CSE-571
Probabilistic Robotics

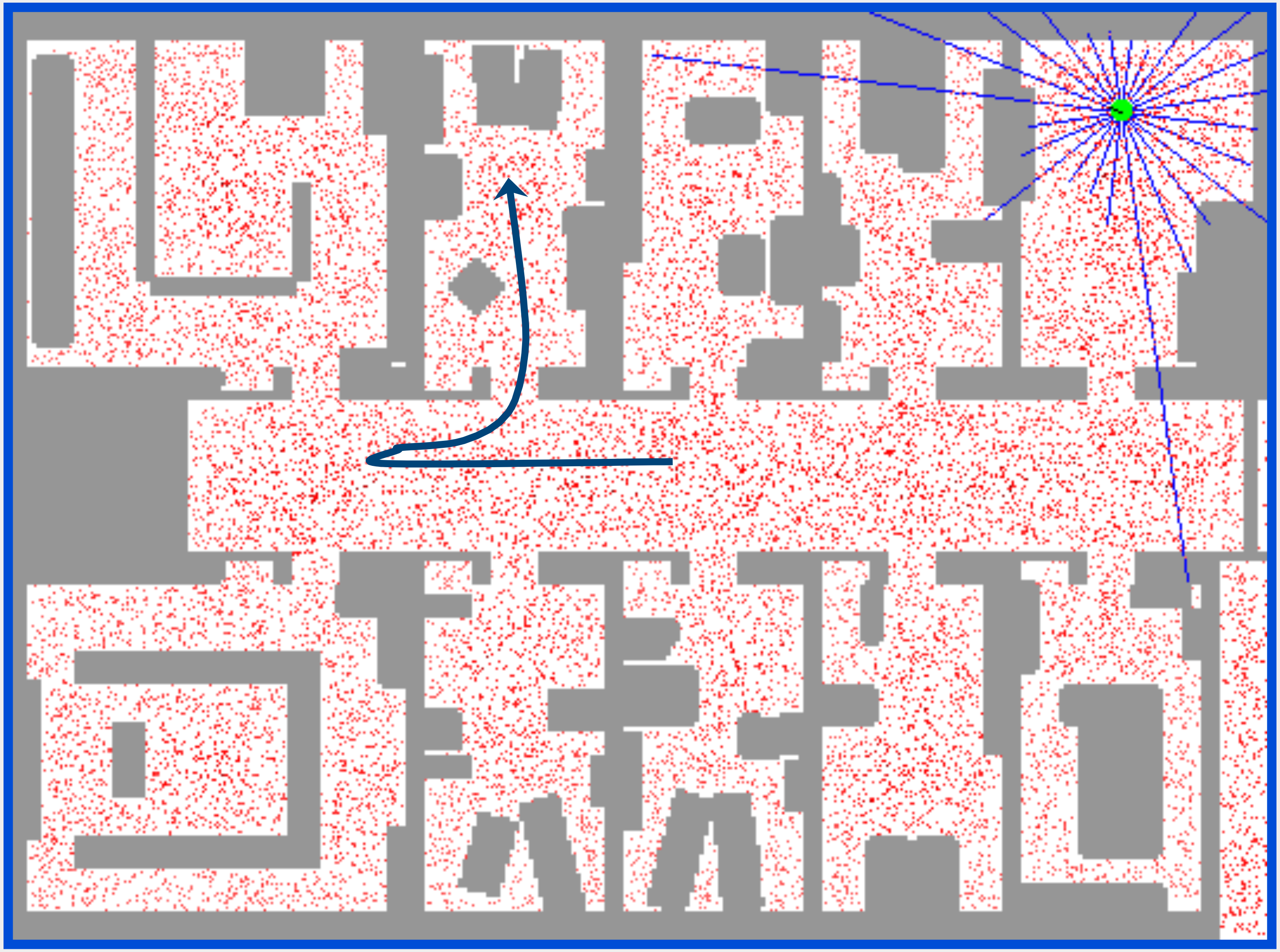
Bayes Filter Implementations

Particle filters

Motivation

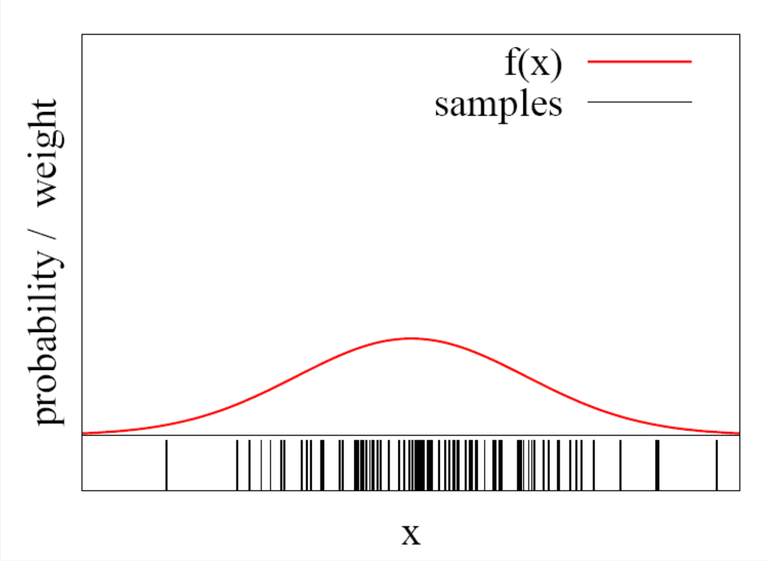
- So far, we discussed the
  - Kalman filter: Gaussian, linearization problems
  - Discrete filter: high memory complexity
- Particle filters are a way to **efficiently** represent **non-Gaussian distributions**
- Basic principle
  - Set of state hypotheses ("particles")
  - Survival-of-the-fittest

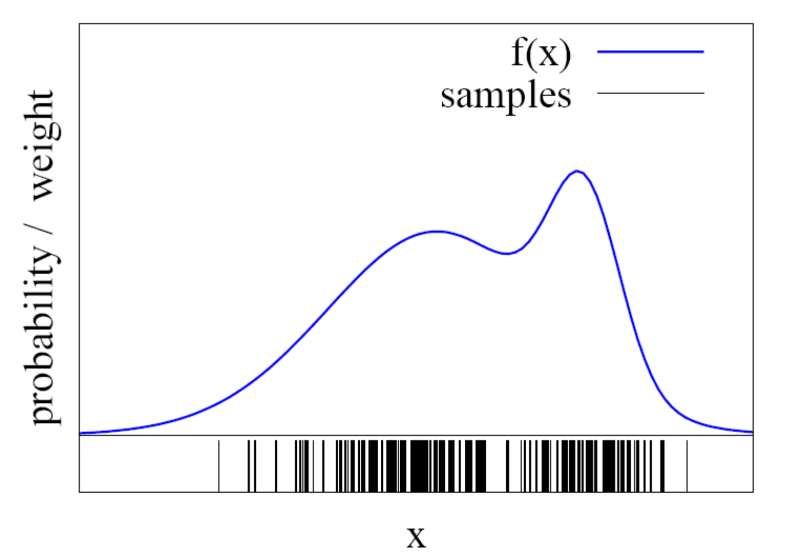
Sample-based Localization (sonar)



Function Approximation

- Particle sets can be used to approximate functions



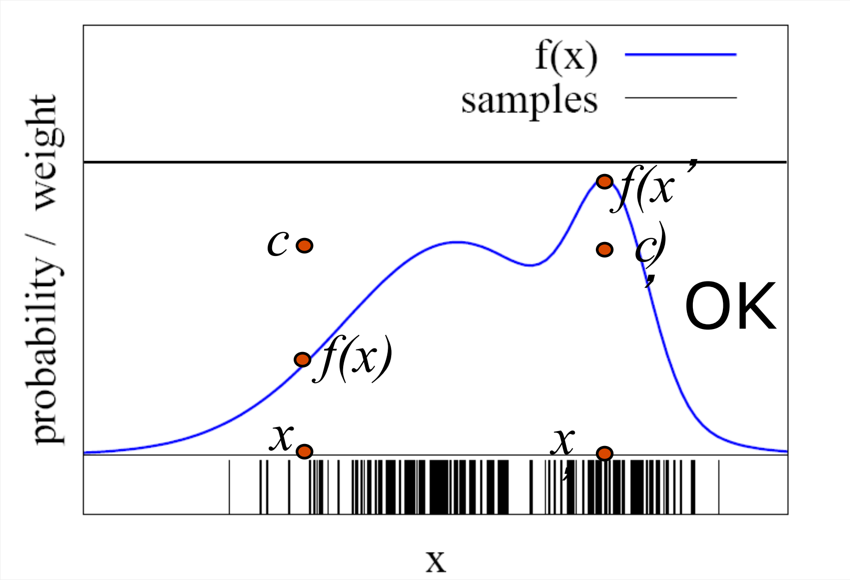


- The more particles fall into an interval, the higher the probability of that interval

- How to draw samples form a function/distribution?

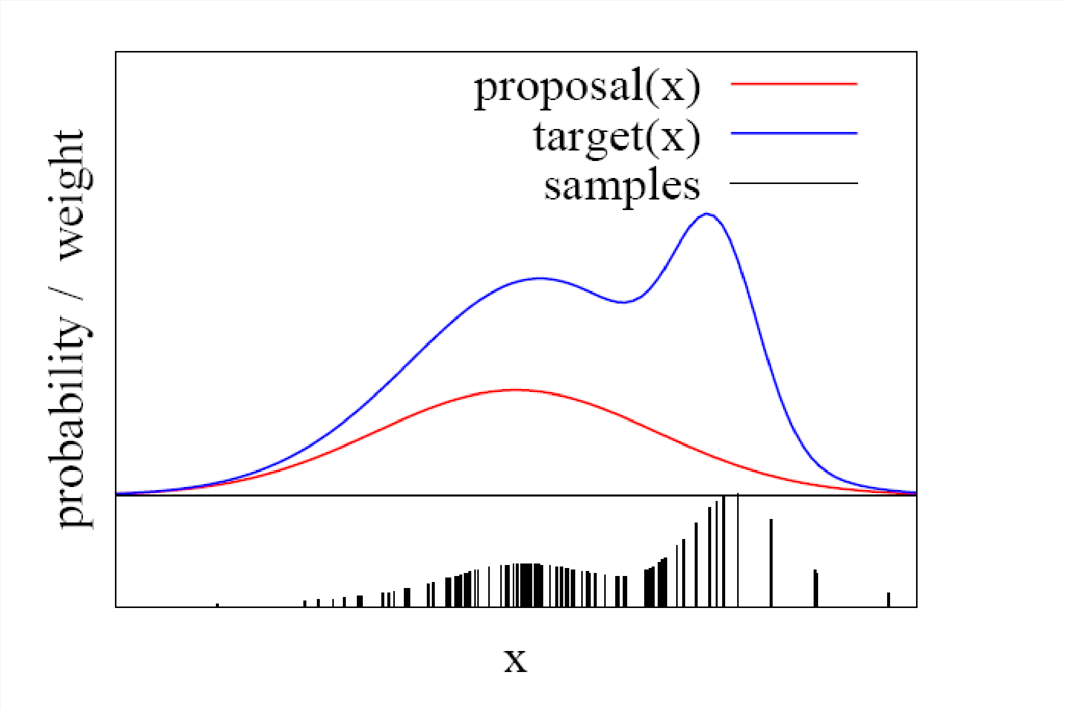
Rejection Sampling

- 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



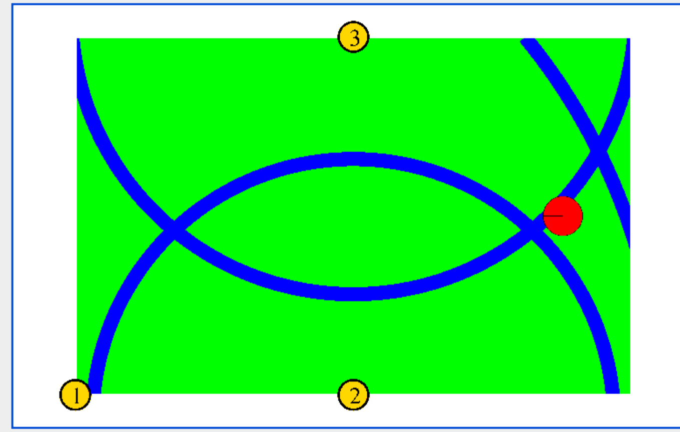
Importance Sampling Principle

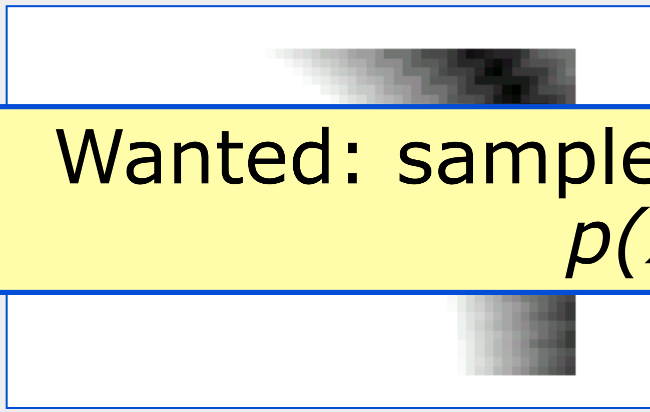
- We can even use a different distribution g to generate samples from f
- By introducing an importance weight w, we can account for the "differences between g and f"
- $w = f / g$
- f is often called target
- g is often called proposal

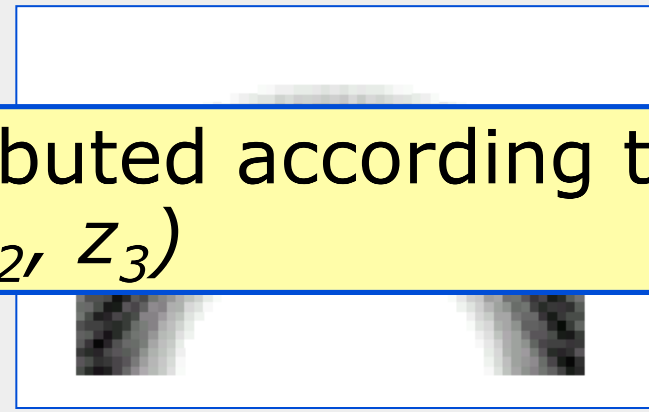


Importance Sampling with Resampling: Landmark Detection Example

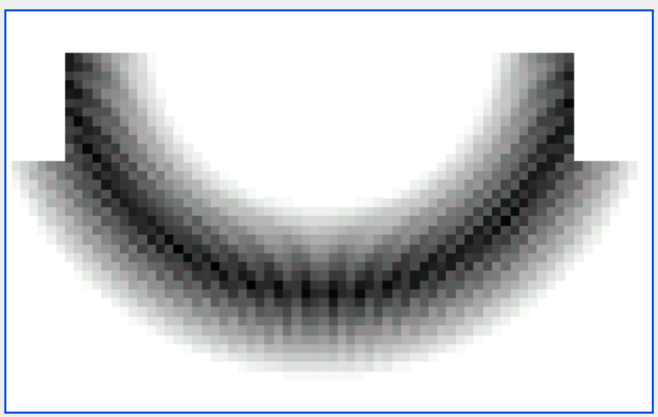
Distributions





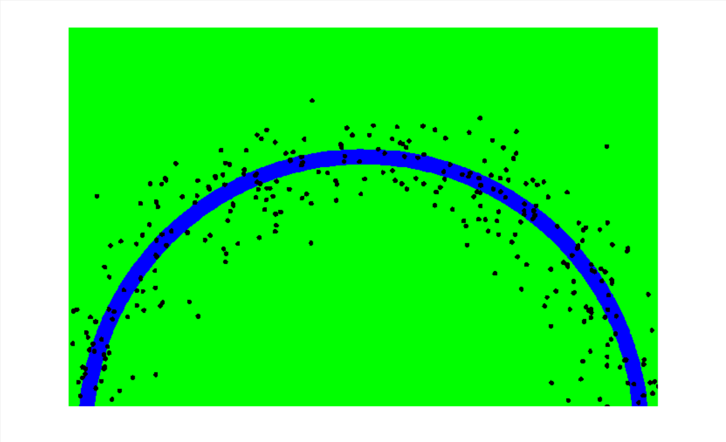


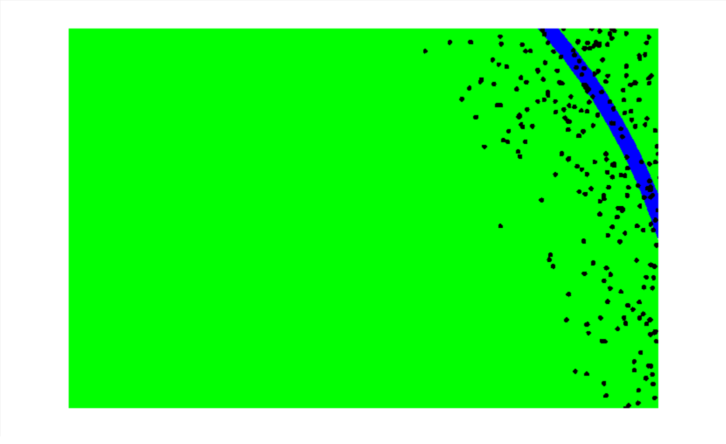
Wanted: samples distributed according to
$p(x|\, z_1, z_2, z_3)$

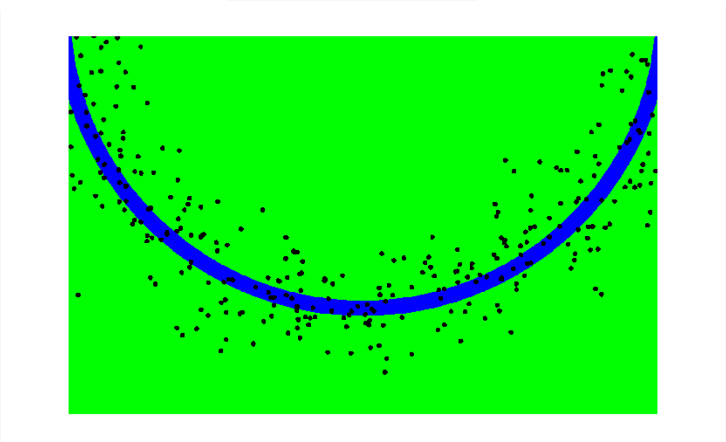


This is Easy!

