Topic 12 Particle Filters: The art of hedging your bets

Topic 12 Particle Filters: The art of hedging your bets

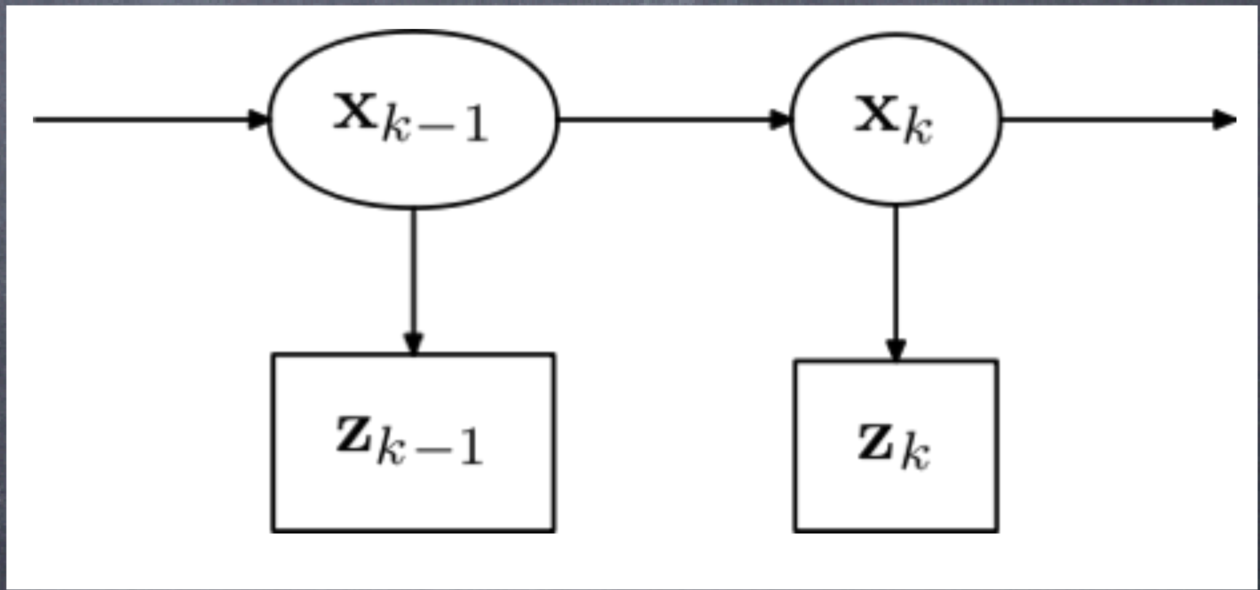
Bayesian filtering recap

posterior — likelihood — dynamics — prior

$$p(\mathbf{x}_k|\mathbf{Z}_k) = \alpha\, p(\mathbf{Z}_k|\mathbf{x}_k) \int p(\mathbf{x}_k|\mathbf{x}_{k-1}) p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1})\, d\mathbf{x}_{k-1}$$

"belief at t=k" — "update" — "predict" — "belief at t=k-1"

- distribution considers all possible robot poses
- assume one true pose
- recursive Markovian inference
- model, not algorithm



Bayesian filitering recap

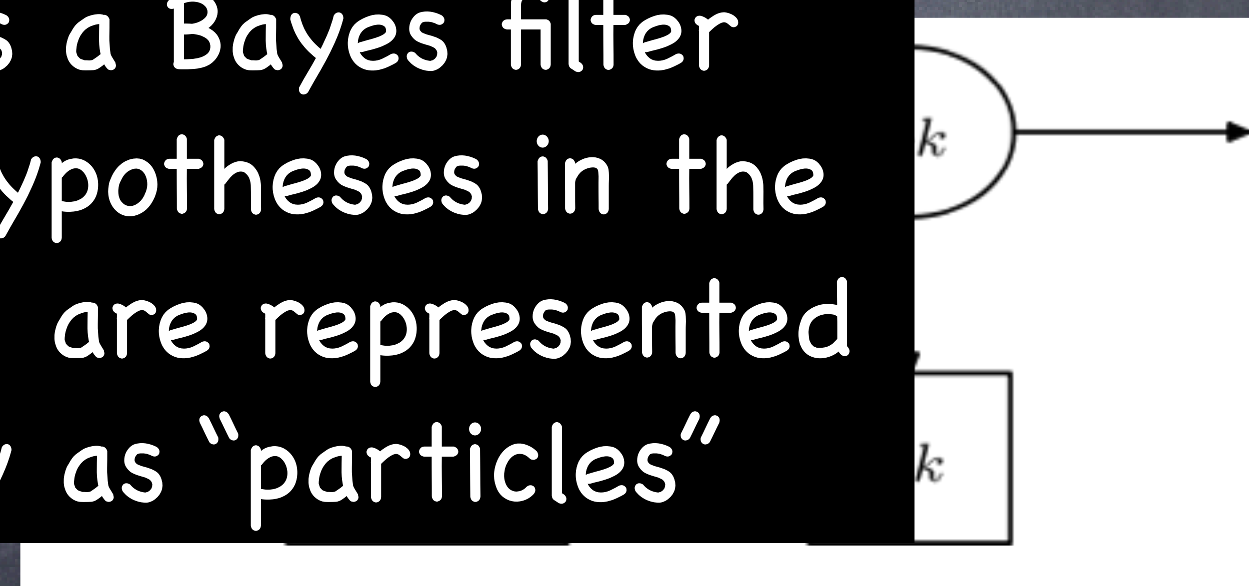
posterior — **likelihood** — **dynamics** — **prior**

$$p(\mathbf{x}_k|\mathbf{Z}_k) = \alpha\, p(\mathbf{Z}_k|\mathbf{x}_k) \int p(\mathbf{x}_k|\mathbf{x}_{k-1}) p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1})\, d\mathbf{x}_{k-1}$$

"belief at t=k" — **"update"** — **"predict"** — **"belief at t=k-1"**

- distribution considers all pos
- ass
- rec infe
- model, not algorithm

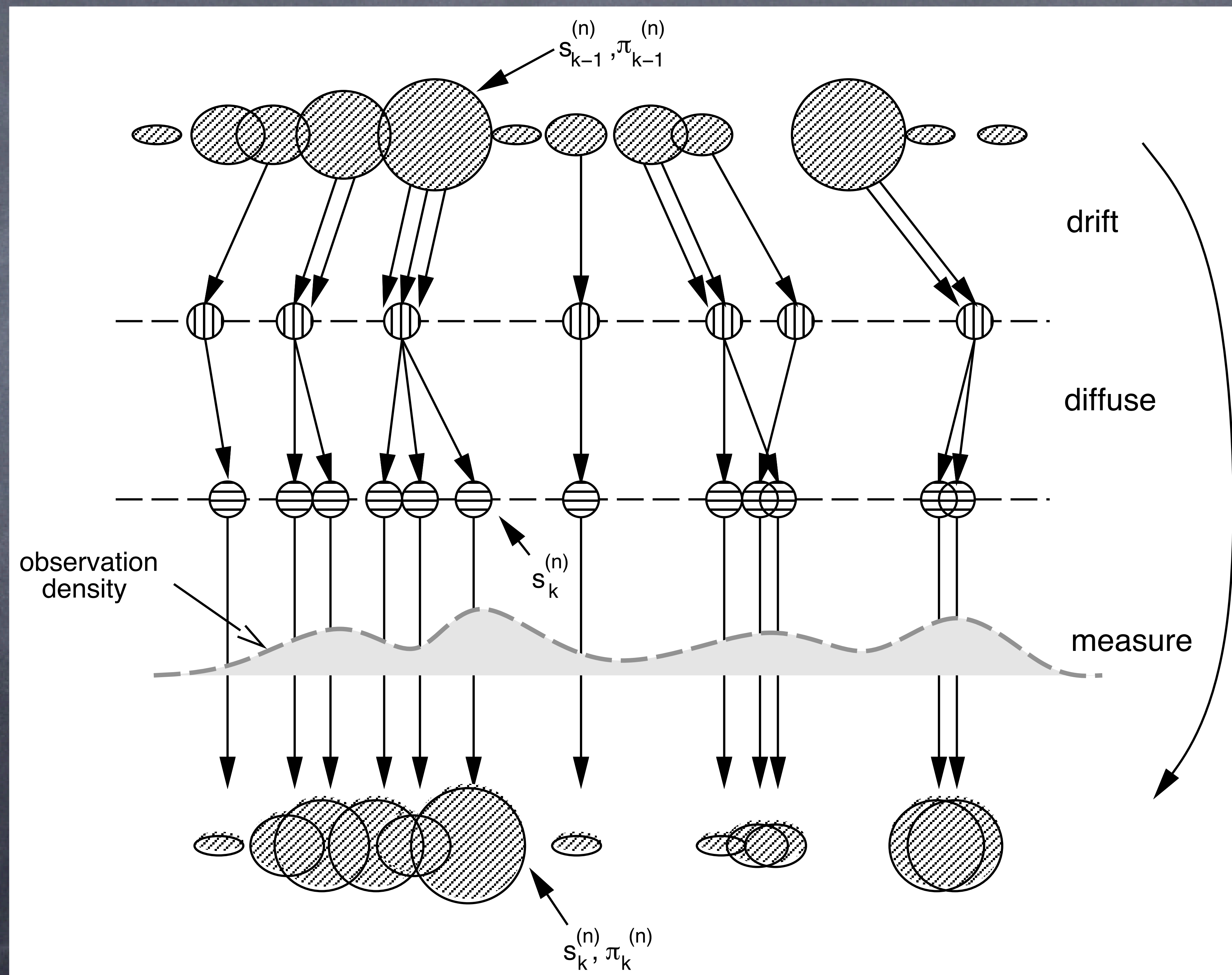
Particle filter is a Bayes filter where multiple hypotheses in the belief distribution are represented computationally as "particles"



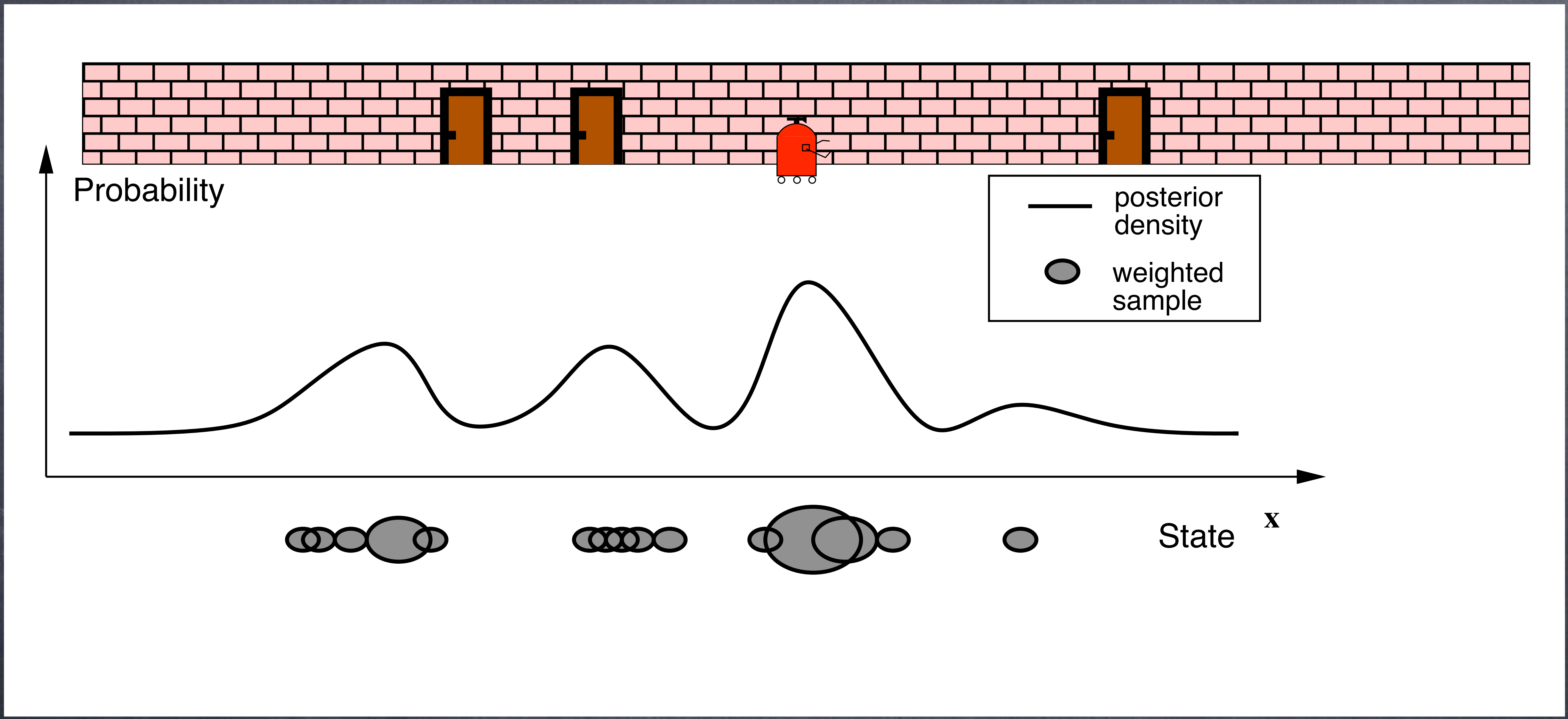
Condensation Algorithm

[Isard, Blake 1998]

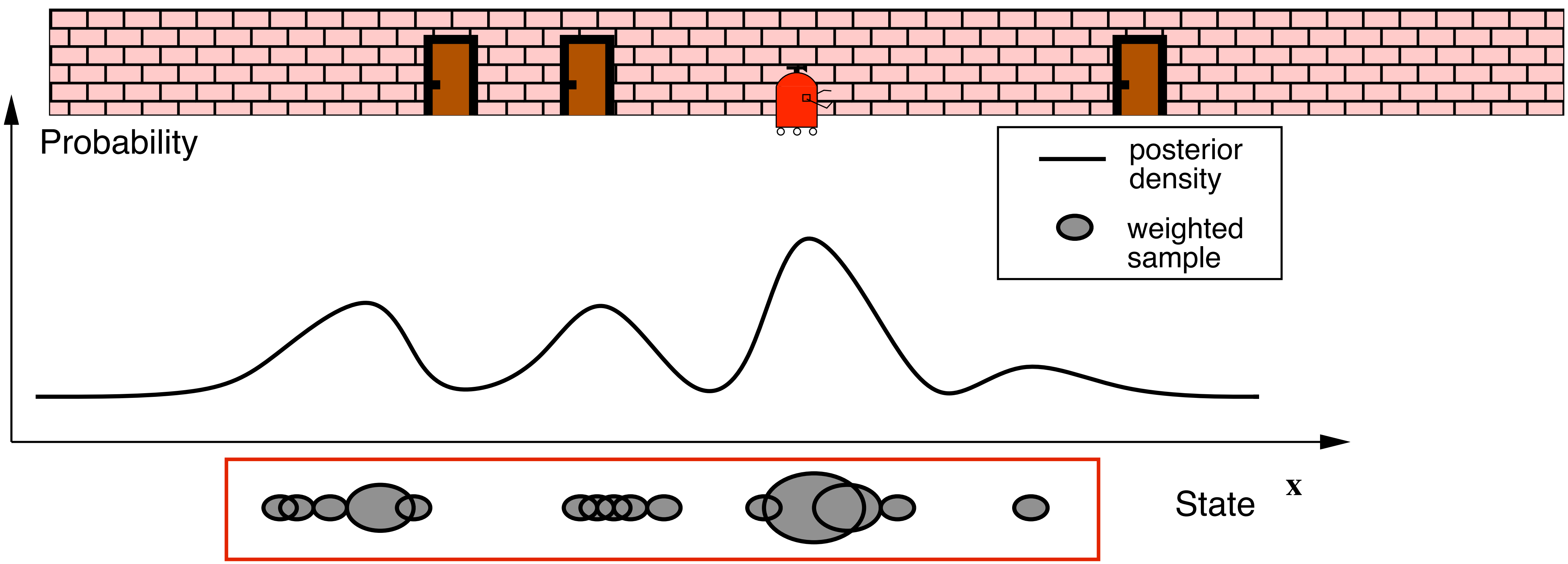
- Condensation is one algorithm for particle filtering
- State belief represented as particle hypotheses



Representing belief as particles

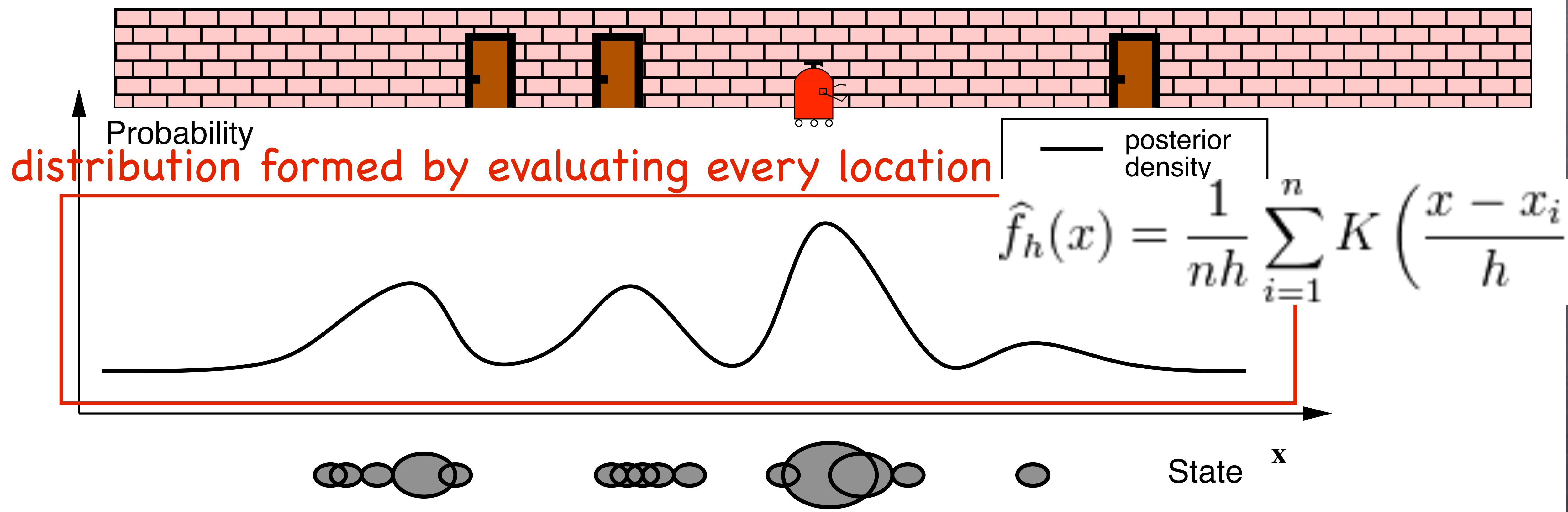


Representing belief as particles



each particle is a hypothesis of robot's pose, weighted by the probability of the pose

Representing belief as particles



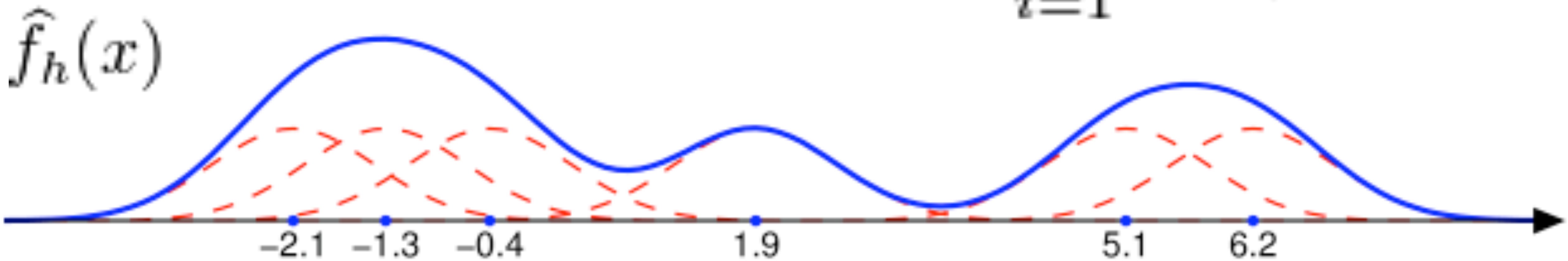
$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

each particle is a hypothesis of robot's pose, weighted by the probability of the pose

Representing belief as particles

Tangent: How do particles form a distribution?

