Lecture: Localisation 2

Particle Filtering 101

MOBILE ROBOT CONTROL 2023

Koen de Vos

Department of Mechanical Engineering – Control Systems Technology

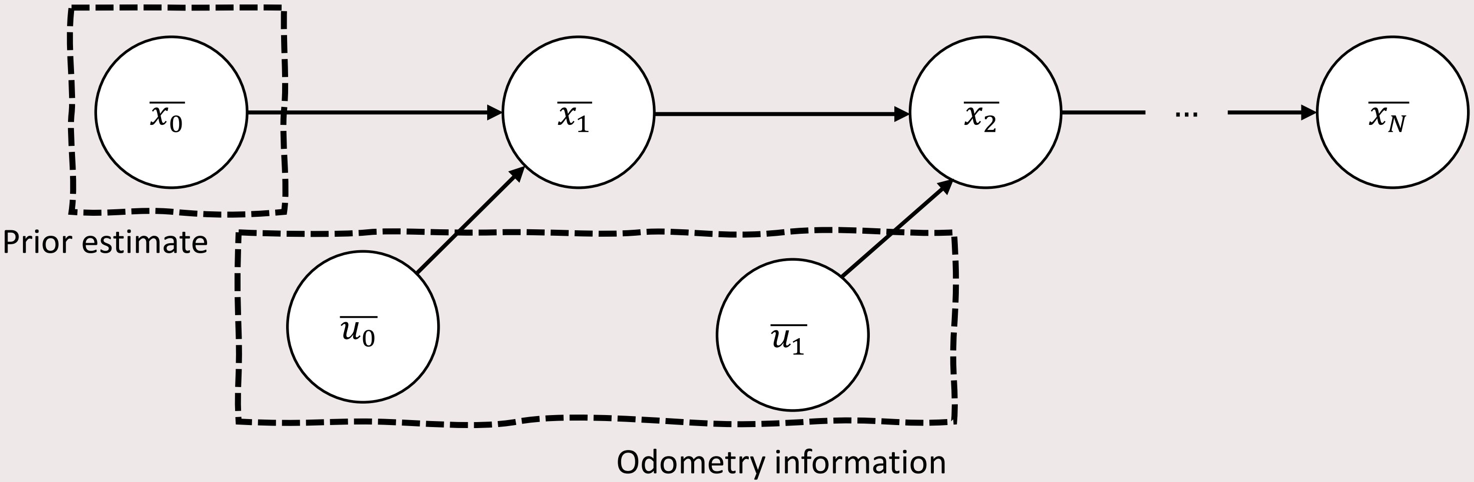
TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Recursive State Estimation

