Introduction to Aerial Robotics
Lecture 9

Shaojie Shen

Assistant Professor

Dept. of ECE, HKUST

10 November 2015

Outline

- Extended Kalman Filter
- Particle Filter

Extended Kalman Filter

Bayes' Filter

State

Control input

- **Prior:** $p(x_0)$
- **Process model:** $f(x_t \mid x_{t-1}, u_t)$
- **Measurement model:** $g(z_t \mid x_t)$

Measurement

- **Prediction step:**
- $p(x_t \mid z_{1:t-1}, u_{1:t}) = \int f(x_t \mid x_{t-1}, u_t)\, p(x_{t-1} \mid z_{1:t-1}, u_{1:t-1})\, dx_{t-1}$
- **Update step:**
- $p(x_t \mid z_{1:t}, u_{1:t}) = \dfrac{g(z_t \mid x_t)\, p(x_t \mid z_{1:t-1}, u_{1:t})}{\int g(z_t \mid x'_t)\, p(x'_t \mid z_{1:t-1}, u_{1:t})\, dx'_t}$

Assumptions

- The prior state of the robot is represented by a Gaussian distribution
  - $p(x_0) \sim N(\mu_0, \Sigma_0)$
- The process model $f(x_t \mid x_{t-1}, u_t)$ is linear with additive Gaussian white noise
  - $x_t = A_t\, x_{t-1} + B_t\, u_t + n_t$
  - $n_t \sim N(0, Q_t)$
- The measurement model $g(z_t \mid x_t)$ is linear with additive Gaussian white noise
  - $z_t = C_t\, x_t + v_t$
  - $v_t \sim N(0, R_t)$

Kalman Filter

- Prior:
  - $p(x_0) \sim N(\mu_0, \Sigma_0)$
- Transition model:
  - $x_t = A_t\ x_{t-1} + B_t\ u_t + n_t$
  - $n_t \sim N(0, Q_t)$
- Measurement model:
  - $z_t = C_t\, x_t + v_t$
  - $v_t \sim N(0, R_t)$

- Prior:
  - μ_{t-1}, Σ_{t-1}
- Prediction:
  - $\bar{\mu}_t = A_t\ \mu_{t-1} + B_t\, u_t$
  - $\bar{\Sigma}_t = A_t\, \Sigma_{t-1}\, A_t^T + Q_t$
- Update:
  - $\mu_t = \bar{\mu}_t + K_t(z_t - C_t\, \bar{\mu}_t)$
  - $\Sigma_t = \bar{\Sigma}_t - K_t\, C_t\, \bar{\Sigma}_t$
  - $K_t = \bar{\Sigma}_t\, C_t^T\, (C_t\, \bar{\Sigma}_t\, C_t^T + R_t)^{-1}$

Example Problem

$$x_t = x_{t-1} + u_t + n_t$$

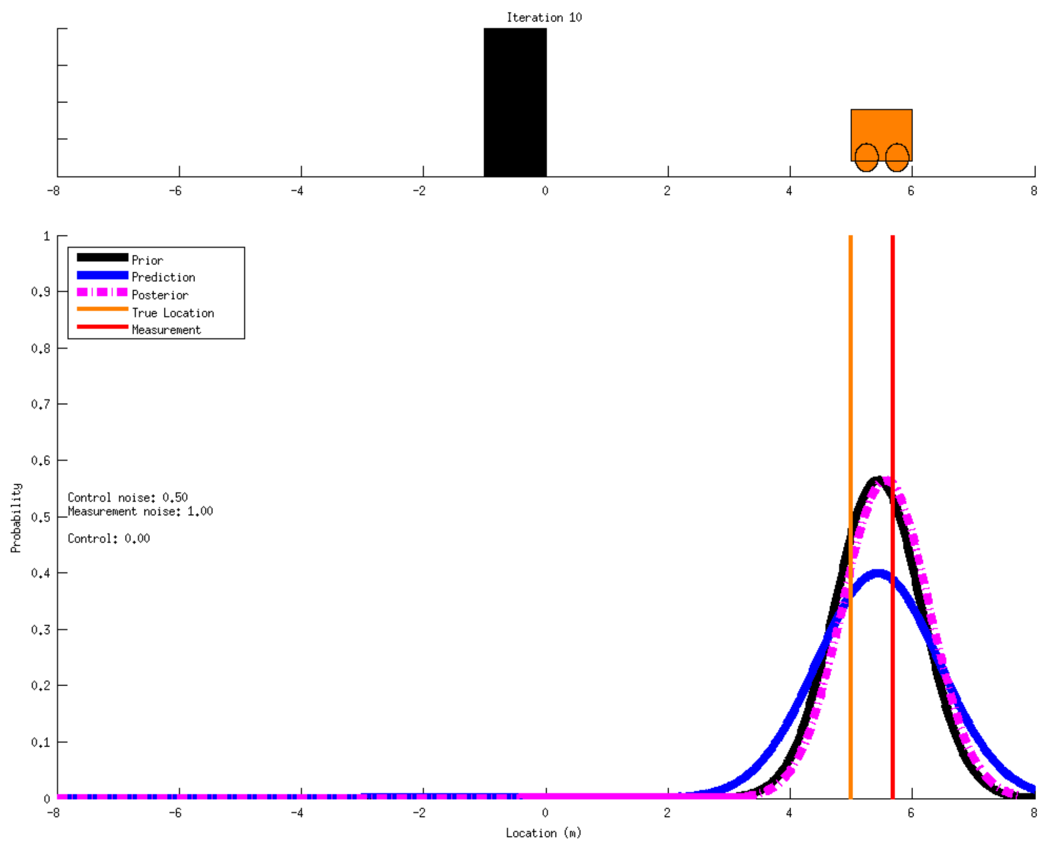
$$Q_t = 0.5$$

$$A_t = B_t = 1$$

$$z_t = x_t + v_t$$

$$R_t = 1.0$$

$$C_t = 1$$

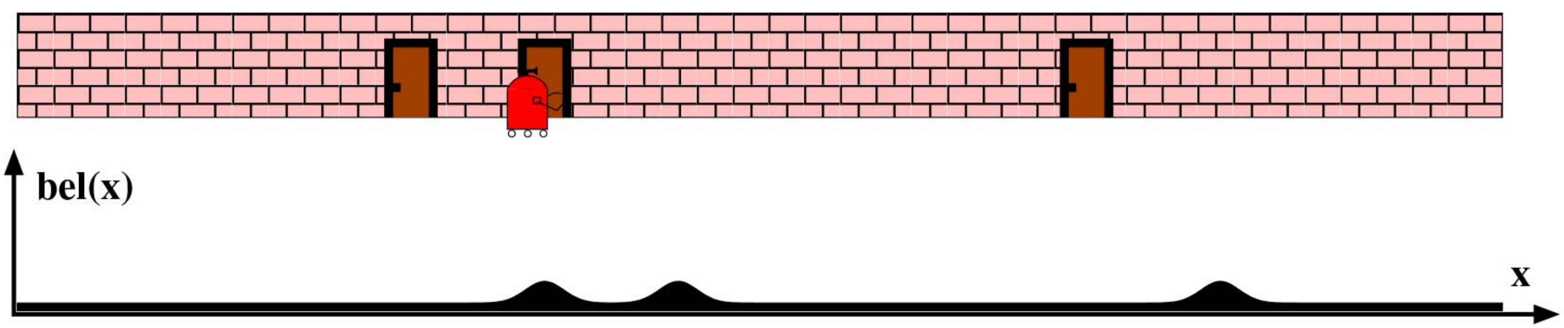


Continuous Dynamics

- Can convert continuous time systems
- $\dot{x} = f(x, u, n) = A\,x + B\,u + U\,n$
- Into discrete time systems using one-step Euler integration
- $x_t = F\,x_{t-1} + G\,u_t + V\,n_t$
- $F = (I + \delta t\,A), G = \delta t\,B,\ V = \delta t\,U$
- This will introduce some error, but the observations can help correct it
- Prediction:
  - $\bar{\mu}_t = F\,\mu_{t-1} + G\,u_t$
  - $\bar{\Sigma}_t = F\,\Sigma_{t-1} F^T + V\,Q\,V^T$

Kalman Filter Discussion

- **Advantages:**
  - Simple
  - Purely matrix operations
    - Computationally efficient, even for high dimensional systems
- **Disadvantages:**
  - Assumes everything is linear and Gaussian
  - Unimodal distribution
    - Cannot handle multiple hypotheses



Extended Kalman Filter (EKF)

Assumptions

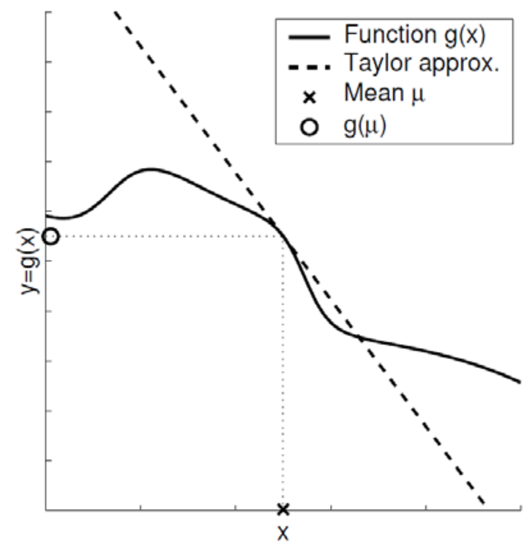
- The prior state of the robot is represented by a Gaussian distribution
  - $p(x_0) \sim N(\mu_0, \Sigma_0)$
- The continuous time process model is:
  - $\dot{x} = f(x, u, n)$
  - $n_t \sim N(0, Q_t)$ is Gaussian white noise
- The observation model is:
  - $z = h(x, v)$
  - $v_t \sim N(0, R_t)$ is Gaussian white noise

Prediction

- Process model is nonlinear
- Need to convert the continuous dynamics to a discrete time system
- Look over a finite time interval $\tau = [t', t)$, where $t - t' = \delta t$
  - $t' \rightarrow t - 1$, $\bar{t}$ is an infinitesimal step before t
- Options:
  - Integrate the process model over the time horizon τ
    - $x_{\bar{t}} = \Phi(\bar{t}; x_{t-1}, u, n)$
    - Difficult to do in general
  - Use numerical integration
    - One-step Euler integration

Prediction – Linearization

- Linearize the dynamics about $x = \mu_{t-1},\ u = u_t,\ n = 0$
  - $\dot{x} \approx f(\mu_{t-1}, u_t, 0) + \left.\frac{\partial f}{\partial x}\right|_{\mu_{t-1},u_t,0} (x - \mu_{t-1}) + \left.\frac{\partial f}{\partial u}\right|_{\mu_{t-1},u_t,0} (u - u_t) + \left.\frac{\partial f}{\partial n}\right|_{\mu_{t-1},u_t,0} (n - 0)$
- Let:
  - $A_t = \left.\frac{\partial f}{\partial x}\right|_{\mu_{t-1},u_t,0}$
  - $B_t = \left.\frac{\partial f}{\partial u}\right|_{\mu_{t-1},u_t,0}$
  - $U_t = \left.\frac{\partial f}{\partial n}\right|_{\mu_{t-1},u_t,0}$
- Linear dynamics:
  - $\dot{x} \approx f(\mu_{t-1}, u_t, 0) + A_t(x - \mu_{t-1}) + B_t(u - u_t) + U_t(n - 0)$



Prediction – Discrete Time

- One-step Euler integration
  - $x_{\bar{t}} \approx x_{t-1} + f(x_{t-1}, u_t, n_t)\,\delta t$
  - $\approx x_{t-1} + \delta t\, f(\mu_{t-1}, u_t, 0) + \delta t\, A_t\,(x_{t-1} - \mu_{t-1}) + \delta t\, B_t(u_t - u_t)$
    $+ \delta t\, U_t(n_t - 0)$
  - $\approx x_{t-1} + \delta t\, f(\mu_{t-1}, u_t, 0) + \delta t\, A_t\,(x_{t-1} - \mu_{t-1}) + \delta t\, U_t(n_t - 0)$
  - $\approx (I + \delta t\, A_t)\, x_{t-1} + \delta t\, U_t\, n_t + \delta t(f(\mu_{t-1}, u_t, 0) - A_t\, \mu_{t-1})$
  - $\approx F_t\, x_{t-1} + V_t\, n_t + \delta t(f(\mu_{t-1}, u_t, 0) - A_t\, \mu_{t-1})$
- Prediction:
  - $\bar{\mu}_t = \mu_{t-1} + \delta t\, f(\mu_{t-1}, u_t, 0)$
  - $\bar{\Sigma}_t = F_t\, \Sigma_{t-1}\, F_t^T + V_t\, Q_t\, V_t^T$

Update – Linearization

- Linearize the observation model about $x = \bar{\mu}_t,\ v = 0$
  - $g(x, v) \approx g(\bar{\mu}_t, 0) + \left.\frac{\partial g}{\partial x}\right|_{\bar{\mu}_t, 0} (x - \bar{\mu}_t) + \left.\frac{\partial g}{\partial v}\right|_{\bar{\mu}_t, 0} (v - 0)$
- Let:
  - $C_t = \left.\frac{\partial g}{\partial x}\right|_{\bar{\mu}_t, 0}$
  - $W_t = \left.\frac{\partial g}{\partial v}\right|_{\bar{\mu}_t, 0}$
- Linear observation model:
  - $z_t = g(x_t, v_t) \approx g(\bar{\mu}_t, 0) + C_t\,(x_t - \bar{\mu}_t) + W_t\, v_t$

Update

- Follow the same derivation as the Kalman Filter
- $\begin{bmatrix} x_t \\ z_t \end{bmatrix} = \begin{bmatrix} I & 0 \\ C_t & W_t \end{bmatrix} \begin{bmatrix} x_{\bar{t}} \\ v_t \end{bmatrix} + \begin{bmatrix} 0 \\ g(\bar{\mu}_t, 0) - C_t \bar{\mu}_t \end{bmatrix}$
- Mean:
  - $E[X_t] = E[\bar{X}_t] = \bar{\mu}_t$
  - $E[Z_t] = E[C_t\, \bar{X}_t + W_t\, V_t + g(\bar{\mu}_t, 0) - C_t \bar{\mu}_t]$
  - $\qquad = C_t\, \bar{\mu}_t + g(\bar{\mu}_t, 0) - C_t \bar{\mu}_t$
  - $\qquad = g(\bar{\mu}_t, 0)$

Update

- Follow the same derivation as the Kalman Filter
- $\begin{bmatrix} x_t \\ z_t \end{bmatrix} = \begin{bmatrix} I & 0 \\ C_t & W_t \end{bmatrix} \begin{bmatrix} x_{\bar{t}} \\ v_t \end{bmatrix} + \begin{bmatrix} 0 \\ g(\bar{\mu}_t, 0) - C_t \bar{\mu}_t \end{bmatrix}$
- Covariance:
  - $\Sigma = \begin{bmatrix} I & 0 \\ C_t & W_t \end{bmatrix} \begin{bmatrix} \bar{\Sigma}_t & 0 \\ 0 & R_t \end{bmatrix} \begin{bmatrix} I & C_t^T \\ 0 & W_t^T \end{bmatrix}$
  - $= \begin{bmatrix} \bar{\Sigma}_t & \bar{\Sigma}_t C_t^T \\ C_t \bar{\Sigma}_t & C_t \bar{\Sigma}_t C_t^T + W_t R_t W_t^T \end{bmatrix}$

Update

- Recall that for a multivariate Guassian $Y = \begin{bmatrix} X \\ Z \end{bmatrix}$ with mean $\mu = \begin{bmatrix} \mu_X \\ \mu_Z \end{bmatrix}$ and covariance $\Sigma = \begin{bmatrix} \Sigma_{XX} & \Sigma_{XZ} \\ \Sigma_{ZX} & \Sigma_{ZZ} \end{bmatrix}$
- The conditional density $f_{X|Z}(x \mid Z = z)$ is Gaussian with
  - $\mu_{X|Z} = \mu_X + \Sigma_{XZ}\Sigma_{ZZ}^{-1}(z - \mu_Z)$
  - $\Sigma_{X|Z} = \Sigma_{XX} - \Sigma_{XZ}\,\Sigma_{ZZ}^{-1}\,\Sigma_{ZX}$
- Result:
  - $\mu_t = \bar{\mu}_t + K_t\,(z_t - g(\bar{\mu}_t, 0))$
  - $\Sigma_t = \bar{\Sigma}_t - K_t\,C_t\,\bar{\Sigma}_t$
  - $K_t = \bar{\Sigma}_t\,C_t^T\,(C_t\,\bar{\Sigma}_t\,C_t^T + W_t\,R\,W_t^T)^{-1}$

Extended Kalman Filter

- Prediction step:
  - $\bar{\mu}_t = \mu_{t-1} + \delta t\, f(\mu_{t-1}, u_t, 0)$
  - $\bar{\Sigma}_t = F_t\, \Sigma_{t-1}\, F_t^T + V_t\, Q_t\, V_t^T$

$$\left.\begin{aligned} &\dot{x} = f(x, u, n) \\ &n_t \sim N(0, Q_t) \end{aligned}\right\} \text{Assumptions}$$

$$\left.\begin{aligned} &A_t = \left.\frac{\partial f}{\partial x}\right|_{\mu_{t-1}, u_t, 0} \\ &U_t = \left.\frac{\partial f}{\partial n}\right|_{\mu_{t-1}, u_t, 0} \end{aligned}\right\} \text{Linearization}$$

$$\left.\begin{aligned} &F_t = I + \delta t\, A_t \\ &V_t = \delta t\, U_t \end{aligned}\right\} \text{Discretization}$$

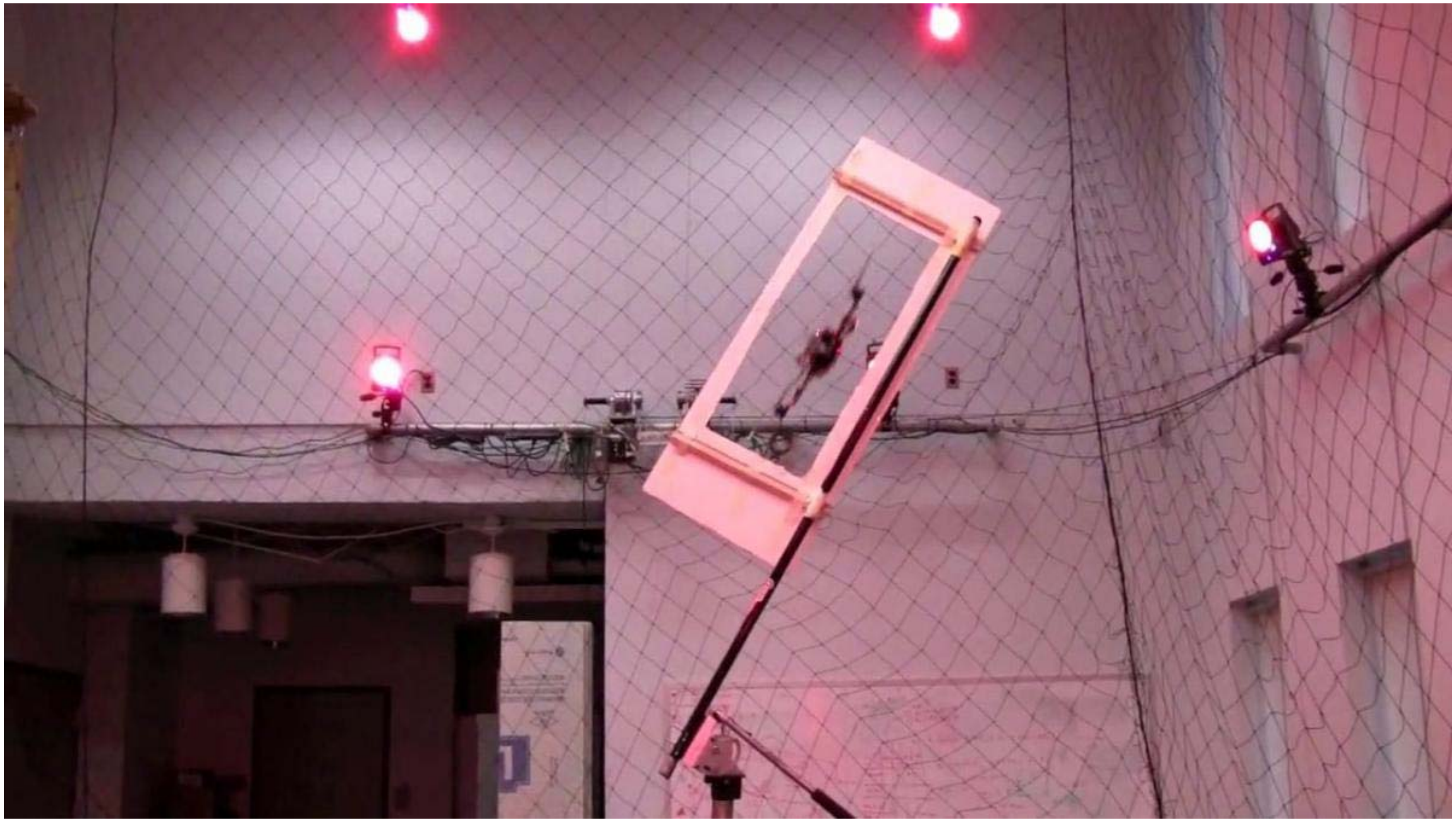
- Update step:
  - $\mu_t = \bar{\mu}_t + K_t\, (z_t - g(\bar{\mu}_t, 0))$
  - $\Sigma_t = \bar{\Sigma}_t - K_t\, C_t\, \bar{\Sigma}_t$
  - $K_t = \bar{\Sigma}_t\, C_t^T\, (C_t\, \bar{\Sigma}_t\, C_t^T + W_t\, R\, W_t^T)^{-1}$

$$\left.\begin{aligned} &z_t = g(x_t, v_t) \\ &v_t \sim N(0, R_t) \end{aligned}\right\} \text{Assumptions}$$

$$\left.\begin{aligned} &C_t = \left.\frac{\partial g}{\partial x}\right|_{\bar{\mu}_t, 0} \\ &W_t = \left.\frac{\partial g}{\partial v}\right|_{\bar{\mu}_t, 0} \end{aligned}\right\} \text{Linearization}$$

Example Problem

Quadrotor with a Good Velocity Sensor



State

- Can accurately estimate the commanded linear velocity using the motion tracking system and the angular velocity using a gyroscope
- $\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \mathbf{b}_g \end{bmatrix} = \begin{bmatrix} \text{position} \\ \text{orientation} \\ \text{gyroscope bias} \end{bmatrix} \in \mathbf{R}^9$
- Use Z-X-Y Euler angle parameterization of $SO(3)$ for orientation
  - $\mathbf{q} = [\phi, \theta, \psi]^T = [\text{roll}, \text{pitch}, \text{yaw}]^T$
  - $R = \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\psi c\theta s\phi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix}$