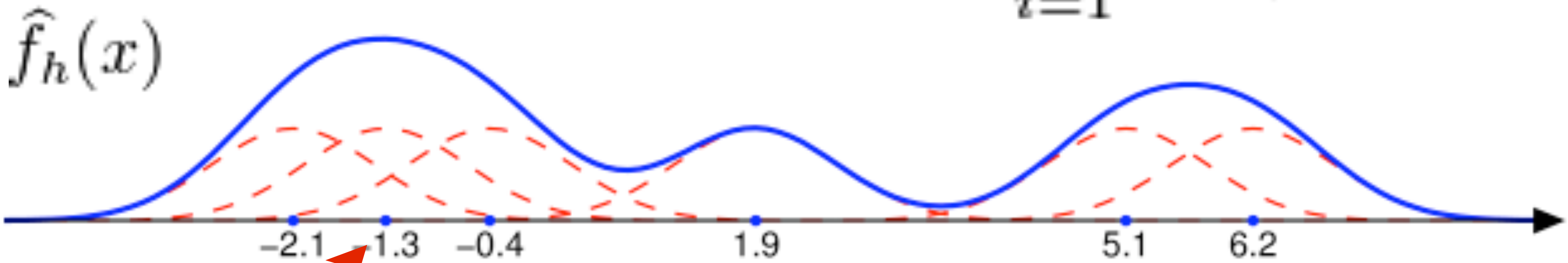
$$\widehat{f}_h(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

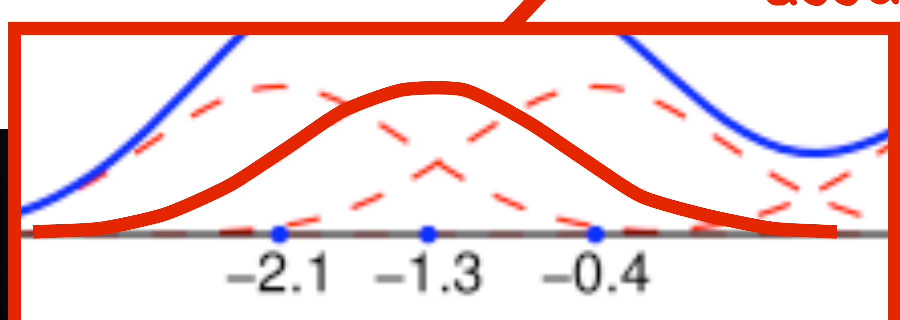


Representing belief as particles

Tangent: How do particles form a distribution?

$$\widehat{f}_h(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$





assume a "kernel" function K at every point, a "Gaussian" in this case

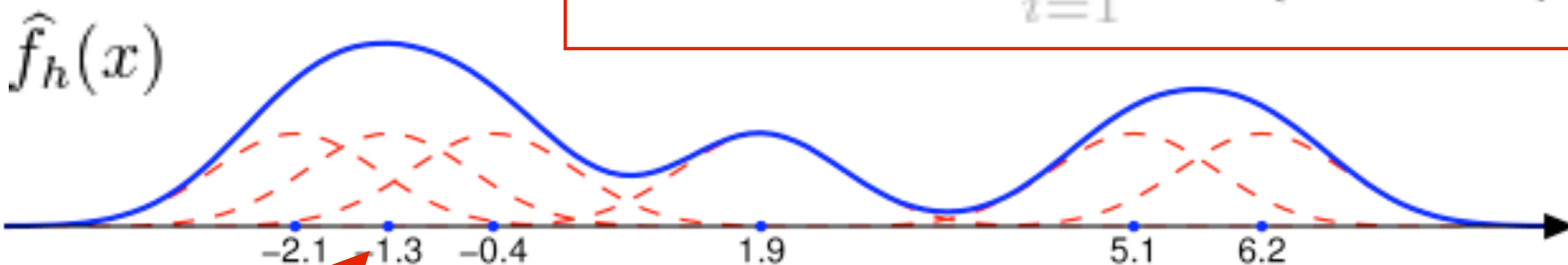
$$\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Representing belief as particles

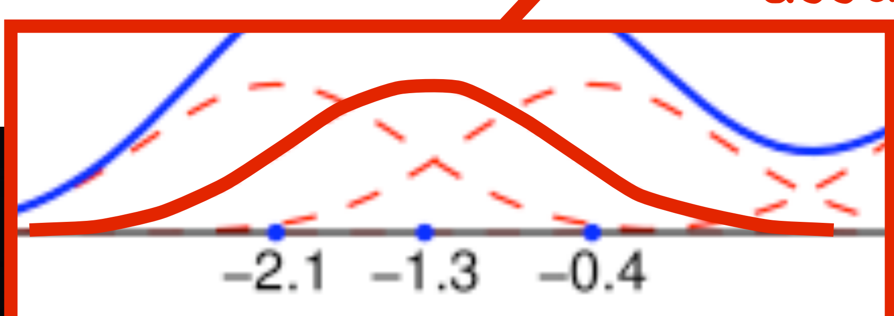
Tangent: How do particles form a distribution?

kernel shape results from evaluating one K across all x

$$\hat{f}_h(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$



assume a “kernel” function K at every point, a “Gaussian” in this case



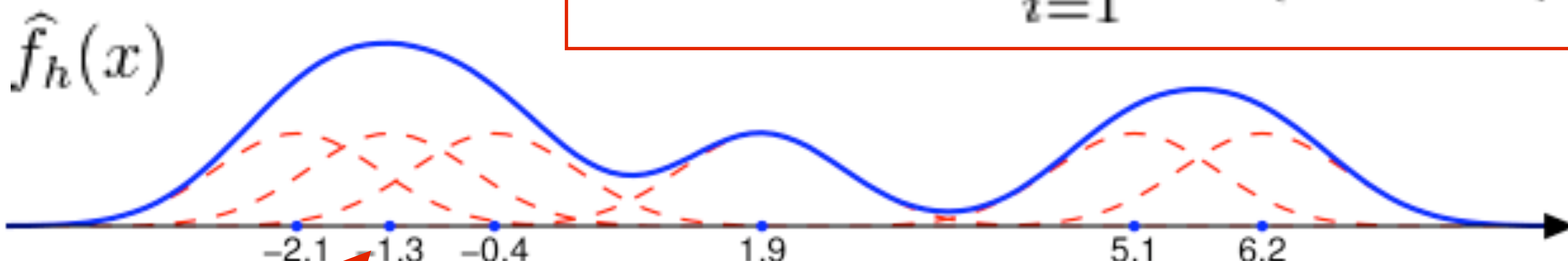
$$\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

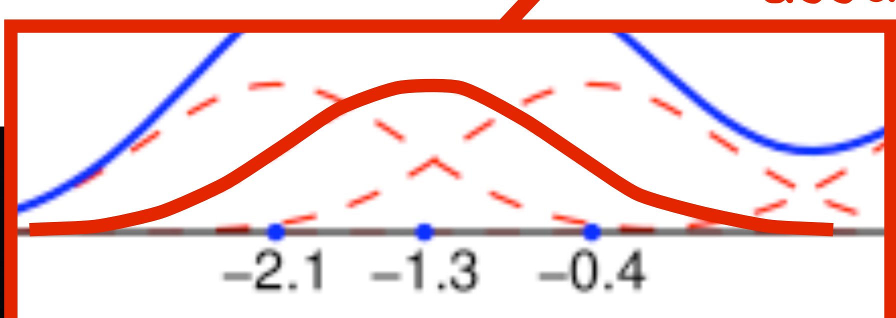
Representing belief as particles

Tangent: How do particles form a distribution?

distribution results from integrating over all kernels K across all x

$$\hat{f}_h(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

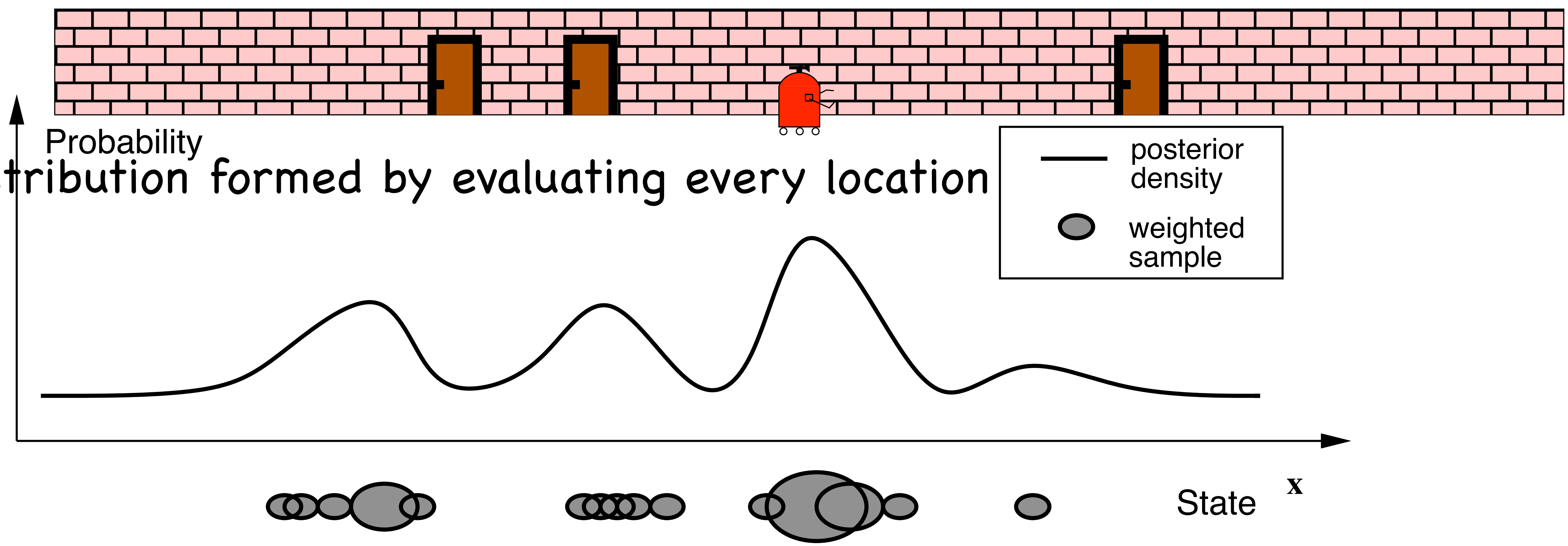




assume a "kernel" function K at every point, a "Gaussian" in this case

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

Representing belief as particles

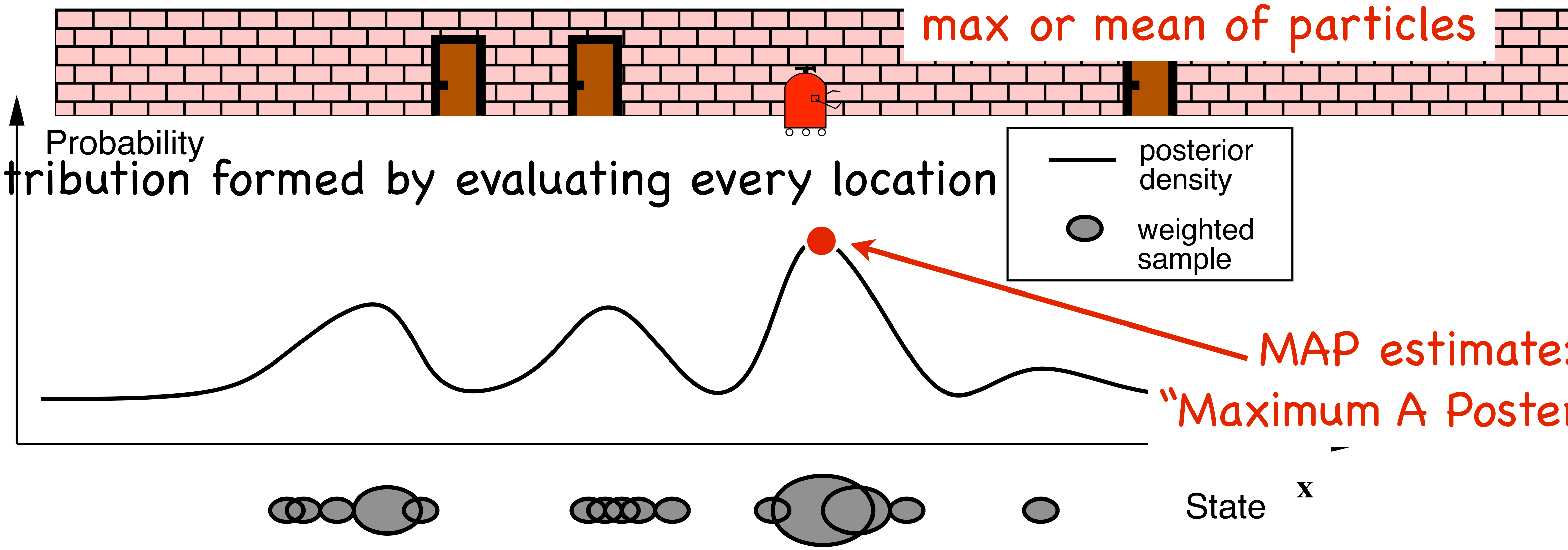


distribution formed by evaluating every location

each particle is a hypothesis of robot's pose, weighted by the probability of the pose

Representing belief as particles

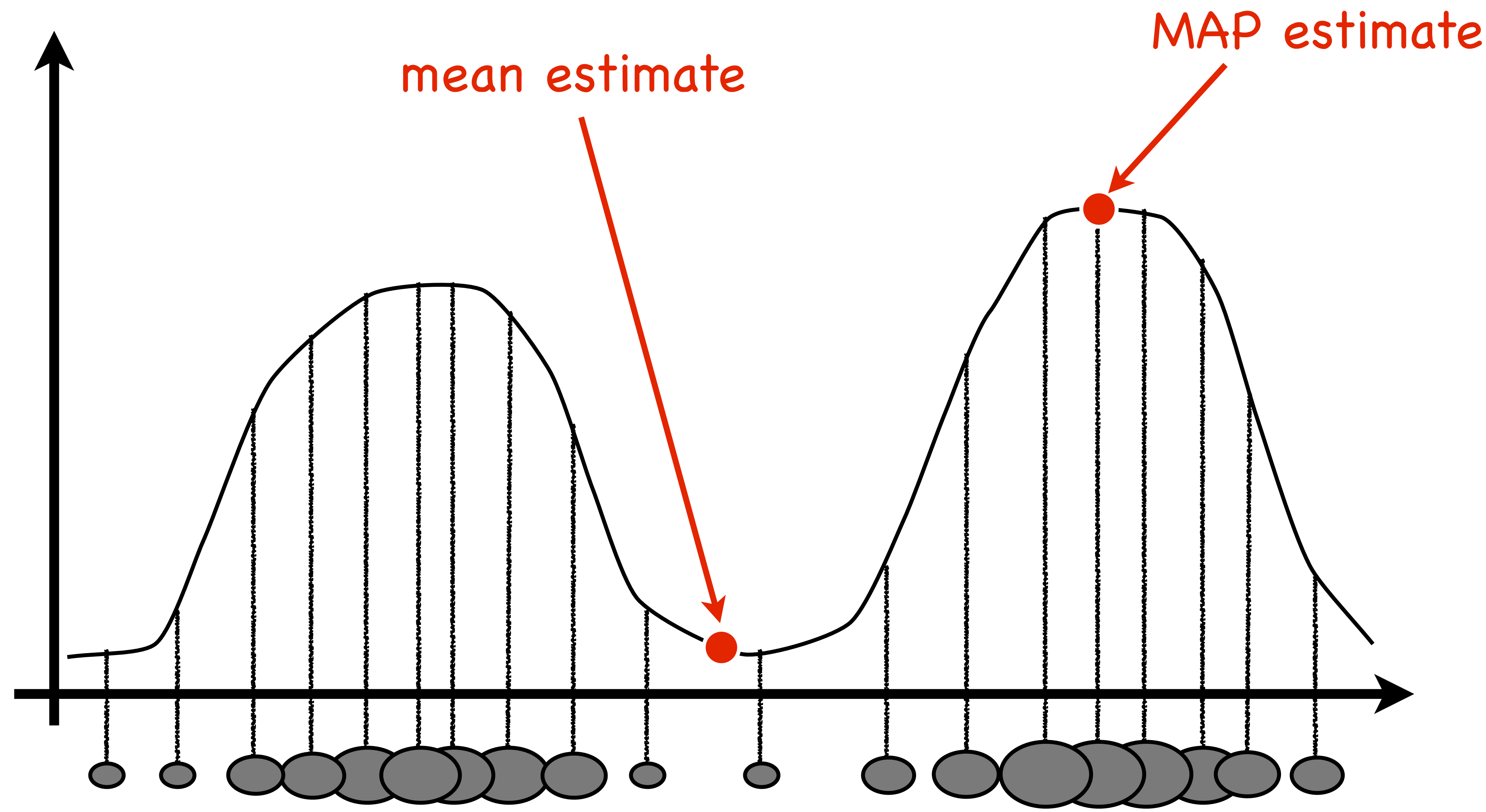
estimate robot pose as max or mean of particles



distribution formed by evaluating every location

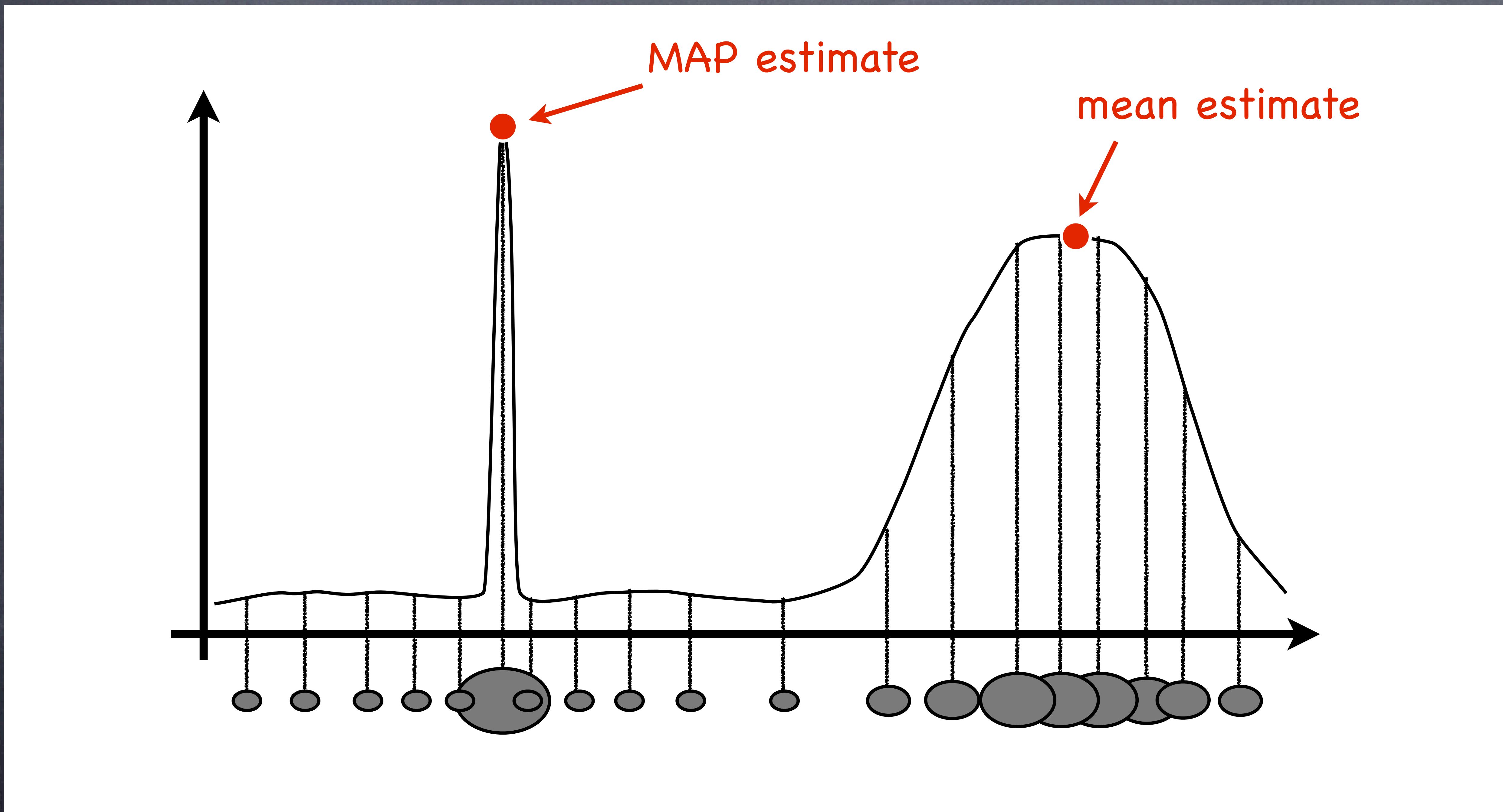
MAP estimate: “Maximum A Posteriori”

each particle is a hypothesis of robot’s pose, weighted by the probability of the pose


mean estimate
MAP estimate

Also mention robust mean: avg. of points around MAP

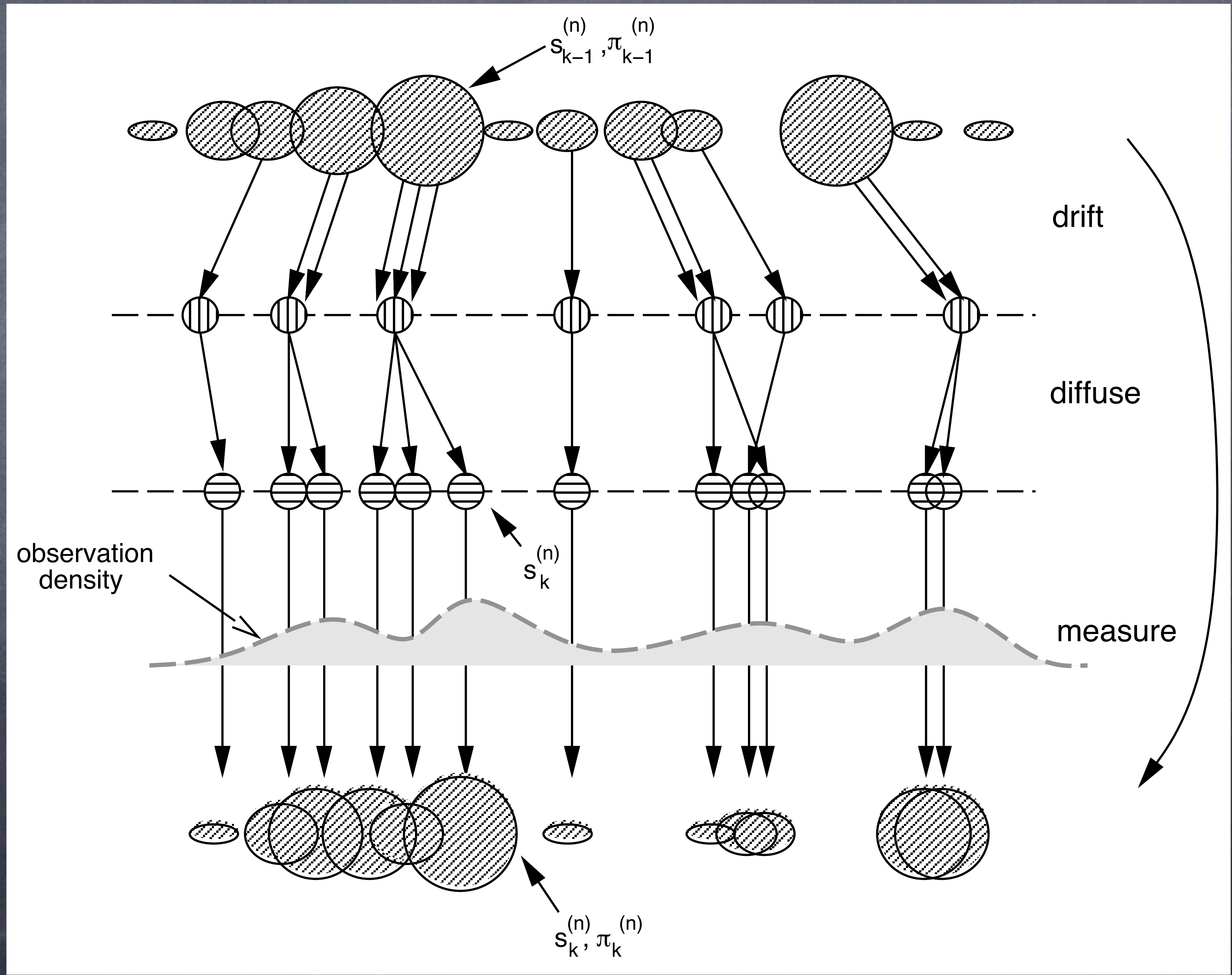
$$\bar{\mathbf{x}}_s = \sum_{j=1}^{K} \mathbf{x}_s^j w_j : |\mathbf{x}_s^j - \mathbf{x}_s^{max}| \leq \epsilon$$



