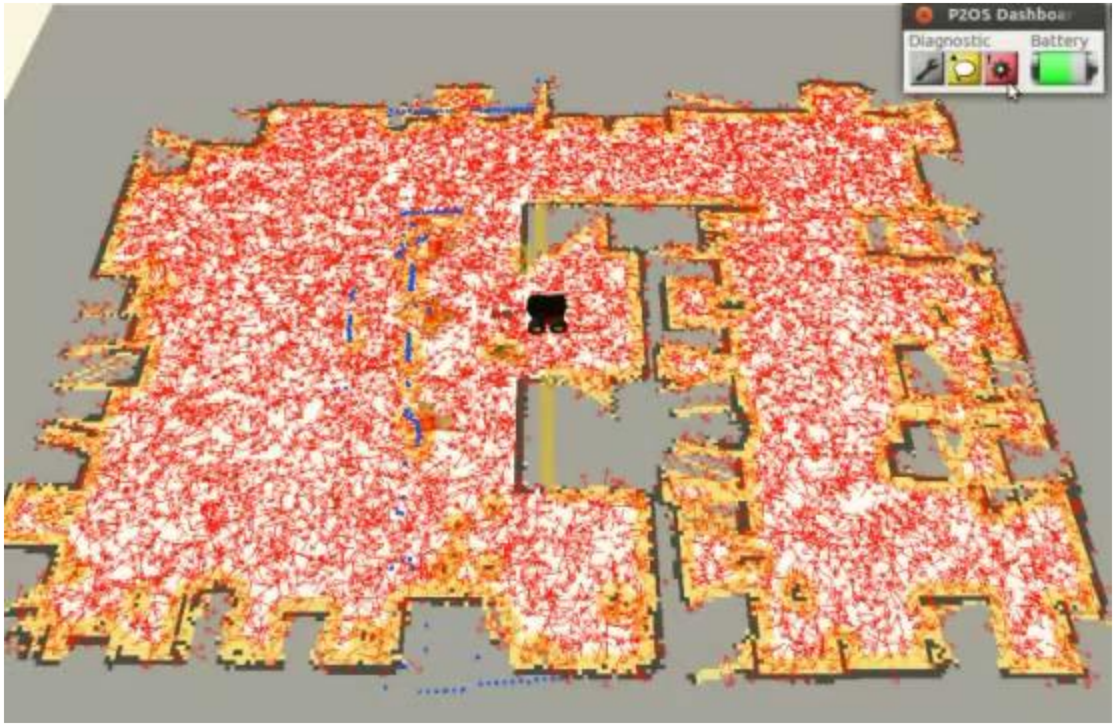
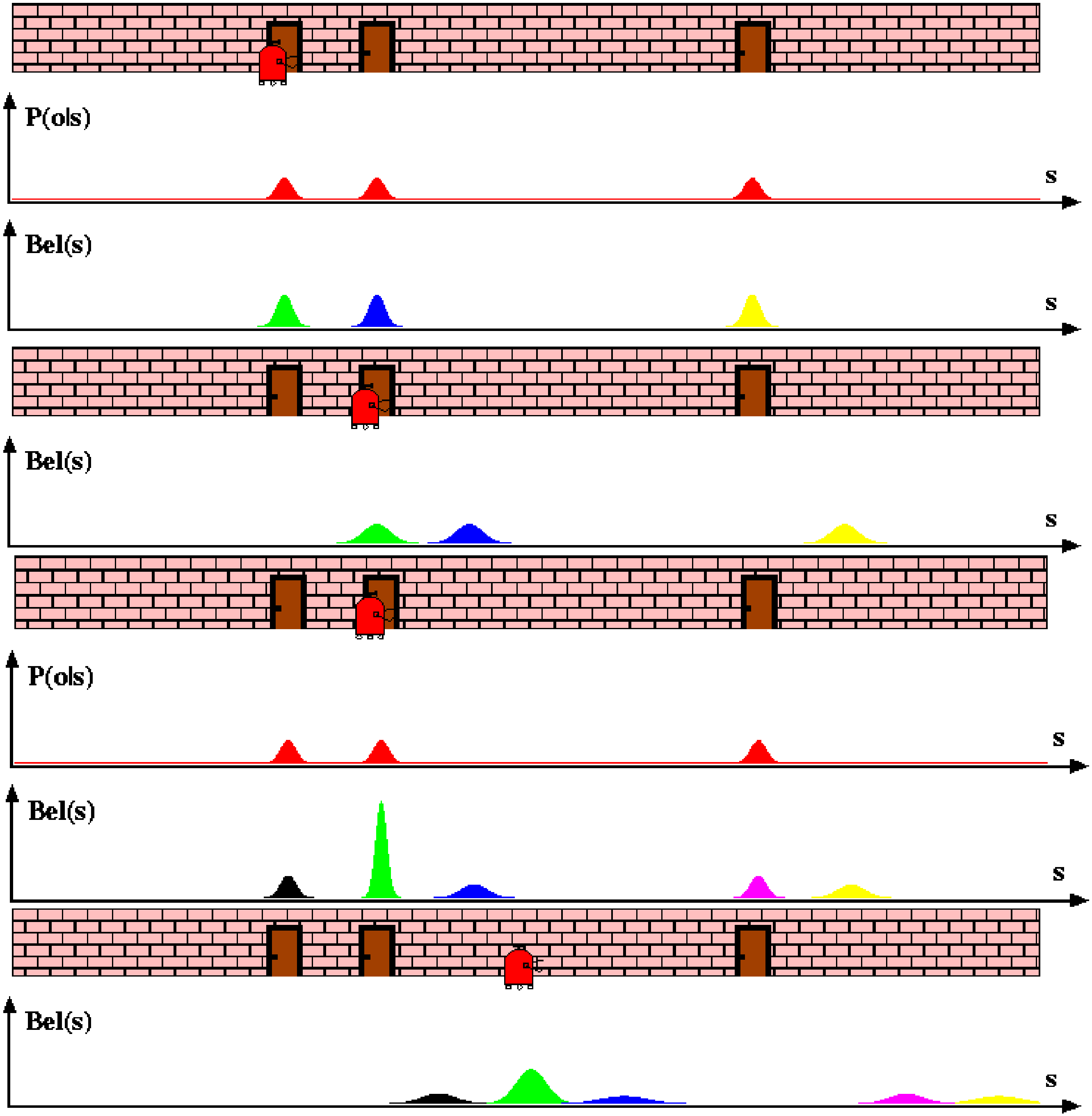
Chapter 9: Bayesian Methods

CS 336/436

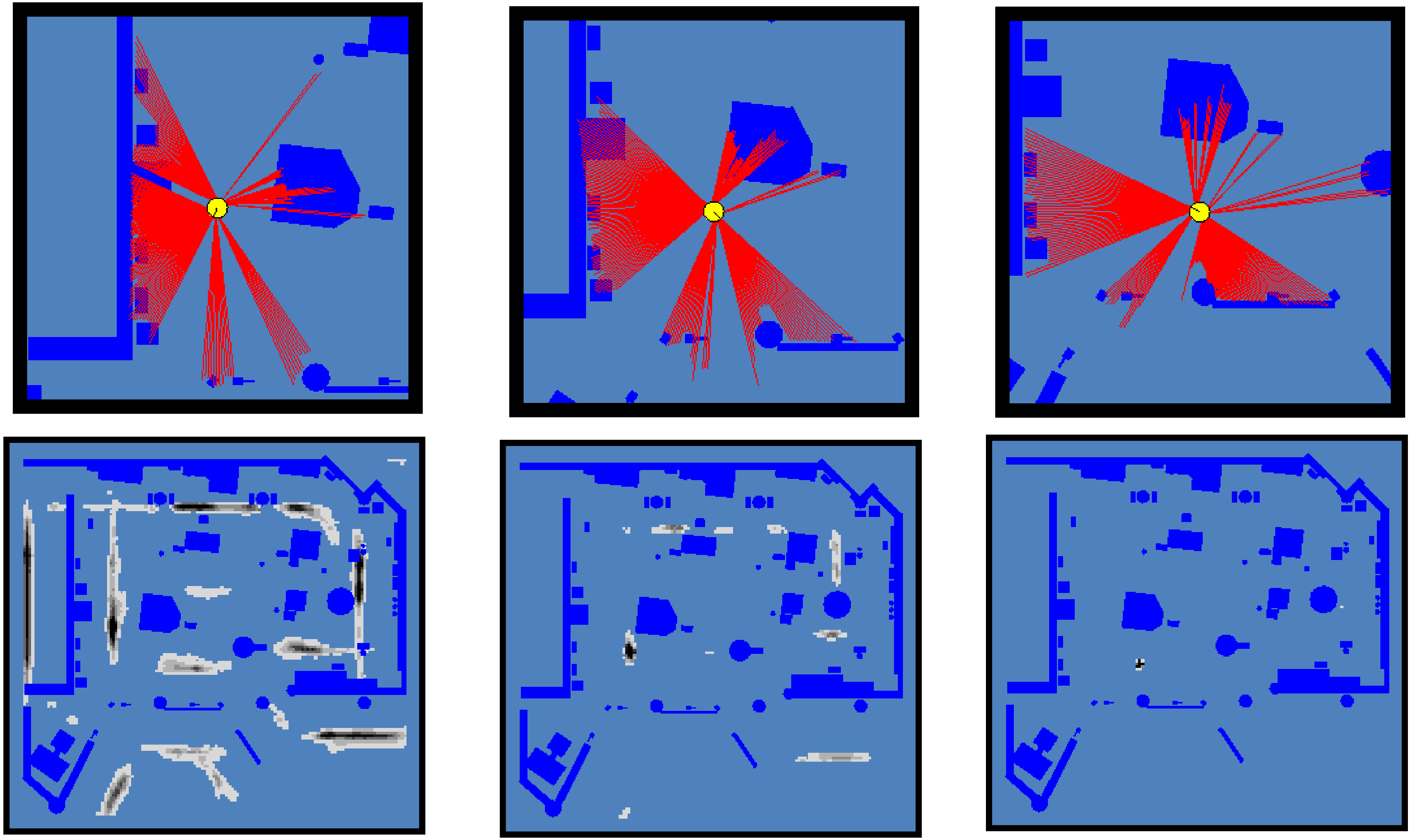
Gregory D. Hager


P2OS Dashboar
Diagnostic
Battery

A Simple Example



Why Not Just Use a Kalman Filter?



Recall Robot Localization

- Given
  - Sensor readings $y_{1,} y_2, \ldots y_k = y_{1:k}$
  - Known control inputs $u_0, u_2, .. u_k = u_{0:k}$
  - Known model $P(x_{t+1} \mid x_{t,} u_t)$ with initial $P(x_1 \mid u_0)$
  - **Known map** $P(y_t \mid x_t)$ → Most likely sensor reading given state x
- Compute
  - $P(x_t \mid y_{1:t-1,} u_{0:t-1})$ → Most likely state x at time t given a sequence of commands $u_{0:t-1}$ and measurements $y_{1:t-1}$

This is just a probabilistic representation of what you've already learned!

Let's try to connect the dots and do a couple of examples

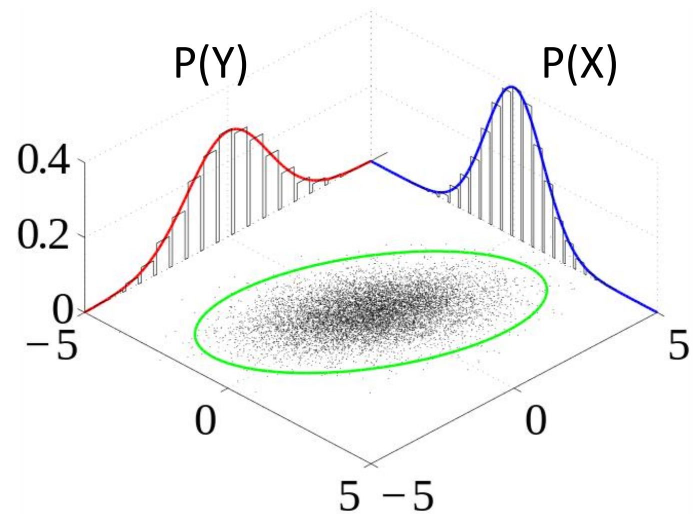
Some Probability Reminders

- P(x,y) = P(x | y) P(y)
- P(x) = $\sum_y$ P(x, y) = $\sum_y$ P(x | y) P(y)
- If x is independent of z given y
  - P(x | y, z) = P(x | y)
- Two important assumptions
  1. Markov

     $$P(x_k \mid x_{k-1}, \ldots, x_0) = P(x_k \mid x_{k-1})$$

  2. Observation

     $$P(y_k \mid x_k, \ldots, x_0) = P(y_k \mid x_k)$$

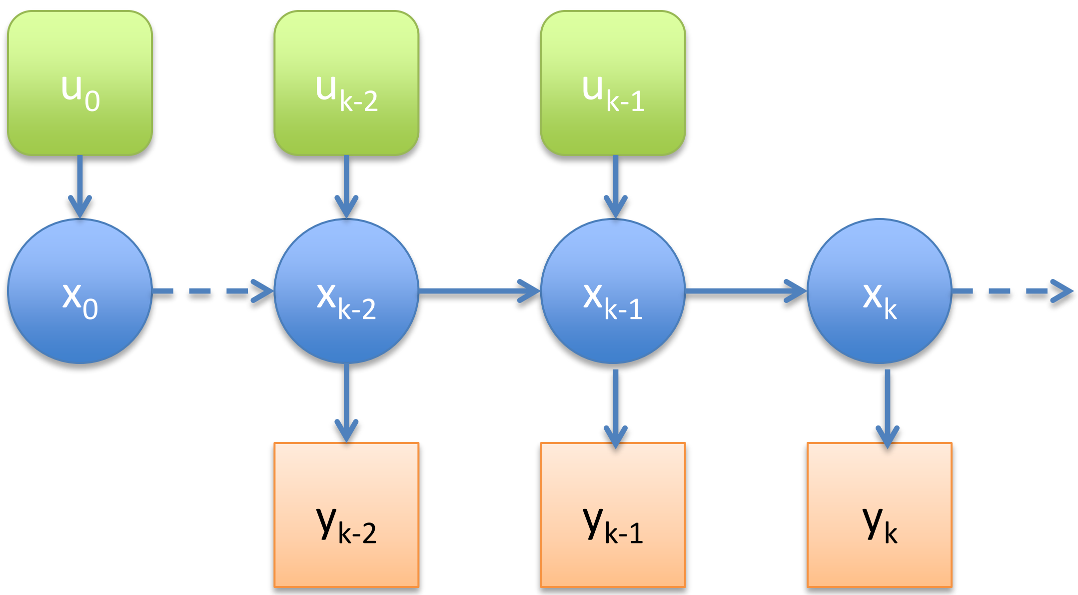


wikipedia

Bayes Filter

- Given a sequence of measurements $y_1, \ldots, y_k$ and a sequence of commands $u_0, \ldots, u_{k-1}$: $\{ y_{1:k}, u_{0:k-1} \}$
- Given a sensor model $P(y_k \mid x_k)$
- Given a dynamic model $P(x_k \mid x_{k-1})$
- Given a prior probability $P(x_0)$

Find $P(x_k \mid y_{1:k}, u_{0:k-1})$



Bayesian Localization

- Recall Bayes Theorem:

$$P(x \mid y) = P(y \mid x)\, P(x) / P(y)$$

- Also remember conditional independence

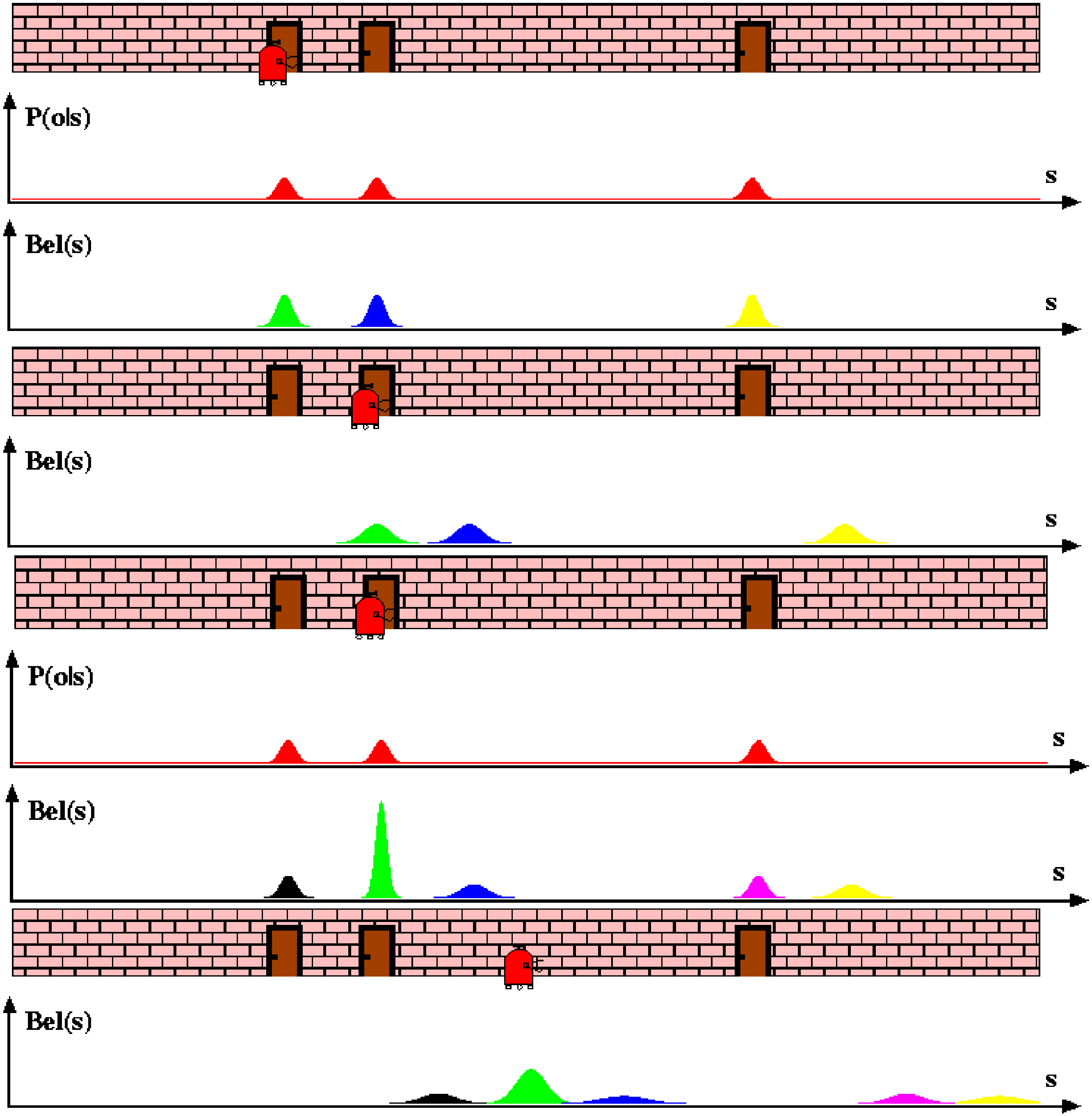
- Think of x as the state of the robot and y as the data we know

$P(x_k \mid u_{0:k-1}, y_{1:k}) \longrightarrow$ *Posterior Probability Distribution*

$$P(x_k \mid u_{0:k-1}, y_{1:k}) = P(y_k \mid x_k, u_{0:k-1}, y_{1:k-1})\ P(x_k \mid u_{0:k-1}, y_{1:k-1}) \ /\ P(y_k \mid u_{0:k-1}, y_{1:k-1})$$

$$= \eta_k \underbrace{P(y_k \mid x_k)}_{\textbf{observation}} \int_{xk-1} \underbrace{P(x_k \mid u_{k-1}, x_{k-1})}_{\textbf{state prediction}}\ \underbrace{P(x_{k-1} \mid u_{0:k-2}, y_{1:k-1})}_{\textbf{recursive instance}}$$

A Simple Example



Ways of Representing Probabilities

- We have already seen Kalman filters
  - represent probabilities as Gaussians
  - Gaussians are *conjugate* distributions
- How arbitrary distributions are represented?
  - Mixtures of Gaussians
- Suppose instead state space is partitioned and probability in partition is constant
  - $P(x \mid y) = P(y \mid x)\, P(x) / P(y) \approx P(x_i \mid y) = \eta\, P(y \mid x_i)\, P(x_i)$
  - $\eta = \sum_i P(y \mid x_i)\, P(x_i)$
  - Thus, updating from observations is a simple multiplication of prior probability by likelihood of observation
  - $P(x_i(k) \mid u(k-1:0), y(k-1)) = \sum_j P(x_i(k) \mid u(k-1), x_j(k-1))\, P(x_j(k-1) \mid u(k-2:0), y(k-1))$
  - Thus, updating using dynamical model is simply a discrete convolution (blurring) of the prior by the driving noise of the planned motion

A blast from the past: http://arxiv.org/pdf/1304.2757.pdf

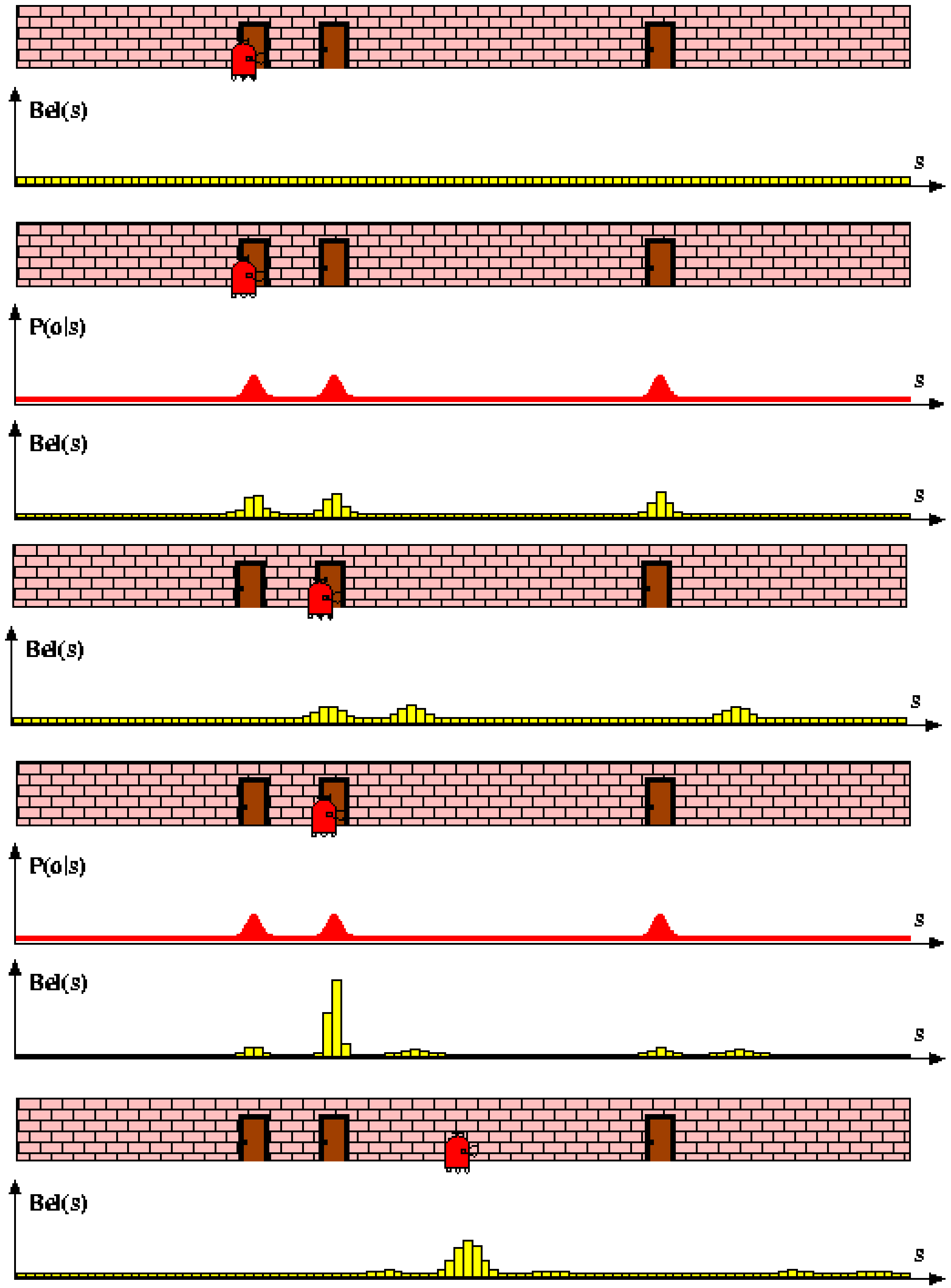
How Do We Think about Motion?

- Suppose we have $P(x_k)$
- We have $P(x_{k+1} \mid x_k, u_k)$

- Put together
- $P(x_{k+1}) = \int P(x_{k+1} \mid x_k, u_k) P(x_k)\, dx_k$

What is the probability distribution for x_{k+1} given the command u_k and all the previous states x_k?

Piecewise Constant Representation



Updating from observations is a simple multiplication of prior probability by likelihood of observation

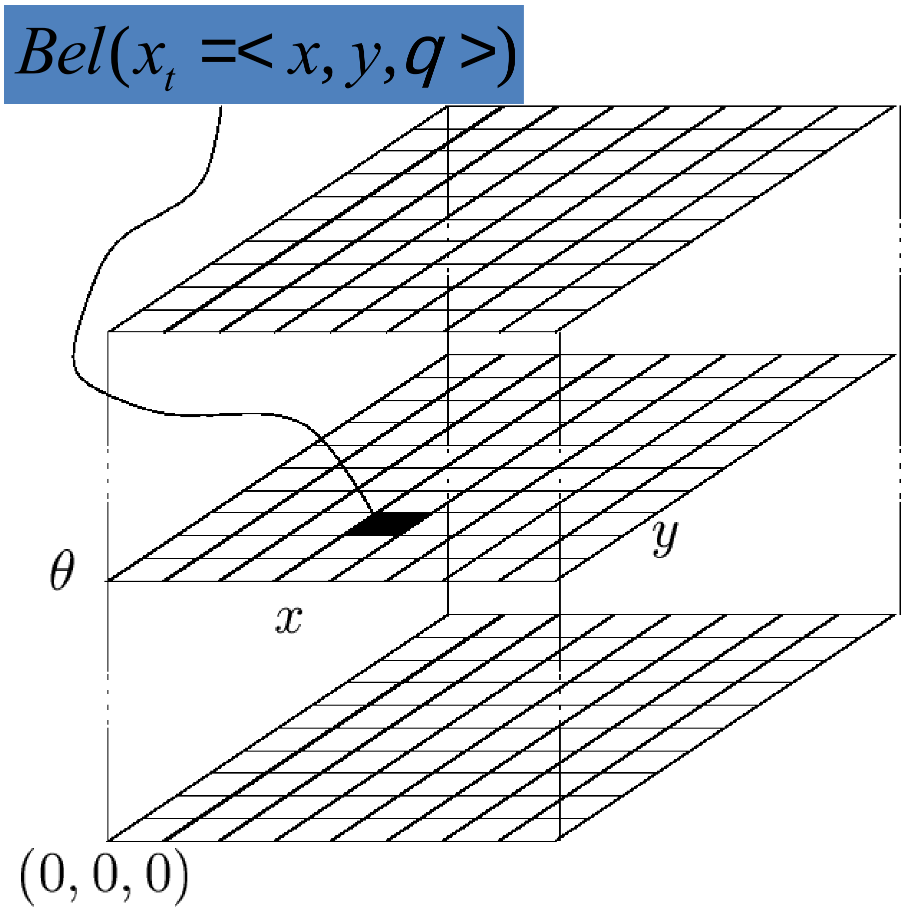
Updating using dynamical model is simply a discrete convolution (blurring) of the prior by the driving noise of the planned motion

Updating from observations is a simple multiplication of prior probability by likelihood of observation

Updating using dynamical model is simply a discrete convolution (blurring) of the prior by the driving noise of the planned motion

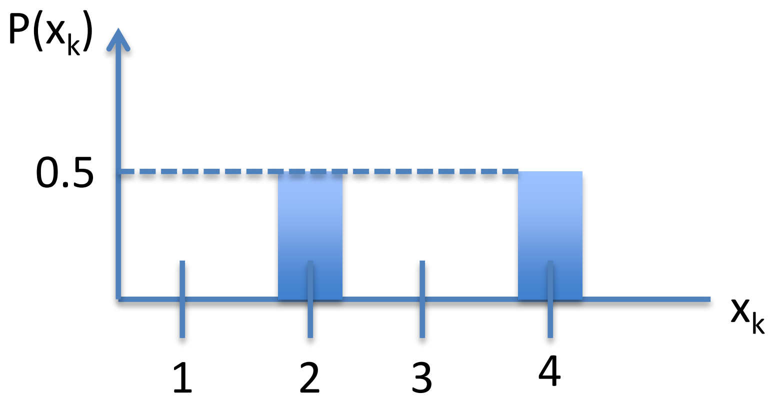
Piecewise Constant Representation (Mobile Robot)

Position of a mobile robot: (x, y, θ)



Propagating Motion

Prior $P(x_k)$



State space X = { 1, 2, 3, 4}

$P(x_{k+1} \mid x_k, u_k)$	$X_{k+1}=1$	$X_{k+1}=2$	$X_{k+1}=3$	$X_{k+1}=4$
$X_k=1$	0.25	0.5	0.25	0
$X_k=2$	0	0.25	0.5	0.25
$X_k=3$	0	0	0.25	0.75
$X_k=4$	0	0	0	1

Transition matrix: The probability $P(j|i)$ of moving from i to j is given by $P_{i,j}$. Each row must sum to 1.

