POLITECNICO
MILANO 1863

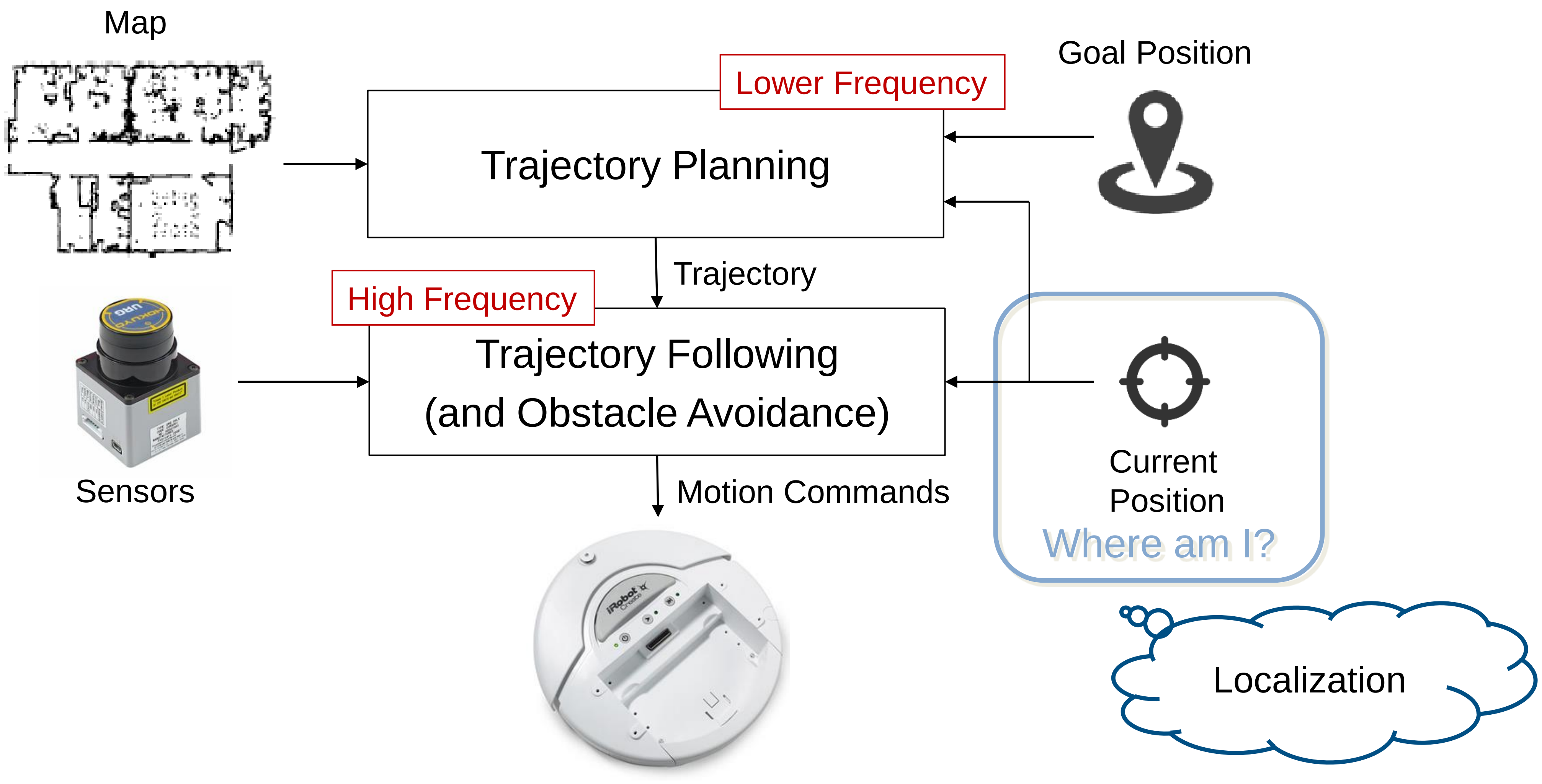
# Robotics

*Robot Localization – Sensor Models and Bayesian Filtering*

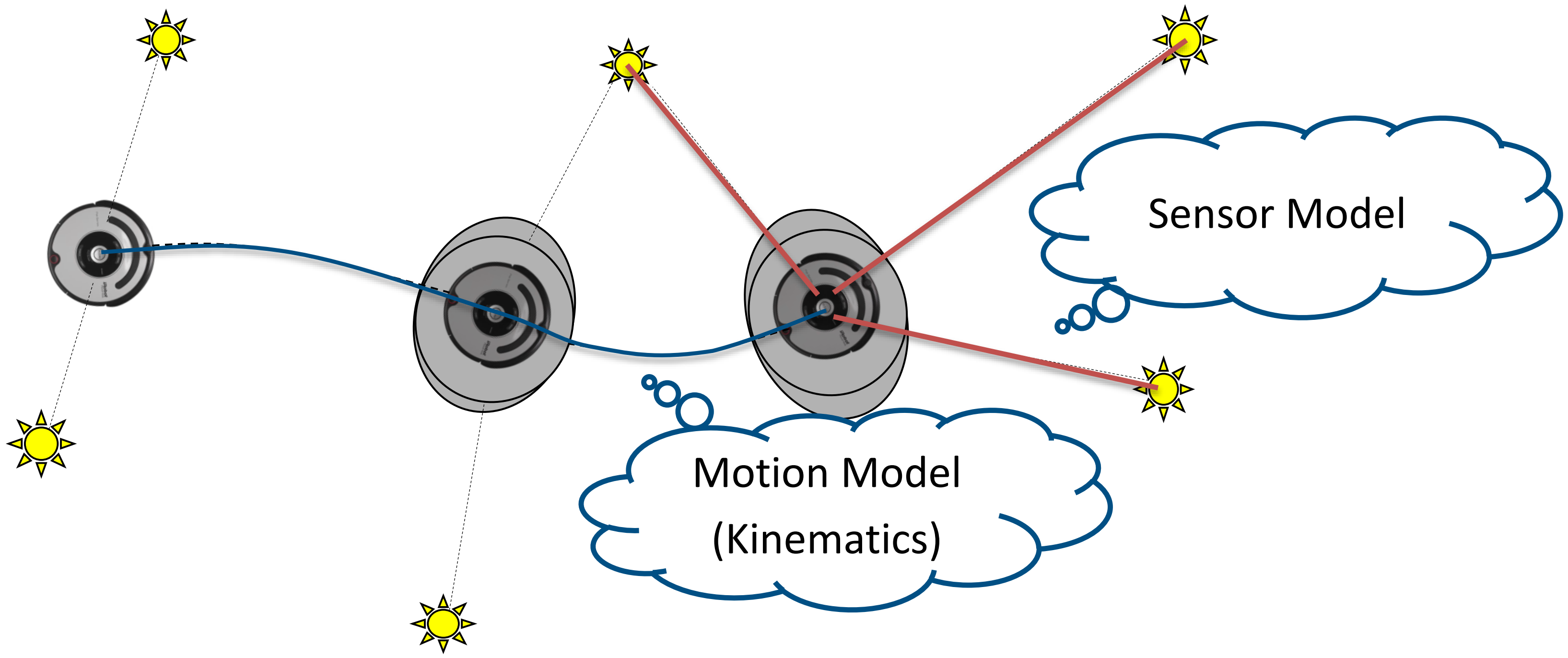

Matteo Matteucci
*matteo.matteucci@polimi.it*

*Artificial Intelligence and Robotics Lab - Politecnico di Milano*

# A Simplified Sense-Plan-Act Architecture

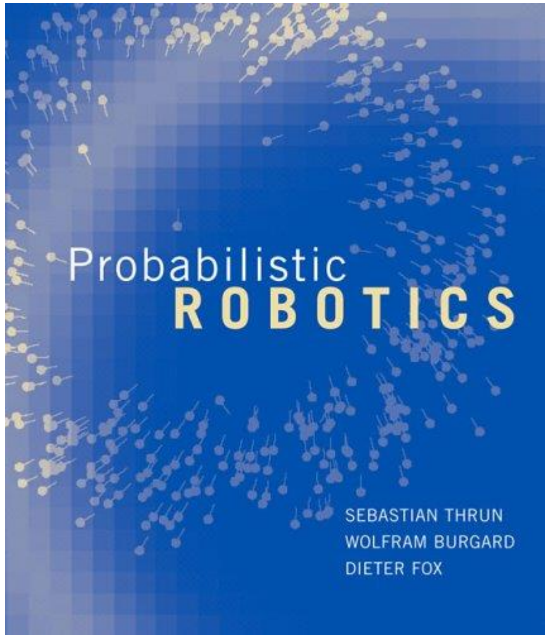


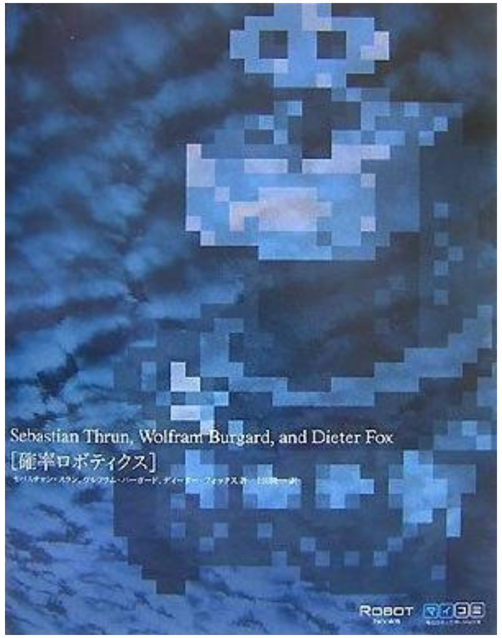
# Localization with Knowm Map



## Disclaimer …

Slides from now on have been heavily “inspired” by the teaching material kindly provided with: S. Thrun, D. Fox, W. Burgard. “*Probabilistic Robotics*”. MIT Press, 2005



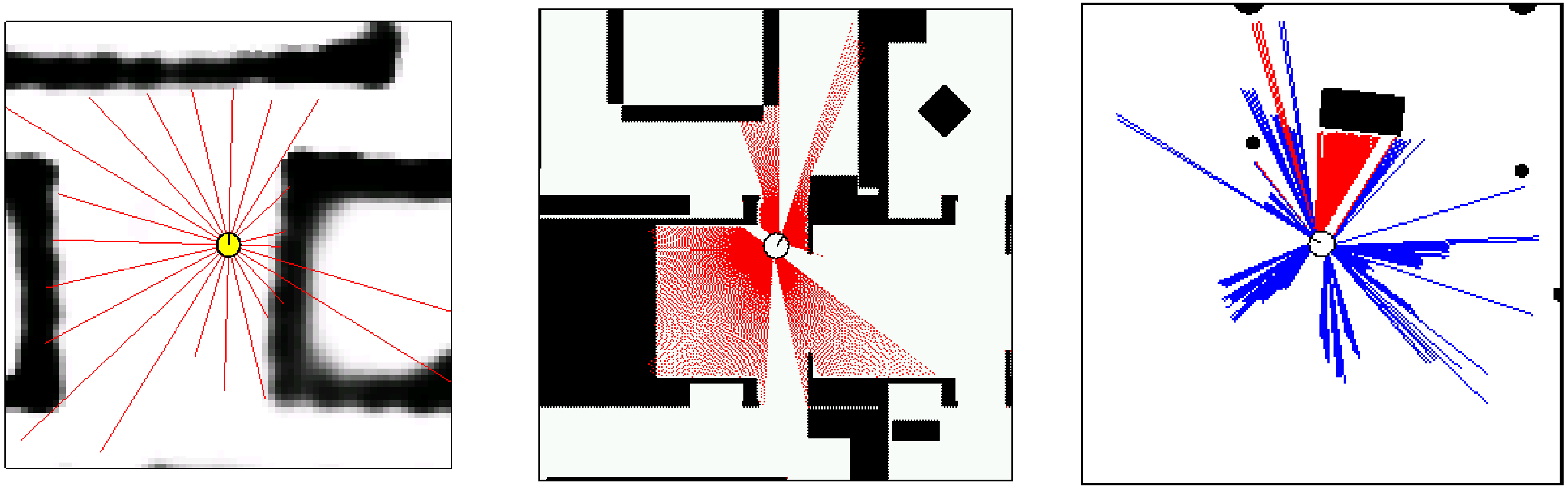




http://robots.stanford.edu/probabilistic-robotics/

You can refer to the original source for deeper analysis and references on the topic …

# Range Sensors Models

The sensor model describes P(z|x), i.e., the probability of a measurement z given that the robot is at position x.



# Proximity Sensors

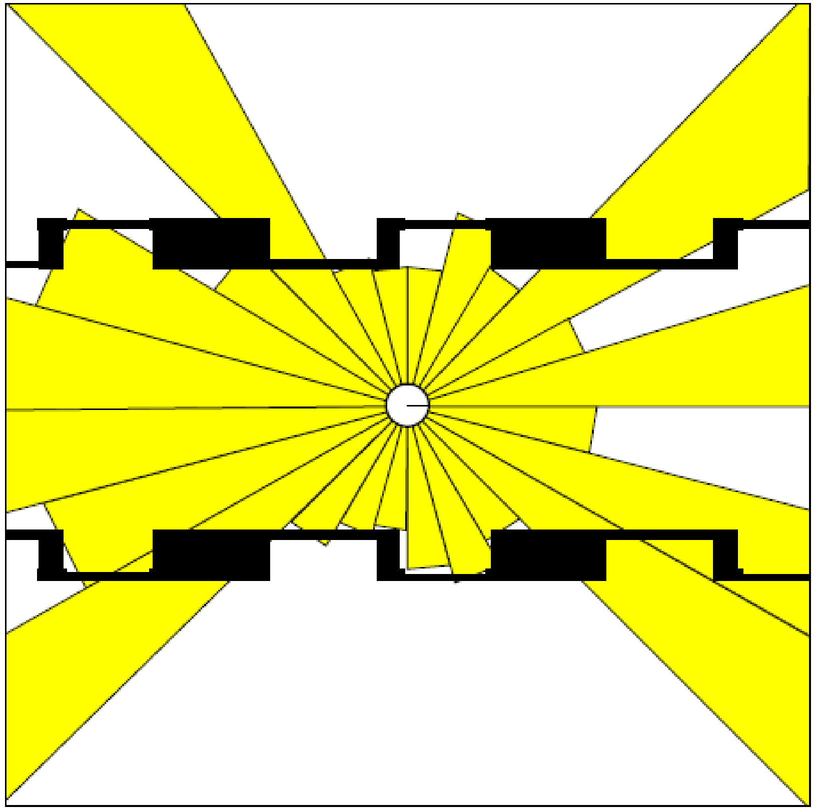
The sensor model describes P(z|x), i.e., the probability of a measurement z given that the robot is at position x.

In particular a scan *z* consists of *K* measurements.

$$z = \{z_1, z_2, \ldots, z_K\}$$

Individual measurements are independent given robot position and surrounding map.

$$P(z \mid x, m) = \prod_{k=1}^{K} P(z_k \mid x, m)$$

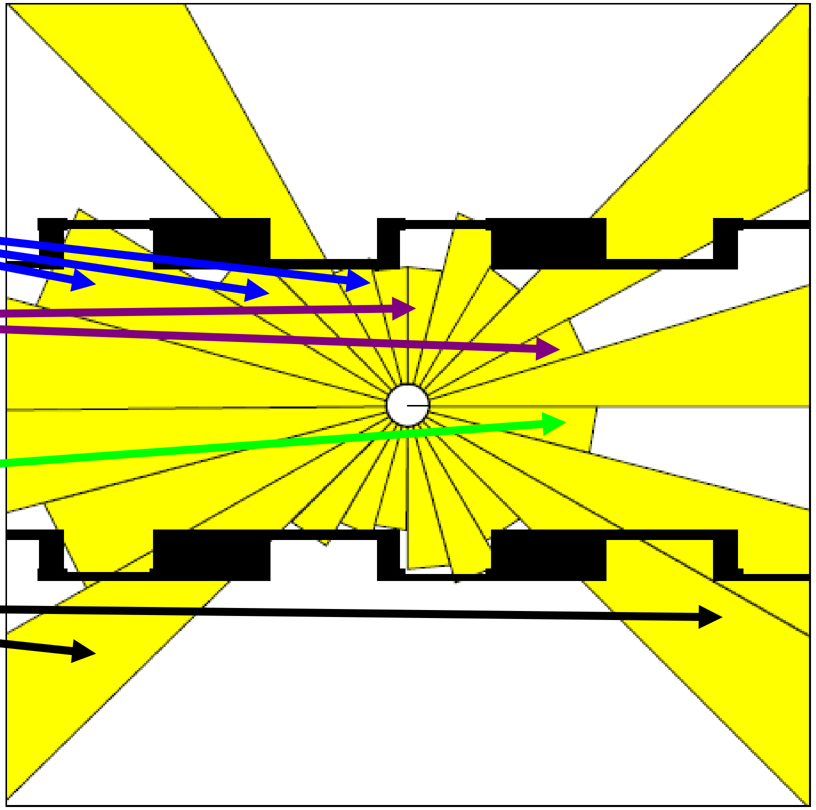


# Typical Measurement Errors of an Range Measurements

The sensor model describes P(z|x), i.e., the probability of a measurement z given that the robot is at position x.

Masurements can come from:

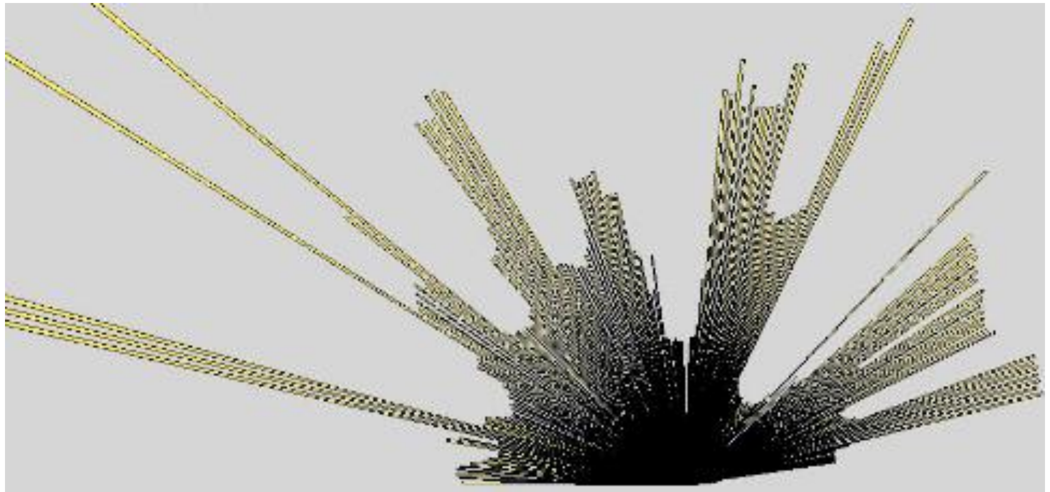
1. Beams reflected by obstacles
2. Beams reflected by persons or caused by crosstalk
3. Random measurements
4. Max range measurements



# Distance perception: Laser Range Finder

Lasers are definitely more accurate sensors

- 180 ranges over 180° (up to 360 °)
- 1 to 64 planes scanned, 10-75 scans/s
- <1cm range resolution
- Max range up to 50-80 m
- Issues with mirrors, glass, and matte black.





< 1000 €


~ 6000 €

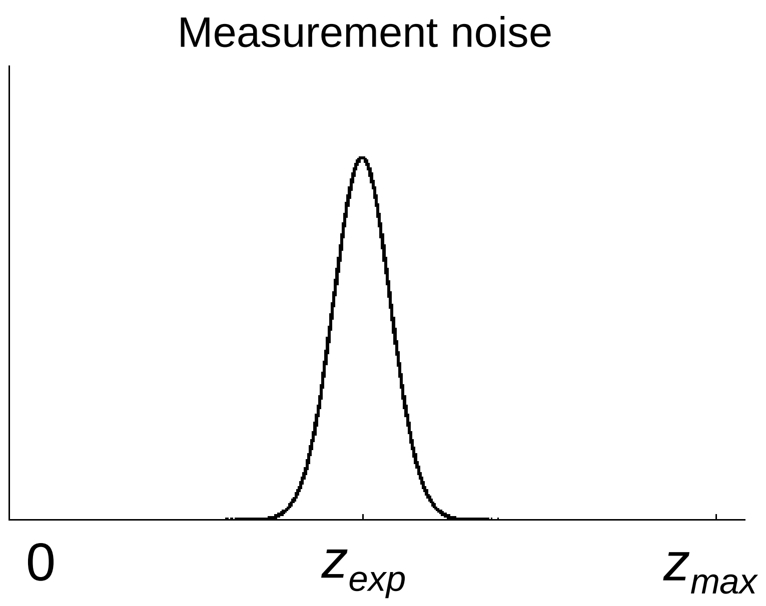

~ 10.000 €

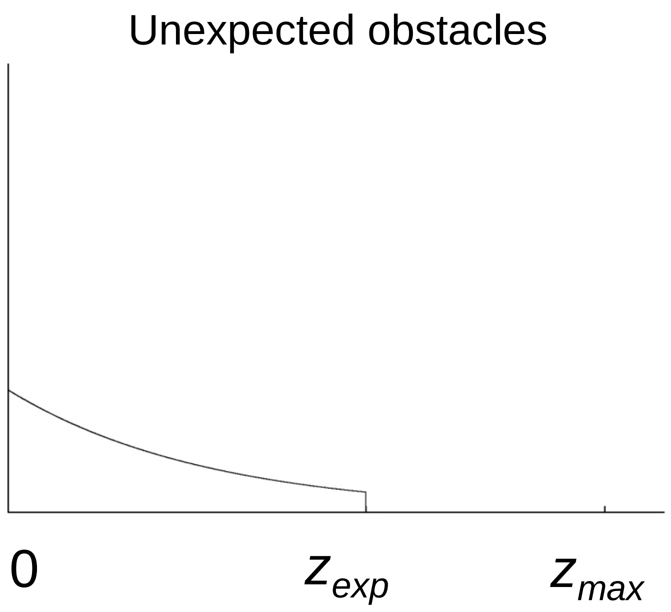
~ 40.000 €

> 80.000 €

# Beam Sensor Model (I)

The laser range finder model describe each single measurement using



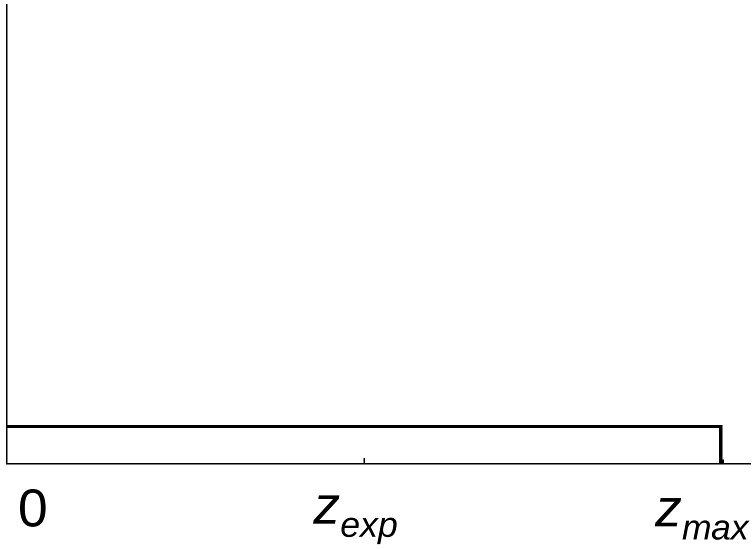

$$P_{hit}(z \mid x, m) = \eta \frac{1}{\sqrt{2\pi b}} e^{-\frac{1}{2}\frac{(z - z_{\exp})^2}{b}}$$

$$P_{\text{unexp}}(z \mid x, m) = \begin{Bmatrix} \eta\ \lambda\ \mathrm{e}^{-\lambda z} & z < z_{\text{exp}} \\ 0 & otherwise \end{Bmatrix}$$

# Beam Sensor Model (II)

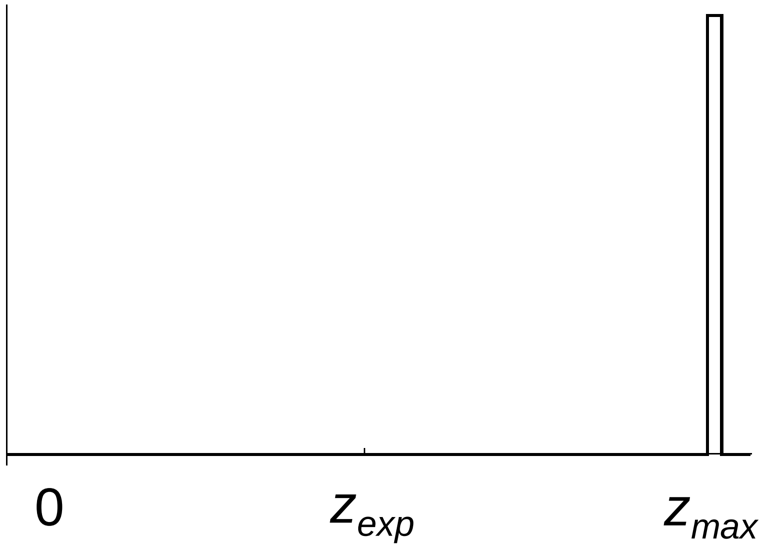
The laser range finder model describe each single measurement using

Random measurement




$$P_{rand}(z \mid x, m) = \eta \frac{1}{z_{\max}}$$

Max range

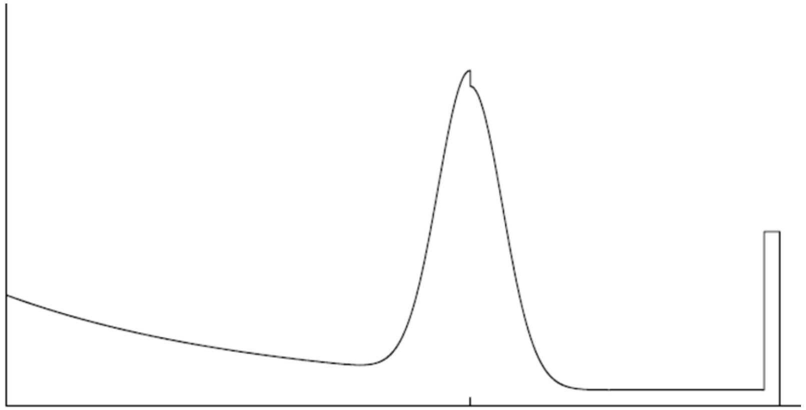


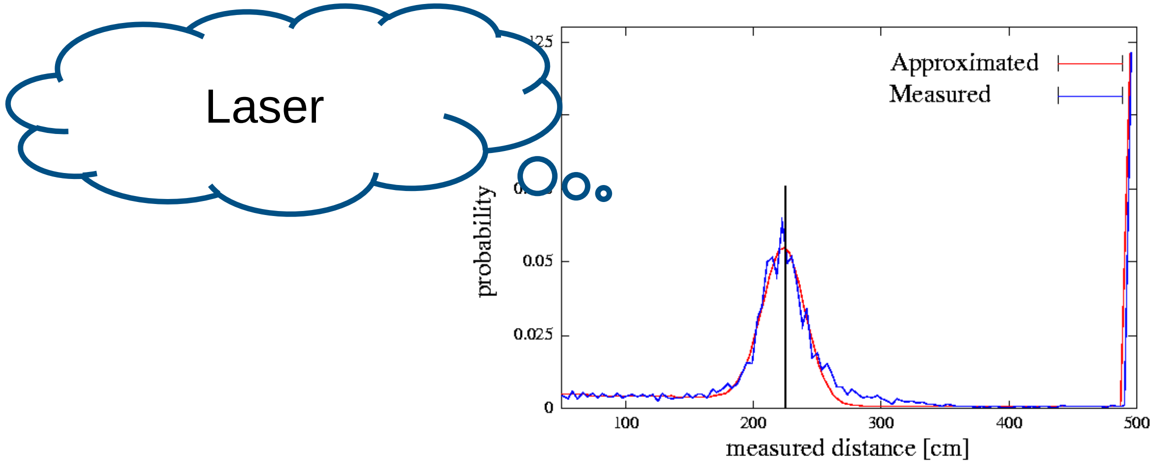

$$P_{\max}(z \mid x, m) = \eta \frac{1}{z_{small}}$$

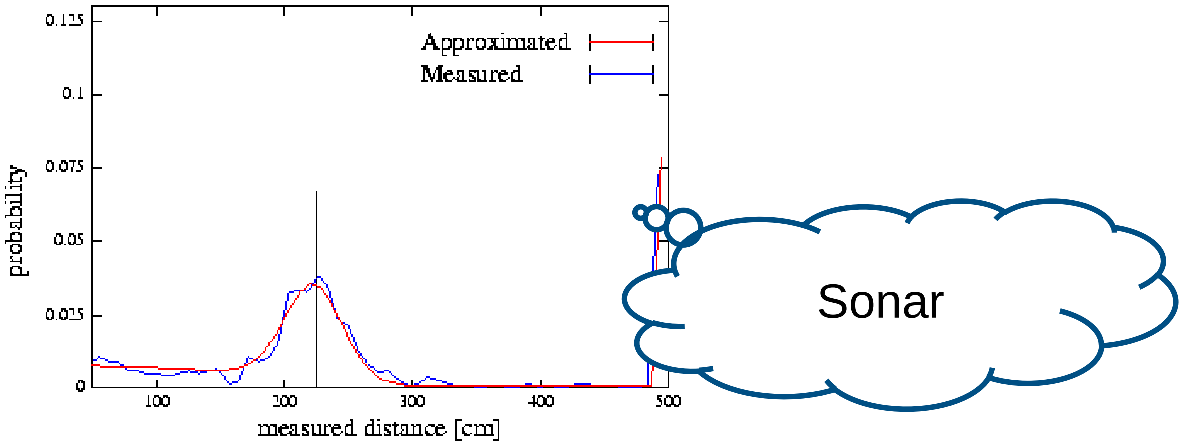
# Beam Sensor Model (III)

The laser range finder model describe each single measurement using

$$P(z \mid x, m) = \begin{pmatrix} \alpha_{\text{hit}} \\ \alpha_{\text{unexp}} \\ \alpha_{\max} \\ \alpha_{\text{rand}} \end{pmatrix}^T \cdot \begin{pmatrix} P_{\text{hit}}(z \mid x, m) \\ P_{\text{unexp}}(z \mid x, m) \\ P_{\max}(z \mid x, m) \\ P_{\text{rand}}(z \mid x, m) \end{pmatrix}$$

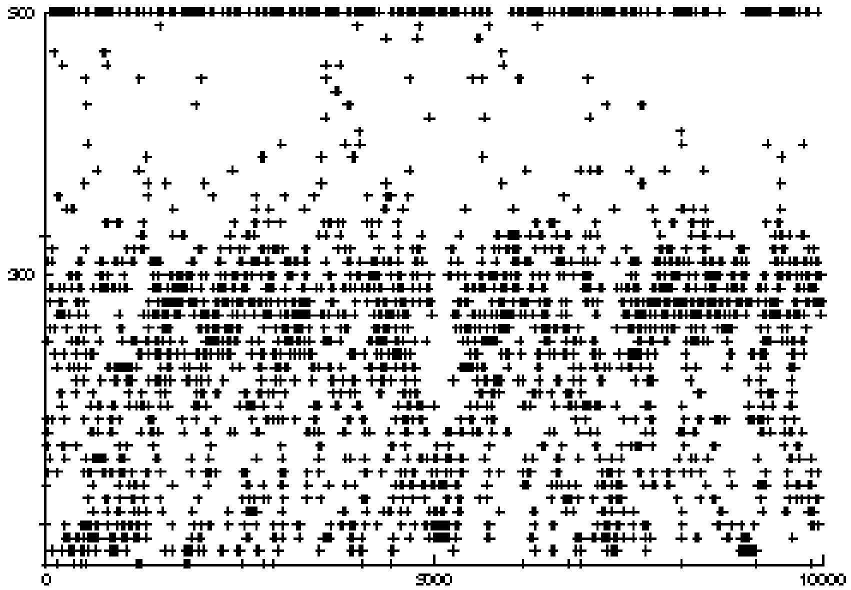




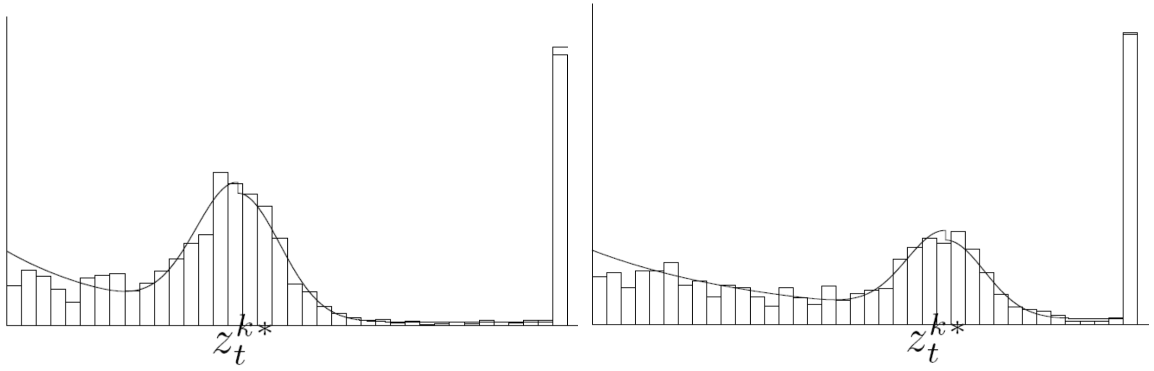



# Sensor Model Calibration (Sonar)

Acquire some data from the sensor, e.g., when the target is at 300 cm and 400 cm



Sonar

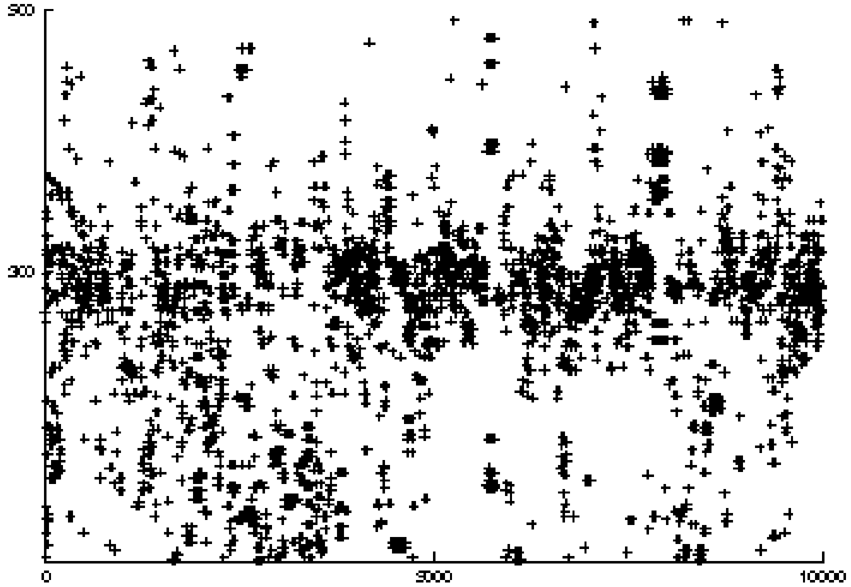



300 cm 400 cm

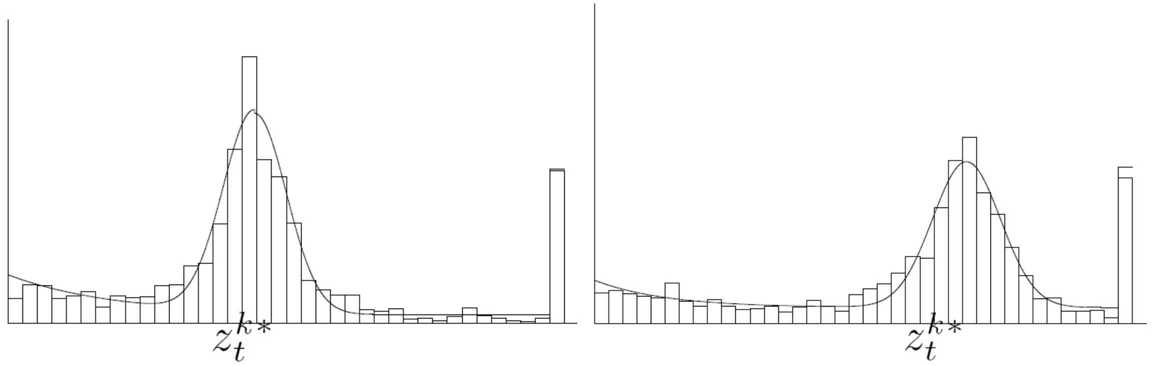
Then estimate the model parameters via maximum likelihood: $P(z \mid z_{\exp})$

