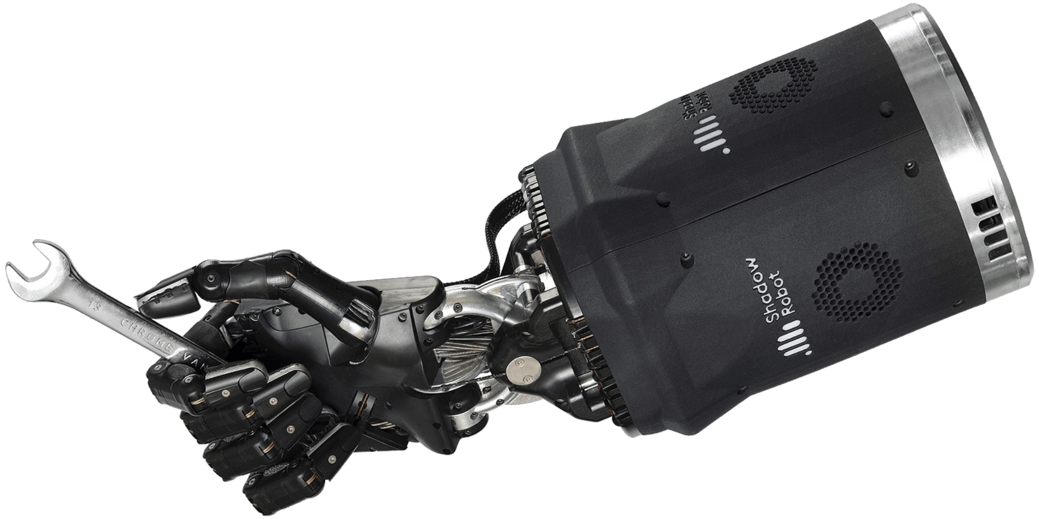
Autonomous Robotics

Winter 2026

Abhishek Gupta, Siddhartha Srinivasa

TAs: Carolina Higuera, Entong Su, Rishabh Jain



Class Outline

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

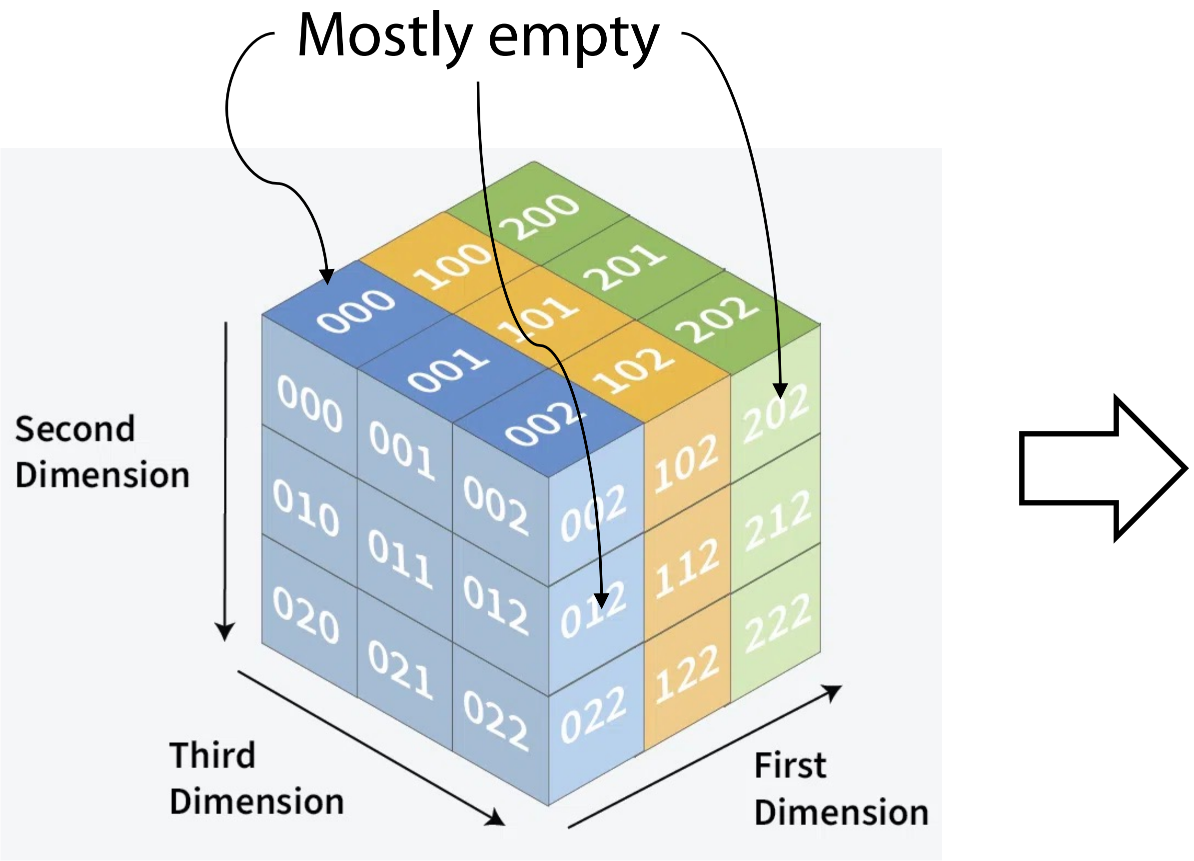
Control
- Feedback Control
- PID Control
- MPC
- LQR

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

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

Recap

Going from Bins to Particles



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

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?

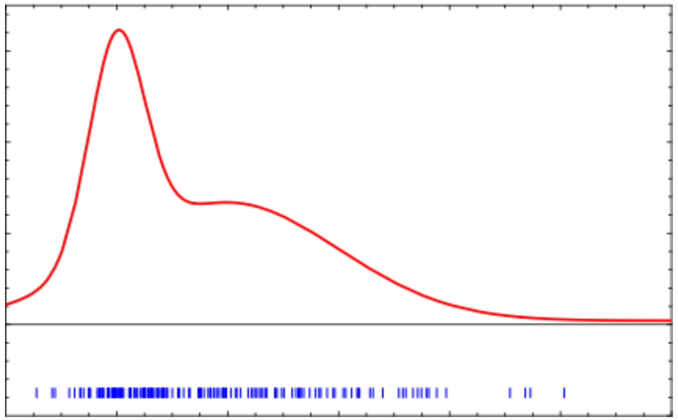
Belief Distribution as Particles

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

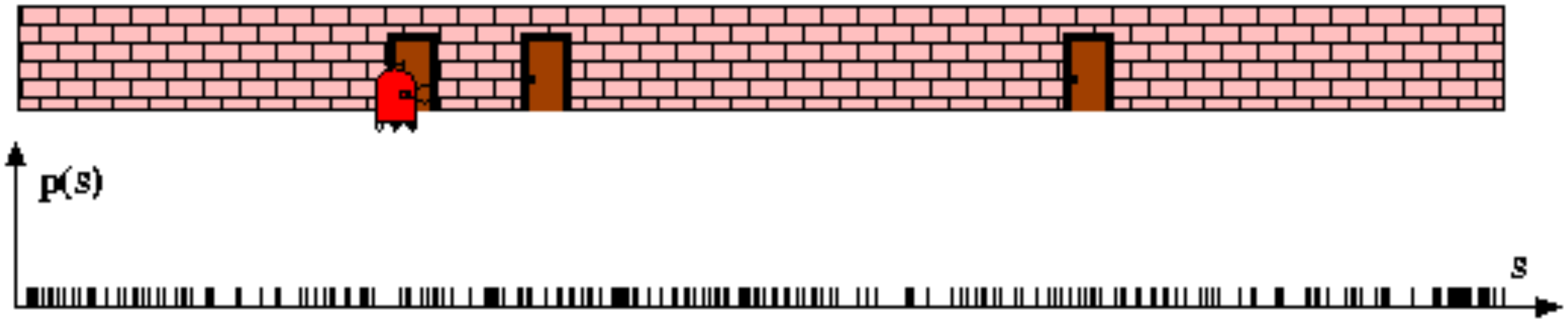
Dynamics Update
$$\overline{Bel}(x_t) = \int p(x_t|u_t, x_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

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?

Propagating Belief Through Dynamics: Initial

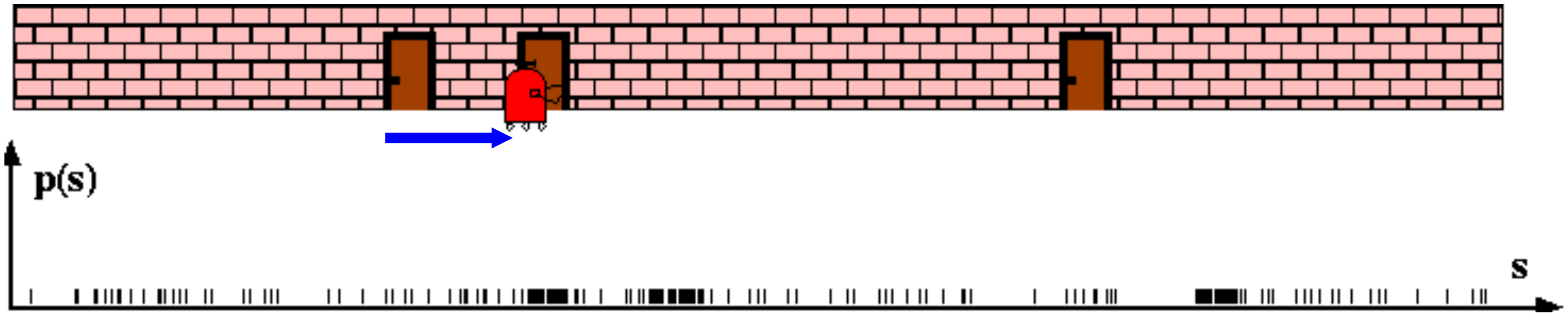


Propagating Belief Through Dynamics: Robot Motion

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

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

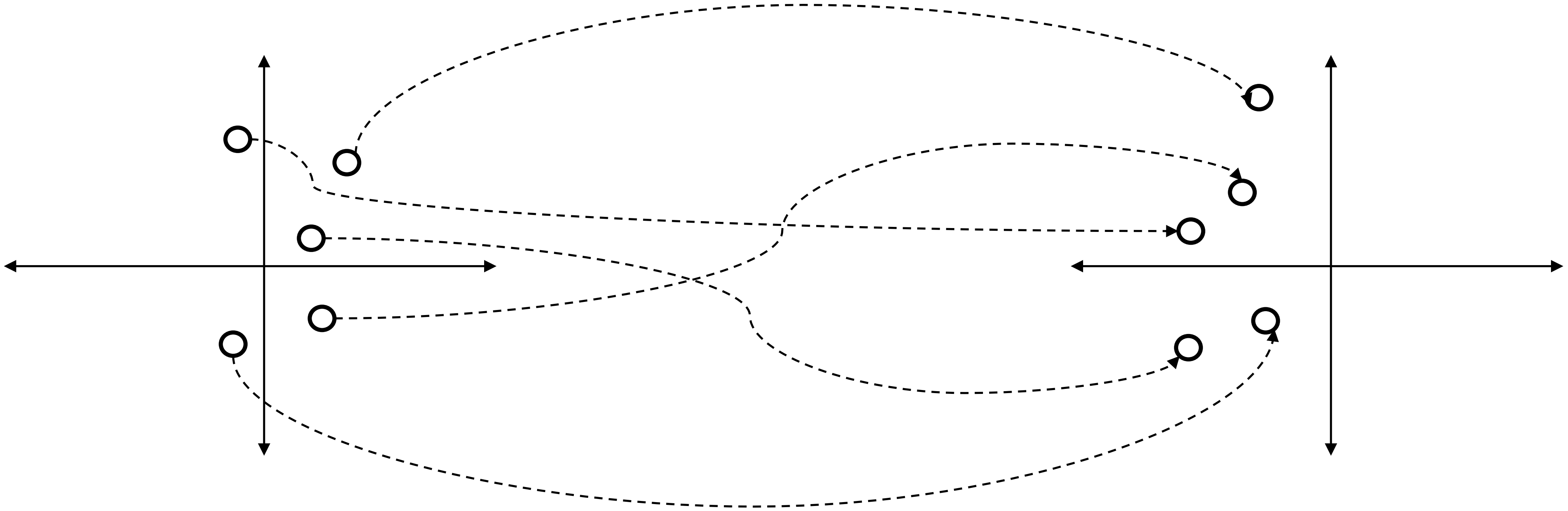


Dynamics Update:

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

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

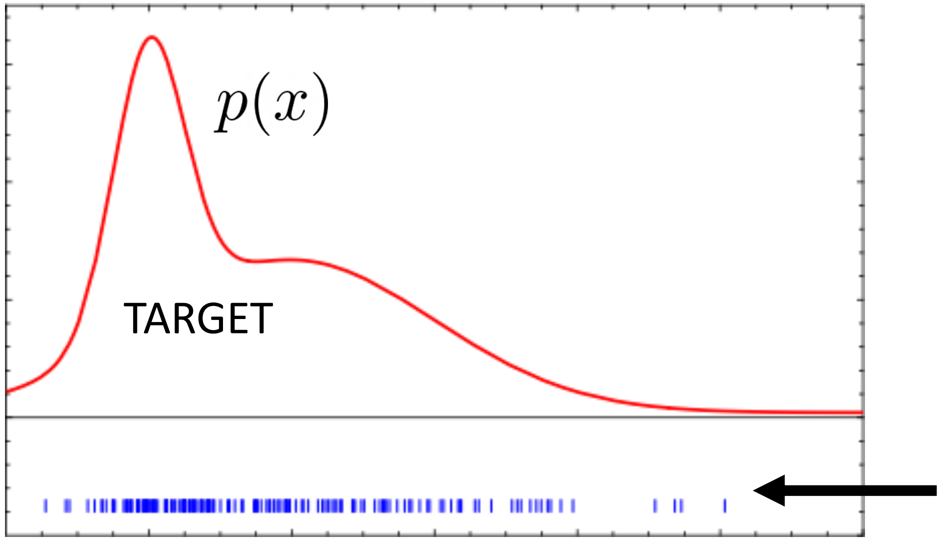
Detour: What is Importance Sampling?

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

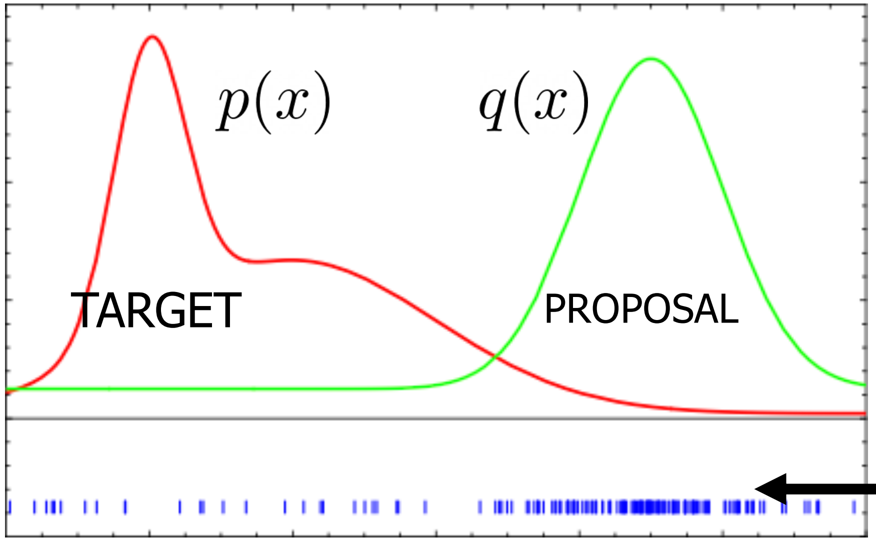
Importance Sampling



Don't know how to sample from target!

Importance Sampling

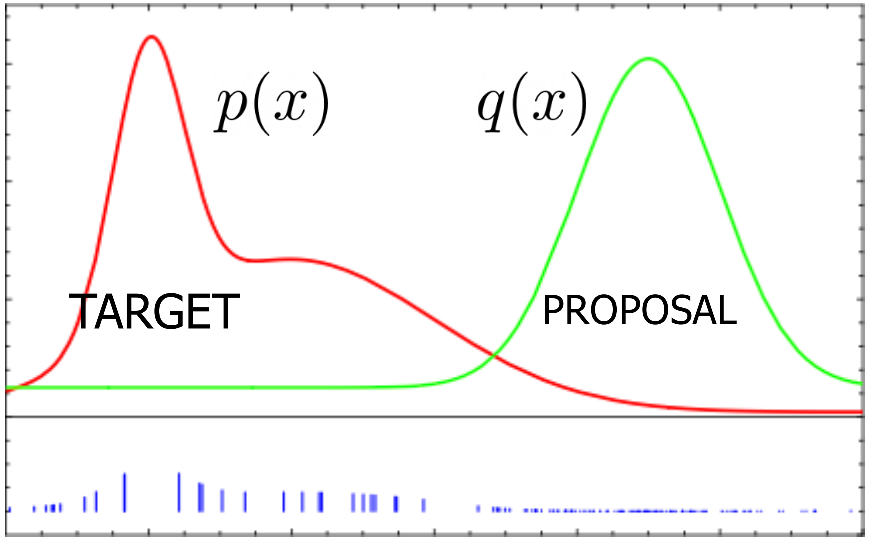
1. Sample from an (easy) proposal distribution



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)

IMPORTANCE WEIGHT

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

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

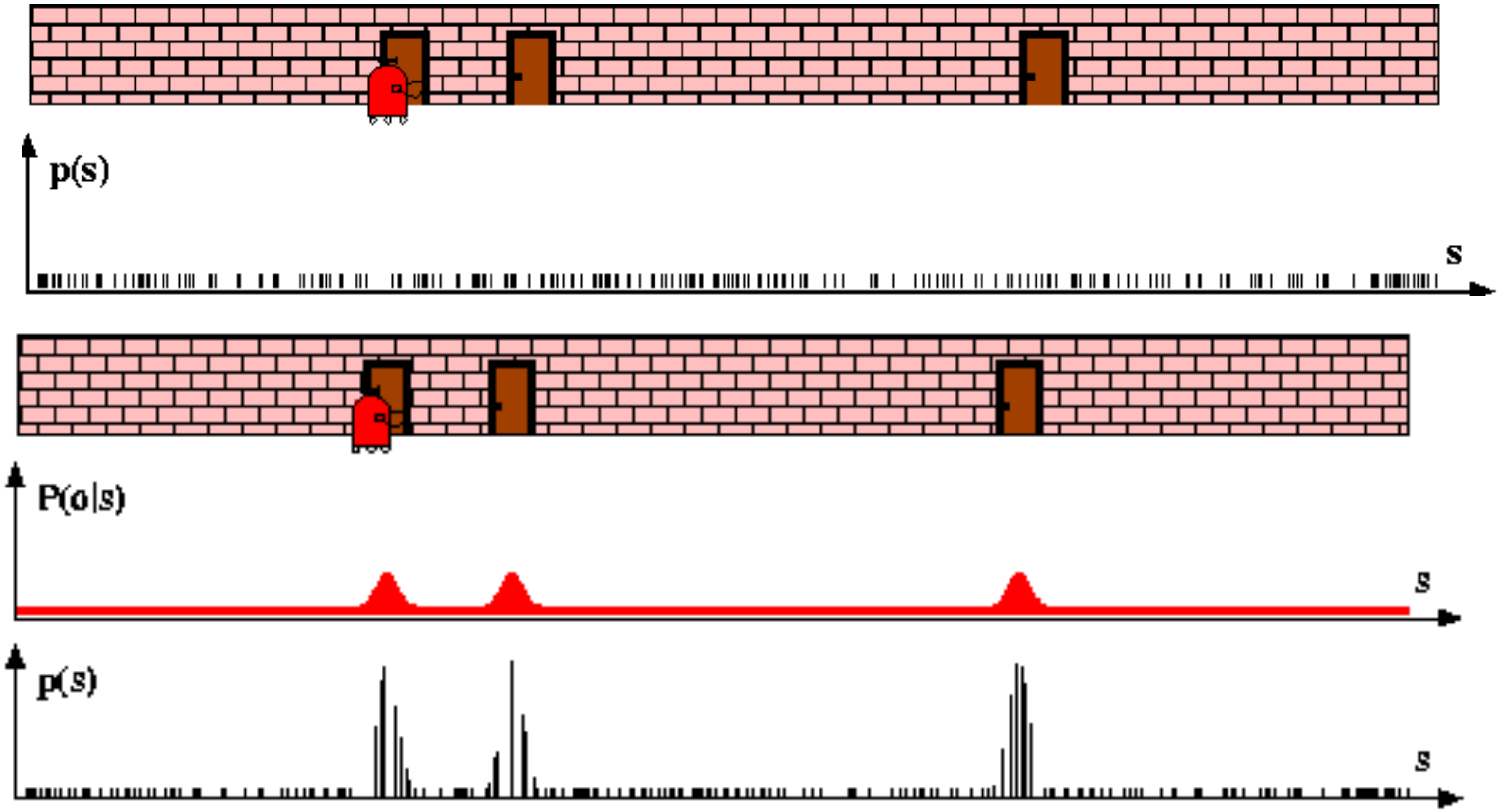
Importance Weight (Ratio)

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

Sensor Information: Importance Sampling

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

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

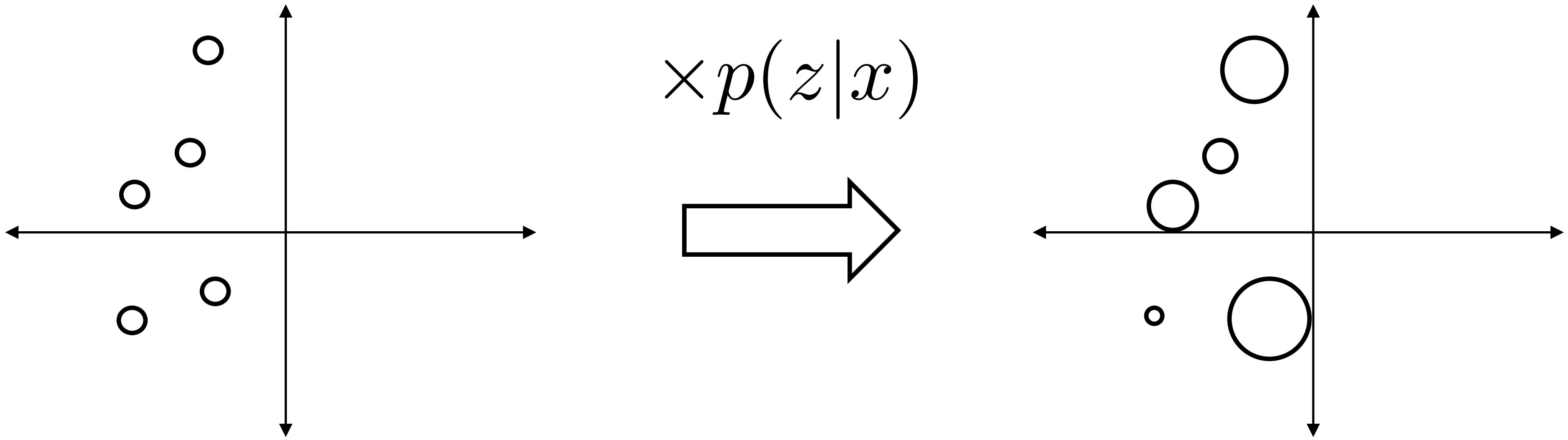


Measurement Update

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

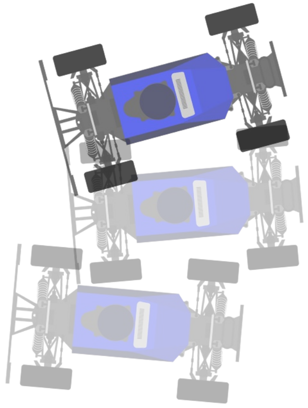
$$Bel(x_t) = \frac{P(z_t|x_t)\overline{Bel}(x_t)}{\int P(z_t|x_t)\overline{Bel}(x_t)dx_t}$$