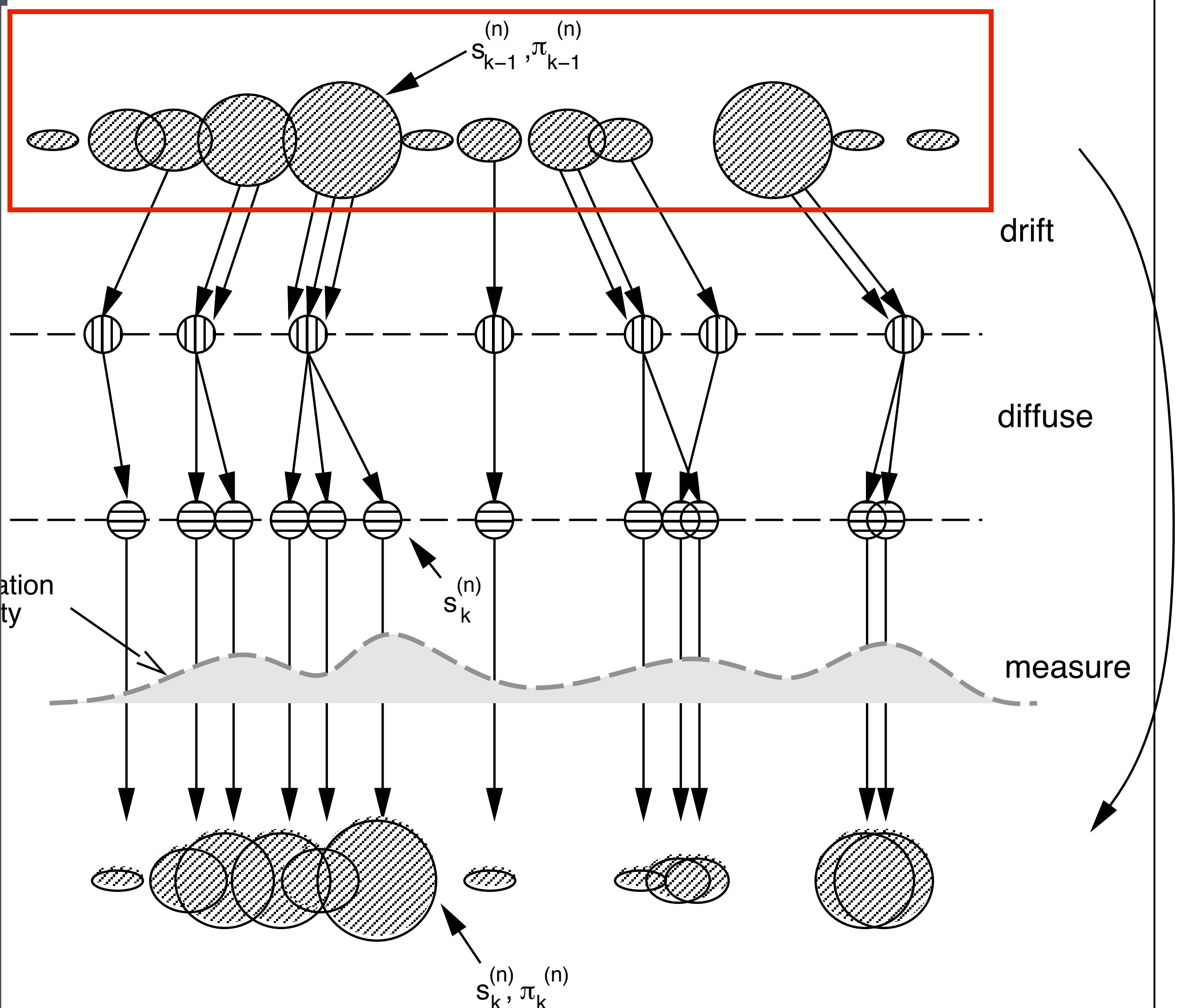
Condensation Algorithm

[Isard, Blake 1998]

- Condensation is one algorithm for particle filtering



sample set:
particle locations and weights



Iterate

From the "old" sample-set $\{\mathbf{s}_{t-1}^{(n)}, \pi_{t-1}^{(n)}, c_{t-1}^{(n)}, n = 1, \ldots, N\}$ at time-step $t-1$, construct a "new" sample-set $\{\mathbf{s}_t^{(n)}, \pi_t^{(n)}, c_t^{(n)}\}, n = 1, \ldots, N$ for time t.

Construct the n^{th} of N new samples as follows:

1. **Select** a sample $\mathbf{s}_t'^{(n)}$ as follows:

   (a) generate a random number $r \in [0, 1]$, uniformly distributed.

   (b) find, by binary subdivision, the smallest j for which $c_{t-1}^{(j)} \geq r$

   (c) set $\mathbf{s}_t'^{(n)} = \mathbf{s}_{t-1}^{(j)}$

2. **Predict** by sampling from

$$p(\mathbf{x}_t | \mathbf{x}_{t-1} = \mathbf{s}_t'^{(n)})$$

   to choose each $\mathbf{s}_t^{(n)}$. For instance, in the case that the dynamics are governed by a linear stochastic differential equation, the new sample value may be generated as: $\mathbf{s}_t^{(n)} = \mathbf{A}\mathbf{s}_t'^{(n)} + B\mathbf{w}_t^{(n)}$ where $\mathbf{w}_t^{(n)}$ is a vector of standard normal random variates, and BB^T is the process noise covariance — see section 5.

3. **Measure** and weight the new position in terms of the measured features $\mathbf{z}_t$:

$$\pi_t^{(n)} = p(\mathbf{z}_t | \mathbf{x}_t = \mathbf{s}_t^{(n)})$$

   then normalise so that $\sum_n \pi_t^{(n)} = 1$ and store together with cumulative probability as $(\mathbf{s}_t^{(n)}, \pi_t^{(n)}, c_t^{(n)})$ where

$$\begin{aligned} c_t^{(0)} &= 0, \\ c_t^{(n)} &= c_t^{(n-1)} + \pi_t^{(n)} \quad (n = 1, \ldots, N). \end{aligned}$$

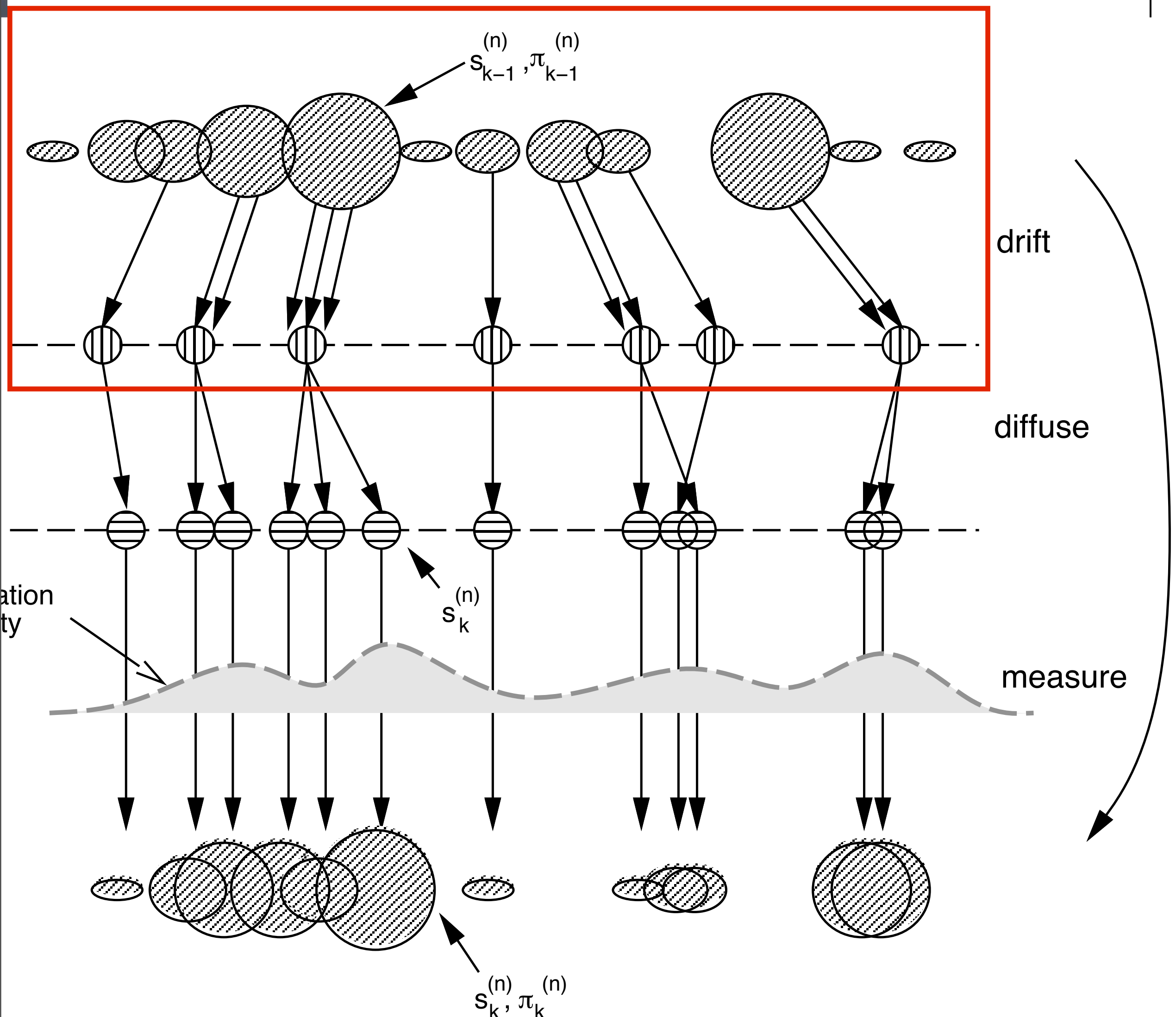
Once the N samples have been constructed: **estimate**, if desired, moments of the tracked position at time-step t as

$$\mathcal{E}[f(\mathbf{x}_t)] = \sum_{n=1}^{N} \pi_t^{(n)} f\left(\mathbf{s}_t^{(n)}\right)$$

obtaining, for instance, a mean position using $f(\mathbf{x}) = \mathbf{x}$.

Figure 6: **The** Condensation **algorithm**

create new particle set based on "importance" of current particles



Iterate

From the "old" sample-set $\{\mathbf{s}_{t-1}^{(n)}, \pi_{t-1}^{(n)}, c_{t-1}^{(n)}, n = 1, \ldots, N\}$ at time-step $t-1$, construct a "new" sample-set $\{\mathbf{s}_t^{(n)}, \pi_t^{(n)}, c_t^{(n)}\}, n = 1, \ldots, N$ for time t.

Construct the n^{th} of N new samples as follows:

1. **Select** a sample $\mathbf{s}_t'^{(n)}$ as follows:
   (a) generate a random number $r \in [0, 1]$, uniformly distributed.
   (b) find, by binary subdivision, the smallest j for which $c_{t-1}^{(j)} \geq r$
   (c) set $\mathbf{s}_t'^{(n)} = \mathbf{s}_{t-1}^{(j)}$
2. **Predict** by sampling from

$$p(\mathbf{x}_t | \mathbf{x}_{t-1} = \mathbf{s}_t'^{(n)})$$

   to choose each $\mathbf{s}_t^{(n)}$. For instance, in the case that the dynamics are governed by a linear stochastic differential equation, the new sample value may be generated as: $\mathbf{s}_t^{(n)} = \mathbf{A}\mathbf{s}_t'^{(n)} + B\mathbf{w}_t^{(n)}$ where $\mathbf{w}_t^{(n)}$ is a vector of standard normal random variates, and BB^T is the process noise covariance — see section 5.
3. **Measure** and weight the new position in terms of the measured features $\mathbf{z}_t$:

$$\pi_t^{(n)} = p(\mathbf{z}_t | \mathbf{x}_t = \mathbf{s}_t^{(n)})$$

   then normalise so that $\sum_n \pi_t^{(n)} = 1$ and store together with cumulative probability as $(\mathbf{s}_t^{(n)}, \pi_t^{(n)}, c_t^{(n)})$ where

$$\begin{aligned} c_t^{(0)} &= 0, \\ c_t^{(n)} &= c_t^{(n-1)} + \pi_t^{(n)} \quad (n = 1, \ldots, N). \end{aligned}$$

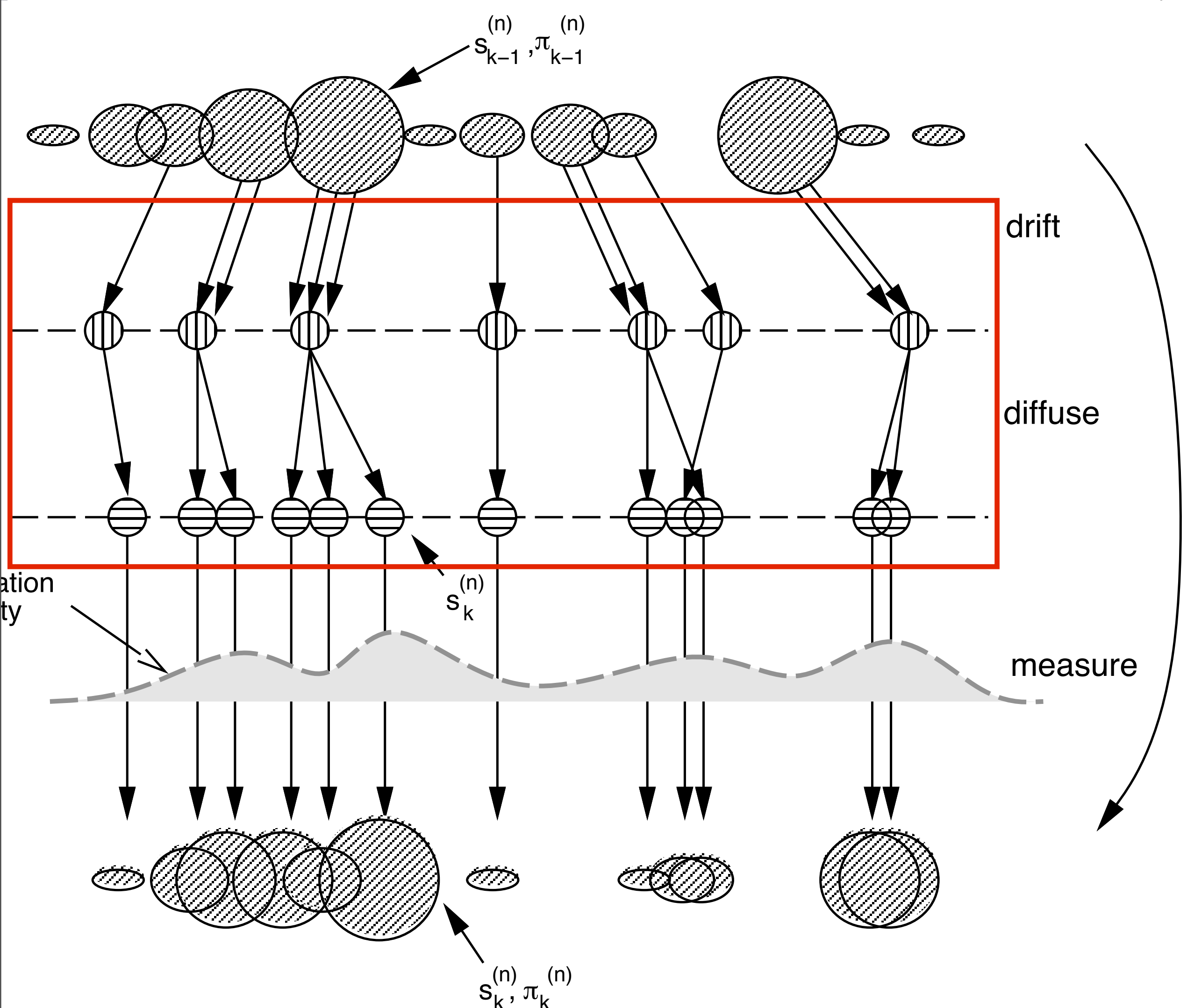
Once the N samples have been constructed: **estimate**, if desired, moments of the tracked position at time-step t as

$$\mathcal{E}[f(\mathbf{x}_t)] = \sum_{n=1}^{N} \pi_t^{(n)} f\left(\mathbf{s}_t^{(n)}\right)$$

obtaining, for instance, a mean position using $f(\mathbf{x}) = \mathbf{x}$.

Figure 6. The CONDENSATION algorithm.

predict movement of particles based on dynamics with noise



Iterate

From the "old" sample-set $\{\mathbf{s}_{t-1}^{(n)}, \pi_{t-1}^{(n)}, c_{t-1}^{(n)}, n = 1, \ldots, N\}$ at time-step $t-1$, construct a "new" sample-set $\{\mathbf{s}_t^{(n)}, \pi_t^{(n)}, c_t^{(n)}\}, n = 1, \ldots, N$ for time t.

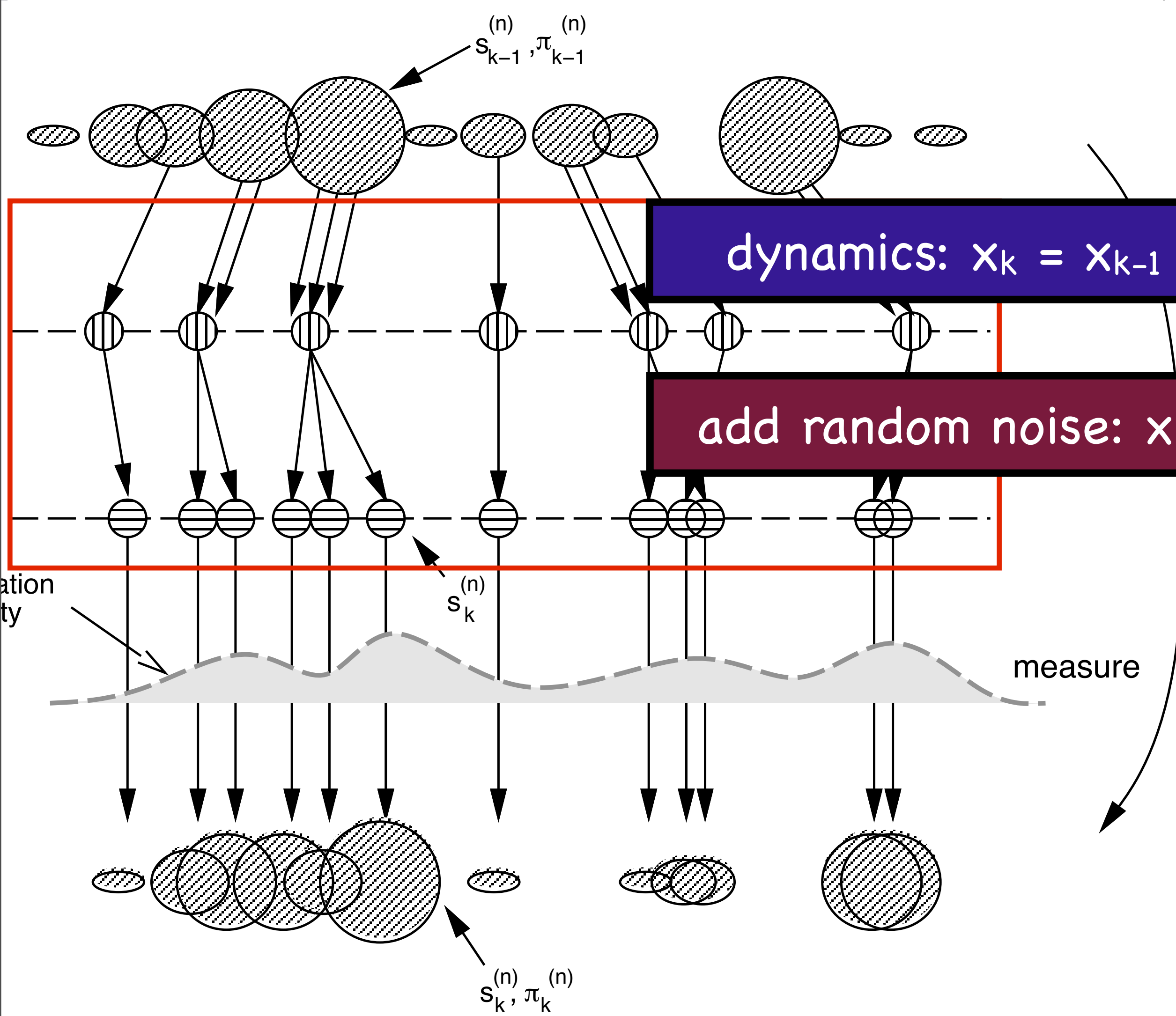
Construct the n^{th} of N new samples as follows:

1. **Select** a sample $\mathbf{s}_t'^{(n)}$ as follows:
   (a) generate a random number $r \in [0, 1]$, uniformly distributed.
   (b) find, by binary subdivision, the smallest j for which $c_{t-1}^{(j)} \geq r$
   (c) set $\mathbf{s}_t'^{(n)} = \mathbf{s}_{t-1}^{(j)}$
2. **Predict** by sampling from
$$p(\mathbf{x}_t|\mathbf{x}_{t-1} = \mathbf{s}_t'^{(n)})$$
to choose each $\mathbf{s}_t^{(n)}$. For instance, in the case that the dynamics are governed by a linear stochastic differential equation, the new sample value may be generated as: $\mathbf{s}_t^{(n)} = \mathbf{A}\mathbf{s}_t'^{(n)} + B\mathbf{w}_t^{(n)}$ where $\mathbf{w}_t^{(n)}$ is a vector of standard normal random variates, and BB^T is the process noise covariance — see section 5.
3. **Measure** and weight the new position in terms of the measured features $\mathbf{z}_t$:
$$\pi_t^{(n)} = p(\mathbf{z}_t|\mathbf{x}_t = \mathbf{s}_t^{(n)})$$
then normalise so that $\sum_n \pi_t^{(n)} = 1$ and store together with cumulative probability as $(\mathbf{s}_t^{(n)}, \pi_t^{(n)}, c_t^{(n)})$ where
$$\begin{aligned} c_t^{(0)} &= 0, \\ c_t^{(n)} &= c_t^{(n-1)} + \pi_t^{(n)} \quad (n = 1, \ldots, N). \end{aligned}$$

Once the N samples have been constructed: **estimate**, if desired, moments of the tracked position at time-step t as
$$\mathcal{E}[f(\mathbf{x}_t)] = \sum_{n=1}^{N} \pi_t^{(n)} f\left(\mathbf{s}_t^{(n)}\right)$$
obtaining, for instance, a mean position using $f(\mathbf{x}) = \mathbf{x}$.

Figure 6. The CONDENSATION algorithm.

predict movement of particles based on dynamics with noise



dynamics: $x_k = x_{k-1} + f(x_{k-1}, x_{k-1})$

add random noise: $x_k = x_k + \varepsilon$

(odometry for our localization)

Iterate

From the "old" sample-set $\{\mathbf{s}_{t-1}^{(n)}, \pi_{t-1}^{(n)}, c_{t-1}^{(n)}, n = 1, \ldots, N\}$ at time-step $t-1$, construct a "new" sample-set $\{\mathbf{s}_t^{(n)}, \pi_t^{(n)}, c_t^{(n)}\}, n = 1, \ldots, N$ for time t.

