MTRN3100 Robot Design
Week 9 – Localisation II

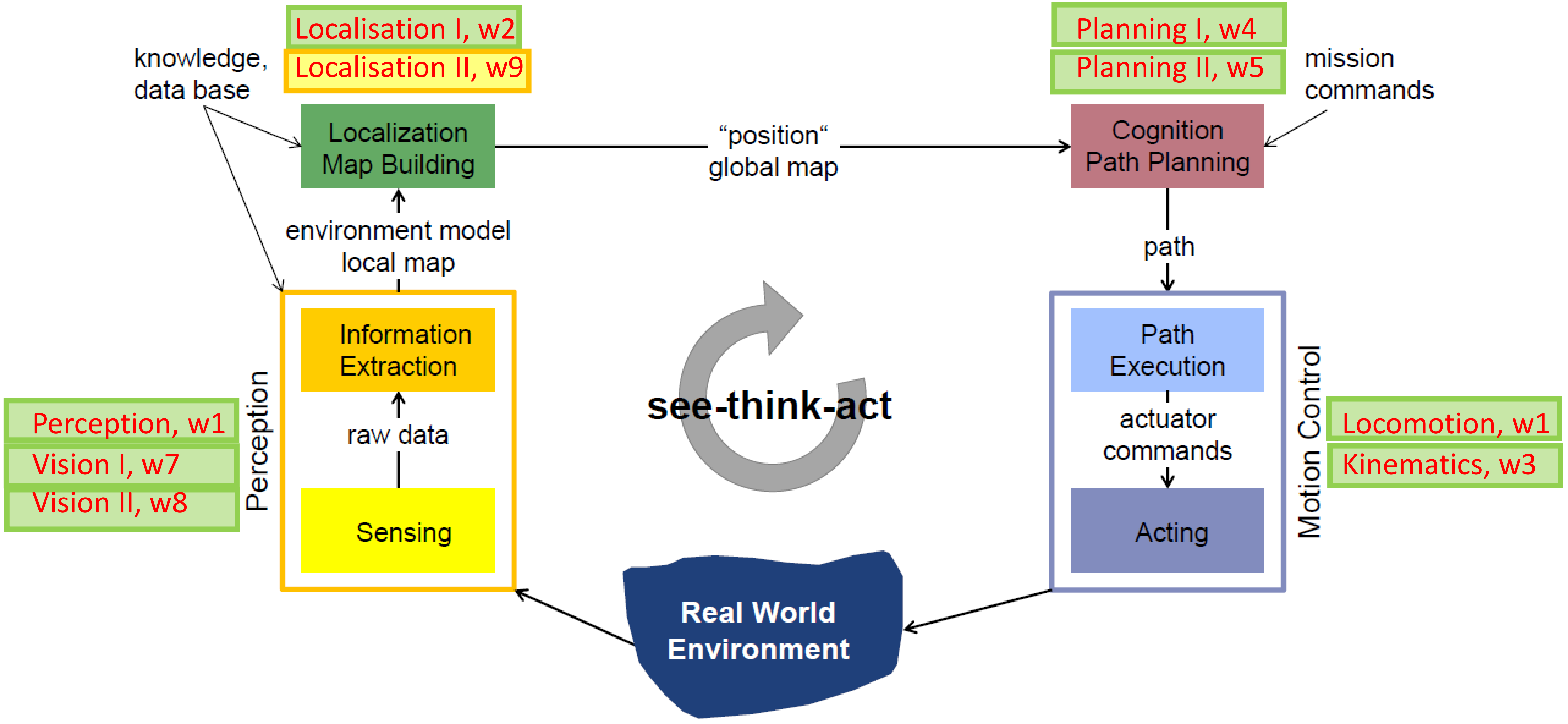
Liao "Leo" Wu, Senior Lecturer

School of Mechanical and Manufacturing Engineering

University of New South Wales, Sydney, Australia

https://www.drliaowu.com/

The See-Think-Act cycle



What we have learnt in Localisation I

- Introduction to localisation
  - Map-based approach vs Behaviour-based approach
- Map representations
  - Continuous line-based
  - Cell decomposition
    - Exact cell decomposition
    - Fixed cell decomposition
    - Adaptive cell decomposition
  - Topological map
- Localisation methods
  - Localisation based on landmarks/artificial markers/external sensors
  - Dead reckoning/odometry
  - Probabilistic map based localisation
  - Simultaneous Localisation and Mapping (SLAM)

Today’s agenda

- Probabilistic map-based localisation
  - Markov localisation
  - Particle Filter localisation
  - Kalman Filter localisation

- Simultaneous Localisation and Mapping (SLAM)
  - Extended Kalman Filter SLAM
  - Graph-based SLAM
  - Particle Filter SLAM

Probabilistic Map-Based Localisation

Probabilistic map-based localisation

- Localisation:
  - The process that the robot determines its position in the environment.

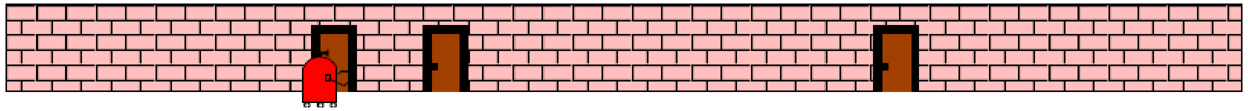
- Map-Based:
  - Assuming a map of the environment is known.

- Probabilistic:
  - The data coming from the robot sensors are affected by measurement errors, and therefore we can only compute the probability of the location of the robot in a given configuration.

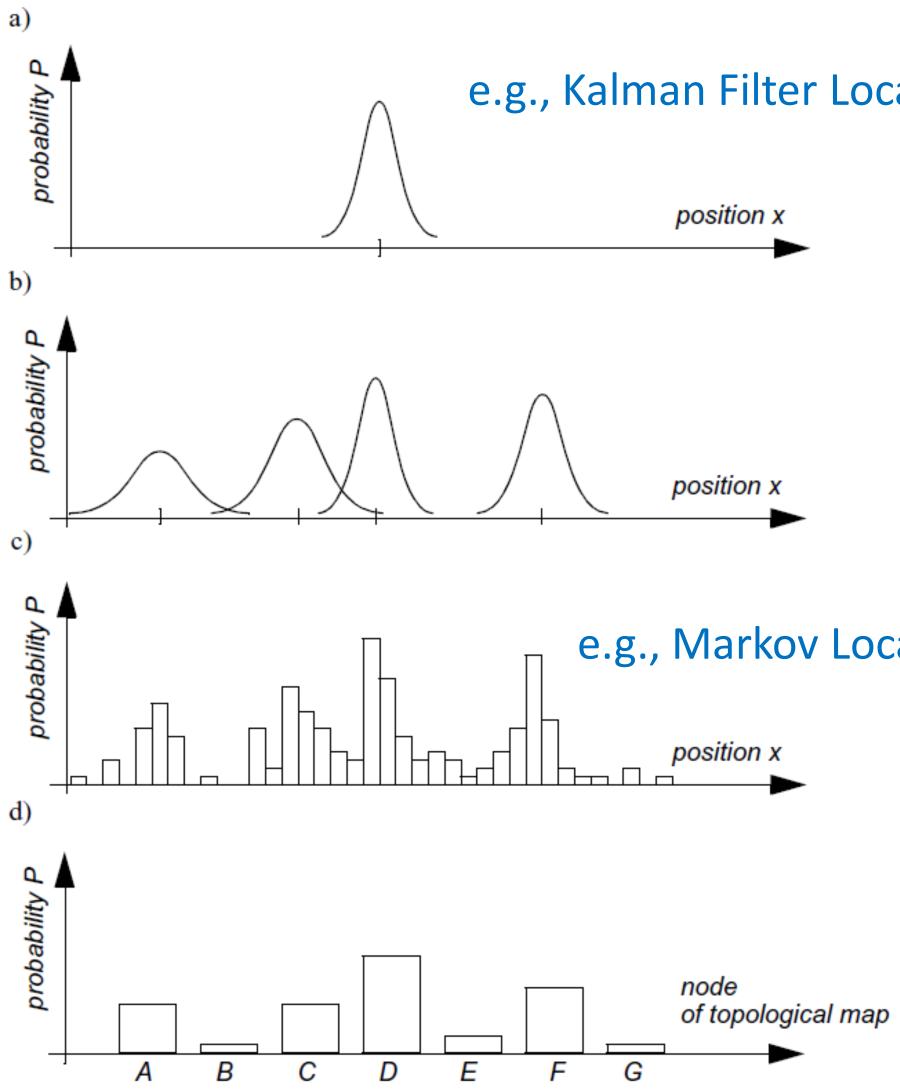
Probabilistic map-based localisation – Four ingredients

- 1. Belief representation
  - A representation of the robot’s belief regarding its position on the map
- 2. Probability theory
  - Theorem of total probability
  - Bayes rule
- 3. Motion model
  - Odometry model
- 4. Sensing model
  - Measurement model

1. Belief representation



- a) Continuous map with single-hypothesis belief
- b) Continuous map with multiple-hypothesis belief
- c) Discretised grid map with probability values for all possible robot positions
- d) Discretised topological map with probability values for all possible nodes



e.g., Kalman Filter Localisation

e.g., Markov Localisation

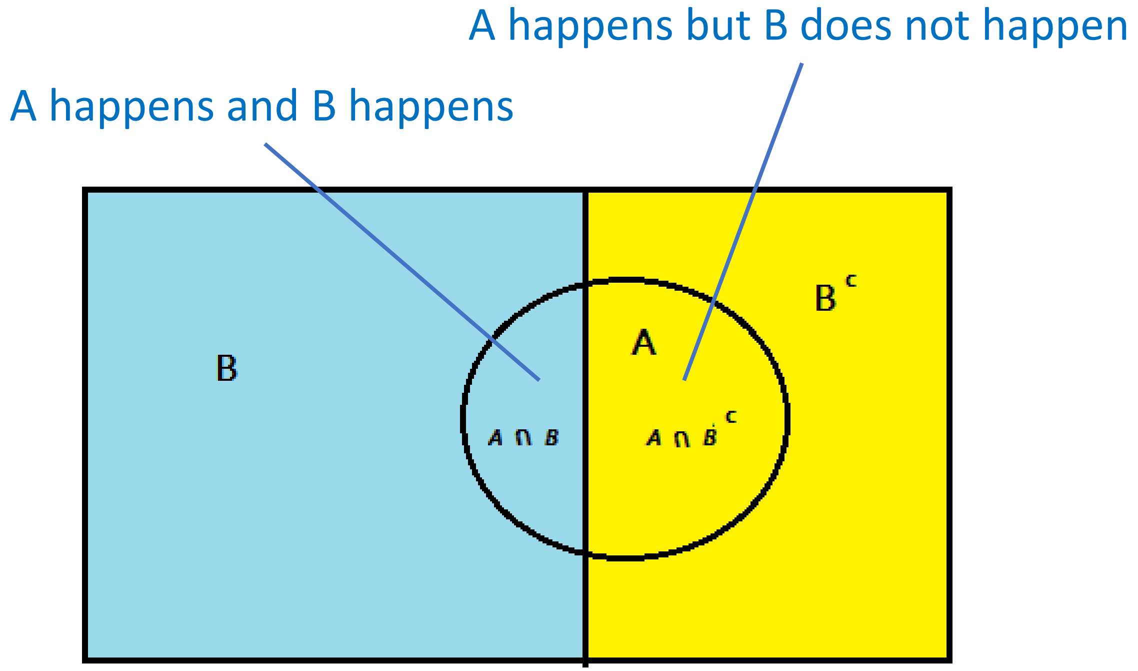
2. Probability theory

- *2.1 Theorem of total probability*

  - For discrete probabilities
    - $p(x) = \sum_y p(x|y)p(y)$

  - For continuous probabilities
    - $p(x) = \int_y \, p(x|y)p(y)dy$

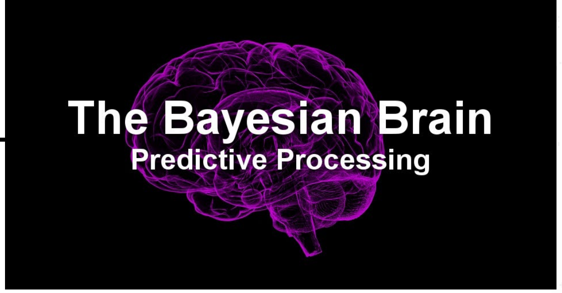
  - Here $p(x|y) = \frac{p(x,y)}{p(y)}$ is called *conditional probability*



2. Probability theory

- *2.2 Bayes rule*

$$\underbrace{p(x|y)}_{\text{posterior}} = \frac{\overbrace{p(y|x)}^{\text{likelihood}}\,\overbrace{p(x)}^{\text{prior}}}{p(y)}$$



Thomas Bayes
(1701 - 1761)

- Case study (**data made up**)
  - Given the following statistics, what is the probability that a person had infected COVID-19 if the person had a dry cough symptom?
  - 1. 0.2% of people worldwide could infect COVID-19.
  - 2. If a person had infected COVID-19, the possibility that the person had a dry cough symptom is 80%.
  - 3. If a person had not infected COVID-19, the possibility that the person had a dry cough symptom is 8.3%.

$$p(x) = 0.2\%$$

$$p(y|x) = 80\%$$

$$\begin{aligned} p(y) &= p(y|x)p(x) + p(y|{\sim}x)p({\sim}x) \\ &= 80\% \times 0.2\% + 8.3\% \times 99.8\% \\ &= 8.4434\% \end{aligned}$$

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \frac{80\% \times 0.2\%}{8.4434\%} = 1.89\%$$

2. Probability theory

- *2.2 Bayes rule*

posterior likelihood prior

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

What if we know that the person has had close contact with a confirmed infection case?

- Case study (**data made up**)
  - Given the following statistics, what is the probability that a person had infected COVID-19 if the person had a dry cough symptom?
  - 1. ~~0.2~~ 50% of people ~~worldwide~~ (with close contact of confirmed infection cases) could infect COVID-19.
  - 2. If a person had infected COVID-19, the possibility that the person had a dry cough symptom is 80%.
  - 3. If a person had not infected COVID-19, the possibility that the person had a dry cough symptom is 8.3%.

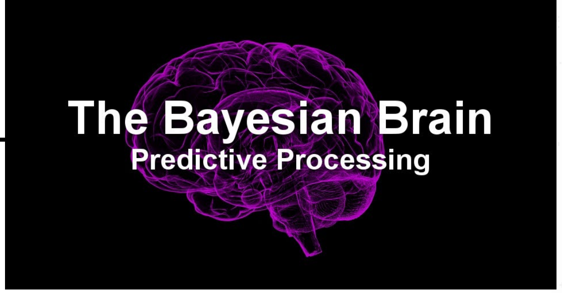
$$p(x) = \cancel{0.2}50\%$$

$$p(y|x) = 80\%$$

$$\begin{aligned} p(y) &= p(y|x)p(x) + p(y|{\sim}x)p({\sim}x) \\ &= 80\% \times \cancel{0.2}50\% + 8.3\% \times \cancel{99.8}50\% \\ &= \cancel{8.4434}44.15\% \end{aligned}$$

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \frac{80\% \times \cancel{0.2}50\%}{\cancel{8.4434}44.15\%} = \cancel{1.89}90.6\%$$

2. Probability theory



Thomas Bayes
(1701 - 1761)

- *2.2 Bayes rule*
  - $$\underbrace{p(x|y)}_{\text{posterior}} = \frac{\overbrace{p(y|x)}^{\text{likelihood}}\overbrace{p(x)}^{\text{prior}}}{p(y)} = \eta p(y|x)p(x)$$
- Case study (**data made up**)
  - Given the following statistics, what is the probability that a person had infected COVID-19 if the person had a dry cough symptom?
  - 1. 0.2% of people worldwide could infect COVID-19.
  - 2. If a person had infected COVID-19, the possibility that the person had a dry cough symptom is 80%.
  - 3. If a person had not infected COVID-19, the possibility that the person had a dry cough symptom is 8.3%.

$$p(x) = 0.2\%$$

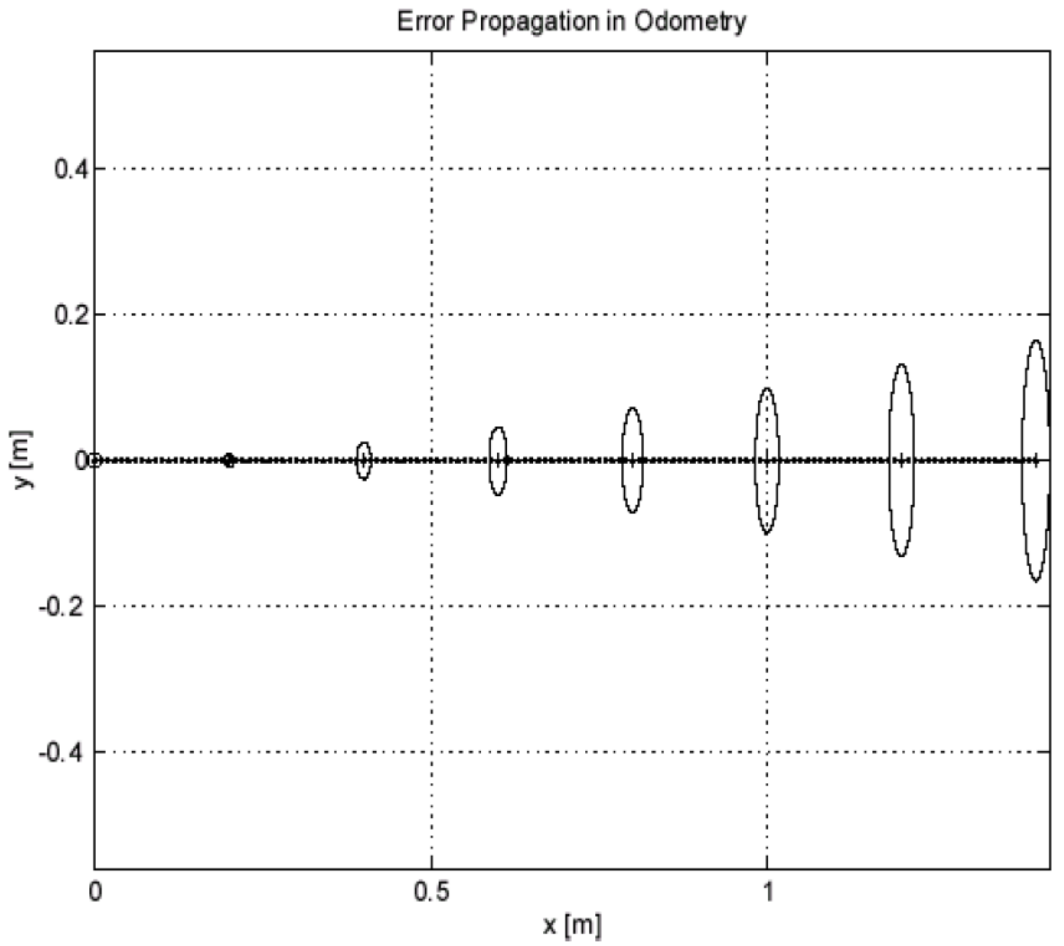
$$p(y|x) = 80\%$$

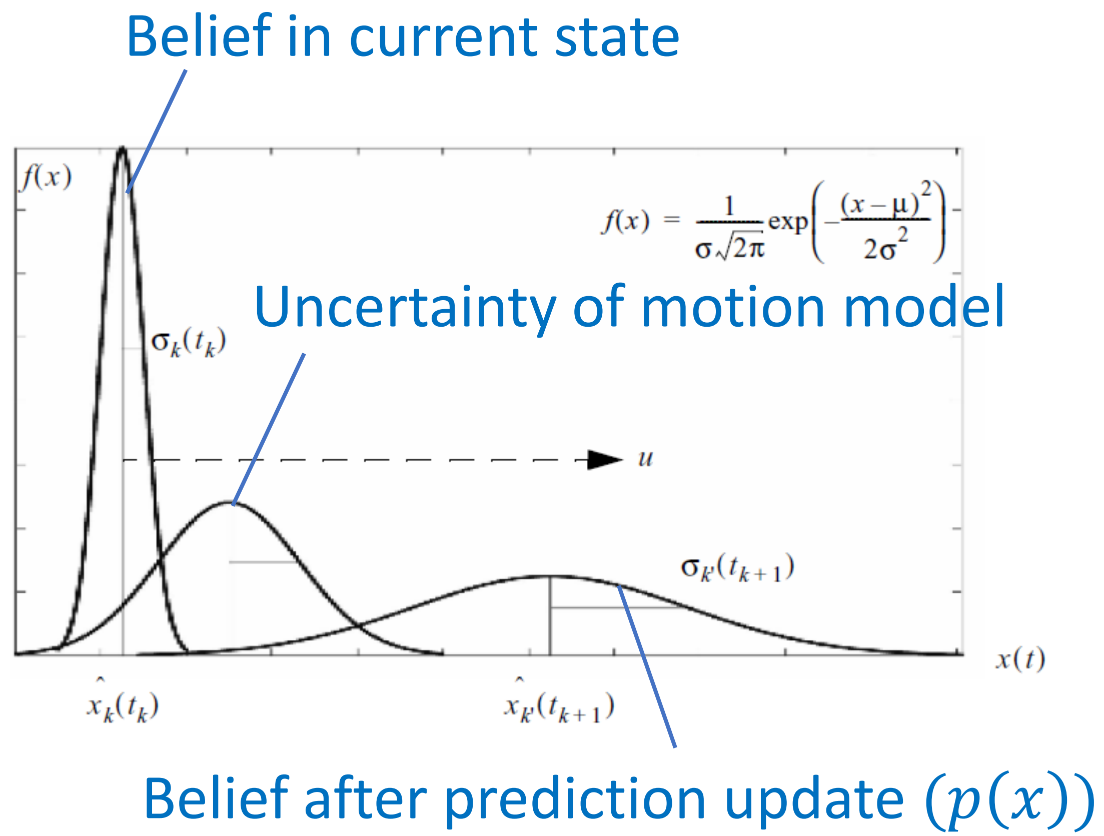
$$\begin{aligned} p(y) &= p(y|x)p(x) + p(y|{\sim}x)p({\sim}x) \\ &= 80\% \times 0.2\% + 8.3\% \times 99.8\% \\ &= 8.4434\% \end{aligned}$$

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \frac{80\% \times 0.2\%}{8.4434\%} = 1.89\%$$

3. Prediction (action) update

- Applying the *theorem of total probability* and using the motion model





Belief in current state
$f(x)$
$f(x) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$
Uncertainty of motion model
$\sigma_k(t_k)$
u
$\sigma_{k'}(t_{k+1})$
$x(t)$
$\hat{x}_k(t_k)$
$\hat{x}_{k'}(t_{k+1})$
Belief after prediction update $(p(x))$

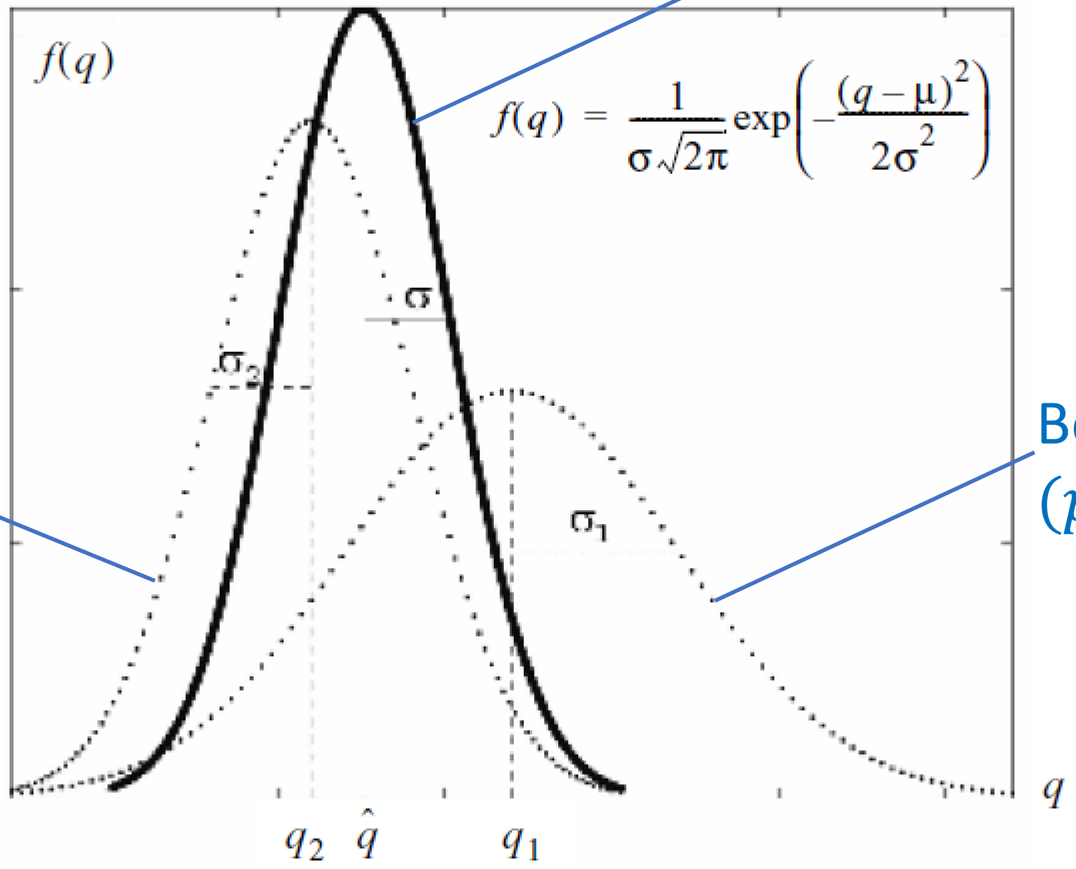
4. Measurement (perception) update

- Applying the *Bayes rule* and using the sensing model

Belief after measurement update

Belief generated by sensing model $(p(y|x))$

Belief after prediction update $(p(x))$



Markov Localisation

Markov localisation

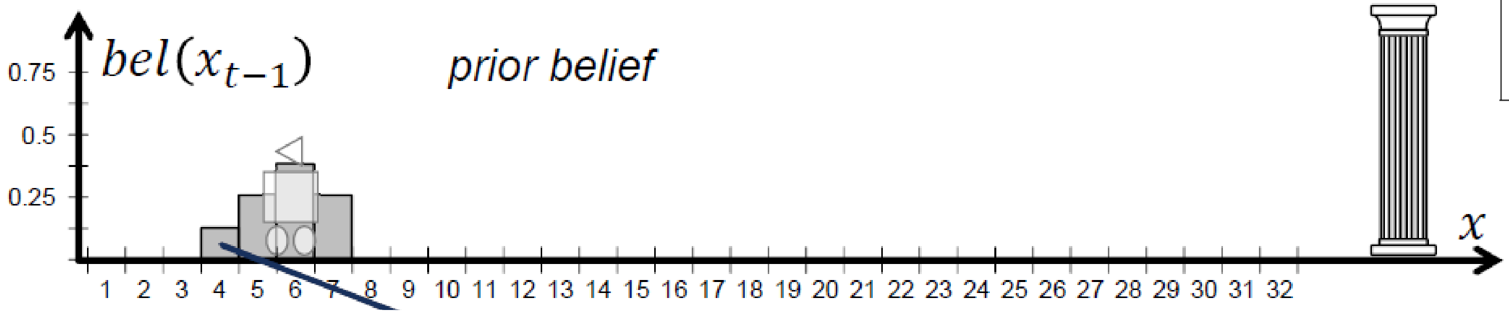
- Markov assumption
  - The output x_t is a function ONLY of the previous state x_{t-1} and its most recent actions (odometry) u_t and perception z_t
- A general algorithm
  - $\overline{bel}(\,)$ is belief after prediction update (Theorem of total probability)
  - $bel(\,)$ is belief after measurement update (Bayes' rule)
  - m is the information of the map

