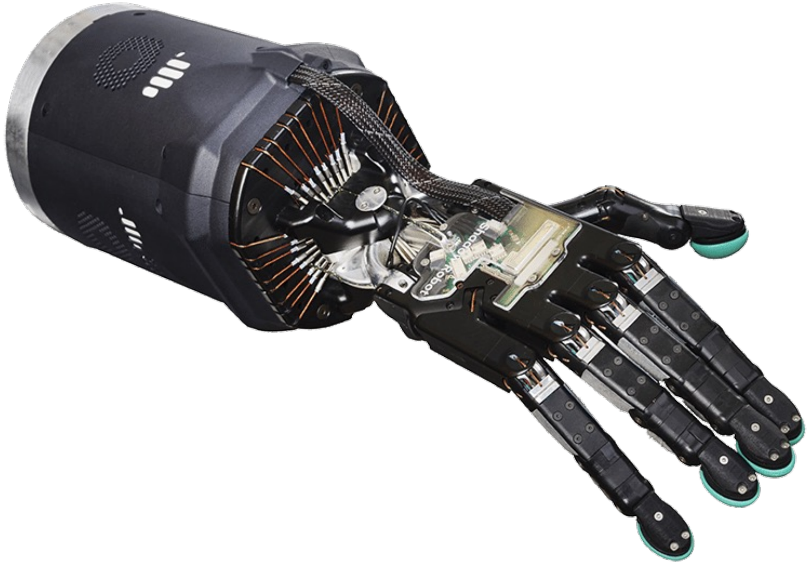
Autonomous Robotics

Winter 2025

Abhishek Gupta

TAs: Carolina Higuera, Entong Su, Bernie Zhu



Class Outline

State Estimation
- Robotic System Design
- Filtering
- Localization
- SLAM

Control
- Feedback Control
- PID Control
- MPC
- LQR

Planning
- Search
- Heuristic Search
- Motion Planning
- Lazy Search

Learning
- Imitation Learning
- Policy Gradient
- Actor-Critic
- Model-Based RL

Logistics

- Project 1 due on Jan 21 EOD
- Project 2 released next week

- Post questions, discuss any issues you are having on Ed.
- Students with **no** access to 002, e-mail us with your student ID.
- Students that have not been added to the class, email abhgupta@cs.washington.edu with the subject-line "Waitlisted for CSE478"

Recap

Motion Model

$$\theta_t = \theta_{t-1} + \Delta\theta = \theta_{t-1} + \frac{v}{L}\tan\delta\Delta t$$

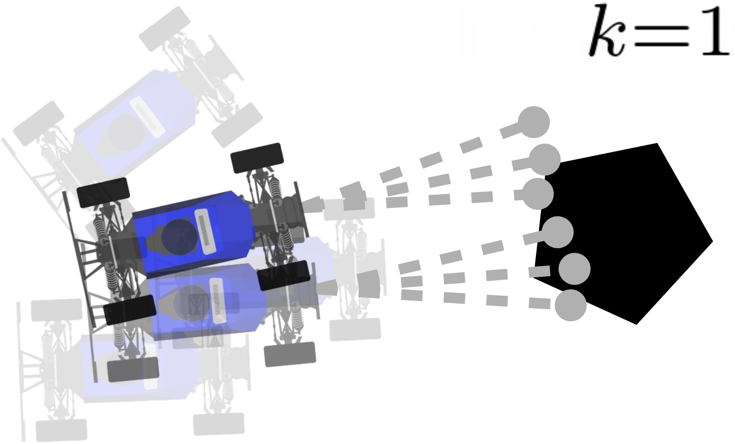
$$x_t = x_{t-1} + \Delta x = x_{t-1} + \frac{L}{\tan\delta}(\sin\theta_t - \sin\theta_{t-1})$$

$$y_t = y_{t-1} + \Delta y = y_{t-1} + \frac{L}{\tan\delta}(\cos\theta_{t-1} - \cos\theta_t)$$

Measurement Model

$$P(z_t|x_t, m) = P(z_t^1, z_t^2, \cdots, z_t^K|x_t, m)$$

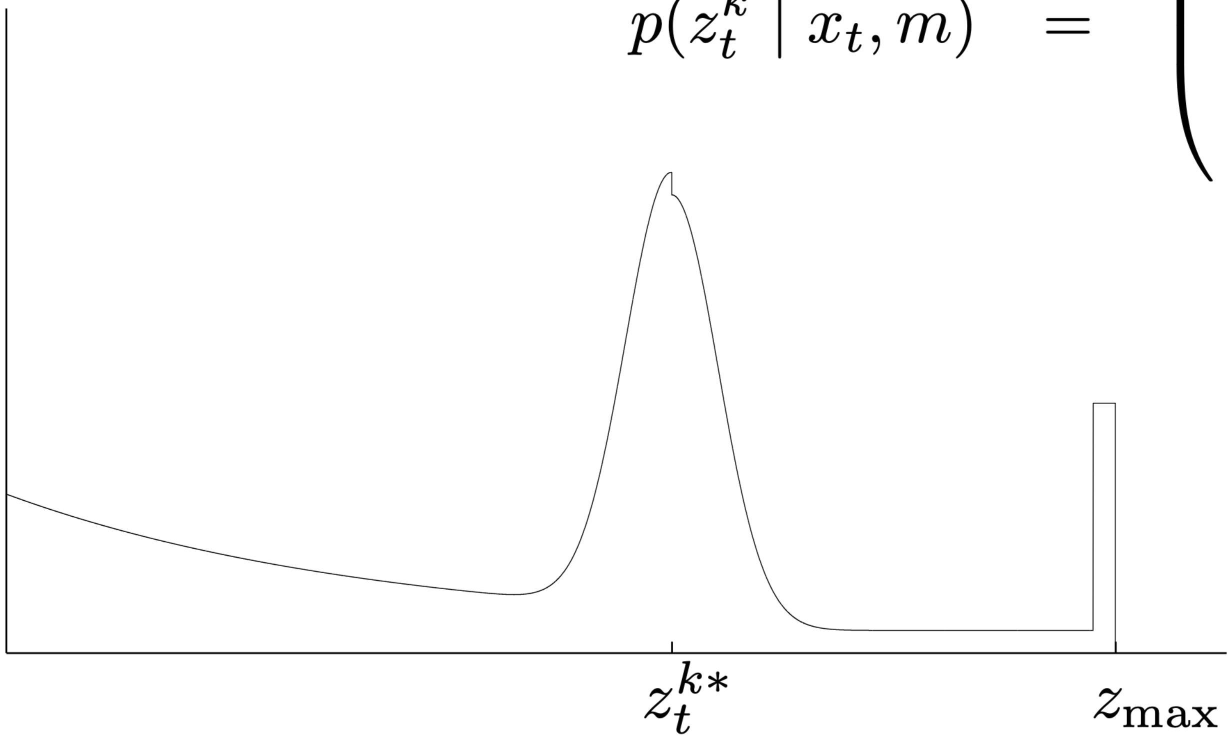
$$= \prod_{k=1}^{K} P(z_t^k|x_t, m)$$



Measurement Model

$$p(z_t^k \mid x_t, m) = \begin{pmatrix} z_{\text{hit}} \\ z_{\text{short}} \\ z_{\max} \\ z_{\text{rand}} \end{pmatrix}^T \cdot \begin{pmatrix} p_{\text{hit}}(z_t^k \mid x_t, m) \\ p_{\text{short}}(z_t^k \mid x_t, m) \\ p_{\max}(z_t^k \mid x_t, m) \\ p_{\text{rand}}(z_t^k \mid x_t, m) \end{pmatrix}$$

Weights sum to 1



Lecture Outline

Recap

↓

Motion Models

↓

Observation Models

↓

Particle Filtering

Why is the Bayes filter challenging to implement?

Key Idea: Apply Markov to get a recursive update!

Step 0. Start with the belief at time step t-1

$$bel(x_{t-1})$$

Step 1: Prediction - push belief through dynamics given action

$$\overline{bel}(x_t) = \sum P(x_t|u_t, x_{t-1})bel(x_{t-1})$$

Step 2: Correction - apply Bayes rule given measurement

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

Intractable due to discretization

How does discretization work for Bayesian filters?

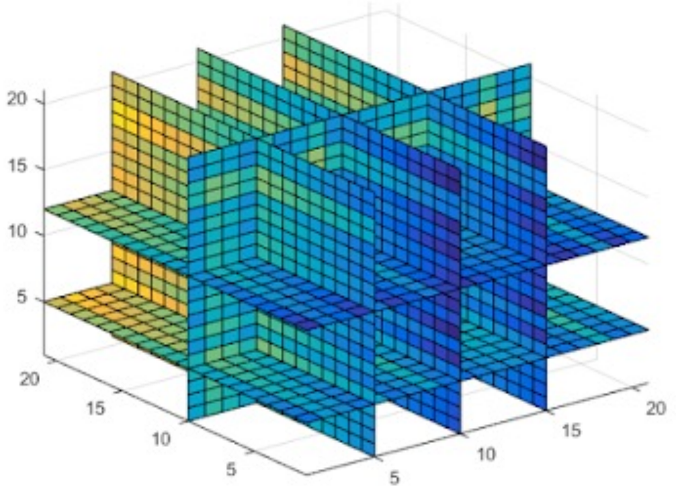
$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

X-COORDINATE - Discretize into K bins

Y-COORDINATE - Discretize into K bins

HEADING - Discretize into K bins

Overall K^3 bins

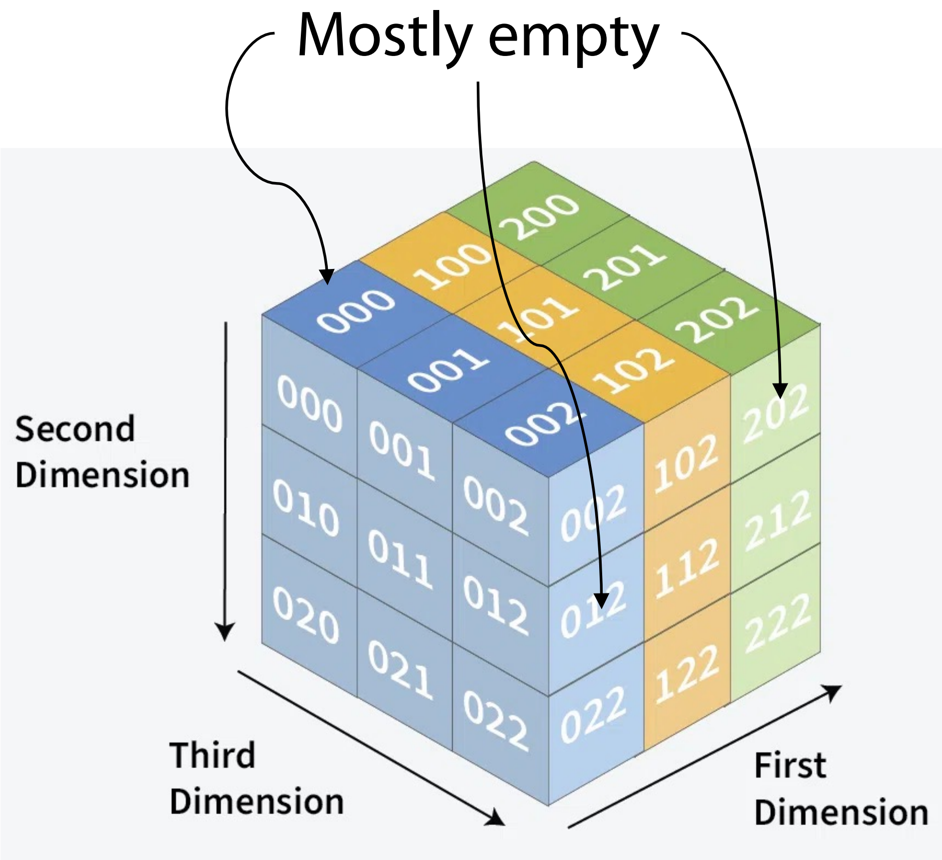


Exponentially expensive with dimension for each summation

Many of these bins will be empty!

How can we do better?

Let’s change our way of thinking



$$[s_1, s_1, s_2, s_{10}, s_{40}, s_{40}, s_{40}, s_{55}, s_{55}]$$

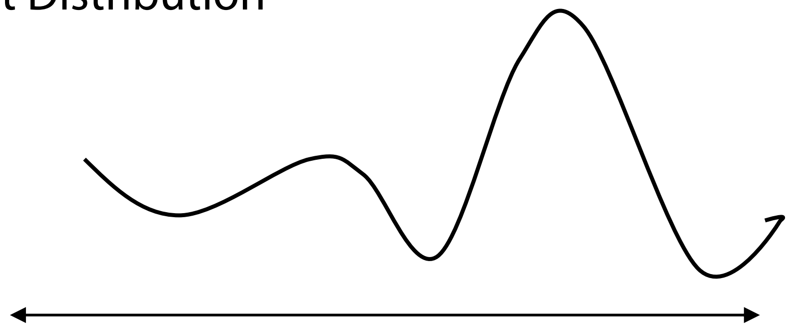
Keep a list of only the states with likelihood, with number of repeat instances proportional to probability

No discretization per dimension!

