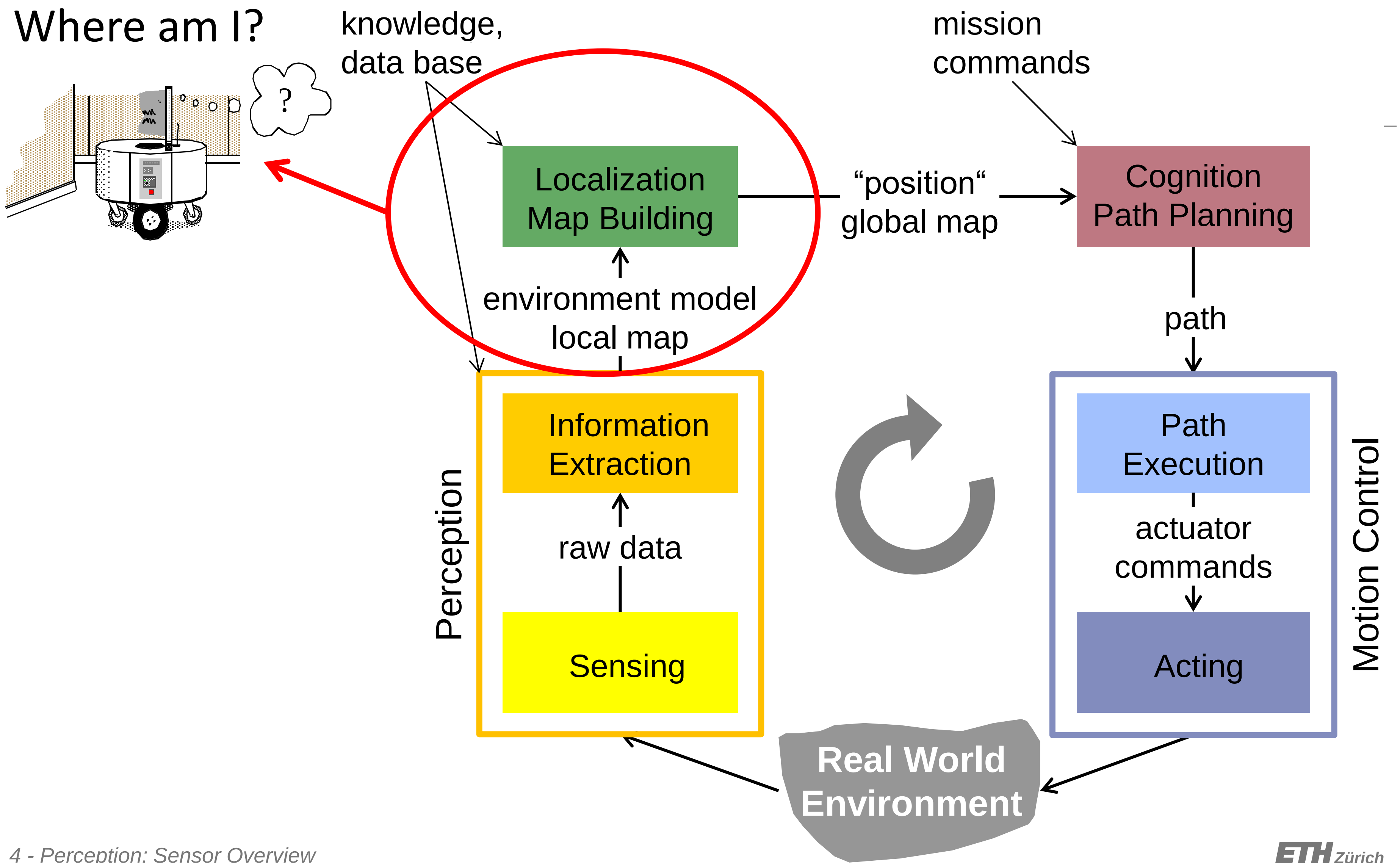
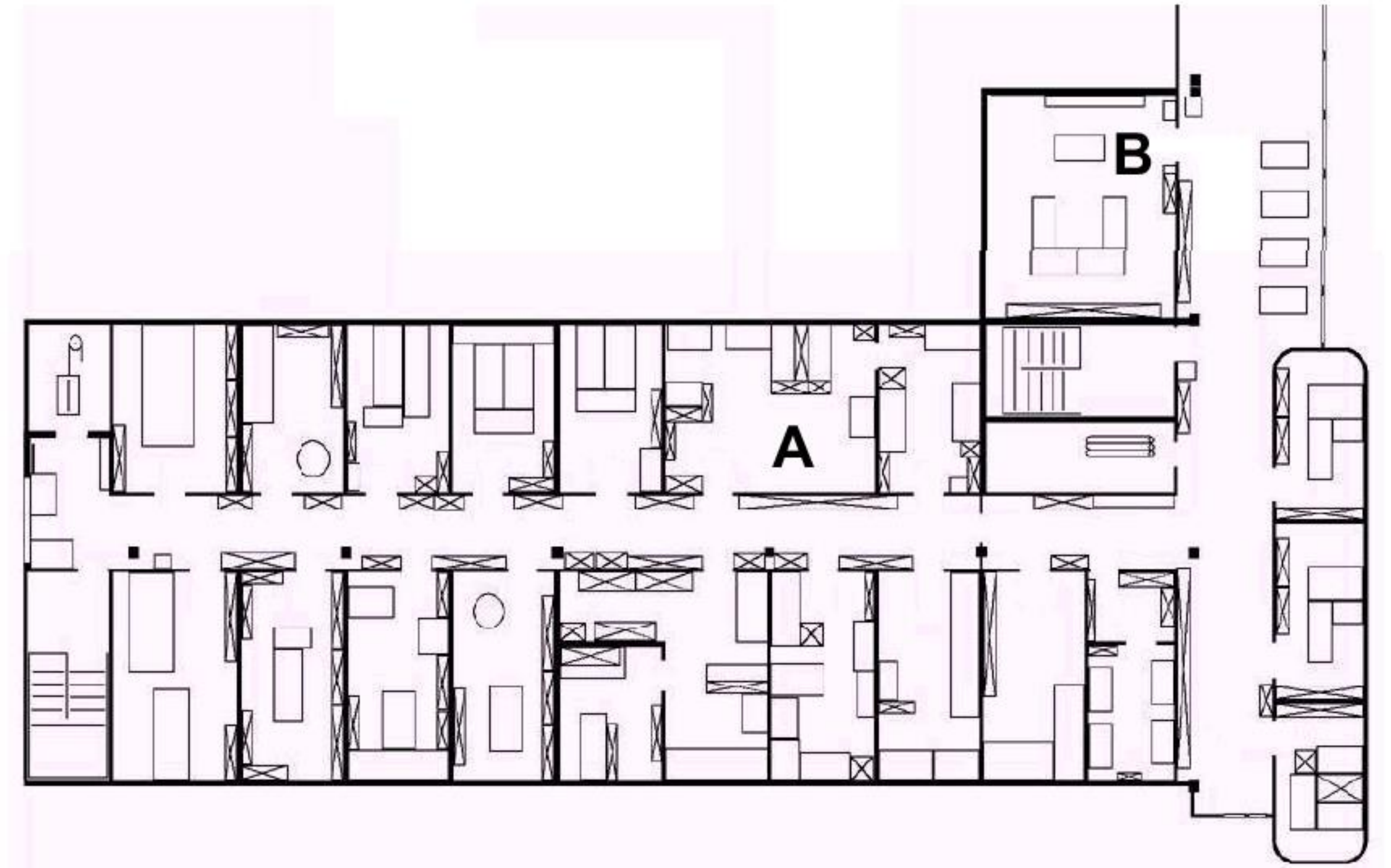
Introduction
Where are we?



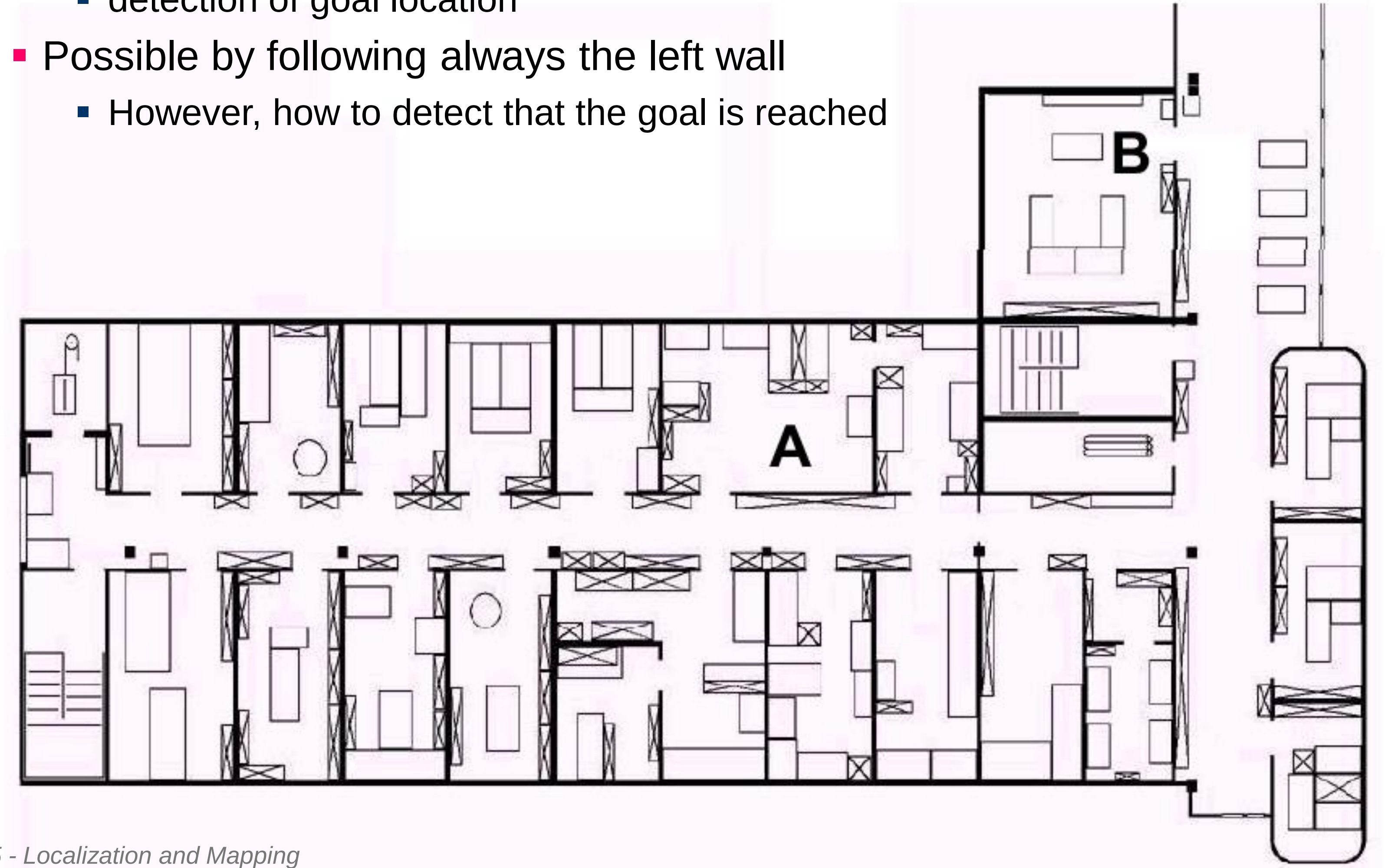
Introduction
Do we need to localize or not?

- To go from A to B, does the robot need to know where it is?



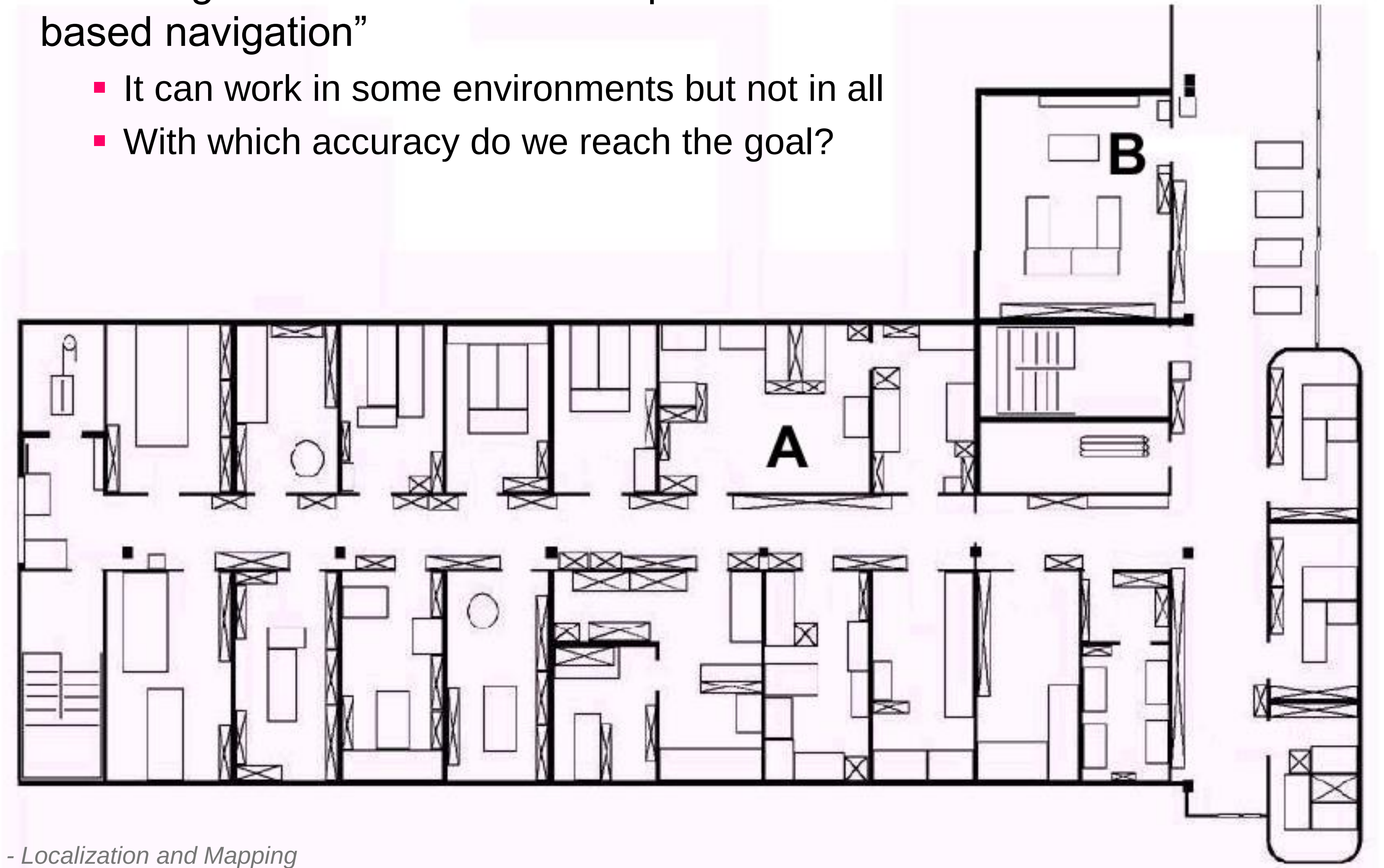
Introduction
Do we need to localize or not?

- How to navigate between A and B
  - navigation without hitting obstacles
  - detection of goal location
- Possible by following always the left wall
  - However, how to detect that the goal is reached



Introduction
Do we need to localize or not?

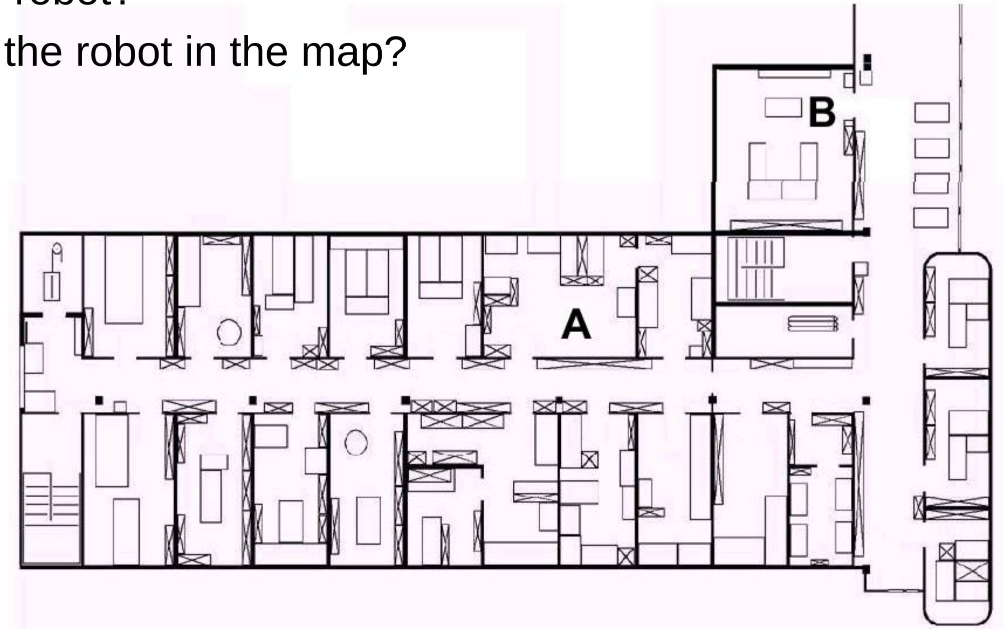
- Following the left wall is an example of “behavior based navigation”
  - It can work in some environments but not in all
  - With which accuracy do we reach the goal?



ETH Zürich

Introduction
Do we need to localize or not?

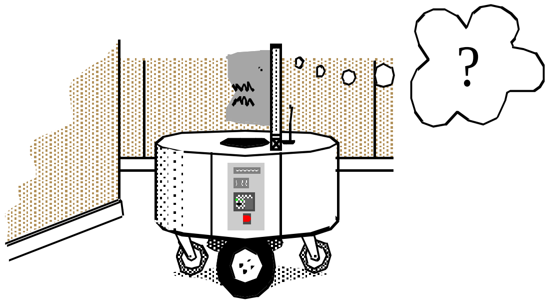
- As opposed to behavior based navigation is “map based navigation”
  - Assuming that the map is known, at every time step the robot has to know where it is. How?
    - If we know the start position, we can use wheel odometry or dead reckoning. Is this enough? What else can we use?
- But how do we represent the map for the robot?
- And how do we represent the position of the robot in the map?