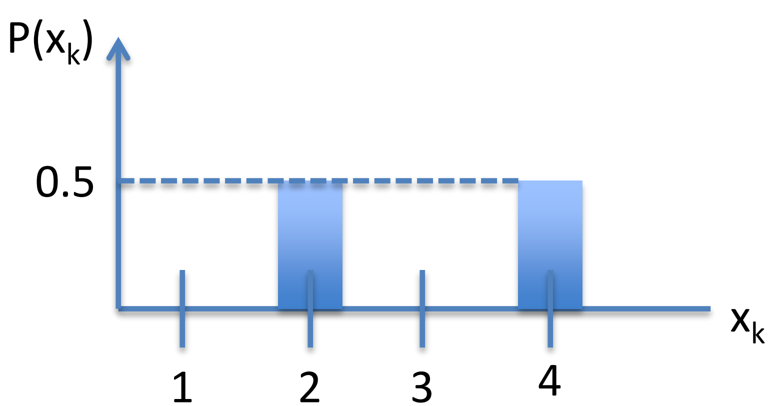
Compute $P(x_{k+1}) = \int P(x_{k+1} \mid x_k, u_k) P(x_k) dx_k$

Propagating Motion

State space X = { 1, 2, 3, 4}

$P(x_{k+1} \mid x_k, u_k)$	$X_{k+1}=1$	$X_{k+1}=2$	$X_{k+1}=3$	$X_{k+1}=4$
$X_k=1$	0.25	0.5	0.25	0
$X_k=2$	0	0.25	0.5	0.25
$X_k=3$	0	0	0.25	0.75
$X_k=4$	0	0	0	1

Prior $P(x_k)$



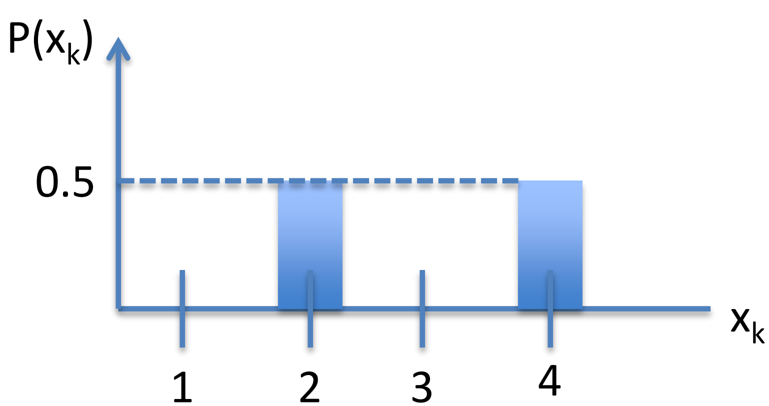
$$
\begin{aligned}
P(x_{k+1}=1) &= \sum_{x' \in X} P(x_{k+1}=1 \mid x'_k, u_k) P(x'_k) \\
&= P(x_{k+1}=1 \mid x'_k=1, u_k) P(x'_k=1) + \\
&\quad P(x_{k+1}=1 \mid x'_k=2, u_k) P(x'_k=2) + \\
&\quad P(x_{k+1}=1 \mid x'_k=3, u_k) P(x'_k=3) + \\
&\quad P(x_{k+1}=1 \mid x'_k=4, u_k) P(x'_k=4) \\
&= 0.25 \times 0 + 0 \times 0.5 + 0 \times 0 + 0 \times 0.5 \\
&= 0
\end{aligned}
$$

Propagating Motion

State space X = { 1, 2, 3, 4}

$P(x_{k+1} \mid x_k, u_k)$	$X_{k+1}=1$	$X_{k+1}=2$	$X_{k+1}=3$	$X_{k+1}=4$
$X_k=1$	0.25	0.5	0.25	0
$X_k=2$	0	0.25	0.5	0.25
$X_k=3$	0	0	0.25	0.75
$X_k=4$	0	0	0	1

Prior $P(x_k)$



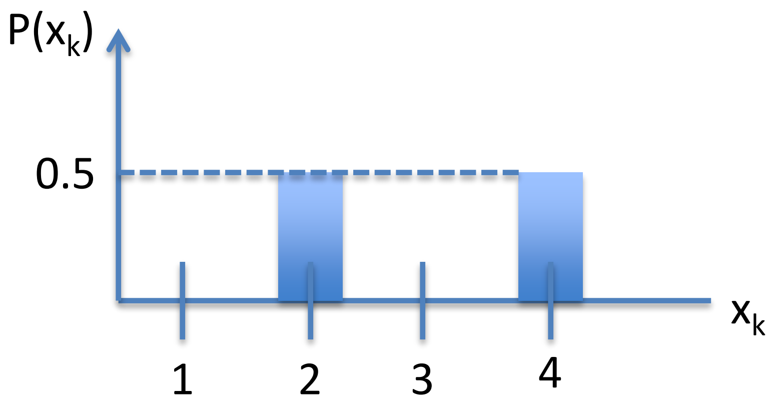
$$
\begin{aligned}
P(x_{k+1}=2) &= \sum_{x' \in X} P(x_{k+1}=2 \mid x'_k, u_k) P(x'_k) \\
&= P(x_{k+1}=2 \mid x'_k=1, u_k) P(x'_k=1) + \\
&\quad P(x_{k+1}=2 \mid x'_k=2, u_k) P(x'_k=2) + \\
&\quad P(x_{k+1}=2 \mid x'_k=3, u_k) P(x'_k=3) + \\
&\quad P(x_{k+1}=2 \mid x'_k=4, u_k) P(x'_k=4) \\
&= 0.5 \times 0 + 0.25 \times 0.5 + 0 \times 0 + 0 \times 0.5 \\
&= 0.125
\end{aligned}
$$

Propagating Motion

State space X = { 1, 2, 3, 4}

$P(x_{k+1} \mid x_k, u_k)$	$X_{k+1}=1$	$X_{k+1}=2$	$X_{k+1}=3$	$X_{k+1}=4$
$X_k=1$	0.25	0.5	0.25	0
$X_k=2$	0	0.25	0.5	0.25
$X_k=3$	0	0	0.25	0.75
$X_k=4$	0	0	0	1

Prior $P(x_k)$



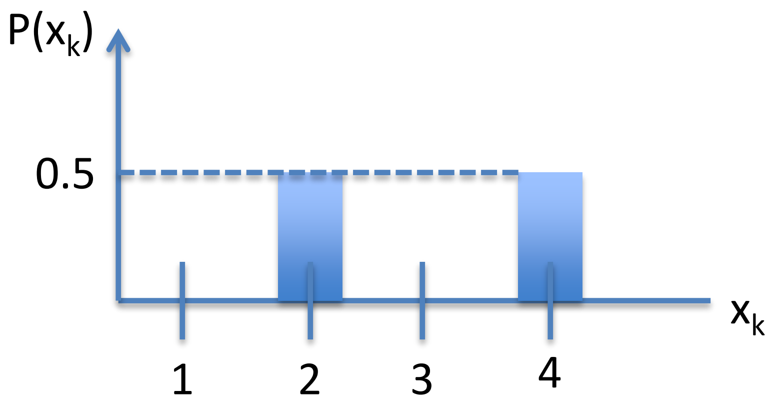
$$
\begin{aligned}
P(x_{k+1}=3) &= \sum_{x' \in X} P(x_{k+1}=3 \mid x'_k, u_k) P(x'_k) \\
&= P(x_{k+1}=3 \mid x'_k=1, u_k) P(x'_k=1) + \\
&\quad P(x_{k+1}=3 \mid x'_k=2, u_k) P(x'_k=2) + \\
&\quad P(x_{k+1}=3 \mid x'_k=3, u_k) P(x'_k=3) + \\
&\quad P(x_{k+1}=3 \mid x'_k=4, u_k) P(x'_k=4) \\
&= 0.25 \times 0 + 0.5 \times 0.5 + 0.25 \times 0 + 0 \times 0.5 \\
&= 0.25
\end{aligned}
$$

Propagating Motion

State space X = { 1, 2, 3, 4}

$P(x_{k+1} \mid x_k, u_k)$	$X_{k+1}=1$	$X_{k+1}=2$	$X_{k+1}=3$	$X_{k+1}=4$
$X_k=1$	0.25	0.5	0.25	0
$X_k=2$	0	0.25	0.5	0.25
$X_k=3$	0	0	0.25	0.75
$X_k=4$	0	0	0	1

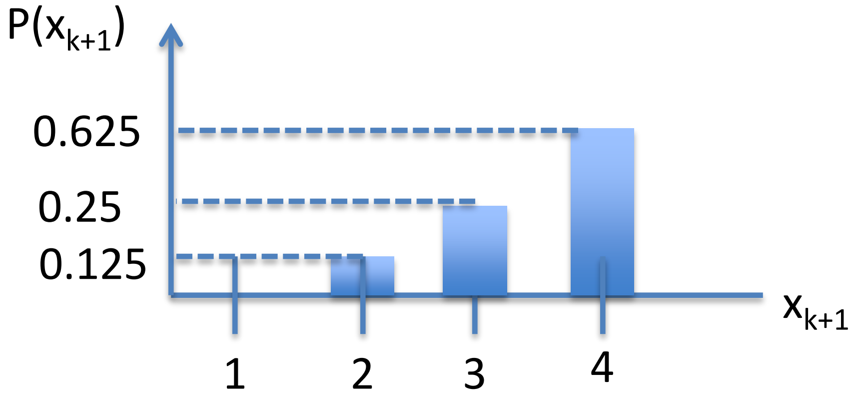
Prior $P(x_k)$



$$
\begin{aligned}
P(x_{k+1}=4) &= \sum_{x' \in X} P(x_{k+1}=4 \mid x'_k, u_k) P(x'_k) \\
&= P(x_{k+1}=4 \mid x'_k=1, u_k) P(x'_k=1) + \\
&\quad P(x_{k+1}=4 \mid x'_k=2, u_k) P(x'_k=2) + \\
&\quad P(x_{k+1}=4 \mid x'_k=3, u_k) P(x'_k=3) + \\
&\quad P(x_{k+1}=4 \mid x'_k=4, u_k) P(x'_k=4) \\
&= 0 \times 0 + 0.25 \times 0.5 + 0.75 \times 0 + 1 \times 0.5 \\
&= 0.625
\end{aligned}
$$

Propagating Motion

$$P(x_{k+1}) = \int P(x_{k+1} \mid x_k, u_k) P(x_k) dx_k$$



Discrete Bayes Filter Algorithm

Algorithm **Discrete_Bayes_filter**($u_{0:k-1}$, $y_{1:k}$, P(x_0))

P(x) = P(x_0)

for i=1:k

 for all states $x \in X$

  $$P'(x) = \sum_{x' \in X} P(x \mid u_{i-1}, x') P(x')$$ Prediction given prior dist. and command

 end for

 η=0

 for all states $x \in X$

  $$P(x) = P(y_i \mid x) P'(x)$$ Update using measurement

  $\eta = \eta + P(x)$

 end for

 for all states $x \in X$ Normalize to 1

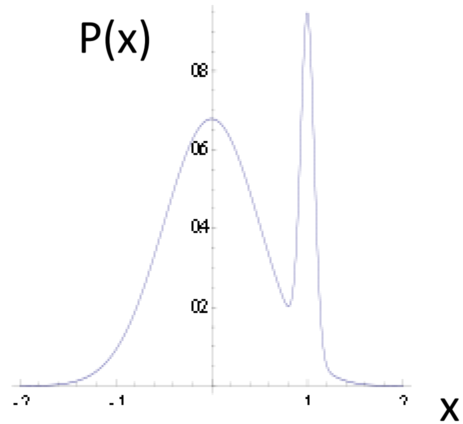
  P(x) = P(x) / η

 end for

end for

Note About the Posterior

- It is a probability distribution



- What do we do with it?
  - Maximum likelihood: $\arg\max_{x} P(z \mid x)$
  - Mean Squared Error: $E[(P(x) - P(\hat{x}))^2]$

Bayes Filter Ingredients

- Motion model

  P(x_k | $x_{k\text{-}1}$, $u_{k\text{-}1}$)
- Observation model

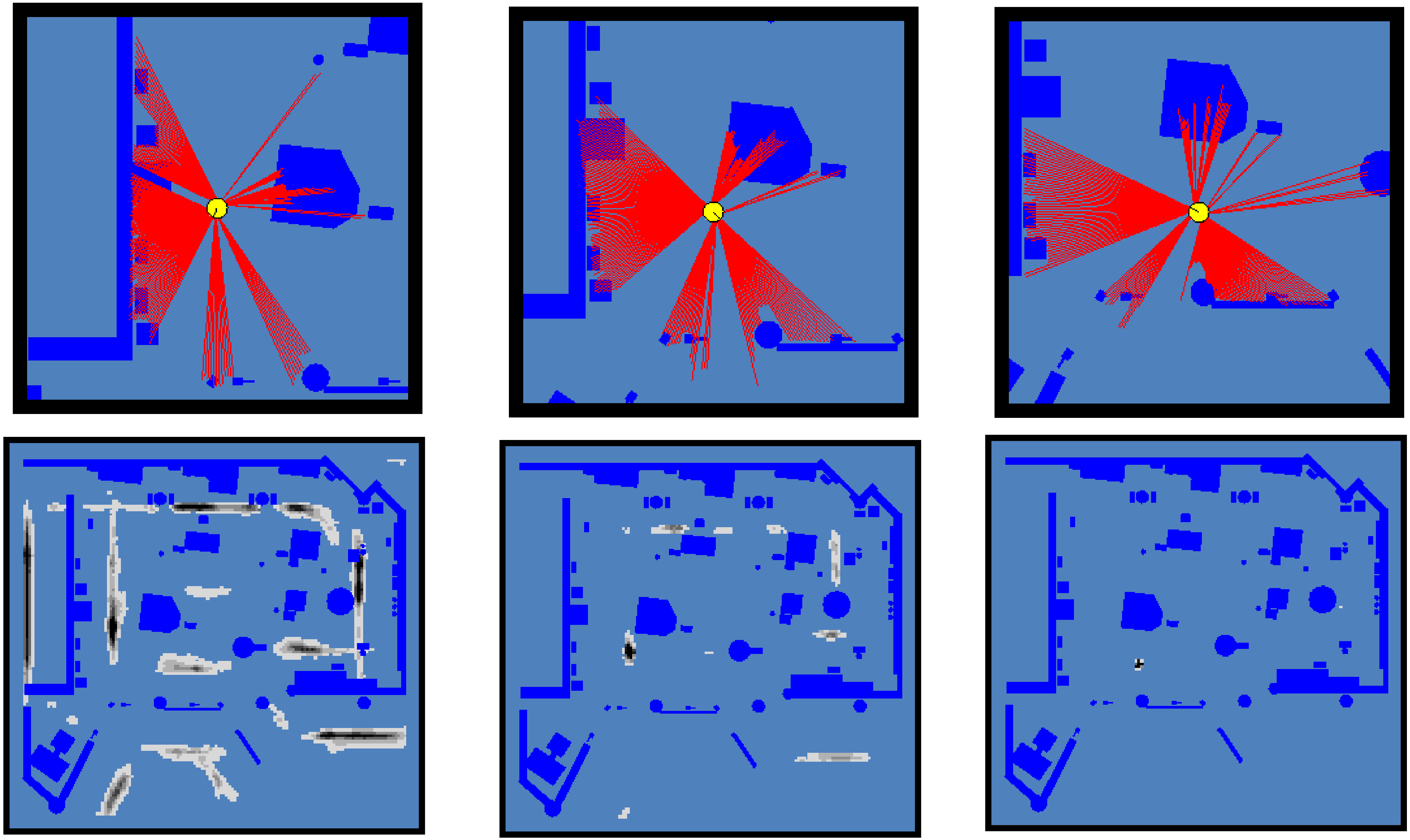
  P(y_k | x_k)
- Bayes estimator

  MAP, MSE,

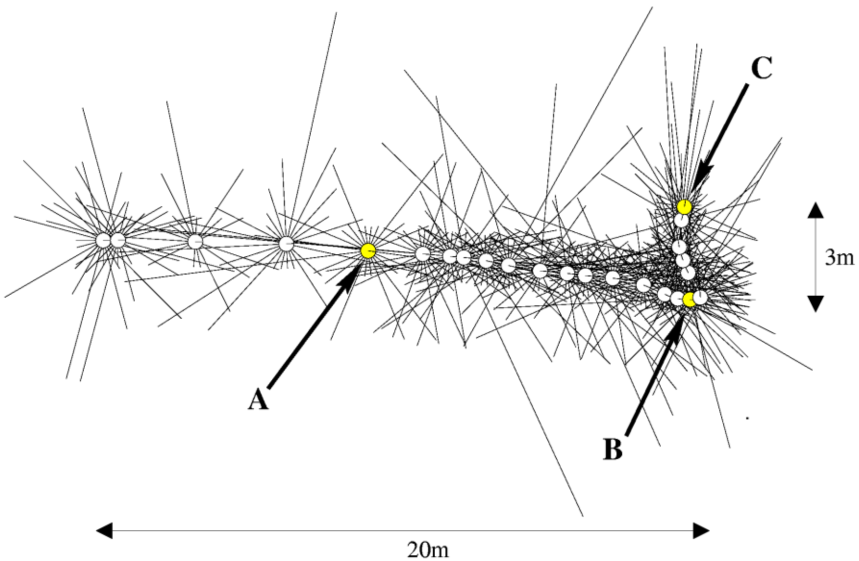
How To Get Likelihoods?

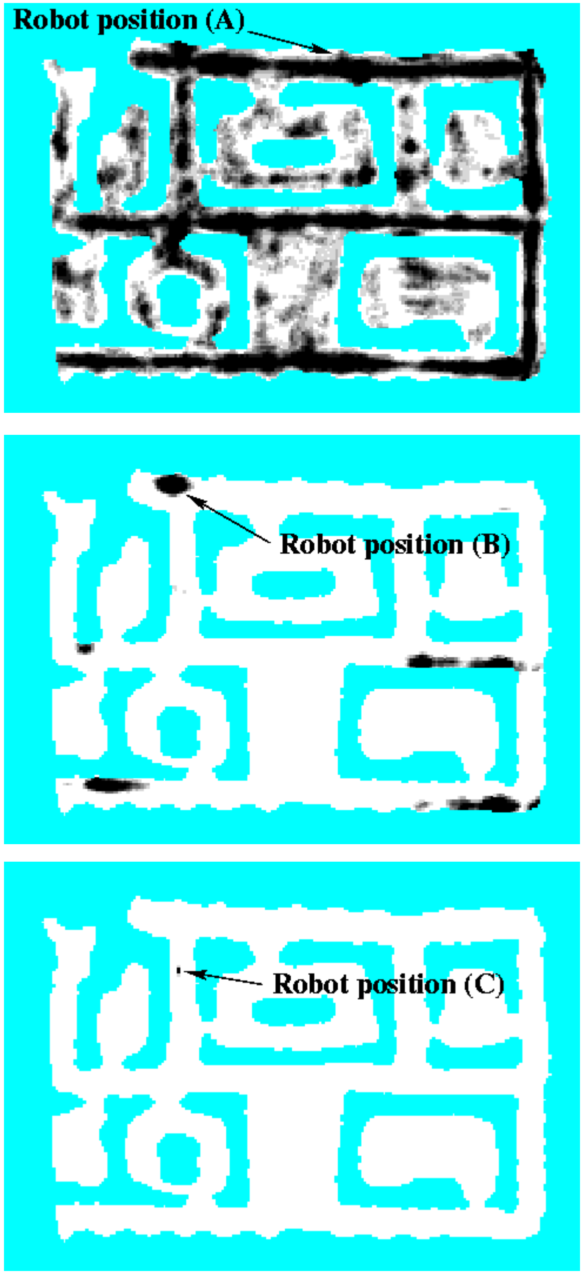
- How do we get $p(y \mid x)$?
- In the discrete case, x is a fixed value
- For a fixed value and *known* map, we can predict/simulate sensor readings y* (recall $y = h(x) + v$)
- But, we know $y^* - y \sim v$ for whatever distribution v has
  - v is Gaussian with covariance Λ, then $P(y \mid x) = G(y^*-y\,;\,0,\Lambda)$
  - v could be represented with an empirical histogram; $P(y \mid x)$ is a table lookup

Grid-based Localization



Sonars and Occupancy Grid Map





Localization Algorithms - Comparison

	Kalman filter	Multi-hypothesis tracking	Grid-based (fixed/variable)		
Sensors	Gaussian	Gaussian	Non-Gaussian		
Posterior	Gaussian	Multi-modal	Piecewise constant		
Efficiency (memory)	++	++	-/+		
Efficiency (time)	++	++	o/+		
Implementation	+	o	+/o		
Accuracy	++	++	+/++		
Robustness	-	+	++		
Global localization	No	Yes	Yes		

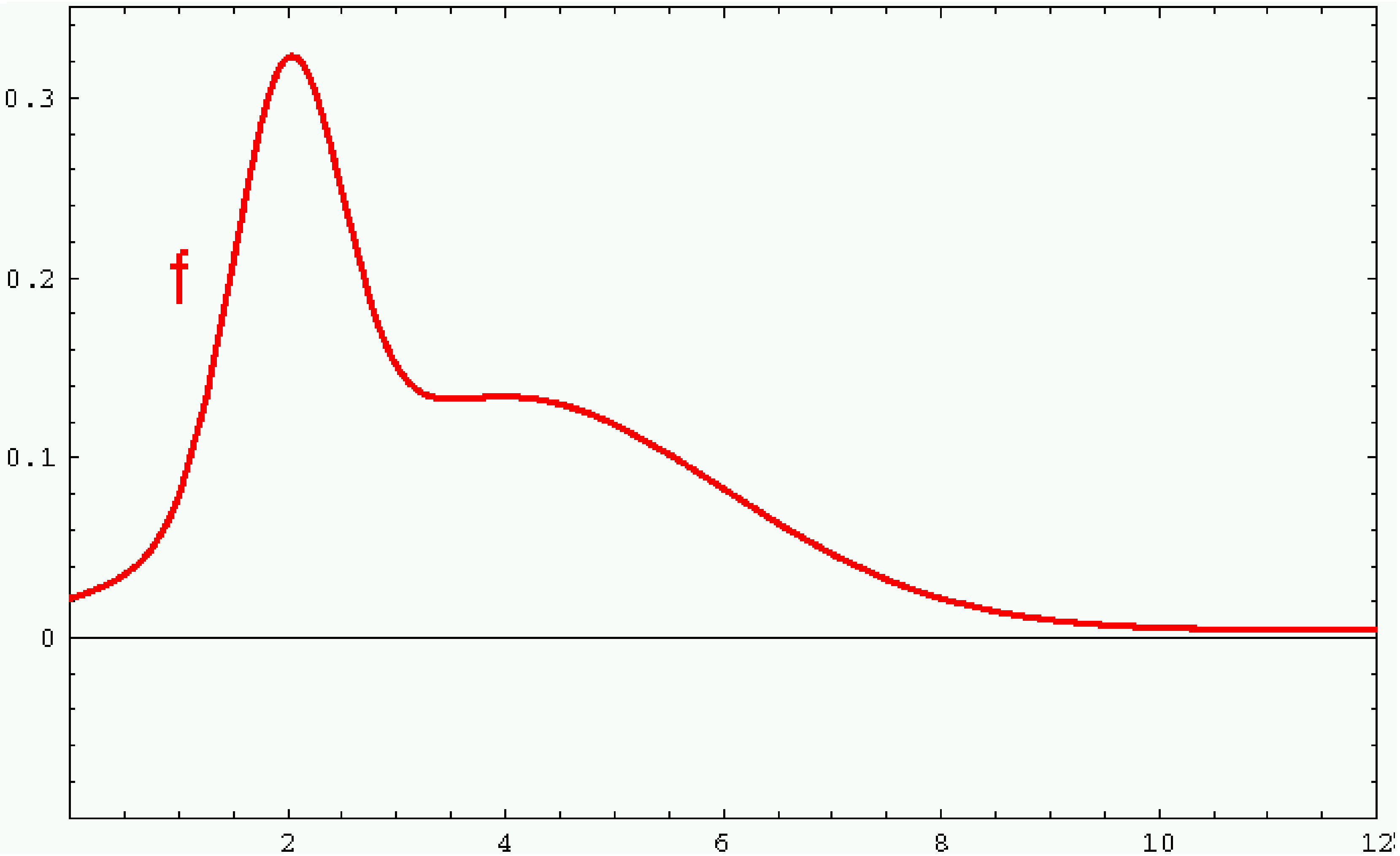
Particle Filters

- 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]d

Sample-based Density Representation



Particle Filter Algorithm

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

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

draw x^i_t from $p(x_t \mid x^i_{t-1}, u_{t-1})$

Importance factor for x^i_t:

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

Monte Carlo Localization

- Monte-Carlo-Localization(a, z, N, map)
  - S = N samples from P(X(t)) from previous call
  - for i = 1 to N
    - S[i] = sample from P(X(t+1) | X(t) = S[i], A = a)
    - W[i] = 1
    - for j = 1 to M do
      - z* = expected-sensor-reading(j,S[i],map)
      - W[i] = W[i] * P(Z = z(j) | Z* = Z*)
  - S = weighted-sample-with-replacement(N, S, W)
  - return S

- Note that S is a discrete representation of the probability of robot location

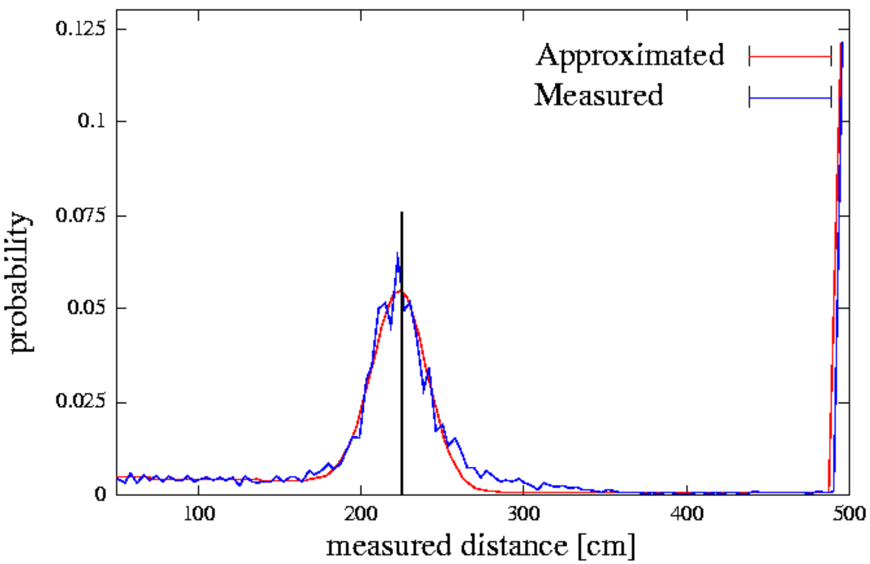
The Likehihood Function

- Generating the sensor likelihood is essentially a sensor simulation
  - can be expensive
  - pre-compute
  - approximate

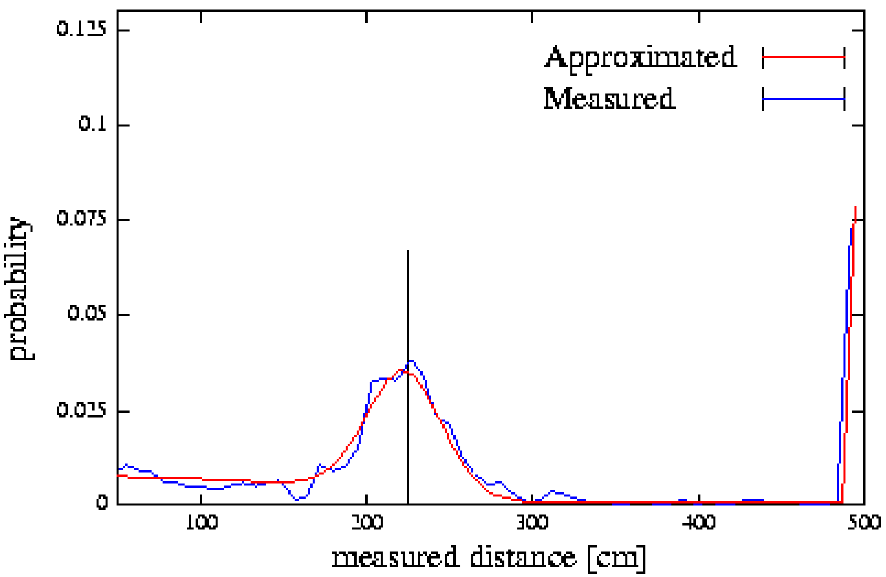
- A good fast approximation is often a weighted sum of
  - a nominal model that is fast to compute
  - other deviations that are modeled as random elements

Proximity Sensor Model

Tuned model that takes into account normal reflection, unexpected returns and randomness and out of range



Laser sensor



Sonar sensor

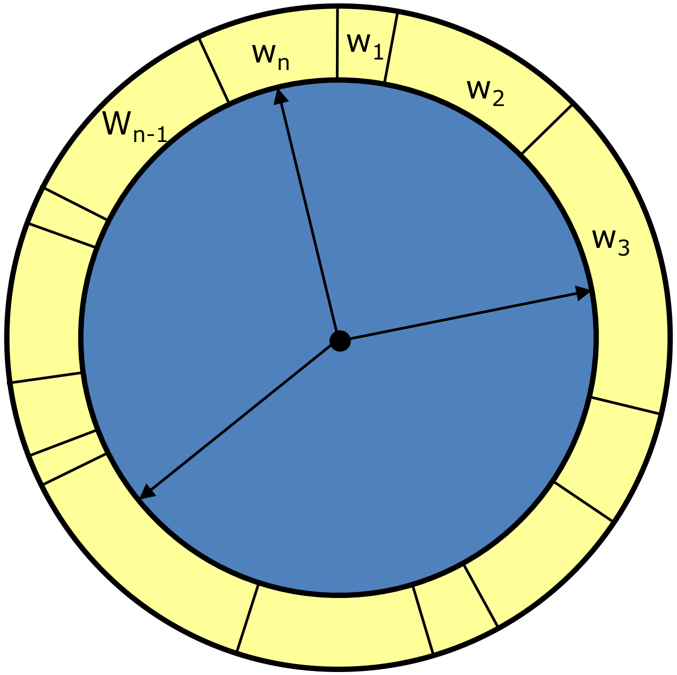
Resampling

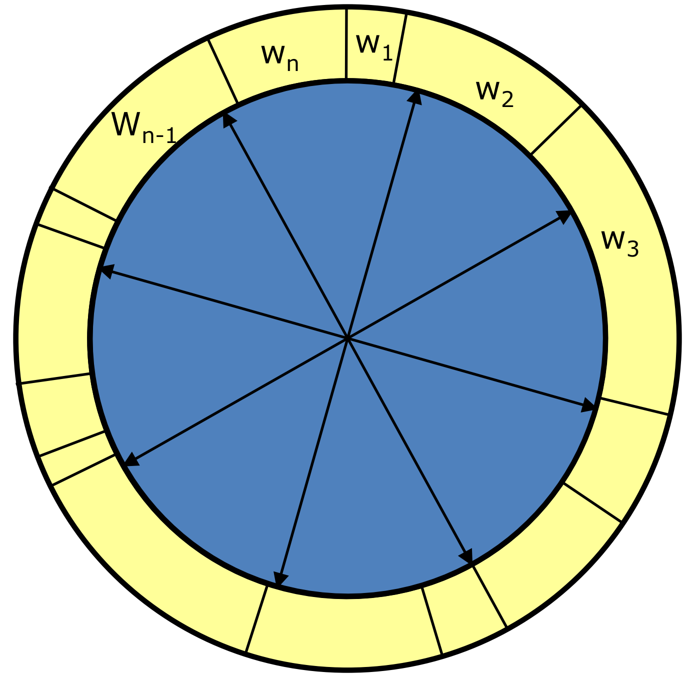
- **Given**: Set *S* of weighted samples.

