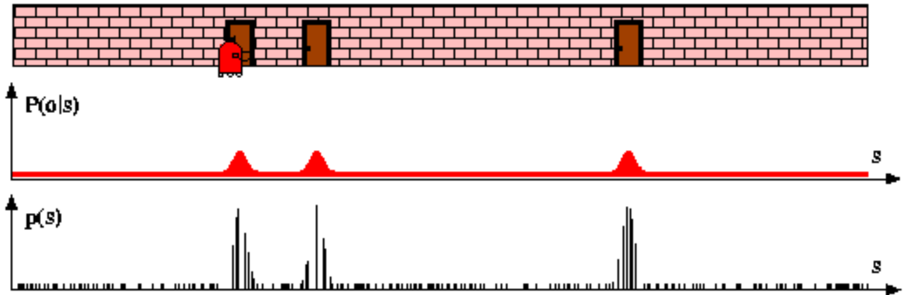
# CS 354
# Autonomous Robotics

# Particle Filters




Instructors: Dr. Kevin Molloy and
Dr. Nathan Sprague

# Objectives

**Localization** Process of determining where a mobile robot is located with respect to its environment.

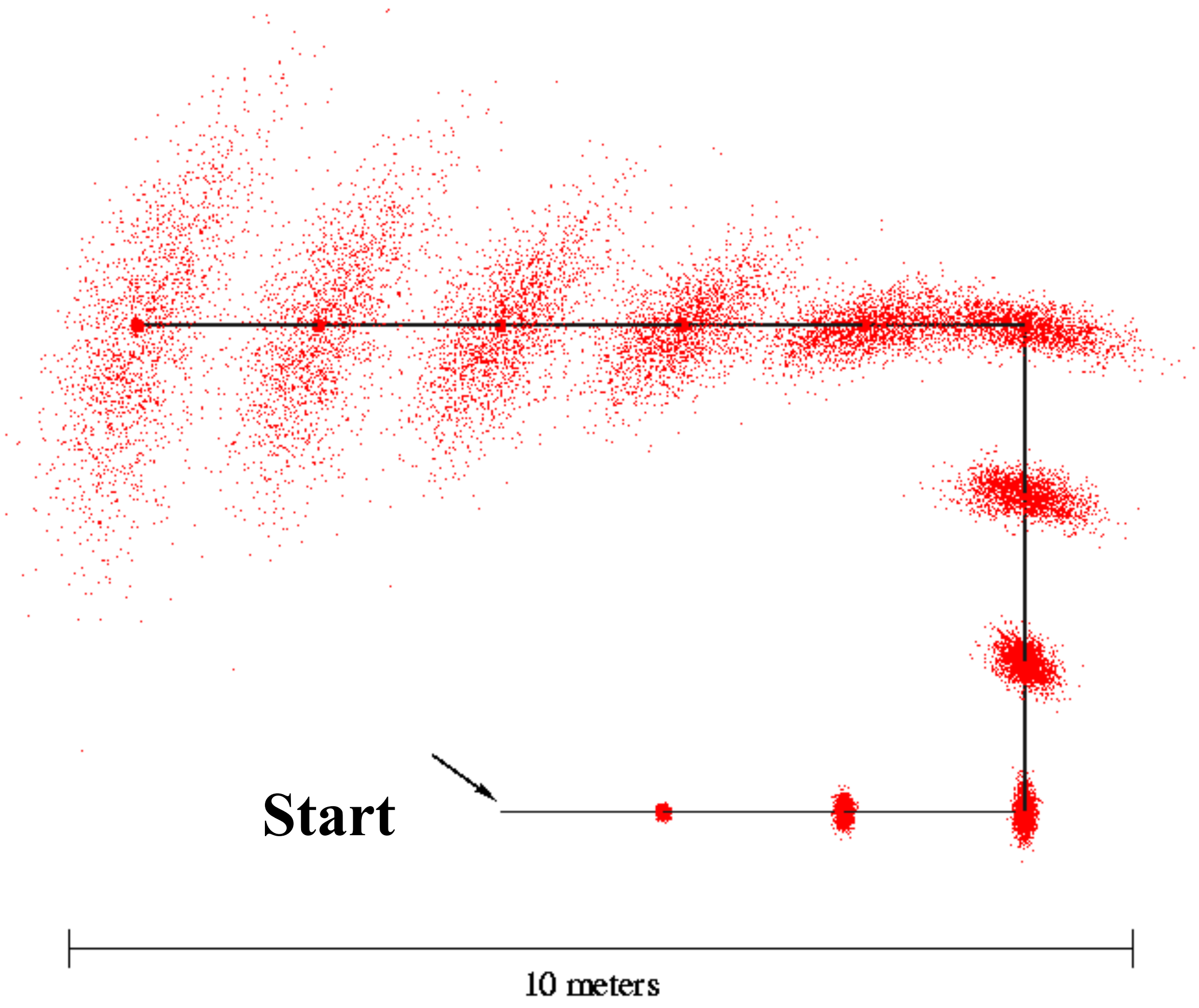
Methods we know so far:

- Grid-based localization and tracking
- Kalman Filters

Today we are going to discuss **particle filters**.

- Represent belief by random samples
- Estimation of non-Gaussian, nonlinear processes
- Monte Carlo filter, Survival of the fittest

# Motion Model Reminder



# Particle Filter Algorithm

Particle Filer ($X_{t-1}$, $u_t$, $z_t$)

Inputs:

  $X_{t-1}$ – The previous particles

  $u_t$ – the control signal

  $z_t$ - the sensor value

Output: $X_t$ – Updated particles

$Xbar_t = []$

$M = size(X_{t-1})$

For m = 0 to M-1 do

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

  $w_t^{[m]} = p(z_t \mid x_t^{[m]}) w_{t-1}^{[m]}$

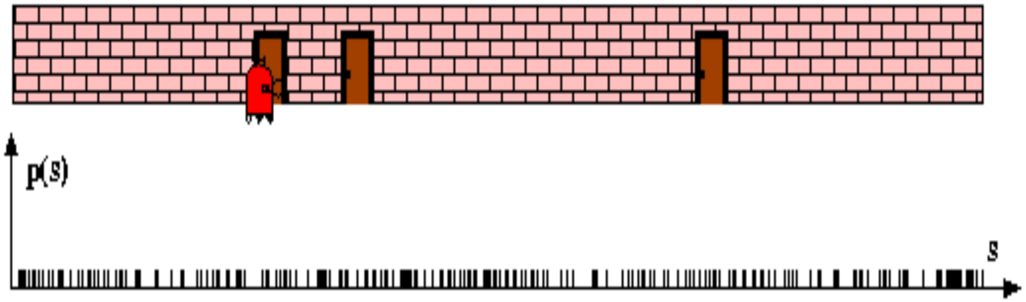
  $Xbar_t = Xbar_t \cup \{<x_t^{[m]}, w_t^{[m]}>\}$

For m = 0 to M -1 do

  Draw i with probability prop $w_t^{[i]}$

  $X_t = X_t \cup \{x_t^{[i]}, 1/M\}$

# Particle Filter Algorithm




Particle Filer ($X_{t-1}$, $u_t$, $z_t$)

Inputs:

- $X_{t-1}$ – The previous particles
- $u_t$ – the control signal
- $z_t$ – the sensor value

Output: $X_t$ – Updated particles

$Xbar_t$ = []

M = size($X_{t-1}$)

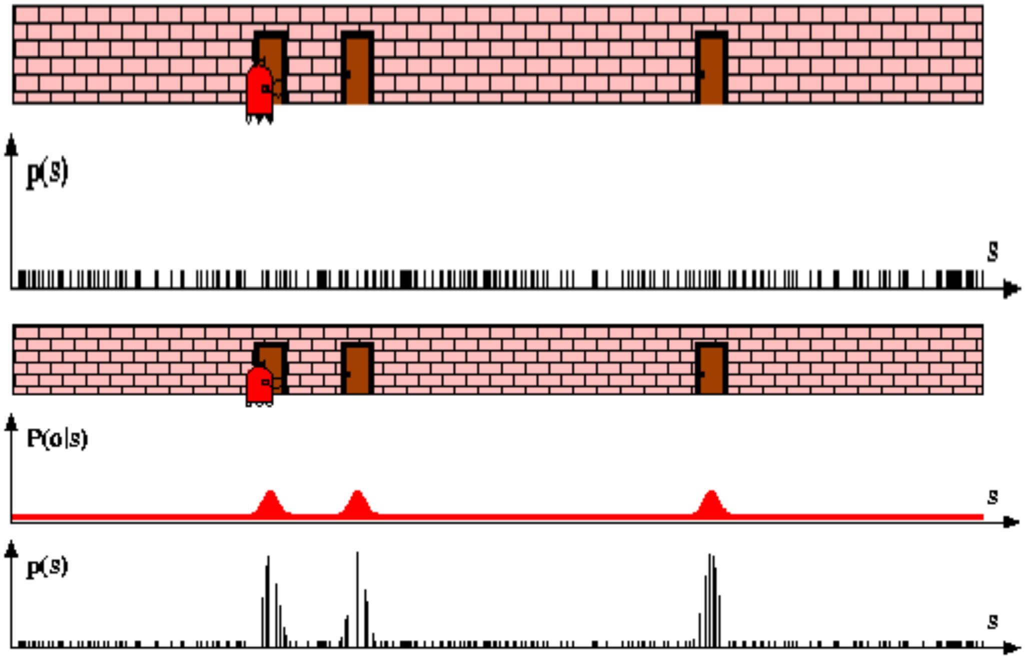
For m = 0 to M-1 do

- sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$
- $w_t^{[m]} = p(z_t \mid x_t^{[m]}) w_{t-1}^{[m]}$
- $Xbar_t = Xbar_t \cup \{<x_t^{[m]}, w_t^{[m]}>\}$

