Principles of Safe Autonomy: Lecture 12-13: Filtering and Robot Localization

Sayan Mitra

2023

Reference: Probabilistic Robotics by Sebastian Thrun, Wolfram Burgard, and Dieter Fox

Slides: From the book's website

Announcements

- Exam1 Regrade & Resubmit due: 10/13
- Pitch Presentation: 10/24, 10/26
- MP3 Due: 10/27
- For GEM and F1-Tenth groups, finish 3 Safety Training Online & upload safety training certificate to box ASAP
- More details on campuswire

Review from last time: Beliefs

Belief: Robot's knowledge about the state of the environment

True state is unknowable / measurable typically, so, robot must infer state from data and we have to distinguish this inferred/estimated state from the actual state x_t

$bel(x_t) = p(x_t | z_{1:t}, u_{1:t})$

Posterior distribution over state at time t given all past measurements and control.

This will be calculated in two steps:

1. Prediction: $\overline{bel}(x_t) = p(x_t | z_{1:t-1}, u_{1:t})$
2. Correction: Calculating $bel(x_t)$ from $\overline{bel}(x_t)$ a.k.a measurement update

Recursive Bayes Filter

Algorithm Bayes_filter($bel(x_{t-1}), u_t, z_t$)

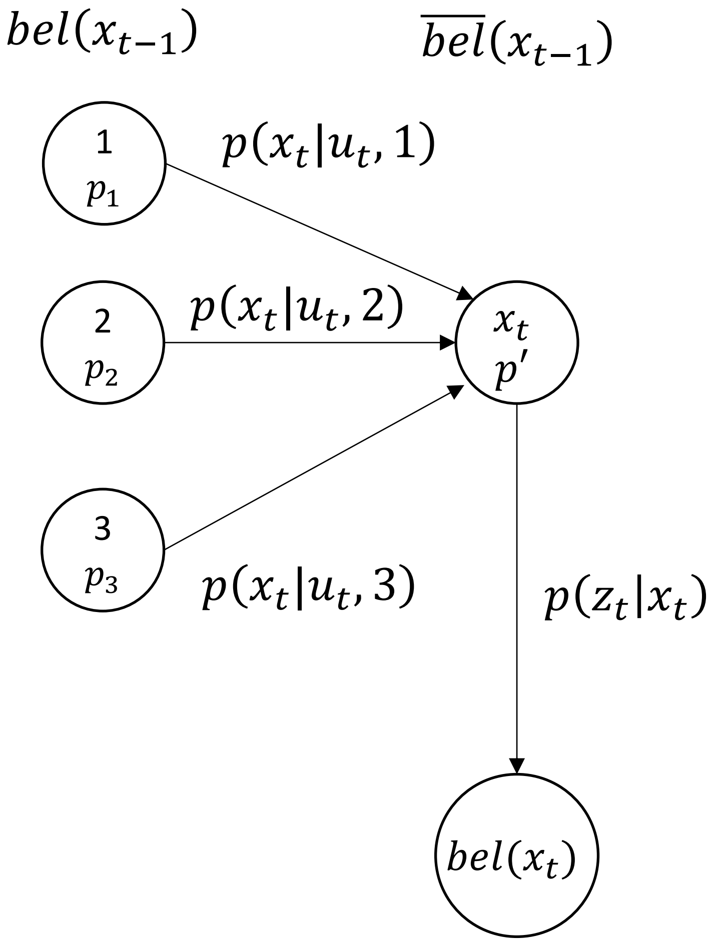
for all x_t do:

$$\overline{bel}(x_t) = \int p(x_t|u_{t,}x_{t-1})bel(x_{t-1})dx_{t-1}$$

$$bel(x_t) = \eta\, p(z_t|x_t)\, \overline{bel}(x_t)$$

end for

return $bel(x_t)$



Histogram Filter or Discrete Bayes Filter

Finitely many states $x_i, x_k, etc.$ Random state vector X_t

$p_{k,t}$: belief at time t for state x_k; discrete probability distribution

Algorithm Discrete_Bayes_filter($\{p_{k,t-1}\}, u_t, z_t$**):**

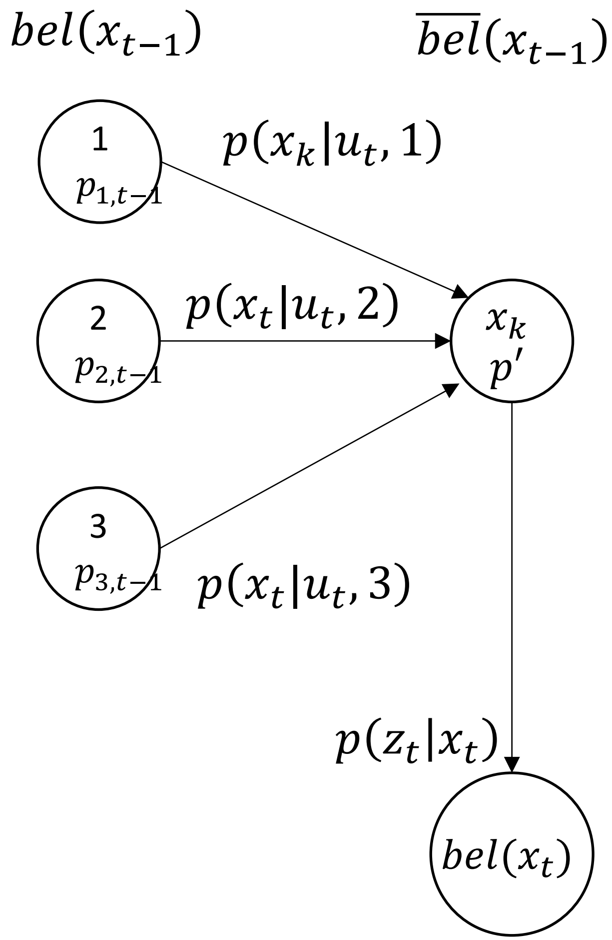
for all k do:

$$\bar{p}_{k,t} = \sum_i p(X_t = x_k | u_t, X_{t-1} = x_i) p_{i,t-1}$$

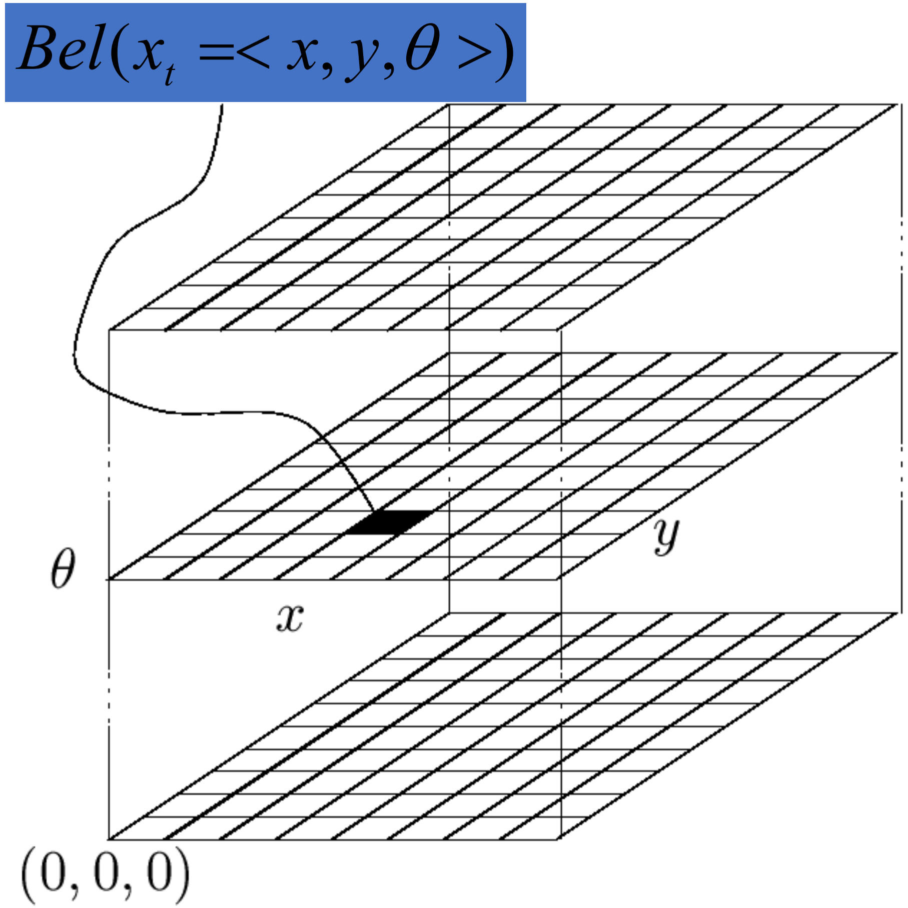
$$p_{k,t} = \eta \, p(z_t \, | X_t = x_k) \bar{p}_{k,t}$$

end for

return $\{p_{k,t}\}$



Piecewise Constant Representation of beliefs

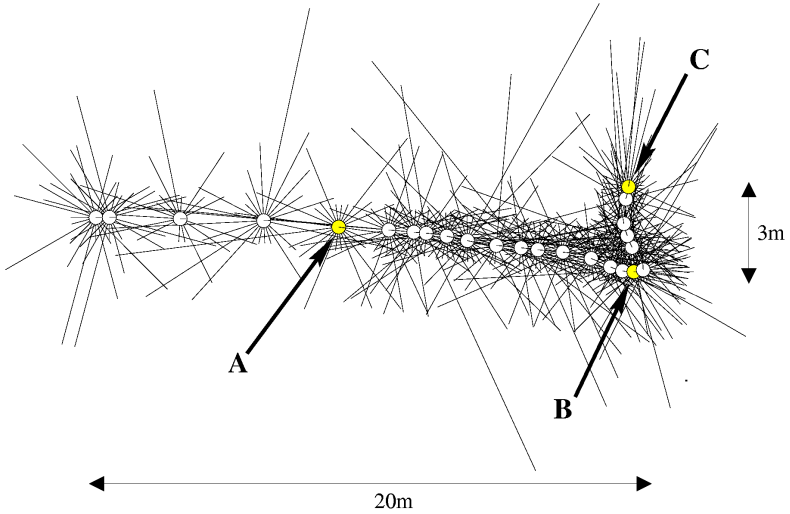


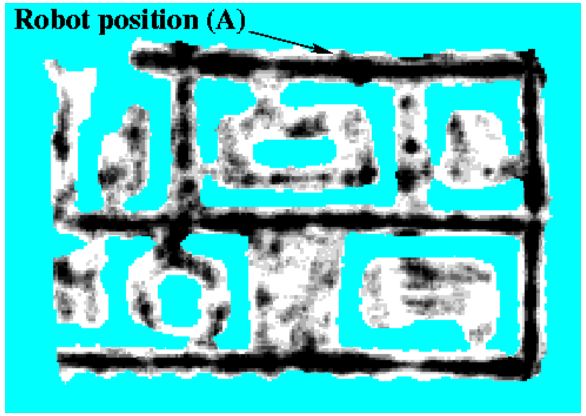
Fixing an input u_t we can compute the new belief

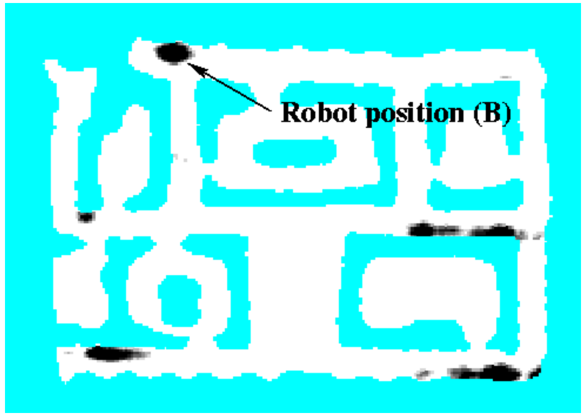
Outline of filtering module

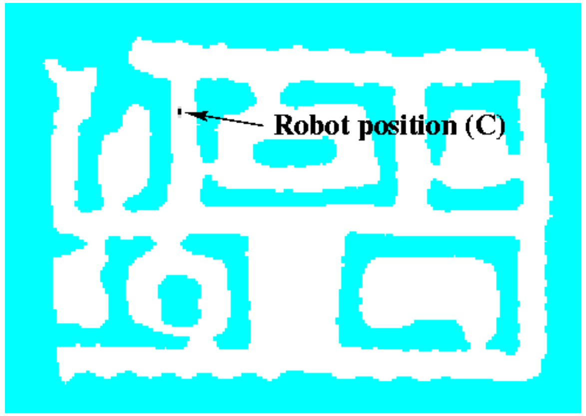
- Particle filter
  - Nonparametric representation of distributions with samples
  - Weighted particles
  - Importance sampling
- Monte Carlo localization
- Examples
- Conclusions

Sonars & Occupancy Grid Map









Monte Carlo Localization

Represents beliefs by particles

Particle Filters

- Represent belief by finite number of parameters (just like histogram filter)
- But, they differ in how the parameters (particles) are generated and populate the state space
- Key idea: represent belief $bel(x_t)$ by a random set of state samples
- Advantages
  - The representation is approximate and nonparametric and therefore can represent a broader set of distributions than e.g., Gaussian
  - Can handle nonlinear tranformations
- Related ideas: Monte Carlo filter, Survival of the fittest, Condensation, Bootstrap filter, Filtering: [Rubin, 88], [Gordon et al., 93], [Kitagawa 96], Dynamic Bayesian Networks: [Kanazawa et al., 95]d

Particle filtering algorithm

$X_t = x_t^{[1]}, x_t^{[2]}, \ldots x_t^{[M]}$ particles

Algorithm Particle_filter(X_{t-1}, u_t, z_t):
$\bar{X}_{t-1} = X_t = \emptyset$

for all m in [M] do:

  sample $x_t^{[m]} \sim p(x_t | u_t, x_{t-1}^{[m]})$

  $w_t^{[m]} = p\left(z_t \middle| x_t^{[m]}\right)$

  $\bar{X}_t = \bar{X}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$

end for

for all m in [M] do:

  draw i $with\ probability \propto w_t^{[i]}$

  add $x_t^{[i]}$ to X_t

end for

return X_t

ideally, $x_t^{[m]}$ is selected with probability prop. to $p(x_t \mid z_{1:t}, u_{1:t})$

$\bar{X}_{t-1}$ is the temporary particle set

// sampling from state transition dist.

// calculates *importance factor* w_t or weight

// resampling or importance sampling; these are distributed according to $\eta\, p\left(z_t \middle| x_t^{[m]}\right) \overline{bel}(x_t)$

// survival of fittest: moves/adds particles to parts of the state space with higher probability

Importance Sampling

suppose we want to compute $E_f[I(x \in A)]$ but we can only sample from density g

$E_f[I(x \in A)]$

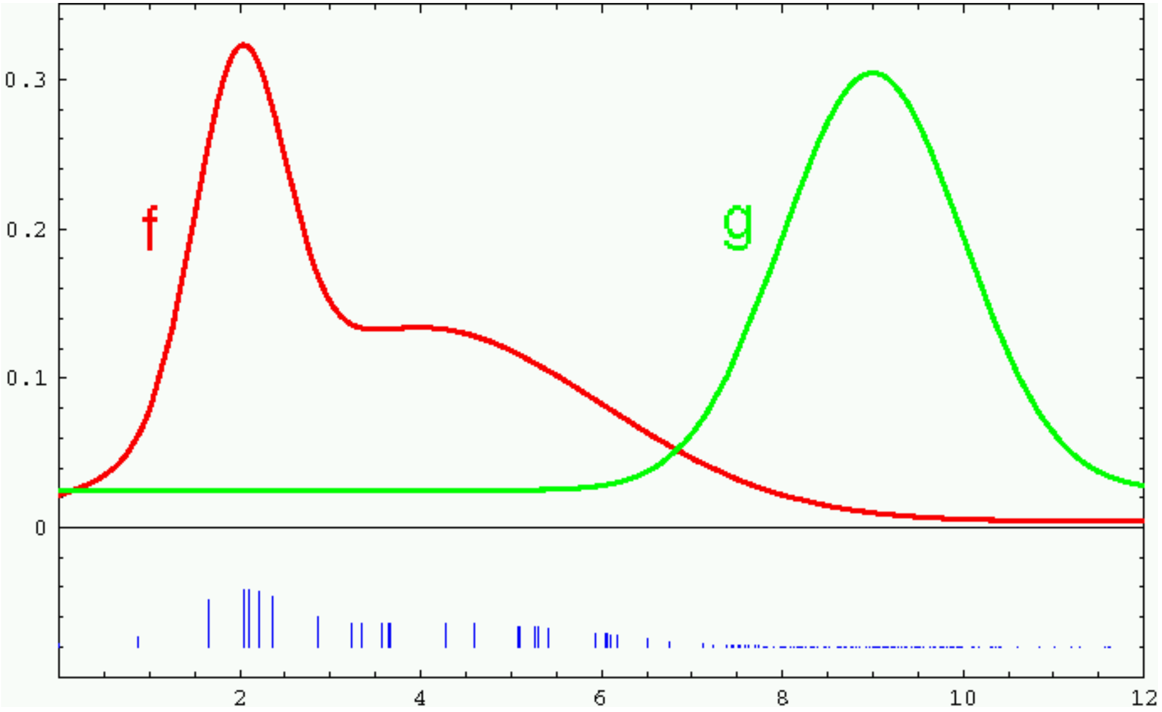
$= \int f(x) I(x \in A) dx$

$= \int \frac{f(x)}{g(x)} g(x) I(x \in A) dx$, provided $g(x) > 0$

$= \int w(x) g(x) I(x \in A) dx$

$= E_g[w(x) I(x \in A)]$

We need $f(x) > 0 \Rightarrow g(x) > 0$



Weight samples: $w = f / g$

Monte Carlo Localization (MCL)

$X_t = x_t^{[1]}, x_t^{[2]}, \ldots x_t^{[M]}$ particles

Algorithm MCL(X_{t-1}, u_t, z_t,m):

