ECE276A: Sensing & Estimation in Robotics
Lecture 9: Particle Filter SLAM

Instructor:

Nikolay Atanasov: natanasov@ucsd.edu

Teaching Assistants:

Qiaojun Feng: qif007@eng.ucsd.edu

Tianyu Wang: tiw161@eng.ucsd.edu

Ibrahim Akbar: iakbar@eng.ucsd.edu

You-Yi Jau: yjau@eng.ucsd.edu

Harshini Rajachander: hrajacha@eng.ucsd.edu

UC San Diego

JACOBS SCHOOL OF ENGINEERING

Electrical and Computer Engineering

Simultaneous Localization & Mapping (SLAM)

- Chicken-and-egg problem:
  - **Mapping**: given the robot state trajectory $x_{0:T}$, build a map m of the environment
  - **Localization**: given a map m of the environment, localize the robot and estimate its trajectory $x_{0:T}$
- SLAM is a parameter estimation problem for the parameters $x_{0:T}$ and m. Given a "dataset" of the robot inputs $u_{0:T-1}$ and observations $z_{0:T}$, maximize the data likelihood conditioned on the parameters (MLE) or the posterior likelihood of the parameters given the data (MAP) or use Bayesian Inference to maintain the posterior likelihood of the parameters given the data:
  - **MLE**: $\max_{x_{0:T},m} \log\ p(z_{0:T}, u_{0:T-1} \mid x_{0:T}, m)$
  - **MAP**: $\max_{x_{0:T},m} \log\ p(x_{0:T}, m \mid z_{0:T}, u_{0:T-1})$
  - **BI**: maintain $p(x_{0:T}, m \mid z_{0:T}, u_{0:T-1})$

Simultaneous Localization & Mapping (SLAM)

- Solutions to the SLAM problem exploit the decomposition of the joint pdf due to the Markov assumptions:

$$p(x_{0:T}, m, z_{0:T}, u_{0:T-1}) = \underbrace{p_{0|0}(x_0, m)}_{\text{prior}} \prod_{t=0}^{T} \underbrace{p_h(z_t \mid x_t, m)}_{\text{observation model}} \prod_{t=1}^{T} \underbrace{p_f(x_t \mid x_{t-1}, u_{t-1})}_{\text{motion model}}$$

- The MLE formulation becomes:

$$\max_{x_{0:T}, m} \sum_{t=0}^{T} \log p_h(z_t \mid x_t, m) + \sum_{t=1}^{T} \log p_f(x_t \mid x_{t-1}, u_{t-1})$$

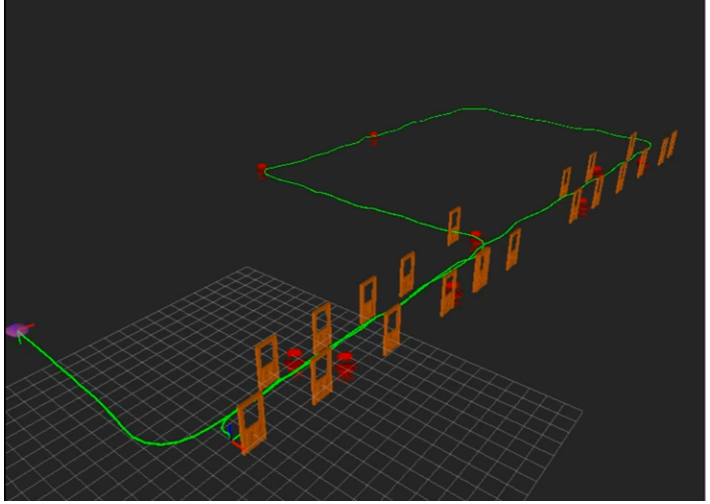
- The MAP formulation is equivalent with the addition of a prior $\log p_{0|0}(x_0, m)$ to the objective function
- The BI formulation uses Bayesian smoothing to maintain $p(x_{0:T}, m \mid z_{0:T}, u_{0:t-1})$

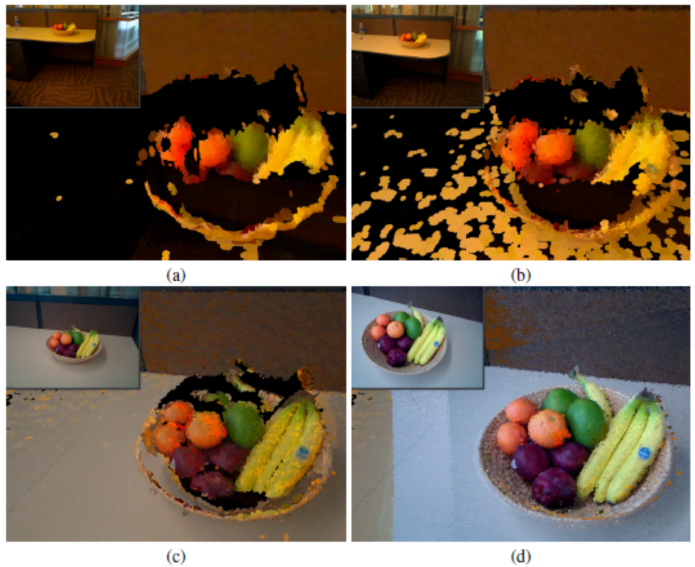
Simultaneous Localization & Mapping (SLAM)

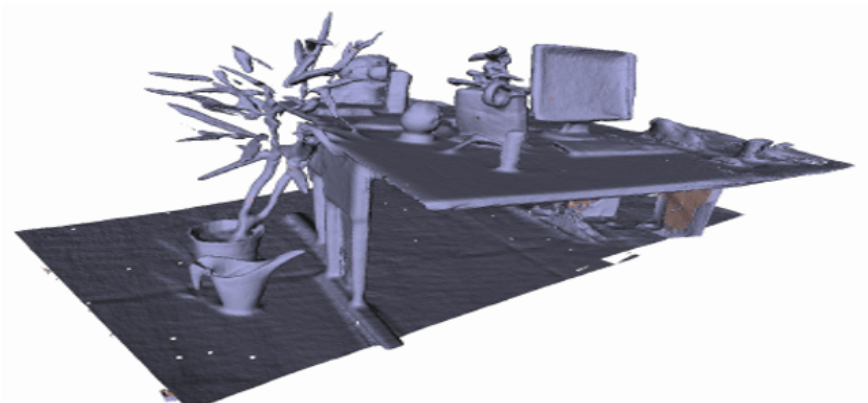
- Early SLAM approaches (with worse performance) were based on simplified versions of the MLE/MAP/BI formulations:
  - Bayes filtering to maintain only $p(x_t, m \mid z_{0:t}, u_{0:t-1})$
  - EM treating x_t is a hidden variable. Given an inital map $m^{(0)}$, e.g., obtained from the first observation, iterate:
    - E: Estimate the distribution of x_t given $m^{(i)}$
    - M: Update $m^{(i+1)}$ by maximizing (over m) the log-likelihood of the measurements conditioned on x_t and m
- The implementation of any of the SLAM approaches depends on the particular representations of the robot states x_t, map m, observations z_t, and control inputs u_t

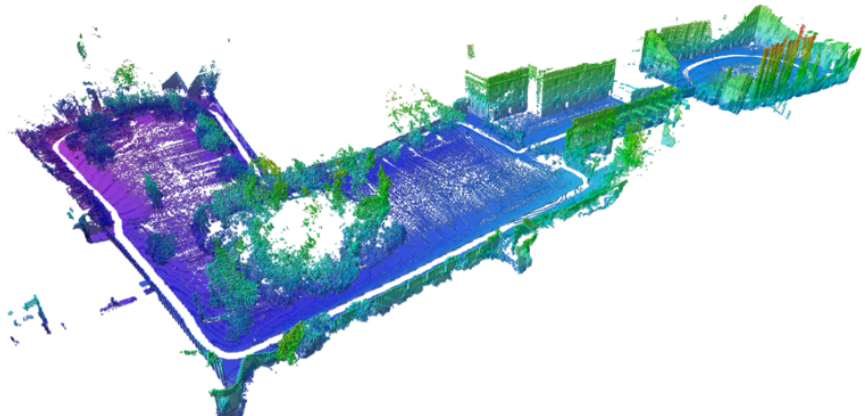
Map Representations

- **Landmark-based**: a collection of objects, each having a position, orientation, and object class
- **Occupancy grid**: a discretization of space into cells with a binary occupancy model
- **Surfels**: a collection of oriented discs containing photometric information
- **Polygonal mesh**: a collection of points and connectivity information among them, forming polygons