Andrey Markov (1856 - 1922)

```
1:  Algorithm Markov_localization($bel(x_{t-1}), u_t, z_t, m$):
2:      for all $x_t$ do
3:          $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}, m)\ bel(x_{t-1})\ dx$ (prediction update)
4:          $bel(x_t) = \eta\ p(z_t \mid x_t, m)\ \overline{bel}(x_t)$ (measurement update)
5:      endfor
6:      return $bel(x_t)$
```

Markov localisation – Case study

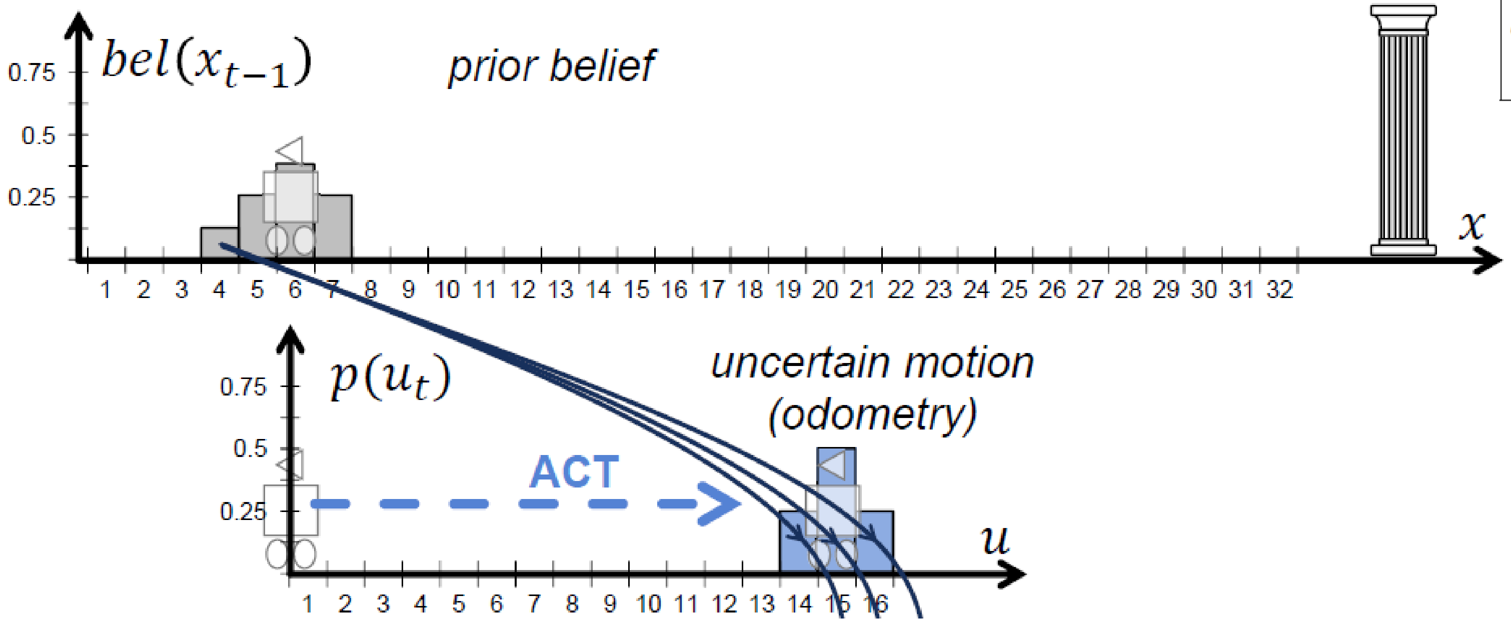
```
1:  Algorithm Markov_localization($bel(x_{t-1}), u_t, z_t, m$):
2:      for all $x_t$ do
3:          $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}, m)\, bel(x_{t-1})\, dx$   (prediction update)
4:          $bel(x_t) = \eta\, p(z_t \mid x_t, m)\, \overline{bel}(x_t)$   (measurement update)
5:      endfor
6:      return $bel(x_t)$
```



This is the belief of the position of the robot in state t-1.

Markov localisation – Case study

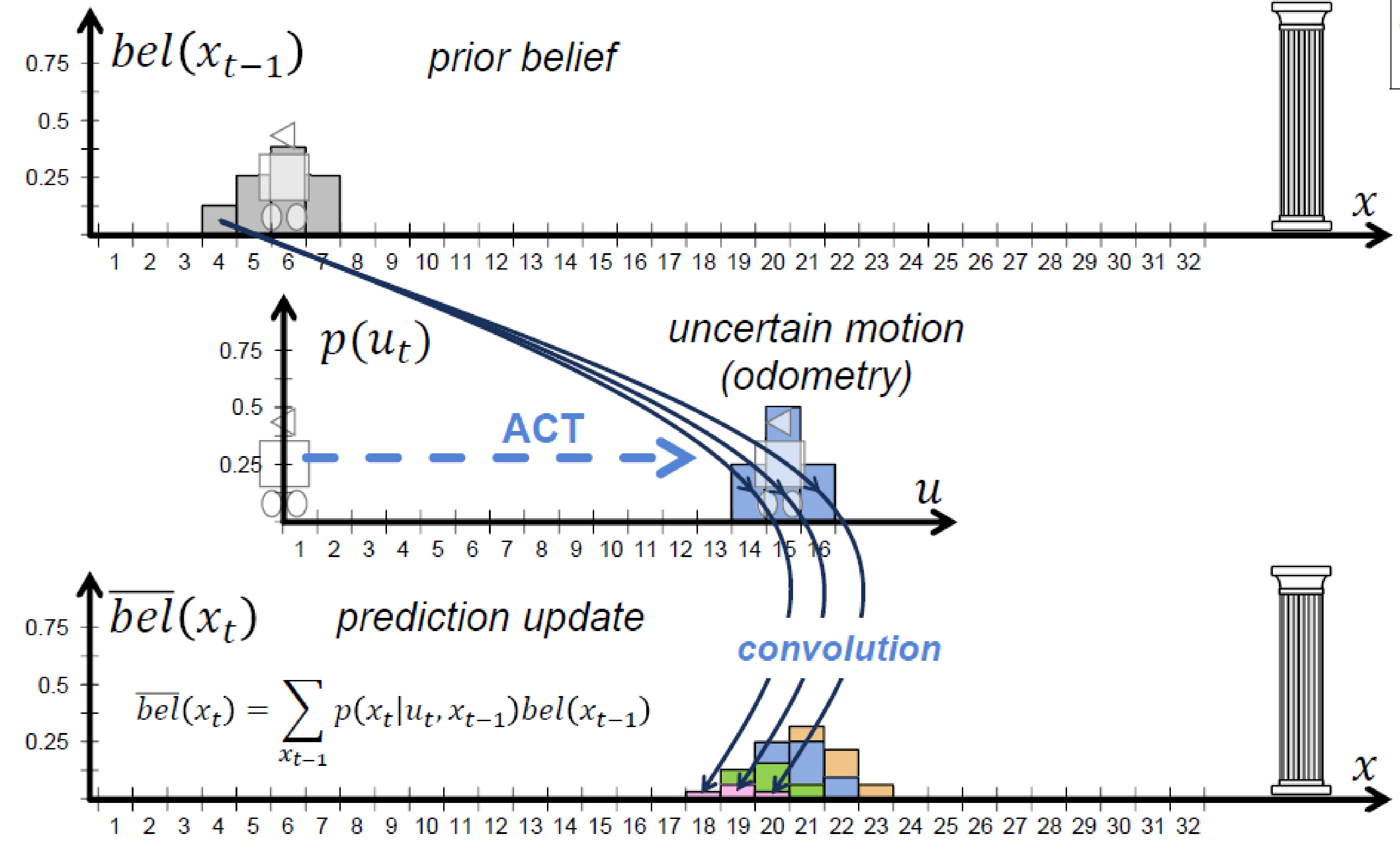
```
1:  Algorithm Markov_localization($bel(x_{t-1}), u_t, z_t, m$):
2:      for all $x_t$ do
3:          $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}, m)\ bel(x_{t-1})\ dx$   (prediction update)
4:          $bel(x_t) = \eta\ p(z_t \mid x_t, m)\ \overline{bel}(x_t)$   (measurement update)
5:      endfor
6:      return $bel(x_t)$
```



This is the belief of the position of the robot in state t-1.

The robot moves forward for 15 steps but there is uncertainty associated with the motion.

Markov localisation – Case study



```
1:  Algorithm Markov_localization($bel(x_{t-1}), u_t, z_t, m$):
2:      for all $x_t$ do
3:          $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}, m)\, bel(x_{t-1})\, dx$   (prediction update)
4:          $bel(x_t) = \eta\, p(z_t \mid x_t, m)\, \overline{bel}(x_t)$   (measurement update)
5:      endfor
6:      return $bel(x_t)$
```

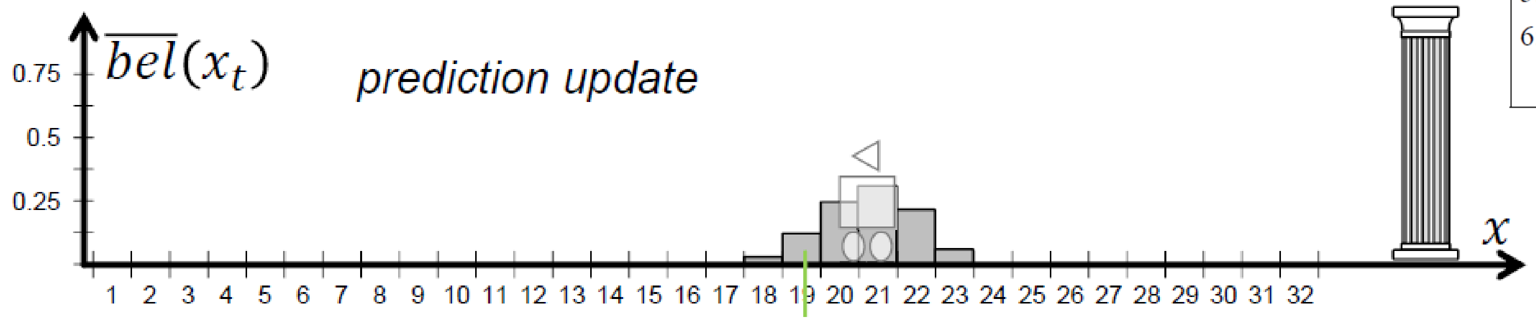
This is the belief of the position of the robot in state t-1.

The robot moves forward for 15 steps but there is uncertainty associated with the motion.

This is the belief after the prediction update (line 3 of the algorithm).

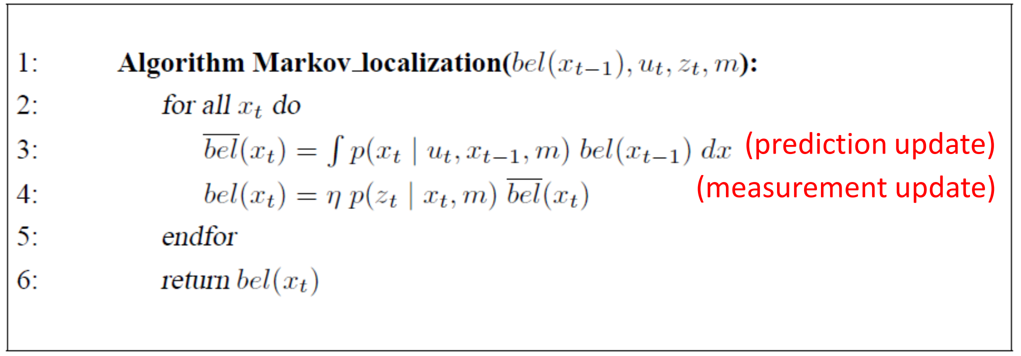
Markov localisation – Case study

```
1: Algorithm Markov_localization($bel(x_{t-1}), u_t, z_t, m$):
2:     for all $x_t$ do
3:         $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}, m)\, bel(x_{t-1})\, dx$   (prediction update)
4:         $bel(x_t) = \eta\, p(z_t \mid x_t, m)\, \overline{bel}(x_t)$   (measurement update)
5:     endfor
6:     return $bel(x_t)$
```

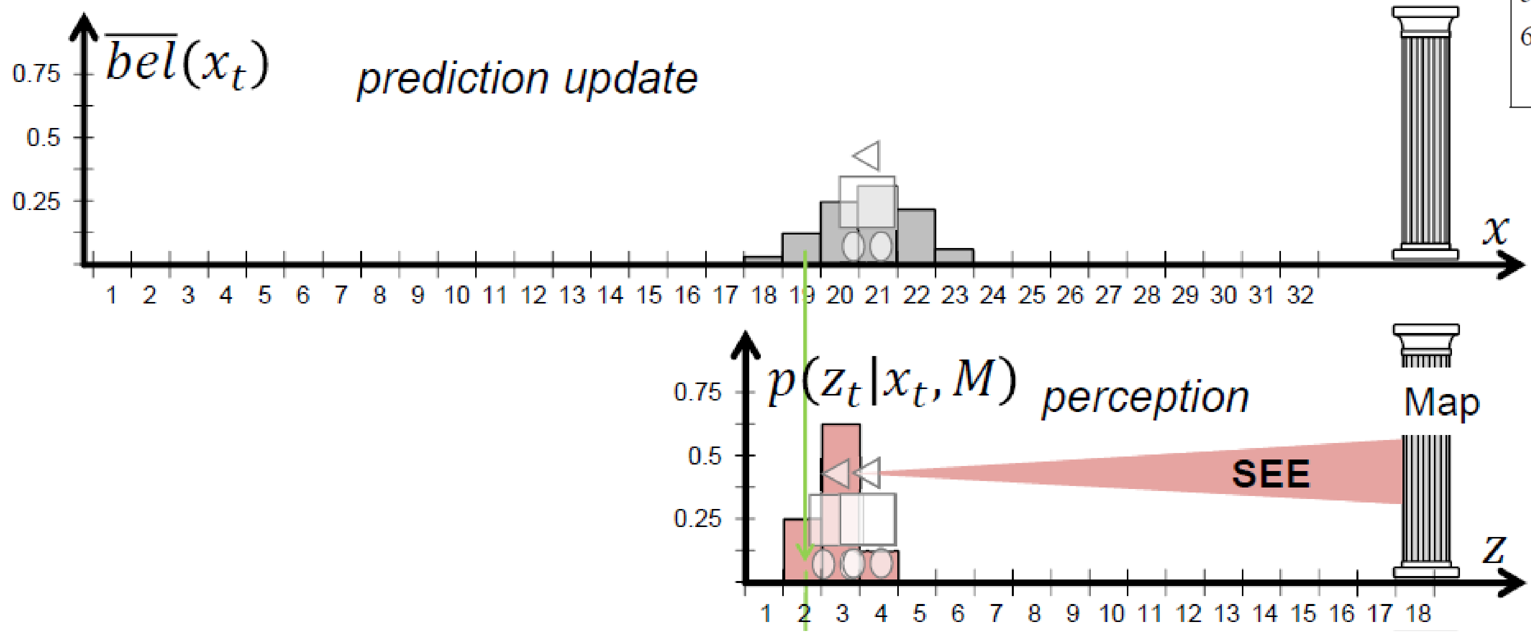


This is the belief after the prediction update (line 3 of the algorithm).

Markov localisation – Case study



```
1:  Algorithm Markov_localization($bel(x_{t-1}), u_t, z_t, m$):
2:      for all $x_t$ do
3:          $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}, m)\ bel(x_{t-1})\ dx$   (prediction update)
4:          $bel(x_t) = \eta\ p(z_t \mid x_t, m)\ \overline{bel}(x_t)$   (measurement update)
5:      endfor
6:      return $bel(x_t)$
```

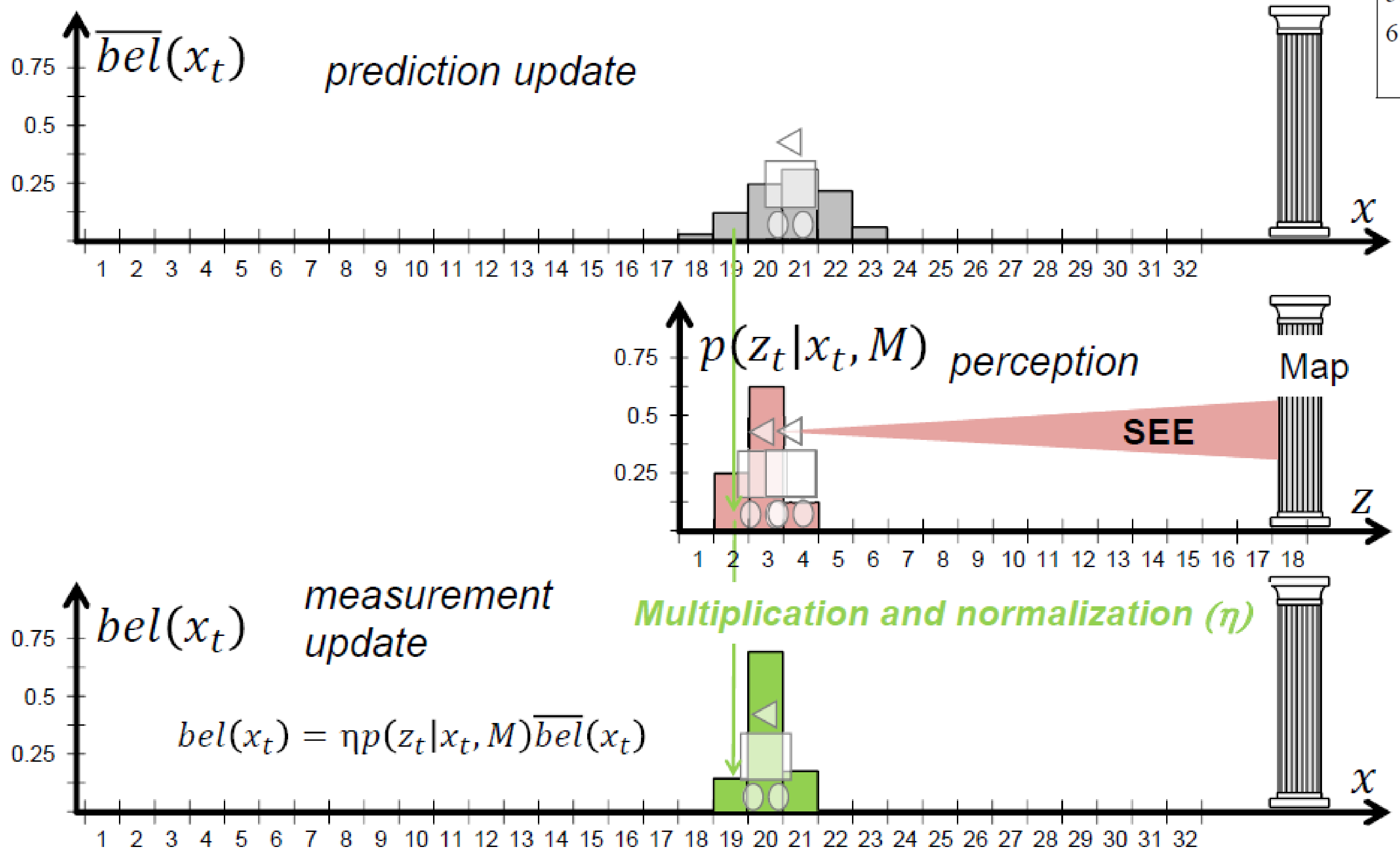


This is the belief after the prediction update (line 3 of the algorithm).

The robot detects the distance from the landmark but there is uncertainty associated with the measurement. Note – the map is known.

Markov localisation – Case study

```
1: Algorithm Markov_localization($bel(x_{t-1}), u_t, z_t, m$):
2:     for all $x_t$ do
3:         $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}, m)\ bel(x_{t-1})\ dx$   (prediction update)
4:         $bel(x_t) = \eta\ p(z_t \mid x_t, m)\ \overline{bel}(x_t)$   (measurement update)
5:     endfor
6:     return $bel(x_t)$
```



This is the belief after the prediction update (line 3 of the algorithm).

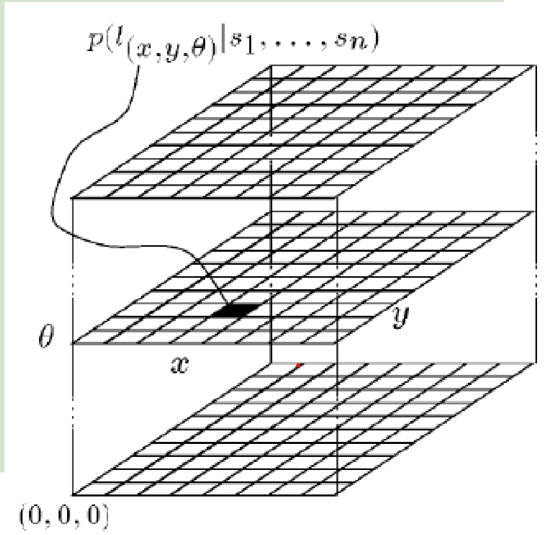
The robot detects the distance from the landmark but there is uncertainty associated with the measurement. Note – the map is known.

This is the belief of the position of the robot after the measurement update (line 4 of the algorithm).

Markov localisation - Summary

Advantages

- Localisation starting from any unknown position (global localisation)
- Recovers from ambiguous situation
- Can represent any arbitrary probability density function over the robot position

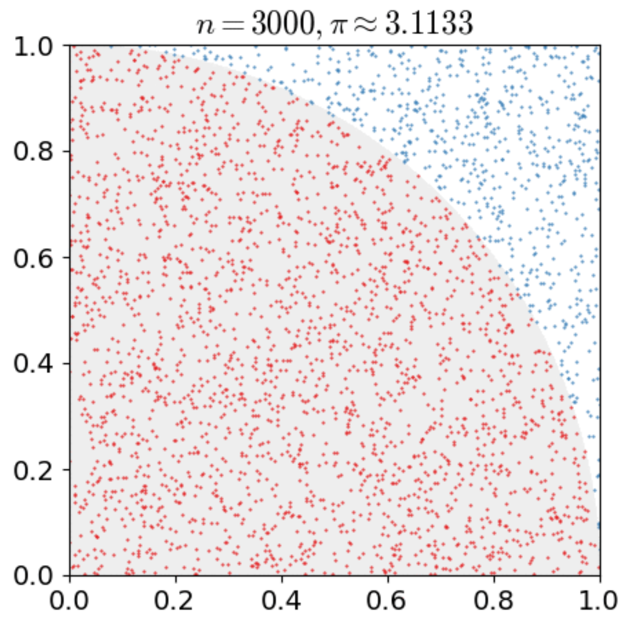


Disadvantages

- To update the probability of all positions within the whole state space at any time requires a discrete representation of the space (grid).
- The required memory and calculation power can thus become expensive if a fine grid is used.
- Example
  - 30m x 30m environment
  - Cell size of 0.1m x 0.1m x 1 deg
    - 300 x 300 x 360 = 32.4 million cells!

Particle Filter Localisation

Particle filter localisation (Monte Carlo localisation)



- Monte Carlo (MC) methods are a subset of computational algorithms that use the process of repeated random sampling to make numerical estimations of unknown parameters.
- There are a broad spectrum of Monte Carlo methods, but they all share the commonality that they rely on random number generation to solve deterministic problems.

