Recall: The two update steps in robot localization

1. ACTION (or prediction) update: the robot moves and estimates its position through its proprioceptive sensors. During this step, the robot uncertainty grows. This step uses the Total Probability formula to update belief

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

$$p(x) = \int_y p(x|y)p(y)dy$$

2. PERCEPTION (or measurement) update: the robot makes an observation using its exteroceptive sensors and correct its position by opportunely combining its belief before the observation with the probability of making exactly that observation. During this step, the robot uncertainty shrinks. This step uses the Bayes Rule to update belief.

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

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}$$

Recall: Markov versus Kalman localization

- Two approaches exist to represent the probability distribution and to compute the Total Probability and Bayes Rule during the Action and Perception phases
- Markov approach:
  - The configuration space is divided into many cells. Each cell contains the probability of the robot to be in that cell.
  - The probability distribution of the sensors model is also discrete.
  - During Action and Perception, all the cells are updated. Therefore, the computational cost is very high.
- Kalman approach:
  - The probability distribution of both the robot configuration and the sensor model is assumed to be continuous and Gaussian!
  - Need only to update mean value μ and covariance Σ. Therefore the computational cost is very low!

Sample-based Localization

- Maintain multiple estimates of robot's location
- Track possible robot positions, given all previous measurements
- Key idea: represent the belief that a robot is at a particular location by a set of "samples", or "particles"

Represent *Bel*(x) by set of N weighted, random samples, called *particles*:

$$S = \{s_i | i = 1..N\}$$

where a sample, s_i, is of the form: $\langle\langle x_i, y_i, \theta_i\rangle, w_i\rangle$

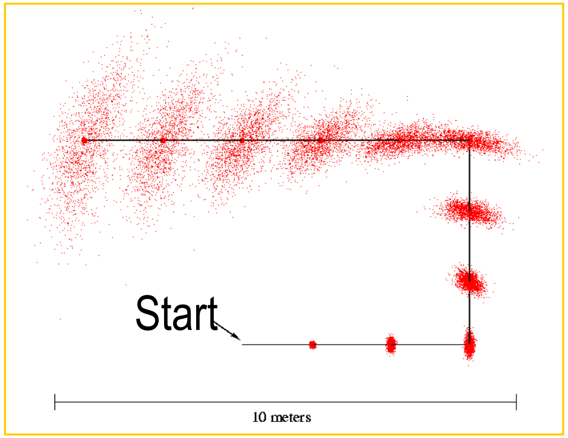
Here, $\langle x_i, y_i, \theta_i\rangle$ represents robot's position (location and orientation)

w_i represents a weight, where sum of all w's is 1 (analogous to discrete probability)

Updating beliefs using Particle Filters (PF)

- As before, 2 models: Action (Motion) Model, Perception (Sensing) Model
- Robot Motion Model:
  - When robot moves, PF generates N new samples that approximate robot's position after motion command.
  - Each sample is generated by randomly drawing from previous sample set, with likelihood determined by *w* values.
    - A new sample x_t^i is generated from $p(x_t|x_{t-1}, u_{t-1})$ using x_{t-1}^i and u_{t-1}
    - The corresponding weight is computed by $w_t^i = p(z_t|x_t^i)$ and then normalized

Sampling-based approximation of position belief for non-sensing robot



Particle Filter Algorithm

1. Algorithm **particle_filter**(S_{t-1}, u_{t-1}, z_t):
2. $S_t = \emptyset, \quad \eta = 0$
3. **For** $i = 1 \dots n$ ***Generate new samples***
4. Sample index j(i) from the discrete distribution given by w_{t-1}
5. Sample x_t^i from $p(x_t | x_{t-1}, u_{t-1})$ using $x_{t-1}^{j(i)}$ and u_{t-1}
6. $w_t^i = p(z_t | x_t^i)$ ***Compute importance weight***
7. $\eta = \eta + w_t^i$ ***Update normalization factor***
8. $S_t = S_t \cup \{< x_t^i, w_t^i >\}$ ***Insert into new particle set***
9. **For** $i = 1 \dots n$
10. $w_t^i = w_t^i / \eta$ ***Normalize weight***

Particle Filter Algorithm

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

draw x^i_{t-1} from $Bel(x_{t-1})$

draw x^i_t from $p(x_t \mid x^i_{t-1}, u_{t-1})$

Importance factor for x^i_t:

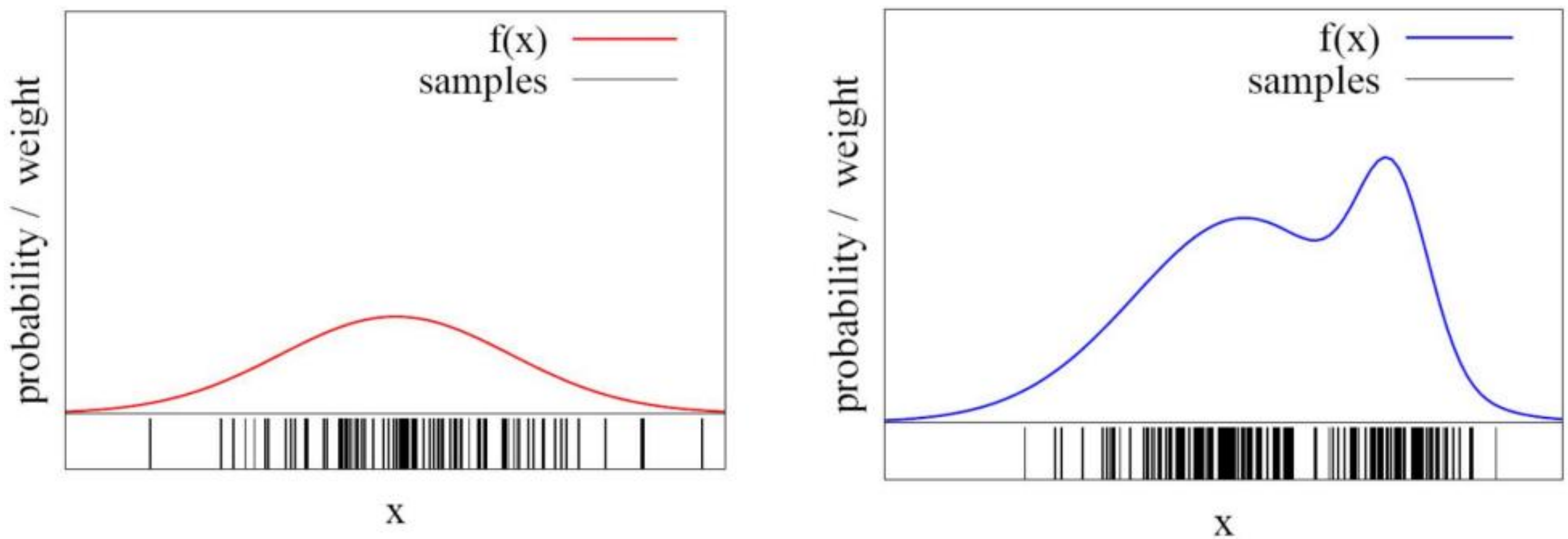
$$w^i_t = \frac{\text{target distribution}}{\text{proposal distribution}}$$