Introduction
Definitions

- Global localization
  - The robot is not told its initial position
  - Its position must be estimated from scratch

- Position Tracking
  - A robot knows its initial position and “only” has to accommodate small errors in its odometry as it moves



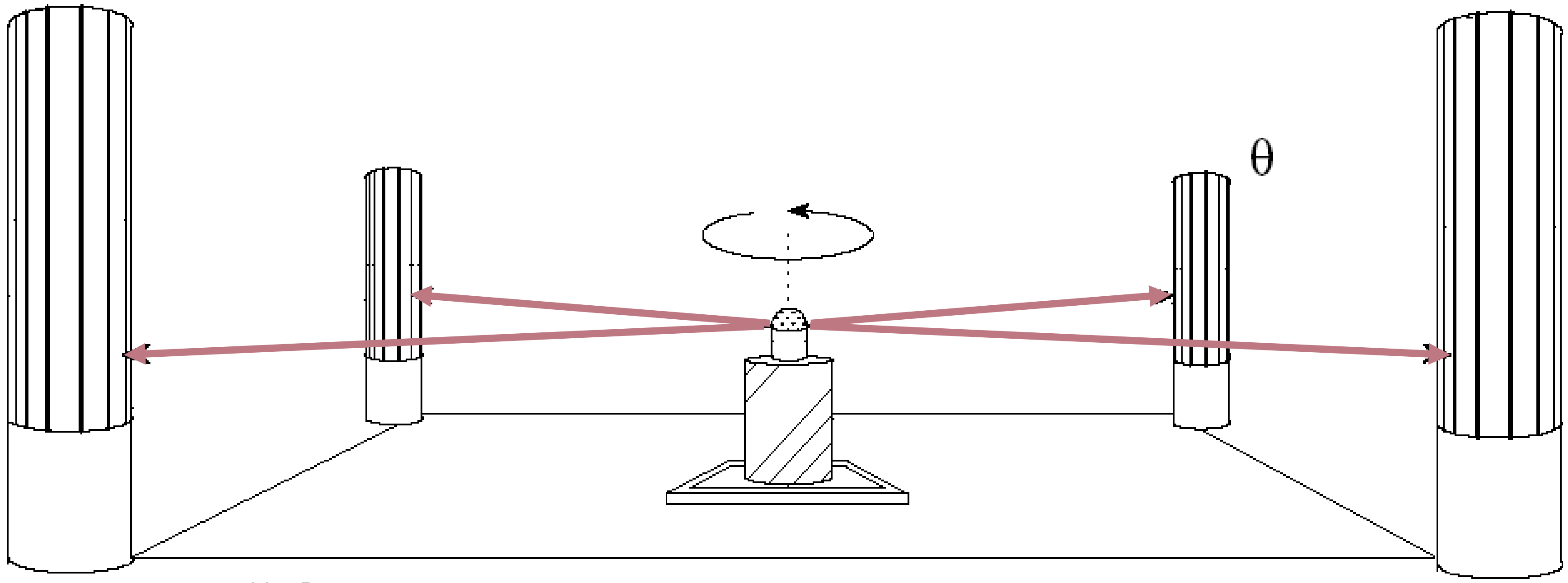
Introduction
How to localize?

- Localization based on external sensors, beacons or landmarks
- Odometry
- Map Based Localization (without external sensors or artificial landmarks. Just use robot onboard sensors)
  - Example: Probabilistic Map Based Localization

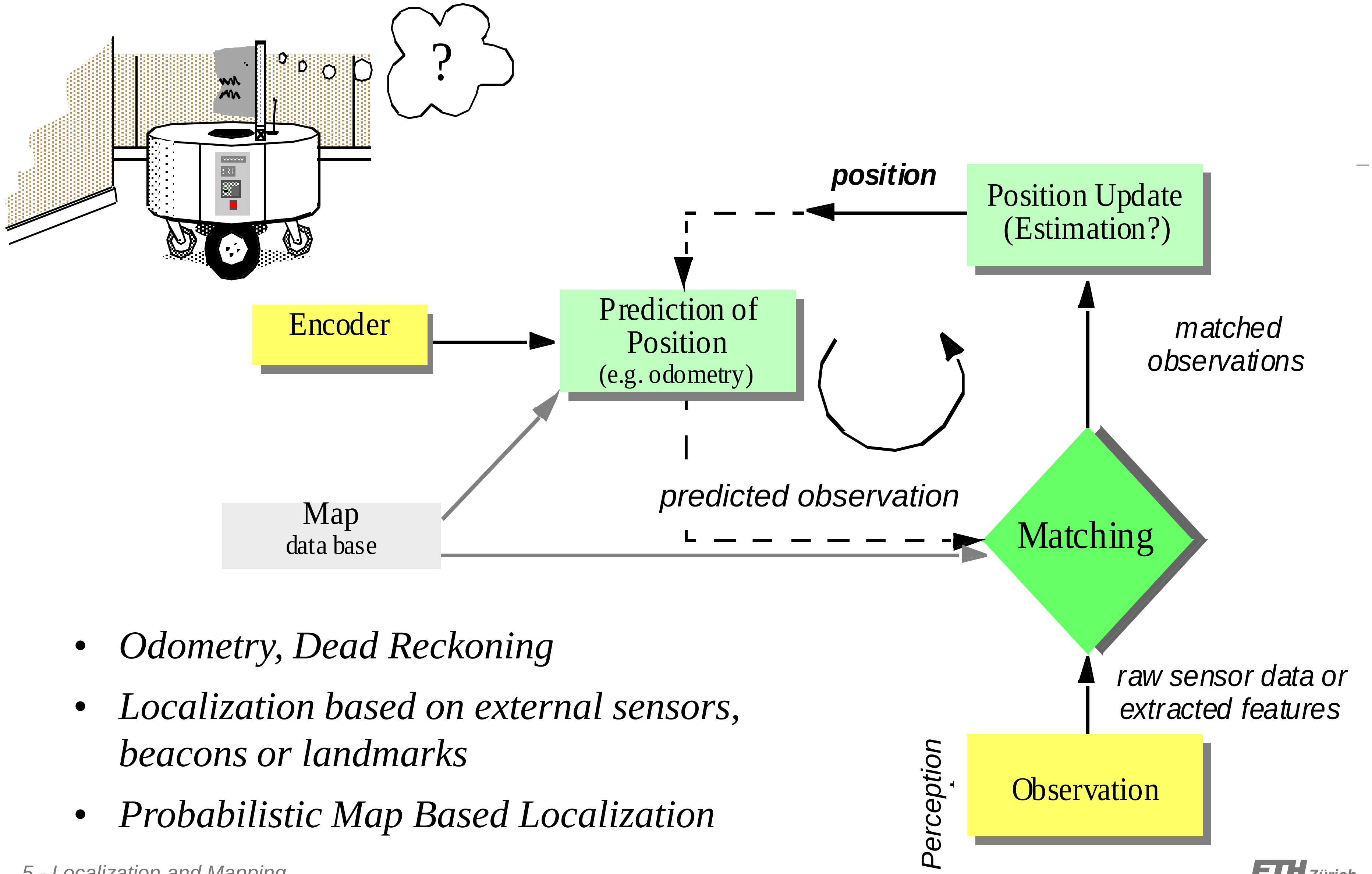
Introduction
Beacon Based Localization

Triangulation

- Ex 1: Poles with highly reflective surface and a laser for detecting them
- Ex 2: Coloured beacons and an omnidirectional camera for detecting them (example: RoboCup or autonomous robots in tennis fields)



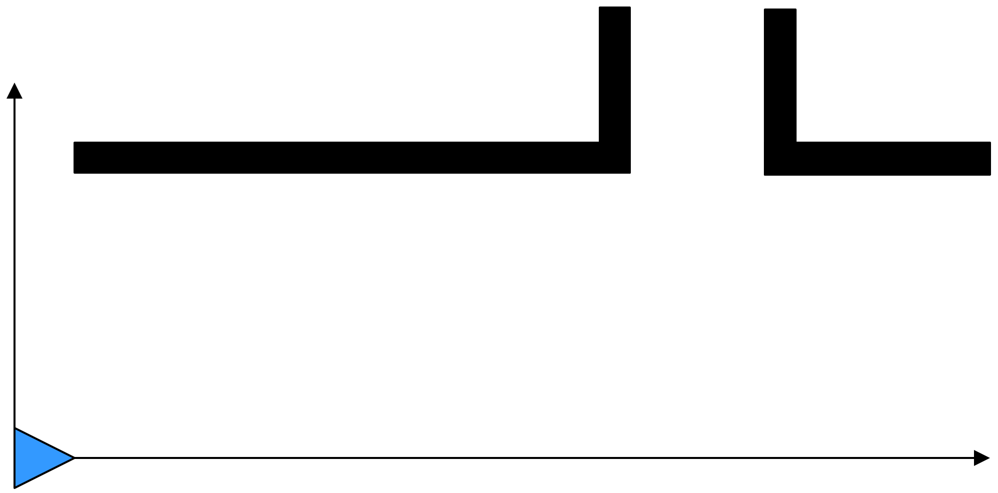
Introduction
Map-based localization


?
position
Position Update (Estimation?)
Encoder
Prediction of Position (e.g. odometry)
matched observations
predicted observation
Map data base
Matching
raw sensor data or extracted features
Perception
Observation

- *Odometry, Dead Reckoning*
- *Localization based on external sensors, beacons or landmarks*
- *Probabilistic Map Based Localization*

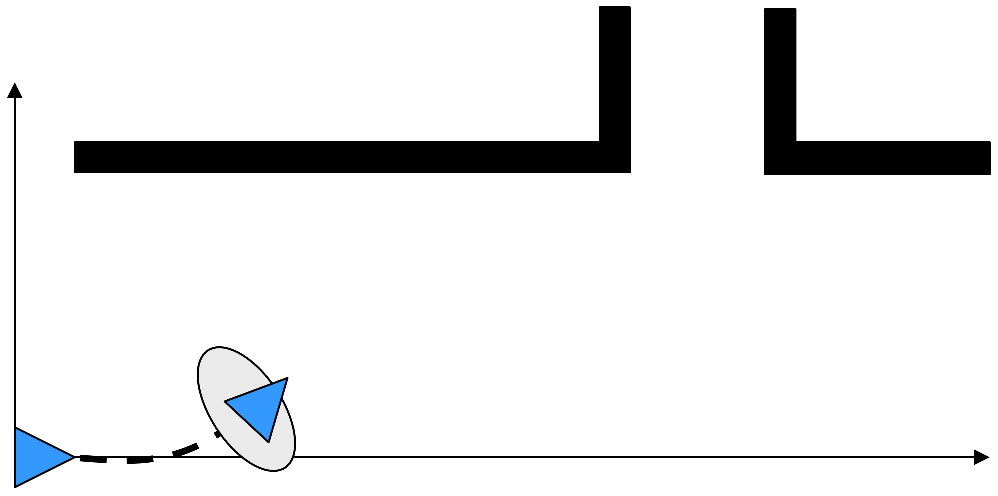
Introduction
Map-based localization

- Consider a mobile robot moving in a known environment.



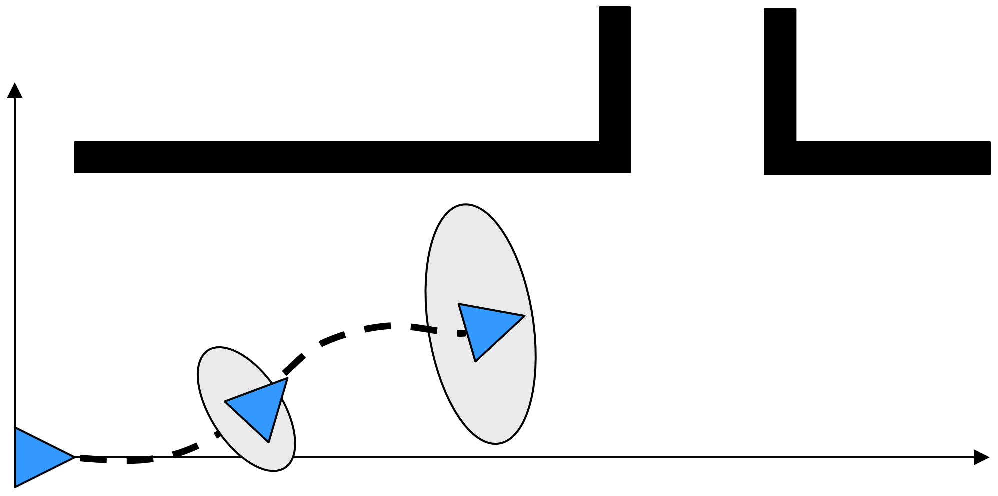
Introduction
Map-based localization

- Consider a mobile robot moving in a known environment.
- As it starts to move, say from a precisely known location, it can keep track of its motion using odometry.



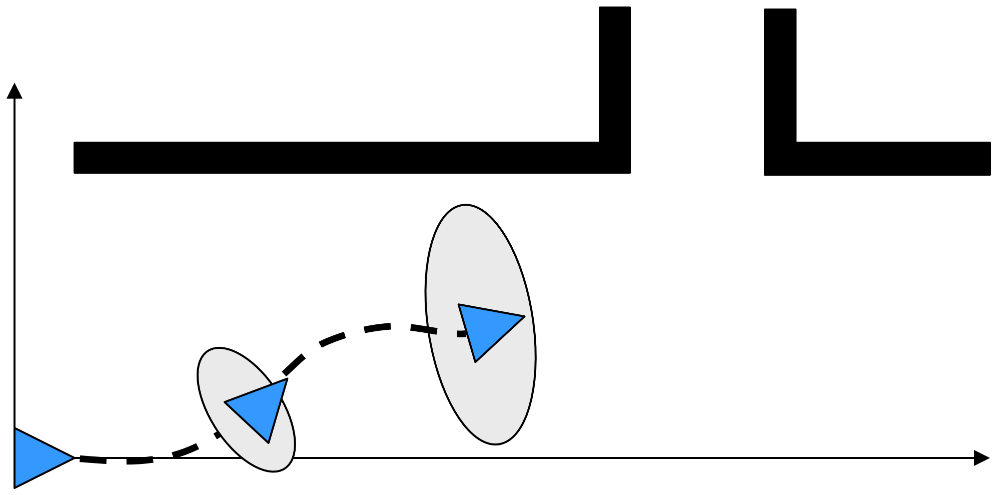
Introduction
Map-based localization

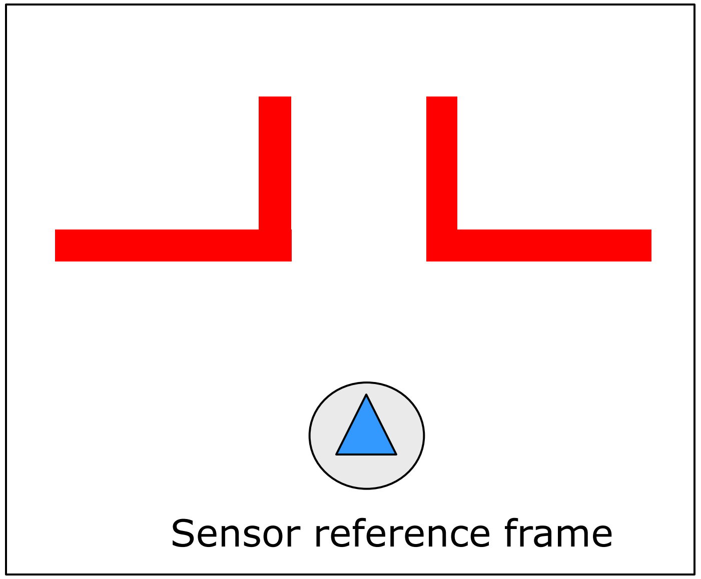
- Consider a mobile robot moving in a known environment.
- As it starts to move, say from a precisely known location, it can keep track of its motion using odometry.



Introduction
Map-based localization

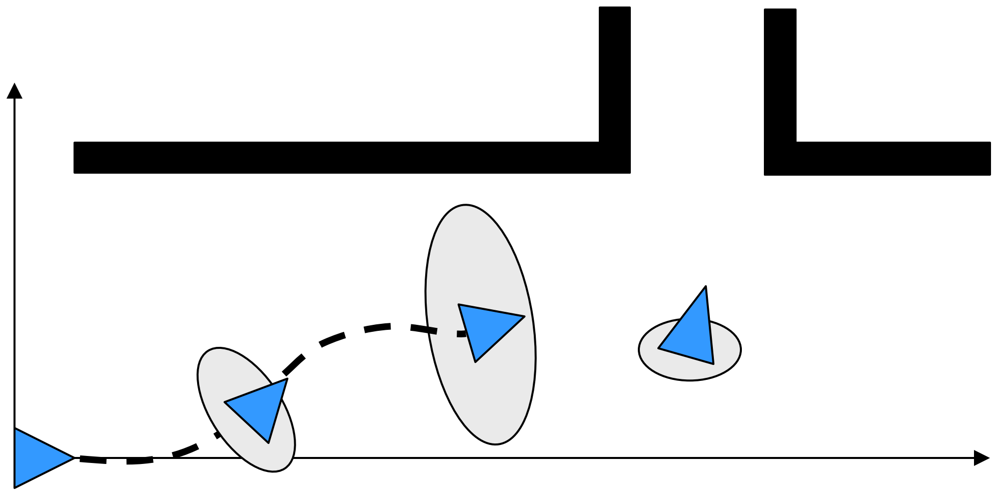
- Consider a mobile robot moving in a known environment.
- As it starts to move, say from a precisely known location, it can keep track of its motion using odometry.

