ECE 484: Principles of Safe Autonomy (Fall 2025)
Lecture 13: Filtering and Localization

Professor: Huan Zhang

https://publish.illinois.edu/safe-autonomy/

https://huan-zhang.com

huanz@illinois.edu

Slides adapted from Prof. Sayan Mitra's slides for Spring 2025;

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

Some slides are from the book's website

Outline of state estimation module

Problem. Estimate the current state x_t of the system from knowledge about past observations $z_{0:t}$, control inputs $u_{0:t}$, and map m

Bayes filter and its variations:

- Grid localization (previous lecture)
- Particle filter (this lecture)
- Kalman filter (this lecture)

Histogram Filter or Discrete Bayes Filter

Notation: $bel(X_t = x_k) \coloneqq p_{k,t}$

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:

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

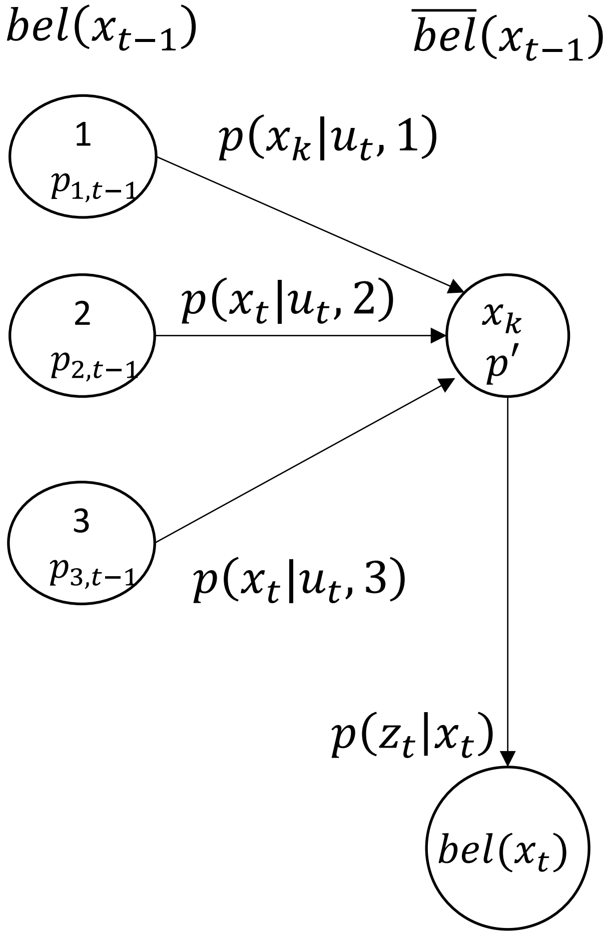
Prediction step with motion model

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

correction step with measurement model

end for

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



Bayes Filter: Continuous Distributions

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

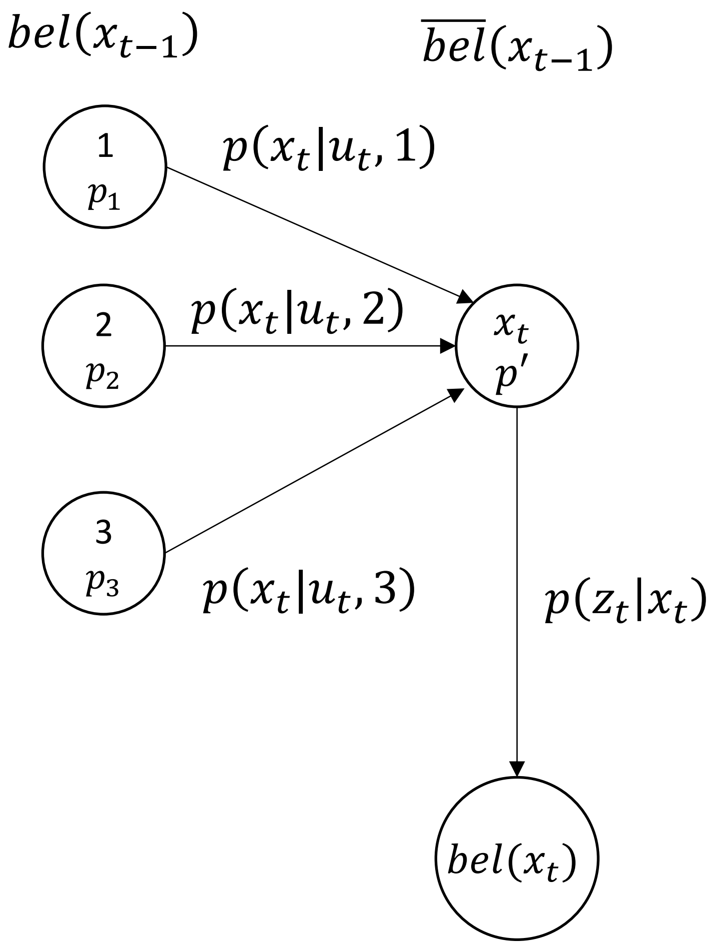
for all x_t do:

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

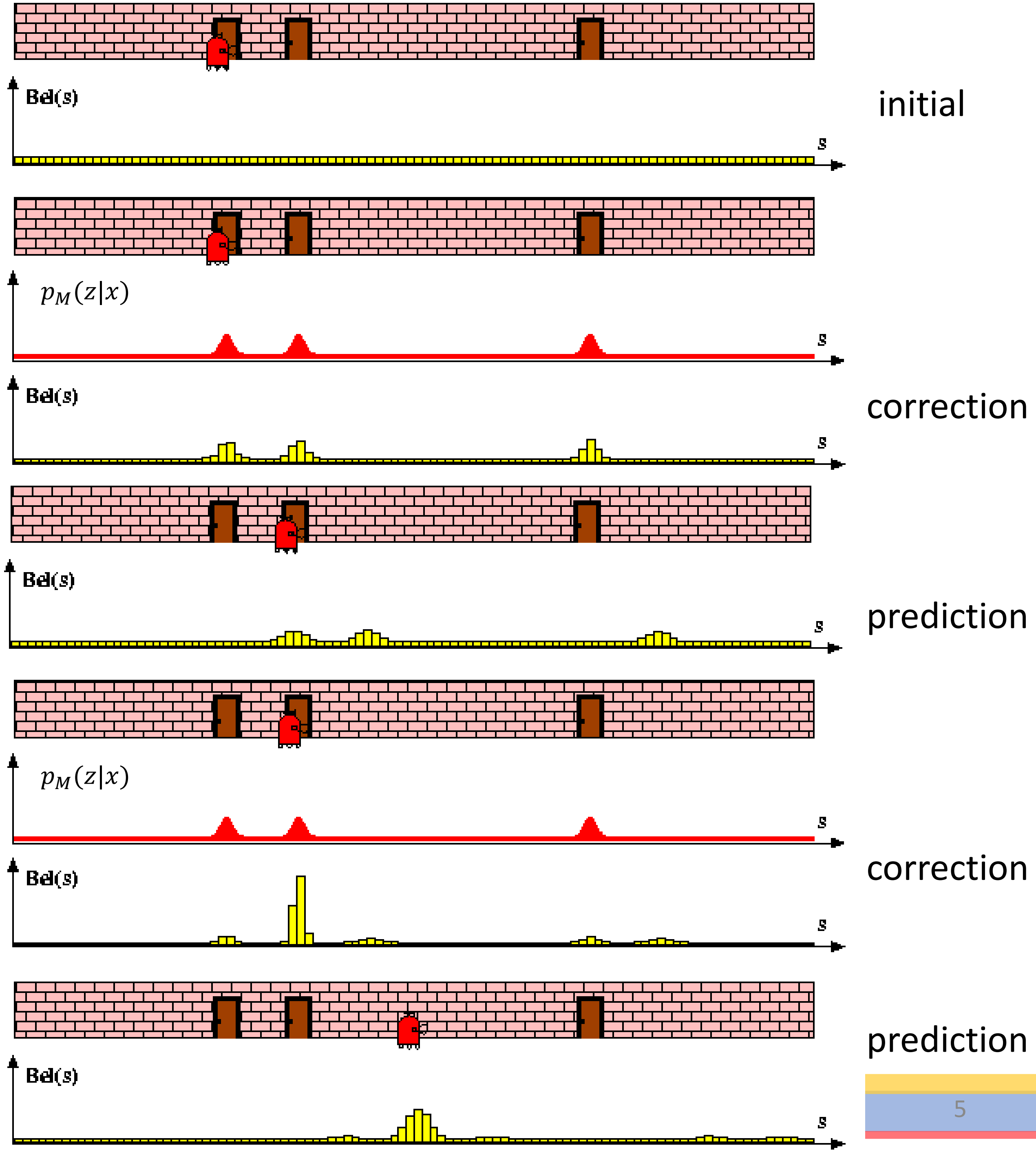
$$bel(x_t) = \eta\, p(z_t|x_t)\, \overline{bel}(x_t)$$

end for

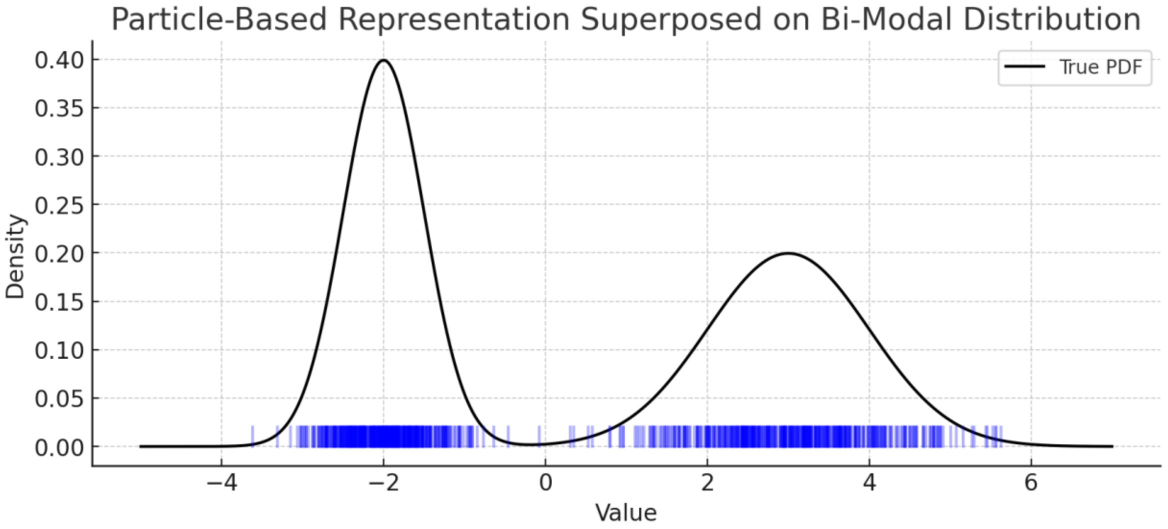
return $bel(x_t)$



Grid localization, $bel(x_t)$ represented by a histogram over grid



Particle Filters



- Belief represented by finite number of parameters or particles
- Advantages
  - The representation is approximate and **nonparametric** and therefore can represent a broader set of distributions e.g., bimodal distributions
  - Can handle nonlinear transformations, e.g., under motion and measurements
- Related ideas: Monte Carlo filter, Survival of the fittest, Condensation, Bootstrap filter, Filtering: [Rubin, 88], [Gordon '93], [Kitagawa 96], Dynamic Bayesian Networks: [Kanazawa '95]

Particle filtering algorithm

$X_t \coloneqq \{x_t^{[1]}, x_t^{[2]}, \ldots x_t^{[M]}\}$ set of particles

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

$\bar{X}_t = X_t = \emptyset$

for all m in [M] do:

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

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

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

for all m in [M] do:

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

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

return X_t

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

$\bar{X}_t$ is the temporary particle set

sampling new particles using motion model p_D

calculates *importance factor* w_t or weight according to measurement p_M

before resampling particles in $\bar{X}_t$ distributed ~ $\overline{bel}(x_t)$

after resampling particles X_t distributed ~ $bel(x_t) = \eta\, p\left(z_t \middle| x_t^{[m]}\right)\ \overline{bel}(x_t)$

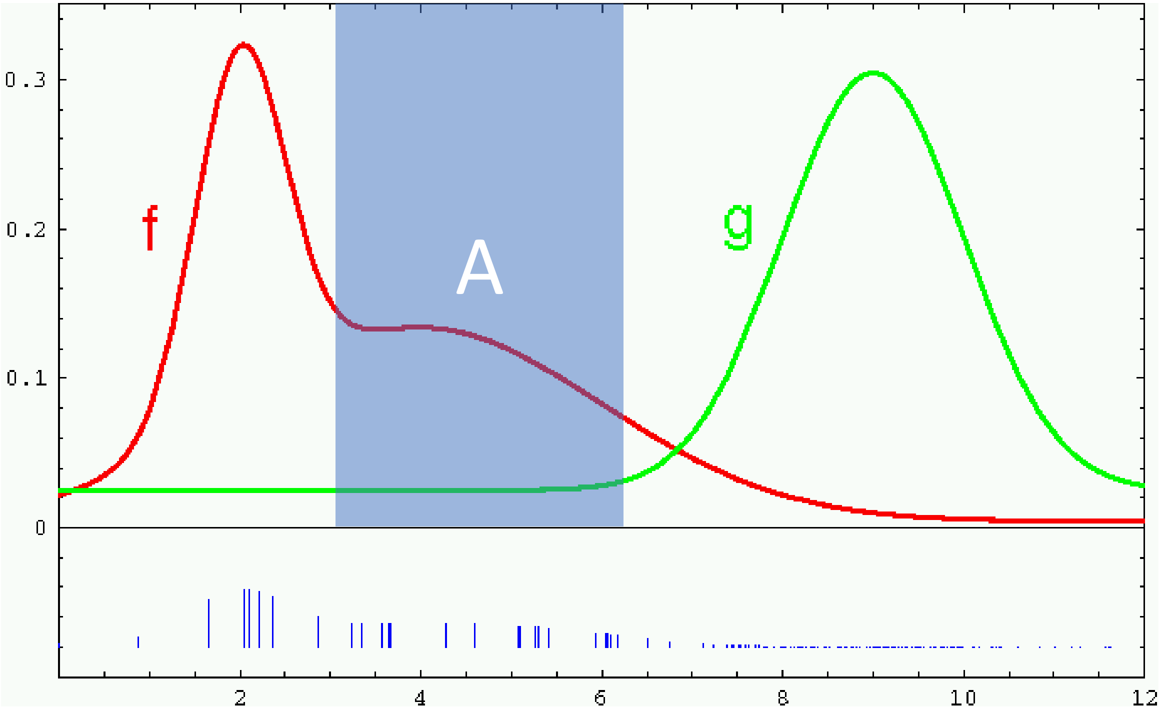
survival of fittest: moves/adds particles to parts of the state space with higher probability, lower probability particles are eliminated

Importance Sampling

For the particle filter f is $bel(x_t)$ and g corresponds to $\overline{bel}(x_t)$

suppose we want to compute $P_f(x \in A) = E_f[I(x \in A)]$

but we can only sample according to 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 $f(x) > 0 \Rightarrow g(x) > 0$

The ratio w(x) = f(x) / g(x) **is the weight of the sample**

w(x_t) = $bel(x_t)$ / $\overline{bel}(x_t) \propto p(z_t|x_t)$ Measurement model

Monte Carlo Localization (MCL)

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

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

$\bar{X}_t = X_t = \emptyset$

for all m in [M] do:

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

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

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

for all m in [M] do:

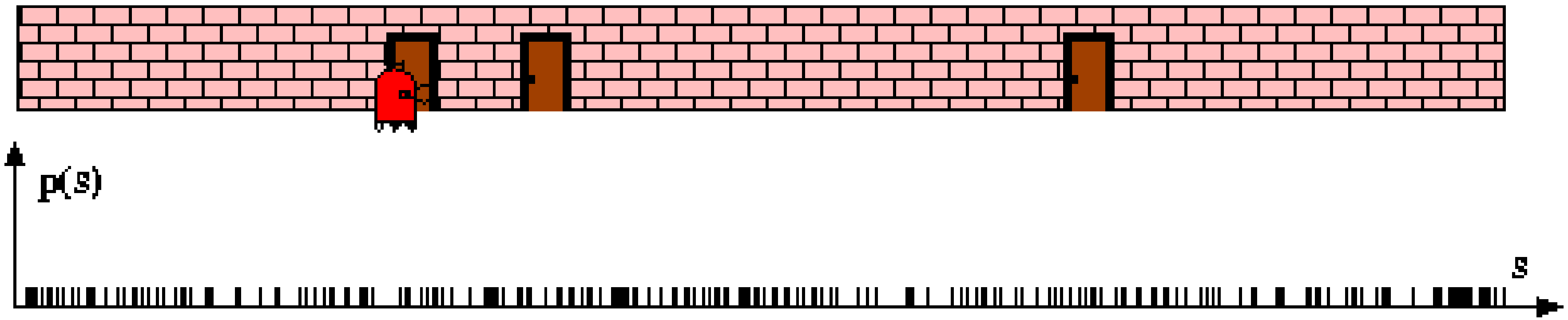
draw $i\ with\ probability\ \propto w_t^{[i]}$