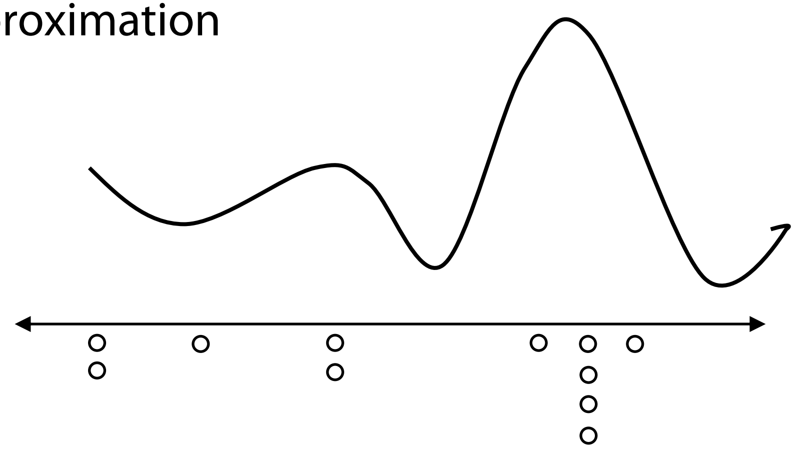
Is this even a useful/valid representation of belief?

Let’s change our way of thinking

Target Distribution



”Particle” Approximation



Is this even a useful/valid representation of belief?

Depends what we want to do with the probability distribution!

→ Typically we want to compute averages (expectations)

Downstream Usage of Estimated Probability Distributions

What do we actually intend to do with the belief $bel(x_{t+1})$?

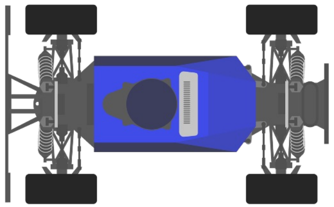
→ Often times we will be evaluating the expected value

$$\mathbb{E}[f] = \int_x f(x) bel(x) dx$$

Mean position: $f(x) \equiv x$

Probability of collision: $f(x) \equiv \mathbb{I}(x \in \mathcal{O})$

Mean value / cost-to-go: $f(x) \equiv V(x)$



Computing Expectations without Closed Form Likelihoods

Monte-Carlo Simulation

$$\mathbb{E}_{x \sim Bel(x_t)}[f(x)] = \int_x f(x) Bel(x) dx \approx \sum_x f(x) Bel(x)$$

Sample from the belief: $x_1, \cdots, x_N \sim Bel(x_t)$

$$\mathbb{E}_{x \sim Bel(x_t)}[f(x)] \approx \frac{1}{N} \sum_i^N f(x^{(i)})$$

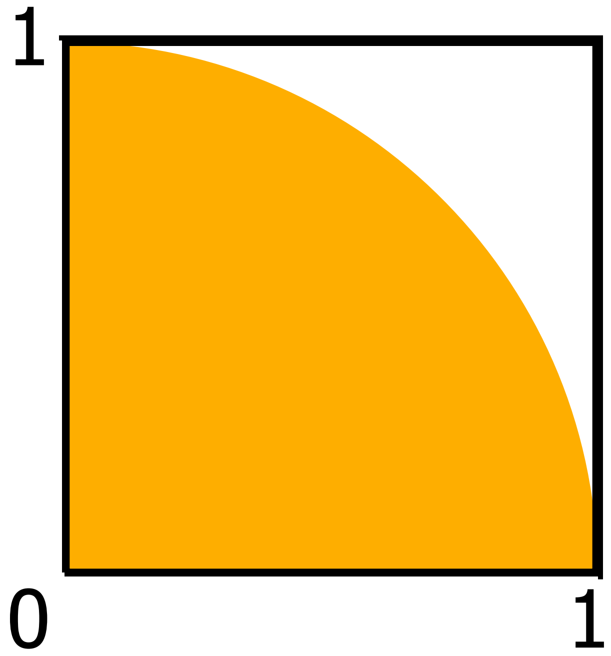
Don’t require closed form distributions (Gaussian/Beta, etc), just samples (particles)!

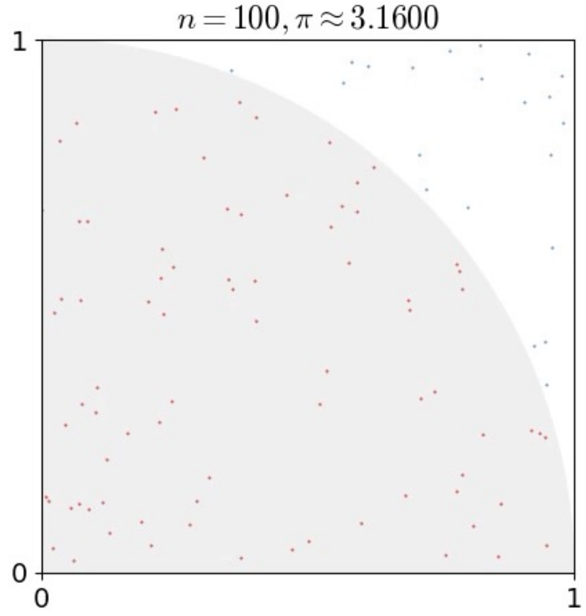
→ Replace fancy math by brute force simulation!!

Examples of Monte Carlo Estimation

$$\mathbb{E}[\mathbb{I}(x \in \mathcal{O})] = P(x \in \mathcal{O}) = \frac{\pi}{4} \approx \frac{1}{N} \sum \mathbb{I}(x^{(i)} \in \mathcal{O})$$

1. Sample points uniformly from unit square
2. Count number in quarter-circle (i.e. $\|x_i\| \leq 1$)
3. Divide by N, multiply by 4





→ Exercise: What are other practical problems where this is useful?

ADAPTED FROM WIKIPEDIA

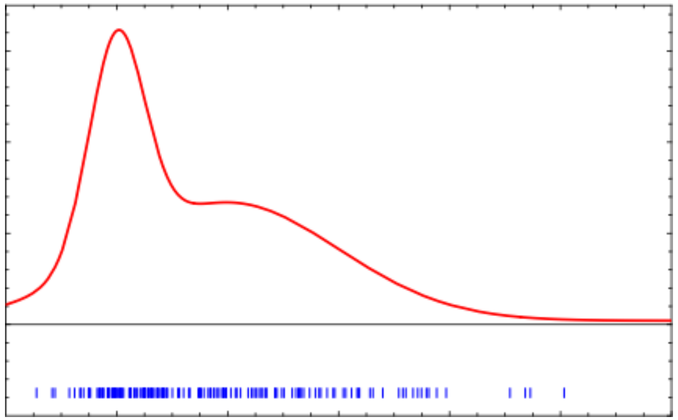
Bringing this Back to Estimation – Belief Distribution

Let's consider the Bayesian filtering update

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

Represent the belief with a set of particles! Each is a hypothesis of what the state might be.

Higher likelihood regions have more particles



How do we “propagate” belief across timesteps with particles?

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

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

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

How do we sample from the product of two distributions?

How do we compute conditioning/normalization with particles?

Lecture Outline

Particle Based Representations in Filtering

↓

Particle Filter

↓

Particle Filter w/ Resampling

↓

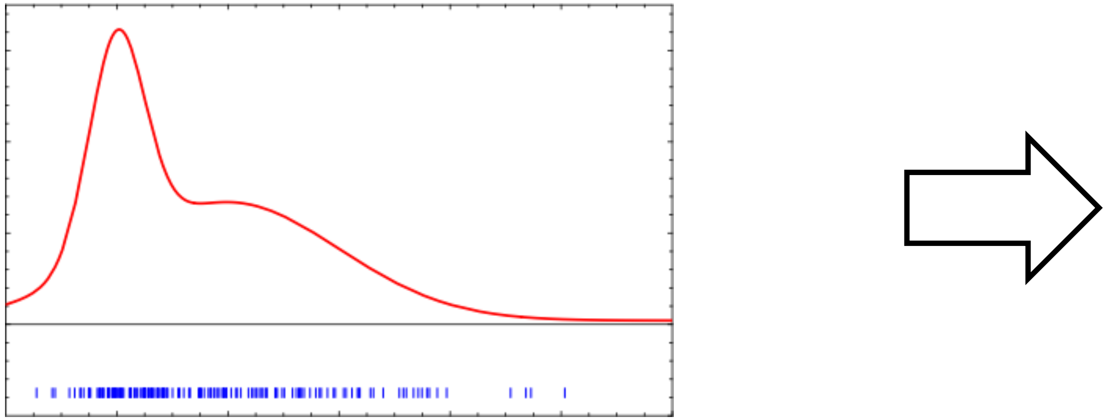
Practical Considerations

Dynamics Step: Propagating Belief Through Dynamics

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

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

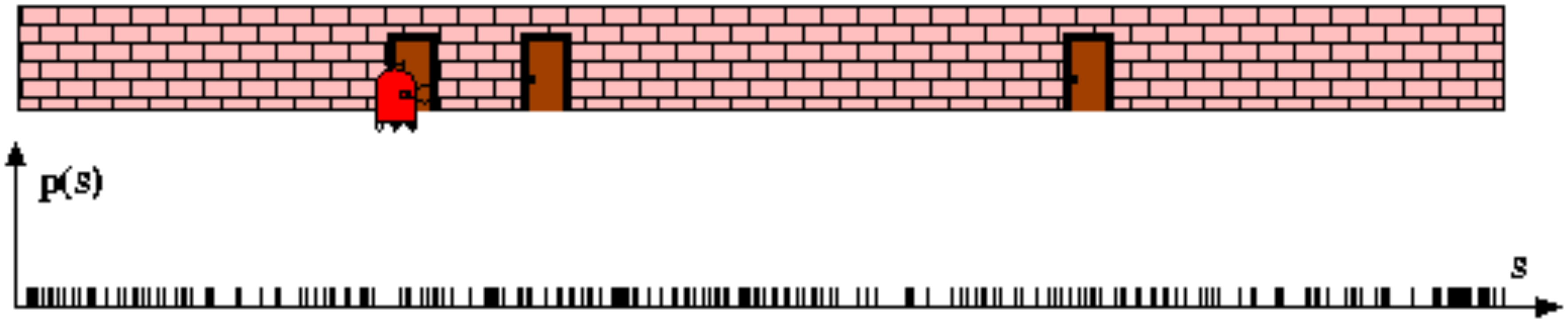
How do we sample from the product of two distributions?



???

Treat each particle as point estimate of actual state and propagate through the dynamics

Propagating Belief Through Dynamics: Initial

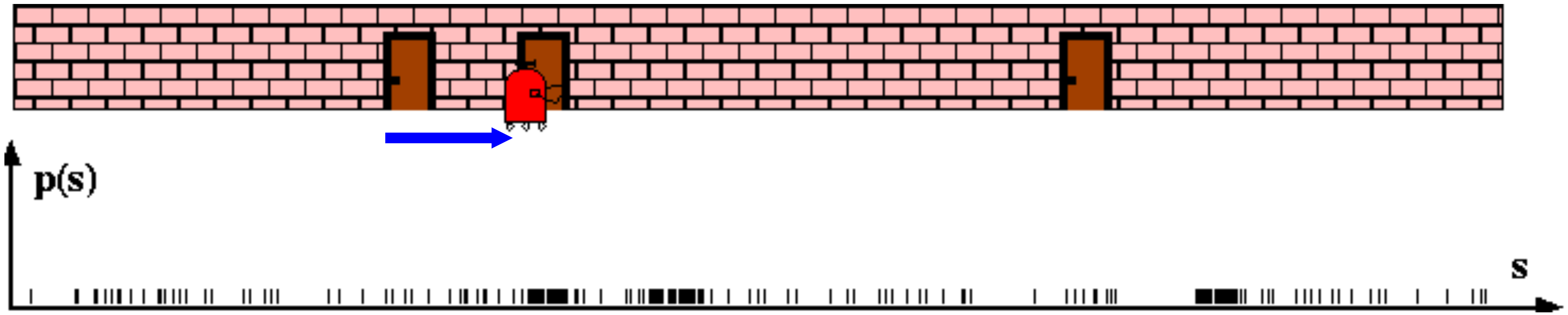


Propagating Belief Through Dynamics: Robot Motion

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

Push samples forward according to dynamics

Take every x_{t-1} in previous belief, run motion model forward with x_{t-1} and u_t to get new particles

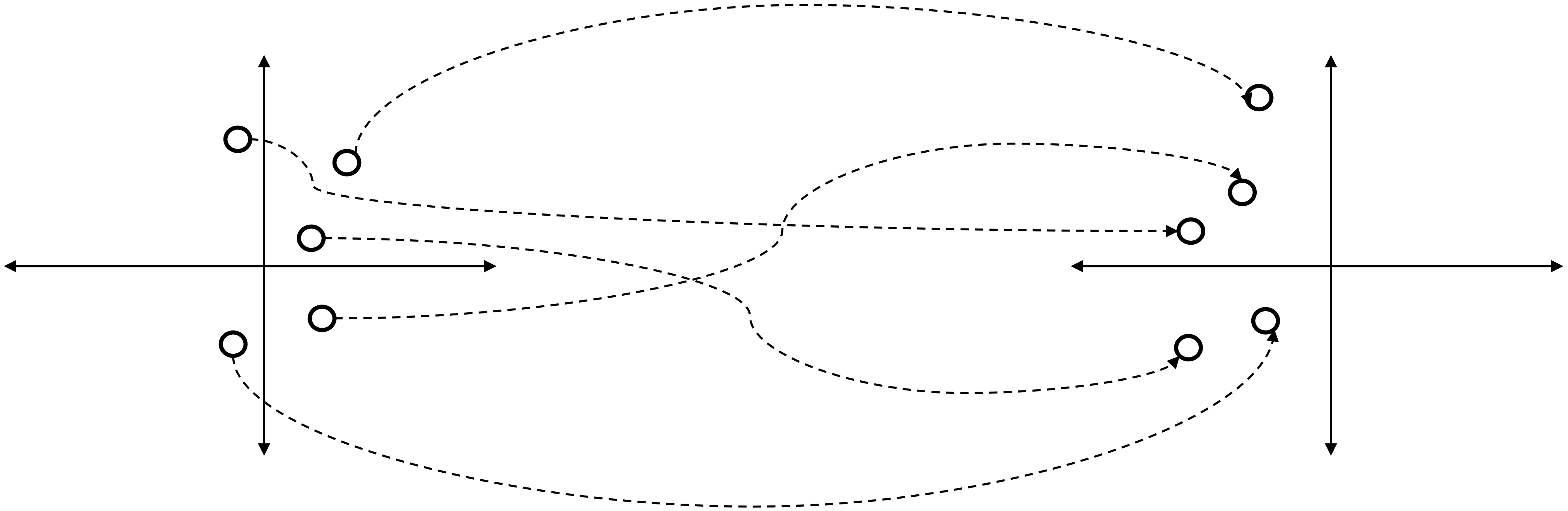


Dynamics Update:

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

Sample forward using the dynamics model:

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



How do we “propagate” belief across timesteps with particles?