Process Model

- **Assumption:** the motion tracking system gives a noisy estimate of the linear velocity
  - $\mathbf{v}_m = \dot{\mathbf{p}} + \mathbf{n}_v$
- **Assumption:** the gyroscope gives a noisy estimate of the angular velocity
  - $\omega_m = \omega + \mathbf{b}_g + \mathbf{n}_g$
- **Assumption:** the drift in the gyroscope bias is described by a Gaussian, white noise process
  - $\dot{\mathbf{b}}_g = \mathbf{n}_{bg}$
  - $\mathbf{n}_{bg} \sim N(0, Q_g)$

Process Model

- ω_m is in the body frame, $\mathbf{q}$ is in the world frame
- **Recall:** the angular velocity in the body frame is given by
- $$\omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} c\theta & 0 & -c\phi s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = G(\mathbf{q})\dot{\mathbf{q}}$$
- Process model:
- $$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{v}_m - \mathbf{n}_v \\ G(\mathbf{x}_2)^{-1}(\omega_m - \mathbf{x}_3 - \mathbf{n}_g) \\ \mathbf{n}_{bg} \end{bmatrix}$$

Observation Model

- Use a camera to measure the pose of the robot
- Use theory of projective geometry
  - Covered earlier in the semester
- Can estimate the position and orientation of the robot using a minimum of 4 features on the ground plane, e.g., AR Tags
  - Can recover $\mathbf{q}$ from the rotation matrix R
- $$\begin{aligned} \mathbf{z} &= \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \end{bmatrix} + \mathbf{v} \\ &= \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \mathbf{b}_g \end{bmatrix} + \mathbf{v} \\ &= C\,\mathbf{x} + \mathbf{v} \end{aligned}$$

Example Problem

Quadrotor with a Good Acceleration Sensor

State

- Can accurately estimate the commanded linear acceleration and angular velocity using the IMU
- $\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \\ \mathbf{x}_5 \end{bmatrix} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \dot{\mathbf{p}} \\ \mathbf{b}_g \\ \mathbf{b}_{\boldsymbol{a}} \end{bmatrix} = \begin{bmatrix} \text{position} \\ \text{orientation} \\ \text{linear velocity} \\ \text{gyroscope bias} \\ \text{accelerometer bias} \end{bmatrix} \in \mathbf{R}^{15}$
- Use Z-X-Y Euler angle parameterization of $SO(3)$ for orientation
  - $\mathbf{q} = [\phi, \theta, \psi]^T = [\text{roll}, \text{pitch}, \text{yaw}]^T$
  - $R = \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\psi c\theta s\phi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix}$

Process Model – Gyroscope

- **Assumption:** the gyroscope gives a noisy estimate of the angular velocity
  - $\omega_m = \omega + \mathbf{b}_g + \mathbf{n}_g$
- **Assumption:** the drift in the gyroscope bias is described by a Gaussian, white noise process
  - $\dot{\mathbf{b}}_g = \mathbf{n}_{bg}$
  - $\mathbf{n}_{bg} \sim N(0, Q_g)$

Process Model – Accelerometer

- **Assumption:** the accelerometer gives a noisy estimate of the linear acceleration
  - $\mathbf{a}_m = R(\mathbf{q})^T(\ddot{\mathbf{p}} - \mathbf{g}) + \mathbf{b}_a + \mathbf{n}_a$
- **Assumption:** the drift in the accelerometer bias is described by a Gaussian, white noise process
  - $\dot{\mathbf{b}}_a = \mathbf{n}_{ba}$
  - $\mathbf{n}_{ba} \sim N(0, Q_a)$

Process Model

- Process model:
- $\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{x}_3 \\ G(\mathbf{x}_2)^{-1}\left(\omega_m - \mathbf{x}_4 - \mathbf{n}_g\right) \\ \mathbf{g} + R(\mathbf{x}_2)(\mathbf{a}_m - \mathbf{x}_5 - \mathbf{n}_a) \\ \mathbf{n}_{bg} \\ \mathbf{n}_{ba} \end{bmatrix}$

Observation Model

- Use a camera to measure:
  - The pose of the robot (using AR Tags)
  - The linear velocity (using optical flow)
- $\mathbf{z} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \dot{\mathbf{p}} \end{bmatrix} + \mathbf{v}$

$$= \begin{bmatrix} I & 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \dot{\mathbf{p}} \\ \mathbf{b}_g \\ \mathbf{b}_a \end{bmatrix} + \mathbf{v}$$

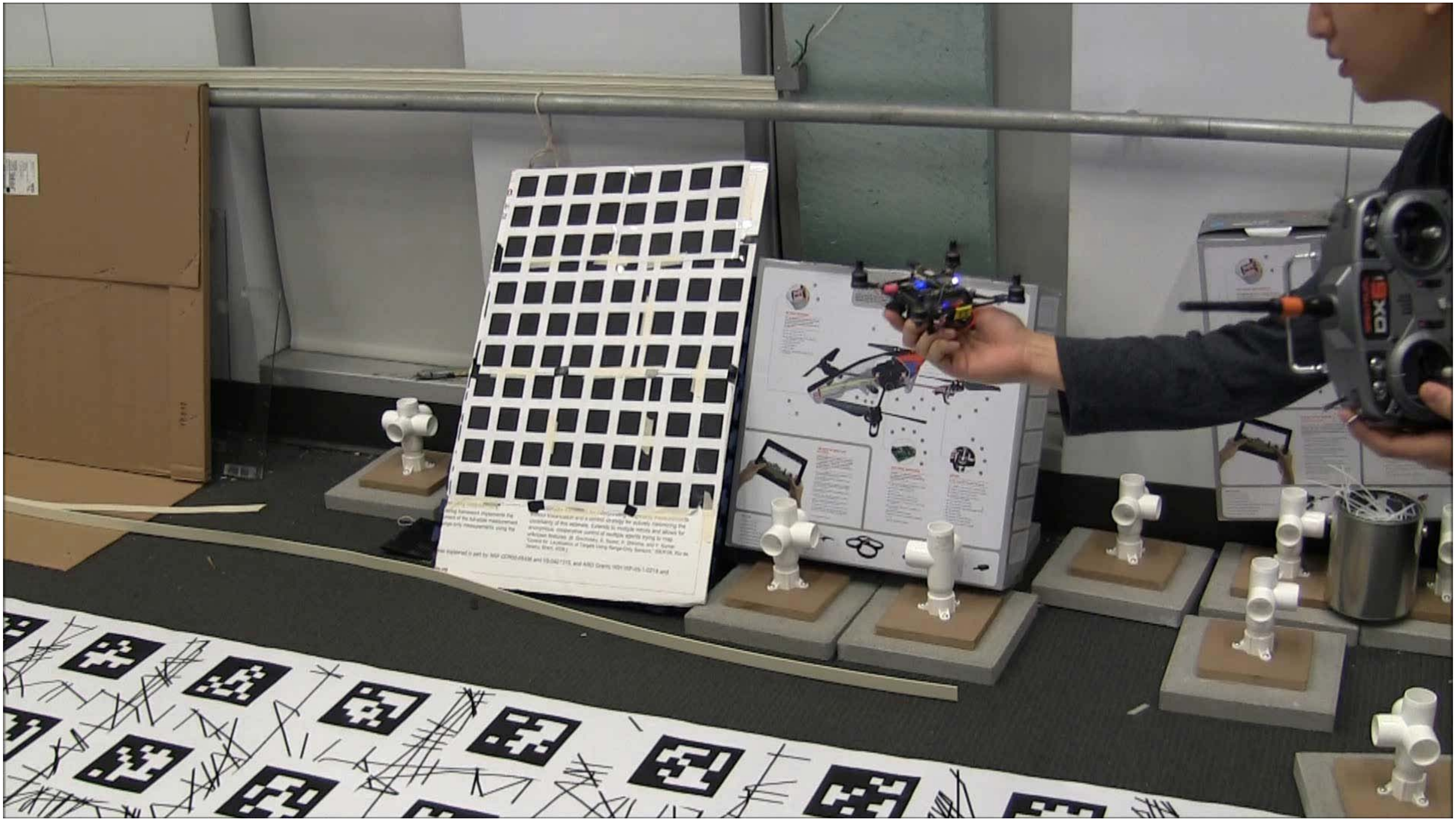
$$= C\,\mathbf{x} + \mathbf{v}$$

Observation Model

- Use a camera to measure:
  - The pose of the robot (using AR Tags)
- $\mathbf{z} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \dot{\mathbf{p}} \end{bmatrix} + \mathbf{v}$

$$= \begin{bmatrix} I & 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 \\ \cancel{0} & \cancel{0} & \cancel{I} & \cancel{0} & \cancel{0} \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \dot{\mathbf{p}} \\ \mathbf{b}_g \\ \mathbf{b}_{\boldsymbol{a}} \end{bmatrix} + \mathbf{v}$$

$$= C\,\mathbf{x} + \mathbf{v}$$



Recap

Bayes' Filter

State

Control input

- **Prior:** $p(x_0)$
- **Process model:** $f(x_t \mid x_{t-1}, u_t)$
- **Measurement model:** $g(z_t \mid x_t)$

Measurement

- **Prediction step:**
- $p(x_t \mid z_{1:t-1}, u_{1:t}) = \int f(x_t \mid x_{t-1}, u_t)\, p(x_{t-1} \mid z_{1:t-1}, u_{1:t-1})\, dx_{t-1}$
- **Update step:**
- $p(x_t \mid z_{1:t}, u_{1:t}) = \dfrac{g(z_t \mid x_t)\, p(x_t \mid z_{1:t-1}, u_{1:t})}{\int g(z_t \mid x_t')\, p(x_t' \mid z_{1:t-1}, u_{1:t})\, dx_t'}$

Assumptions

- The prior state of the robot is represented by a Gaussian distribution
  - $p(x_0) \sim N(\mu_0, \Sigma_0)$
- The continuous time process model is:
  - $\dot{x} = f(x, u, n)$
  - $n_t \sim N(0, Q_t)$ is Gaussian white noise
- The observation model is:
  - $z = h(x, v)$
  - $v_t \sim N(0, R_t)$ is Gaussian white noise

Extended Kalman Filter

- Prediction step:
  - $\bar{\mu}_t = \mu_{t-1} + \delta t\, f(\mu_{t-1}, u_t, 0)$
  - $\bar{\Sigma}_t = F_t\, \Sigma_{t-1}\, F_t^T + V_t\, Q_t\, V_t^T$

$$\left.\begin{aligned} &- \ \dot{x} = f(x, u, n) \\ &- \ n_t \sim N(0, Q_t) \end{aligned}\right\} \text{Assumptions}$$

$$\left.\begin{aligned} &- \ A_t = \left.\frac{\partial f}{\partial x}\right|_{\mu_{t-1}, u_t, 0} \\ &- \ U_t = \left.\frac{\partial f}{\partial n}\right|_{\mu_{t-1}, u_t, 0} \end{aligned}\right\} \text{Linearization}$$

$$\left.\begin{aligned} &- \ F_t = I + \delta t\, A_t \\ &- \ V_t = \delta t\, U_t \end{aligned}\right\} \text{Discretization}$$

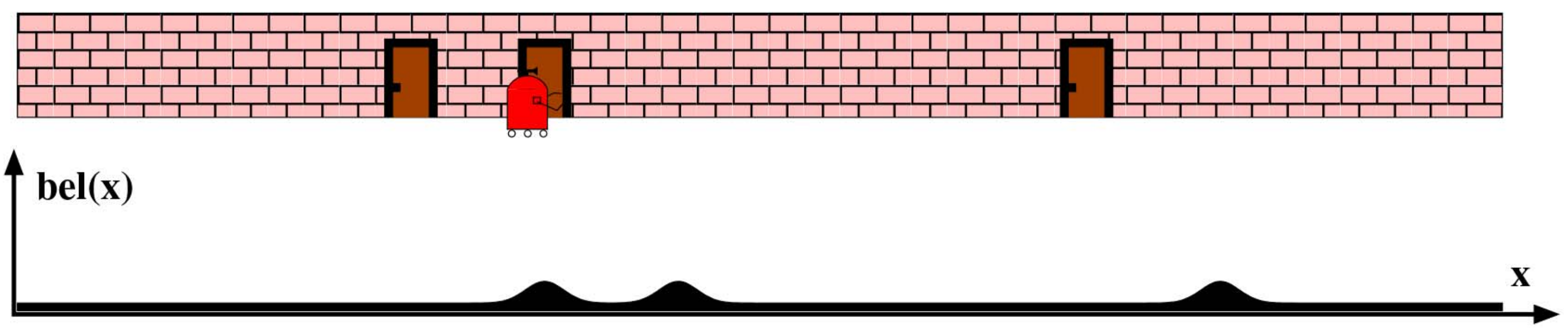
- Update step:
  - $\mu_t = \bar{\mu}_t + K_t\, (z_t - g(\bar{\mu}_t, 0))$
  - $\Sigma_t = \bar{\Sigma}_t - K_t\, C_t\, \bar{\Sigma}_t$
  - $K_t = \bar{\Sigma}_t\, C_t^T\, (C_t\, \bar{\Sigma}_t\, C_t^T + W_t\, R\, W_t^T)^{-1}$

$$\left.\begin{aligned} &- \ z_t = g(x_t, v_t) \\ &- \ v_t \sim N(0, R_t) \end{aligned}\right\} \text{Assumptions}$$

$$\left.\begin{aligned} &- \ C_t = \left.\frac{\partial g}{\partial x}\right|_{\bar{\mu}_t, 0} \\ &- \ W_t = \left.\frac{\partial g}{\partial v}\right|_{\bar{\mu}_t, 0} \end{aligned}\right\} \text{Linearization}$$

EKF Discussion

- **Advantages:**
  - Simple
  - Computationally efficient, even for high dimensional systems
  - Works with generic process and observation models
- **Disadvantages:**
  - Must calculate the Jacobian of the process and observation models
  - Unimodal distribution
    - Cannot handle multiple hypotheses



Other Applications

- Can use Bayesian filtering methods in many domains:
  - Pose estimation
  - Parameter estimation
  - Map building
  - Simultaneous localization and mapping (SLAM)
  - Feature tracking
  - Target tracking

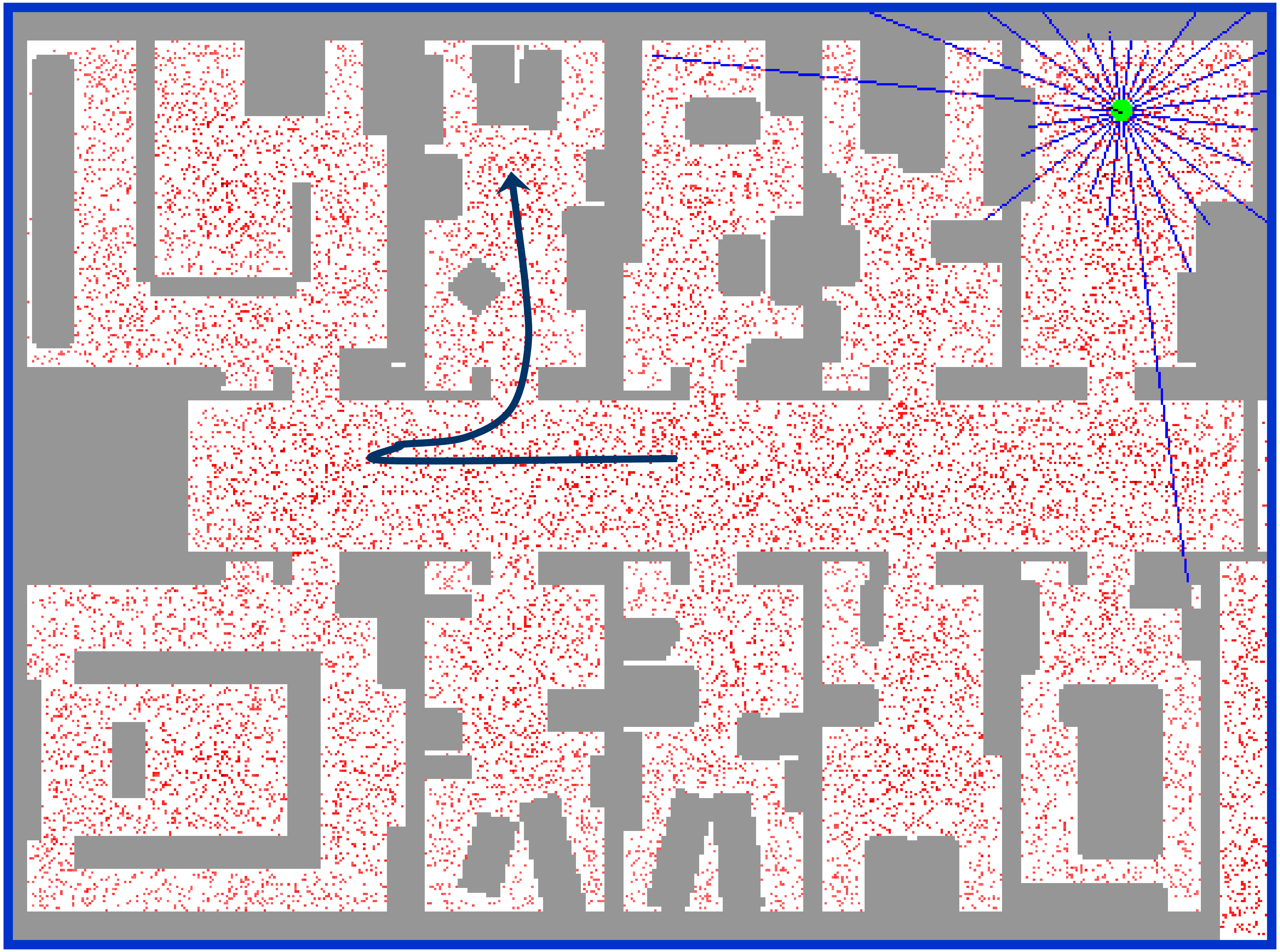
Particle Filter

Probabilistic Robotics

Bayes Filter Implementations

Particle filters

Sample-based Localization (sonar)



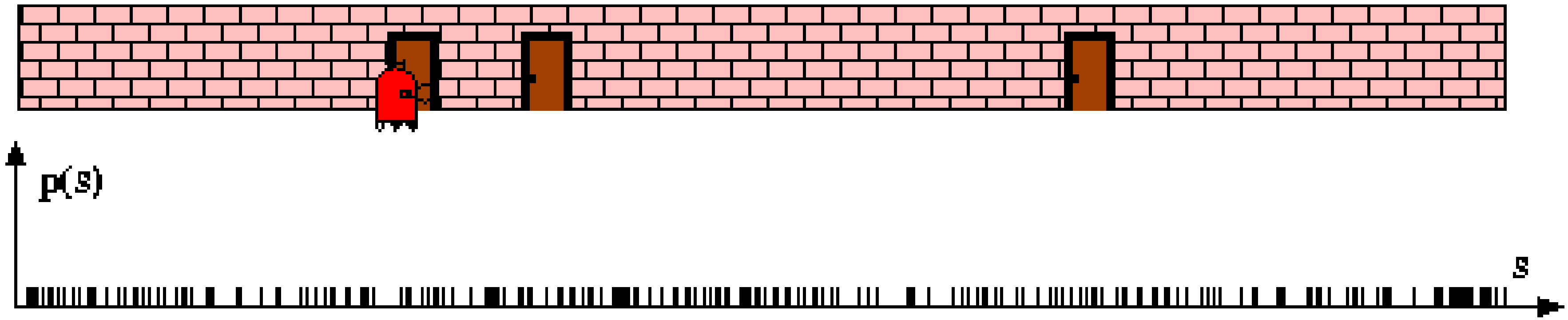
Particle Filters

- Represent belief by random samples
- Estimation of non-Gaussian, nonlinear processes

- Monte Carlo filter, Survival of the fittest, Condensation, Bootstrap filter, Particle filter

- Filtering: [Rubin, 88], [Gordon et al., 93], [Kitagawa 96]
- Computer vision: [Isard and Blake 96, 98]
- Dynamic Bayesian Networks: [Kanazawa et al., 95]