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

return X_t

Plug in motion and measurement models in the particle filter

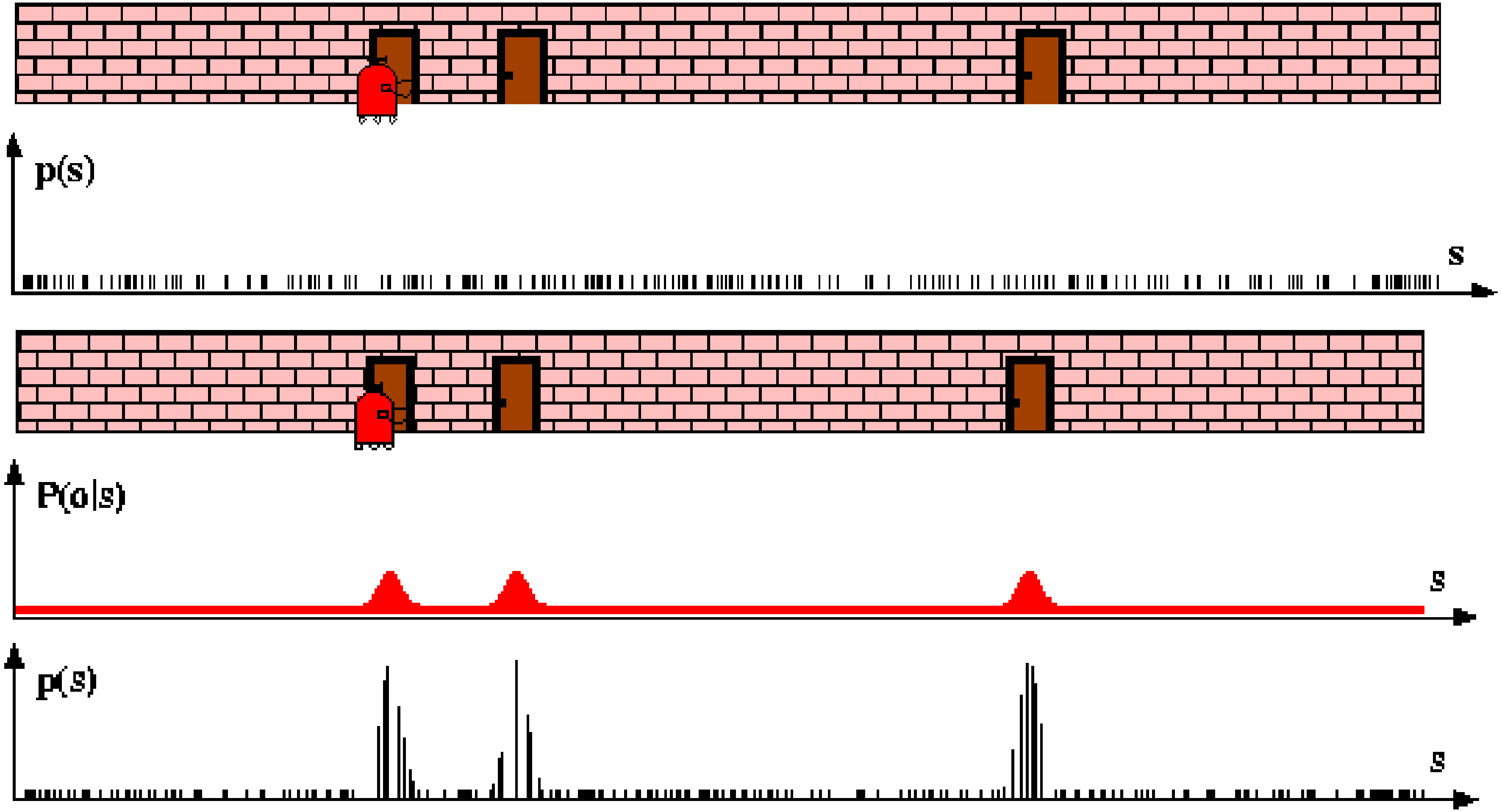
Particle Filters



Sensor Information: Importance Sampling

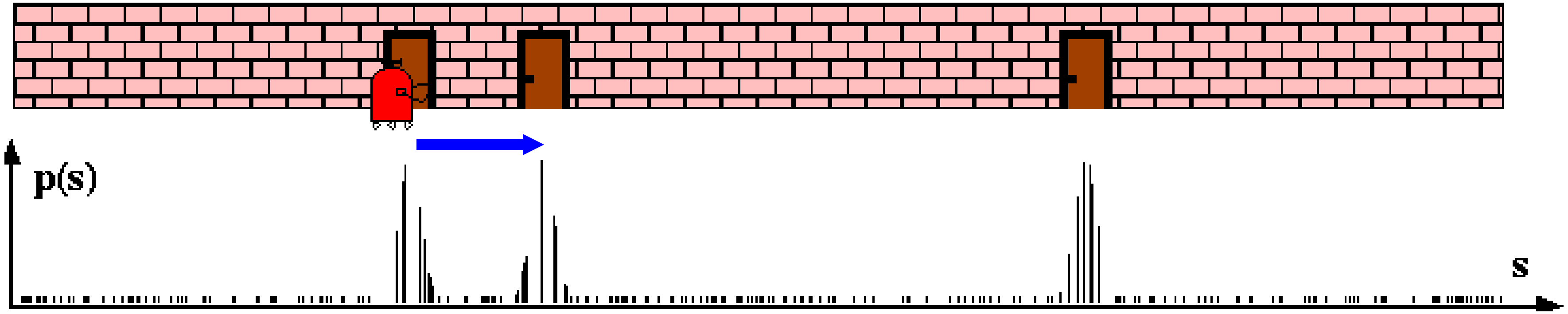
$$Bel(x) \leftarrow \alpha\, p(z \mid x)\, Bel^{-}(x)$$

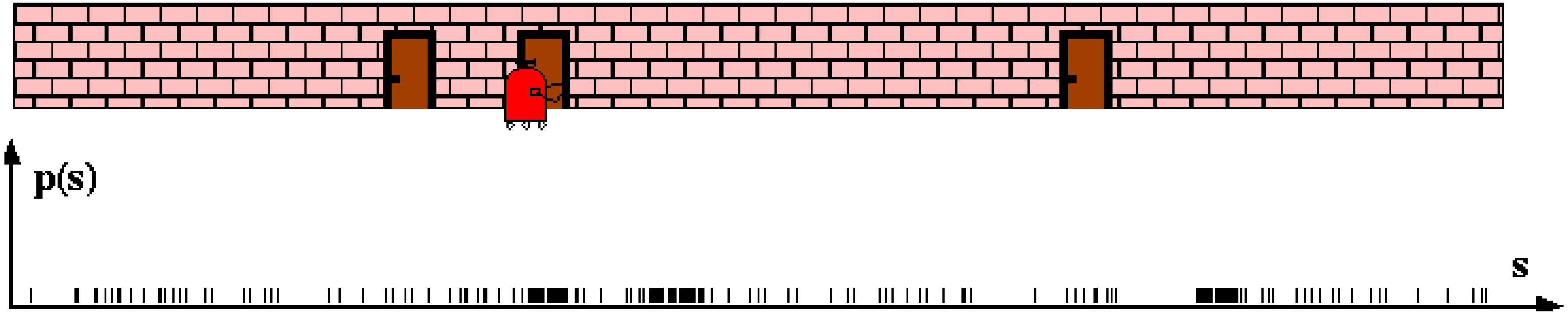
$$w \leftarrow \frac{\alpha\, p(z \mid x)\, Bel^{-}(x)}{Bel^{-}(x)} = \alpha\, p(z \mid x)$$



Robot Motion

$$\overline{Bel}(x) \leftarrow \int p(x \mid u, x')\, Bel(x')\, dx'$$

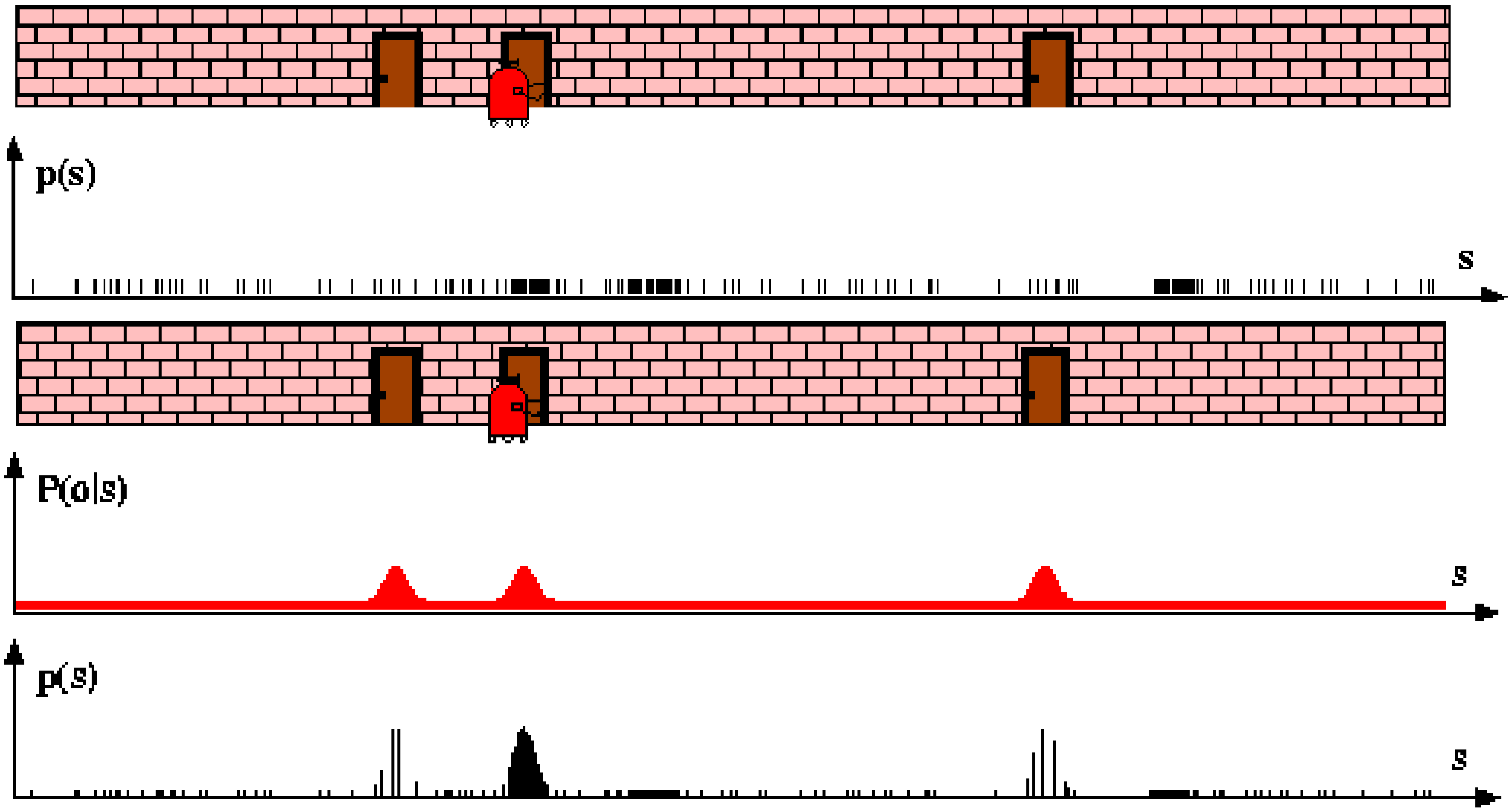




Sensor Information: Importance Sampling

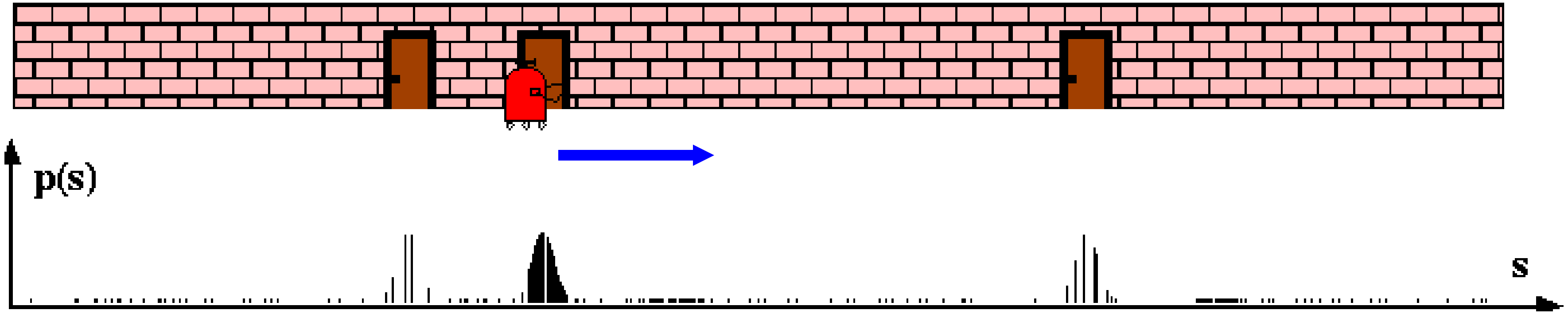
$$Bel(x) \leftarrow \alpha\, p(z \mid x)\, Bel^{-}(x)$$

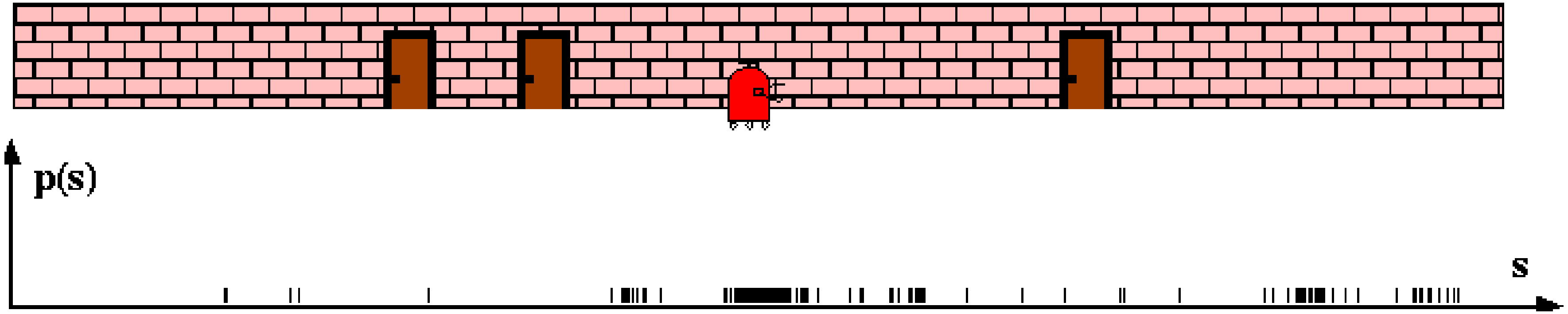
$$w \leftarrow \frac{\alpha\, p(z \mid x)\, Bel^{-}(x)}{Bel^{-}(x)} = \alpha\, p(z \mid x)$$



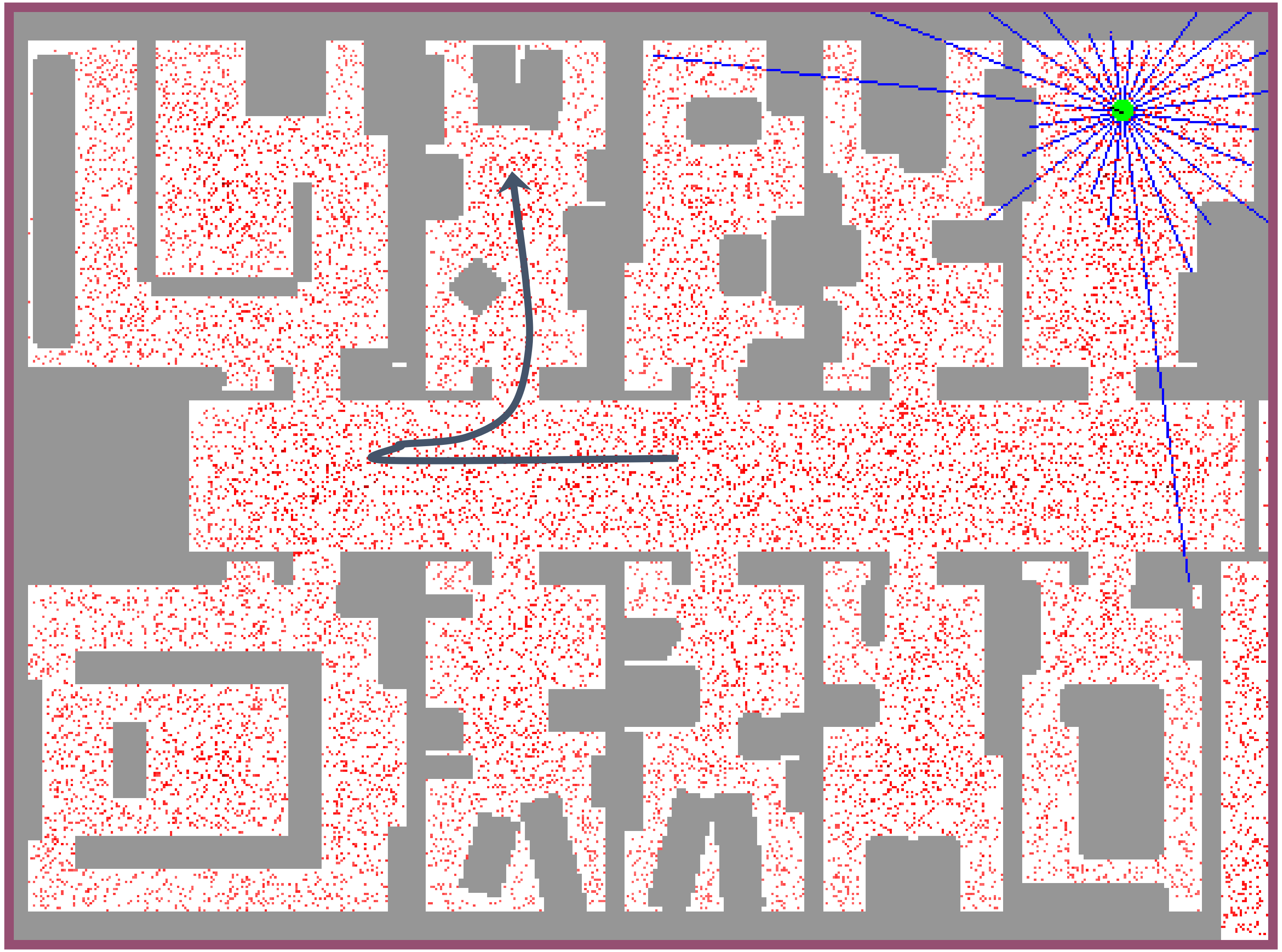
Robot Motion

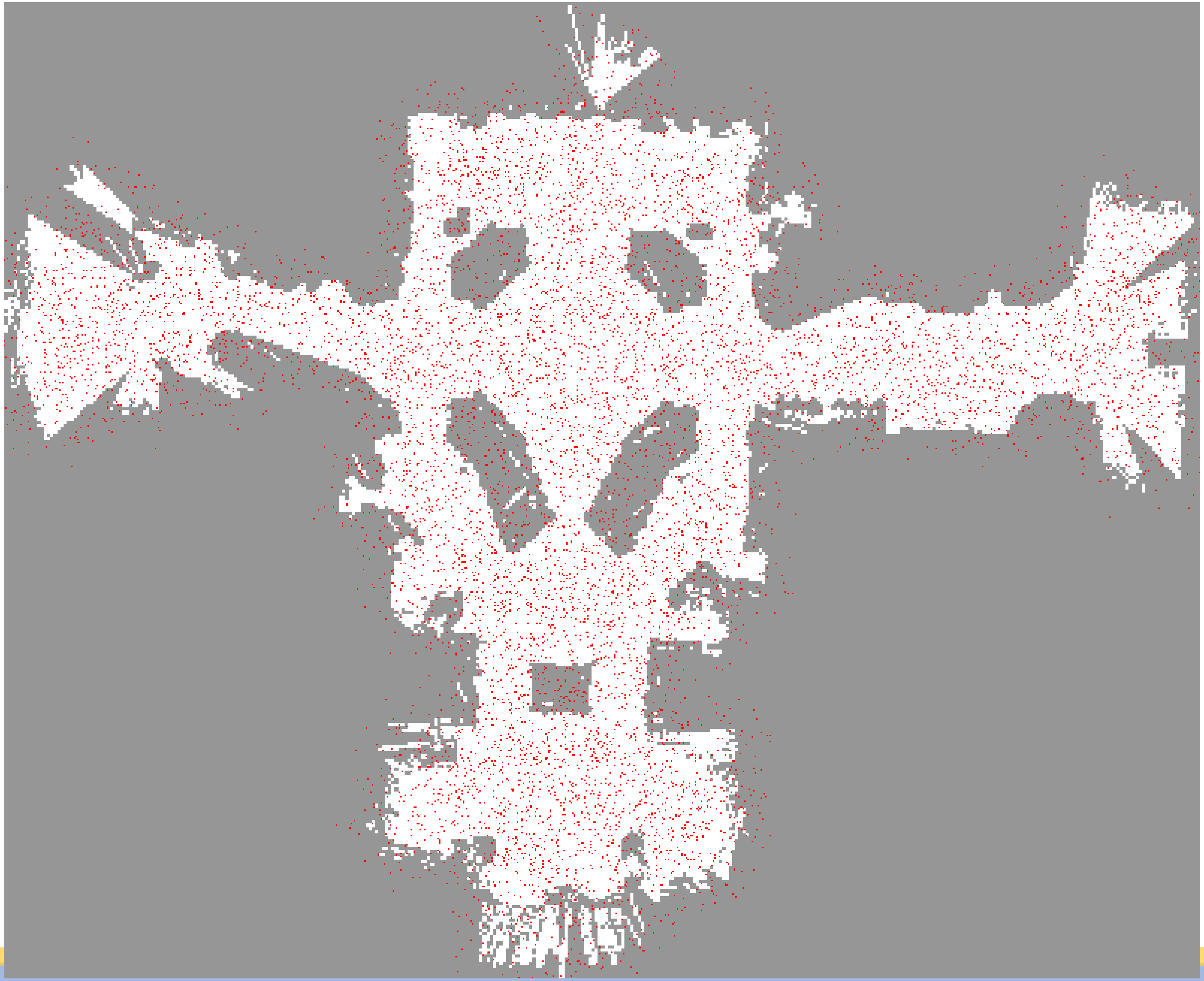
$$\overline{Bel}(x) \leftarrow \int p(x \mid u, x')\, Bel(x')\, dx'$$

