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

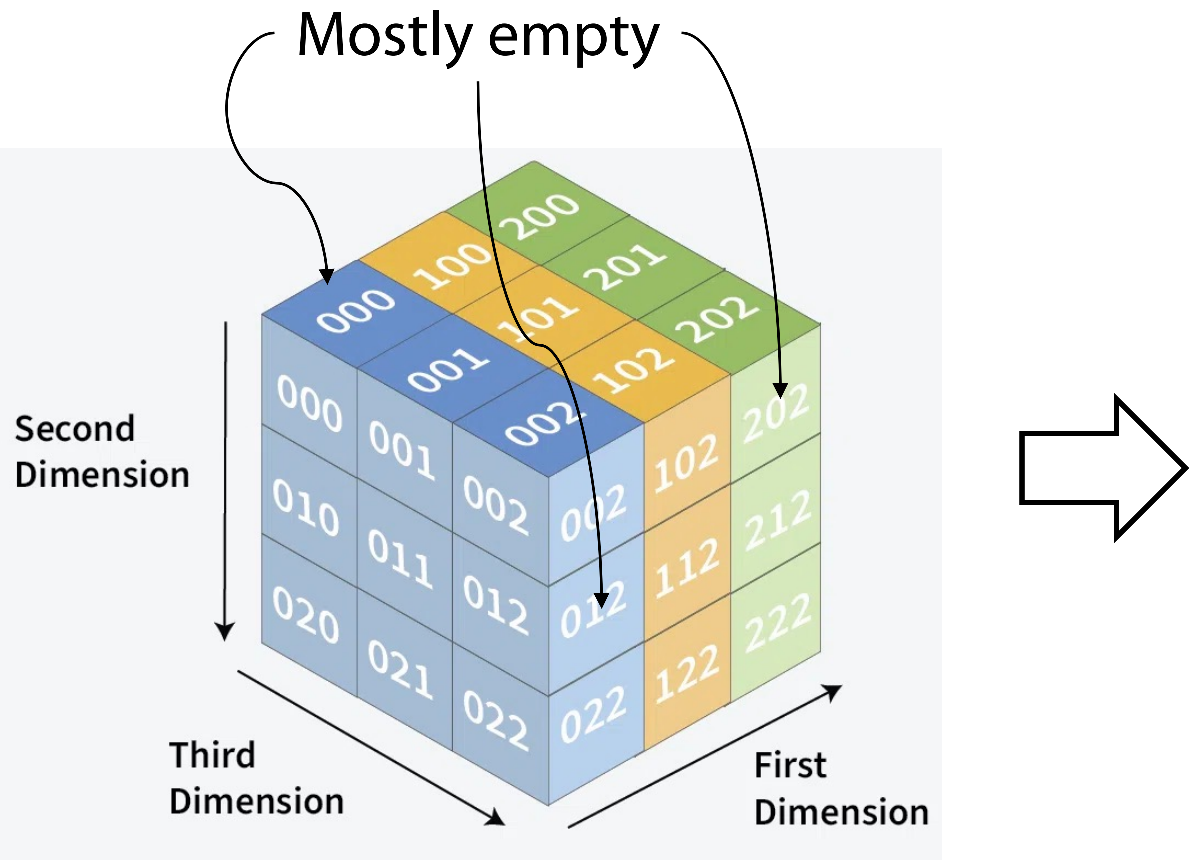
Logistics

- Project 2 underway, start early!
- First seeded paper discussion groups will be sent out Saturday for next Friday

- 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

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!

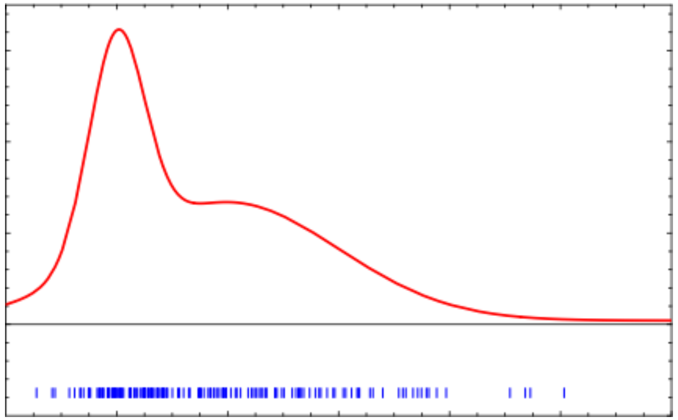
Bringing this Back to Estimation – Belief Distribution

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?

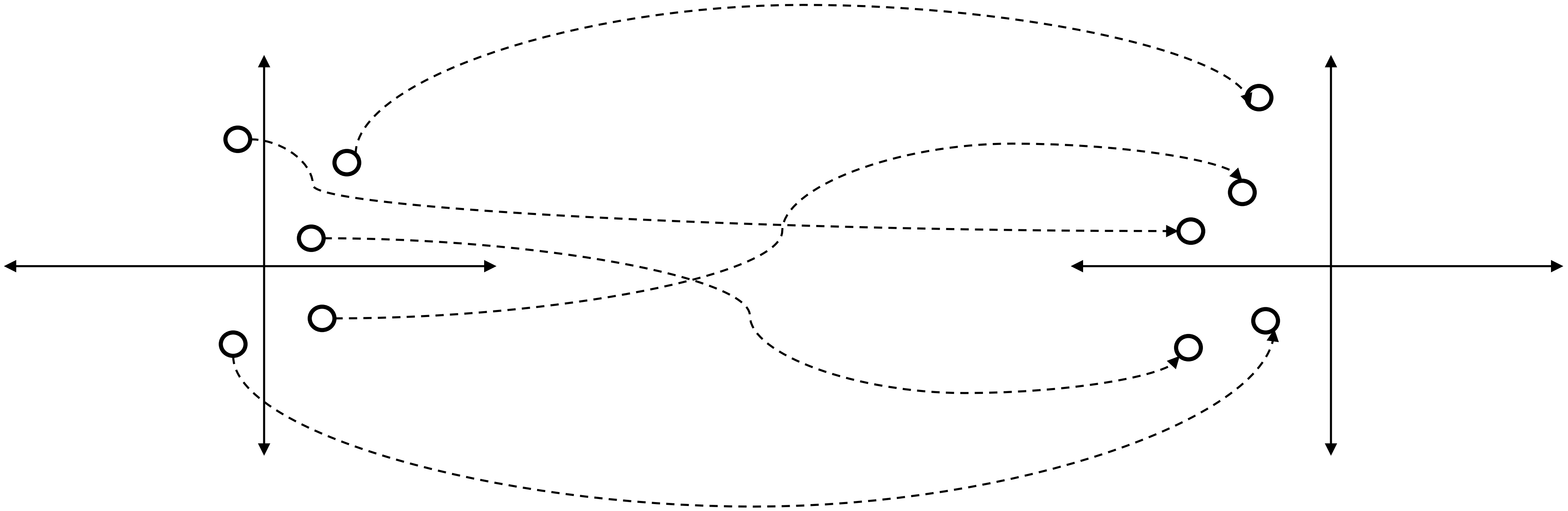
How do we compute conditioning/normalization with particles?

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

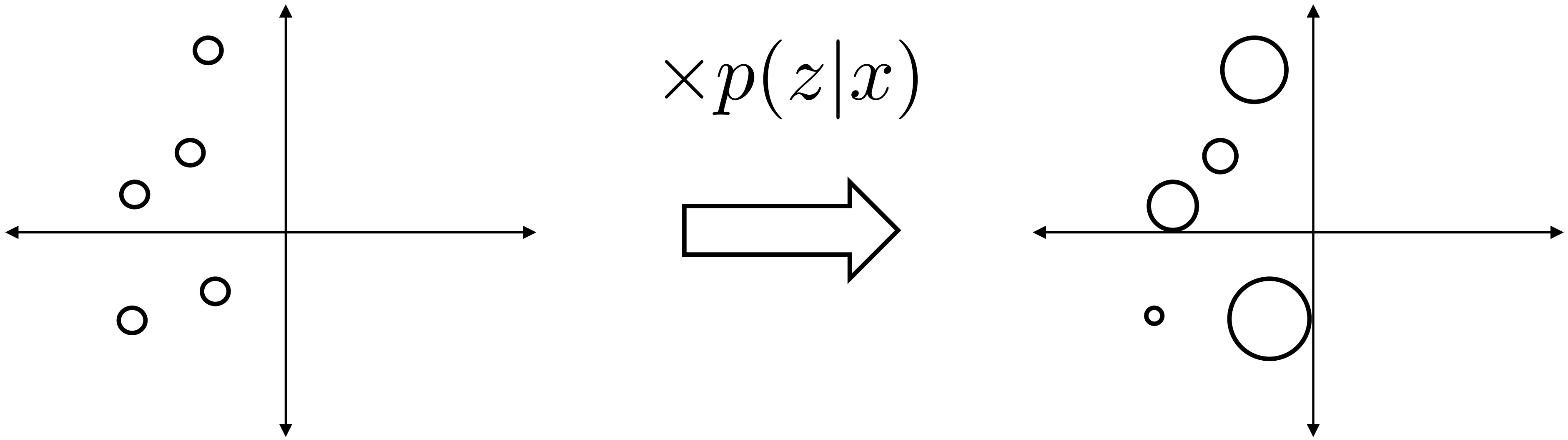


Measurement Update

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

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

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



Reweight particles according to measurement likelihood

Lecture Outline

Recap

↓

Particle Filter w/ Resampling

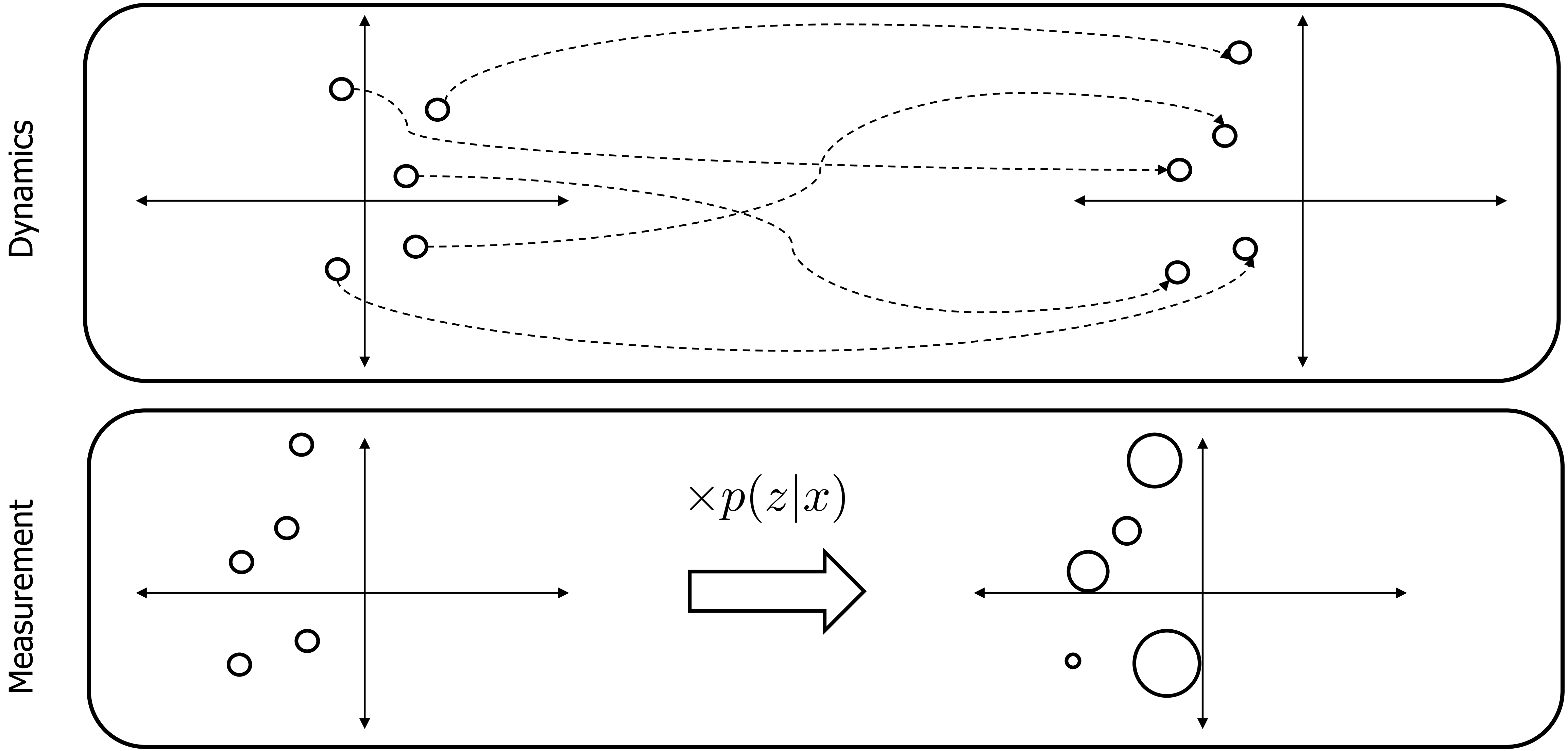
↓

Fixes to particle filter resampling

↓

Kalman Filters

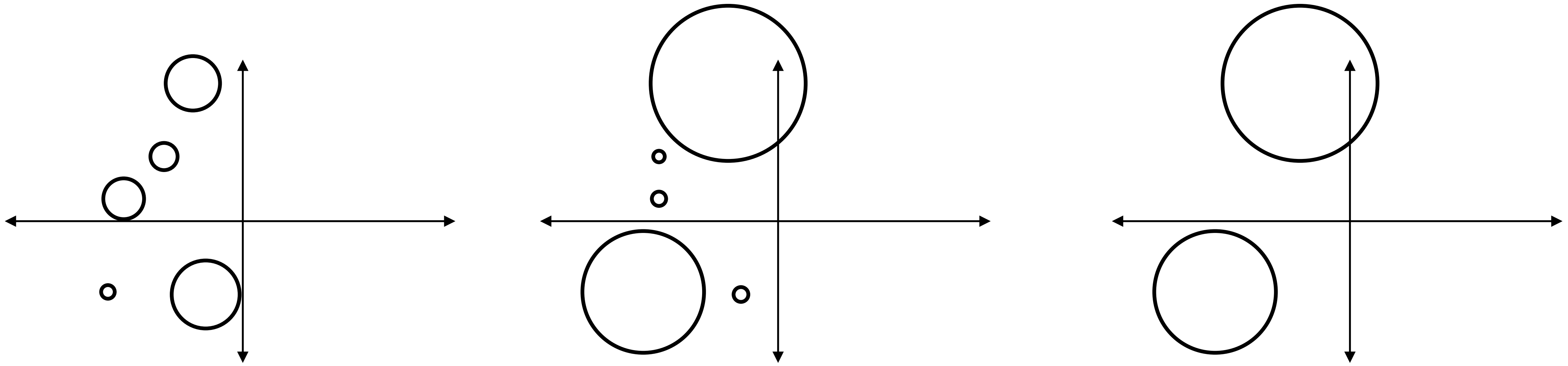
What happens across multiple steps?



