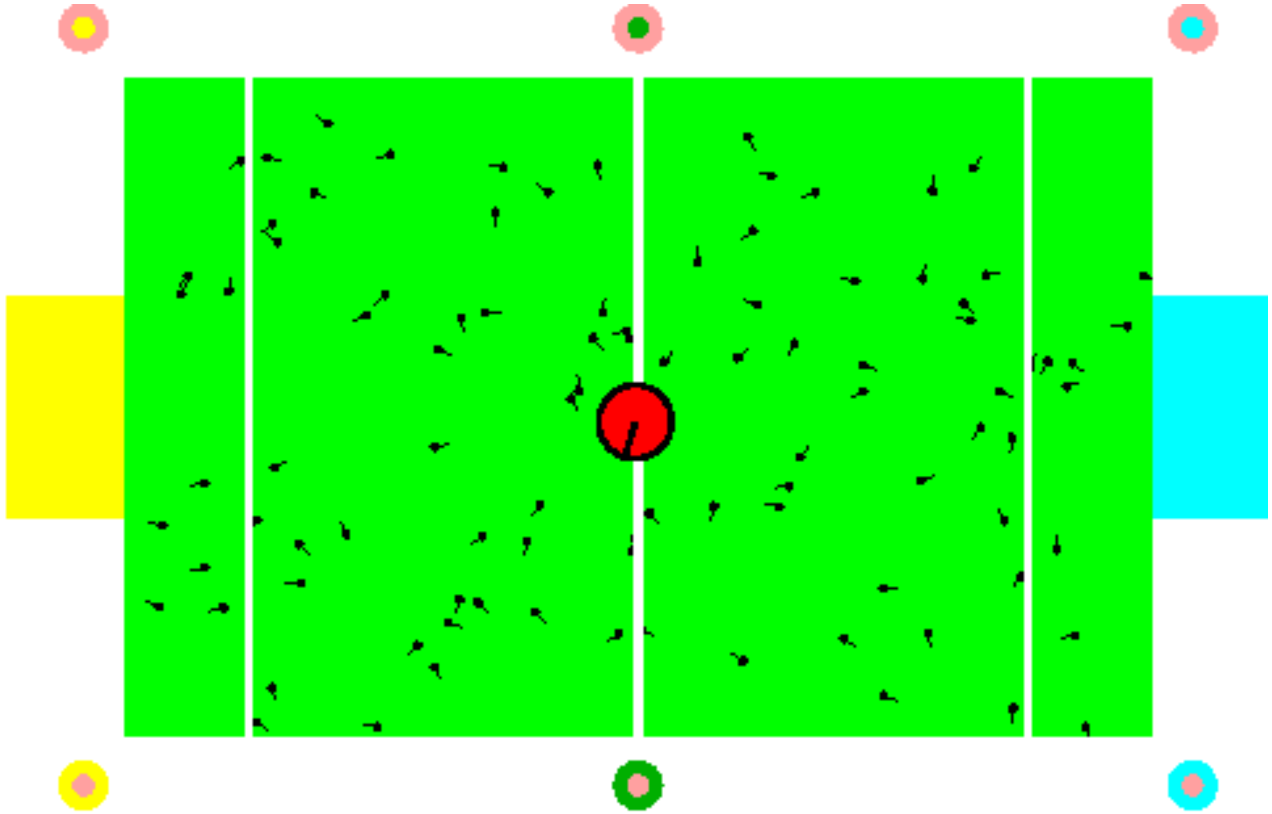
Application: Rhino and Albert Synchronized in Munich and Bonn





[Robotics And Automation Magazine, to appear]

Localization for AIBO robots



Limitations

- 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

- Randomly insert samples (the robot can be teleported at any point in time).
- Insert random samples proportional to the average likelihood of the particles (the robot has been teleported with higher probability when the likelihood of its observations drops).

