ROS-I Basic Training “Mobile Manipulation”

Localisation

M. Stuettgen, N. Limpert

Mobile Autonomous Systems and Cognitive Robotics Institute (MASCOR)

August 9, 2017

Overview

- Dealing with Uncertainty
- Pobabilistic Models:
  Hidden Markov Models and Bayes Filter
- Markov Localisation
- Importance Sampling
- Landmark-based Localisation
- Grid-Mapping

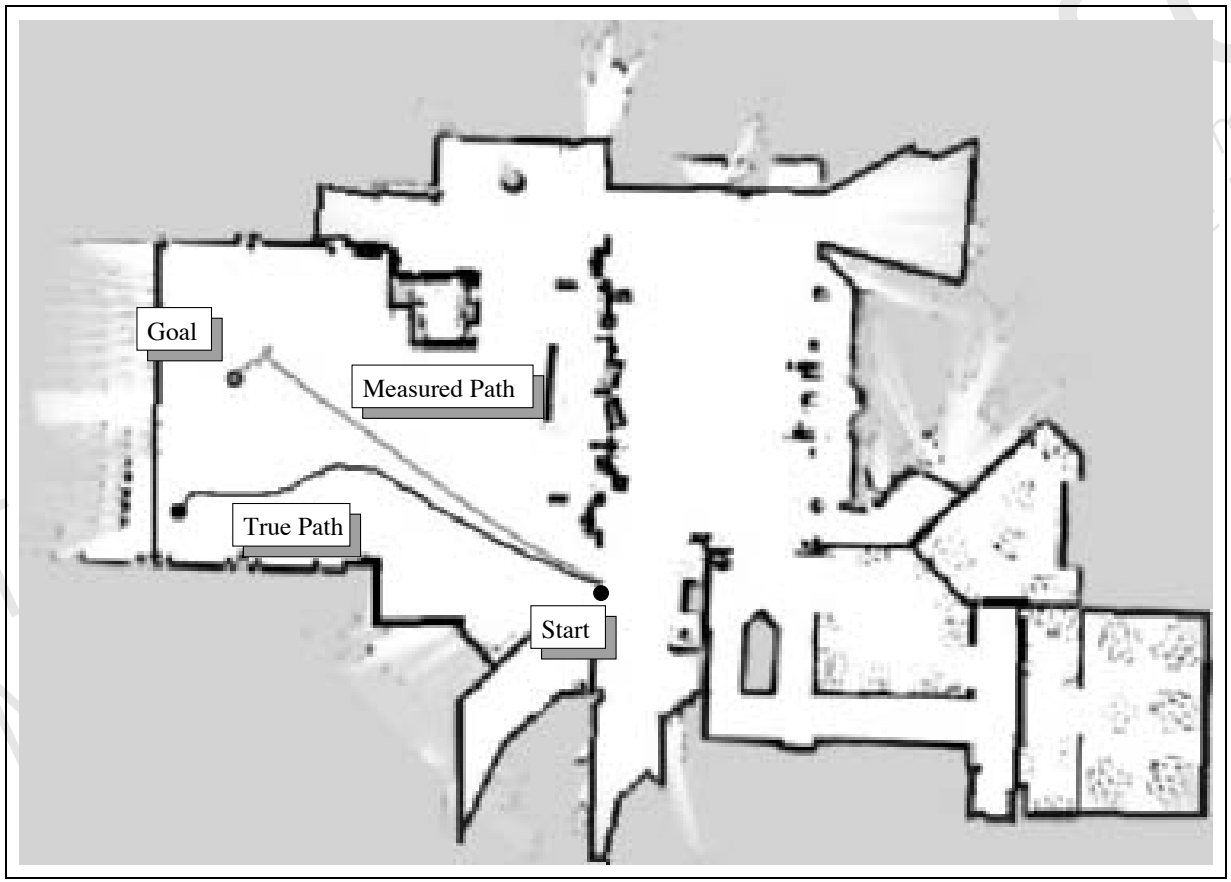
Learning Objectives

You will

- get to know the basic mathematical foundation underlying probabilistic localisation and mapping.
- learn how to infer the robot's position given a map and regular sensory updates.
- see how a map can be created from sensors input

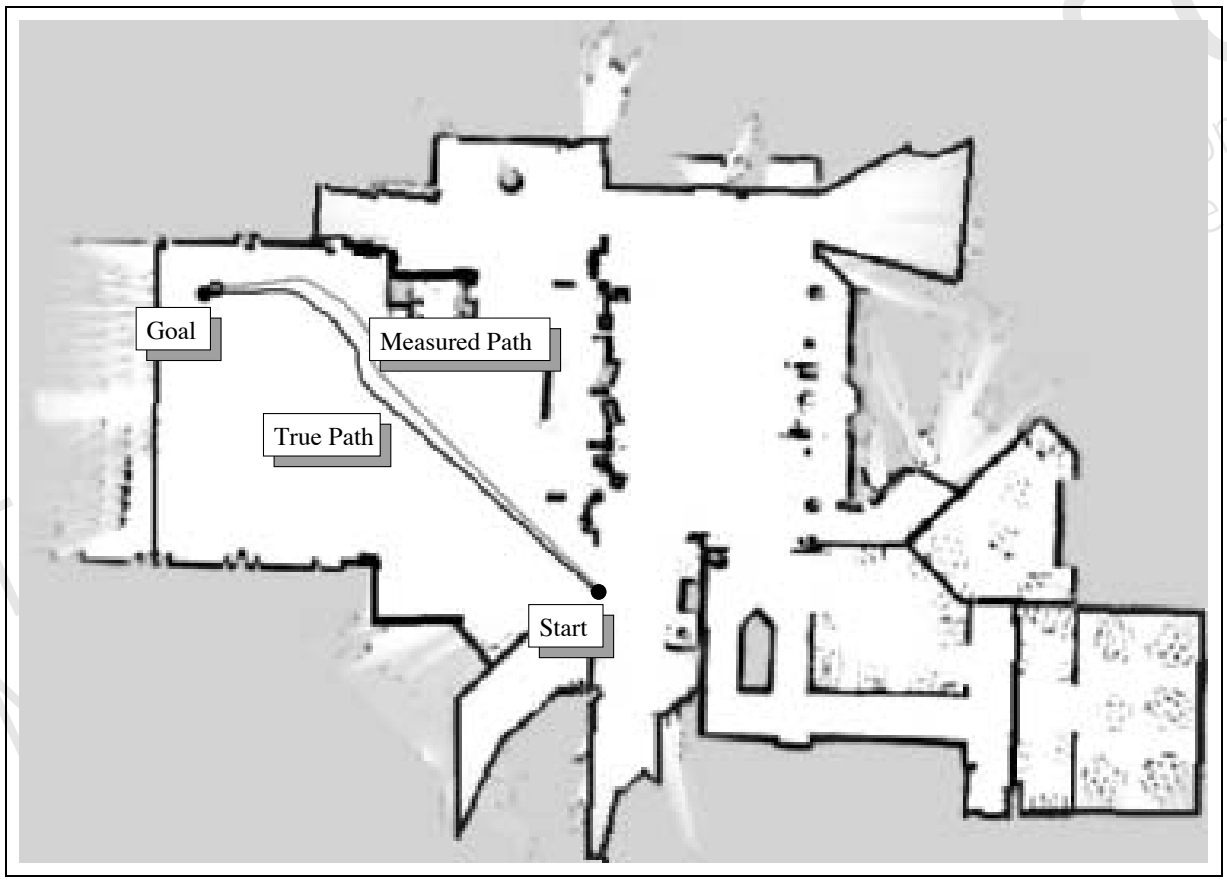
Outline

Dealing with Uncertainty



Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

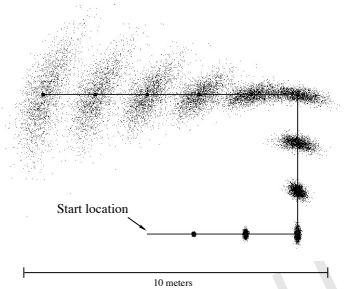
Dealing with Uncertainty



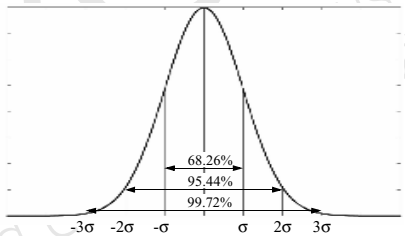
Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Dealing with Uncertainty

Uncertainty in Actuators



Uncertainty in Sensors



Recall:

Basic Questions

1. Where have I been already?
2. Where am I now?
3. Where am I going?
4. What's the best way to get there?

Problems:

- Localisation
- Mapping

Bayes Rule

Conditional Probabilities

$$P(o \mid s_1) = \frac{P(o \wedge s_1)}{P(s_1)} \qquad P(s_1 \mid o) = \frac{P(o \wedge s_1)}{P(o)}$$

$$P(o \wedge s_1) = P(s_1 \mid o) \cdot P(o)$$

Bayes Rule

Conditional Probabilities

$$P(o \mid s_1) = \frac{P(o \wedge s_1)}{P(s_1)} \qquad P(s_1 \mid o) = \frac{P(o \wedge s_1)}{P(o)}$$

$$P(o \wedge s_1) = P(s_1 \mid o) \cdot P(o)$$

$$\Rightarrow \quad P(o \mid s_1) = \frac{P(s_1 \mid o) \cdot P(o)}{P(s_1)}$$

Bayes Rule

Conditional Probabilities

$$P(o \mid s_1) = \frac{P(o \wedge s_1)}{P(s_1)} \qquad P(s_1 \mid o) = \frac{P(o \wedge s_1)}{P(o)}$$

$$P(o \wedge s_1) = P(s_1 \mid o) \cdot P(o)$$

$$\Rightarrow \quad P(o \mid s_1) = \frac{P(s_1 \mid o) \cdot P(o)}{P(s_1)}$$

can be computed in advance
(save in a table)

$\frac{1}{2}$
ignore for now

Bayes Filter

```
bayesfilter(Bel(x_{t-1}), z_t, u_t)
begin
    forall the x_t do
        // 1.  Prediction Step
        bel̄(x_t) = ∫ p(x_t|u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}
        // 2.  Correction Step
        bel(x_t) = η p(z_t|x_t) bel̄(x_t)
    end
    return  Bel(x_t)
end
```

Bayes Filter

```
bayesfilter(Bel(x_{t-1}), z_t, u_t)
begin
    forall the x_t do
        // 1.  Prediction Step
        bel̄(x_t) = ∫ p(x_t|u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}
        // 2.  Correction Step
        bel(x_t) = η p(z_t|x_t) bel̄(x_t)
    end
    return  Bel(x_t)
end
```

Bayes Filter

bayesfilter($Bel(x_{t-1})$, z_t, u_t)
begin
  forall the x_t **do**
    // 1. Prediction Step

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

    // 2. Correction Step

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

  end
  return $Bel(x_t)$
end

$$bel(x_t) = \eta \underbrace{p(z_t|x_t)}_{\text{Sensor model}} \overline{bel}(x_t)$$

Bayes Filter

bayesfilter($Bel(x_{t-1})$, z_t, u_t)
begin
  forall the x_t **do**
    // 1. Prediction Step

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

    // 2. Correction Step

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

  end
  return $Bel(x_t)$
end

$$bel(x_t) = \eta \underbrace{p(z_t|x_t)}_{\text{Sensor model}} \overline{bel}(x_t)$$

$$\eta = \left[\sum_{x_k} p(z_t|x_k) p(x_k|z_1, u_1, \ldots, z_{t-1}, u_{t-1})\right]^{-1}$$

Outline

Applying Bayes Rule

o : cell x, y is occupied (of position l in map m)

s_1, s_2 : sensor readings

$$P(o \mid s_1) = \frac{P(s_1 \mid o) \cdot P(o)}{P(s_1)}$$

can be pre-computed in table!

$\frac{1}{2}$
ignore for now

Applying Bayes Rule

o : cell x, y is occupied (of position l in map m)

s_1, s_2 : sensor readings

more difficult for 2 sensor readings:

$$P(o \mid s_2, s_1) = \frac{P(s_2 \mid o, s_1) \cdot P(o \mid s_1)}{P(s_2 \mid s_1)}$$

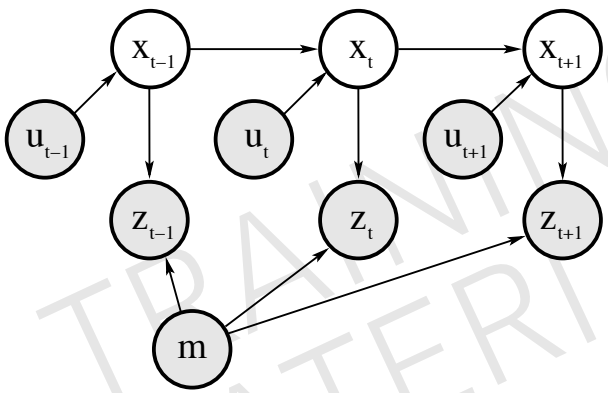
not practicable for $T(= 100)$ sensor readings ??