Construct the n^{th} of N new samples as follows:

1. **Select** a sample $\mathbf{s}_t'^{(n)}$ as follows:

   (a) generate a random number $r \in [0, 1]$, uniformly distributed.

   (b) find, by binary subdivision, the smallest j for which $c_{t-1}^{(j)} \geq r$

   ... $\mathbf{s}_{t-1}^{(j)}$

2. **Predict** by sampling from

$$p(\mathbf{x}_t | \mathbf{x}_{t-1} = \mathbf{s}_t'^{(n)})$$

to choose each $\mathbf{s}_t^{(n)}$. For instance, in the case that the dynamics are governed by a linear stochastic differential equation, the new sample value may be generated as: $\mathbf{s}_t^{(n)} = \mathbf{A}\mathbf{s}_t'^{(n)} + B\mathbf{w}_t^{(n)}$ where $\mathbf{w}_t^{(n)}$ is a vector of standard normal random variates, and BB^T is the process noise covariance — see section 5.

3. **Measure** and weight the new position in terms of the measured features $\mathbf{z}_t$:

$$\pi_t^{(n)} = p(\mathbf{z}_t | \mathbf{x}_t = \mathbf{s}_t^{(n)})$$

then normalise so that $\sum_n \pi_t^{(n)} = 1$ and store together with cumulative probability as $(\mathbf{s}_t^{(n)}, \pi_t^{(n)}, c_t^{(n)})$ where

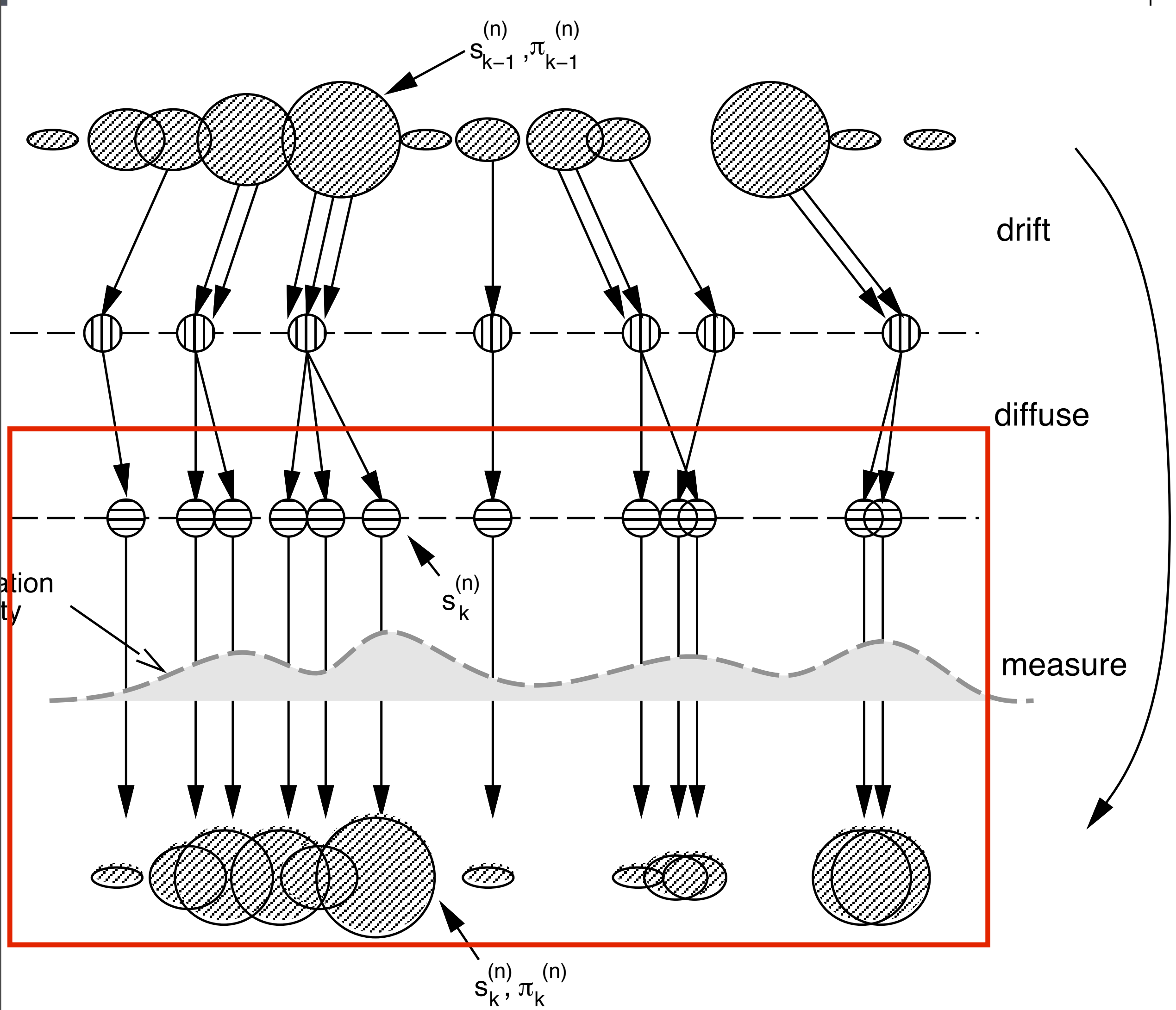
$$\begin{aligned} c_t^{(0)} &= 0, \\ c_t^{(n)} &= c_t^{(n-1)} + \pi_t^{(n)} \quad (n = 1, \ldots, N). \end{aligned}$$

Once the N samples have been constructed: **estimate**, if desired, moments of the tracked position at time-step t as

$$\mathcal{E}[f(\mathbf{x}_t)] = \sum_{n=1}^{N} \pi_t^{(n)} f\left(\mathbf{s}_t^{(n)}\right)$$

obtaining, for instance, a mean position using $f(\mathbf{x}) = \mathbf{x}$.

update particle weights based on observed likelihood



Iterate

From the "old" sample-set $\{\mathbf{s}_{t-1}^{(n)}, \pi_{t-1}^{(n)}, c_{t-1}^{(n)}, n = 1, \ldots, N\}$ at time-step $t-1$, construct a "new" sample-set $\{\mathbf{s}_t^{(n)}, \pi_t^{(n)}, c_t^{(n)}\}, n = 1, \ldots, N$ for time t.

Construct the n^{th} of N new samples as follows:

1. **Select** a sample $\mathbf{s}_t'^{(n)}$ as follows:
   (a) generate a random number $r \in [0, 1]$, uniformly distributed.
   (b) find, by binary subdivision, the smallest j for which $c_{t-1}^{(j)} \geq r$
   (c) set $\mathbf{s}_t'^{(n)} = \mathbf{s}_{t-1}^{(j)}$
2. **Predict** by sampling from

$$p(\mathbf{x}_t | \mathbf{x}_{t-1} = \mathbf{s}_t'^{(n)})$$

   to choose each $\mathbf{s}_t^{(n)}$. For instance, in the case that the dynamics are governed by a linear stochastic differential equation, the new sample value may be generated as: $\mathbf{s}_t^{(n)} = \mathbf{A}\mathbf{s}_t'^{(n)} + B\mathbf{w}_t^{(n)}$ where $\mathbf{w}_t^{(n)}$ is a vector of standard normal random variates, and BB^T is the process noise covariance — see section 5.
3. **Measure** and weight the new position in terms of the measured features $\mathbf{z}_t$:

$$\pi_t^{(n)} = p(\mathbf{z}_t | \mathbf{x}_t = \mathbf{s}_t^{(n)})$$

   then normalise so that $\sum_n \pi_t^{(n)} = 1$ and store together with cumulative probability as $(\mathbf{s}_t^{(n)}, \pi_t^{(n)}, c_t^{(n)})$ where

$$\begin{aligned} c_t^{(0)} &= 0, \\ c_t^{(n)} &= c_t^{(n-1)} + \pi_t^{(n)} \quad (n = 1, \ldots, N). \end{aligned}$$

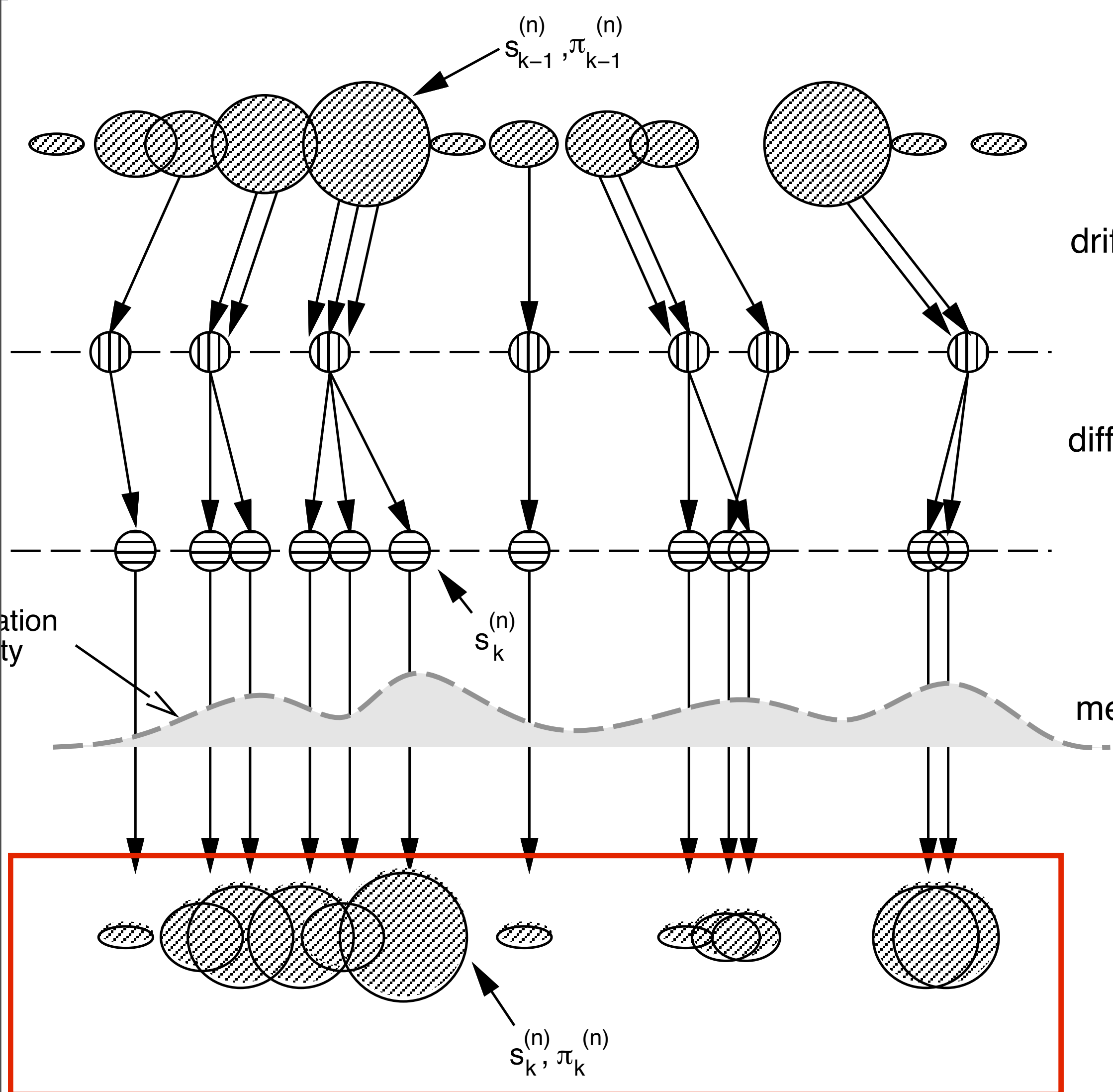
Once the N samples have been constructed: **estimate**, if desired, moments of the tracked position at time-step t as

$$\mathcal{E}[f(\mathbf{x}_t)] = \sum_{n=1}^{N} \pi_t^{(n)} f\left(\mathbf{s}_t^{(n)}\right)$$

obtaining, for instance, a mean position using $f(\mathbf{x}) = \mathbf{x}$.

Figure 6. The CONDENSATION algorithm

estimate state as the mean (weighted average) of particles



Iterate

From the "old" sample-set $\{\mathbf{s}_{t-1}^{(n)}, \pi_{t-1}^{(n)}, c_{t-1}^{(n)}, n = 1, \ldots, N\}$ at time-step $t-1$, construct a "new" sample-set $\{\mathbf{s}_t^{(n)}, \pi_t^{(n)}, c_t^{(n)}\}, n = 1, \ldots, N$ for time t.

Construct the n^{th} of N new samples as follows:

1. **Select** a sample $\mathbf{s}_t'^{(n)}$ as follows:
   (a) generate a random number $r \in [0, 1]$, uniformly distributed.
   (b) find, by binary subdivision, the smallest j for which $c_{t-1}^{(j)} \geq r$
   (c) set $\mathbf{s}_t'^{(n)} = \mathbf{s}_{t-1}^{(j)}$
2. **Predict** by sampling from

$$p(\mathbf{x}_t | \mathbf{x}_{t-1} = \mathbf{s}_t'^{(n)})$$

to choose each $\mathbf{s}_t^{(n)}$. For instance, in the case that the dynamics are governed by a linear stochastic differential equation, the new sample value may be generated as: $\mathbf{s}_t^{(n)} = \mathbf{A}\mathbf{s}_t'^{(n)} + B\mathbf{w}_t^{(n)}$ where $\mathbf{w}_t^{(n)}$ is a vector of standard normal random variates, and BB^T is the process noise covariance — see section 5.

3. **Measure** and weight the new position in terms of the measured features $\mathbf{z}_t$:

$$\pi_t^{(n)} = p(\mathbf{z}_t | \mathbf{x}_t = \mathbf{s}_t^{(n)})$$

then normalise so that $\sum_n \pi_t^{(n)} = 1$ and store together with cumulative probability as $(\mathbf{s}_t^{(n)}, \pi_t^{(n)}, c_t^{(n)})$ where

$$\begin{aligned} c_t^{(0)} &= 0, \\ c_t^{(n)} &= c_t^{(n-1)} + \pi_t^{(n)} \quad (n = 1, \ldots, N). \end{aligned}$$

Once the N samples have been constructed: **estimate**, if desired, moments of the tracked position at time-step t as

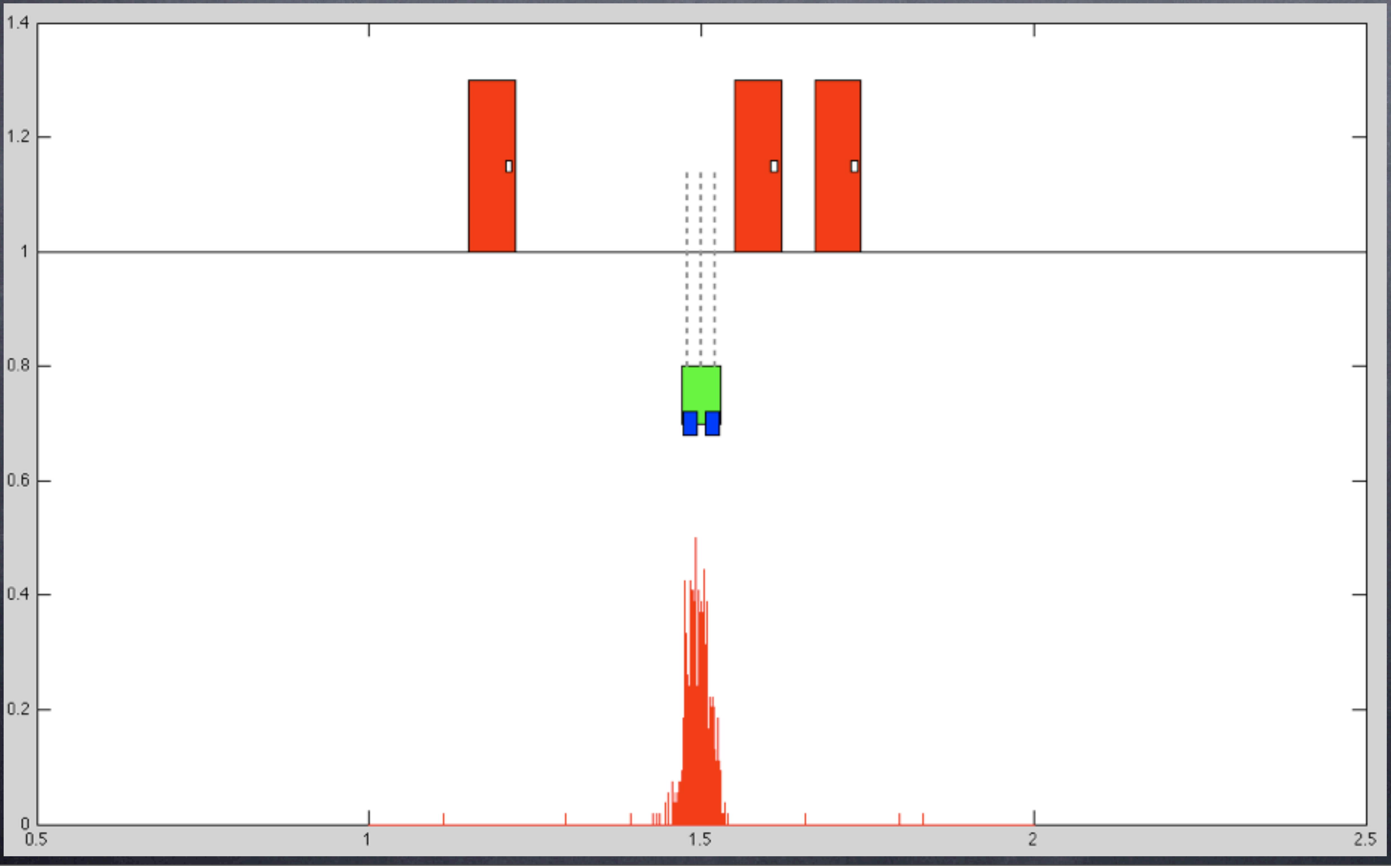
$$\mathcal{E}[f(\mathbf{x}_t)] = \sum_{n=1}^{N} \pi_t^{(n)} f\left(\mathbf{s}_t^{(n)}\right)$$

obtaining, for instance, a mean position using $f(\mathbf{x}) = \mathbf{x}$.

Figure 6. The Condensation algorithm.

matlab example

- /course/cs148/pub/particlefilter_hallway.m



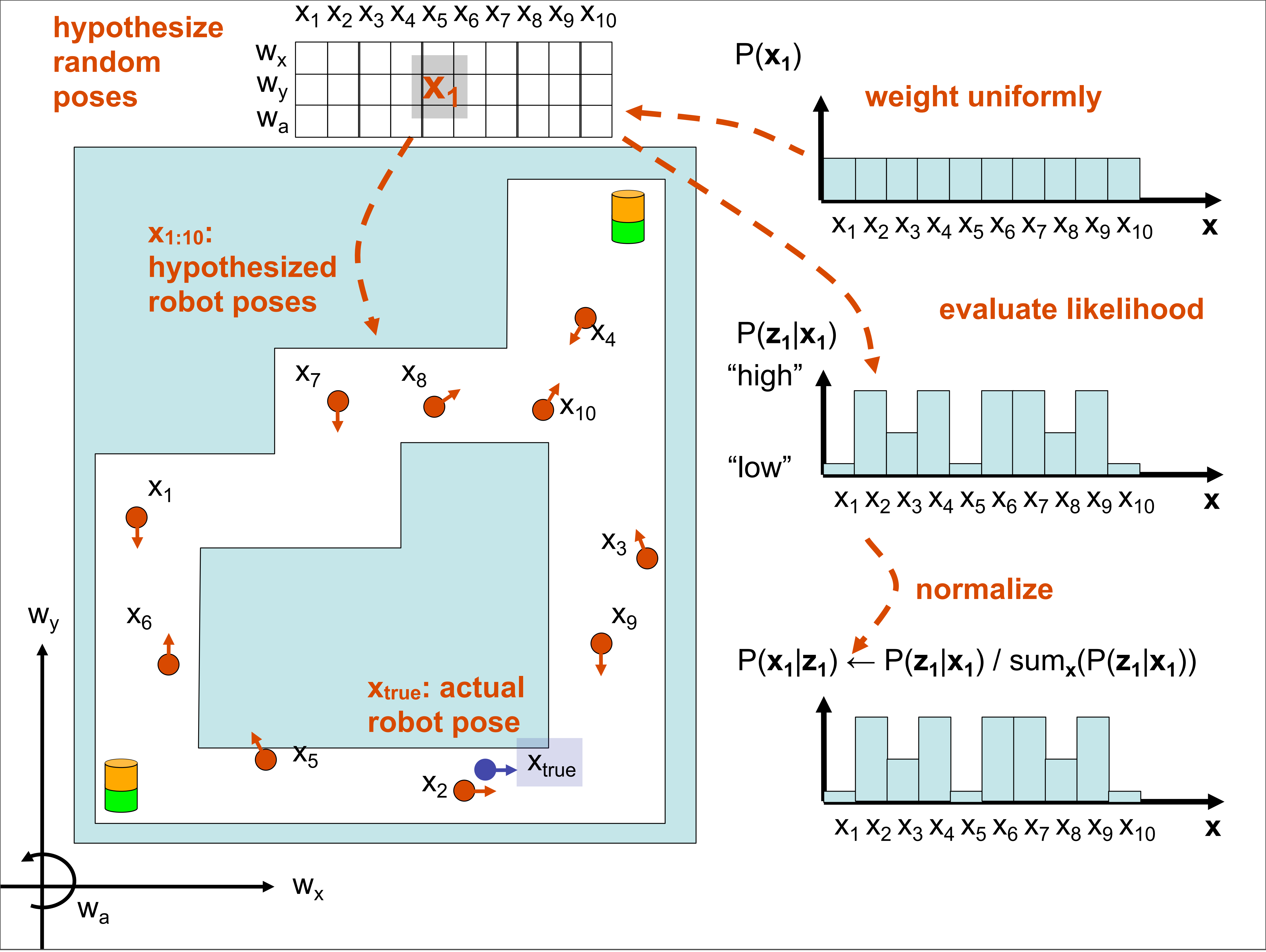
CIT 5th floor localization example (Roomba Pac-Man)

cs148 from Fall 2006

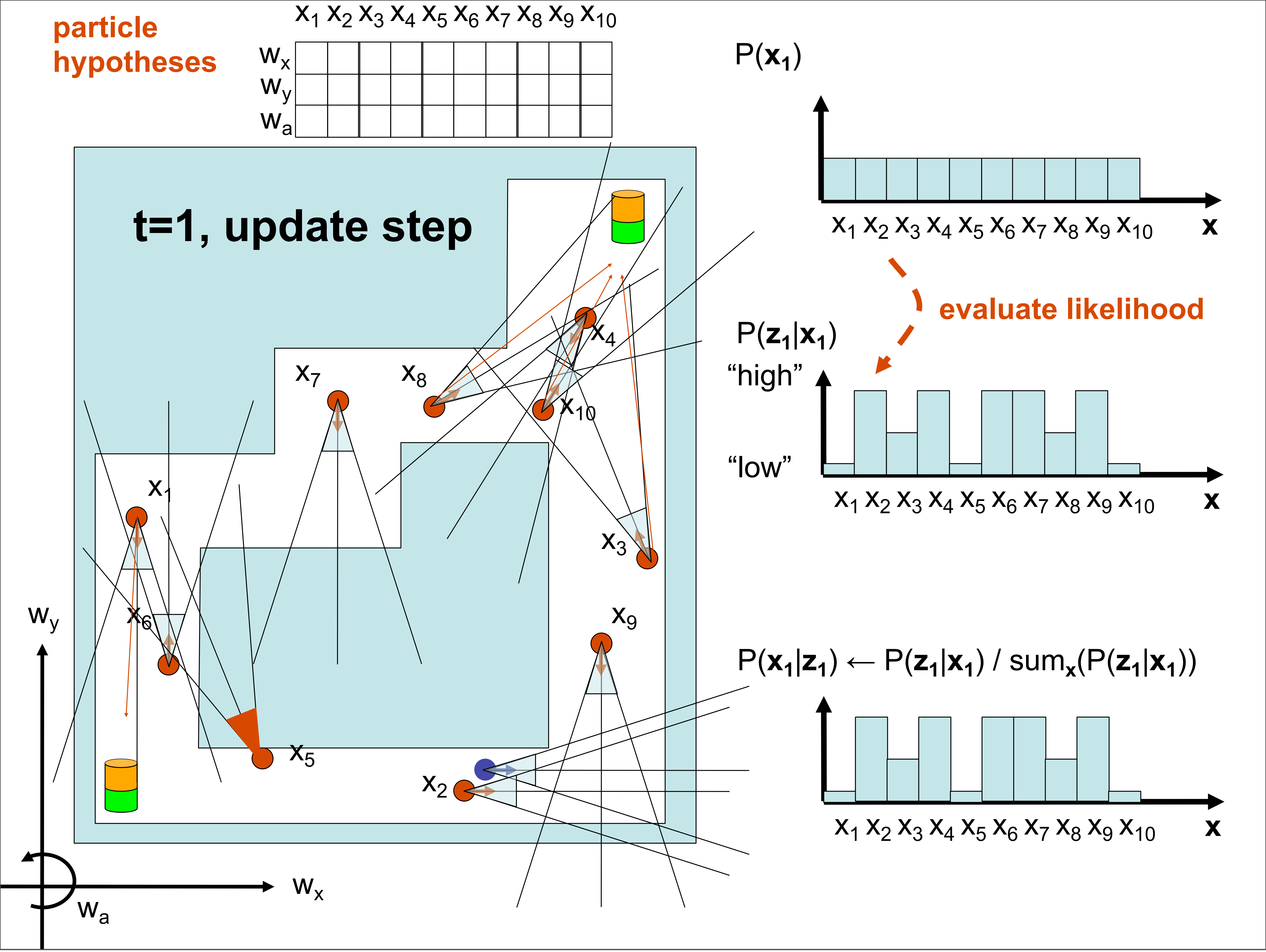
Roomba Pac-Man finding “power-up”

