Robotics
Spring 2023

Abhishek Gupta
TAs: Yi Li, Srivatsa GS

Recap: Course Overview

- Filtering/Smoothing
- Localization
- Mapping
- SLAM

- Search
- Motion Planning
- TrajOpt
- Stability/Certification

- MDPs and RL
- Imitation Learning
- Solving POMDPs

Lecture Outline

Unscented Kalman Filter

↓

Discrete Bayesian Filters

↓

Particle Filters

Recap: EKF

Initial Prior

$p(x_0)$

Linearize dynamics

$$x_{t+1} = g(x_t, u_t) + \epsilon_t \approx g(\mu_t, u_t) + \frac{\partial g(x_t, u_t)}{\partial x_t}\bigg|_{x_t=\mu_t} (x_t - \mu_t) + \epsilon_t$$

Dynamics/Prediction
(given some u)

Estimate $\overline{Bel}(x_t)$

$$p(x_{t+1}|z_{0:t}, u_{0:t}) \sim \mathcal{N}(g(\mu_t, u_t), G\Sigma_{t|0:t}G^T + Q_t)$$

Linearize measurement

$$z_t = h(x_t) + \delta_t \approx h(\bar{\mu}_t) + \frac{\partial h(x_t)}{\partial x_t}\bigg|_{x_t=\bar{\mu}_t} (x_t - \bar{\mu}_t) + \delta_t$$

Measurement/Correction
(given some z)

Estimate $Bel(x_t)$

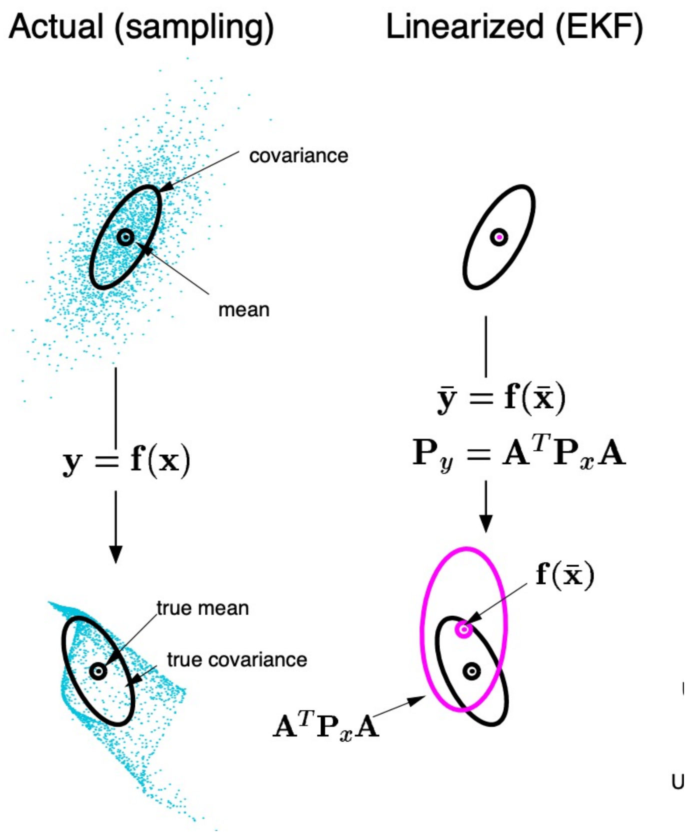
$$p(x_{t+1}|z_{0:t+1}, u_{0:t}) = \mathcal{N}(\mu_{t+1|0:t} + K_{t+1}(z_{t+1} - h(\bar{\mu}_t), (I - K_{t+1}H)\Sigma_{t+1|0:t})$$

When does the EKF struggle?

- With discontinuous dynamics, the linearization will not be valid
- For very non-linear functions, the first order Taylor approximation is poor
- The EKF can drift over time because of growing linearization errors
- Jacobian may be very expensive to compute and invert

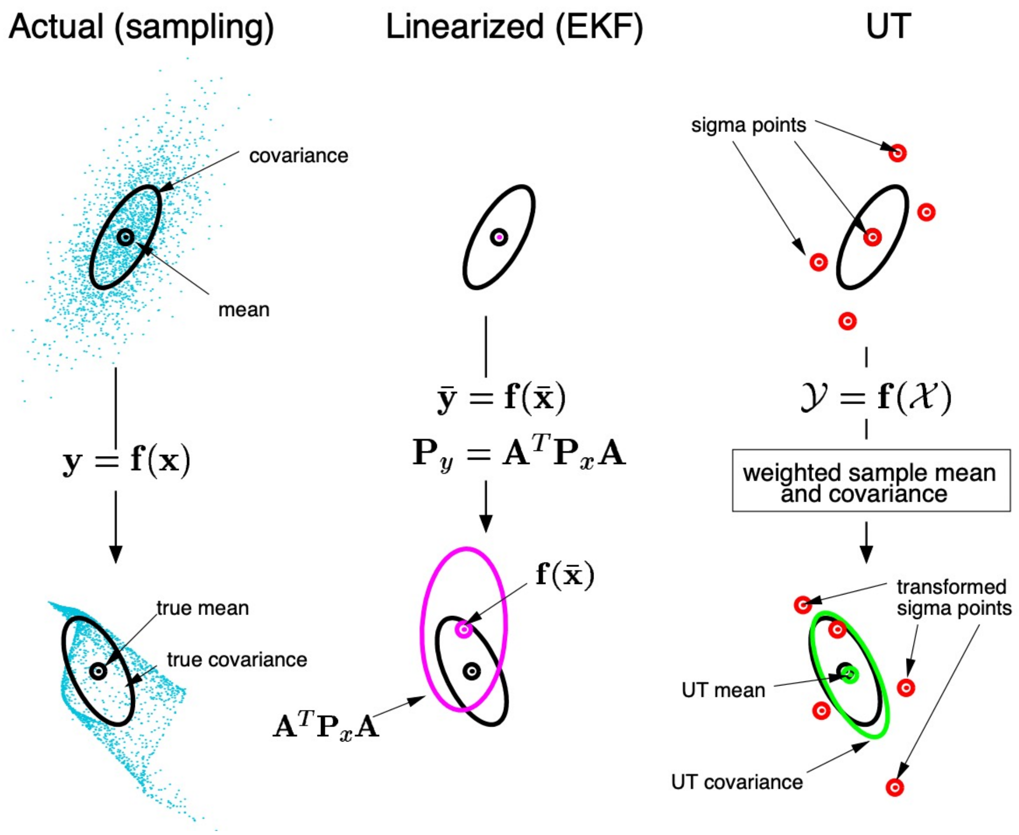
How can we achieve closer approximations?

- Extended Kalman filters first linearize then send through Gaussian, can be quite poor when the dynamics/measurements are quite non-linear



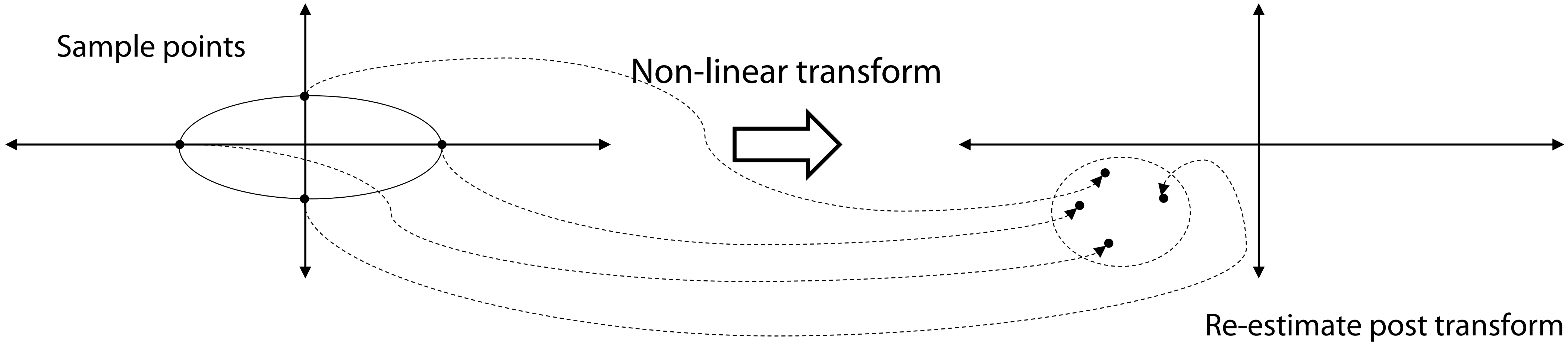
How can we achieve closer approximations?

- Extended Kalman filters first linearize then send through Gaussian, can be quite poor when the dynamics/measurements are quite non-linear



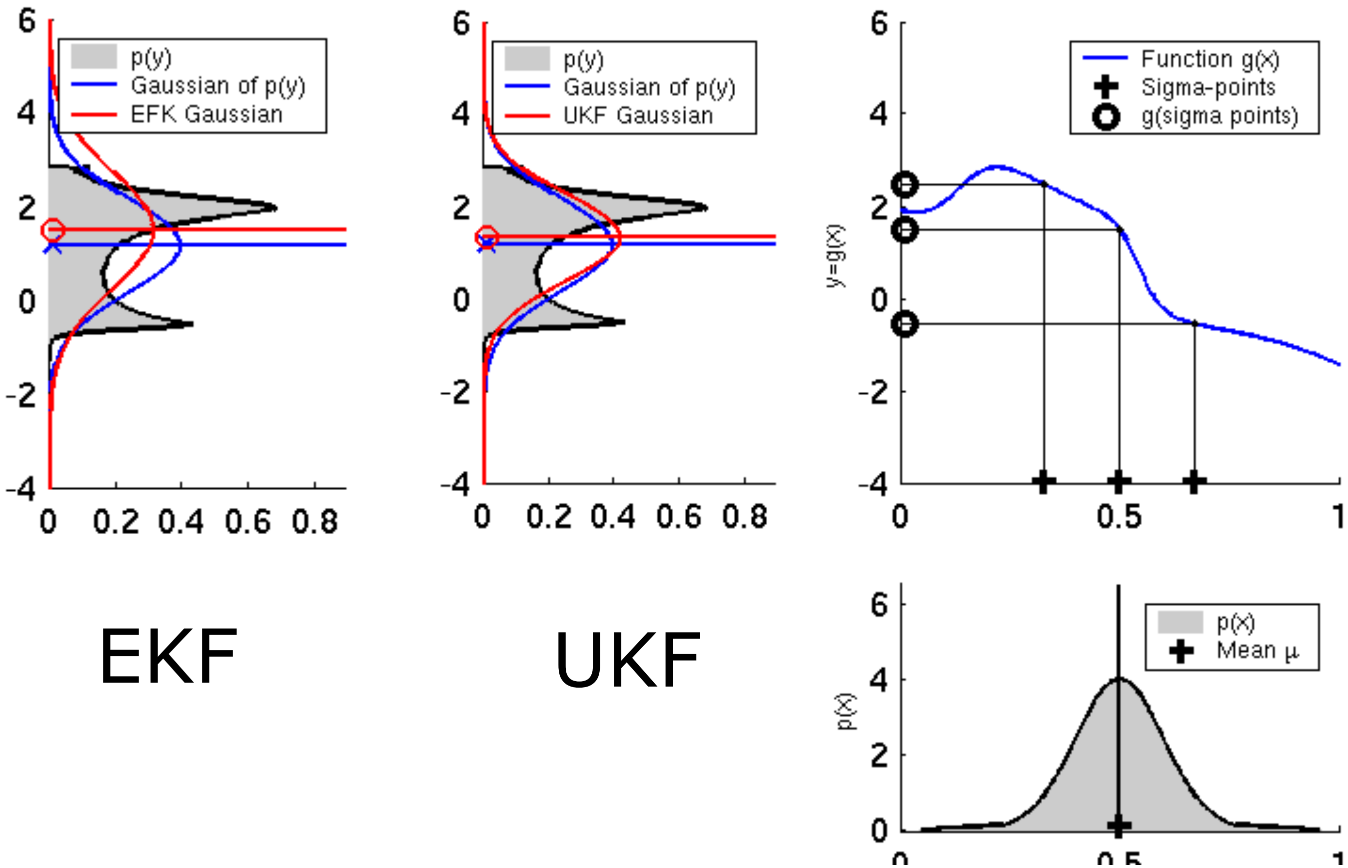
How can we achieve closer approximations?

- Idea: Rather than linearizing first and then propagate, propagate through non-linear transform and re-estimate Gaussian

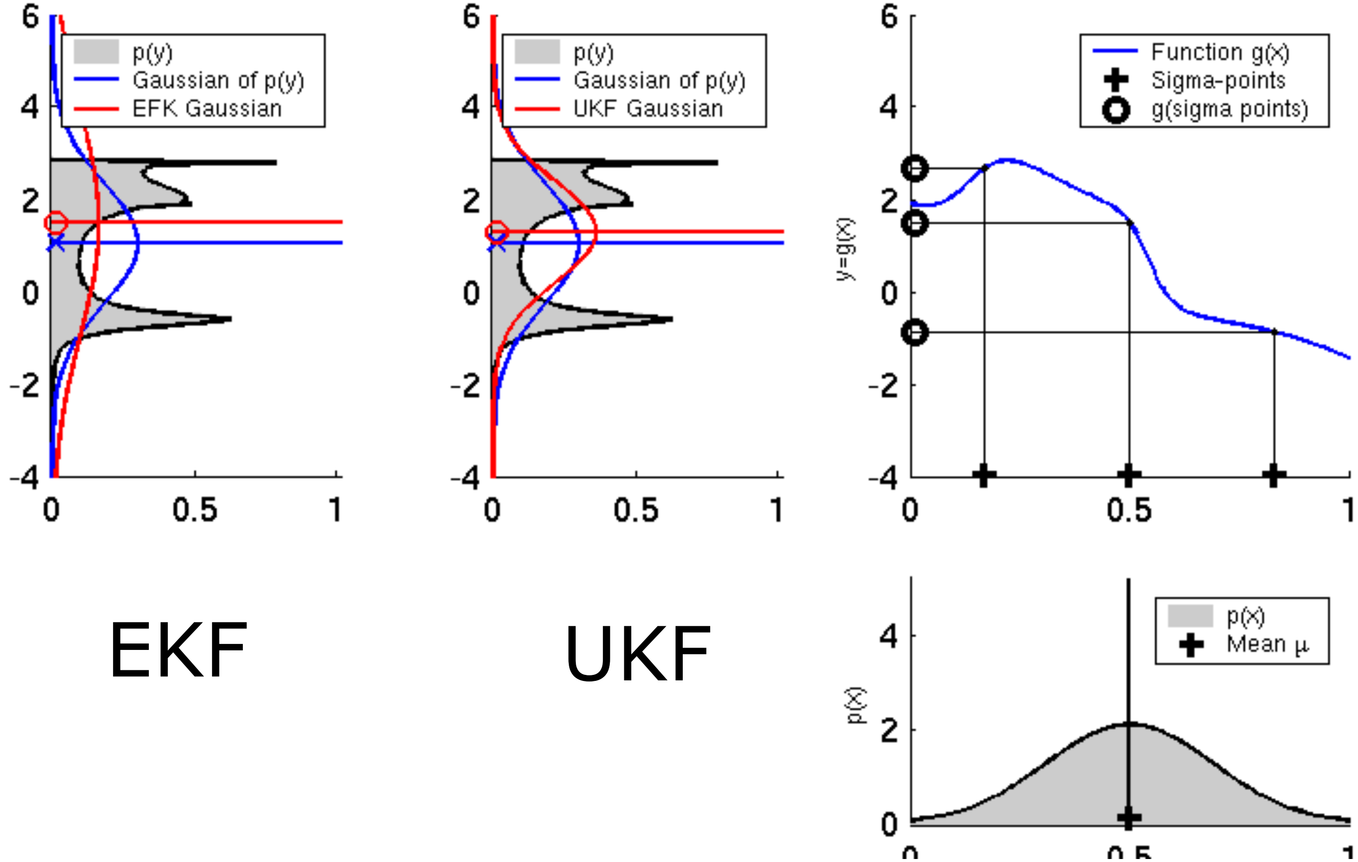


- Ensure that first and second moments (mean and covariance) match as closely as possible on re-estimation

Linearization via Unscented Transform



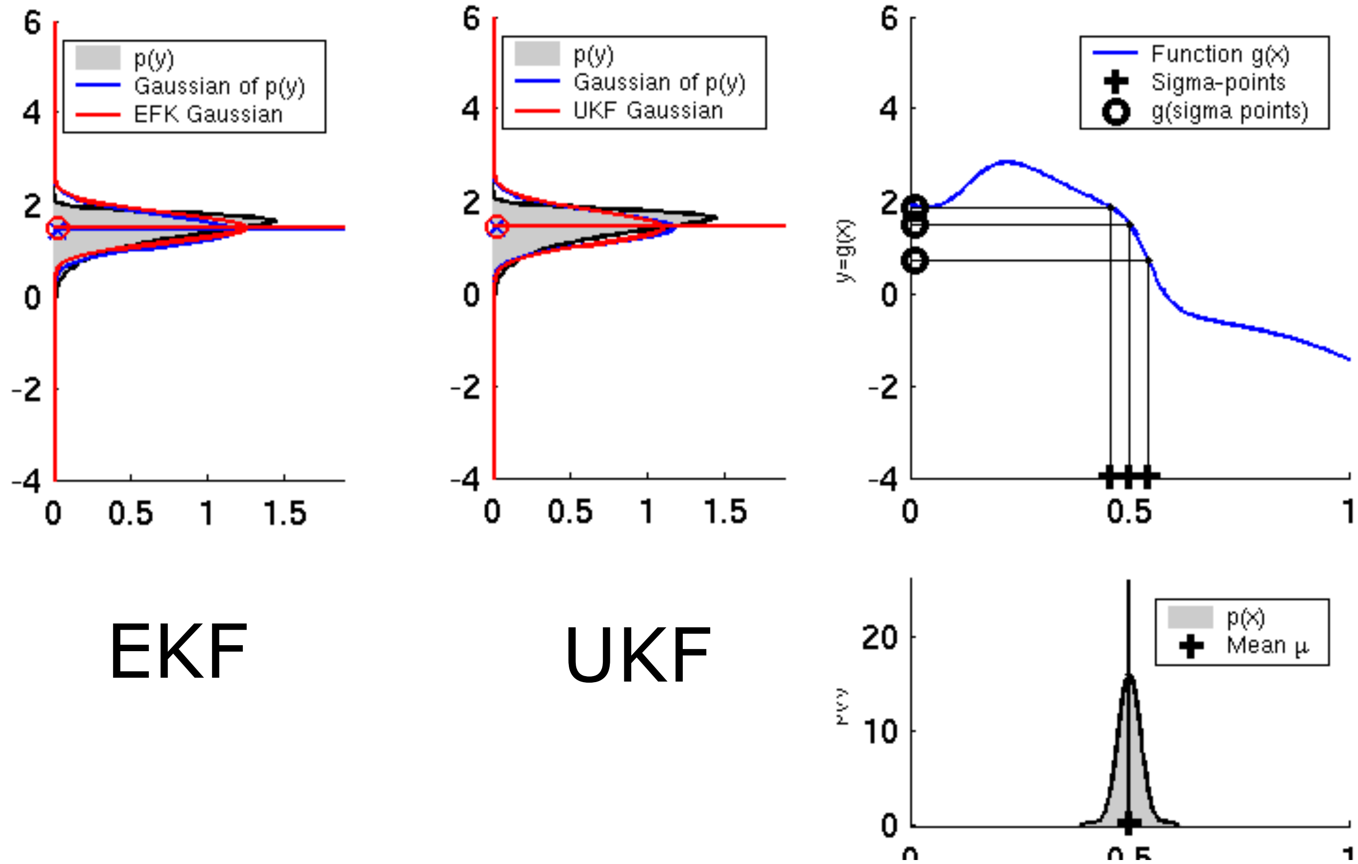
UKF Sigma-Point Estimate (2)



EKF

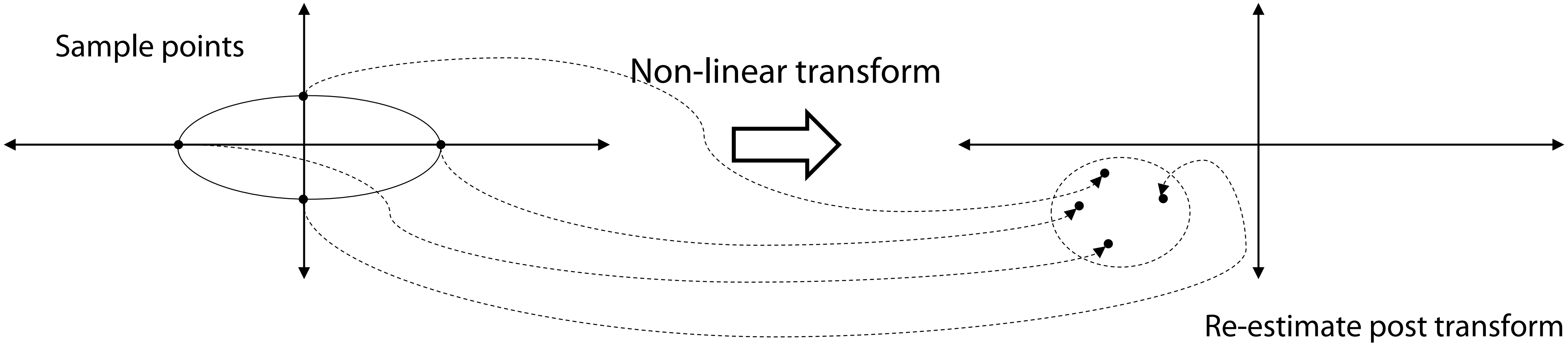
UKF

UKF Sigma-Point Estimate (3)



How can we achieve closer approximations?

- Idea: Rather than linearizing first and then propagate, propagate through non-linear transform and re-estimate Gaussian



- Question 1: What points should we send through non linearity?
- Question 2: How should we restimate the means and covariances?
- Question 3: Why can this be better than the EKF?

Sigma Points

- Question 1: What points should we send through non linearity?
  - Choose minimal points (2N + 1 to send through non-linearity to match 1,2 moments of a Gaussian

Sigma points

$$\chi^0 = \mu$$

$$\chi^i = \mu \pm \left(\sqrt{(n+\lambda)\Sigma}\right)_i$$

Weights

$$w_m^0 = \frac{\lambda}{n+\lambda} \qquad w_c^0 = \frac{\lambda}{n+\lambda}$$

$$w_m^i = w_c^i = \frac{1}{2(n+\lambda)} \qquad \text{for } i = 1,\ldots,2n$$

- What is a matrix square root?

$$L = \sqrt{\Sigma} \quad \text{if} \quad LL^T = \Sigma$$

- Why these points → they ensure that the moments match. Not a unique choice!

Unscented Transform

- Question 2: How should we re-estimate the means and covariances?

Sigma points

$$\chi^0 = \mu$$

$$\chi^i = \mu \pm \left(\sqrt{(n+\lambda)\Sigma}\right)_i$$

Weights

$$w_m^0 = \frac{\lambda}{n+\lambda} \qquad w_c^0 = \frac{\lambda}{n+\lambda}$$

$$w_m^i = w_c^i = \frac{1}{2(n+\lambda)} \qquad \text{for } i = 1,\ldots,2n$$

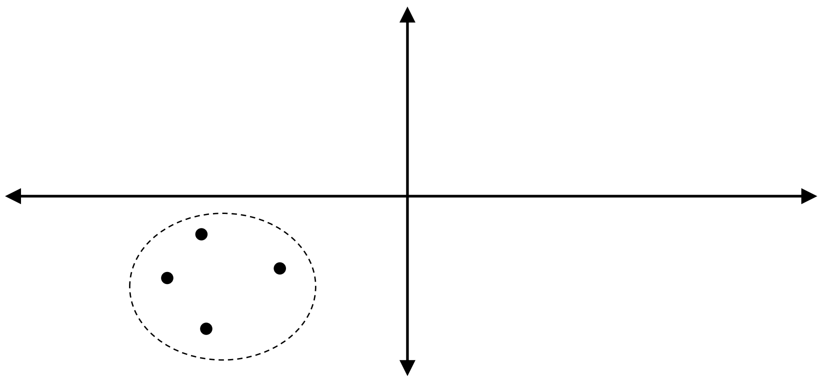
Pass sigma points through nonlinear function

$$\psi^i = g(\chi^i)$$

Recover mean and covariance

$$\mu' = \sum_{i=0}^{2n} w_m^i \psi^i$$

$$\Sigma' = \sum_{i=0}^{2n} w_c^i (\psi^i - \mu)(\psi^i - \mu)^T$$



Re-estimate post transform

Why do these make sense?