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





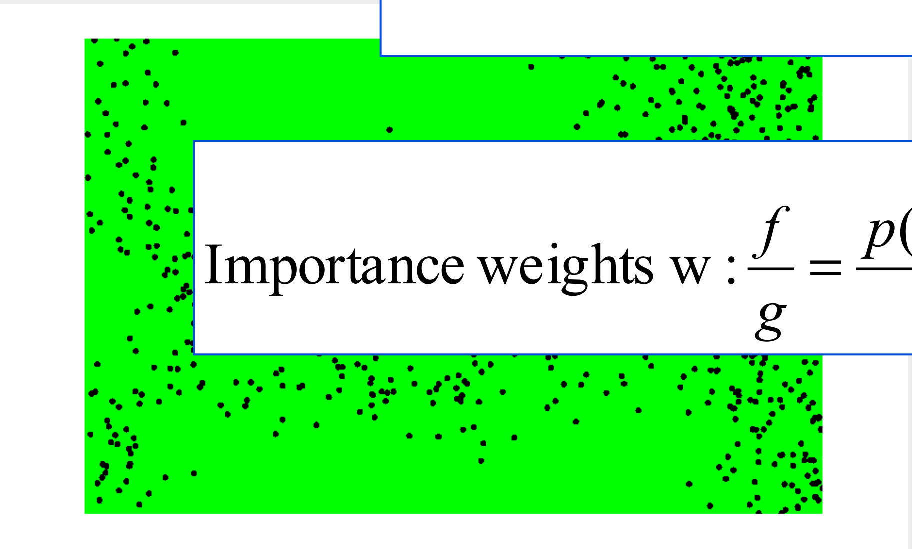


Importance Sampling with Resampling

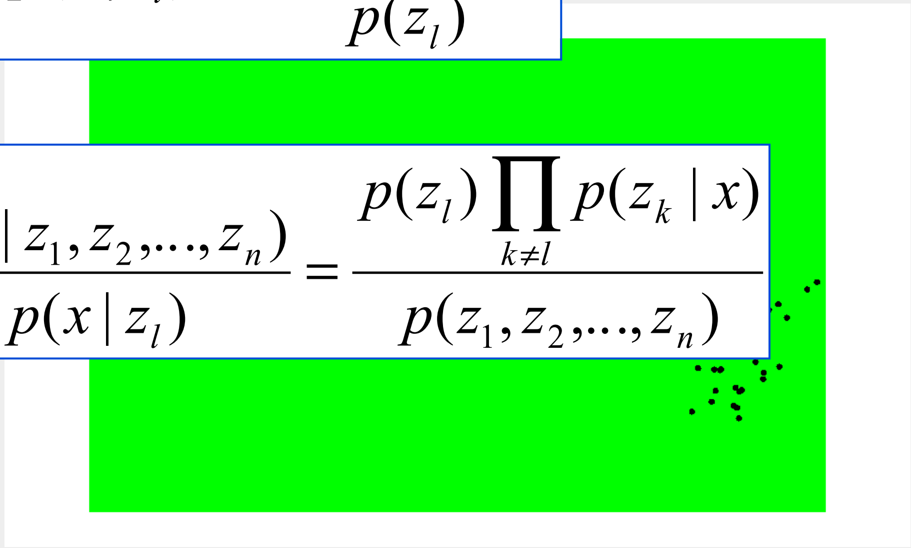
$$\text{Target distribution f}: p(x \mid z_1, z_2, \ldots, z_n) = \frac{\prod_k p(z_k \mid x) \quad p(x)}{p(z_1, z_2, \ldots, z_n)}$$

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

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



Weighted samples



After resampling

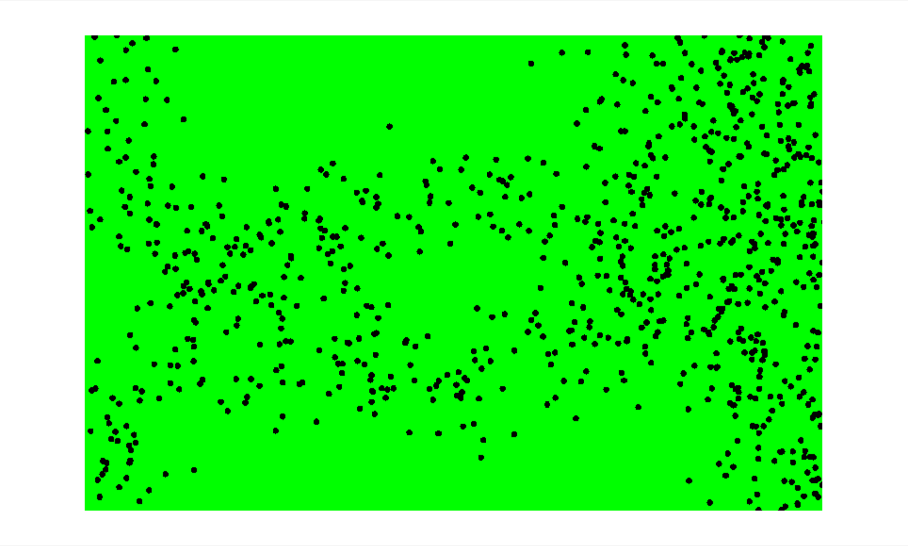
Importance Sampling with Resampling

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

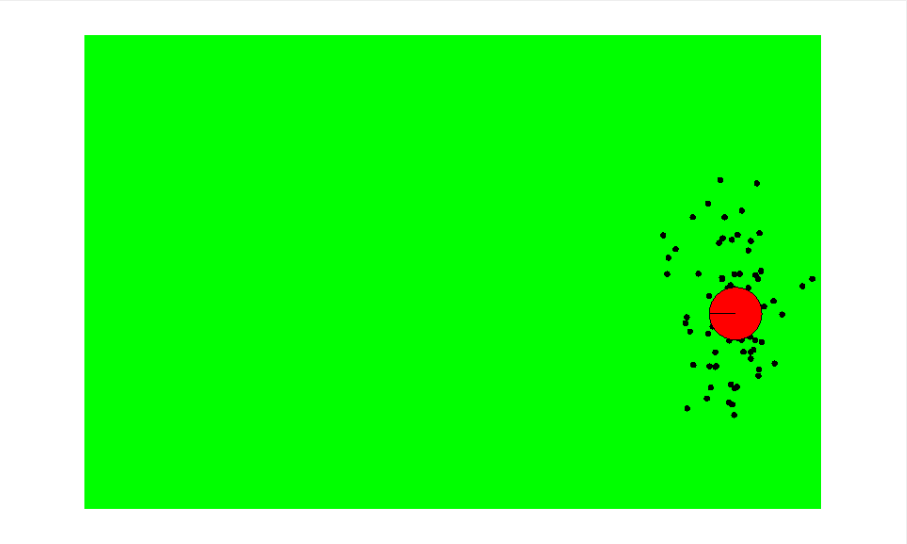
$$\text{Sampling distribution g}: p(x \mid z_l) = \frac{p(z_l \mid x) p(x)}{p(z_l)}$$

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

Importance Sampling with Resampling

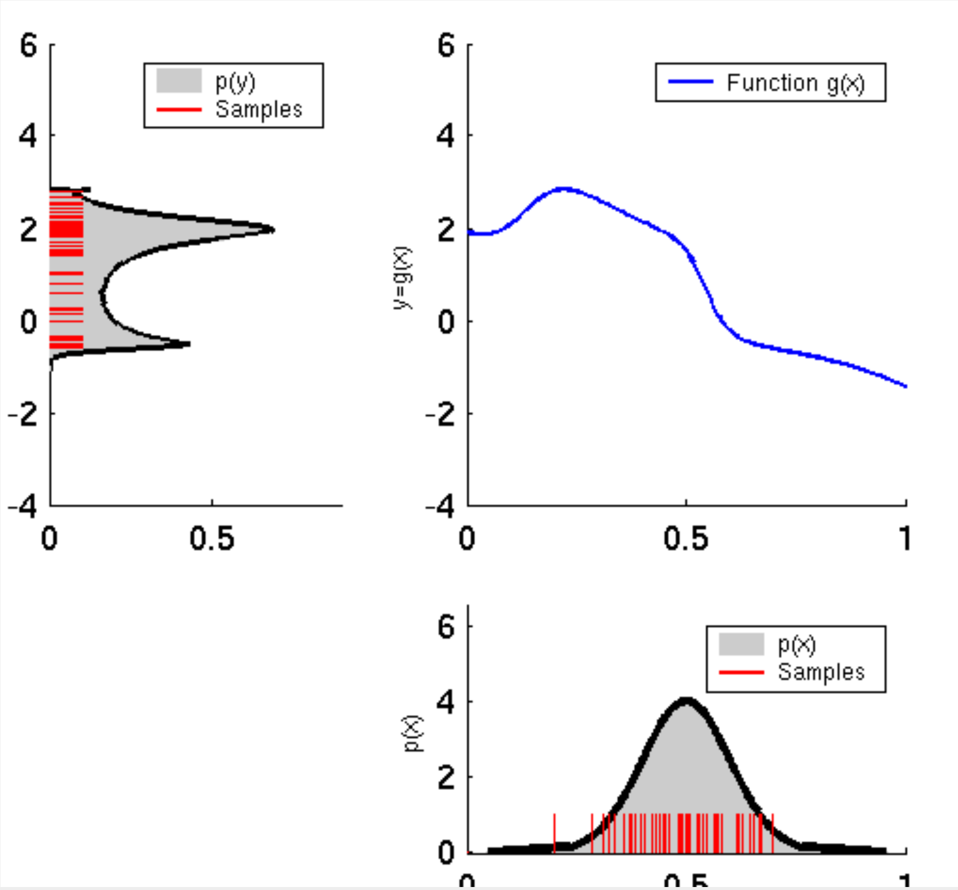


Weighted samples

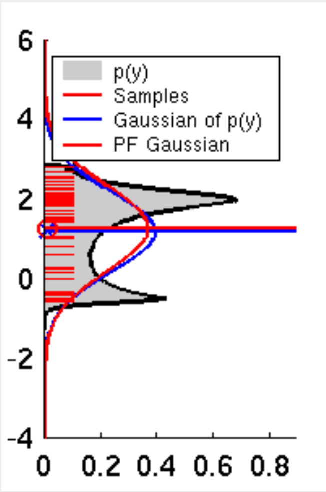


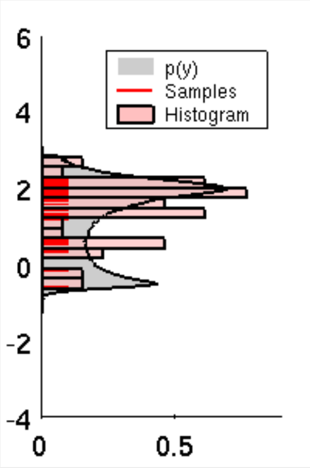
After resampling

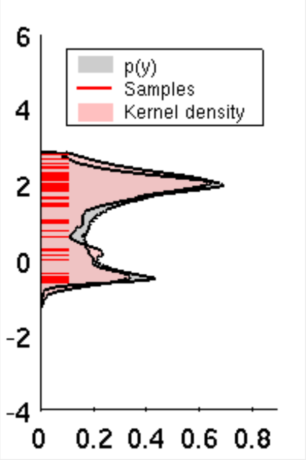
Particle Filter Projection

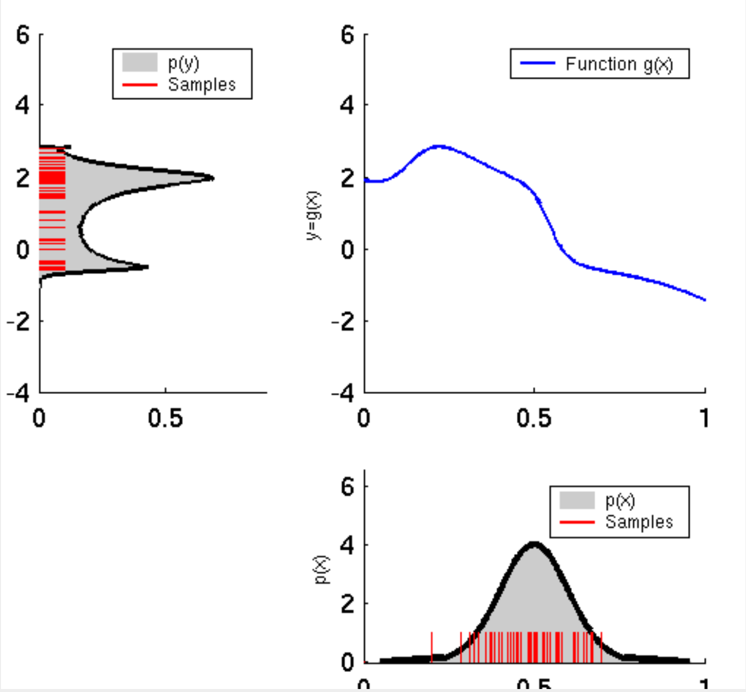


Density Extraction

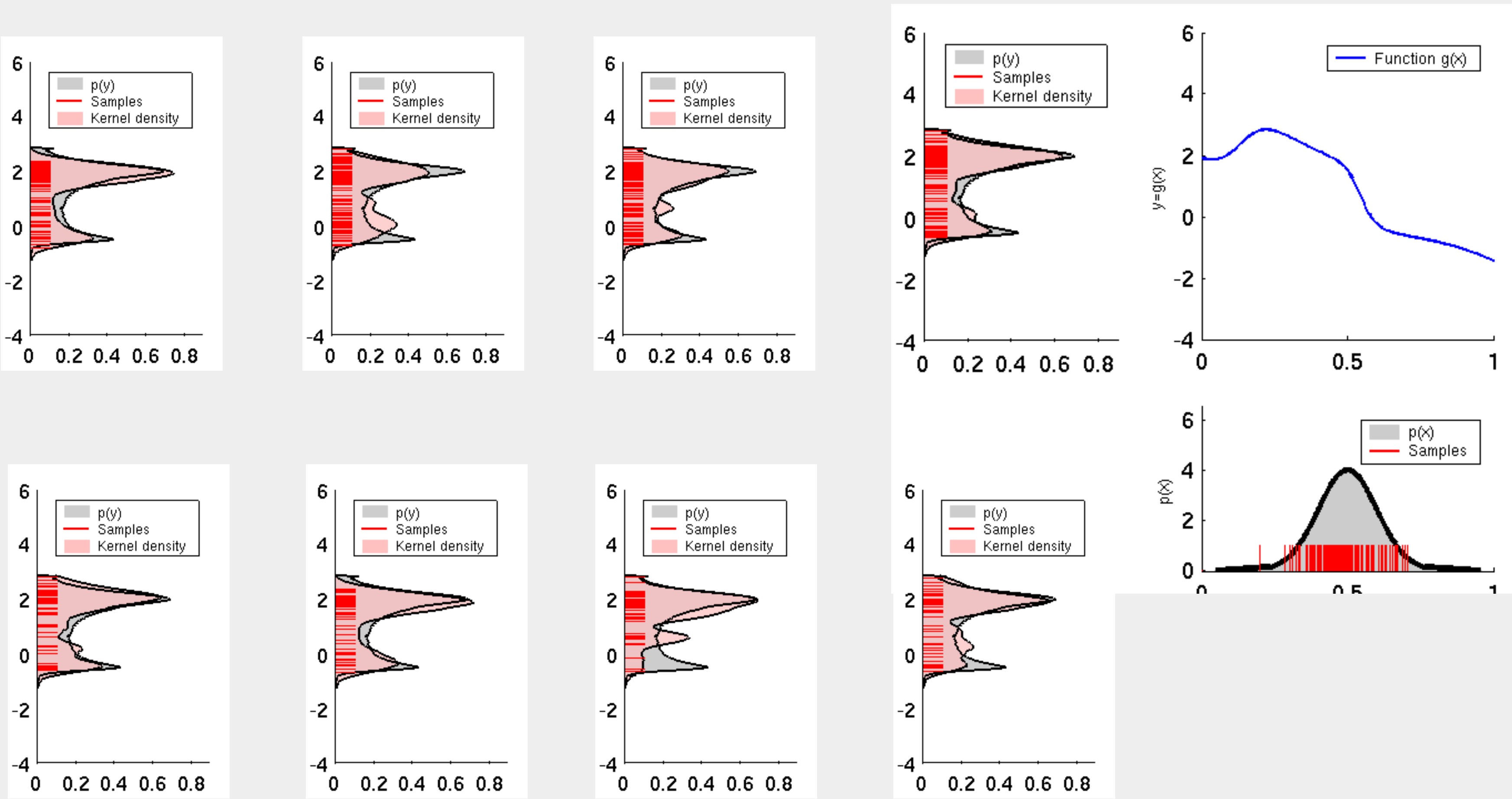




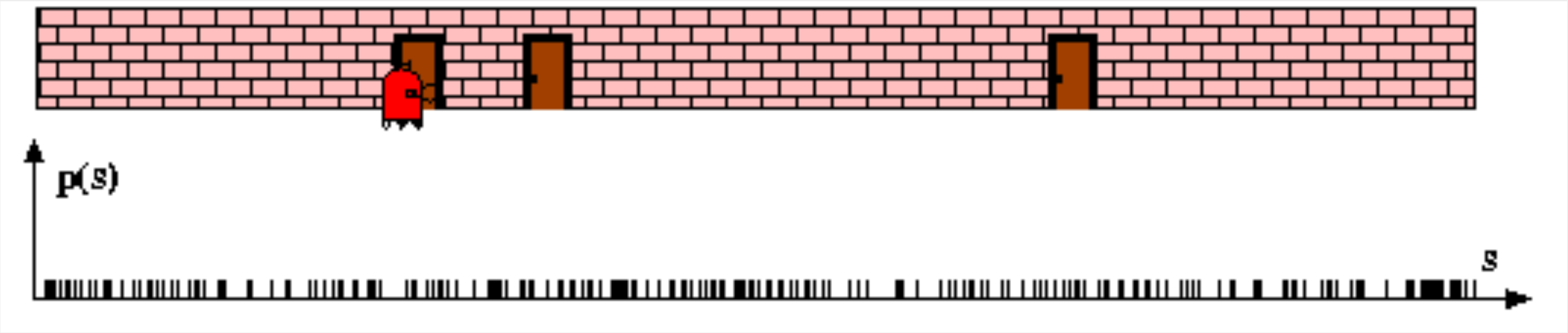




Sampling Variance



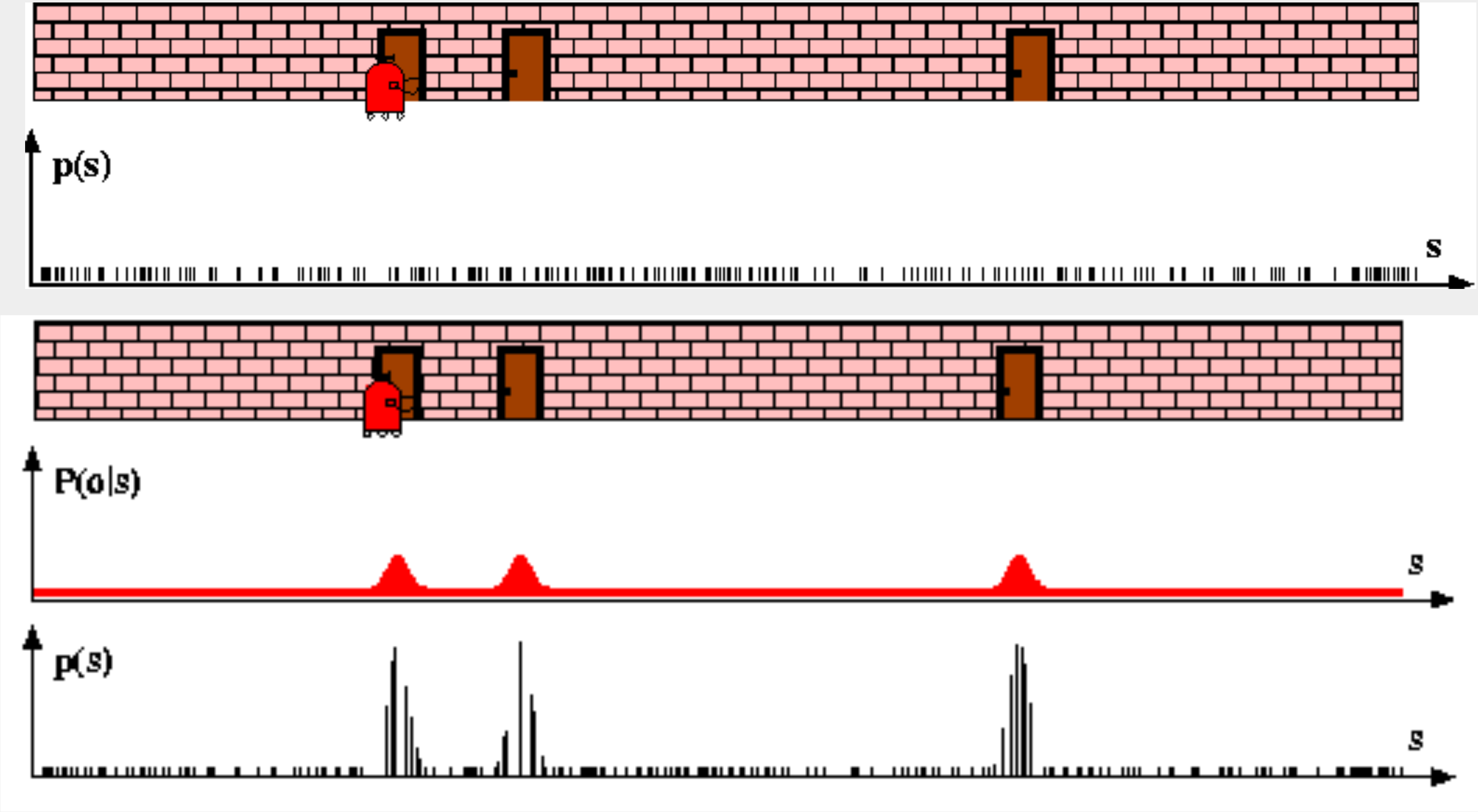
Particle Filters



Sensor Information: Importance Sampling

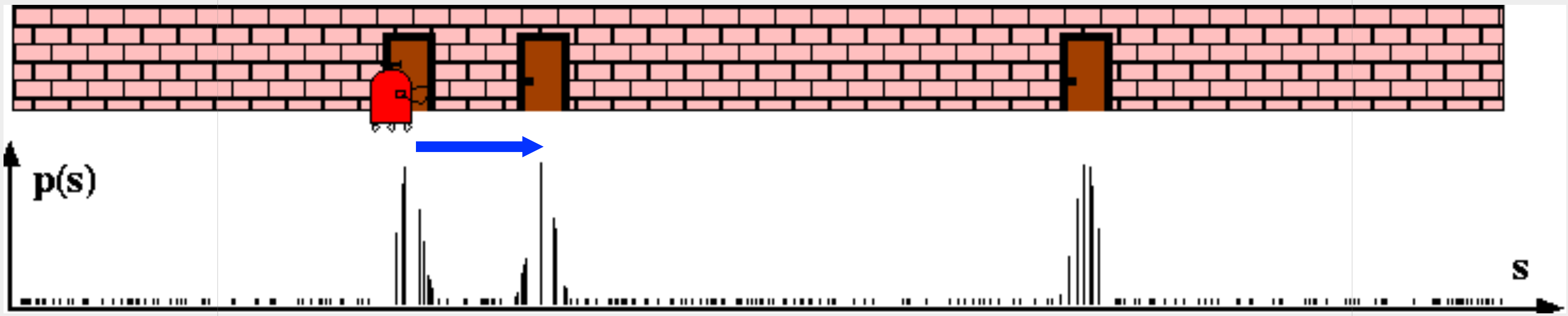
$$Bel(x) \leftarrow \alpha\, p(z \mid x)\, Bel^{-}(x)$$