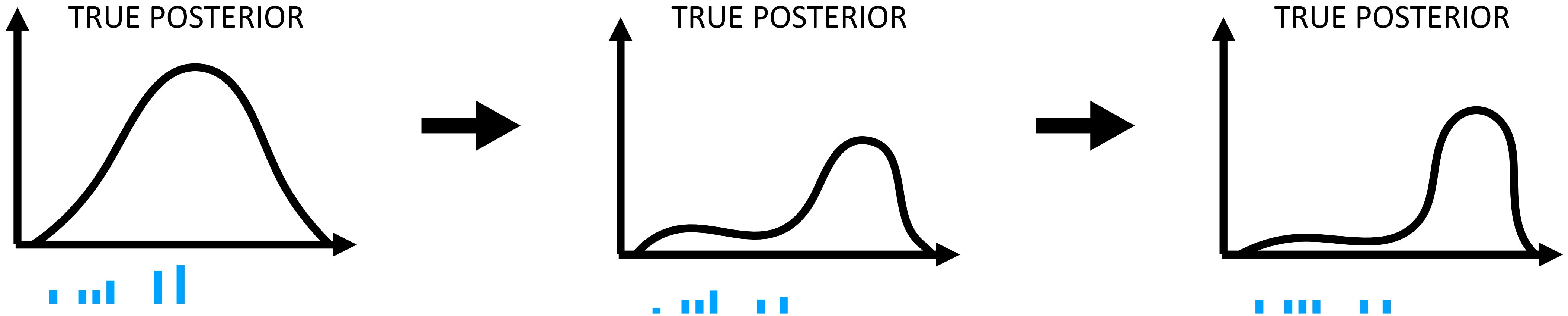
Importance weights get multiplied at each step

Why might this be bad?

Importance weights get multiplied at each step



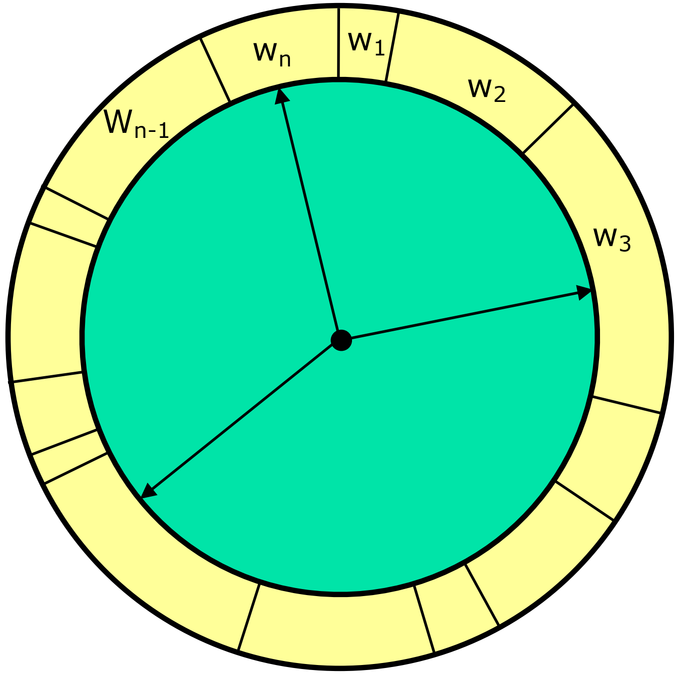
1. May blow up and get numerically unstable over many steps
2. Particles stay stuck in unlikely regions



Resampling

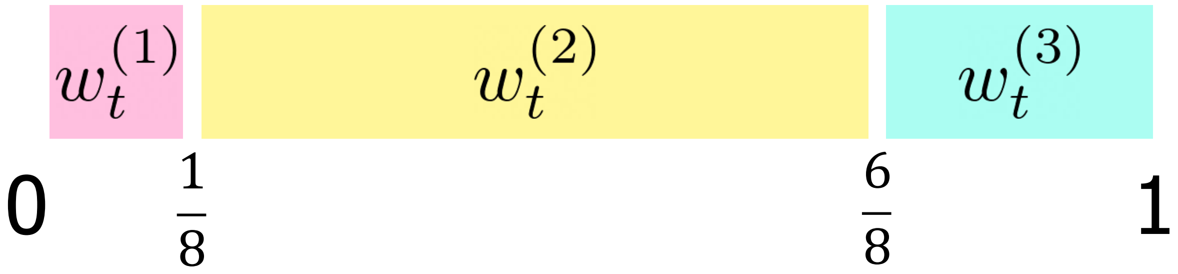
- **Given**: Set ***S*** of weighted samples (from measurement step) with weights w_i
- **Wanted** : unweighted random sample, where the probability of drawing $\boldsymbol{x_i}$ is given by $\boldsymbol{w_i}$.
- Typically done ***n*** times with replacement to generate new sample set ***S'***.

Resampling



Here are your random numbers:

0.97
0.26
0.72



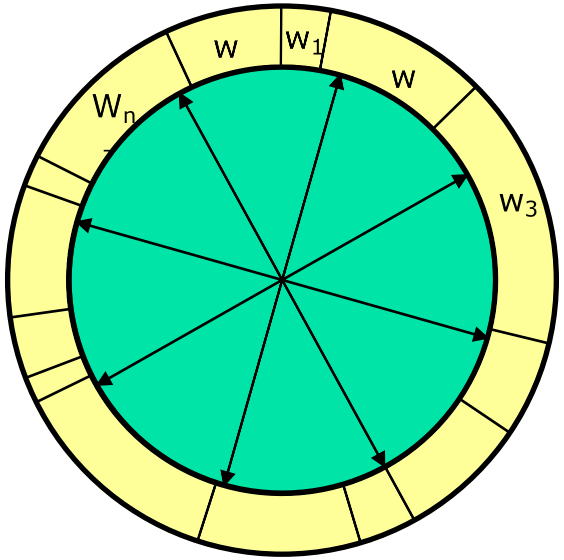
- Spin a roulette wheel
- Space according to weights
- Pick samples based on where it lands

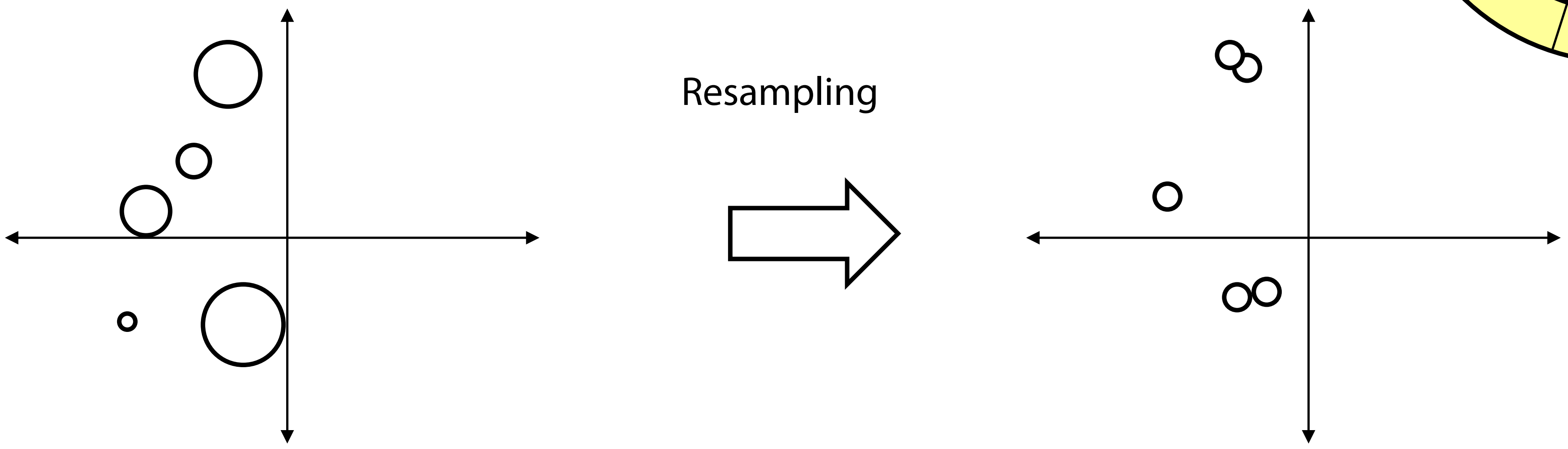
Resampling in a particle filter

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

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

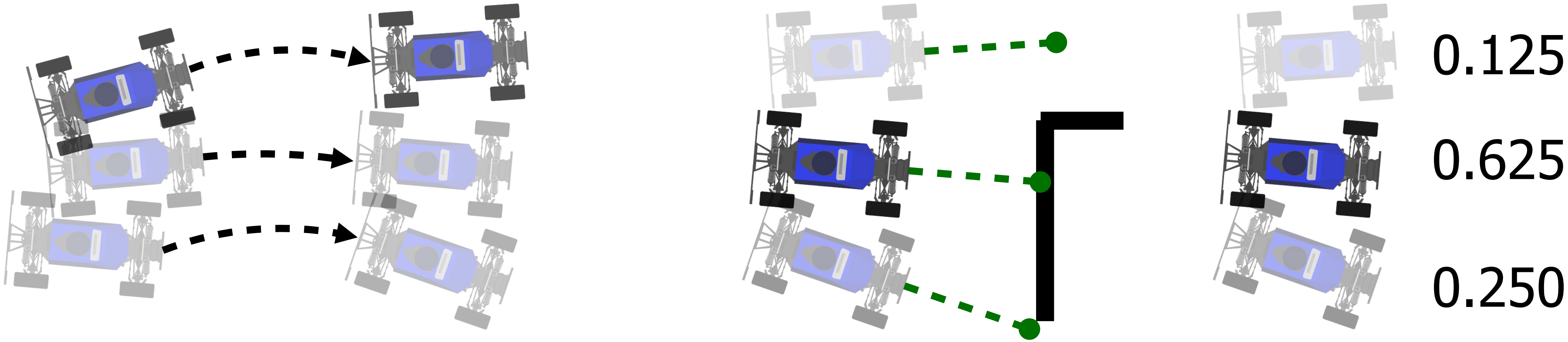
$$w_i = \frac{P(z_t|x_t^i)}{\sum_j P(z_t|x_t^j)}$$





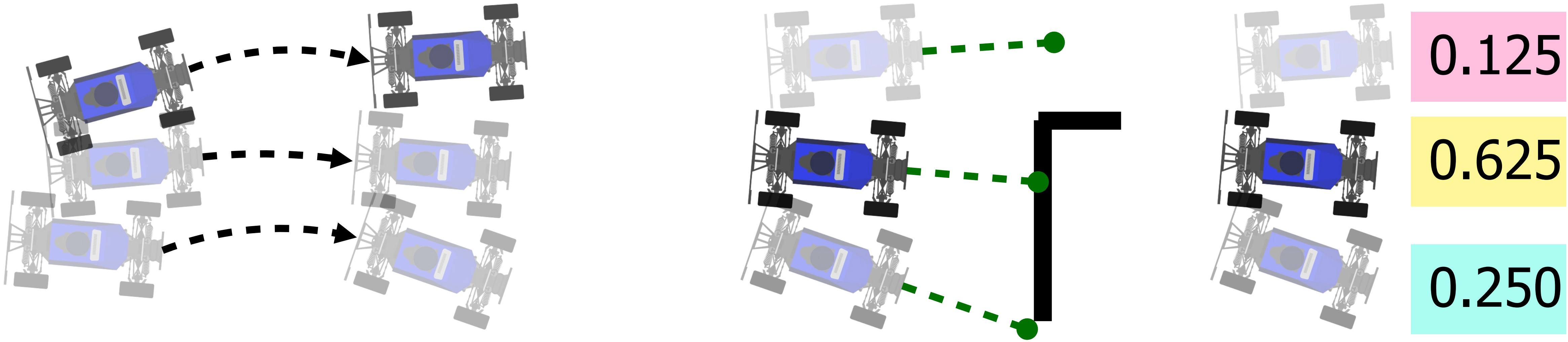
Resample particles from weighted distribution to give unweighted set of particles

