

University of Pisa  
Master of Science in Computer Science  
**Course of Robotics (ROB)**  
A.Y. 2018/19

THE BIORBOTICS  
INSTITUTE



Scuola Superiore  
Sant'Anna

# Fundamentals of robot navigation



**Cecilia Laschi**  
The BioRobotics Institute  
Scuola Superiore Sant'Anna  
[cecilia.laschi@santannapisa.it](mailto:cecilia.laschi@santannapisa.it)





# Robot definition

Scuola Superiore  
Sant'Anna

*A robot* is an autonomous system  
which exists in the physical world,  
can sense its environment,  
and can act on it to achieve some goals

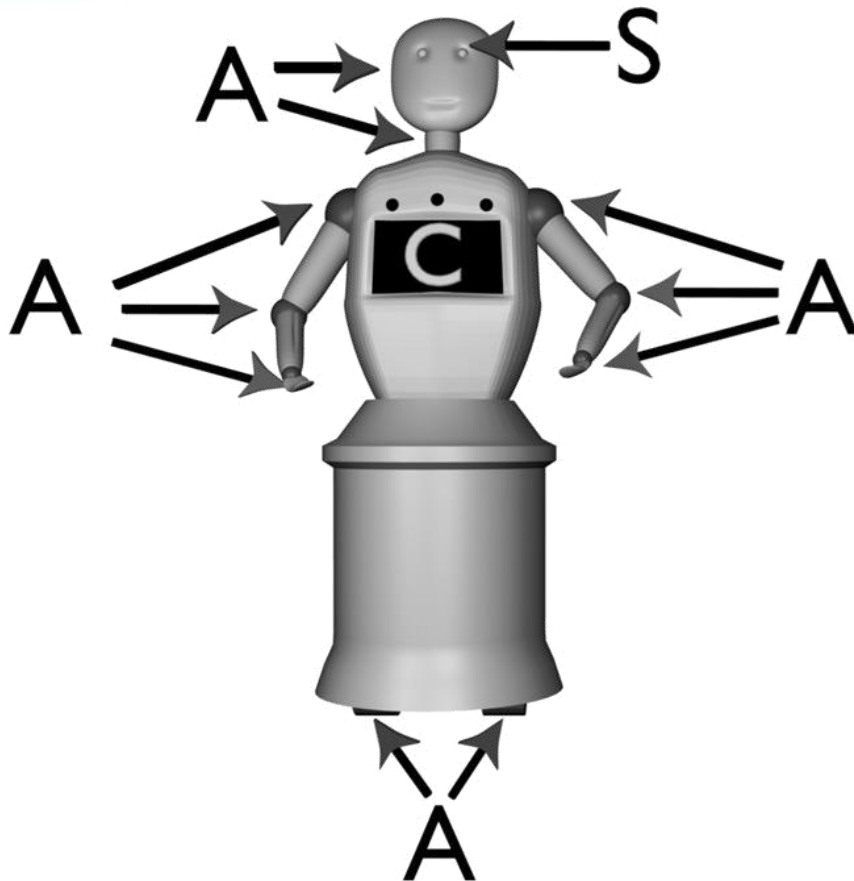




# What's in a robot?

## Robot components

Scuola Superiore  
Sant'Anna



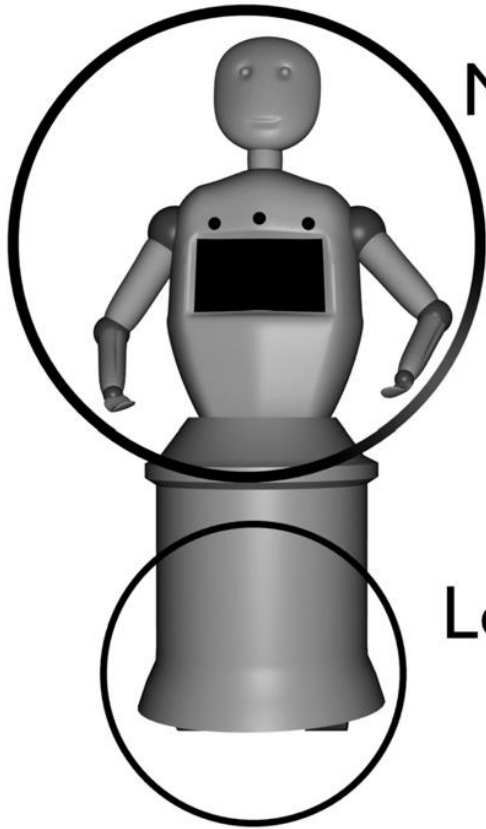
### Legend

Actuator

Controller

Sensor

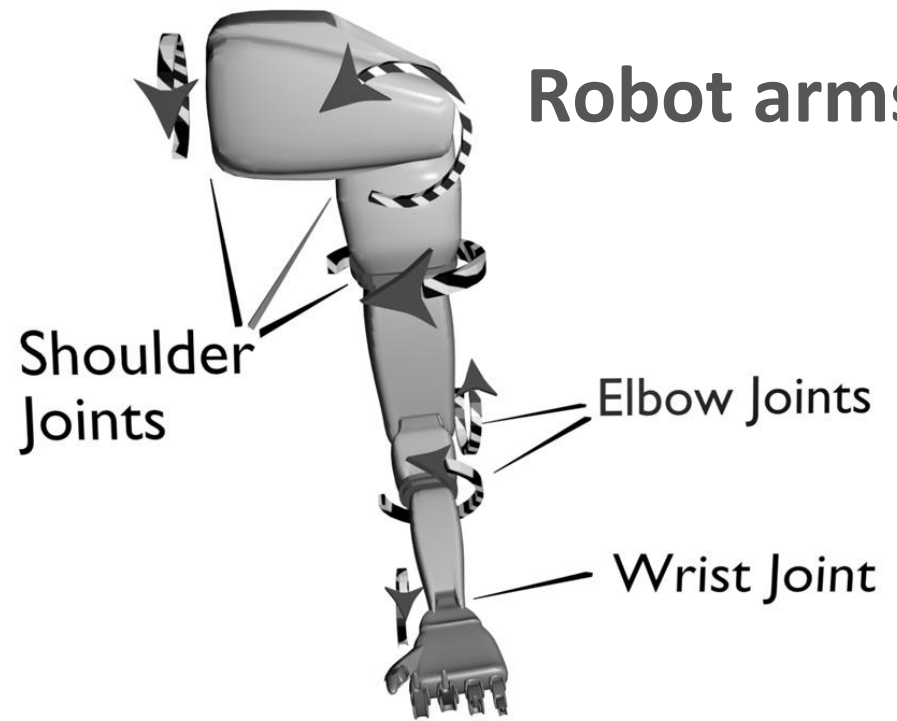
# Effectors



Manipulation

Locomotion

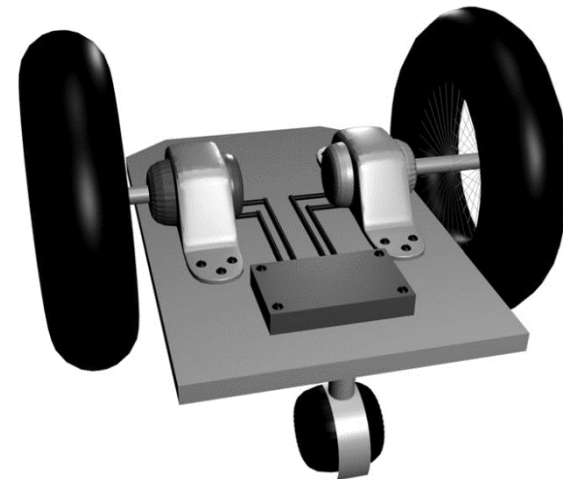
# Robot arms



Shoulder Joints

Elbow Joints

Wrist Joint



# Mobile robots



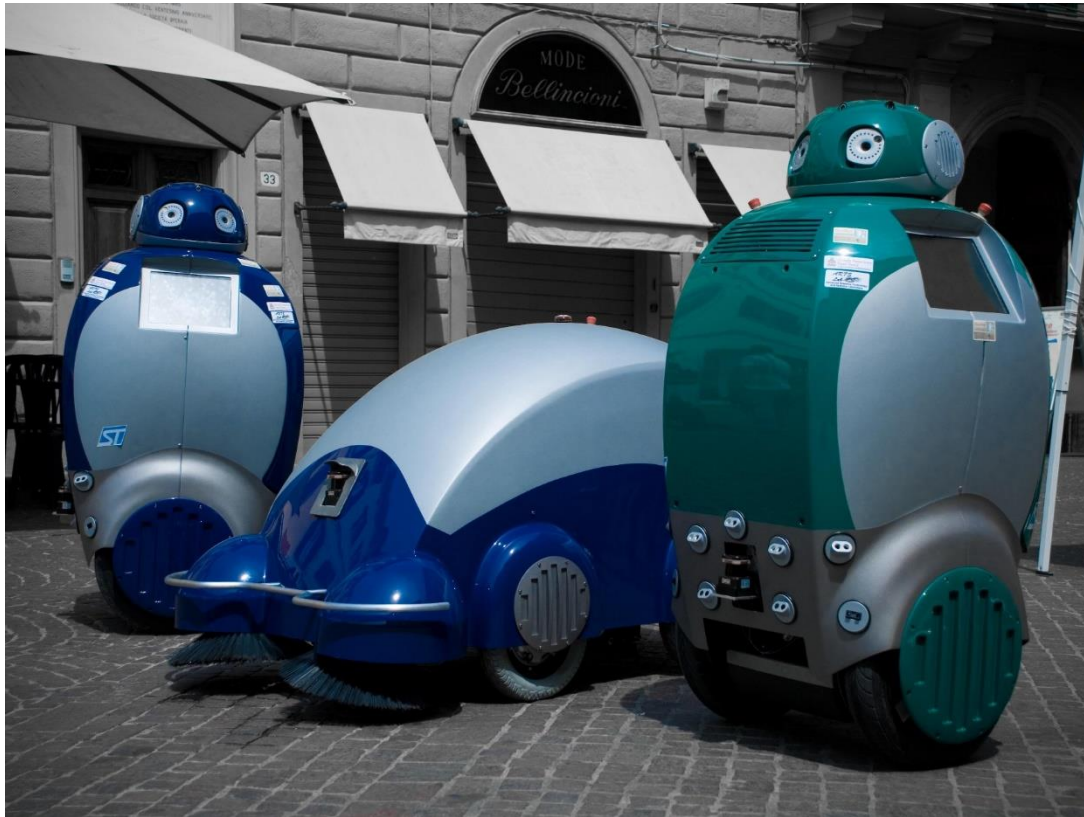


# Outline of the lecture

- Fundamental problems of robot navigation
- Maps and environment models
  - Metric maps and topological maps
- Planning techniques
  - Path Planning and Path Following
- Localization methods and systems
  - Odometry and systems based on active beacons and landmarks



# Mobile robots and autonomous navigation



**DustBot (DustCart & DustClean)**



**Autonomous cars**

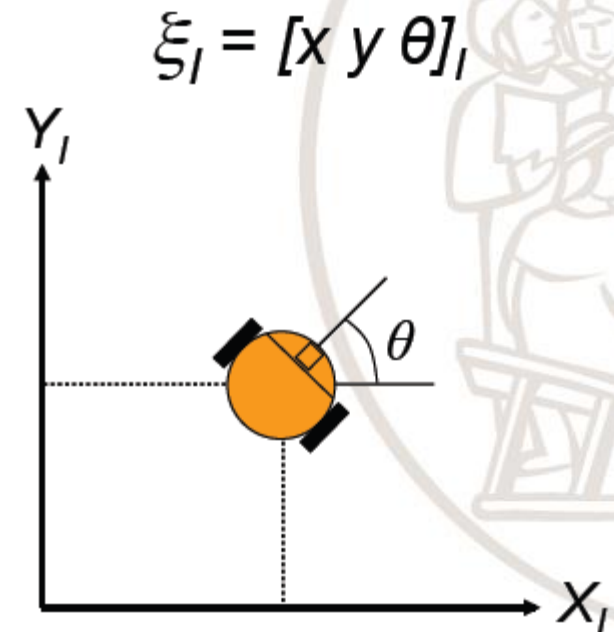
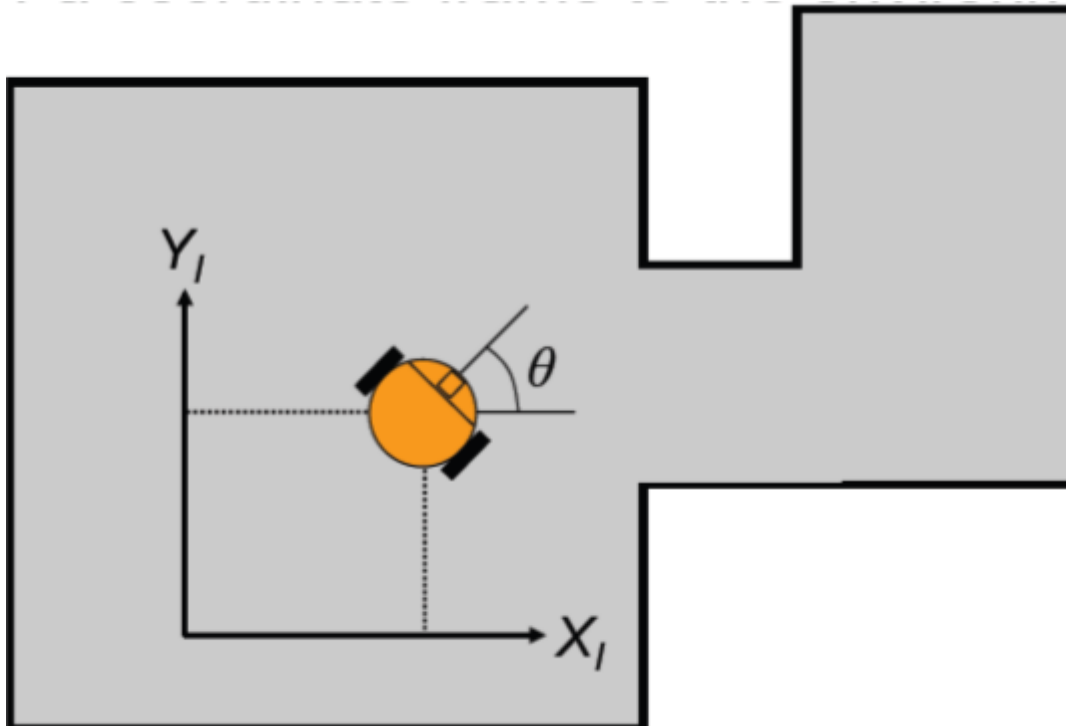




# Position of a mobile robot

Scuola Superiore  
Sant'Anna

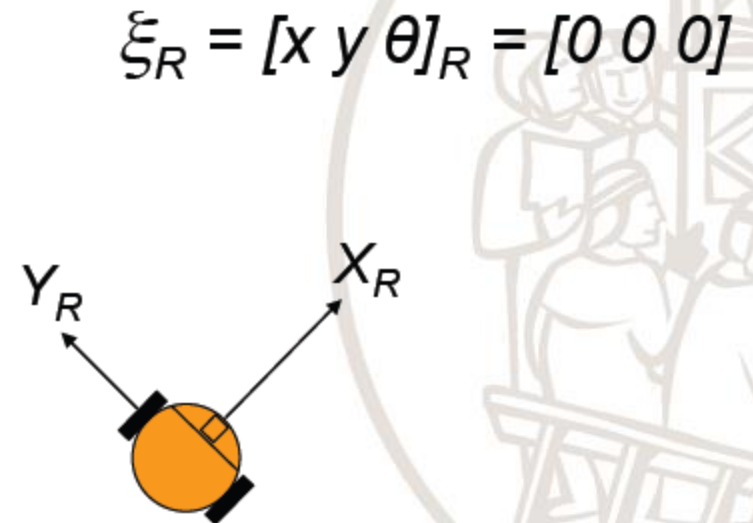
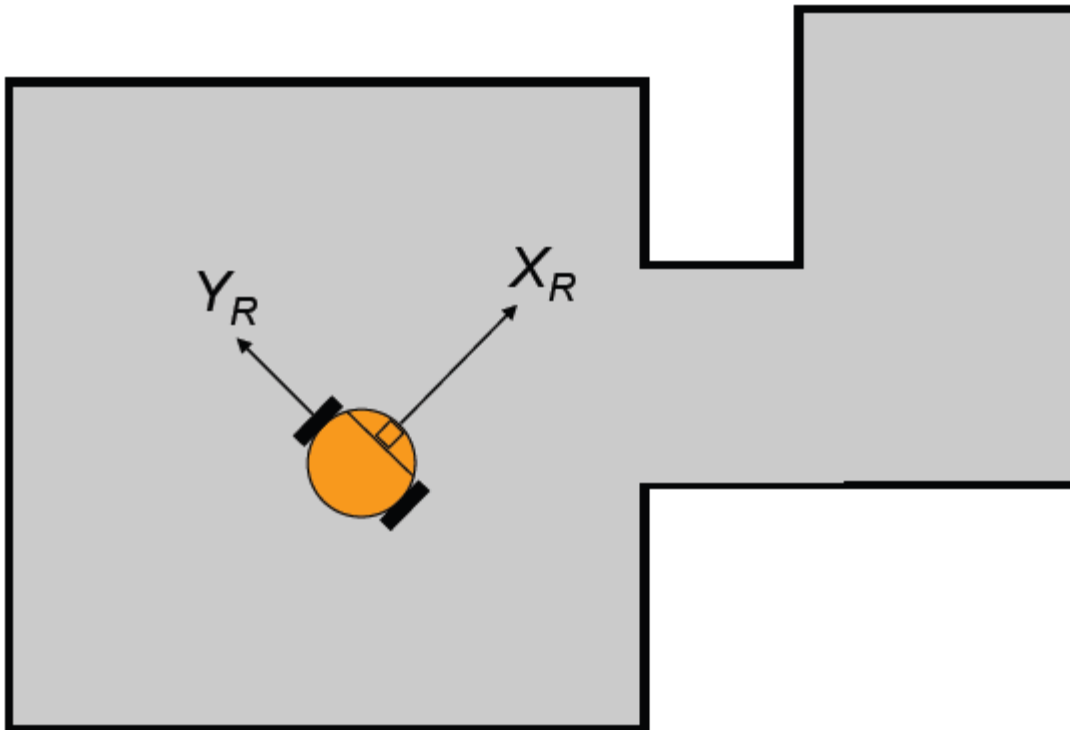
Reference coordinate system fixed in the world