$\bar{X}_{t-1} = X_t = \emptyset$

for all m in [M] do:

$$x_t^{[m]} = \boldsymbol{sample\_motion\_model}(u_t\ x_{t-1}^{[m]})$$

$$w_t^{[m]} = \boldsymbol{measurement\_model}(z_t, x_t^{[m],m})$$

$$\bar{X}_t = \bar{X}_t + \langle\, x_t^{[m]}, w_t^{[m]} \rangle$$

end for

for all m in [M] do:

draw i $with\ probability\ \propto w_t^{[i]}$

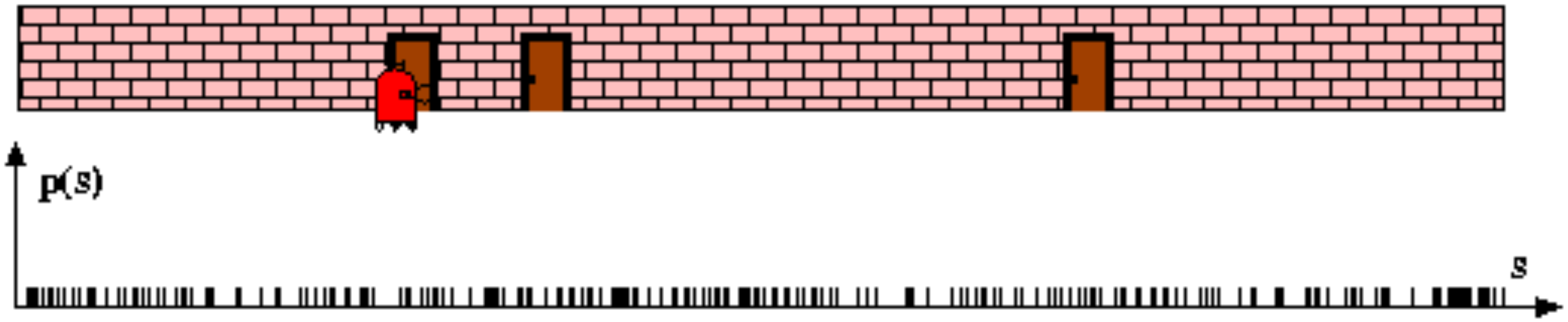
add $x_t^{[i]}$ $to\ X_t$

end for

return X_t

Plug in motion and measurement models in the particle filter

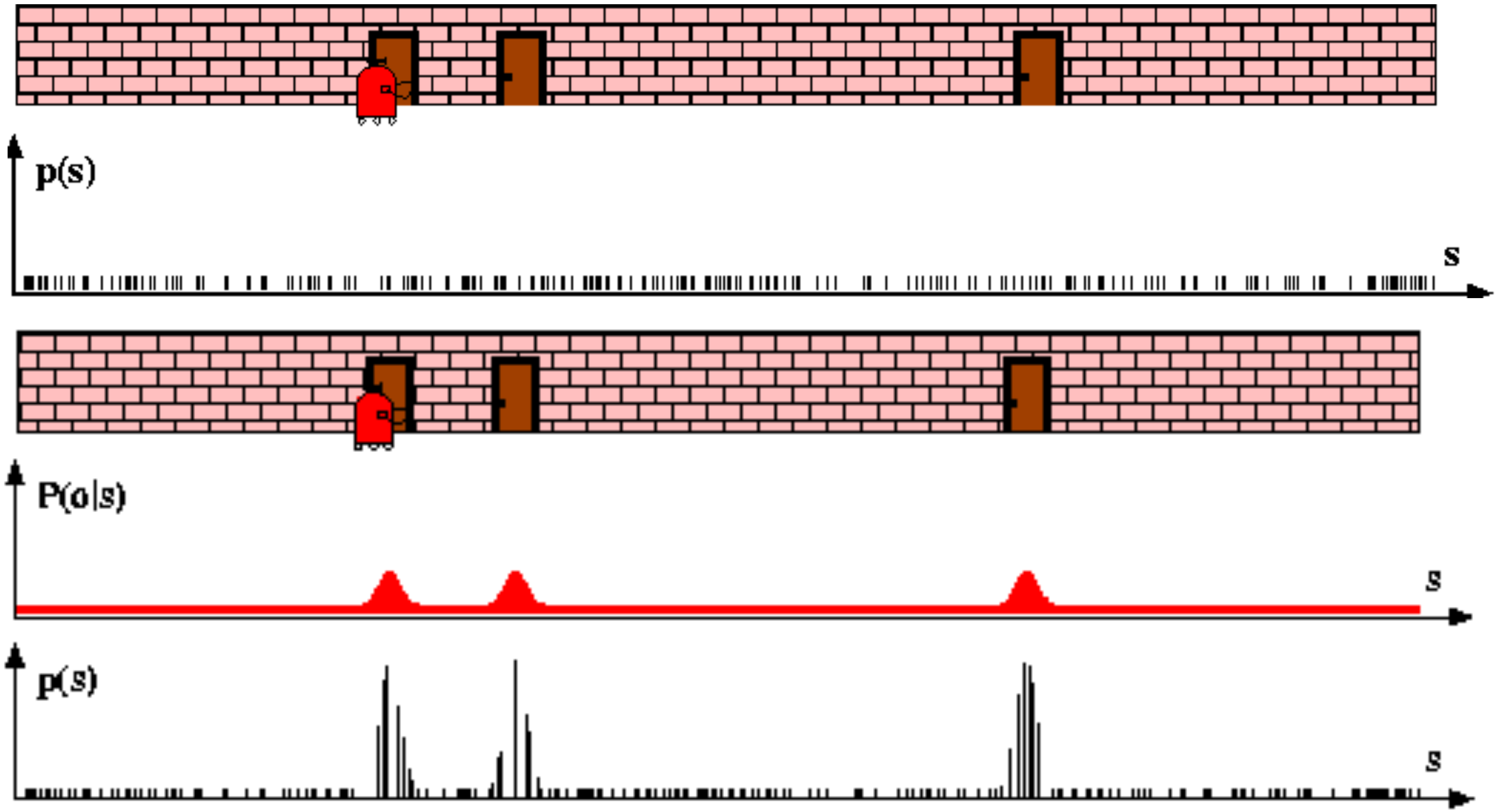
Particle Filters



Sensor Information: Importance Sampling

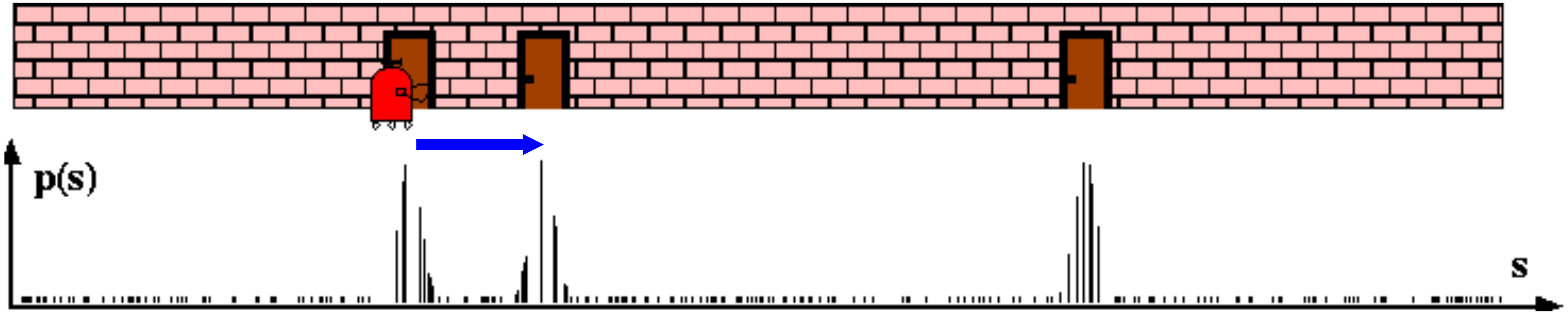
$$Bel(x) \leftarrow \alpha\ p(z\,|\,x)\, Bel^{-}(x)$$

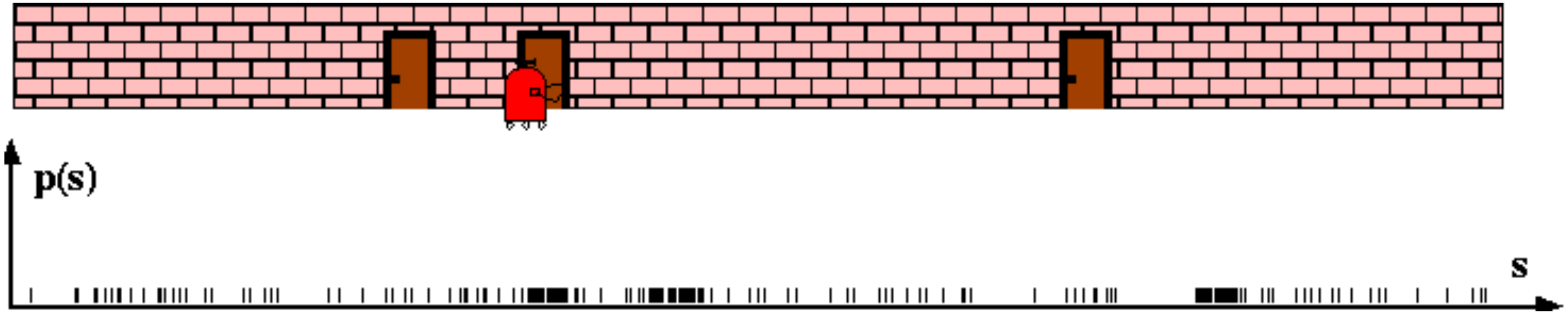
$$w \leftarrow \frac{\alpha\ p(z\,|\,x)\, Bel^{-}(x)}{Bel^{-}(x)} = \alpha\ p(z\,|\,x)$$



Robot Motion

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

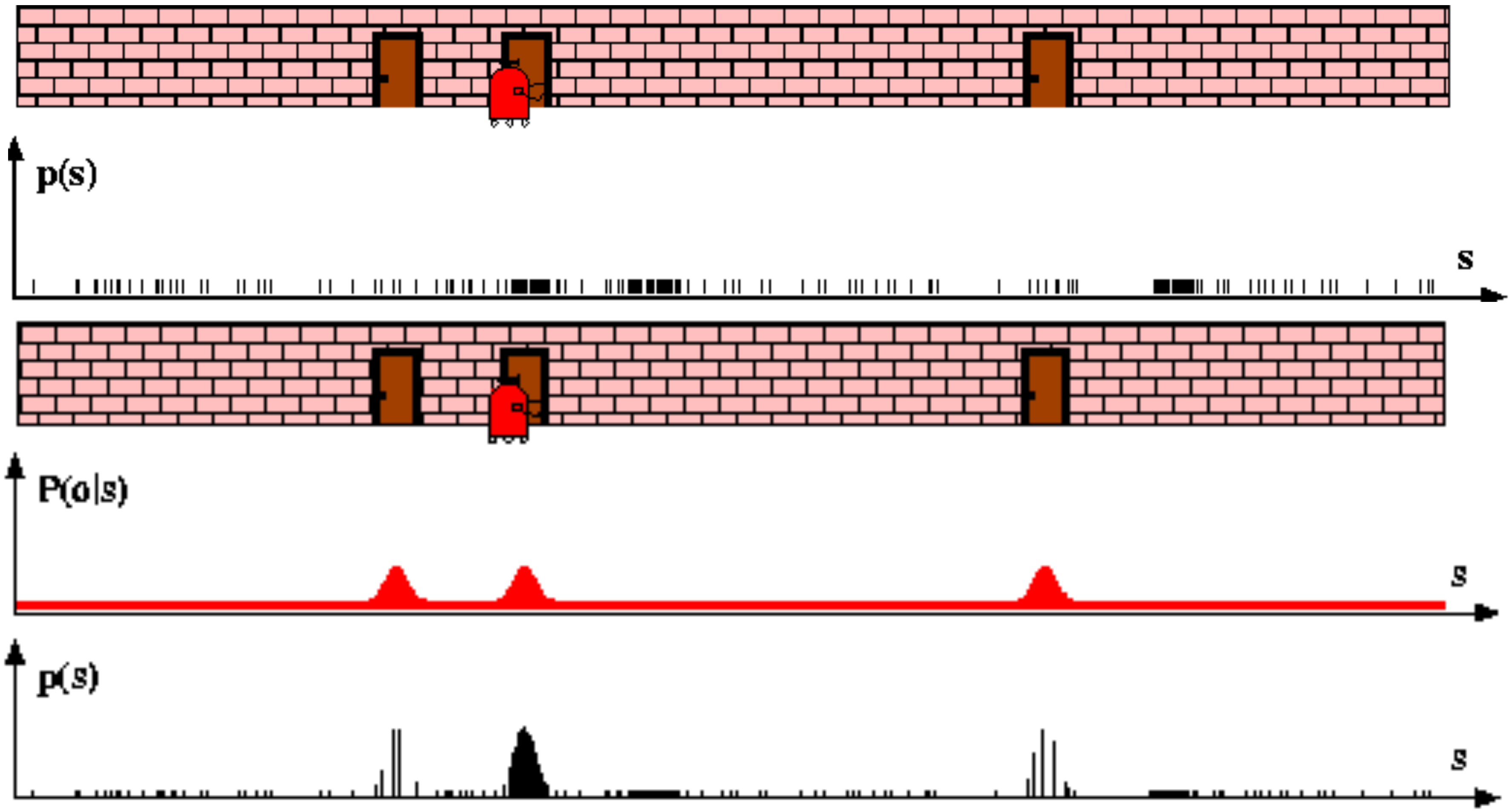




Sensor Information: Importance Sampling

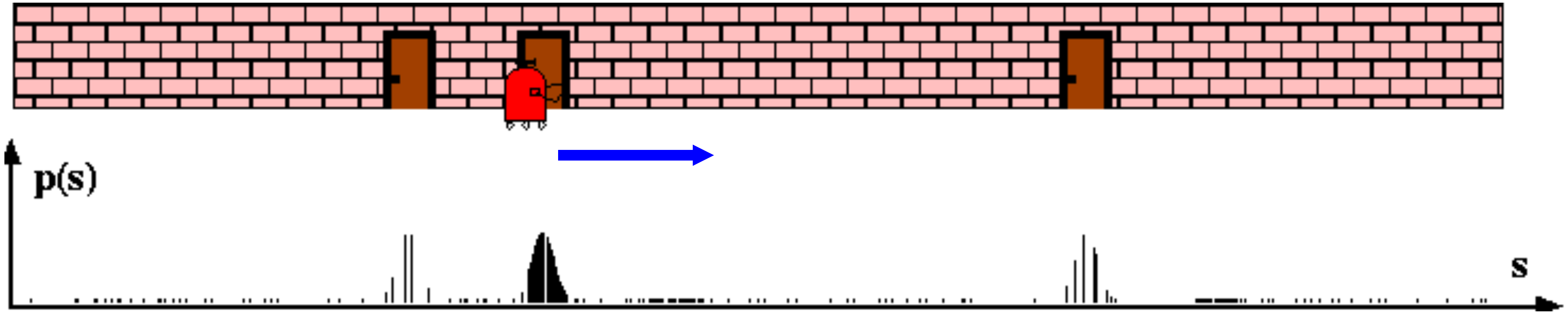
$$Bel(x) \leftarrow \alpha\, p(z|x)\, Bel^-(x)$$

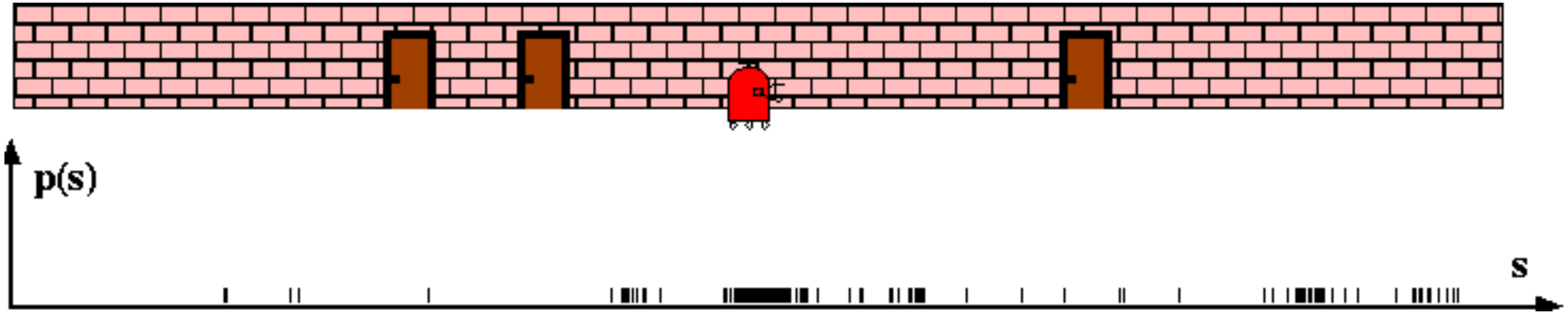
$$w \leftarrow \frac{\alpha\, p(z|x)\, Bel^-(x)}{Bel^-(x)} = \alpha\, p(z|x)$$