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

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

↓

How do we compute conditioning/normalization with particles?

Sensor Information: Measurement Update

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

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

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

Weight each particle - Can compute a per sample weight.
Distribution represented as set of weighted samples

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

Not ad hoc! → exactly the same as importance sampling

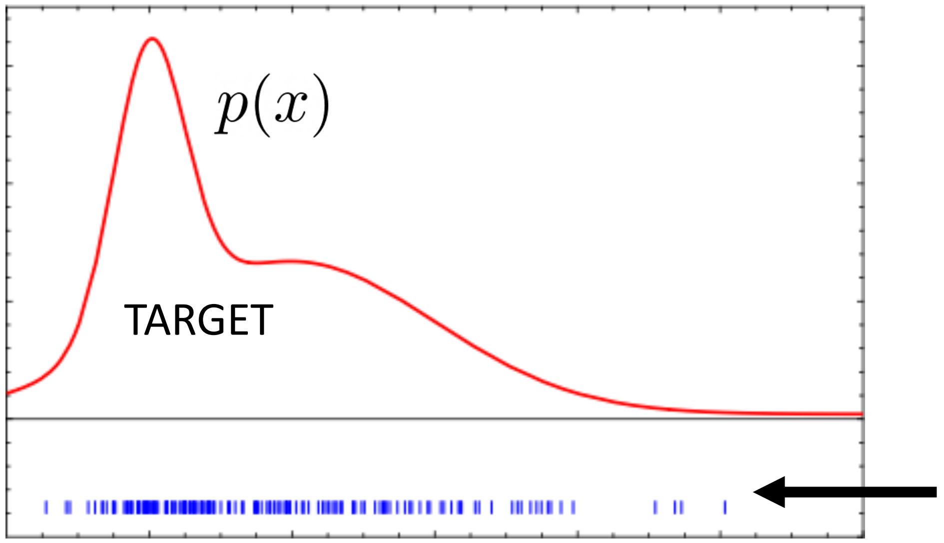
Detour: What is Importance Sampling?

How can we sample from a
“complex” distribution p(x) using a simple distribution q(x)?

Importance Sampling

1. Sample from an (easy) proposal distribution
2. Reweight samples to match the target distribution

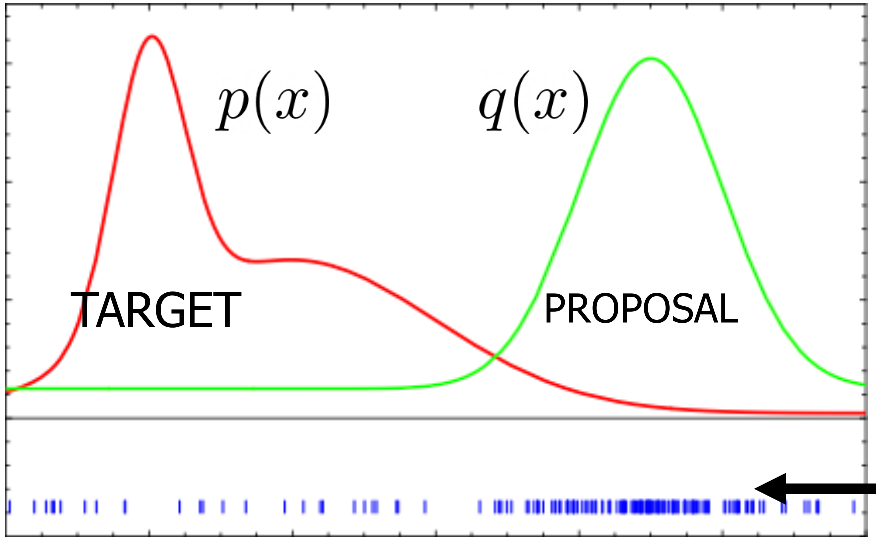
Importance Sampling



Don't know how to sample from target!

Importance Sampling

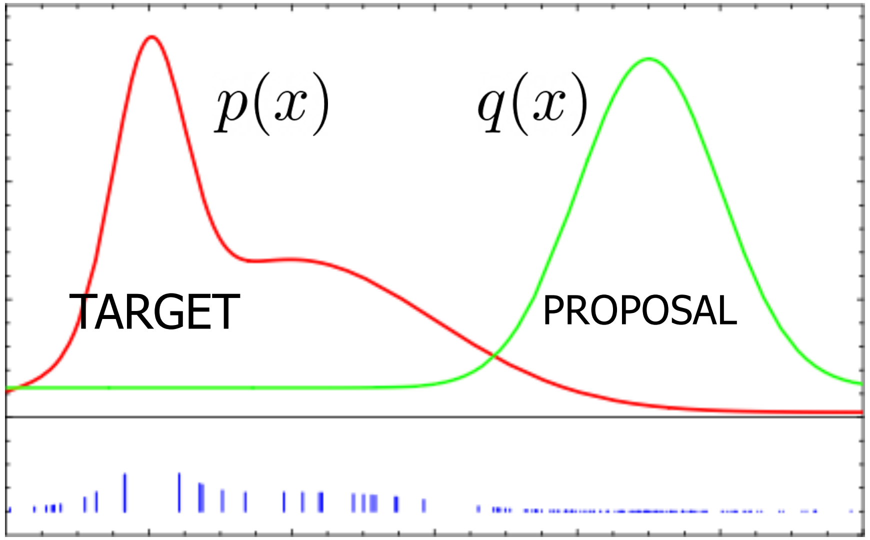
1. Sample from an (easy) proposal distribution



Can sample from proposal distribution

Importance Sampling

1. Sample from an (easy) proposal distribution
2. Reweight samples to match the target distribution



Importance Sampling

$$\mathbb{E}_{x\sim p(x)}[f(x)] = \sum p(x)f(x)$$

Expected value with p(x)

$$= \sum p(x)f(x)\frac{q(x)}{q(x)}$$

$$= \sum q(x)\frac{p(x)}{q(x)}f(x)$$

$$= \mathbb{E}_{x\sim q(x)}\left[\frac{p(x)}{q(x)}f(x)\right]$$

Expected value with q(x)

$$\approx \frac{1}{N}\sum_{i=1}^{N}\left[\frac{p(x^{(i)})}{q(x^{(i)})}f(x^{(i)})\right]$$

IMPORTANCE WEIGHT

Monte Carlo estimate

Measurement Update with Importance Sampling

Target Distribution: Posterior

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

p(x)

Proposal Distribution: After applying motion model

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

q(x)

Measurement Update with Importance Sampling

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

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

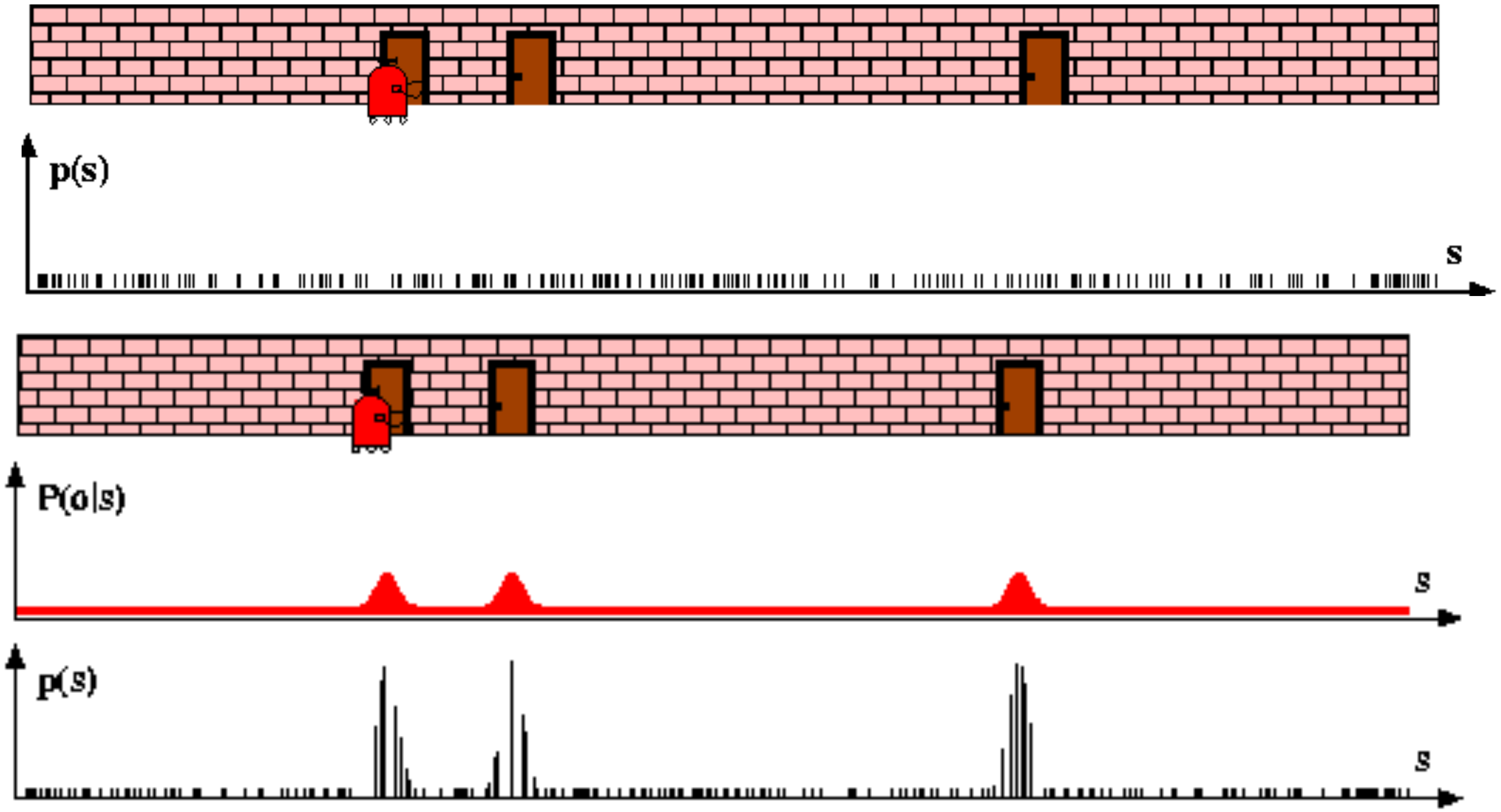
Importance Weight (Ratio)

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

Sensor Information: Importance Sampling

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

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

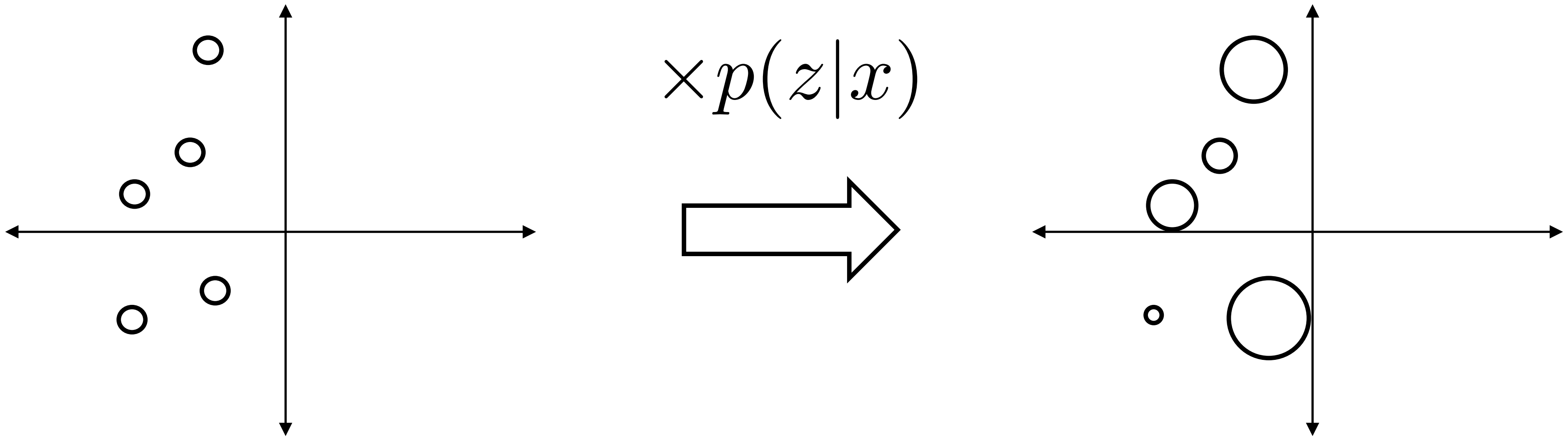


Measurement Update

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

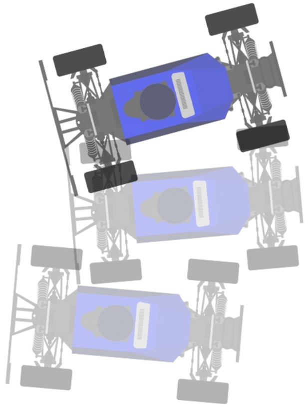
$$Bel(x_t) = \frac{P(z_t|x_t)\overline{Bel}(x_t)}{\int P(z_t|x_t)\overline{Bel}(x_t)dx_t}$$

