Introduction to Aerial Robotics
Lecture 9

Shaojie Shen
Associate Professor
Dept. of ECE, HKUST

18 April 2023

Outline

- Kalman Filter
  - Recap: Kalman Filter
  - Extended Kalman Filter
  - Augmented State Extended Kalman Filter
- Particle Filter

Recap: Kalman Filter

Markov Property

- **Definition:** The future state of the system is conditionally independent of the past states given the current state
  - $p(x_{t+1} | x_{0:t}) = p(x_{t+1} | x_t)$
  - $p(z_t \mid x_t, z_{1:t-1}, u_{1:t}) = p(z_t \mid x_t)$
  - $p(x_t \mid x_{t-1}, z_{1:t-1}, u_{1:t}) = p(x_t \mid x_{t-1}, u_t)$
- **Question:**
  - Which of the following satisfy the Markov assumption?
    - A first order system with $x = [\text{position}], u = [\text{velocity}]$
    - A second order system with $x = [\text{position}], u = [\text{acceleration}]$
      - How about with $x = [\text{position}, \text{velocity}], u = [\text{acceleration}]$

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

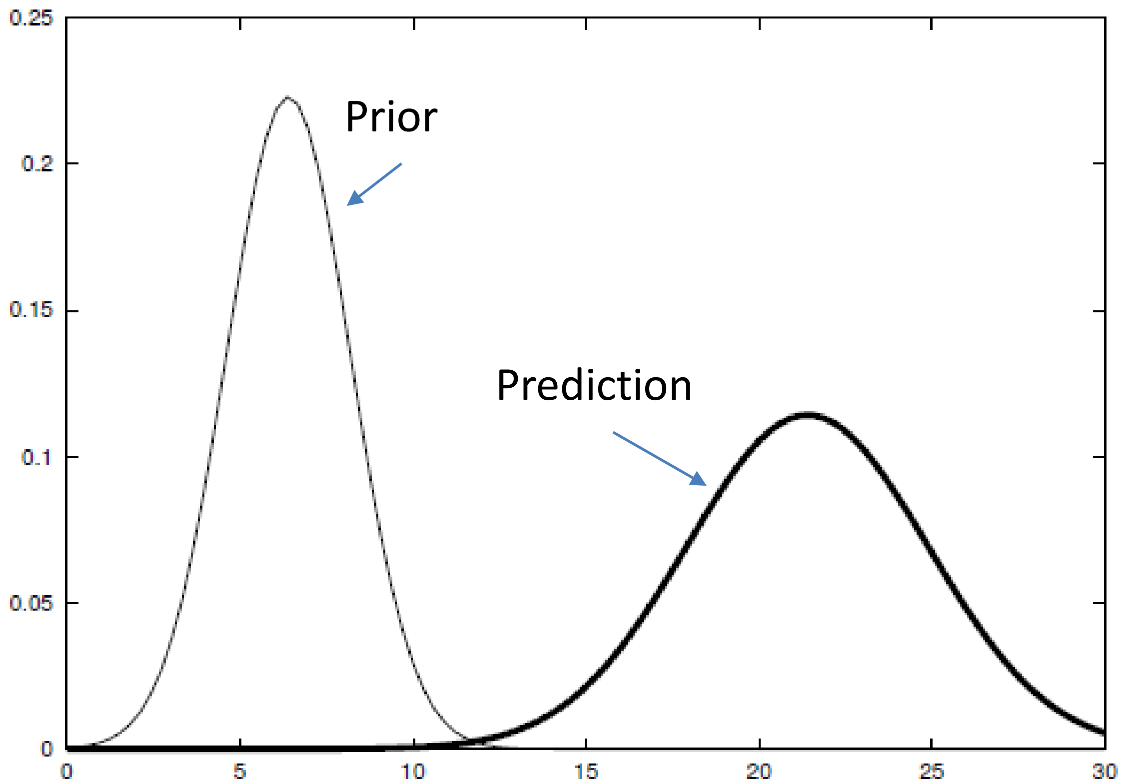
- **Prediction step:**
- $p(x_t \mid z_{1:t-1}, u_{1:t}) = \int f(x_t \mid x_{t-1}, u_t)\, p(x_{t-1} \mid z_{1:t-1}, u_{1:t-1})\, dx_{t-1}$
- **Update step:**
- $p(x_t \mid z_{1:t}, u_{1:t}) = \dfrac{g(z_t \mid x_t)\, p(x_t \mid z_{1:t-1}, u_{1:t})}{\int g(z_t \mid x'_t)\, p(x'_t \mid z_{1:t-1}, u_{1:t})\,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

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
  - $x_t = A_t\, x_{t-1} + B_t\, u_t + n_t$
  - $n_t \sim N(0, Q_t)$
  - $x_t, n_t \in \mathbf{R}^n, u_t \in \mathbf{R}^m$, $A_t, Q_t \in \mathbf{R}^{n\times n}$, and $B_t \in \mathbf{R}^{n\times m}$
- The measurement model $g(z_t \mid x_t)$ is linear with additive Gaussian white noise
  - $z_t = C_t\, x_t + v_t$
  - $v_t \sim N(0, R_t)$
  - $z_t, v_t \in \mathbf{R}^p, C_t \in \mathbf{R}^{p\times n}$, and $R_t \in \mathbf{R}^{p\times p}$

Kalman Filter – Prediction

- Bayes: $p(x_t \mid z_{1:t-1}, u_{1:t}) = \int f(x_t \mid x_{t-1}, u_t)\, p(x_{t-1} \mid z_{1:t-1}, u_{1:t-1})\, dx_{t-1}$



[Thrun, Burgard, and Fox. *Probabilistic Robotics*. 2005]

Kalman Filter – Prediction

- Bayes:
  - $p(x_t \mid z_{1:t-1}, u_{1:t}) = \int f(x_t \mid x_{t-1}, u_t)\, p(x_{t-1} \mid z_{1:t-1}, u_{1:t-1})\, dx_{t-1}$
- $x_t = A_t\, x_{t-1} + B_t u_t + n_t$
- $n_t \sim N(0, Q_t)$
- Prior: $p(x_{t-1} \mid z_{1:t-1}, u_{1:t-1}) \sim N(\mu_{t-1}, \Sigma_{t-1})$
- Prediction:
  - $\bar{\mu}_t = A\, \mu_{t-1} + B\, u_t$
  - $\bar{\Sigma}_t = A\, \Sigma_{t-1}\, A^T + Q$

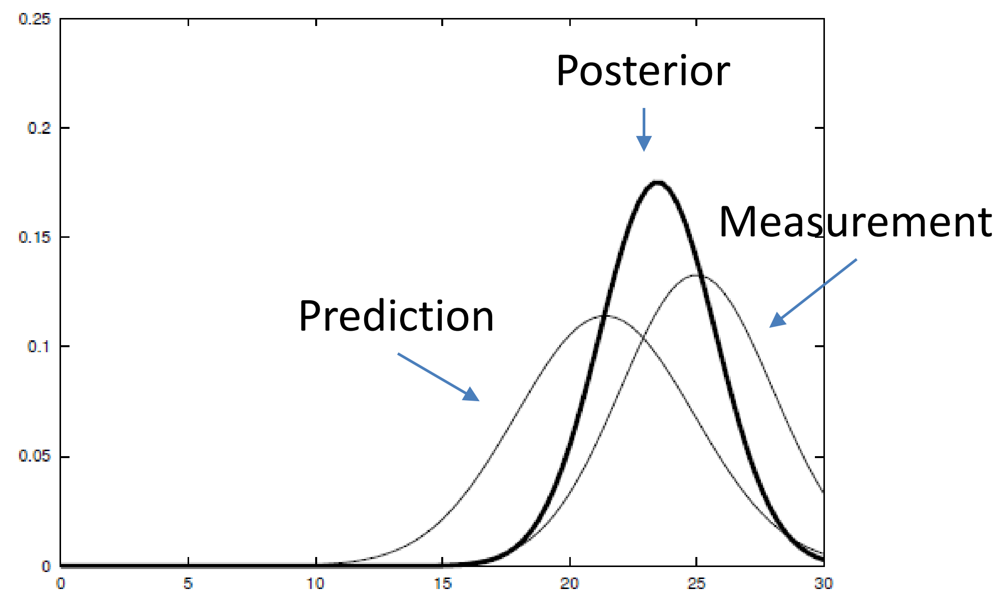
Kalman Filter – Update

- Bayes: $p(x_t \mid z_{1:t}, u_{1:t}) = \dfrac{g(z_t \mid x_t)\, p(x_t \mid z_{1:t-1}, u_{1:t})}{\int g(z_t \mid x_t')\, p\left(x_t' \mid z_{1:t-1}, u_{1:t}\right) dx_t'}$
- The observation model is $z_t = C_t\, \bar{x}_t + v_t, \quad v_t \sim N(0, R_t)$
- The best update without a measurement is to set $x_t = \bar{x}_t$
- $\begin{bmatrix} x_t \\ z_t \end{bmatrix} = \begin{bmatrix} I & 0 \\ C & I \end{bmatrix} \begin{bmatrix} \bar{x}_t \\ v_t \end{bmatrix}$
- **Question:** Is this a jointly normal distribution?
- $\mu = \begin{bmatrix} \bar{\mu}_t \\ C\bar{\mu}_t \end{bmatrix}$
- $\Sigma = \begin{bmatrix} I & 0 \\ C & I \end{bmatrix} \begin{bmatrix} \bar{\Sigma}_t & 0 \\ 0 & R \end{bmatrix} \begin{bmatrix} I & C^T \\ 0 & I \end{bmatrix} = \begin{bmatrix} \bar{\Sigma}_t & \bar{\Sigma}_t C^T \\ C\bar{\Sigma}_t & C\bar{\Sigma}_t C^T + R \end{bmatrix}$

Kalman Filter – Update

- The distribution of x_t conditioned on z_t is thus normal with
- $\mu_{x_t|z_t} = \bar{\mu}_t + \bar{\Sigma}_t C^T (C\bar{\Sigma}_t C^T + R)^{-1} (z_t - C\bar{\mu}_t)$
- $\Sigma_{x_t|z_t} = \bar{\Sigma}_t - \bar{\Sigma}_t C^T (C\bar{\Sigma}_t C^T + R)^{-1} C\bar{\Sigma}_t$

- Define the Kalman gain K_t
- $K_t = \bar{\Sigma}_t C^T (C\bar{\Sigma}_t C^T + R)^{-1}$
- $\mu_t = \bar{\mu}_t + K_t (z_t - C\bar{\mu}_t)$
- $\Sigma_t = \bar{\Sigma}_t - K_t C\bar{\Sigma}_t$



[Thrun, Burgard, and Fox. *Probabilistic Robotics*. 2005]

Kalman Gain

- $K_t = \bar{\Sigma}_t \, C^T \, (C \, \bar{\Sigma}_t \, C^T + R)^{-1}$
- **Intuition:** How much to trust the sensor vs. the prediction
- **Example:**
  - Perfect sensor $R = 0$
    - $K_t = \bar{\Sigma}_t \, C^T \, (C \, \bar{\Sigma}_t \, C^T + R)^{-1} = C^{-1}$
    - $\mu_t = \bar{\mu}_t + K_t(z_t - C \, \bar{\mu}_t) = C^{-1} z_t$
    - $\Sigma_t = \bar{\Sigma}_t - K_t \, C \, \bar{\Sigma}_t = 0$
  - Horrible sensor $R \to \infty$
    - $K_t = \bar{\Sigma}_t \, C^T \, (C \, \bar{\Sigma}_t \, C^T + R)^{-1} \to 0$
    - $\mu_t = \bar{\mu}_t + K_t(z_t - C \, \bar{\mu}_t) \to \bar{\mu}_t$
    - $\Sigma_t = \bar{\Sigma}_t - K_t \, C \, \bar{\Sigma}_t \to \bar{\Sigma}_t$

Kalman Filter

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

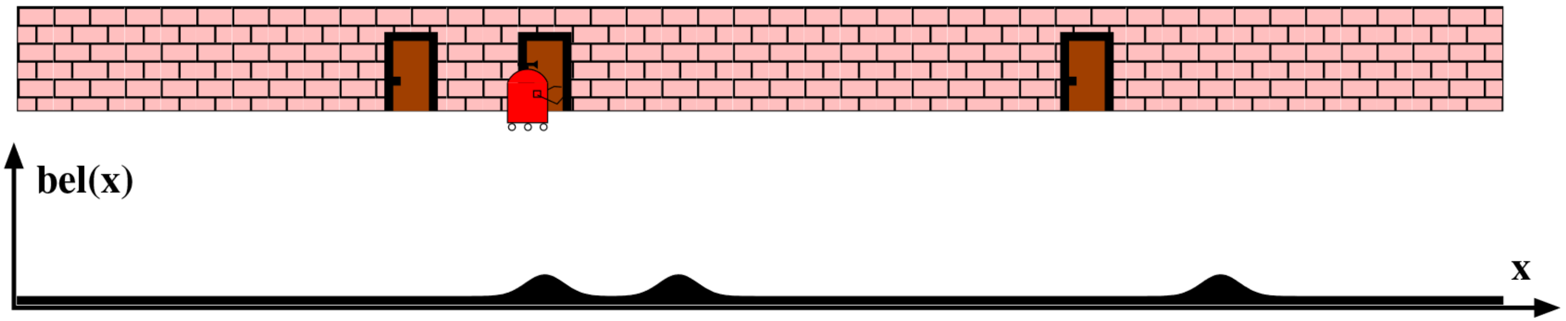
- Prior:
  - μ_{t-1}, Σ_{t-1}
- Prediction:
  - $\bar{\mu}_t = A_t\, \mu_{t-1} + B_t\, u_t$
  - $\bar{\Sigma}_t = A_t\, \Sigma_{t-1}\, A_t^T + Q_t$
- Update:
  - $\mu_t = \bar{\mu}_t + K_t(z_t - C_t\, \bar{\mu}_t)$
  - $\Sigma_t = \bar{\Sigma}_t - K_t\, C_t\, \bar{\Sigma}_t$
  - $K_t = \bar{\Sigma}_t\,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

Kalman Filter Facts

- If the distribution is not Gaussian, the Kalman filter is the minimum variance linear estimator
  - The noise must be uncorrelated with the initial state x_0
- The variance never increases due to receiving a measurement
- The variance update is independent of the measurement realization
- Prediction and update can happen in arbitrary order as long as the control input and measurements are temporally sorted

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 (to handle nonlinear systems) Kalman Filter

Assumptions for EKF

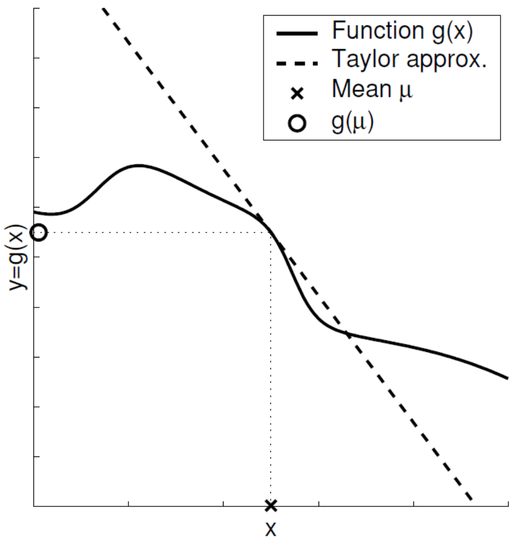
- The prior state of the robot is represented by a Gaussian distribution
  - $p(x_0) \sim N(\mu_0, \Sigma_0)$
- The continuous time process model is:
  - $\dot{x} = f(x, u, n)$
  - $n_t \sim N(0, Q_t)$ is Gaussian white noise
- The measurement model is:
  - $z = g(x, v)$
  - $v_t \sim N(0, R_t)$ is Gaussian 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 measurement 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:
  - $\bar{\mu}_t = \mu_{t-1} + \delta t\, f(\mu_{t-1}, u_t, 0)$
  - $\bar{\Sigma}_t = F_t\, \Sigma_{t-1}\, F_t^T + V_t\, Q_t\, V_t^T$

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

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

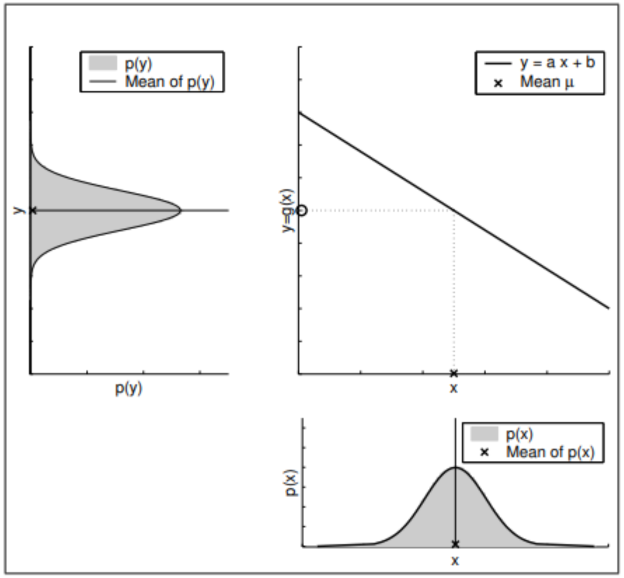
$$\left.\begin{array}{l} F_t = I + \delta t\, A_t \\ V_t = \delta t\, U_t \end{array}\right\} \text{Discretization}$$

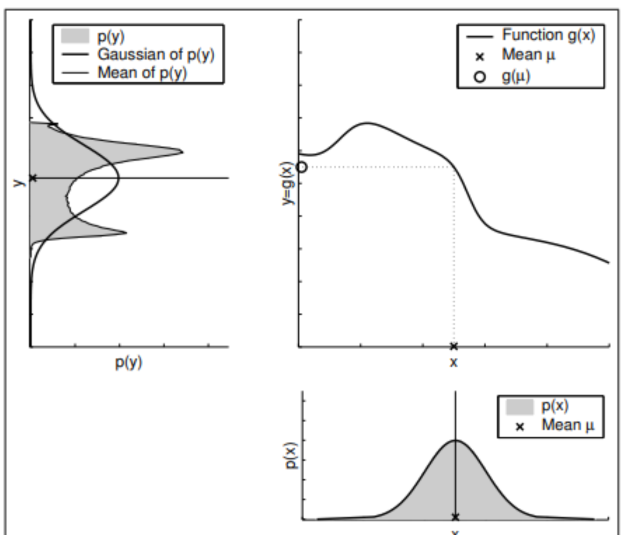
- Update step:
  - $\mu_t = \bar{\mu}_t + K_t\, (z_t - g(\bar{\mu}_t, 0))$
  - $\Sigma_t = \bar{\Sigma}_t - K_t\, C_t\, \bar{\Sigma}_t$
  - $K_t = \bar{\Sigma}_t\, C_t^T\, (C_t\, \bar{\Sigma}_t\, C_t^T + W_t\, R\, W_t^T)^{-1}$

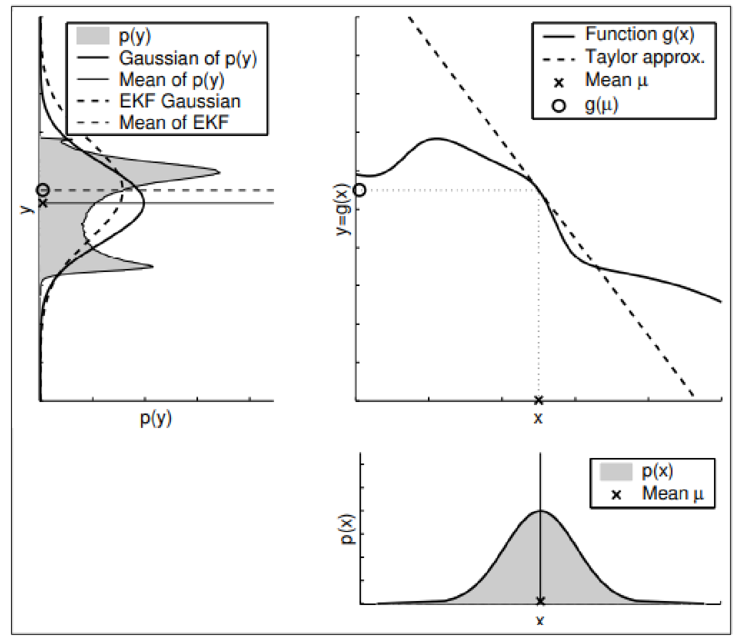
$$\left.\begin{array}{l} z_t = g(x_t, v_t) \\ v_t \sim N(0, R_t) \end{array}\right\} \text{Assumptions}$$

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

More on Linearization

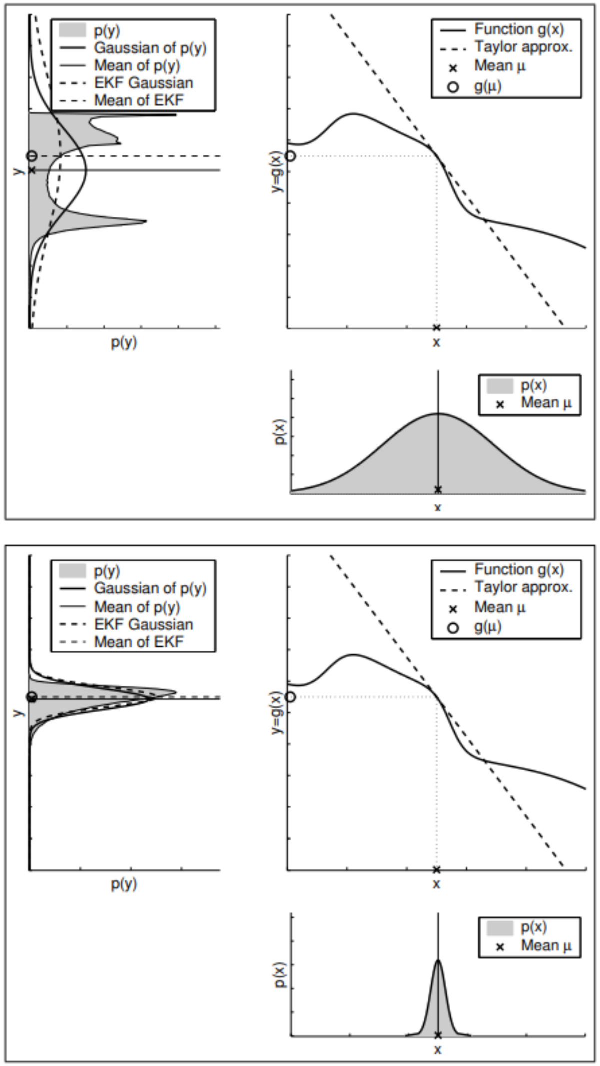




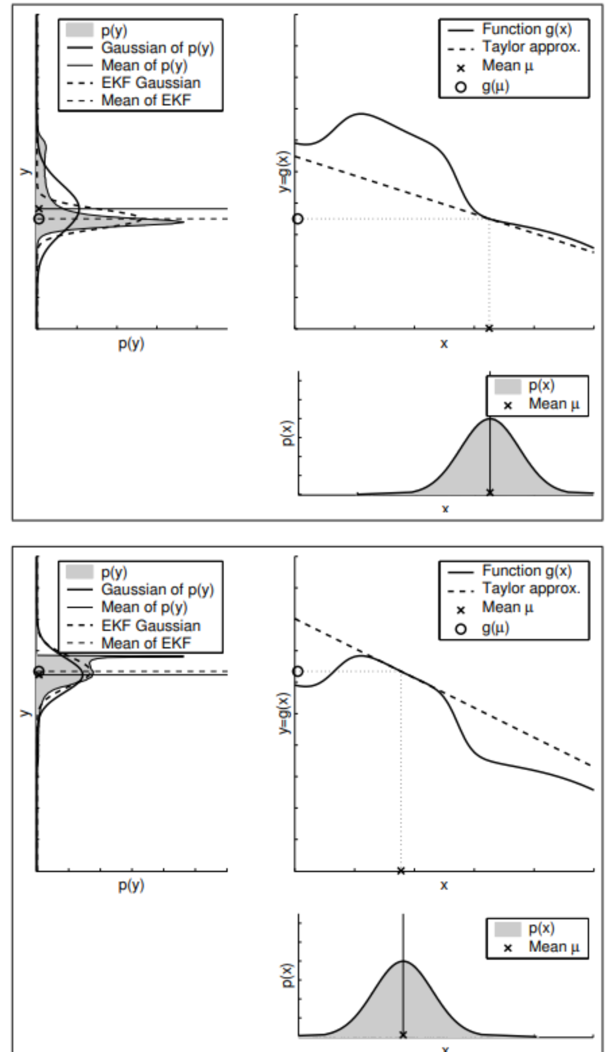


EKF aims to generate Gaussian Approximation of the random variable under nonlinear function