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

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

Resampling





- Roulette wheel
- Binary search, log n

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

Resampling Algorithm

1. Algorithm **systematic_resampling**(*S*,*n*):

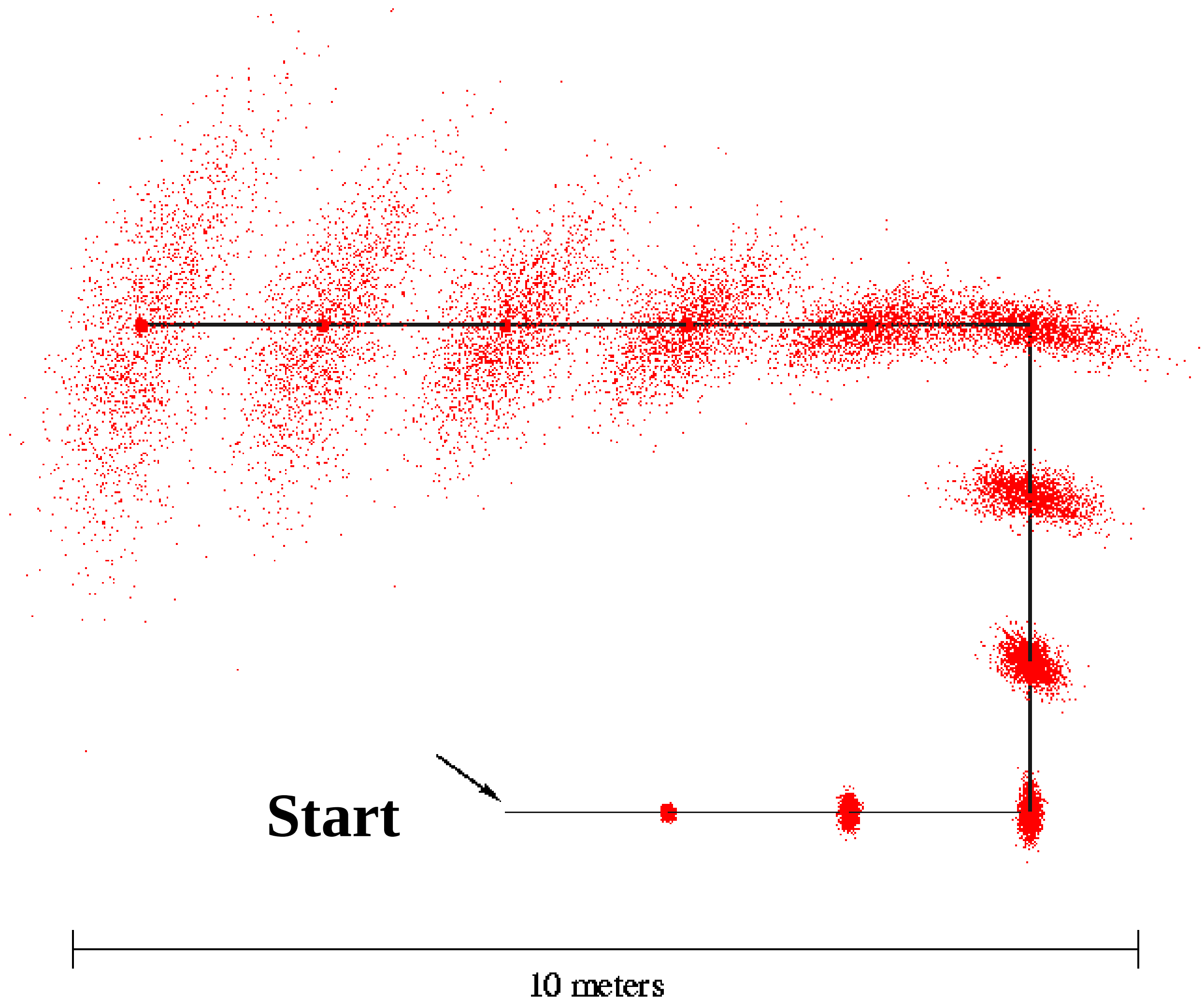
2. $S' = \emptyset, c_1 = w^1$
3. **For** $i = 2 \ldots n$ ***Generate cdf***
4. $\quad c_i = c_{i-1} + w^i$
5. $u_1 \sim U[0, 1/n], i = 1$ ***Initialize threshold***

6. **For** $j = 1 \ldots n$ ***Draw samples …***
7. $\quad$ **While** ($u_j > c_i$) ***Skip until next threshold reached***
8. $\quad\quad i = i + 1$
9. $\quad S' = S' \cup \left\{ < x^i, 1/n > \right\}$ ***Insert***
10. $\quad u_j = u_j + 1/n$ ***Increment threshold***

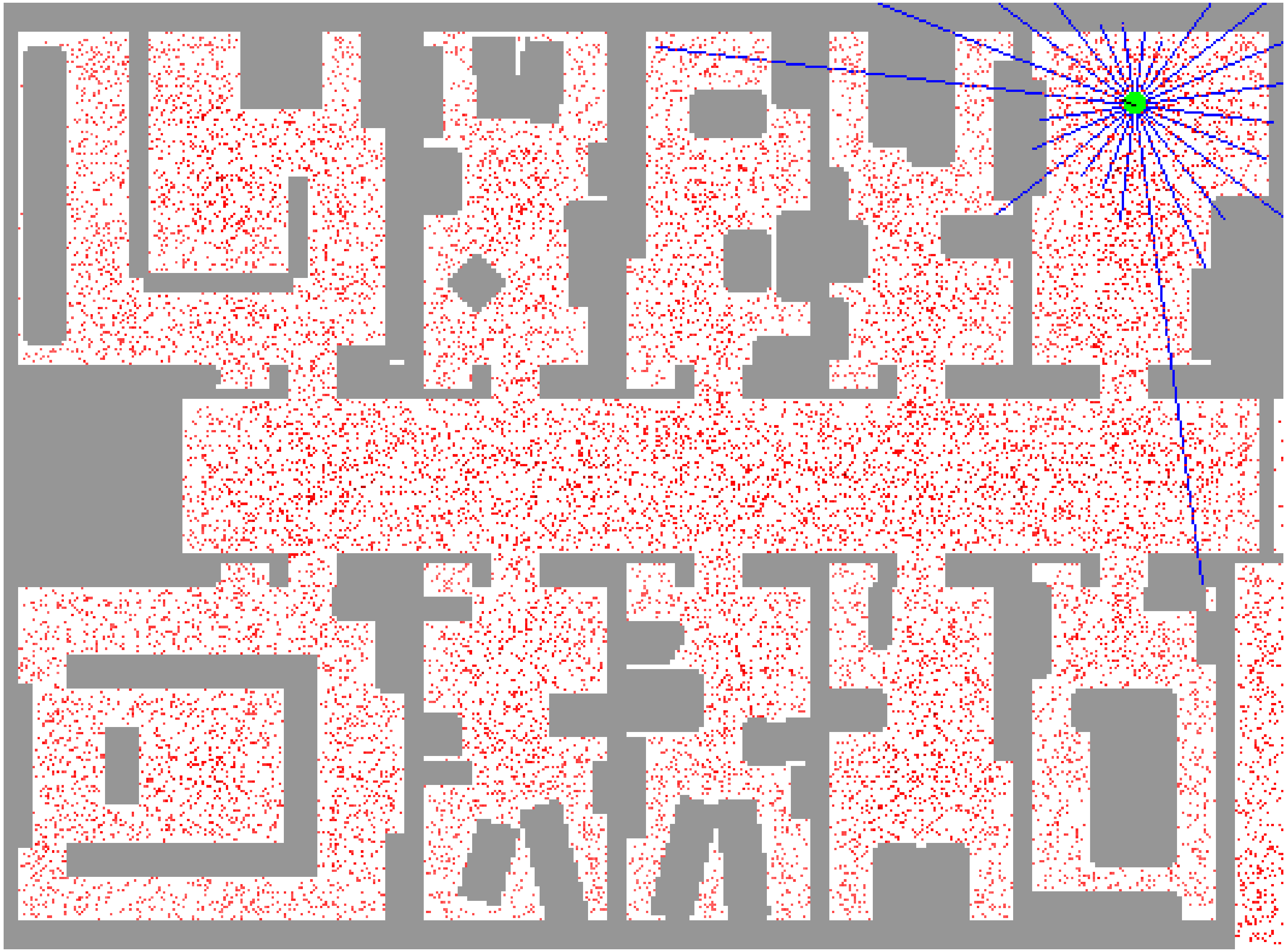
11. **Return** *S'*

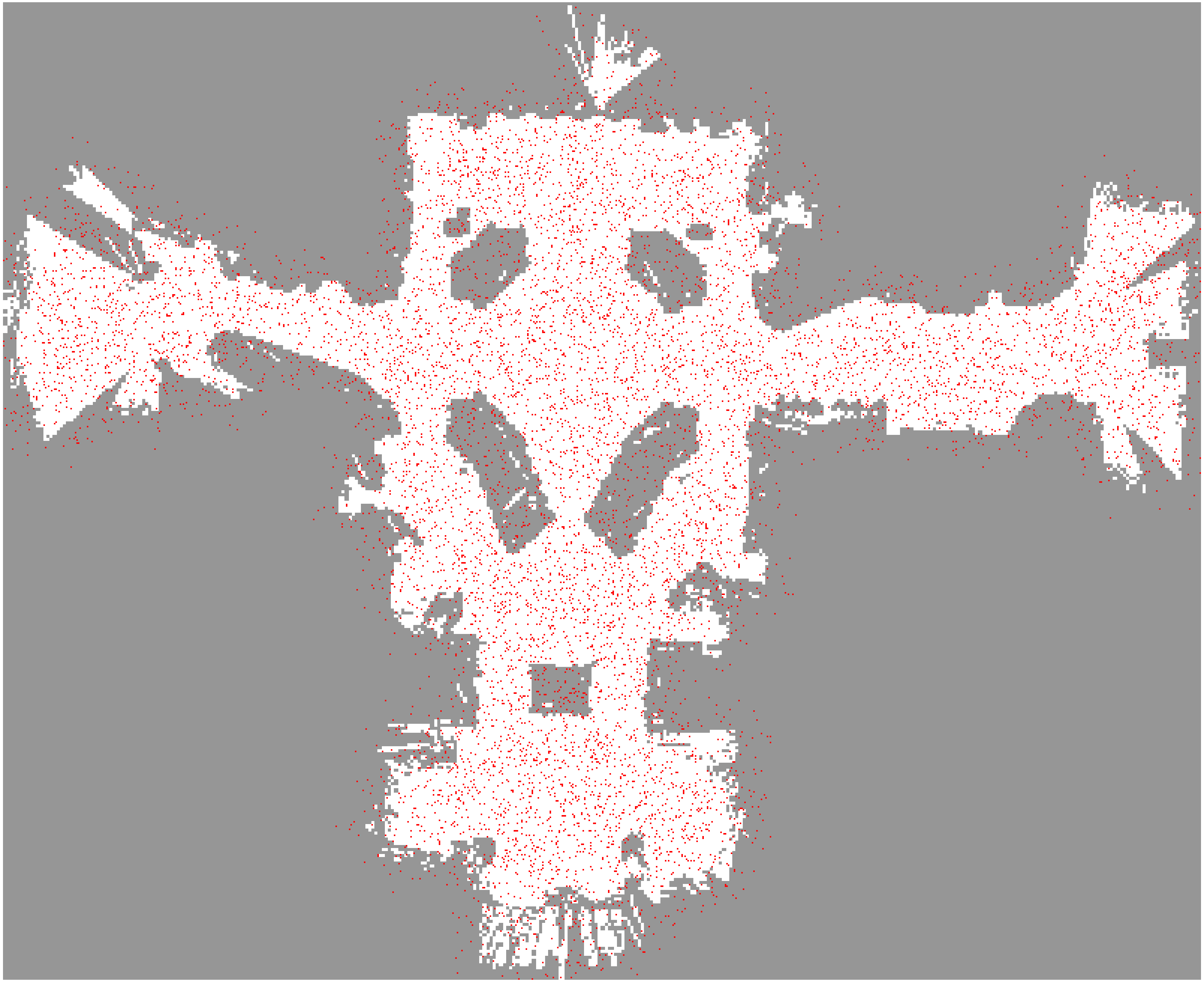
Also called **stochastic universal sampling**

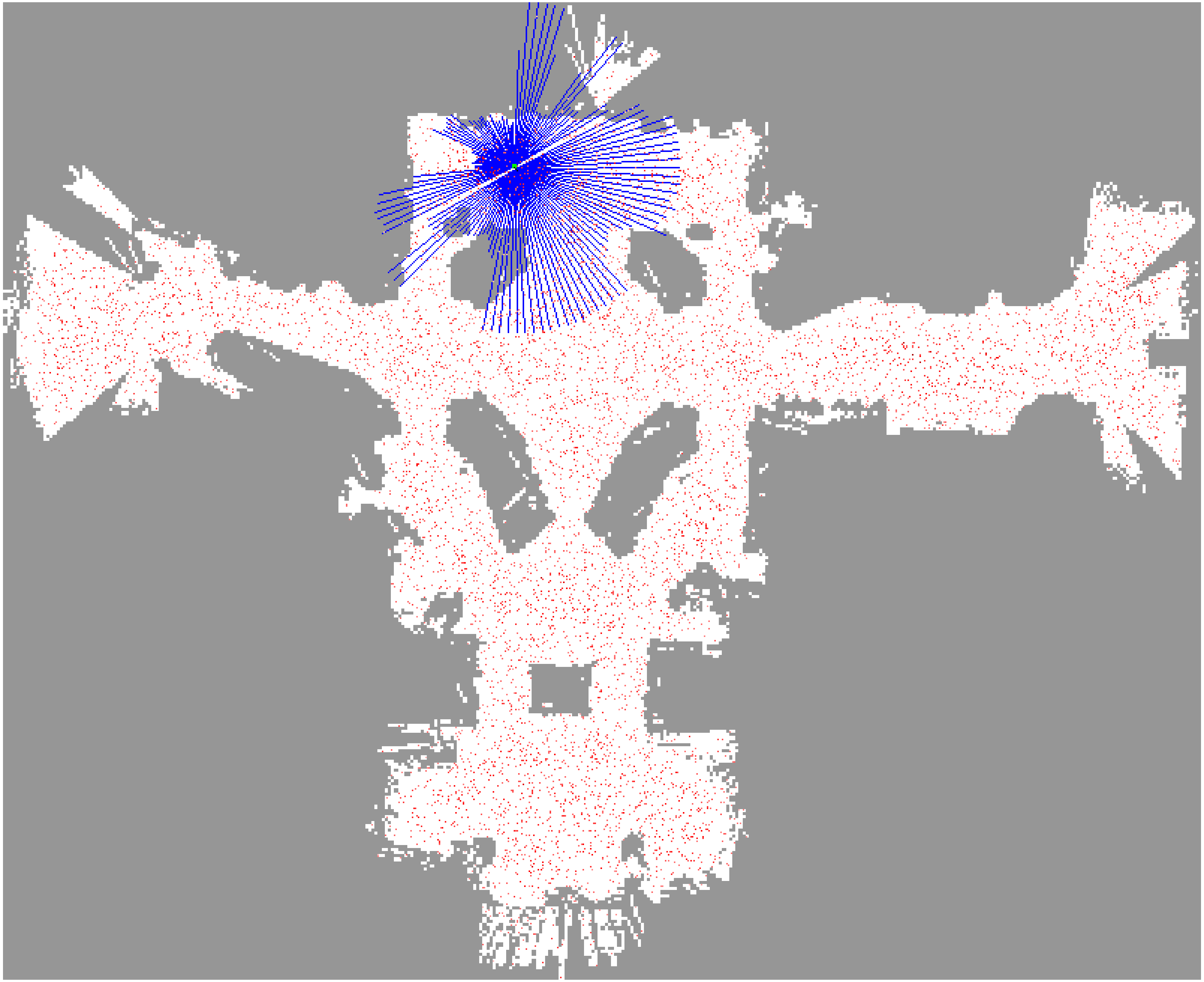
Motion Model Reminder

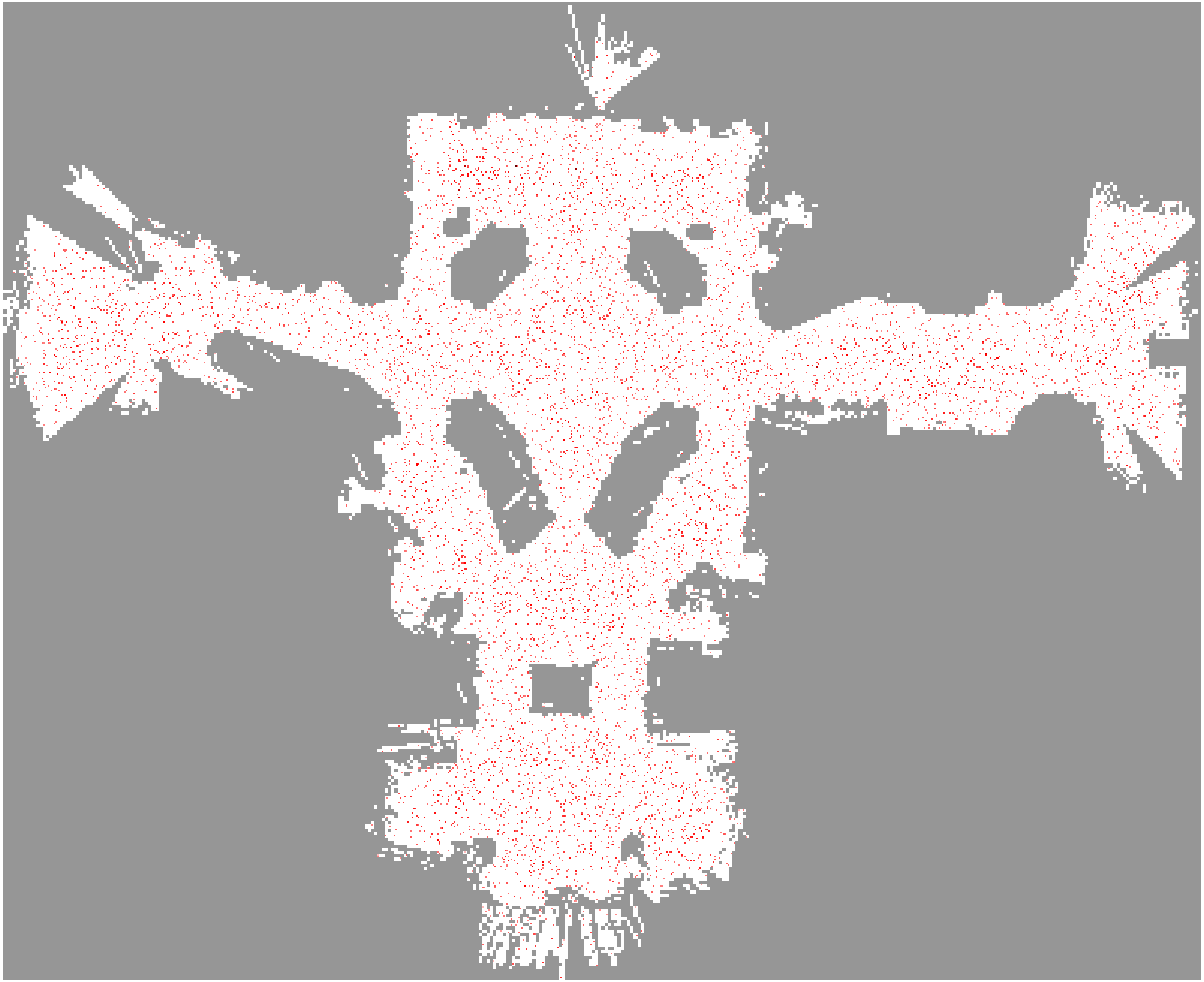


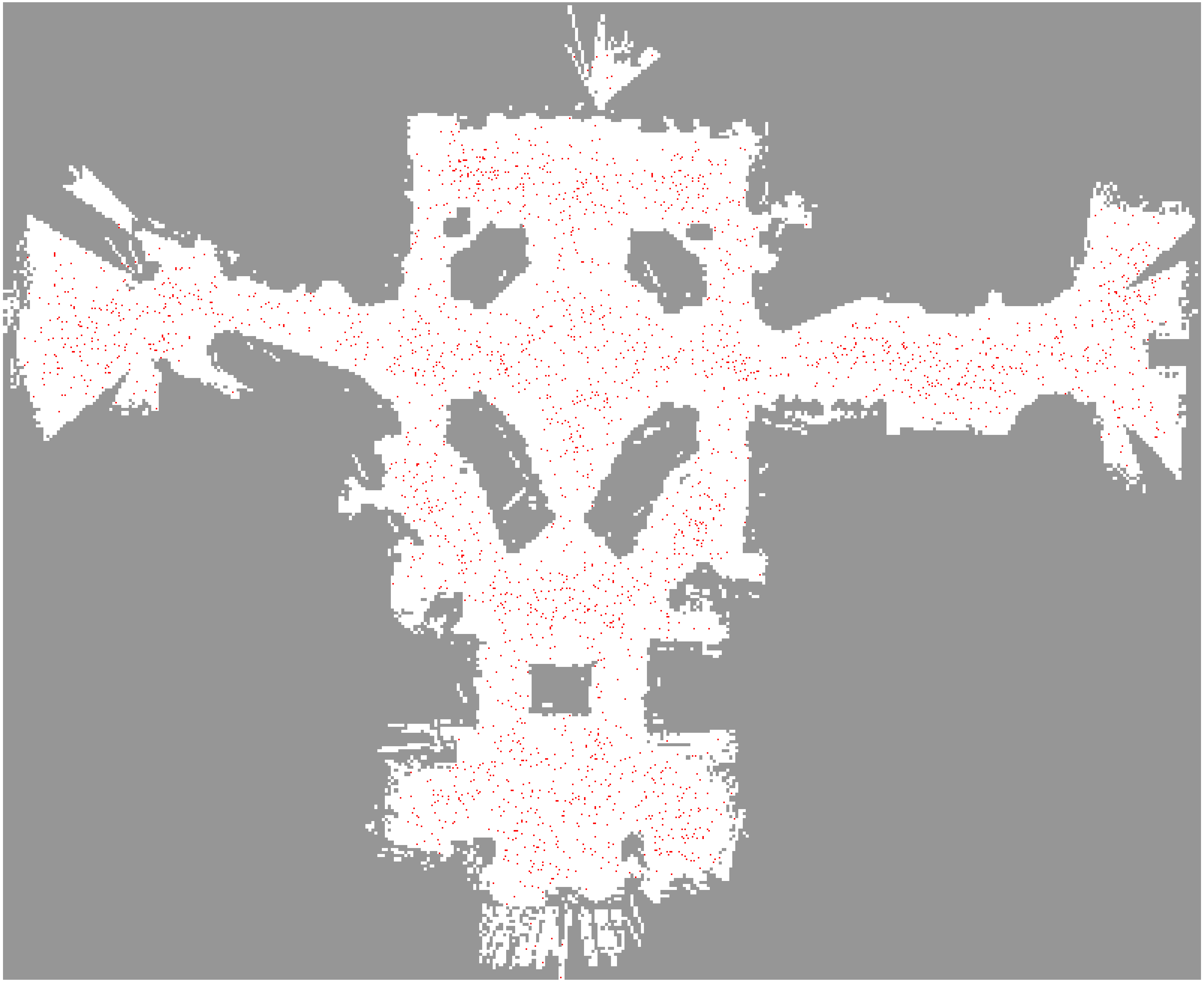
Sample-based Localization (sonar)