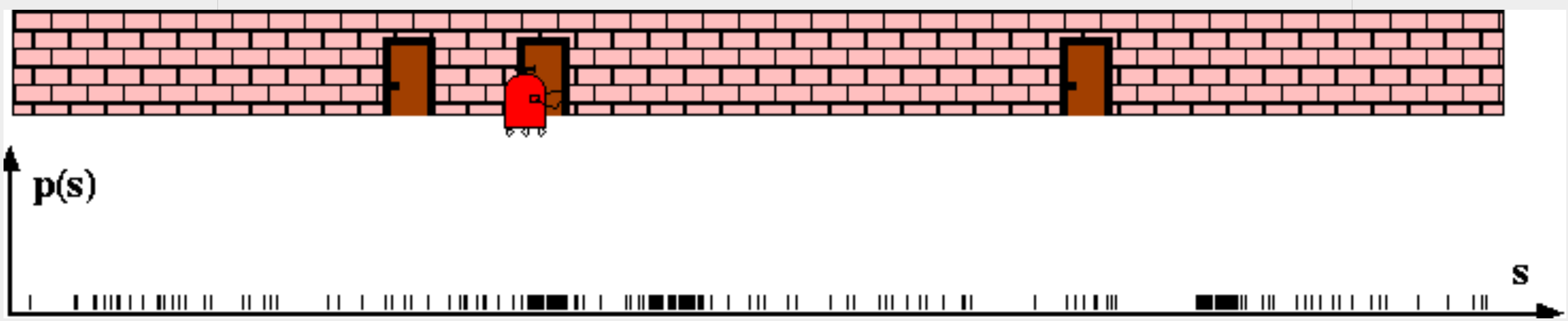
$$w \leftarrow \frac{\alpha\, p(z \mid x)\, Bel^{-}(x)}{Bel^{-}(x)} = \alpha\, p(z \mid x)$$



Robot Motion

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

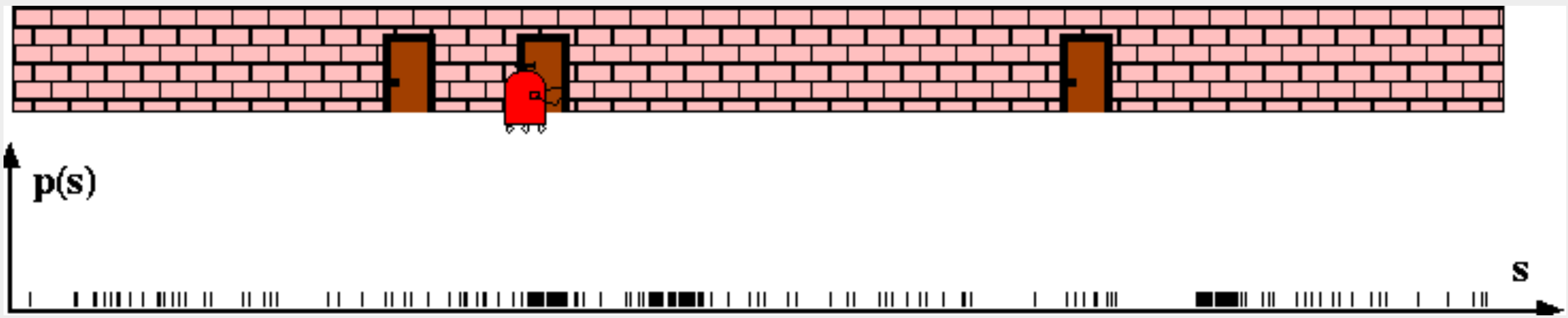


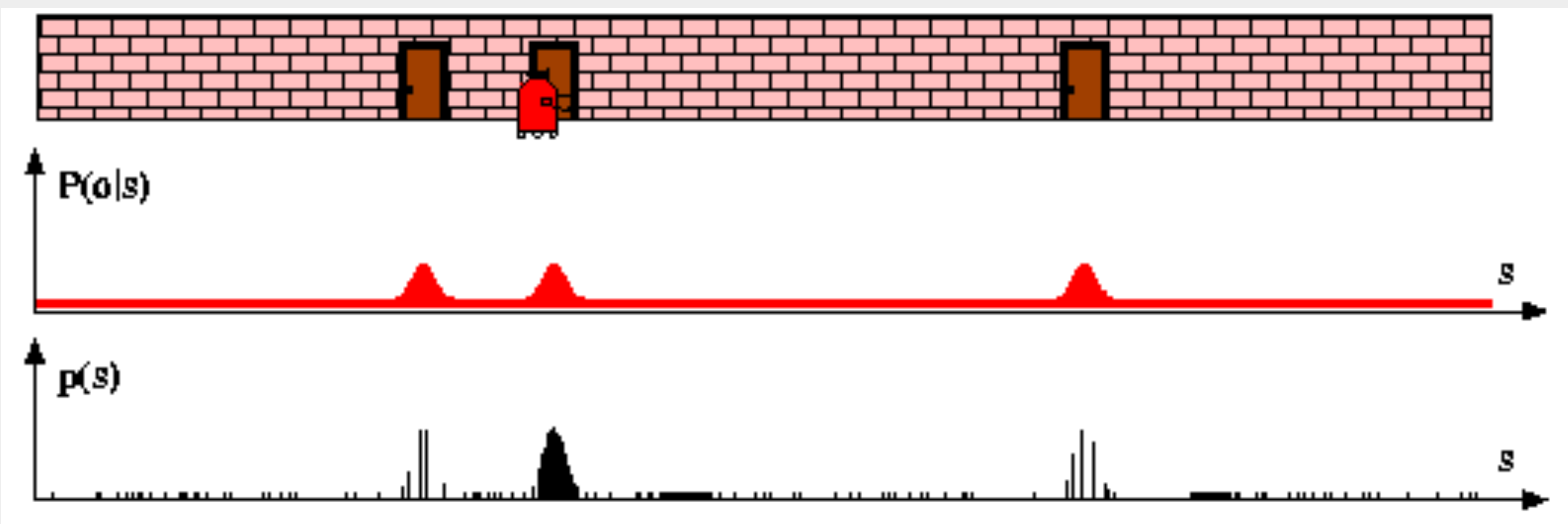


Sensor Information: Importance Sampling

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

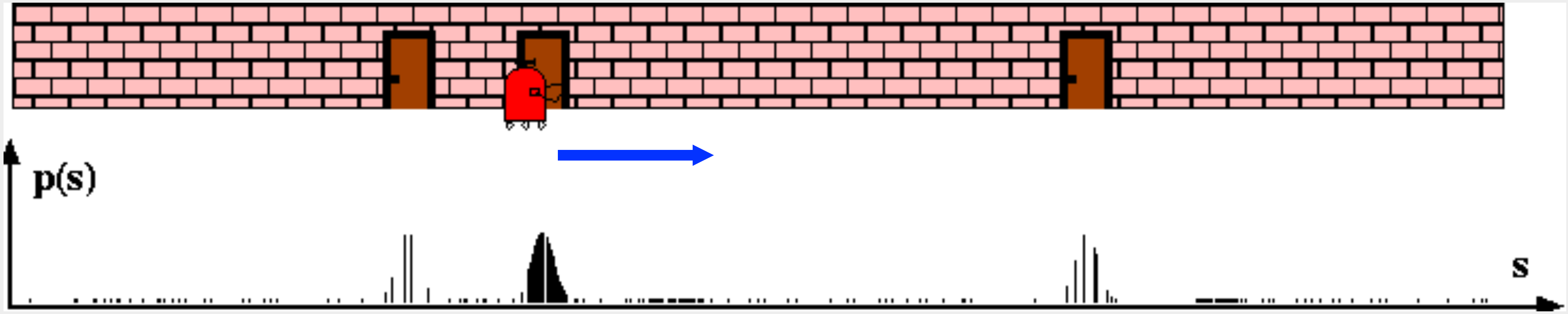
$$w \leftarrow \frac{\alpha\, p(z\,|\,x)\, Bel^{-}(x)}{Bel^{-}(x)} = \alpha\, p(z\,|\,x)$$

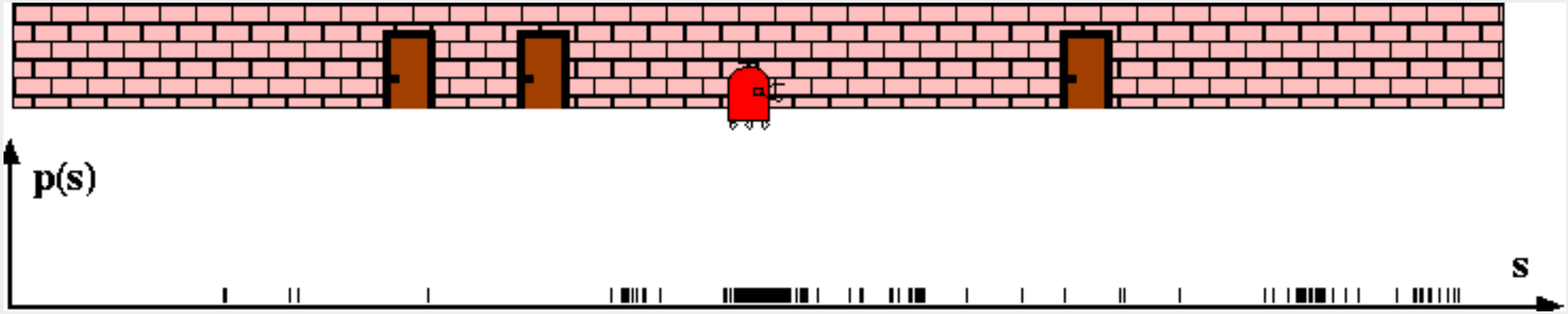




Robot Motion

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





Particle Filter Algorithm

1. Algorithm **particle_filter**(S_{t-1}, u_{t-1} z_t):
2. $S_t = \emptyset, \quad \eta = 0$
3. **For** $i = 1 \ldots n$ ***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** $i = 1 \ldots n$
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 $p(x_t \mid x^i_{t-1}, u_{t-1})$

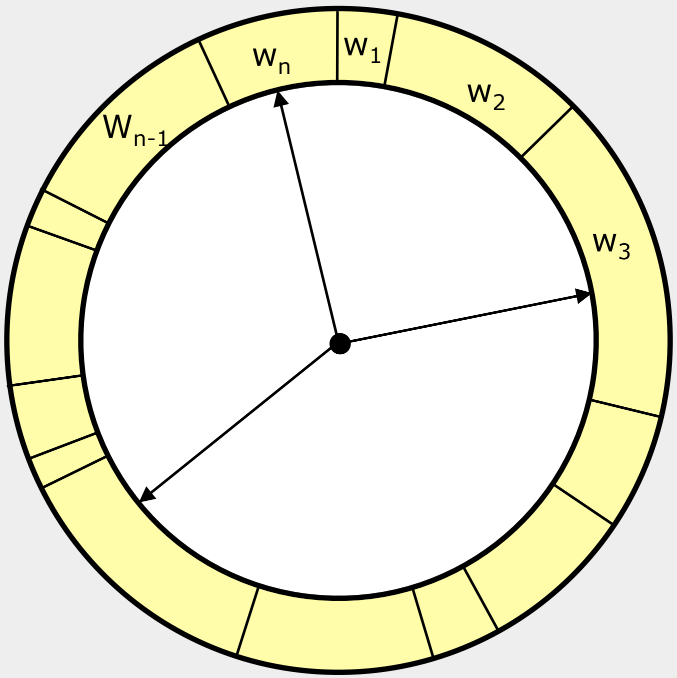
Importance factor for x^i_t:

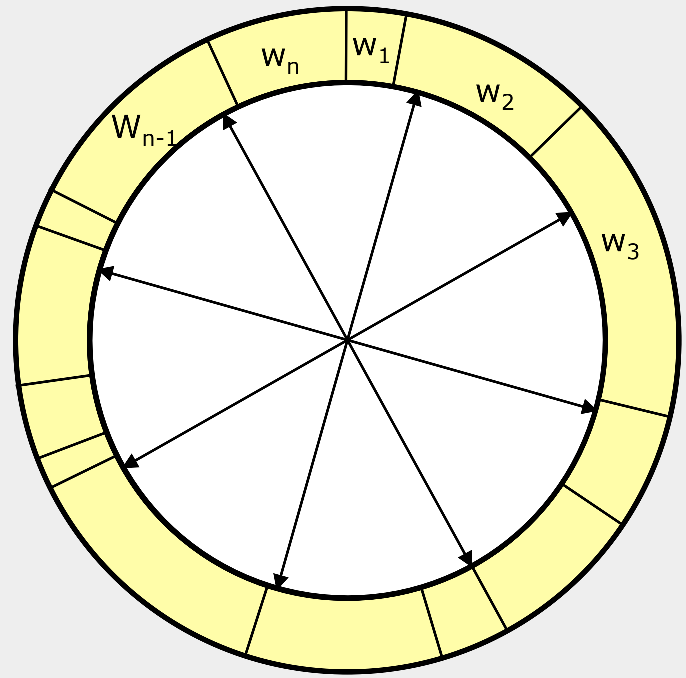
$$w_t^i = \frac{\text{target distribution}}{\text{proposal distribution}} = \frac{\eta\, p(z_t \mid x_t)\, p(x_t \mid x_{t-1}, u_{t-1})\, Bel(x_{t-1})}{p(x_t \mid x_{t-1}, u_{t-1})\, Bel(x_{t-1})} \propto p(z_t \mid x_t)$$

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

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

$S' = \emptyset, c_1 = w^1$

For $i = 2 \ldots n$ — ***Generate cdf***

$\quad c_i = c_{i-1} + w^i$

$u_1 \sim U[0, n^{-1}], i = 1$ — ***Initialize threshold***

For $j = 1 \ldots n$ — ***Draw samples …***

$\quad$ **While** ($u_j > c_i$) — ***Skip until next threshold reached***

$\quad\quad i = i + 1$

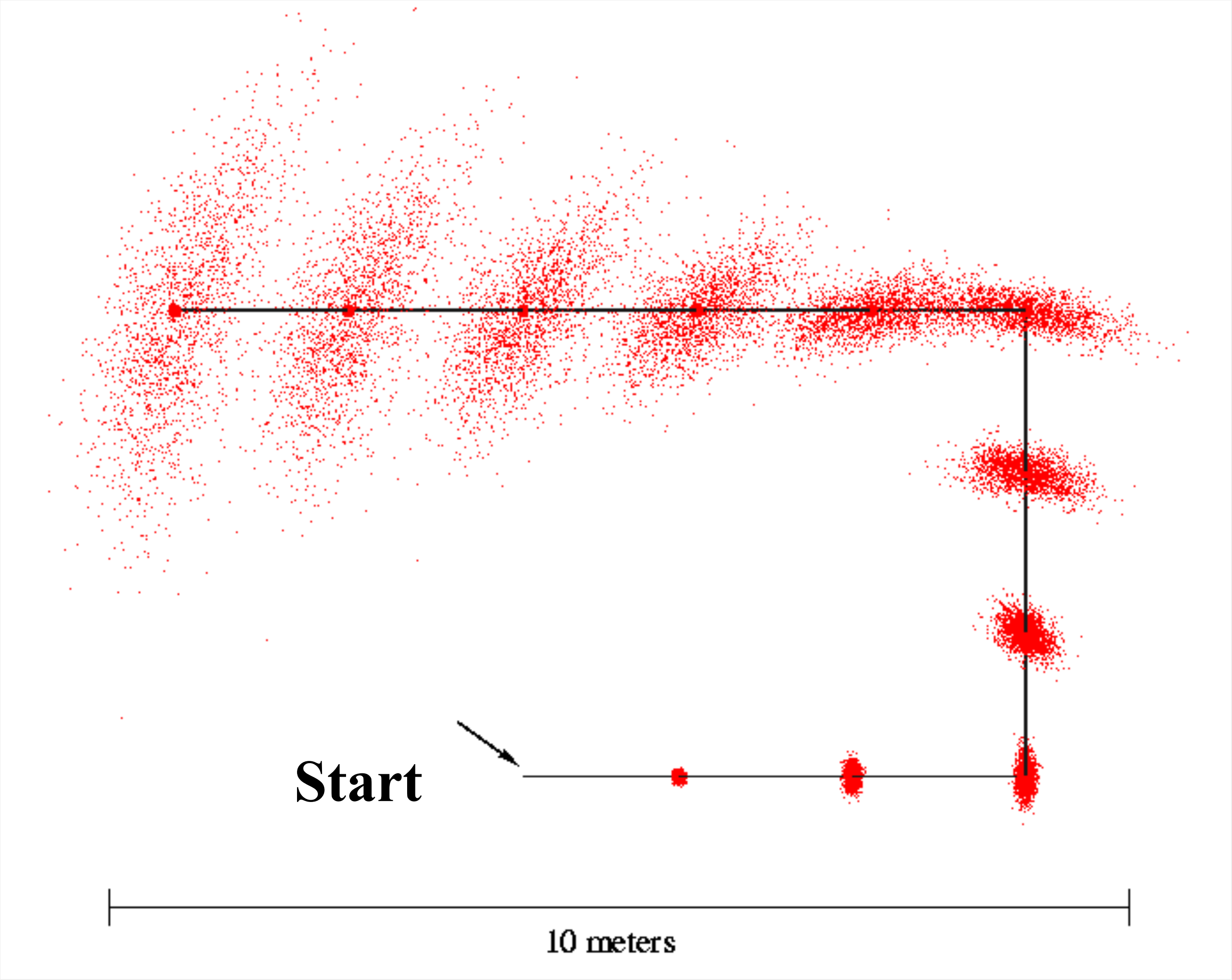
$\quad S' = S' \cup \{ < x^i, n^{-1} > \}$ — ***Insert***

$\quad u_j = u_j + n^{-1}$ — ***Increment threshold***

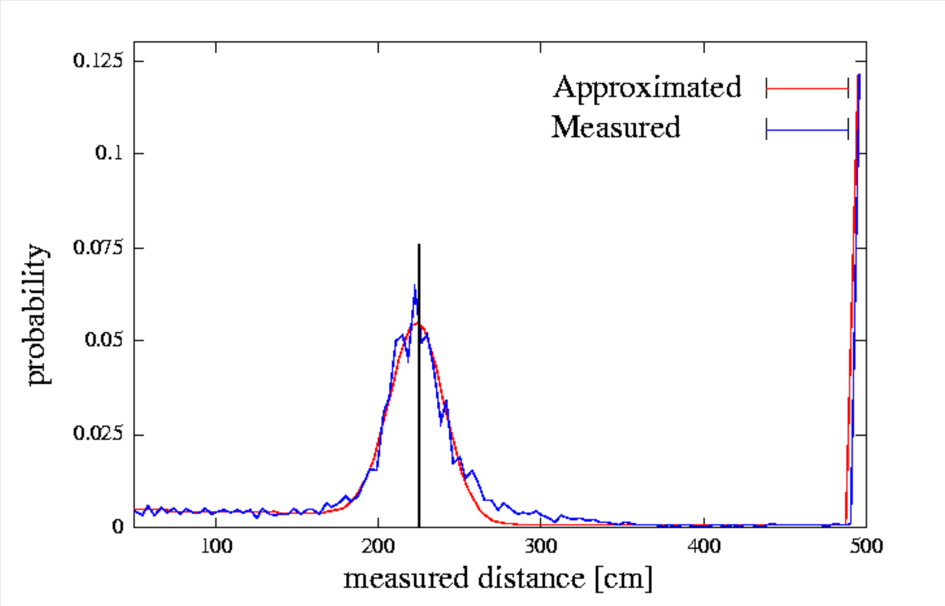
Return S'

