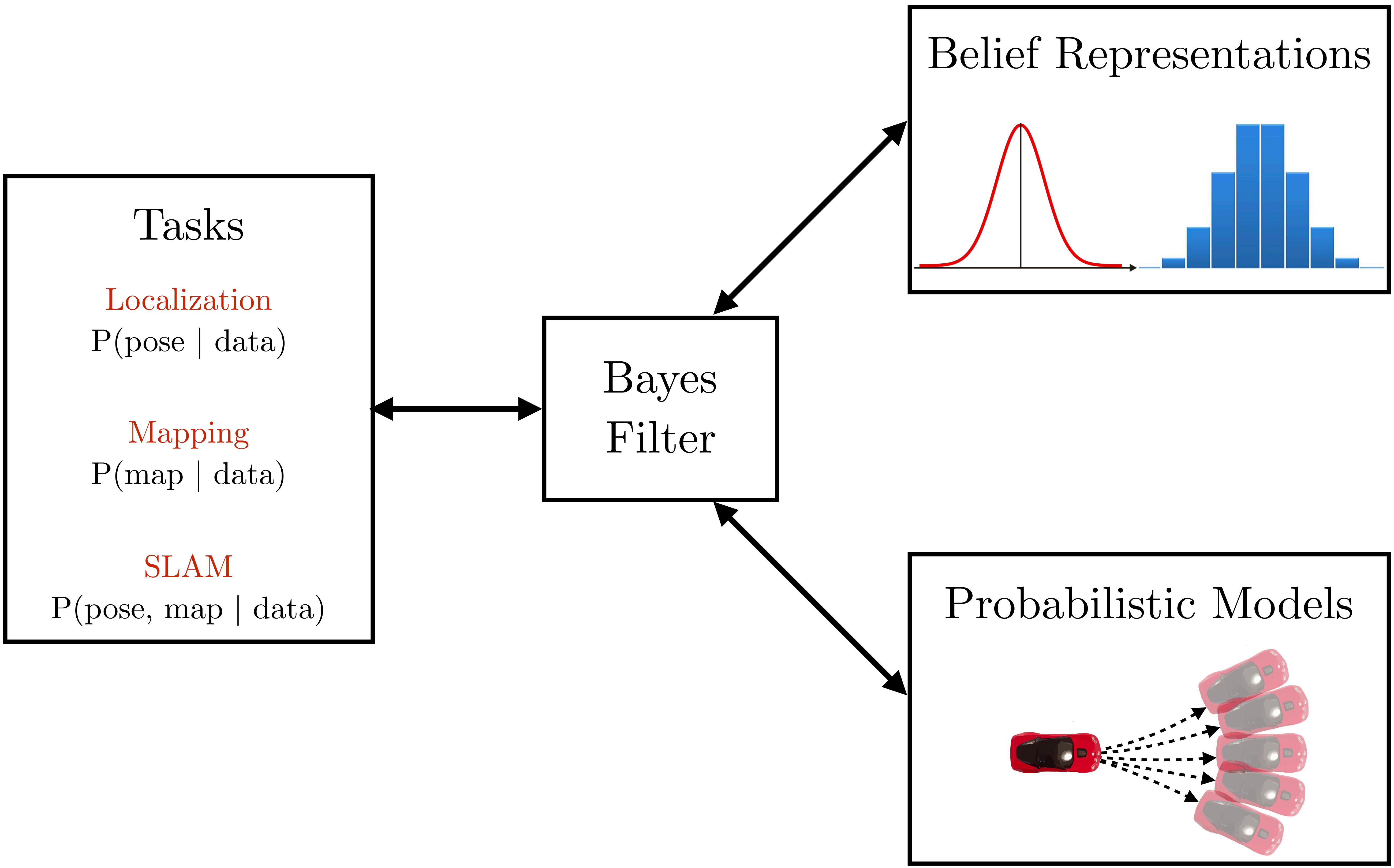
Particle Filters

Instructor: Chris Mavrogiannis

TAs: Kay Ke, Gilwoo Lee, Matt Schmittle

*Slides based on or adapted from Sanjiban Choudhury and Dieter Fox

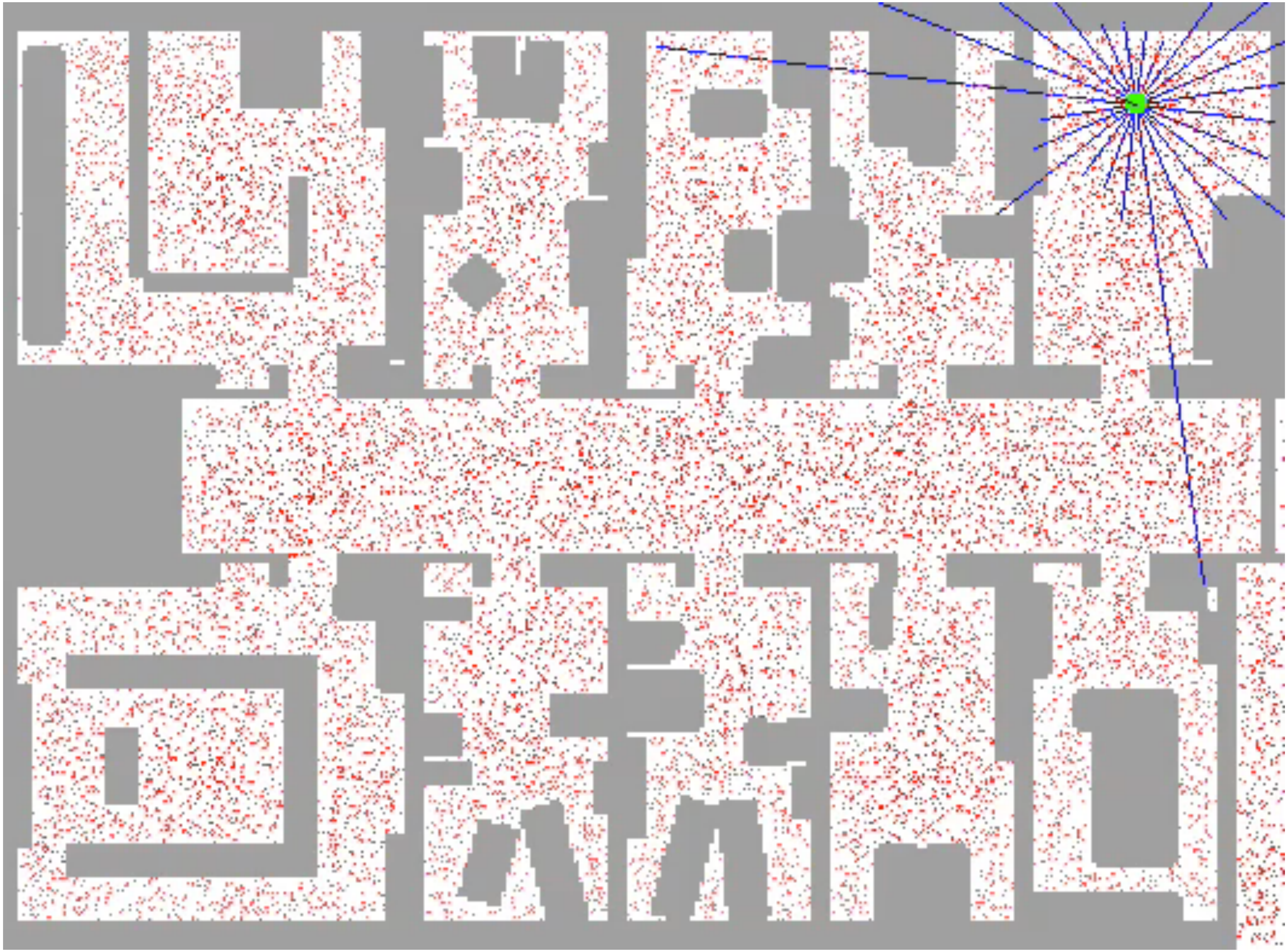
Assembling Bayes filter



Tasks that we will cover

Tasks	Belief Representation	Probabilistic Models
Localization P(pose | data) (Week 3)	Gaussian / Particles	Motion model Measurement model
Mapping P(map | data) (Week 4)	Discrete (binary)	Inverse measurement model
SLAM P(pose, map | data) (Week 4)	Particles+Gaussian (pose, landmarks)	Motion model, measurement model, correspondence model

Example: Indoor localization



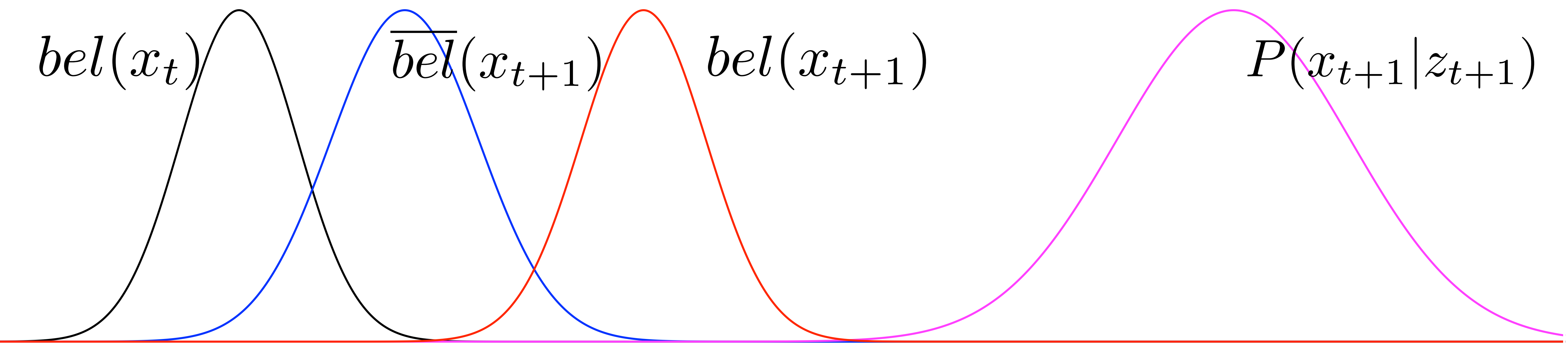
Today's objective

1. Understand the need for non-parametric filtering when faced with complex pdf in continuous space.

2. Importance sampling as an effective tool for dealing with complex pdf

Why can't we just use parametric filters?

Everything is a Gaussian - prior, motion, observation, posterior!



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

(Gaussian) (Gaussian) (Gaussian) (Gaussian)

Good things about parametric filters

We have so far been thinking about parametric filter (Kalman)

1. They are exact (when correct model)

   E.g. Kalman Filter

2. They are efficient to compute

   E.g. Sparse matrix inversion

Problems with parametric filters

1. Posterior has to have a fixed functional form (e.g. Gaussian)

   - even if our prior was a Gaussian, if control/measurement model is non-linear, posterior is NOT a Gaussian

2. We can always approximate with parametric belief (e.g. EKF)

   - what if true posterior was multi-modal? danger of losing a mode completely

How can we realize Bayes filters in a non-parametric fashion?

Tracking a landing pad with laser only

(Arora et al.)

Question: What are our options for non-parametric belief representations?

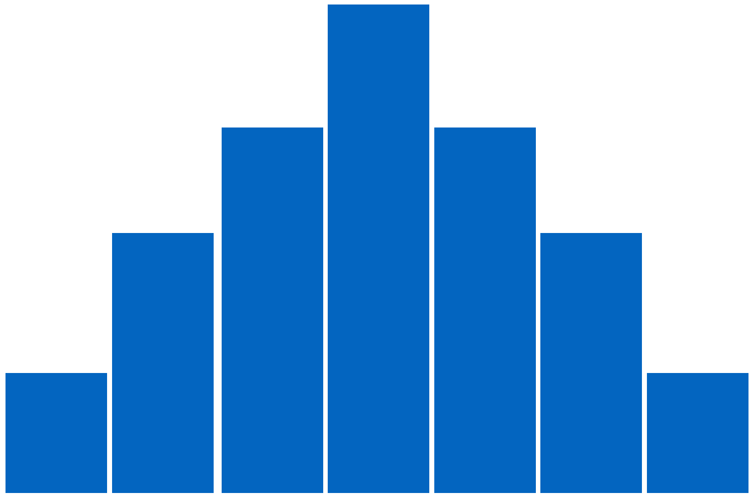
1. Histogram filter

2. Normalized importance sampling

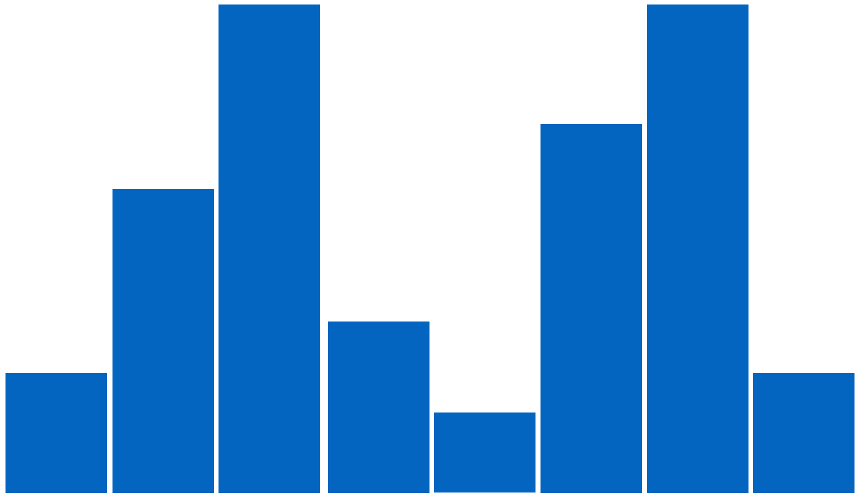
3. Particle filter

Approach 1: Histogram filter

Simplest approach - discretize the space!