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



Reweight particles according to measurement likelihood

Normalized Importance Sampling



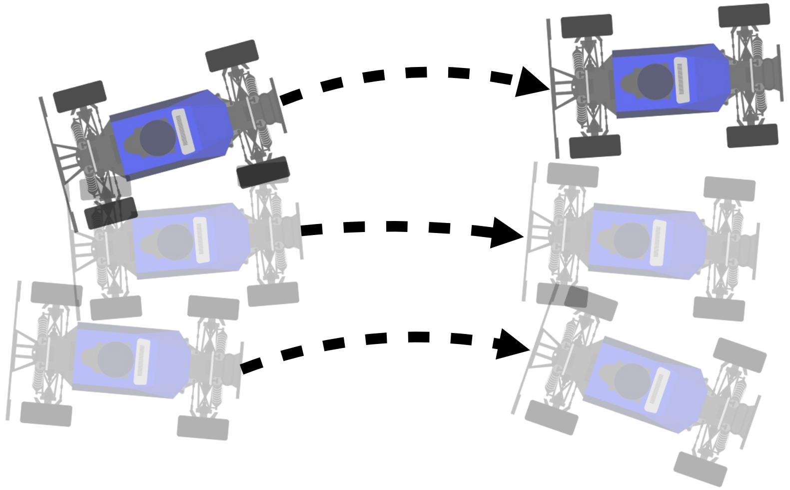
0.50

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



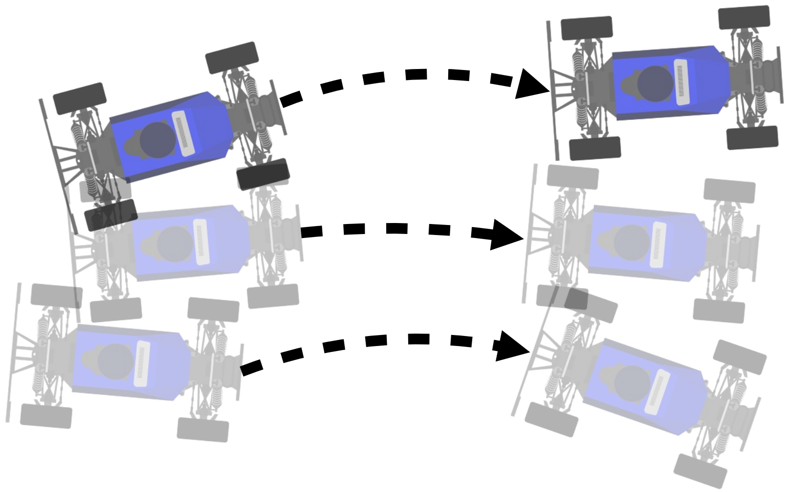
0.50

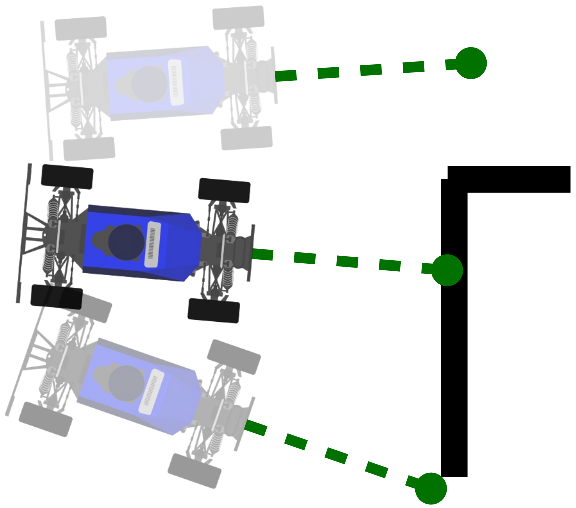
0.25

0.25

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

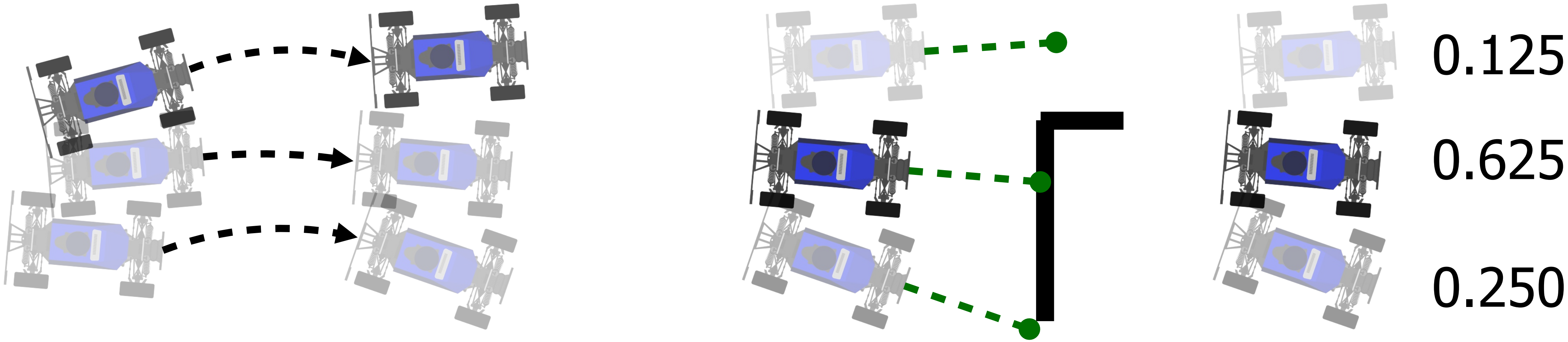
Normalized Importance Sampling





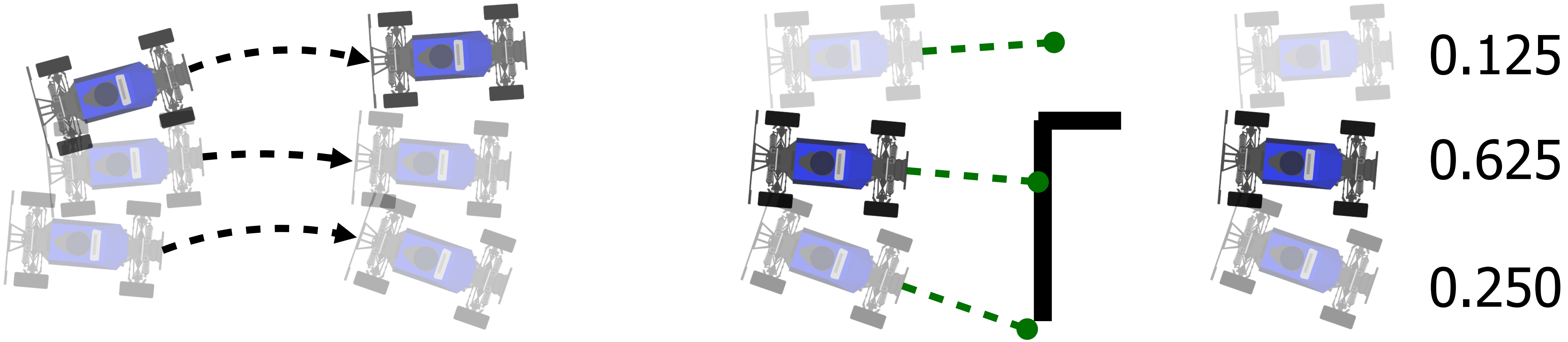
$$w_t^{(i)} = P(z_t | \bar{x}_t^{(i)}) w_{t-1}^{(i)}$$

Normalized Importance Sampling



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

Normalized Importance Sampling

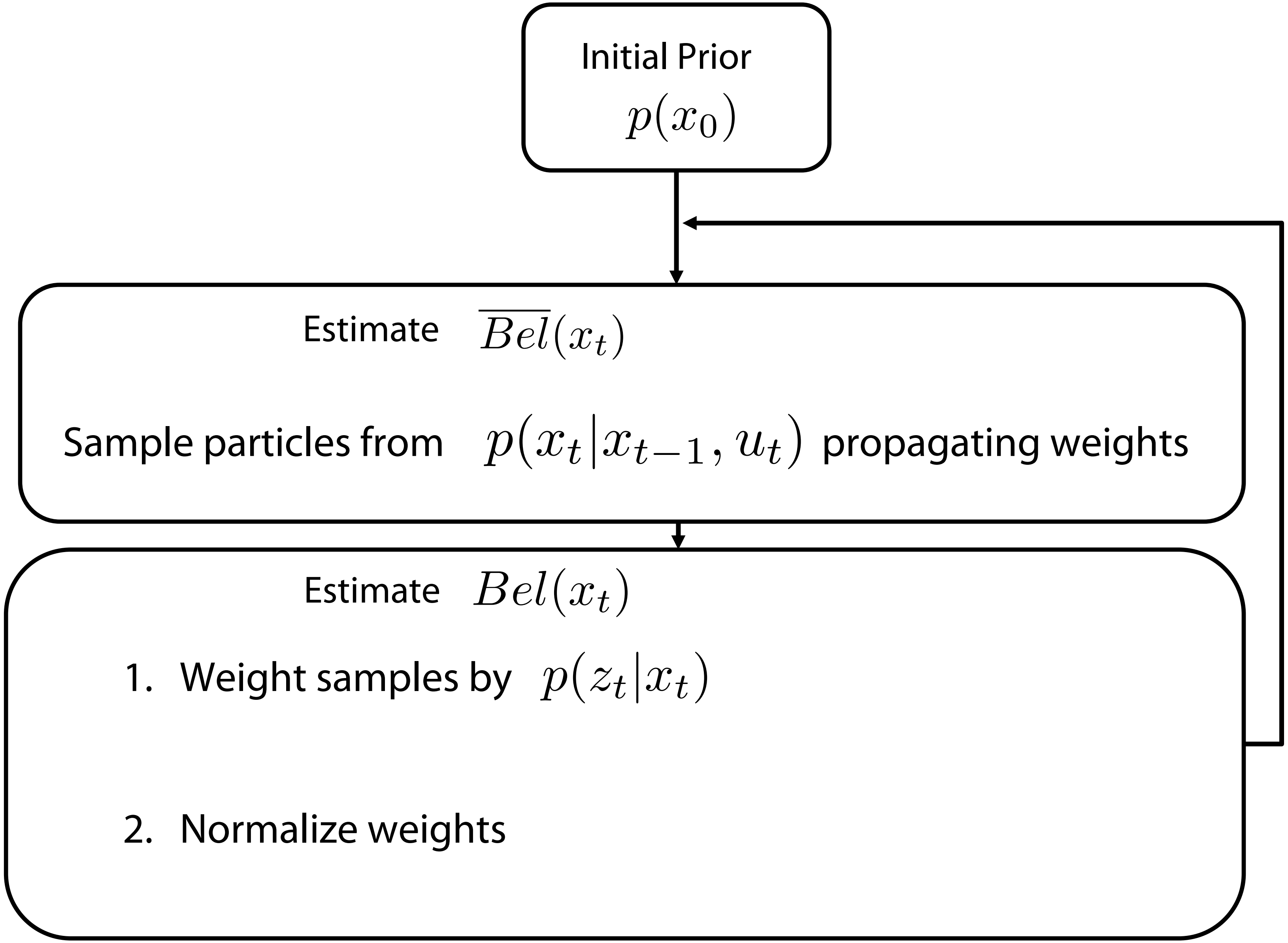


$$Bel(x_t) = \left\{ \begin{array}{cccc} \bar{x}_t^{(1)} & \bar{x}_t^{(2)} & \cdots & \bar{x}_t^{(M)} \\ w_t^{(1)} & w_t^{(2)} & \cdots & w_t^{(M)} \end{array} \right\}$$

