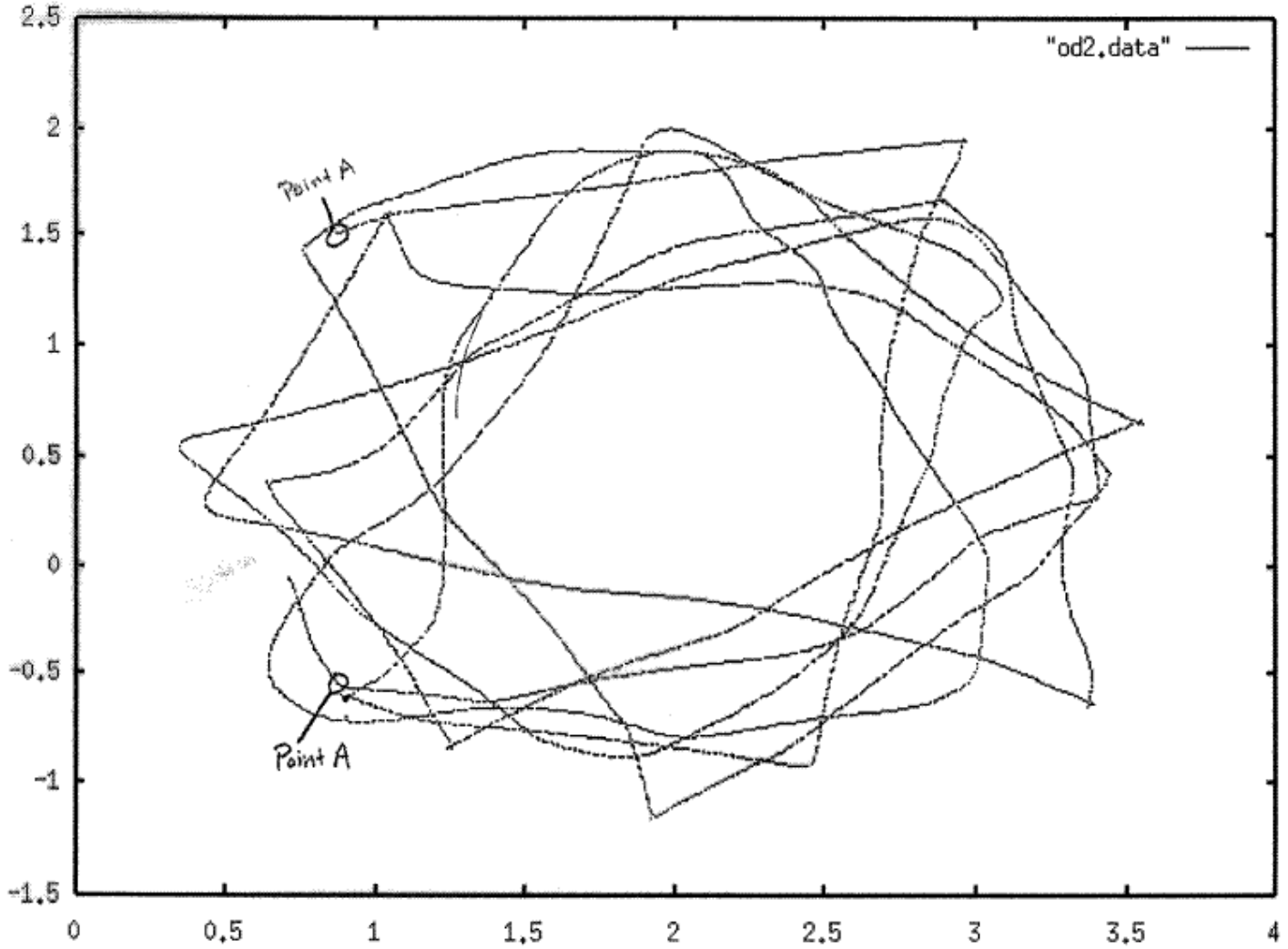
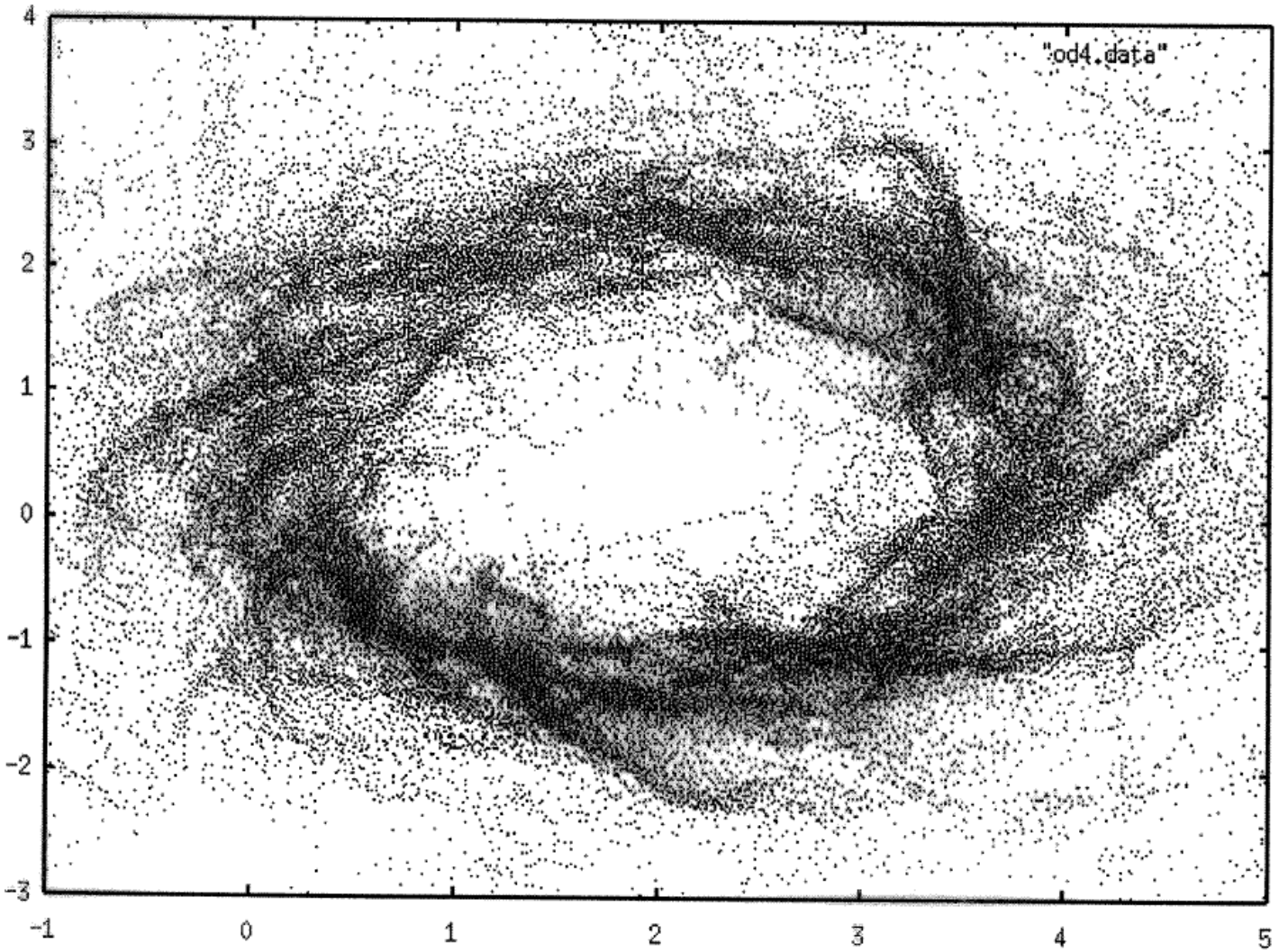
Localization: *"Where am I?"*

- The map-building method we studied assumes that the robot knows its location.
  - Precise (x,y,θ) coordinates in the same frame of reference as the occupancy grid map.
  - Pose = position + orientation.
- This assumes that odometry is accurate, which is often false.
- We will need to relocalize at each step.
  - Localization: determine sufficiently accurate pose.

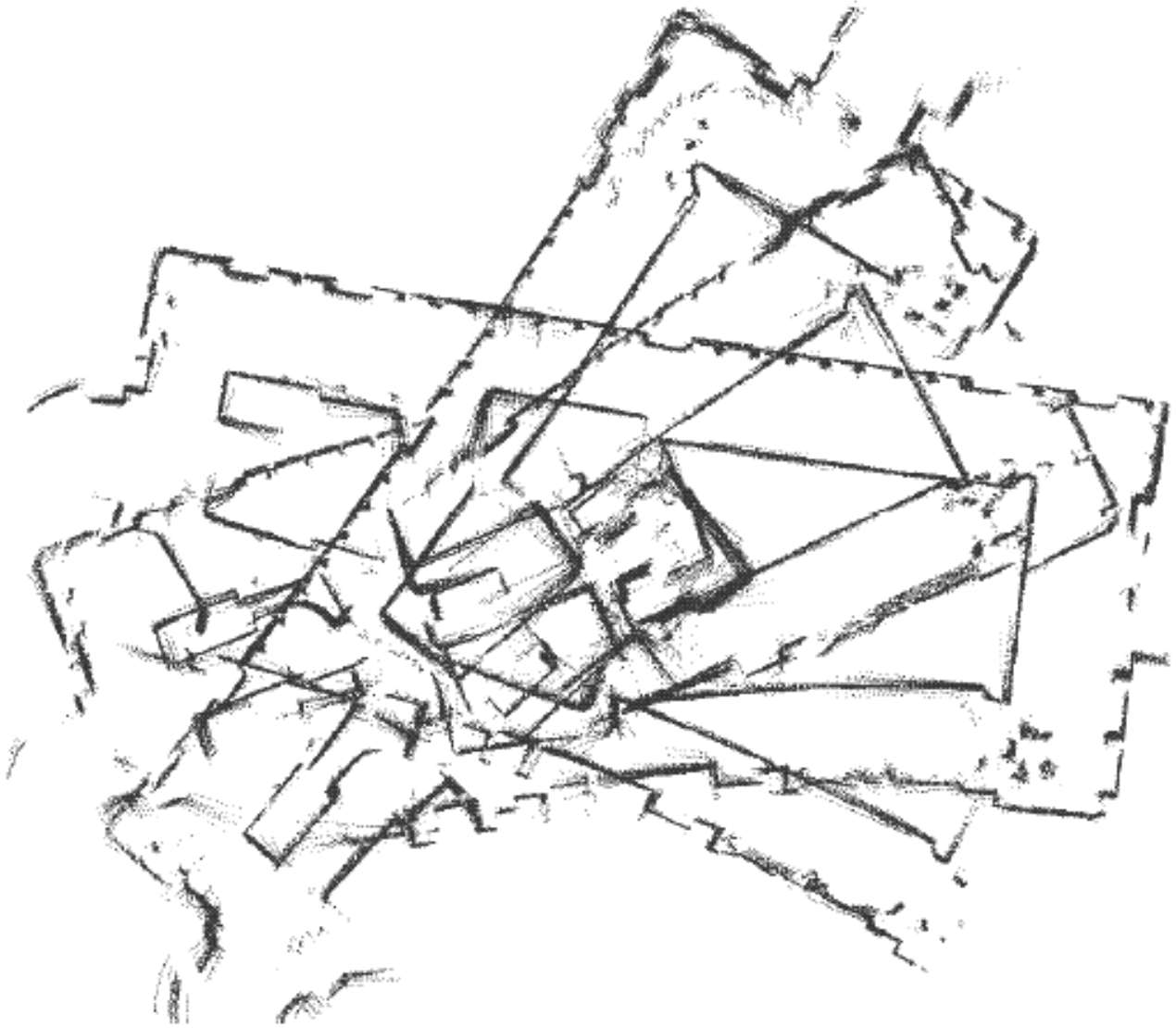
Odometry-Only Tracking:
6 times around a 2m x 3m area