Sigma points

$$\chi^0 = \mu$$

$$\chi^i = \mu \pm \left(\sqrt{(n+\lambda)\Sigma}\right)_i$$

Weights

$$w_m^0 = \frac{\lambda}{n+\lambda} \qquad w_c^0 = \frac{\lambda}{n+\lambda}$$

$$w_m^i = w_c^i = \frac{1}{2(n+\lambda)} \qquad \text{for } i = 1,\ldots,2n$$

Filtering with the Unscented Transform

- Given the tool of the unscented transform, let us revisit the nonlinear filter

$$p(x_0)$$

Dynamics

Estimate $\overline{Bel}(x_t)$

$$\overline{Bel}(x_t) = \int p(x_t|x_{t-1}, u_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

→ Directly use unscented transform for dynamics

Measurement

Estimate $Bel(x_t)$

$$Bel(x_t) = \eta p(z_t|x_t)\overline{Bel}(x_t) = \frac{p(z_t|x_t)\overline{Bel}(x_t)}{\sum_{x'} p(z_t|x_t = x')\overline{Bel}(x_t = x')}$$

→ Estimate empirical covariance matrix with unscented transform and do Kalman filter

Unscented KF Dynamics Step

- Sample Sigma points given current belief, send them through non-linear dynamics
- Re-estimate the post-update belief using the unscented transform

$$\chi^0 = \mu_{t|0:t}$$

$$\chi^i = \mu_{t|0:t} \pm (\sqrt{(n+\lambda)\Sigma_{t|0:t}})_i$$

$$w_m^0 = \frac{\lambda}{n+\lambda} \qquad w_c^0 = \frac{\lambda}{n+\lambda}$$

$$w_m^i = w_c^i = \frac{1}{2(n+\lambda)} \qquad \text{for } i = 1,\ldots,2n$$

Pass sigma points through nonlinear function $\psi^i = g(\chi^i, u_t)$

$$\mu_{t+1|0:t} = \sum_{i=0}^{2n} w_m^i \psi^i$$

$$\Sigma_{t+1|0:t} = \sum_{i=0}^{2n} w_c^i (\psi^i - \mu_{t+1|0:t})(\psi^i - \mu_{t+1|0:t})^T + Q$$

Unscented KF Measurement Step

- More tricky because now C/H is not known! How to compute Kalman gain?

$$K_{t+1} = \Sigma_{t+1|0:t} C^T (C \Sigma_{t+1|0:t} C^T + R_{t+1})^{-1}$$

Cross covariance under forward transform

Covariance under forward transform

Remember from earlier

Diagonal Covariance

$$\begin{aligned} \Sigma_{t+1|0:t} &= \mathbb{E}\left[(X_{t+1|0:t} - \mu_{t+1|0:t})(X_{t+1|0:t} - \mu_{t+1|0:t})^T\right] \\ &= \mathbb{E}\left[(AX_{t|0:t} + Bu_t + \epsilon_t - A\mu_{t|0:t} - Bu_t)(AX_{t|0:t} + Bu_t + \epsilon_t - A\mu_{t|0:t} - Bu_t)^T\right] \\ &= A\mathbb{E}\left[(X_{t|0:t} - \mu_{t|0:t})(X_{t|0:t} - \mu_{t|0:t})^T\right] A^T + Q_t \\ &= A\Sigma_{t|0:t} A^T + Q_t \end{aligned}$$

Cross Covariance

$$\Sigma_{t,t+1|0:t} = \mathbb{E}\left[(X_{t|0:t} - \mu_{t|0:t})(X_{t+1|0:t} - \mu_{t+1|0:t})^T\right]$$

$$\Sigma_{t,t+1|0:t} = \Sigma_{t|0:t} A^T$$

Unscented KF Measurement Step

- More tricky because now C/H is not known! How to compute Kalman gain?

$$K_{t+1} = \Sigma_{t+1|0:t} C^T (C \Sigma_{t+1|0:t} C^T + R_{t+1})^{-1}$$

Cross covariance under forward transform

Covariance under forward transform

Send sigma points through non-linear measurement model $\bar{\psi}^i = h(x_t)$

$$\bar{z} = \sum_{i=0}^{2n} w_m^i \bar{\psi}^i \qquad S = \sum_{i=0}^{2n} w_c^i (\bar{\psi}^i - \bar{z})(\bar{\psi}^i - \bar{z})^T \qquad T = \sum_{i=0}^{2n} w_c^i (\psi^i - \mu_{t+1|0:t})(\bar{\psi}^i - \bar{z})^T$$

$$K_{t+1} = T S^{-1}$$

Then use standard KF measurement update

Cross covariance

Covariance

UKF Pseudocode

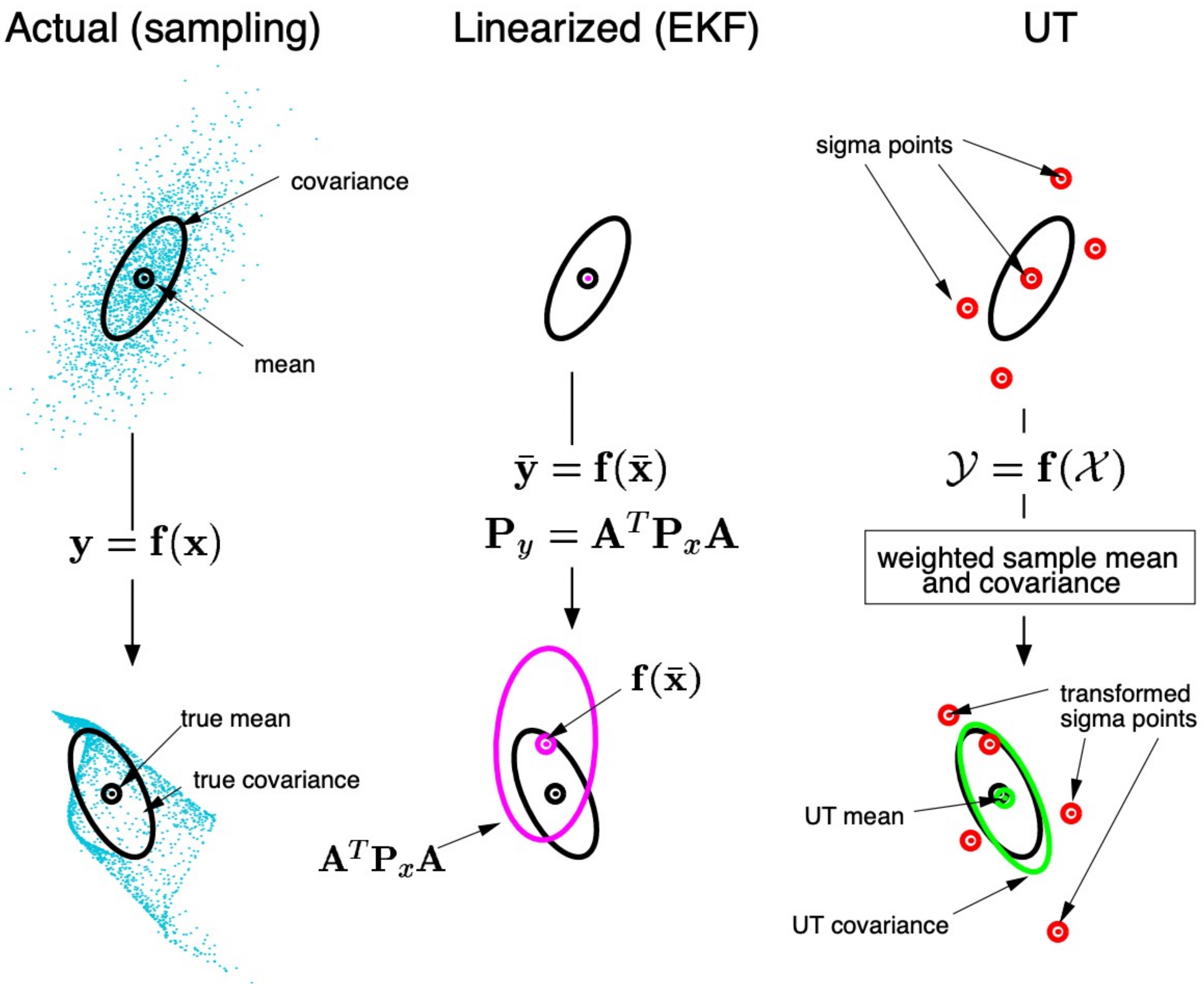
def Unscented_Kalman_filter($\mu_{t|0:t}$, $\Sigma_{t|0:t}$, u_t, z_{t+1}):

1. Dynamics
   1. Sample Sigma Points from $\mathcal{N}(\mu_{t|0:t}, \Sigma_{t|0:t})$
   2. Send them through $g(x_t, u_t)$
   3. Compute $\mu_{t+1|0:t}, \Sigma_{t+1|0:t}$ via UT
2. Measurement:
   1. Sample Sigma Points from $\mathcal{N}(\mu_{t+1|0:t}, \Sigma_{t+1|0:t})$
   2. Send them through $h(x_t)$
   3. Compute T, S as cross covariance and covariance
   4. Compute $K_{t+1} = TS^{-1}$
   5. Use standard KF updates
3. Return mean, cov ,

$$x_{t+1} = g(x_t, u_t) + \epsilon_t$$
$$z_t = h(x_t) + \delta_t$$
$$\epsilon_t \sim \mathcal{N}(0, Q)$$
$$\delta_t \sim \mathcal{N}(0, R)$$

Reminder of the model

How well does this do?



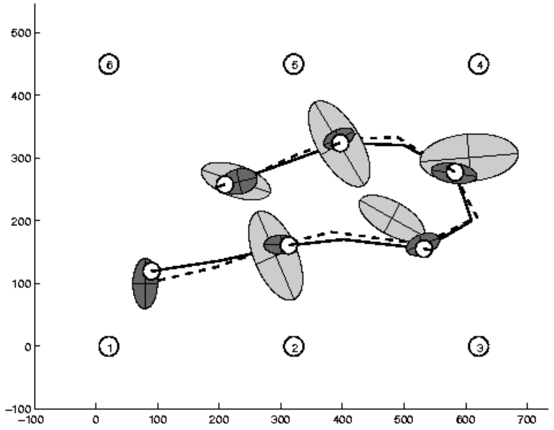
When/why is the UKF better than the EKF?

- EKF:
  - First linearize then propagate
  - Misses higher order terms

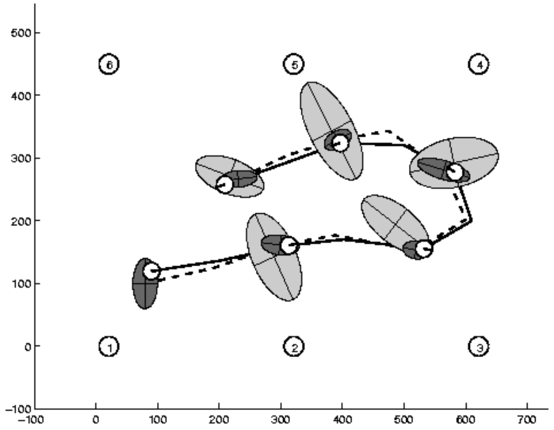
- UKF:
  - First propagate then linearize
  - Approximates the higher order terms as well

Approximately the same if sigma points are close to linearization point

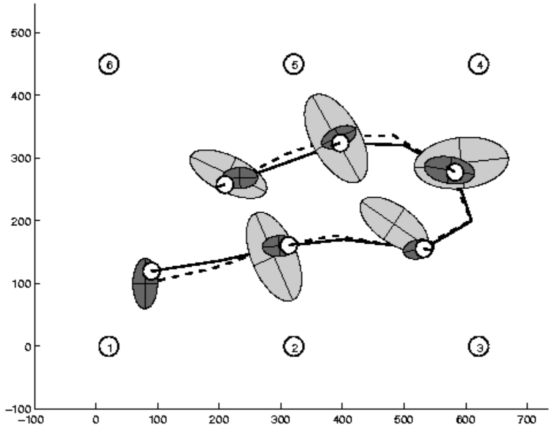
Estimation Sequence



EKF

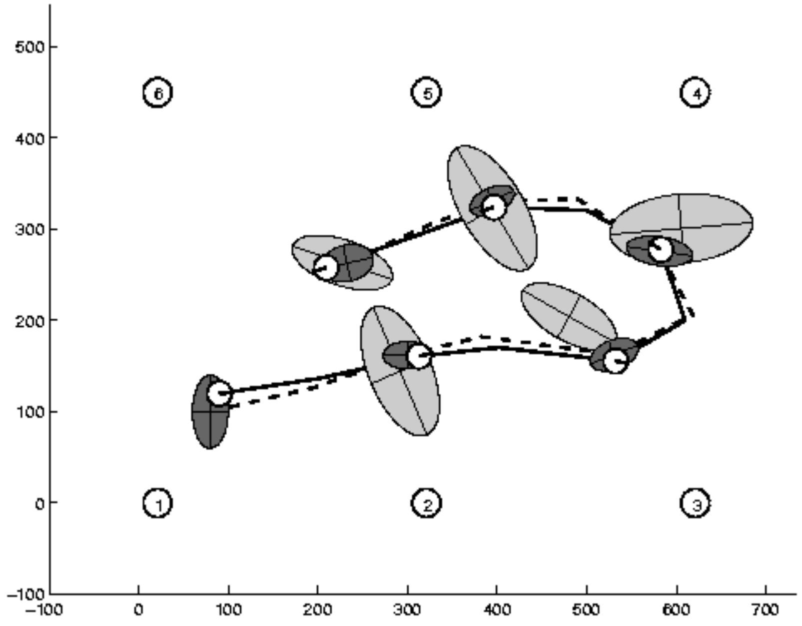


PF

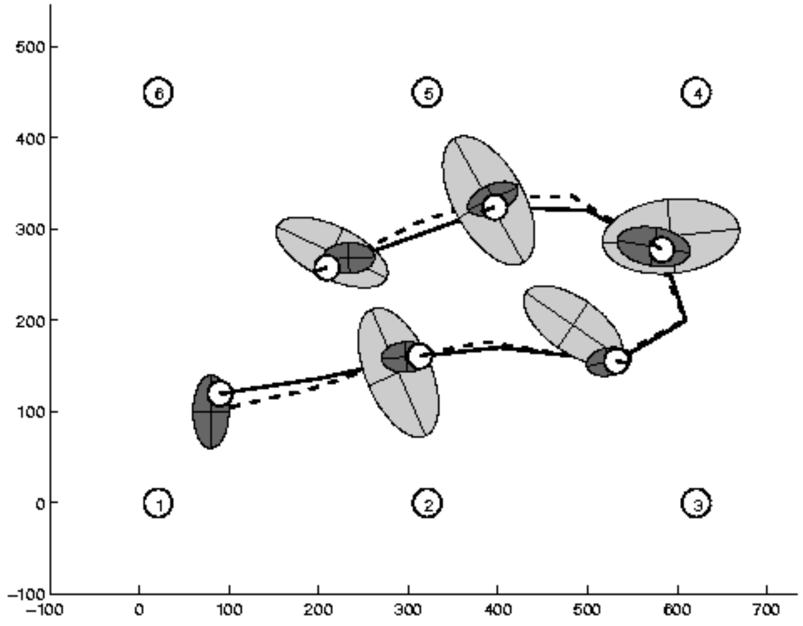


UKF

Estimation Sequence

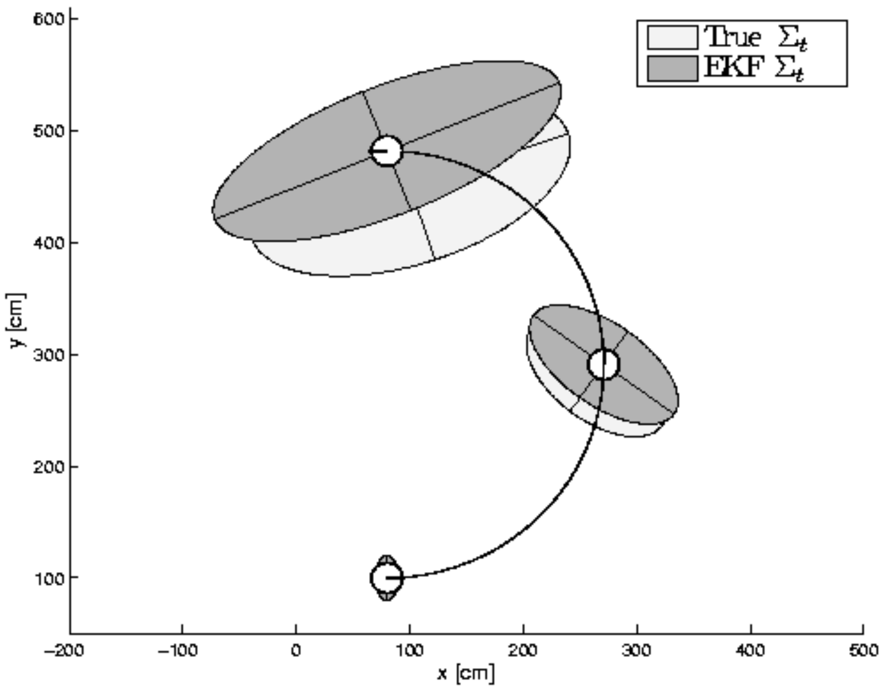


EKF

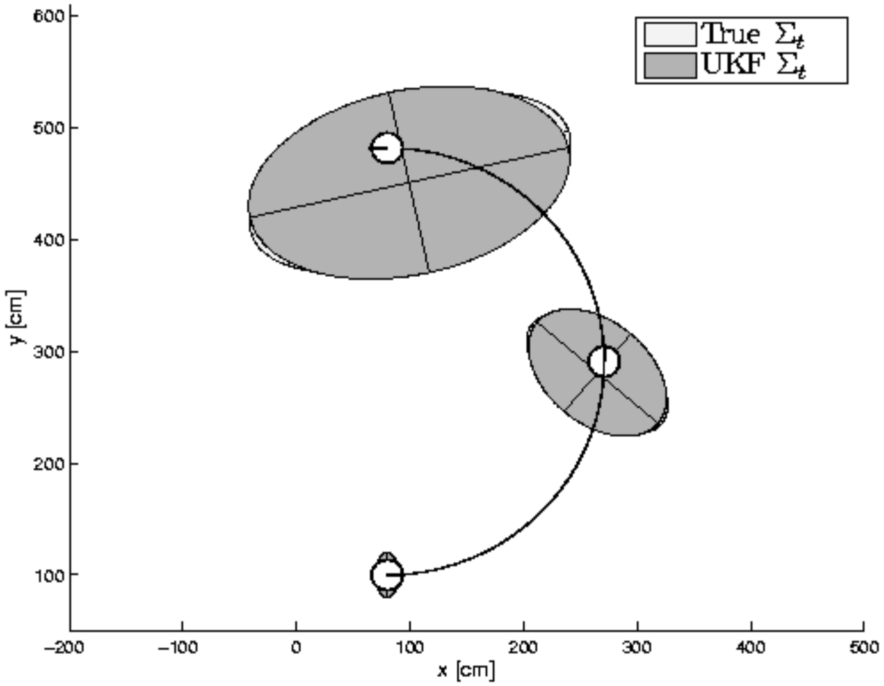


UKF

Prediction Quality

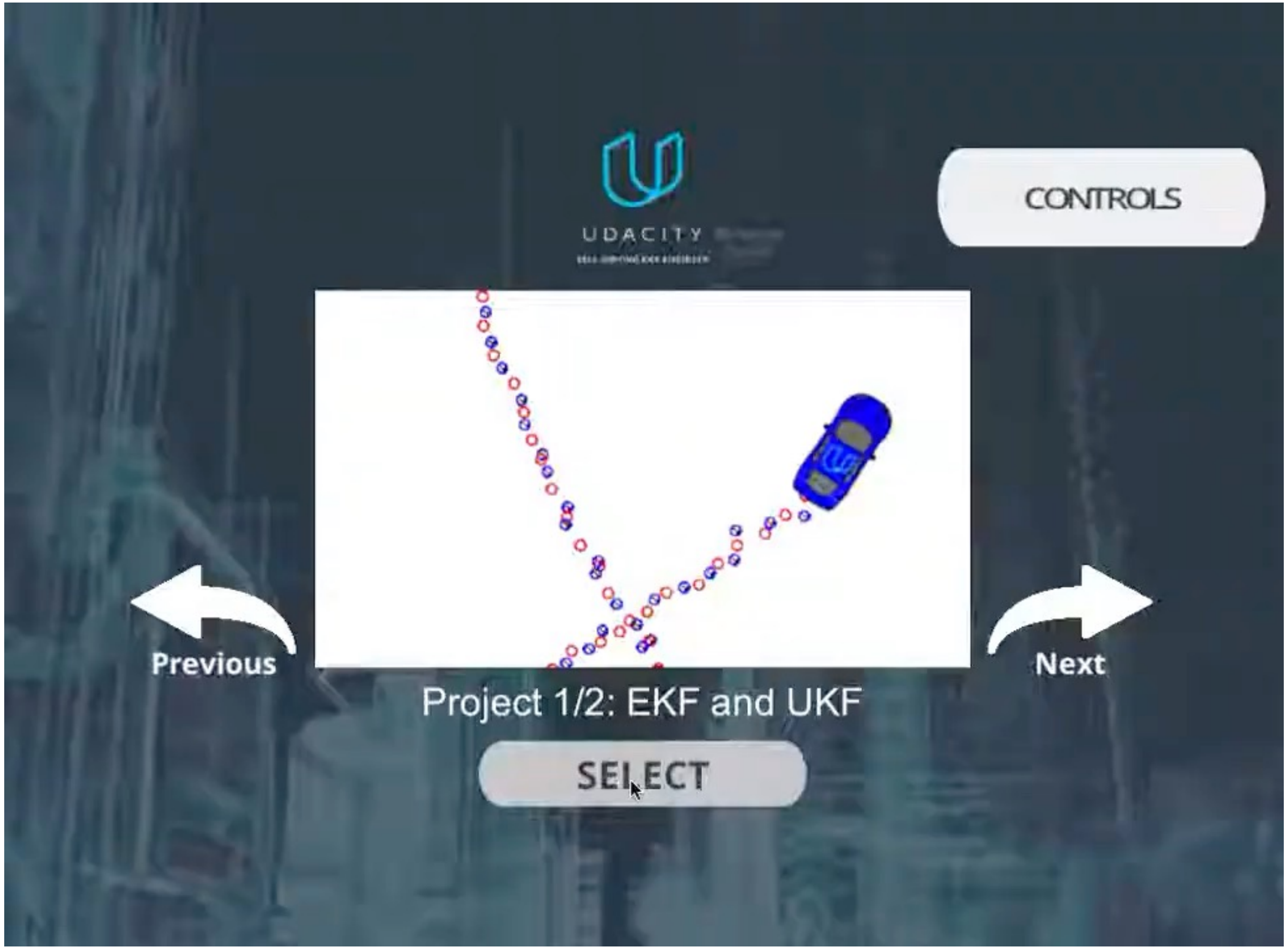


EKF



UKF

UKF in Action



UKF Summary

- Highly efficient: Same complexity as EKF, with a constant factor slower in typical practical applications
- Better linearization than EKF: Accurate in first two terms of Taylor expansion (EKF only first term)
- Derivative-free: No Jacobians needed
- Still not optimal!

Lecture Outline

Extended Kalman Filter

↓

Discrete Bayesian Filters

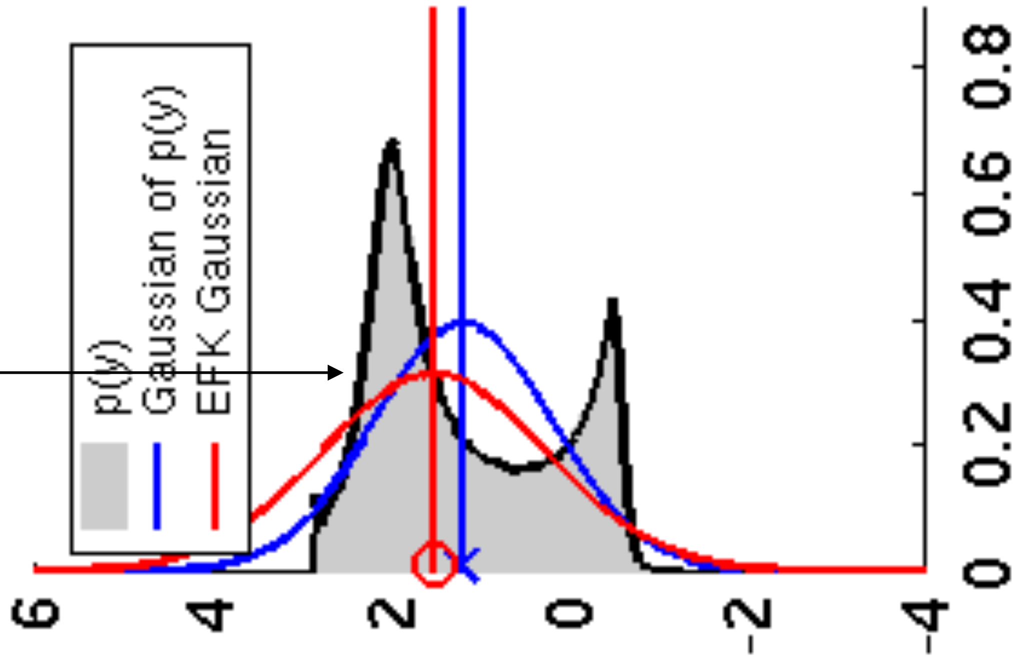
↓

Particle Filters

When do EKF/UKF fail?

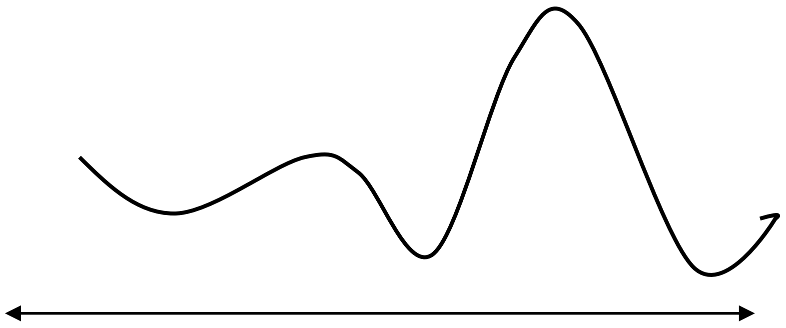
- Non-linear functions
- Non-Gaussian functions

How can we represent this better?

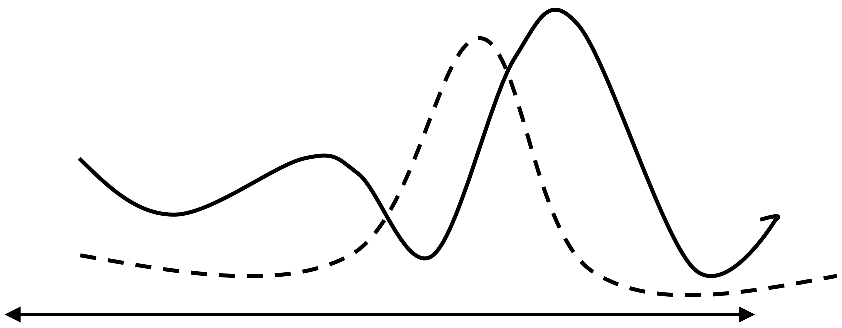


Multimodality in Probability Distributions

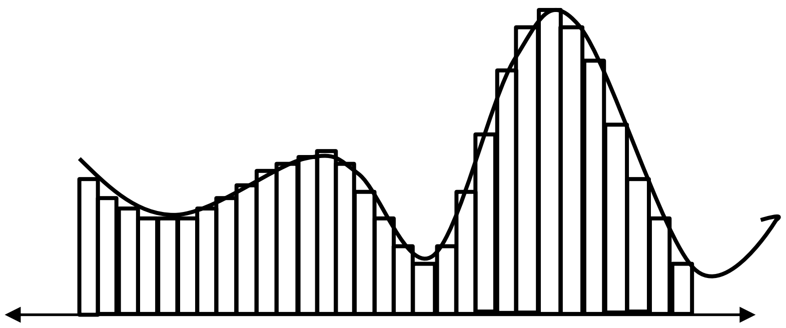
Target Distribution



Gaussian Approximation



Categorical Approximation



Can we leverage categorical distributions for filtering/localization?

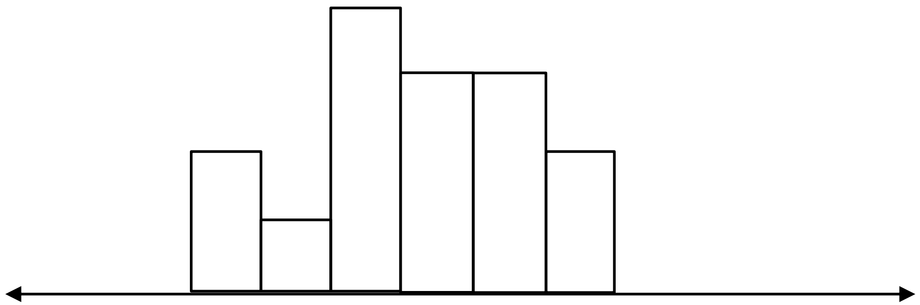
Idea 1: Discrete Bayes Filter

Remember the idea behind Bayesian filters

$$Bel(x_t) = P(x_t|u_{0:t-1}, z_{0:t})$$
$$= \eta\, p(z_t|x_t) \int P(x_t|u_{t-1}, x_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

We made these Gaussian

Why did we jump through all those hoops? → dealing with the integrals
What if the state were discrete?



Easily multimodal and tractable

All integrals are sums! Multimodality is not an issue

Idea 2: Histogram Filter

But the world is continuous, how can we apply this machinery?