Overall Particle Filter algorithm – v1

Dynamics/Prediction

Measurement/Correction



Lecture Outline

Particle Based Representations in Filtering

↓

Particle Filter

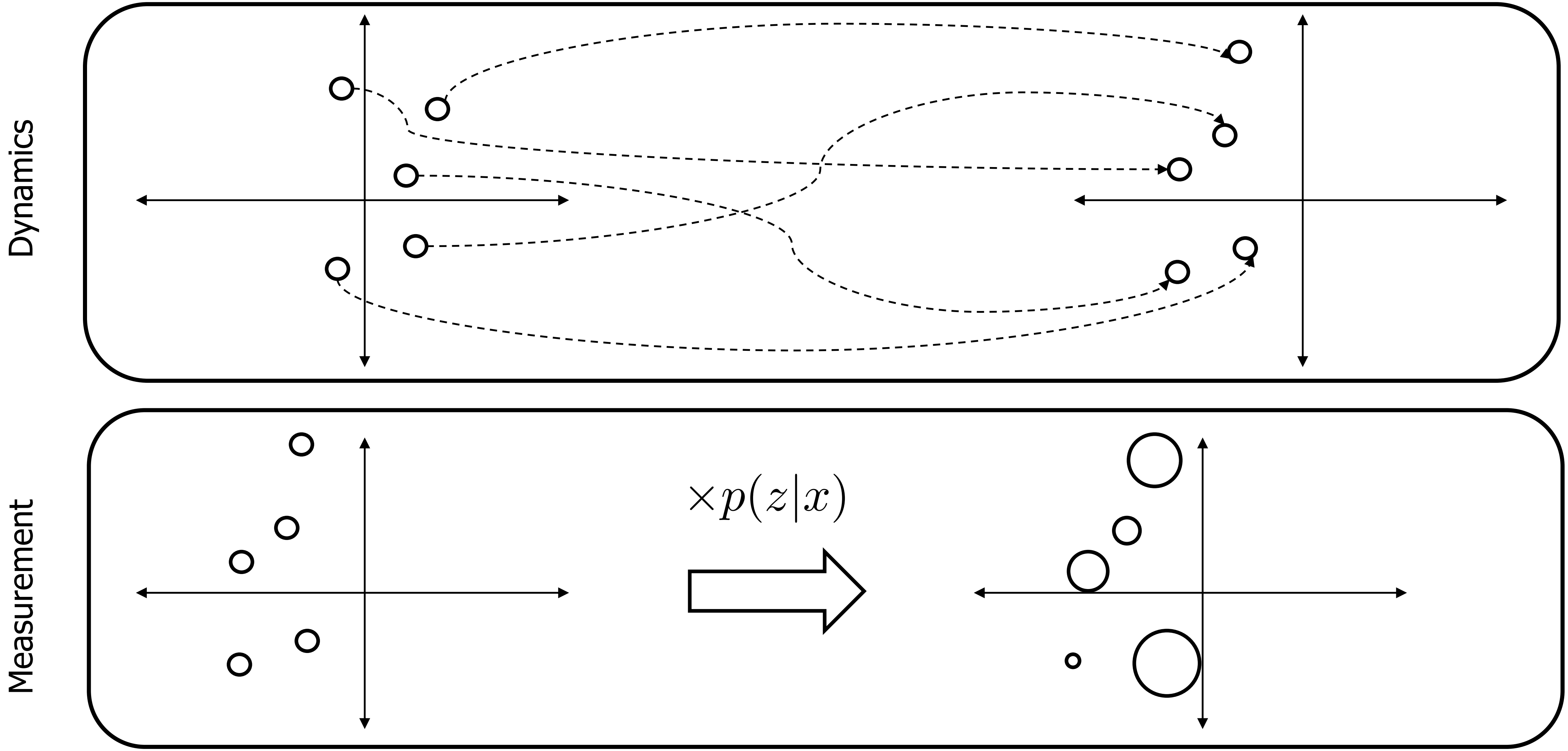
↓

Particle Filter w/ Resampling

↓

Practical Considerations

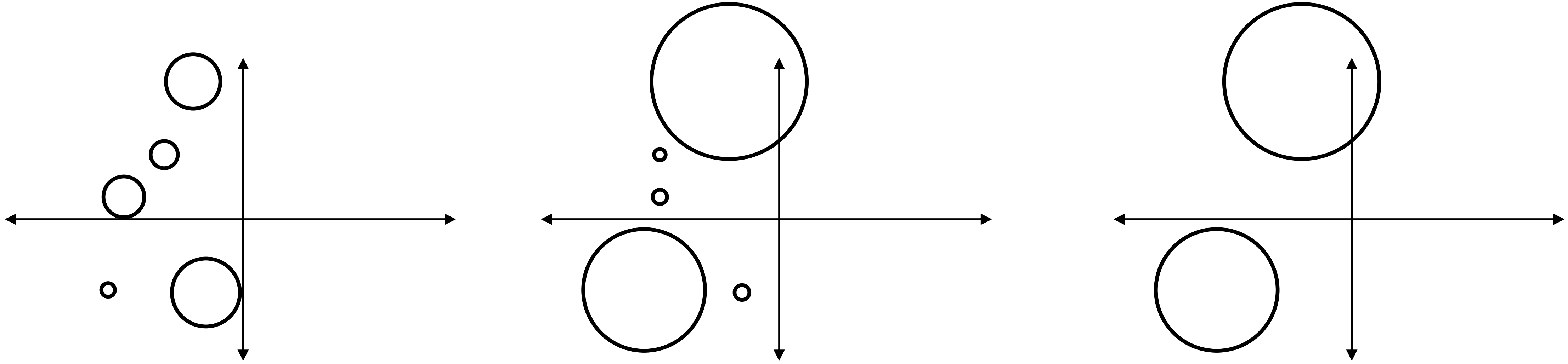
What happens across multiple steps?



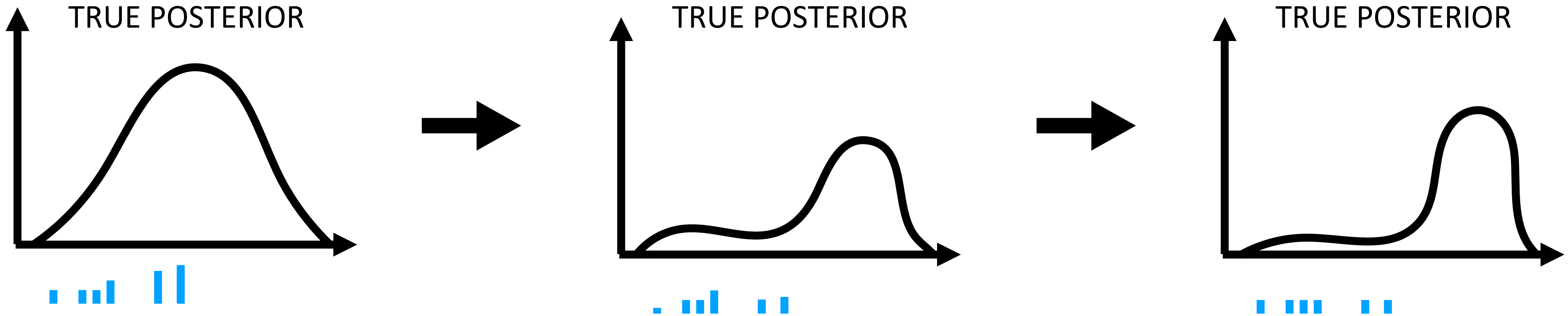
Importance weights get multiplied at each step

Why might this be bad?

Importance weights get multiplied at each step



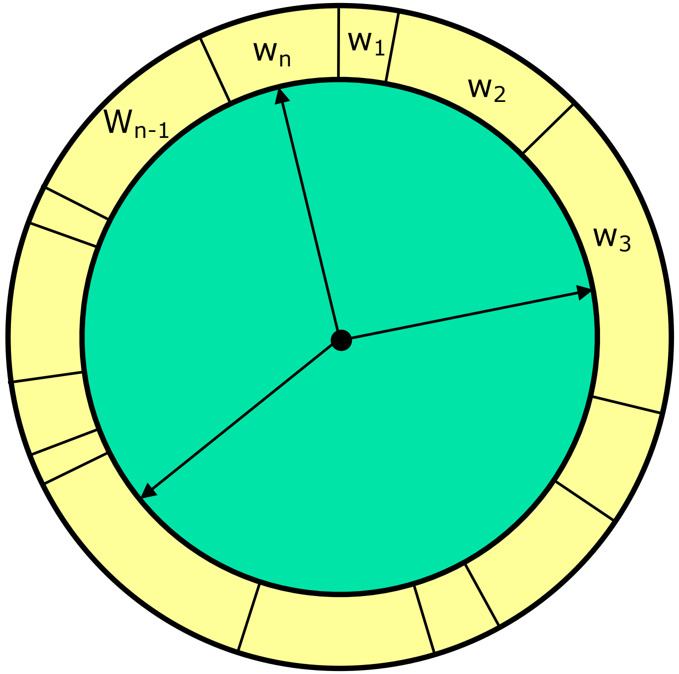
1. May blow up and get numerically unstable over many steps
2. Particles stay stuck in unlikely regions



Resampling

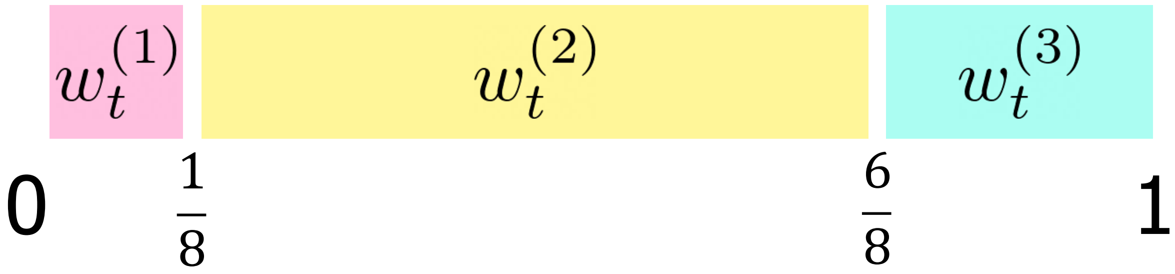
- **Given**: Set ***S*** of weighted samples (from measurement step) with weights w_i
- **Wanted** : unweighted random sample, where the probability of drawing $\boldsymbol{x_i}$ is given by $\boldsymbol{w_i}$.
- Typically done ***n*** times with replacement to generate new sample set ***S'***.

Resampling



Here are your random numbers:

0.97
0.26
0.72



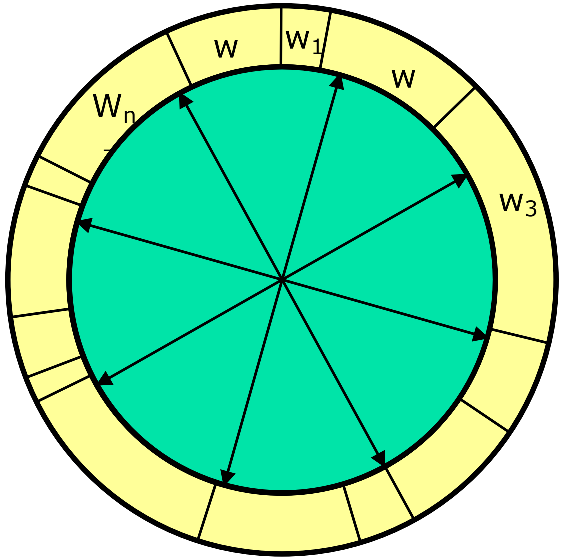
- Spin a roulette wheel
- Space according to weights
- Pick samples based on where it lands