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



Reweight particles according to measurement likelihood

Normalized Importance Sampling



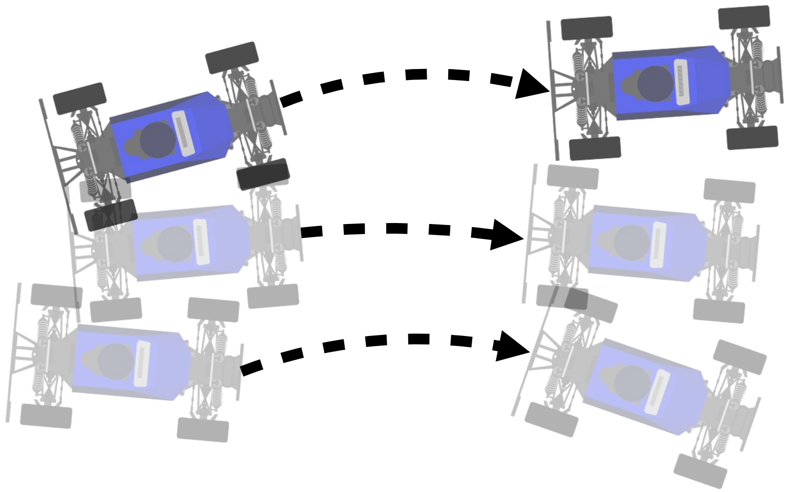
0.50

0.25

0.25

$$Bel(x_{t-1}) = \left\{ \begin{array}{cccc} x_{t-1}^{(1)} & x_{t-1}^{(2)} & \cdots & x_{t-1}^{(M)} \\ w_{t-1}^{(1)} & w_{t-1}^{(2)} & \cdots & w_{t-1}^{(M)} \end{array} \right\}$$

Normalized Importance Sampling



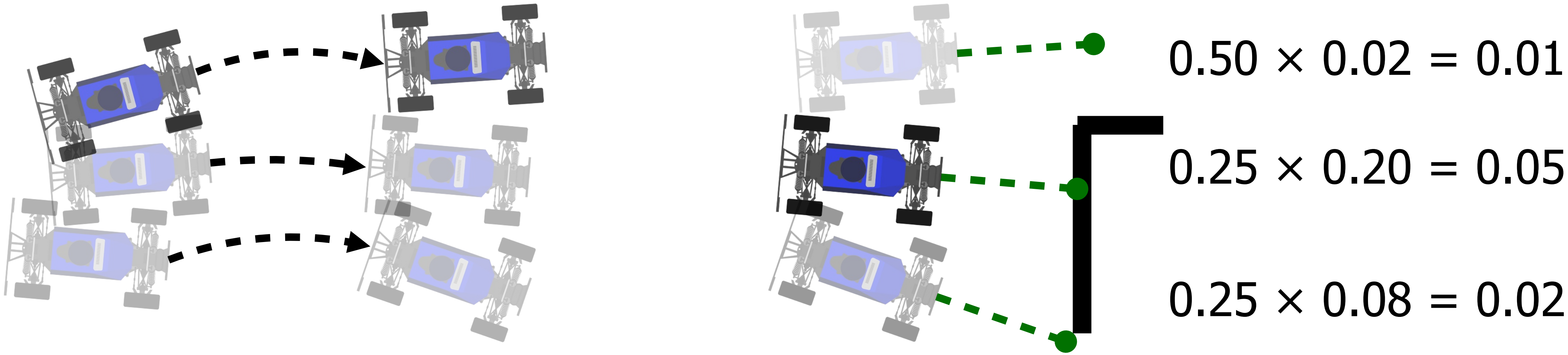
0.50

0.25

0.25

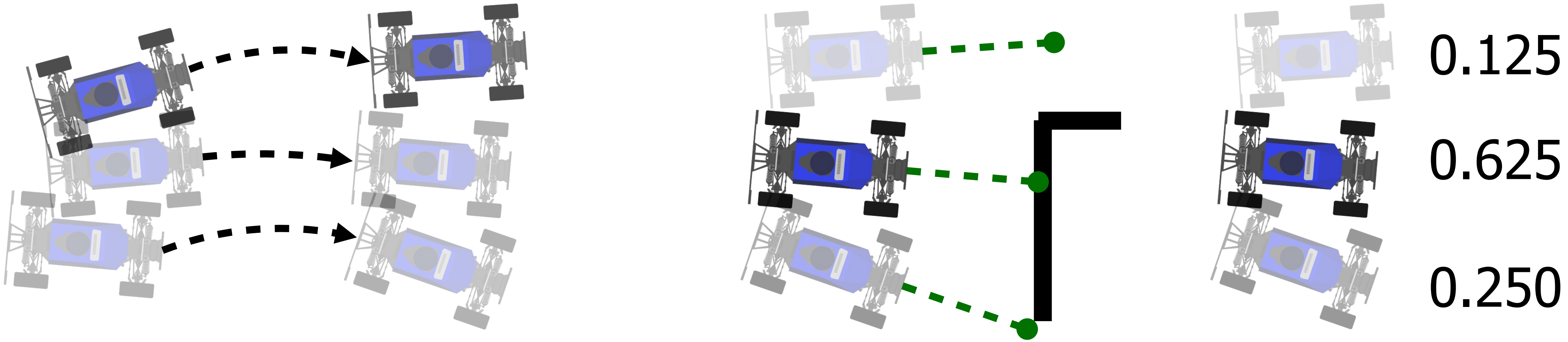
$$\bar{x}_t^{(i)} \sim P(x_t | u_t, x_{t-1})$$

Normalized Importance Sampling



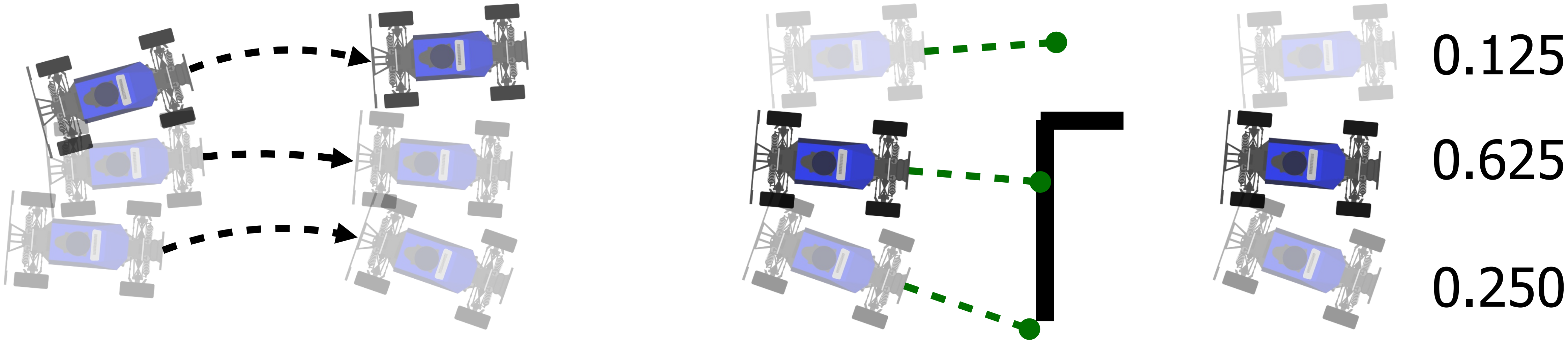
$$w_t^{(i)} = P(z_t|\bar{x}_t^{(i)})w_{t-1}^{(i)}$$

Normalized Importance Sampling



$$w_t^{(i)} = \frac{w_t^{(i)}}{\sum_i w_t^{(i)}}$$

Normalized Importance Sampling

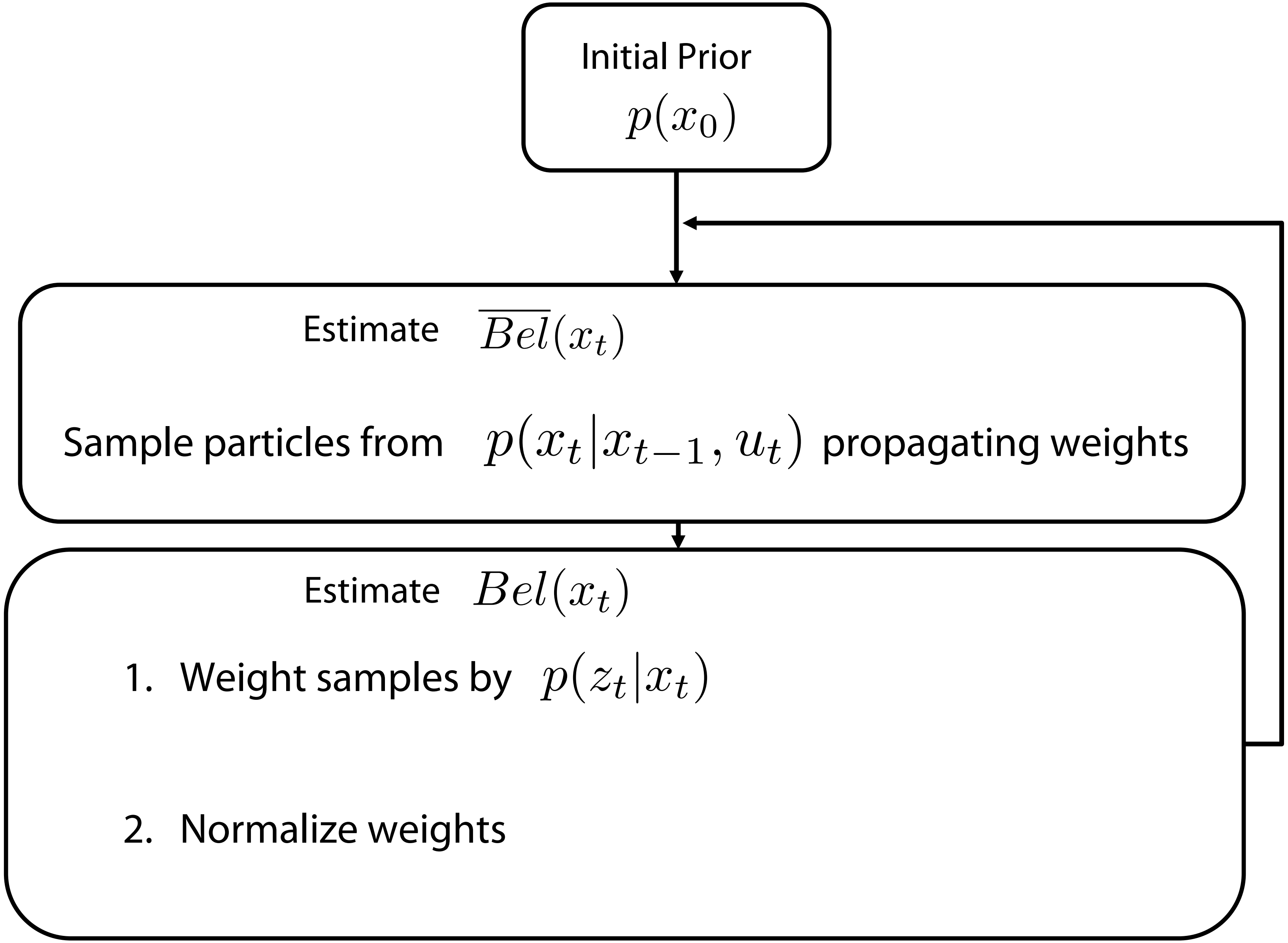


$$Bel(x_t) = \left\{ \begin{array}{cccc} \bar{x}_t^{(1)} & \bar{x}_t^{(2)} & \cdots & \bar{x}_t^{(M)} \\ w_t^{(1)} & w_t^{(2)} & \cdots & w_t^{(M)} \end{array} \right\}$$

Overall Particle Filter algorithm – v1

Dynamics/Prediction

Measurement/Correction



Lecture Outline

Particle Based Representations in Filtering

↓

Particle Filter

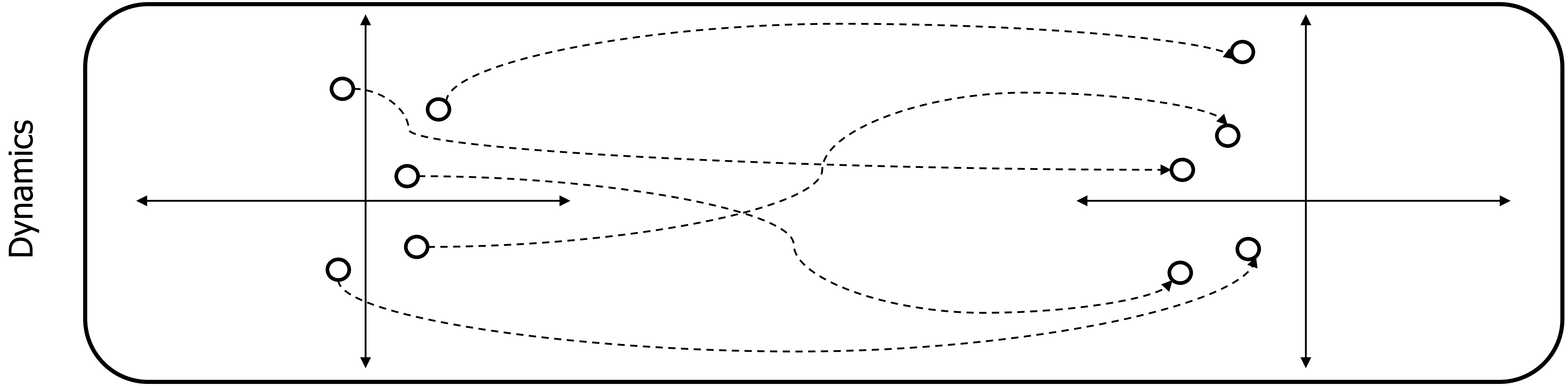
↓

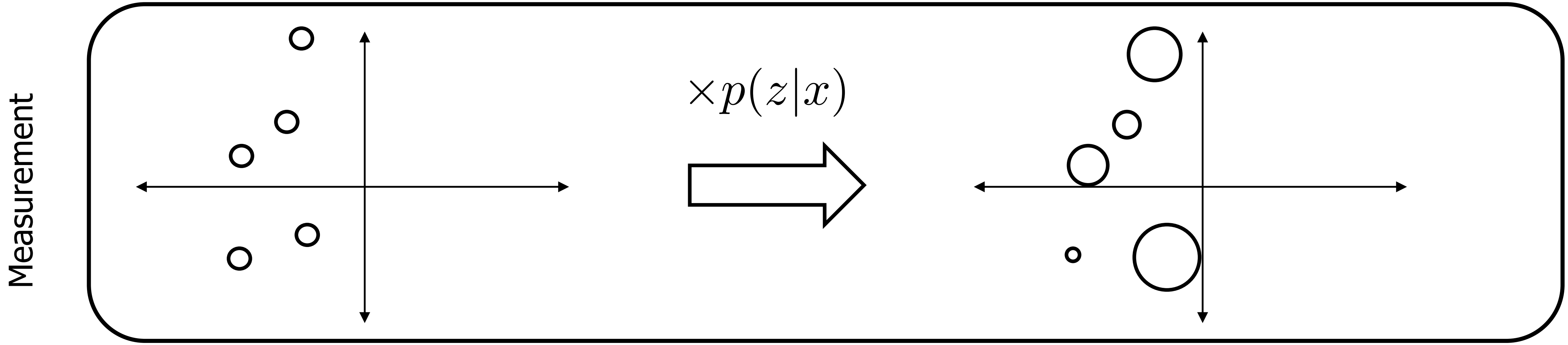
Particle Filter w/ Resampling

↓

Practical Considerations

What happens across multiple steps?