# Sensor Model Calibration (Laser)

Acquire some data from the sensor, e.g., when the target is at 300 cm and 400 cm




Laser



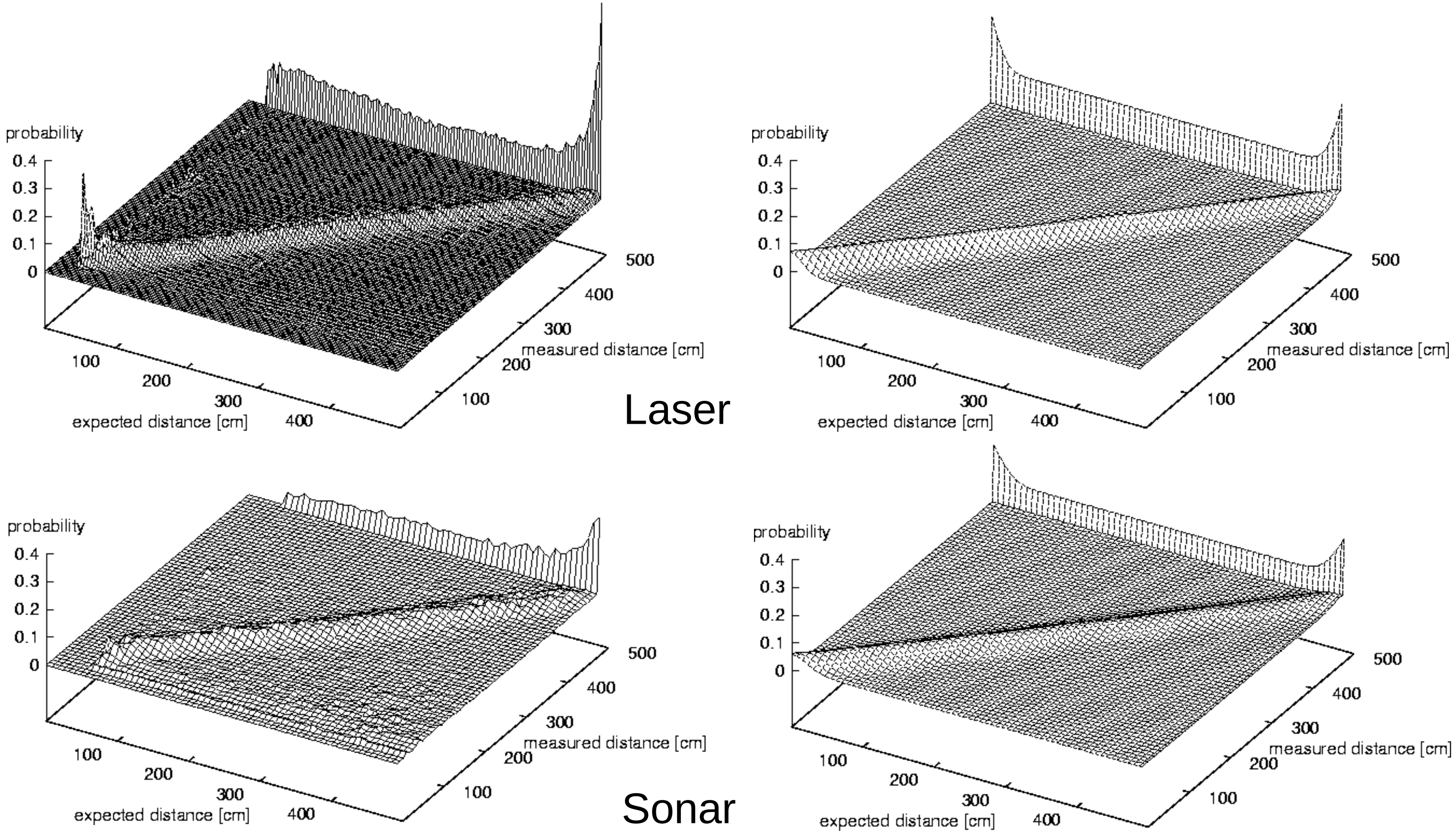

300 cm

400 cm

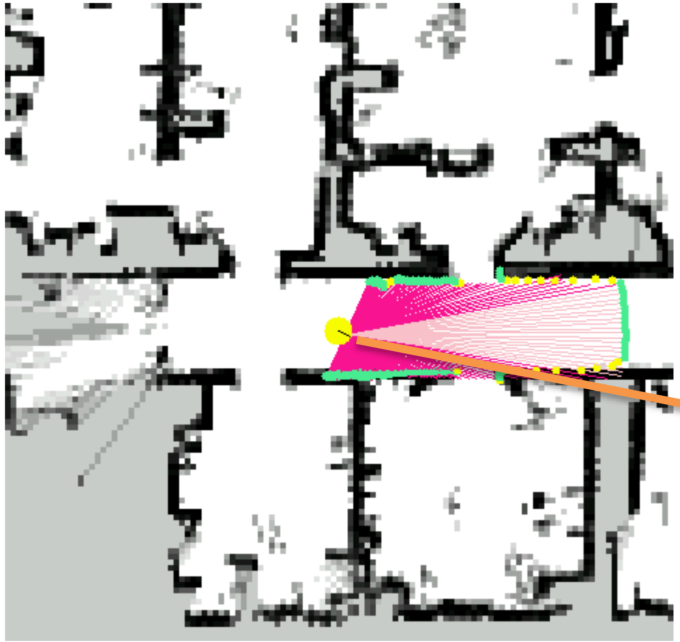
Then estimate the model parameters via maximum likelihood: $P(z \mid z_{\exp})$

# Discete Model for Range Sensor

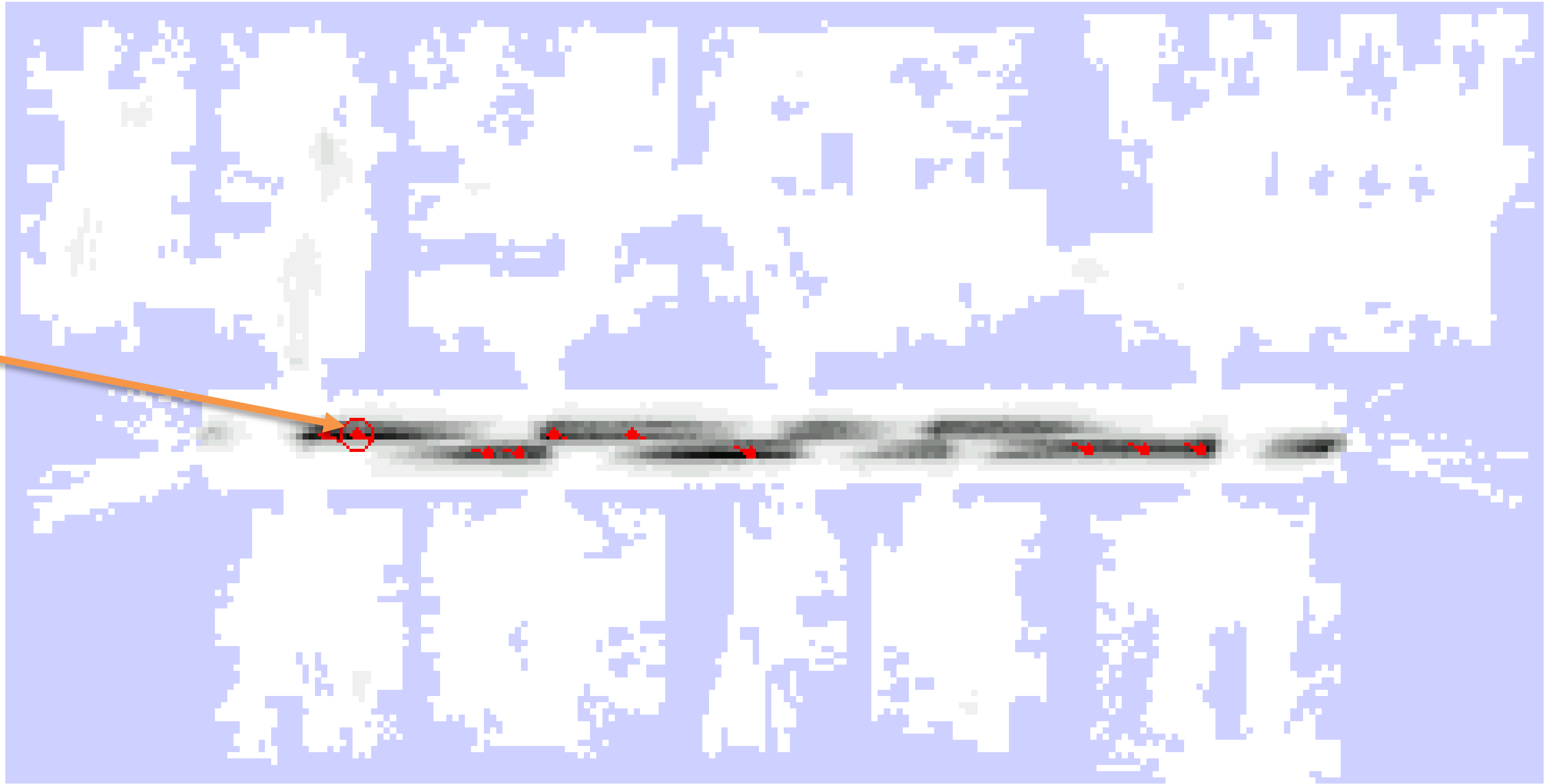
Instead of densities, consider discrete steps along the sensor beam



# Sensor Model Likelihood



$z$



$P(z|x,m)$

# Scan Sensor Model

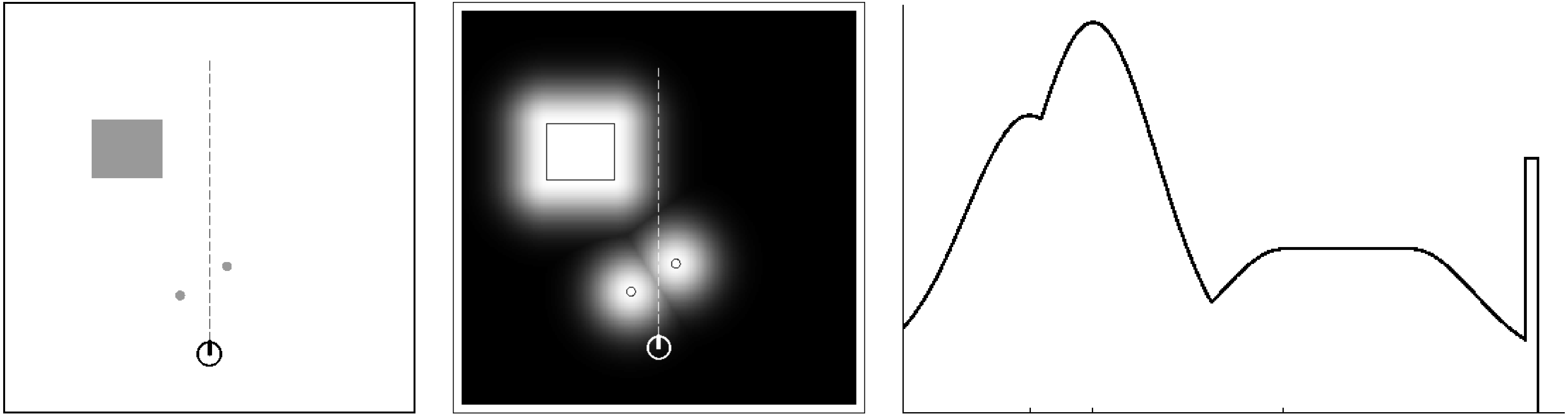
The beam sensor model assumes independence between beams and between physical causes of measurements and has some issues:

- Overconfident because of independency assumptions
- Need to learn parameters from data
- A different model should be learned for different angles w.r.t. obstacles
- Inefficient because it uses ray tracing

The Scan sensor model simplifies with:

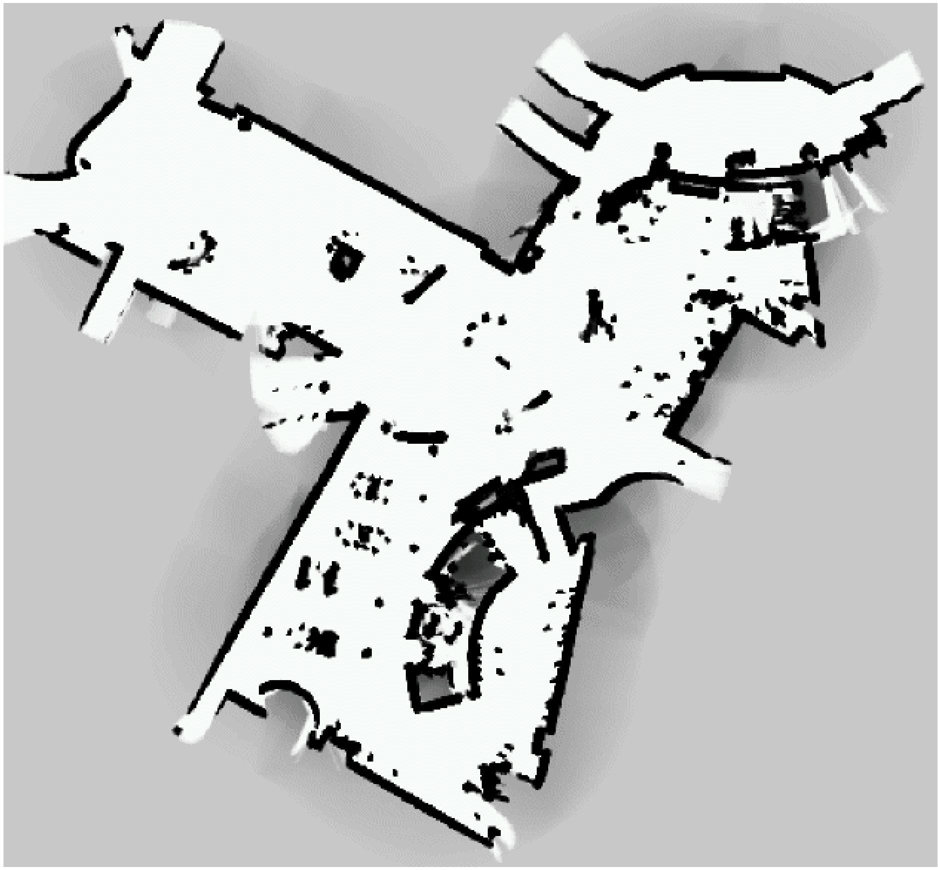
- Gaussian distribution with mean at distance to closest obstacle,
- Uniform distribution for random measurements, and
- Small uniform distribution for max range measurements

# Scan Sensor Model Example

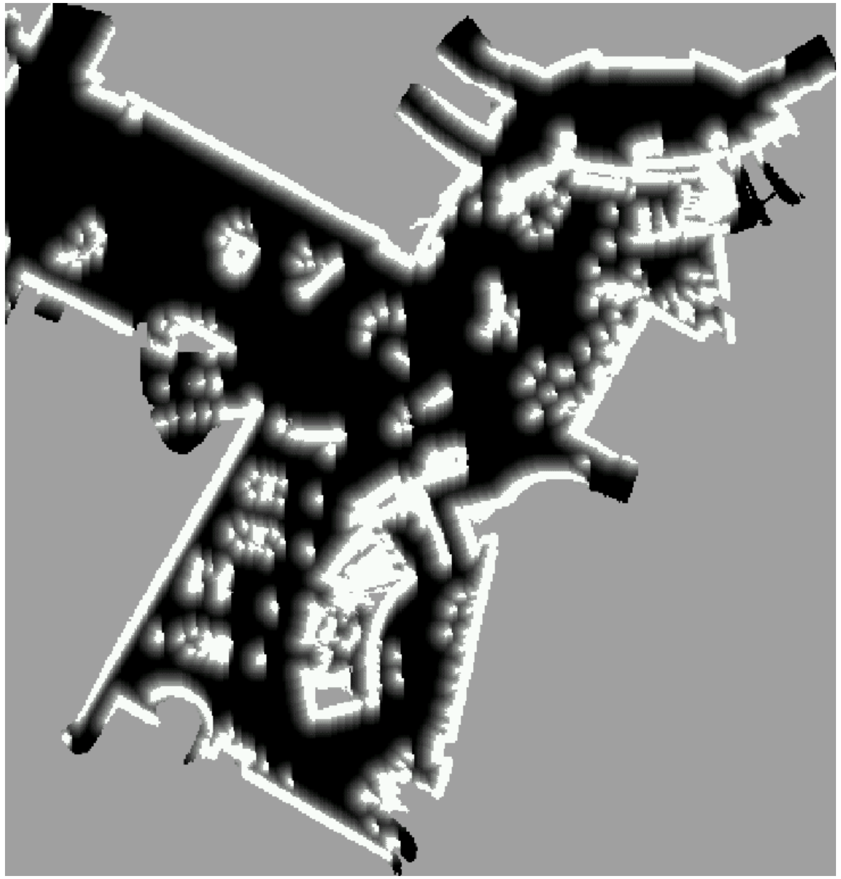


Map *m* | Likelihood field | *P(z|x,m)*

# San Jose Tech Museum



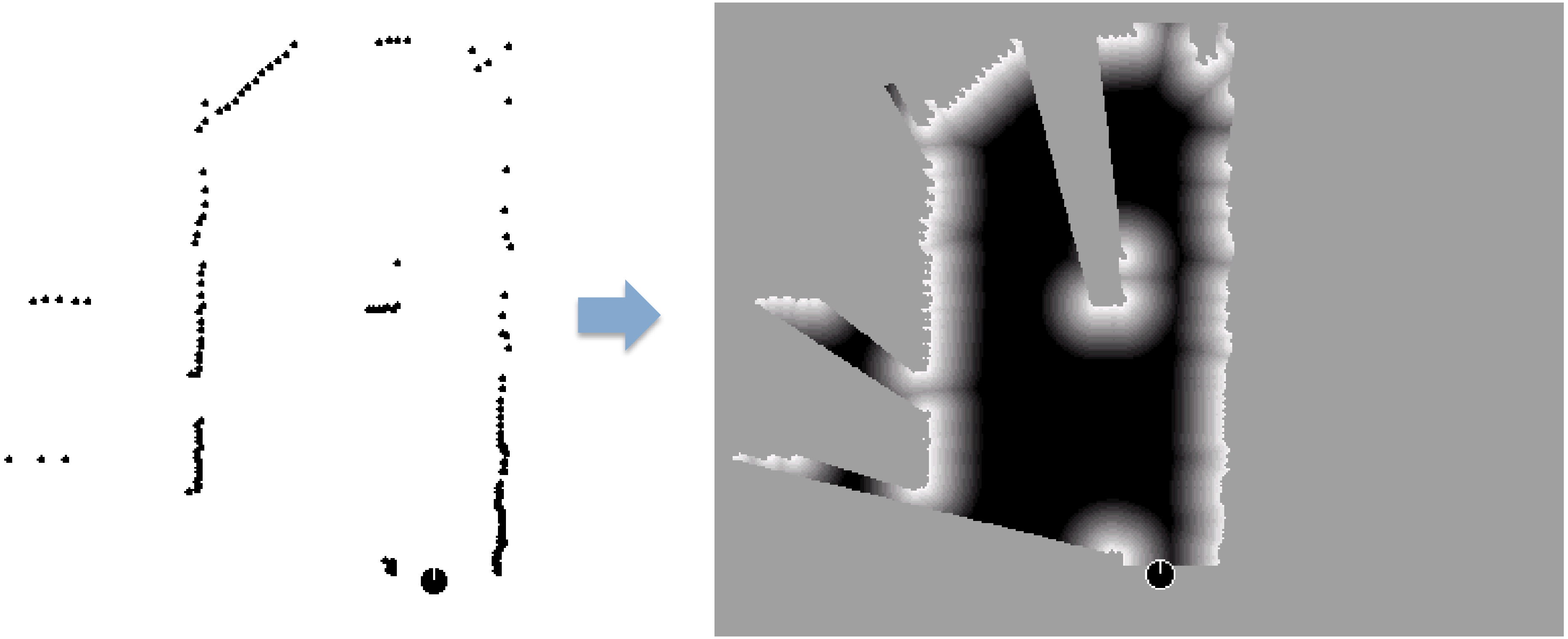
Occupancy grid map



Likelihood field

# Scan Sensor Matching

Extract likelihood field from scan and use it to match different scan:

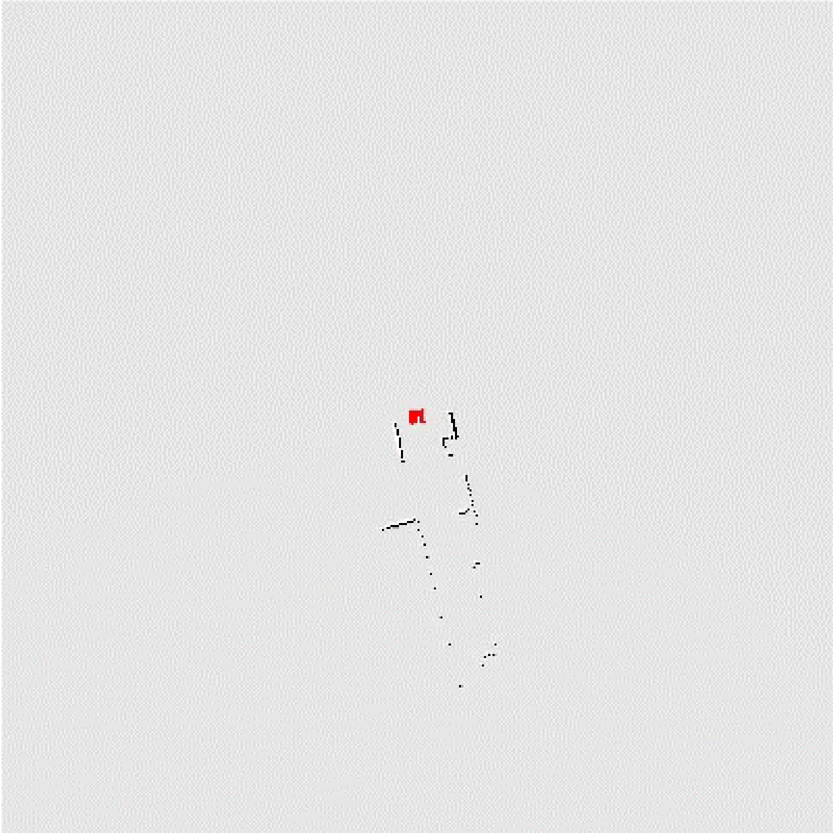


# Scan Sensor Matching

Extract likelihood field from scan and use it to match different scan:

- Highly efficient, uses 2D tables only.
- Smooth with respect to small changes in robot position
- Allows gradient descent (scan matching)
- Ignores physical properties of beams.

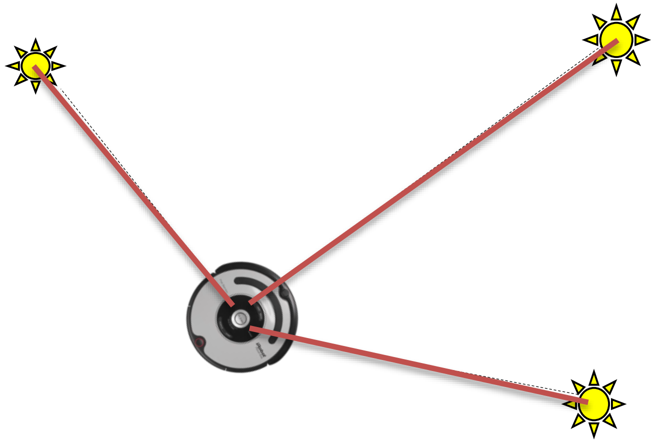
However it does not work with sonars

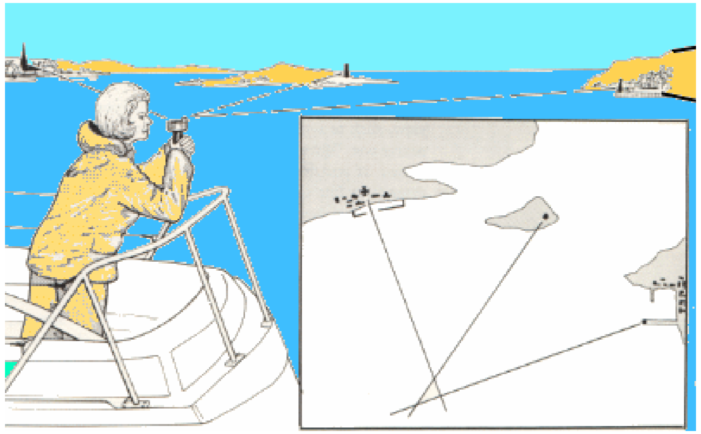


# Landmarks

Landmark sensors provides

- Distance (or)
- Bearing (or)
- Distance and bearing

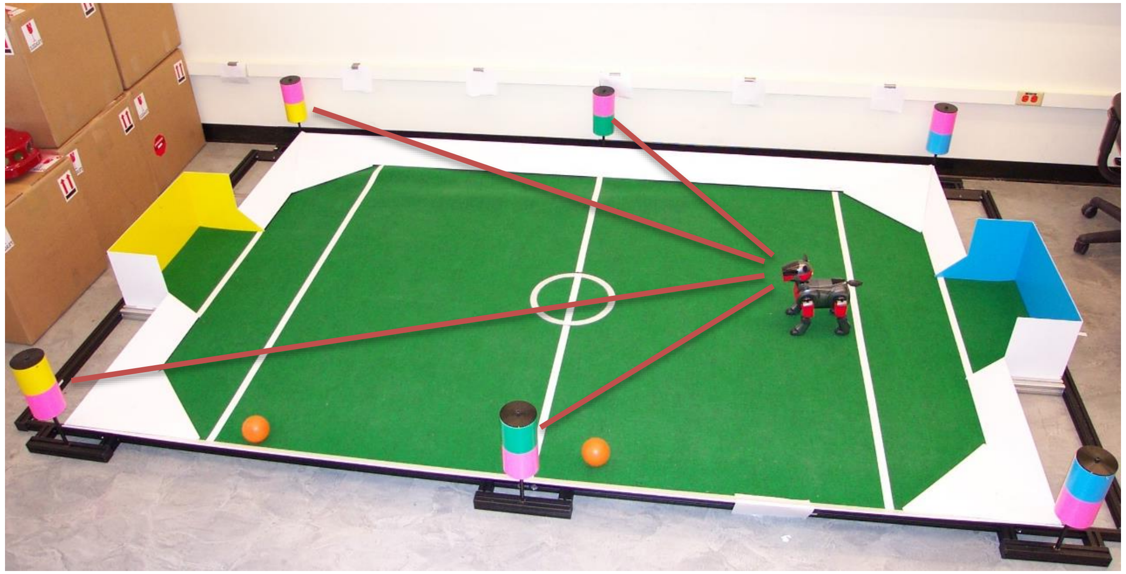




Can be obtained via

- Active beacons (e.g., radio, GPS)
- Passive (e.g., visual, retro-reflective)

Standard approach is triangulation



# Landmark Models with Uncertainty

Explicitly modeling uncertainty in sensing is key to robustness:

- Determine parametric model of noise free measurement
- Analyze sources of noise
- Add adequate noise to parameters (eventually mix in densities for noise)
- Learn (and verify) parameters by fitting model to data

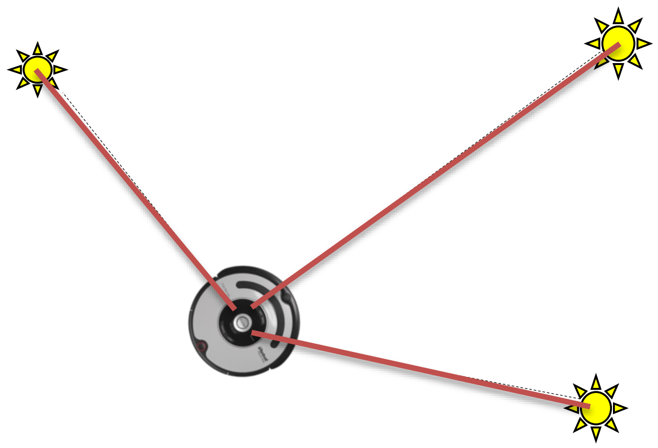
The likelihood of measurement is given by “probabilistically comparing” the actual with the expected measurement.

