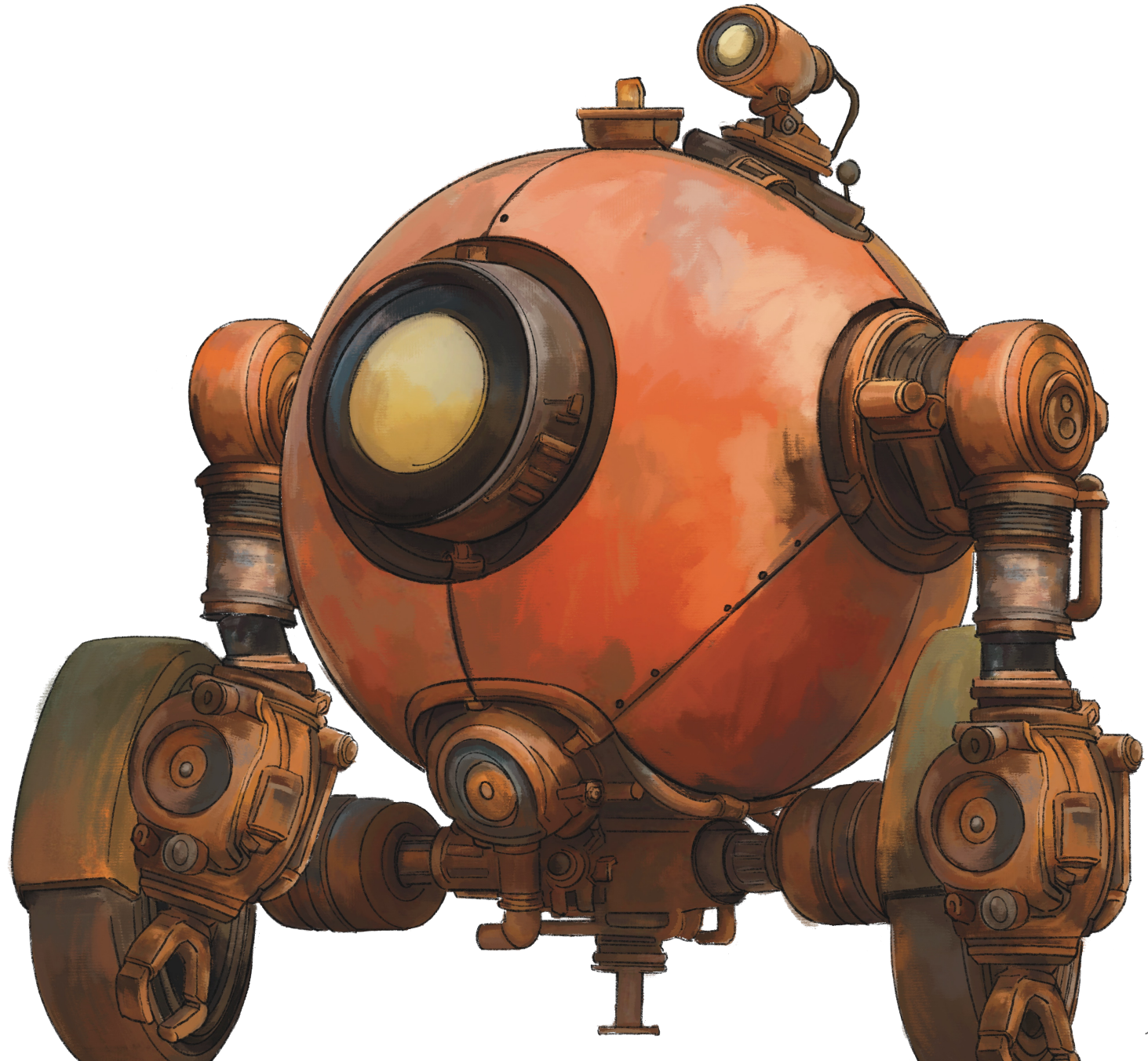


CS 3630, Fall 2026

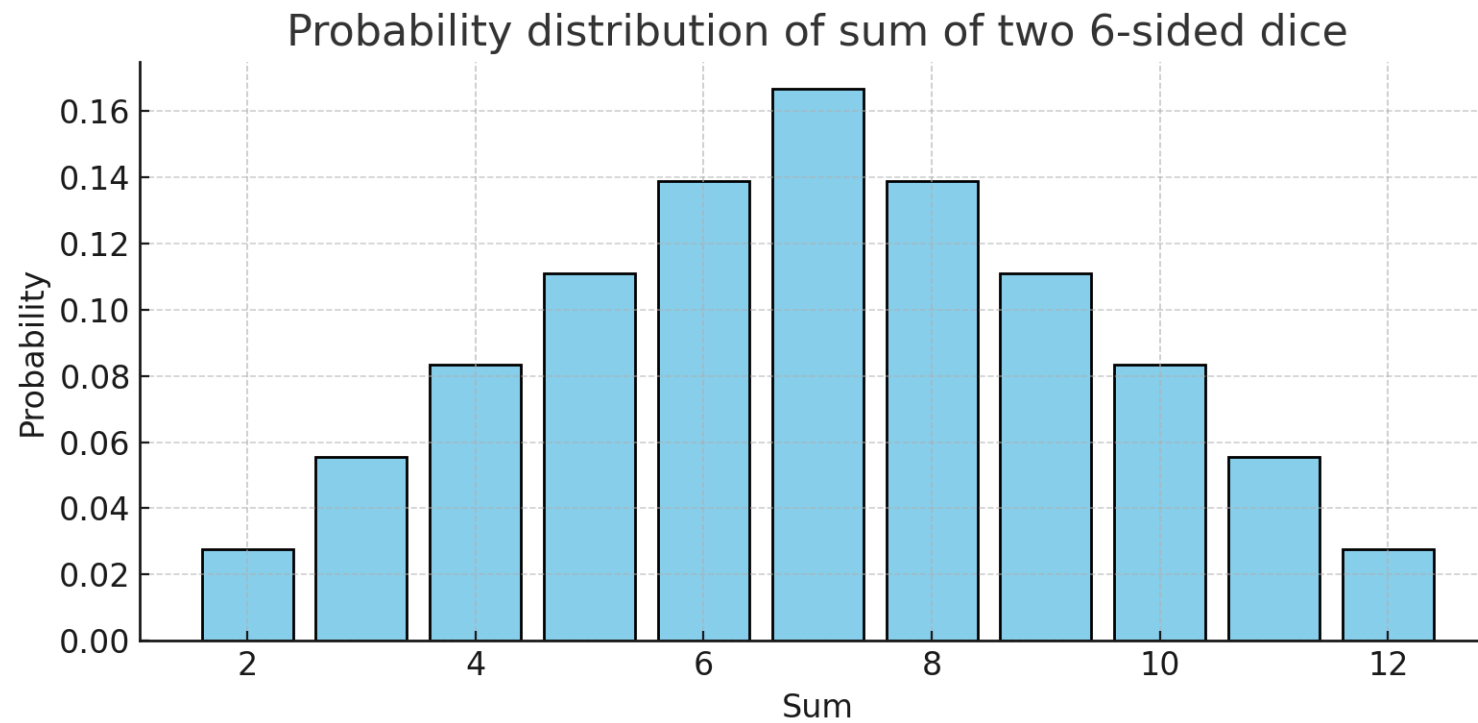
*Lecture 15:
A Logistics Robot:
Perception*

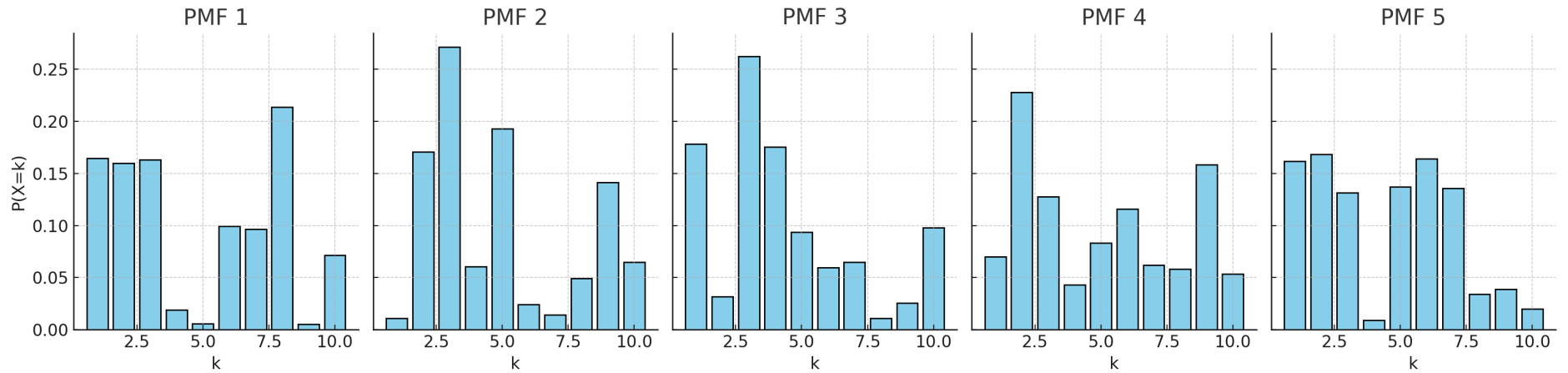


A large, rusted, spherical robot with a single large eye and mechanical limbs. The robot is primarily orange-brown with visible rust and wear. It has a large, circular, glowing yellow eye in the center of its face. The body is spherical with various mechanical details, including rivets and bolts. The limbs are thick and mechanical, with circular joints. The background is dark.

Recap

Motion uncertainty accumulates like dice sum

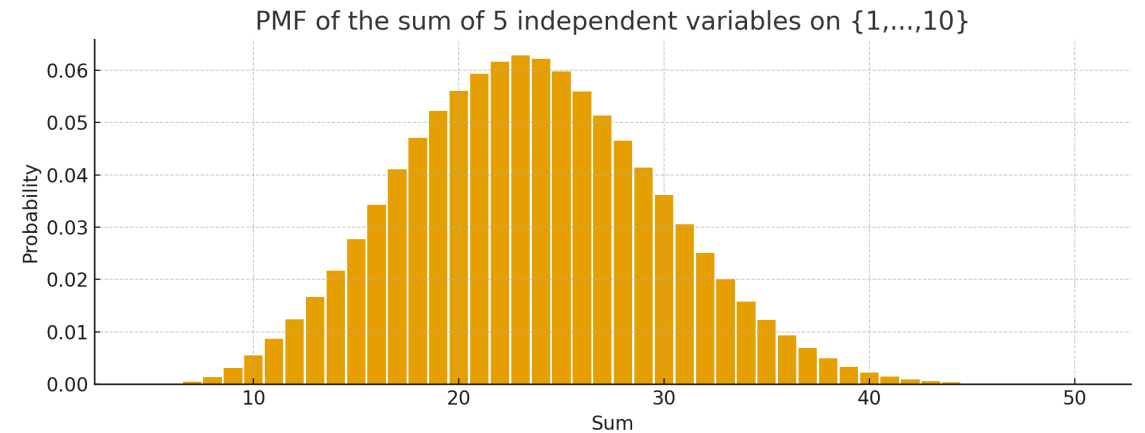




Even crazier....

If we take 5 *arbitrary* PMFs (crazy dice) on the numbers 1..10, and look at the PMF on the sum, we get something already very close to a Gaussian:

Mean?



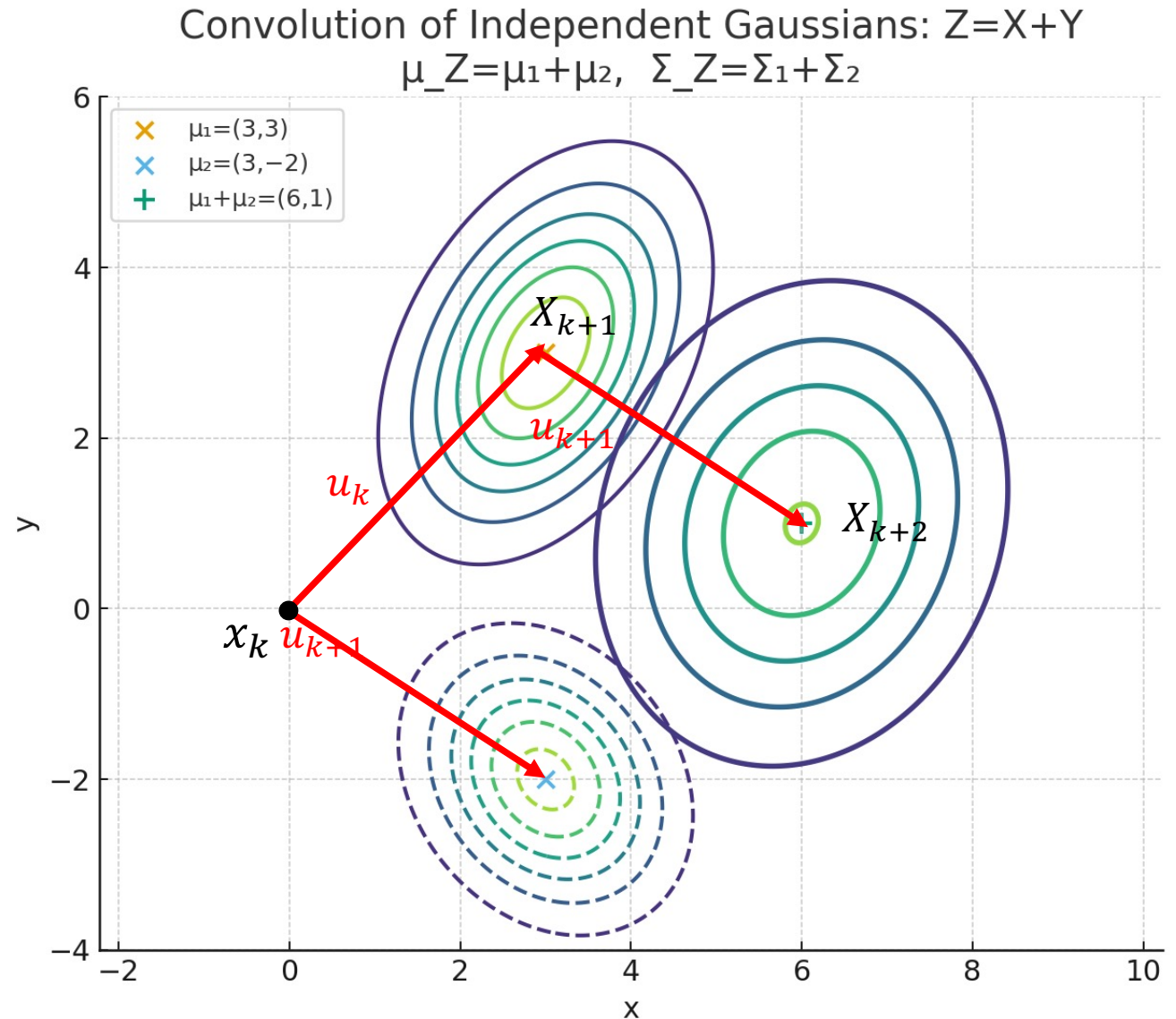
Bivariate Gaussians

For our motion model, we'll use

$$x_{k+1} = x_k + u_k + \eta_k$$

with $x_{k+1}, x_k, u_k, \eta_k \in \mathbb{R}^2$ and $\eta_k \sim N(0, \Sigma)$.

X_{k+2} is a bivariate Gaussian,
 $X_{k+2} \sim N(x_k + u_k + u_{k+1}, \Sigma_1 + \Sigma_2)$



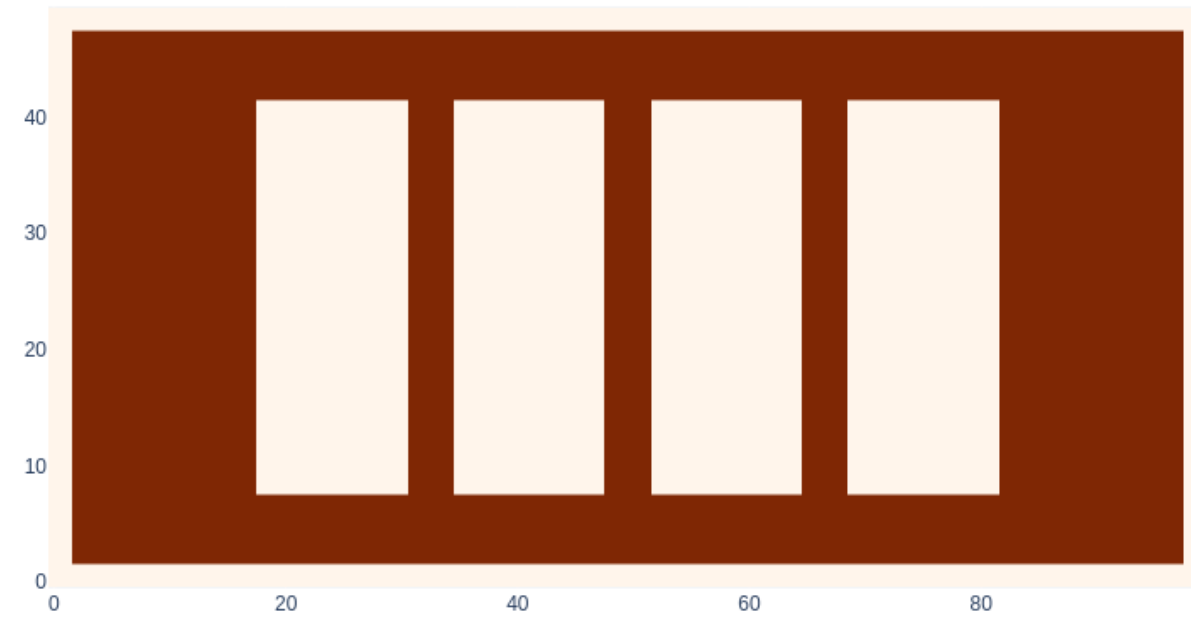
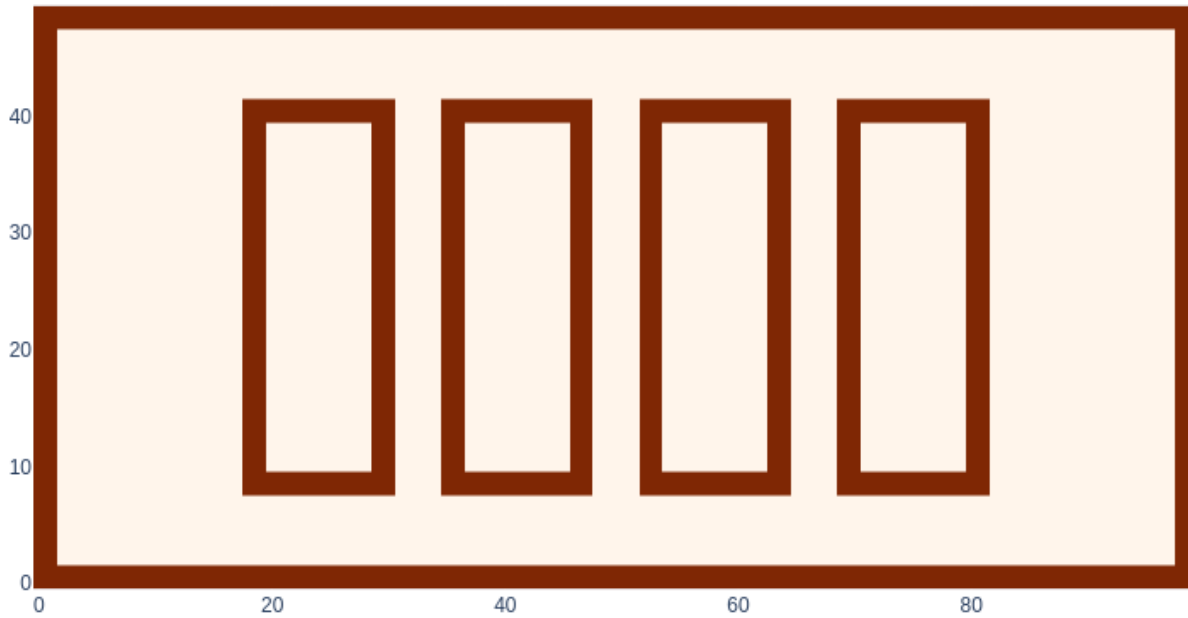
Likelihood for the Ideal Proximity Sensor



- We can plot the likelihood for each possible value of z_k .

$$\mathcal{L}(x_k; z_k = ON) = \begin{cases} 1 & d(x_k) \leq d_0 \\ 0 & \text{otherwise} \end{cases}$$

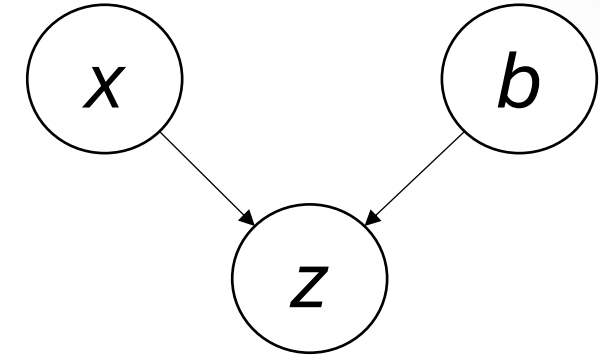
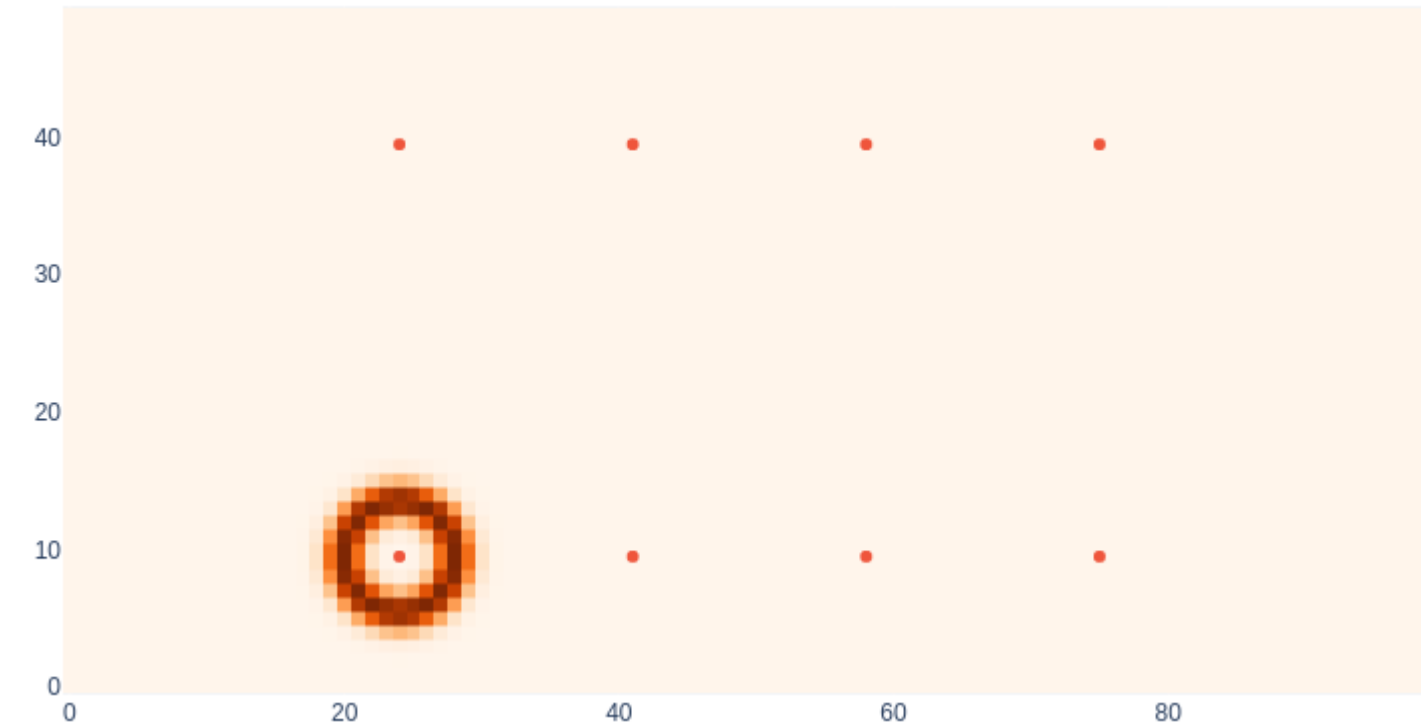
$$\mathcal{L}(x_k; z_k = OFF) = \begin{cases} 0 & d(x_k) \leq d_0 \\ 1 & \text{otherwise} \end{cases}$$



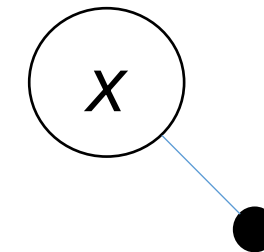
- The likelihood is a function of x_k . It is not a probability distribution!
- The specific form of the likelihood depends on which value of z_k was observed.