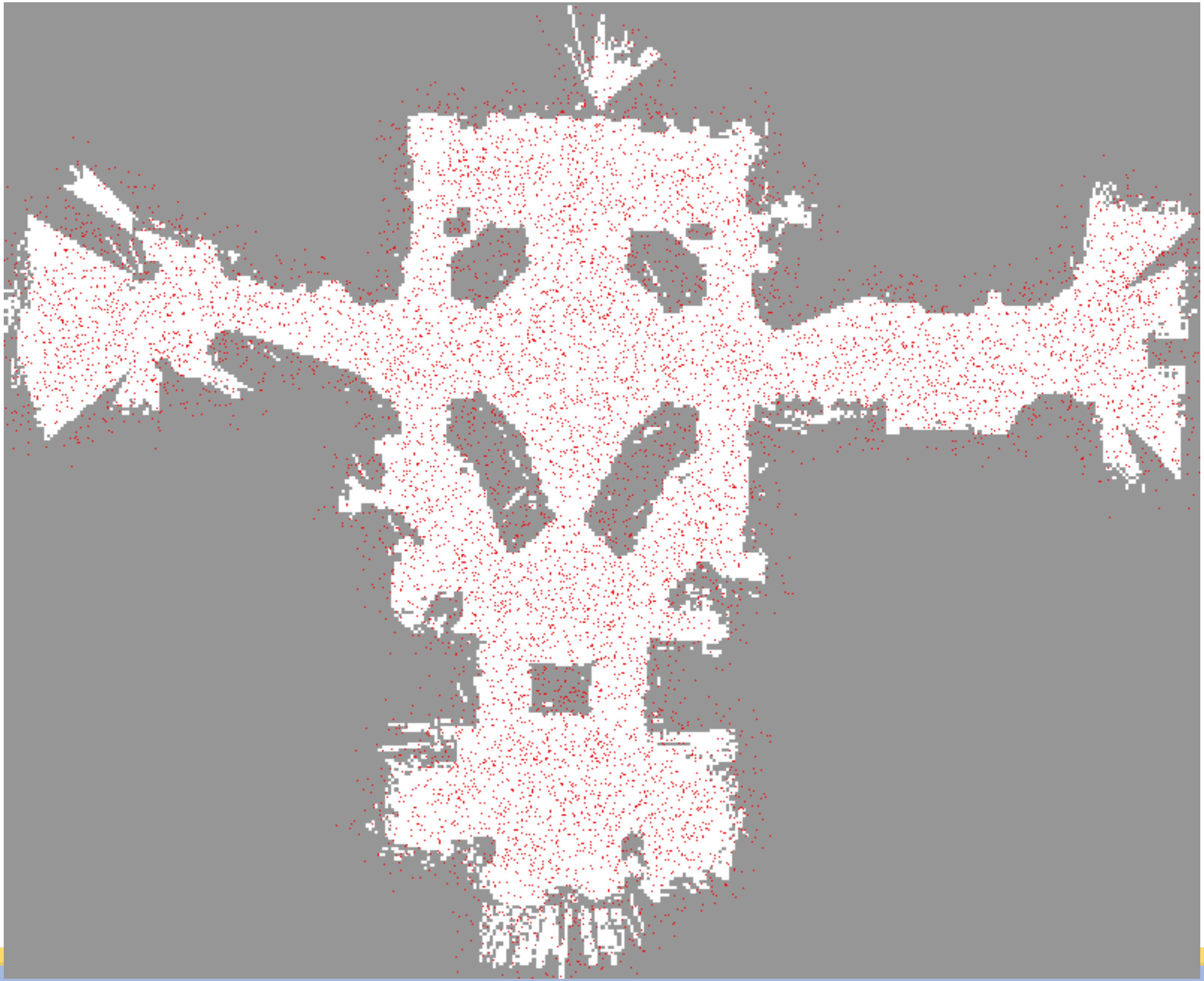


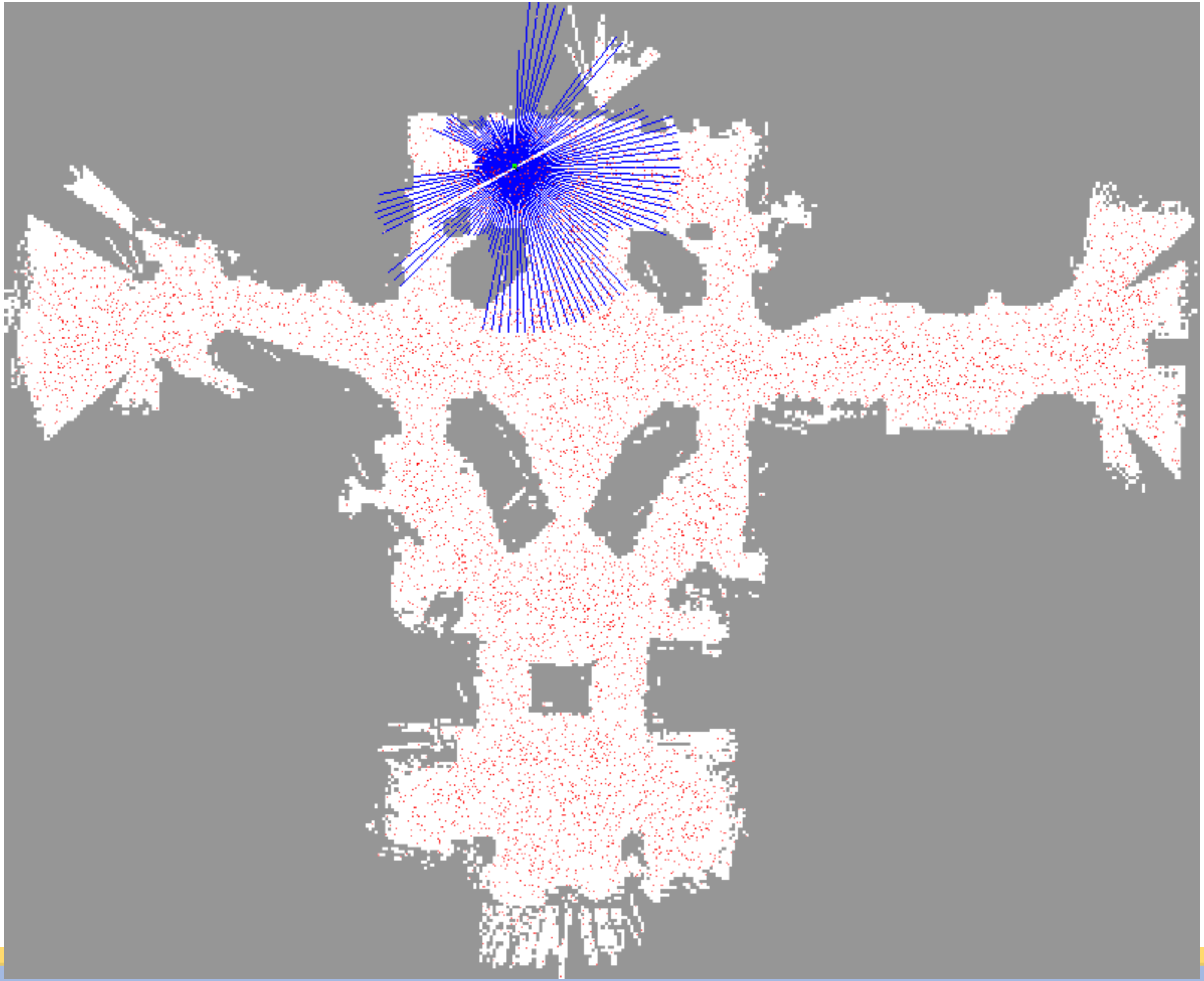
Robot Motion

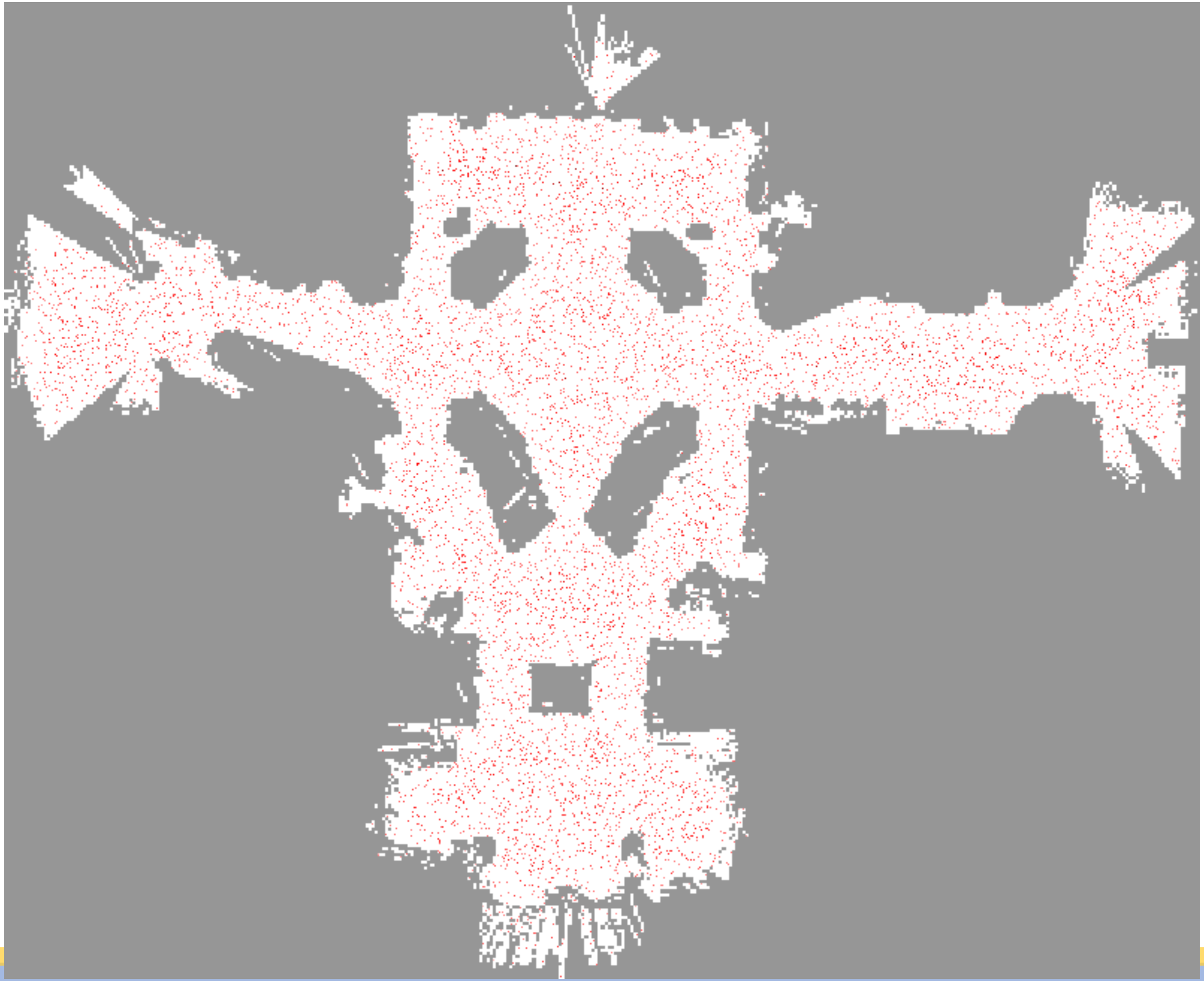
$$Bel^{-}(x) \leftarrow \int p(x \mid u, x')\, Bel(x')\, \mathrm{d}x'$$