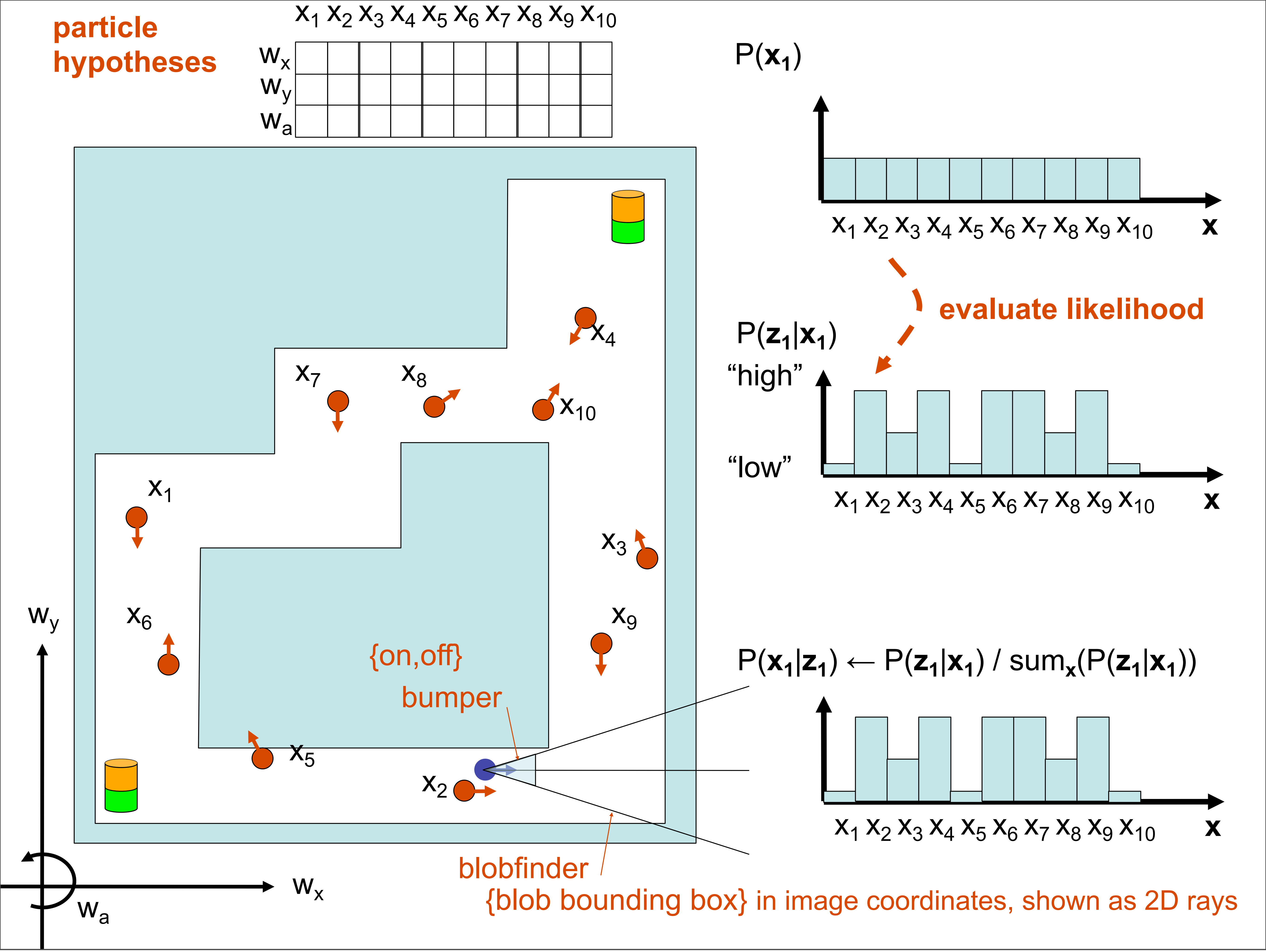
Roomba Pac-Man avoiding “ghost”

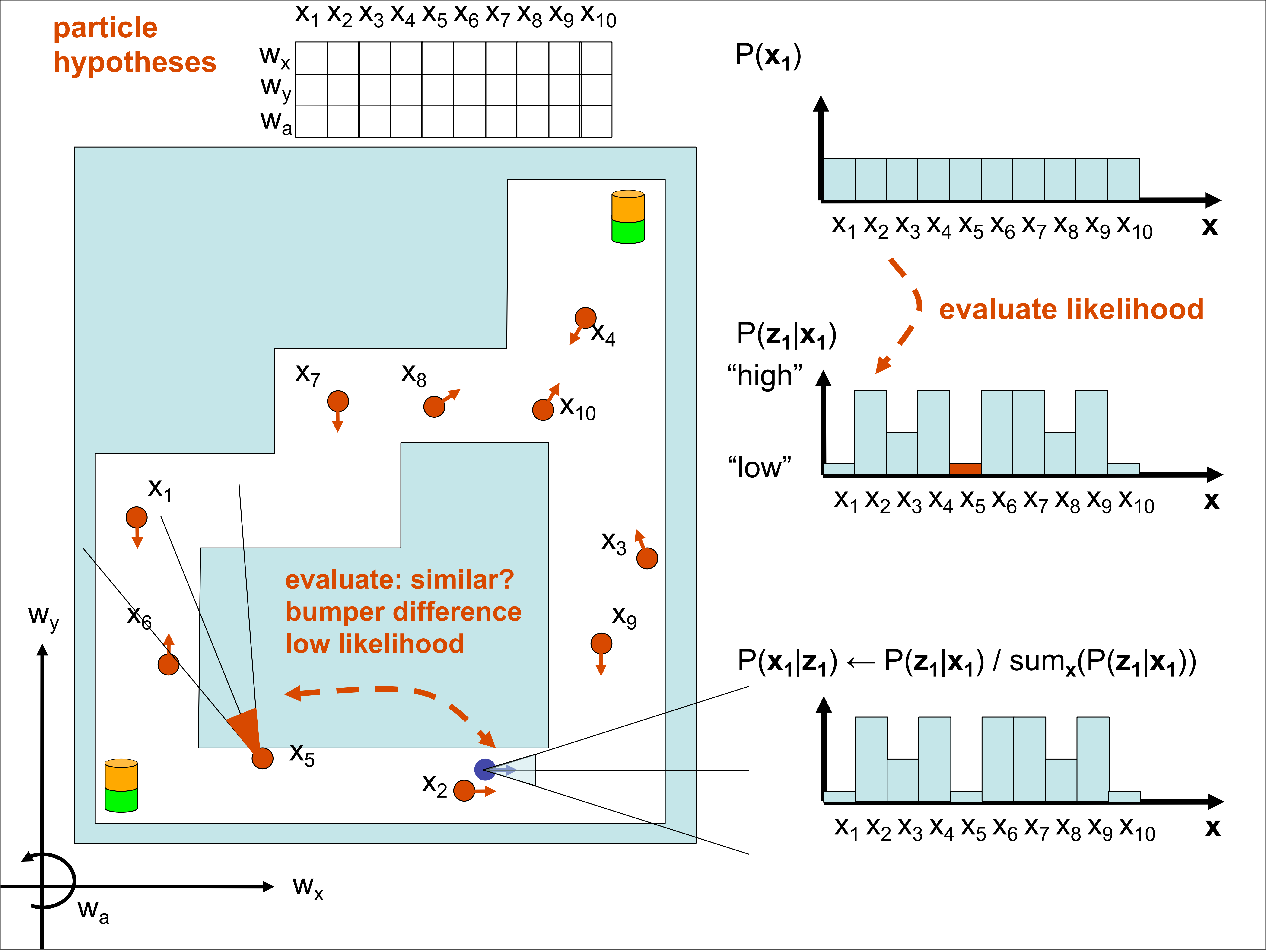
Roomba Pac-Man consuming “pellets”

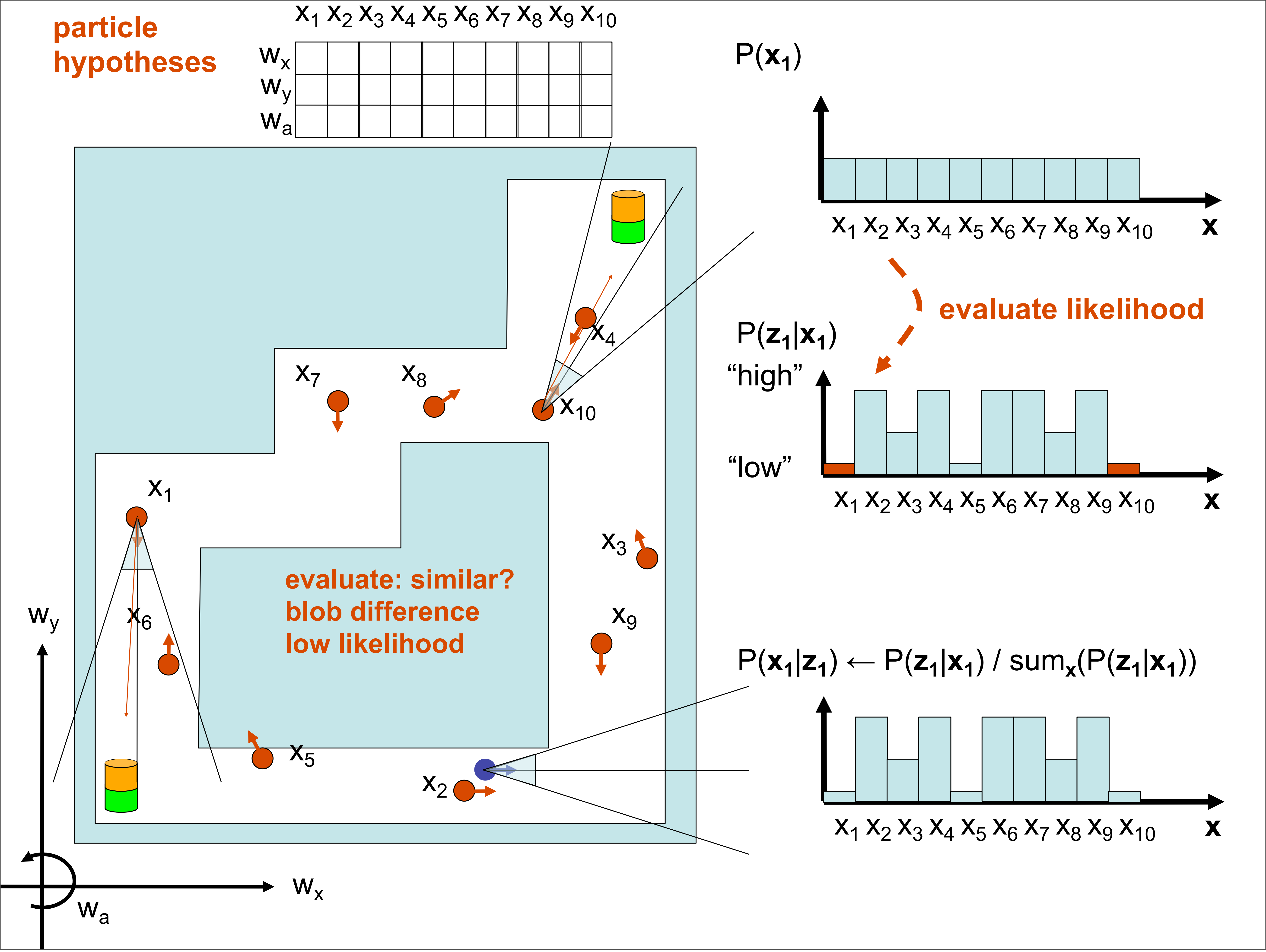


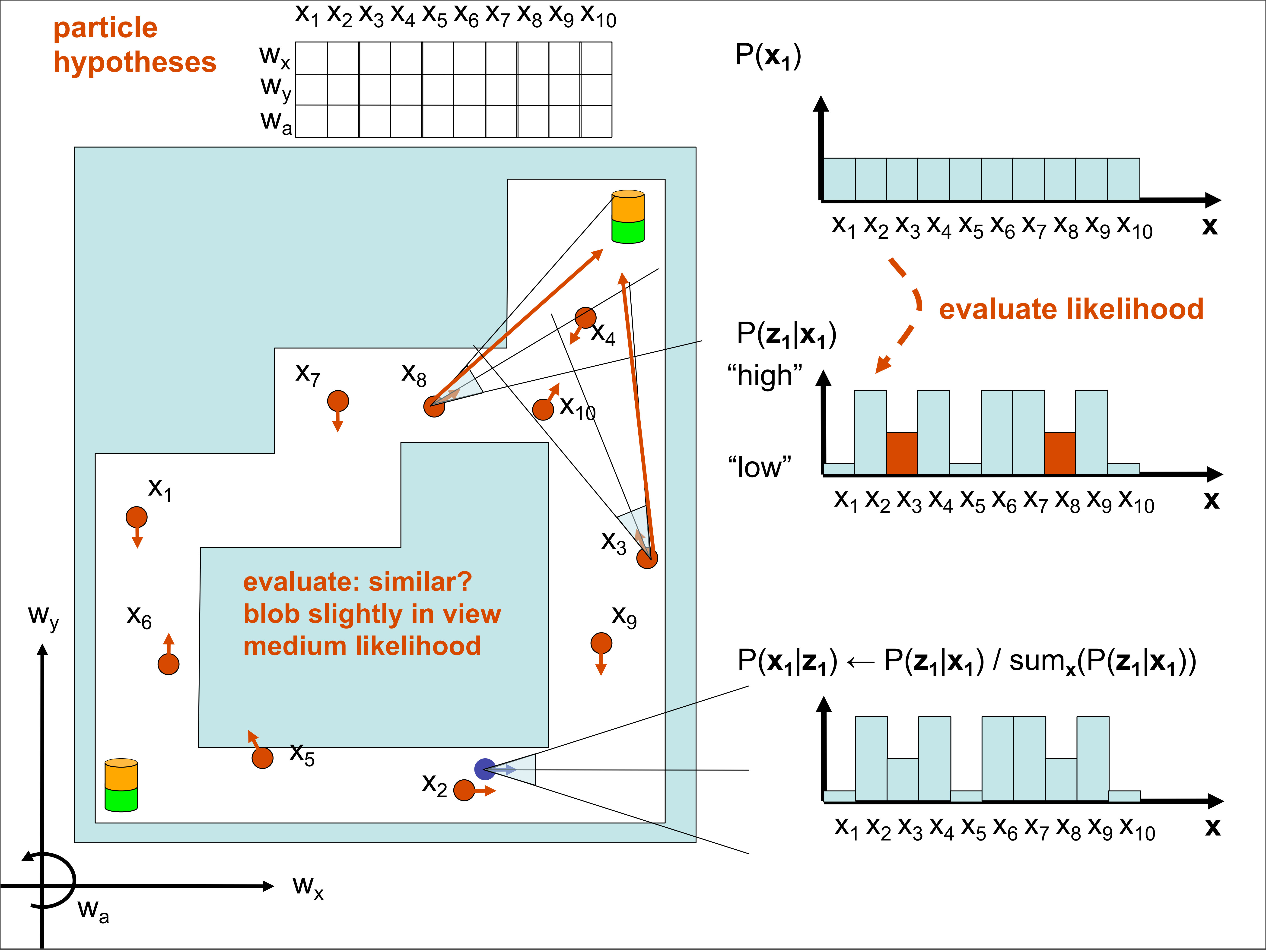
hypothesize random poses
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
wx
wy
wa
x1
P(x1)
weight uniformly
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
x
x1:10: hypothesized robot poses
x4
x7
x8
x10
evaluate likelihood
P(z1|x1)
"high"
"low"
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
x
x1
x3
normalize
x6
x9
wy
P(x1|z1) ← P(z1|x1) / sumx(P(z1|x1))
xtrue: actual robot pose
x5
xtrue
x2
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
x
wx
wa

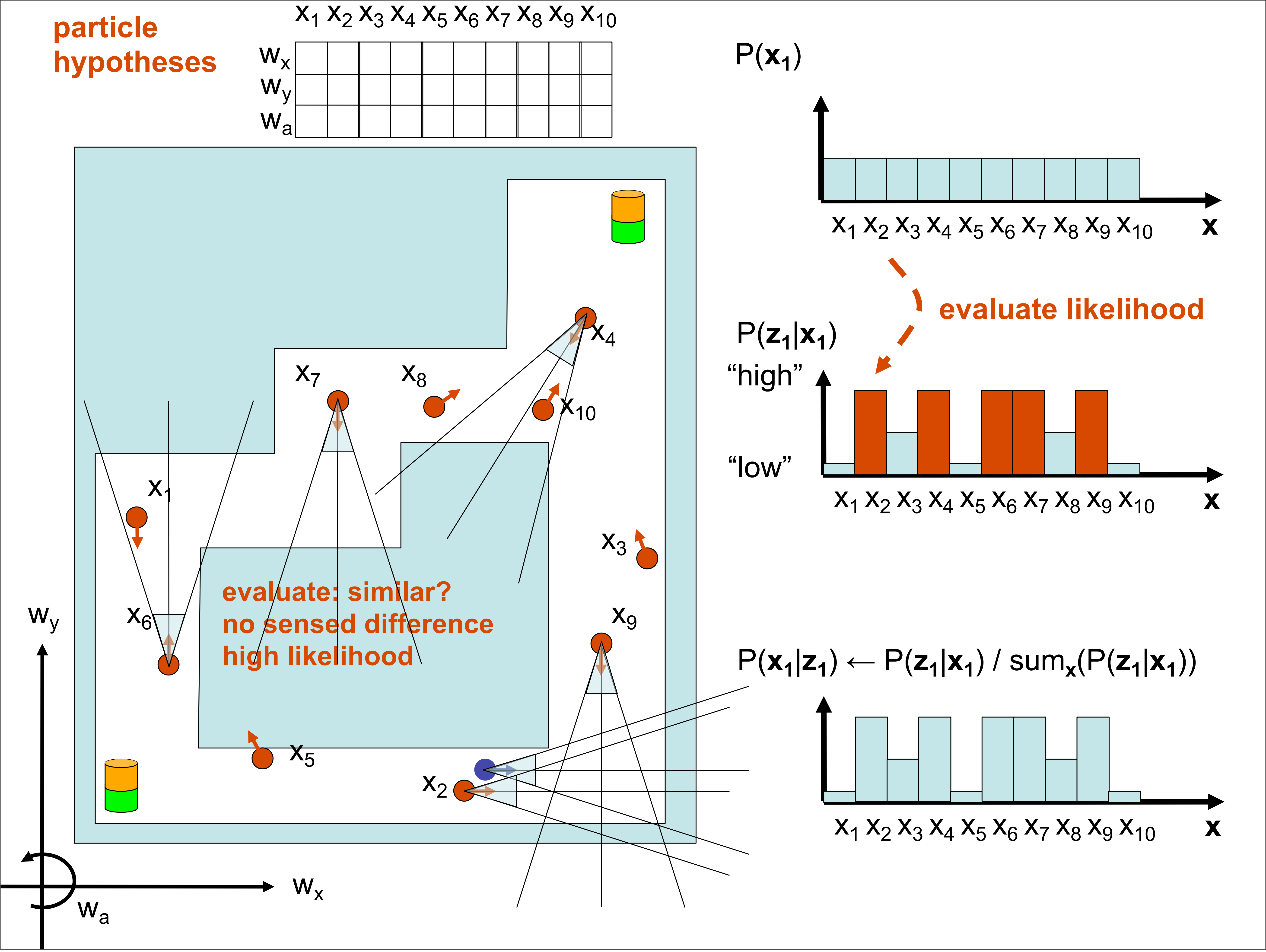

particle hypotheses
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
wx
wy
wa
t=1, update step
x7
x8
x4
x10
x1
x3
x6
x9
x5
x2
wy
wx
wa
P(x1)
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x
evaluate likelihood
P(z1|x1)
"high"
"low"
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x
P(x1|z1) ← P(z1|x1) / sumx(P(z1|x1))
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x


particle hypotheses
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
wx
wy
wa
P(x1)
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
x
evaluate likelihood
P(z1|x1)
"high"
"low"
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
x
P(x1|z1) ← P(z1|x1) / sumx(P(z1|x1))
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
x
x1
x2
x3
x4
x5
x6
x7
x8
x9
x10
{on,off}
bumper
wy
wx
wa
blobfinder
{blob bounding box} in image coordinates, shown as 2D rays