Measurement Likelihood

- For the case $b_i = b_0$ and $z_k = 4.03$, we obtain the plot for $\mathcal{L}(x_k; 4.03, b_0)$ shown below (and in book).



$$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z_k - h(x_k; b_i))^2}{2\sigma^2}}$$



Localization: the roadmap

Goal

Maintain a belief about robot location as actions and readings arrive.

Bayes filter

Predict with the motion model, then correct with the sensor likelihood.

Implementations

A grid and a set of particles represent the same kind of belief differently.

The recurring question: where could the robot be, and how plausible is each location?

Perception

In this chapter, the role of perception is to solve the **localization** problem, i.e., to determine an estimate of x_t , the robot's state at time t .

- Mathematically, the problem is to estimate the state x_t , given the action history $u_1 \dots u_n$ and sensing history $z_1 \dots z_n$

$$Bel(x_t) = P(x_t | u_1, z_1, u_2 \dots z_{t-1}, u_{t-1}, z_t)$$

- Computationally, this is a difficult problem.
- We'll see two approaches:
 - Particle Filtering
 - Markov Localization
- The Bayes filter is the workhorse in these.

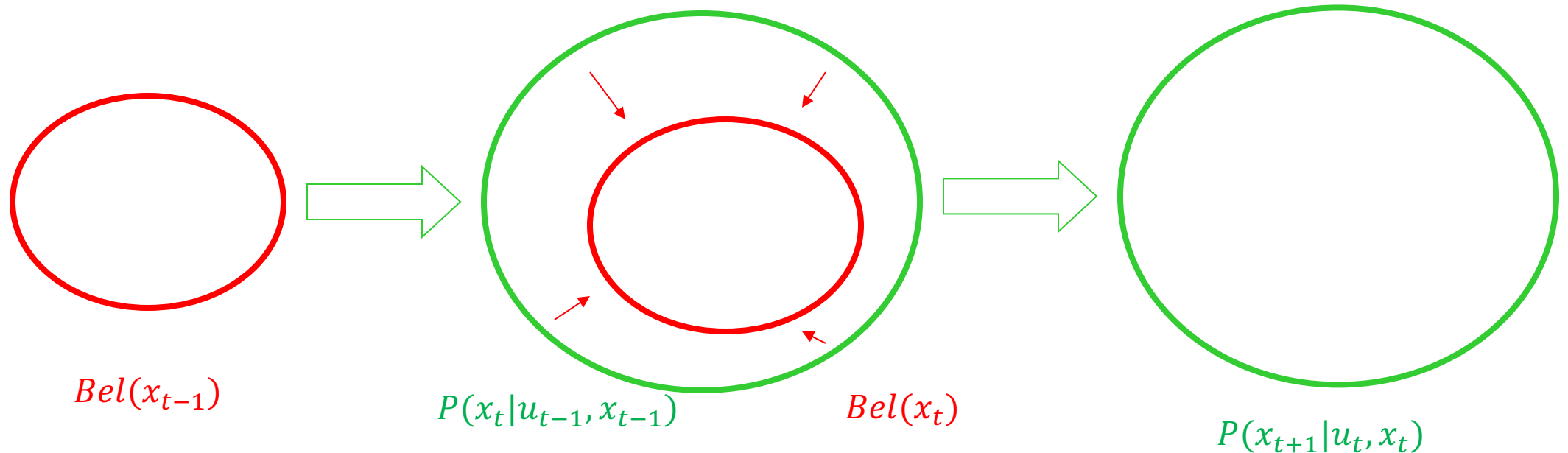
The Bayes Filter

The **Bayes filter** is the culmination of all the work we've done in applying probability theory to the representation of uncertainty in *state*, *actions*, and *sensing*.

- **Prior**: probabilistic description of *uncertainty in the state* (before acting or sensing at time t).
 - **Motion model**: conditional probability that describes *uncertainty in the actions*.
 - **Sensor model**: conditional probability model that describes *uncertainty in the sensor measurements*.
- The output of the Bayes filter at time t is $Bel(x_t)$.

The Bayes Filter

- Two phases:
 - a. Prediction Phase (uncertainty grows)
 - b. Measurement Phase (uncertainty reduction)

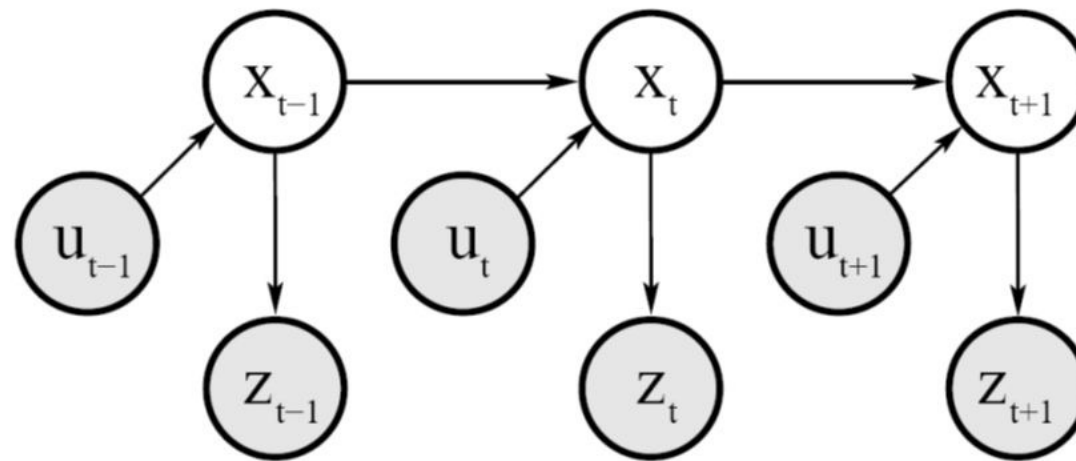


Bayes Filters: Framework

- Let x be the state of the robot (e.g., its location)
- **Given:**
 - Stream of observations z and action data $u: \{u_1, z_1 \dots, u_{t-1}, z_t\}$
 - Sensor model $P(z|x) \rightarrow$ *likelihood function* $\mathcal{L}(x;z)$ when z is given.
 - Motion model $P(x_t|u_{t-1}, x_{t-1})$.
 - Prior probability of the system state $P(x)$.
- **Wanted:**
 - Estimate of the state X of a dynamical system.
 - The posterior of the state is also, as before, sometimes called the **Belief**:

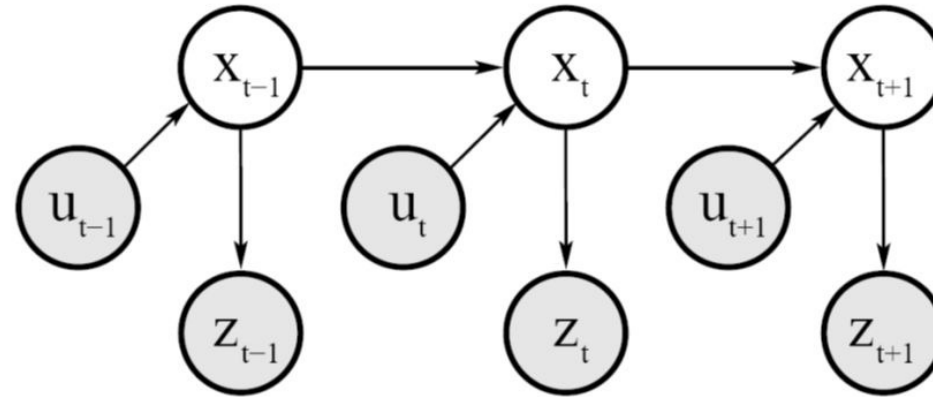
$$Bel(x_t) = P(x_t|u_1, z_1 \dots, u_{t-1}, z_t)$$

- We can put all of this into our nice Bayes net formalism, for *modeling* purposes.
- The robot's state at time t is stochastically dependent on its state at time $t - 1$ and the control input u_t . The measurement z_t depends stochastically on the state at time t .
- Gray elements are observable and white are hidden.



I.e., a *continuous* hidden Markov model (HMM) / dynamic Bayesian network (DBN).

Markov Assumption



$$p(z_t | x_{1:t}, z_{1:t-1}, u_{1:t}) = p(z_t | x_t)$$

$$p(x_t | x_{1:t-1}, z_{1:t-1}, u_{1:t}) = p(x_t | x_{t-1}, u_t)$$

Underlying Assumptions

- Static world
- Independent noise

"The future is independent of the past given the present."

Bayes Rule

$$P(x|z) \propto \mathcal{L}(x; z)P(x) = \text{likelihood} \cdot \text{prior}$$

x is robot pose and z is sensor data

$p(x|z)$ → *Posterior* probability of x given z

$\mathcal{L}(x; z)$ → *Likelihood* function of state x given measurement z

$p(x)$ → *Prior* probability distribution on state x

Bayes Filters

z = observation
 u = action
 x = state

Belief that robot is
in state $X = x_t$ at
time step t

If I was in state x_{t-1} and I
executed action u_{t-1} what is
the probability that I arrive to
state x_t

Weight this
probability by the
belief that I was
actually in state x_{t-1}

$$Bel(x_t) \propto \mathcal{L}(x_t; z_t) \int P(x_t | u_{t-1}, x_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

How likely is the
state x_t given
that I saw the
observation z_t

Integrate over all possible previous states, x_{t-1}

Bayes Filters

$$\boxed{Bel(x_t)} = P(x_t | u_1, z_1, \dots, u_{t-1}, z_t)$$

$$\text{Bayes} = \eta P(z_t | x_t, u_1, z_1, \dots, u_{t-1}) P(x_t | u_1, z_2, \dots, u_{t-1})$$

$$\text{Markov/Likelihood} \propto \mathcal{L}(x_t; z_t) P(x_t | u_1, z_1, \dots, u_{t-1})$$

$$\text{Total prob.} \propto \mathcal{L}(x_t; z_t) \int P(x_t | u_1, z_1, \dots, u_{t-1}, x_{t-1}) P(x_{t-1} | u_1, z_1, \dots, u_{t-1}) dx_{t-1}$$

$$\text{Markov} \propto \mathcal{L}(x_t; z_t) \int P(x_t | u_{t-1}, x_{t-1}) P(x_{t-1} | u_1, z_1, \dots, u_{t-1}) dx_{t-1}$$

$$\propto \mathcal{L}(x_t; z_t) \int P(x_t | u_{t-1}, x_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

Bayes filtering: predict-update

Prediction uses the motion model

Propagate the previous belief through the action and motion uncertainty.

Correction uses the sensor likelihood

Reweight predicted states by agreement with the reading, then normalize.

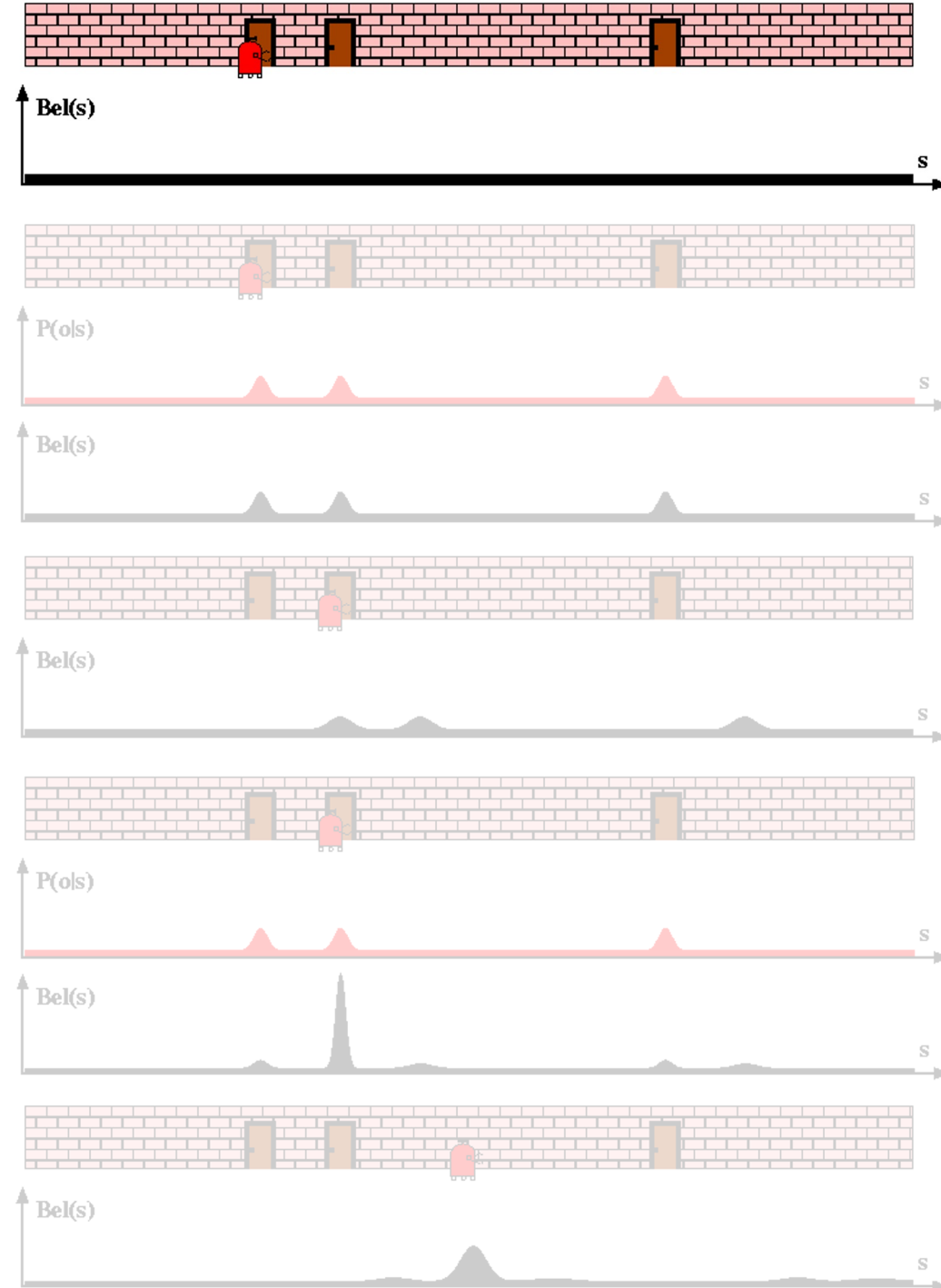
Checkpoint: which step uses the action, and which step uses the new measurement?

Bayes Filters for Robot Localization

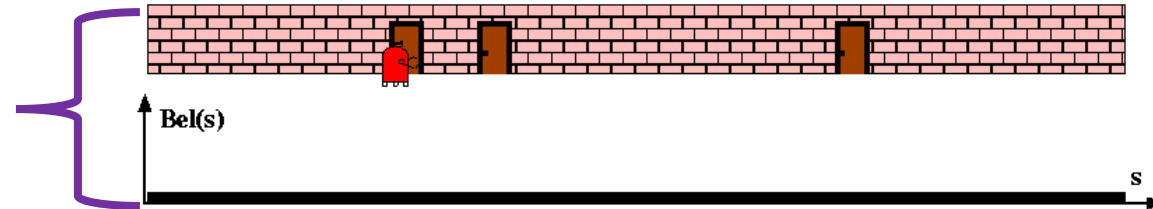
Let's see how it works using a simple example:

- The robot moves from left to right.
- From time to time, it takes a sensor reading.

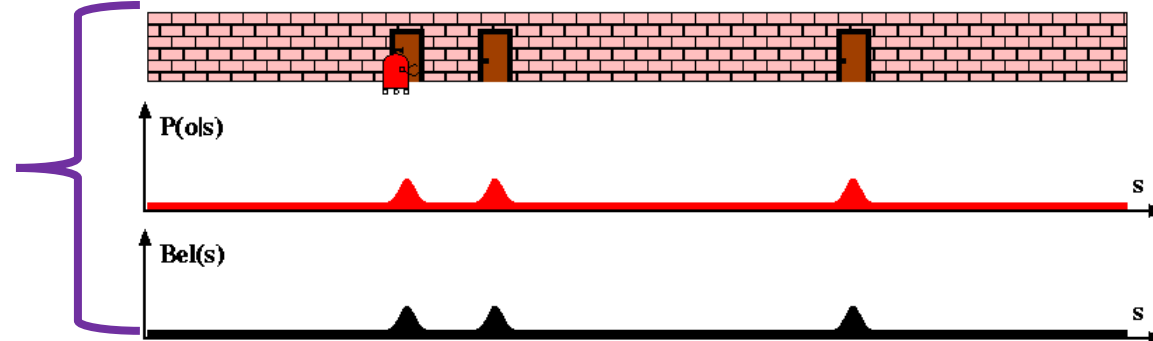
How is the state estimate updated??



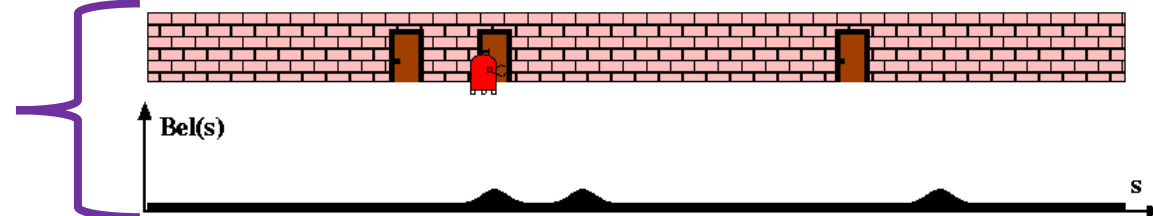
Initial Guess: Could be anywhere...



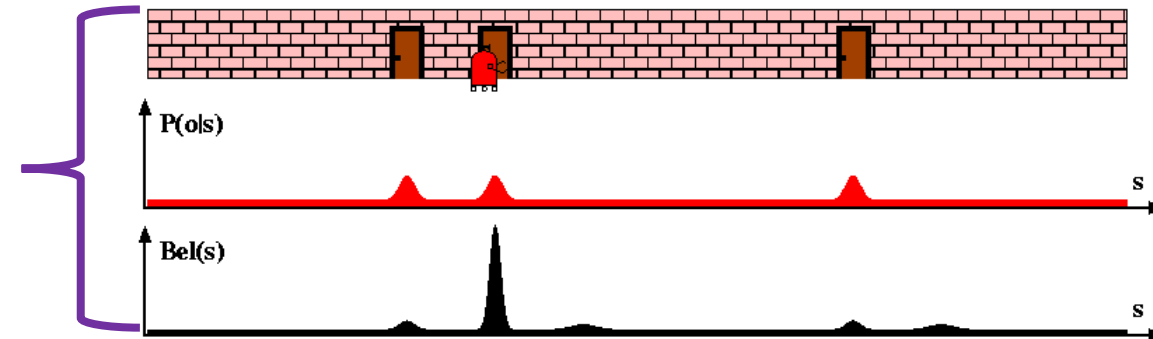
Take a measurement: we're probably in front of a door...