Also called **stochastic universal sampling**

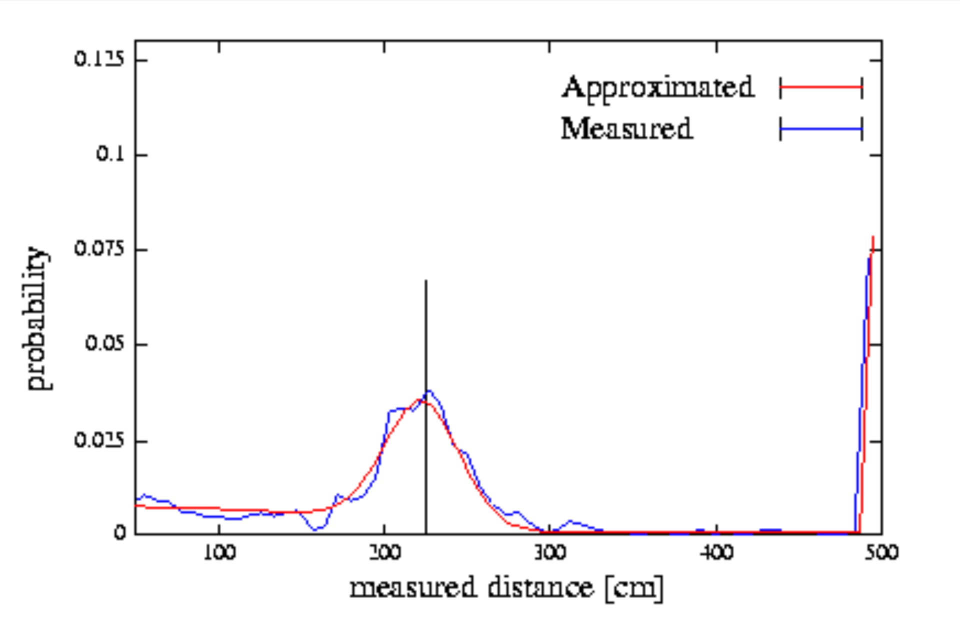
Motion Model Reminder



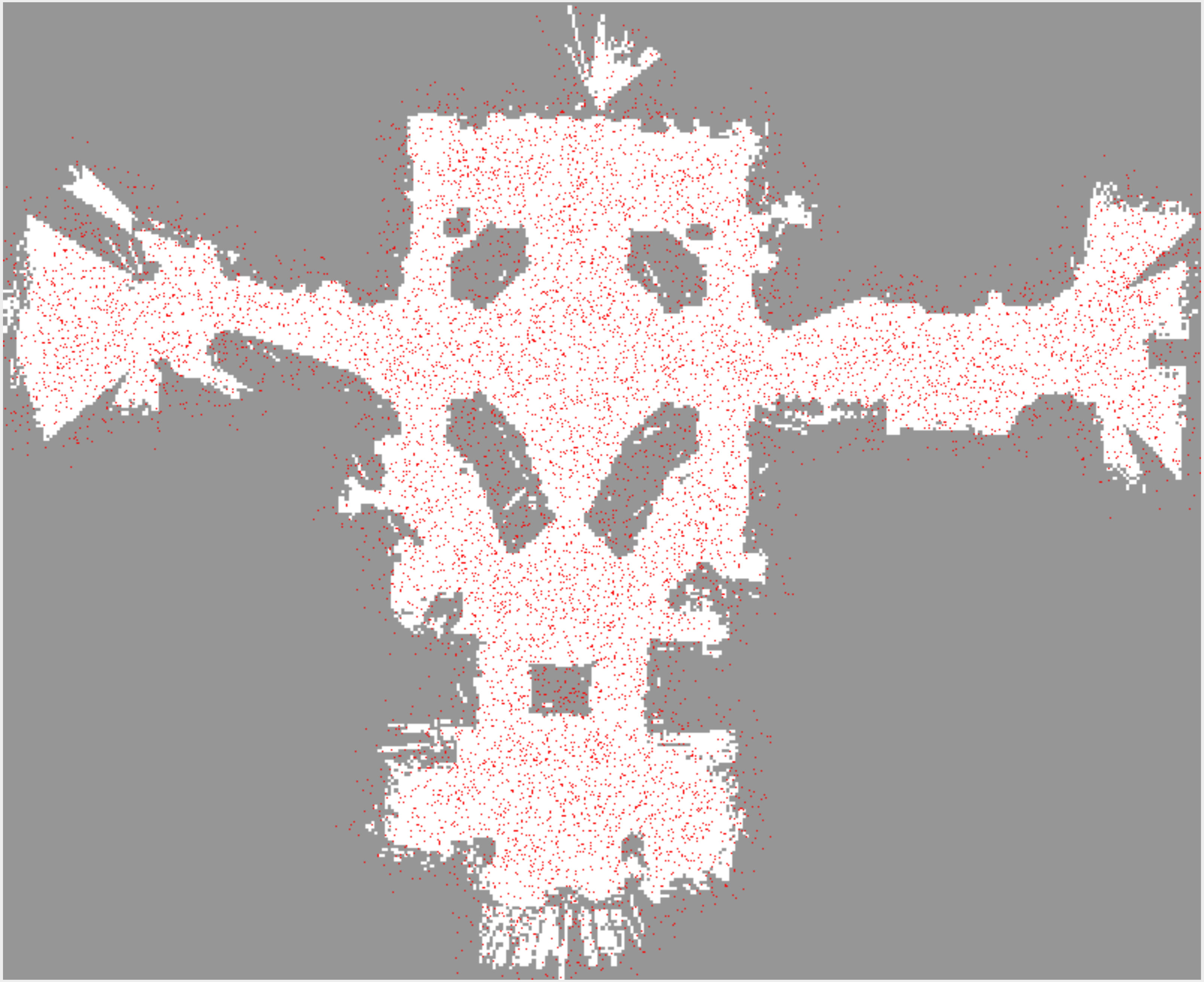
Proximity Sensor Model Reminder

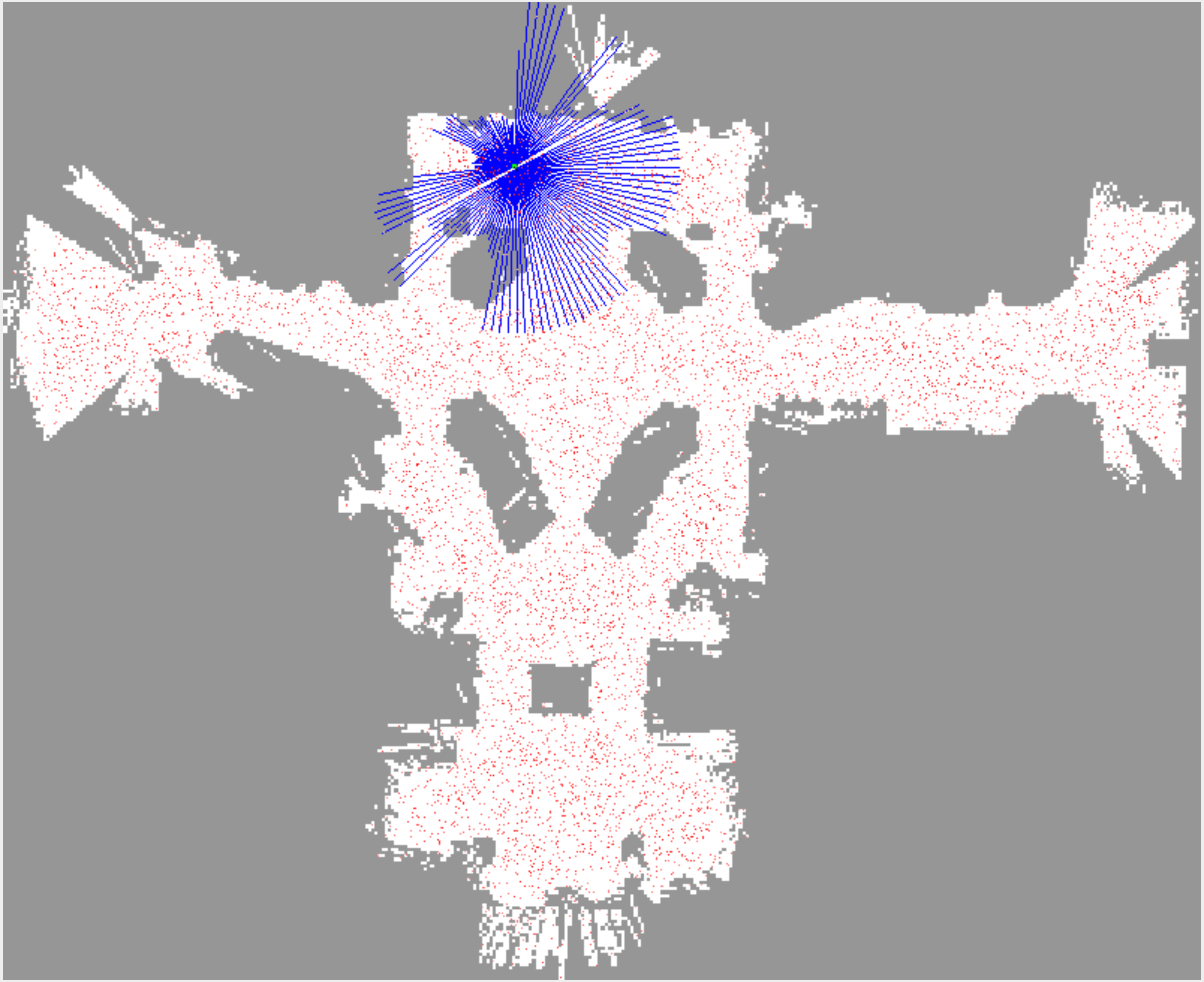


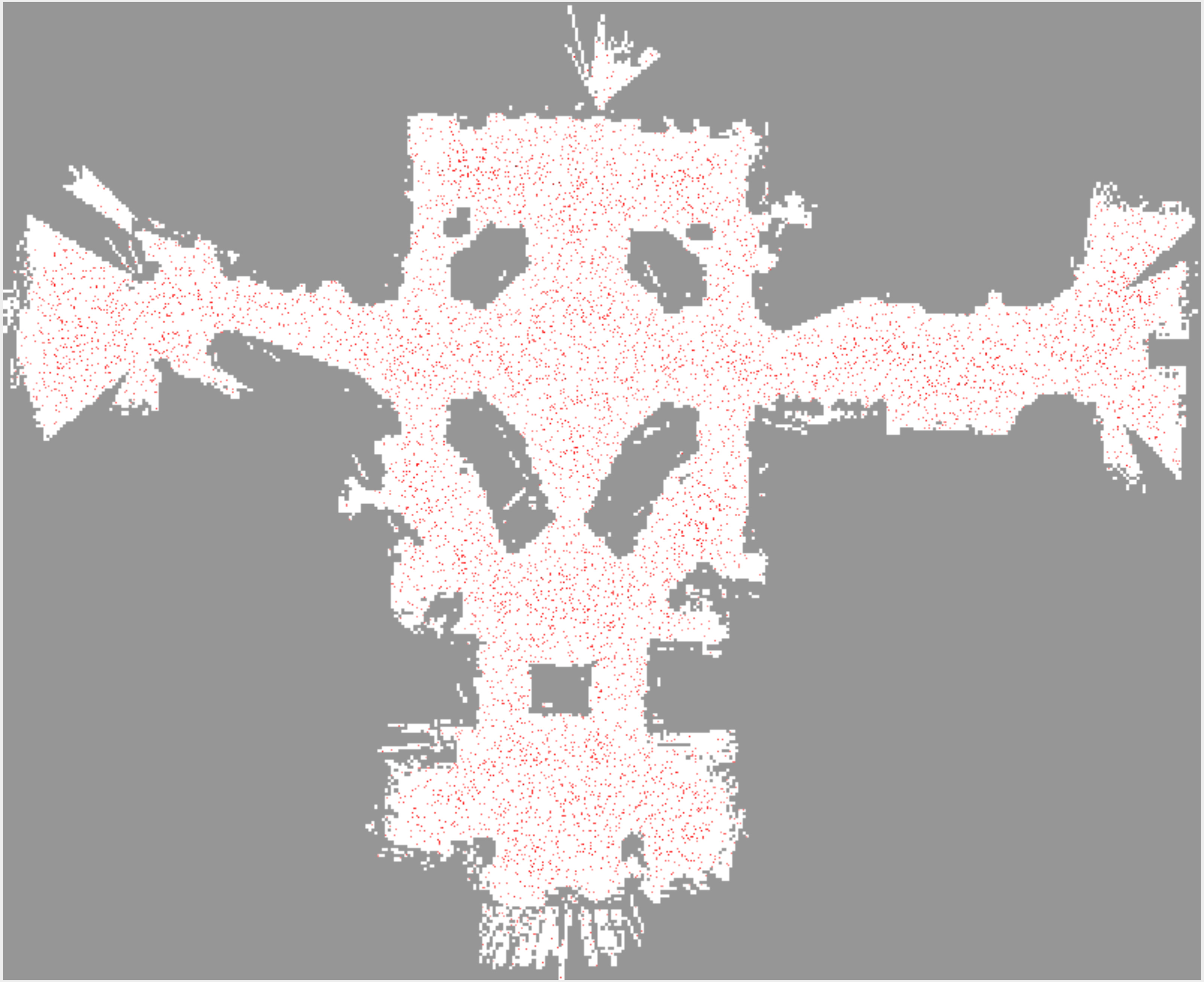
Laser sensor

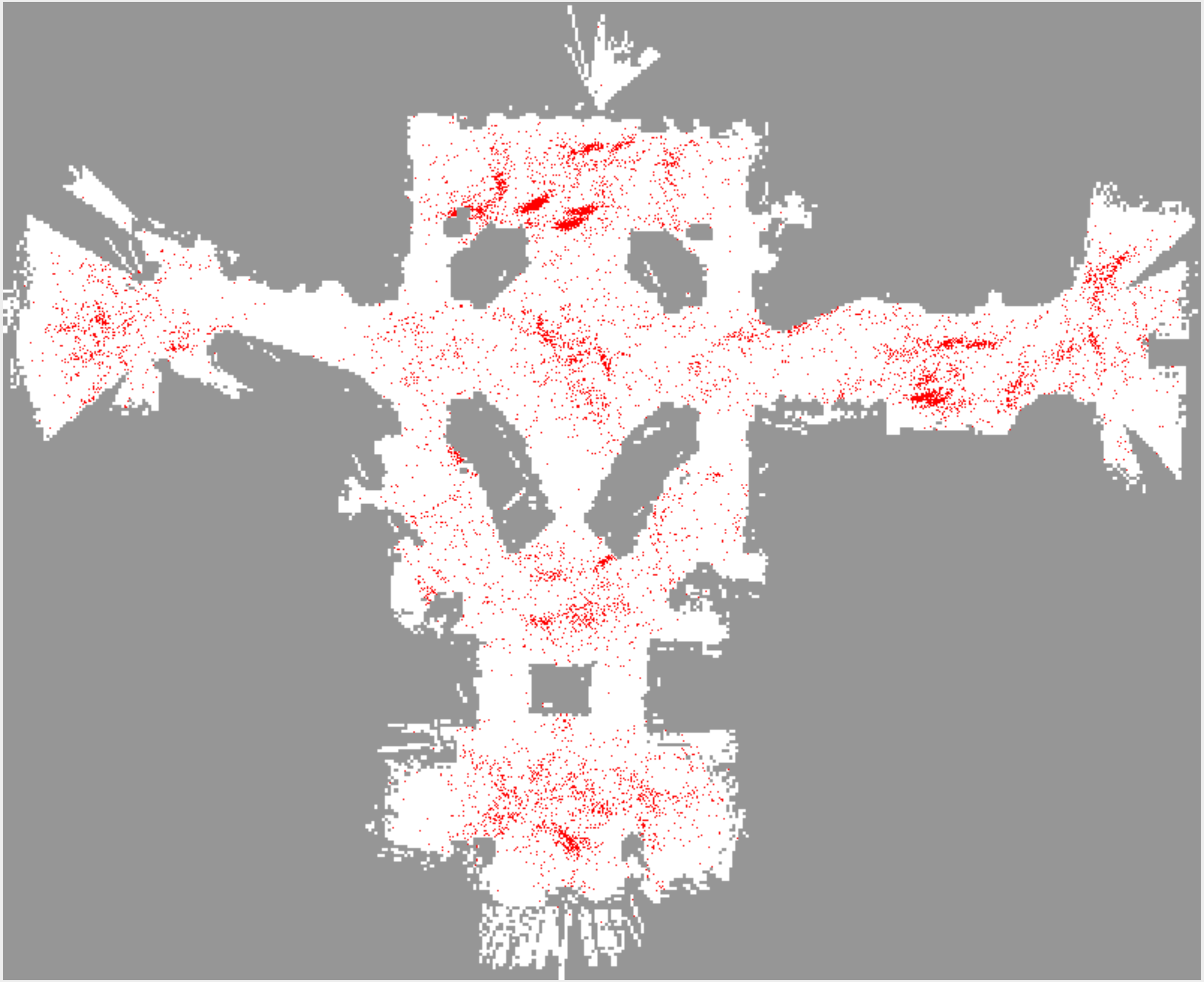