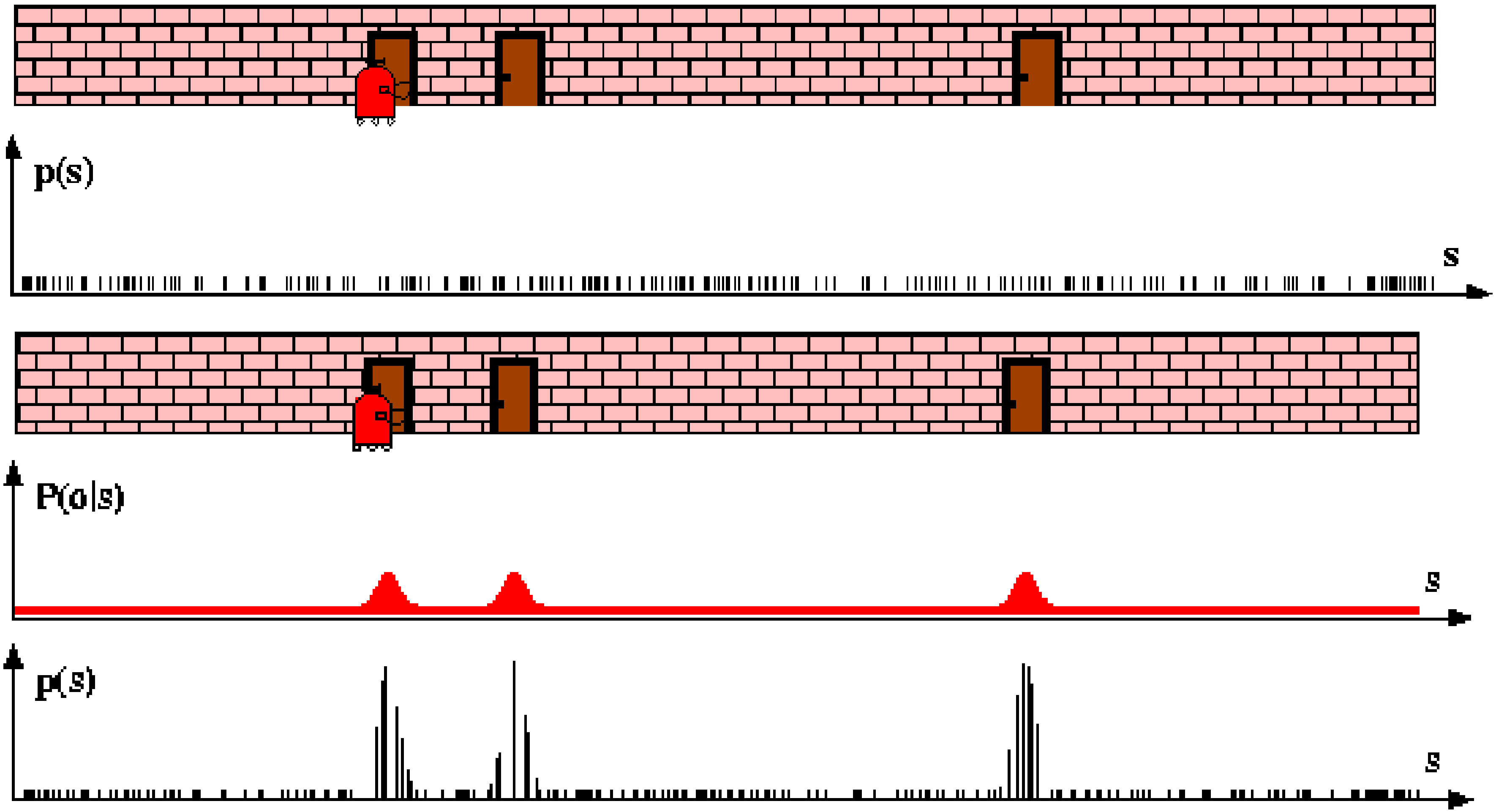
Particle Filters



Sensor Information: Importance Sampling

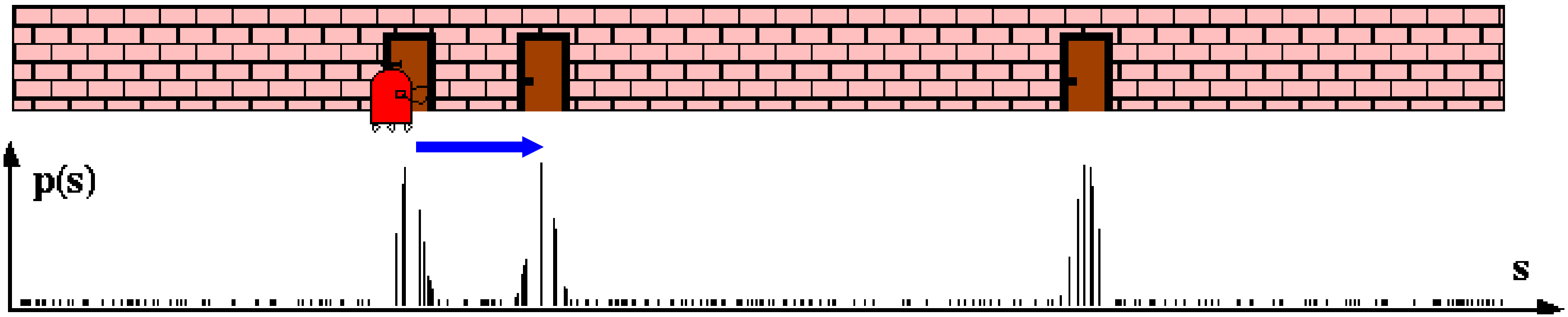
$$Bel(x) \leftarrow \alpha\ p(z \mid x)\ Bel^{-}(x)$$

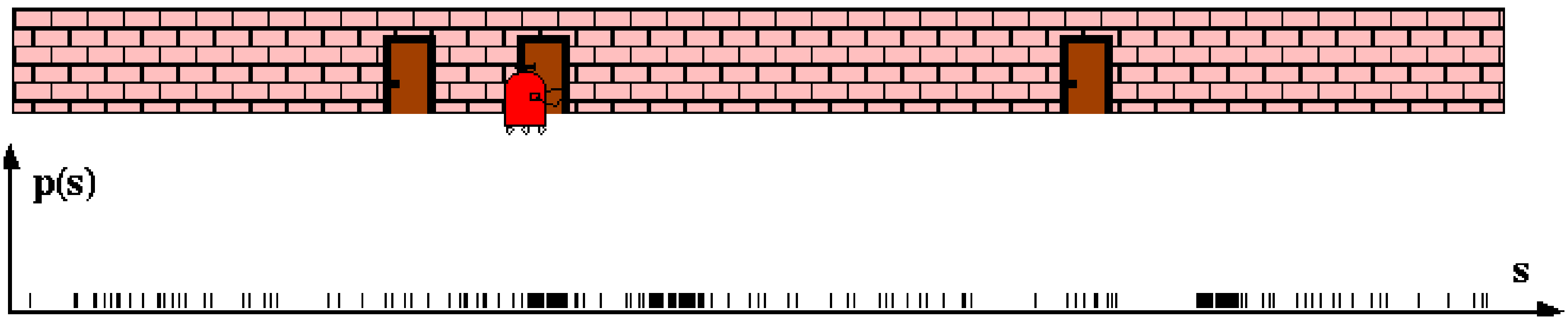
$$w \leftarrow \frac{\alpha\ p(z \mid x)\ Bel^{-}(x)}{Bel^{-}(x)} = \alpha\ p(z \mid x)$$



Robot Motion

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

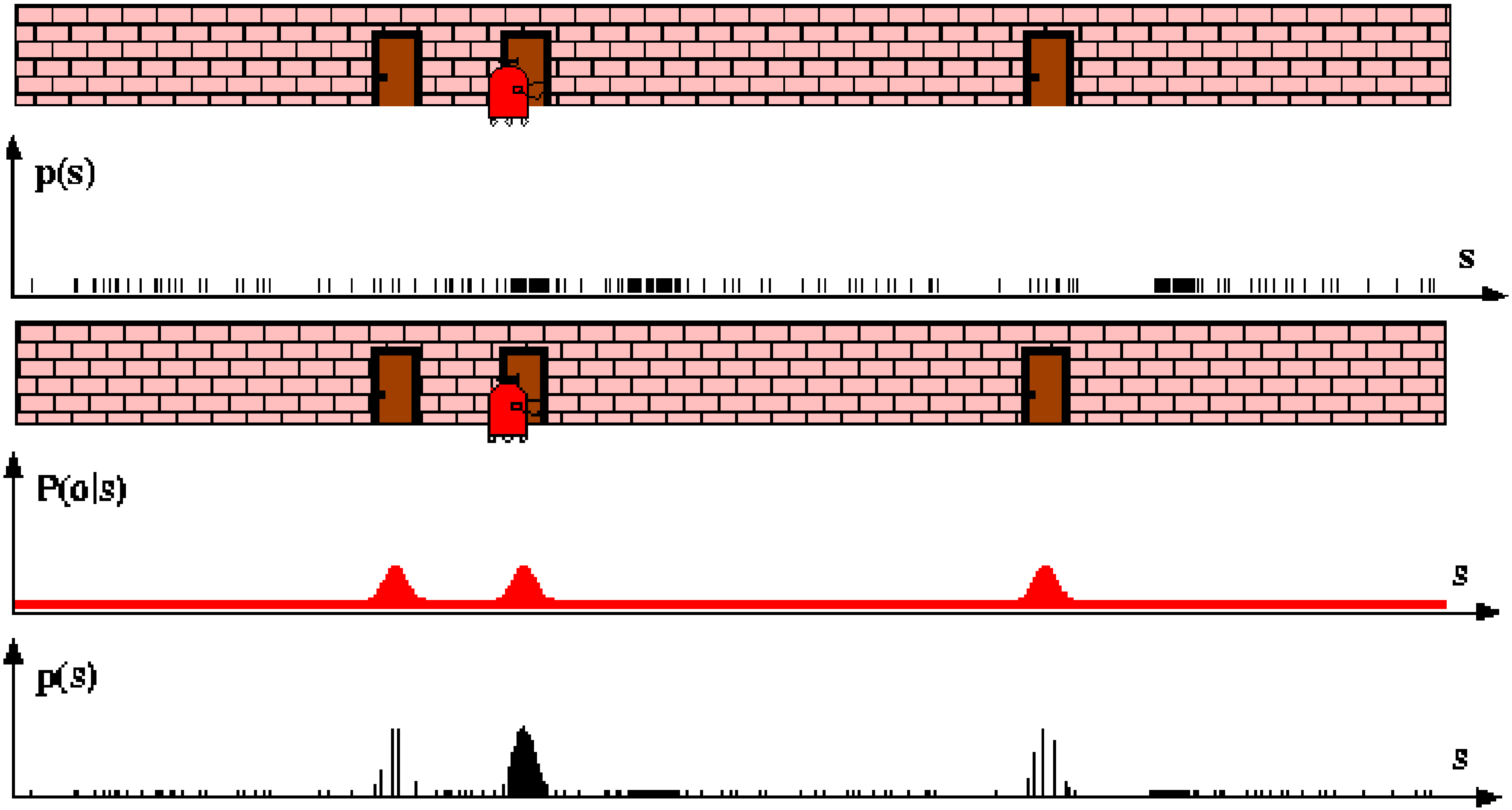




Sensor Information: Importance Sampling

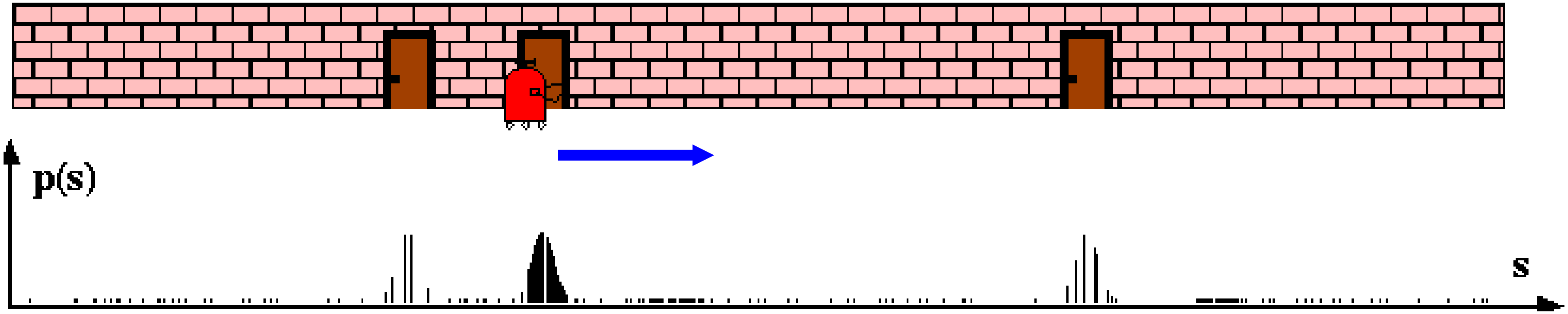
$$Bel(x) \leftarrow \alpha\ p(z \mid x)\ Bel^{-}(x)$$

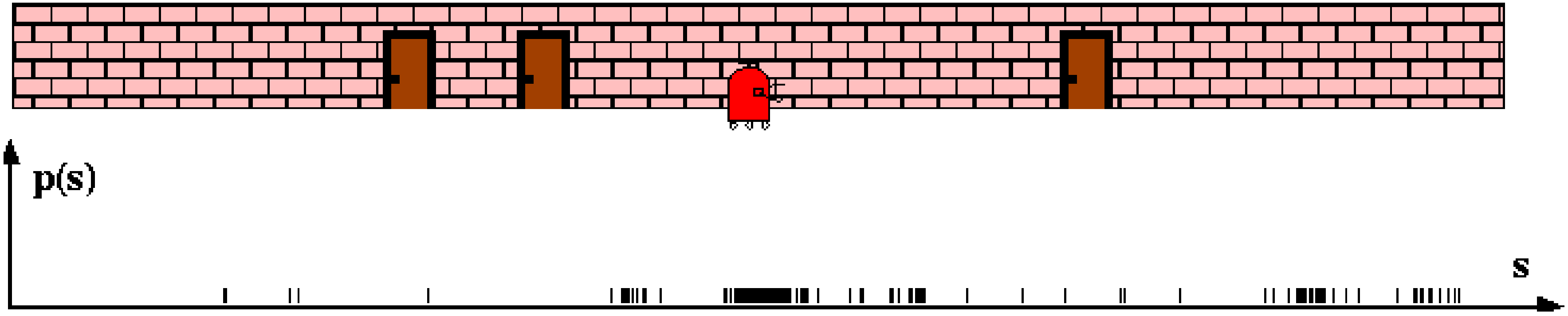
$$w \leftarrow \frac{\alpha\ p(z \mid x)\ Bel^{-}(x)}{Bel^{-}(x)} = \alpha\ p(z \mid x)$$



Robot Motion

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





Particle Filter Algorithm

1. Algorithm **particle_filter**(S_{t-1}, u_{t-1} z_t):
2. $S_t = \emptyset, \quad \eta = 0$
3. **For** $i = 1 \dots n$ ***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** $i = 1 \dots n$
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:

$$\begin{aligned} w^i_t &= \frac{\text{target distribution}}{\text{proposal distribution}} \\ &= \frac{\eta\; p(z_t \mid x_t)\; p(x_t \mid x_{t-1}, u_{t-1})\; Bel\,(x_{t-1})}{p(x_t \mid x_{t-1}, u_{t-1})\; Bel\,(x_{t-1})} \\ &\propto p(z_t \mid x_t) \end{aligned}$$

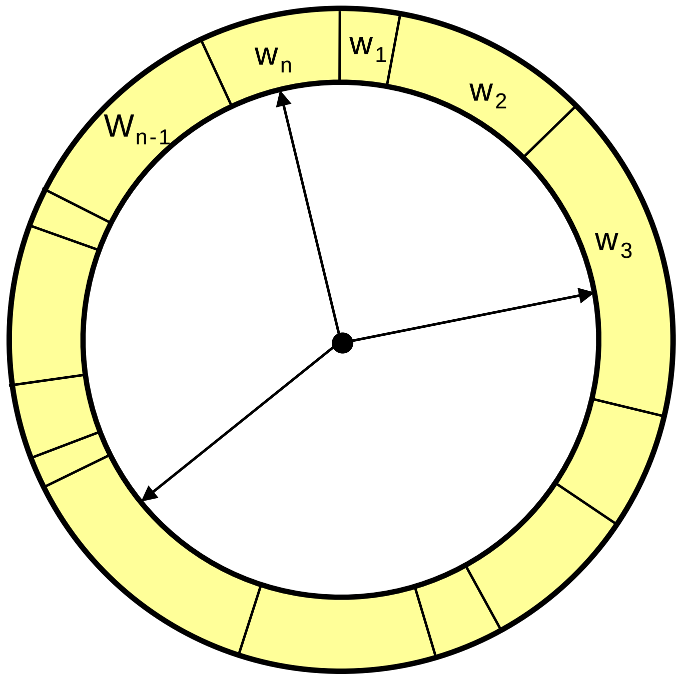
Resampling

- **Given**: Set S of weighted samples.

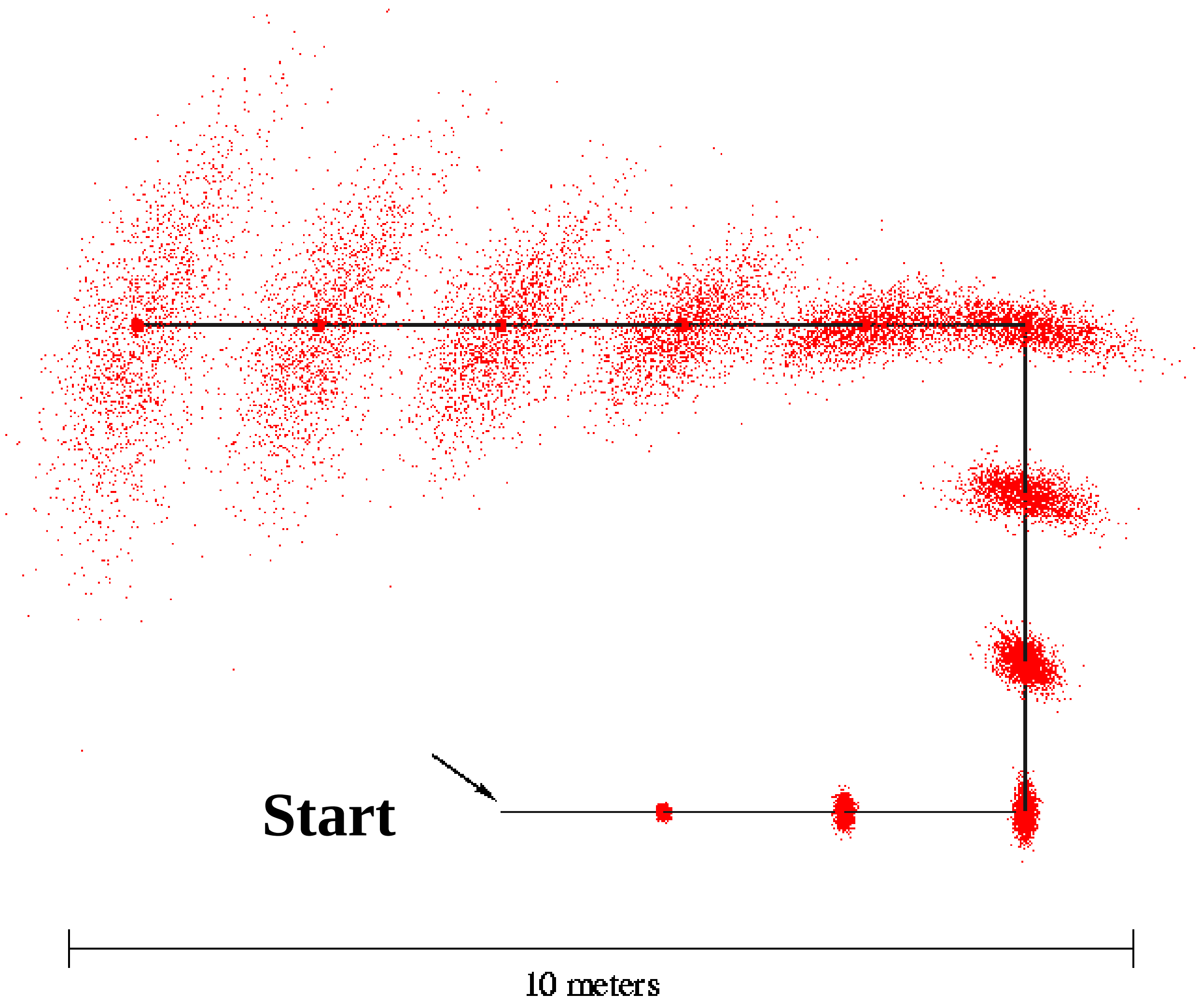
- **Wanted** : Random sample, where the probability of drawing x_i is given by w_i.

- Typically done n times with replacement to generate new sample set S'.