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

$$\propto p(z_t|x_t)$$

Function Approximation

- Particle sets can be used to approximate functions

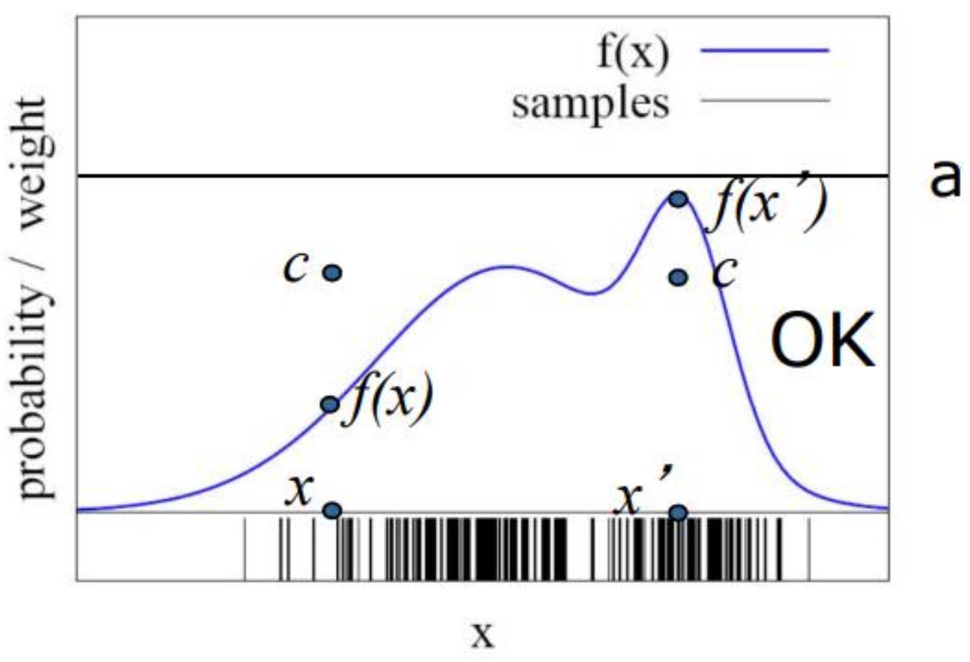


- The more particles fall into an interval, the higher the probability of that interval

- How to draw samples from a function/distribution?

Rejection Sampling

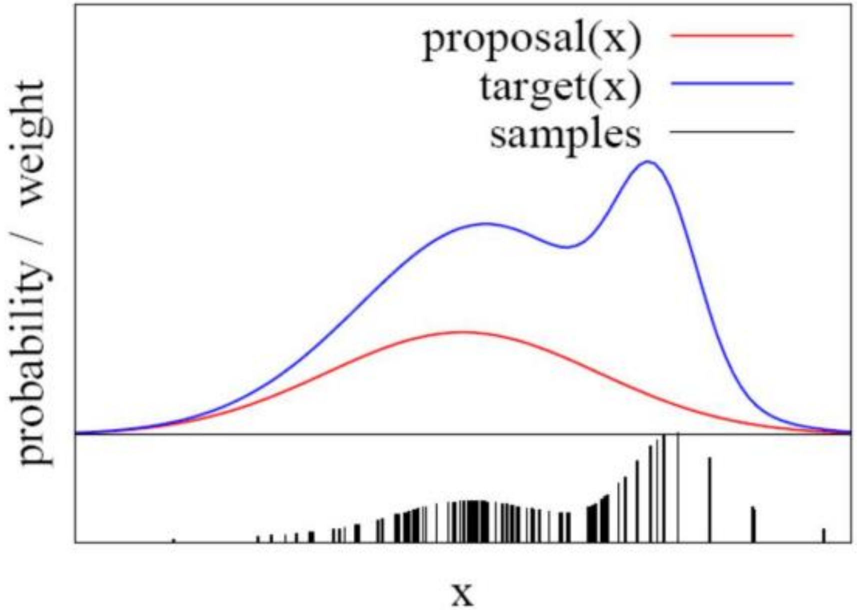
- Let us assume that f(x) < a for all x
- Sample x from a uniform distribution
- Sample c from [0, a]
- If f(x) > c, then keep the sample
- otherwise, reject the sample



Importance Sampling Principle

- We can even use a different distribution *g* to generate samples from *f*
- By introducing an importance weight *w*, we can account for the "differences between *g* and *f*"