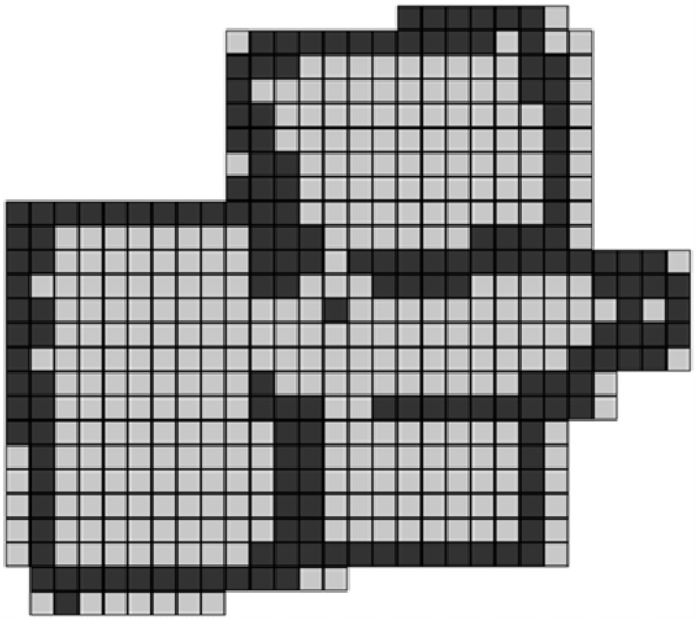








Markov Localization in Occupancy Grid Maps



- **Occupancy grid map**: a grid m with free ($m_i = 0$) and occupied ($m_i = 1$) cells
- **Lidar-based Localization**: Given an occupancy grid map m, a sequence of velocity inputs $u_{0:t-1}$, and a sequence of lidar scans $z_{0:t}$, infer the state x_t of a differential-drive robot
- **Approach**:
  - Use a delta-mixture to represent the pdf of the robot state at time t:

$$p_{t|t}(x_t) := p(x_t \mid z_{0:t}, u_{0:t-1}) \approx \sum_{k=1}^{N_{t|t}} \alpha_{t|t}^{(k)} \delta\left(x_t; \mu_{t|t}^{(k)}\right) \qquad \mu_{t|t}^{(k)} \in SE(2)$$

  - Use the particle filter to propagate the pdf over time
  - **Prediction step**: use the differential-drive motion model
  - **Update step**: use the laser correlation observation model

Markov Localization in Occupancy Grid Maps

- **Prediction step**: for every particle $\mu_{t|t}^{(k)}$, $k = 1, \ldots, N_{t|t}$ compute:

$$\mu_{t+1|t}^{(k)} = f\left(\mu_{t|t}^{(k)}, u_t + \epsilon_t\right)$$

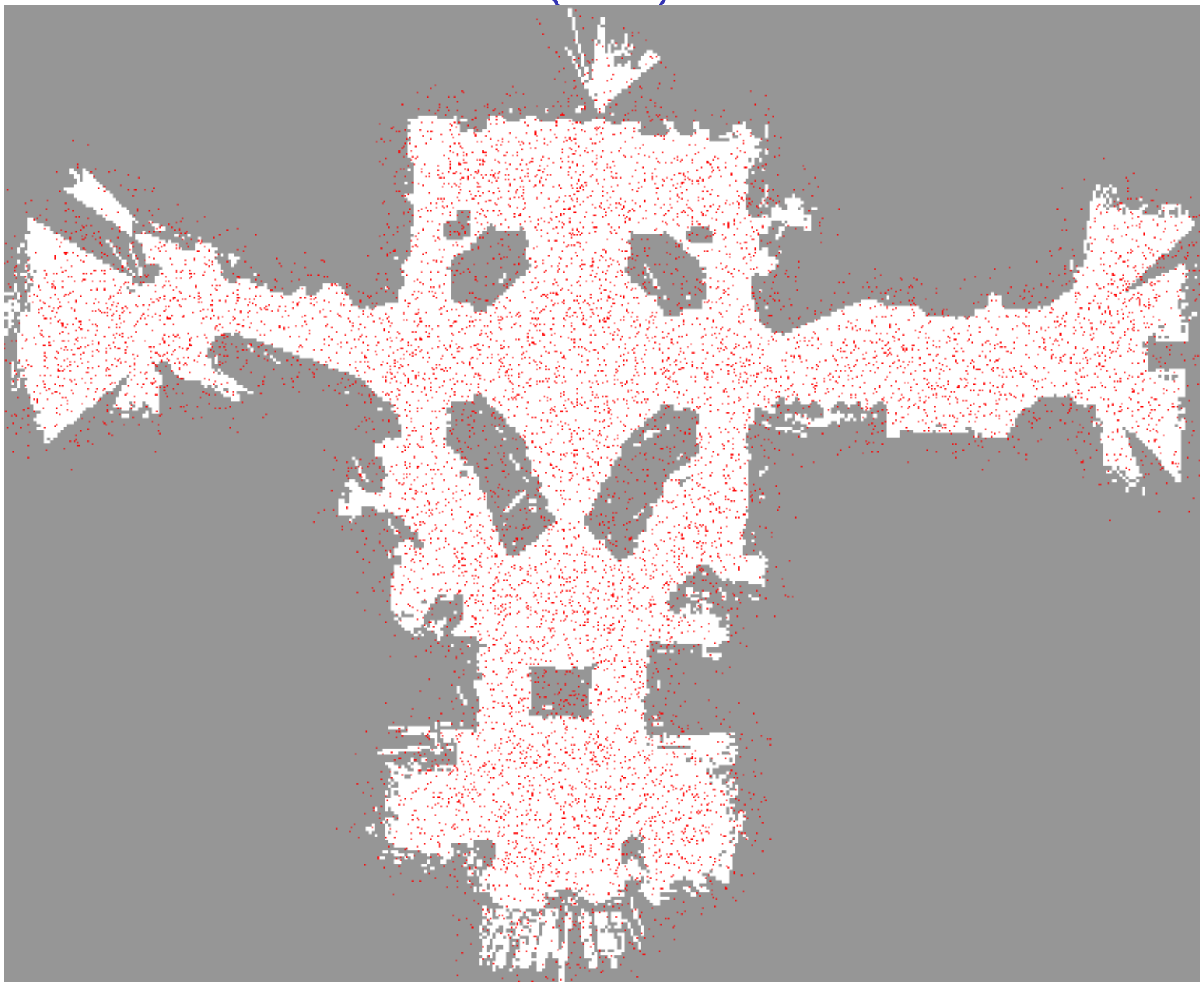
where f is the differential-drive motion model, $u_t = (v_t, \omega_t)$ is the linear and angular velocity input (either known or obtained from the Encoders and IMU), and $\epsilon_t \sim \mathcal{N}(0, \mathcal{E})$ is a 2-D Gaussian motion noise

- **Update step**:
  - Transform the scan z_{t+1} to the world frame using $\mu_{t+1|t}^{(k)}$ for $k = 1, \ldots, N_{t|t}$ and find all cells $y_{t+1}^{(k)}$ in the grid corresponding to the scan
  - Update the particle weights using the laser correlation model:

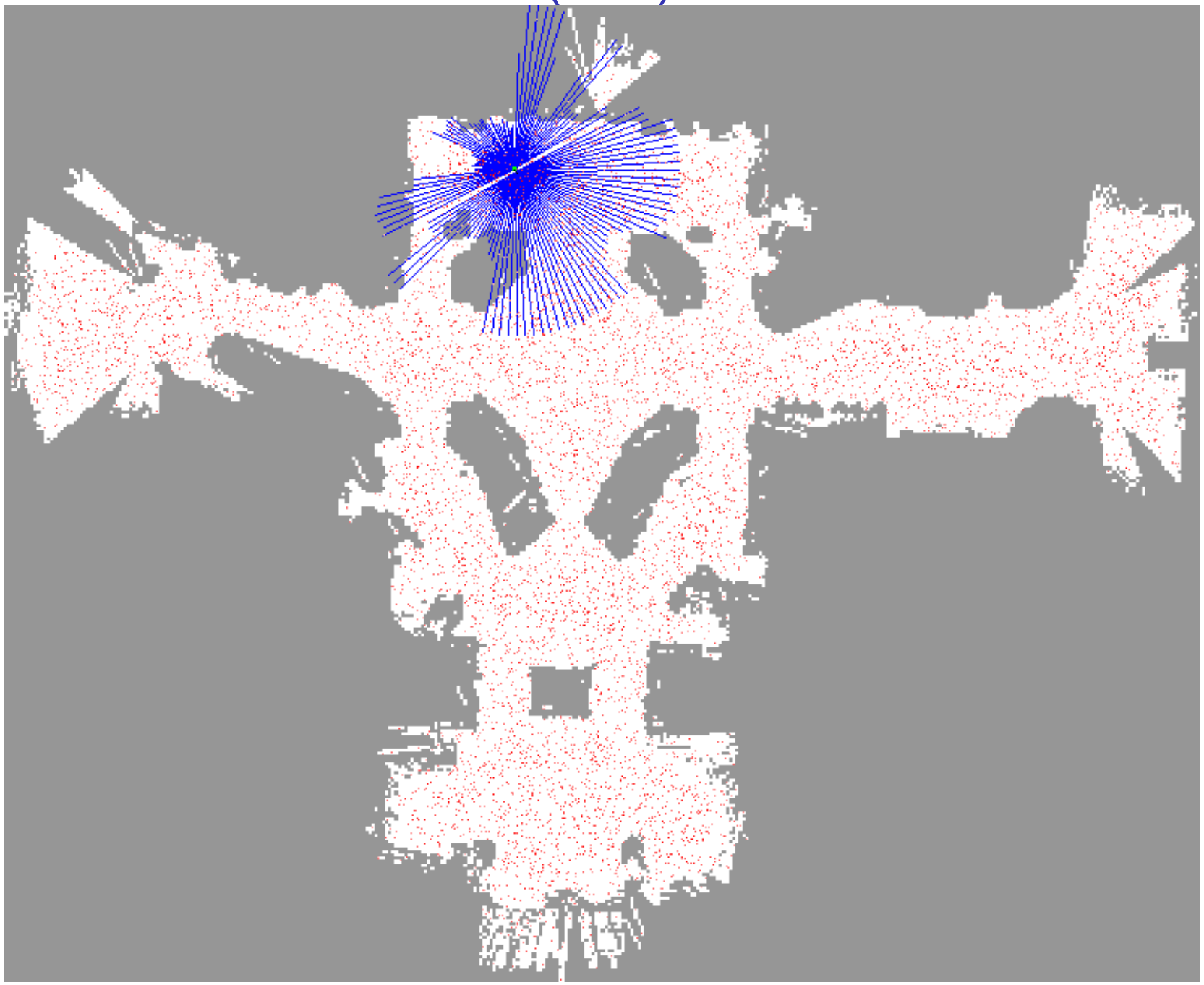
$$p_h(z_{t+1} \mid \mu_{t+1|t}^{(k)}, m) \propto \exp\left(\textbf{corr}\left(y_{t+1}^{(k)}, m\right)\right)$$