# Position of a mobile robot

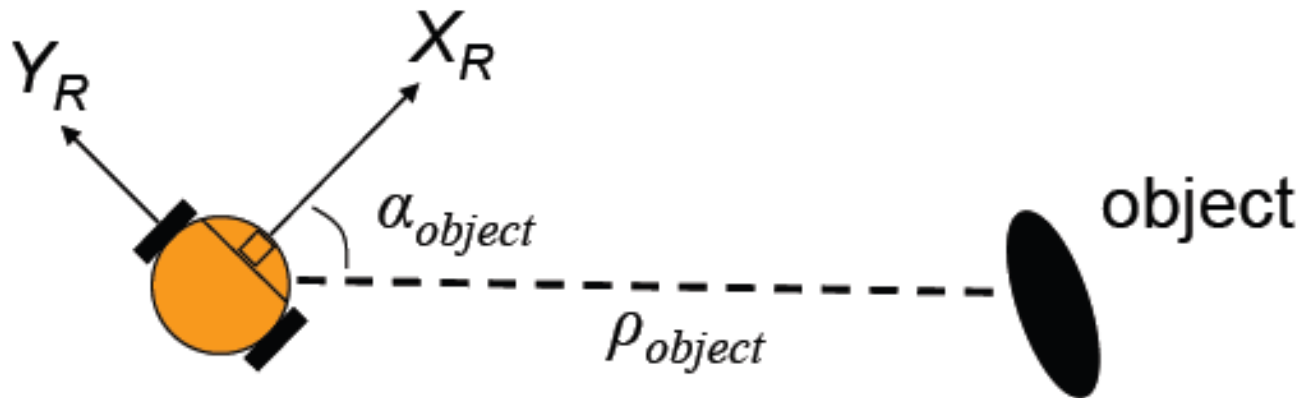
Reference coordinate system fixed on the robot







# Position of an obstacle

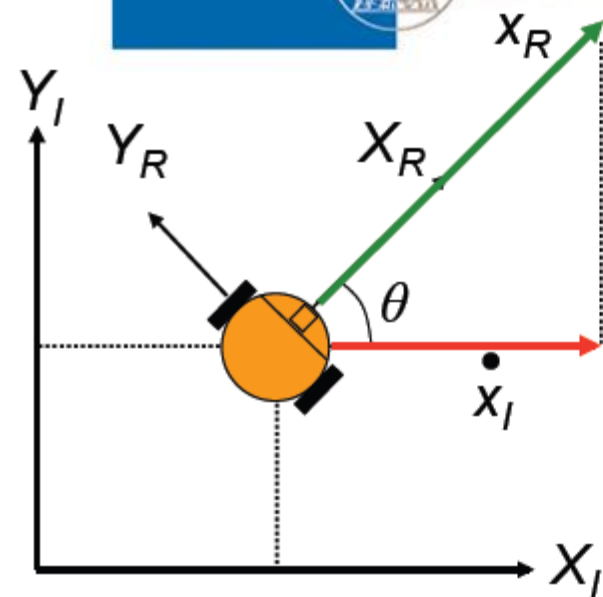


# Transformations between the world reference frame and the robot reference frame, in velocity space

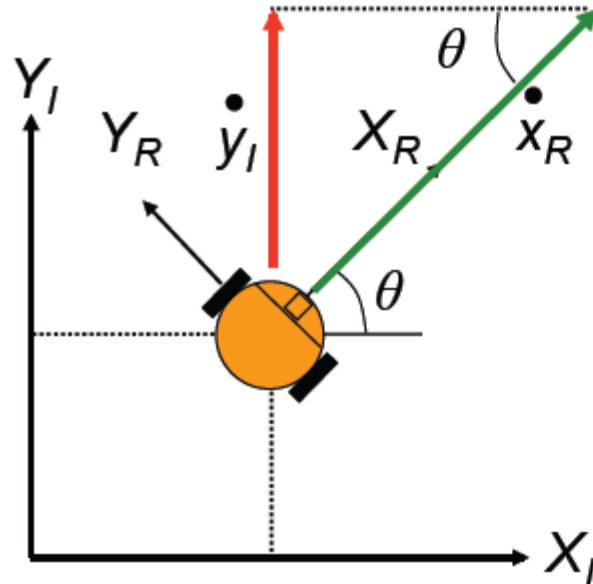
THE BIROBOTICS INSTITUTE



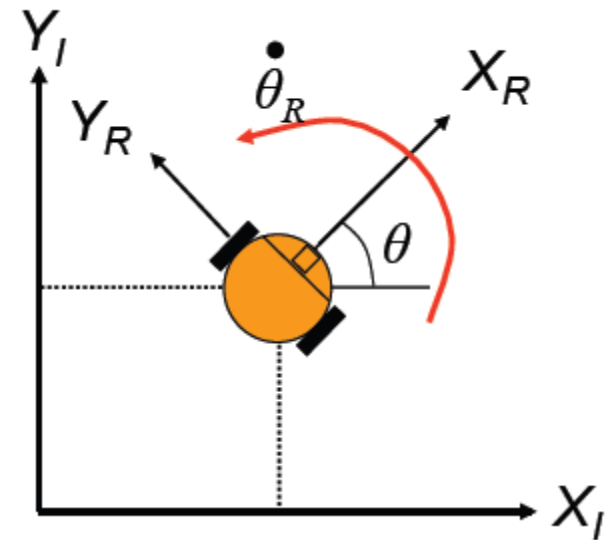
Scuola Superiore Sant'Anna



$$\dot{x}_I = \dot{x}_R \cos(\theta)$$



$$\dot{y}_I = \dot{x}_R \sin(\theta)$$



$$\dot{\theta}_I = \dot{\theta}_R$$

or

$$\begin{aligned} \dot{x}_r &= v \\ \dot{y}_r &= 0 \\ \dot{\theta}_r &= \omega \end{aligned}$$

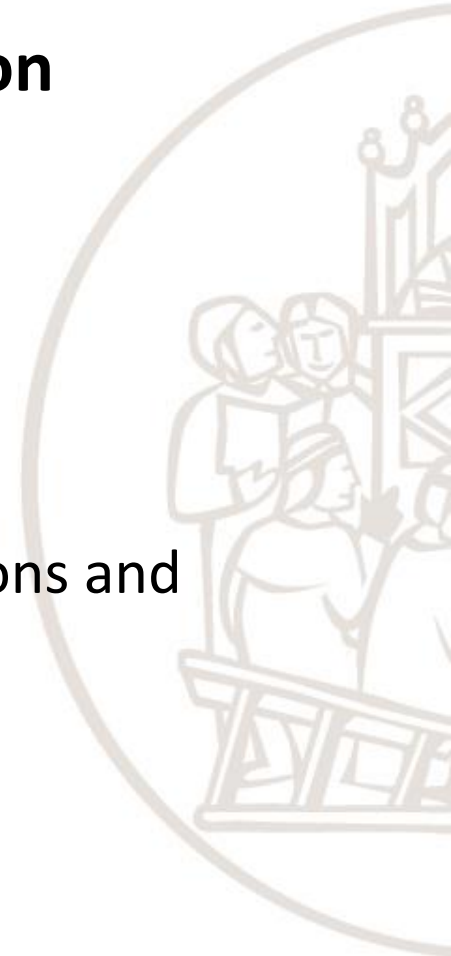
$$\begin{bmatrix} \dot{x}_I \\ \dot{y}_I \\ \dot{\theta}_I \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 & 0 \\ \sin(\theta) & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\theta}_R \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}$$



# Outline of the lecture

Scuola Superiore  
Sant'Anna

- **Fundamental problems of robot navigation**
- Maps and environment models
  - Metric maps and topological maps
- Planning techniques
  - Path Planning and Path Following
- Localization methods and systems
  - Odometry and systems based on active beacons and landmarks





# Robot navigation problem

To reach a final position from a starting position, given in geometric or sensory terms, avoiding obstacles

The classical questions to solve are:

- **Where am I?**
- **Where are the other objects around me?**
- **How can I reach a desired position?**

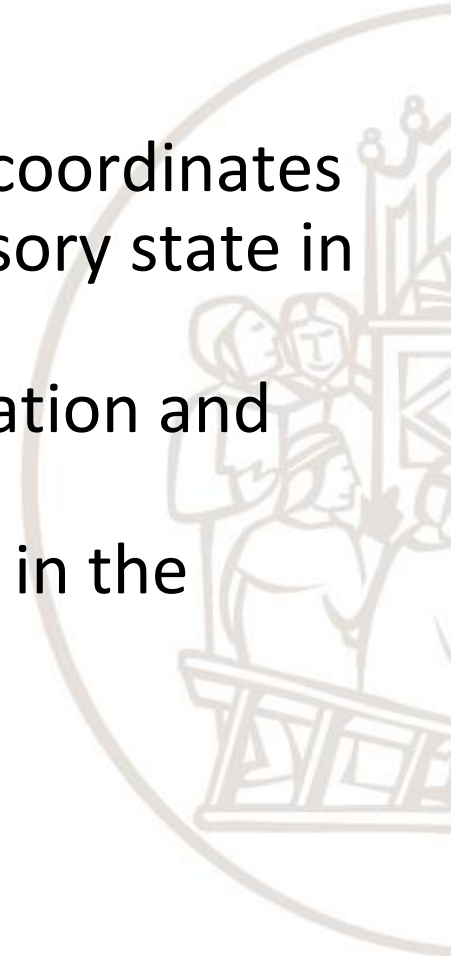




# Classical approach

In the classical approach, the answers are:

- **Localization:** geometric position ( $x, y, \theta$  coordinates in an absolute reference system) or sensory state in the robot environment
- **Maps or Models:** environment formalization and representation
- **Planning:** planning of robot movements in the environment





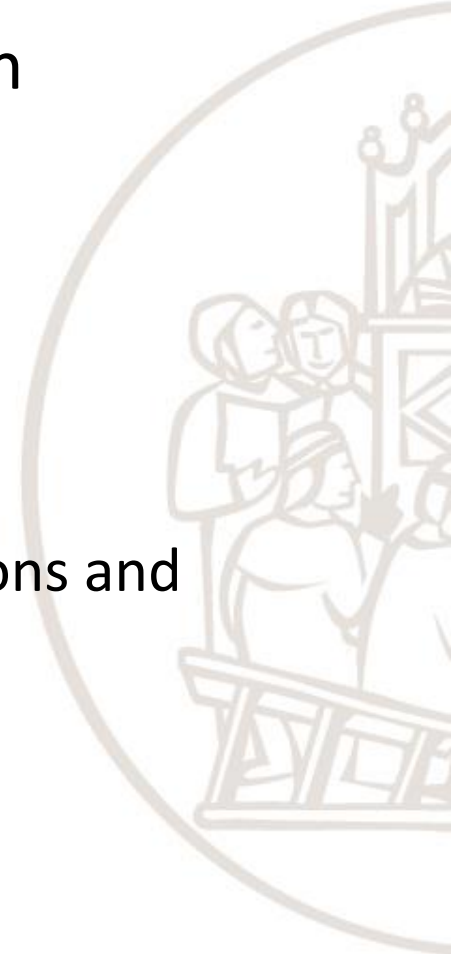
# Reactive systems

- *In **reactive systems*** robots interact with the world through sensors and actuators.
- Knowledge is not modelled nor stored, but it is extracted 'on-line' from the world itself, through sensors.
- The robot behaviours are defined as reactions to the information perceived from the environment.
- Only the second and third questions are relevant and the answers are given in terms of actions.



# Outline of the lecture

- Fundamental problems of robot navigation
- **Maps and environment models**
  - **Metric maps and topological maps**
- Planning techniques
  - Path Planning and Path Following
- Localization methods and systems
  - Odometry and systems based on active beacons and landmarks



# Maps and world representation

- **Metric Maps (Geometric):**

they represent the objects in the world on the basis of their size and coordinates in the absolute reference frame

- **Topological Maps:**

they represent the objects in the world (points of interest) on the basis of their characteristics and of the relations among them





# Geometric maps

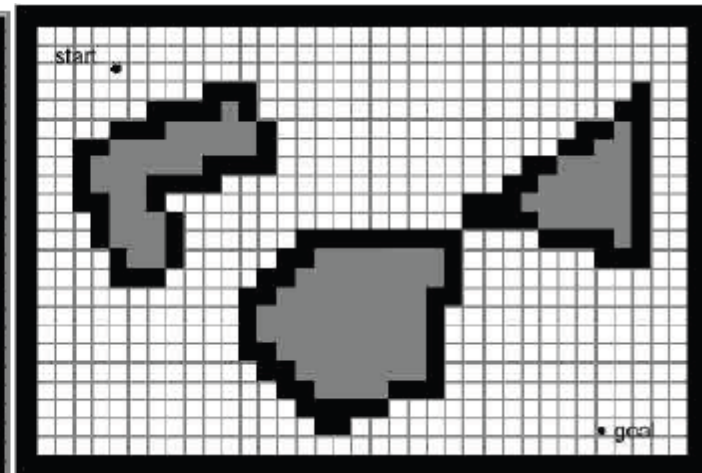
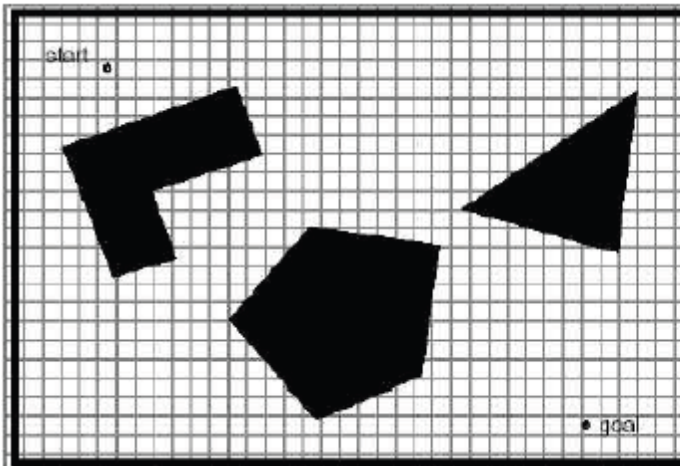
Main methods for world representation through metric maps:

- Occupancy grid
- Geometric description



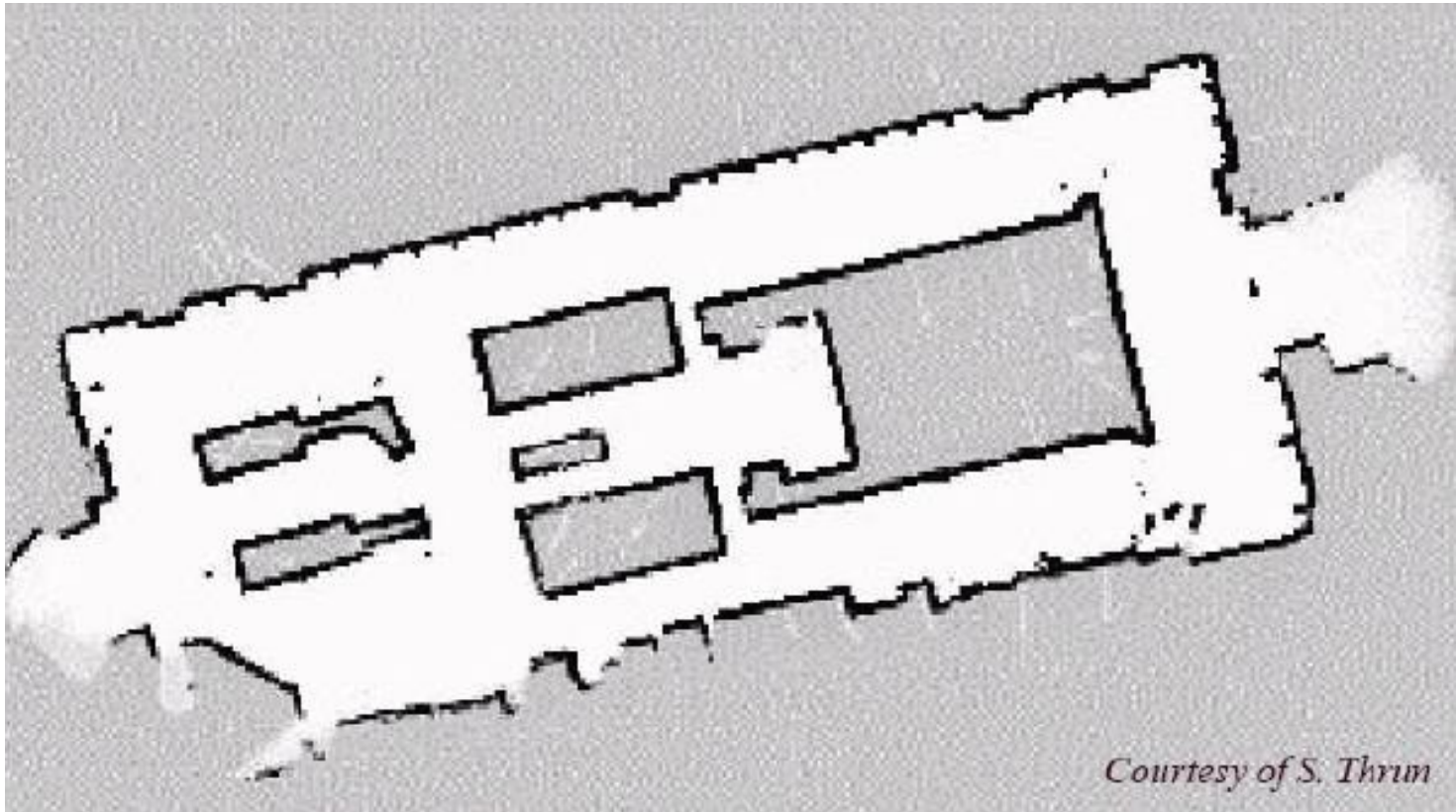
# Occupancy Grid

- The environment is represented on a bi-dimensional grid.
- A value is associated to each grid element, which indicates the cell state (free/occupied)