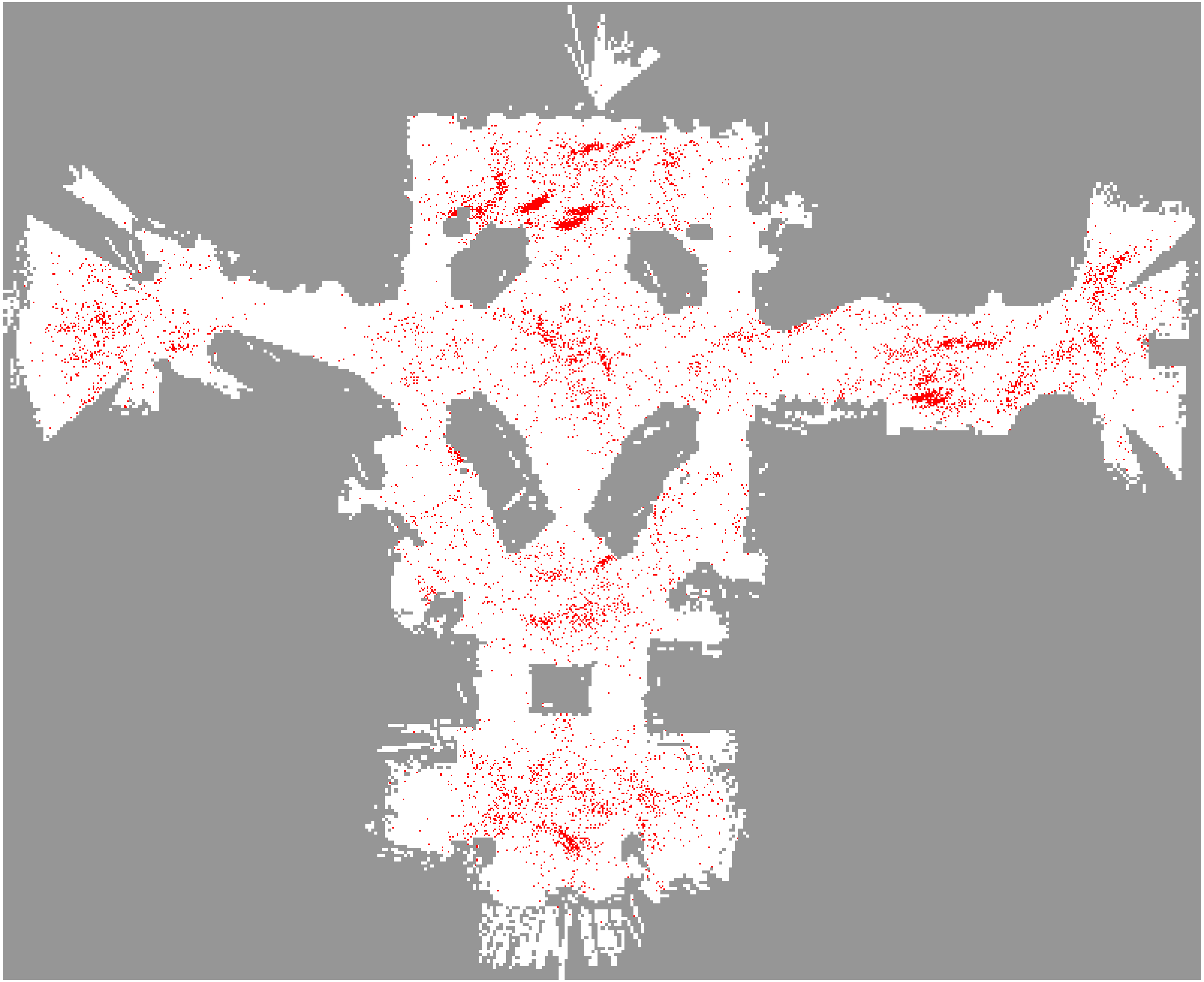


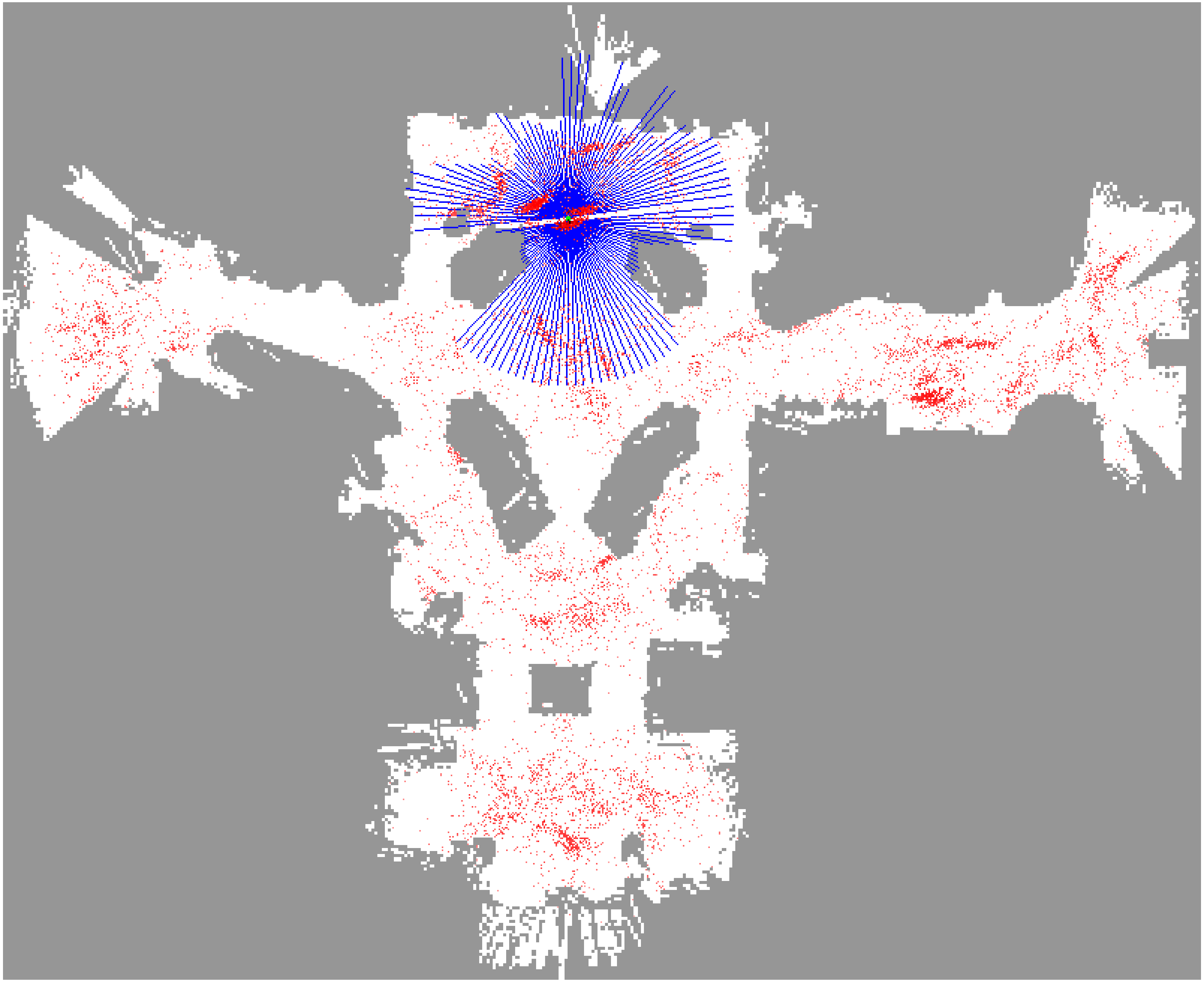


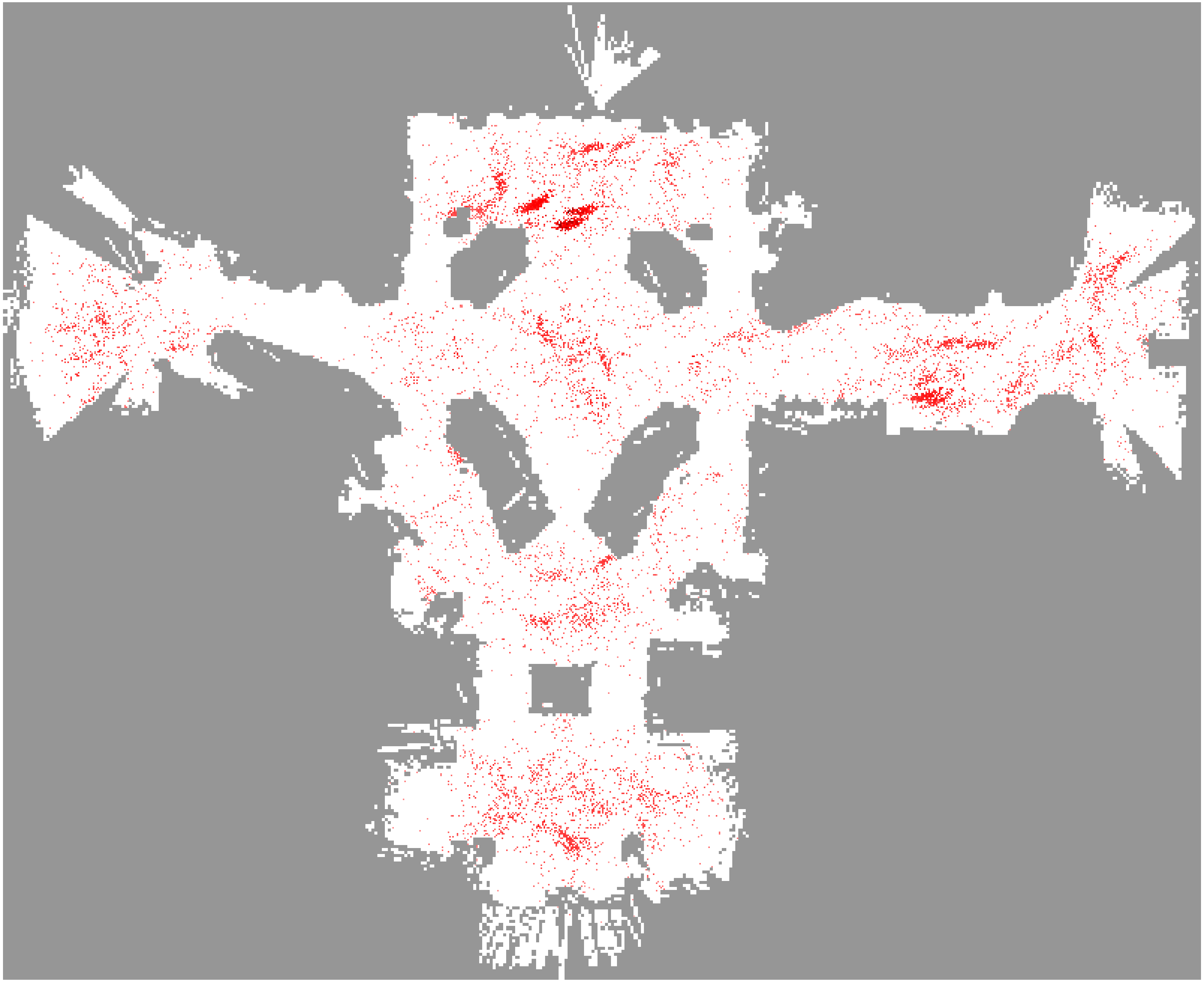


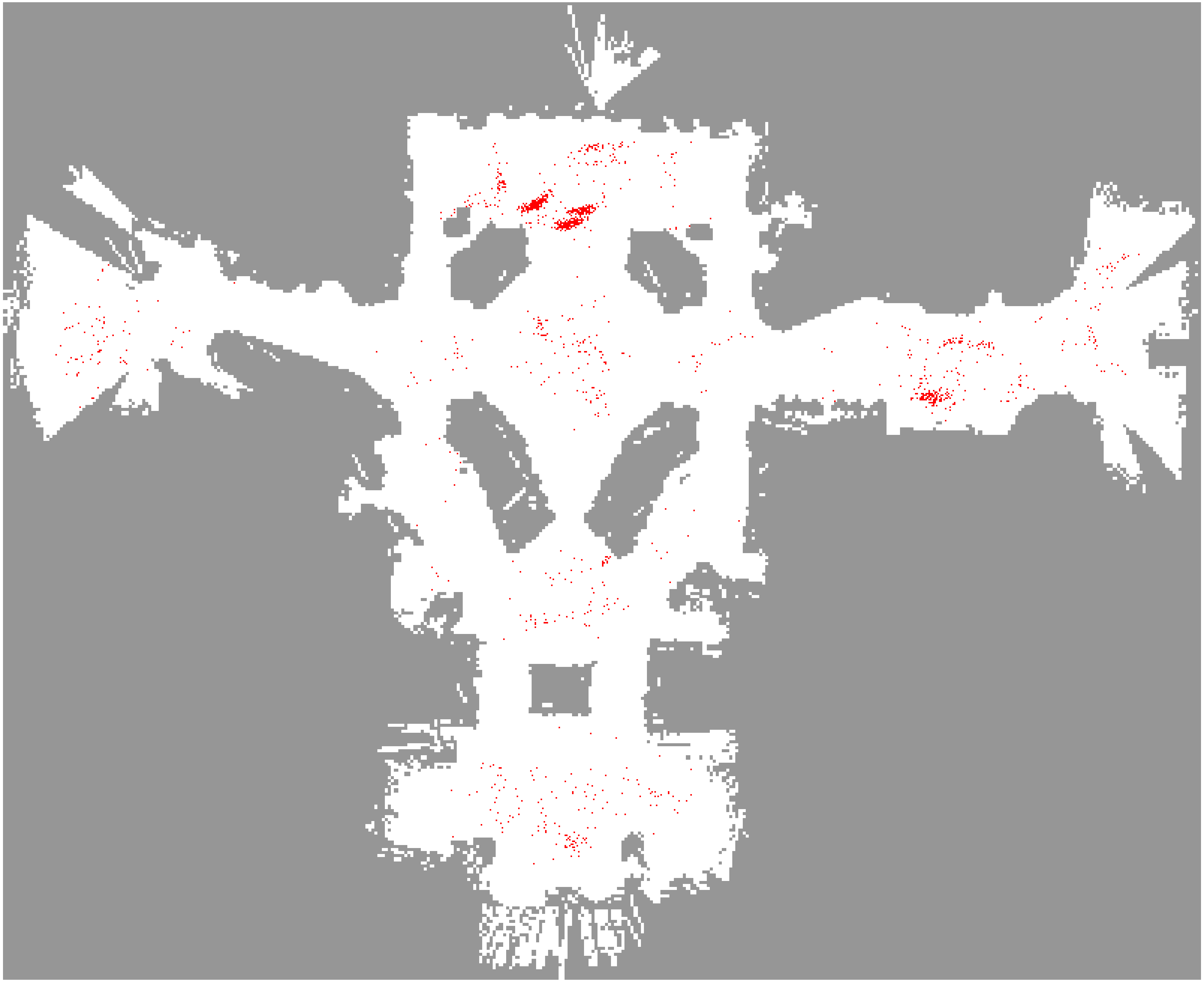


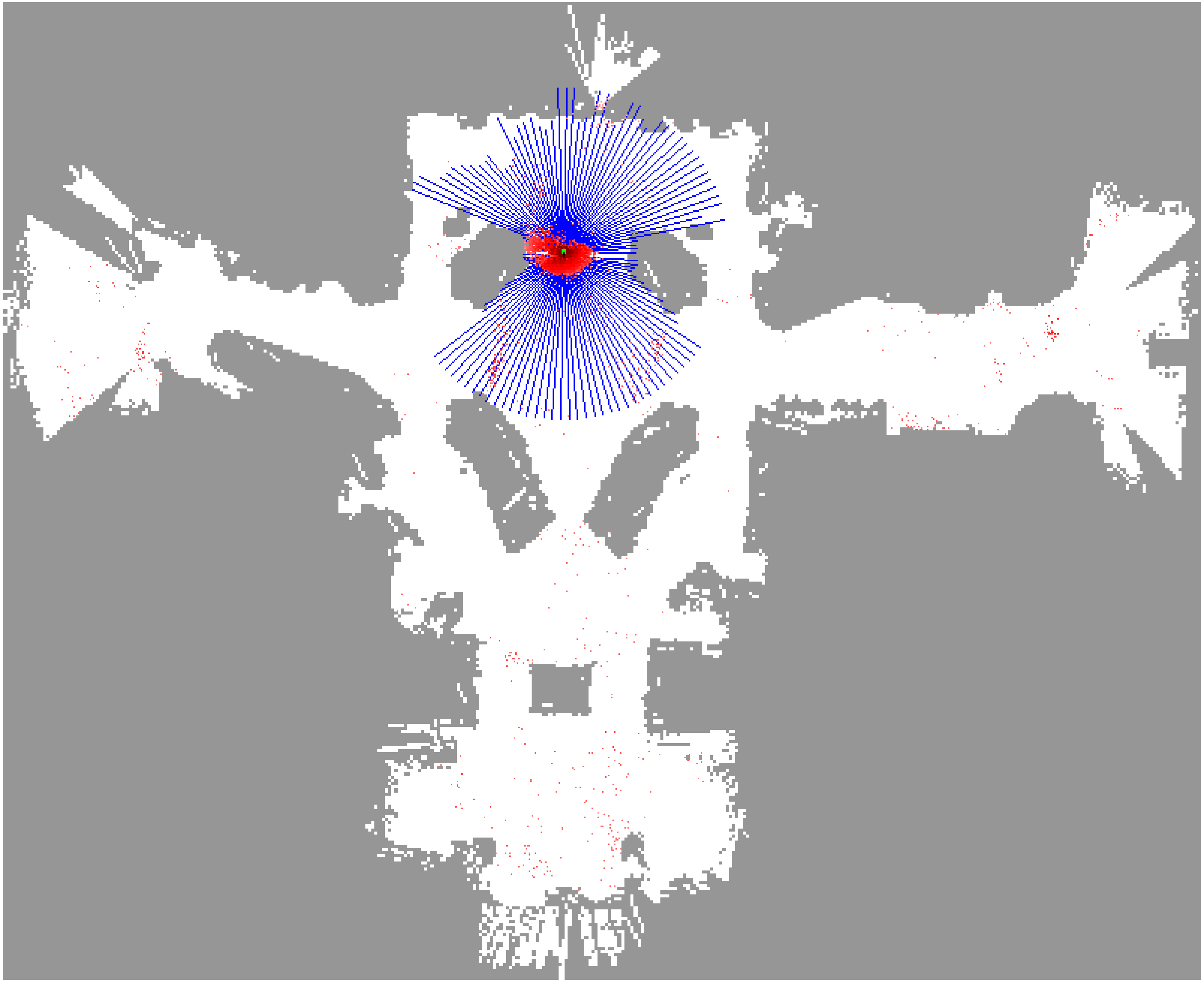


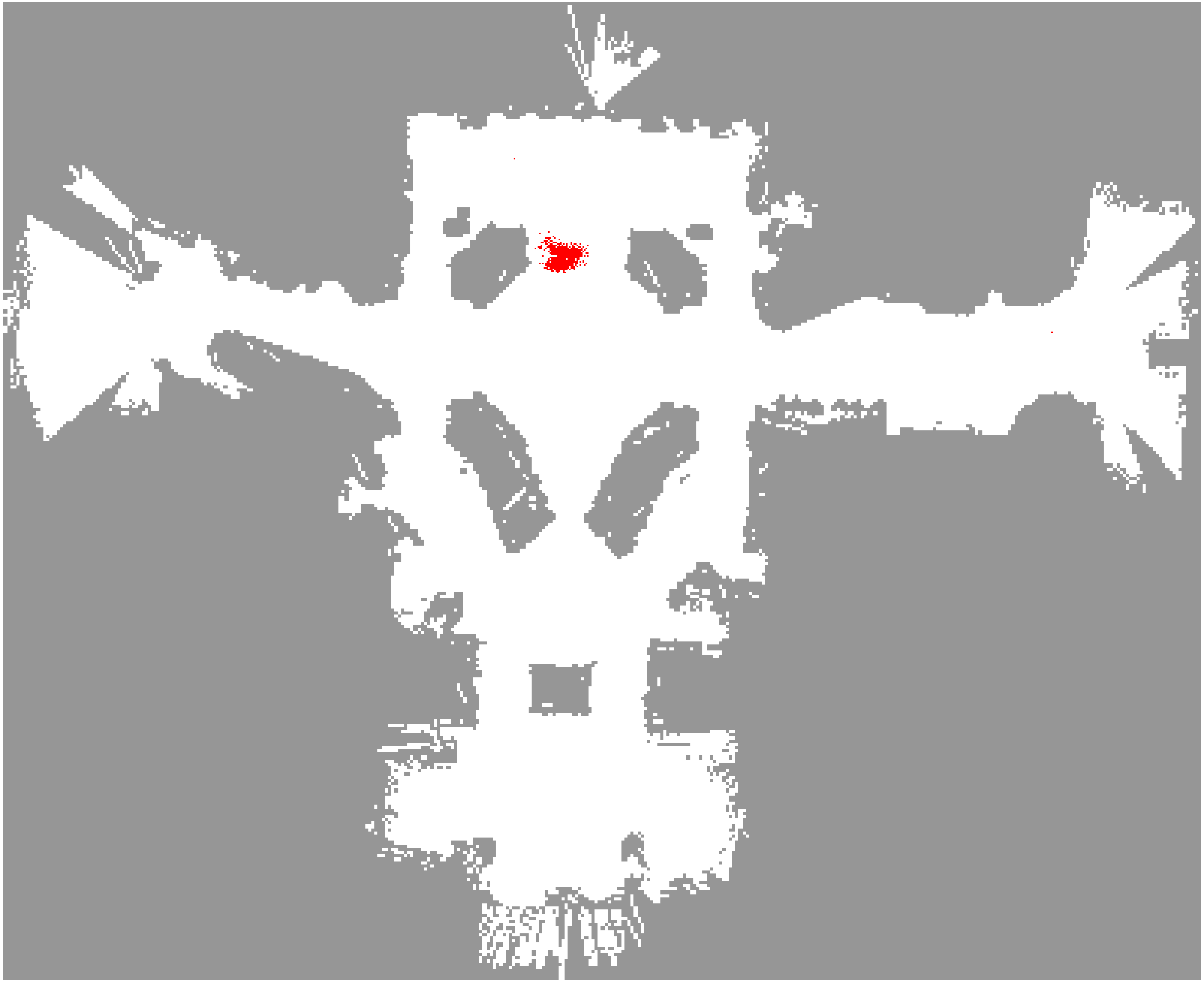


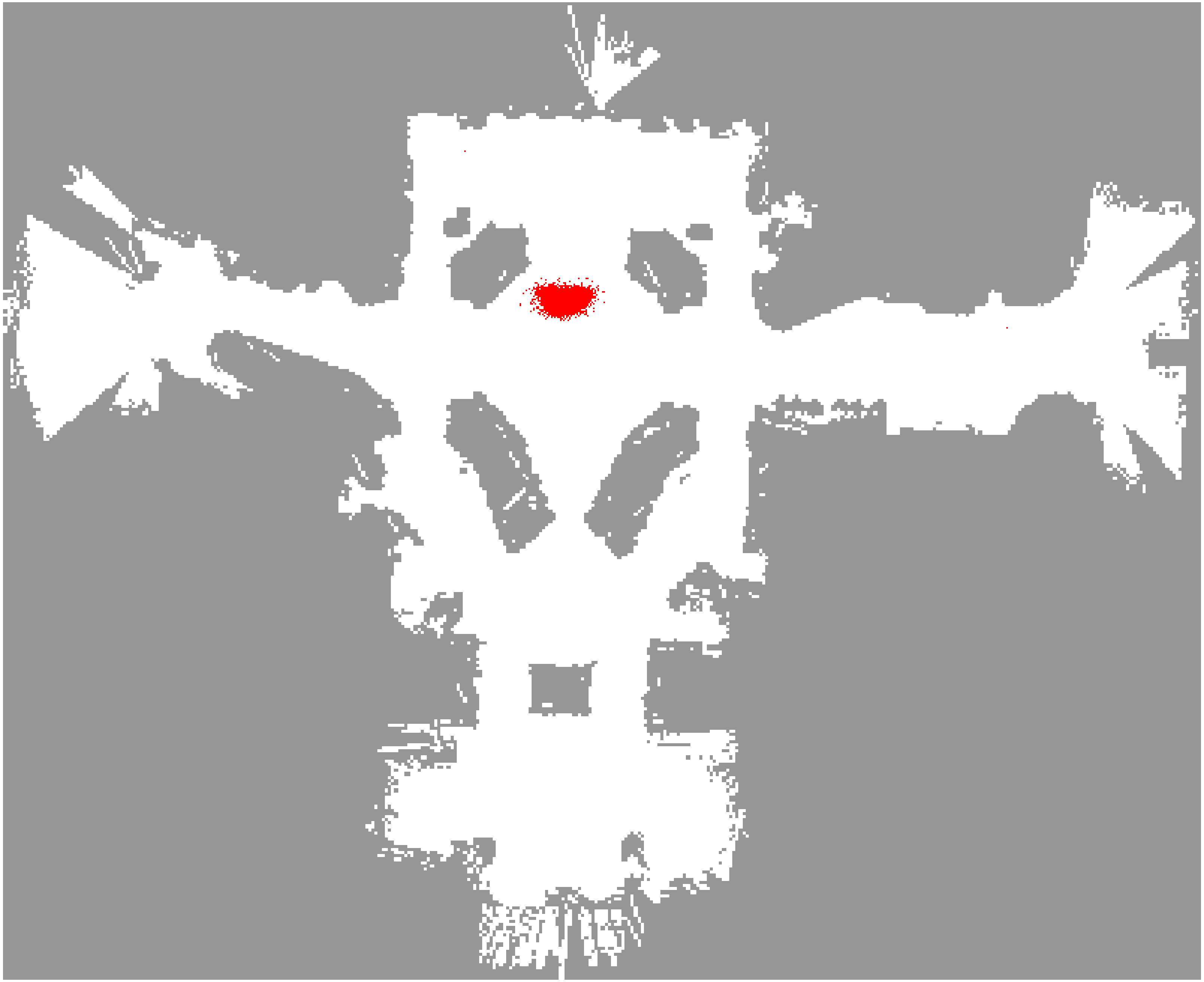


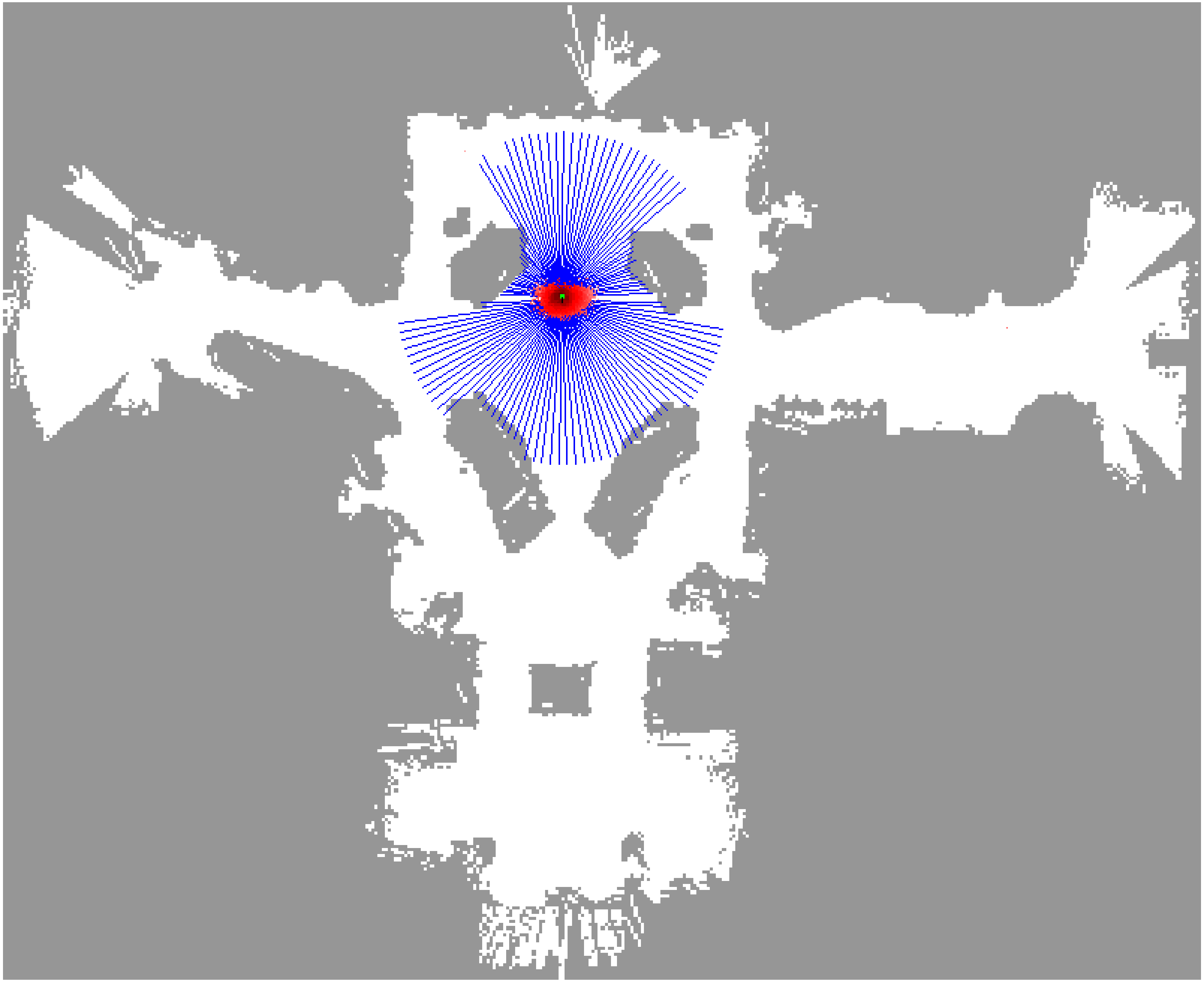


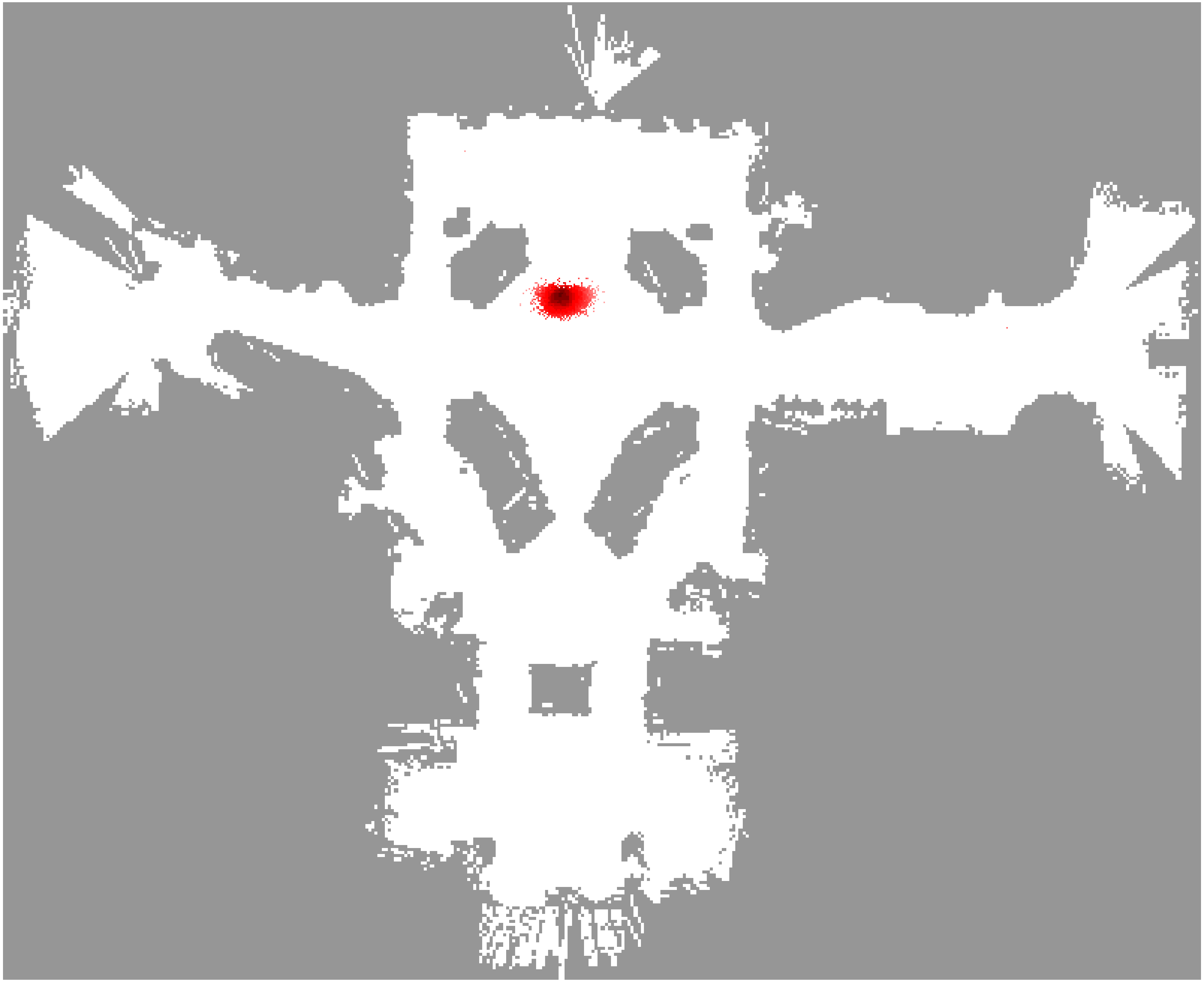