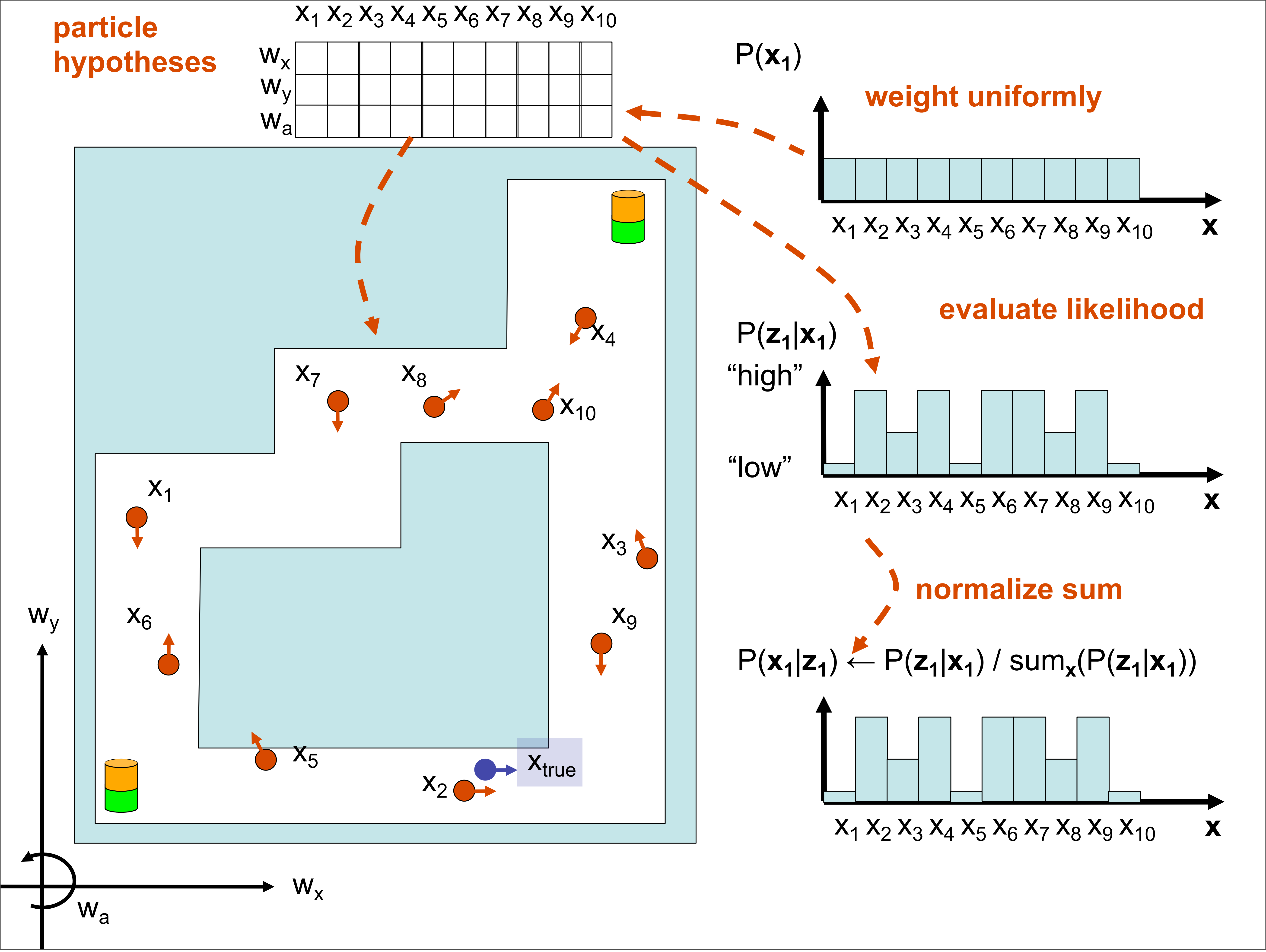

particle hypotheses
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
wx
wy
wa
P(x1)
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
x
evaluate likelihood
P(z1|x1)
"high"
"low"
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
x
x4
x7
x8
x10
x1
x3
evaluate: similar?
bumper difference
low likelihood
wy
x6
x9
x5
x2
P(x1|z1) ← P(z1|x1) / sumx(P(z1|x1))
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
x
wx
wa


particle hypotheses
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
wx
wy
wa
P(x1)
evaluate likelihood
P(z1|x1)
“high”
“low”
P(x1|z1) ← P(z1|x1) / sumx(P(z1|x1))
evaluate: similar?
blob difference
low likelihood
x
wy
wx
wa

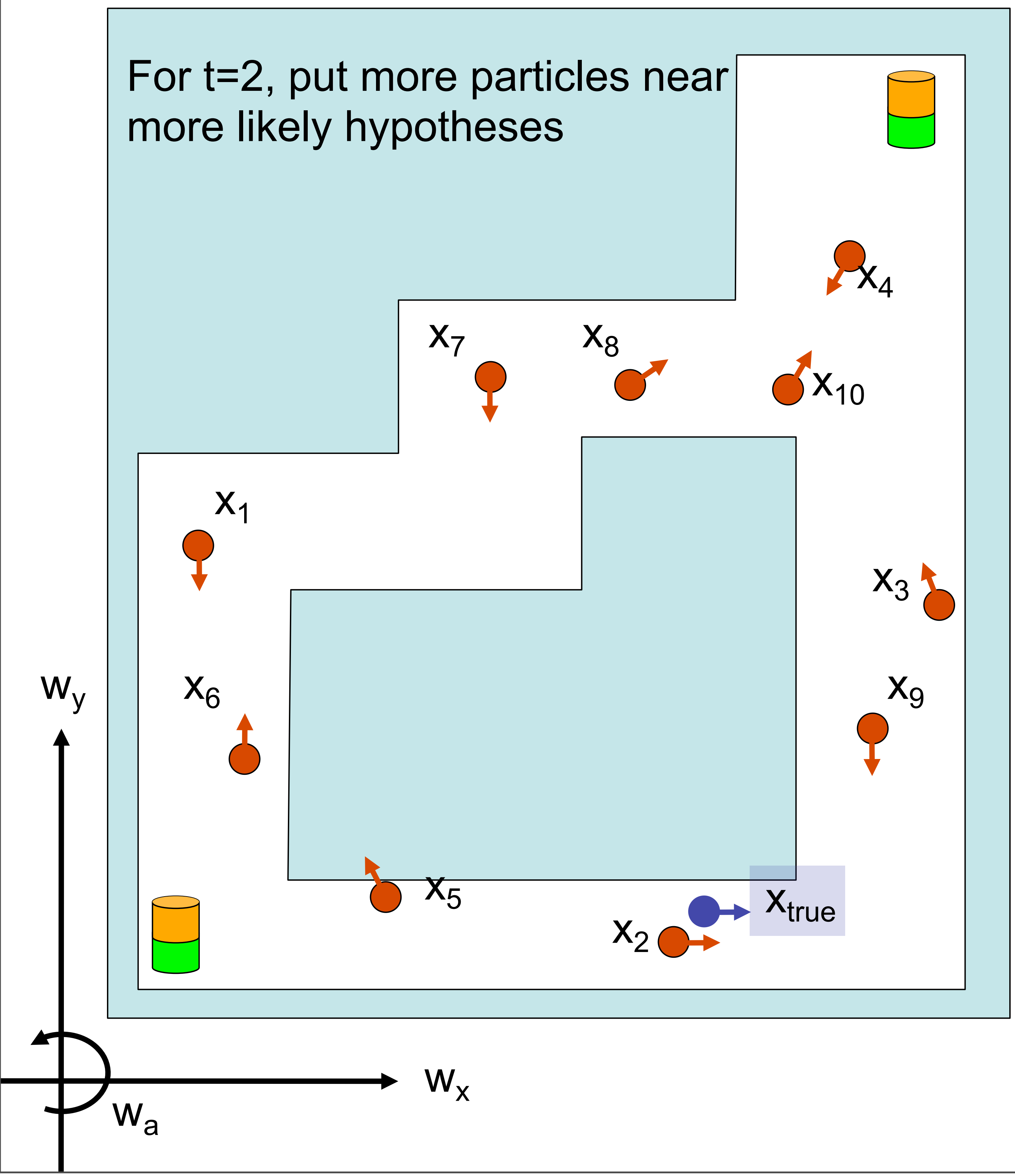

particle hypotheses
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
wx
wy
wa
P(x1)
evaluate likelihood
P(z1|x1)
"high"
"low"
P(x1|z1) ← P(z1|x1) / sumx(P(z1|x1))
evaluate: similar?
blob slightly in view
medium likelihood
x



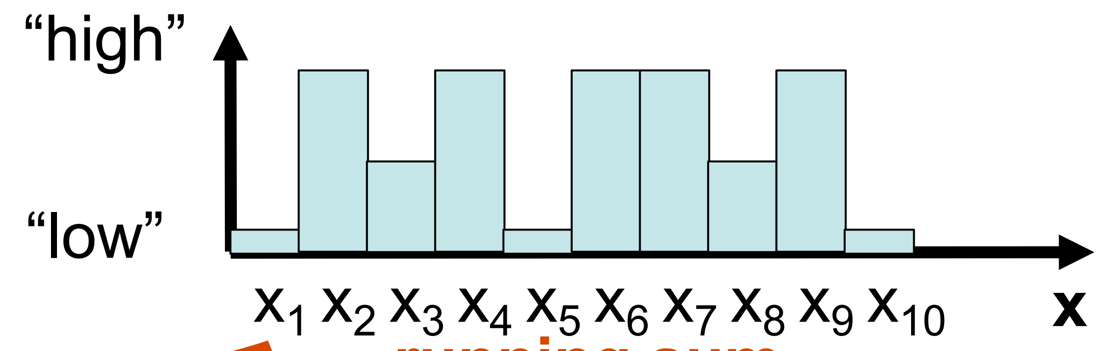
particle hypotheses
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
wx
wy
wa
P(x1)
evaluate likelihood
P(z1|x1)
"high"
"low"
P(x1|z1) ← P(z1|x1) / sumx(P(z1|x1))
x
evaluate: similar?
no sensed difference
high likelihood


particle hypotheses
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
wx
wy
wa
P(x1)
weight uniformly
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
x
evaluate likelihood
P(z1|x1)
"high"
"low"
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
x
normalize sum
P(x1|z1) ← P(z1|x1) / sumx(P(z1|x1))
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
x
x4
x7
x8
x10
x1
x3
x6
x9
x5
xtrue
x2
wy
wx
wa

t=1, predict step: Importance Sampling

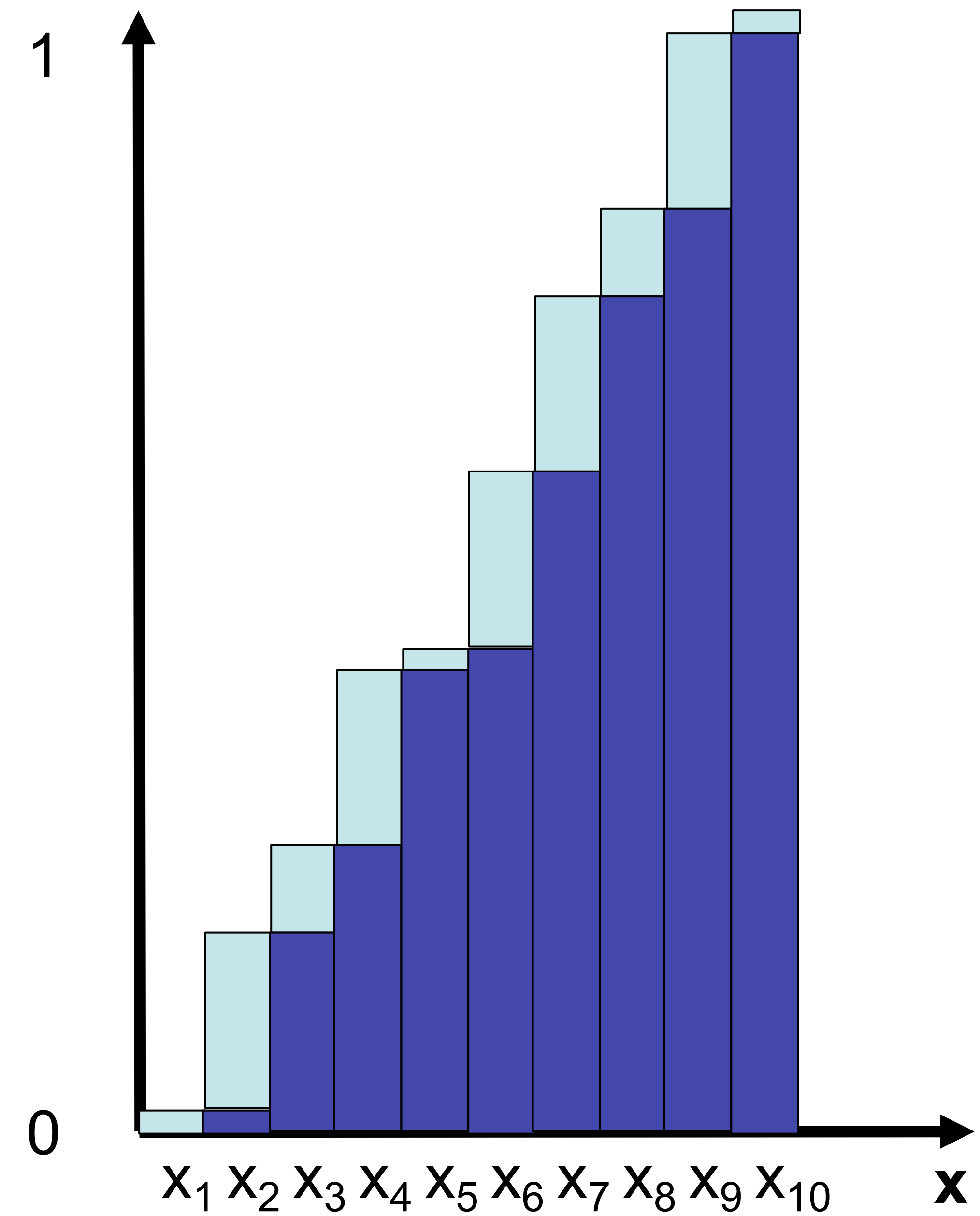


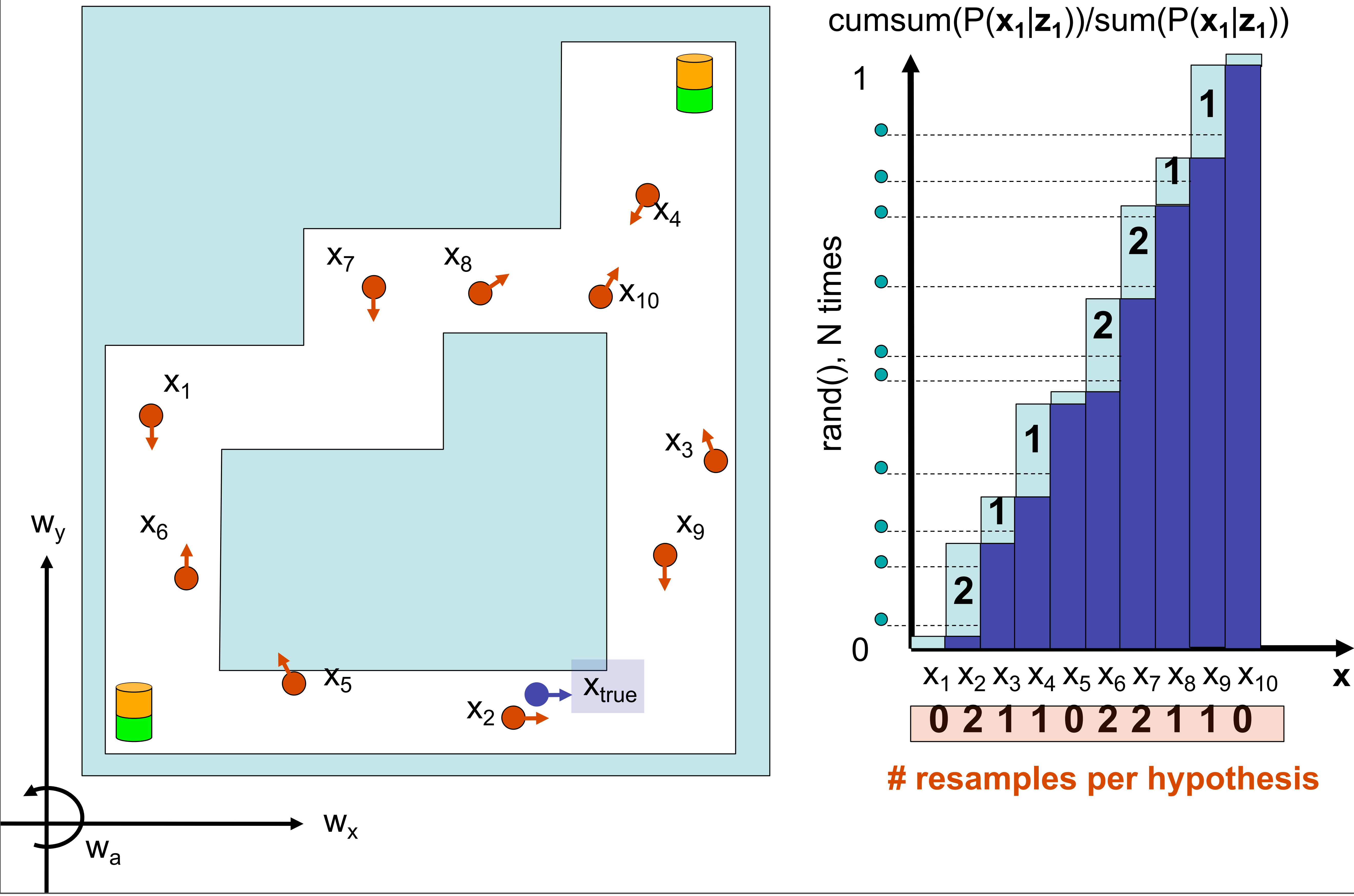
$P(\mathbf{x_1}|\mathbf{z_1}) \leftarrow P(\mathbf{z_1}|\mathbf{x_1}) \ / \ sum_x(P(\mathbf{z_1}|\mathbf{x_1}))$