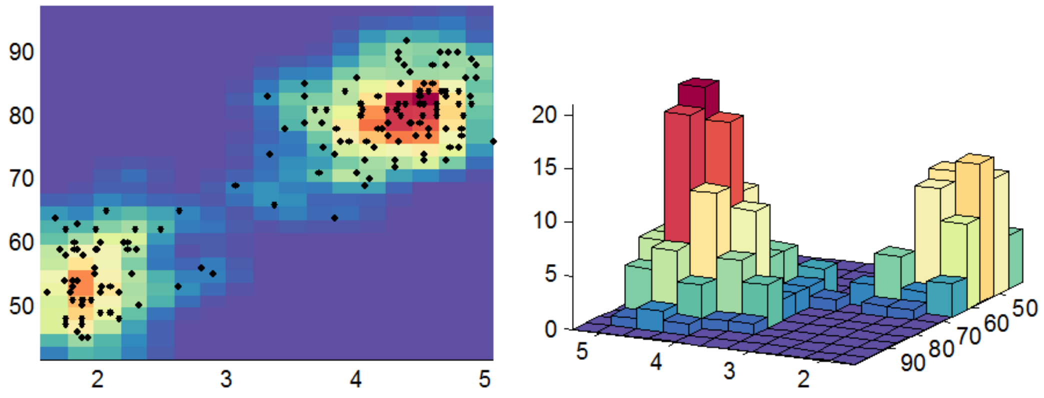
Just discretize!

Assumption – value is piecewise constant within a bin – use the mean

$$\hat{x}_{k,t} = |\mathbf{x}_{k,t}|^{-1} \int_{\mathbf{x}_{k,t}} x_t \, dx_t$$

$$p(z_t \mid \mathbf{x}_{k,t}) \approx p(z_t \mid \hat{x}_{k,t})$$

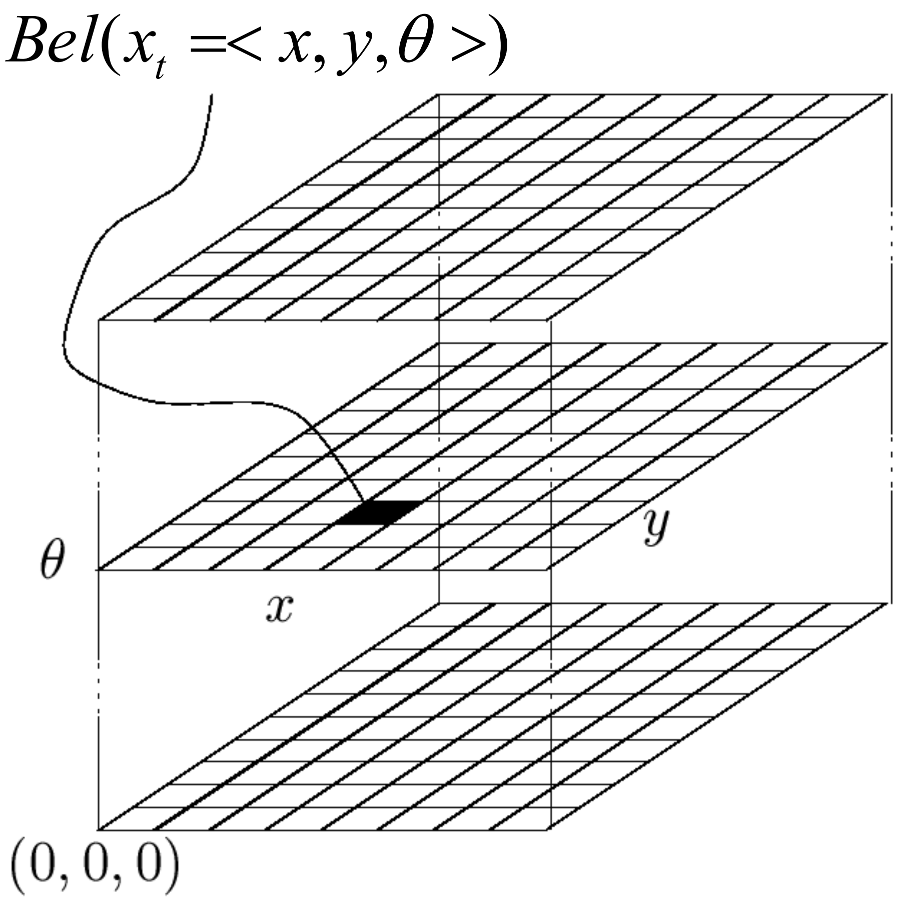
$$p(\mathbf{x}_{k,t} \mid u_t, \mathbf{x}_{i,t-1}) \approx \frac{\eta}{|\mathbf{x}_{k,t}|} p(\hat{x}_{k,t} \mid u_t, \hat{x}_{i,t-1})$$



Approximation errors drops with finer discretization

The number of states might blow up → more on this later

Why is this a reasonable assumption to make?

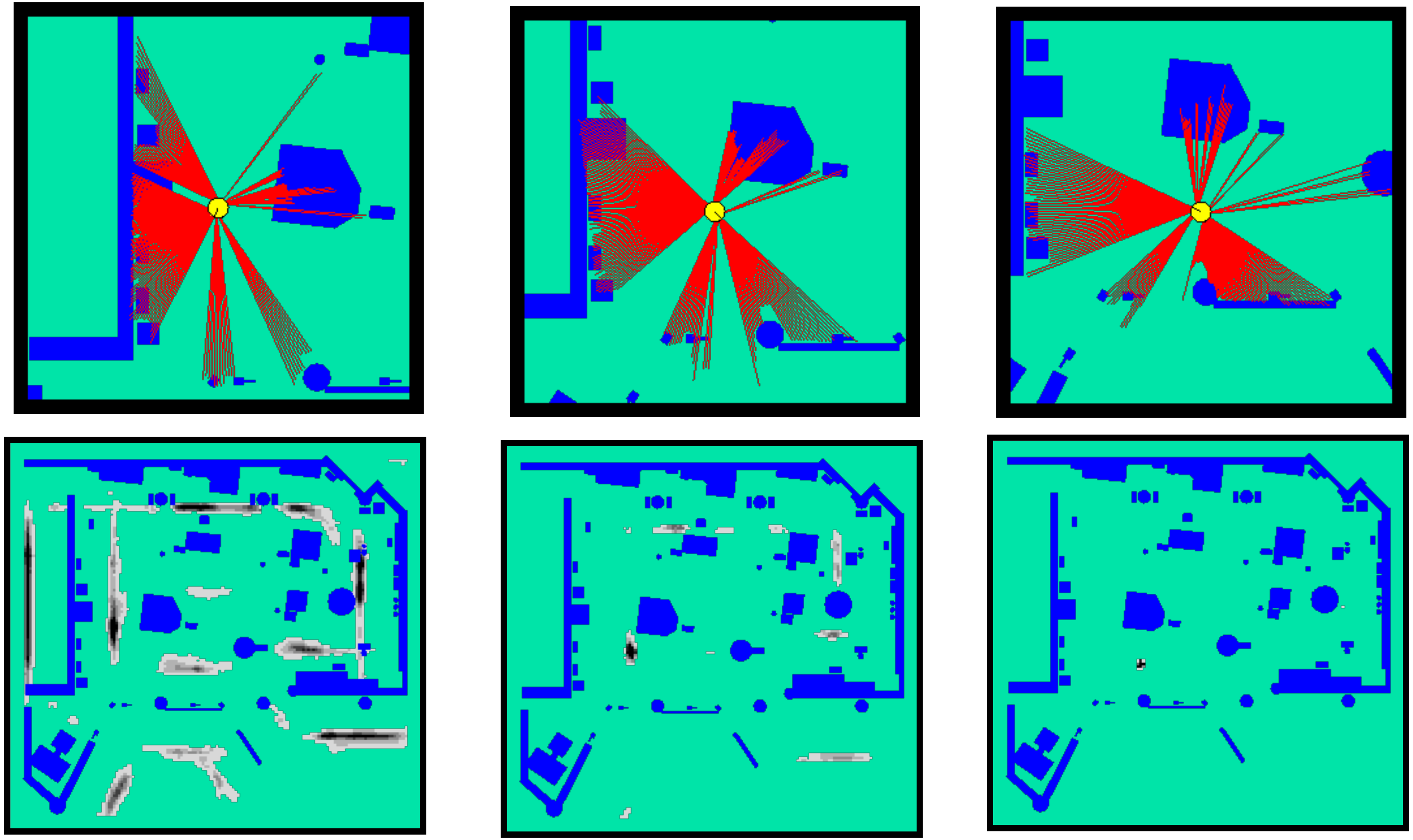


Assuming the value doesn't change significantly within a bin

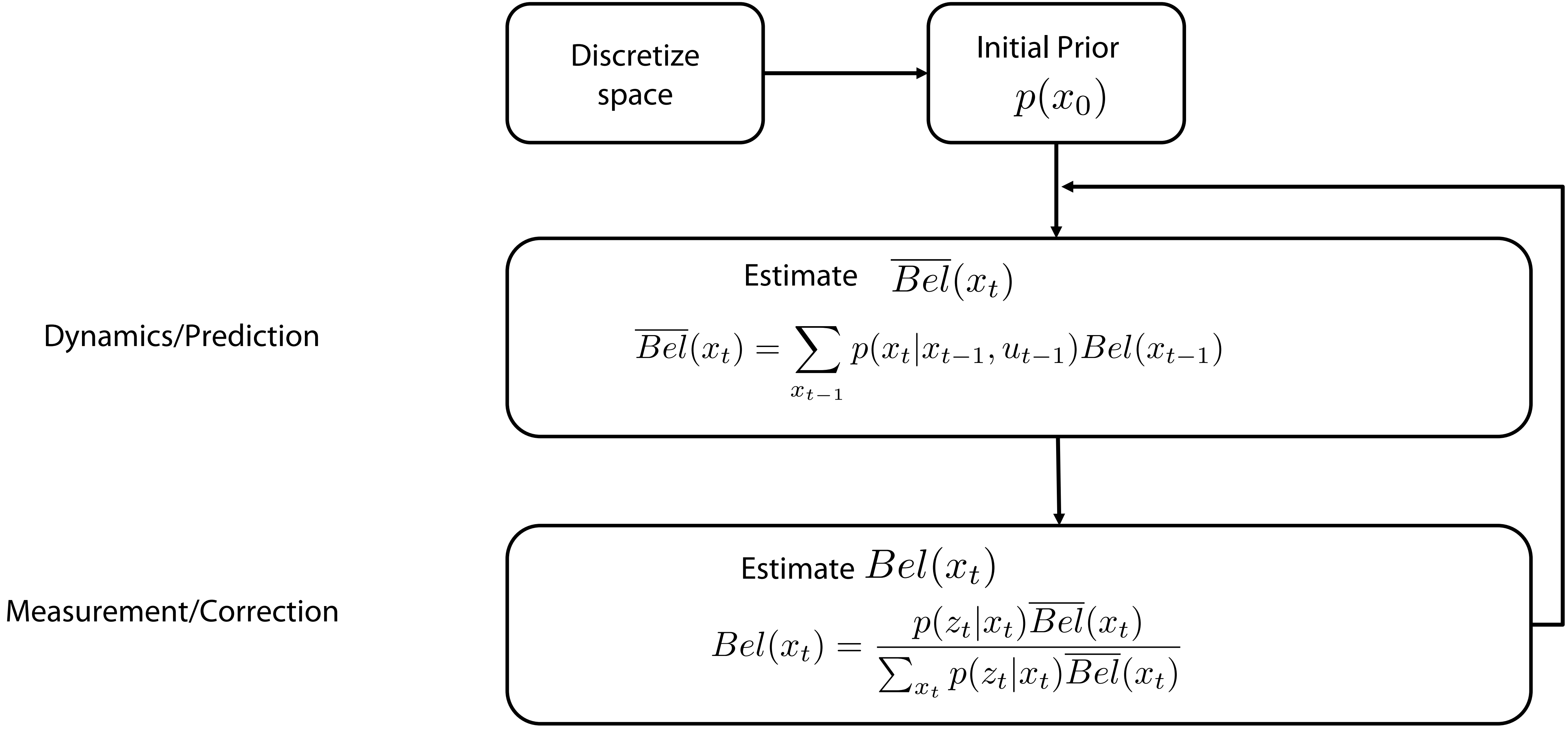
$$
\begin{aligned}
p(z_t \mid \mathbf{x}_{k,t}) &\approx p(z_t \mid \hat{x}_{k,t}) \\
p(\mathbf{x}_{k,t} \mid u_t, \mathbf{x}_{i,t-1}) &\approx \frac{\eta}{|\mathbf{x}_{k,t}|}\, p(\hat{x}_{k,t} \mid u_t, \hat{x}_{i,t-1})
\end{aligned}
$$

Using the mean is reasonable if the variance is bounded

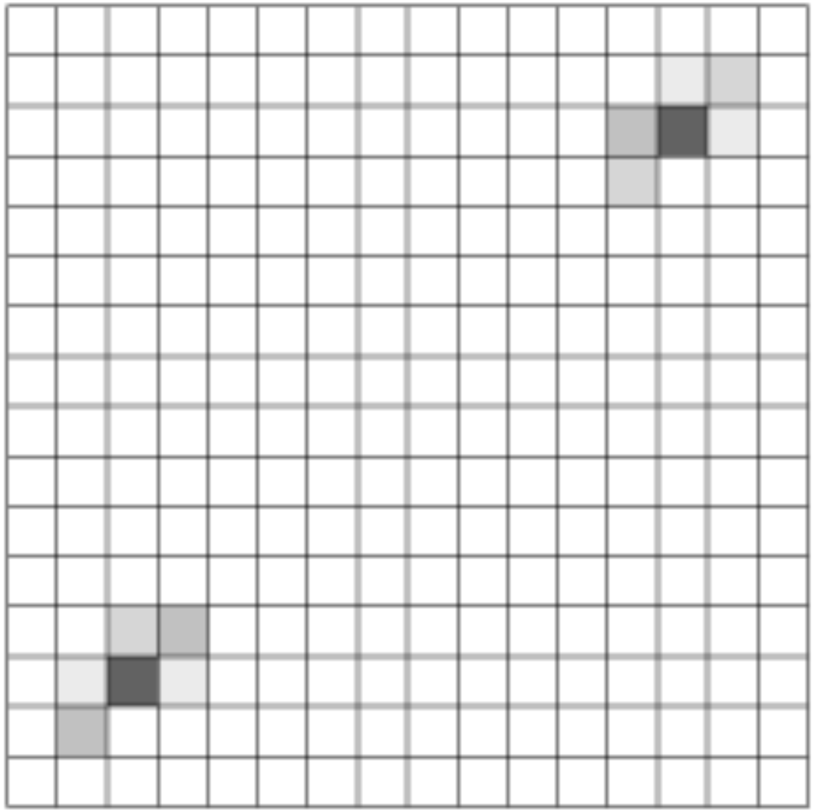
Grid-based Localization



Overall histogram filter algorithm



Challenges with Static Discretization



- Scales poorly with dimension:
  - Exponential bins in largely empty space
- Not adaptive as the posterior changes
- Unclear how to perform discretization

Lecture Outline

Unscented Kalman Filter

↓

Discrete Bayesian Filters

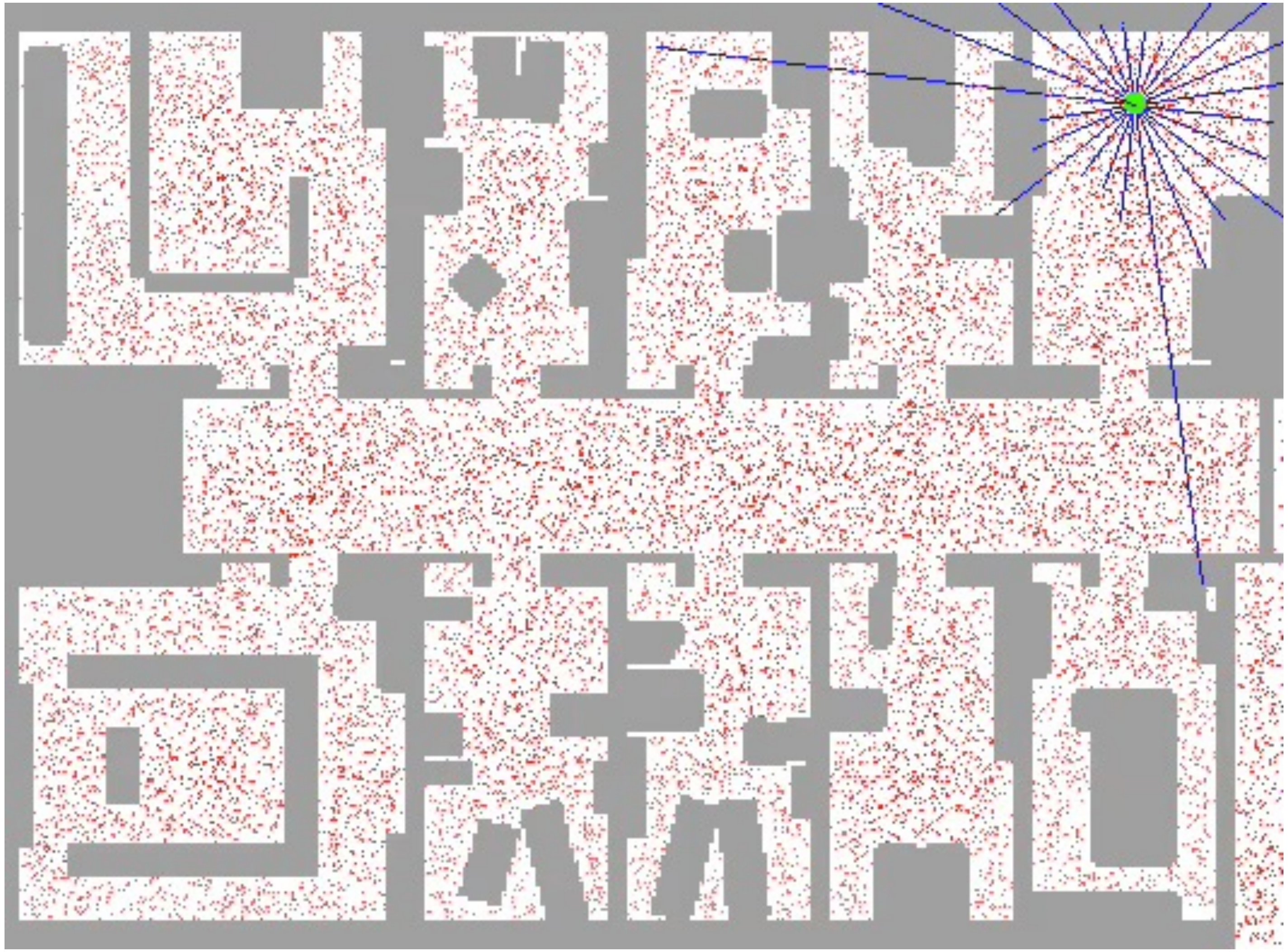
↓

Particle Filters

Particle Filters: Motivation

- So far, we discussed the
  - Kalman filter: Gaussian, linearization problems, discrete Bayes filters (eg histogram filters)
- Histogram filters are great but they waste space and are non adaptive
- Particle filters are a way to **efficiently** represent **non-Gaussian distributions adaptively**
- Basic principle
  - Set of state hypotheses (“particles”)
  - Survival-of-the-fittest

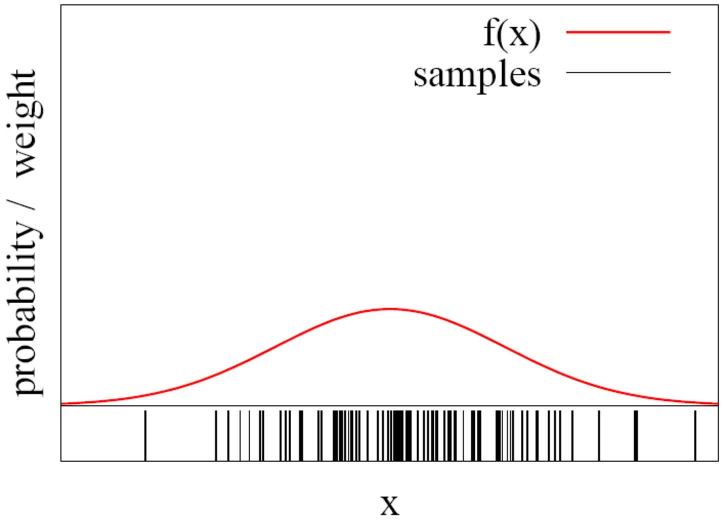
Sample-based Localization (sonar)

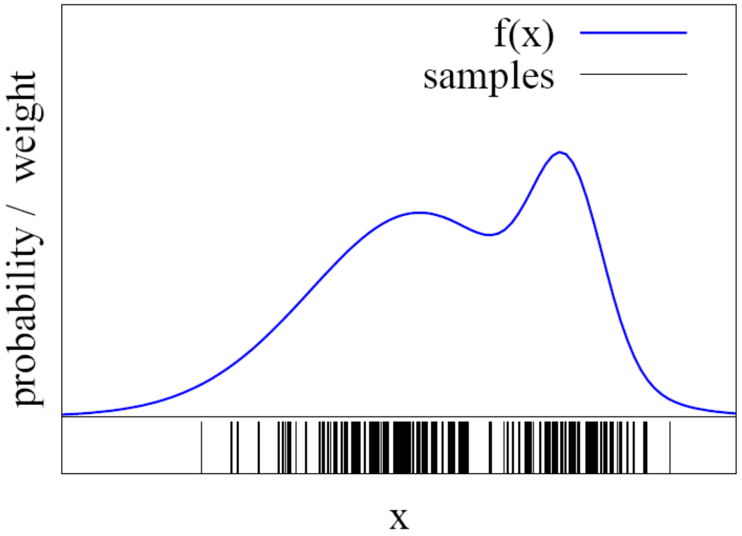


Let’s introduce some tools

Density Approximation

- Particle sets can be used to approximate densities



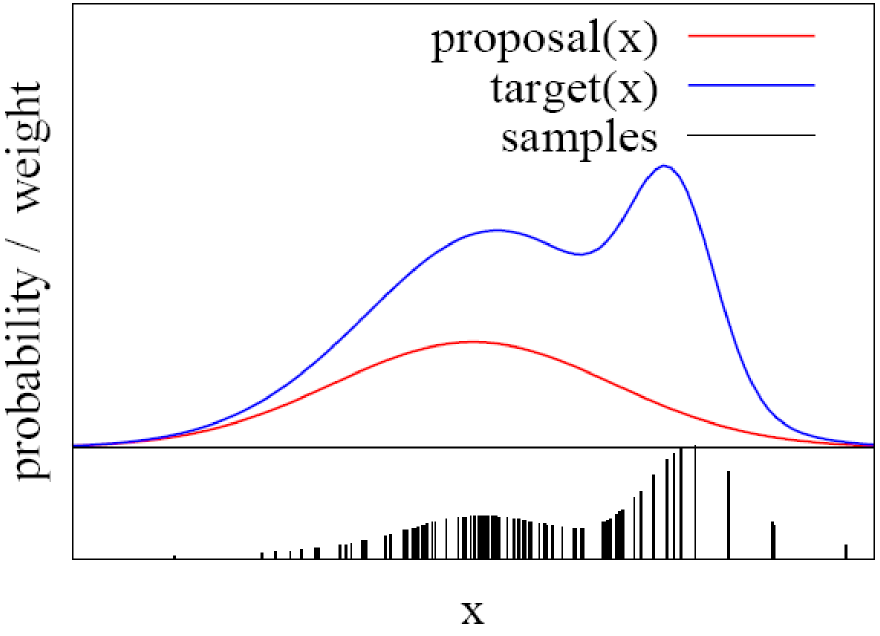


- The more particles fall into an interval, the higher the probability of that interval

- How to draw samples form a function/distribution?

Importance Sampling Principle

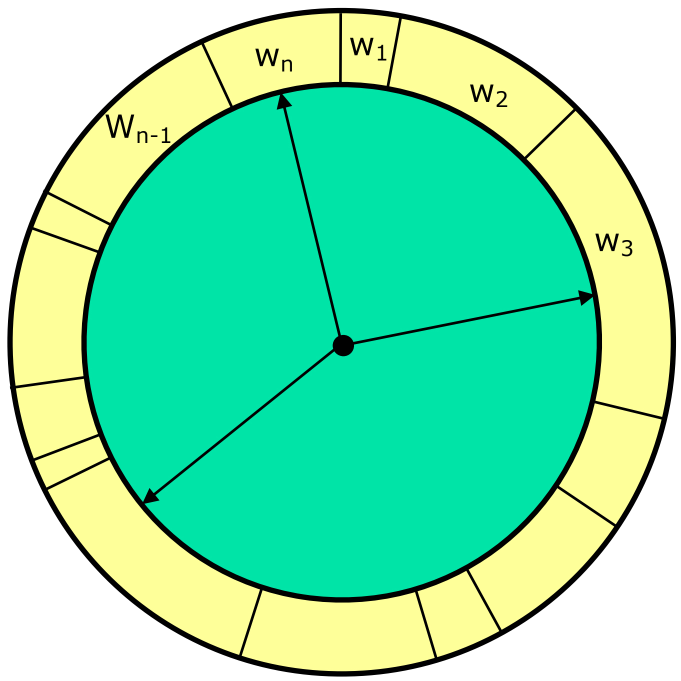
- We can even use a different distribution ***g*** to generate samples from ***f***
- By introducing an importance weight w, we can account for the "differences between ***g*** and ***f*** "
- ***w = f / g***
- ***f*** is often called target
- ***g*** is often called proposal