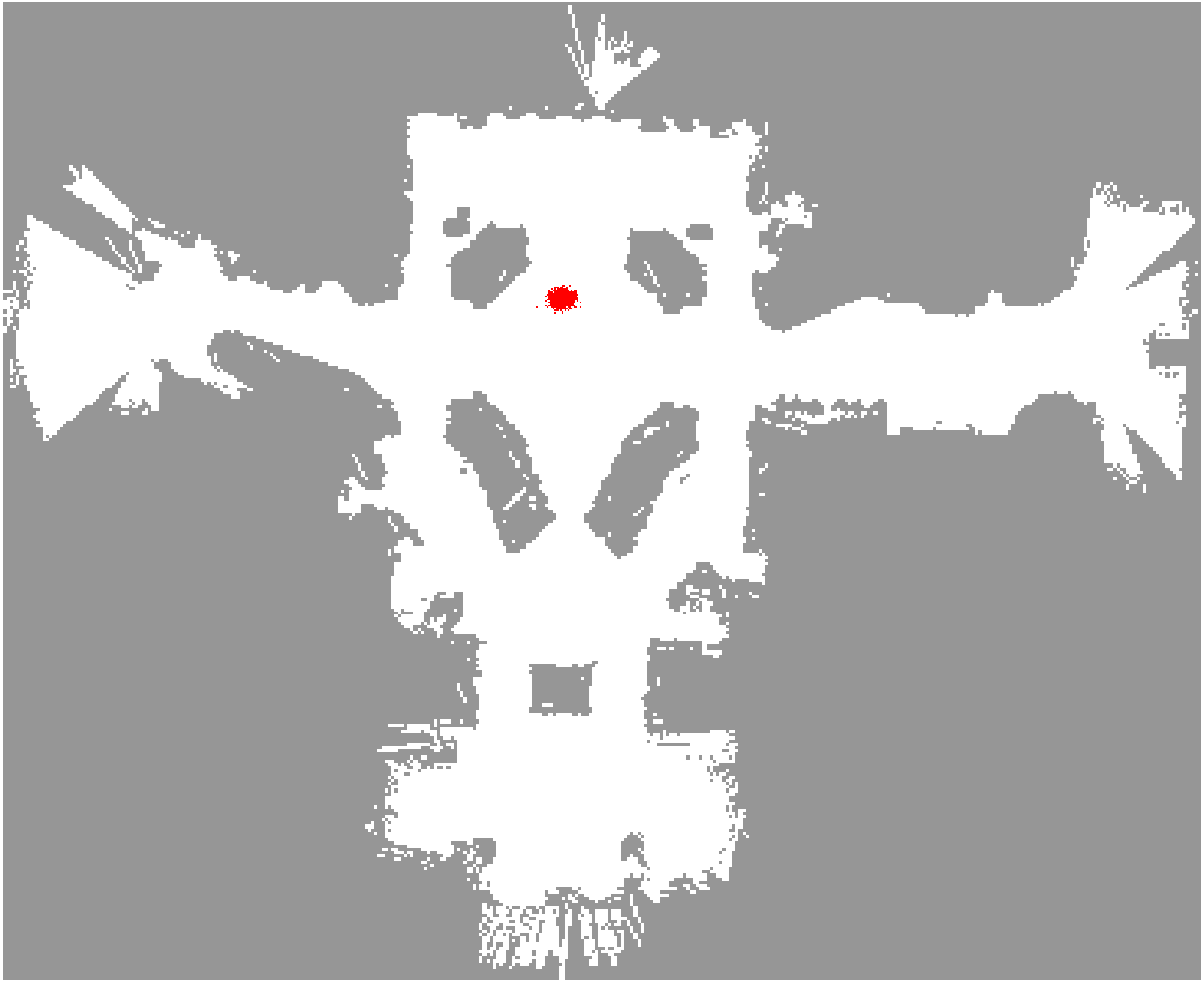


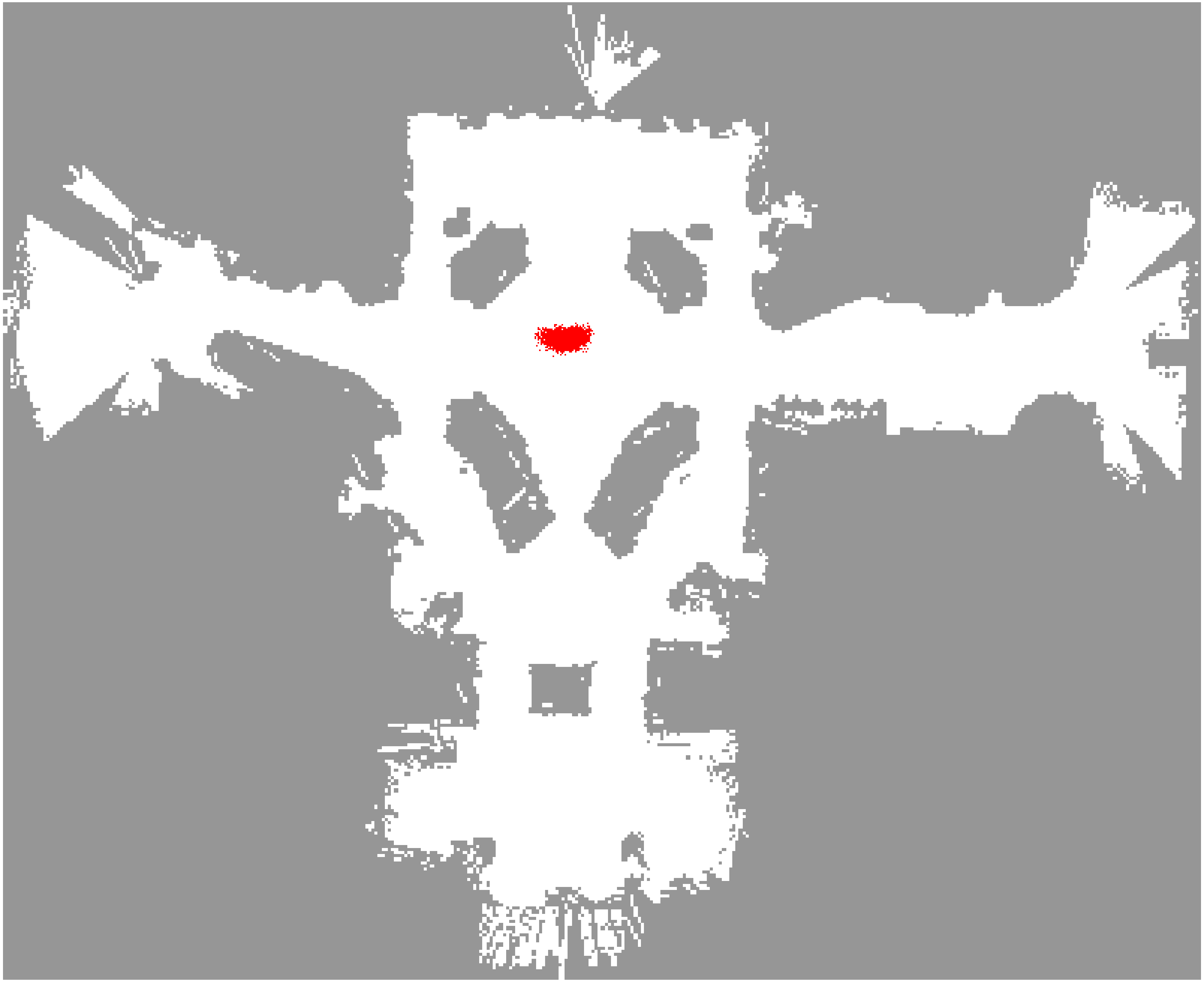


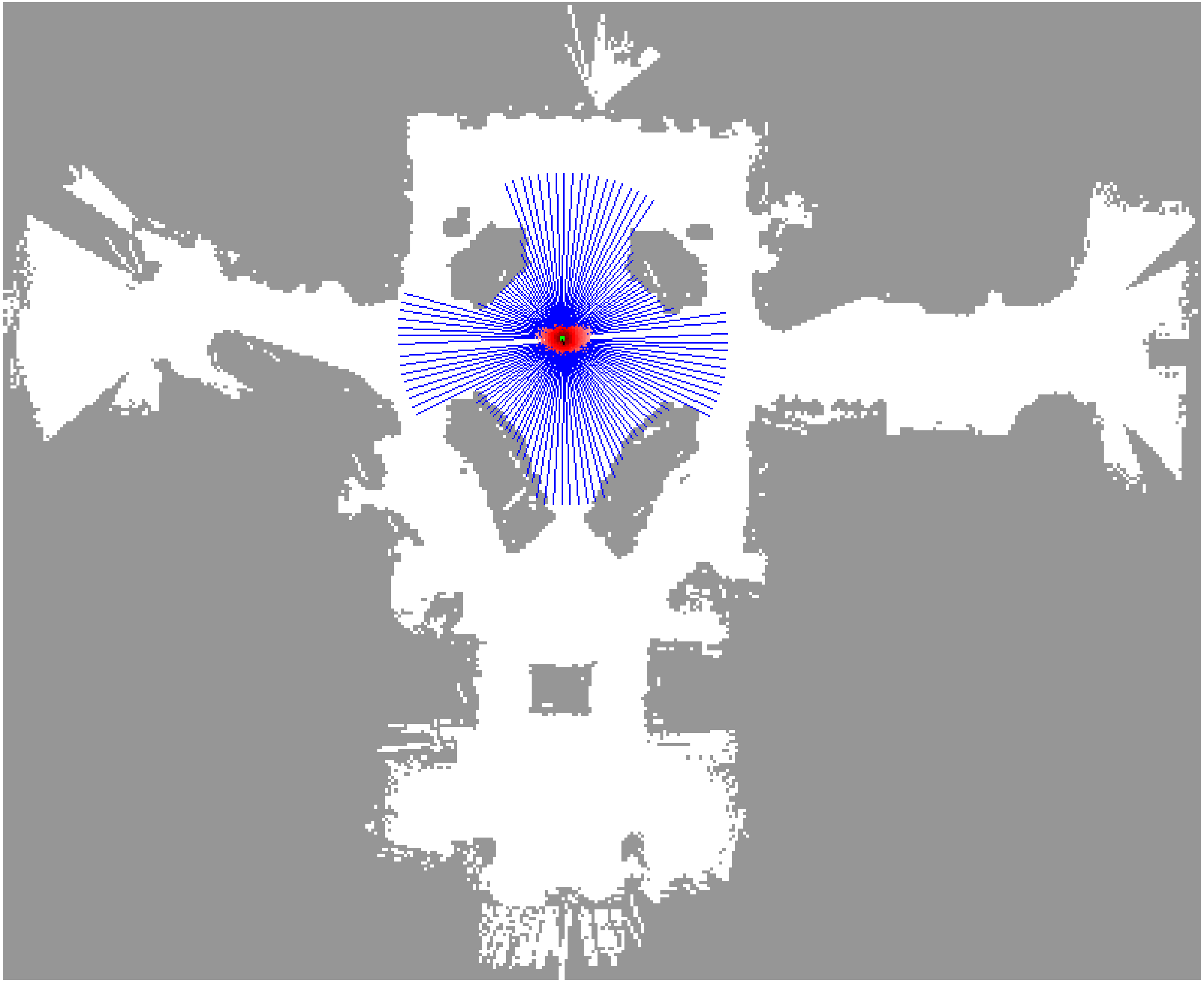


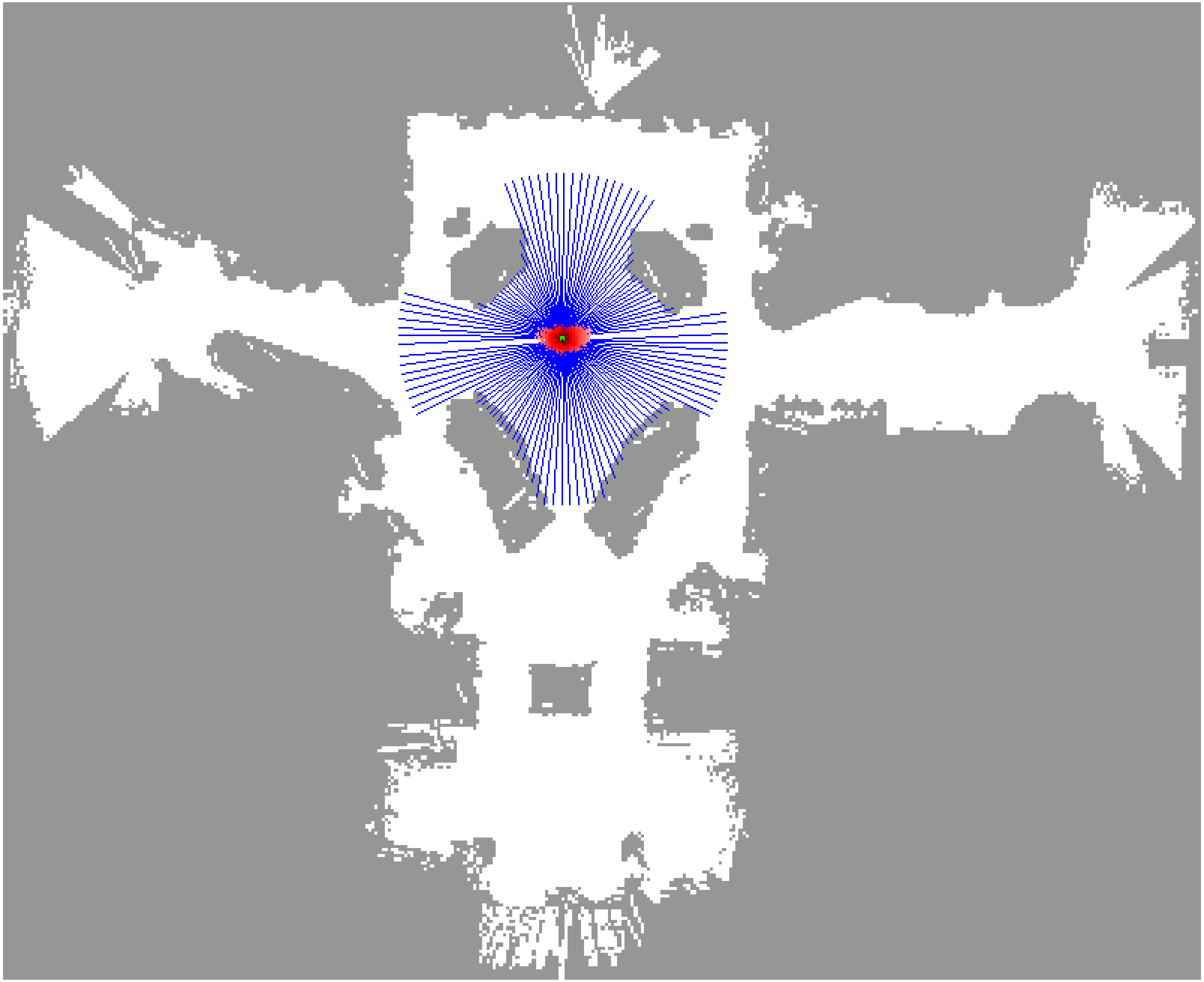






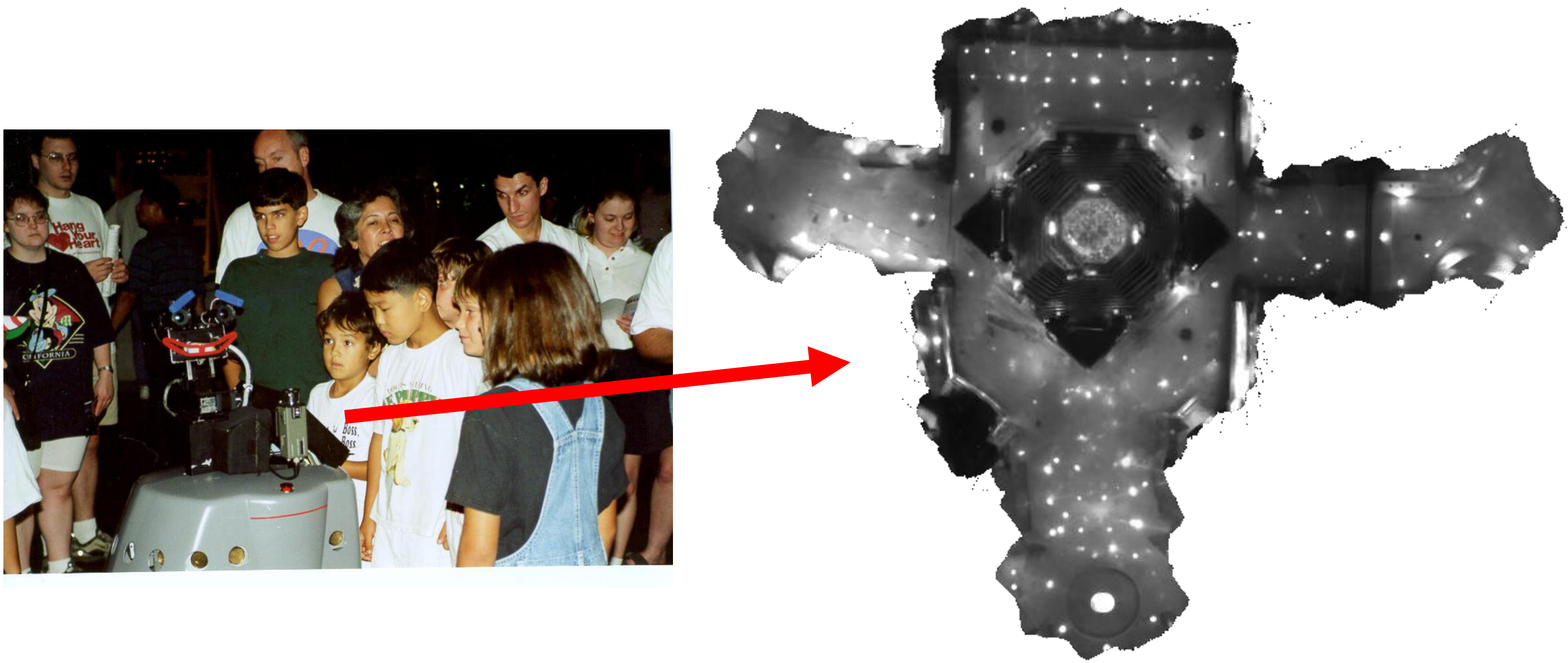






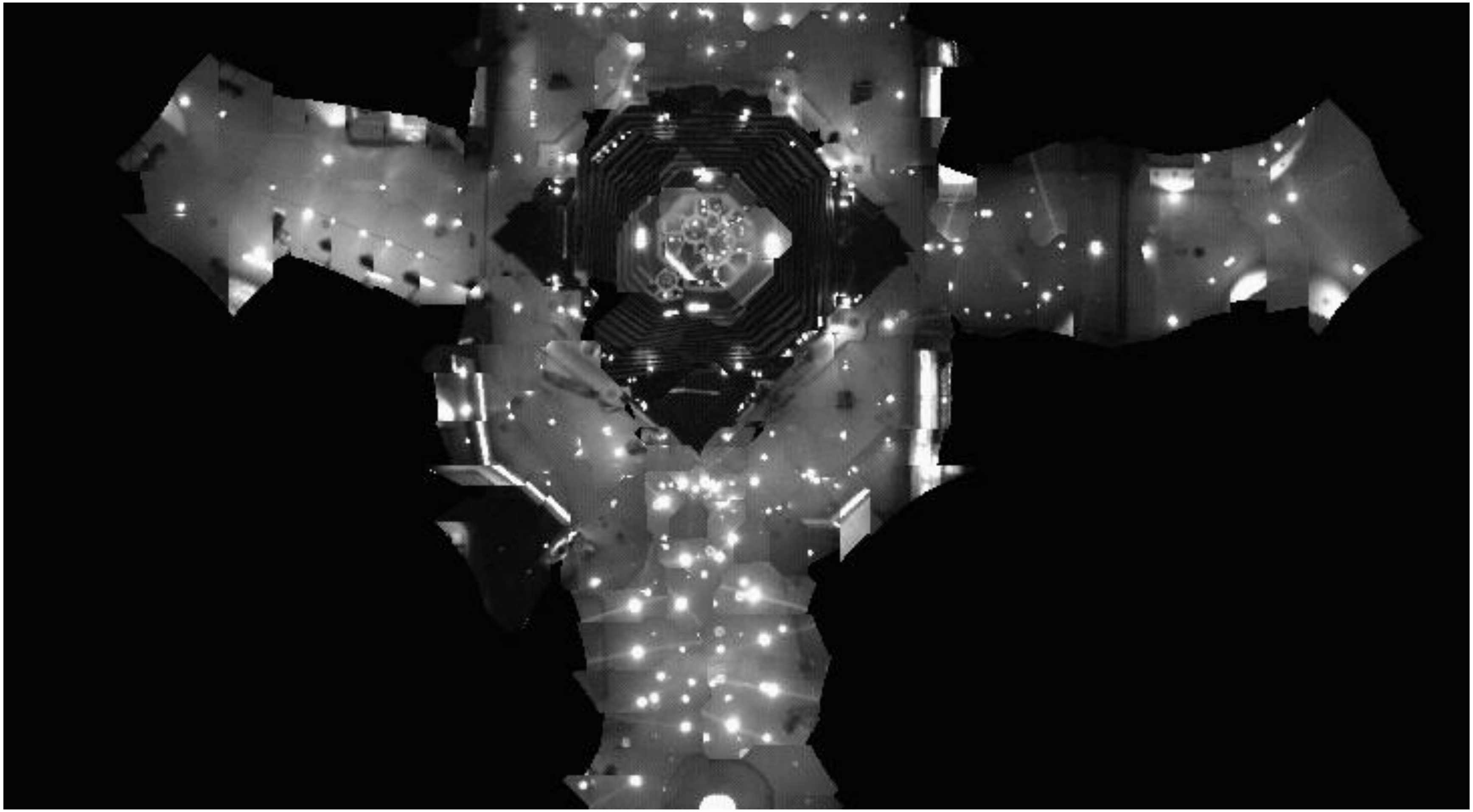
The Minerva Experience

Using Ceiling Maps for Localization

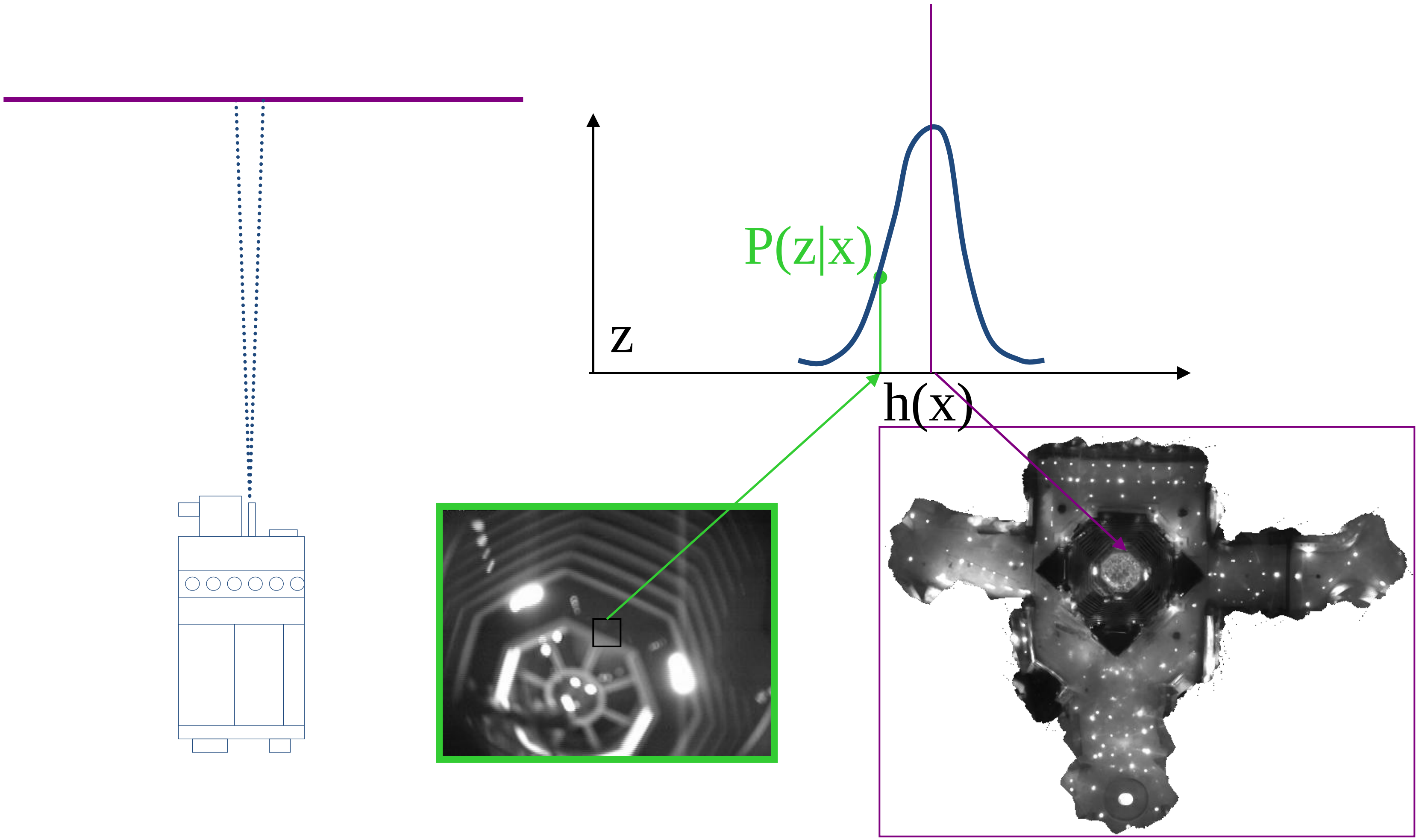


[Dellaert et al. 99]