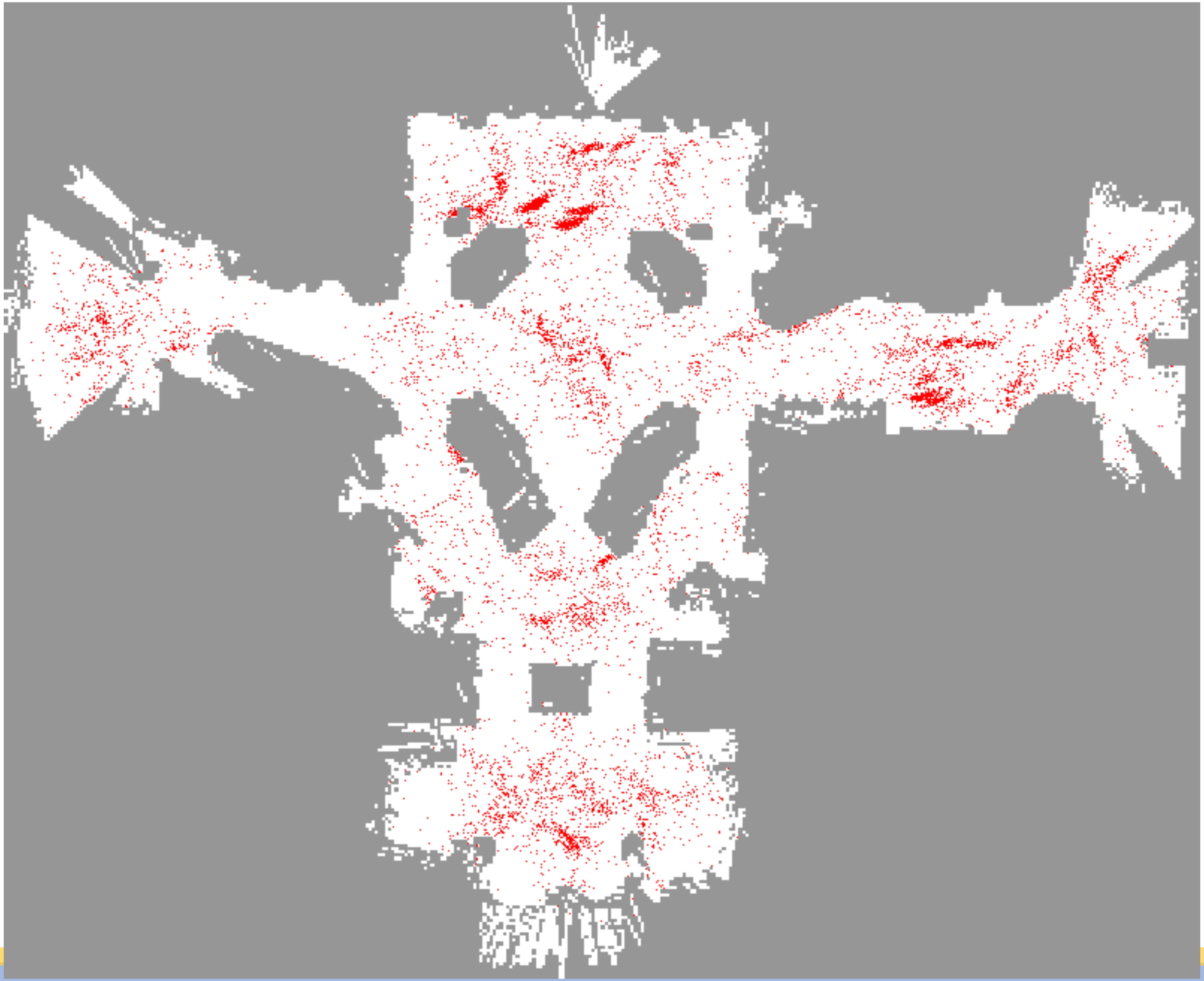


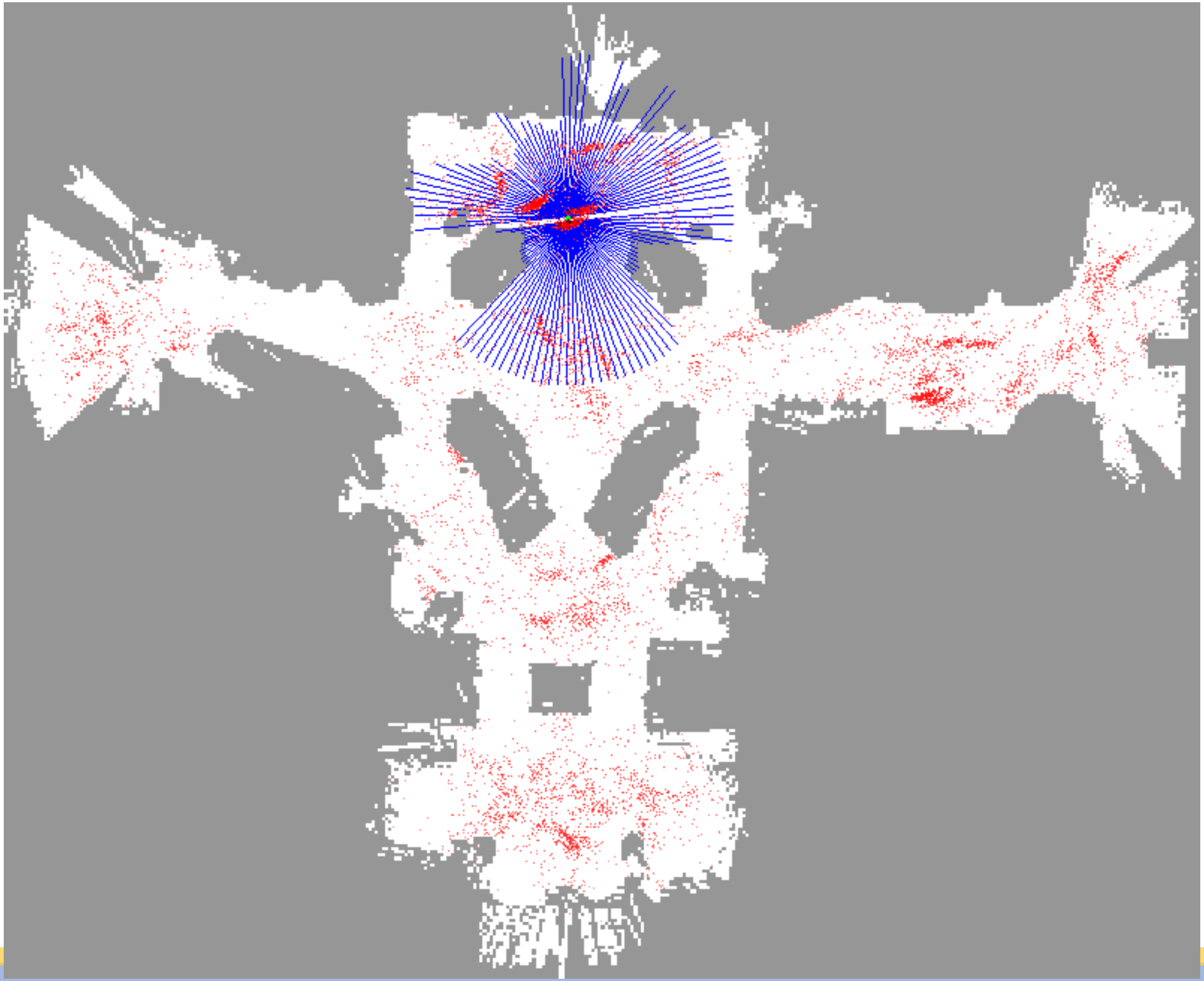


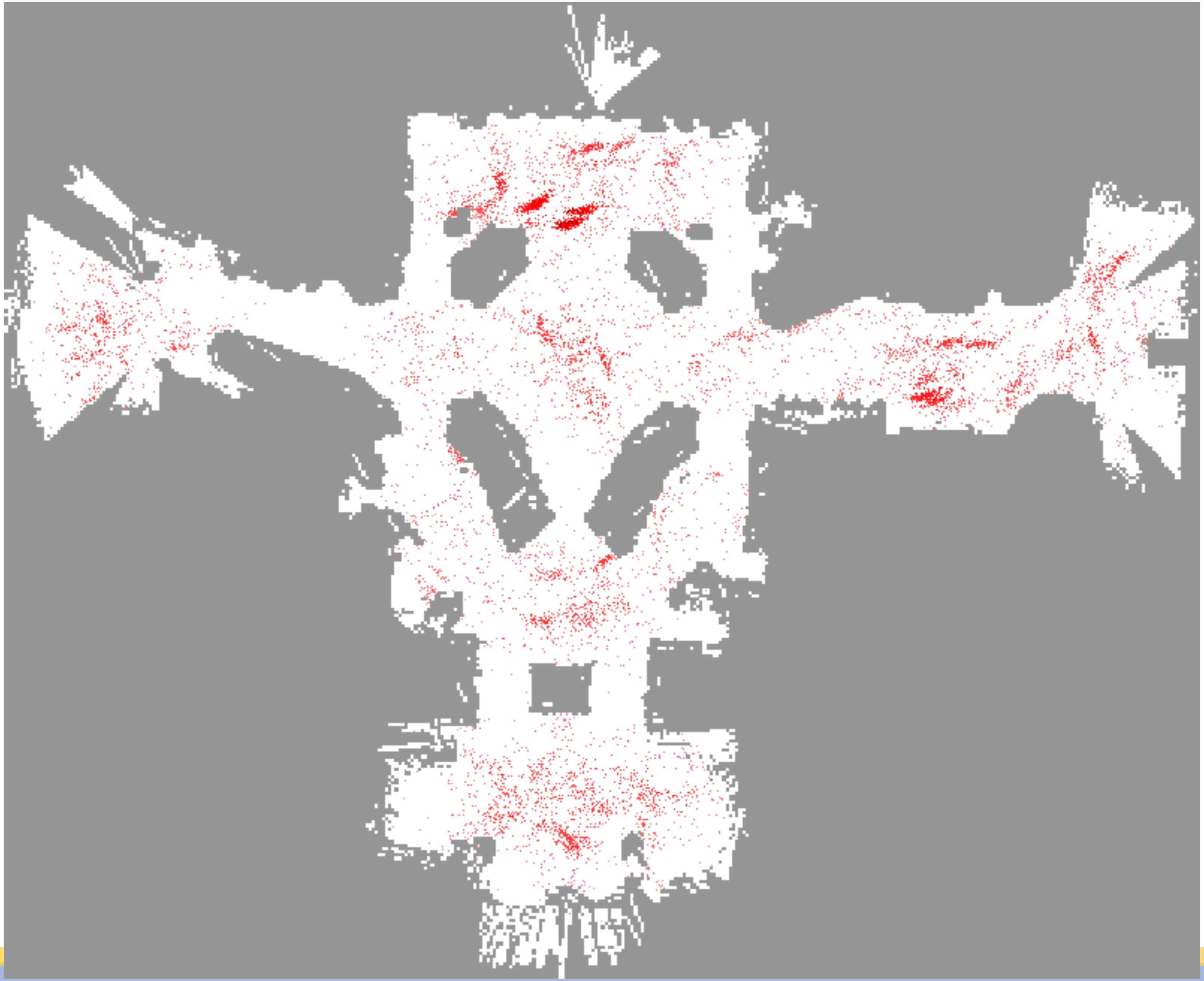


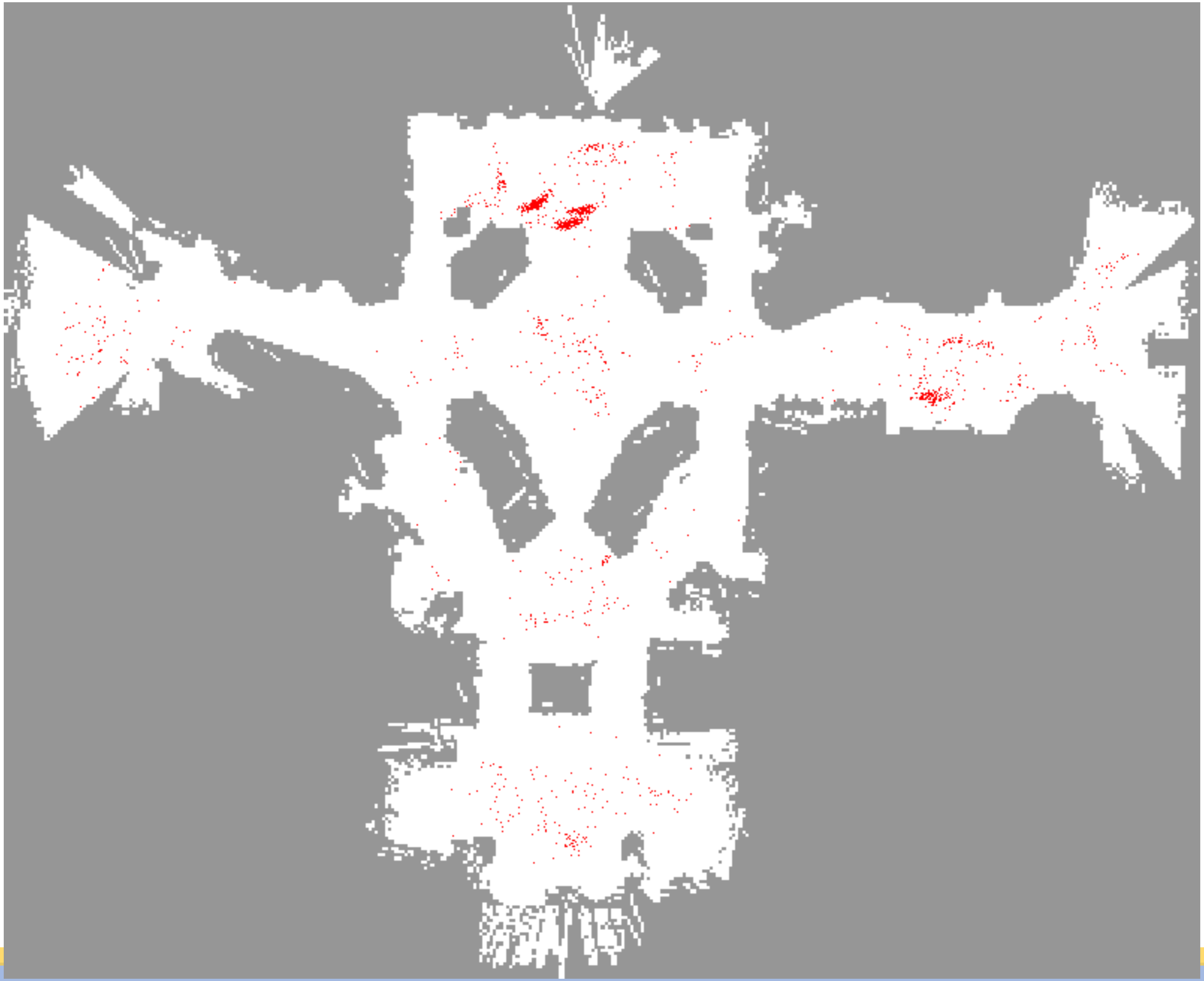


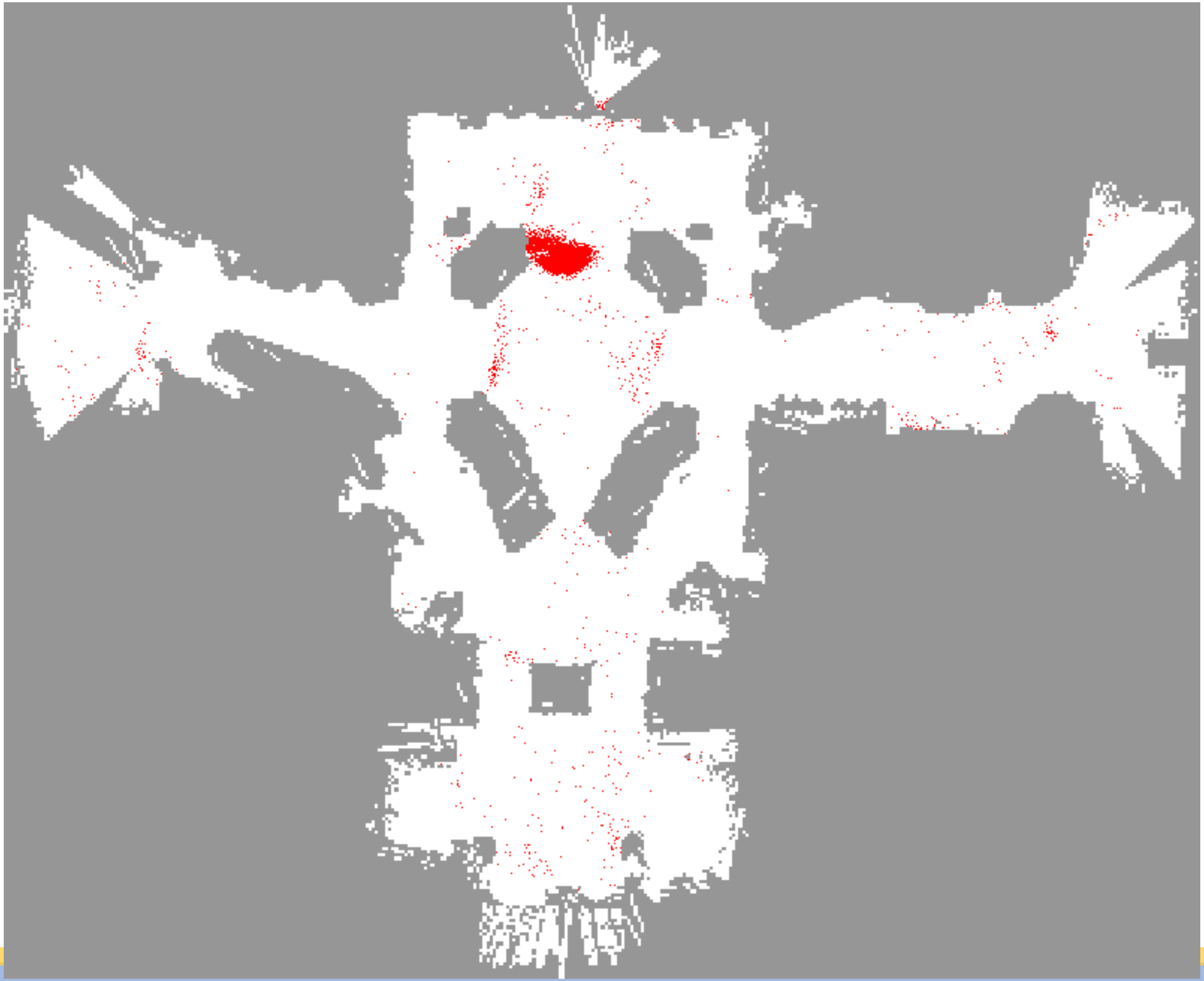


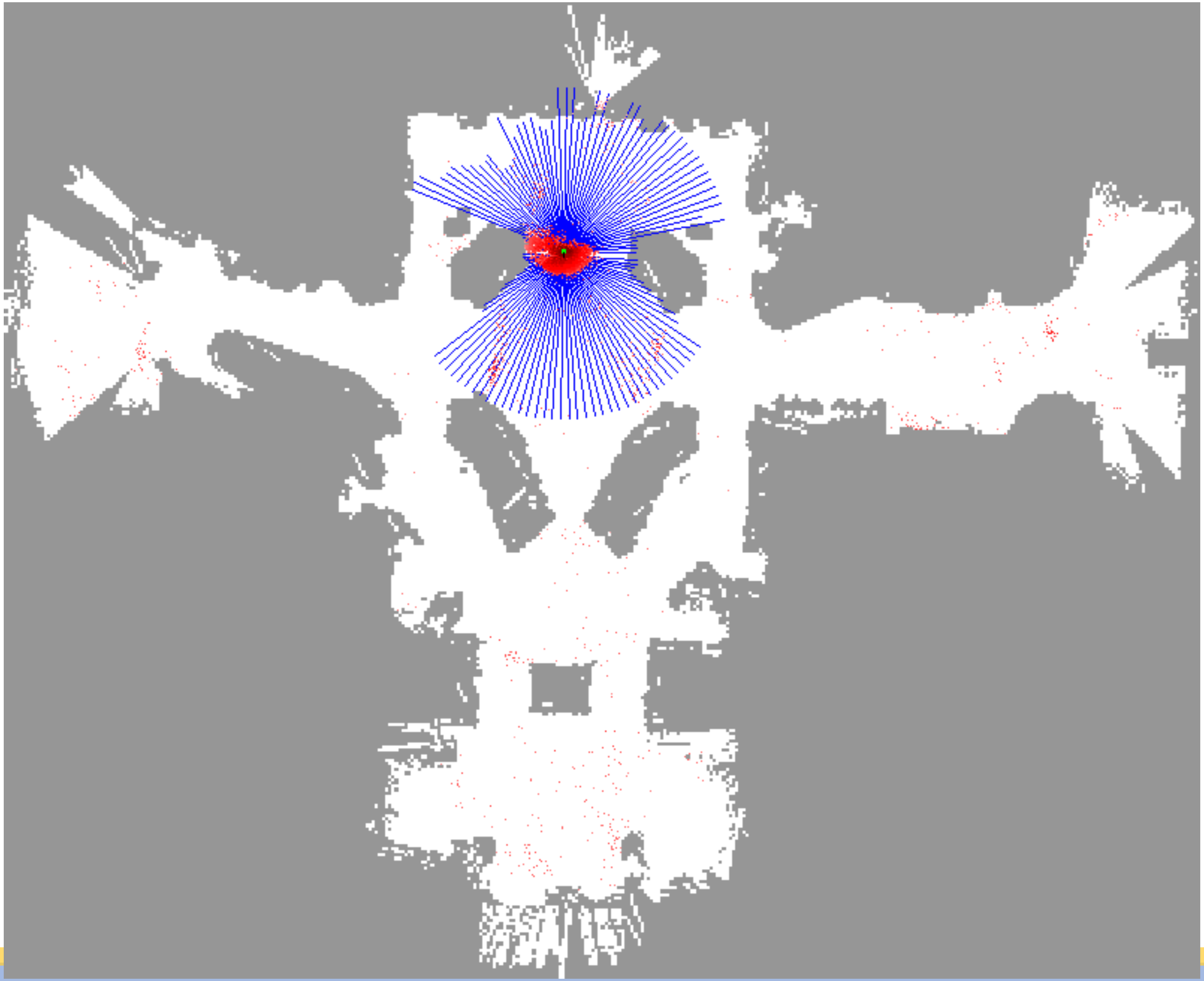