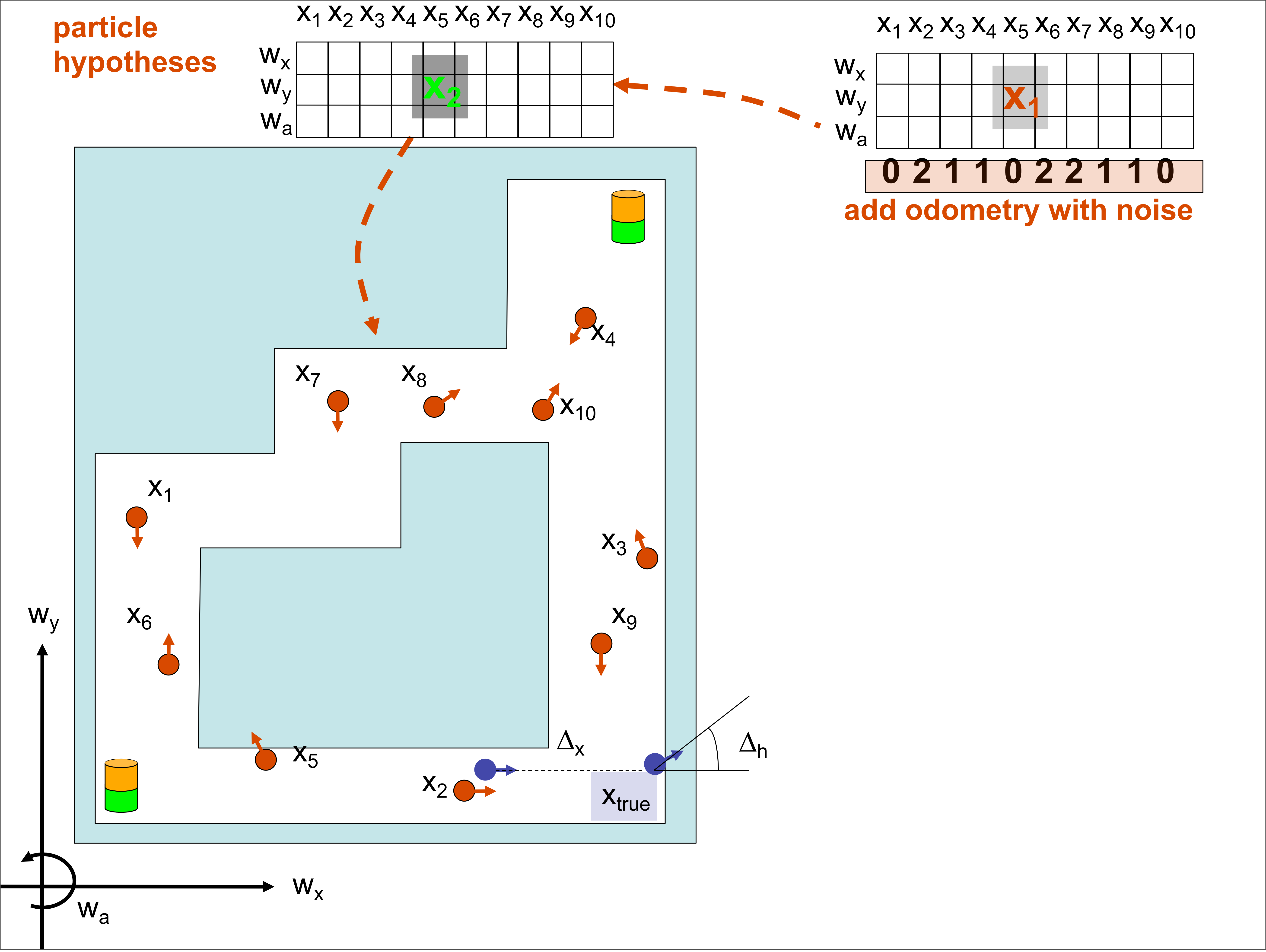
running sum, normalize range

$cumsum(P(\mathbf{x_1}|\mathbf{z_1}))/sum(P(\mathbf{x_1}|\mathbf{z_1}))$

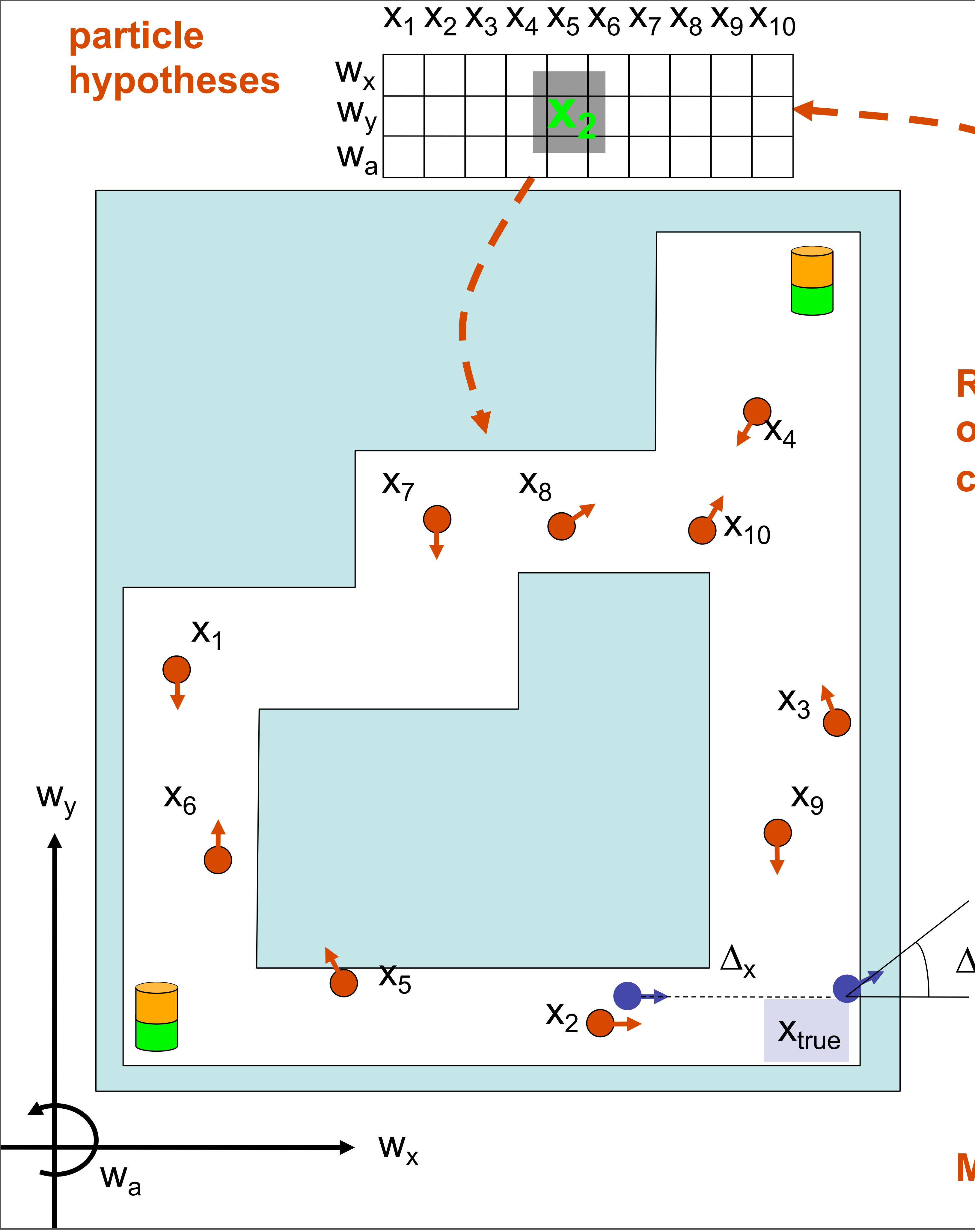


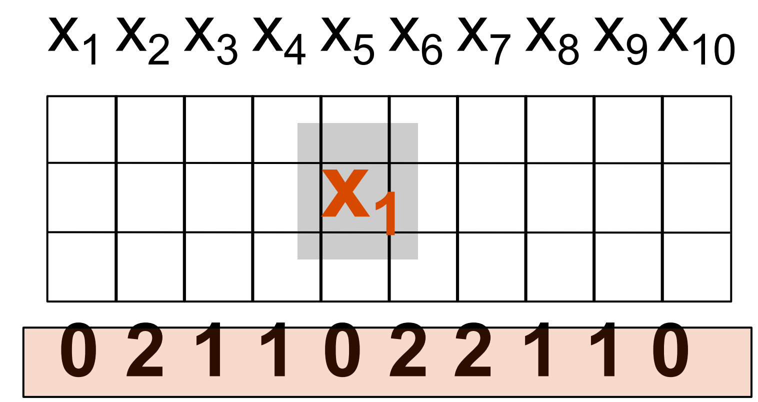


cumsum(P(x1|z1))/sum(P(x1|z1))
1
0
rand(), N times
2
1
1
2
2
1
1
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
x
0 2 1 1 0 2 2 1 1 0
resamples per hypothesis
x1
x2
x3
x4
x5
x6
x7
x8
x9
x10
xtrue
wy
wx
wa


particle hypotheses
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
wx
wy
wa
x2
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
wx
wy
wa
x1
0 2 1 1 0 2 2 1 1 0
add odometry with noise
x4
x7
x8
x10
x1
x3
x6
x9
wy
x5
Δx
Δh
x2
xtrue
wx
wa

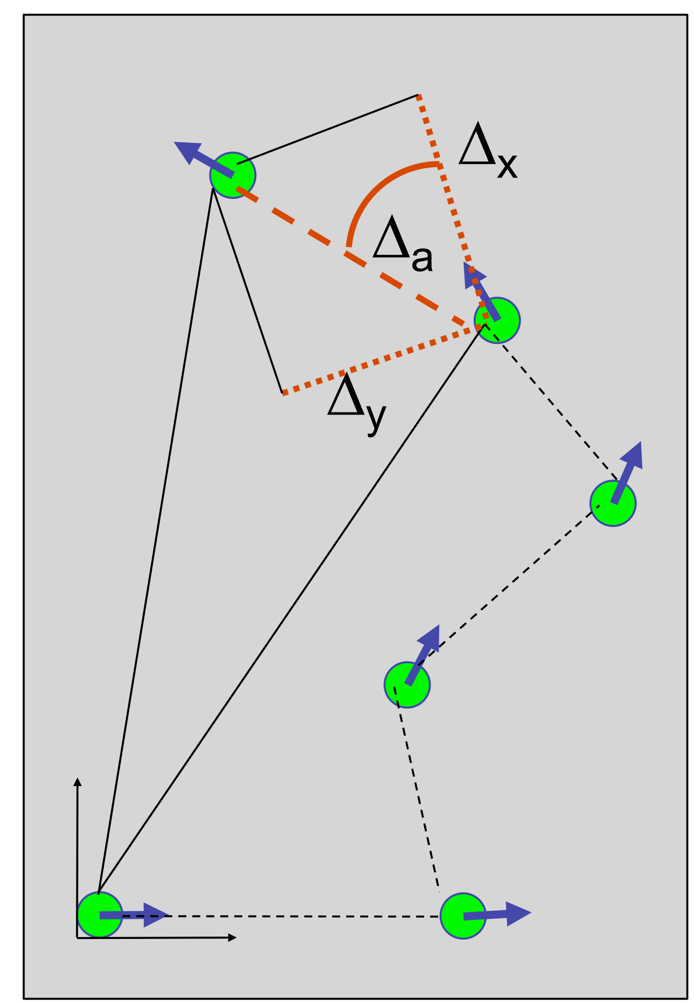
particle hypotheses



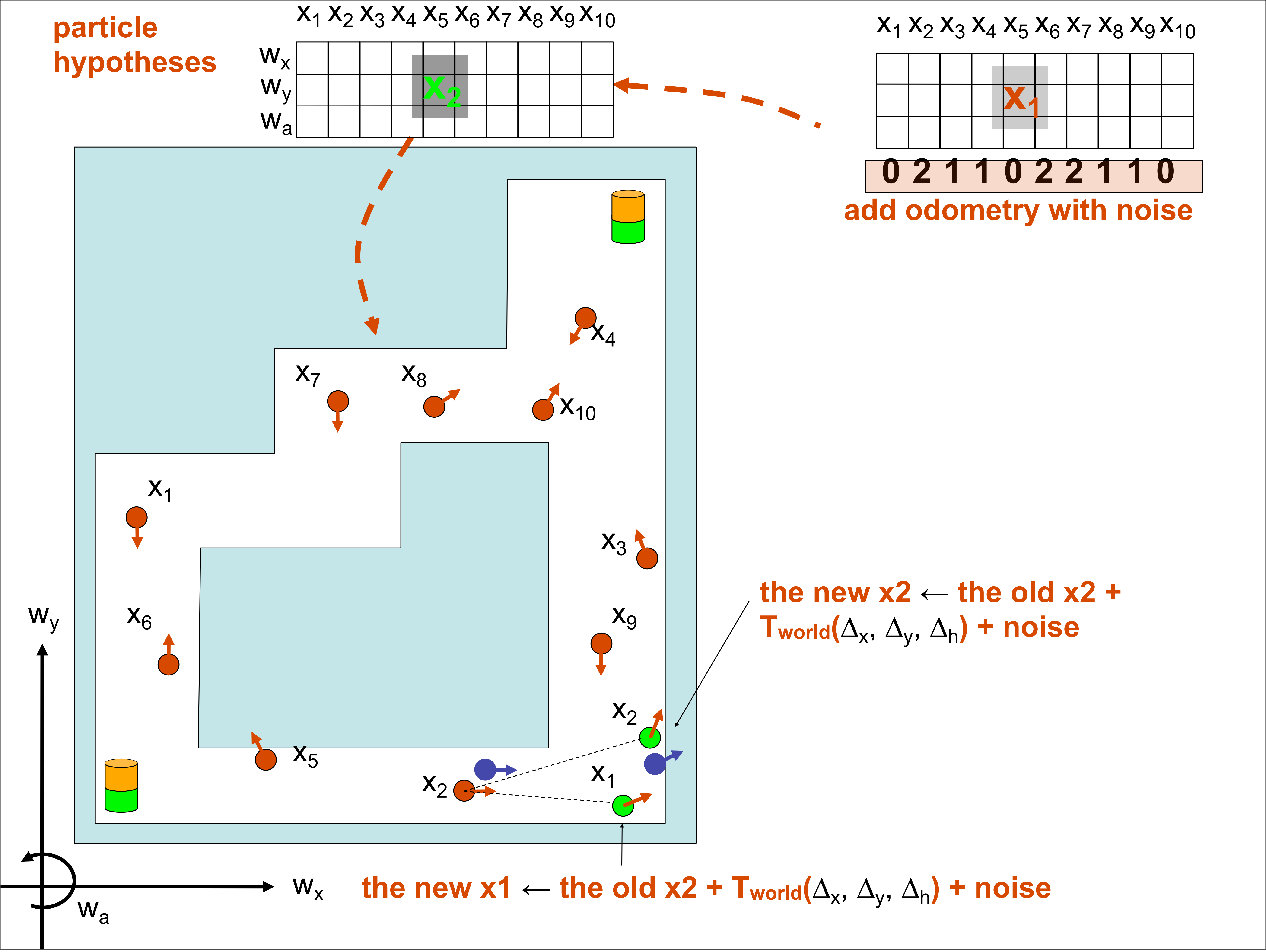


add odometry with noise

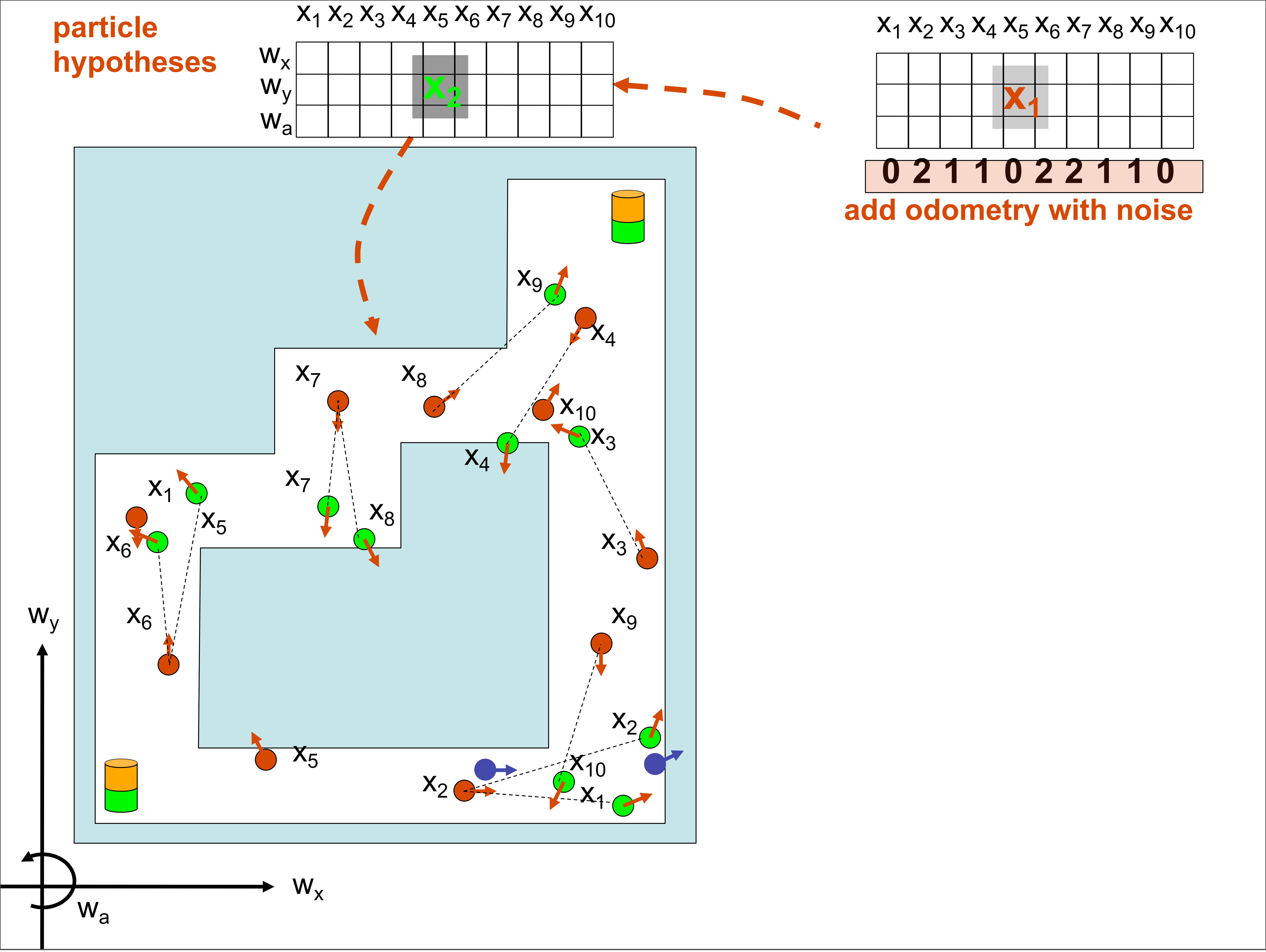
Remember: rotate and translate odometry (Δ_x Δ_y Δ_a) in world coordinates, then add to pose

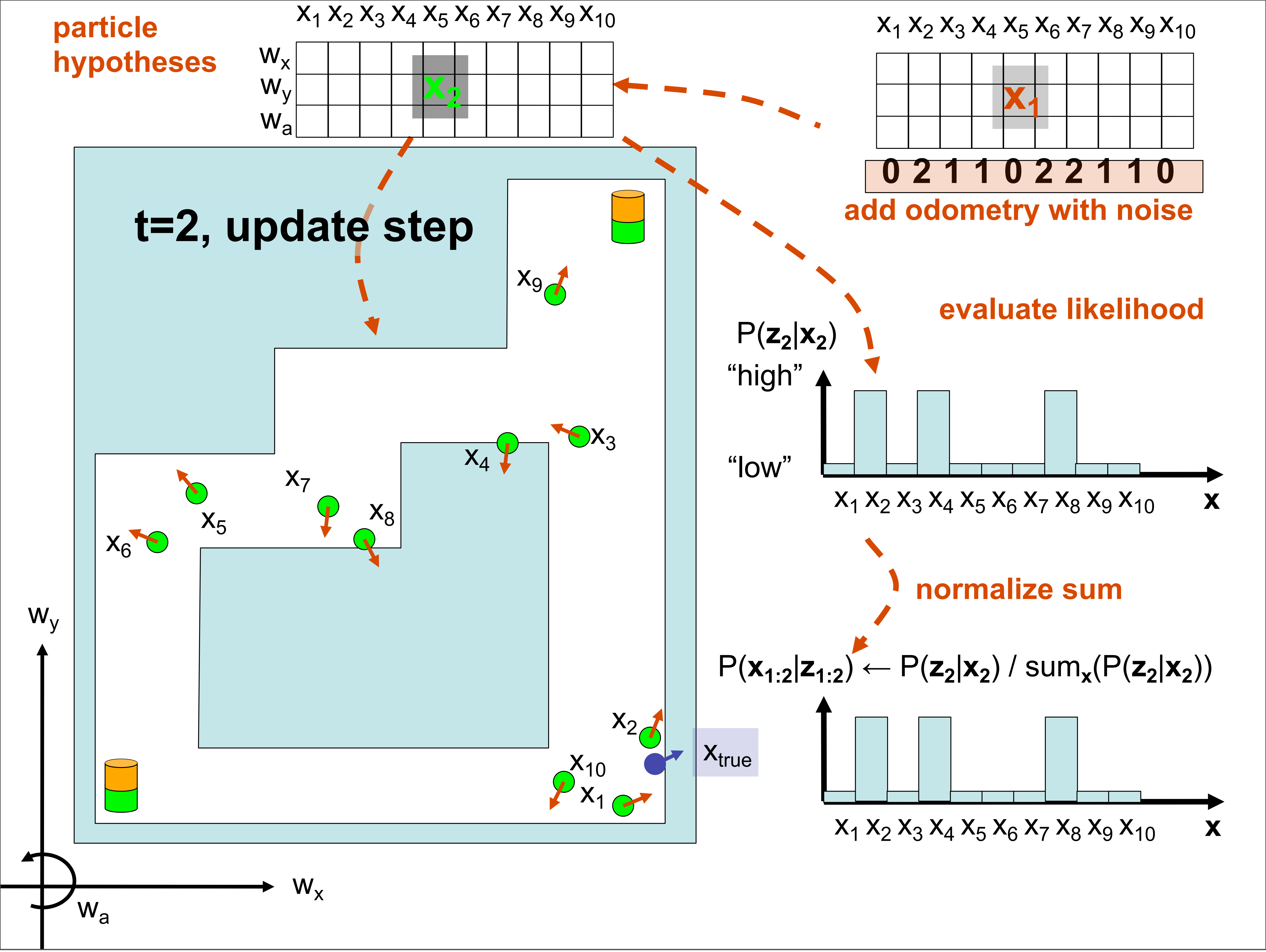


More on odometry next lecture



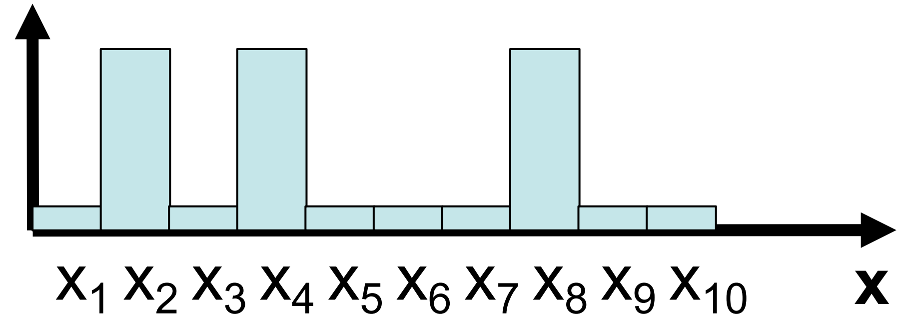
particle hypotheses
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
wx
wy
wa
x2
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
x1
0 2 1 1 0 2 2 1 1 0
add odometry with noise
x4
x7
x8
x10
x1
x3
x6
x9
x2
x5
x2
x1
wy
wx
wa
the new x2 ← the old x2 + Tworld(Δx, Δy, Δh) + noise
the new x1 ← the old x2 + Tworld(Δx, Δy, Δh) + noise

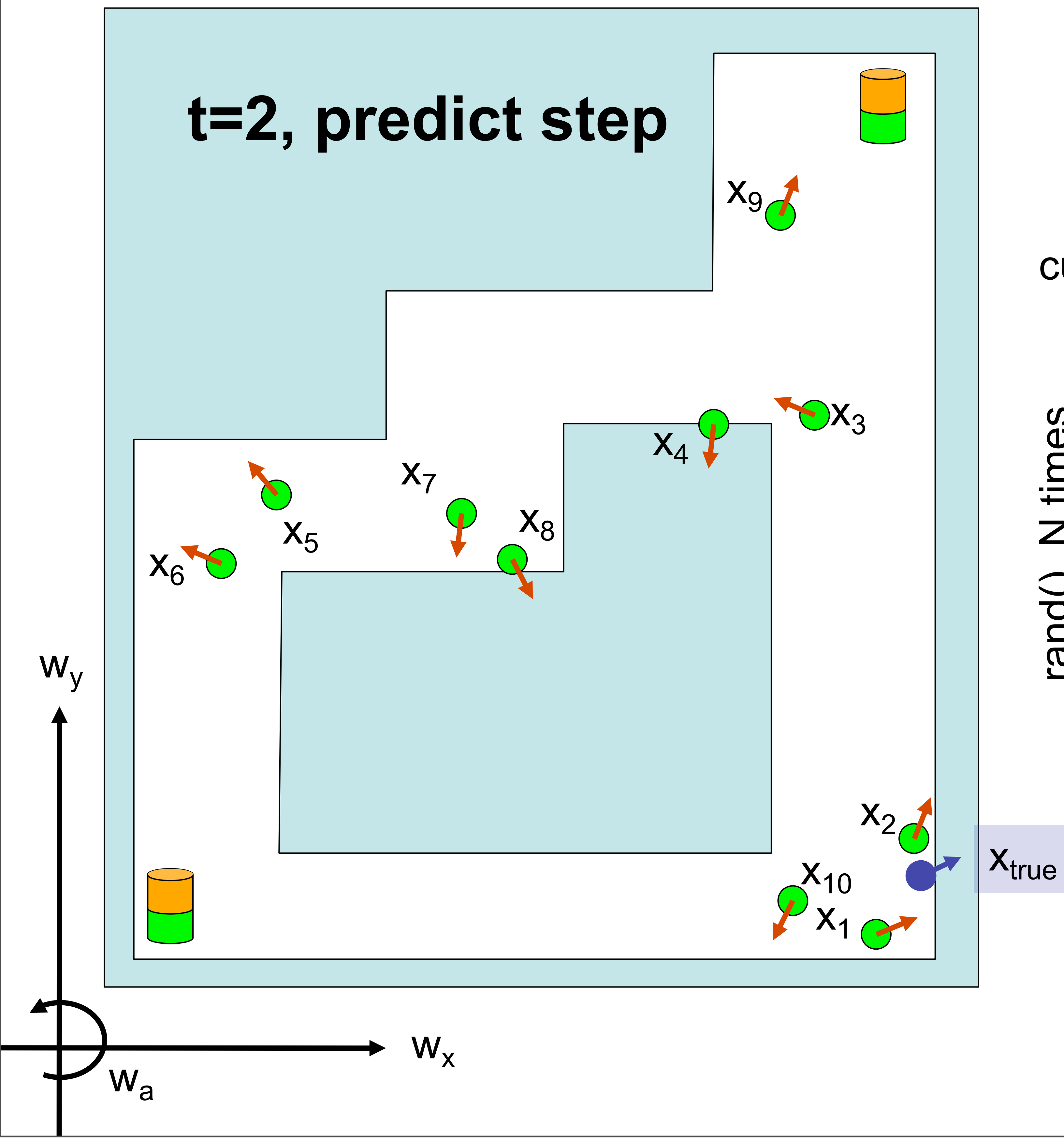

particle hypotheses
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
wx
wy
wa
x2
x1
0 2 1 1 0 2 2 1 1 0
add odometry with noise
wy
wx
wa