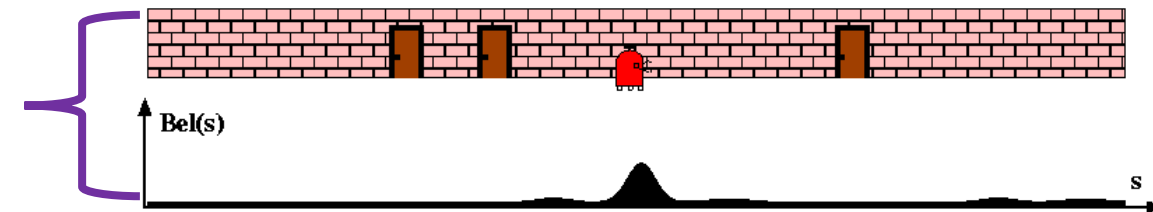
Execute an action – move to the right by about a meter... probability mass “spreads out”



Take another measurement. It seems we're in front of a door again (red). Given what we believed before about position, the most likely place now is the second door.



Execute an action – move to the right by about a meter... probability mass “spreads out”



Markov Localization

Markov localization approximates the state space using a **discrete grid**.

At time t , the value in the grid cell x^{ij} represent the probability that $x_t = x^{ij}$.

At time $t + 1$, every grid cell updates its probability value based on:

- Prediction from the motion model
- Observation from sensors

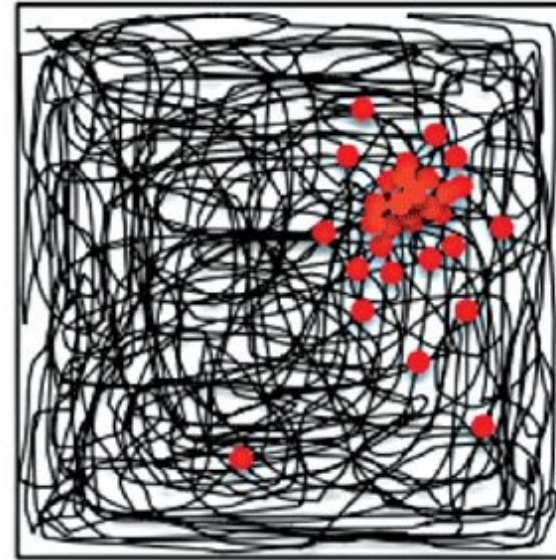
This is a grid-cell-centric view of probability updating.

Instead of keeping track of moving probability mass (e.g., particles), each grid cell pays attention to the probability mass that arrives to its specific location.

Place and grid cells

- Nobel Prize 2014

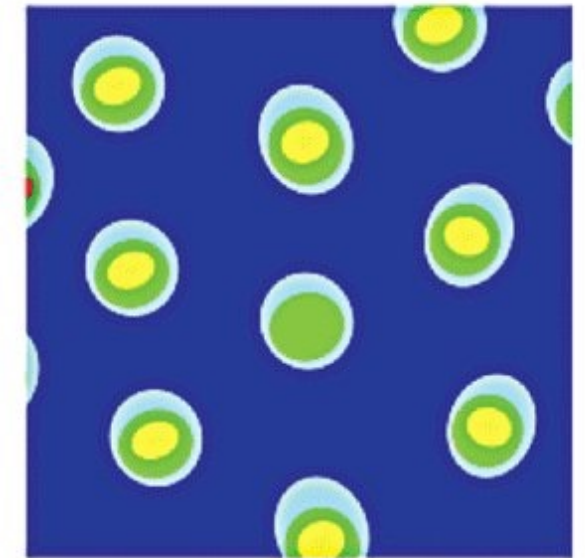
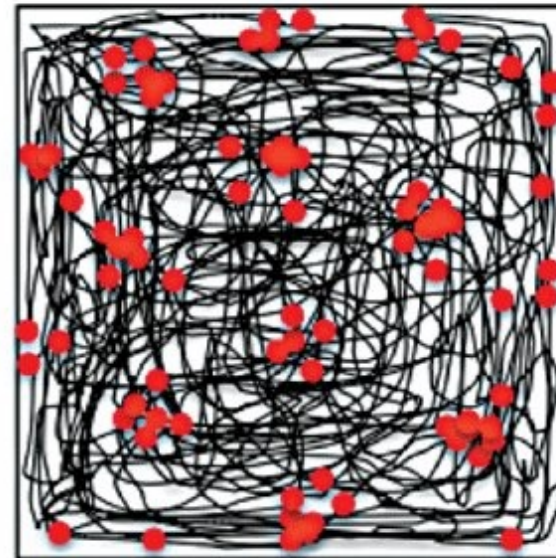
1. Place cell



B. Rate maps



2. Grid cell



Markov Localization

- Perception (or sensing) model: represents likelihood that robot senses a particular reading at a particular position.

$$P(x) \propto L(x; z)P(x)$$

Likelihood of position x given the measurement z , times the prior probability the robot is in position x

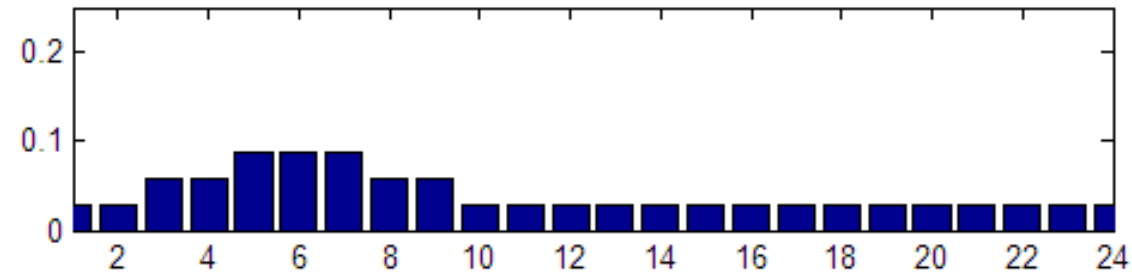
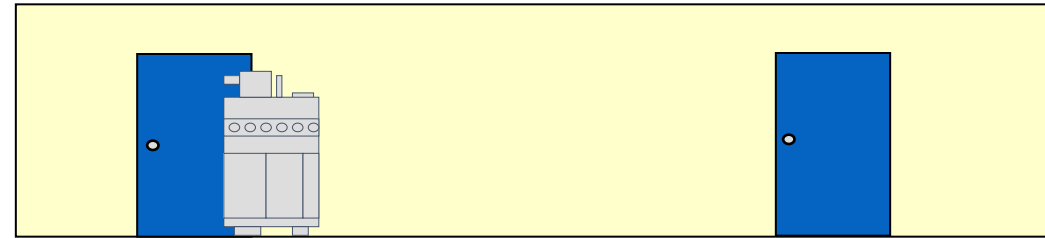
- Action (or motion) model: represents movements of robot

$$P(x) = \sum P(x|u, x')P(x')$$

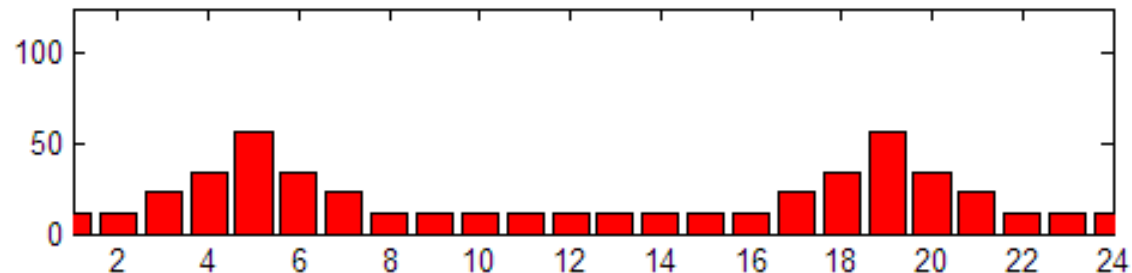
Probability that action u from position x' moves the robot to position x , weighted by the probability that the robot is in position x' , summed over all possible x' where the robot might have been.

➤ **Perform these computations at every grid cell, at each time t .**

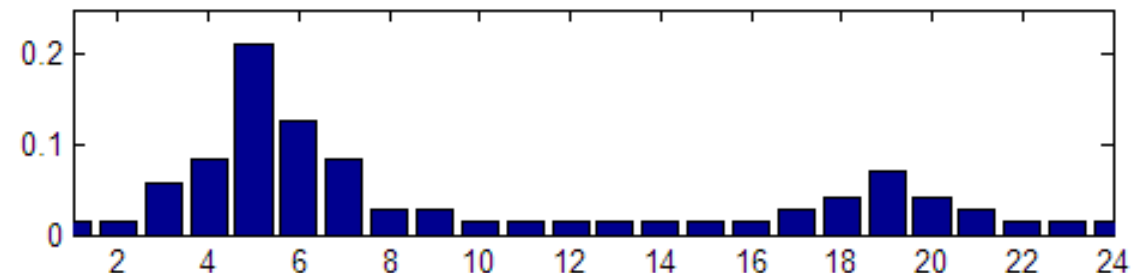
Markov Localization: a 1D Example



Prediction
 $P(S)$



Likelihood
 $L(S;o)$

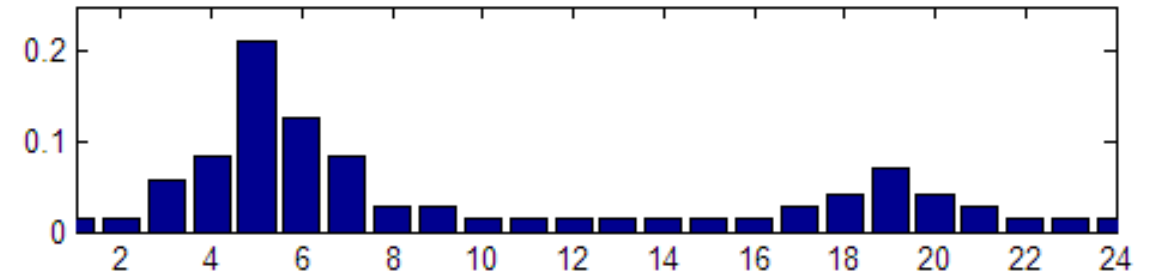
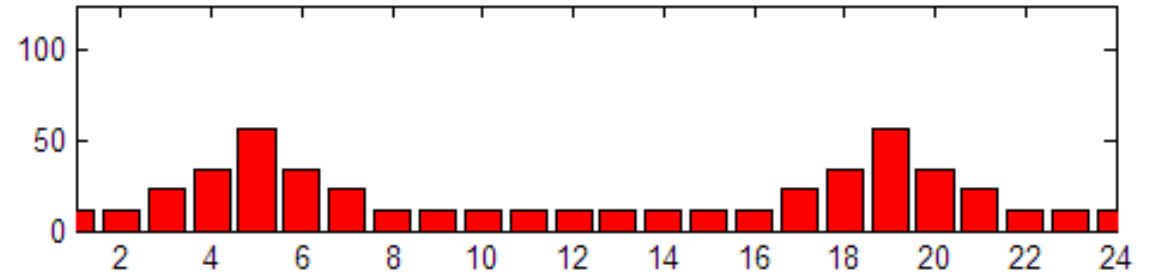
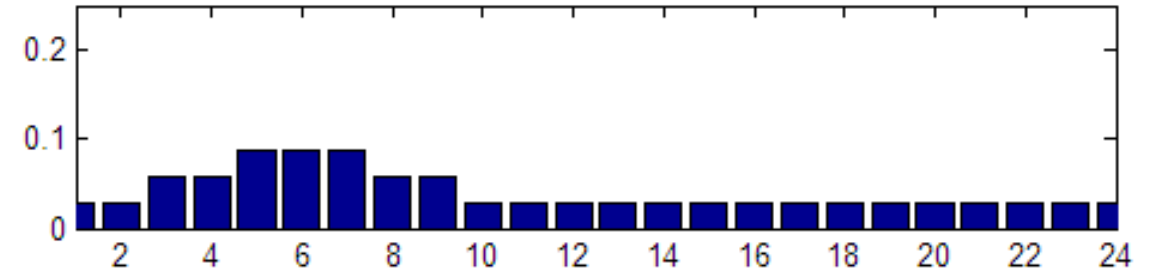
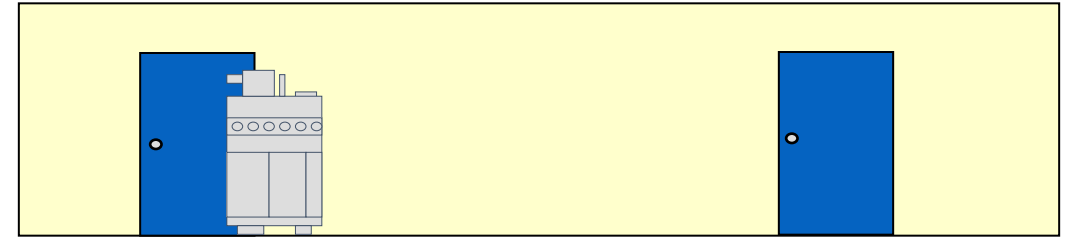


Posterior
 $P(S|o)$

Each bin in the histogram is updated in each step.

Group Activity: how do we implement Markov Loc ?

$$P(X_k | \mathcal{Z}^{k-1}, \mathcal{U}^k) = \boxed{\text{????}}$$
$$P(X_k | \mathcal{Z}^k, \mathcal{U}^k) \propto \boxed{\text{????}}$$



Implementing Markov Localization

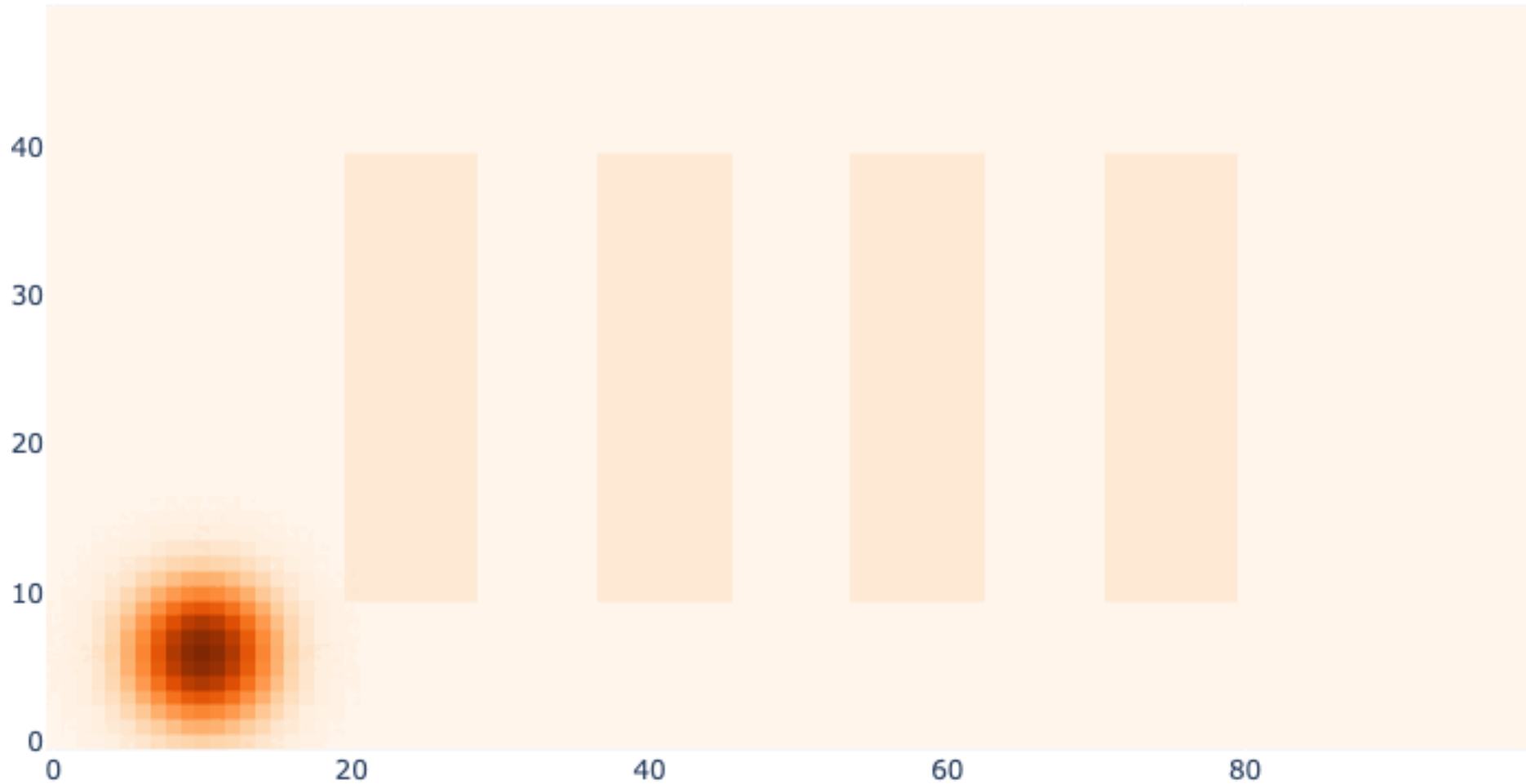
$$P(X_k | \mathcal{Z}^{k-1}, \mathcal{U}^k) = \sum_{x_{k-1}} P(X_k | x_{k-1}, u_{k-1}) P(x_{k-1} | \mathcal{Z}^{k-1}, \mathcal{U}^{k-1}).$$

$$P(X_k | \mathcal{Z}^k, \mathcal{U}^k) \propto L(X_k; z_k) P(X_k | \mathcal{Z}^{k-1}, \mathcal{U}^k).$$

- In practice, many grid cells have very small probability values.
- We can speed computation by ignoring these cells, with little risk of going astray in our state estimation.
- If we care about the robot's orientation, then we need to add a θ dimension to our grid.

Recall: propagation *without* sensor

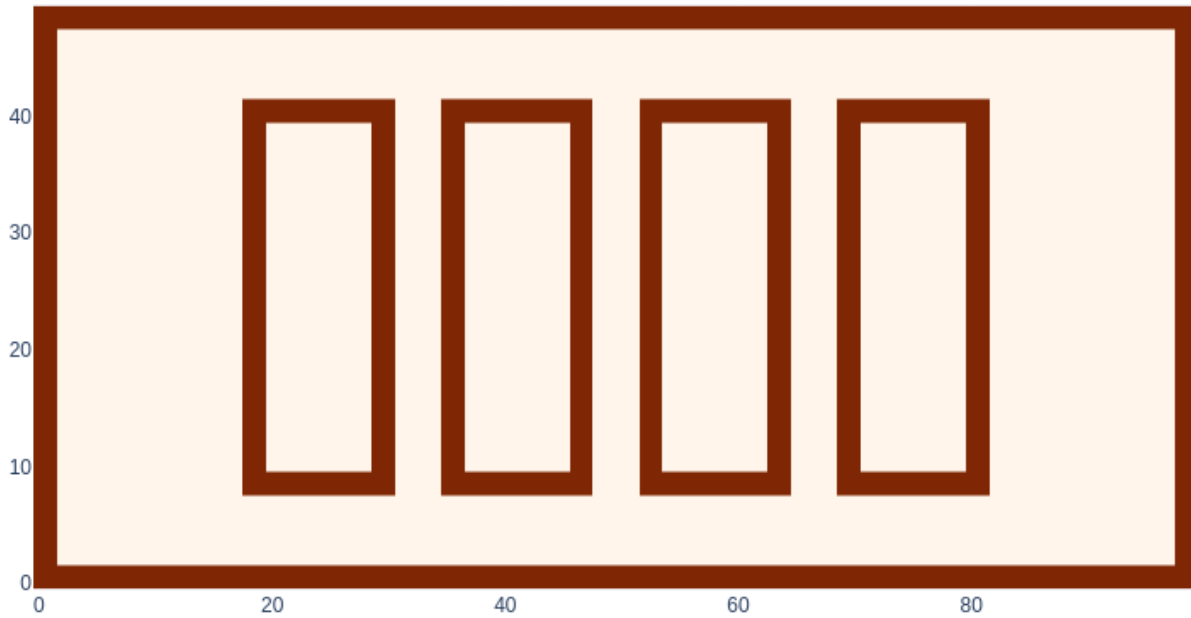
- Uncertainty grows without bounds:



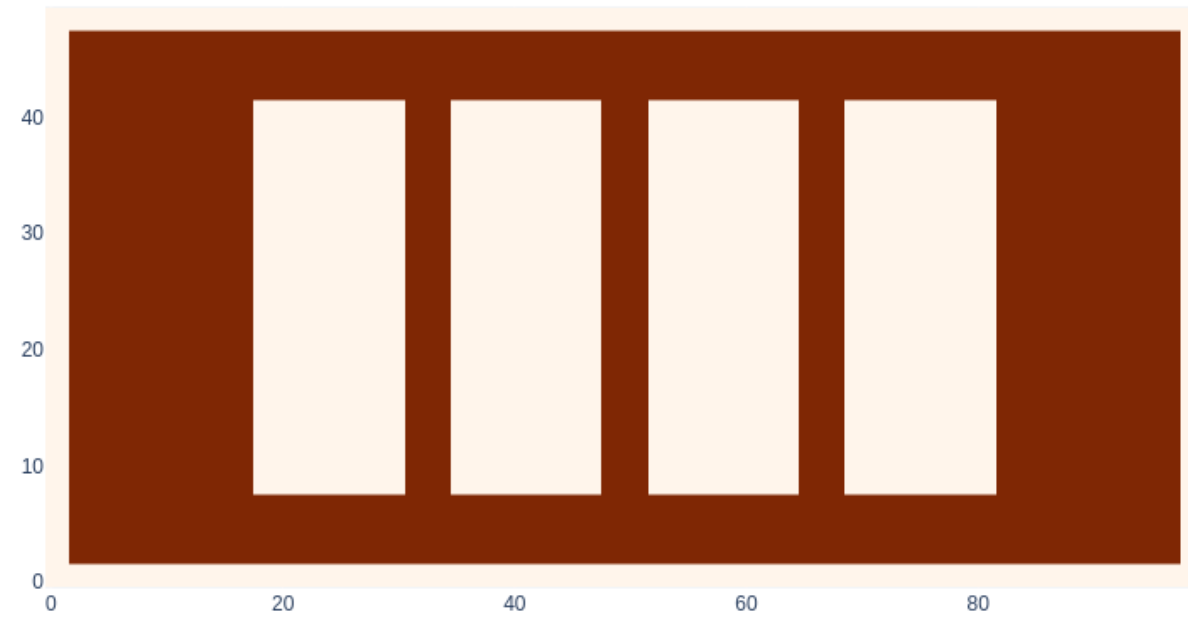
Recall: Likelihood images for the Proximity Sensor