More on Linearization



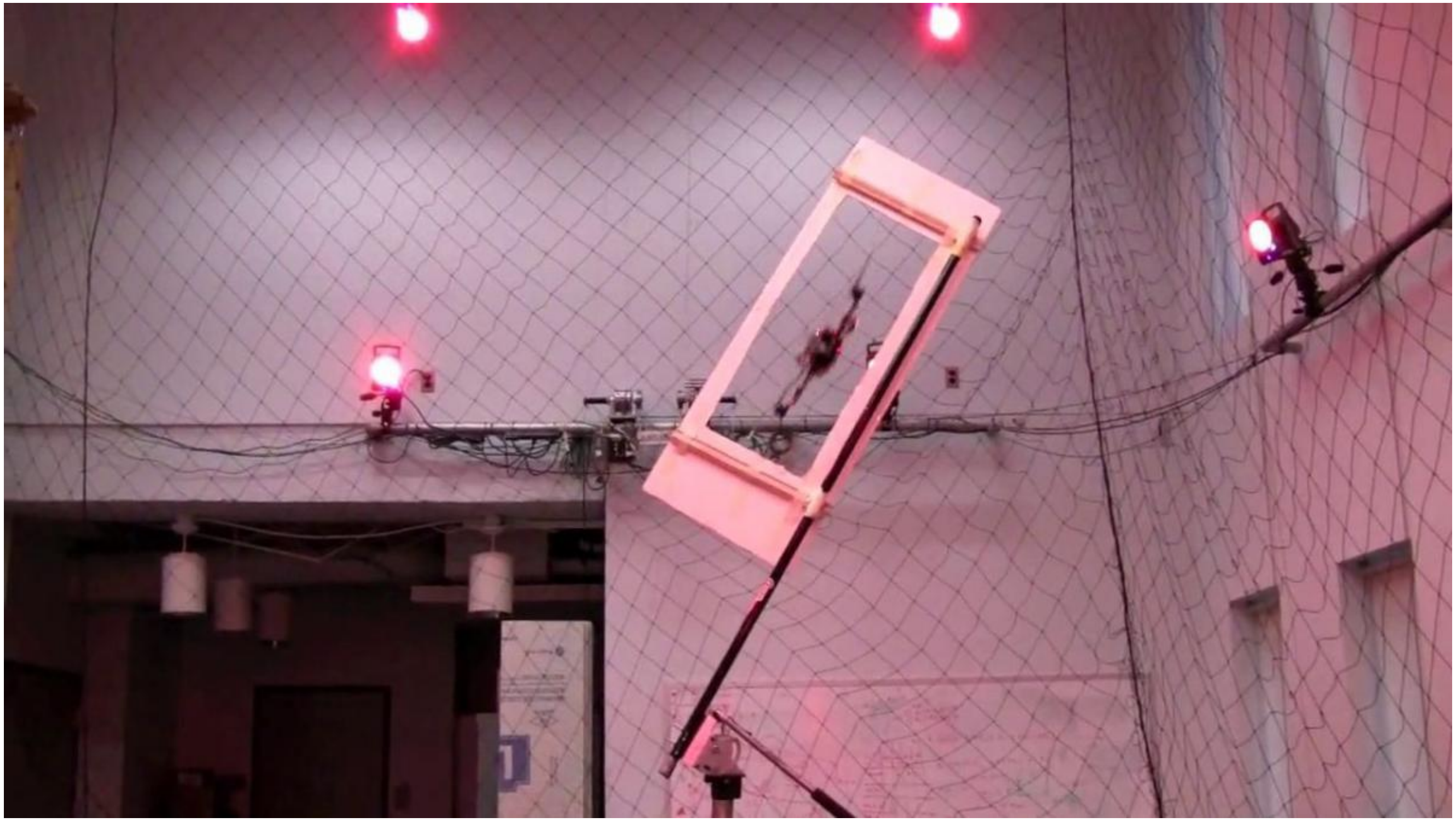
Different uncertainties of the random variable



Different nonlinearities of the function

Example Problem

Quadrotor with a Good Velocity Sensor

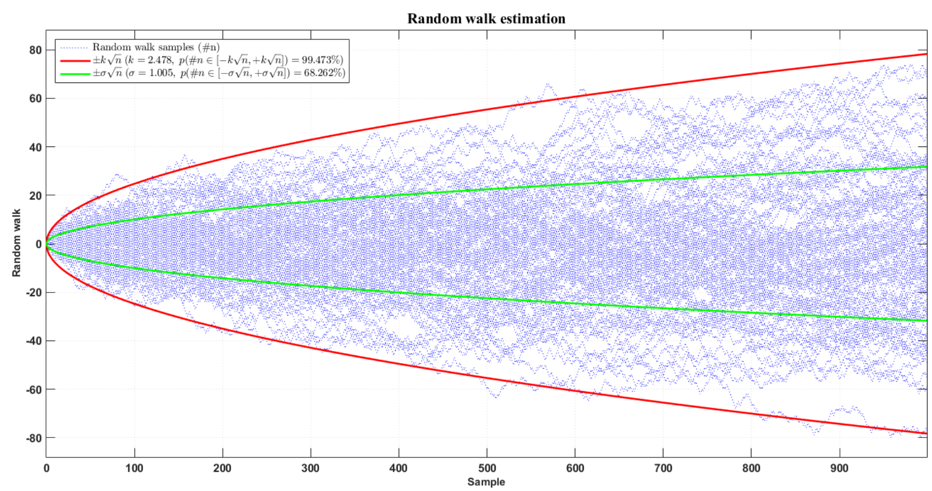


State

- Can 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
  - $\mathbf{q} = [\phi, \theta, \psi]^T = [\text{roll}, \text{pitch}, \text{yaw}]^T$
  - $\boldsymbol{R} = \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\psi c\theta s\phi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix}$
  - Quaternion-based rotation representation is also possible and is better, refer to your L2 supplementary slides for details

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
  - $\boldsymbol{\omega}_m = \boldsymbol{\omega} + \mathbf{b}_g + \mathbf{n}_g$
- **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

- $\boldsymbol{\omega}_m$ is in the body frame, $\mathbf{q}$ is in the world frame
- **Recall:** the angular velocity in the body frame is given by
- $\boldsymbol{\omega} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} c\theta & 0 & -c\phi s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = G(\mathbf{q})\dot{\mathbf{q}}$
- Use mocap and gyroscope measurements as process input $u = [\mathbf{v}_m, \boldsymbol{\omega}_m]$
- Process model:
- $\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{v}_m - \mathbf{n}_v \\ G(\mathbf{x}_2)^{-1}(\boldsymbol{\omega}_m - \mathbf{x}_3 - \mathbf{n}_g) \\ \mathbf{n}_{bg} \end{bmatrix}$
- How to obtain the covariance matrix for $\mathbf{n}_g$ and $\mathbf{n}_{bg}$?
  - Recall the definition of diagnostic and causal information (L8)
  - Sensor characterization using specialized setup

Measurement Model

- Use a camera to measure the pose of the robot
- Use theory of projective geometry
- Can estimate the position and orientation of the robot using a minimum of 4 features on the ground plane, e.g., utilizing markers
  - Can recover $\mathbf{q}$ from the rotation matrix $\boldsymbol{R}$
- $\mathbf{z} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \end{bmatrix} + \mathbf{v}$

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

  $= C\,\mathbf{x} + \mathbf{v}$
- How to obtain the covariance matrix for $\mathbf{v}$?
  - Utilize your evaluation results from Project 2 Phase 1 w.r.t. mocap

Quadrotor with a Good Acceleration Sensor

State

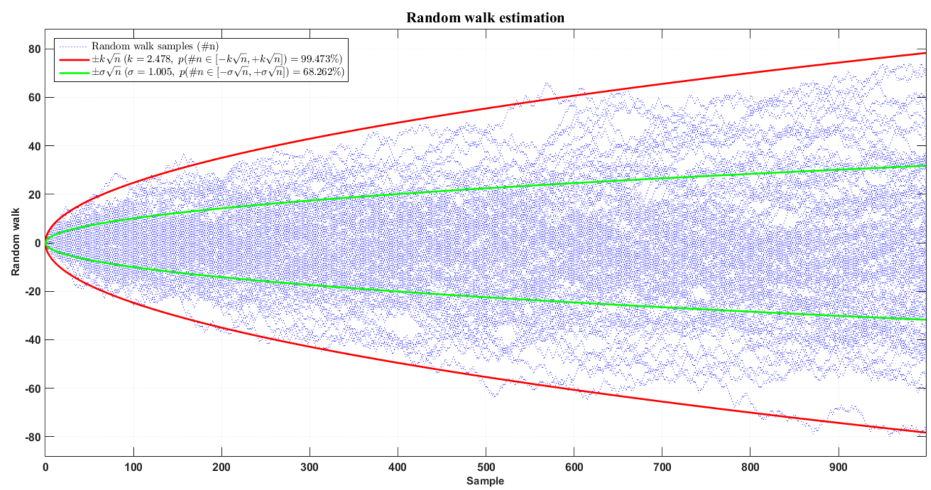
- Can estimate the commanded linear acceleration and angular velocity using the IMU

- $$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \\ \mathbf{x}_5 \end{bmatrix} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \dot{\mathbf{p}} \\ \mathbf{b}_g \\ \mathbf{b}_a \end{bmatrix} = \begin{bmatrix} \text{position} \\ \text{orientation} \\ \text{linear velocity} \\ \text{gyroscope bias} \\ \text{accelerometer bias} \end{bmatrix} \in \mathbf{R}^{15}$$

- Use Z-X-Y Euler angle parameterization of $SO(3)$ for orientation
  - $\mathbf{q} = [\phi, \theta, \psi]^T = [\text{roll}, \text{pitch}, \text{yaw}]^T$
  - $$\boldsymbol{R} = \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\psi c\theta s\phi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix}$$

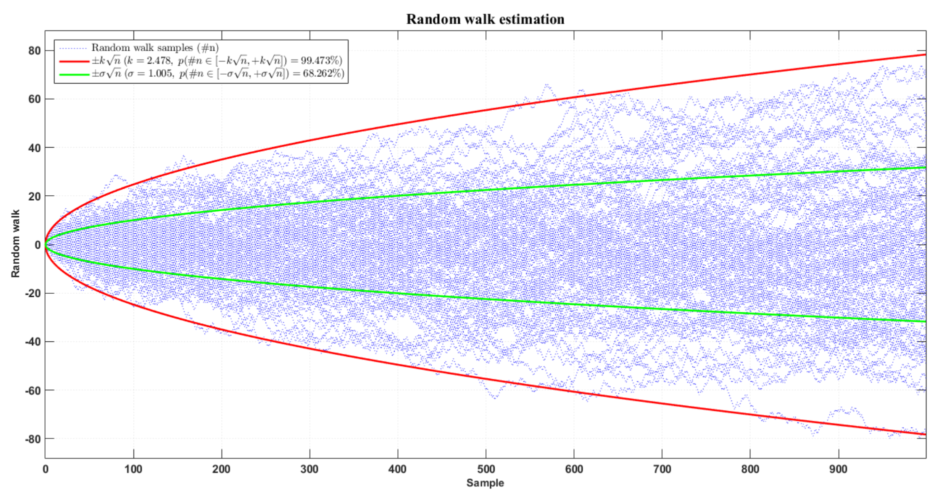
Process Model – Gyroscope

- **Assumption:** the gyroscope gives a noisy estimate of the angular velocity
  - $\boldsymbol{\omega}_m = \boldsymbol{\omega} + \mathbf{b}_g + \mathbf{n}_g$
- **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
  - $\mathbf{a}_m = \boldsymbol{R}(\mathbf{q})^T(\ddot{\mathbf{p}} - \mathbf{g}) + \mathbf{b}_a + \mathbf{n}_a$
- **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:
- $$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{x}_3 \\ G(\mathbf{x}_2)^{-1}(\omega_m - \mathbf{x}_4 - \mathbf{n}_g) \\ \mathbf{g} + R(\mathbf{x}_2)(\mathbf{a}_m - \mathbf{x}_5 - \mathbf{n}_a) \\ \mathbf{n}_{bg} \\ \mathbf{n}_{ba} \end{bmatrix}$$

Measurement Model

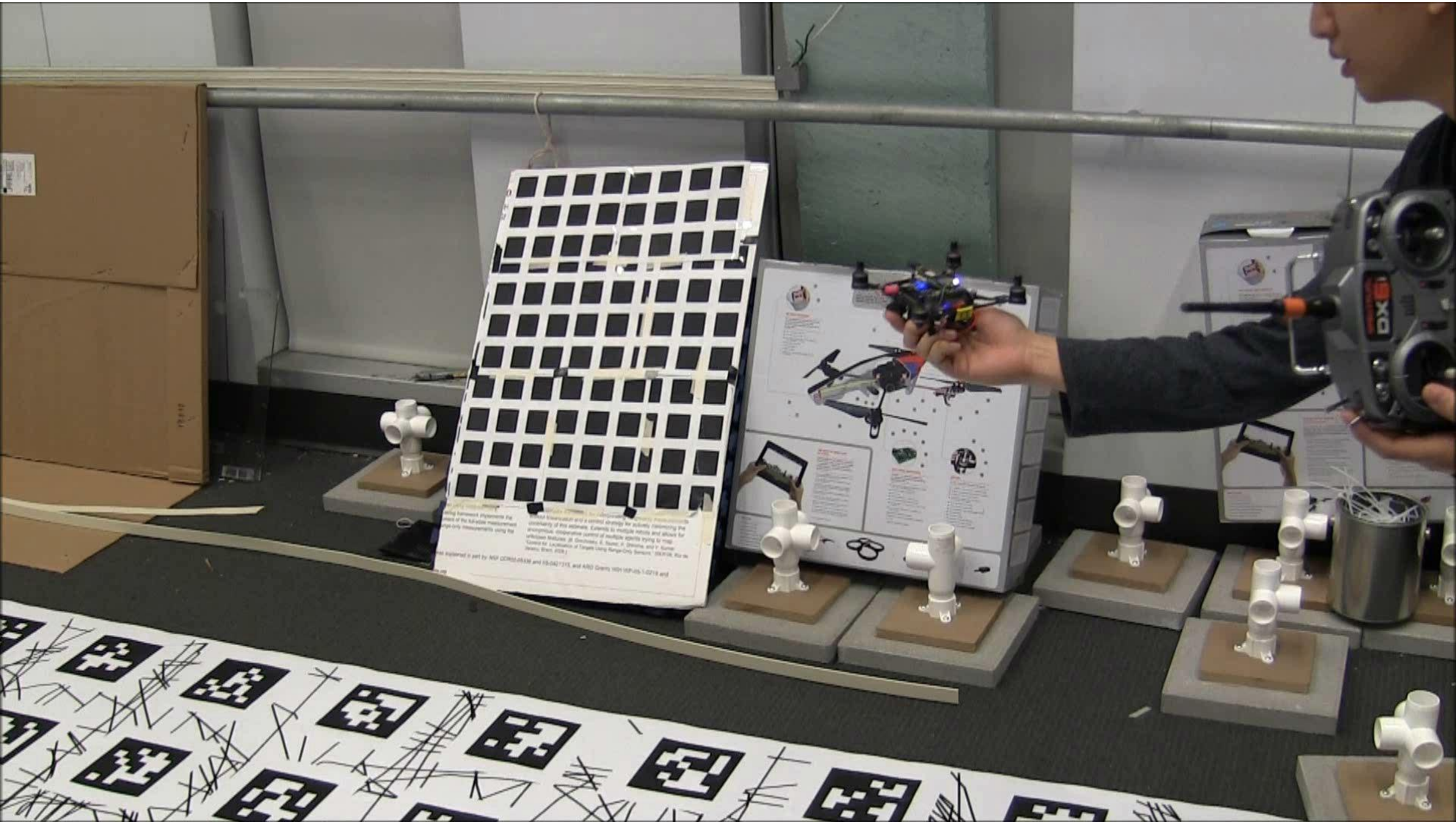
- Use a camera to measure:
  - The pose of the robot (using markers)
- $\mathbf{z} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \end{bmatrix} + \mathbf{v}$

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

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

Measurement Model

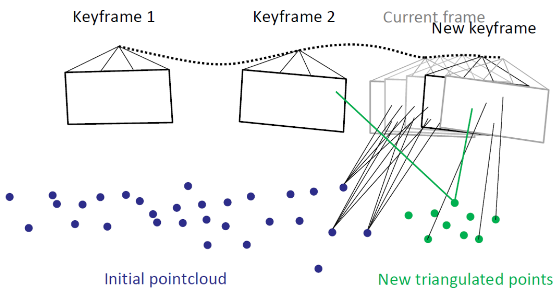
- Use a camera to measure:
  - The pose of the robot (using markers)
  - The body frame linear velocity (using optical flow)
  - Multi-sensor fusion, robust to single sensor failure
  - May also split it into two measurement models
- $$\mathbf{z} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \boldsymbol{R}(\mathbf{q})^T \dot{\mathbf{p}} \end{bmatrix} + \mathbf{v} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \boldsymbol{R}(\mathbf{x}_2)^T \mathbf{x}_3 \end{bmatrix} + \mathbf{v} = g(\mathbf{x}, \mathbf{v})$$



Augmented State EKF for Fusing Relative Measurements

Motivation

- In some situations, only relative state measurements are available.
  - Project 2 Phase 2: keyframe-based visual odomerty
  - Relative pose between current frame and the latest keyframe
- The measurement depends on the current state and a previous state.
- The measurement model is:
  - $z_{t|t_i} = g(x_t, x_{t_i}, v_{t|t_i})$
  - $v_{t|t_i} \sim N(0, R_t)$ is Gaussian white noise
  - Where t_i is the time that the keyframe is generated



- However, this violates the Markov assumption. How to deal with it?

Augmented State and Covariance

- Copy the part of the original state $\mathbf{x} \in \mathbb{R}^n$ affected by the measurements ($\mathbf{x}_{t_i}^i \in \mathbb{R}^{n_i}, n_i < n$) and augment the original state, where t_i is the timestamp of the previous state. There are at total m augmented states.

- The full state vector with i augmented states
  - $$\check{\mathbf{x}} = \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_{t_1}^1 \\ \vdots \\ \mathbf{x}_{t_m}^m \end{bmatrix}$$

- The full covariance matrix with i augmented states
  - $$\check{\boldsymbol{\Sigma}} = \begin{bmatrix} \boldsymbol{\Sigma}^{\mathbf{x}\mathbf{x}} & \boldsymbol{\Sigma}^{\mathbf{x}\mathbf{x}_{t_1}^1} & \cdots & \boldsymbol{\Sigma}^{\mathbf{x}\mathbf{x}_{t_m}^m} \\ \boldsymbol{\Sigma}^{\mathbf{x}_{t_1}^1\mathbf{x}} & \boldsymbol{\Sigma}^{\mathbf{x}_{t_1}^1\mathbf{x}_{t_1}^1} & & \boldsymbol{\Sigma}^{\mathbf{x}_{t_1}^1\mathbf{x}_{t_m}^m} \\ \vdots & & \ddots & \vdots \\ \boldsymbol{\Sigma}^{\mathbf{x}_{t_m}^m\mathbf{x}} & \boldsymbol{\Sigma}^{\mathbf{x}_{t_m}^m\mathbf{x}_{t_1}^1} & \cdots & \boldsymbol{\Sigma}^{\mathbf{x}_{t_m}^m\mathbf{x}_{t_m}^m} \end{bmatrix}$$

State Augmentation and Removal

- Binary selection matrix $\mathbf{B}_i$ to select part of the original state
  - $\mathbf{x}_{t_i}^i = \mathbf{B}_i\mathbf{x}$
- State augmentation operator $\mathbf{M}^+$
  - m augmented states already exists, adding the $m+1$ augmented state
  - $\mathbf{M}^+ = \begin{bmatrix} \mathbf{I}_n & \mathbf{0}_{n\times\sum_{k=1}^m n_k} \\ \mathbf{0}_{\sum_{k=1}^m n_k\times n} & \mathbf{I}_{\sum_{k=1}^m n_k} \\ \mathbf{B}_{m+1} & \mathbf{0}_{n_{m+1}\times\sum_{k=1}^m n_k} \end{bmatrix}$
- State removal operator $\mathbf{M}^-$ of an augmented state $\mathbf{x}_{t_j}^j$
  - $\mathbf{M}^- = \begin{bmatrix} \mathbf{I}_{\boldsymbol{a}} & \mathbf{0}_{a\times n_j} & \mathbf{0}_{a\times b} \\ \mathbf{0}_{b\times a} & \mathbf{0}_{b\times n_j} & \mathbf{I}_{\boldsymbol{b}} \end{bmatrix}$
  - $a = n + \sum_{k=1}^{j-1} n_k$
  - $\mathrm{b} = \sum_{k=j+1}^{m} n_k$
- The updated state vector and covariance matrix
  - $\check{\mathbf{x}}^\pm = \mathbf{M}^\pm\check{\mathbf{x}}$
  - $\check{\boldsymbol{\Sigma}}^\pm = \mathbf{M}^\pm\check{\boldsymbol{\Sigma}}\mathbf{M}^{\pm T}$

Prediction for Augmented State EKF

- For the system
  - $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{n})$
  - $\boldsymbol{n}_t \sim N(\mathbf{0}, \boldsymbol{Q}_t)$ is Gaussian white noise

- Recall the prediction of the original EKF:
  - $\bar{\mathbf{x}}_t = \mathbf{x}_{t-1} + \delta t\, f(\mathbf{x}_{t-1}, \boldsymbol{u}_t, \mathbf{0})$
  - $\bar{\boldsymbol{\Sigma}}_t = \boldsymbol{F}_t \boldsymbol{\Sigma}_{t-1} \boldsymbol{F}_t^T + \boldsymbol{V}_t\, \boldsymbol{Q}_t\, \boldsymbol{V}_t^T$

Prediction for Augmented State EKF

- Prediction only affect the main state, by separating the main state and the augmented states:
  - $\check{\mathbf{x}}_{t-1} = \begin{bmatrix} \mathbf{x}_{t-1} \\ \mathbf{x}_{t-1}^{\mathrm{Aug}} \end{bmatrix}$
  - $\check{\boldsymbol{\Sigma}}_{t-1} = \begin{bmatrix} \boldsymbol{\Sigma}_{t-1}^{\mathbf{xx}} & \boldsymbol{\Sigma}_{t-1}^{\mathbf{x}\mathbf{x}^{\mathrm{Aug}}} \\ \boldsymbol{\Sigma}_{t-1}^{\mathbf{x}^{\mathrm{Aug}}\mathbf{x}} & \boldsymbol{\Sigma}_{t-1}^{\mathbf{x}^{\mathrm{Aug}}\mathbf{x}^{\mathrm{Aug}}} \end{bmatrix}$