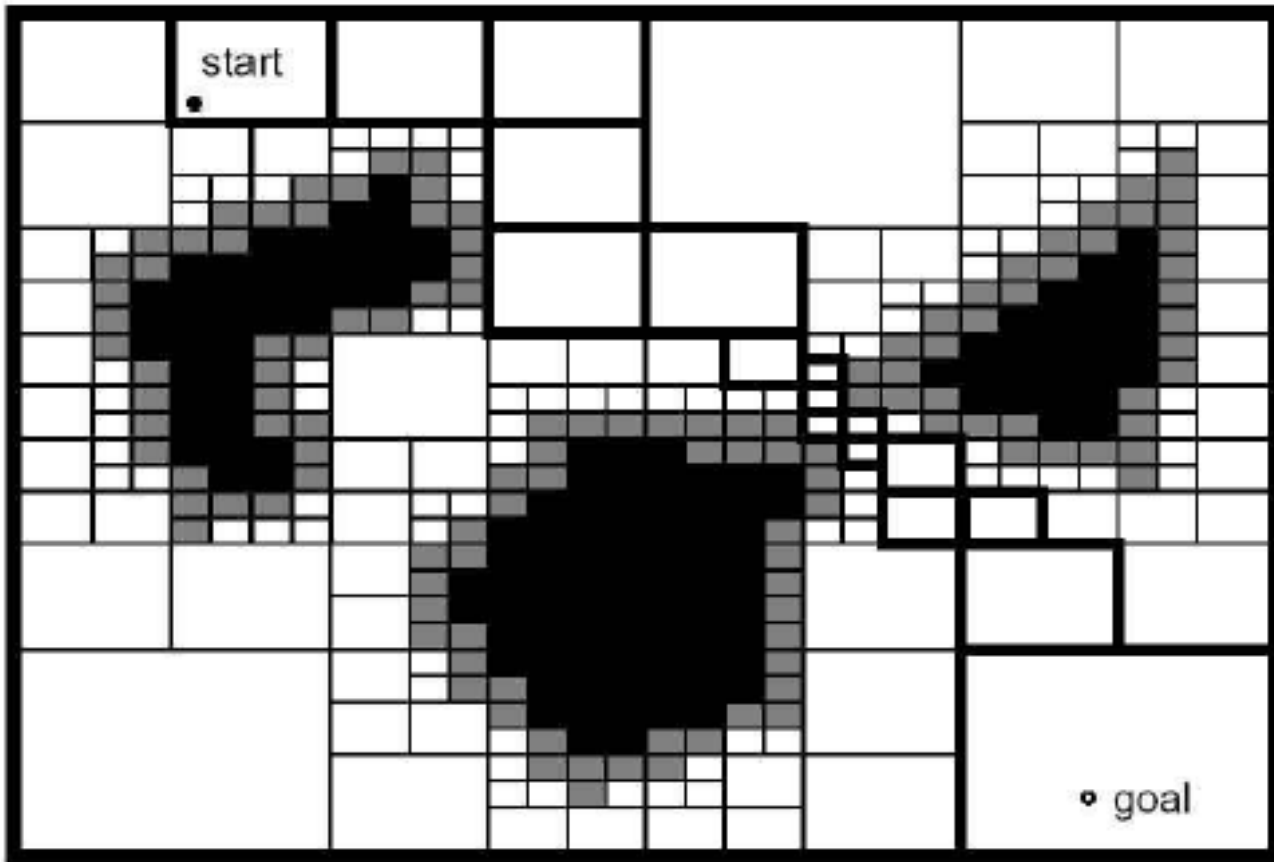
# Occupancy Grid

- Fixed grid



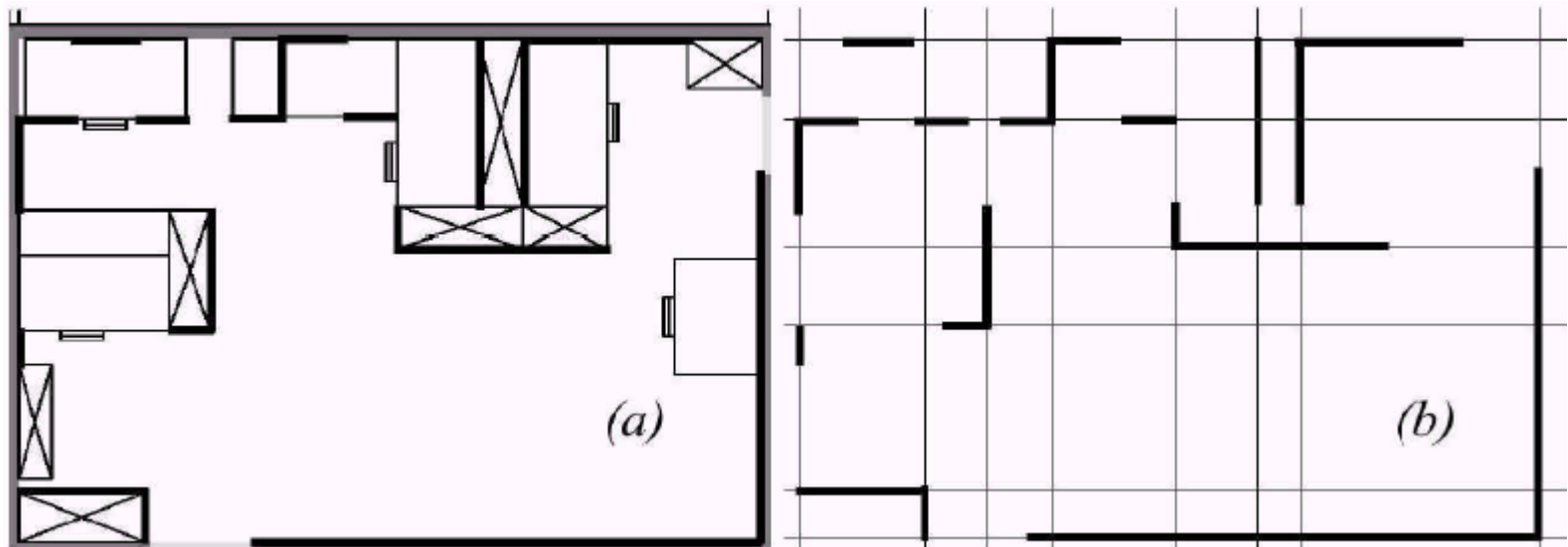
# Occupancy Grid

- Variable grid



# Geometric description

- The environment is represented through a geometric description, usually in terms of segments, obstacles and free space.



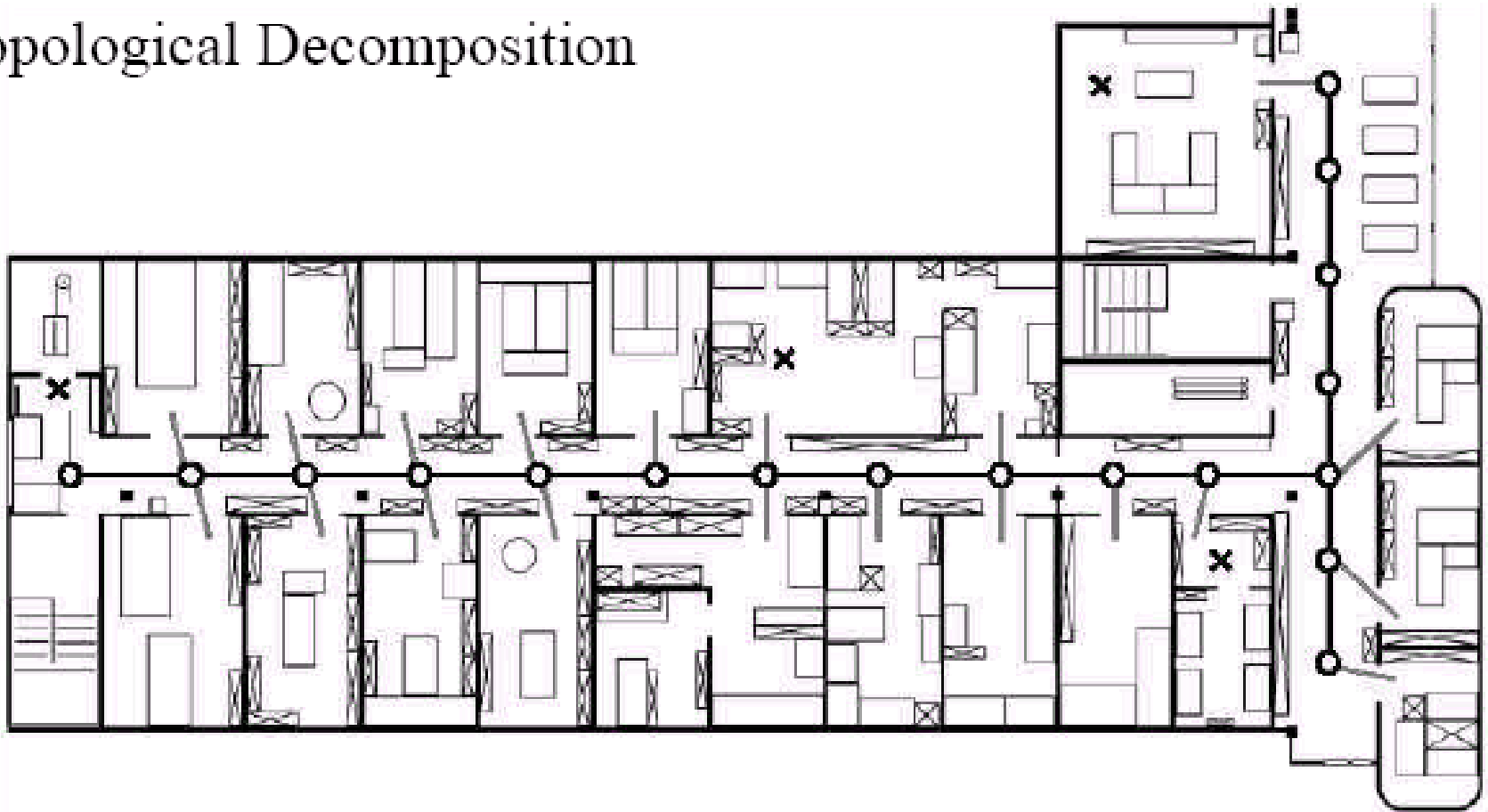
# Topological maps

- The environment is defined in terms of points of interest, relevant for the robot, and of the relations among points of interest.
- A point of interest is an object (natural or artificial) which is relevant for robot navigation (e.g. walls, doors) or for robot tasks (e.g. tables, beds, appliances).
- A point of interest can be defined by a position in the robot space or by a sensory state.



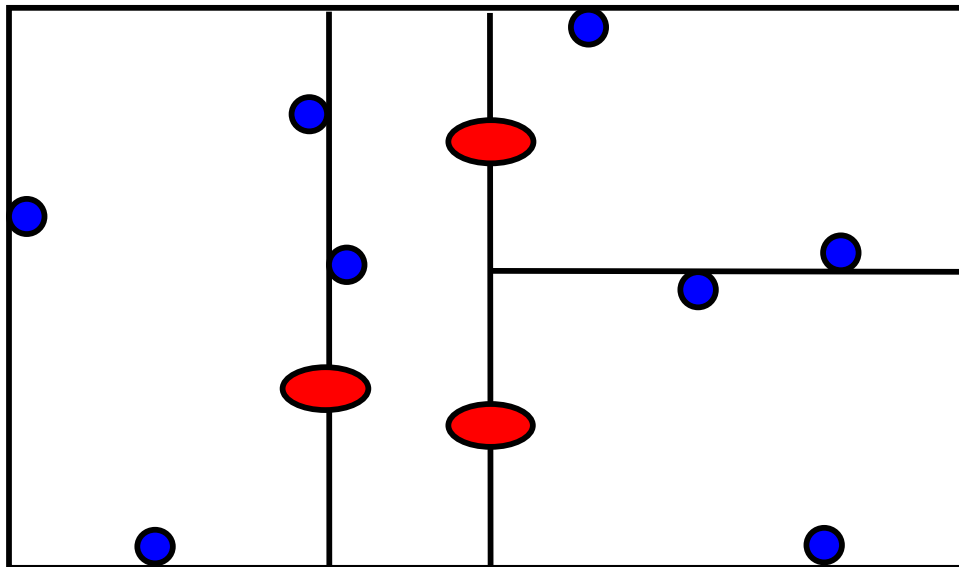
# Topological maps

## Topological Decomposition



# Topological maps

Example: map of a home environment with some points of interest



Door



Points of interest:

desk

bed

fridge

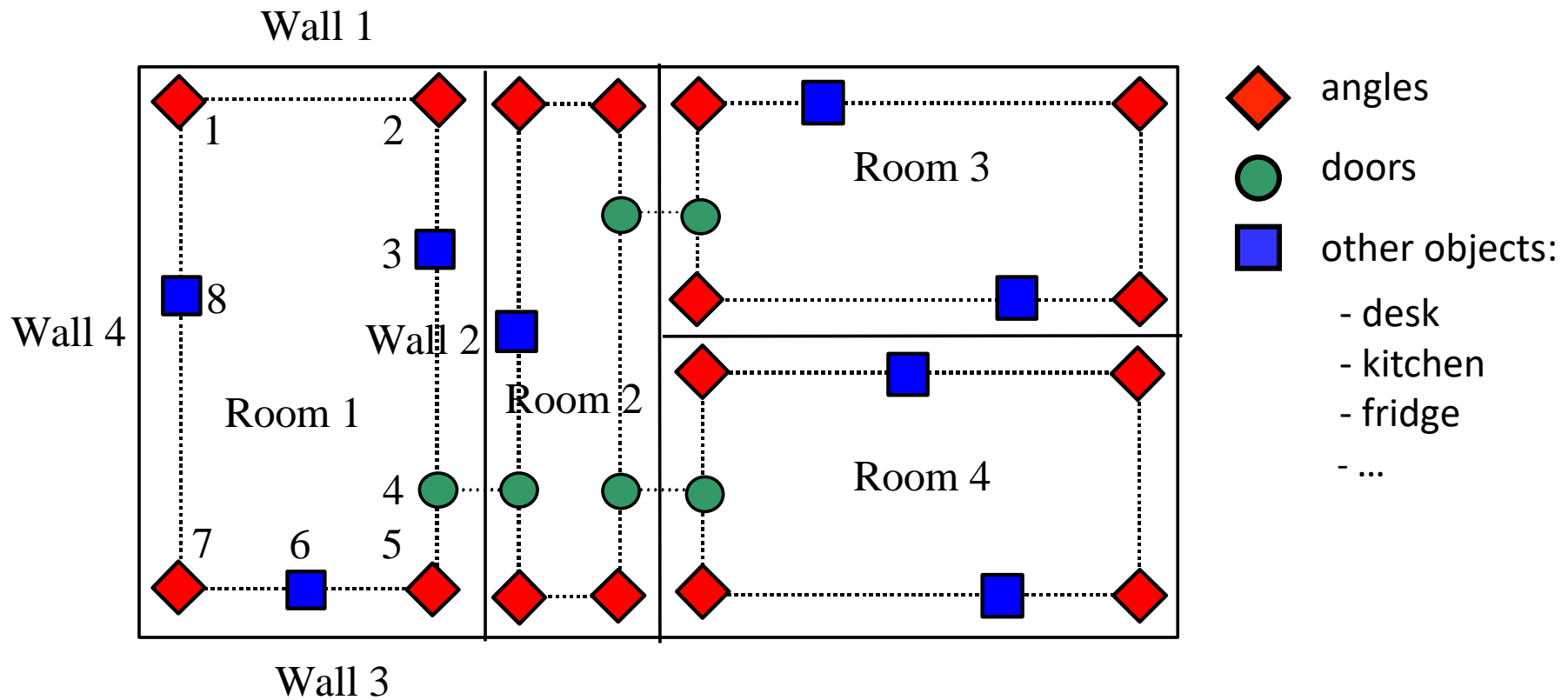
...