Important Assumption:

$$P(s_2 \mid o, s_1) = P(s_2 \mid o)$$

why ok? → later

Markov Localisation



Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Integration of a Map

- in sensor model:

$$p(z_t|x_t, m)$$

- in motion model:

$$p(x_t|u_t, x_{t-1}, m)$$

Algorithm: Markov Localisation

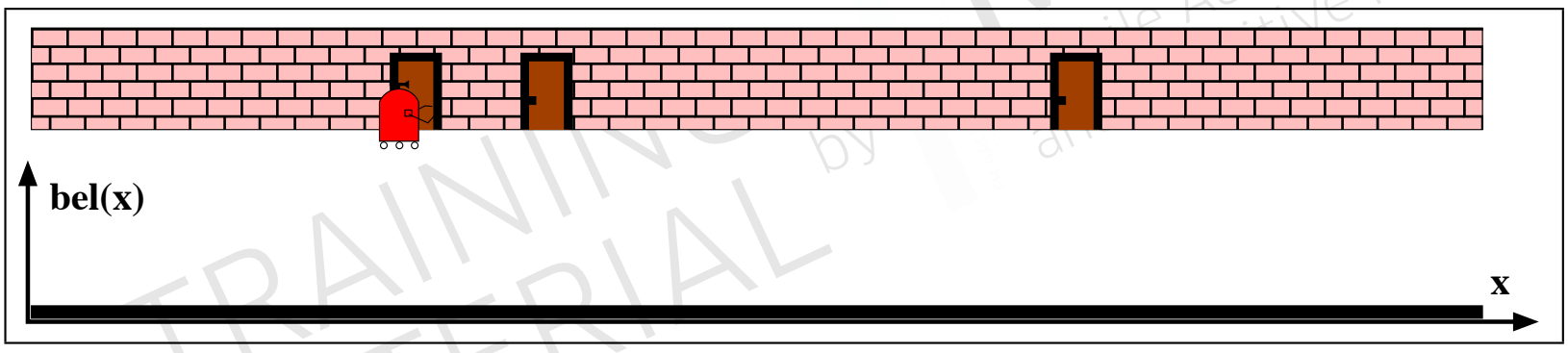
Markov_localization($bel(x_{t-1})$, z_t, u_t)
begin
  forall the x_t **do**

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

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

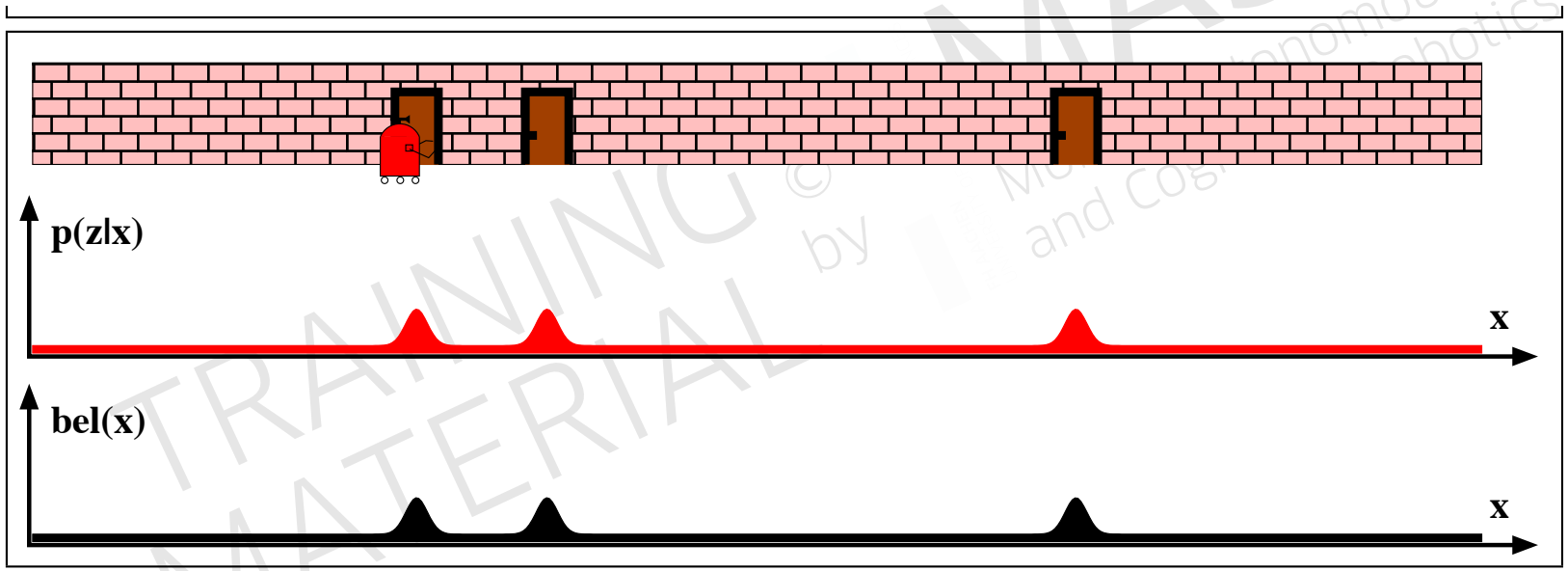
  end
  return $bel(x_t)$
end

Markov Localisation



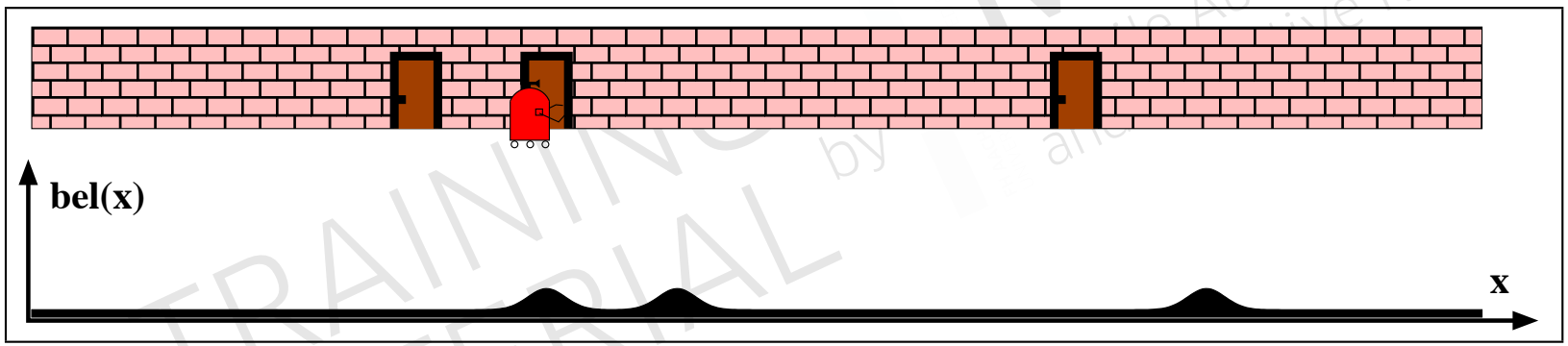
Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Markov Localisation



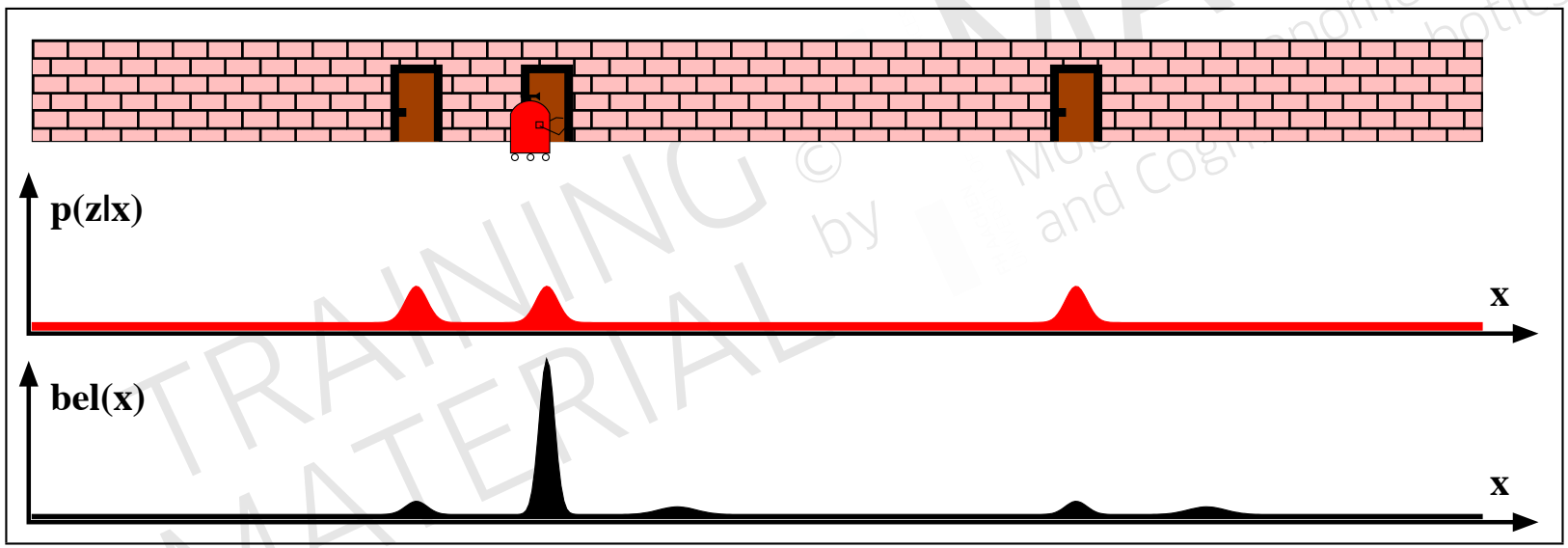
Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Markov Localisation



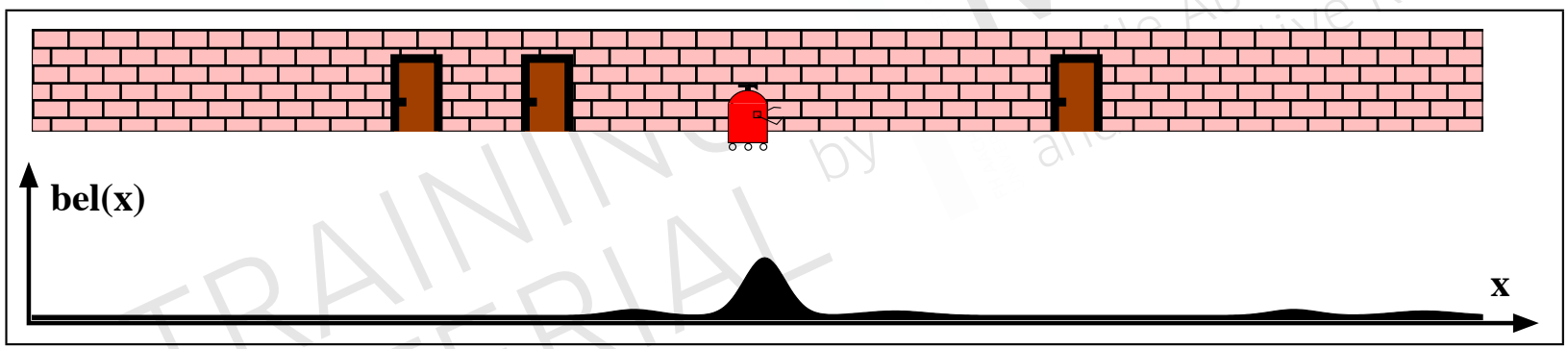
Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Markov Localisation



Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Markov Localisation



Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Grid Localisation

Idea:

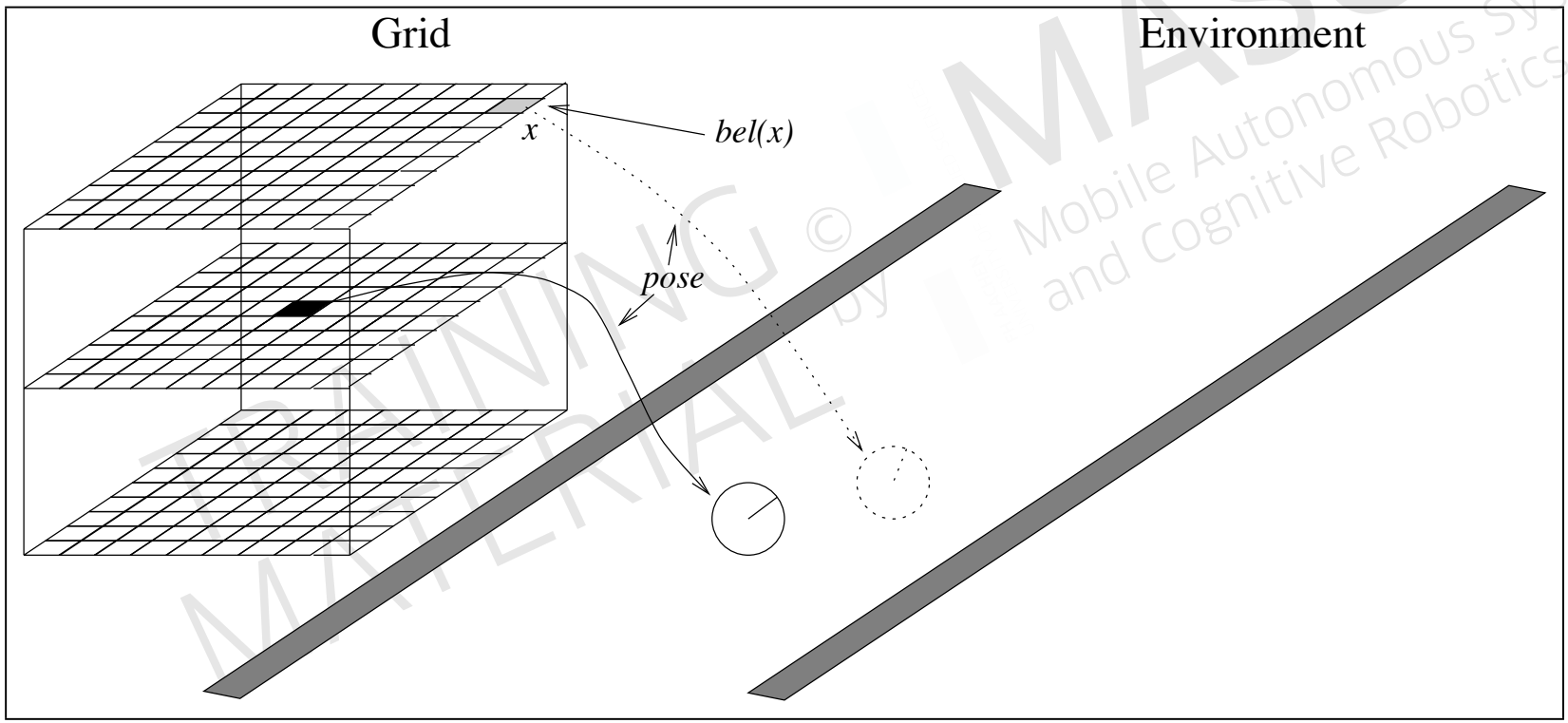
Approximating the a posteriori probabilities by histogram filters (discrete Bayes filter) over cell decomposition of the state space.