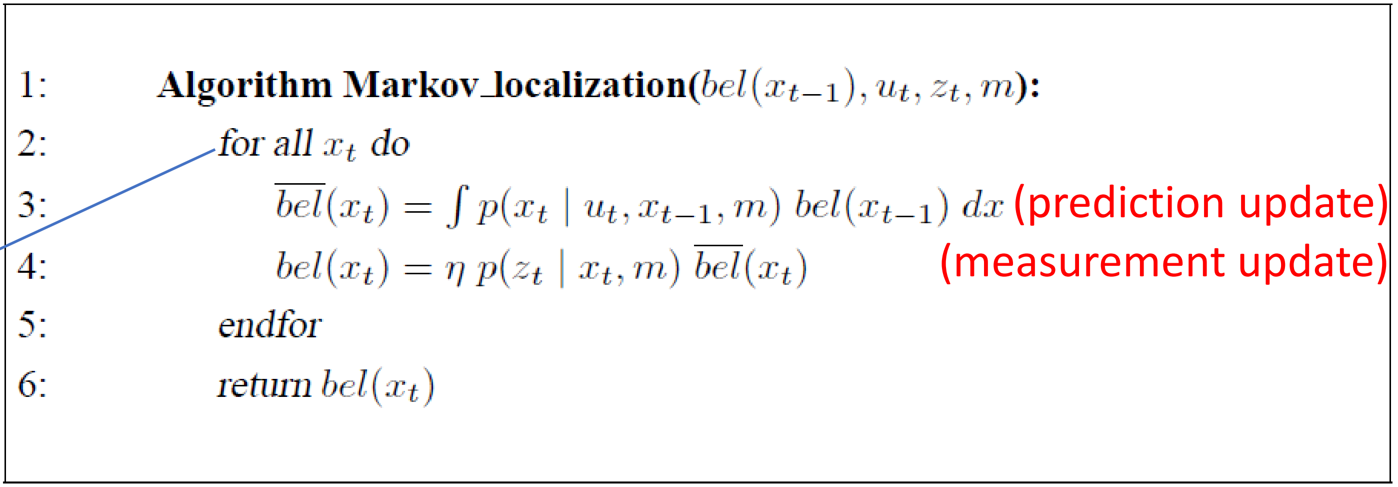
Monte Carlo Casino, Monaco

- *"Being secret, the work of John von Neumann and Stanislaw Ulam required a code name. A colleague of von Neumann and Ulam, Nicholas Metropolis, suggested using the name Monte Carlo, which refers to the Monte Carlo Casino in Monaco where Ulam's uncle would borrow money from relatives to gamble."*

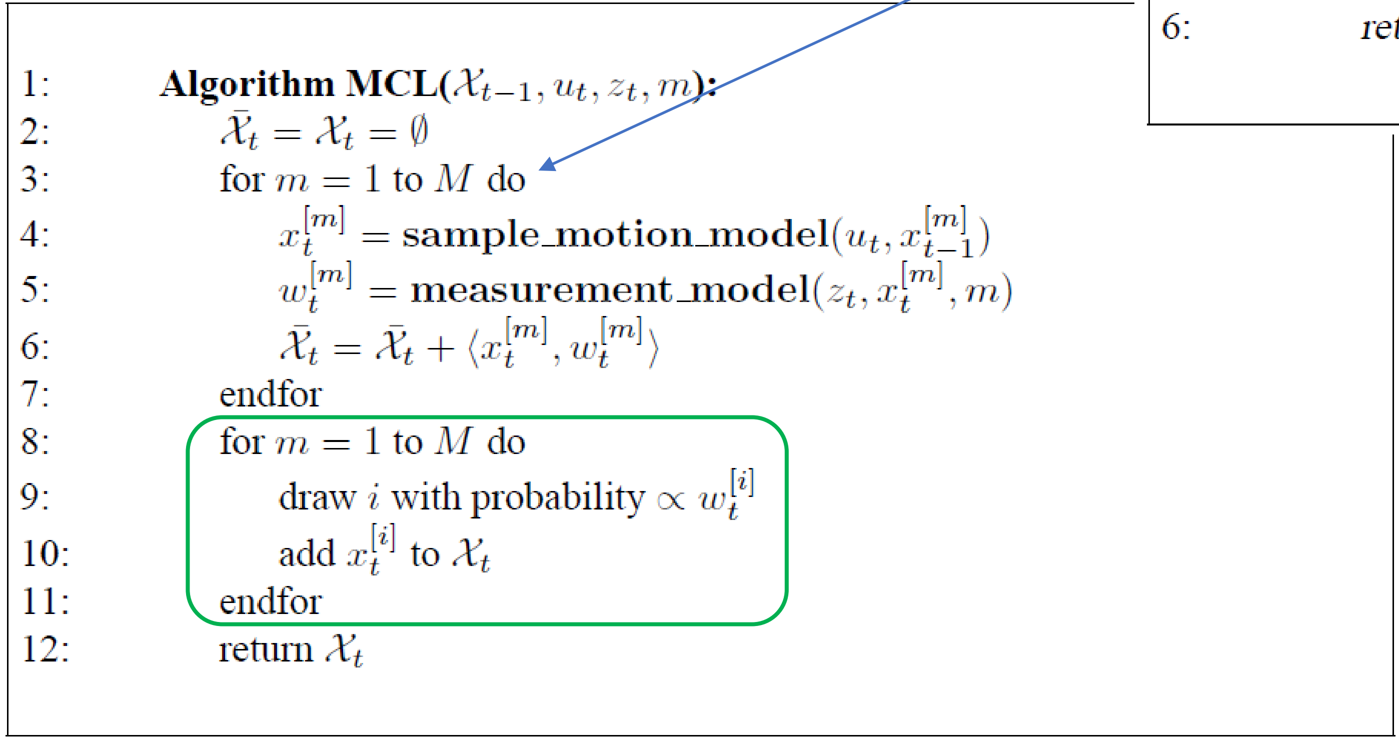
https://en.wikipedia.org/wiki/Monte_Carlo
https://en.wikipedia.org/wiki/Monte_Carlo_method
https://towardsdatascience.com/an-overview-of-monte-carlo-methods-675384eb1694

Particle filter localisation (Monte Carlo localisation)

An extension of Markov localisation with Monte Carlo sampling.



```
1:  Algorithm Markov_localization($bel(x_{t-1}), u_t, z_t, m$):
2:      for all $x_t$ do
3:          $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}, m)\, bel(x_{t-1})\, dx$ (prediction update)
4:          $bel(x_t) = \eta\, p(z_t \mid x_t, m)\, \overline{bel}(x_t)$ (measurement update)
5:      endfor
6:      return $bel(x_t)$
```



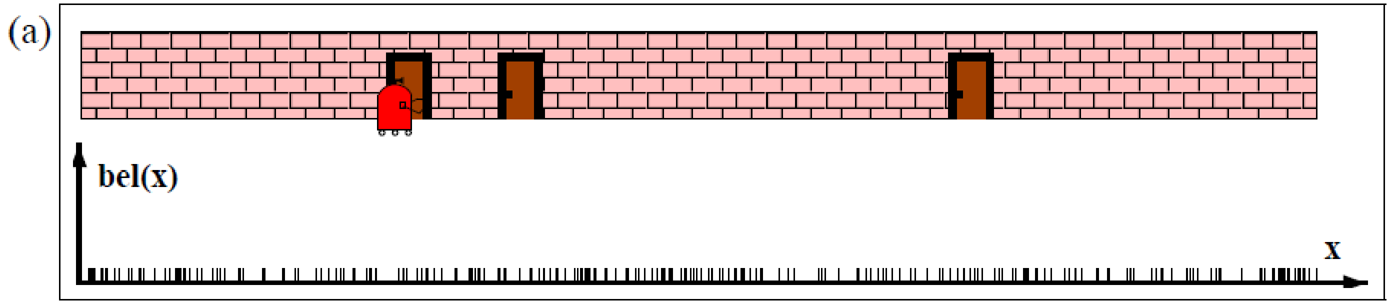
```
1:  Algorithm MCL($\mathcal{X}_{t-1}, u_t, z_t, m$):
2:      $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3:      for $m = 1$ to $M$ do
4:          $x_t^{[m]}$ = sample_motion_model($u_t, x_{t-1}^{[m]}$)
5:          $w_t^{[m]}$ = measurement_model($z_t, x_t^{[m]}, m$)
6:          $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7:      endfor
8:      for $m = 1$ to $M$ do
9:          draw $i$ with probability $\propto w_t^{[i]}$
10:         add $x_t^{[i]}$ to $\mathcal{X}_t$
11:     endfor
12:     return $\mathcal{X}_t$
```

Instead of calculating all the possible states, just sample a subset.

And an additional process: resampling.

Particle filter localisation (Monte Carlo localisation)

Initialisation - Randomly and uniformly sampled particles.
No motion at the beginning.



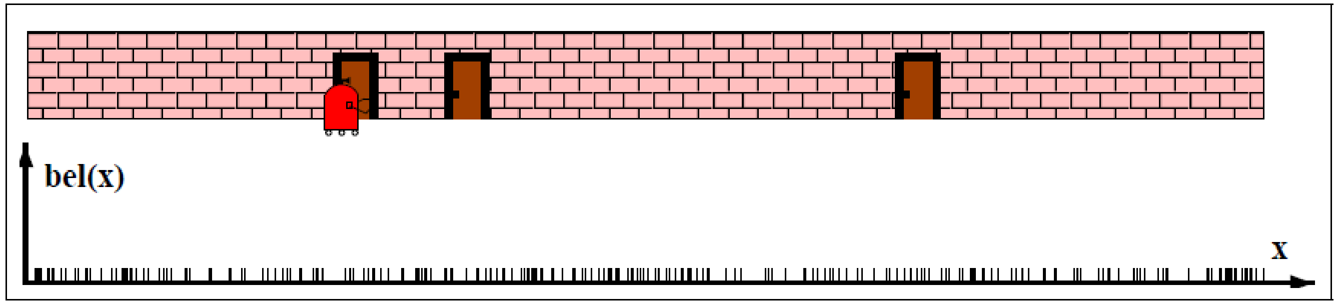
```
1:  Algorithm MCL($\mathcal{X}_{t-1}, u_t, z_t, m$):
2:      $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3:      for $m = 1$ to $M$ do
4:          $\boxed{x_t^{[m]} = \text{sample\_motion\_model}(u_t, x_{t-1}^{[m]})}$
5:          $w_t^{[m]} = \text{measurement\_model}(z_t, x_t^{[m]}, m)$
6:          $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7:      endfor
8:      for $m = 1$ to $M$ do
9:          draw $i$ with probability $\propto w_t^{[i]}$
10:         add $x_t^{[i]}$ to $\mathcal{X}_t$
11:     endfor
12:     return $\mathcal{X}_t$
```

Particle filter localisation (Monte Carlo localisation)

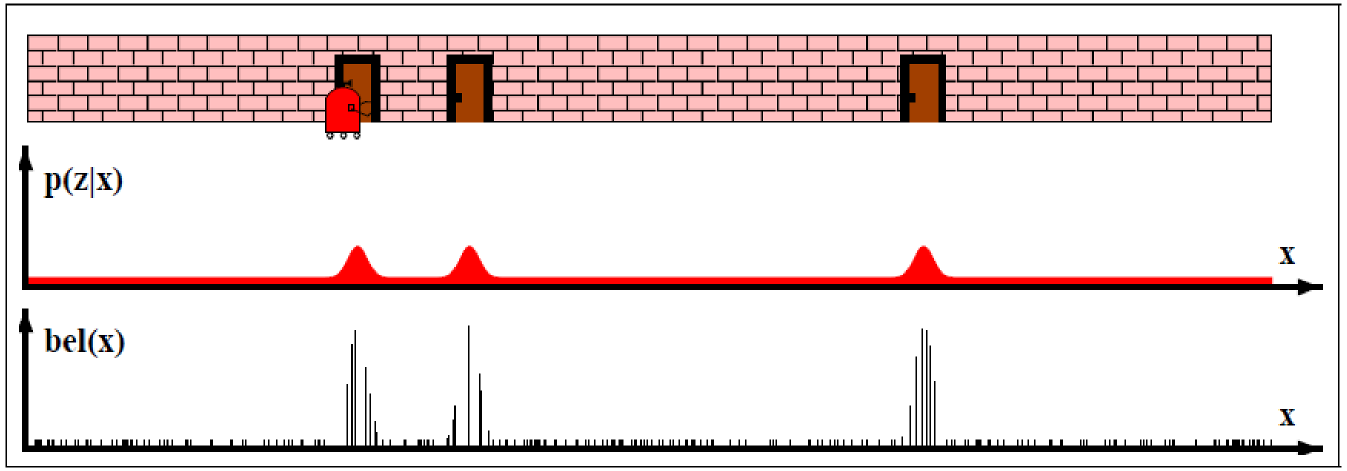
Measurement update - Assign weights to the particles based on the measurement model.

```
1:  Algorithm MCL($\mathcal{X}_{t-1}, u_t, z_t, m$):
2:      $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3:      for $m = 1$ to $M$ do
4:          $x_t^{[m]}$ = sample_motion_model($u_t, x_{t-1}^{[m]}$)
5:          $w_t^{[m]}$ = measurement_model($z_t, x_t^{[m]}, m$)
6:          $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7:      endfor
8:      for $m = 1$ to $M$ do
9:          draw $i$ with probability $\propto w_t^{[i]}$
10:         add $x_t^{[i]}$ to $\mathcal{X}_t$
11:     endfor
12:     return $\mathcal{X}_t$
```

(a)

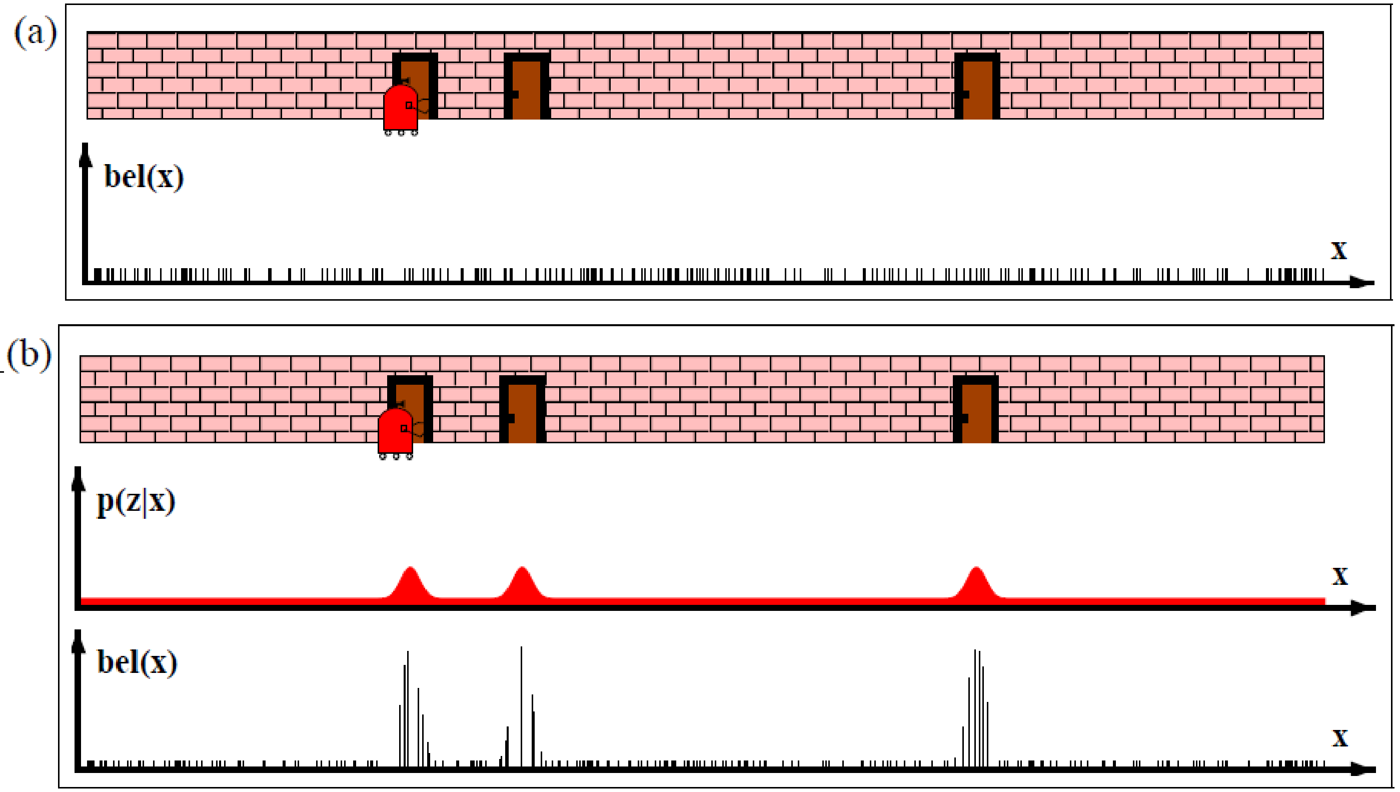


(b)



Particle filter localisation (Monte Carlo localisation)

Resample – more density around the particles with higher weights.



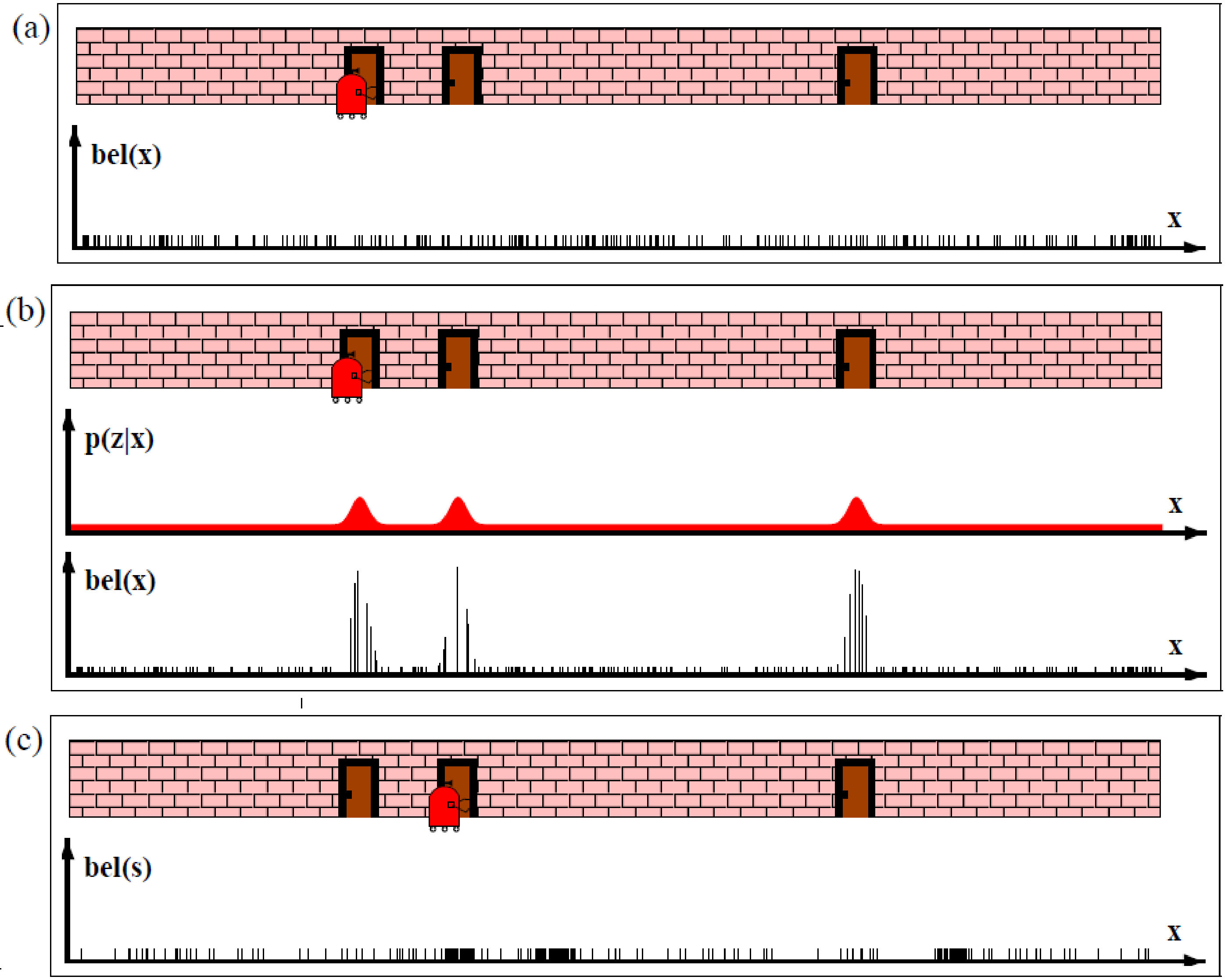
```
1:  Algorithm MCL($\mathcal{X}_{t-1}, u_t, z_t, m$):
2:      $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3:      for $m = 1$ to $M$ do
4:          $x_t^{[m]} =$ sample_motion_model$(u_t, x_{t-1}^{[m]})$
5:          $w_t^{[m]} =$ measurement_model$(z_t, x_t^{[m]}, m)$
6:          $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7:      endfor
8:      for $m = 1$ to $M$ do
9:          draw $i$ with probability $\propto w_t^{[i]}$
10:         add $x_t^{[i]}$ to $\mathcal{X}_t$
11:     endfor
12:     return $\mathcal{X}_t$
```

Particle filter localisation (Monte Carlo localisation)

Resample – more density around the particles with higher weights.

Prediction update – applying motion model to the particles.

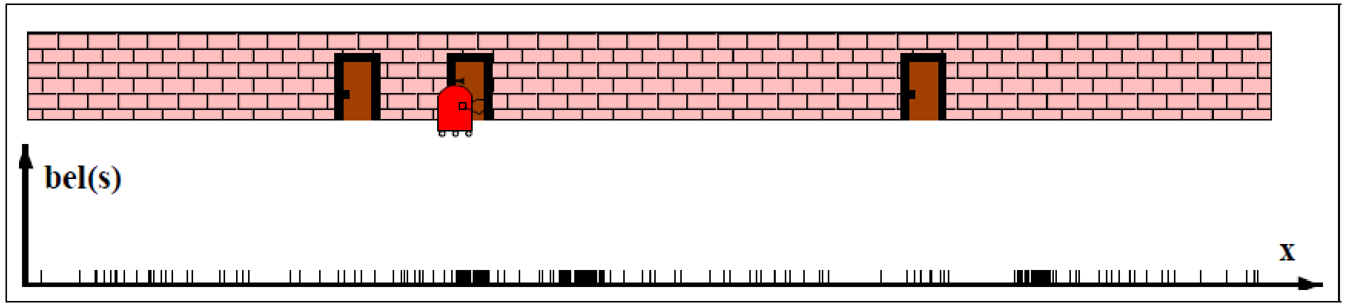
```
1:  Algorithm MCL($\mathcal{X}_{t-1}, u_t, z_t, m$):
2:      $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3:      for $m = 1$ to $M$ do
4:          $x_t^{[m]} = \textbf{sample\_motion\_model}(u_t, x_{t-1}^{[m]})$
5:          $w_t^{[m]} = \textbf{measurement\_model}(z_t, x_t^{[m]}, m)$
6:          $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7:      endfor
8:      for $m = 1$ to $M$ do
9:          draw $i$ with probability $\propto w_t^{[i]}$
10:         add $x_t^{[i]}$ to $\mathcal{X}_t$
11:     endfor
12:     return $\mathcal{X}_t$
```



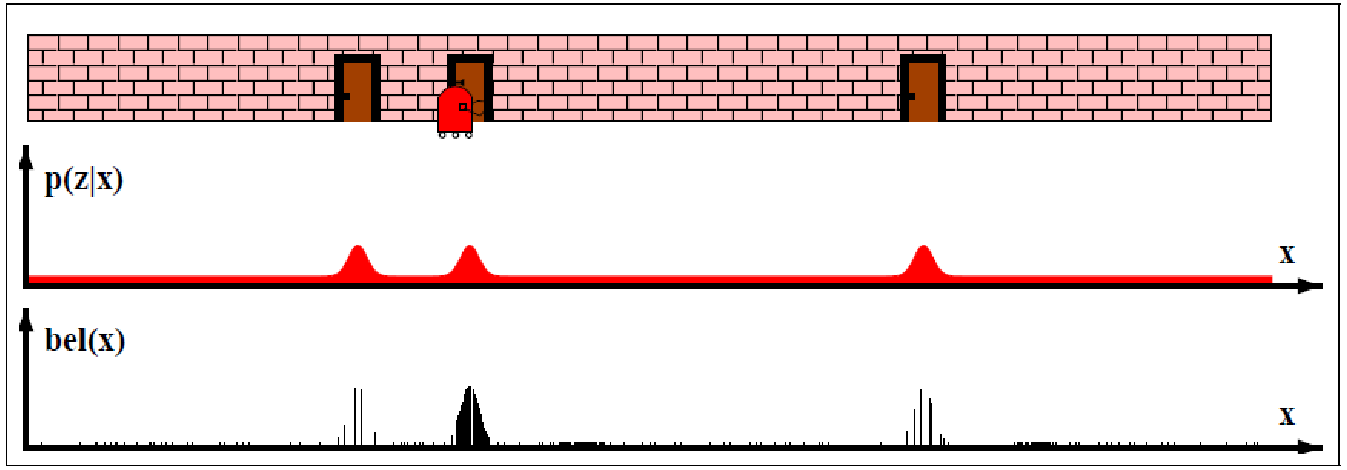
Particle filter localisation (Monte Carlo localisation)

Measurement update - Assign weights to the particles based on the measurement model.

(c)



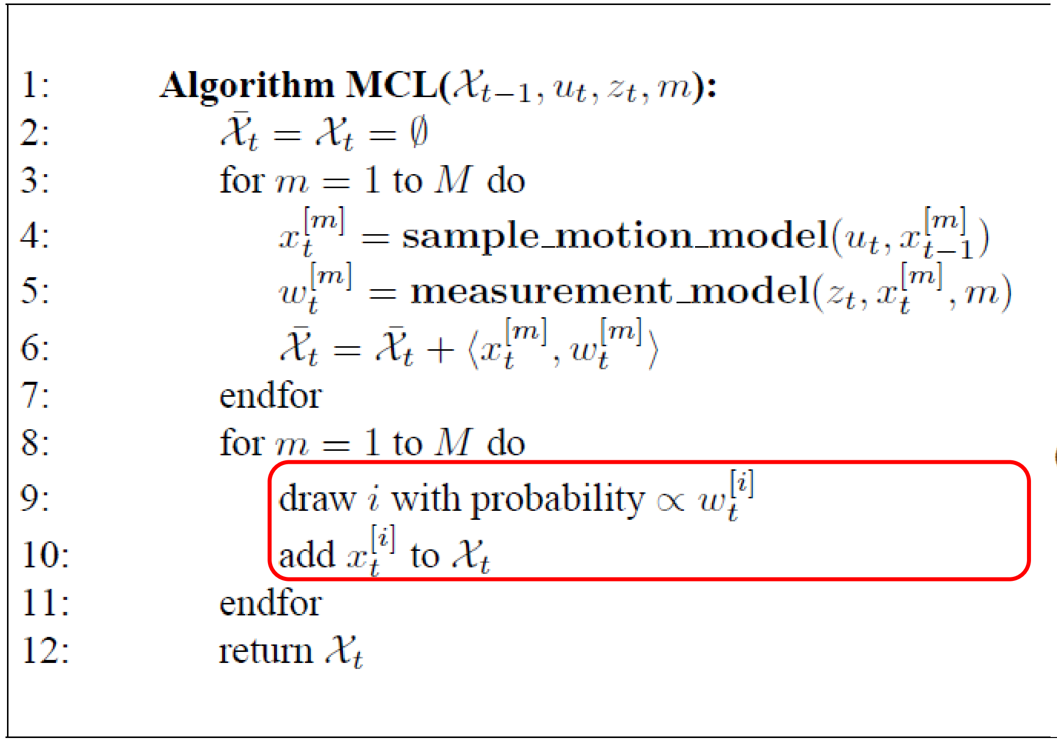
(d)



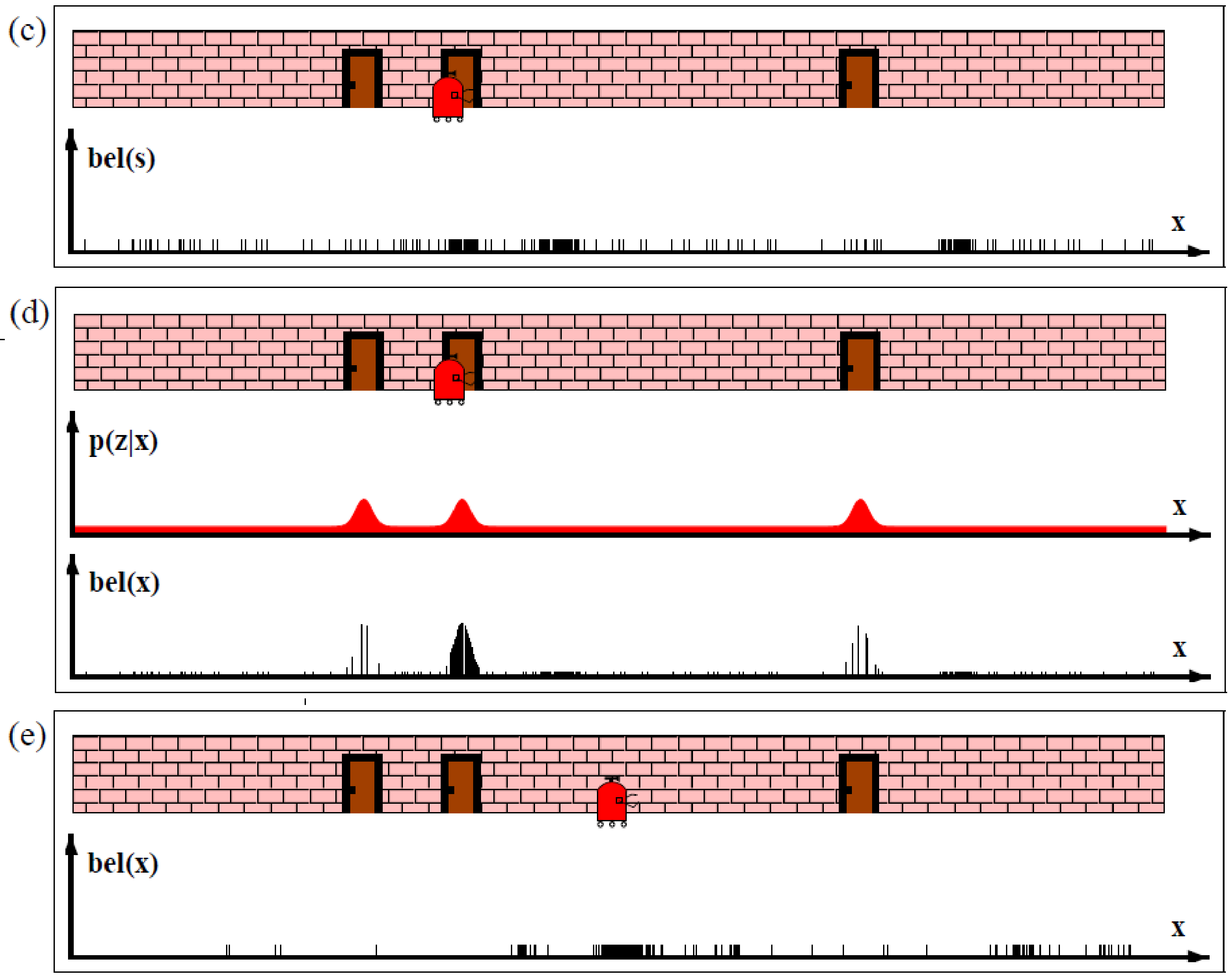
```
1:  Algorithm MCL($\mathcal{X}_{t-1}, u_t, z_t, m$):
2:      $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3:      for $m = 1$ to $M$ do
4:          $x_t^{[m]} = \text{sample\_motion\_model}(u_t, x_{t-1}^{[m]})$
5:          $w_t^{[m]} = \text{measurement\_model}(z_t, x_t^{[m]}, m)$
6:          $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7:      endfor
8:      for $m = 1$ to $M$ do
9:          draw $i$ with probability $\propto w_t^{[i]}$
10:         add $x_t^{[i]}$ to $\mathcal{X}_t$
11:     endfor
12:     return $\mathcal{X}_t$
```