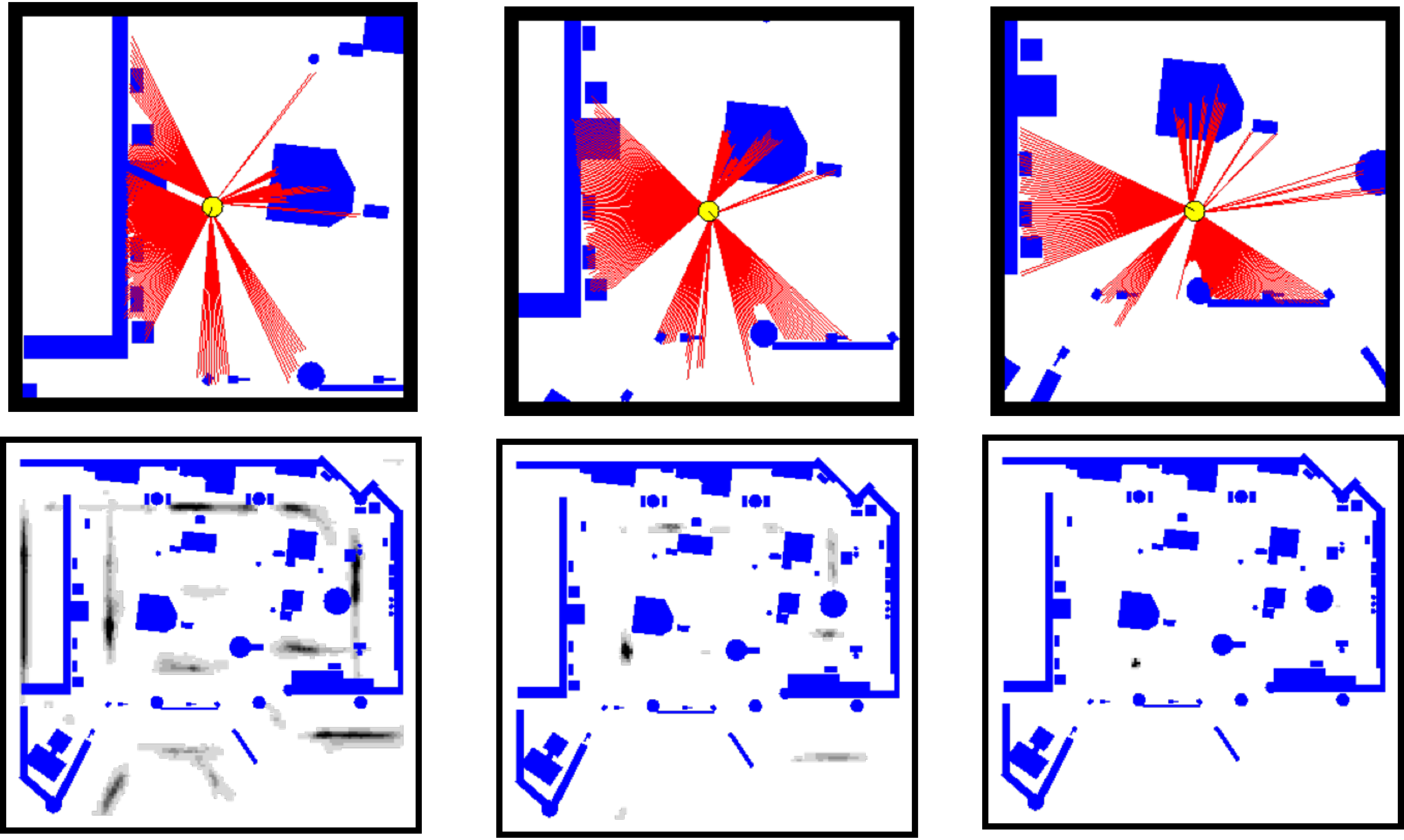


Prior $bel(x_t)$



Posterior $bel(x_{t+1})$

Example: Grid-based localization



Issues with grid-based localization

1. Curse of dimensionality

   Remedy: Adaptive discretization

2. Wasted computational effort

   Remedy: Pre-cache measurements from cell centers

3. Wasted memory resources

   Remedy: Update a select number of cells only

If discretization is expensive,
can we sample?

Monte-Carlo method

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

Ans: 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:	$f(x) \equiv V(x)$

Monte-Carlo method

Problem: Can't evaluate the integral below since we don't know bel

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

Solution: Sample from the distribution $x_1, \ldots, x_N \sim bel(x)$

Monte Carlo Estimate $\longrightarrow$ $\mathbb{E}[f] \approx \frac{1}{N}\sum_{i}^{N} f(x_i)$

(originated in Los Alamos)

$+$ Incremental, any-time.

$+$ Converges to the true expectation under a mild set of assumptions

Lots of general applications!

Can we always sample?

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

How can we sample from the product of two distributions?

Question:

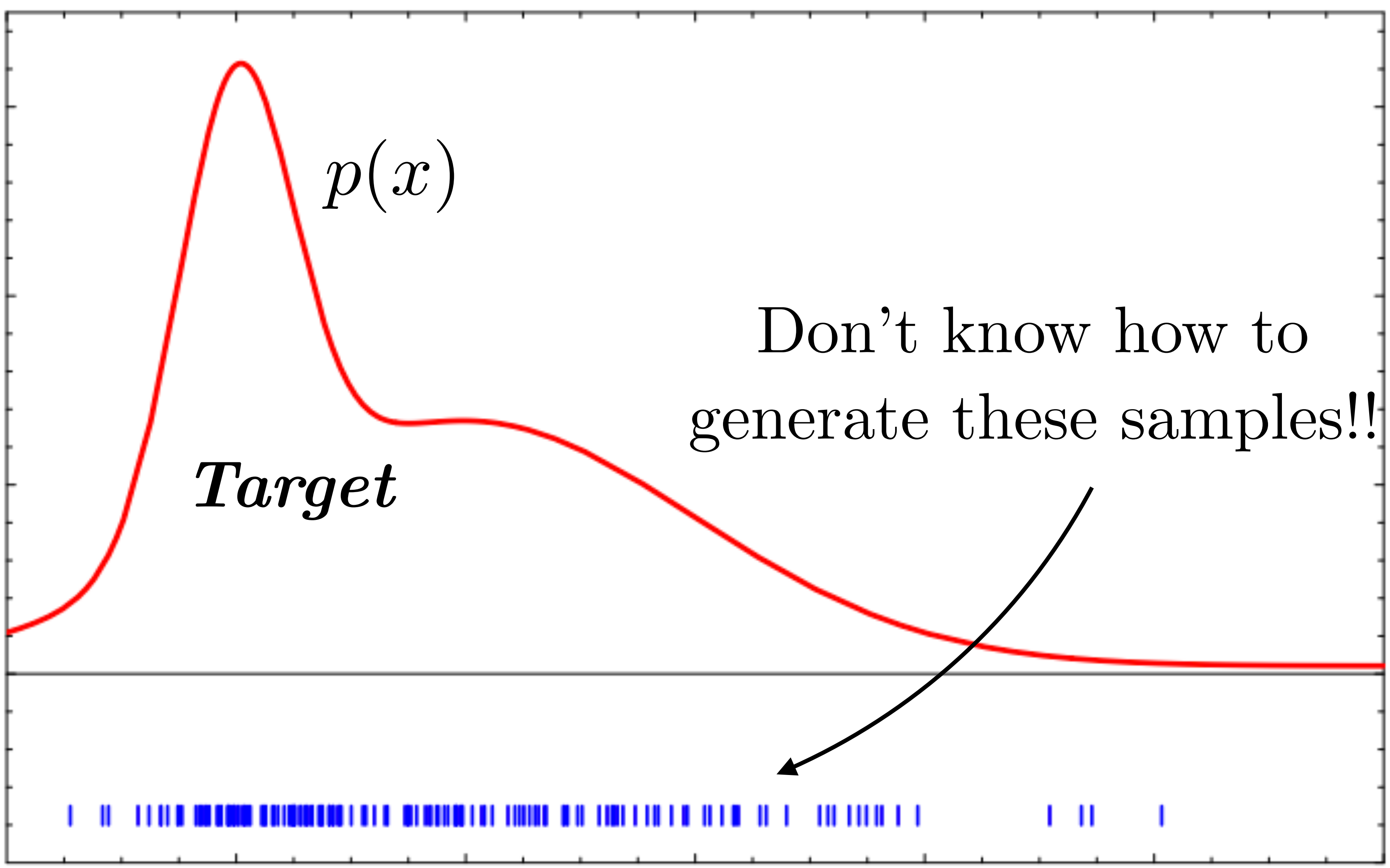
How can we sample from a complex distribution p(x)?

Solution: Importance sampling

Trick:

1. Sample from a proposal distribution (easy),
2. Reweigh samples to fix it!

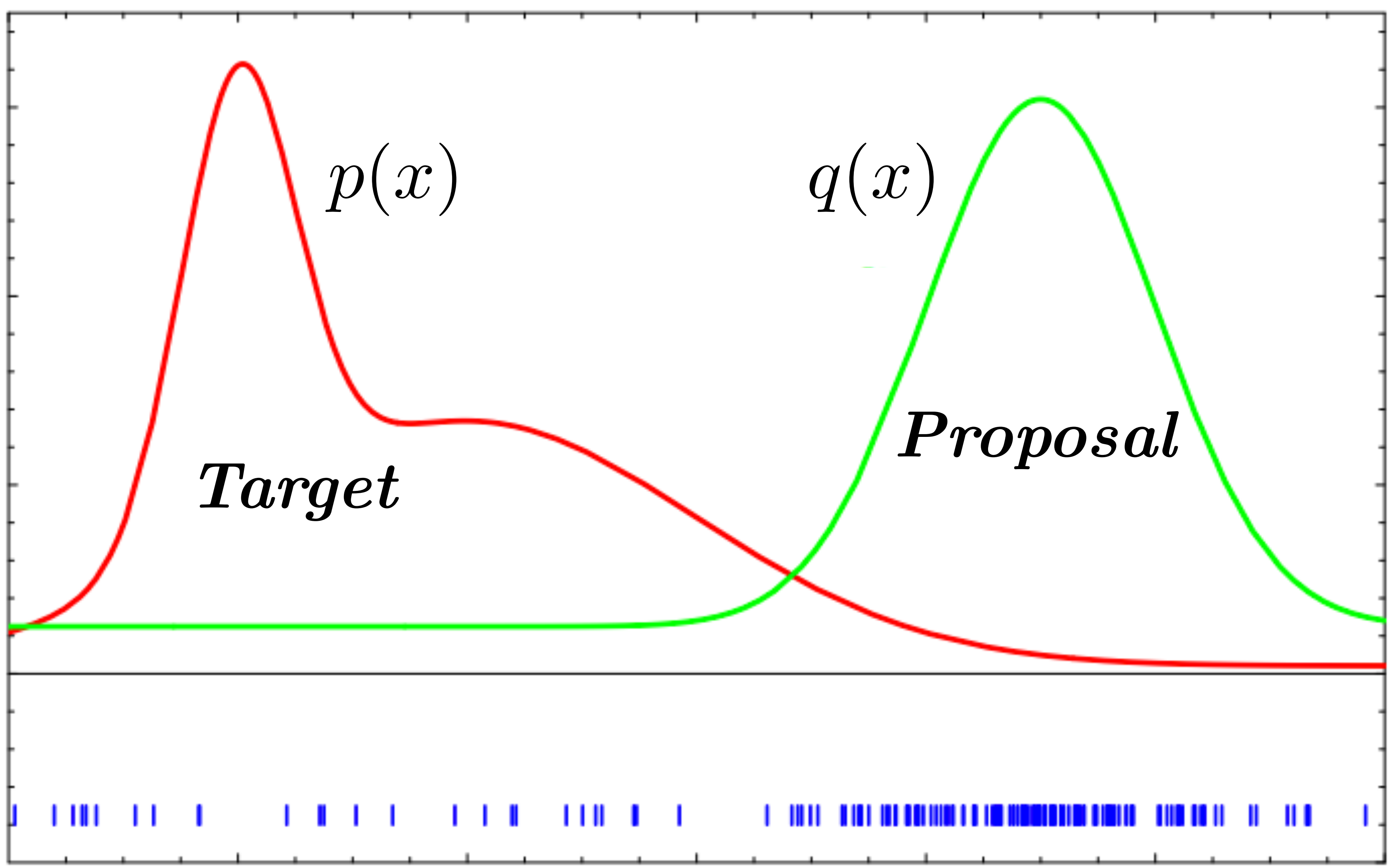
Solution: Importance sampling



Solution: Importance sampling

Trick:

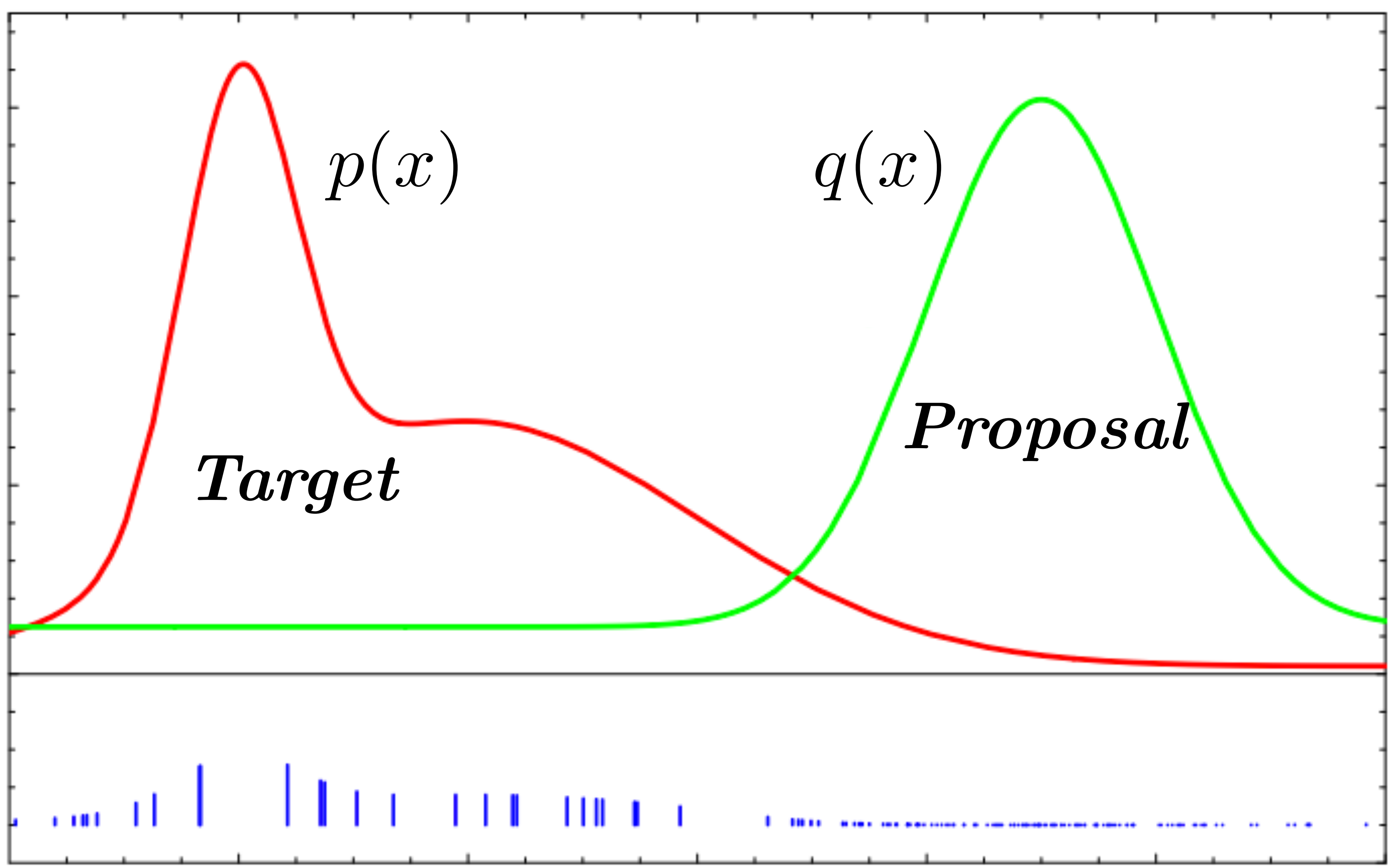
1. Sample from a proposal distribution (easy),



Solution: Importance sampling

Trick:

1. Sample from a proposal distribution (easy),
2. Reweigh samples to fix it!



Solution: Importance sampling

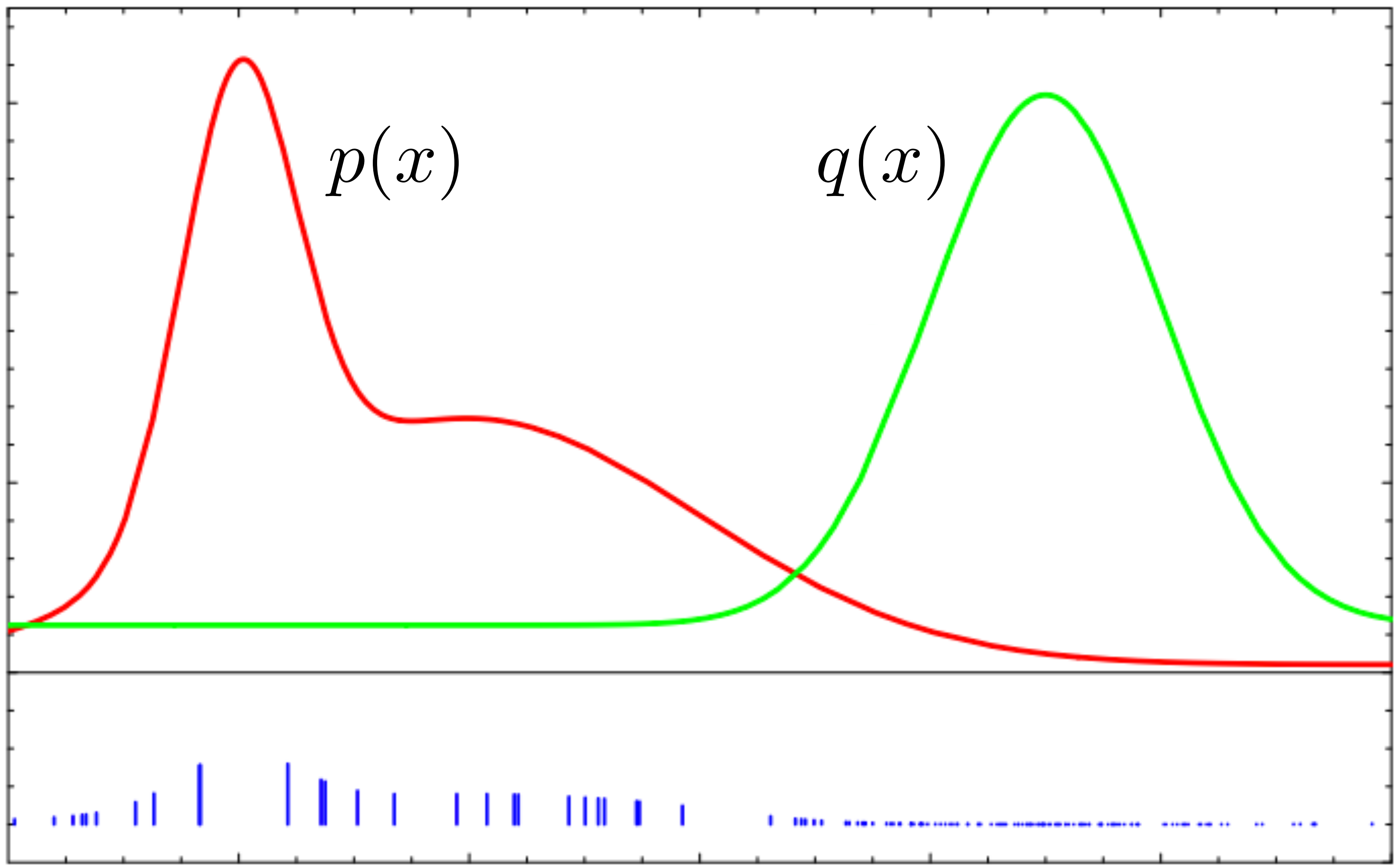
Trick: Sample from a proposal distribution (easy), reweigh samples to fix it!

$$
\begin{aligned}
\mathop{\mathrm{E}}_{p(x)}[f(x)] &= \sum p(x) f(x) \\
&= \sum p(x) f(x) \frac{q(x)}{q(x)} \\
&= \sum q(x) \frac{p(x)}{q(x)} f(x) \\
&= \mathop{\mathrm{E}}_{q(x)}[\frac{p(x)}{q(x)} f(x)] \\
&\approx \frac{1}{N} \sum_{i=1}^{N} \boxed{\frac{p(x_i)}{q(x_i)}} f(x_i).
\end{aligned}
$$

Importance Weight

Convergence precondition: $p(x) > 0$ whenever $q(x) > 0$

Question: What makes a good proposal distribution?



Applying importance sampling to Bayes filtering

Target distribution : Posterior

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

Proposal distribution : After applying motion model

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

Importance ratio:

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

Question: What are our options for non-parametric belief representations?

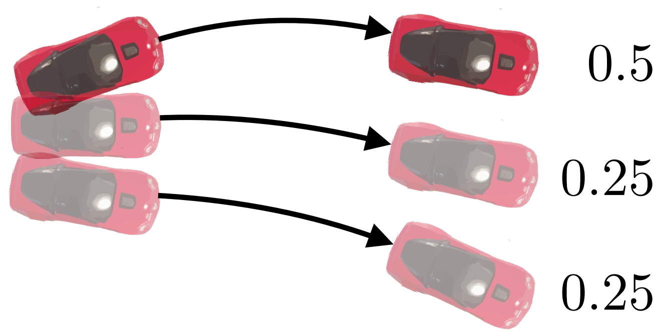
1. Histogram filter

2. Normalized importance sampling

3. Particle filter

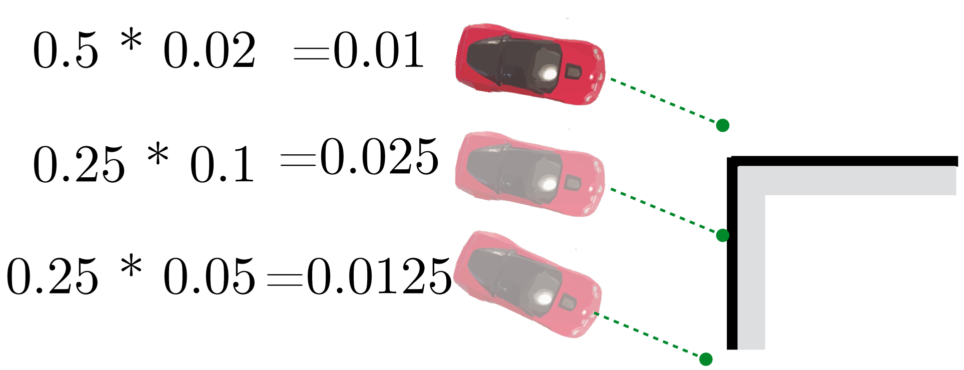
Approach 2: Normalized Importance Sampling

$$bel(x_{t-1}) = \left\{ \begin{matrix} x_{t-1}^1 \\ w_{t-1}^1 \end{matrix}, \begin{matrix} x_{t-1}^2 \\ w_{t-1}^2 \end{matrix}, \cdots, \begin{matrix} x_{t-1}^M \\ w_{t-1}^M \end{matrix} \right\}$$



for $i = 1$ to M

$$\text{sample } \overline{x}_t^i \sim P(x_t | u_t, x_t^i)$$



for $i = 1$ to M

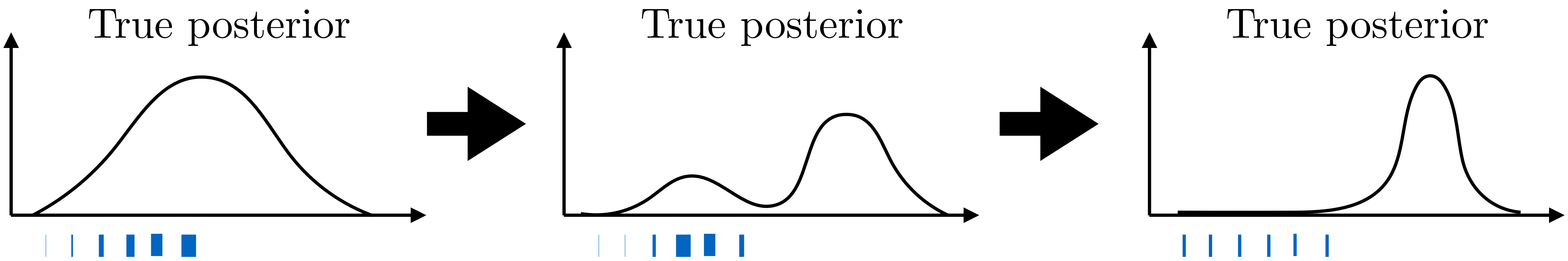
$$w_t^i = P(z_t | \bar{x}_t^i) w_{t-1}^i$$

for $i = 1$ to M

$$w_t^i = \frac{w_t^i}{\sum_i w_t^i} \qquad bel(x_t) = \left\{ \begin{matrix} \bar{x}_t^1 \\ w_t^1 \end{matrix}, \cdots, \begin{matrix} \bar{x}_t^M \\ w_t^M \end{matrix} \right\}$$

Problem: What happens after enough iterations?