# Representation of a topological map through a graph

- Assign a number to each room
- For each room, number the walls clockwise
- For each wall, number the points of interest clockwise

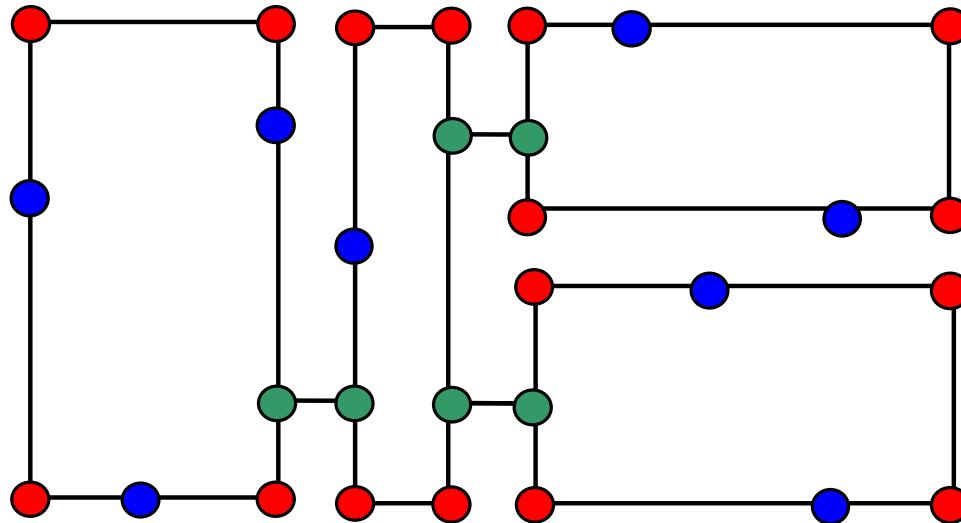


# Representation of a topological map through a graph

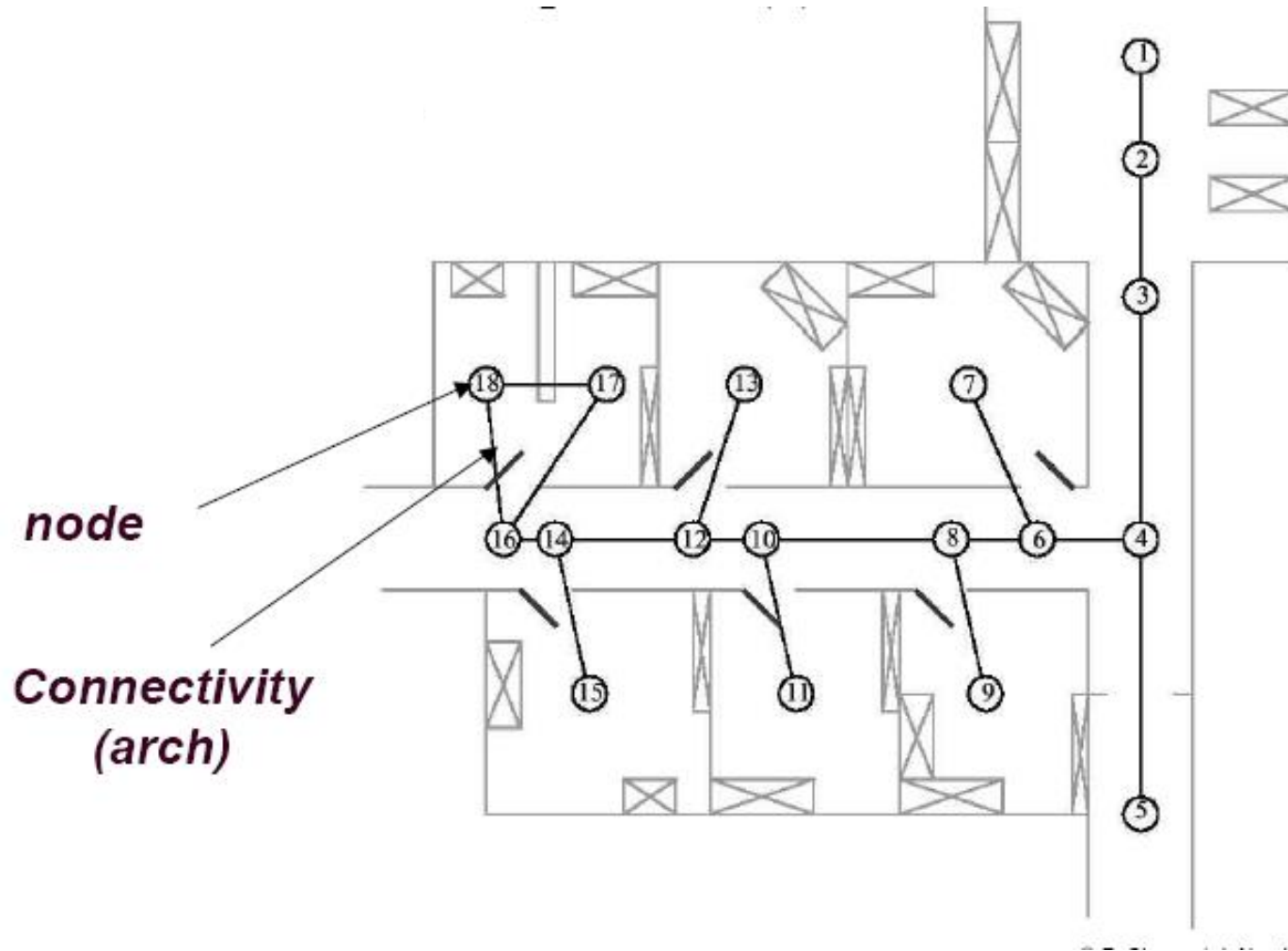
$$G = (N, E)$$

$N = \{\text{points of interest}\}$

$E = \{(p, q) \mid (p \in N, q \in N, p = q \pm 1) \vee (p \text{ and } q \text{ represent the same door for two different rooms})\}$



# Representation of a topological map through a graph





# Outline of the lecture

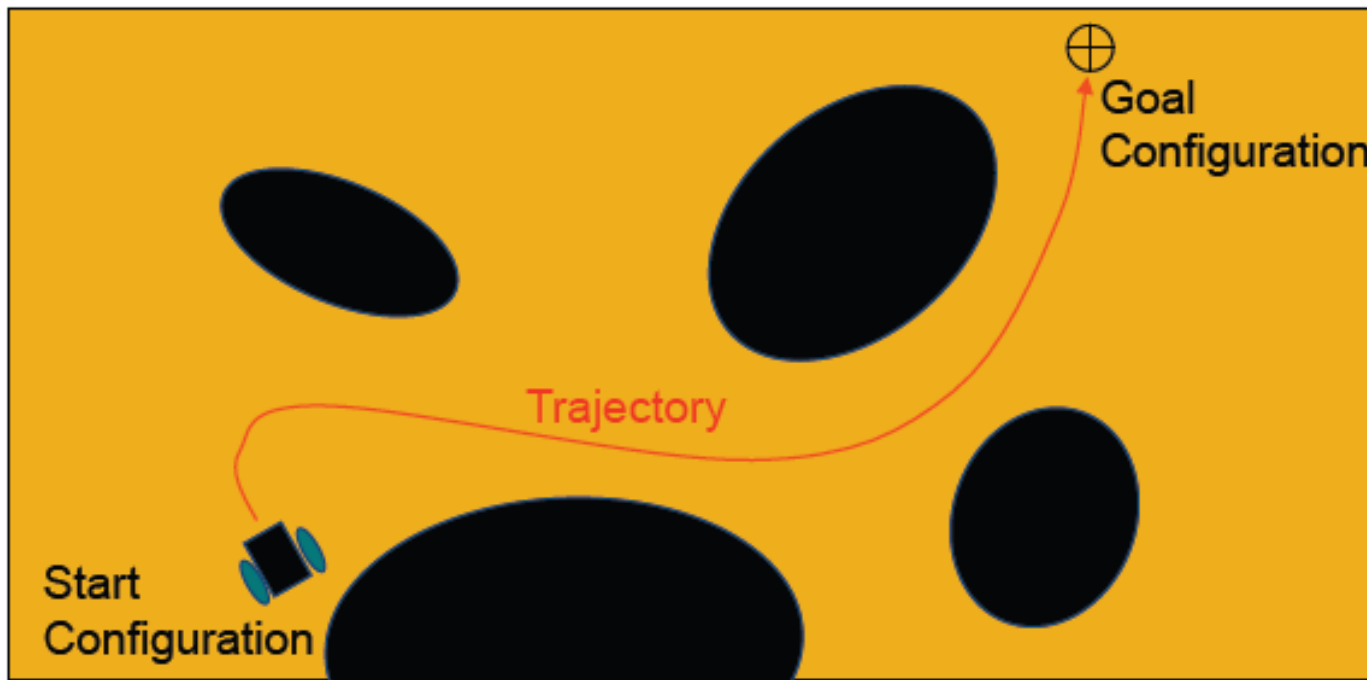
Scuola Superiore  
Sant'Anna

- Fundamental problems of robot navigation
- Maps and environment models
  - Metric maps and topological maps
- **Planning techniques**
  - **Path Planning and Path Following**
- Localization methods and systems
  - Odometry and systems based on active beacons and landmarks



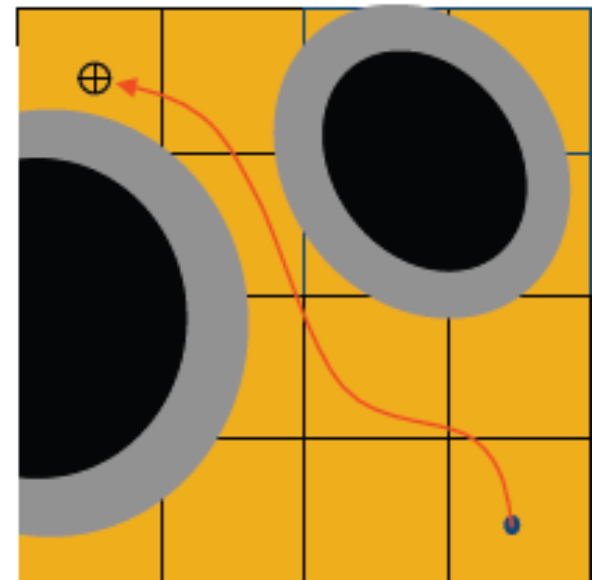
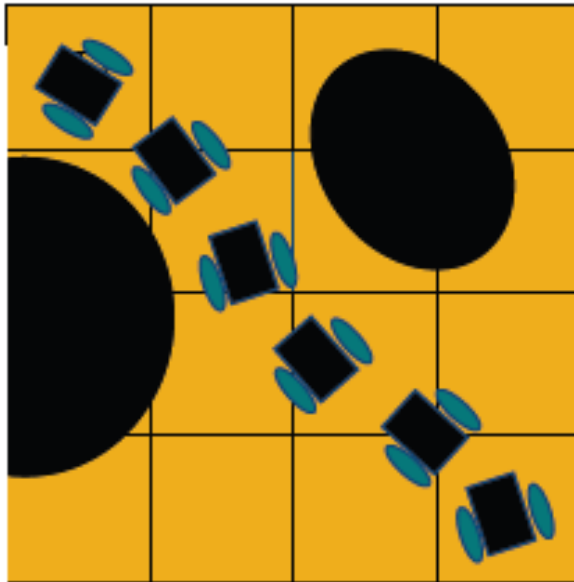
# Planning and world models

The objective of planning is to find a trajectory for the robot to reach a goal configuration, from a start configuration (its current position), avoiding obstacles



# Planning and world models

- The robot size can be used to increase the obstacle size and then consider the robot as a point



# Planning and world models

Planning is divided in:

- **Path Planning:** techniques for finding trajectories for the robot to reach the goal configuration, avoiding obstacles
- **Path Following:** techniques for executing the trajectories generated by the Path Planning, avoiding unexpected obstacles.



# Configuration Space

- Space is named Configuration Space or  $C_{\text{space}}$  (configurations that the robot can reach).
- The robot is represented in  $C_{\text{space}}$  as a point.
- Obstacles are represented in  $C_{\text{space}}$ .
- The region of obstacles is named  $C_{\text{obstacle}}$ .
- The region of free space is named  $C_{\text{free}}$ .
- A path is a trajectory between two configurations  $q_{\text{init}}$  and  $q_{\text{goal}}$  of  $C_{\text{space}}$  belonging to  $C_{\text{free}}$ .





# Path Planning and world models for geometric maps

Main Path Planning techniques based on geometric maps:

- **Roadmaps**
- **Cell Decomposition**
- **Pontential Fields**



# Roadmaps

- The Roadmap approach consists of connecting some points of the free space in a network, named Roadmap, made of unidimensional curves in the free space.
- The Path Planning problem becomes connecting the start and the goal configurations by finding a path in the roadmap.



# Roadmap

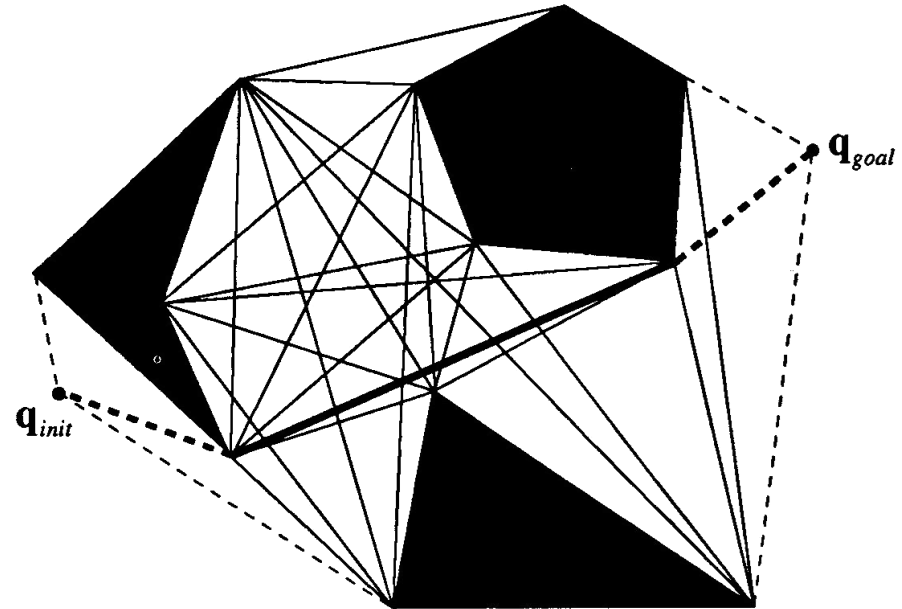
Main Path Planning techniques based on the Roadmap approach:

- **Visibility Graph**
- **Voronoi Diagram**



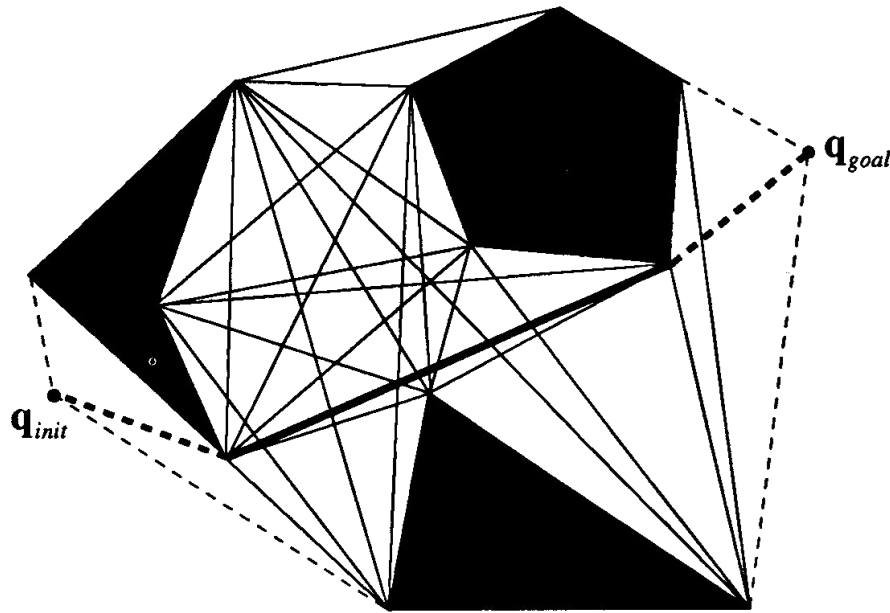
# Visibility Graph

- The visibility graph is a graph  $G$  whose nodes are the initial and goal configurations,  $q_{init}$  and  $q_{goal}$ , and all vertexes of the polygons which represent the obstacles in the map
- The edges of  $G$  are all the segments that connect two nodes in  $G$  and that do not intersect the obstacle polygons
- A weight can be associated to the edges, corresponding to the distance between the nodes that they connect
- A path from  $q_{init}$  and  $q_{goal}$  can be found on the graph  $G$  by using an algorithm for minimum paths which minimizes the distance



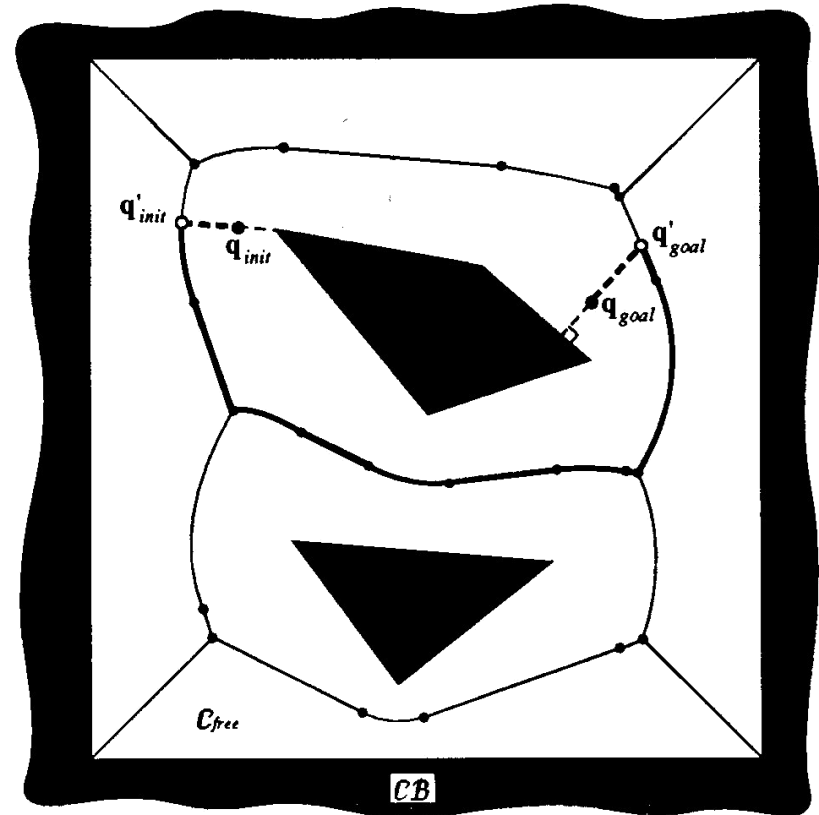
# Visibility Graph

Example: visibility graph and the path found to navigate from  $q_{init}$  to  $q_{goal}$



# Voronoi Diagram

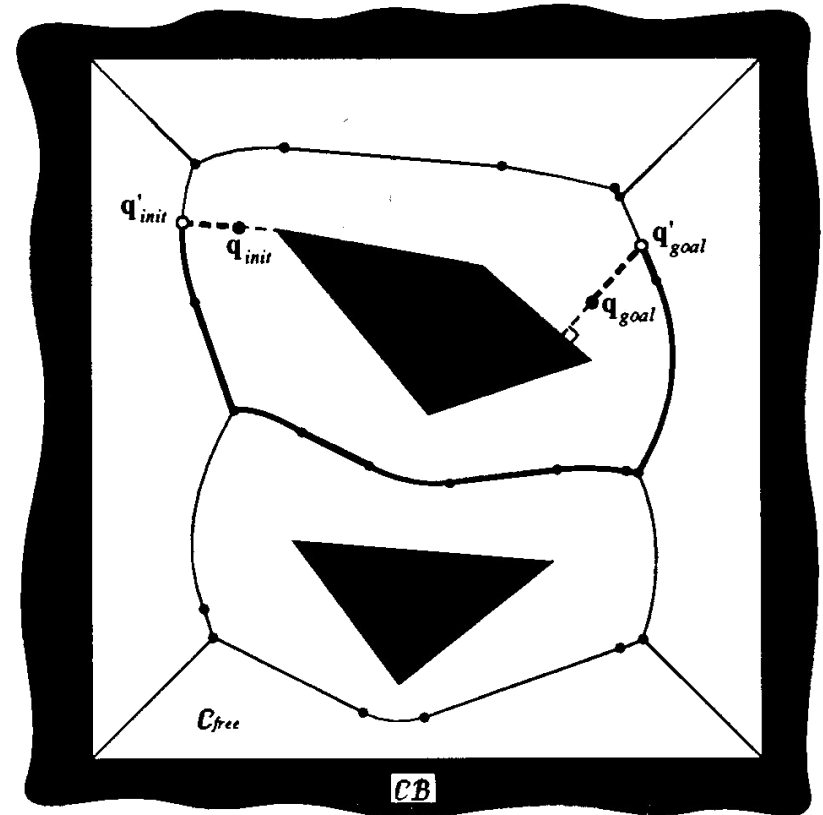
- It consists of defining all the free configurations in the free space, equidistant from the obstacle region
- If the obstacles are polygons, the Voronoi diagram consists of a finite set of segments and parabolic curves (roadmap).