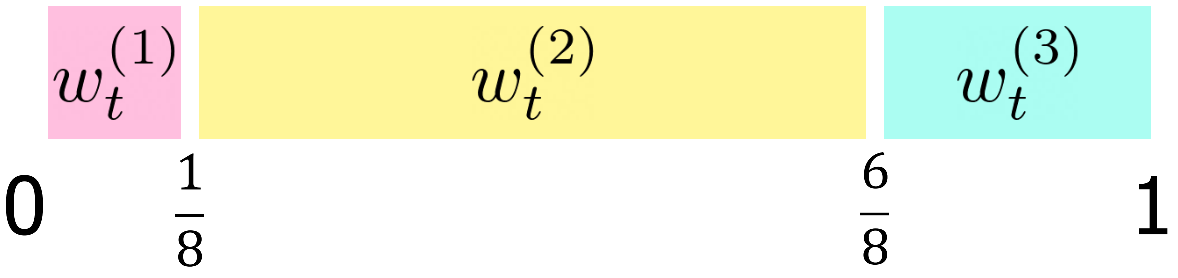
Original: Normalized Importance Sampling



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

New: Normalized Importance Sampling with Resampling

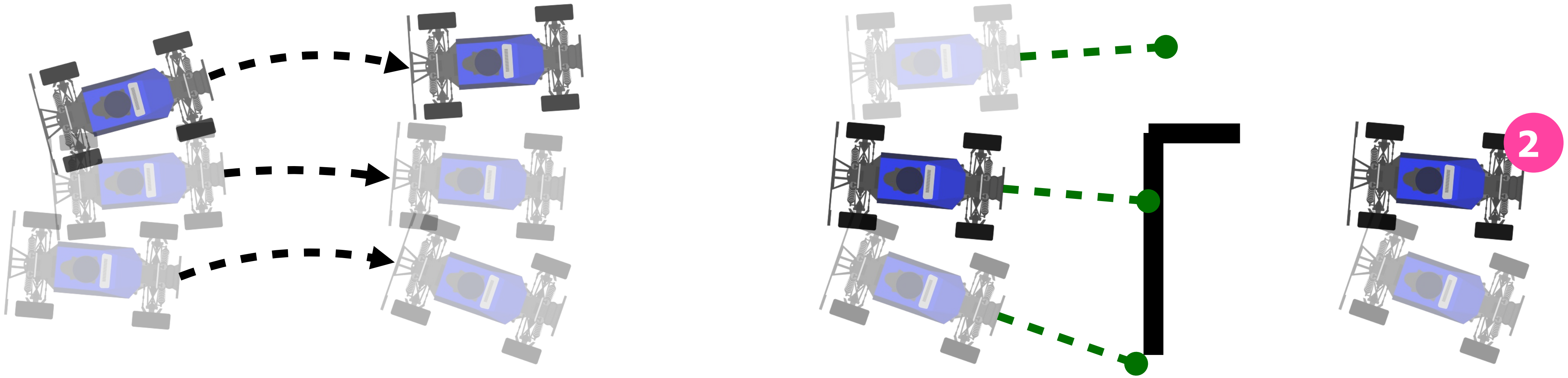




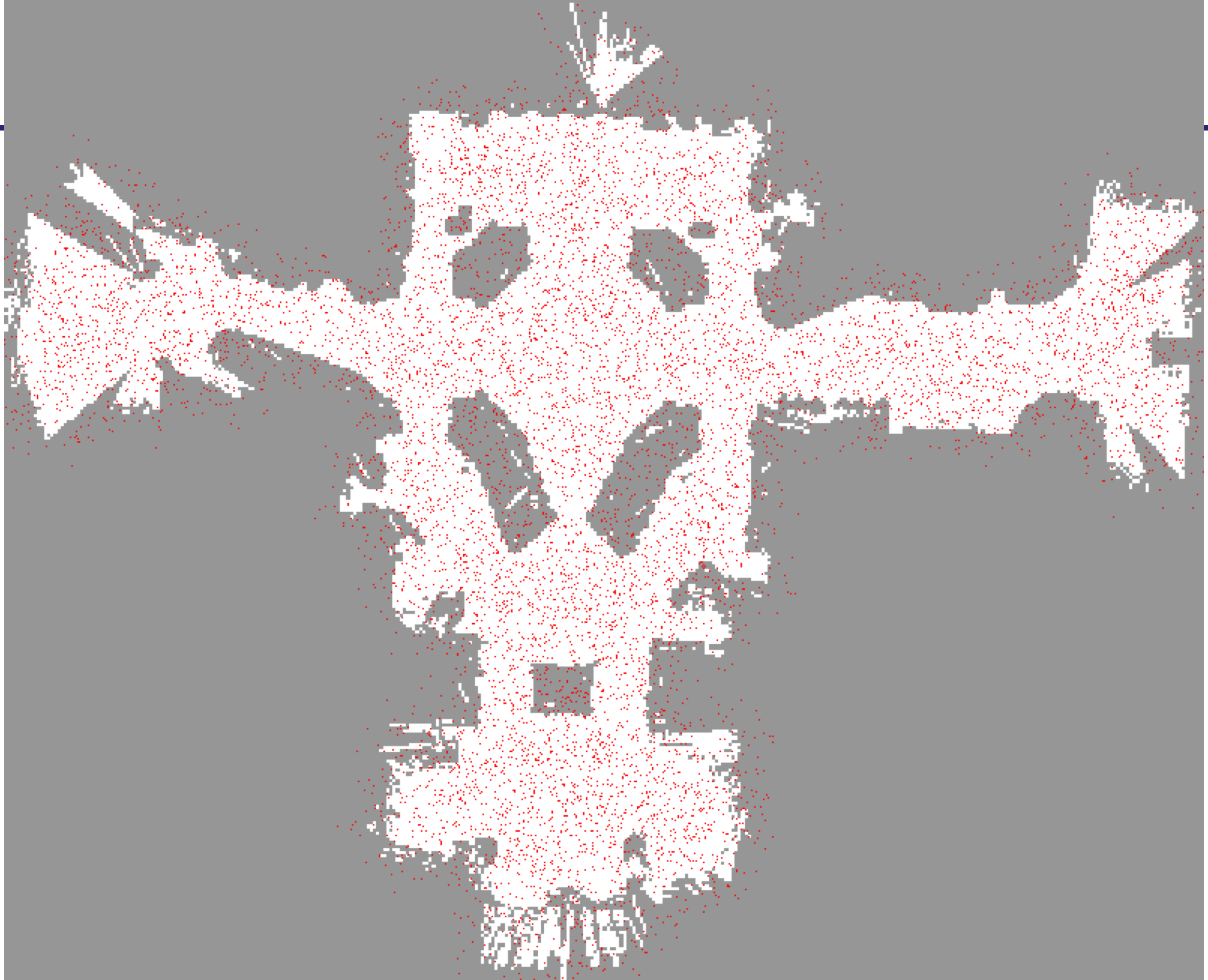
Here are your random numbers:

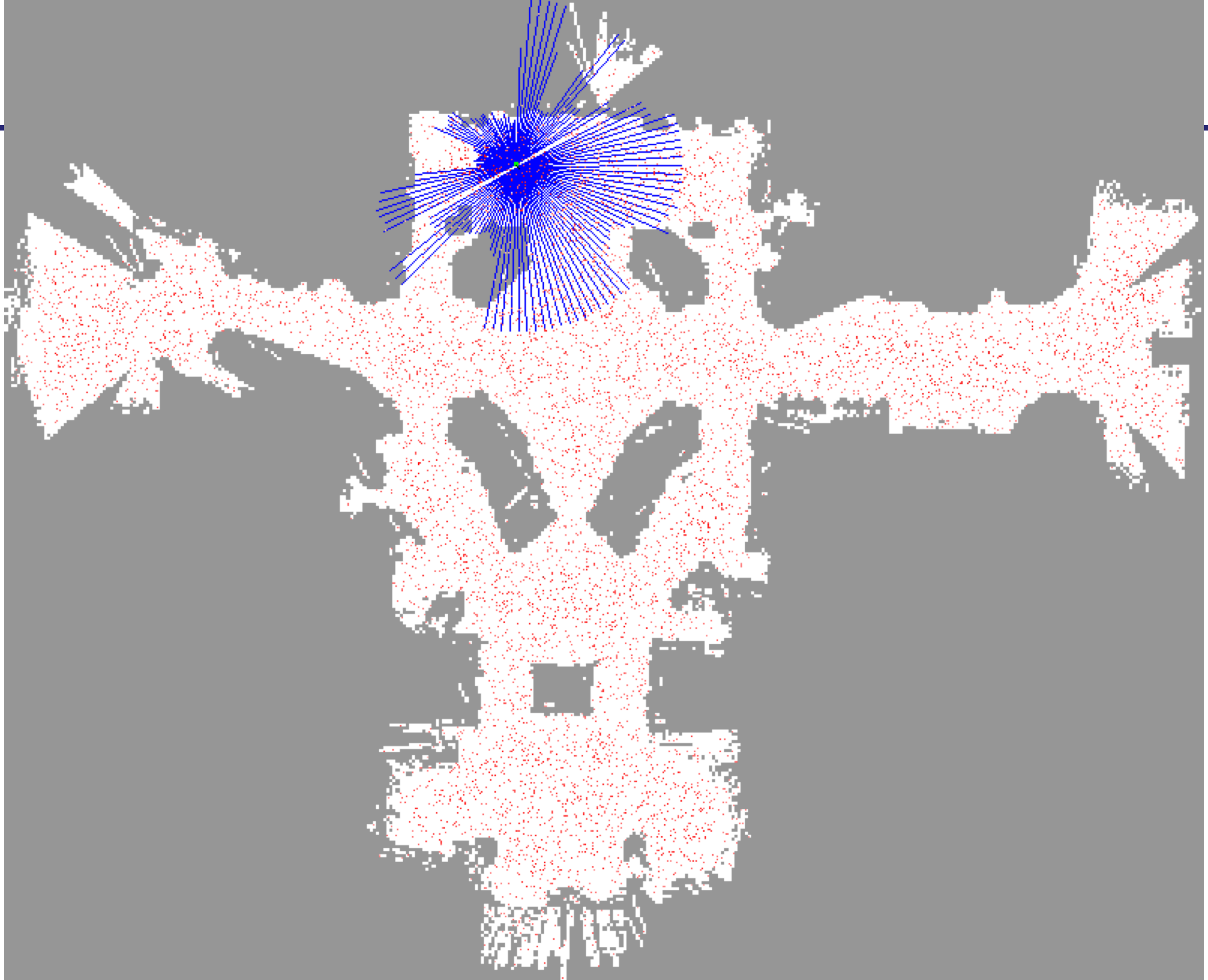
0.97
0.26
0.72

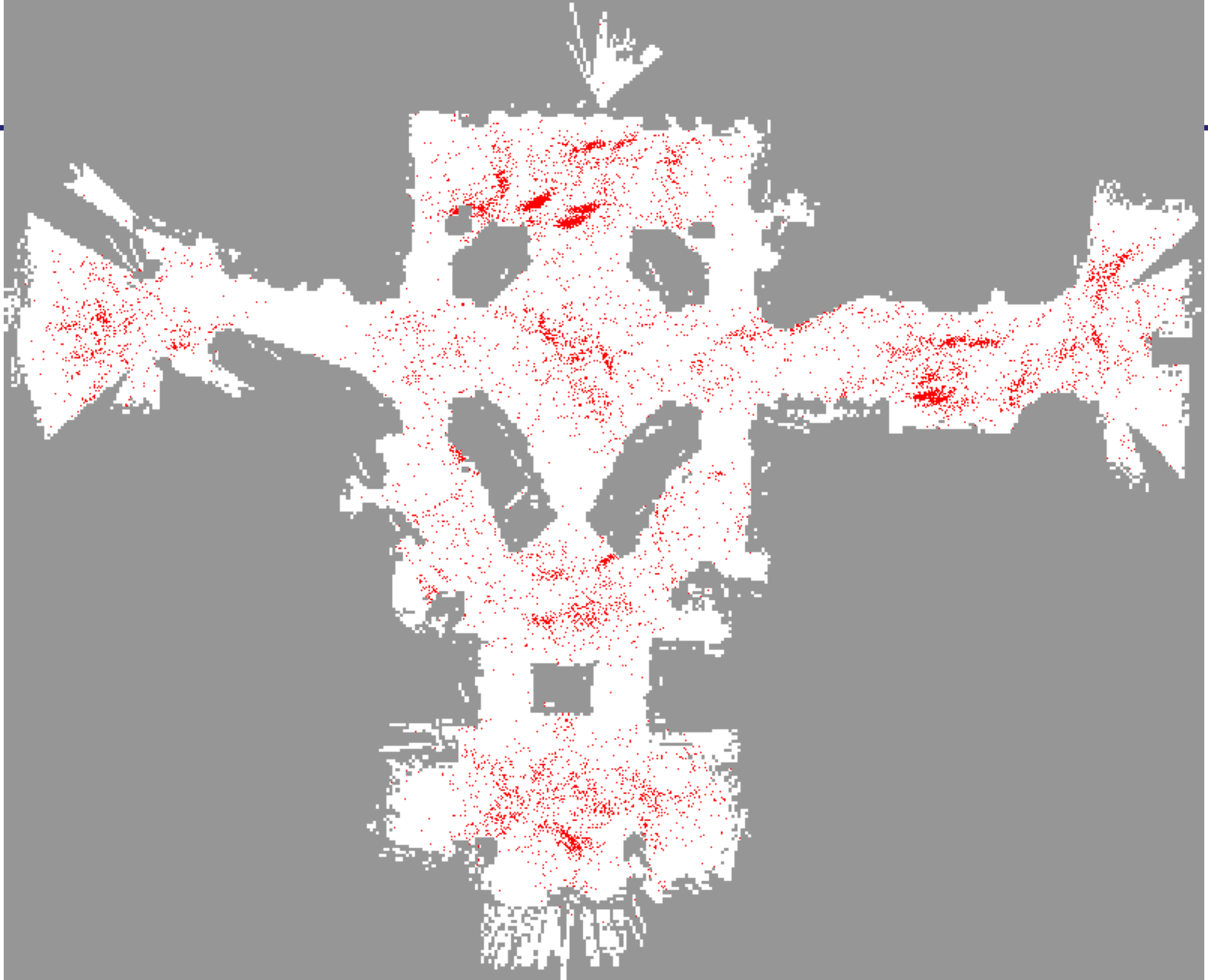
New: Normalized Importance Sampling with Resampling