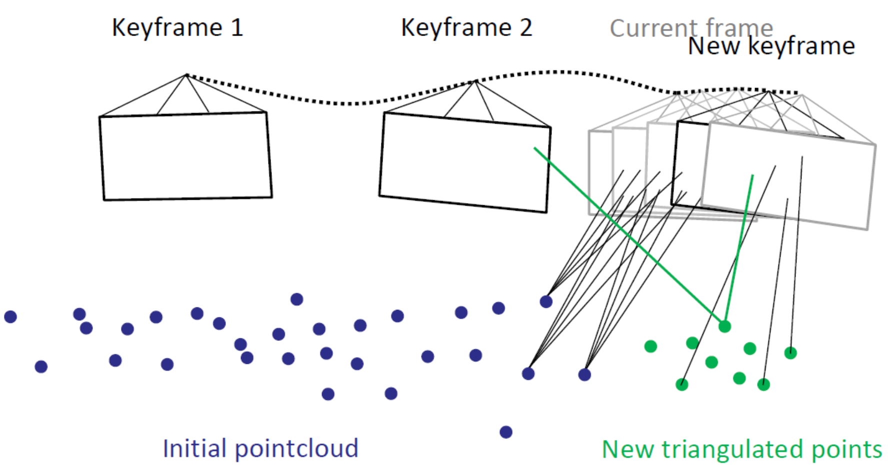
- Using the result of the prediction of the main state:
  - $\bar{\check{\mathbf{x}}}_t = \begin{bmatrix} \bar{\mathbf{x}}_t \\ \mathbf{x}_{t-1}^{\mathrm{Aug}} \end{bmatrix}$ (augmented states remain unchanged during prediction)
  - $\bar{\check{\boldsymbol{\Sigma}}}_t = \begin{bmatrix} \bar{\boldsymbol{\Sigma}}_t^{\mathbf{xx}} & \boldsymbol{F}_t \boldsymbol{\Sigma}_{t-1}^{\mathbf{x}\mathbf{x}^{\mathrm{Aug}}} \\ \boldsymbol{\Sigma}_{t-1}^{\mathbf{x}^{\mathrm{Aug}}\mathbf{x}} \boldsymbol{F}_t^{T} & \boldsymbol{\Sigma}_{t-1}^{\mathbf{x}^{\mathrm{Aug}}\mathbf{x}^{\mathrm{Aug}}} \end{bmatrix}$

Update for Augmented State EKF

- For a relative measurement at t with respect to t_i
  - $\mathbf{z}_{t|t_i} = g(\mathbf{x}_t, \mathbf{x}_{t_i}, \boldsymbol{v}_{t|t_i})$
  - $\boldsymbol{v}_{t|t_i} \sim N(\mathbf{0}, \mathbf{R}_t)$ is Gaussian white noise
- After linearization
  - $\mathbf{z}_{t|t_i} \approx g\left(\bar{\mathbf{x}}_t, \mathbf{x}_{t_i}, 0\right) + \boldsymbol{C}_t(\check{\mathbf{x}}_t - \bar{\check{\mathbf{x}}}_t) + \mathbf{W}_t \boldsymbol{v}_{t|t_i}$
  - $\boldsymbol{C}_t = [\frac{\partial g}{\partial \mathbf{x}_t}\Big|_{\bar{\mathbf{x}}_t}, \mathbf{0}, \frac{\partial g}{\partial \mathbf{x}_{t_i}}\Big|_{\mathbf{x}_{t_i}}, \mathbf{0}]$
- Update as the original EKF
  - $\boldsymbol{K}_t = \bar{\check{\boldsymbol{\Sigma}}}_t \boldsymbol{C}_t^T \left(\boldsymbol{C}_t \bar{\check{\boldsymbol{\Sigma}}}_t \boldsymbol{C}_t^T + \mathbf{W}_t \mathbf{R}_t \mathbf{W}_t^T\right)^{-1}$
  - $\check{\mathbf{x}}_t = \bar{\check{\mathbf{x}}}_t + \boldsymbol{K}_t(\mathbf{z}_{t|t_i} - g(\mathbf{x}_t, \mathbf{x}_{t_i}, \mathbf{0}))$
  - $\check{\boldsymbol{\Sigma}}_t = \bar{\check{\boldsymbol{\Sigma}}}_t - \boldsymbol{K}_t \boldsymbol{C}_t \bar{\check{\boldsymbol{\Sigma}}}_t$

Example Problem

Quadrotor with a Good Acceleration Sensor and Keyframe-based Visual Odometry



State

- To fuse the relative pose measurement from single-keyframe visual odometry, the state is augmented as:
- $\check{\mathbf{x}} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \\ \mathbf{x}_5 \\ \mathbf{x}_{\mathrm{K}_1} \\ \mathbf{x}_{\mathrm{K}_2} \end{bmatrix} = \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \dot{\mathbf{p}} \\ \mathbf{b}_g \\ \mathbf{b}_a \\ \mathbf{p}_{\mathrm{K}} \\ \mathbf{q}_{\mathrm{K}} \end{bmatrix} = \begin{bmatrix} \text{position} \\ \text{orientation} \\ \text{linear velocity} \\ \text{gyroscope bias} \\ \text{accelerometer bias} \\ \text{keyframe position} \\ \text{keyframe orientation} \end{bmatrix} \in \mathbf{R}^{21}$

Process Model

- The process model is:
- $$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{x}_3 \\ G(\mathbf{x}_2)^{-1}\left(\omega_m - \mathbf{x}_4 - \mathbf{n}_g\right) \\ \mathbf{g} + \boldsymbol{R}(\mathbf{x}_2)(\mathbf{a}_m - \mathbf{x}_5 - \mathbf{n}_a) \\ \mathbf{n}_{bg} \\ \mathbf{n}_{ba} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$
- The main state evolve as normal, while the augmented states stay unchanged

Measurement Model

- The measurement model of the relative transform w.r.t. the keyframe (of the visual odometry) is
- $\mathbf{z}_{t|t_i} = \begin{bmatrix} \mathbf{R}(\mathbf{q}_\mathrm{K})^T(\mathbf{p} - \mathbf{p}_\mathrm{K}) \\ \mathrm{euler}(\mathbf{R}(\mathbf{q}_\mathrm{K})^T\mathbf{R}(\mathbf{q})) \end{bmatrix} + \boldsymbol{v}_{t|t_i} = \begin{bmatrix} \mathbf{R}\left(\mathbf{x}_{\mathrm{K}_2}\right)^T(\mathbf{x}_1 - \mathbf{x}_{\mathrm{K}_1}) \\ \mathrm{euler}(\mathbf{R}\left(\mathbf{x}_{\mathrm{K}_2}\right)^T\mathbf{R}(\mathbf{x}_2)) \end{bmatrix} + \boldsymbol{v}_{t|t_i}$
- The relative position $\mathbf{R}(\mathbf{q}_\mathrm{K})^T(\mathbf{p} - \mathbf{p}_\mathrm{K})$ is expressed in the camera frame associated with the keyframe, and so is the relative rotation $\mathbf{R}(\mathbf{q}_\mathrm{K})^T\mathbf{R}(\mathbf{q})$
- The function $\mathrm{euler}(\mathbf{R})$ converts a rotation matrix into Euler angles
- Linearization is left for your own exercise

Changing Keyframe

- When the keyframe is changed, the augmented state of the old keyframe is removed:
  - $\mathbf{M}^{-} = \begin{bmatrix} \mathbf{I}_{15} & \mathbf{0}_{15\times 6} \\ \mathbf{0}_{6\times 15} & \mathbf{0}_{6\times 6} \end{bmatrix}$
  - $\check{\mathbf{x}}^{-} = \mathbf{M}^{-}\check{\mathbf{x}}$
  - $\check{\mathbf{\Sigma}}^{-} = \check{\mathbf{\Sigma}} = \mathbf{M}^{-}\check{\mathbf{\Sigma}}\mathbf{M}^{-T}$

- Then the augmented states of the new keyframe is added:
  - $\mathbf{M}^{+} = \begin{bmatrix} \mathbf{I}_{15} \\ \mathbf{I}_{6\times 15} \end{bmatrix}$
  - $\check{\mathbf{x}}_{\text{new}} = \mathbf{M}^{+}\check{\mathbf{x}}^{-}$
  - $\check{\mathbf{\Sigma}}_{\text{new}} = \check{\mathbf{\Sigma}}^{+} = \mathbf{M}^{+}\check{\mathbf{\Sigma}}^{-}\mathbf{M}^{+T}$

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

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY
電子及計算機工程學系
DEPARTMENT OF ELECTRONIC & COMPUTER ENGINEERING

Extended Kalman Filter

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

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

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

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

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

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

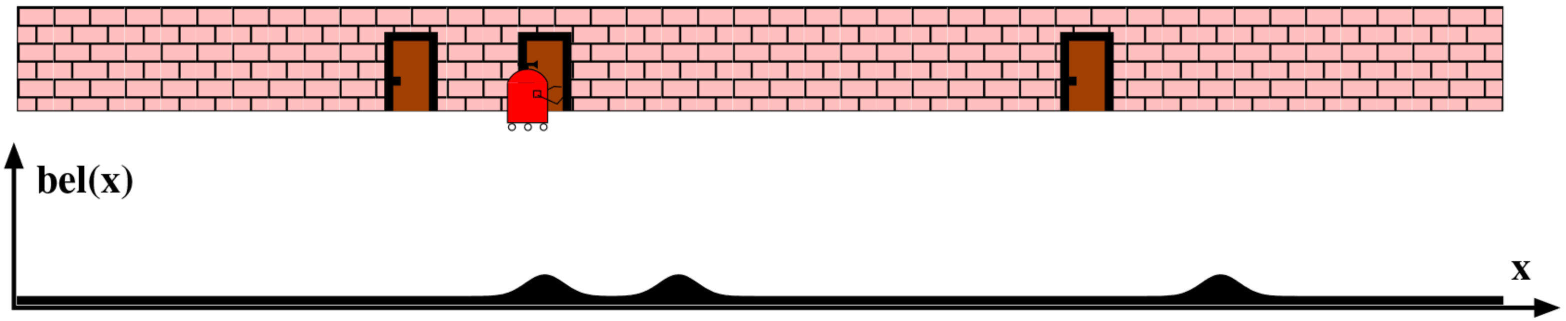
  $$\left.\begin{array}{l} - \ C_t = \left.\frac{\partial g}{\partial x}\right|_{\bar{\mu}_t, 0} \\ - \ W_t = \left.\frac{\partial g}{\partial v}\right|_{\bar{\mu}_t, 0} \end{array}\right\} \text{Linearization}$$

Applications

- EKF is widely used in following applications
  - Pose estimation
  - Parameter estimation
  - Map building
  - Simultaneous localization and mapping (SLAM)
  - Feature tracking
  - Target tracking

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
  - No guarantee of global convergence
  - Unimodal distribution



Particle Filter

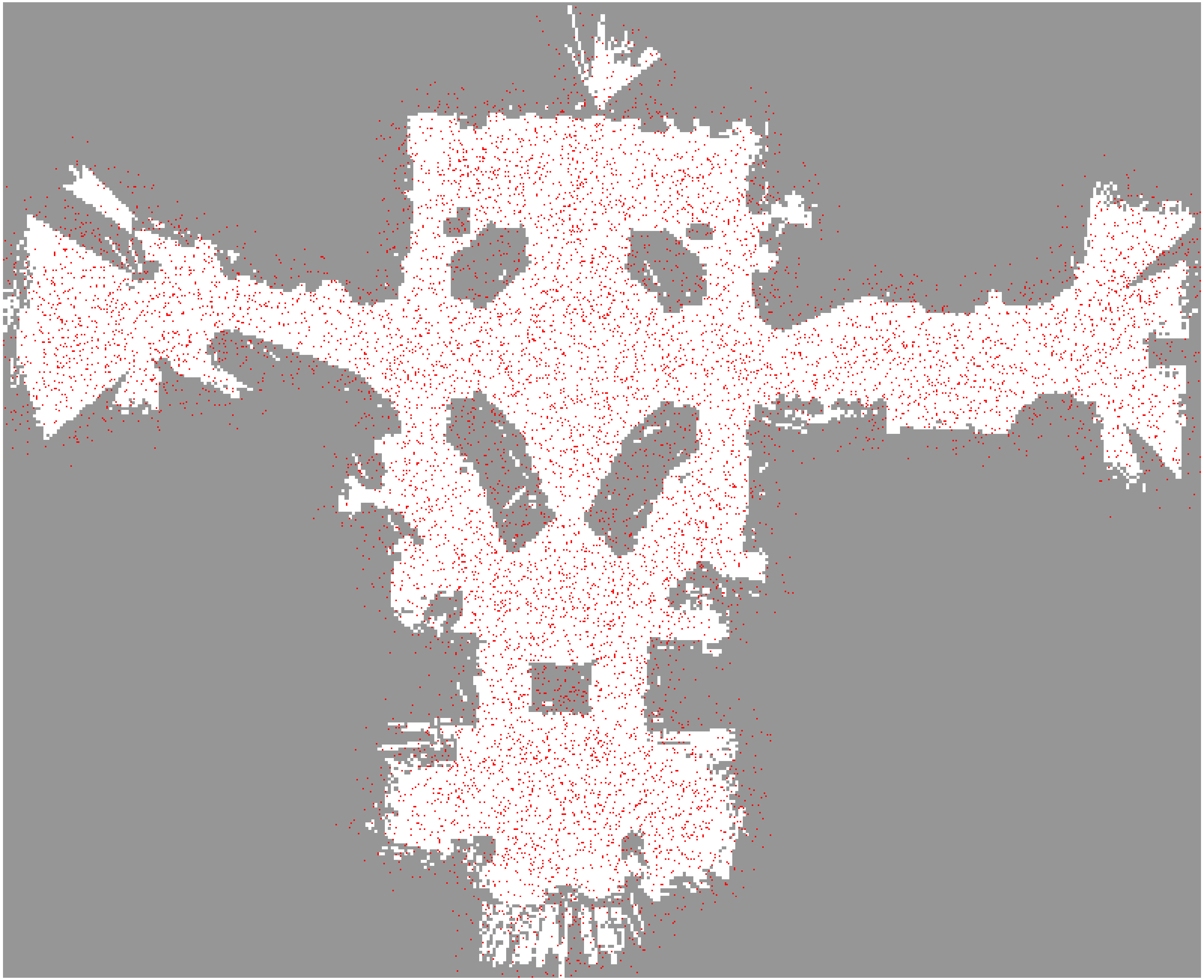
Modified from slides from Sebastian Thrun, Wolfram Burgard, and Dieter Fox

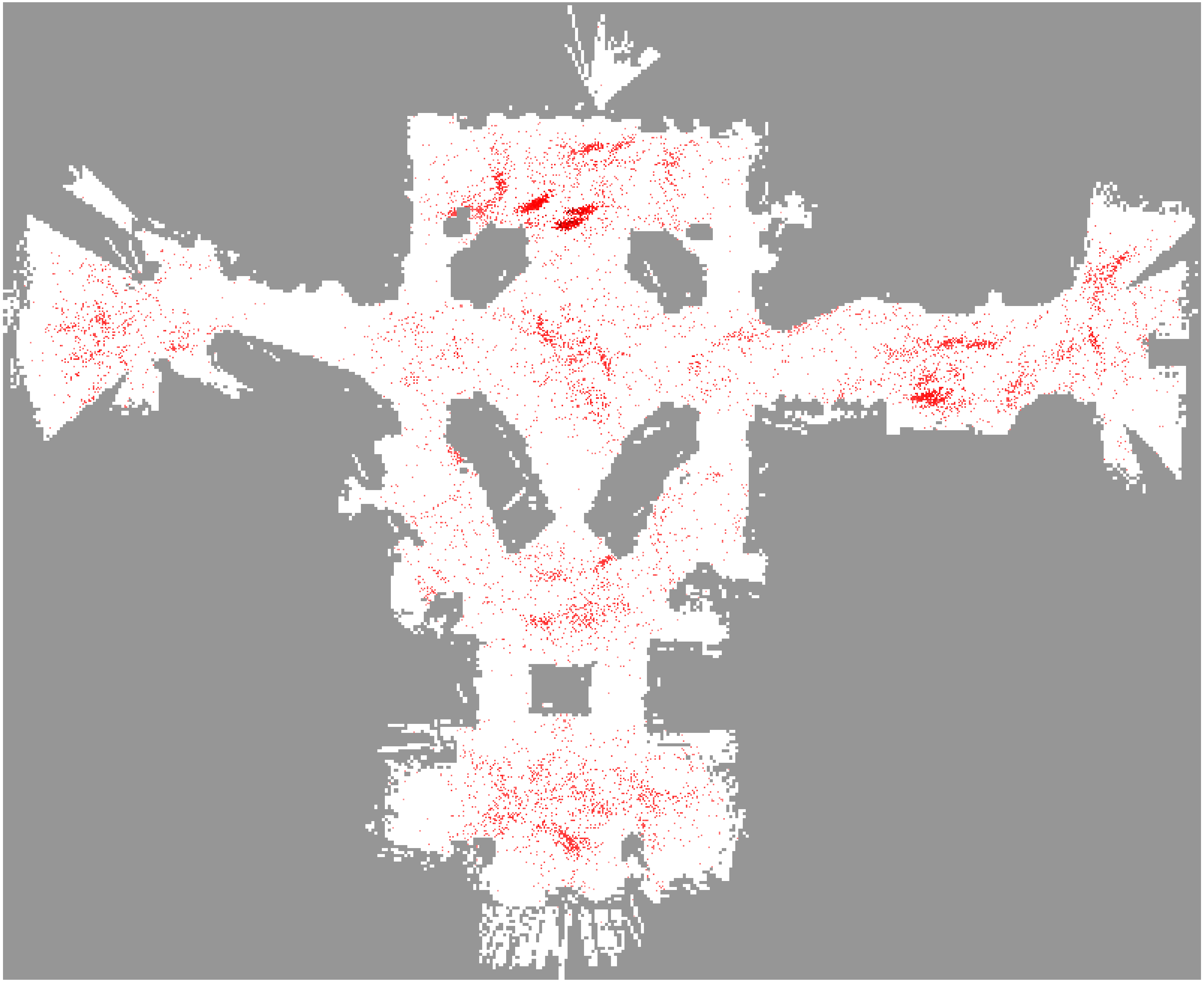
Particle Filters

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

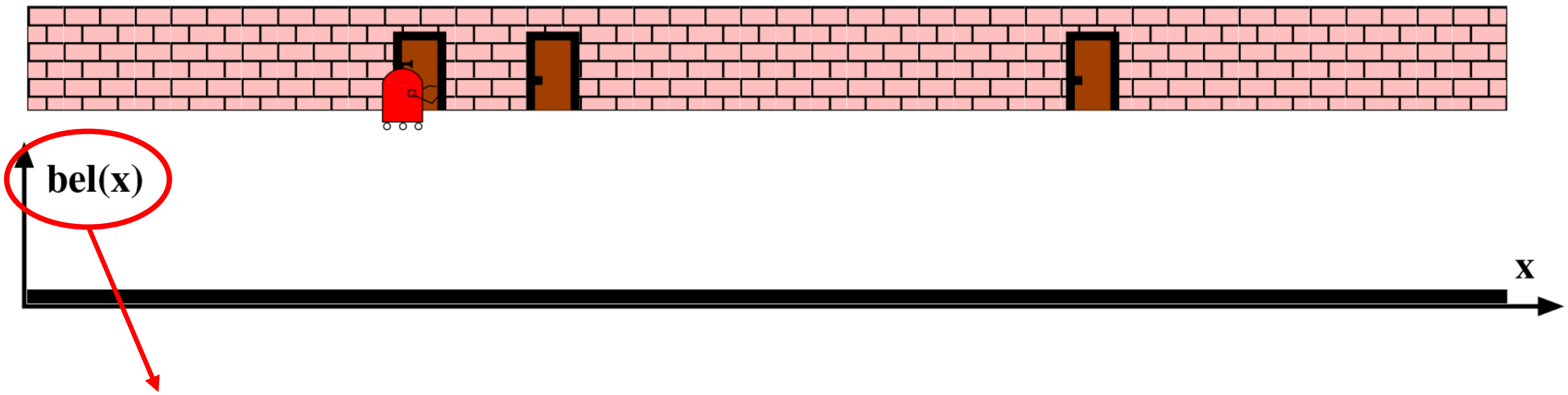
- Monte Carlo filter, Survival of the fittest, Condensation, Bootstrap filter, Particle filter

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



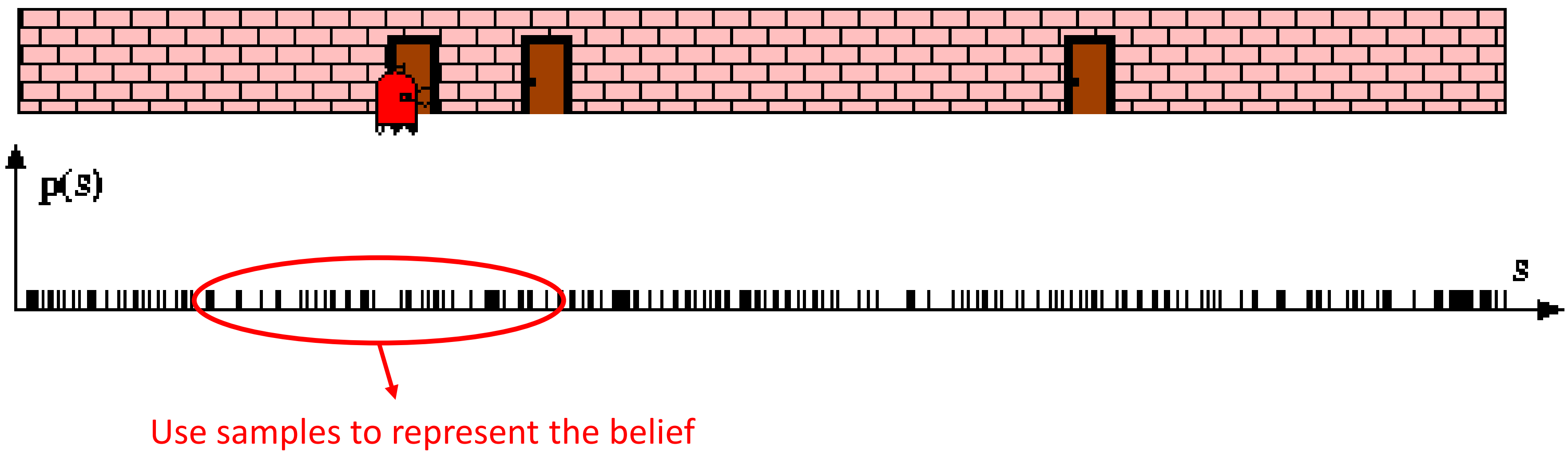


Particle Filters



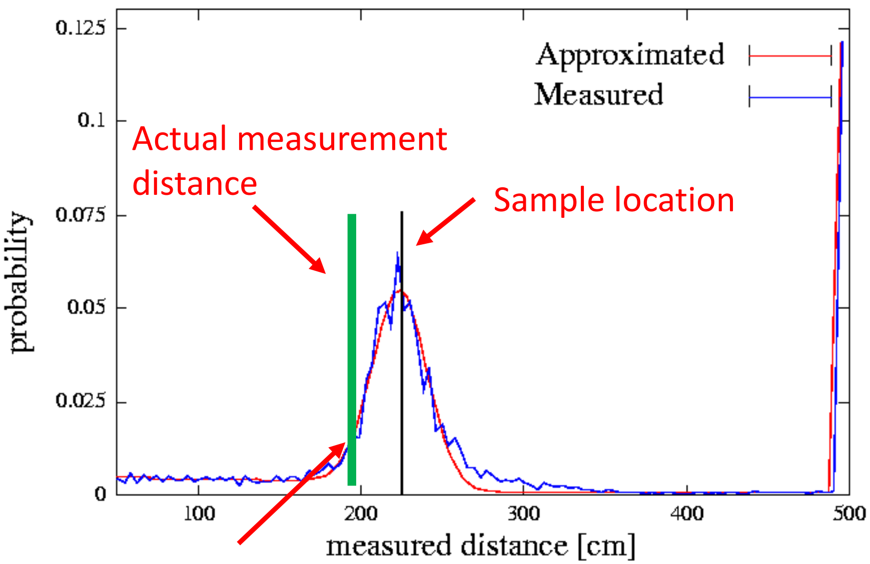
Use "belief" (or probability distribution) to represent robot state (pose)

Particle Filters



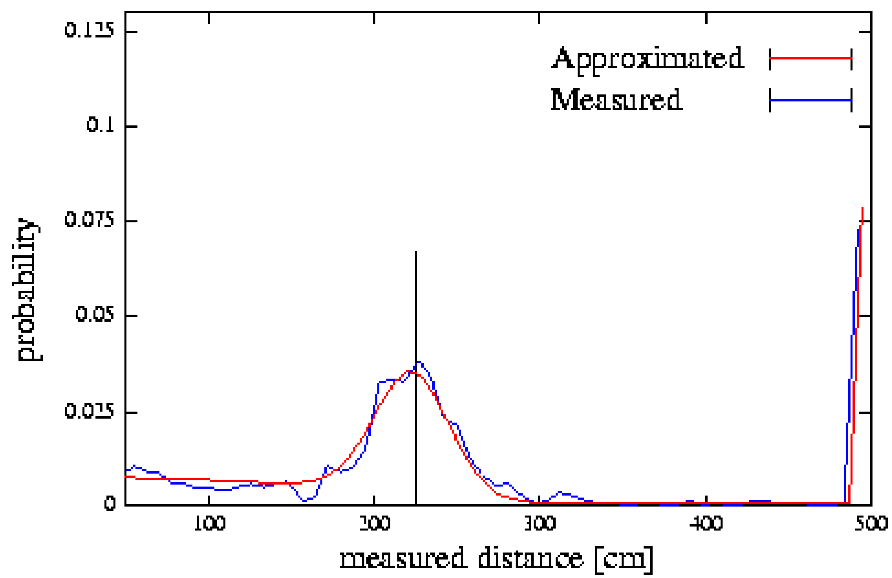
Measurement Model: Door Proximity Sensor

$$p(z|x)$$



The probability of receiving this measurement at this location p(z|x)