For m = 0 to M -1 do

- Draw i with probability prop $w_t^{[i]}$
- $X_t = X_t \cup \{x_t^{[i]}, 1/M\}$

# Particle Filter Algorithm




Particle Filer ($X_{t-1}$, $u_t$, $z_t$)

Inputs:

$X_{t-1}$ – The previous particles

$u_t$ – the control signal

$z_t$ – the sensor value

Output: $X_t$ – Updated particles

$Xbar_t$ = []

M = size($X_{t-1}$)

For m = 0 to M-1 do

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

$w_t^{[m]} = p(z_t \mid x_t^{[m]}) w_{t-1}^{[m]}$

$Xbar_t = Xbar_t \cup \{<x_t^{[m]}, w_t^{[m]}>\}$

For m = 0 to M -1 do

Draw i with probability prop $w_t^{[i]}$

$X_t = X_t \cup \{x_t^{[i]}, 1/M\}$

# Particle Filter Algorithm

Particle Filer ($X_{t-1}$, $u_t$, $z_t$)

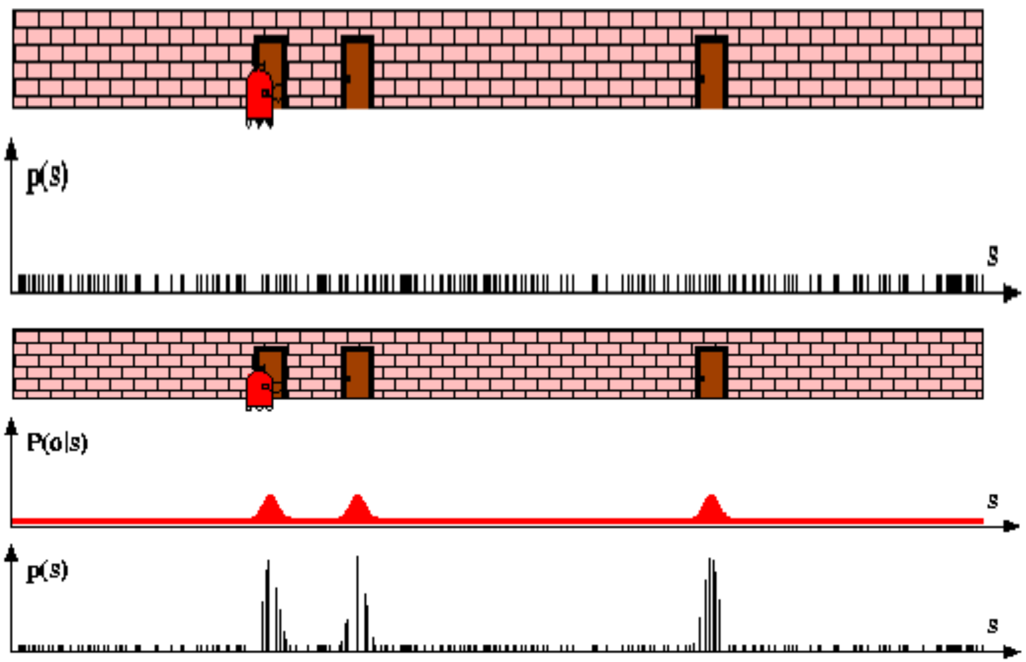
Inputs:

  $X_{t-1}$ – The previous particles

  $u_t$ – the control signal

  $z_t$ - the sensor value

Output: $X_t$ – Updated particles




$Xbar_t$ = []

M = size($X_{t-1}$)

For m = 0 to M-1 do

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

  $w_t^{[m]} = p(z_t \mid x_t^{[m]}) w_{t-1}^{[m]}$

  $Xbar_t$ = $Xbar_t$ U {<$x_t^{[m]}$, $w_t^{[m]}$>}

For m = 0 to M -1 do

  Draw i with probability prop $w_t^{[i]}$

  $X_t$ = $X_t$ U {$x_t^{[i]}$, 1/M}

# Particle Filter Algorithm

Particle Filer ($X_{t-1}$, $u_t$, $z_t$)

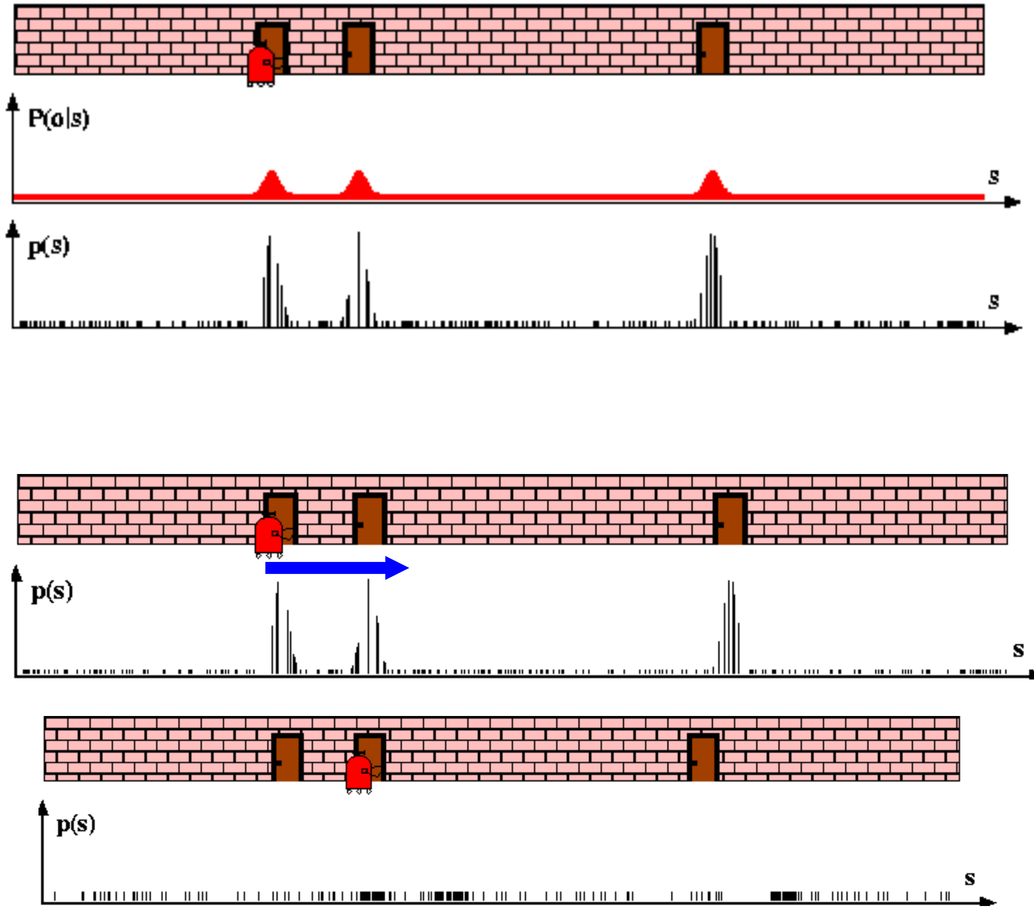
Inputs:

  $X_{t-1}$ – The previous particles

  $u_t$ – the control signal

  $z_t$ – the sensor value

Output: $X_t$ – Updated particles




$Xbar_t = []$

$M = size(X_{t-1})$

For m = 0 to M-1 do

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

  $w_t^{[m]} = p(z_t \mid x_t^{[m]}) w_{t-1}^{[m]}$

  $Xbar_t = Xbar_t \cup \{<x_t^{[m]}, w_t^{[m]}>\}$

For m = 0 to M -1 do

  Draw i with probability prop $w_t^{[i]}$

  $X_t = X_t \cup \{x_t^{[i]}, 1/M\}$

# Particle Filter Algorithm

Particle Filer ($X_{t-1}$, $u_t$, $z_t$)

Inputs:

  $X_{t-1}$ – The previous particles

  $u_t$ – the control signal

  $z_t$ – the sensor value

Output: $X_t$ – Updated particles

$Xbar_t$ = []

M = size($X_{t-1}$)