Recap

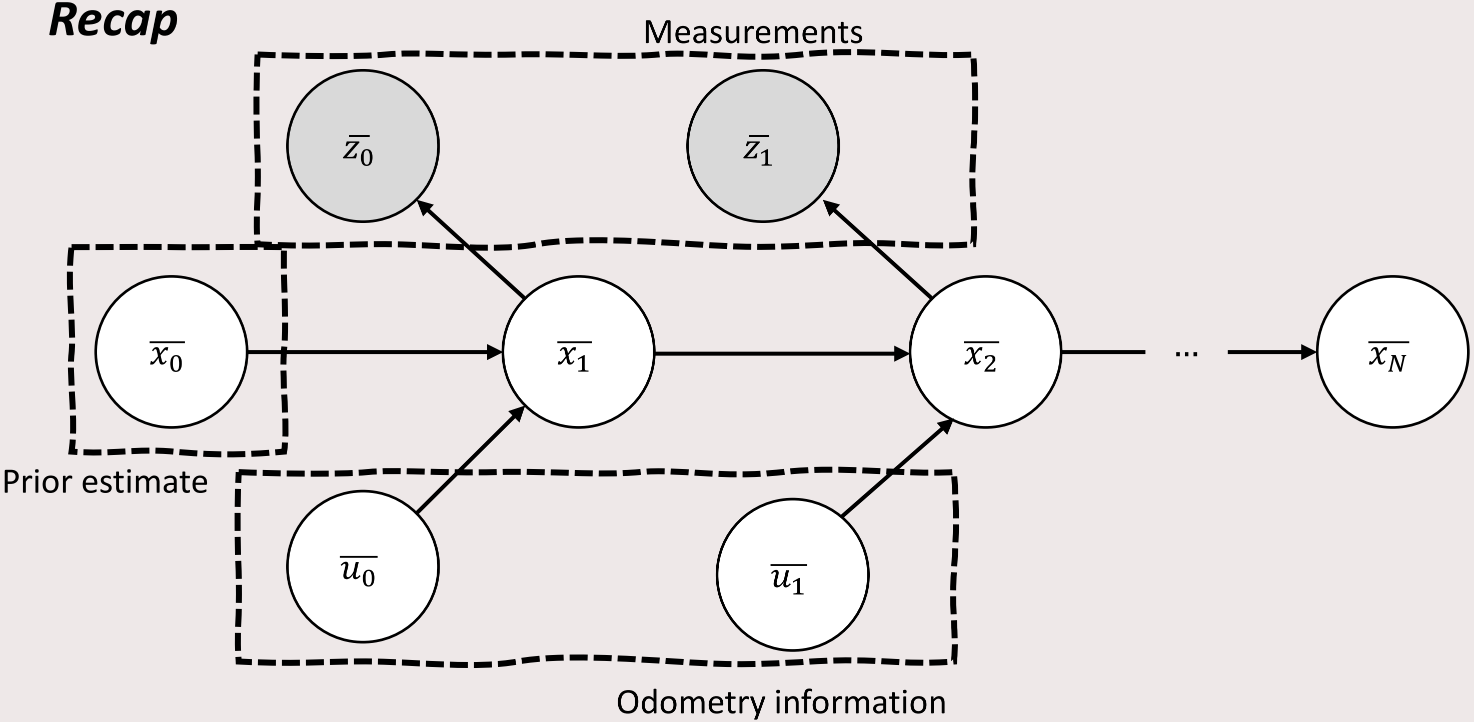
Recursive State-Estimation

Recap

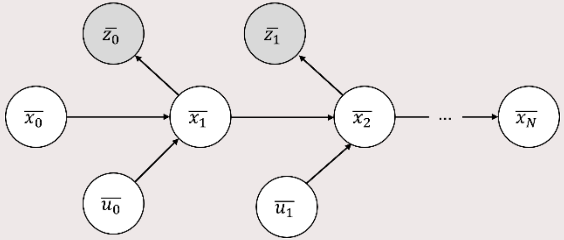


Recursive State-Estimation

Recap



Recursive State Estimation
Recap



Our belief of the current state:

$$bel(x_t) \coloneqq p(x_t | z_{1:t-1}, u_{1:t-1})$$

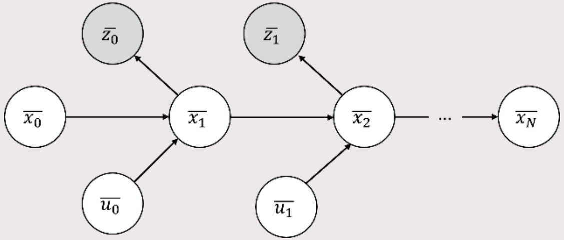
Measurement update:

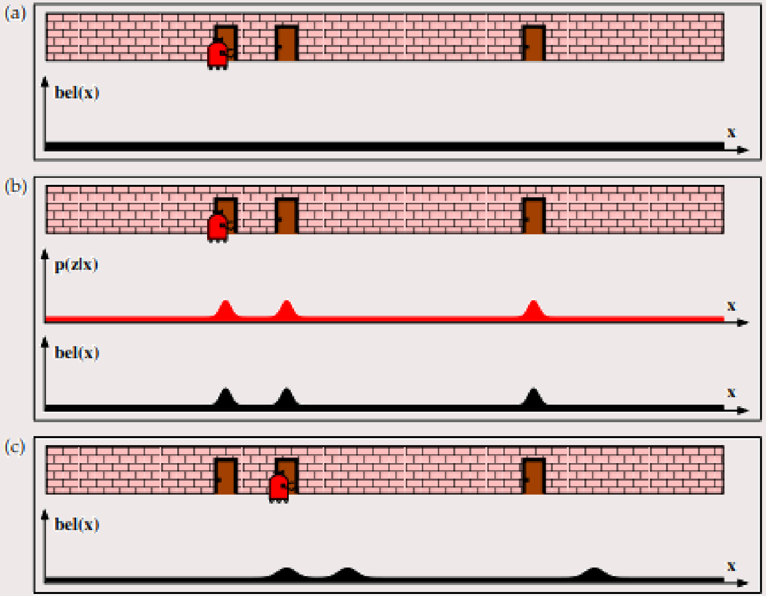
$$bel(x_t) = p(z_t | x_t)\overline{bel}(x_t)$$

Control (dead-reckoning) update:

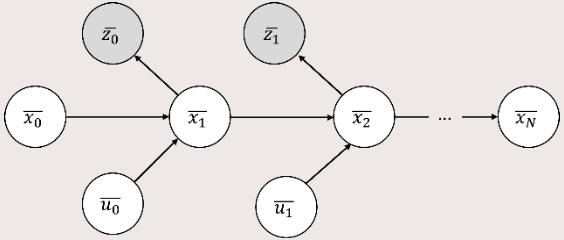
$$\overline{bel}(x_t) = \int p(x_t | u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$$

Recursive State Estimation
Recap





Recursive State Estimation
Recap



However:

- The Bayes filter is generally intractable

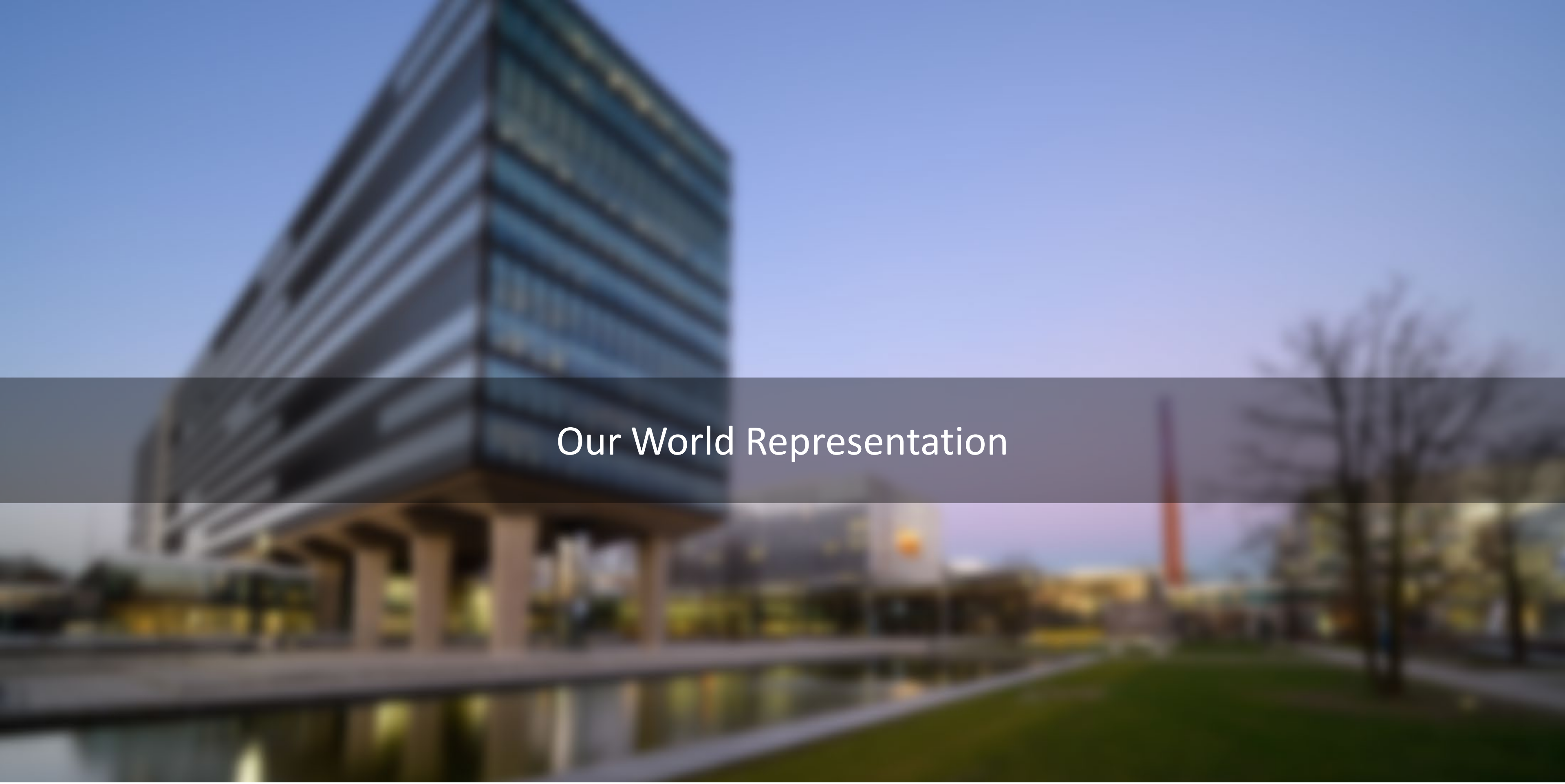
We need to approximate

Goal of this Lecture

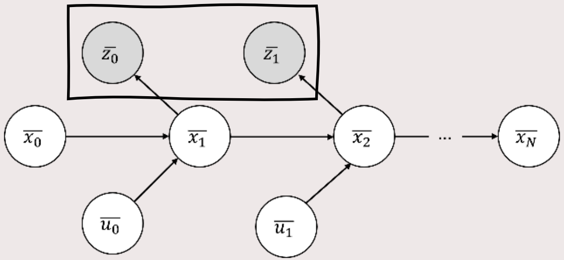
How do we represent our World?

What do we mean by “measurements”?

How do we solve the Localization
Problem using a Particle Filtering approach?



Our World Representation



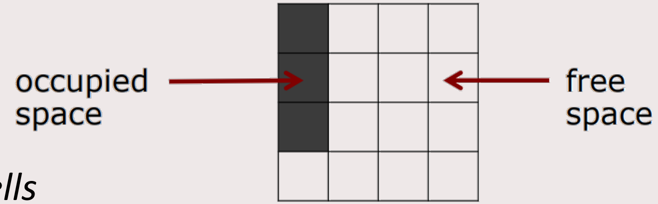
Map Representation

Occupancy Grid Maps

- Grid Map -> The world is discretized in a large amount of cells
- Occupancy -> Each cell is either occupied or free
- Often assumed to be static