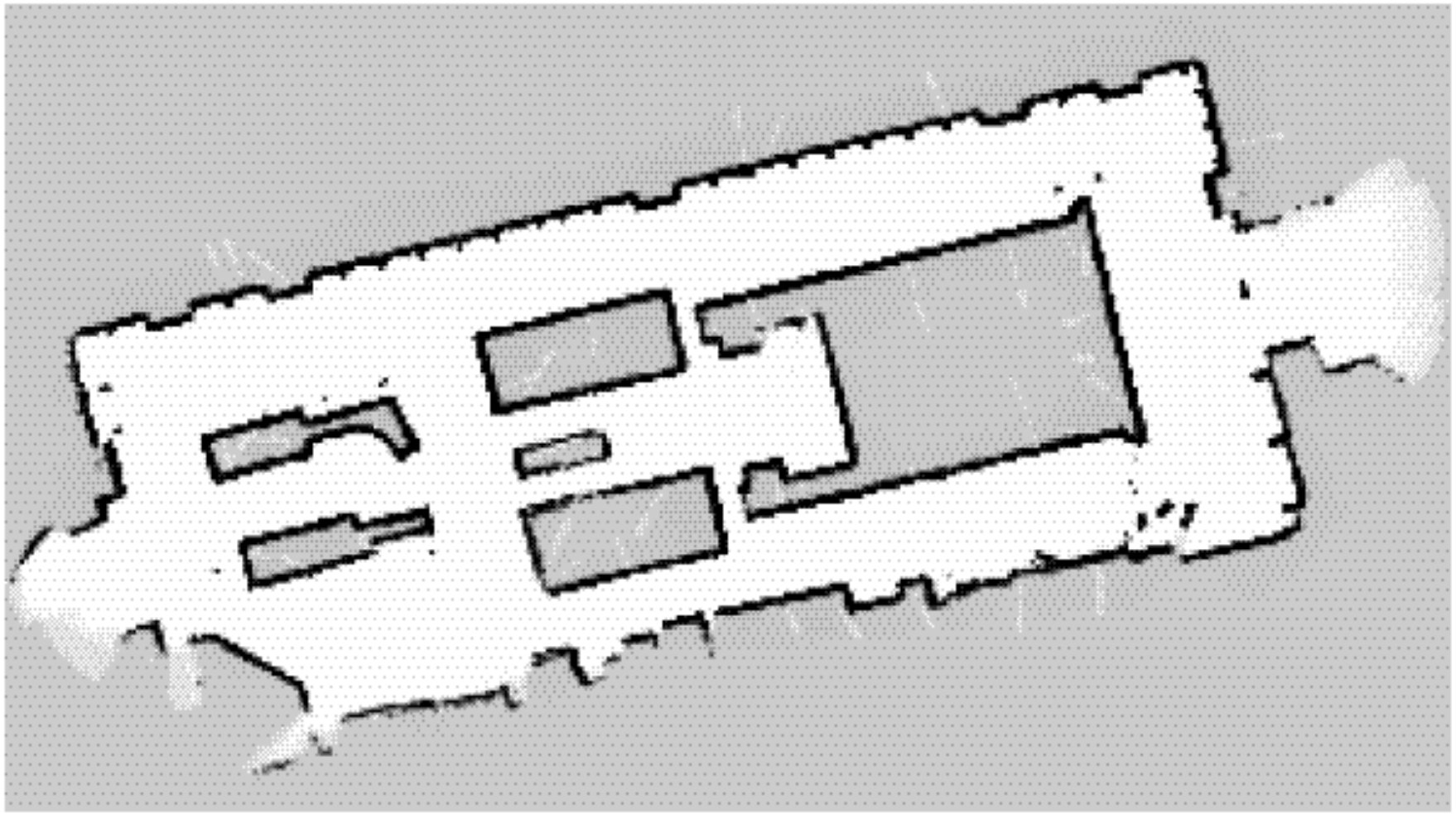
Merging Laser Range Data Based on Odometry-Only Tracking



Mapping Without Localization



Mapping With Localization

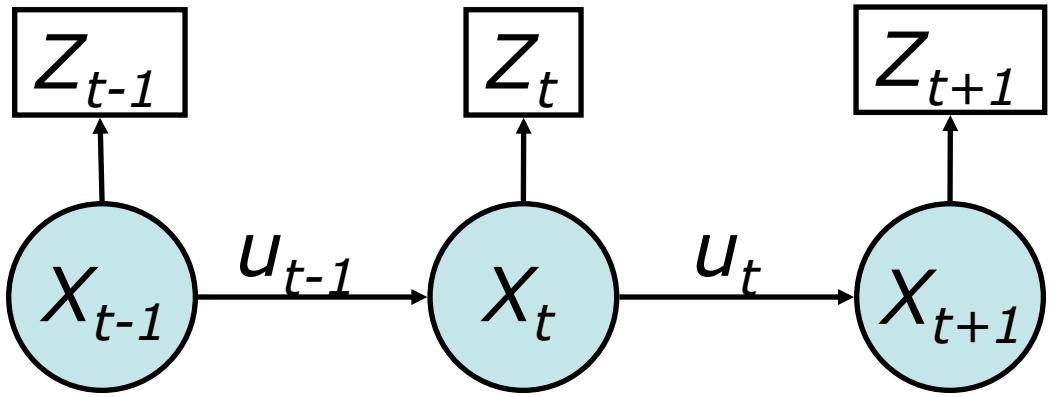


SLAM: Simultaneous Localization and Mapping

Alternate at each motion step:

1. **Localization**:
   - *Assume accurate map*.
   - Match sensor readings against the map to update location after motion.
2. **Mapping**:
   - *Assume known location in the map*.
   - Update map from sensor readings.

Modeling Action and Sensing

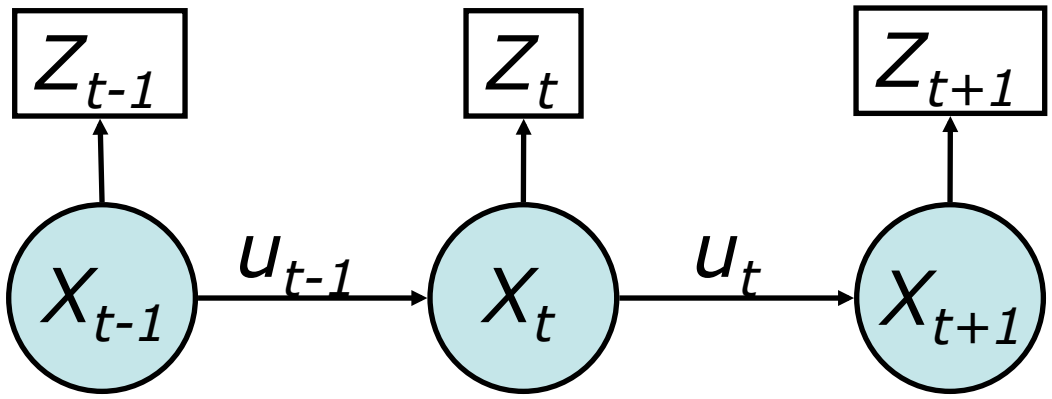


- Action model: $P(x_t \mid x_{t-1}, u_{t-1})$
- Sensor model: $P(z_t \mid x_t)$
- What we want to know is *Belief*:

$$Bel(x_t) = P(x_t \mid u_1, z_2 \ldots, u_{t-1}, z_t)$$

the posterior probability distribution of x_t, given the past history of actions and sensor inputs.

The Markov Assumption



- *Given the present, the future is independent of the past.*
- Given the state x_t, the observation z_t is independent of the past.

$$P(z_t \mid x_t, u_1, z_2 \ldots, u_{t-1}) = P(z_t \mid x_t)$$

Markov Localization

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

- $Bel(x_{t-1})$ and $Bel(x_t)$ are prior and posterior probabilities of location x.
- $P(x_t \mid u_{t-1}, x_{t-1})$ is the action model, giving the probability distribution over result of u_{t-1} at x_{t-1}.
  - Iterate over values of x_{t-1}. The action u_{t-1} is a constant.
  - x_t is the argument to $Bel(x_t)$.
- $P(z_t \mid x_t)$ is the sensor model, giving the probability distribution over sense images z_t at x_t.
  - z_t is the actual observation, so it's a constant.
- η is a normalization constant, ensuring that total probability mass over x_t is 1.

Markov Localization

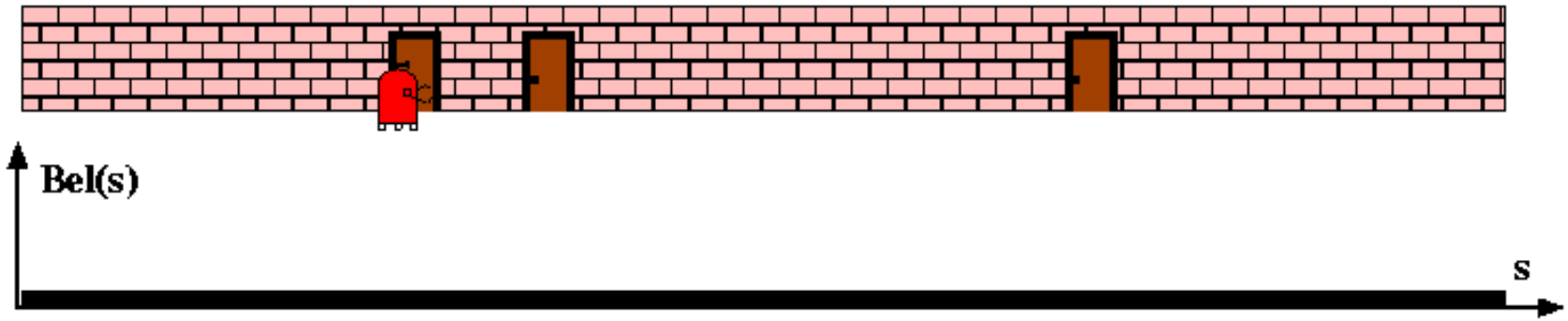
- Evaluate $Bel(x_t)$ for every possible state x_t.
- **Prediction** phase:

$$Bel^{-}(x_t) = \int P(x_t \mid u_{t-1}, x_{t-1})\, Bel(x_{t-1})\, dx_{t-1}$$

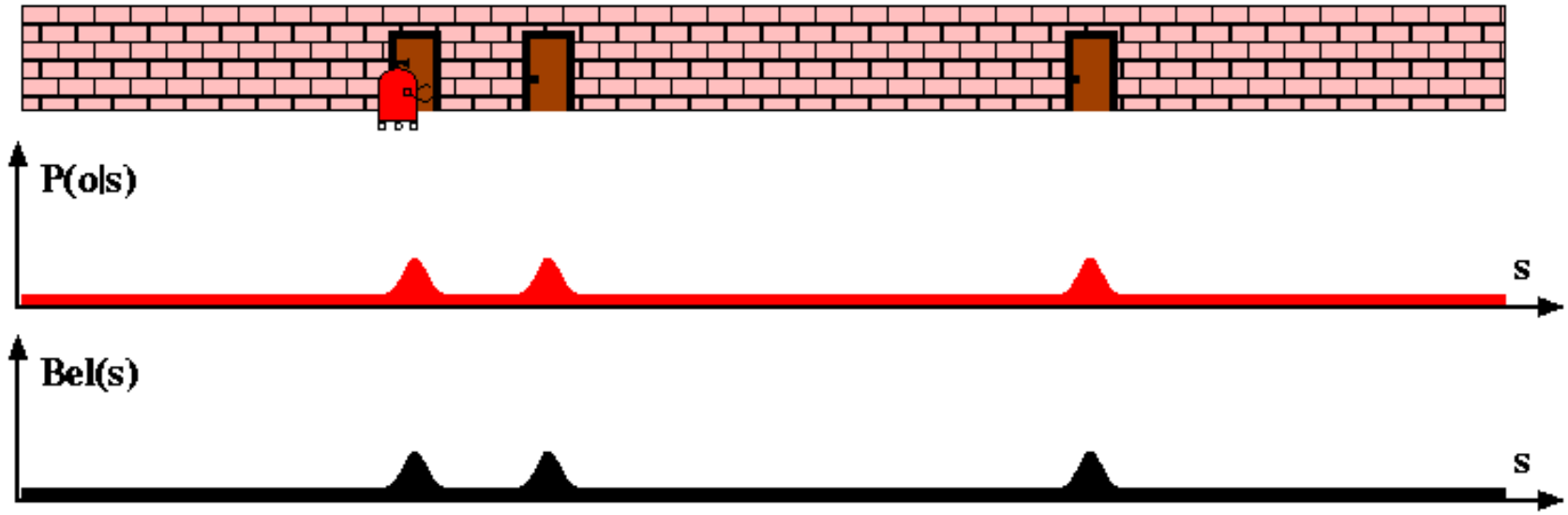
  - Integrate over every possible state x_{t-1} to apply the probability that action u_{t-1} could reach x_t from there.
- **Correction** phase:

$$Bel(x_t) = \eta\ P(z_t \mid x_t)\, Bel^{-}(x_t)$$

  - Weight each state x_t with likelihood of observation z_t.