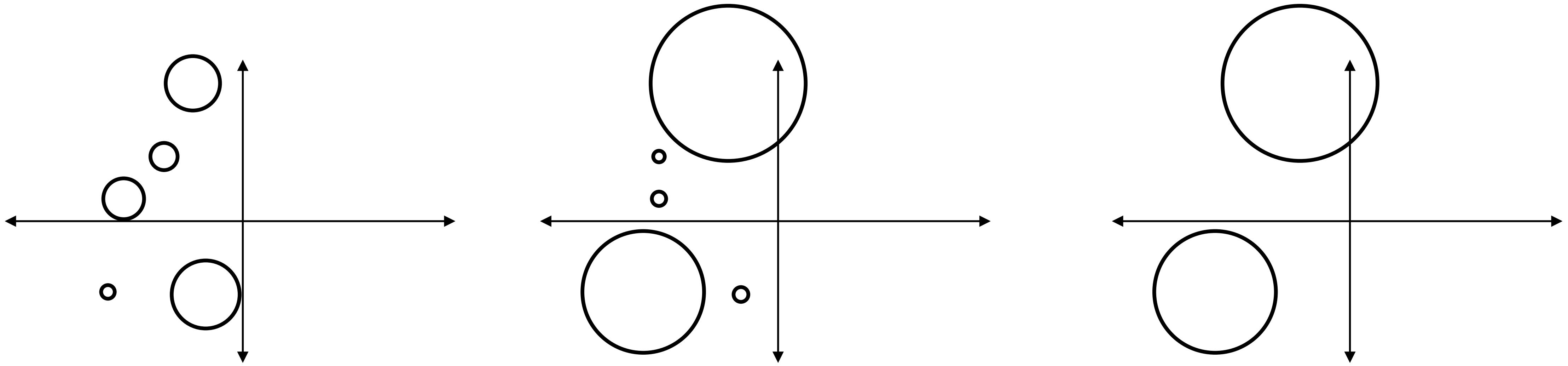




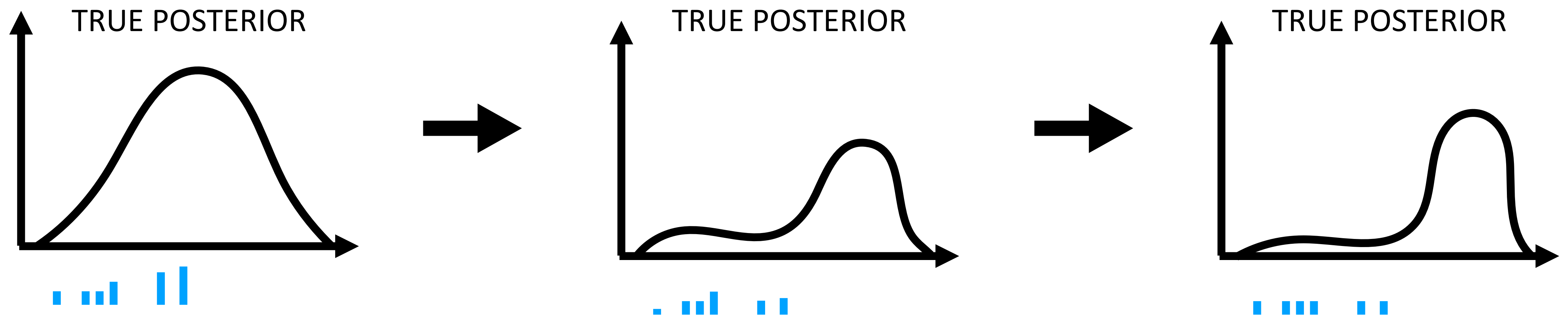
Importance weights get multiplied at each step

Why might this be bad?

Importance weights get multiplied at each step



1. May blow up and get numerically unstable over many steps
2. Particles stay stuck in unlikely regions



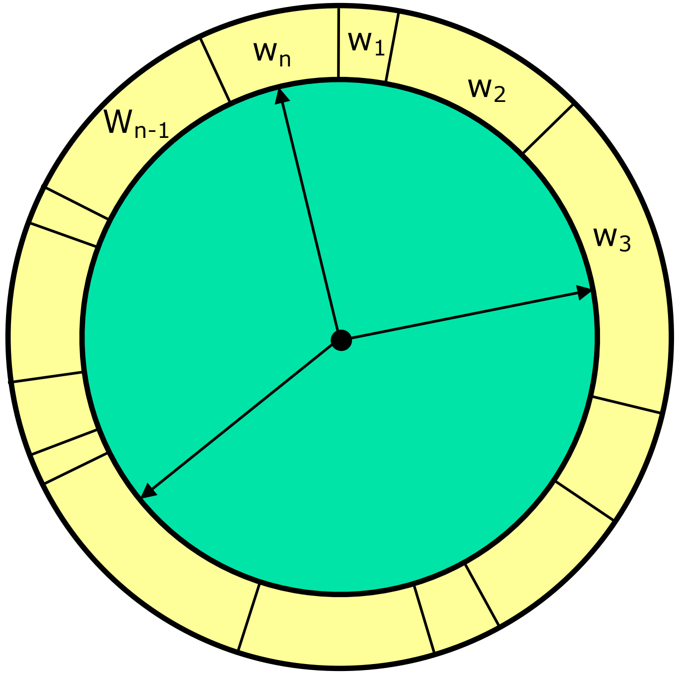
Resampling

- **Given**: Set ***S*** of weighted samples (from measurement step) with weights w_i

- **Wanted** : unweighted random sample, where the probability of drawing $\boldsymbol{x_i}$ is given by $\boldsymbol{w_i}$.

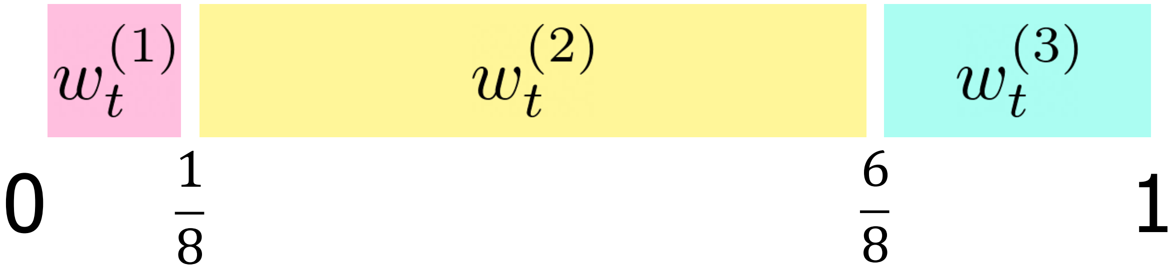
- Typically done ***n*** times with replacement to generate new sample set ***S'***.

Resampling



Here are your random numbers:

```
0.97
0.26
0.72
```



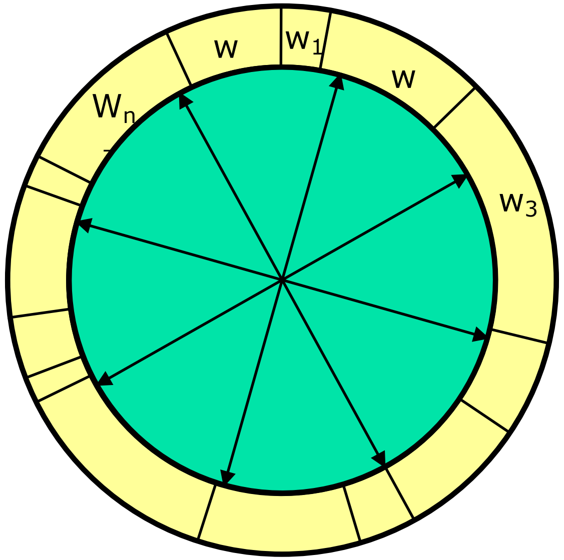
- Spin a roulette wheel
- Space according to weights
- Pick samples based on where it lands

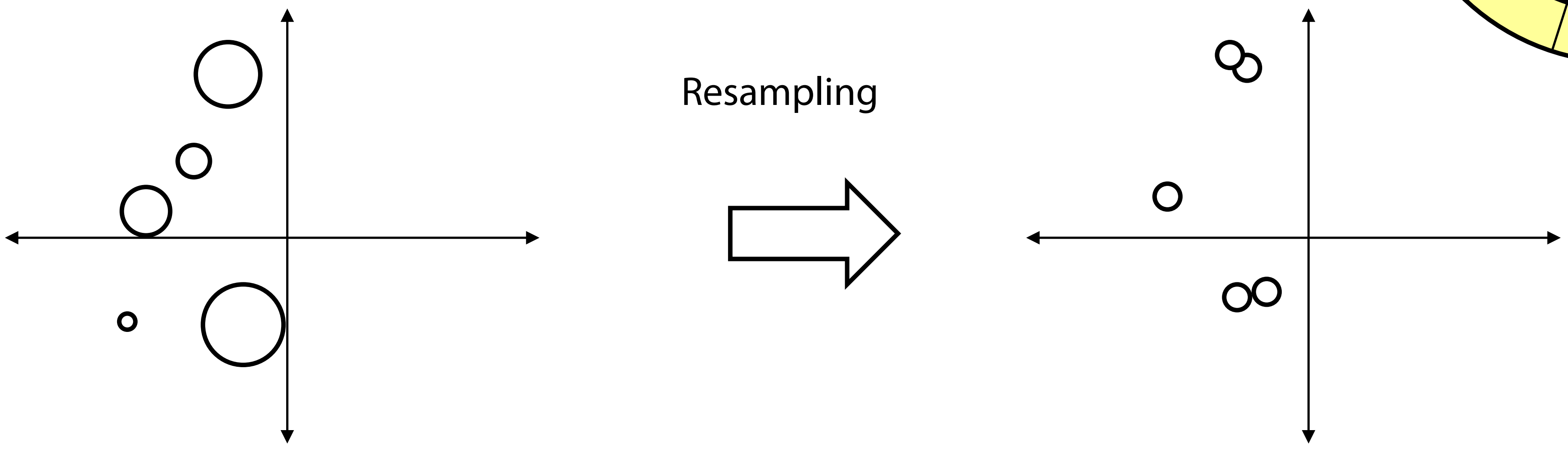
Resampling in a particle filter

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

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

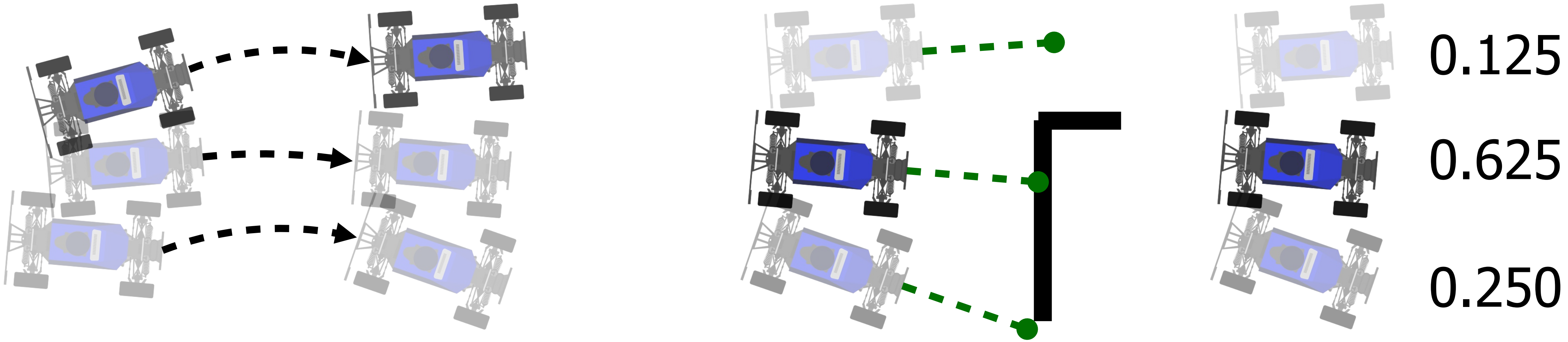
$$w_i = \frac{P(z_t|x_t^i)}{\sum_j P(z_t|x_t^j)}$$