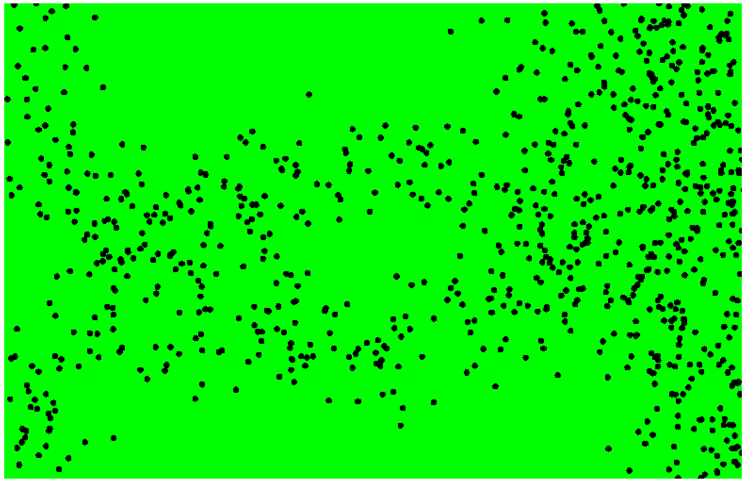
- Importance weight *w* = *f* / *g*
- *f* is called *target*
- *g* is called *proposal*



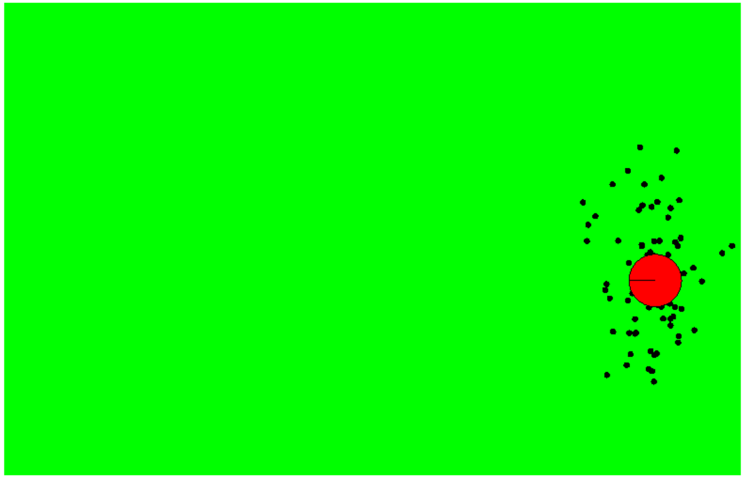
Resampling

- **Given**: Set S of weighted samples $\{(x_i, w_i) \mid i = 1, 2, \ldots\}$.
- **Wanted** : Random sample, where the probability of drawing x_i is given by w_i.
- Typically done n times with replacement to generate new sample set S'.