Particle filter localisation (Monte Carlo localisation)

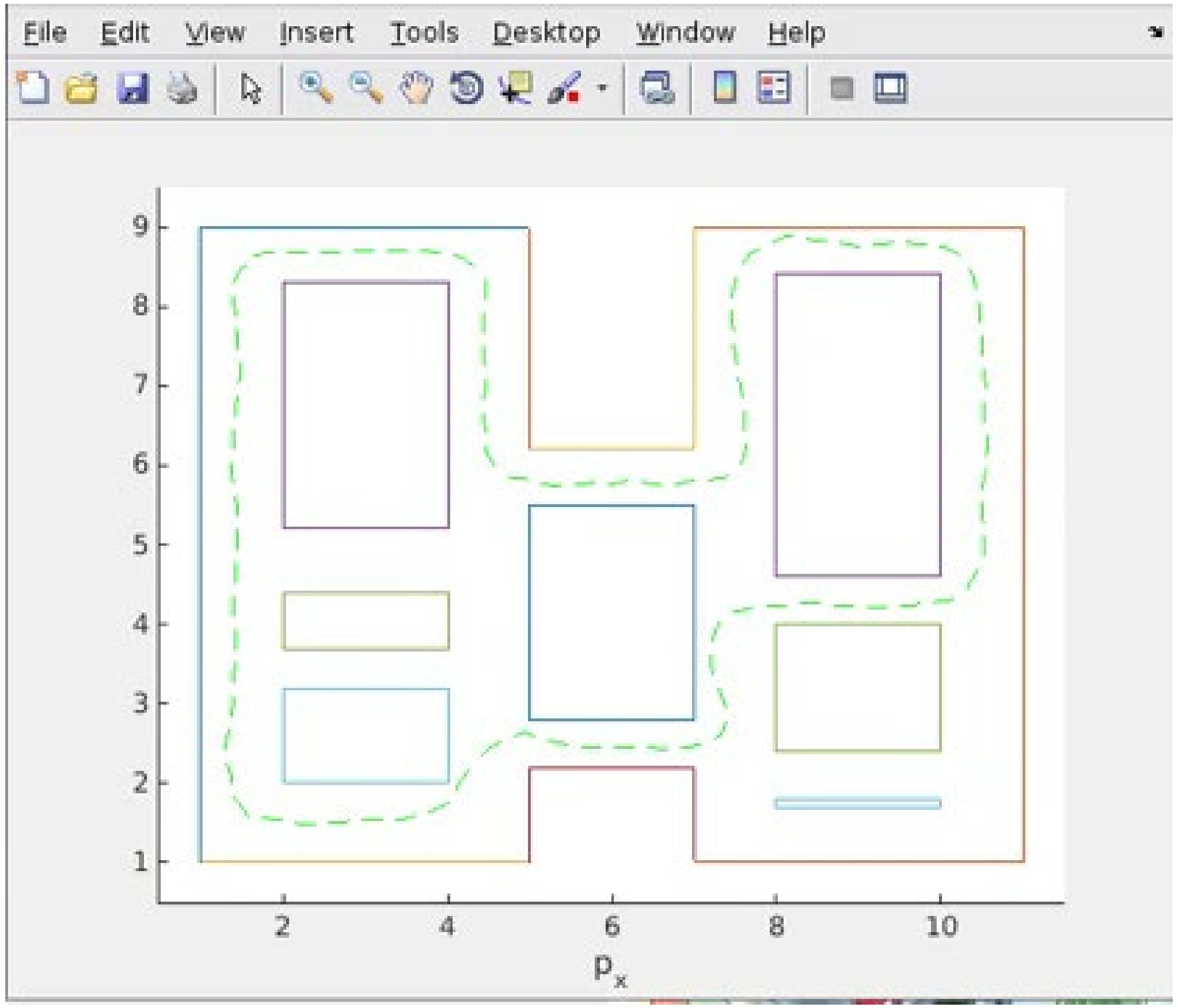
Resample – More density around the particles with higher weights.



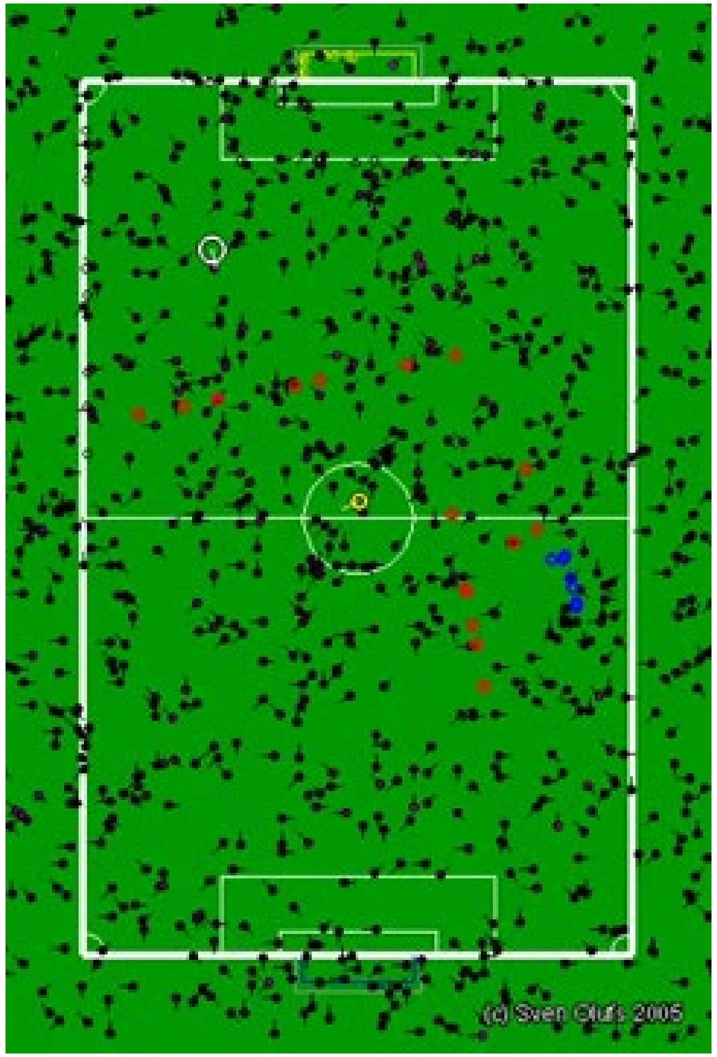
```
1:  Algorithm MCL($\mathcal{X}_{t-1}, u_t, z_t, m$):
2:      $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3:      for $m = 1$ to $M$ do
4:          $x_t^{[m]} = \text{sample\_motion\_model}(u_t, x_{t-1}^{[m]})$
5:          $w_t^{[m]} = \text{measurement\_model}(z_t, x_t^{[m]}, m)$
6:          $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7:      endfor
8:      for $m = 1$ to $M$ do
9:          draw $i$ with probability $\propto w_t^{[i]}$
10:         add $x_t^{[i]}$ to $\mathcal{X}_t$
11:     endfor
12:     return $\mathcal{X}_t$
```



Particle filter localisation – Example I: Tracking



Particle filter localisation – Example II: Soccer robot



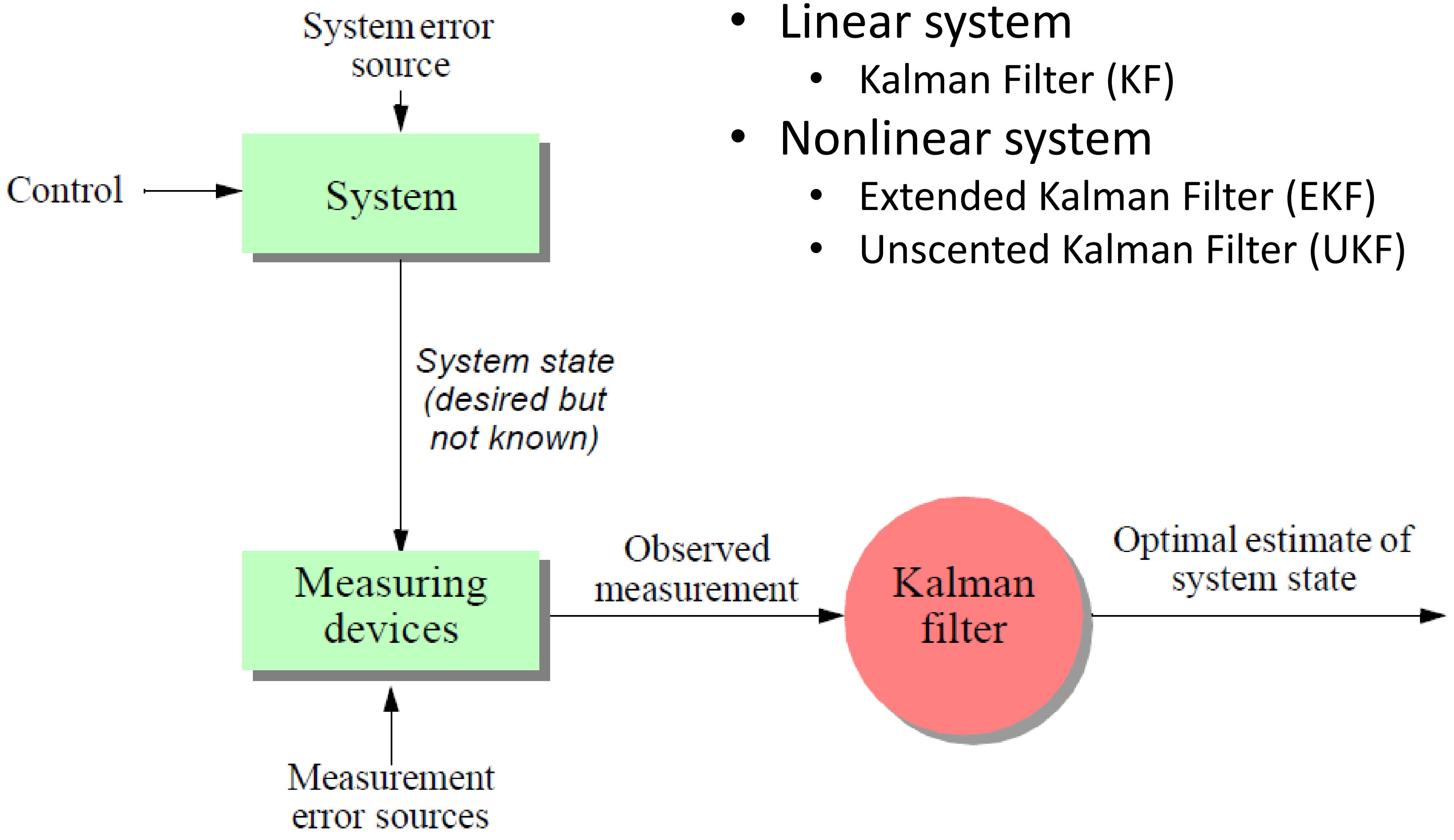
https://www.youtube.com/watch?v=fuMoj_YesHc

Particle filter localisation – Summary

Advantages	Disadvantages
• Localisation starting from any unknown position (global localisation) • Recovers from ambiguous situation • Can represent any arbitrary probability density function over the robot position • Less computational burden compared to Markov Localisation	• Complete nature of Markov Localisation is violated by sampling approaches • Still requires large computation resources

Kalman Filter Localisation

Kalman filter localisation

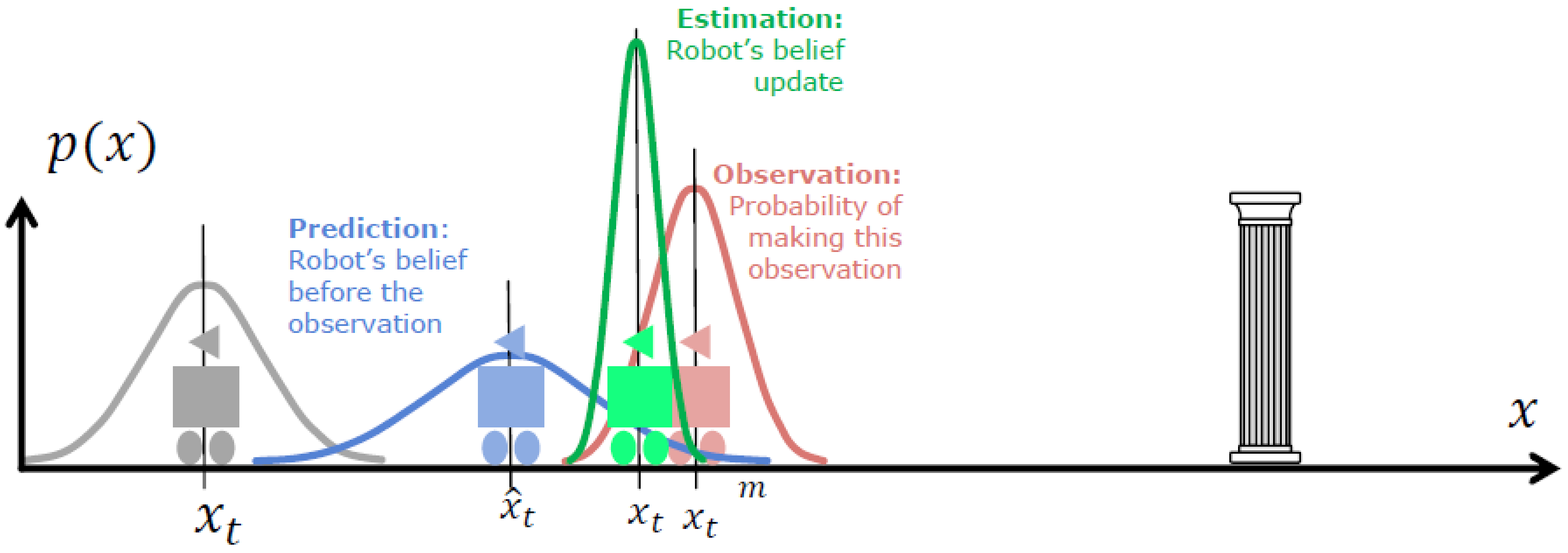

System error source
Control
System
System state (desired but not known)
Measuring devices
Observed measurement
Kalman filter
Optimal estimate of system state
Measurement error sources

- Linear system
  - Kalman Filter (KF)
- Nonlinear system
  - Extended Kalman Filter (EKF)
  - Unscented Kalman Filter (UKF)

Rudolf Emil Kalman
(1930 - 2016)

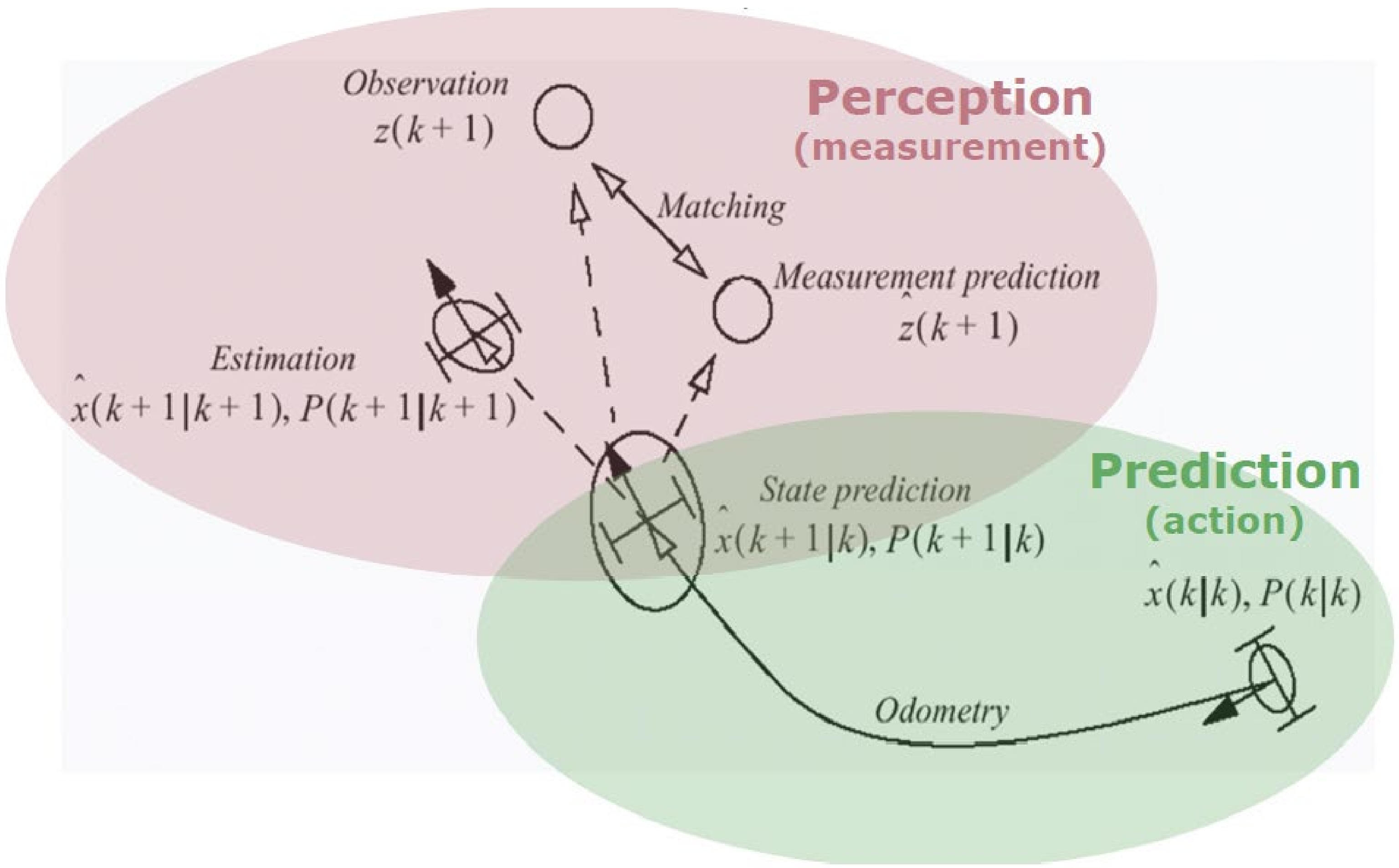
Kalman filter localisation

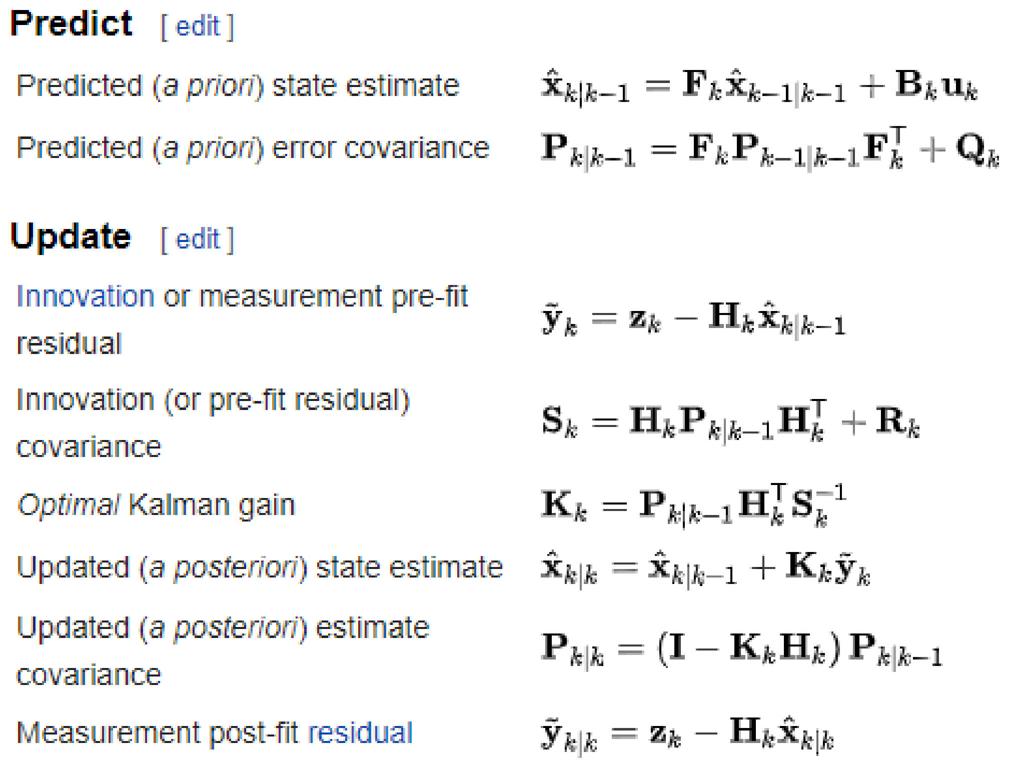
- Prediction update
  - Applying the *theorem of total probability* and using previous estimate and odometry
- Measurement update
  - Observation with on-board sensors
  - Measurement prediction based on prediction and map
  - Matching of observation and map
  - Estimation: position update



Estimation: Robot's belief update
Observation: Probability of making this observation
Prediction: Robot's belief before the observation
$p(x)$
x
x_t
$\hat{x}_t$
x_t
x_t
m

Kalman filter localisation





Predict [edit]

Predicted (*a priori*) state estimate	$\hat{\mathbf{x}}_{k\mid k-1} = \mathbf{F}_k \hat{\mathbf{x}}_{k-1\mid k-1} + \mathbf{B}_k \mathbf{u}_k$
Predicted (*a priori*) error covariance	$\mathbf{P}_{k\mid k-1} = \mathbf{F}_k \mathbf{P}_{k-1\mid k-1} \mathbf{F}_k^\mathsf{T} + \mathbf{Q}_k$

Update [edit]

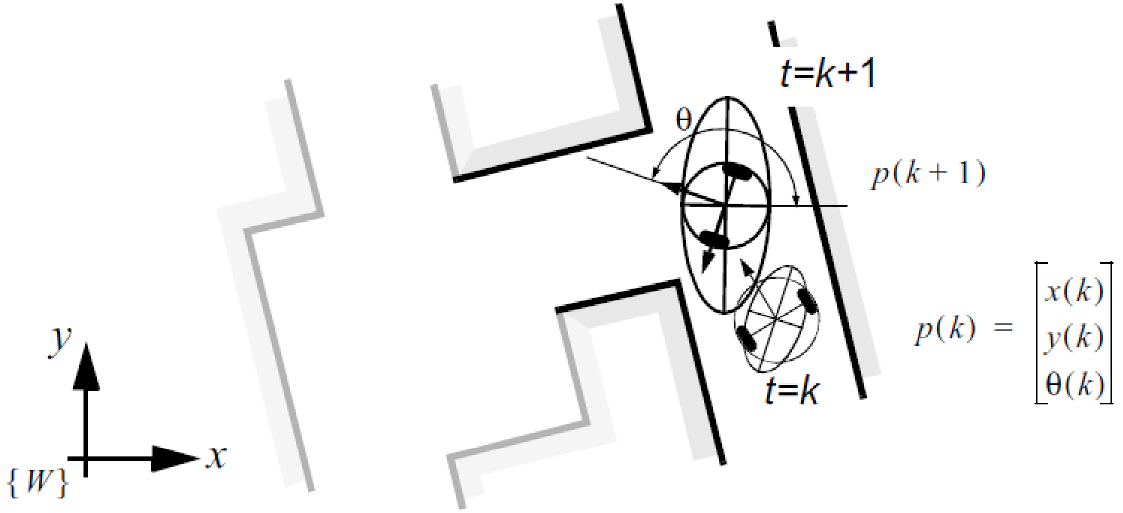
Innovation or measurement pre-fit residual	$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k\mid k-1}$
Innovation (or pre-fit residual) covariance	$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k\mid k-1} \mathbf{H}_k^\mathsf{T} + \mathbf{R}_k$
Optimal Kalman gain	$\mathbf{K}_k = \mathbf{P}_{k\mid k-1} \mathbf{H}_k^\mathsf{T} \mathbf{S}_k^{-1}$
Updated (*a posteriori*) state estimate	$\hat{\mathbf{x}}_{k\mid k} = \hat{\mathbf{x}}_{k\mid k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k$
Updated (*a posteriori*) estimate covariance	$\mathbf{P}_{k\mid k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k\mid k-1}$
Measurement post-fit residual	$\tilde{\mathbf{y}}_{k\mid k} = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k\mid k}$

Kalman filter localisation – Case study

- 1. Position prediction – Based on odometry model

$$\hat{p}(k+1|k) = \hat{p}(k|k) + u(k) = \hat{p}(k|k) + \begin{bmatrix} \frac{\Delta s_r + \Delta s_l}{2}\cos\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r + \Delta s_l}{2}\sin\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r - \Delta s_l}{b} \end{bmatrix}$$

$$\Sigma_p(k+1|k) = \nabla_p f \cdot \Sigma_p(k|k) \cdot \nabla_p f^T + \nabla_u f \cdot \Sigma_u(k) \cdot \nabla_u f^T$$