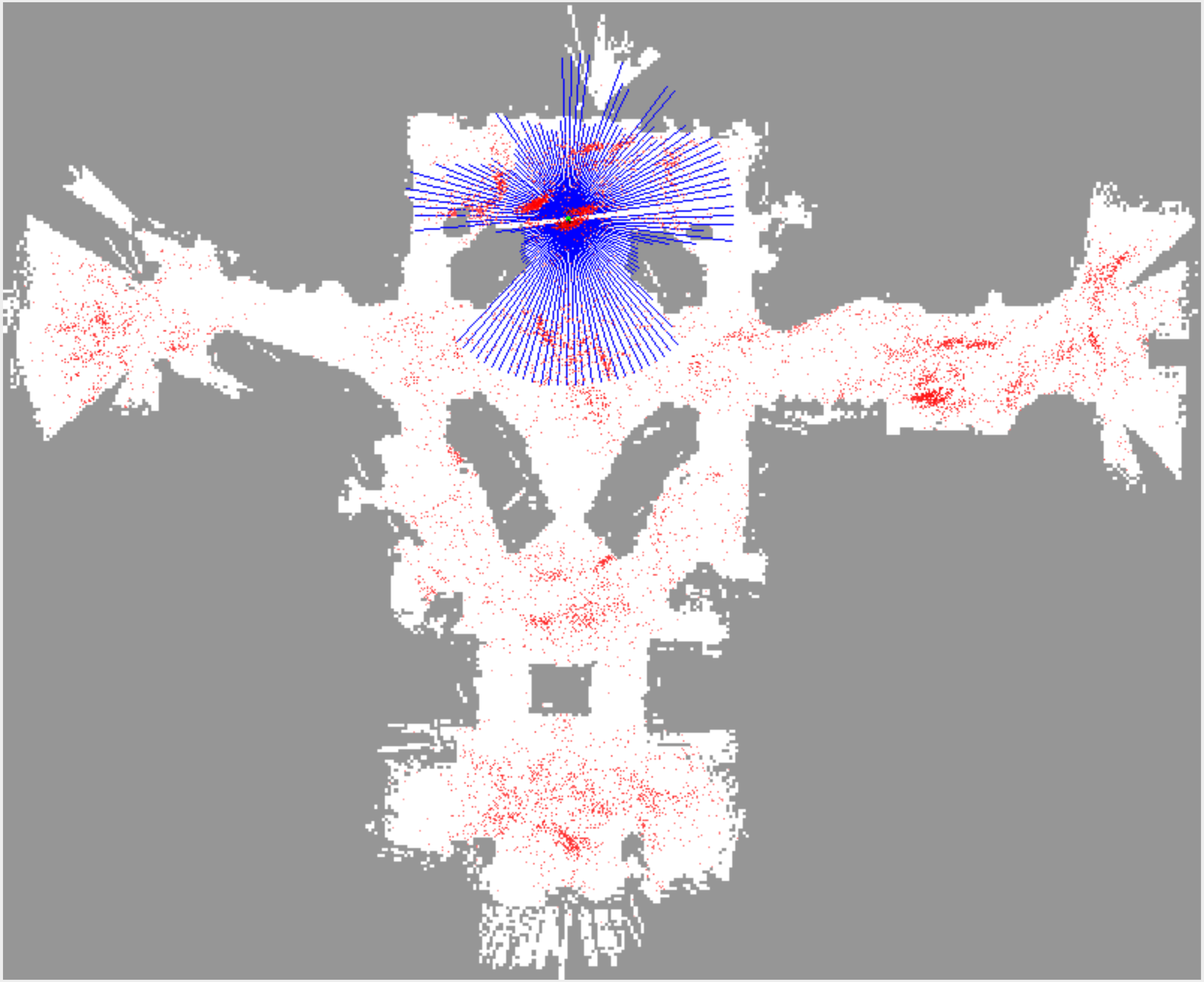
Sonar sensor

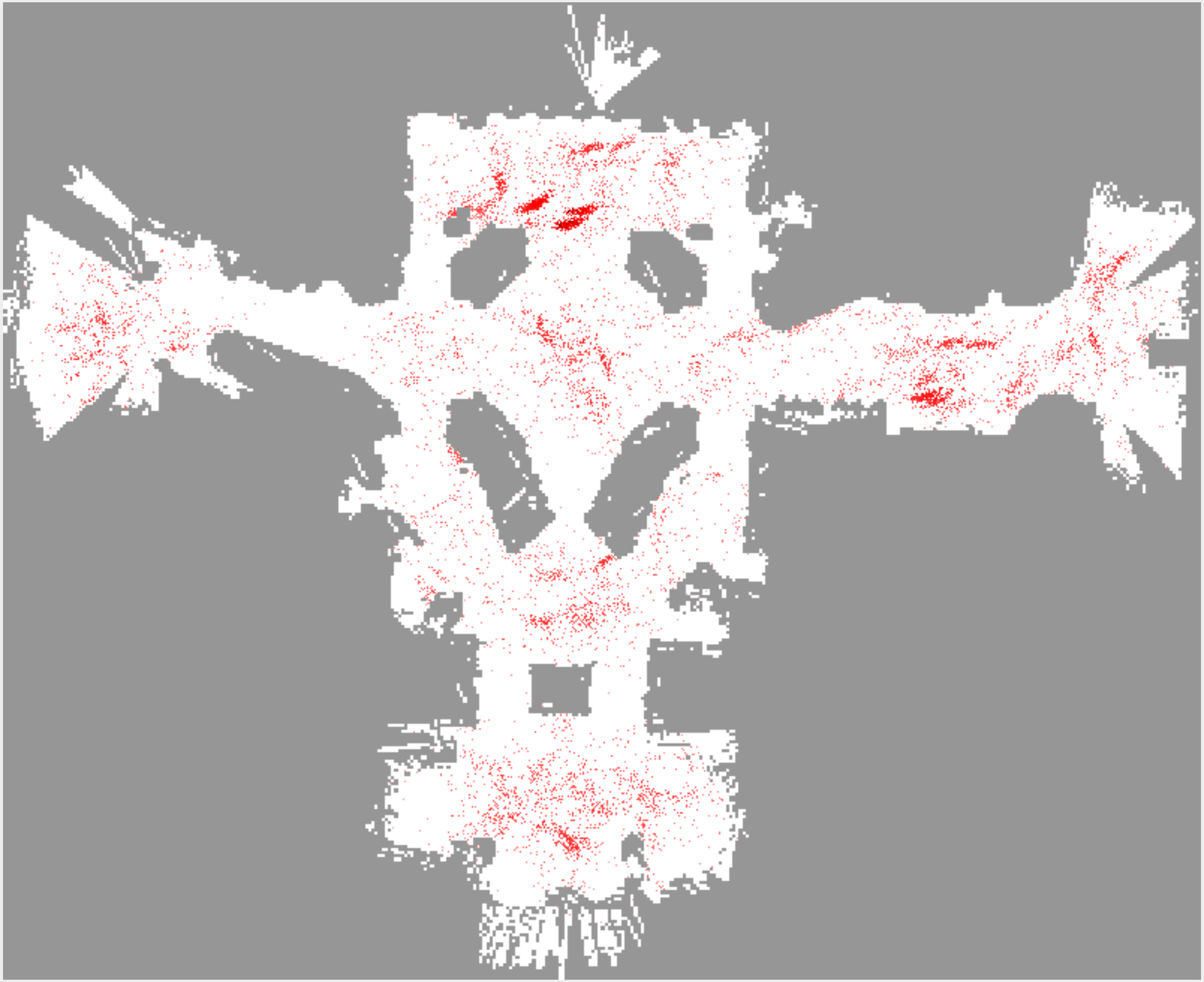


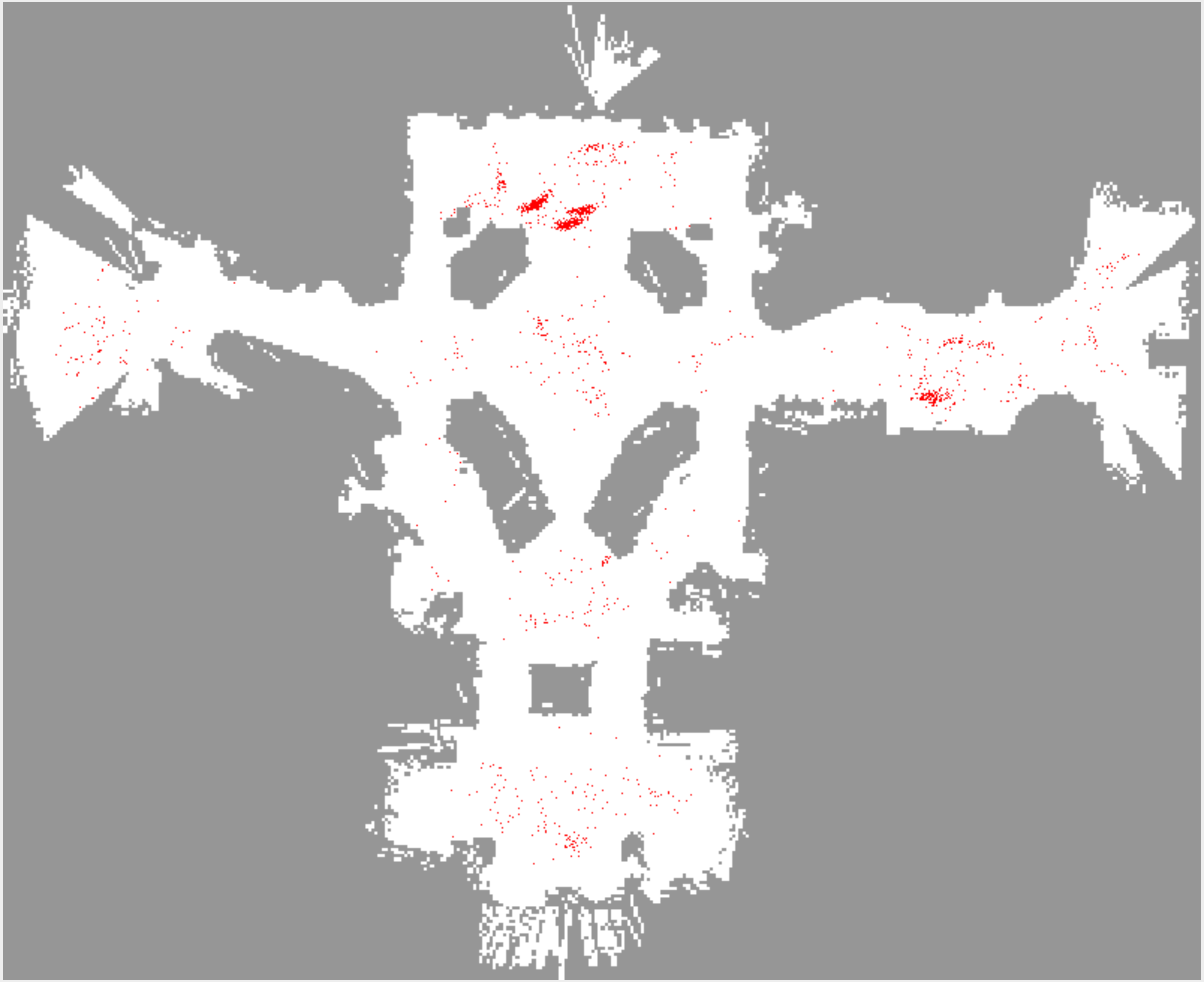


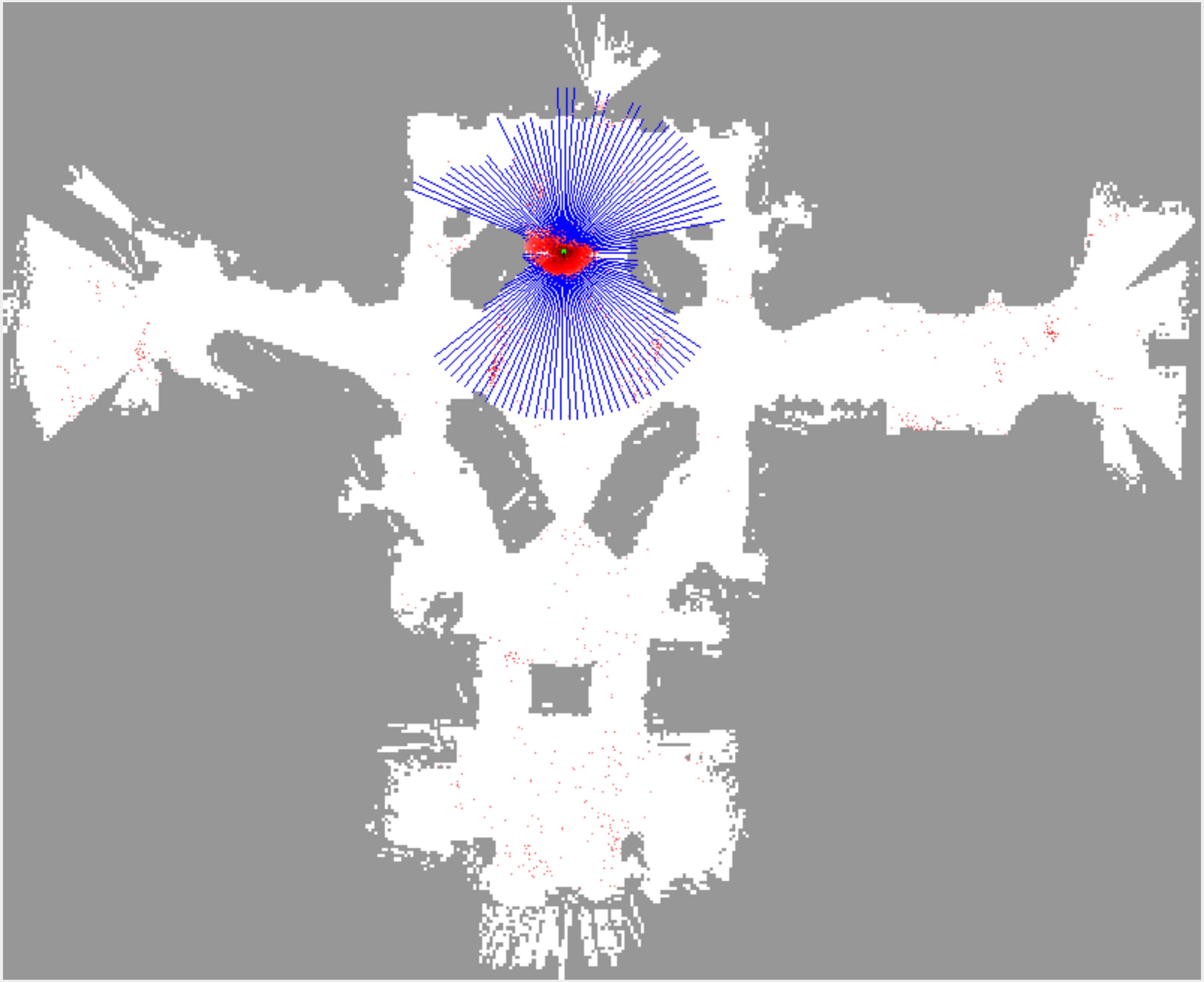


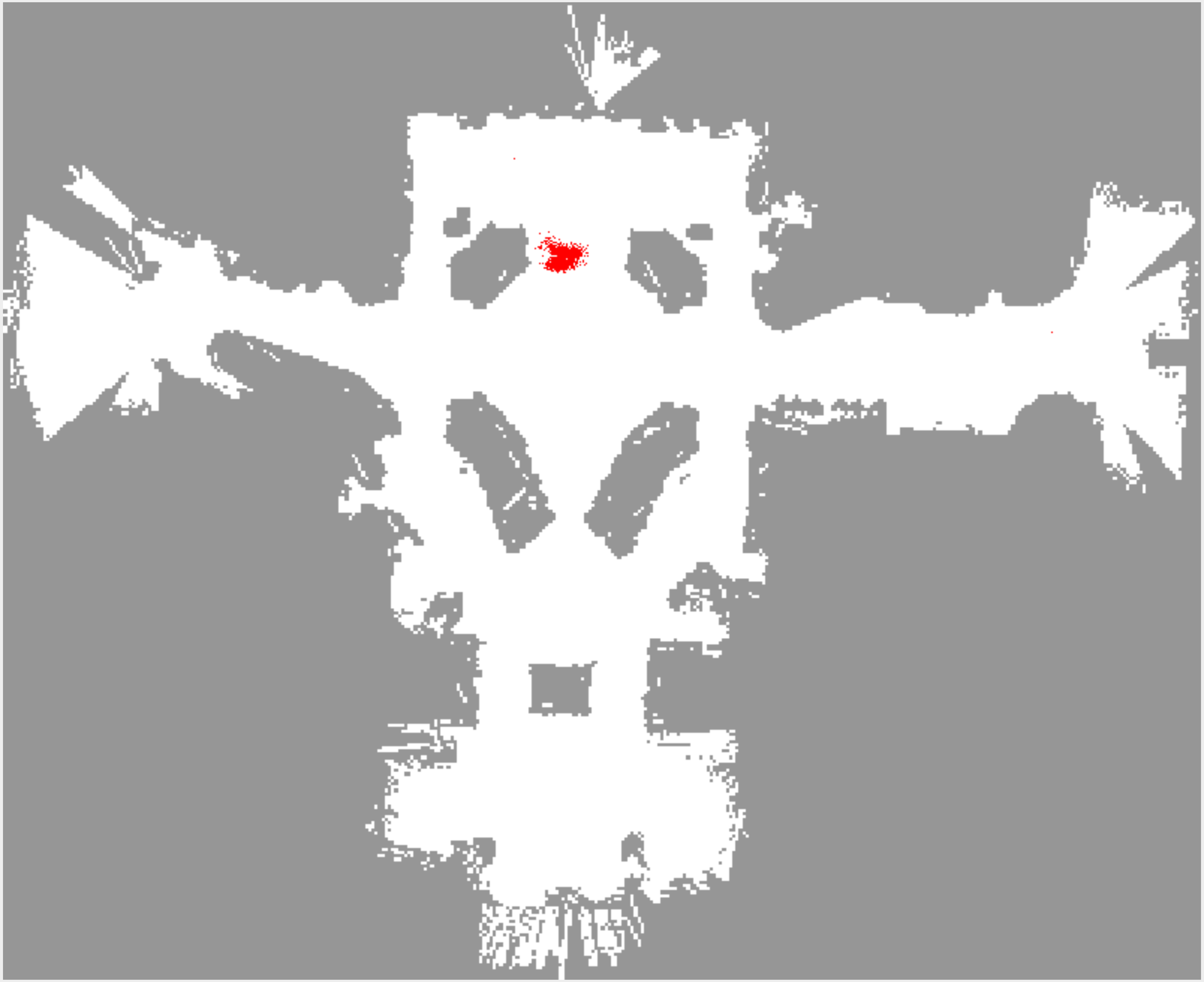


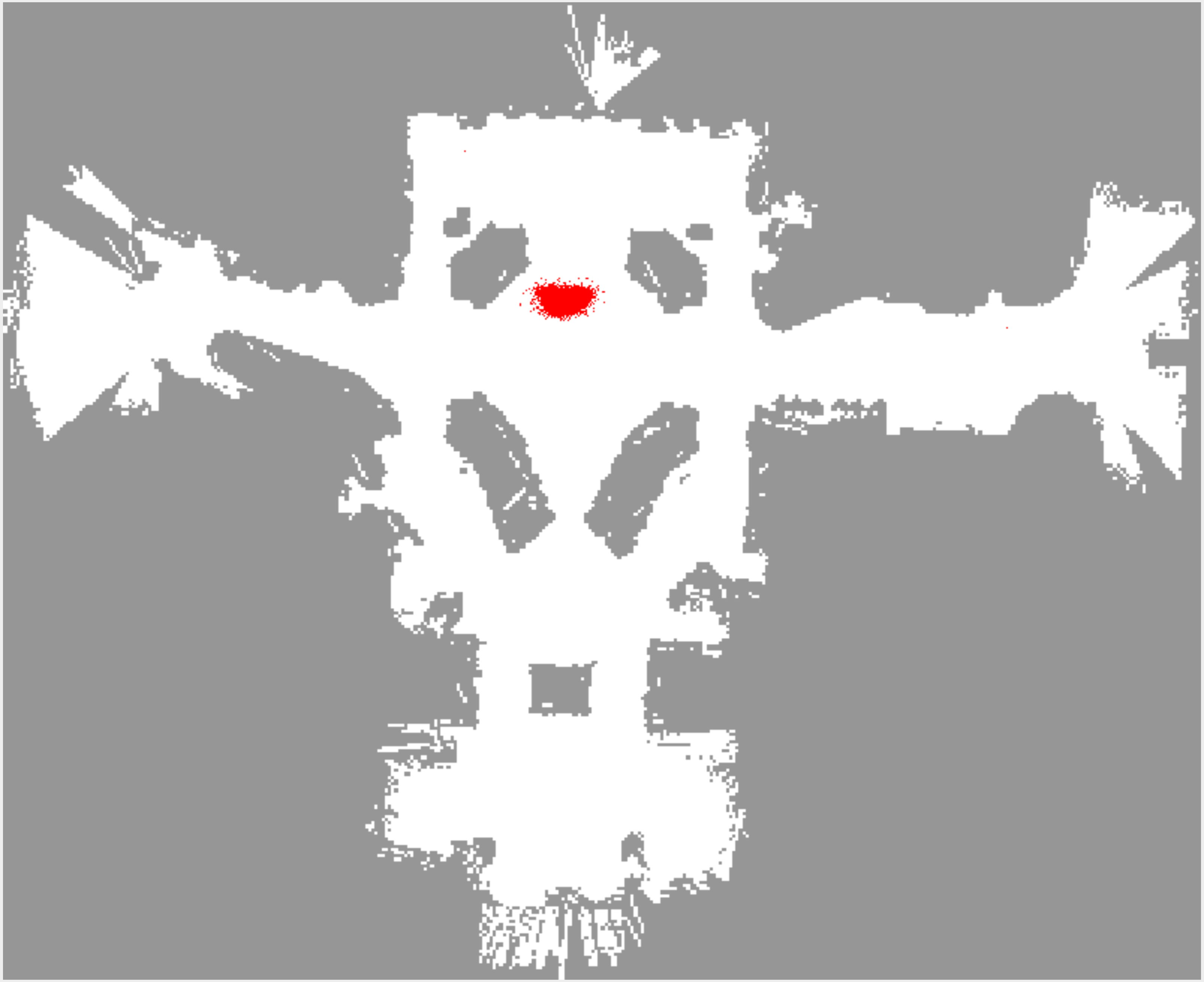


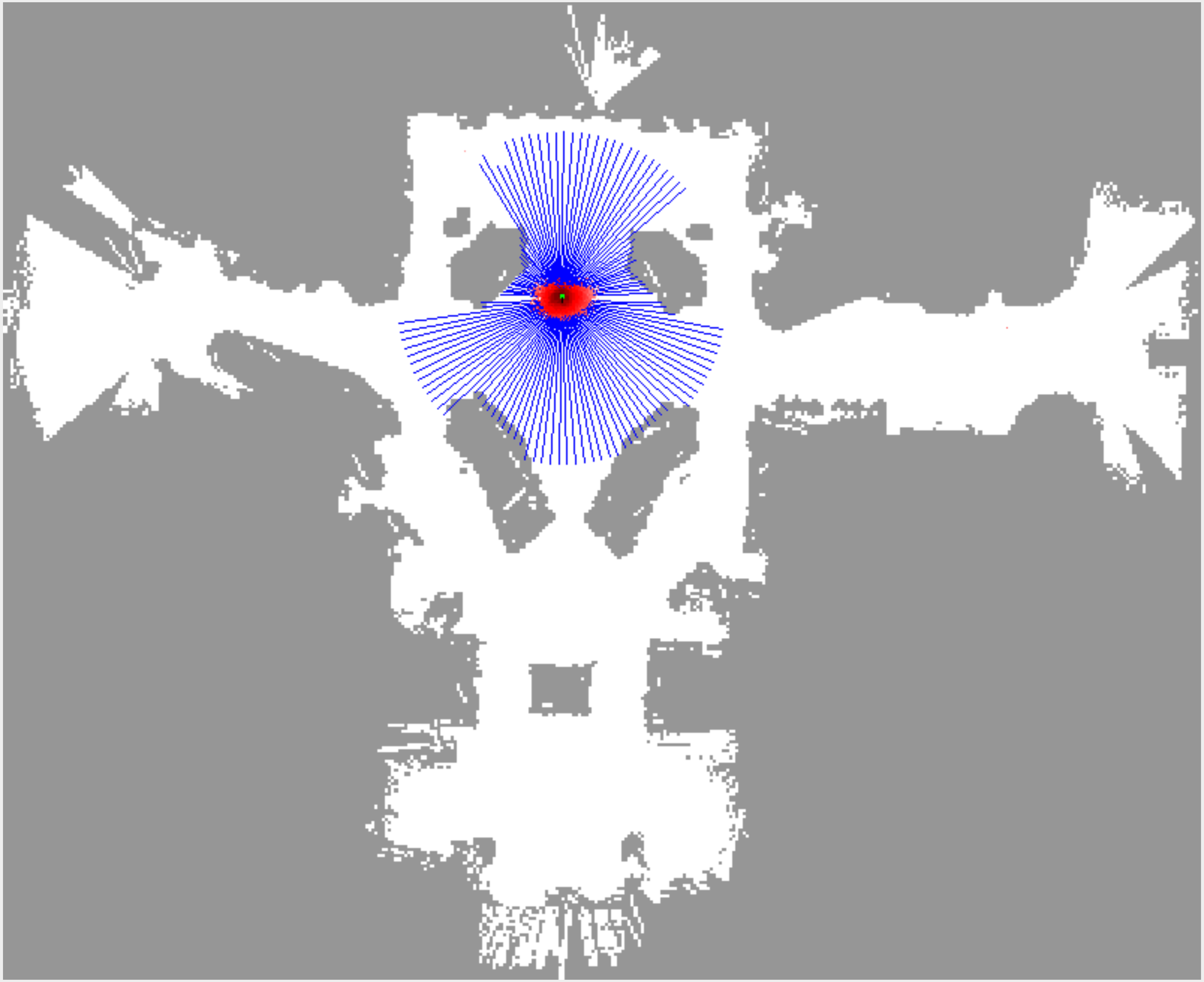


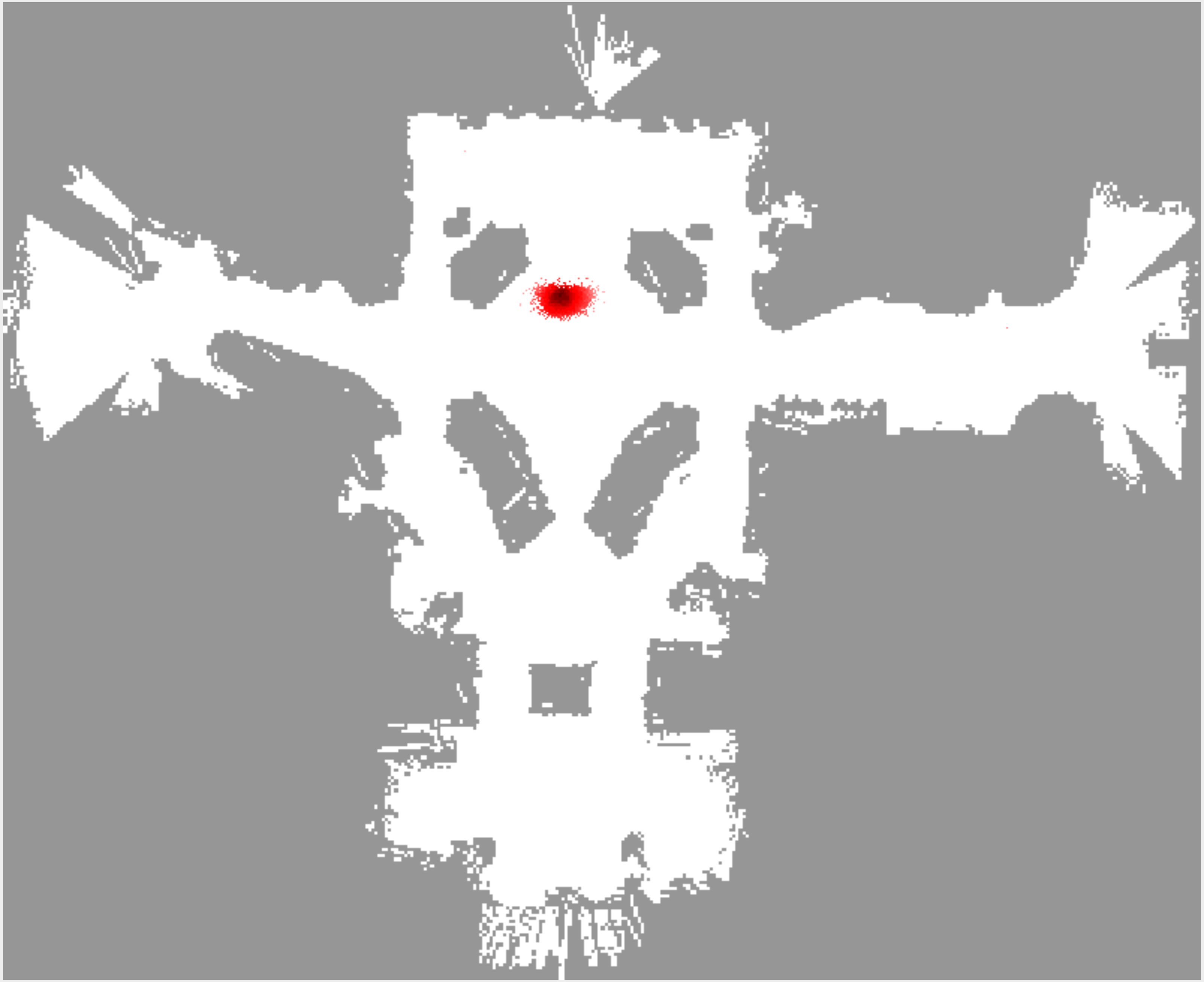