# Voronoi Diagram

Given two configurations  $q_{init}$  and  $q_{goal}$ , a path is given by:

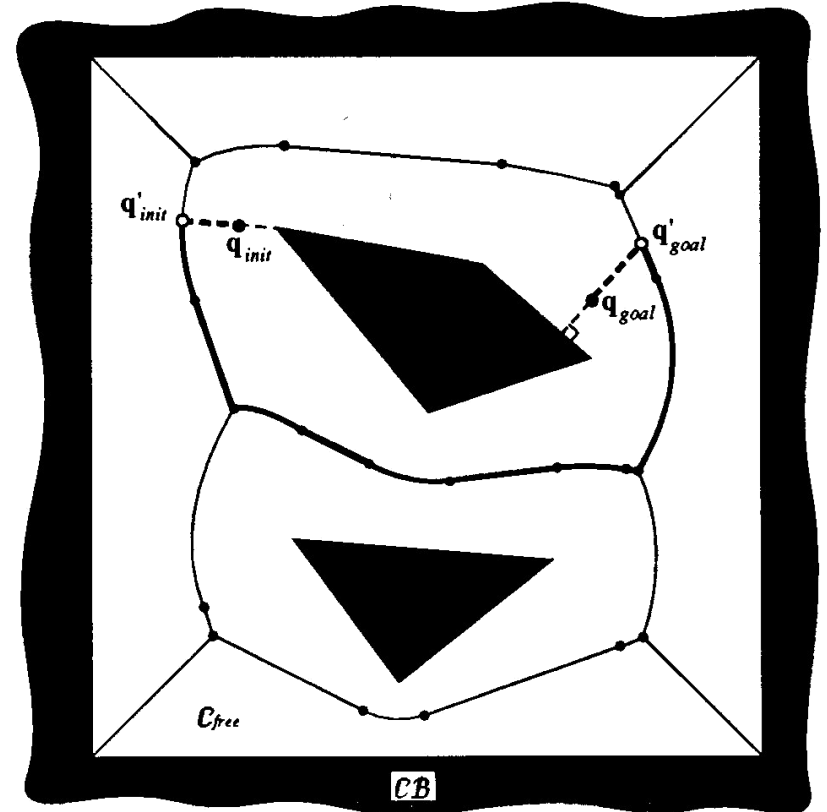
- Connecting  $q_{init}$  and  $q_{goal}$  to the roadmap in  $q'_{init}$  and  $q'_{goal}$ .
- Finding a path on the Voronoi diagram which connects  $q'_{init}$  and  $q'_{goal}$ .



# Voronoi Diagram

Example: Voronoi diagram and the path found to navigate from  $q_{init}$  to  $q_{goal}$

The advantage of this technique is that the trajectories tend to maximise the distance of the robot from obstacles.





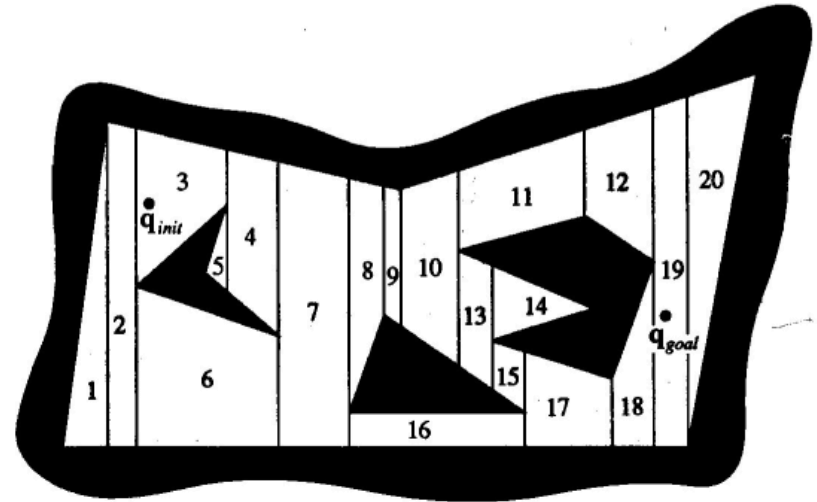
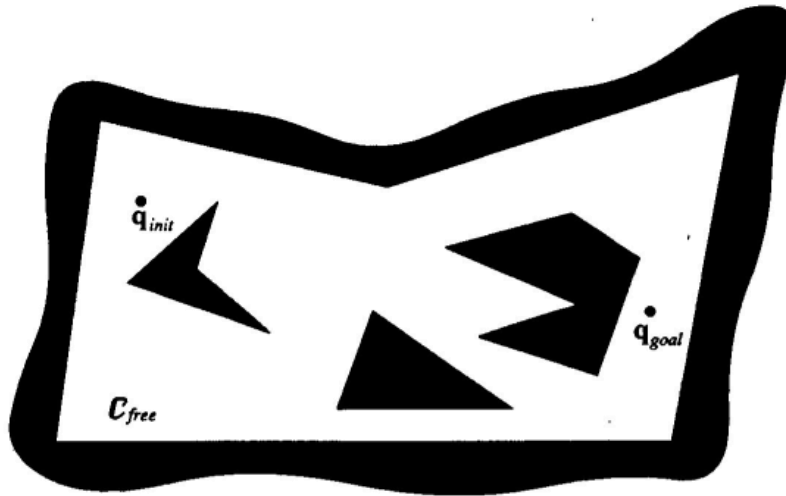
# Cell Decomposition

- It consists of decomposing the free space in regions, named cells, such that a path between two adjacent cells can be easily found.
- The map is represented by a graph named **connectivity graph**.
- The graph nodes are the cells in the free space.
- Two nodes in the graph are connected if and only if the two cells that they represent are adjacent.

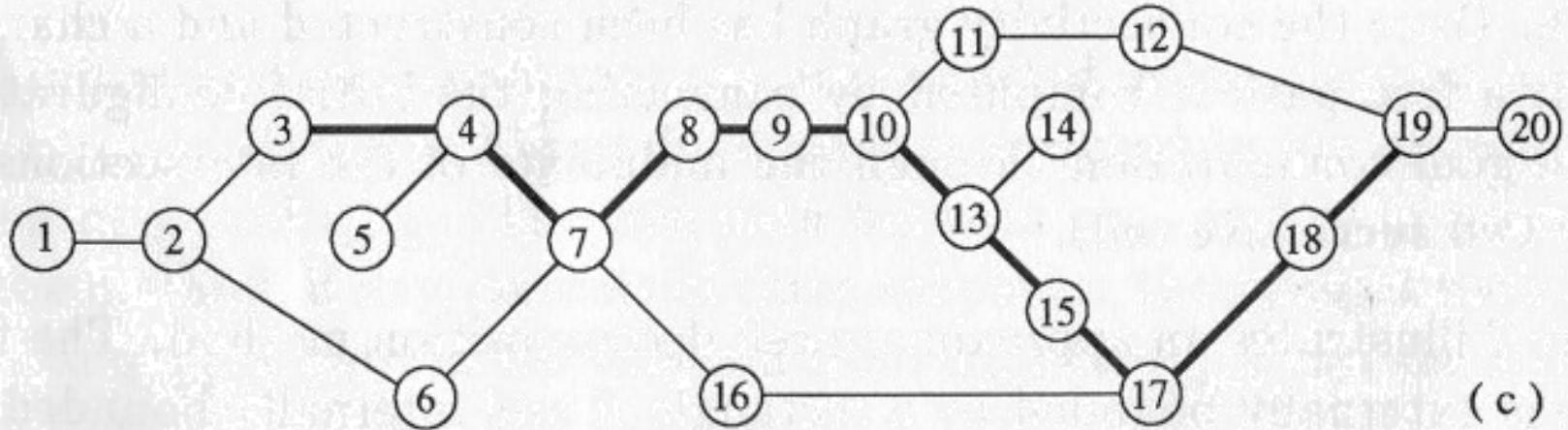


# Cell Decomposition

Example of map and its cell decomposition



Connectivity graph associated to the map and the path found (in bold)



# Cell Decomposition

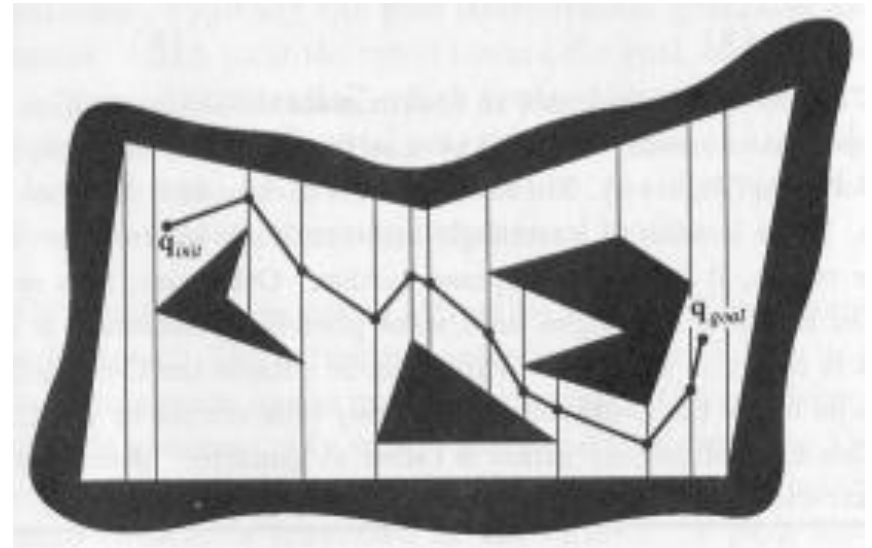
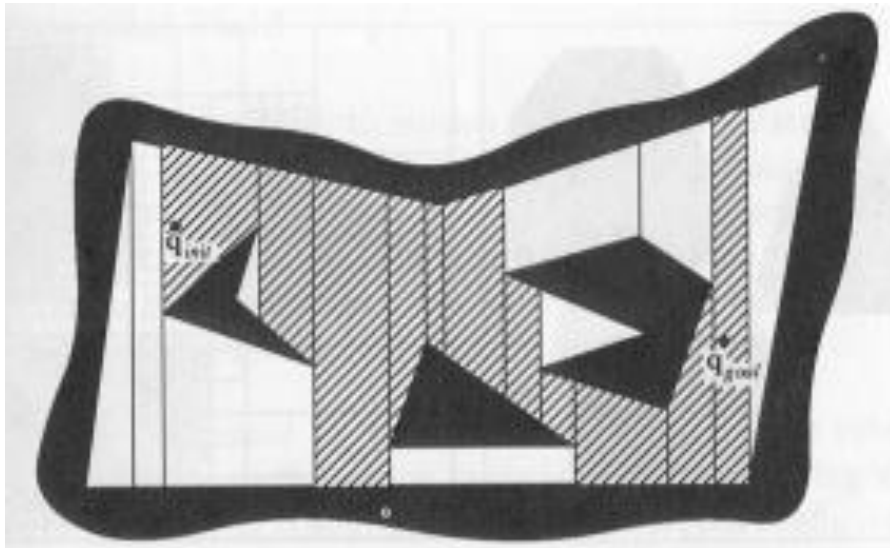
- A trajectory for the robot is found by searching a path on the graph that connects the nodes containing  $q_{init}$  and  $q_{goal}$ .
- The result of the graph search is a sequence of cells named **canal**.
- The path is found by connecting the mid points of the canal cell boundaries.



# Cell Decomposition

The results of the graph search are:

- The canal (grey cells)
- The path found (line in bold)



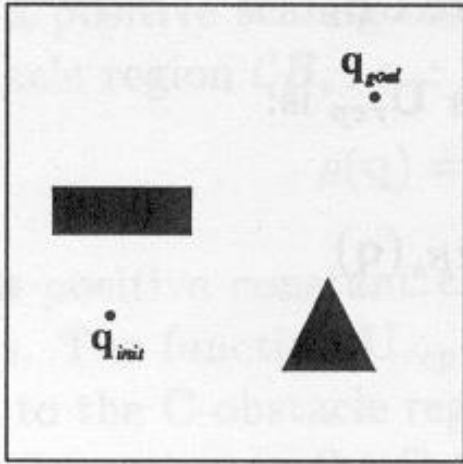
# Potential fields

- The robot is a point in space that moves under the influence of an **artificial potential** produced by the goal configuration and by the obstacles
- The final configuration generates an **attractive potential** which pulls the robot towards the goal
- The obstacles generate a **repulsive potential** which pushes the robot away from them
- The sum of attractive and repulsive potentials generates a force which moves the robot towards the goal, while avoiding obstacles

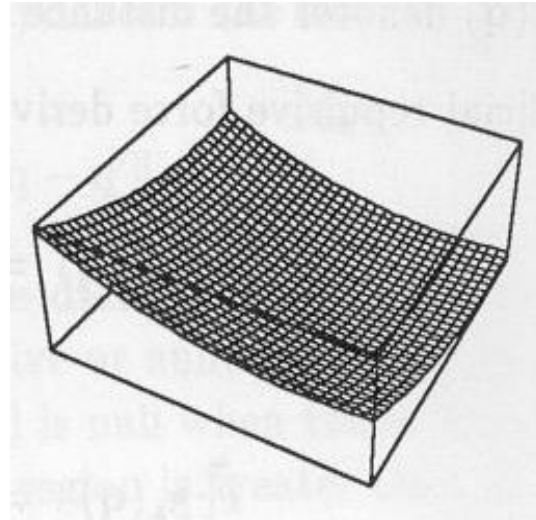


# Potential fields

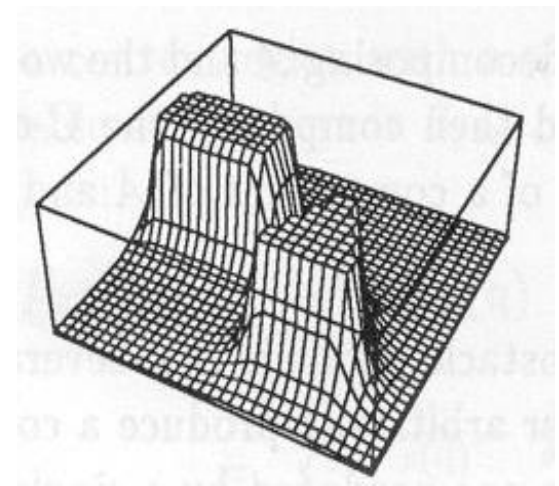
Example of attractive and repulsive potential



World map



Hyperbolic attractive potential function



Repulsive potential function



# Potential fields

- Differentiable potential function  $U$  with a local minimum in the point  $q_{\text{goal}}$

$$U(q) = U(q)_{\text{att}} + U(q)_{\text{rep}}$$

$U(q)_{\text{att}}$  Attractive potential function

$U(q)_{\text{rep}}$  Repulsive potential function

- For each point  $q$  in space, the motion direction is given by the force function  $F$

$$F(q) = -\nabla U(q) = -(F_{\text{att}}(q) + F_{\text{rep}}(q))$$

$$\nabla U(q) = (\delta U / \delta x, \delta U / \delta y)$$



# Potential fields

## Criteria for choosing the **attractive potential**

- Function with a local minimum in the point  $q_{\text{goal}}$

$$U_{\text{att}}(q) = \frac{1}{2} \xi \rho_{\text{goal}}^2(q) \quad \text{parabolic potential}$$

where

$$\rho_{\text{goal}}^2(q) = \|q - q_{\text{goal}}\|^2 \quad \text{Euclidean distance}$$

so that

$$F_{\text{att}}(q) = -\xi (q - q_{\text{goal}})$$





# Potential fields

## Criteria for choosing the **repulsive potential**

- Creating a protective barrier around the obstacle region, to avoid robot contact with the obstacles
- The repulsive force should not affect the robot motion when it is far from obstacles