- *Note that large environments may require a large amount of cells*

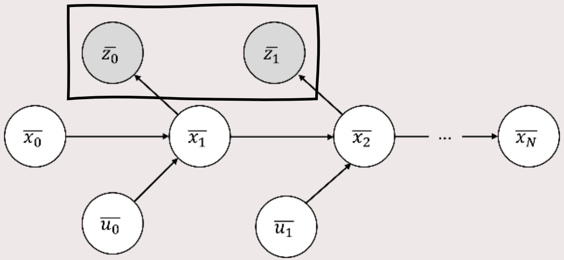
- *No representation for higher level features, e.g. doors*



http://ais.informatik.uni-freiburg.de/teaching/ws22/mapping/

The Measurement Model

Recursive State Estimation
Measurements



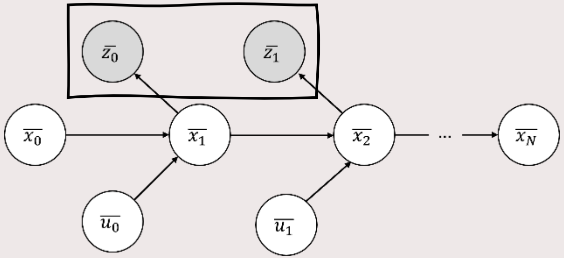
So far we've seen:

Measurement update:

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

But what is z_t and how do we express $p(z_t|x_t)$?

Recursive State Estimation
Beam-based model

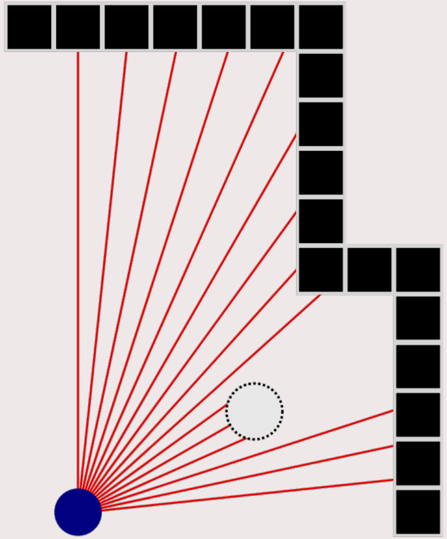


For now, let's define a measurement as a vector of ranges:

- $z_k = \begin{bmatrix} (r_0, \theta_0) \\ (r_1, \theta_1) \\ \vdots \\ (r_2, \theta_2) \end{bmatrix}$,

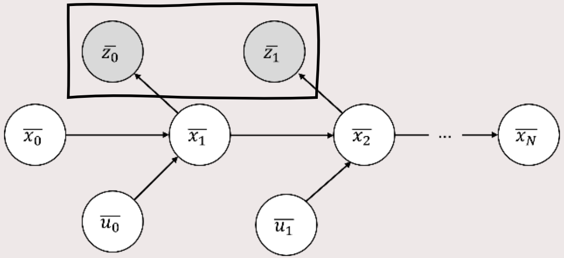
- Given a map, a robot pose, and *appropriate algorithms* we can generate a prediction of this measurement

- $z_k^* = \begin{bmatrix} (r_0^*, \theta_0) \\ (r_1^*, \theta_1) \\ \vdots \\ (r_2^*, \theta_2) \end{bmatrix}$



TU/e

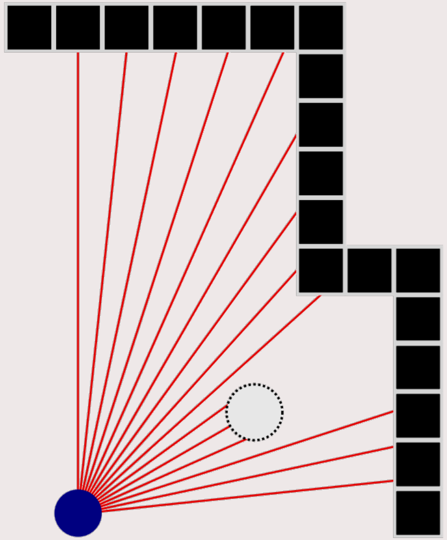
Recursive State Estimation
Beam-based model



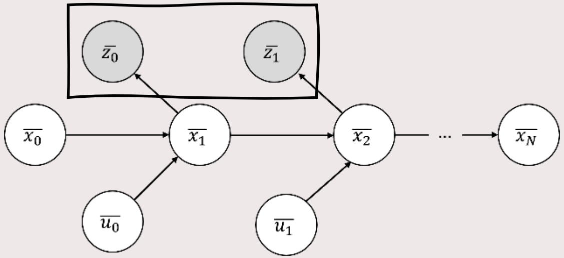
Appropriate algorithms?

A family of algorithms called Ray casters.

Don’t worry about them for now,
we have provided you with one for the assignment.

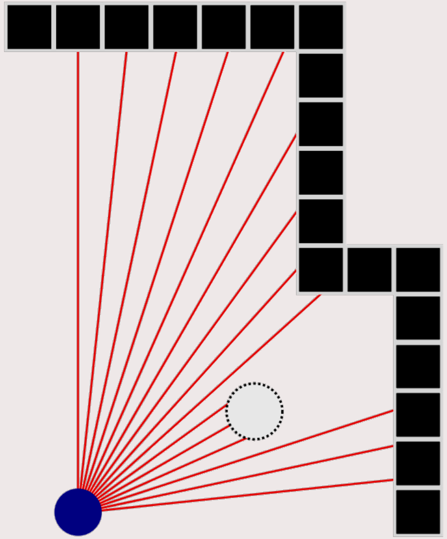


Recursive State Estimation
Beam-based model

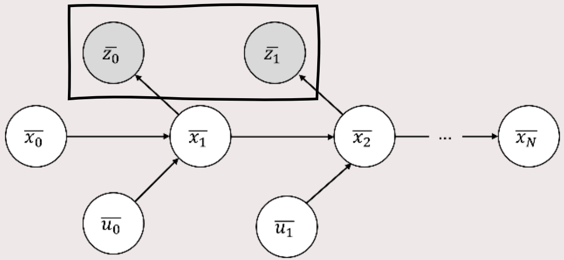


Observe that we now have a measurement and a measurement prediction in the ideal (modeled) case.