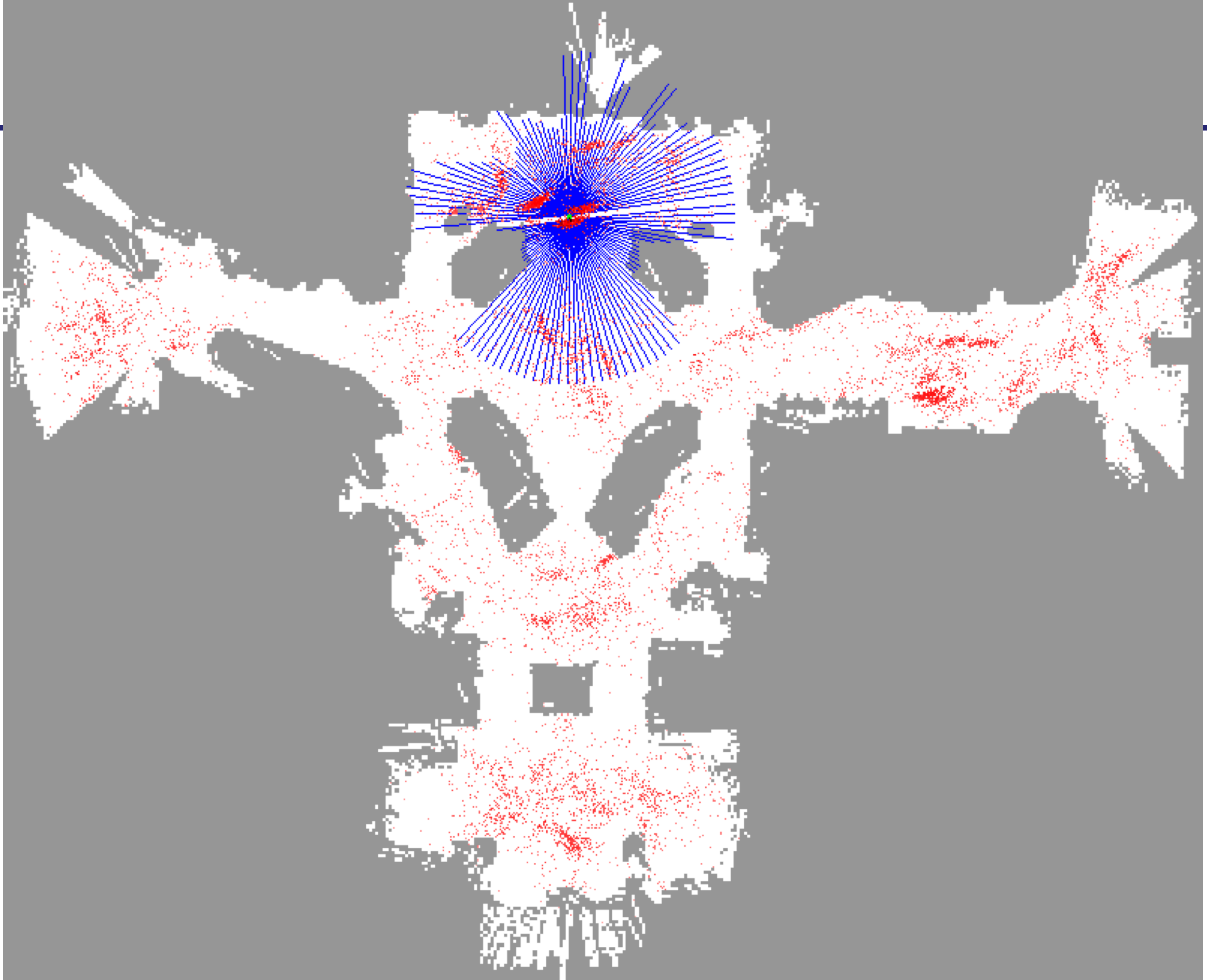


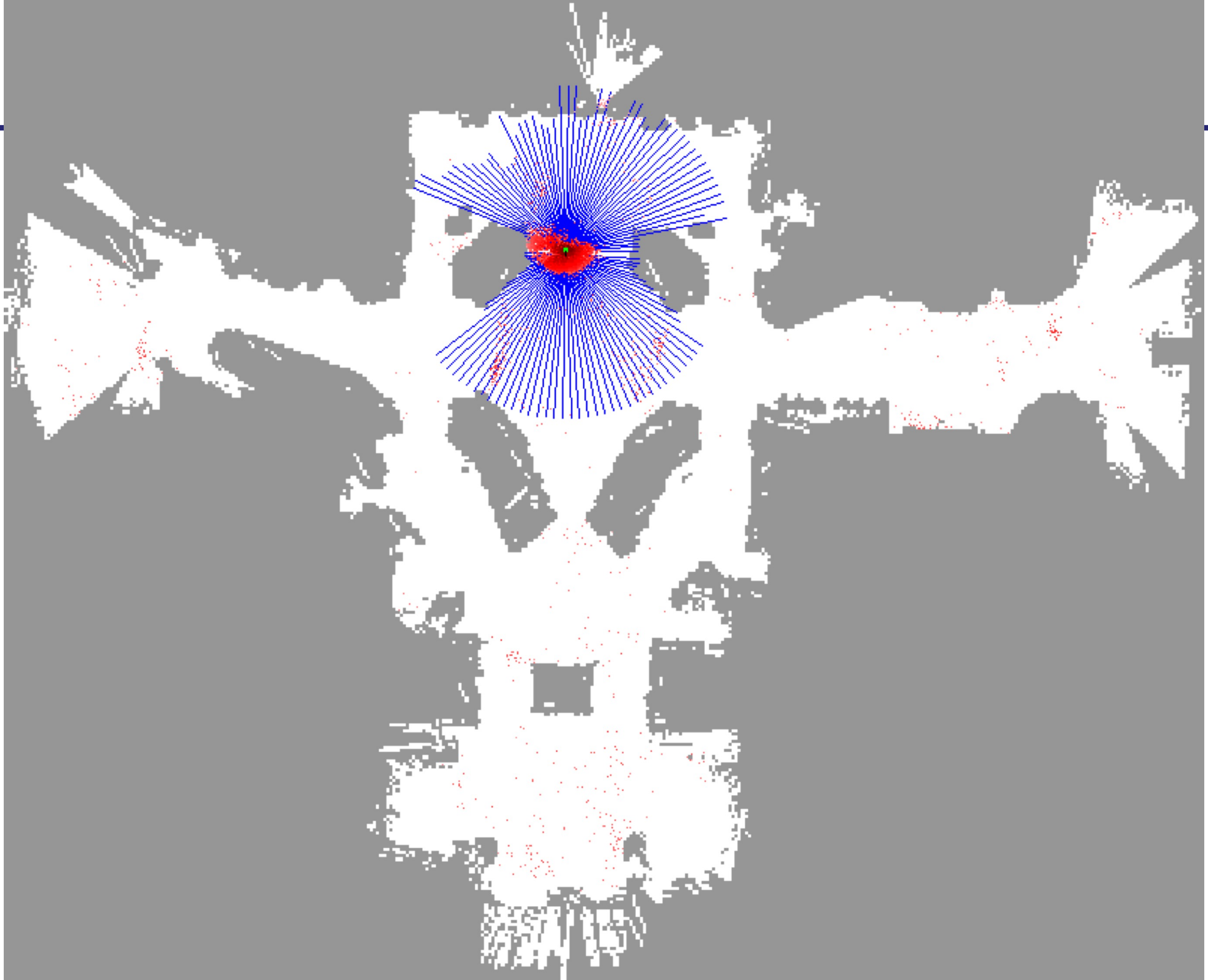
$$x_t^{(i)} \sim w_t^{(i)}, \; Bel(x_t) = \left\{ \begin{array}{ccc} x_t^{(1)} & \cdots & x_t^{(M)} \\ \frac{1}{M} & \cdots & \frac{1}{M} \end{array} \right\}$$