The belief is approximated by $bel(x_t) = \{p_{k,t}\}$

```
Discrete_Bayes_filter({p_{k,t-1}}, u_t, z_t)
begin
    forall the k do
        p̄_{k,t} = Σ_i p(X_t = x_k | u_t, X_{t-1} = x_t) p_{i,t-1}

        p_{k,t} = η p(z_t | X_t = x_k) p̄_{k,t}
    end
    return {p_{k,t}}
end
```

Representing the Probability Distribution



Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

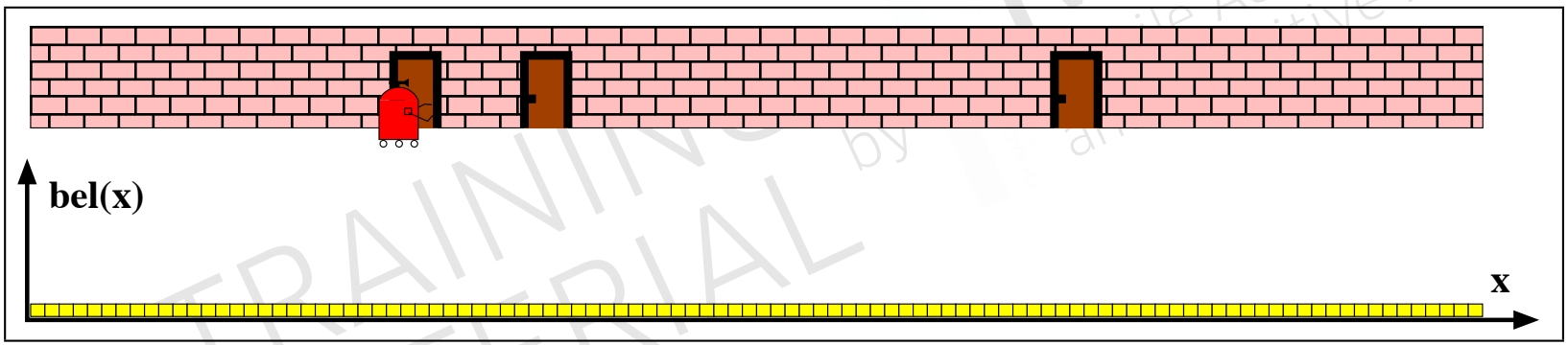
Algorithm: Grid Localisation

```
Grid_Bayes_filter({p_{k,t-1}, u_t, z_t, m)
begin
    forall the k do
        p̄_{k,t} = Σ_i p_{i,t-1} motion_model(mean(x_k), u_t, mean(x_i))

        p_{k,t} = η p̄_{k,t} sensor_model(z_t, mean(x_k), m)
    end
    return {p_{k,t}}
end
```

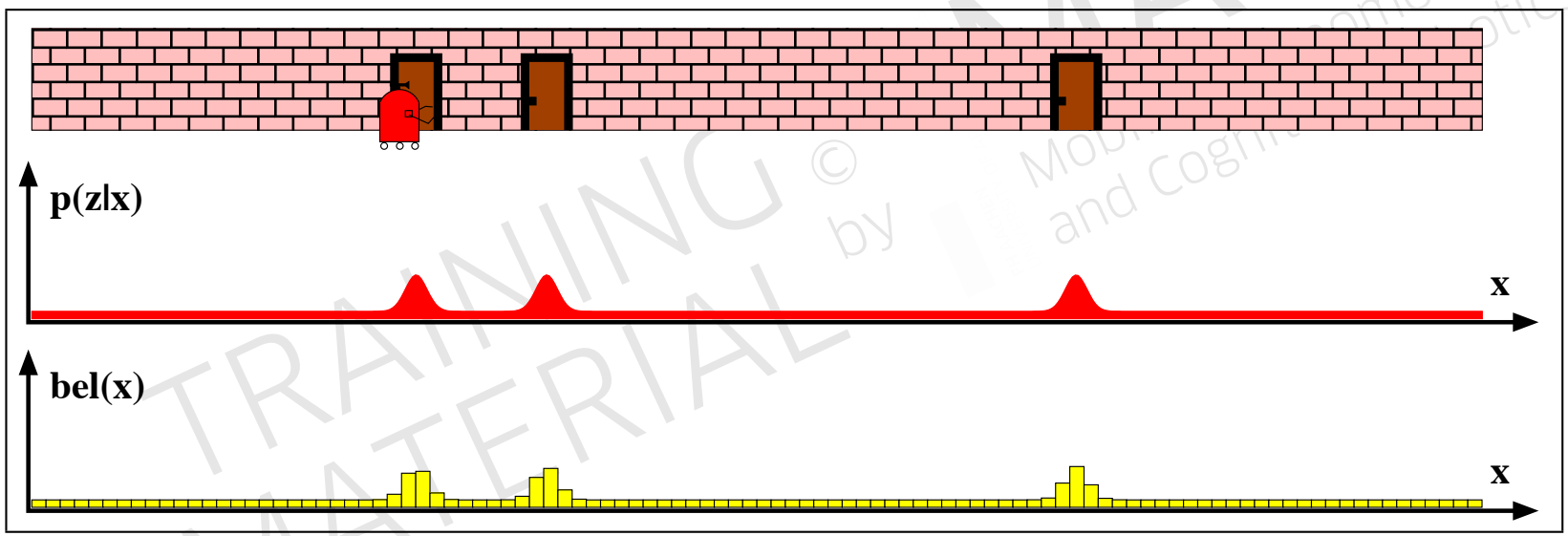
mean yields centre of mass of a grid cell.

Grid Localisation



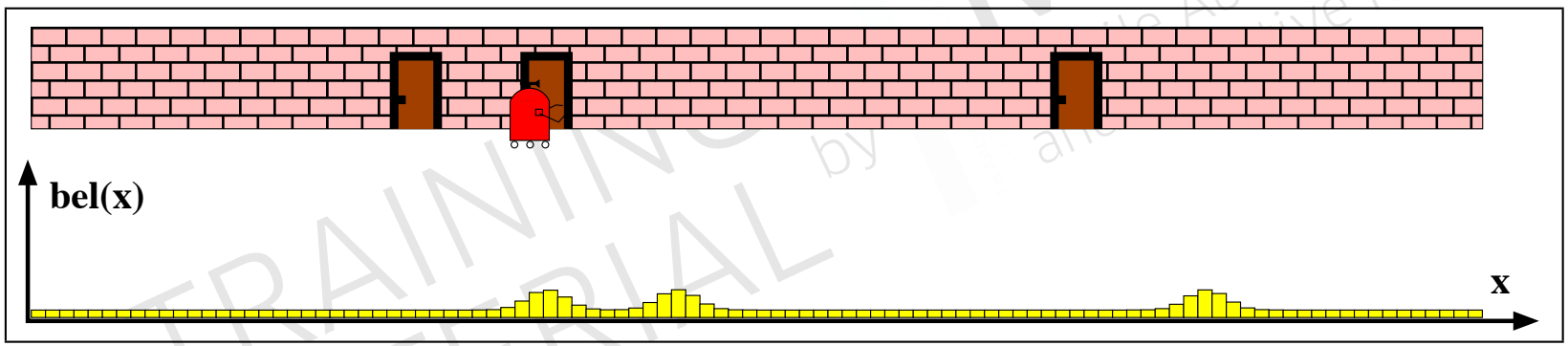
Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Grid Localisation



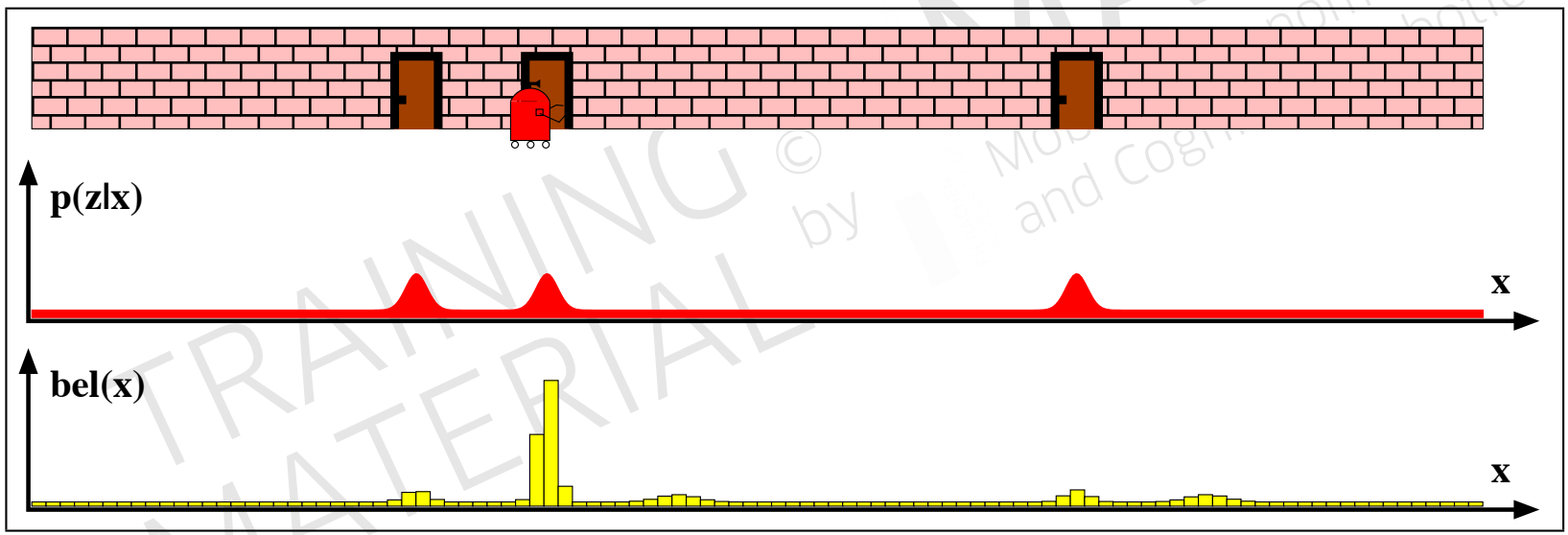
Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Grid Localisation



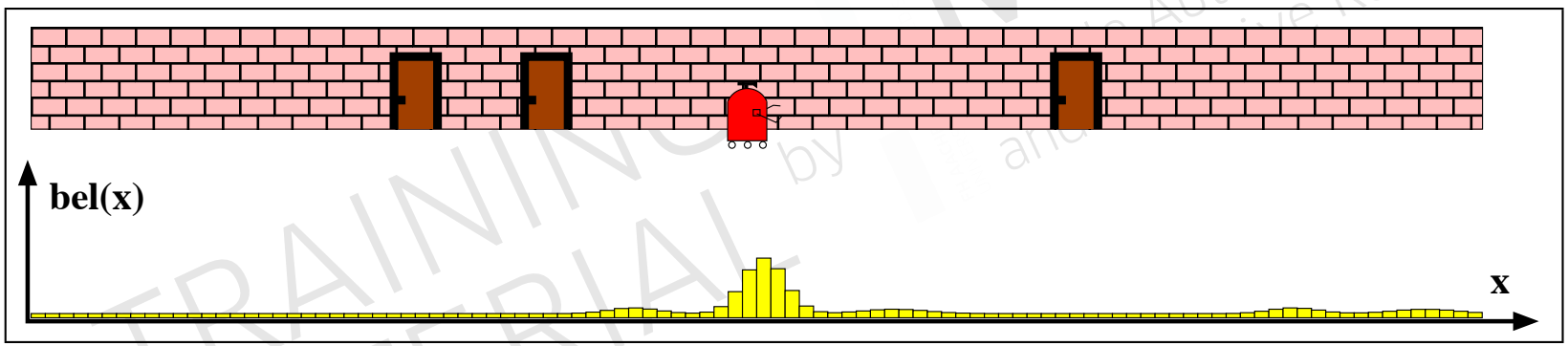
Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Grid Localisation



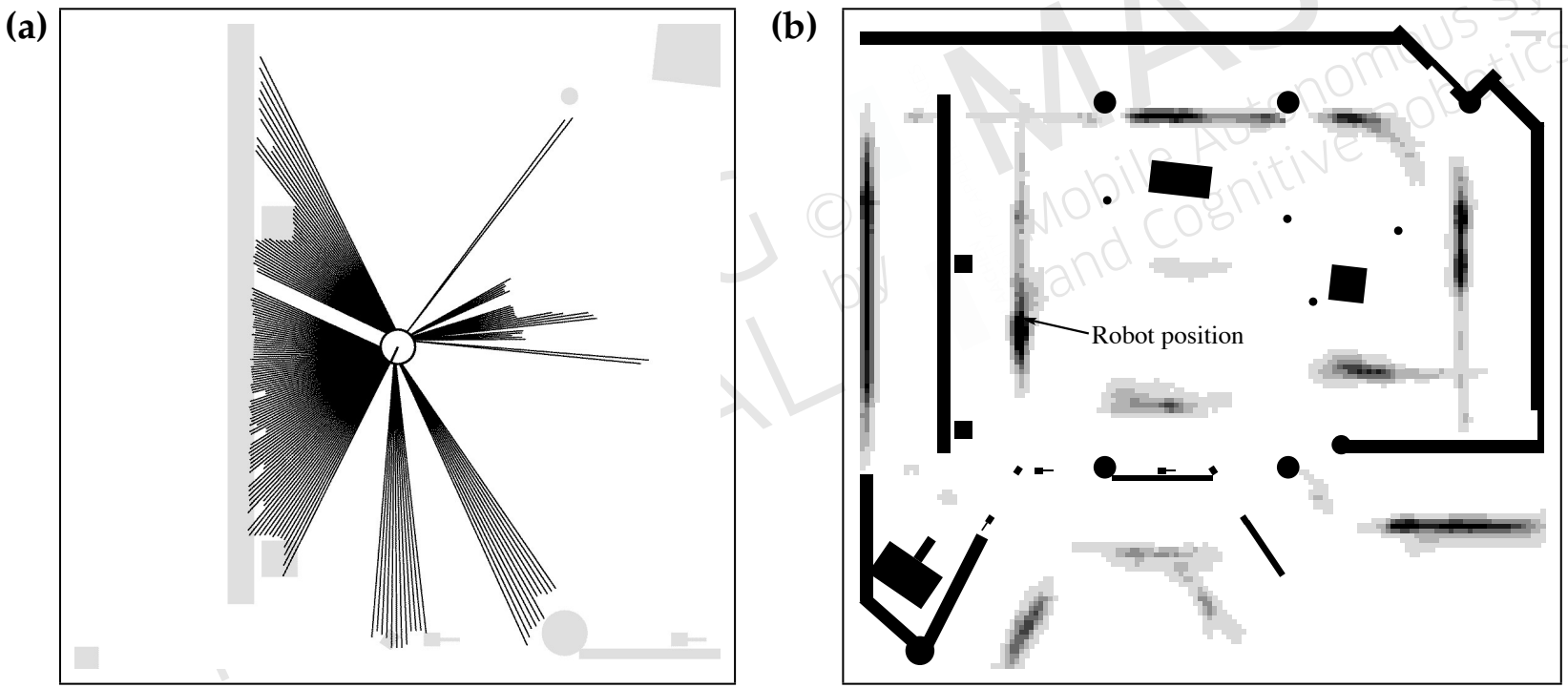
Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Grid Localisation