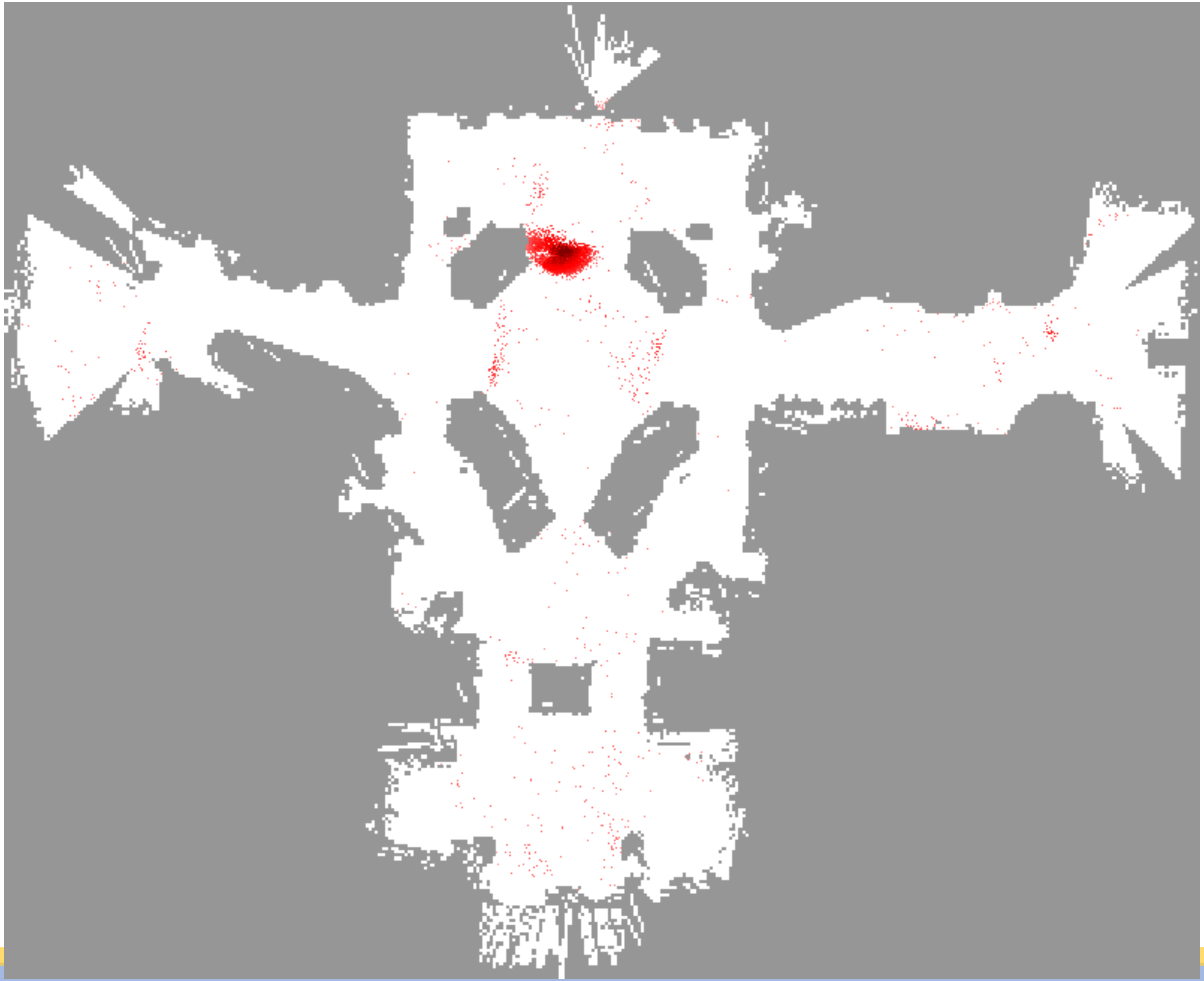


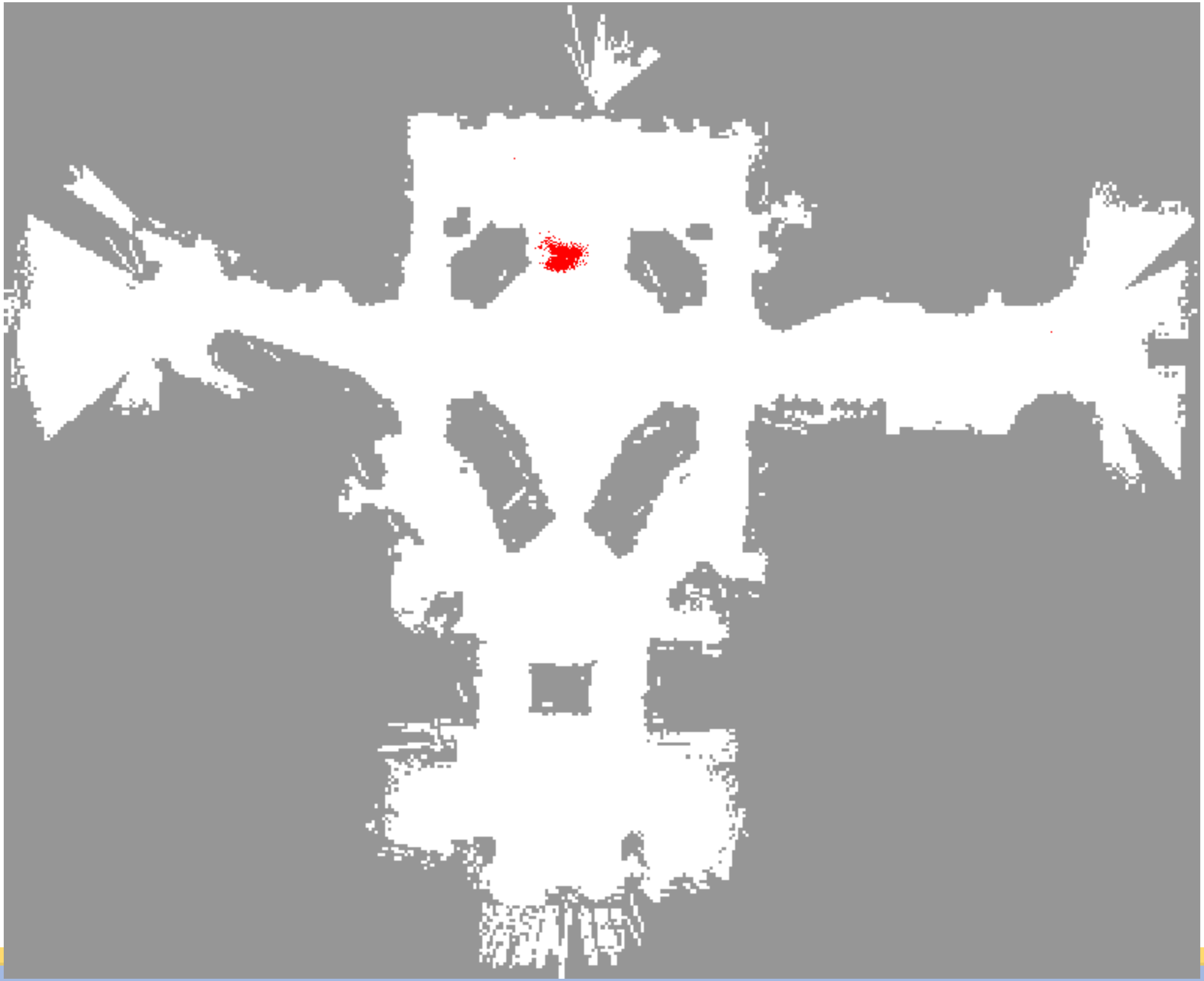


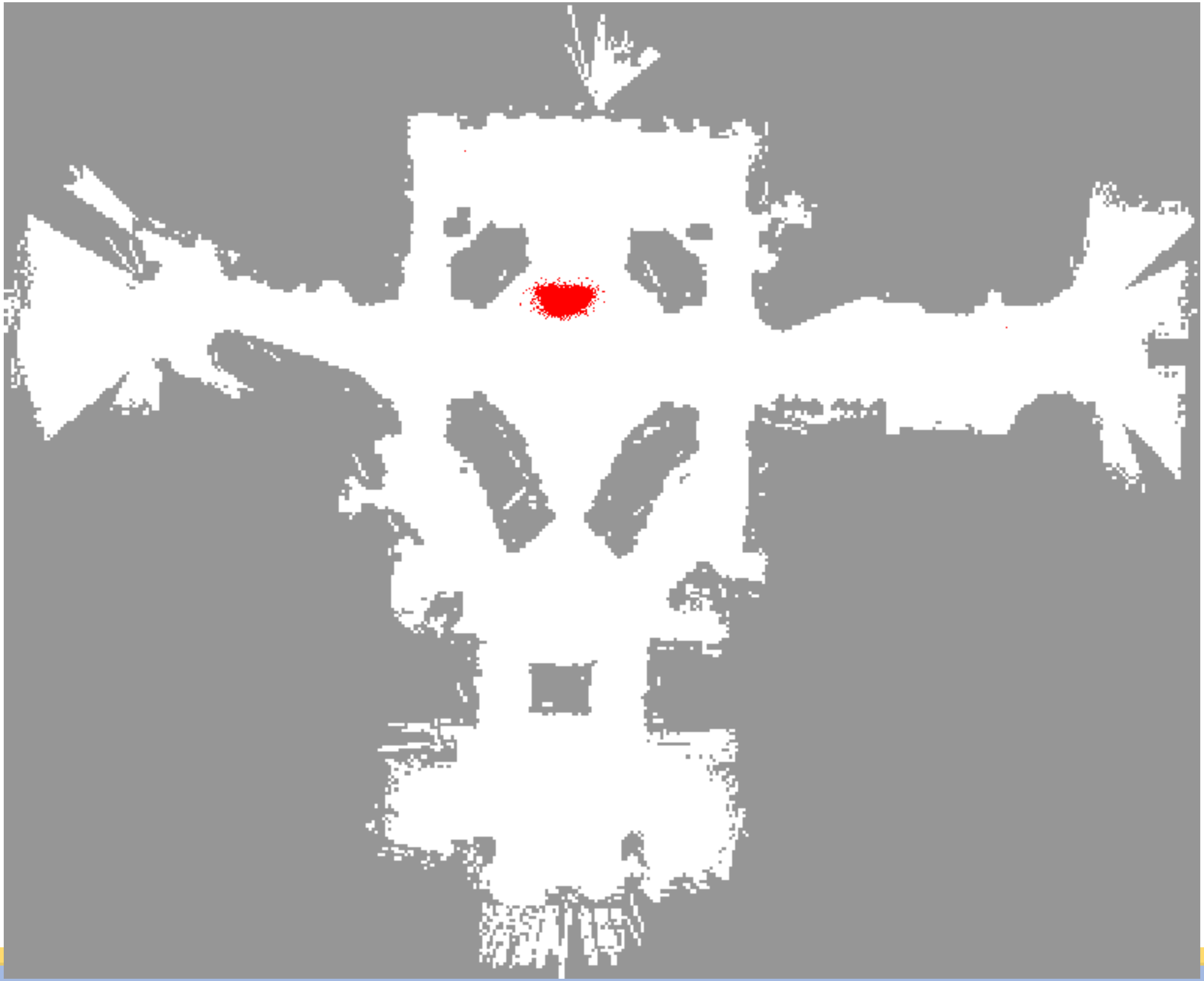


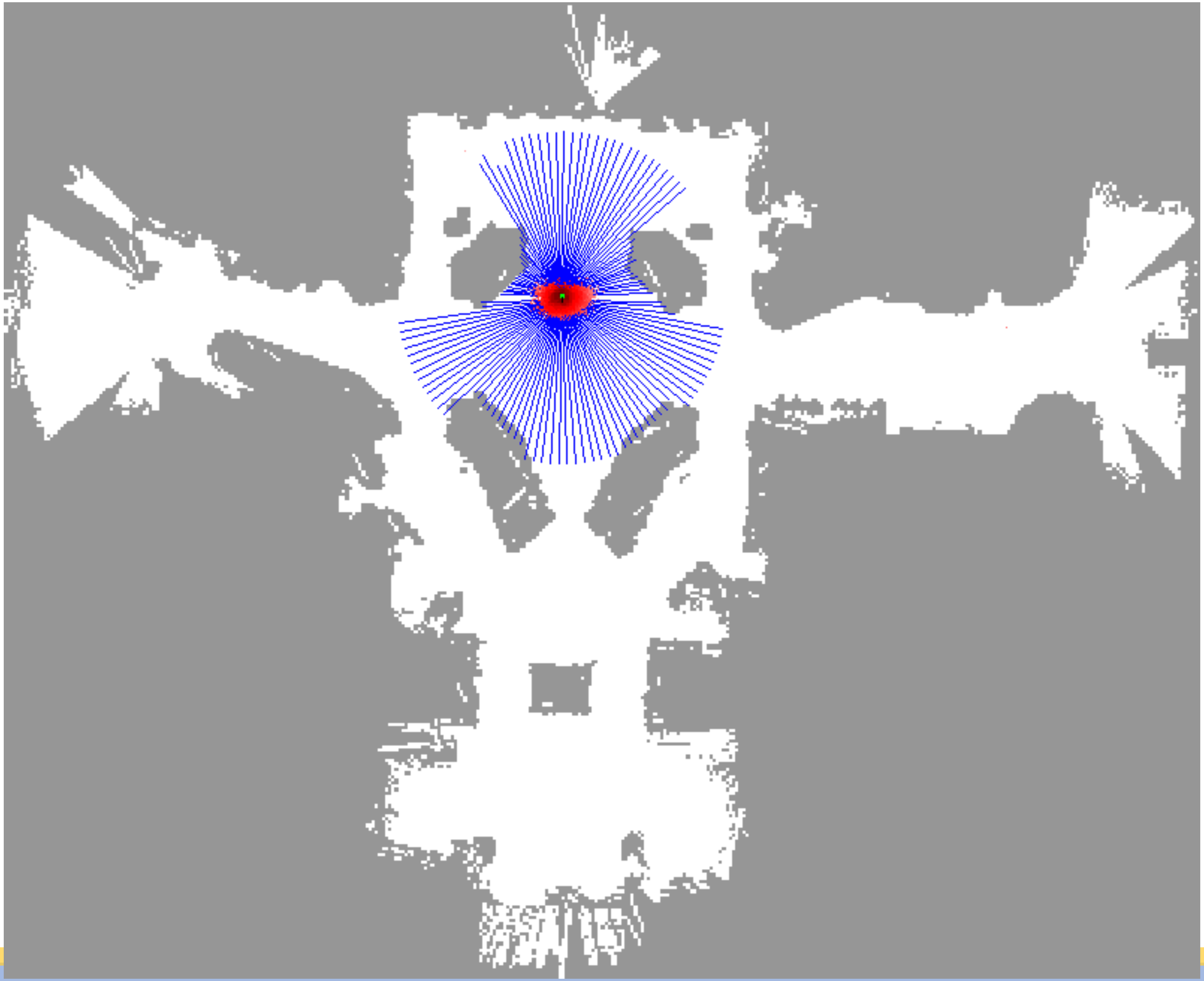


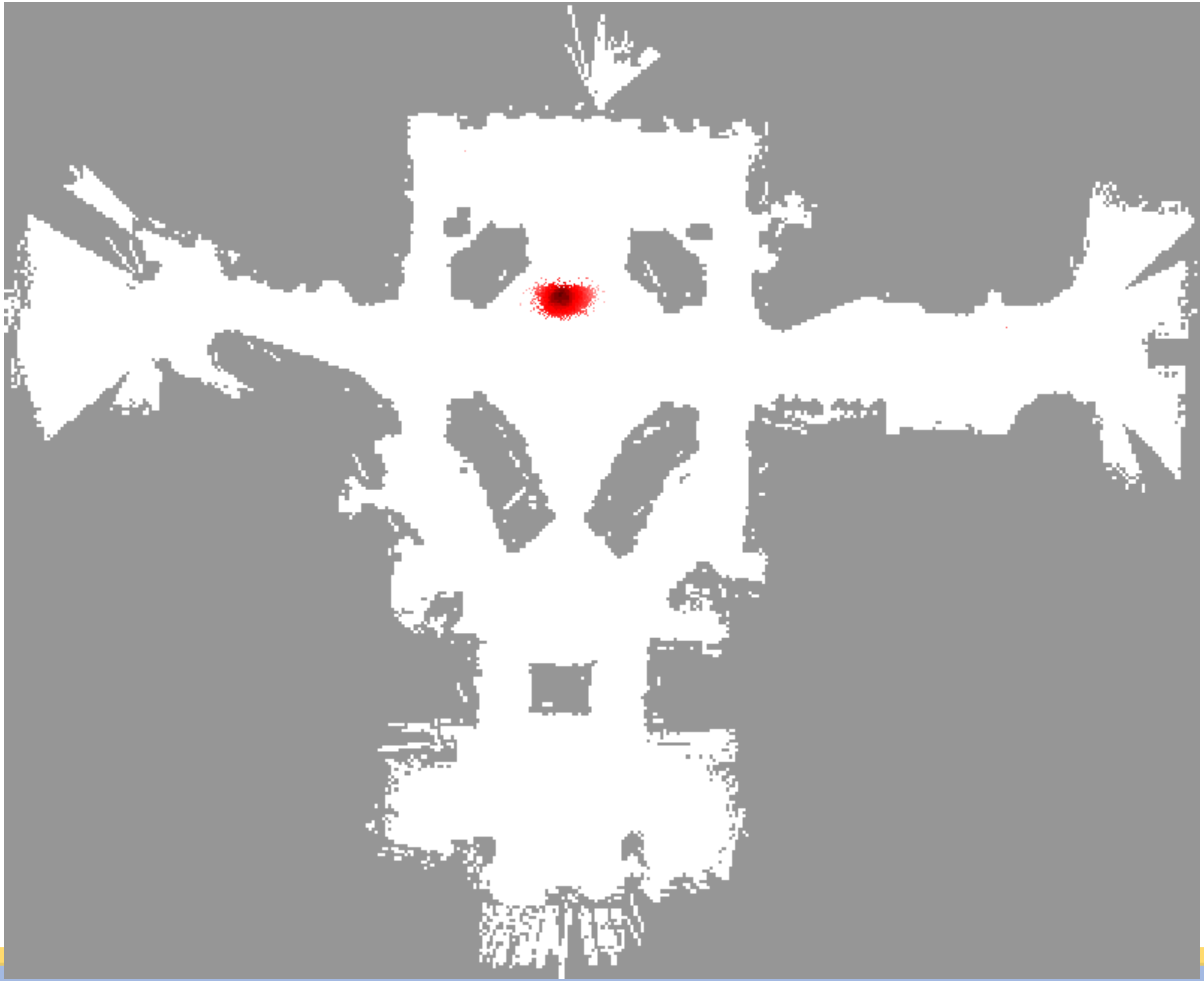


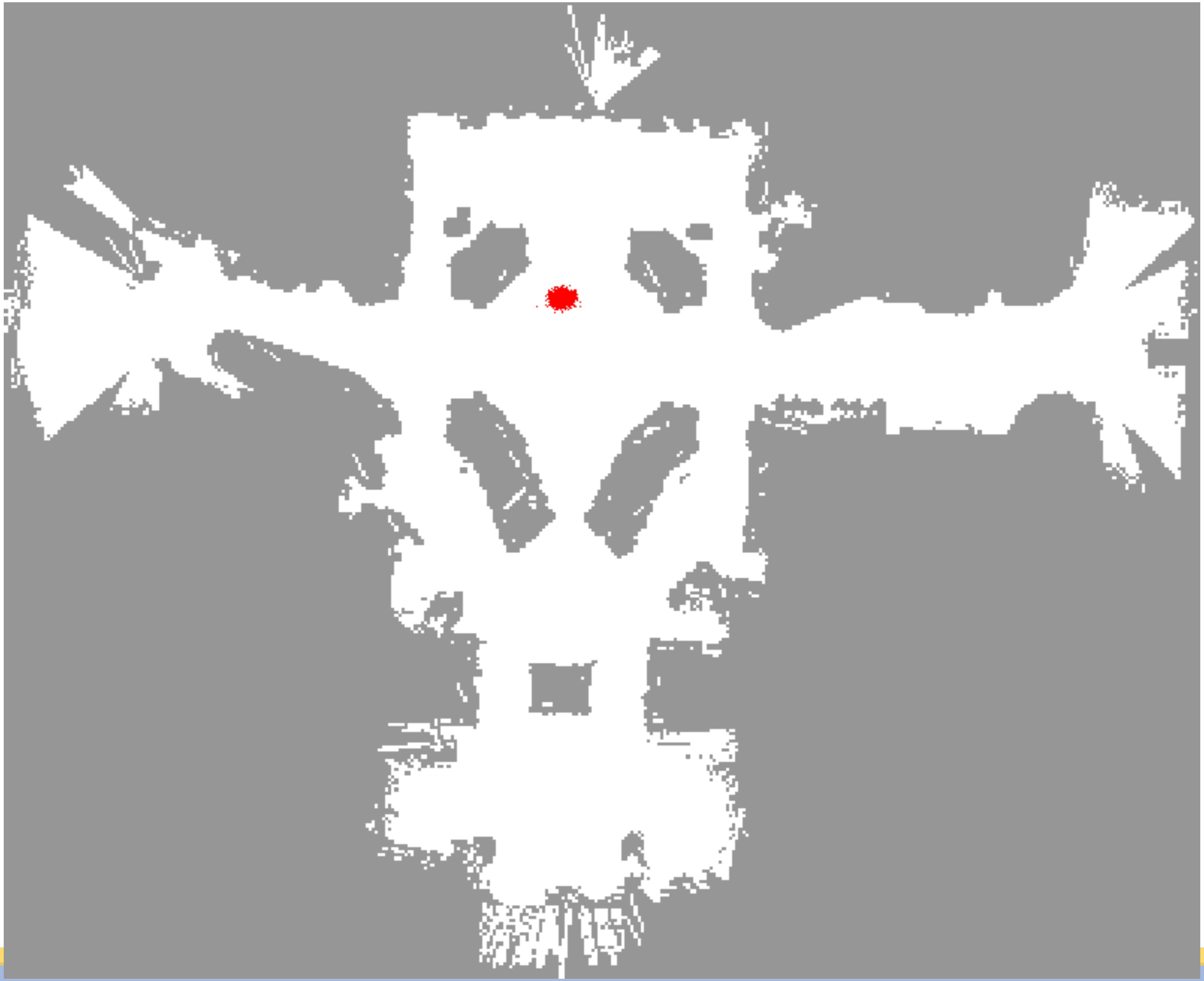


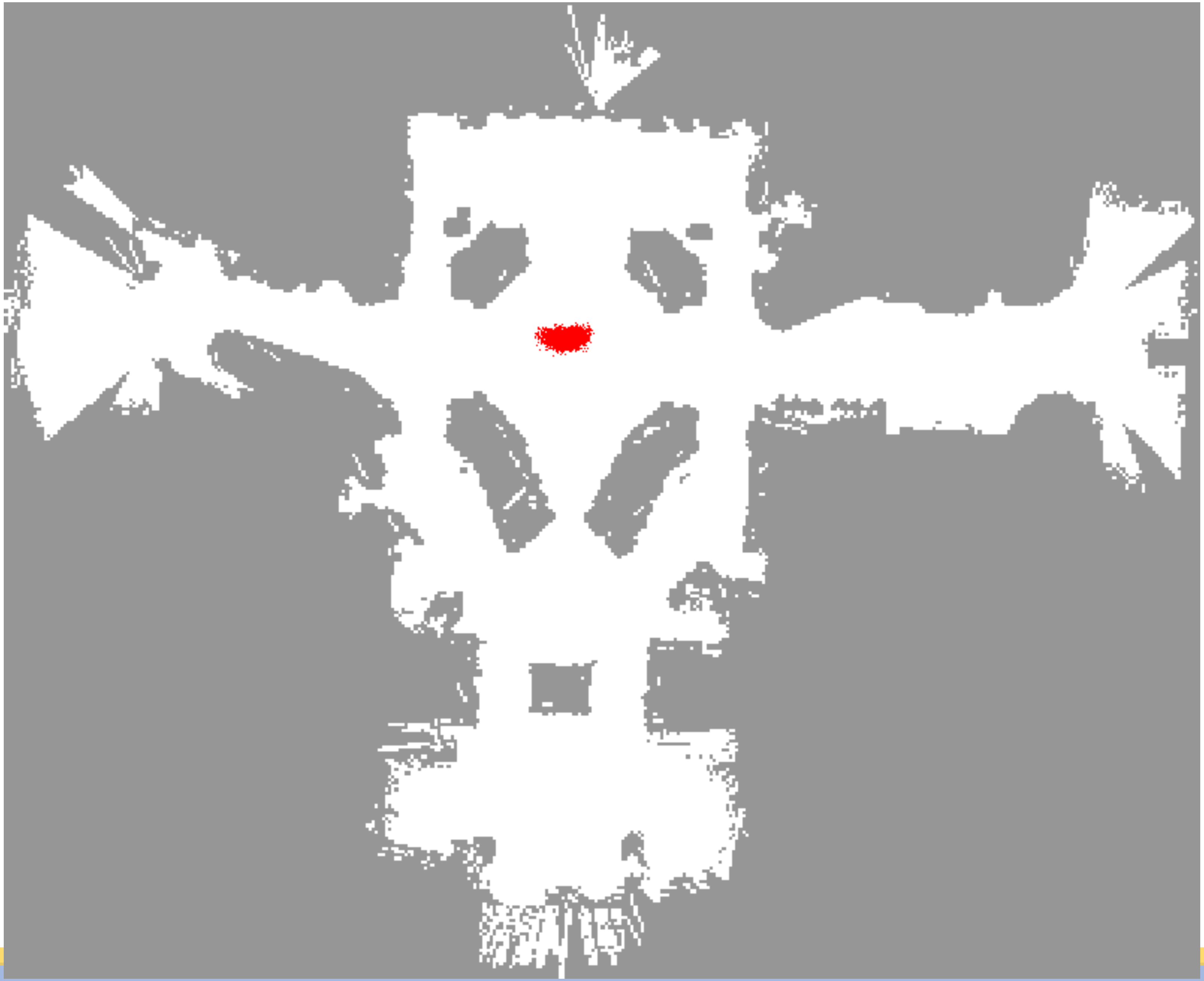