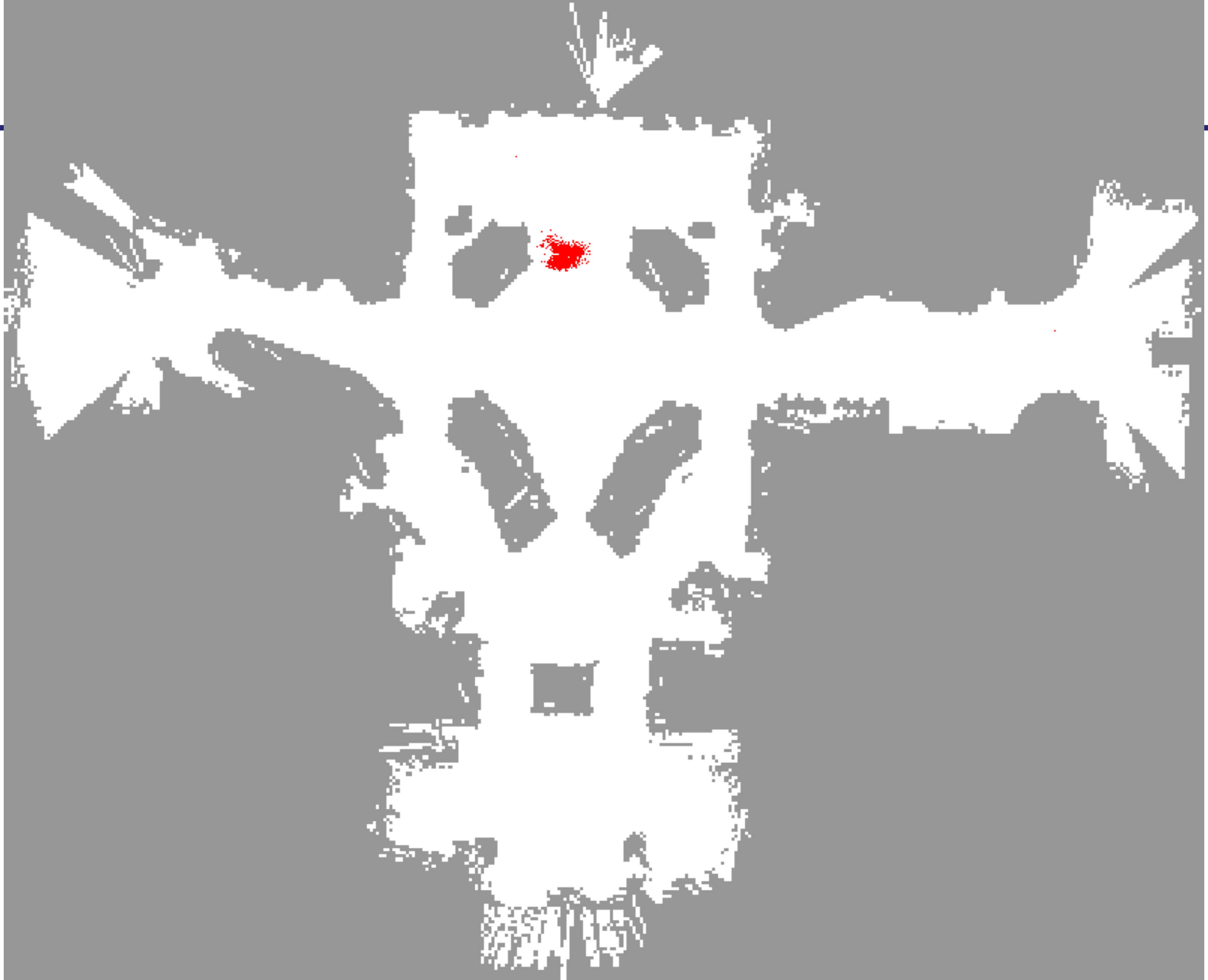




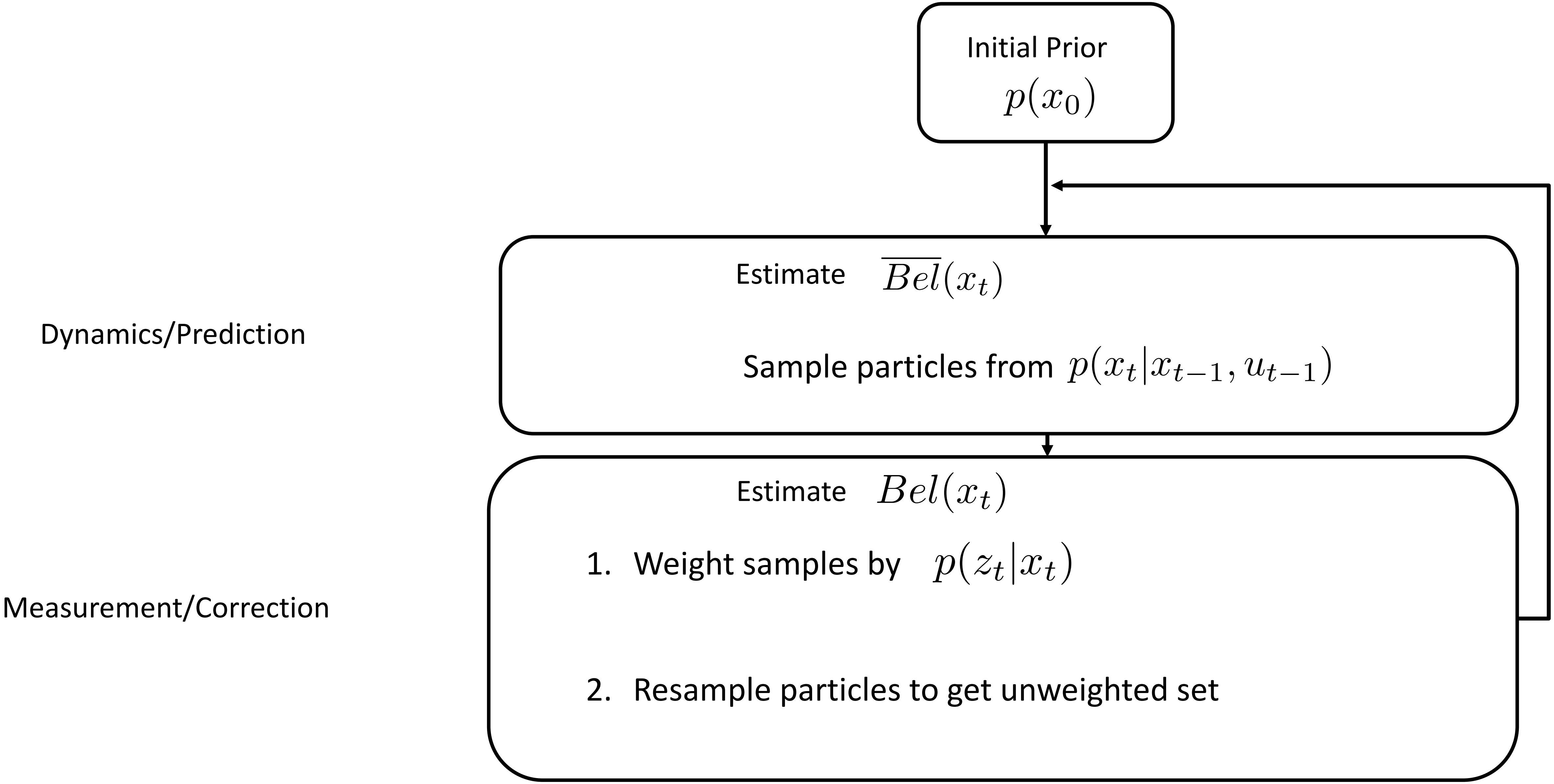








Overall Particle Filter algorithm – v2



Lecture Outline

Recap

↓

Particle Filter w/ Resampling

↓

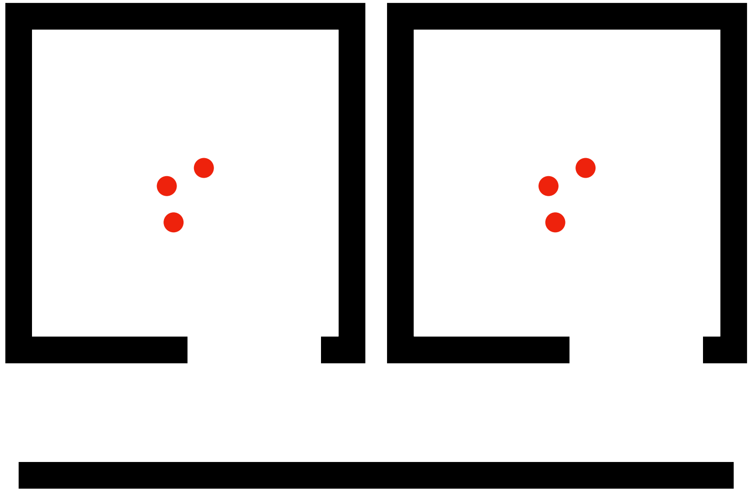
Fixes to particle filter resampling

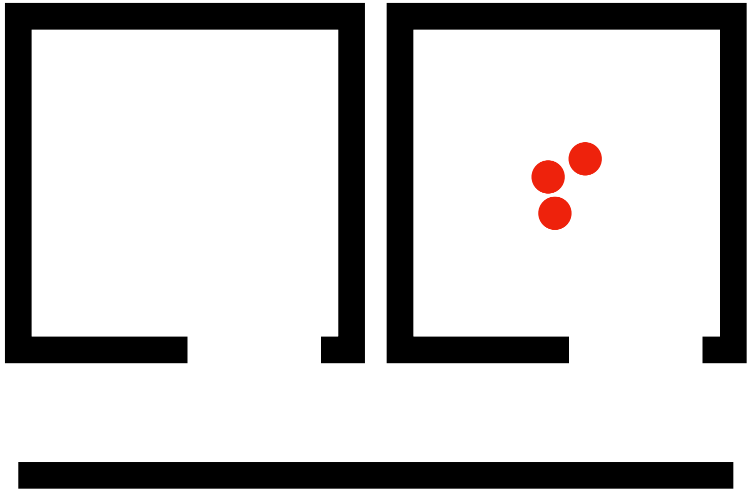
↓

Kalman Filters

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

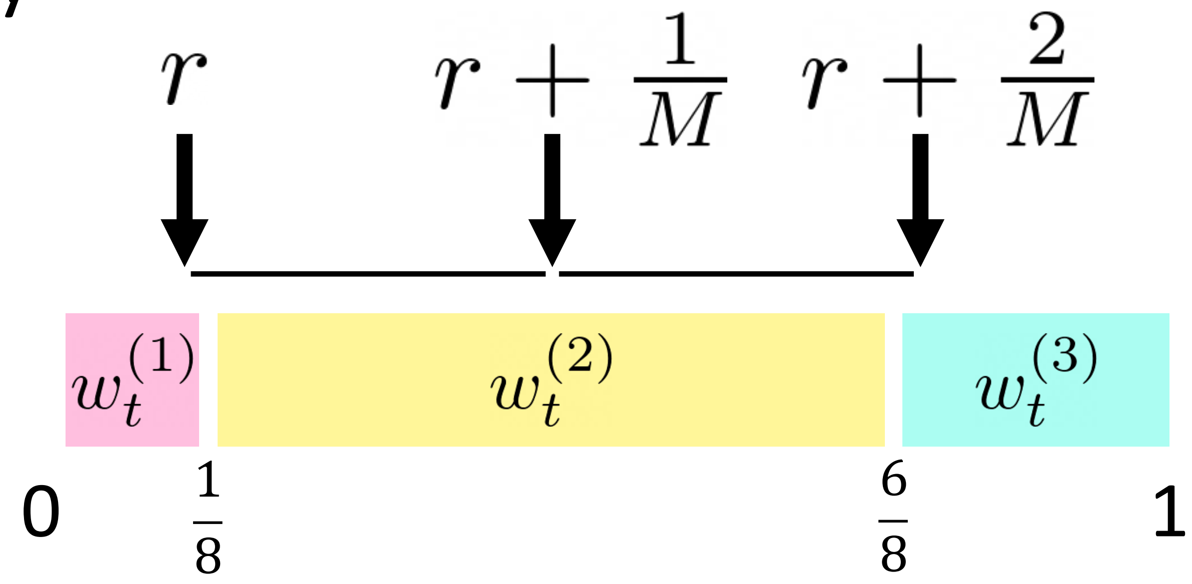
All particles migrate to one room!

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

Lecture Outline

Recap

↓

Particle Filter w/ Resampling

↓

Fixes to particle filter resampling

↓

Kalman Filters

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

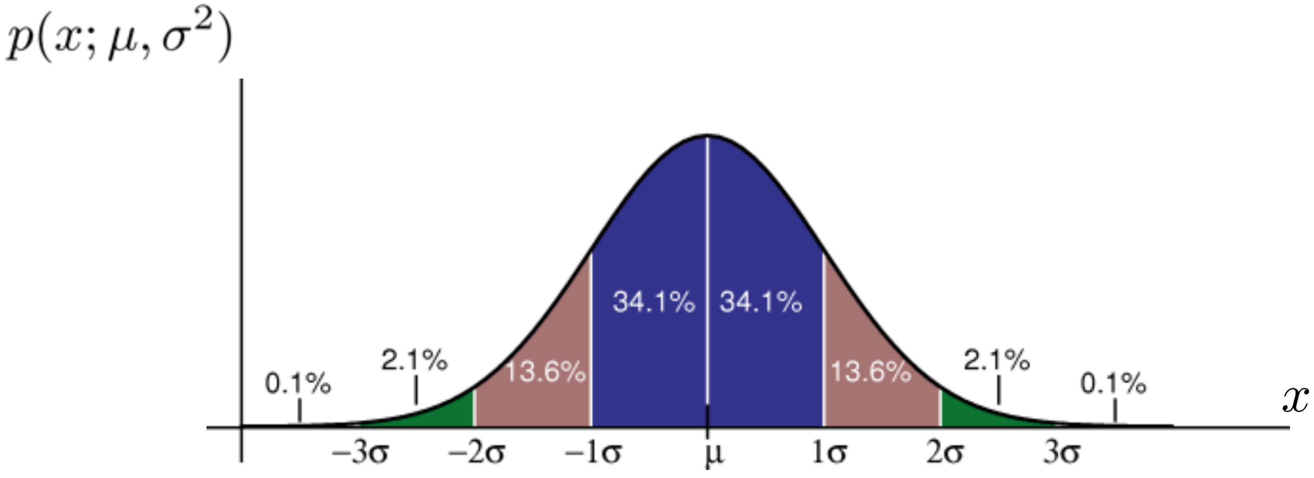
Let’s take a little Gaussian detour

Gaussians (1D)

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

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

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



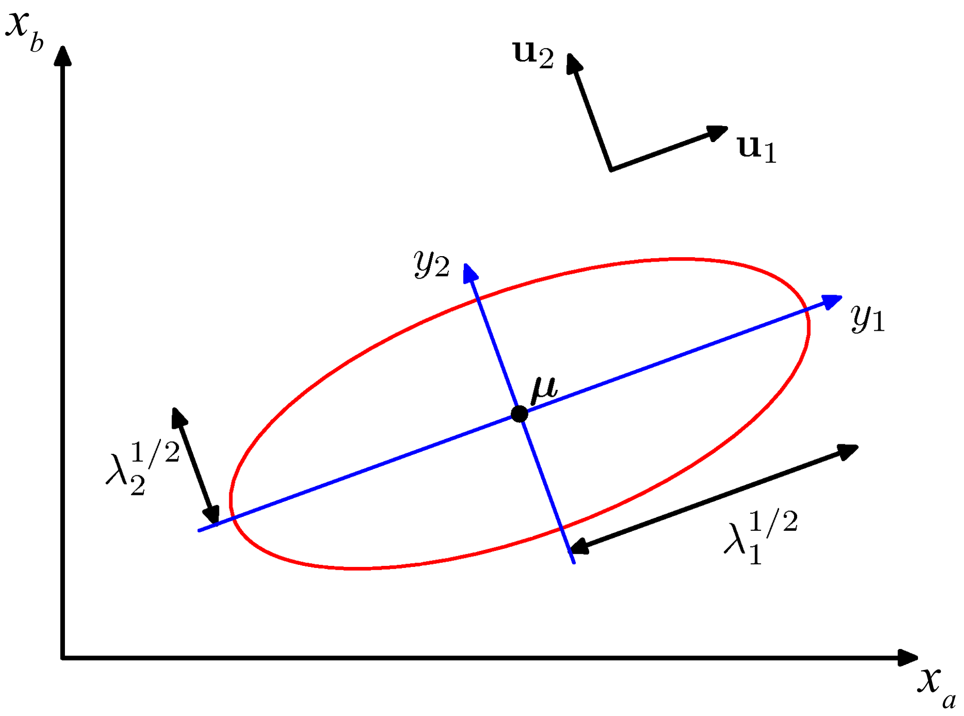
Gaussians (2D) – we won't get too deep into this!