- If $N_{eff} := \frac{1}{\sum_{k=1}^{N_{t|t}} \left(\alpha_{t|t}^{(k)}\right)^2} \leq N_{threshold}$, **resample** the particle set $\left\{\mu_{t+1|t+1}^{(k)}, \alpha_{t+1|t+1}^{(k)}\right\}$ via stratified or sample importance resampling

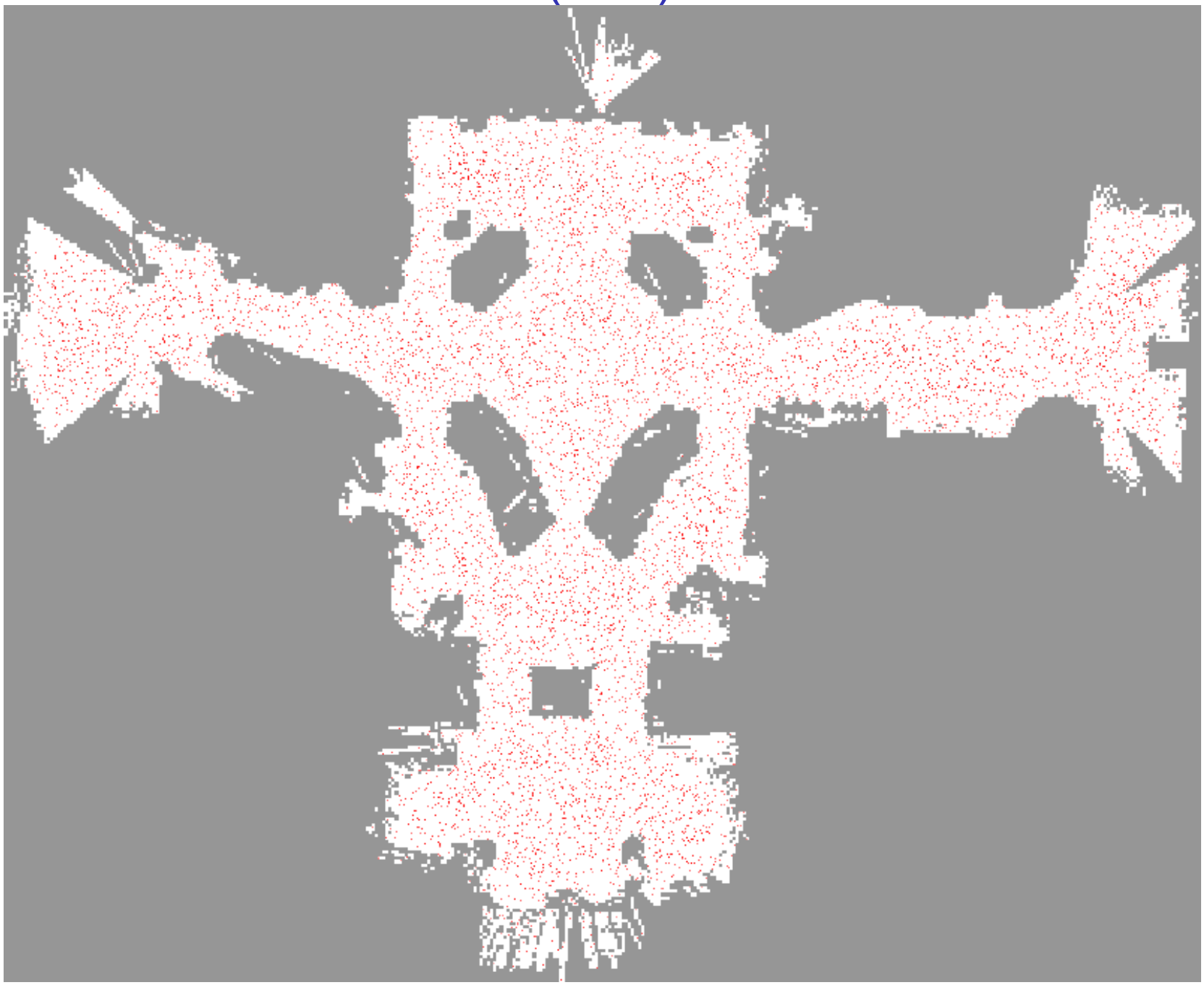
Particle Filter Localization (2-D)



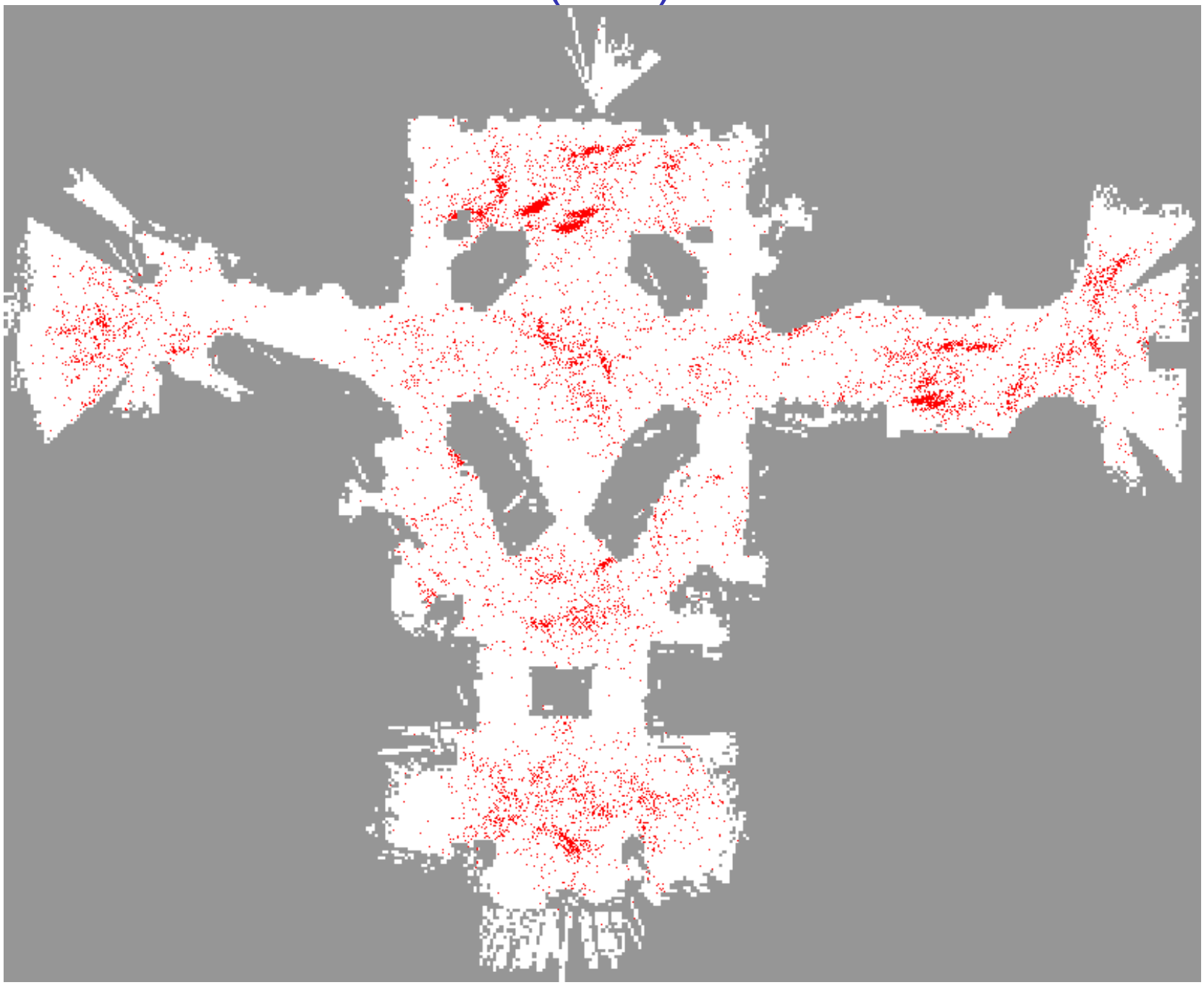
Particle Filter Localization (2-D)