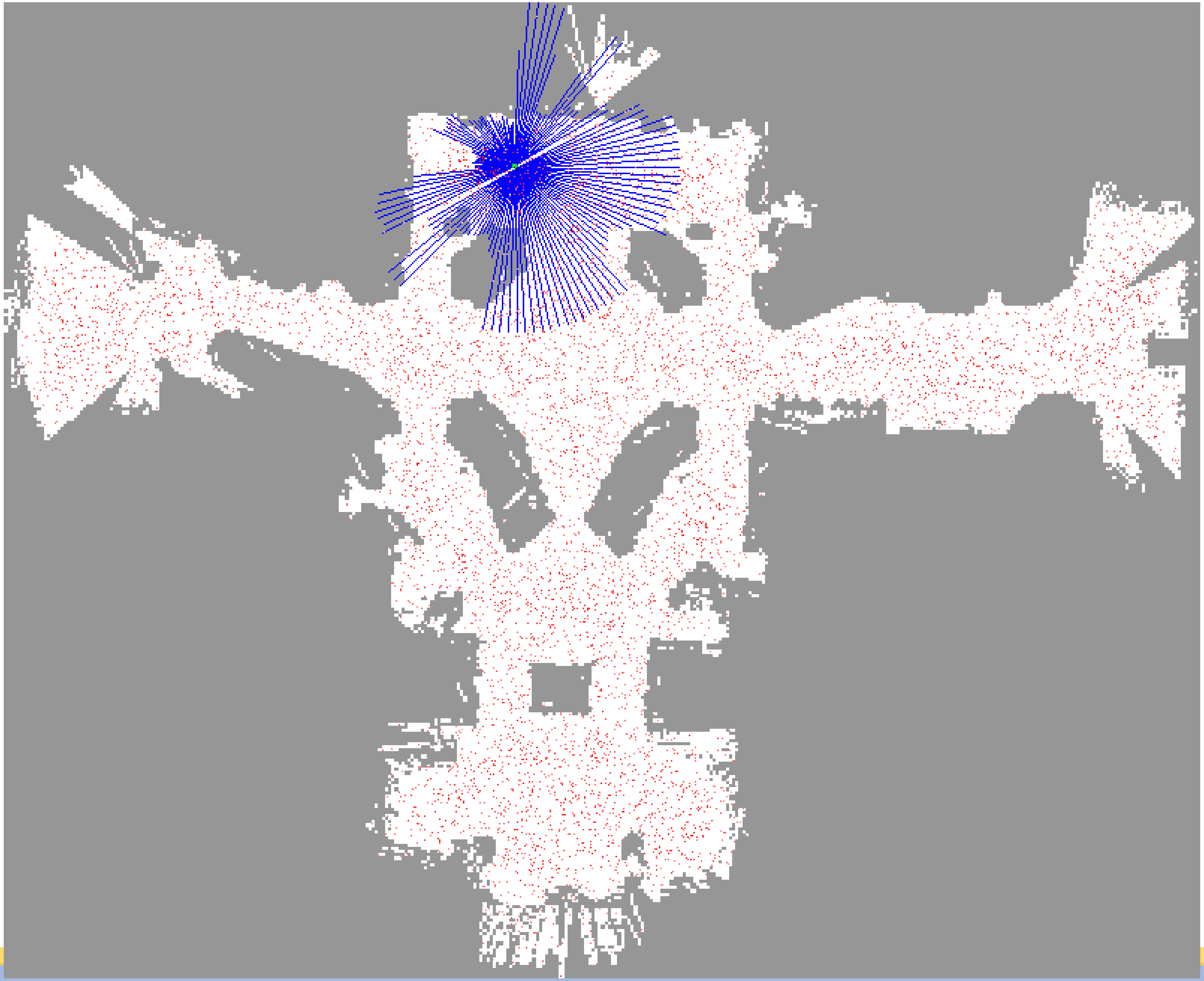


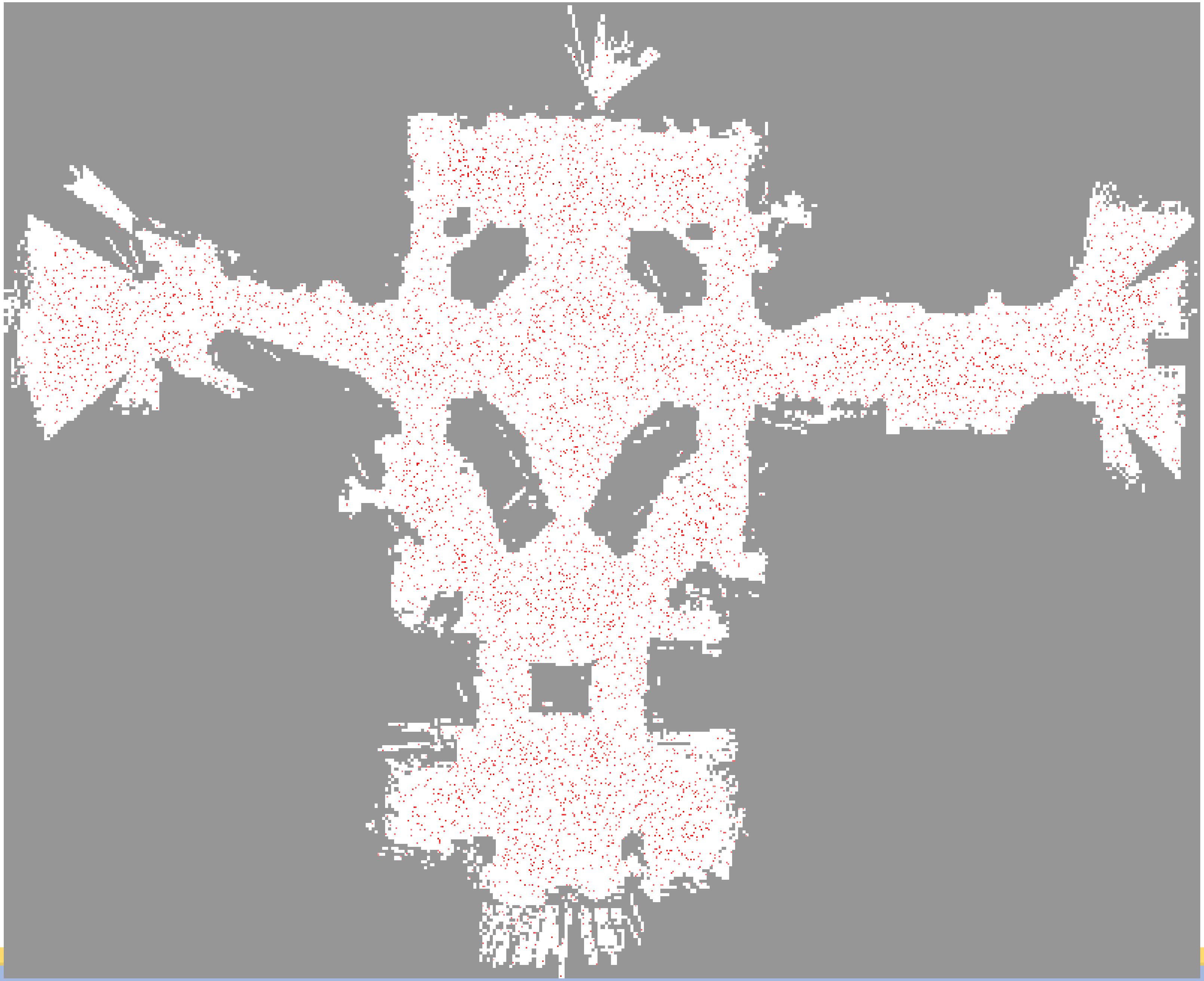


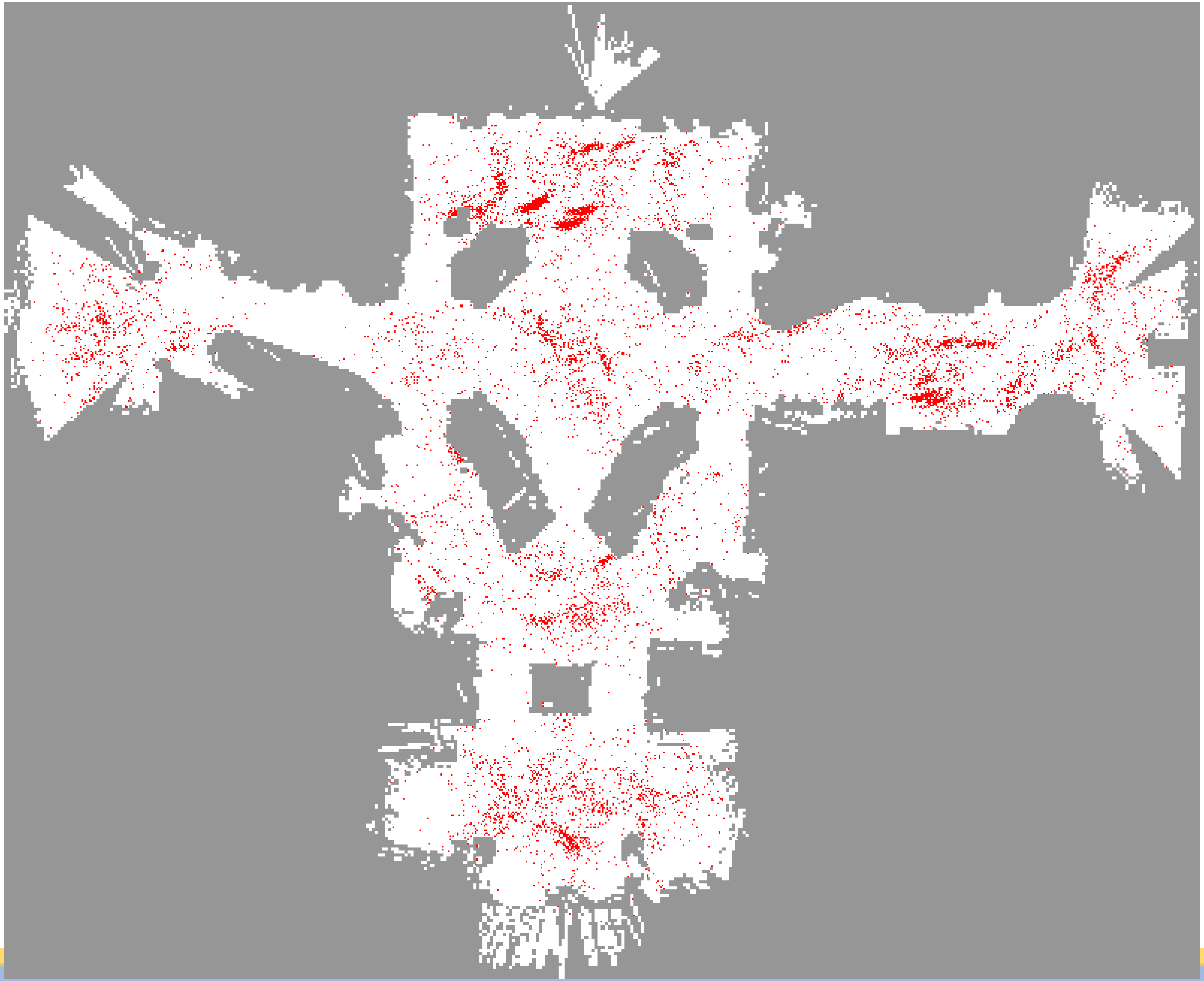
Sample-based Localization (sonar)