Random Samples Vision-Based Localization

936 Images, 4MB, .6secs/image

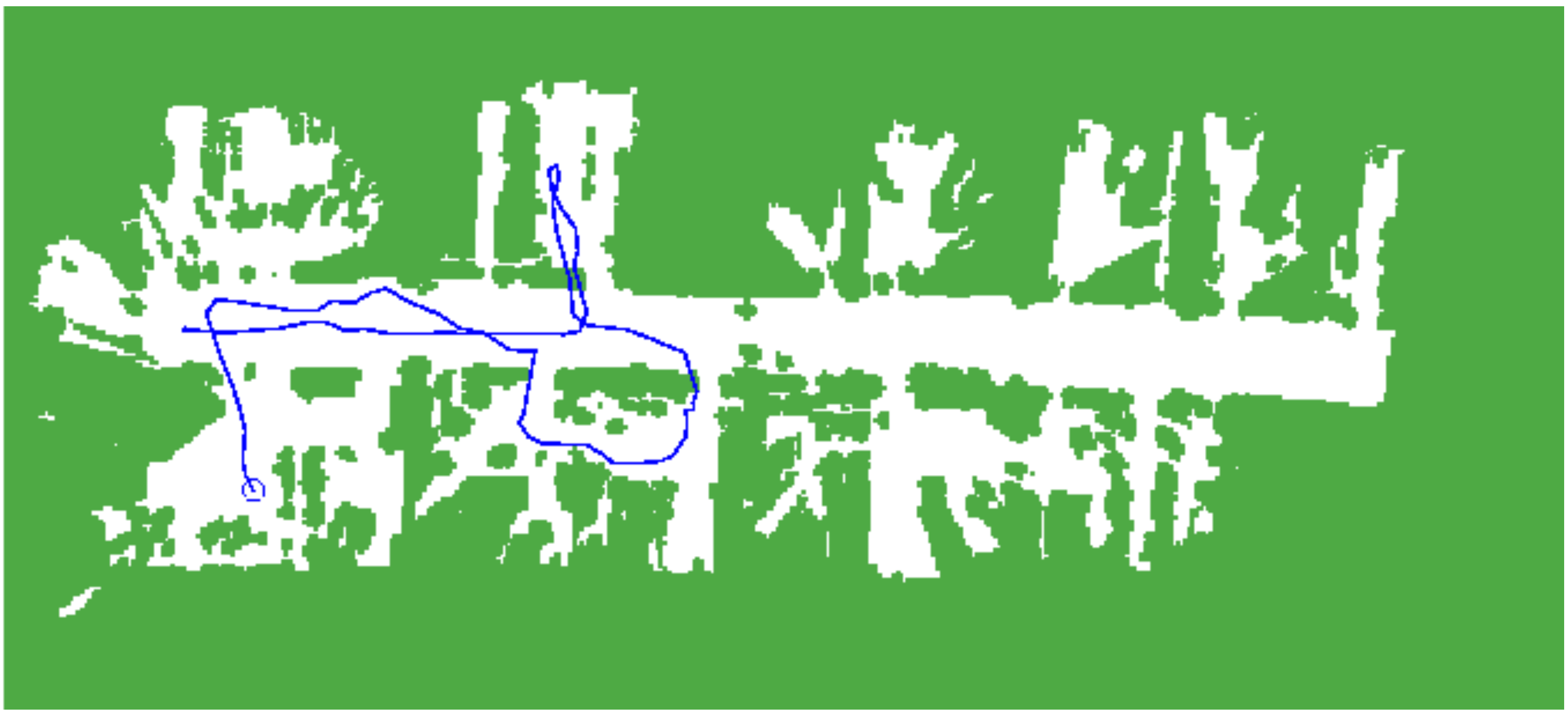
Trajectory of the robot:

Odometry Information

Image Sequence

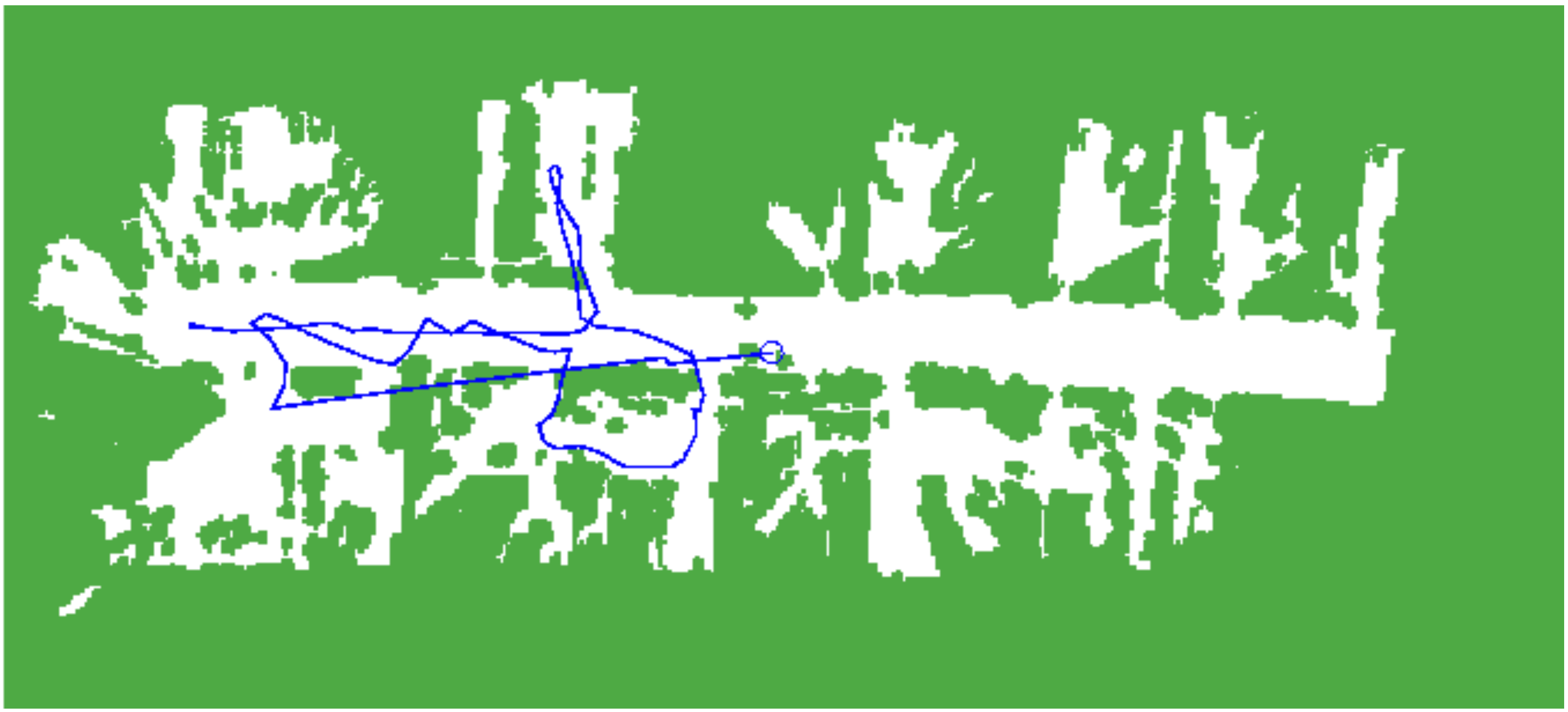
Resulting Trajectories

Position tracking:

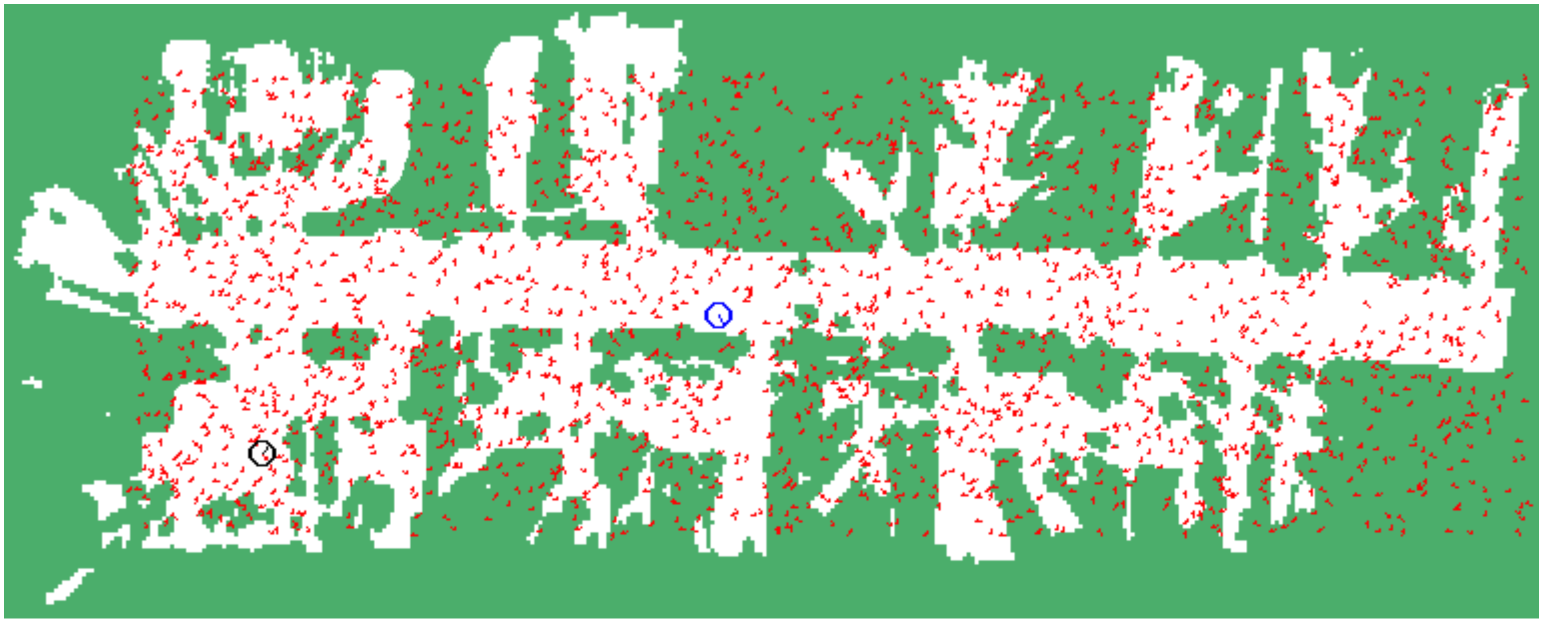


Resulting Trajectories

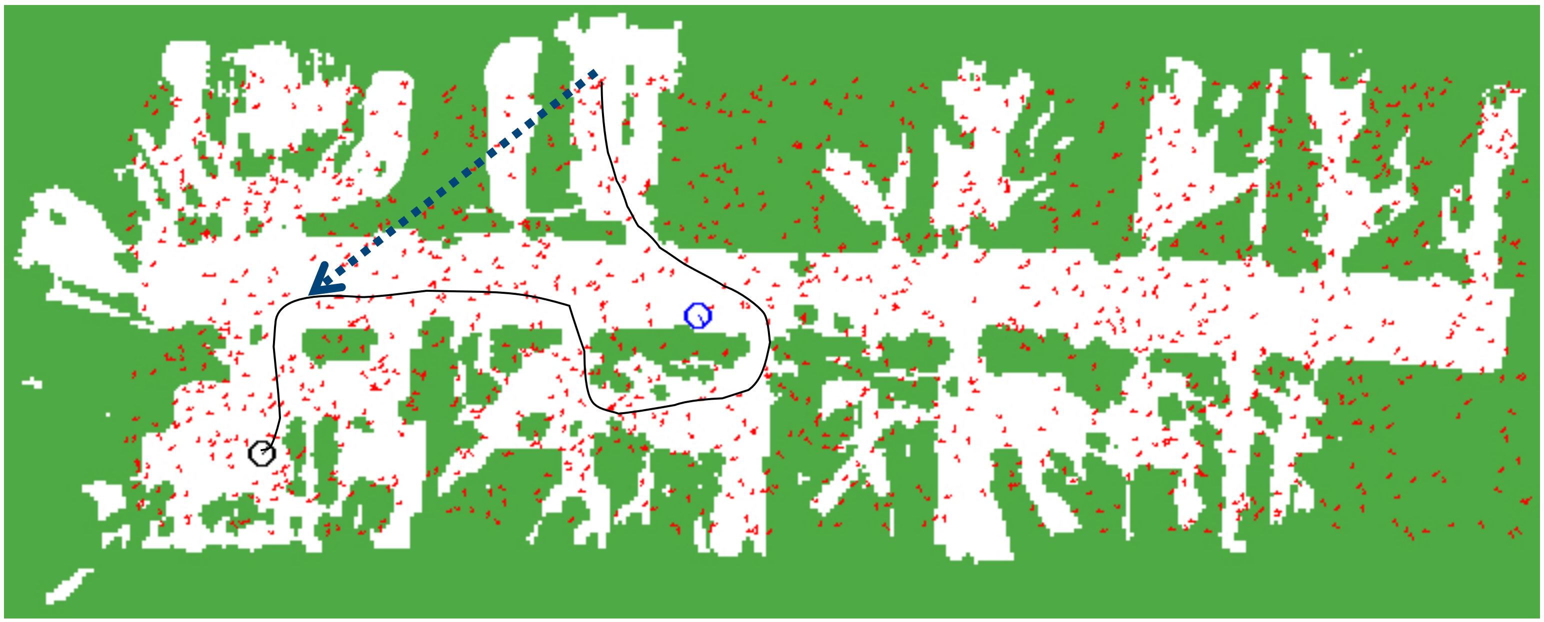
Global localization:



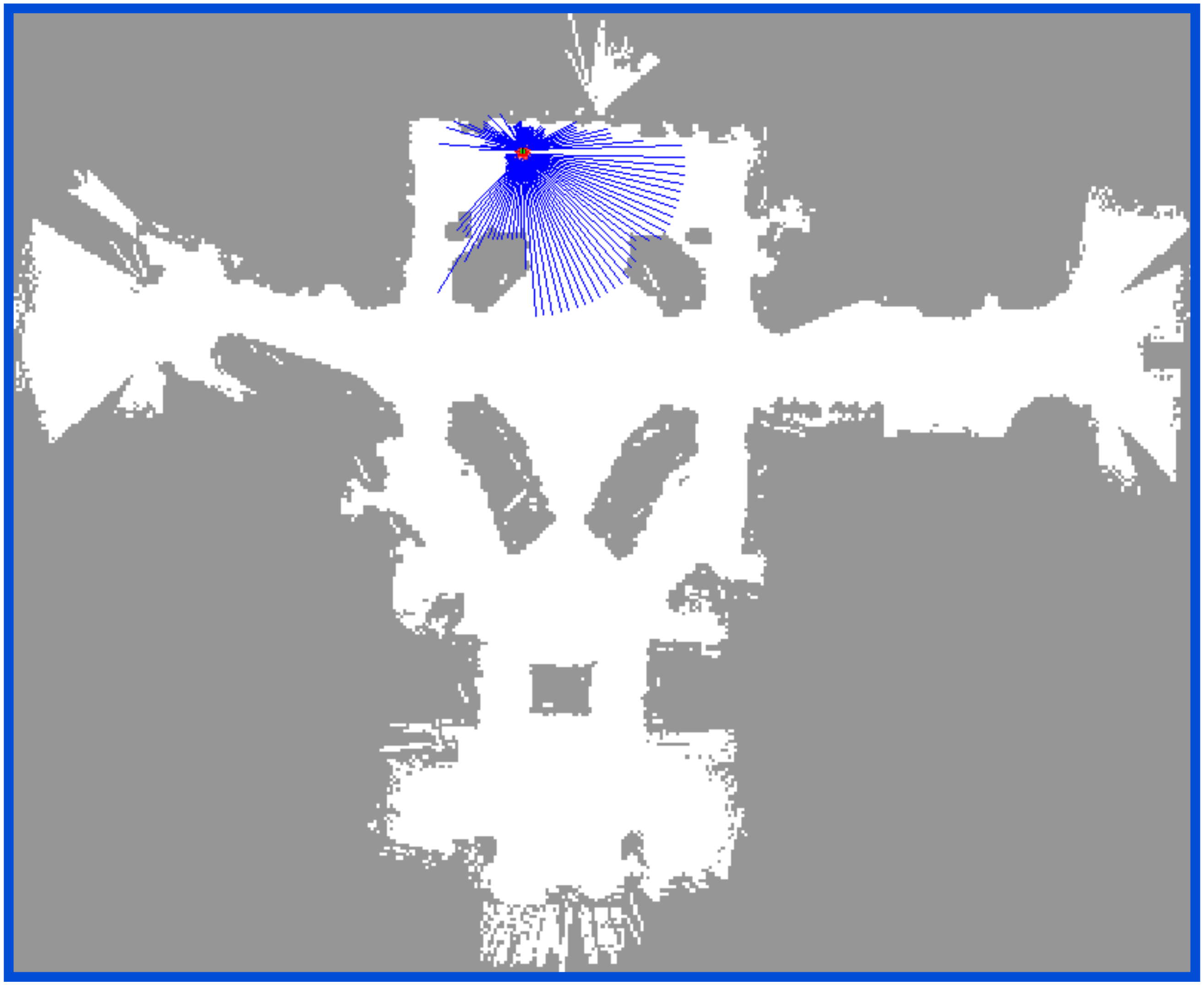
Global Localization



Kidnapping the Robot



Recovery from Failure



Importance Sampling with Resampling: Landmark Detection Example

Probabilistic Model

1. Algorithm **landmark_detection_model**(z,x,m):
   $z = \langle i, d, \alpha \rangle, x = \langle x, y, \theta \rangle$
2. $\hat{d} = \sqrt{(m_x(i) - x)^2 + (m_y(i) - y)^2}$
3. $\hat{\alpha} = \mathrm{atan2}(m_y(i) - y, m_x(i) - x) - \theta$
4. $p_{\mathrm{det}} = \mathrm{prob}(\hat{d} - d, \varepsilon_d) \cdot \mathrm{prob}(\hat{\alpha} - \alpha, \varepsilon_\alpha)$
   $z_{\mathrm{det}} p_{\mathrm{det}} + z_{\mathrm{fp}} P_{\mathrm{uniform}}(z \mid x, m)$
5. Return

Computing likelihood of landmark measurement

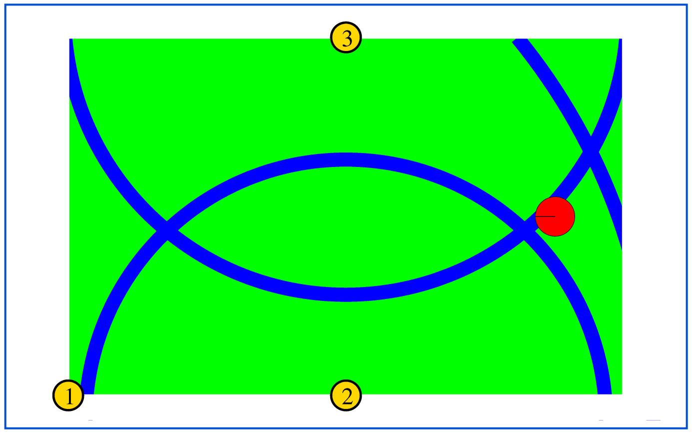
Landmark – distance, bearing and signature