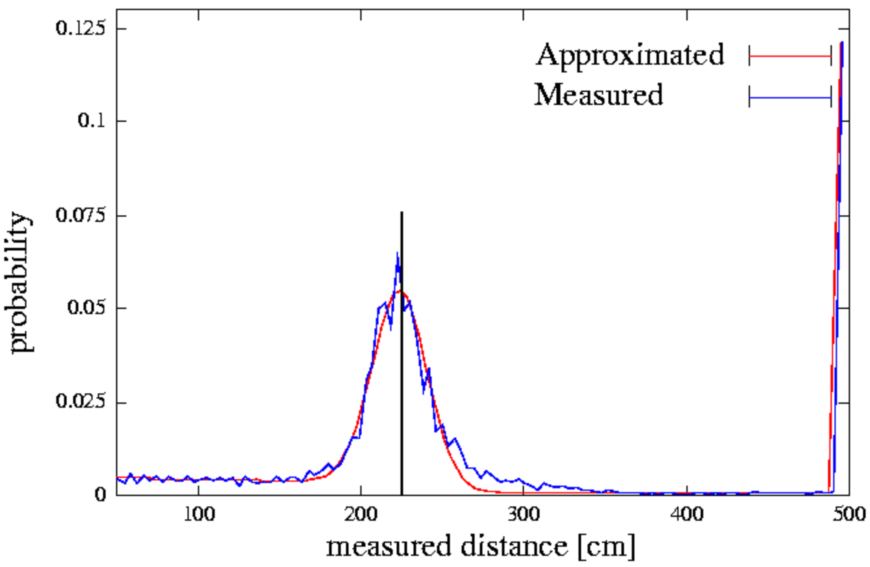
Resampling



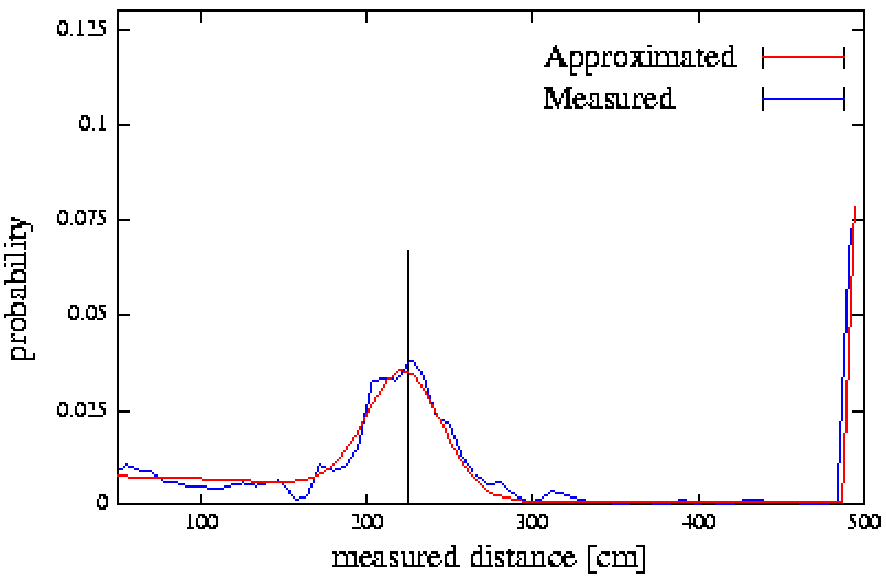
Motion Model Reminder



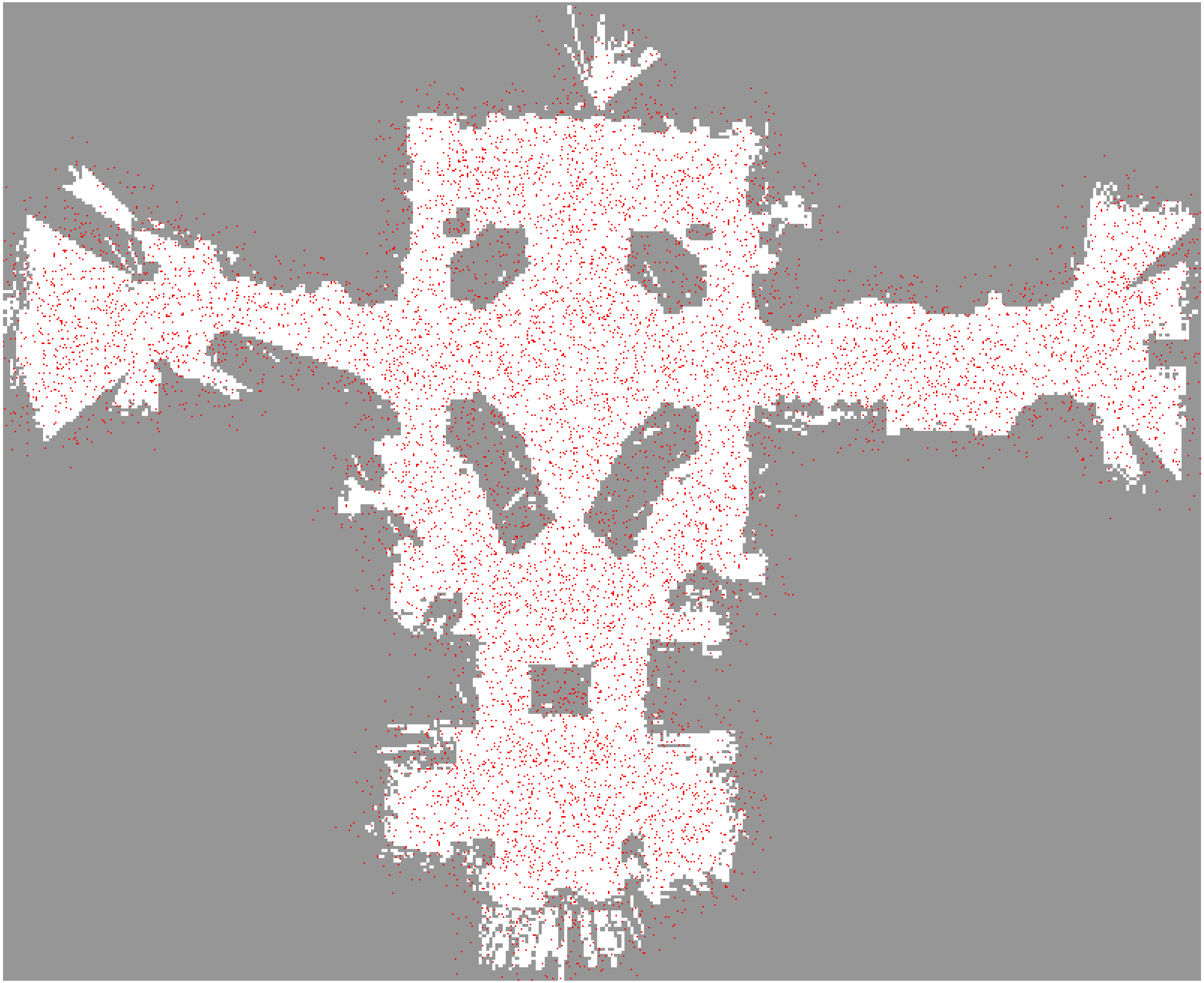
Proximity Sensor Model Reminder

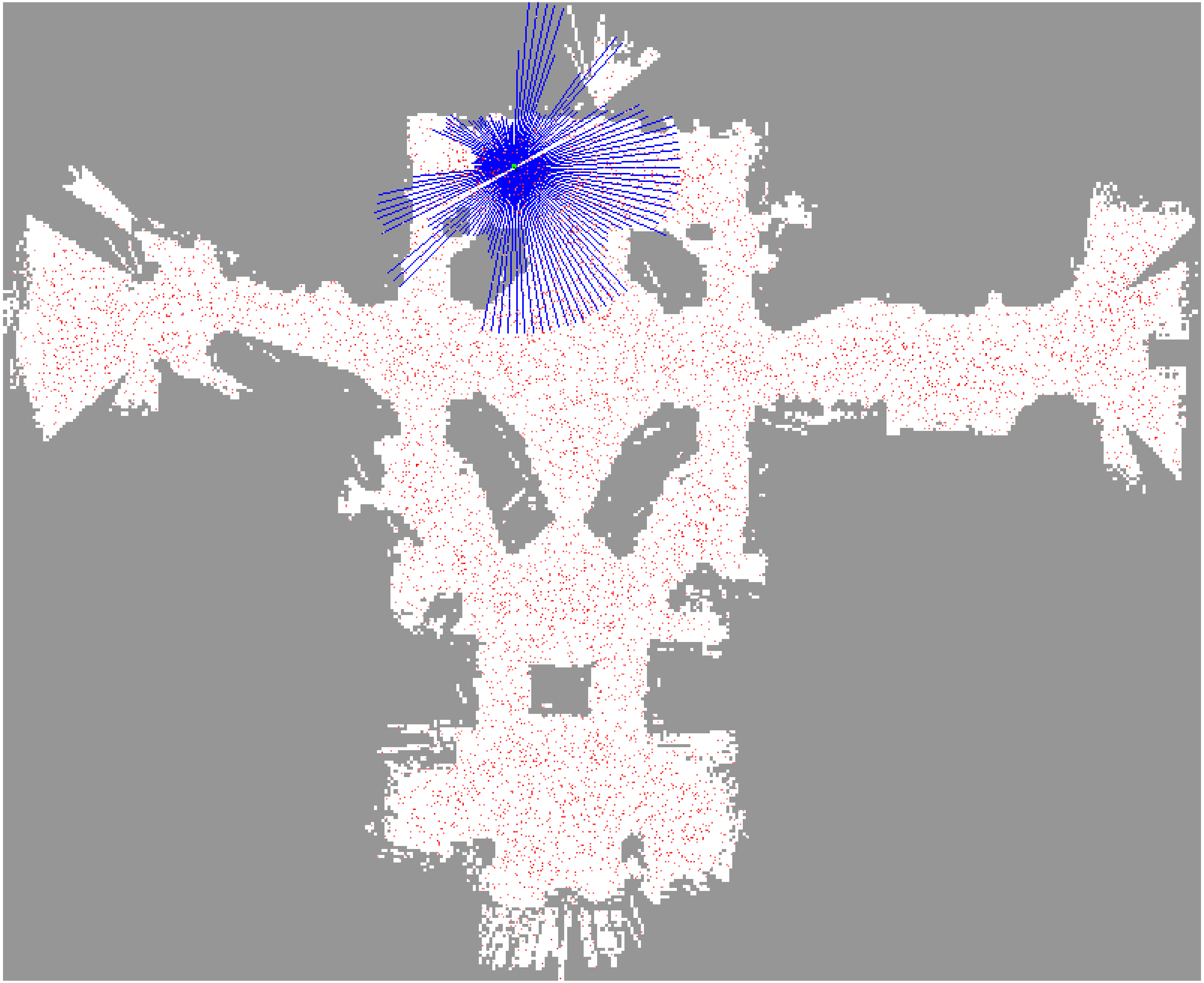


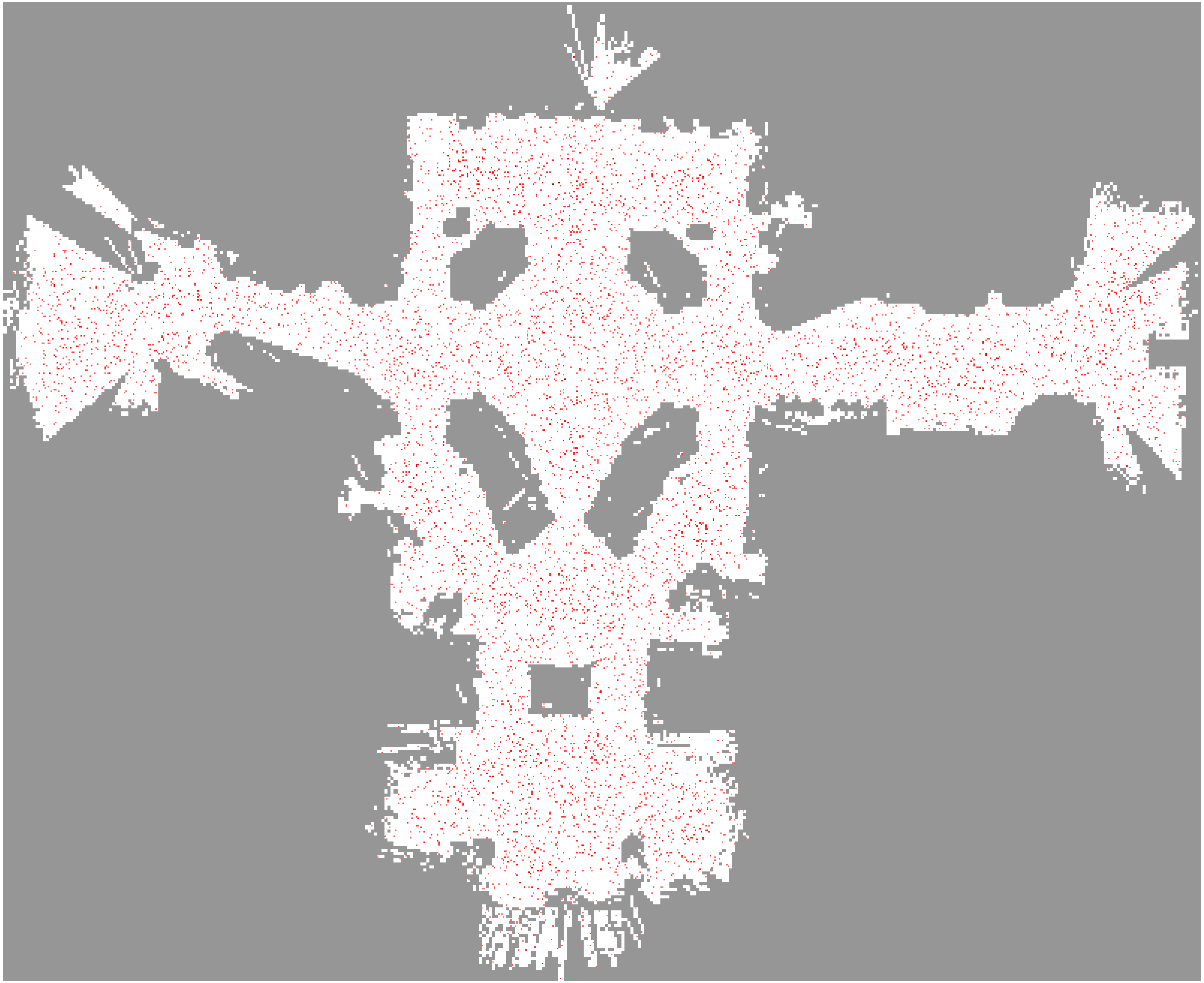
Laser sensor

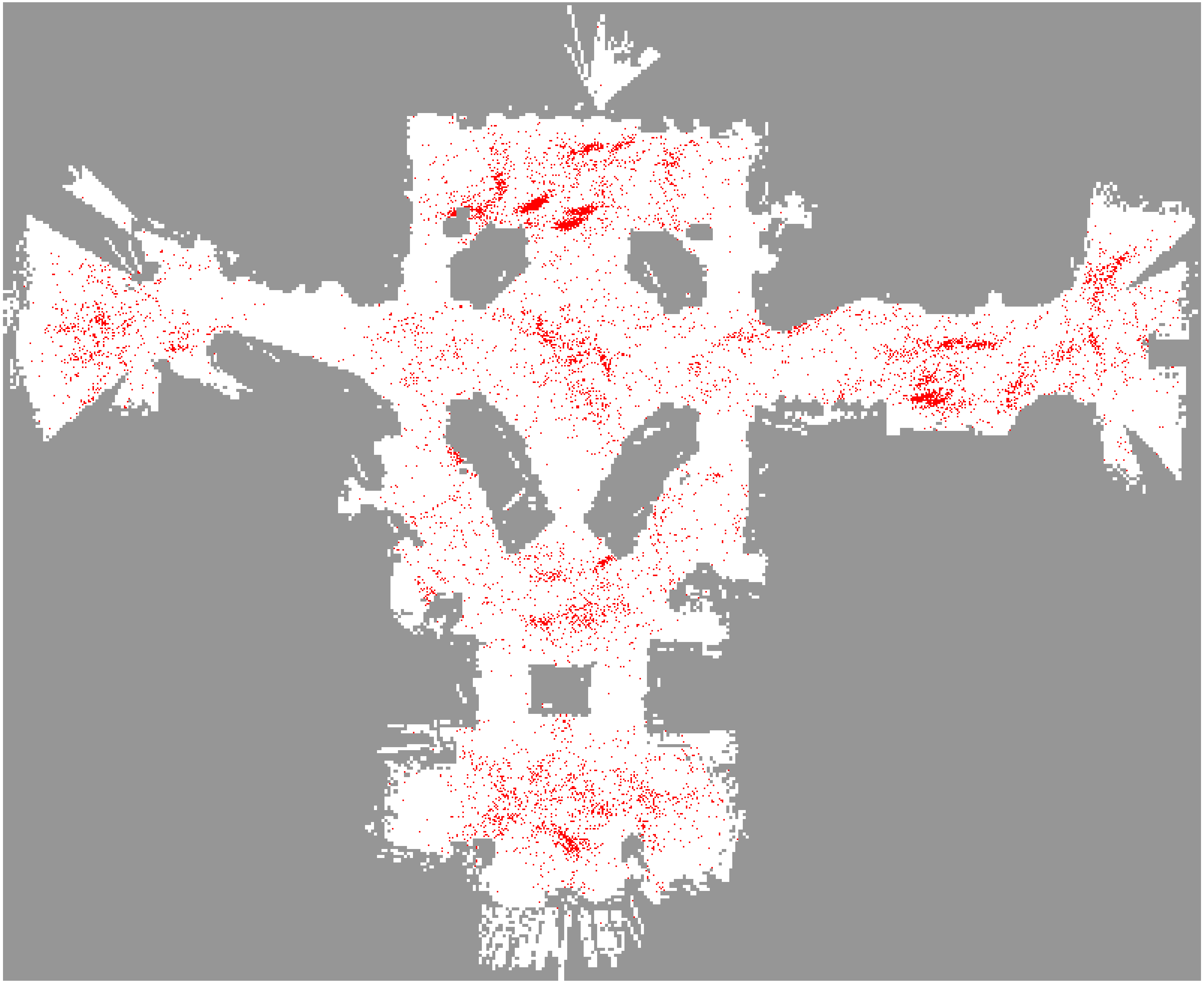


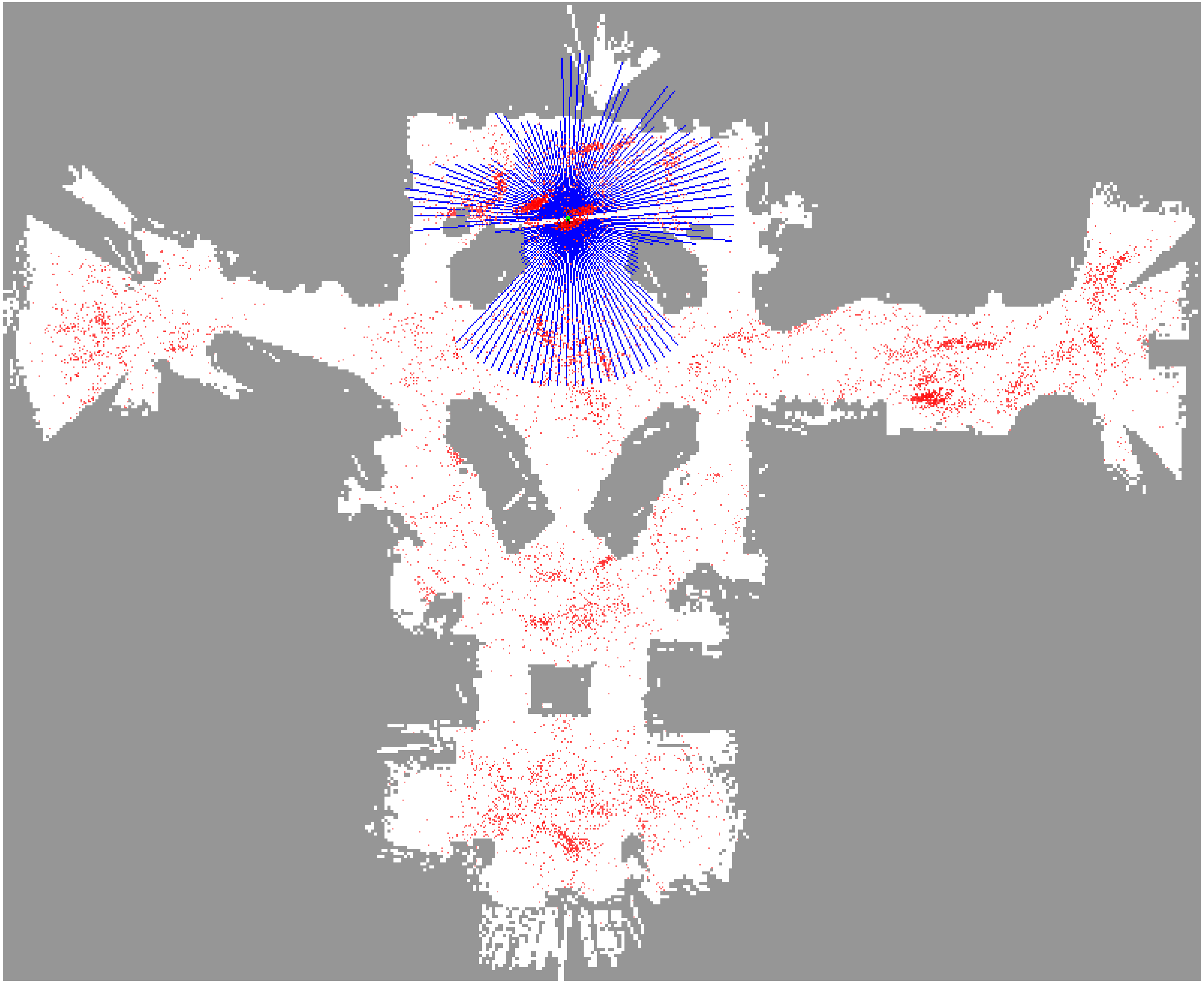
Sonar sensor