Core Idea:
The mismatch between the two tell us something about whether the robot pose is correct.

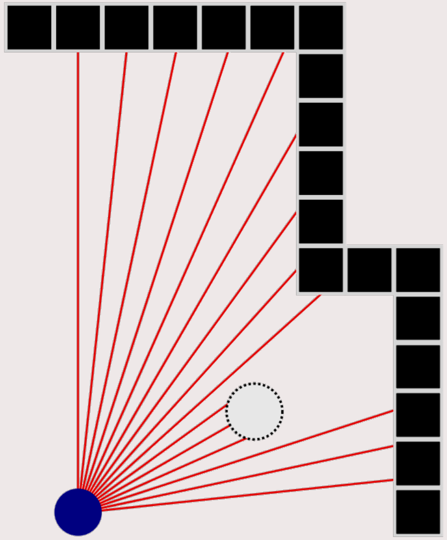


Recursive State Estimation
Beam-based model

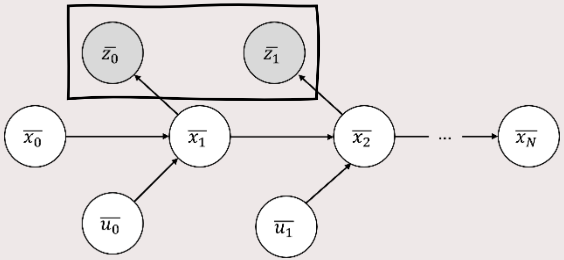


How to quantify this
mismatch as a probability $p(z_t|x_t)$?

- For a single ray, we identify four sources of "disturbances"
  - Local Measurement noise
  - Unexpected Obstacles
  - Failures (Glass, Black obstacles)
  - Random measurements
- We assign each source a distribution and probability of occurring



Recursive State Estimation
Beam-based model



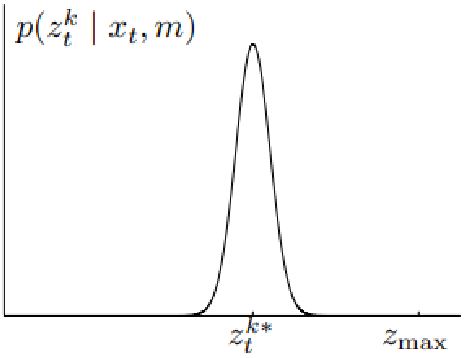
Local Measurement Noise

$$p_{short}(z_t^k | x_t, m) = \begin{cases} \eta\, \mathcal{N}(z_t^k; z_t^{k*}, \sigma_{hit}^2) & if\ 0 \leq z_t^k \leq z_{max} \\ 0 & otherwise \end{cases}$$

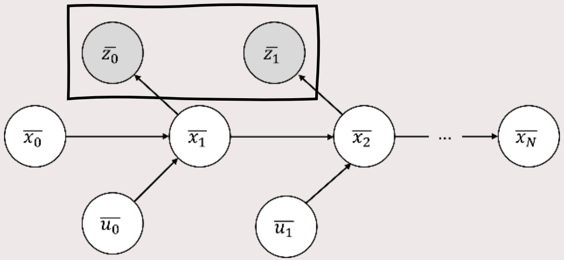
$$\mathcal{N}(z_t^k; z_t^{k*}, \sigma_{hit}^2) = \frac{1}{\sqrt{2\pi\sigma_{hit}^2}} e^{-\frac{1}{2}\frac{(z_t^k - z_t^{k*})^2}{\sigma_{hit}^2}}$$

$$\eta = \left(\int_0^{z_{max}} \mathcal{N}(z_t^k; z_t^{k*}, \sigma_{hit}^2) dz_t^k \right)^{-1}$$

(a) Gaussian distribution p_{hit}



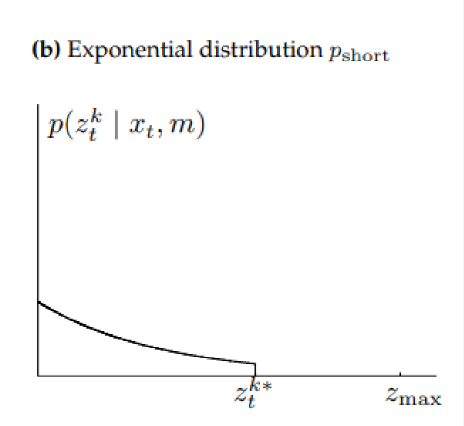
Recursive State Estimation
Beam-based model