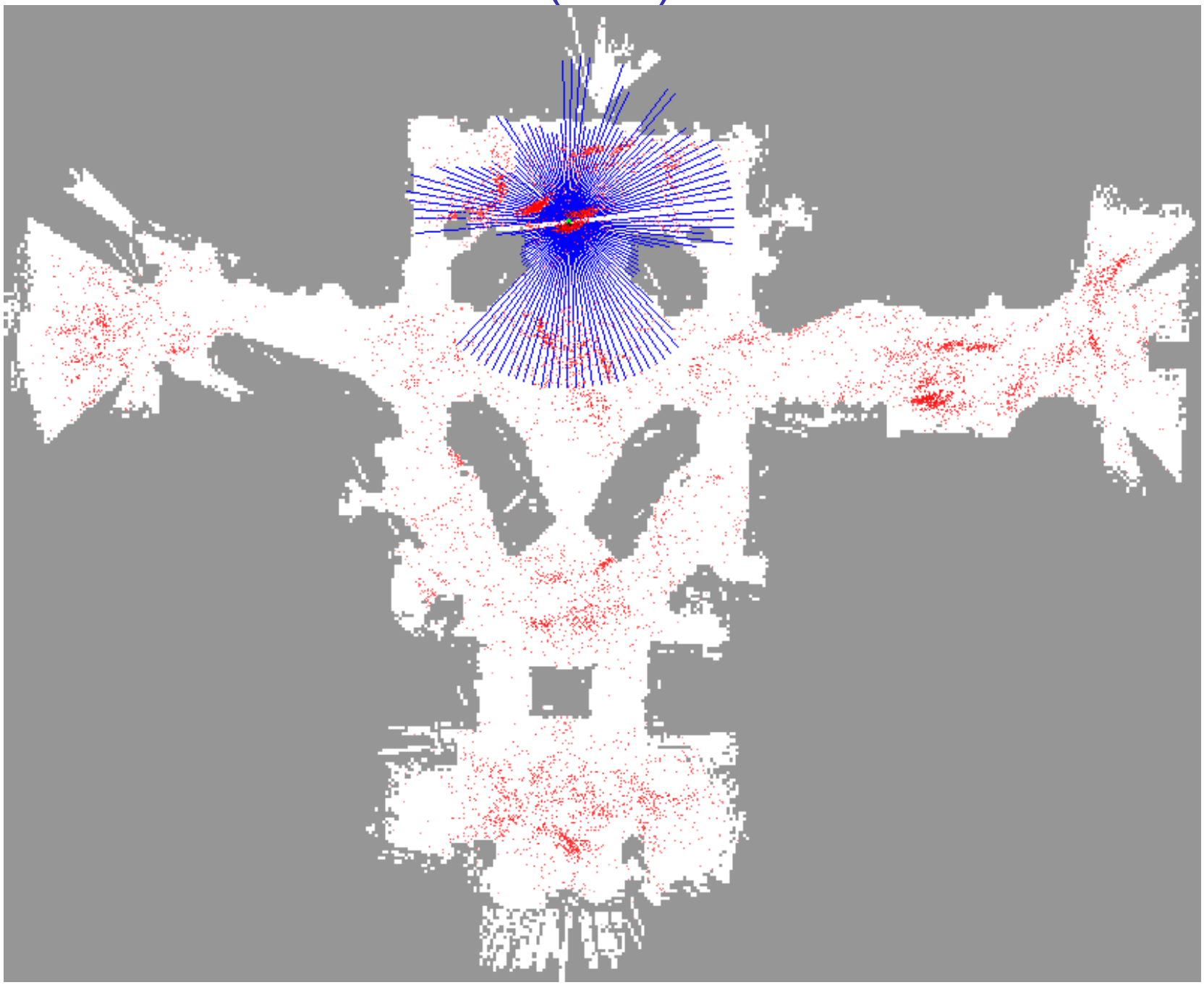
Particle Filter Localization (2-D)



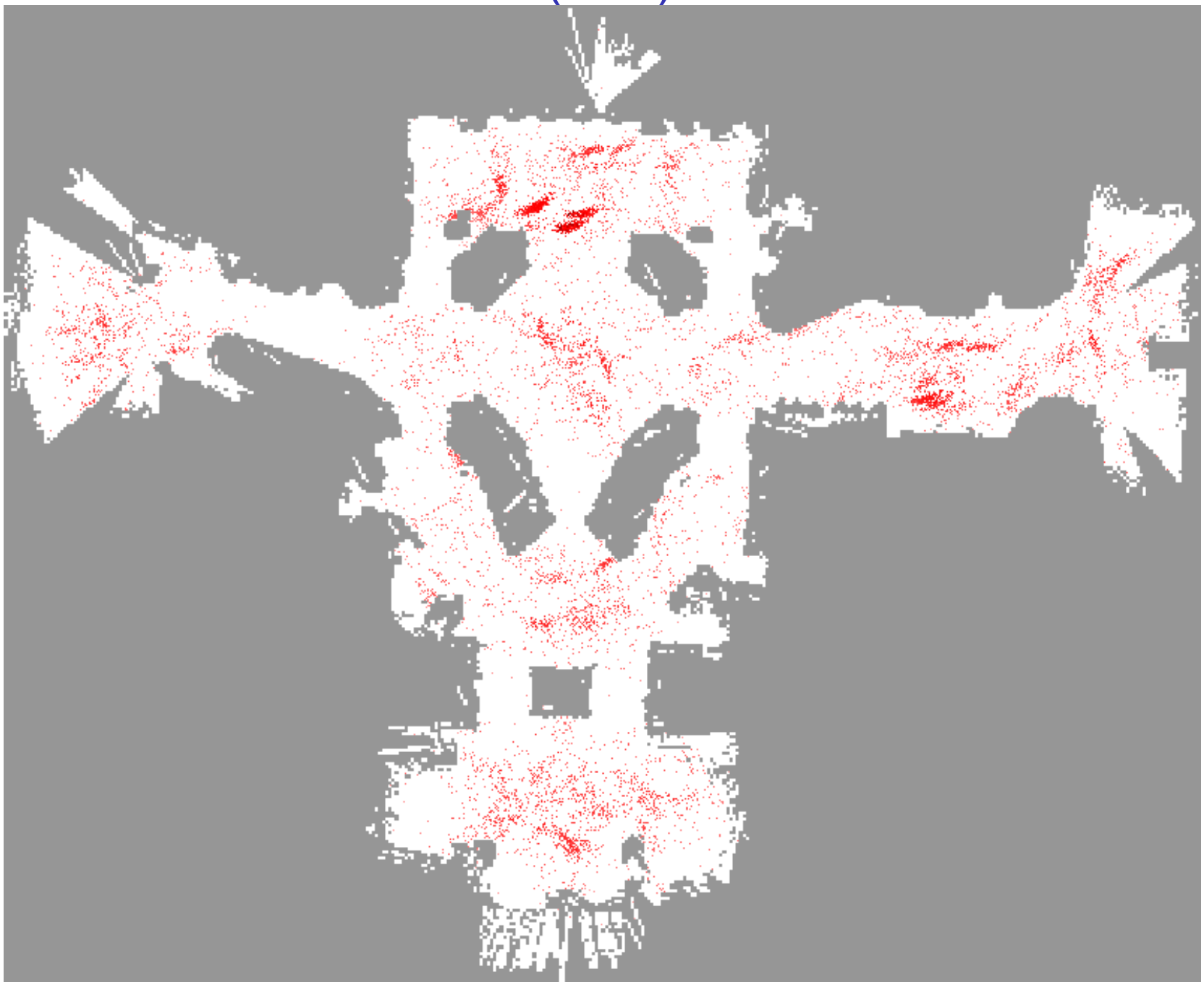
Particle Filter Localization (2-D)



Particle Filter Localization (2-D)



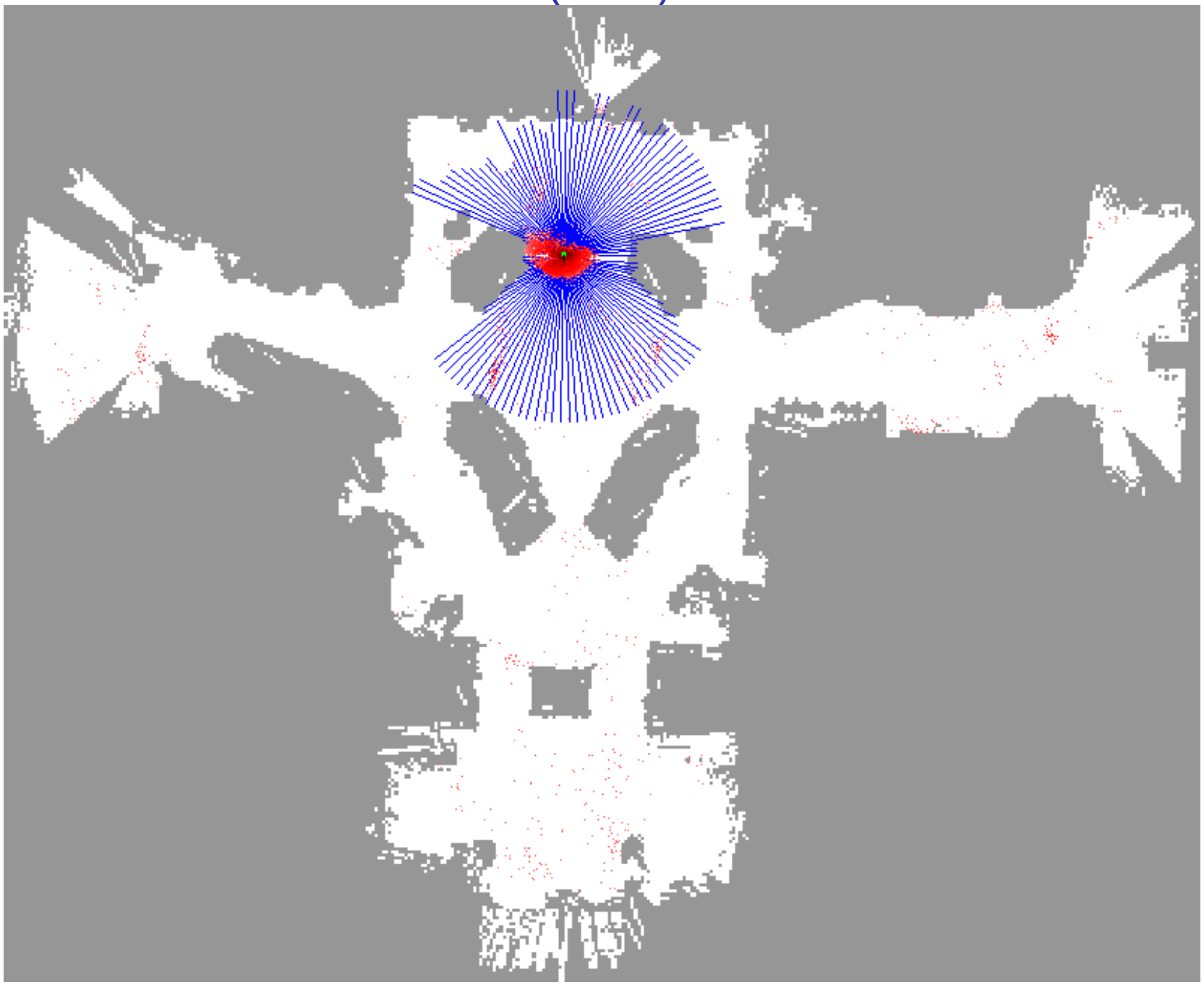
Particle Filter Localization (2-D)