Particles don't move - can get stuck in regions of low probability



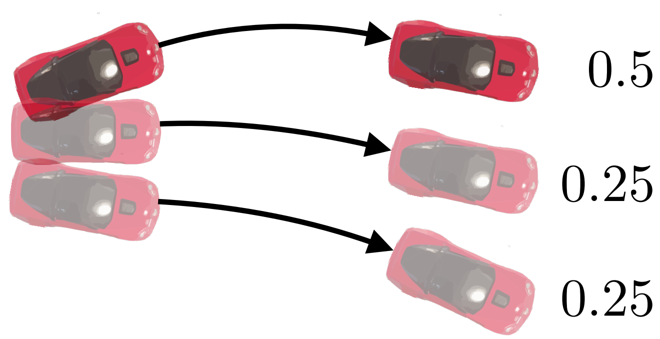
This is a problem of histogram filters too...

Key Idea: Resample!

Why? Get rid of bad particles

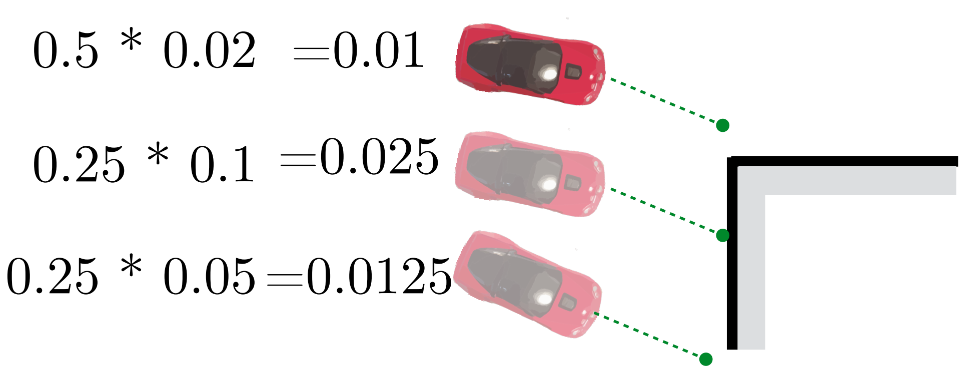
Approach 3: Particle Filtering (with IS)

$$bel(x_{t-1}) = \left\{ \begin{matrix} x^1_{t-1} \\ w^1_{t-1} \end{matrix}, \begin{matrix} x^2_{t-1} \\ w^2_{t-1} \end{matrix}, \cdots, \begin{matrix} x^M_{t-1} \\ w^M_{t-1} \end{matrix} \right\}$$



for $i = 1$ to M

$$\text{sample } \bar{x}^i_t \sim P(x_t | u_t, x^i_t)$$



for $i = 1$ to M

$$w^i_t = P(z_t | \bar{x}^i_t) w^i_{t-1}$$

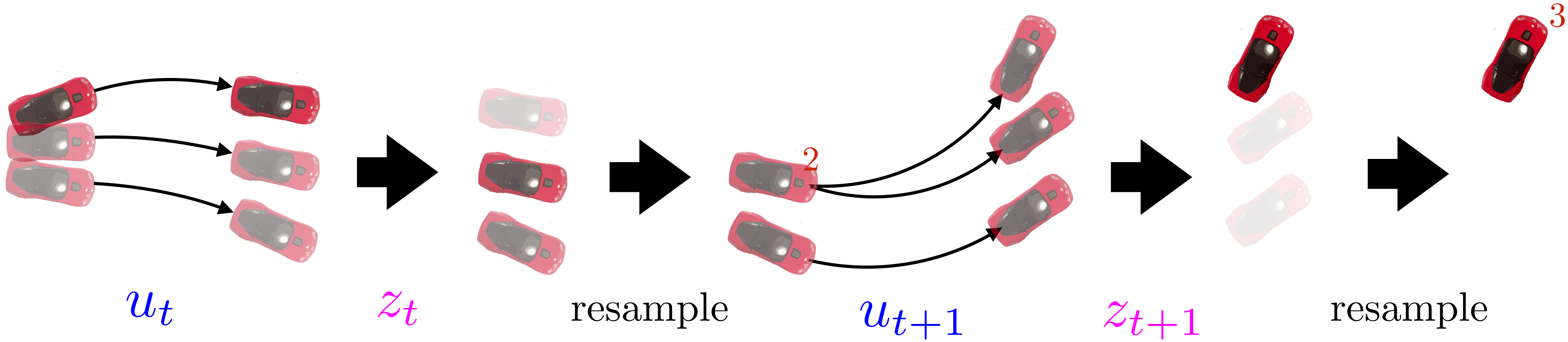
all weights = 1/M

for $i = 1$ to M

$$\text{sample } x^i_t \sim w^i_t$$

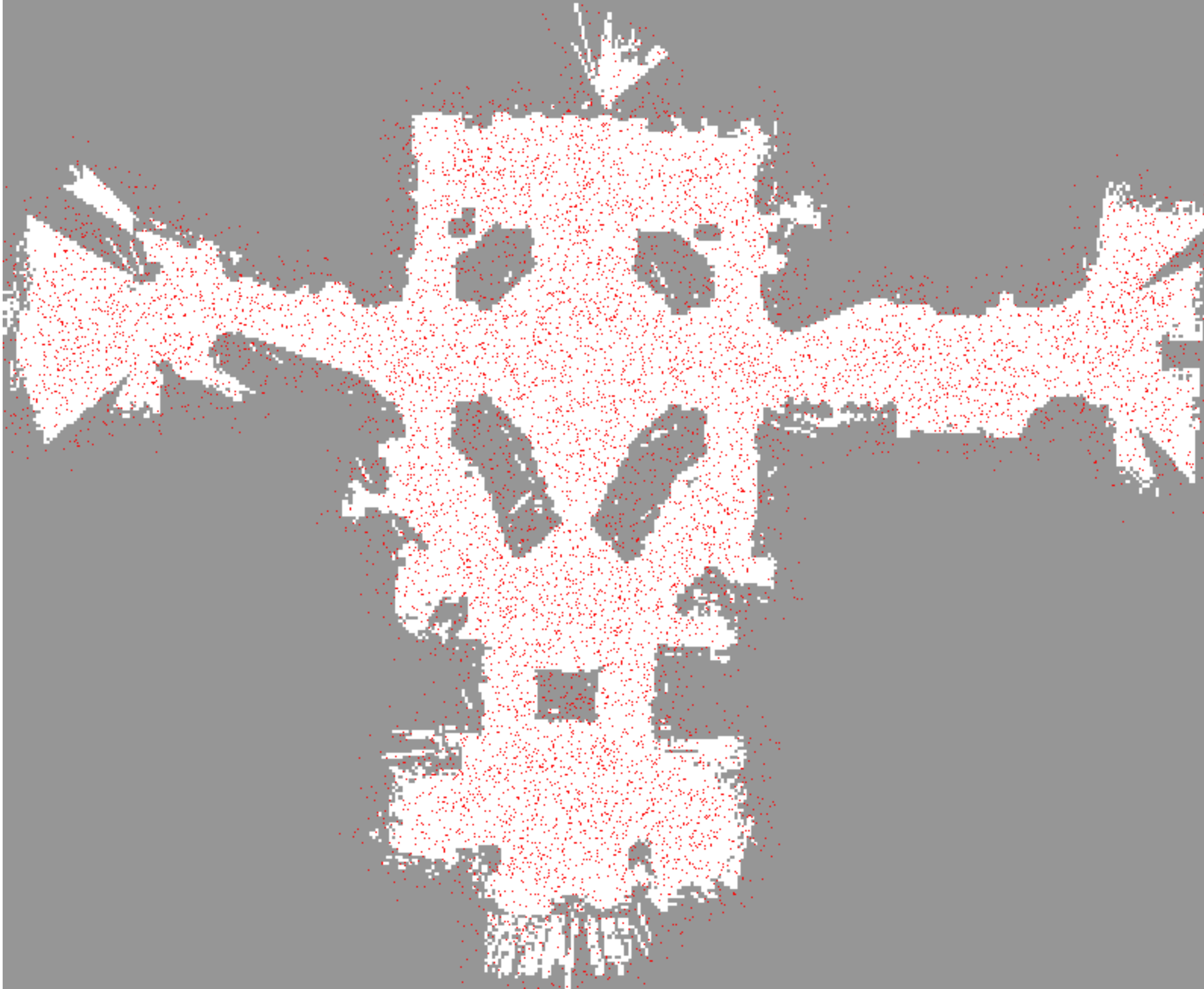
$$bel(x_t) = \left\{ \begin{matrix} x^1_t \\ 1 \end{matrix}, \cdots, \begin{matrix} x^M_t \\ 1 \end{matrix} \right\}$$