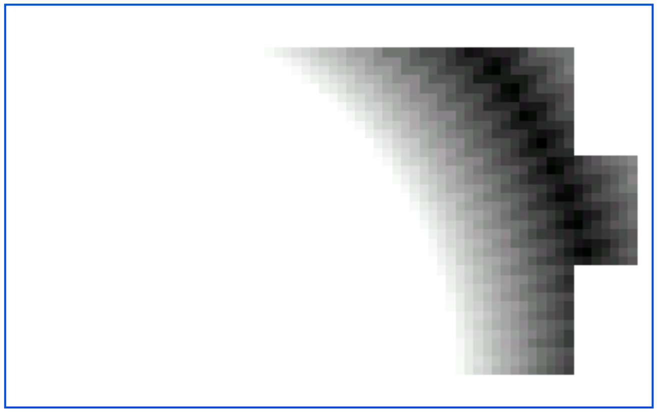
Landmark model

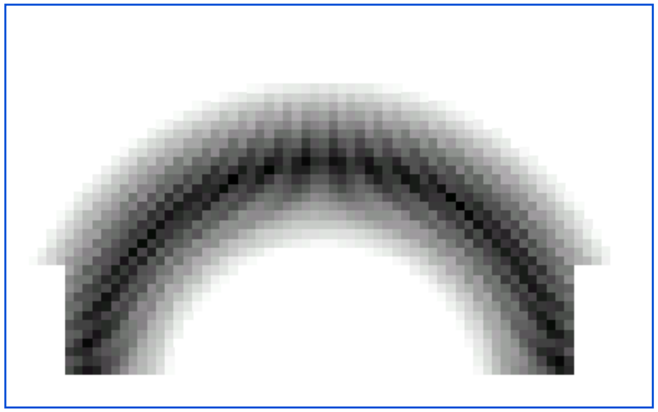
- How to sample likely poses, given the landmark measurement ?
- Given a landmark measurement, robot can lie on a circle. Generate samples along the circle.

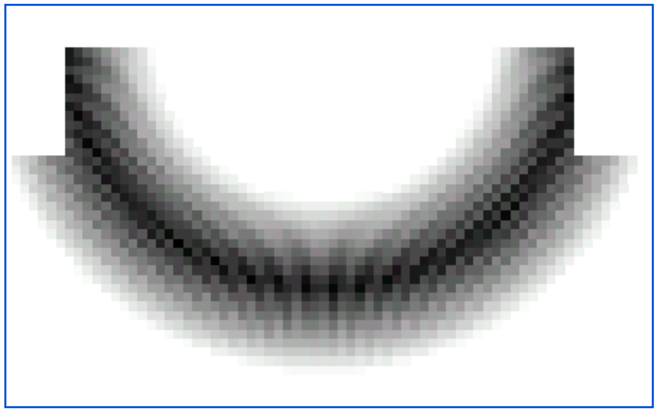
- Idea: use the inverse sensor model. Given distance and bearing (possibly noisy), generate samples along circle.
- Do it for every landmark