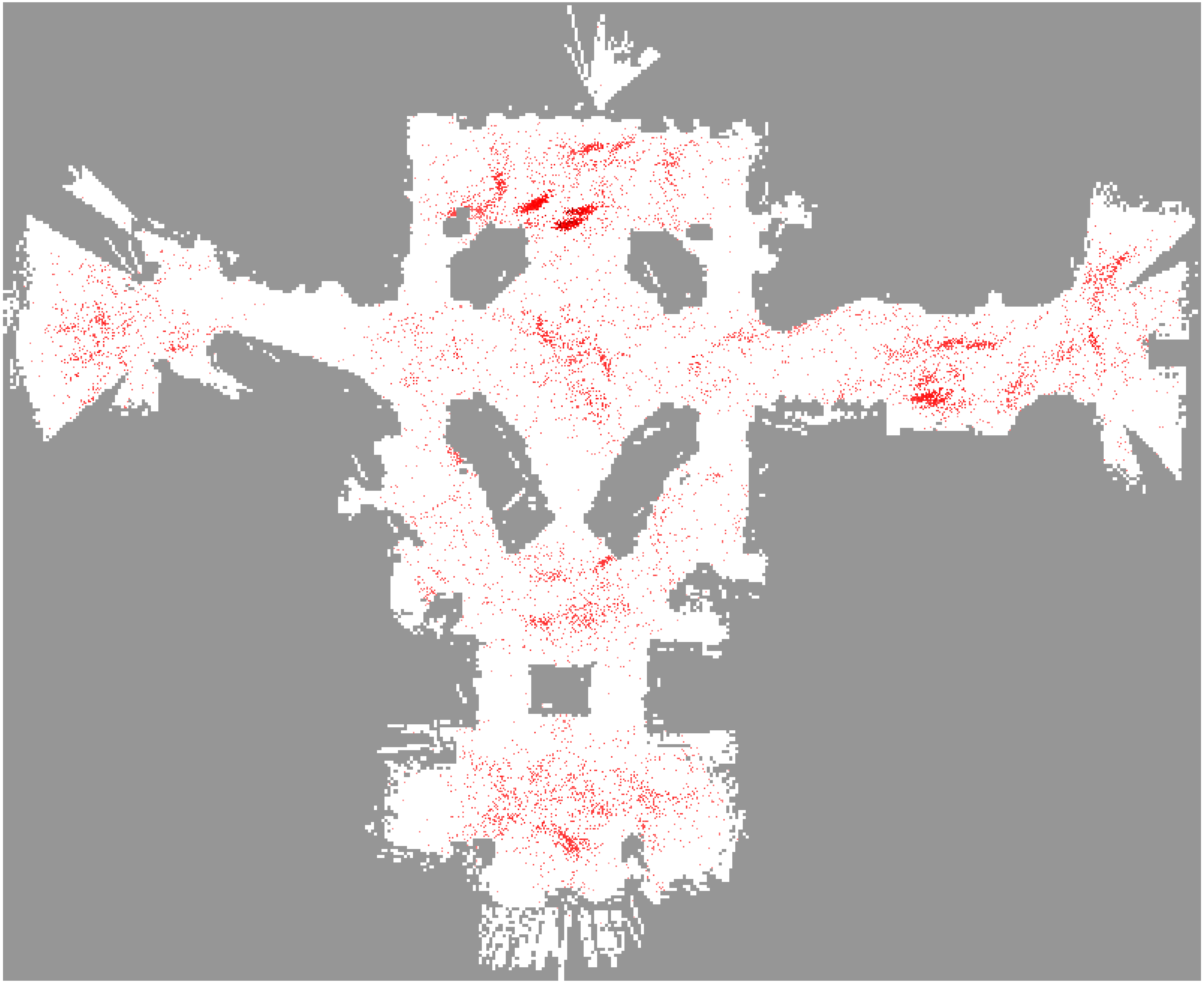


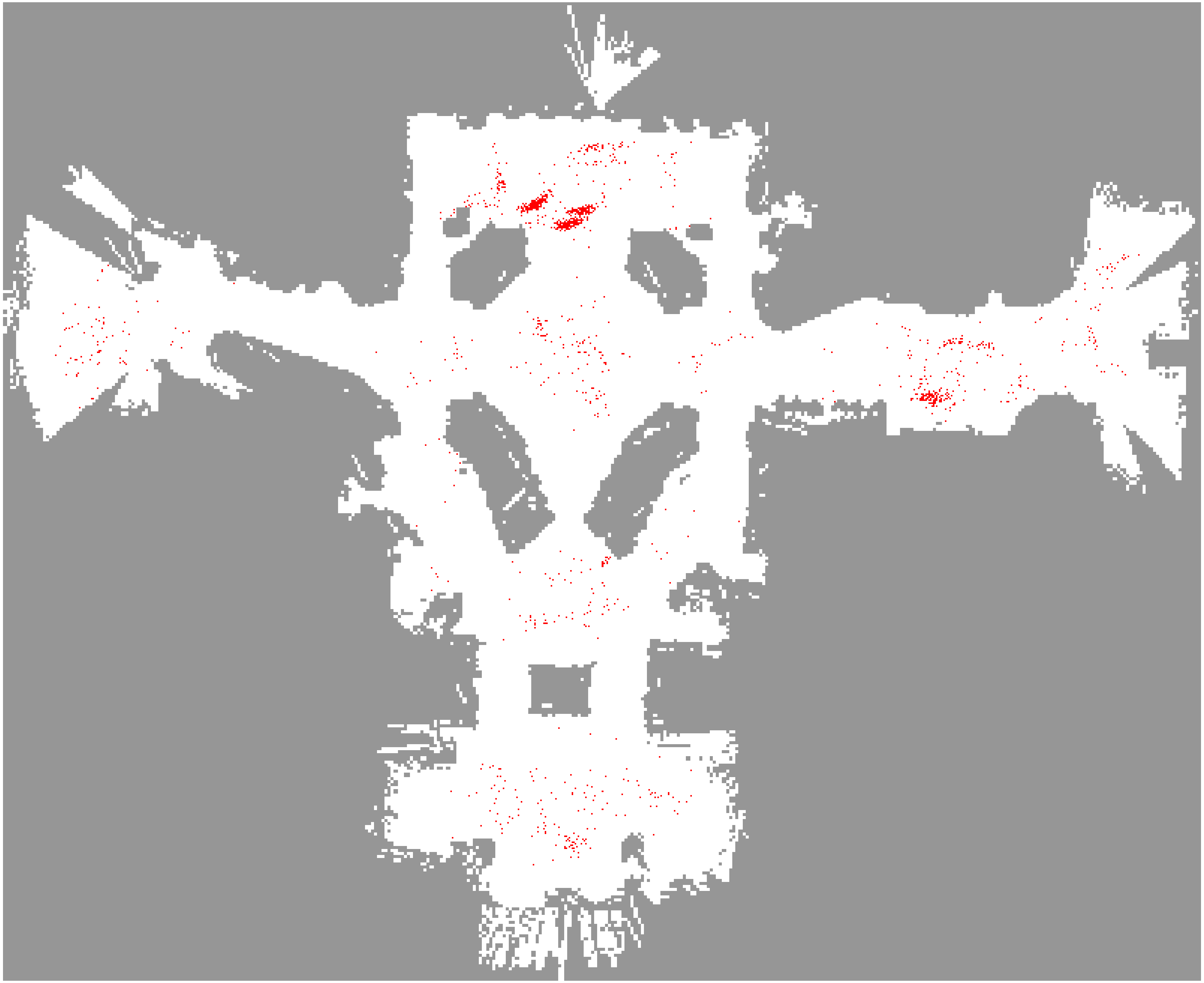


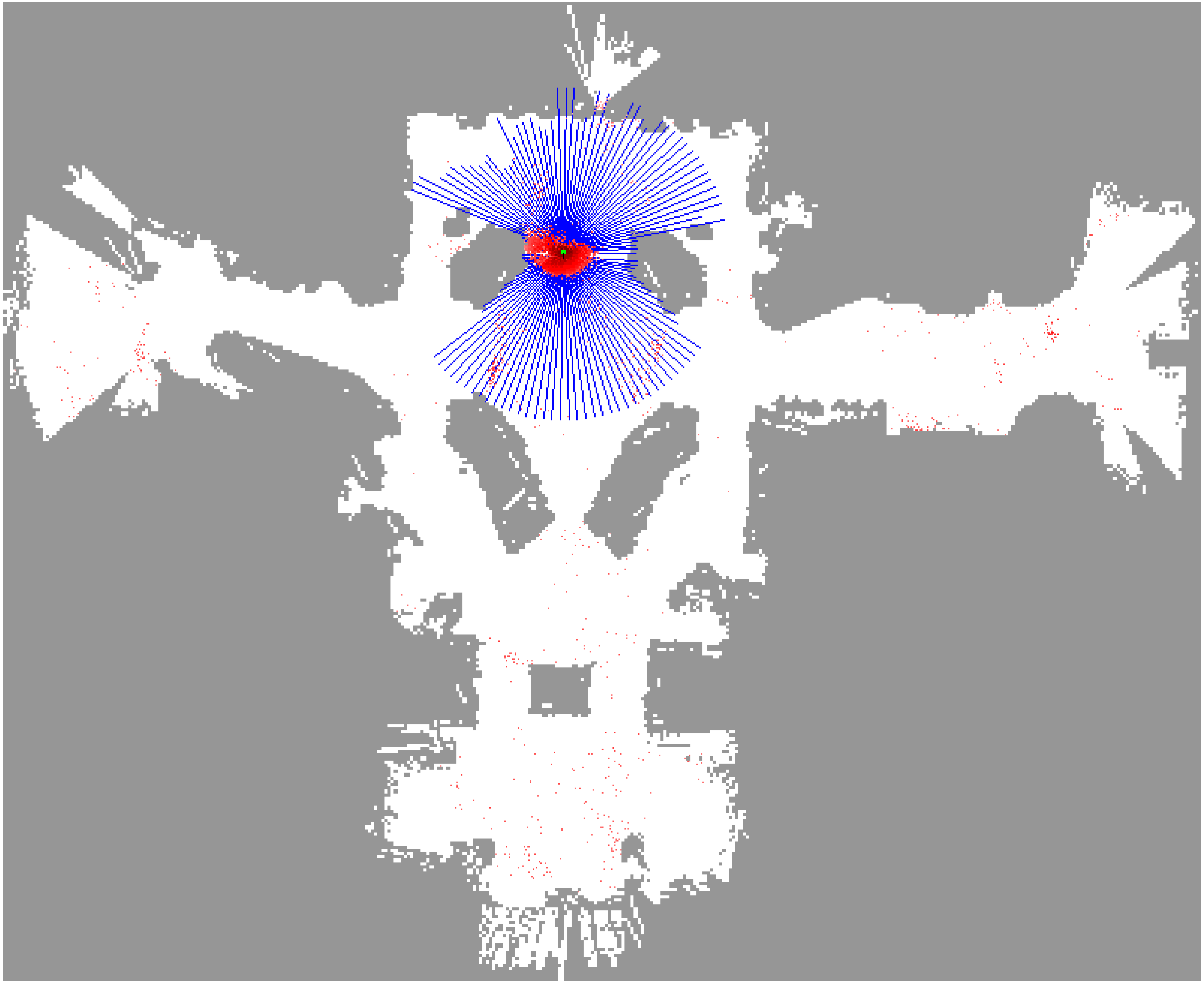


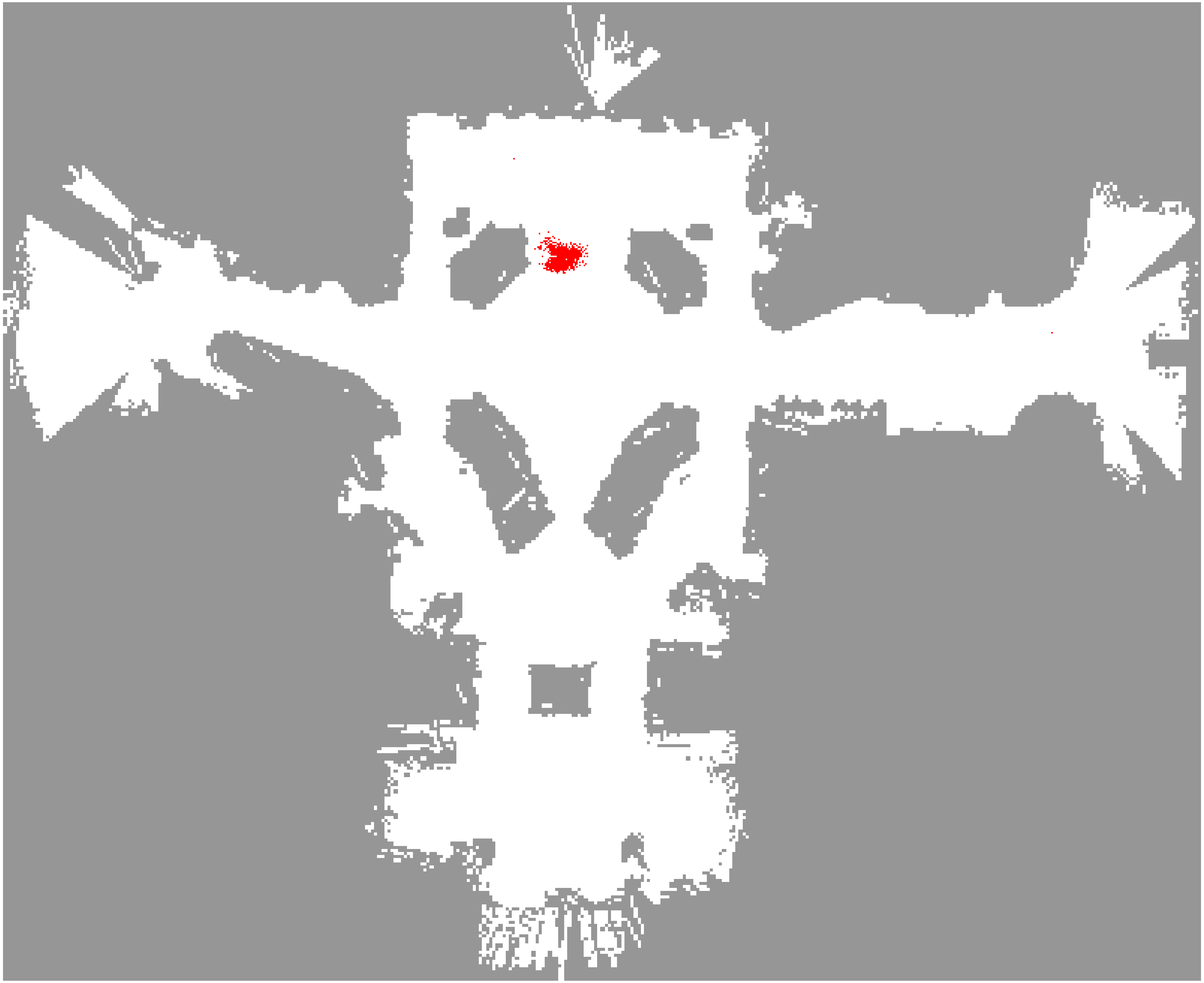


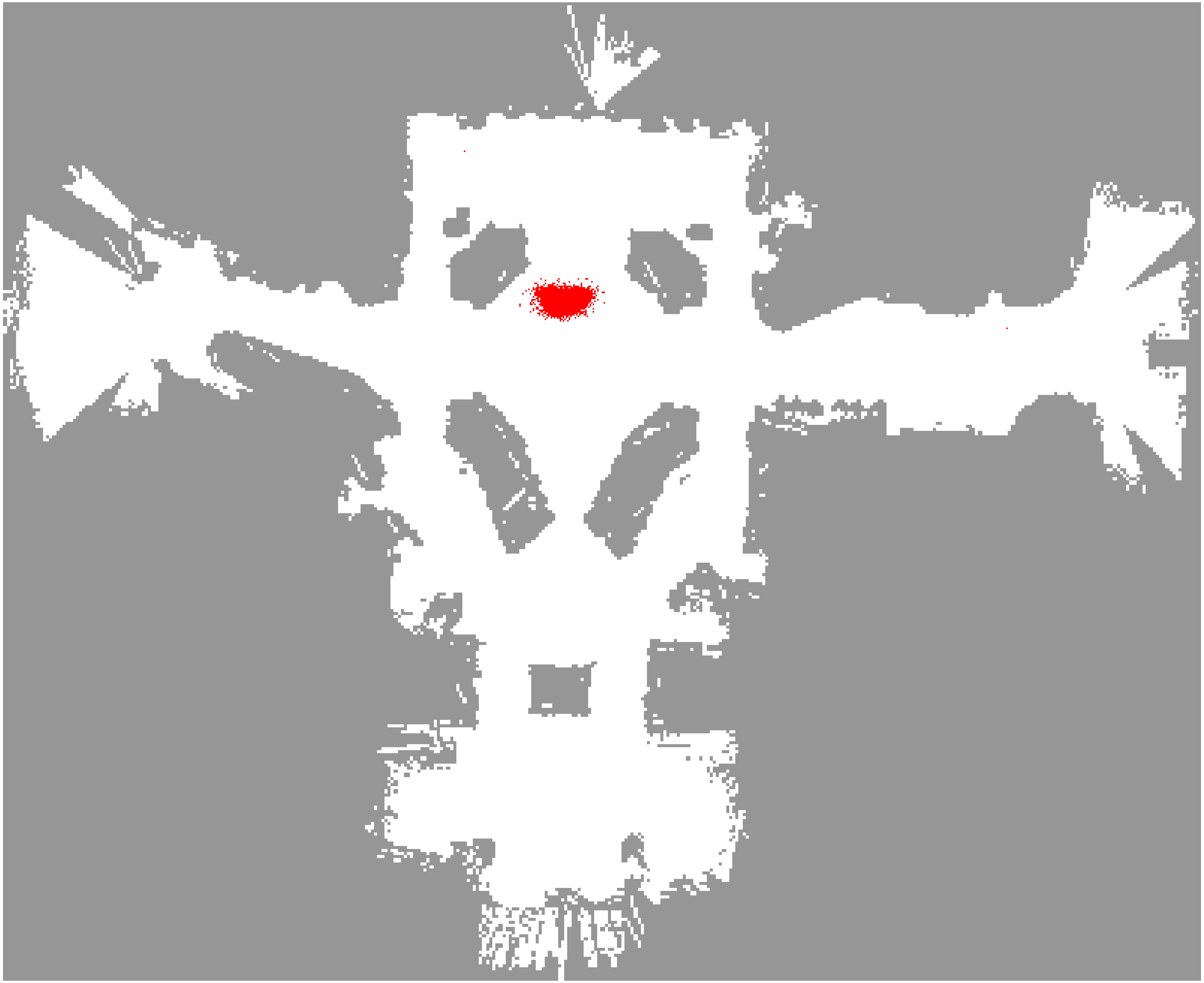


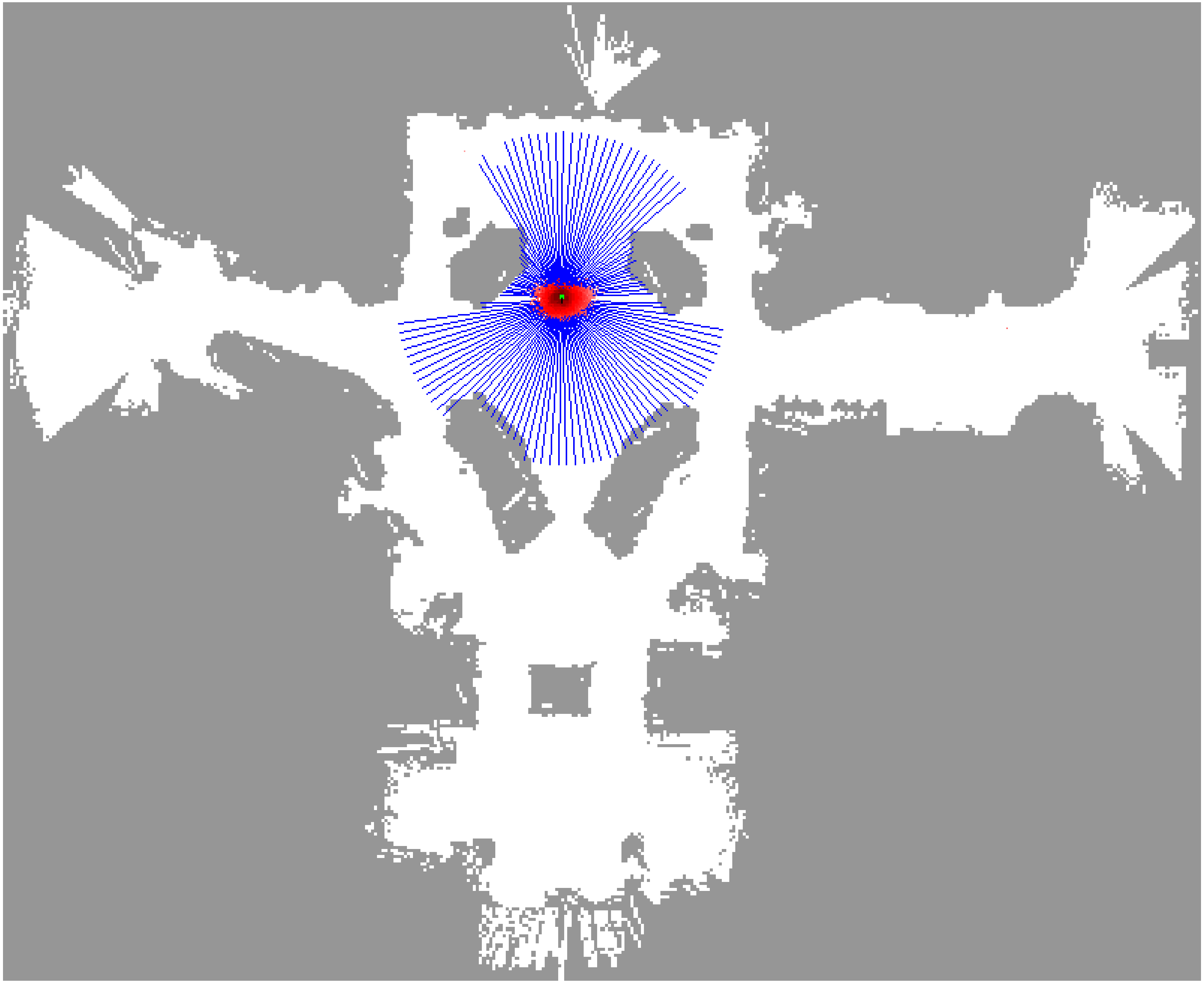


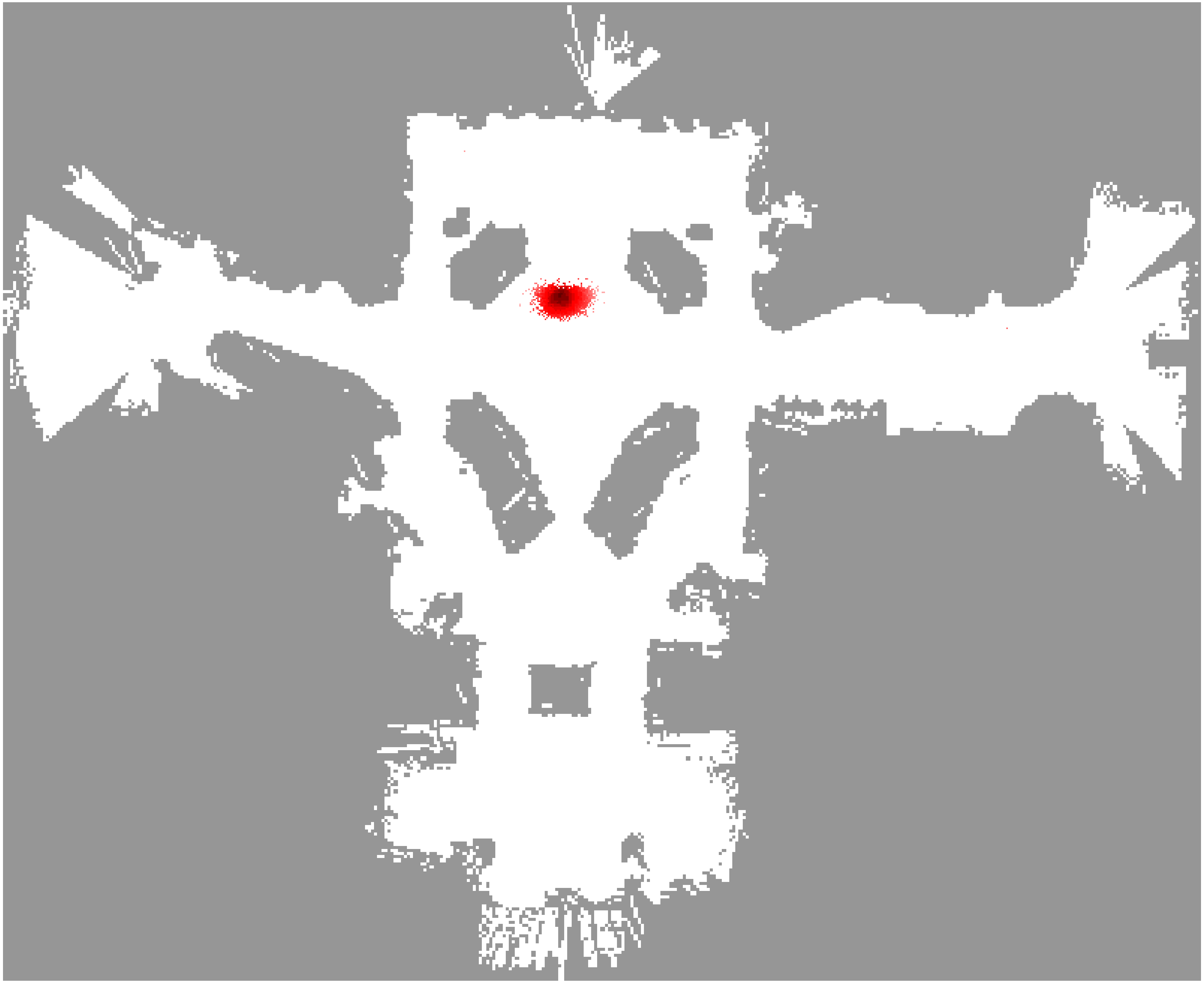


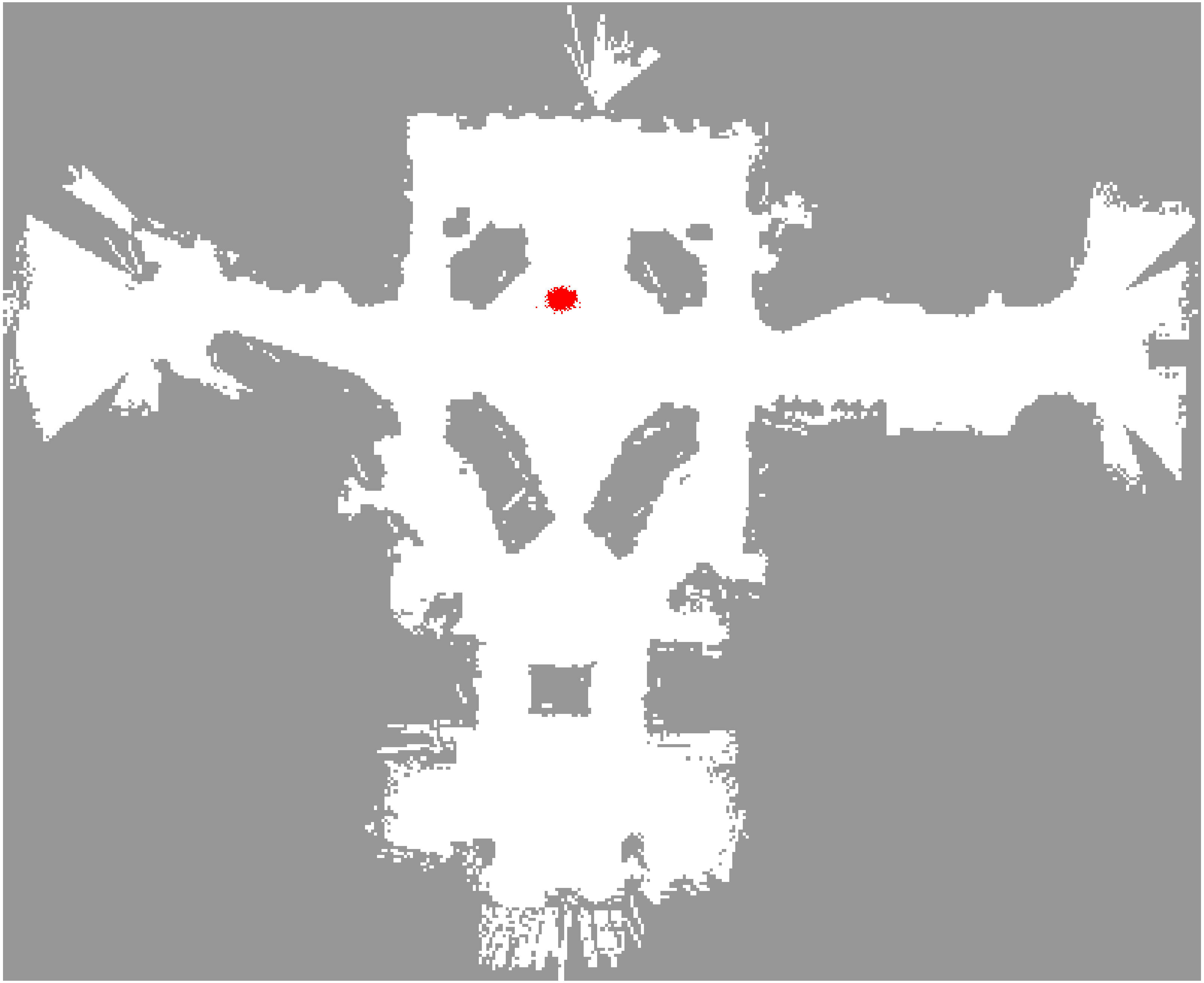


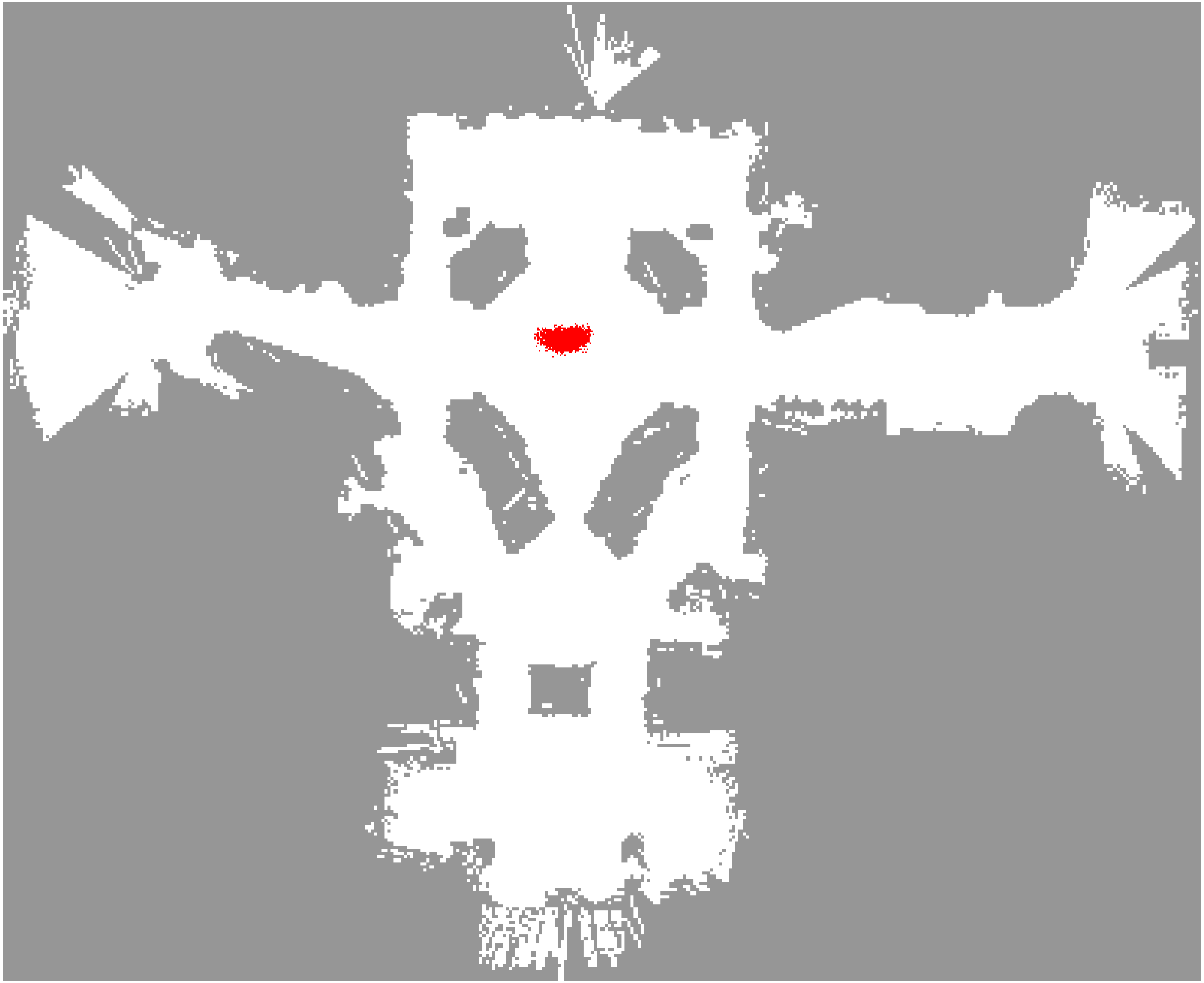


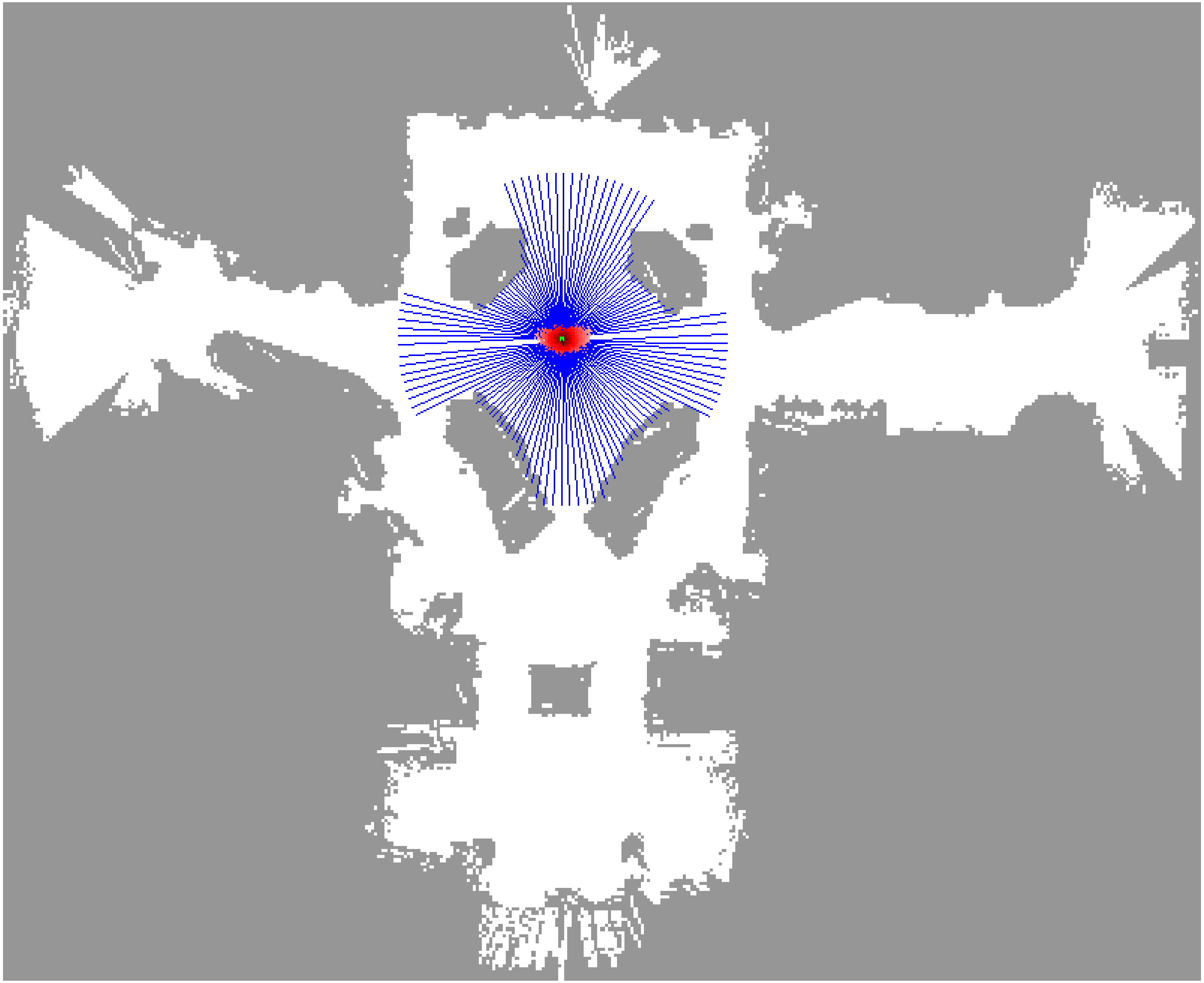