- We can plot the likelihood for for each possible value of z_k .

$$\mathcal{L}(x_k; z_k = ON) = \begin{cases} 1 & d(x_k) \leq d_0 \\ 0 & \text{otherwise} \end{cases}$$

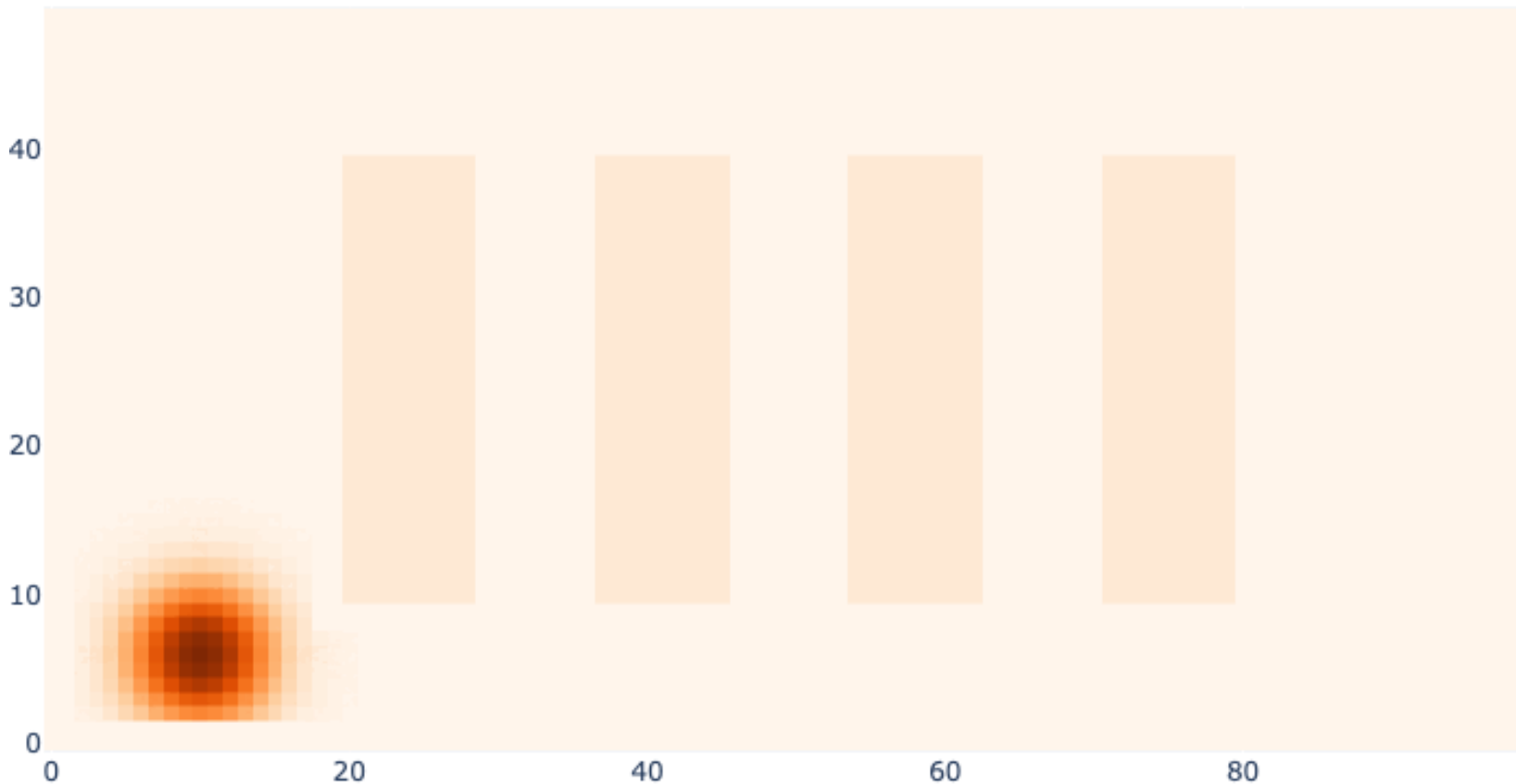


$$\mathcal{L}(x_k; z_k = OFF) = \begin{cases} 0 & d(x_k) \leq d_0 \\ 1 & \text{otherwise} \end{cases}$$



Markov Localization

- The robot moves through the world, and each cell in the grid updates its probability estimate after each motion model step, by multiplying with the likelihood image.



Grid and particle representations

Grid localization

Store probability in fixed cells. Update the probability assigned to each cell.

Particle localization

Store sampled states and weights. Concentrate samples where the belief has support.

Both approximate Bayes filtering. What changes is how we represent and compute the belief.

The Particle Filter

Particle filters represent a probability density function as a set of weighted samples.

The weighted samples are

1. Pushed through the motion model (including uncertainty)
 2. Reweighted based on sensor measurements (using the sensor model)
 3. Resampled using the new weights to define a probability distribution on the sample set.
- The approach is easy to implement and has low computational overhead.
 - Complexity does not grow exponentially with dimension of the state space.

Two localization problems

- “Global” localization
 - Figure out where the robot is, **but we don’t know where the robot started**
 - Sometimes called the “kidnapped robot problem”
- “Position tracking”
 - Figure out where the robot is, given that **we know where the robot started**

➤ To solve these problems at time t , we estimate

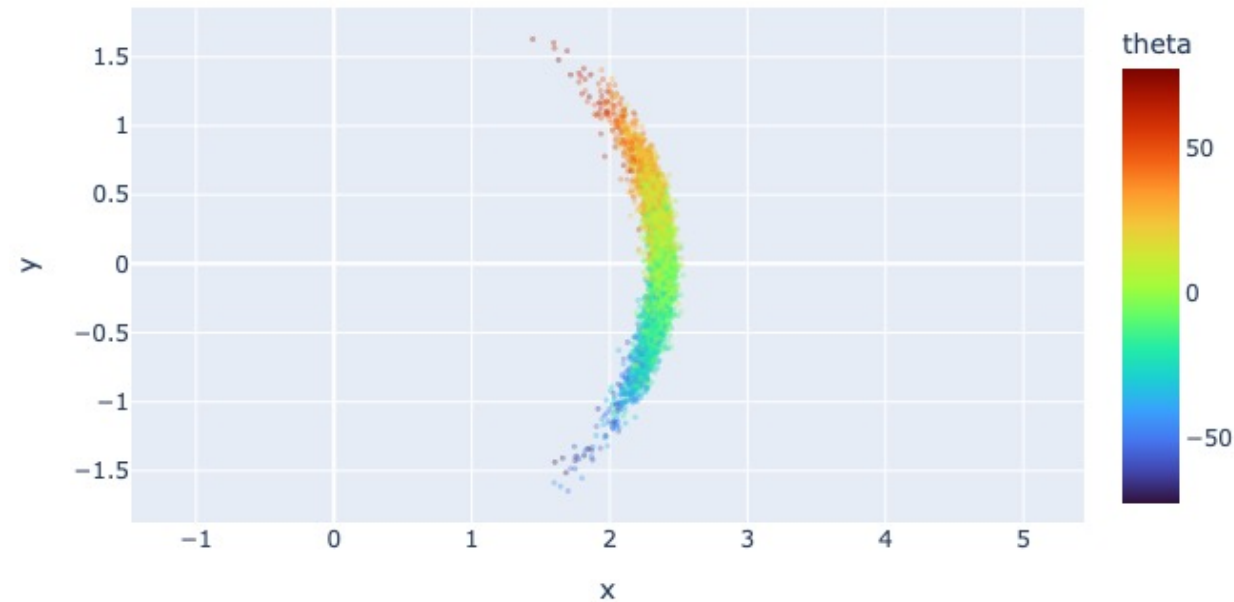
$$Bel(x_t) = P(x_t | u_1, z_1, u_2, \dots, z_t)$$

➤ **The hard part: it’s not feasible to exactly calculate or represent $Bel(x_t)$.**

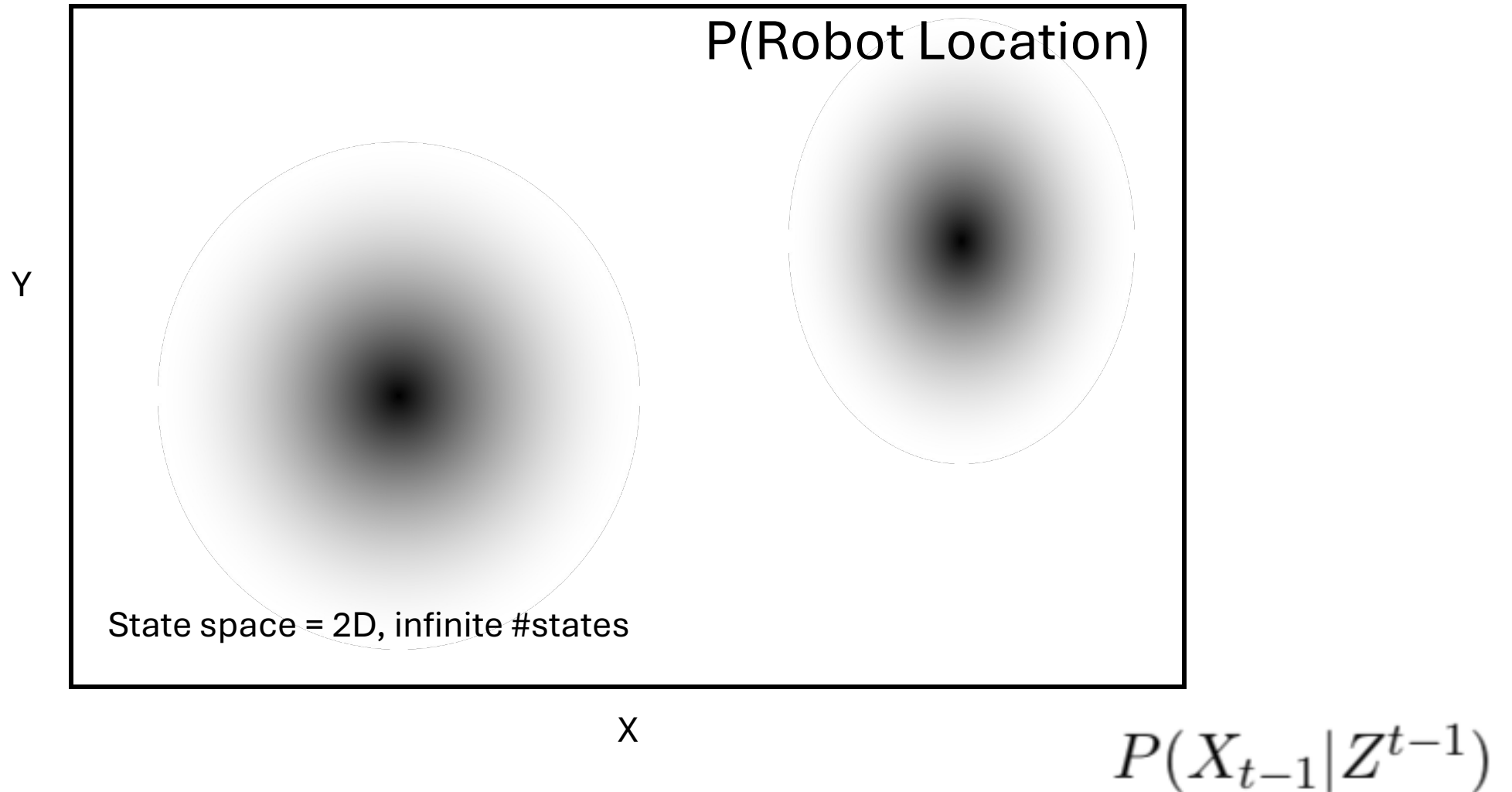
Sampling to Approximate Densities

- Densities can become arbitrarily complex, even with Gaussian noise.
- One issue is nonlinear measurement and noise models.
- A second issue is the curse of dimensionality (for grid-based methods).
- One way out: **sampling!**

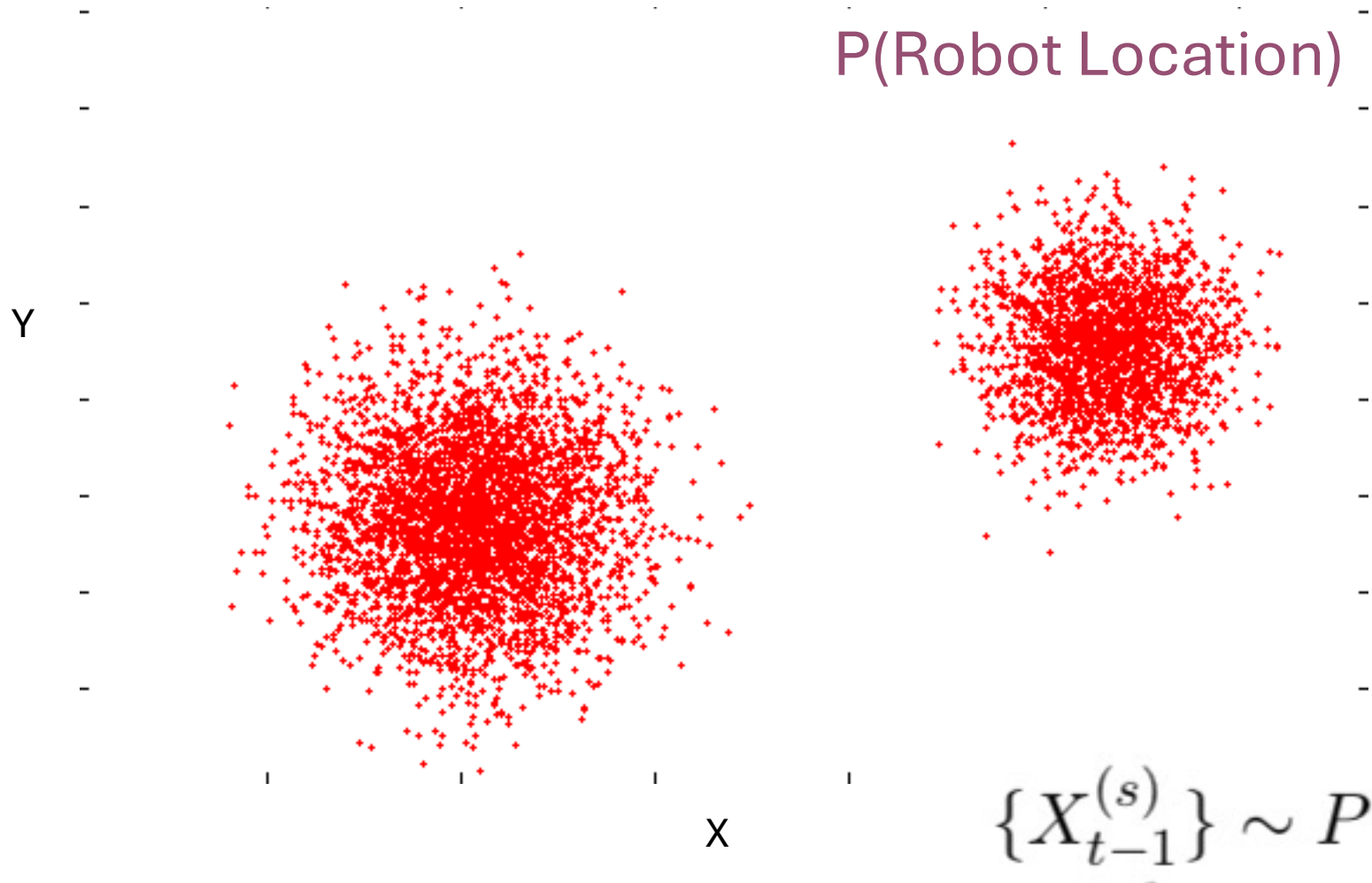
Pose2 random-walk samples → pronounced banana distribution



Probability of Robot Location



Sampling as Representation



Particle Filter

- Represent $p(x)$ by set of N weighted, random samples, called *particles*, of the form: $\langle (x_i, y_i), w_i \rangle$

(x_i, y_i) represents robot's pose

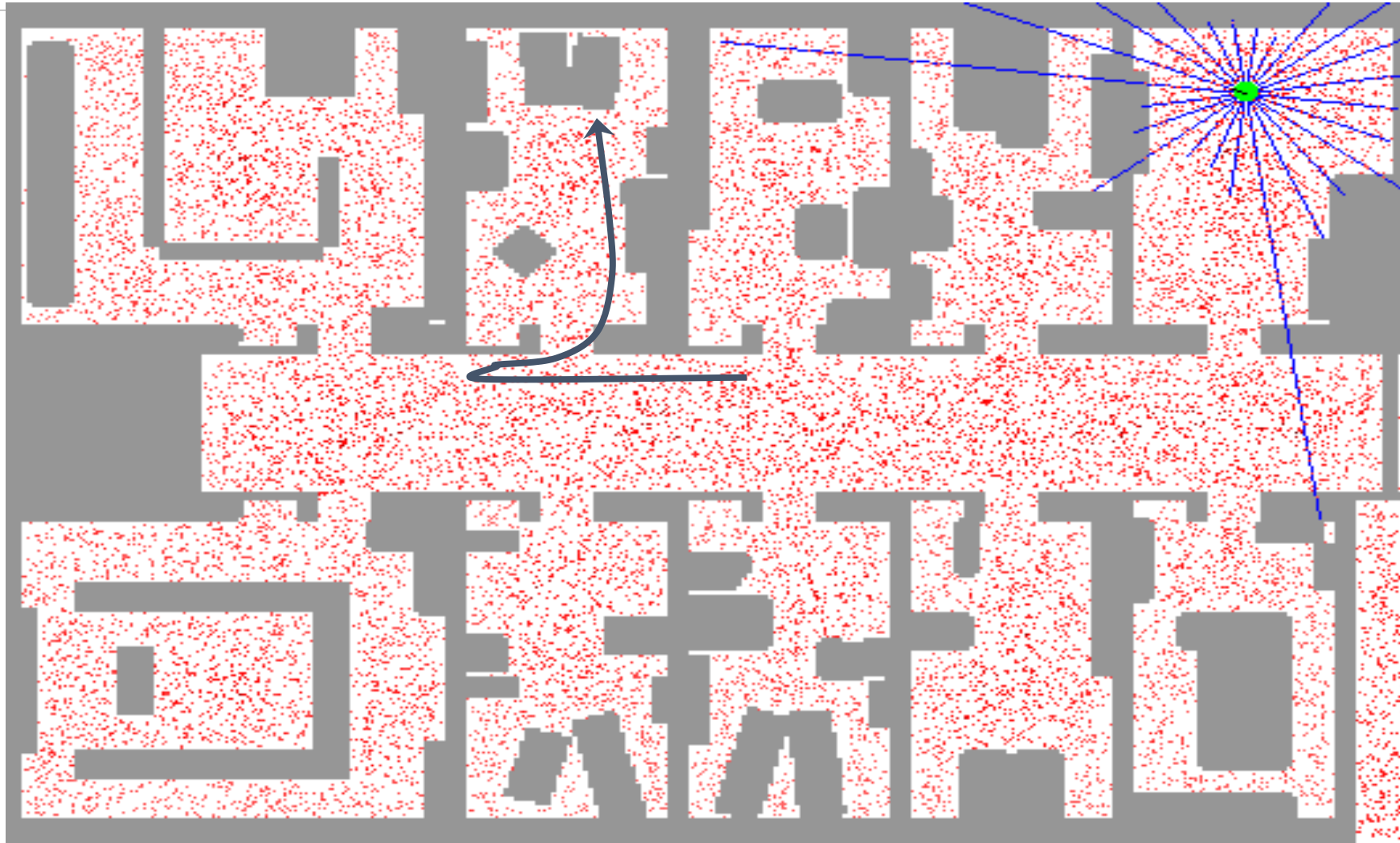
w_i represents a weight, where $\sum w_i = 1$

- A.K.A. **Monte Carlo Localization (MCL)**
 - Refers to techniques that are stochastic (random / non-deterministic)
 - Used in many modeling and simulation approaches

Sampling Advantages

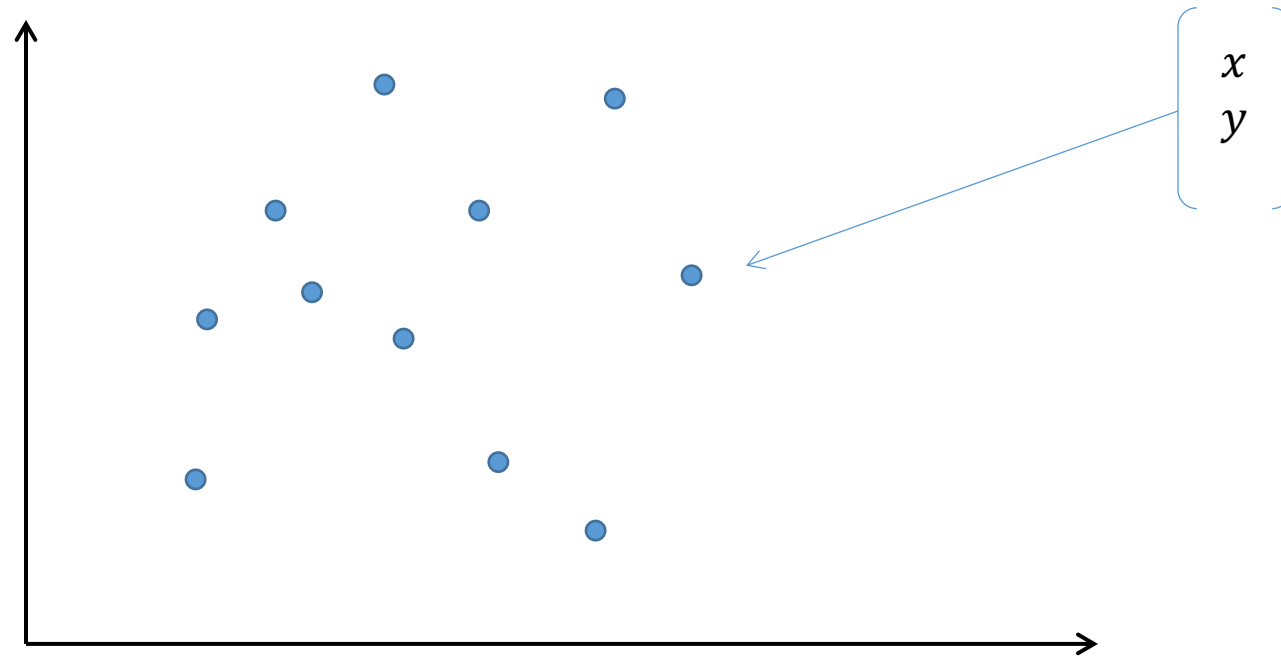
- Arbitrary densities
 - Memory = $O(\text{\#samples})$
 - Only in “Typical Set”
 - Great visualization tool !
-
- minus: Approximate