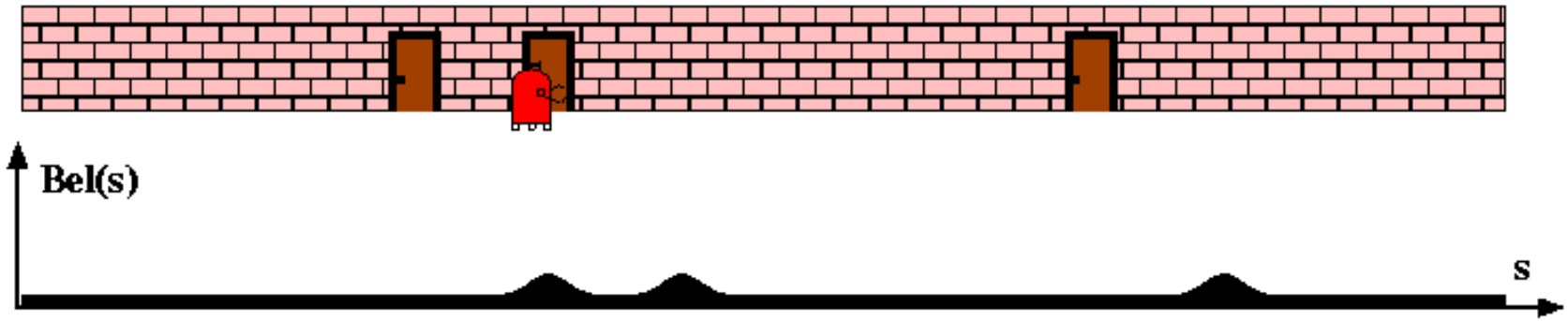


Uniform prior probability $Bel^{-}(x_0)$



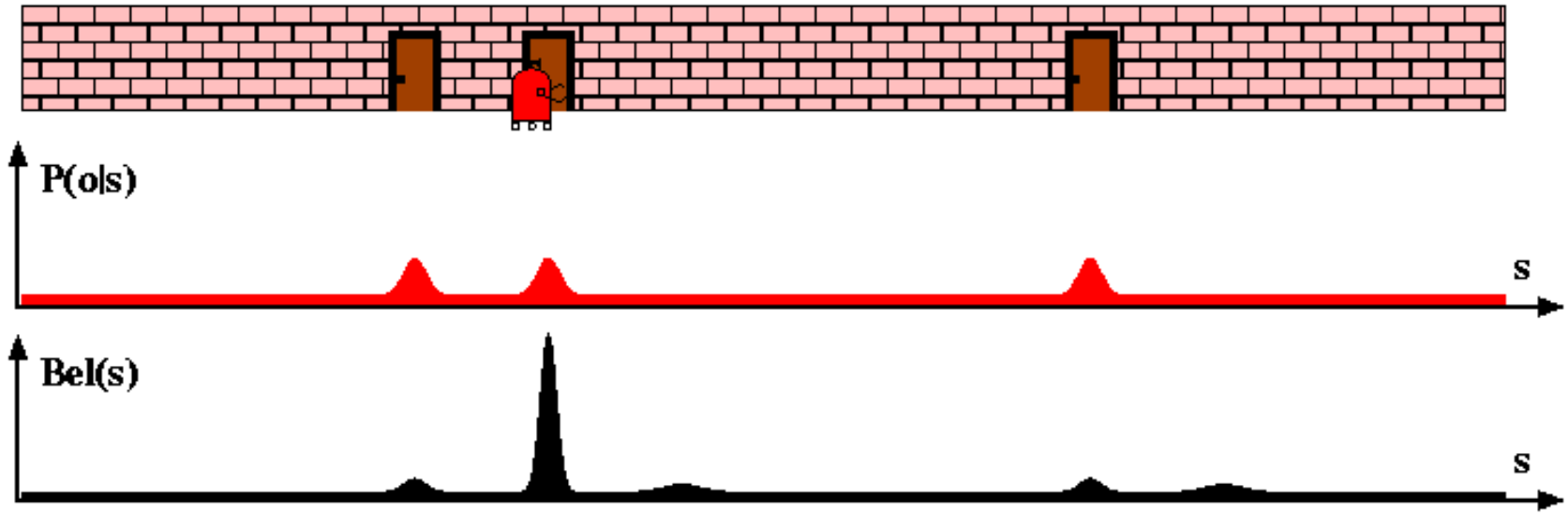
Sensor information $P(z_0 \mid x_0)$

$$Bel(x_0) = \eta \; P(z_0 \mid x_0) \; Bel^{-}(x_0)$$



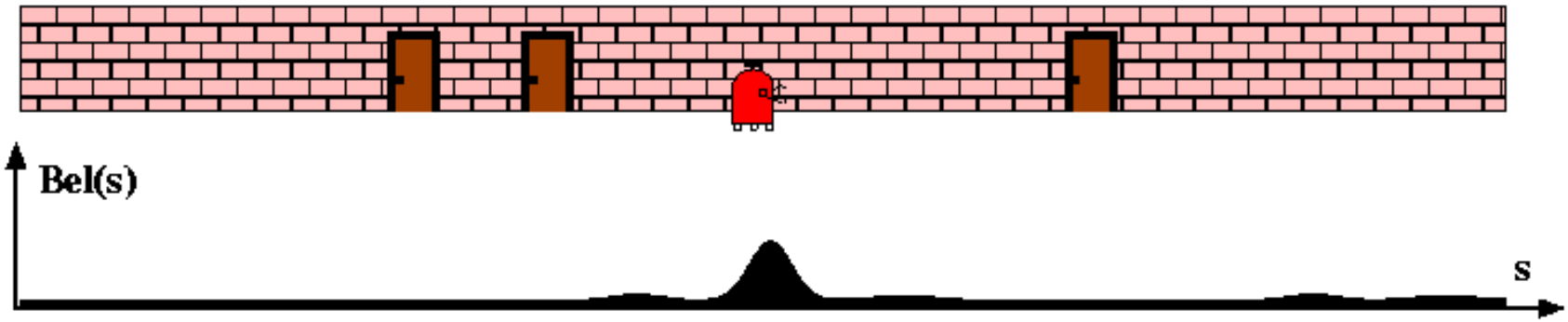
Apply the action model $P(x_1 \mid u_0, x_0)$

$$Bel^{-}(x_1) = \int P(x_1 \mid u_0, x_0)\, Bel(x_0)\, dx_0$$



Combine with observation $P(z_1 \mid x_1)$

$$Bel(x_1) = \eta \; P(z_1 \mid x_1) \; Bel^{-}(x_1)$$



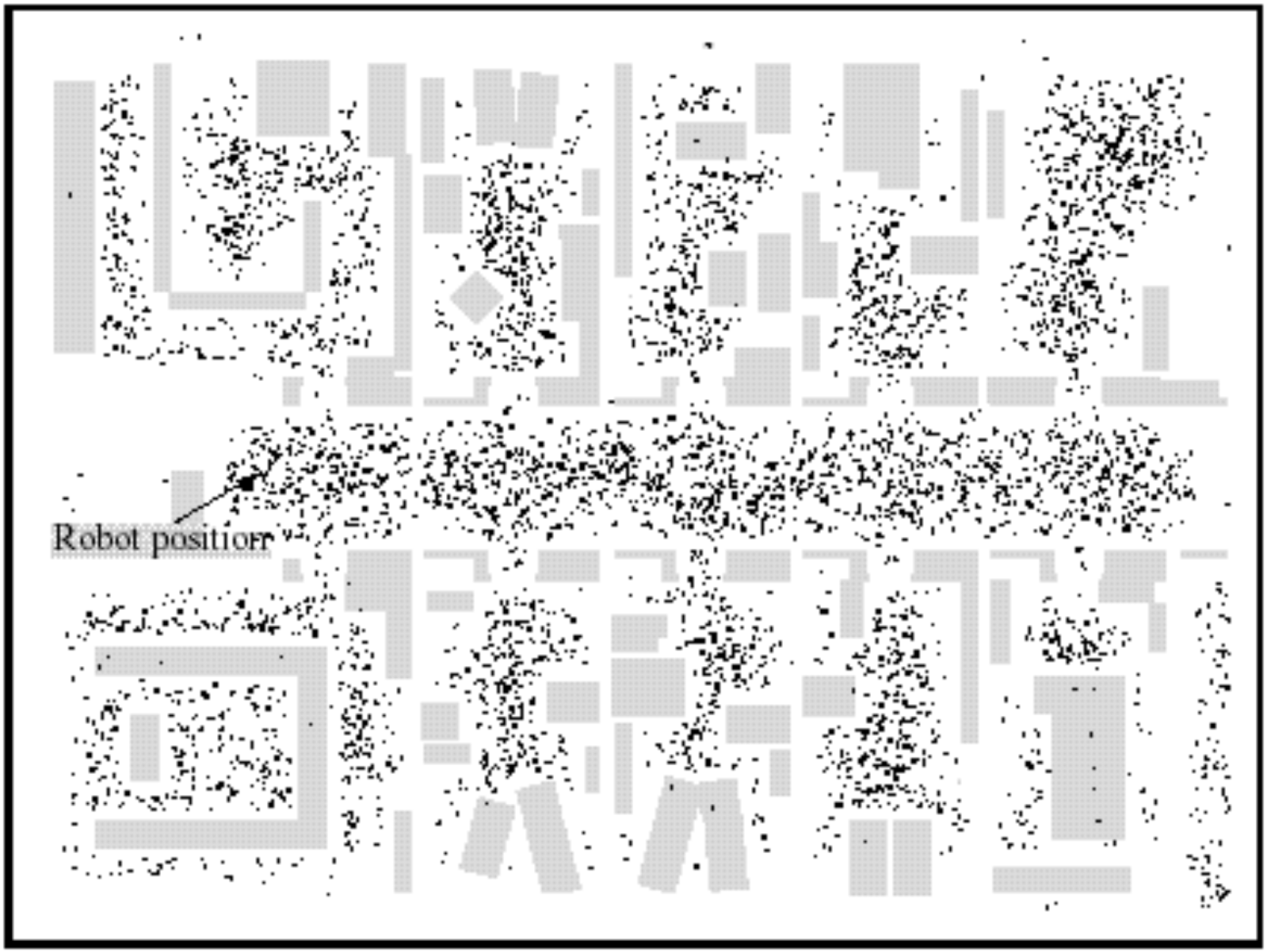
Action model again: $P(x_2 \mid u_1, x_1)$

$$Bel^{-}(x_2) = \int P(x_2 \mid u_1, x_1)\, Bel(x_1)\, dx_1$$

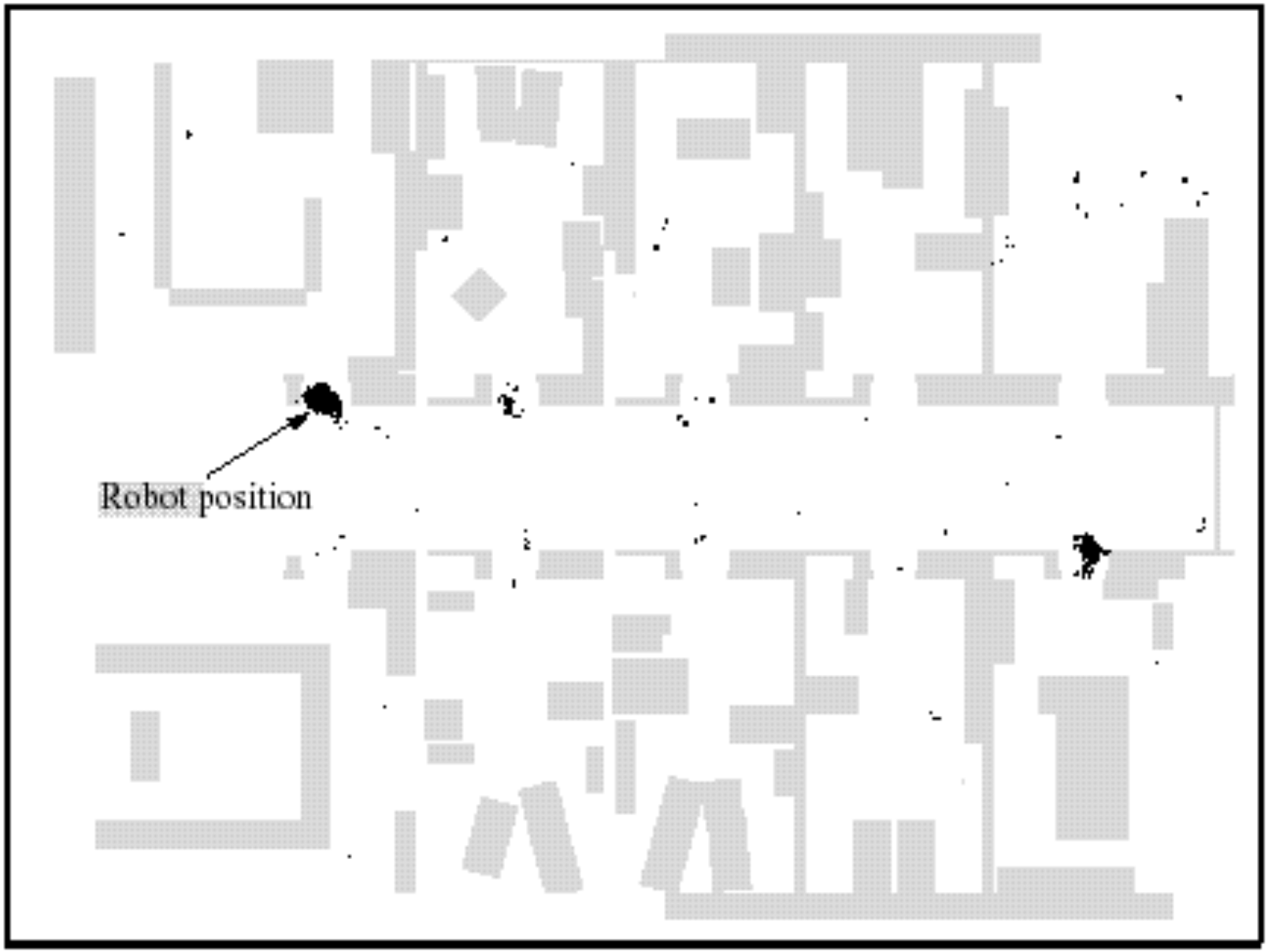
Local and Global Localization

- Most localization is *local*:
  - Incrementally correct belief in position after each action.
- *Global* localization is more dramatic.
  - Where in the entire environment am I?
- The "kidnapped robot problem"
  - Includes detecting that I am lost.