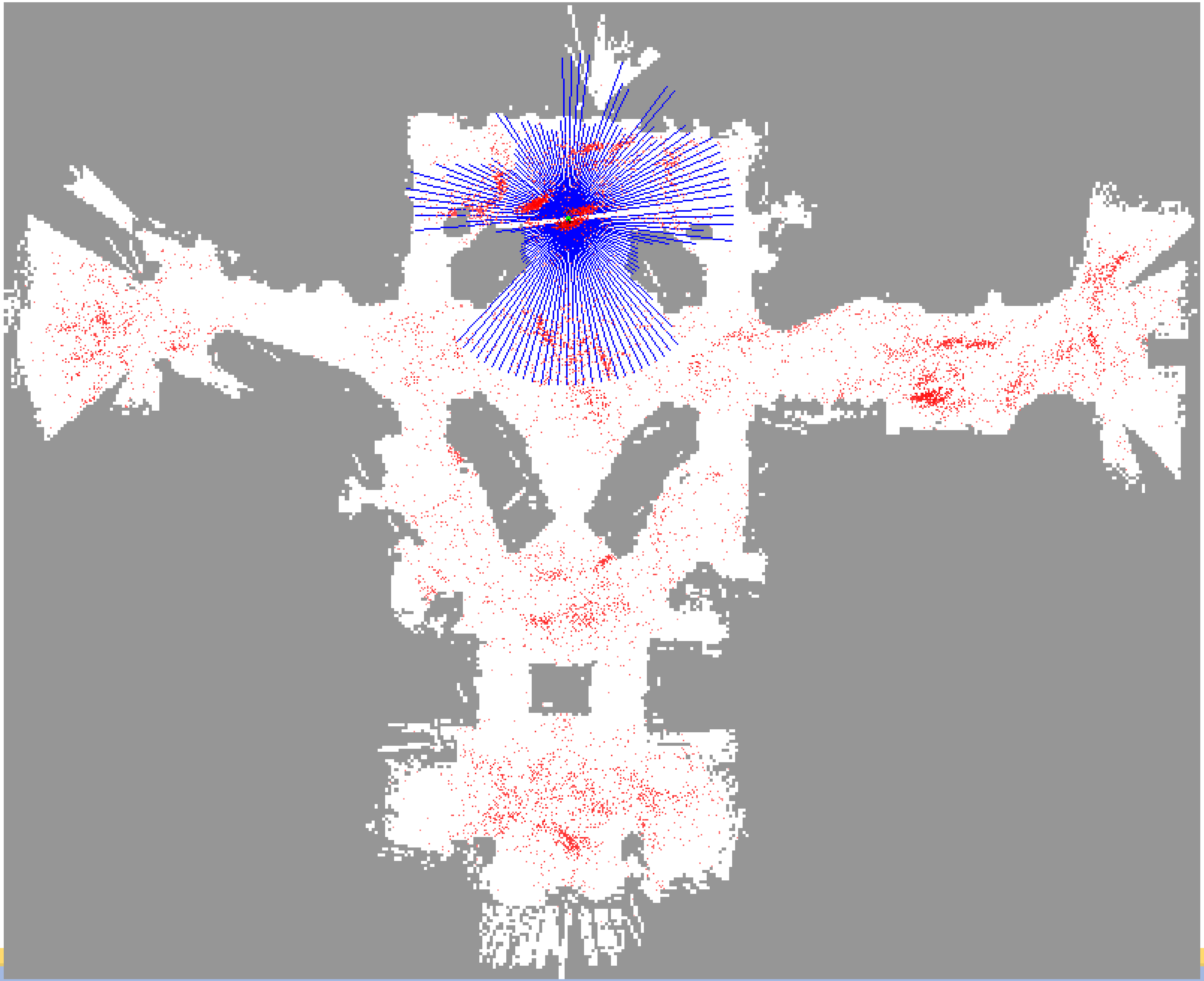


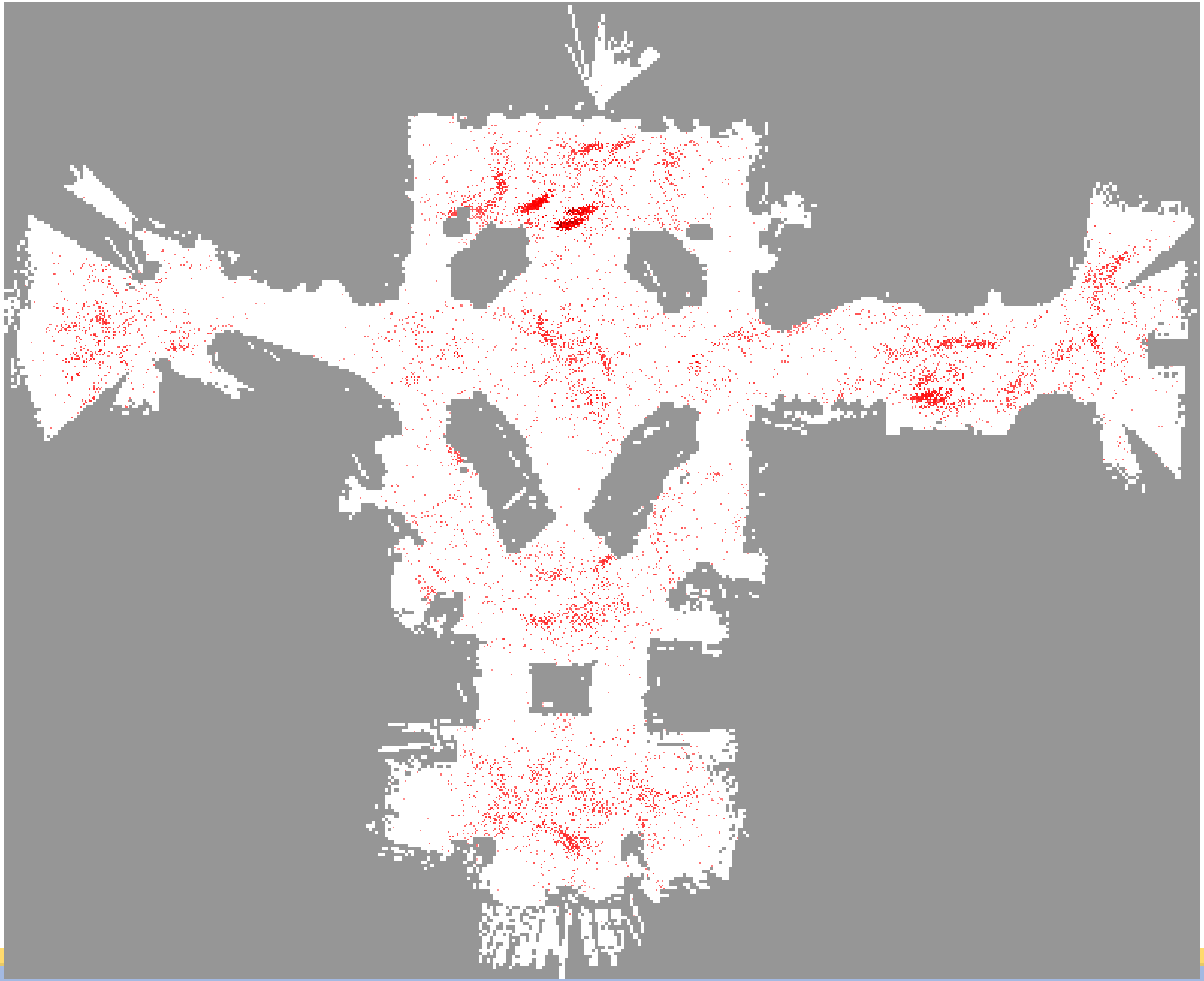


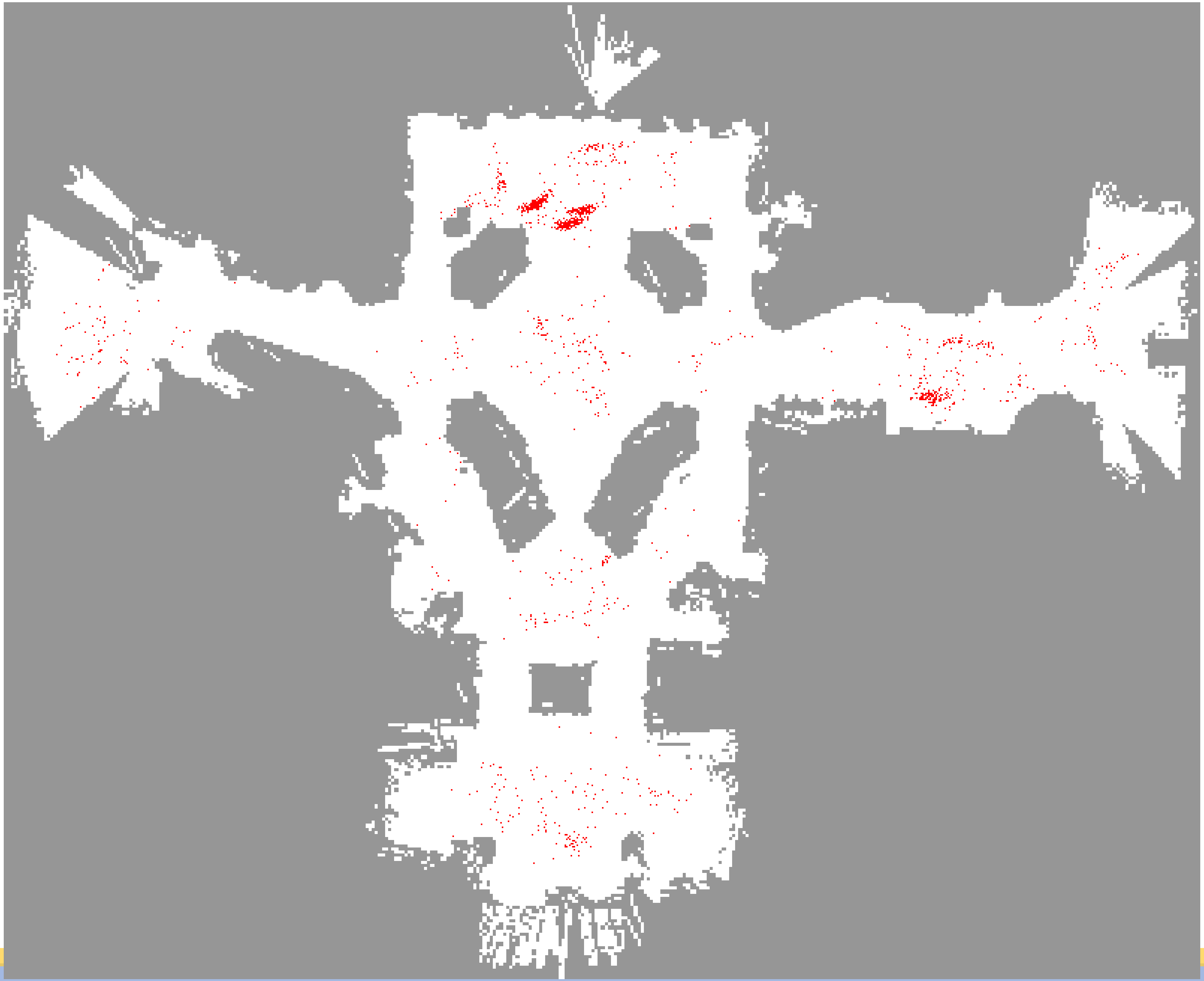


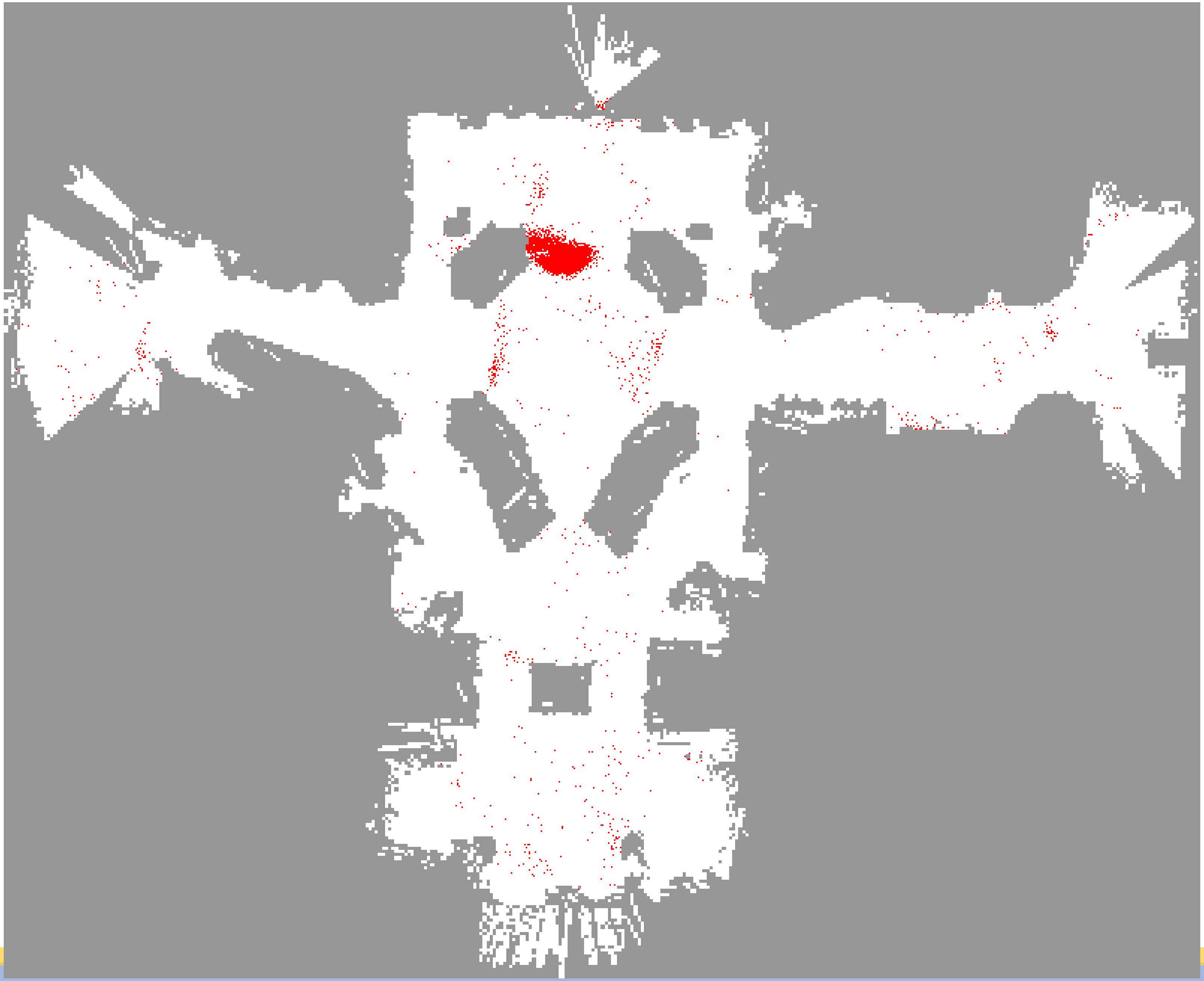


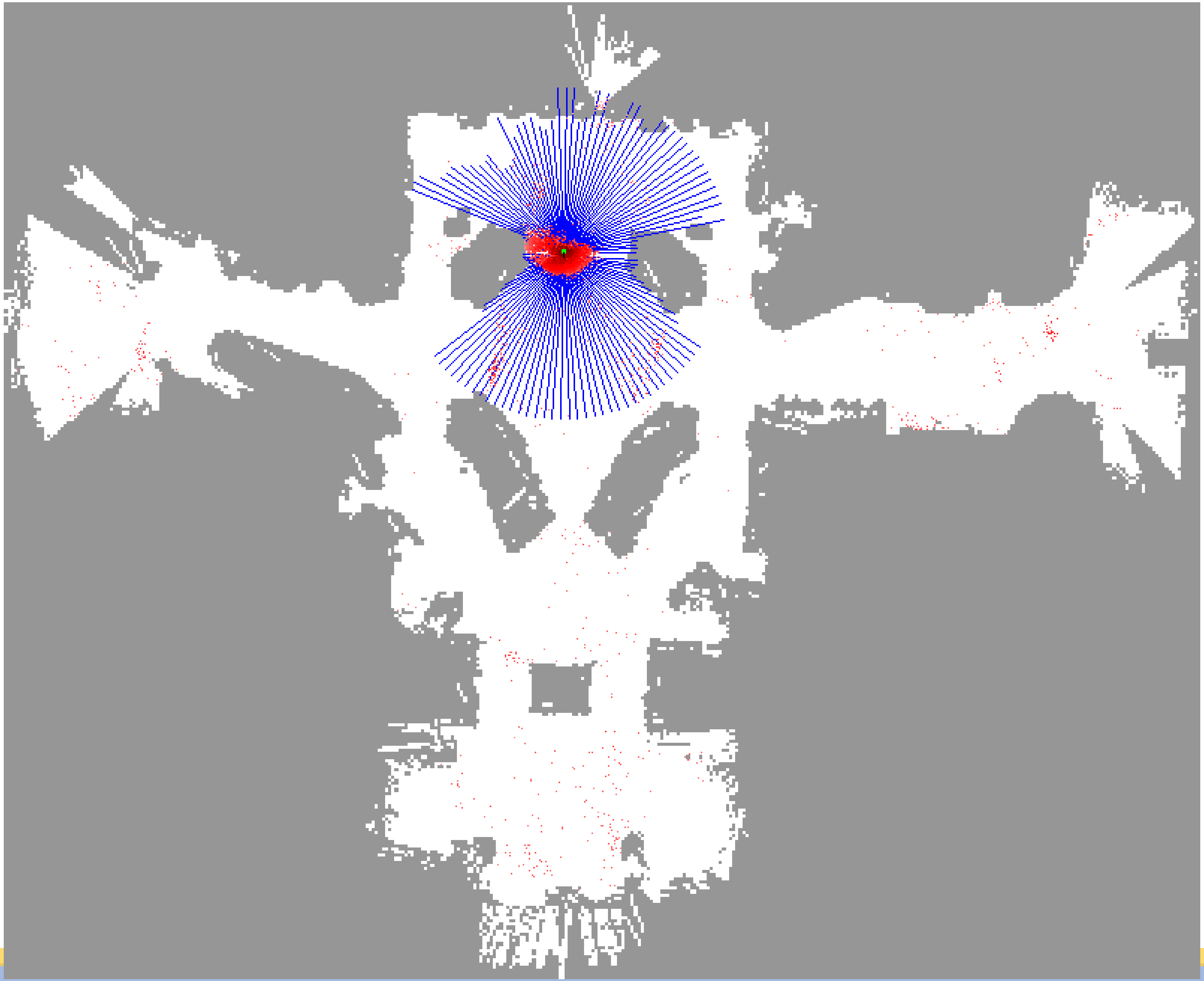


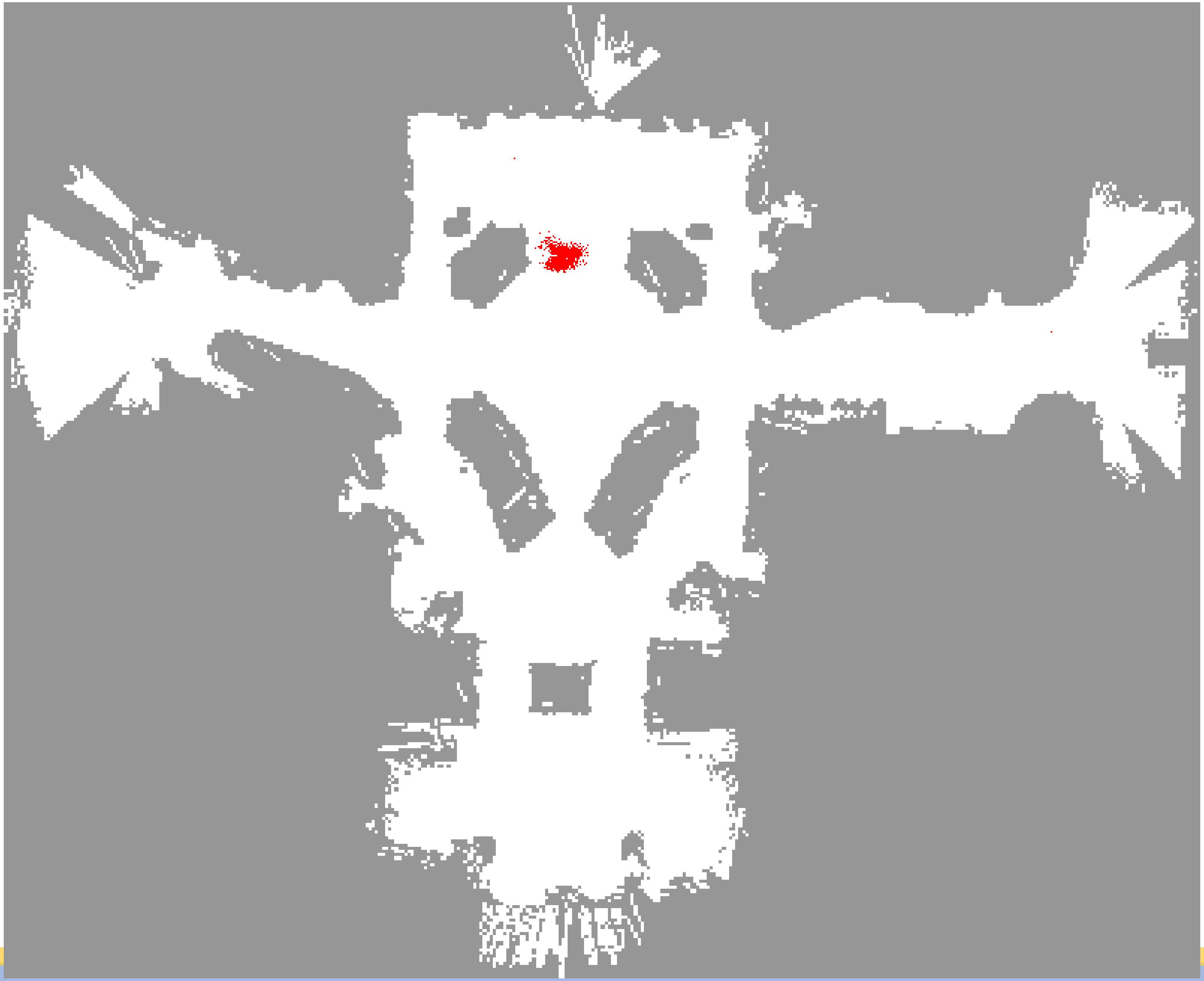