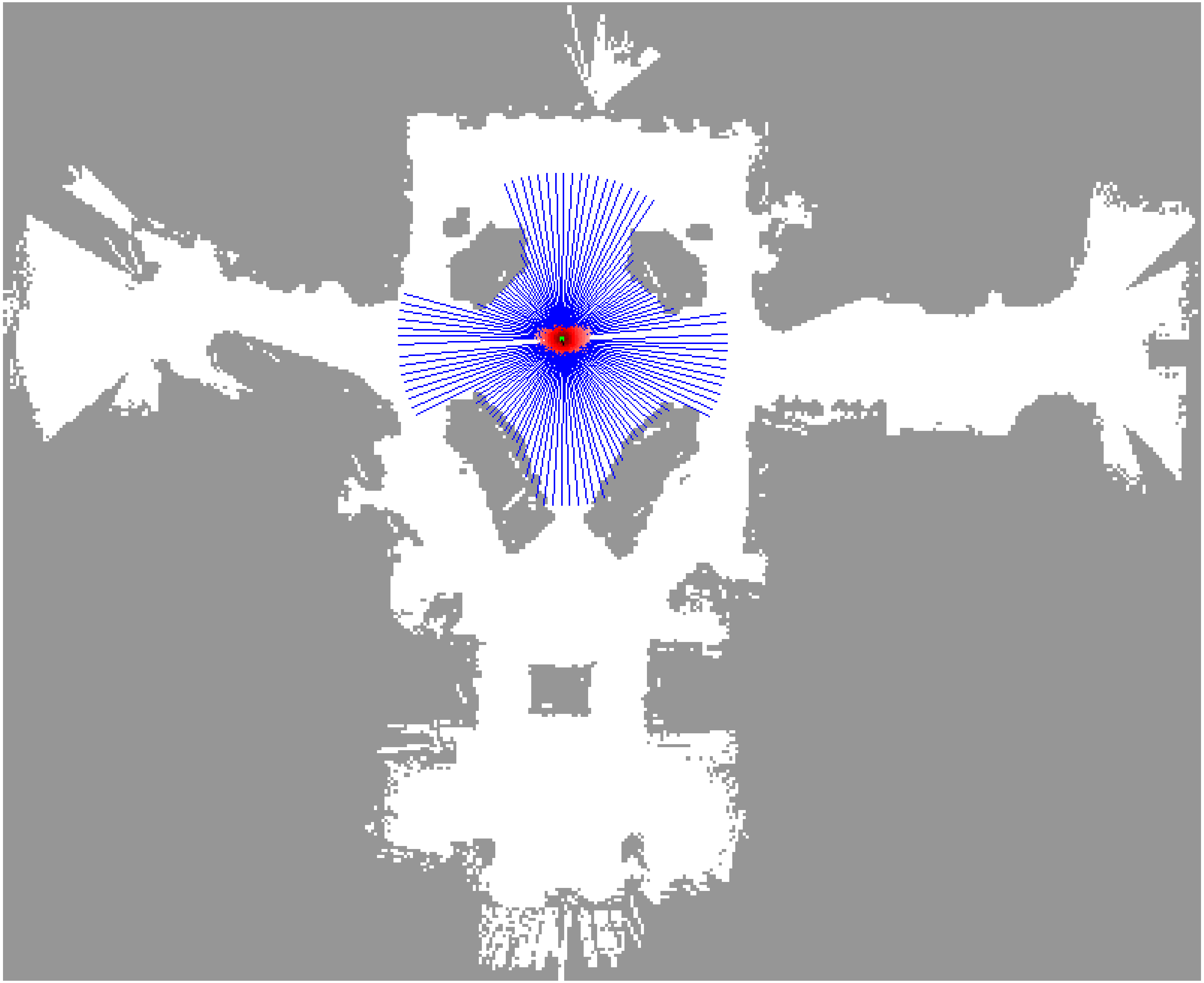




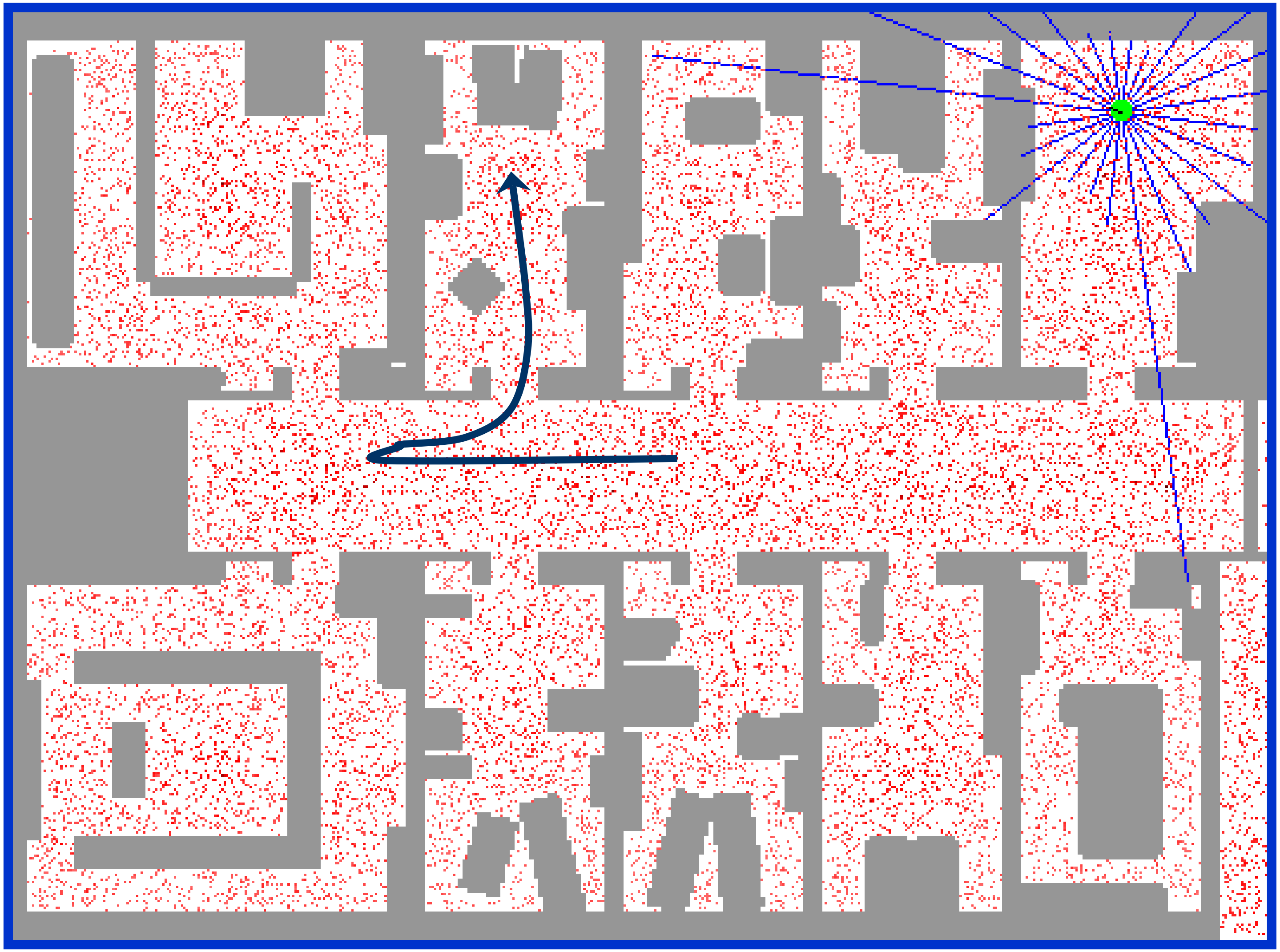




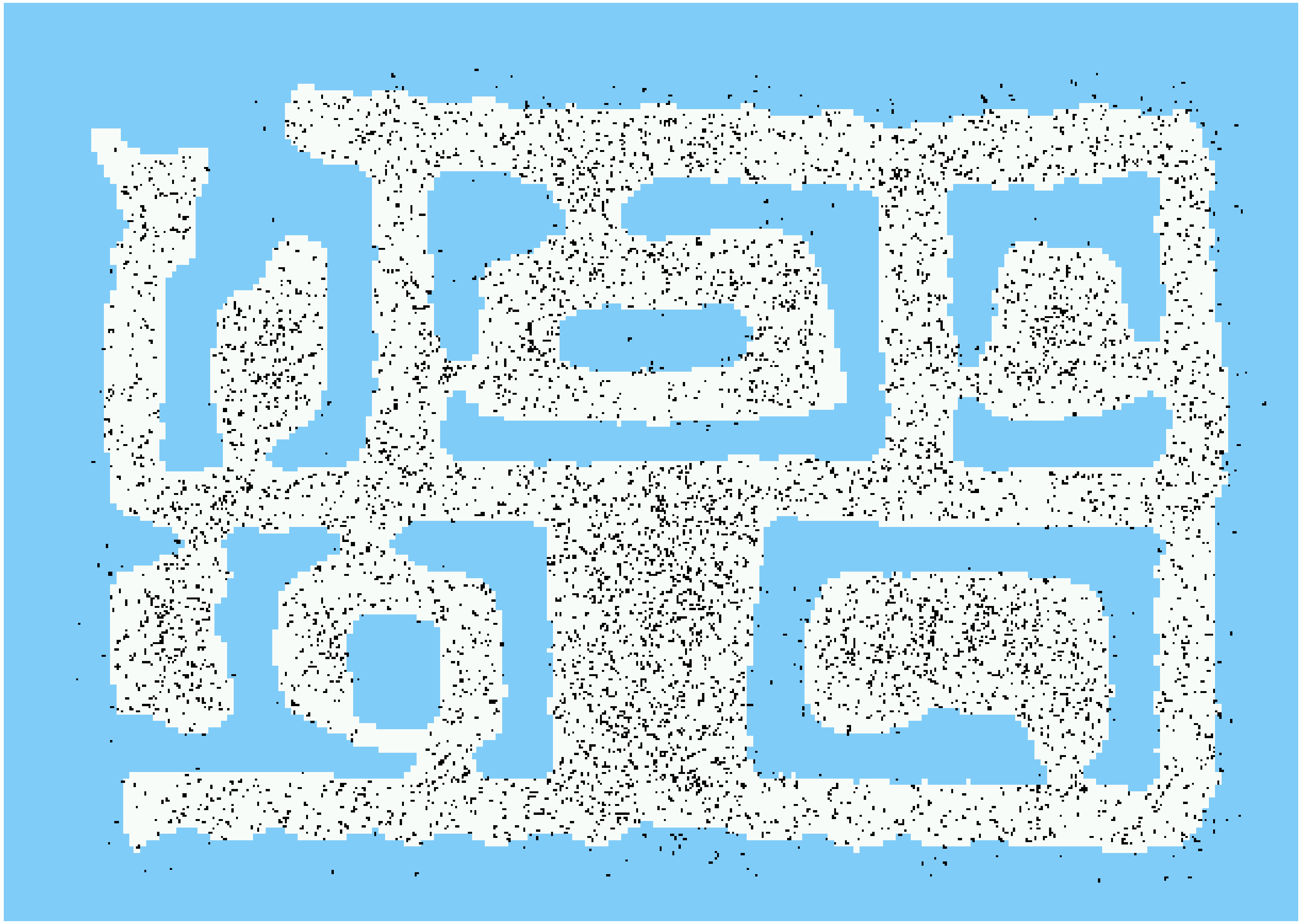




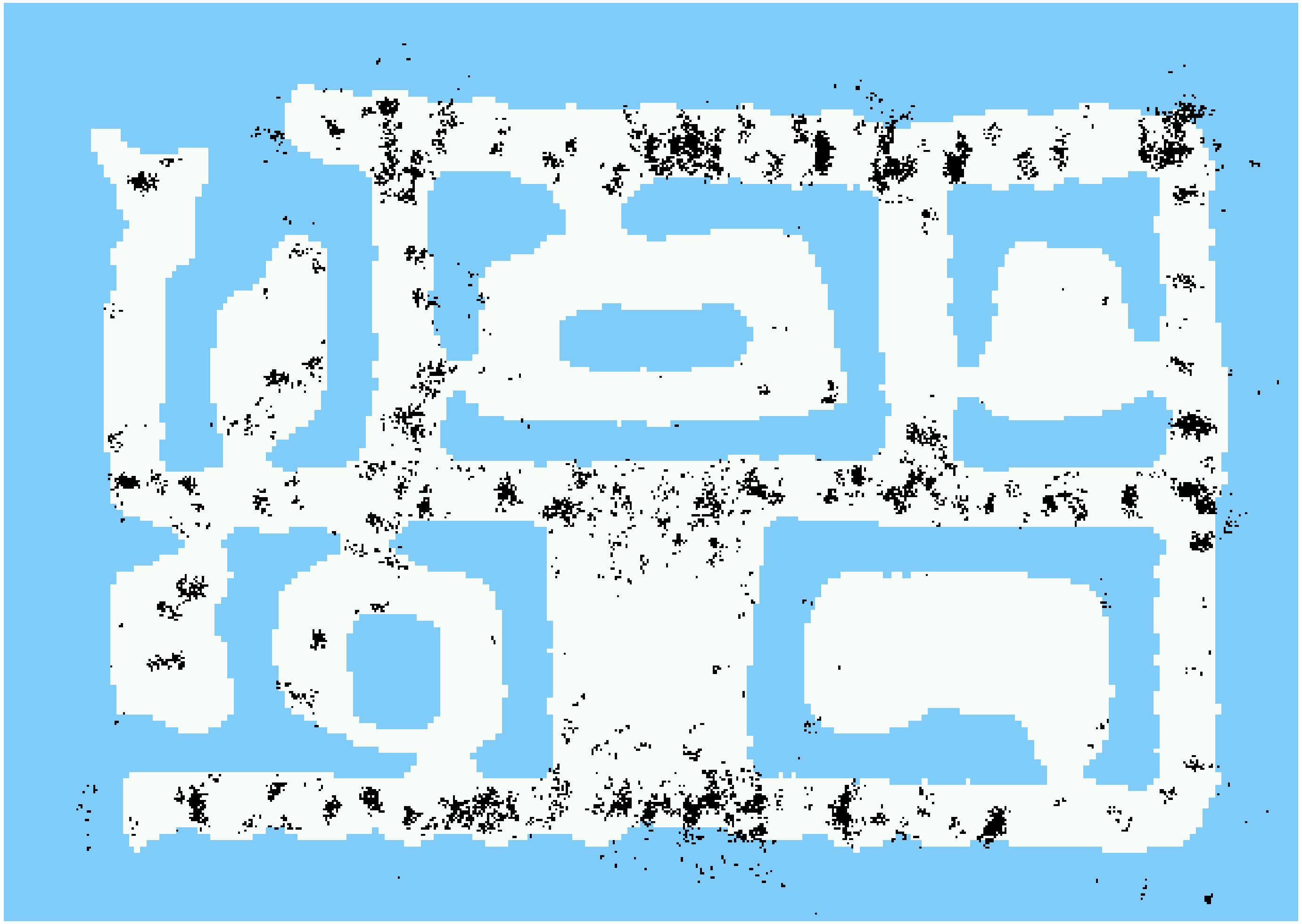
Sample-based Localization (sonar)



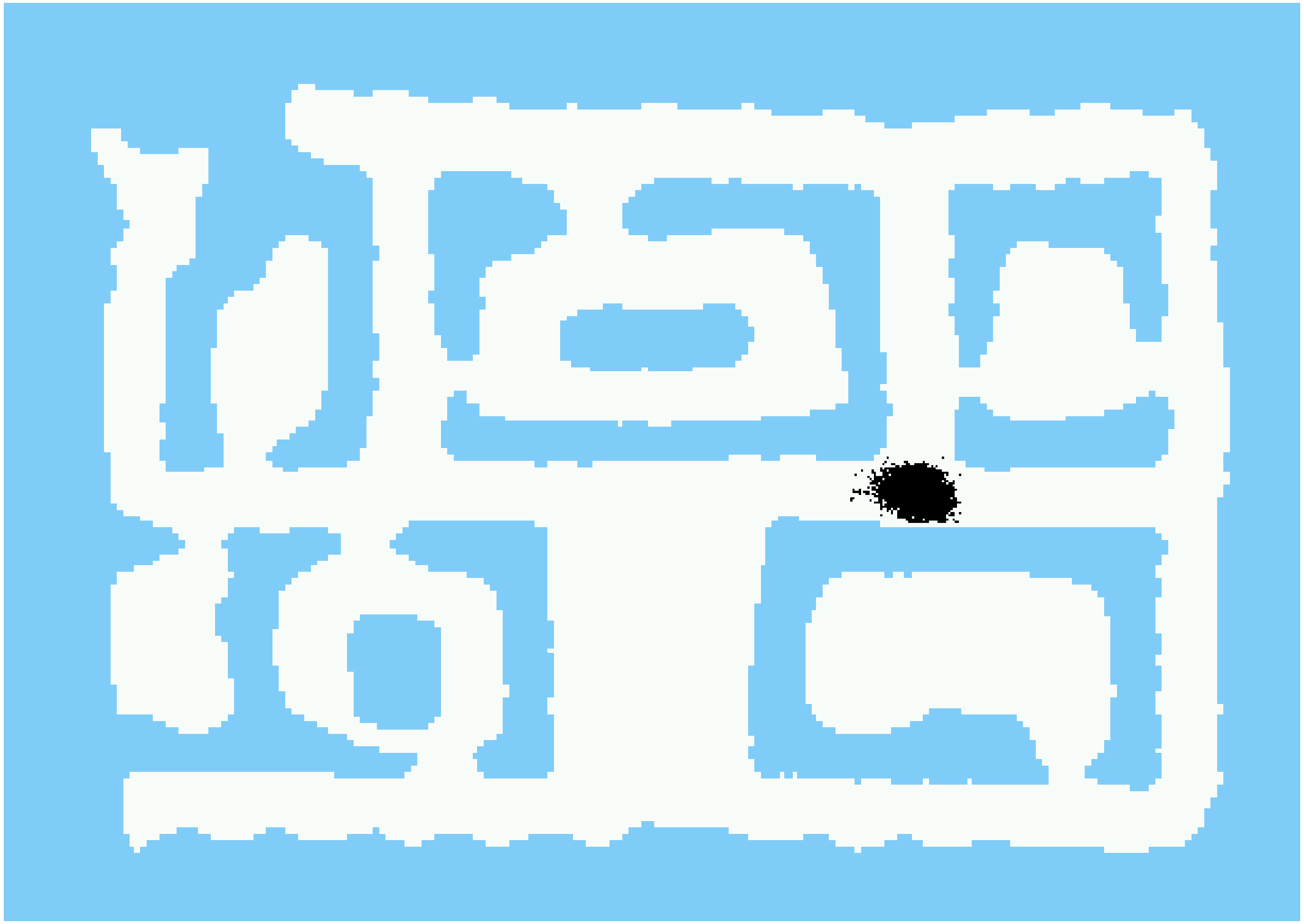
Initial Distribution



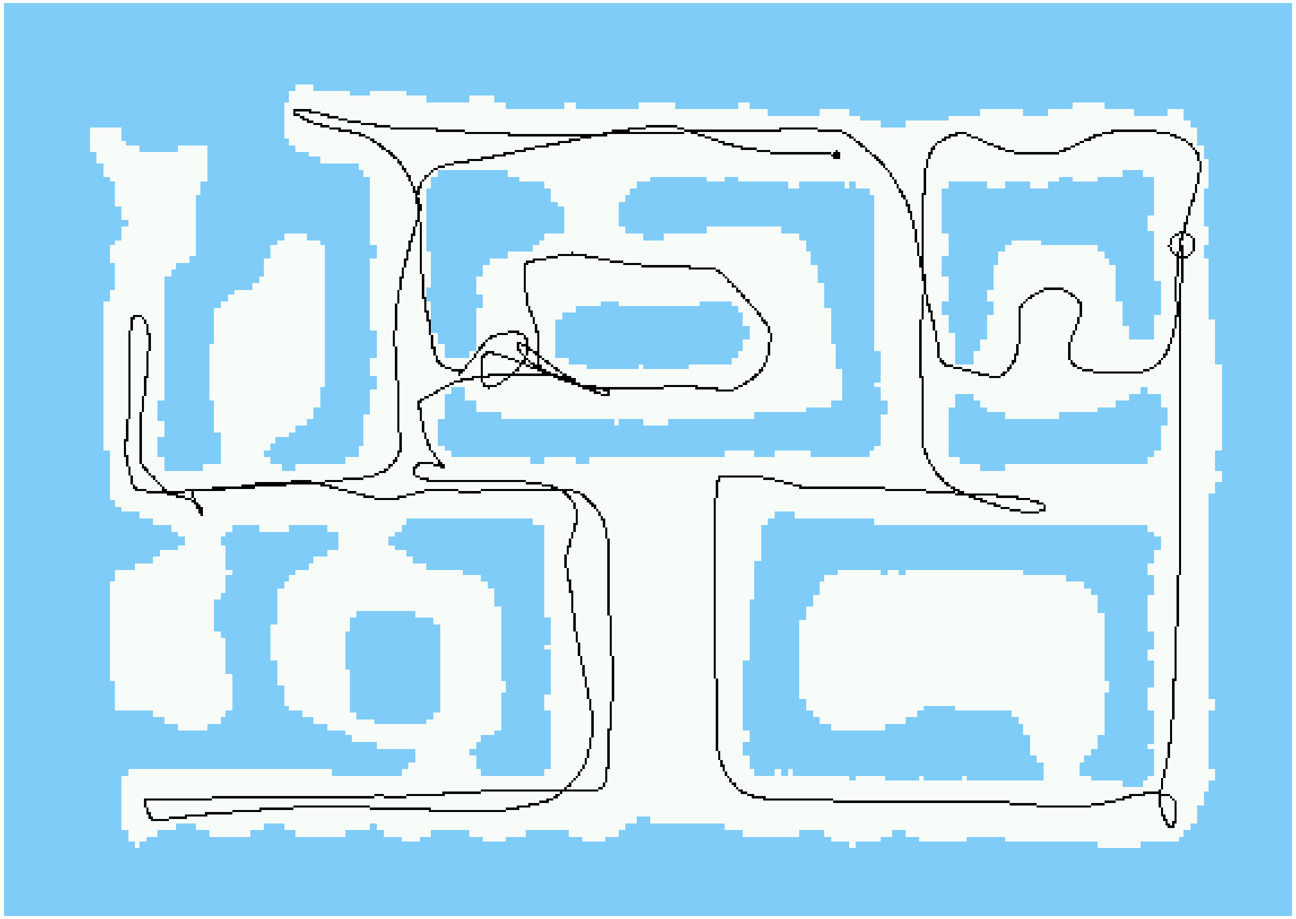
After Incorporating Ten Ultrasound Scans



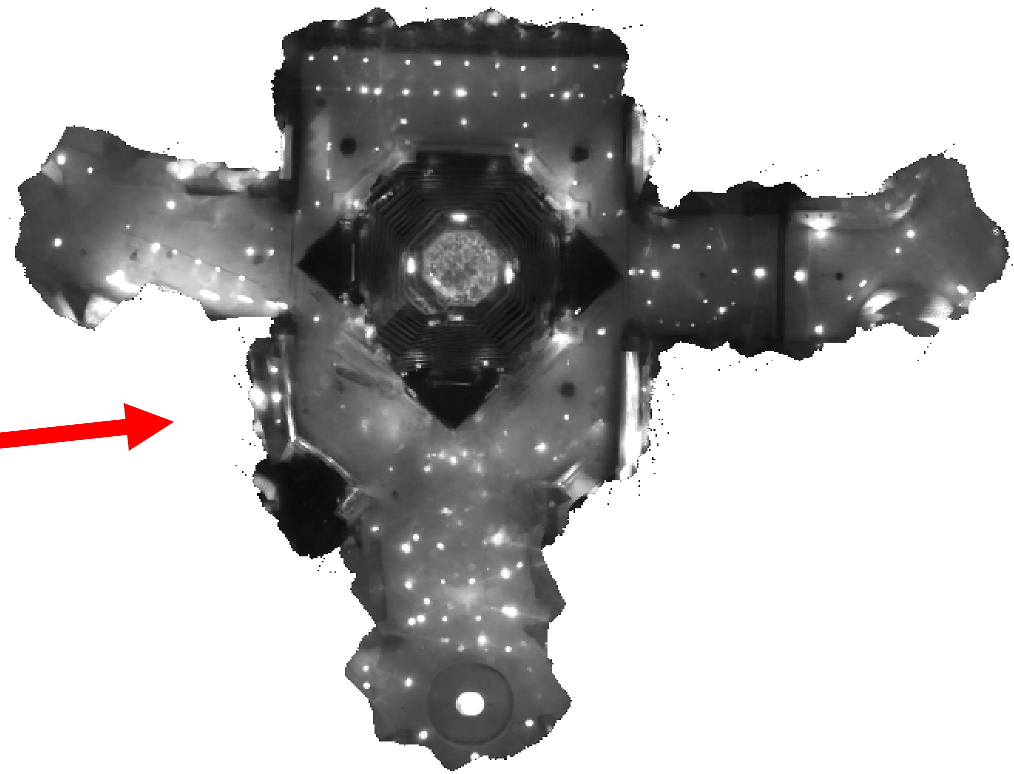