Ceiling Light Localization

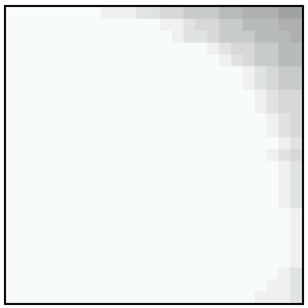


Vision-based Localization

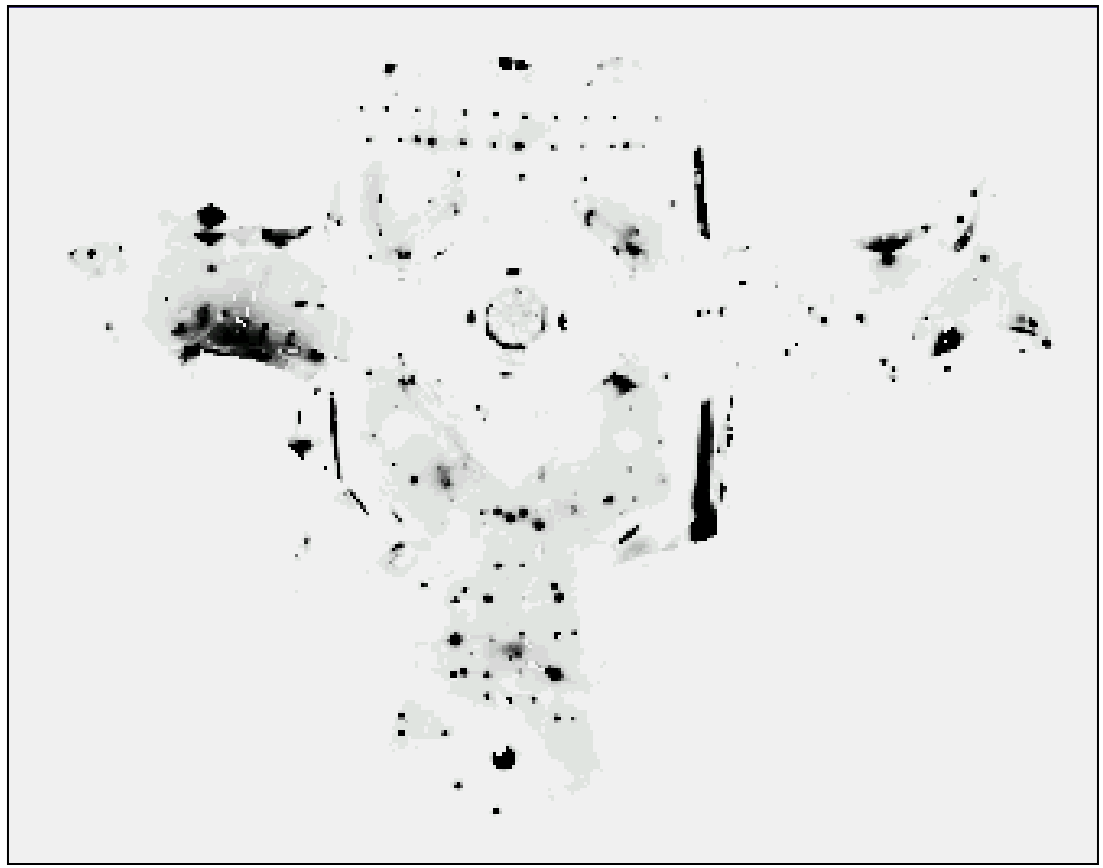


Under a Light

Measurement z:

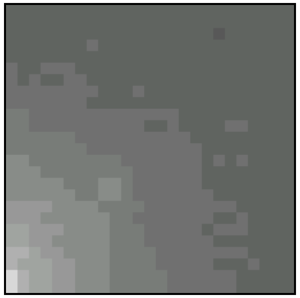


***P(z|x)*:**

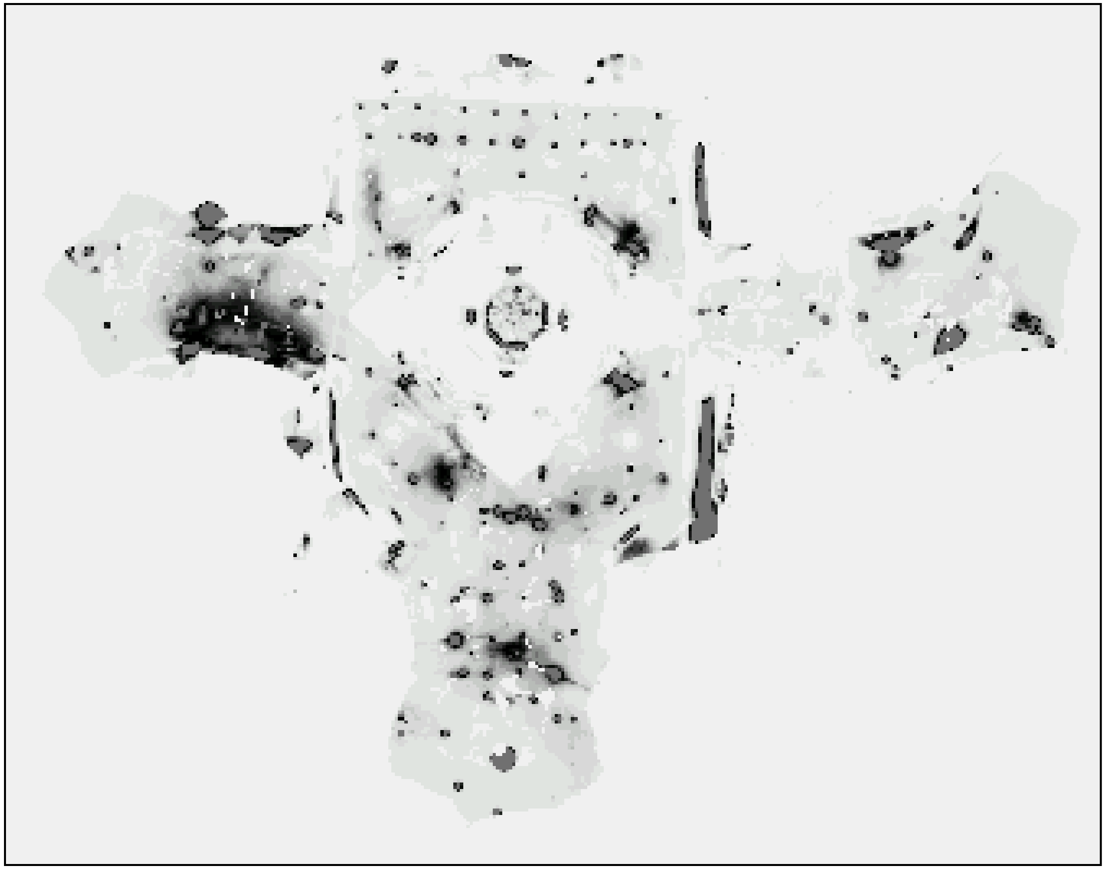


Next to a Light

Measurement z:



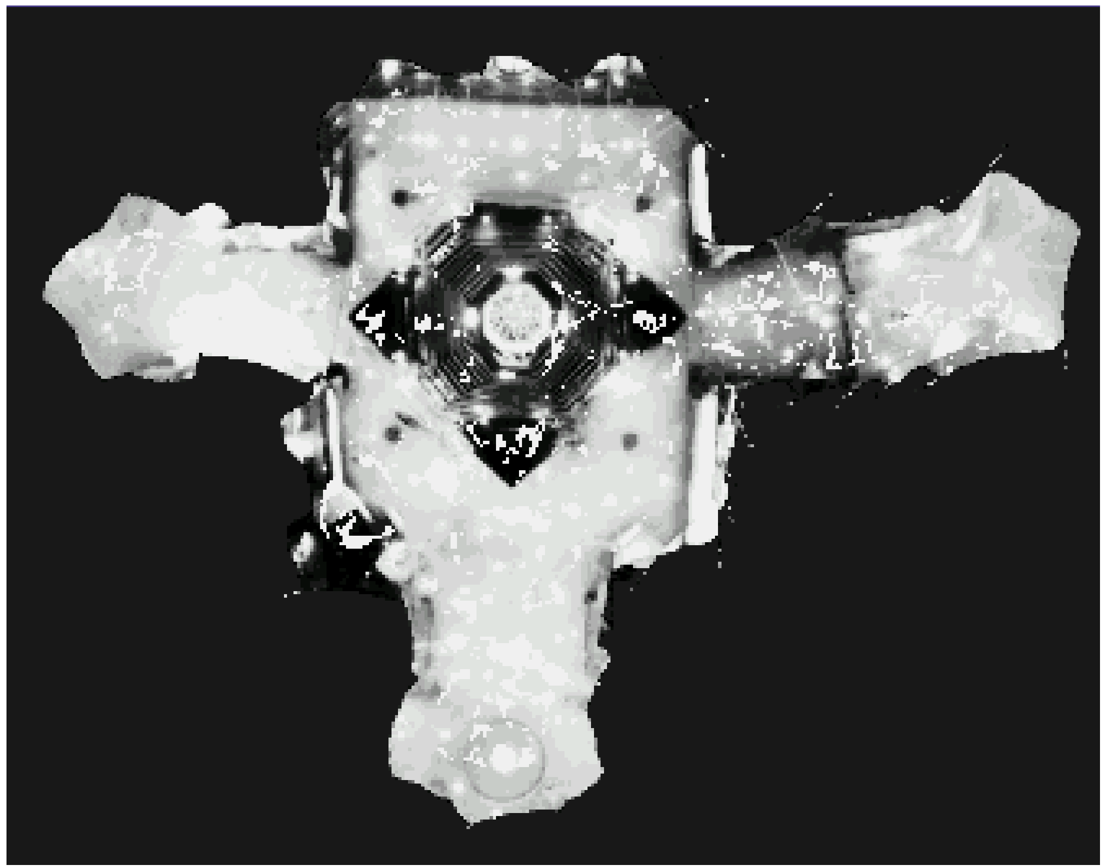
***P(z|x)*:**



Elsewhere

Measurement z:

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



Sample-based Localization Demos

http://www.cs.washington.edu/ai/Mobile_Robotics//mcl/

Localization Algorithms - Comparison

	Kalman filter	Multi-hypothesis tracking	Topological maps	Grid-based (fixed/variable)	Particle filter
Sensors	Gaussian	Gaussian	Features	Non-Gaussian	Non-Gaussian
Posterior	Gaussian	Multi-modal	Piecewise constant	Piecewise constant	Samples
Efficiency (memory)	++	++	++	-/+	+/++
Efficiency (time)	++	++	++	o/+	+/++
Implementation	+	o	+	+/o	++
Accuracy	++	++	-	+/++	++
Robustness	-	+	+	++	+/++
Global localization	No	Yes	Yes	Yes	Yes

Bayes Filters for Robot Localization

Grid fixed/variable resolution

Piecewise constant approximation
Arbitrary posteriors
Non-linear dynamics/observations
Optimal, converges to true posterior
Exponential in state dimensions
Global localization

Topological

Abstract state space
Arbitrary, discrete posteriors
Abstract dynamics/observations
One-dimensional graph
Global localization

Particle filter

Sample-based approximation
Arbitrary posteriors
Non-linear dynamics/observations
Optimal, converges to true posterior
Exponential in state dimensions
Global localization

Discrete

Kalman filter

First and second moment
Linear dynamics/observations
Optimal (linear, Gaussian)
Quadratic in state dimension
Position tracking

Extended / unscented Kalman filter

First and second moment
Non-linear dynamics/observ.
Not optimal (linear approx.)
Quadratic in state dimension
Position tracking

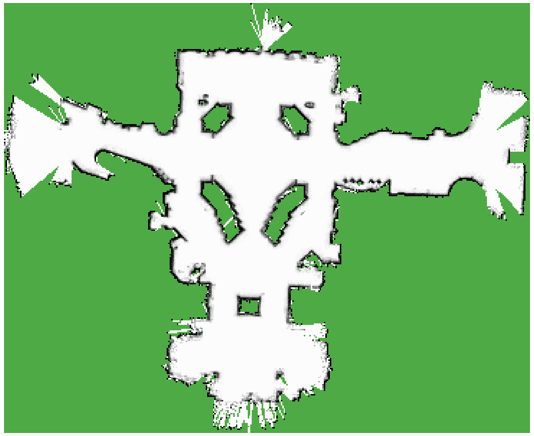
Multi-hypothesis tracking (EKF)

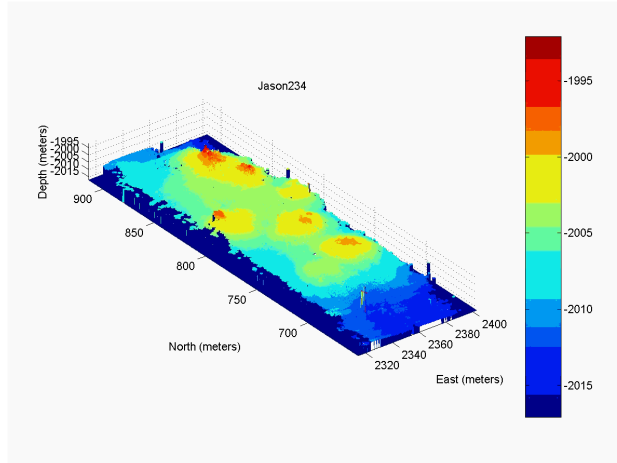
Multi-modal Gaussian
Non-linear dynamics/observations
Not optimal (linear approx.)
Polynomial in state dimension
Global localization

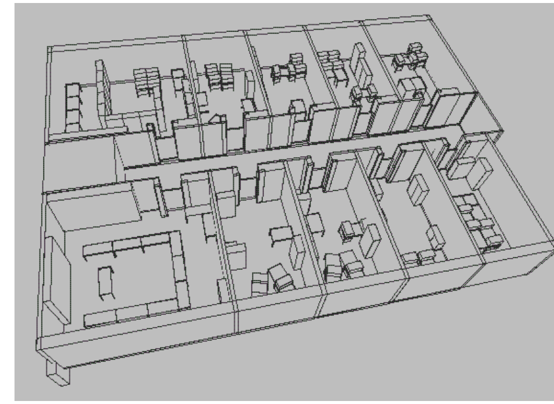
Continuous

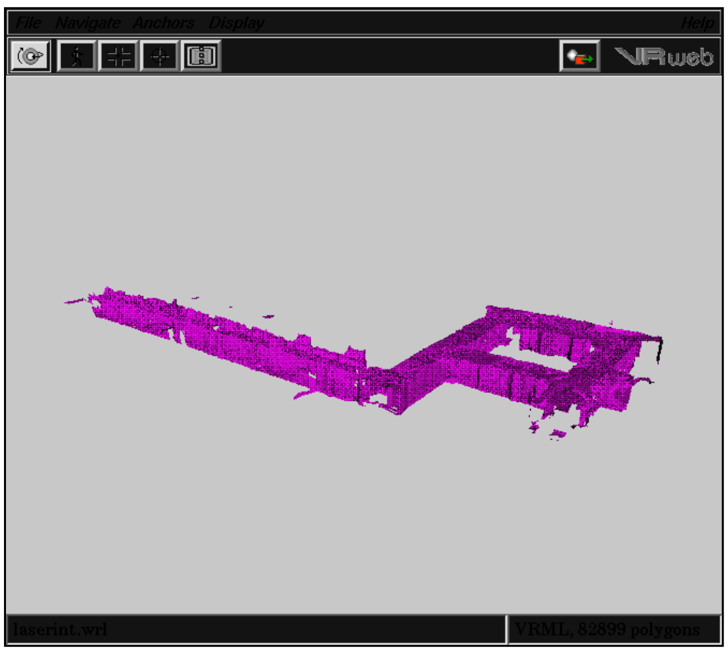
Bayes filters

Some Maps




Jason234
Depth (meters)
North (meters)
East (meters)
900
850
800
750
700
2320
2340
2360
2380
2400
-1995
-2000
-2005
-2010
-2015




File Navigate Anchors Display
Help
VRweb
laserint.wrl
VRML, 82899 polygons

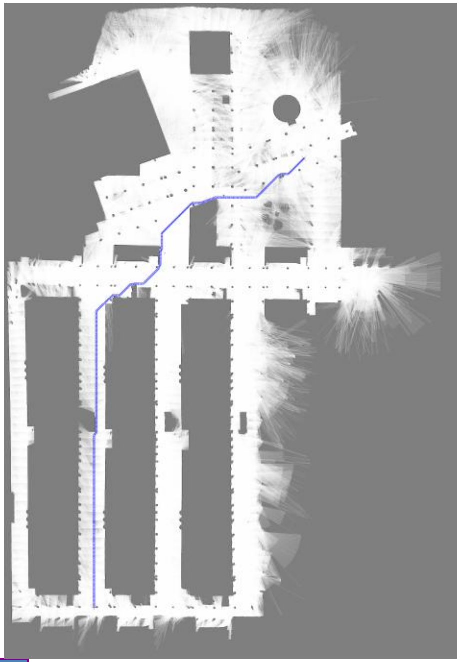

File Navigate Anchors Display
Help
VRweb
Drag mouse to walk/move/look around
VRML, 80470 polygons, te

Problems in Mapping

- Sensor interpretation
  - How do we extract relevant information from raw sensor data?
  - How do we represent and integrate this information over time?
  - Do we map for the purpose of localization or do we map for “human consumption” (dense vs sparse)?
- Robot locations have to be known
  - How can we estimate them during mapping?

Occupancy Grid Maps

- Introduced by Moravec and Elfes in 1985
- Represent environment by a grid.
- Estimate the probability that a location is occupied by an obstacle.
- Key assumptions
  - Occupancy of individual cells is independent

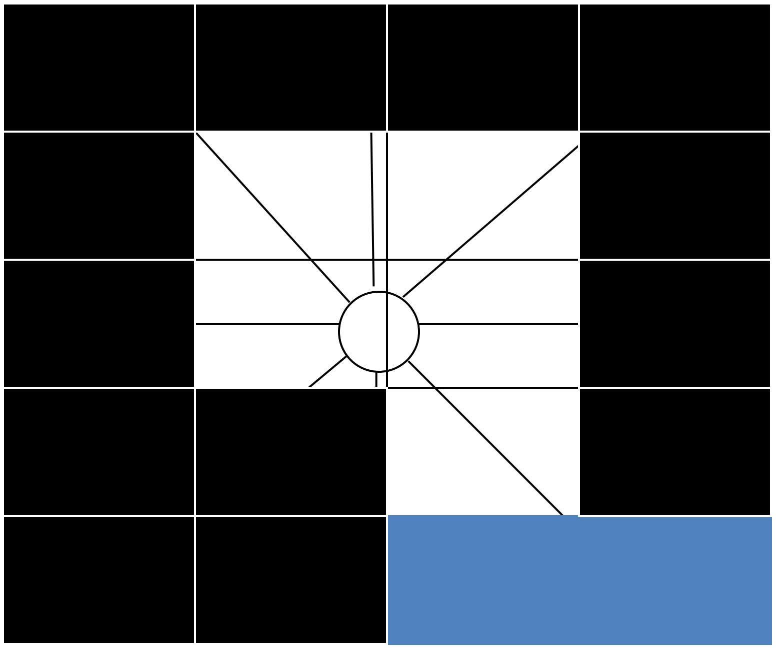


$$
\begin{aligned}
Bel(m_t) &= P(m_t \mid x(1:k), y(1:k)) \\
&= \prod_{x,y} Bel(m_t^{[xy]} \mid x(1:k), y(1:k))
\end{aligned}
$$

  - Robot positions x(1:k) are known!

The Basic Intuition

- A ray that flies through an area of space indicates that the space is probably free
- A ray where that reflects at a point indicates something there
- Otherwise, we don't know



One Idea: Simple Counting

- For every cell count
  - hits(*x*,*y*): number of cases where a beam ended at <*x*,*y*>
  - misses(*x*,*y*): number of cases where a beam passed through <*x*,*y*>

$$Bel(m^{[xy]}) = \frac{\mathrm{hits}(x, y)}{\mathrm{hits}(x, y) + \mathrm{misses}(x, y)}$$

- Assumption: $P(occupied(x,y)) = P(reflects(x,y))$

- Many cases where this is not a good approximation … e.g. sonar reflection model

Updating Occupancy Grid Maps

- Note that the following also can be derived:

$$
\begin{aligned}
&P(\O m^{[xy]} \mid x(1:k), y(1:k)) = \\
&\qquad P(\O m^{[xy]} \mid y(k), x(k)) P(y(k) \mid x(k)) P(\O m^{[xy]} \mid x(1:k-1), y(1:k-1))
\end{aligned}
$$

- Now, consider the odds ratio:

$$
Odds(x) = \frac{P(x)}{1 - P(x)}
$$

$$
= \frac{P(x)}{P(\neg x)}
$$

$$
\begin{aligned}
&O(m^{[xy]} \mid x(1:k), y(1:k)) = \\
&\qquad h O(m^{[xy]} \mid y(k), x(k)) O(m^{[xy]} \mid x(1:k-1), y(1:k-1))
\end{aligned}
$$

inverse sensor model

Prior odds

Updating Occupancy Grid Maps

- Typically updated using inverse sensor model and log odds ratio (log *ab* = log *a* + log *b*)

$$\log O(m \mid x(1:k), y(1:k)) = \log O(m \mid x(k), y(k)) + \log O(m \mid x(1:k-1), y(1:k-1)) + \log \eta$$

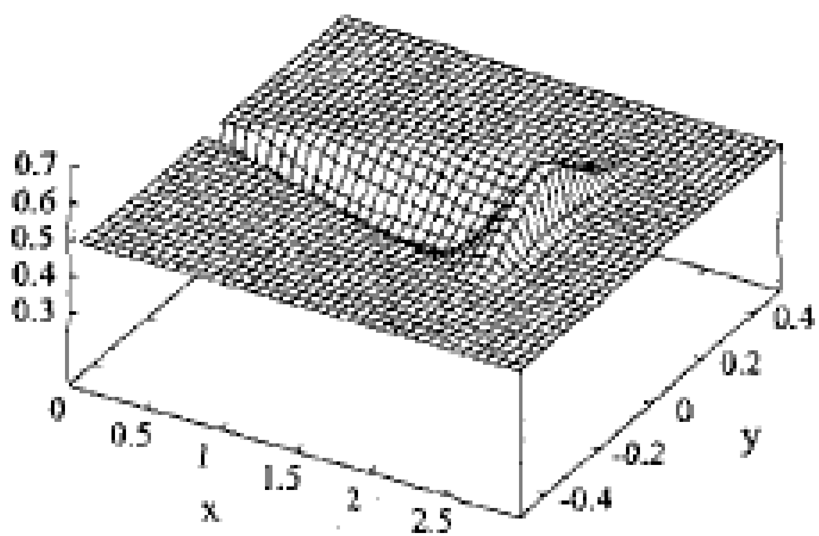
and then recover the probability with

$$P(x) = \left[1 + \frac{1}{Odds(x)}\right]^{-1}$$

$$P(m \mid x(1:k), y(1:k)) = \left[1 + \frac{1 - P(m \mid x(k), y(k))}{P(m \mid x(k), y(k)} \frac{\eta}{1-\eta} \frac{1 - P(m \mid x(1:k-1), y(1:k-1)}{P(m \mid x(1:k-1), y(1:k-1)}\right]^{-1}$$

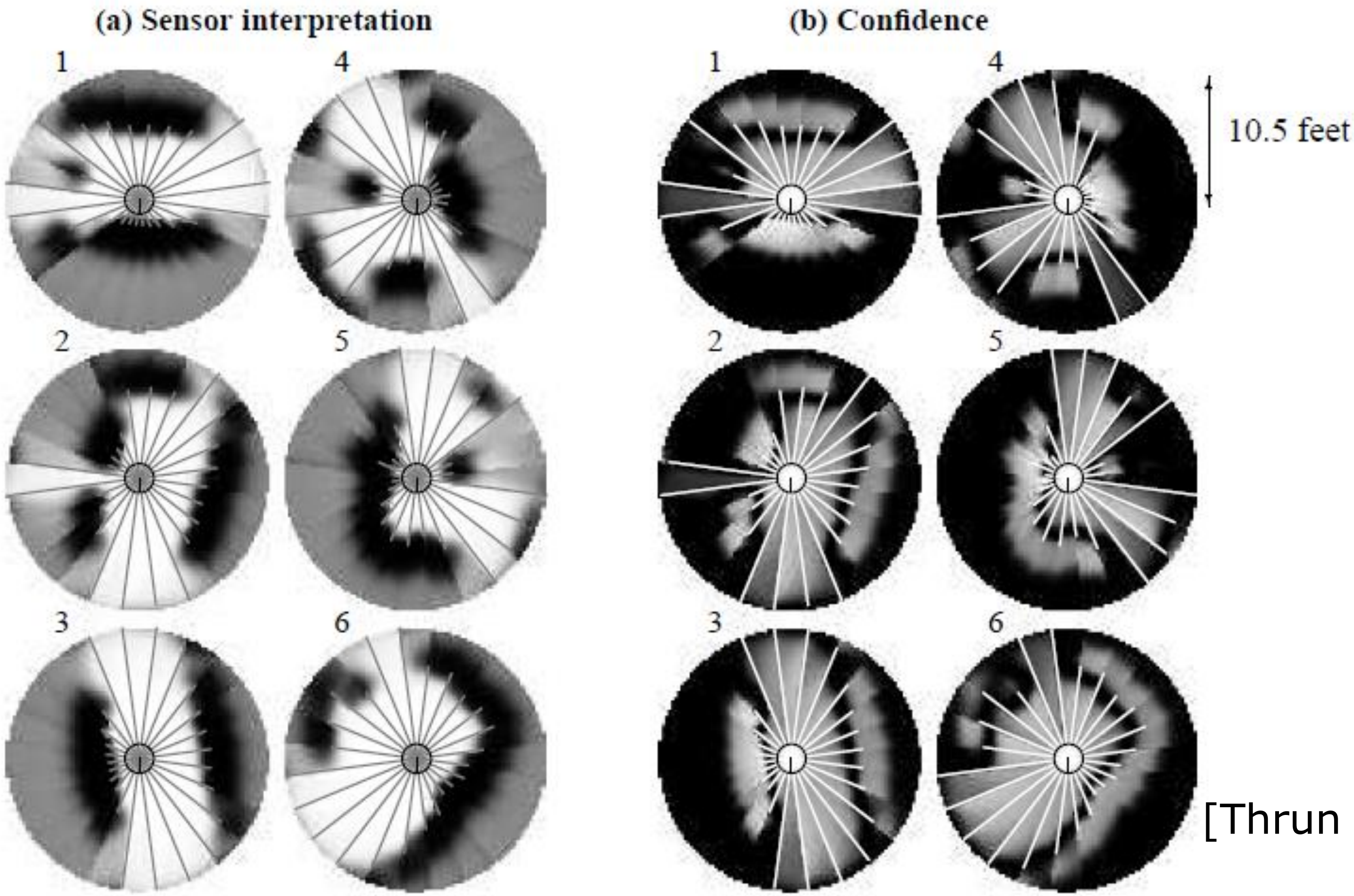
The Inverse Sensor Model

- The probability a cell is occupied given observation and localization $P(m \mid x(k), y(k))$
  Assume that all cells are independent $P(m) = \prod_l P(m_l)$
  then specify $P(m_l \mid x(k), y(k))$, which is the probability of cell m_l is occupied given the measurement $y(k)$ in position $x(k)$



Learning Inverse Sensor Model

- Learn probabilities with neural network.
- Consider four beams simultaneously.