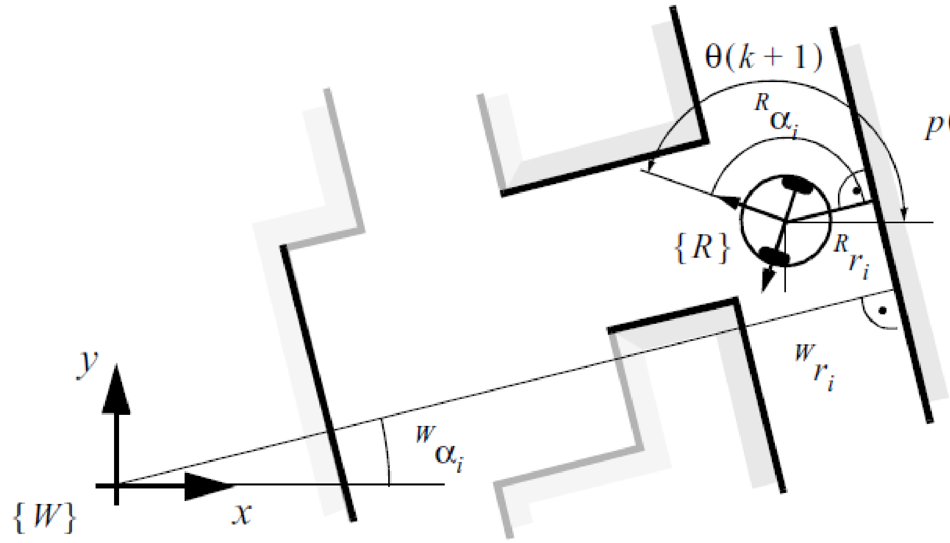

t=k+1
θ
$p(k+1)$
$p(k) = \begin{bmatrix} x(k) \\ y(k) \\ \theta(k) \end{bmatrix}$
t=k
y
x
{W}

Kalman filter localisation – Case study

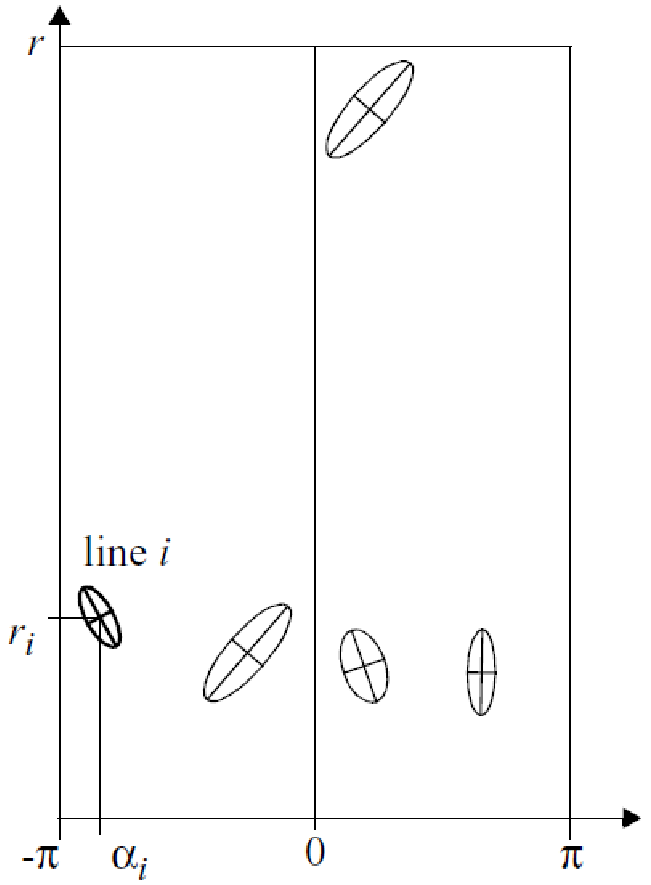
- 2. Measurement prediction – Based on map and predicted position

$$\hat{z}_i(k+1) = {}^R\begin{bmatrix}\alpha_{t,i} \\ r_{t,i}\end{bmatrix} = h_i(z_{t,i}, \hat{p}(k+1|k))$$

$$= \begin{bmatrix} {}^W\alpha_{t,i} - {}^W\hat{\theta}(k+1|k) \\ {}^W r_{t,i} - ({}^W\hat{x}(k+1|k)\cos({}^W\alpha_{t,i}) + {}^W\hat{y}(k+1|k)\sin({}^W\alpha_{t,i})) \end{bmatrix}$$



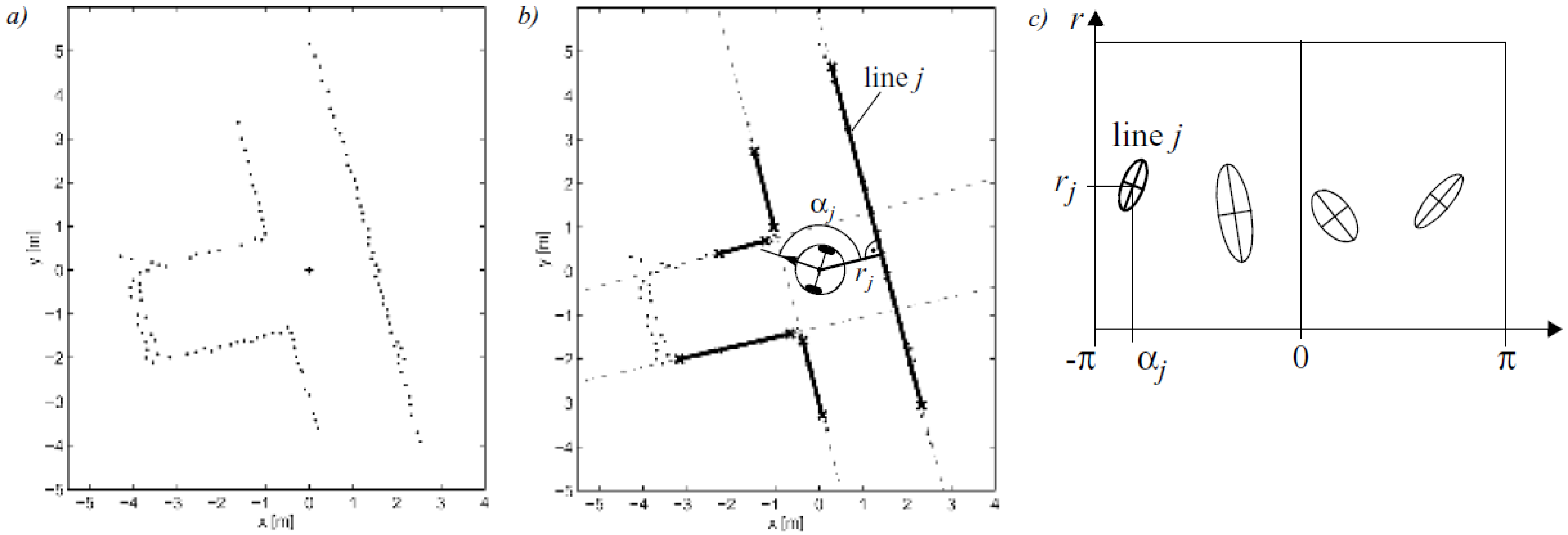
$$p(k+1) = \begin{bmatrix} x(k+1) \\ y(k+1) \\ \theta(k+1) \end{bmatrix}$$



Kalman filter localisation – Case study

- 3. Observation – Using laser rangefinder

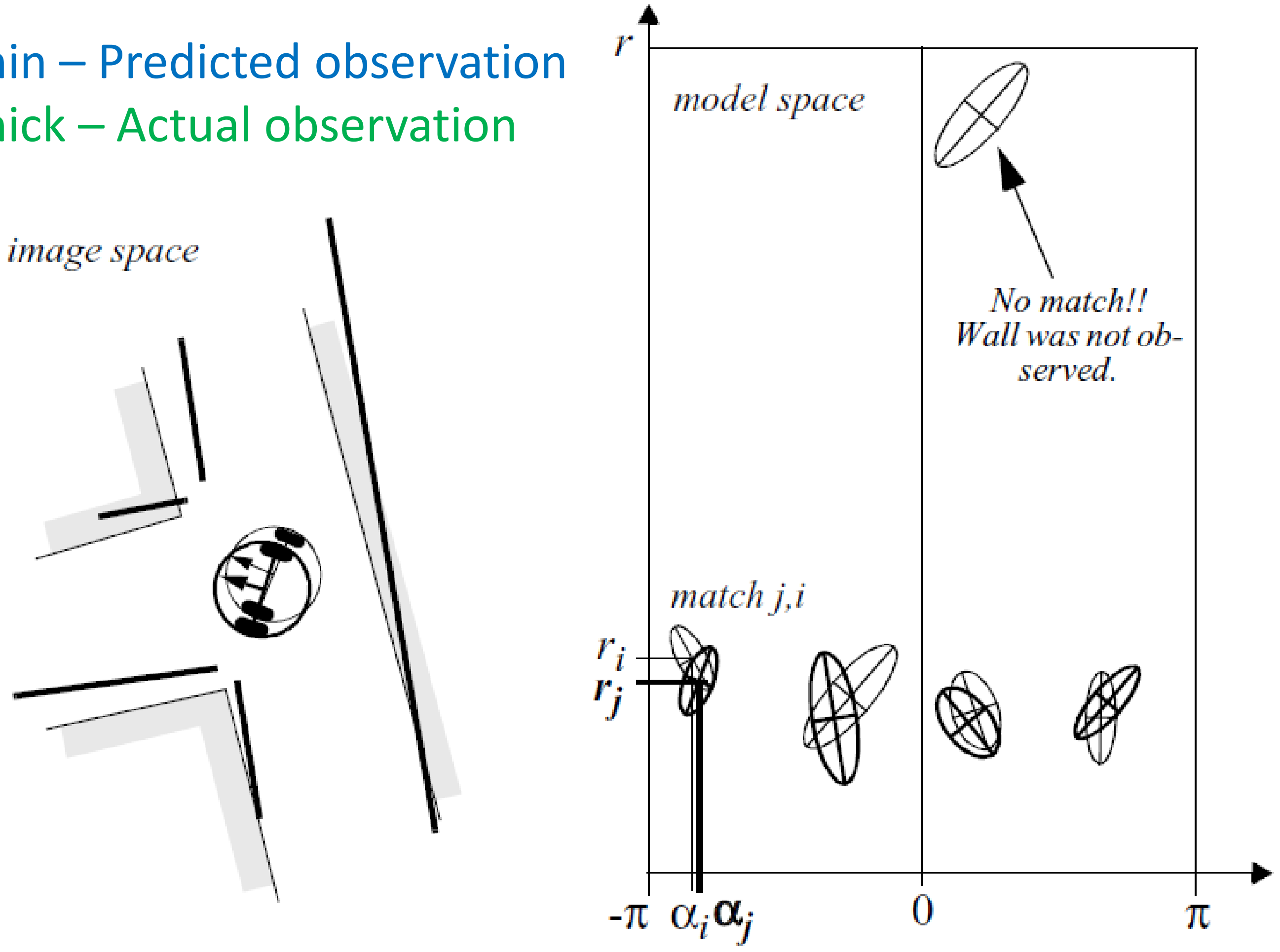
$$z_j(k+1) = {}^{R}\begin{bmatrix} \alpha_j \\ r_j \end{bmatrix} \qquad \Sigma_{R,j} = \begin{bmatrix} \sigma_{\alpha\alpha} & \sigma_{\alpha r} \\ \sigma_{r\alpha} & \sigma_{rr} \end{bmatrix}_j$$



Kalman filter localisation – Case study

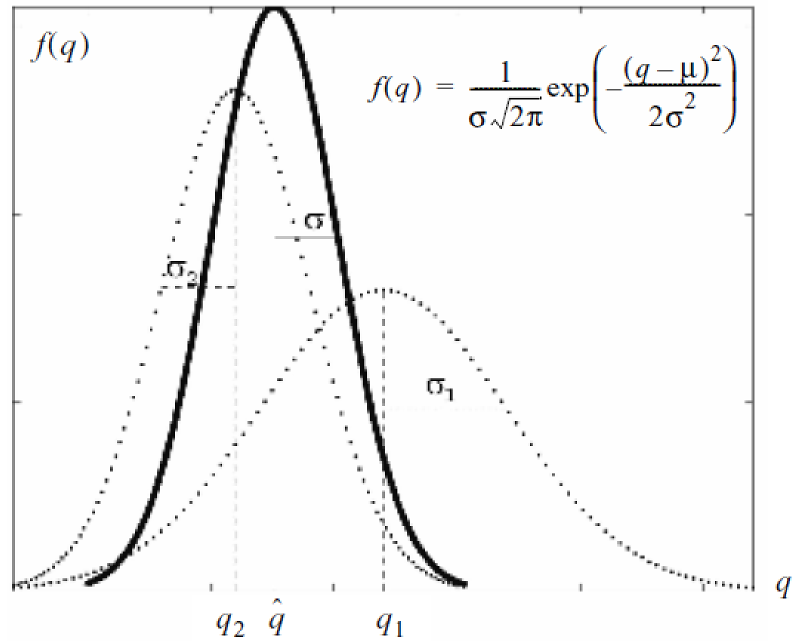
- 4. Matching – Between predicted and actual observation

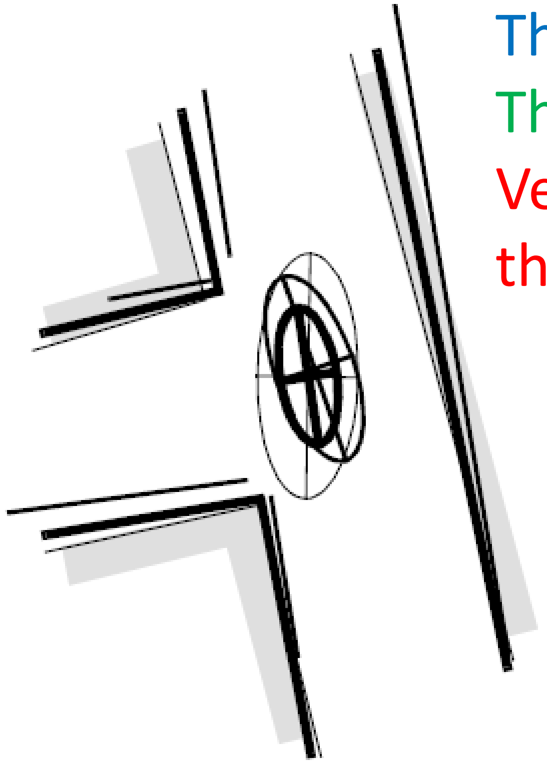
Thin – Predicted observation
Thick – Actual observation



Kalman filter localisation – Case study

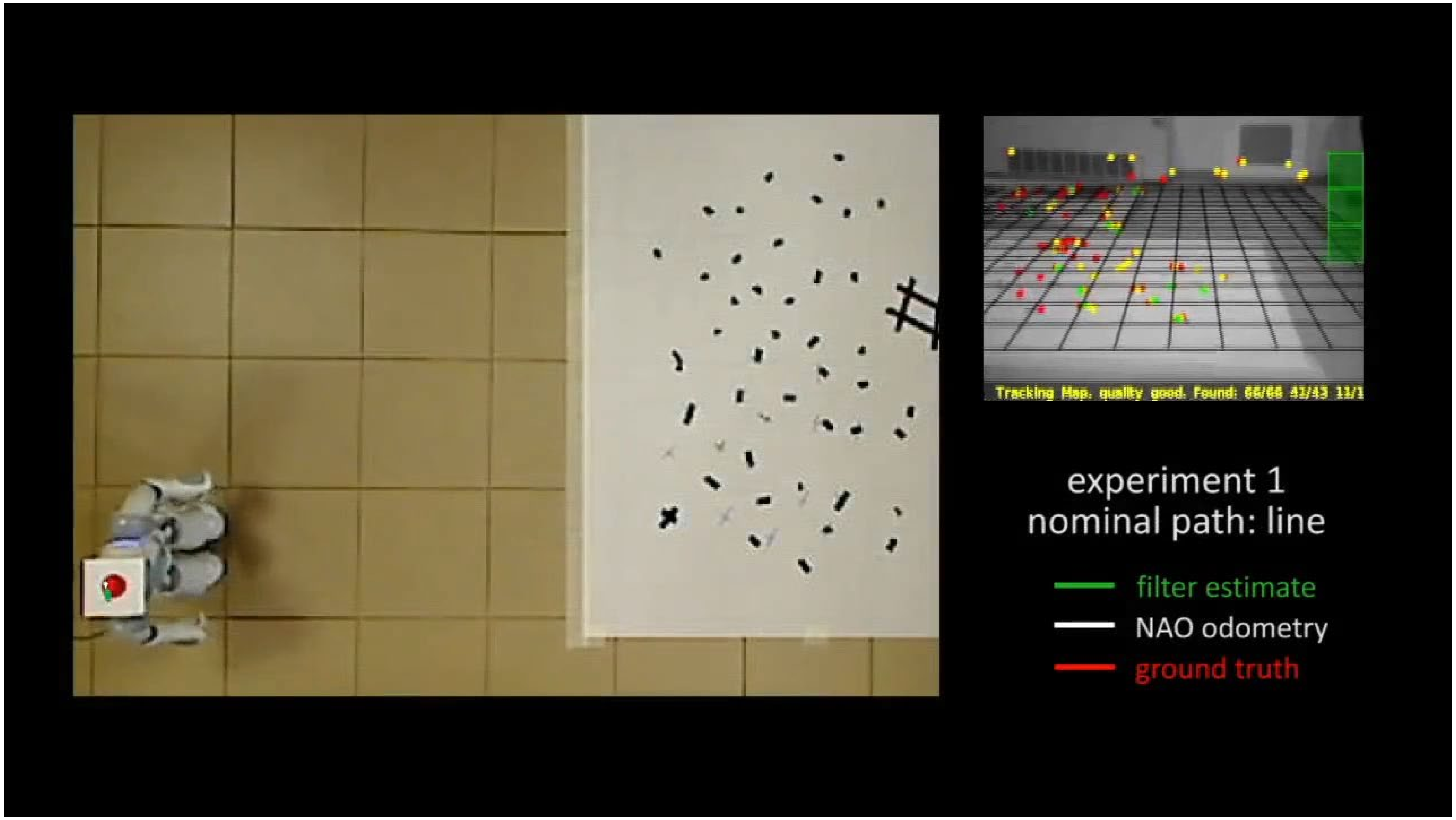
- 5. Estimation – Applying the Bayes rule





Thin – Prediction of robot position
Thick – Position Measurement
Very thick – updated estimate of the robot position

Kalman filter localisation – Example



https://www.youtube.com/watch?v=h548R3N49No

Kalman filter localisation – Summary

Advantages	Disadvantages
• Inherently very precise • Accurate mathematical formulation • Very efficient • Compact representation of the probability distribution (Gaussian assumption)	• Not suited for discrete map representation (not an analytic model, e.g., occupancy grid) • Not suited for global localisation problem as the Gaussian assumption for the probability distribution taken by Kalman filter is violated • If the uncertainty of the robot becomes too large (e.g., collision with an object), the Kalman filter will fail and the position is definitively lost.

slido

Which of these probabilistic map-based localisation methods do you think are suited for a REAL Micromouse? (select one or more)

ⓘ Start presenting to display the poll results on this slide.

What if the map is unknown?

SLAM

What is SLAM (Simultaneous Localisation and Mapping)?

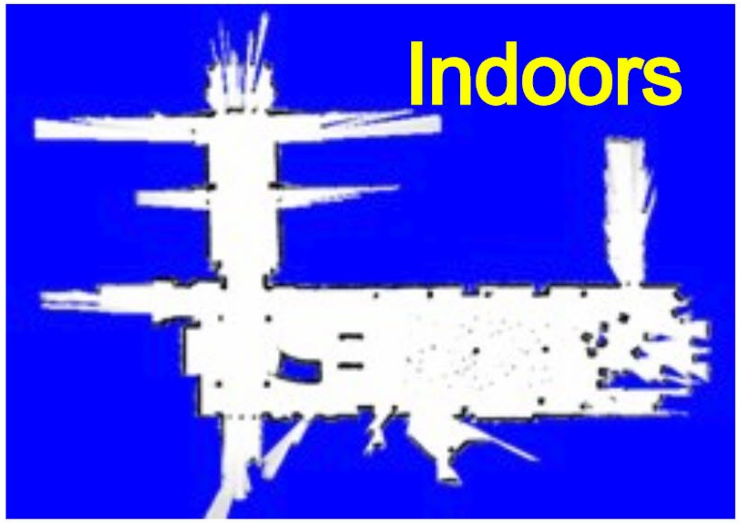
- Problem statement
  - The computational problem of constructing or updating a map of an unknown environment while simultaneously keeping track of an agent's location within it.

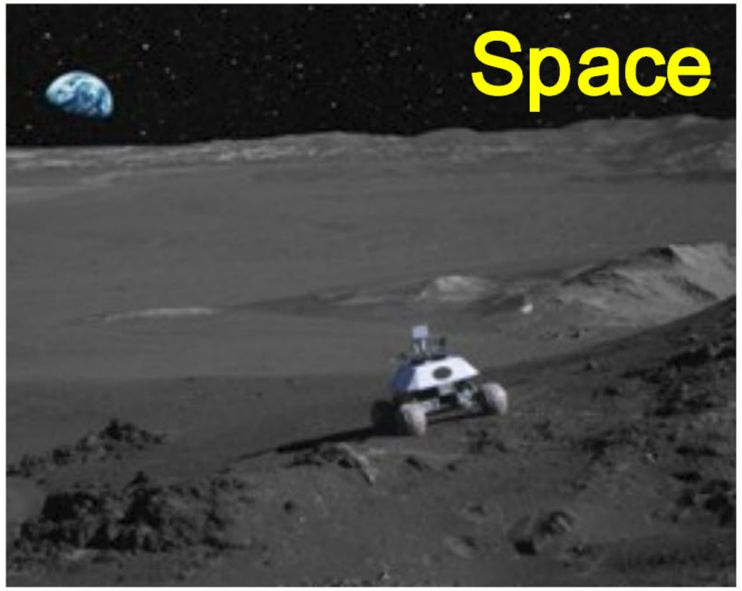
- When is it necessary?
  - When there is no prior knowledge about the environment (in contrast to map-based localisation), and
  - When the localisation of the robot cannot be determined exclusively on external positioning systems like GPS (in contrast to pure mapping or localisation without mapping)

- One of the essential competences of a truly autonomous mobile robot.

What is SLAM (Simultaneous Localisation and Mapping)?

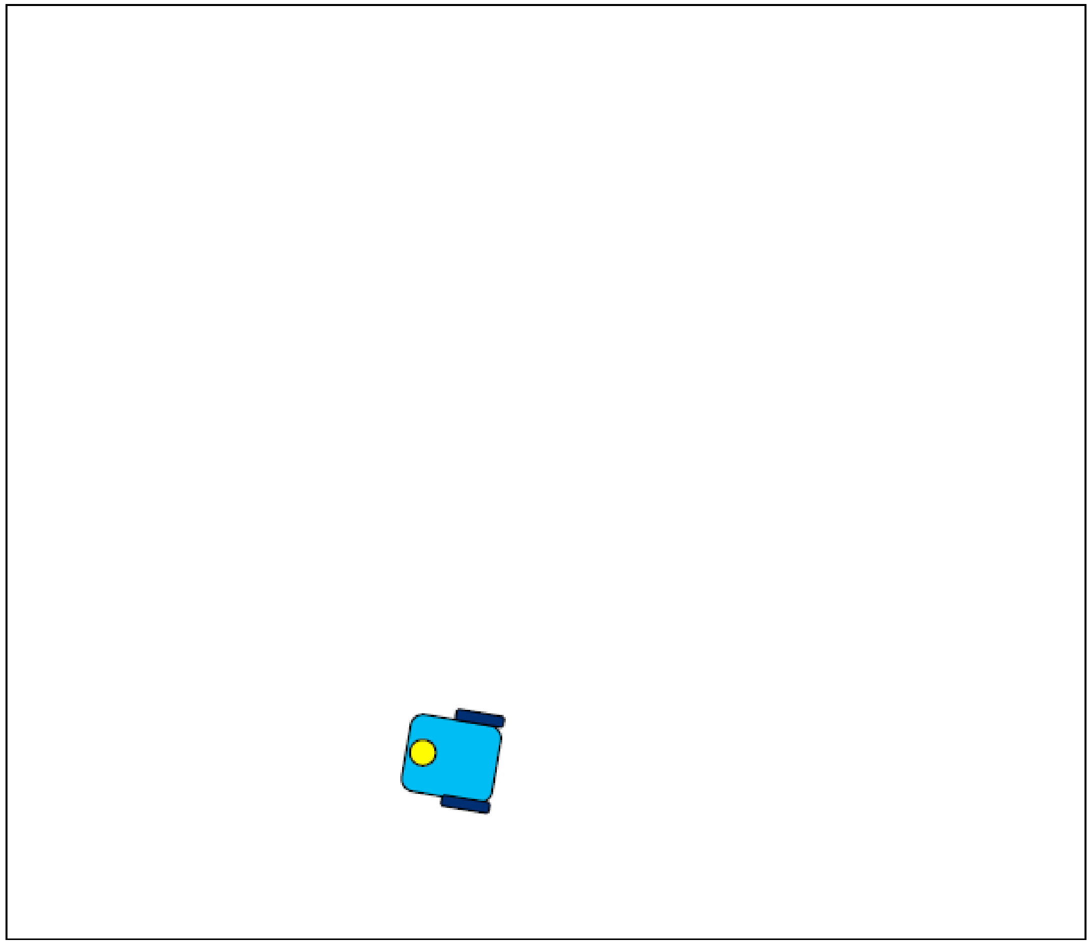
- Wide applications:
  - Field robots
    - UGV, UAV, AUV, self-driving cars, planetary rovers…
  - Medical robots
    - e.g., SLAM inside body
  - VR
  - AR
  - …





SLAM

- One of the most challenging problems in mobile robotics.
- The *chicken-or-egg* dilemma
  - For localisation the robot needs to know the map;
  - For mapping the robot needs to know its location.
- Loop closure
  - The uncertainty keeps accumulating until the robot observes features whose location has already been estimated.