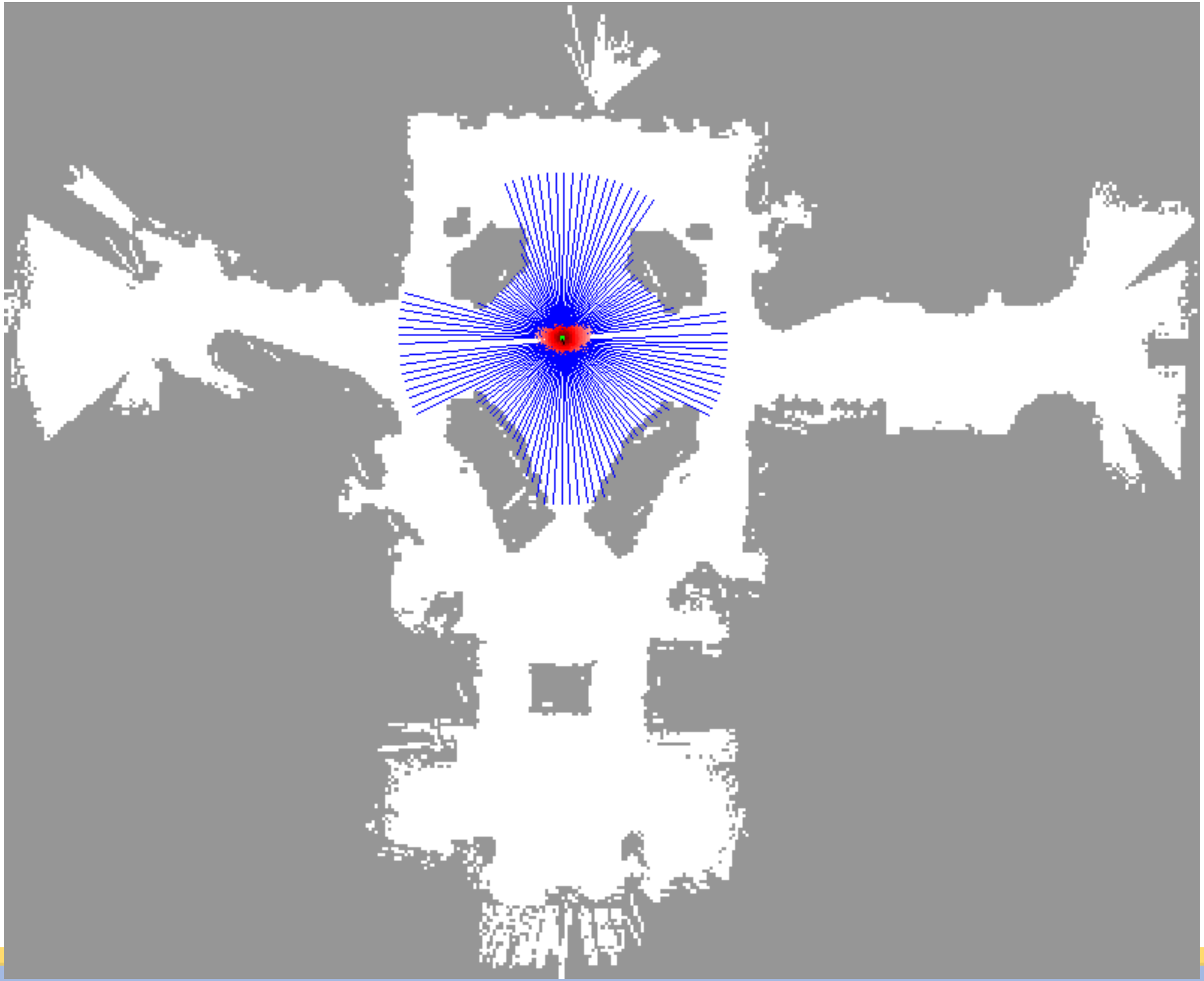


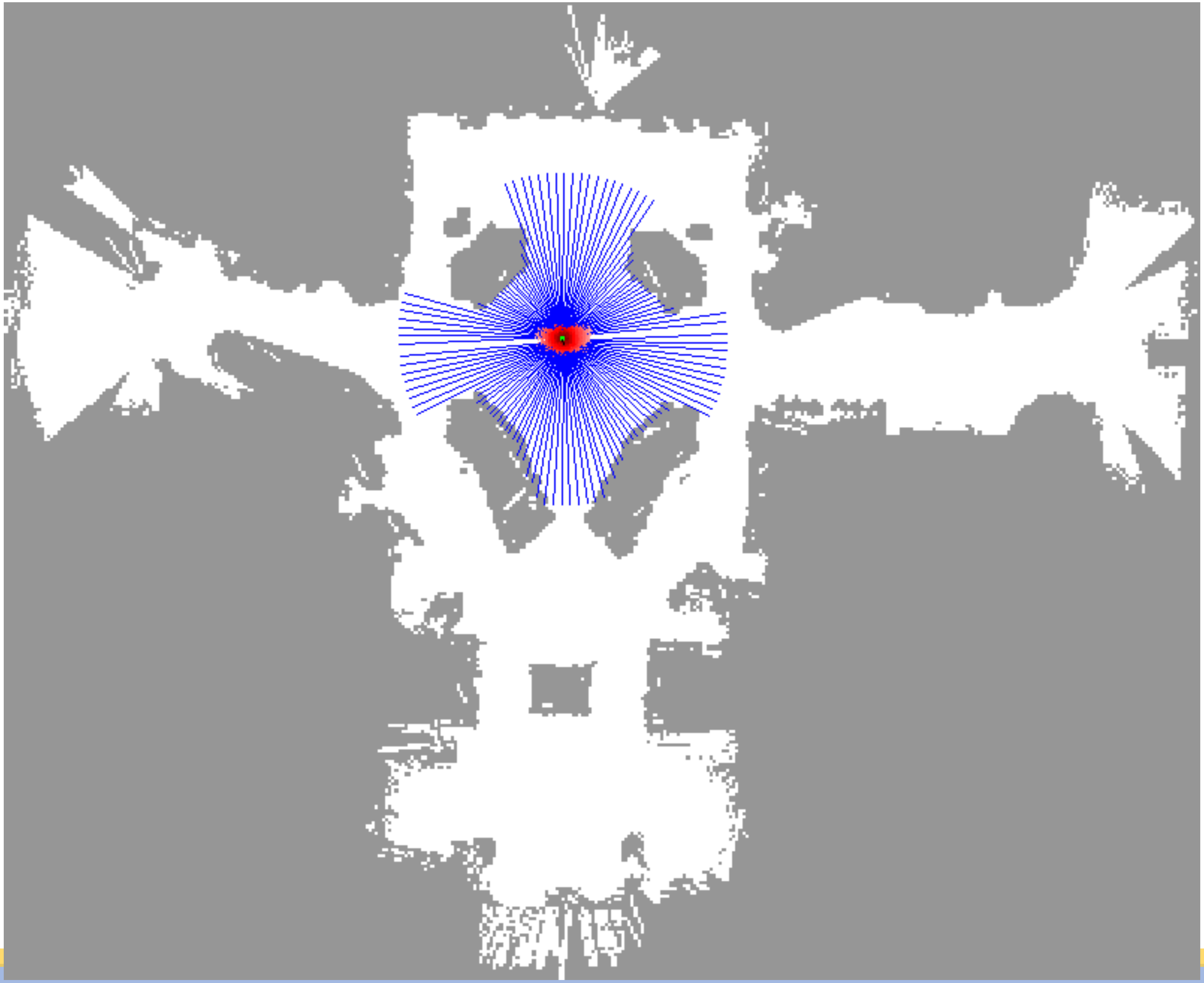




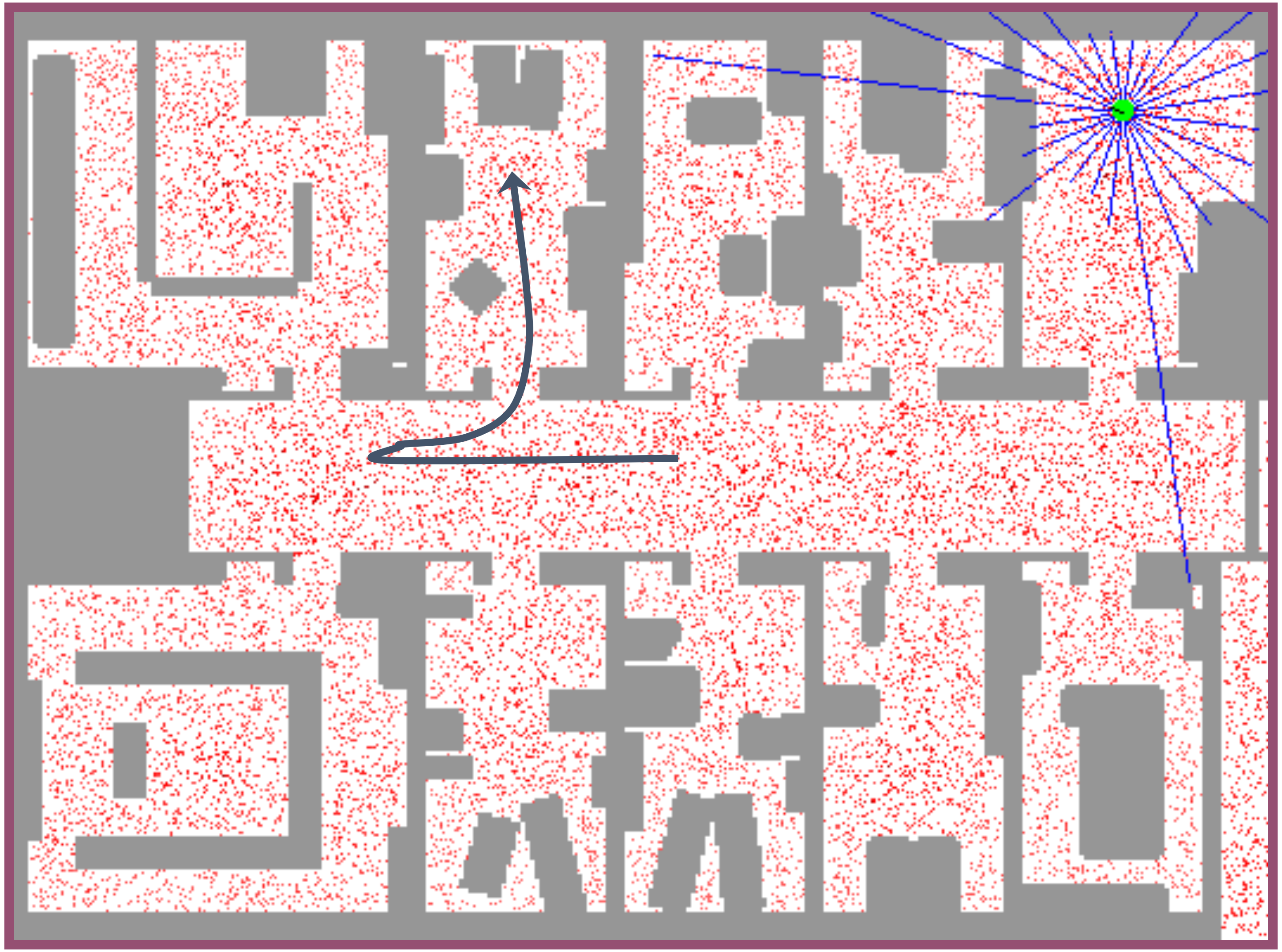




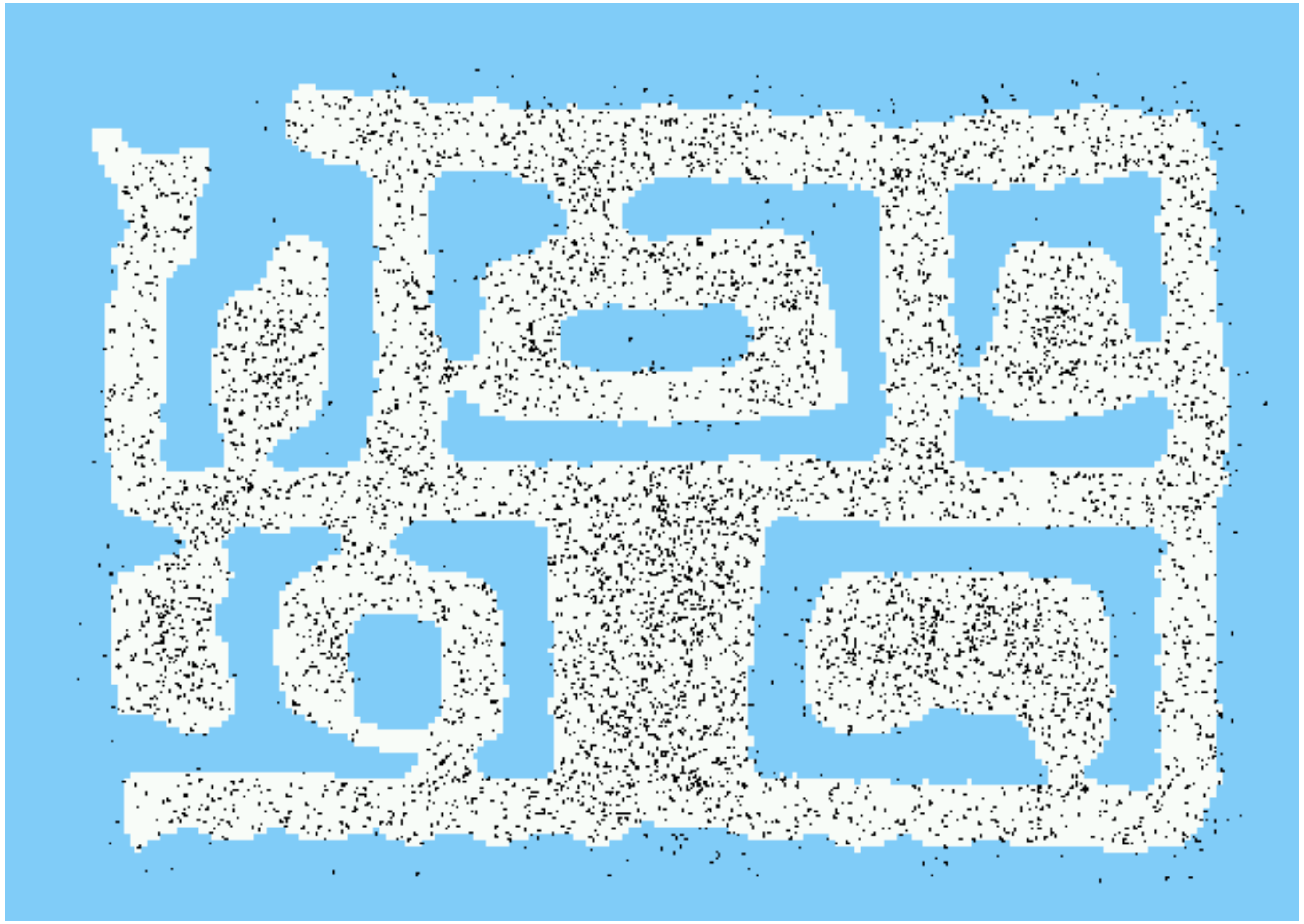




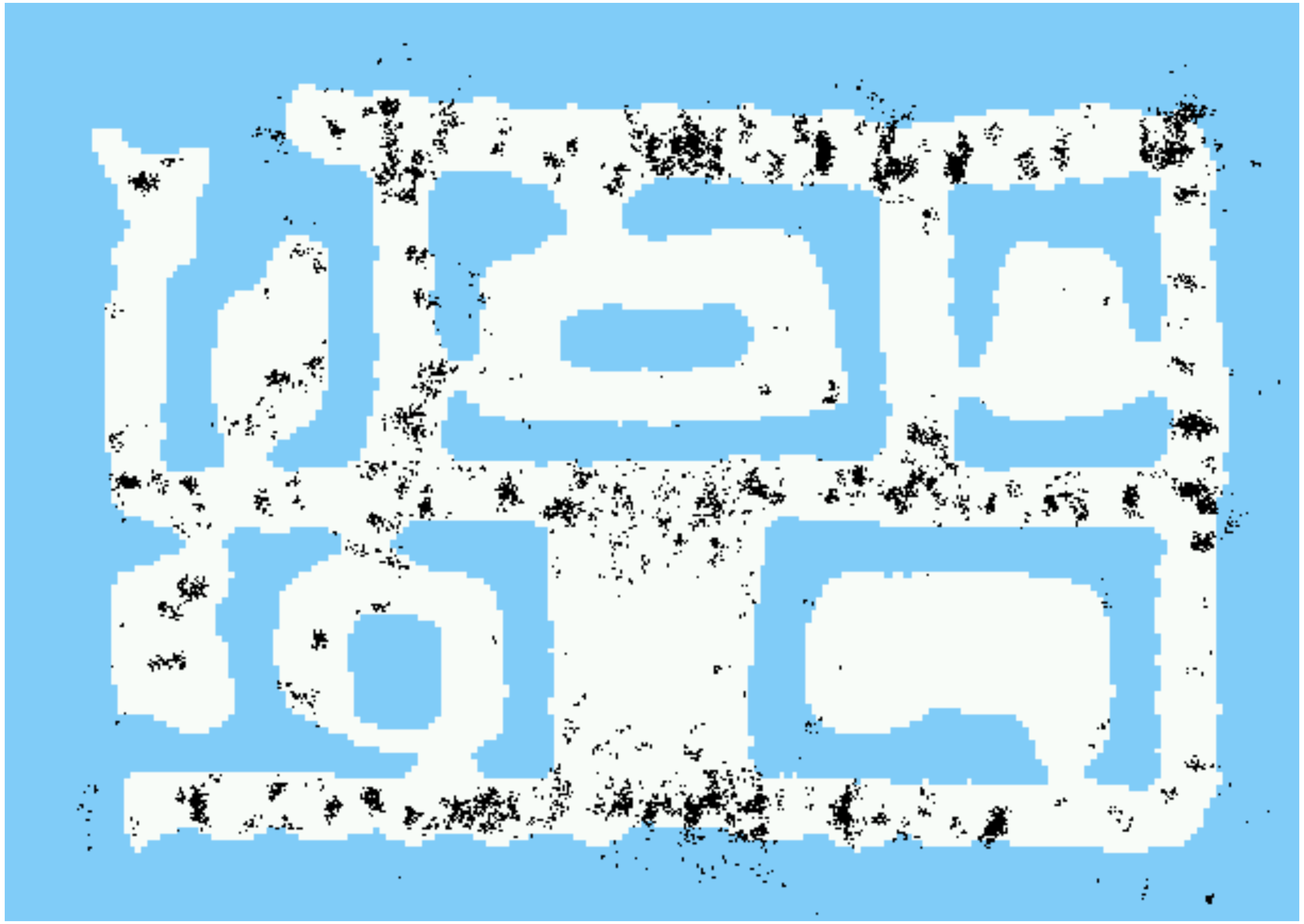
Sample-based Localization (sonar)