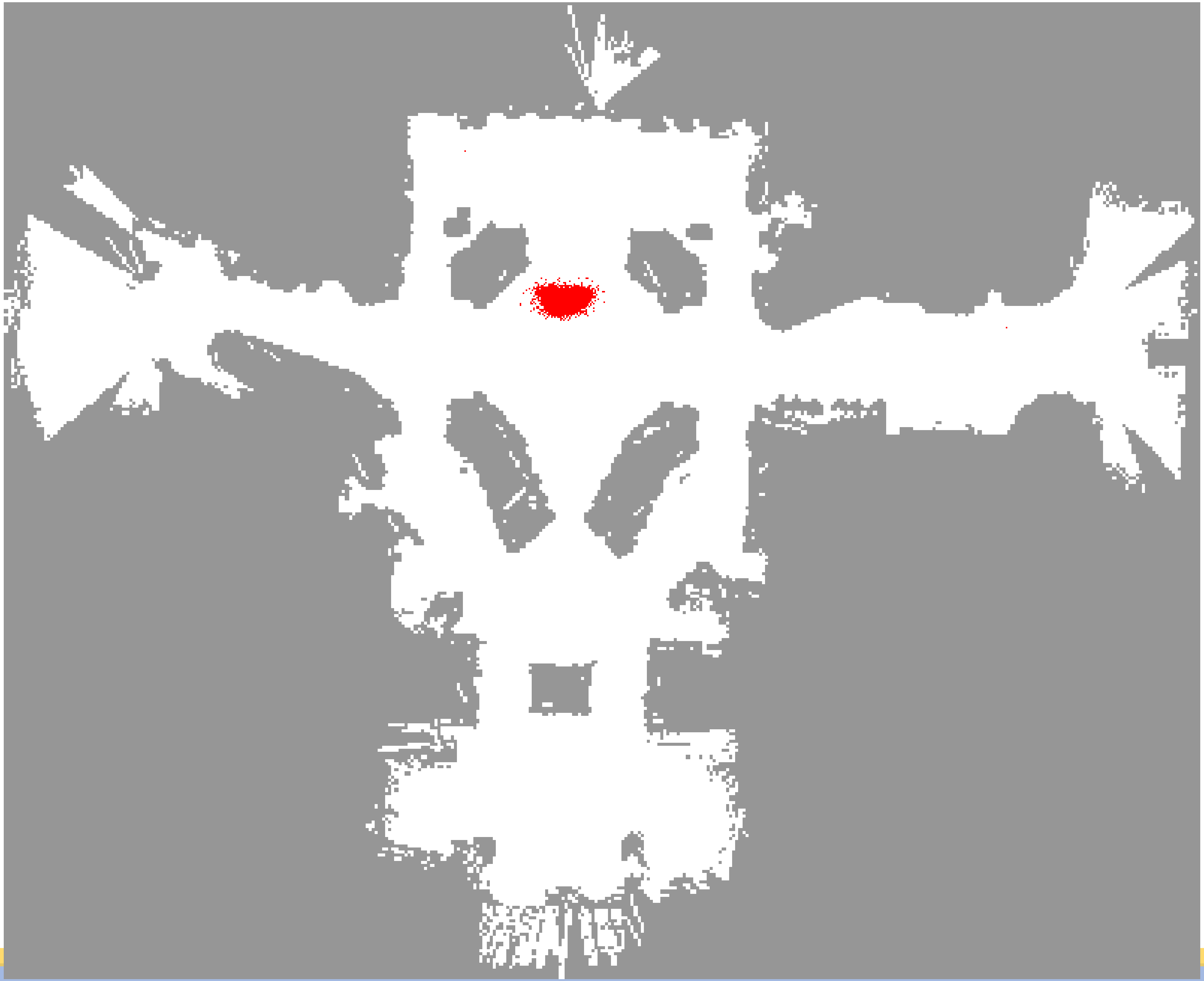


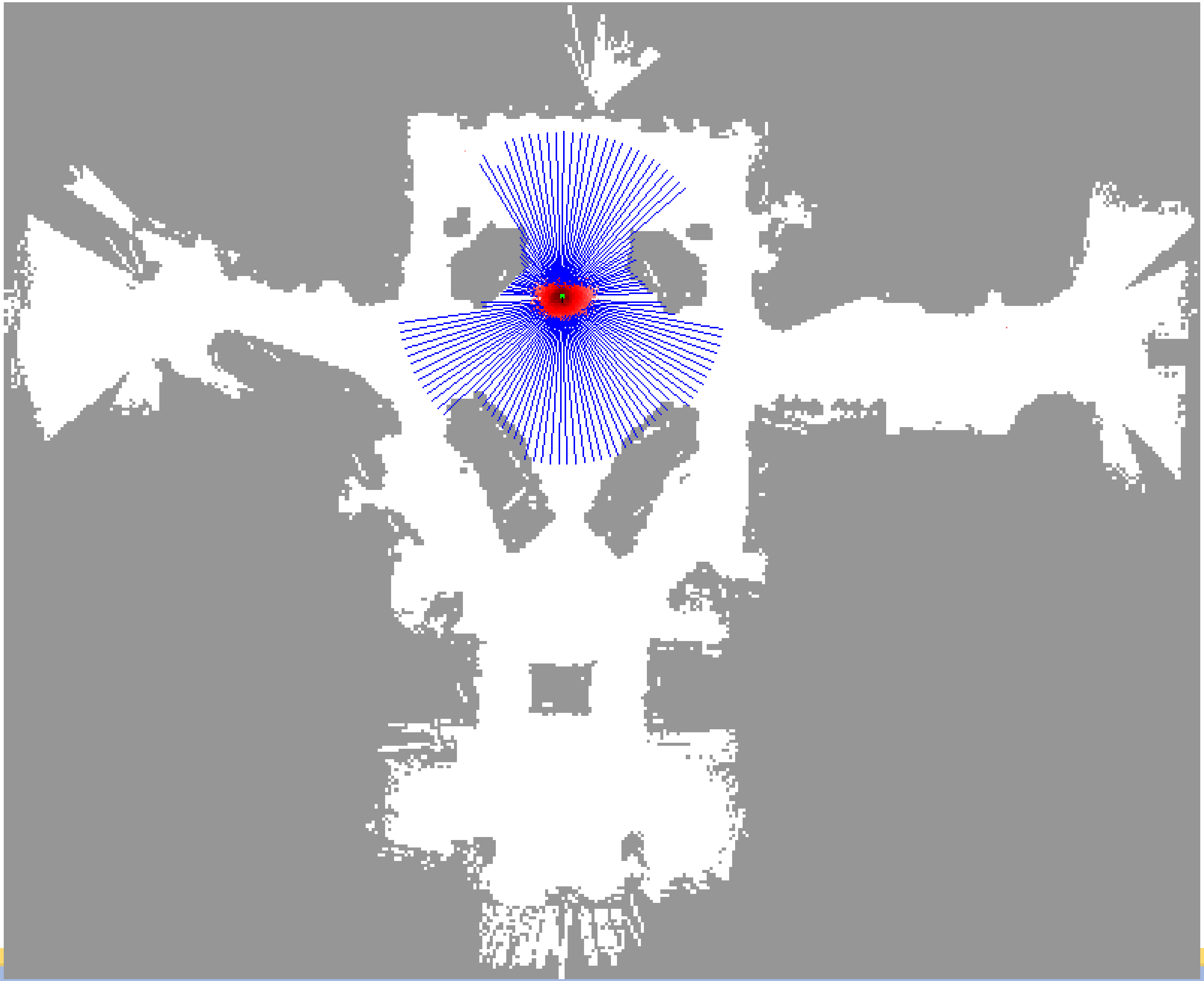


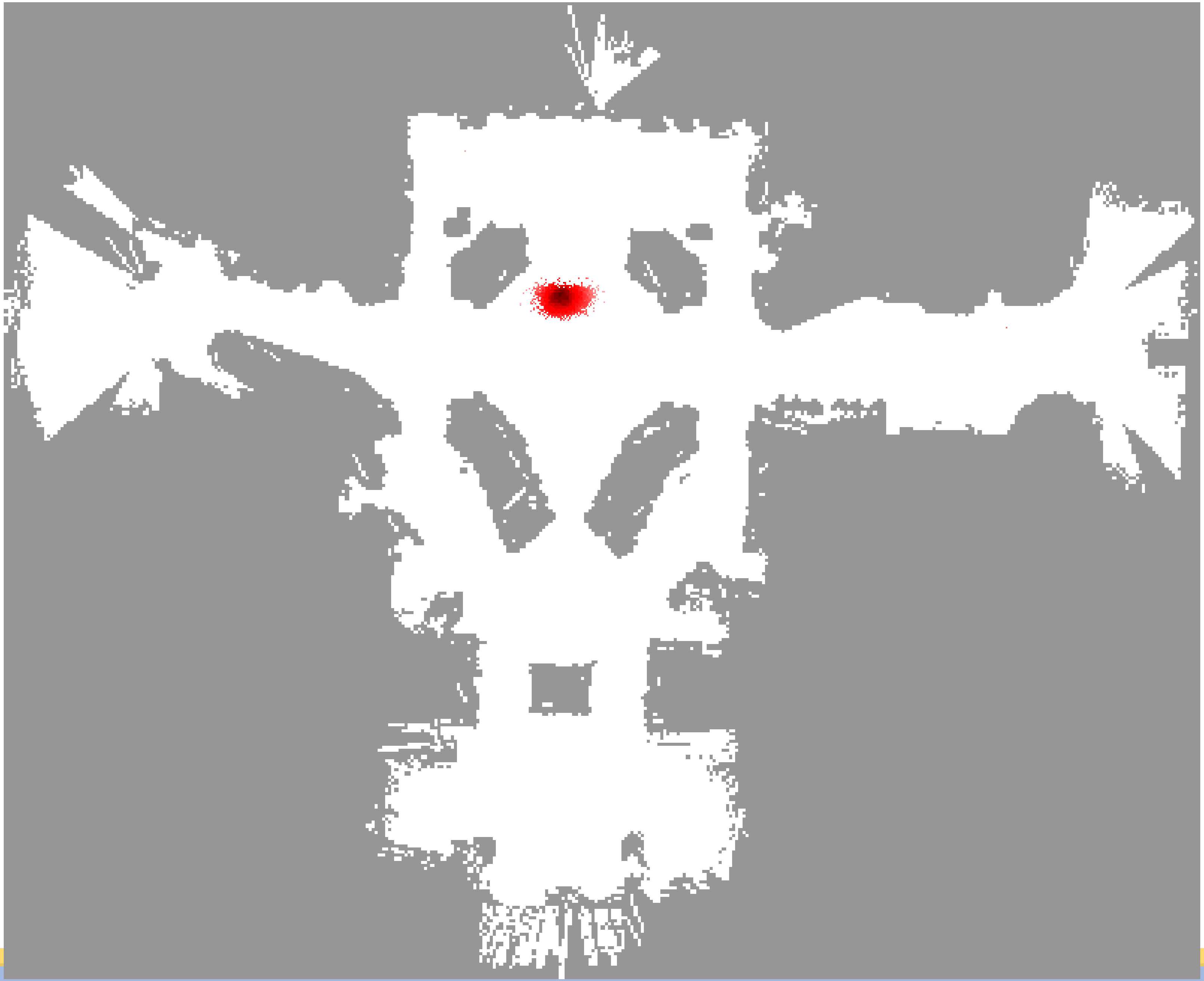


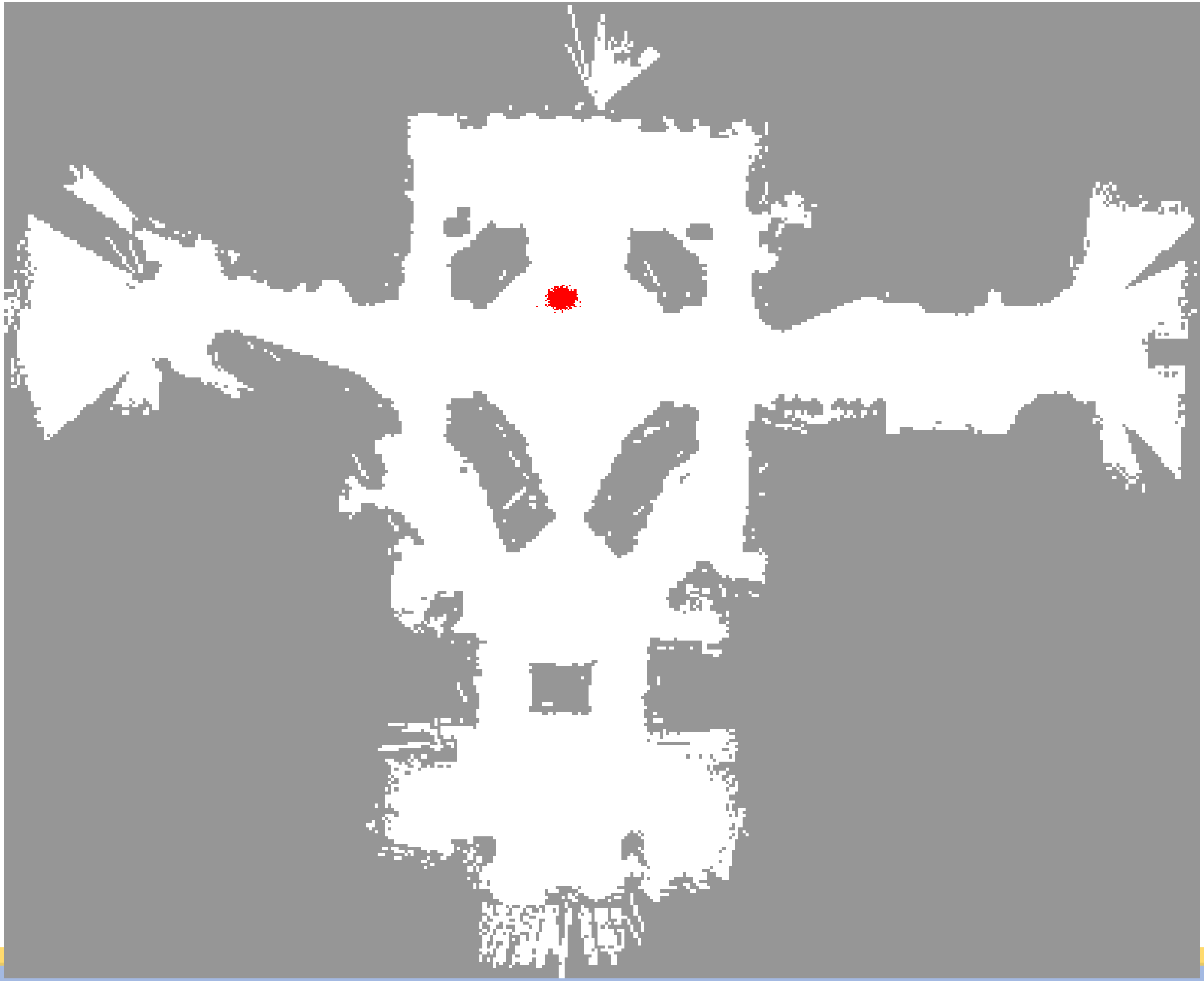


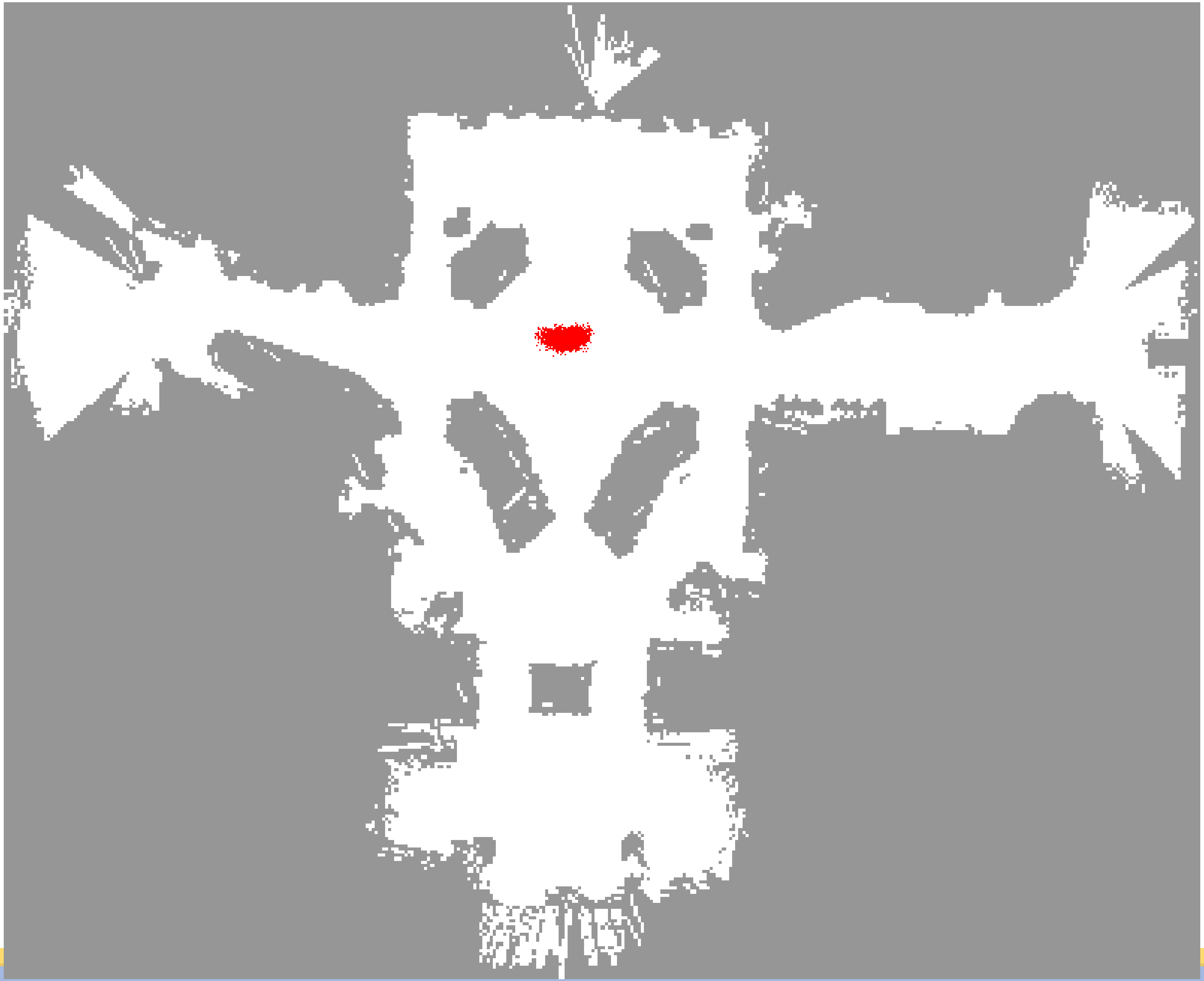


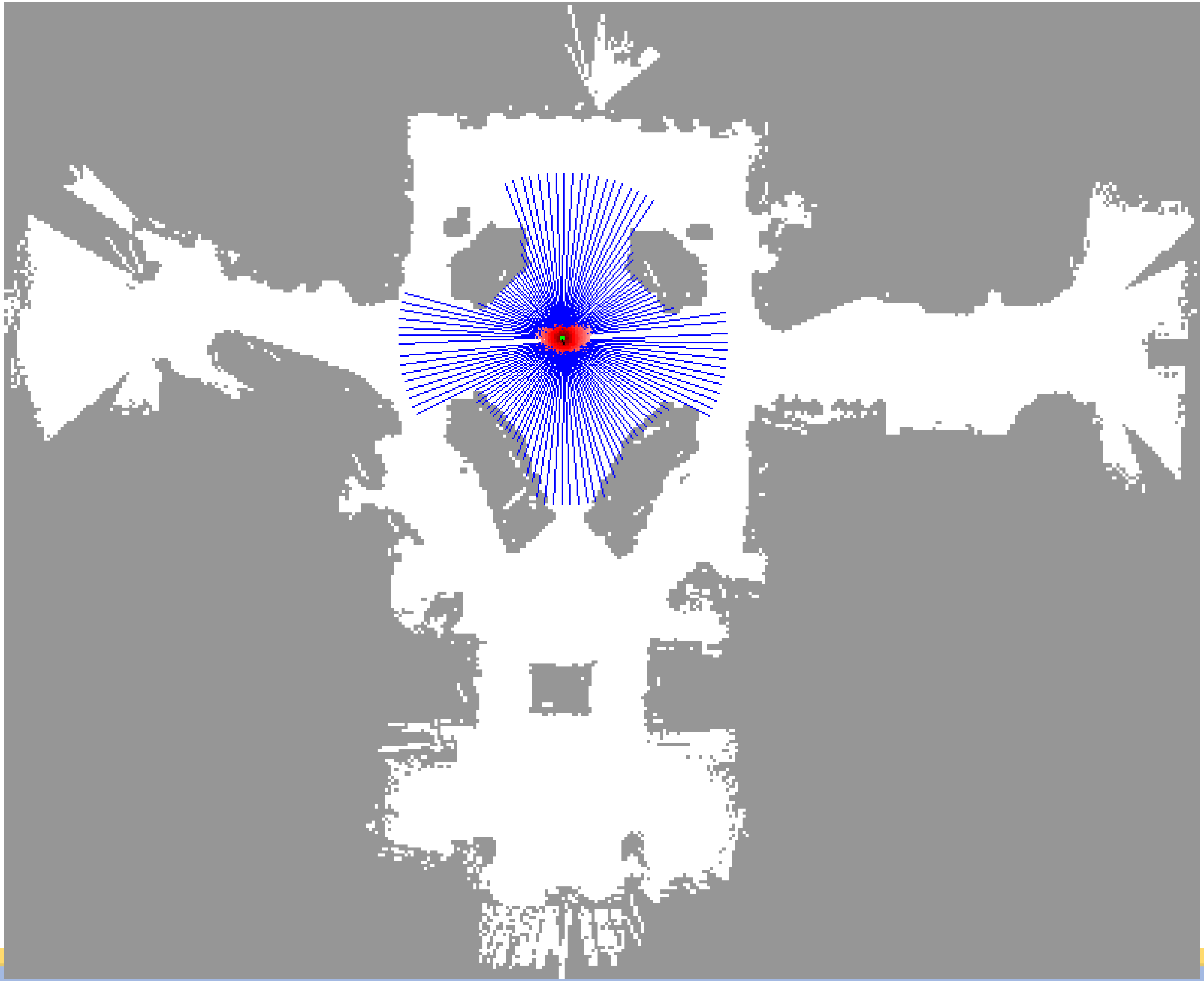