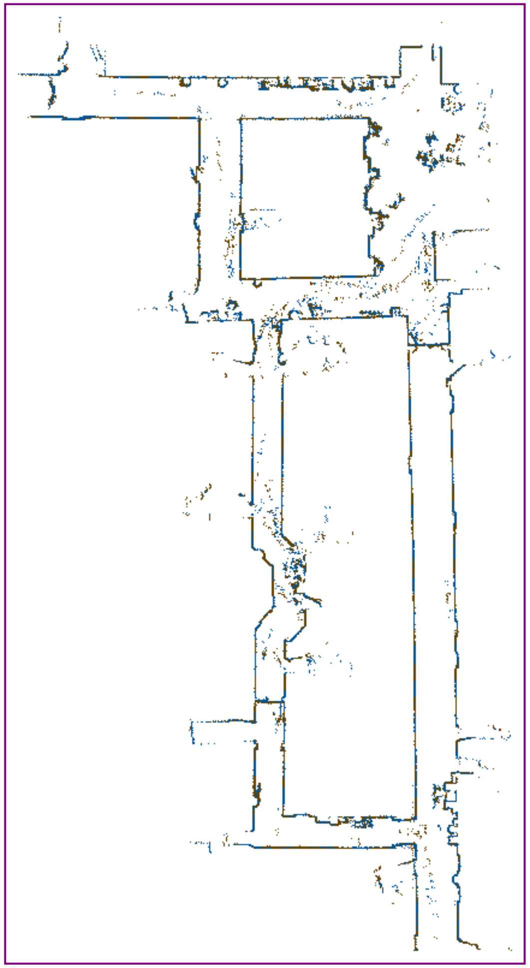
[Thrun ’98]

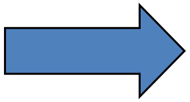
Occupancy Grid Algorithm

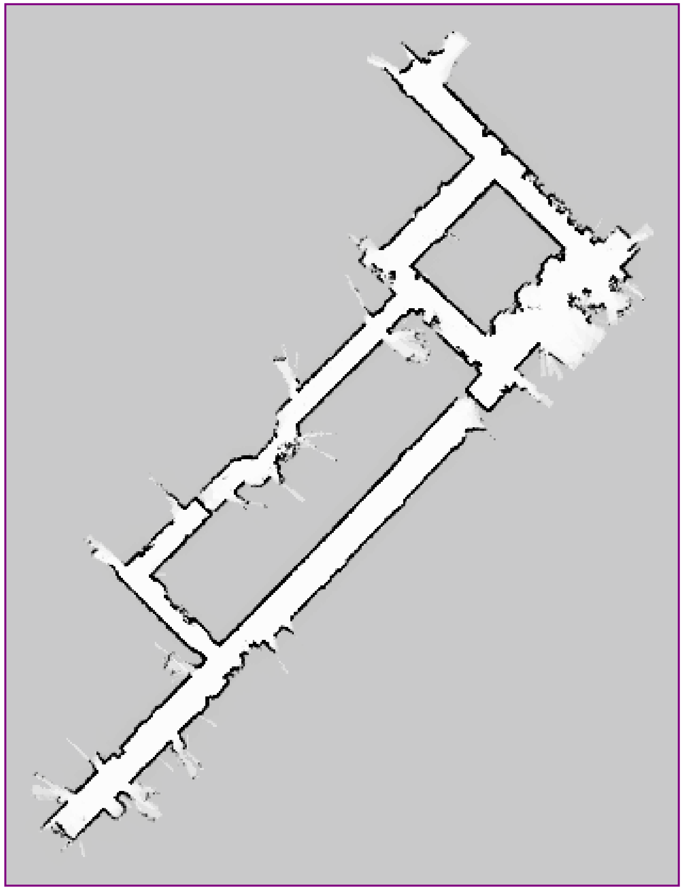
Algorithm **Occupancy_grid**($x_{1:k}$, $y_{1:k}$, $P_0(m)$):

1. $P_m = P_0(m)$
2. for i=1 to k
3. $$P_m = \left[1 + \frac{1 - P(m \mid x(k), y(k))}{P(m \mid x(k), y(k)} \frac{\eta}{1-\eta} \frac{1-P_m}{P_m}\right]^{-1}$$

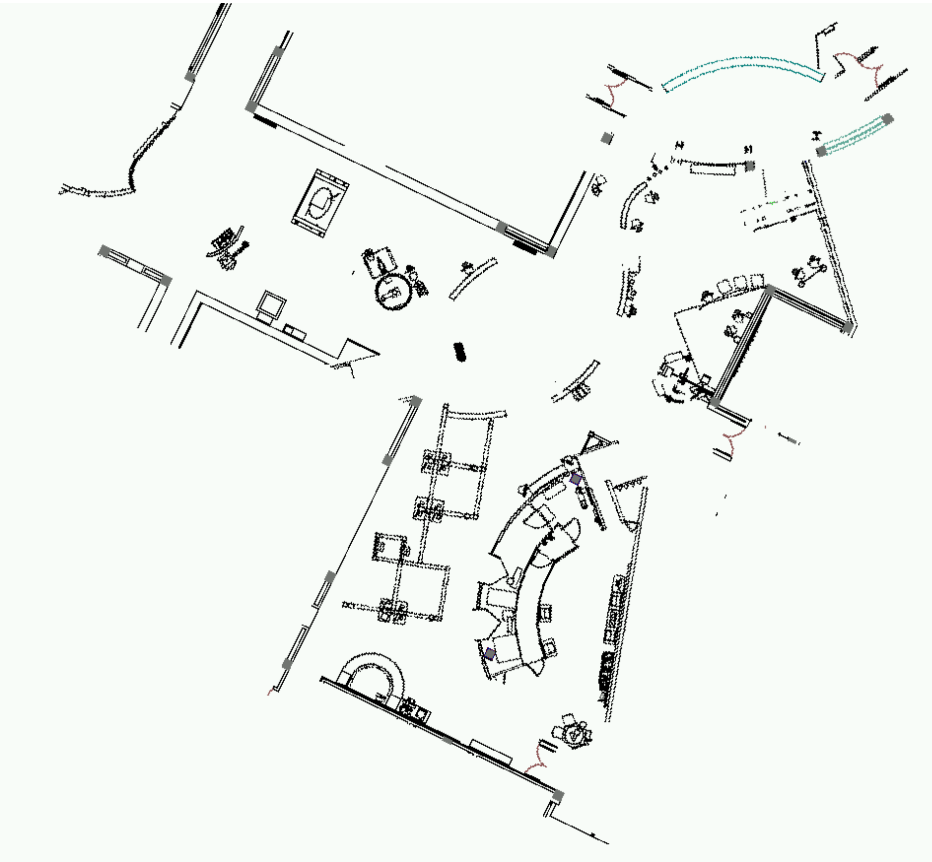
Occupancy Grids: From scans to maps



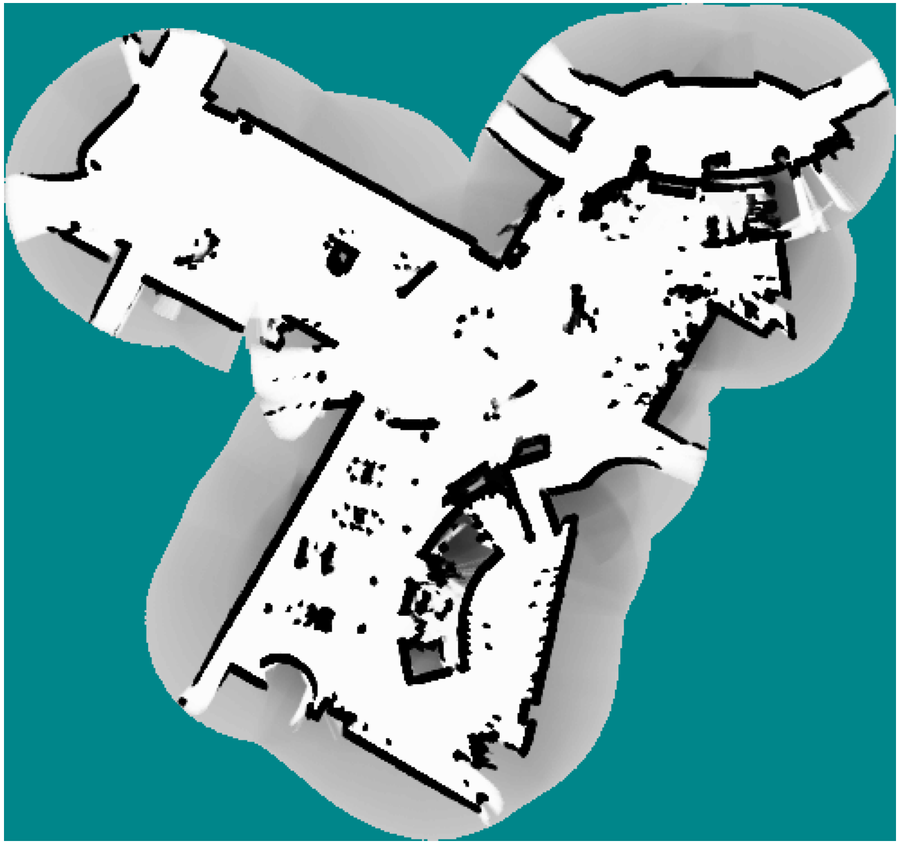




Tech Museum, San Jose



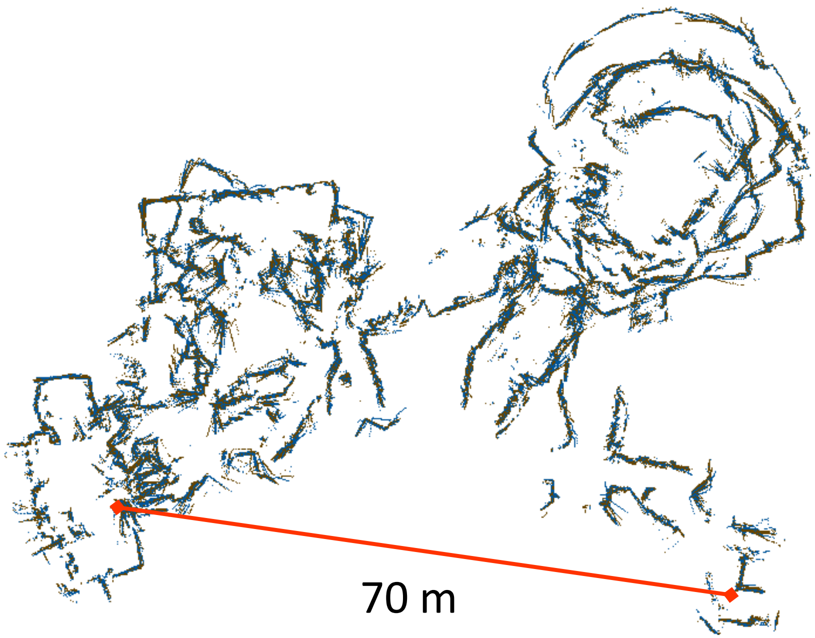
CAD map

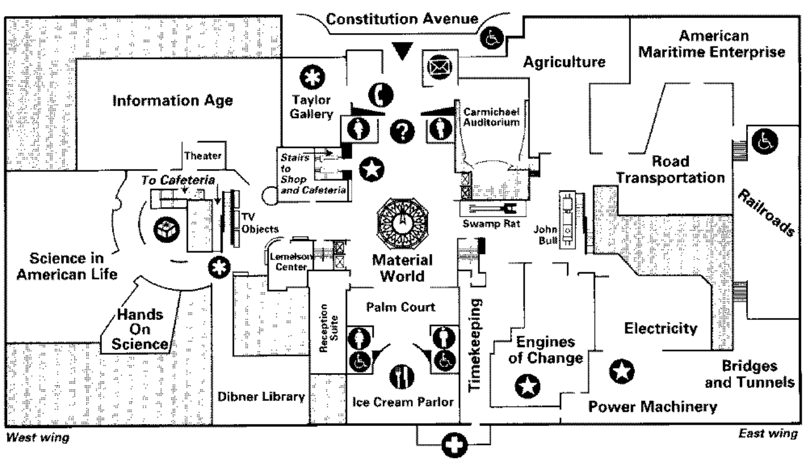


occupancy grid map

Concurrent Mapping and Localization

- Chicken-and-egg problem
  - Mapping with known poses is “simple”
  - Localization with known map is “simple”
  - But in combination the problem is hard!





Mapping with Expectation Maximization

- **Idea**: Maximum likelihood with unknown data association.

$$Bel(m, x(k)) \equiv P(x(1:k), m \mid u(0:k-1), y(1:k))$$

$$Bel(m, x(k)) = \eta \, p(y(k) \mid m, x(k)) \int p(x(k) \mid x(k-1), u(k-1)) Bel(m, x(k-1)) \, dx_{t-1}$$

Take this apart: first estimate location given map, then estimate map given location

- **EM**: Maximize log-likelihood by iterating

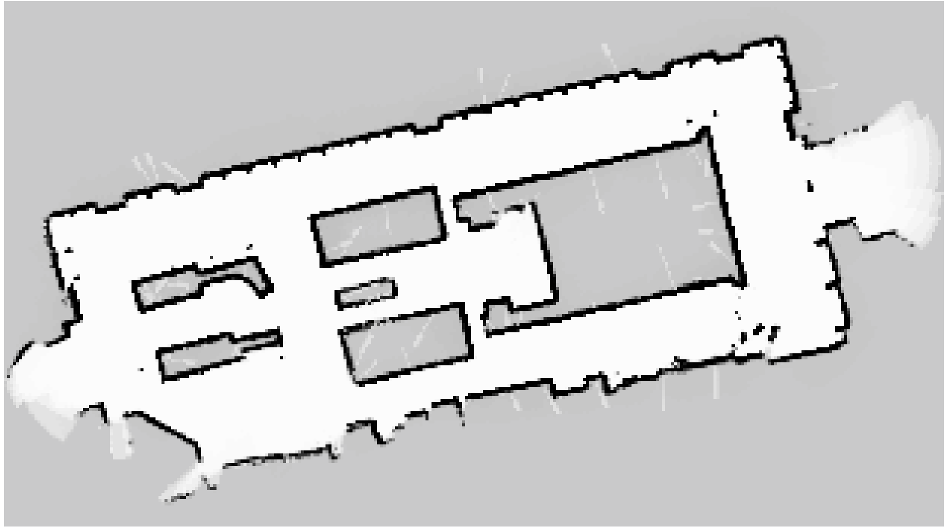
E-step: $Q[x(1:k) \mid m^{[k]}] = E_{m^{[k]}}[\log \; p(x(1:k) \mid m^{[k]}, u(0:k-1), y(1:k))]$

$\rightarrow$ Localization (bi-directional)

M-step: $m^{[k+1]} = \underset{m}{\operatorname{argmax}} \; Q[x(1:k) \mid m^{[k]}]$

$\rightarrow$ Mapping with known poses

EM Mapping, Example (width 45 m)



CSIRO

SLAM

- Idea: Given the true trajectory of the robot, all landmark detections are independent.

P(x(1:k),m | u(0:k-1),y(1:k)) =

P(m | x(1:k), y(1:k), u(0:k-1)) P(x(1:k) | y(1:k) u(0:k-1)) =

P(m | x(1:k),y(1:k)) P(x(1:k) | y(1:k),u(0:k-1))

  – The first term is “easy” (mapping given location and data)
  – The second term is “easy” (predict location from prior data)
  – The “hard” part: we know have to represent distributions on *trajectories*!

- We can use Rao-Blackwellised particle filters to estimate robot locations and landmark locations. (FastSLAM, Montemerlo)

- Update can be done efficiently (O(m log n)).

Sufficient Statistic

- A statistic is a function $T(X_1, X_2, \ldots X_n)$ of the random samples $X_1, X_2, \ldots, X_n$

Examples:

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - \overline{X}\right)^2$$

$$T = \max\left\{X_1, X_2, \ldots, X_n\right\}$$

Sufficient Statistic

- Given $X_1, X_2, \ldots, X_n$, is there a small set of statistics that can contain all the information about the samples?
- If $f_\theta(x)$ is a probability distribution (θ is parameter), then T is a sufficient statistic for θ if we can factorize

$$f_\theta(x) = h(x)\, g_\theta(T(x))$$

Function of the statistic

Not a function of θ

- If T is a sufficient statistic then it contain all the information to compute an estimate of θ

Sufficient Statistic

- Given $X_1, X_2, \ldots, X_n$, is there a statistic that can contain all the information about the samples?
  - Poisson distribution:
    - $X_1, X_2, \ldots, X_n$ are independent samples from a Poisson distribution with parameter λ. Then a sufficient statistic $T(X)$ of λ is

$$T(X) = \sum_{i=1}^{n} X_i$$

    - Note that $T(X)$ does not depend on λ

Sufficient Statistic

- Given $X_1, X_2, \ldots, X_n$, is there a statistic that can contain all the information about the samples?
  - Exponential distribution:
    - $X_1, X_2, \ldots, X_n$ are independent and exponentially distributed with expected value λ. Then a sufficient statistic $T(X)$ of λ is

$$T(X) = \sum_{i=1}^{n} X_i$$

    - Note that $T(X)$ does not depend on λ

Rao-Blackwell Theorem

- **Improve** the efficiency of an *estimator* by taking its *conditional expectation* with respect to a *sufficient statistic*
  - Estimator $\hat{\theta}(X)$: Is a statistic (rule) used to estimate an unobservable parameter θ in a population
    - The average of N random samples is an estimator of the population's average (i.e. height)
  - Sufficient statistic $T(X)$: Given $T(X)$, the distribution of observable samples X does not depends on the unobservable parameter θ
  - Rao Blackwell estimator of θ is defined by

$$\hat{\theta}_R(X) \equiv E[\hat{\theta}(X) \mid T(X)]$$

  and has better mean squared error than $\hat{\theta}(X)$

What About SLAM?

- Recall the Bayes filtering problem. The posterior satisfies the recursion

$$P(z_{0:k} \mid y_{1:k}) = \eta_k P(y_k \mid z_k) \int_{z_{k-1}} P(z_k \mid z_{k-1}) P(z_{0:k-1} \mid y_{1:k-1})$$

where z_k is hidden (put up with “z” instead of “x” for now)

- That integral is often not tractable and numeric approximations with samples is often used

Marginalize the State Space in Two

- Suppose we can divide z_k in two groups: x_k and m_k such that $P(z_k|z_{k-1}) = P(m_k|x_{k-1:k}, m_{k-1})P(x_k|x_{k-1})$ and assume that $P(m_{0:k}|y_{1:k}, x_{0:k})$ is tractable
- Then we can marginalize $x_{0:k}$ from the posterior and focus on estimating $P(x_{0:k}|y_{1:k})$ which is a "smaller" problem. Essentially we convert the problem to

$$P(x_{0:k}, m_{0:k} \mid y_{1:k}) = \underbrace{P(m_{0:k}|y_{1:k}, x_{0:k})}_{\text{Optimal Filt.}} \quad \underbrace{P(x_{0:k}|y_{1:k})}_{\text{Particle Filt.}}$$

and the posterior distribution of $P(x_{0:k}|y_{1:k})$ is given

$$P(x_{0:k} \mid y_{1:k}) = n_k P(y_k \mid x_k) \int_{x_{k-1}} P(x_k \mid x_{k-1}) P(x_{0:k-1} \mid y_{1:k-1})$$

The dimension of $P(x_{0:k}|y_{1:k})$ is smaller than $P(x_{0:k}, m_{0:k} \mid y_{1:k})$

Rao-Blackwellized SLAM

Compute a posterior over the map and possible trajectories of the robot :

map and trajectory

measurements

$$p(x_{1:k}, m \mid y_{1:k}, u_{0:k-1})$$
$$= p(m \mid x_{1:k}, y_{1:k}, u_{0:t-1})\, p(x_{1:k} \mid y_{1:k}, u_{0:k-1})$$

mapping

Localization

map

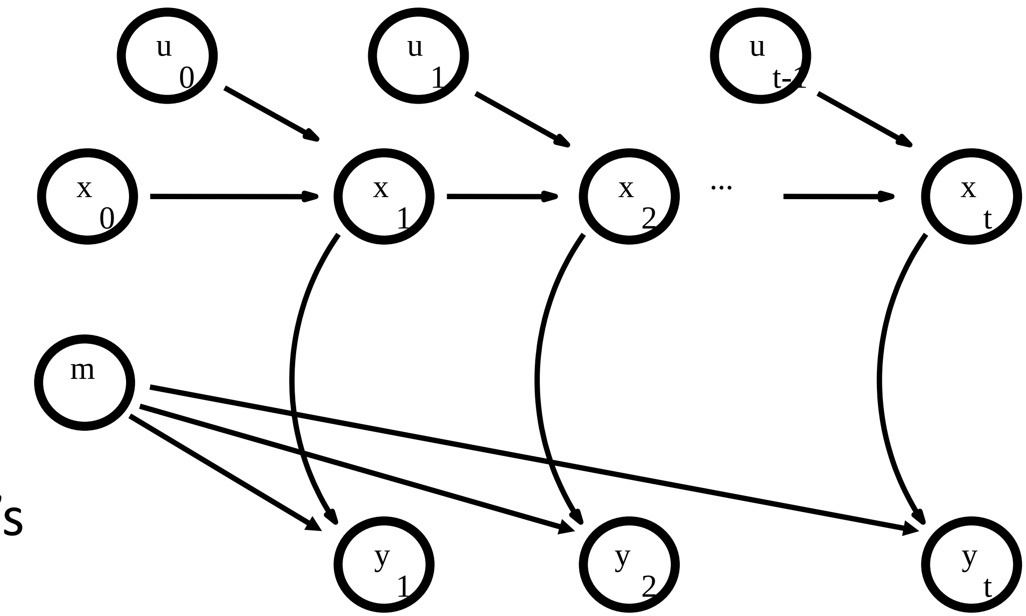
robot motion

trajectory

Begin Courtesy Dieter Fox

A Graphical Model of Rao-Blackwellized SLAM

- If we know the map
  - Estimate localize at each step $x_{1:k}$
- If we know locations $x_{1:k}$
  - Compute the map
- Particle filtering
  - Each particle represent the posterior trajectory
  - Compute the map corresponding to the particle's trajectory
  - Particle's weight is given by the most likelihood of the most recent observation given the map



Courtesy Dieter Fox

Rao-Blackwellized SLAM

- Break it down even further if a map m_k consists of N individual landmarks $l_i = N(\mu_i, \Sigma_i)$ then

$$P(x_{1:k}, m_k | y_{1:k}, u_{0:k-1}) = P(x_{1:k} | y_{1:k}, u_{0:k-1}) \overbrace{P(m_k | x_{1:k}, y_{1:k}, u_{0:k-1})}^{\prod_i N(\mu_i, \Sigma_i)}$$

- Rao-Blackwellized particle filter (RBPF) maintains an individual map for each sample and updates this map based on the trajectory estimate of the sample
- Landmark are filtered individually and have low dimensionality
- If M particles with N landmarks there is NM landmark filters

FastSLAM

	Robot Pose	Kalman Filters			
Particle #1	x, y, θ	μ_1, Σ_1	μ_2, Σ_2	…	μ_N, Σ_N
Particle #2	x, y, θ	μ_1, Σ_1	μ_2, Σ_2	…	μ_N, Σ_N
Particle #3	x, y, θ	μ_1, Σ_1	μ_2, Σ_2	…	μ_N, Σ_N
⋮					
Particle M	x, y, θ	μ_1, Σ_1	μ_2, Σ_2	…	μ_N, Σ_N

[Begin courtesy of Mike Montemerlo]

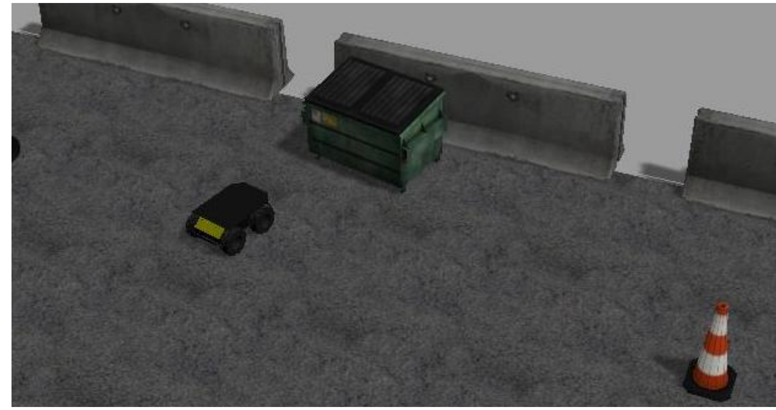
FastSLAM Algorithm O(MN)

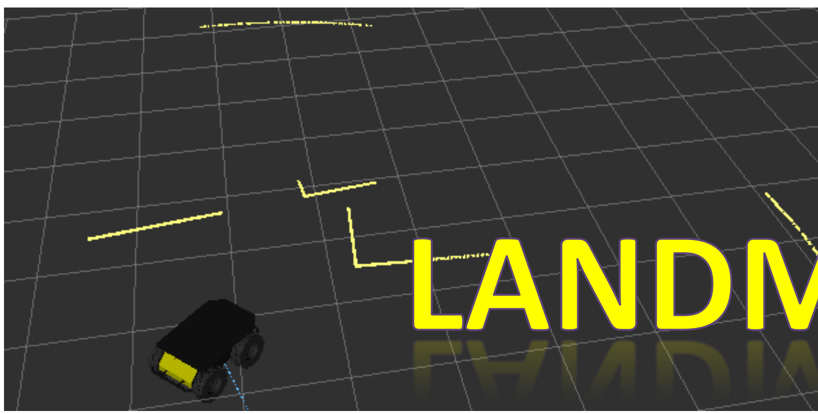
- Sample a new robot pose for each particle
  - $x_k = g(x_{k-1}, u_k) + v$
  - add this to trajectory giving $x_{1:k}$
- Update the landmark EKFs in each particle
  - We have a ``known'' (estimate) trajectory; run EKFs for each landmark
- Calculate an importance weight (difference between actual observation, y_k, and expected observation, with covariance Z)

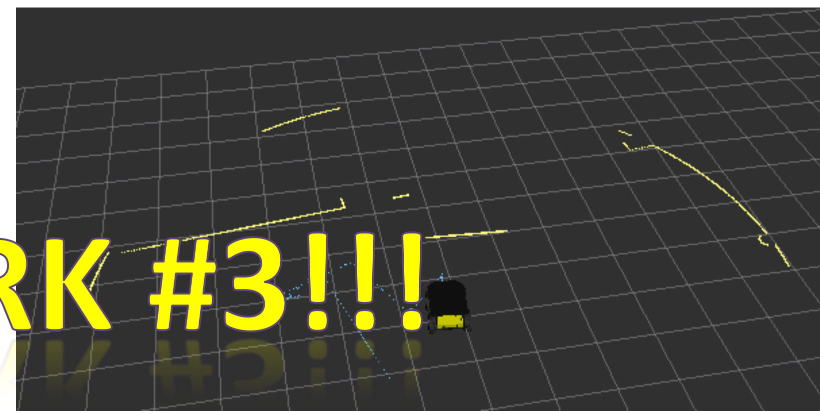
$$w_k = \frac{1}{\sqrt{2\pi Y_{n,k}}} \exp\left(-\frac{1}{2} \left(y_k - \hat{y}_{n,k} \right)^T Y_{n,k}^{-1} \left(y_k - \hat{y}_{n,k} \right) \right)$$

- Resample particle set

SLAM Data Association







LANDMARK #3!!!

Angle increment = 0.00436940183863

Index1= [0 1 2 ... 532 533 534 ... 717 718 719]

Range1=[6.0000, 5.9996, 5.9985, ..., 1.8759, 1.8771, 1.8784, ..., 3.0107, 3.0284, 3.0447]

Index2= [0 1 2 ... 302 303 304 ... 717 718 719]

Range2=[2.0254, 2.0294, 2.0347, ..., 2.0254, 2.0294, 2.0347, ..., 6.0000, 6.0000, 6.0000]

SLAM Data Association

- In practice we use a variable n_k to associate each measurement to a landmark number (id#)
  - For example $n_3 = 8$ means that measurement at time $k=3$, y_3, is associated with the landmark #8
- How do we get this n_k?

$$\hat{n}_k = \underset{n_k}{\arg\max}\, P(y_k \mid n_k, y_{k-1}, \hat{n}_{k-1}, \hat{x}_k, u_{k-1})$$

This is a "Maximum Likelihood" estimator.