For m = 0 to M-1 do

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

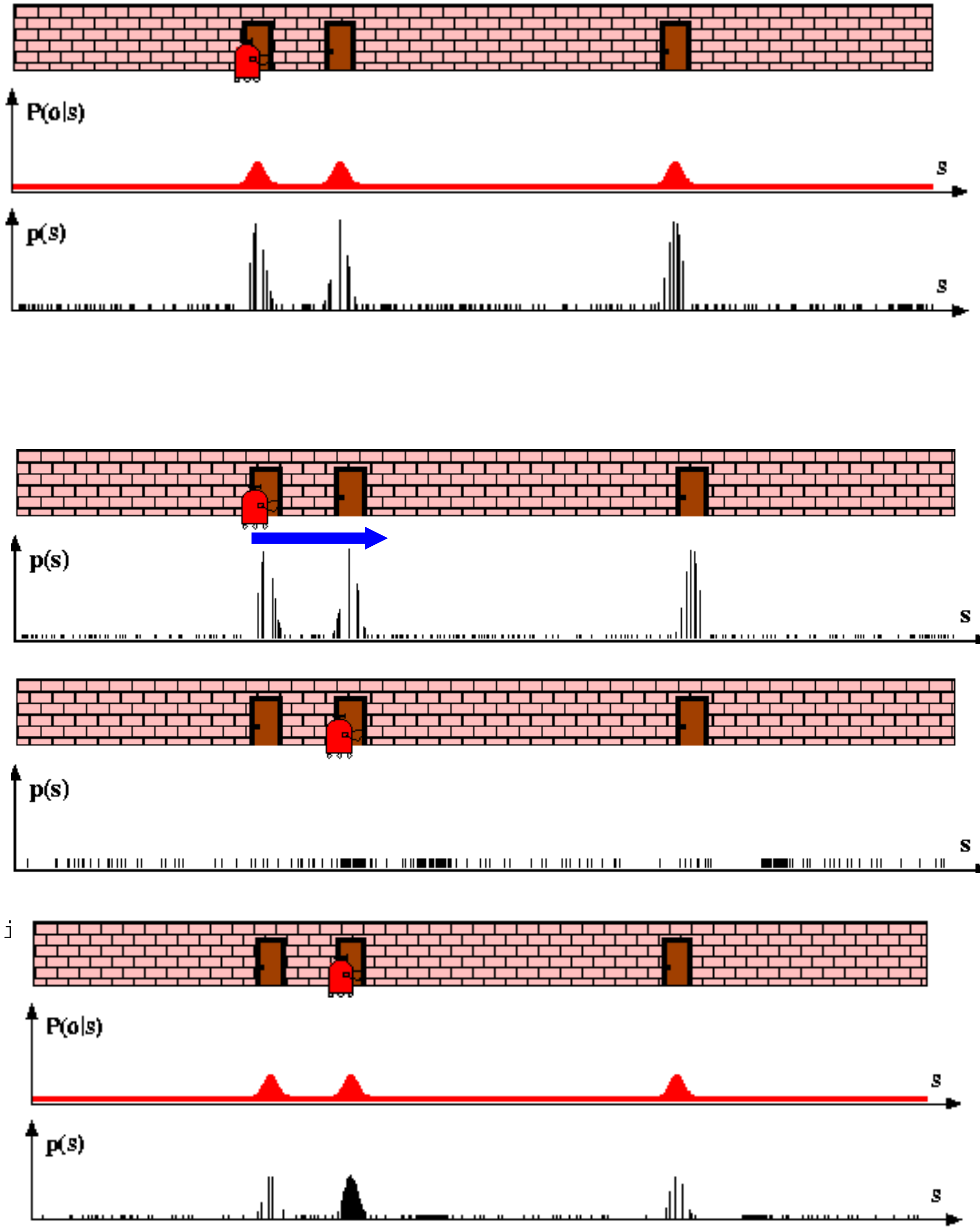
  $w_t^{[m]} = p(z_t \mid x_t^{[m]}) w_{t-1}^{[m]}$

  $Xbar_t = Xbar_t \cup \{<x_t^{[m]}, w_t^{[m]}>\}$

For m = 0 to M -1 do

  Draw i with probability prop $w_t^{[i}$

  $X_t = X_t \cup \{x_t^{[i]}, 1/M\}$



# Particle Filter Algorithm

Particle Filer ($X_{t-1}$, $u_t$, $z_t$)

Inputs:

- $X_{t-1}$ – The previous particles
- $u_t$ – the control signal
- $z_t$ – the sensor value

Output: $X_t$ – Updated particles

$Xbar_t$ = []

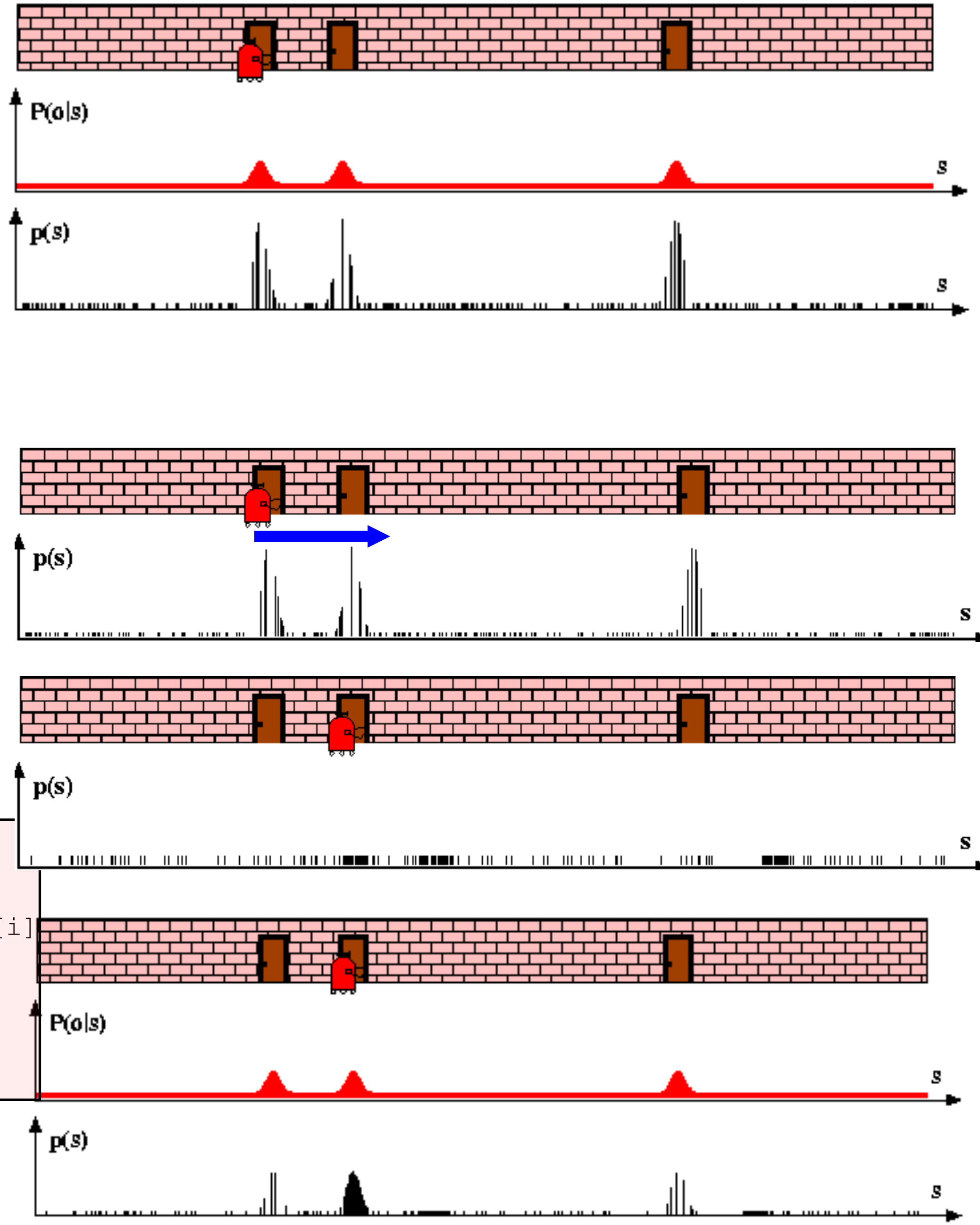
M = size($X_{t-1}$)

For m = 0 to M-1 do

- sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$
- $w_t^{[m]} = p(z_t \mid x_t^{[m]}) w_{t-1}^{[m]}$
- $Xbar_t = Xbar_t \cup \{<x_t^{[m]}, w_t^{[m]}>\}$

For m = 0 to M -1 do

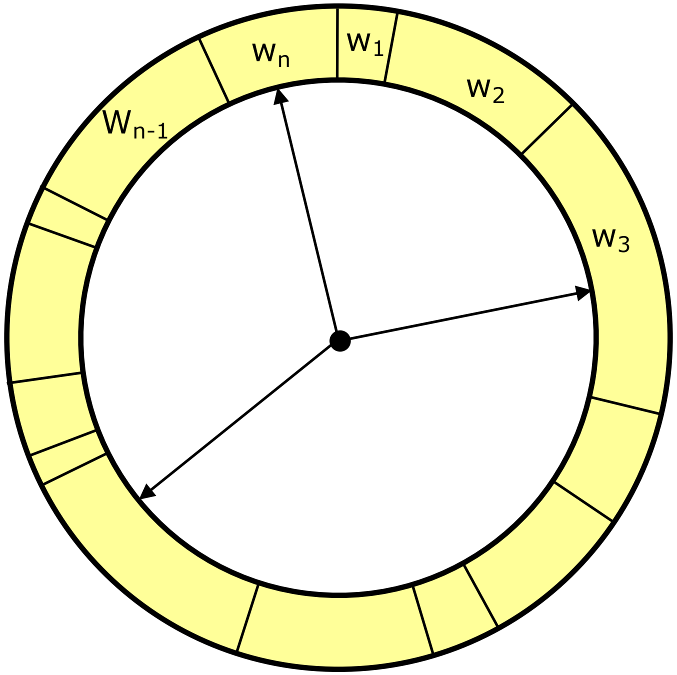
- Draw i with probability prop $w_t^{[i]}$
- $X_t = X_t \cup \{x_t^{[i]}, 1/M\}$

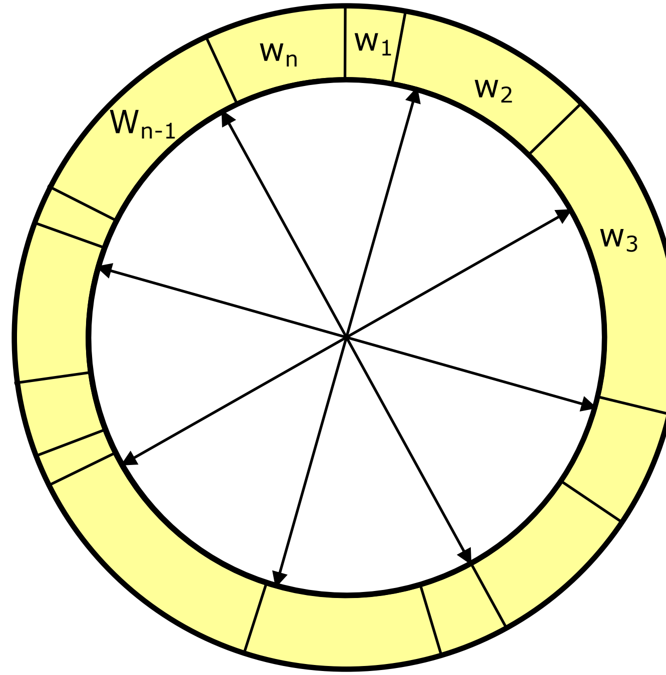


# Resampling

- **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'*.