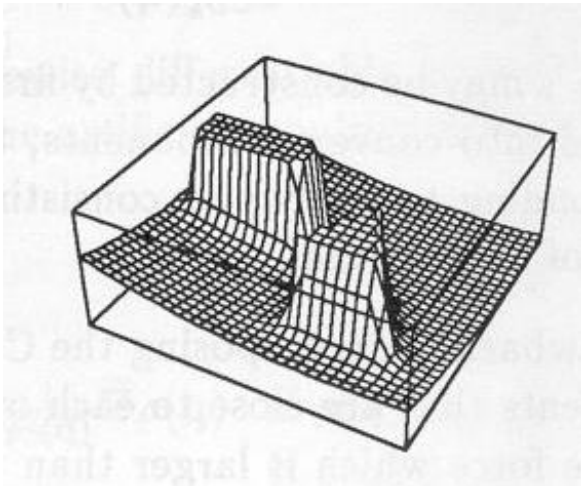
$$U_{\text{rep}}(q) = \begin{cases} \frac{1}{2}\eta \left( \frac{1}{\rho(q)} - \frac{1}{\rho(q_0)} \right)^2 & \text{se } \rho(q) \leq \rho_0 \\ 0 & \text{se } \rho(q) > \rho_0 \end{cases}$$

where

$$\rho(q) = \min ||q - q' || \quad q' \in C_{\text{obstacle}}$$

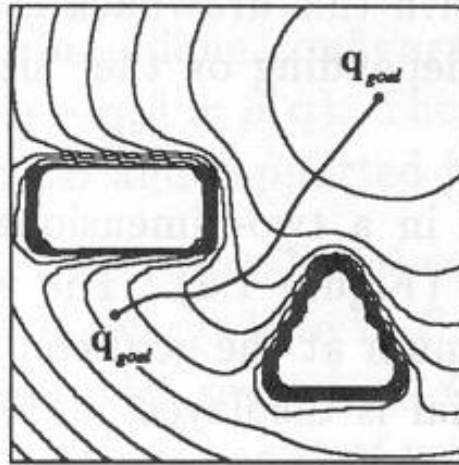


# Potential fields

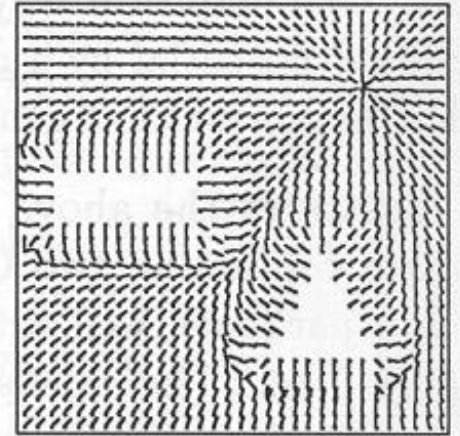


Total potential function

$$U = U_{\text{att}} + U_{\text{rep}}$$



Curves of the total potential function and the trajectory found



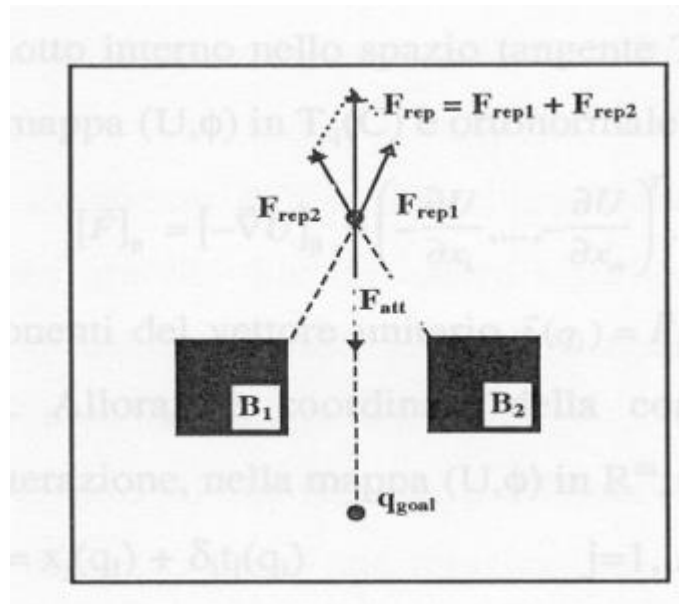
Matrix of the gradient vector orientations



# Potential fields

Problem of local minima of the resulting function:

they can occur when the sum of repulsive forces nullifies the attractive force in points different from  $q_{\text{goal}}$



# Path Planning for topological maps

Example of path on a topological map:

- Follow the wall on the right;
- Turn right;
- Follow the wall on the right;
- Stop in front of the doors;
- Enter the door and turn left
- Follow the wall on the left;
- Stop when reached the desk;





# Path Planning for topological maps

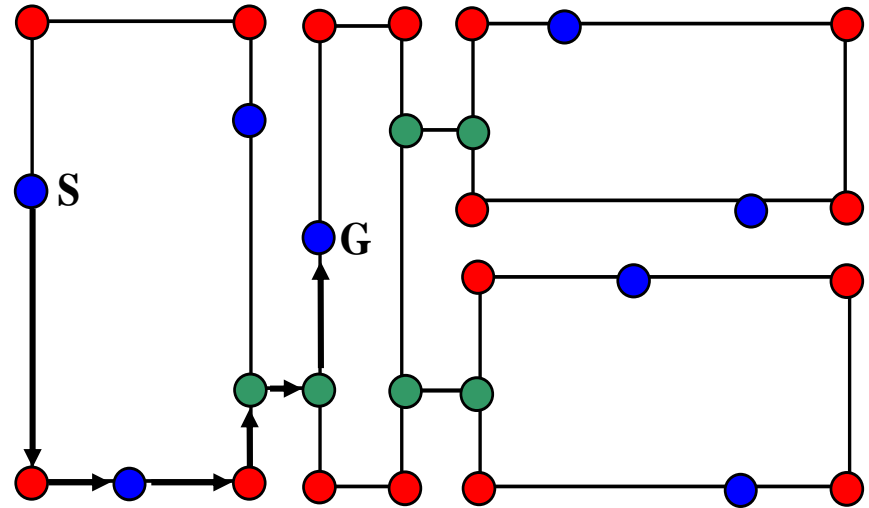
- **Rules for translating a path in a sequence of commands:**
  - For the **Start** node, the command generated is **follow the wall on the right** or **follow the wall on the left**, depending on the order of the adjacent node to reach.
  - For the intermediate nodes of type **Angle**, the command generated is **change wall on the right** or **change wall on the left**.
  - For the intermediate nodes of type **Door**, the command generated is **go straight** if the robot does not have to enter the adjacent room or **enter the door and turn left (right)** if the robot has to enter the adjacent room and it has to follow the wall on the left (right).
  - For the other intermediate nodes, different from the goal node, the command is **follow the wall**.
  - When the robot reaches the **Goal** node, the command generated is **Stop**.



# Path Planning for topological maps

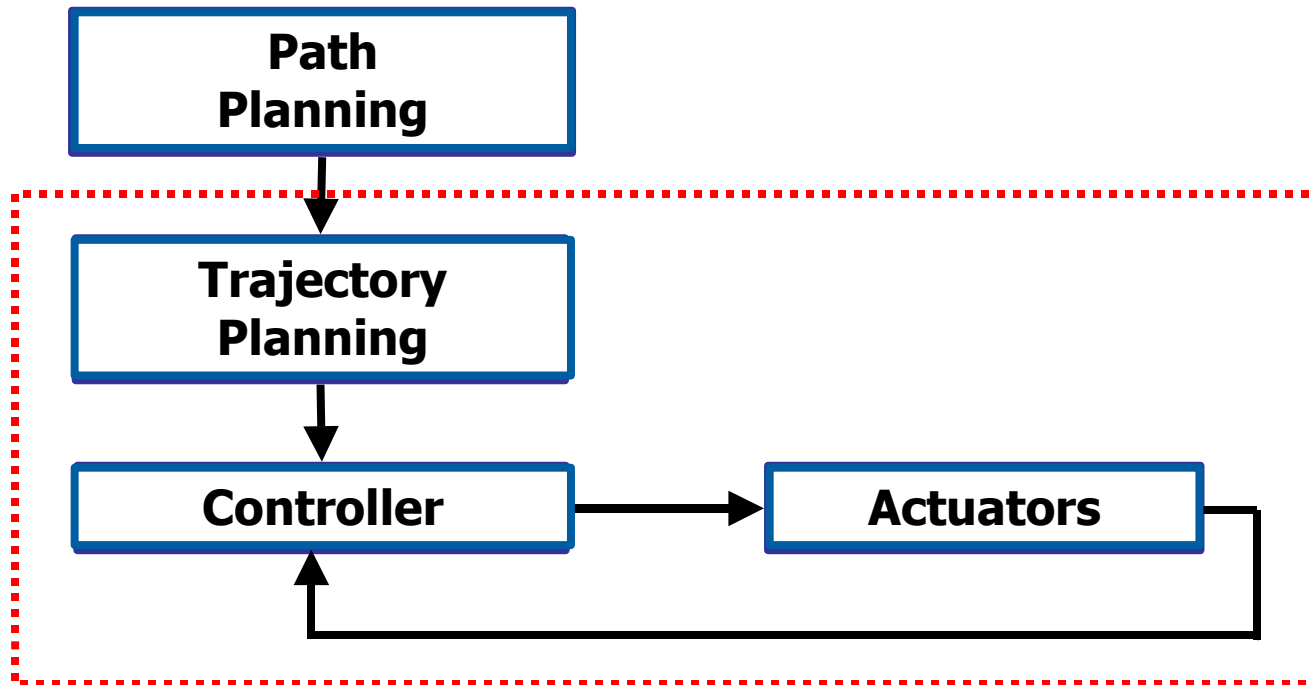
Path generated for reaching the final point G starting from the initial point S:

- 1) follow the wall on the right;
- 2) change wall on the right;
- 3) follow the wall on the right;
- 4) change wall on the right;
- 5) enter the door and turn left;
- 6) follow the wall on the left;
- 7) stop when the goal point of interest is reached.



# Path Following

- It has the role of making the robot follow the paths generated by the Path Planner



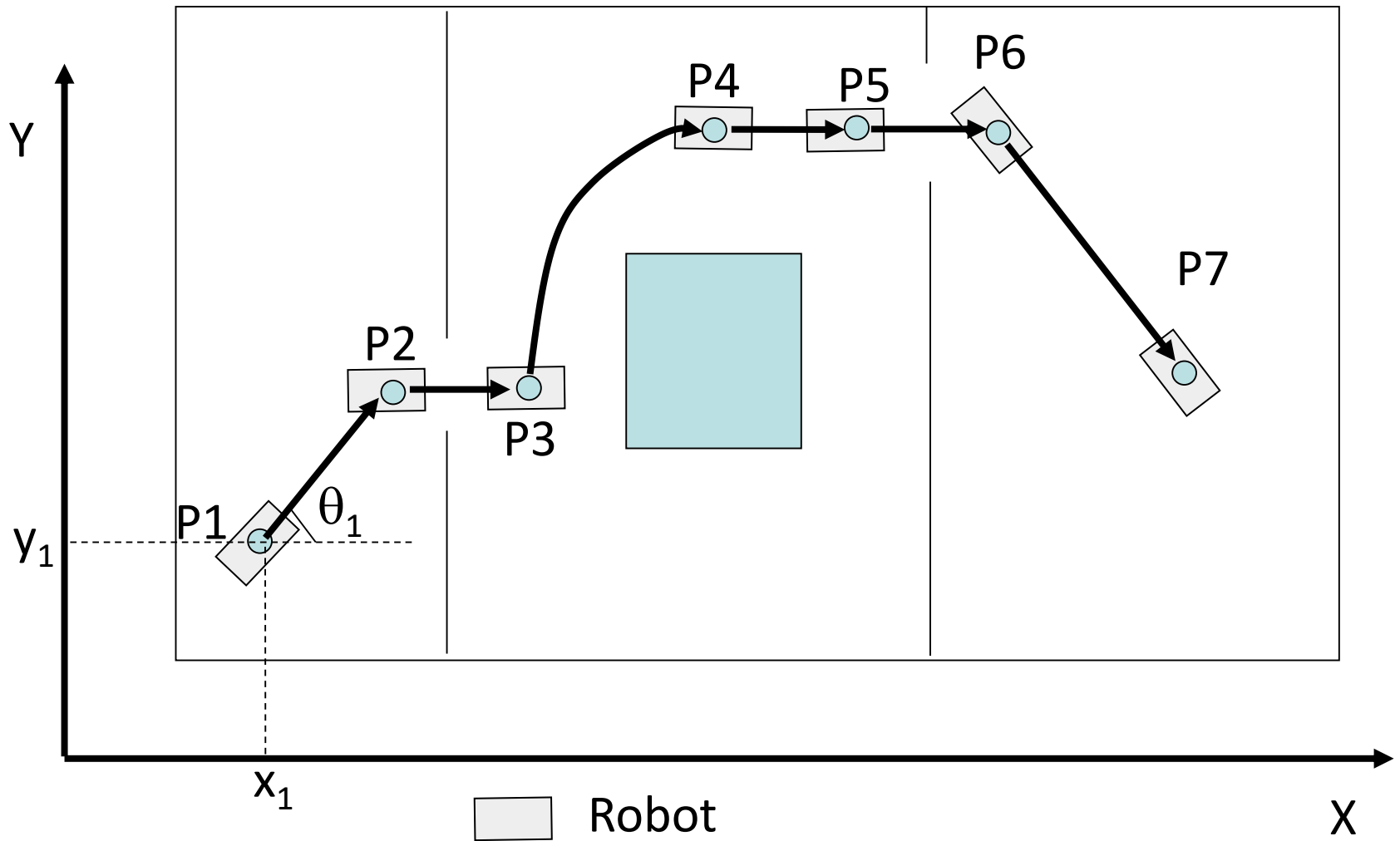


# Path Following

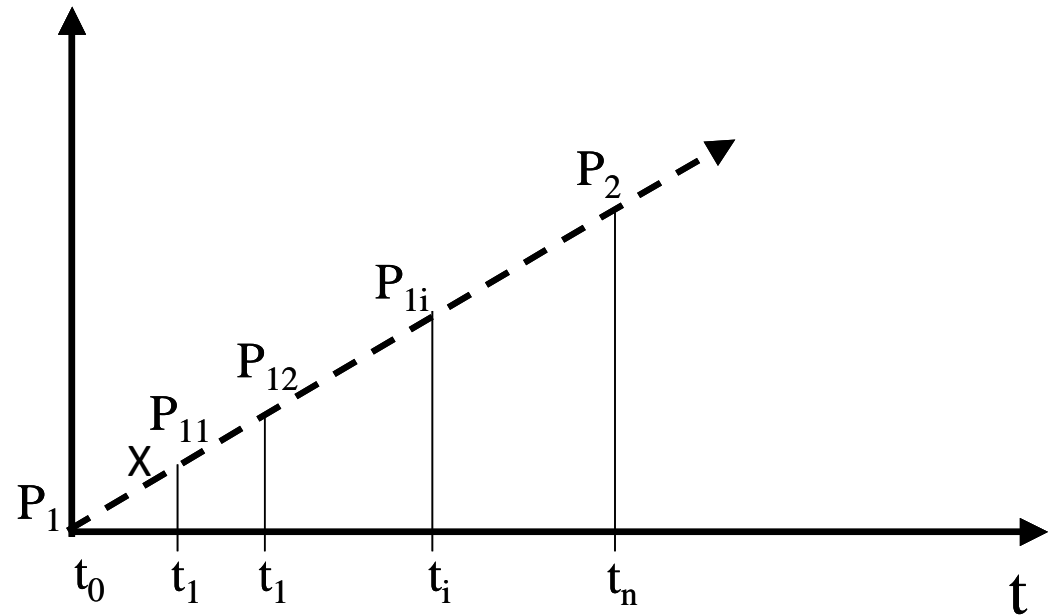
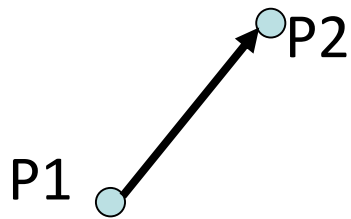
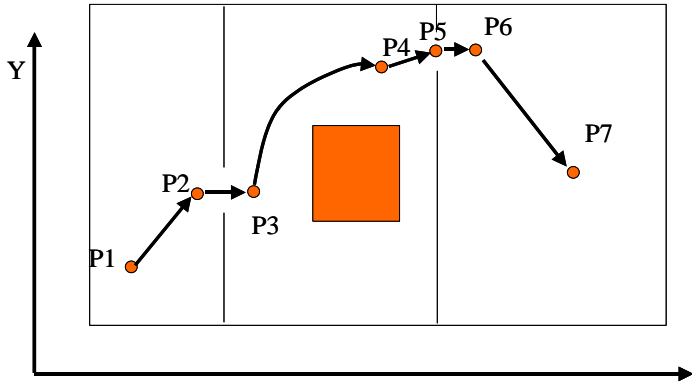
- **Path planning:** it finds a sequence of points in space that the robot has to reach (path)
  - $(x_1, y_1, \theta_1), \dots (x_{i-1}, y_{i-1}, \theta_{i-1}), (x_i, y_i, \theta_i), \dots (x_n, y_n, \theta_n)$
- **Trajectory planning:** it finds the trajectory and the time law that the robot has to follow between each couple of points (not necessarily a linear trajectory)
- **Controller:** it makes the robot execute the trajectory found by the trajectory planner



