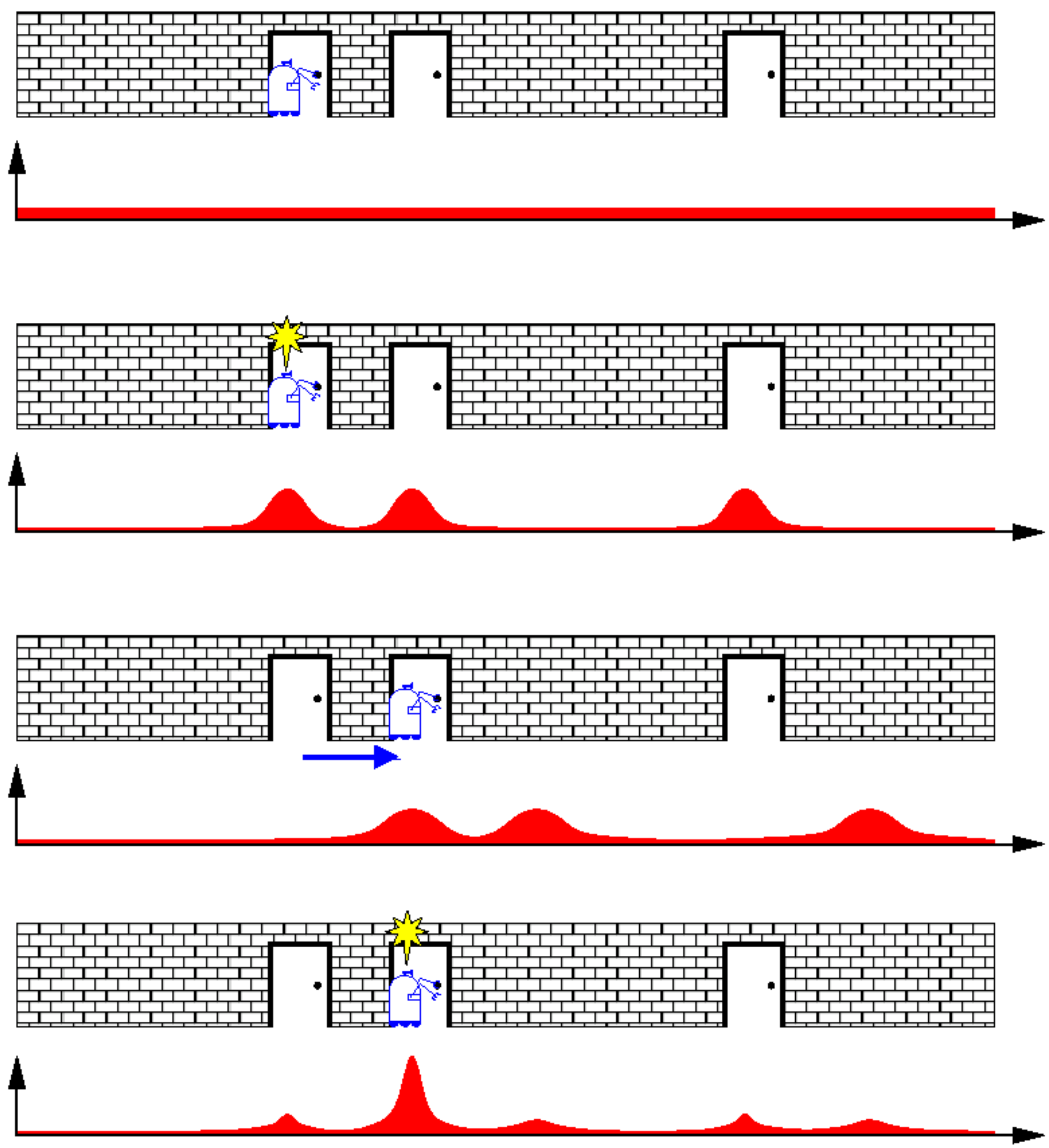
Probabilistic Robotics

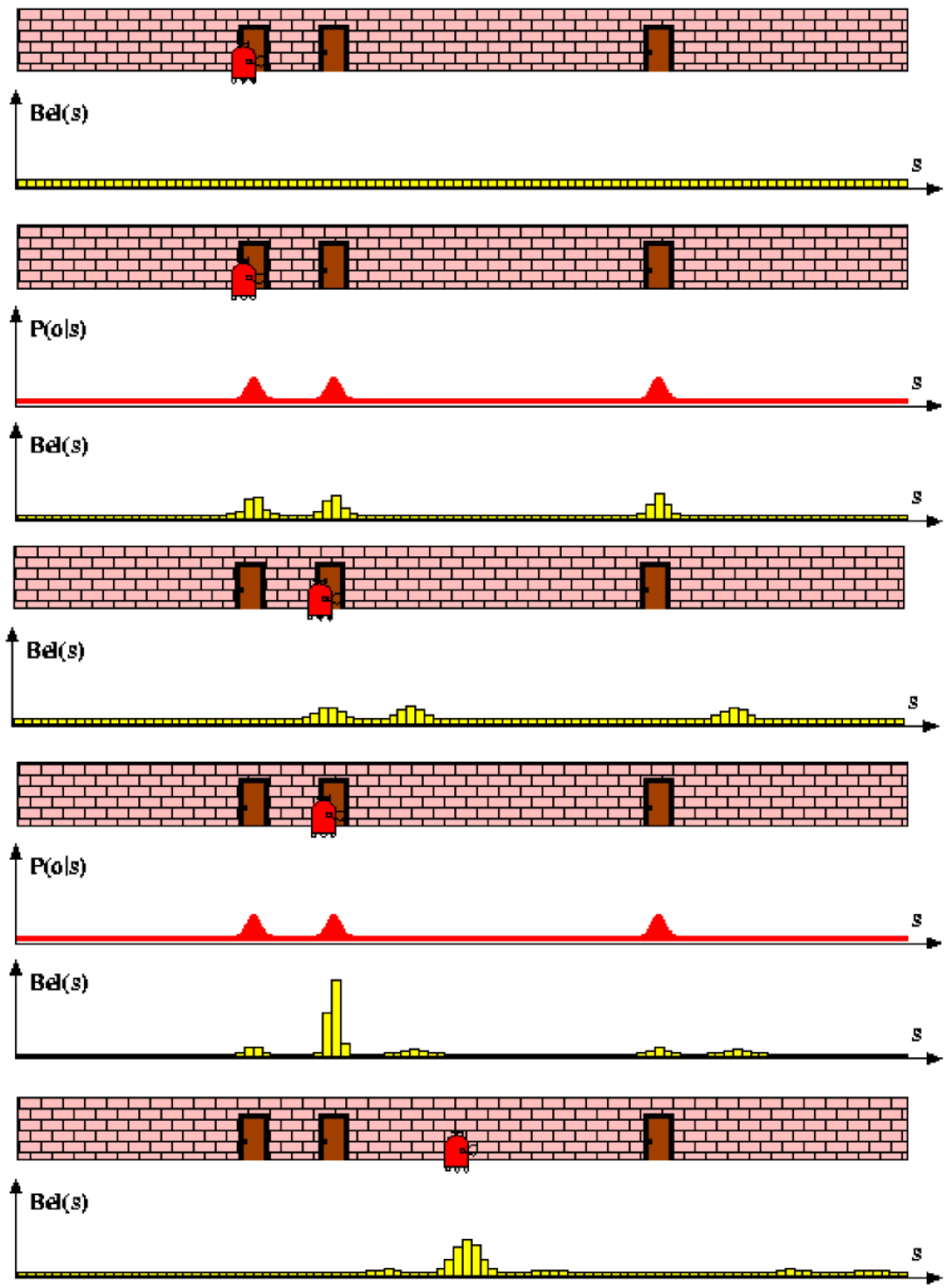
Discrete Filters and Particle Filters Models

Some slides adopted from: Wolfram Burgard, Cyrill Stachniss, Maren Bennewitz, Kai Arras and Probabilistic Robotics Book

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



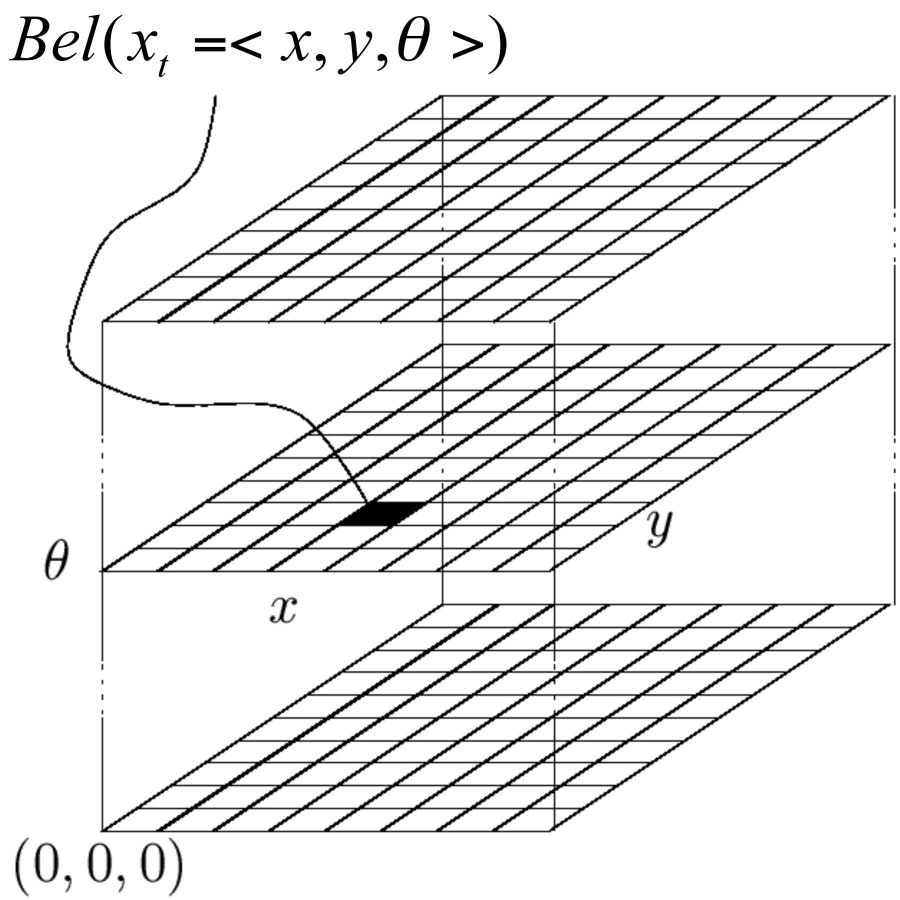
Piecewise Constant



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.

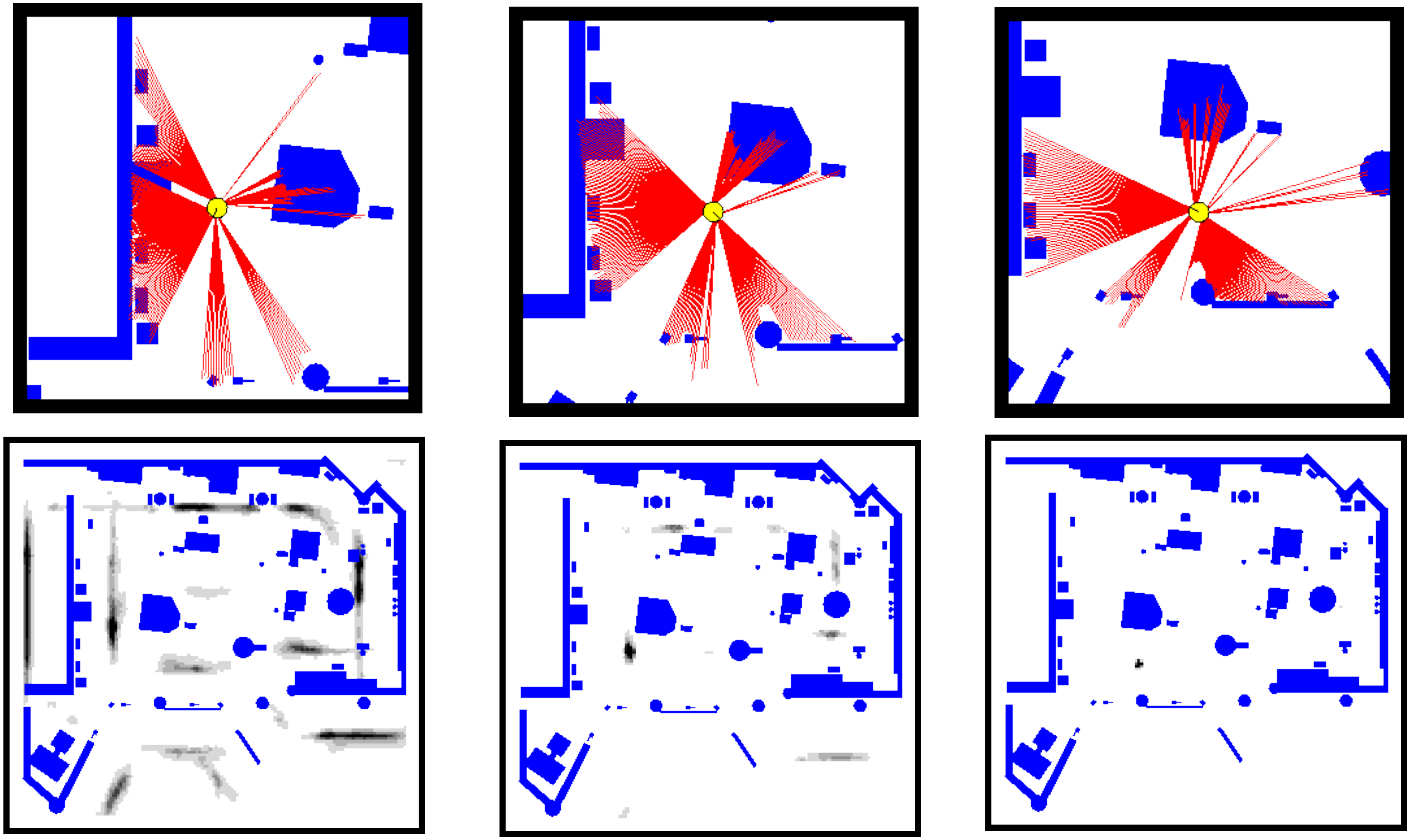
Implementation (2)

- 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:

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

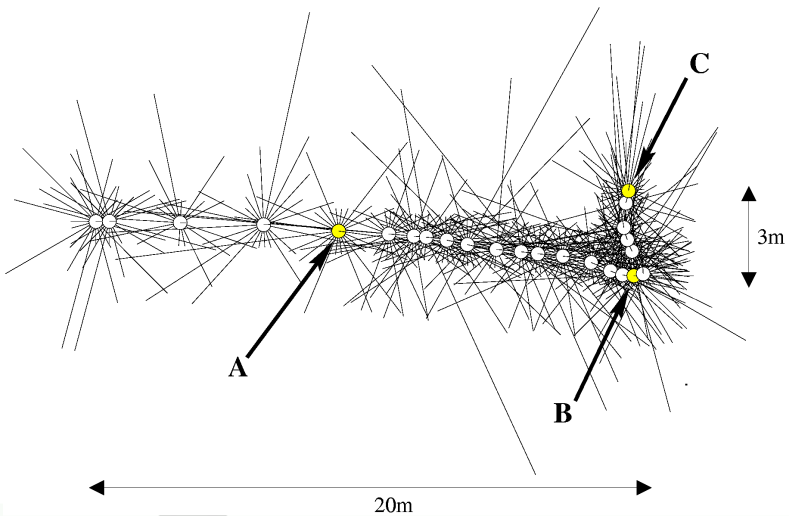
  - Fewer arithmetic operations
  - Easier to implement