Particle Filter Localization (using sonar)



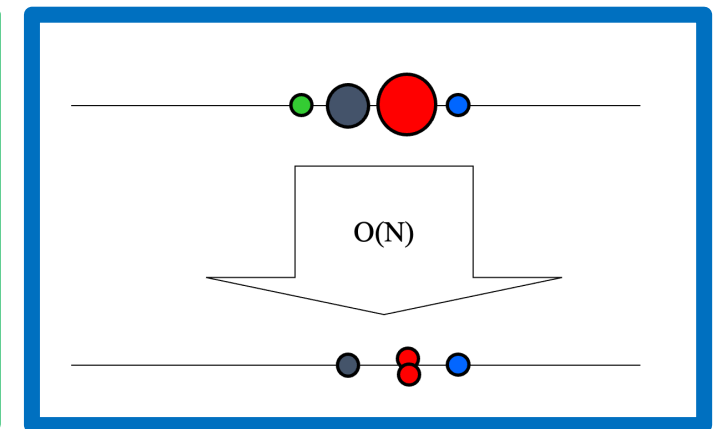
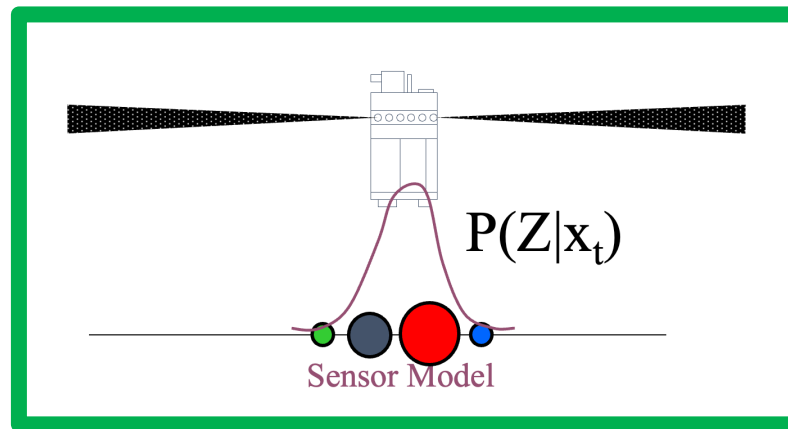
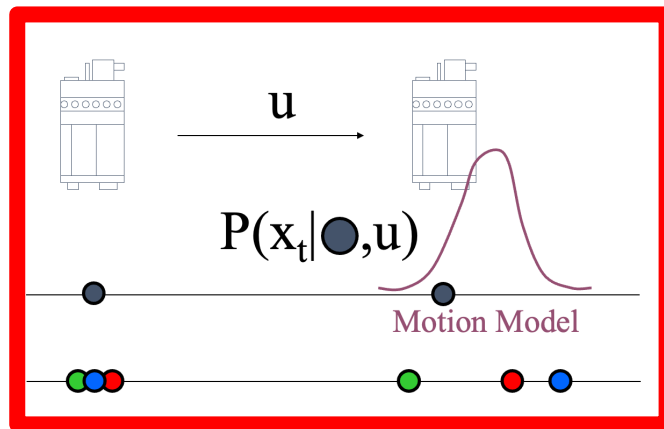
Particles

- You can think of each particle as a guess about where the robot might be

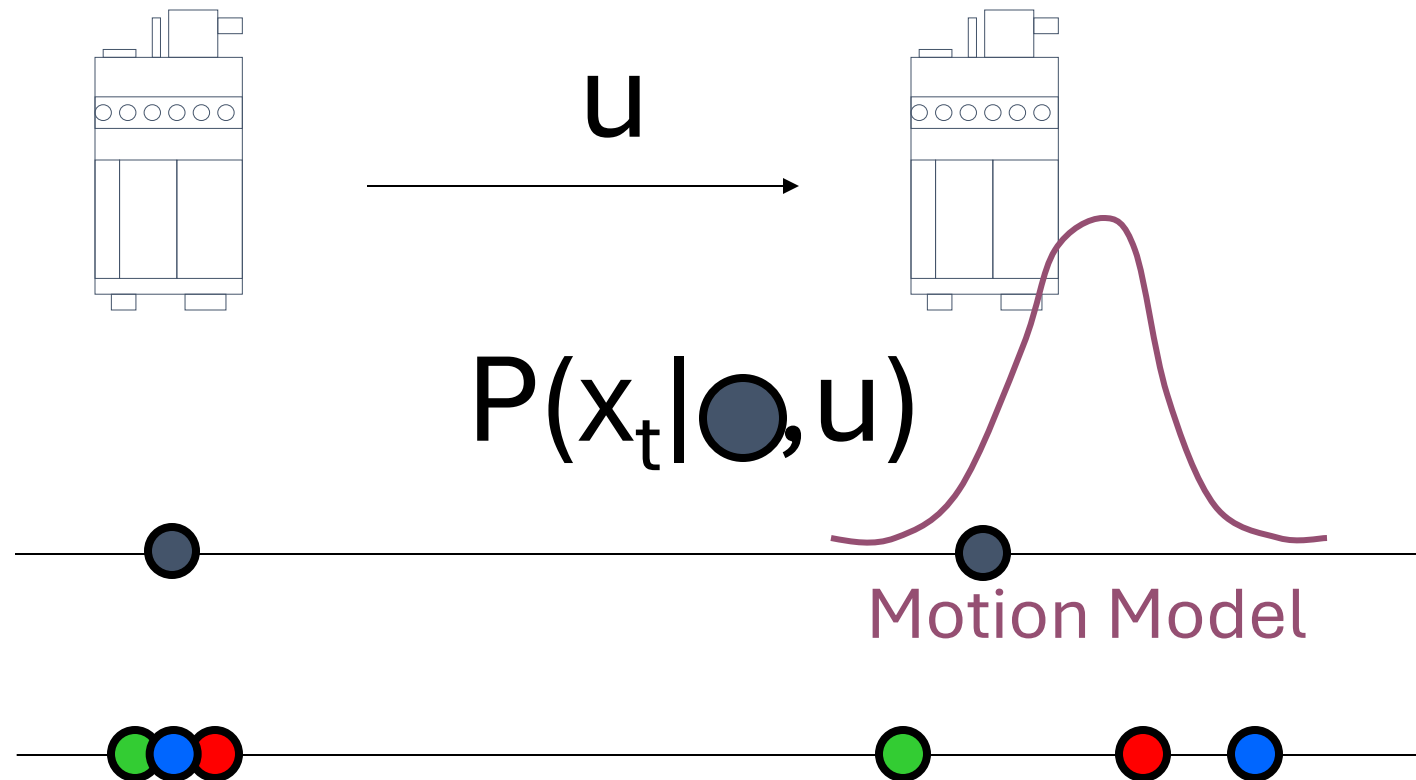
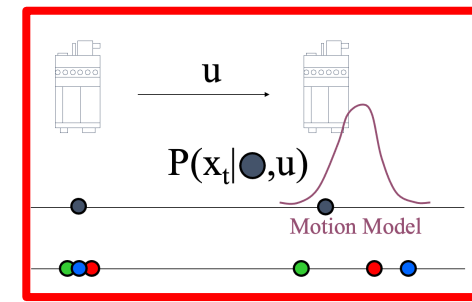


Monte Carlo Localization steps

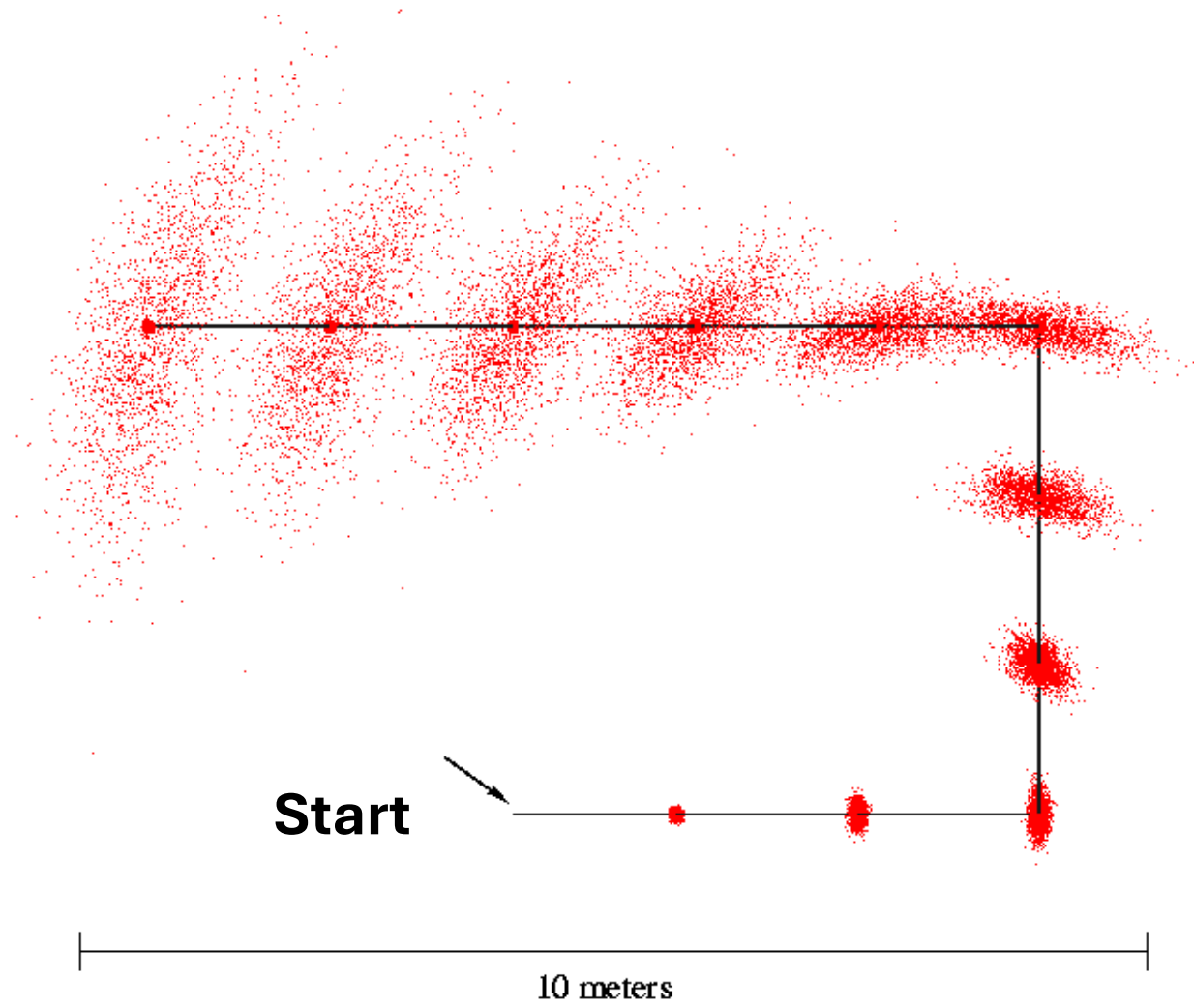
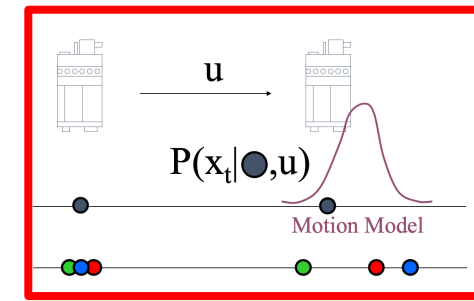
- Prediction Phase
- Measurement Phase
- Resampling Step



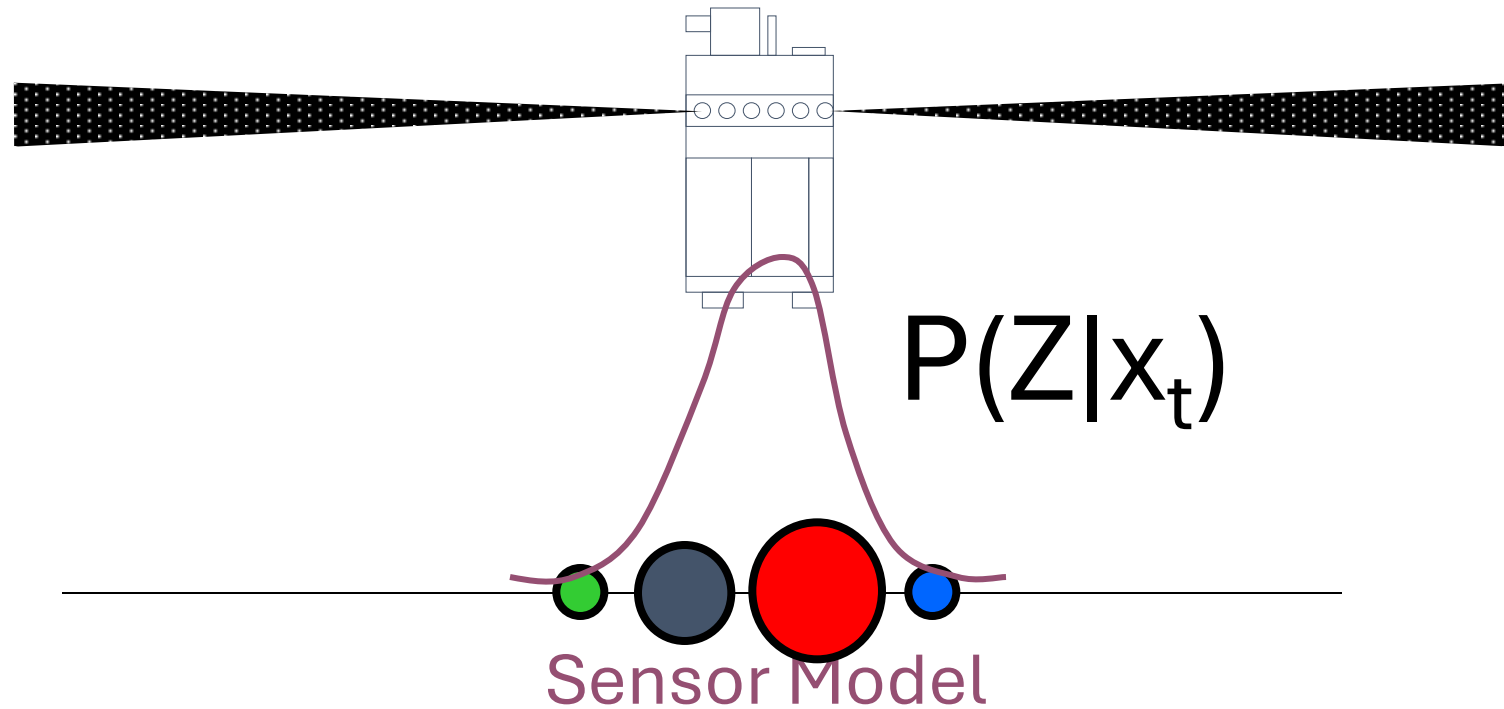
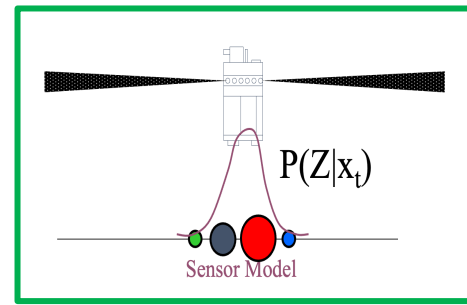
1. Prediction Phase



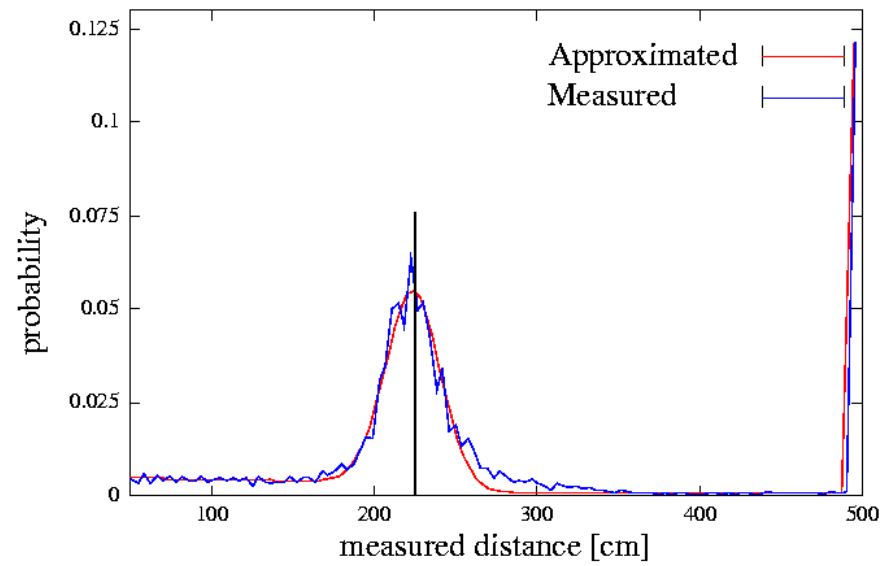
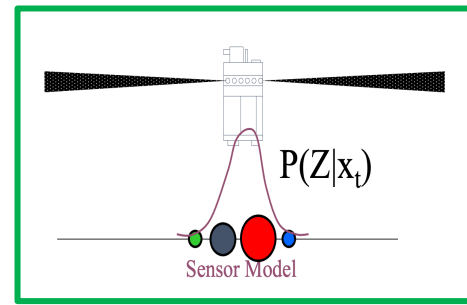
Motion Model for a Car-Like Robot