# Path planning



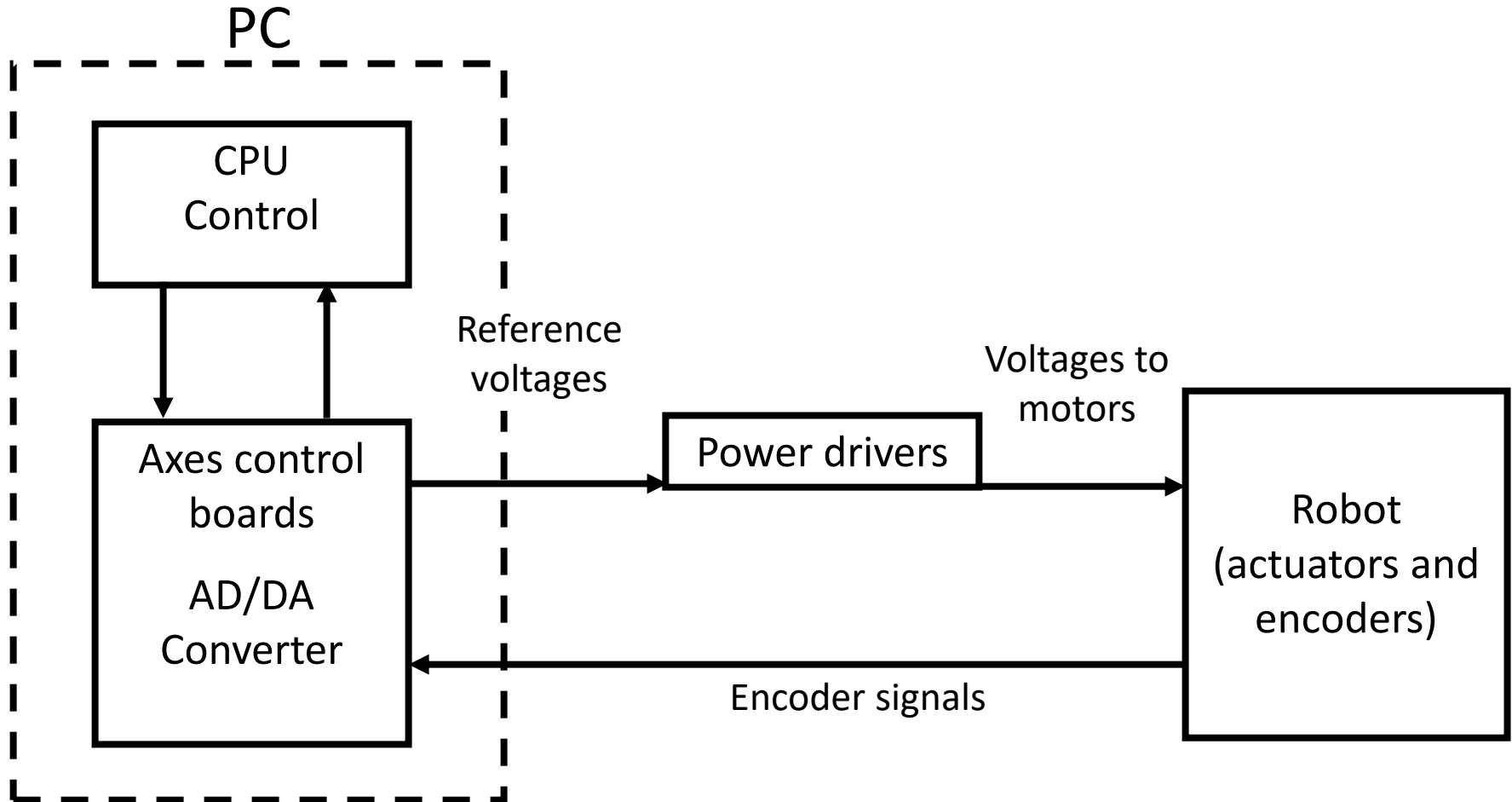
# Trajectory planning



Trajectory generated by the trajectory planner



# Hardware architecture of a mobile robot



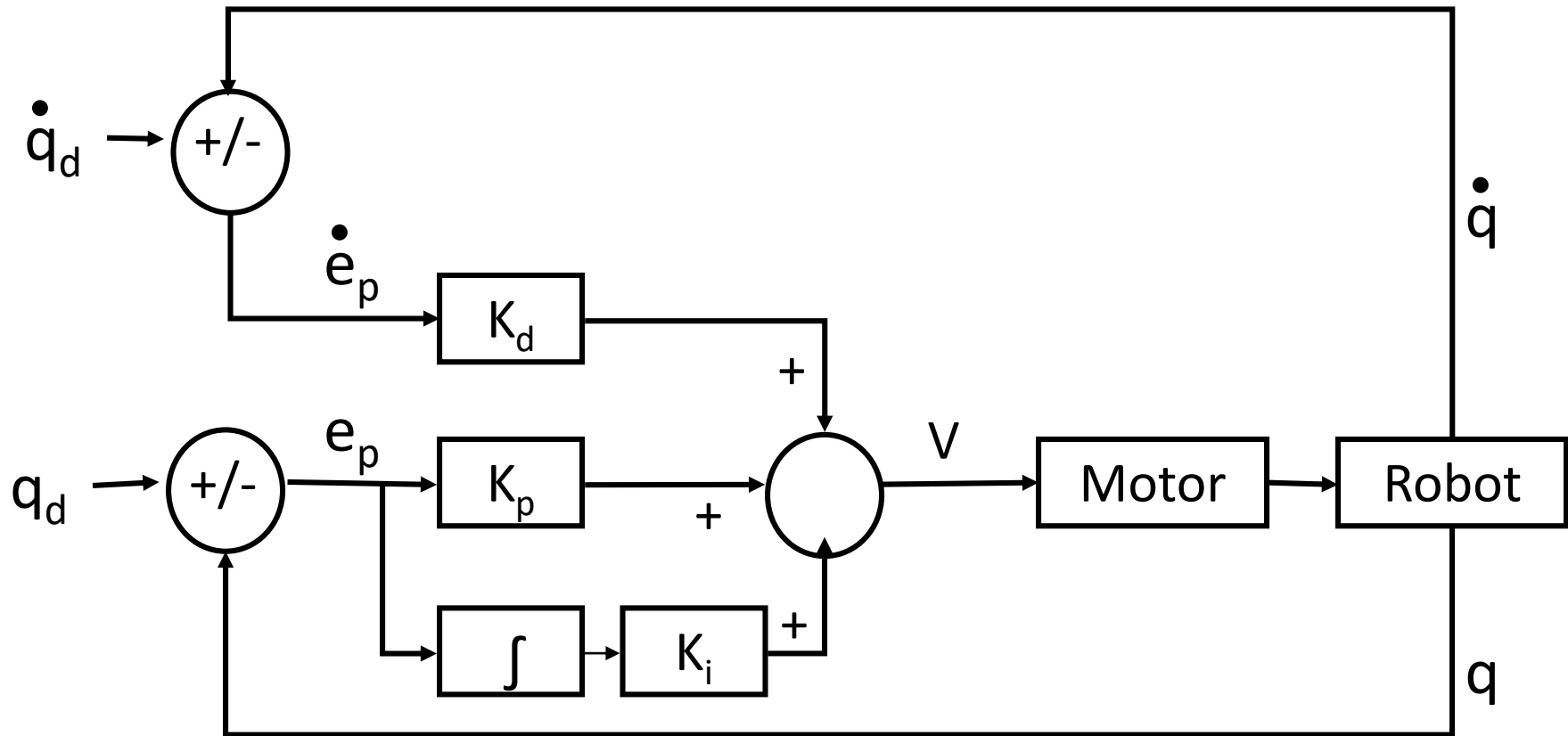
# Controller

## Actuator control:

- **Position control:** it consists of setting a position to reach.  
The robot controller finds the velocities and the accelerations to set to the motors for reaching the desired position (inverse kinematics).
- **Velocity control:** it consists of setting a velocity and an acceleration to the wheel motors.



# Position control: Proportional, Integrative and Derivative control (PID)



$$V = K_p e_p + K_d \dot{e}_p + K_i \int e_p$$



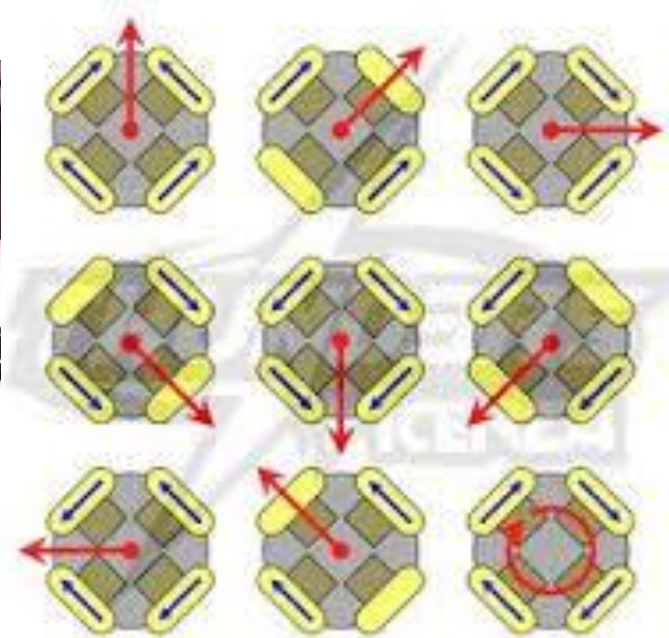
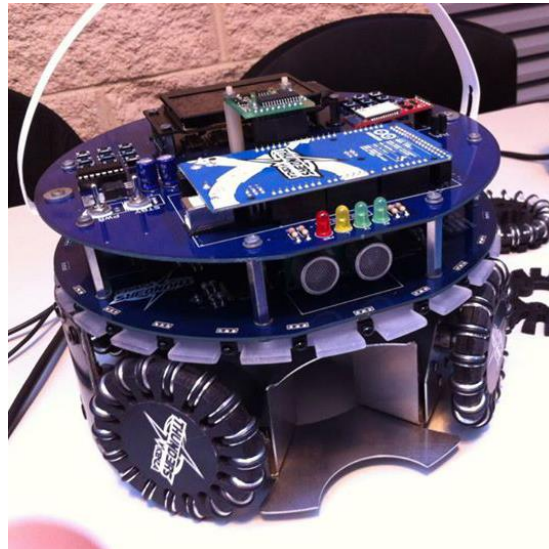
# Path Following

- It is not always possible to follow the path planned by the Path Planner
- Problems to face:
  - Non omnidirectional mobile bases
  - Unexpected obstacles



# Path Following

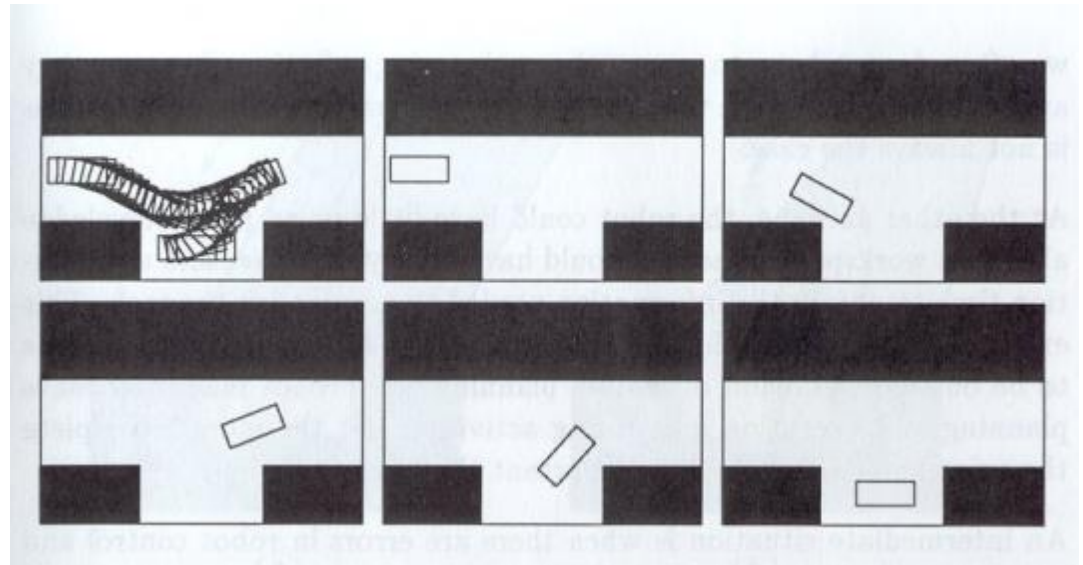
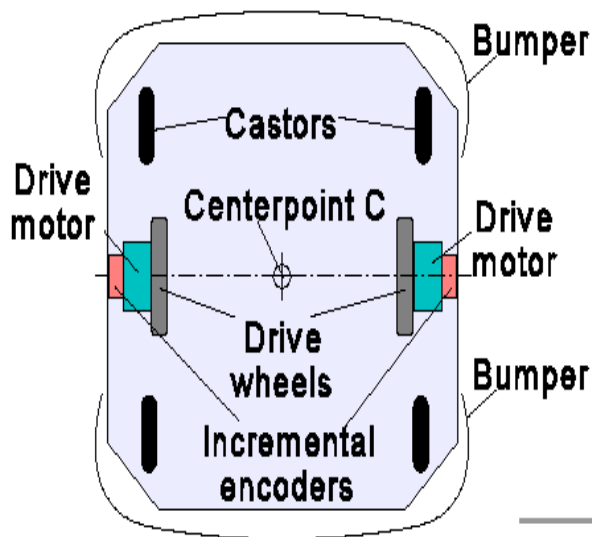
- **Omnidirectional mobile base:**
  - Can move in any direction
  - Can follow the trajectory given by the Path Planner





# Path Following

- **Non omnidirectional mobile base:**
  - Cannot move in any direction, due to its structure (e.g. car-like robot)
  - Can not always follow the trajectory given by the Path Planner



# Path Following – obstacle avoidance

The problem of unexpected obstacles:

- unexpected obstacles are detected by the robot through ultrasound sensors or laser ranger
- the robot controller has to modify the trajectory to follow, in order to avoid obstacles
- obstacle avoidance techniques:
  - Based on occupancy grid
  - Based on potential fields

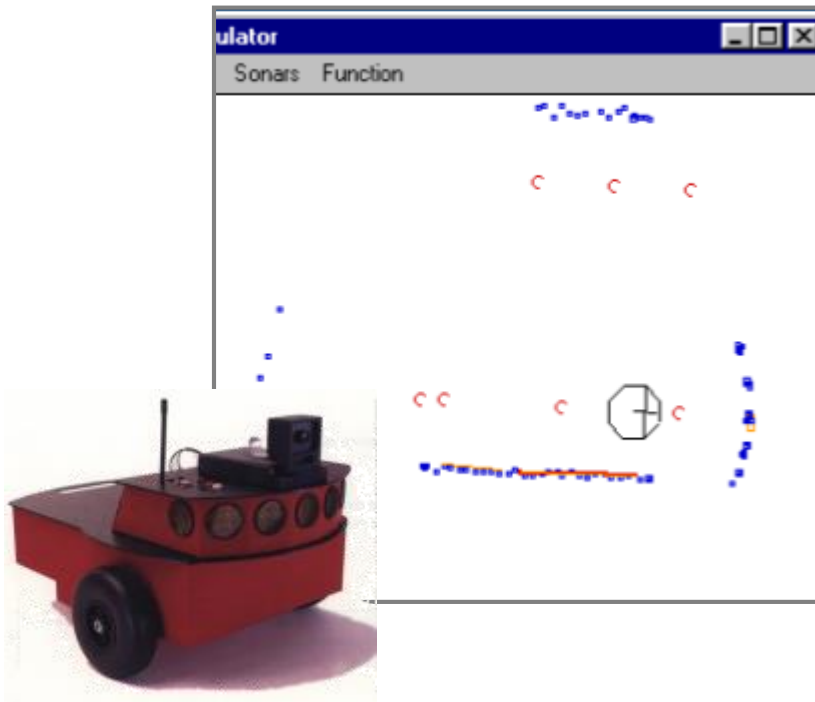


# Sensors for obstacle detection

- Sensory systems:

Ultrasound sensors

Laser Ranger



# Outline of the lecture

- Fundamental problems of robot navigation
- Maps and environment models
  - Metric maps and topological maps
- Planning techniques
  - Path Planning and Path Following
- **Localization methods and systems**
  - **Odometry and systems based on active beacons and landmarks**



# Localization

## Localization methods

- Dead Reckoning - Odometry
- Active beacons
- Natural and Artificial Landmarks



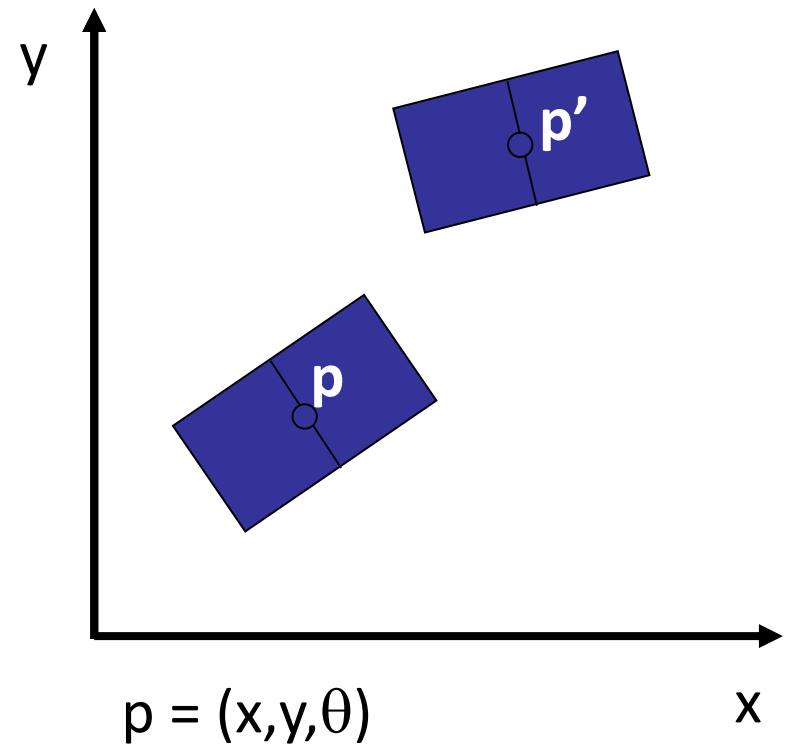
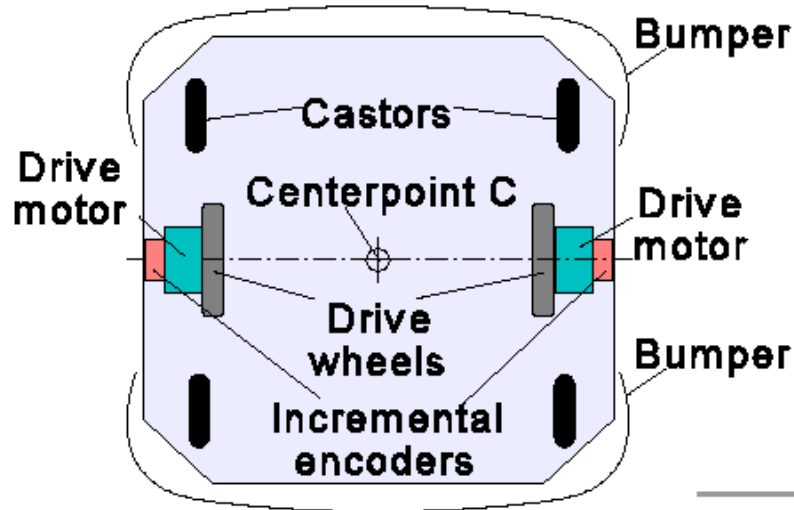
# Odometry - Dead Reckoning

- It is based on counting the robot wheel turns (measured by the encoders) during navigation
- It provides a good accuracy for small movements
- The error tends to cumulate in time with the distance run (poor accuracy for long distances).
- The odometric information is rectified by using alternative localization methods.