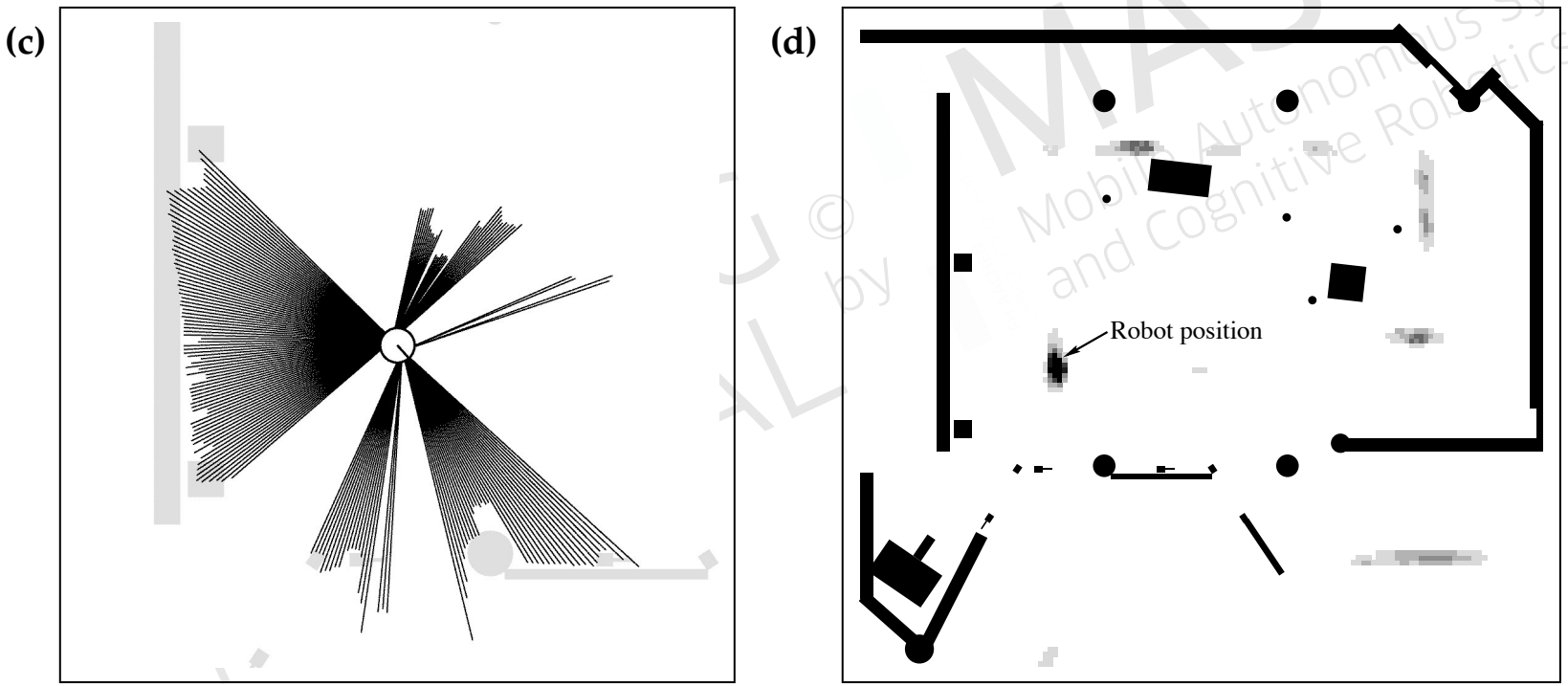
Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Localisation with LRF



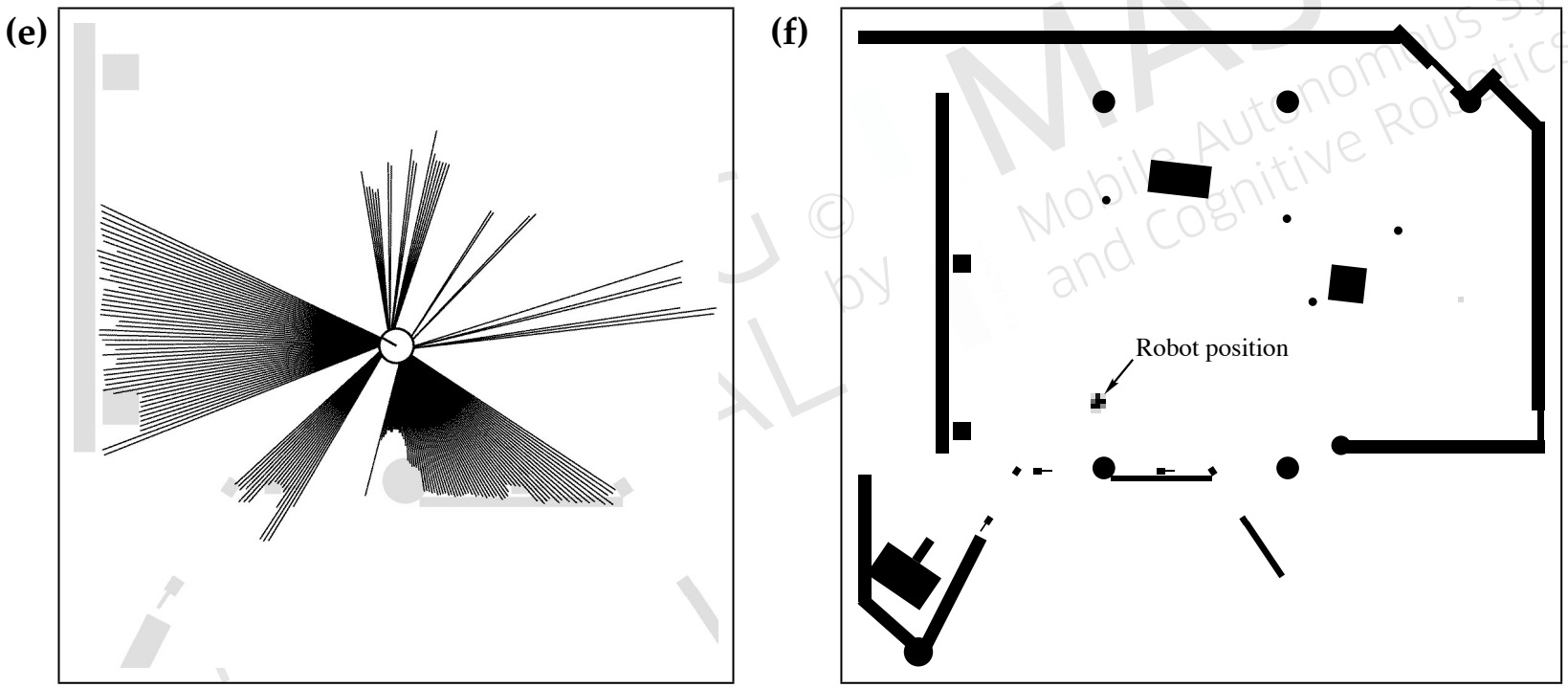
Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Localisation with LRF



Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

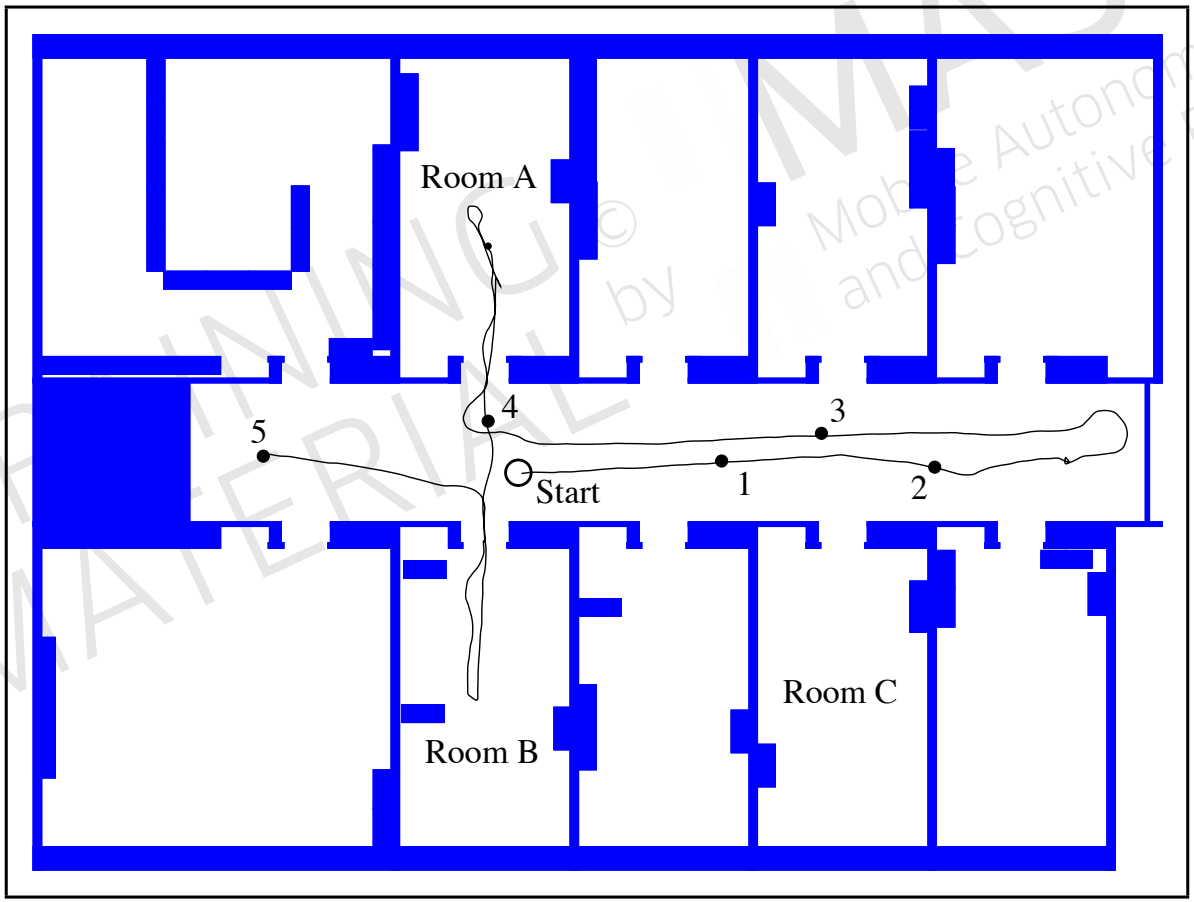
Localisation with LRF



Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Lokalisation with Sonar (1)

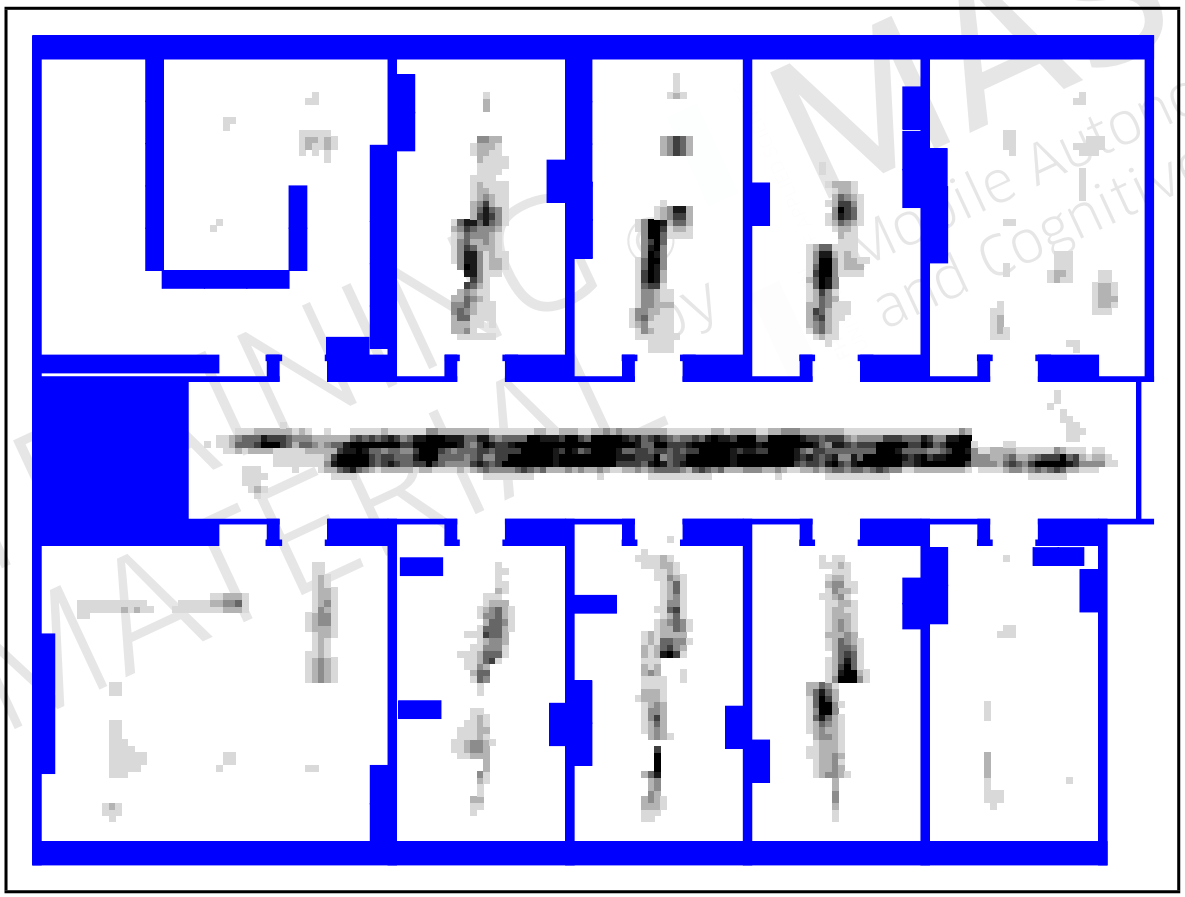
(a) Path and reference poses



Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Lokalisation with Sonar (1)

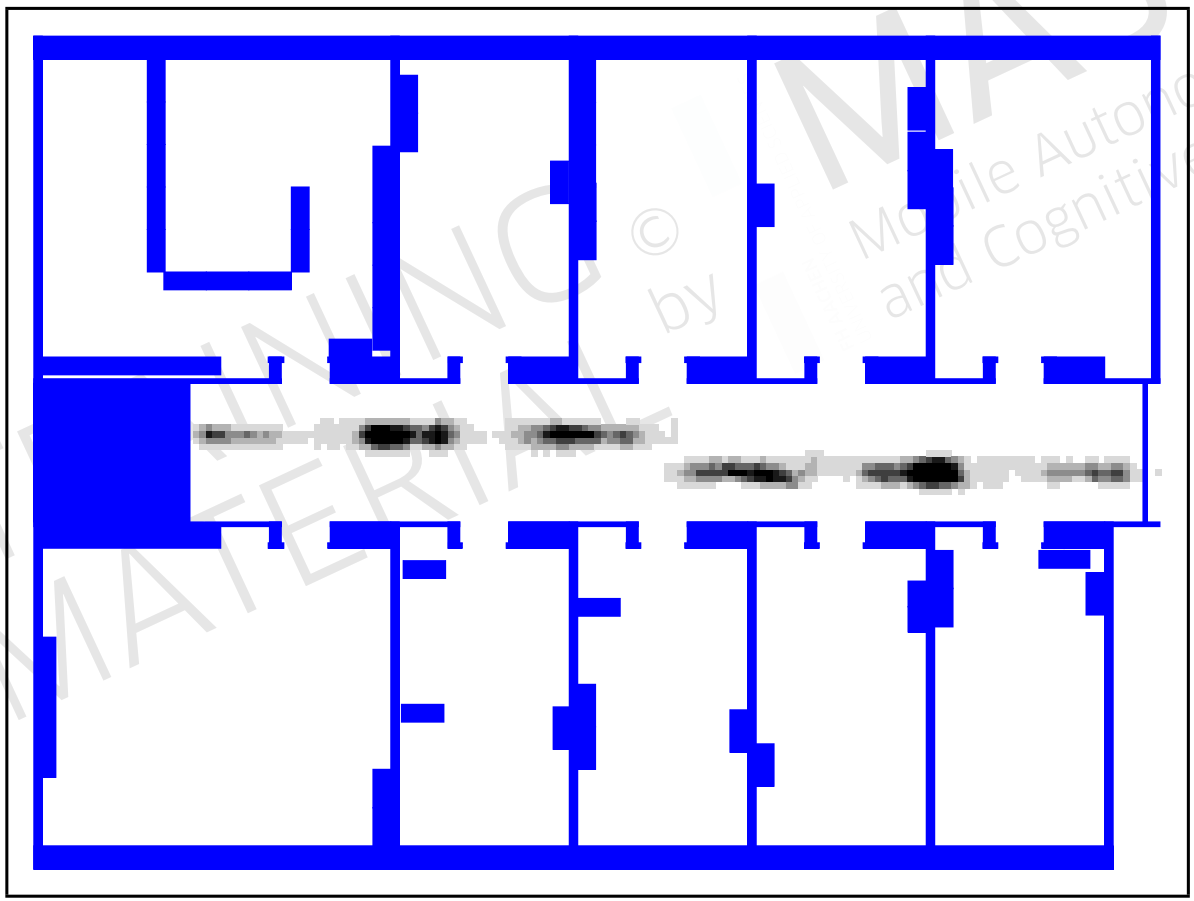
(b) Belief at reference pose 1



Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Lokalisation with Sonar (1)