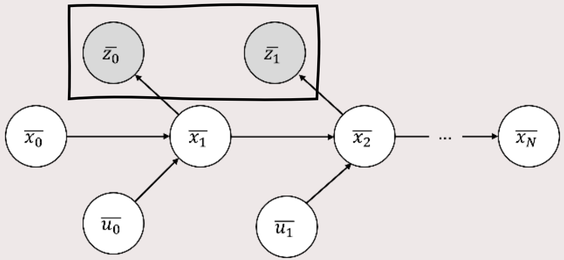
Unexpected Obstacles

$$p_{short}(z_t^k|x_t, m) = \begin{cases} \eta\lambda_{short}e^{-\lambda_{short}\, z_t^k} & if\ 0 \le z_t^k \le z_t^{k*} \\ 0 & otherwise \end{cases}$$

$$\eta = \frac{1}{1 - e^{-\lambda_{short}\, z_t^{k*}}}$$

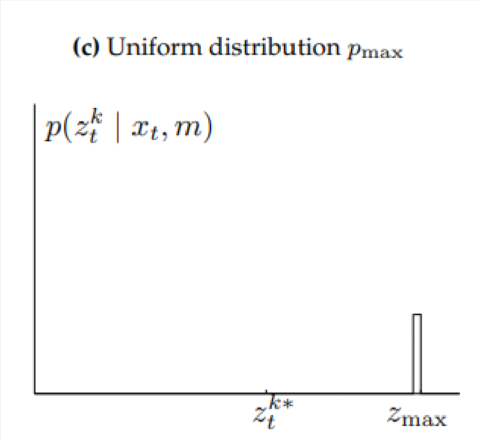


Recursive State Estimation
Beam-based model

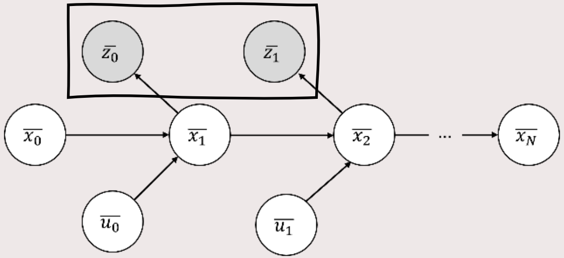


Failures (Glass, Black obstacles)

$$p_{max}\left(z_t^k \middle| x_t, m\right) = \begin{cases} 1 & z_k^t = z_{max} \\ 0 & otherwise \end{cases}$$

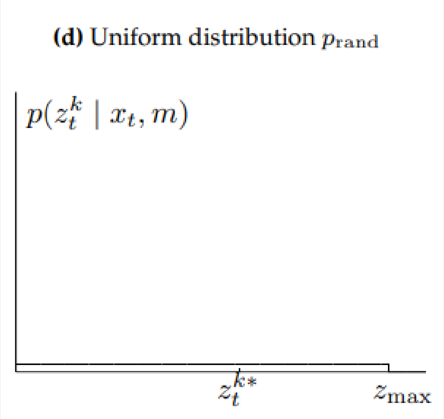


Recursive State Estimation
Beam-based model

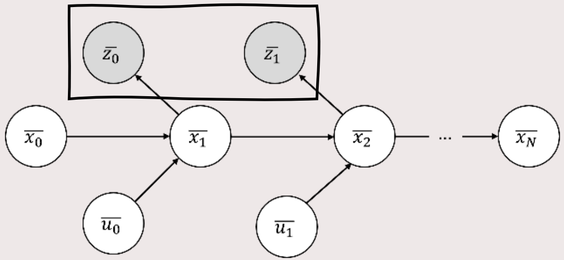


Random measurements

$$p_{rand}\left(z_t^k \middle| x_t, m\right) = \begin{cases} \dfrac{1}{z_{max}} & if\ 0 \le z_k^t < z_{max} \\ 0 & otherwise \end{cases}$$

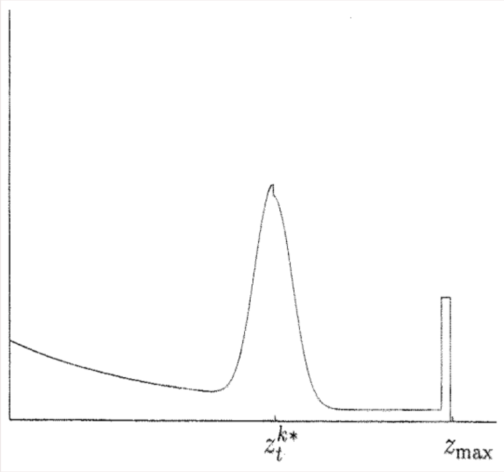


Recursive State Estimation
Beam-based model

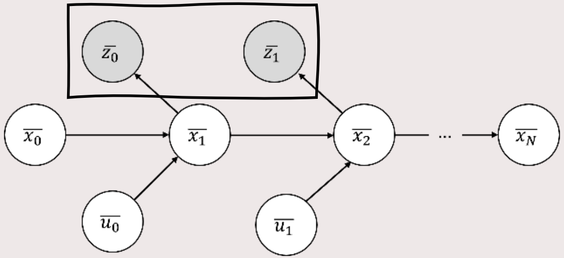


Taking the weighted average of these distributions yields:

$$p(z_t^k|x_t,m) = z_{hit}\, p_{hit}\,(z_t^k|x_t,m) + z_{short} p_{short}(z_t^k|x_t,m) + z_{max} p_{max}(z_t^k|x_t,m) + z_{rand} p_{rand}(z_t^k|x_t,m)$$



Recursive State Estimation
Beam-based model



Probability of entire measurement vector by assuming **independence** of rays.