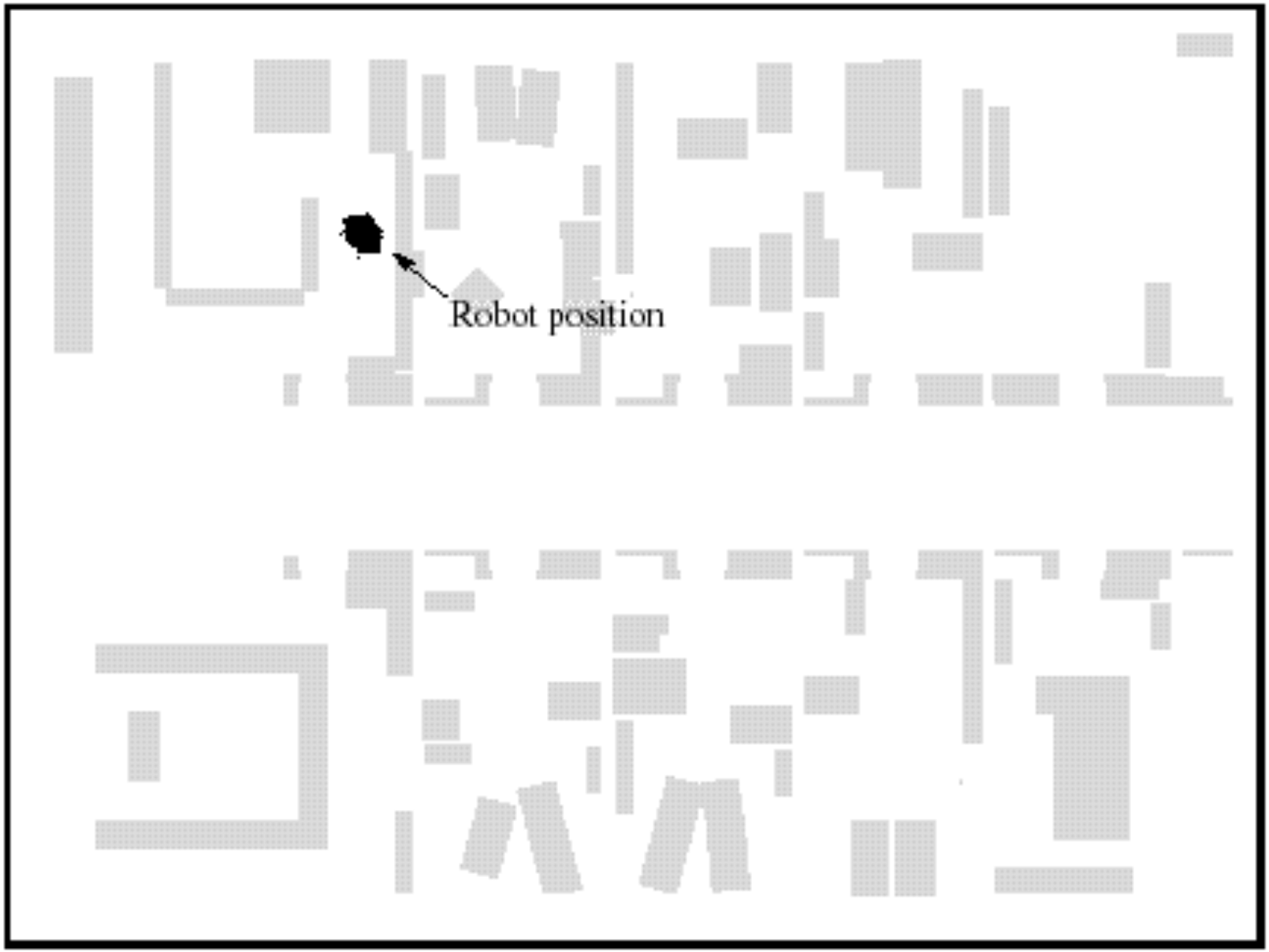
Initial belief $Bel(x_0)$



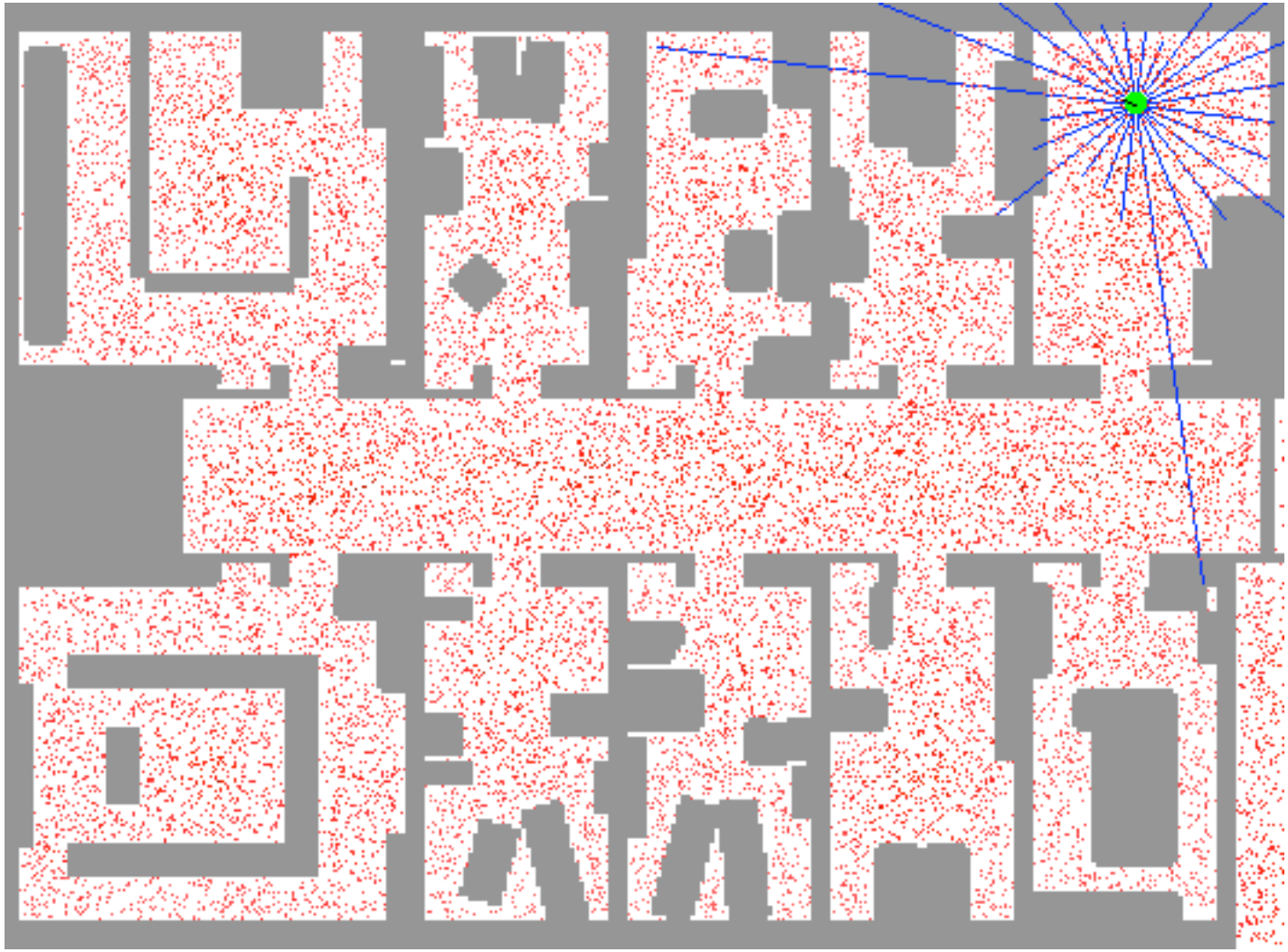
Intermediate Belief $Bel(x_t)$



Final Belief $Bel(x_t)$



Global Localization Movie



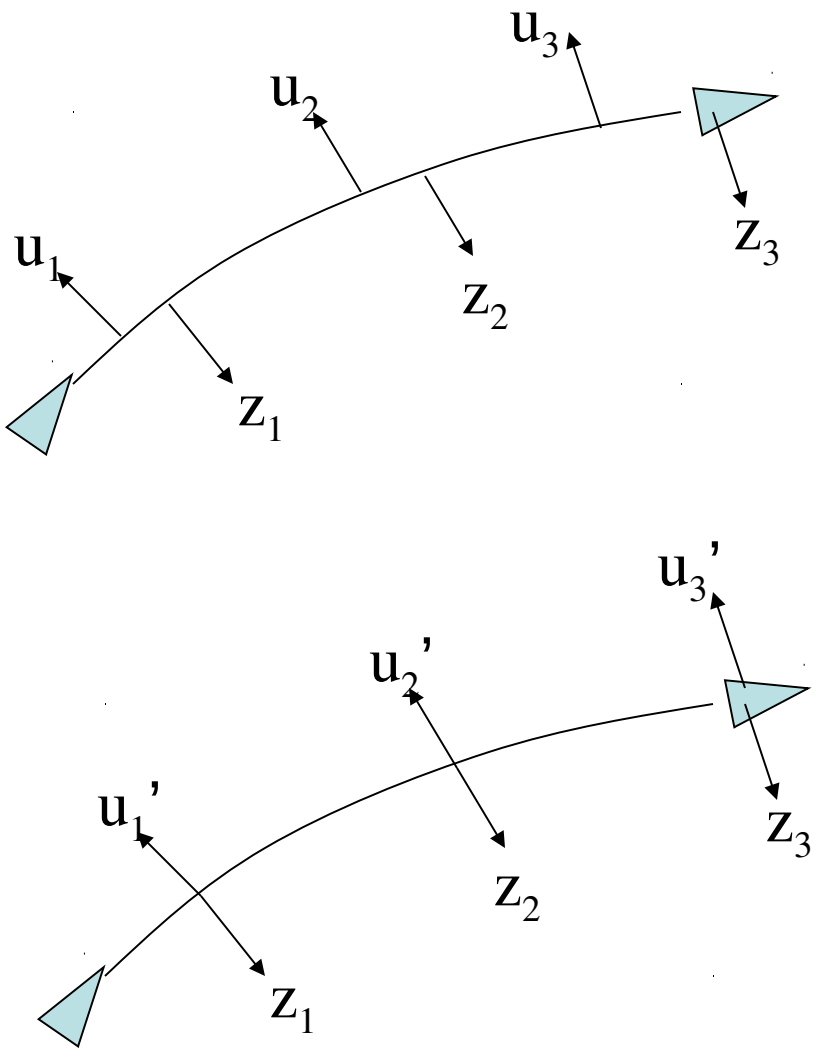
Action and Sensor Models

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

- The Markov localization equation depends on two types of knowledge about the robot.
- The **action model**: $P(x_t \mid u_{t-1}, x_{t-1})$
  - Given a state x_t and odometry u_t, the distribution over possible next states x_{t+1}
- The **sensor model**: $P(z_t \mid x_t)$
  - Given a state x_t, the distribution over possible sensor measurements z_t.