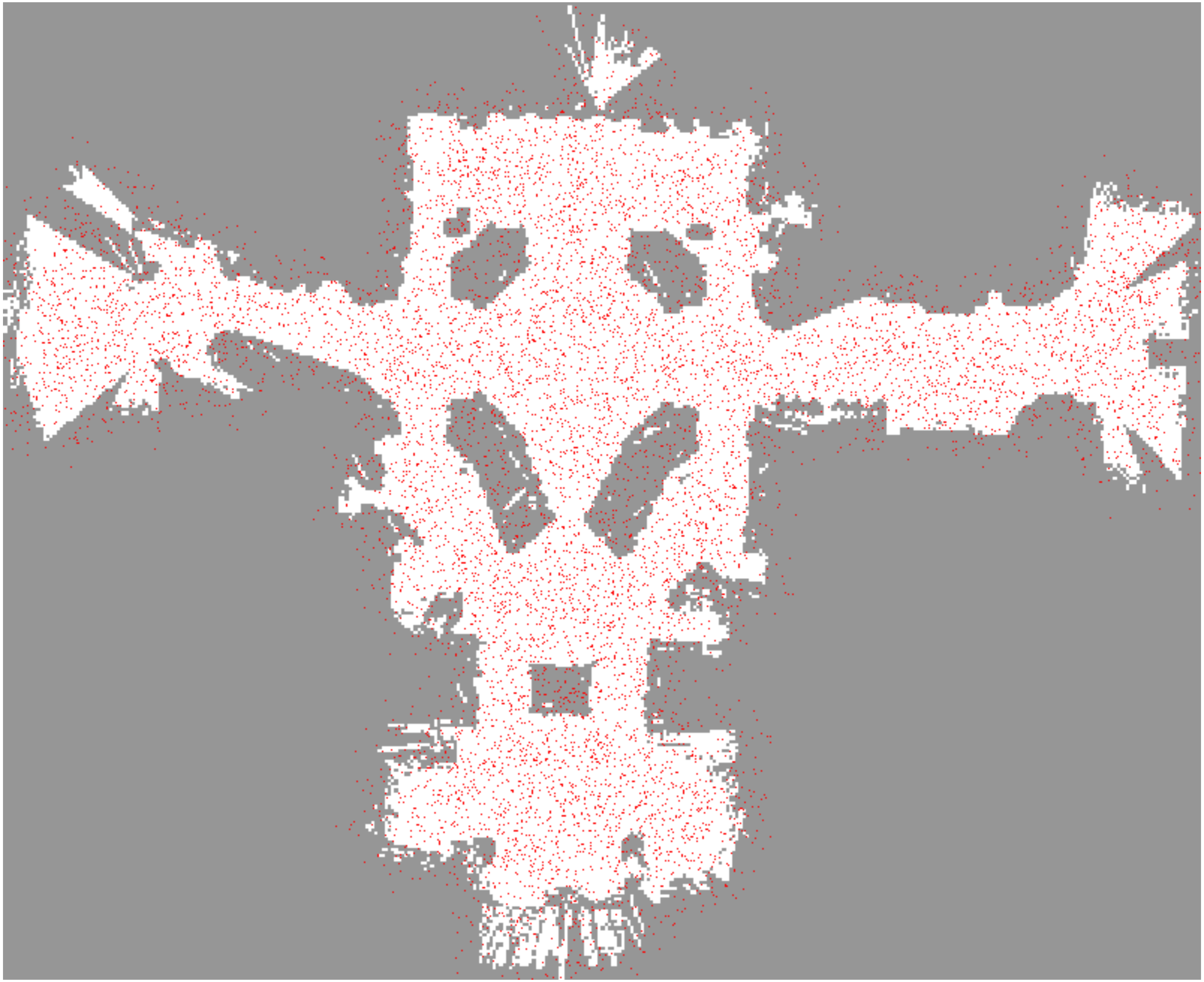
# Resampling

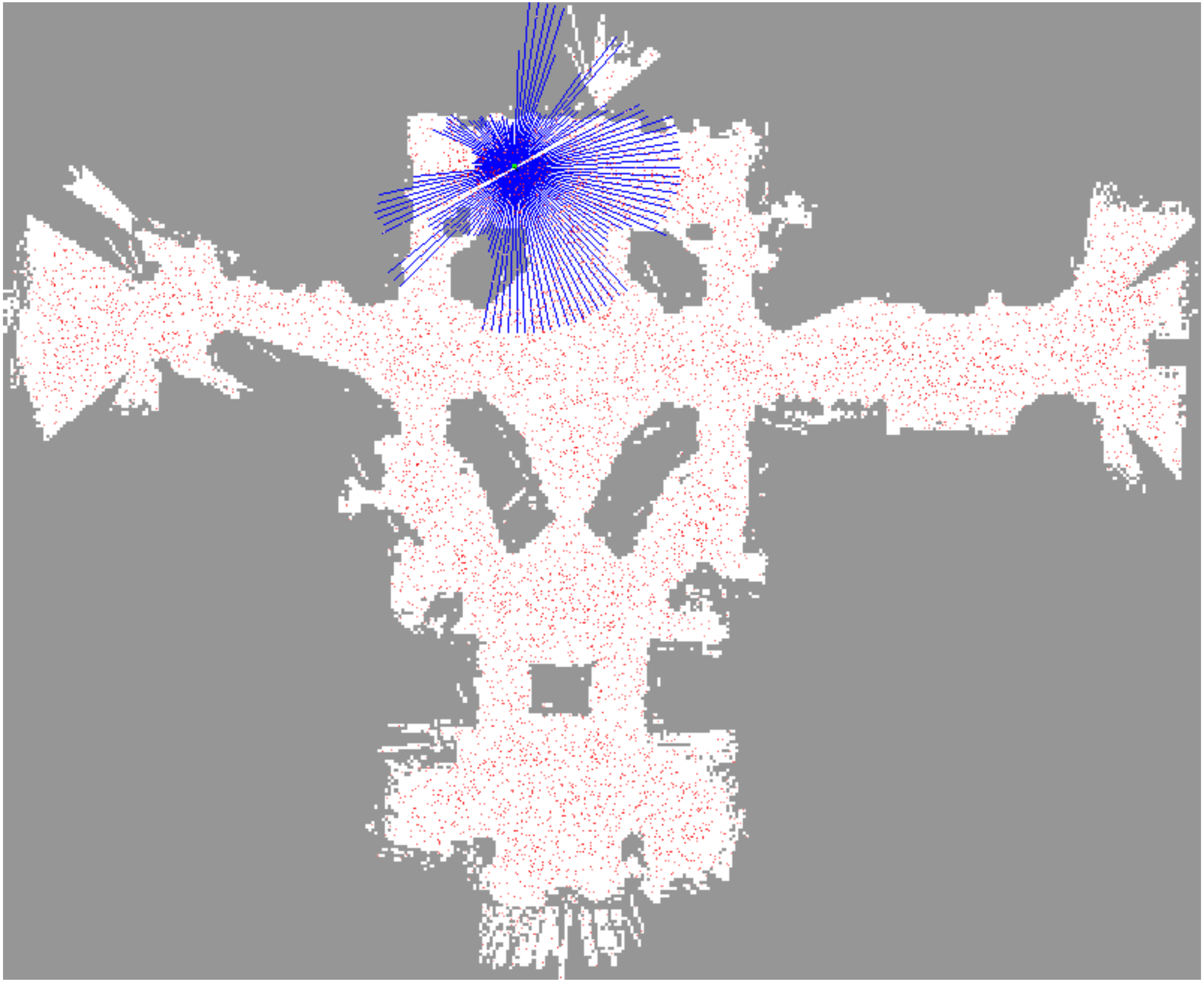


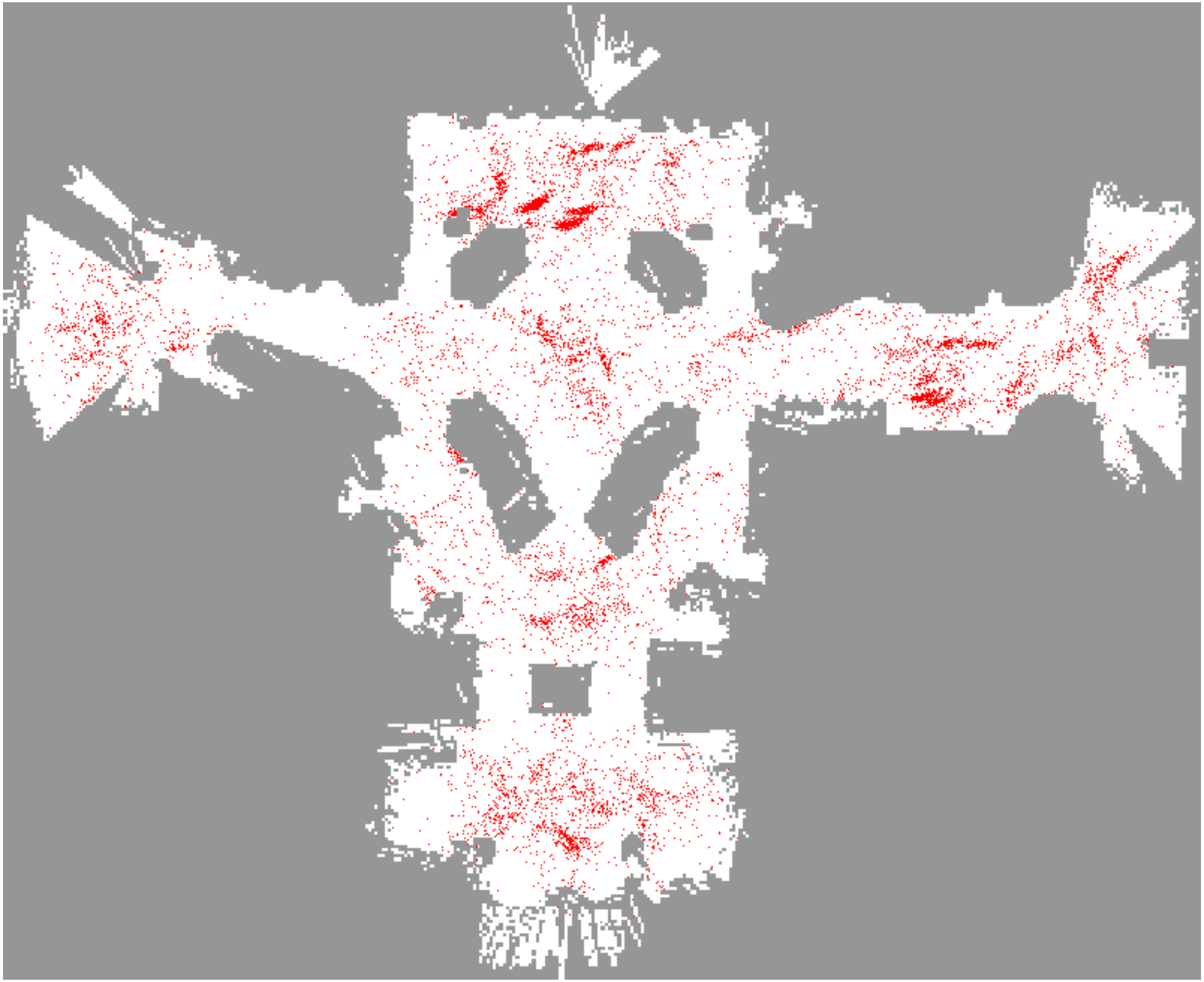



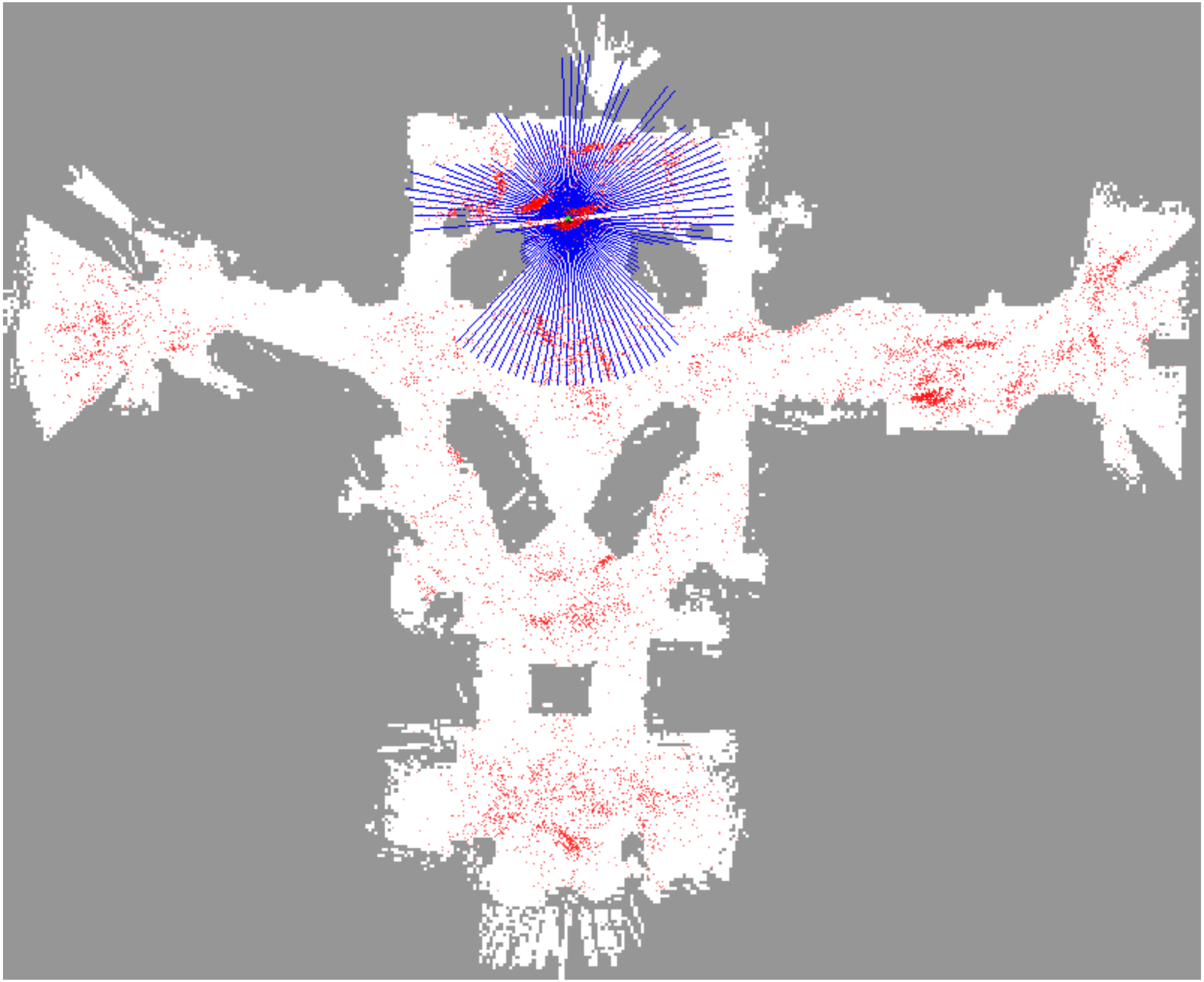