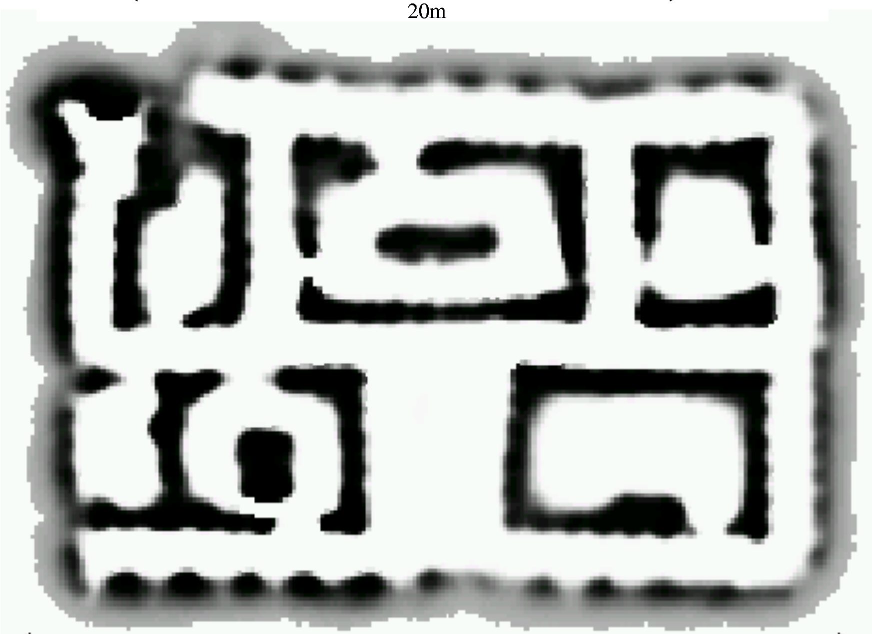
Grid-based Localization

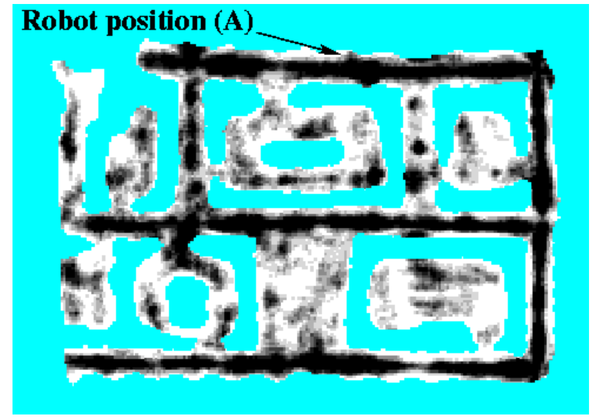


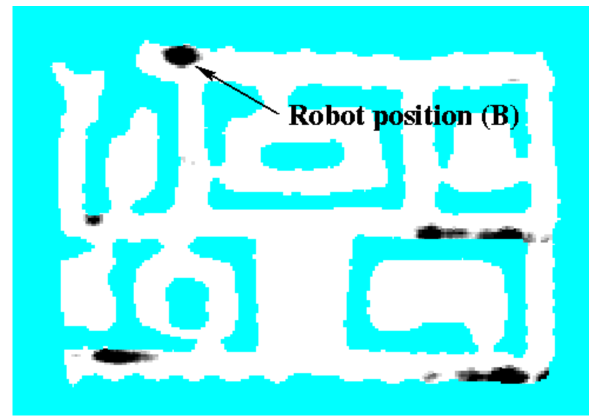
Application Example: Rhino

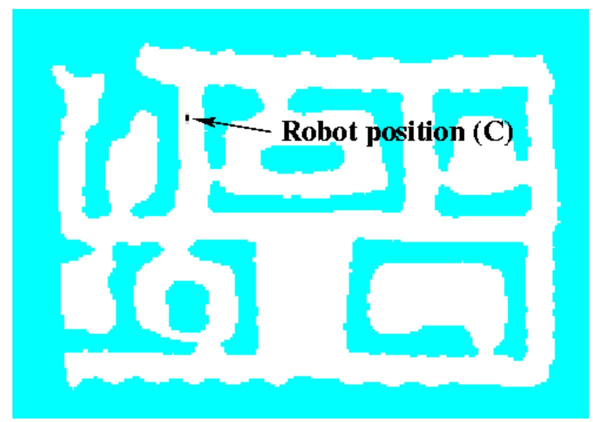
Sonars and Occupancy Grid Map











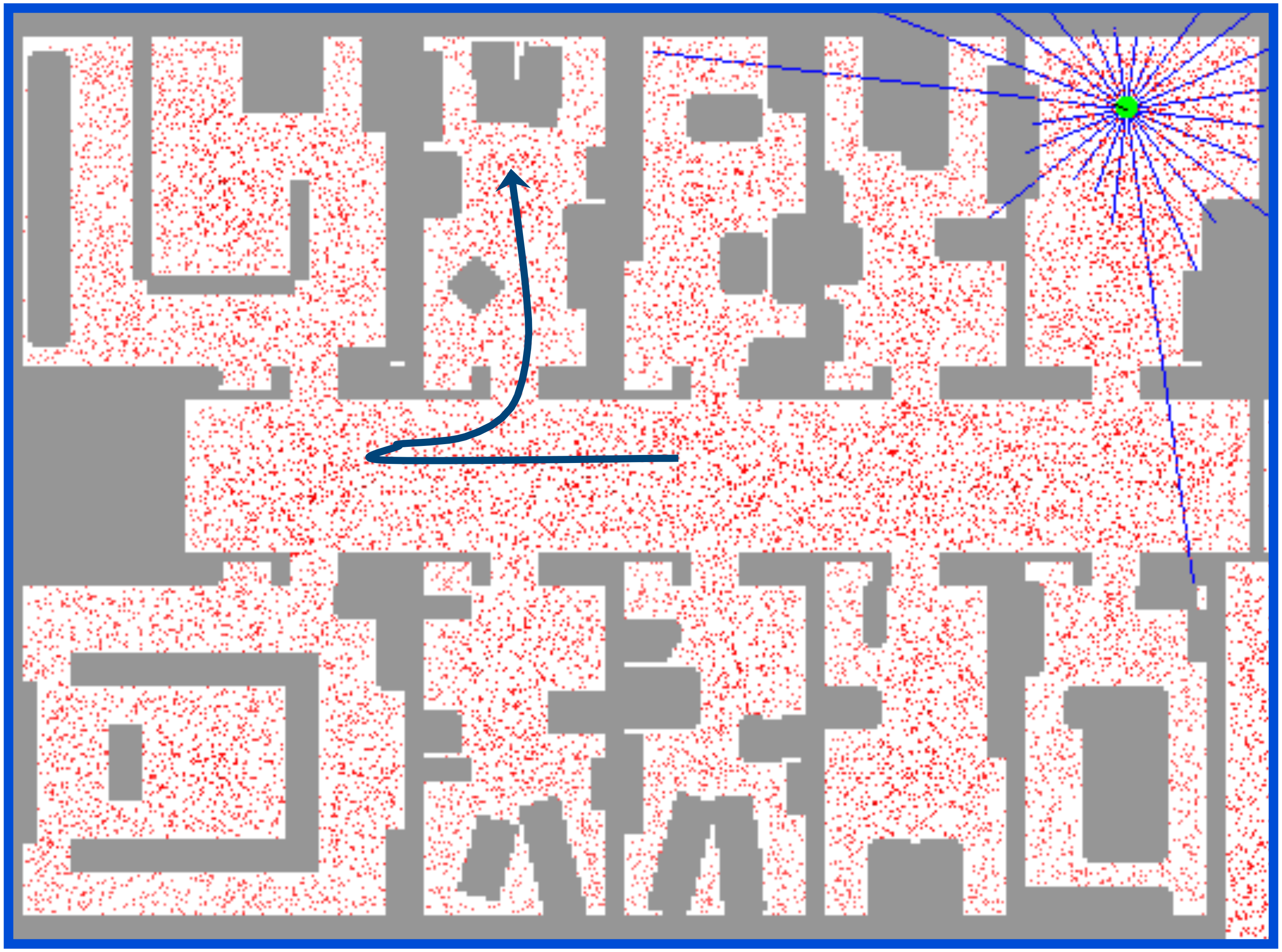
Motivation

- Recall: Discrete filter
  - Discretize the continuous state space
  - High memory complexity
  - Fixed resolution (does not adapt to the belief)

- Particle filters are a way to **efficiently** represent **non-Gaussian distribution**

- Basic principle
  - Set of state hypotheses ("particles")
  - Survival-of-the-fittest

Sample-based Localization (sonar)



Mathematical Description

- Set of weighted samples

$$S = \left\{ \left\langle s^{[i]}, w^{[i]} \right\rangle \mid i = 1, \ldots, N \right\}$$

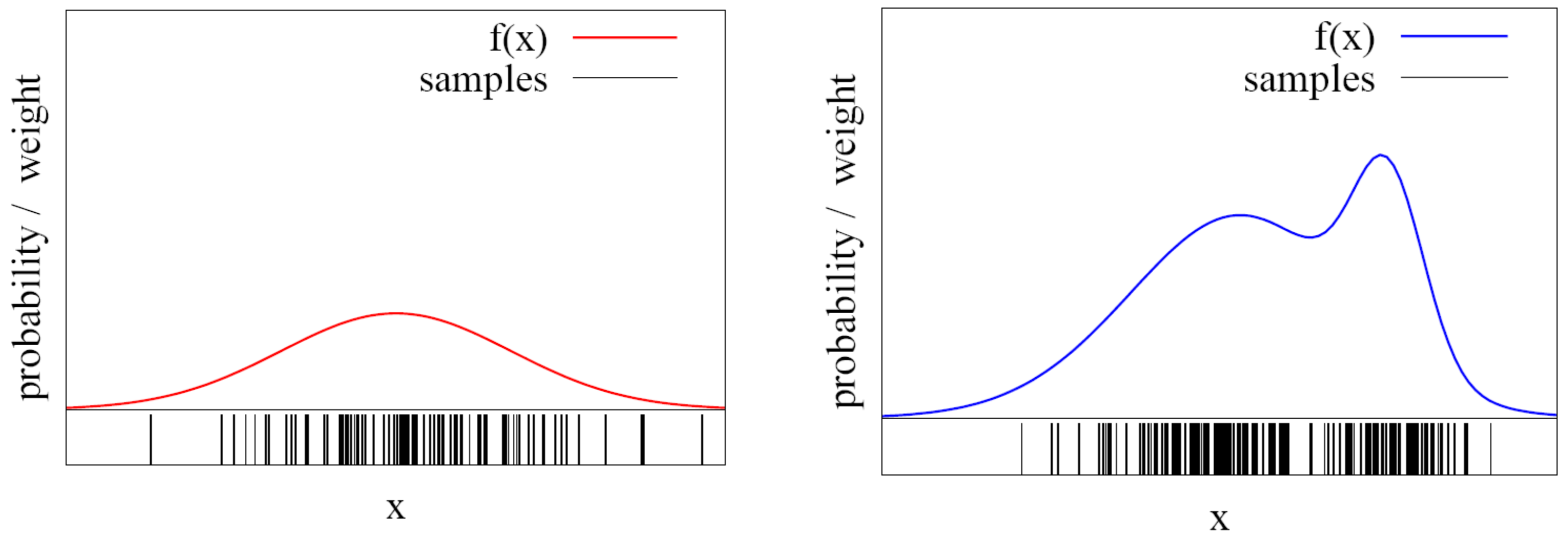
State hypothesis   Importance weight

- The samples represent the posterior

$$p(x) = \sum_{i=1}^{N} w_i \cdot \delta_{s^{[i]}}(x)$$

Function Approximation

- Particle sets can be used to approximate functions

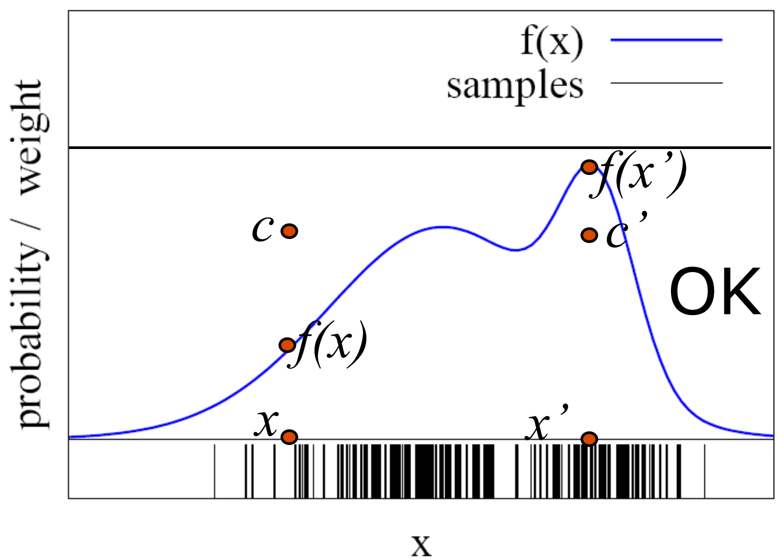


- The more particles fall into an interval, the higher the probability of that interval

- How to draw samples form a function/distribution?

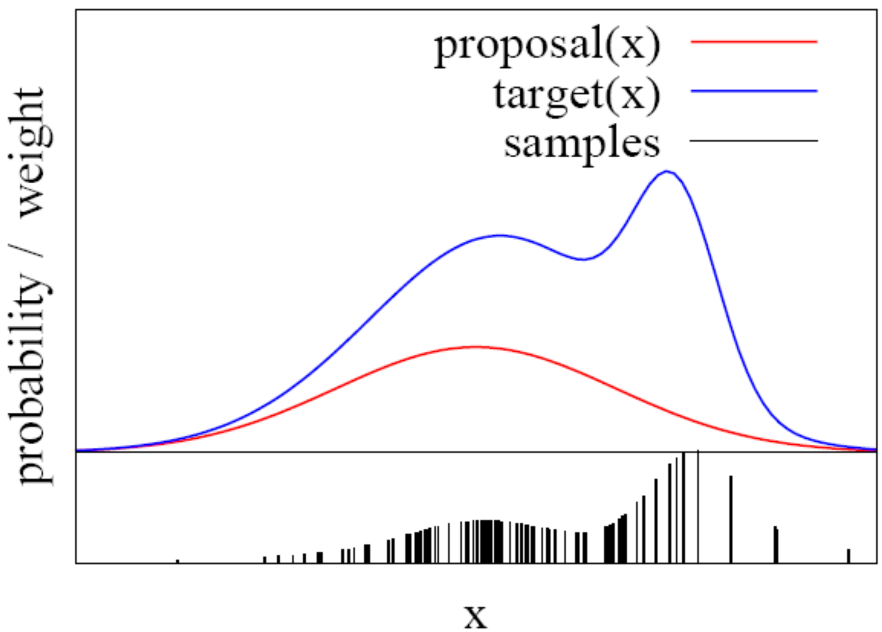
Rejection Sampling

- Let us assume that $f(x)<1$ for all x
- Sample x from a uniform distribution
- Sample c from [0,1]
- if $f(x) > c$ keep the sample
  otherwise reject the sampe

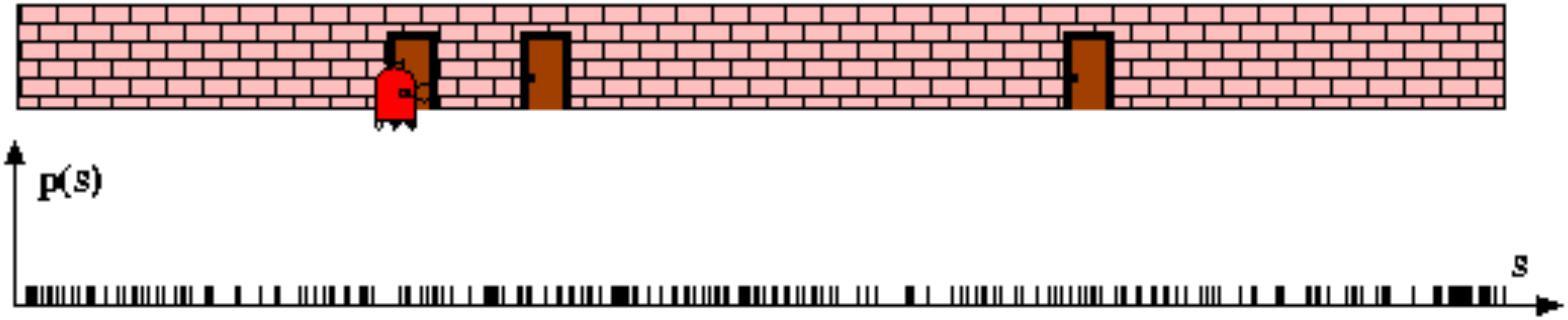


Importance Sampling Principle

- We can even use a different distribution g to generate samples from f
- By introducing an importance weight w, we can account for the "differences between g and f"
- $w = f / g$
- f is often called target
- g is often called proposal
- Pre-condition: $f(x)>0 \rightarrow g(x)>0$



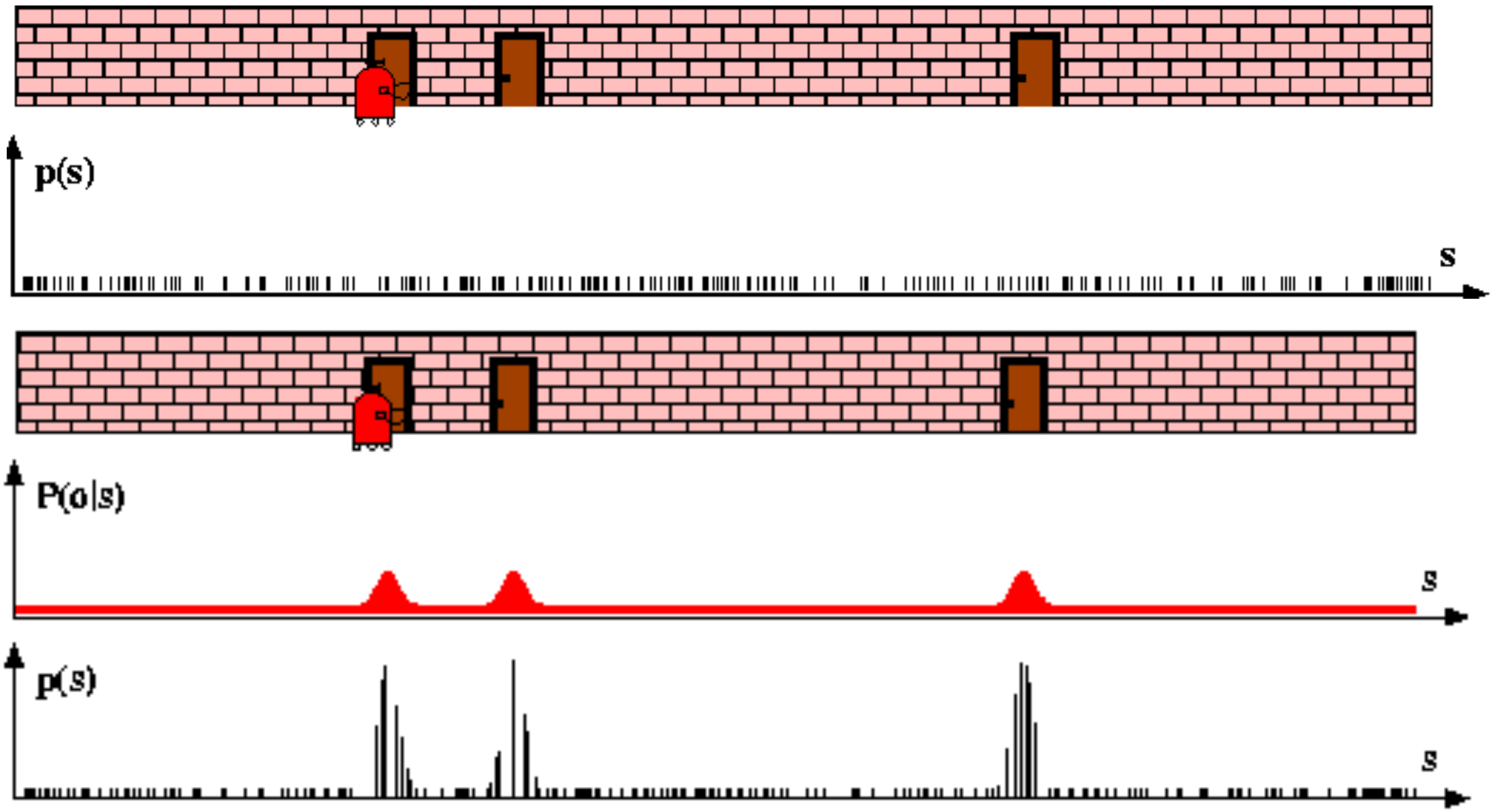
Particle Filters



Sensor Information: Importance Sampling

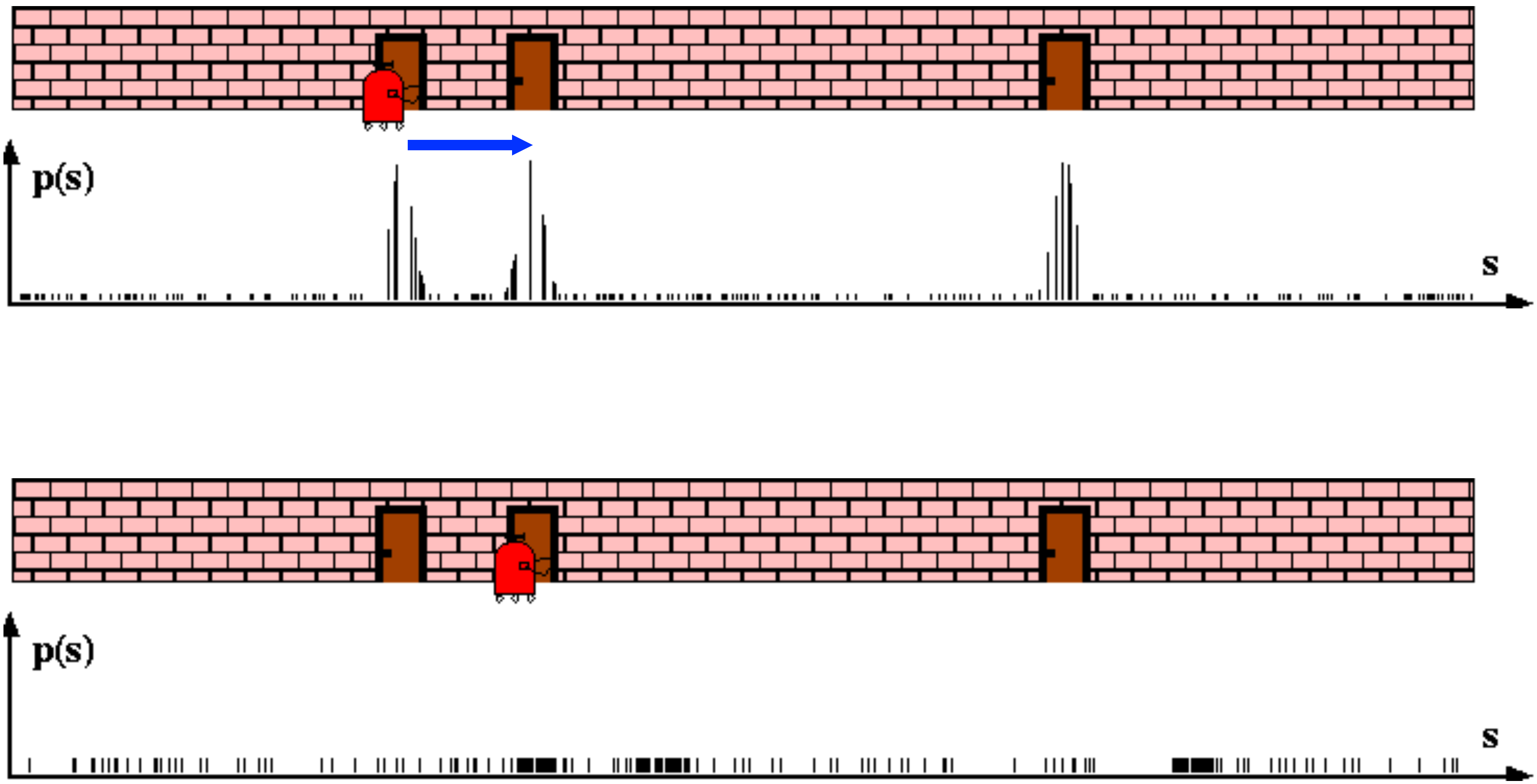
$$Bel(x) \leftarrow \alpha\ p(z \mid x)\, Bel^{-}(x)$$