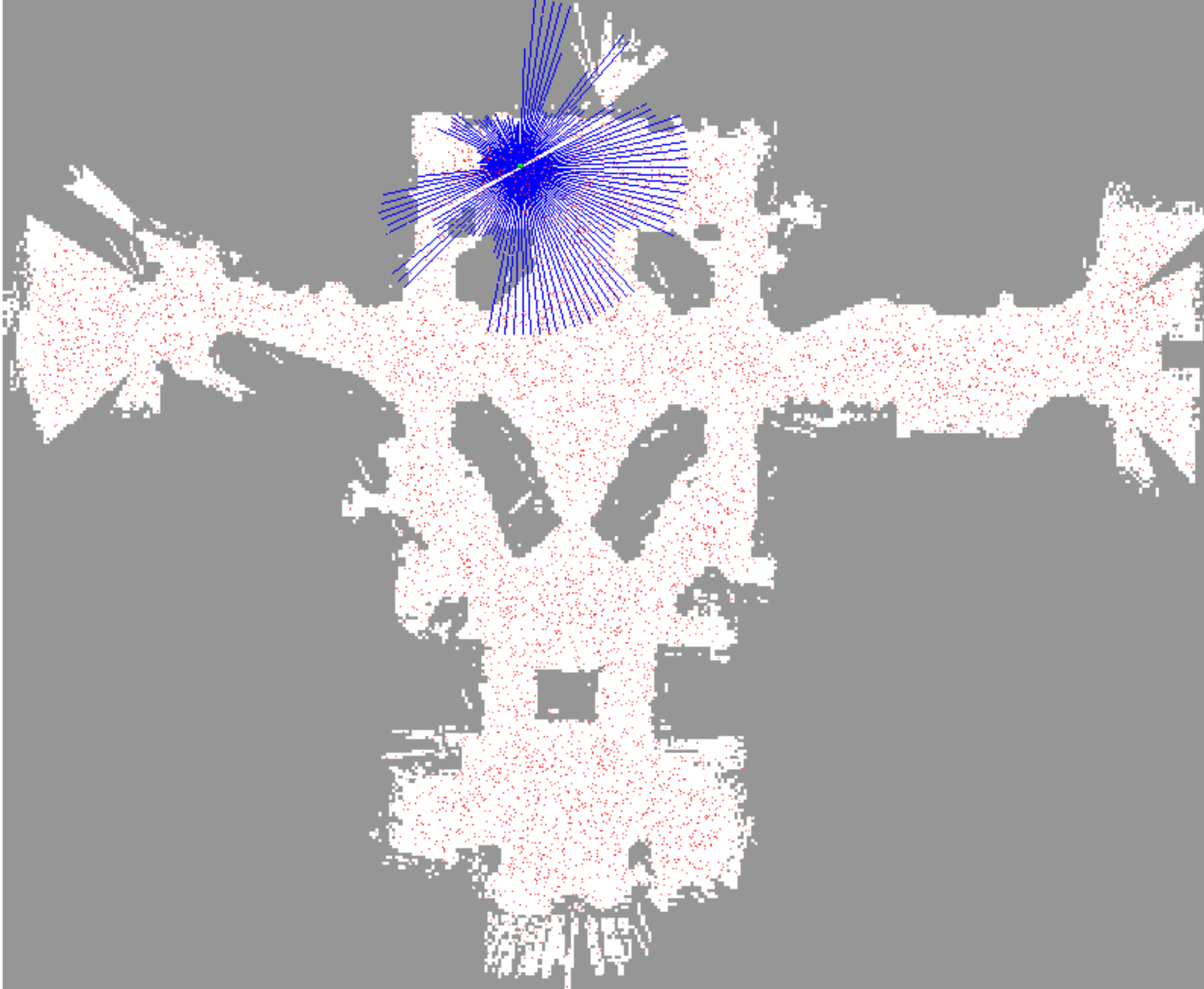
Virtues of resampling

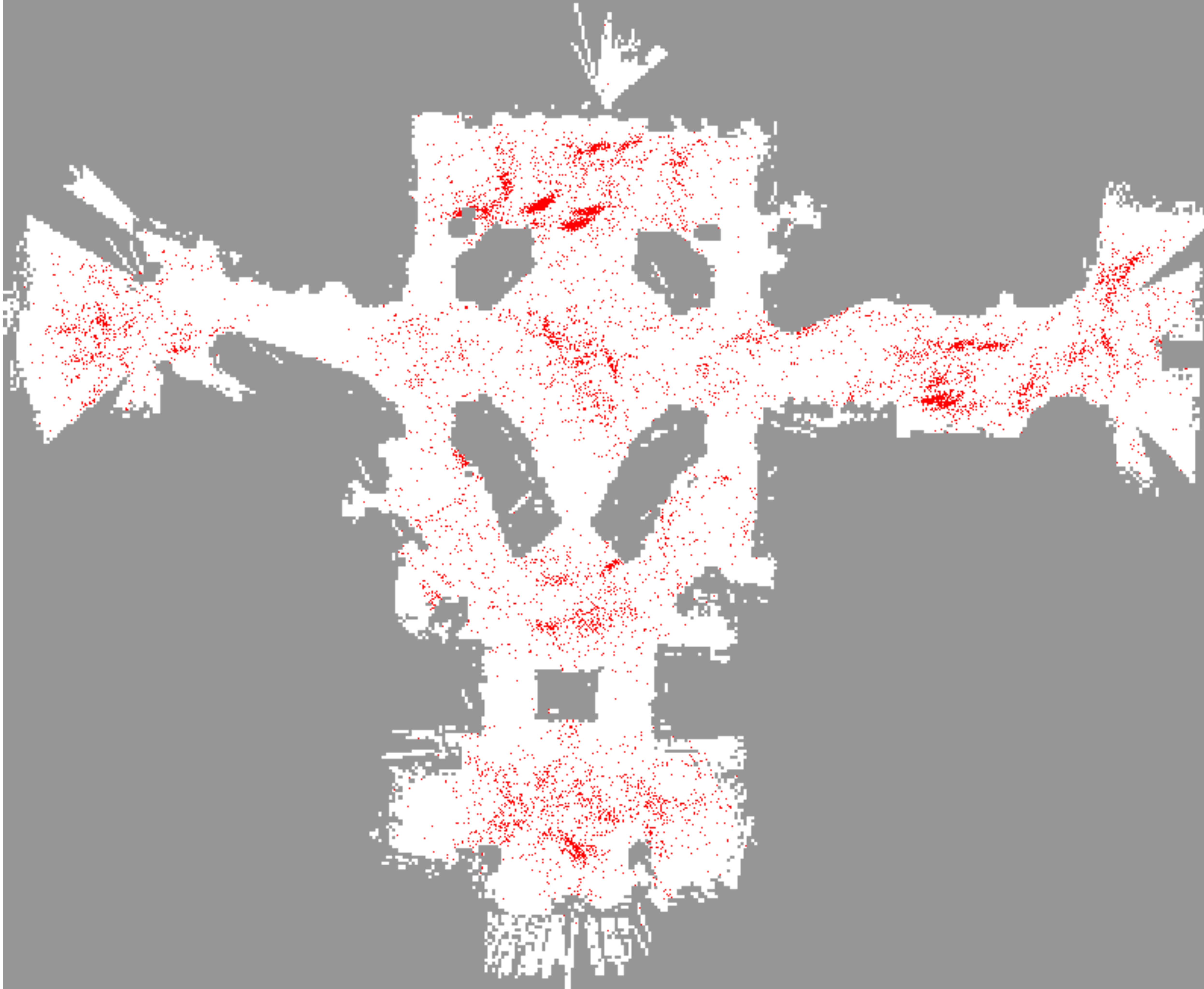


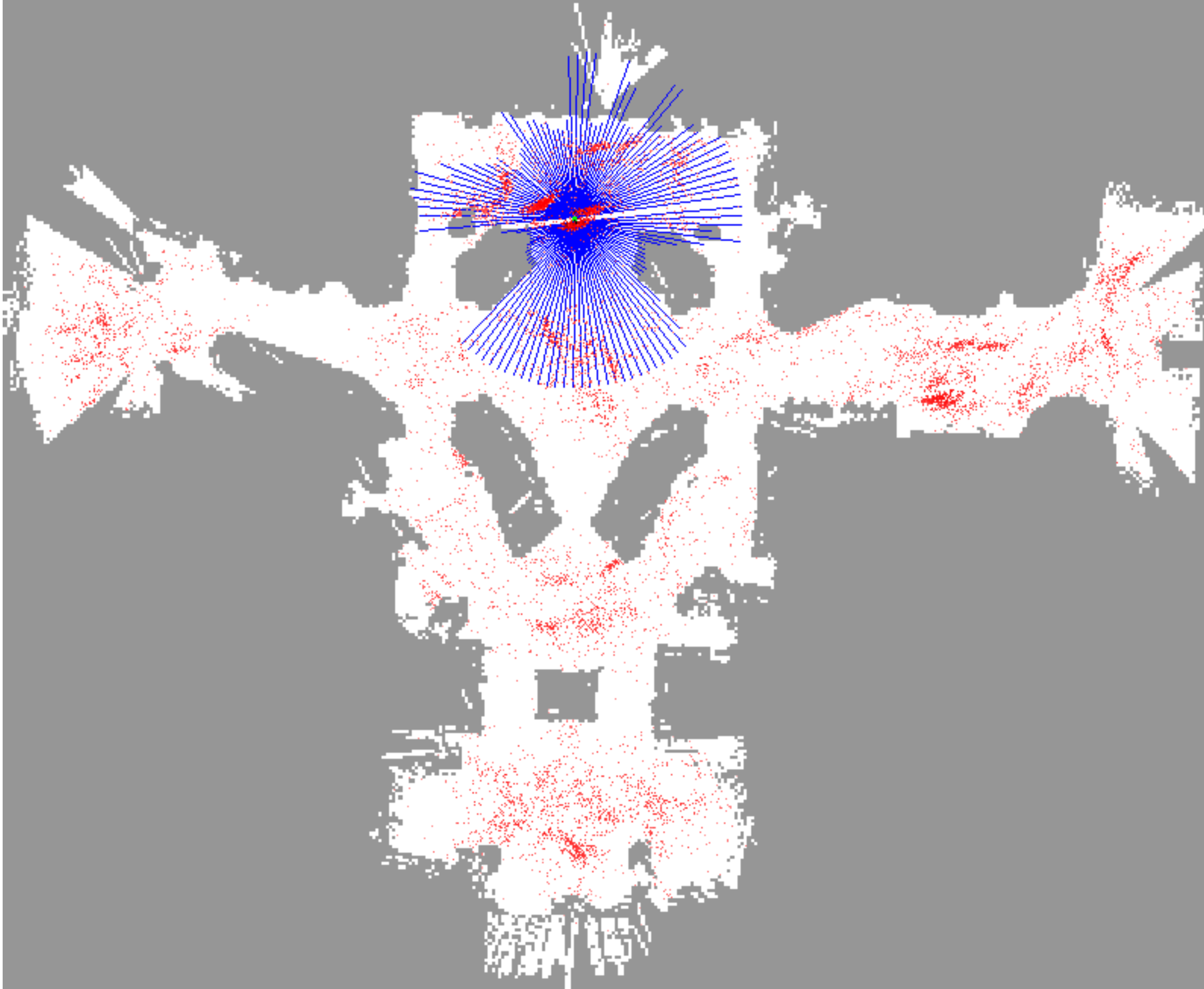
Why use particle filters?

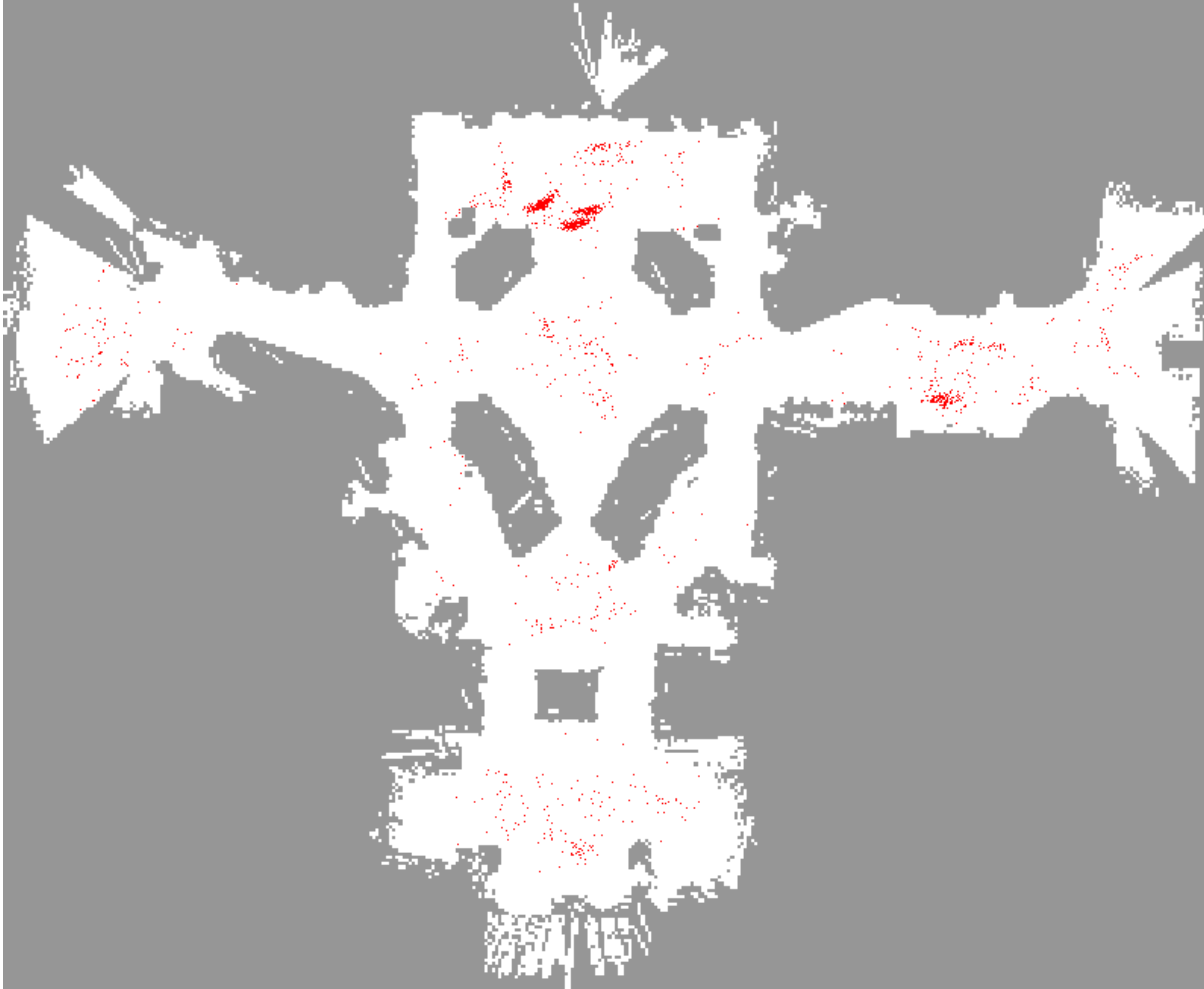
1. Can answer any query
2. Will work for any distribution, including multi-modal (unlike Kalman filter)
3. Scale well in computational resources (embarrassingly parallel)
4. Easy to implement!