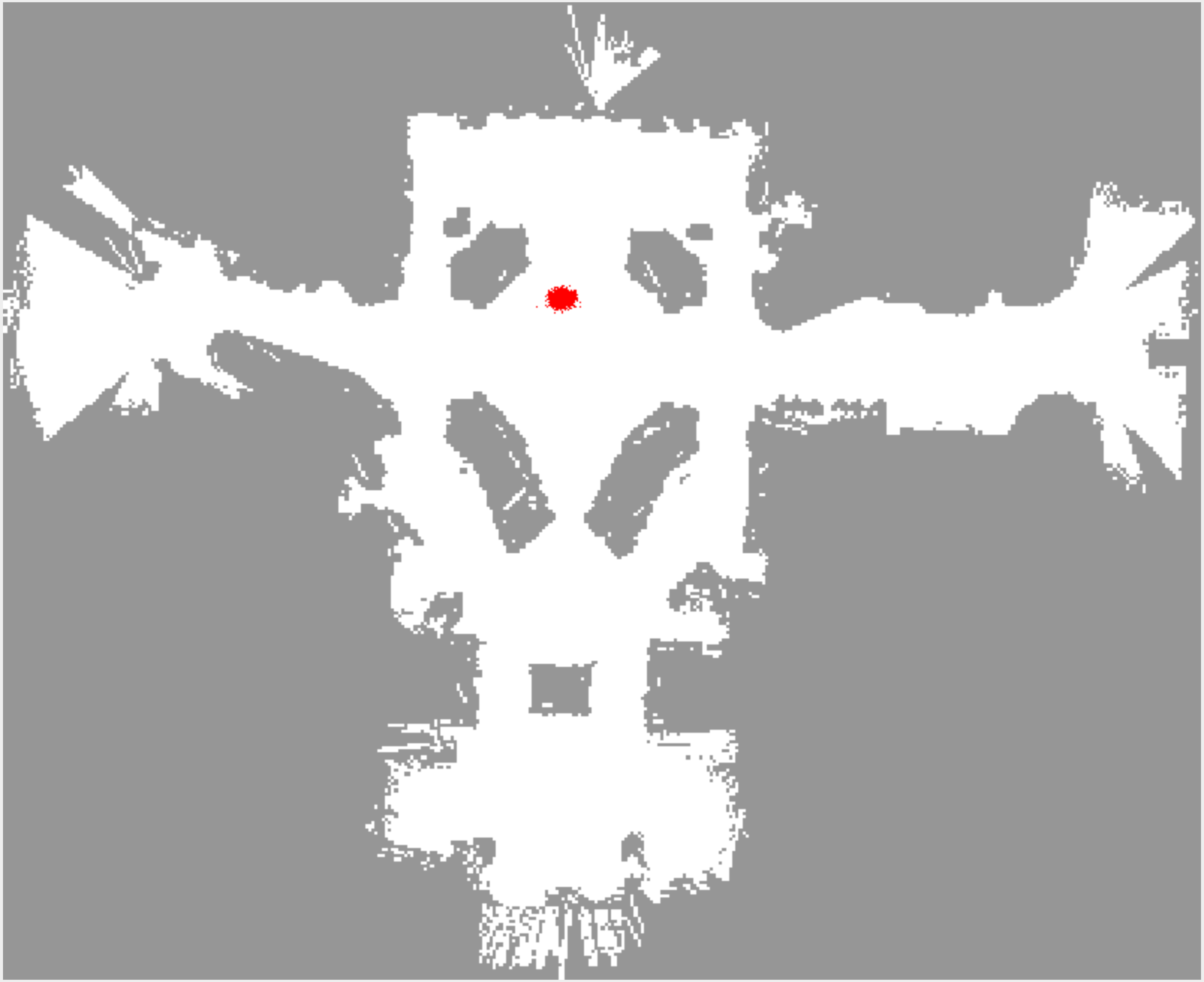


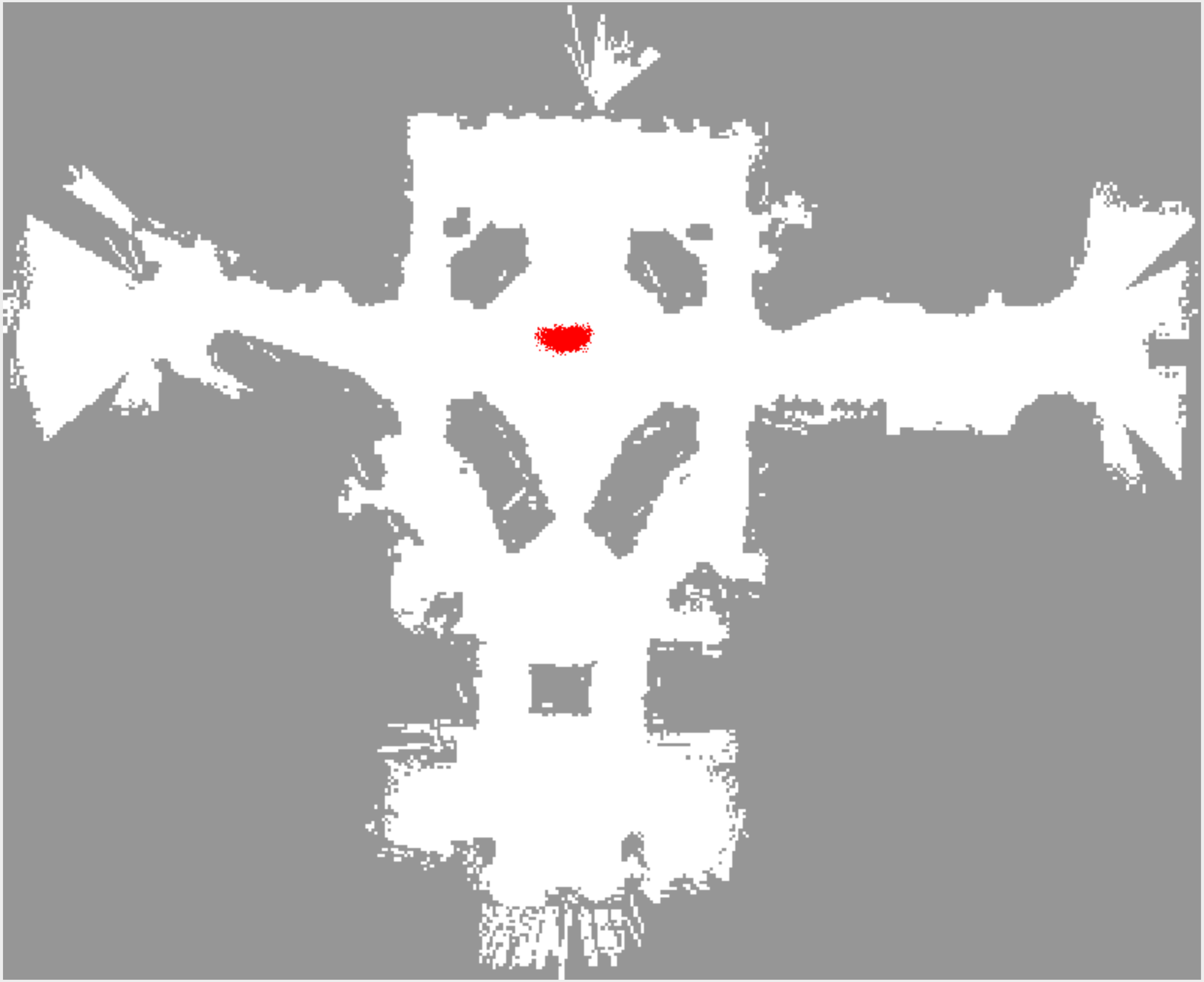


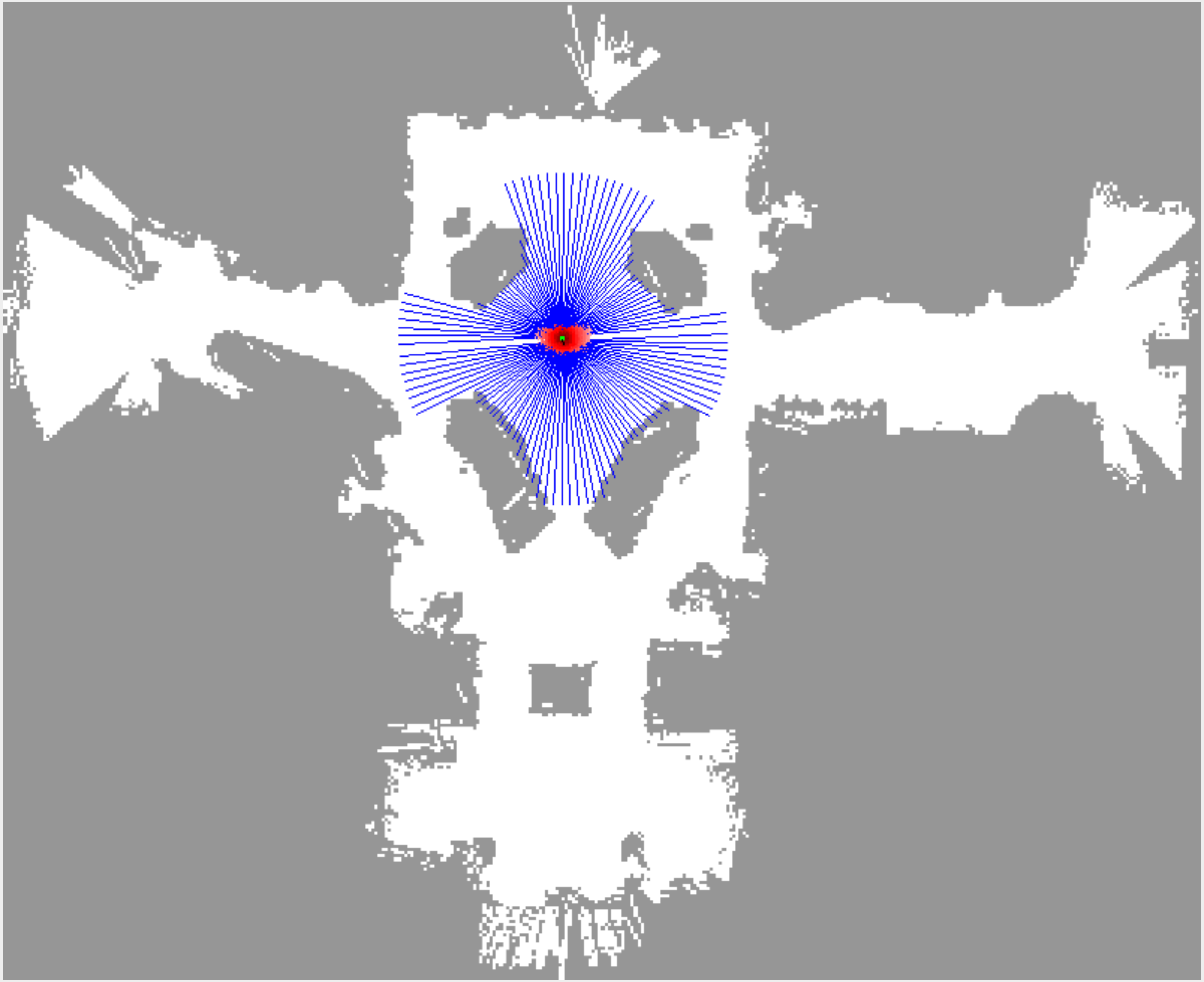


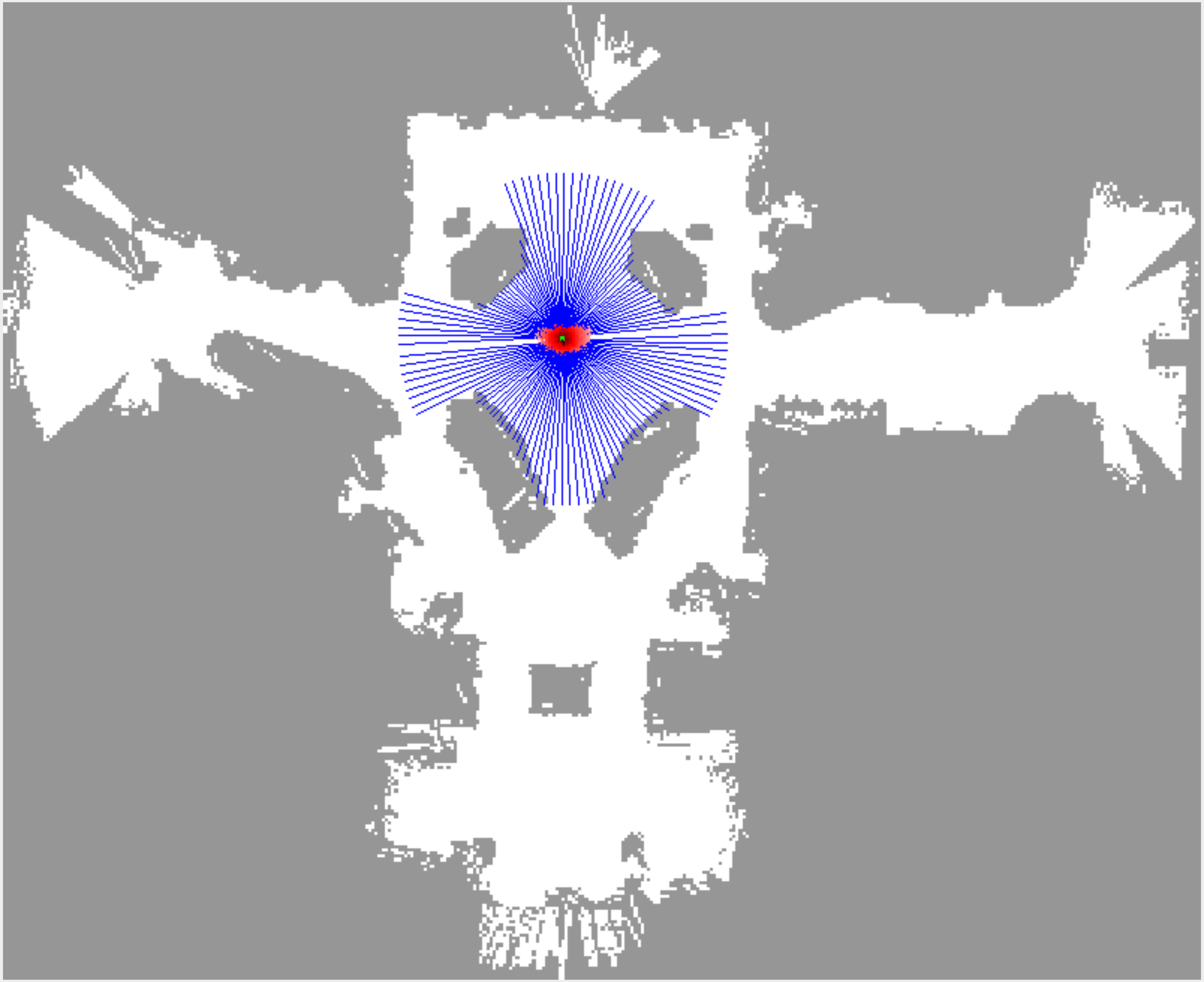




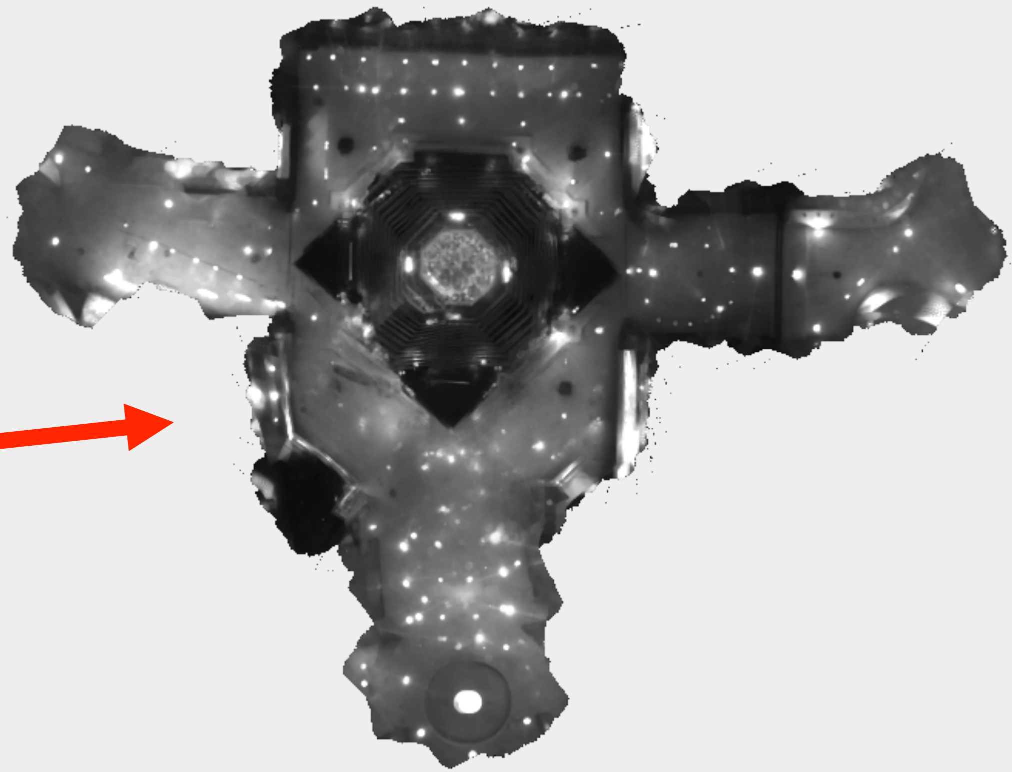






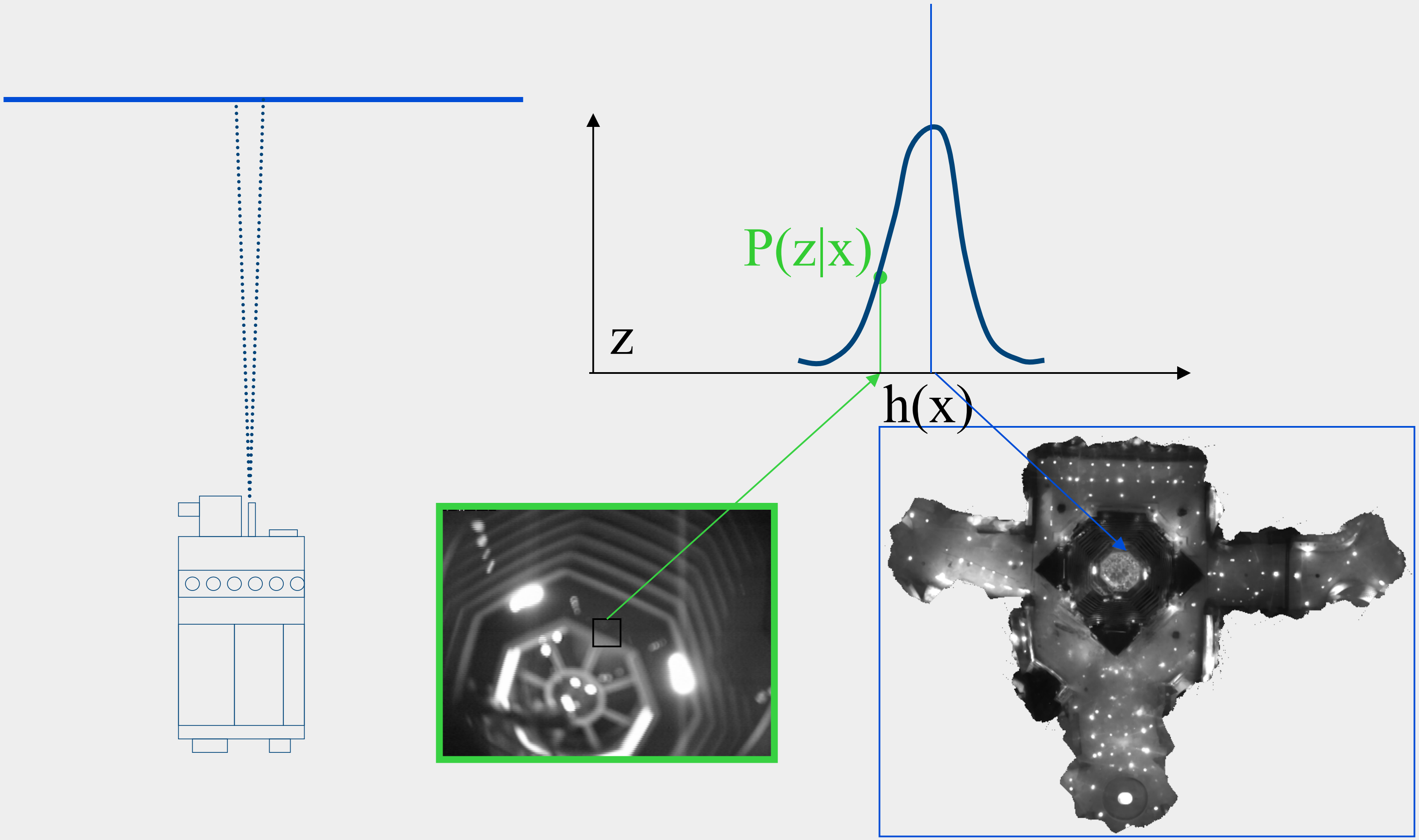


Using Ceiling Maps for Localization



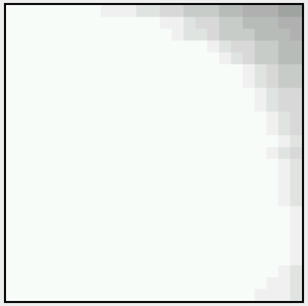