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



Robot Motion

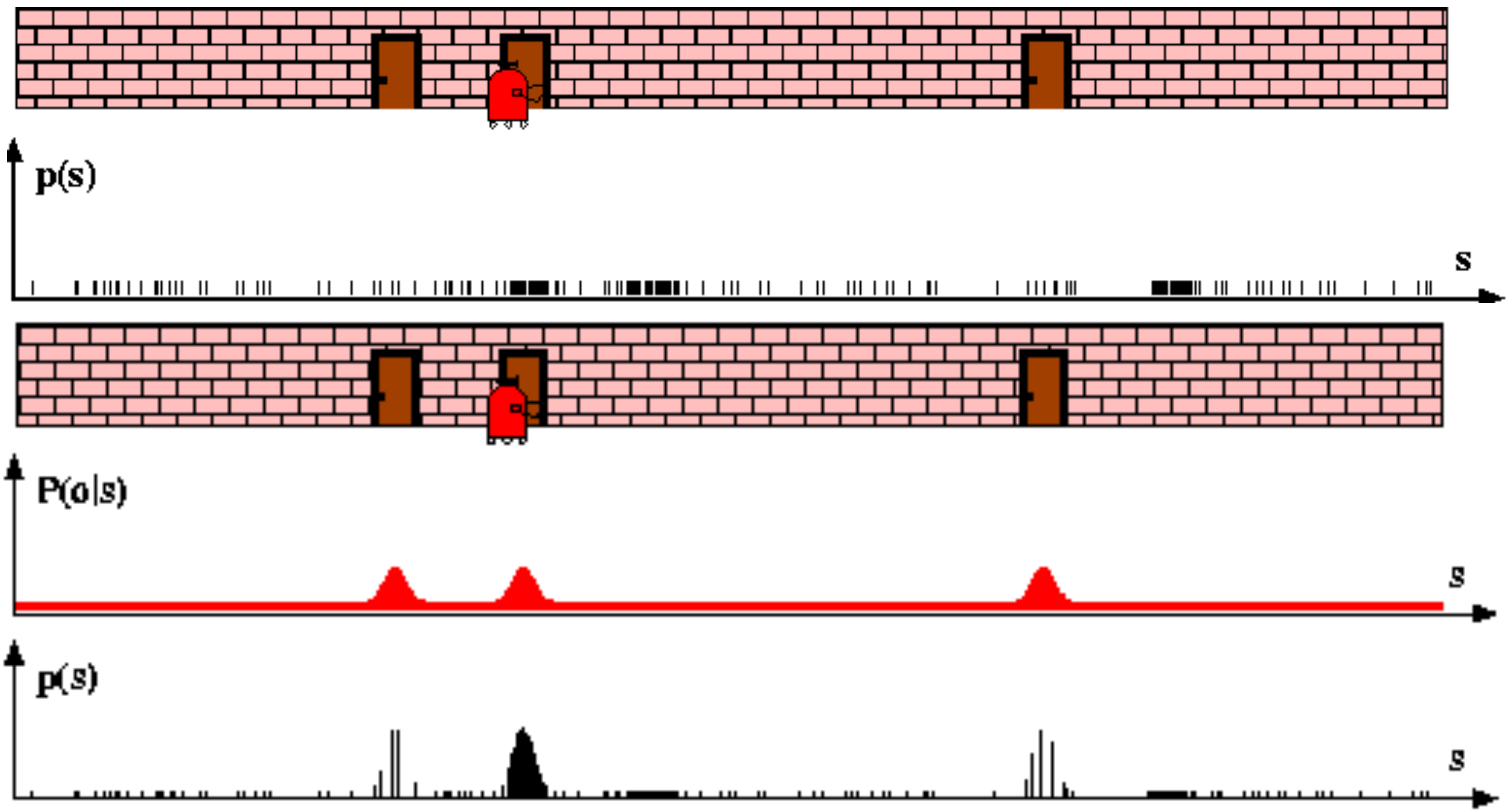
$$Bel^{-}(x) \quad \leftarrow \quad \int p(x \mid u, x')\, Bel(x') \ \mathrm{d}\, x'$$



Sensor Information: Importance Sampling

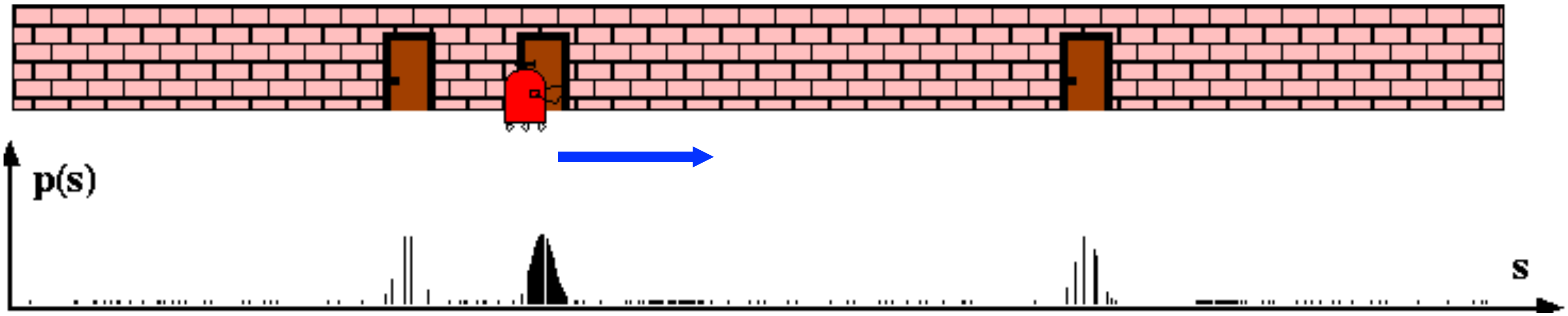
$$Bel(x) \leftarrow \alpha\ p(z \mid x)\, Bel^{-}(x)$$

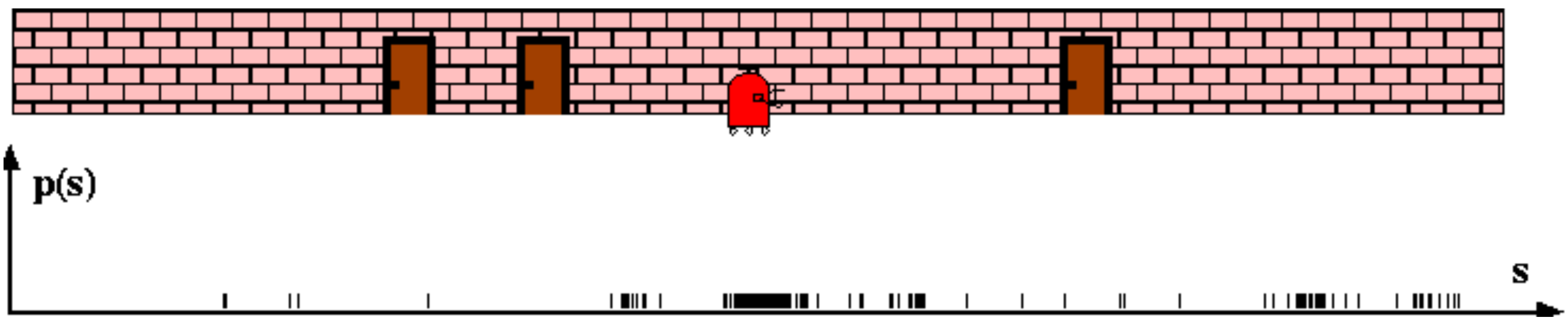
$$w \leftarrow \frac{\alpha\ p(z \mid x)\, Bel^{-}(x)}{Bel^{-}(x)} = \alpha\ p(z \mid x)$$



Robot Motion

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





Particle Filter Algorithm

- Sample the next generation for particles using the proposal distribution

- Compute the importance weights :

  $$weight = target\ distribution\ /\ proposal\ distribution$$

- Resampling: “Replace unlikely samples by more likely ones”

- [Derivation of the MCL equations on the blackboard]

Particle Filter Algorithm

1. Algorithm **particle_filter**(S_{t-1}, u_{t-1} z_t):
2. $S_t = \emptyset, \quad \eta = 0$
3. **For** $i = 1 \dots n$ ***Generate new samples***
4. Sample index *j(i)* from the discrete distribution given by w_{t-1}
5. 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}
6. $w_t^i = p(z_t \mid x_t^i)$ ***Compute importance weight***
7. $\eta = \eta + w_t^i$ ***Update normalization factor***
8. $S_t = S_t \cup \{< x_t^i, w_t^i >\}$ ***Insert***
9. **For** $i = 1 \dots n$
10. $w_t^i = w_t^i / \eta$ ***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:

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

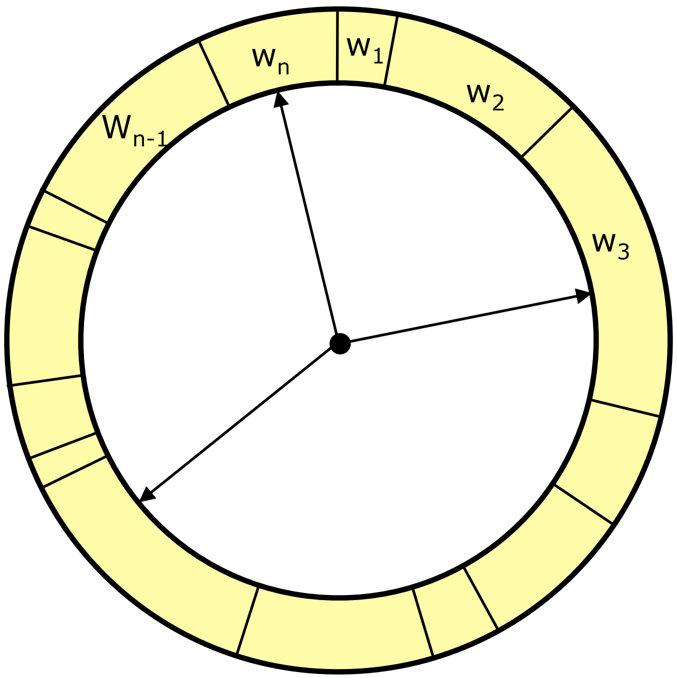
$$= \frac{\eta\; p(z_t \mid x_t)\; p(x_t \mid x_{t-1}, u_{t-1})\; Bel\,(x_{t-1})}{p(x_t \mid x_{t-1}, u_{t-1})\; Bel\,(x_{t-1})}$$

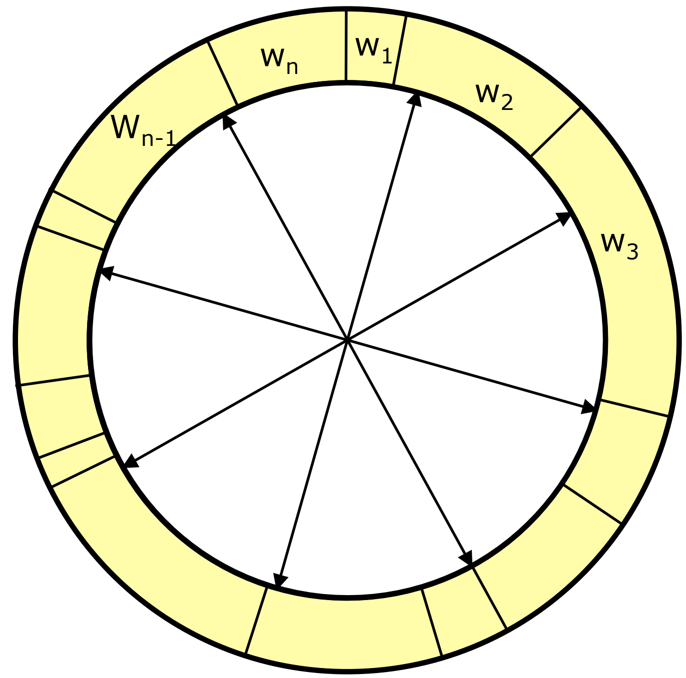
$$\propto p(z_t \mid x_t)$$

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. $\quad c_i = c_{i-1} + w^i$
5. $u_1 \sim U[0, n^{-1}], i = 1$ ***Initialize threshold***

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

11. **Return** *S'*

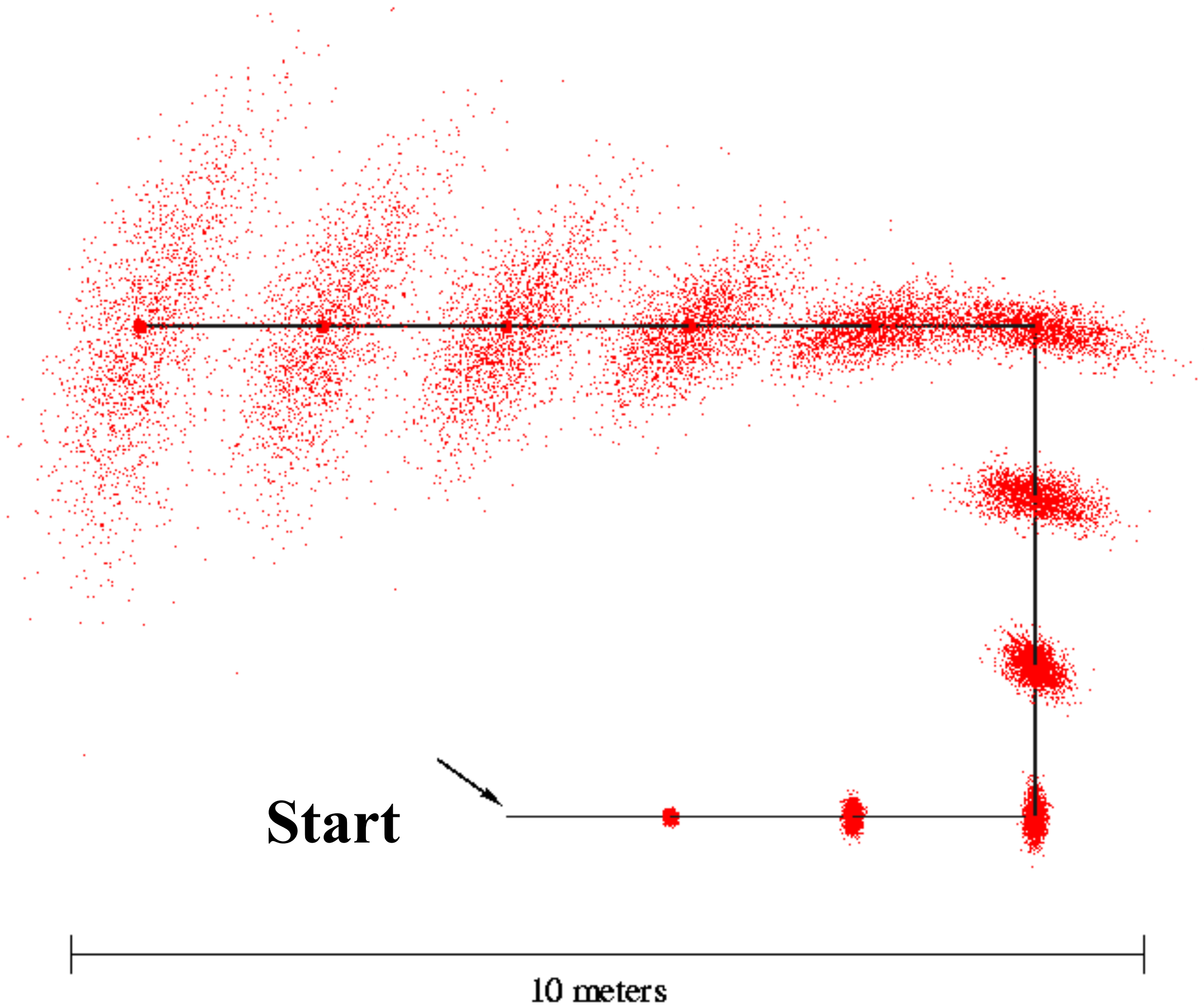
Also called **stochastic universal sampling**

Mobile Robot Localization

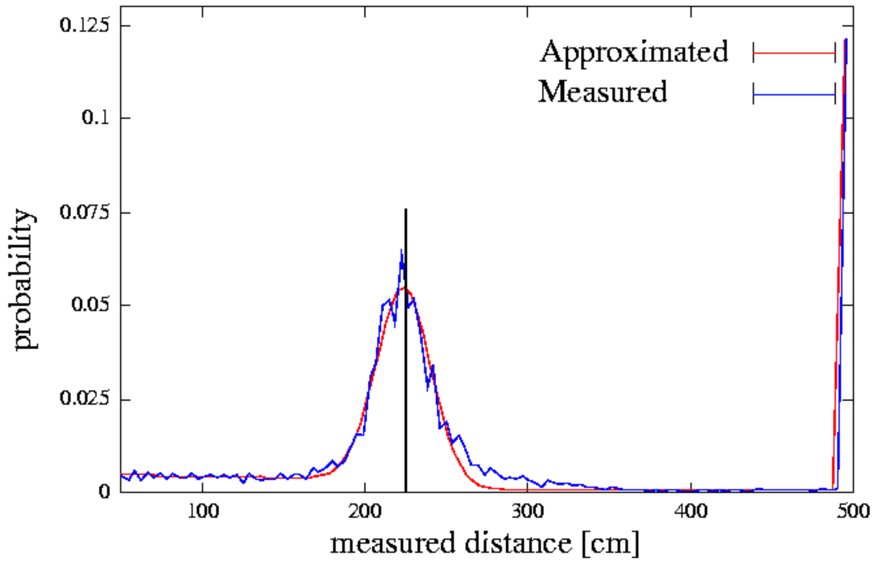
- Each particle is a potential pose of the robot
- Proposal distribution is the motion model of the robot (prediction step)
- The observation model is used to compute the importance weight (correction step)

[For details, see PDF file on the lecture web page]