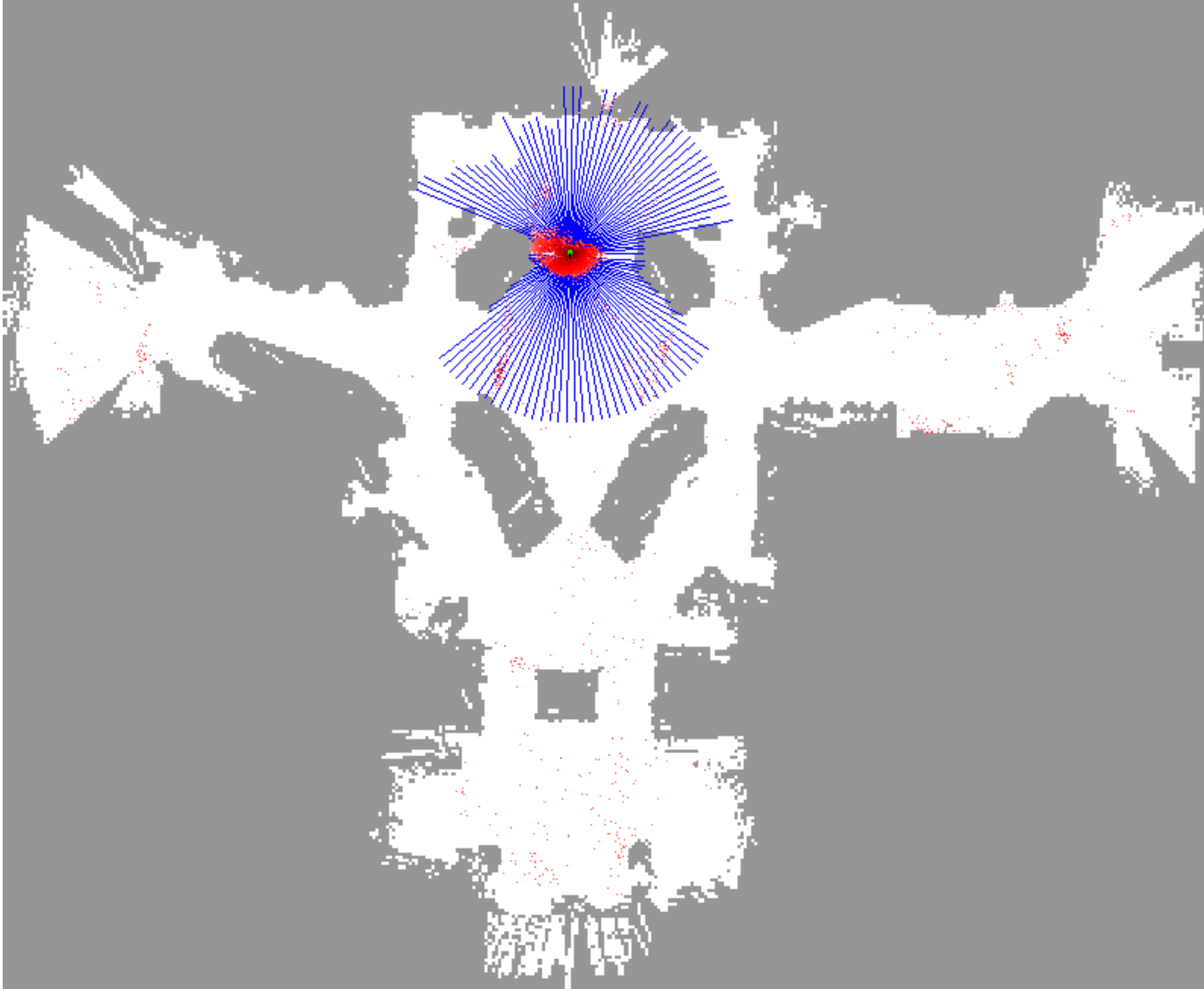


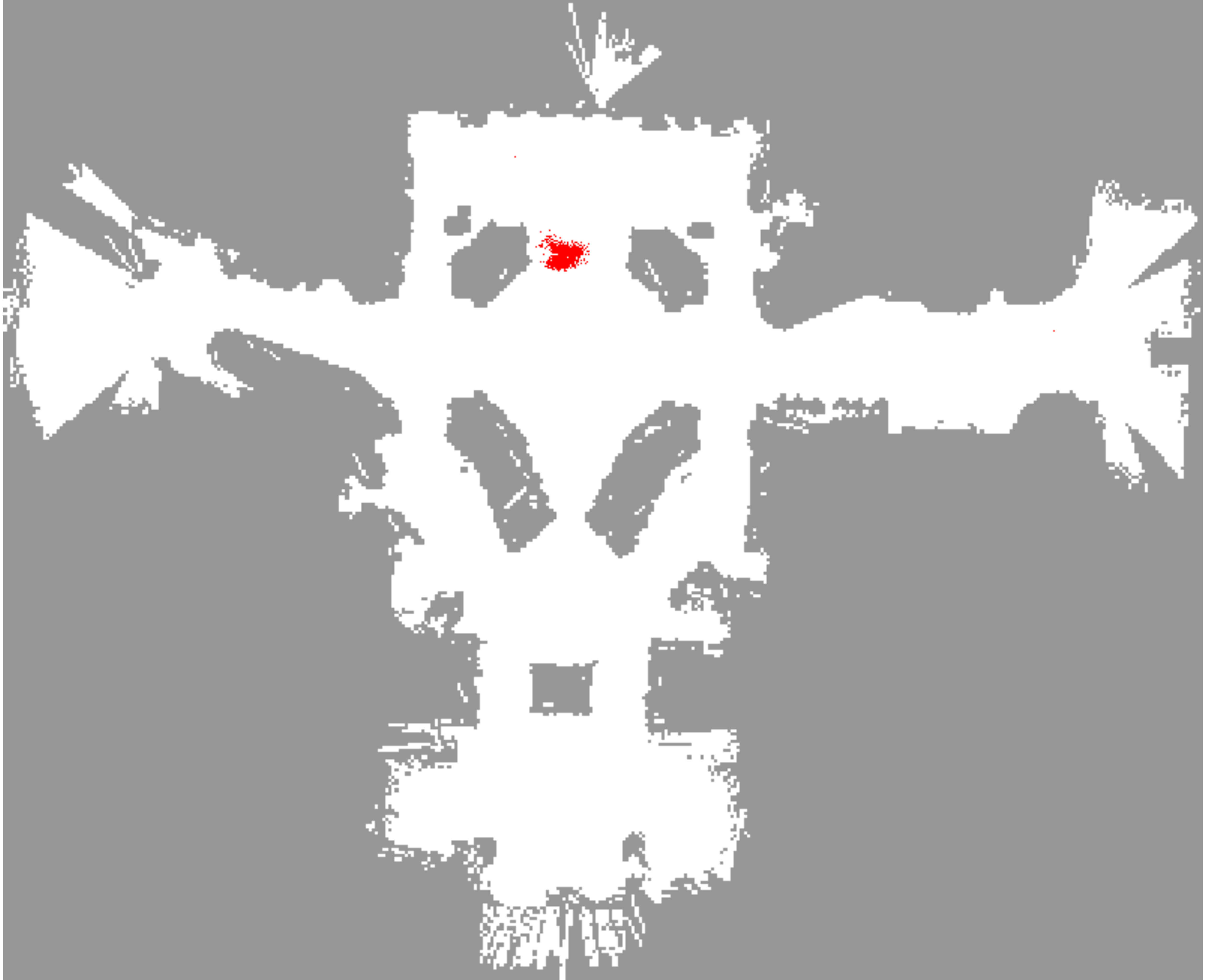












Same fundamental Bayes rule again and again ...

Non-parametric Filters

Grid up state space

Histogram Filter

Use a fixed set of samples

Normalized Importance Sampling

Resample

Particle Filter

Are we done?

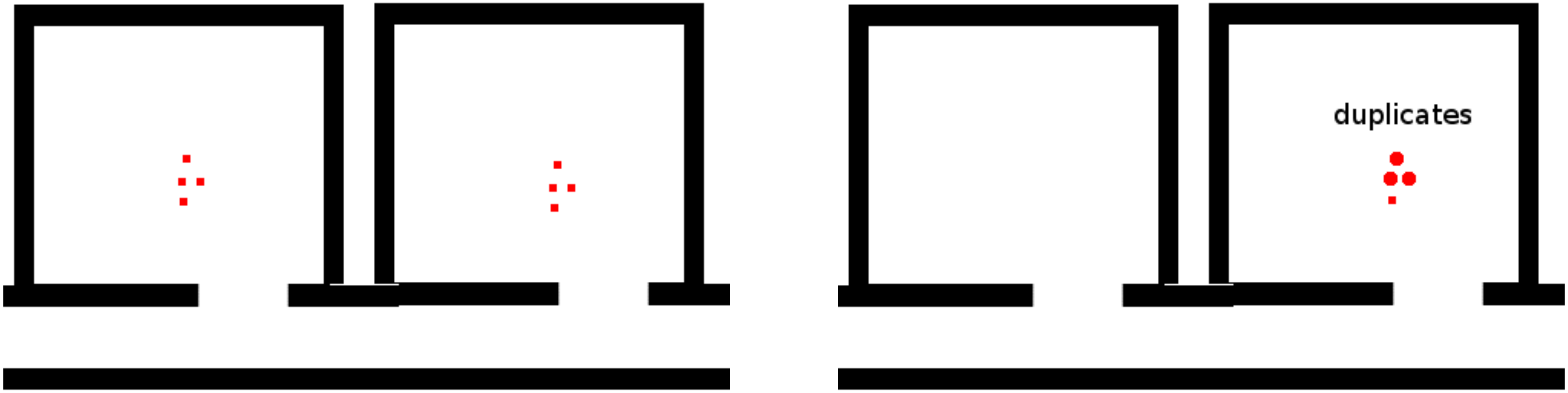
No!

Lots of practical problems to deal with

Problem 1: Two room challenge

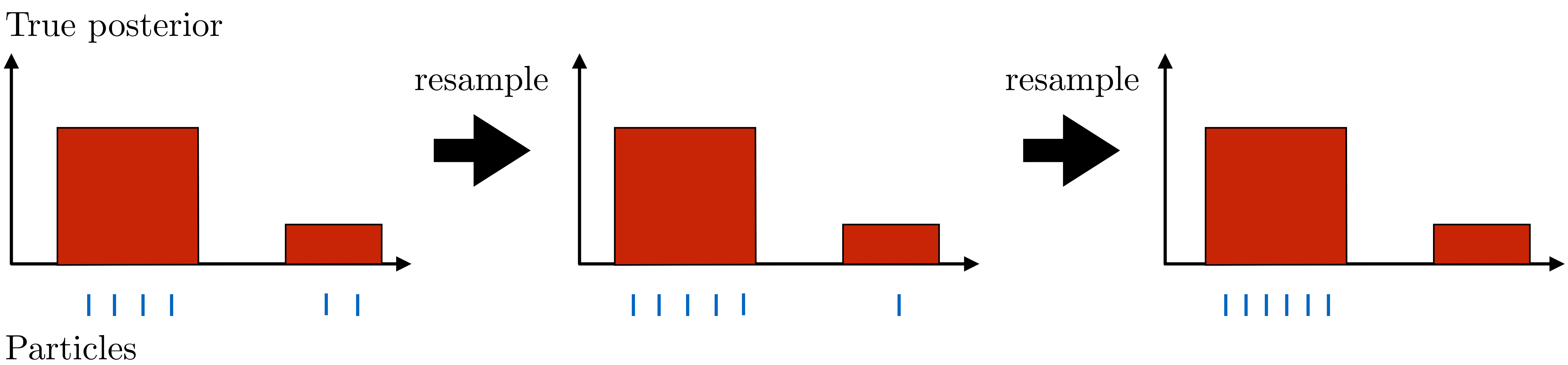
Given: Particles equally distributed, no motion, no observation

What happens?



All particles migrate to the other room!!