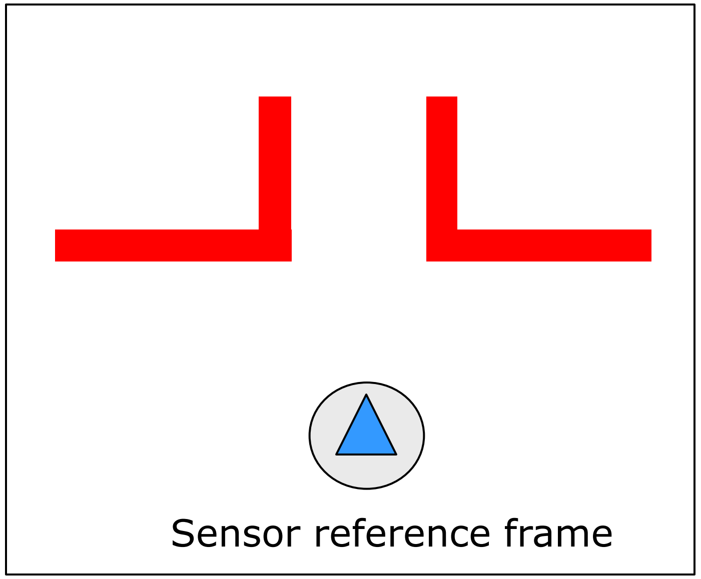




Introduction
Map-based localization

- Consider a mobile robot moving in a known environment.
- As it starts to move, say from a precisely known location, it can keep track of its motion using odometry.
- The robot makes an observation and updates its position and uncertainty





Introduction
Map-based localization

Ingredients

- Position estimation (odometry)
- Error (uncertainty) propagation
- Position representation
- Map representation

Odometry

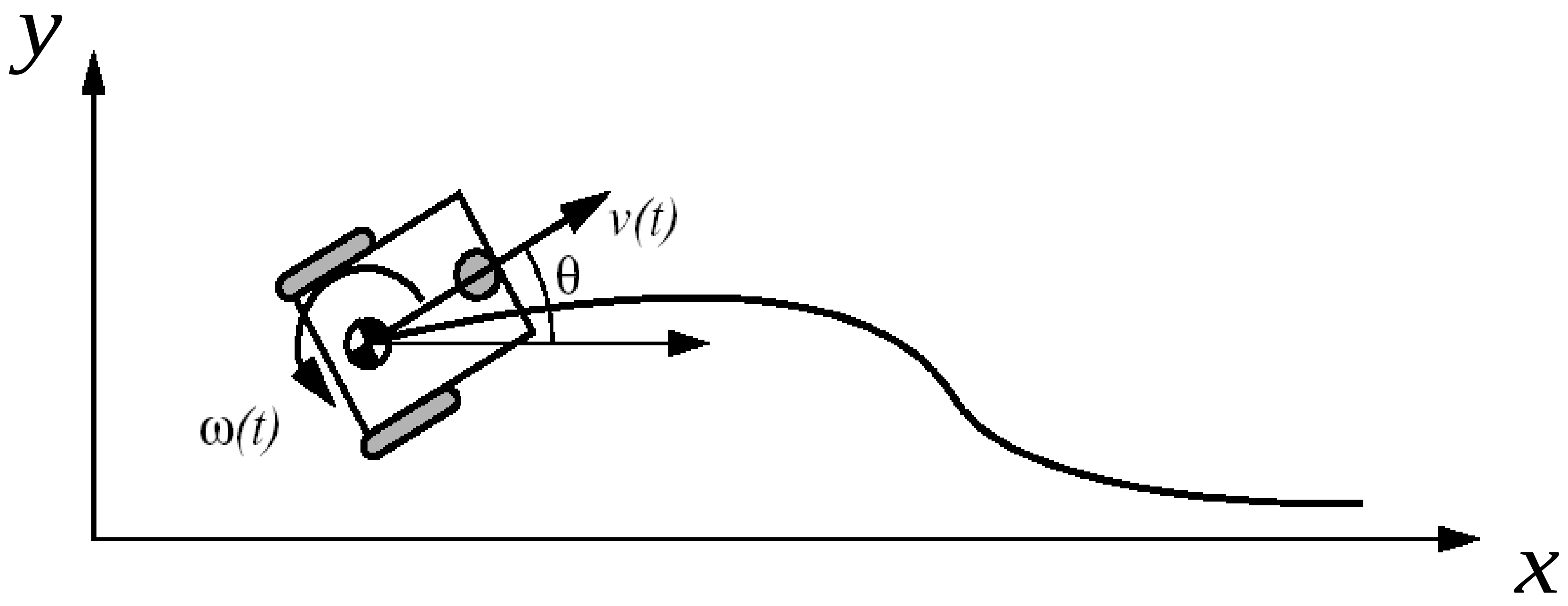
- Definition
  - **Dead reckoning** (also **deduced reckoning** or **odometry**) is the process of calculating vehicle's current position by using a previously determined position and estimated speeds over the elapsed time

- Robot motion is recovered by integrating proprioceptive sensor velocities readings
  - Pros: Straightforward
  - Cons: Errors are integrated -> unbound

- Heading sensors (e.g., gyroscope) help to reduce the accumulated errors but drift remains

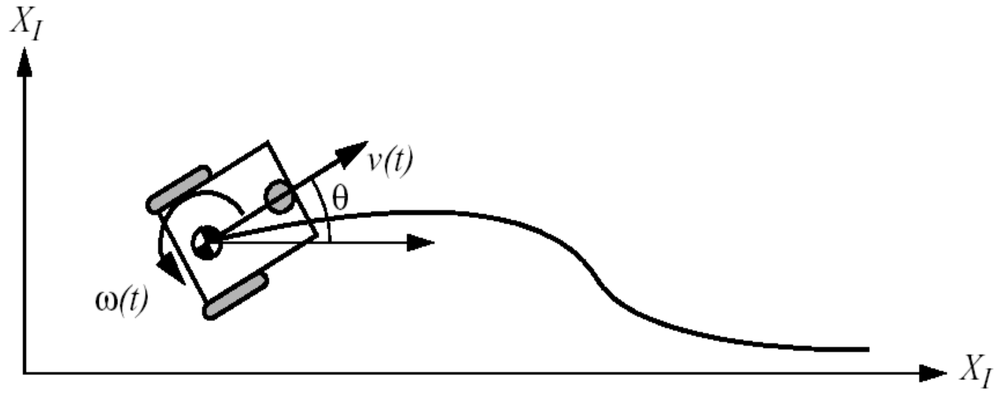
Odometry
The Differential Drive Robot (1)

$$x = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \qquad \hat{x}_t = x_{t-1} + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix} = f(x_{t-1}, u_t)$$



Odometry
Wheel Odometry

- Kinematics



$$
\begin{cases}
\hat{x}_t = f(x_{t-1}, u_t) = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} \Delta s \cos(\theta + \frac{\Delta\theta}{2}) \\ \Delta s \sin(\theta + \frac{\Delta\theta}{2}) \\ \Delta\theta \end{bmatrix} \\
\Delta s = \frac{\Delta s_r + \Delta s_l}{2} \\
\Delta\theta = \frac{\Delta s_r - \Delta s_l}{b}
\end{cases}
$$

$\rightarrow$ This term comes from the application of the Instantaneous Center of Rotation

Can you demonstrate these equations?

Odometry
Odometric Error Propagation

- Error model

$$P_t = F_{x_{t-1}} \cdot \Sigma_{x_{t-1}} \cdot F_{x_{t-1}}{}^T + F_{\Delta S} \cdot \Sigma_{\Delta S} \cdot F_{\Delta S}{}^T$$

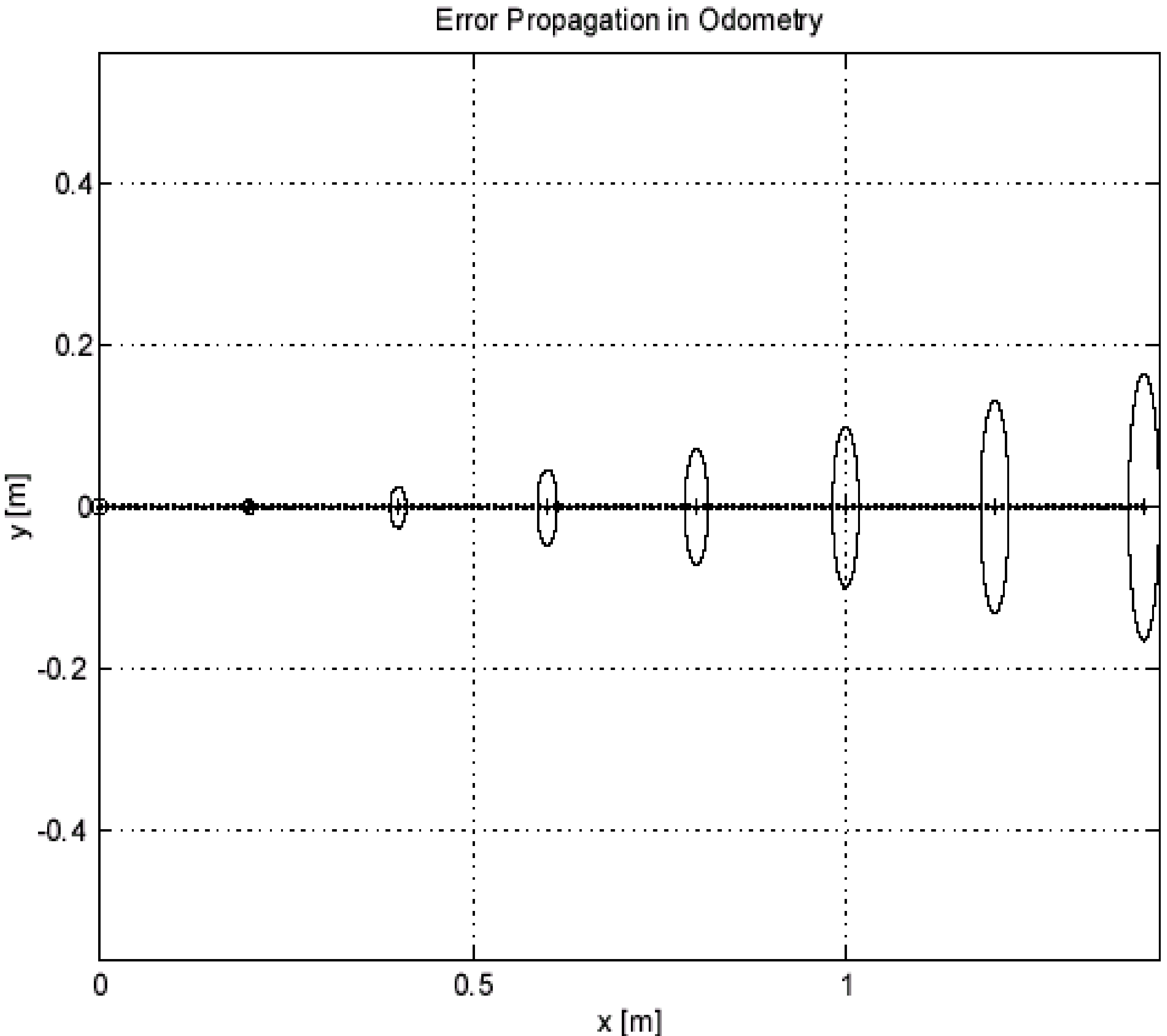
$$\Sigma_{\Delta S} = \begin{bmatrix} k_r |\Delta s_r| & 0 \\ 0 & k_l |\Delta s_l| \end{bmatrix}$$

$$F_{x_{t-1}} = \nabla f_{x_{t-1}} = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial \theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\Delta s \sin(\theta + \Delta\theta/2) \\ 0 & 1 & \Delta s \cos(\theta + \Delta\theta/2) \\ 0 & 0 & 1 \end{bmatrix}$$

$$F_{\Delta S} = \begin{bmatrix} \frac{1}{2}\cos\left(\theta + \frac{\Delta\theta}{2}\right) - \frac{\Delta s}{2b}\sin\left(\theta + \frac{\Delta\theta}{2}\right) & \frac{1}{2}\cos\left(\theta + \frac{\Delta\theta}{2}\right) + \frac{\Delta s}{2b}\sin\left(\theta + \frac{\Delta\theta}{2}\right) \\ \frac{1}{2}\sin\left(\theta + \frac{\Delta\theta}{2}\right) + \frac{\Delta s}{2b}\cos\left(\theta + \frac{\Delta\theta}{2}\right) & \frac{1}{2}\sin\left(\theta + \frac{\Delta\theta}{2}\right) - \frac{\Delta s}{2b}\cos\left(\theta + \frac{\Delta\theta}{2}\right) \\ \frac{1}{b} & -\frac{1}{b} \end{bmatrix}$$

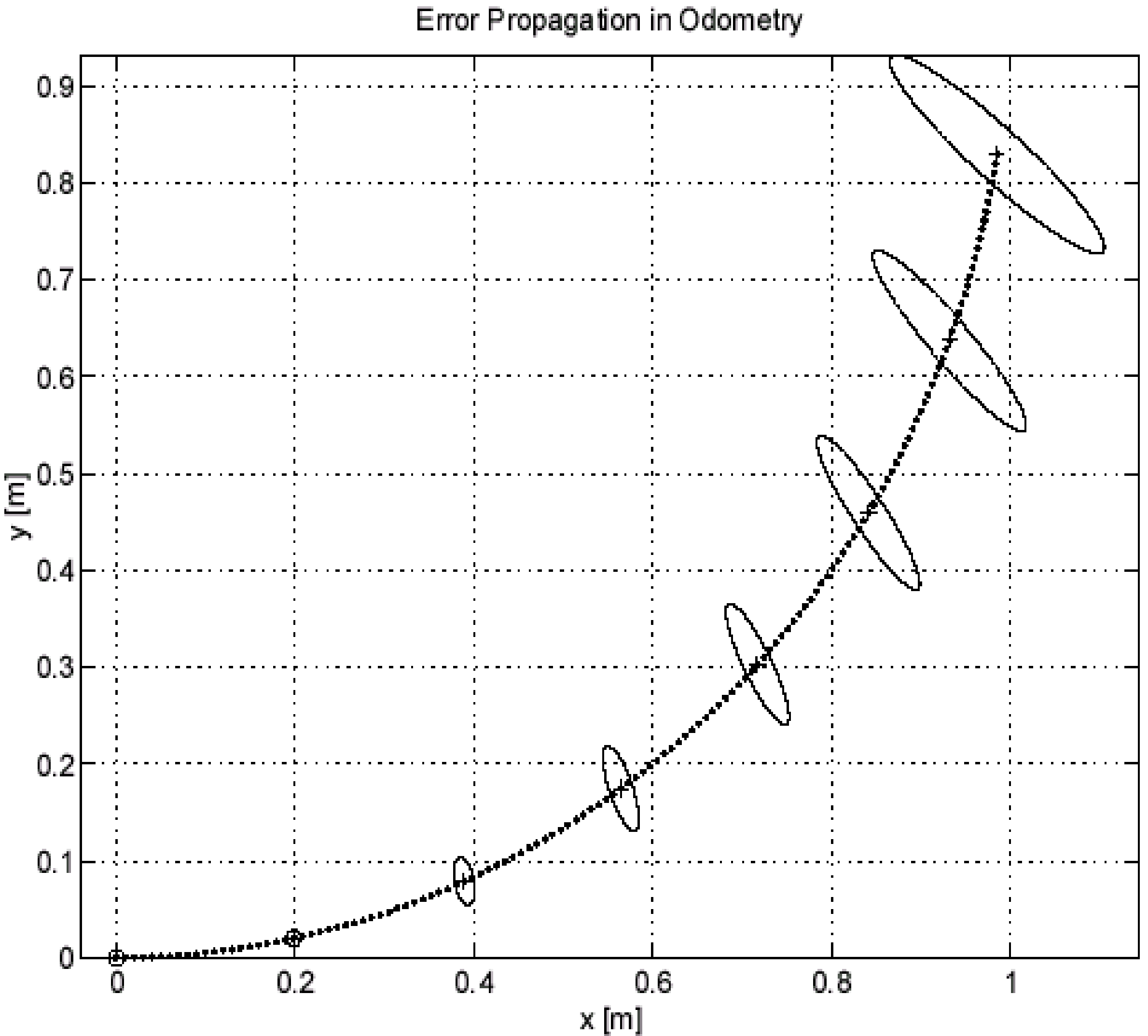
Odometry
Growth of Pose uncertainty for Straight Line Movement

- Note: Errors perpendicular to the direction of movement are growing much faster!



Odometry
Growth of Pose uncertainty for Movement on a Circle

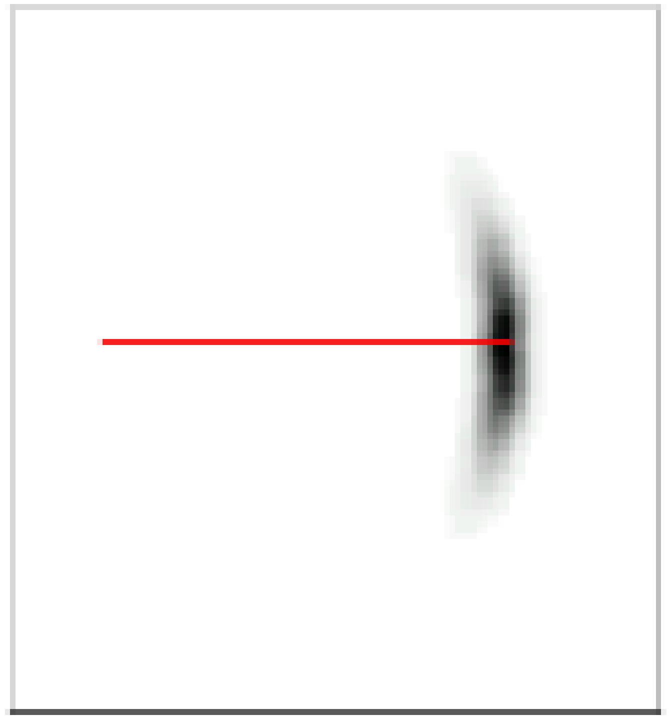
- Note: Errors ellipse does not remain perpendicular to the direction of movement!

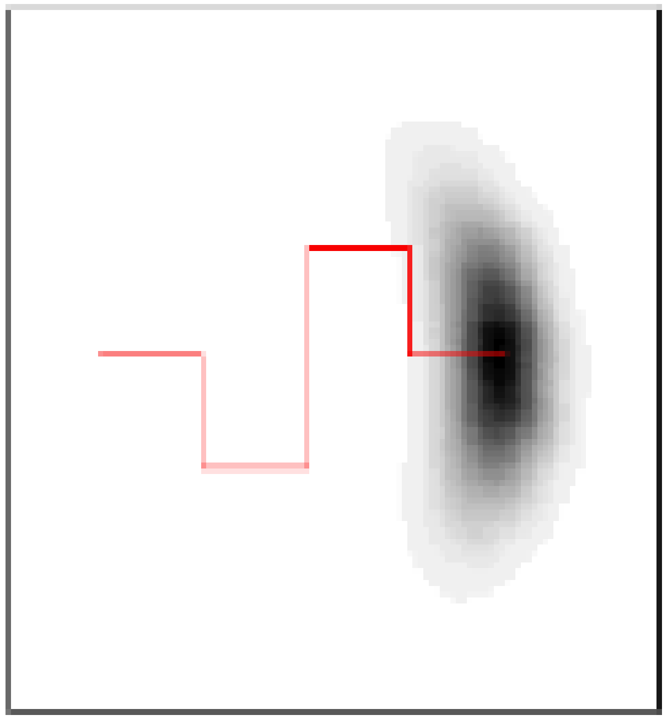


Odometry
Example of non-Gaussian error model

- Note: Errors are not shaped like ellipses!