Importance Sampling with Resampling

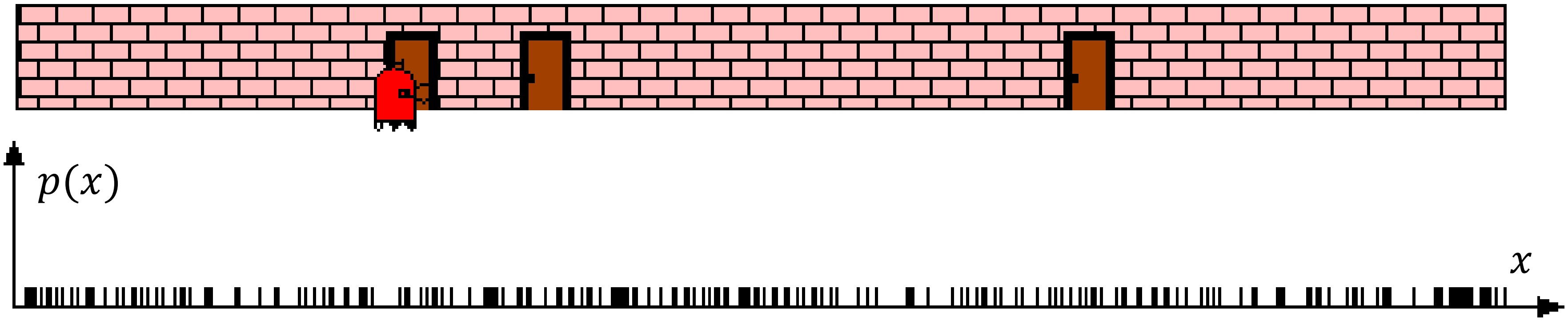


Weighted samples



After resampling

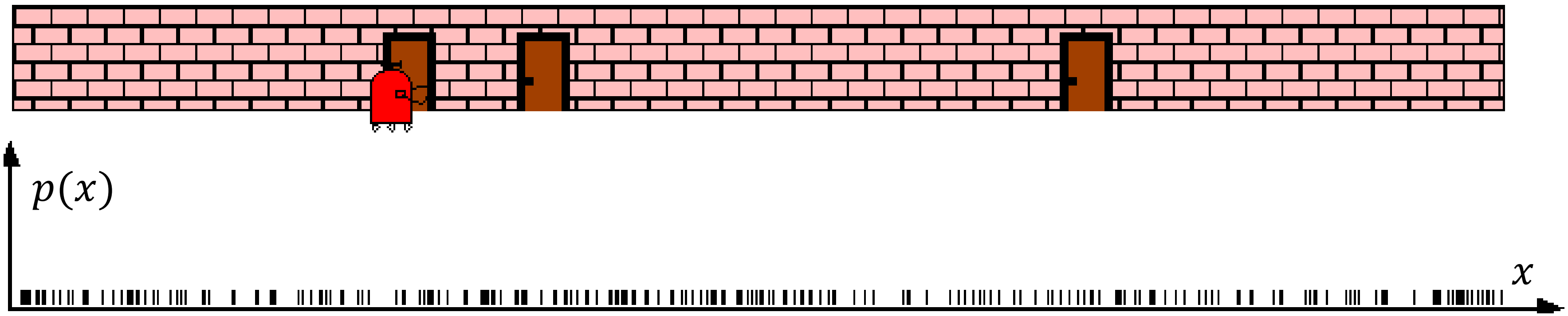
Particle Filters

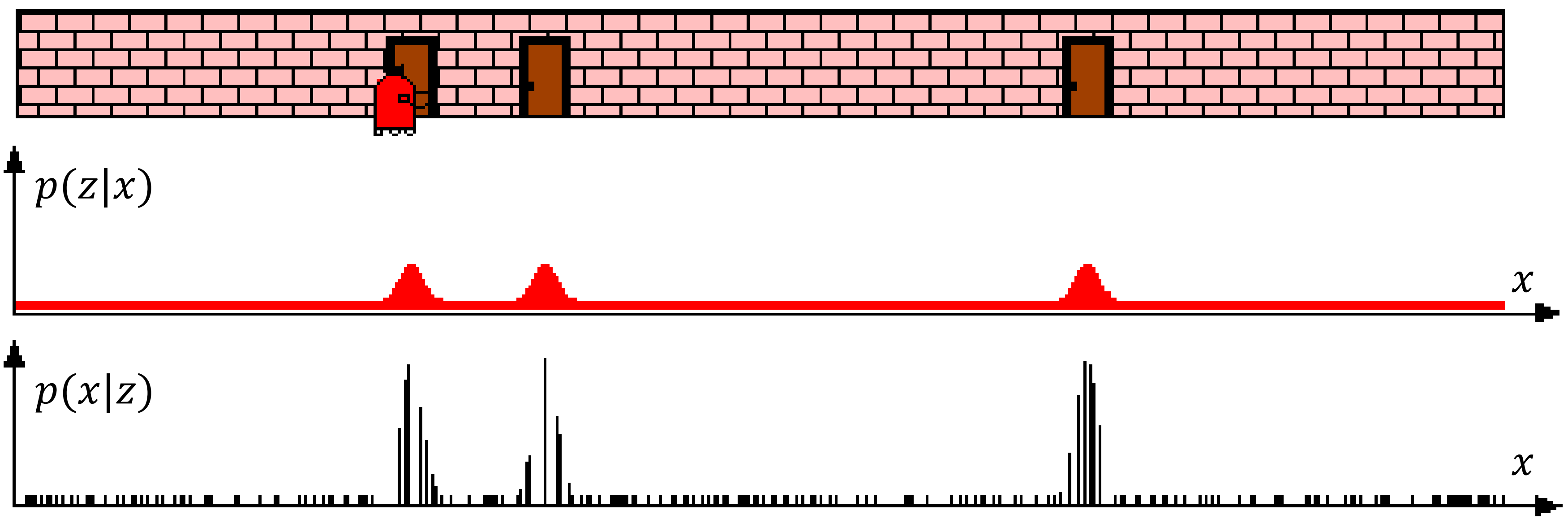


Sensor Information: Importance Sampling

$$Bel(x) \leftarrow \alpha p(z|x) Bel^{-}(x)$$
$$w \leftarrow \alpha p(z|x)$$

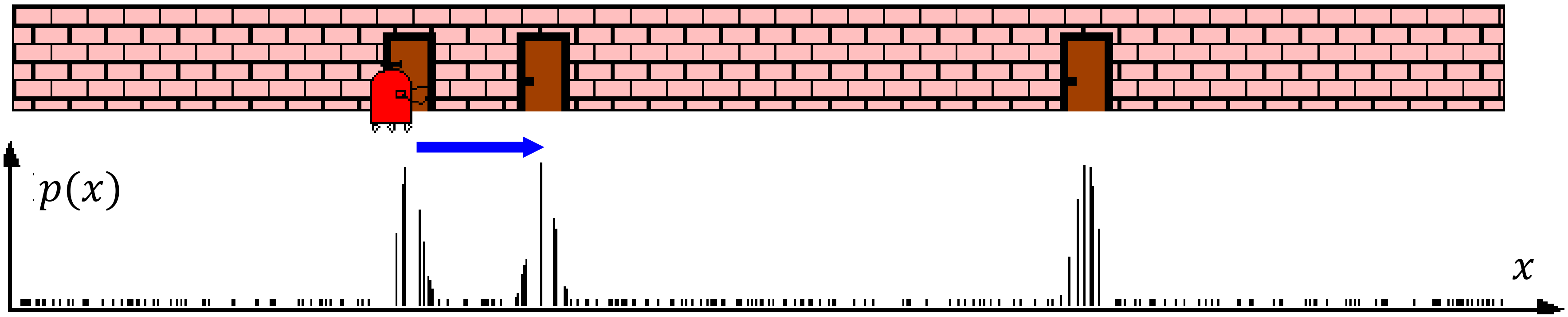