Reason: Resampling increases variance



Resampling collapses particles, reduces diversity, increases variance w.r.t true posterior

Fix 1: Choose when to resample

Key idea: If variance of weights low, don't resample

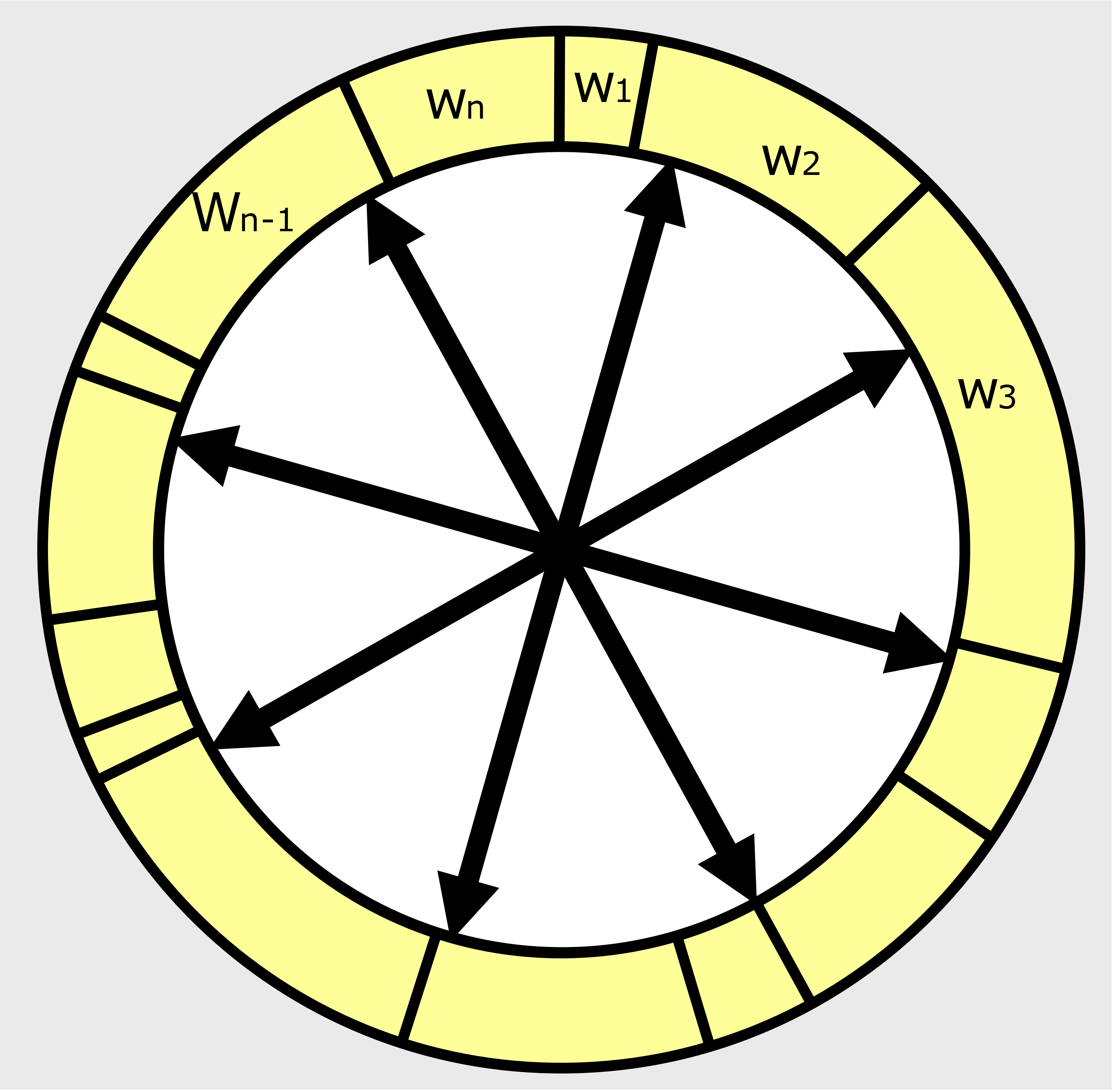
We can implement this condition in various ways

1. All weights are equal - don't resample

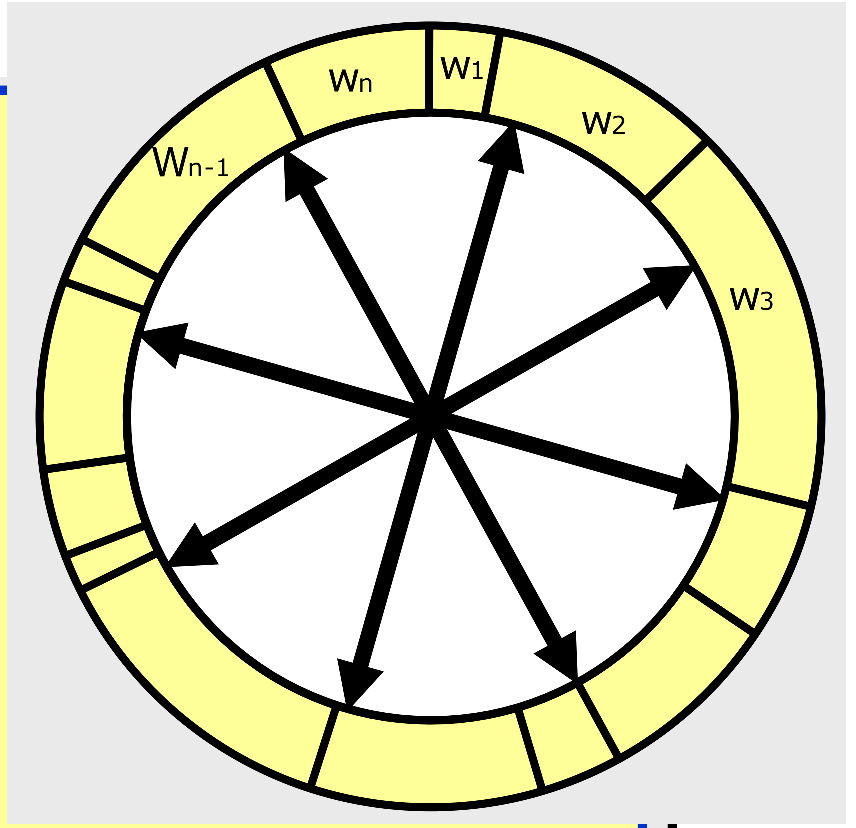
2. Entropy of weights high - don't resample

3. Ratio of max to min weights low - don't resample

Fix 2: Low variance sampling



Fix 2: Low variance sampling



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

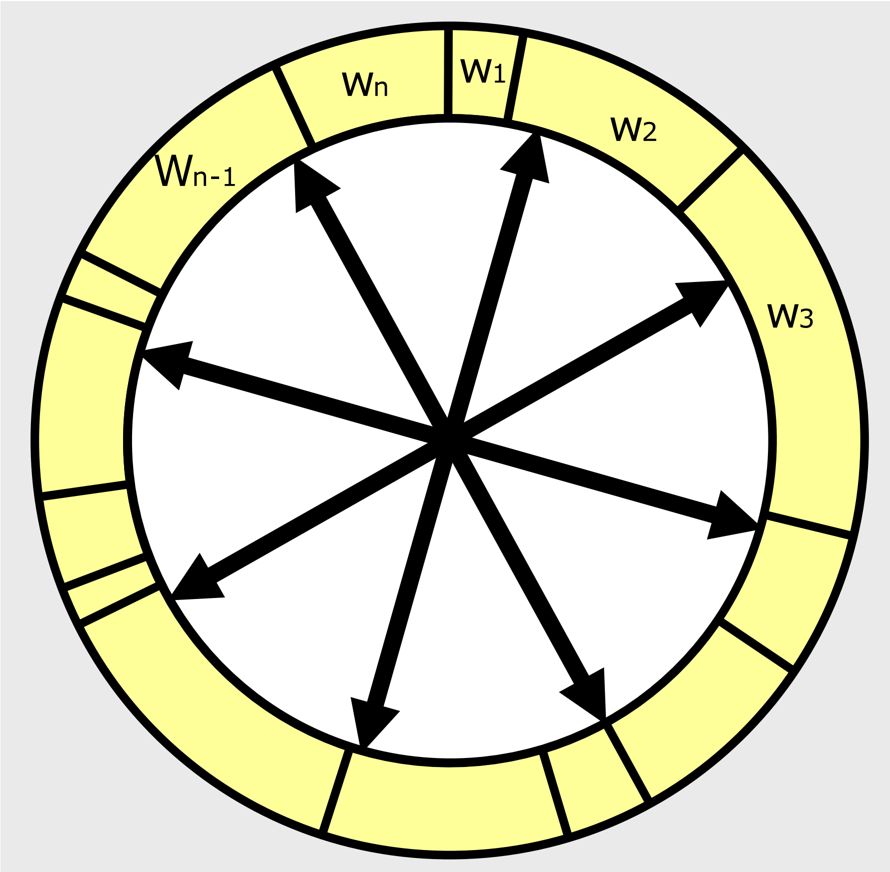
2. $S' = \emptyset, c_1 = w^1$ *Assumption: weights sum to 1*
3. **For** $i = 2 \ldots n$ ***Generate cdf***
4. $c_i = c_{i-1} + w^i$
5. $u_1 \sim U[0, n^{-1}], i = 1$ ***Initialize threshold***

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

11. **Return** S'

Also called **stochastic universal sampling**

Why does this work?



1. What happens when all weights equal?

2. What happens if you have ONE large weight and many tiny weights?

$$w1 = 0.5, \;\; w2 = 0.5/1000, \, w3 = 0.5/1000, \, ... \, w1001 = 0.5/1000$$

Problem 2: Particle Starvation

No particles in the vicinity of the current state

Why?

1. Unlucky set of samples

2. Committed to the wrong mode in a multi-modal scenario

3. Bad set of measurements

Fix: Add new particles

Which distribution should be used to add new particles?

1. Uniform distribution

2. Biased around last good measurement

3. Directly from the sensor model

Fix: Add new particles

When should we add new samples?

Key Idea: As soon as importance weights become too small, add more samples

1. Threshold the total sum of weights

2. Fancy estimator that checks rate of change.

Problem 3: Observation model too good!

Observation model is so peaky, that all particles die!

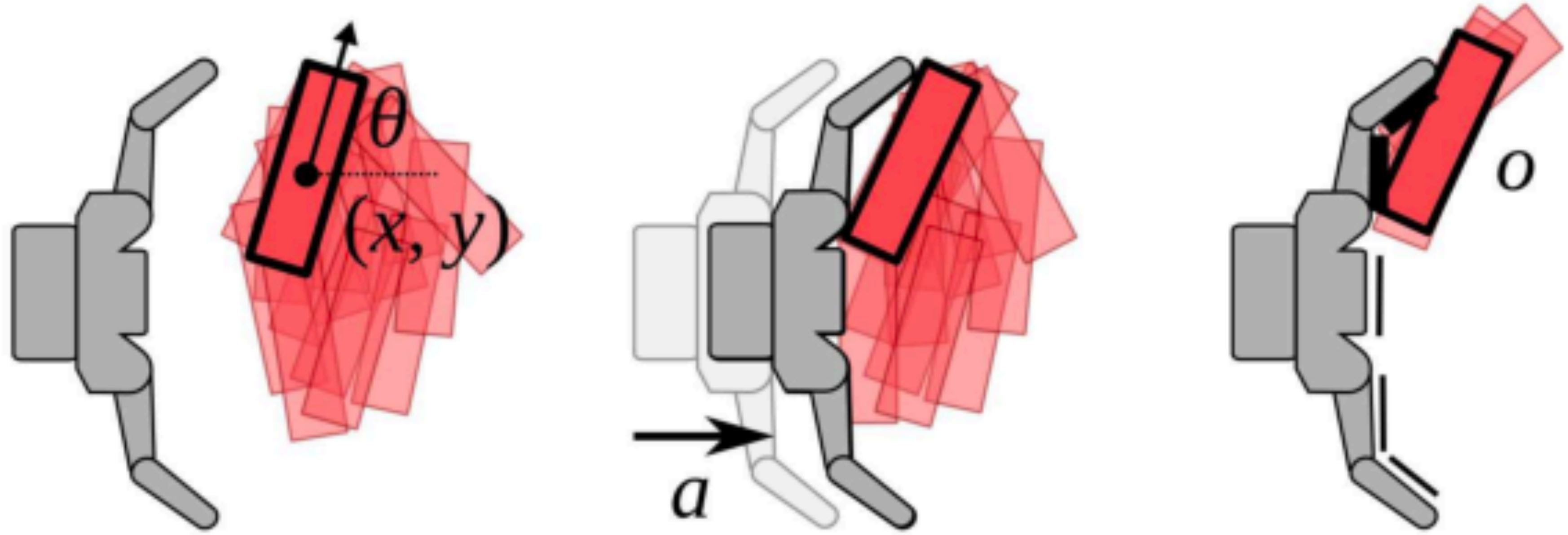
Fixes

1. Sample from a better proposal distribution than motion model!

2. Squash the observation model (apply a power of $1/m$ to all probabilities. m observations count as one)

3. Last resort: Smooth your observation model with a Gaussian (you are pretending your observation model is worse than it is)

Fix 1: Sample from a better proposal distribution



Koval et al. 2017

Contact observation may kill ALL particles!