- Roulette wheel
- Binary search, n log n

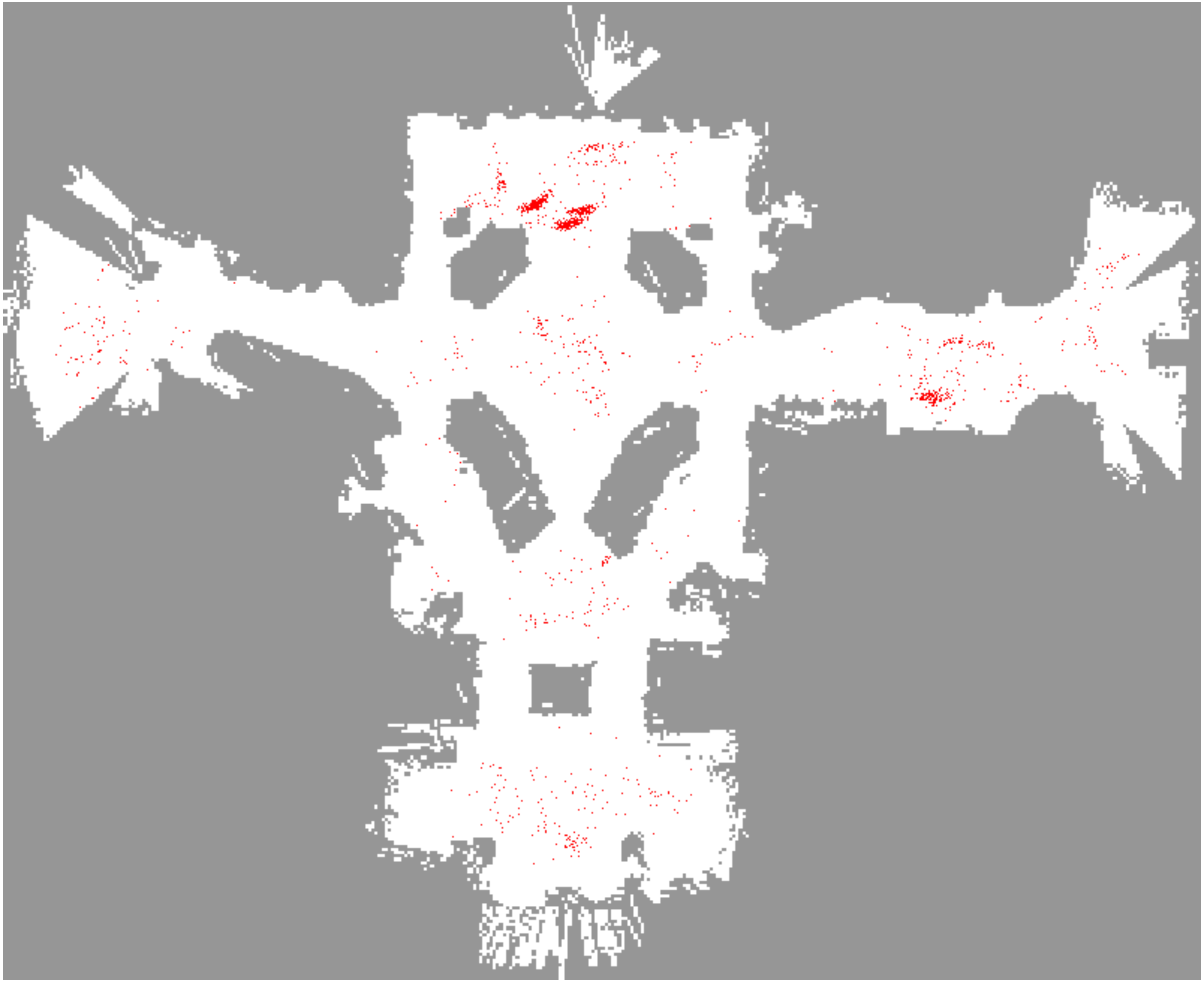
- Stochastic universal sampling
- Systematic resampling
- Linear time complexity
- Easy to implement, low variance