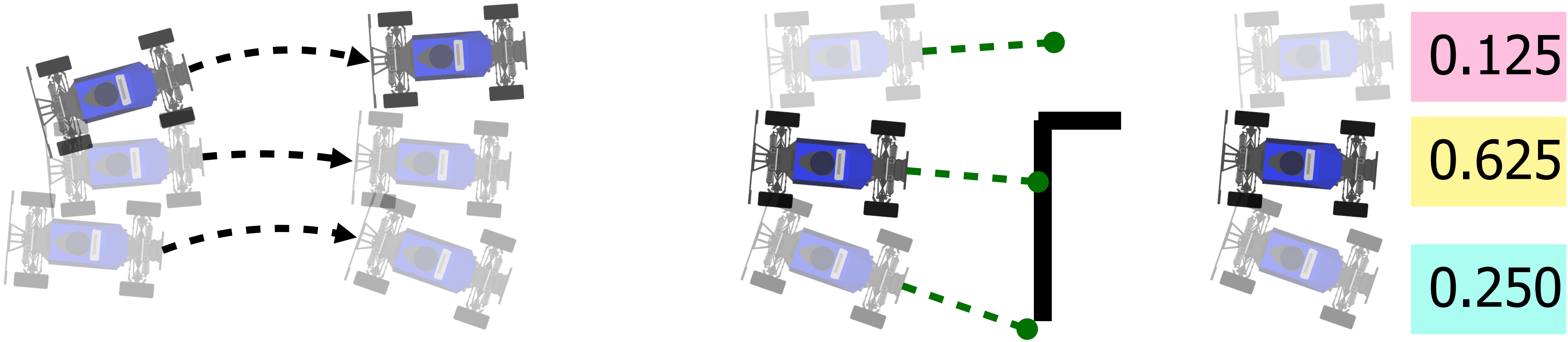
Resample particles from weighted distribution to give unweighted set of particles

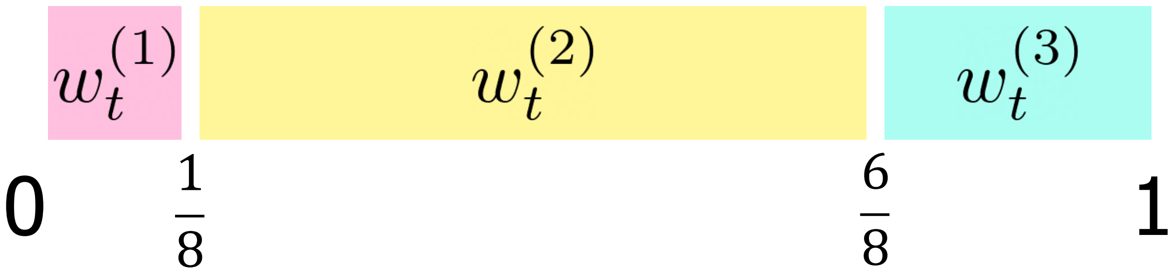
Original: Normalized Importance Sampling



$$Bel(x_t) = \left\{ \begin{array}{cccc} \bar{x}_t^{(1)} & \bar{x}_t^{(2)} & \cdots & \bar{x}_t^{(M)} \\ w_t^{(1)} & w_t^{(2)} & \cdots & w_t^{(M)} \end{array} \right\}$$

New: Normalized Importance Sampling with Resampling

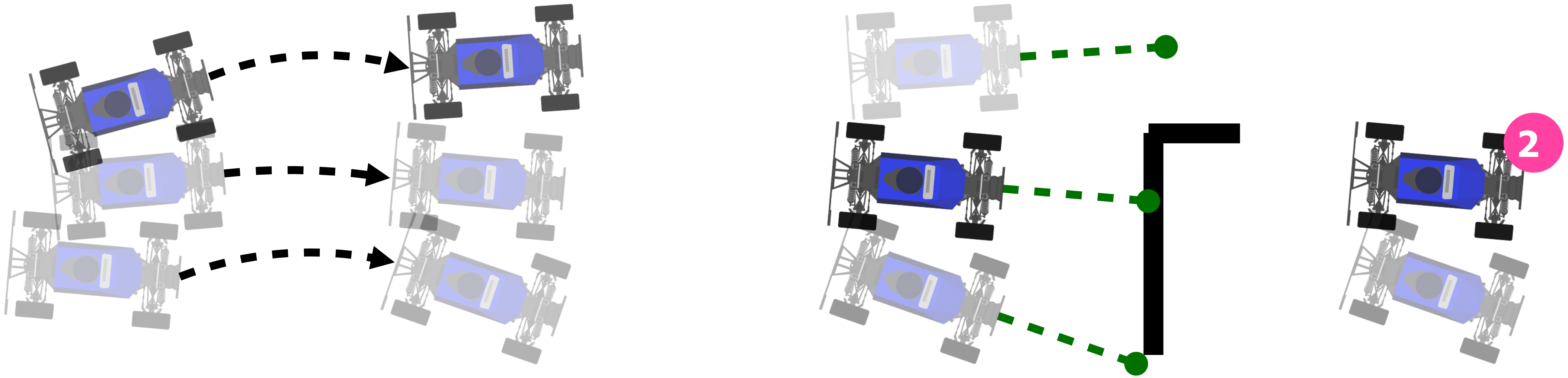




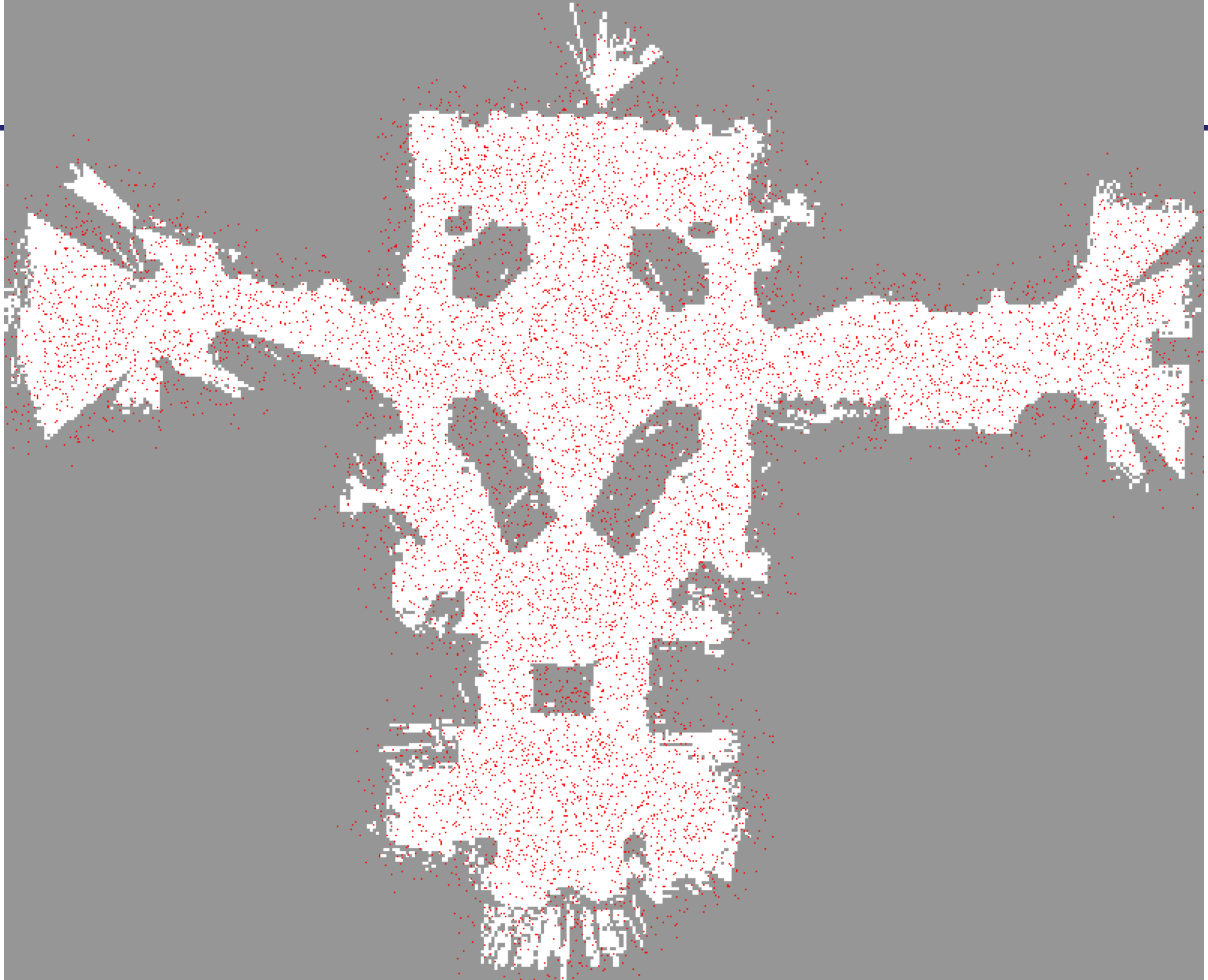
Here are your random numbers:

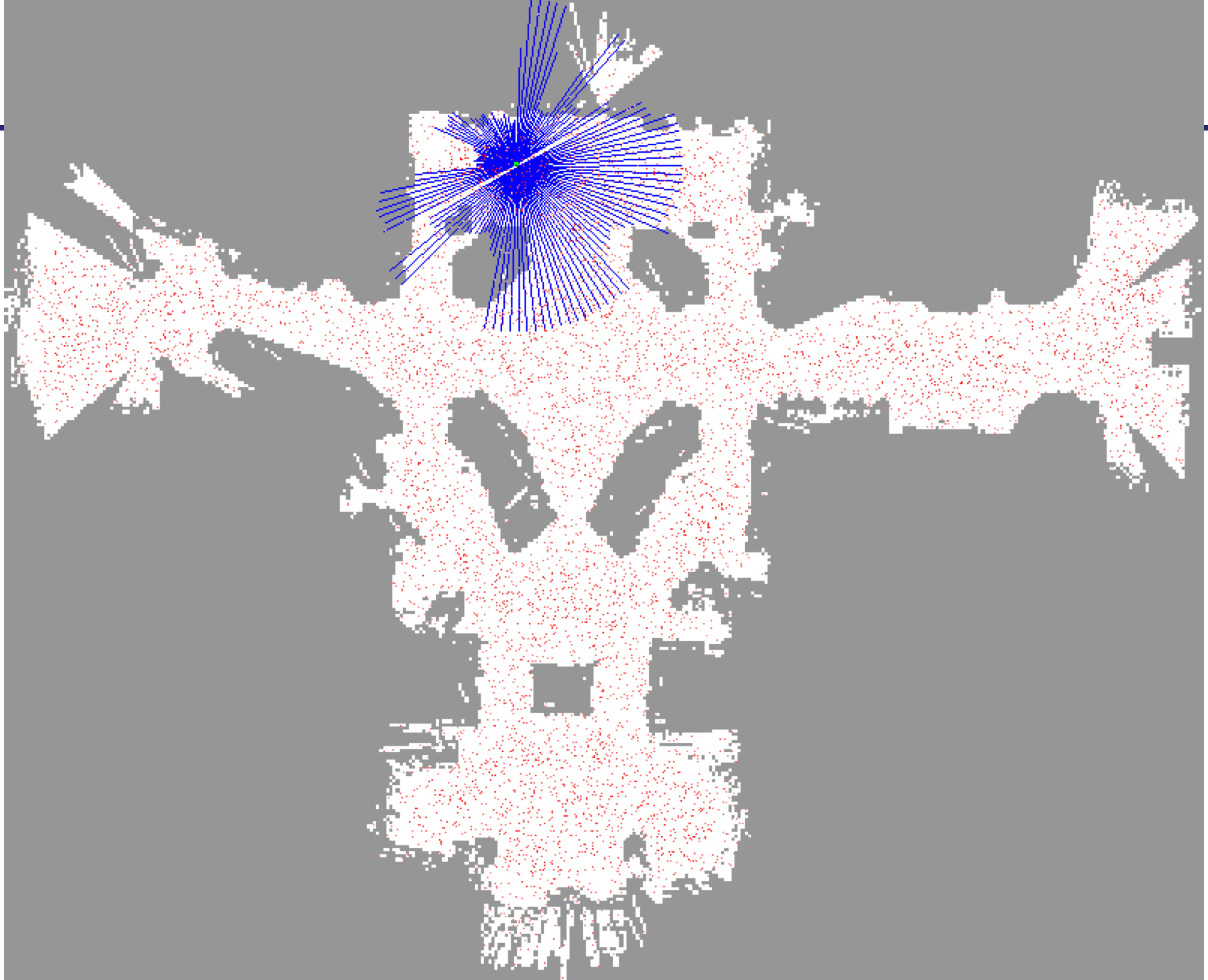
0.97
0.26
0.72

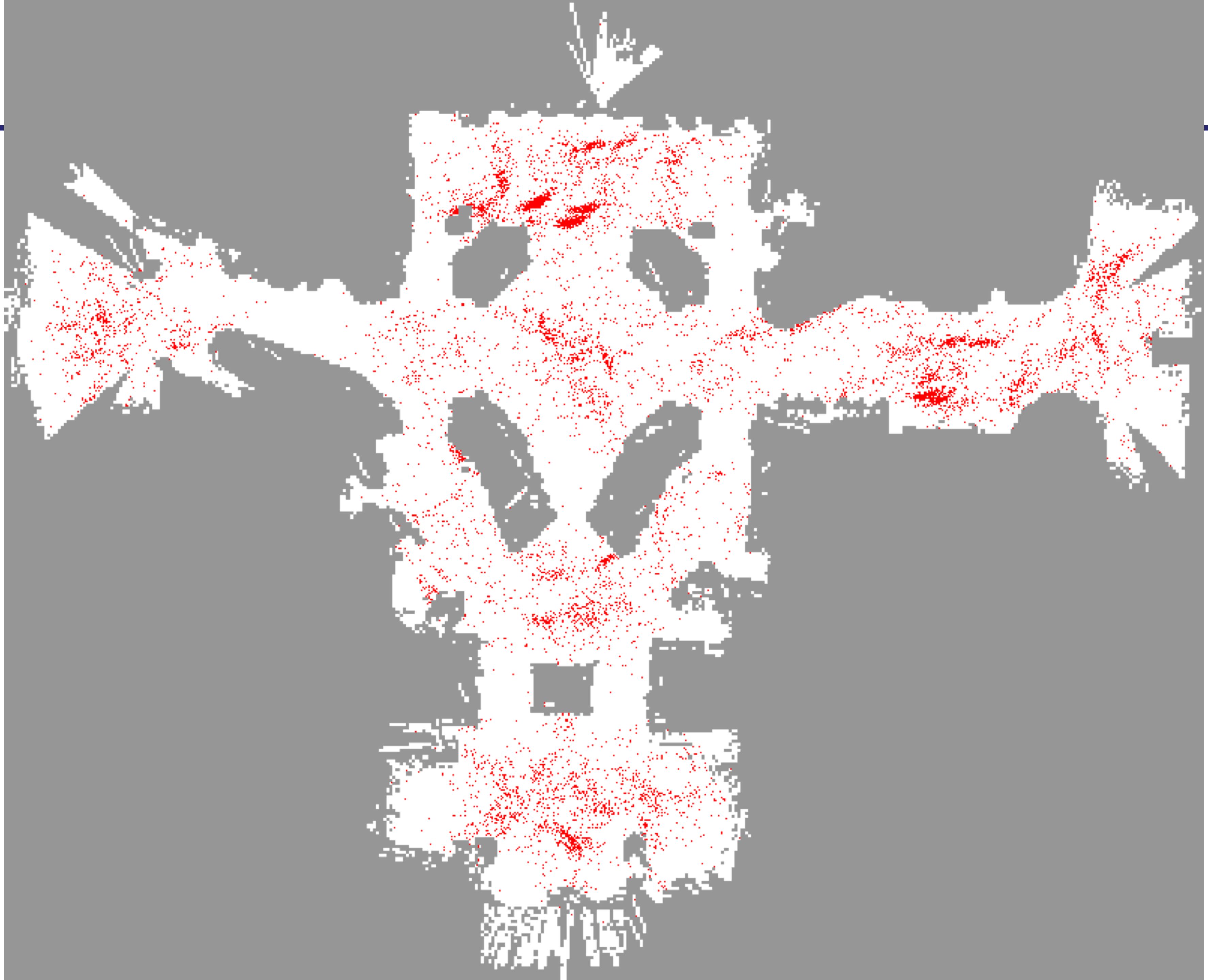
New: Normalized Importance Sampling with Resampling

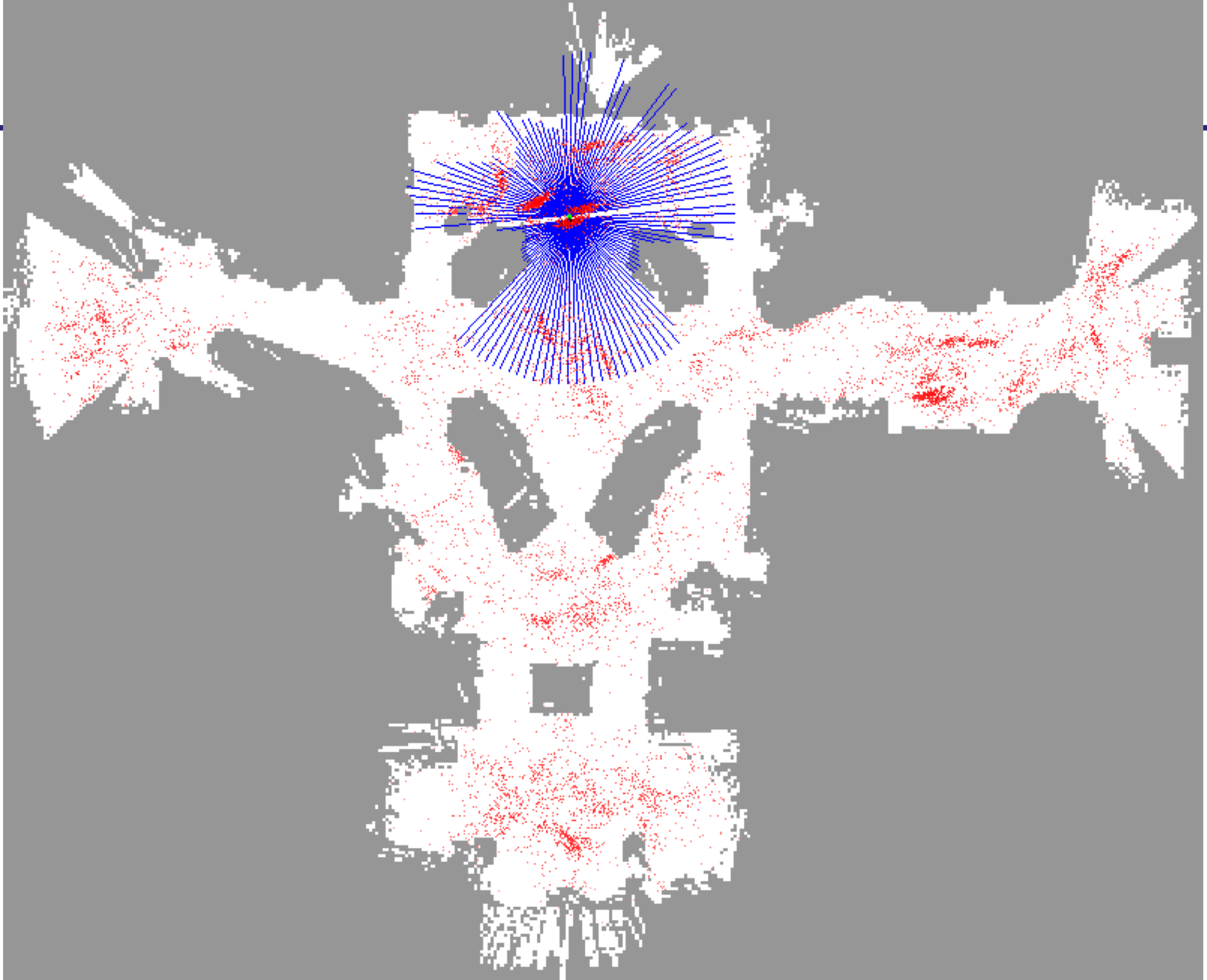


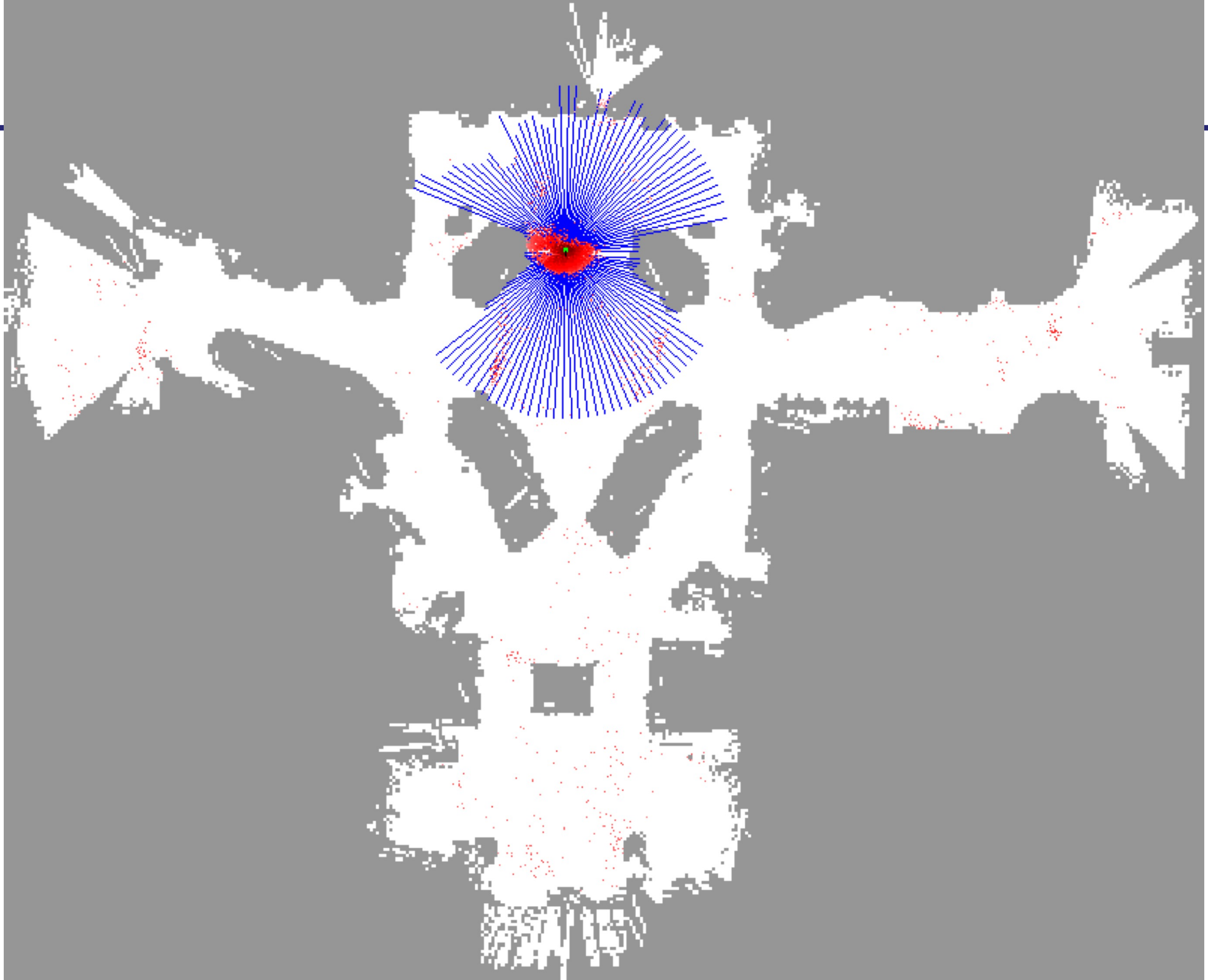
$$x_t^{(i)} \sim w_t^{(i)}, \; Bel(x_t) = \left\{ \begin{array}{ccc} x_t^{(1)} & \cdots & x_t^{(M)} \\ \frac{1}{M} & \cdots & \frac{1}{M} \end{array} \right\}$$