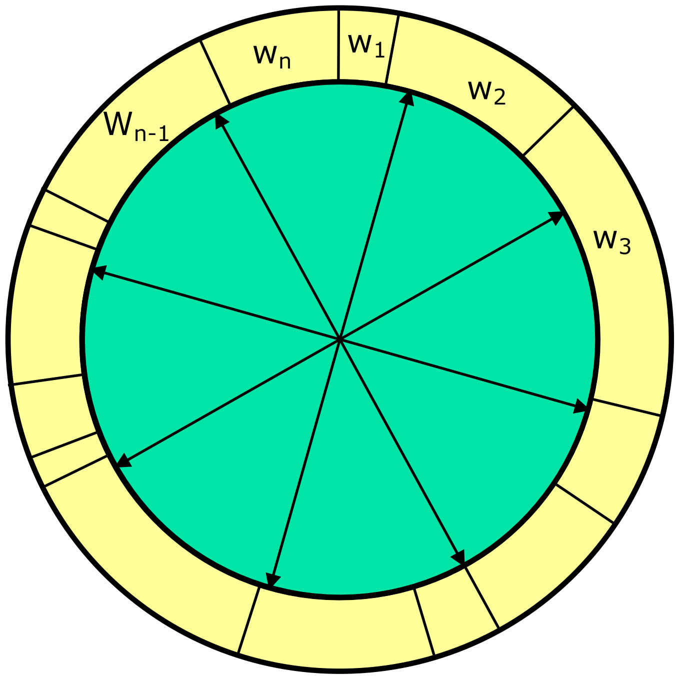


Resampling

- **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: Efficient Techniques





- Roulette wheel
- Binary search, n log n

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

Resampling: Efficient Techniques

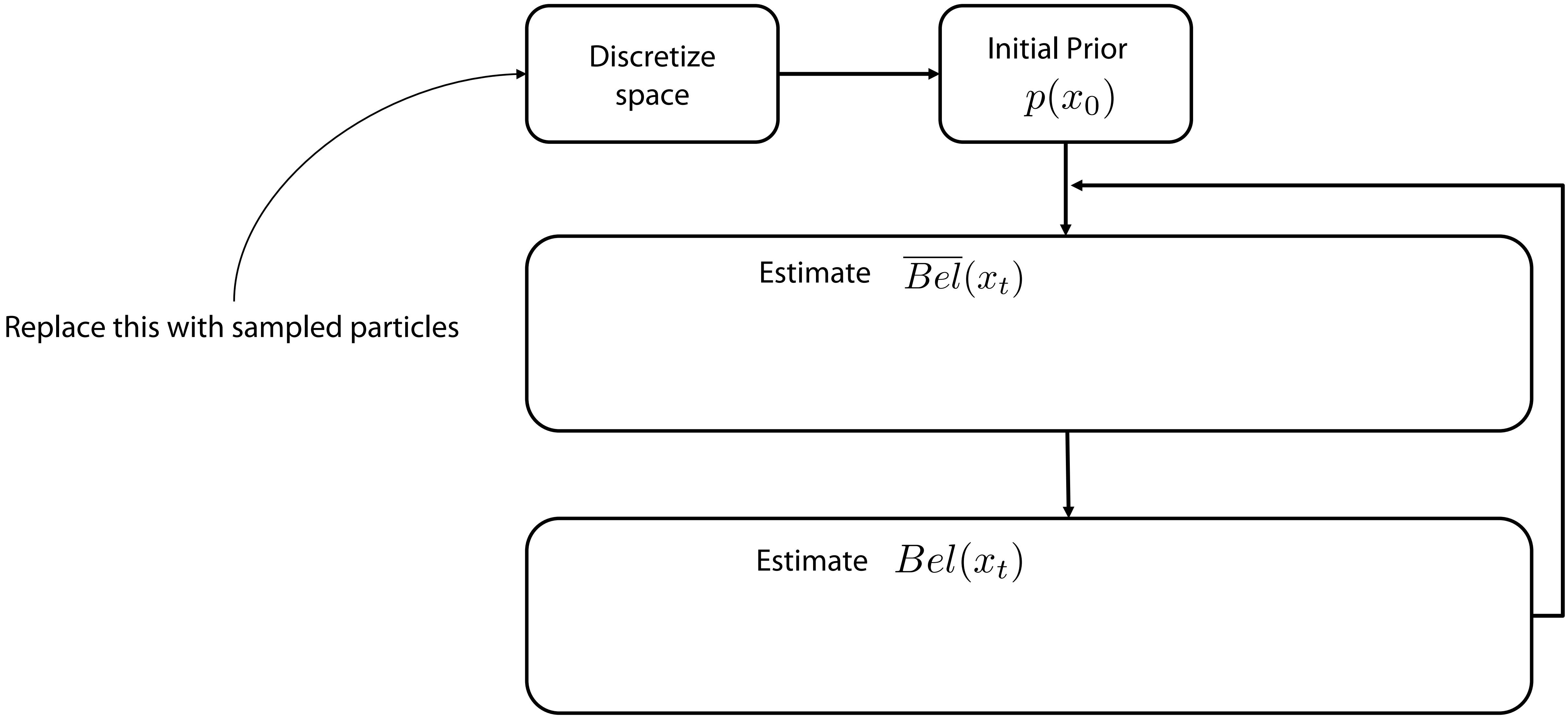
Pseudocode for low-variance sampling

```
Algorithm Low_variance_sampler(X_t, W_t):
    X̄_t = ∅
    r = rand(0; M^{-1})
    c = w_t^{[1]}
    i = 1
    for m = 1 to M do
        u = r + (m - 1) · M^{-1}
        while u > c
            i = i + 1
            c = c + w_t^{[i]}
        endwhile
        add x_t^{[i]} to X̄_t
    endfor
    return X̄_t
```

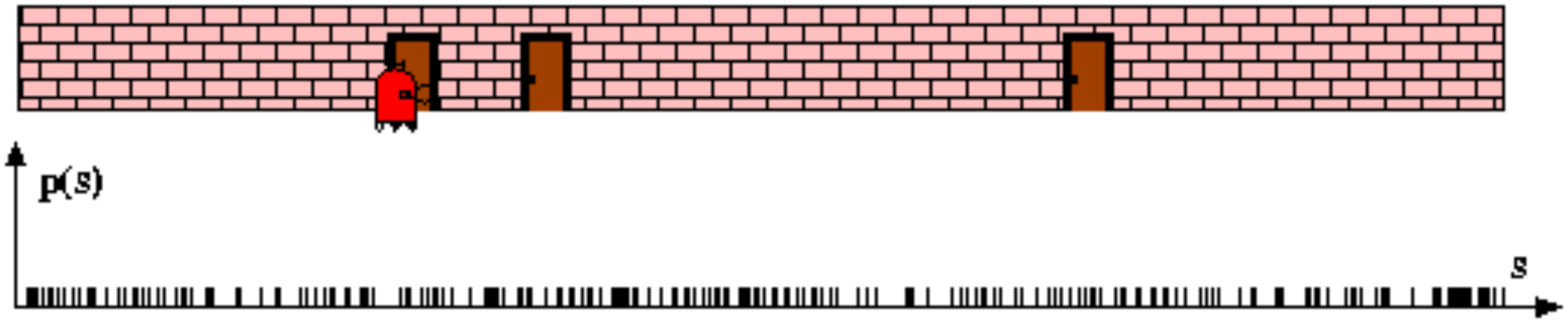
Table 4.4 Low variance resampling for the particle filter. This routine uses a single random number to sample from the particle set $\mathcal{X}$ with associated weights $\mathcal{W}$, yet the probability of a particle to be resampled is still proportional to its weight. Furthermore, the sampler is efficient: Sampling M particles requires $O(M)$ time.

Let’s put these pieces together

Particle Filters



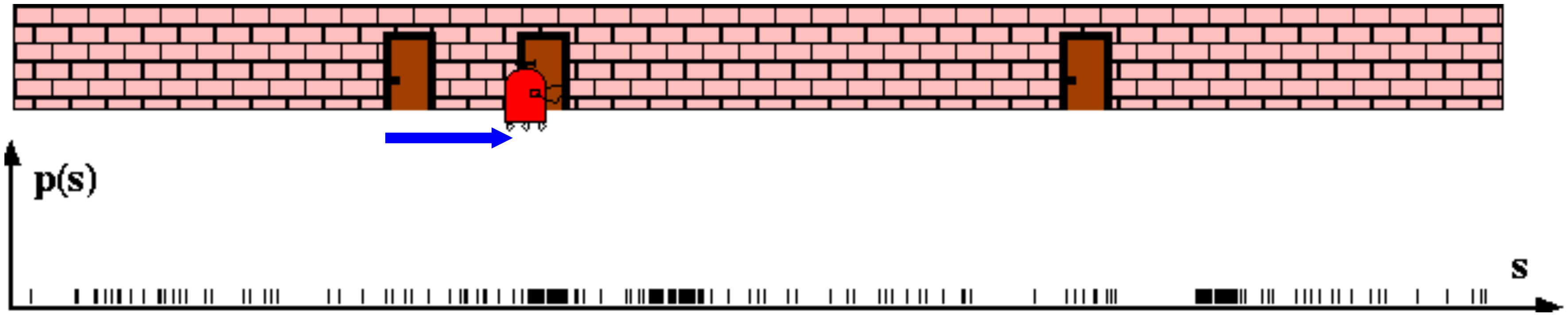
Particle Filters



Robot Motion

$$\overline{Bel}(x_t) = \int P(x_t | u_{t-1}, x_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

Push samples forward according to dynamics

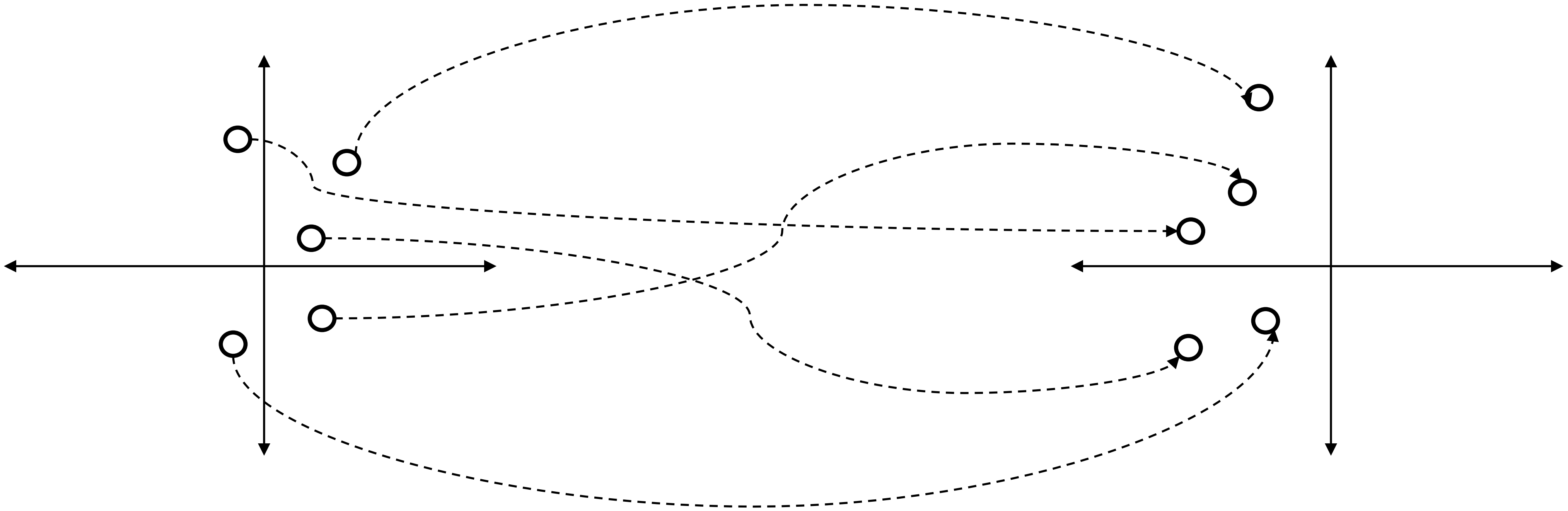


Dynamics Update:

$$\overline{Bel}(x_t) = \int P(x_t|u_{t-1}, x_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

Sample forward using the dynamics model:

1. No gaussian requirement
2. No linearity requirement, just push forward distribution



Sensor Information: Measurement Update

Can no longer just push forward with evidence, need to normalize

$$Bel(x_t) = \eta P(z_t|x_t)\overline{Bel}(x_t)$$

$$Bel(x_t) = \frac{P(z_t|x_t)\overline{Bel}(x_t)}{\int P(z_t|x_t)\overline{Bel}(x_t)dx_t}$$

Looks a lot like importance sampling!

Can compute a per sample importance weight

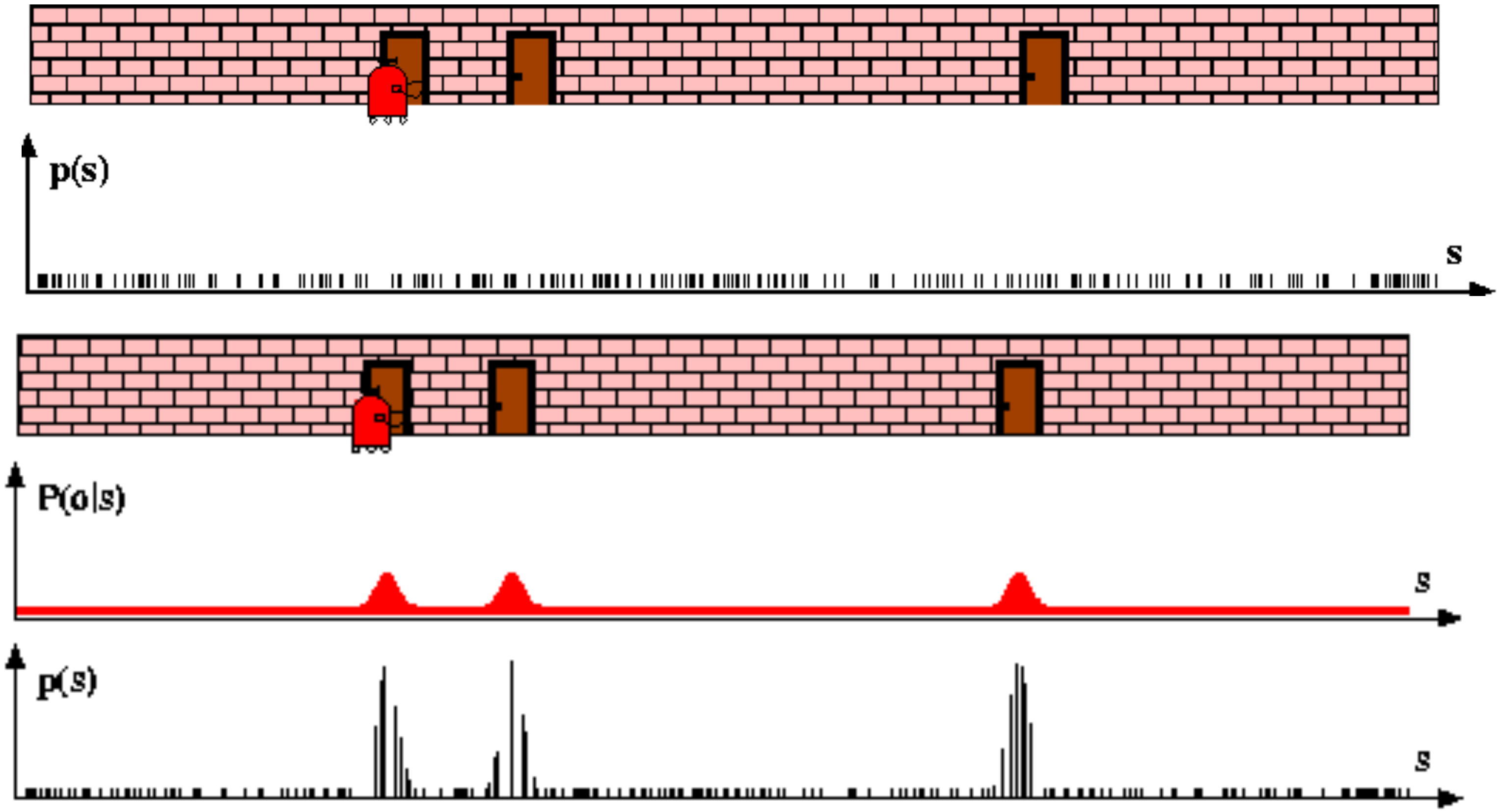
$$w_i = \frac{P(z_t|x_t^i)}{\sum_j P(z_t|x_t^j)}$$

Distribution can be represented as a set of weighted samples

Sensor Information: Importance Sampling

Can compute a weighted set of samples by weighting by (normalized) evidence

$$Bel(x_t) = \eta P(z_t|x_t)\overline{Bel}(x_t) \qquad w_i = \frac{P(z_t|x_t^i)}{\sum_j P(z_t|x_t^j)}$$

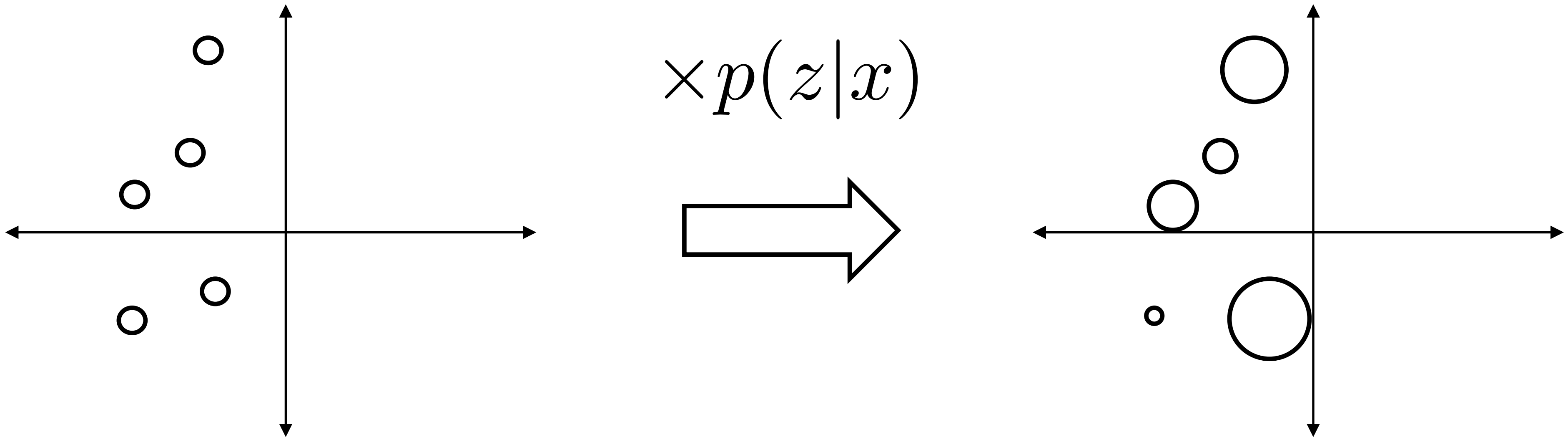


Measurement Update

$$Bel(x_t) = \eta P(z_t|x_t)\overline{Bel}(x_t)$$

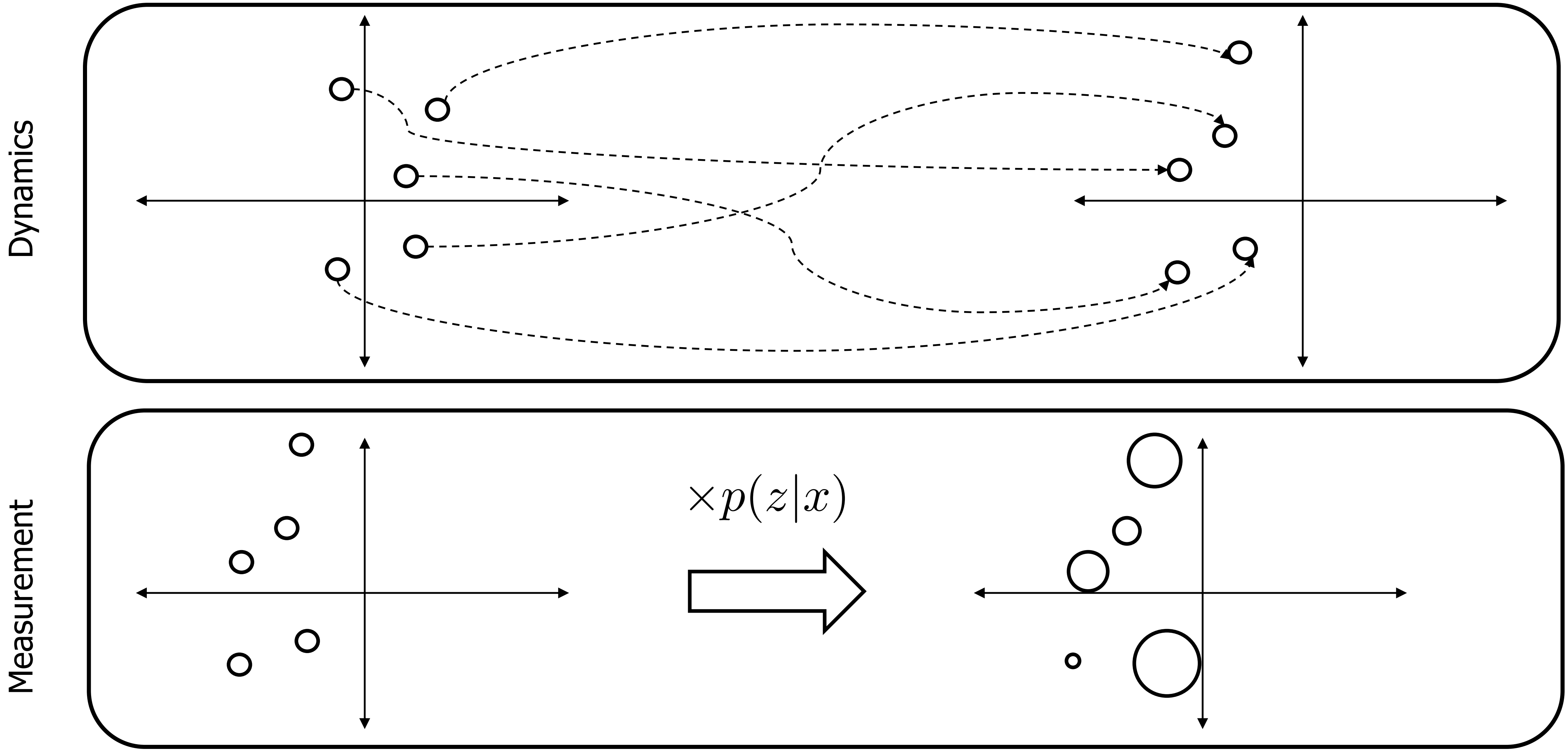
$$Bel(x_t) = \frac{P(z_t|x_t)\overline{Bel}(x_t)}{\int P(z_t|x_t)\overline{Bel}(x_t)dx_t}$$

$$w_i = \frac{P(z_t|x_t^i)}{\sum_j P(z_t|x_t^j)}$$



Reweight particles according to measurement likelihood

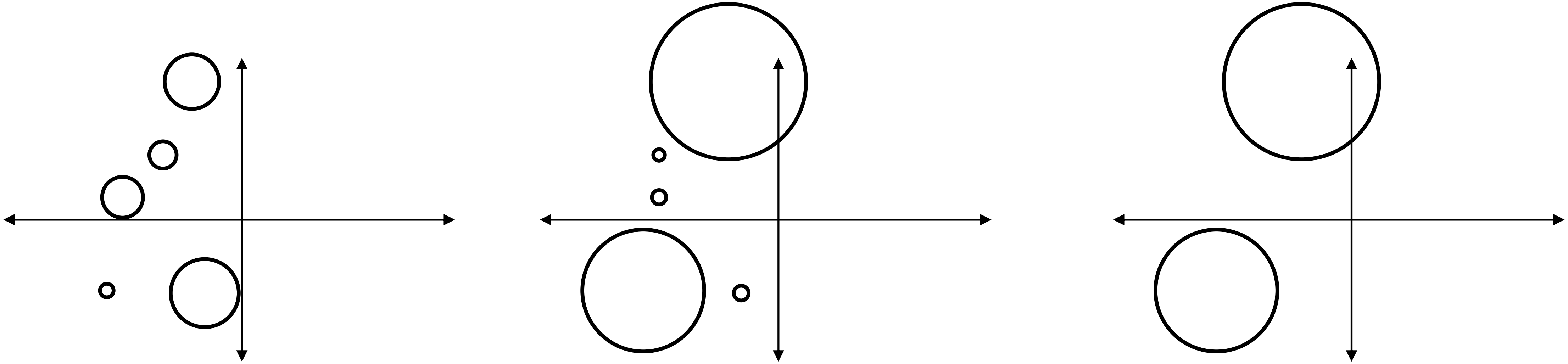
What happens across multiple steps?



Importance weights get multiplied at each step

Why might this be bad?

Importance weights get multiplied at each step



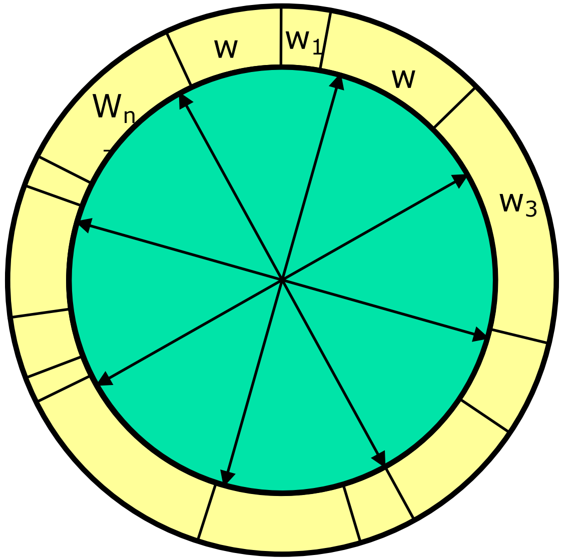
1. May blow up and get numerically unstable over many steps
2. Evidence doesn't affect samples themselves, just weights