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

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

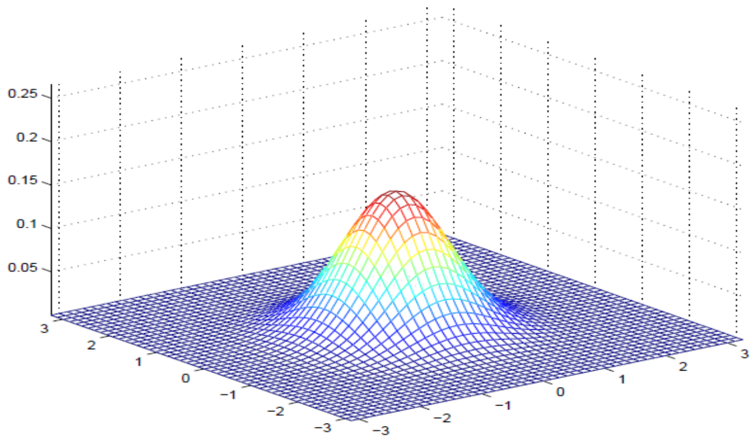
$$\Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix}$$

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

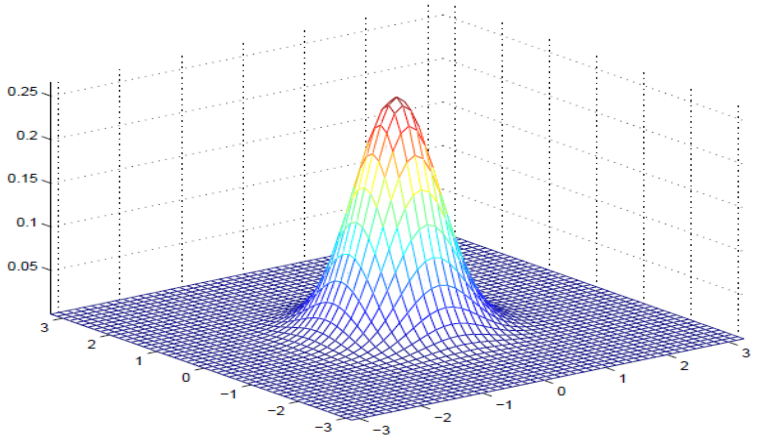


2D examples

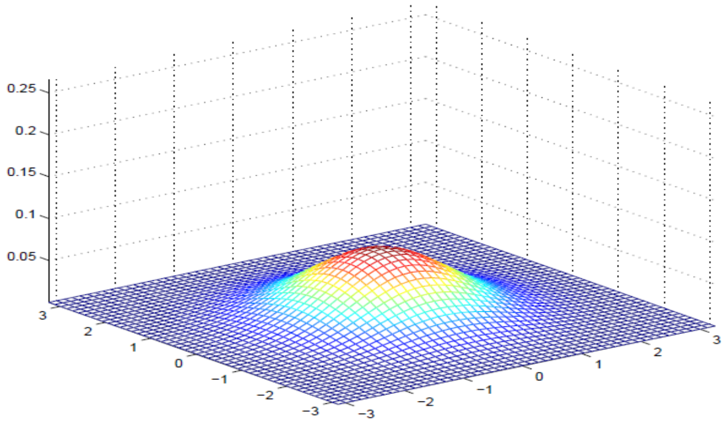
Slide from Pieter Abbeel



- μ = [0; 0]
- Σ = [1 0 ; 0 1]



- μ = [0; 0]
- Σ = [.6 0 ; 0 .6]



- μ = [0; 0]
- Σ = [2 0 ; 0 2]

Important Identities: Gaussians

Forward propagation

$$\begin{cases} X \sim \mathcal{N}(\mu, \Sigma) \\ Y = AX + B + \epsilon \\ \epsilon \sim \mathcal{N}(0, Q) \end{cases} \implies Y \sim \mathcal{N}(A\mu + B, A\Sigma A^T + Q)$$

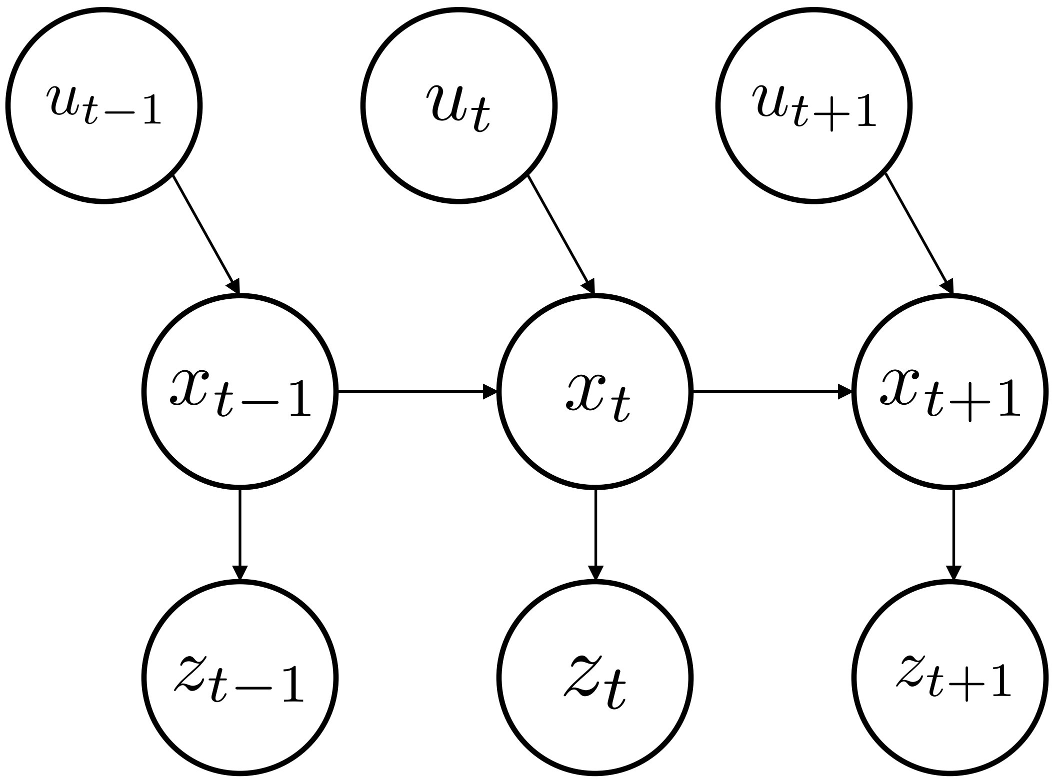
Conditioning

$$\begin{cases} X \sim \mathcal{N}(\mu, \Sigma) \\ Y = CX + B + \delta \\ \delta \sim \mathcal{N}(0, R) \end{cases} \implies X|Y = y_0 \sim \mathcal{N}(\mu + K(y_0 - C\mu), (I - KC)\Sigma)$$

- Marginalization and conditioning in Gaussians results in Gaussians
- We stay in the "Gaussian world" as long as we start with Gaussians and perform only linear transformations.

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

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

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

Linear Gaussian

with a measurement

$$z_t = c x_t + \delta_t$$

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

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***

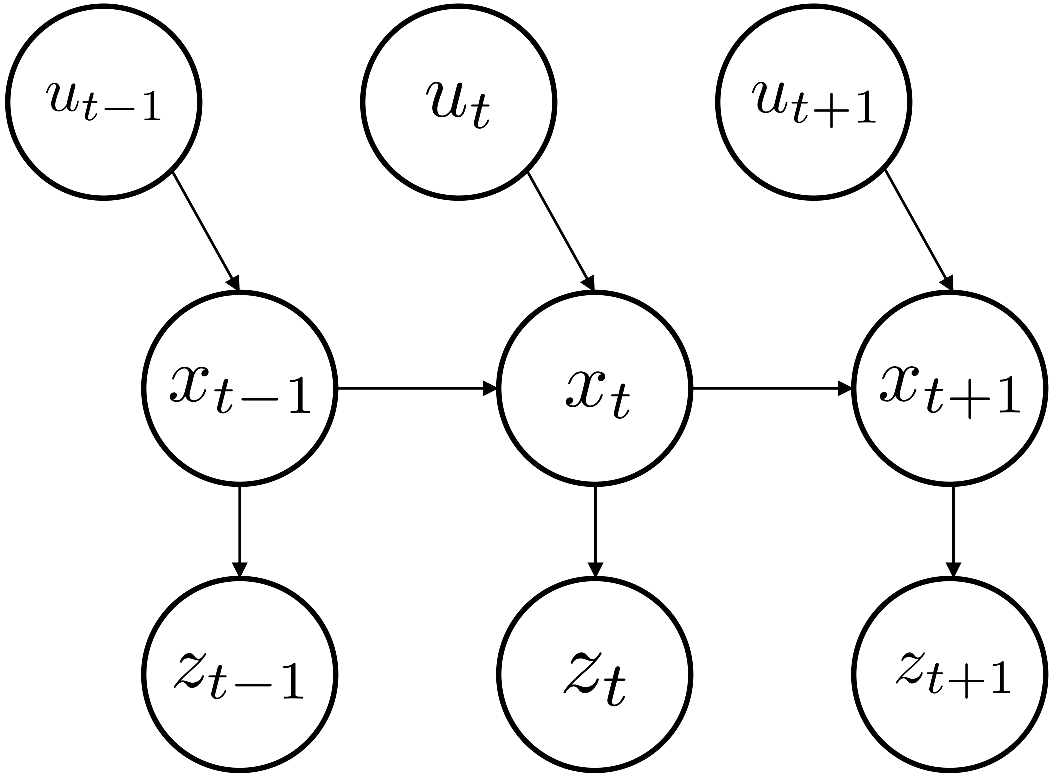
C Matrix (k x n) that describes how to map the state $\boldsymbol{x_t}$ to an observation $\boldsymbol{z_t}$.

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)

Bayes Filters

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

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

Linear Gaussian

Step 2: Correction - apply Bayes rule given measurement

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

Linear Gaussian Systems: Initialization

- Initial belief is normally distributed:

$$Bel(x_0) = \mathcal{N}(\mu_0, \Sigma_0)$$

- $Bel(x_t)$ at any step t is: $\mathcal{N}(\mu_{t|0:t}, \Sigma_{t|0:t})$
- $\overline{Bel}(x_t)$ at any step t is: $\mathcal{N}(\mu_{t|0:t-1}, \Sigma_{t|0:t-1})$

Linear Gaussian Systems: Prediction

- Integrate the effect of one action under the dynamics, before measurement comes in

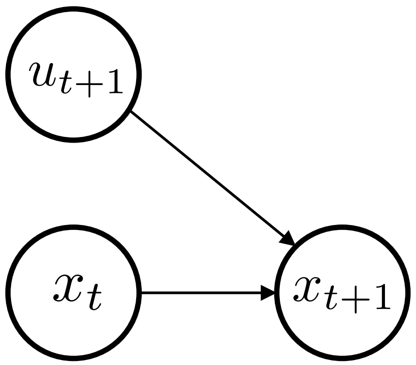
$$x_{t+1} = Ax_t + Bu_{t+1} + \epsilon_{t+1} \qquad \epsilon_{t+1} \sim \mathcal{N}(0, Q_{t+1})$$

$$p(x_{t+1}|x_t, u_{t+1}) = \mathcal{N}(Ax_t + Bu_{t+1}, Q_{t+1})$$

$\overline{Bel}(x_{t+1})$ $\qquad\qquad$ $Bel(x_t)$

$$p(x_{t+1}|z_{0:t}, u_{0:t+1}) = \int p(x_t|u_{0:t}, z_{0:t})p(x_{t+1}|u_{t+1}, x_t)dx_t$$

Gaussian, easy!



Linear Gaussian Systems: Prediction

- Integrate the effect of one action under the dynamics, before measurement comes in

$$x_{t+1} = Ax_t + Bu_{t+1} + \epsilon_{t+1} \qquad \epsilon_{t+1} \sim \mathcal{N}(0, Q_{t+1})$$

$$p(x_{t+1}|x_t, u_{t+1}) = \mathcal{N}(Ax_t + Bu_{t+1}, Q_{t+1})$$

$$\underbrace{p(x_{t+1}|z_{0:t}, u_{0:t+1})}_{\overline{Bel}(x_{t+1})} = \int \underbrace{p(x_t|u_{0:t}, z_{0:t})}_{Bel(x_t)} p(x_{t+1}|u_{t+1}, x_t) dx_t$$

Gaussian, easy!

$$\begin{cases} X \sim \mathcal{N}(\mu, \Sigma) \\ Y = AX + B + \epsilon \implies Y \sim \mathcal{N}(A\mu + B, A\Sigma A^T + Q) \\ \epsilon \sim \mathcal{N}(0, Q) \end{cases}$$

Linear Gaussian Systems: Prediction

- Integrate the effect of one action under the dynamics, before measurement comes in

$$p(x_t|u_{0:t}, z_{0:t}) = \mathcal{N}(\mu_{t|0:t}, \Sigma_{t|0:t})$$

$$x_{t+1} = Ax_t + Bu_{t+1} + \epsilon_{t+1}$$

$$\epsilon_{t+1} \sim \mathcal{N}(0, Q_{t+1})$$

$$\begin{cases} X \sim \mathcal{N}(\mu, \Sigma) \\ Y = AX + B + \epsilon \implies Y \sim \mathcal{N}(A\mu + B, A\Sigma A^T + Q) \\ \epsilon \sim \mathcal{N}(0, C) \end{cases}$$

Previous belief $p(x_t|u_{0:t}, z_{0:t}) = \mathcal{N}(\mu_{t|0:t}, \Sigma_{t|0:t})$

Belief Update $p(x_{t+1}|u_{0:t+1}, z_{0:t}) = \mathcal{N}(A\mu_{t|0:t} + Bu_{t+1}, A\Sigma_{t|0:t}A^T + Q_{t+1})$

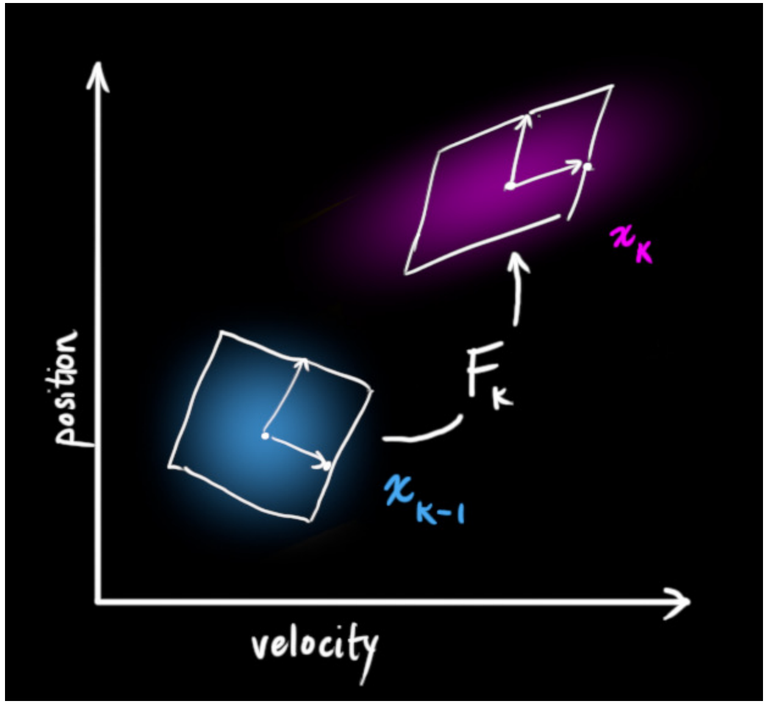
Intuition: Scale and shift the mean according to dynamics, uncertainty grows quadratically!