$$w = \frac{Bel(x)}{Bel^{-}(x)}$$

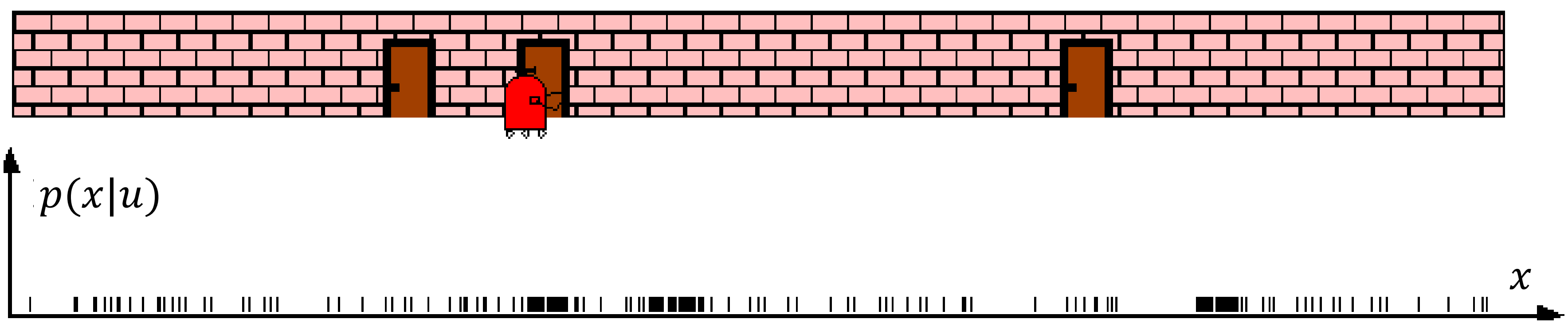




Robot Motion

$$Bel^{-}(x) \leftarrow \int p(x|u,x')Bel(x')\ \mathrm{d}x'$$

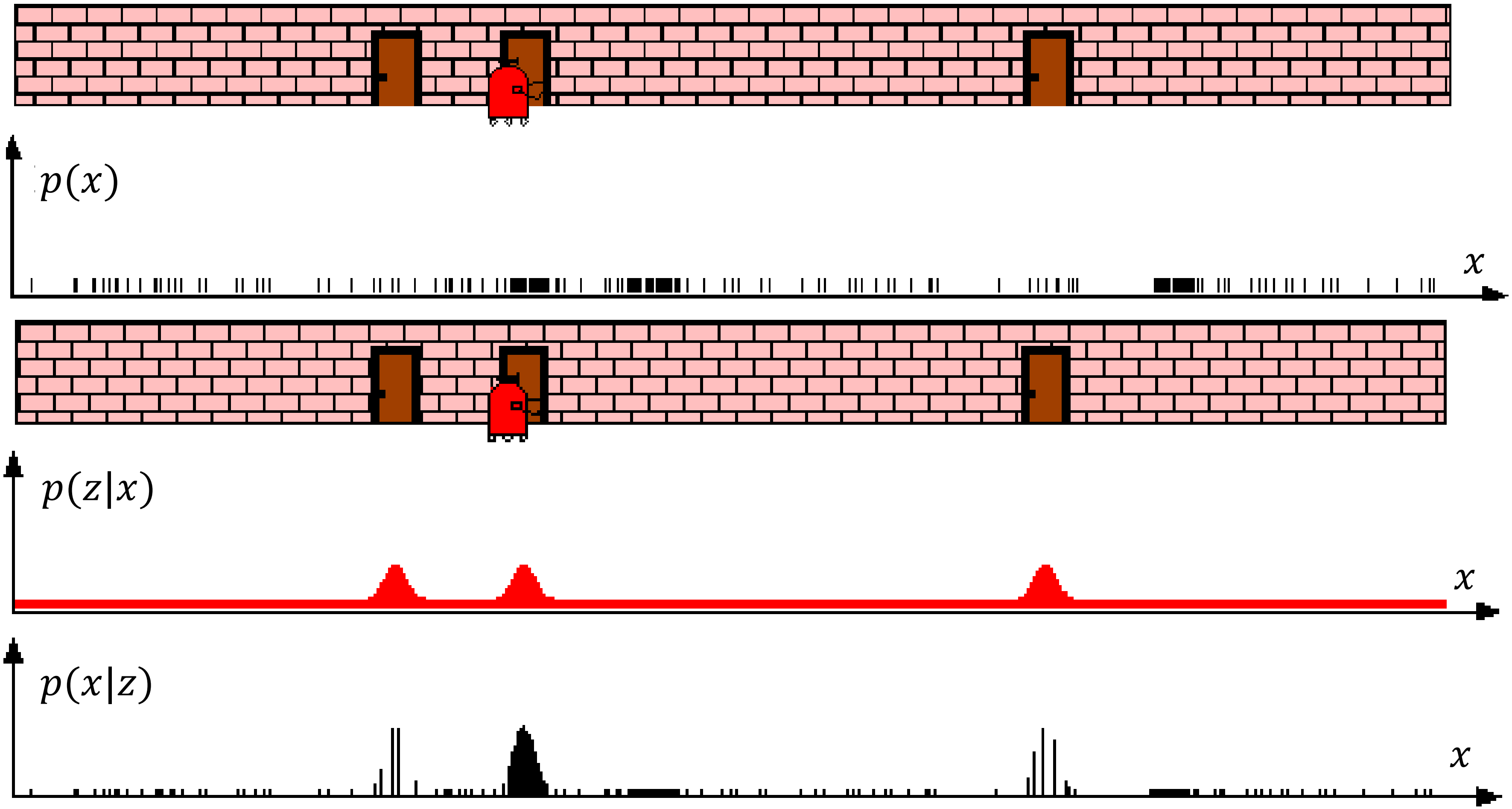




Sensor Information: Importance Sampling

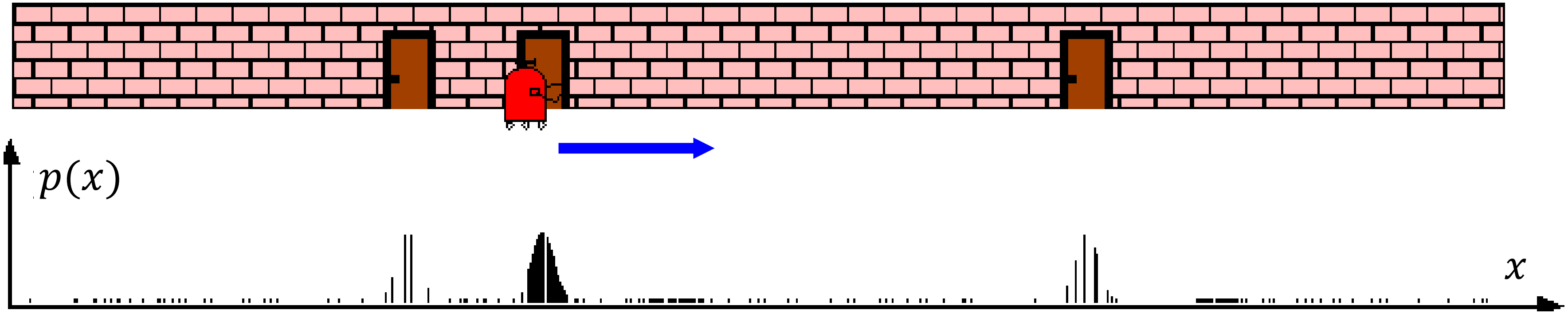
$$Bel(x) \leftarrow \alpha p(z|x) Bel^{-}(x)$$
$$w \leftarrow \alpha p(z|x)$$