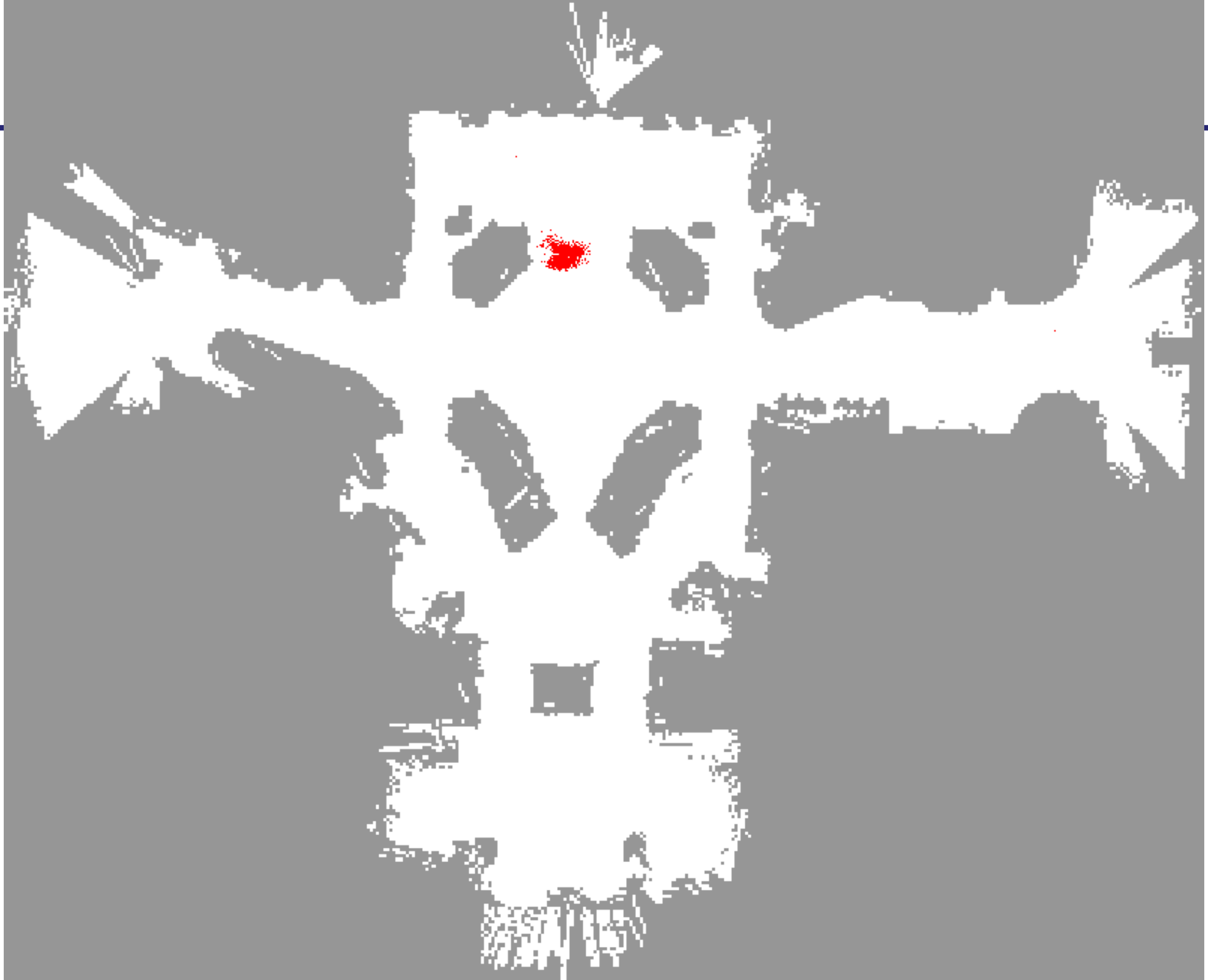




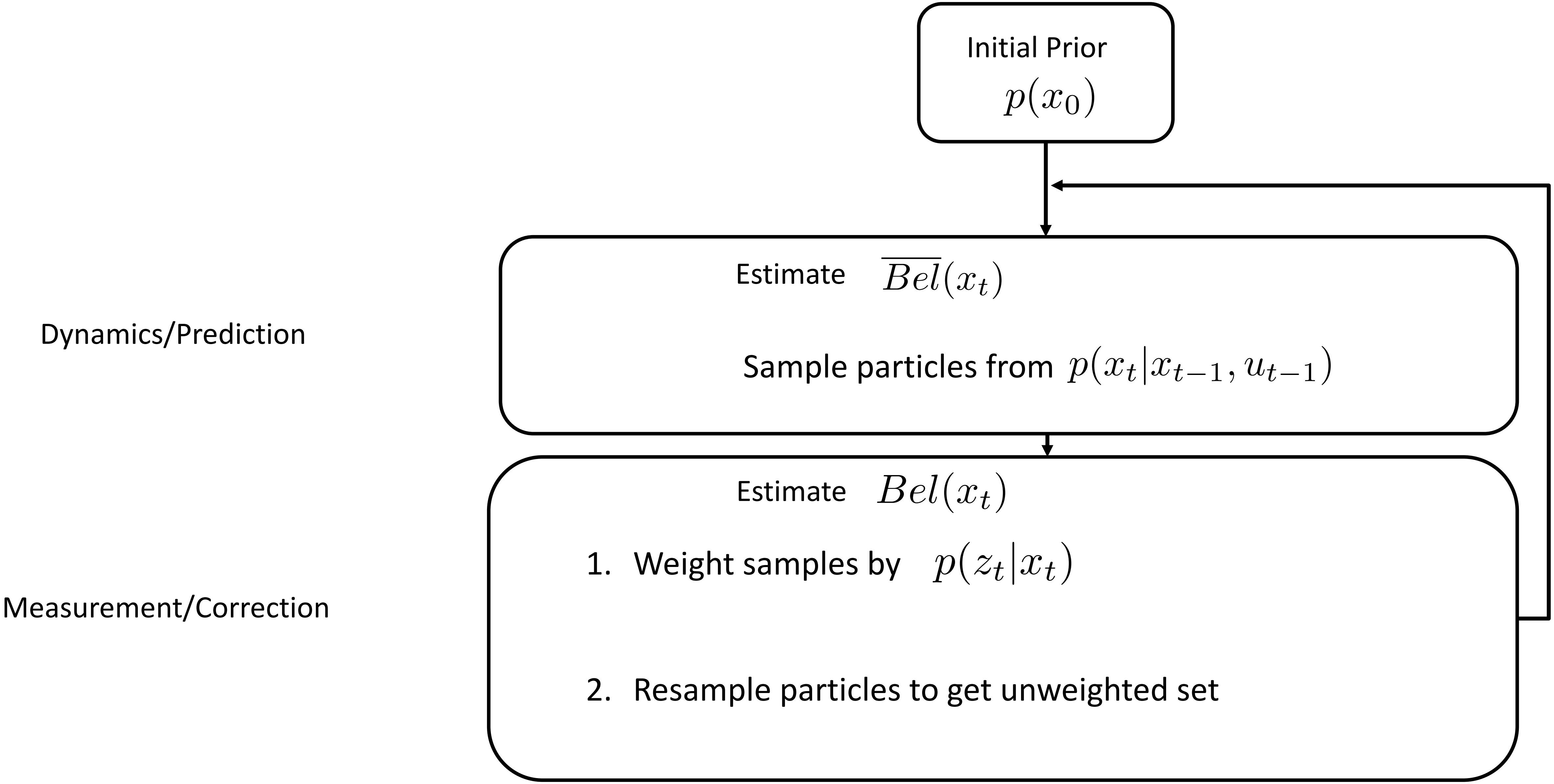








Overall Particle Filter algorithm – v2



Lecture Outline

Particle Based Representations in Filtering

↓

Particle Filter

↓

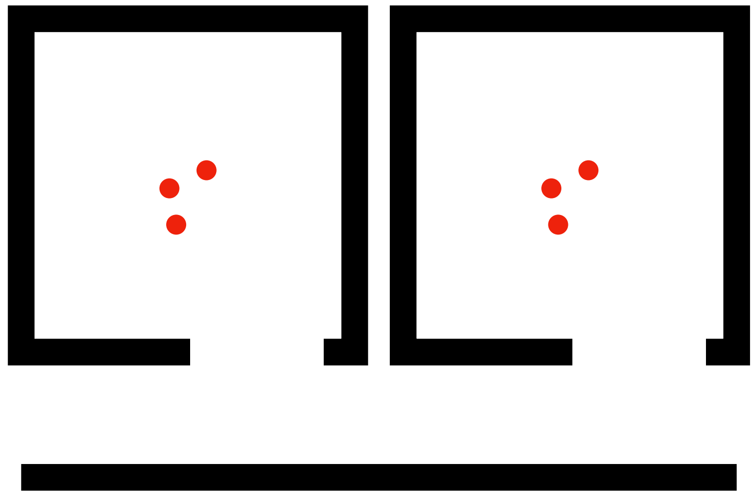
Particle Filter w/ Resampling

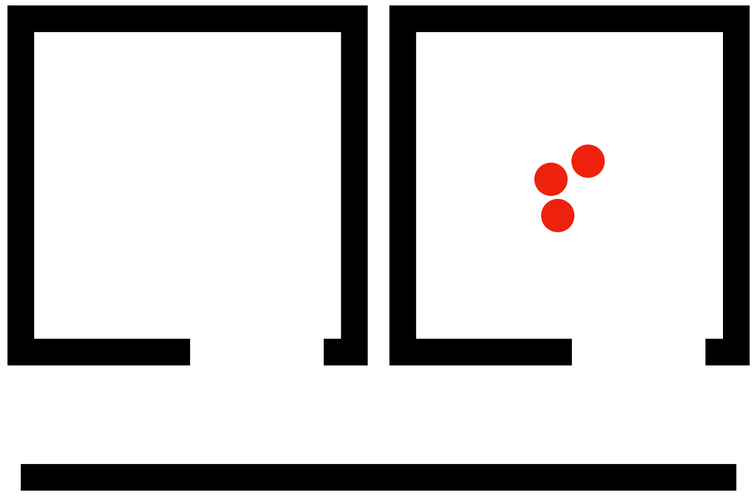
↓

Practical Considerations

Problem 1: Two Room Challenge

Particles begin equally distributed, no motion or observation

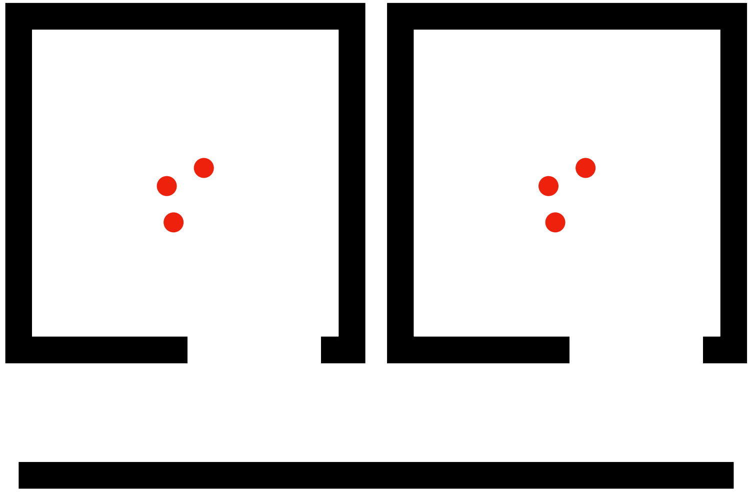


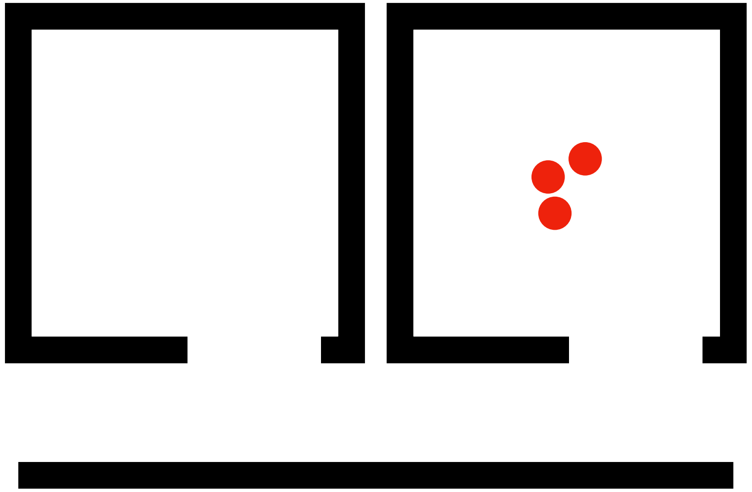


All particles migrate to one room!

Reason: Resampling Increases Variance

50% prob. of resampling particle from Room 1 vs Room 2
31% prob. of preserving 50-50 particle split





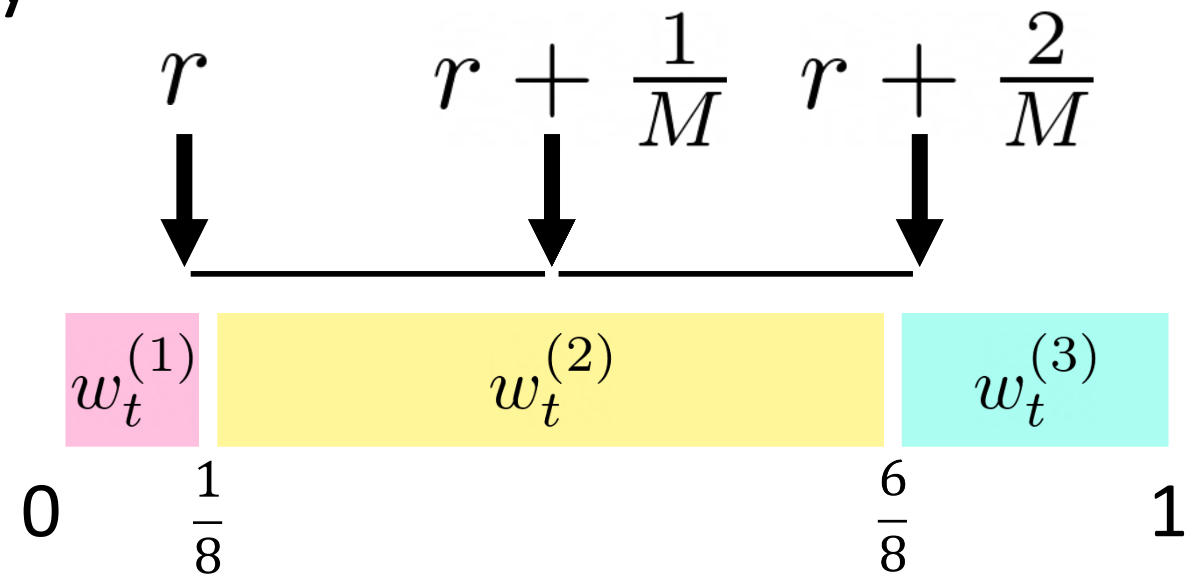
All particles migrate to one room!

Idea 1: Judicious Resampling

- Key idea: resample less often! (e.g., if the robot is stopped, don't resample). Too often may lose particle diversity, infrequently may waste particles
- Common approach: don't resample if weights have low variance
- Can be implemented in several ways: don't resample when…
  - …all weights are equal
  - …weights have high entropy
  - …ratio of max to min weights is low

Idea 2: Low-Variance Resampling

- Sample one random number $r \sim [0, \frac{1}{M}]$
- Covers space of samples more systematically (and more efficiently)
- If all samples have same importance weight, won't lose particle diversity



Other Practical Concerns

- How many particles is enough?
  - Typically need more particles at the beginning (to cover possible states)
    - KLD Sampling (Fox, 2001) adaptively increases number of particles when state uncertainty is high, reduces when state uncertainty is low
- Particle filtering with overconfident sensor models
  - Squash sensor model prob. with power of 1/m (Lecture 3)
  - Sample from better proposal distribution than motion model
    - Manifold Particle Filter (Koval et al., 2017) for contact sensors
- Particle starvation: no particles near current state

MuSHR Localization Project

- Implement kinematic car motion model
- Implement different factors of single-beam sensor model
- Combine motion and sensor model with the Particle Filter algorithm

Class Outline

State Estimation

- Robotic System Design
- Filtering
- Localization
- SLAM

Control

- Feedback Control
- PID Control
- MPC
- LQR

Planning

- Search
- Heuristic Search
- Motion Planning
- Lazy Search

Learning

- Imitation Learning
- Policy Gradient
- Actor-Critic
- Model-Based RL