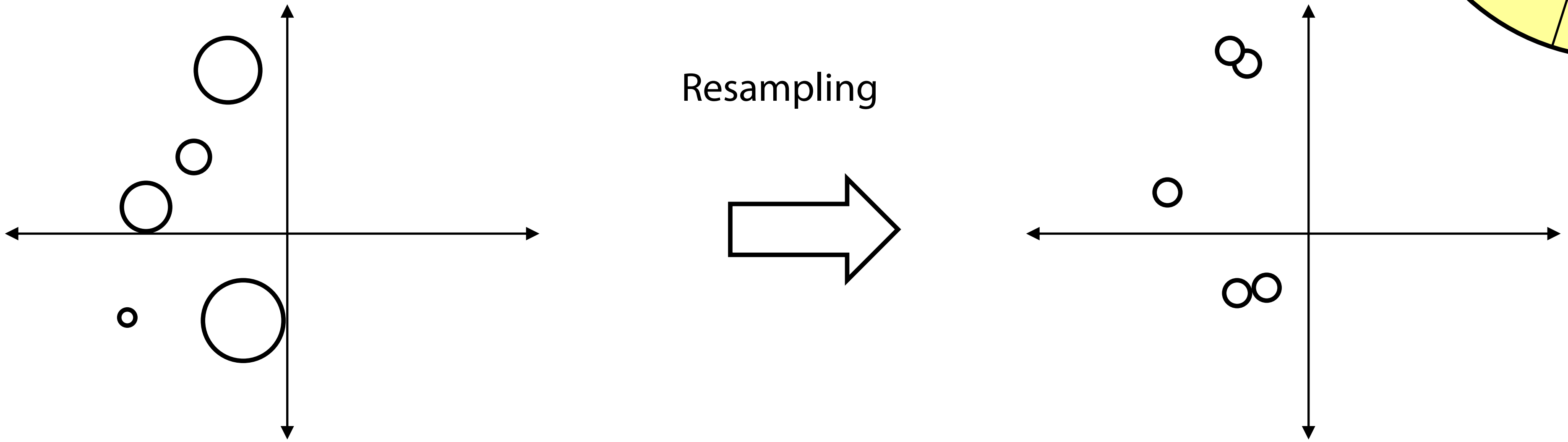
Resampling in a particle filter

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

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

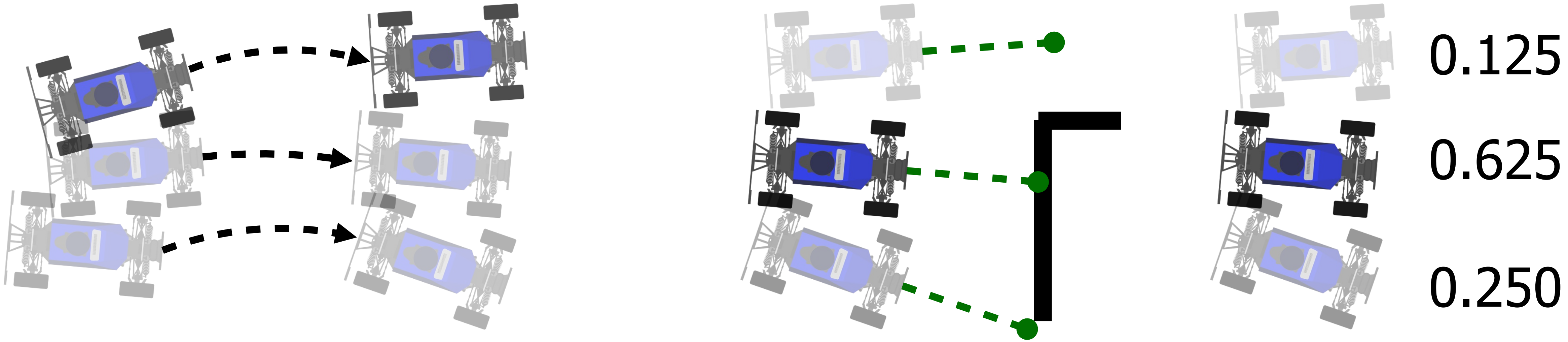
$$w_i = \frac{P(z_t|x_t^i)}{\sum_j P(z_t|x_t^j)}$$





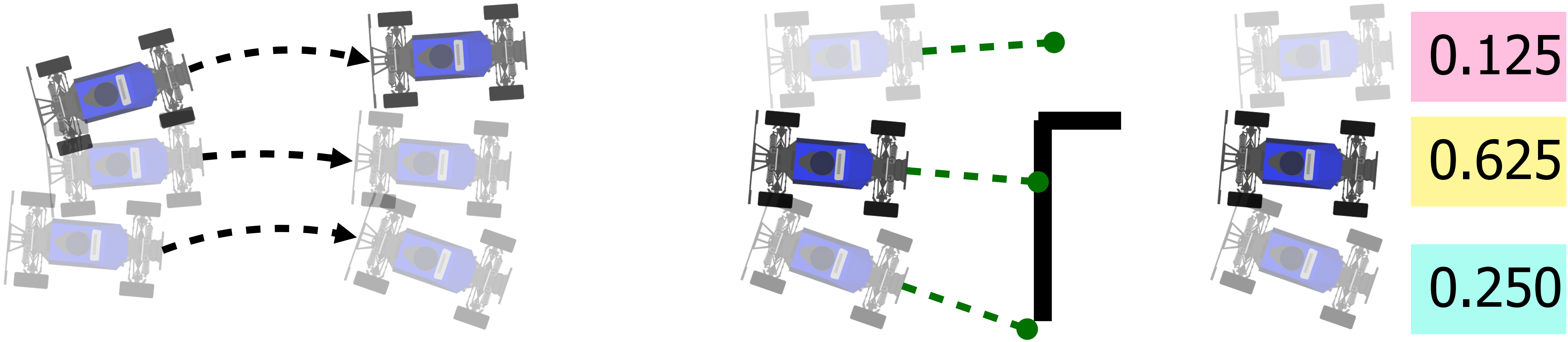
Resample particles from weighted distribution to give unweighted set of particles

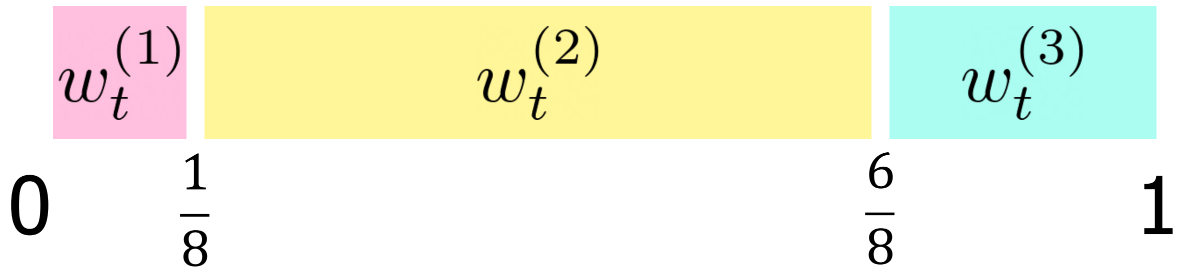
Original: Normalized Importance Sampling



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

New: Normalized Importance Sampling with Resampling

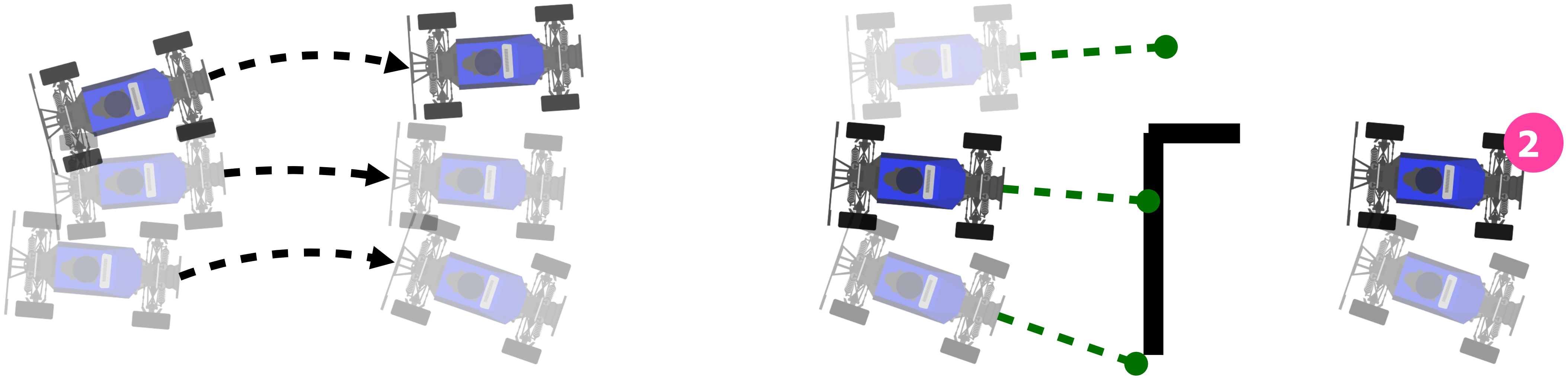




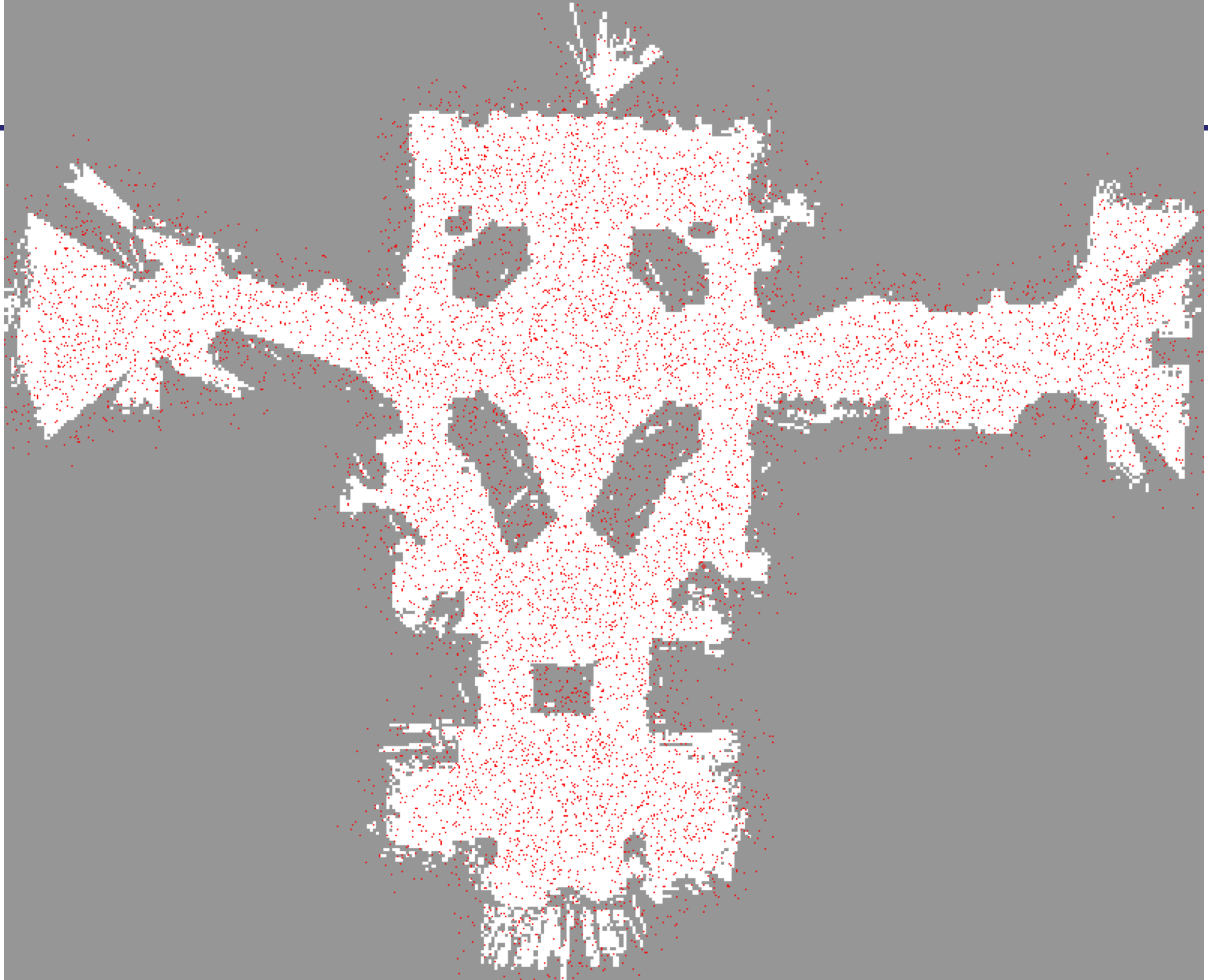
Here are your random numbers:

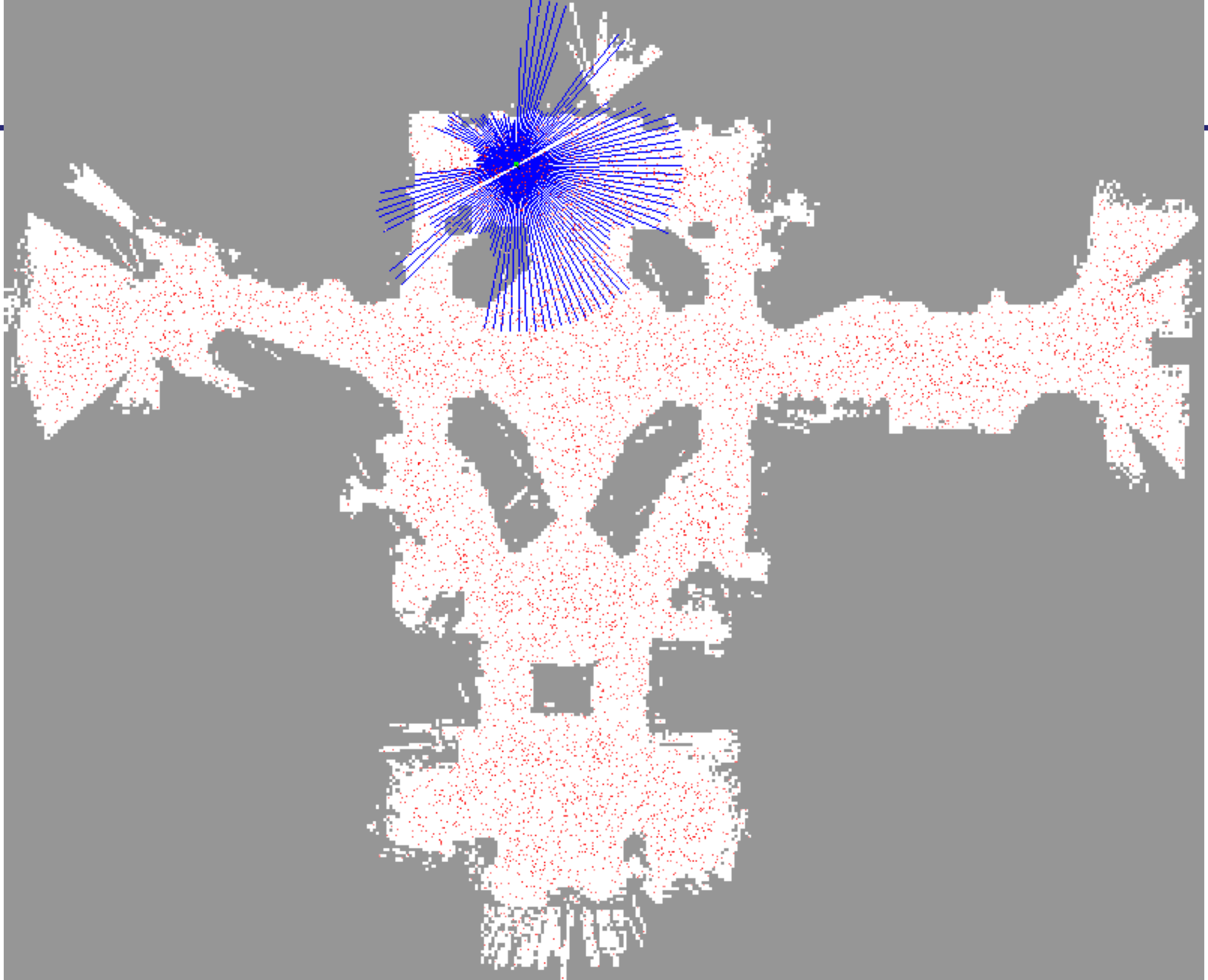
0.97
0.26
0.72

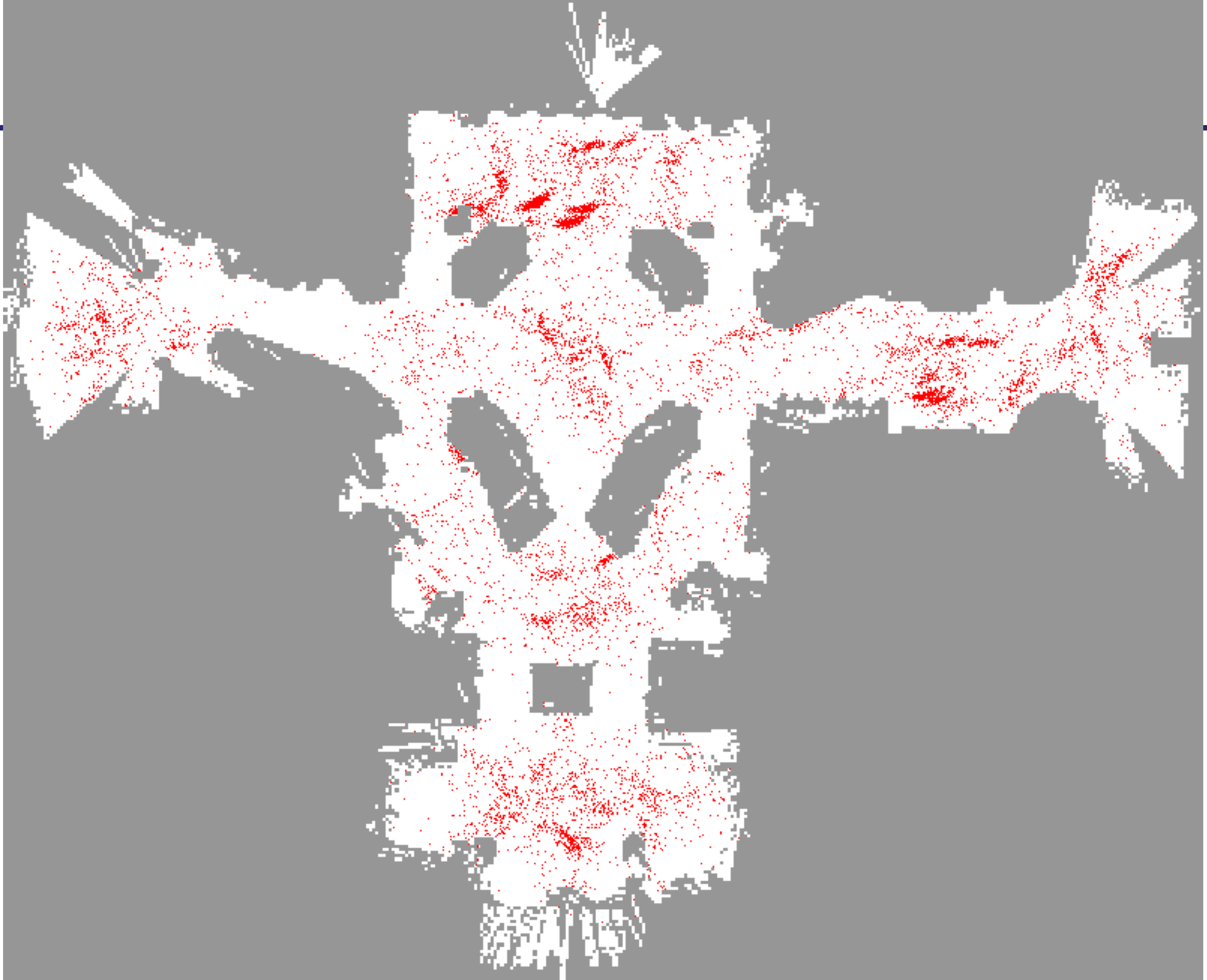
New: Normalized Importance Sampling with Resampling