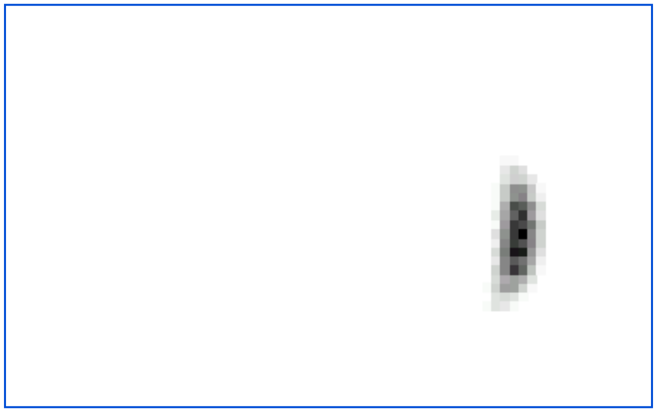
Distributions



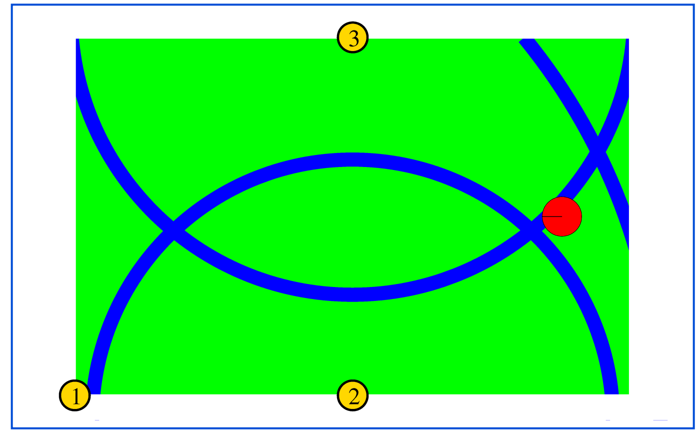


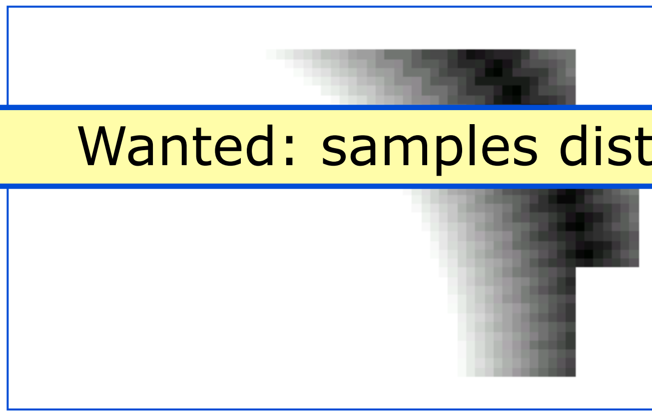


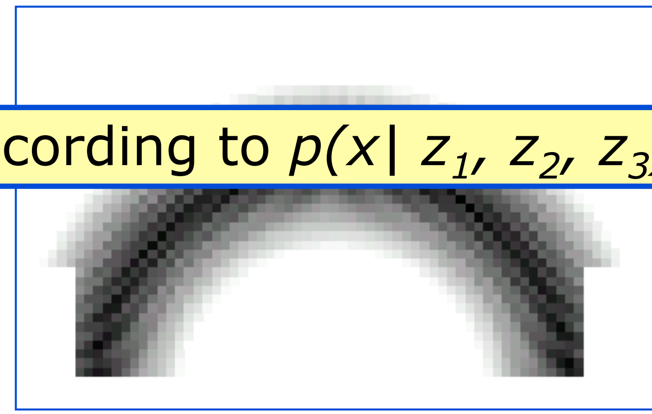




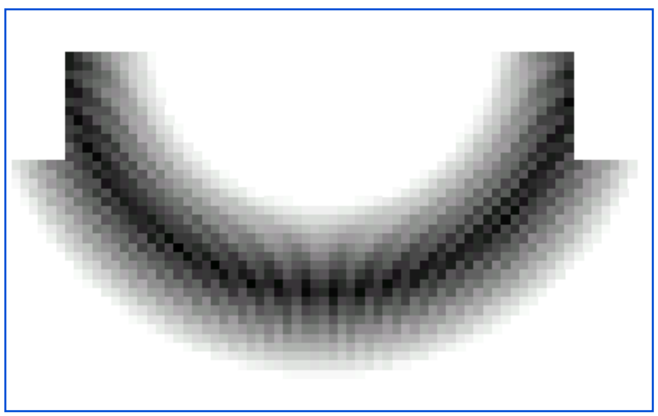
Distributions





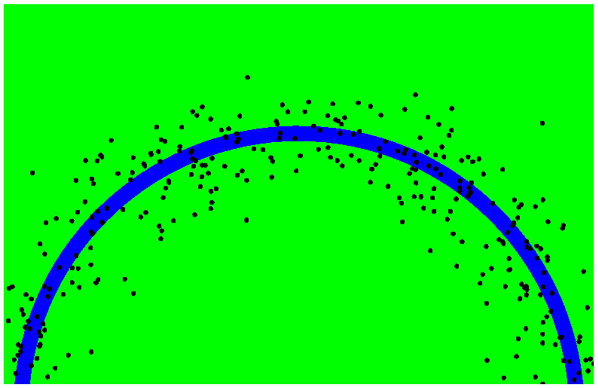


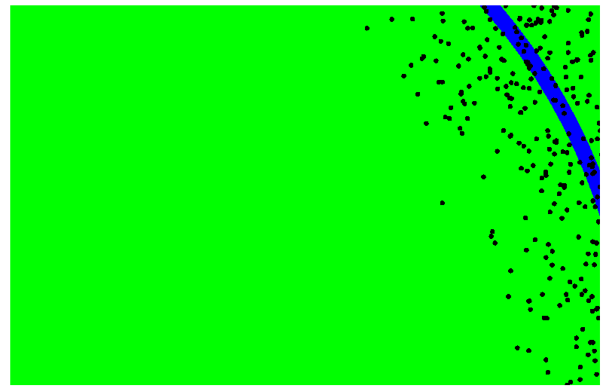
Wanted: samples distributed according to $p(x|\ z_1, z_2, z_3)$