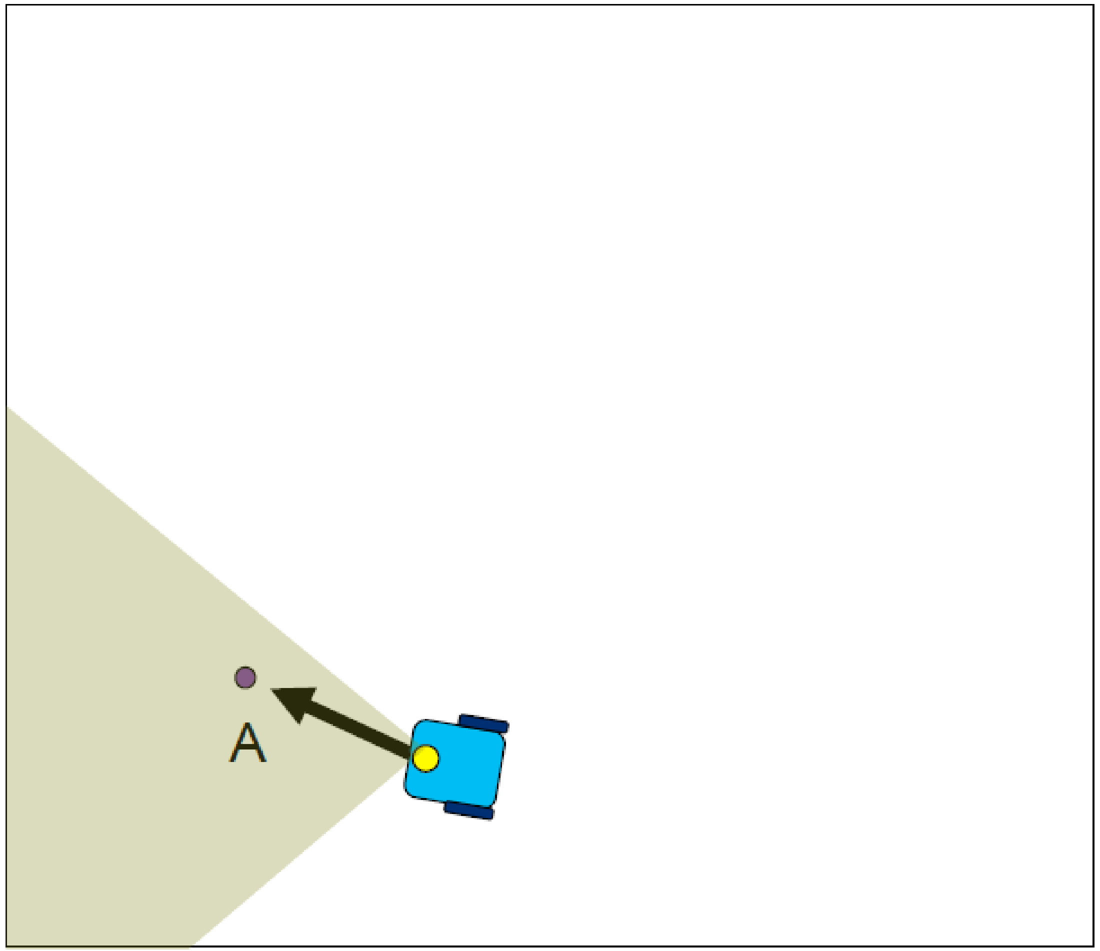

The robot starts with no knowledge about the environment.

SLAM

- One of the most challenging problems in mobile robotics.

- The *chicken-or-egg* dilemma
    - For localisation the robot needs to know the map;
    - For mapping the robot needs to know its location.

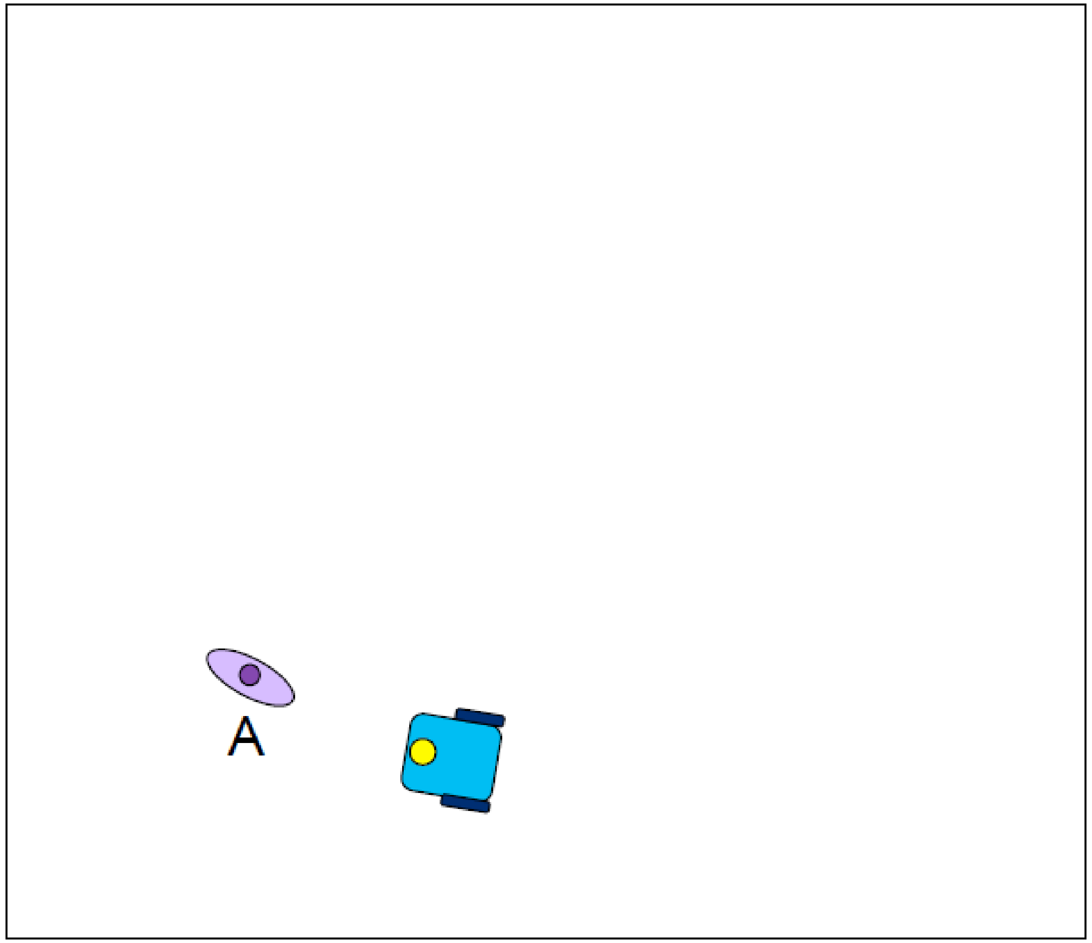
- Loop closure
    - The uncertainty keeps accumulating until the robot observes features whose location has already been estimated.



The robot observes a feature in the environment.

SLAM

- One of the most challenging problems in mobile robotics.
- The *chicken-or-egg* dilemma
  - For localisation the robot needs to know the map;
  - For mapping the robot needs to know its location.
- Loop closure
  - The uncertainty keeps accumulating until the robot observes features whose location has already been estimated.



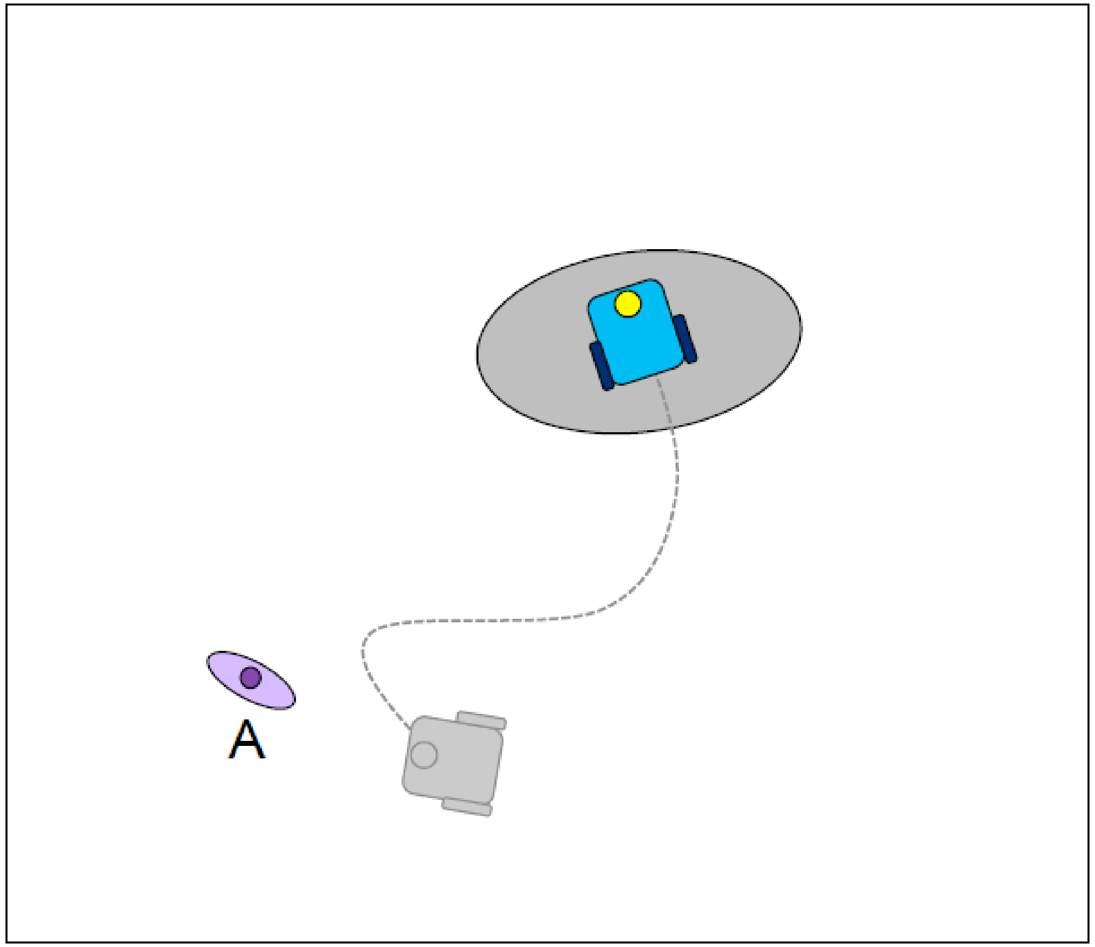
The robot starts building the map with the feature.

SLAM

- One of the most challenging problems in mobile robotics.

- The *chicken-or-egg* dilemma
  - For localisation the robot needs to know the map;
  - For mapping the robot needs to know its location.

- Loop closure
  - The uncertainty keeps accumulating until the robot observes features whose location has already been estimated.



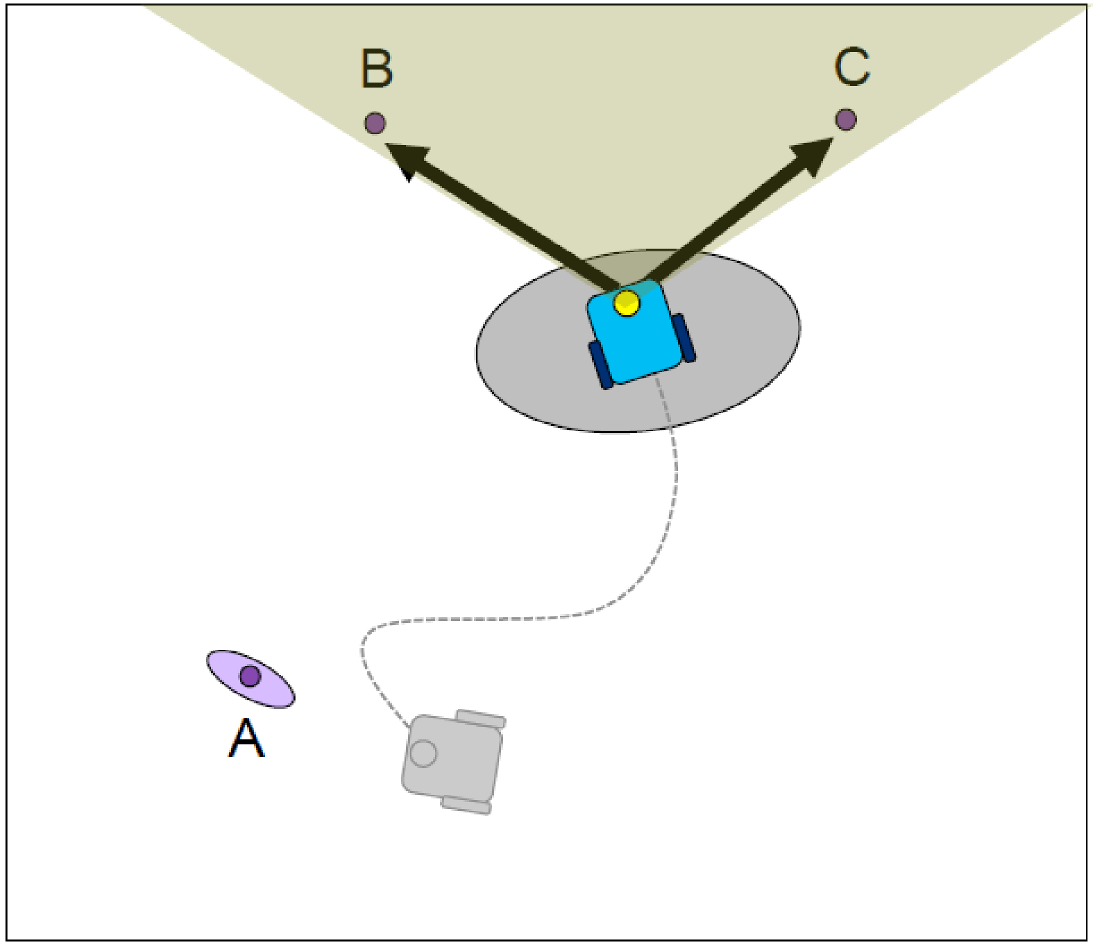
The robot travels for a distance and tracks its location with some uncertainty.

SLAM

- One of the most challenging problems in mobile robotics.

- The *chicken-or-egg* dilemma
  - For localisation the robot needs to know the map;
  - For mapping the robot needs to know its location.

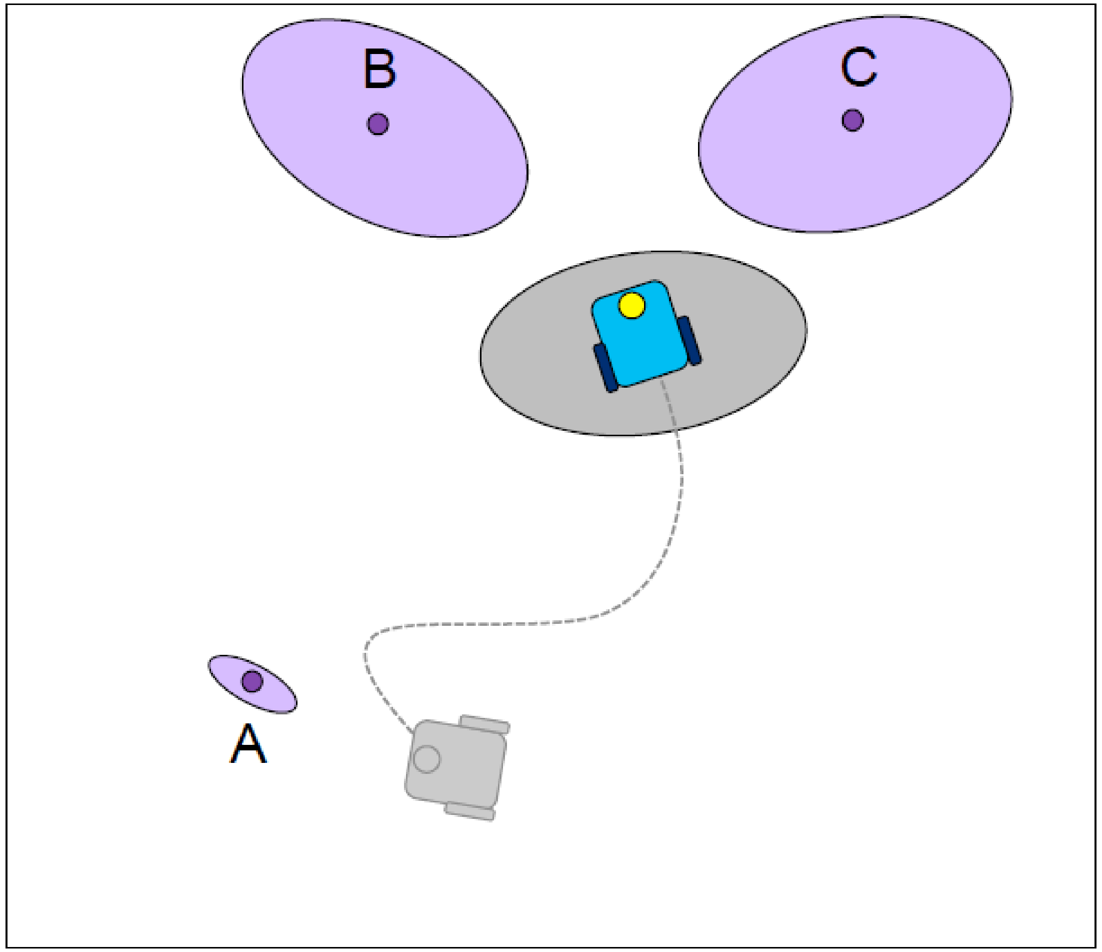
- Loop closure
  - The uncertainty keeps accumulating until the robot observes features whose location has already been estimated.



The robot observes two additional features in the environment.

SLAM

- One of the most challenging problems in mobile robotics.
- The *chicken-or-egg* dilemma
  - For localisation the robot needs to know the map;
  - For mapping the robot needs to know its location.
- Loop closure
  - The uncertainty keeps accumulating until the robot observes features whose location has already been estimated.



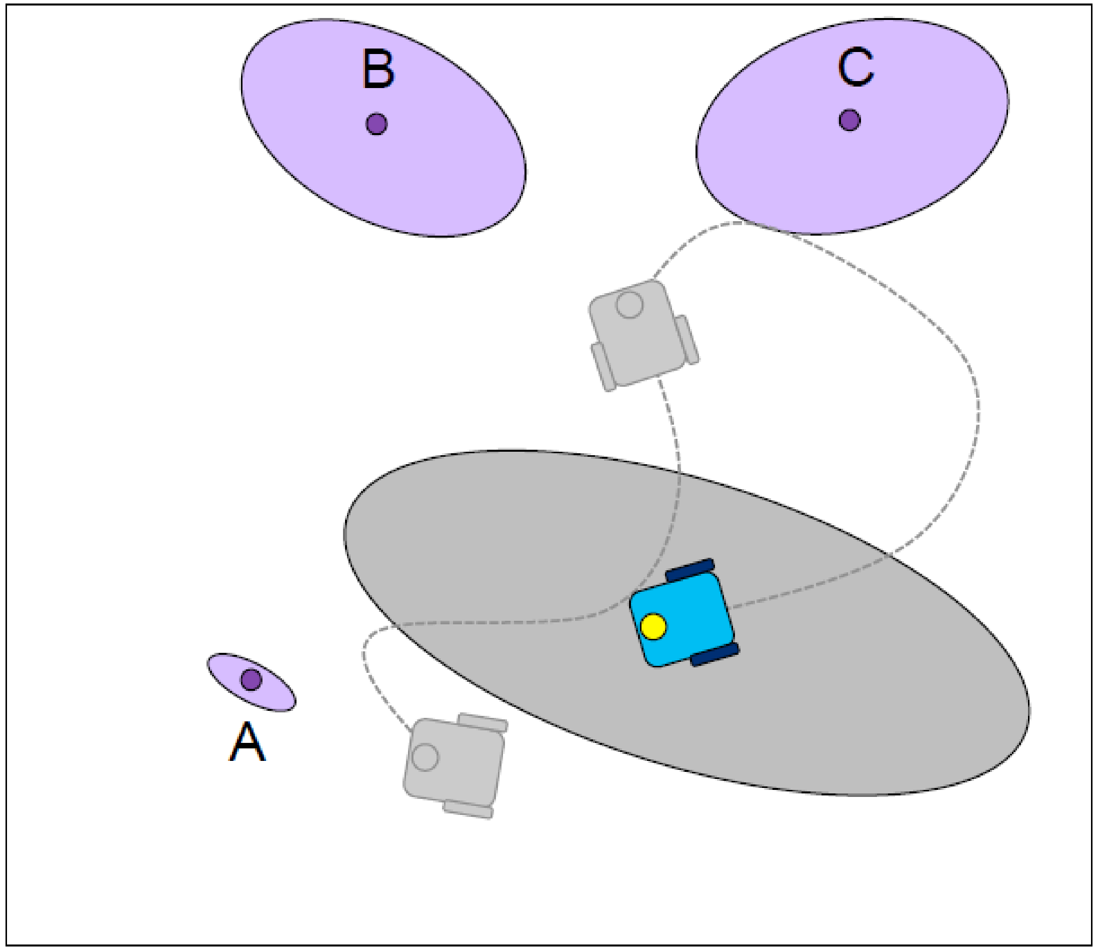
The robot maps these two features with larger uncertainty due to the uncertainty of its own pose.

SLAM

- One of the most challenging problems in mobile robotics.

- The *chicken-or-egg* dilemma
  - For localisation the robot needs to know the map;
  - For mapping the robot needs to know its location.

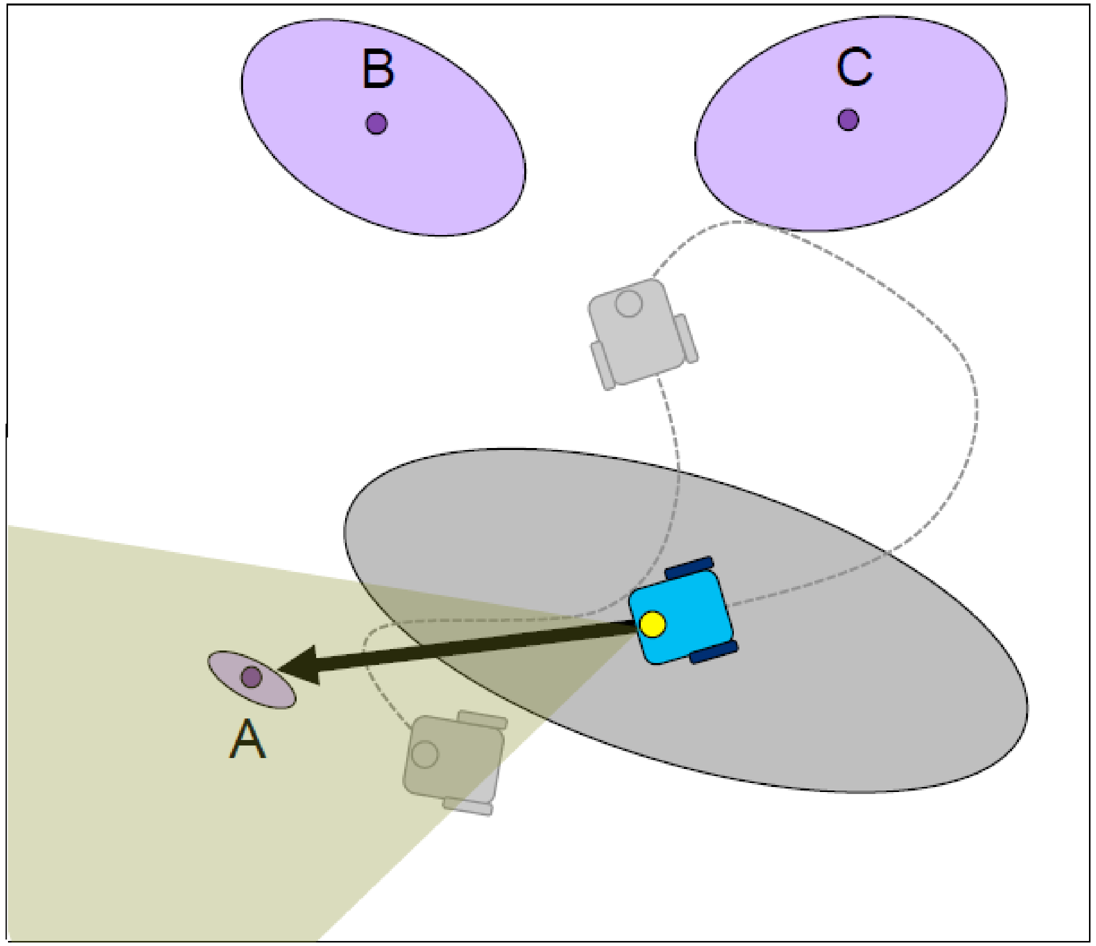
- Loop closure
  - The uncertainty keeps accumulating until the robot observes features whose location has already been estimated.



The robot continues moving and the uncertainty of its location accumulates.

SLAM

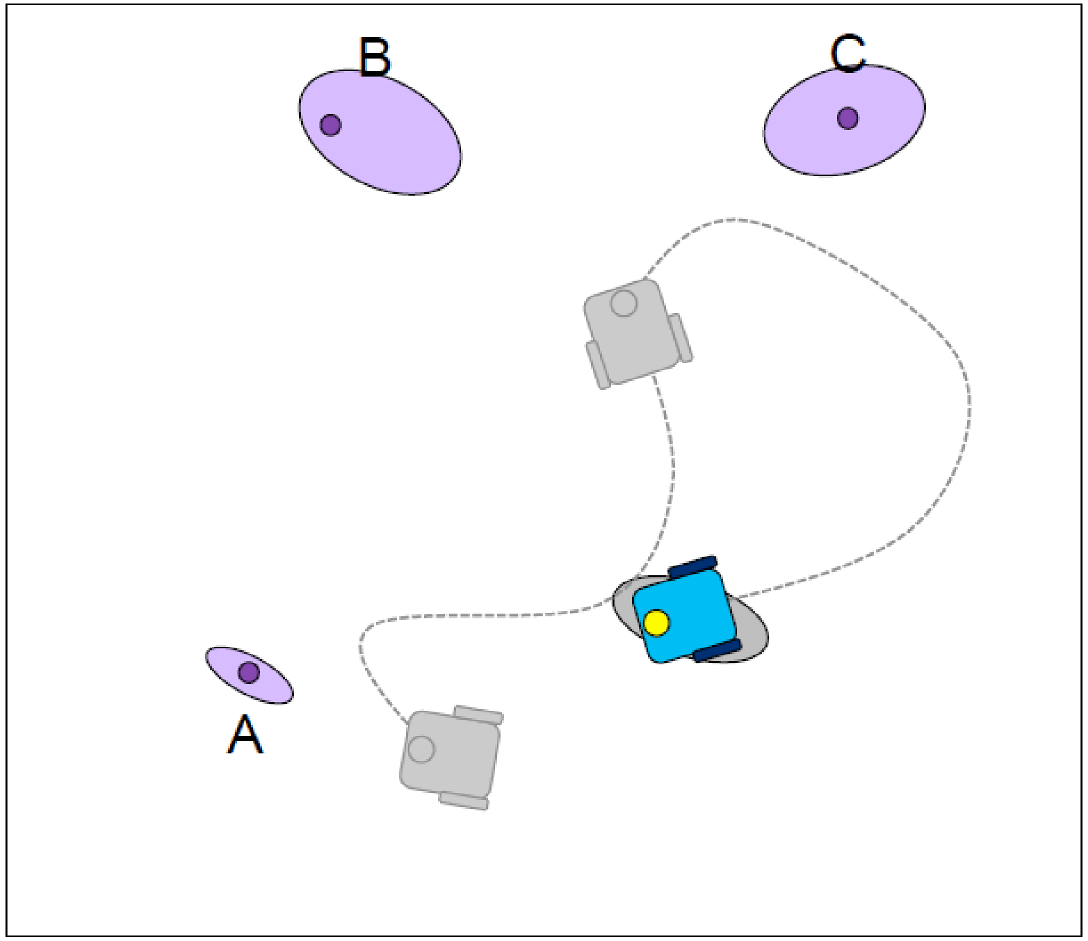
- One of the most challenging problems in mobile robotics.
- The *chicken-or-egg* dilemma
  - For localisation the robot needs to know the map;
  - For mapping the robot needs to know its location.
- Loop closure
  - The uncertainty keeps accumulating until the robot observes features whose location has already been estimated.