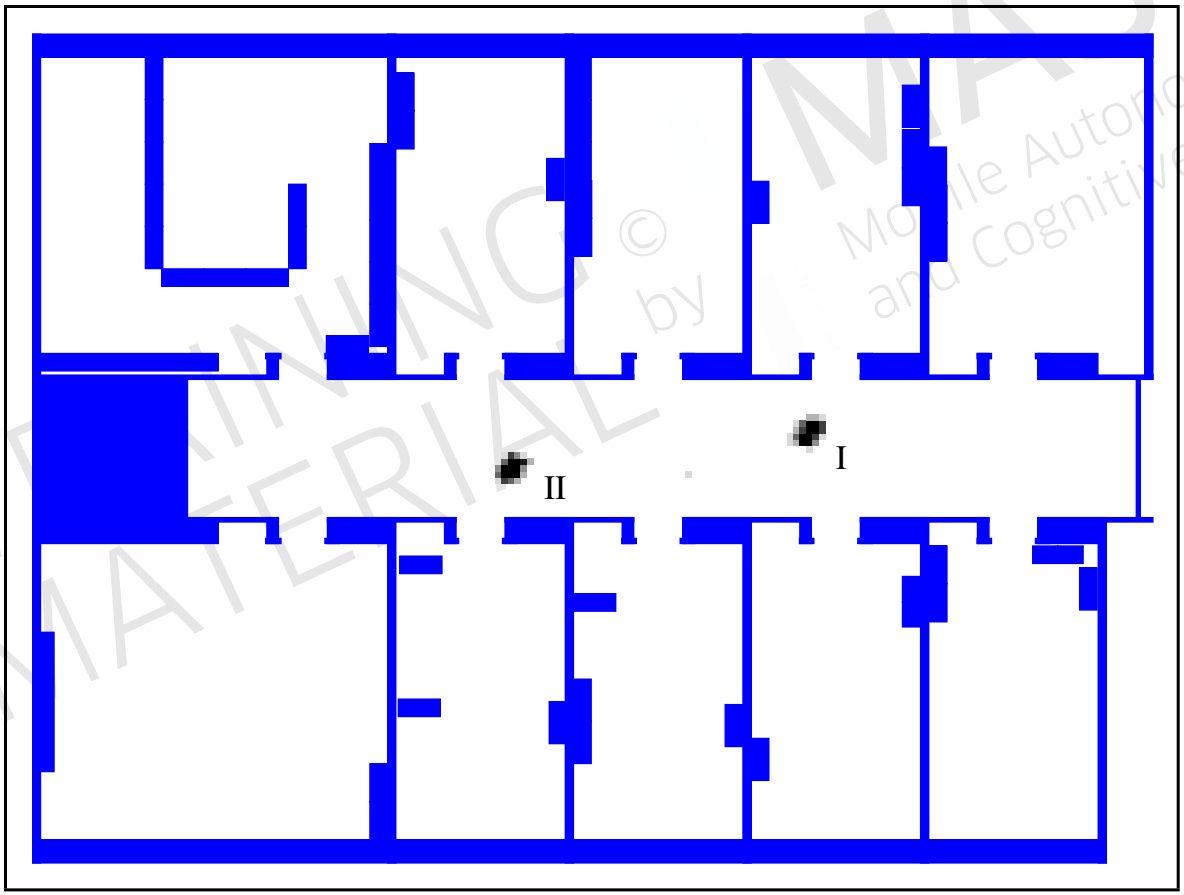
(c) Belief at reference pose 2



Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Lokalisation with Sonar (1)

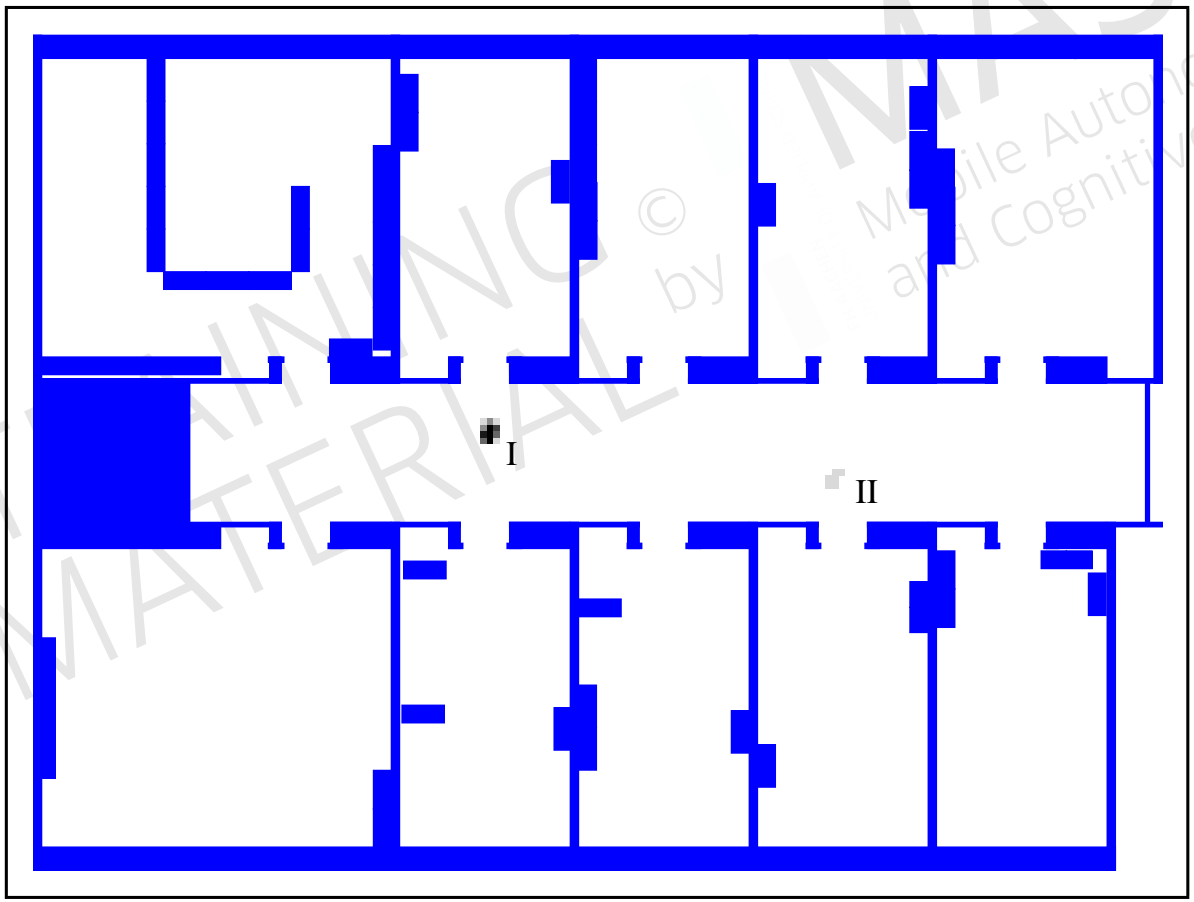
(d) Belief at reference pose 3



Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Lokalisation with Sonar (1)

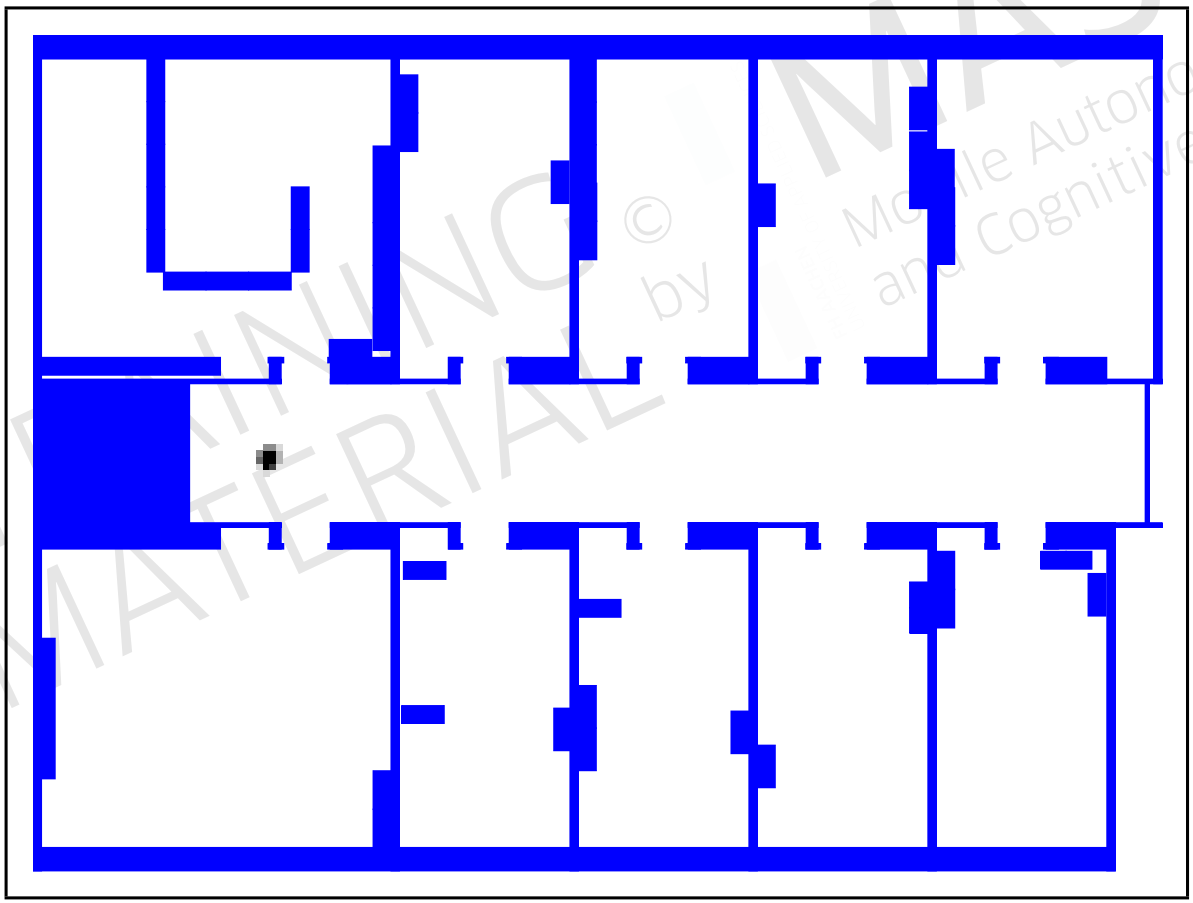
(e) Belief at reference pose 4



Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

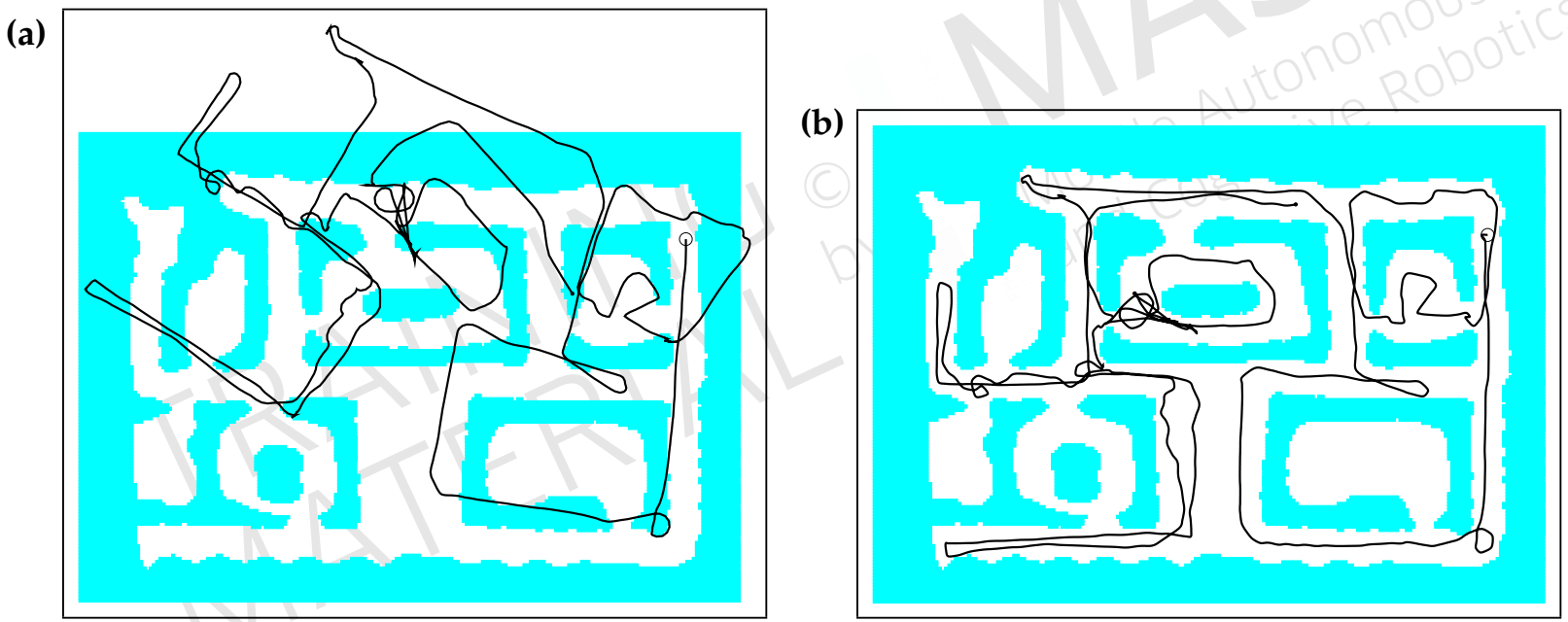
Lokalisation with Sonar (1)