# Landmark Detection Model

For landmak $i$ in map $m$ the measurement $z = (i, d, \alpha)$ for a robot at $(x, y, \theta)$ is given by

$$\hat{d} = \sqrt{(m_x(i) - x)^2 + (m_y(i) - y)^2}$$

$$\hat{a} = \mathrm{atan2}(m_y(i) - y, m_x(i) - x) - \theta$$



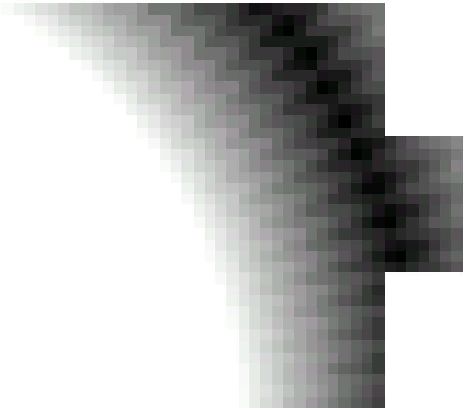
Detection probability might depend on the distance/bearing

$$p_{\mathrm{det}} = \mathrm{prob}(\hat{d} - d, \varepsilon_d) \cdot \mathrm{prob}(\hat{\alpha} - \alpha, \varepsilon_\alpha)$$

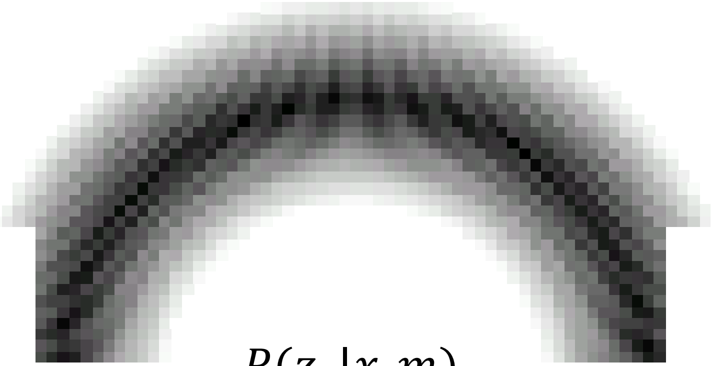
Then we have to take into account false positives too

$$z_{\mathrm{det}} p_{\mathrm{det}} + z_{\mathrm{fp}} P_{\mathrm{uniform}}(z \mid x, m)$$

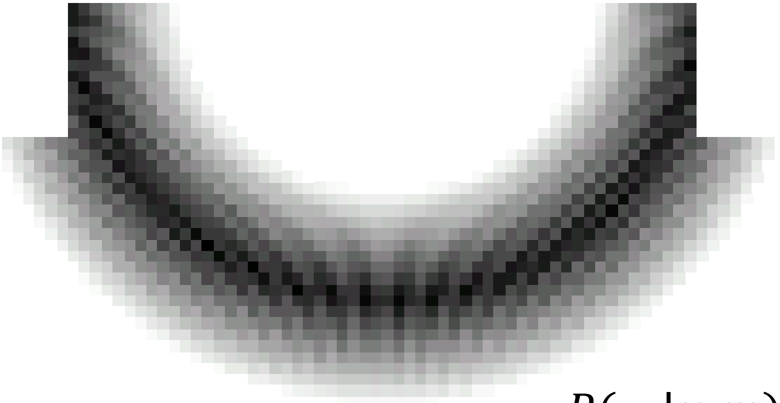
# RoboCup Example



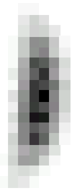
$P(z_1|x,m)$



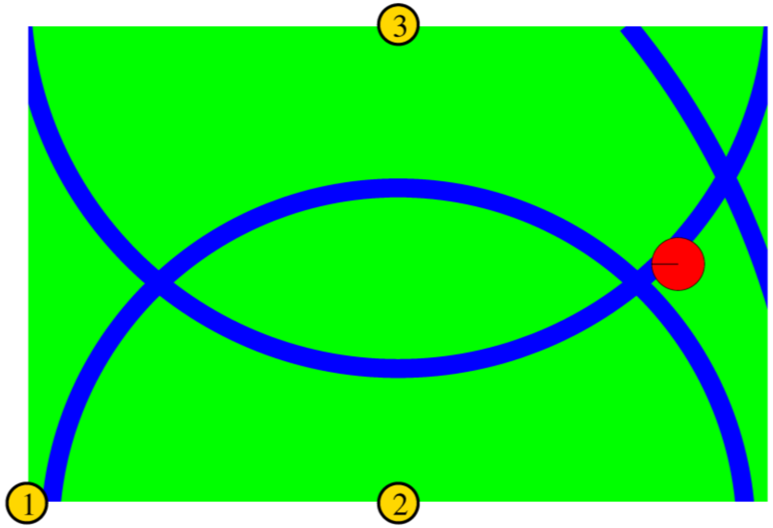
$P(z_2|x,m)$



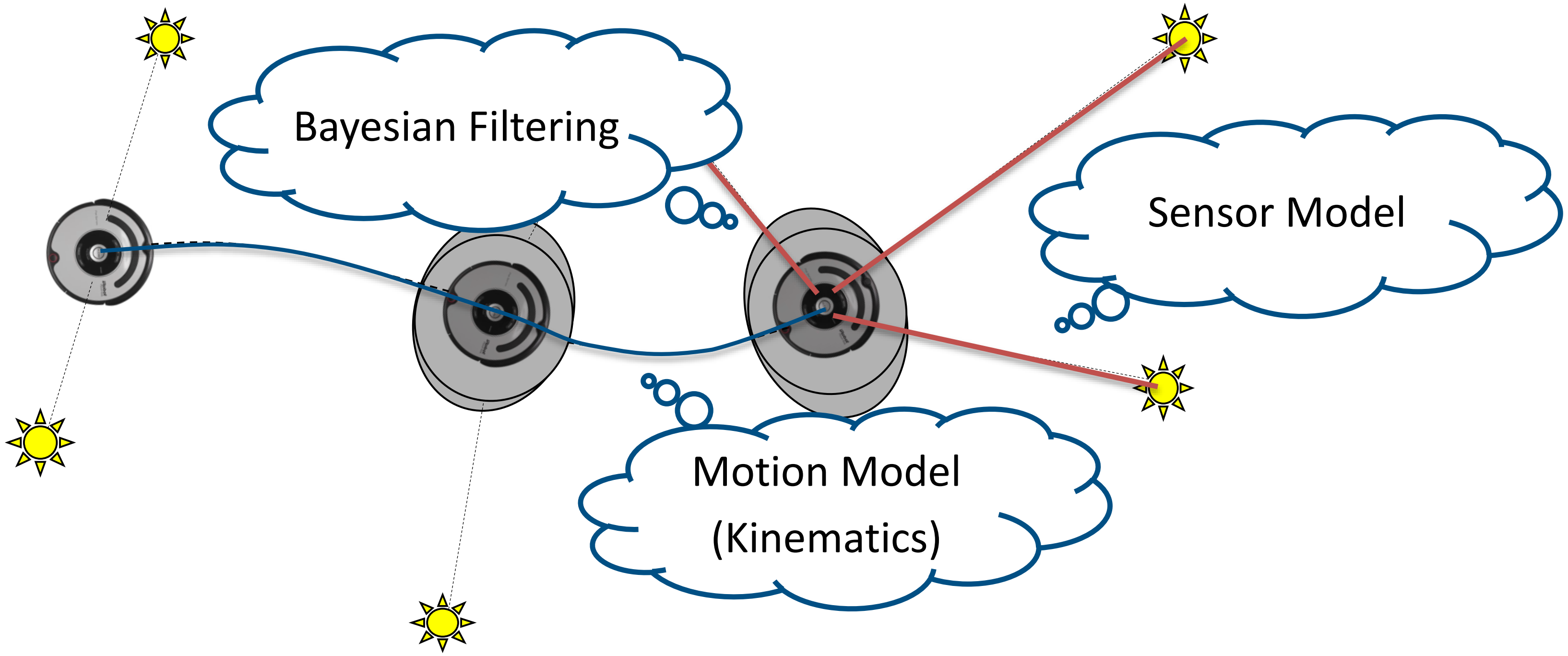
$P(z_3|x,m)$



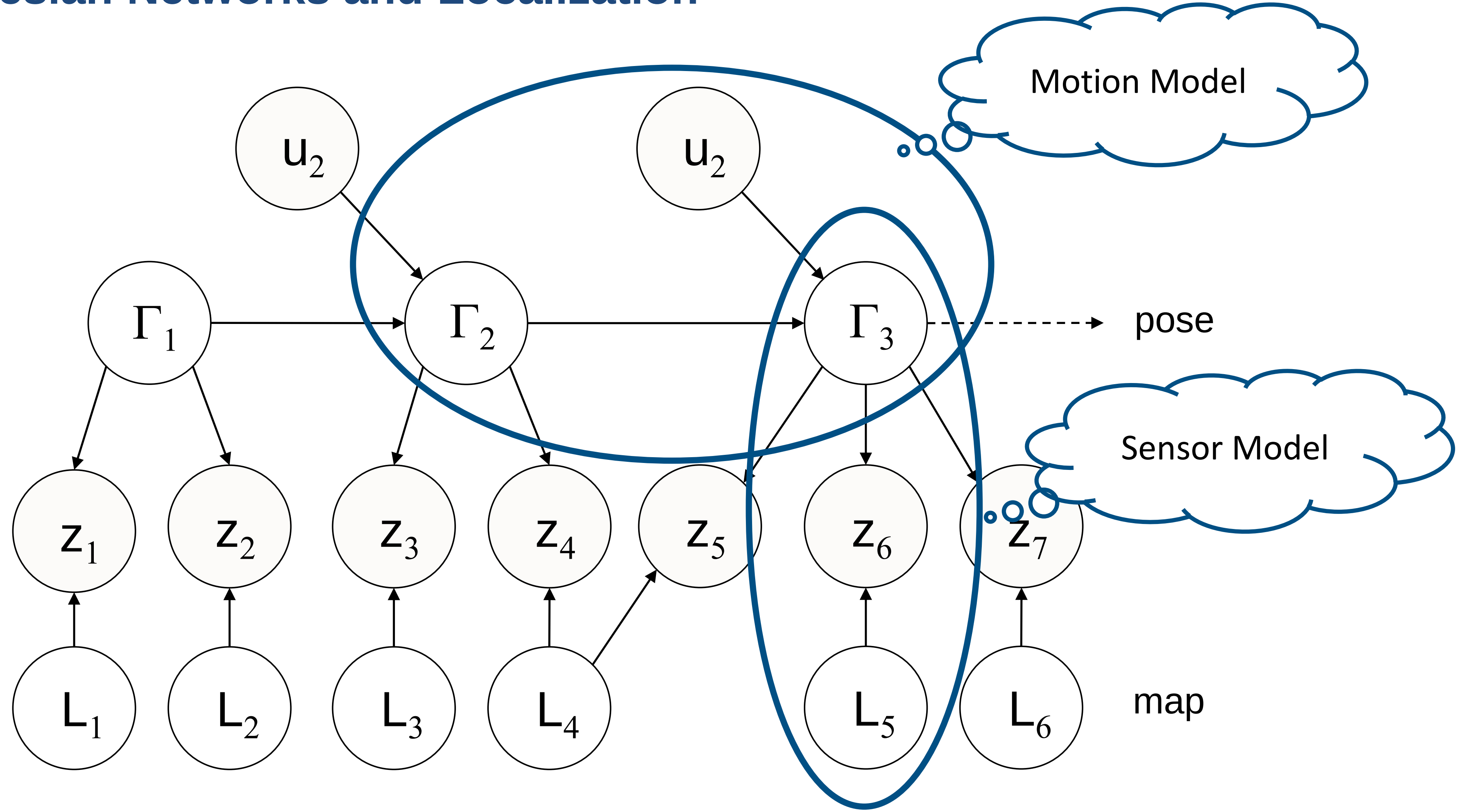
$P(z_1, z_2, z_3|x,m)$



# Localization with Knowm Map



# Dynamic Bayesian Networks and Localization




$$\text{Filtering}: \quad p(\Gamma_t \mid Z_{1:t}, U_{1:t}, l_1, \ldots, l_N) = \iiint_{1:t-1} p(\Gamma_{1:t} \mid Z_{1:t}, U_{1:t}, l_1, \ldots, l_N)$$

## Bayesian Filtering Framework

We want to compute an estimate of the posterios  probabibility of robot state $x_t$

$$Bel(x_t) = P(x_t \mid u_1, z_1 \ldots, u_t, z_t)$$

from the stream of information about movement and sensors

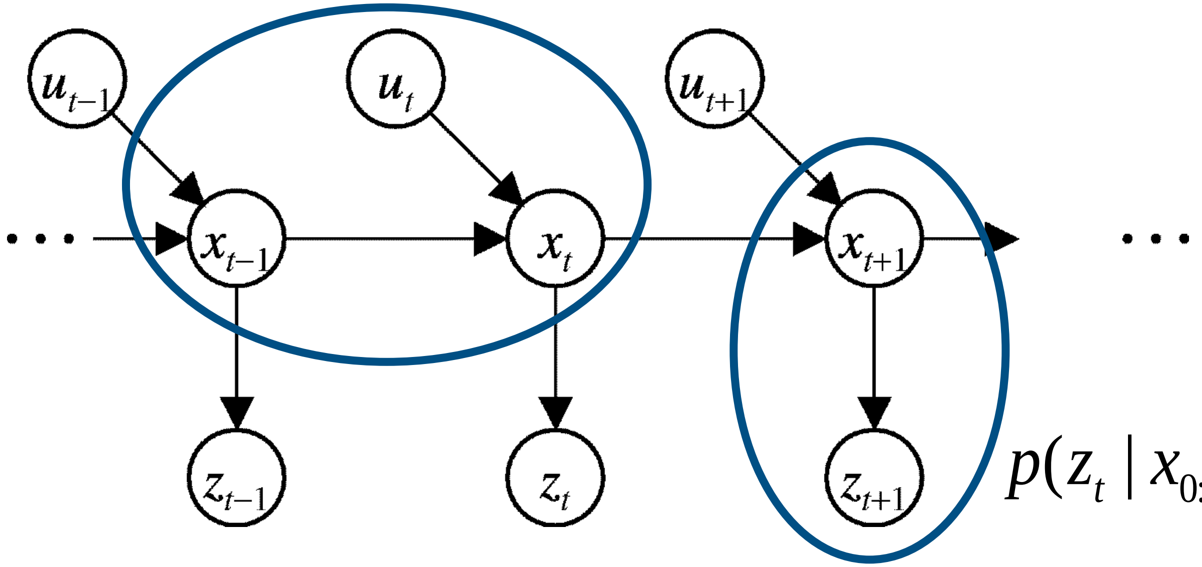
$$d_t = \{u_1, z_1 \ldots, u_t, z_t\}$$

In particular we assume known:

- The prior probability of the system state $P(x_0)$
- The motion model $P(x'|x, u)$
- The sensor model $P(z|x, m)$

# Markov Assumptions

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




$$p(z_t \mid x_{0:t}, z_{1:t}, u_{1:t}) = p(z_t \mid x_t)$$

Underlining assumption behind Bayes filtering:

- Perfect model, no approximation errors
- Static and stationary world
- Independent noise

# Bayes Filters

z = observation
$u$ = action
$x$ = state
m = map

$$\boxed{Bel(x_t)} = P(x_t \mid u_1, z_1 \ldots, u_t, z_t, m)$$

Bayes
$$= \eta \; P(z_t \mid x_t, u_1, z_1, \ldots, u_t, m) \; P(x_t \mid u_1, z_1, \ldots, u_t, m)$$

Markov
$$= \eta \; P(z_t \mid x_t, m) \; P(x_t \mid u_1, z_1, \ldots, u_t, m)$$

Total prob.
$$= \eta \; P(z_t \mid x_t, m) \int P(x_t \mid u_1, z_1, \ldots, u_t, x_{t-1}, m) \; P(x_{t-1} \mid u_1, z_1, \ldots, u_t, m) \; dx_{t-1}$$

Markov
$$= \eta \; P(z_t \mid x_t, m) \int P(x_t \mid u_t, x_{t-1}) \; P(x_{t-1} \mid u_1, z_1, \ldots, u_t, m) \; dx_{t-1}$$

Markov
$$= \eta P(z_t \mid x_t, m) \int P(x_t \mid u_t, x_{t-1}) \, P(x_{t-1} \mid u_1, z_1, \ldots, z_{t-1}, m) \; dx_{t-1}$$

$$\boxed{= \eta \; P(z_t \mid x_t, m) \int P(x_t \mid u_t, x_{t-1}) \; Bel(x_{t-1}) \; dx_{t-1}}$$

# Bayes Filter Algorithm

$$Bel(x_t) = \eta \; P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1}) \; Bel(x_{t-1}) \; dx_{t-1}$$

**How to represent such belief?**

Algorithm Bayes_filter( *Bel(x), d* ):

$\eta$=0

if *d* is a perceptual data item *z* then

For all *x* do

$$Bel'(x) = P(z \mid x) Bel(x)$$

$$\eta = \eta + Bel'(x)$$

For all *x* do

$$Bel'(x) = \eta^{-1} Bel'(x)$$

else if *d* is an action data item *u* then

For all *x* do

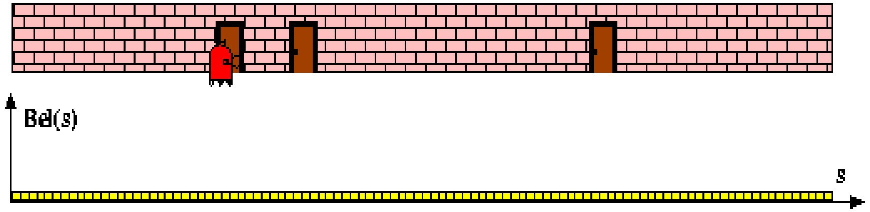
$$Bel'(x) = \int P(x \mid u, x') \; Bel(x') \; dx'$$

return *Bel'(x)*

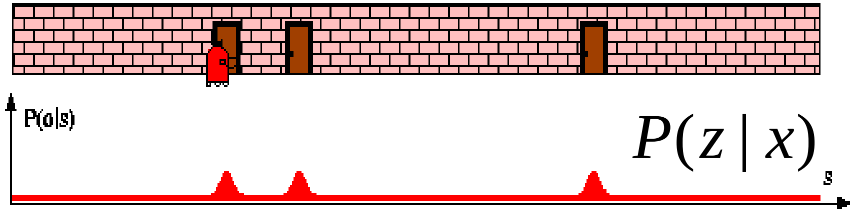
Based on such representation:

- Discrete filters
- Kalman filters
- Sigma-point filters
- Particle filters
- ...

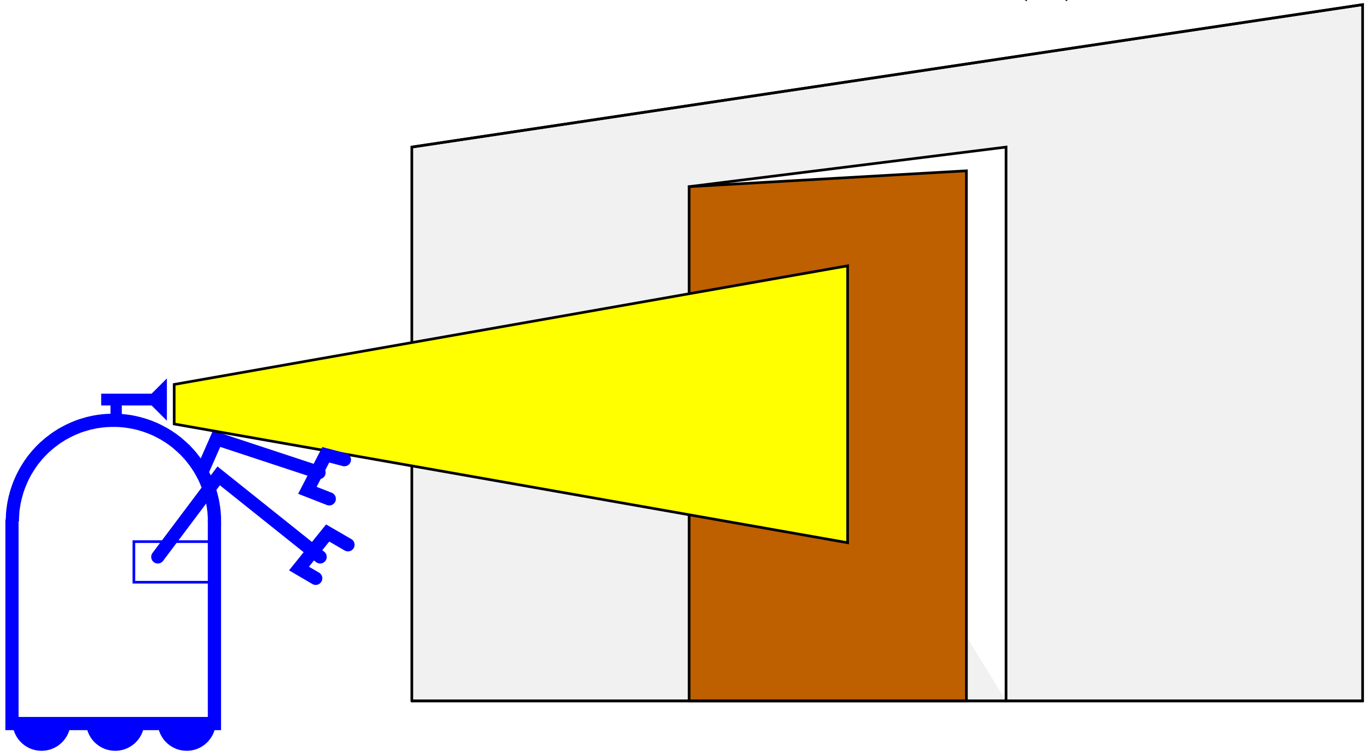
# Piecewise Constant Approximation



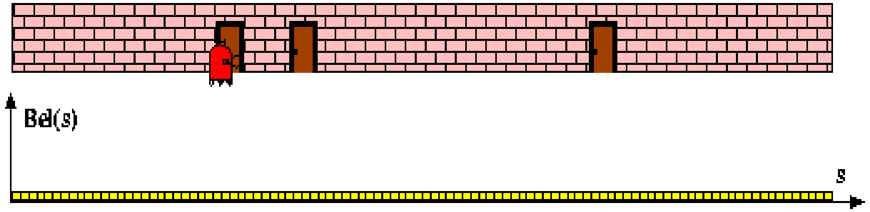

$Bel(x)$



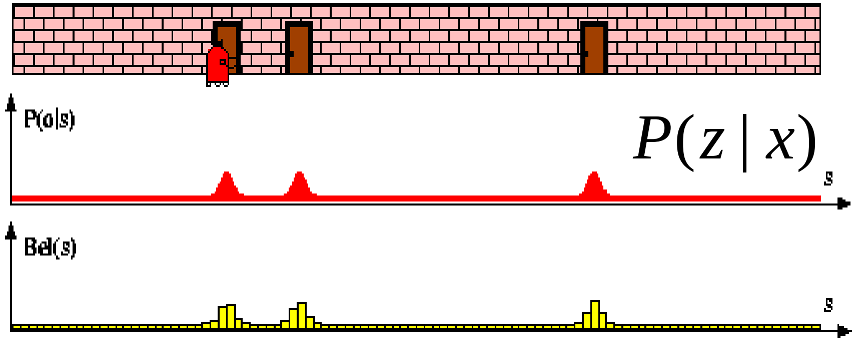

$P(z \mid x)$



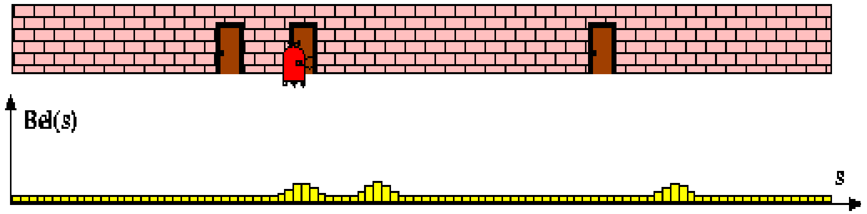
# Piecewise Constant Approximation



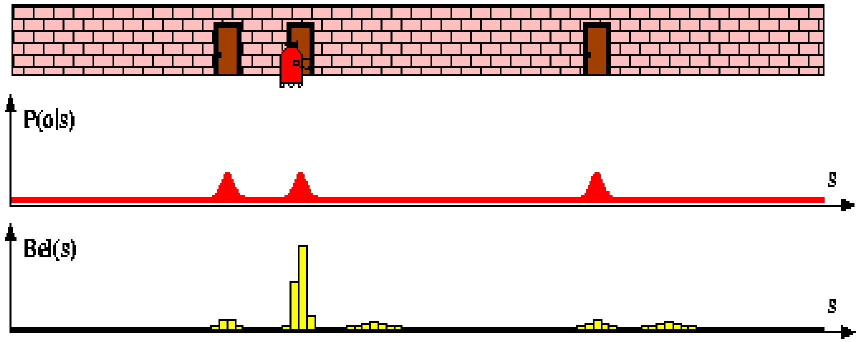

$$Bel(x_0)$$



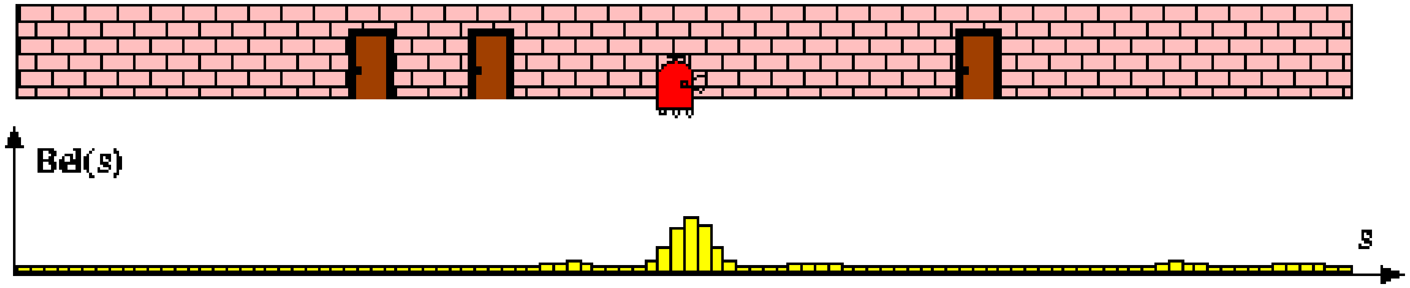

$$Bel'(x_0) = P(z \mid x) Bel(x_0)$$




$$Bel(x_1) = P(x_1 \mid x_0, u_1)$$




$$Bel'(x_1) = P(z \mid x_1) Bel(x_1)$$



# Discrete Bayesian Filter Algorithm

Algorithm Discrete_Bayes_filter( *Bel(x),d* ):

h=0

If *d* is a perceptual data item *z* then

For all *x* do

$$Bel'(x) = P(z \mid x) Bel(x)$$
$$\eta = \eta + Bel'(x)$$

For all *x* do

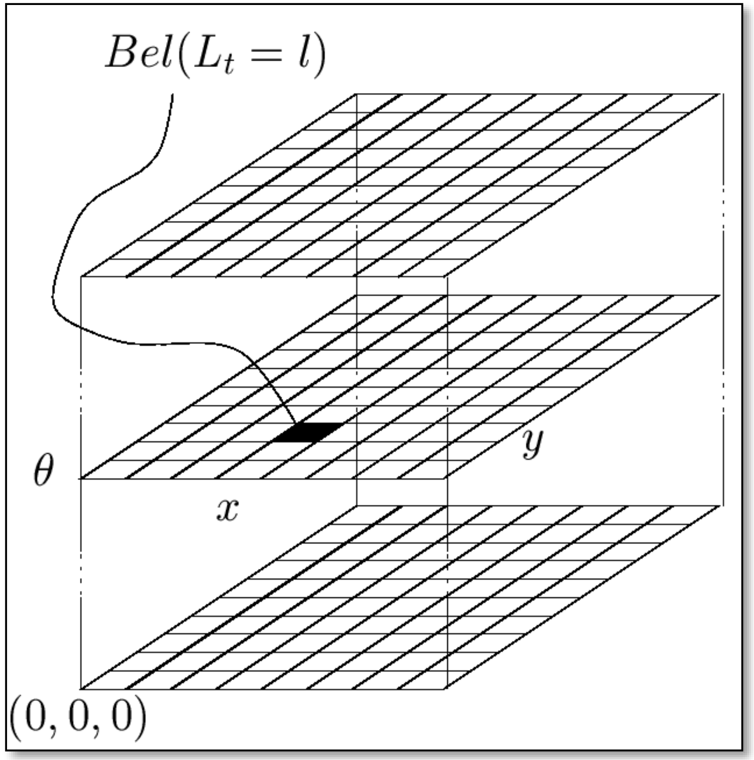
$$Bel'(x) = \eta^{-1} Bel'(x)$$

Else if *d* is an action data item *u* then

For all *x* do

$$Bel'(x) = \sum_{x'} P(x \mid u, x')\, Bel(x')$$

Return *Bel'(x)*



# Tis and Tricks

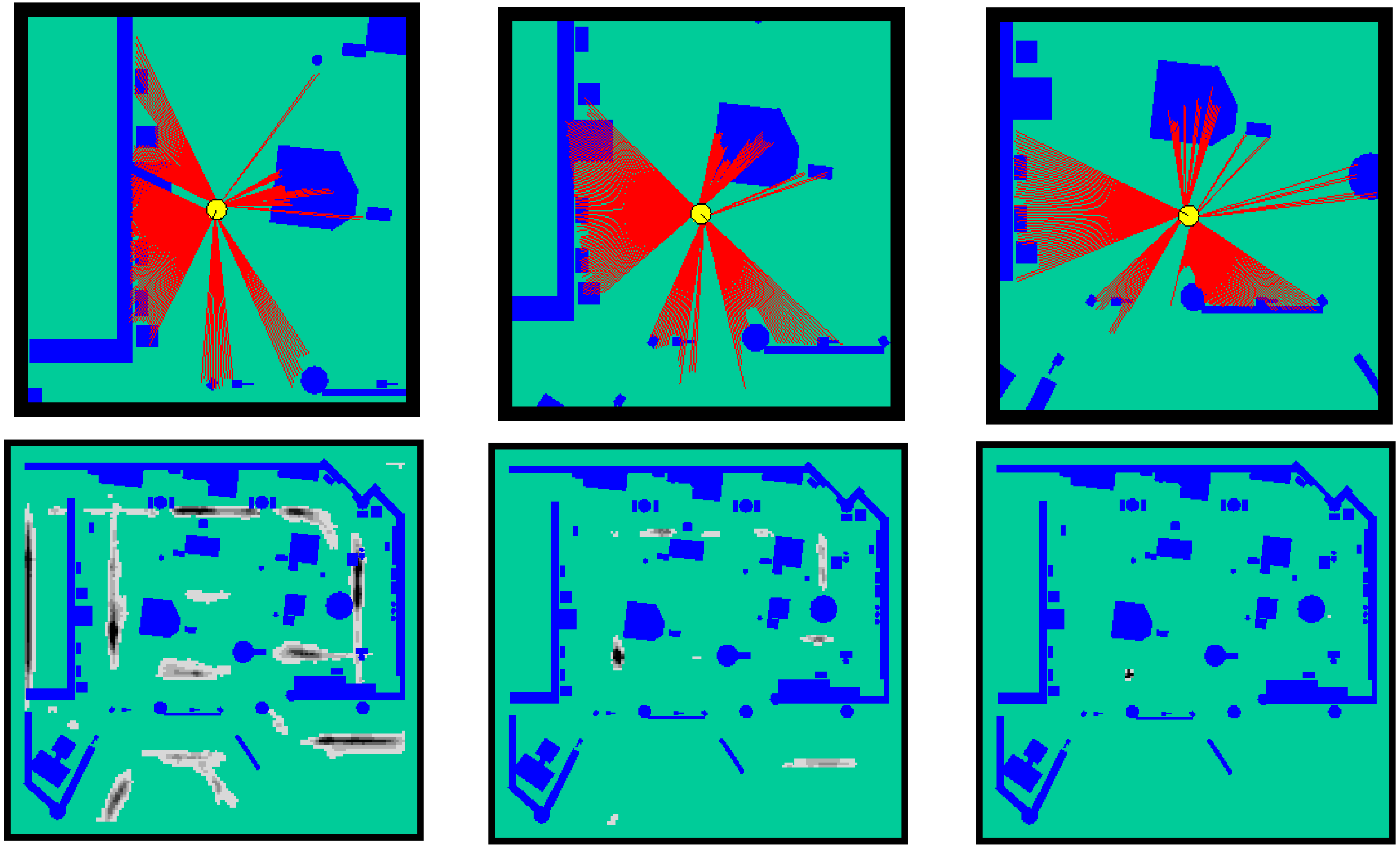
Belief update upon sensory input and normalization iterates over all cells

- When the belief is peaked (e.g., during position tracking), avoid updating irrelevant parts.
- Do not update entire sub-spaces of the state space and monitor whether the robot is de-localized or not by considering likelihood of observations given the active components

To update the belief upon robot motions, assumes a bounded Gaussian model to reduce the update from $O(n^2)$ to $O(n)$

- Update by shifting the data in the grid according to measured motion
- Then convolve the grid using a Gaussian Kernel.

# Grid Based Localization



## Bayes Filter Algorithm

$$Bel(x_t) = \eta \; P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1}) \; Bel(x_{t-1}) \; dx_{t-1}$$

*How to represent such belief?*

Algorithm Bayes_filter( *Bel(x), d* ):

$\eta$=0

If *d* is a perceptual data item *z* then

For all *x* do

$$Bel'(x) = P(z \mid x) Bel(x)$$

$$\eta = \eta + Bel'(x)$$

For all *x* do

$$Bel'(x) = \eta^{-1} Bel'(x)$$

Else if *d* is an action data item *u* then

For all *x* do

$$Bel'(x) = \int P(x \mid u, x') \; Bel(x') \; dx'$$

Return *Bel'(x)*

Based on such representation:

- Discrete filters
- Kalman filters
- Sigma-point filters
- Particle filters
- ...

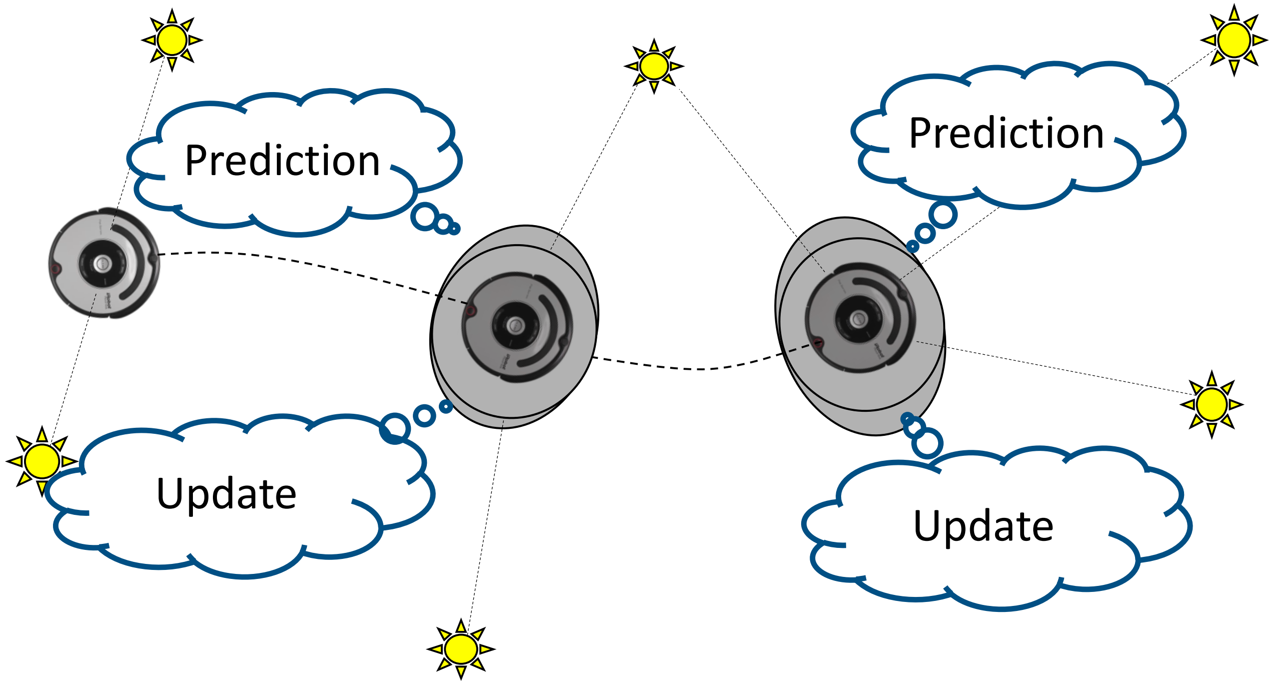
# Bayes Filter Reminder

$$\boxed{Bel(x_t) = \eta \; P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1}) \; Bel(x_{t-1}) \; dx_{t-1}}$$

Prediction: $$\overline{Bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}) \; Bel(x_{t-1}) \; dx_{t-1}$$

Correction/Update: $$Bel(x_t) = \eta \; p(z_t \mid x_t) \, \overline{Bel}(x_t)$$

# Localization with Knowm Map



# Bayes Filter Reminder

$$Bel(x_t) = \eta \; P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1}) \; Bel(x_{t-1}) \; dx_{t-1}$$

Prediction: $$\overline{Bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}) \; Bel(x_{t-1}) \; dx_{t-1}$$

Correction/Update: $$Bel(x_t) = \eta \; p(z_t \mid x_t) \, \overline{Bel}(x_t)$$

Can we compute the integrals ($\eta$ is an integral too) in closed form for continuos distributions?

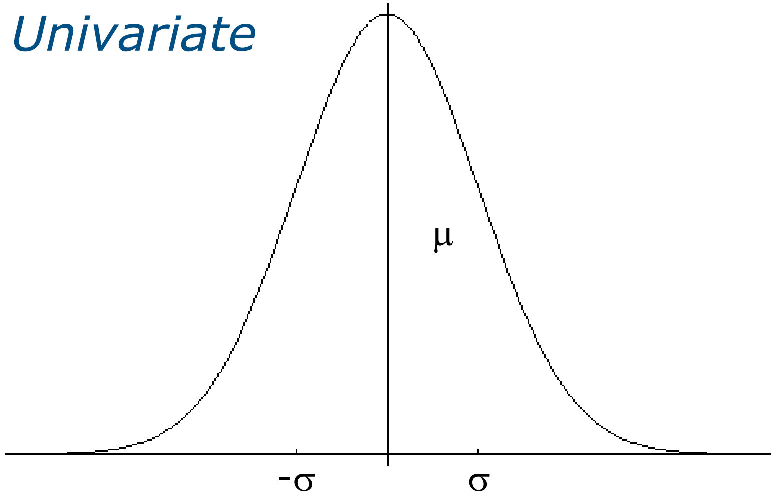
NO!