Laser sensor

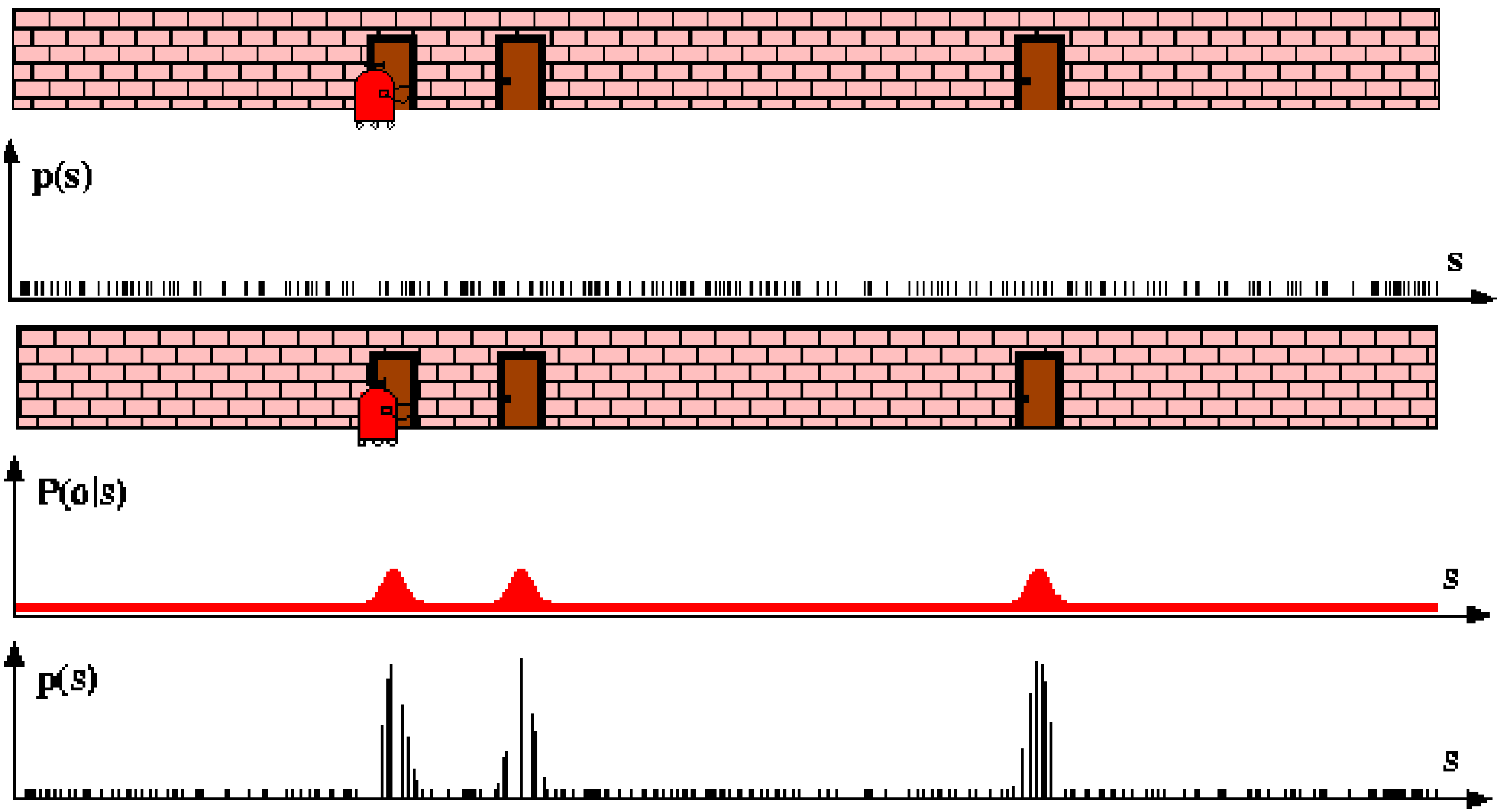


Sonar sensor

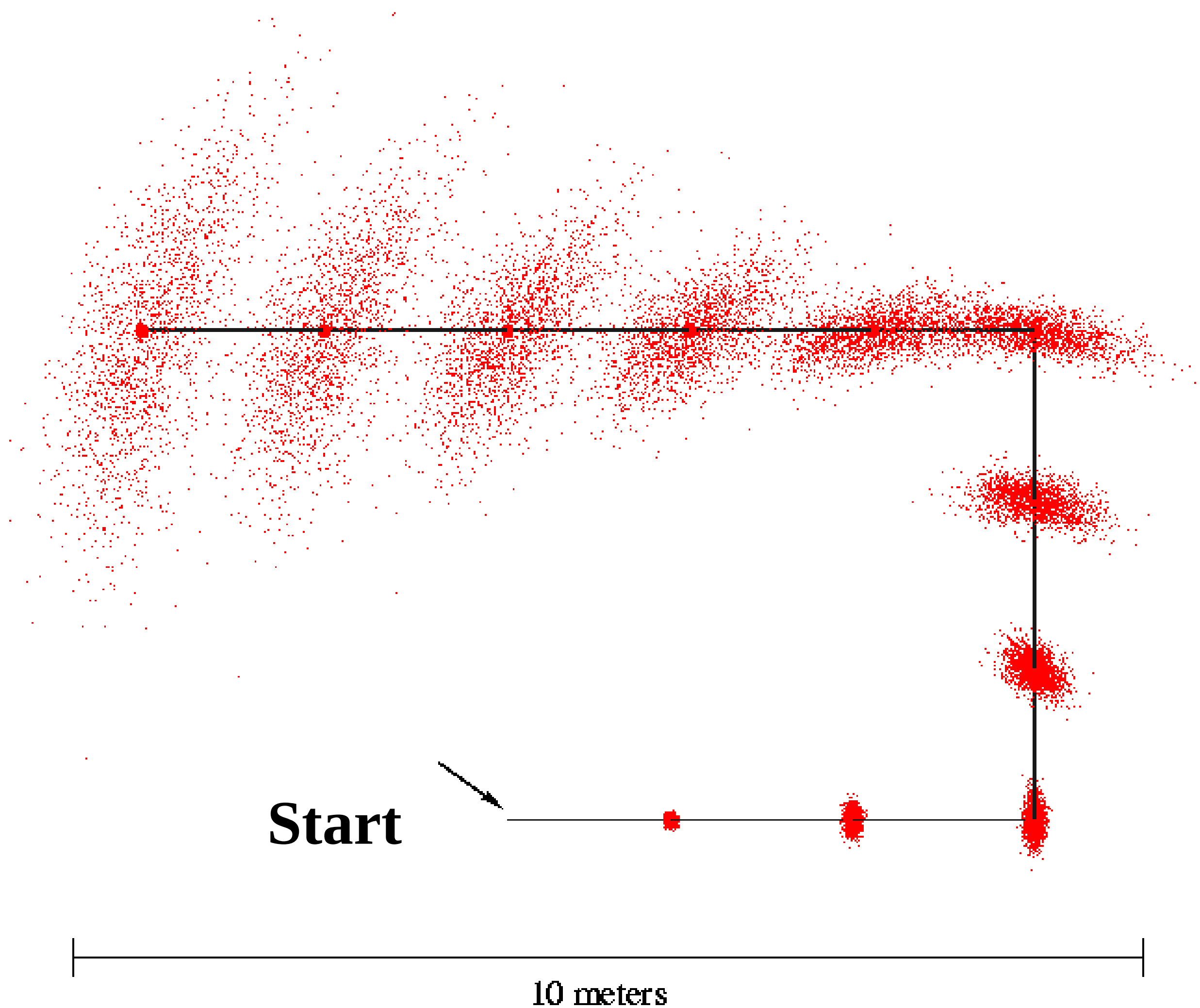
Sensor Information: Importance Sampling

$$w_t \leftarrow \frac{\alpha\, p(z_t|x_t)\, Bel^-(x_t)}{Bel^-(x_t)} = \alpha\, p(z_t|x_t)$$

$$Bel(x_t) \leftarrow \alpha\, p(z_t|x_t)\, Bel^-(x_t)$$

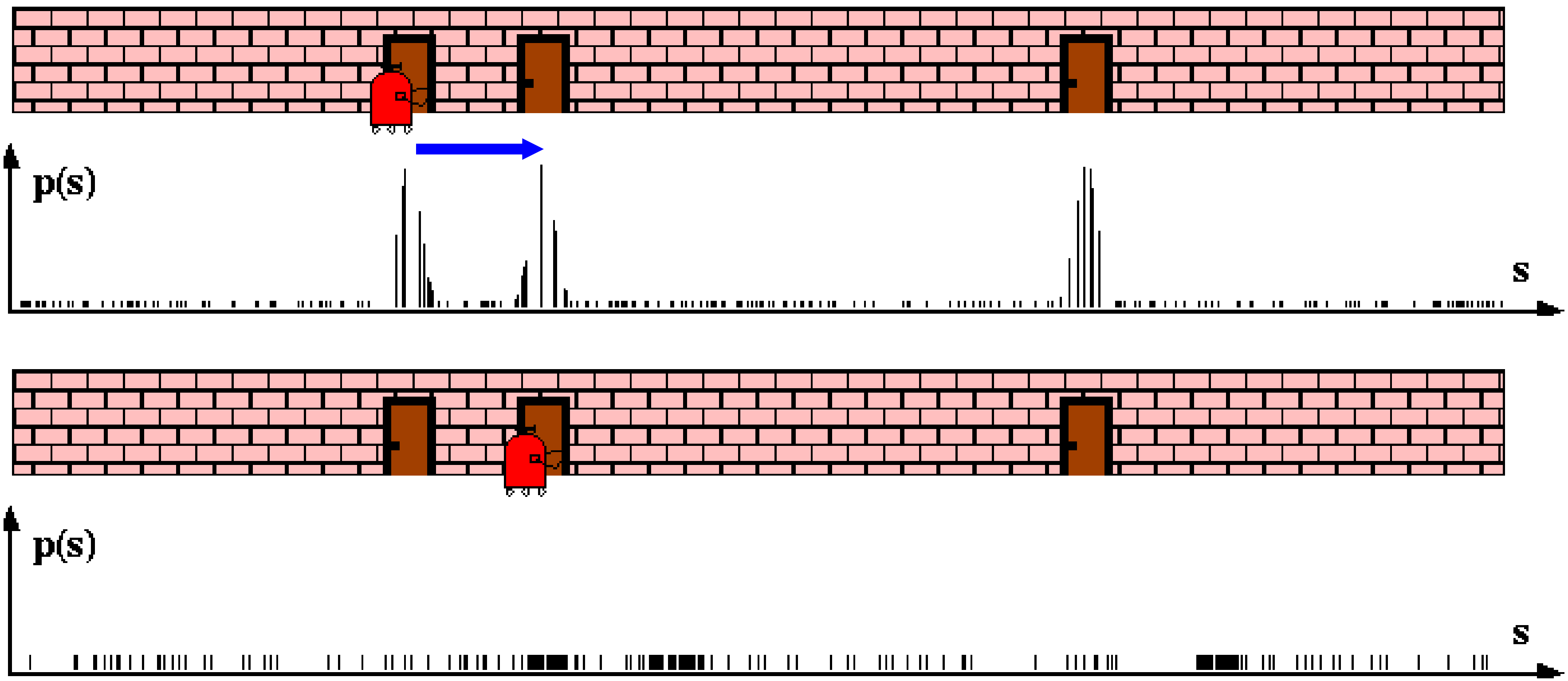


Process (Motion) Model



Resampling and Robot Motion

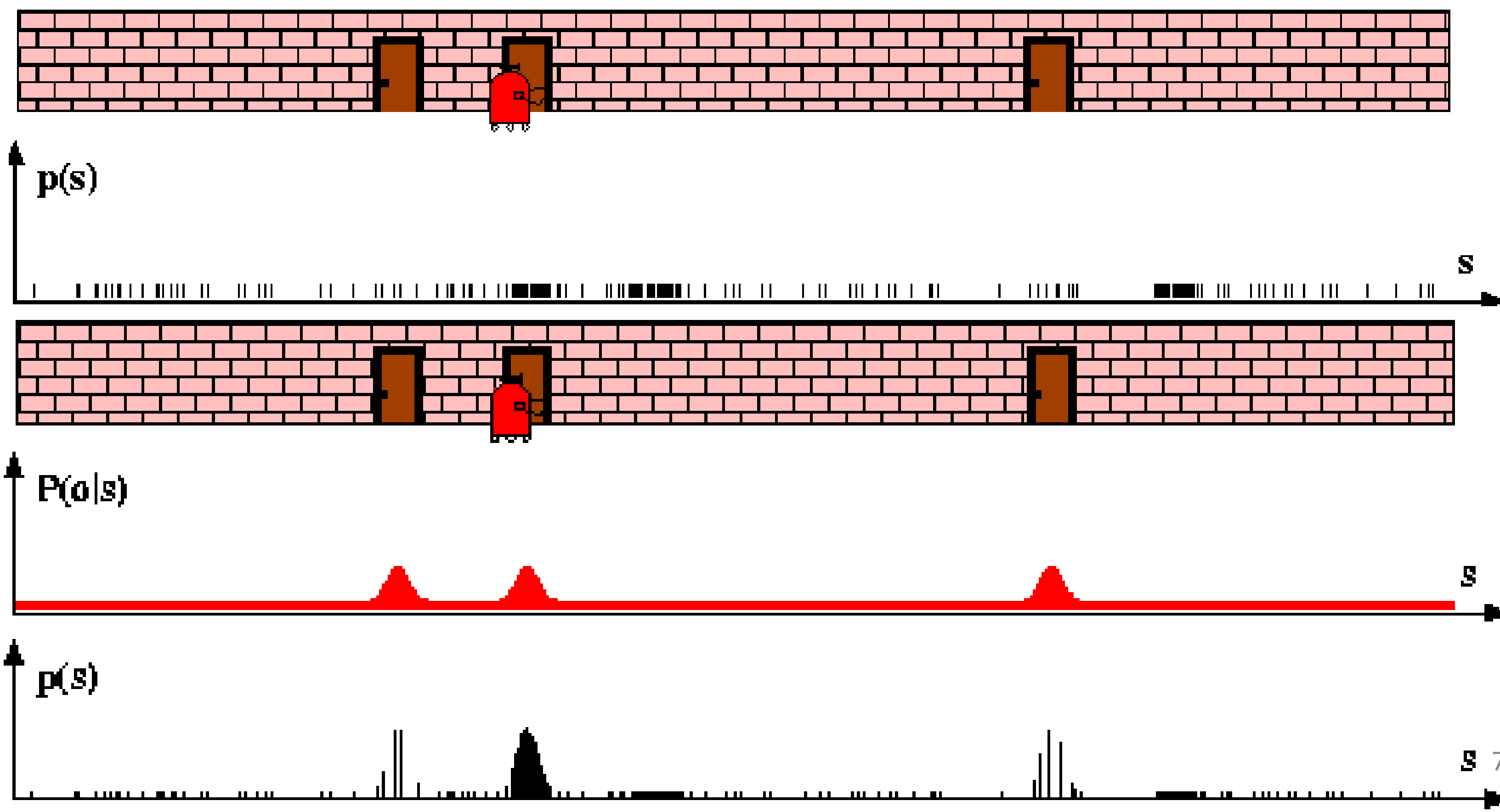
$$Bel^{-}(x_{t+1}) \leftarrow \int p(x_{t+1} | x_{t,} u_{t+1}) Bel(x_t)\, \mathrm{d}x_t$$



Sensor Information: Importance Sampling

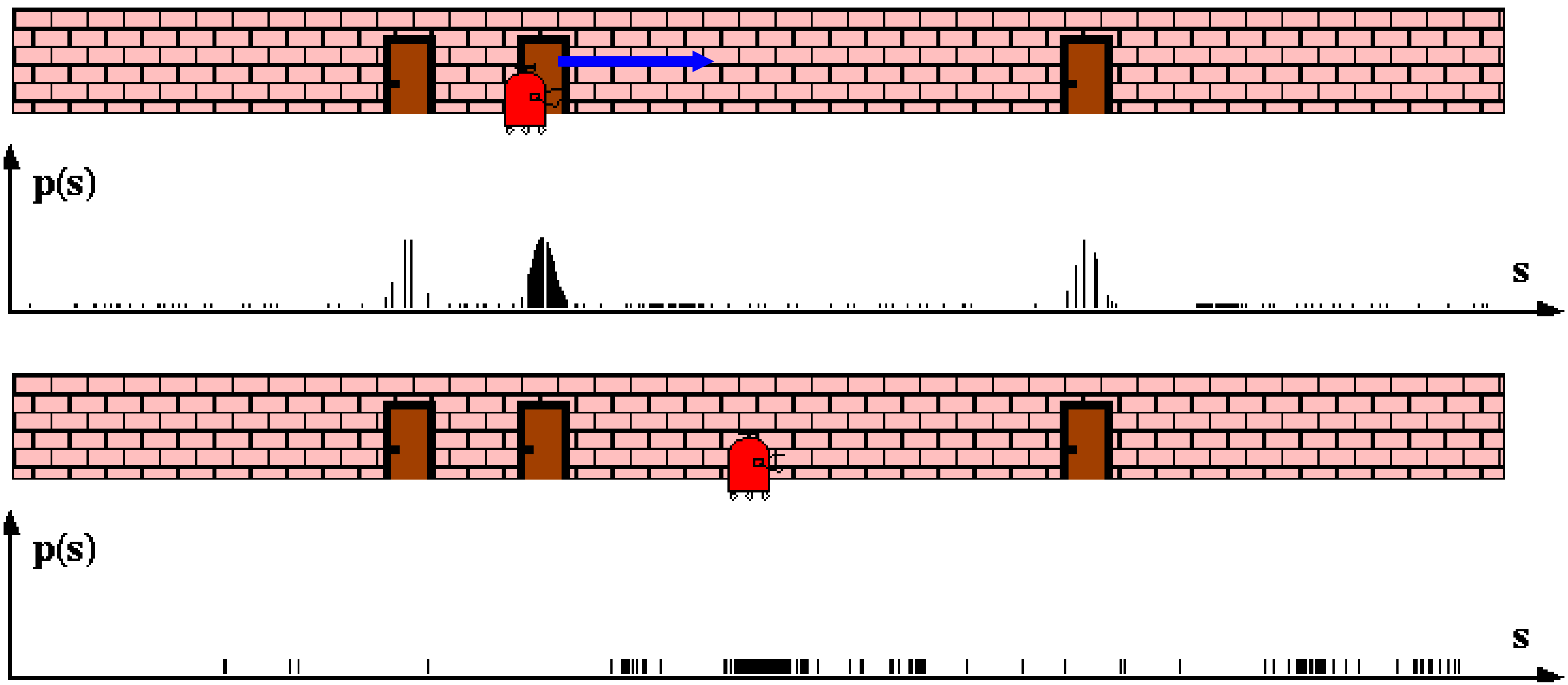
$$w_{t+1} \leftarrow \frac{\alpha\, p(z_{t+1}|x_{t+1})\, Bel^{-}(x_{t+1})}{Bel^{-}(x_{t+1})} = \alpha\, p(z_{t+1}|x_{t+1})$$

$$Bel(x_{t+1}) \leftarrow \alpha\, p(z_{t+1}|x_{t+1})\, Bel^{-}(x_{t+1})$$



Resampling and Robot Motion

$$Bel^{-}(x_{t+2}) \leftarrow \int p(x_{t+2}|x_{t+1,}u_{t+1})Bel(x_{t+1})\,\mathrm{d}x_{t+1}$$



Particle Filter

- Algorithm **particle_filter** (S_{t-1}, u_t, z_t):
- $S_t = \emptyset, \quad \eta = 0$
- **For** $i = 1 \dots n$ *Generate new samples*
  - Sample index $j(i)$ from the discrete distribution given by w_{t-1}
  - Sample x_t^i from $p(x_t | x_{t-1}, u_t)$ using $x_{t-1}^{j(i)}$ and u_t
  - $w_t^i = p(z_t | x_t^i)$ *Compute importance weight*
  - $\eta = \eta + w_t^i$ *Update* normalization factor
  - $S_t = S_t \cup \{< x_t^i, w_t^i >\}$ *Insert*
- **For** $i = 1 \dots n$
  - $w_t^i = w_t^i / \eta$ *Normalize weights*

Particle Filter as Bayes' Filter

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

- **Prediction step:**
- $p(x_t \mid z_{1:t-1}, u_{1:t}) = \int f(x_t \mid x_{t-1}, u_t)\, p(x_{t-1} \mid z_{1:t-1}, u_{1:t-1})\, dx_{t-1}$

draw x_{t-1}^i from $Bel(x_{t-1})$

draw x_t^i from $p(x_t \mid x_{t-1}^i, u_t)$

importance factor for x_t^i: $w_t^i \propto p(z_t \mid x_t)$

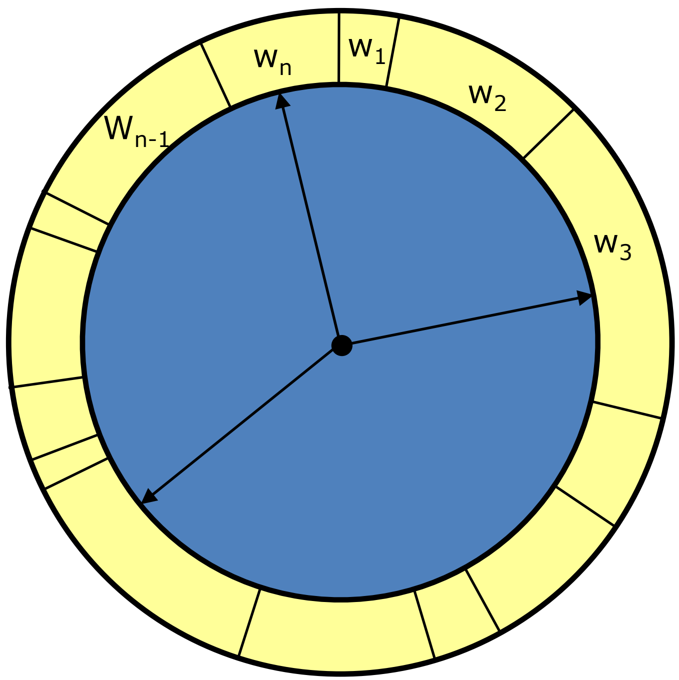
- **Update step:**
- $p(x_t \mid z_{1:t}, u_{1:t}) = \dfrac{g(z_t \mid x_t)\, p(x_t \mid z_{1:t-1}, u_{1:t})}{\int g(z_t \mid x_t')\, p(x_t' \mid z_{1:t-1}, u_{1:t})\, dx_t'}$