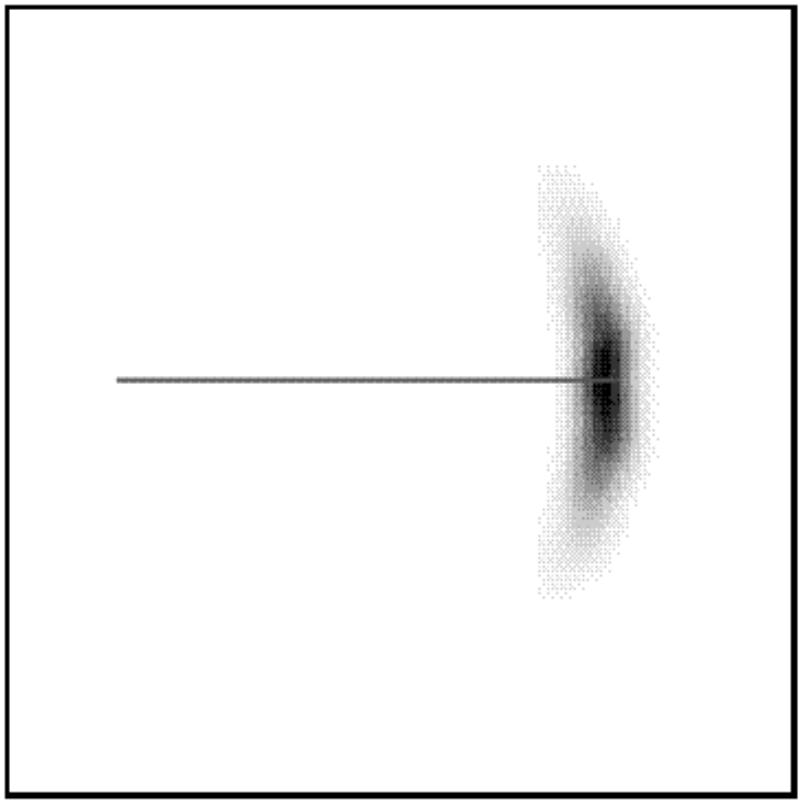
Interpolate Observation Times

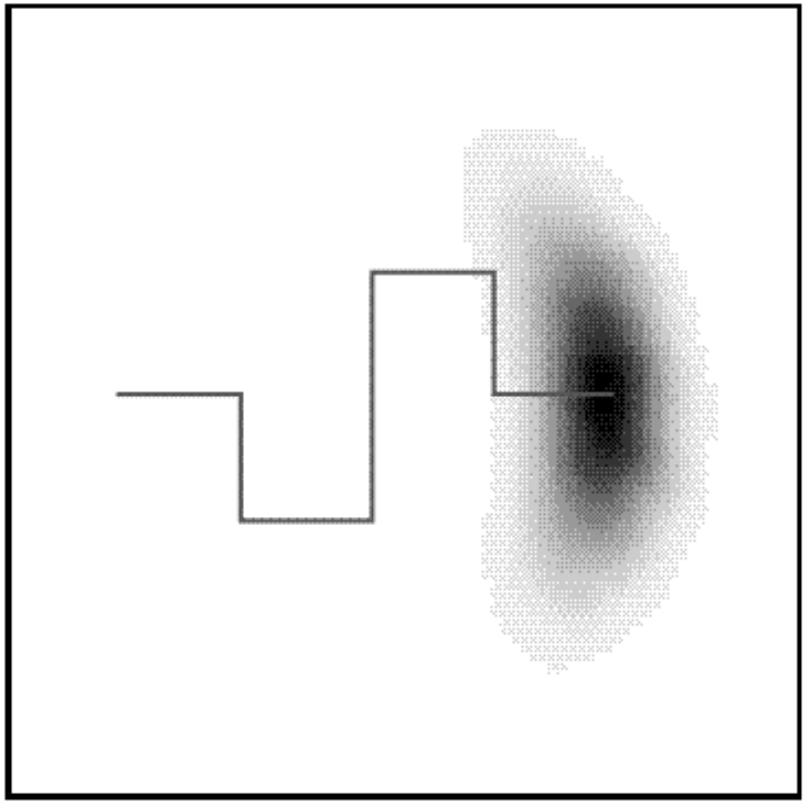
- Odometry u_t and laser scans z_t actually arrive at slightly different times.
- Interpolate to give estimated odometry u_t' at the same time as the laser scan z_t.



Action Model $P(x_t \mid u_{t-1}, x_{t-1})$

- Probability density function over poses, after traveling 40m or 80m.



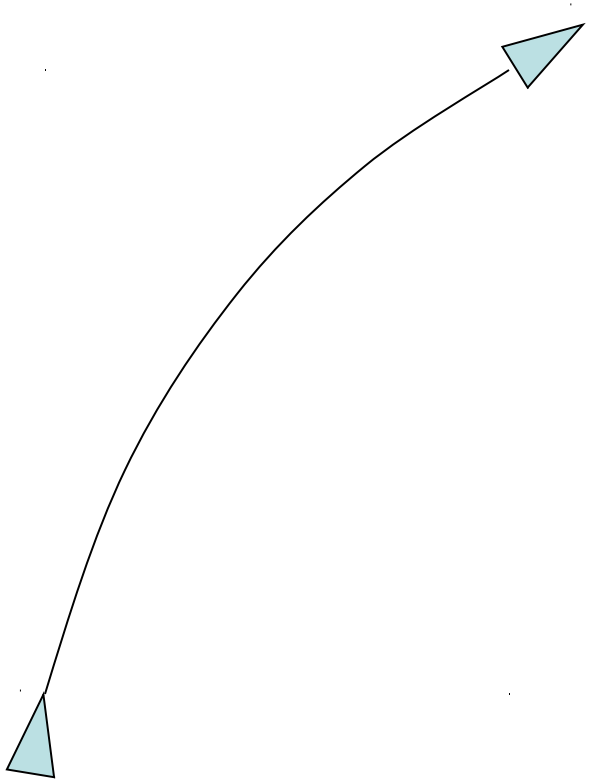


The Action Error Model

Suppose odometry gives:

- (x_1, y_1, φ_1)
- (x_2, y_2, φ_2)

in a slowly drifting frame of reference.



The Action Error Model

From odometry get:

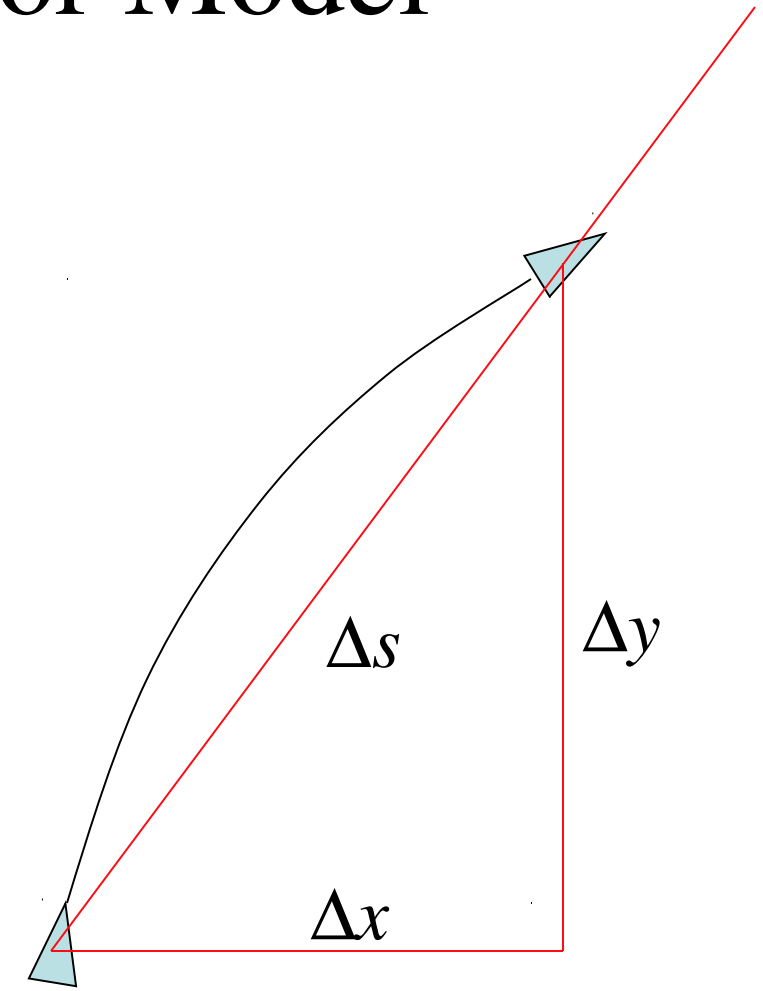
- (x_1, y_1, φ_1)
- (x_2, y_2, φ_2)

in a slowly drifting frame of reference.

Then:

- $(\Delta x, \Delta y, \Delta\varphi)$
- $\Delta x^2 + \Delta y^2 = \Delta s^2$

- Δs and $\Delta\varphi$ are relatively reliable, and independent of the frame of reference.



The Action Error Model

From odometry get:

- (x_1, y_1, φ_1)
- (x_2, y_2, φ_2)

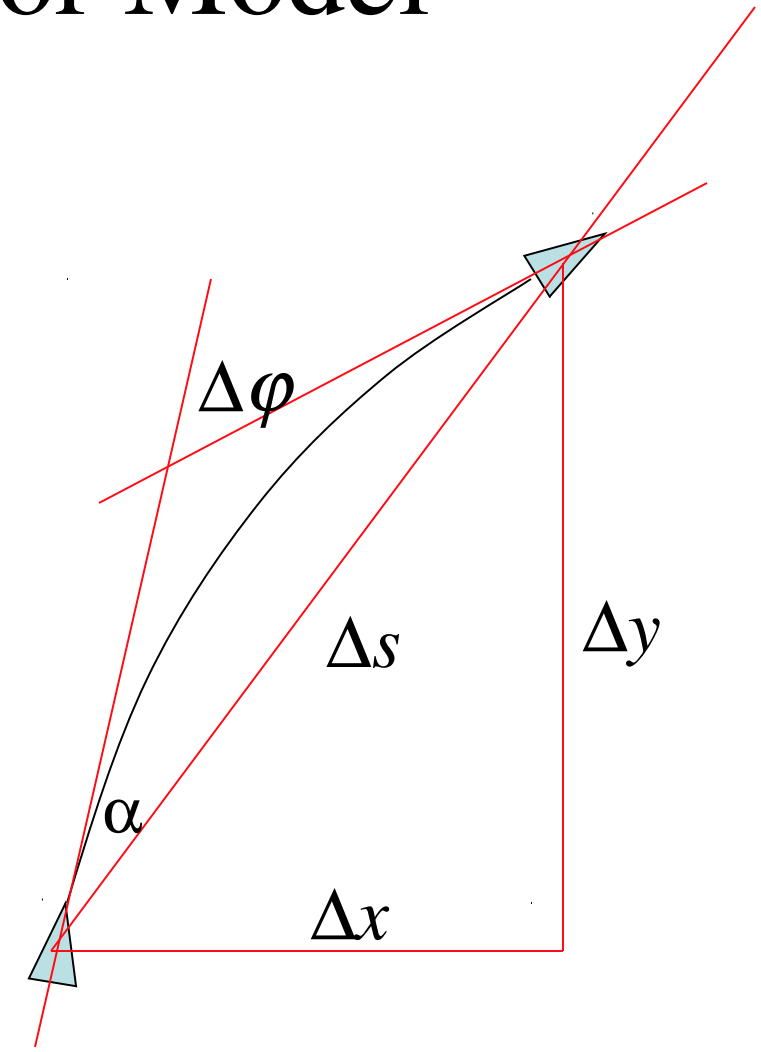
in a slowly drifting frame of reference.