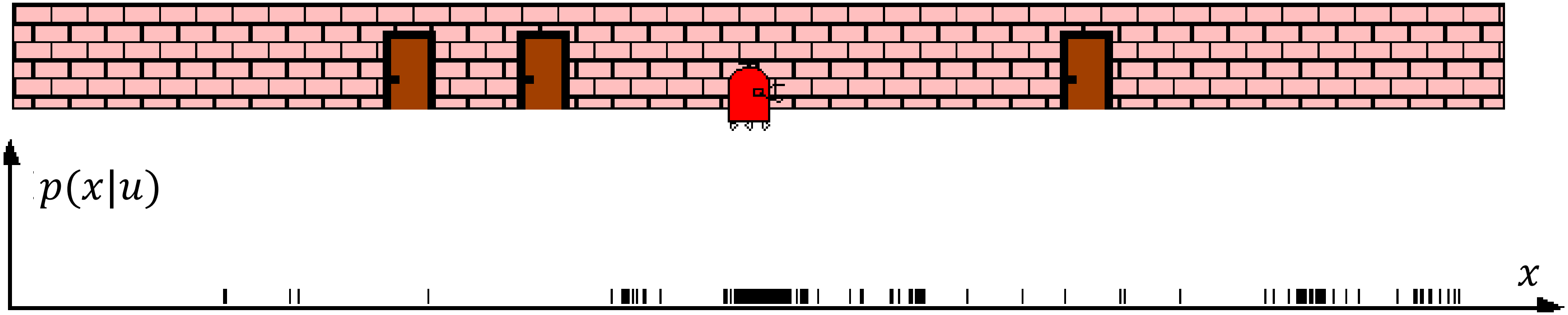
$$w = \frac{Bel(x)}{Bel^{-}(x)}$$



Robot Motion

$$Bel^{-}(x) \quad \leftarrow \quad \int p(x|u,x')Bel(x') \,\mathrm{d}x'$$





The particle filter algorithm

1. Sample the next generation for particles using the proposal distribution
2. Compute the importance weights:

   weight = target distribution / proposal distribution
3. Resampling: “Replace unlikely samples by more likely ones”

- Each particle is a potential pose of the robot
- Proposal distribution is the motion model of the robot (prediction step)
- The observation model is used to compute the importance weight (correction step)

The particle filter algorithm

A variant of the Bayes filter based on importance sampling

Algorithm Particle_filter($\mathcal{X}_{t-1}, u_t, z_t$)**:**
    $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
    for $m = 1$ to M do
        sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$
        $w_t^{[m]} = p(z_t \mid x_t^{[m]})$
        $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
    endfor
    for $m = 1$ to M do
        draw i with probability $\propto w_t^{[i]}$
        add $x_t^{[i]}$ to $\mathcal{X}_t$
    endfor
    return $\mathcal{X}_t$

Resampling Algorithm

Algorithm **systematic_resampling**(S,n):	
$S' = \emptyset, c_0 = 0$	
For $i = 1 \ldots n$	**Generate cdf** (cumulative distribution function)
$\quad c_i = c_{i-1} + w^i$	$c_i = w_1 + w_2 + \cdots + w_i$
$u_1 \sim U(0, n^{-1}],\ i = 1$	**Initialize threshold** $0 < u_1 \leq \frac{1}{n}$
For $j = 1 \ldots n$	**Draw samples …**
$\quad$ **While** ($u_j > c_i$)	**Skip until next threshold reached**
$\quad\quad i = i + 1$	$c_{i-1} < u_j \leq c_i$
$\quad S' = S' \cup \{< x^i, n^{-1} >\}$	**Insert**
$\quad u_{j+1} = u_j + n^{-1}$	**Increment threshold** $u_j = u_1 + \frac{j-1}{n}$
Return S'	

Also called **stochastic universal sampling**

Resampling Algorithm: Example

$$c_0 = 0$$

$$c_i = w_1 + w_2 + \cdots + w_i$$

$$0 < u_1 \leq \frac{1}{n}$$

$$u_j = u_1 + \frac{j-1}{n}$$

$$c_{i-1} < u_j \leq c_i$$

n	1	2	3	4	5	6	7	8	9	10
w	0.10	0.15	0.05	0.25	0.15	0.05	0.06	0.04	0.10	0.05
c	0.10	0.25	0.30	0.55	0.70	0.75	0.81	0.85	0.95	1.00
u	0.02	0.12	0.22	0.32	0.42	0.52	0.62	0.72	.82	0.92
S	x1	x2	x2	x4	x4	x4	x5	x6	x8	x9

u	0.07	0.17	0.27	0.37	0.47	0.57	0.67	0.77	0.87	0.97
s	x1	x2	x3	x4	x4	x5	x5	x7	x9	x10

Resampling Algorithm: Example

$c_0 = 0$

$c_i = w_1 + w_2 + \cdots + w_i$

$u_1 = 0.02$

$0 < u_1 \leq \frac{1}{n}$

$u_j = u_1 + \frac{j-1}{n}$

While $(u_j > c_i)$
$\quad i = i + 1$
$S' = S' \cup \{< x^i, n^{-1} >\}$
$u_{j+1} = u_j + n^{-1}$