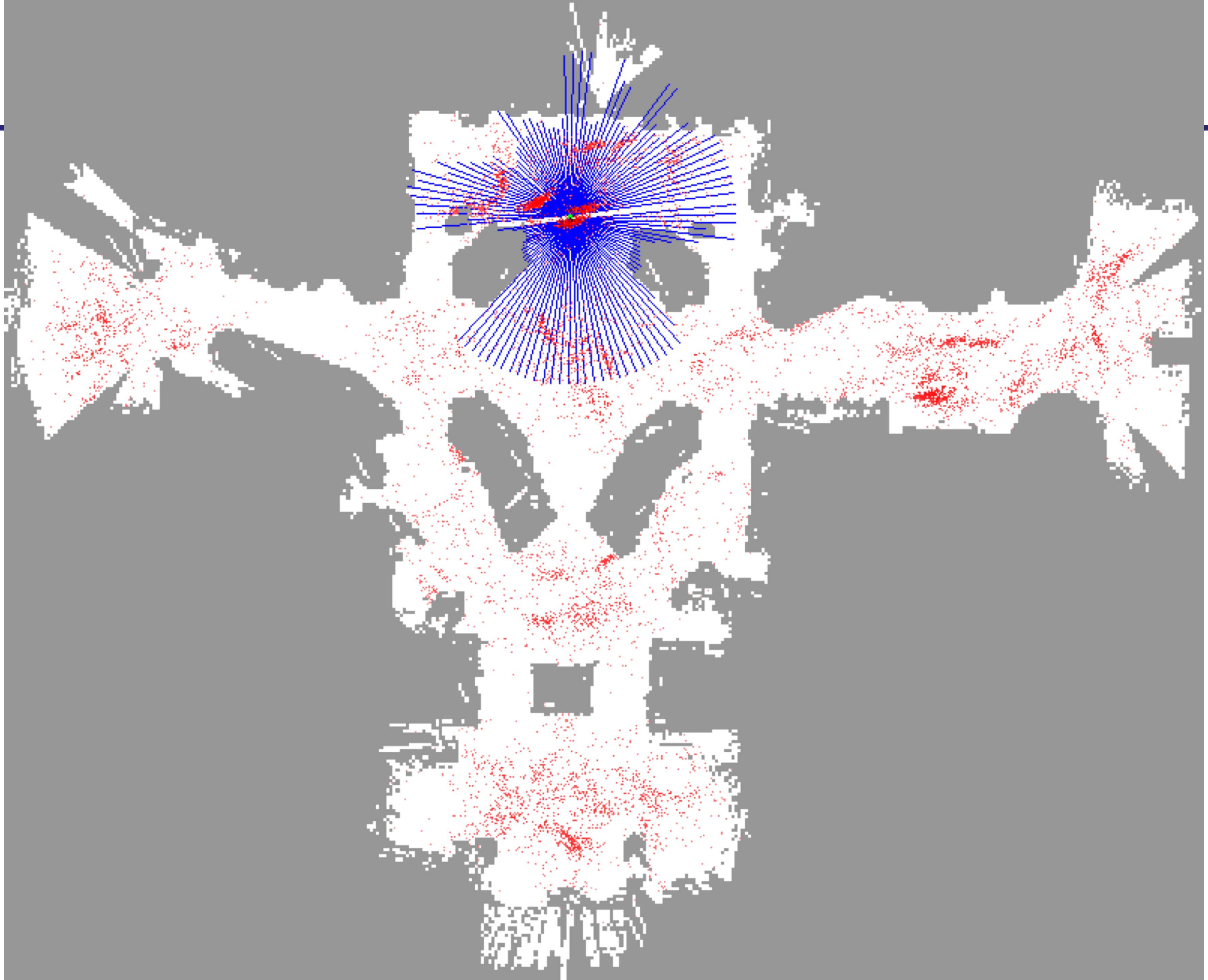


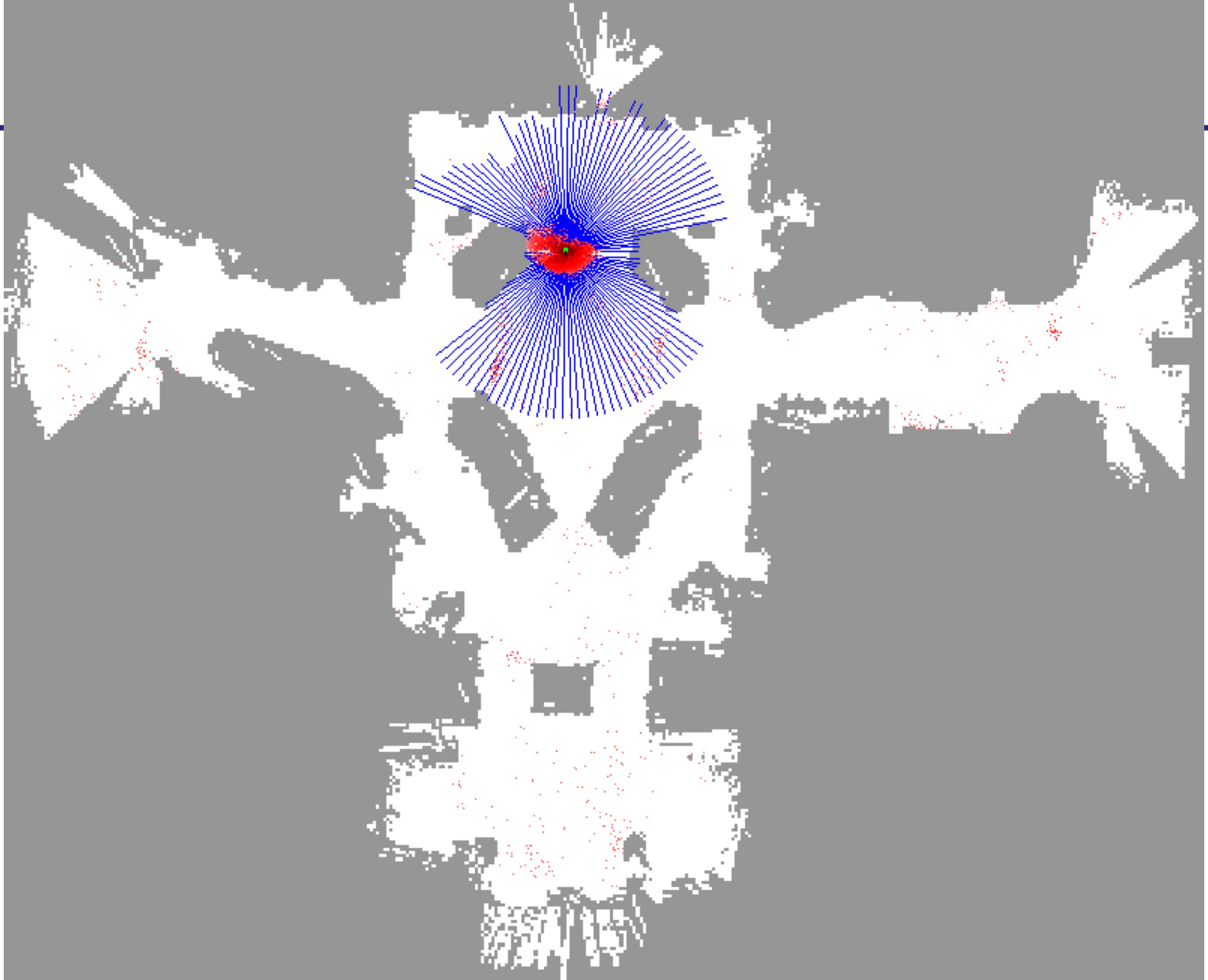
$$x_t^{(i)} \sim w_t^{(i)}, \; Bel(x_t) = \begin{Bmatrix} x_t^{(1)} & \cdots & x_t^{(M)} \\ \frac{1}{M} & \cdots & \frac{1}{M} \end{Bmatrix}$$

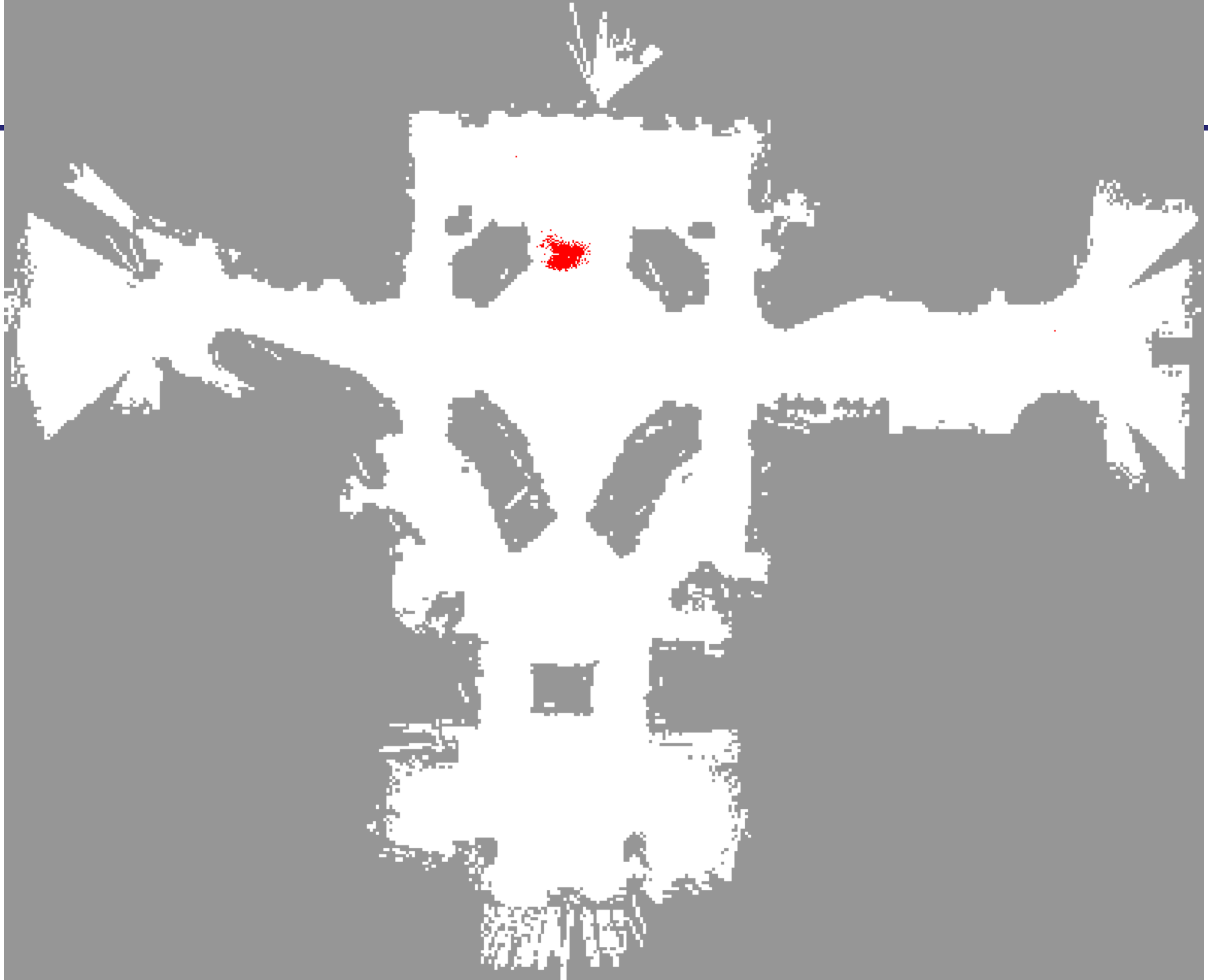












Overall Particle Filter algorithm – v2

Initial Prior

$p(x_0)$

Dynamics/Prediction

Estimate $\overline{Bel}(x_t)$

Sample particles from $p(x_t | x_{t-1}, u_{t-1})$

Measurement/Correction

Estimate $Bel(x_t)$

1. Weight samples by $p(z_t | x_t)$
2. Resample particles to get unweighted set

Lecture Outline

Particle Based Representations in Filtering

↓

Particle Filter

↓

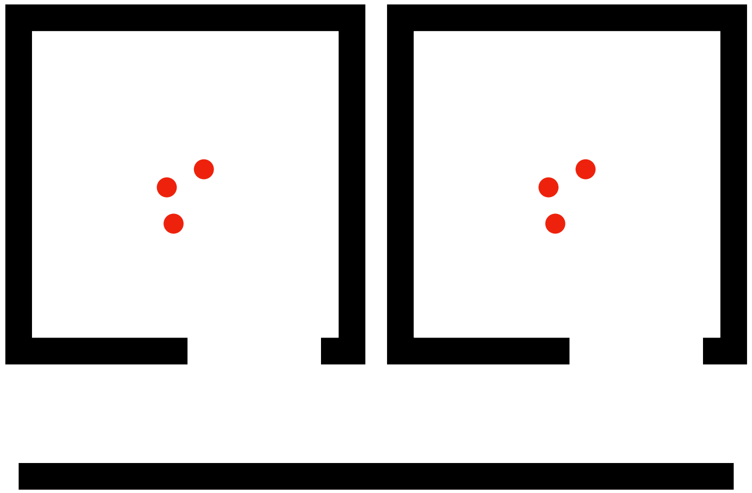
Particle Filter w/ Resampling

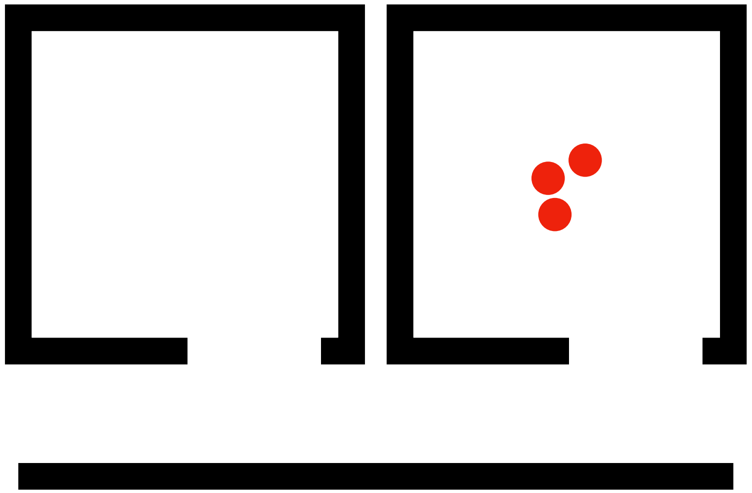
↓

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

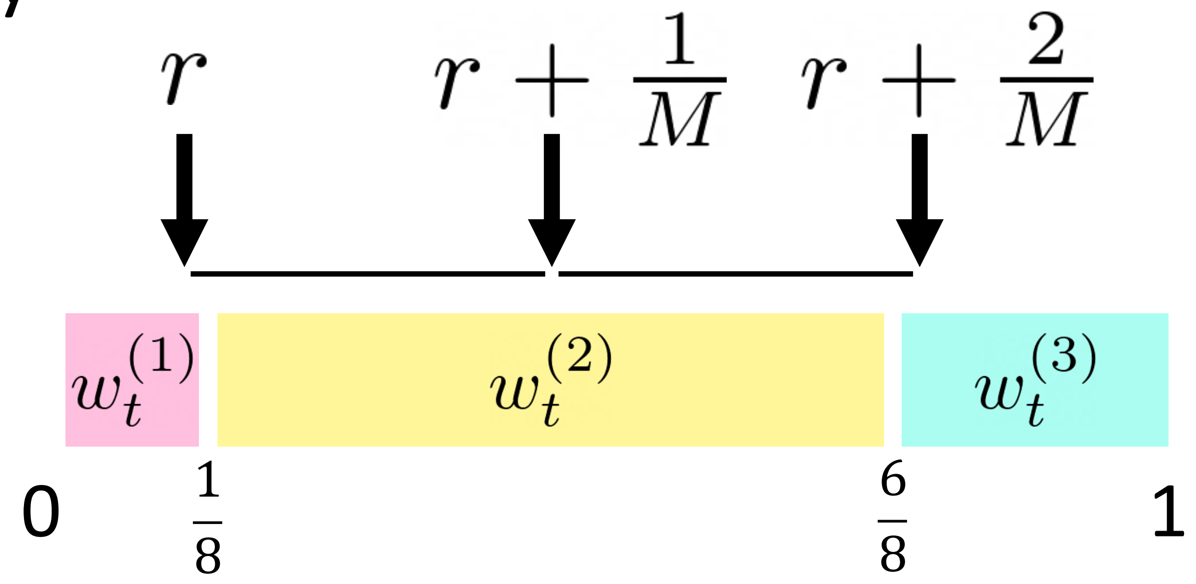
All particles migrate to one room!