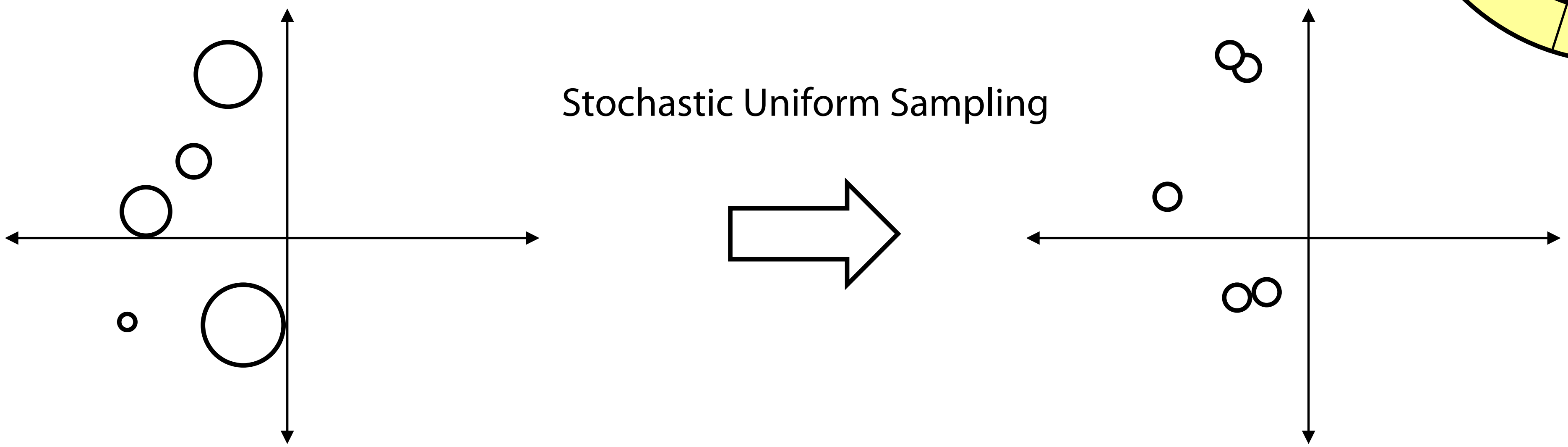
Measurement Update: Resampling

$$Bel(x_t) = \eta P(z_t|x_t)\overline{Bel}(x_t)$$

$$Bel(x_t) = \frac{P(z_t|x_t)\overline{Bel}(x_t)}{\int P(z_t|x_t)\overline{Bel}(x_t)dx_t}$$

$$w_i = \frac{P(z_t|x_t^i)}{\sum_j P(z_t|x_t^j)}$$



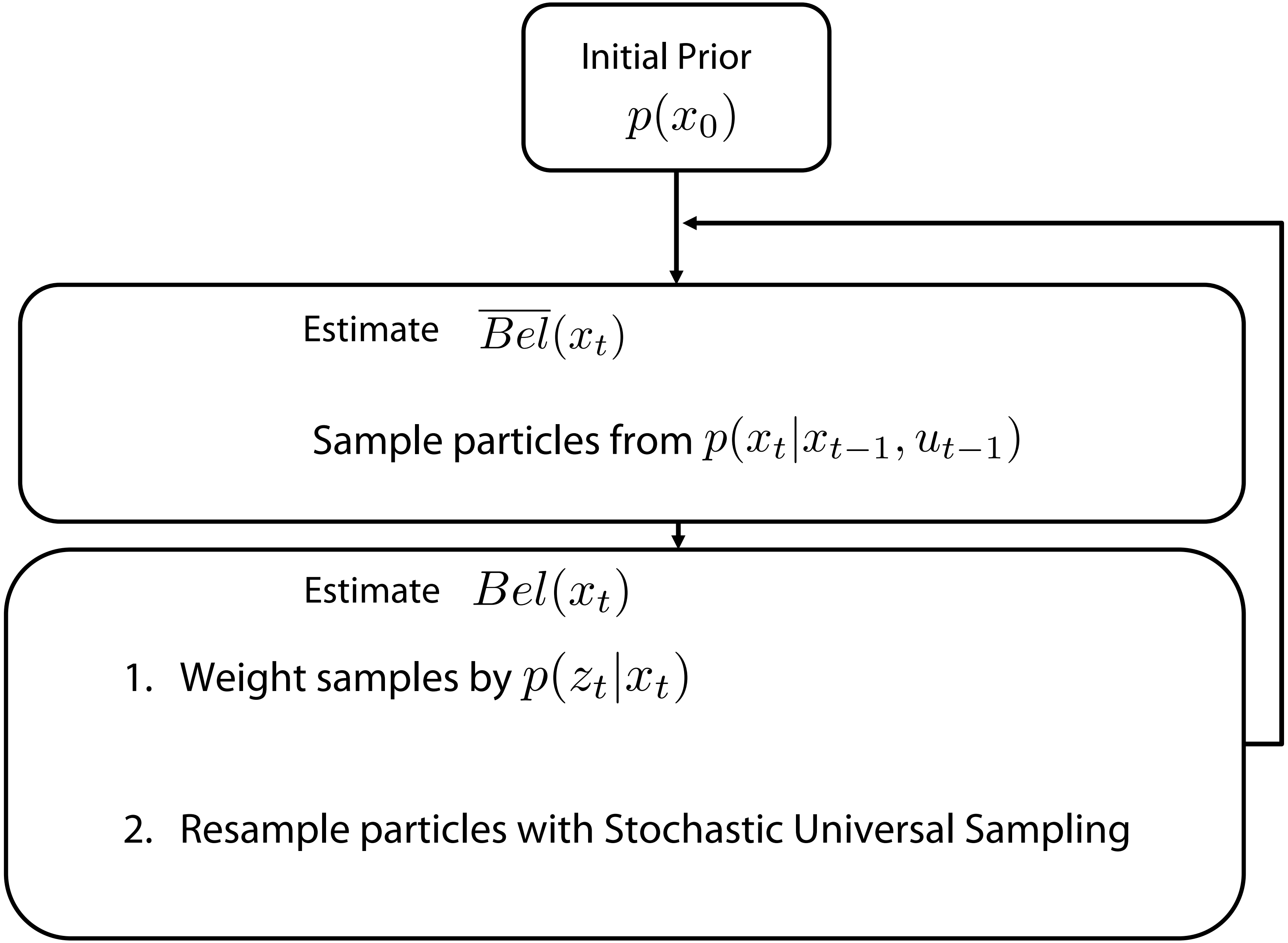


Resample particles from weighted distribution with SUS

Overall Particle Filter algorithm

Dynamics/Prediction

Measurement/Correction



Particle Filter Algorithm

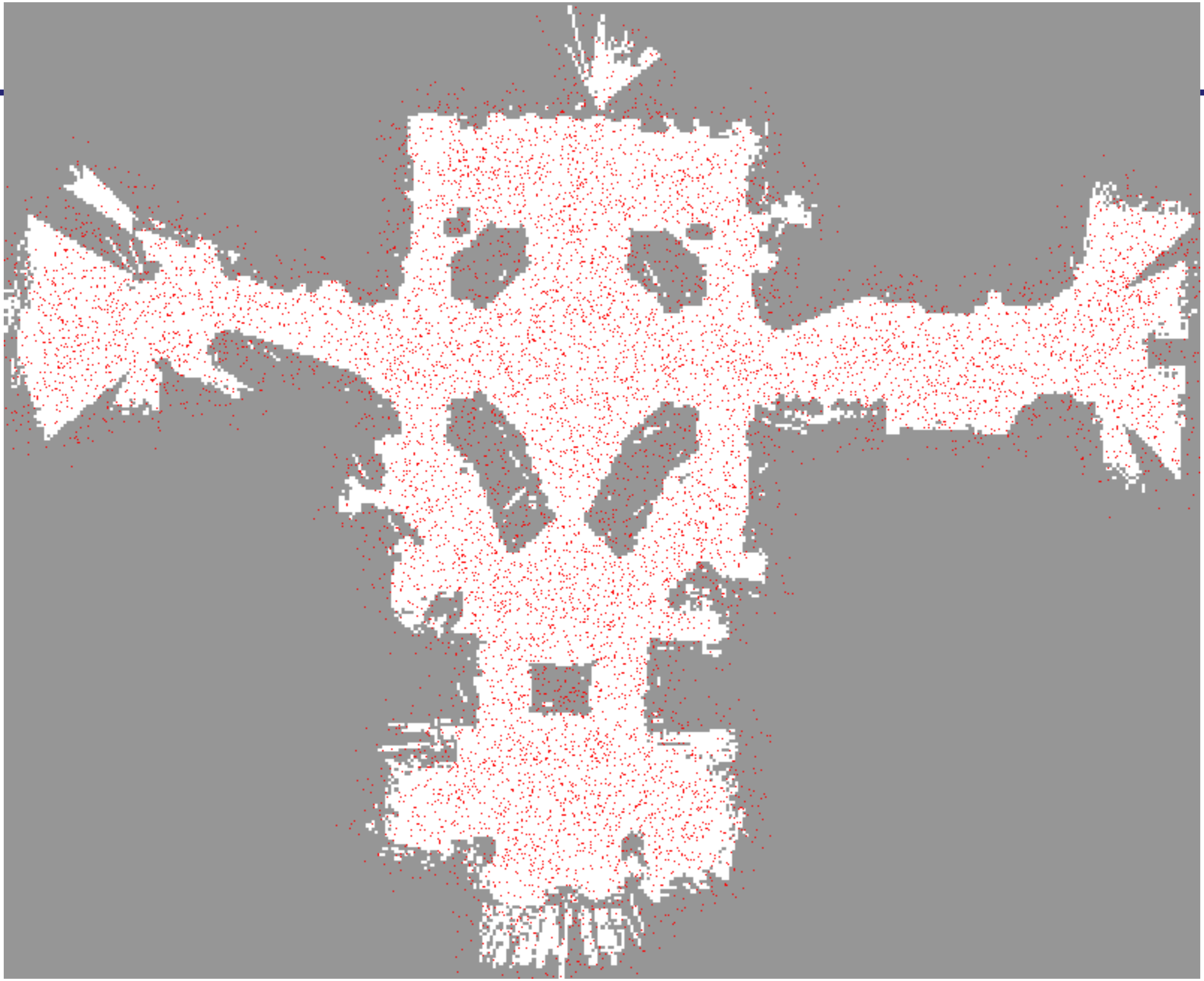
$$Bel(x_t) = \eta\, p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, u_{t-1})\, Bel(x_{t-1})\, dx_{t-1}$$

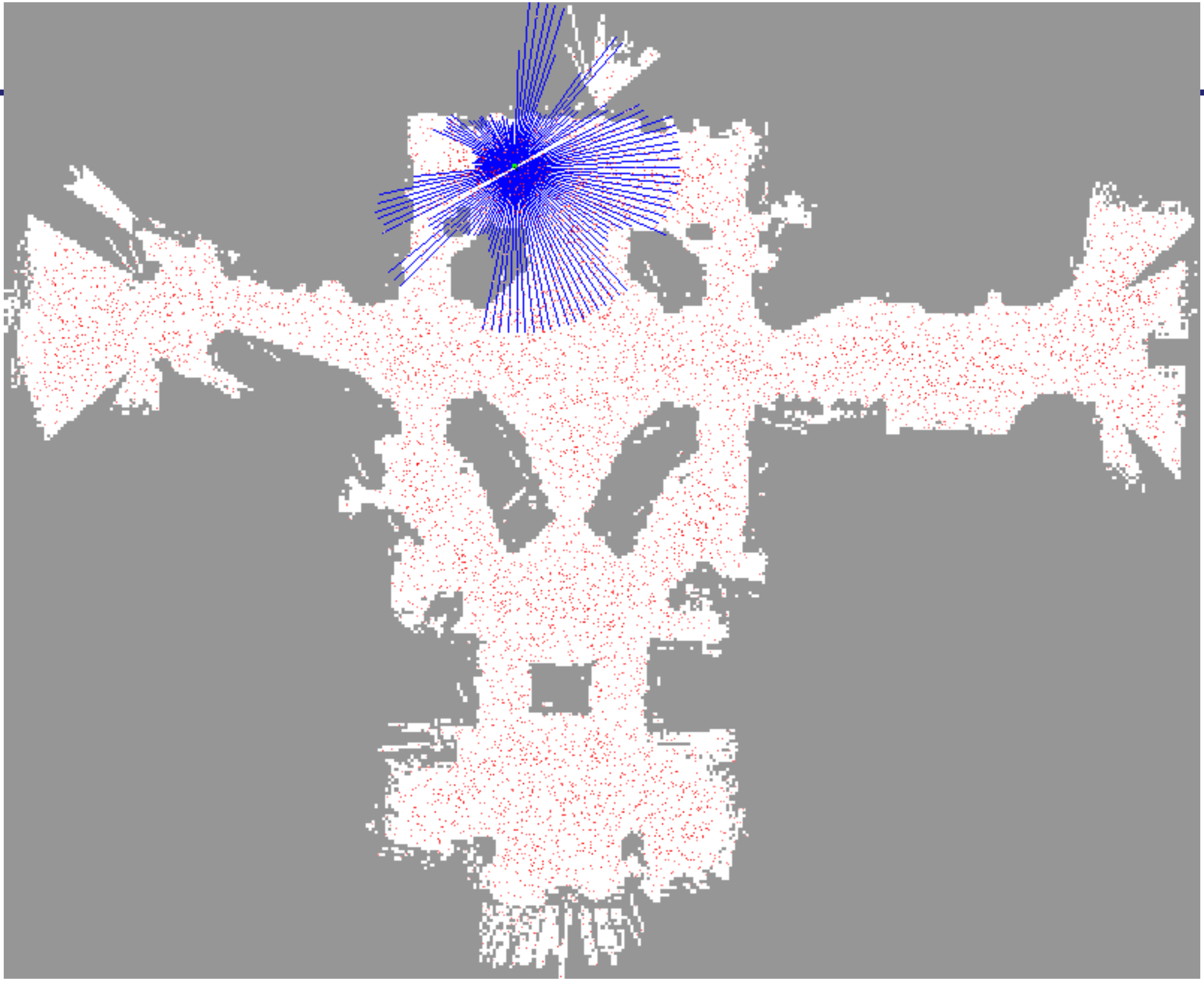
- $p(x_t \mid x_{t-1}, u_{t-1})$ → draw x^i_t from $p(x_t \mid x^i_{t-1}, u_{t-1})$
- $Bel(x_{t-1})$ → draw x^i_{t-1} from $Bel(x_{t-1})$
- $p(z_t \mid x_t)$ → Importance factor for x^i_t:

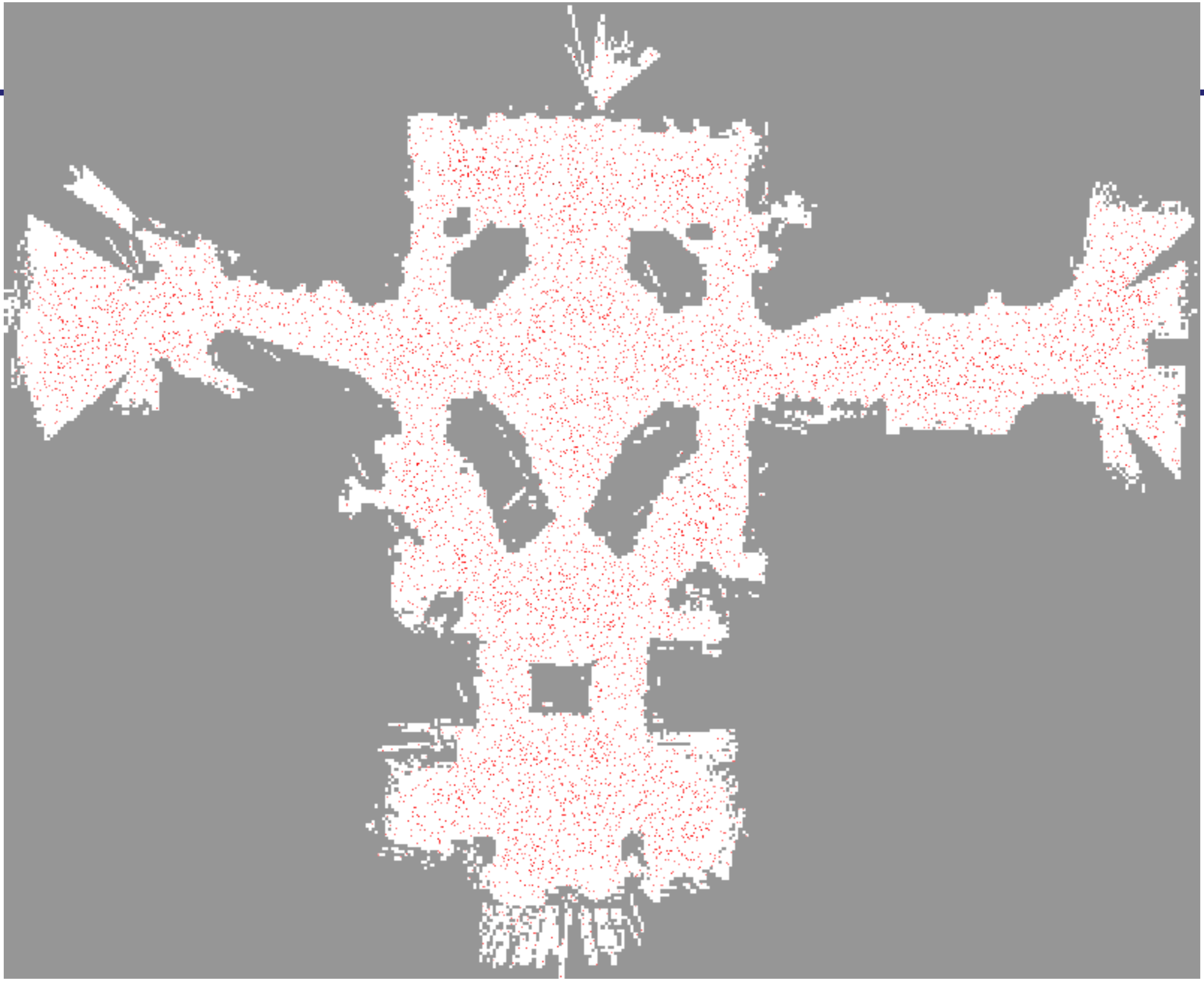
$$w^i_t = \frac{\text{target distribution}}{\text{proposal distribution}} = \frac{\eta\, p(z_t \mid x_t)\; p(x_t \mid x_{t-1}, u_{t-1})\; Bel(x_{t-1})}{p(x_t \mid x_{t-1}, u_{t-1})\; Bel(x_{t-1})} \propto p(z_t \mid x_t)$$

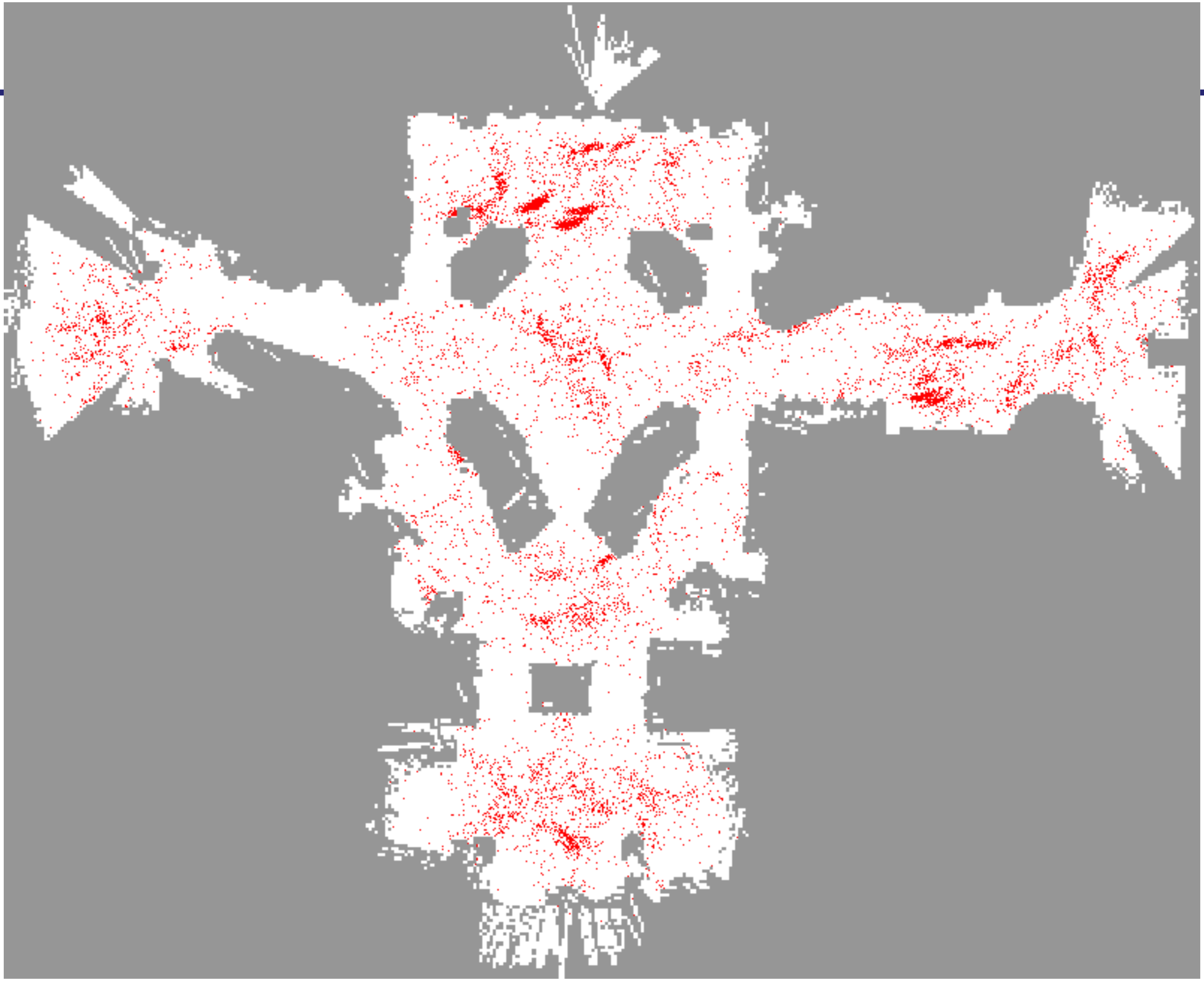
Particle Filter Algorithm

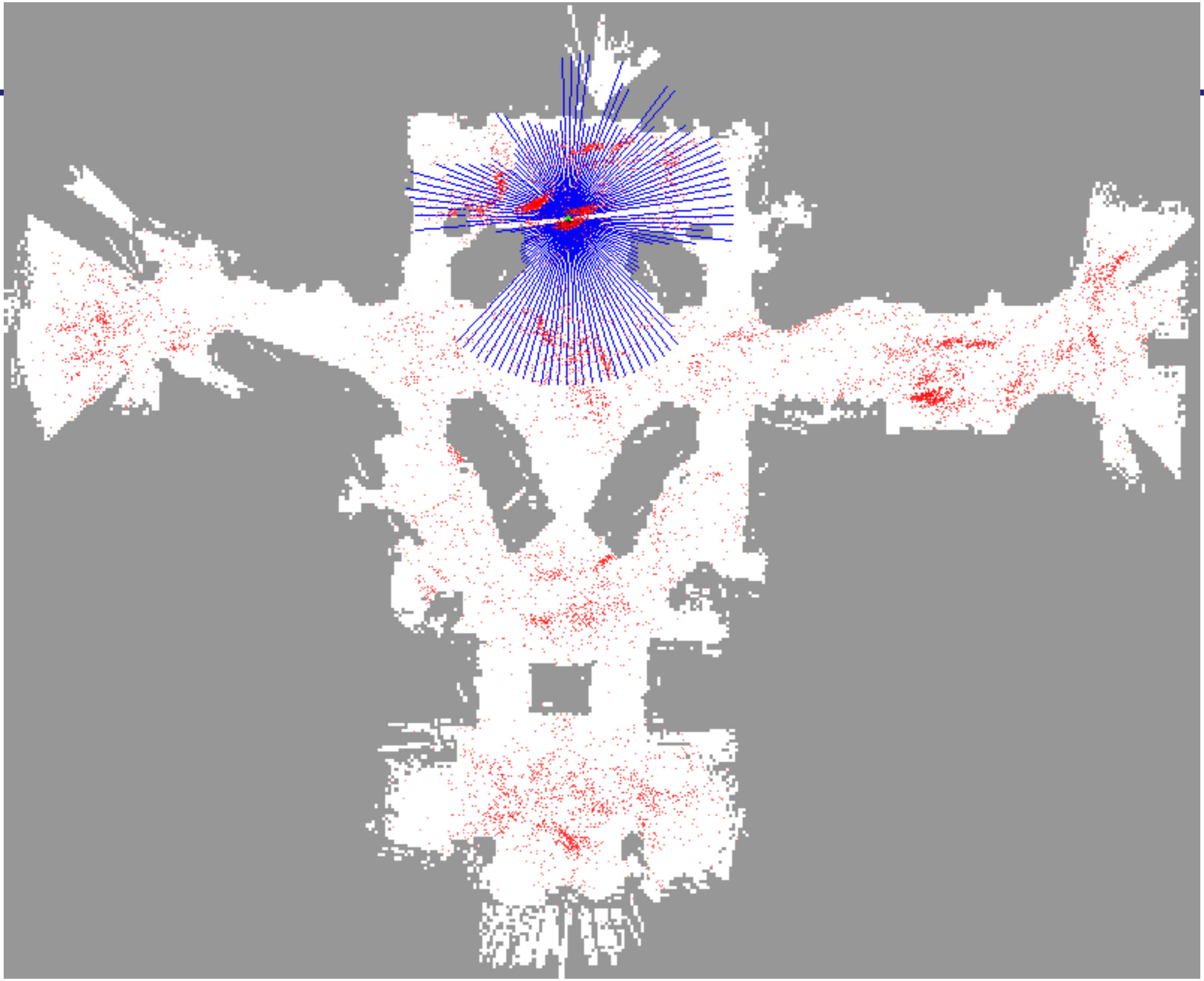
1. Algorithm **particle_filter**(S_{t-1}, u_{t-1} z_t):
2. $S_t = \emptyset, \quad \eta = 0$
3. **For** $i = 1 \ldots n$ ***Generate new samples***
4. Sample index $j(i)$ from the discrete distribution given by w_{t-1}
5. Sample x_t^i from $p(x_t \mid x_{t-1}, u_{t-1})$ using $x_{t-1}^{j(i)}$ and u_{t-1}
6. $w_t^i = p(z_t \mid x_t^i)$ ***Compute importance weight***
7. $\eta = \eta + w_t^i$ ***Update normalization factor***
8. $S_t = S_t \cup \{< x_t^i, w_t^i >\}$ ***Insert***
9. **For** $i = 1 \ldots n$
10. $w_t^i = w_t^i / \eta$ ***Normalize weights***