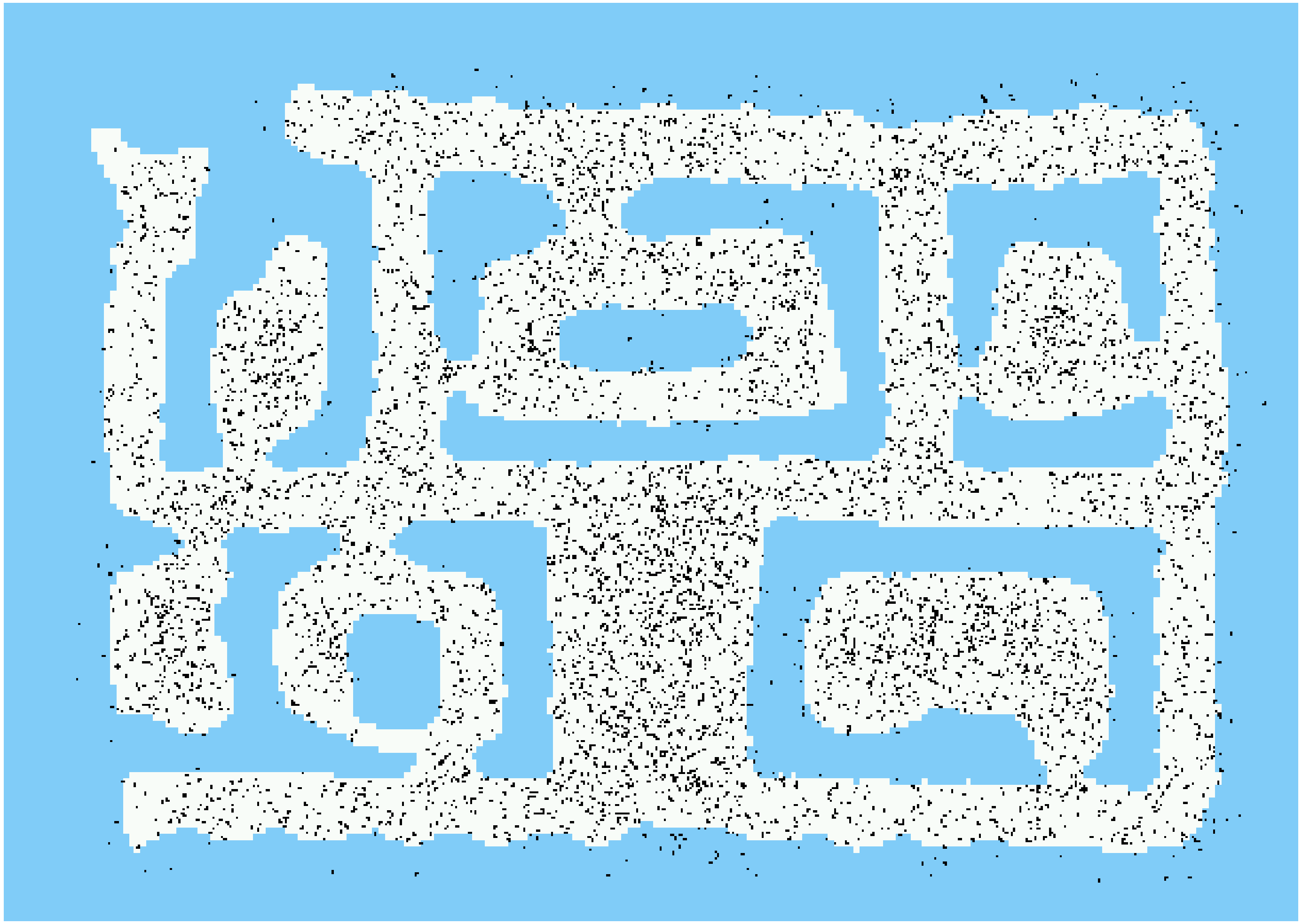
n	1	2	3	4	5	6	7	8	9	10
w	0.10	0.15	0.05	0.25	0.15	0.05	0.06	0.04	0.10	0.05
c	0.10	0.25	0.30	0.55	0.70	0.75	0.81	0.85	0.95	1.00
u	0.02	0.12	0.22	0.32	0.42	0.52	0.62	0.72	.82	0.92
S	x1	x2	x2	x4	x4	x4				

Initially
$i = 1$

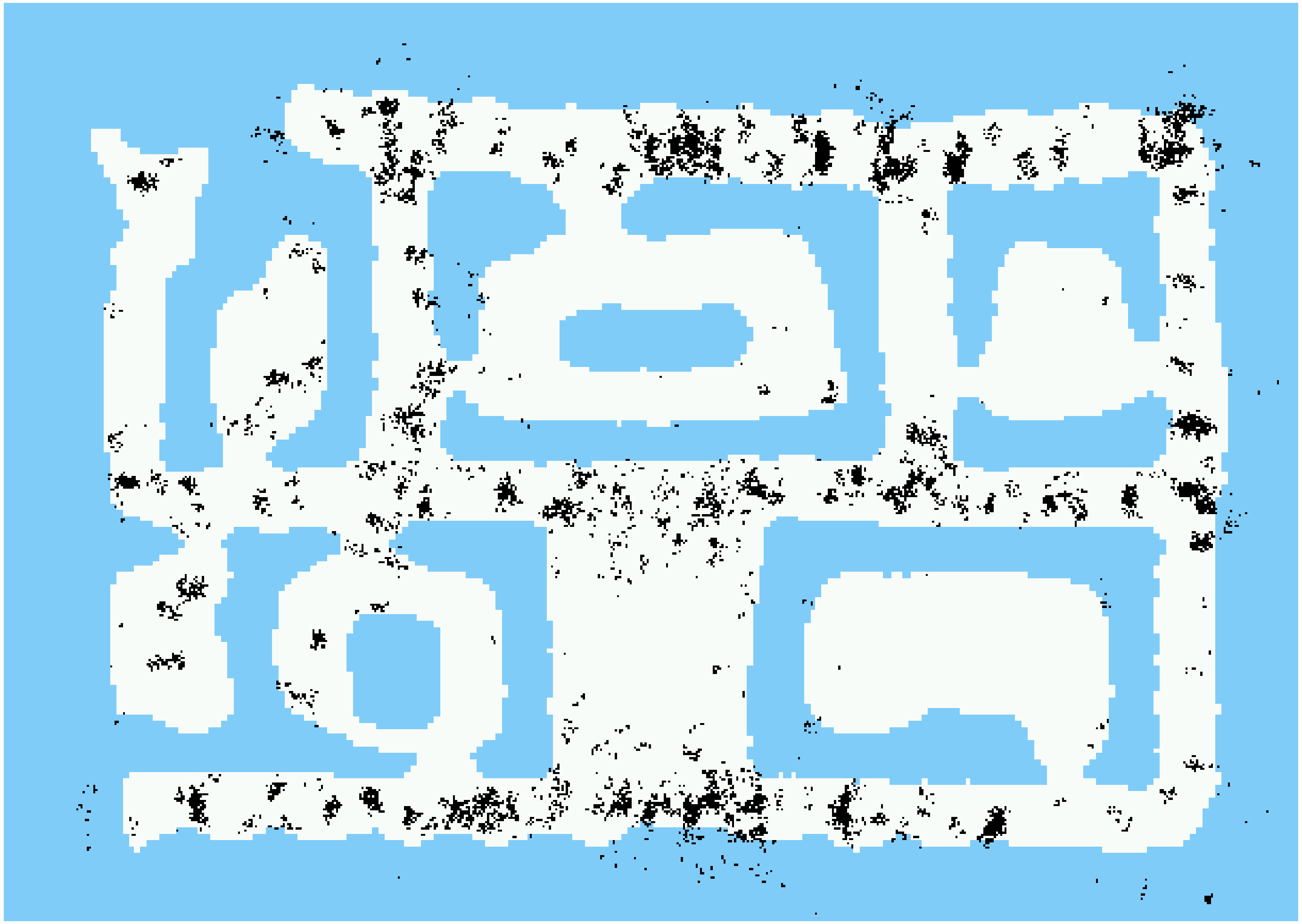
$j = 1$	2	3	4	5	6	7	8	9	10
$i = 1$	2	2	4	4	4	5			