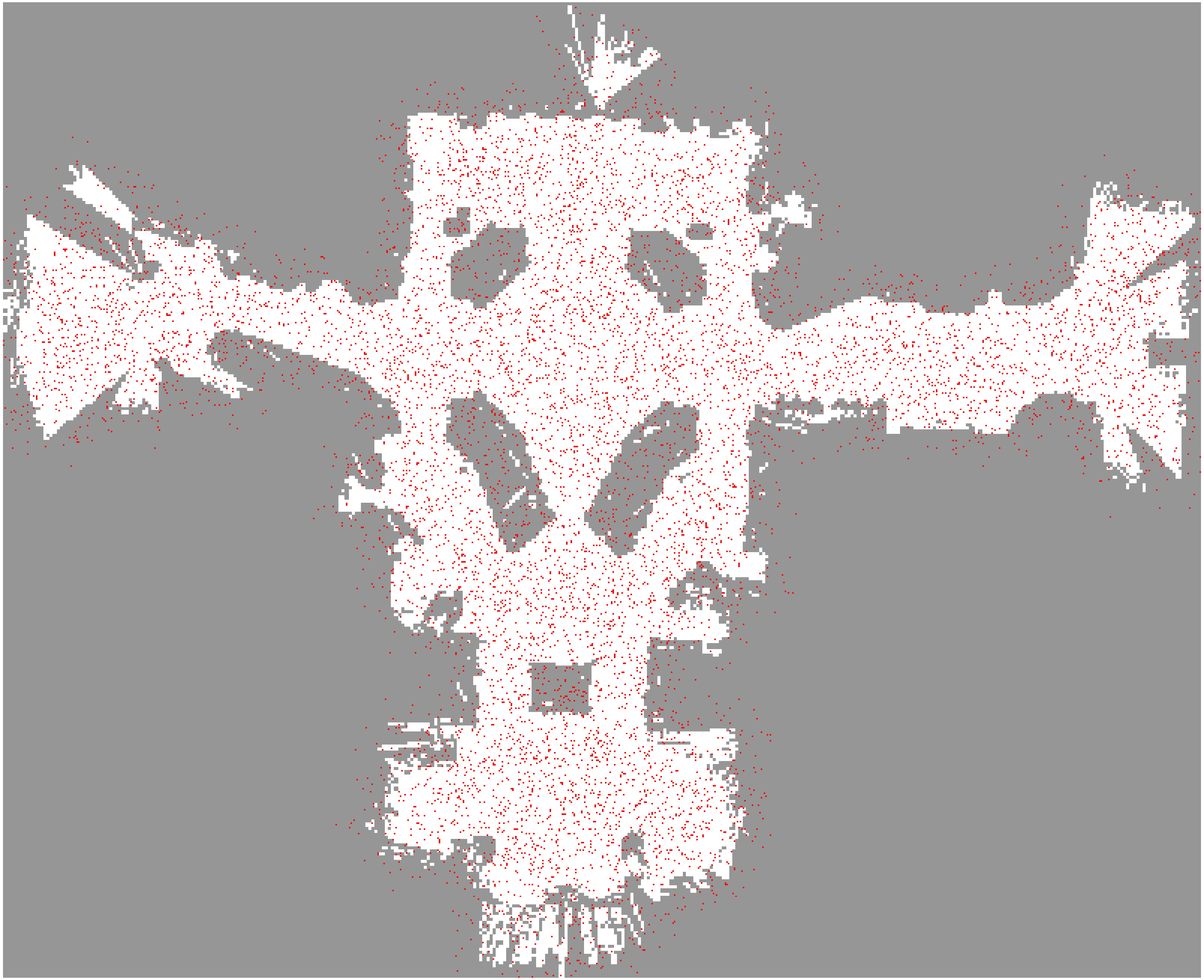
Resampling

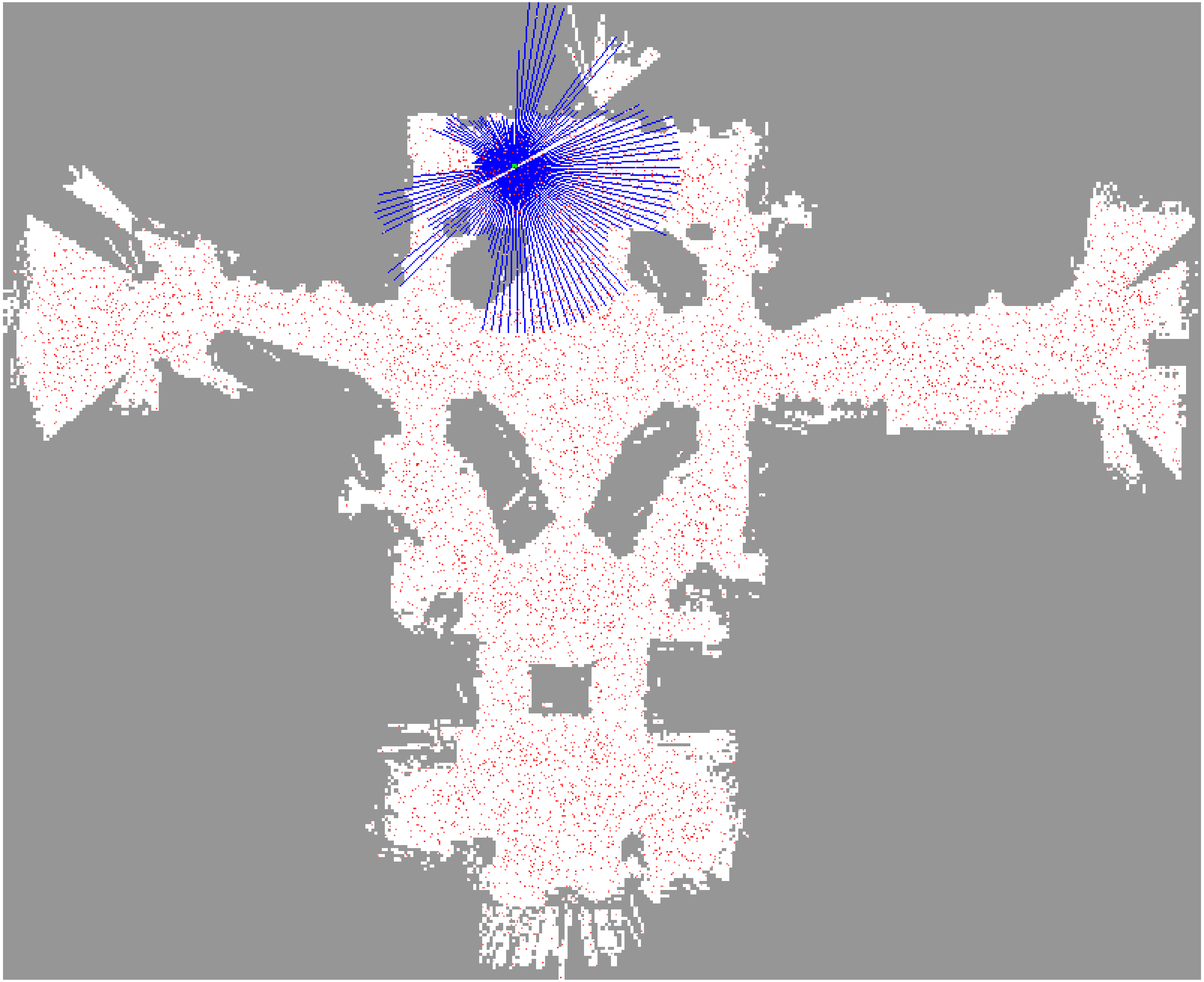
- Algorithm **particle_filter** (S_{t-1}, u_t, z_t):
- $S_t = \emptyset, \quad \eta = 0$
- **For** $i = 1 \dots n$ *Generate new samples*
  - Sample index $j(i)$ from the discrete distribution given by w_{t-1} **How?**
  - Sample x_t^i from $p(x_t | x_{t-1}, u_t)$ using $x_{t-1}^{j(i)}$ and u_t
  - $w_t^i = p(z_t | x_t^i)$ *Compute importance weight*
  - $\eta = \eta + w_t^i$ *Update* normalization factor
  - $S_t = S_t \cup \{< x_t^i, w_t^i >\}$ *Insert*
- **For** $i = 1 \dots n$
  - $w_t^i = w_t^i / \eta$ *Normalize weights*

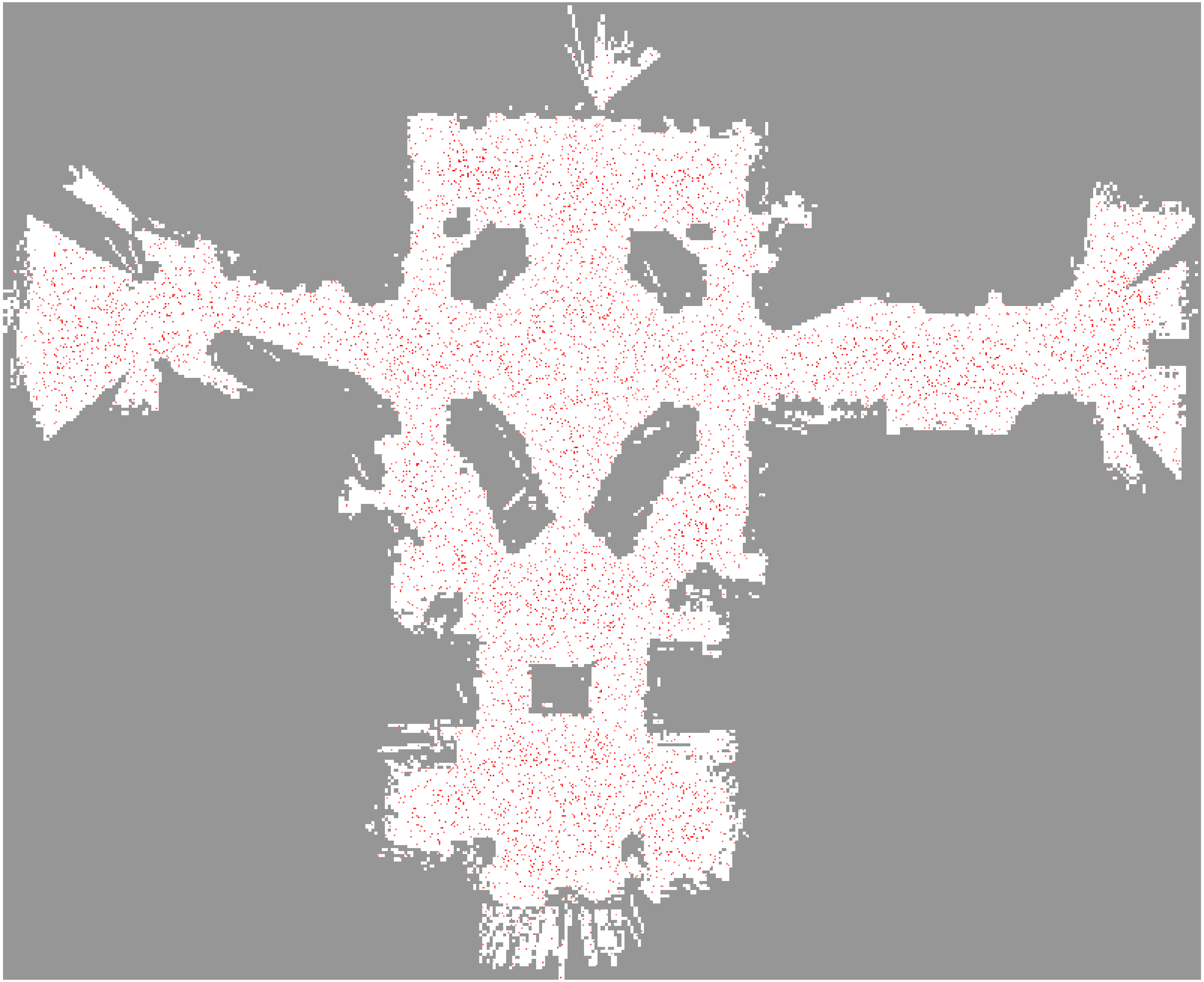
Resampling

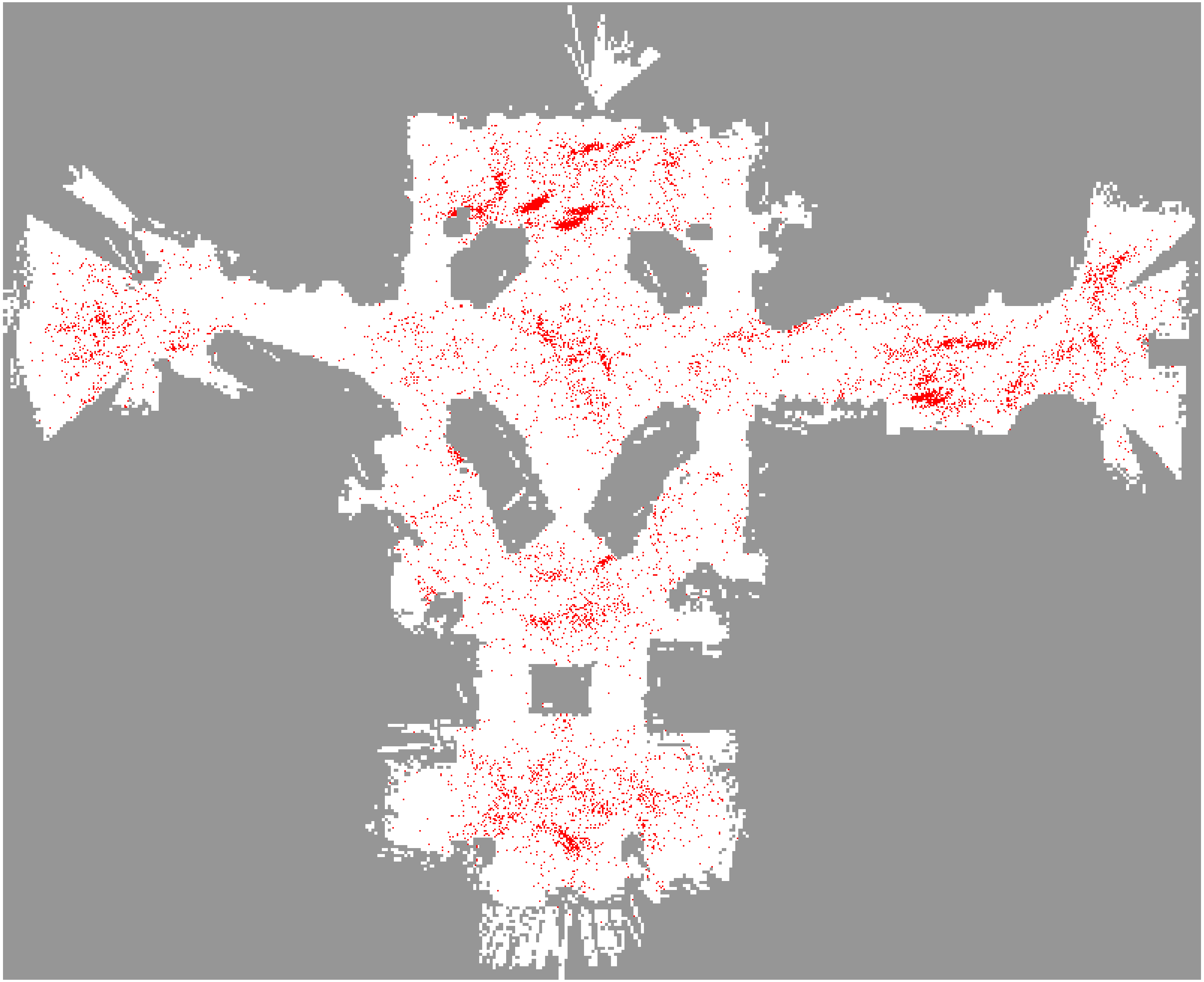
- **Given**: Set S of weighted samples
- **Wanted** : Random sample, where the probability of drawing x_t^i is given by w_t^i
- Typically done *n* times with replacement to generate new sample set S' with uniform weight, but different density