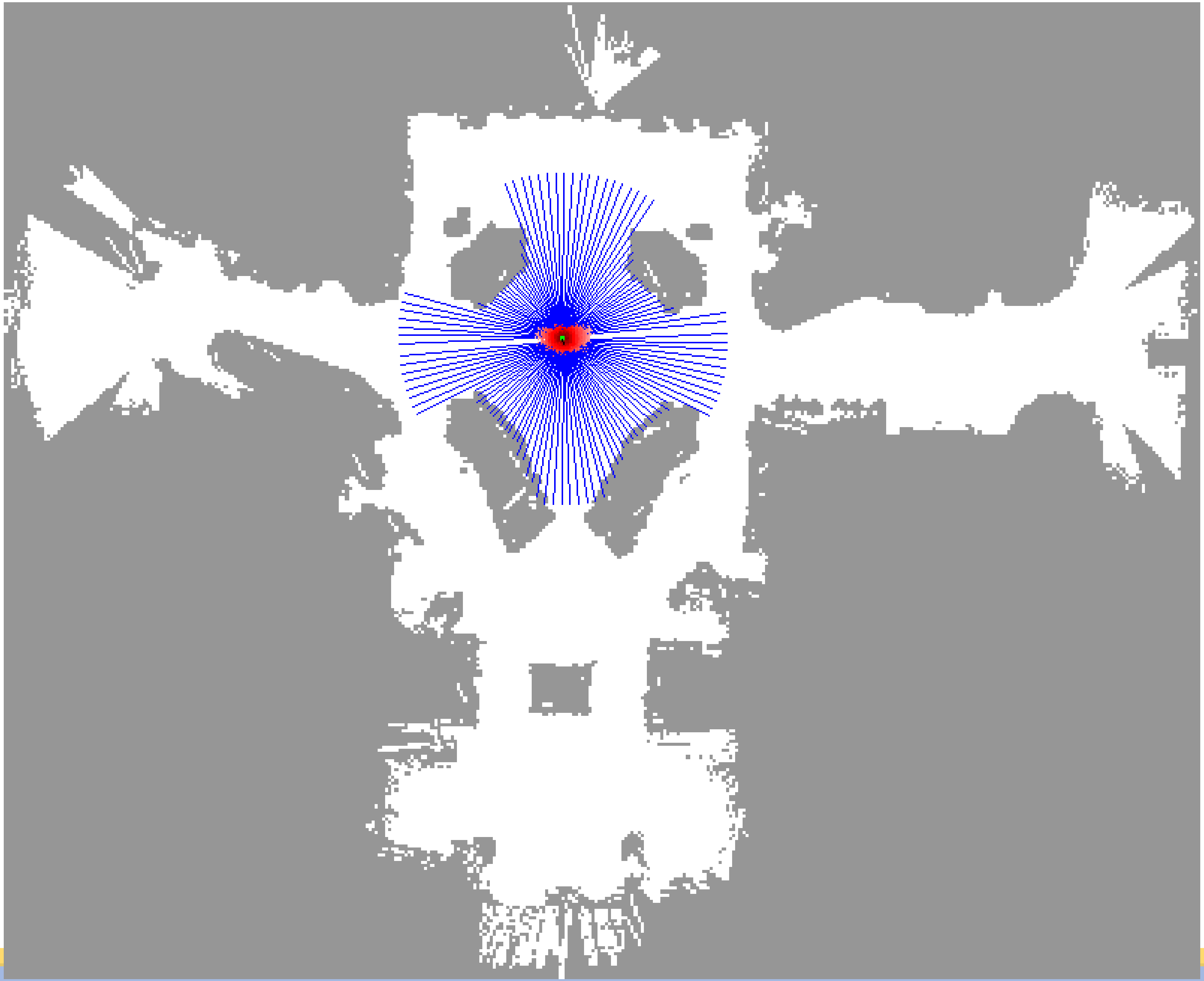




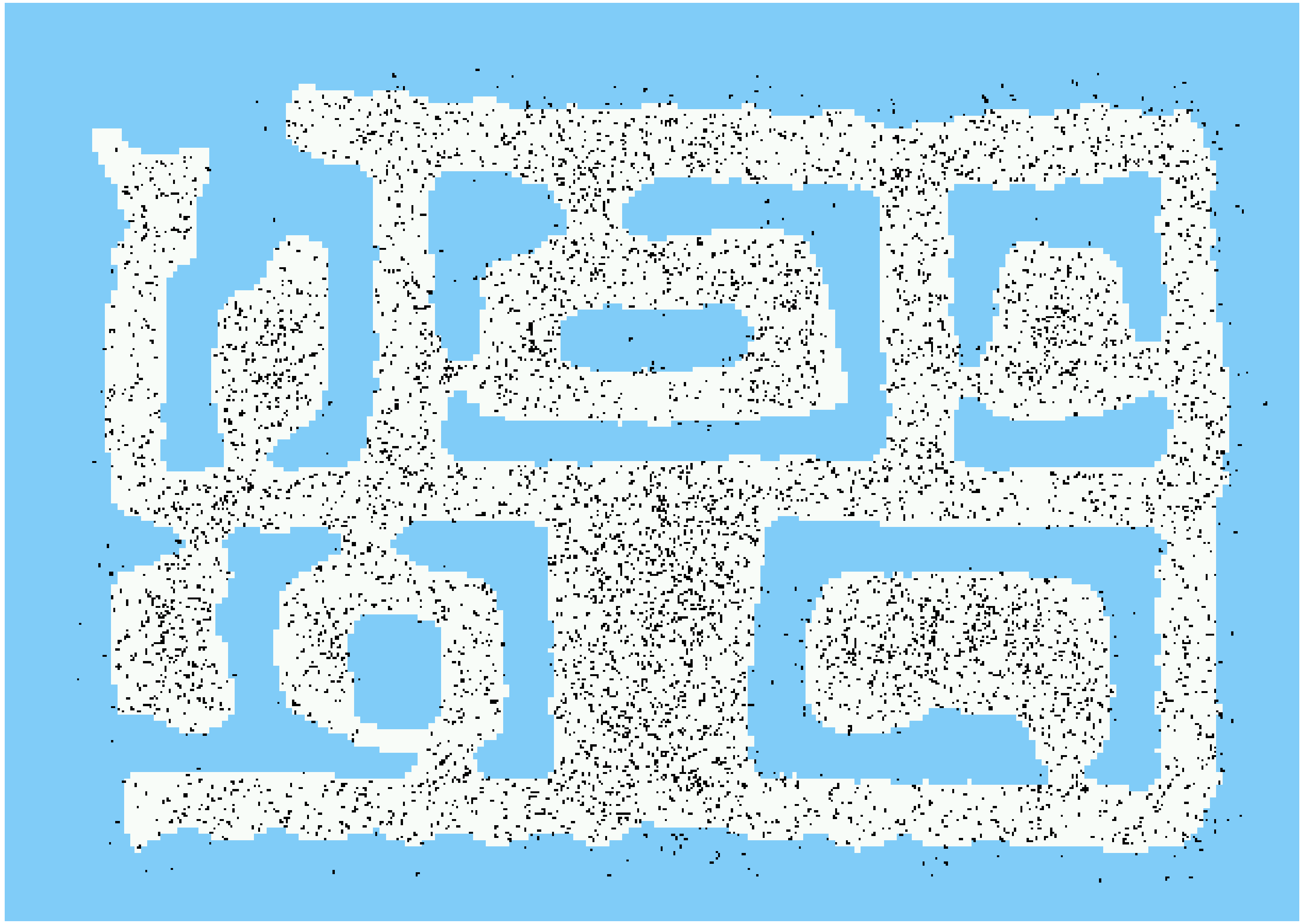




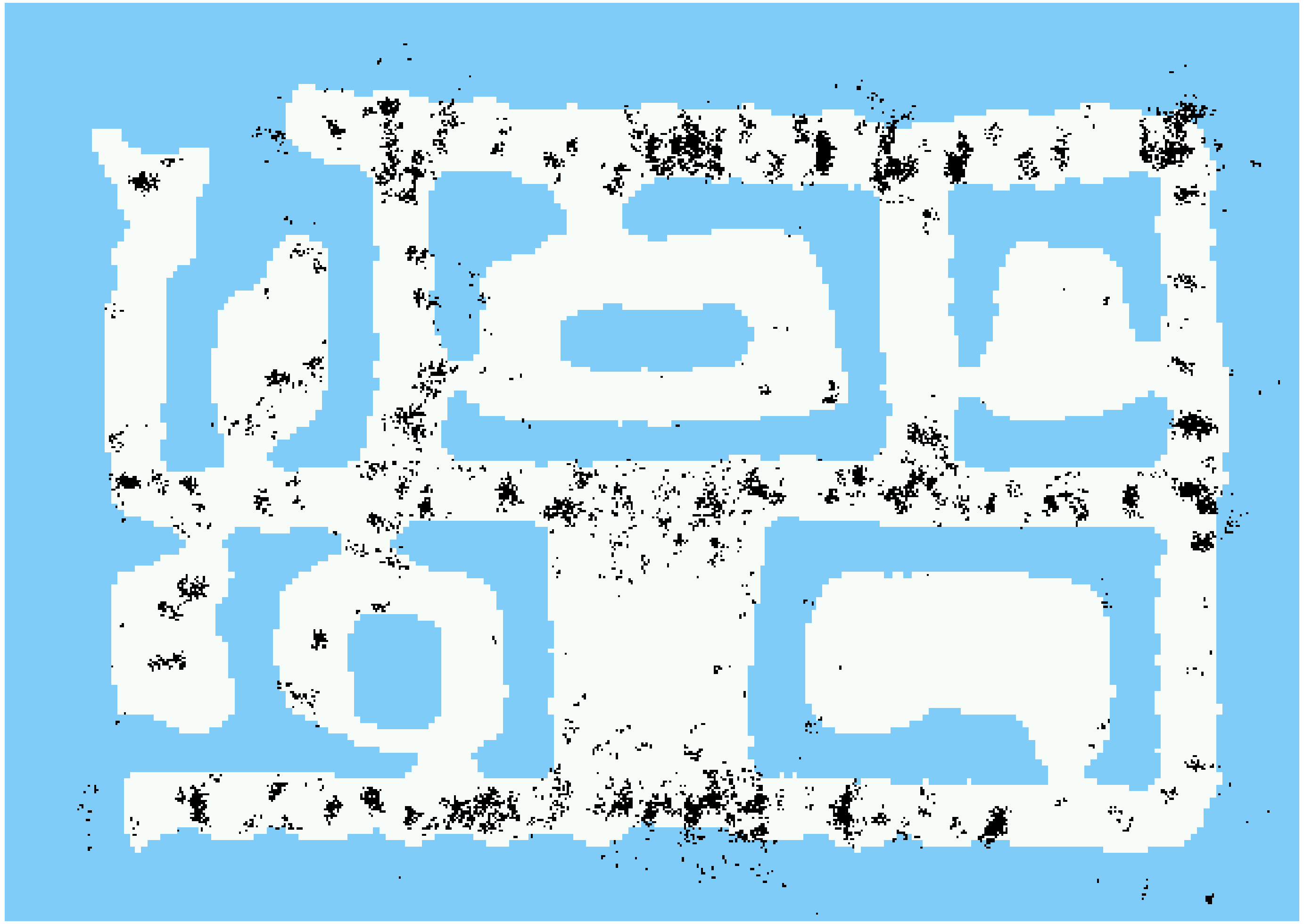




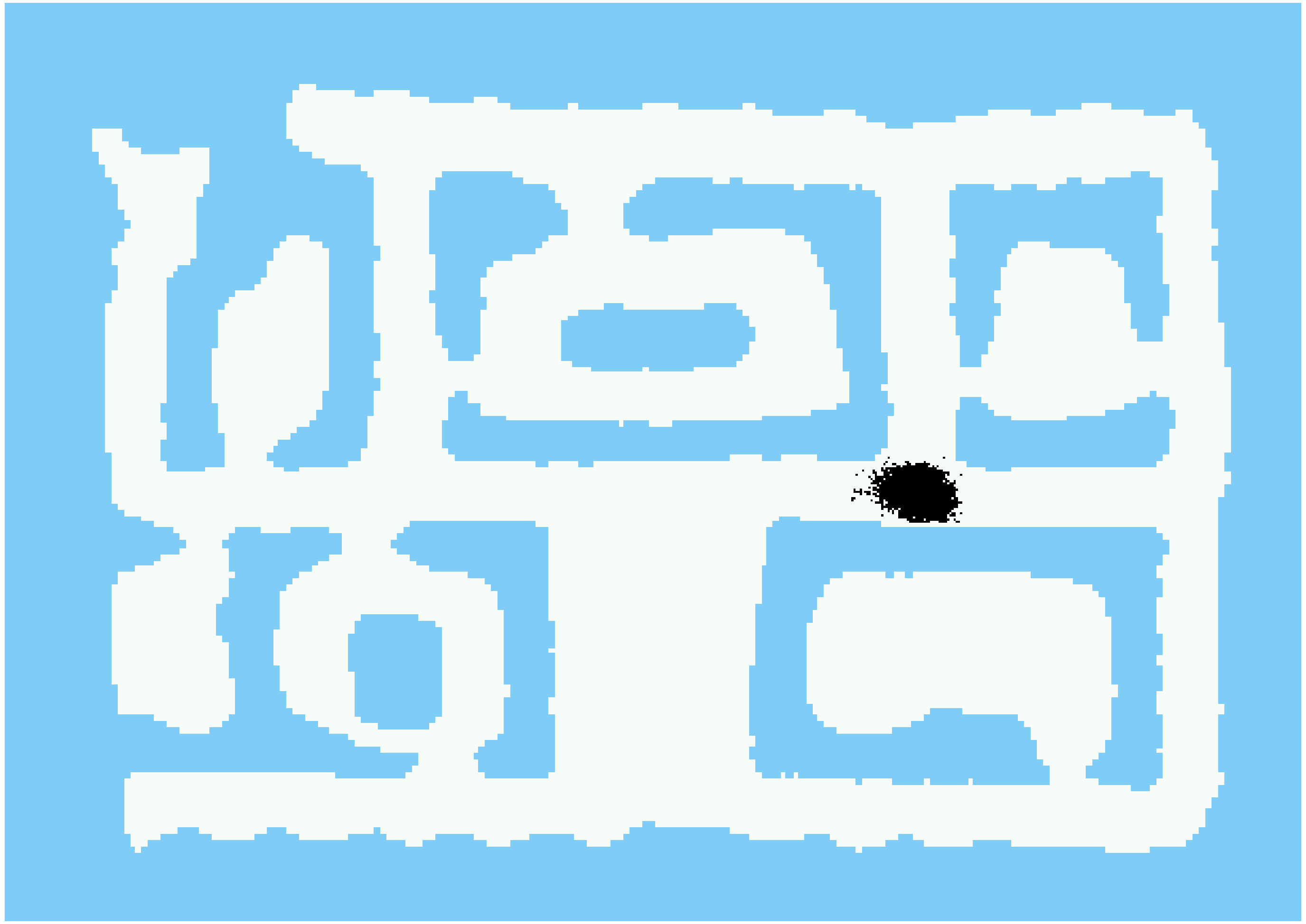
Initial Distribution



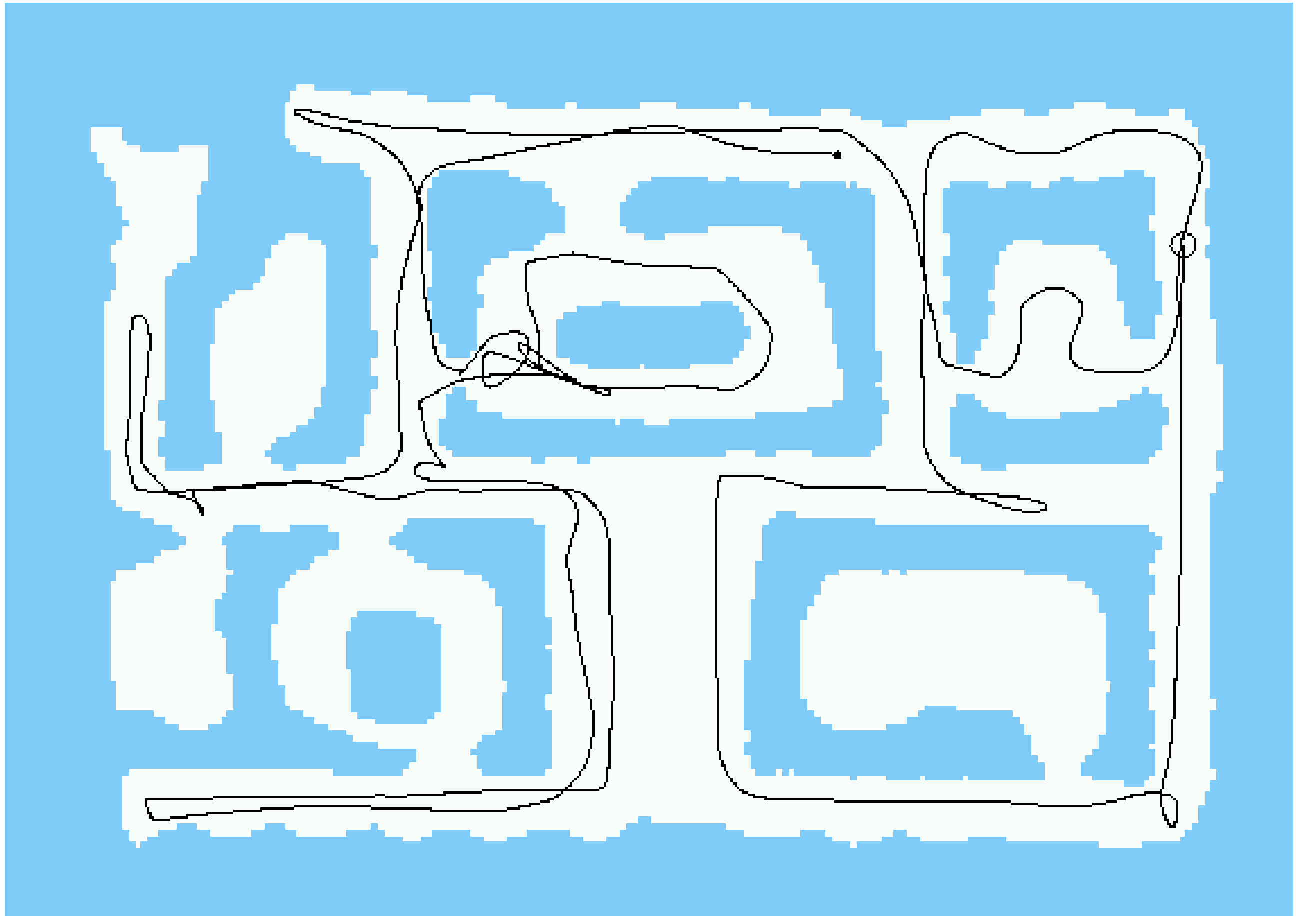
After Incorporating Ten Ultrasound Scans