Then:

- $(\Delta x, \Delta y, \Delta\varphi)$
- $\Delta x^2 + \Delta y^2 = \Delta s^2$

and:

- $\alpha = \text{atan2}(\Delta y, \Delta x) - \varphi_1$
- (Measure angle counter-clockwise from x-axis)

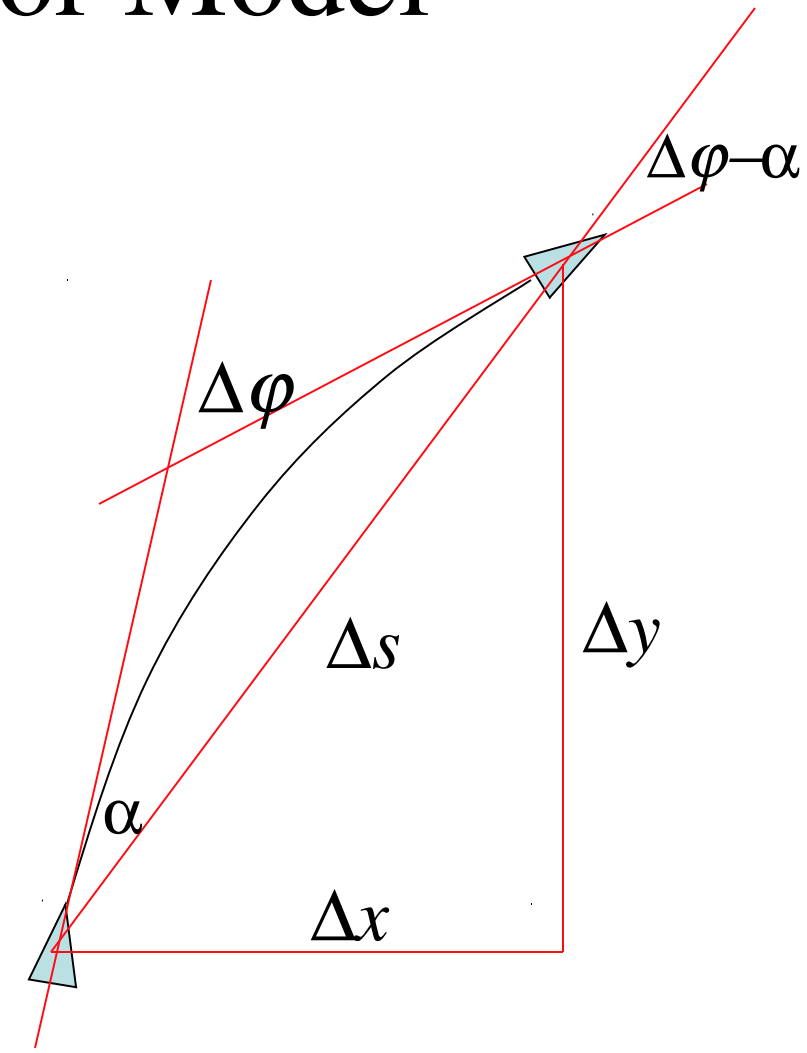


The Action Error Model

From odometry get:

- (x_1, y_1, φ_1)
- (x_2, y_2, φ_2)
- $(\Delta x, \Delta y, \Delta\varphi)$
- $\Delta x^2 + \Delta y^2 = \Delta s^2$
- $\alpha = \text{atan2}(\Delta y, \Delta x) - \varphi_1$

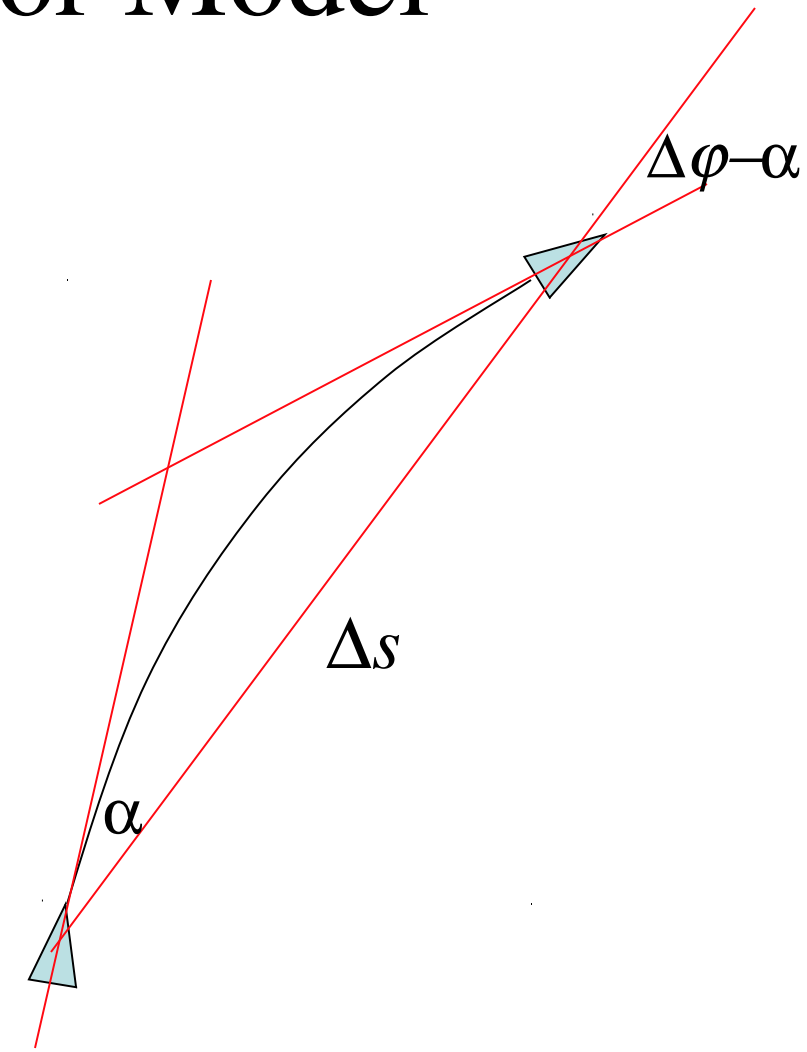
- Model the action as:
  - Turn(α)
  - Travel(Δs)
  - Turn($\Delta\varphi - \alpha$)



The Action Error Model

- Model the action as:
  - Turn($\alpha + \varepsilon_1$)
  - Travel($\Delta s + \varepsilon_2$)
  - Turn($\Delta\varphi - \alpha + \varepsilon_3$)

  where
  - $\varepsilon_1 \sim N(0, k_1\alpha)$
  - $\varepsilon_2 \sim N(0, k_2\Delta s)$
  - $\varepsilon_3 \sim N(0, k_1(\Delta\varphi - \alpha))$
- This combines three Gaussian errors.
  - Std dev proportional to action magnitude



Tune the Action Error Model

- Model the action as:
  - Turn($\alpha + \varepsilon_1$)
  - Travel($\Delta s + \varepsilon_2$)
  - Turn($\Delta\varphi - \alpha + \varepsilon_3$)

  where
  - $\varepsilon_1 \sim N(0, k_1\alpha)$
  - $\varepsilon_2 \sim N(0, k_2\Delta s)$
  - $\varepsilon_3 \sim N(0, k_1(\Delta\varphi - \alpha))$
- Tune the model by finding k_1 and k_2.

- Compute error between:
  - odometry observed
  - odometry corrected by localization.
- Divide by turn or travel magnitude.
- Compute standard deviations
  - $k_1 = 1.0$
  - $k_2 = 0.4$

The Action Model $P(x_t \mid u_{t-1}, x_{t-1})$

- Given a small motion $u_{t-1} = (\alpha, \Delta s, \Delta\varphi - \alpha)$

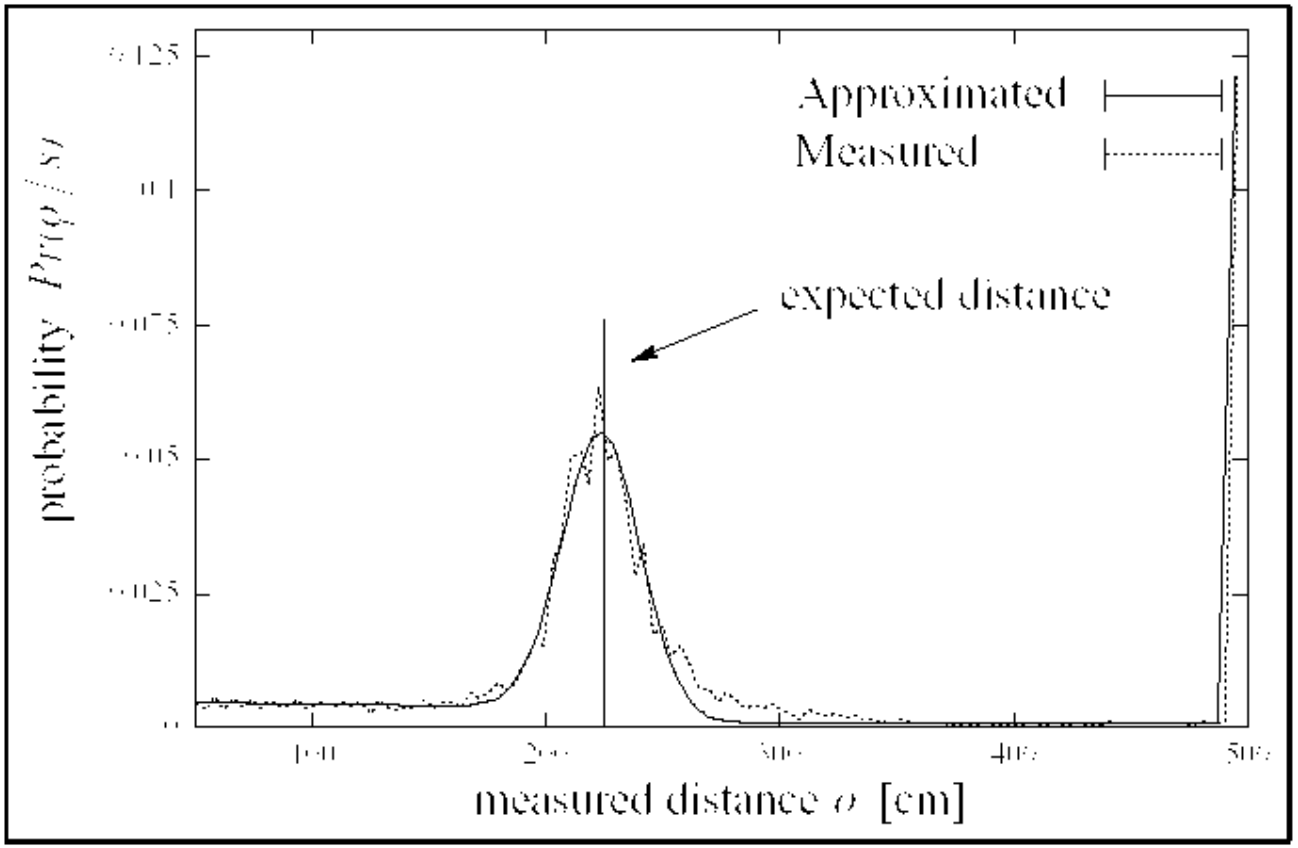
$$\begin{pmatrix} x_t \\ y_t \\ \theta_t \end{pmatrix} = \begin{pmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{pmatrix} + \begin{pmatrix} (\Delta s + \varepsilon_2)\cos(\theta_{t-1} + \alpha + \varepsilon_1) \\ (\Delta s + \varepsilon_2)\sin(\theta_{t-1} + \alpha + \varepsilon_1) \\ \Delta\varphi + \varepsilon_1 + \varepsilon_3 \end{pmatrix}$$

where

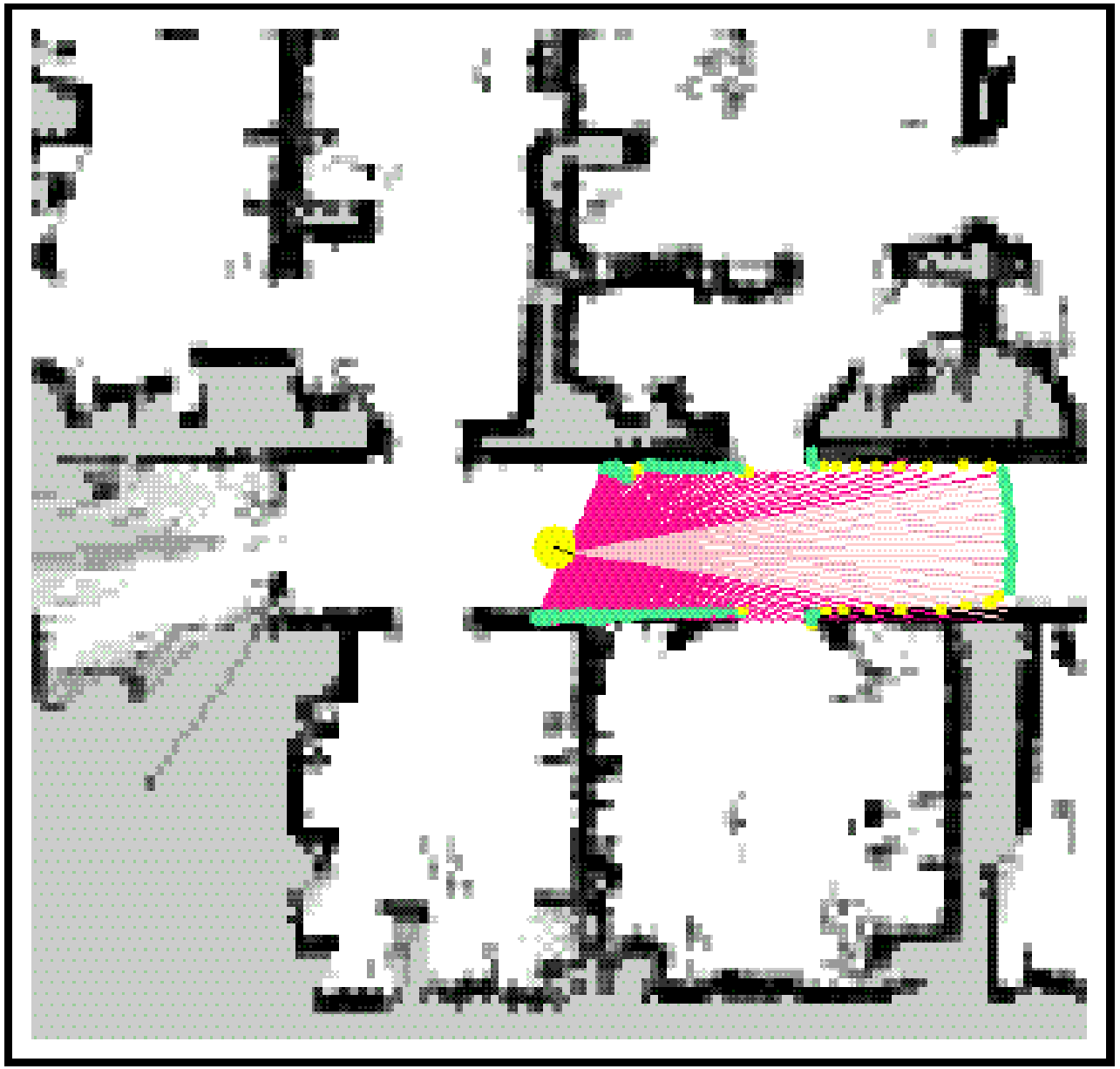
- $\varepsilon_1 \sim N(0, k_1\alpha)$
- $\varepsilon_2 \sim N(0, k_2\Delta s)$
- $\varepsilon_3 \sim N(0, k_1(\Delta\varphi - \alpha))$