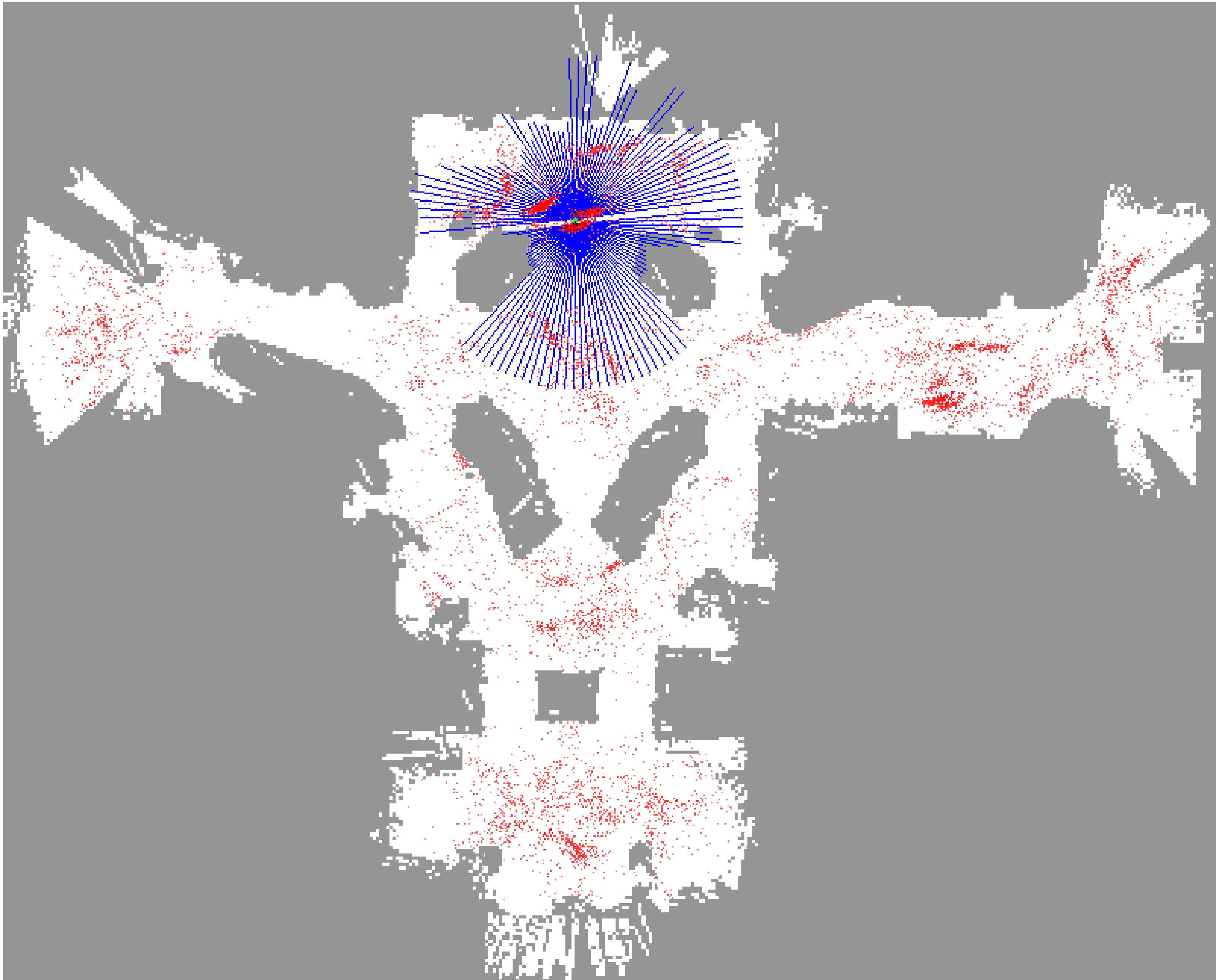


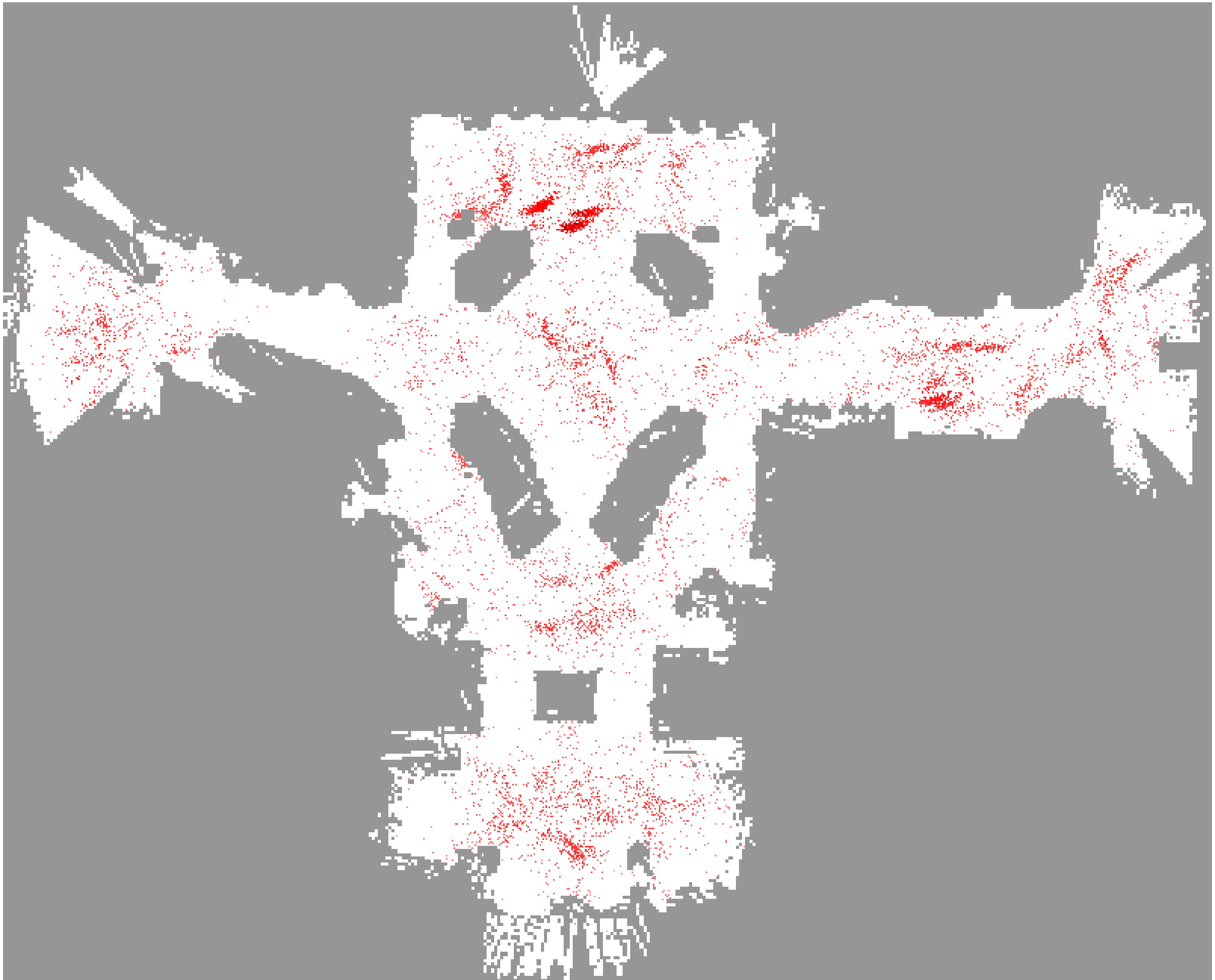


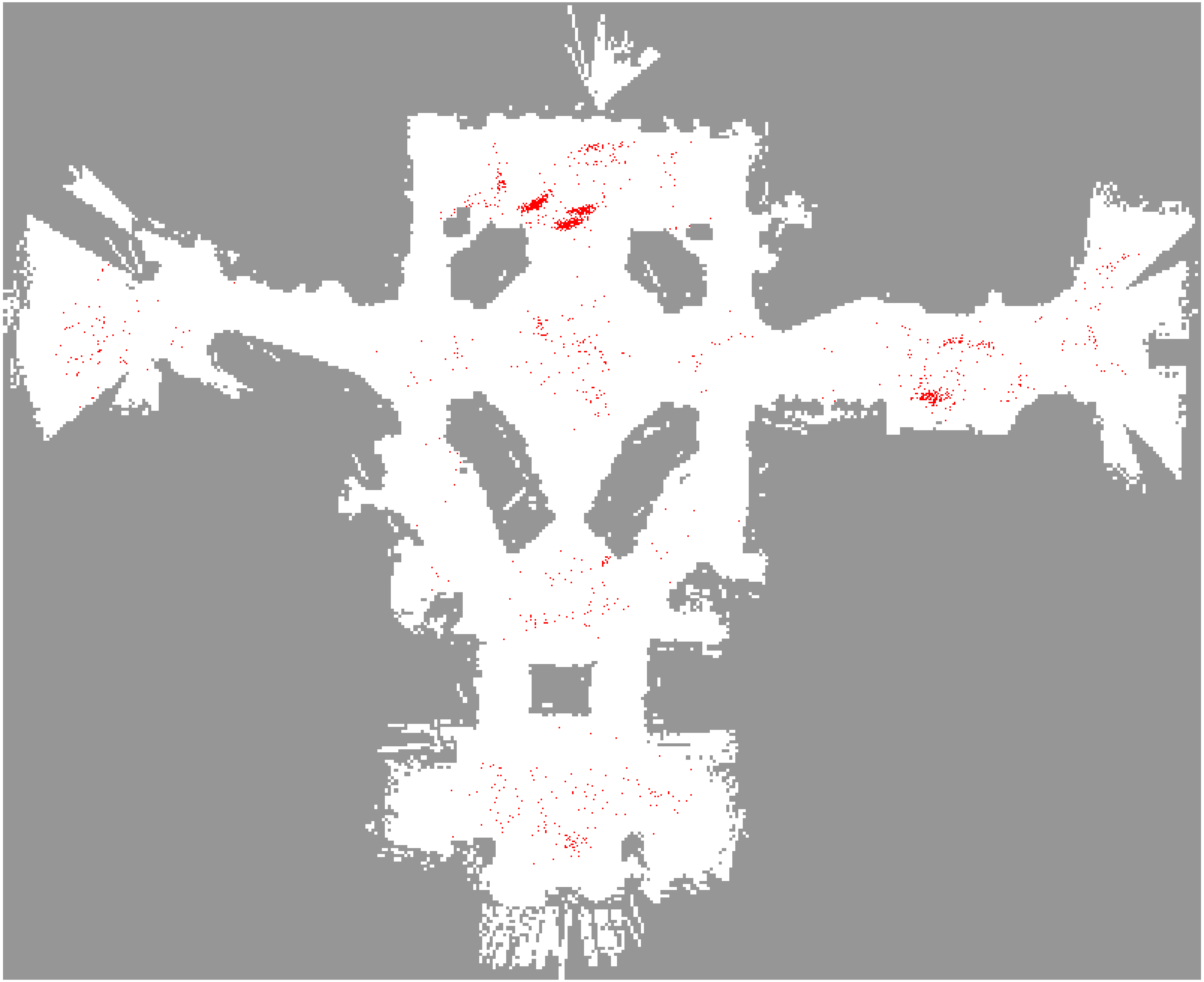


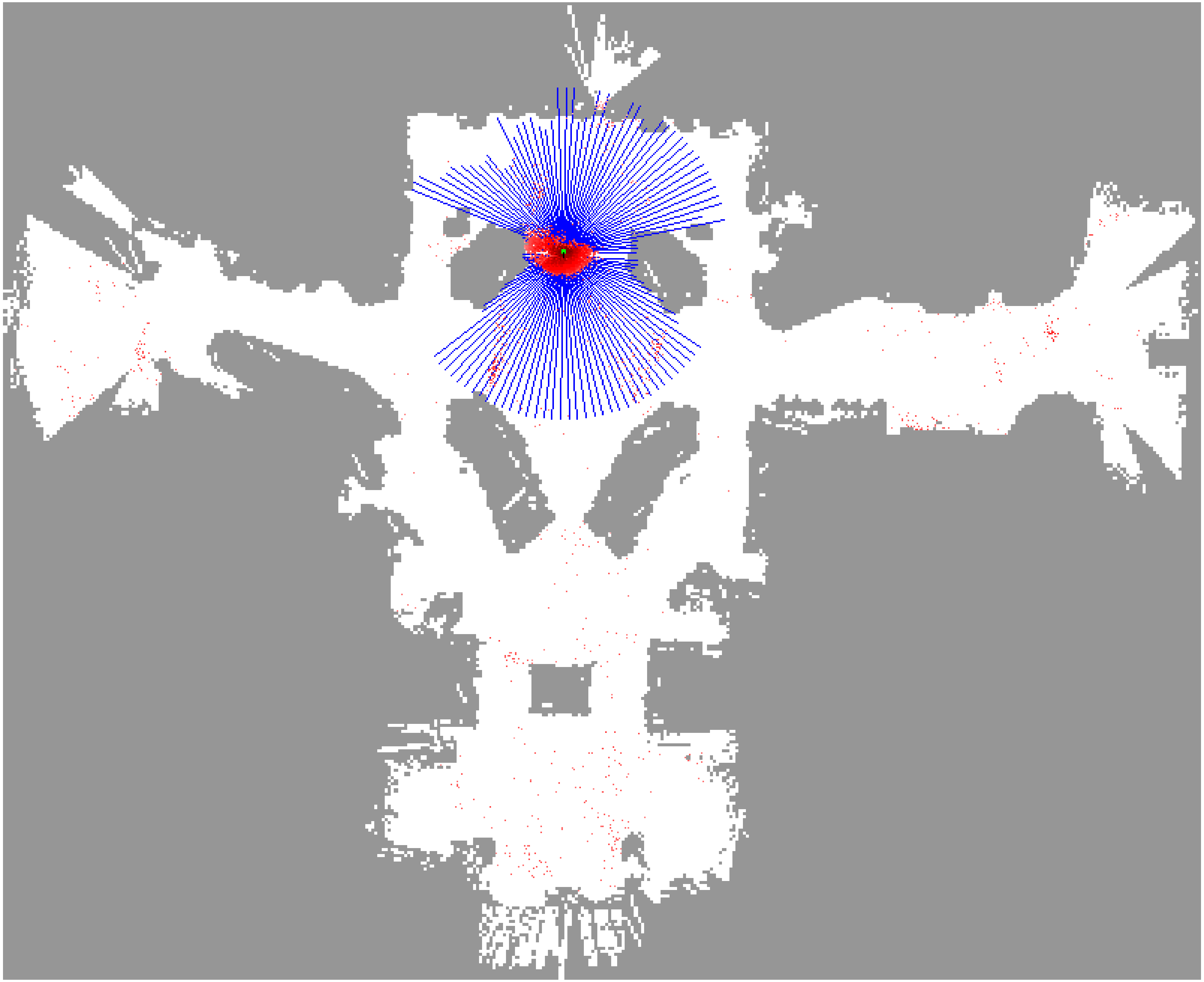


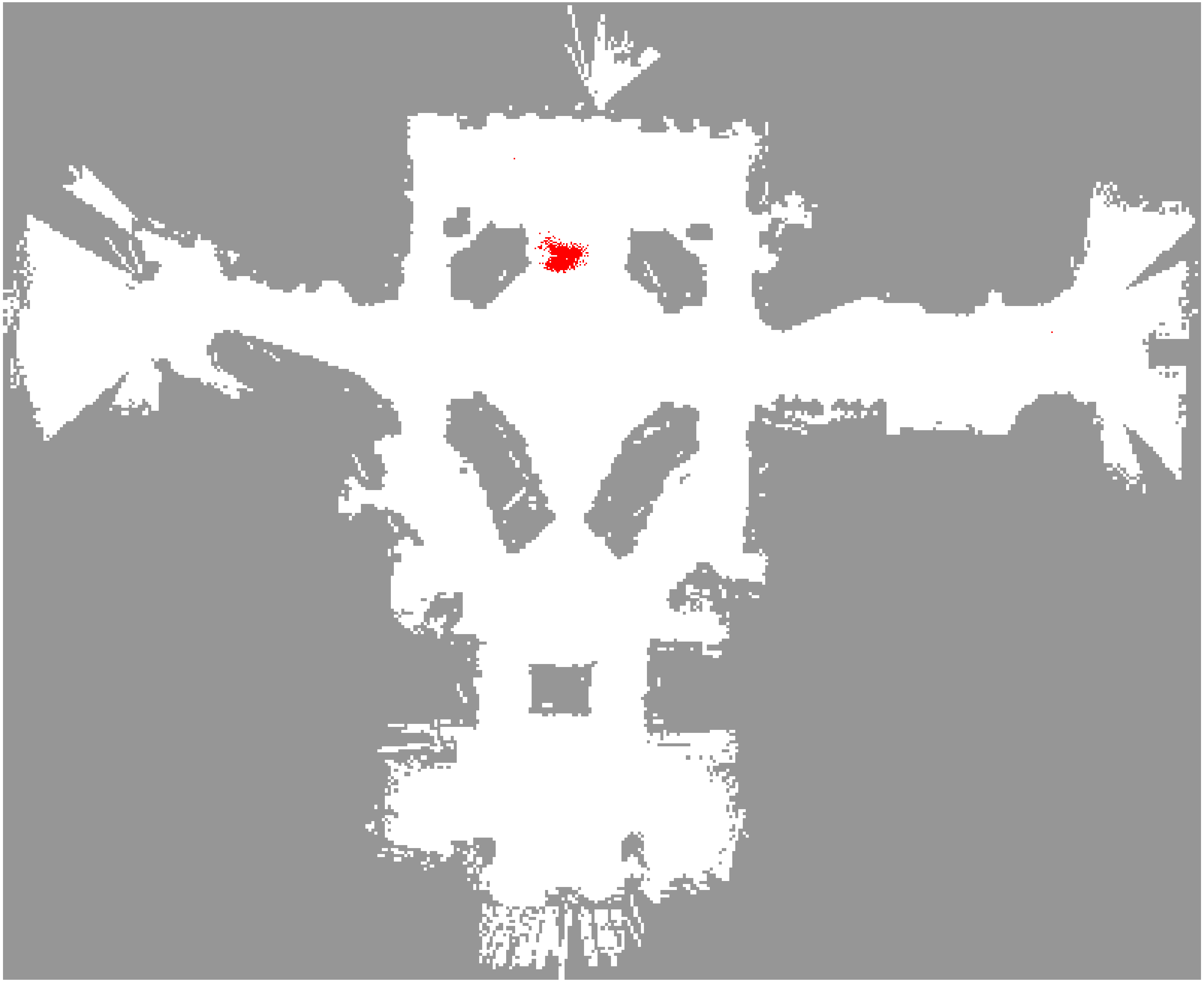


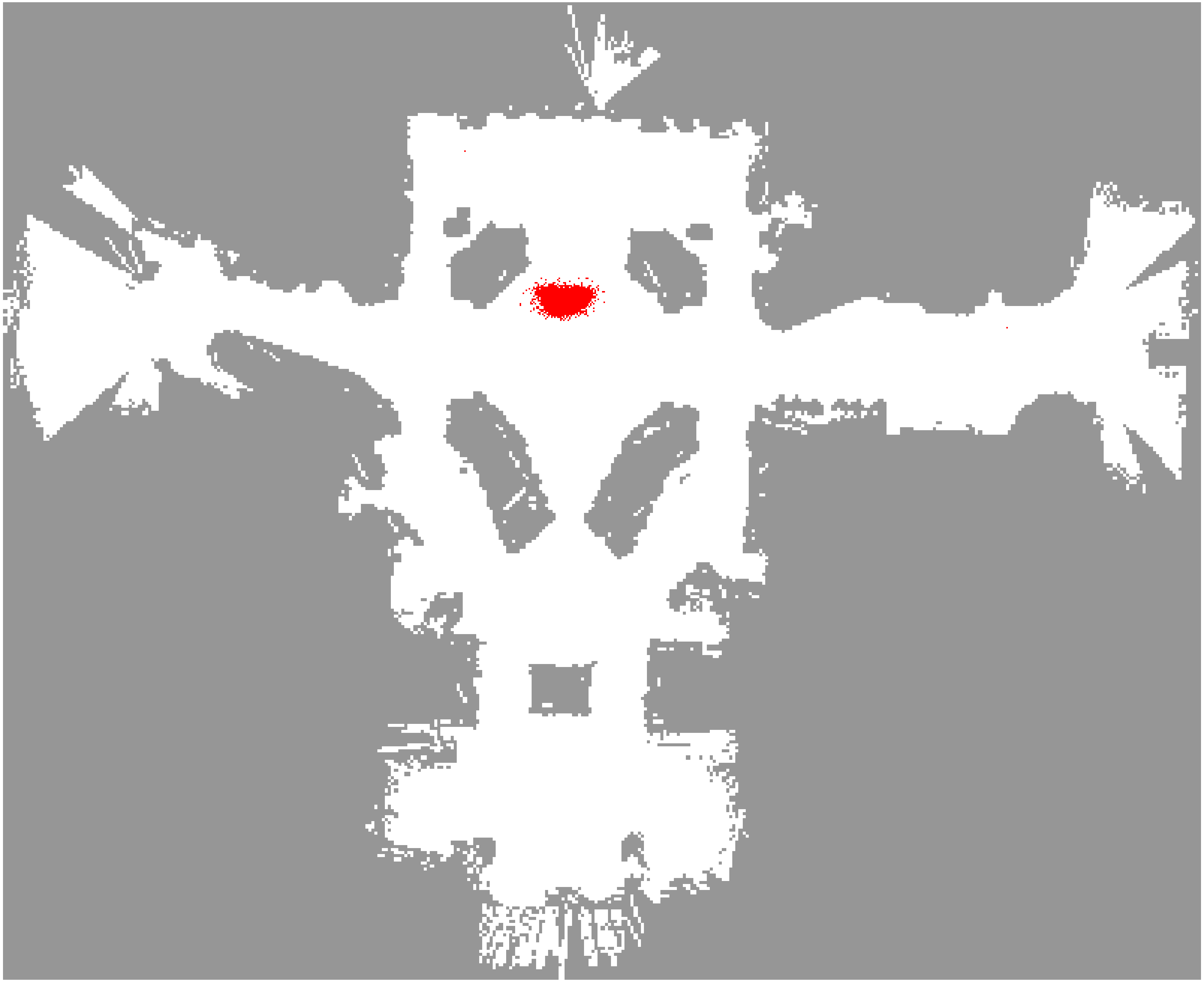


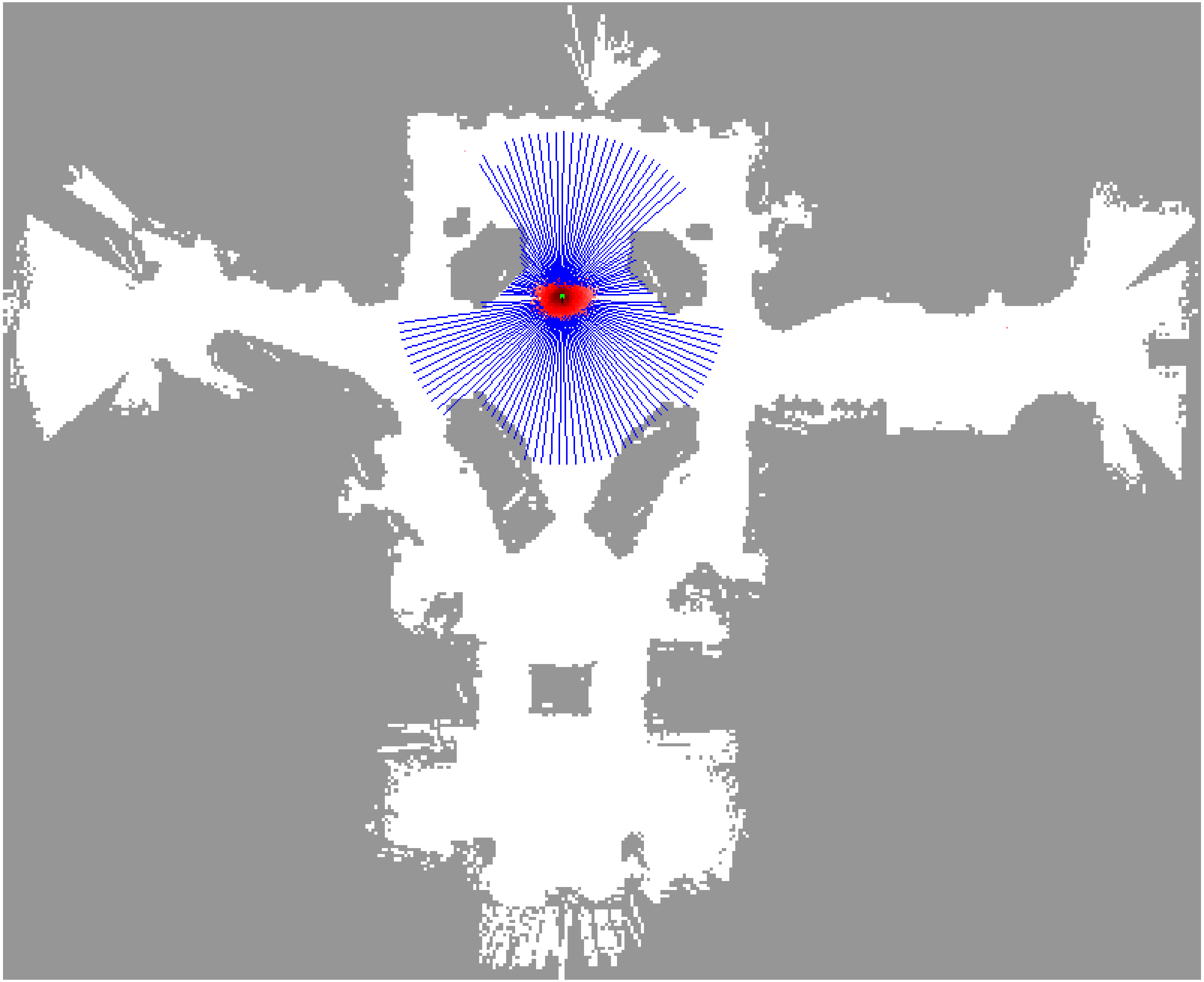


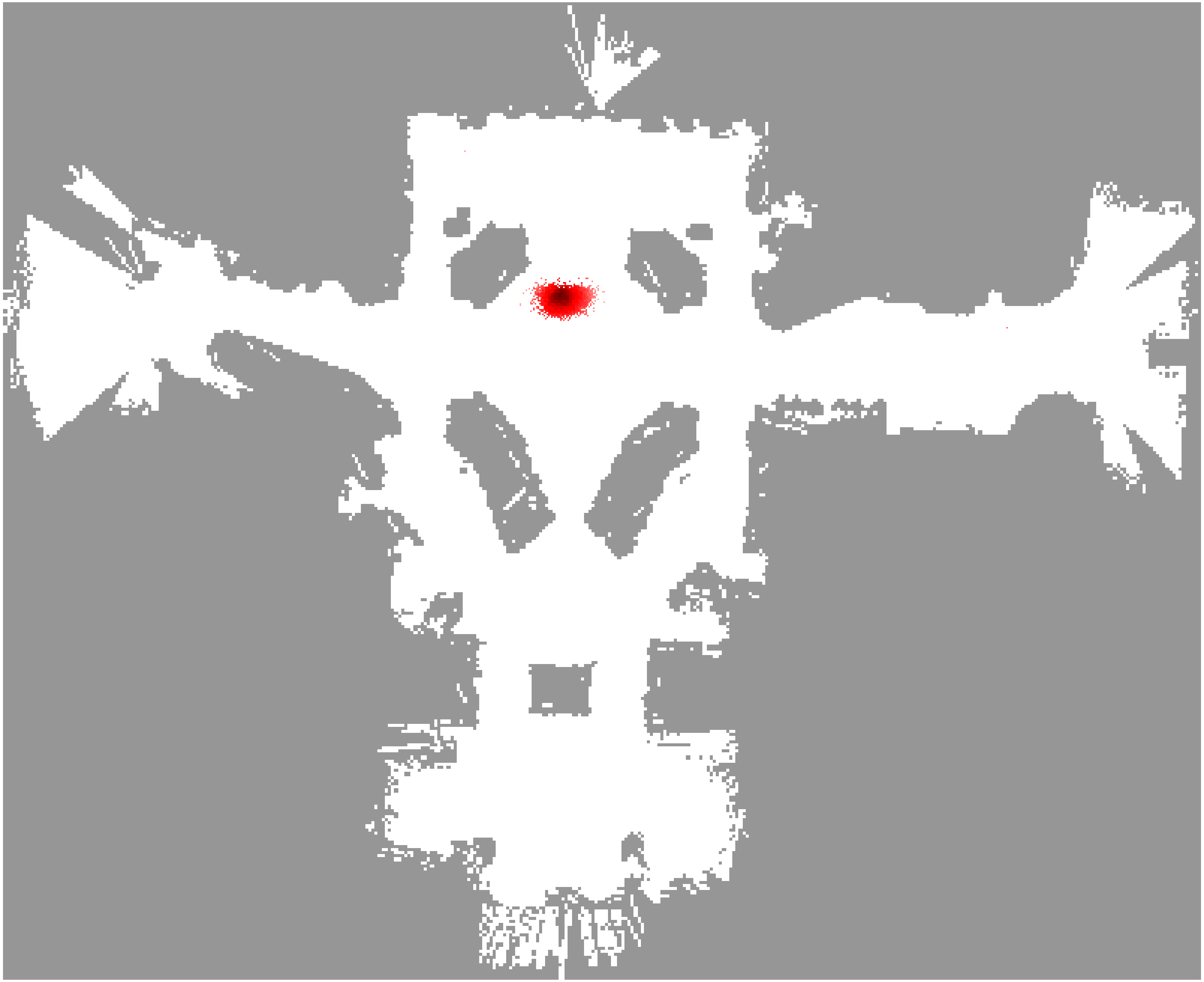


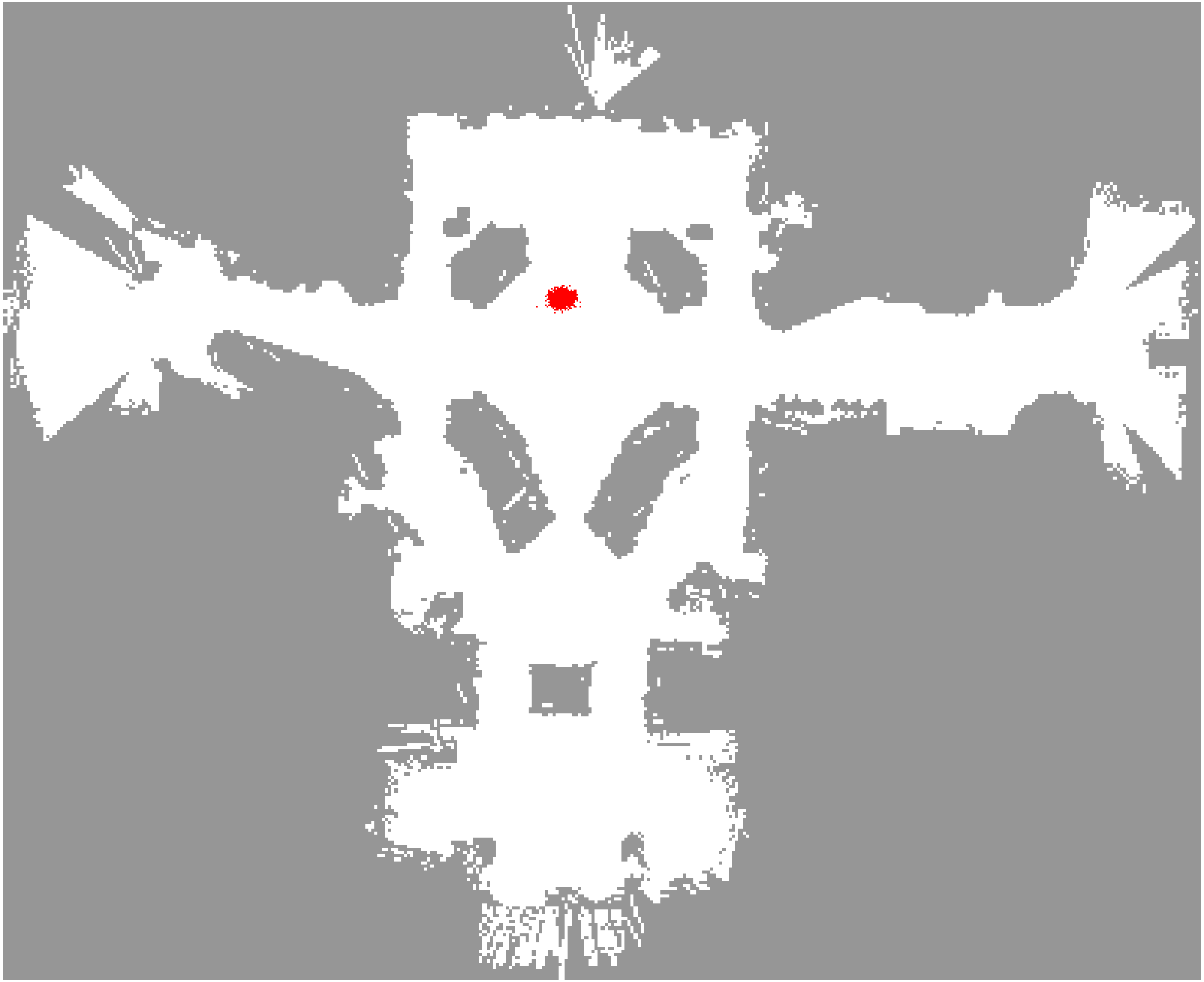


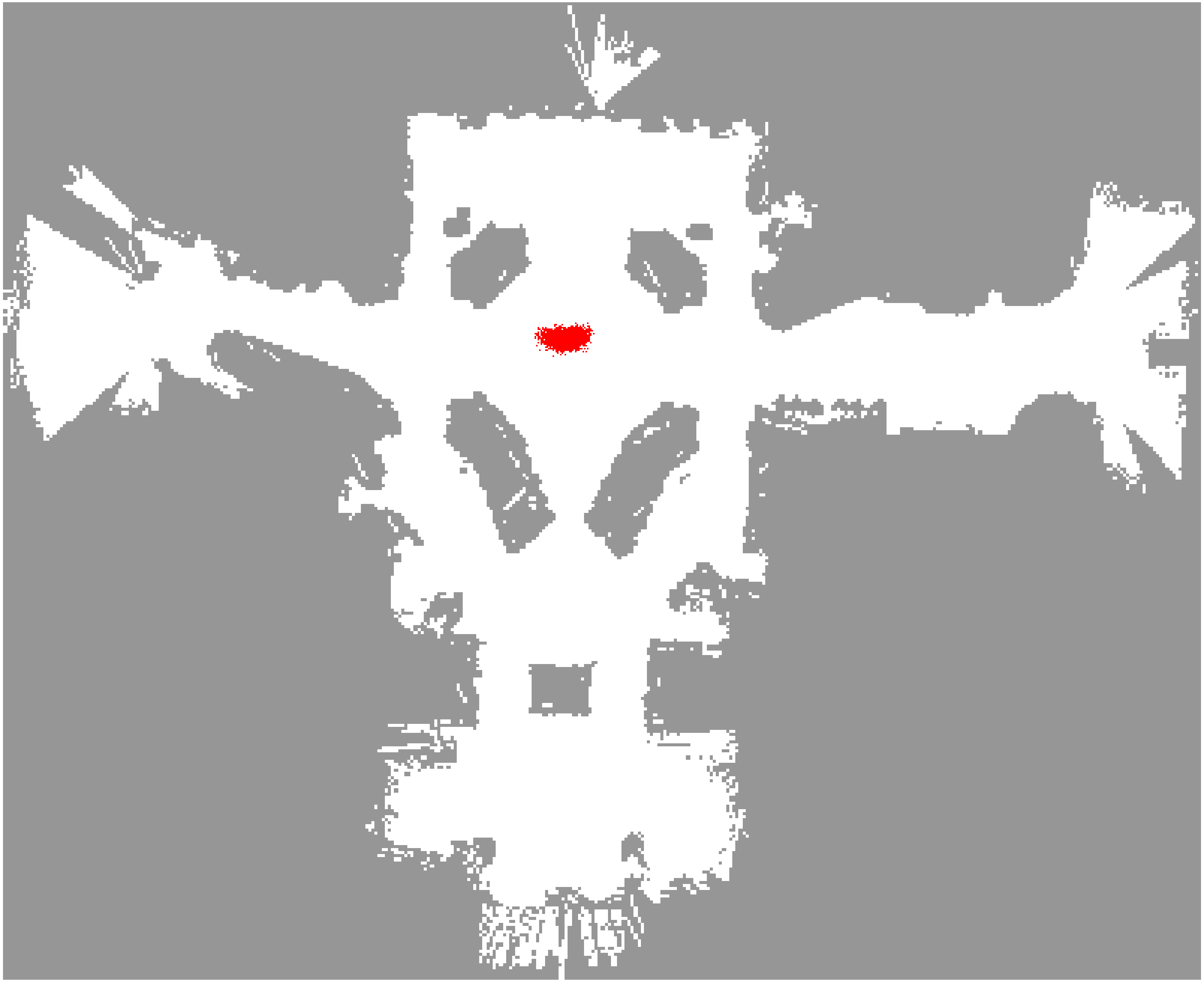


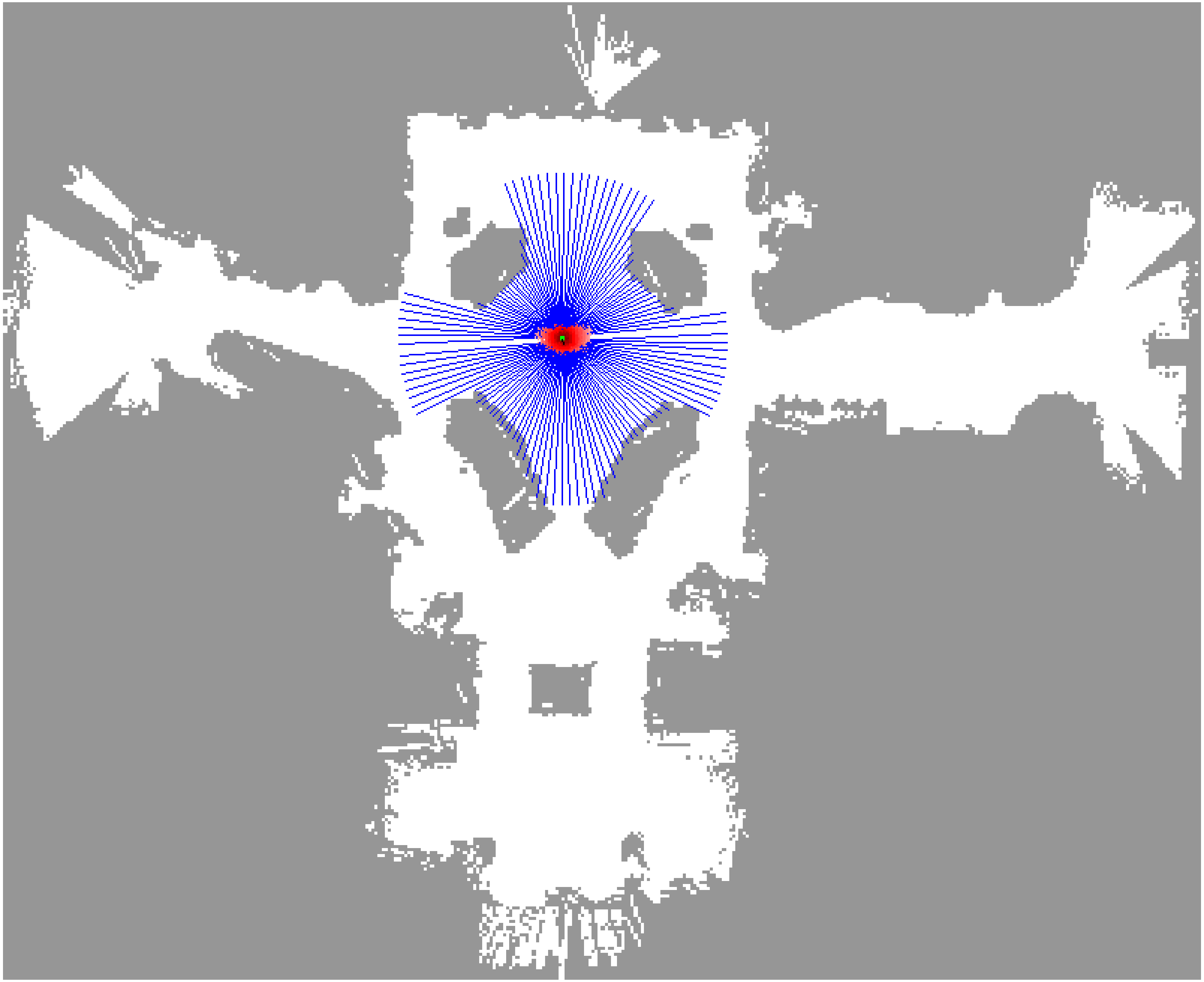


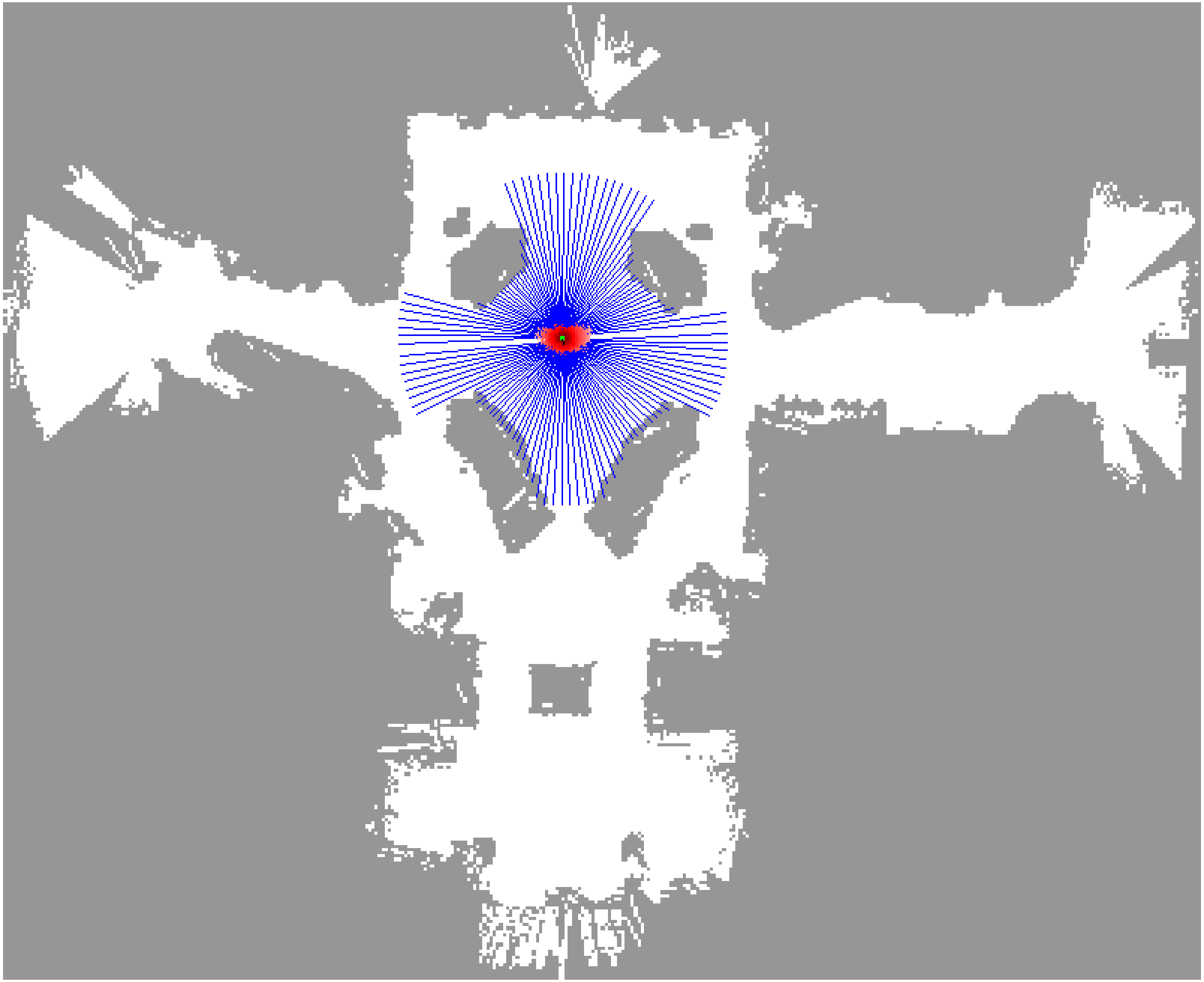




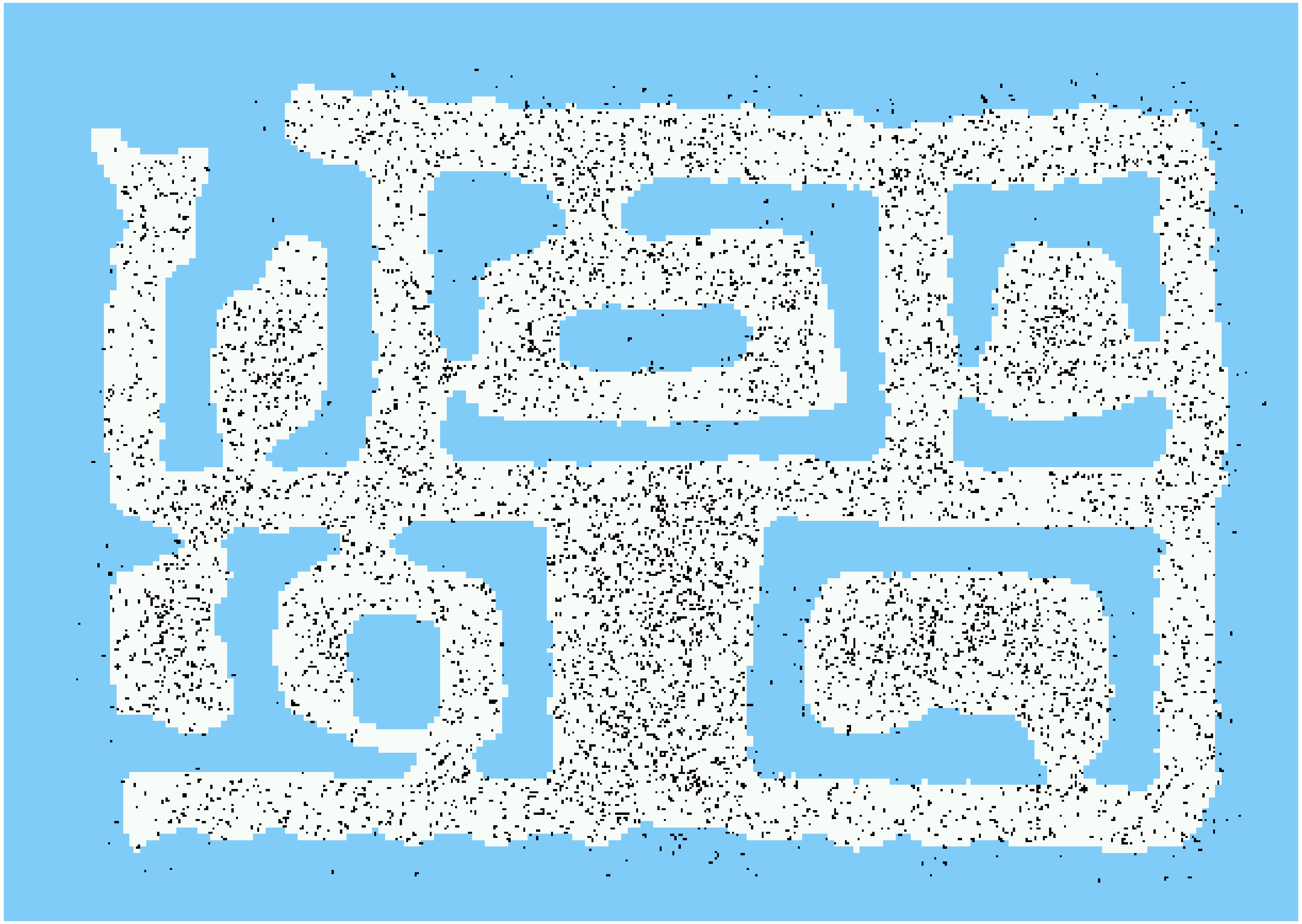




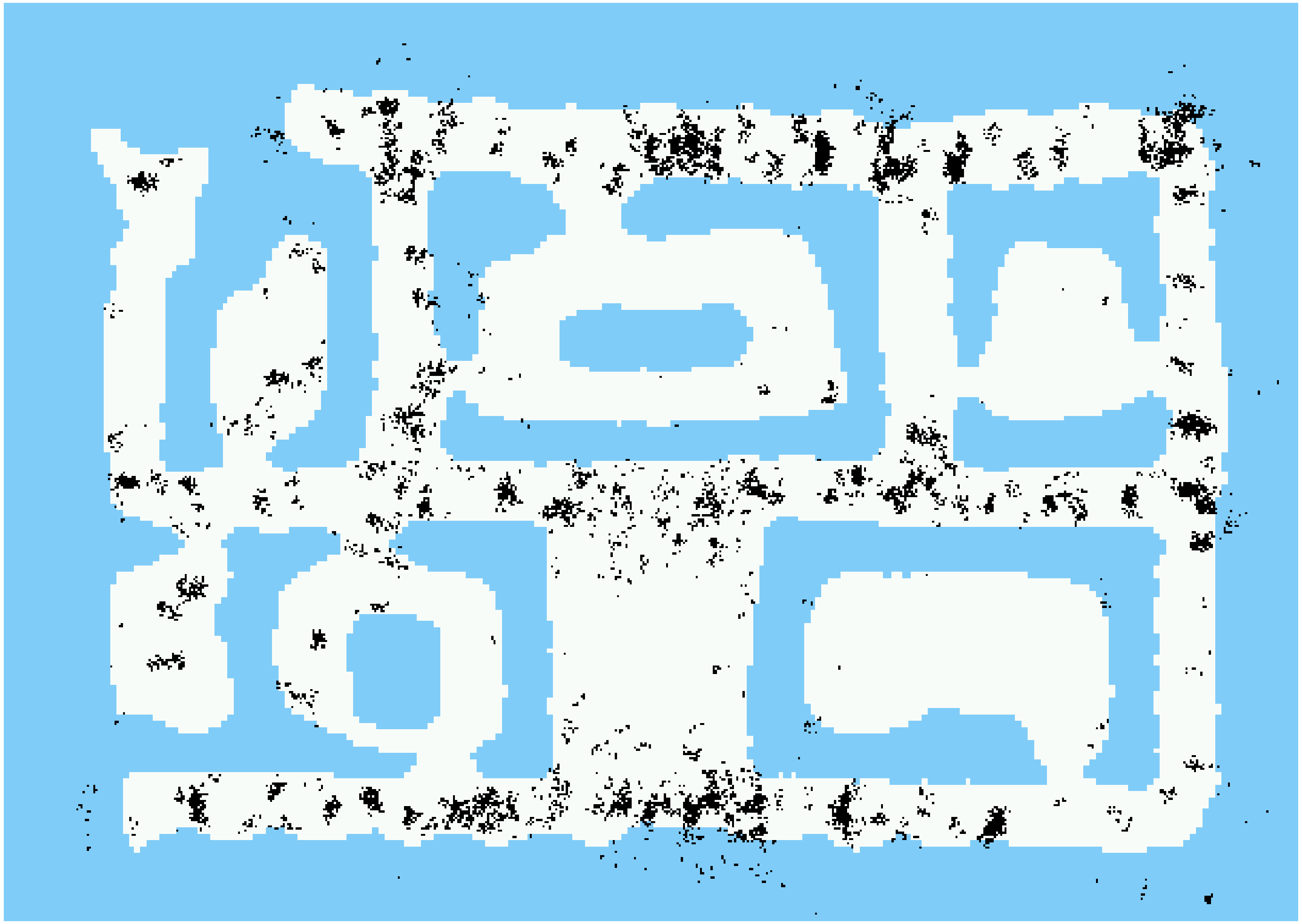




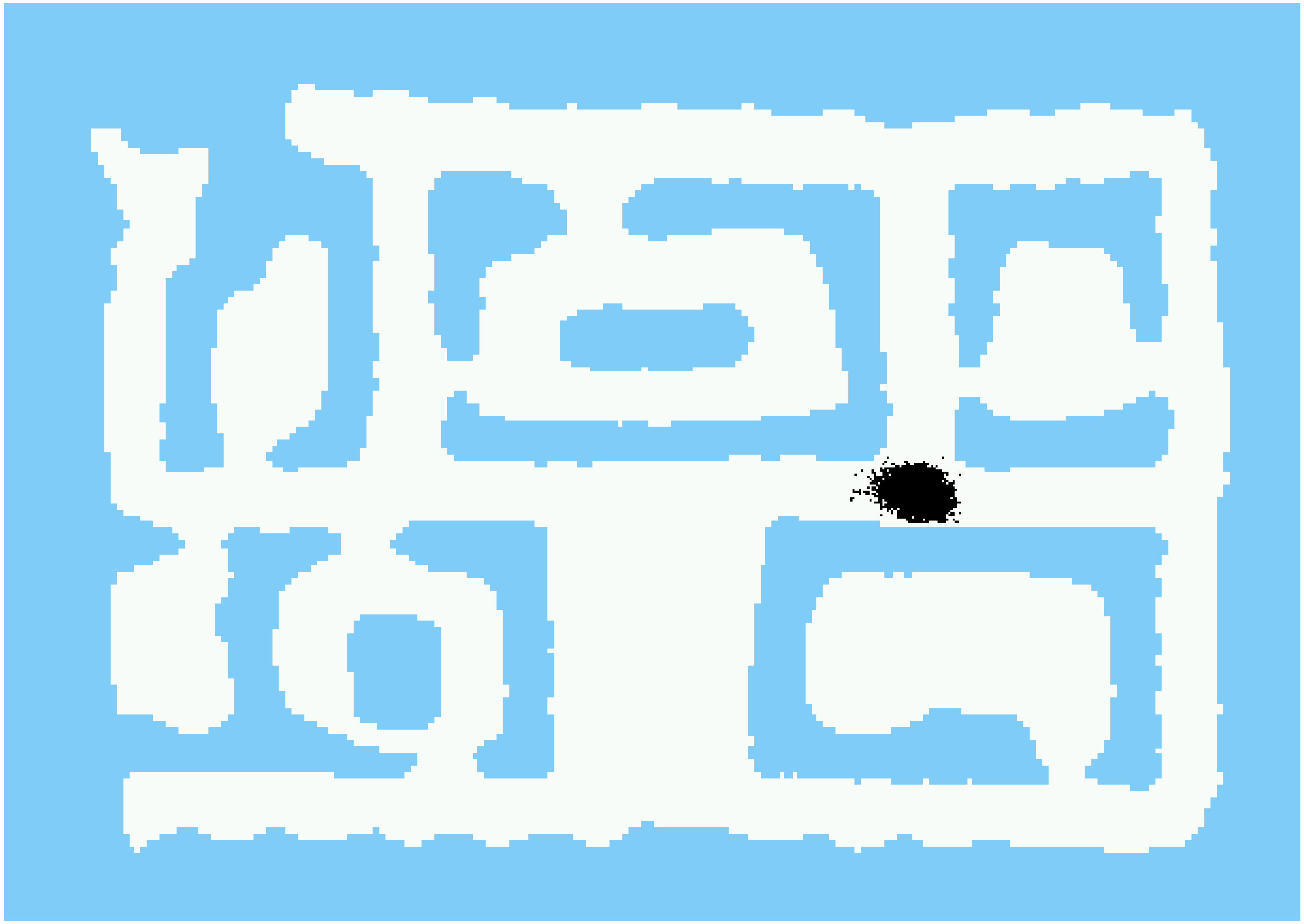
Initial Distribution



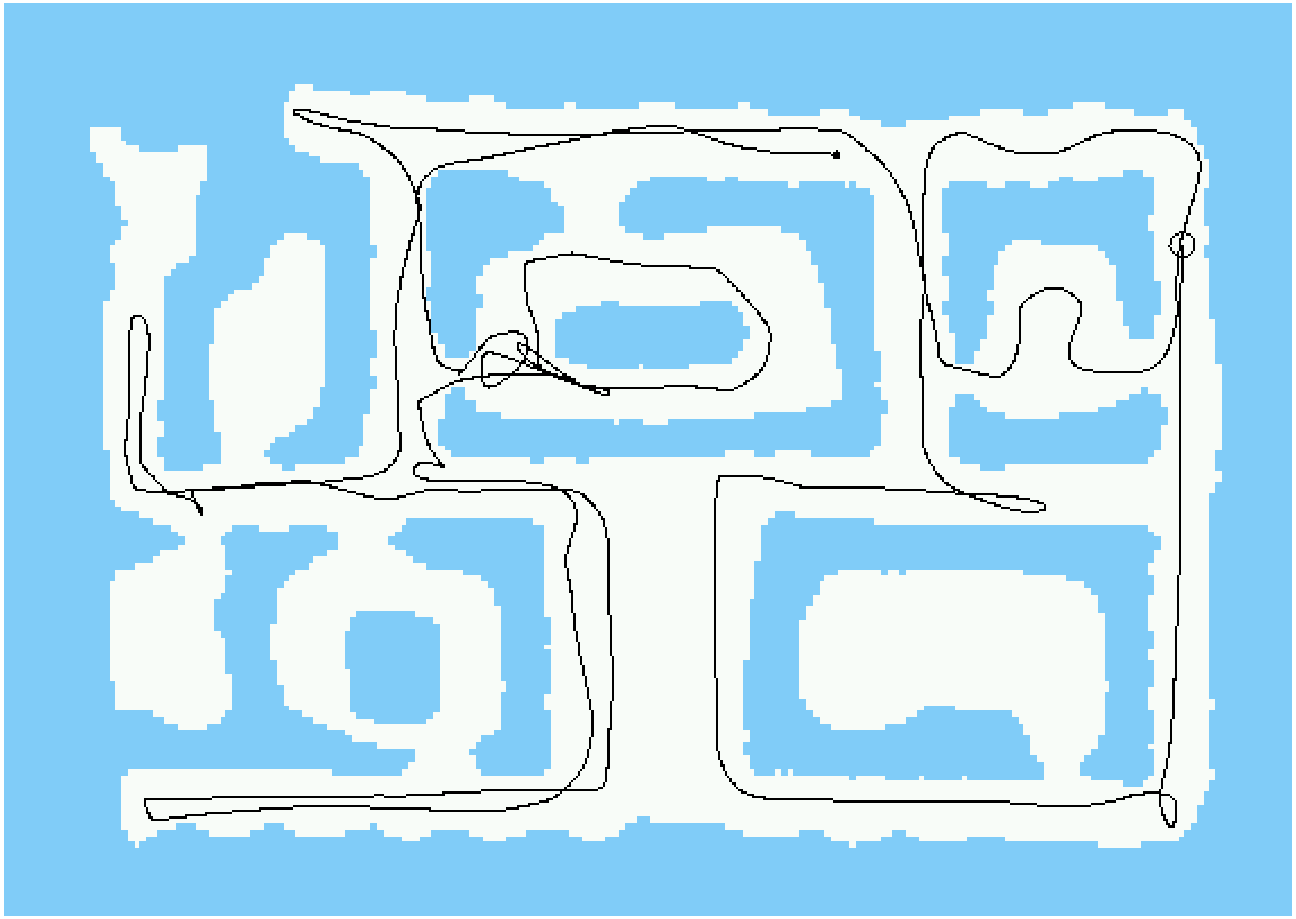
After Incorporating 10 Sonar Scans



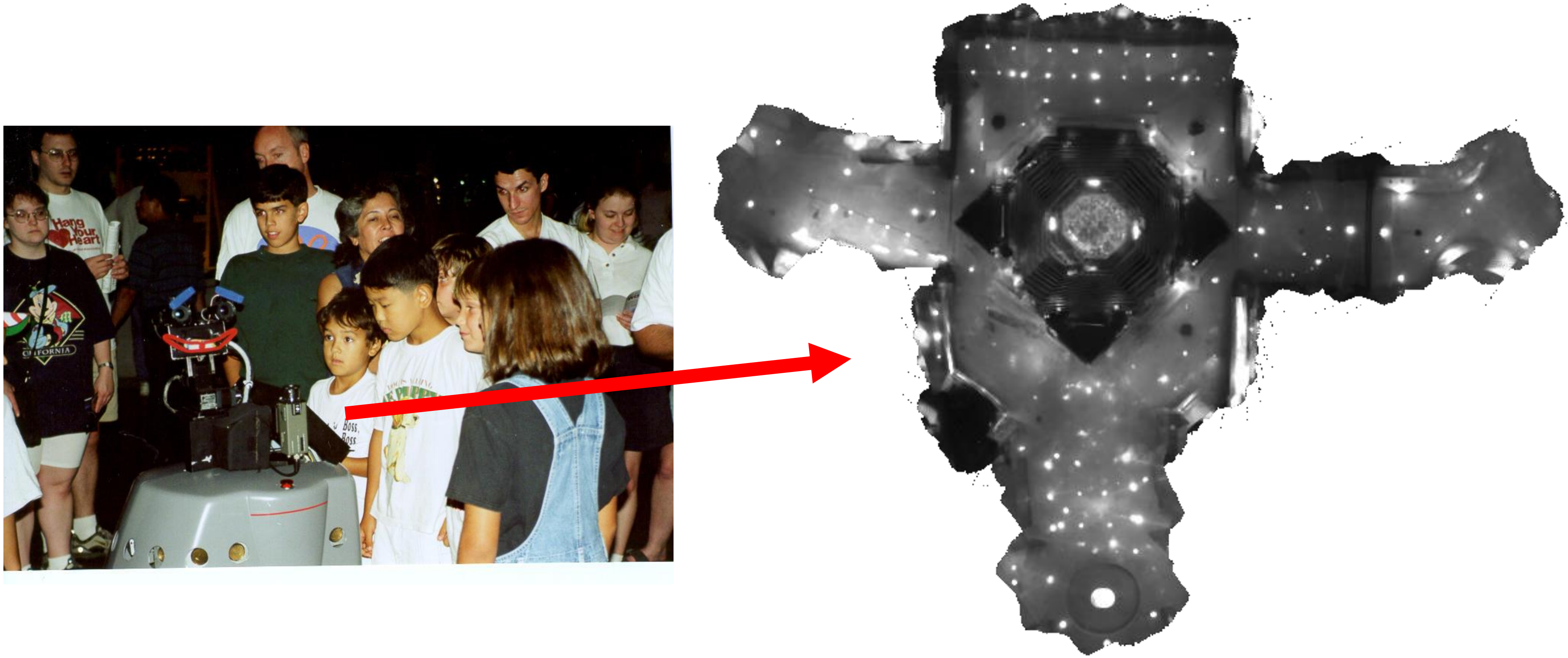
After Incorporating 65 Sonar Scans



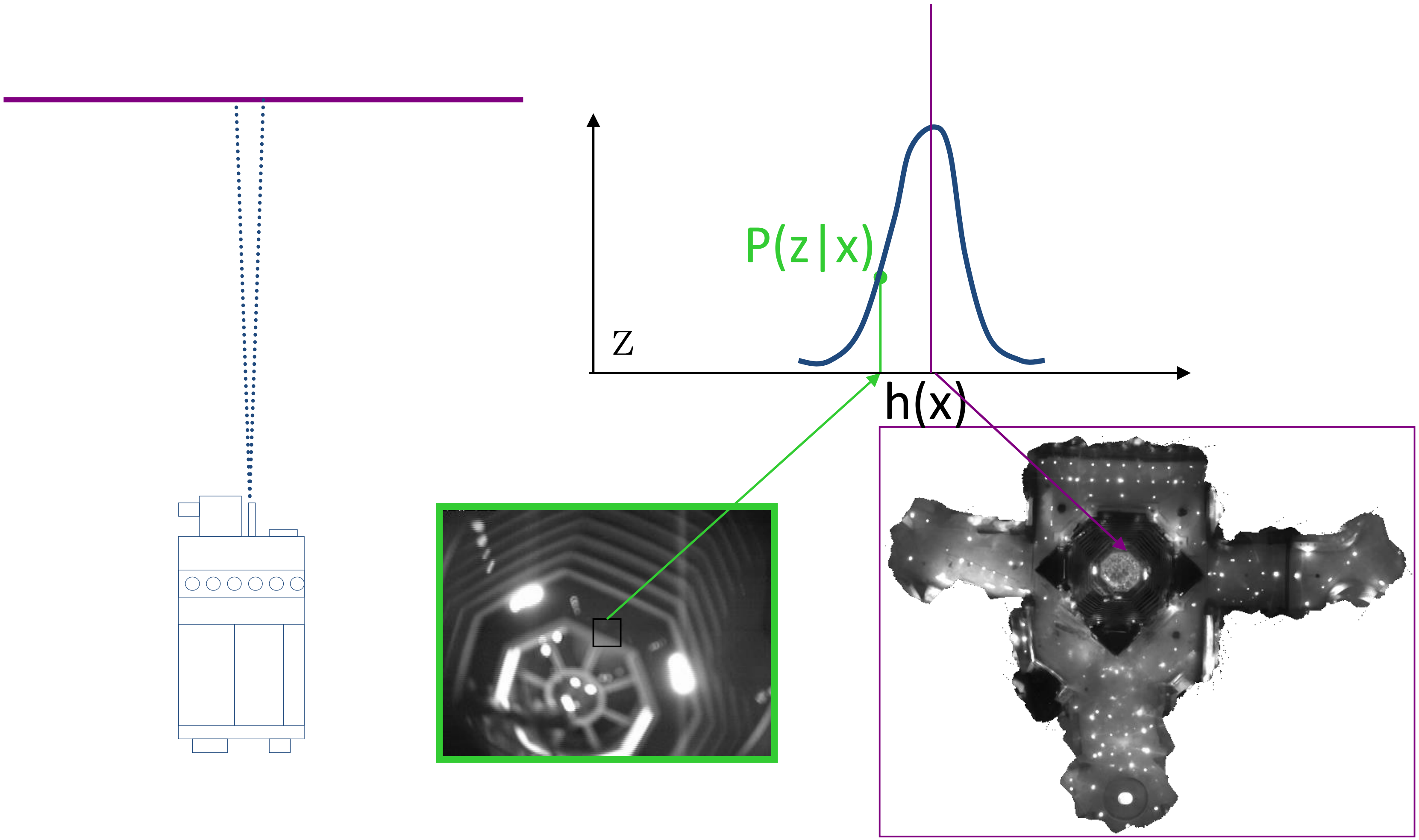