The robot observes a feature that has been observed before (location known in its belief space).

SLAM

- One of the most challenging problems in mobile robotics.
- The *chicken-or-egg* dilemma
  - For localisation the robot needs to know the map;
  - For mapping the robot needs to know its location.
- Loop closure
  - The uncertainty keeps accumulating until the robot observes features whose location has already been estimated.



The uncertainty of the robot's location and the other features shrinks (loop closure).

SLAM - Methods

- Three main paradigms
  - Extended Kalman Filter (EKF) SLAM
  - Graph-based SLAM
  - Particle filter SLAM

Extended Kalman Filter (EKF) SLAM

- Proceeds exactly like the standard EKF that is used for robot localisation

- Only differentiated in that it uses an extended state vector that consists of both the robot pose and all the features in the map:

$$y_t = [\underbrace{x_t}_{\text{Robot pose}}, \underbrace{m_1, \ldots, m_{n-1}}_{\text{Features in the map}}]^T$$

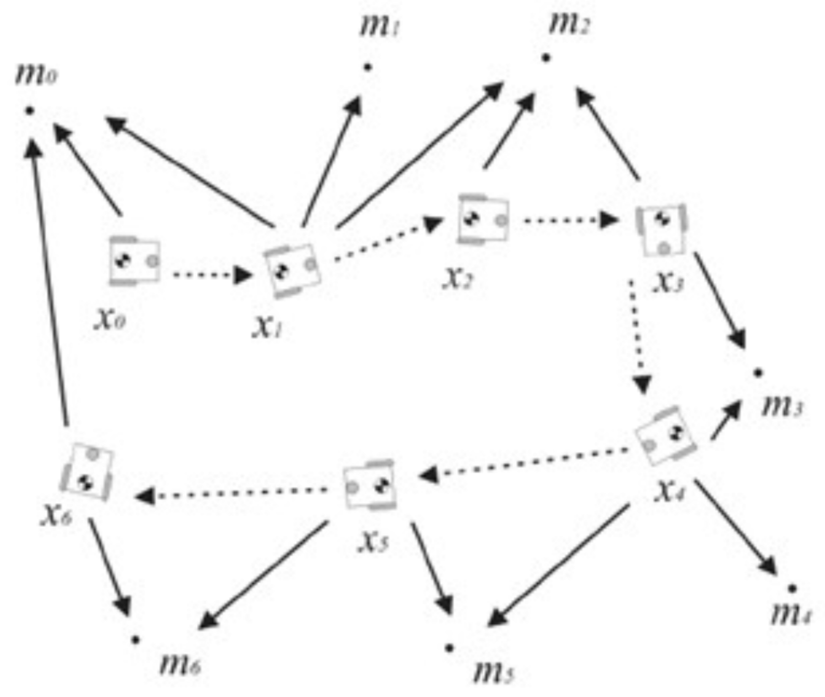
EKF SLAM – Example: MonoSLAM

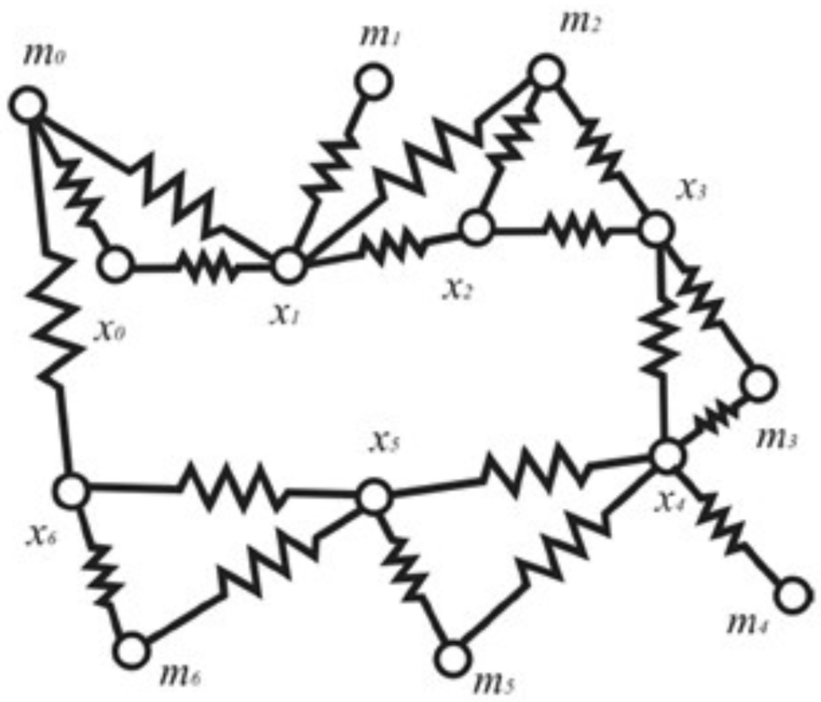
Real-Time
Camera Tracking
in Unknown Scenes

https://www.youtube.com/watch?v=mimAWVm-0qA

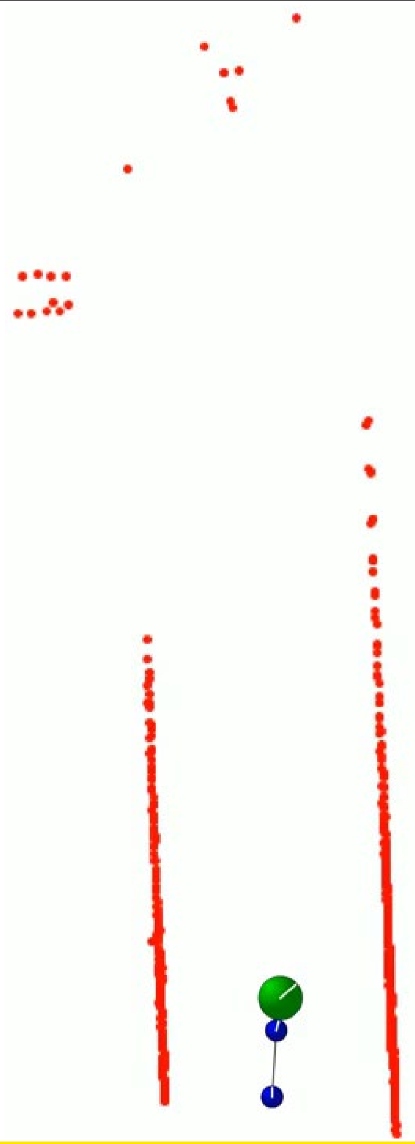
Graph-based SLAM

- Treat the FULL SLAM problem as a graph optimization
- Solve for the constraints between poses and landmarks
- Globally consistent solution, but infeasible for large-scale SLAM





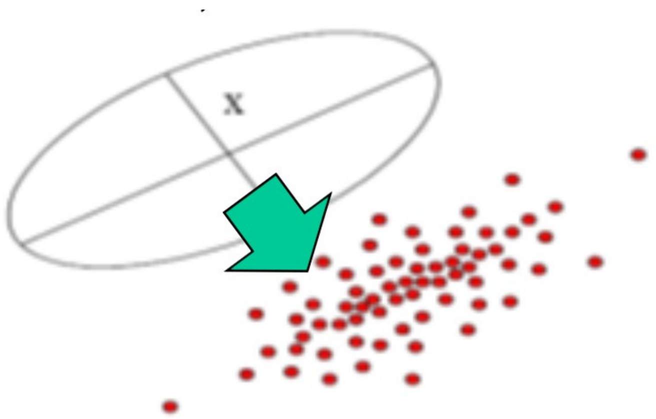
Graph-based SLAM - Example

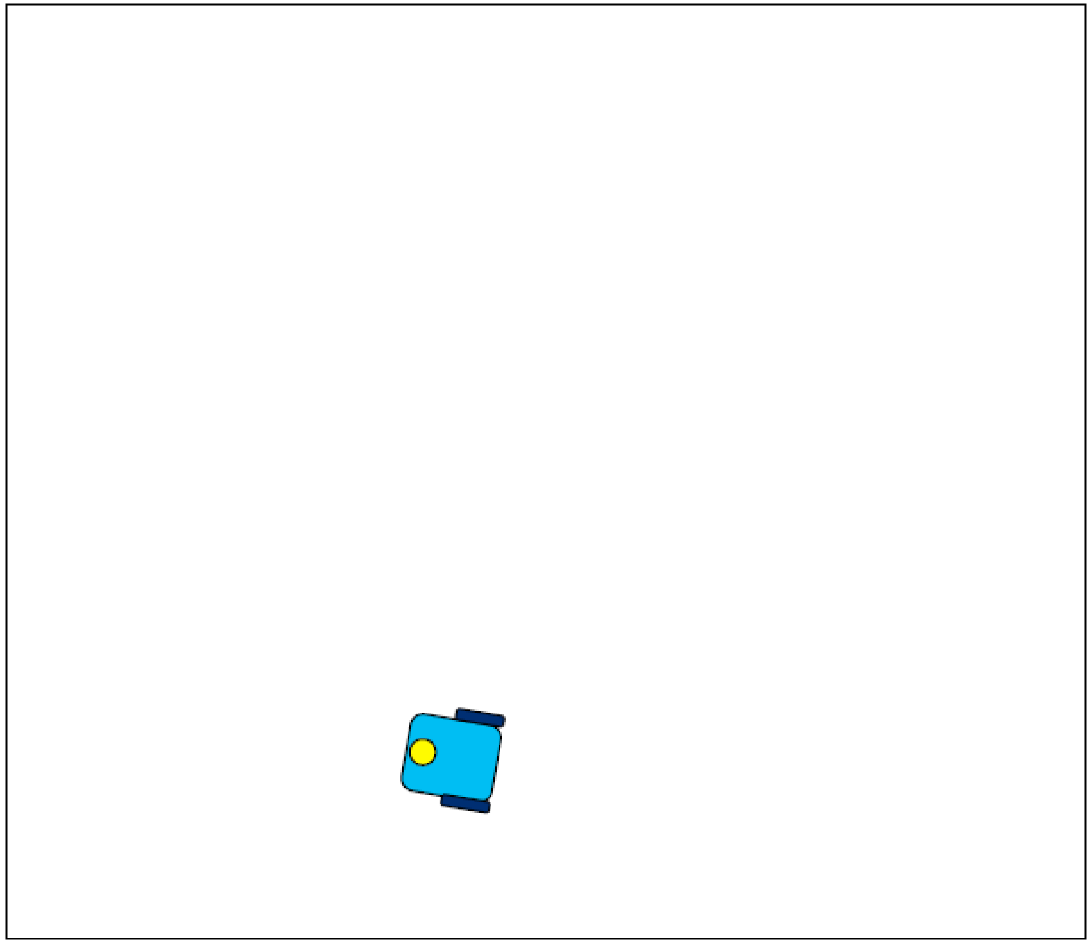


https://www.youtube.com/watch?v=8BUhMhk3JB0

Particle filter SLAM

- Uses particles to represent the uncertainty

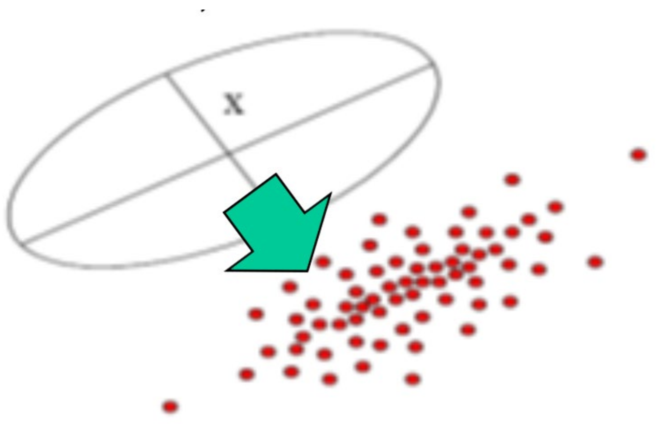


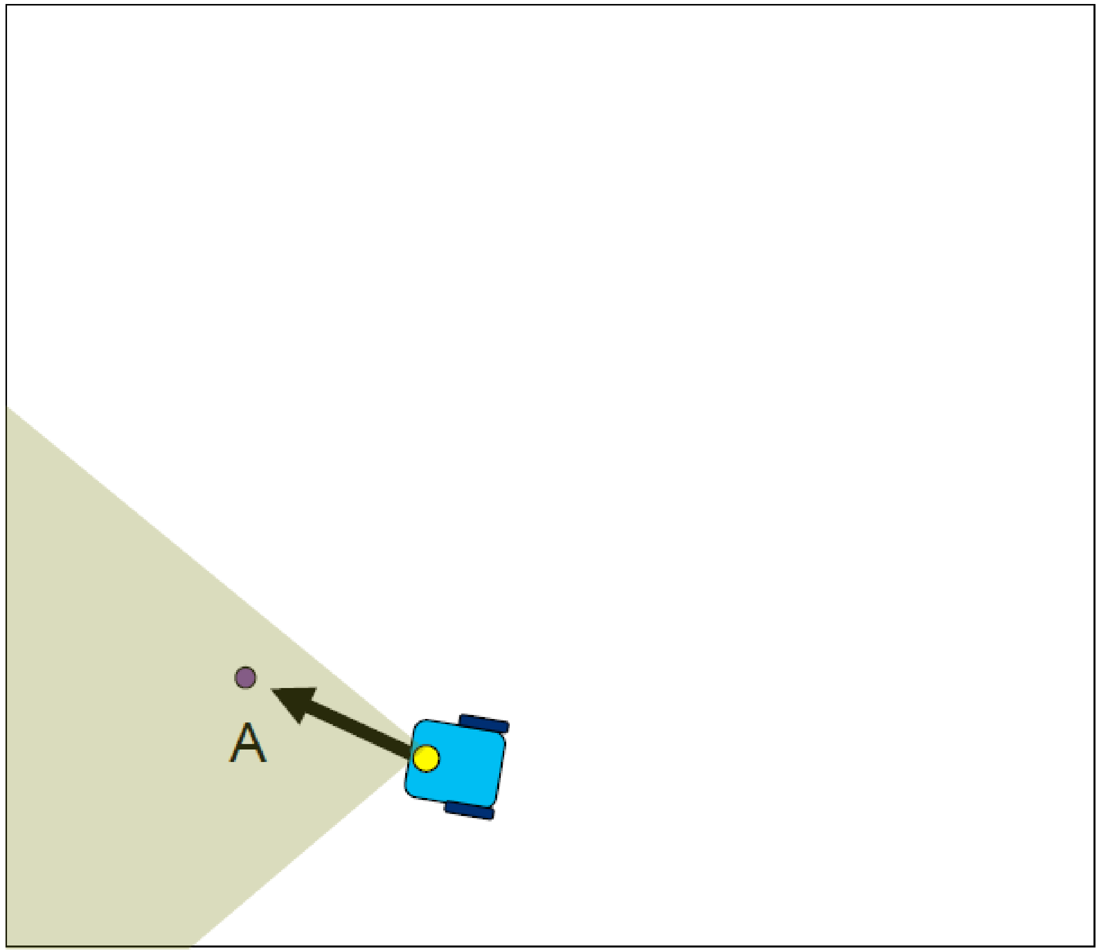


The robot starts with no knowledge about the environment.

Particle filter SLAM

- Uses particles to represent the uncertainty

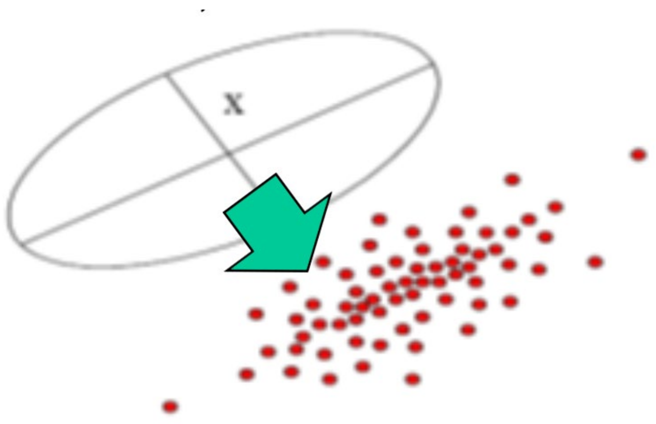


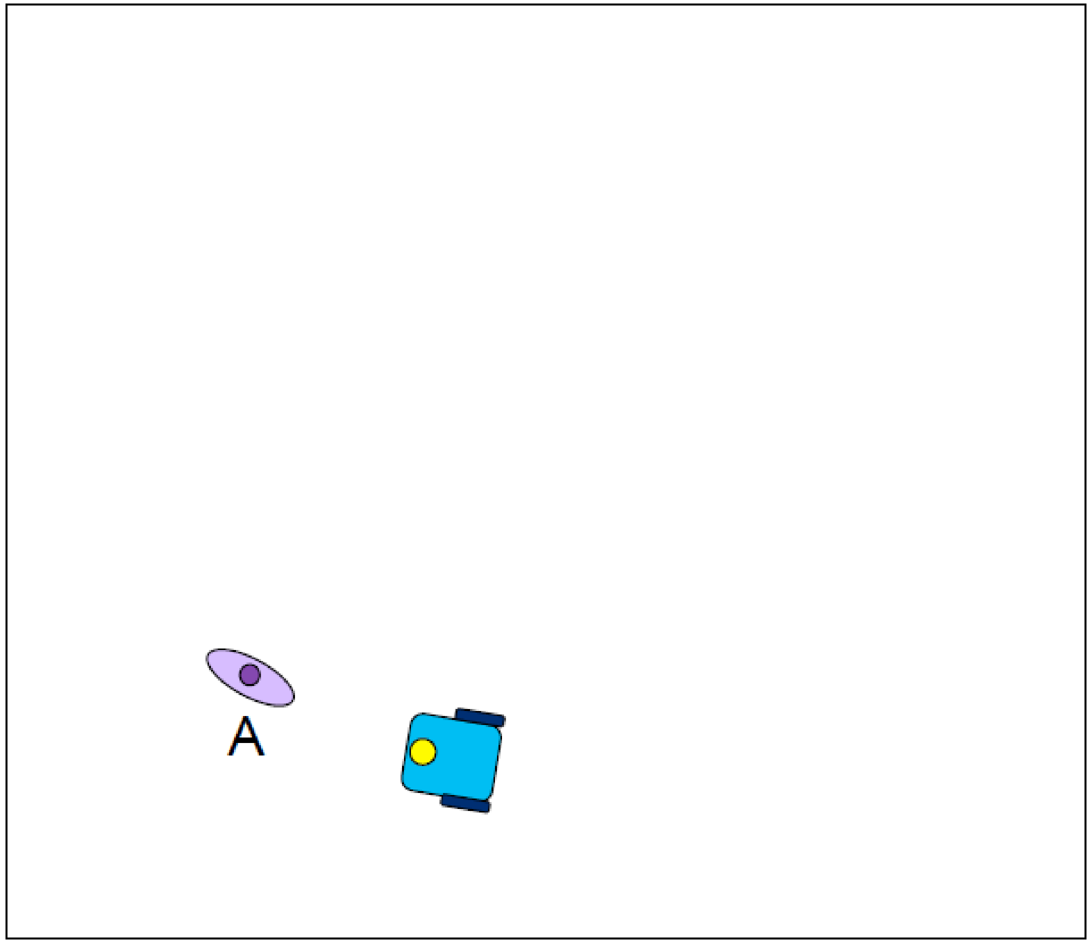


The robot observes a feature in the environment.

Particle filter SLAM

- Uses particles to represent the uncertainty

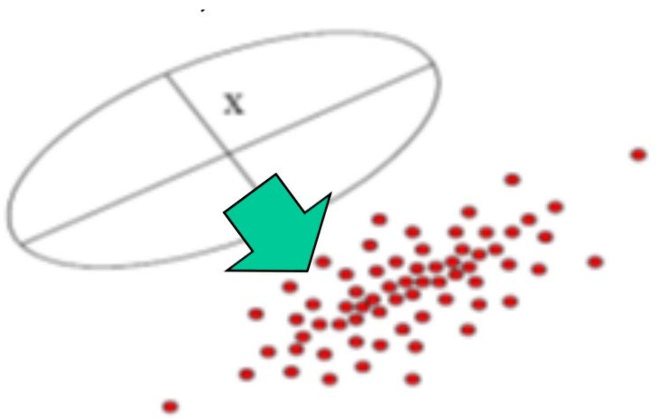


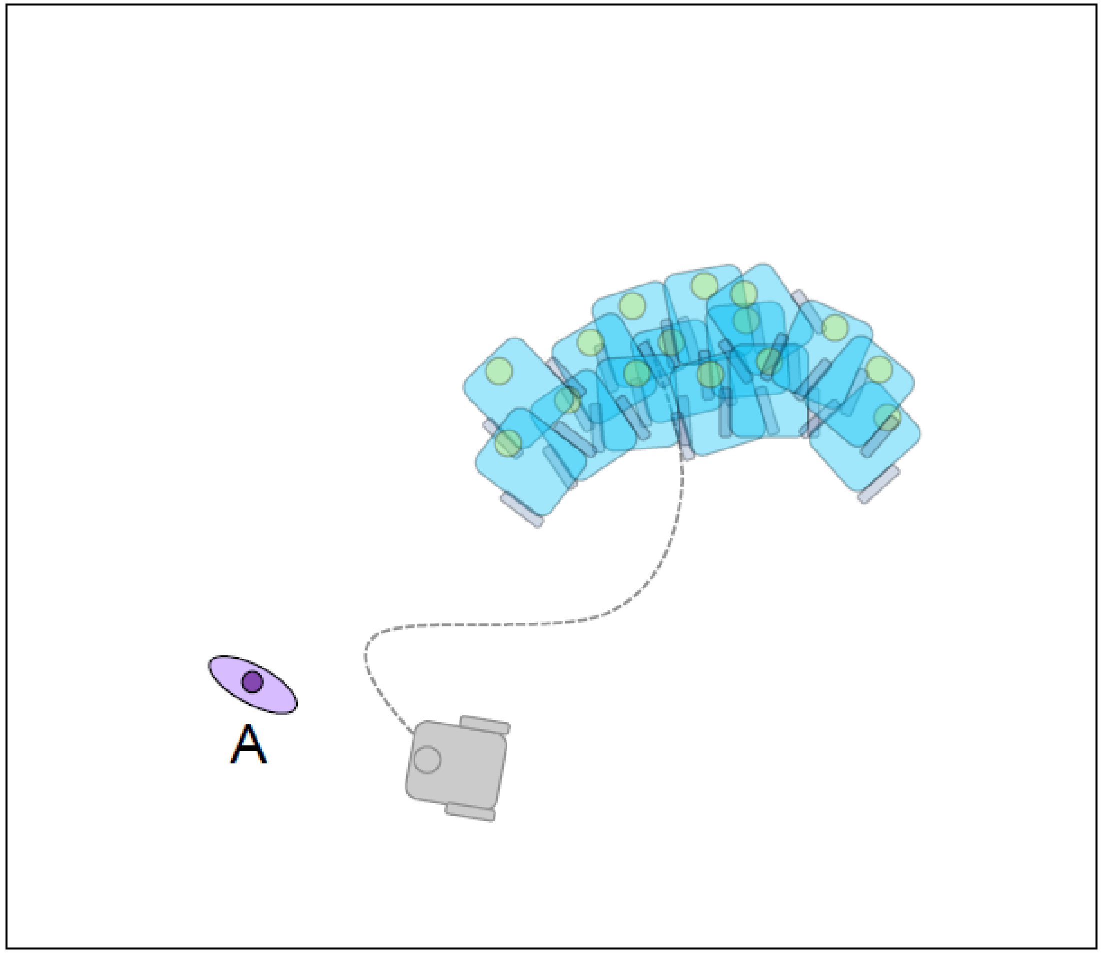


The robot starts building the map with the feature.

Particle filter SLAM

- Uses particles to represent the uncertainty

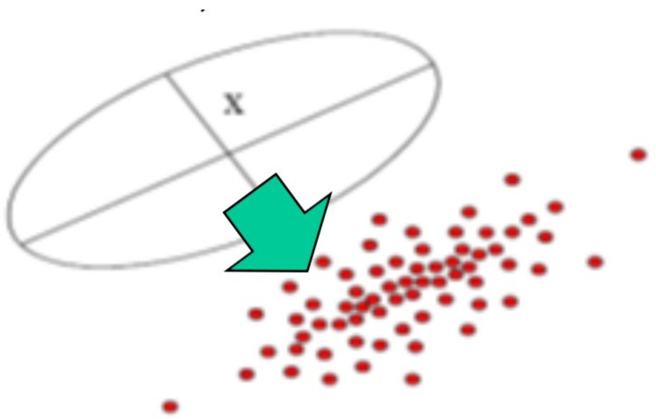


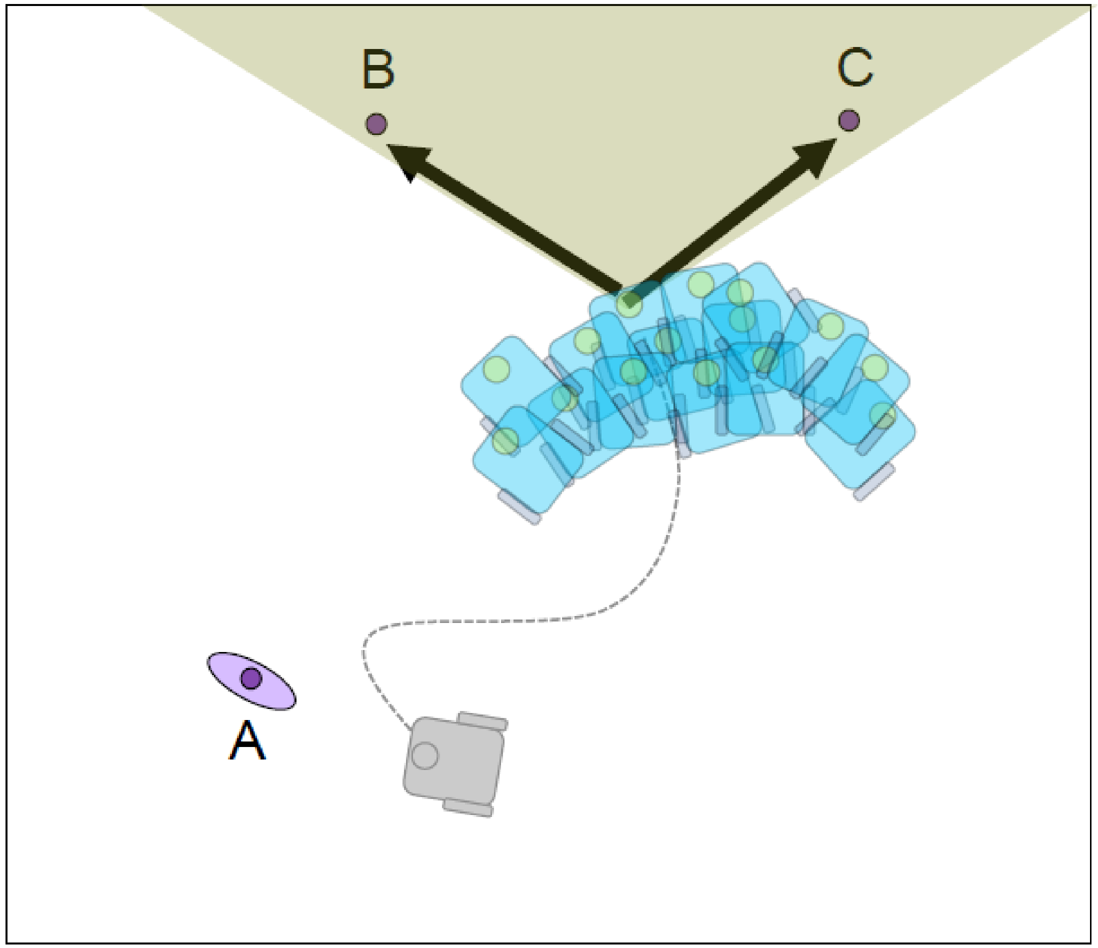


The robot travels for a distance. Particles are used to represent the uncertainty.