Sensor Model $P(z_t \mid x_t)$

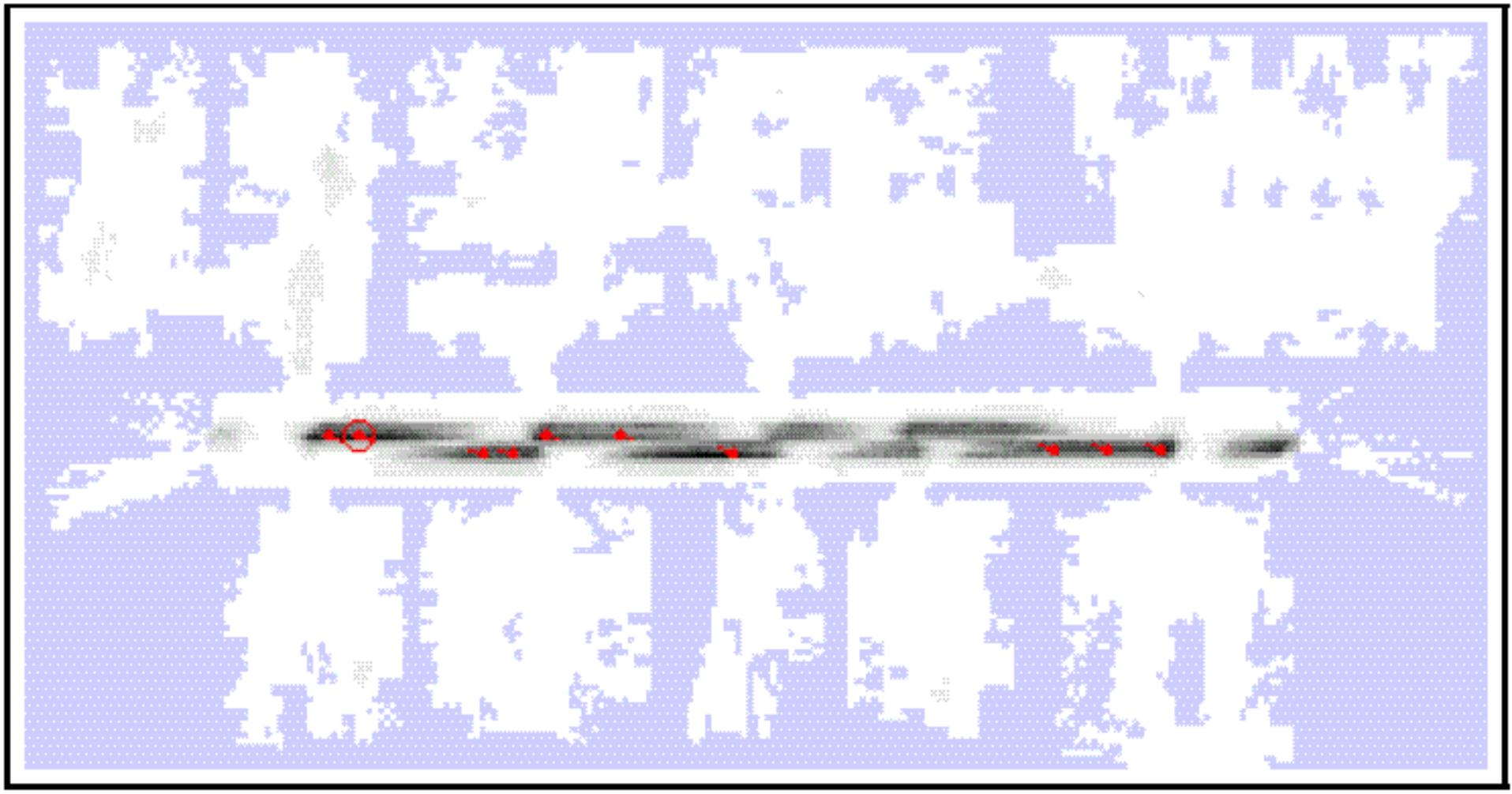
- For a given range measurement at a given location x_t in the map:



Laser Rangefinder Scan z_t



Density $P(z_t \mid x_t)$ by Location



Computing $P(z_t \mid x_t)$

- Measurement probabilities are too small to be meaningful:

$$P(z_t \mid x_t) = \prod_{i=1}^{180} P(ray(i) = d_i \mid x_t)$$

- But log probabilities can be accumulated

$$\log P(z_t \mid x_t) = \sum_{i=1}^{180} \log P(ray(i) = d_i \mid x_t)$$

  - without numerical underflow.

Q: Didn't we use log odds before?

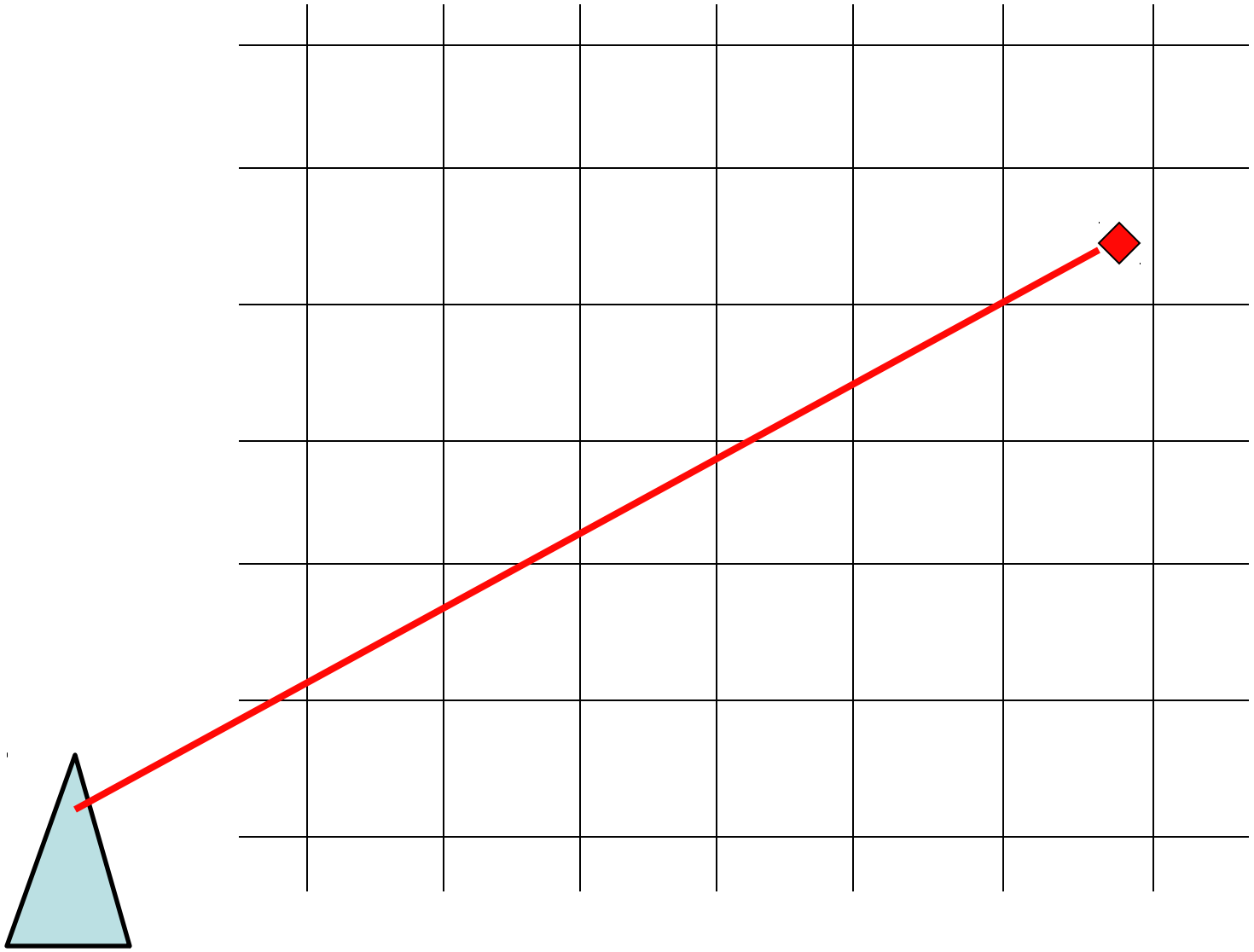
- Yes, but that was in the occupancy grid.
  - It is useful as a way to represent $P(occ(i,j))$
  - Certainty on either side of ignorance

- Here, we're doing Markov localization.

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

- We need to compare values of $P(z_t \mid x_t)$.
  - Odds would just be a distraction
  - Log helps us avoid numerical underflow

$$P(ray(i) = d_i \mid x_t)$$



Computing log $P(ray(i) = d_i \mid x_t)$

- If *ray(i)* terminates at the first obstacle:

  $\log P(ray(i)=d_i \mid x_t) = -4$

- If *ray(i)* terminates before an obstacle:

  $\log P(ray(i)=d_i \mid x_t) = -8$

- If *ray(i)* terminates after an obstacle:

  $\log P(ray(i)=d_i \mid x_t) = -12$

- Add them up for i=1 to 180.

$$\log P(z_t \mid x_t) = \sum_{i=1}^{180} \log P(ray(i) = d_i \mid x_t)$$

  - $\log P(z_t \mid x_t)$ totals between −720 and −2160
  - Exponentiating would give zero!

Markov Localization

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

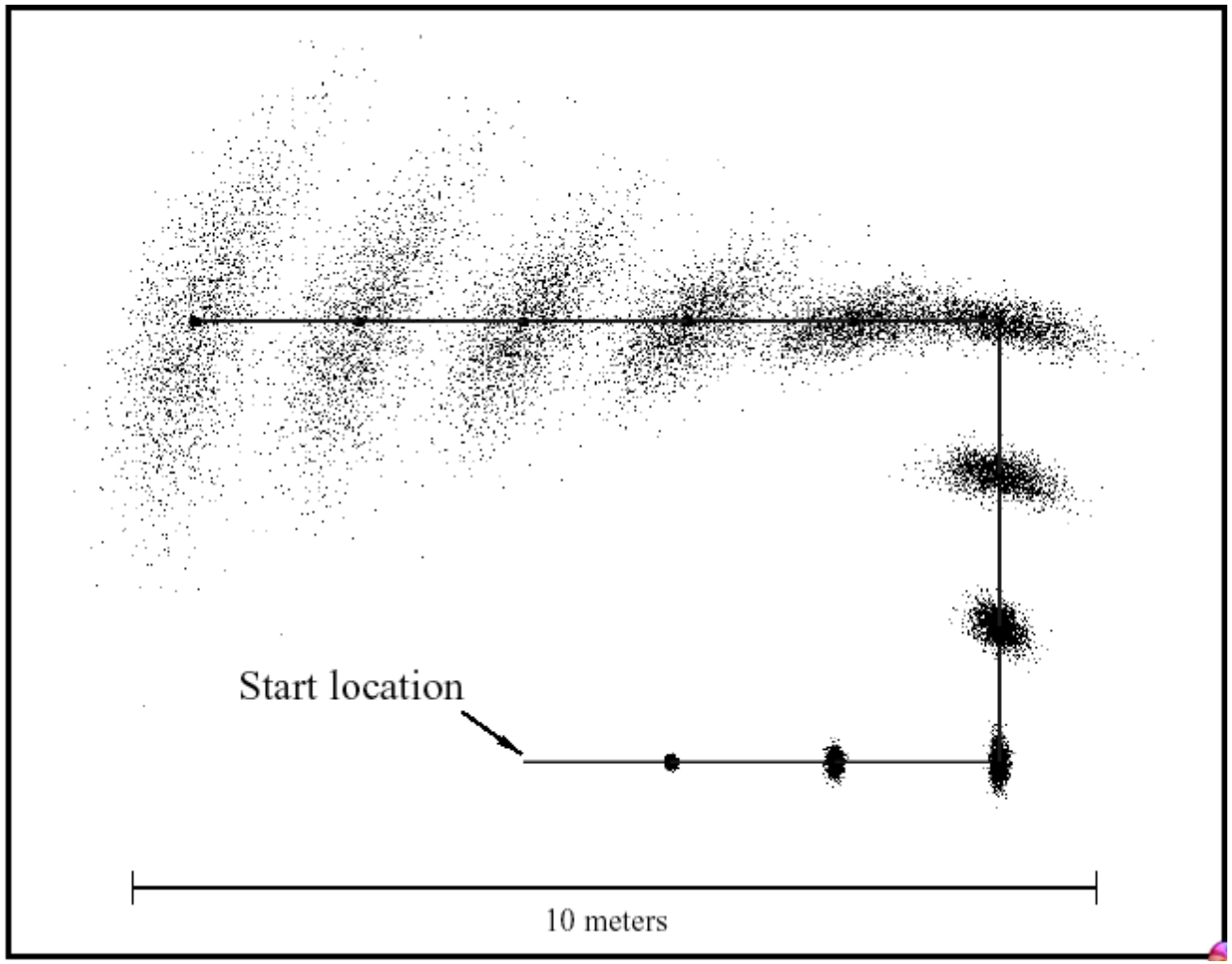
- The integral is evaluated over all x_{t-1}.
  - It computes the probability of reaching x_t from any location x_{t-1}, using the action u_{t-1}.
- The equation is evaluated for every x_t.
  - It computes the posterior probability distribution of x_t.
- Computational efficiency is a problem.
  - $o(k^2)$ if there are k poses x_t.
  - k reflects resolution of position and orientation.