Courtesy AI Lab, Stanford





[Fox, Thrun, Burgard, Dellaert, 2000]

Odometry
Error sources

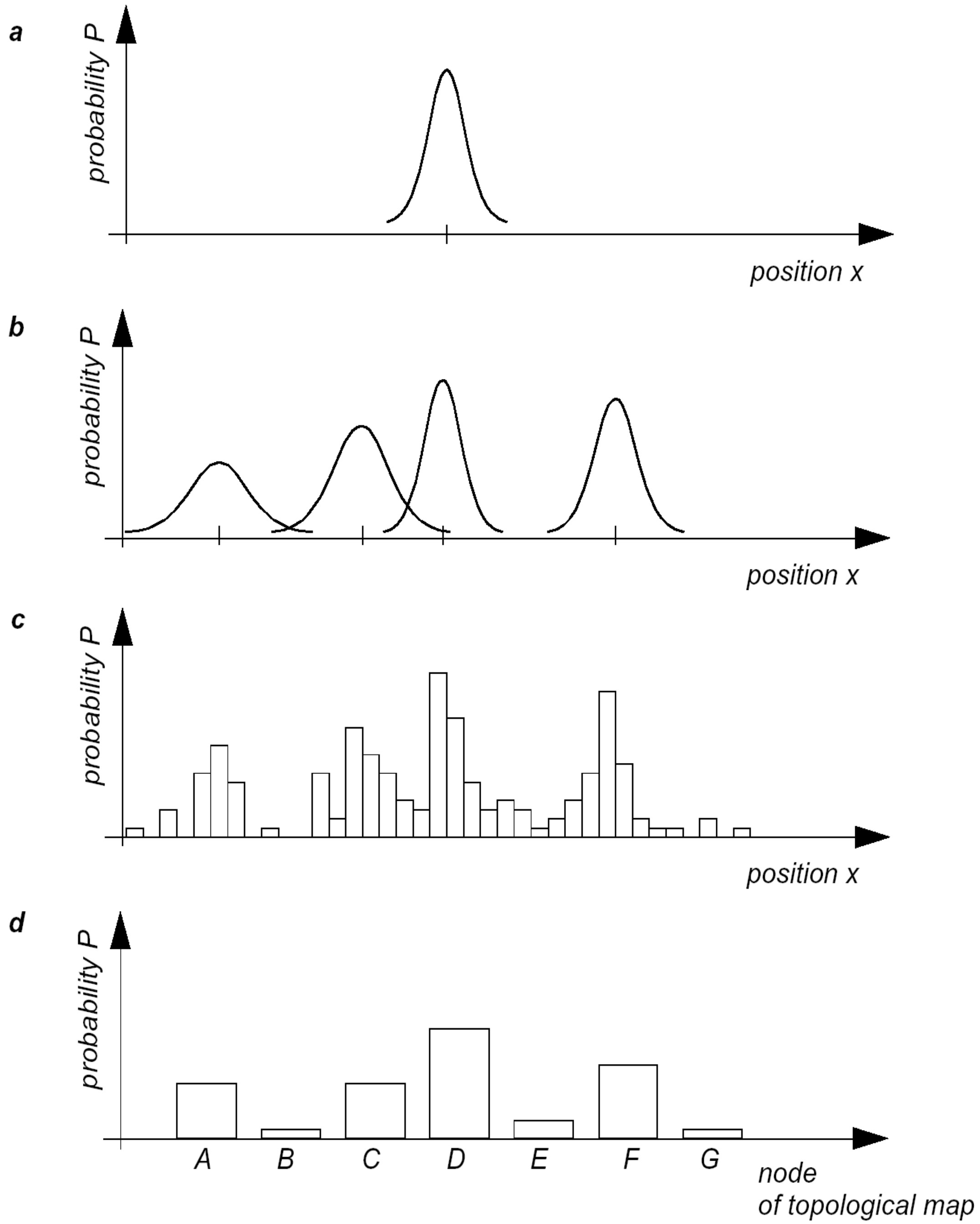
Deterministic (Systematic) ⟷ Non-Deterministic (Non-Systematic)

- Deterministic errors can be eliminated by proper calibration of the system.
- Non-Deterministic errors are random errors. They have to be described by error models and will always lead to uncertain position estimate.

- Major Error Sources in Odometry:
  - Limited resolution during integration (time increments, measurement resolution)
  - Misalignment of the wheels (deterministic)
  - Unequal wheel diameter (deterministic)
  - Variation in the contact point of the wheel (non deterministic)
  - Unequal floor contact (slippage, non planar …) (non deterministic)

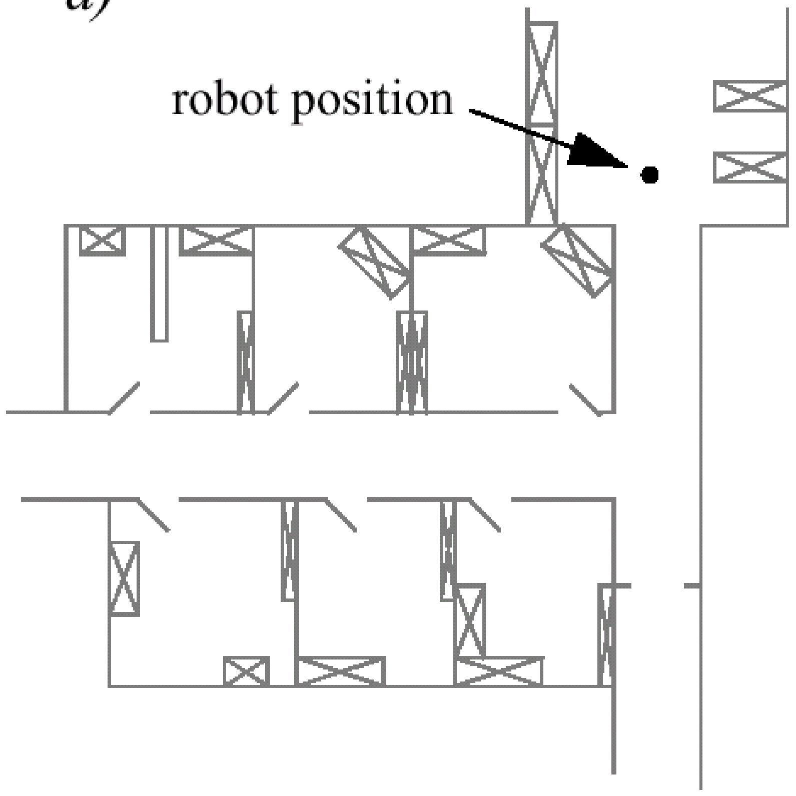
Belief Representation

- a) Continuous map
  with single hypothesis
  probability distribution
- b) Continuous map
  with multiple hypotheses
  probability distribution
- c) Discretized map
  with probability distribution
- d) Discretized topological
  map with probability
  distribution



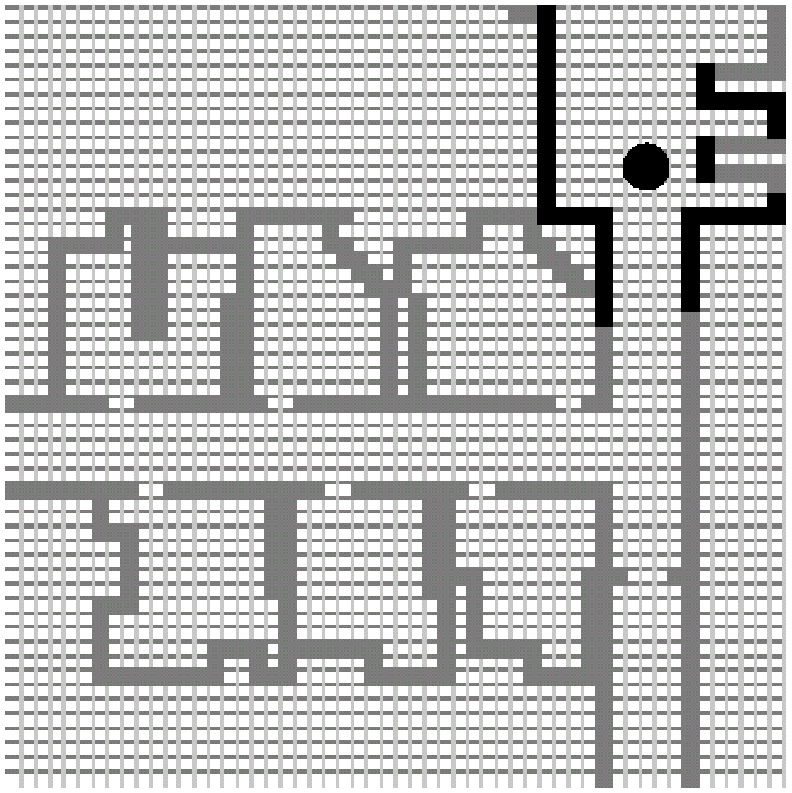
Belief Representation
Single-hypothesis Belief – Continuous Line-Map

a)

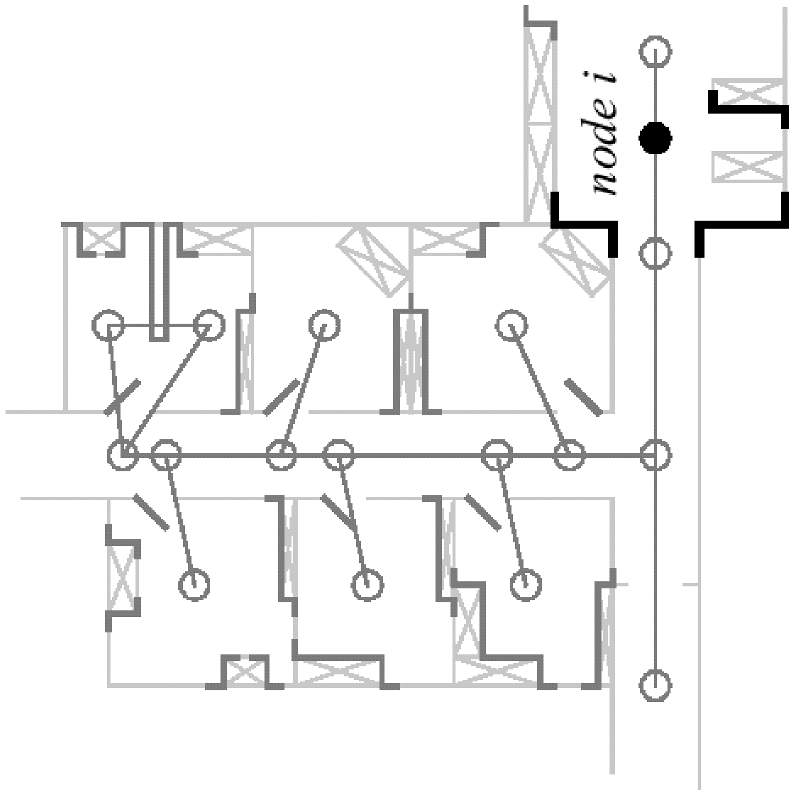


Belief Representation
Single-hypothesis Belief – Grid and Topological Map

c)



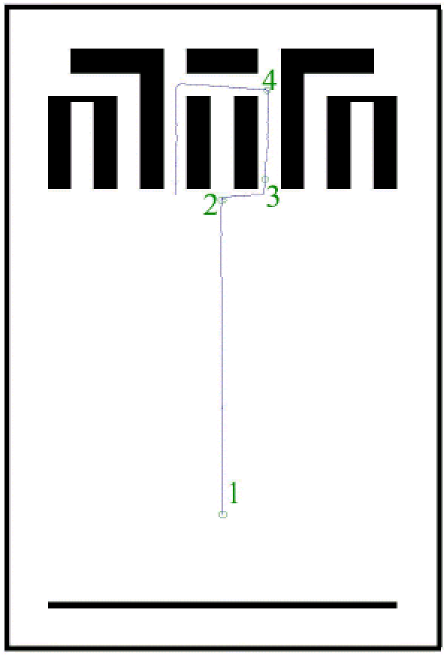
d)



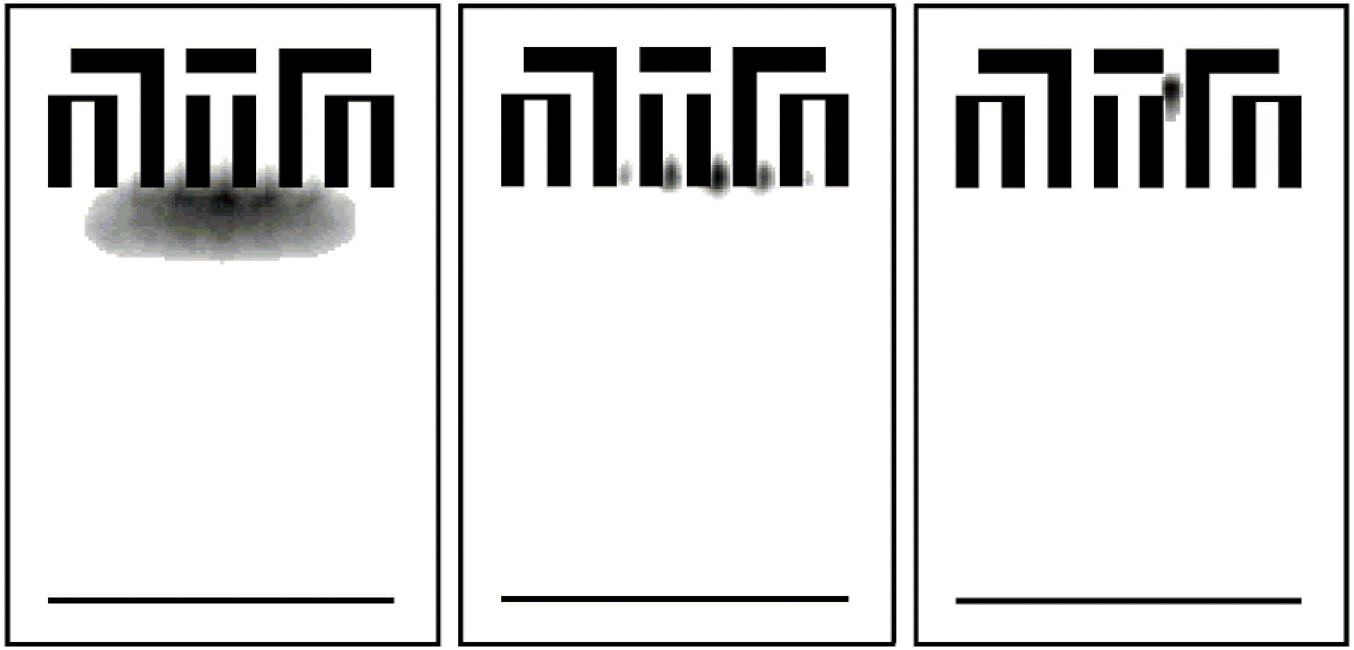
Belief Representation
Grid-based Representation - Multi Hypothesis

- Grid size around 20 cm^2.