Estimated Path



Using Ceiling Maps for Localization

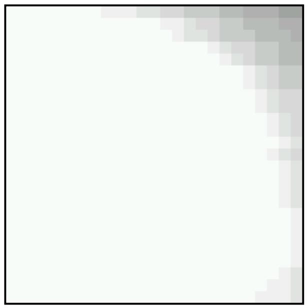


Vision-Based Localization

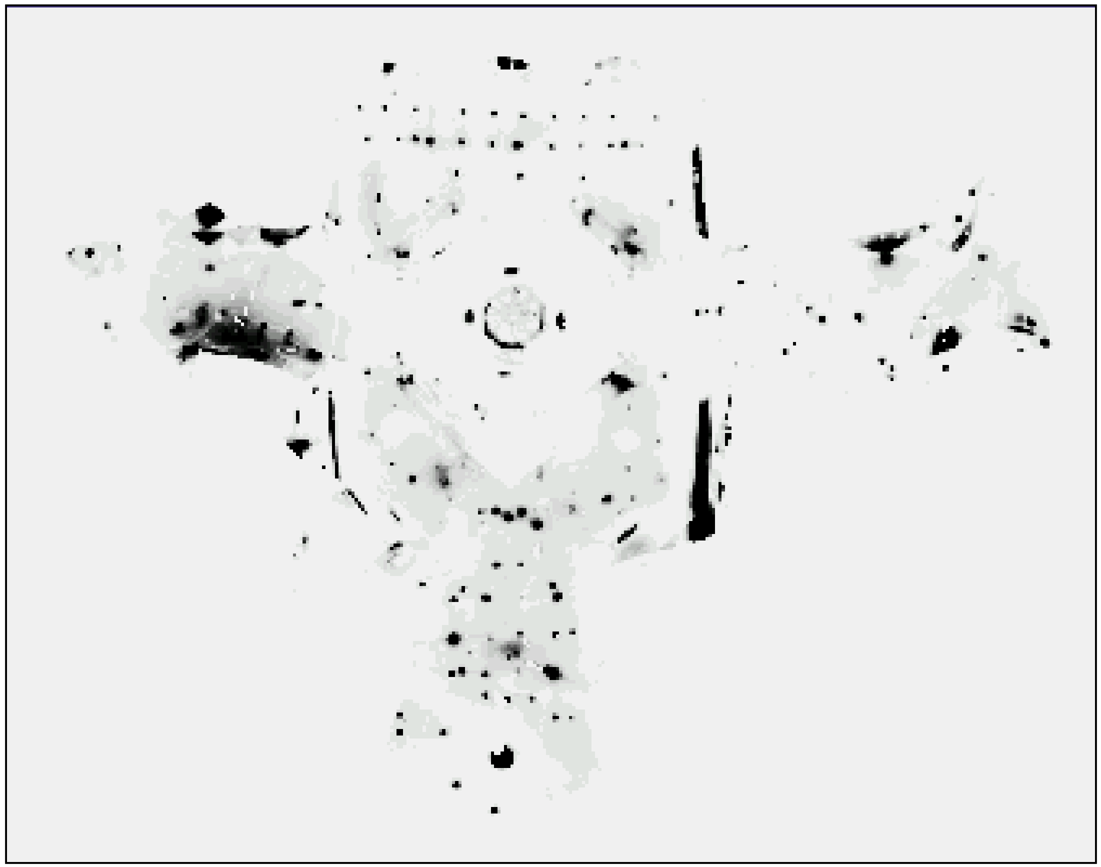


Under a Light

Measurement z:

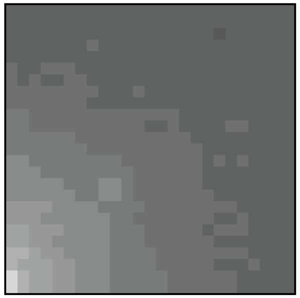


$P(z|x)$:

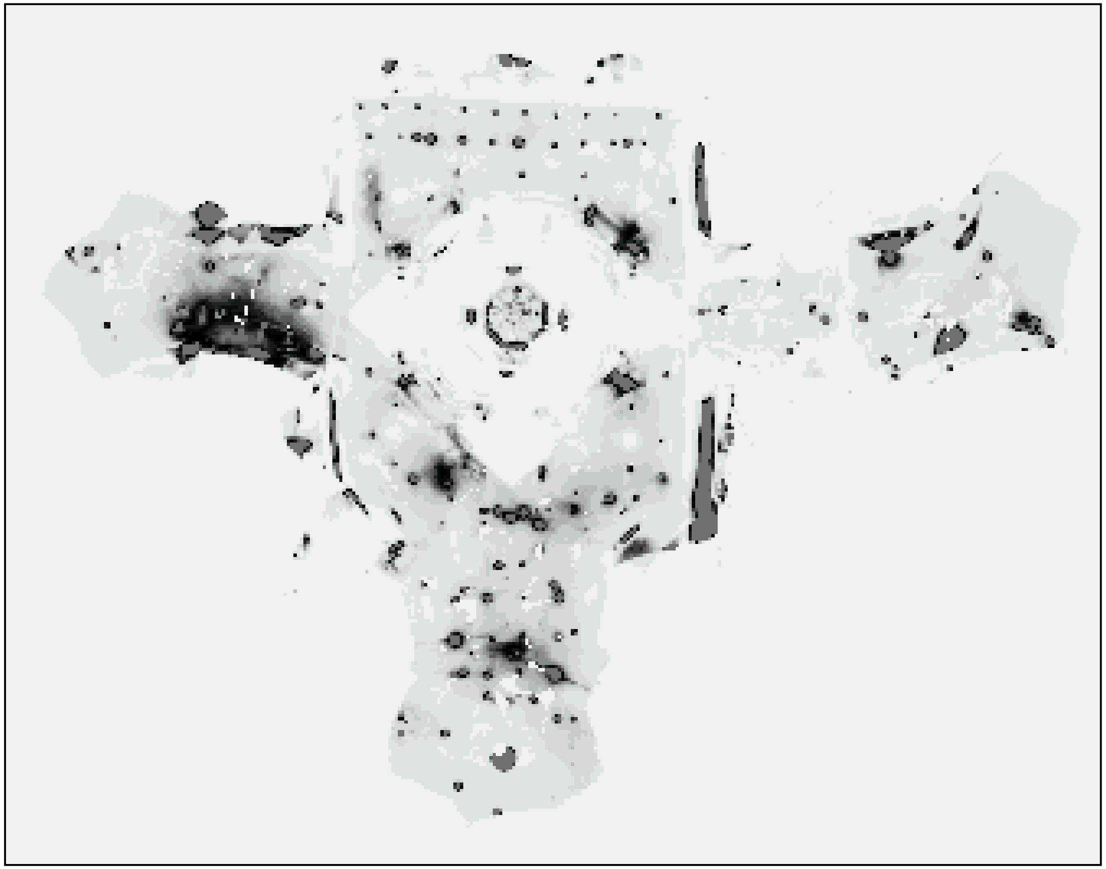


Next to a Light

Measurement z:



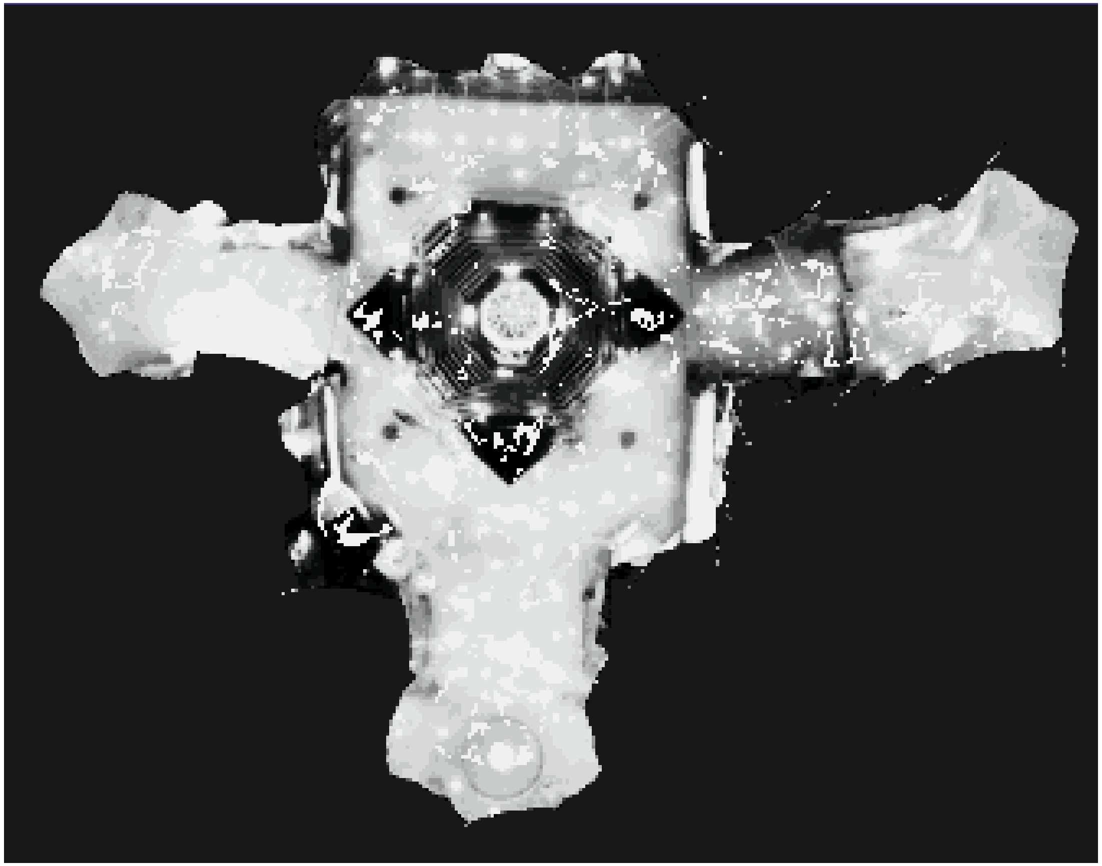
$P(z|x)$:



Elsewhere

Measurement z:

$P(z|x)$:



Limitations

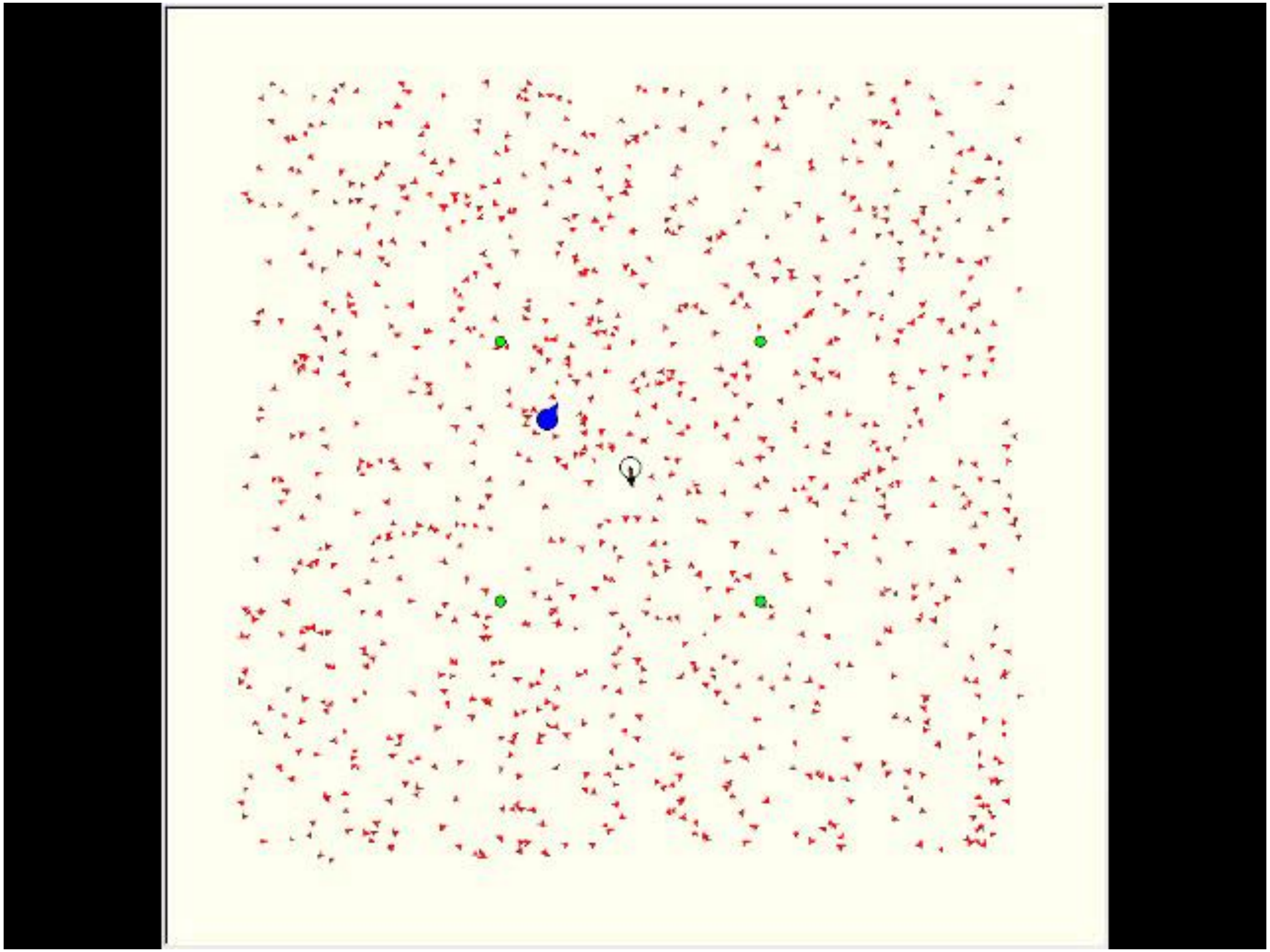
- The approach described so far is able to
  - track the pose of a mobile robot and to
  - globally localize the robot

- How can we deal with the kidnapped robot problem?

The Kidnapped Robot Problem

- 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)

The Kidnapped Robot Problem



Summary

- Particle filters are implementations 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 the re-sampling step, new particles are drawn with a probability proportional to the likelihood of the observation.

Summary

- Pros:
  - Able to represent arbitrary distribution
  - Able to handle nonlinear systems without linearization
- Cons:
  - May need lots of particles to represent high dimensional state space, computational complexity increases significantly w.r.t state dimension
  - Particle degeneracy problem
- Applications:
  - Widely used for low dimensional problems: robot pose tracking, target tracking, etc.
  - Used for initialization of global localization to resolve the multi-modal issue, then switch to unimodal (e.g. EKF) methods
  - Used to be popular for SLAM, but not anymore

Reading

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

Logistics

- Project 3, phase 1 is extended for one week, due 04/28
- Project 3, phase 2 is released, due 04/28