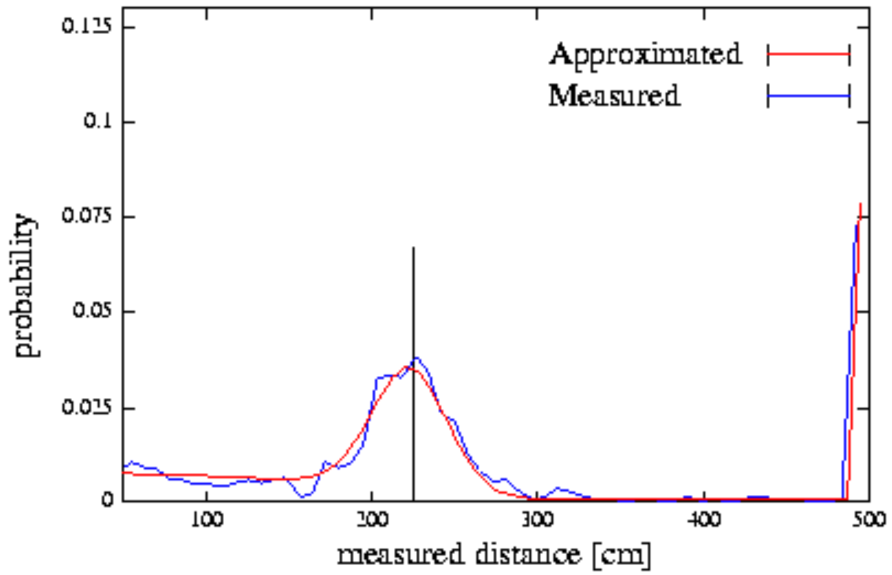
Motion Model Reminder



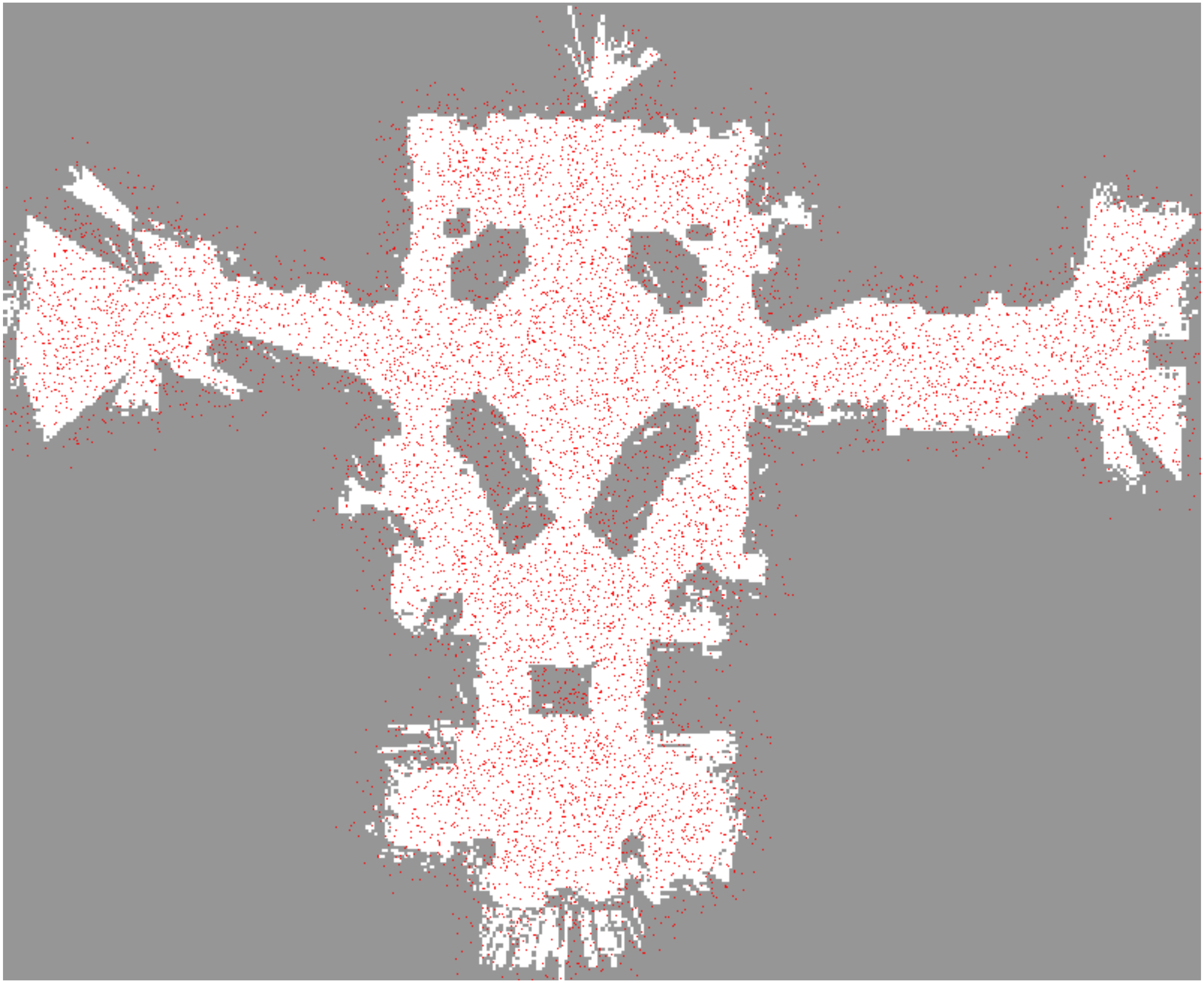
Proximity Sensor Model Reminder

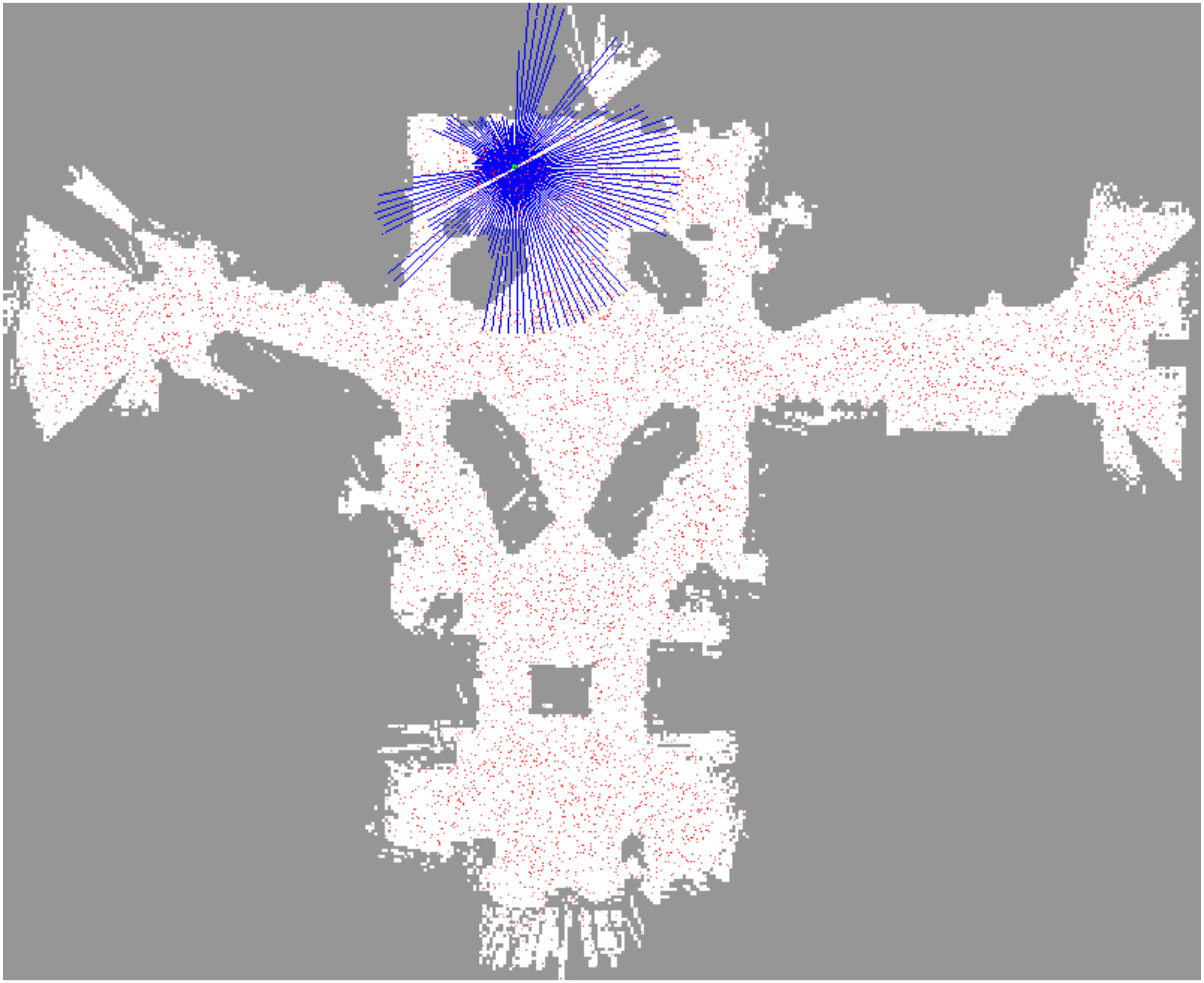


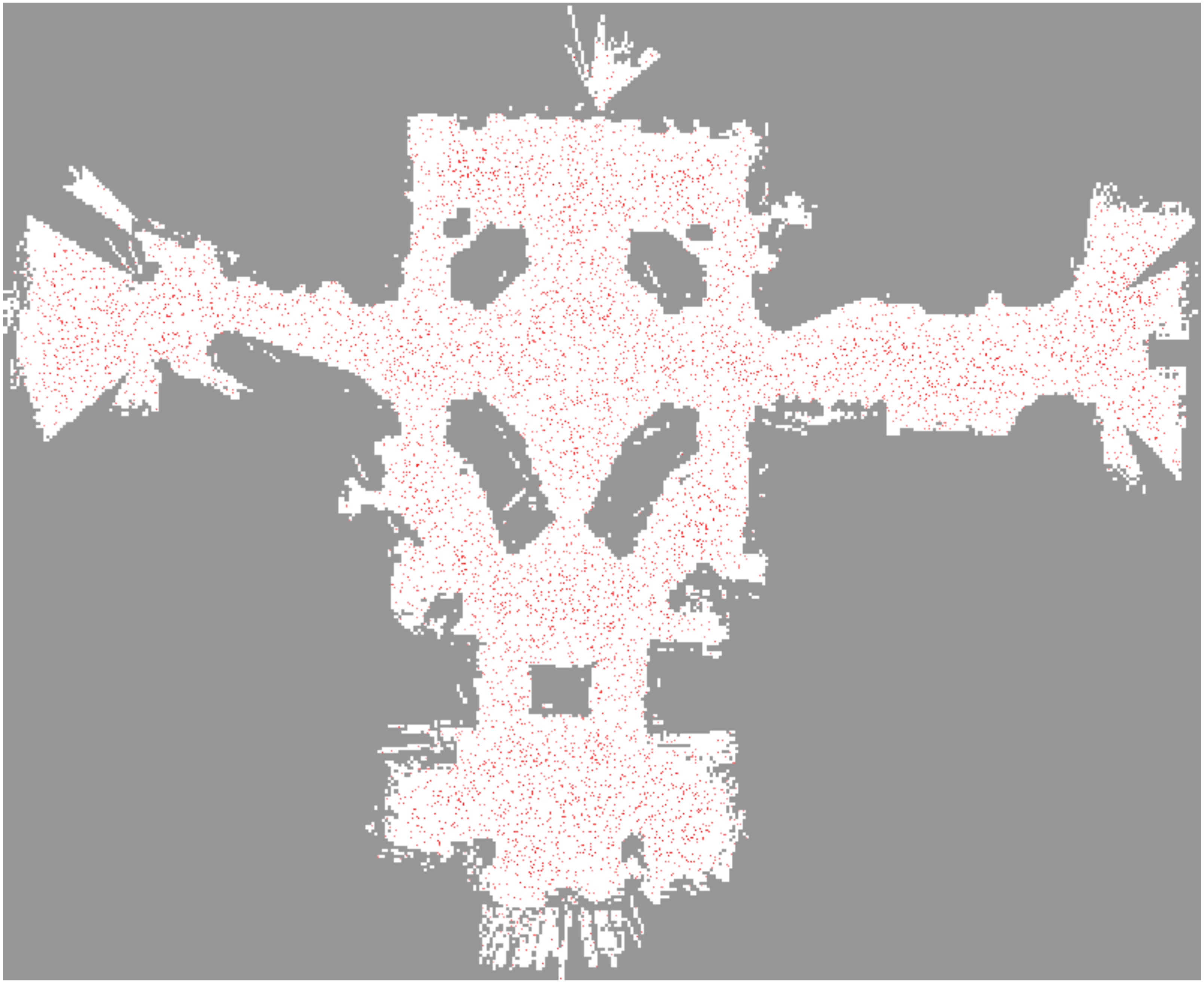
Laser sensor

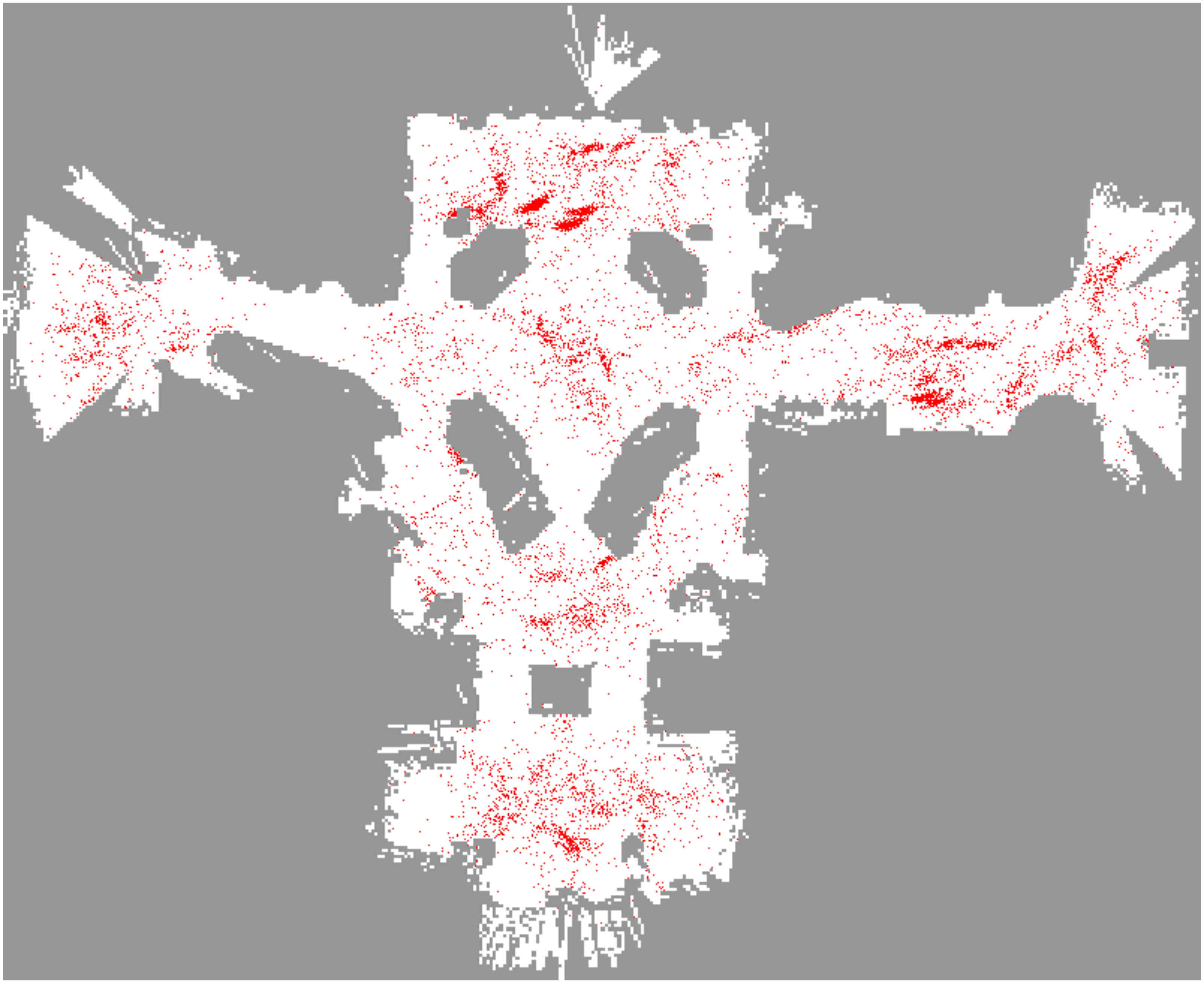