Is there any continuous distribution for which this is possible?

YES!

# Gaussian Distribution

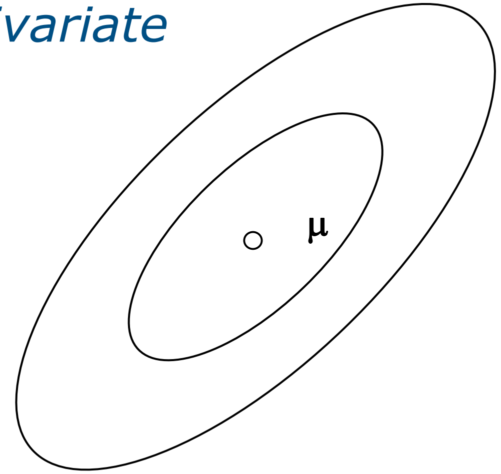
*Univariate*




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

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

*Multivariate*

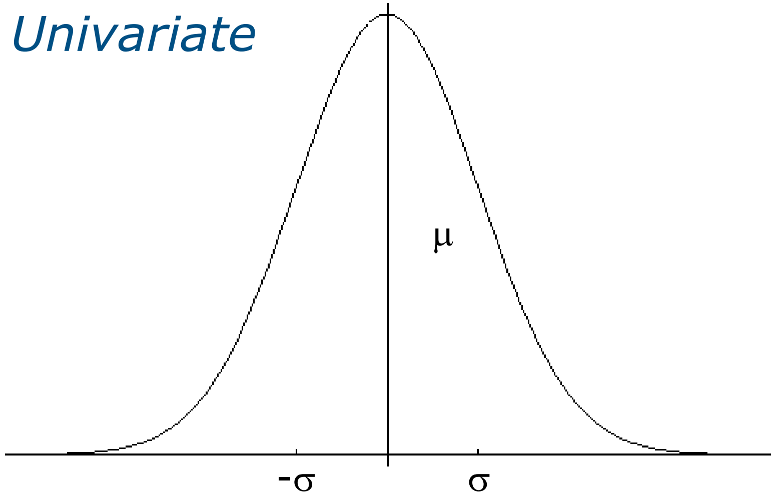



$$p(\mathbf{x}) \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

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

# Properties of Gaussian Distribution

*Univariate*




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

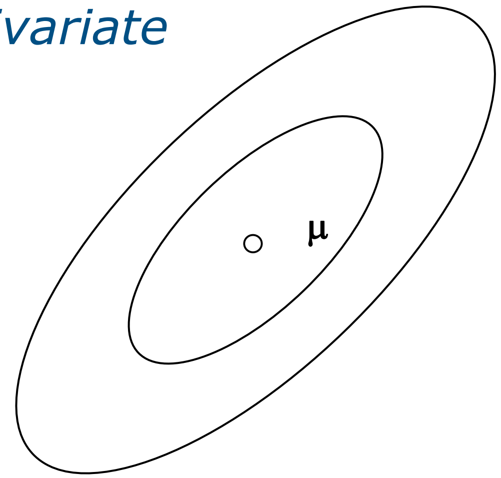
$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}$$

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

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

# Properties of Gaussian Distribution

*Multivariate*




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

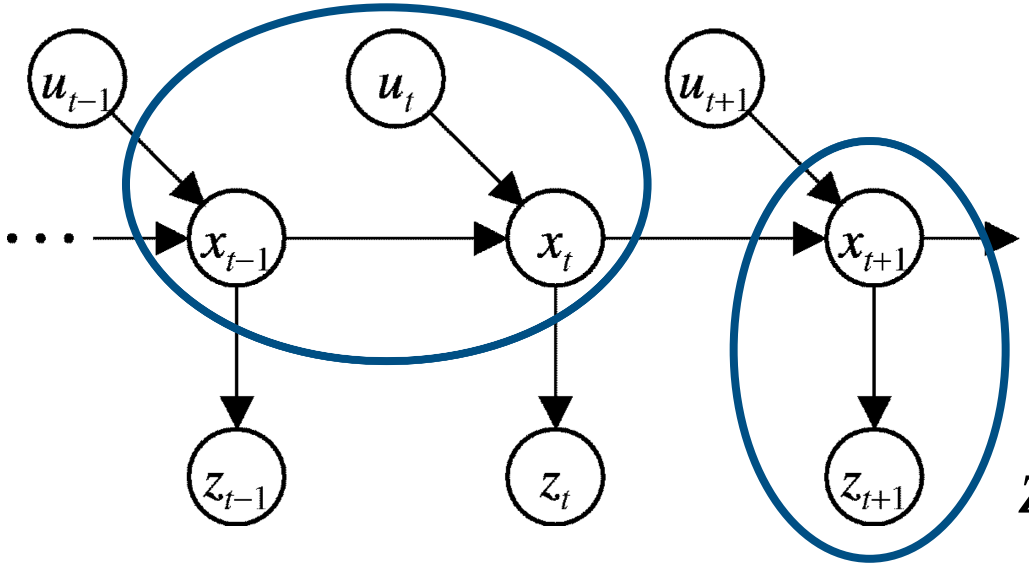
$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^t \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$$

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

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

# Discrete Time Kalman Filter

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$$

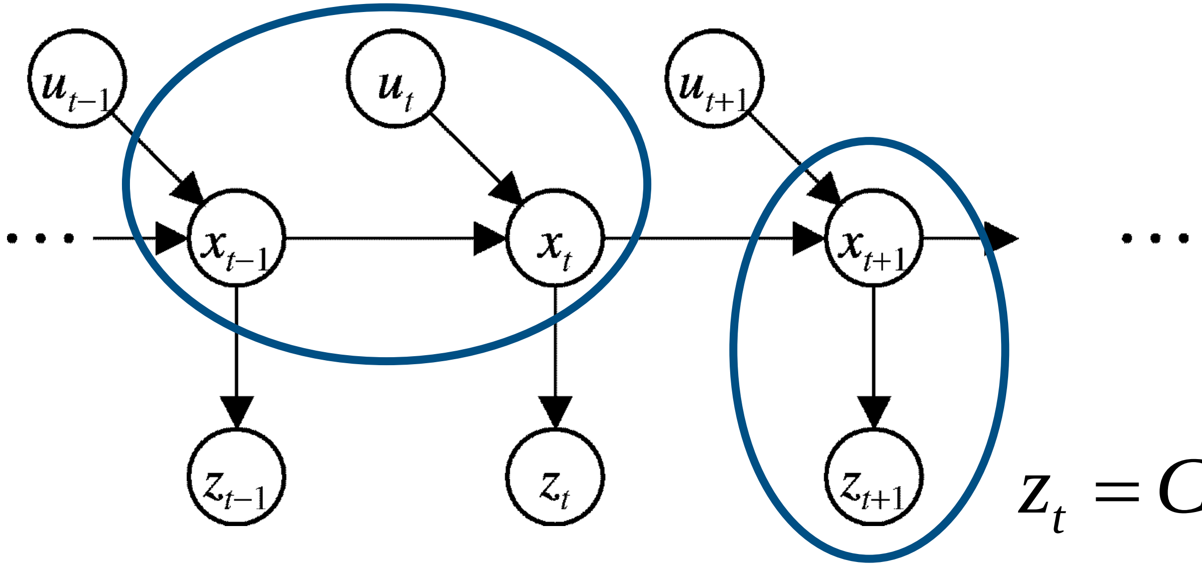



$$z_t = C_t x_t + \delta_t$$

- $A_t$ (n x n) describes how state evolves from t to t-1 w/o controls or noise
- $B_t$ (n x l) describes how control $u_t$ changes the state from t to t-1
- $C_t$ (k x n) describes how to map the state $x_t$ to an observation $z_t$
- $\varepsilon_t$ $\delta_t$ random variables representing process and measurement noise assumed independent and normally distributed with covariance $R_t$ and $Q_t$ respectively.

# Linear Gaussian Systems

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$$




$$z_t = C_t x_t + \delta_t$$

Initial belief is normally distributed: $Bel(x_0) = N\left(x_0; \mu_0, \Sigma_0\right)$

Dynamics are linear function of state and control plus additive noise:

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t \quad \Rightarrow \quad p(x_t \mid u_t, x_{t-1}) = N\left(x_t; A_t x_{t-1} + B_t u_t, R_t\right)$$

Observations are linear function of state plus additive noise:

$$z_t = C_t x_t + \delta_t \quad \Rightarrow \quad p(z_t \mid x_t) = N\left(z_t; C_t x_t, Q_t\right)$$

# Linear Gaussian System: Prediction

Prediction:

$$\overline{Bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}) \quad \cdot \quad Bel(x_{t-1})\, dx_{t-1}$$

$$\sim N\left(x_t; A_t x_{t-1} + B_t u_t, R_t\right) \quad \sim N\left(x_{t-1}; \mu_{t-1}, \Sigma_{t-1}\right)$$

$$\overline{Bel}(x_t) = \eta \int \exp\left\{-\frac{1}{2}(x_t - A_t x_{t-1} - B_t u_t)^T R_t^{-1}(x_t - A_t x_{t-1} - B_t u_t)\right\}$$

$$\exp\left\{-\frac{1}{2}(x_{t-1} - \mu_{t-1})^T \Sigma_{t-1}^{-1}(x_{t-1} - \mu_{t-1})\right\} dx_{t-1}$$

$$\overline{Bel}(x_t) = \begin{cases} \bar{\mu}_t = A_t \mu_{t-1} + B_t u_t \\ \bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t \end{cases}$$

Closed form
prediction step

# Linear Gaussian System: Observation

Update: $$Bel(x_t) = \underbrace{\eta \quad p(z_t \mid x_t)}_{\sim N(z_t; C_t x_t, Q_t)} \cdot \underbrace{\overline{bel}(x_t)}_{\sim N\left(x_t; \bar{\mu}_t, \bar{\Sigma}_t\right)}$$

$$Bel(x_t) = \eta \exp\left\{-\frac{1}{2}(z_t - C_t x_t)^T Q_t^{-1}(z_t - C_t x_t)\right\} \exp\left\{-\frac{1}{2}(x_t - \bar{\mu}_t)^T \bar{\Sigma}_t^{-1}(x_t - \bar{\mu}_t)\right\}$$

$$Bel(x_t) = \begin{cases} \mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t) \\ \Sigma_t = (I - K_t C_t)\bar{\Sigma}_t \end{cases} \quad \text{with} \quad K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$

Closed form update step

# Kalman Filter Algorithm

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

Prediction:

$$\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$$

$$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$$

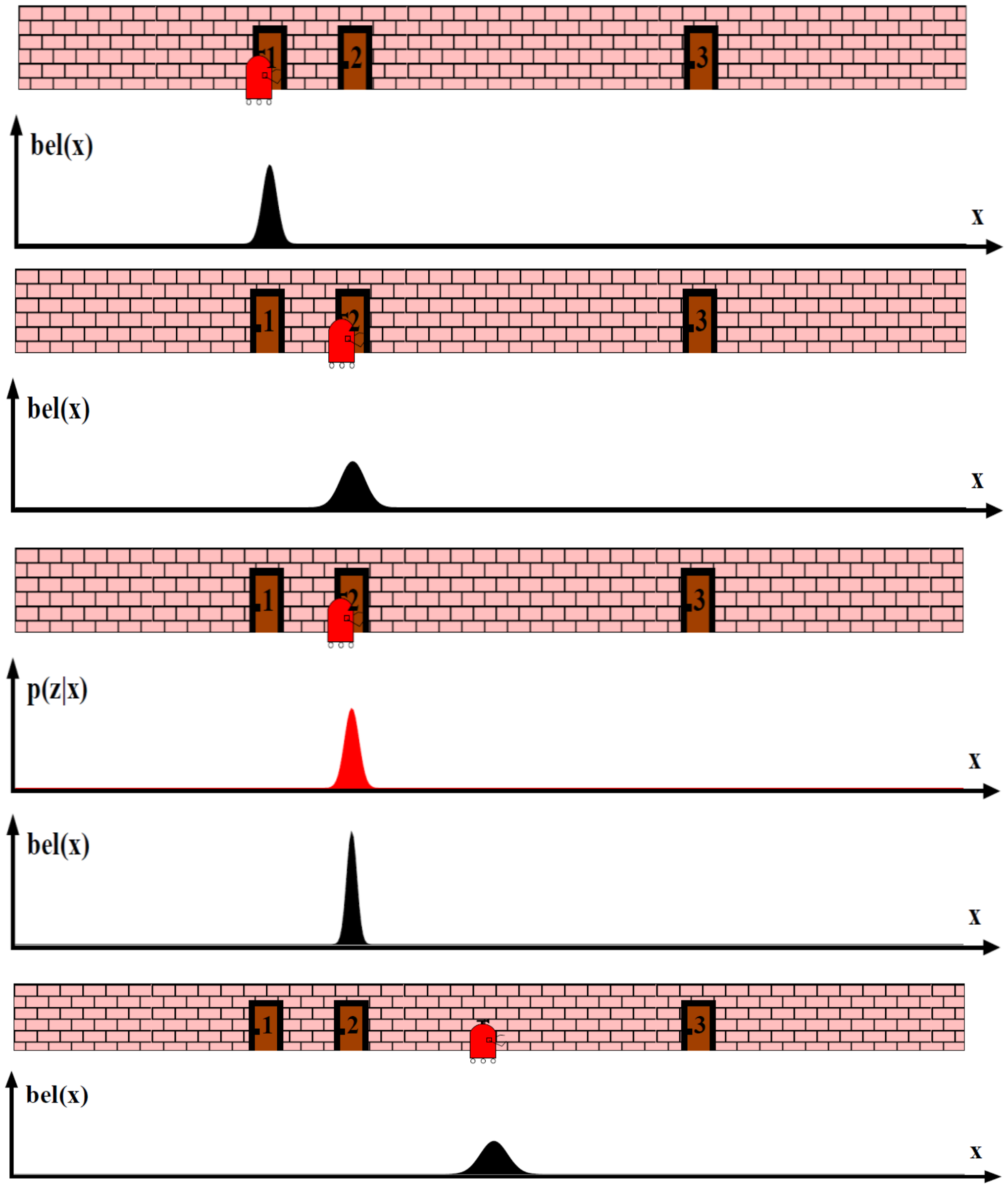
Correction:

$$K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$

$$\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$$

$$\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$$

Return $\mu_t$, $\Sigma_t$



# Kalman Filter Algorithm

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

Prediction:

$$\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$$

$$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$$

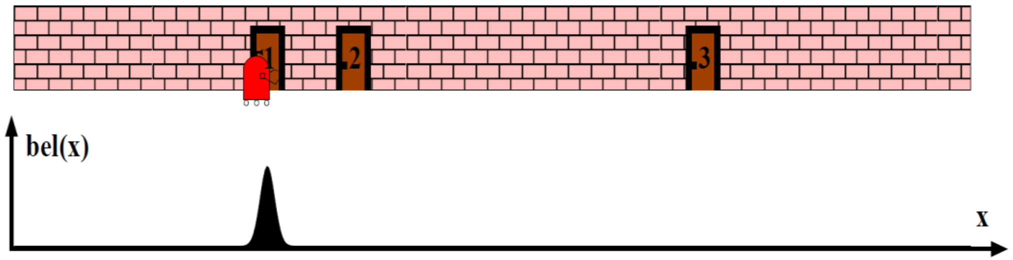
Correction:

$$K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$

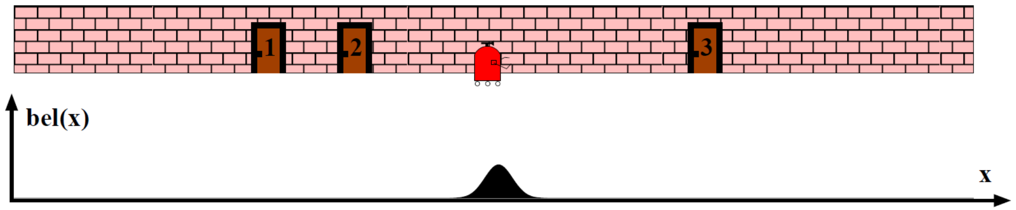
$$\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$$

$$\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$$

Return $\mu_t$, $\Sigma_t$



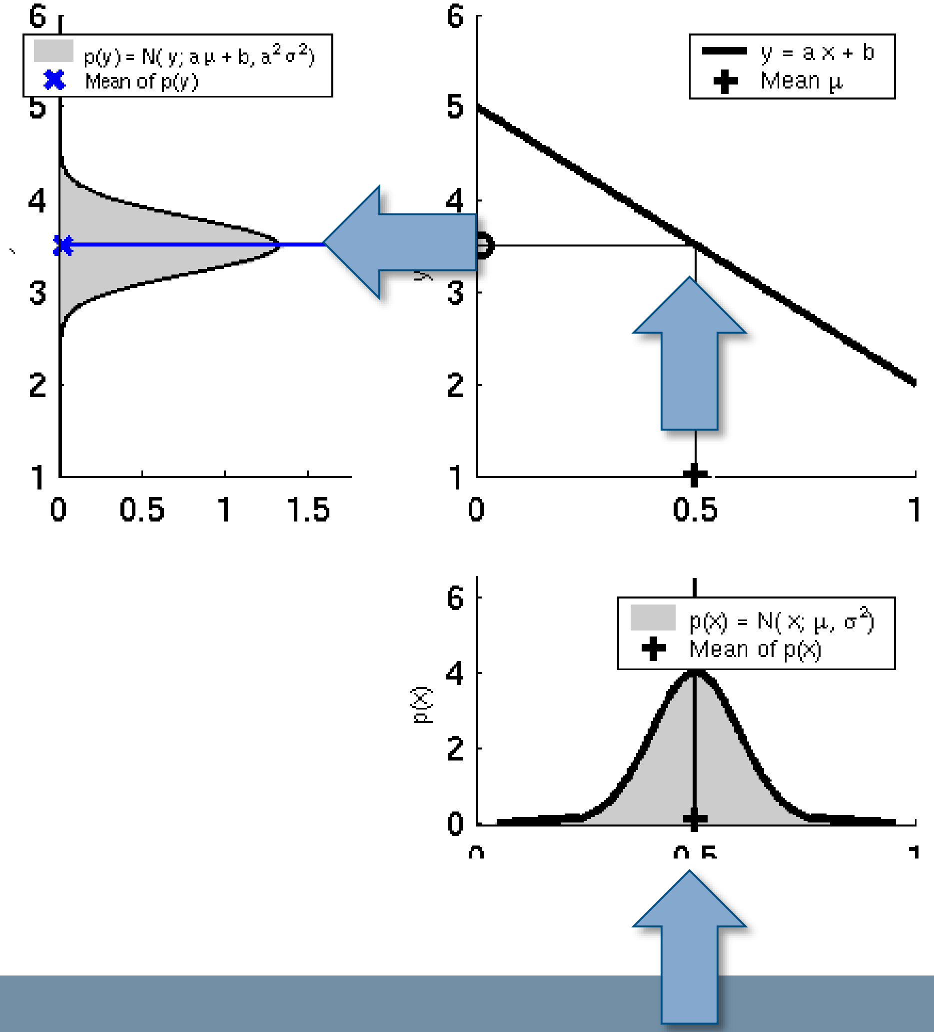

- Polynomial in measurement dimensionality k and state dimensionality n: O(k2.376 + n2)
- Optimal for linear Gaussian systems ☺
- Most robotics systems are nonlinear ☹
- It represents unimodal disributions ☹



# How to Deal with Non Linear Dynamic Systems?

Gaussian noise in linear systems

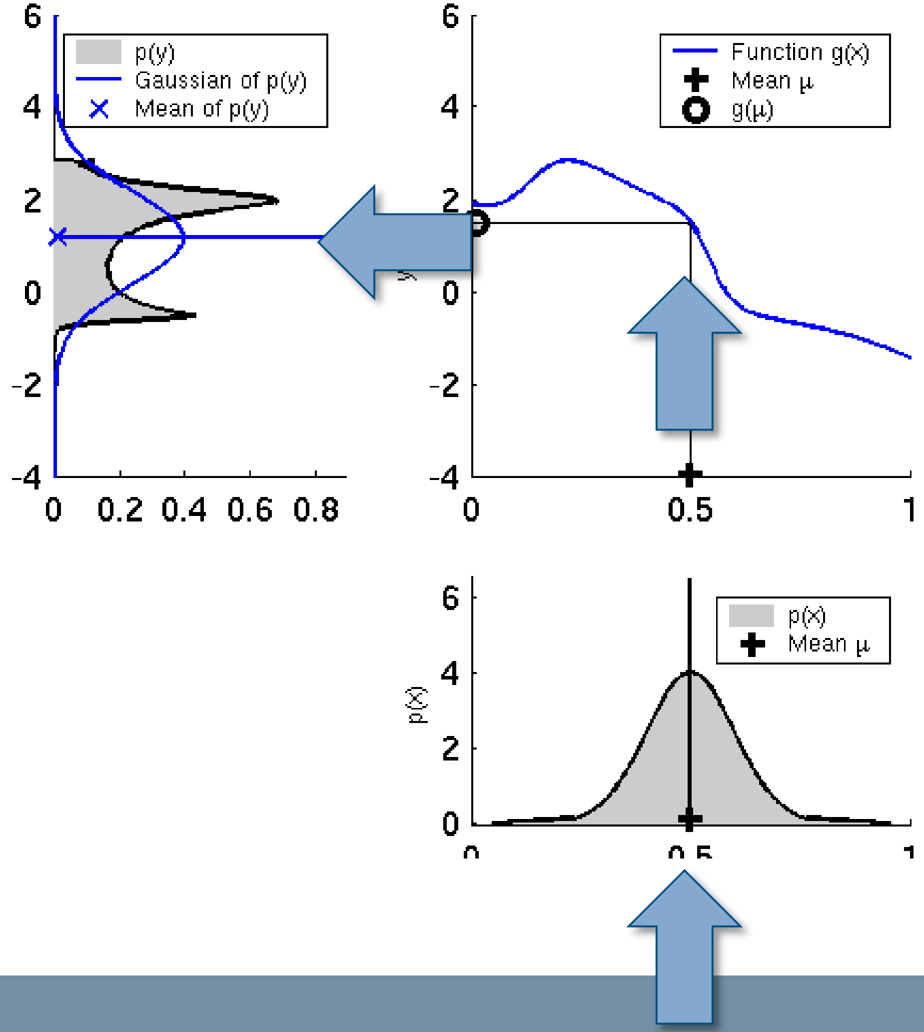
$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$$
$$z_t = C_t x_t + \delta_t$$



# How to Deal with Non Linear Dynamic Systems?

Gaussian noise in non-linear systems

$$x_t = g(u_t, x_{t-1})$$
$$z_t = h(x_t)$$



# Extended Kalman Filter (First order Taylor approximation)

Gaussian noise in non-linear systems

$$x_t = g(u_t, x_{t-1})$$
$$z_t = h(x_t)$$

Prediction:

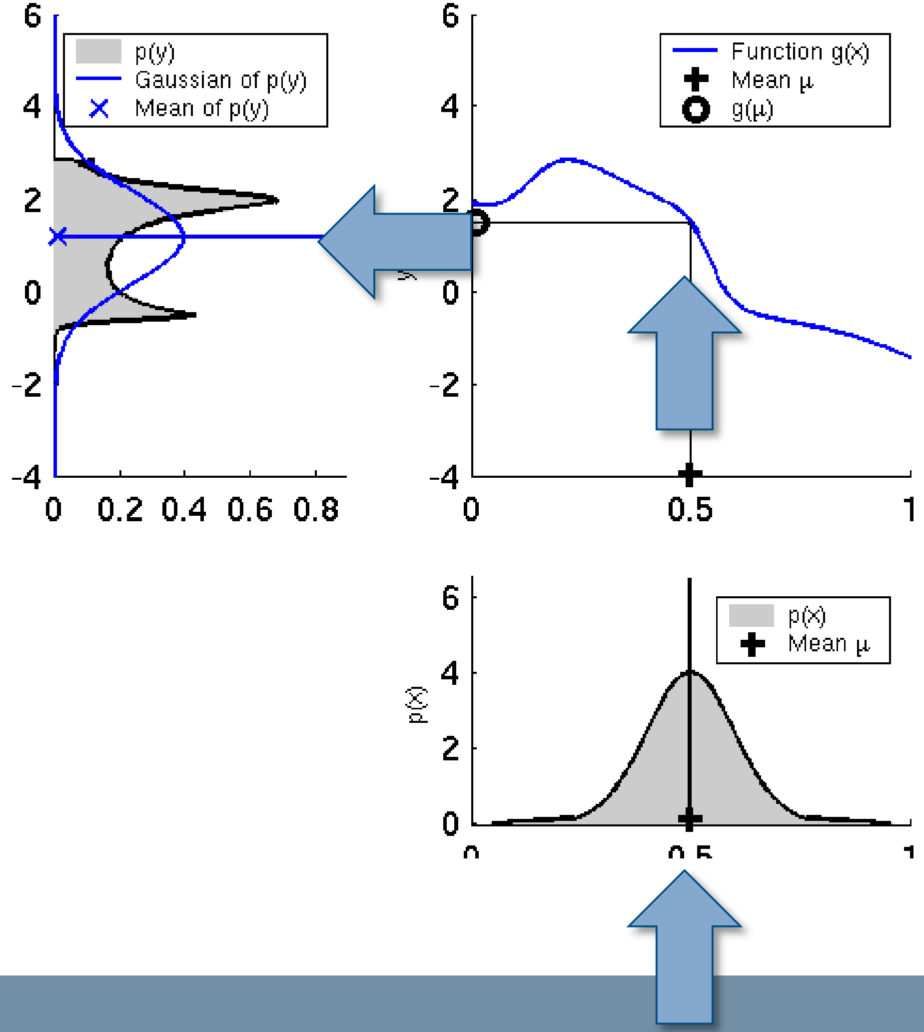
$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}} (x_{t-1} - \mu_{t-1})$$

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + G_t \, (x_{t-1} - \mu_{t-1})$$

Correction

$$h(x_t) \approx h(\bar{\mu}_t) + \frac{\partial h(\bar{\mu}_t)}{\partial x_t} (x_t - \bar{\mu}_t)$$

$$h(x_t) \approx h(\bar{\mu}_t) + H_t \, (x_t - \bar{\mu}_t)$$



# Extended Kalman Filter (First order Taylor approximation)

Gaussian noise in non-linear systems

$$x_t = g(u_t, x_{t-1})$$
$$z_t = h(x_t)$$

Prediction:

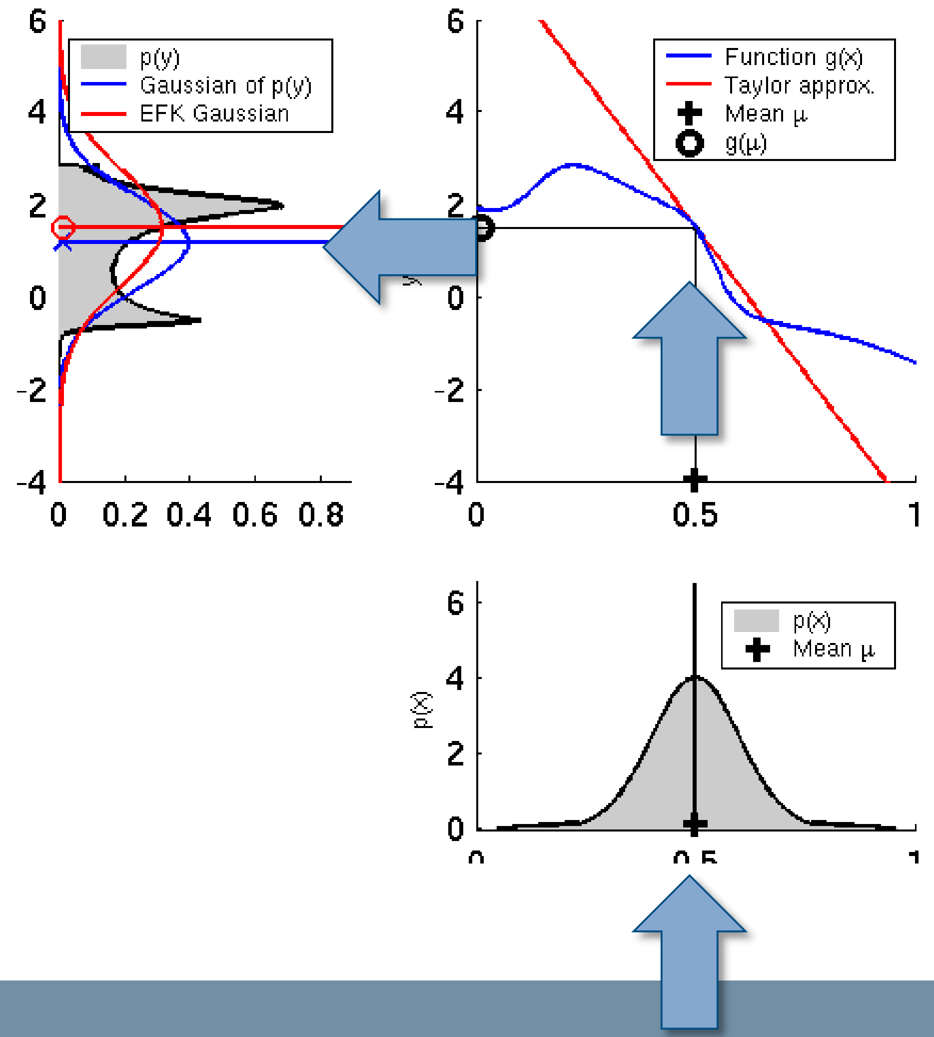
$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}} (x_{t-1} - \mu_{t-1})$$

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + G_t\,(x_{t-1} - \mu_{t-1})$$

Correction

$$h(x_t) \approx h(\bar{\mu}_t) + \frac{\partial h(\bar{\mu}_t)}{\partial x_t} (x_t - \bar{\mu}_t)$$

$$h(x_t) \approx h(\bar{\mu}_t) + H_t\,(x_t - \bar{\mu}_t)$$



# Extended Kalman Filter (First order Taylor approximation)

Gaussian noise in non-linear systems

$$x_t = g(u_t, x_{t-1})$$
$$z_t = h(x_t)$$

Prediction:

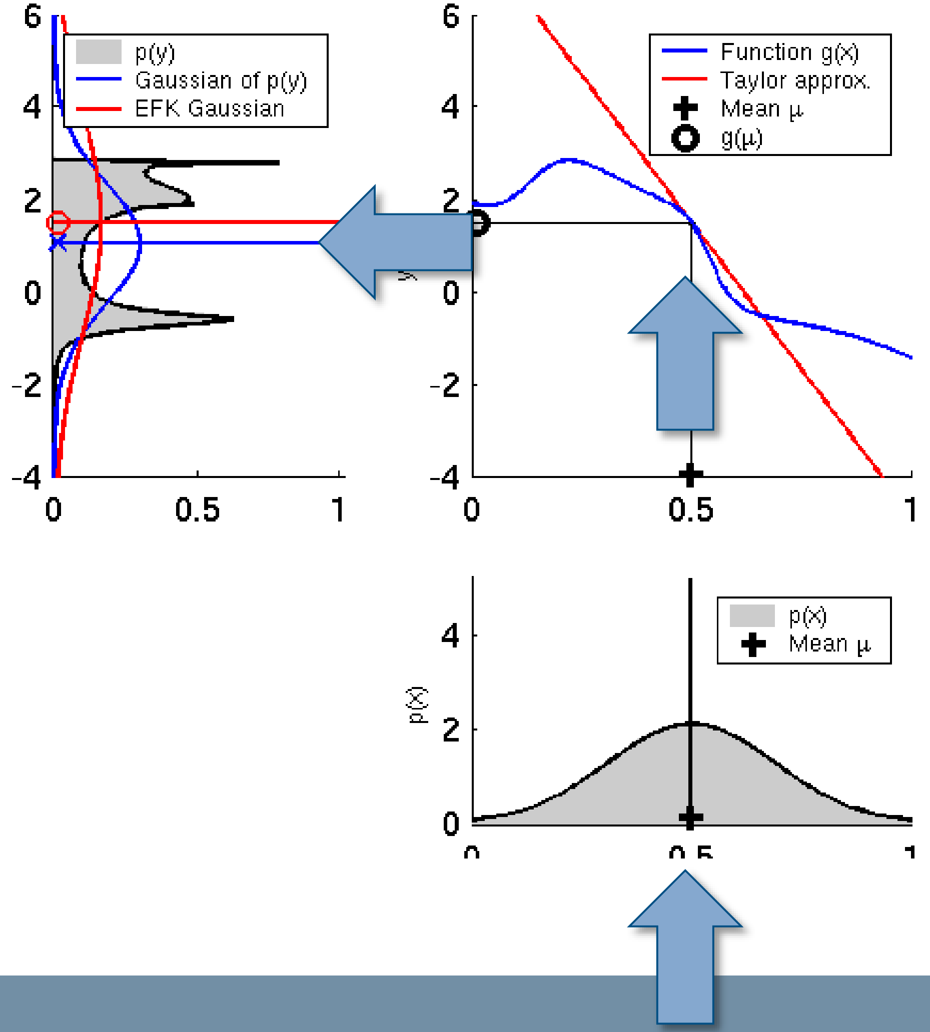
$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}} (x_{t-1} - \mu_{t-1})$$

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + G_t \, (x_{t-1} - \mu_{t-1})$$

Correction

$$h(x_t) \approx h(\bar{\mu}_t) + \frac{\partial h(\bar{\mu}_t)}{\partial x_t} (x_t - \bar{\mu}_t)$$

$$h(x_t) \approx h(\bar{\mu}_t) + H_t \, (x_t - \bar{\mu}_t)$$



# Extended Kalman Filter (First order Taylor approximation)

Gaussian noise in non-linear systems

$$x_t = g(u_t, x_{t-1})$$
$$z_t = h(x_t)$$

Prediction:

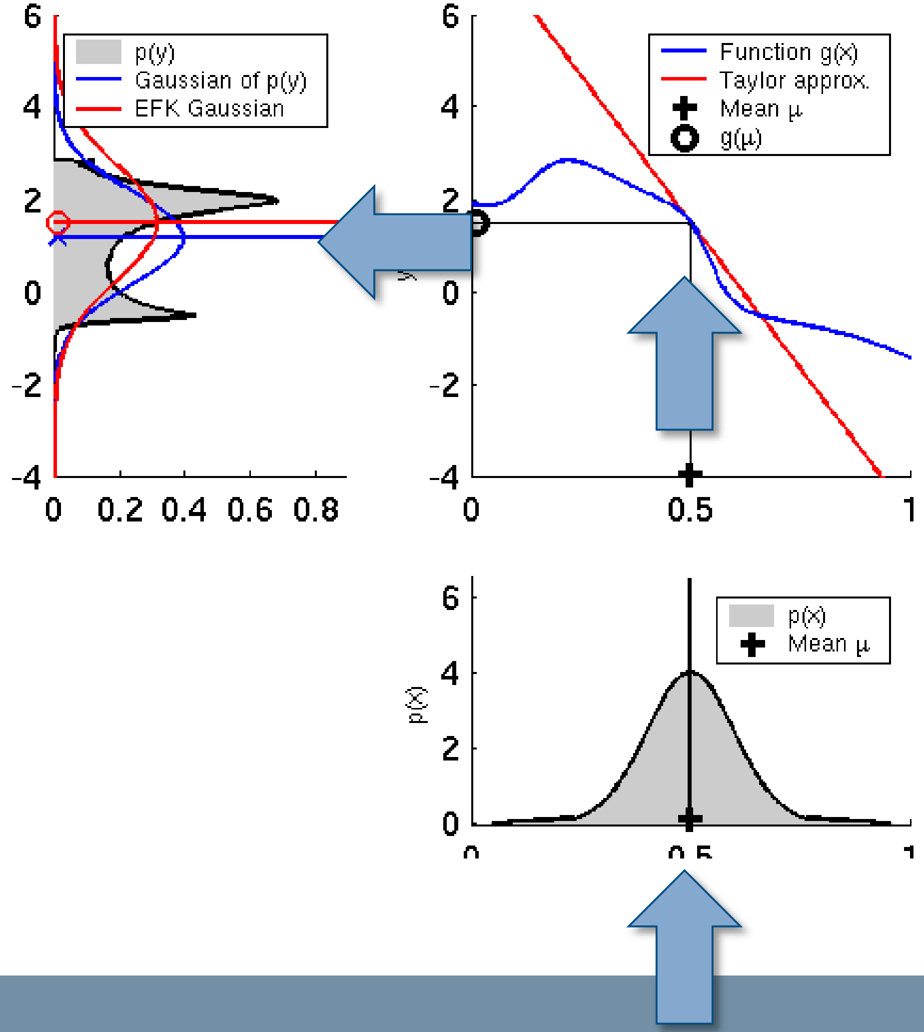
$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}} (x_{t-1} - \mu_{t-1})$$

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + G_t \, (x_{t-1} - \mu_{t-1})$$

Correction

$$h(x_t) \approx h(\bar{\mu}_t) + \frac{\partial h(\bar{\mu}_t)}{\partial x_t} (x_t - \bar{\mu}_t)$$

$$h(x_t) \approx h(\bar{\mu}_t) + H_t \, (x_t - \bar{\mu}_t)$$



# Extended Kalman Filter (First order Taylor approximation)

Gaussian noise in non-linear systems

$$x_t = g(u_t, x_{t-1})$$
$$z_t = h(x_t)$$

Prediction:

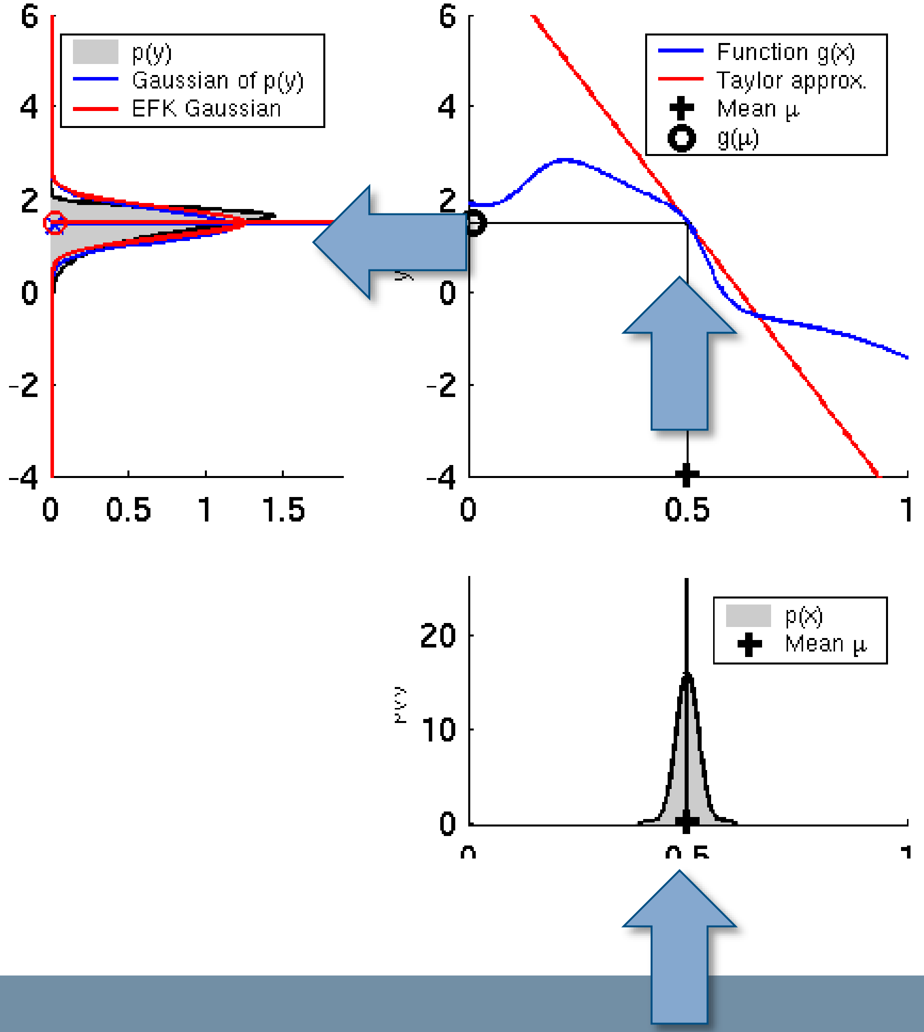
$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}} (x_{t-1} - \mu_{t-1})$$

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + G_t \, (x_{t-1} - \mu_{t-1})$$

Correction

$$h(x_t) \approx h(\bar{\mu}_t) + \frac{\partial h(\bar{\mu}_t)}{\partial x_t} (x_t - \bar{\mu}_t)$$

$$h(x_t) \approx h(\bar{\mu}_t) + H_t \, (x_t - \bar{\mu}_t)$$



# EKF Algorithm

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

$$G_t = \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}} \quad H_t = \frac{\partial h(\bar{\mu}_t)}{\partial x_t}$$

Prediction:

$$\bar{\mu}_t = g(u_t, \mu_{t-1}) \quad \longleftarrow \quad \bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$$

$$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t \quad \longleftarrow \quad \bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$$

Correction:

$$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1} \quad \longleftarrow \quad K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$

$$\mu_t = \bar{\mu}_t + K_t (z_t - h(\bar{\mu}_t)) \quad \longleftarrow \quad \mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$$

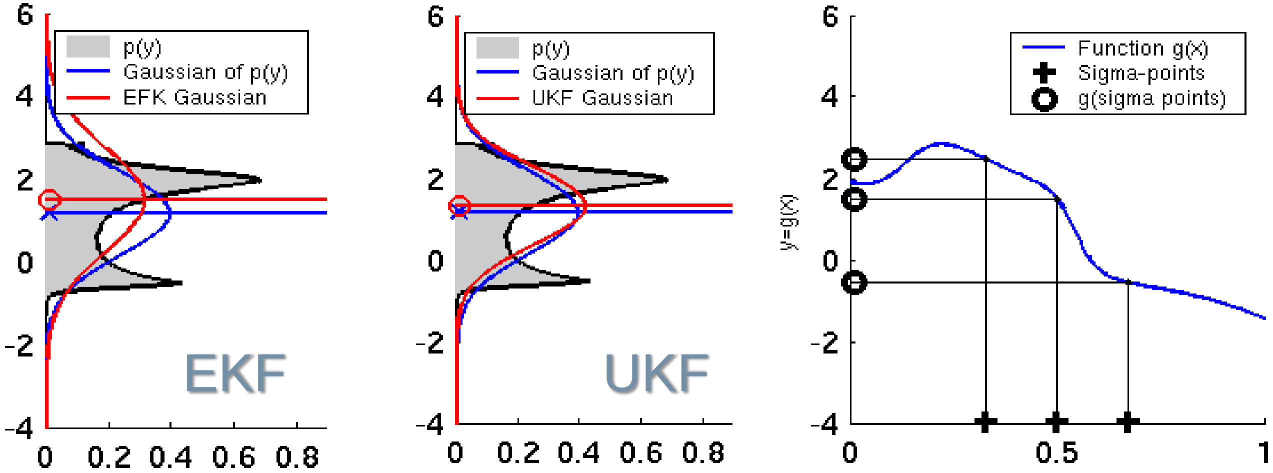
$$\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t \quad \longleftarrow \quad \Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$$

Return $\mu_t$, $\Sigma_t$

# EKF and Friends

Extended Kalman Filter:

- Polynomial in measurement k and state n dimensionality: $O(k^{2.376} + n^2)$
- Not optimal and can diverge if nonlinearities are large!
- Works surprisingly well even when all assumptions are violated!
- There are possible alternative like the Unscented Kalman Transform ...



## Bayes Filter Algorithm

$$Bel(x_t) = \eta\ P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1})\ Bel(x_{t-1})\ dx_{t-1}$$

How to represent such belief?

Algorithm Bayes_filter( *Bel(x), d* ):

$\eta$=0

If *d* is a perceptual data item *z* then

For all *x* do

$$Bel'(x) = P(z \mid x) Bel(x)$$

$$\eta = \eta + Bel'(x)$$

For all *x* do

$$Bel'(x) = \eta^{-1} Bel'(x)$$

Else if *d* is an action data item *u* then

For all *x* do

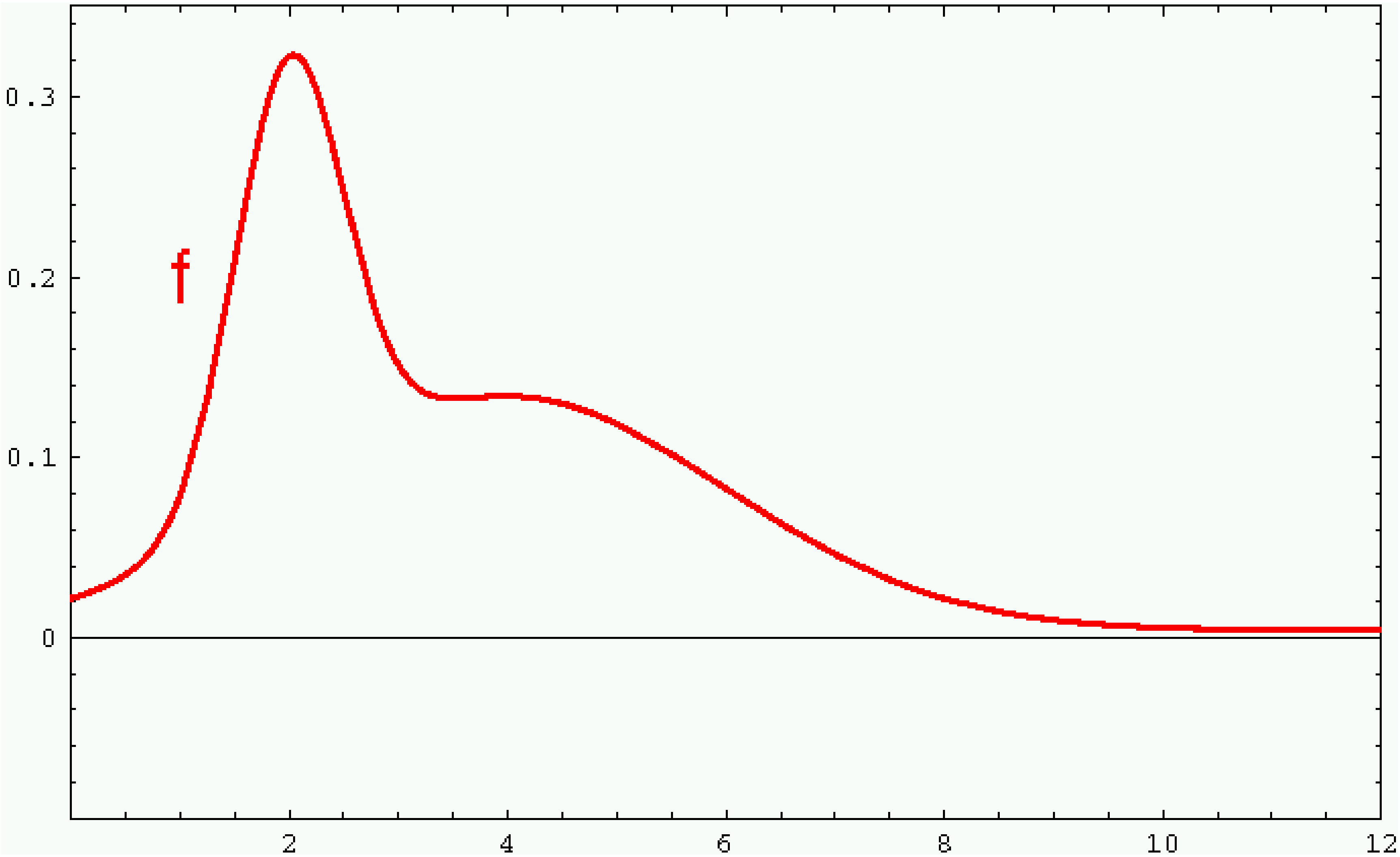
$$Bel'(x) = \int P(x \mid u, x')\ Bel(x')\ dx'$$

Return *Bel'(x)*

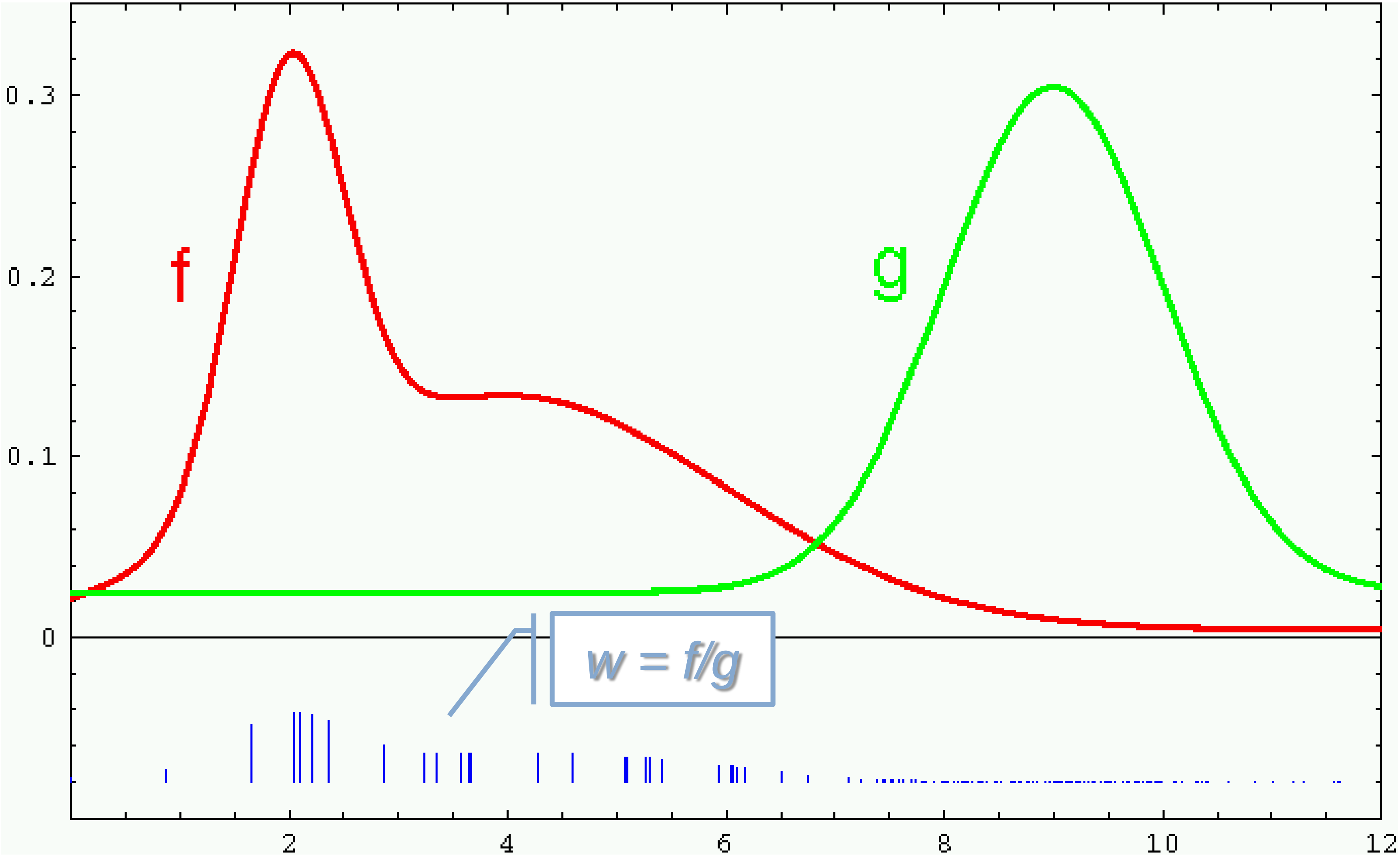
Based on such representation:

- Discrete filters
- Kalman filters
- Sigma-point filters
- Particle filters
- ...

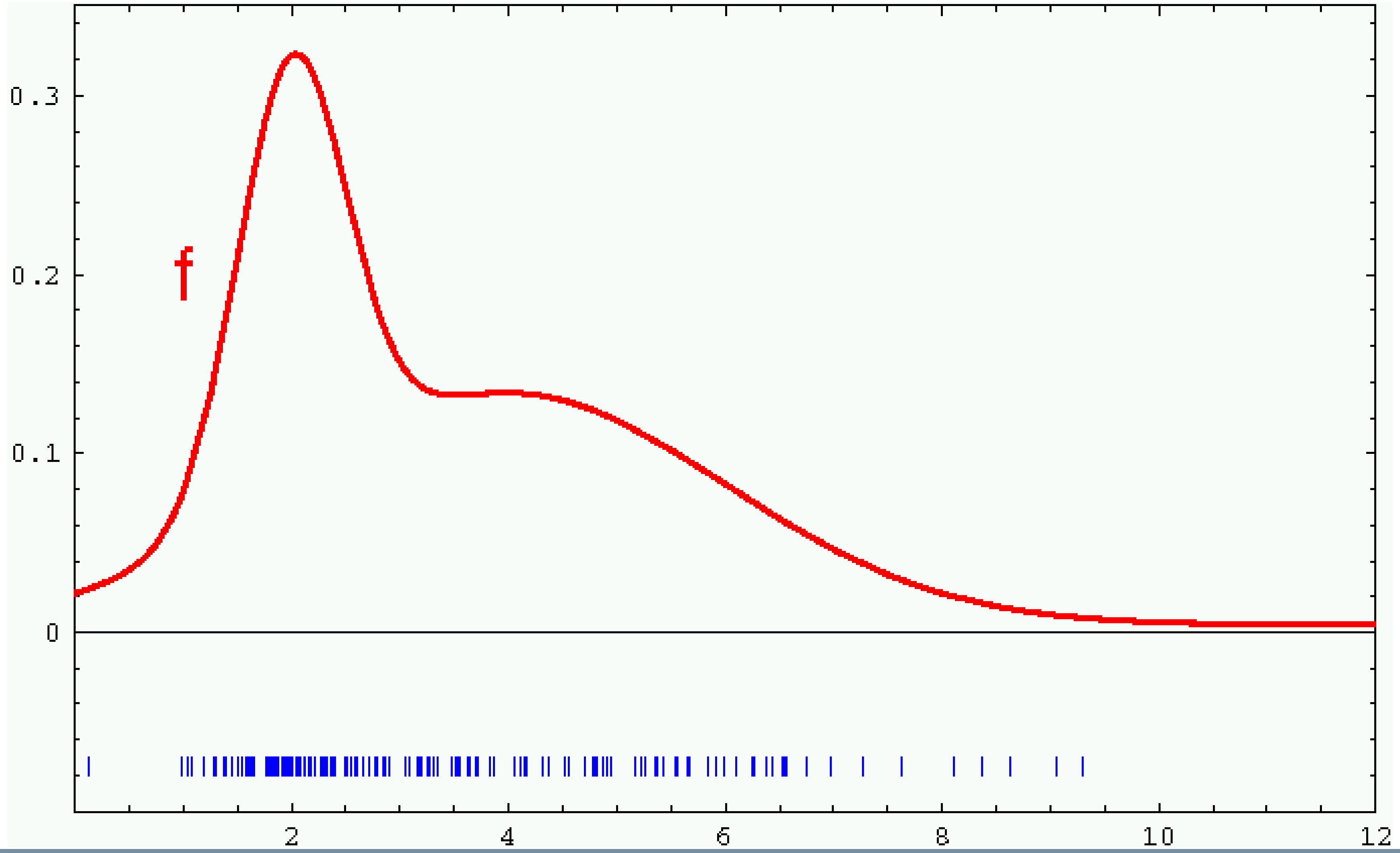
# Importance Resampling



# Importance Resampling



# Importance Resampling (with smoothing)



## Particle Filter Algorithm

$$\boxed{Bel(x_t) = \underbrace{\eta\; p(z_t \mid x_t) \overbrace{\int p(x_t \mid x_{t-1}, u_{t-1})\; Bel(x_{t-1})\; dx_{t-1}}^{g}}_{f}}$$

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 $x^i_t$:

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

# Particle Filter Algorithm

Algorithm **particle_filter**($S_{t-1}$, $u_{t-1}$, $z_t$):

$$S_t = \varnothing, \quad \eta = 0$$

**For** $i = 1 \ldots n$ ***Generate new samples***

Sample index *j(i)* from the discrete distribution given by $w_{t-1}$

Sample $x_t^i$ from $p(x_t \mid x_{t-1}, u_{t-1})$ using $x_{t-1}^{j(i)}$ and $u_{t-1}$

$w_t^i = p(z_t \mid x_t^i)$ ***Compute importance weight***

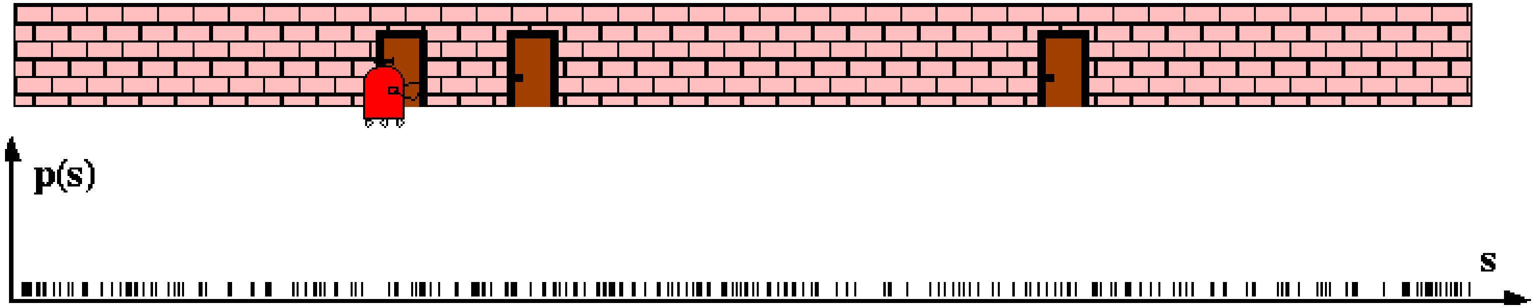
$\eta = \eta + w_t^i$ ***Update normalization factor***

$S_t = S_t \cup \{< x_t^i, w_t^i >\}$ ***Insert***

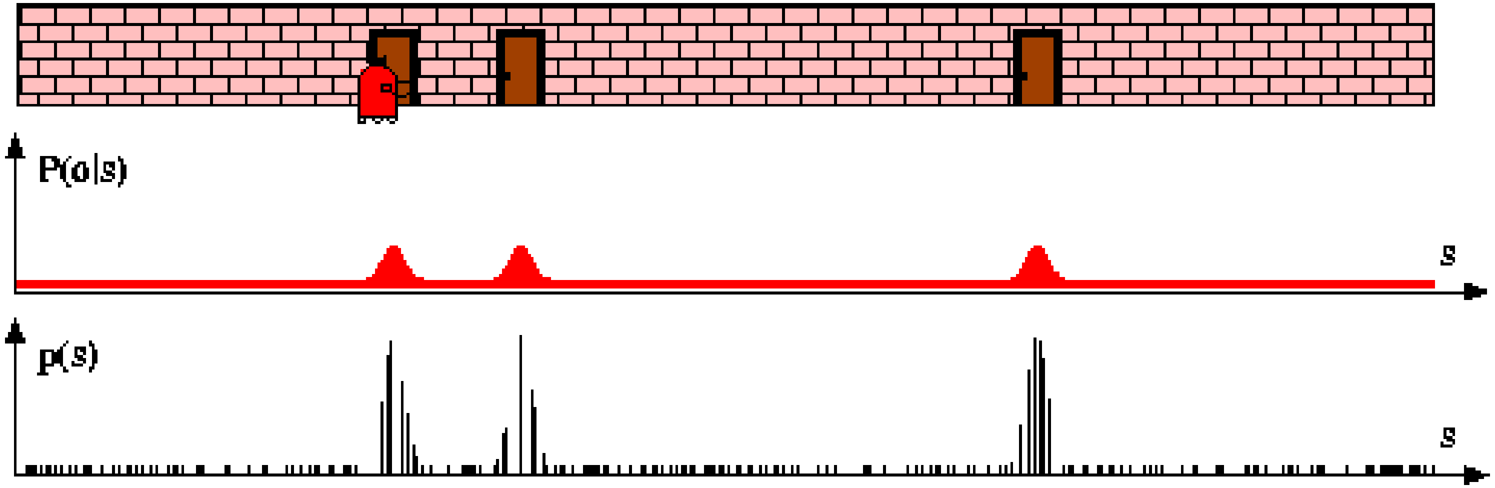
**For** $i = 1 \ldots n$

$w_t^i = w_t^i / \eta$ ***Normalize weights***

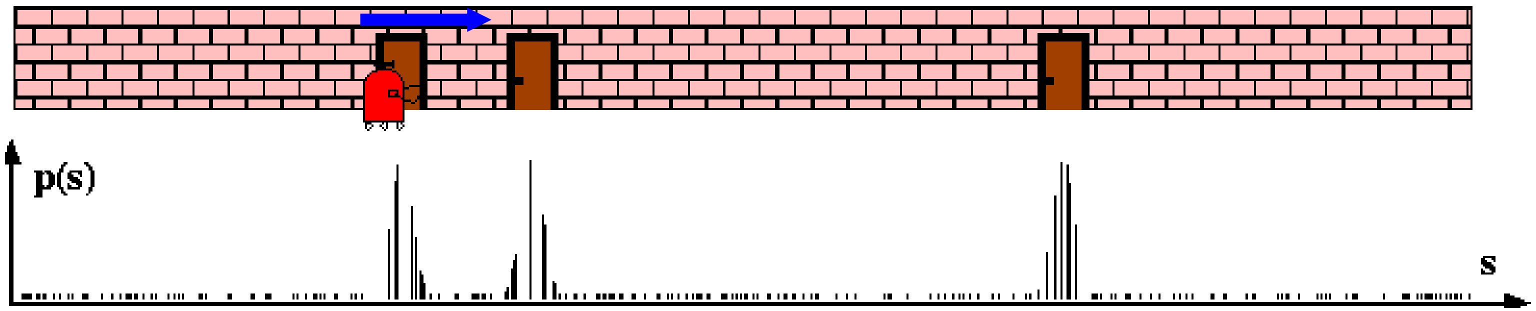
# Sensor Information: Importance Sampling



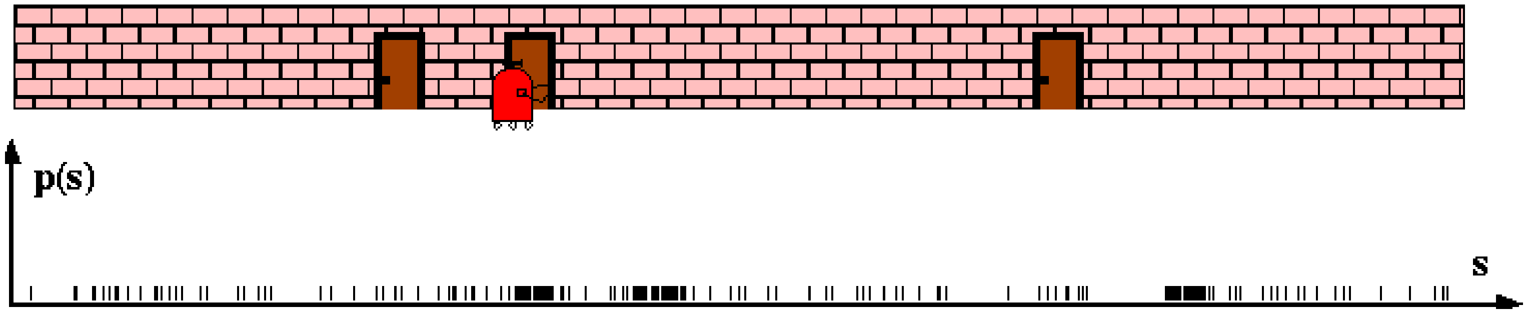

$$Bel(x) \leftarrow \alpha\ p(z\,|\,x)\,Bel^{-}(x) \qquad w \leftarrow \frac{\alpha\ p(z\,|\,x)\,Bel^{-}(x)}{Bel^{-}(x)} = \alpha\ p(z\,|\,x)$$



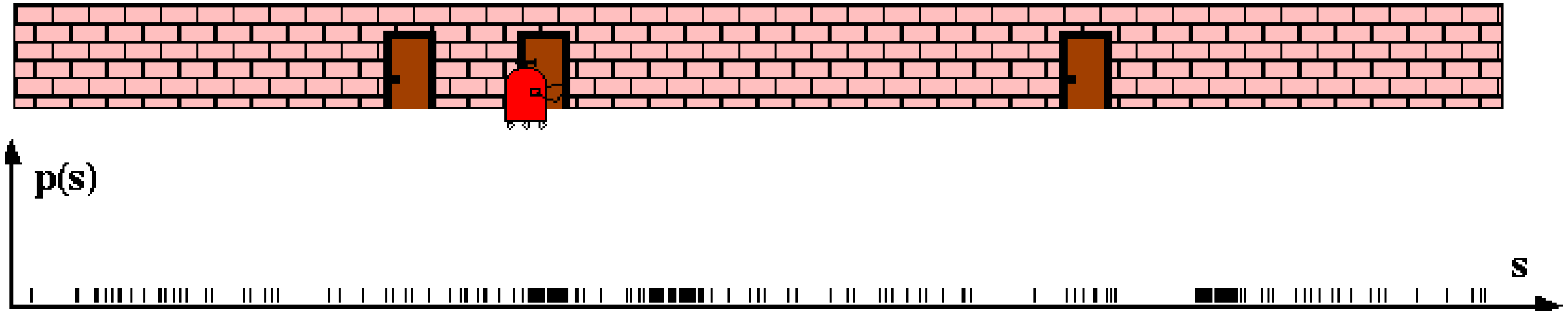
# Sensor Information: Importance Sampling



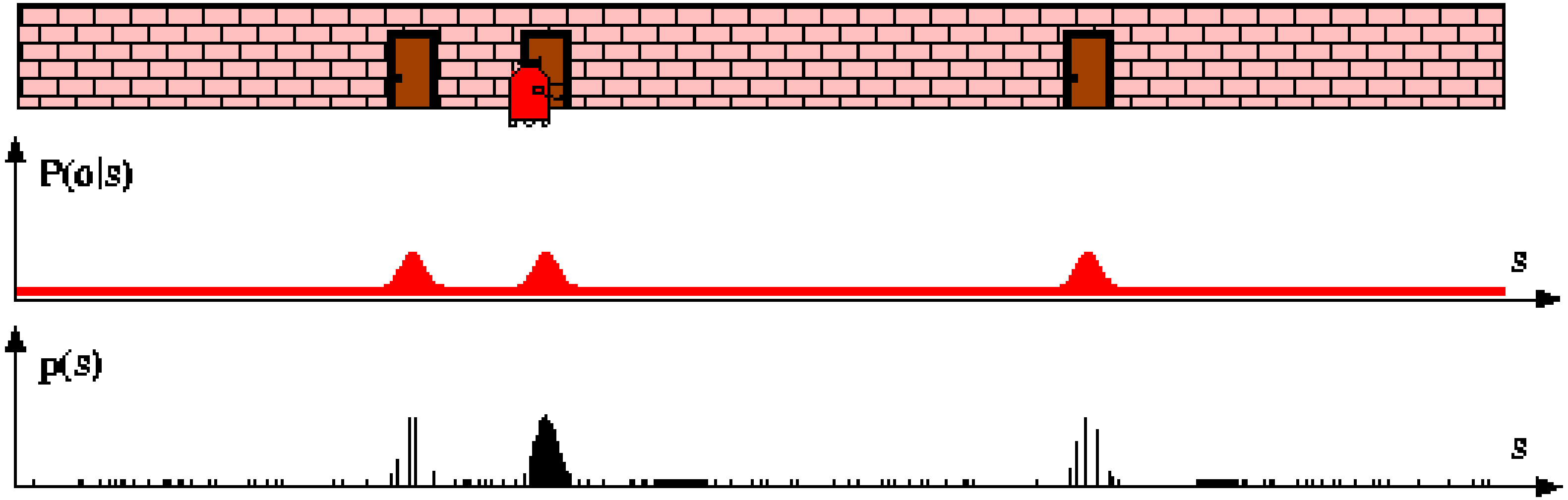

$$Bel^{-}(x) \leftarrow \int p(x \mid u, x')\, Bel(x')\, \mathrm{d}x'$$



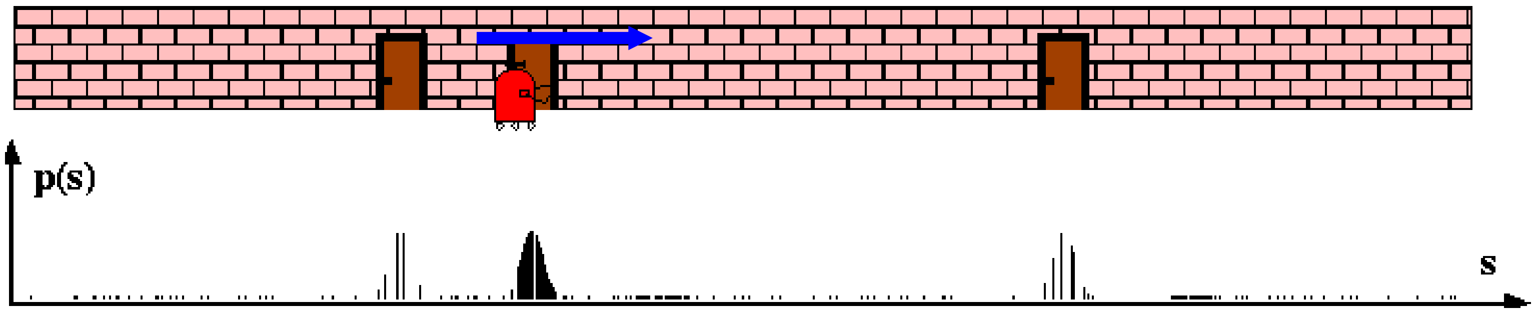
# Sensor Information: Importance Sampling