Particle Filter Localization (2-D)

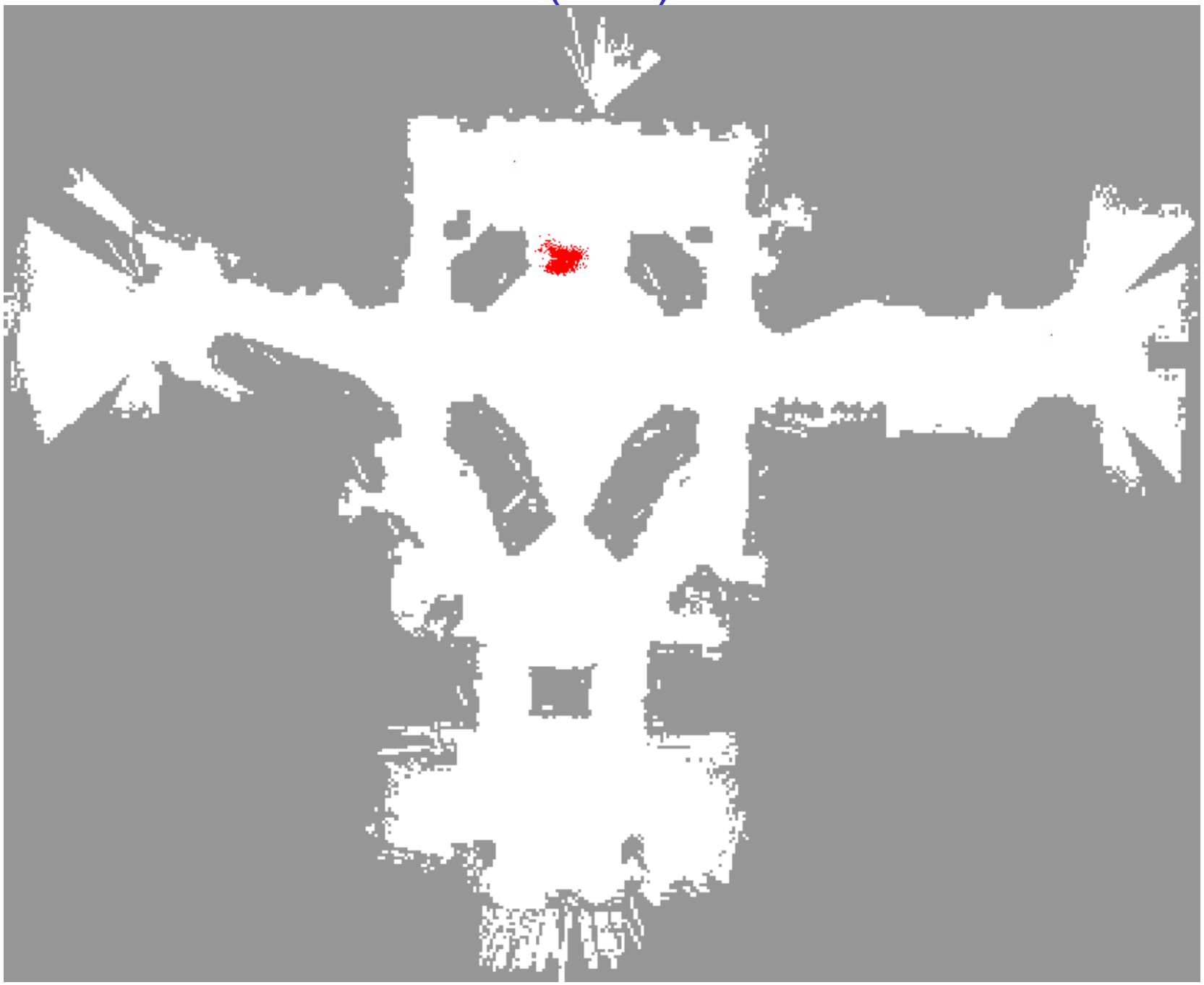
Particle Filter Localization (2-D)

Particle Filter Localization (2-D)

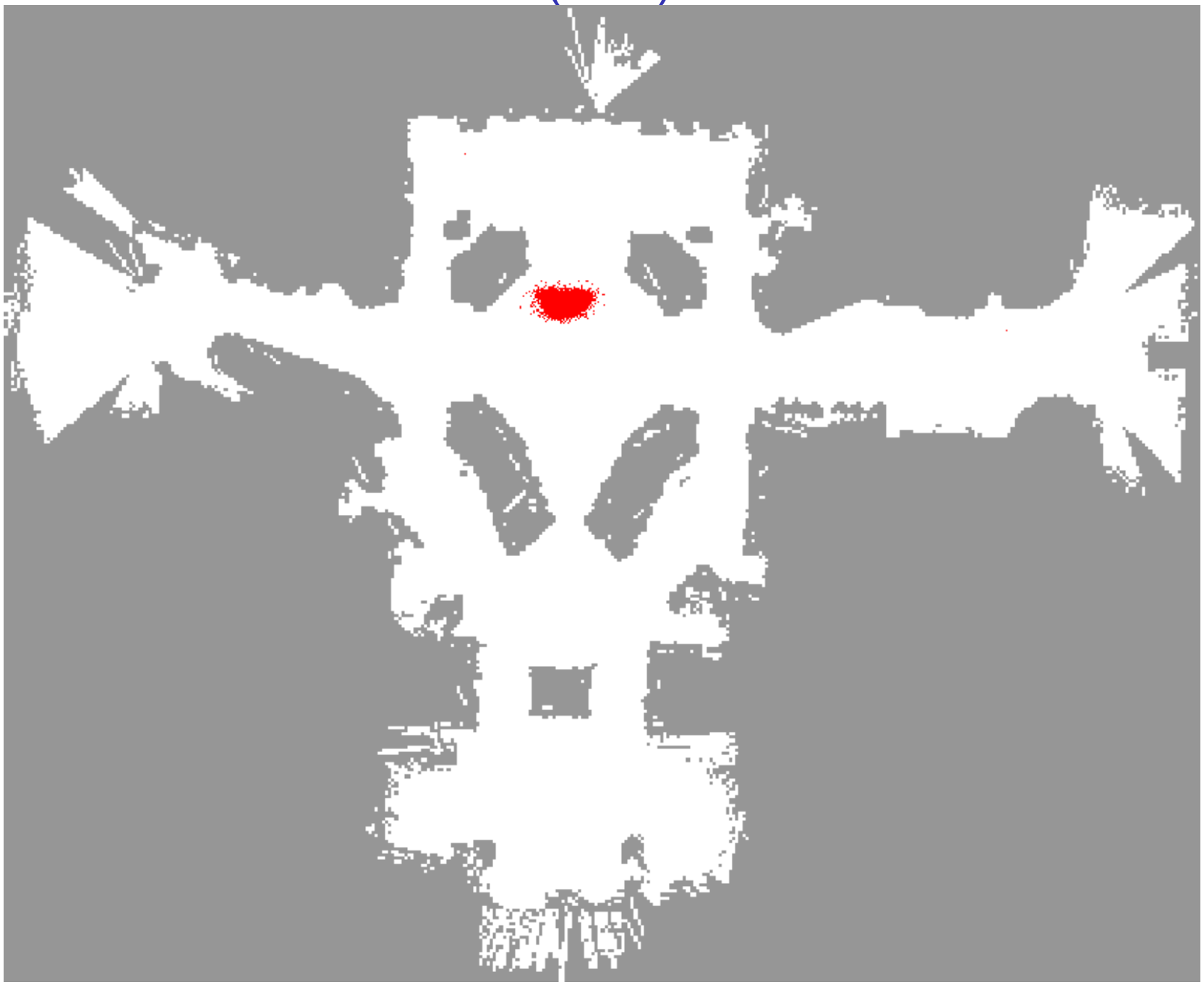


Particle Filter Localization (2-D)