Algorithm beam_range_finder_model(z_t, x_t, m):

$q = 1$

for $k = 1$ to K do

  compute z_t^{k*} for the measurement z_t^k using ray casting

  $p = z_{\text{hit}} \cdot p_{\text{hit}}(z_t^k \mid x_t, m) + z_{\text{short}} \cdot p_{\text{short}}(z_t^k \mid x_t, m)$

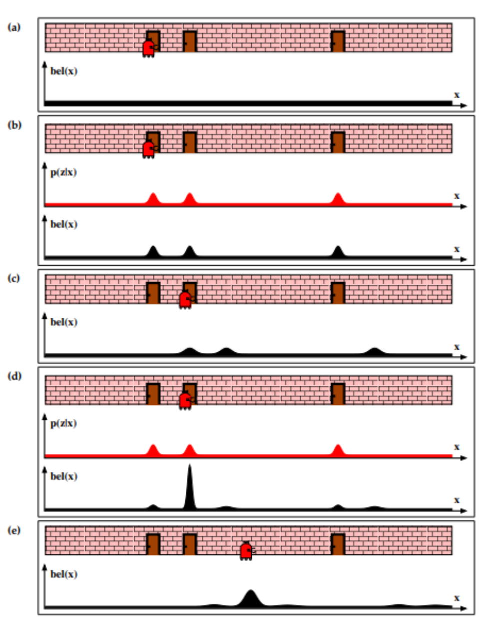
  $\quad + z_{\text{max}} \cdot p_{\text{max}}(z_t^k \mid x_t, m) + z_{\text{rand}} \cdot p_{\text{rand}}(z_t^k \mid x_t, m)$

  $q = q \cdot p$

return q

The Particle Filter Algorithm

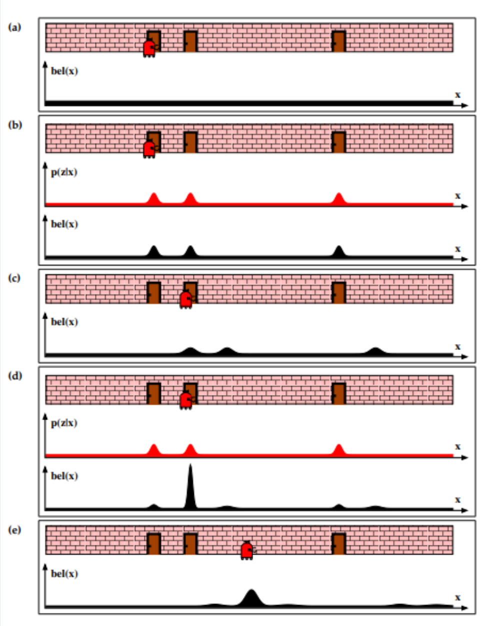
From Bayes Filter to Particle Filter

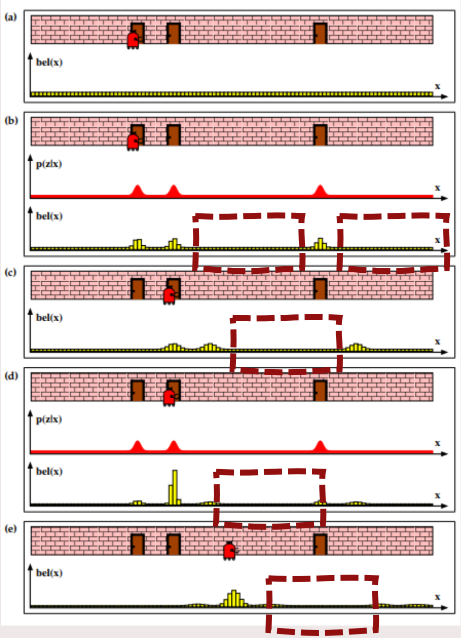


Intractable, since we do not have explicit formulas for the distribution

What if we discretize ("grid") the state space?