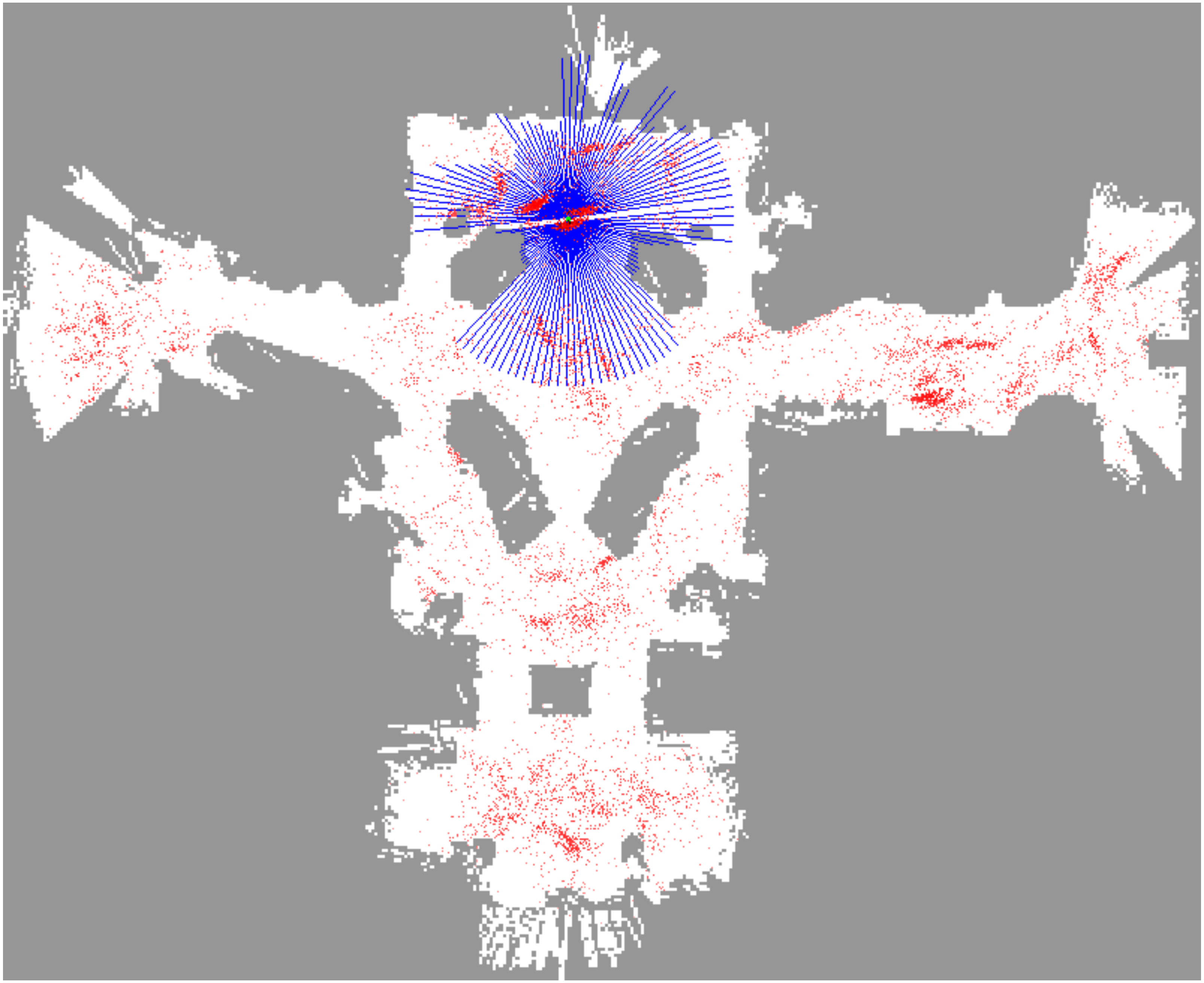
Sonar sensor

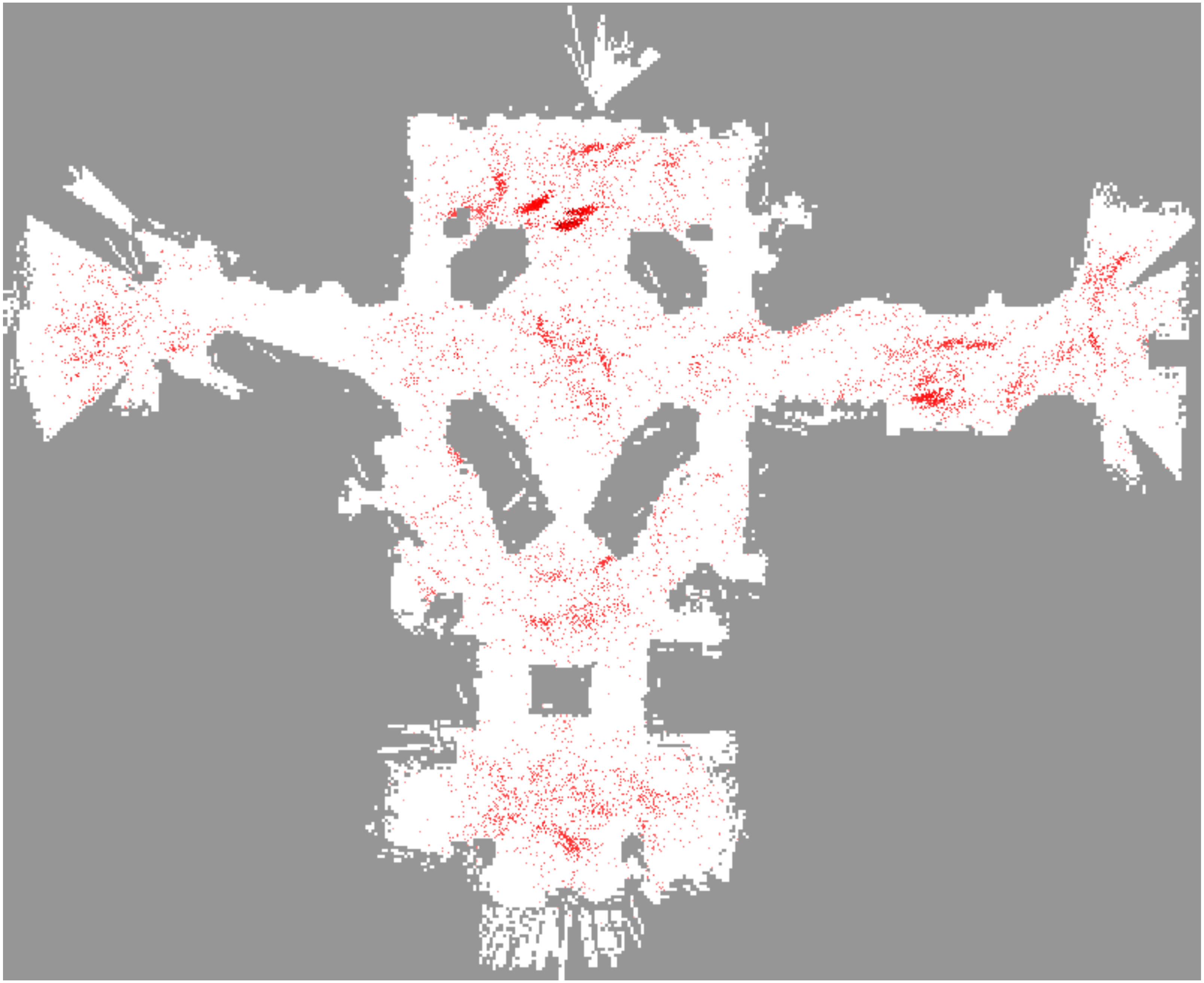


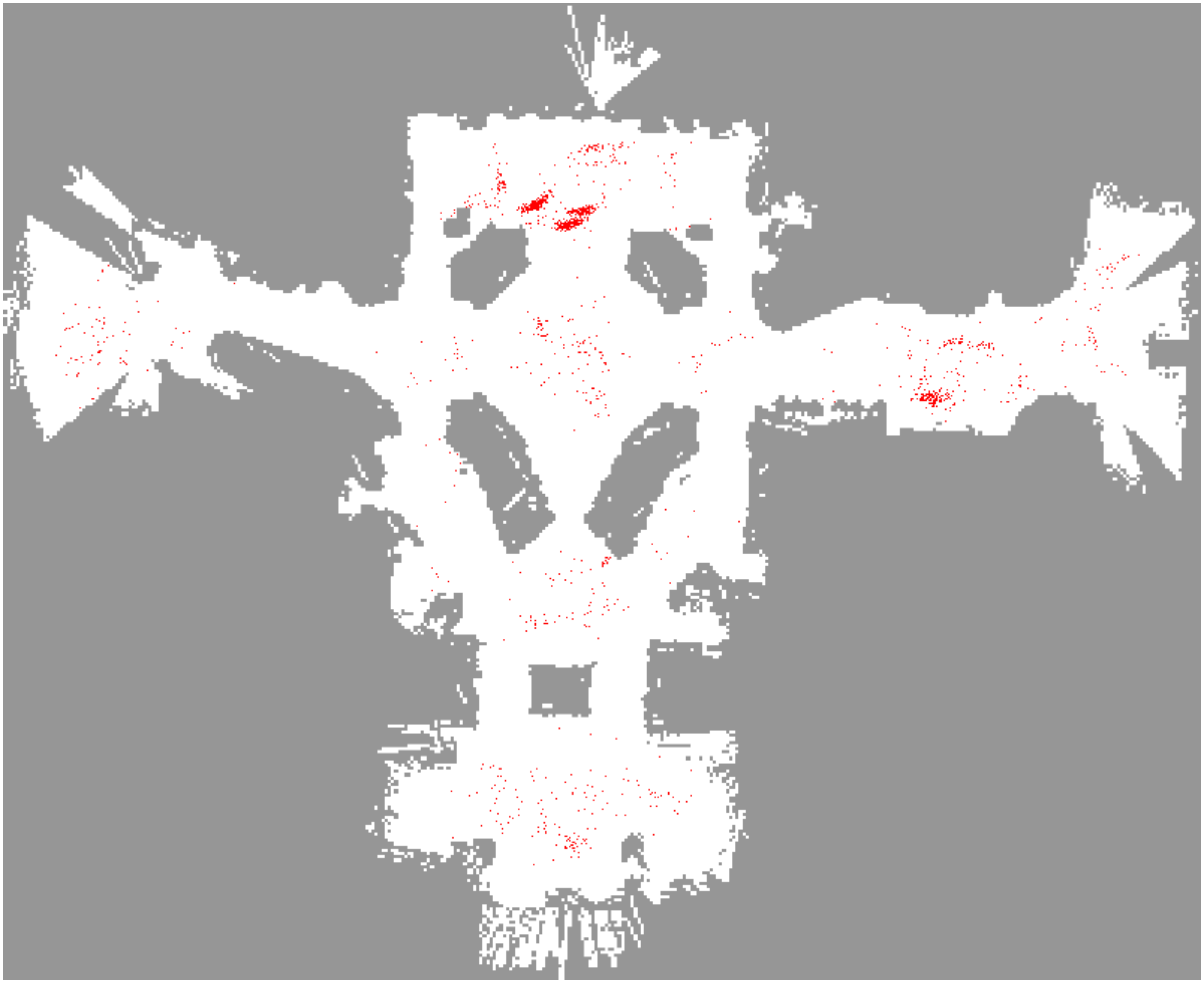


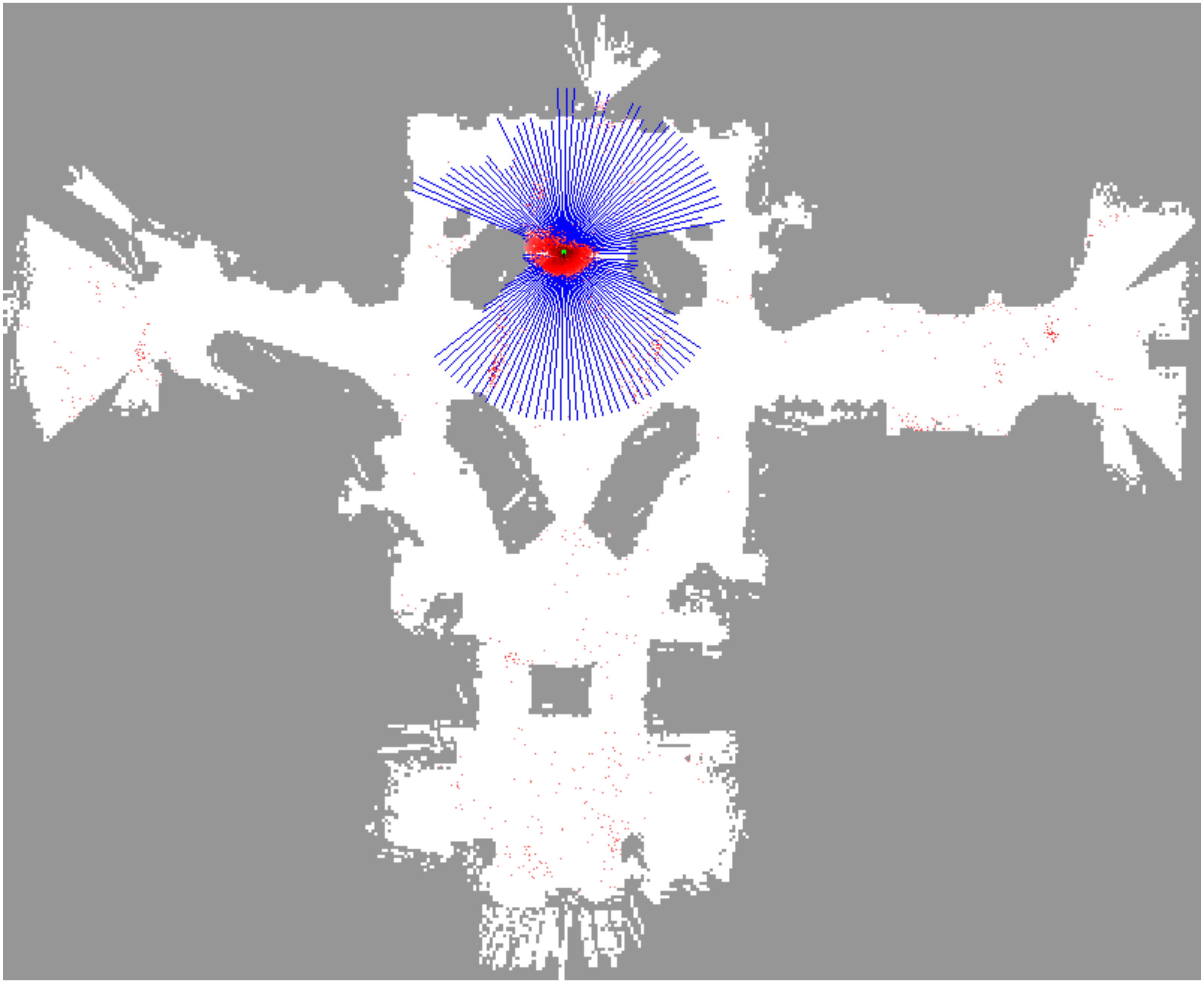


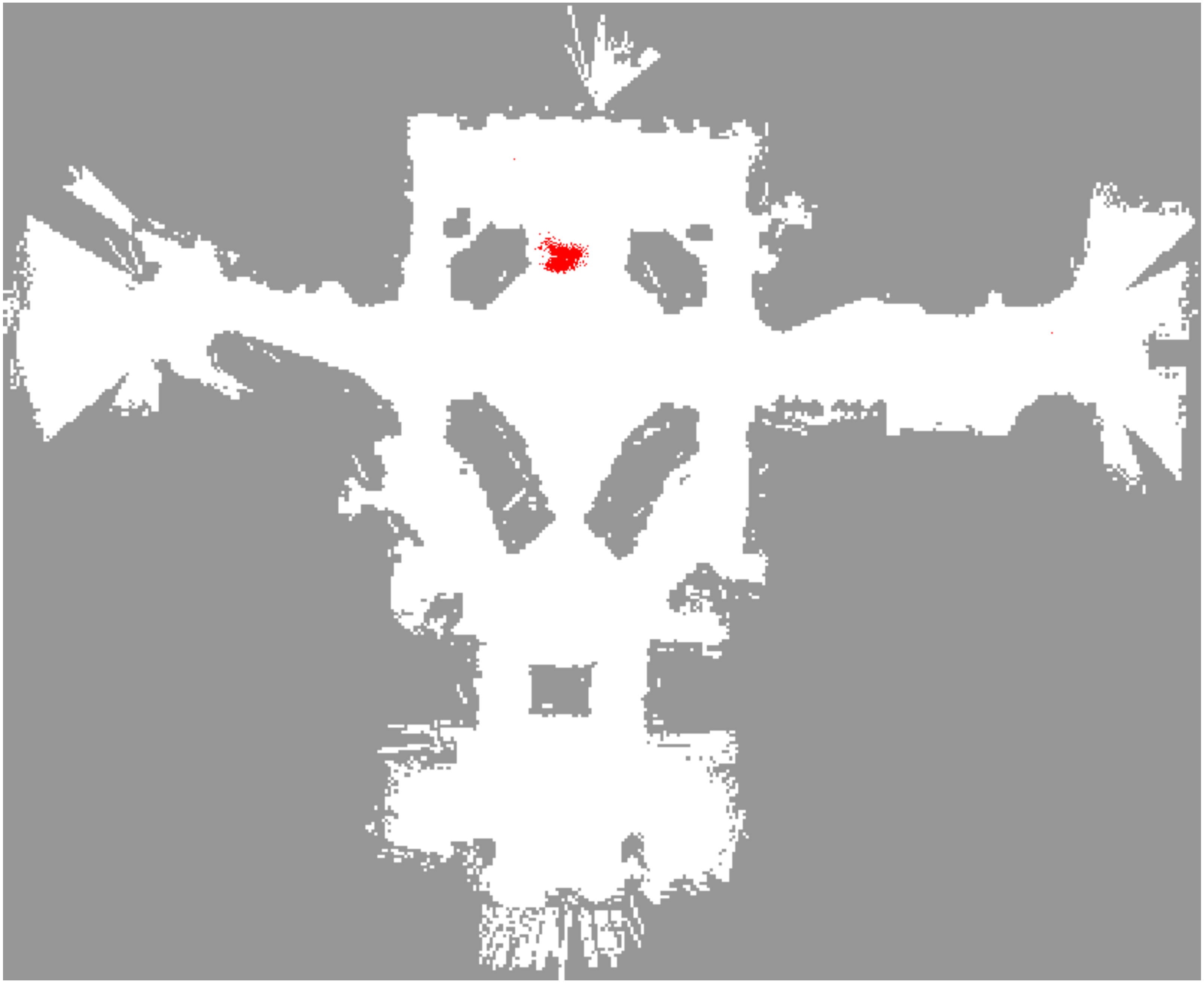


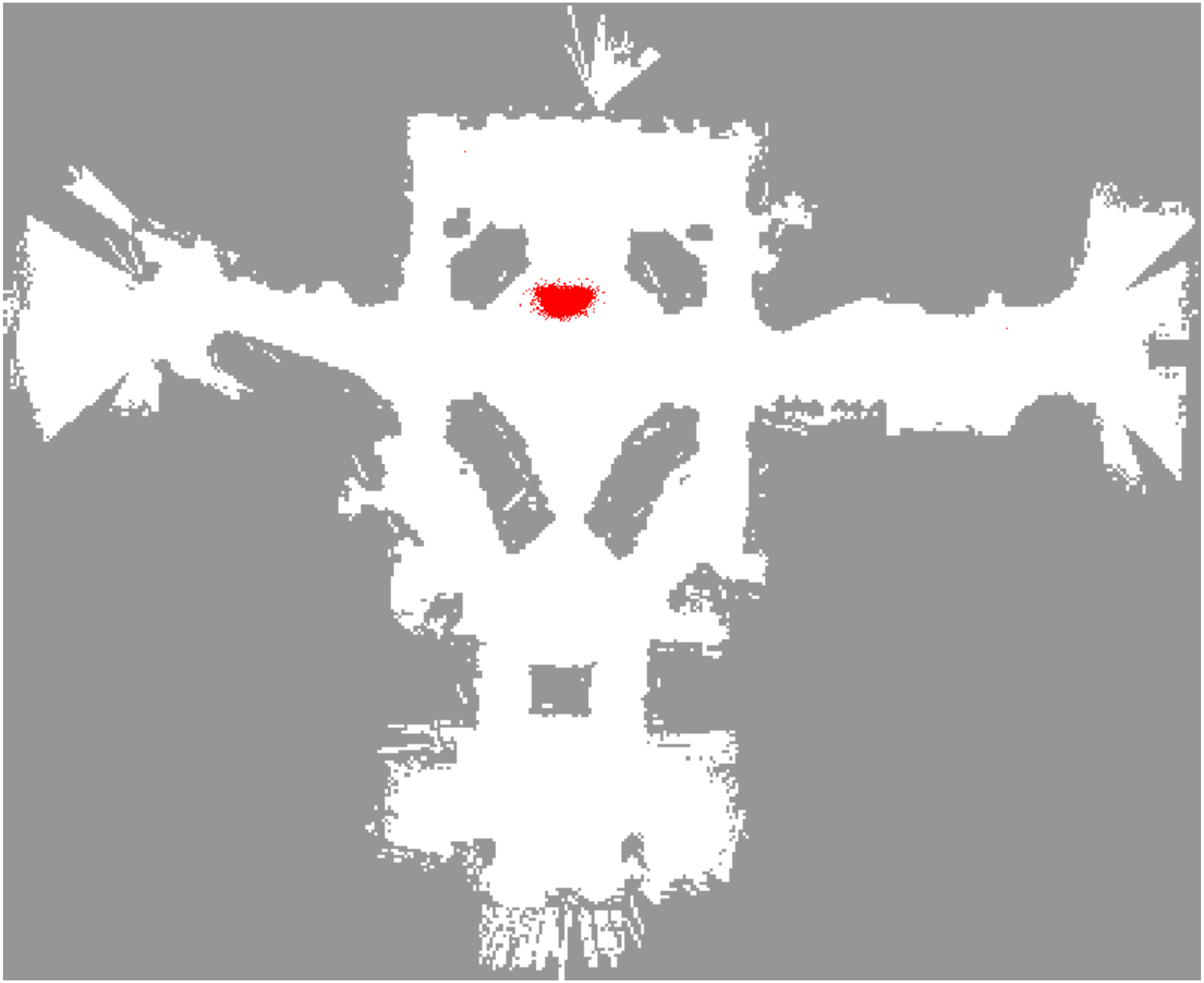