(f) Belief at reference pose 5



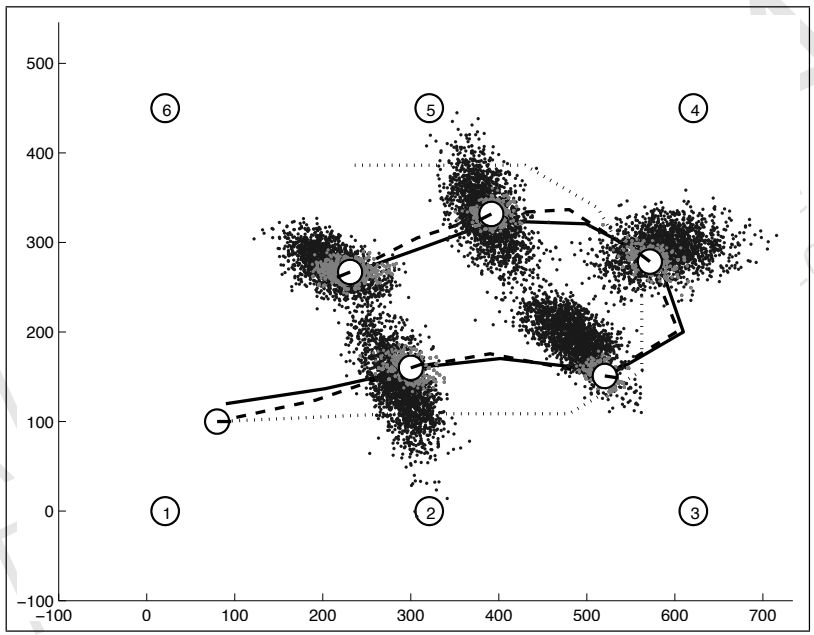
Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Dead Reckoning vs Grid Localisation



Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

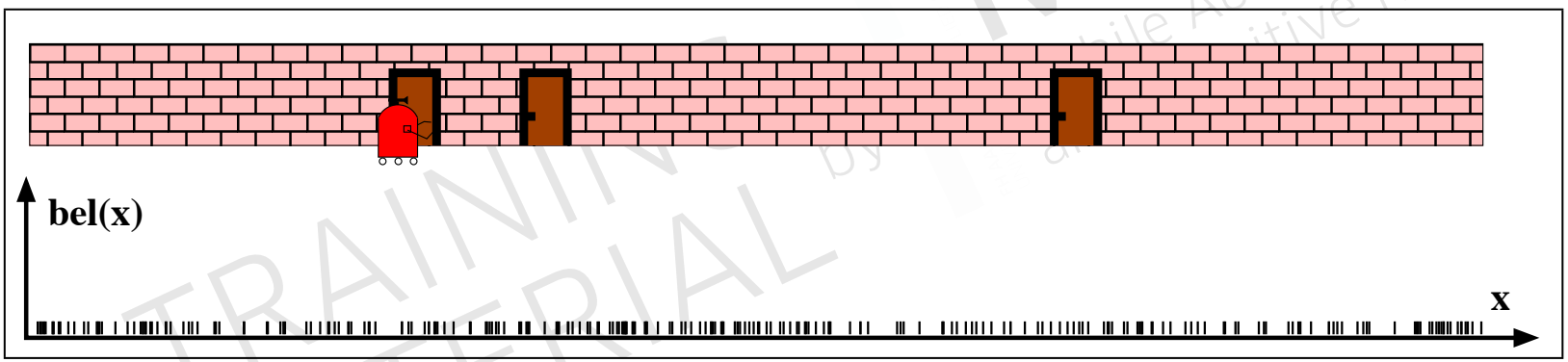
Monte Carlo Localisation



Idea:

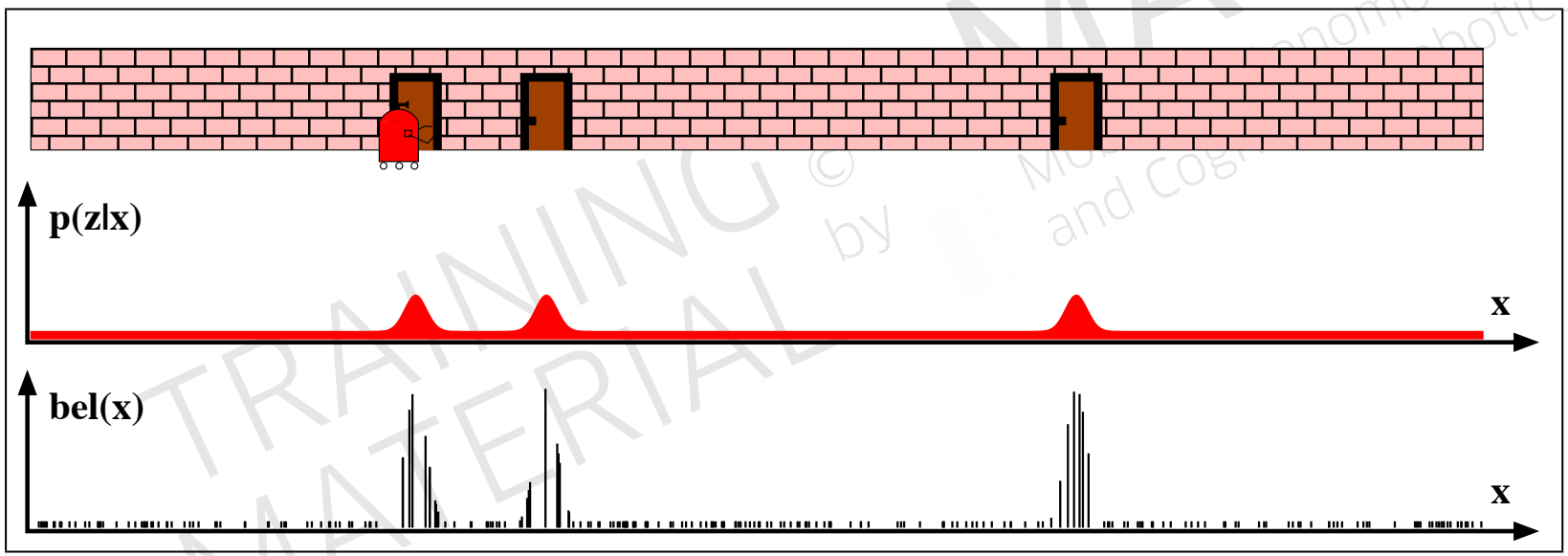
The a posteriori distribution $bel(x_t)$ is represented by a set of samples randomly drawn from this distribution.

Monte Carlo Localisation



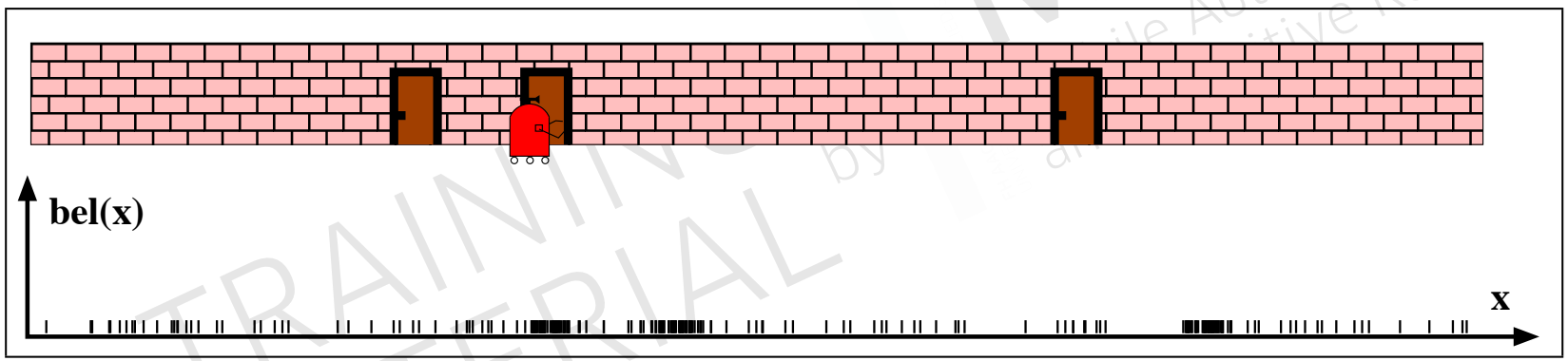
Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Monte Carlo Localisation



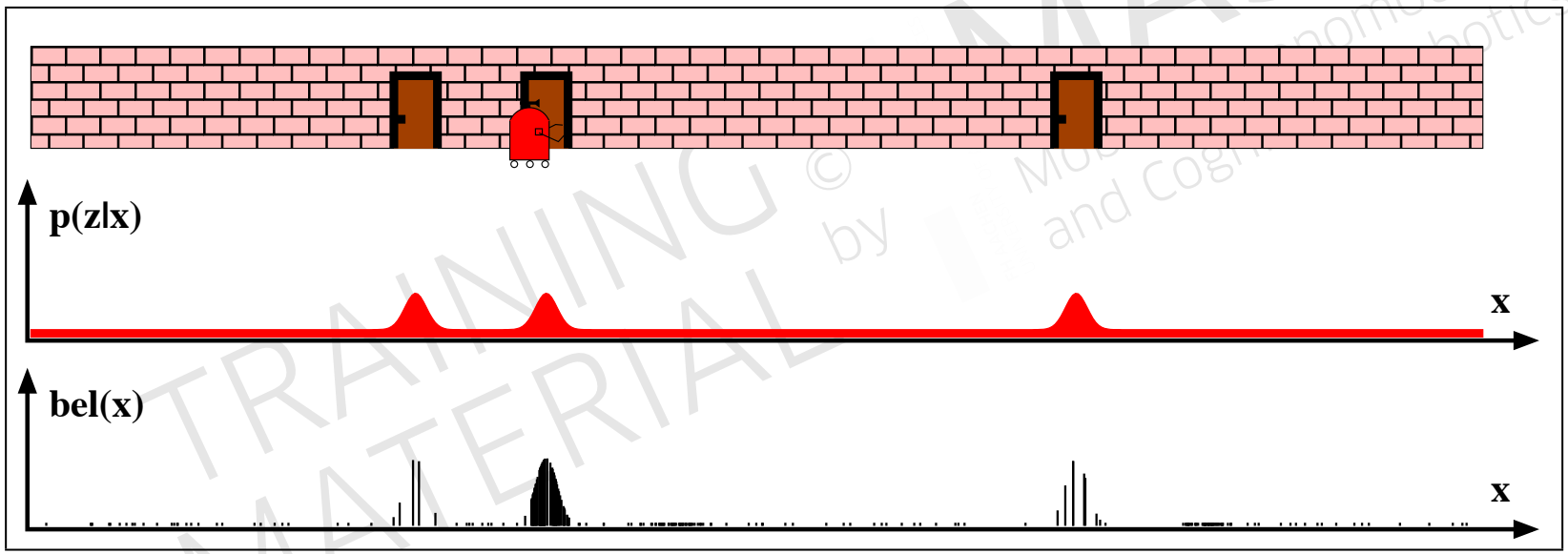
Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Monte Carlo Localisation



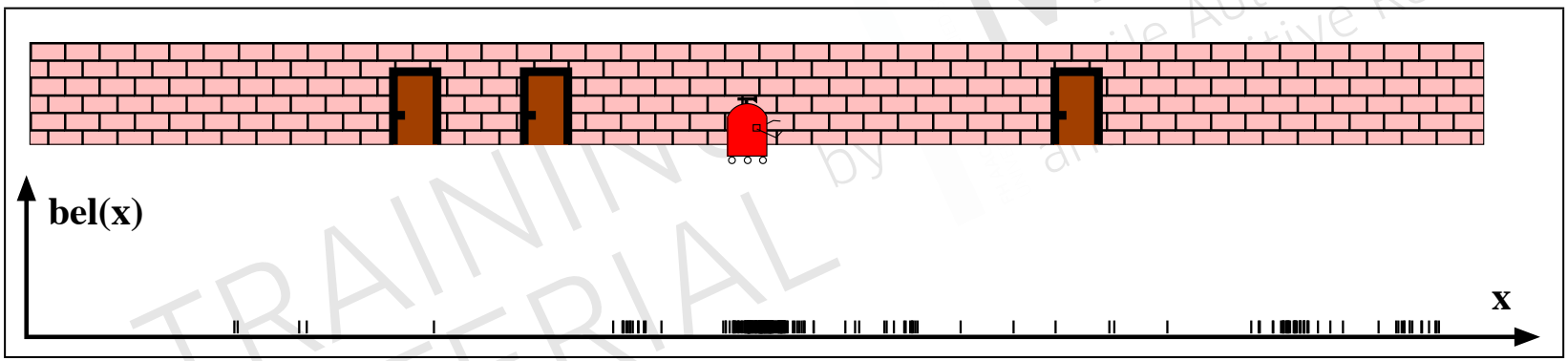
Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Monte Carlo Localisation



Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Monte Carlo Localisation



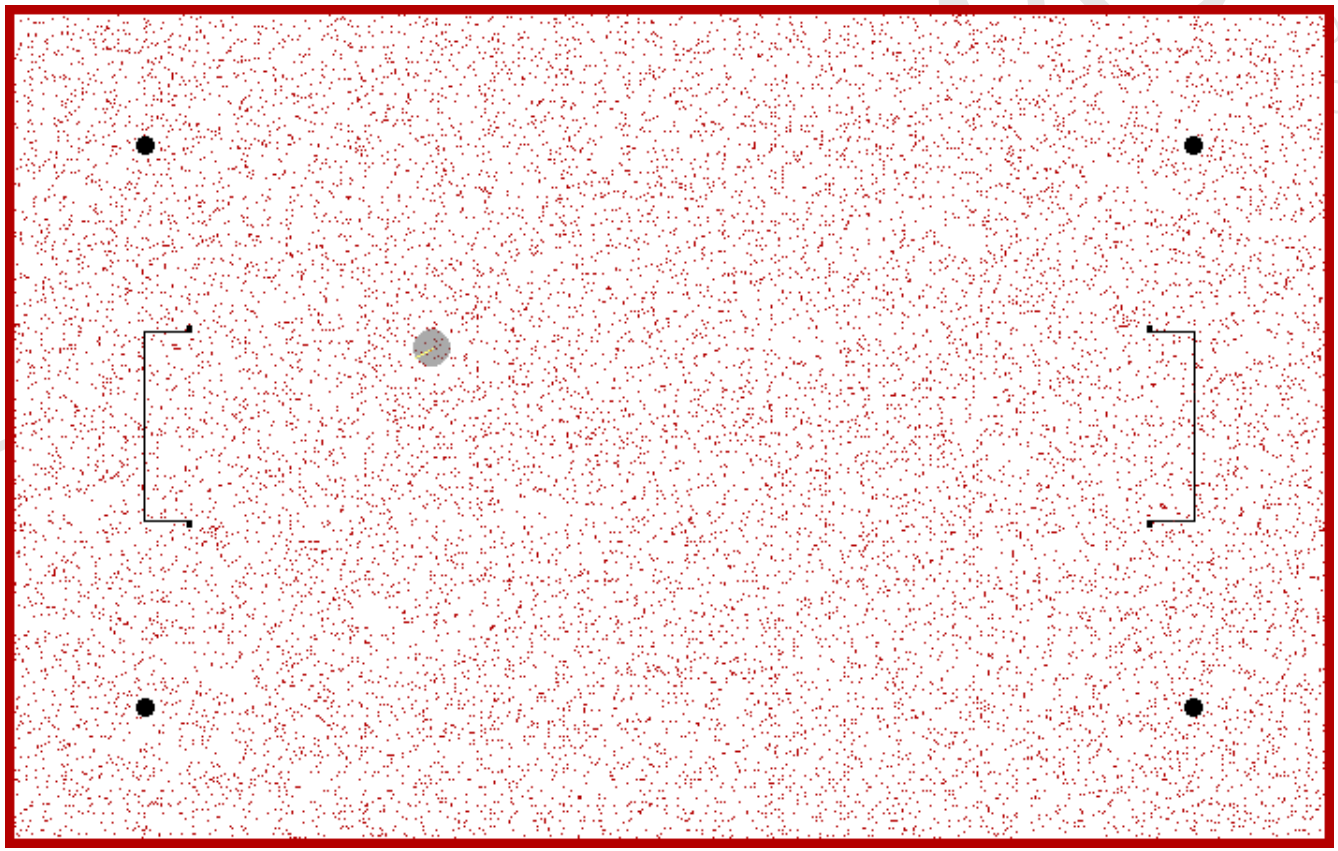
Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Algorithm: Monte-Carlo-Localisation