From Bayes Filter to Particle Filter

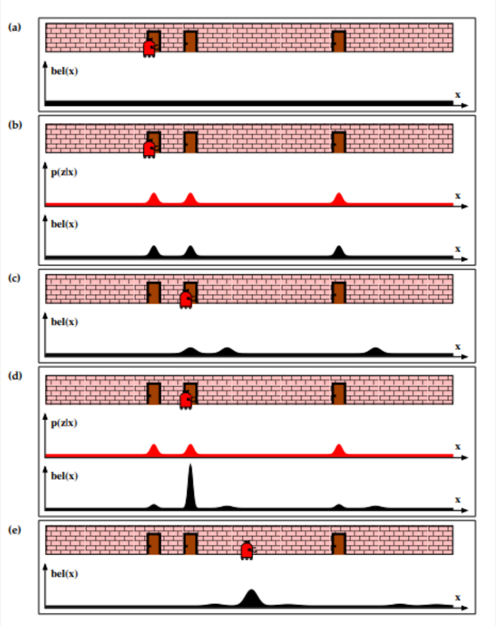


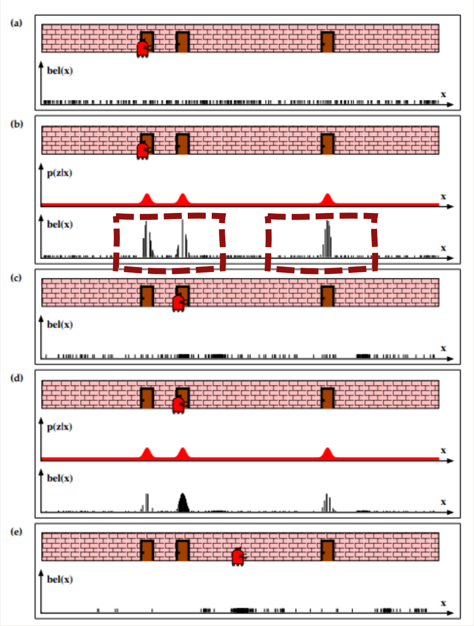


Do we need to keep track of and compute these Areas?

What if we only compute probabilities of regions in which we are likely to be?

From Bayes Filter to Particle Filter





Distribution is sampled randomly. Opportunity to place samples where they are "useful".

Particle Filter (Monte Carlo Localization)

How?

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 $\mathcal{X}_t$

Particle Filter (Monte Carlo Localization)

How?

Algorithm Particle_filter($\mathcal{X}_{t-1}, u_t, z_t$):

$\quad \bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$

$\quad$ for $m = 1$ to M do

$\quad\quad$ sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$

$\quad\quad w_t^{[m]} = p(z_t \mid x_t^{[m]})$

$\quad\quad \bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$

$\quad$ endfor

$\quad$ for $m = 1$ to M do

$\quad\quad$ draw i with probability $\propto w_t^{[i]}$

$\quad\quad$ add $x_t^{[i]}$ to $\mathcal{X}_t$

$\quad$ endfor

$\quad$ return $\mathcal{X}_t$

What we have discussed so far

Particle Filter (Monte Carlo Localization)

How?

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 $\mathcal{X}_t$

Update the m-th particle using the control model

Particle Filter (Monte Carlo Localization)

How?

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 $\mathcal{X}_t$

Set the weight of the updated particle using the measurement model

Particle Filter (Monte Carlo Localization)

How?

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

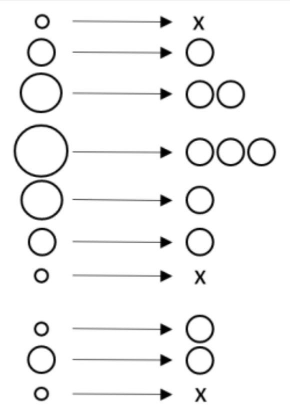
for $m = 1$ to M do

  draw i with probability $\propto w_t^{[i]}$

  add $x_t^{[i]}$ to $\mathcal{X}_t$

endfor

return $\mathcal{X}_t$

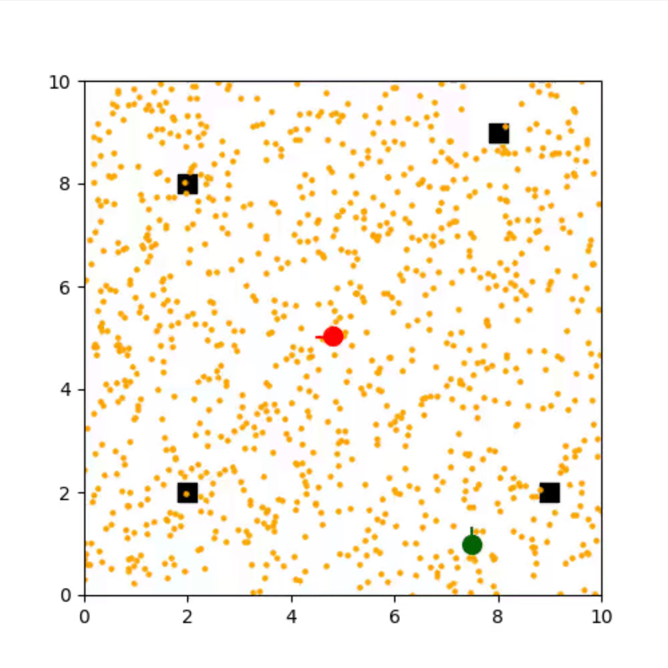


Elfring, J.; Torta, E.; van de Molengraft, R. Particle Filters: A Hands-On Tutorial. *Sensors* **2021**, *21*, 438. https://doi.org/10.3390/s21020438

Resampling: Makes sure that particles stay focused in high likelihood areas

You will see two methods in the Assignment

Now it's your turn



Recommended to use Probabilistic Robotics as a reference