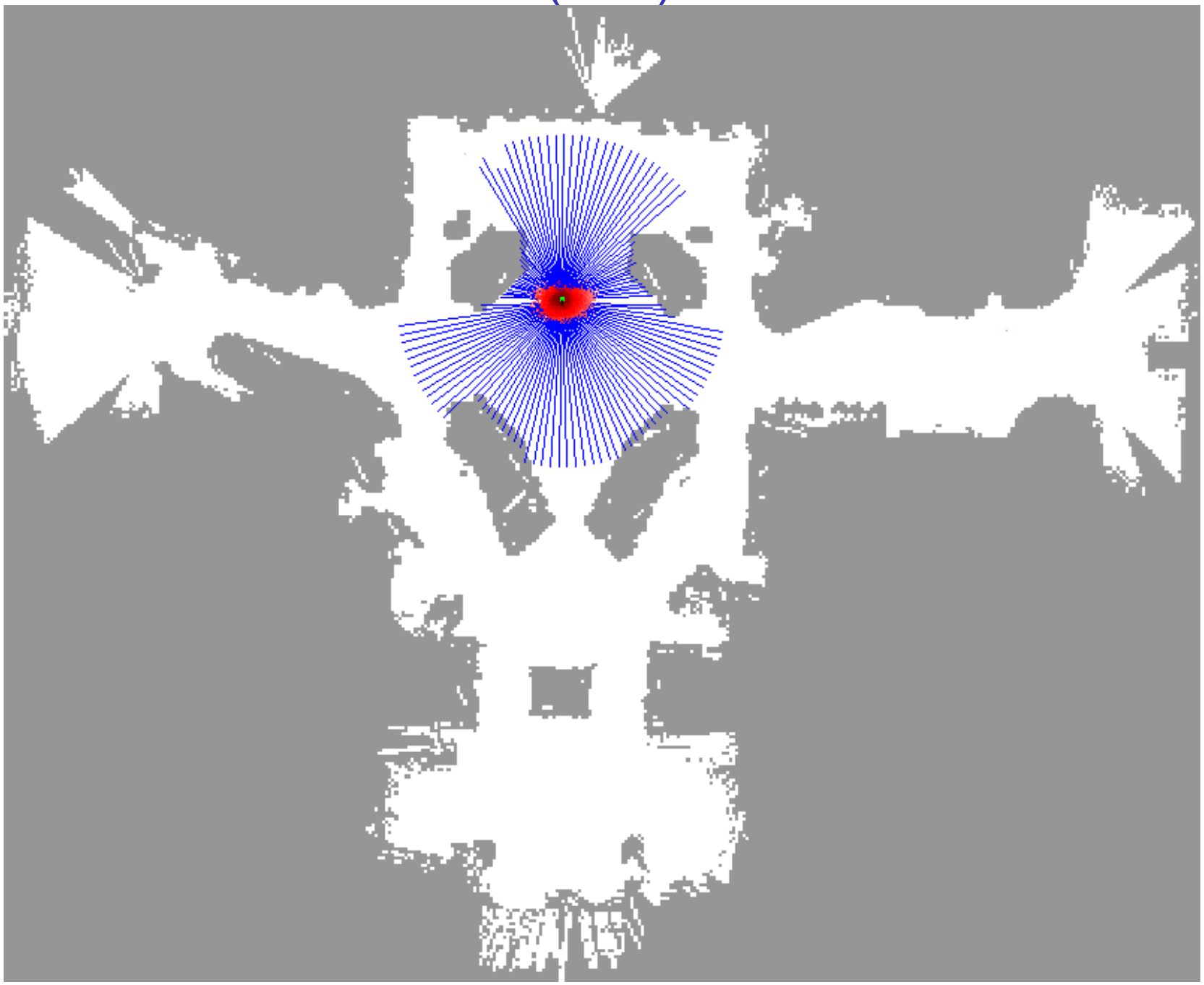
Particle Filter Localization (2-D)



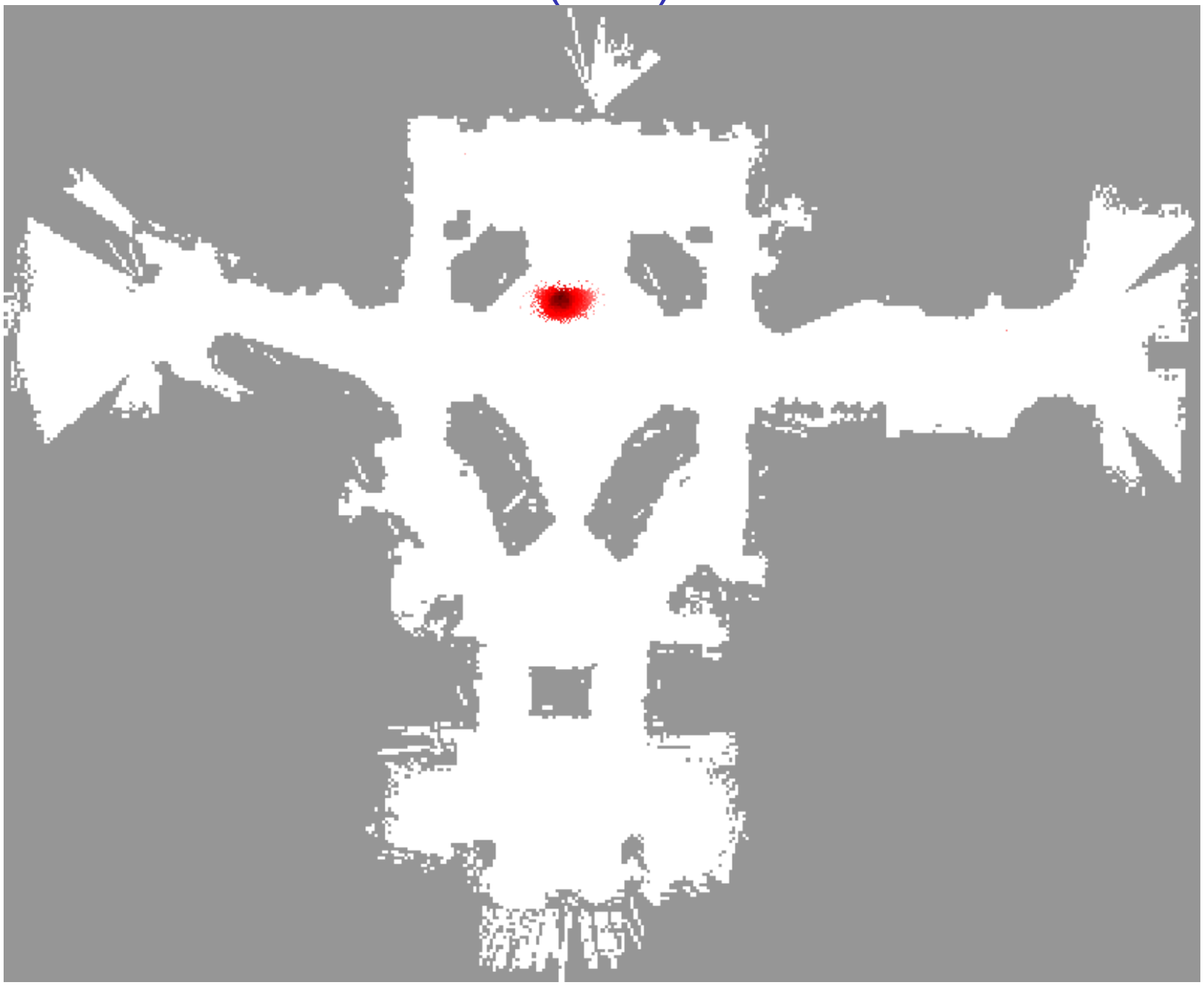
Particle Filter Localization (2-D)



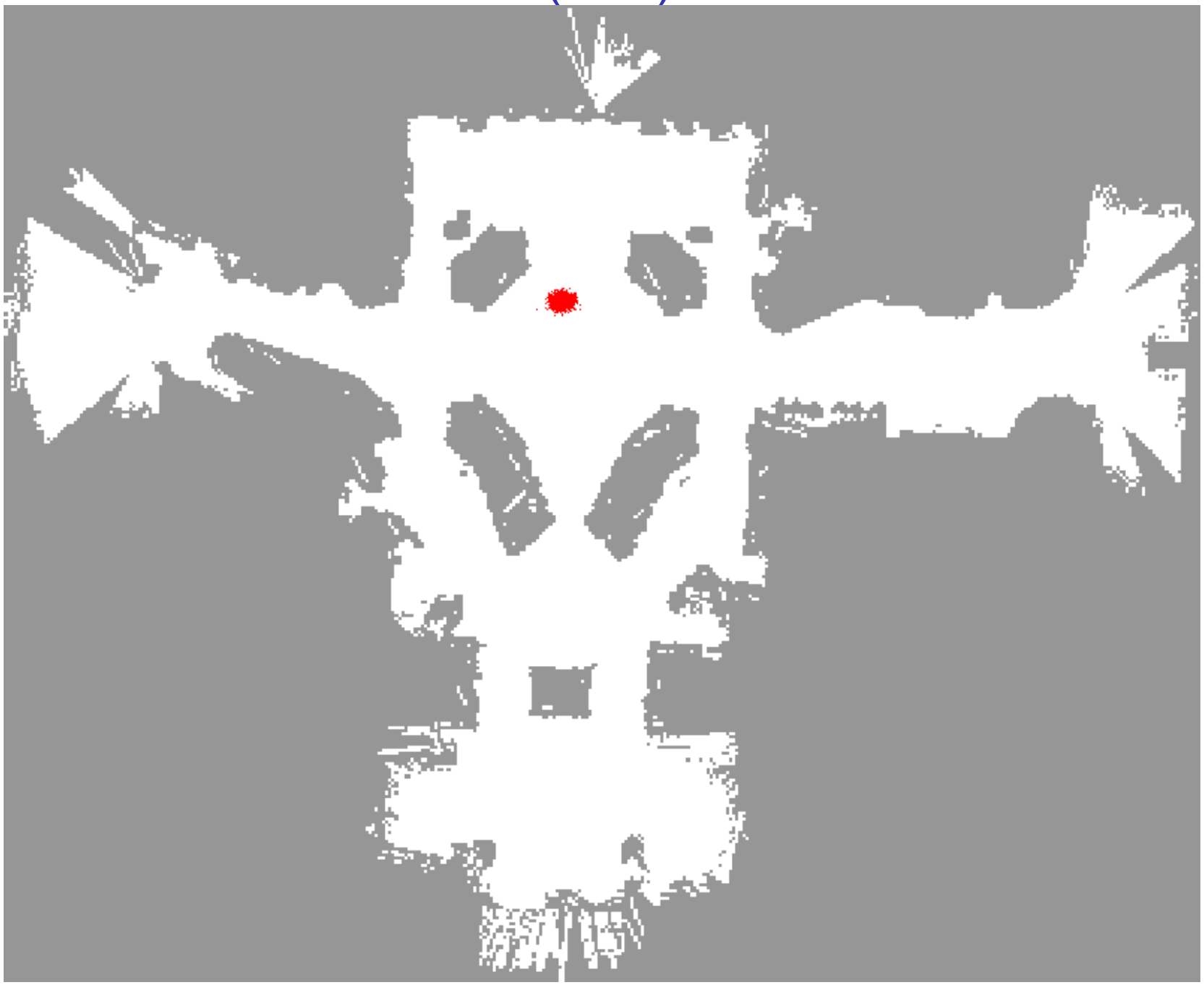
Particle Filter Localization (2-D)