MCL($\mathcal{X}_{t-1}$, u_t, z_t, m)
begin
  $\overline{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
  for $m = 1$ *to* M **do**
    $x_t^{[m]} =$ sample_motion_model$(u_t, x_{t-1}^{[m]})$
    $w_t^{[m]} =$measurement_model$(z_t, x_t^{[m]}, m)$
    $\overline{\mathcal{X}}_t = \overline{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
  end
  for $m = 1$ *to* M **do**
    draw i with probability $\propto w_t^{[i]}$
    add $x_t^{[i]}$ to $\mathcal{X}_t$
  end
  return $\mathcal{X}_t$
end

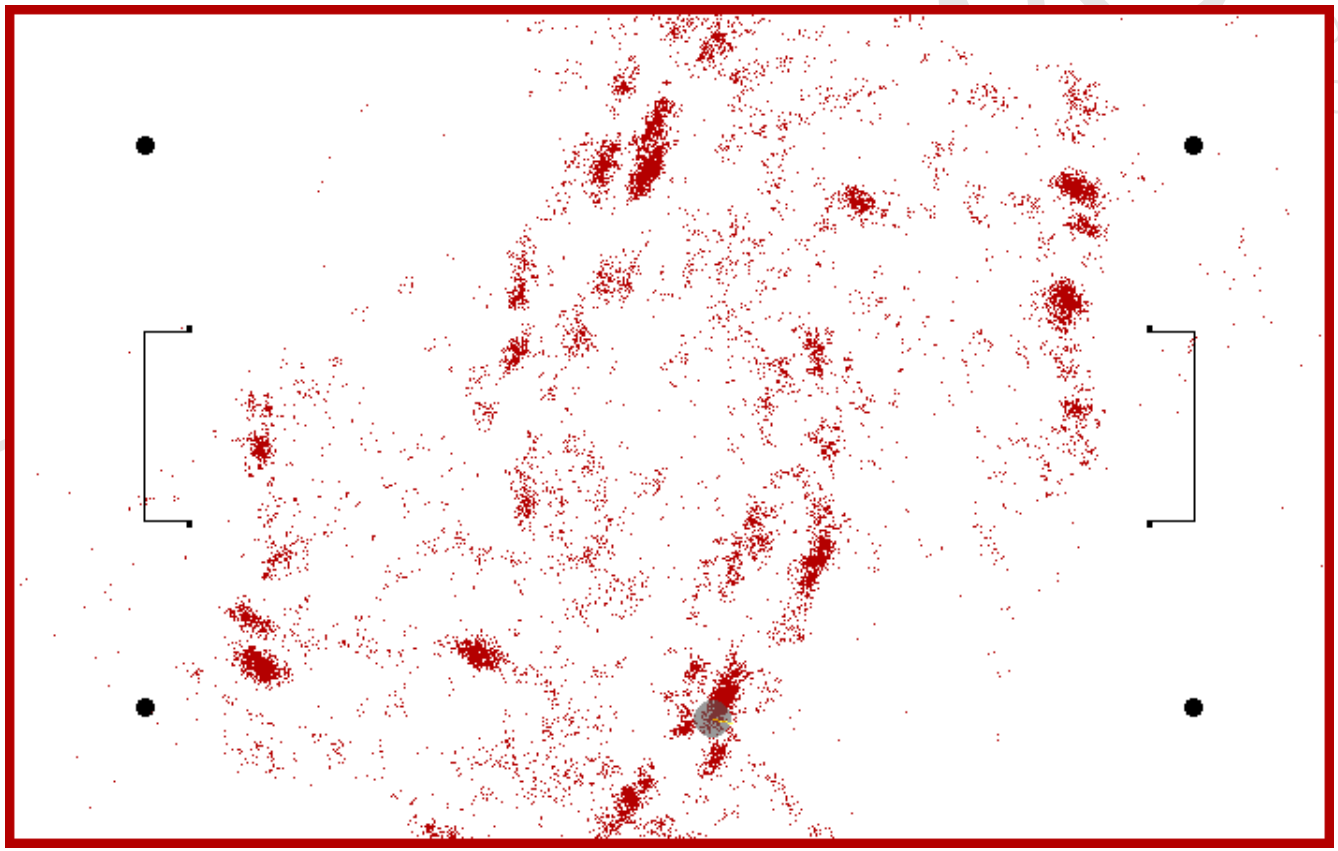
Monte-Carlo-Localisation, Example 1

(Strack, Ferrein, Lakemeyer 2005)



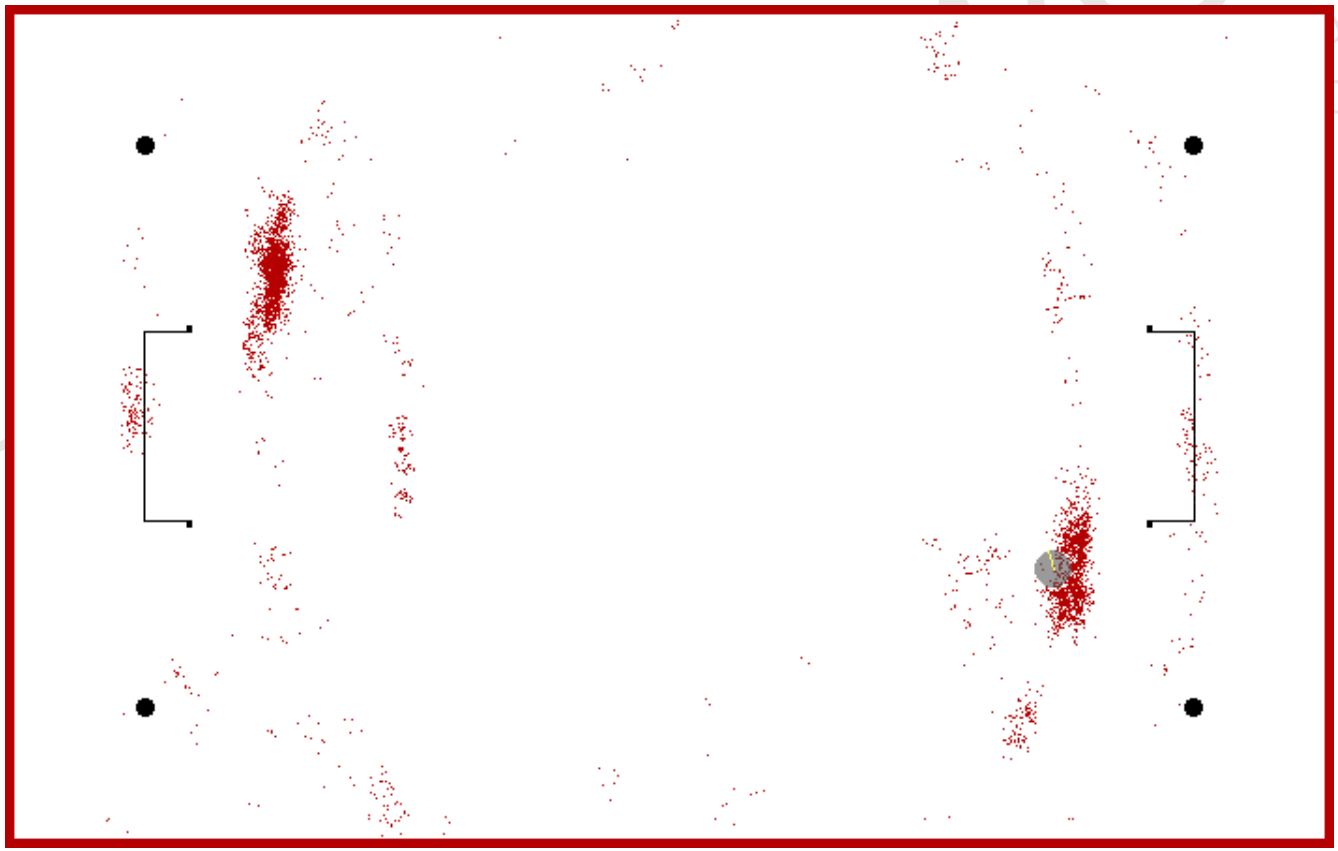
Monte-Carlo-Localisation, Example 1

(Strack, Ferrein, Lakemeyer 2005)



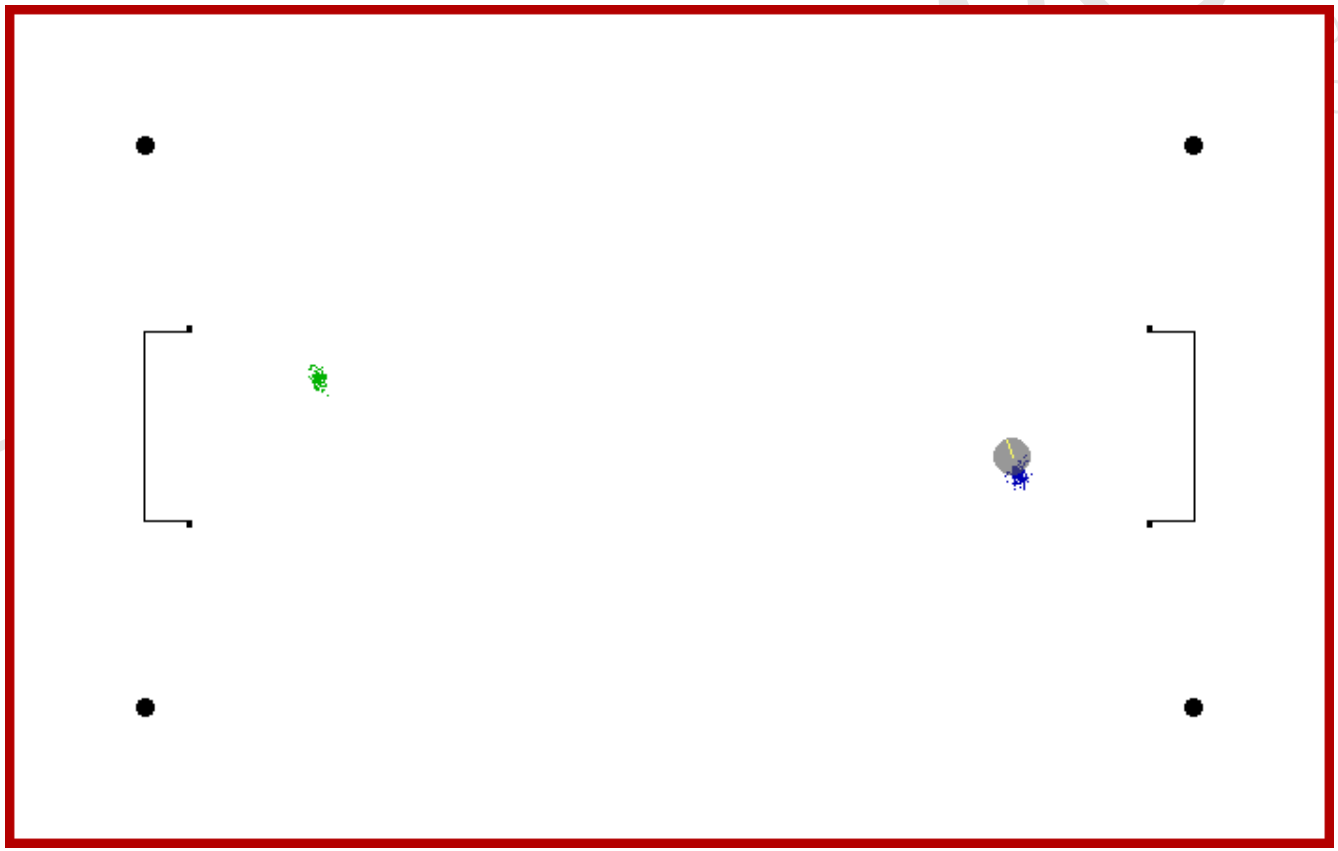
Monte-Carlo-Localisation, Example 1

(Strack, Ferrein, Lakemeyer 2005)

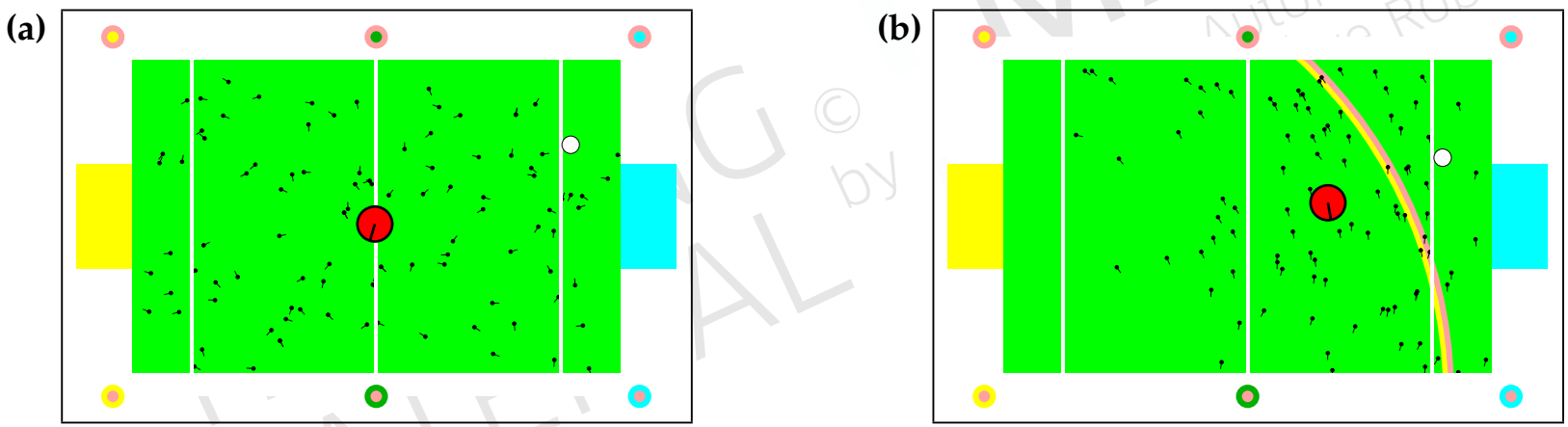


Monte-Carlo-Localisation, Example 1

(Strack, Ferrein, Lakemeyer 2005)