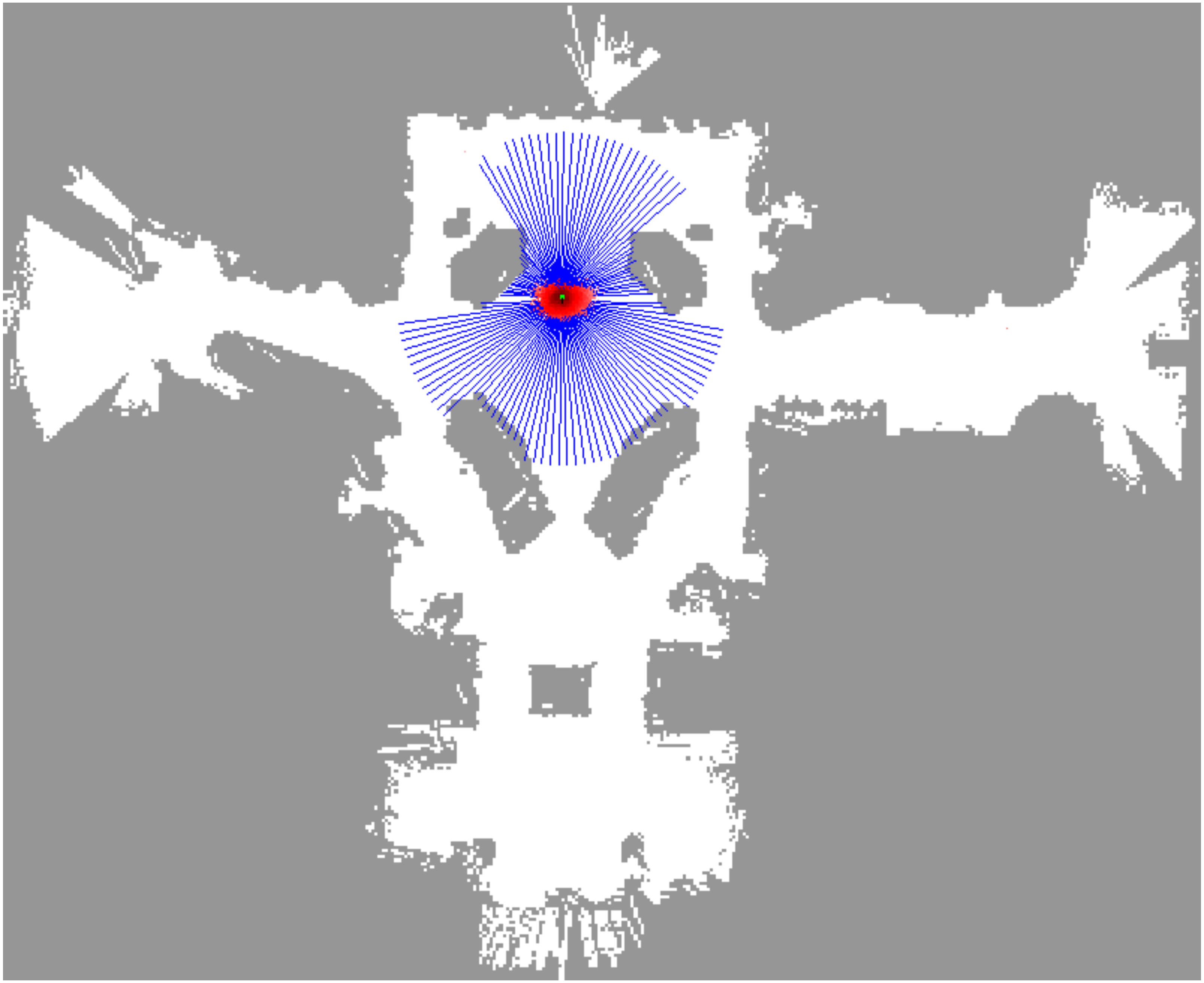


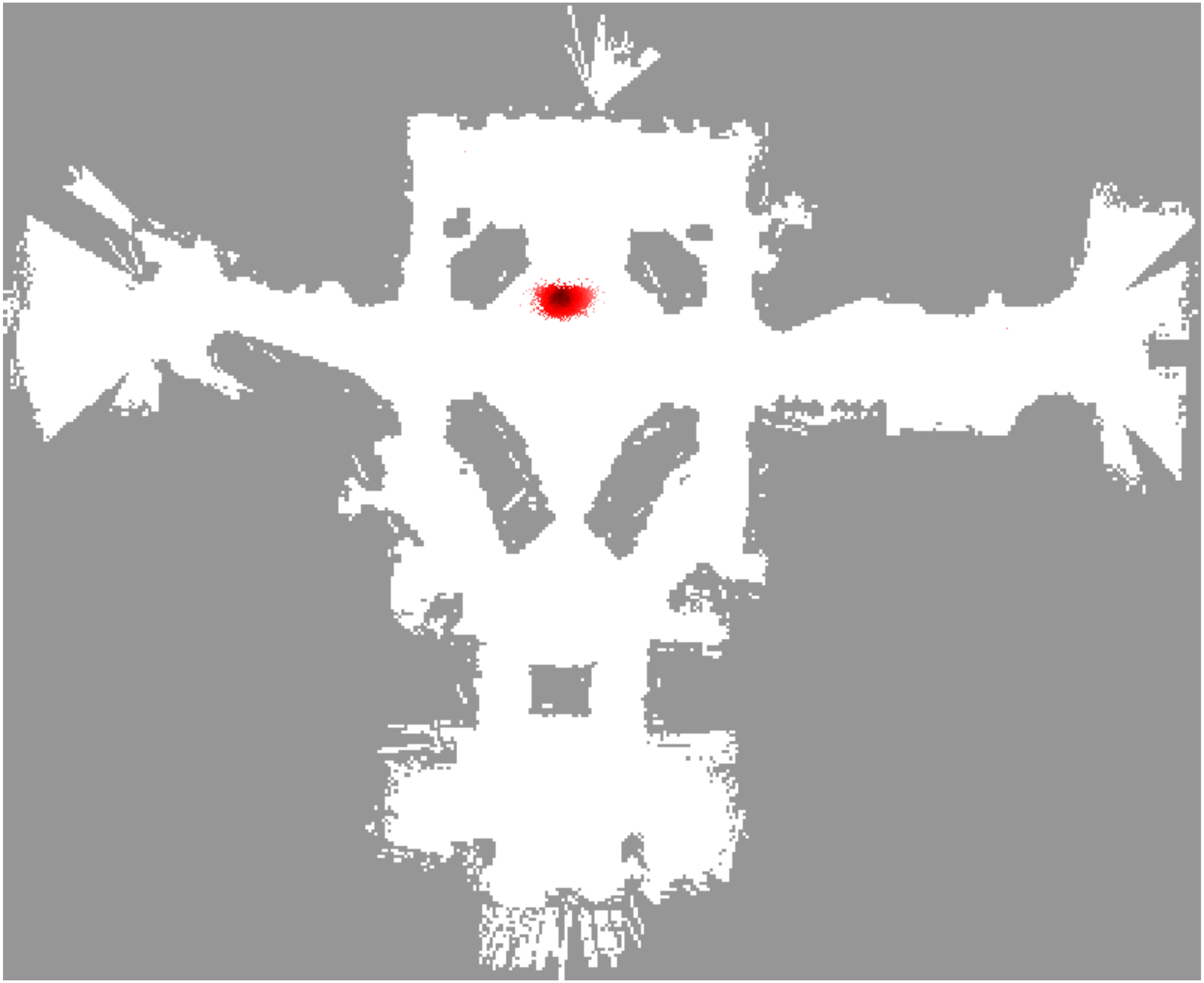


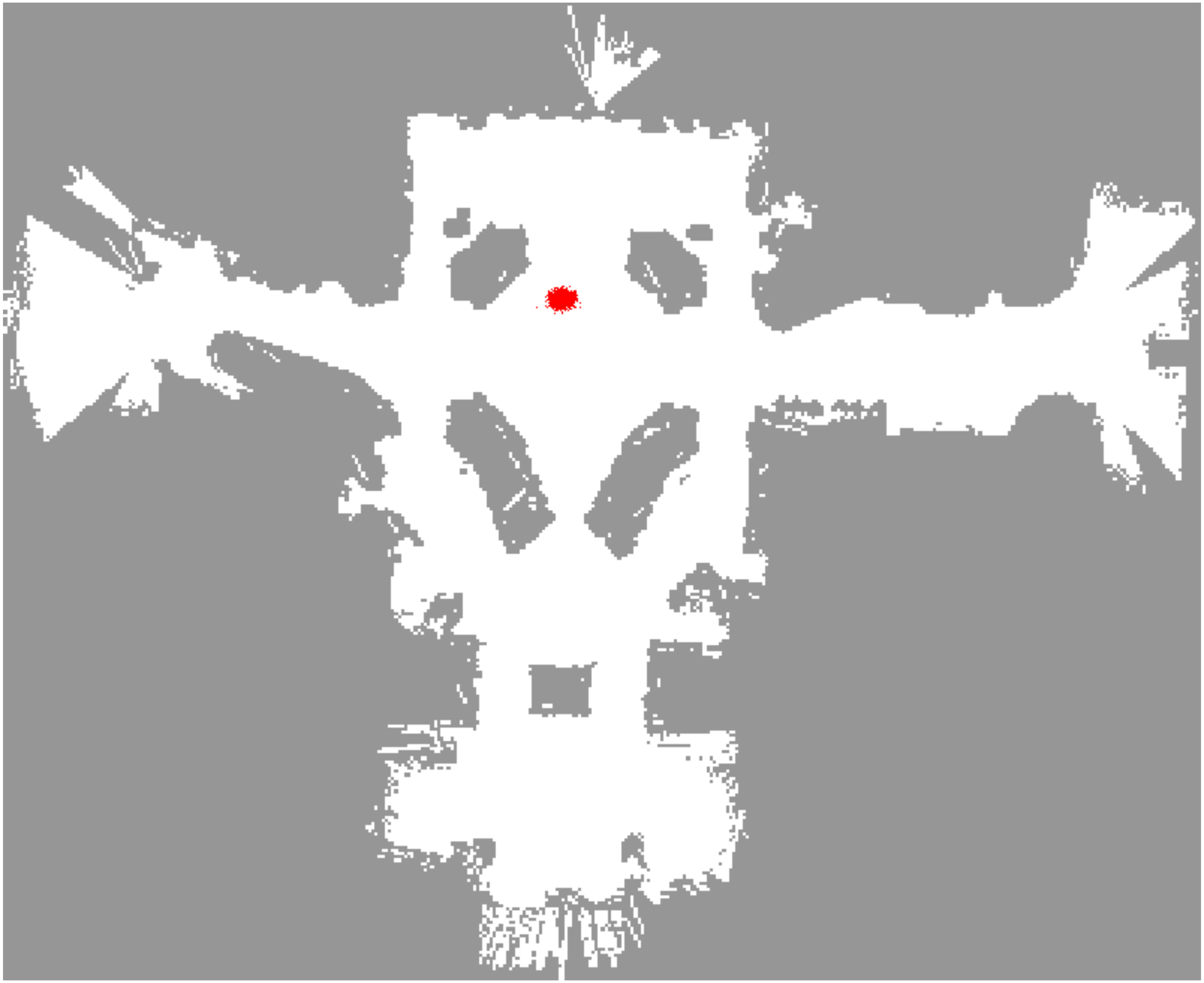


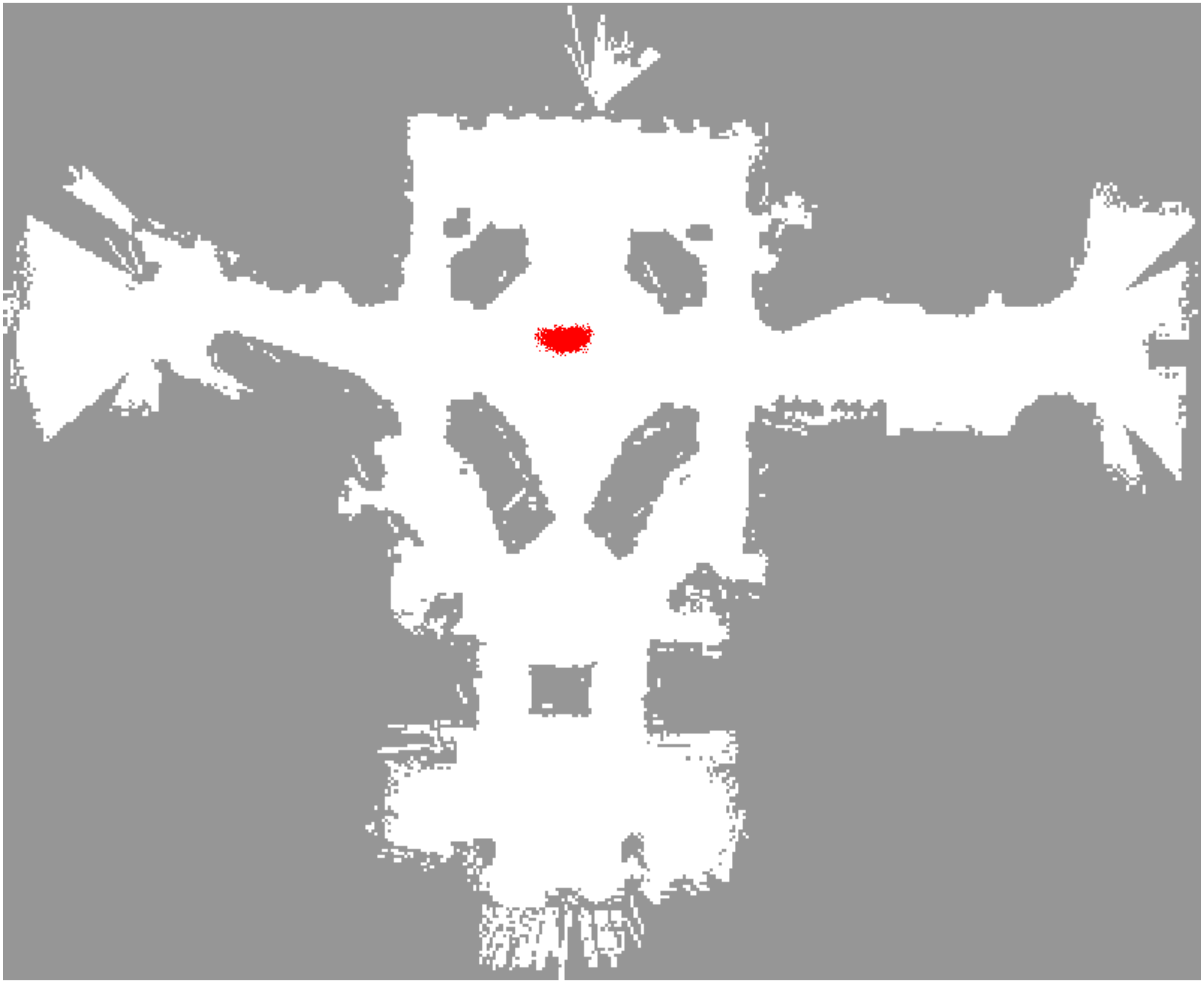


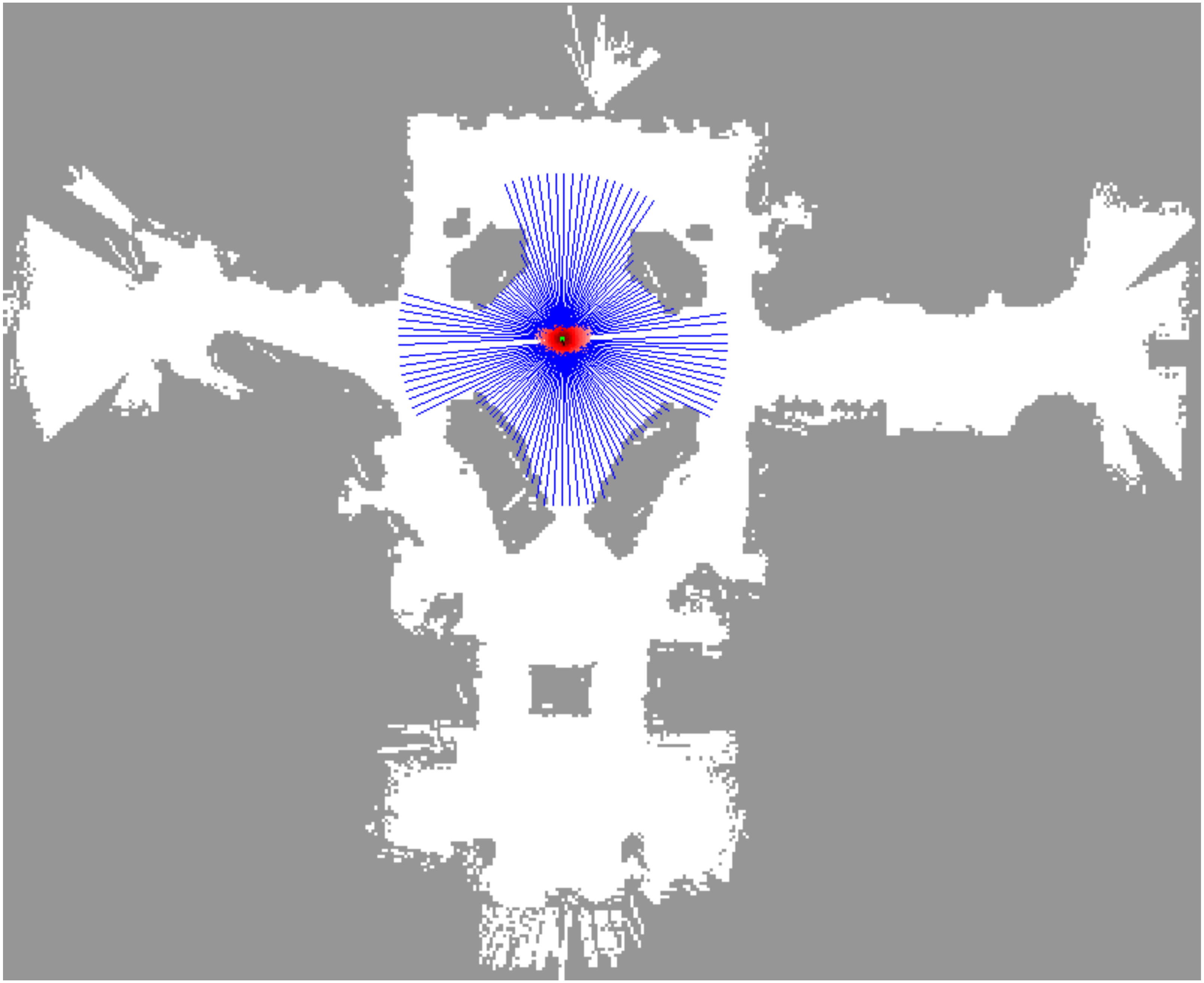


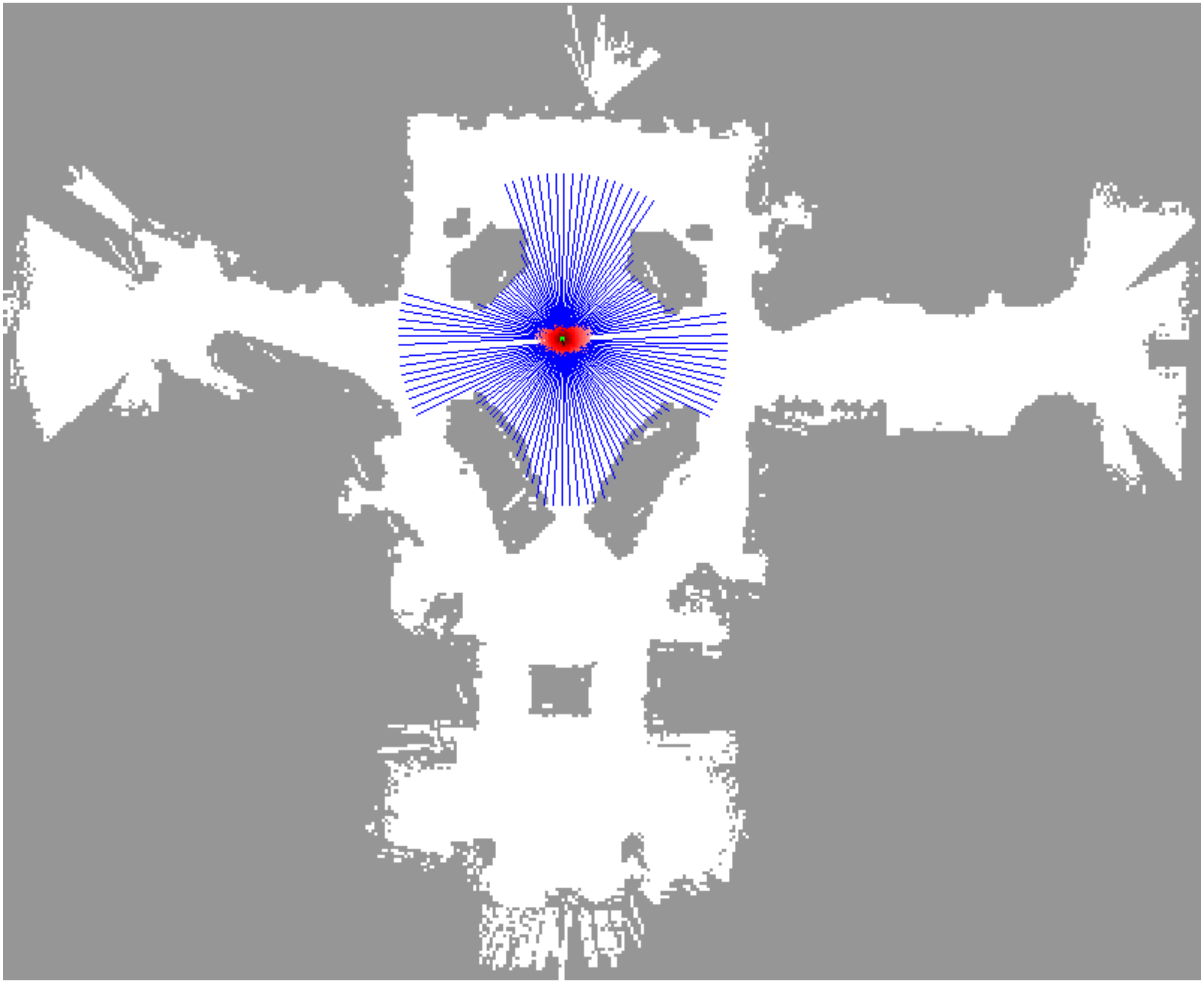




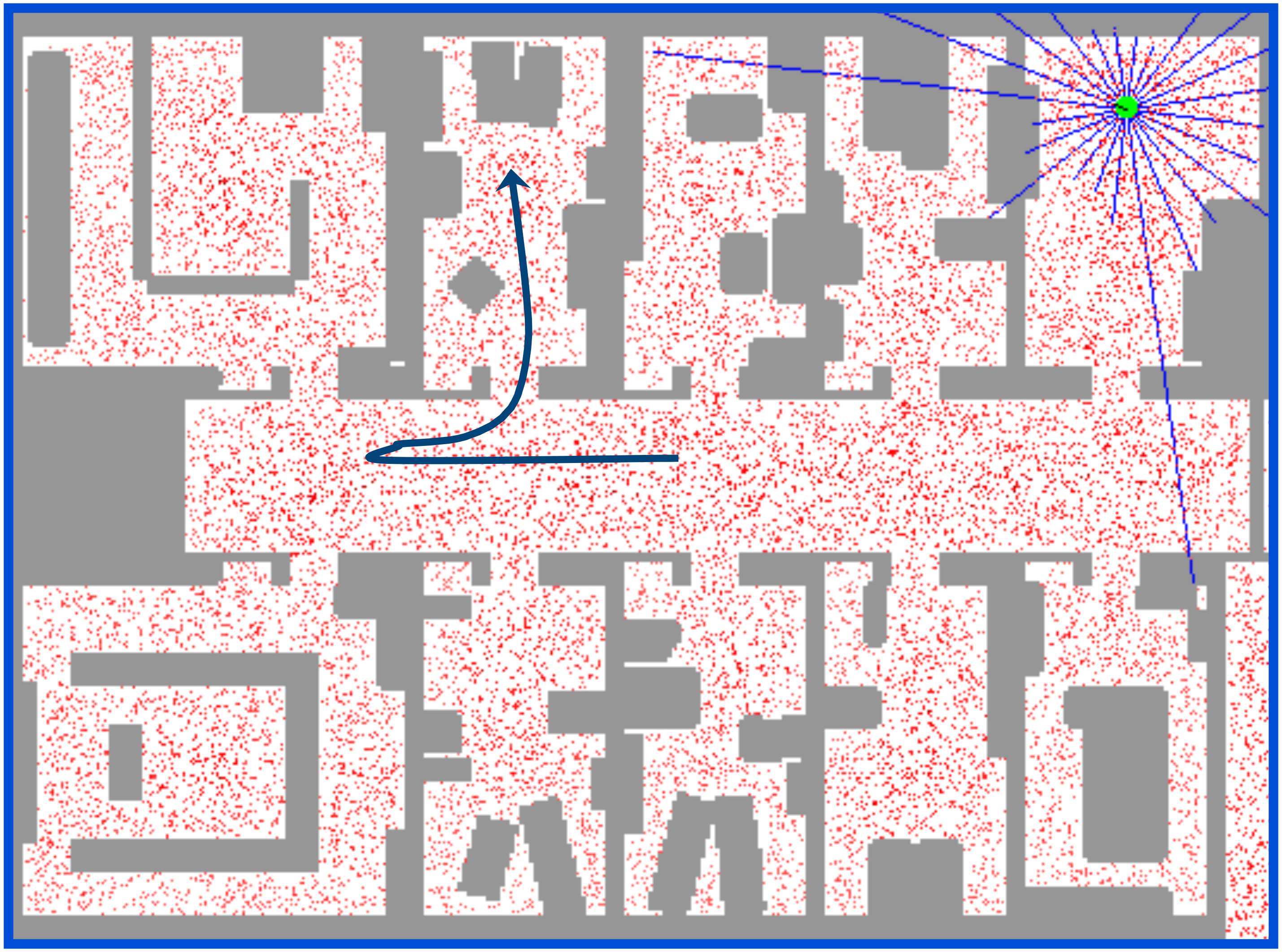