Courtesy of W. Burgard



Path of the robot

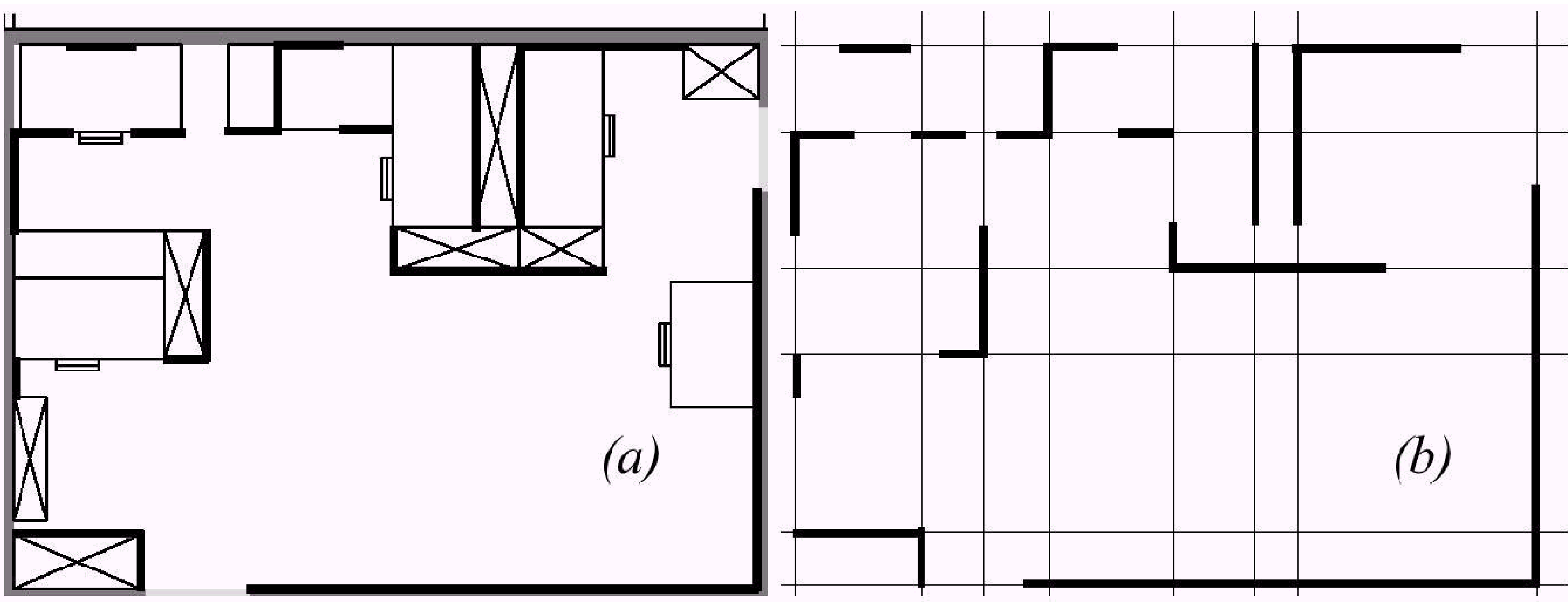


Belief states at positions 2, 3 and 4

Map Representation
Continuous Line-Based

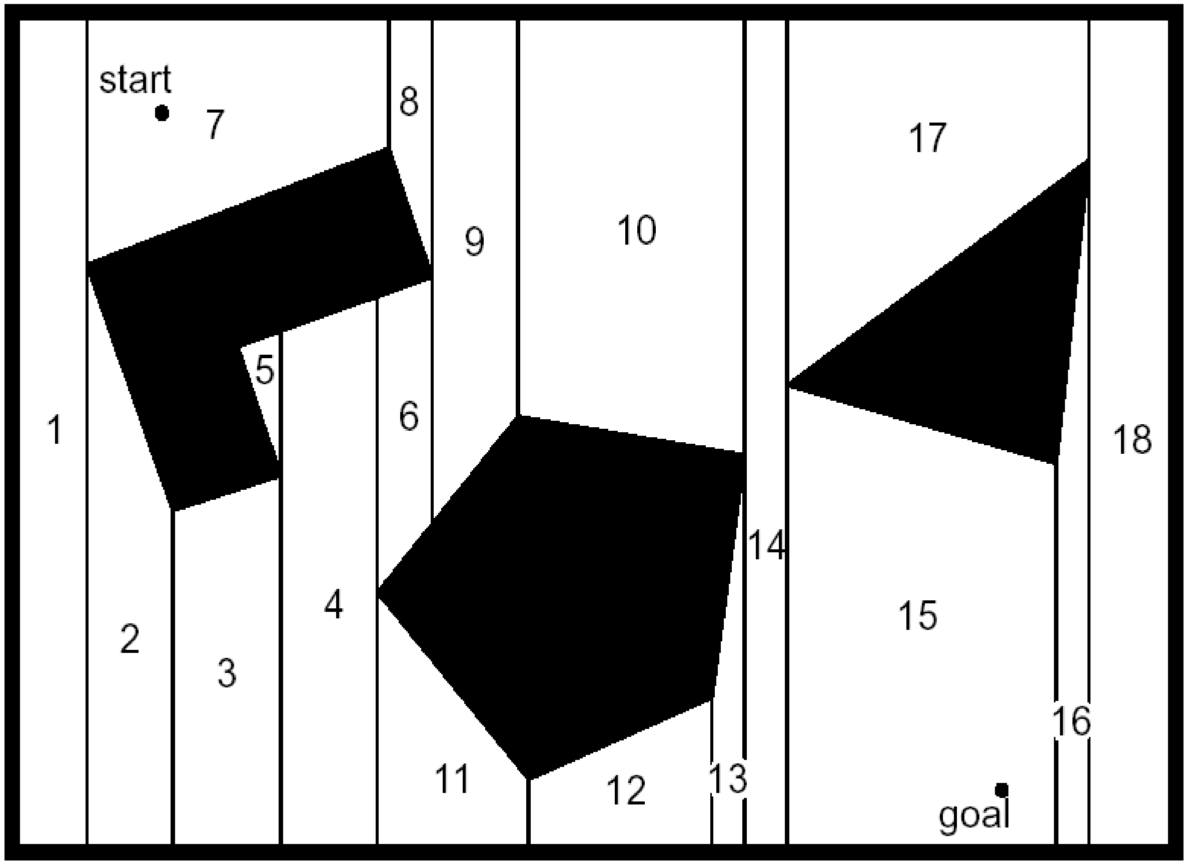
a) Architecture map

b) Representation with set of finite or infinite lines



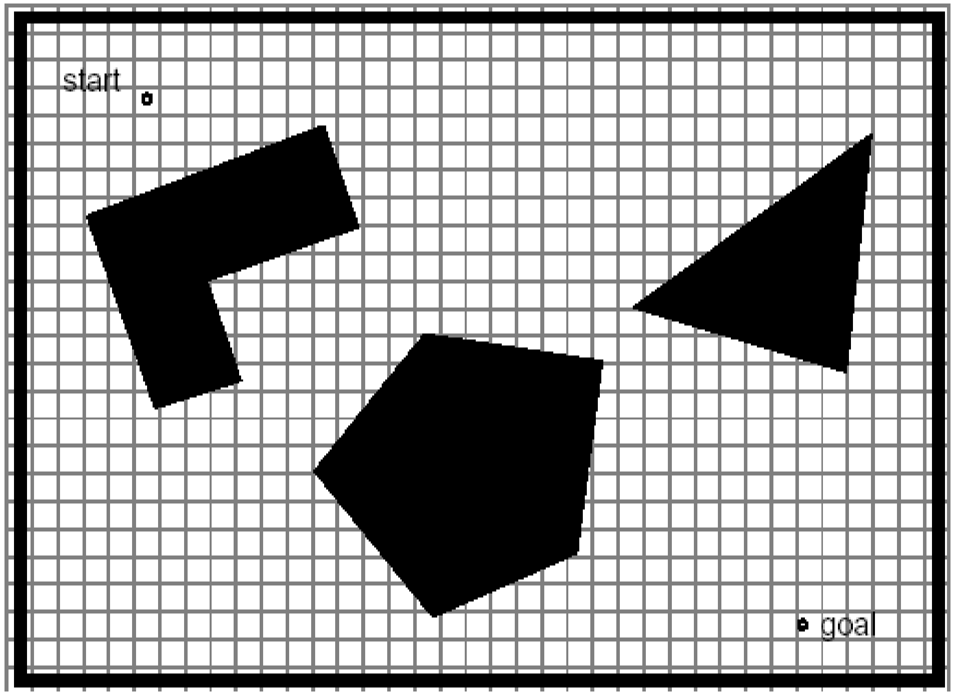
Map Representation
Exact cell decomposition

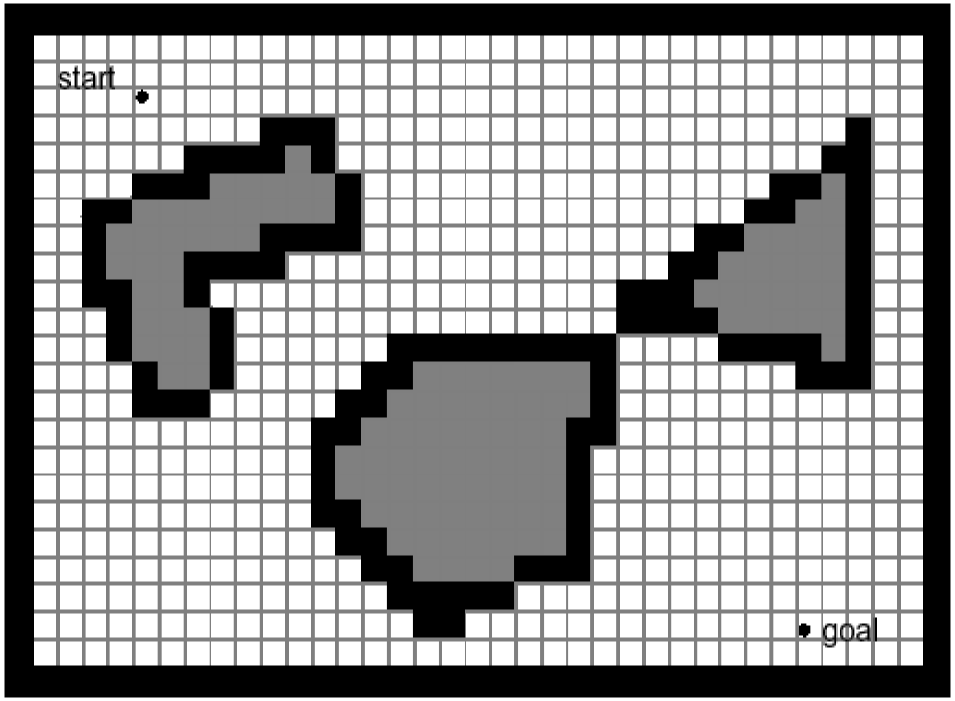
- Exact cell decomposition - Polygons



Map Representation
Approximate cell decomposition

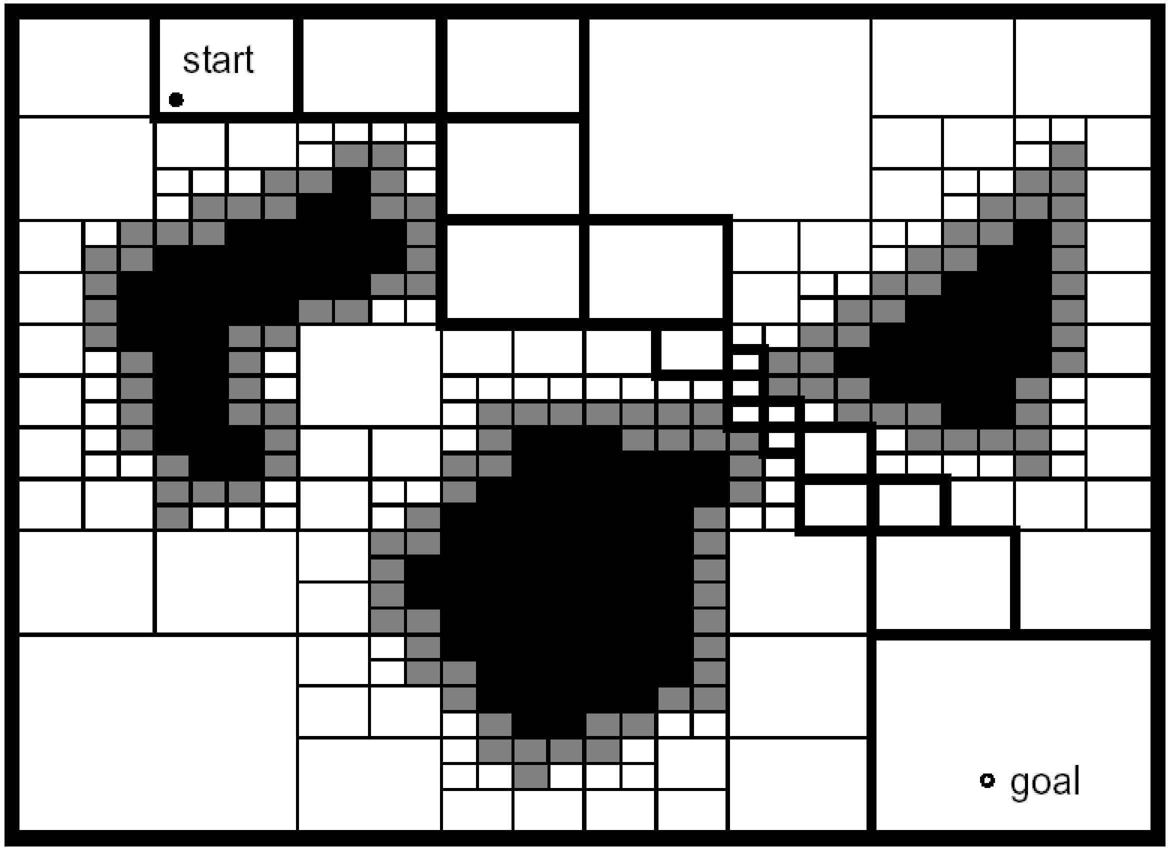
- Fixed cell decomposition
  - Narrow passages disappear





Map Representation
Adaptive cell decomposition

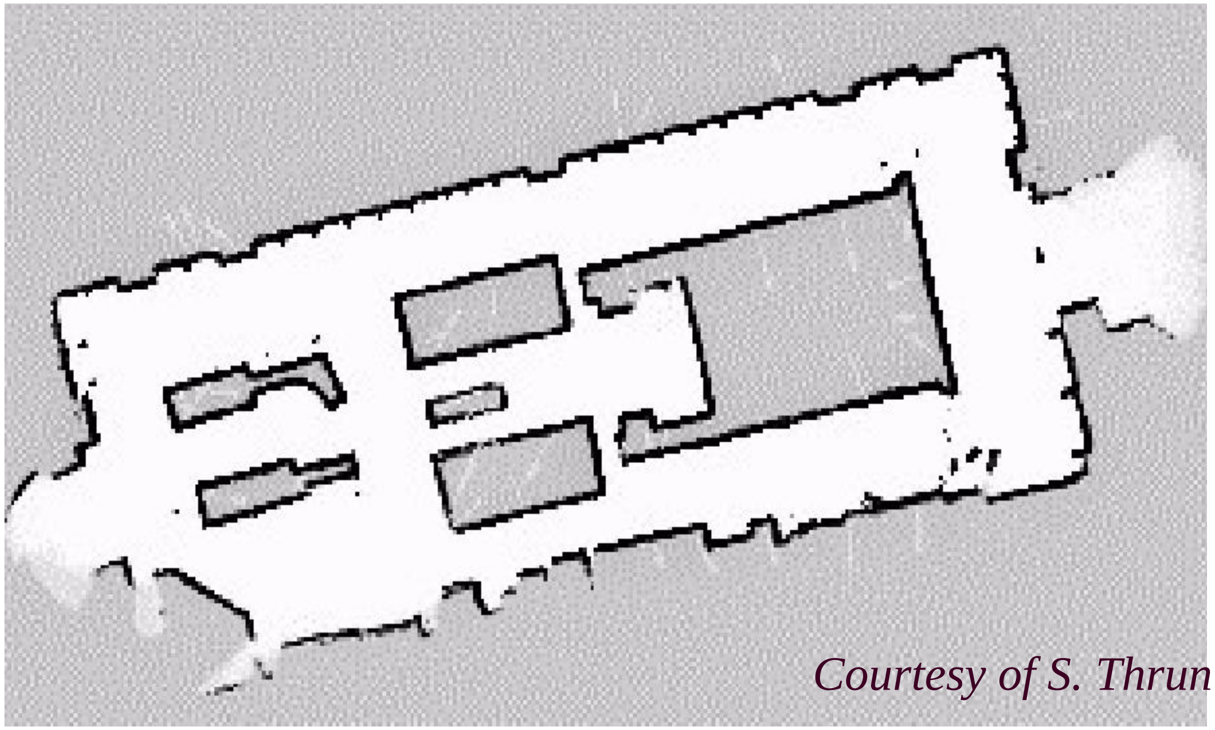
- Exercise: how do we implement an adaptive cell decomposition algorithm?



ETH Zürich

Map Representation
Occupancy grid

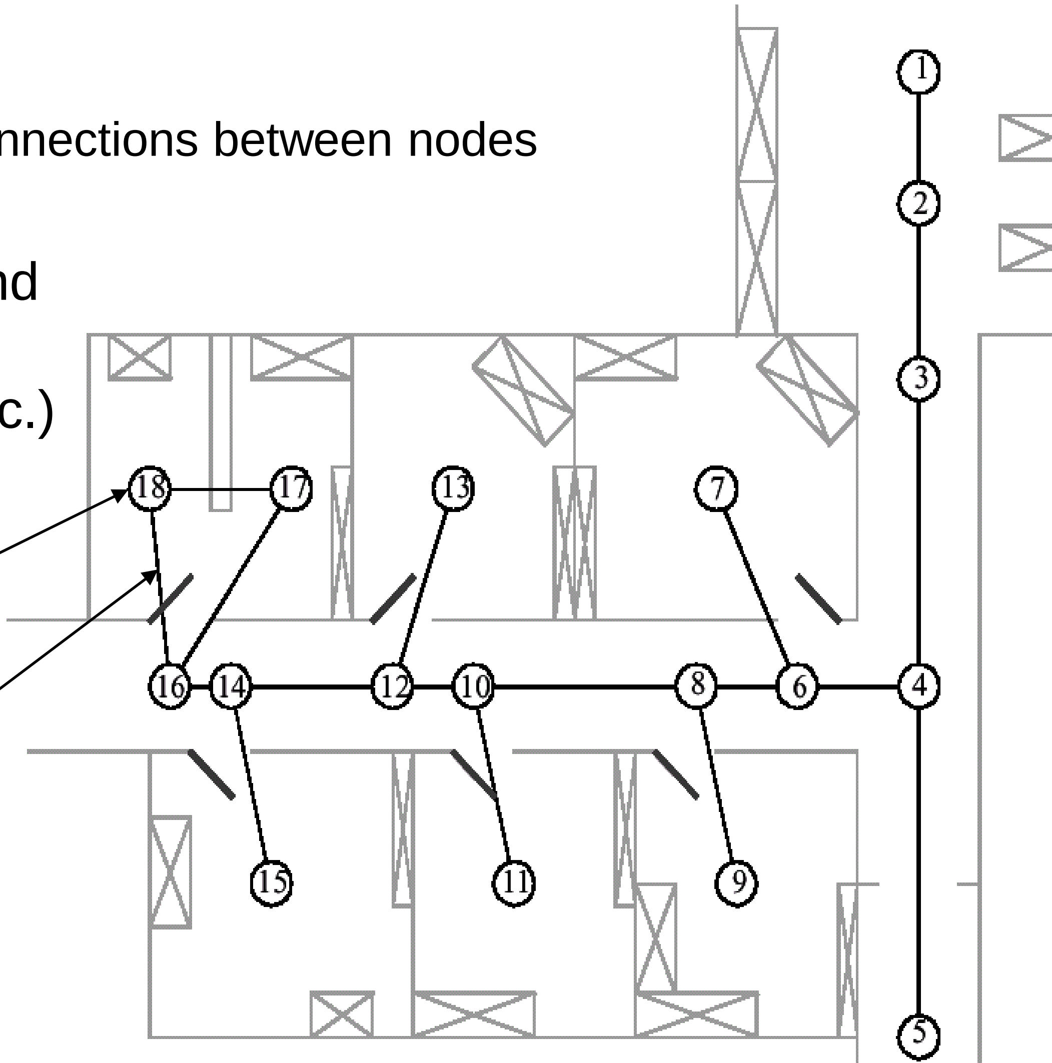
- Fixed cell decomposition: occupancy grid example
  - In occupancy grids, each cell may have a counter where 0 indicates that the cell has not been hit by any ranging measurements and therefore it is likely free-space. As the number of ranging strikes increases, the cell value is incremented and, above a certain threshold, the cell is deemed to be an obstacle
  - The values of the cells are discounted when a ranging strike travels through the cell. This allows us to represent "transient" (dynamic) obstacles