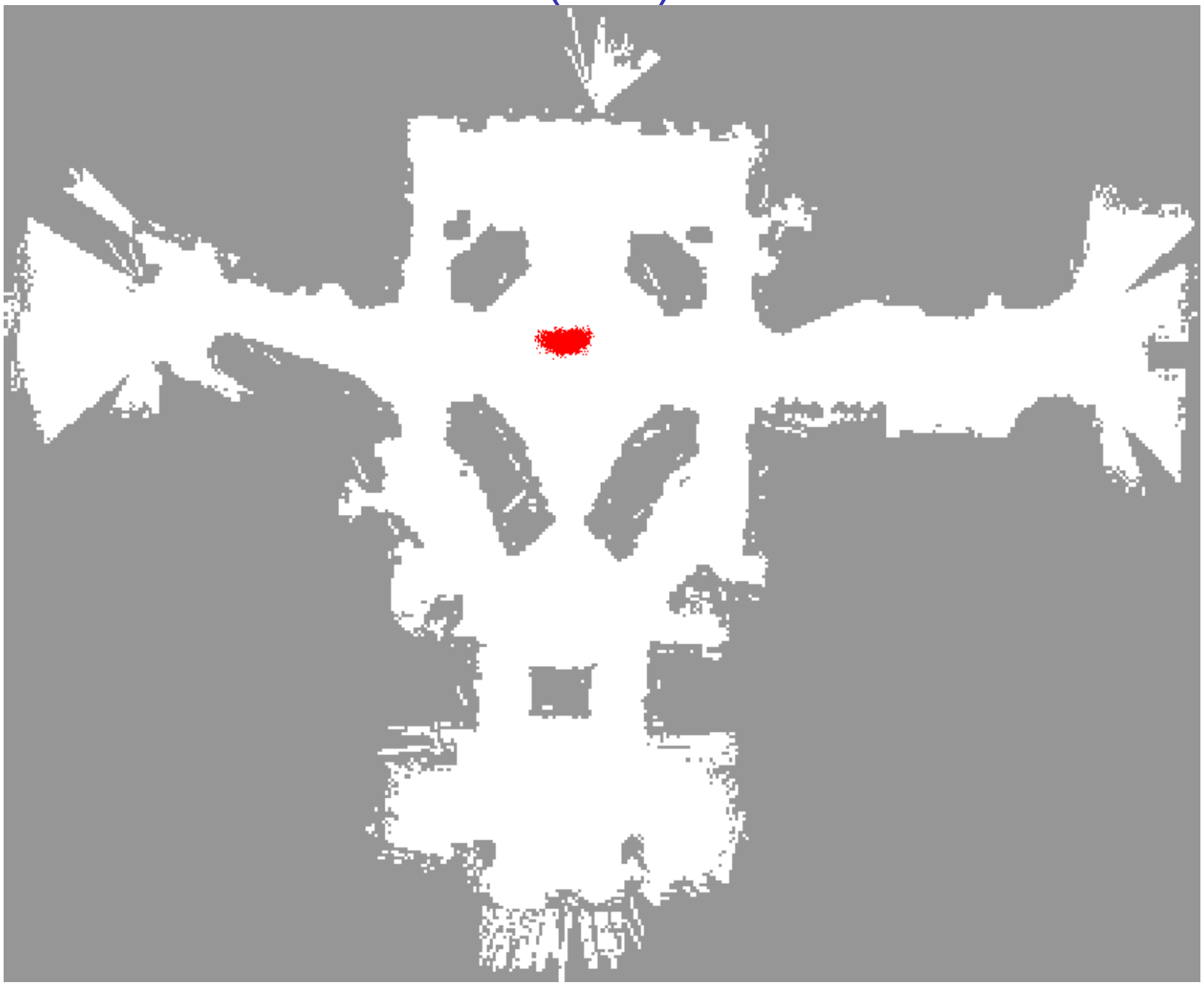
Particle Filter Localization (2-D)



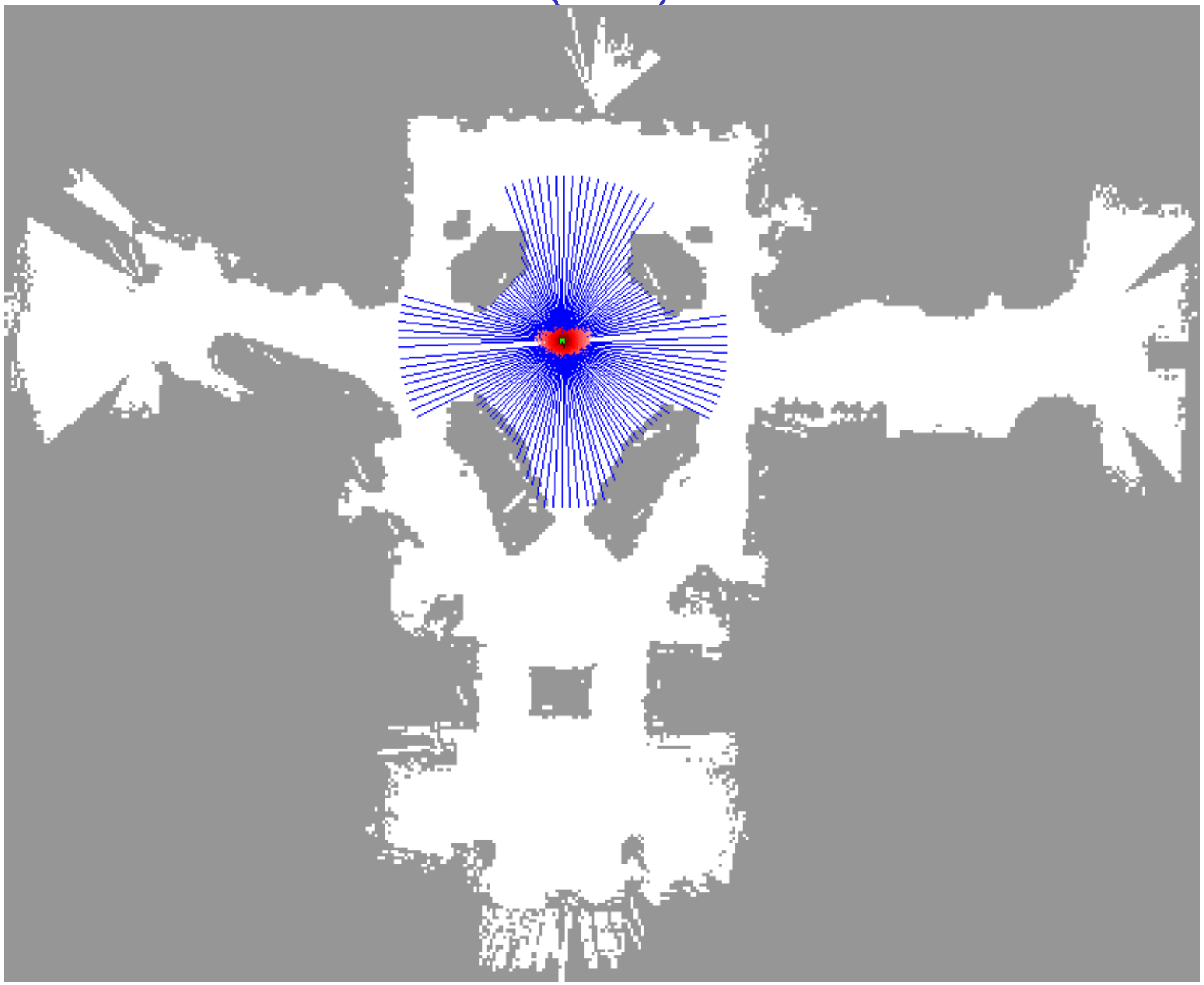
Particle Filter Localization (2-D)



Particle Filter Localization (2-D)