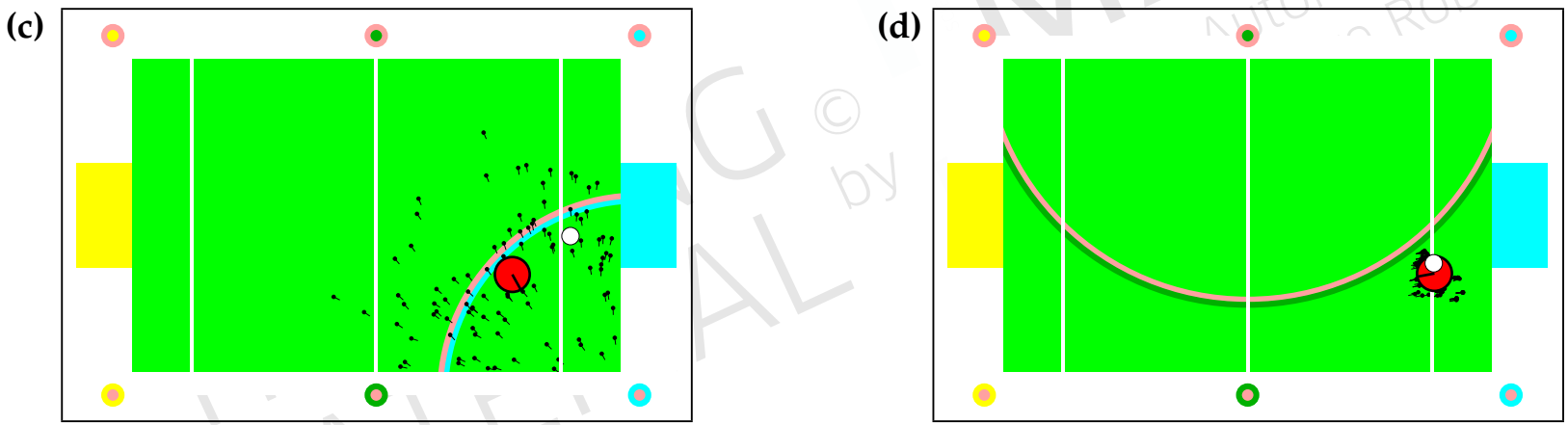


Monte-Carlo-Localisation, Example 2



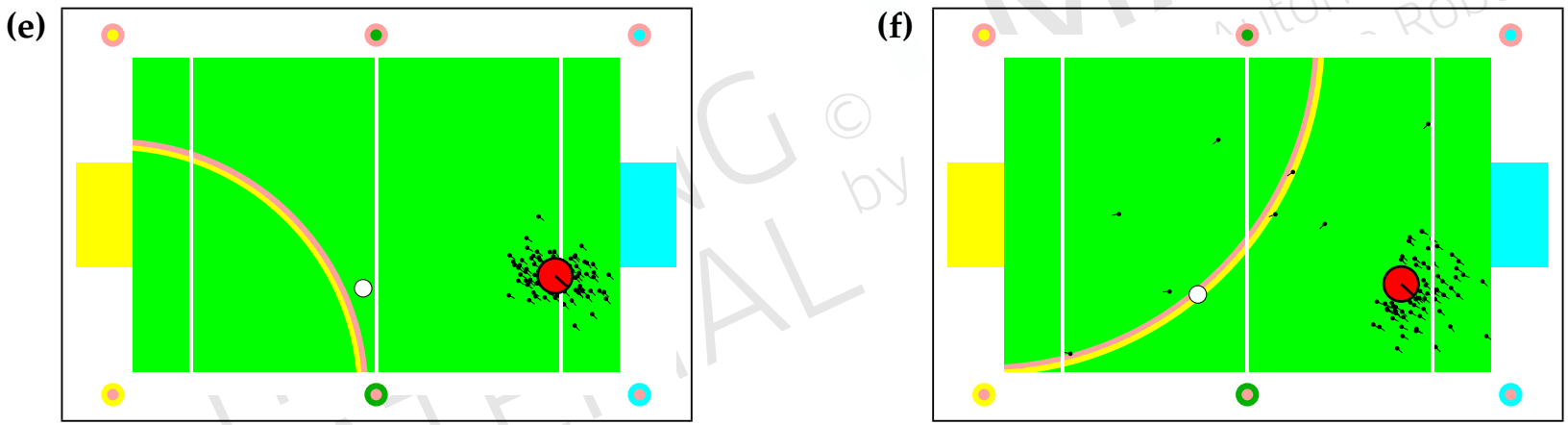
Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Monte-Carlo-Localisation, Example 2



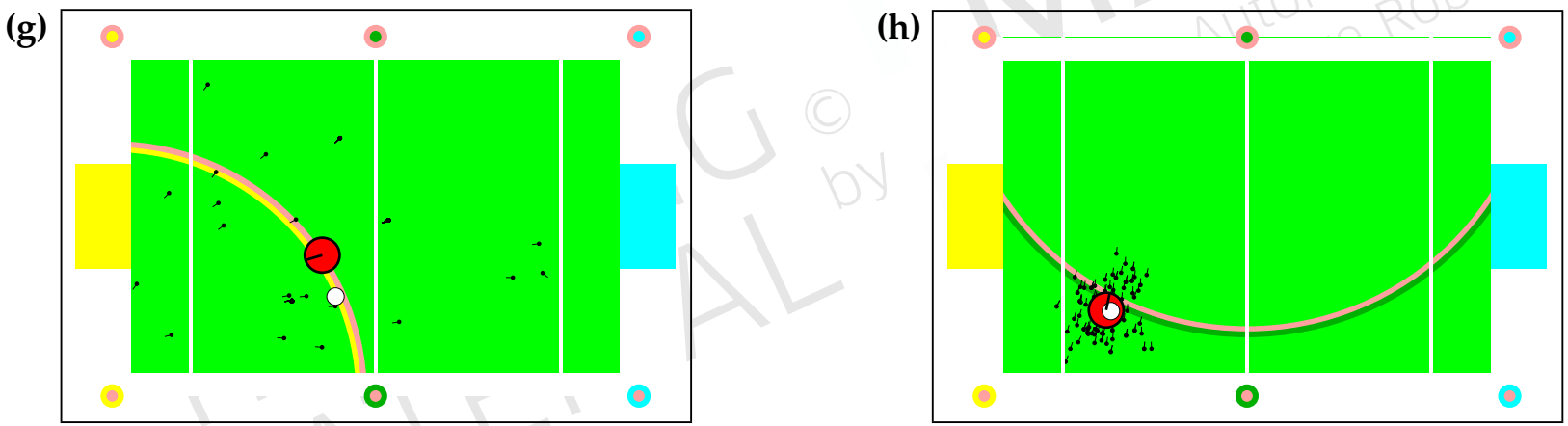
Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Monte-Carlo-Localisation, Example 2



Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Monte-Carlo-Localisation, Example 2



Source: (Thrun, Burgard, Fox, 2005) ©MIT Press

Summary

- Localisation methods presented today rely on the Bayes filter
- Differences in representing the environment and the way the probability distribution is given.
- Different approaches are differently well-suited for different localisation problems (Tracking, Global, Kidnapped-Robot)
- Depends on robot (accuracy of motion) and sensors used (accuracy of measurement)
- Field of active research; we only saw an excerpt

Learning Objectives

You will learn

- how to set up and use AMCL

Adaptive **M**onte **C**arlo **L**ocalization - I

- a probabilistic localization system for a robot moving in 2D
- implements the adaptive Monte Carlo localization approach
- uses a particle filter to track the pose of a robot against a known map

http://wiki.ros.org/amcl

Adaptive **M**onte **C**arlo **L**ocalization - II

subscribes to (default):

- Laser scans : /scan (sensor_msgs/LaserScan)
- Transforms : /tf (tf/tfMessage)
- initial pose : /initialpose (geometry_msgs/PoseWithCovarianceStamped)
- map : /map (nav_msgs/OccupancyGrid)

http://wiki.ros.org/amcl

Adaptive Monte Carlo Localization - III

publishes (default):

- robot's estimated pose in the map, with covariance: /amcl_pose (geometry_msgs/PoseWithCovarianceStamped)
- pose estimates being maintained by the filter: /particlecloud (geometry_msgs/PoseArray)
- the transform from odom : /tf (tf/tfMessage)

http://wiki.ros.org/amcl

Adaptive **M**onte **C**arlo **L**ocalization - IV

most important parameters (selection):

- ~odom_model_type (string, default: ”diff”)
- ~odom_frame_id (string, default: ”odom”)
- ~base_frame_id (string, default: ”base_link”)
- ~global_frame_id (string, default: ”map”)
- ~min_particles (int, default: 100)
- ~max_particles (int, default: 5000)

http://wiki.ros.org/amcl

Adaptive Monte Carlo Localization - IV

example launchfile:

```
<launch>
<node pkg="amcl" type="amcl" name="amcl" output="screen">
  <param name="odom_model_type" value="diff"/>
  <param name="odom_frame_id" value="odom"/>
  <param name="base_frame_id" value="base_link"/>
  <param name="global_frame_id" value="map"/>
  <remap from="scan" to="laserscan"/>
</node>
</launch>
```

http://wiki.ros.org/amcl

Adaptive Monte Carlo Localization - V

example for high uncertaincy:

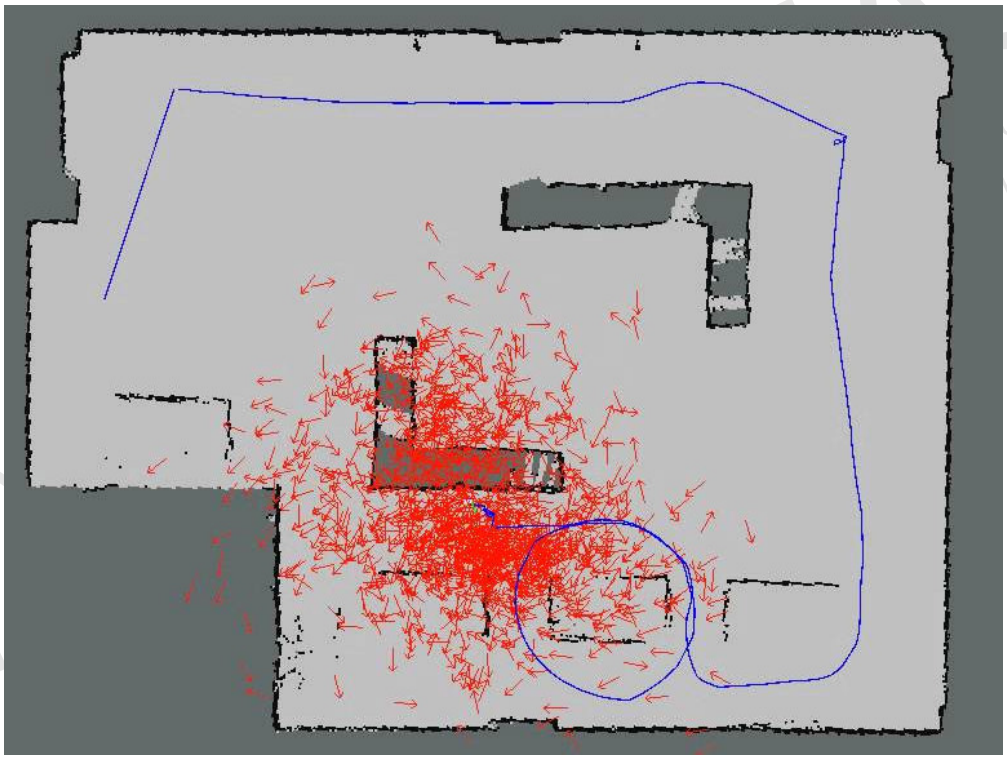


Figure: PR2 Transforms

Adaptive Monte Carlo Localization - VI

example for low uncertaincy:

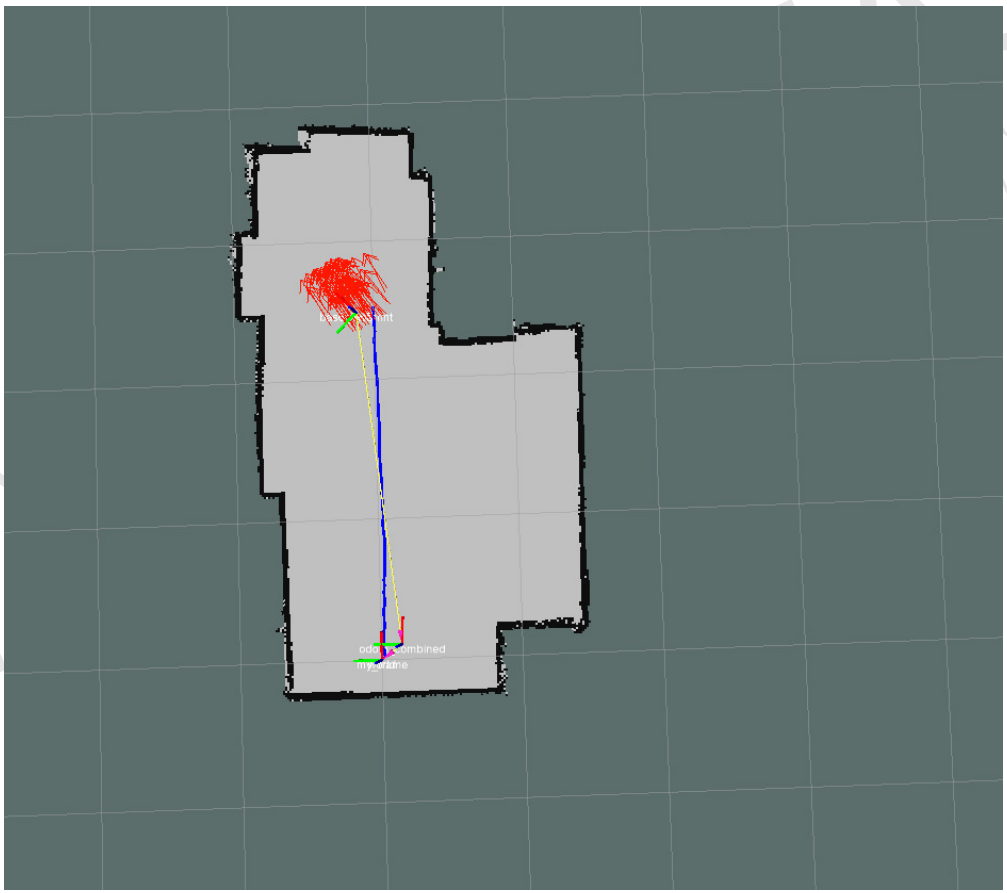


Figure: PR2 Transforms