Particle Filter Example: Initial Distribution

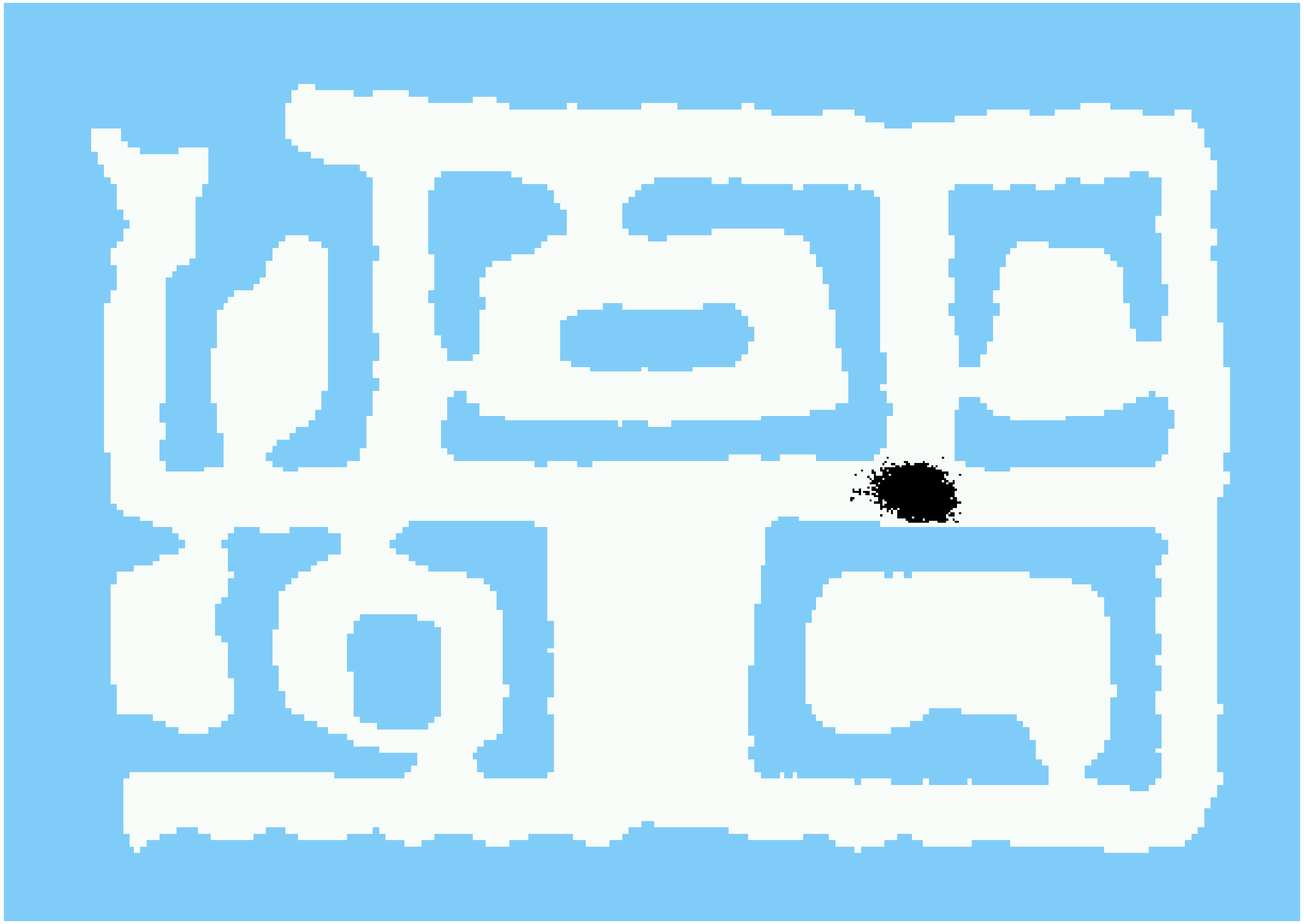
Initially, robot doesn't know where it is



After Incorporating 10 Ultrasound Scans



After Incorporating 65 Ultrasound Scans



Summary – Particle Filters (1/2)

- Particle filters are an implementation of recursive Bayesian filtering
- They represent the posterior by a set of weighted samples
- They can model arbitrary and thus also non-Gaussian distributions
- Proposal to draw new samples
- Weights are computed to account for the difference between the proposal and the target

Summary – Particle Filters (2/2)

- In the context of localization, the particles are propagated according to the motion model.
- They are then weighted according to the likelihood model (likelihood of the observations).
- In a re-sampling step, new particles are drawn with a probability proportional to the likelihood of the observation.
- This leads to one of the most popular approaches to mobile robot localization