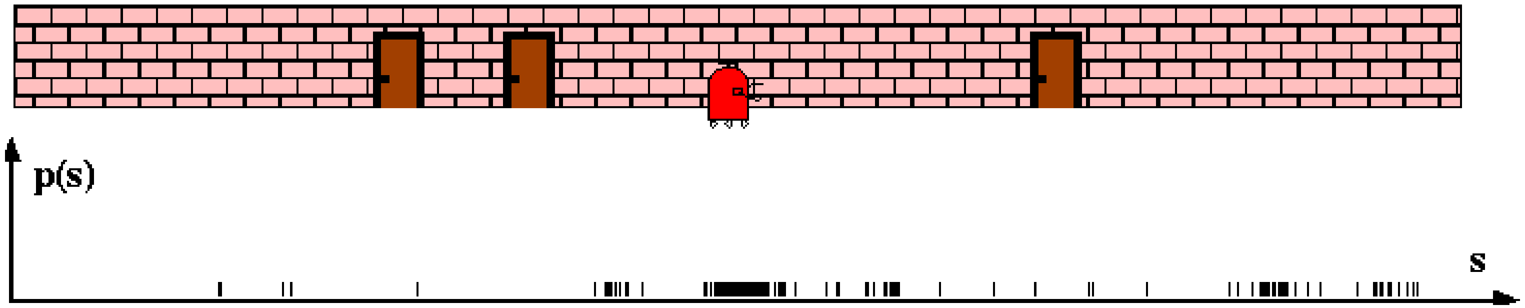

$$Bel(x) \leftarrow \alpha\ p(z\,|\,x)\,Bel^{-}(x) \qquad w \leftarrow \frac{\alpha\ p(z\,|\,x)\,Bel^{-}(x)}{Bel^{-}(x)} = \alpha\ p(z\,|\,x)$$



# Sensor Information: Importance Sampling

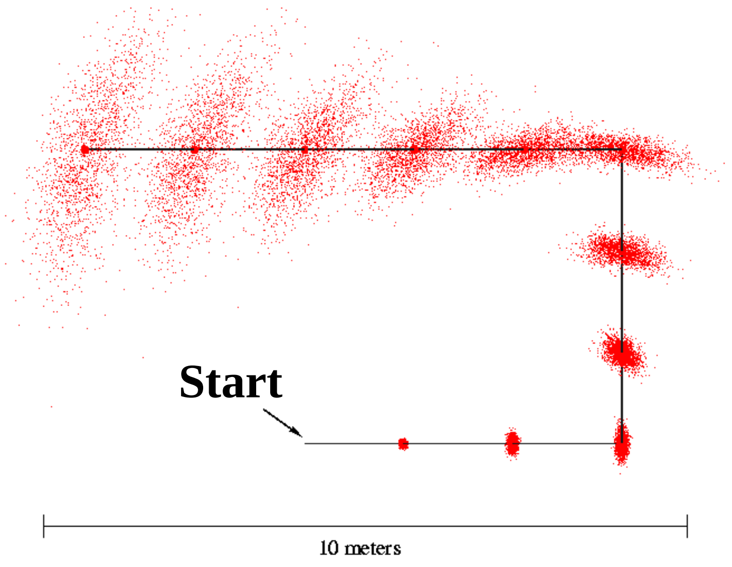



$$Bel^{-}(x) \leftarrow \int p(x \mid u, x')\, Bel(x')\, \mathrm{d}x'$$

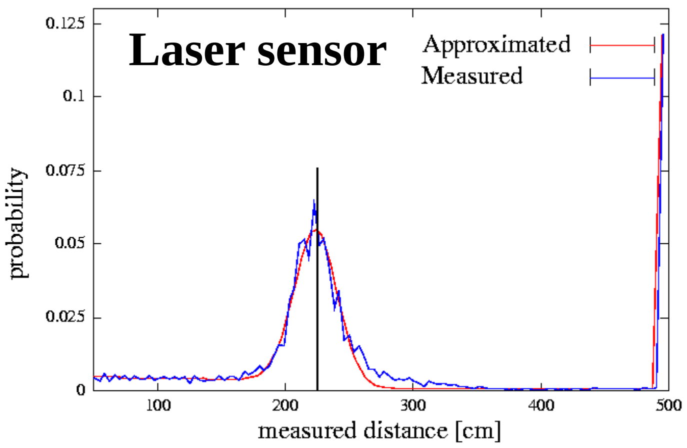


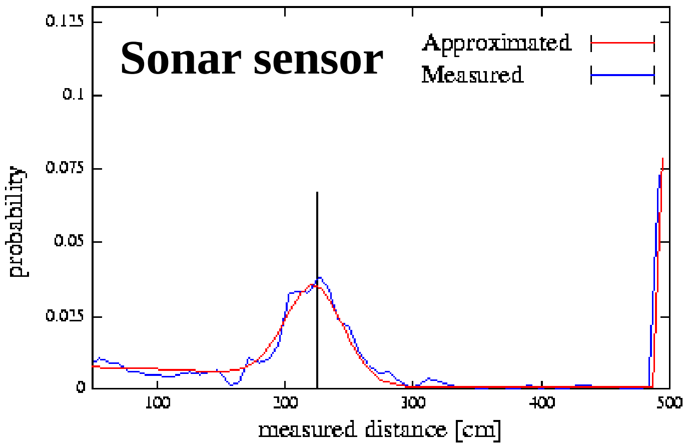
# Monte Carlo Localization with Laser

Stochastic motion model

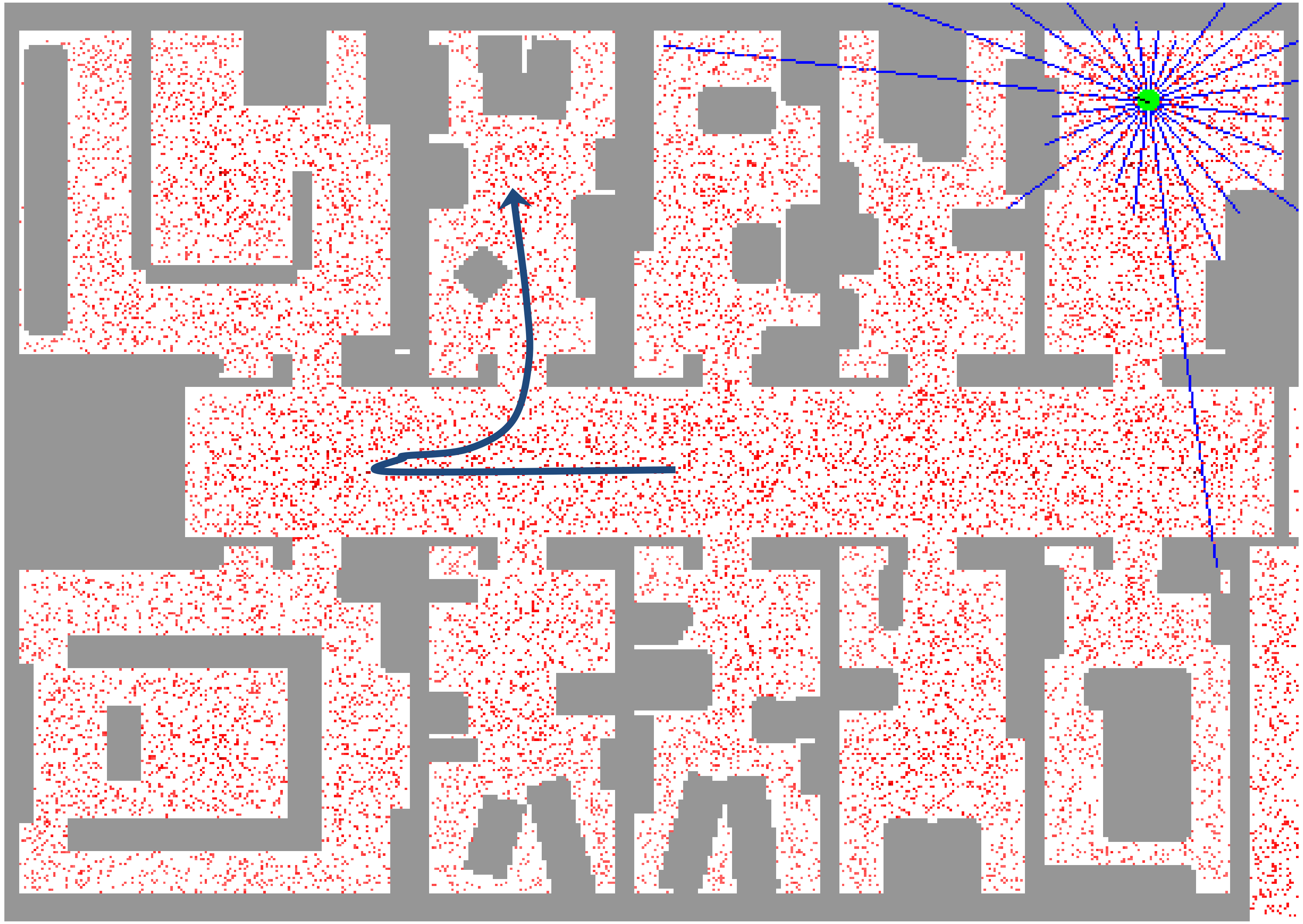


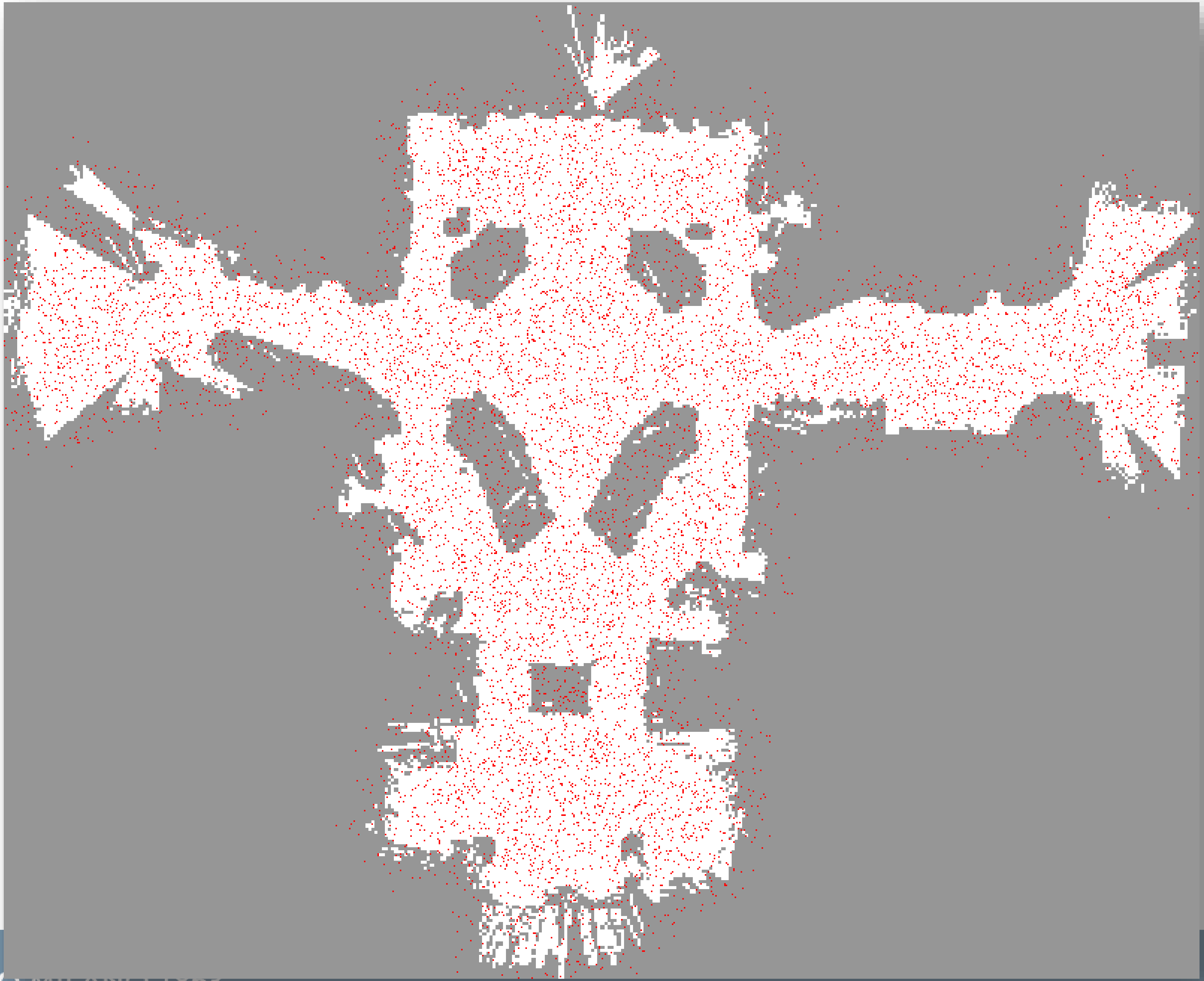

Proximity sensor model

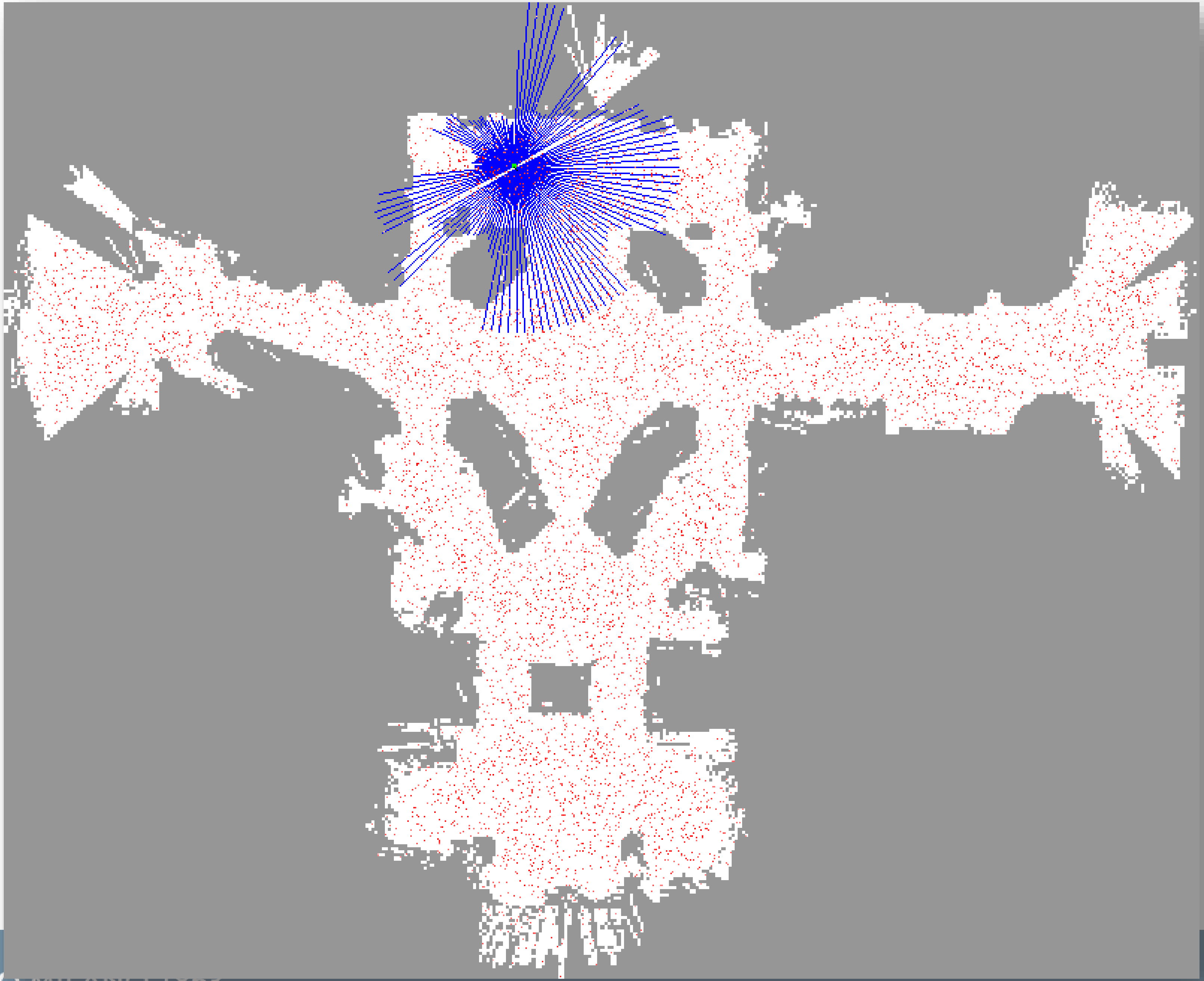


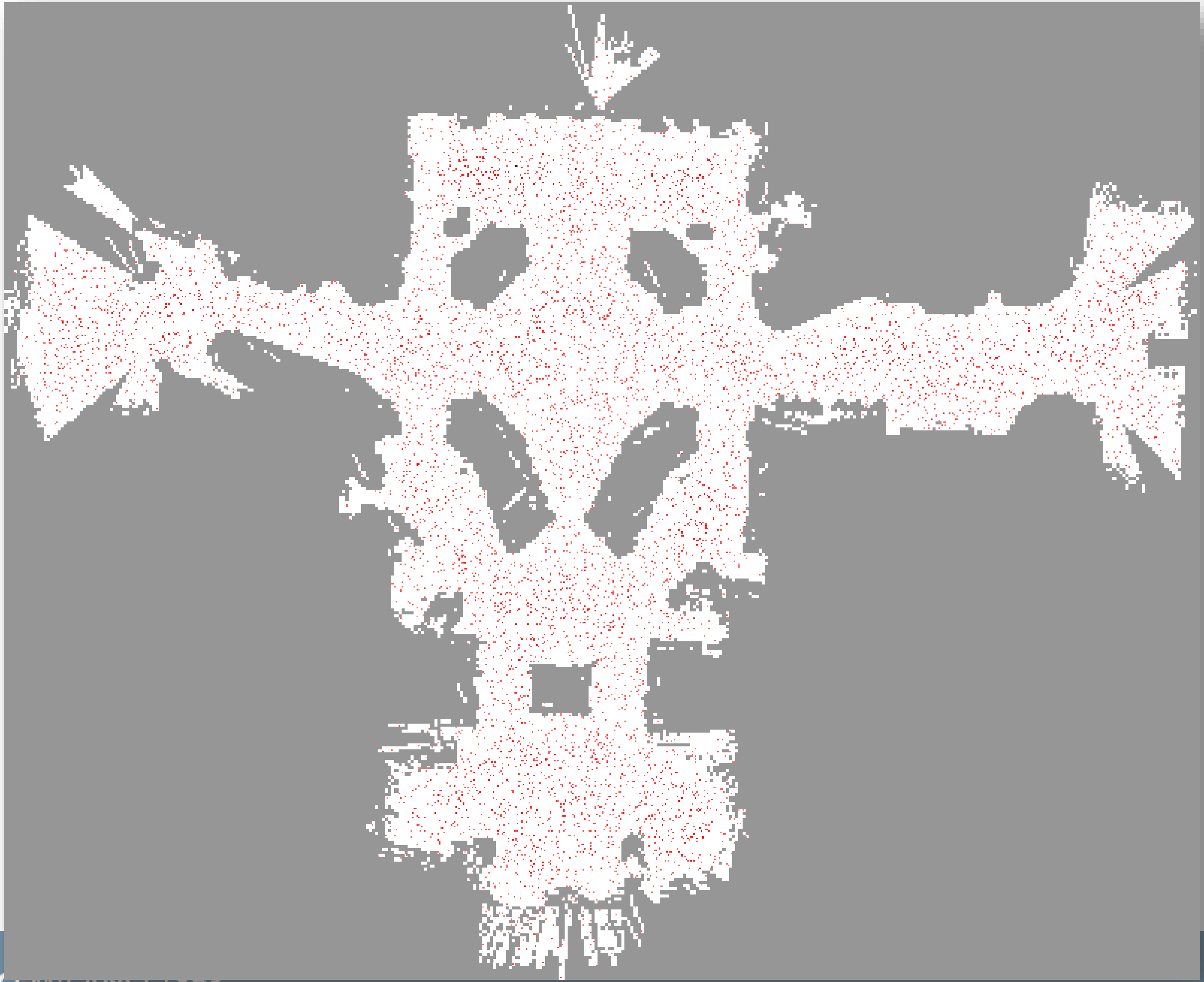



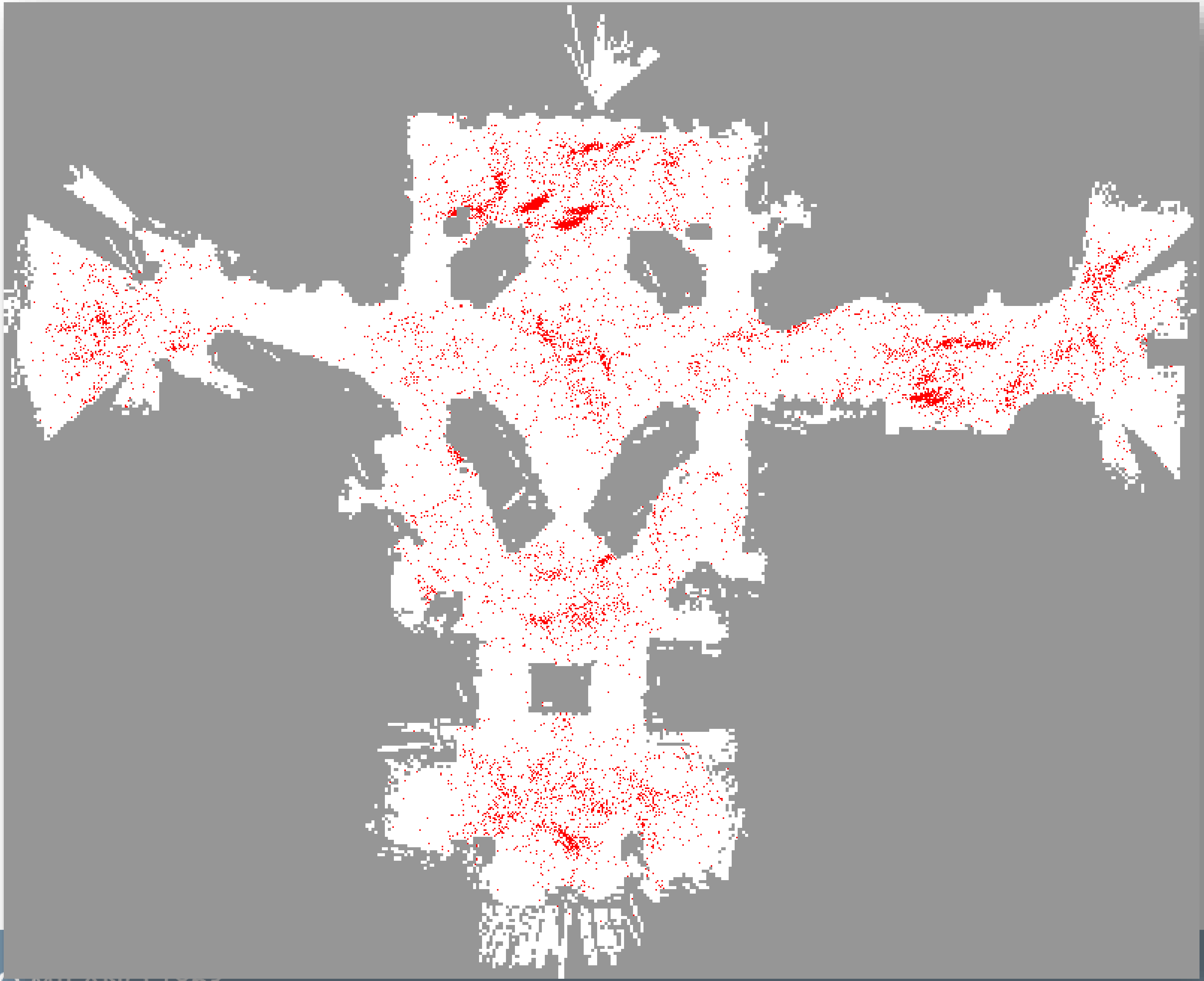
# Sample-based Localization (sonar)