Key Idea: Sample and weigh particles correctly

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

(Sample) (Reweigh)

Problem 4: How many samples is enough?

Example: We typically need more particles at the beginning of run

Key idea: KLD Sampling (Fox et al. 2002)

1. Partition the state-space into bins

2. When sampling, keep track of the number of bins

3. Stop sampling when you reach a statistical threshold that depends on the number of bins

(If all samples fall in a small number of bins -> lower threshold)

KLD sampling

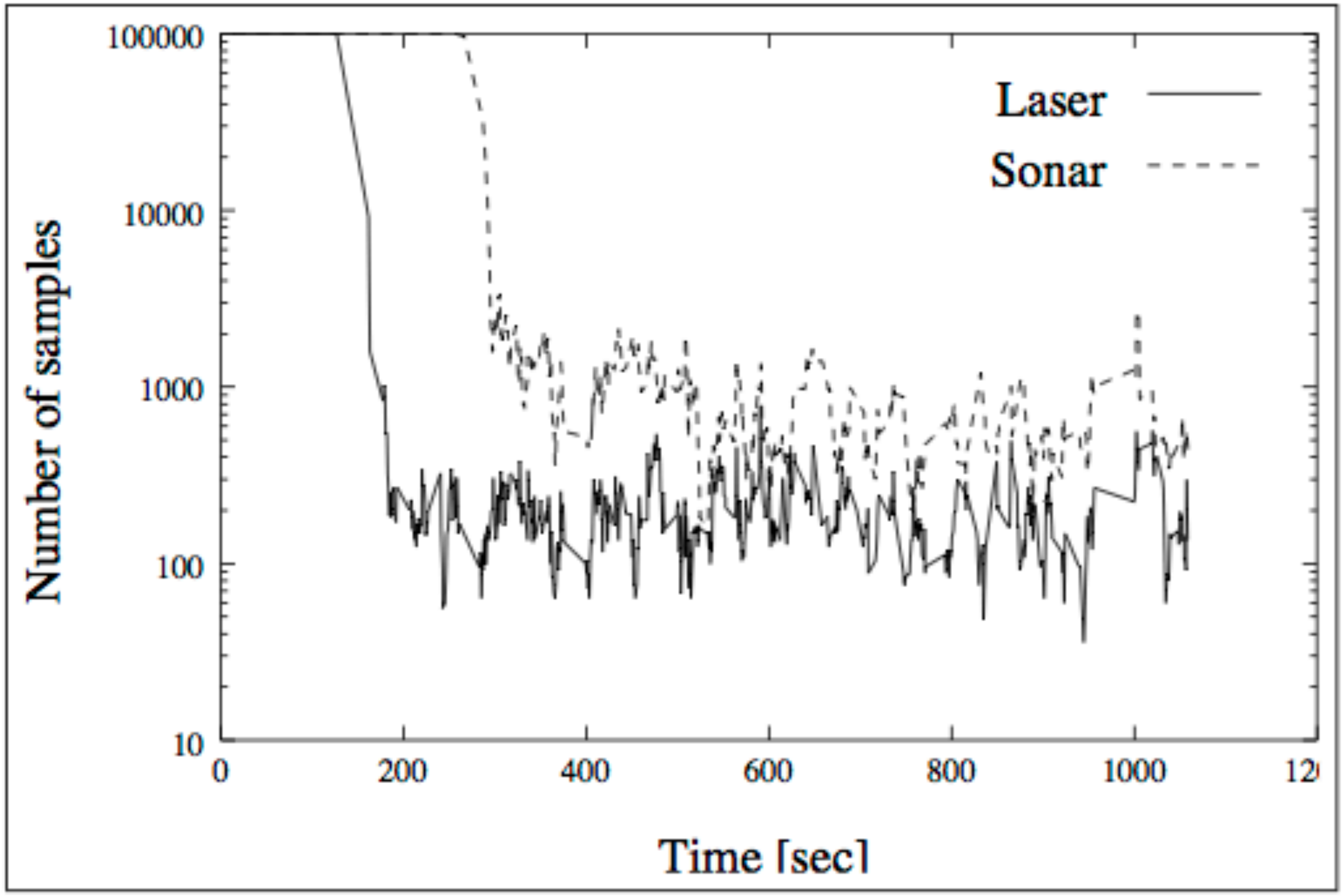


Figure 8.18 KLD-sampling: Typical evolution of number of samples for a global localization run, plotted against time (number of samples is shown on a log scale). The solid line shows the number of samples when using the robot's laser range-finder, the dashed graph is based on sonar sensor data.

Closing: Myth busting Particle filters

1. Particle Filter = Sample from motion model, weight by observation

BUSTED

(sample from any good proposal distribution)

2. Particle filters are for localization

BUSTED

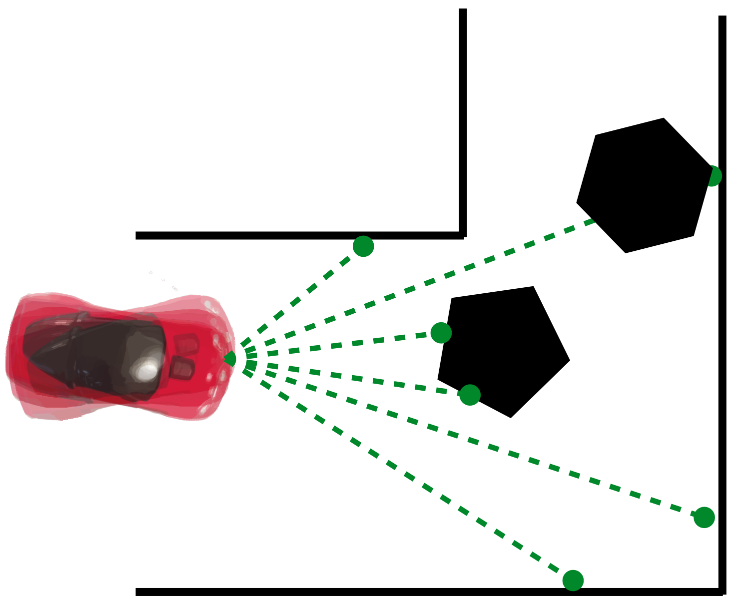
(any continuous space estimation problem)

3. Particle filters are to do with samples

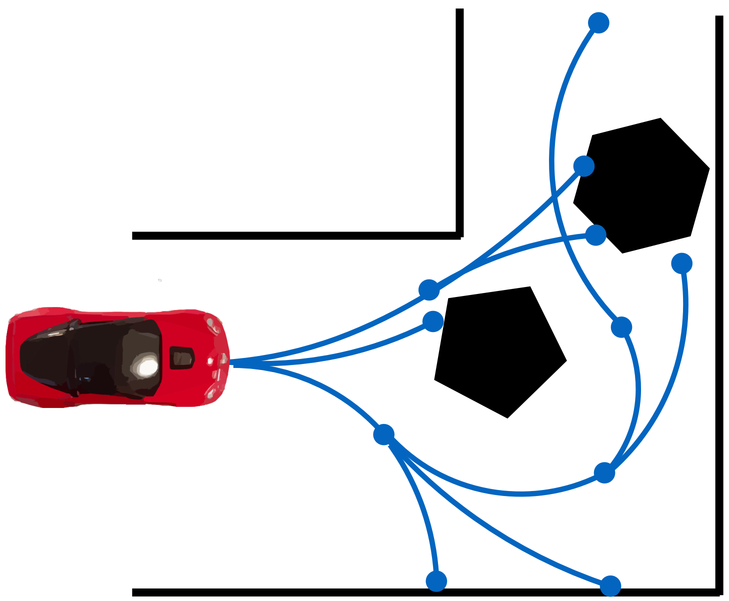
BUSTED

(normalized importance sampling also uses samples but no resampling)

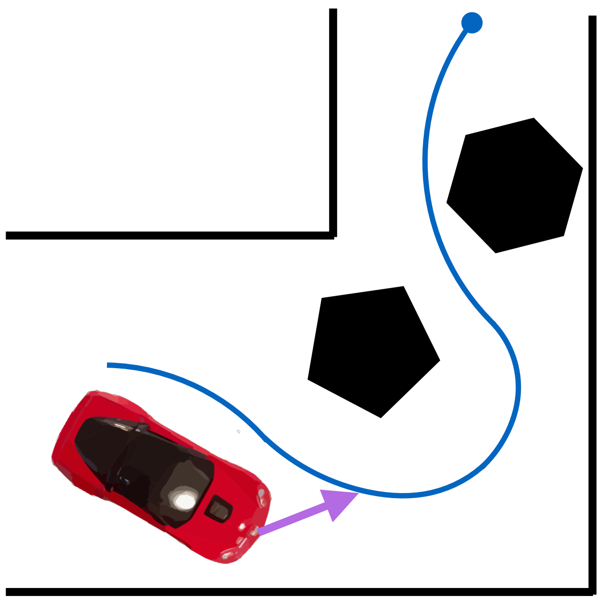
Estimate state



Plan a sequence of motions



Control robot to follow plan



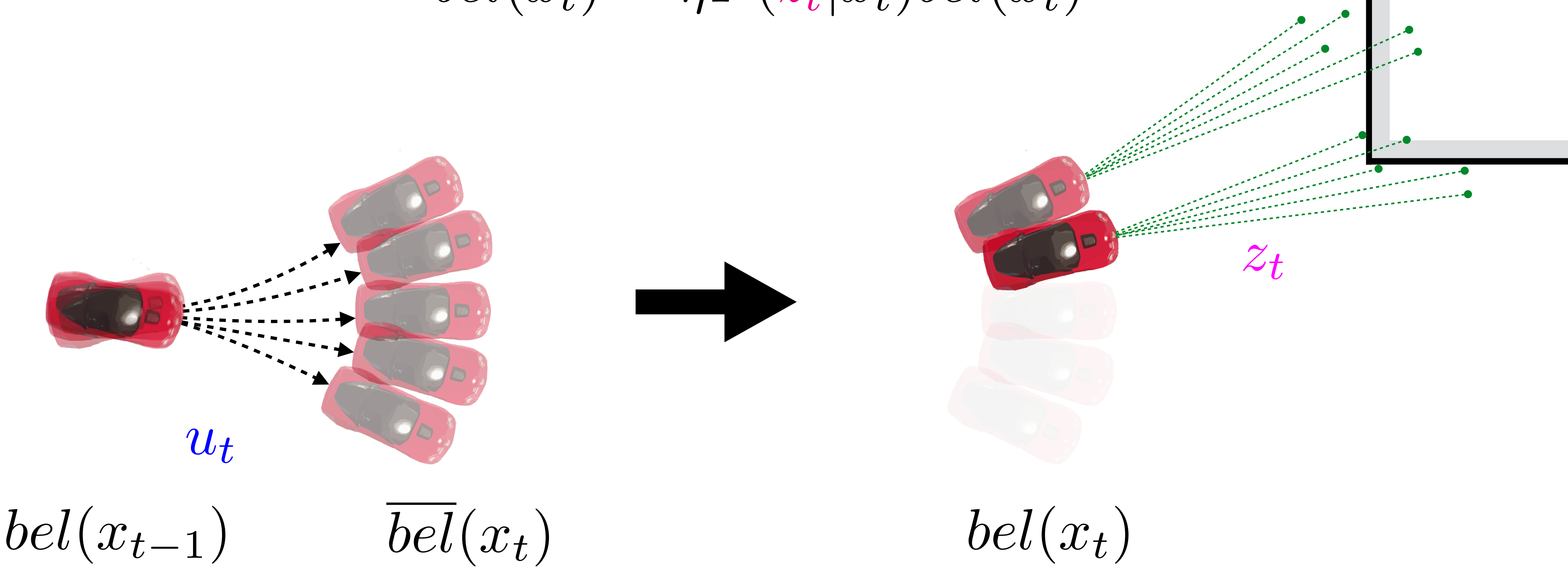
Bayes filter in a nutshell

Step 1: Prediction - push belief through dynamics given action

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

Step 2: Correction - apply Bayes rule given measurement

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



Bayes filter is a powerful tool

Localization Mapping SLAM POMDP