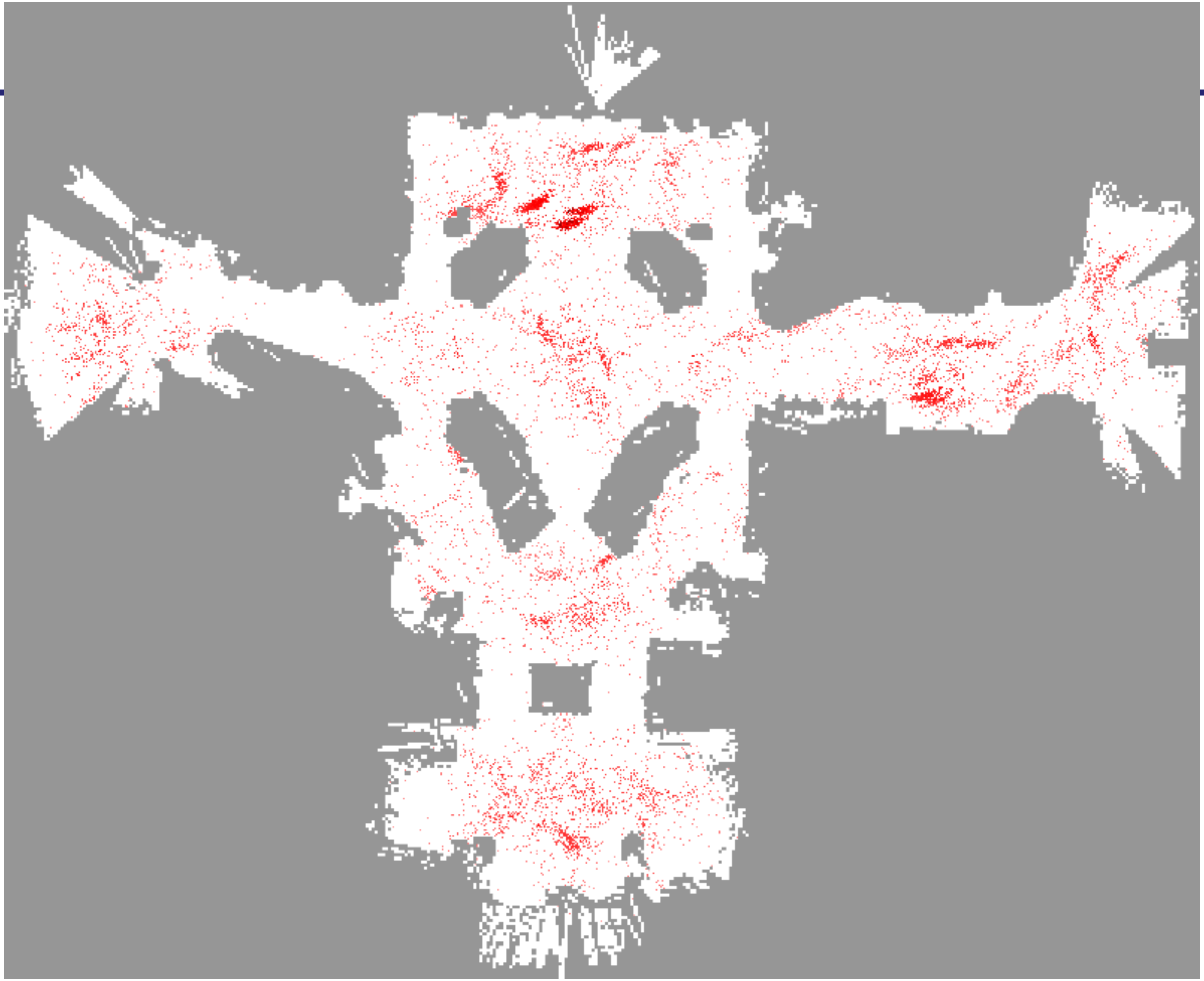


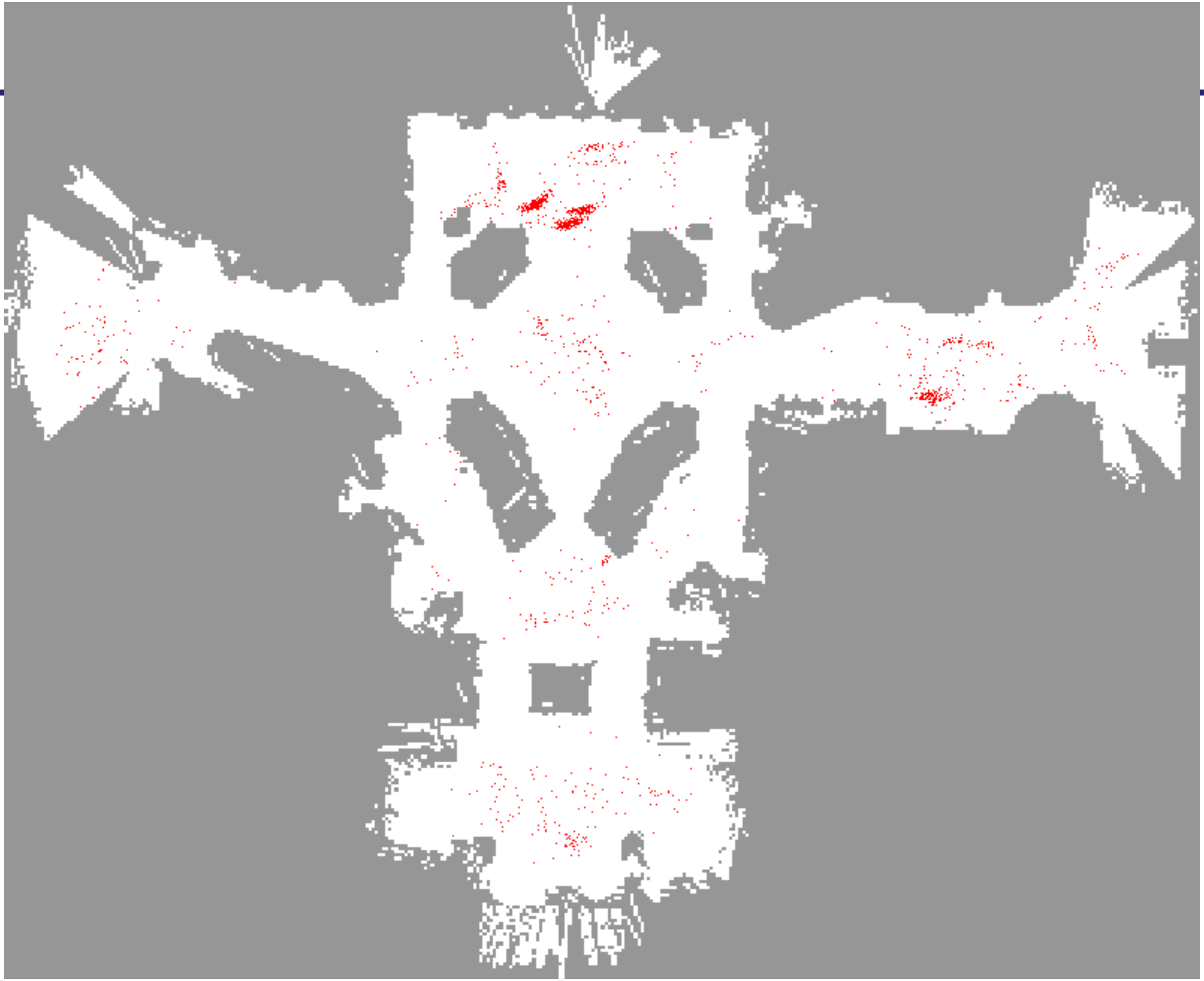


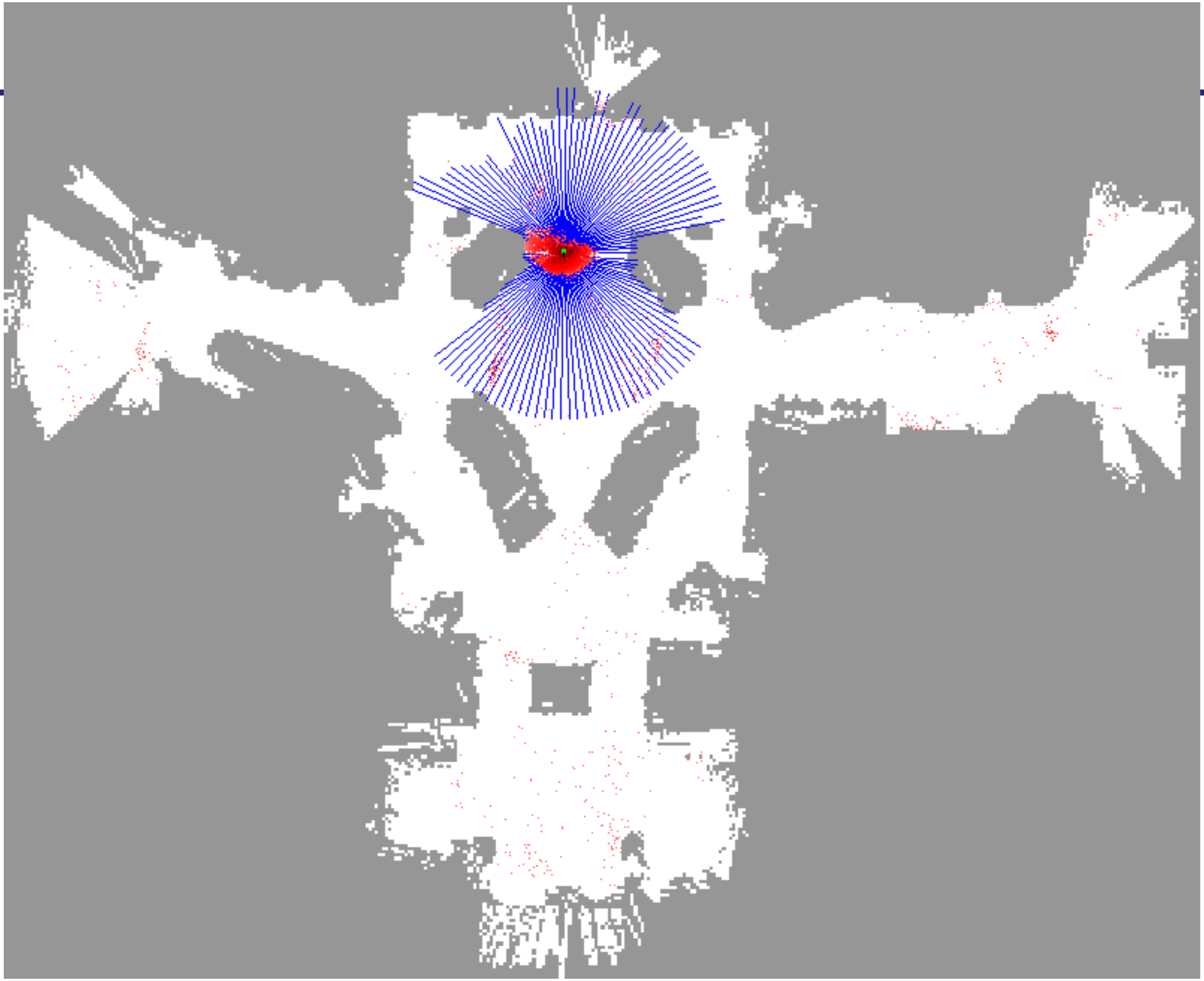


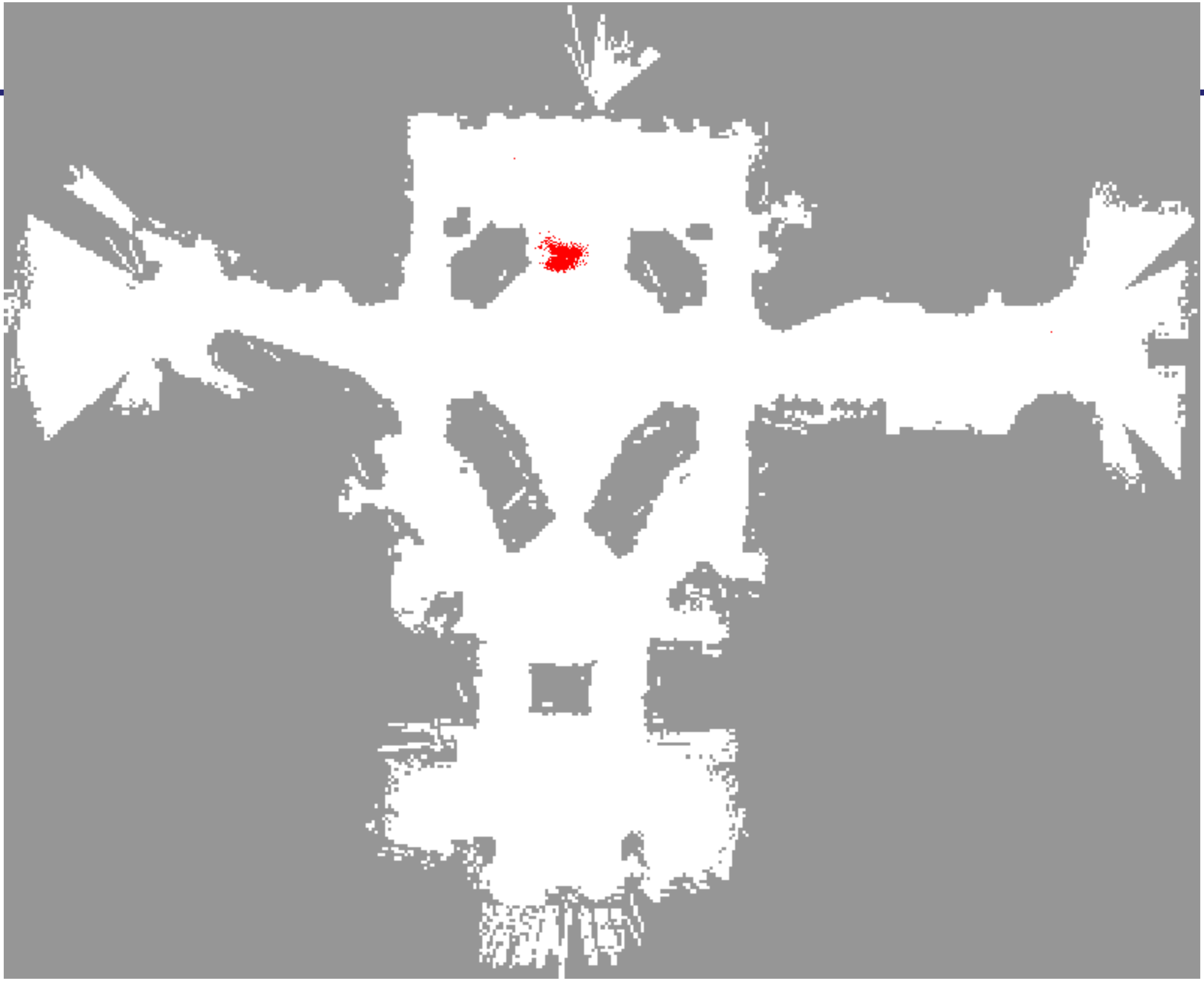


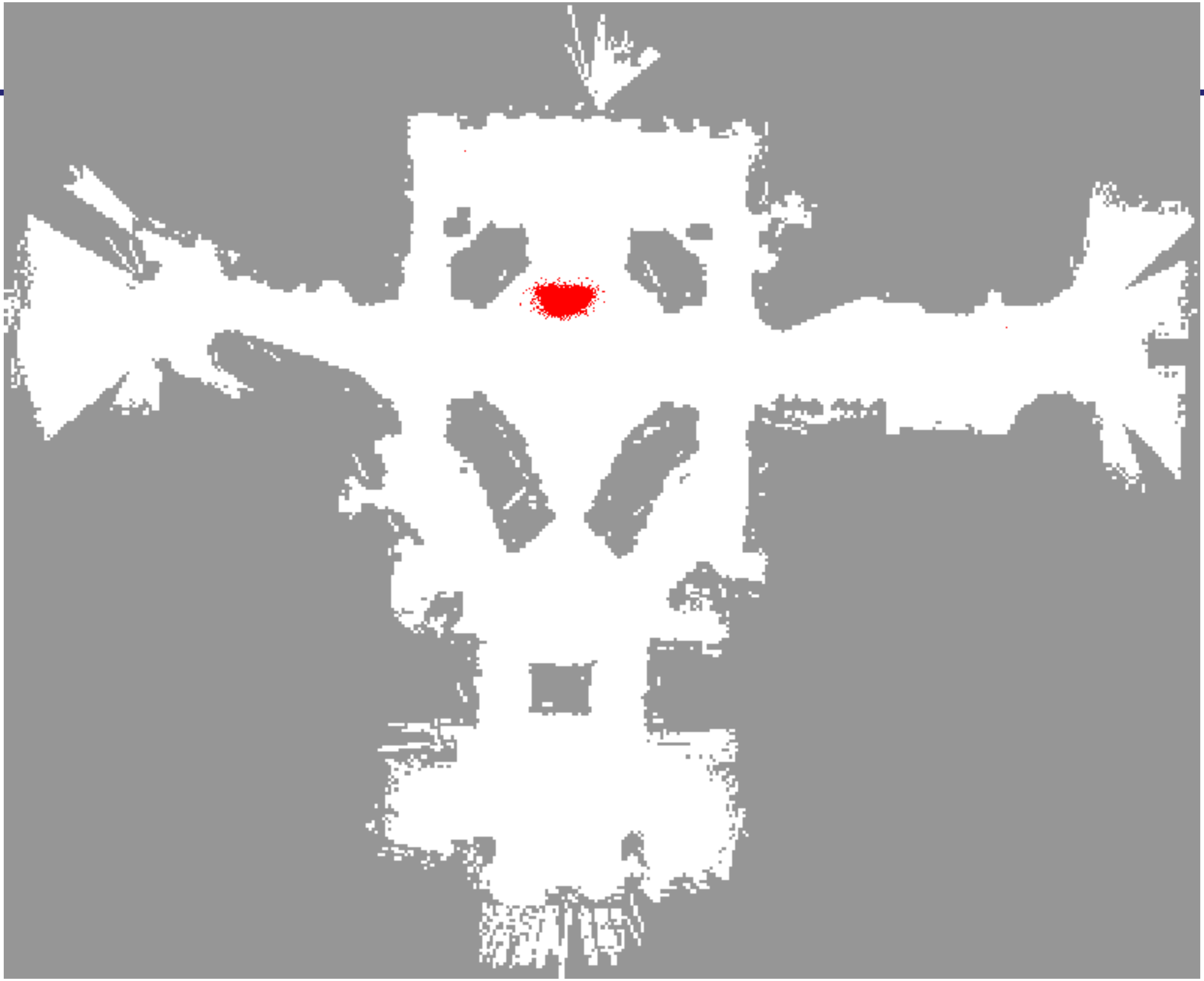


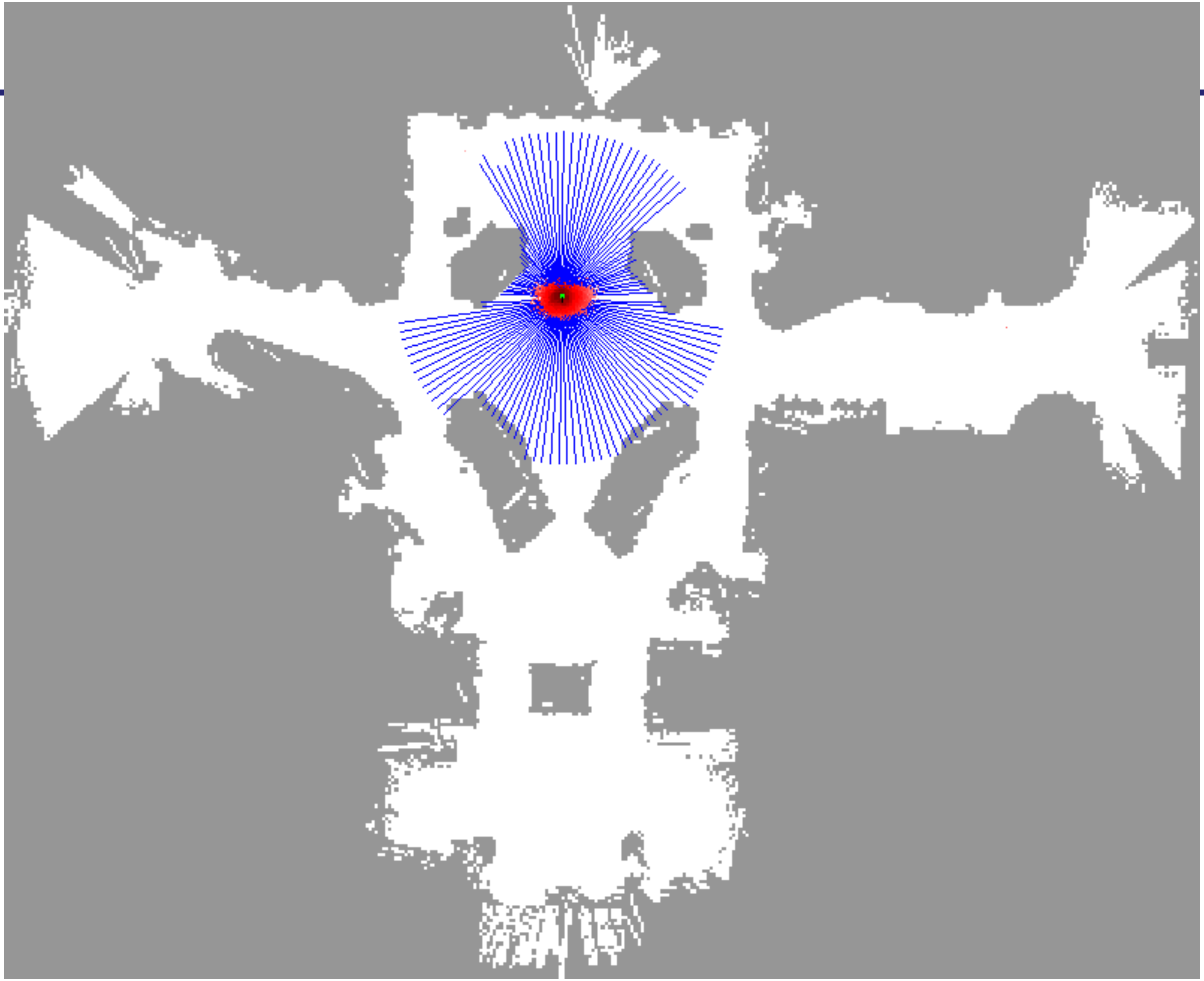


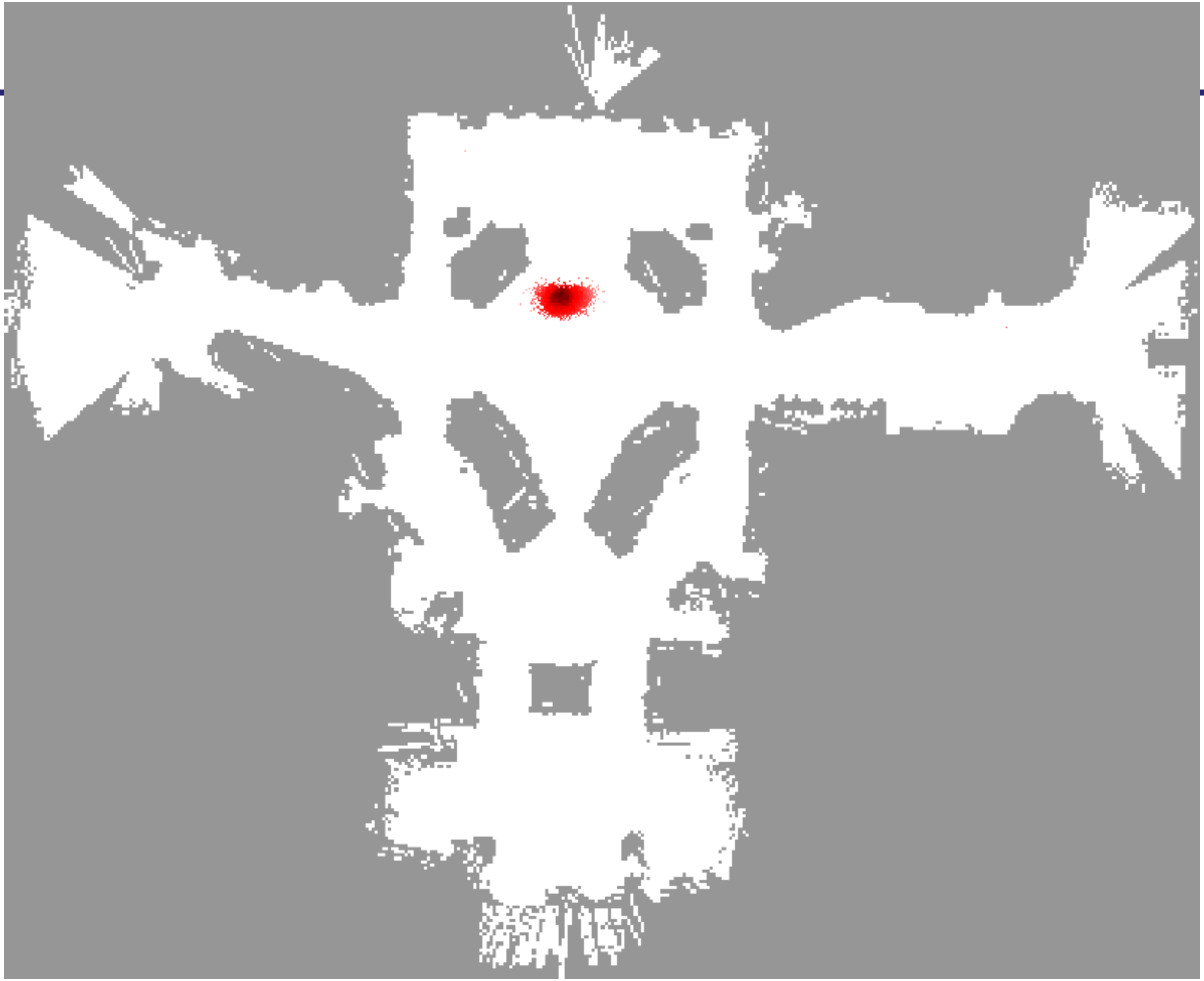


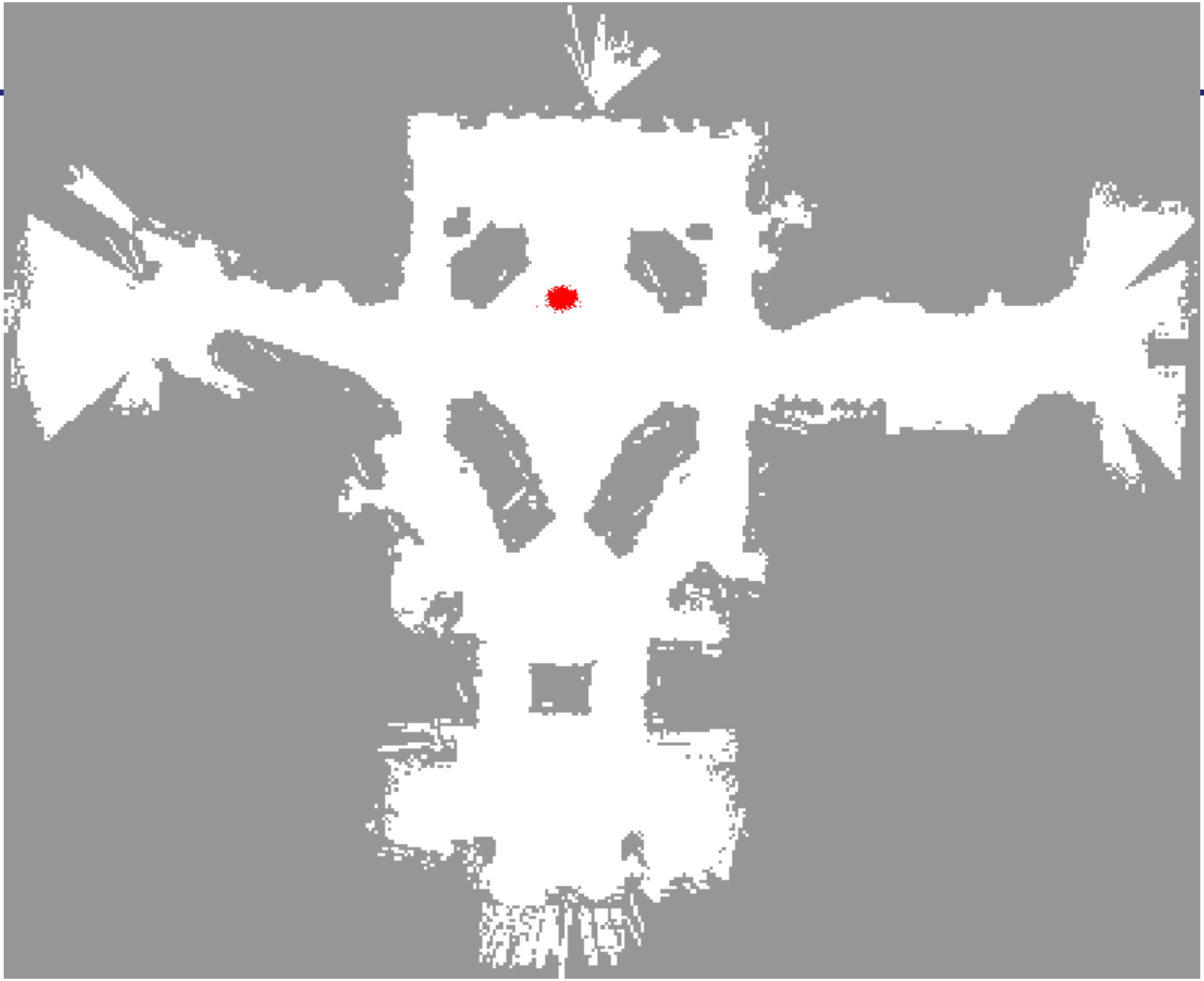


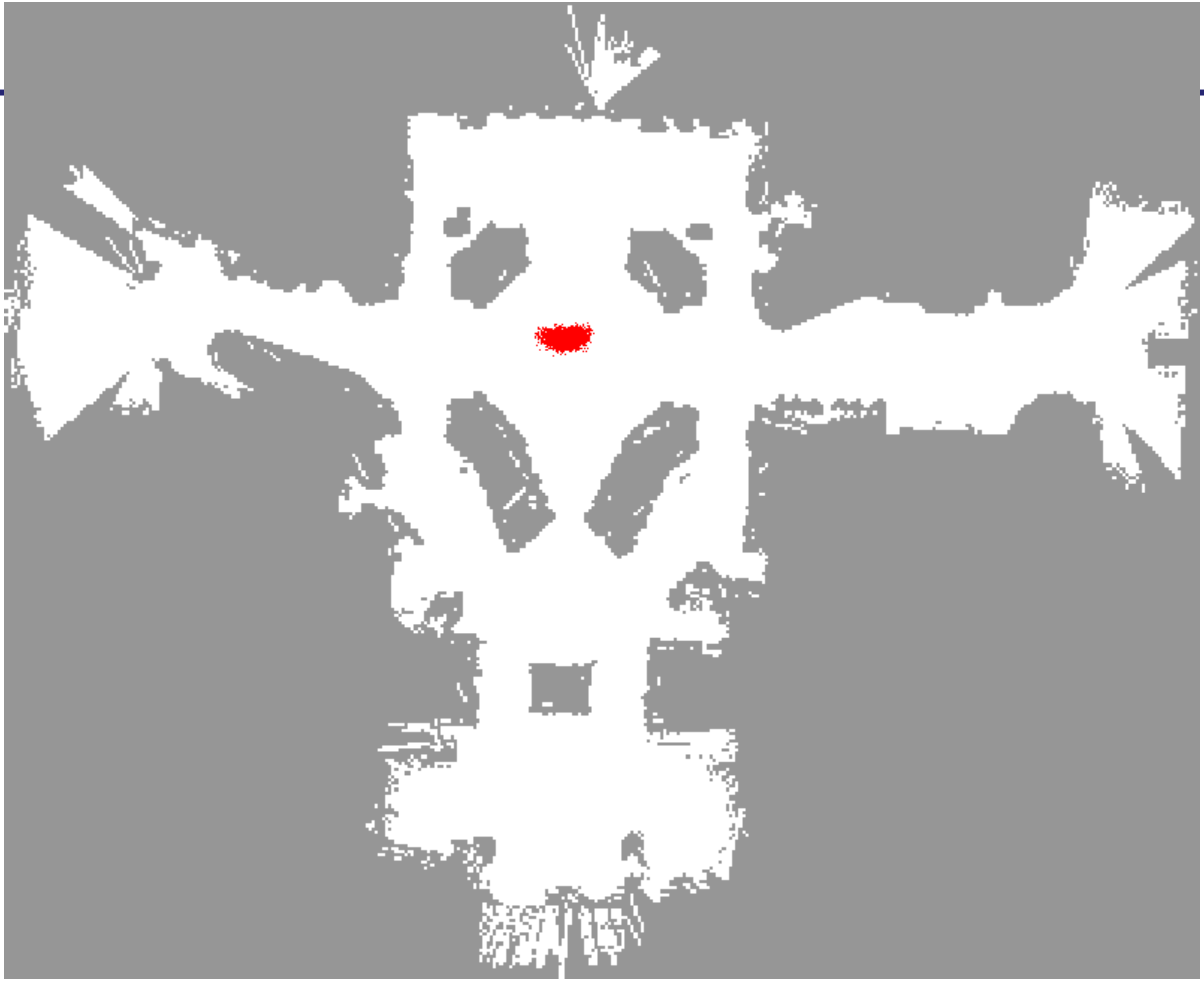


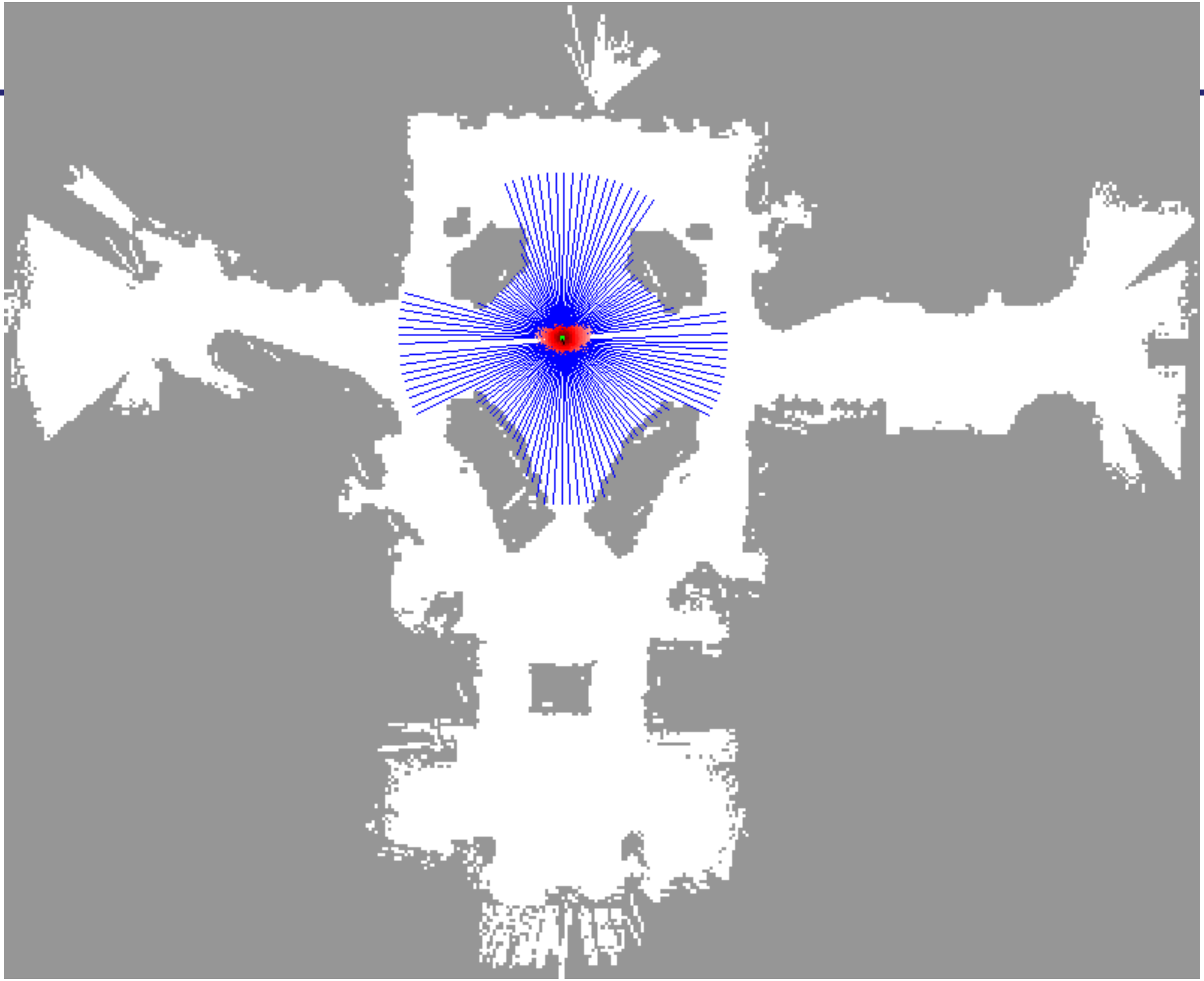


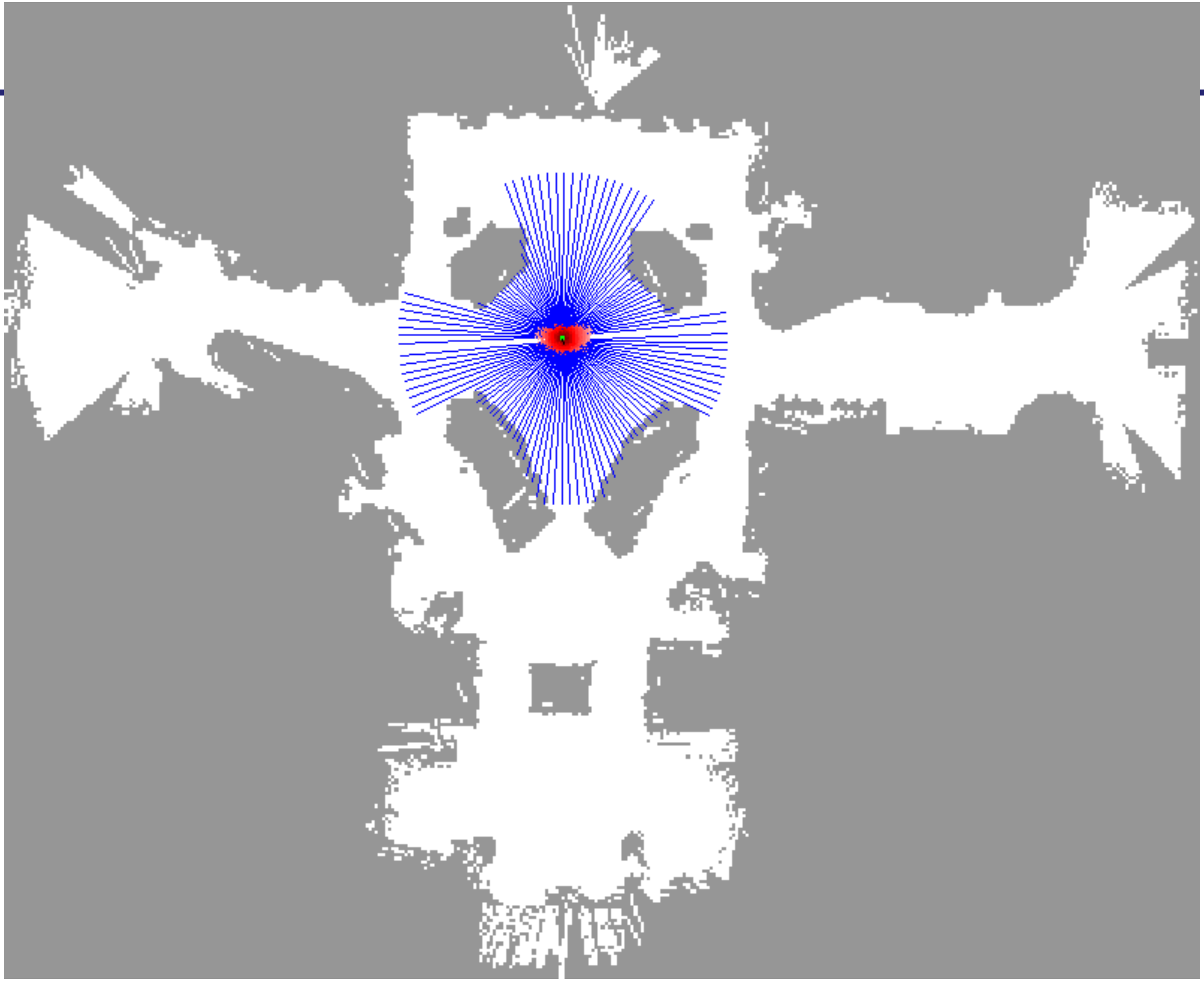




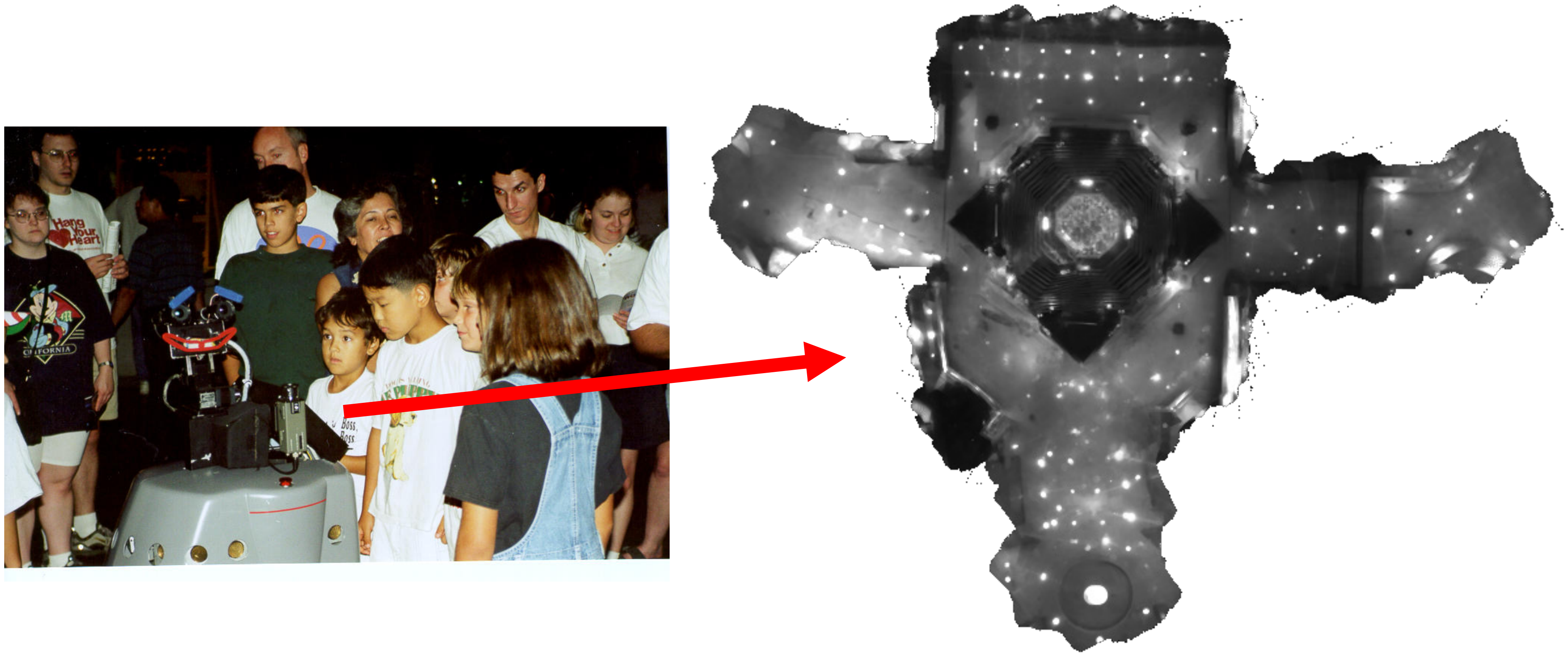