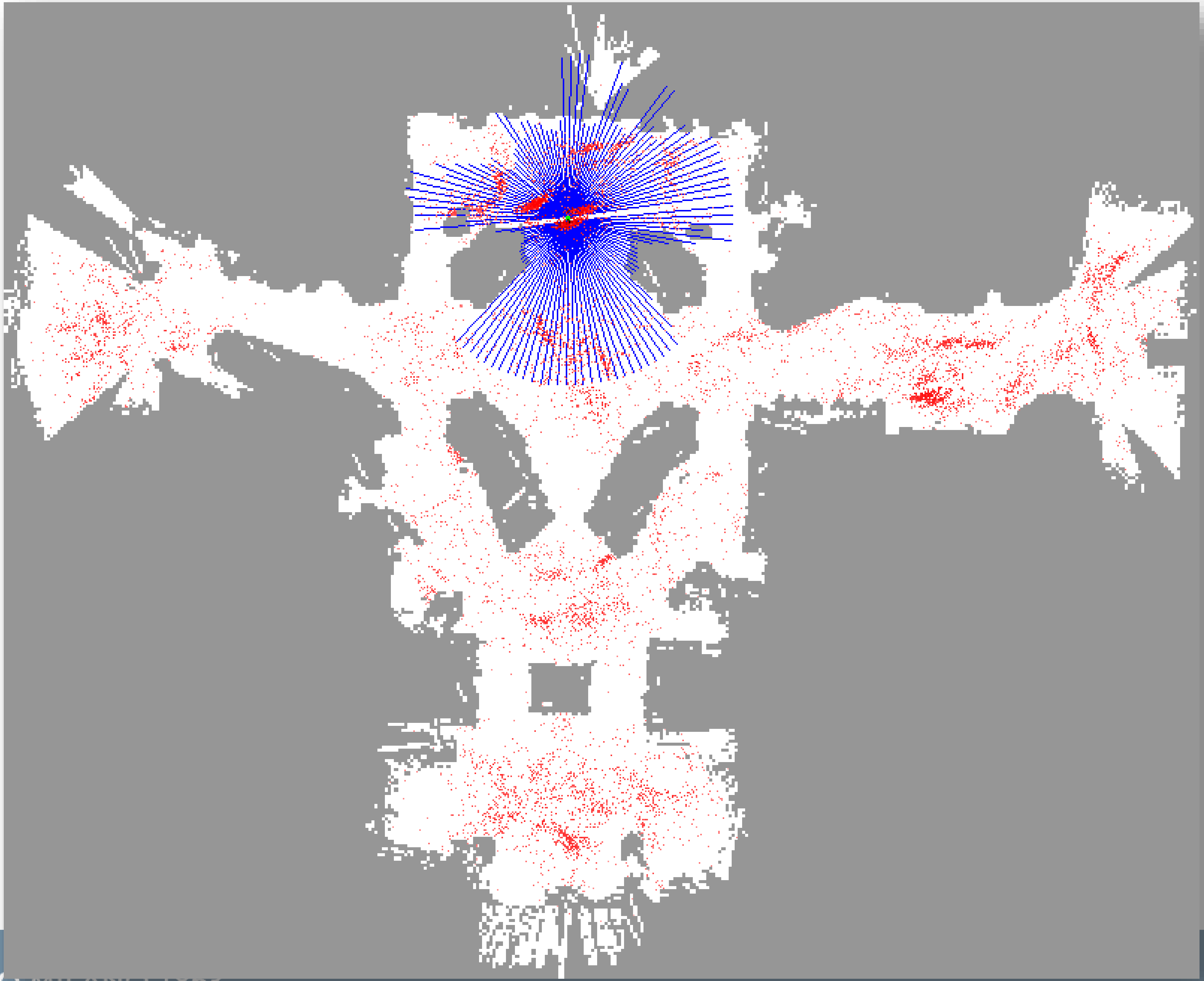


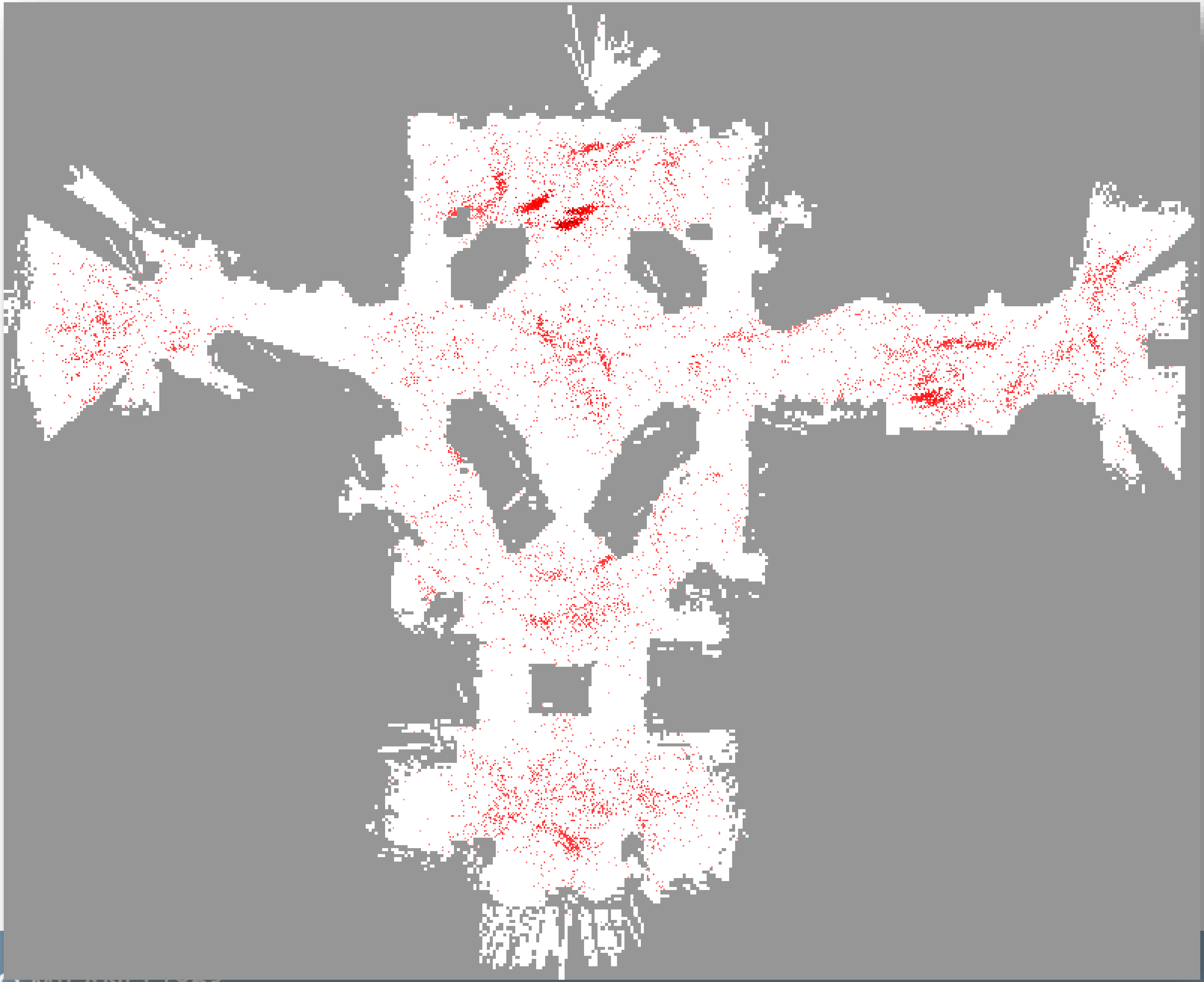


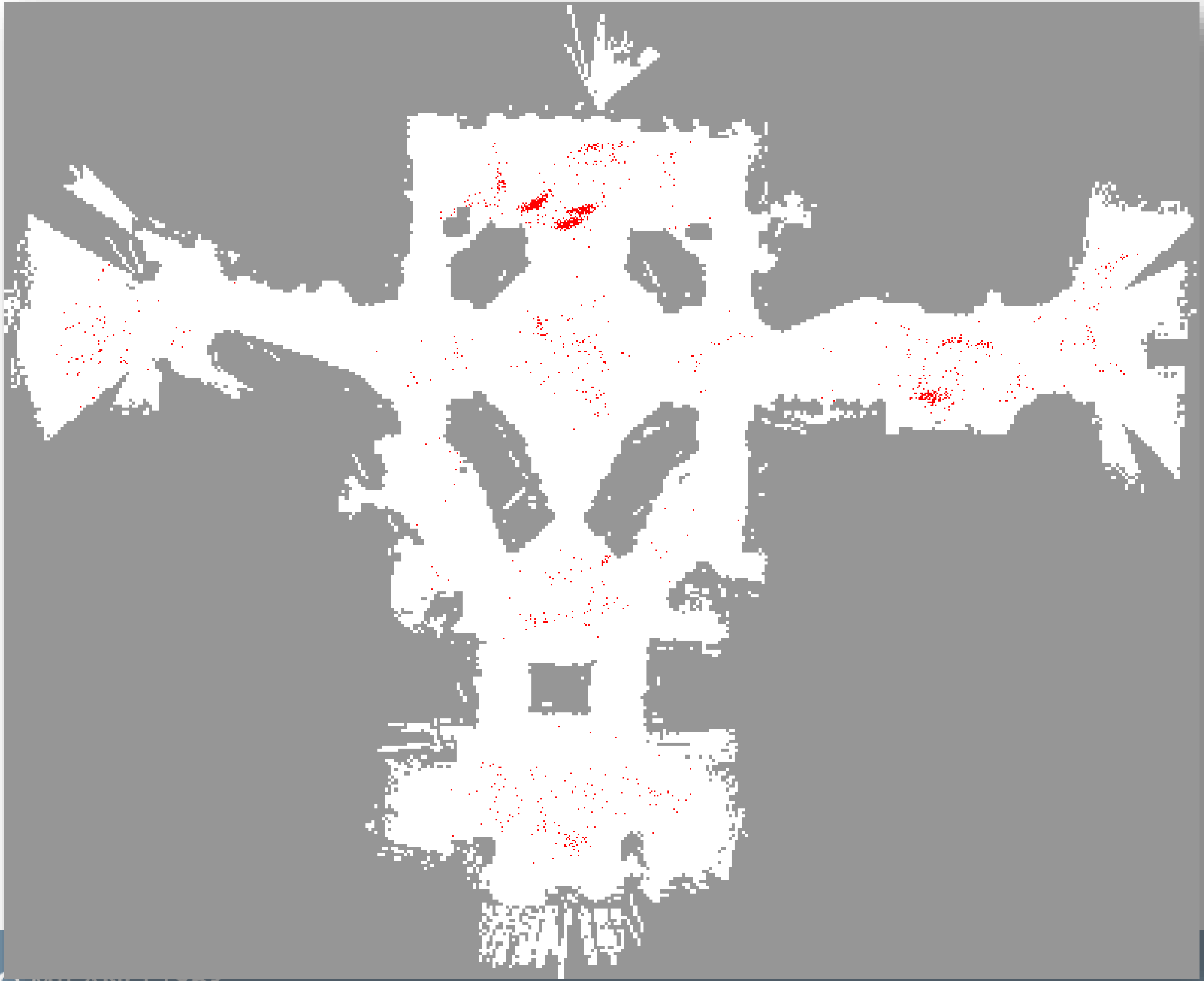


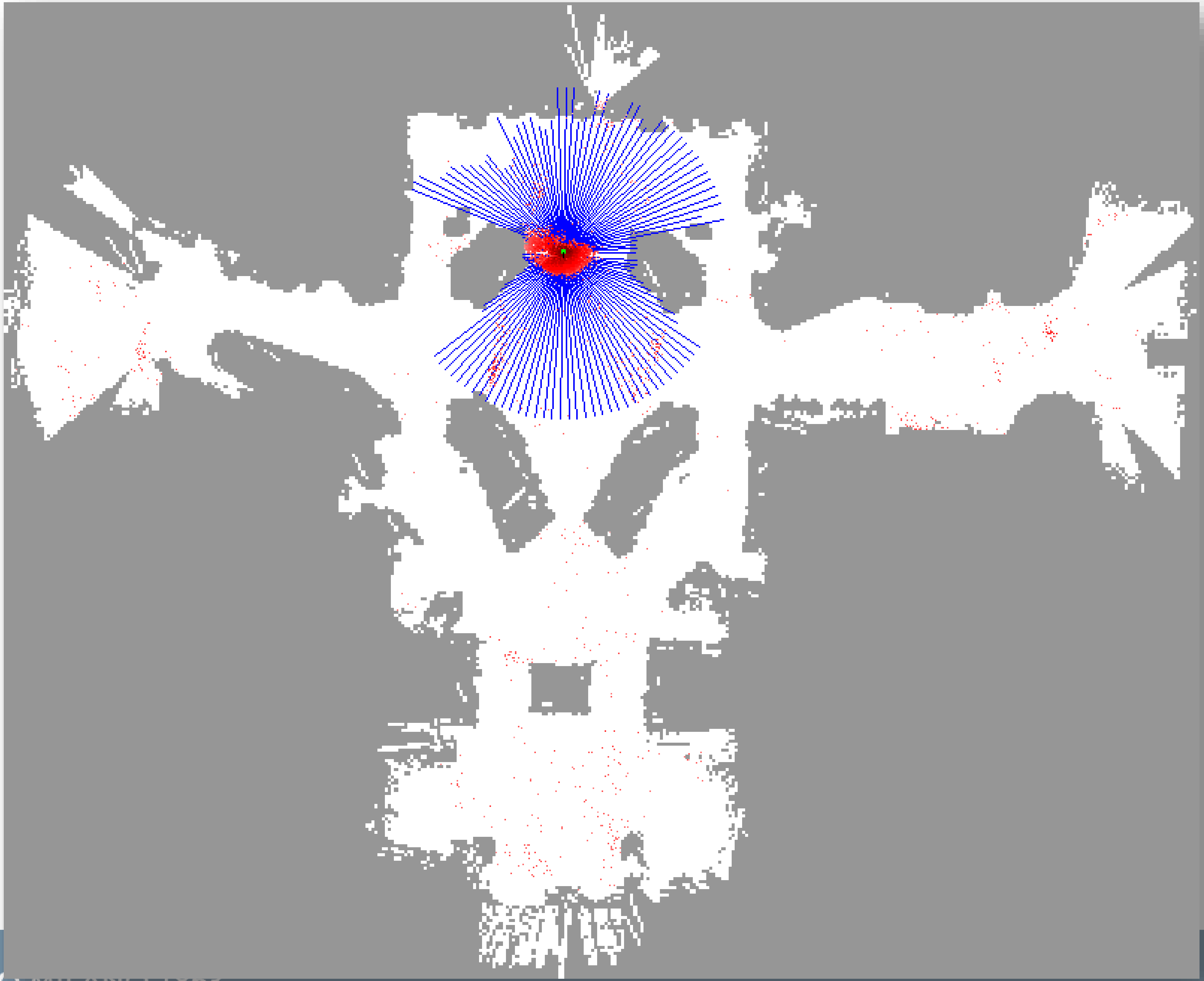


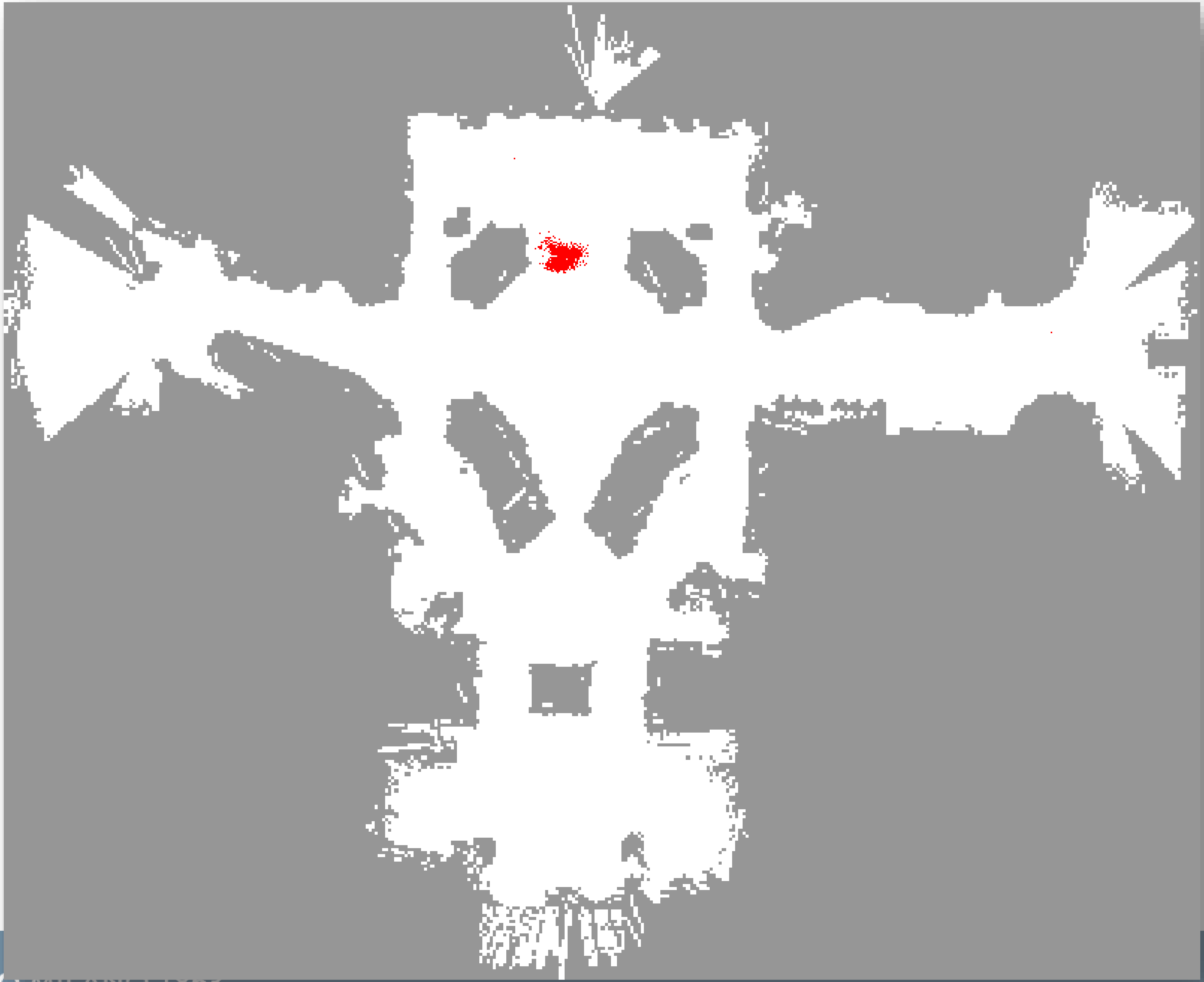


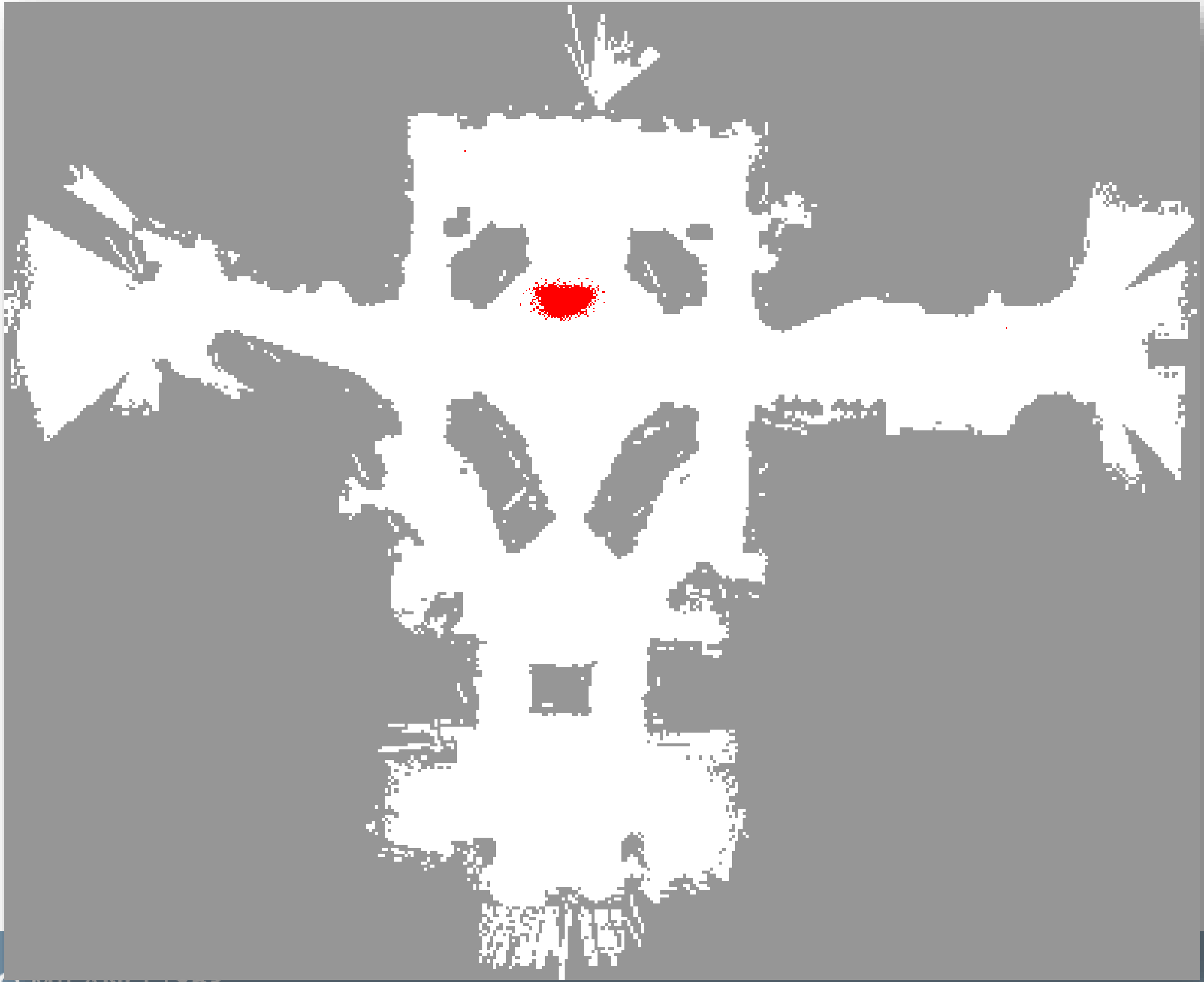


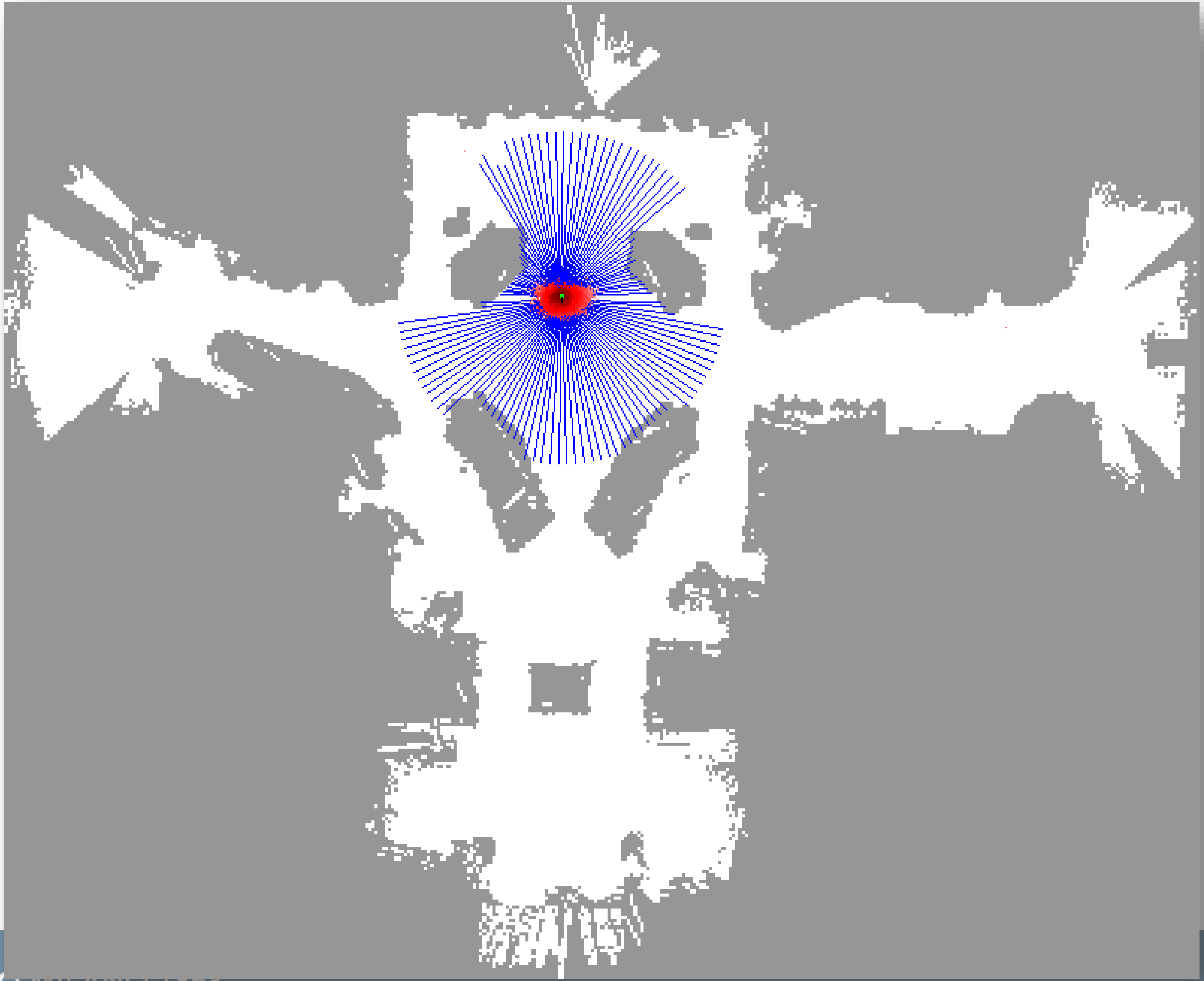


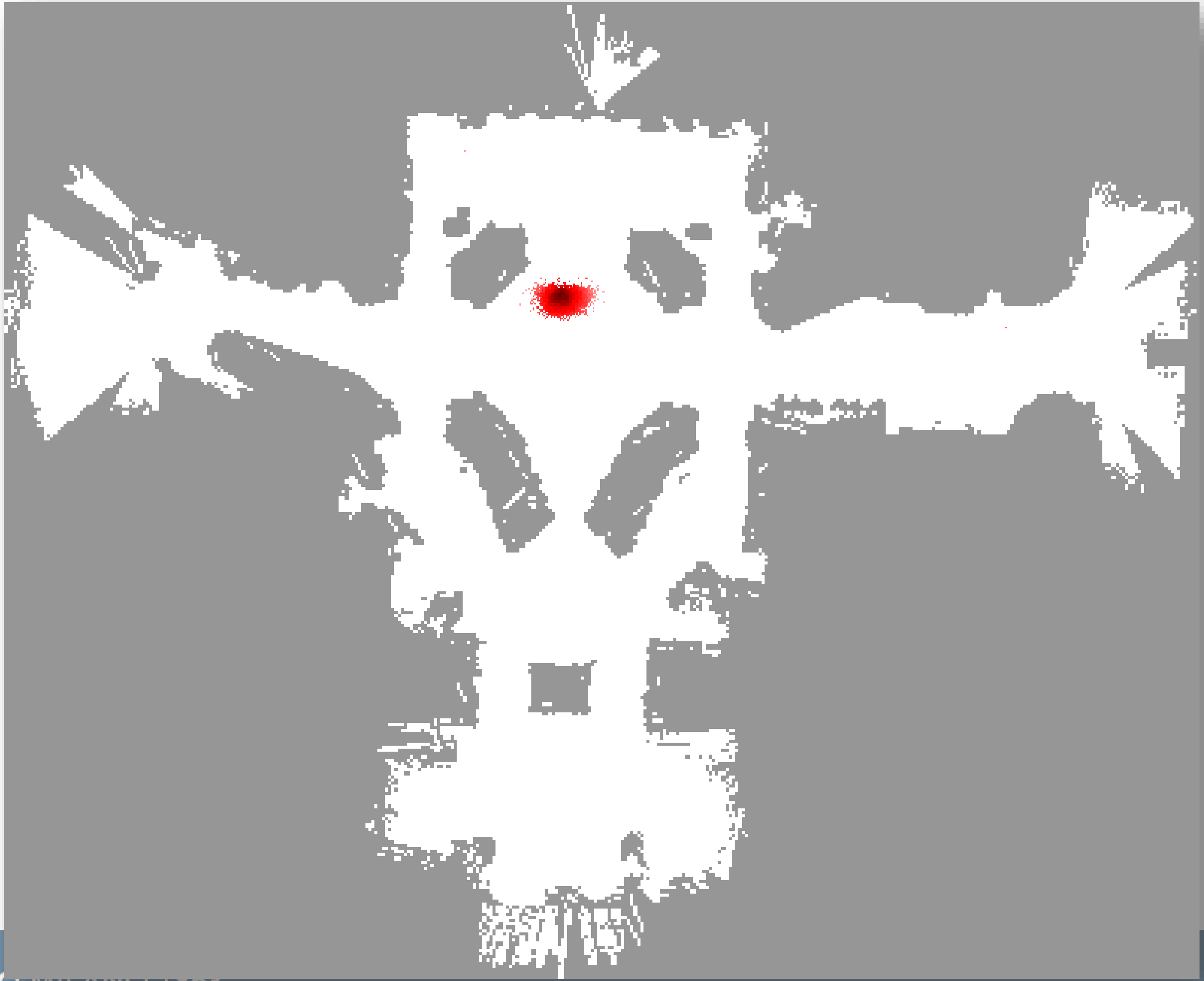


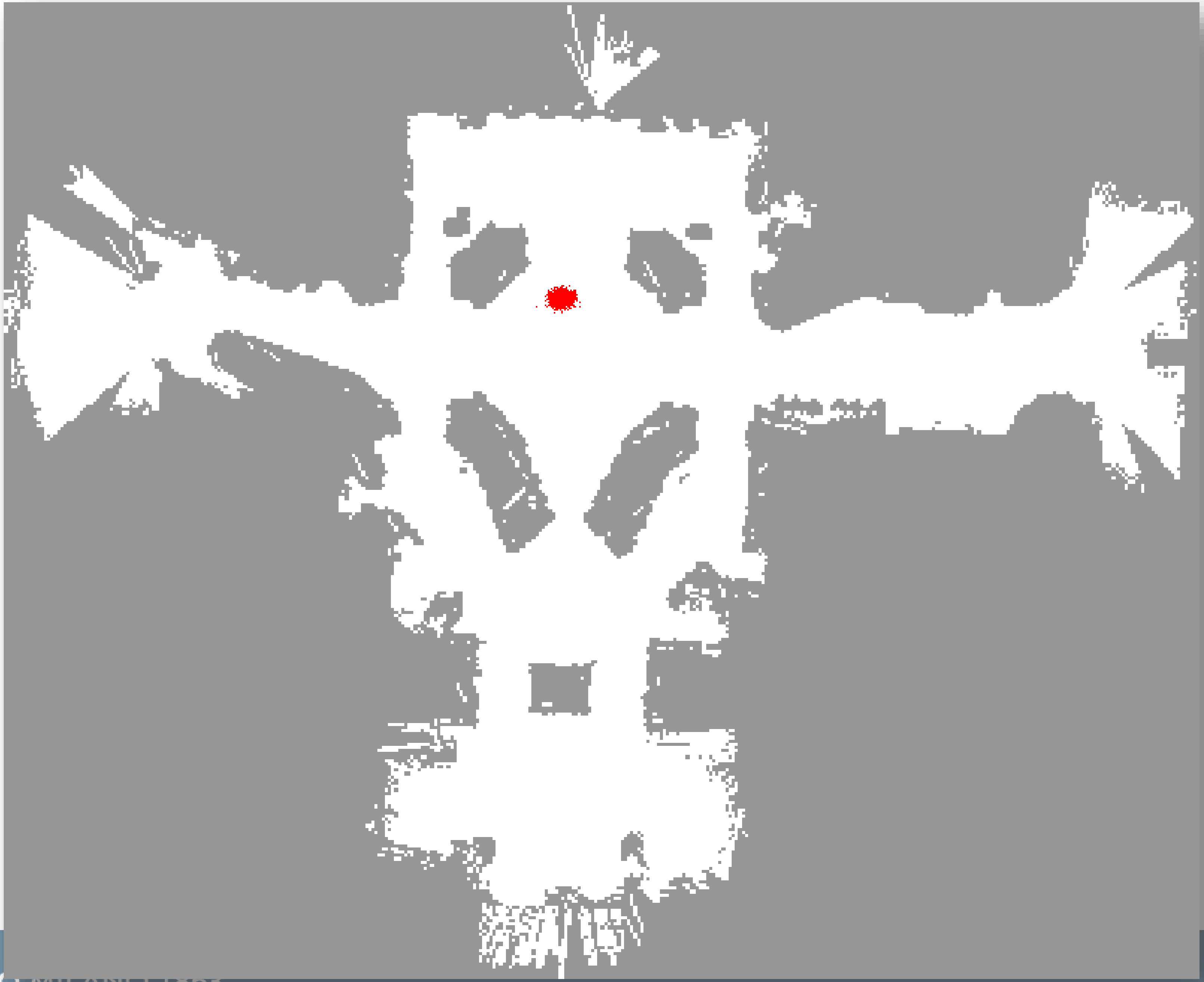


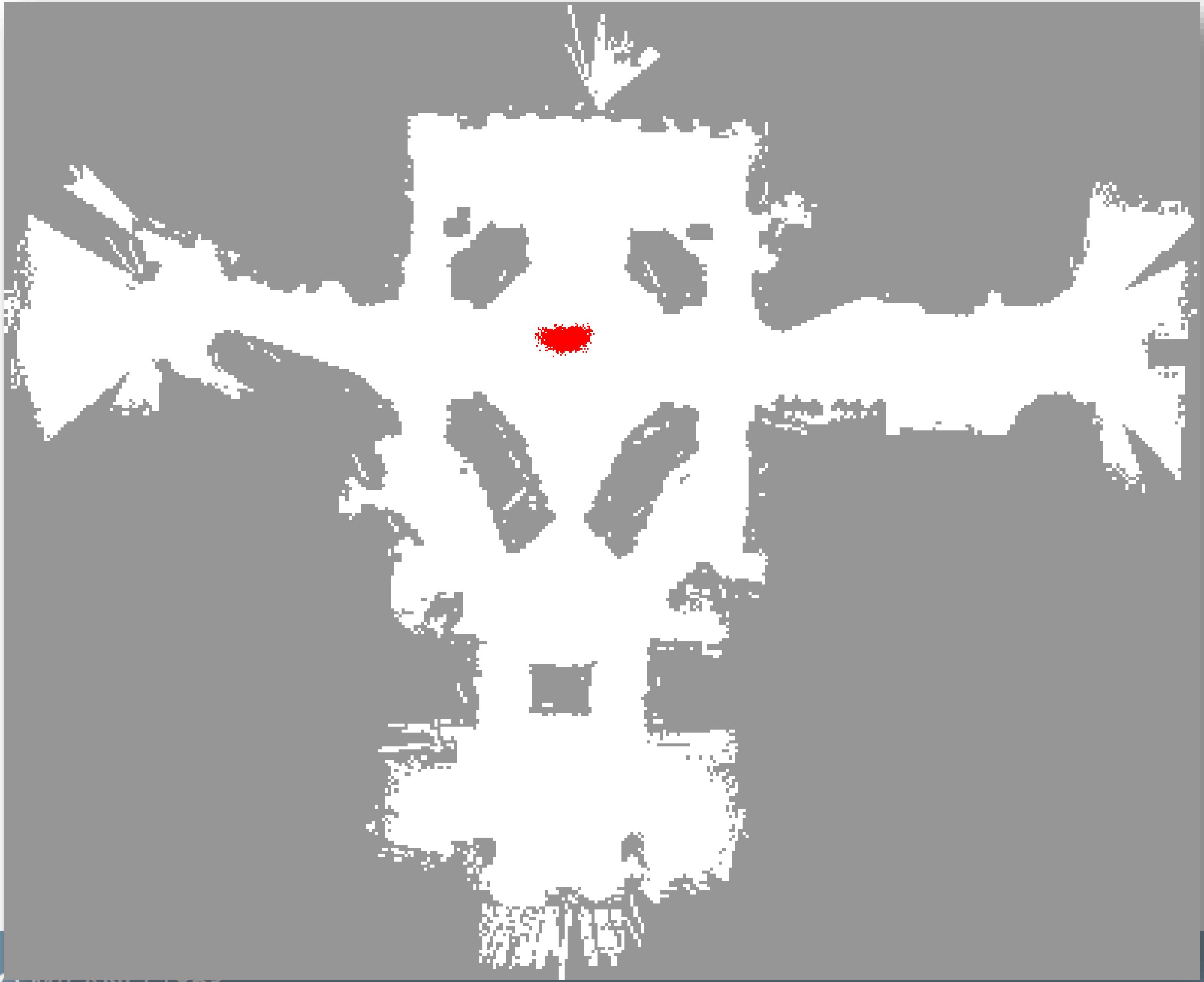


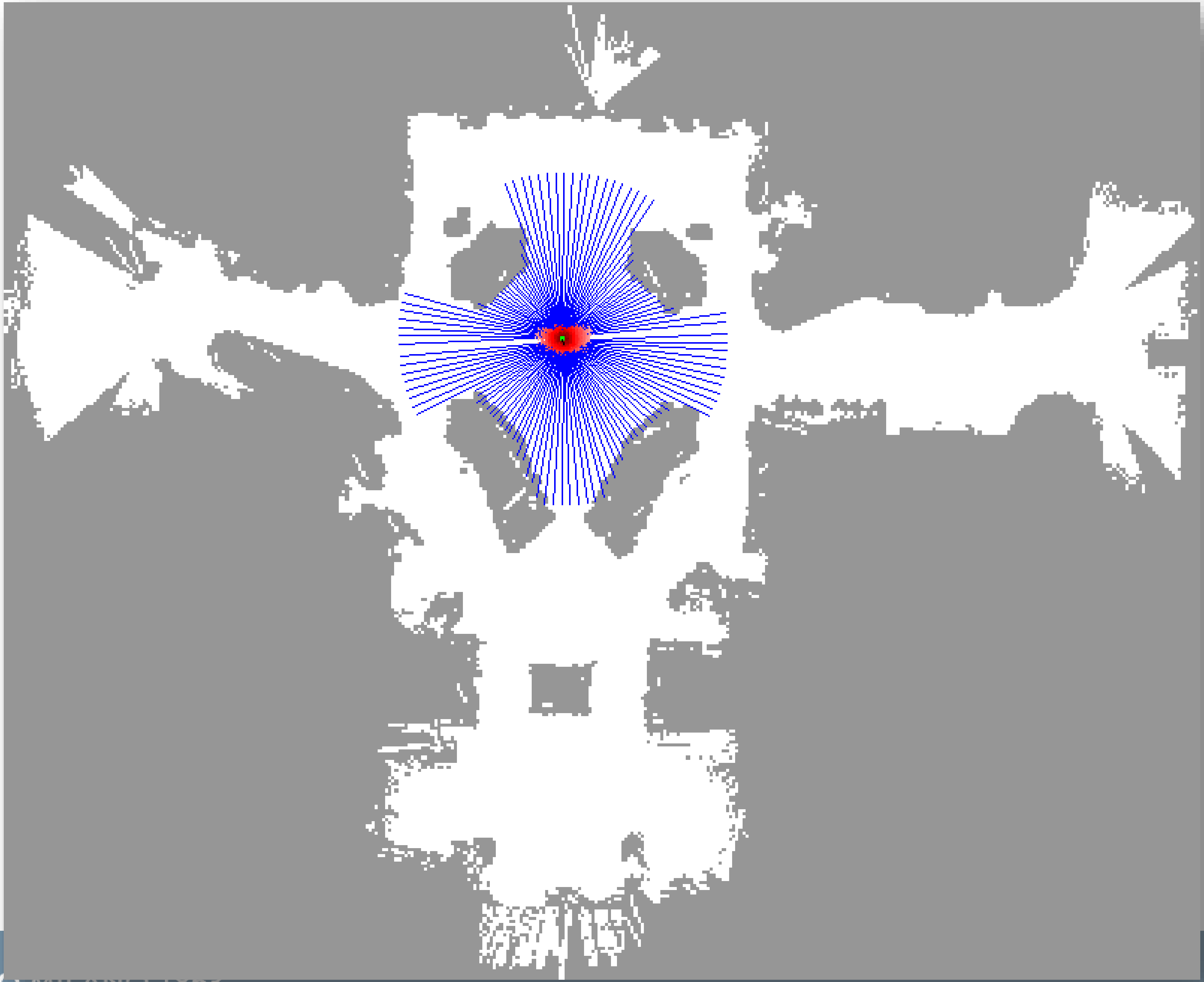


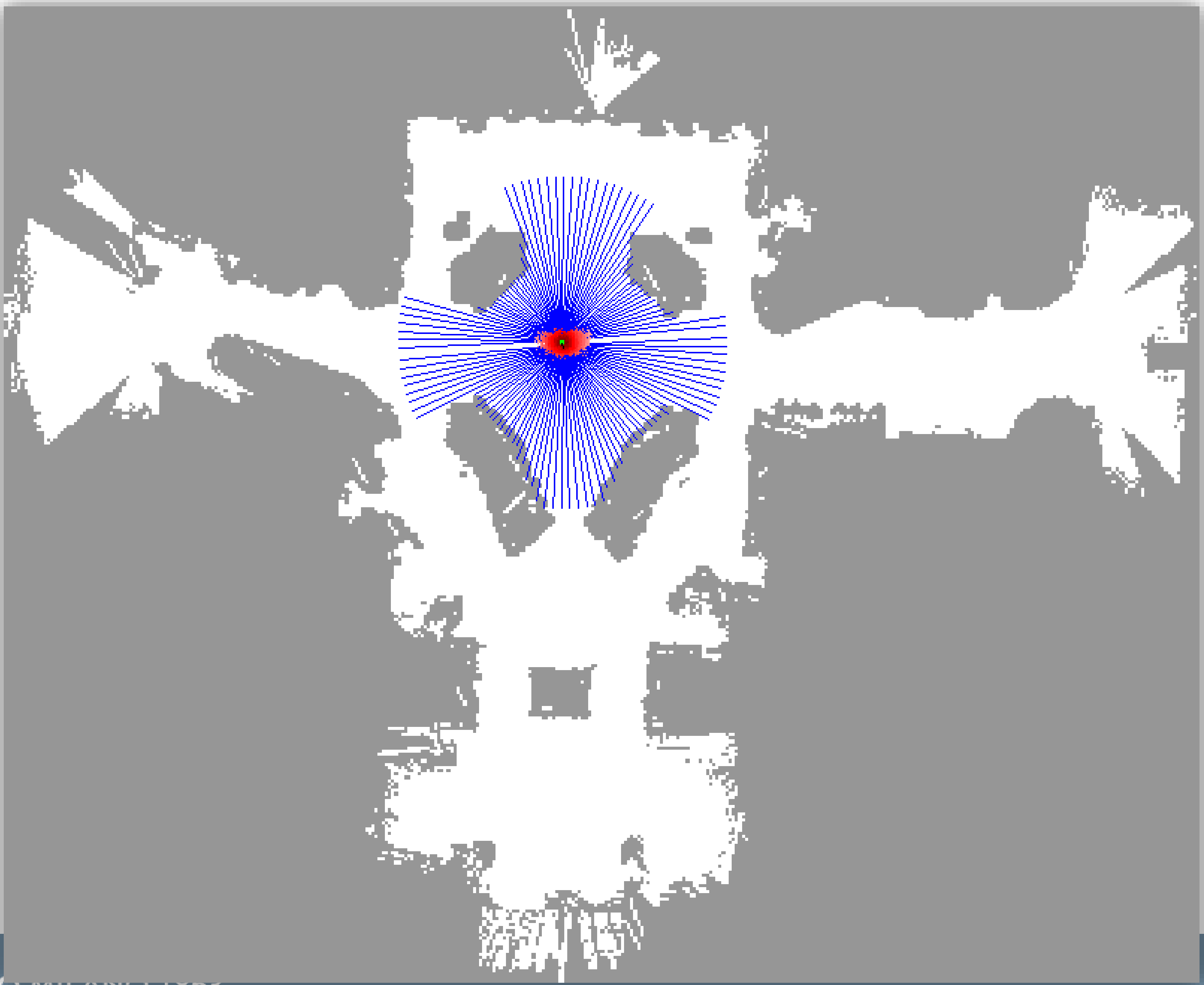




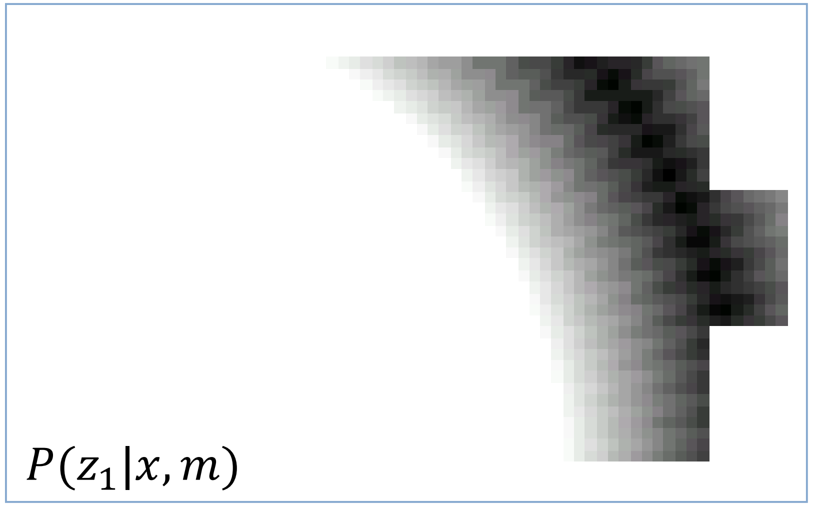


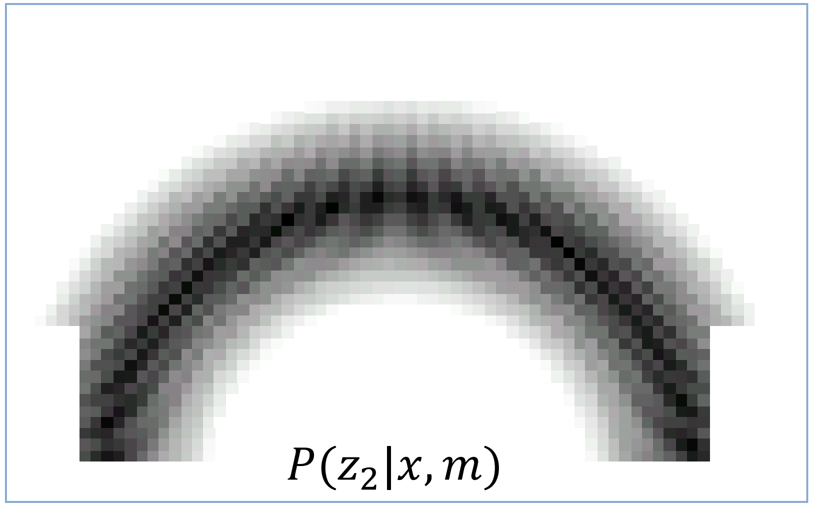


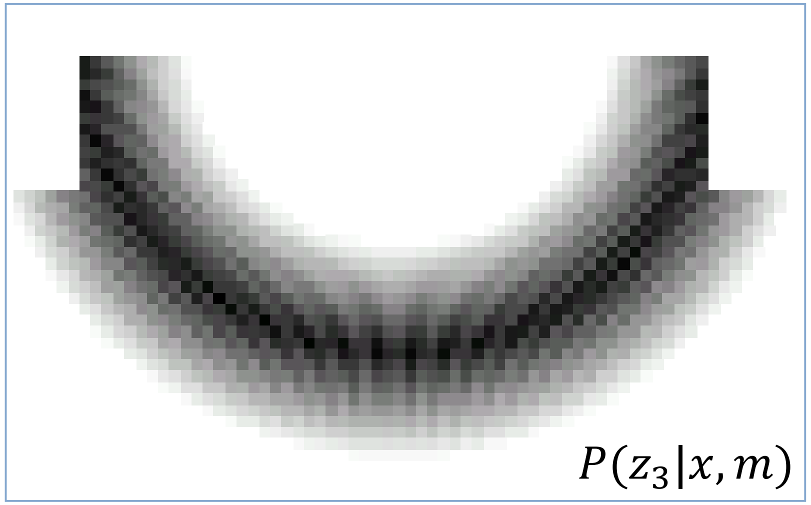


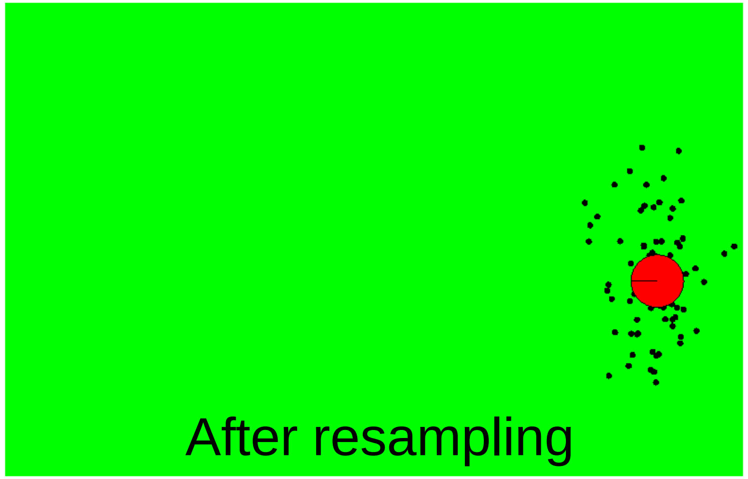


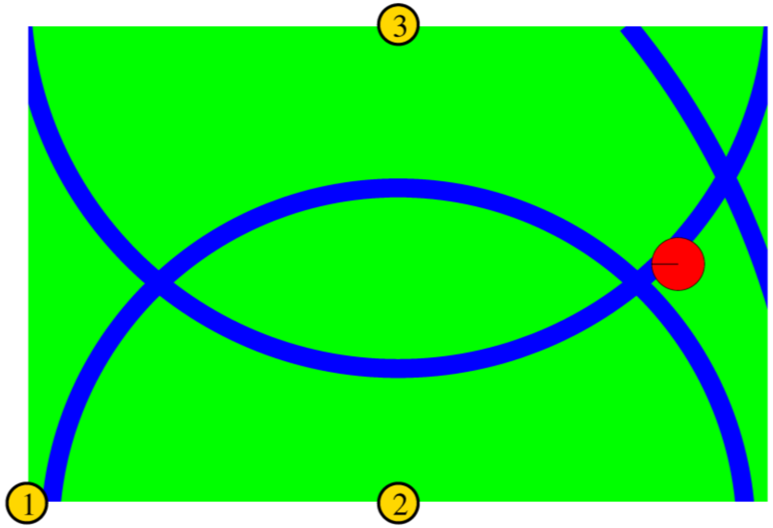
# RoboCup Example


$P(z_1|x,m)$

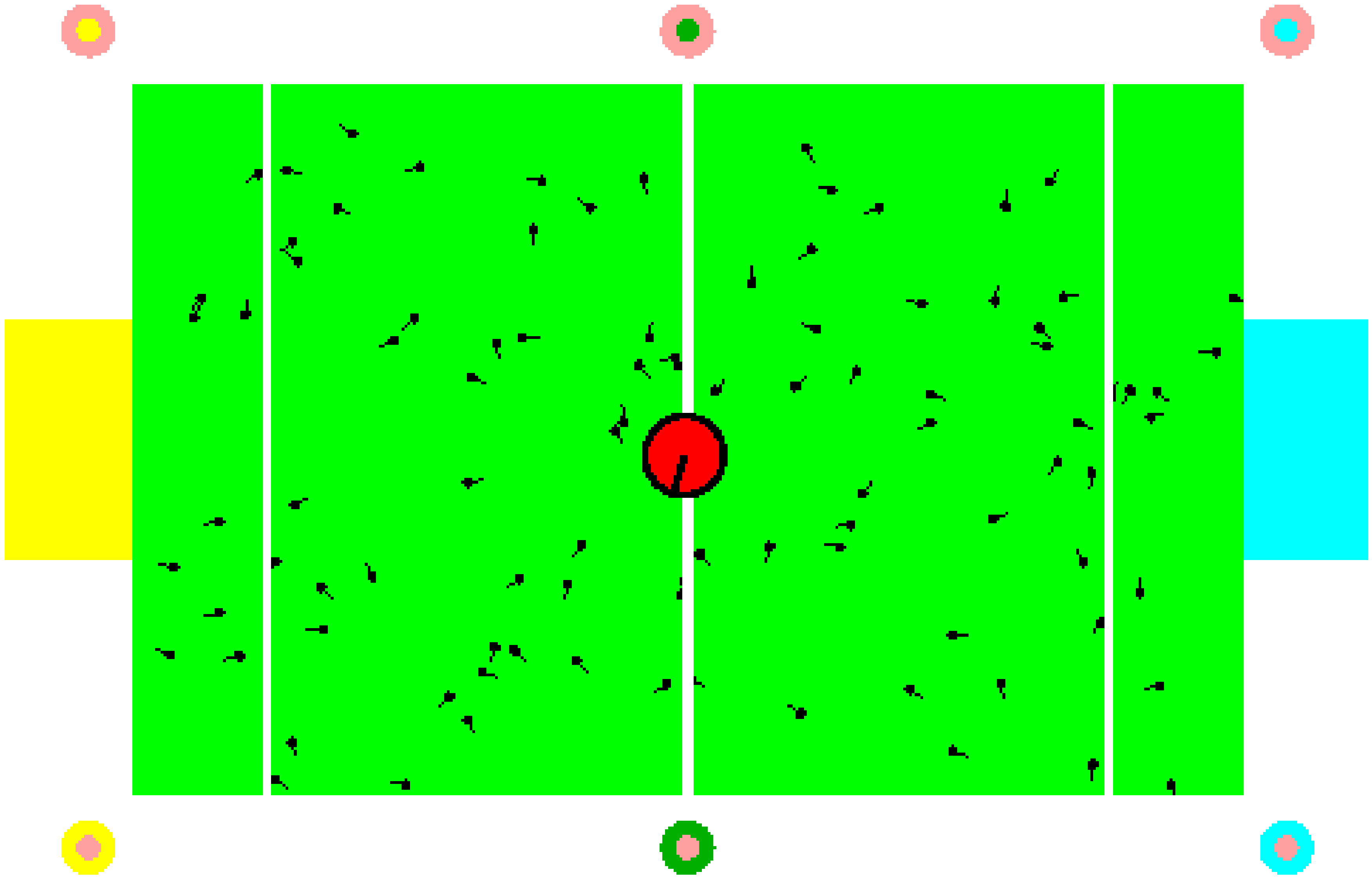

$P(z_2|x,m)$


$P(z_3|x,m)$



# Localization for AIBO robots



# Project Minerva

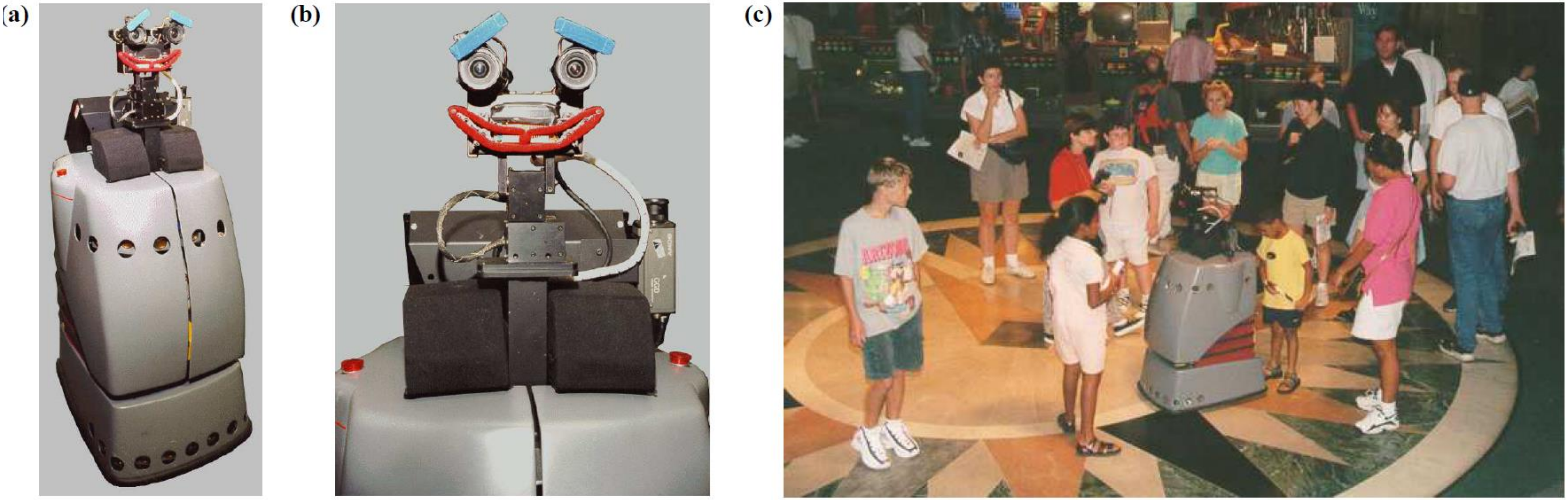