Map Representation
Topological map

- A topological map represents the environment as a graph with nodes and edges.
  - Nodes correspond to spaces
  - Edge correspond to physical connections between nodes

- Topological maps lack scale and distances, but topological relationships (e.g., left, right, etc.) are mantained

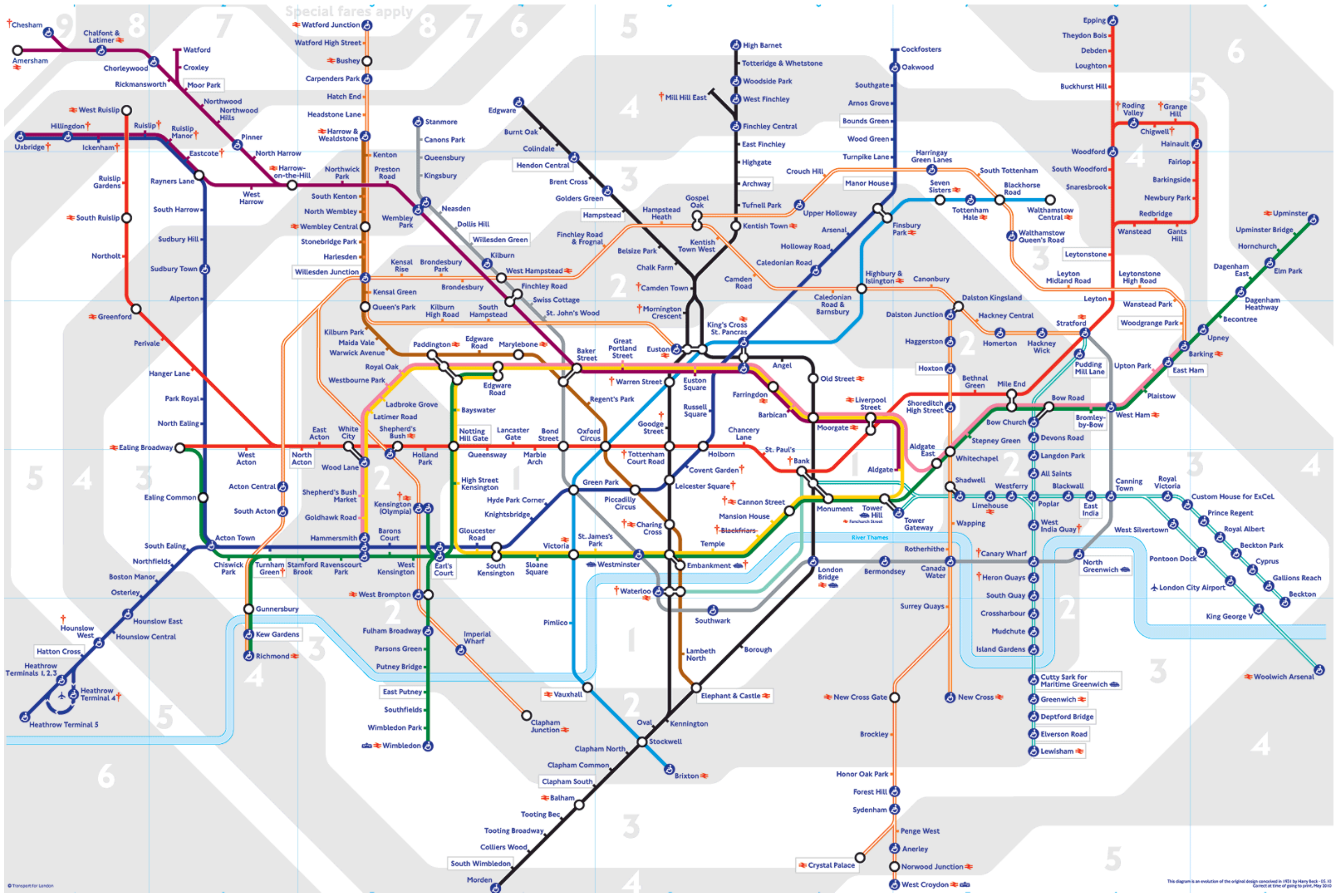


node
(location)

edge
(connectivity)

Map Representation
Topological map

- London underground map



Probabilistic Map Based Localization

Probabilistic Map
Based Localization

Probabilistic Map Based Localization *Outline*

- Definition and illustration of probabilistic map-based localization

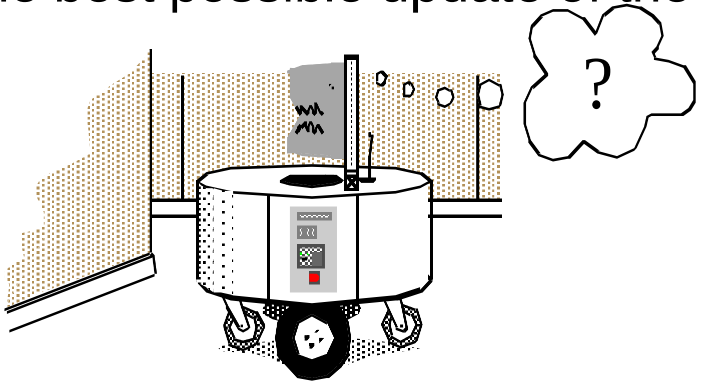
- Two solutions
  - Markov localization (today)
  - Kalman filter localization (next lecture)

Further references:

Thrun, Fox, Burgard, “Probabilistic Robotics,” MIT Press, 2005.

Probabilistic Map Based Localization
The problem

- Consider a mobile robot moving in a known environment.
- As it starts to move (say from a precisely known location) it can keep track of its location using wheel odometry.
- However, after a certain movement the robot will get very uncertain about its position => Thus, it should update its position by making “observations” of the environment
- Observations lead to an estimate of the robot position which can then be fused with the odometric estimation to get the best possible update of the actual robot’s position.



Probabilistic Map Based Localization
Working Principle: Improving belief state by moving

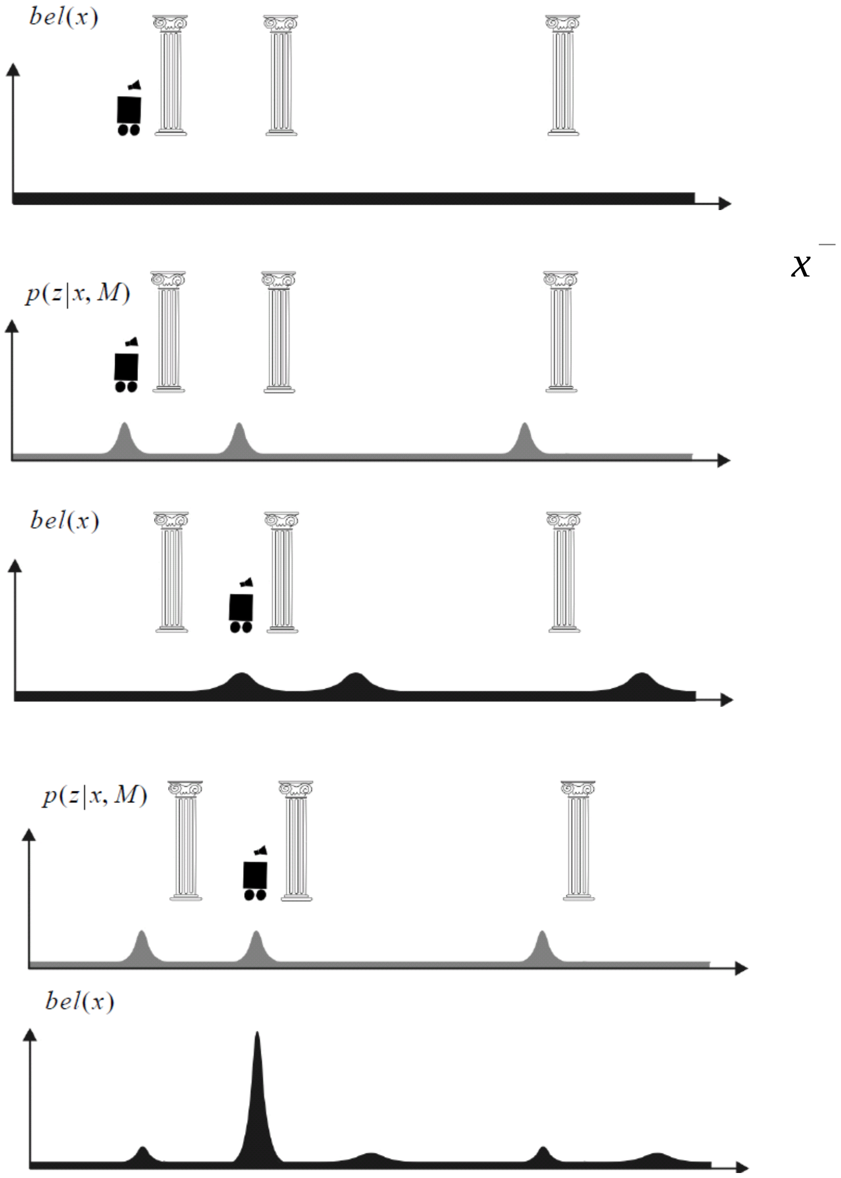
The robot is placed somewhere in the environment but is not told its location

The robot queries its sensors and finds it is next to a pillar

The robot moves one meter forward.
To account for inherent noise in robot motion the new belief is smoother

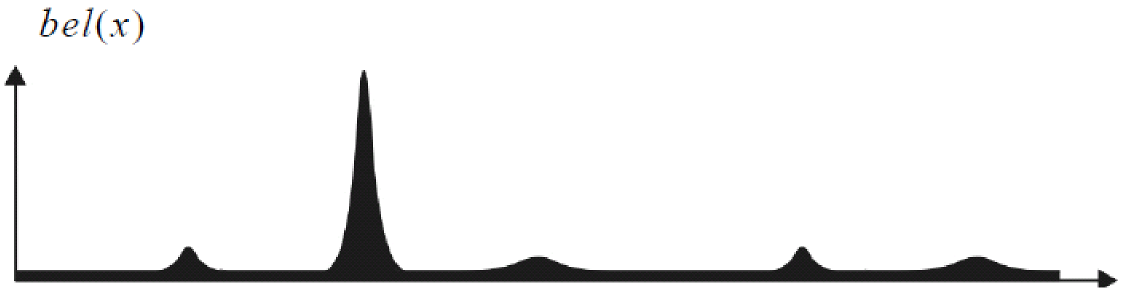
The robot queries its sensors and again it finds itself next to a pillar

Finally, it updates its belief by combining this information with its previous belief



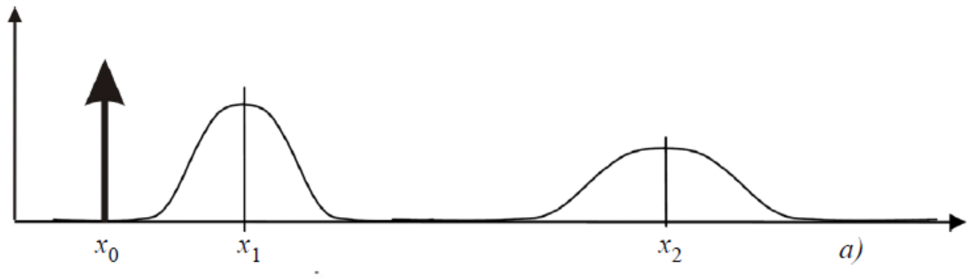
Probabilistic Map Based Localization
Why a probabilistic approach for mobile robot localization?

- Because the data coming from the robot sensors are affected by measurement errors, we can only compute the probability that the robot is in a given configuration.
- The key idea in probabilistic robotics is to represent uncertainty using probability theory: instead of giving a single best estimate of the current robot configuration, probabilistic robotics represents the robot configuration as a probability distribution over the all possible robot poses. This probability is called *belief*.

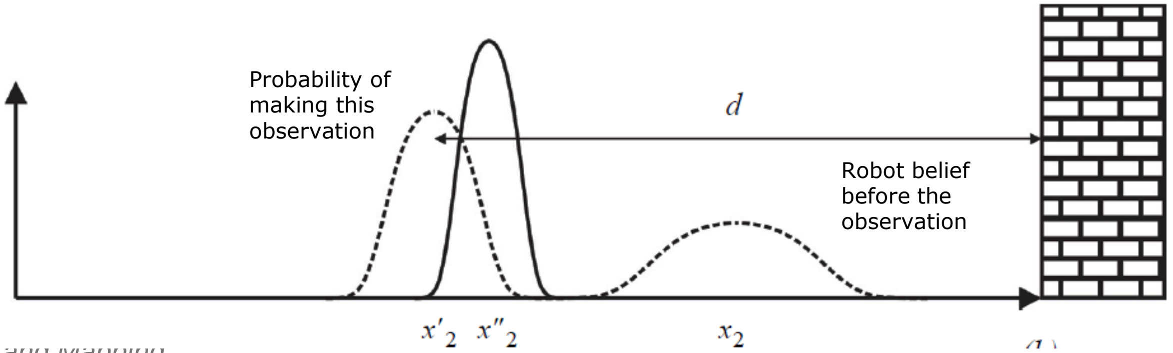


Description of the probabilistic localization problem Action and perception updates

- In robot localization, we distinguish two update steps:
  1. ACTION (or prediction) update:
     - the robot moves and estimates its position through its **proprioceptive** sensors.
       During this step, the robot uncertainty grows.



  2. PERCEPTION (or meausurement) update:
     - the robot makes an observation using its **exteroceptive** sensors and correct its position by opportunely combining its belief before the observation with the probability of making exactly that observation.
       During this step, the robot uncertainty shrinks.



Solution to the probabilistic localization problem

A probabilistic approach to the mobile robot localization problem is a method able to compute the probability distribution of the robot configuration during each Action and Perception step.

The ingredients are:

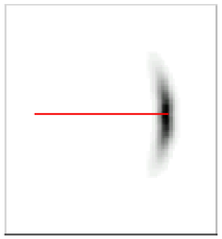
1. The initial probability distribution $p(x)_{t=0}$

2. The statistical error model of the proprioceptive sensors (e.g. wheel encoders)

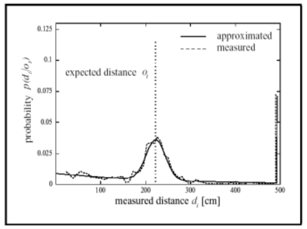
3. The statistical error model of the exteroceptive sensors (e.g. laser, sonar, camera)

4. Map of the environment
(If the map is not known a priori then the robot needs to build a map of the environment and then localize in it. This is called SLAM, Simultaneous Localization And Mapping)

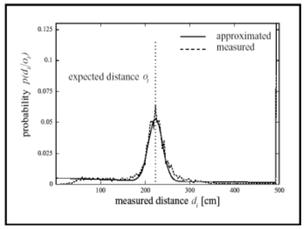
$$p' = f(x, y, \theta, \Delta s_r, \Delta s_l) = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{\Delta s_r + \Delta s_l}{2}\cos\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r + \Delta s_l}{2}\sin\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r - \Delta s_l}{b} \end{bmatrix}$$



$$\Sigma_\Delta = \begin{bmatrix} k_r|\Delta s_r| & 0 \\ 0 & k_l|\Delta s_l| \end{bmatrix}$$



Ultrasound.



Laser range-finder.

Solution to the probabilistic localization problem

How do we solve the Action and Perception updates?

- Action update uses the Theorem of Total probability

$$p(x) = \int_y p(x|y)p(y)dy$$

- Perception update uses the Bayes rule

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

(because of the use of the Bayes rule, probabilistic localization is also called *Bayesian localization*)

Solution to the probabilistic localization problem *Action update*

- In this phase the robot estimates its current position $\overline{bel}(x_t)$ based on the knowledge of the previous position $bel(x_{t-1})$ and the odometric input u_t (The hypothesis that the current robot position depends only on the previous position and the odometric input is called Markov assumption)

- Using the Theorem of Total probability, we compute the robot's belief after the motion as

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

which can be written as a *cross-correlation (sliding inner product)*

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

Solution to the probabilistic localization problem *Perception update*

- In this phase the robot corrects its previous position (i.e. its former belief) by opportunely combining it with the information from its exteroceptive sensors

$$p(x_t \mid z_t) = \frac{p(z_t \mid x_t)\, p(x_t)}{p(z_t)}$$

$$\Downarrow$$

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

where η is a normalization factor which makes the probability integrate to 1.