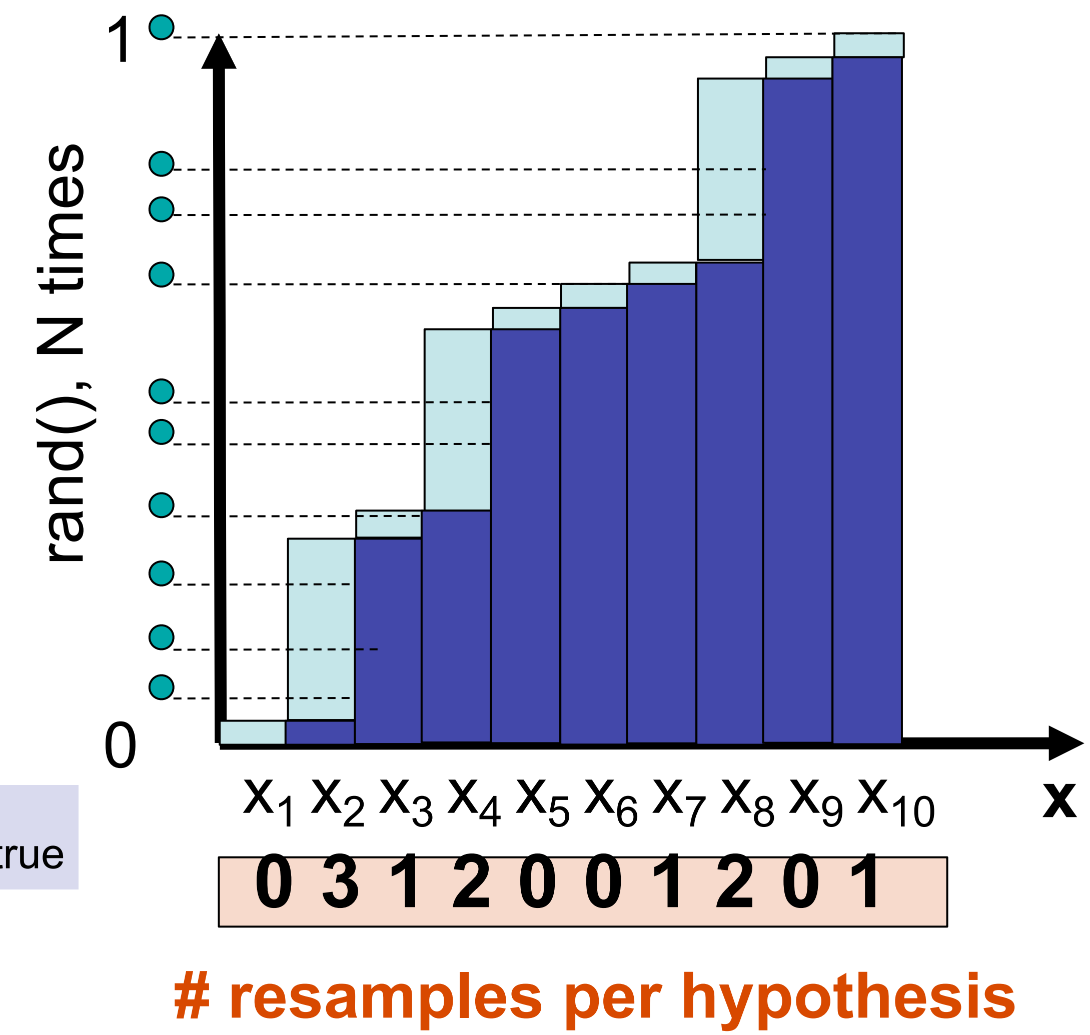
particle hypotheses
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
wx
wy
wa
x2
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
x1
0 2 1 1 0 2 2 1 1 0
add odometry with noise
t=2, update step
x9
x3
x4
x7
x5
x8
x6
evaluate likelihood
P(z2|x2)
"high"
"low"
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
x
normalize sum
P(x1:2|z1:2) ← P(z2|x2) / sumx(P(z2|x2))
wy
x2
xtrue
x10
x1
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
x
wx
wa

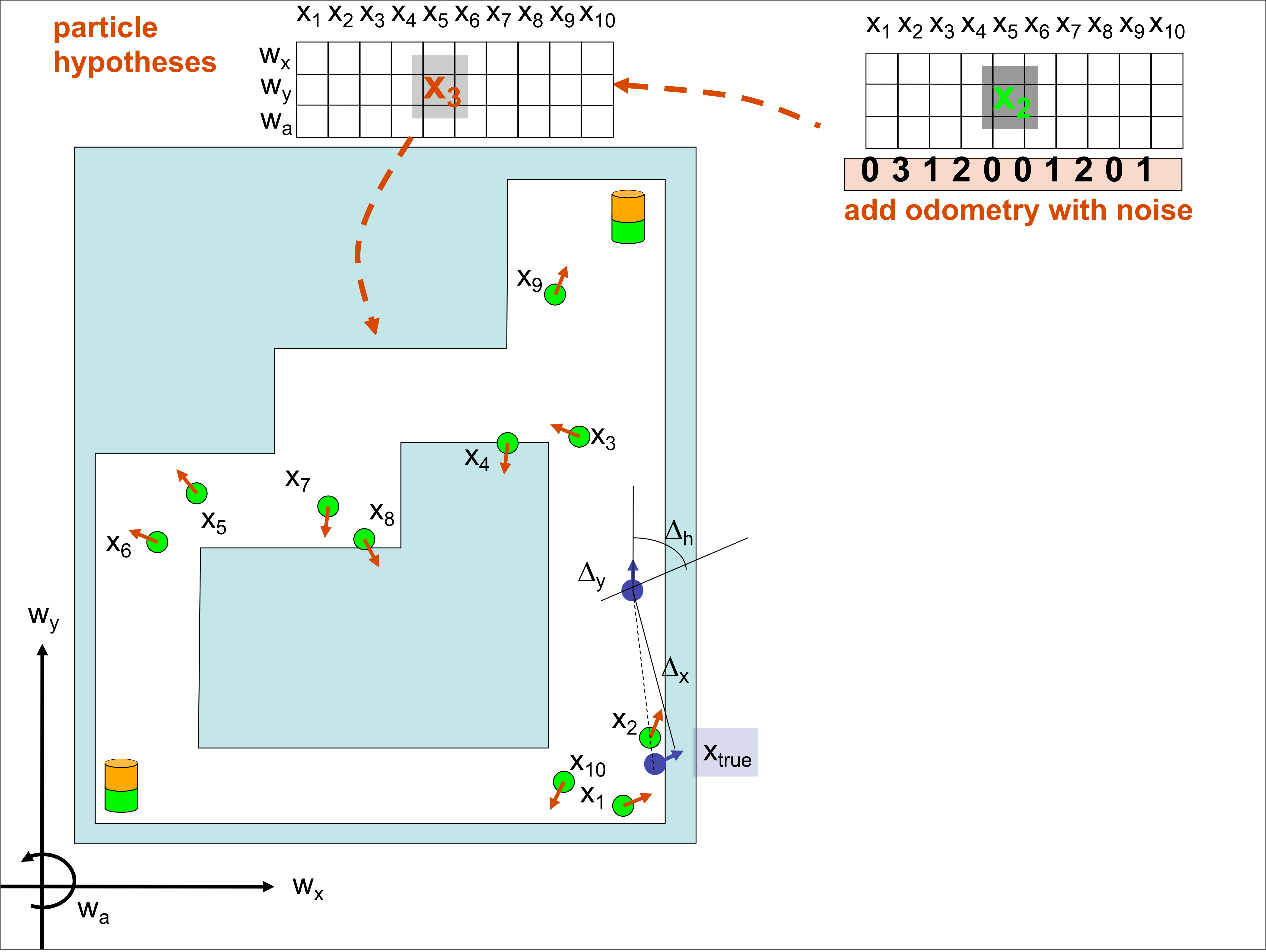
$P(\mathbf{x}_{1:2}|\mathbf{z}_{1:2}) \leftarrow P(\mathbf{z}_2|\mathbf{x}_2) \;/\; \mathrm{sum}_{\mathbf{x}}(P(\mathbf{z}_2|\mathbf{x}_2))$

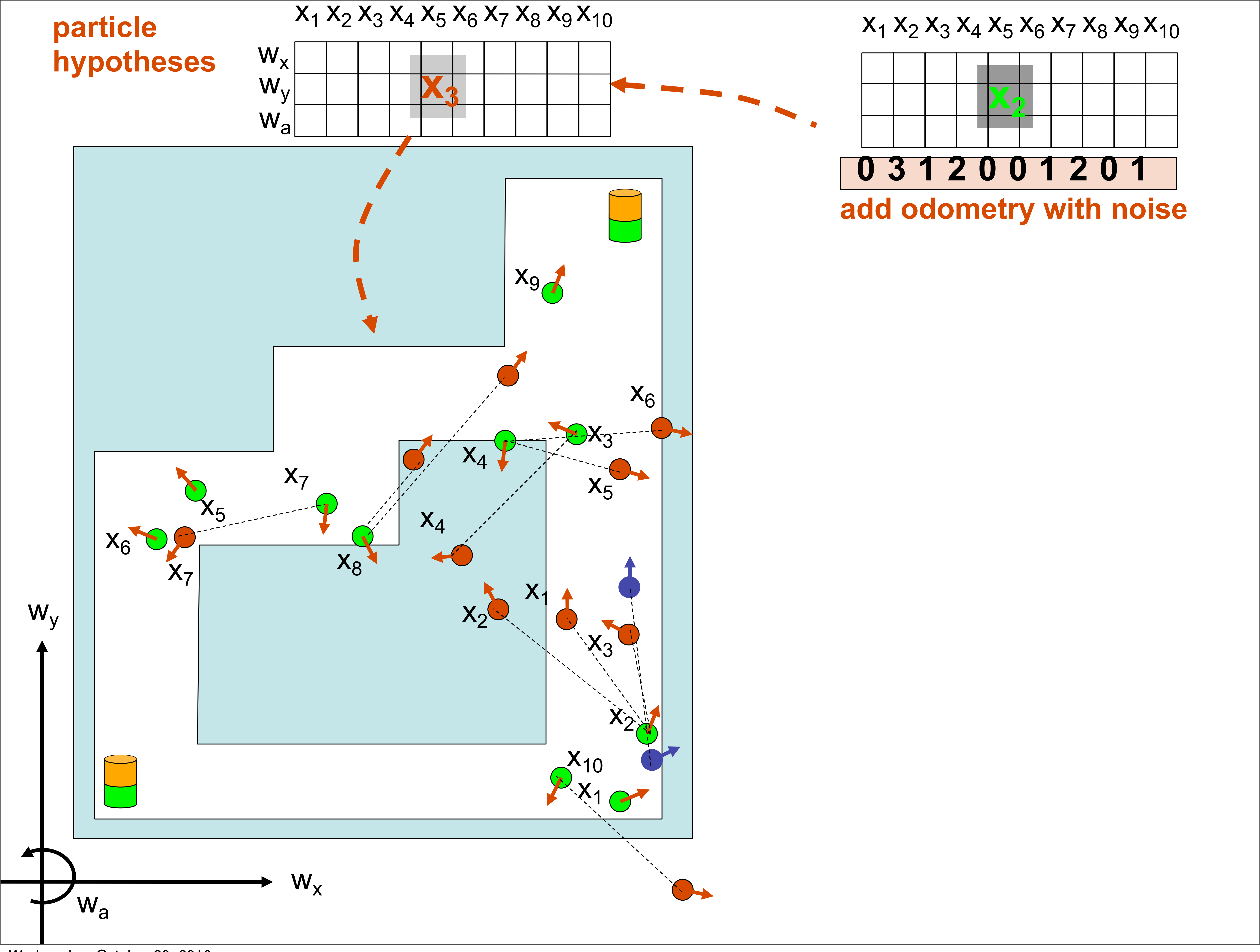


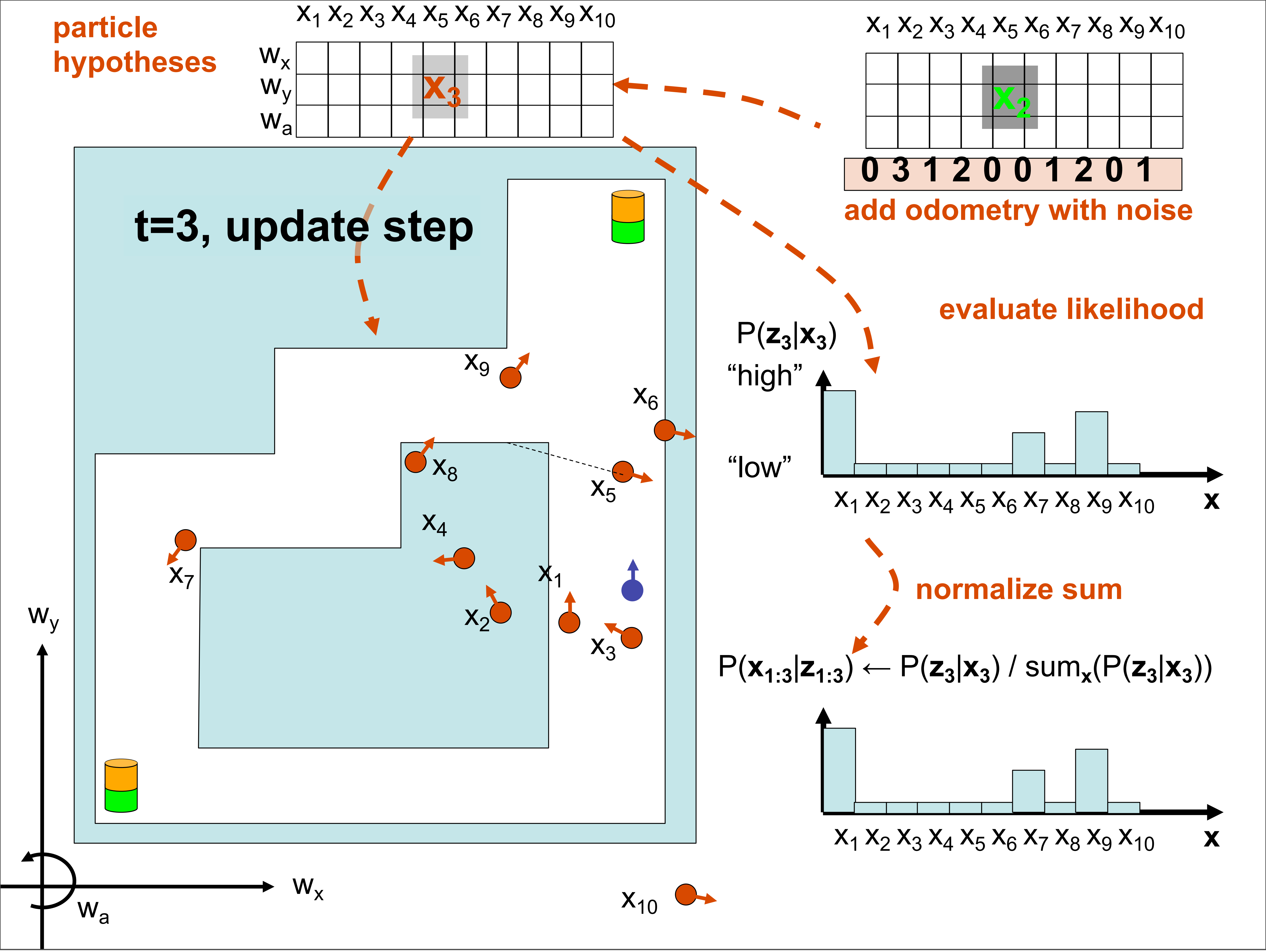


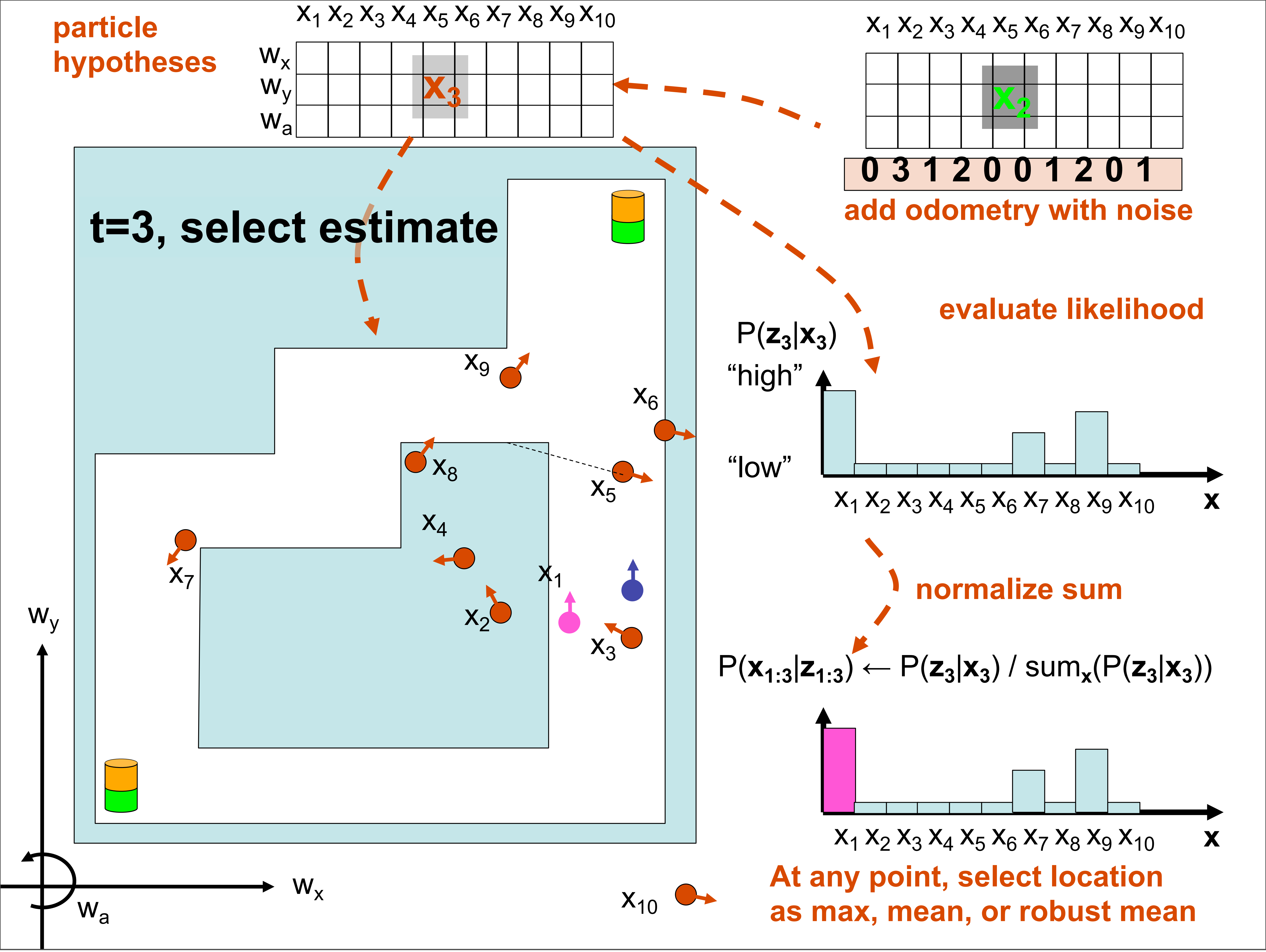
$\mathrm{cumsum}(P(\mathbf{x}_{1:2}|\mathbf{z}_{1:2}))/\mathrm{sum}(P(\mathbf{x}_{1:2}|\mathbf{z}_{1:2}))$




particle hypotheses
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
wx
wy
wa
x3
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
x2
0 3 1 2 0 0 1 2 0 1
add odometry with noise
x9
x3
x4
x7
x5
x8
x6
Δh
Δy
Δx
x2
xtrue
x10
x1
wy
wx
wa


particle hypotheses
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
wx
wy
wa
x3
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
x2
0 3 1 2 0 0 1 2 0 1
add odometry with noise
x9
x6
x3
x4
x5
x7
x5
x6
x7
x8
x4
x1
x2
x3
x2
x10
x1
wy
wx
wa


particle hypotheses
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
wx
wy
wa
x3
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
x2
0 3 1 2 0 0 1 2 0 1
add odometry with noise
t=3, update step
evaluate likelihood
P(z3|x3)
"high"
"low"
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
x
x9
x6
x8
x5
x4
x7
x1
x2
x3
x10
normalize sum
P(x1:3|z1:3) ← P(z3|x3) / sumx(P(z3|x3))
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
x
wy
wx
wa


particle hypotheses
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
wx
wy
wa
x3
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
x2
0 3 1 2 0 0 1 2 0 1
add odometry with noise
t=3, select estimate
evaluate likelihood
P(z3|x3)
"high"
"low"
x9
x6
x8
x5
x4
x7
x1
x2
x3
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
x
normalize sum
P(x1:3|z1:3) ← P(z3|x3) / sumx(P(z3|x3))
wy
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10
x
wx
wa
x10
At any point, select location as max, mean, or robust mean

Open Issues

- How many particles?
- How to evaluate likelihood?
- How much odometry noise?
- What if my estimate is wrong?