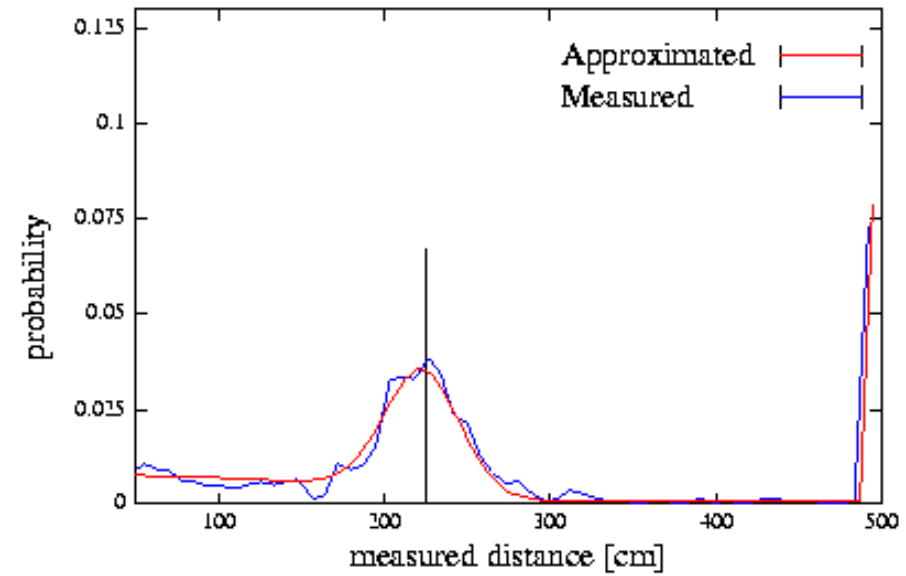
2. Measurement Phase



Sensor Model

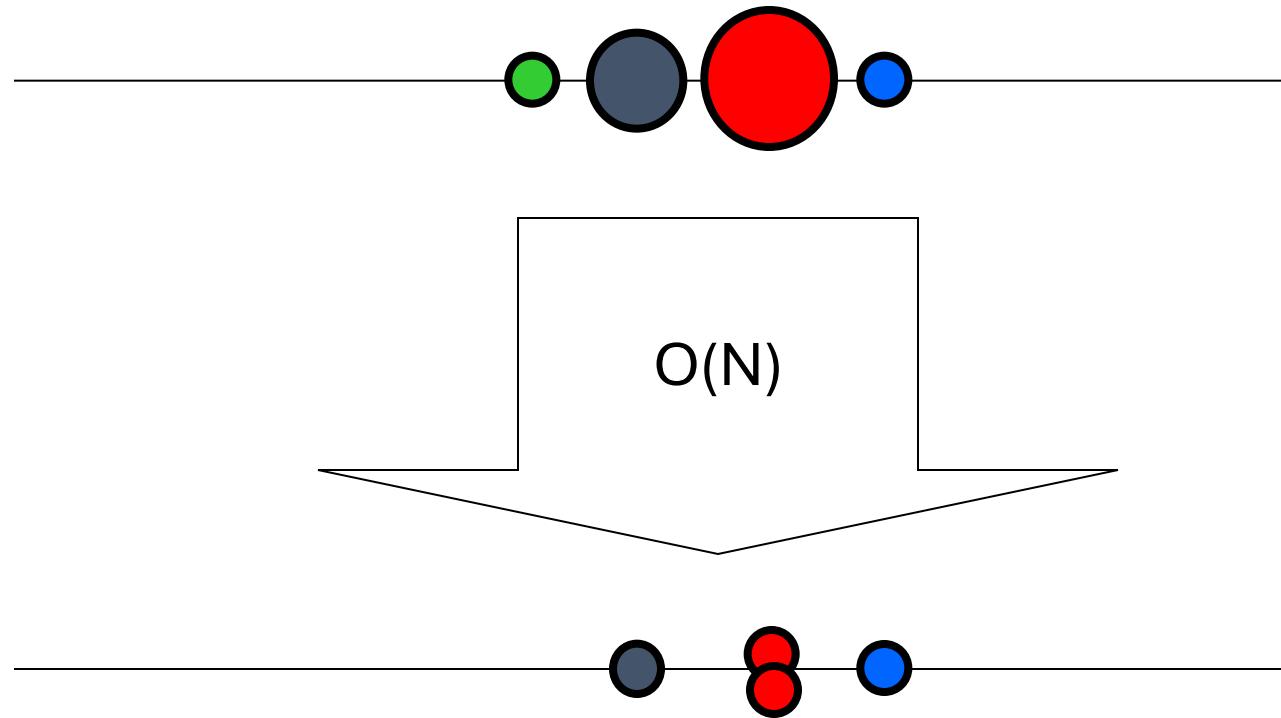


Laser sensor

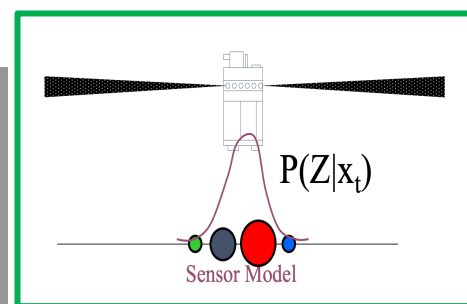
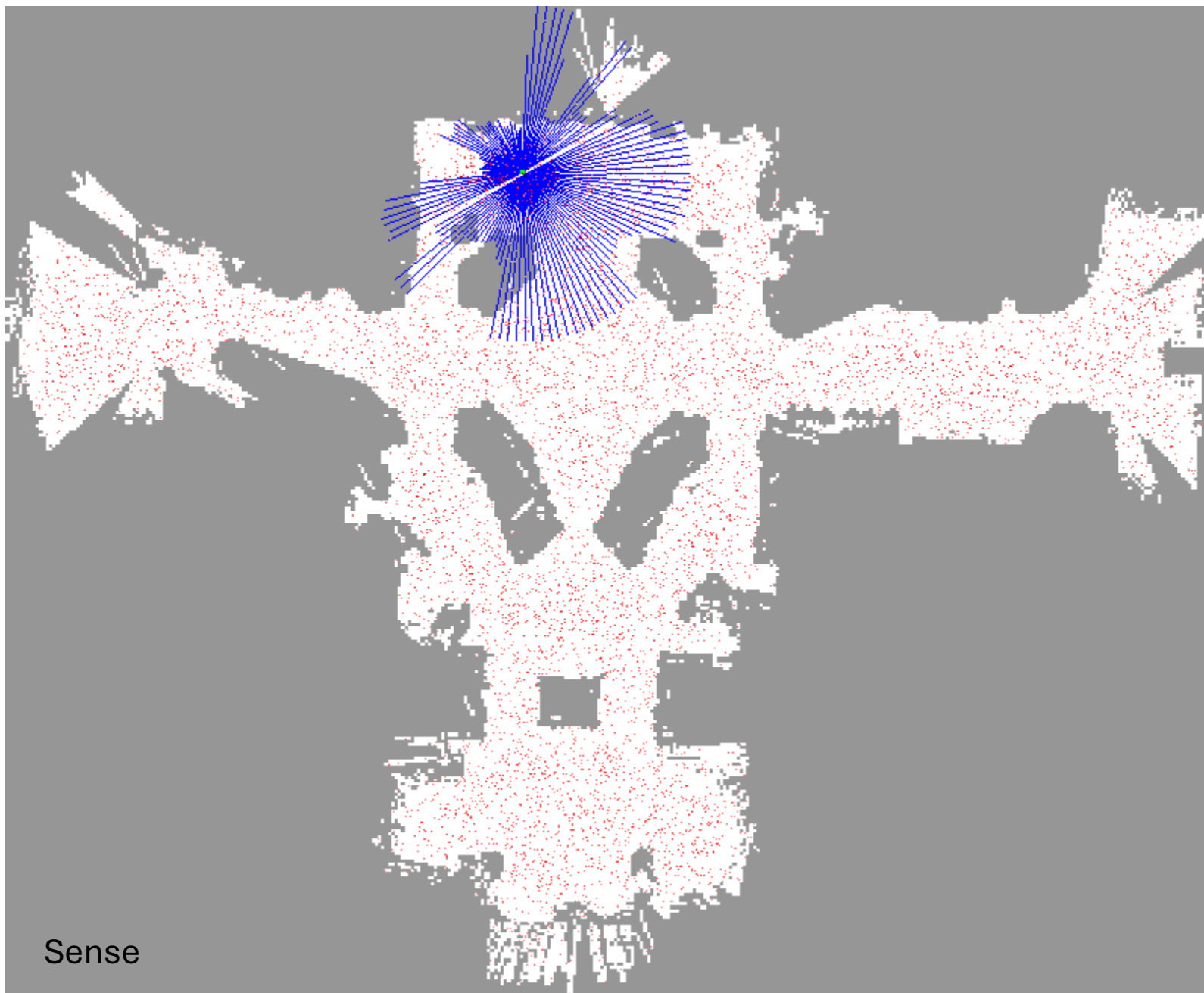


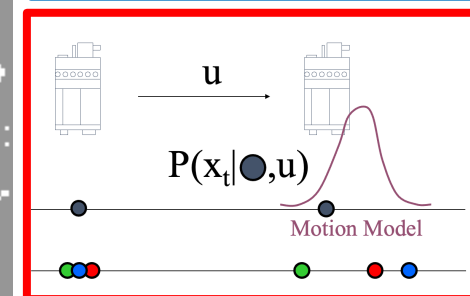
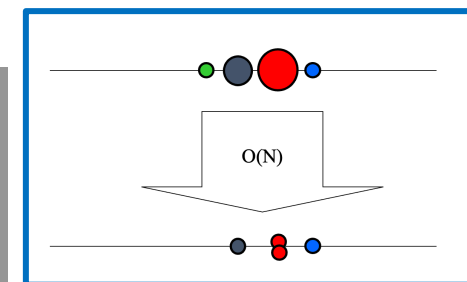
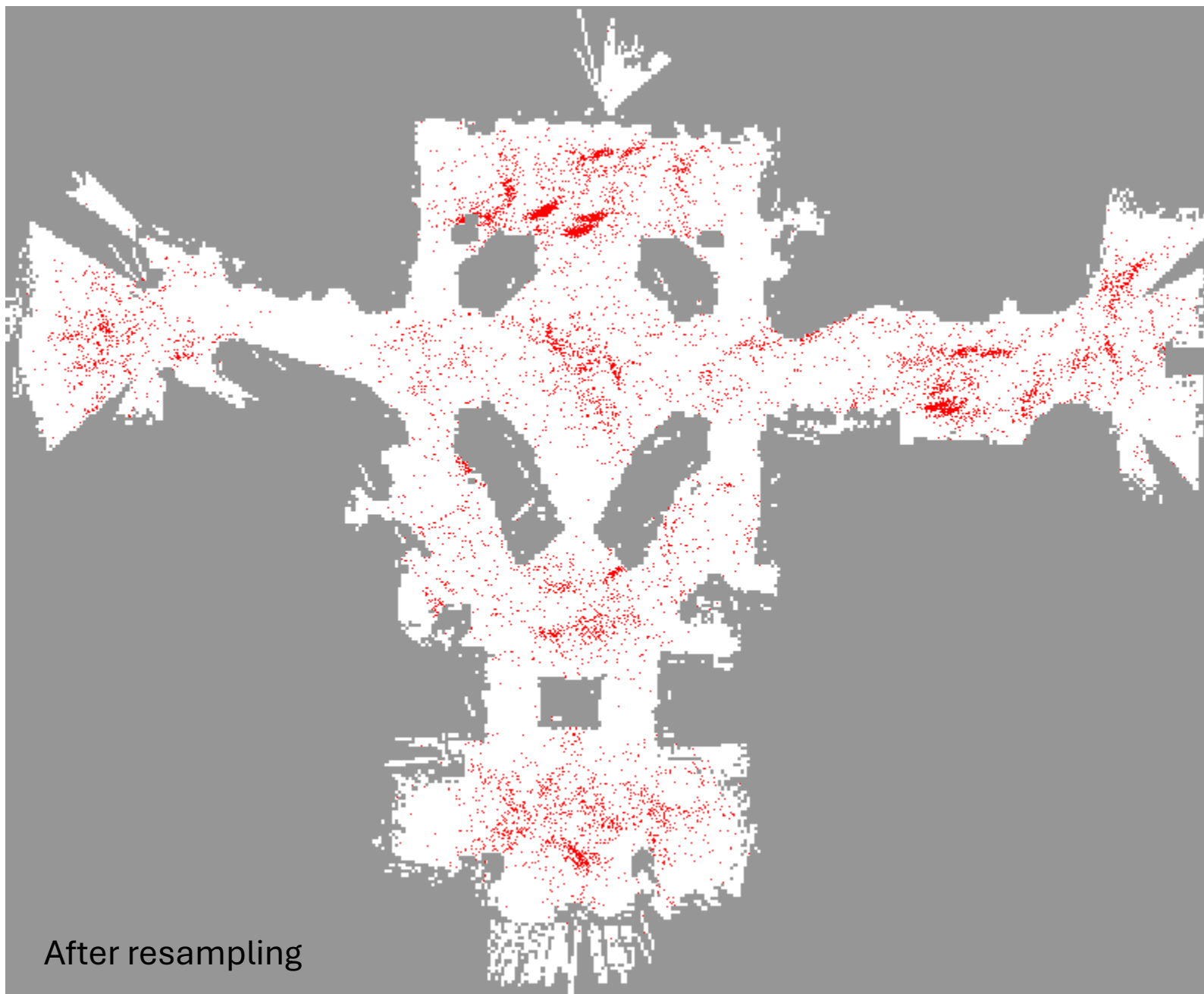
Sonar sensor

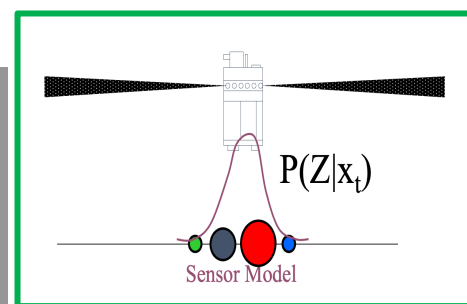
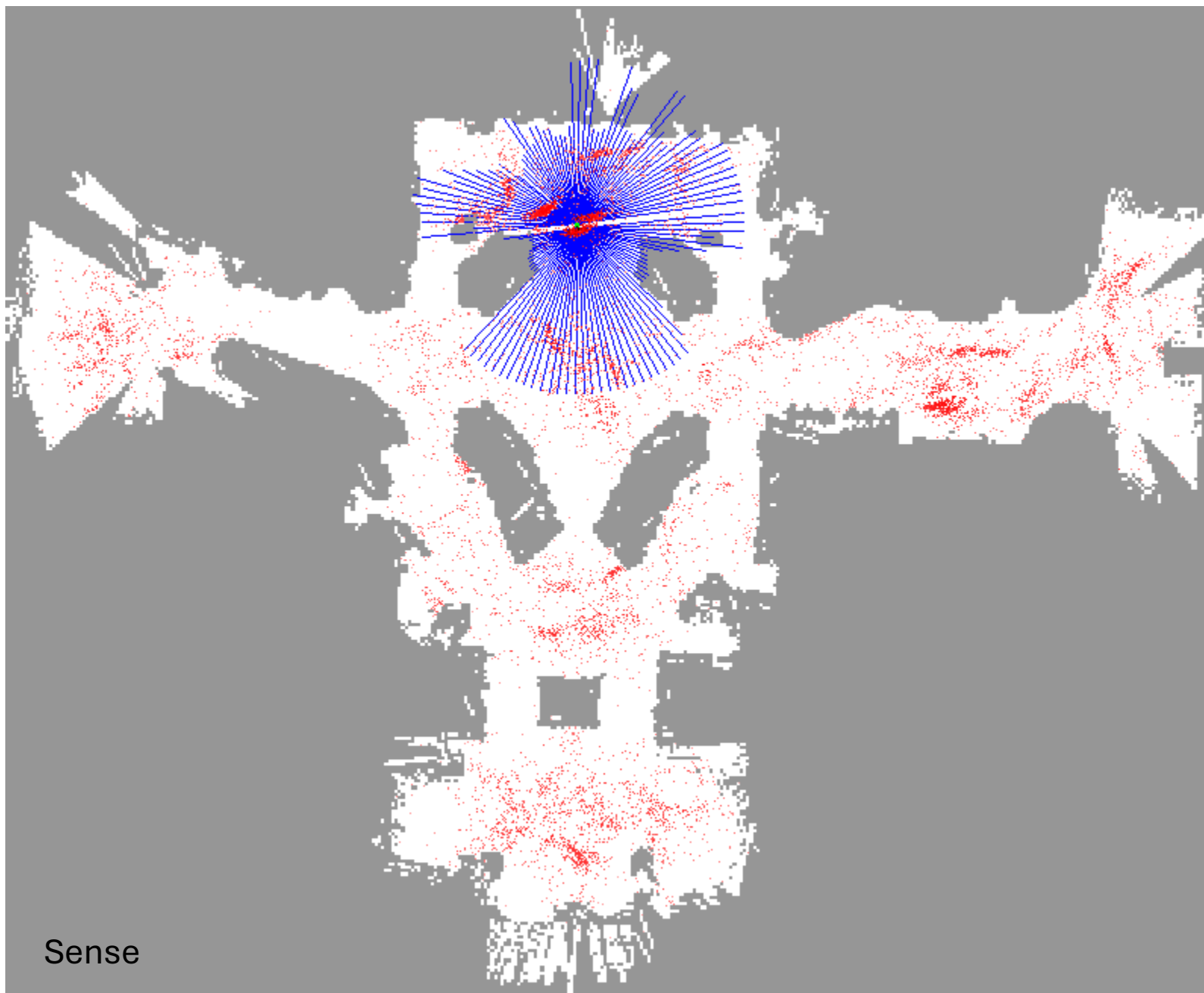
3. Resampling Step

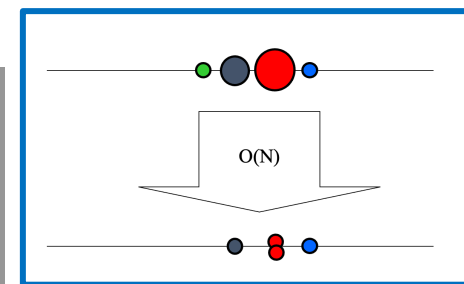
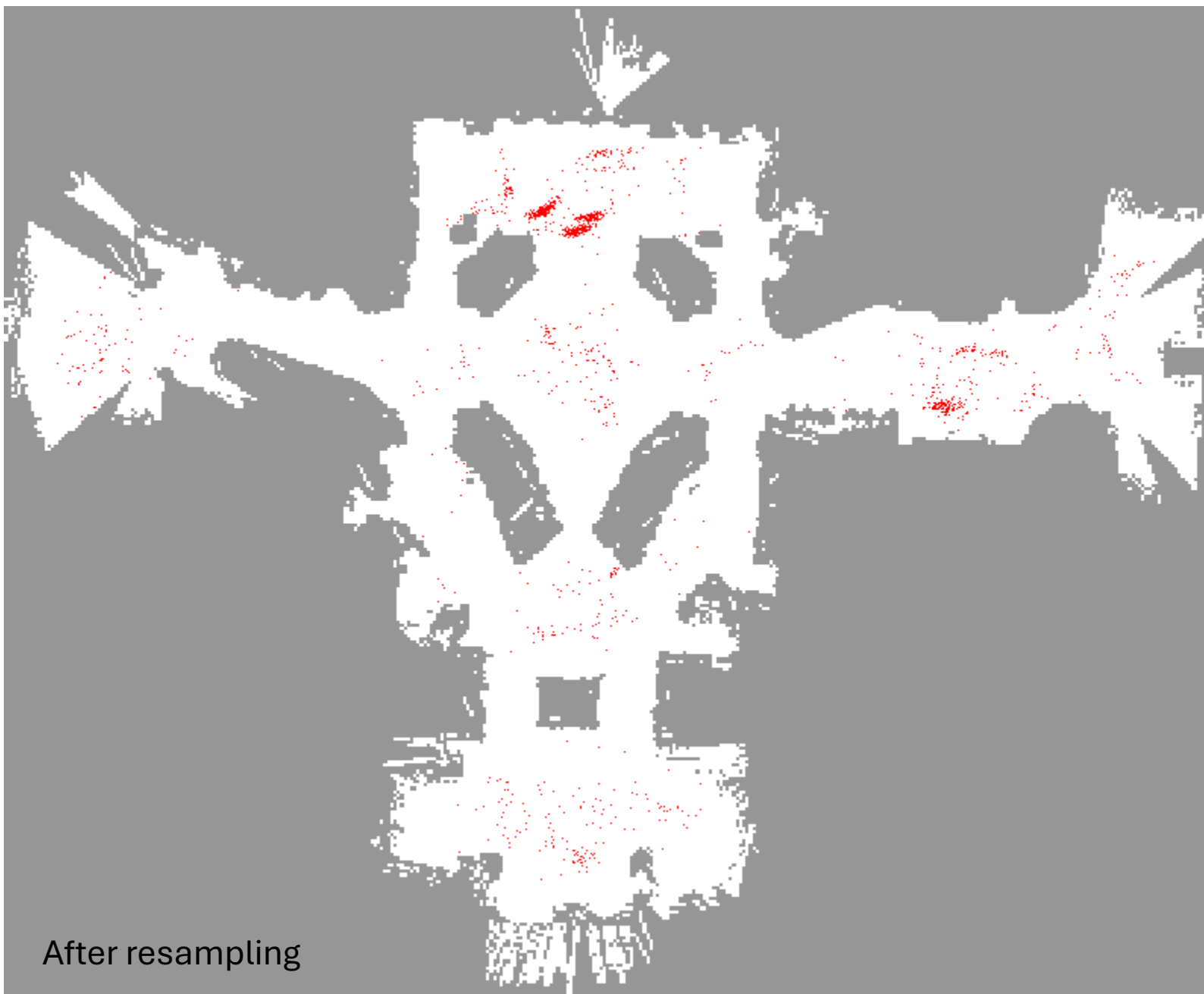


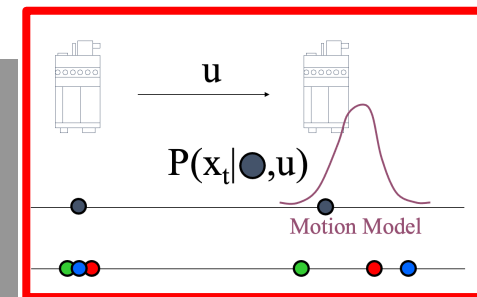


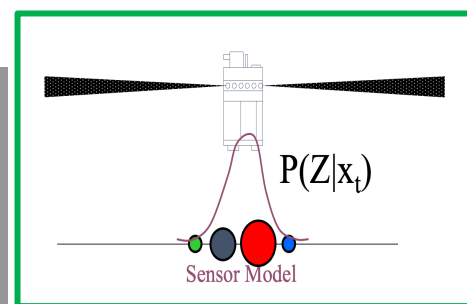
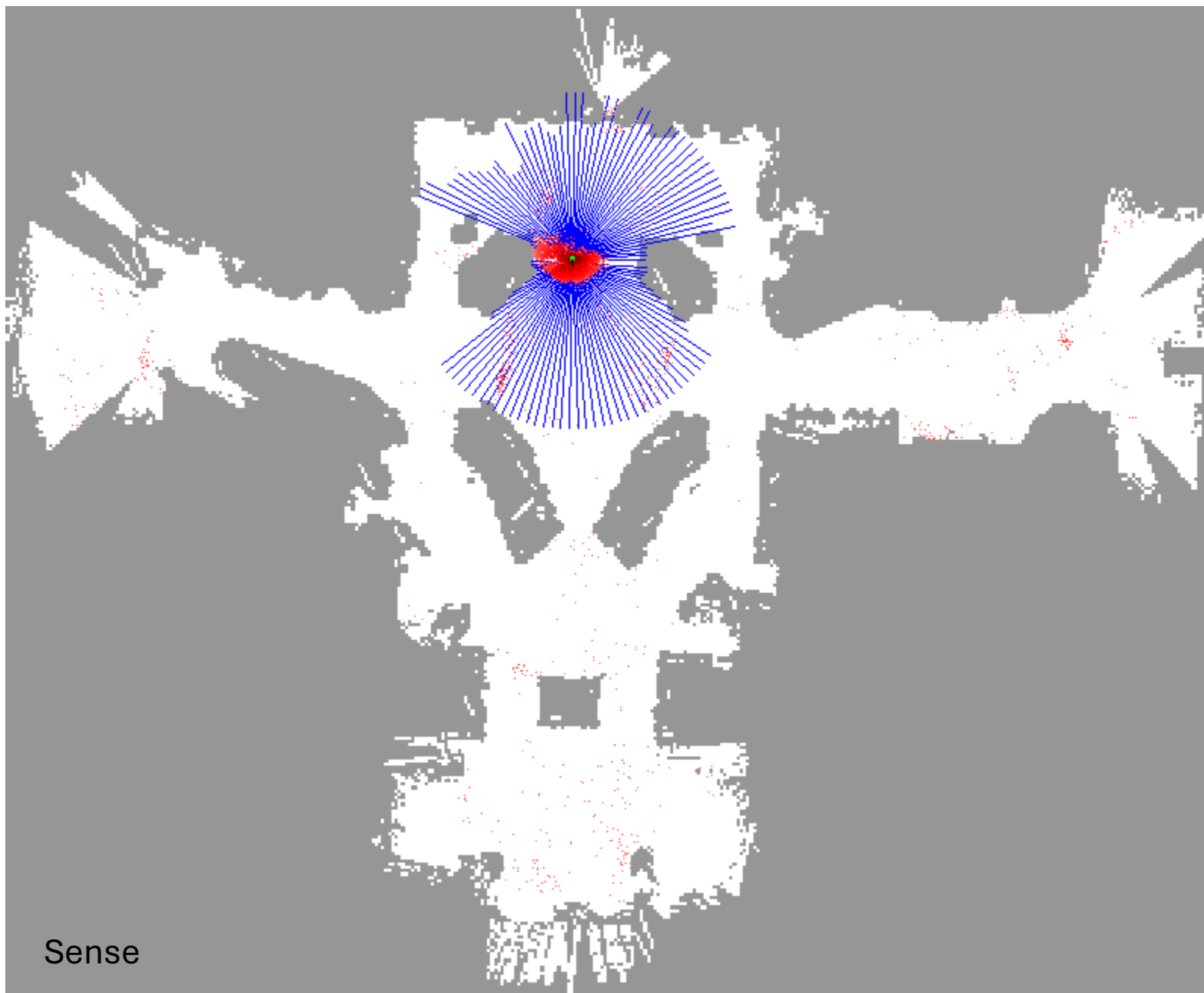