After Incorporating 65 Ultrasound Scans



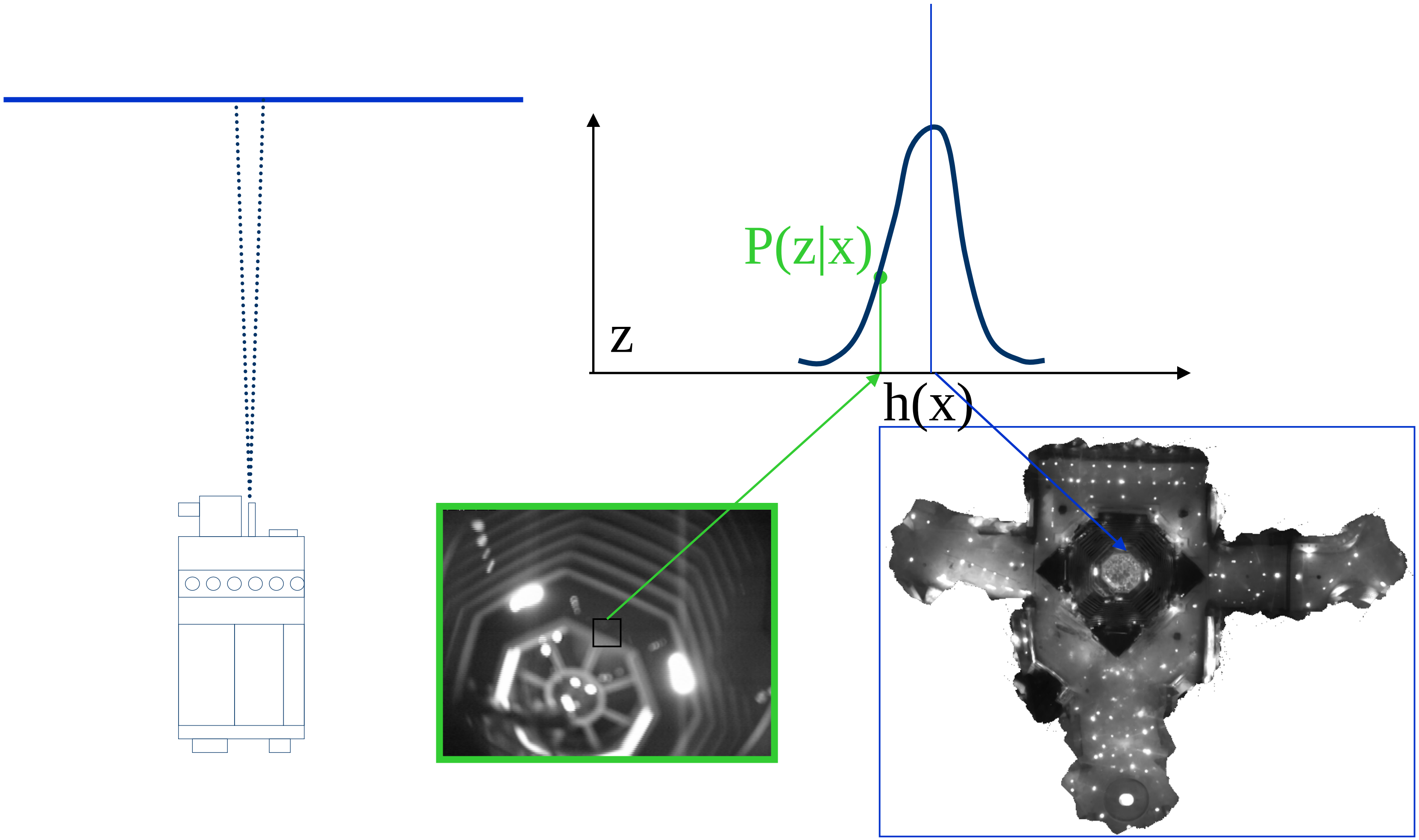
Estimated Path



Using Ceiling Maps for Localization

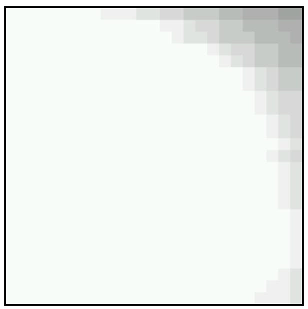


Vision-based Localization

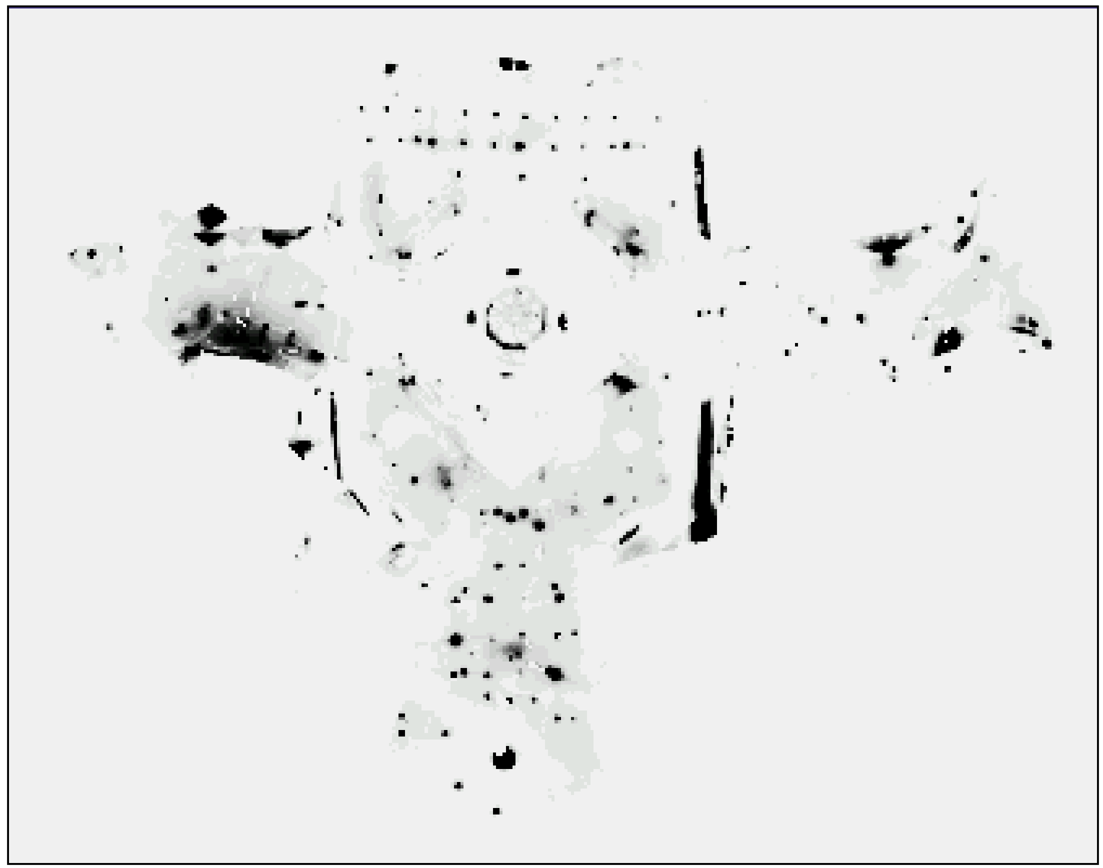


Under a Light

Measurement z:

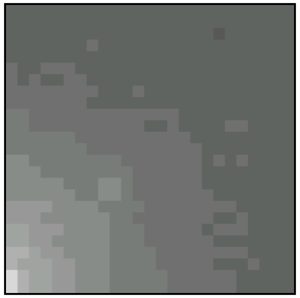


***P(z|x)*:**

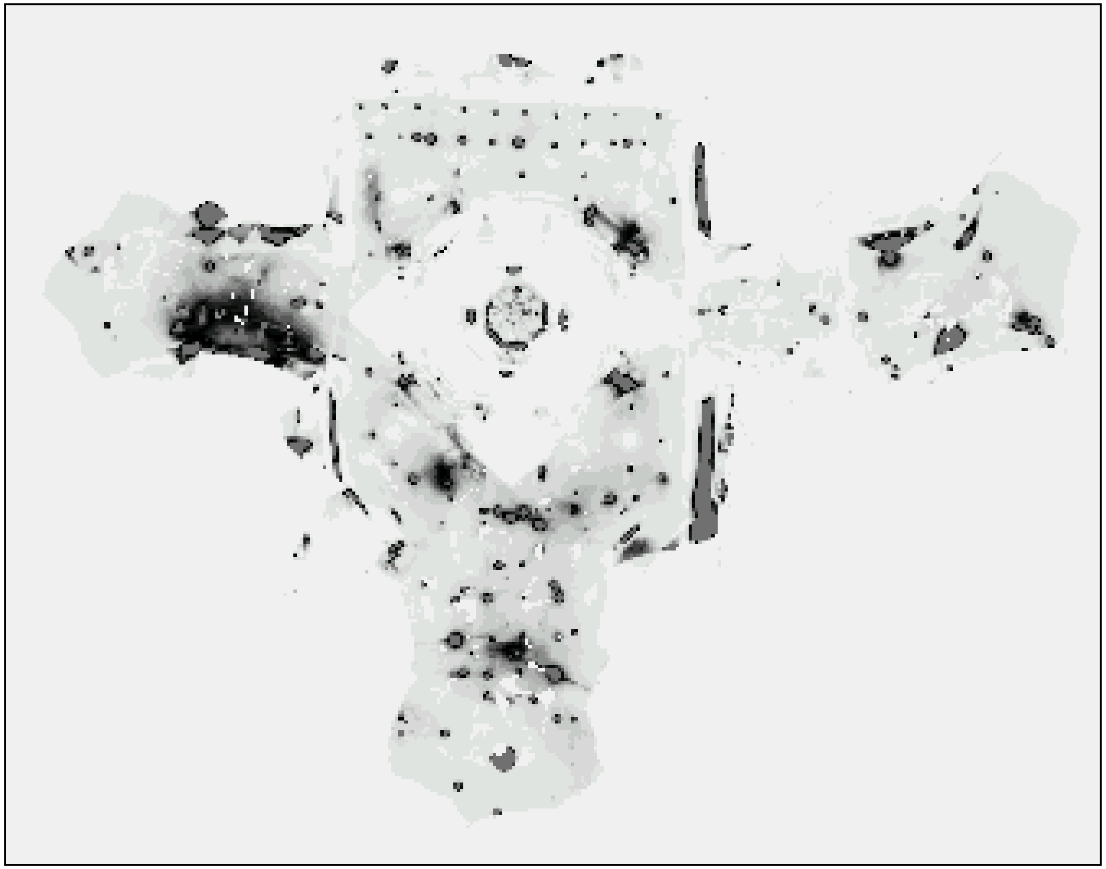


Next to a Light

Measurement z:



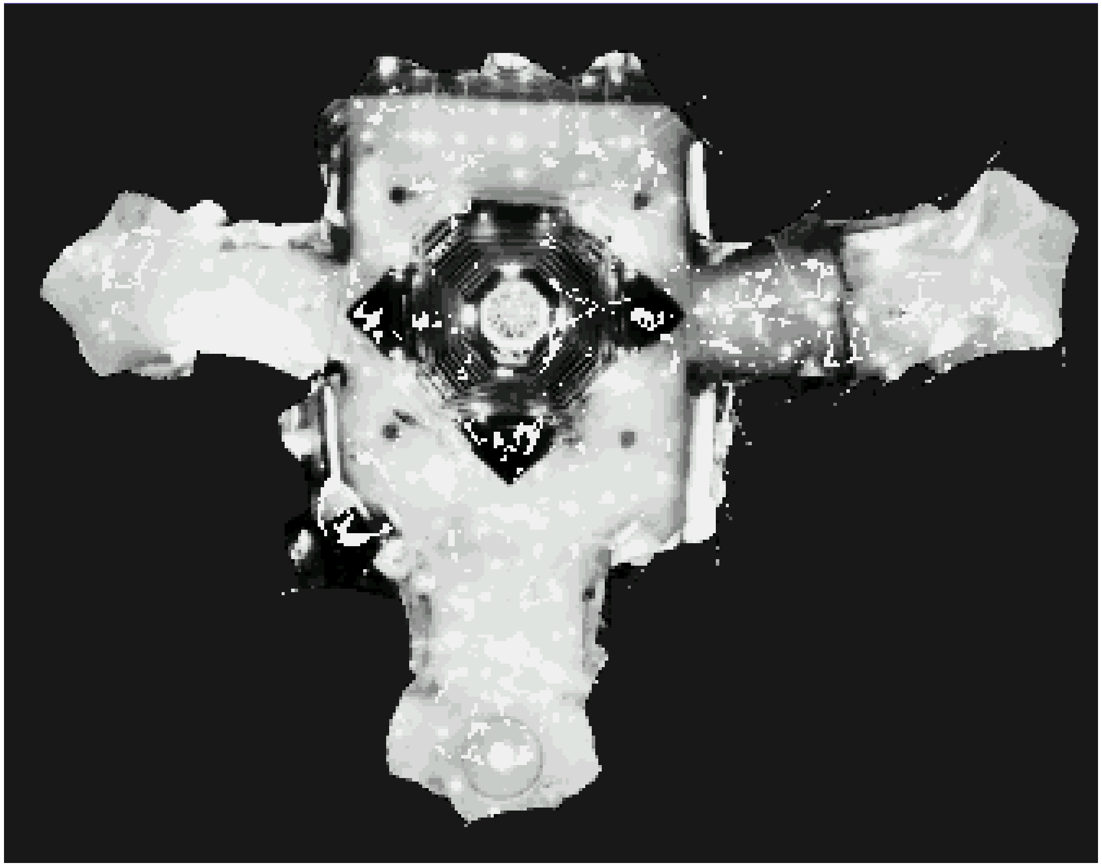
***P(z|x)*:**



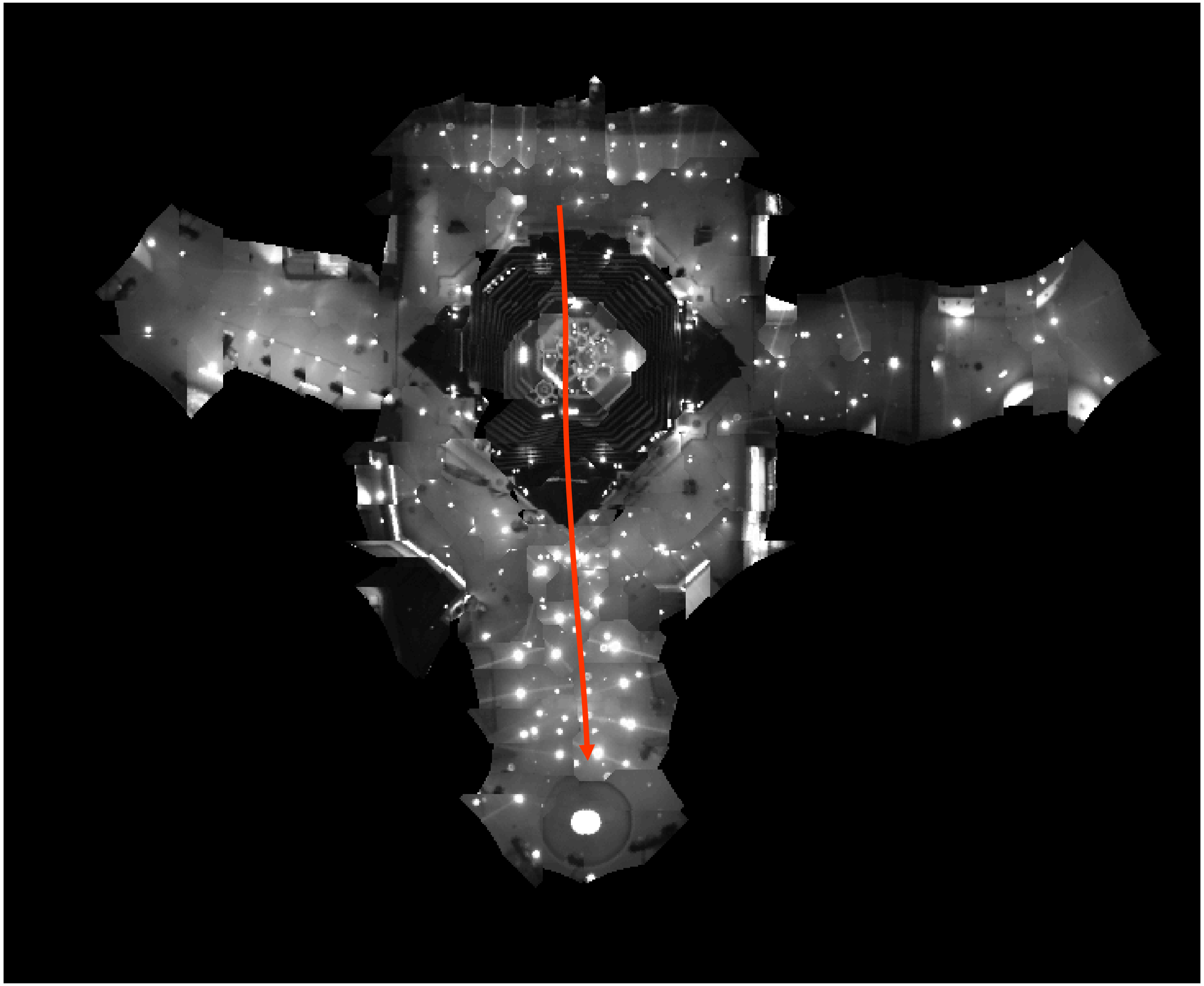
Elsewhere

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.

- How can we deal with localization errors (i.e., the kidnapped robot problem)?

Approaches

- 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).

Random Samples
Vision-Based Localization

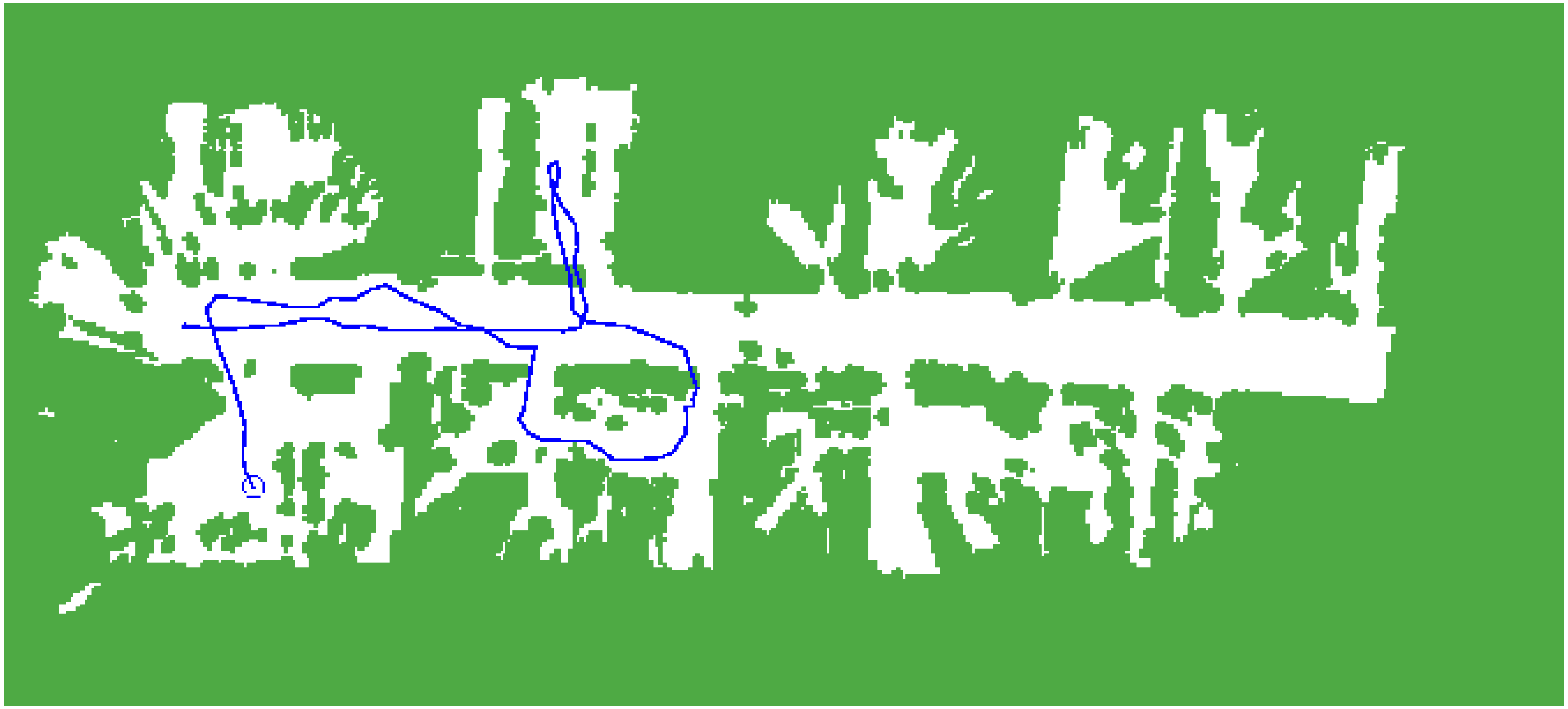
936 Images, 4MB, .6secs/image

Trajectory of the robot:

Odometry Information

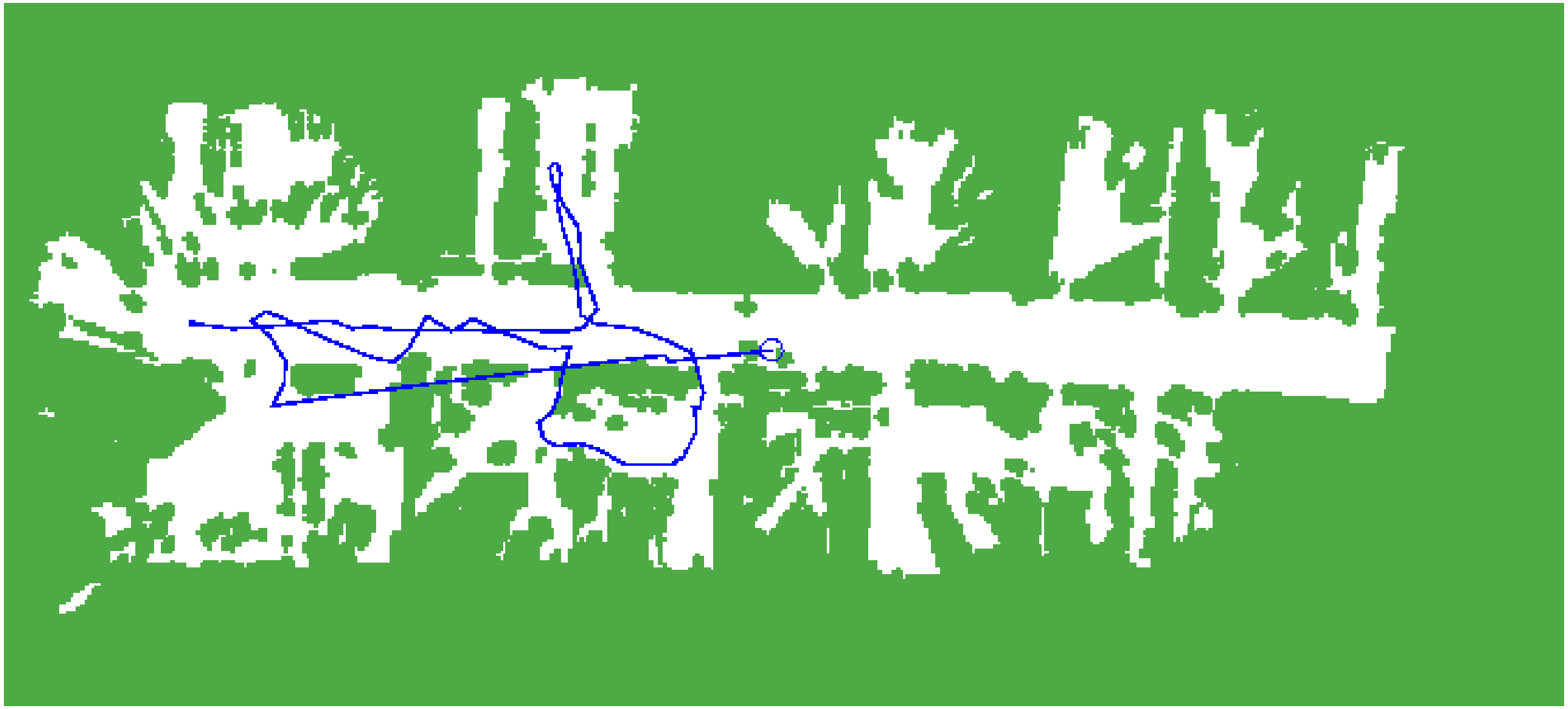
Resulting Trajectories

Position tracking:

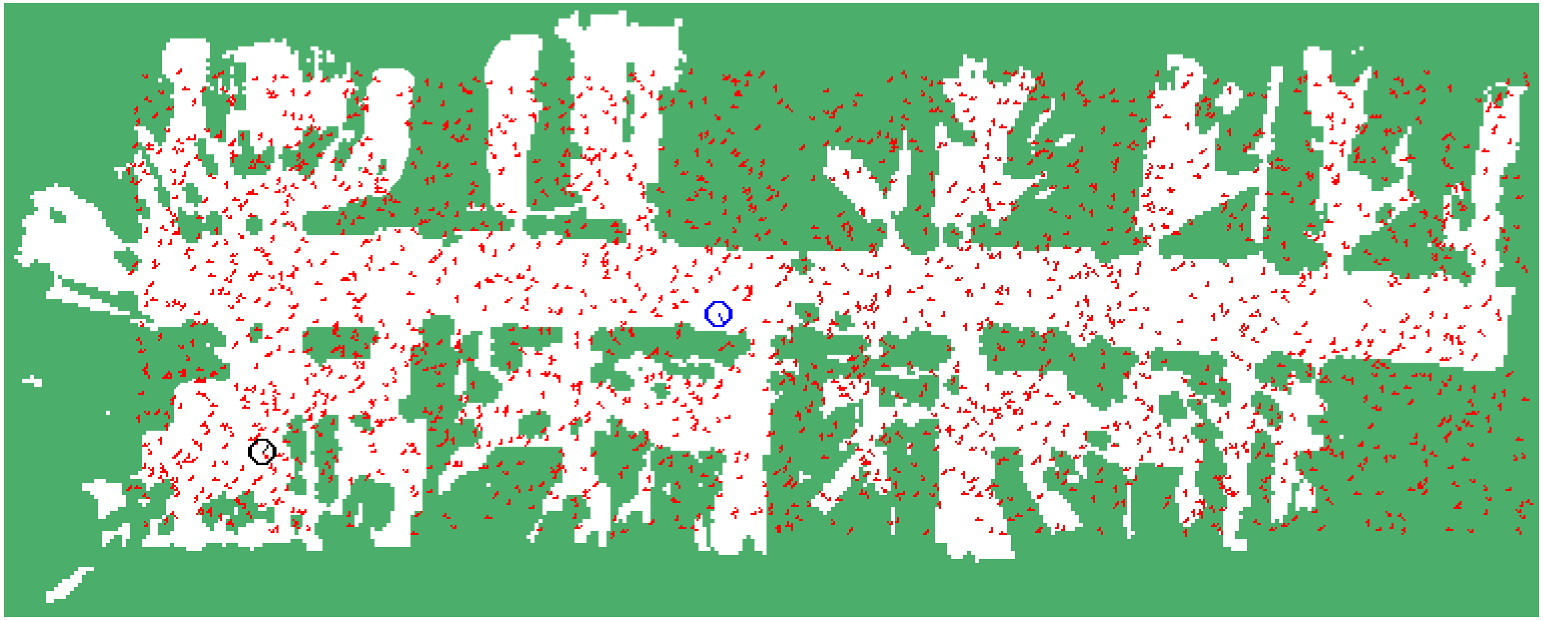


Resulting Trajectories

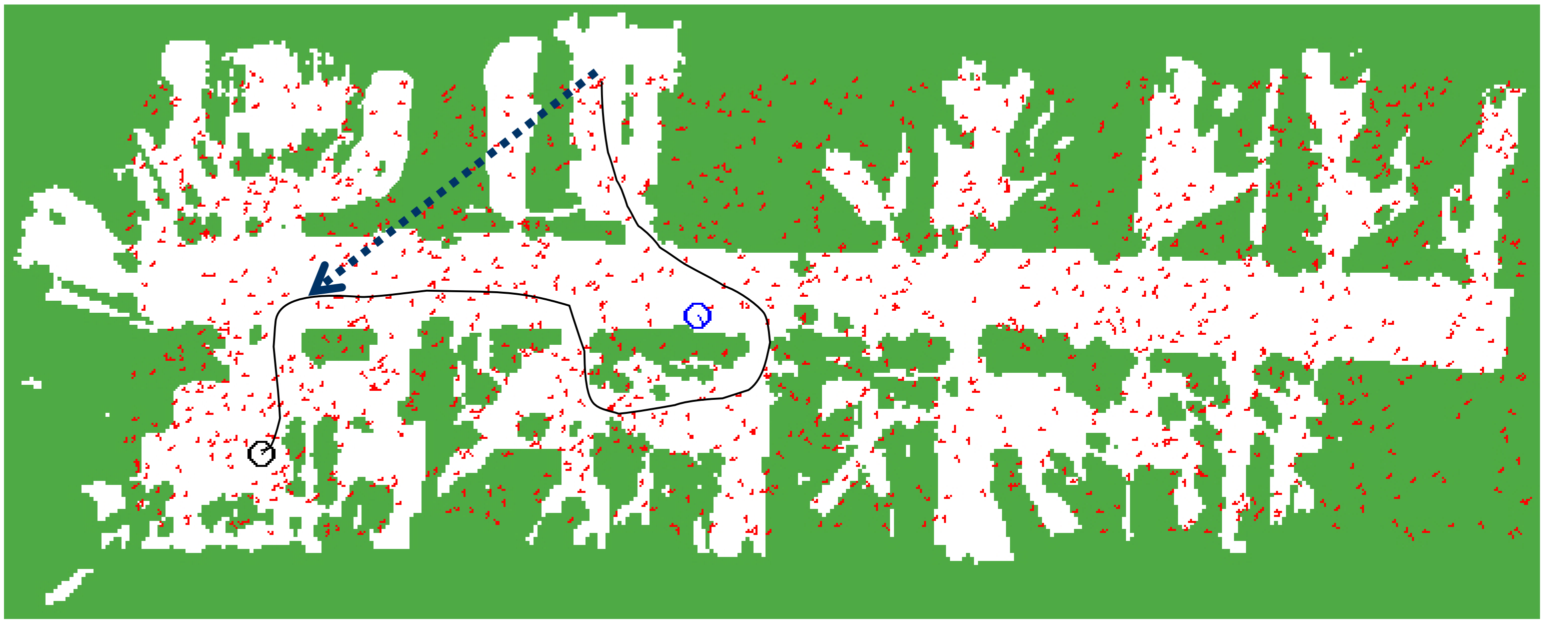
Global localization:



Global Localization



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.

Reading

- “Probabilistic Robotics”, Sebastian Thrun, Wolfram Burgard, and Dieter Fox, Chapter 2, Chapter 3

Logistics

- Great job in Project 1 ☺
- No lab this week ☺
- Project 2, Phase 1 due this Friday (13 Nov.)
- Project 2, Phase 2 to be released this week, due 20 Nov.