[Dellaert et al. 99]

Vision-based Localization



Under a Light

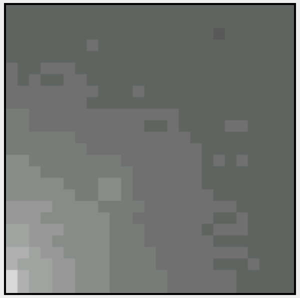
Measurement z:



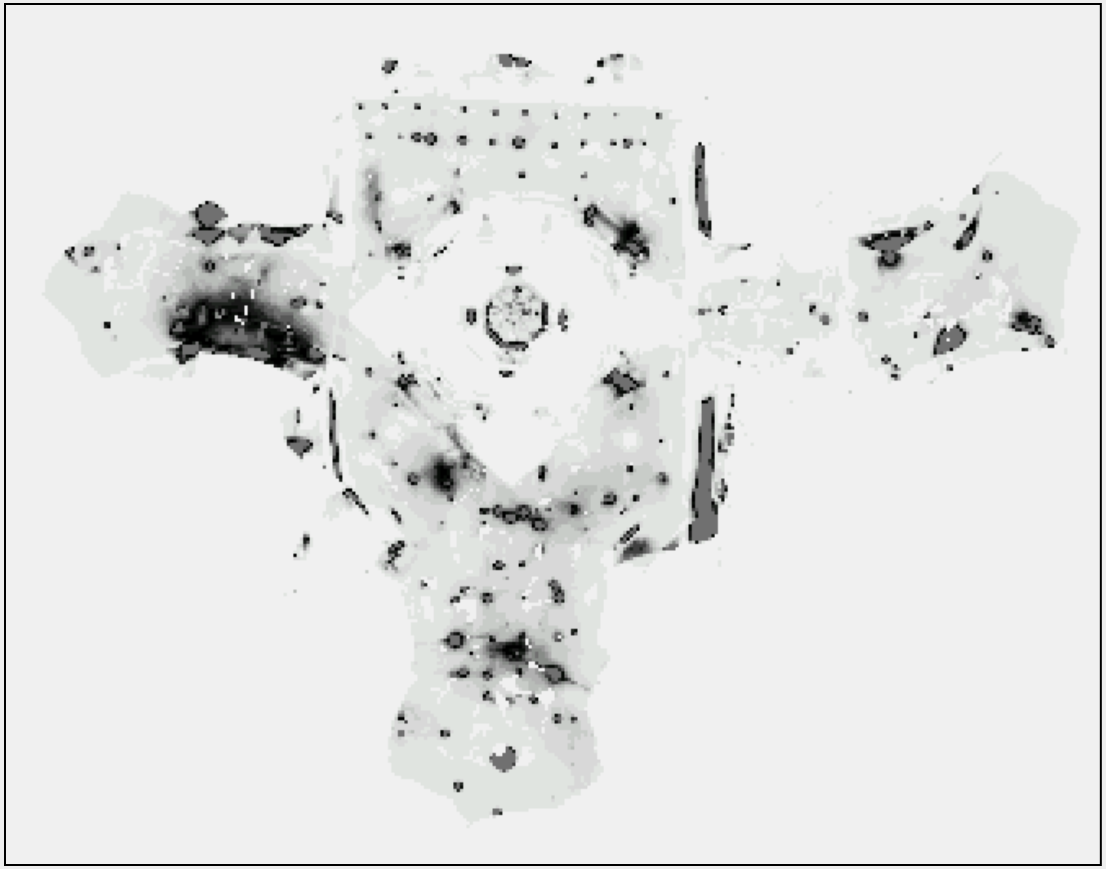
***P(z|x)*:**

Next to a Light

Measurement z:



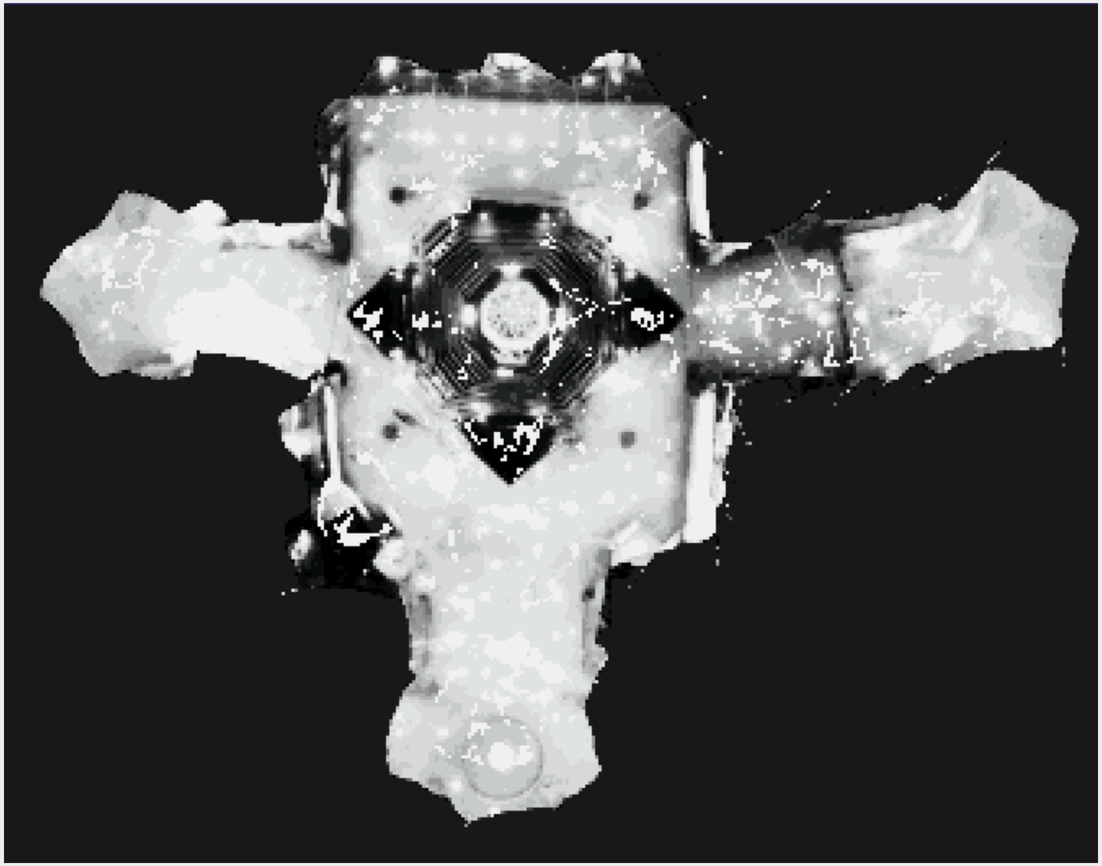
***P(z|x)*:**



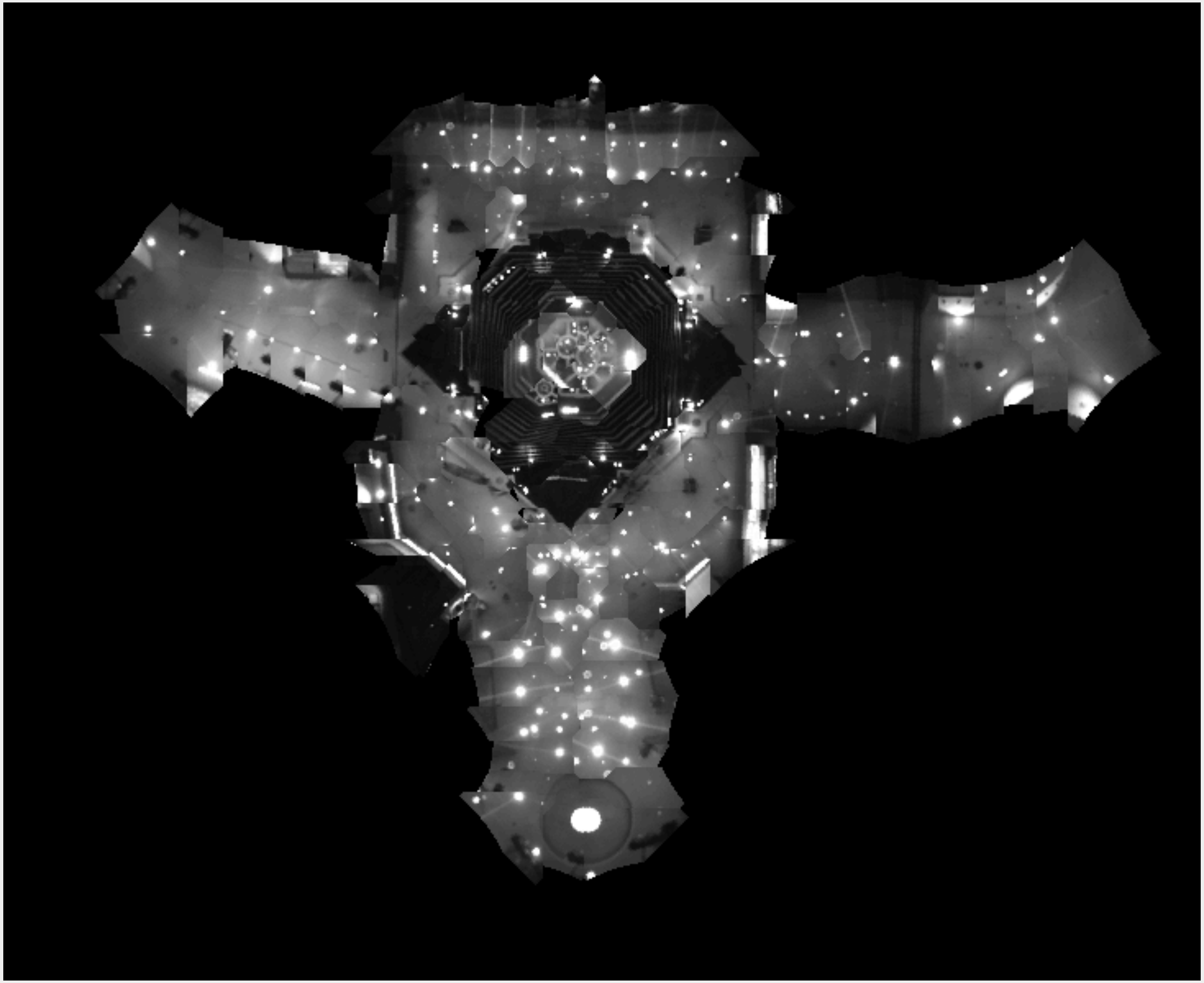
Elsewhere

Measurement z:

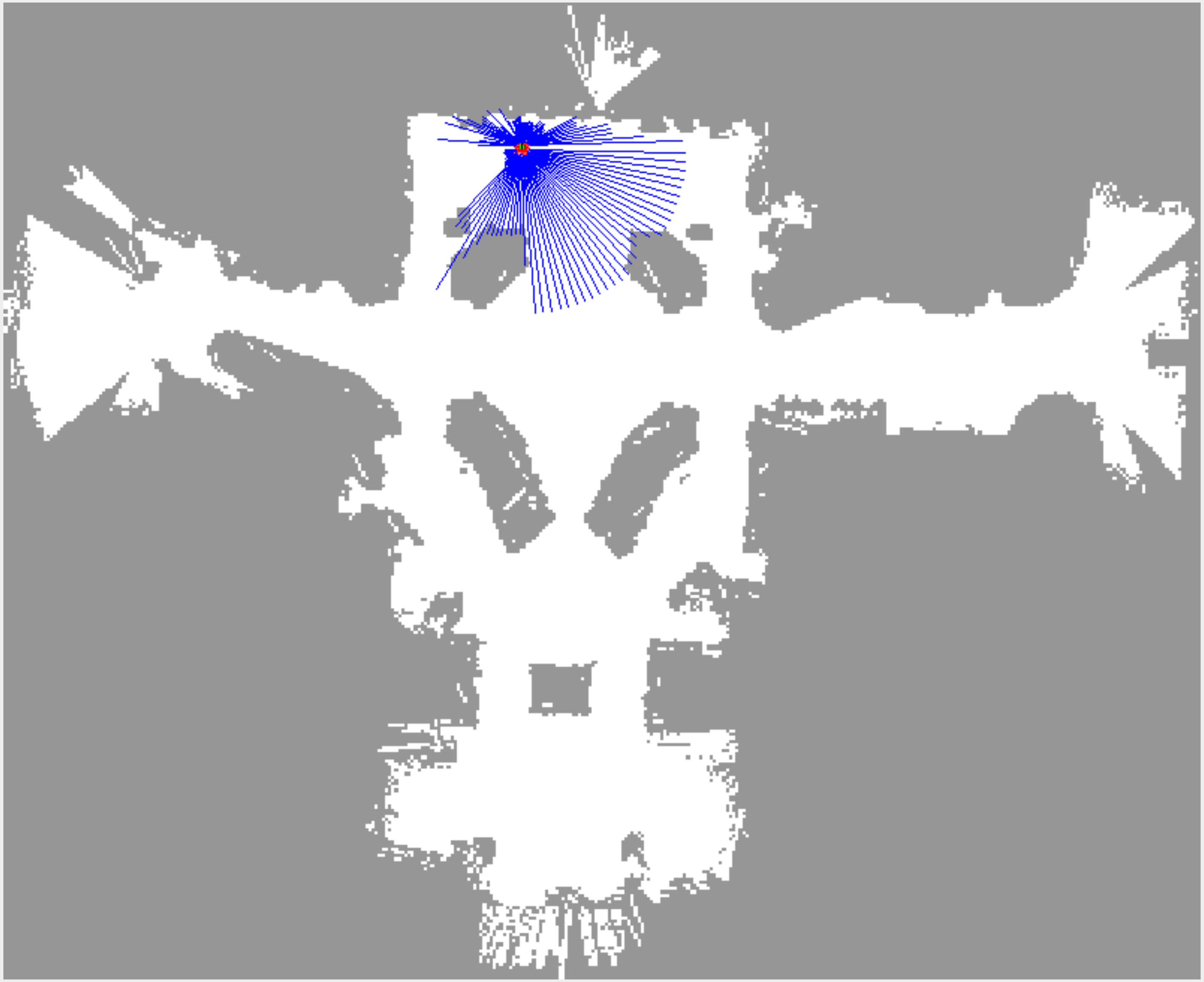
***P(z|x)*:**



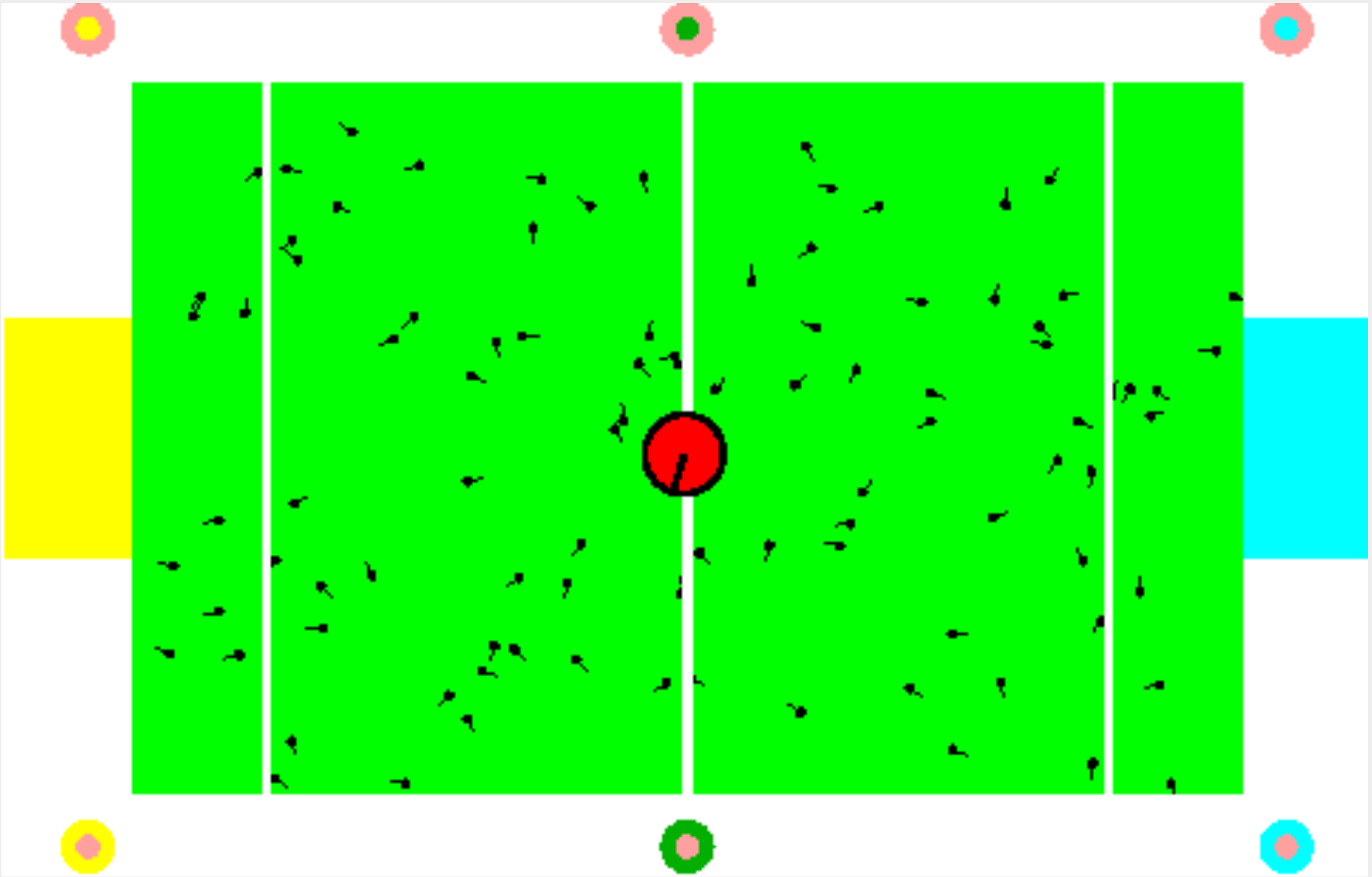
Global Localization Using Vision



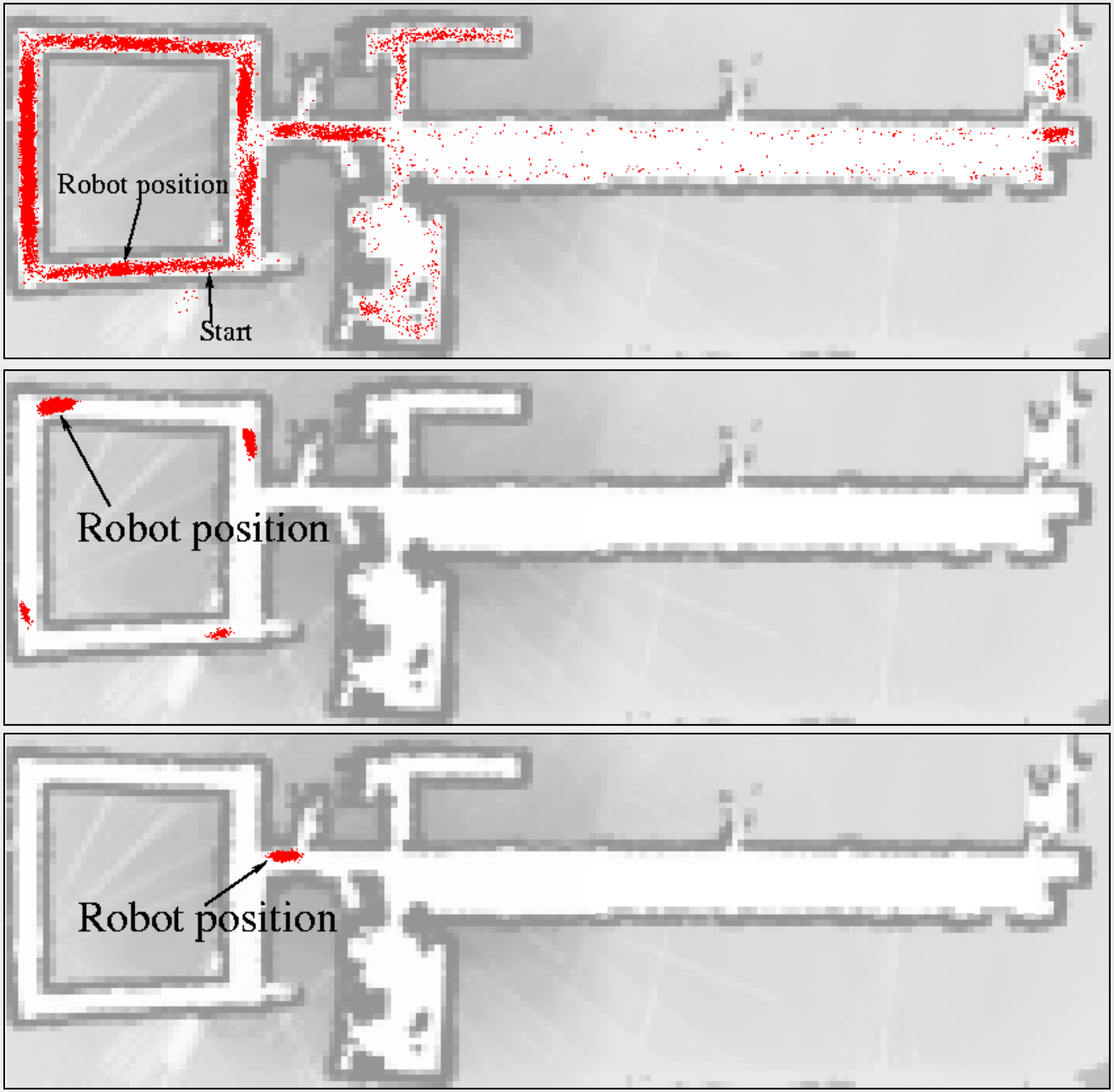
Recovery from Failure



Localization for AIBO robots



Adaptive Sampling



KLD-sampling

- **Idea**:
  - Assume we know the true belief.
  - Represent this belief as a multinomial distribution.
  - Determine number of samples such that we can guarantee that, with probability (1- d), the KL-distance between the true posterior and the sample-based approximation is less than *e*.

- **Observation**:
  - For fixed *d* and *e*, number of samples only depends on number k of bins with support:

$$n = \frac{1}{2\varepsilon}\mathrm{X}^2(k-1,1-\delta) \cong \frac{k-1}{2\varepsilon}\left\{1-\frac{2}{9(k-1)}+\sqrt{\frac{2}{9(k-1)}}\,z_{1-\delta}\right\}^3$$

Adaptive Particle Filter Algorithm

Algorithm **adaptive_particle_filter**(S_{t-1}, u_{t-1} z_t, $\Delta, \varepsilon, \delta$):

$S_t = \emptyset, \quad \alpha = 0, \quad n = 0, \quad k = 0, \quad b = \emptyset$

Do *Generate new samples*

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

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

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

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

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

If (x_t^n falls into an empty bin b) *Update bins with support*

$k=k+1$, b = *non-empty*

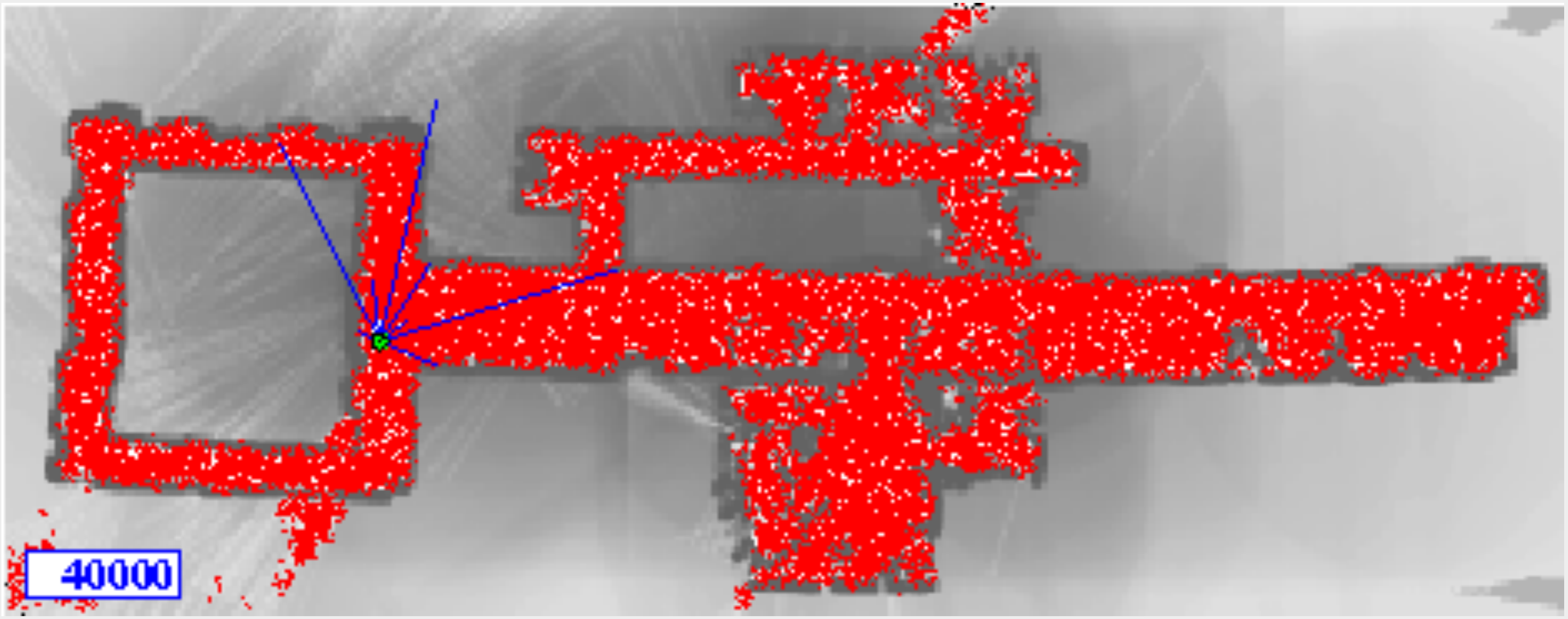
$n=n+1$

While ($n < \frac{1}{2\varepsilon} \mathrm{X}^2(k-1, 1-\delta)$)

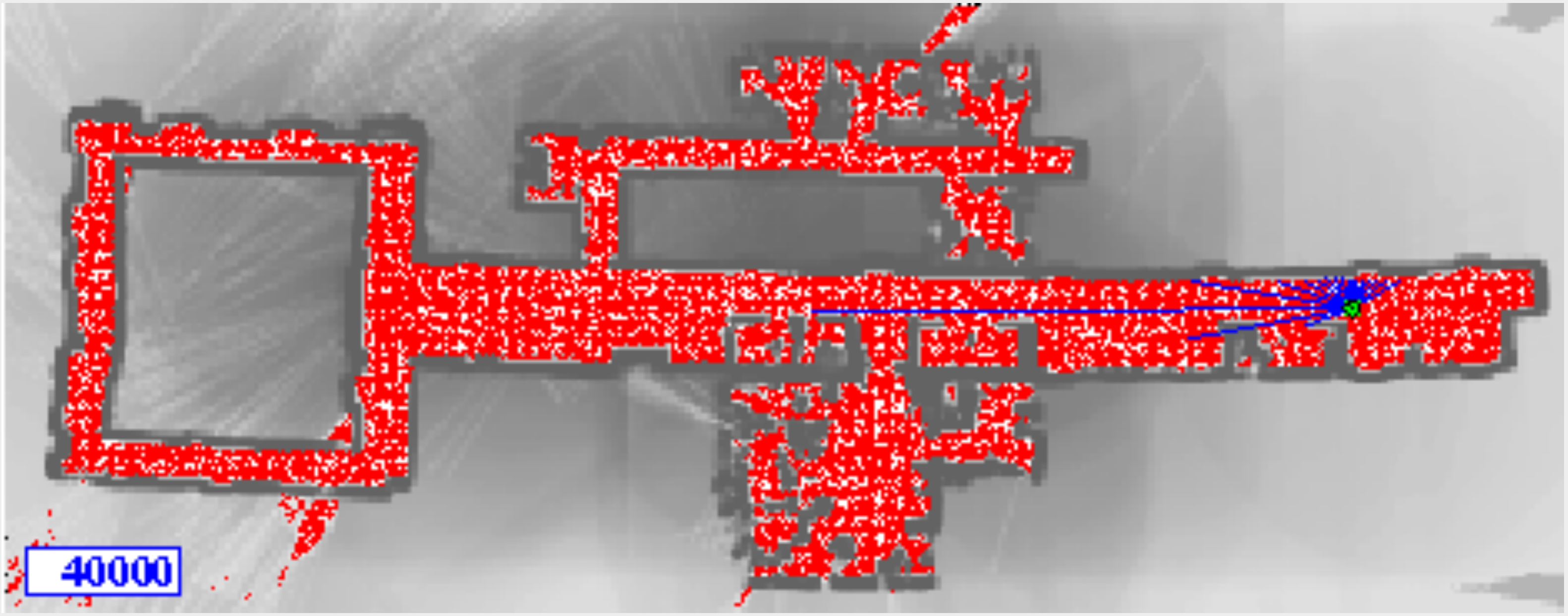
For $i = 1 \ldots n$

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

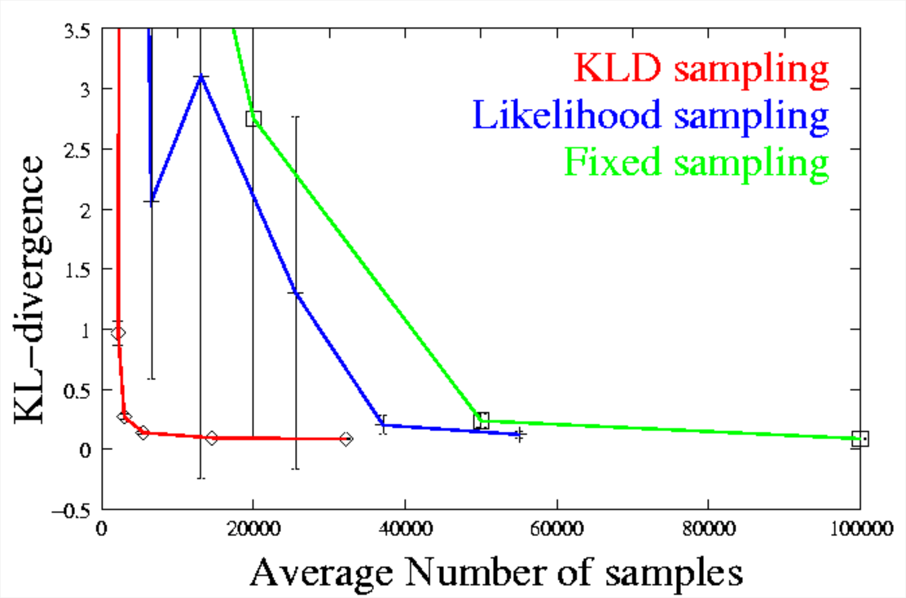
Example Run Sonar

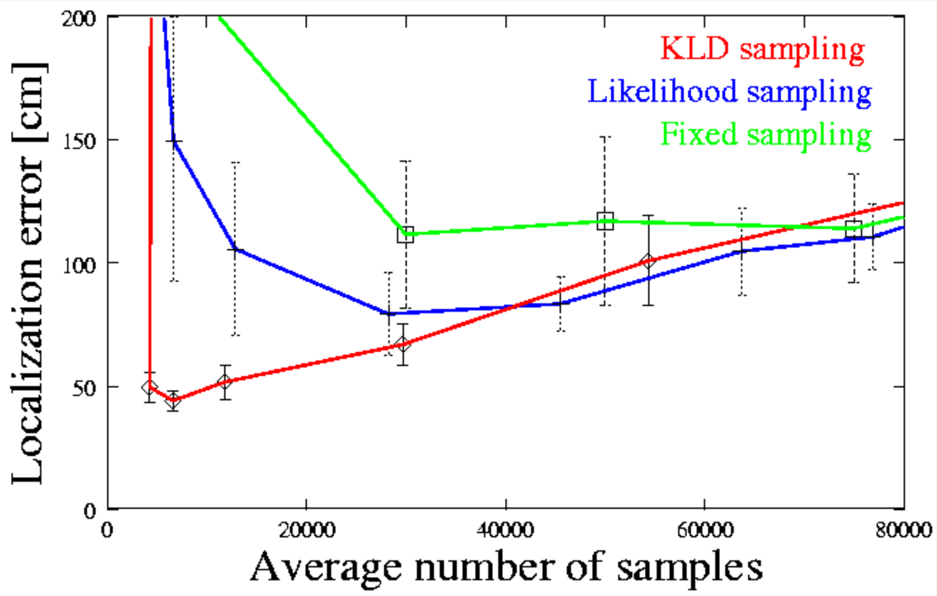


Example Run Laser



Evaluation


KLD sampling
Likelihood sampling
Fixed sampling
3.5
3
2.5
2
1.5
1
0.5
0
−0.5
KL−divergence
0
20000
40000
60000
80000
100000
Average Number of samples


KLD sampling
Likelihood sampling
Fixed sampling
200
150
100
50
0
Localization error [cm]
0
20000
40000
60000
80000
Average number of samples

Localization Algorithms - Comparison

	Kalman filter	Multi-hypothesis tracking	Topological maps	Grid-based (fixed/variable)	Particle filter
Sensors	Gaussian	Gaussian	Features	Non-Gaussian	Non-Gaussian
Posterior	Gaussian	Multi-modal	Piecewise constant	Piecewise constant	Samples
Efficiency (memory)	++	++	++	-/o	+/++
Efficiency (time)	++	++	++	o/+	+/++
Implementation	+	o	+	+/o	++
Accuracy	++	++	-	+/++	++
Robustness	-	+	+	++	+/++
Global localization	No	Yes	Yes	Yes	Yes