Solution to the probabilistic localization problem *The algorithm*

```
for all $x_t$ do
    $\overline{bel}(x_t) = \int p(x_t | u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}$   (prediction update)
    $bel(x_t) = \eta p(z_t | x_t, M) \overline{bel}(x_t)$   (measurement update)
endfor
return $bel(x_t)$
```

This localization algorithm is also called Bayes filter

Illustration of probabilistic bap based localization

Initial probability distribution

$p(x)_{t=0}$

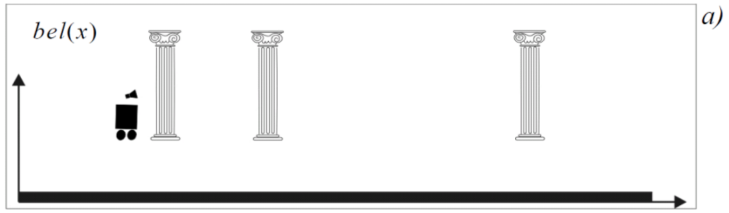


Illustration of probabilistic bap based localization

Initial probability distribution

$$p(x)_{t=0}$$

$$p(z_t \mid x_t)$$

Perception update

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

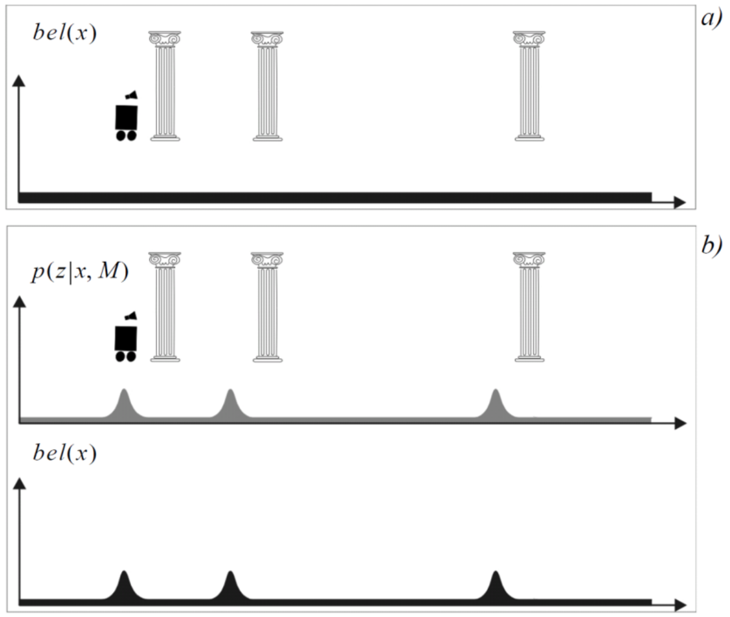


Illustration of probabilistic bap based localization

Initial probability distribution

$$p(x)_{t=0}$$

$$p(z_t \mid x_t)$$

Perception update

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

Action update

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

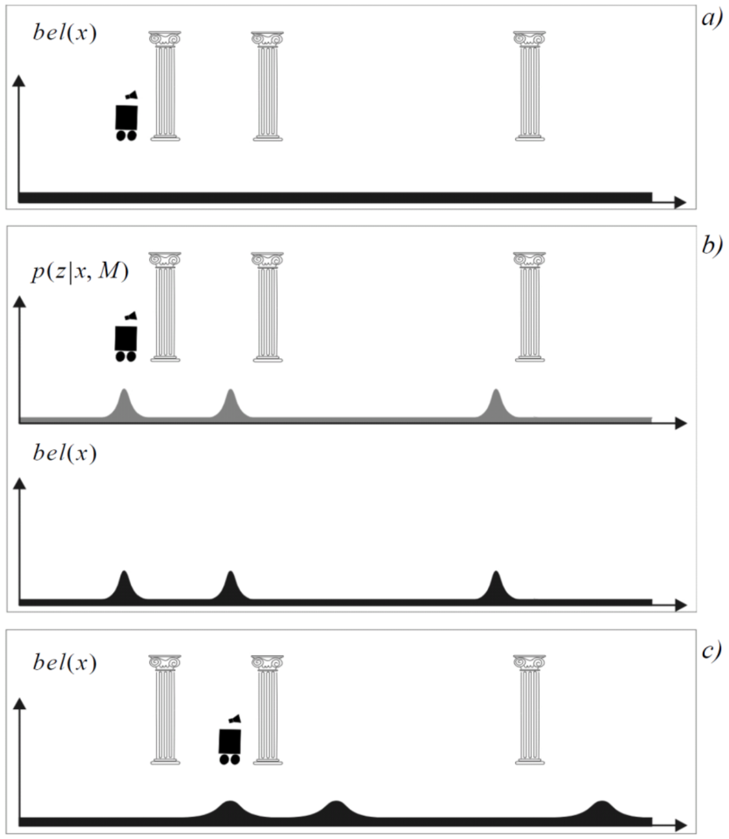


Illustration of probabilistic bap based localization

Initial probability distribution

$$p(x)_{t=0}$$

$$p(z_t \mid x_t)$$

Perception update

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

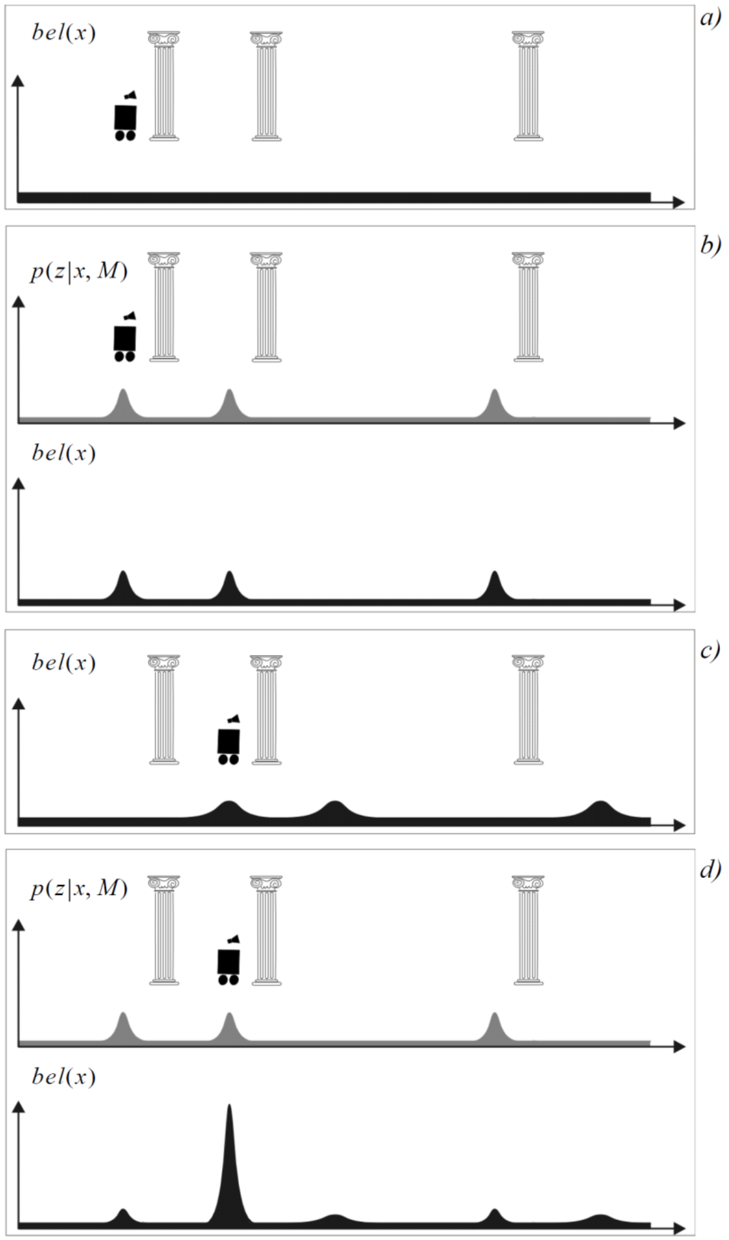
Action update

$$\overline{bel}(x_t) = p(x_t \mid u_t, x_{t-1}) * bel(x_{t-1})$$

$$p(z_t \mid x_t)$$

Perception update

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



Markov versus Kalman localization

Two approaches exist to represent the probability distribution and to compute the Total Probability and Bayes Rule during the Action and Perception phases

Markov	Kalman
• The configuration space is divided into many cells. The configuration space of a robot moving on a plane is 3D dimensional (x,y,θ). Each cell contains the probability of the robot to be in that cell. • The probability distribution of the sensors model is also discrete. • During Action and Perception, all the cells are updated. Therefore, the computational cost is very high	• The probability distribution of both the robot configuration and the sensor model is assumed to be continuous and **Gaussian!** • Since a Gaussian distribution is only described by its mean value μ and covariance Σ, we need only to update μ and Σ. Therefore the computational cost is very low!

Kalman Filter Localization
Markov versus Kalman localization

Markov	Kalman
PROS ▪ localization starting from any unknown position ▪ recovers from ambiguous situation **CONS** ▪ However, 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 very important if a fine grid is used.	**PROS** ▪ Tracks the robot and is inherently very precise and efficient **CONS** ▪ If the uncertainty of the robot becomes to large (e.g. collision with an object) the Kalman filter will fail and the position is definitively lost

Simultaneous Localization and Mapping

What is SLAM?

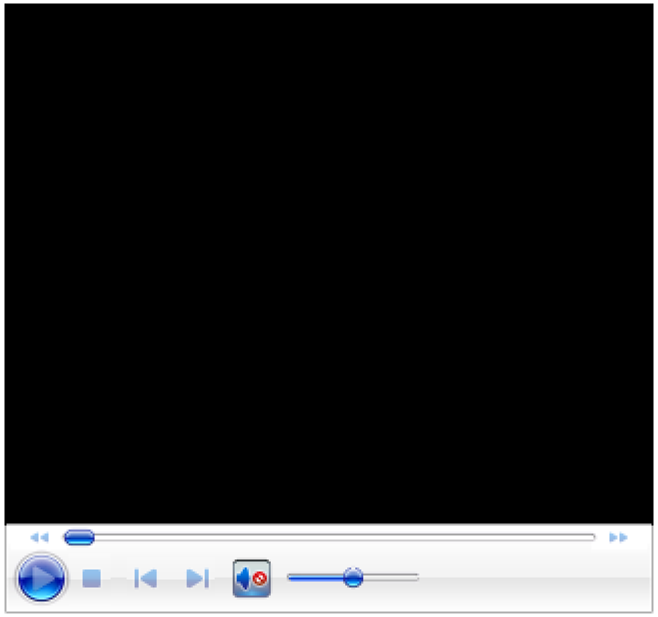
The chicken and egg problem

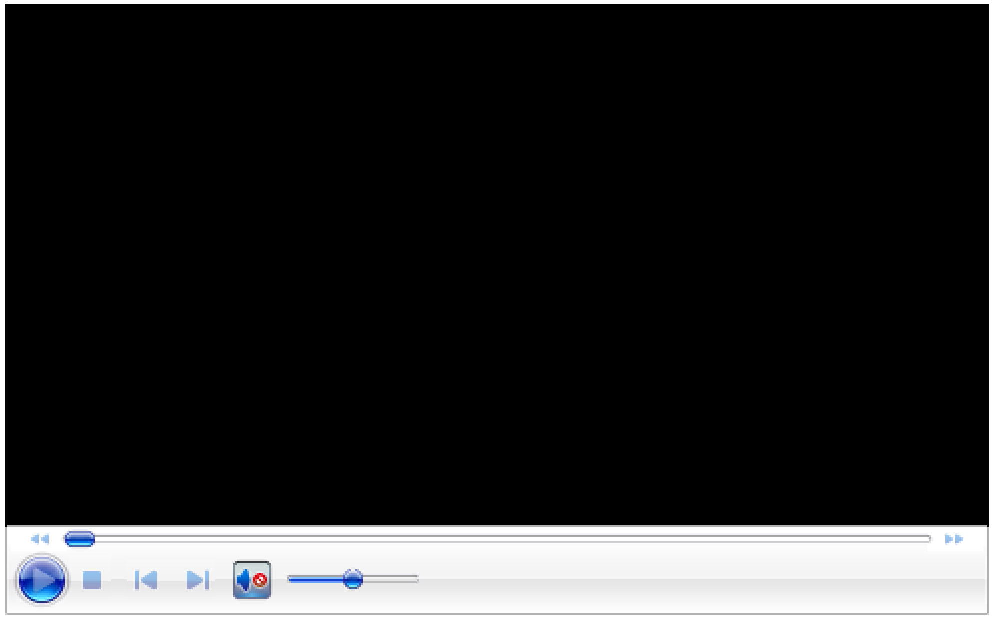
- An unbiased **map** is necessary for localizing the robot

  Pure localization with a known map.
  SLAM: no a priori knowledge of the robot's workspace

- An accurate **pose estimate** is necessary for building a map of the environment

  Mapping with known robot poses.
  SLAM: the robot poses have to be estimated along the way

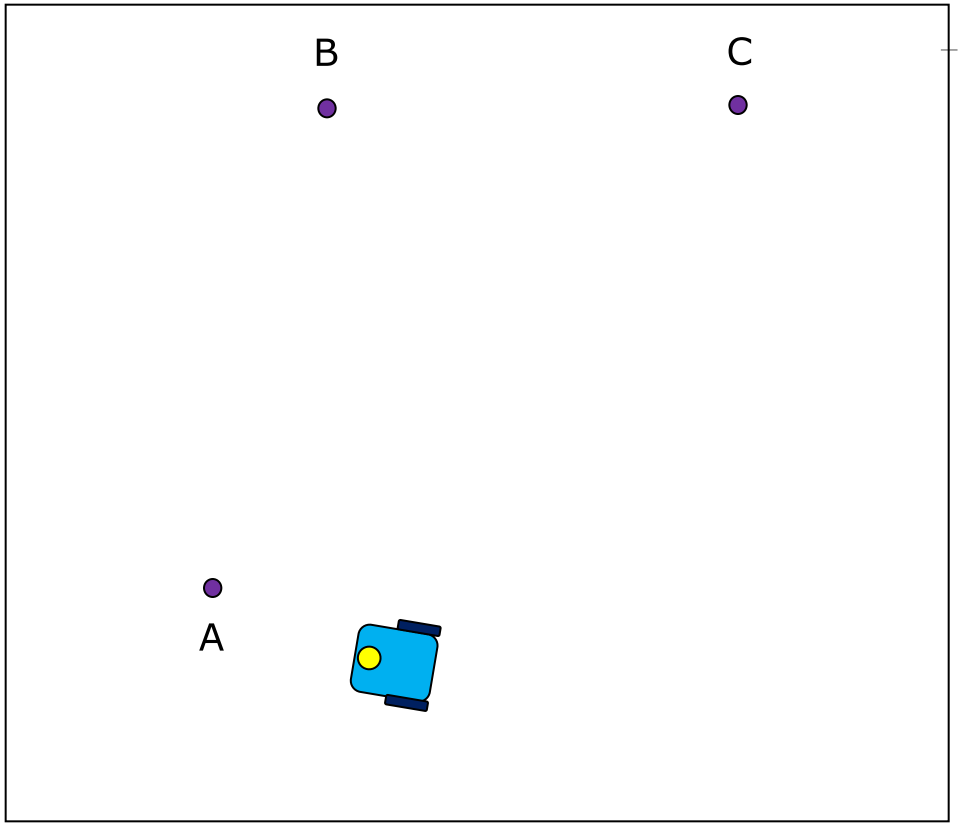

Localization using satellite images
[Senlet and Elgammal, ICRA 2012]


Helicopter pose given by Leica tracker
Video courtesy of Simon Lynen

- SLAM: one of the greatest challenges in probabilistic robotics

How to do SLAM

- Use internal representations for
  - the positions of landmarks (: map)
  - the camera parameters
- Assumption: Robot's uncertainty at starting position is zero

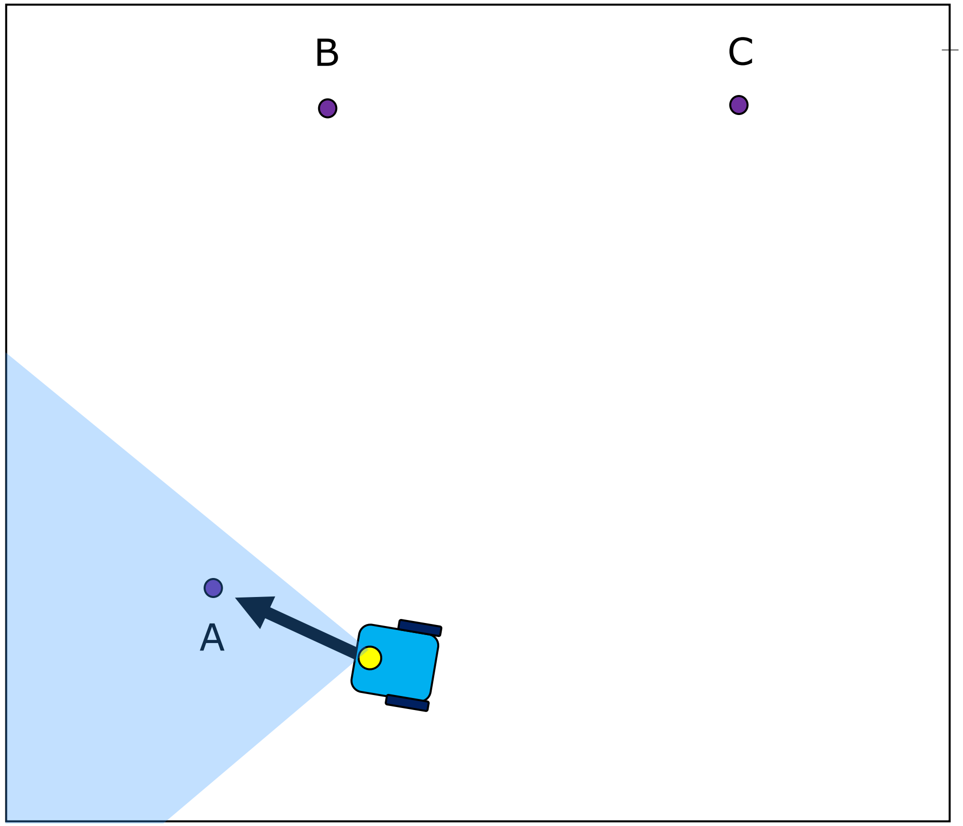


Start: robot has zero uncertainty

How to do SLAM

On every frame:

- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



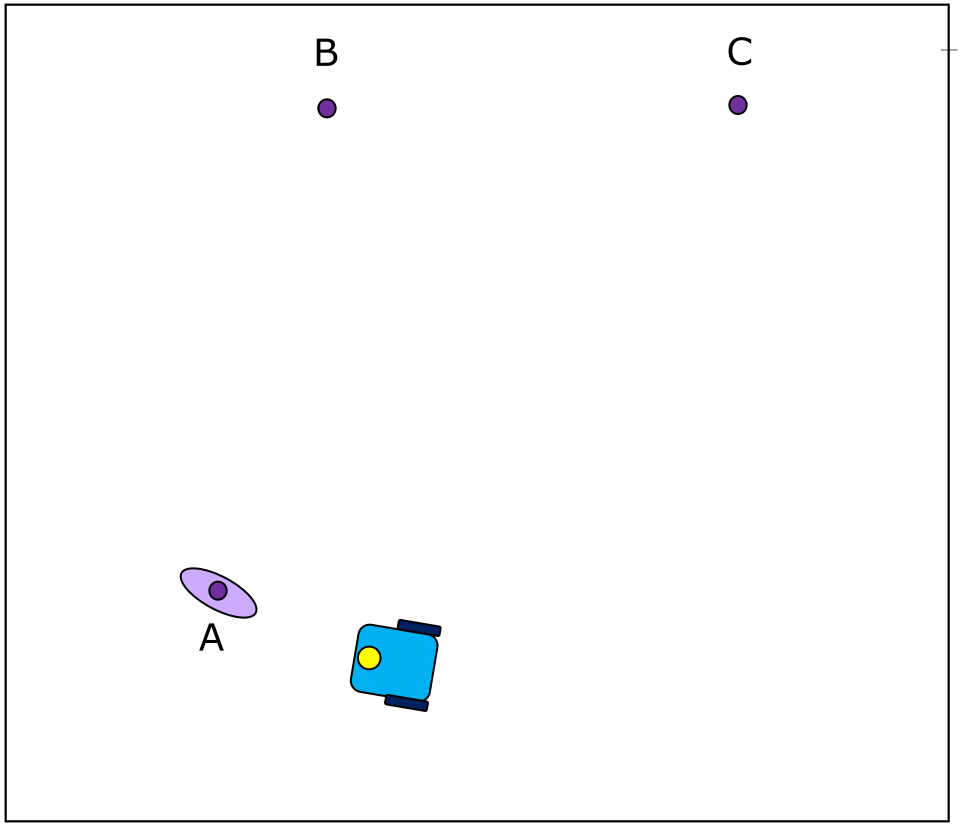
First measurement of feature A

How to do SLAM

- The robot observes a feature which is mapped with an uncertainty related to the **measurement model**