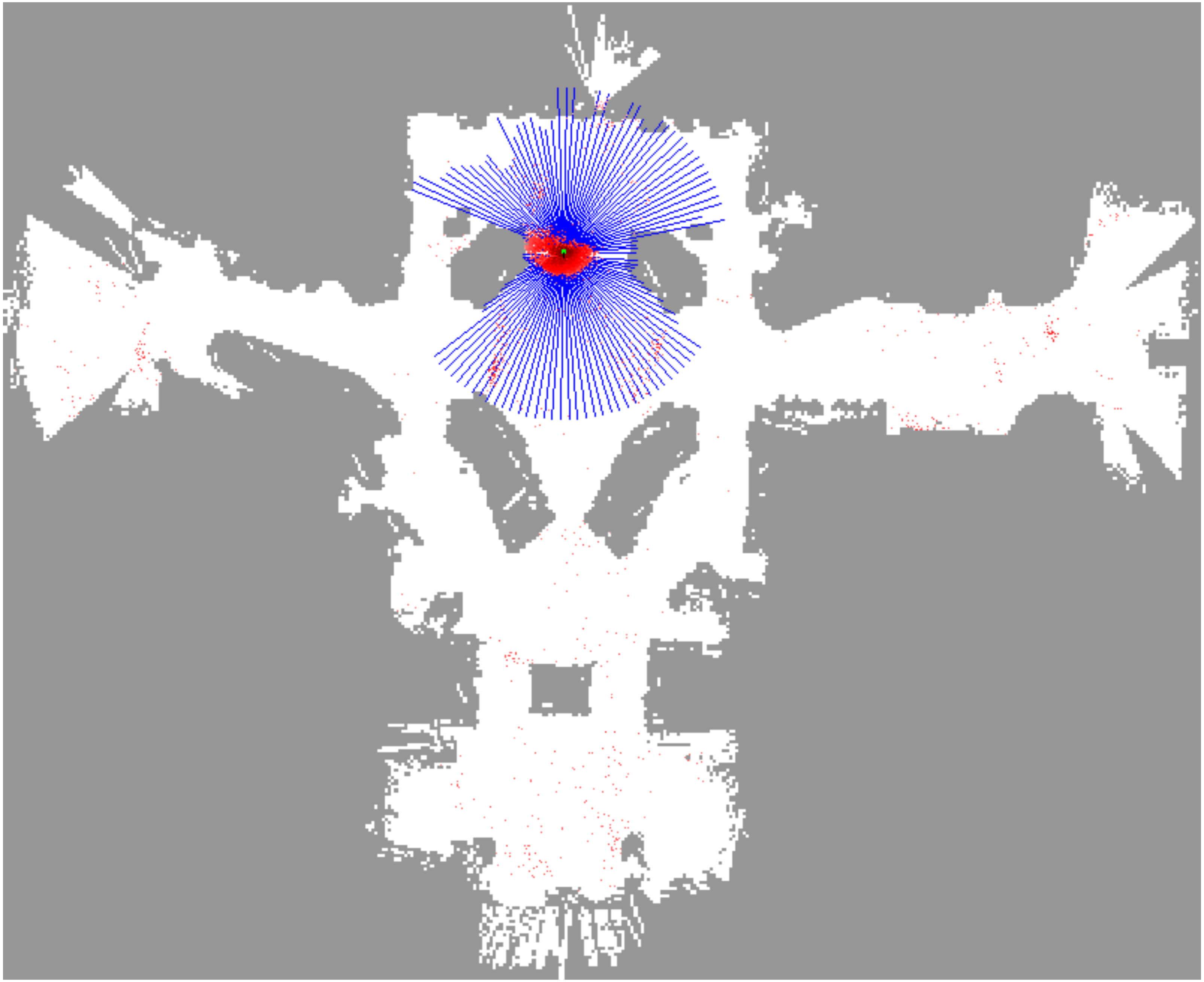


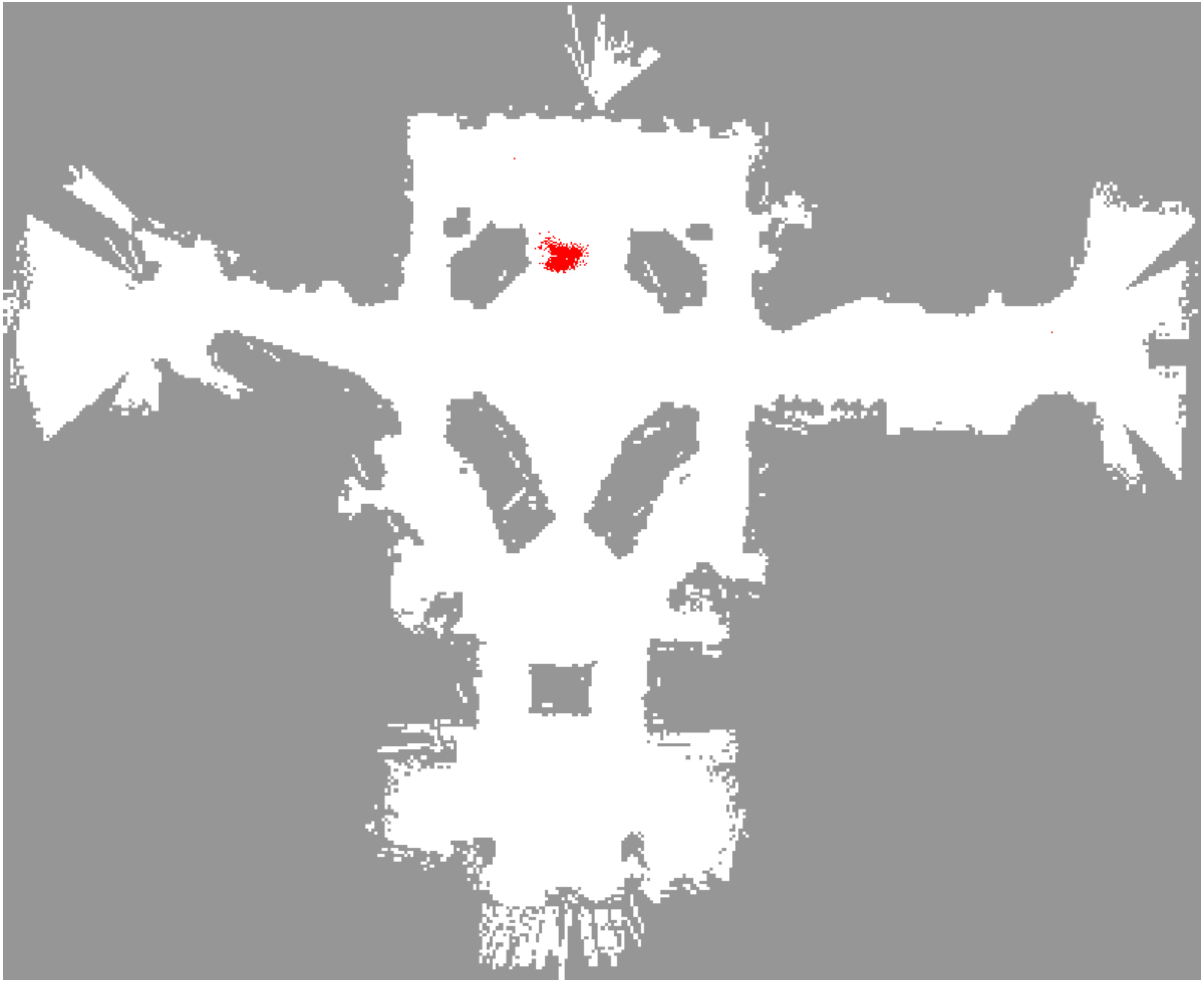


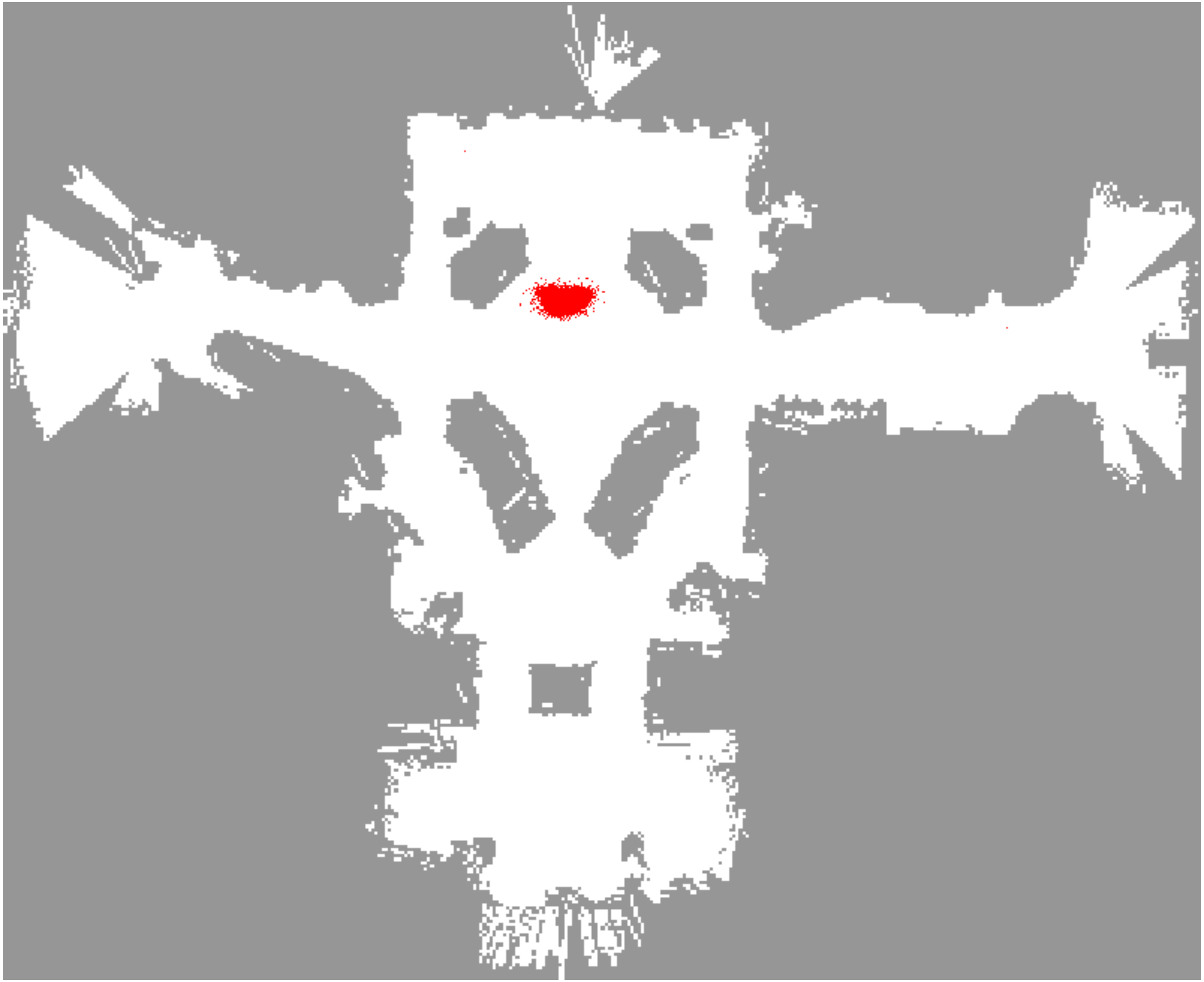


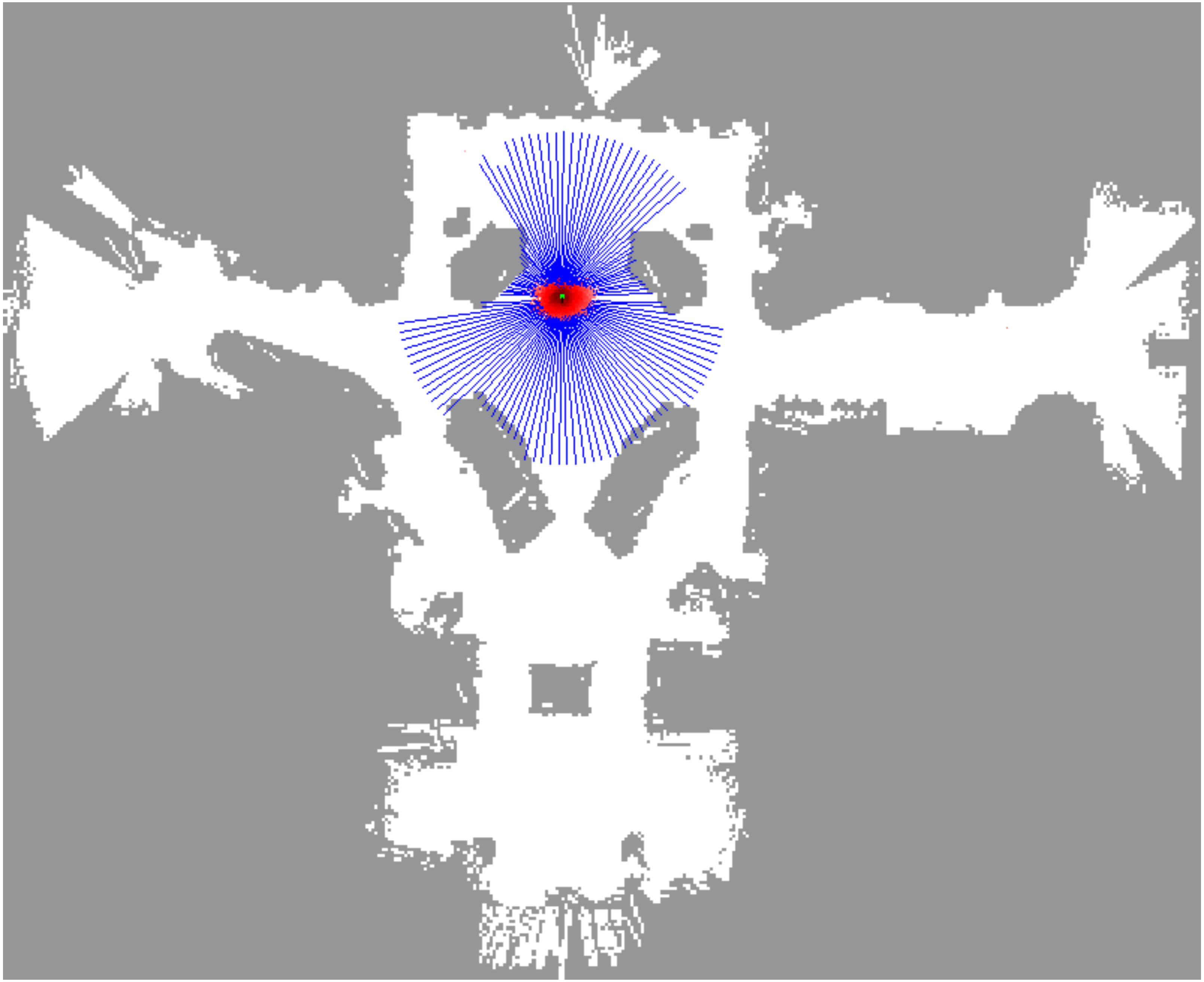


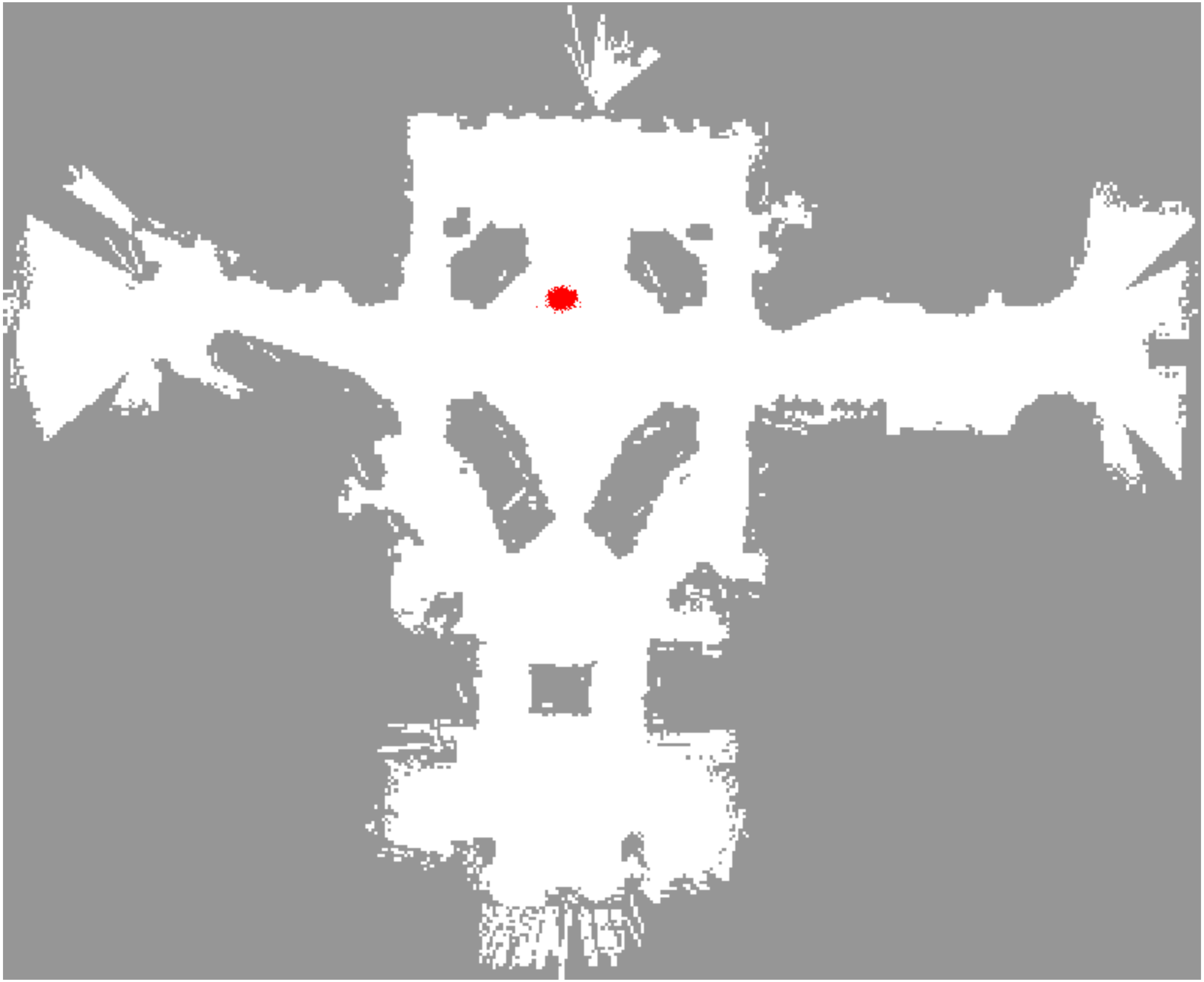


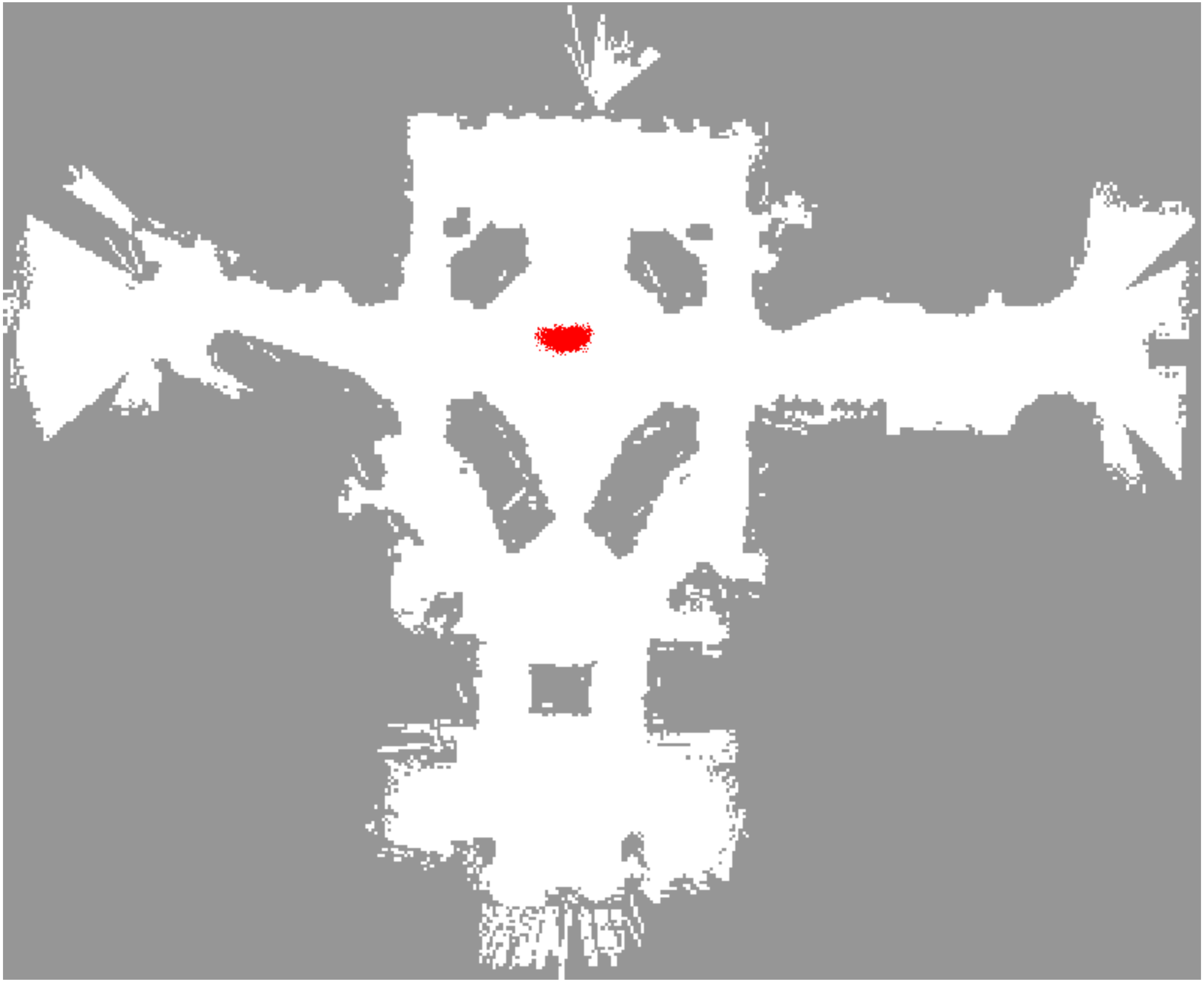


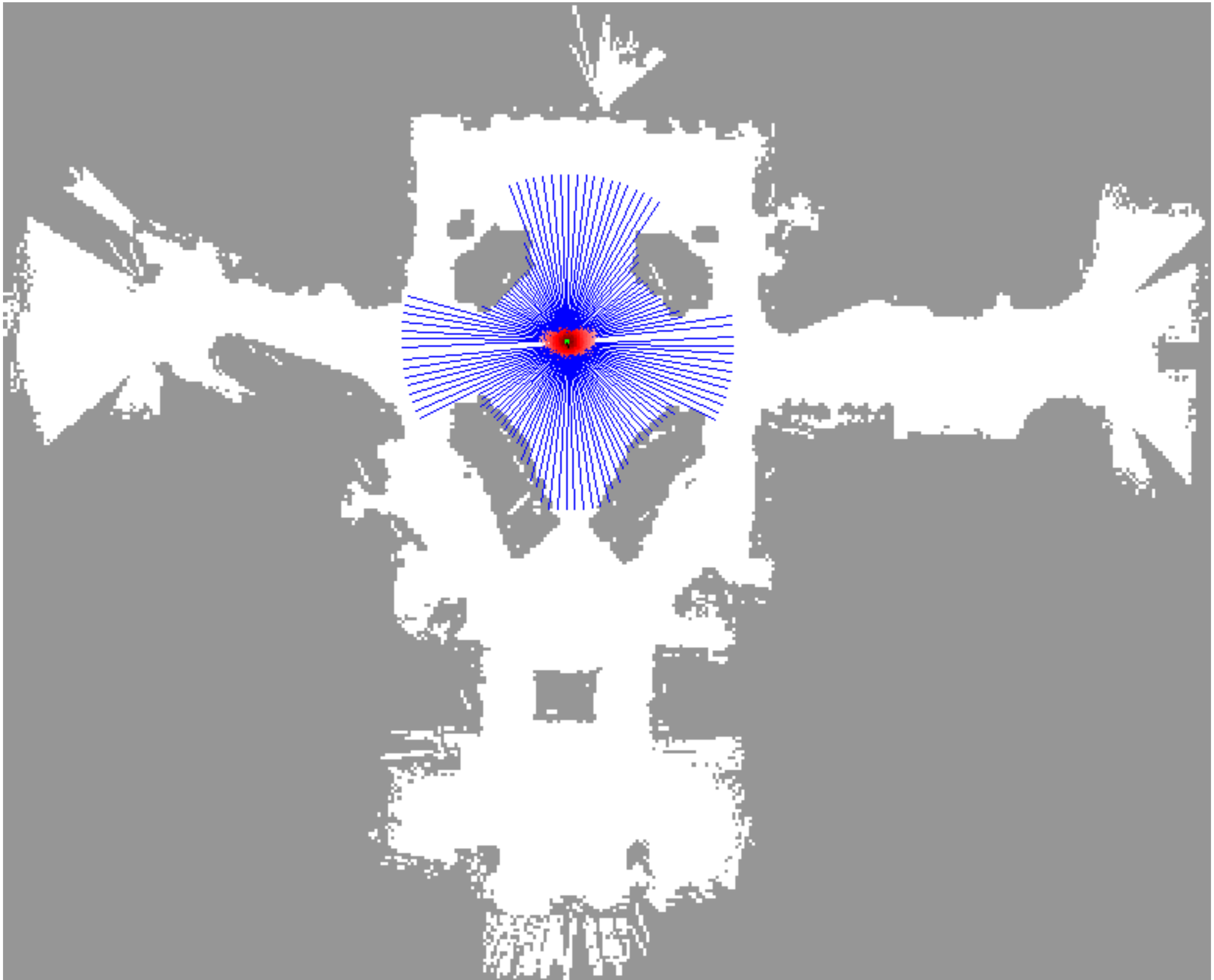












# Video

- Video of tracking through the Smithsonian museum.

# So, where is the robot?

- Average over all particles
- Cluster the particles together and pick the "best" cluster
- Maybe something else?

# Next Problem in Localization Homework

- Augment particle_demo.py to finish implementing a particle filter for the 4 room problem.
  - The motion model says that 50% of the time the robot remains stationary and 50% of the time it moves as requested.
  - Sensor accuracy is 80% (gets the correct room with prob 0.8).
- The methods for the motion model and reweighing the particles are complete. You need to complete:
  - normalize_particles – update the weights so they make a distribution (sum to 1)
  - calc_probability – based on the particles, what is the probability that the robot is in room *x*
  - Resample -- select new particles and assign a uniform weight

# Limitations

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

- Issues:
  - What happens if we resample while the robot is stationary?
  - How can we deal with localization errors (i.e., the kidnapped robot problem)?

# Some Solutions

- 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 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.