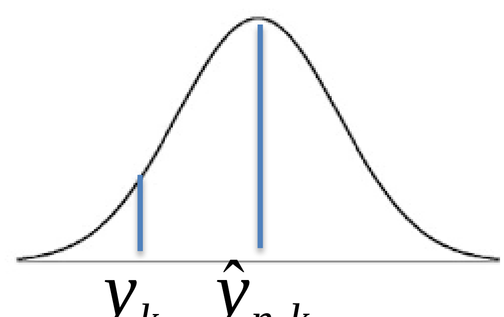
FastSLAM Data Association

- In FastSLAM, each landmark is estimated with an EKF and the likelihood can be estimated from the EKF “innovation”

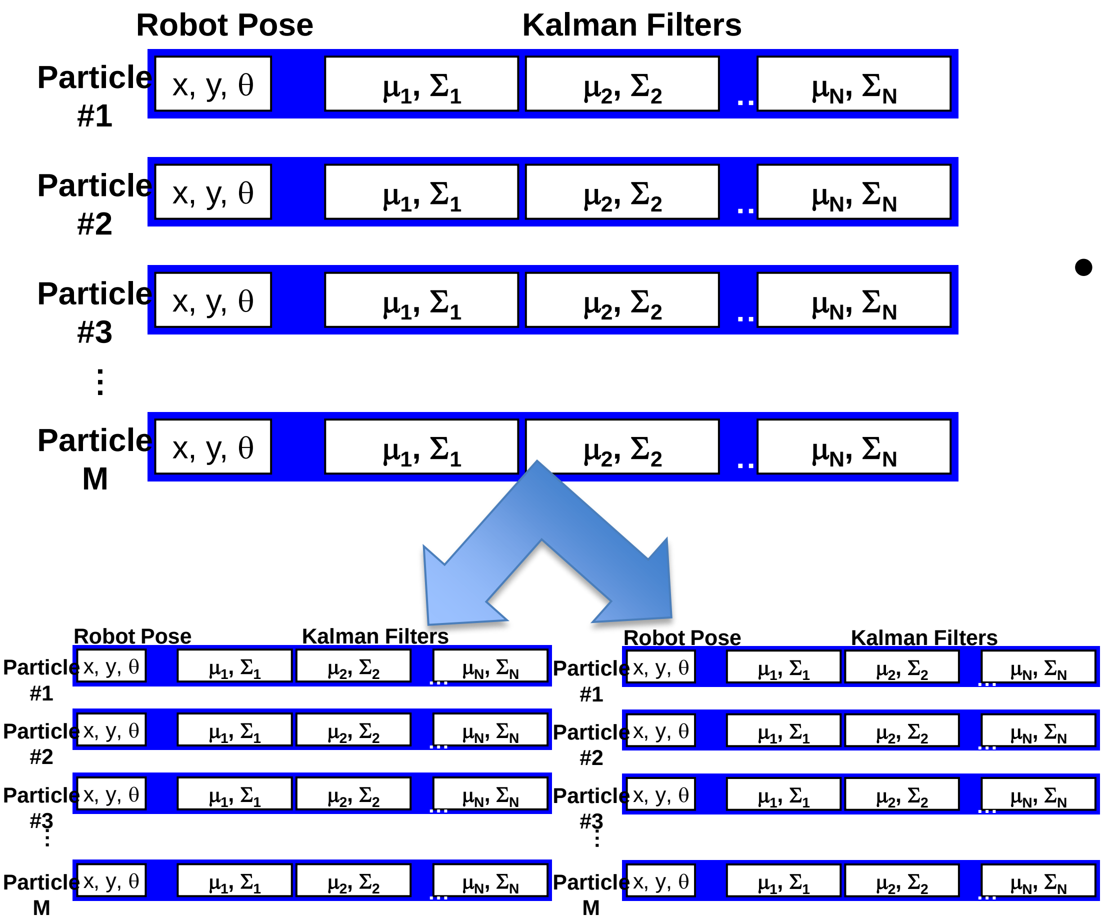
	Robot Pose	Kalman Filters			
Particle #1	x, y, θ	μ_1, Σ_1	μ_2, Σ_2	…	μ_N, Σ_N
Particle #2	x, y, θ	μ_1, Σ_1	μ_2, Σ_2	…	μ_N, Σ_N
Particle #3	x, y, θ	μ_1, Σ_1	μ_2, Σ_2	…	μ_N, Σ_N
⋮					
Particle M	x, y, θ	μ_1, Σ_1	μ_2, Σ_2	…	μ_N, Σ_N

$$P(y_k \mid y_{k-1}, \hat{n}_{k-1}, \hat{x}_k, u_{k-1}) = \frac{1}{\sqrt{2\pi Y_{n,k}}} \exp\left(-\frac{1}{2}\left(y_k - \hat{y}_{n,k}\right)^T Y_{n,k}^{-1}\left(y_k - \hat{y}_{n,k}\right)\right)$$

If the likelihood falls below a threshold, a new landmark is added



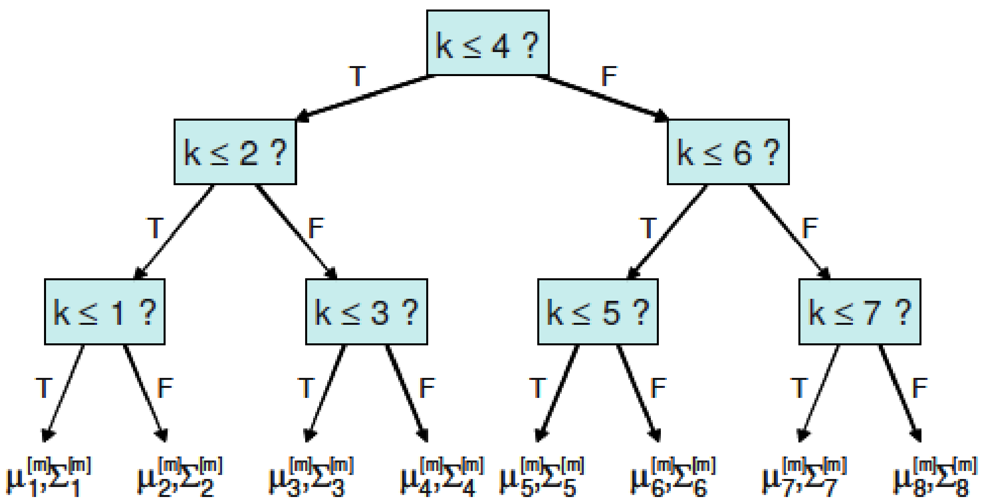
Tree of Landmarks



- FastSLAM complexity is log(MN)
  - M number of particles
  - N number of landmarks
- When we resample (with replacement) the same particle may be duplicated several times
  - Copying is linear in the size of the map
  - Most of the landmarks remain unchanged during a map update (only the visible landmarks are updated)

From Montemerlo 2003

Tree of Landmarks

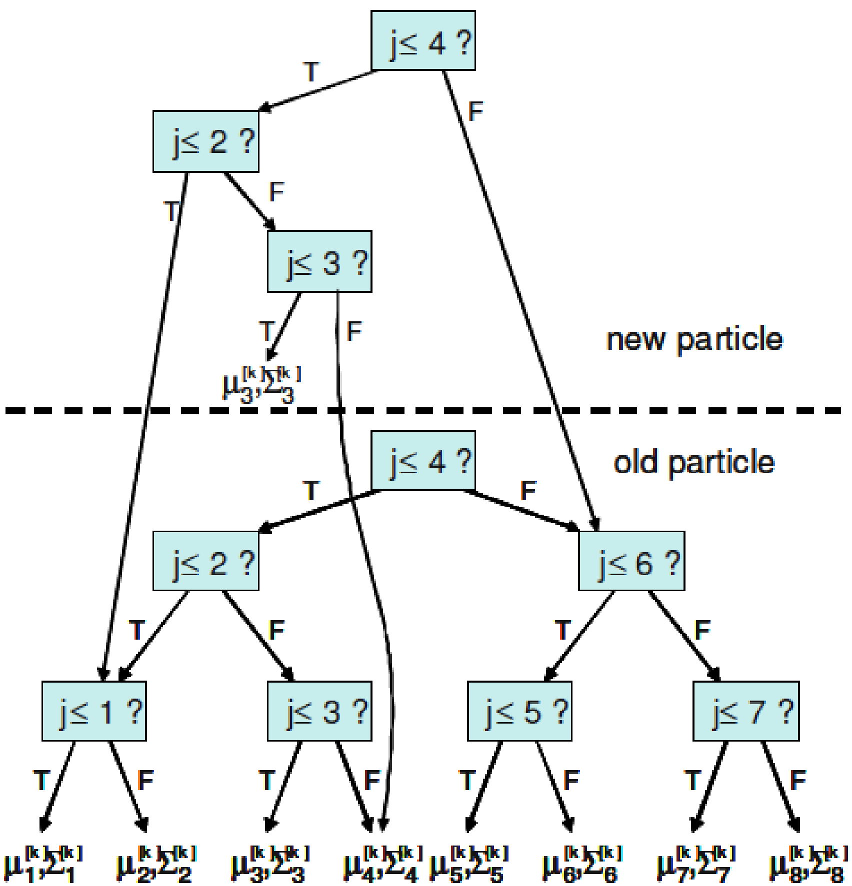


Balanced binary tree with 8 landmarks

- Use a tree of landmarks that is shared between particles
- If the tree is balanced then accessing a landmark takes log(N) and FastSLAM runs in log(M logN)

[Courtesy of Mike Montemerlo]

Tree of Landmarks



- Modifying a landmark #3. Only μ_3 and Σ_3 are modified
  - Avoid duplicating the entire tree
  - Create a new path from the root to landmark #3
  - Copy the missing pointers to the rest of the old tree
  - Keep the old pointers so the other particles can use the old values

[Courtesy of Mike Montemerlo]

FastSLAM: Victoria Park Results

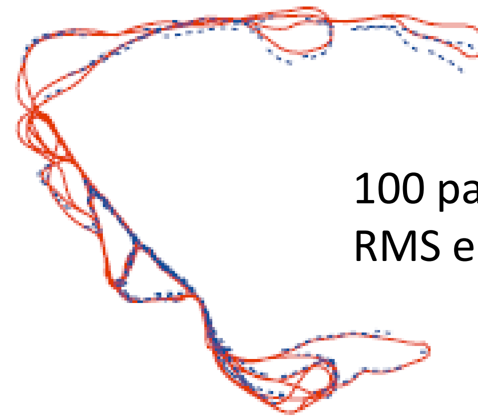
- 4 km traverse
- 100 particles
- GPS ground truth
- Uses negative evidence to remove spurious landmarks
- Uneven terrain

[Courtesy of Mike Montemerlo]

FastSLAM: Victoria Park Results

100 meters away
from it's true position

(a) Vehicle path predicted by the odometry

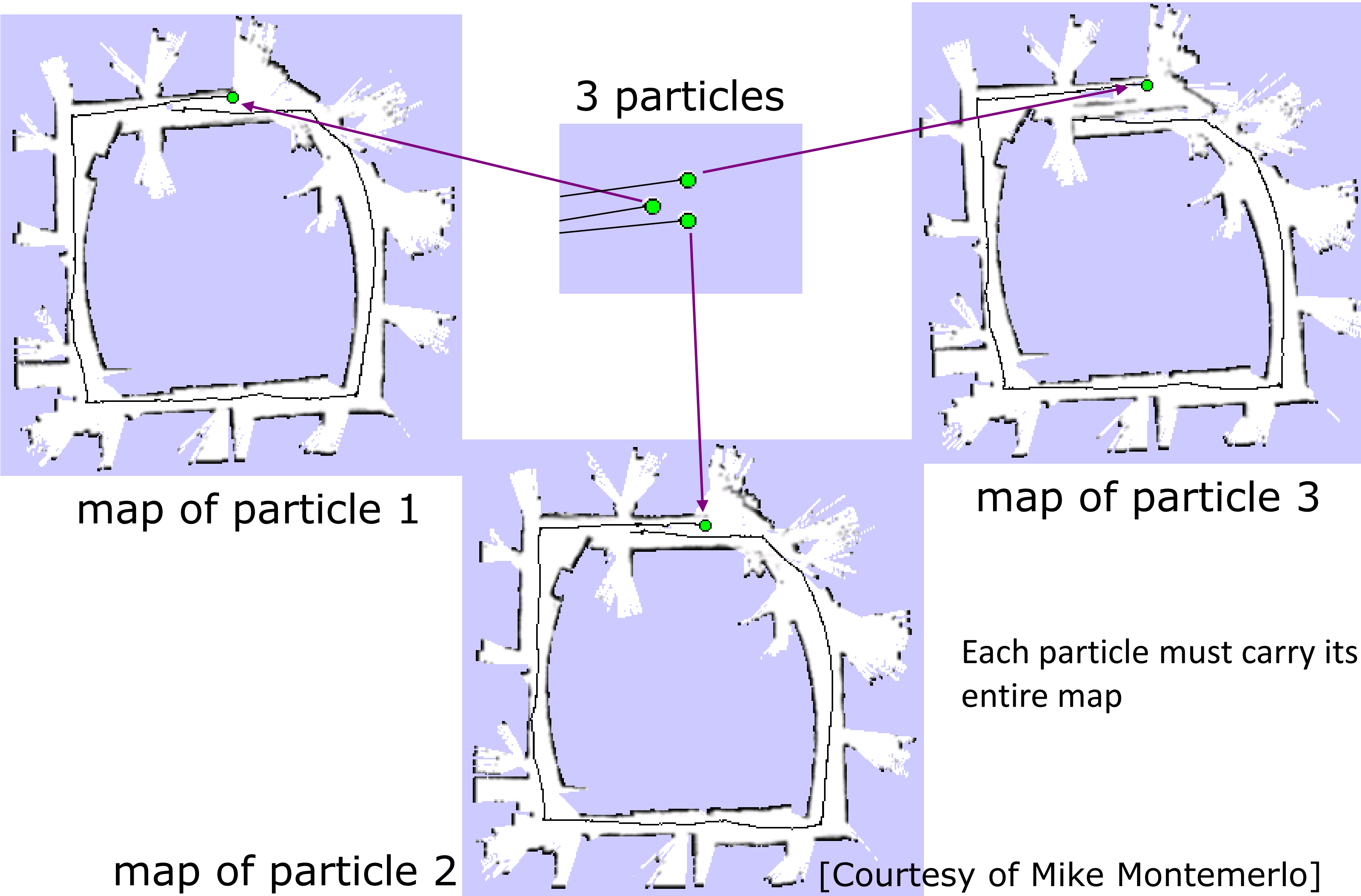


(b) True path (dashed line) and FastSLAM path (solid line)

100 particles
RMS error over 4km is ~4m

[End courtesy of Mike Montemerlo]

Grid-Based FastSLAM (occupancy grid)



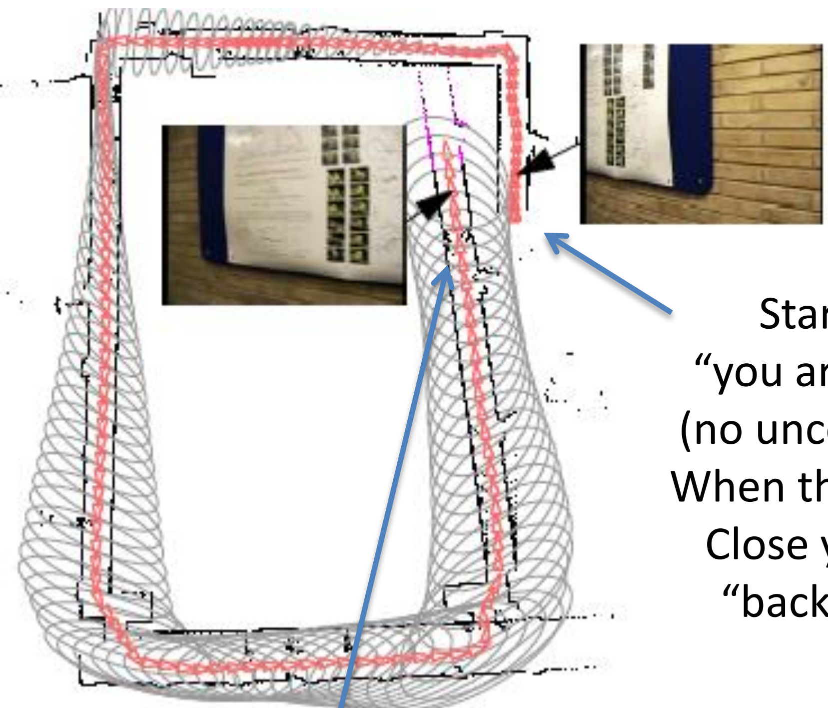
Each particle must carry its entire map

[Courtesy of Mike Montemerlo]

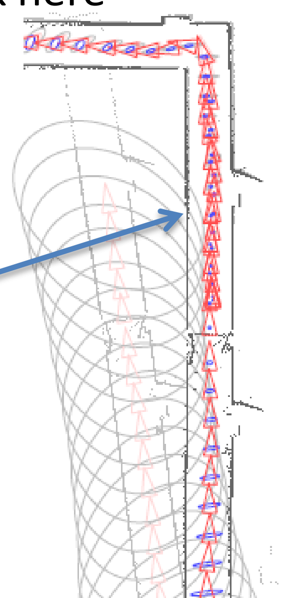
FastSLAM Example

- 500 particles
- 28mx28m
- Length of trajectory 491m
- Map resolution 10cm

Closing the Loop



Start at
“you are here”
(no uncertainty)
When the loop is
Close you are
“back here”



BEFORE

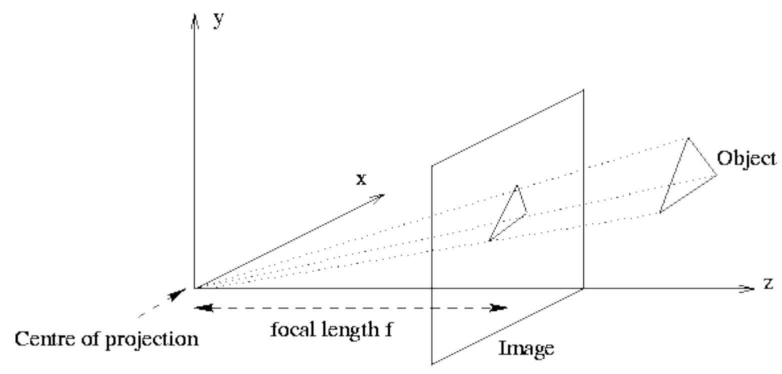
AFTER

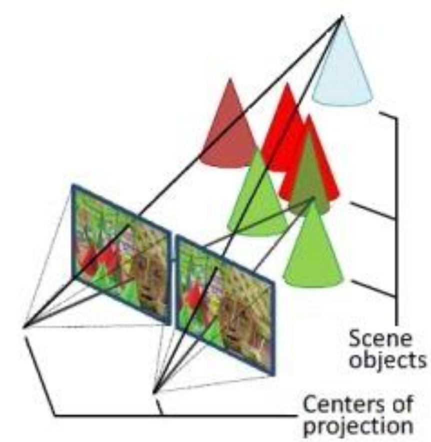
[Newman 2005]

- Recognize a previously landmark
- Typically SLAM will drift, after a long drive the position and landmarks will be off
- Once an previously located object is seen
  - Make the correction
  - Propagate the correction back
  - Uncertainties collapse
    - With great powers comes great responsibility
    - Uncertainty is not a “bad” thing

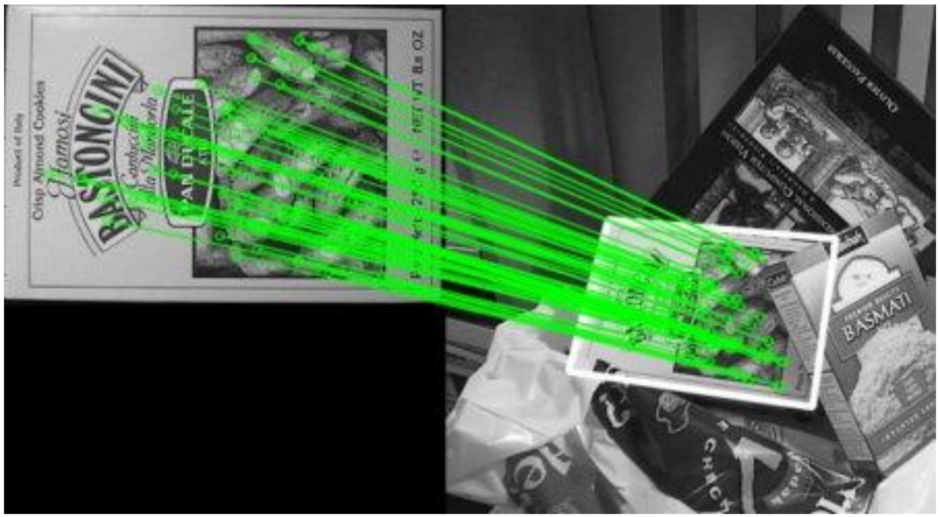
Visual Landmarks

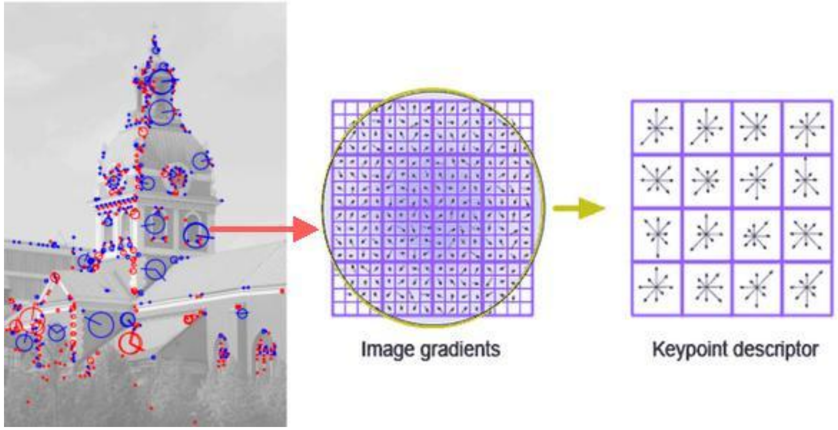
- Use camera images to detect landmarks
  - Single camera (mono) is similar to “bearing only” SLAM
  - Two cameras (stereo) will give you depth as well
  - Detect landmarks (aka “features”) in the image(s)
  - Build an appearance vector around the landmark to describe its “appearance”





Visual Landmarks

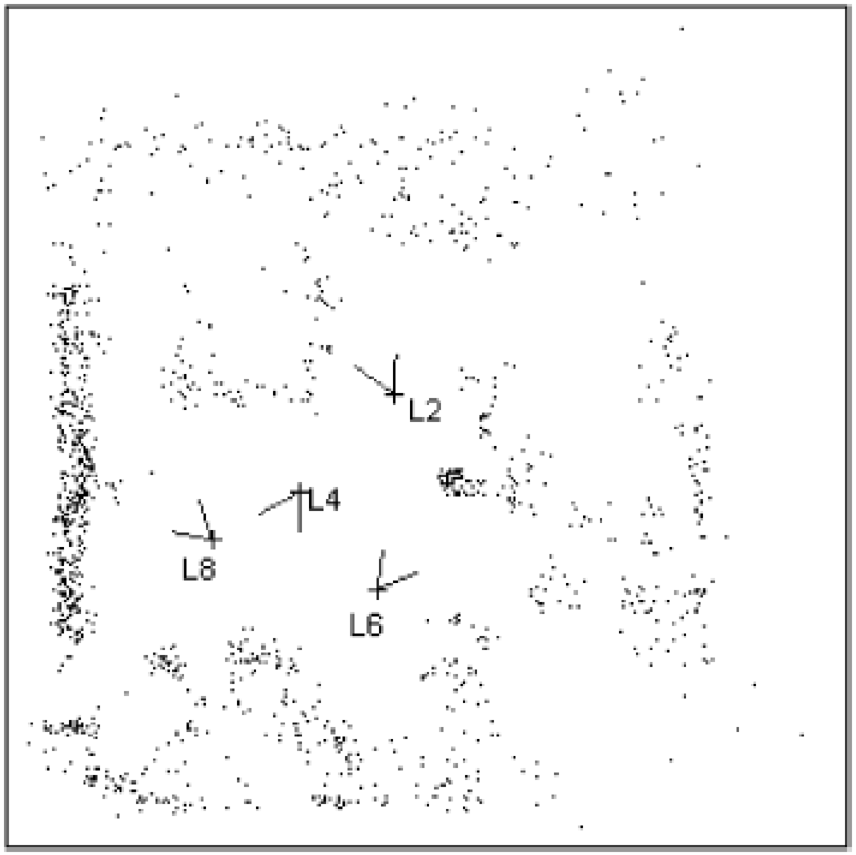




Landmarks have a scale and orientation

Extract landmarks and their descriptors from two images. Then match the descriptors

Visual SLAM



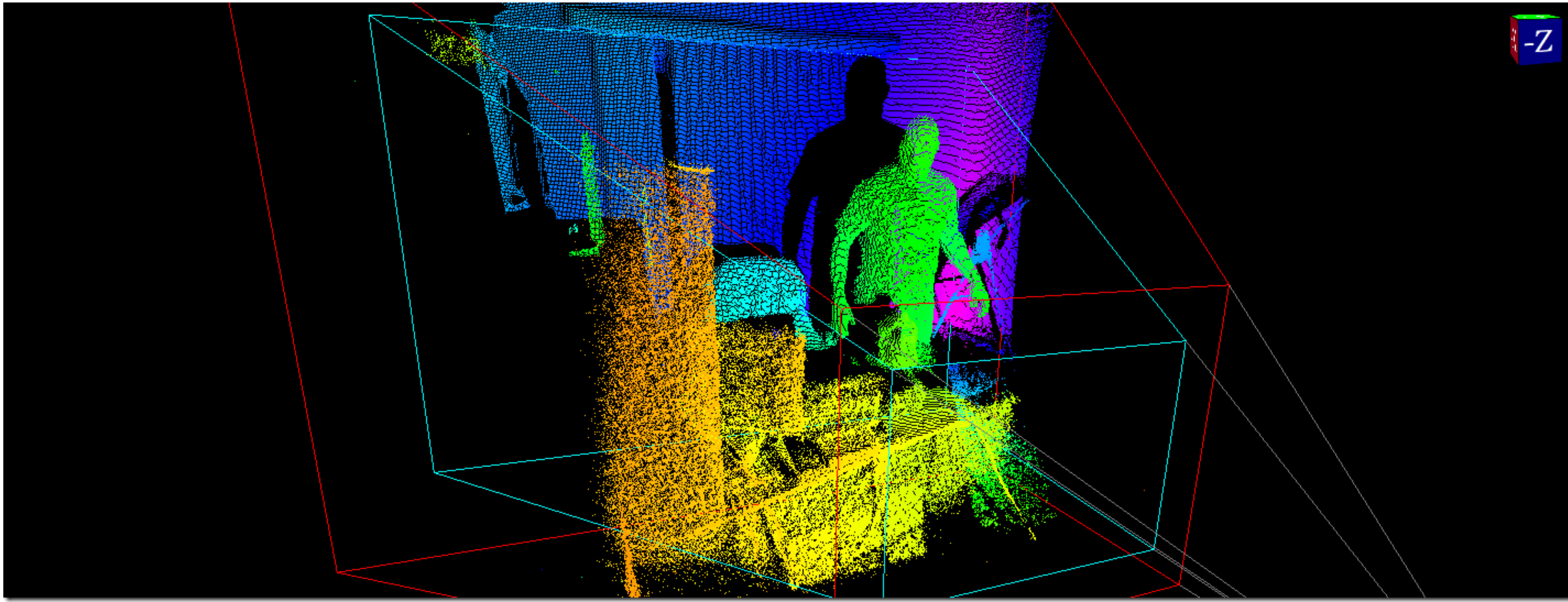
Bird's eye view of the 3D SIFT map

SIFT landmarks from stereo camera
Disparities are indicated by lines

Se, Lowe and Little 2005

RGBD SLAM
(Red Green Blue Depth)

- Use camera and depth
- Provide a dense point cloud



RGBD SLAM

- Build dense, 3D, colored maps
- Lots of data so maps are usually small (small rooms or scanning objects)
- Use 3D data for localization
- Associate visual landmarks to 3D coordinates

Mapping Algorithms - Comparison

	SLAM (Kalman)	EM	ML*	FastSLAM
Output	Posterior	ML/MAP	ML/MAP	Posterior
Convergence	Strong	Weak?	No	Stong
Local minima	No	Yes	Yes	No
Real time	Yes	No	Yes	Yes
Odom. Error	Unbounded	Unbounded	Unbounded	Unbounded
Sensor Noise	Gaussian	Any	Any	Any
# Features	10^3	∞	∞	$>10^3$
Feature uniq	Yes	No	~No	Yes
Raw data	No	Yes	Yes	No

Localization and Mapping Summary

- There are several methods for localization and mapping; two dominant are
  - Kalman filter
    - fast and efficient; very well understood
    - local convergence
    - strong assumptions
  - Hypothesis-based methods (particle filters/Monte Carlo methods)
    - not as fast or efficient; not as well understood
    - global convergence
    - very weak assumptions
- The best methods today are hybrids
  - use hypotheses as necessary
  - use KF-like techniques whenever possible
- The largest revolution in mapping and localization has been data:"It's all in the likelihood function"
  - laser scanners have really revolutionized the trade
  - vision is next?