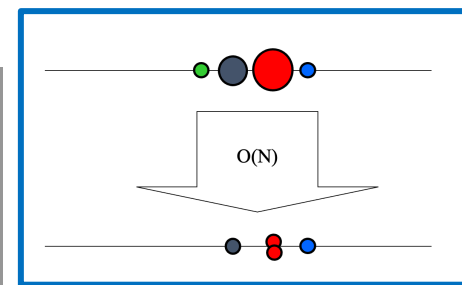
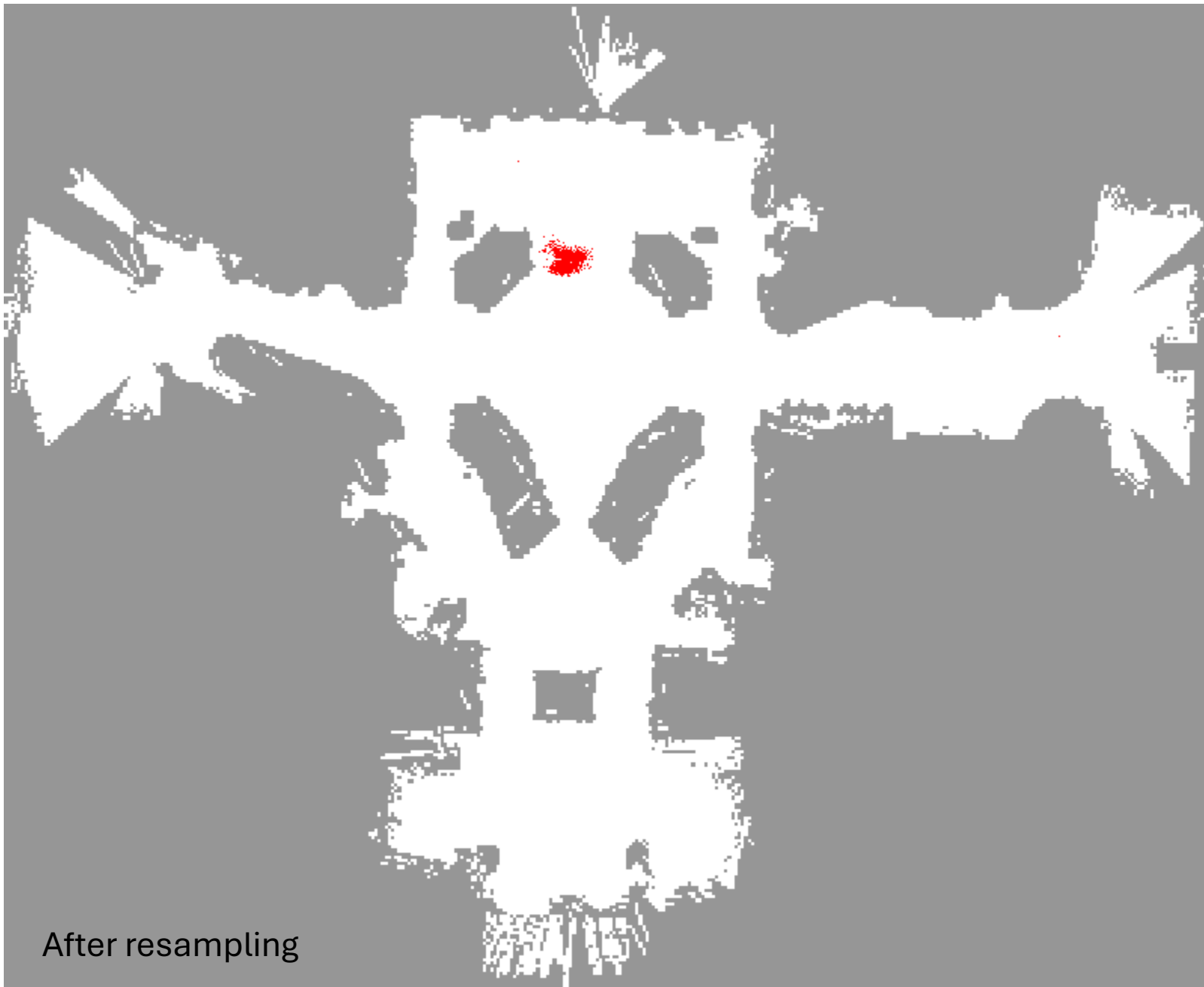


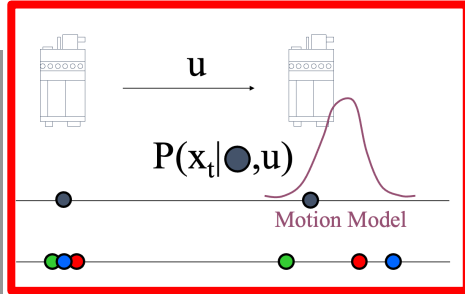
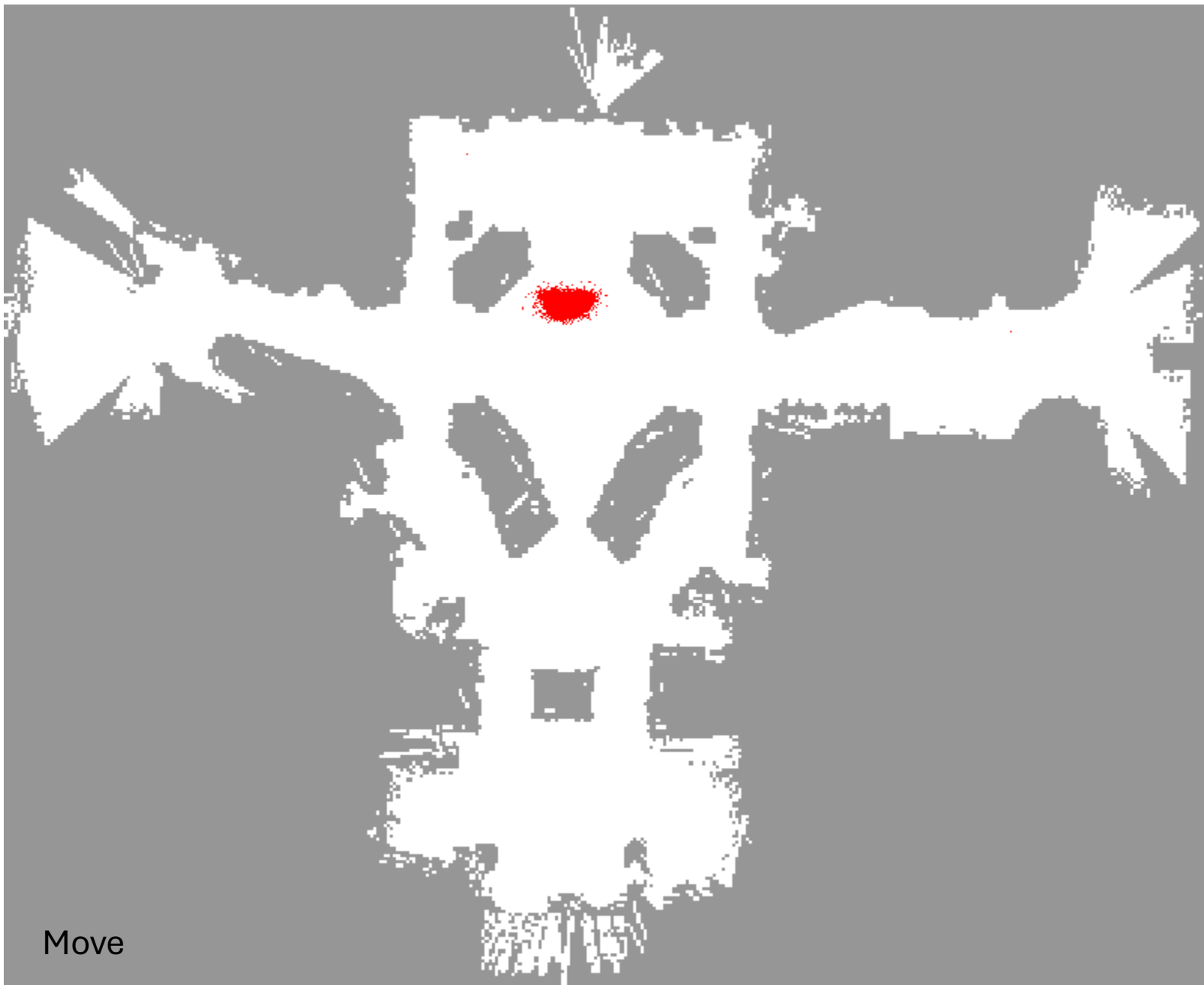


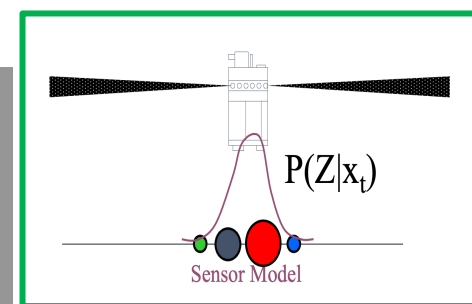
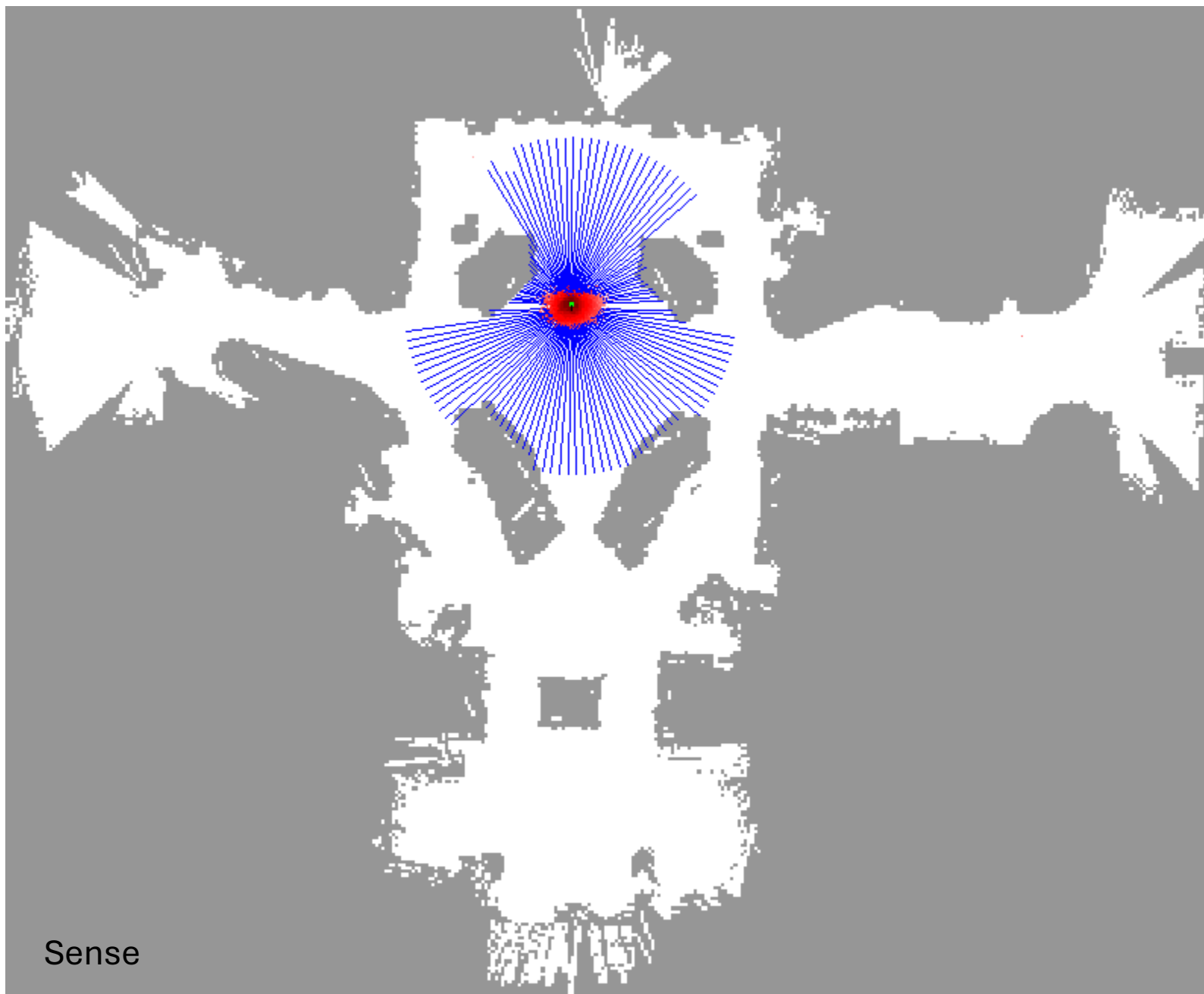


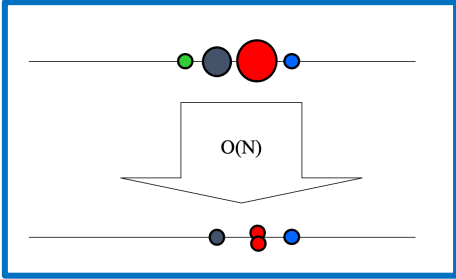
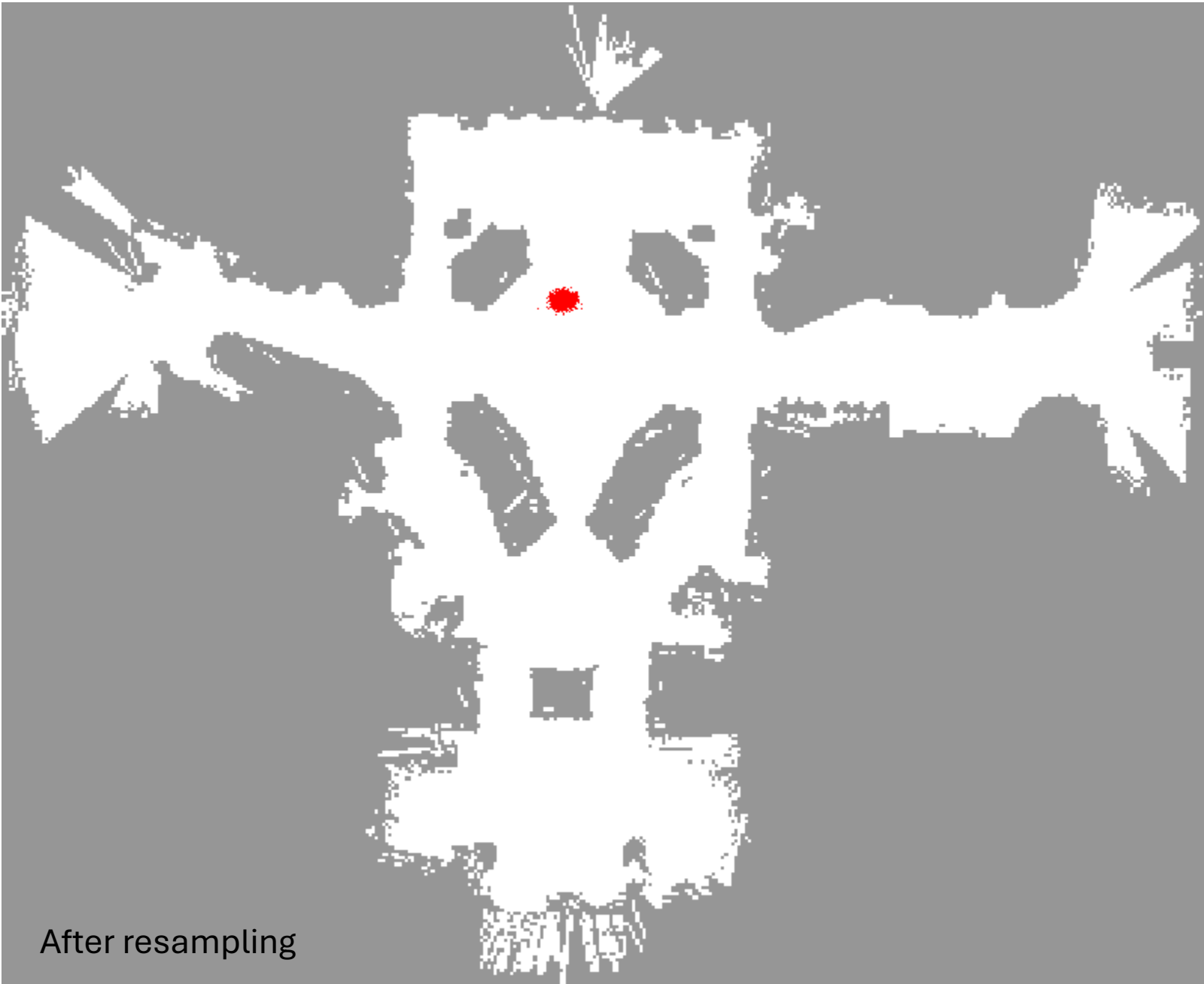


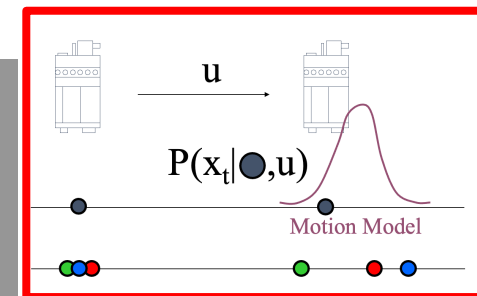
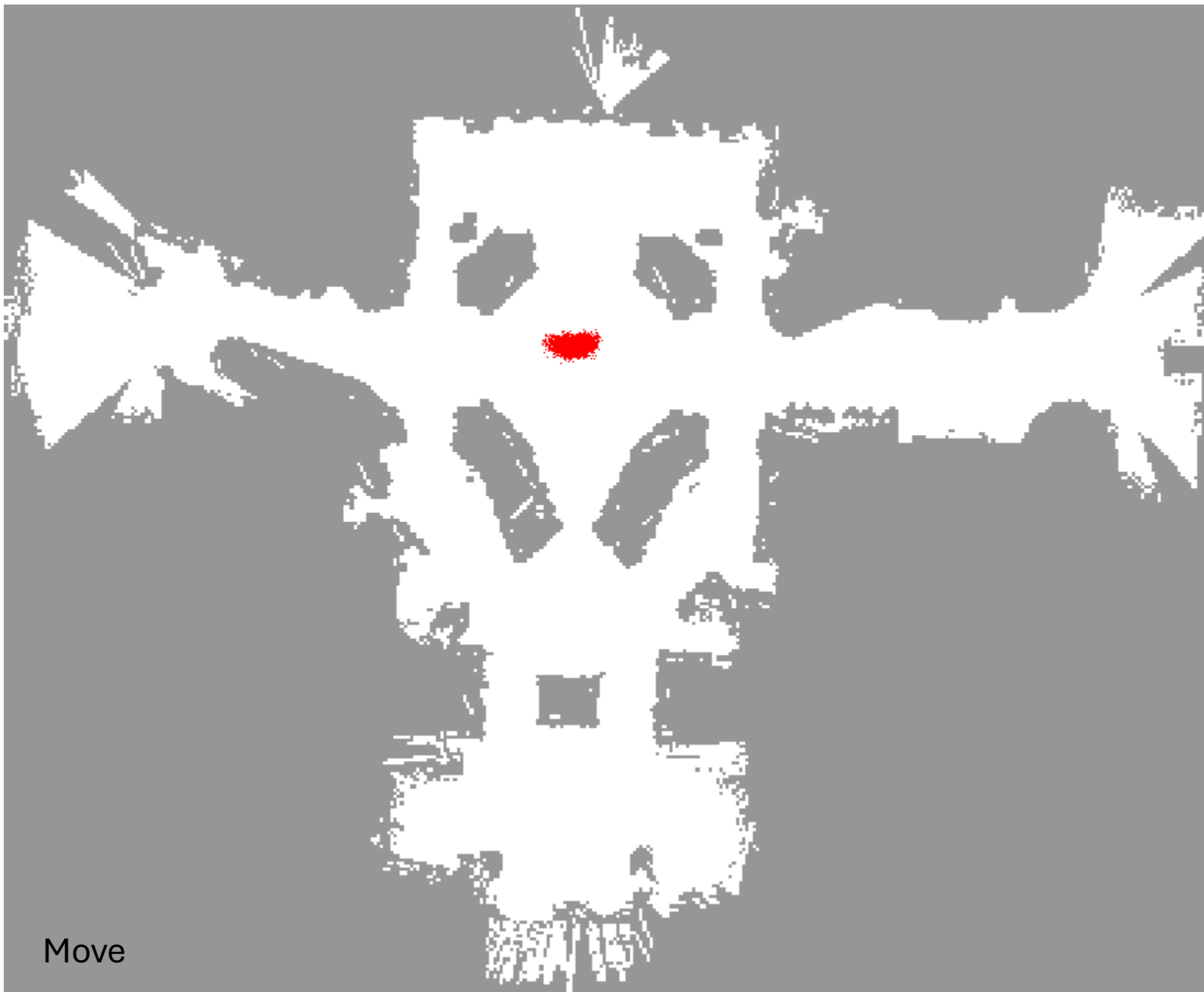