Monte Carlo Simulation

- Given a probability distribution over inputs, computing the distribution over outcomes can be hard.
  - Simulating a concrete instance is easy.
- Sample concrete instances ("particles") from the input distribution.
- Collect the outcomes.
  - The distribution of sample outcomes approximates the desired distribution.
- This has been called "particle filtering."

Actions Disperse the Distribution

- *N* particles approximate a probability distribution.
- The distribution disperses under actions



Monte Carlo Localization

- A concrete instance is a particular *pose*.
  - A *pose* is position plus orientation.
- A probability distribution is represented by a collection of N poses.
  - Each pose has an importance factor.
  - The importance factors sum to 1.
- Initialize with
  - N uniformly distributed poses.
  - Equal importance factors of N^{-1}.

Representing a Distribution

- The distribution $Bel(x_t)$ is represented by a set S_t of N weighted samples:

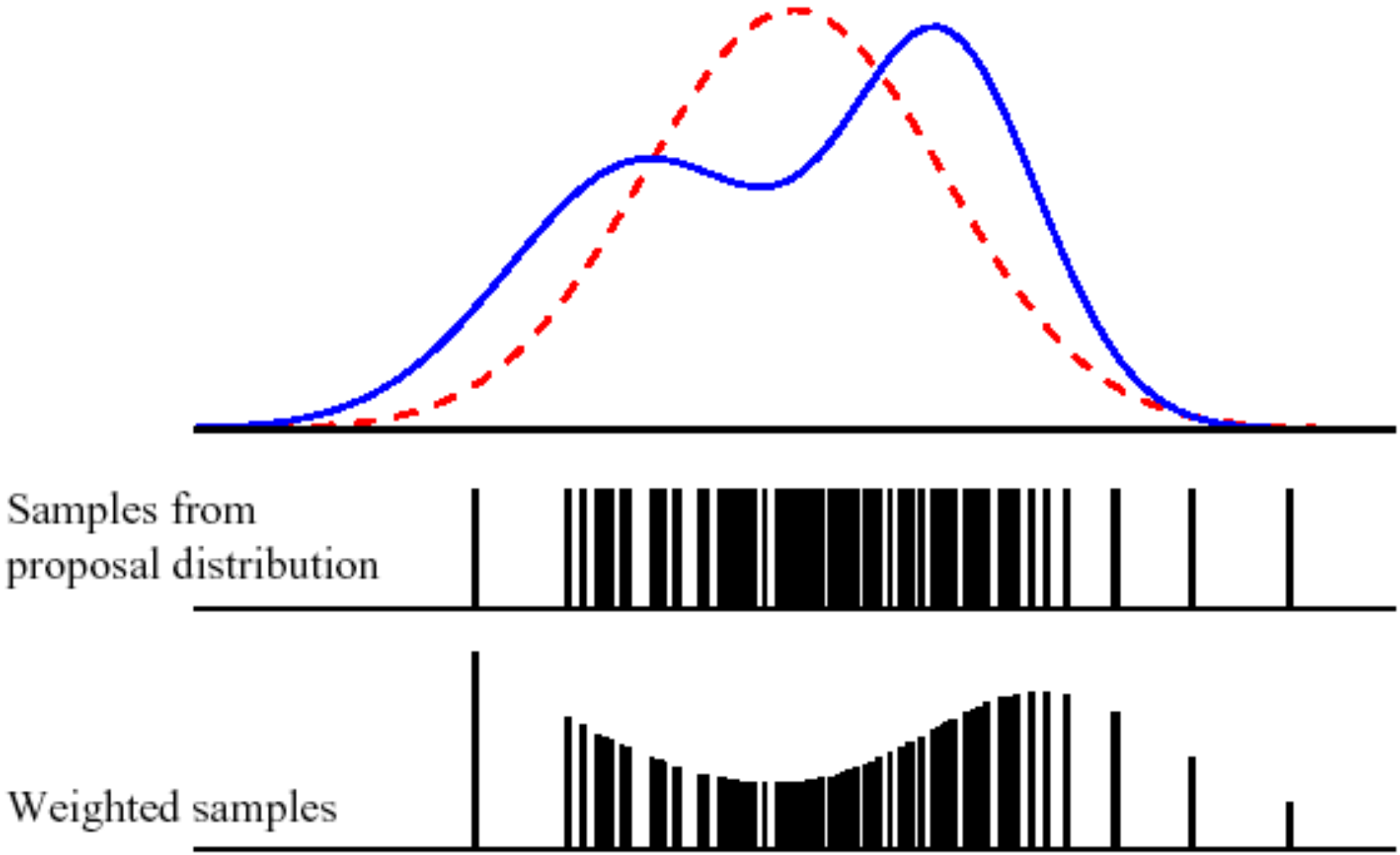
$$S_t = \left\{ \left\langle x_t^{(i)}, w_t^{(i)} \right\rangle \mid i = 1, \cdots N \right\}$$

where $\sum_{i=1}^{N} w_t^{(i)} = 1$

- A *particle filter* is a Bayes filter that uses this sample representation.

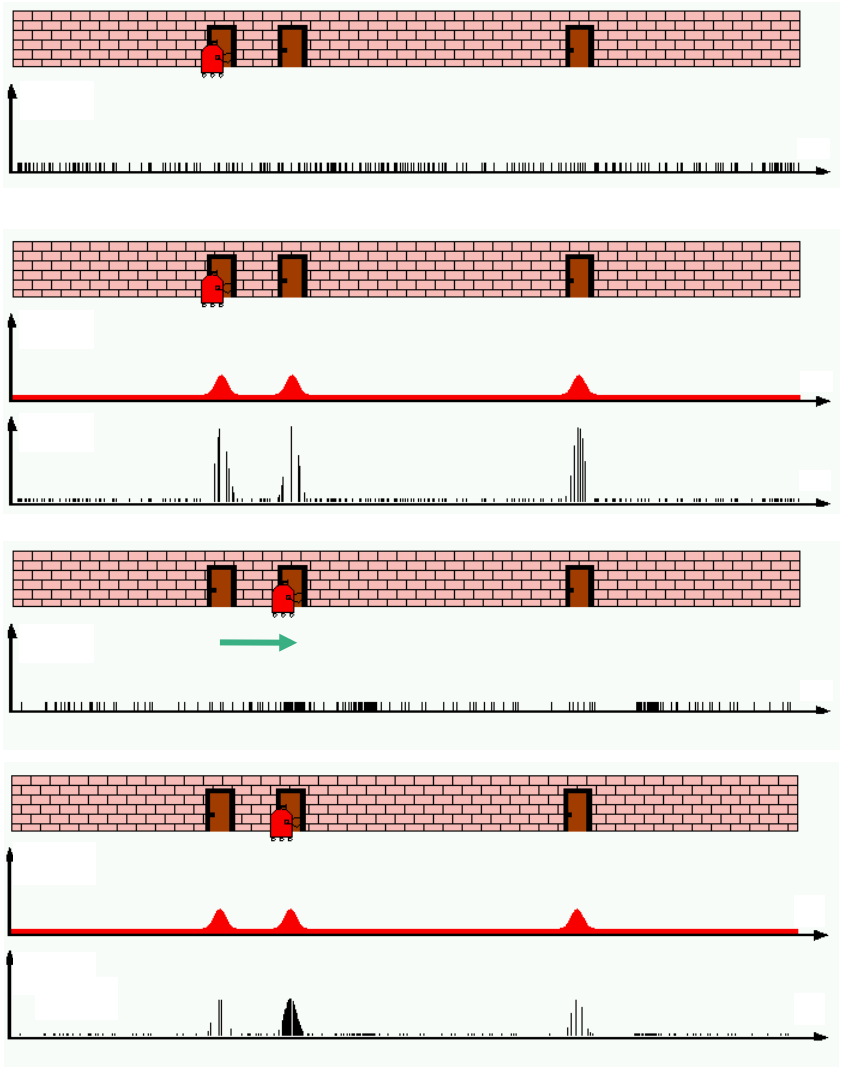
Importance Sampling

- Sample from a *proposal* distribution.
  - Correct to approximate a *target* distribution.



Simple Example

- Uniform distribution
- Weighting by sensor model
- Prediction by action model
- Weighting by sensor model

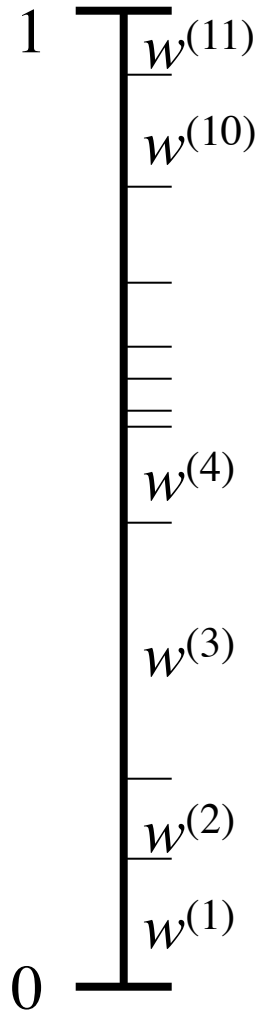


The Basic Particle Filter Algorithm

- **Input**: u_{t-1}, z_t, $S_{t-1} = \left\{ \left\langle x_{t-1}^{(i)}, w_{t-1}^{(i)} \right\rangle \mid i = 1, \cdots N \right\}$
  - $S_t := \varnothing$, $i := 1$, $\alpha := 0$
- **while** $i \leq N$ **do**
  - sample j from the discrete distribution given by the weights in S_{t-1}
  - sample $x_t^{(i)}$ from $p(x_t \mid u_{t-1}, x_{t-1})$ given $x_{t-1}^{(j)}$ and u_{t-1}.
  - $w_t^{(i)} := p(z_t \mid x_t^{(i)})$
  - $\alpha := \alpha + w_t^{(i)}$; $\quad i := i + 1$
  - $S_t := S_t \cup \{\langle x_t^{(i)}, w_t^{(i)} \rangle\}$
- **for** $i := 1$ to N **do** $w_t^{(i)} := w_t^{(i)} / \alpha$
- **return** S_t

Sampling from a Weighted Set of Particles

- Given $S_t = \left\{ \left\langle x_t^{(i)}, w_t^{(i)} \right\rangle \mid i = 1, \cdots N \right\}$
- Draw α from a uniform distribution over [0,1].
- Find the minimum k such that $\sum_{i=1}^{k} w_t^{(i)} > \alpha$
- Return $x_t^{(k)}$



MCL Algorithm

- Repeat to collect N samples.
  - Draw a sample x_{t-1} from the distribution $Bel(x_{t-1})$, with likelihood given by its importance factor.
  - Given x_{t-1} and action u_{t-1}, sample state x_t from the action model distribution $P(x_t \mid u_{t-1}, x_{t-1})$,
  - Assign the importance factor $P(z_t \mid x_t)$ to x_t.
- Normalize the importance factors.
- Repeat for each time-step.

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