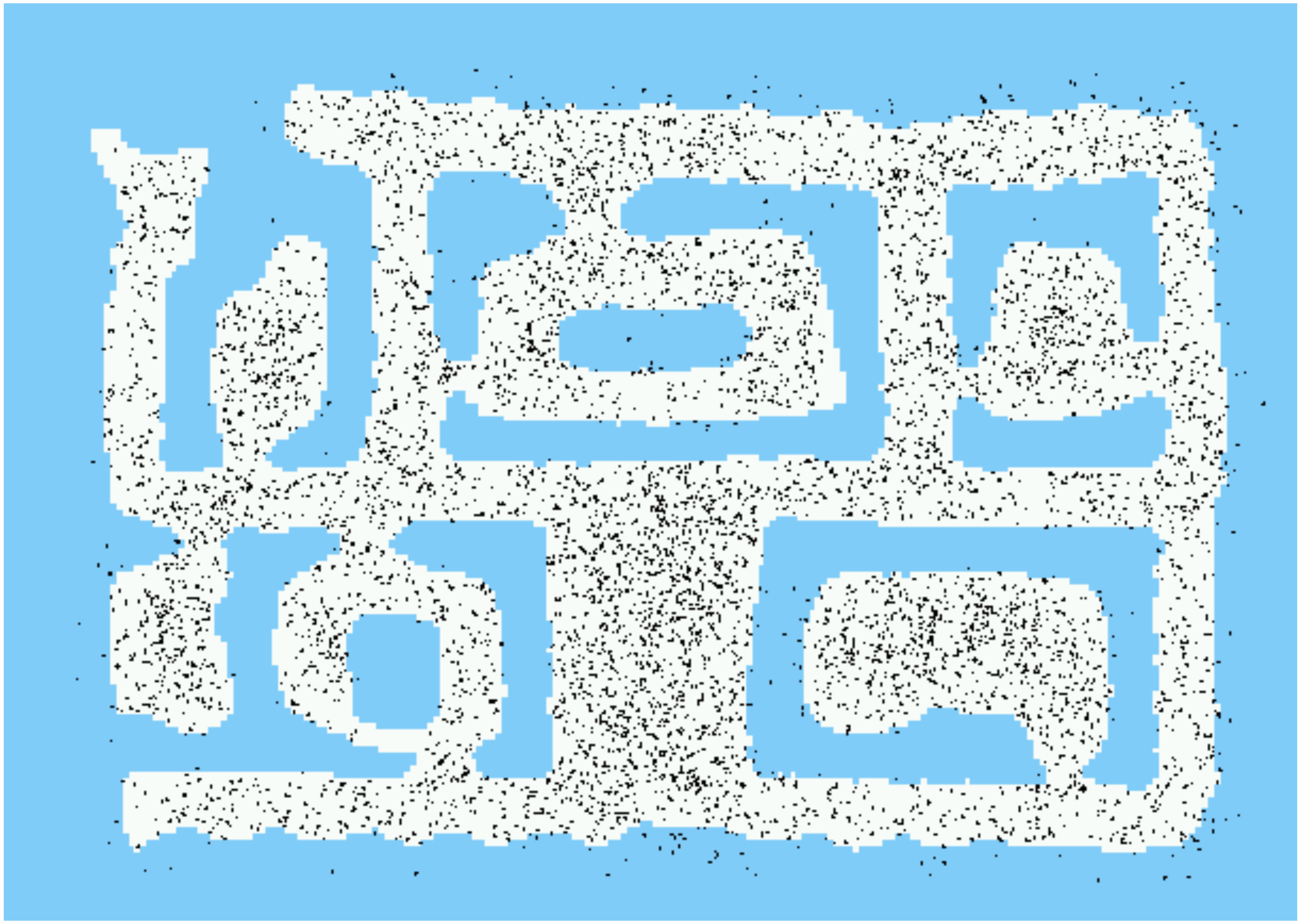




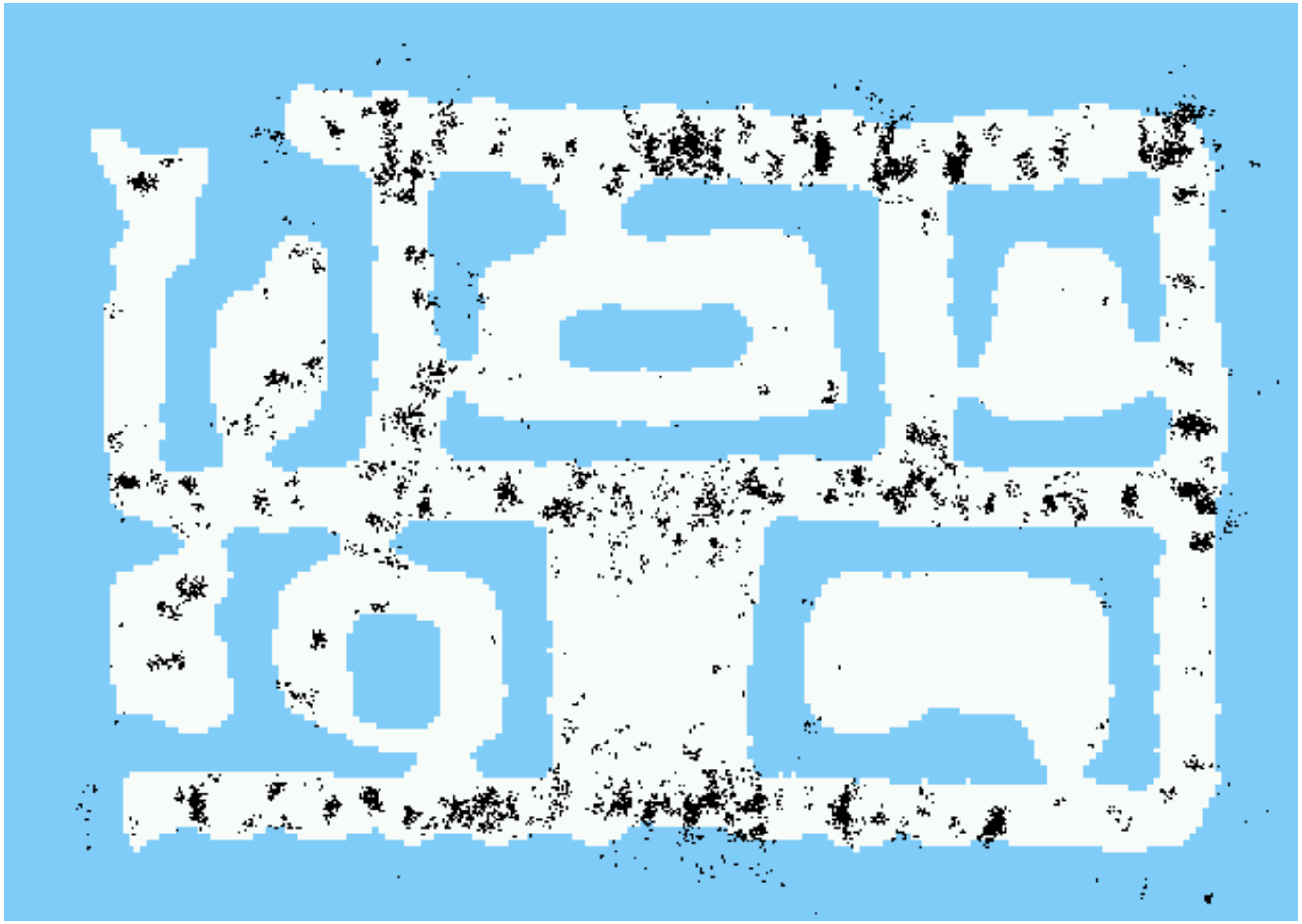
Sample-based Localization (sonar)



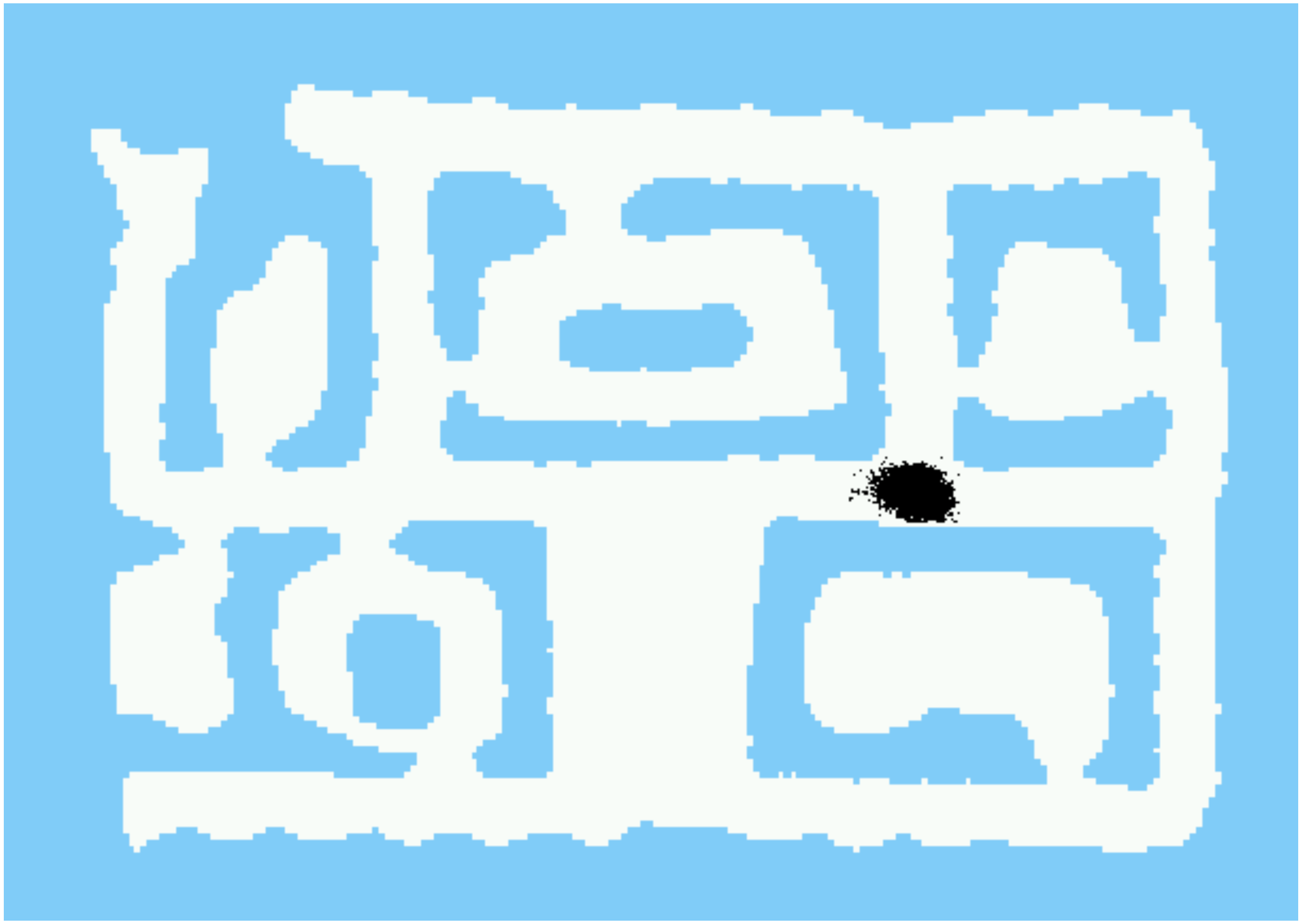
Initial Distribution



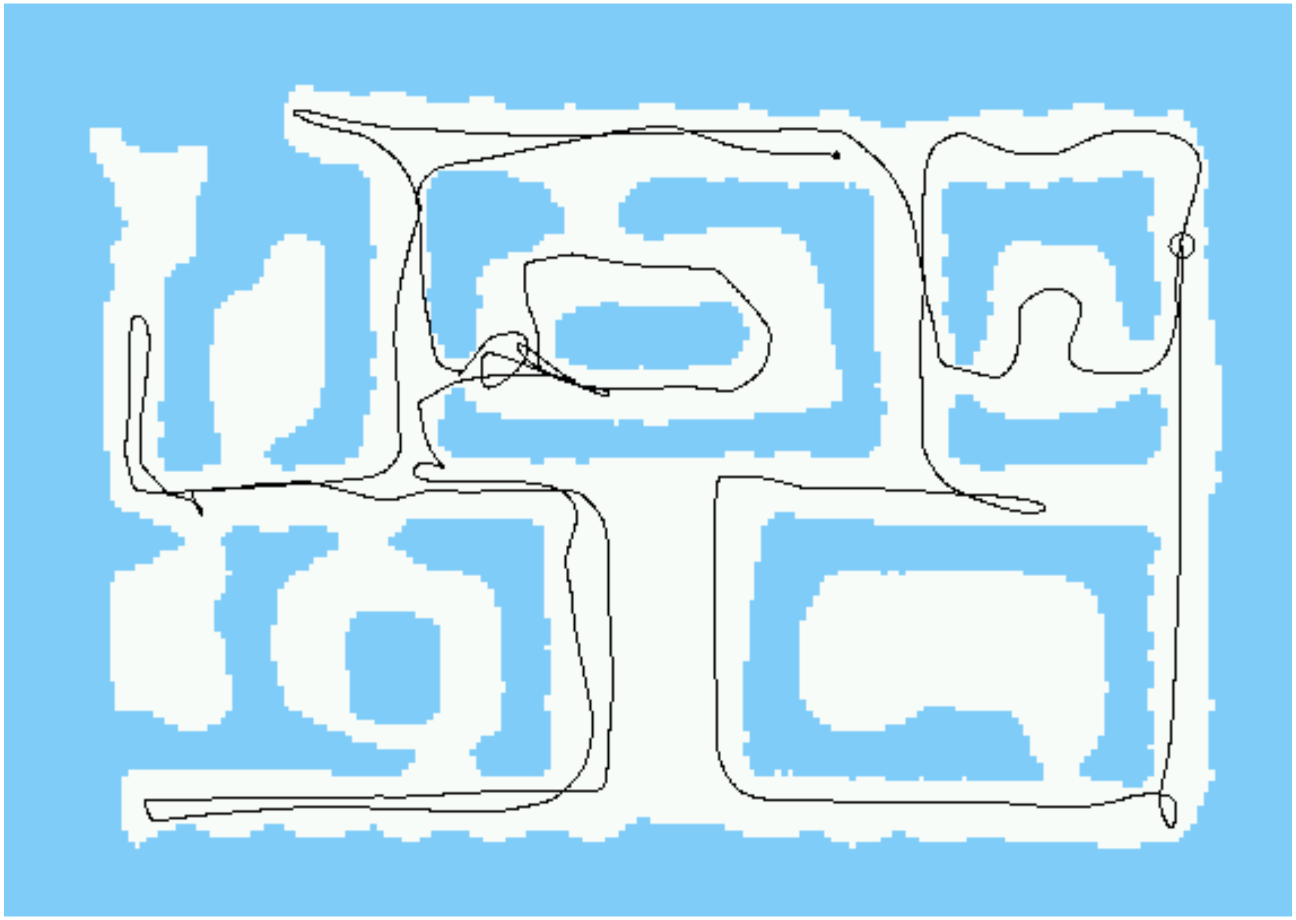
After Incorporating Ten Ultrasound Scans