Particle filter SLAM

- Uses particles to represent the uncertainty

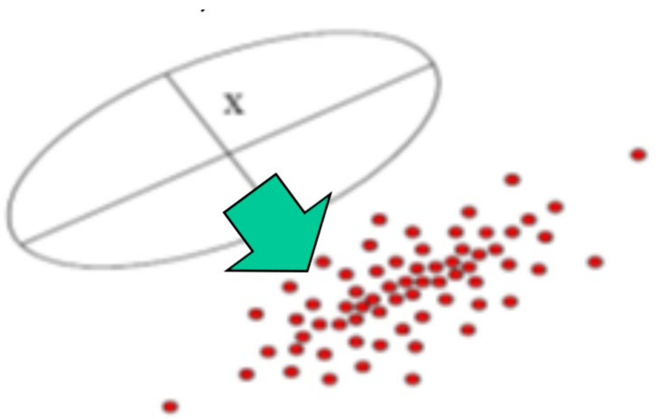


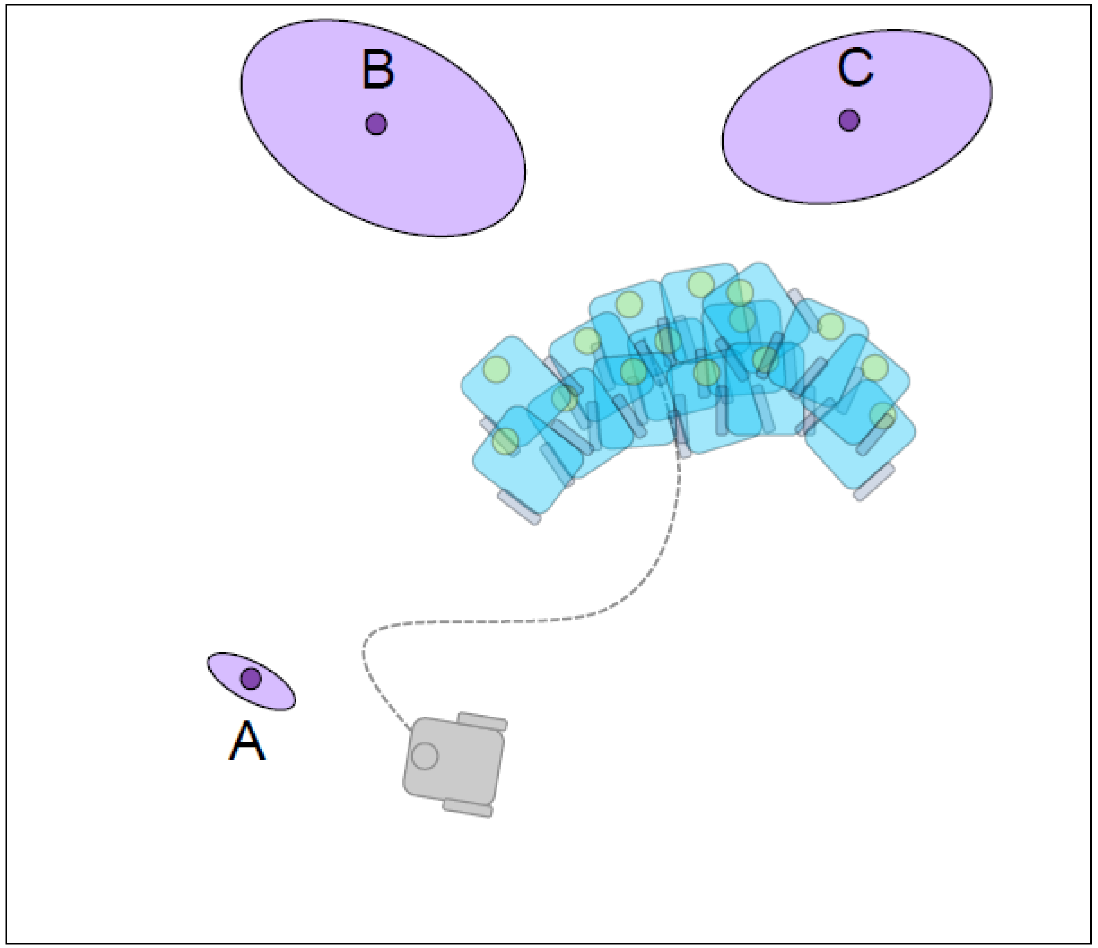


The robot observes two additional features in the map.

Particle filter SLAM

- Uses particles to represent the uncertainty

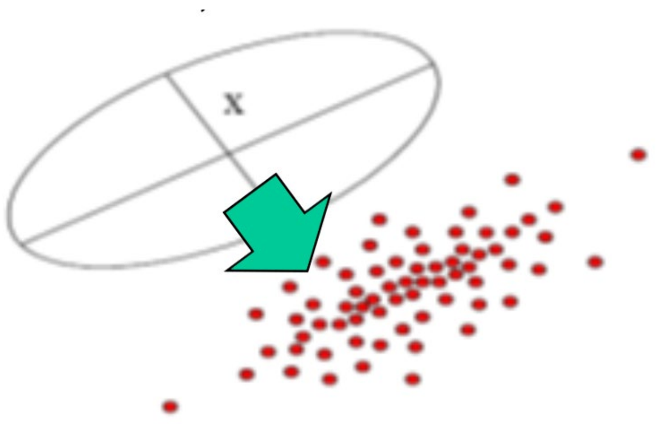


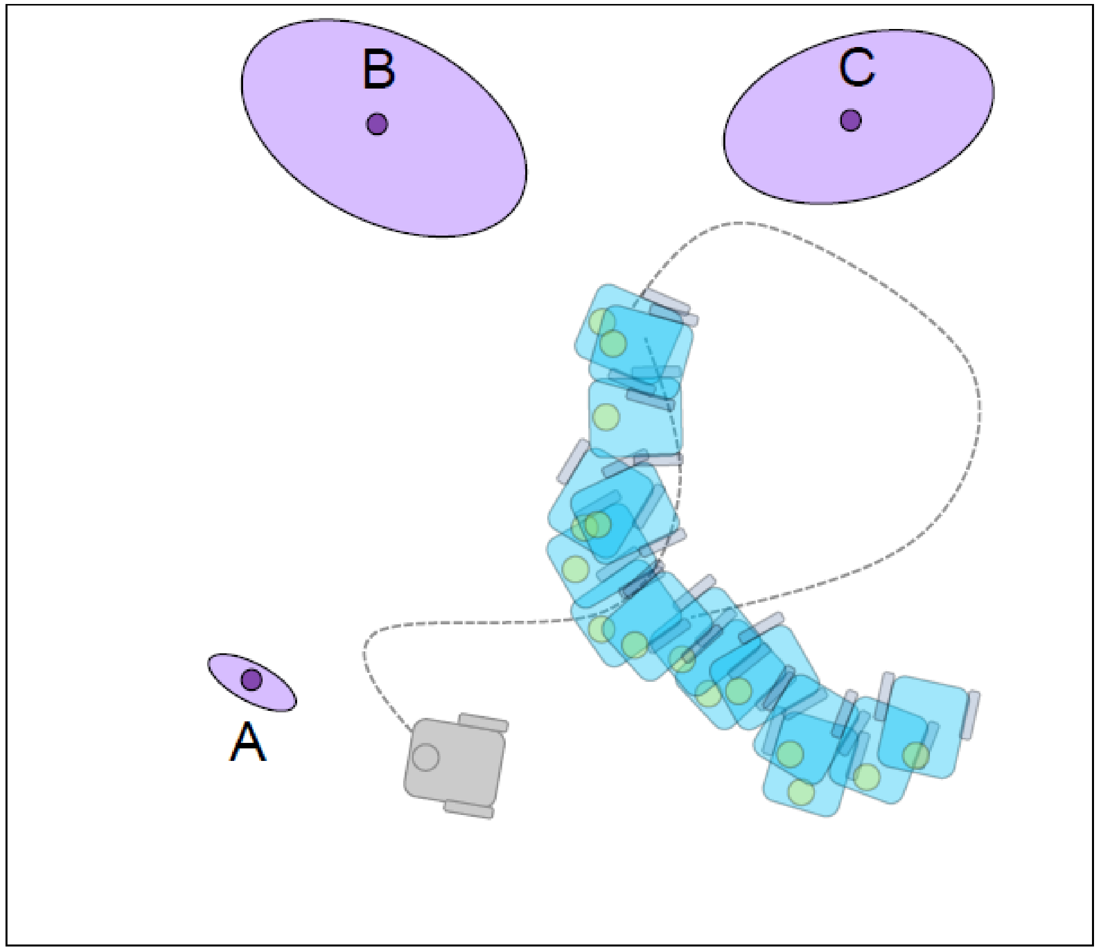


Each particle makes an estimation of the measurements.

Particle filter SLAM

- Uses particles to represent the uncertainty

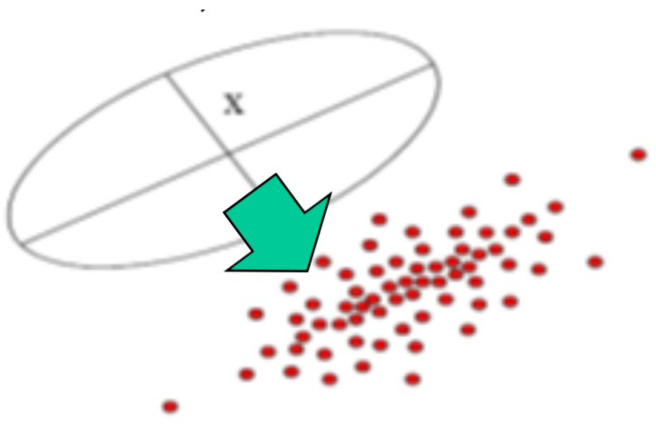


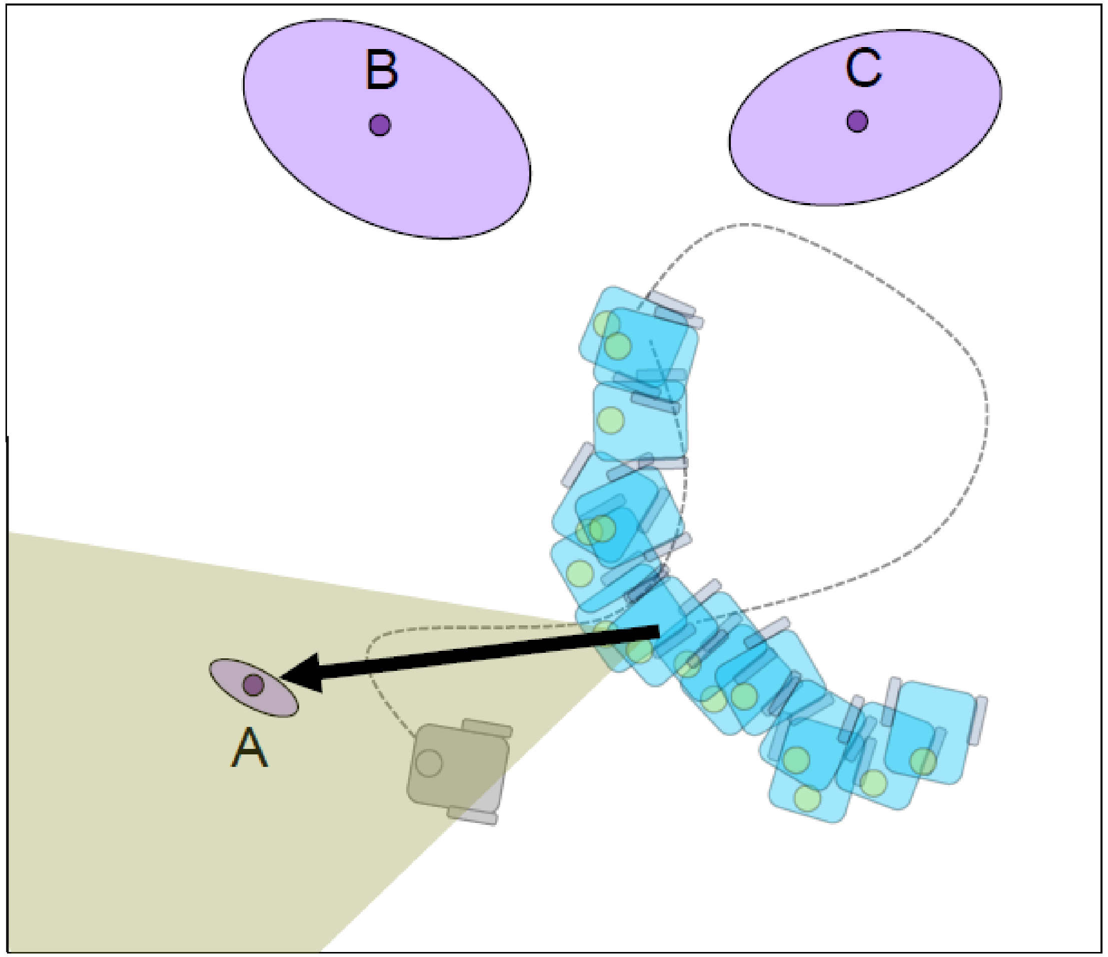


Each particle predicts the next pose based on the motion model.

Particle filter SLAM

- Uses particles to represent the uncertainty

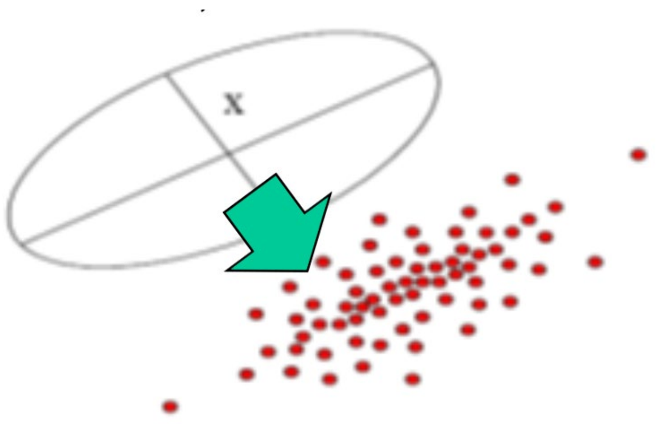


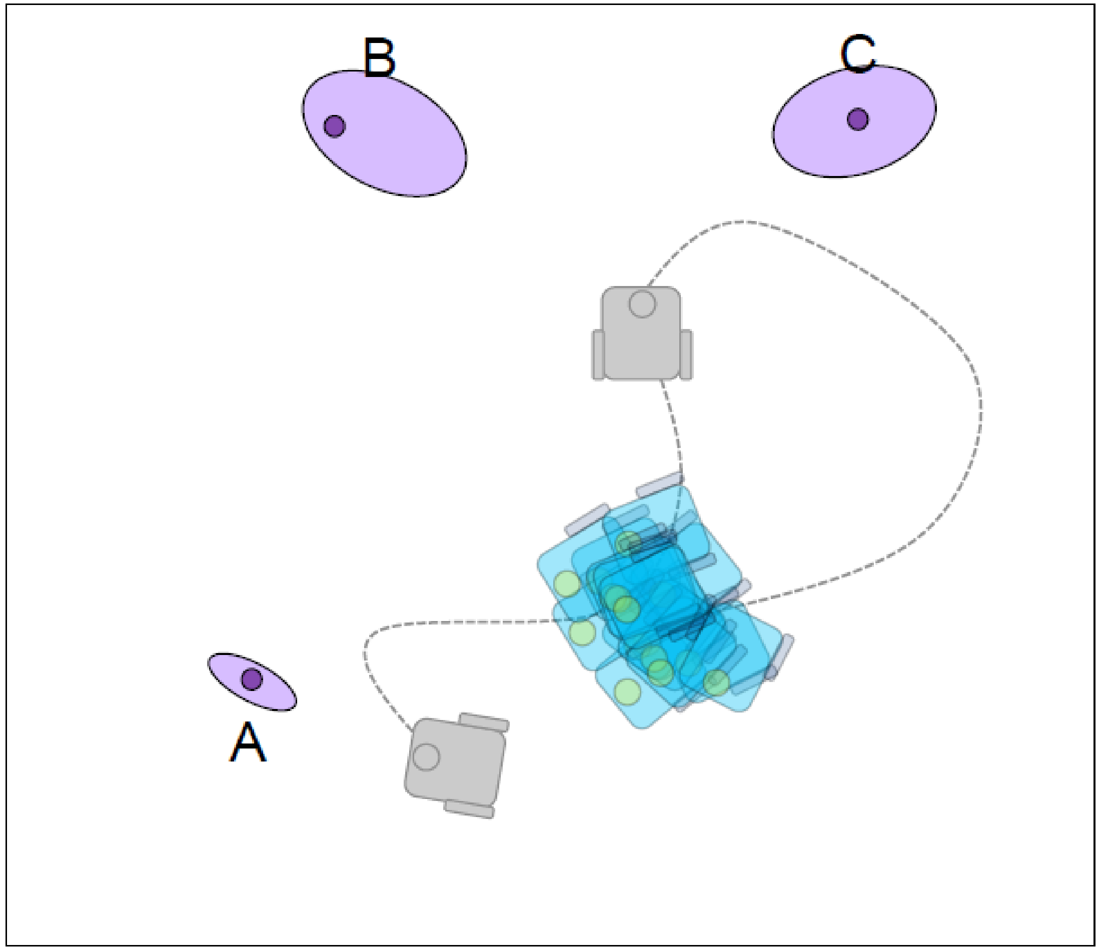


Loop closure. The particles with better match get higher weights.

Particle filter SLAM

- Uses particles to represent the uncertainty





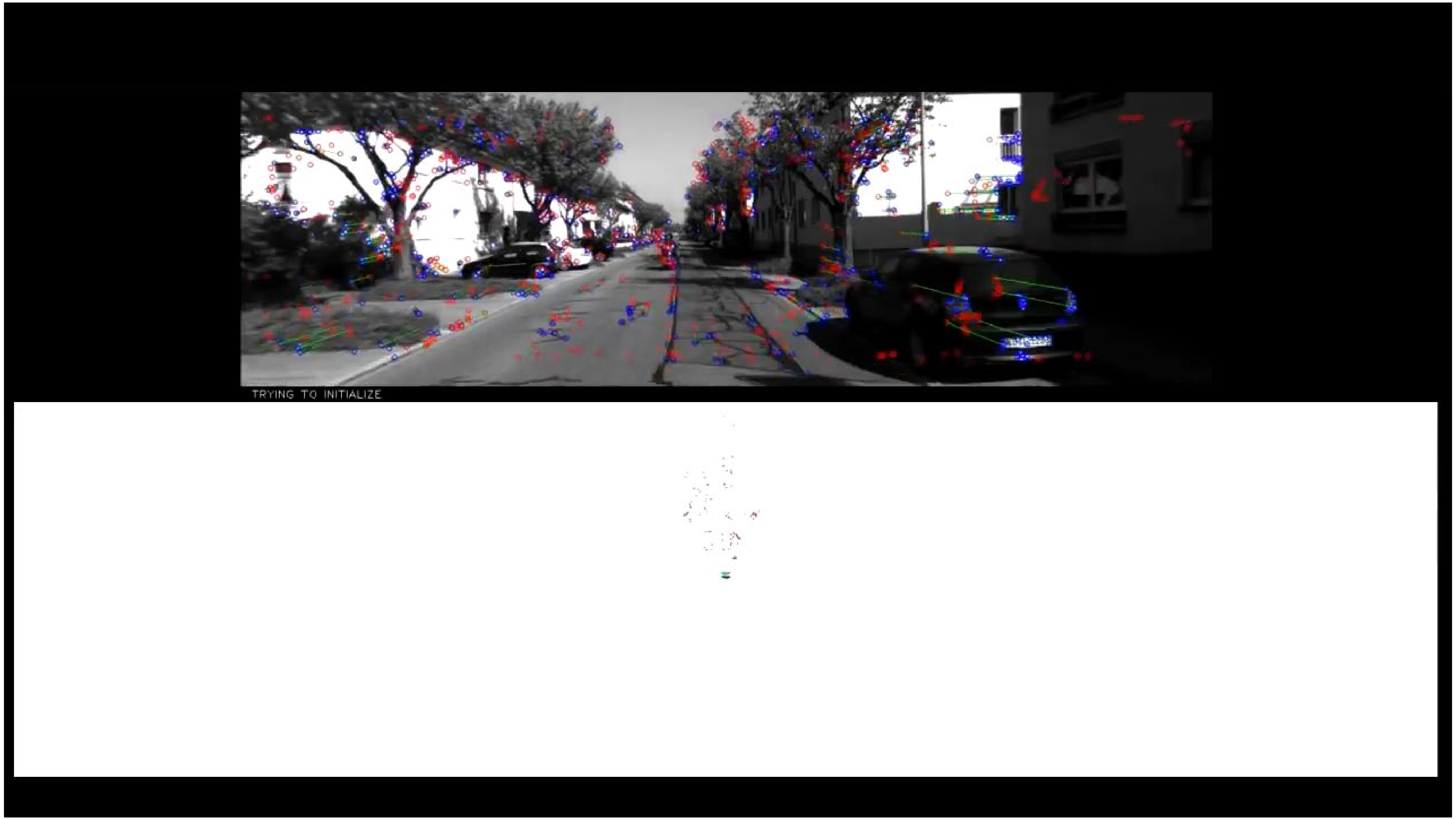
The uncertainty shrinks. The particles are resampled based on weights assigned to the previous particles.

Particle filter SLAM – Example: FastSLAM

SLAM - Summary

EKF SLAM	Graph-based SLAM	Particle Filter SLAM
• Pros • Can run online • Works for problems with perturbations	• Pros • Information can move backward in time • Best possible (most likely) estimate given the data and models	• Pros • Noise densities can be from any distribution • Works for multi-modal distributions • Easy to implement
• Cons • Unimodal estimate • States must be well approximated by a Gaussian • Computationally expensive for large-scale SLAM	• Cons • Computationally demanding • Difficult to provide the online estimates for a controller	• Cons • Does not scale to high-dimensional problems • Requires many particles to have good convergence

Keyframe-based SLAM – Example: ORB-SLAM on KITTI Dataset

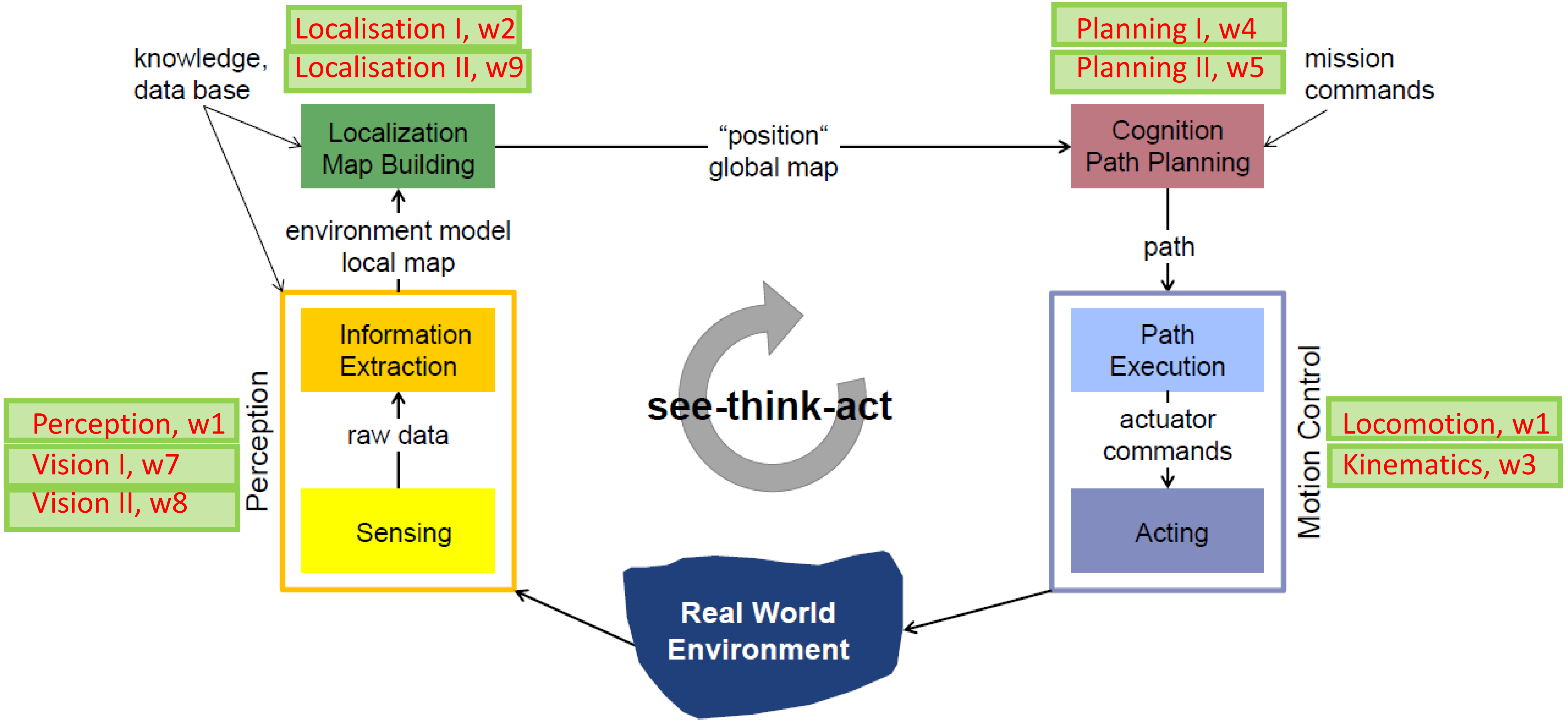


https://www.youtube.com/watch?v=8DISRmsO2YQ
http://www.cvlibs.net/datasets/kitti/

Think about

- What is Bayes rule? How can we use it to calculate conditional probability?
- What is the assumption used in Markov Localisation? What are the pros and cons of Markov Localisation?
- What are the differences between Particle Filter Localisation and Markov Localisation?
- What is Kalman Filter Localisation? How are the four ingredients used in the Kalman Filter Localisation? What are the pros and cons of Kalman Filter Localisation?
- What are the differences between Probabilistic Map-Based Localisation methods and SLAM methods?

The See-Think-Act cycle



Lecture 10: What’s next?

Welcome to provide your feedback.

https://app.sli.do/event/bqhayyzv

(under Q&A)