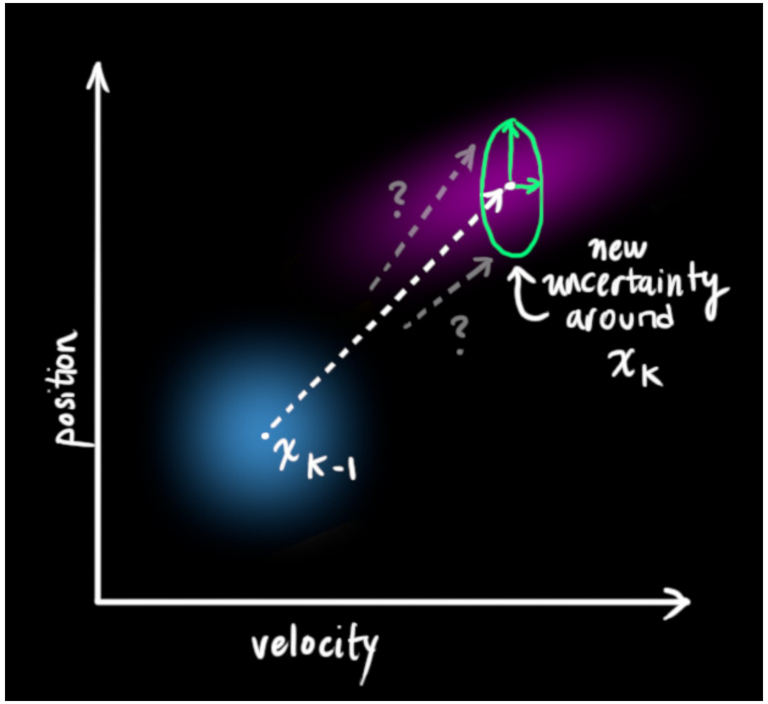
Linear Gaussian Systems: Prediction

Previous belief $$p(x_t|u_{0:t}, z_{0:t}) = \mathcal{N}(\mu_{t|0:t}, \Sigma_{t|0:t})$$

Belief Update $$p(x_{t+1}|u_{0:t+1}, z_{0:t}) = \mathcal{N}(A\mu_{t|0:t} + Bu_{t+1}, A\Sigma_{t|0:t}A^T + Q_{t+1})$$

Intuition: Scale and shift the mean according to dynamics, uncertainty grows!



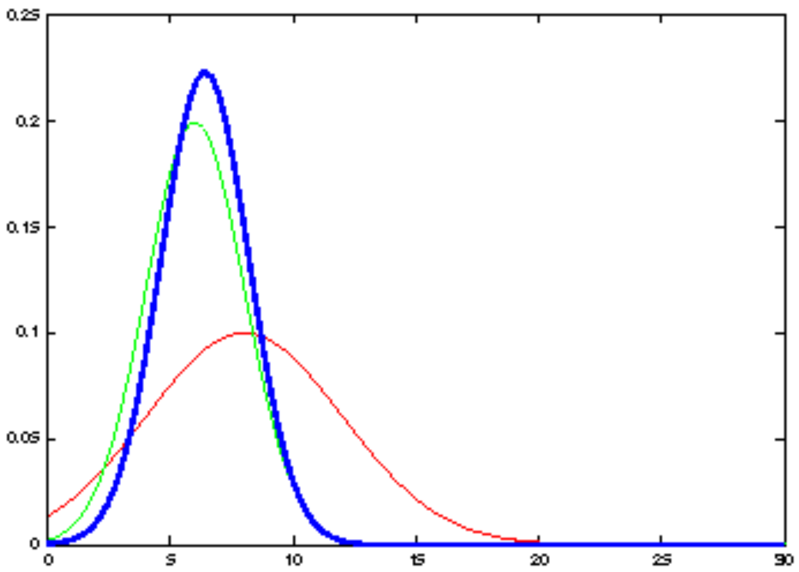


Intuition Behind Prediction Step

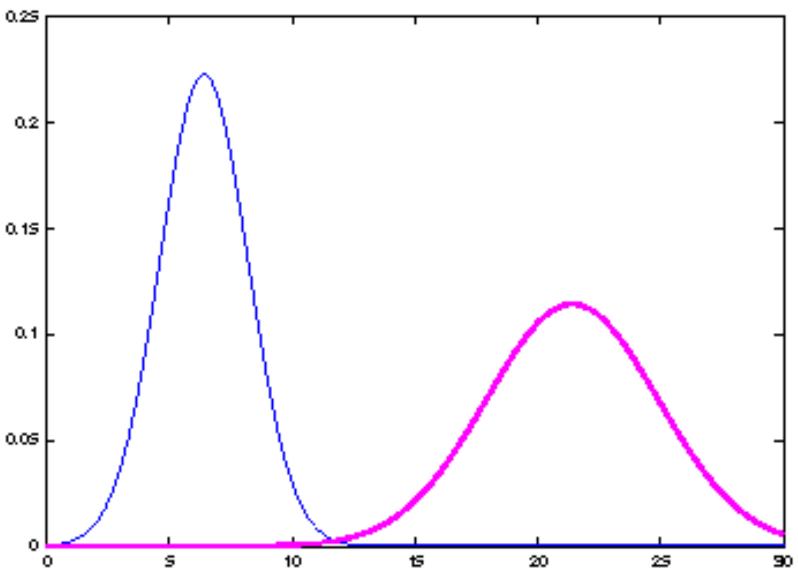
Previous belief $p(x_t | u_{0:t}, z_{0:t}) = \mathcal{N}(\mu_{t|0:t}, \Sigma_{t|0:t})$

Belief Update $p(x_{t+1} | u_{0:t+1}, z_{0:t}) = \mathcal{N}(A\mu_{t|0:t} + Bu_{t+1}, A\Sigma_{t|0:t}A^T + Q_{t+1})$

Intuition: Scale and shift the mean according to dynamics, uncertainty grows!



Belief at x_t



Belief post dynamics → shifted mean, scaled and shifted variance

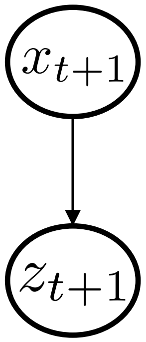
Linear Gaussian Systems: Observations

- Integrate the effect of an observation using sensor model, after dynamics

$$z_{t+1} = Cx_{t+1} + \delta_{t+1} \qquad \delta_{t+1} \sim \mathcal{N}(0, R_{t+1})$$

$$p(z_{t+1}|x_{t+1}) = \mathcal{N}(Cx_{t+1}, R_{t+1})$$

$$\underbrace{p(x_{t+1}|u_{0:t+1}, z_{0:t+1})}_{Bel(x_{t+1})} \propto p(z_{t+1}|x_{t+1}) \underbrace{p(x_{t+1}|u_{0:t+1}, z_{0:t})}_{\overline{Bel}(x_{t+1})}$$



Gaussian, easy to normalize

Slightly harder than the dynamics step!

Linear Gaussian Systems: Observations

- Integrate the effect of an observation using sensor model, after dynamics

$$z_{t+1} = Cx_{t+1} + \delta_{t+1} \qquad \delta_{t+1} \sim \mathcal{N}(0, R_{t+1})$$

$$p(z_{t+1}|x_{t+1}) = \mathcal{N}(Cx_{t+1}, R_{t+1})$$

$$\underbrace{p(x_{t+1}|u_{0:t+1}, z_{0:t+1})}_{Bel(x_{t+1})} \propto p(z_{t+1}|x_{t+1})\underbrace{p(x_{t+1}|u_{0:t+1}, z_{0:t})}_{\overline{Bel}(x_{t+1})}$$

Conditioning

$$\begin{cases} X \sim \mathcal{N}(\mu, \Sigma) \\ Y = CX + B + \delta \\ \delta \sim \mathcal{N}(0, R) \end{cases} \implies X|Y = y_0 \sim \mathcal{N}(\mu + K(y_0 - C\mu), (I - KC)\Sigma)$$

$$K = \Sigma C^T (C \Sigma C^T + R)^{-1}$$

Linear Gaussian Systems: Observations

- Integrate the effect of an observation using sensor model, after dynamics

$$p(x_{t+1}|u_{0:t+1}, z_{0:t}) = \mathcal{N}(\mu_{t+1|0:t}, \Sigma_{t+1|0:t})$$
$$z_{t+1} = Cx_{t+1} + \delta_{t+1}$$
$$\delta_{t+1} \sim \mathcal{N}(0, R_{t+1})$$

$$\begin{cases} X \sim \mathcal{N}(\mu, \Sigma) \\ Y = CX + B + \delta \\ \delta \sim \mathcal{N}(0, R) \end{cases} \implies \begin{array}{c} X|Y = y_0 \sim \mathcal{N}(\mu + K(y_0 - C\mu), (I - KC)\Sigma) \\ K = \Sigma C^T (C\Sigma C^T + R)^{-1} \end{array}$$

Previous belief $p(x_{t+1}|u_{0:t+1}, z_{0:t}) = \mathcal{N}(\mu_{t+1|0:t}, \Sigma_{t+1|0:t})$ Computed from dynamics step

Updated belief $p(x_{t+1}|u_{0:t+1}, z_{0:t+1})$

$$= \mathcal{N}(\mu_{t+1|0:t} + K_{t+1}(z_{t+1} - C\mu_{t+1|0:t}), (I - K_{t+1}C)\Sigma_{t+1|0:t})$$

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

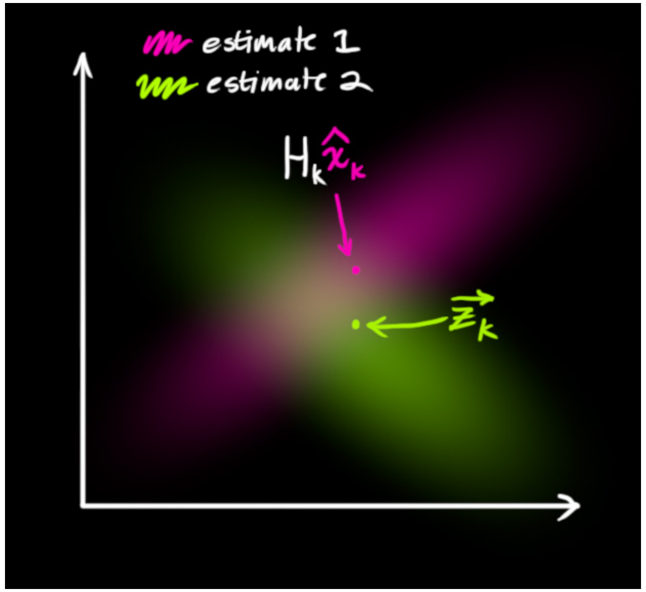
Linear Gaussian Systems: Observations

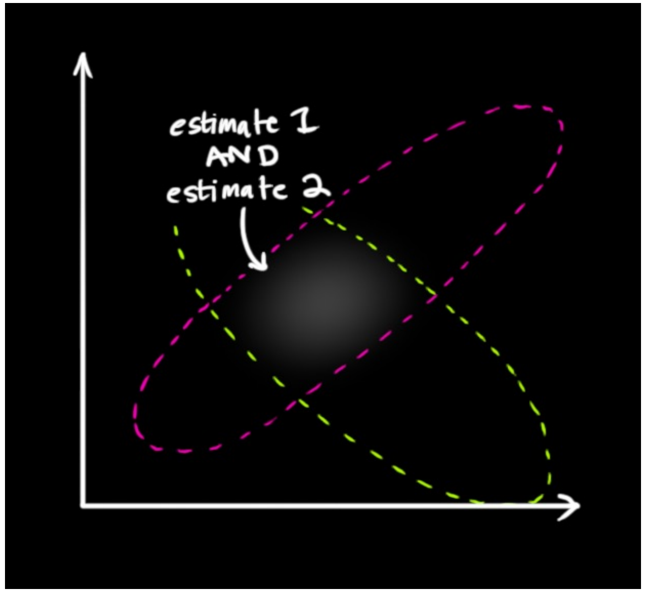
Previous belief $p(x_{t+1}|u_{0:t+1}, z_{0:t}) = \mathcal{N}(\mu_{t+1|0:t}, \Sigma_{t+1|0:t})$ Computed from dynamics step

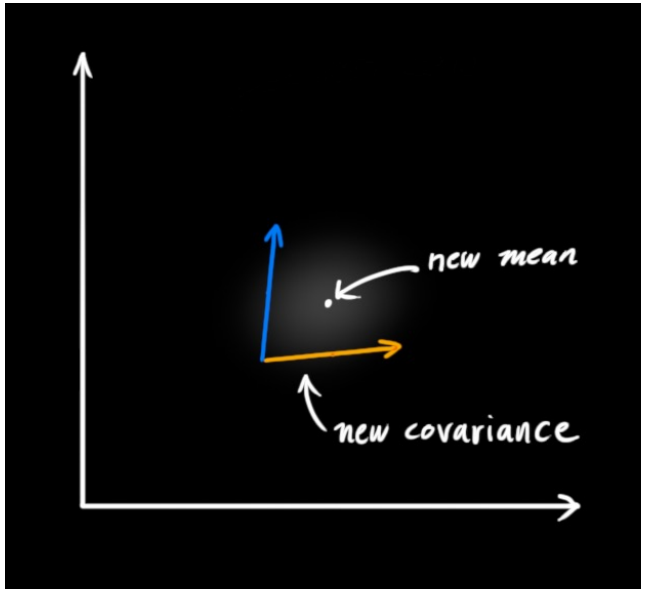
Updated belief $p(x_{t+1}|u_{0:t+1}, z_{0:t+1})$

$$= \mathcal{N}(\mu_{t+1|0:t} + K_{t+1}(z_{t+1} - C\mu_{t+1|0:t}), (I - K_{t+1}C)\Sigma_{t+1|0:t})$$