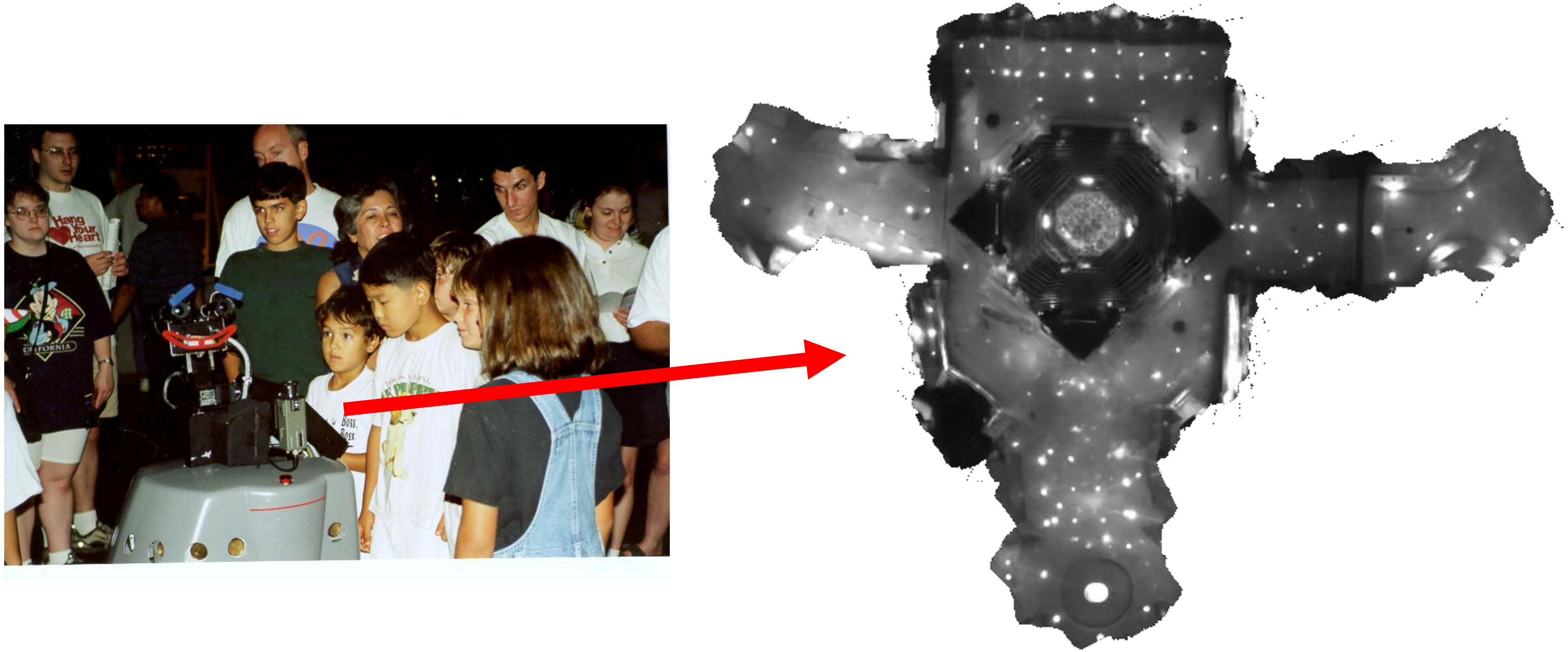
After Incorporating 65 Ultrasound Scans



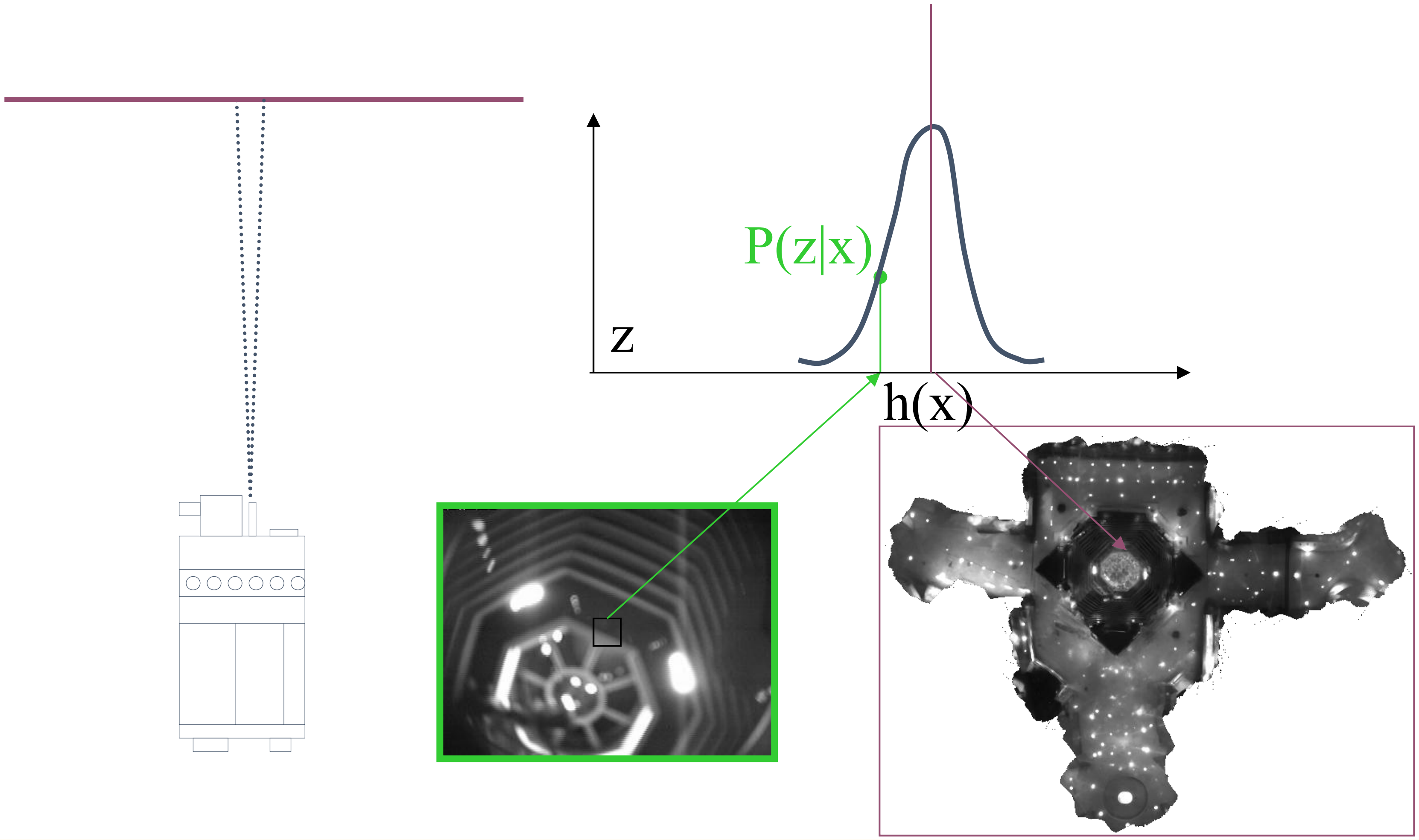
Estimated Path



Using Ceiling Maps for Localization

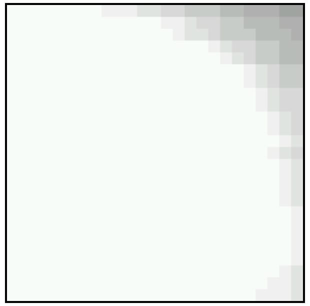


Vision-based Localization

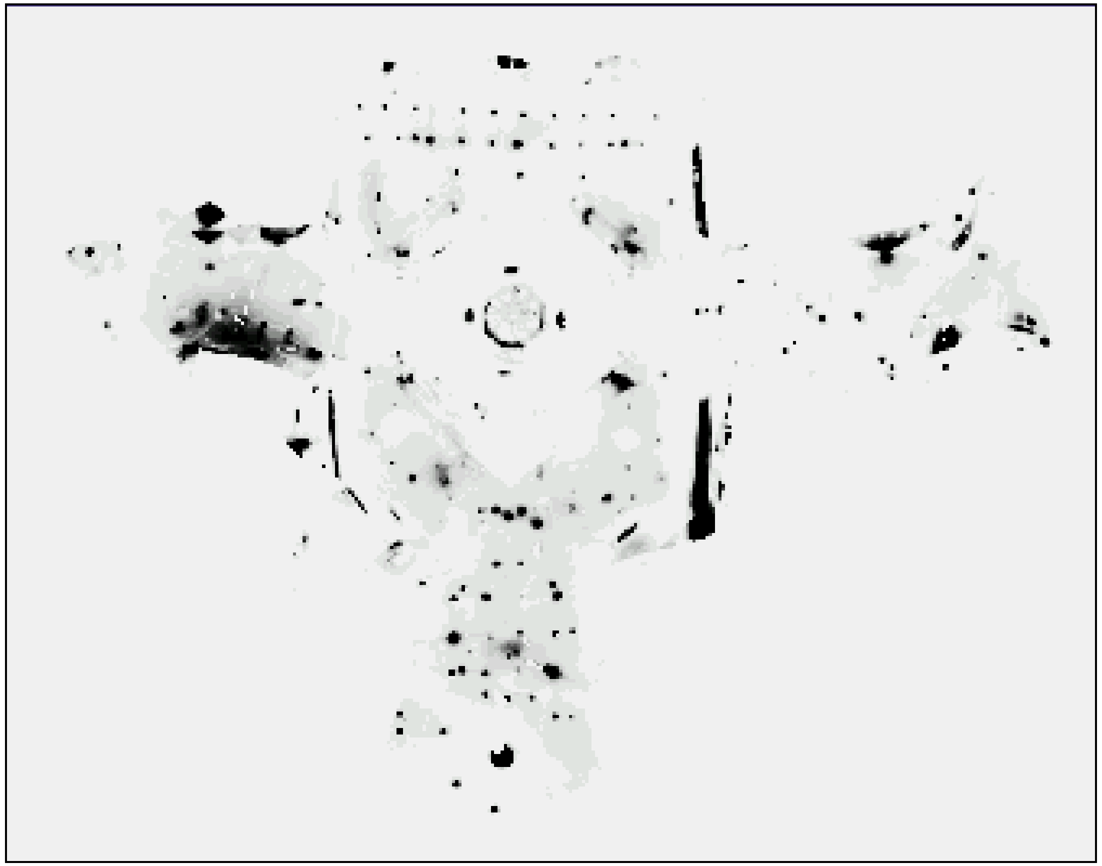


Under a Light:

Measurement z:

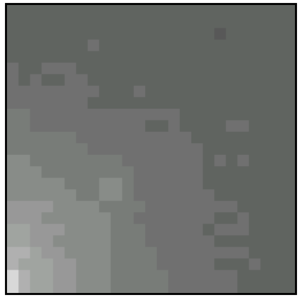


***P(z|x)*:**

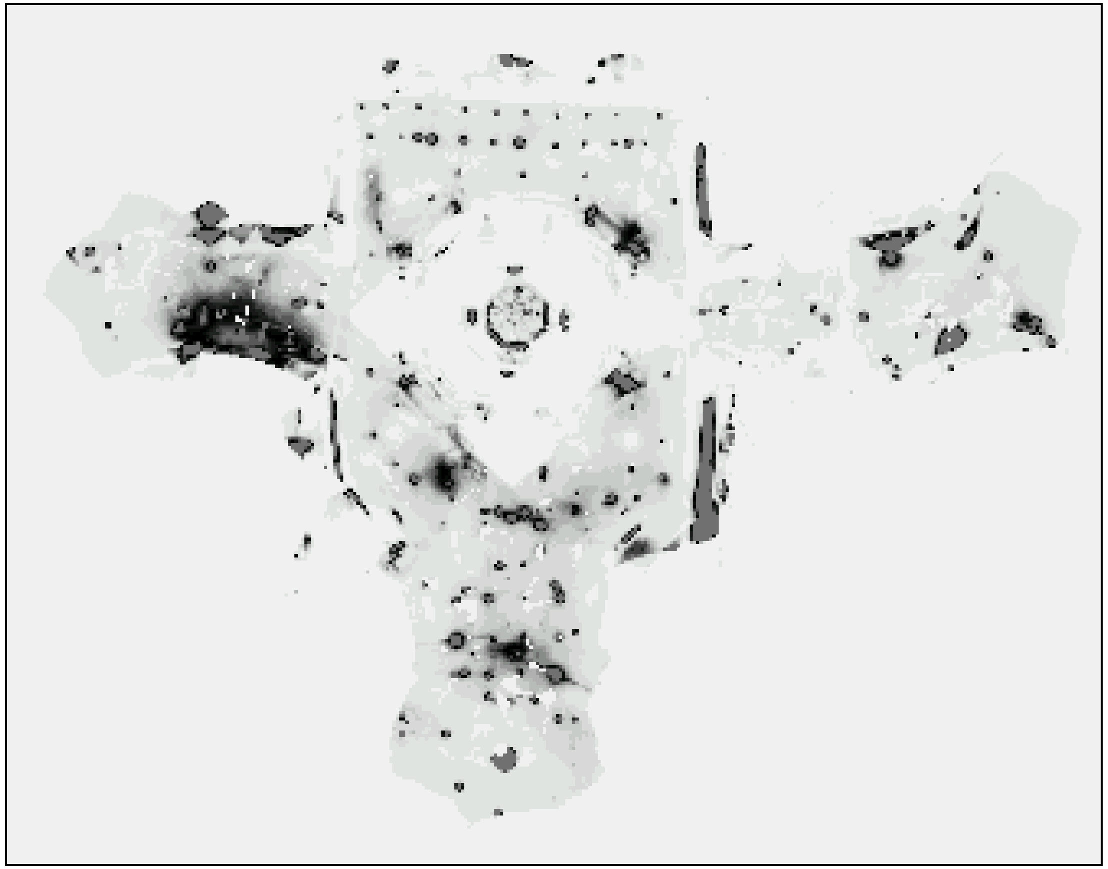


Next to a Light

Measurement z:



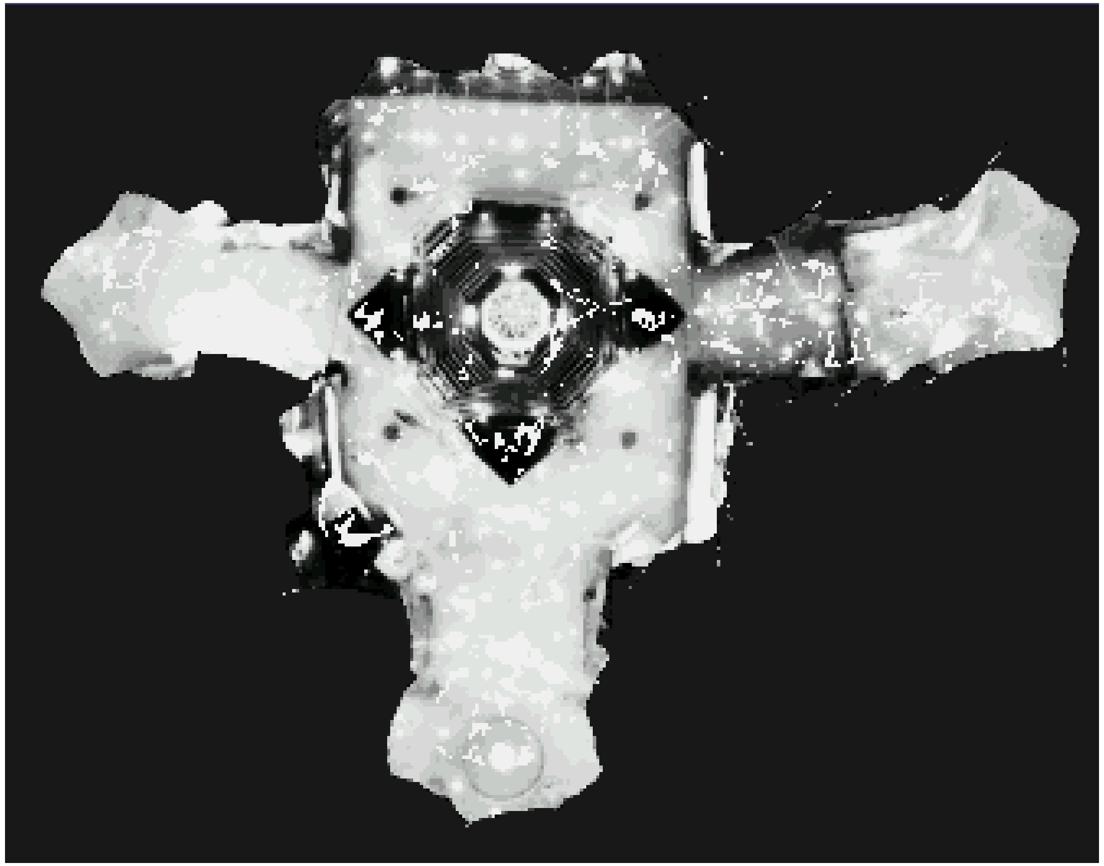
***P(z|x)*:**



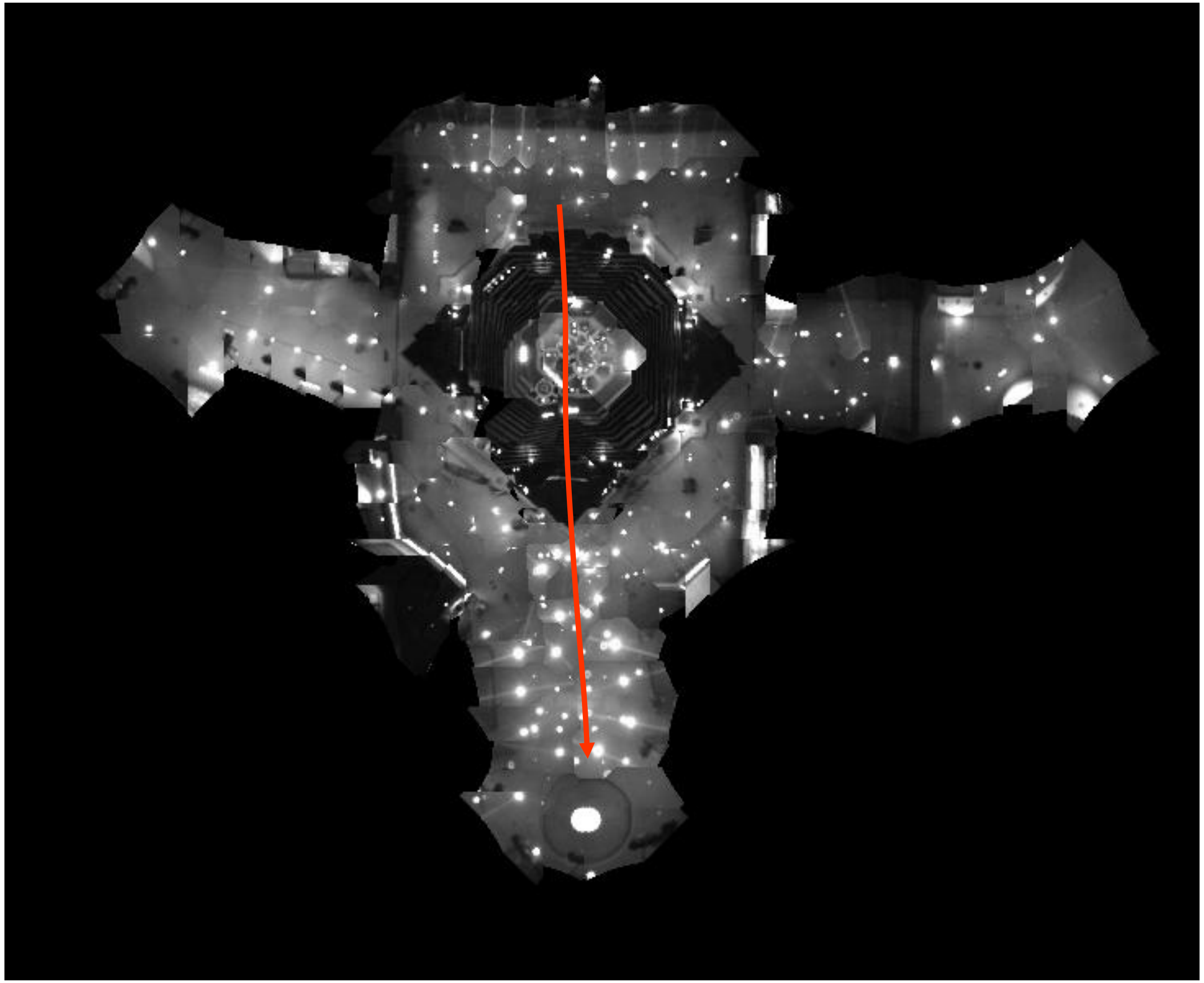
Elsewhere

Measurement z:

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



Global Localization Using Vision



Summary & Limitations

Particle filters implement Bayesian filter representing the posterior by a set of weighted samples

- For localization, the particles are propagated according to the motion model
- Then weighted according to measurement likelihood.
- In a re-sampling step, new particles are drawn with a probability proportional weight

PF can track the pose of a mobile robot and to and globally localize the robot.