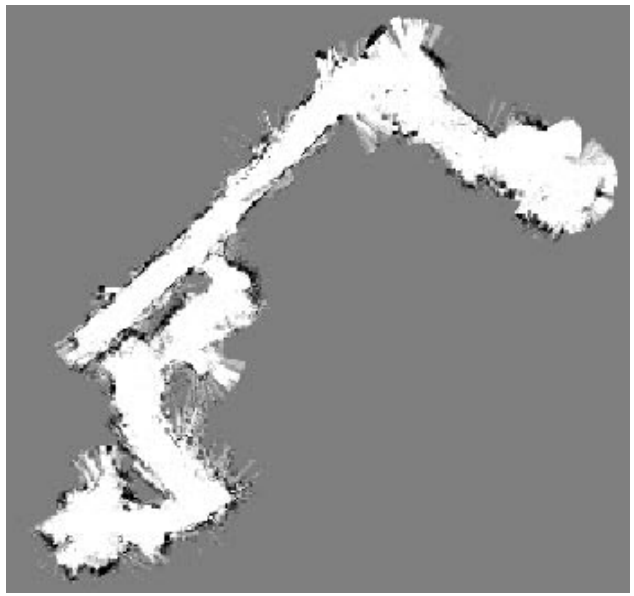
Particle Filter Localization (2-D)



Occupancy Grid Mapping

- **Lidar-based Mapping**: Given the robot trajectory $x_{0:t}$ and a sequence of lidar scans $z_{0:t}$, build an occupancy grid map m of the environment

- Since the map is unknown and the measurements are uncertain, we need to maintain a pdf $p(m \mid z_{0:t}, x_{0:t})$ over the map



- Model the map cells m_i as **independent** Bernoulli random variables

- Given occupancy measurements $z_{0:t}$, the distribution of m_i is:

$$m_i \mid z_{0:t} = \begin{cases} \text{Occupied (1)} & \text{with prob. } \gamma_{i,t} := p(m_i = 1 \mid z_{0:t}, x_{0:t}) \\ \text{Free (0)} & \text{with prob. } 1 - \gamma_{i.t} \end{cases}$$

- To have a probabilistic map representation, we just need to keep a grid of the occupancy probabilities $\gamma_{i,t}$

Occupancy Grid Mapping

- How do we update the map distribution $\gamma_{i,t}$ over time?
- **Bayes Rule**:

$$\begin{aligned}\gamma_{i,t} = p(m_i = 1 \mid z_{0:t}, x_{0:t}) &= \frac{1}{\eta_t} p_h(z_t \mid m_i = 1, x_t) p(m_i = 1 \mid z_{0:t-1}, x_{0:t-1}) \\ &= \frac{1}{\eta_t} p_h(z_t \mid m_i = 1, x_t) \gamma_{i,t-1}\end{aligned}$$

$$(1 - \gamma_{i,t}) = p(m_i = 0 \mid z_{0:t}, x_{0:t}) = \frac{1}{\eta_t} p_h(z_t \mid m_i = 0, x_t)(1 - \gamma_{i,t-1})$$

- The **odds ratio** of a binary random variable m_i updated over time via Bayes rule and measurements $z_{0:t}$ is:

$$\begin{aligned}o(m_i \mid z_{0:t}, x_{0:t}) := \frac{p(m_i = 1 \mid z_{0:t}, x_{0:t})}{p(m_i = 0 \mid z_{0:t}, x_{0:t})} &= \frac{\gamma_{i,t}}{1 - \gamma_{i,t}} \\ = \underbrace{\frac{p_h(z_t \mid m_i = 1, x_t)}{p_h(z_t \mid m_i = 0, x_t)}}_{g_h(z_t \mid m_i, x_t)} &\quad \underbrace{\frac{\gamma_{i,t-1}}{1 - \gamma_{i,t-1}}}_{o(m_i \mid z_{0:t-1}, x_{0:t-1})}\end{aligned}$$

Occupancy Grid Mapping

- Estimating the pdf of m_i conditioned on $z_{0:t}$ is equivalent to accumulating the **log-odds ratio**:

$$\begin{aligned}\lambda(m_i \mid z_{0:t}, x_{0:t}) :&= \log o(m_i \mid z_{0:t}, x_{0:t}) = \log\left(g_h(z_t \mid m_i, x_t) o(m_i \mid z_{0:t-1}, x_{0:t-1})\right) \\ &= \lambda(m_i \mid z_{0:t-1}, x_{0:t-1}) + \log g_h(z_t \mid m_i, x_t) \\ &= \lambda(m_i) + \sum_{s=0}^{t} \log g_h(z_s \mid m_i, x_s)\end{aligned}$$

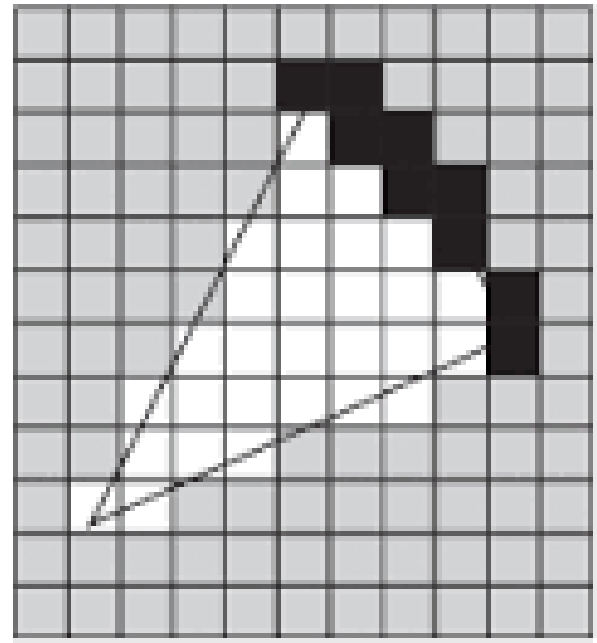
- Probabilistic occupancy grid mapping reduces to keeping track of the cell log-odds:

$$\lambda_{i,t} = \lambda_{i,t-1} + \Delta\lambda_{i,t-1} \quad \leftarrow \text{Measurement “trust”}$$

- Since the map cells are assumed independent, we can use a simpler model (than the lidar correlation model) for $p_h(z_t \mid m_i, x_t)$ by specifying how much we trust the occupancy measurement of cell i:

$$g_h(1 \mid m_i, x_t) = \frac{p_h(z_t = 1 \mid m_i = 1, x_t)}{p_h(z_t = 1 \mid m_i = 0, x_t)} = \frac{80\%}{20\%} = 4 \qquad g_h(0 \mid m_i, x_t) = \frac{1}{4}$$

Occupancy Grid Mapping (Summary)



- Maintain a grid of the map log-odds $\lambda_{i,t}$
- Given a new laser scan z_{t+1}, transform it to the world frame using the robot pose x_{t+1}
- Determine the cells that the lidar beams pass through (e.g., using Bresenham's line rasterization algorithm)
- For each observed cell i, decrease the log-odds if it was observed free or increase the log-odds if the cell was observed occupied:

$$\lambda_{i,t+1} = \lambda_{i,t} + \log g_h(z_{t+1} \mid m_i, x_{t+1})$$

- Constrain $\lambda_{MIN} \leq \lambda_{i,t} \leq \lambda_{MAX}$ to avoid overconfident estimation
- May introduce a decay on $\lambda_{i,t}$ to handle changing maps
- The map pdf $\gamma_{i,t}$ can be recovered from the log-odds $\lambda_{i,t}$:

$$\gamma_{i,t} = p(m_i = 1 \mid z_{0:t}, x_{0:t}) = 1 - \frac{1}{1 + \exp(\lambda_{i,t})}$$

Lidar-based Localization & Occupancy Grid Mapping

- Initial particle set $\mu_{0|0}^{(k)} = (0, 0, 0)^T \in SE(2)$ with weights $\alpha_{0|0}^{(k)} = \frac{1}{N}$
- Use the first laser scan to initialize the map:
  1. convert the scan to Cartesian coordinates and transform it from the body frame to the world frame
  2. convert the scan to cells (via **bresenham2D** or **cv2.drawContours**) and update the map log-odds
- Use the differential-drive model to predict the motion of each particle
- Use the laser scan from each particle to compute map correlation (via **getMapCorrelation**) and update the particle weights
- Choose the best particle, project the laser scan, and update the map log-odds (in general, each particle should maintain its own map)
- **Textured map**: use the RGBD images from the best particle's pose to assign colors to the occupancy grid cells

Popular SLAM Algorithms

- **Occupancy Grid Map**: a grid of Bernoulli random variables
  - **Fast SLAM** (Montemerlo et al.)
    - exploits that the occupancy grid cells are independent conditioned on the robot trajectory:

$$p(x_{0:t}, m \mid z_{0:t}, u_{0:t-1}) = p(x_{0:t} \mid z_{0:t}, u_{0:t-1}) \prod_i p(m_i \mid z_{0:t}, x_{0:t})$$

    - uses a particle filter to maintain the robot trajectory pdf and log-odds mapping to maintain a probabilistic map for every particle
  - **Kinect Fusion** (Newcombe et al.)
    - matches consecutive RGBD point clouds using the iterative closest point (ICP)
    - updates a grid discretization of the truncated signed distance function (TSDF) representing the scene surface via weighted averaging

Popular SLAM Algorithms

- **Landmark-based Map**: a collection of Gaussian random variables
  - **Rao-Blackwellized Particle Filter** uses particles for $x_{0:t}$ and Gaussian distributions for the landmark positions
  - **Kalman Filter** uses Gaussian distributions both for the robot and landmark poses
  - **Factor graphs** (State of the Art)
    - Estimate the whole robot trajectory $x_{0:t}$ using the MAP formulation
    - The log observation and motion models are proportional to the Mahalonobis distance
    - This leads to a **sparse** (due to the Markov assumptions), **nonlinear** (due to the motion and observation models) **least-squares** (due to the Mahalonobis distance) optimization problem
    - The problem can be solved using the Gauss-Newton descent algorithm (an approximation to Newton's method that avoids computing the Hessian)