Intuition: Correct the update linearly according to measurement error from expectation, shrink uncertainty accordingly



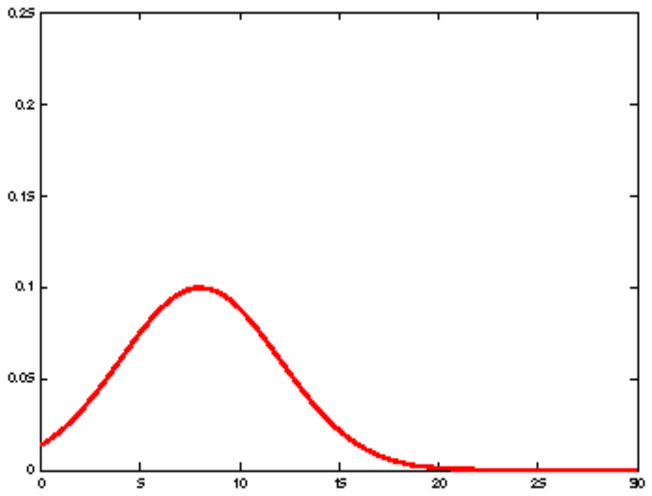


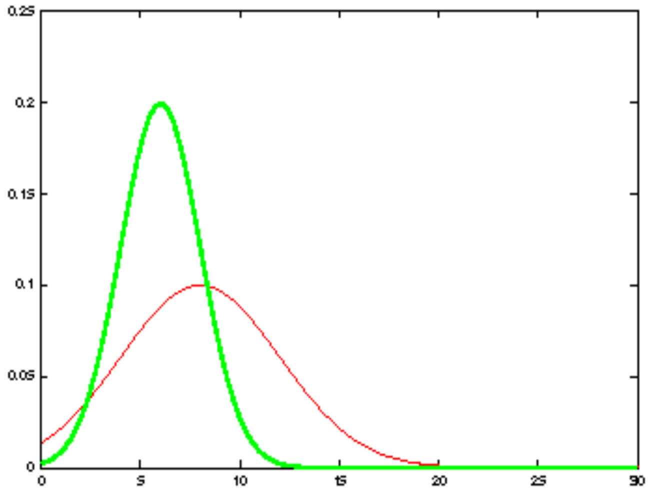


Intuition Behind Correction Step

Previous belief

New Measurement





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

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

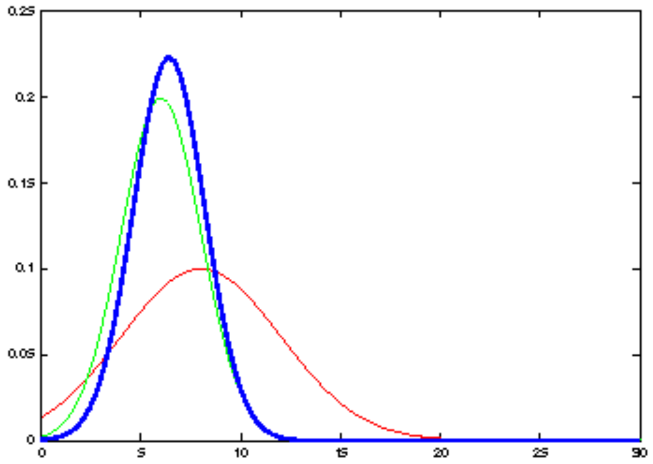
For the sake of simplicity, let's say C = I

$$K_{t+1} = \frac{\Sigma_{t+1|0:t}}{\Sigma_{t+1|0:t} + R}$$

Corrects belief based on measurement

→ Average between mean and measurement based on K

→ Scale down uncertainty based on K



Unpacking the Kalman Gain

Previous belief $p(x_{t+1}|u_{0:t+1}, z_{0:t}) = \mathcal{N}(\mu_{t+1|0:t}, \Sigma_{t+1|0:t})$ Computed from dynamics step

Updated belief $p(x_{t+1}|u_{0:t+1}, z_{0:t+1})$

$$= \mathcal{N}(\mu_{t+1|0:t} + K_{t+1}(z_{t+1} - C\mu_{t+1|0:t}), (I - K_{t+1}C)\Sigma_{t+1|0:t})$$

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

Case 1: Very noisy sensor, R>>Σ

For the sake of simplicity, let's say C = I

$$K_{t+1} = \frac{\Sigma_{t+1|0:t}}{\Sigma_{t+1|0:t} + R}$$

Case 2: Deterministic sensor, R = 0

Kalman Filter Algorithm

Initial Prior

$p(x_0)$

Dynamics/Prediction
(given some u)

Estimate $\overline{Bel}(x_{t+1})$

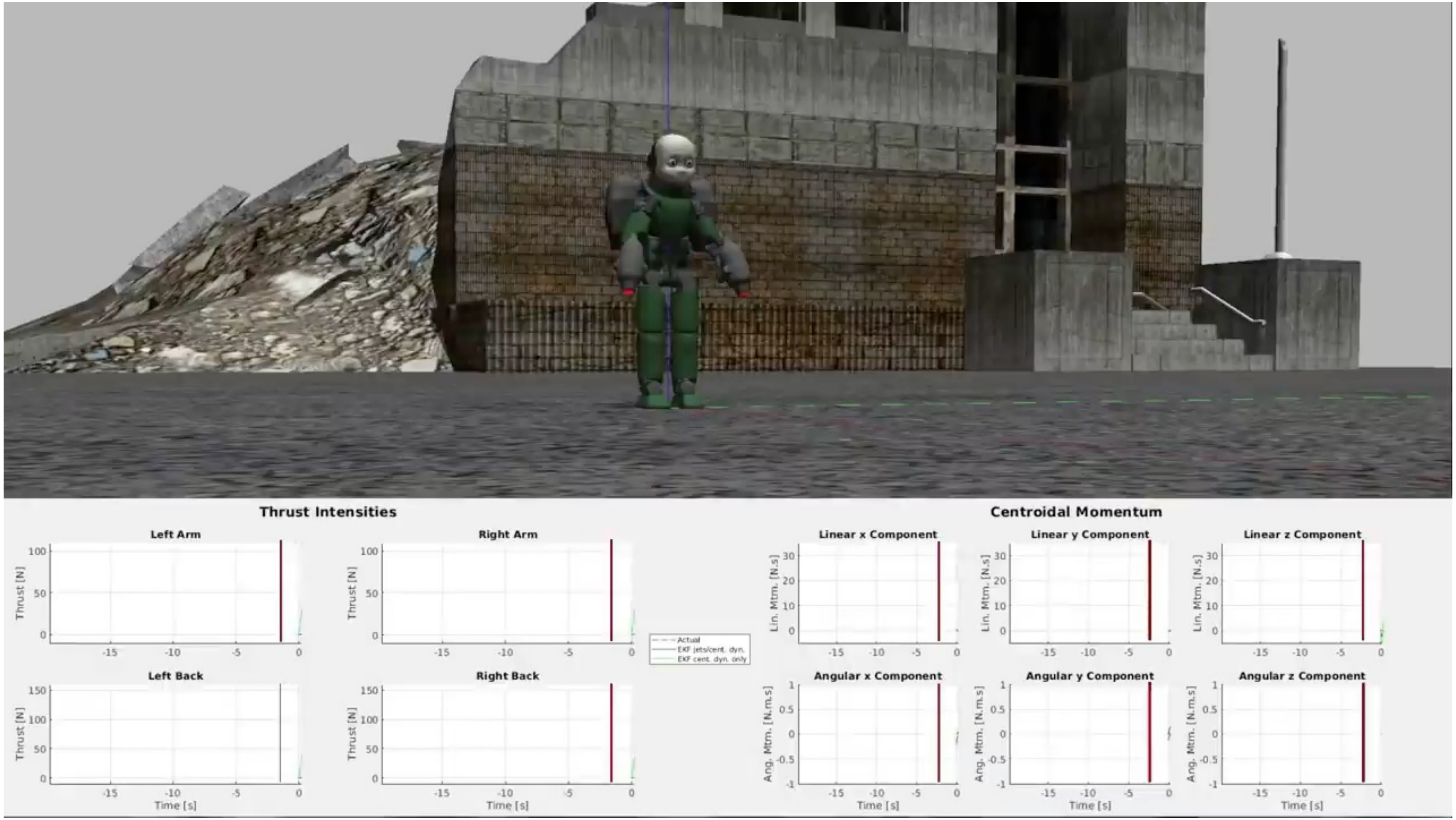
$$p(x_{t+1}|u_{0:t+1}, z_{0:t}) = \mathcal{N}(A\mu_{t|0:t} + Bu_{t+1}, A\Sigma_{t|0:t}A^T + Q_{t+1})$$

Measurement/Correction
(given some z)

Estimate $Bel(x_{t+1})$

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

Kalman Filter in Action



Kalman Filter Summary

- Highly efficient: Polynomial in measurement dimensionality ***k*** and state dimensionality ***n***:

  $O(k^{2.376} + n^2)$

Matrix Inversion (Correction)

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

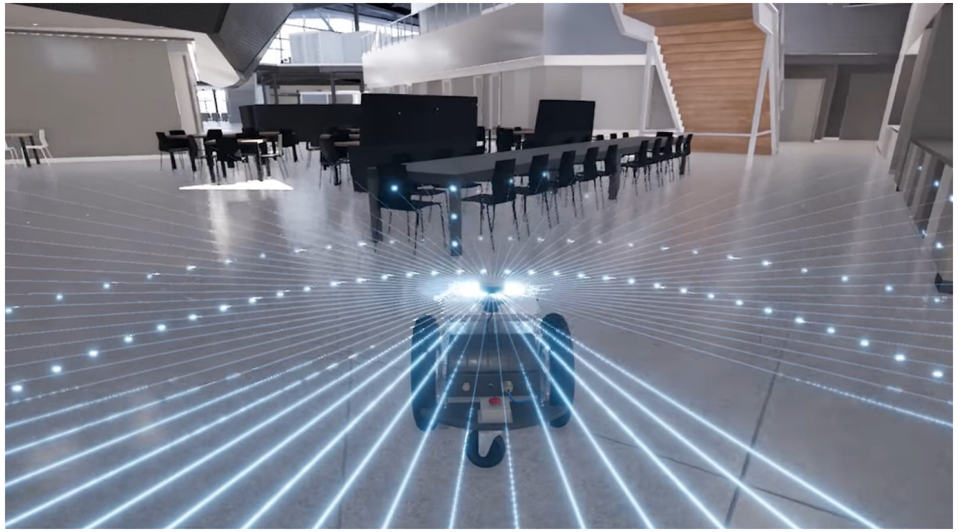
Matrix Multiplication (Prediction)

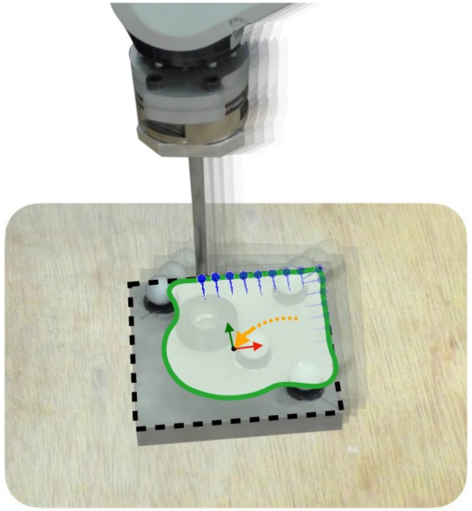
$$p(x_{t+1} | z_{0:t}, u_{0:t+1}) \sim \mathcal{N}(A \mu_{t|0:t} + B u_t, A \Sigma_{t|0:t} A^T + Q_t)$$

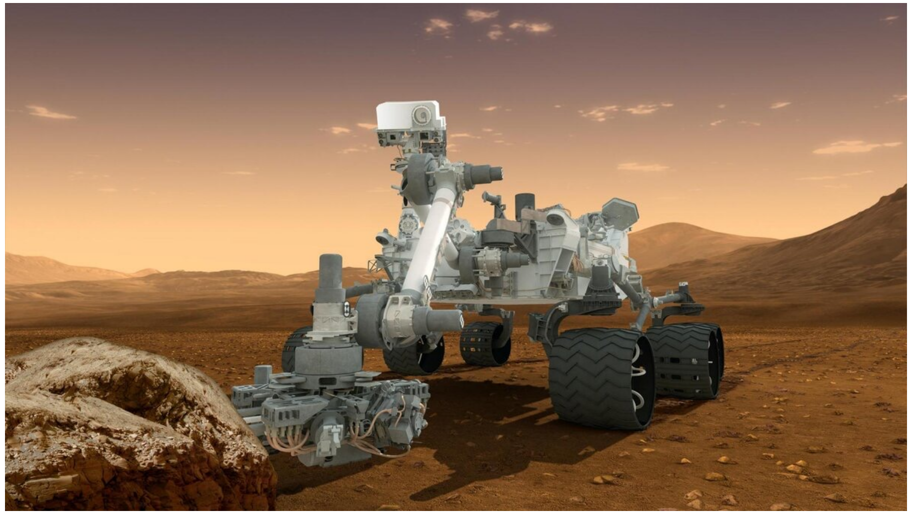
- Optimal for linear Gaussian systems!
- Most robotics systems are nonlinear!

Why should we care?

Still a very widely used technique for estimation/localization/mapping in real problems







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