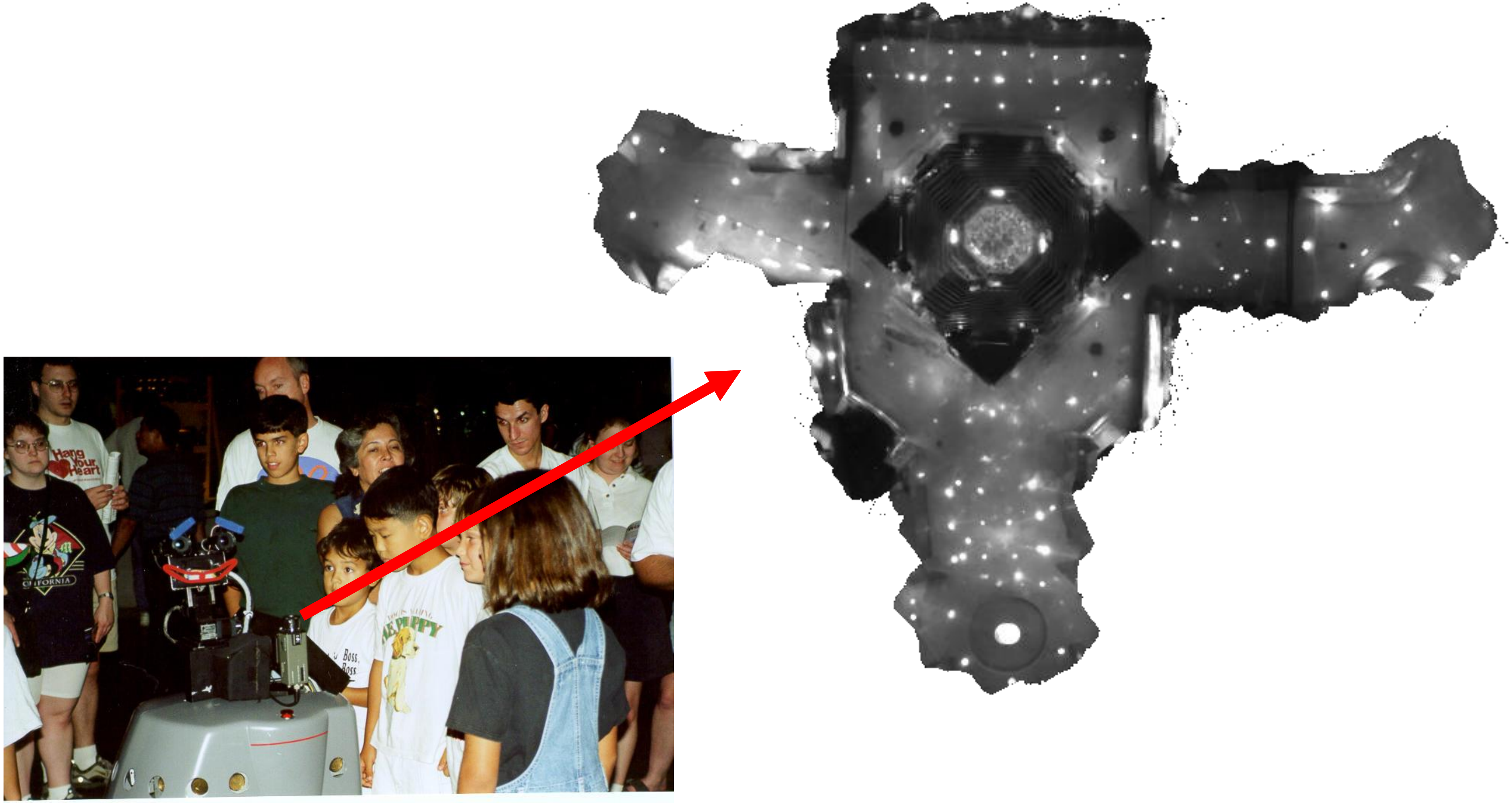
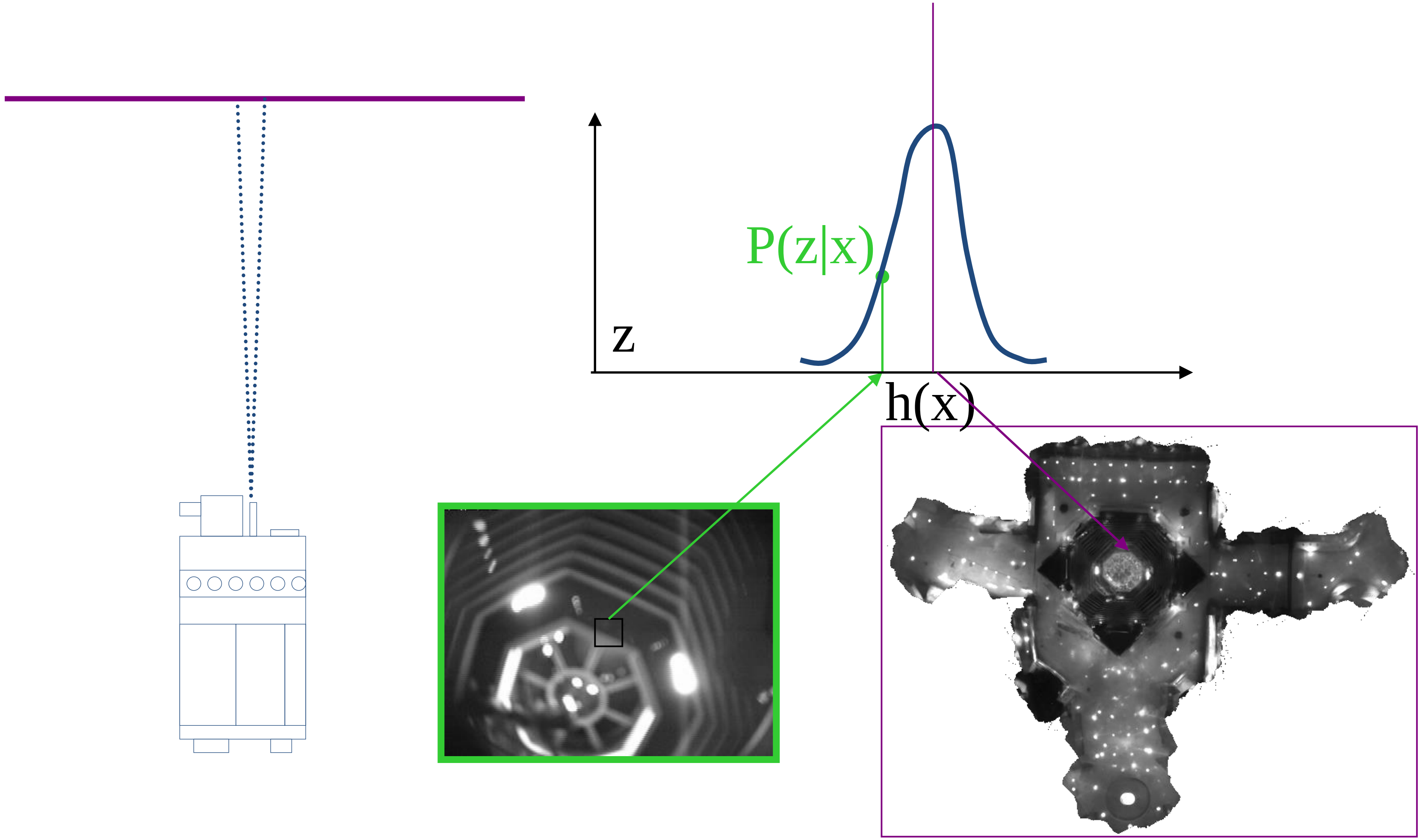



**Figure 1**: (a) Minerva. (b) Minerva's motorized face. (c) Minerva gives a tour in the Smithsonian's National Museum of American History.

# Using Ceiling Maps for Localization

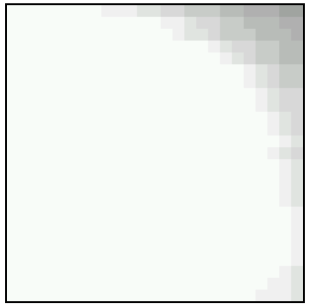


# Vision-based Localization

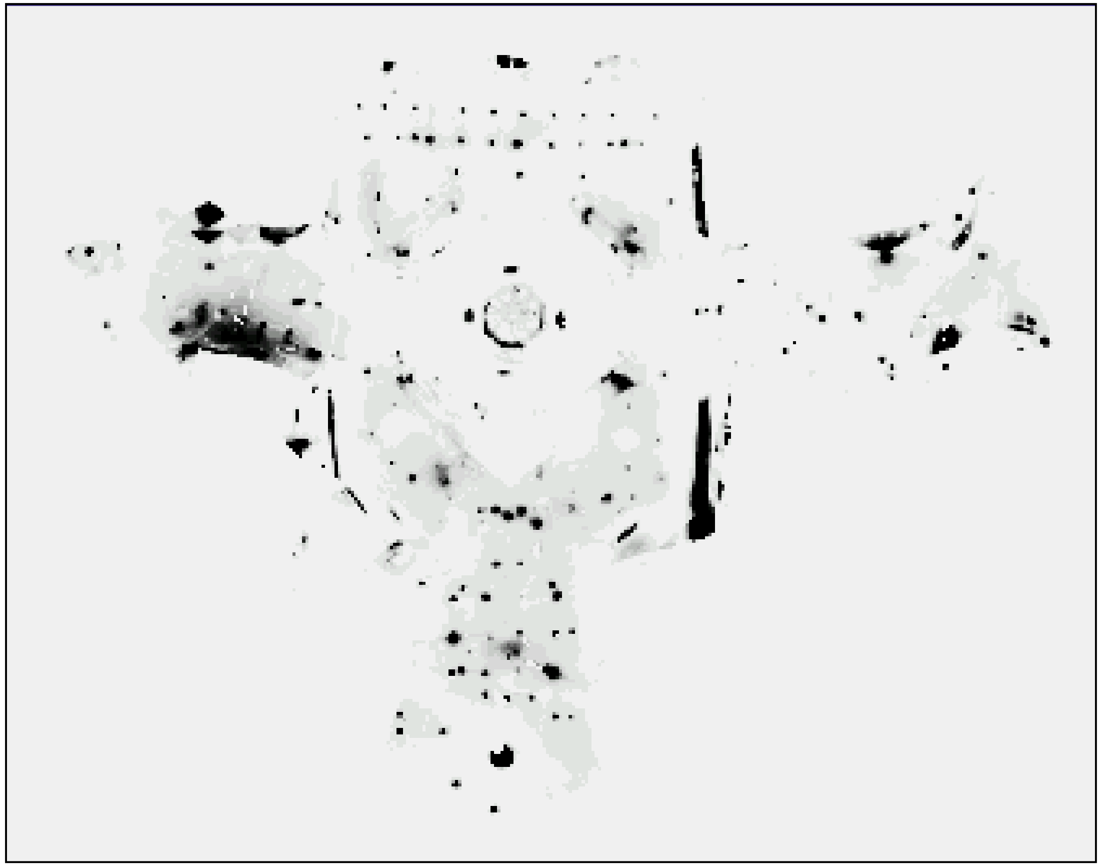


# Under a Light

**Measurement z:** 

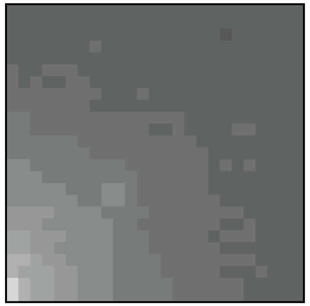


***P(z|x)*:**

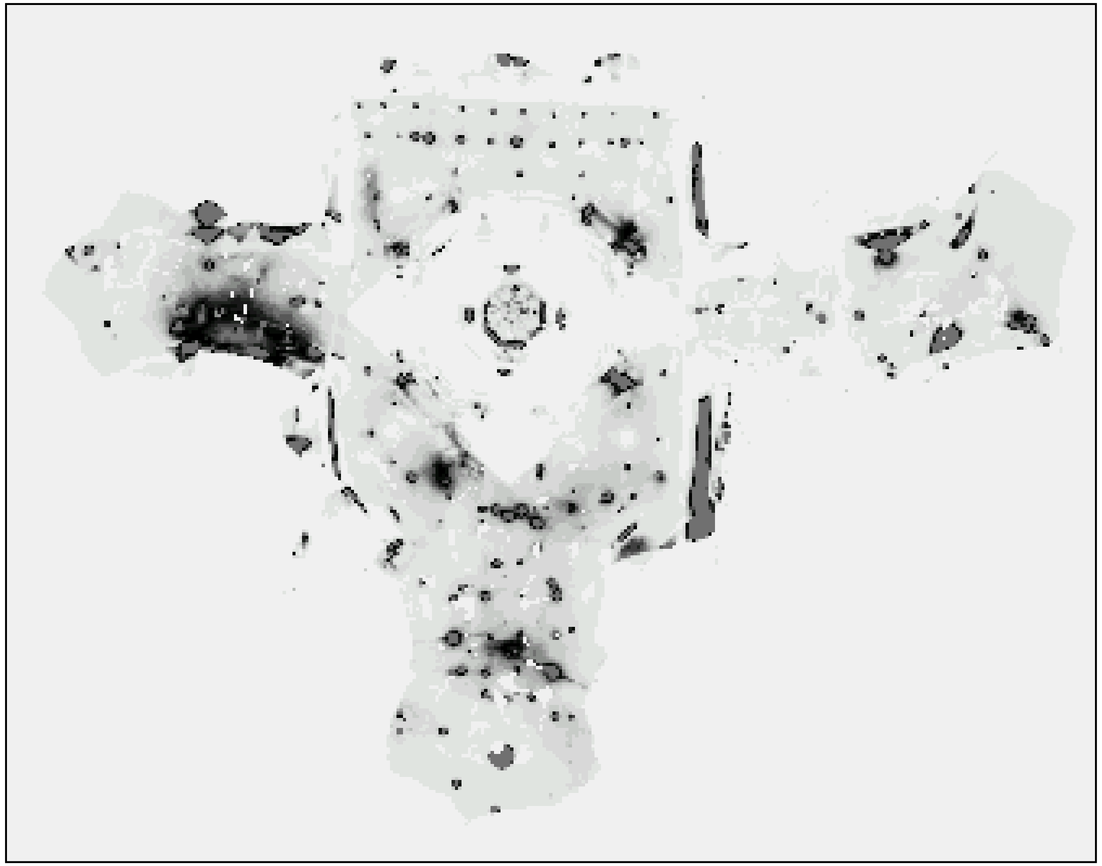


# Next to a Light

**Measurement z:**

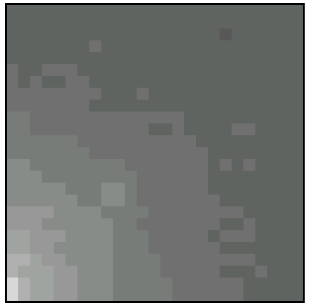


$P(z|x)$:

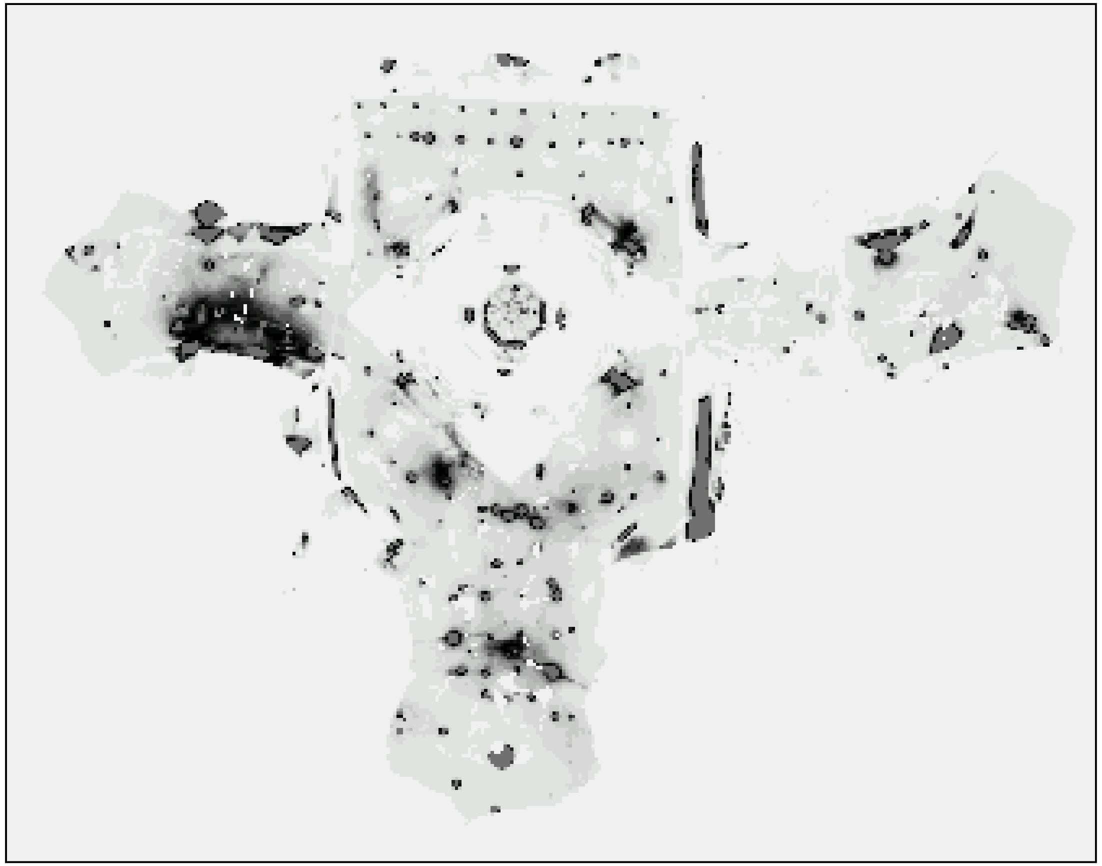


# Next to a Light

**Measurement z:**



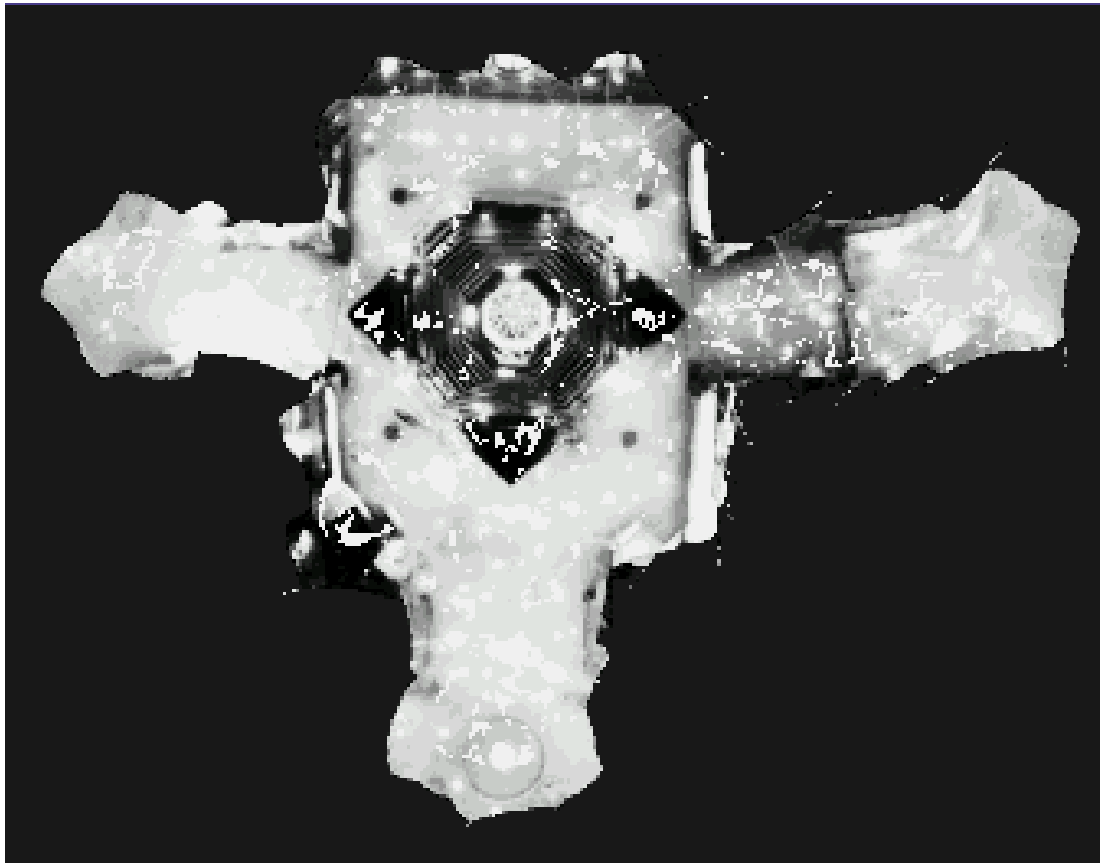
***P(z|x)*:**



# Elsewhere

**Measurement z:**

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



# Global Localization Using Vision