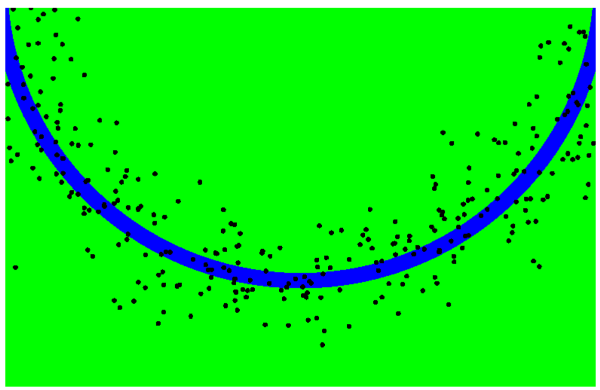


This is Easy!

We can draw samples from $p(x|z_l)$ by adding noise to the detection parameters.







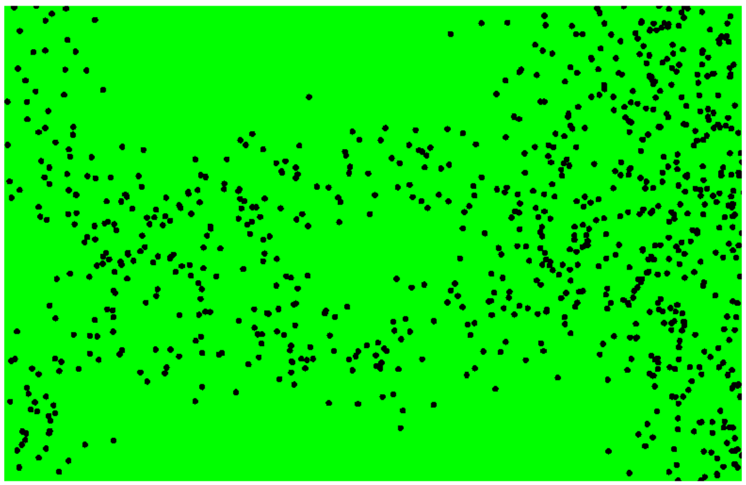
Importance Sampling with Resampling

$$\text{Target distribution f}: p(x \mid z_1, z_2, \ldots, z_n) = \frac{\prod_k p(z_k \mid x) \quad p(x)}{p(z_1, z_2, \ldots, z_n)}$$

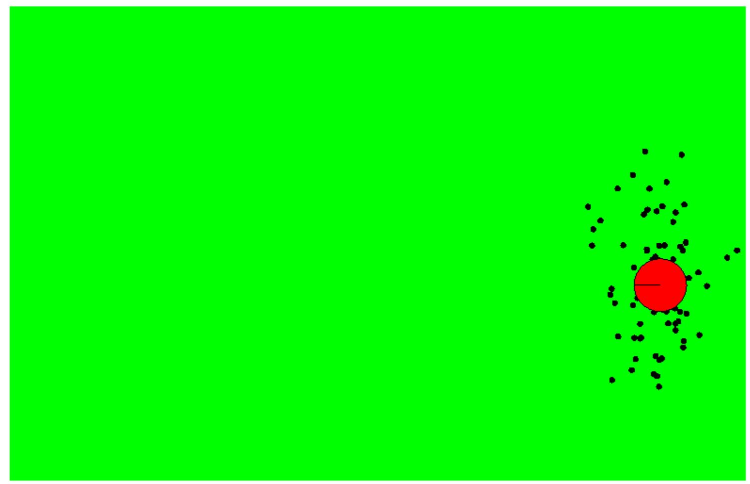
$$\text{Sampling distribution g}: p(x \mid z_l) = \frac{p(z_l \mid x) p(x)}{p(z_l)}$$

$$\text{Importance weights w}: \frac{f}{g} = \frac{p(x \mid z_1, z_2, \ldots, z_n)}{p(x \mid z_l)} = \frac{p(z_l) \prod_{k \neq l} p(z_k \mid x)}{p(z_1, z_2, \ldots, z_n)}$$

Importance Sampling with Resampling



Weighted samples



After resampling

http://www.youtube.com/watch?v=ABzzFMzFE3Y