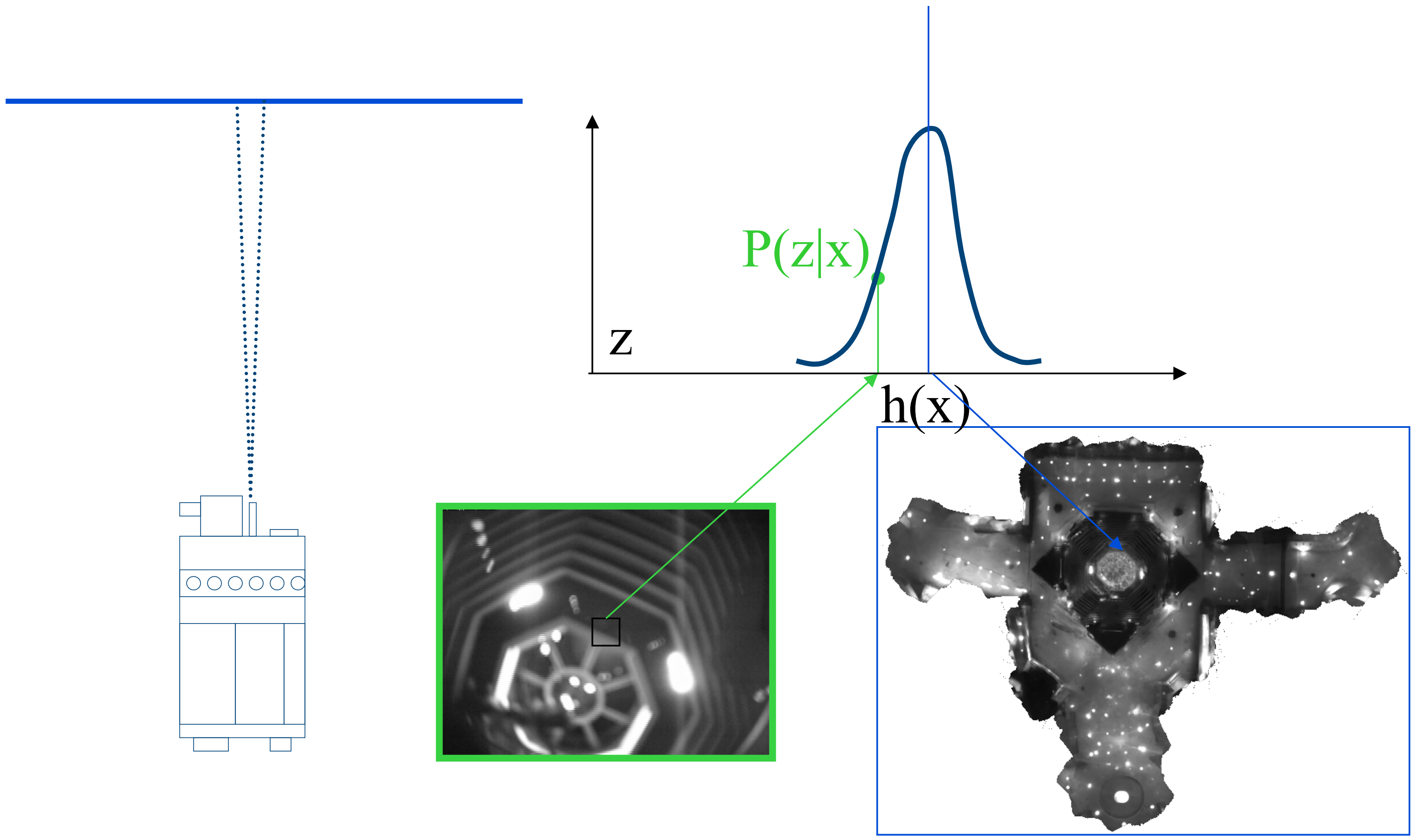
After Incorporating 65 Ultrasound Scans



Estimated Path

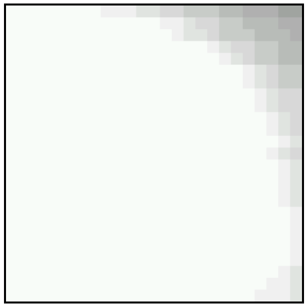


Vision-based Localization

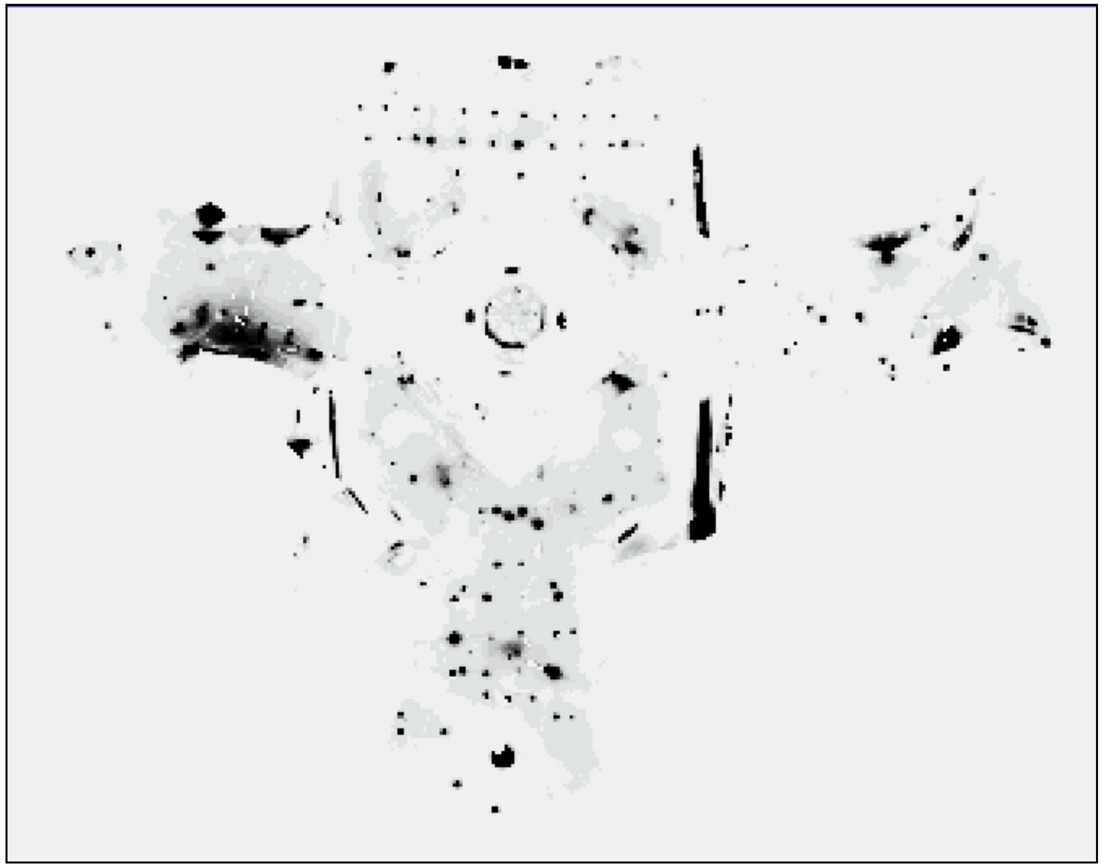


Under a Light

Measurement z:

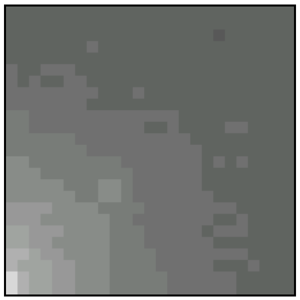


***P(z|x)*:**

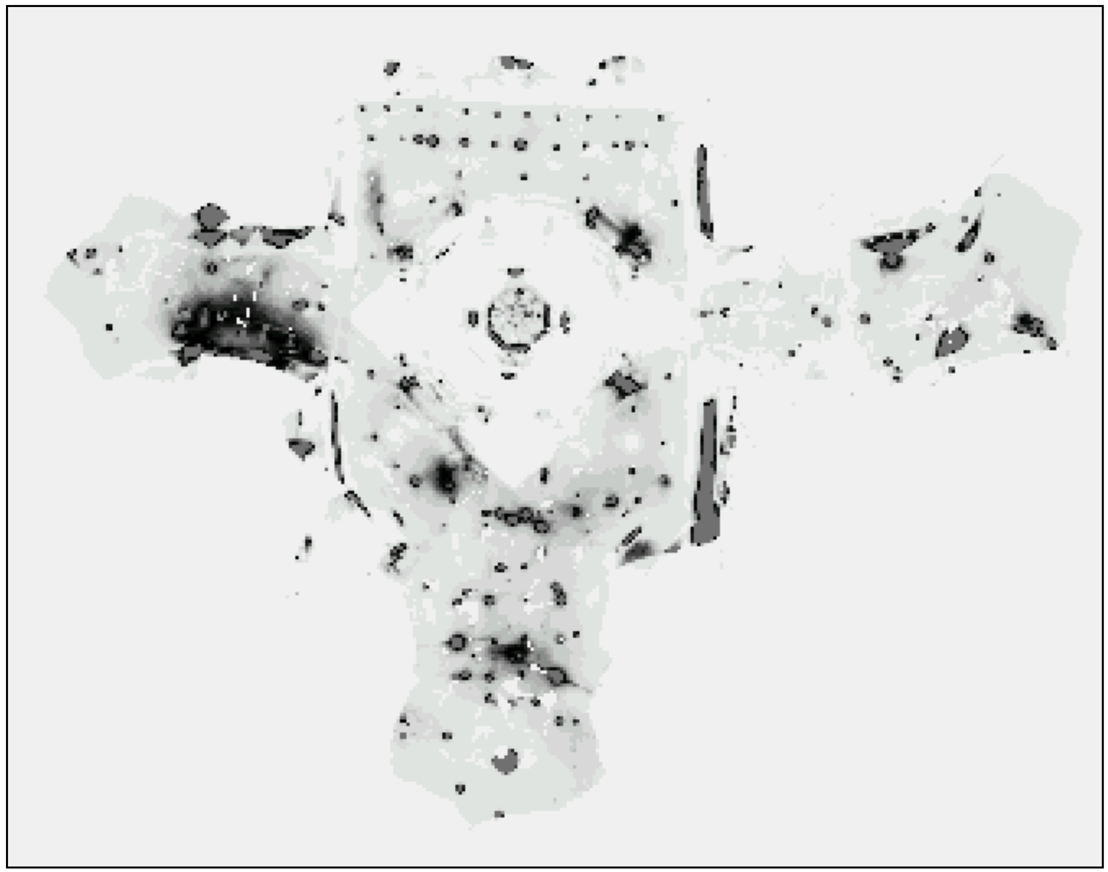


Next to a Light

Measurement z:



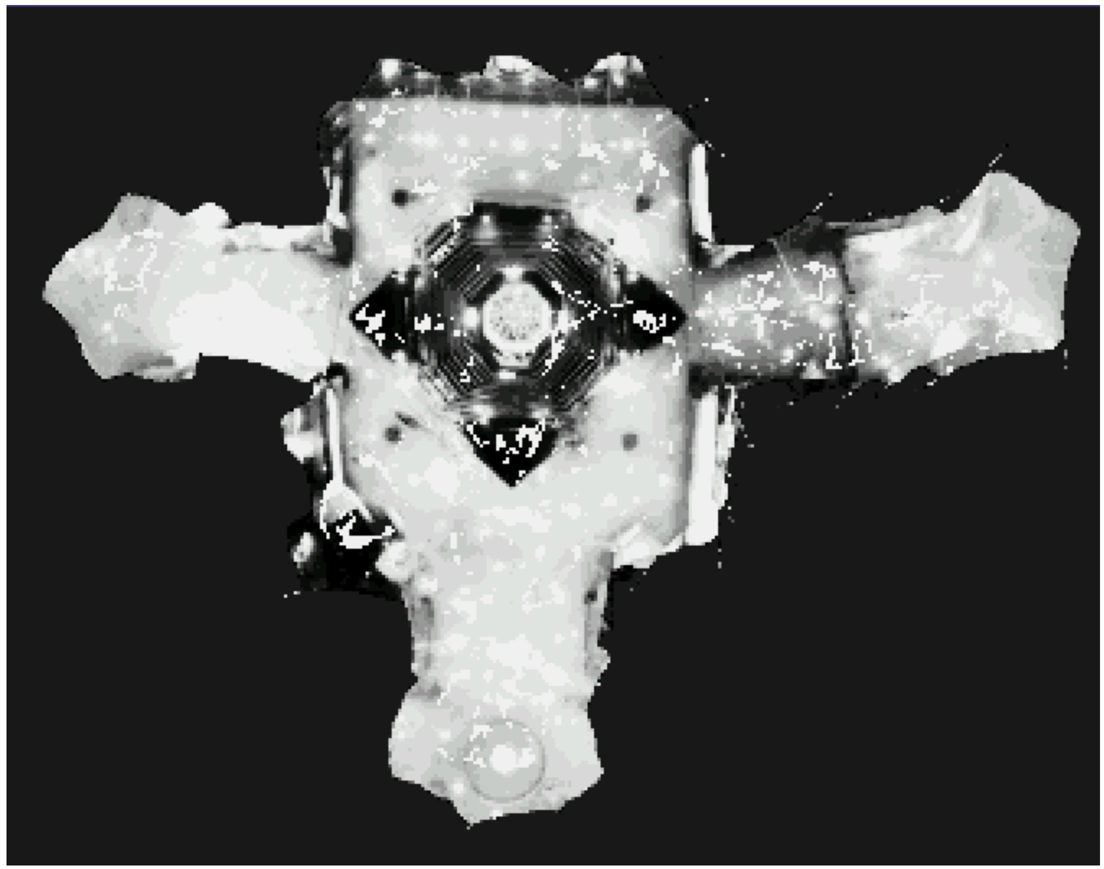
***P(z|x)*:**



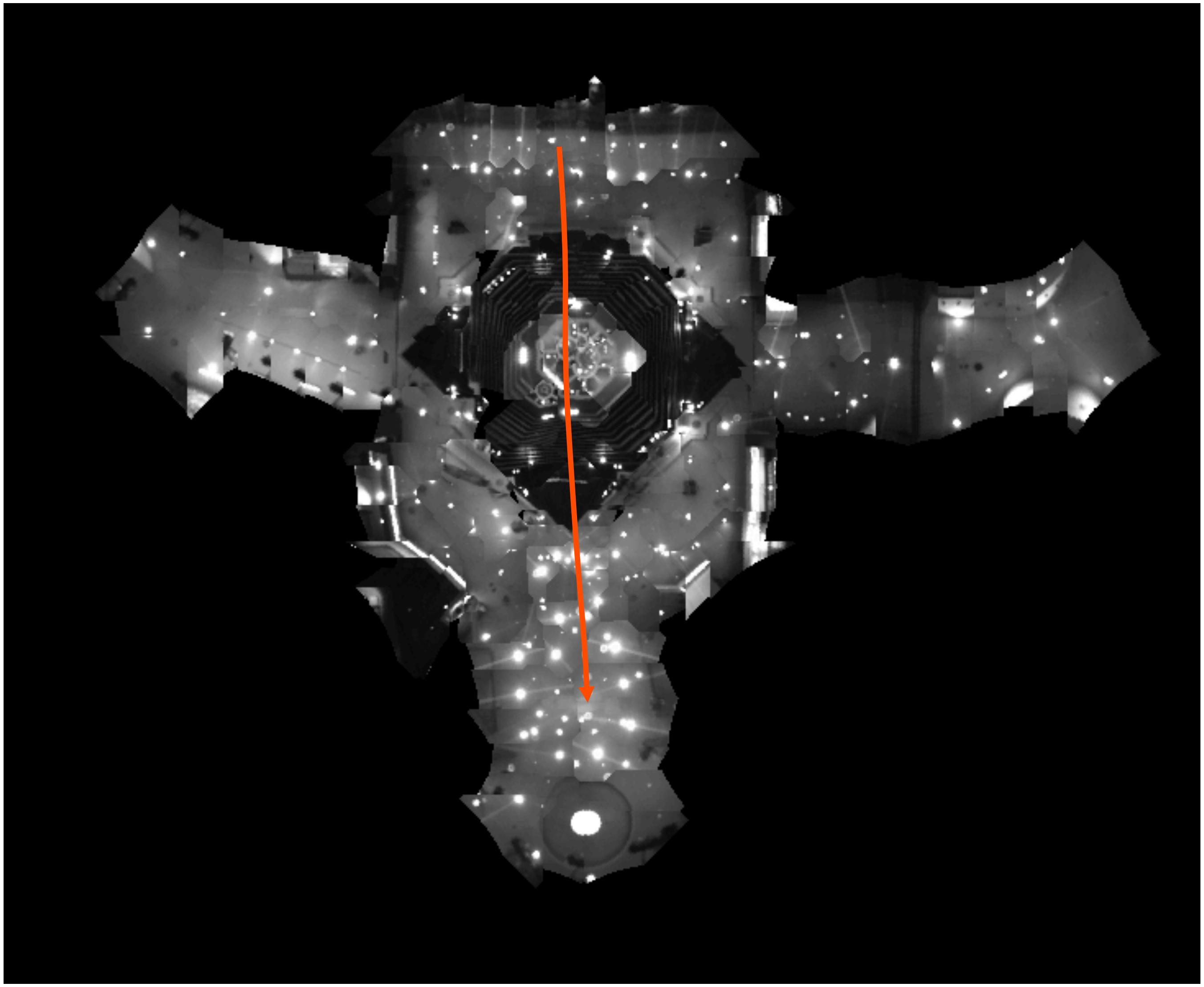
Elsewhere

Measurement z:

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



Global Localization Using Vision



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).

Summary – Particle Filters

- Particle filters are an implementation of recursive Bayesian filtering
- They represent the posterior by a set of weighted samples
- They can model non-Gaussian distributions
- Proposal to draw new samples
- Weight to account for the differences between the proposal and the target
- Monte Carlo filter, Survival of the fittest, Condensation, Bootstrap filter

Summary – PF Localization

- 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.