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
  - 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

Can we get closed form updates for Bayesian Filtering?

Need to choose form of probability distributions

- Dynamics (Prediction)

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

- Measurement (Correction)

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

Tractable computation of Bayesian posteriors

Solution: Linear Gaussian Models

- Dynamics (Prediction)

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

- Measurement (Correction)

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

Model as Linear Gaussian

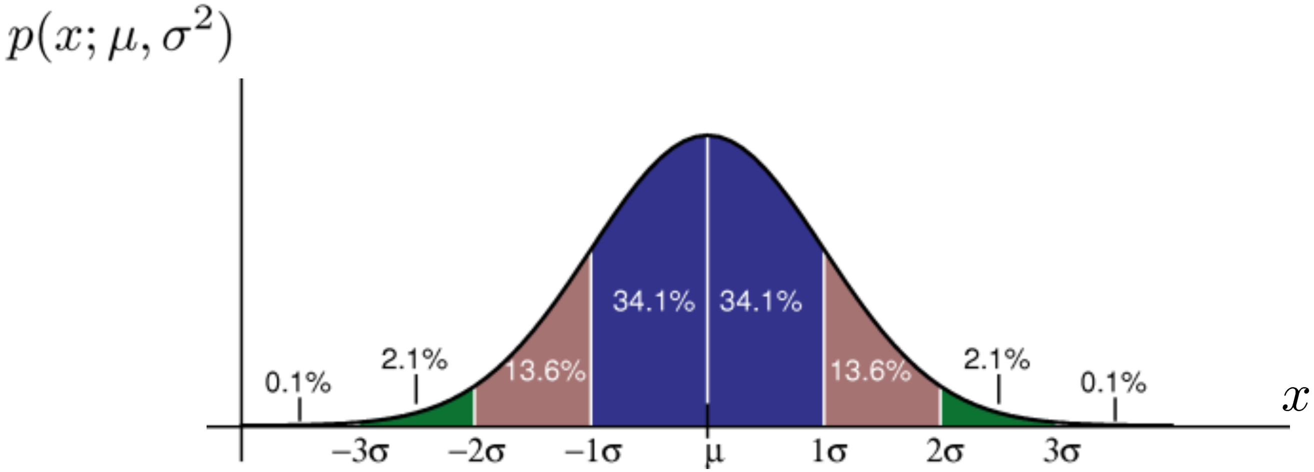
Let’s take a little Gaussian detour

Gaussians (1D)

- Gaussian with mean (μ) and standard deviation (σ)

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

$$p(x; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$



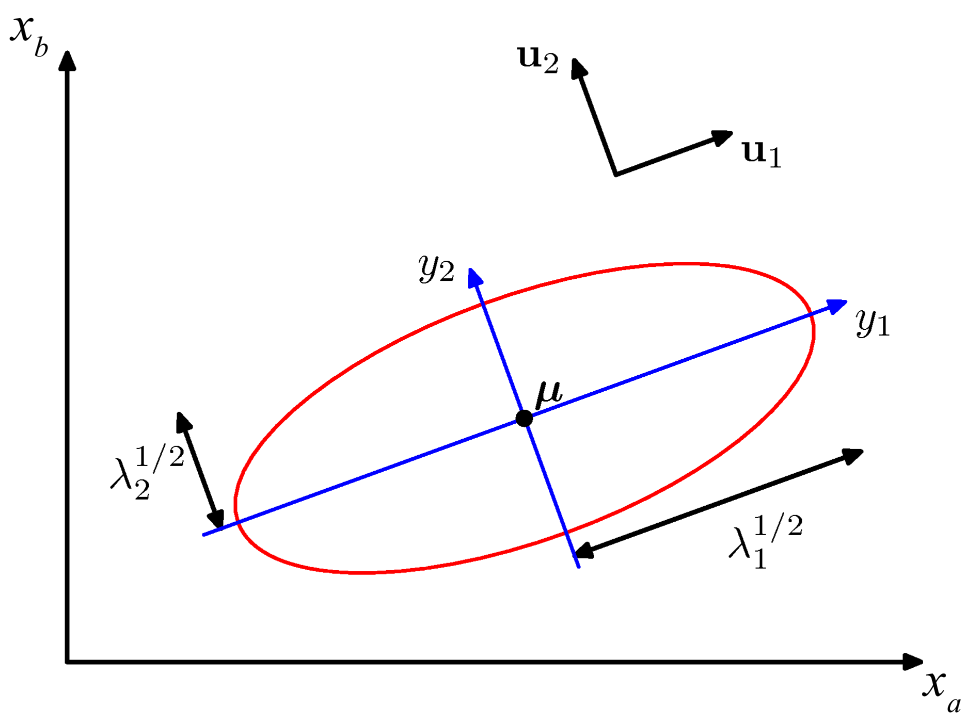
Gaussians (2D) – we won't get too deep into this!

$$p(\mathbf{x}) = \mathrm{N}(\mu, \Sigma)$$

$$\mathbf{x} = \begin{pmatrix} x_a \\ x_b \end{pmatrix}, \quad \mu = \begin{pmatrix} \mu_a \\ \mu_b \end{pmatrix}$$

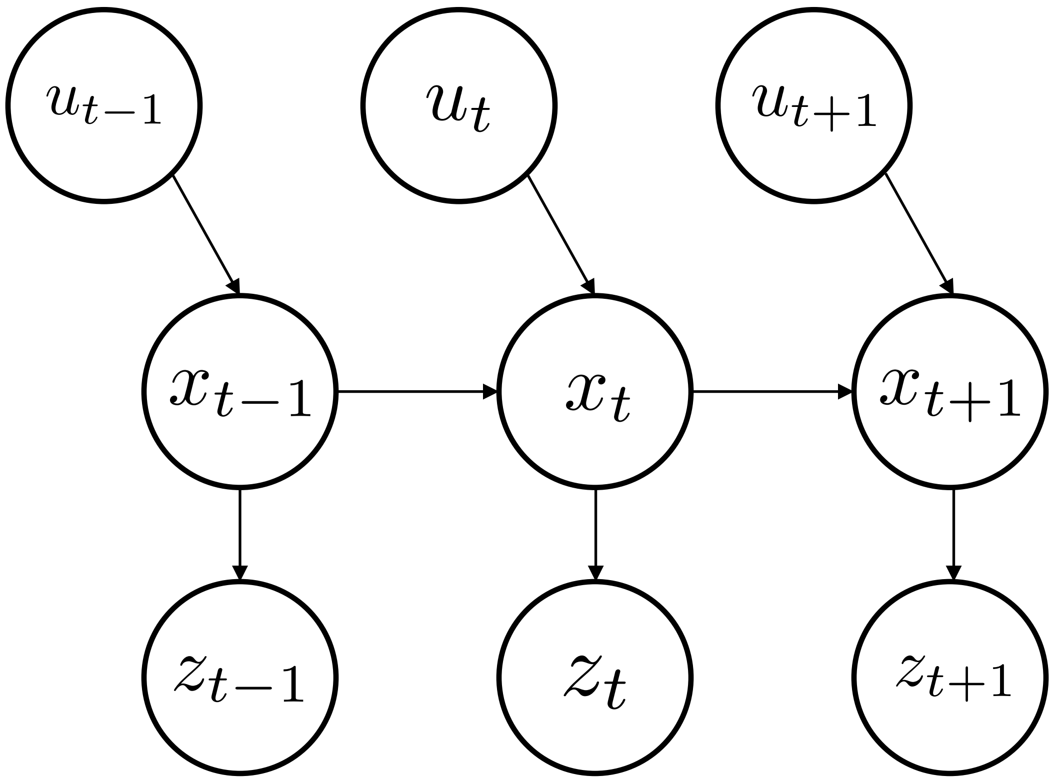
$$\Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix}$$

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



Discrete Kalman Filter

Kalman filter = Bayes filter with Linear Gaussian dynamics and sensor models



Discrete Kalman Filter: Scalar Version

Estimates the state ***x*** of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$x_t = a x_{t-1} + b u_t + \epsilon_t$$

$$\epsilon_t \sim \mathcal{N}(0, q)$$

with a measurement

$$z_t = c x_t + \delta_t$$

$$\delta_t \sim \mathcal{N}(0, r)$$

Linear Gaussian

Discrete Kalman Filter: Matrix Version

Estimates the state ***x*** of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$x_t = Ax_{t-1} + Bu_t + \epsilon_t$$

$$\epsilon_t \sim \mathcal{N}(0, Q)$$

with a measurement

$$z_t = Cx_t + \delta_t$$

$$\delta_t \sim \mathcal{N}(0, R)$$

Linear Gaussian

Components of a Kalman Filter

A Matrix (n x n) that describes how the state evolves from ***t-1*** to ***t*** without controls or noise.

B Matrix (n x l) that describes how the control $\boldsymbol{u_{t-1}}$ changes the state from ***t-1*** to ***t***

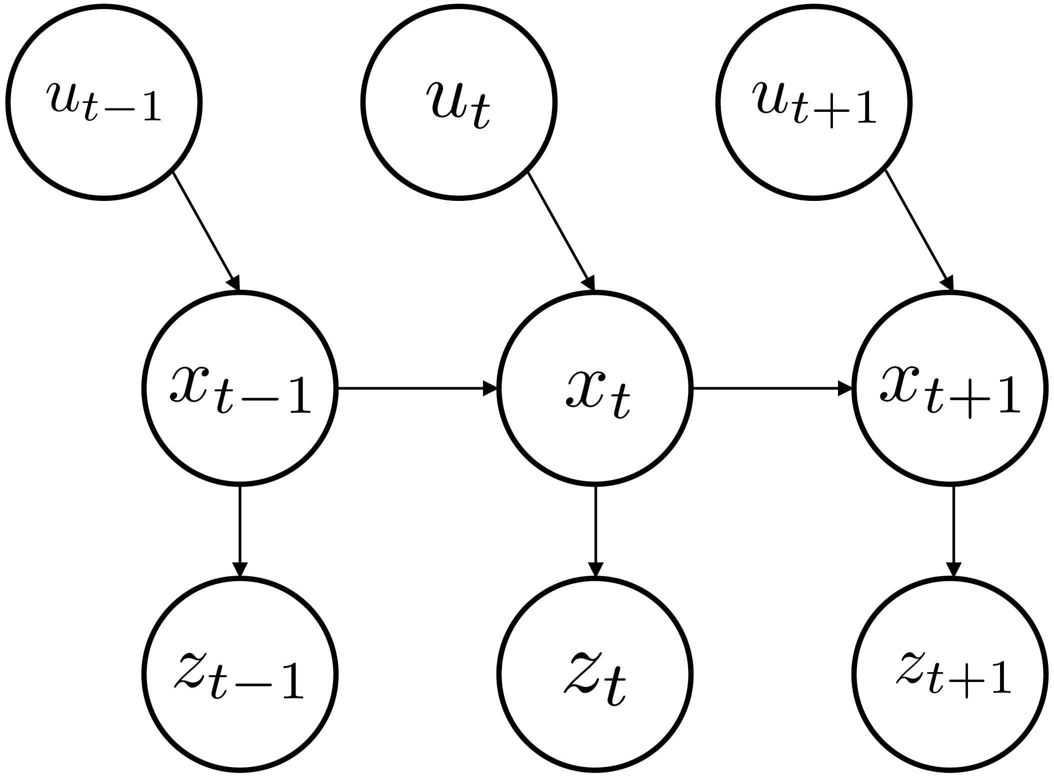
C Matrix (k x n) that describes how to map the state $\boldsymbol{x_t}$ to an observation $\boldsymbol{z_t}$.

ϵ_t

δ_t

Random variables representing the process and measurement noise that are assumed to be independent and normally distributed with covariance ***R*** and ***Q*** respectively.

Goal of the Kalman Filter: Same as Bayes Filter



Belief

$$p(x_t | z_{0:t}, u_{0:t})$$

Idea: recursive update

$$\propto p(z_t|x_t) \int p(x_t|x_{t-1}, u_t) p(x_{t-1}|z_{0:t-1}, u_{0:t-1})$$

Measurement

Dynamics

Recursive Belief

2 step process:

- Dynamics update (incorporate action)
- Measurement update (incorporate sensor reading)

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