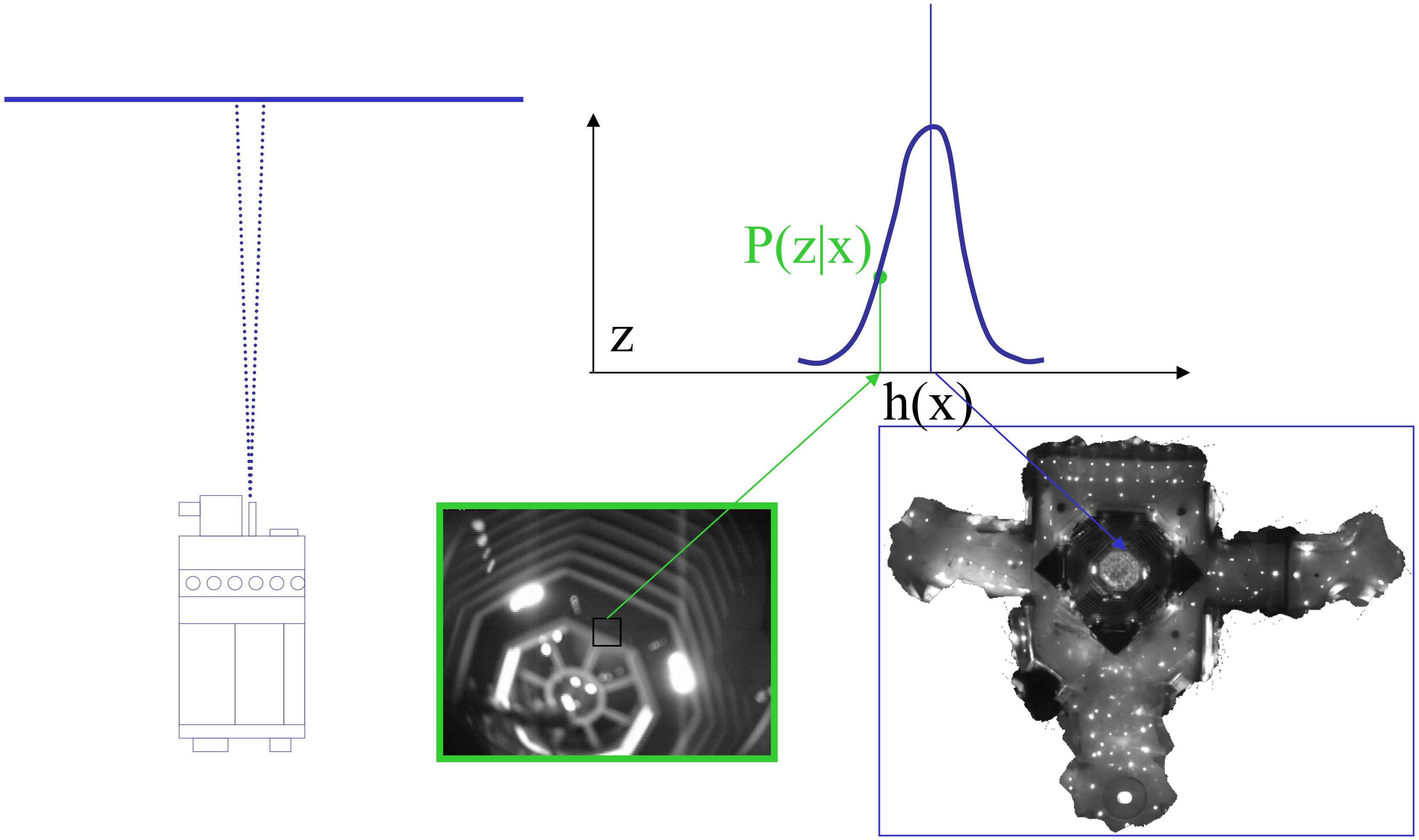


Using Ceiling Maps for Localization



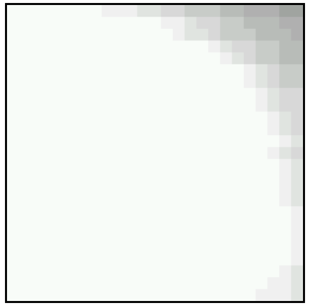
[Dellaert et al. 99]

Vision-based Localization

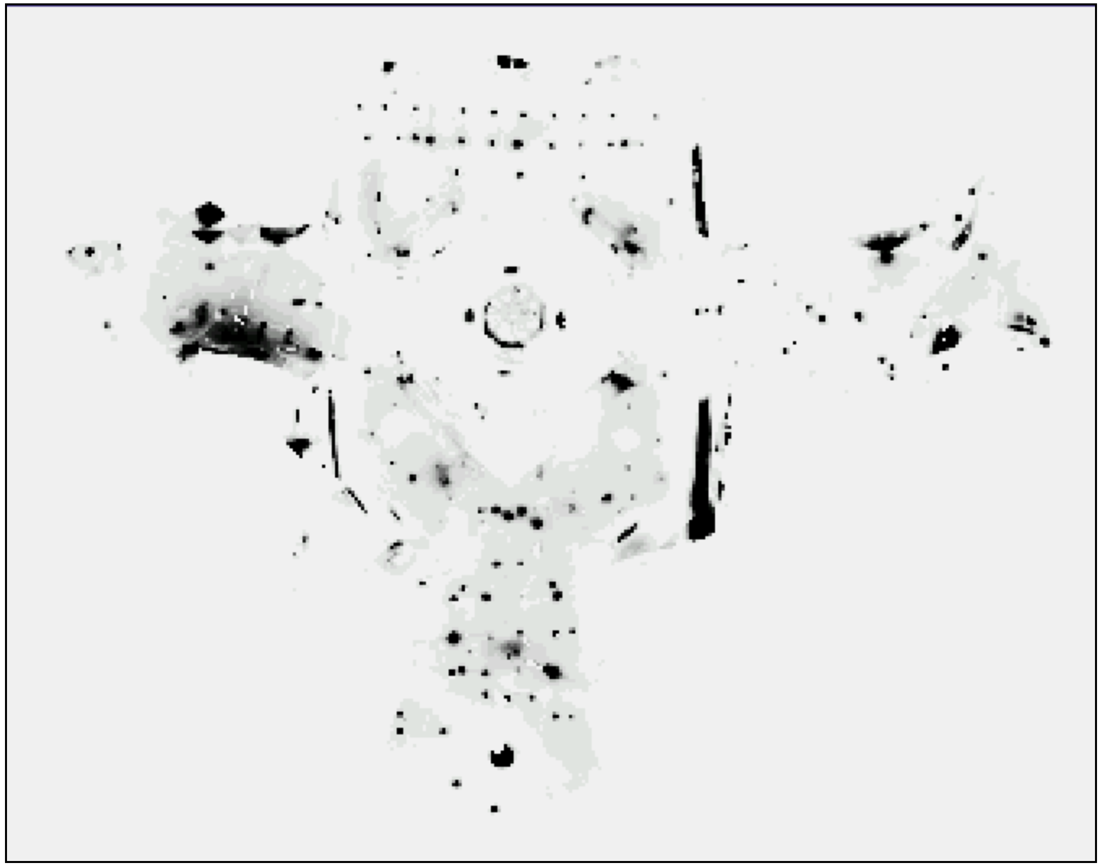


Under a Light

Measurement z:

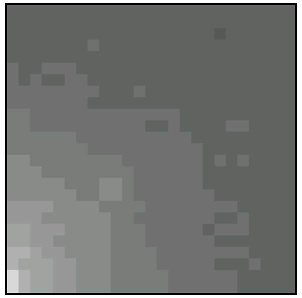


***P(z|x)*:**

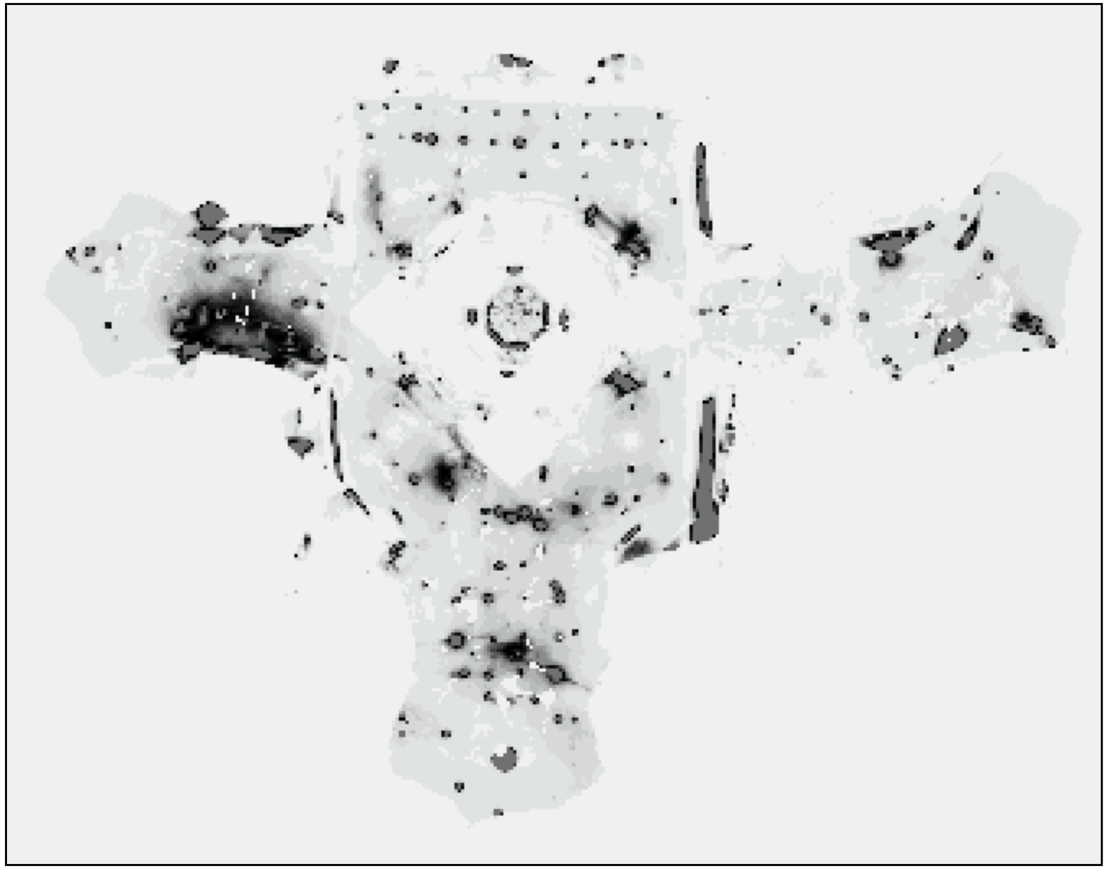


Next to a Light

Measurement z:



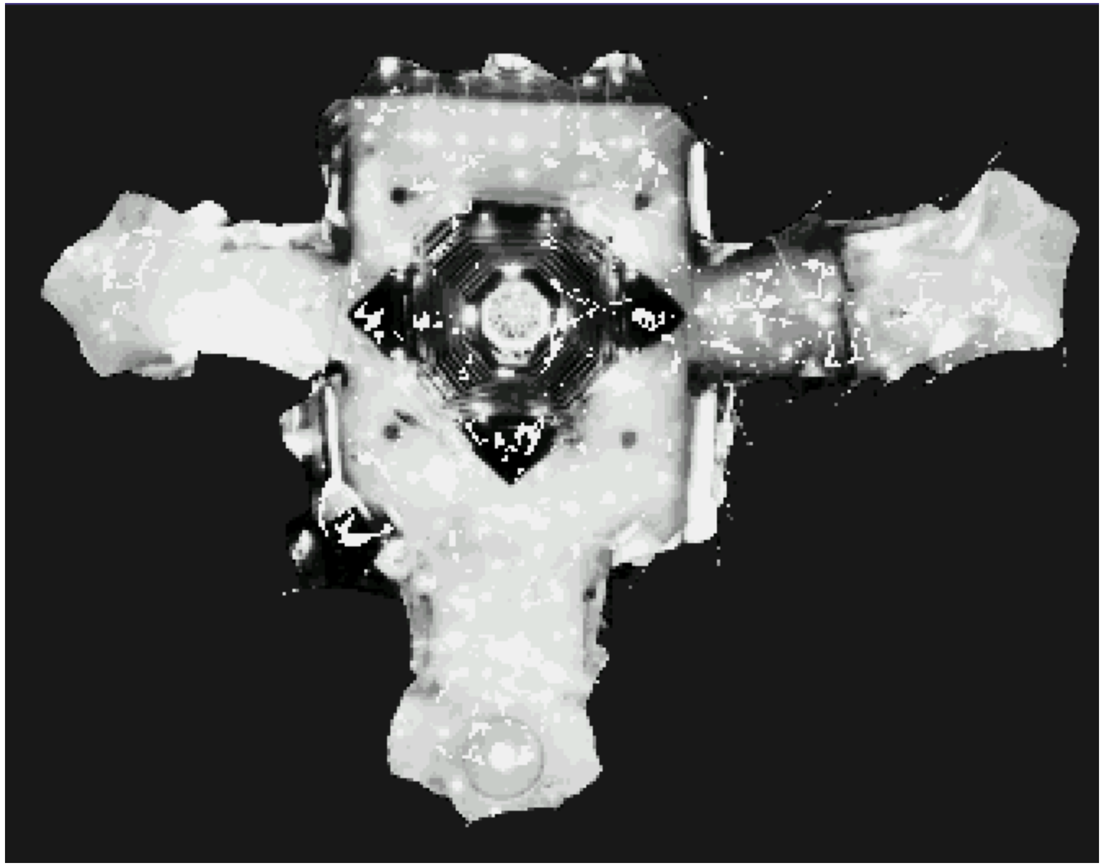
***P(z|x)*:**



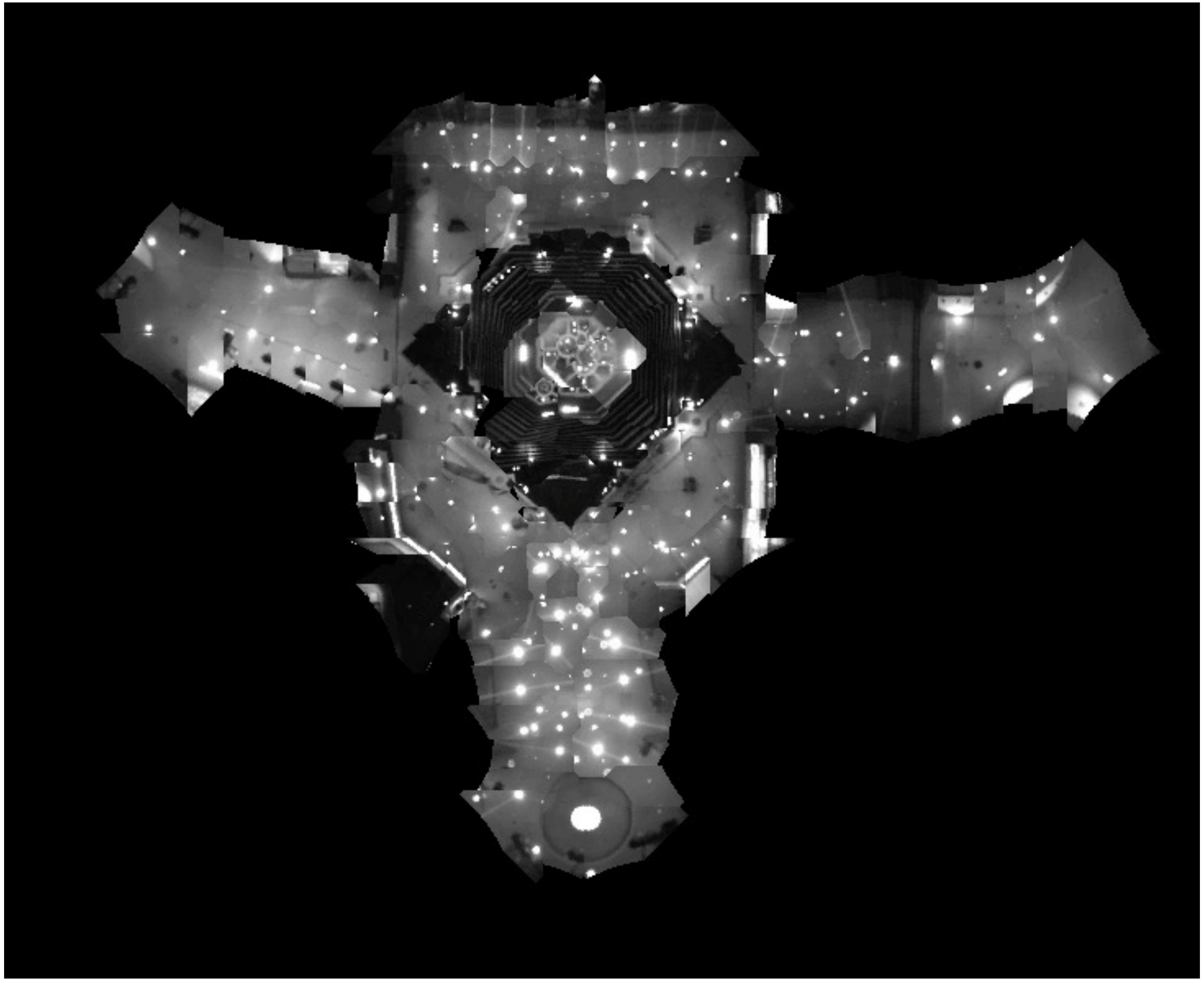
Elsewhere

Measurement z:

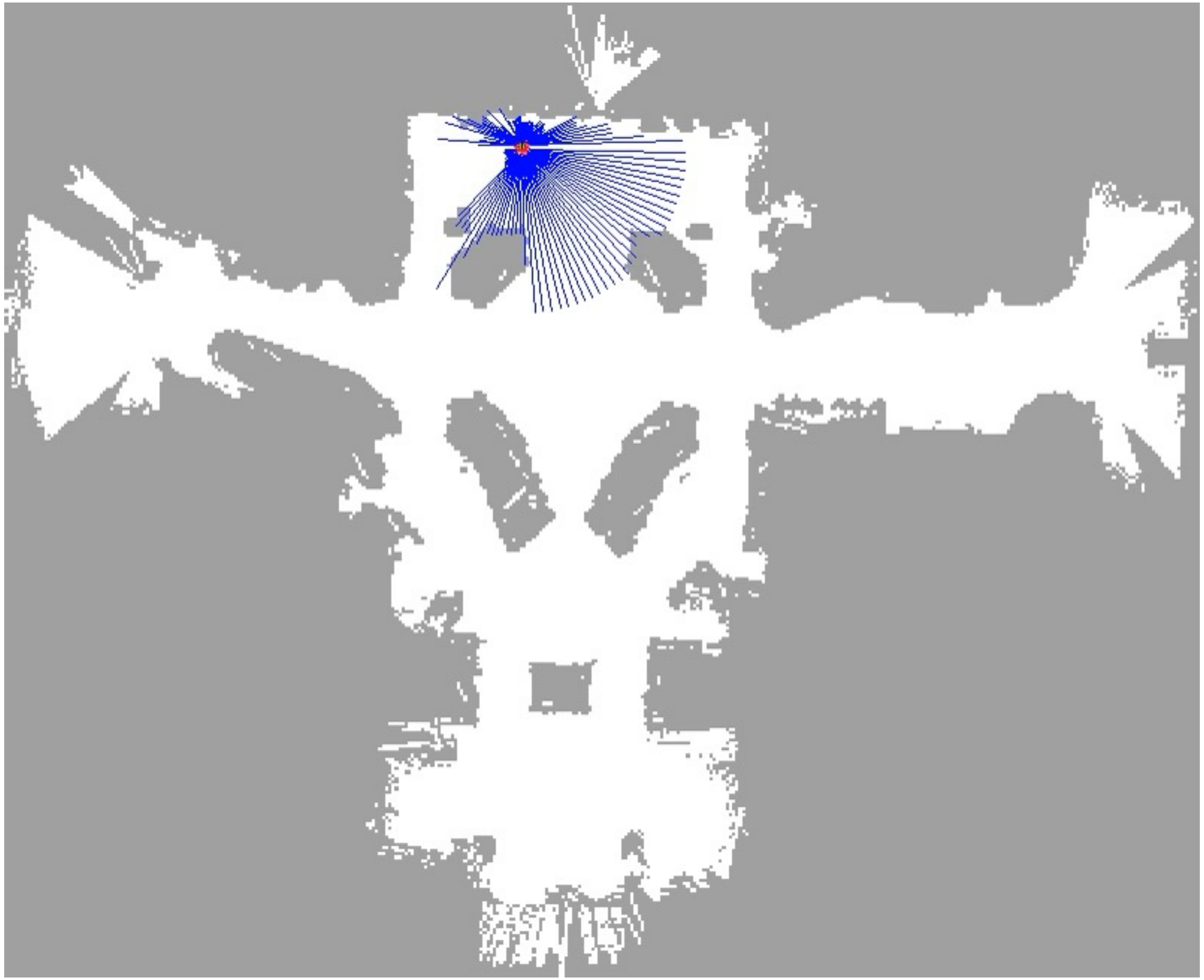
***P(z|x)*:**



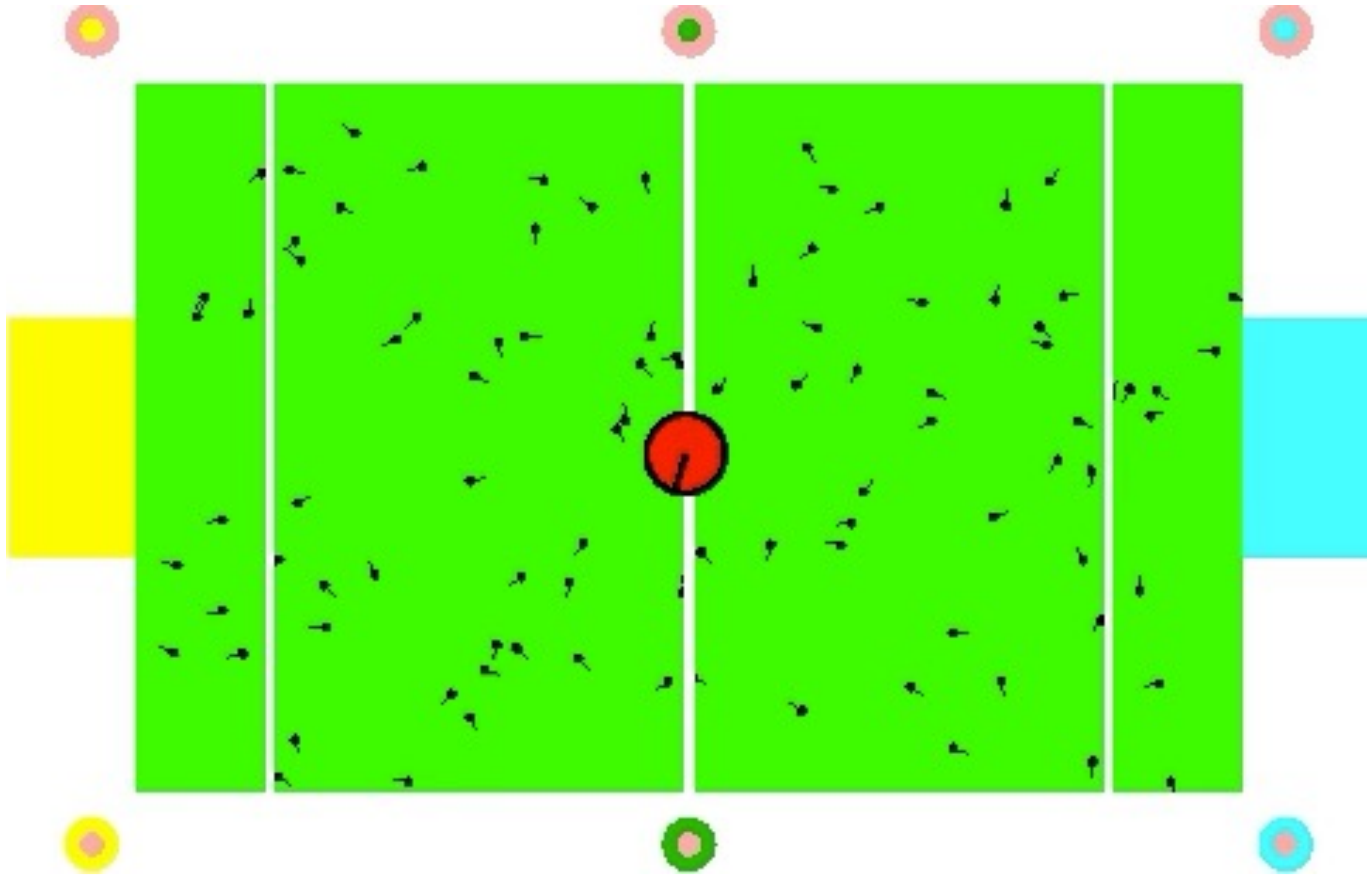
Global Localization Using Vision



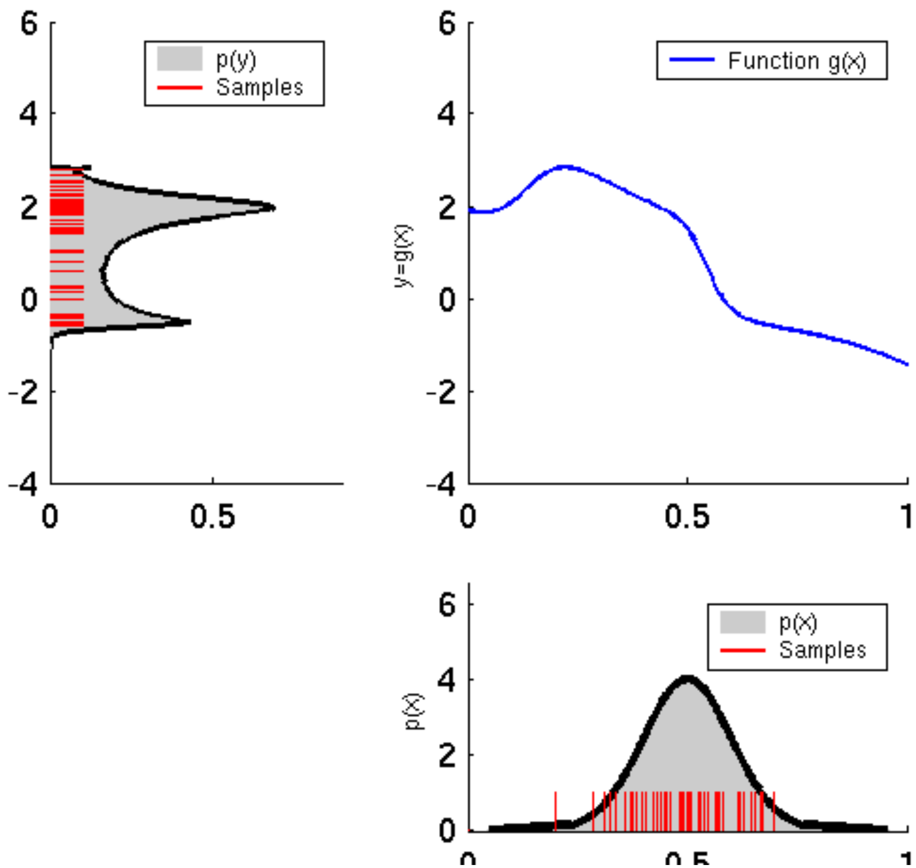
Recovery from Failure



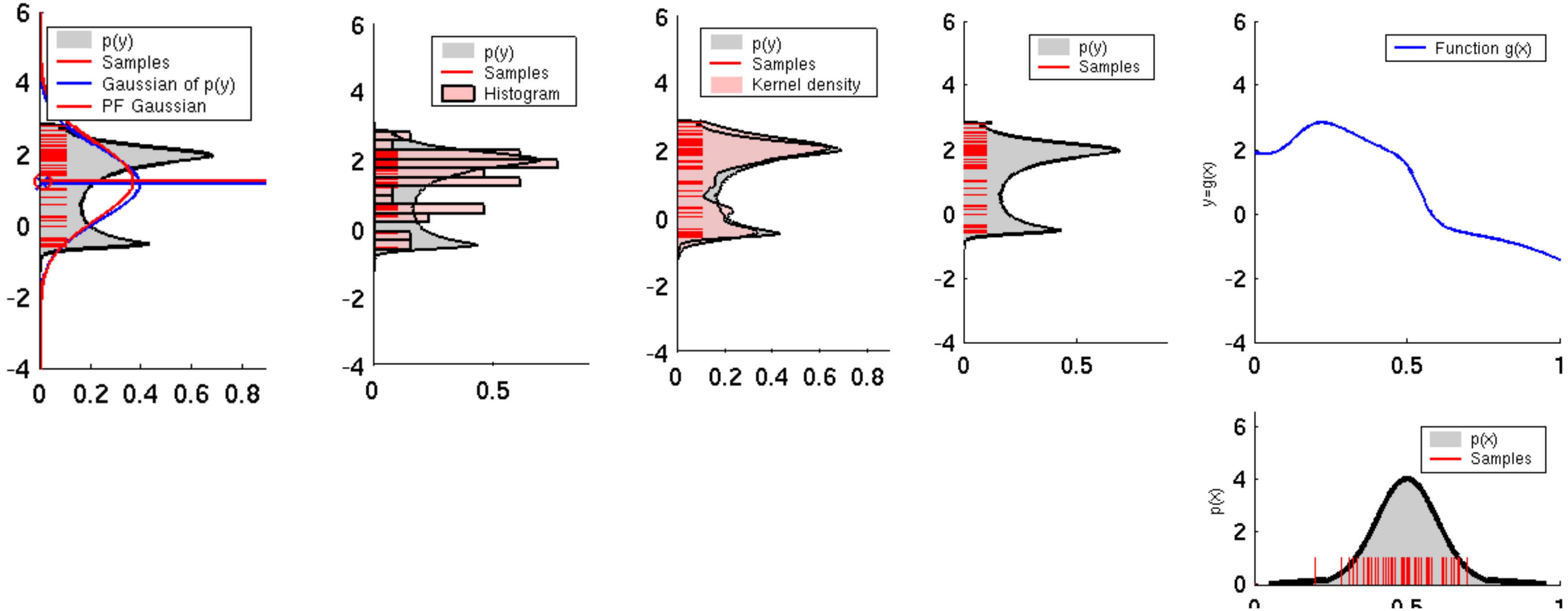
Localization for AIBO robots



Particle Filter Projection



Density Extraction



When might the particle filter fail?

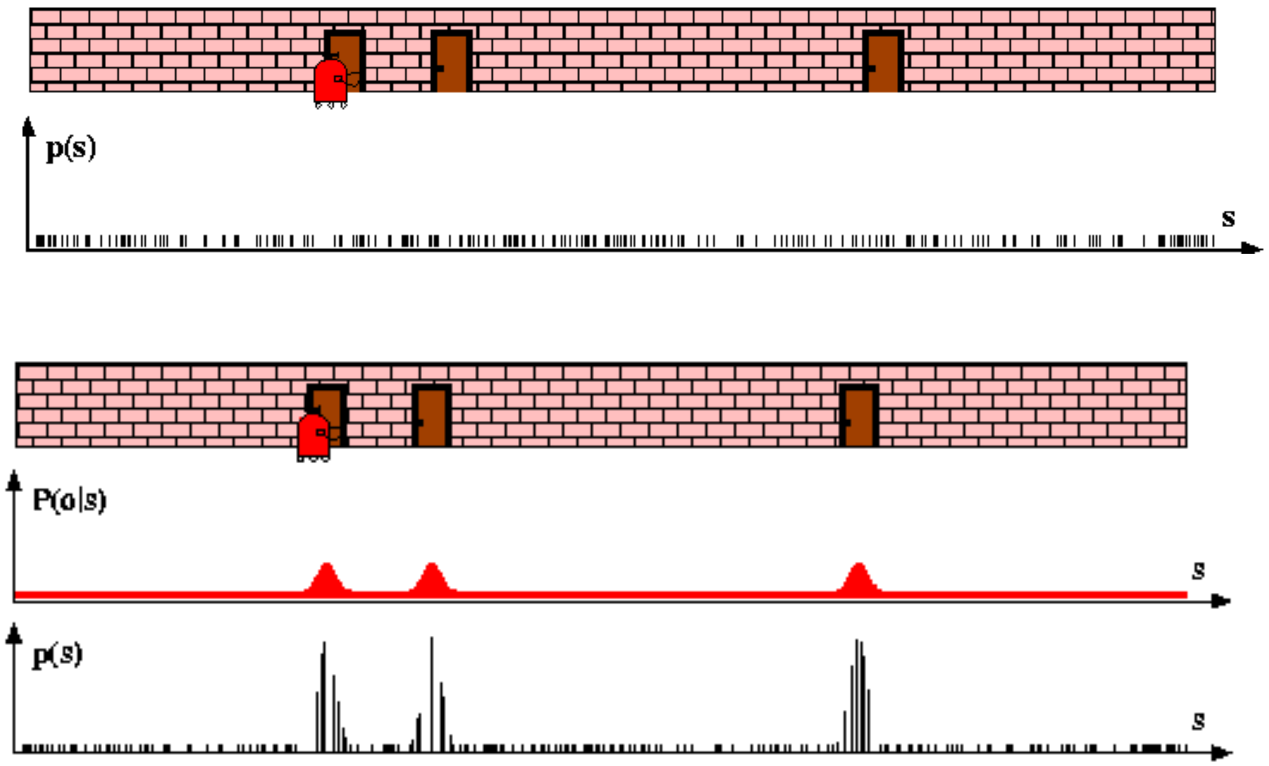
Why might this not work?

- Finitely many samples → introduces bias
- Variance of resampling operation → drops diversity
- Divergence of proposal and target distributions → degenerate importance weights
- Particle deprivation → belief collapse

Finite Numbers of Samples

Importance weights are very high variance for small numbers of particles

$$Bel(x_t) = \eta P(z_t|x_t)\overline{Bel}(x_t) \qquad w_i = \frac{P(z_t|x_t^i)}{\sum_j P(z_t|x_t^j)}$$



Imagine if there was 1 particle

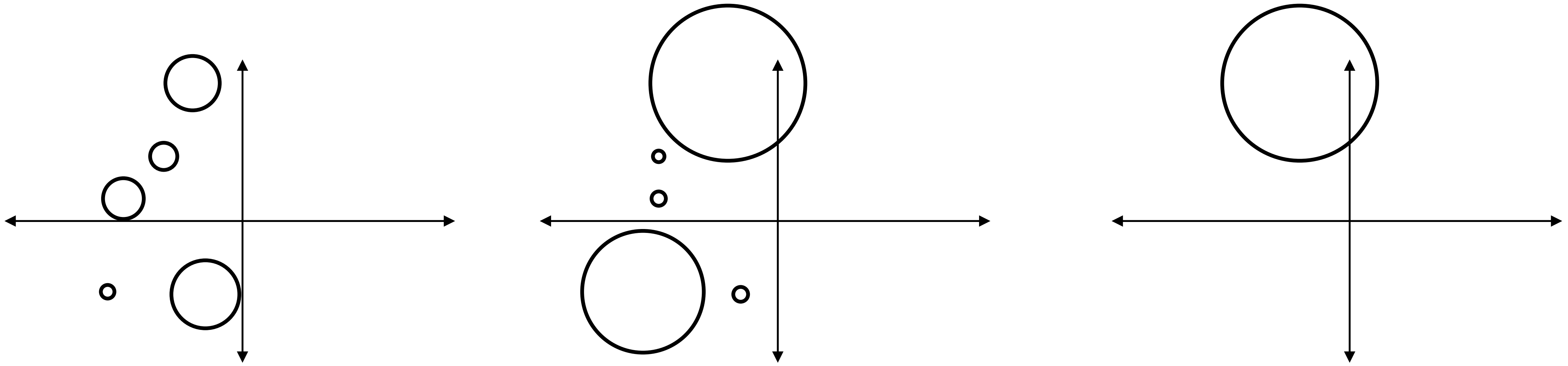
→ Evidence not taken into account at all

→ Low samples cause bias

Variance of Resampling Operation

Imagine the robot didn't move at all, just evidence and resampling

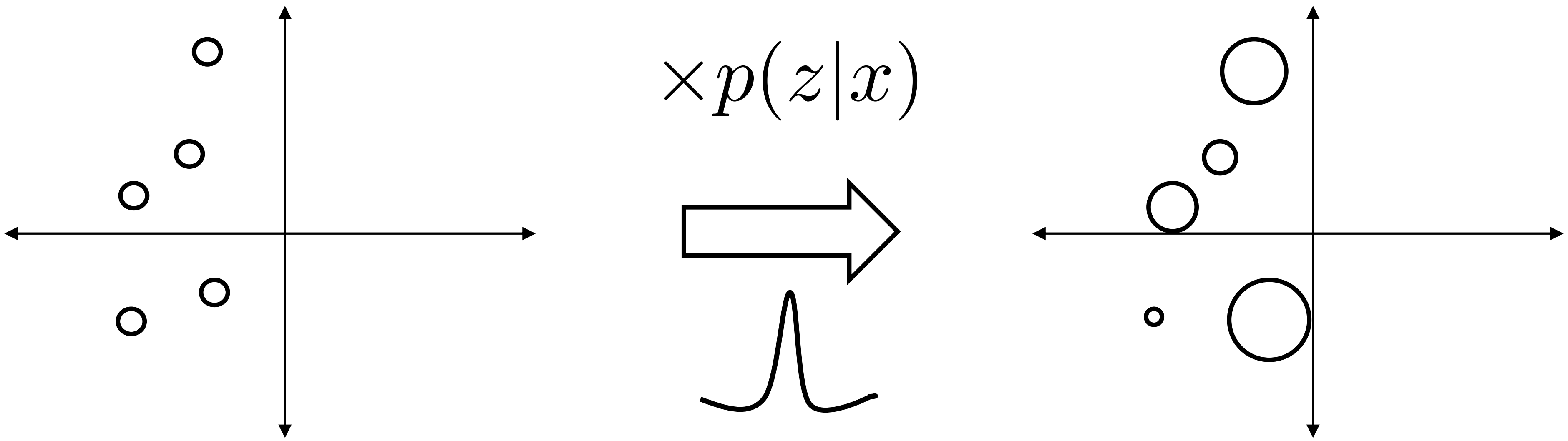
→ Collapses to a single particle with probability 1



Solution: resample less often or use lower variance sampling like SUS

Divergence of Proposal and Target

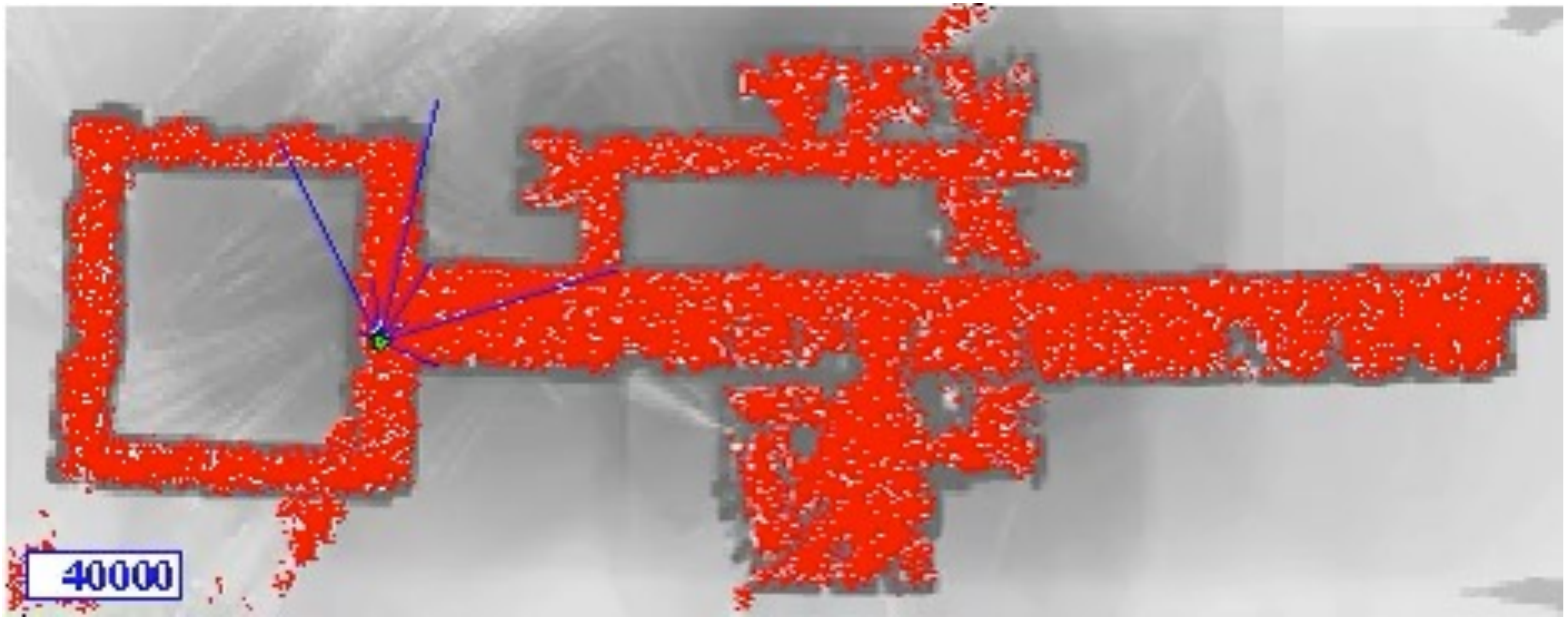
Sharp measurement models result in almost all 0 weights



Add fake noise into the measurement model

How can we do better? → be adaptive!

Adapt the number of particles generated during resampling according to likelihood



KLD-Sampling: Adaptive Particle Filters

Dieter Fox
Department of Computer Science & Engineering
University of Washington
Seattle, WA 98195
Email: fox@cs.washington.edu

Lecture Outline

Unscented Kalman Filter

↓

Discrete Bayesian Filters

↓

Particle Filters

Recap: Course Overview

- Filtering/Smoothing
- Localization
- Mapping
- SLAM

- Search
- Motion Planning
- TrajOpt
- Stability/Certification

- MDPs and RL
- Imitation Learning
- Solving POMDPs