On every frame:

- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

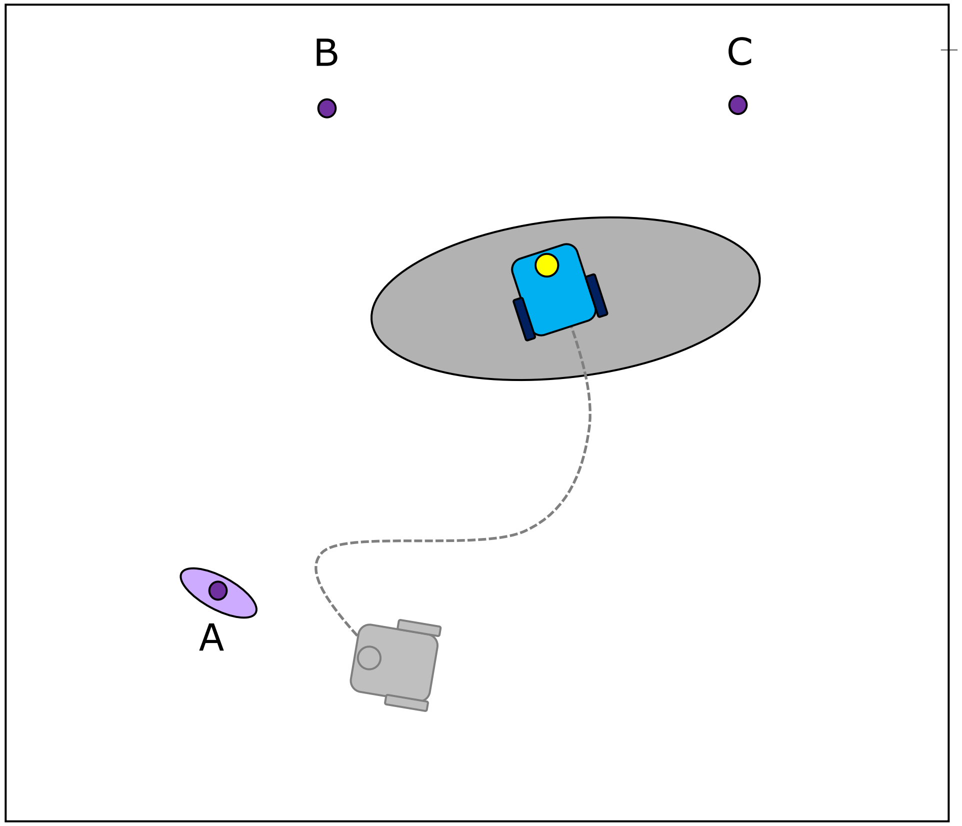


How to do SLAM

- As the robot moves, its pose uncertainty increases, obeying the robot's **motion model**.

On every frame:

- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



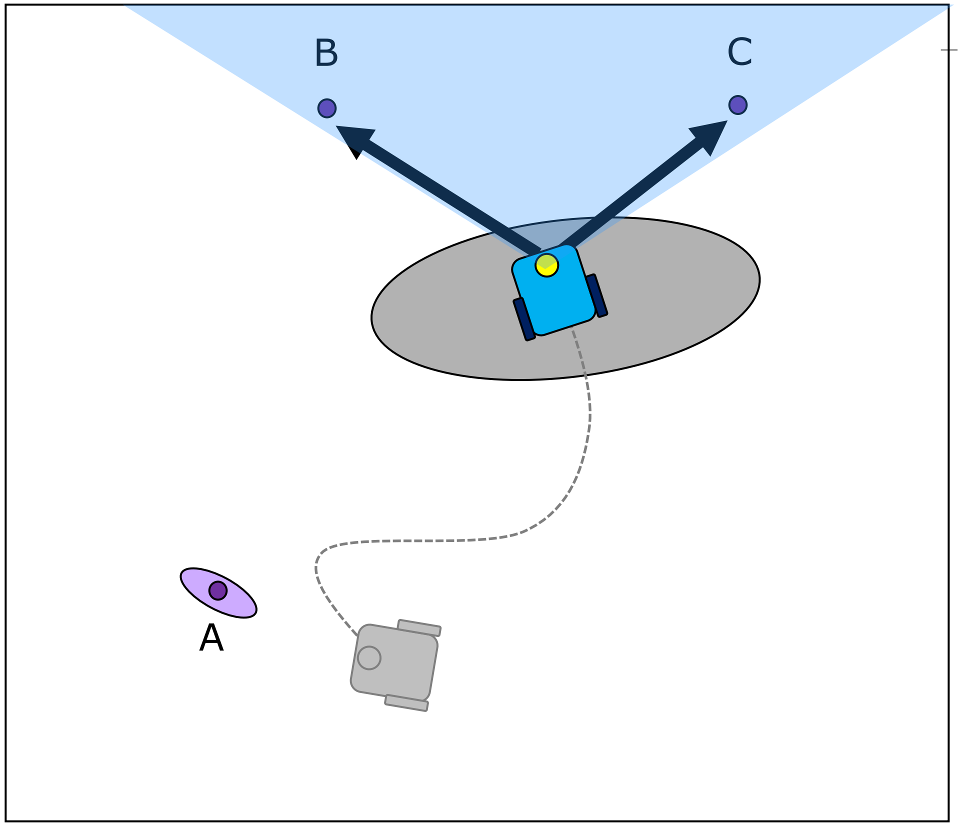
Robot moves forwards: uncertainty grows

How to do SLAM

- Robot observes two new features.

On every frame:

- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



Robot makes first measurements of B & C

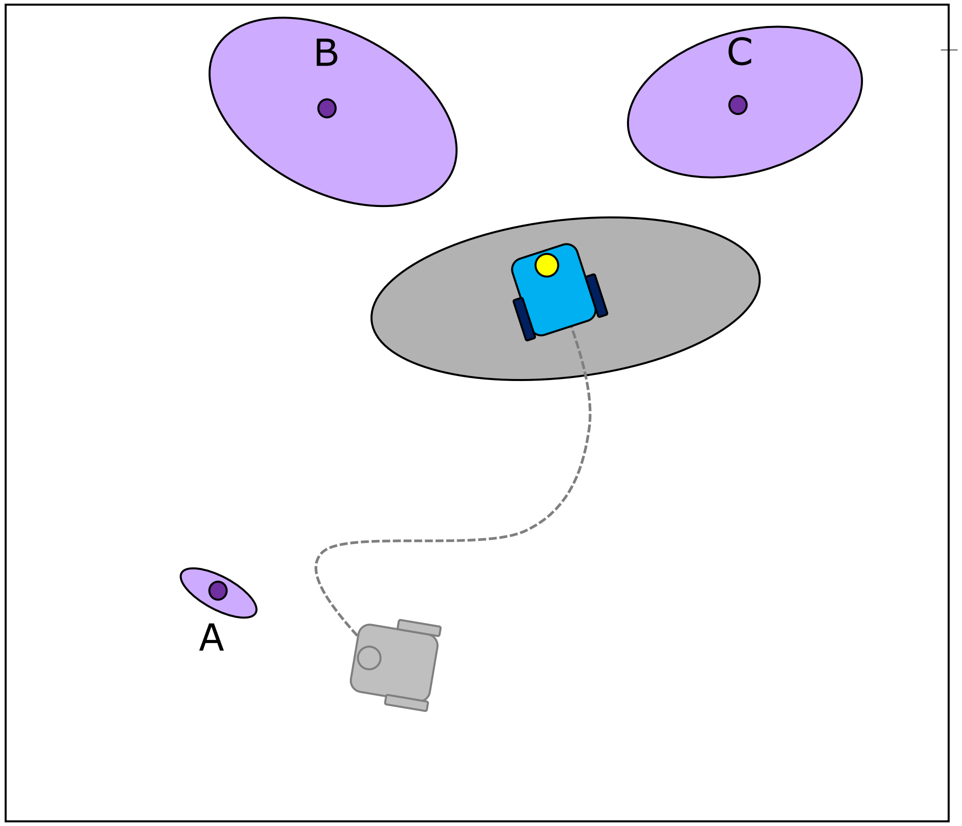
How to do SLAM

- Their position uncertainty results from the **combination** of the measurement error with the robot pose uncertainty.

⇨ map becomes **correlated** with the robot pose estimate.

On every frame:

- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



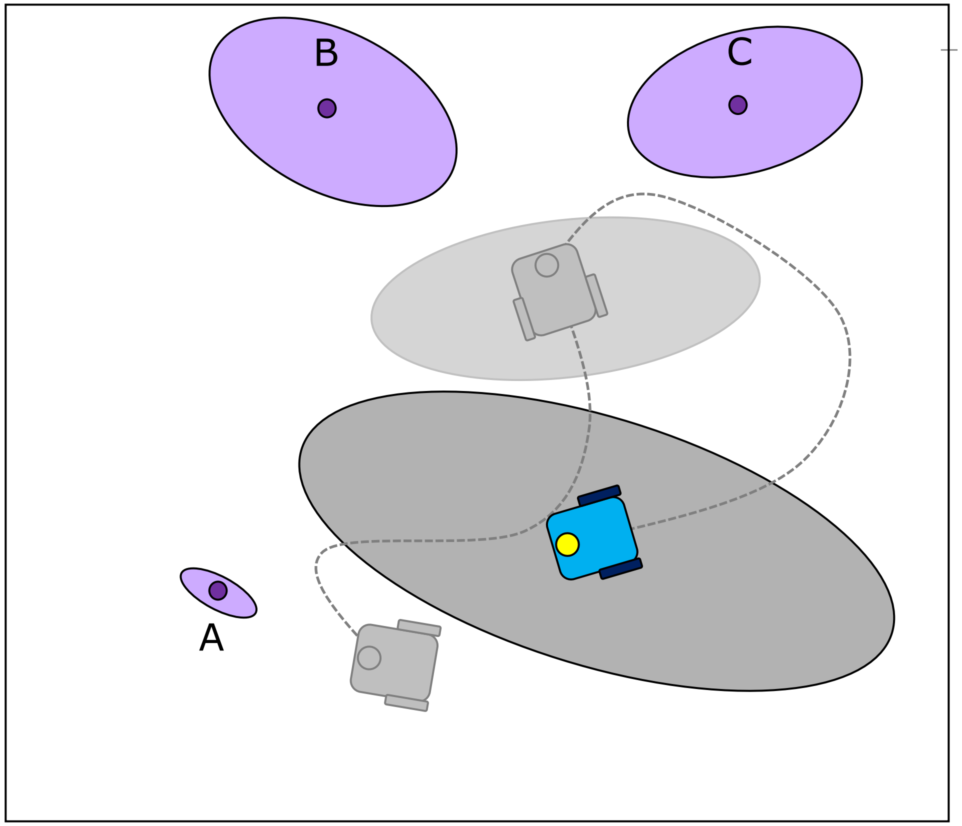
Robot makes first measurements of B & C

How to do SLAM

- Robot moves again and its uncertainty increases (motion model)

On every frame:

- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



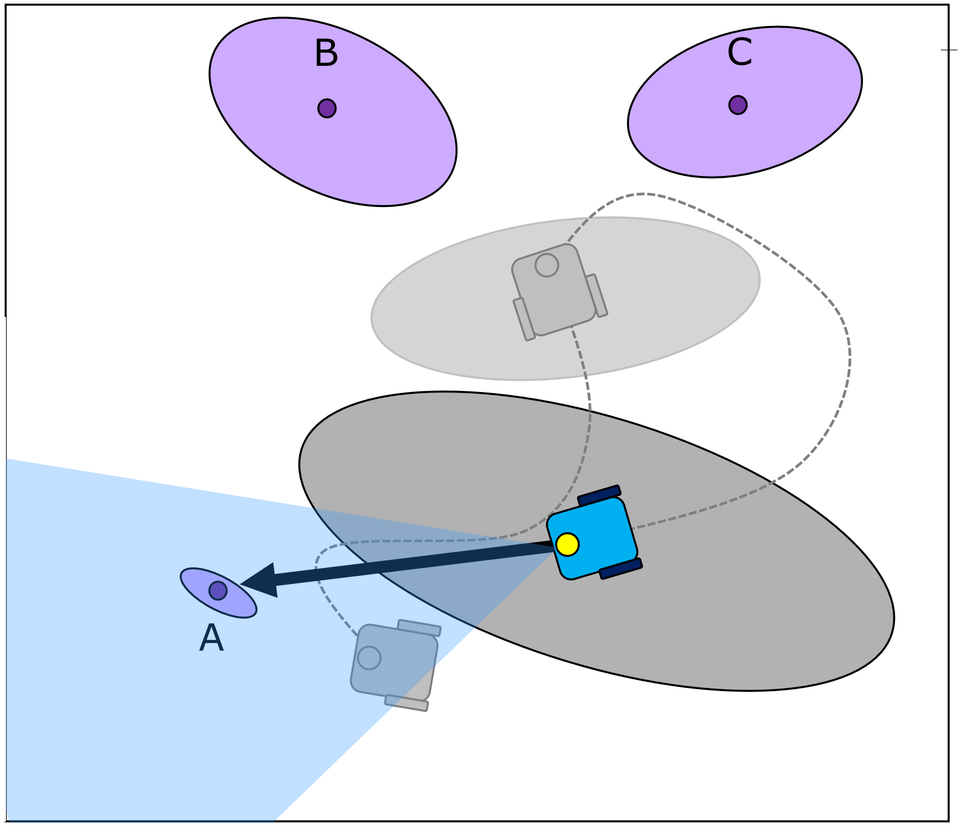
Robot moves again: uncertainty grows more

How to do SLAM

- Robot re-observes an old feature
  ➪ **Loop closure** detection

On every frame:

- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

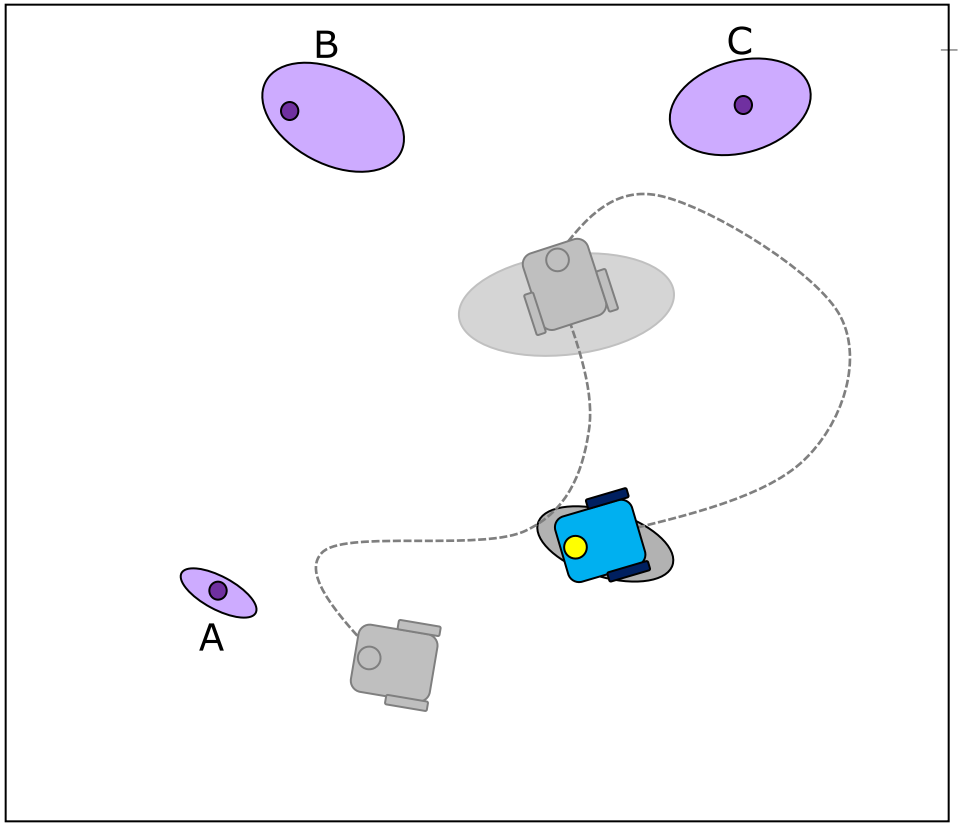


Robot re-measures A: "loop closure"

How to do SLAM

- Robot updates its position: the resulting **pose** estimate becomes **correlated** with the feature **location estimates**.
- Robot’s uncertainty **shrinks** and so does the uncertainty in the rest of the map

On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



Robot re-measures A: “loop closure” uncertainty shrinks

SLAM: Probabilistic Formulation

- Robot **pose** at time t : x_t ⇨ Robot **path** up to this time: $\{x_0, x_1, \ldots, x_t\}$
- Robot **motion** between time $t\text{-}1$ and t : u_t (control inputs / proprioceptive sensor readings) ⇨ Sequence of robot relative motions: $\{u_0, u_1, \ldots, u_t\}$
- The **true map** of the environment: $\{m_0, m_1, \ldots, m_N\}$
- At each time t the robot makes measurements z_i
  ⇨ Set of all measurements (observations): $\{z_0, z_1, \ldots, z_k\}$
- The Full SLAM problem: estimate the posterior
  $p(x_{0:t}, m_{0:n} \mid z_{0:k}, u_{0:t})$
- The Online SLAM problem: estimate the posterior
  $p(x_t, m_{0:n} \mid z_{0:k}, u_{0:t})$

The Three SLAM paradigms

- Some of the most important approaches to SLAM:
  - Extended Kalman Filter SLAM (EKF SLAM)
  - Particle Filter SLAM (FAST SLAM) – tomorrow’s exercise
  - GraphSLAM