- Can we deal with kidnapped robot problem?
  - Randomly insert samples with small probability

Brief introduction to Kalman filter

How to estimate state if you have

- linear model of system and measurement
- hard limit on computation
- Need stronger convergence guarantees

~~Nonparametric method~~

~~Heavy computation~~

~~Can handle nonlinear models~~

~~Does not rely on analytical expression for distributions~~

~~Convergence guarantees only under assumptions on #partiles→ infty~~

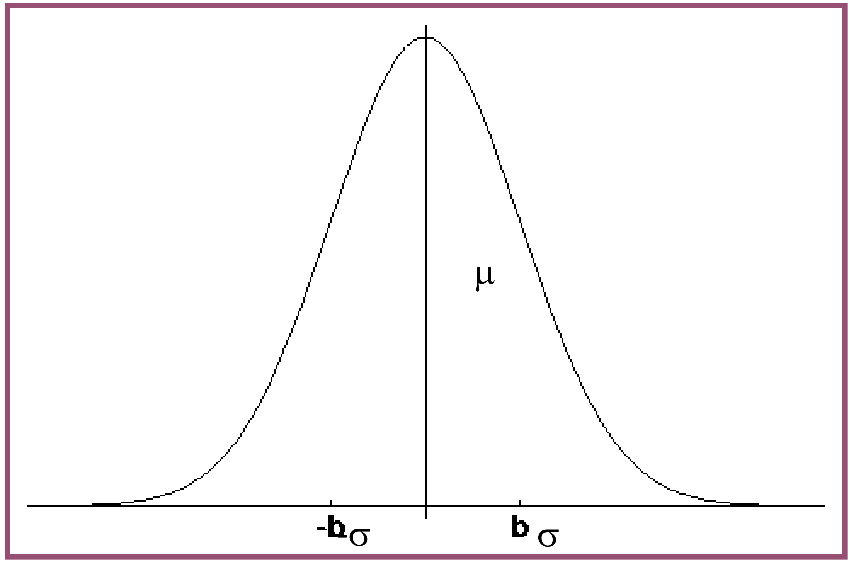
Kalman Filter: State estimation algorithm for linear systems with Gaussian uncertainty

Gaussians

$$p(x) \sim N(\mu, \sigma^2):$$

$$p(x) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}$$

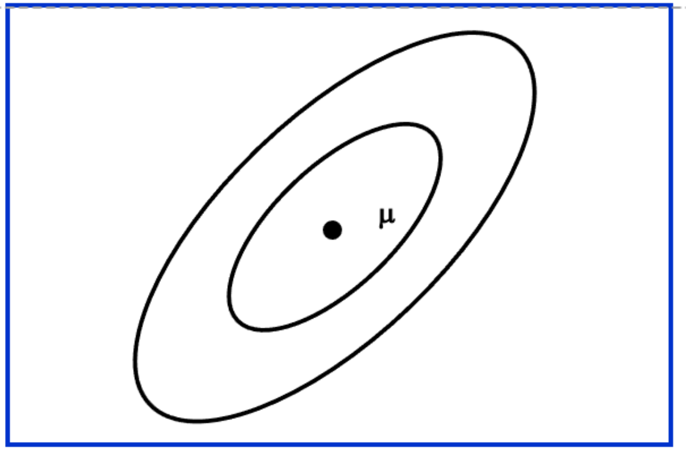
Univariate



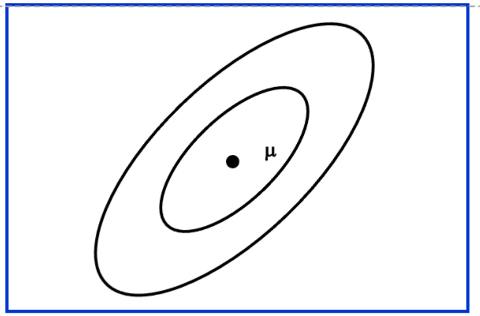
$$p(\mathbf{x}) \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}):$$

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}\left|\boldsymbol{\Sigma}\right|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^t \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$$

Multivariate

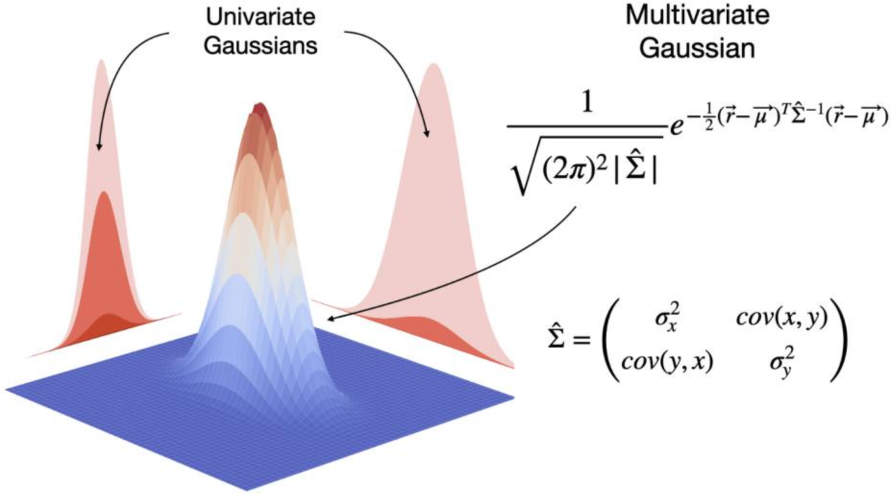


Multivariate Gaussians



$p(\mathbf{x}) \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}):$

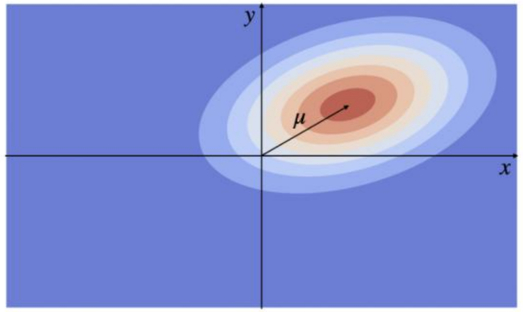
$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2} \left|\boldsymbol{\Sigma}\right|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^t \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})}$$



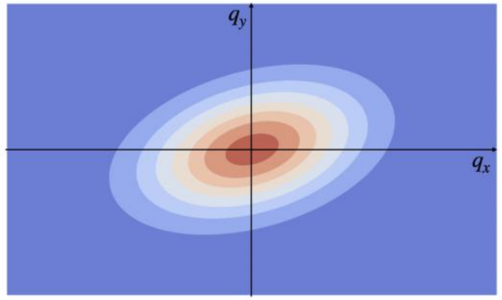
Every single variable x_i in x has a normal distribution $N(\mu_i, \sigma_i)$

If the variables are uncorrelated then the covariance matrix Σ v
be a diagonal matrix with the diagonal terms $\{\sigma_i^2\}$

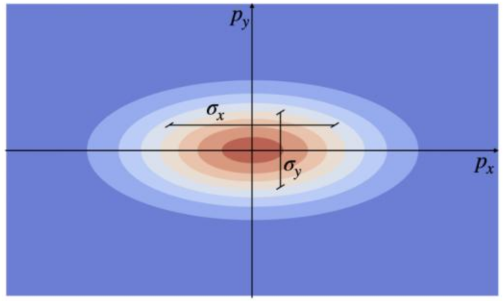
Interactive demo:
https://colab.research.google.com/drive/1Z6v83JRmJWKAVuniS48GKuro18Ky1qtt?usp=sharing



Random bivariate Gaussian distribution



Bivariate Gaussian distribution centered in zero



Bivariate Gaussian distribution diagonalised

Properties of Gaussians

Linear transformations of Gaussians are Gaussians
Gaussian are closed under linear transformations

$$\left.\begin{array}{l} X \sim N(\mu, \sigma^2) \\ Y = aX + b \end{array}\right\} \Rightarrow Y \sim N(a\mu + b, a^2\sigma^2)$$

Products of Gaussian densities is (proportionally) a Gaussian

$$\left.\begin{array}{l} X_1 \sim N(\mu_1, \sigma_1^2) \\ X_2 \sim N(\mu_2, \sigma_2^2) \end{array}\right\} \Rightarrow p(X_1) \cdot p(X_2) \sim N\left(\frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2}\mu_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}\mu_2, \quad \frac{1}{\sigma_1^{-2} + \sigma_2^{-2}}\right)$$

Multivariate Gaussians

$$\left.\begin{array}{l} X \sim N(\mu, \Sigma) \\ Y = AX + B \end{array}\right\} \Rightarrow Y \sim N(A\mu + B, A\Sigma A^T)$$

$$\left.\begin{array}{l} X_1 \sim N(\mu_1, \Sigma_1) \\ X_2 \sim N(\mu_2, \Sigma_2) \end{array}\right\} \Rightarrow p(X_1) \cdot p(X_2) \sim N\left(\frac{\Sigma_2}{\Sigma_1 + \Sigma_2}\mu_1 + \frac{\Sigma_1}{\Sigma_1 + \Sigma_2}\mu_2, \frac{1}{\Sigma_1^{-1} + \Sigma_2^{-1}}\right)$$

We stay in the “Gaussian world” as long as we start with Gaussians and perform only linear transformations.

Discrete Kalman Filter

The Kalman filter estimates state of a Discrete Linear System with Gaussian noise

Note that we no longer have discrete states or measurements! No grids, particles, etc.

$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t$$
$$z_t = C_t x_t + \delta_t$$

x_t: State vector
$\mathrm{u_t}$: Input vector
z_t: Output vector
$\varepsilon_t \sim N(0, \mathrm{Q_t})$: Process noise with covariance $\mathrm{Q_t}$
$\delta_t \sim N(0, R_t)$: Measurement noise with covariance $\mathrm{R_t}$
$p(x_t | x_{t-1}, u_t) = N(A_t x_{t-1} + B_t u_t, \mathrm{Q_t})$
$p(z_t | x_t) = N(C_t x_t, R_t)$

Kalman Filter Algorithm

Kalman_Filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):

Prediction: get $\bar{\mu}_t$ and $\bar{\Sigma}_t$ (linear motion)

1. $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
2. $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^{\top} + Q_t$

Correction: correct $\bar{\mu}_t$ and $\bar{\Sigma}_t$ (linear meas.)

1. $K_t = \bar{\Sigma}_t C_t^{\top} (C_t \bar{\Sigma}_t C_t^{\top} + R_t)^{-1}$
2. $\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$
3. $\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$

Return μ_t, Σ_t

Given $\mathrm{bel}(x_{t-1}) \sim N(\mu_{t-1}, \Sigma_{t-1})$
Apply motion model to find $\bar{x}_t$:
Linear transformation of Gaussian $\mathrm{bel}(x_{t-1})$
where $x_t = A_t x_{t-1} + B_t u_t + \epsilon_t;\ \varepsilon_t \sim N(0, Q_t)$
=> $\bar{x}_t \sim N(\bar{\mu}_t, \bar{\Sigma}_t)$

Given $\bar{x}_t \sim N(\bar{\mu}_t, \bar{\Sigma}_t)$
Apply measurement model to find $\mathrm{bel}(x_t)$:
Product of Gaussians $\bar{x}_t$ and $\mathrm{p}(z_t |\ x_t)$
Where $p(z_t | x_t)$ is a Gaussian (variable is x_t)
=> $\mathrm{bel}(x_t) \sim N(\mu_t, \Sigma_t)$

Kalman Filter Algorithm

Kalman_Filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):

Prediction:

1. $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
2. $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^\top + Q_t$

Correction:

1. $K_t = \bar{\Sigma}_t C_t^\top (C_t \bar{\Sigma}_t C_t^\top + R_t)^{-1}$
2. $\mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t)$
3. $\Sigma_t = (I - K_t C_t)\bar{\Sigma}_t$

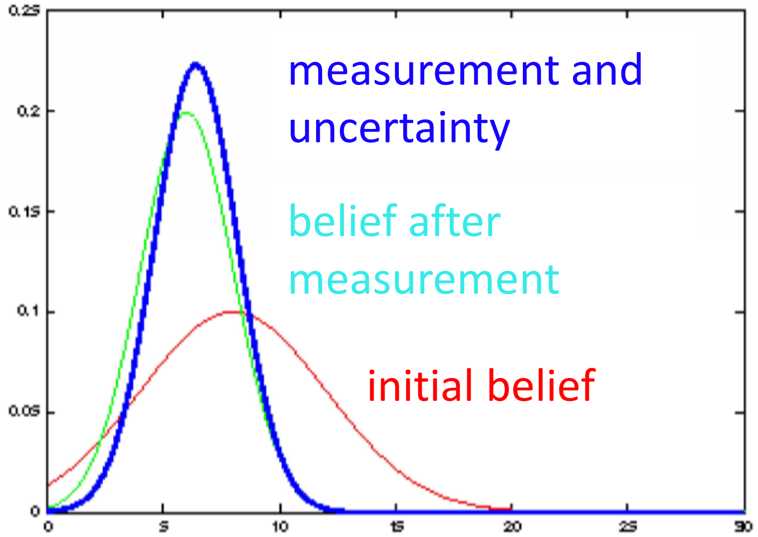
Return μ_t,Σ_t

Kalman Filter represents the belief $bel(x_t)$ by mean μ_t and covariance Σ_t

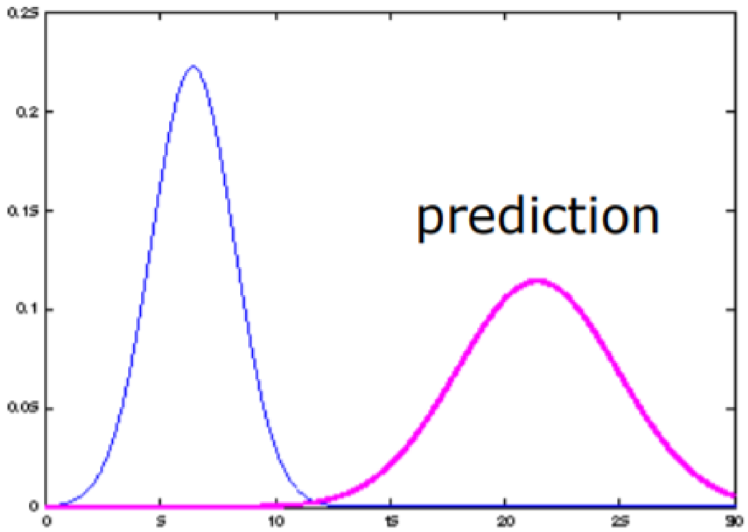
Correction computes the ***Kalman gain*** K_t to weight the impact of new measurements against the predicted value

Higher measurement variance C_t => lower gain, less helpful

innovation = $z_t - C_t \bar{\mu}_t$ reflects how large the deviation is from prediction to actual observations



Apply control action

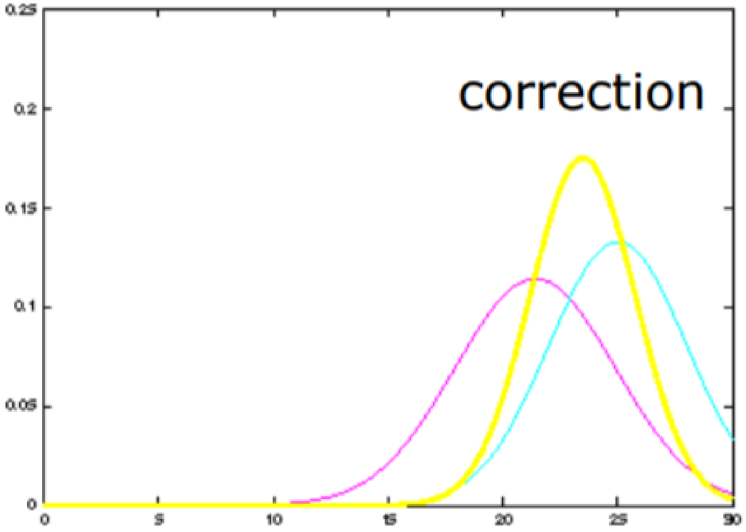


Correction:

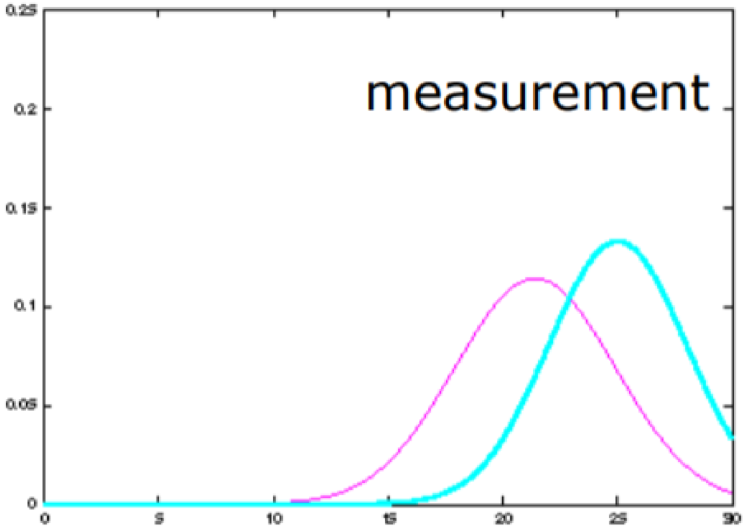
1. $K_t = \bar{\Sigma}_t C_t^{\top}(C_t \bar{\Sigma}_t C_t^{\top} + R_t)^{-1}$
2. $\mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t)$
3. $\Sigma_t = (I - K_t C_t)\bar{\Sigma}_t$

Prediction:

1. $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
2. $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^{\top} + Q_t$



Get sensor measurement



Kalman Filter Example

Demo: https://colab.research.google.com/drive/1qcINZgx8ebwWtRQROh3z8cpvtmuE4Dt0?usp=sharing