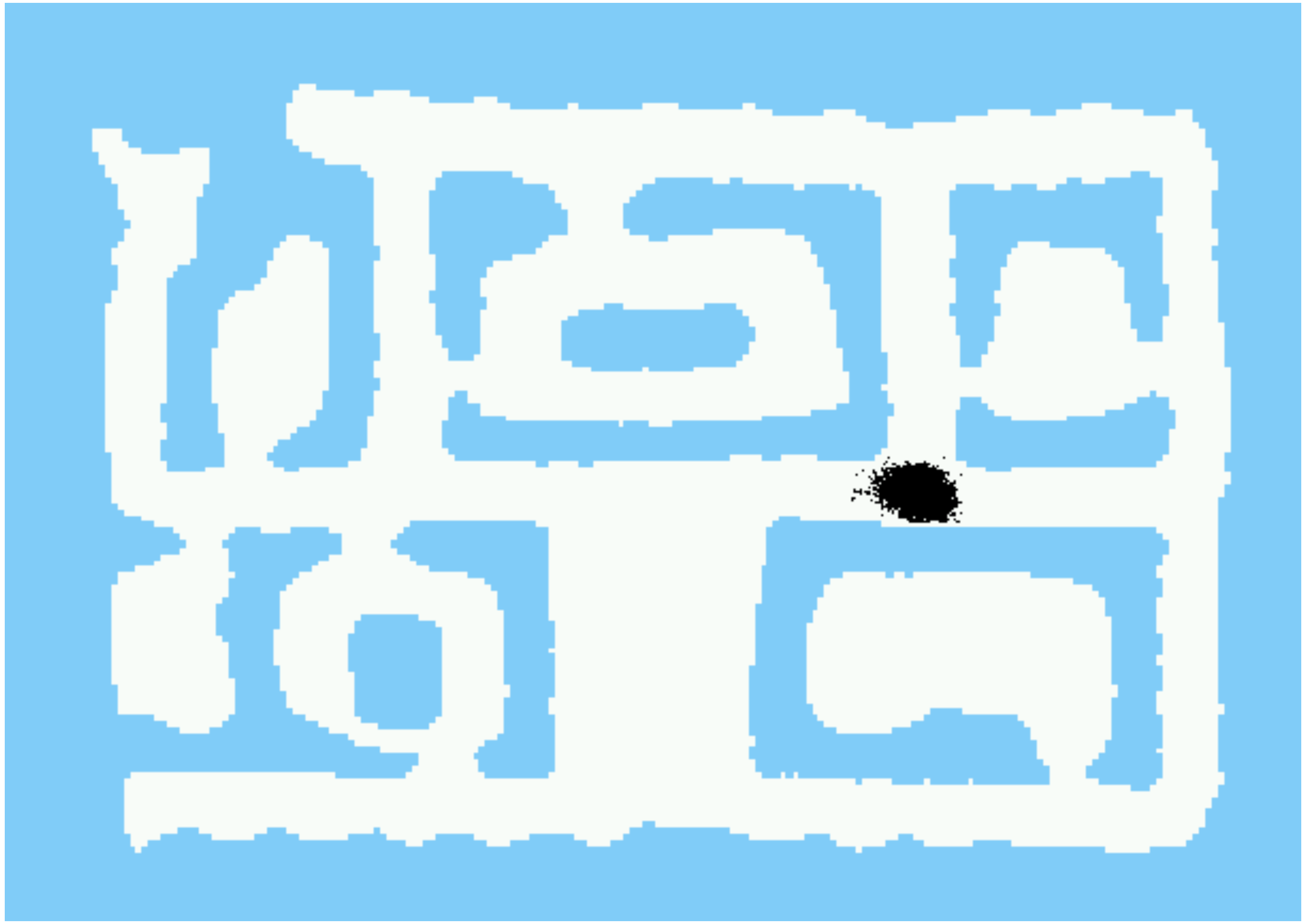
Initial Distribution



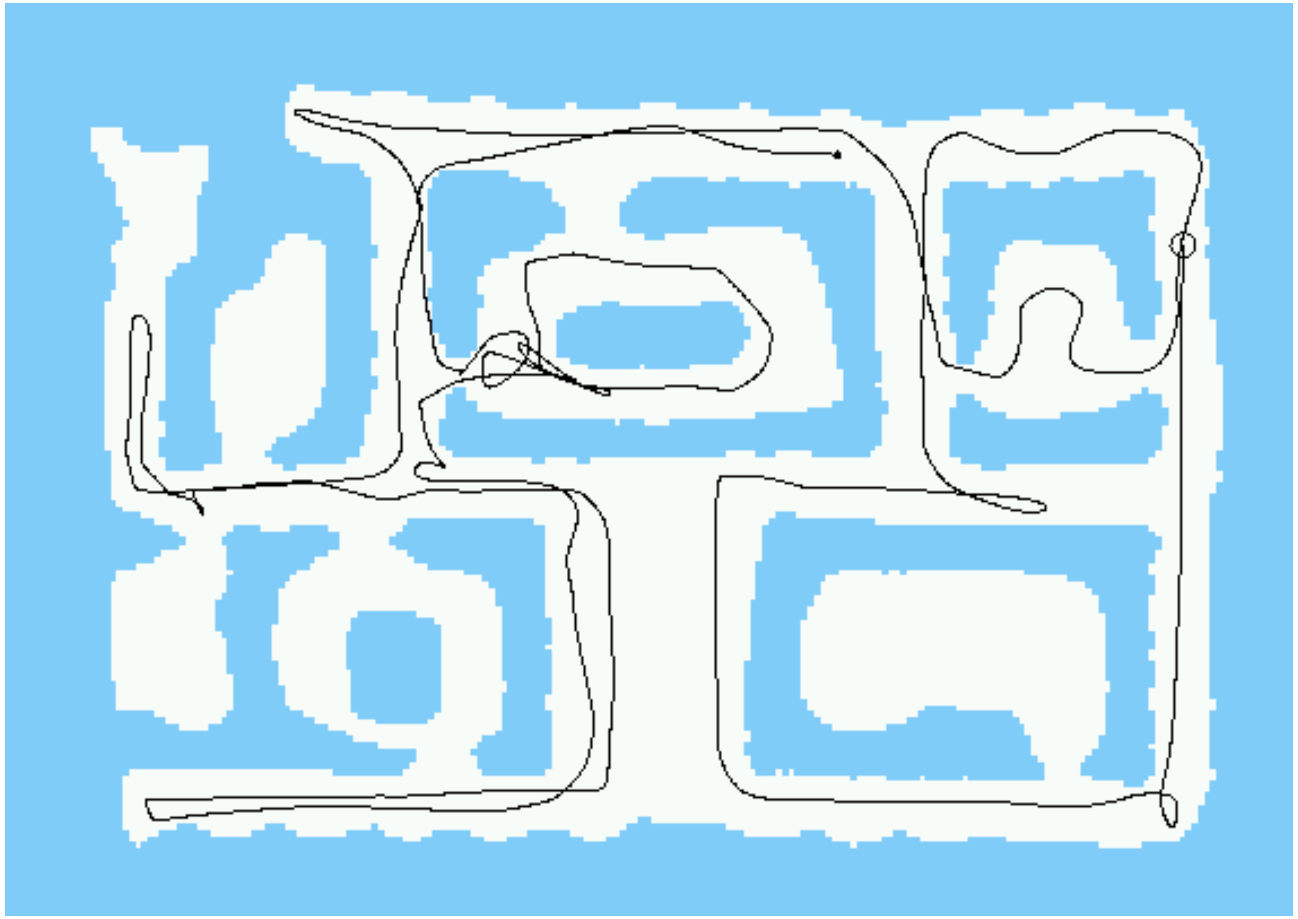
After Incorporating Ten Ultrasound Scans



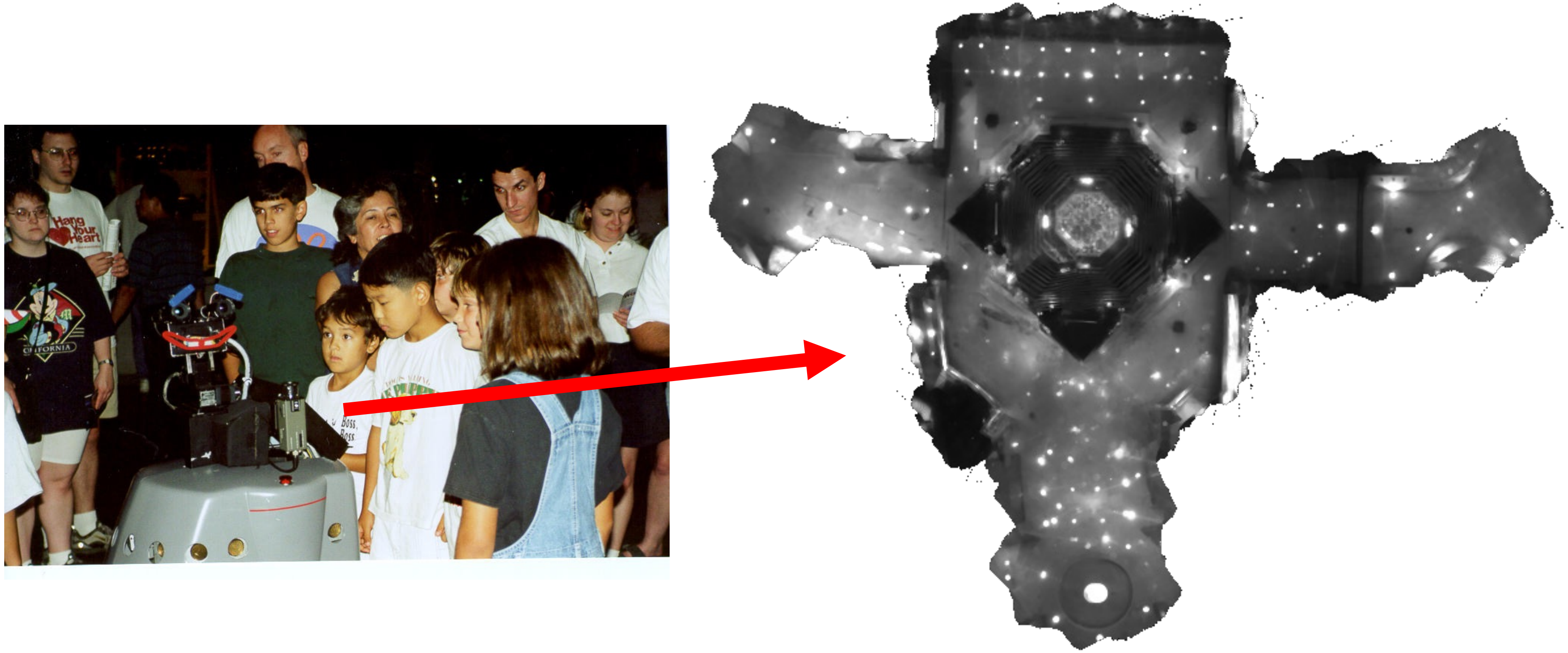
After Incorporating 65 Ultrasound Scans



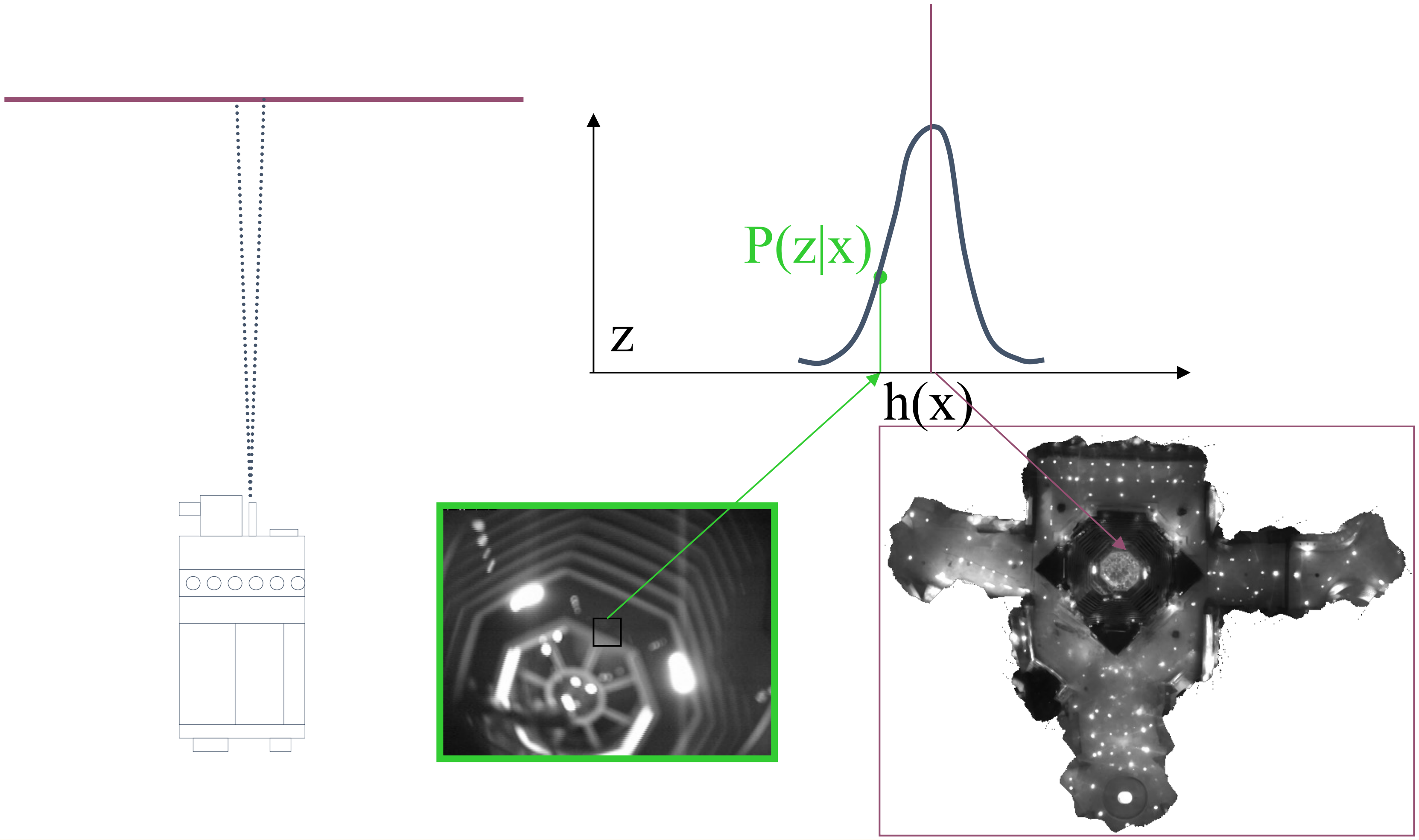
Estimated Path



Using Ceiling Maps for Localization

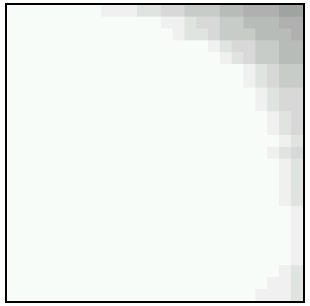


Vision-based Localization

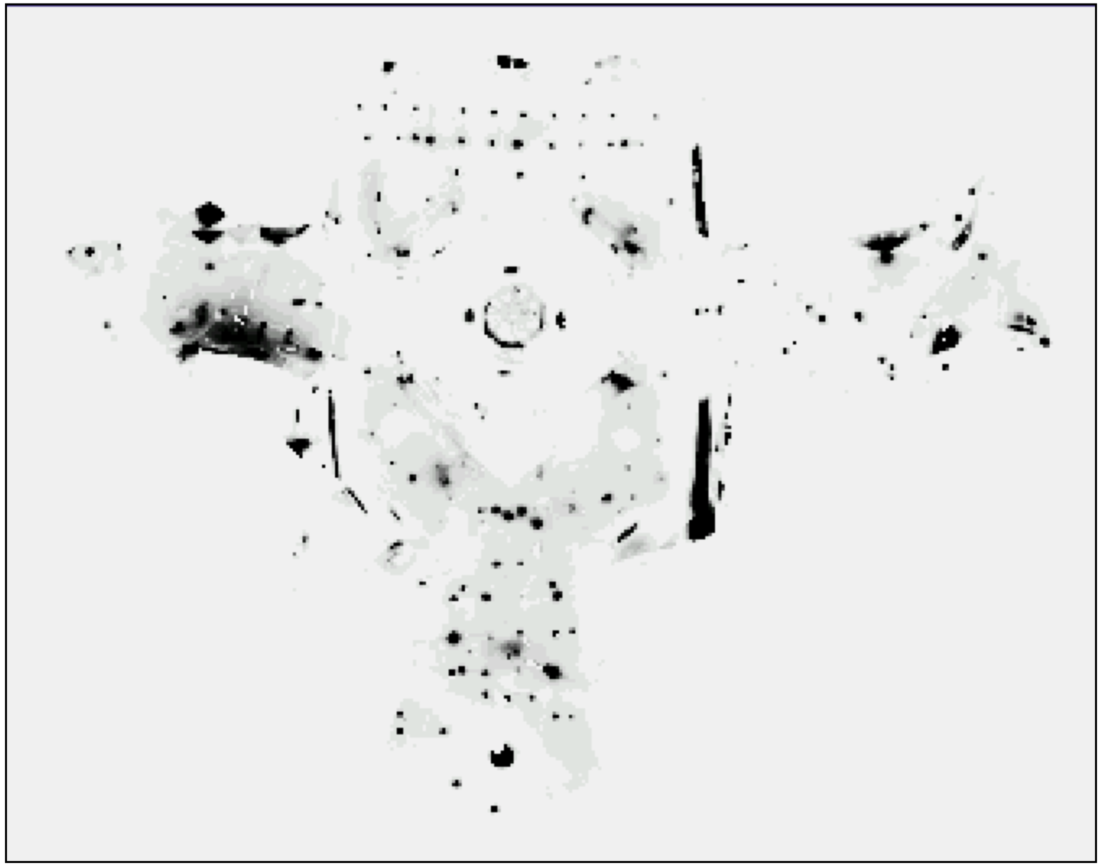


Under a Light:

Measurement z:

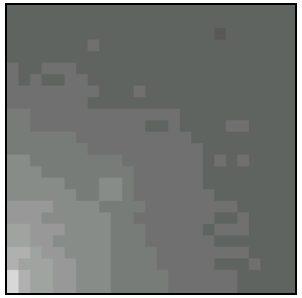


***P(z|x)*:**

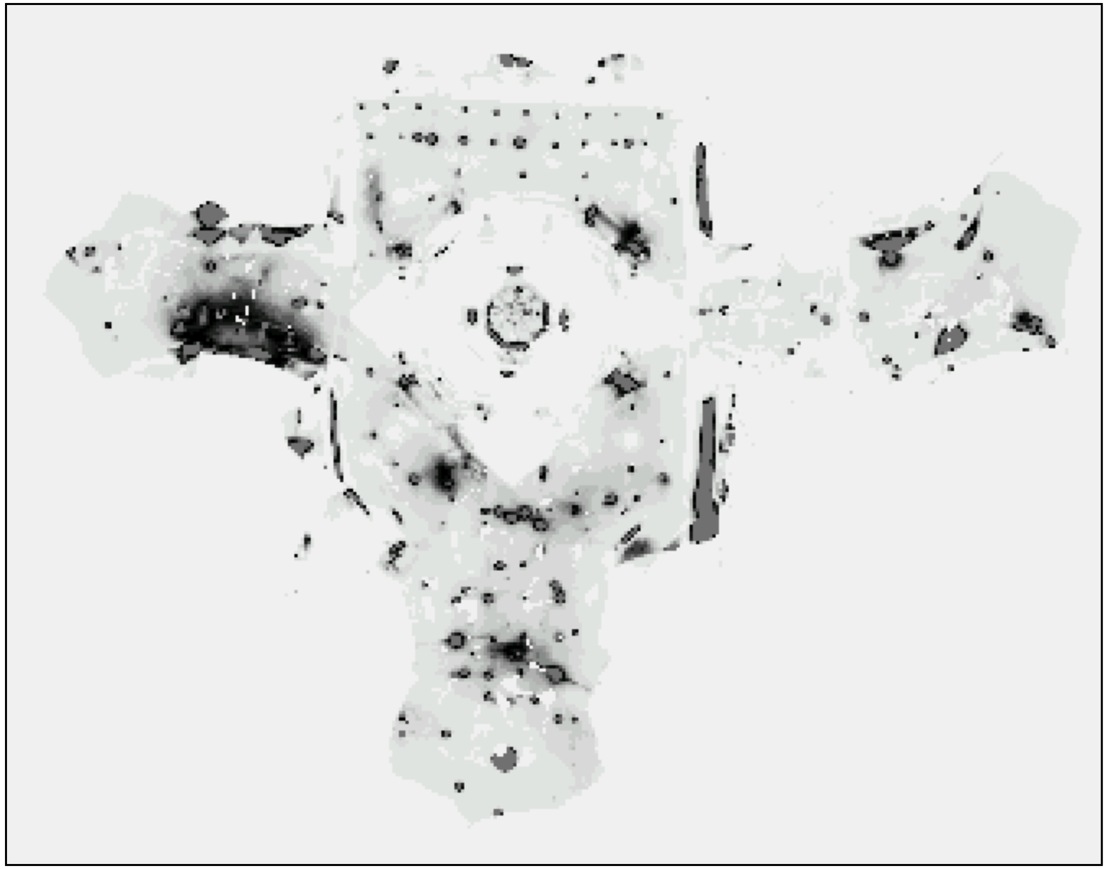


Next to a Light

Measurement z:



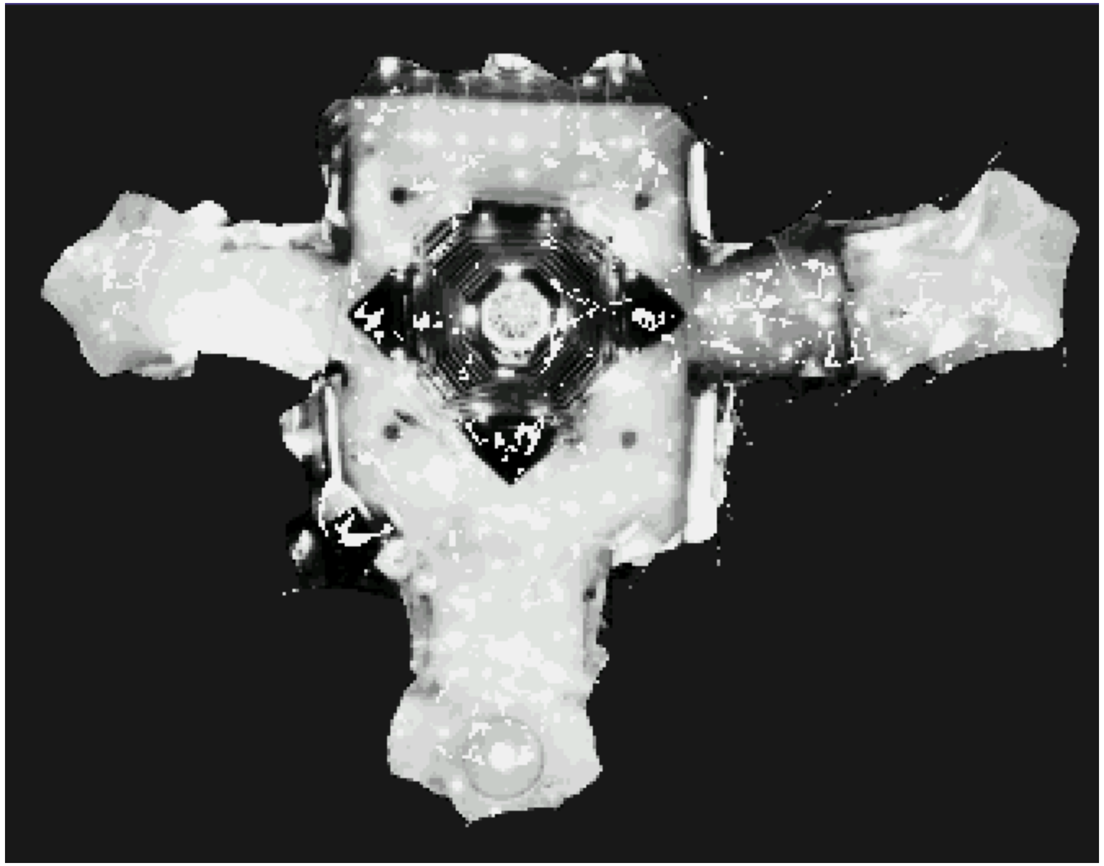
***P(z|x)*:**



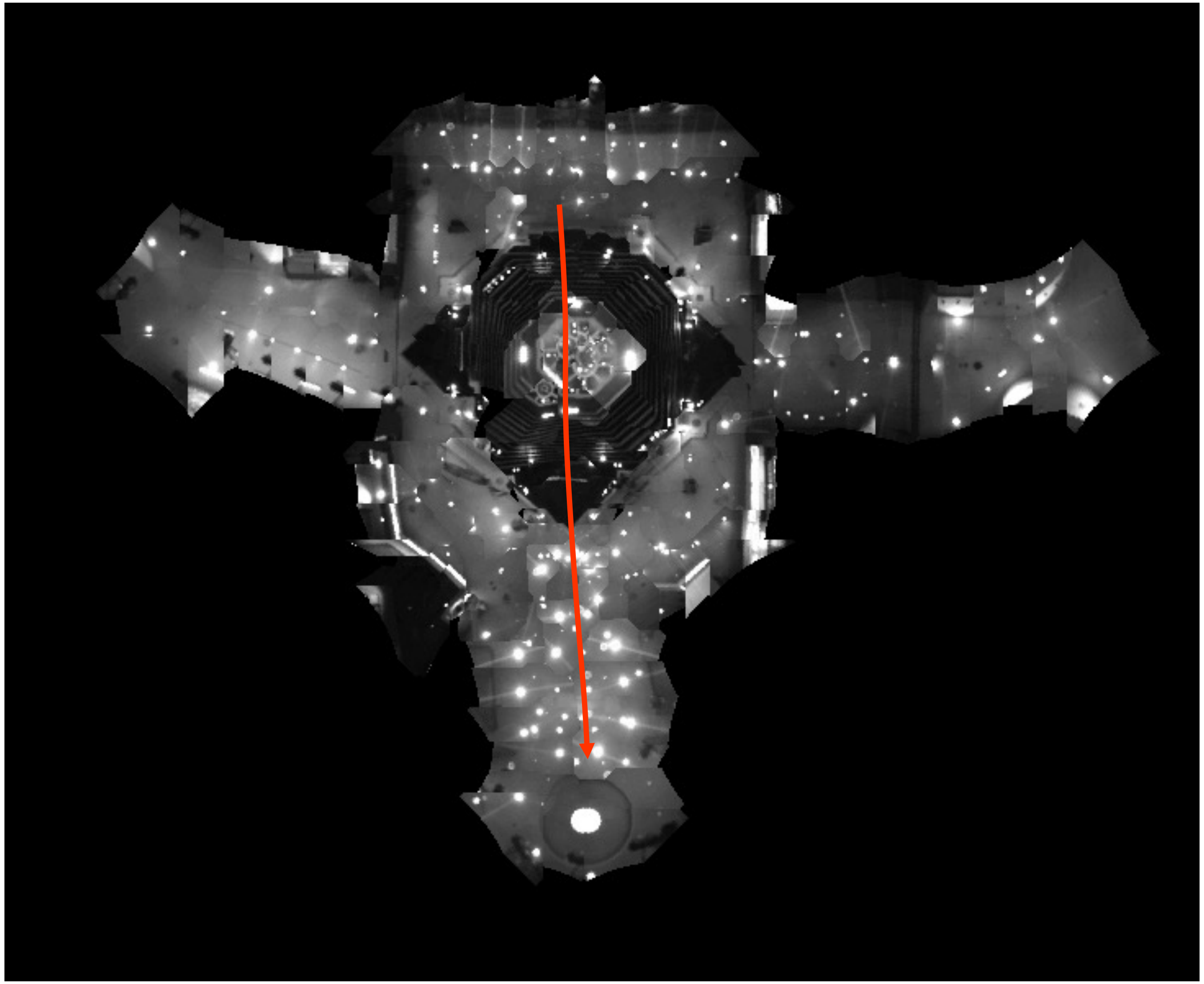
Elsewhere

Measurement z:

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



Global Localization Using Vision



Limitations

- The approach described so far is able to
  - track the pose of a mobile robot and to
  - globally localize the robot.

- Can we deal with localization errors (i.e., the kidnapped robot problem)?
- How to handle localization errors/failures?
  - Particularly serious when the number of particles is small

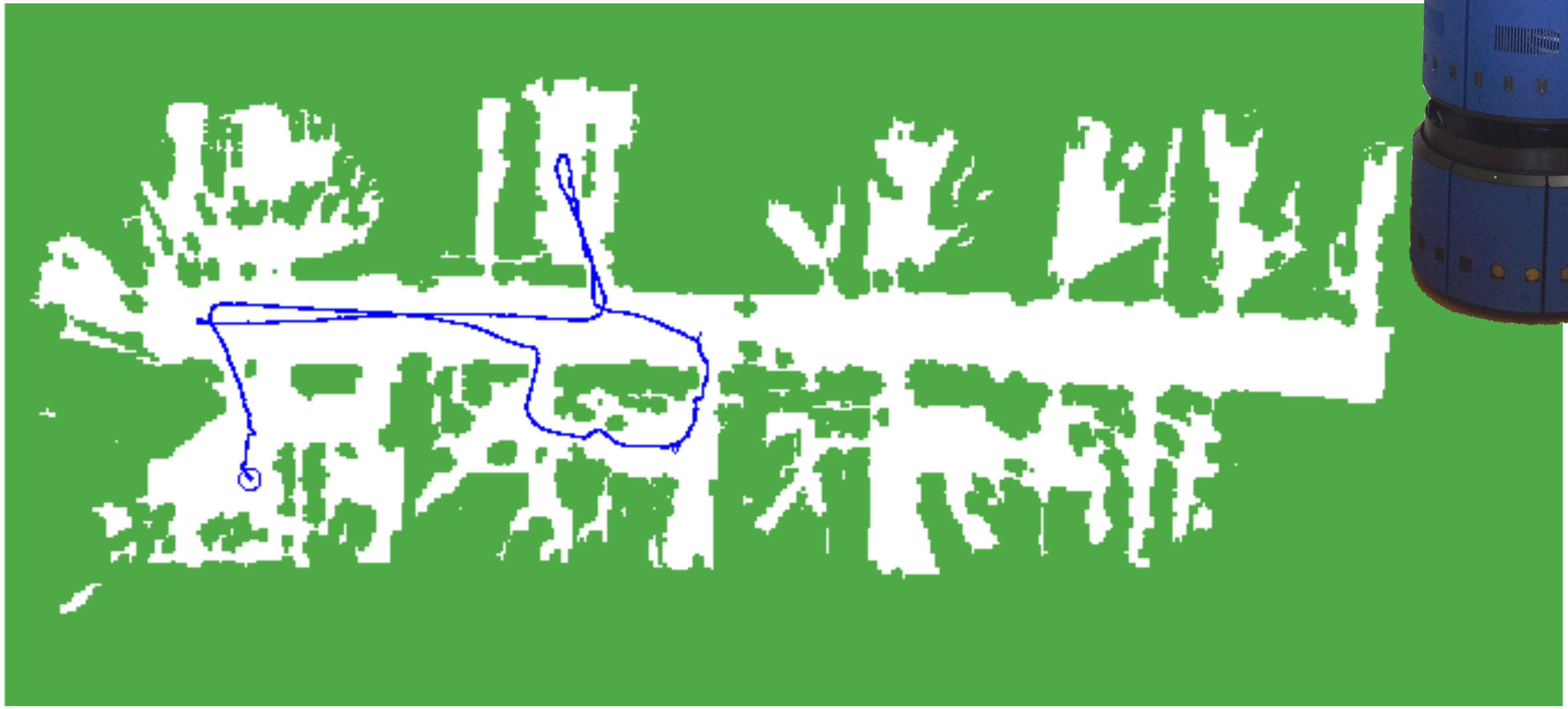
Approaches

- Randomly insert samples
  - Why?
  - The robot can be teleported at any point in time
- How many particles to add? With what distribution?
  - Add particles according to localization performance
  - Monitor the probability of sensor measurements $p(z_t|z_{1:t-1}, u_{1:t}, m)$
  - For particle filters: $p(z_t|z_{1:t-1}, u_{1:t}, m) \approx \frac{1}{M}\sum w_t^{[m]}$
- 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).

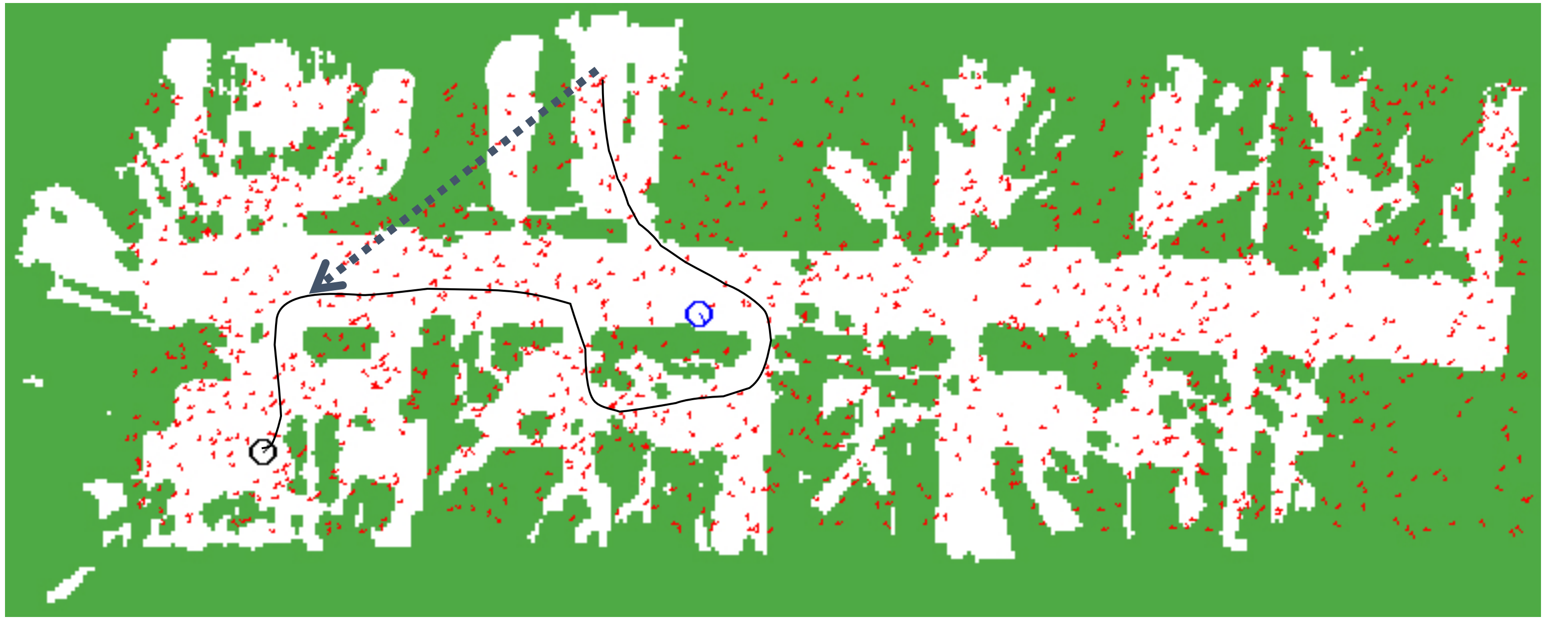
Random Samples Vision-Based Localization

936 Images, 4MB, .6secs/image

Trajectory of the robot:



Kidnapping the Robot



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.