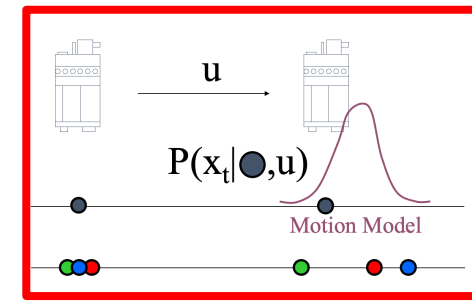




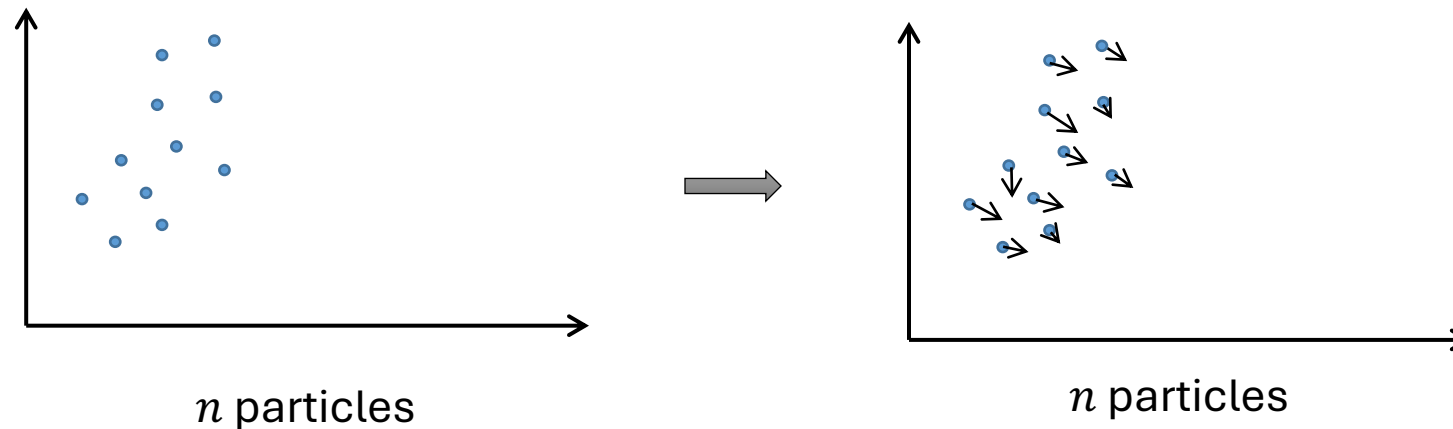




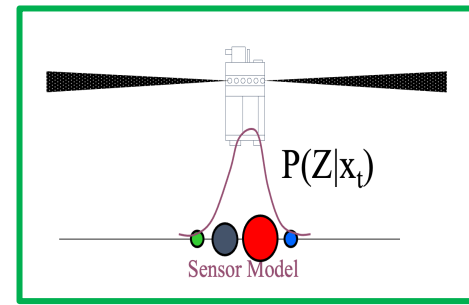
Prediction Phase



- When the command u_{t-1} is executed, each particle is updated to approximate the robot's movement by **sampling** from $p(x_t | x_{t-1}, u_{t-1})$.
- At this stage, typically all particles have equal weight ($w = \frac{1}{N}$).

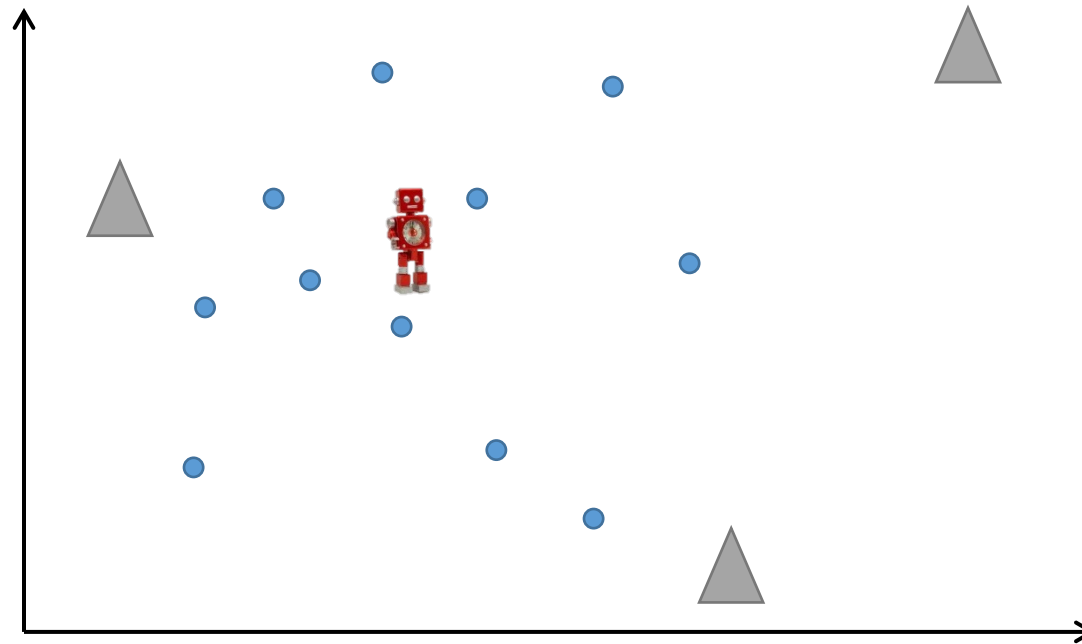
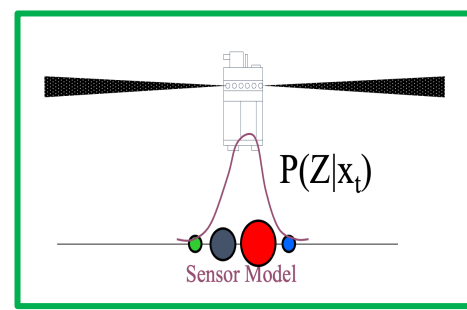


Measurement Phase

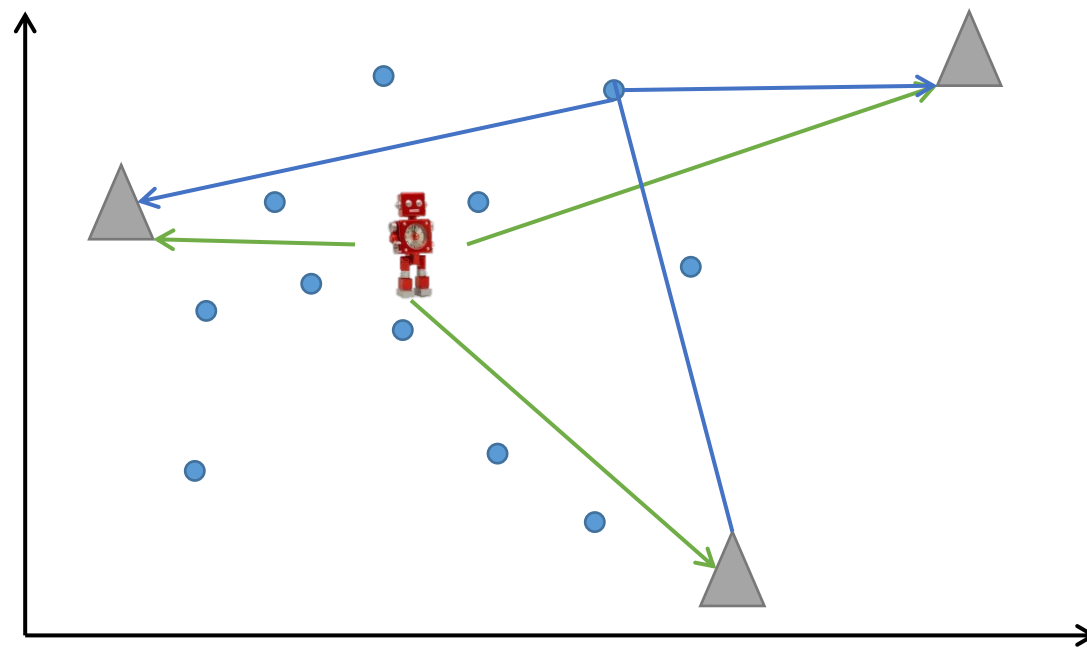
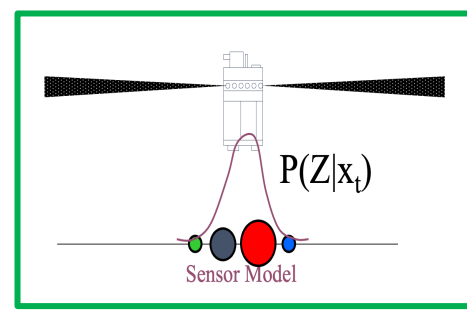


- **Re-weight sample set**, according to the likelihood that robot's current sensors match what would be seen at a given location
 - Let $\langle x, w \rangle$ be a sample.
 - Then, $w \leftarrow \eta P(z|x)$
- z is the sensor measurement;
- η a normalization constant to enforce the sum of w 's equaling 1

Measurement Phase



Measurement Phase

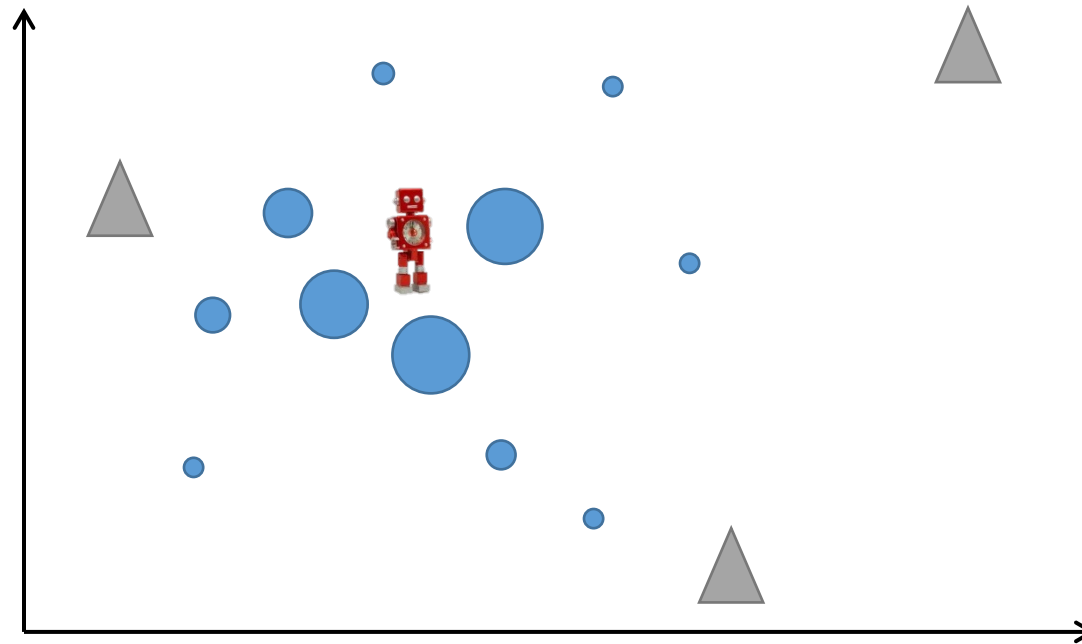
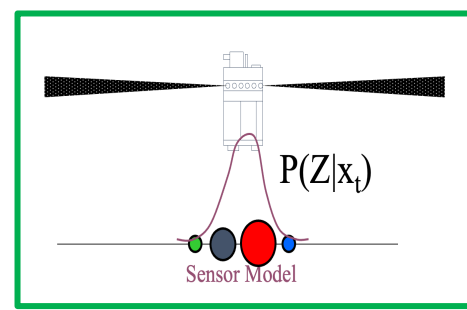


Difference between the
actual measurement
and the
estimated measurement

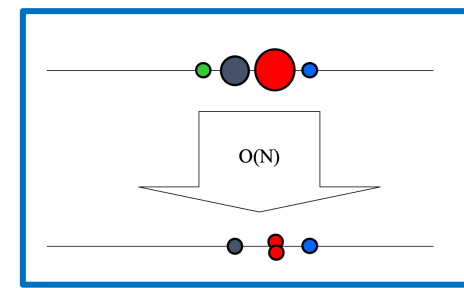


Importance weight

Measurement Phase

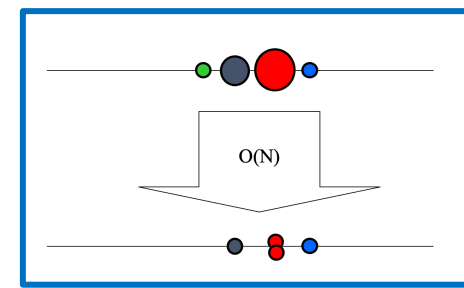


Resampling Step



- After applying the motion update and sensing update, we end up with new positions and weights for particles
- We want to eliminate particles that have very low weight (unlikely to represent robot position) and generate more particles in the more likely areas of the state space.
- **Resample**, according to latest weights
- Add a few uniformly distributed, **random samples**
 - Very helpful in case robot completely loses track of its location

Resampling Step

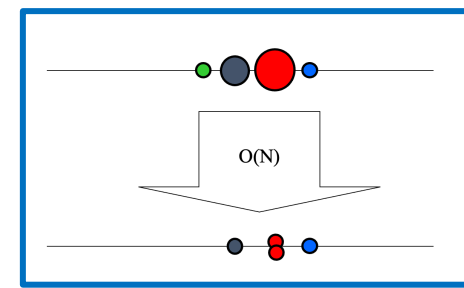


n original particles	Importance Weight $w(x_i)$
	0.2
	0.6
	0.2
	0.8
	0.8
	0.2

$$\sum = 2.8$$

$$\eta = \frac{1}{2.8}$$

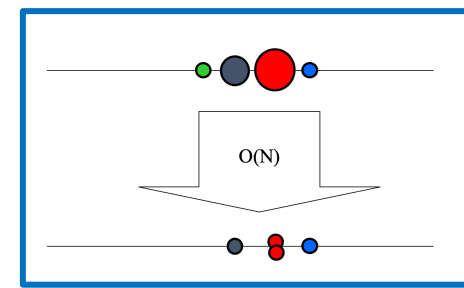
Resampling Step



n original particles	Importance Weight $w(x_i)$	Normalized Probability $p(x_i)$
	0.2	0.07
	0.6	0.21
	0.2	0.07
	0.8	0.29
	0.8	0.29
	0.2	0.07

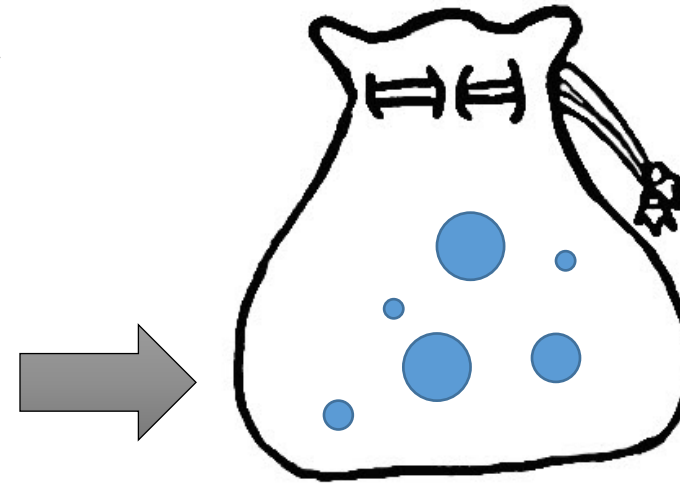
$$\sum = 2.8$$

Resampling Step



n original particles	Importance Weight $w(x_i)$	Normalized Probability $p(x_i)$
	0.2	0.07
	0.6	0.21
	0.2	0.07
	0.8	0.29
	0.8	0.29
	0.2	0.07

$$\sum = 2.8$$



Sample n new particles from the previous set.

- Each particle is chosen with probability $p(x_i)$, with replacement. Add a little random noise to each resampled particle to avoid identical duplicates.

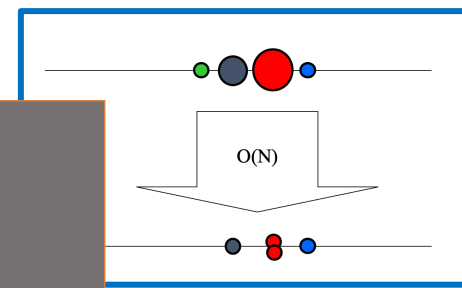
Resampling Step

Is it possible that one of the particles is never chosen?

Yes!

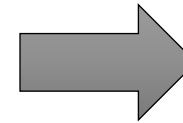
Is it possible that one of the particles is chosen more than once?

Yes!



n original particles	Importance Weight $w(x_i)$	Normalized Probability $p(x_i)$
	0.2	0.07
	0.6	0.21
	0.2	0.07
	0.8	0.29
	0.8	0.29
	0.2	0.07

$$\sum = 2.8$$



Sample n new particles from the previous set.

- Each particle is chosen with probability $p(x_i)$, with replacement.

Particle filtering: putting the steps together

Predict

Move each particle by sampling from the motion model.

Weight

Score each particle with the sensor likelihood and normalize the weights.

Resample

Draw particles according to their weights to reallocate the sample budget.

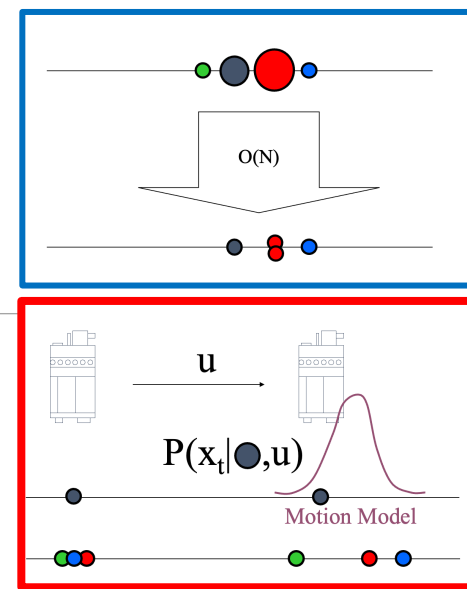
Resampling adds no sensor evidence. The upcoming code resamples before the next prediction.

Particle Filter Algorithm

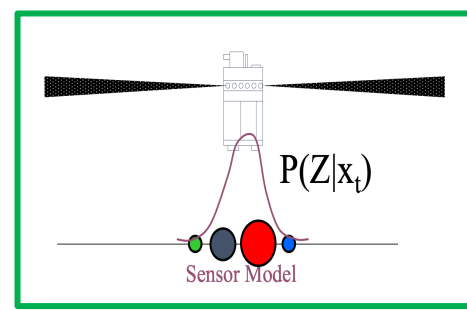
- Algorithm **particle_filter**(X_{t-1}, u_{t-1}, z_t):
- $X_t = \emptyset, \eta = 0$
- Input:
 - u_{t-1} is the action that was executed at time $t - 1$
 - $X_{t-1} = \left\{ \langle x_{t-1}^j, w_j \rangle \right\}_{j=1 \dots N}$ is the set of weighted particles at time $t - 1$
 - z_t is the sensor measurement at time $t - 1$
- Output:
 - $X_t = \left\{ \langle x_t^j, w_j \rangle \right\}_{j=1 \dots N}$ is a set of weighted particles at time t

Particle Filter Algorithm

- Algorithm **particle_filter**(X_{t-1}, u_{t-1}, Z_t):
- $X_t = \emptyset, \eta = 0$
- **For $j = 1 \dots N$ *Generate new samples***
 - Sample index j from discrete index set $\{1, \dots, N\}$ based on w_{t-1}
 - Sample x_t^j from $p(x_t^j | x_{t-1}^j, u_{t-1})$
- NOTES:
 - j indicates a randomly chosen particle based on weights at time $t - 1$
 - x_t^j is determined using only motion model for action, u_{t-1} applied in state x_{t-1}^j

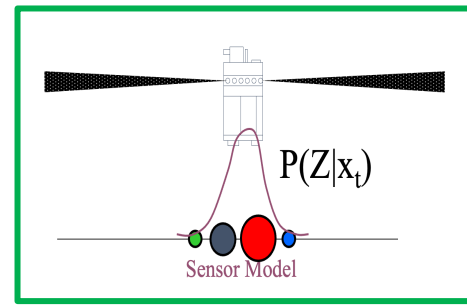


Particle Filter Algorithm



- Algorithm **particle_filter**(X_{t-1}, u_{t-1}, Z_t):
- $X_t = \emptyset, \eta = 0$
- **For $j = 1 \dots N$** *Generate new samples*
 - Sample index j from discrete index set $\{1, \dots, N\}$ based on w_{t-1}
 - Sample x_t^j from $p(x_t^j | x_{t-1}^j, u_{t-1})$
 - $w_t^j = p(z_t | x_t^j)$ *Compute importance weight*
 - $\eta = \eta + w_t^j$ *Update normalization factor*
 - $X_t = X_t \cup \{ \langle x_t^j, w_t^j \rangle \}$ *Add to set of new particles*

Particle Filter Algorithm



- Algorithm **particle_filter**(X_{t-1}, u_{t-1}, z_t):
- $X_t = \emptyset, \eta = 0$
- **For $j = 1 \dots N$** *Generate new samples*
 - Sample index j from discrete index set $\{1, \dots, N\}$ based on w_{t-1}
 - Sample x_t^j from $p(x_t^j | x_{t-1}^j, u_{t-1})$
 - $w_t^j = p(z_t | x_t^j)$ *Compute importance weight*
 - $\eta = \eta + w_t^j$ *Update normalization factor*
 - $X_t = X_t \cup \{ \langle x_t^j, w_t^j \rangle \}$ *Add to set of new particles*
- **For $j = 1 \dots N$**
 - $w_t^j = w_t^j / \eta$ *Normalize weights*