# Odometria - Dead Reckoning

Example of odometry calculation (for small movements)



$$p = (x, y, \theta)$$

$$p' = (x', y', \theta')$$



# Odometry - Dead Reckoning

In a time interval  $T$ , the right and left wheel encoders measure an increase of  $N_L$  e  $N_R$ , respectively

$$C_m = \pi D/n C_e$$

where

$C_m$  = conversion factor that translates the encoder steps in linear distance run by the wheels

$D$  = nominal wheel diameter

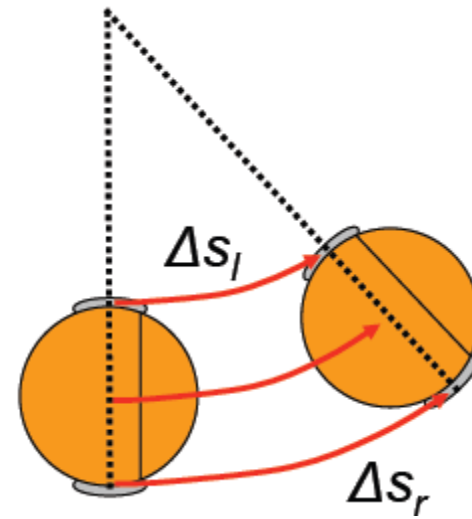
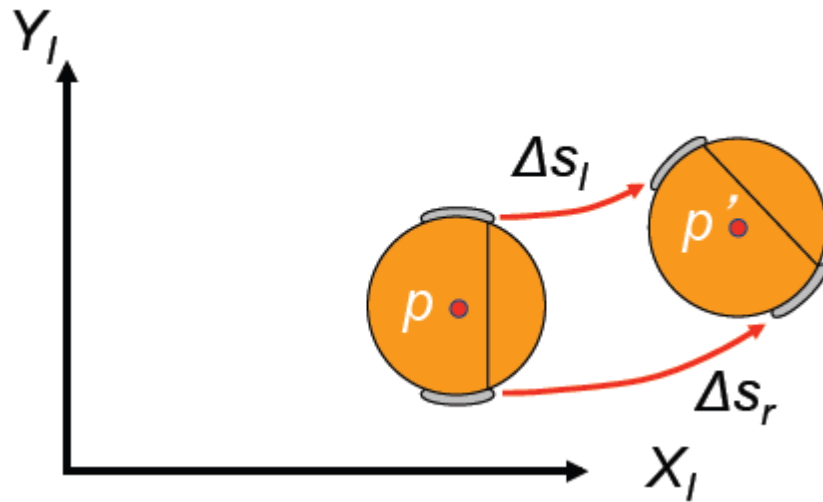
$C_e$  = encoder resolution

$n$  = reduction ratio between motor (where the encoder is) and wheel





# Odometry – dead reckoning



# Odometry - Dead Reckoning

The distance run by the right and left wheels,  $\Delta S_L$  e  $\Delta S_R$ , can be computed as

$$\Delta S_{l/r} = C_m N_{L/R}$$

The distance run by the robot centre  $\Delta S$  is:

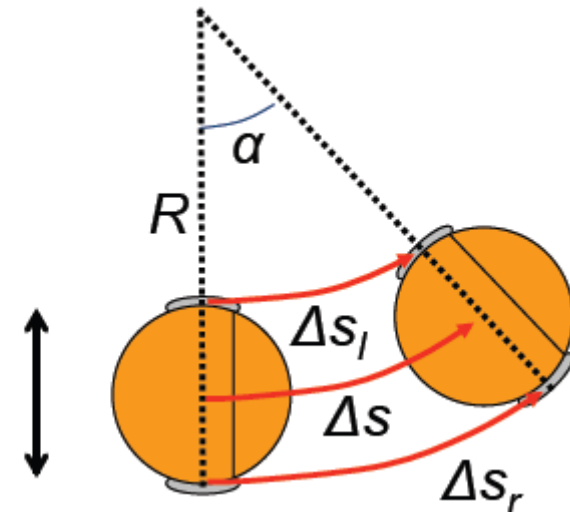
$$\Delta S = (\Delta S_l + \Delta S_r)/2$$

while the robot orientation angle increases of

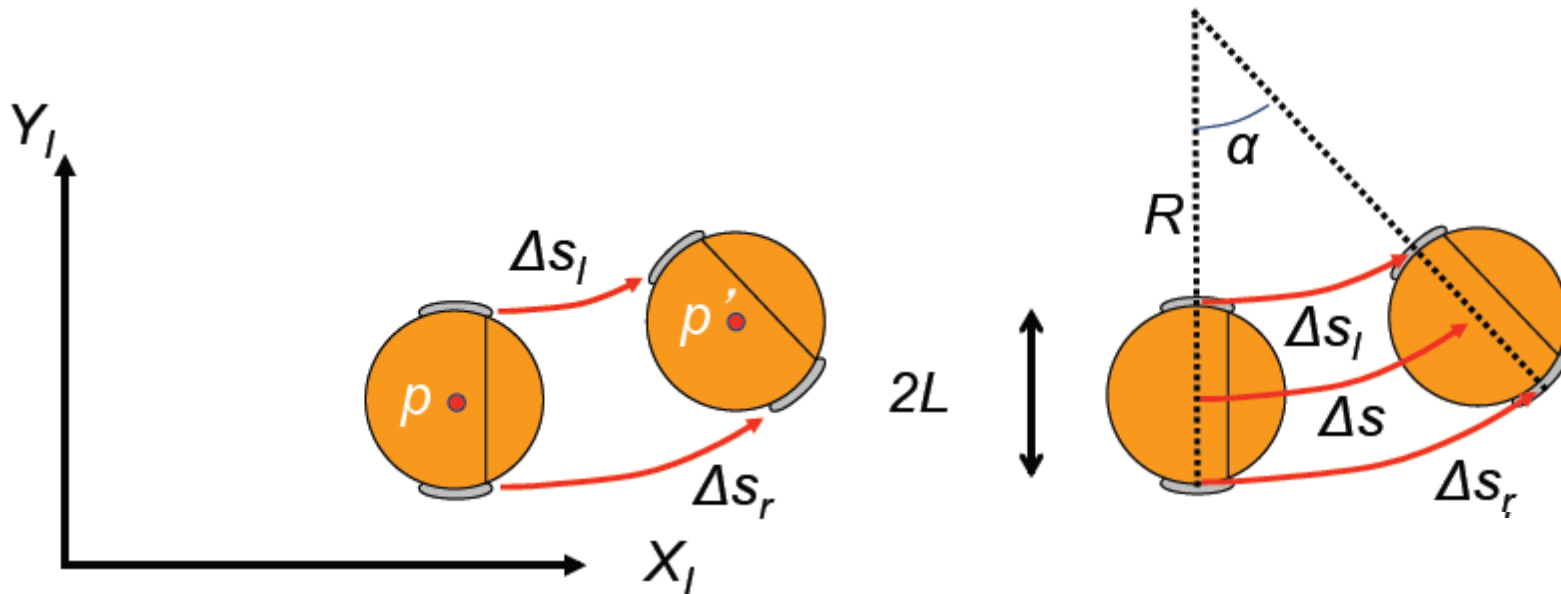
$$\alpha = \arctg (\Delta S_r - \Delta S_l)/2L$$

where  $2L$  is the distance between the wheels (ideally measured as the distance between the points of contact of the wheels on the terrain)

$2L$



# Odometry – dead reckoning



The new robot position  $p'$  is:

$$\theta' = \theta + \alpha$$

$$x' = x + \Delta S \cos \alpha$$

$$y' = y + \Delta S \sin \alpha$$

where  $(x, y, \theta)$  was the position of the robot centre  $p$



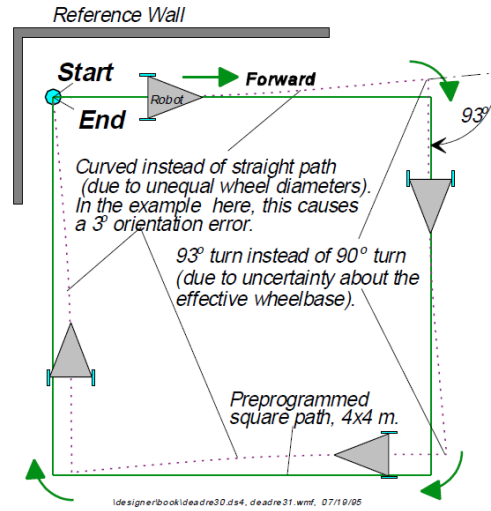
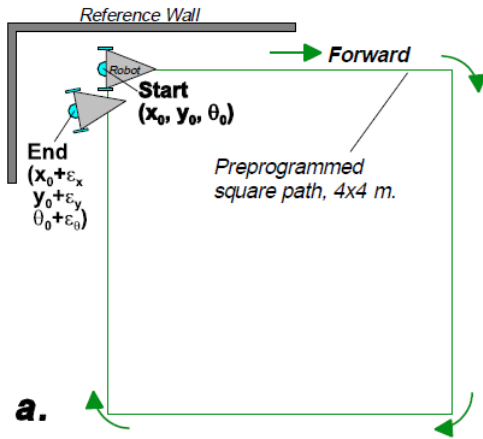
# Odometry - Dead Reckoning

Odometric errors are of two types:

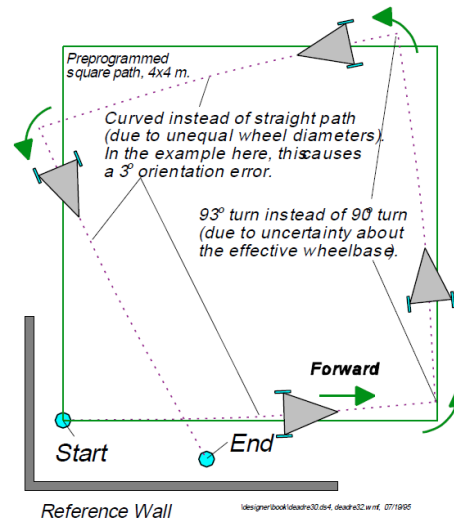
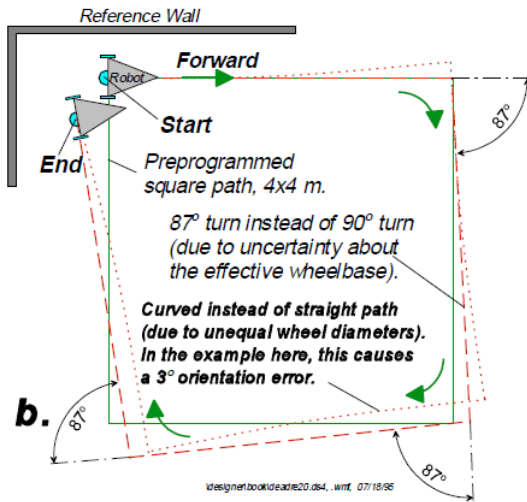
- **Systematic Errors**, due to:
  - different diameters of the two wheels
  - actual size of wheels different from nominal size
  - misalignment of the wheels
  - encoder resolution
- **Non systematic errors**, due to:
  - movements on uneven terrains
  - movements on unexpected objects
  - wheel slippage due to
    - high accelerations
    - slipping terrains
    - external forces (obstacles)



# Odometry - Dead Reckoning



The effect of the two dominant systematic dead-reckoning errors  $E_b$  and  $E_d$ . Note how both errors may cancel each other out when the test is performed in only one direction.



The effect of the two dominant systematic odometry errors  $E_b$  and  $E_d$ : when the square path is performed in the opposite direction one may find that the errors add up.

The unidirectional square path experiment.

a. The nominal path.

b. Either one of the two significant errors  $E_b$  or  $E_d$  can cause the same final position error.



# Active beacons

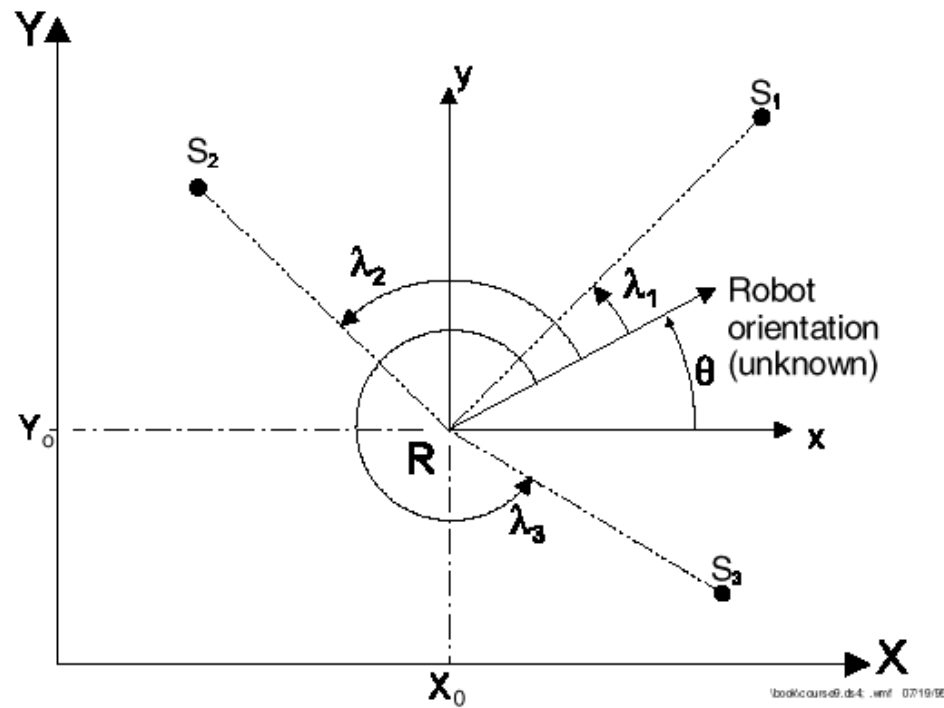
- Localization systems based on active beacons are composed of a set of receiver/transmitter devices (beacons) whose absolute position is known and which are detectable by a sensor on-board the robot



# Active beacons

The localization algorithm is based on the triangulation procedure:

- A rotating unit on the robot can measure the angles  $\lambda_1, \lambda_2, \lambda_3$
- By knowing the position of the 3 beacons, it is possible to derive the robot absolute position  $(X, Y, \theta)$  by triangulation



# Localization systems based on maps

- Localization systems based on maps, or *map matching*, use one or more sensory systems for building a local map.
- The local map is compared to a global map previously stored.
- If a matching is found, the robot finds its position and orientation in space.
- A map can be a CAD model or it can be built using the robot sensory system.

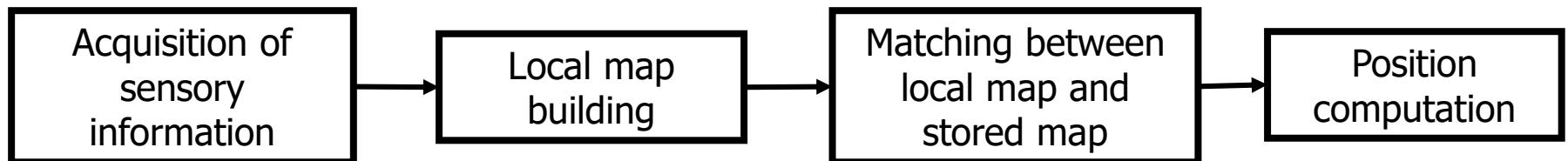




# Localization systems based on maps

To simplify the problem, the current robot position estimated by odometry can be used.

Steps of the localization procedure:

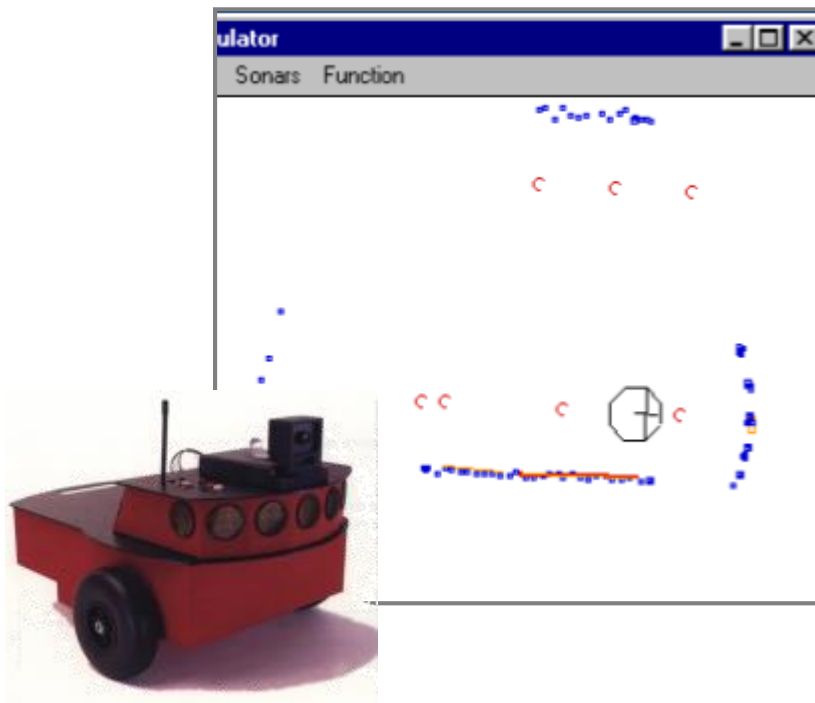


# Localization systems based on maps

- Sensory systems:

Ultrasound sensors

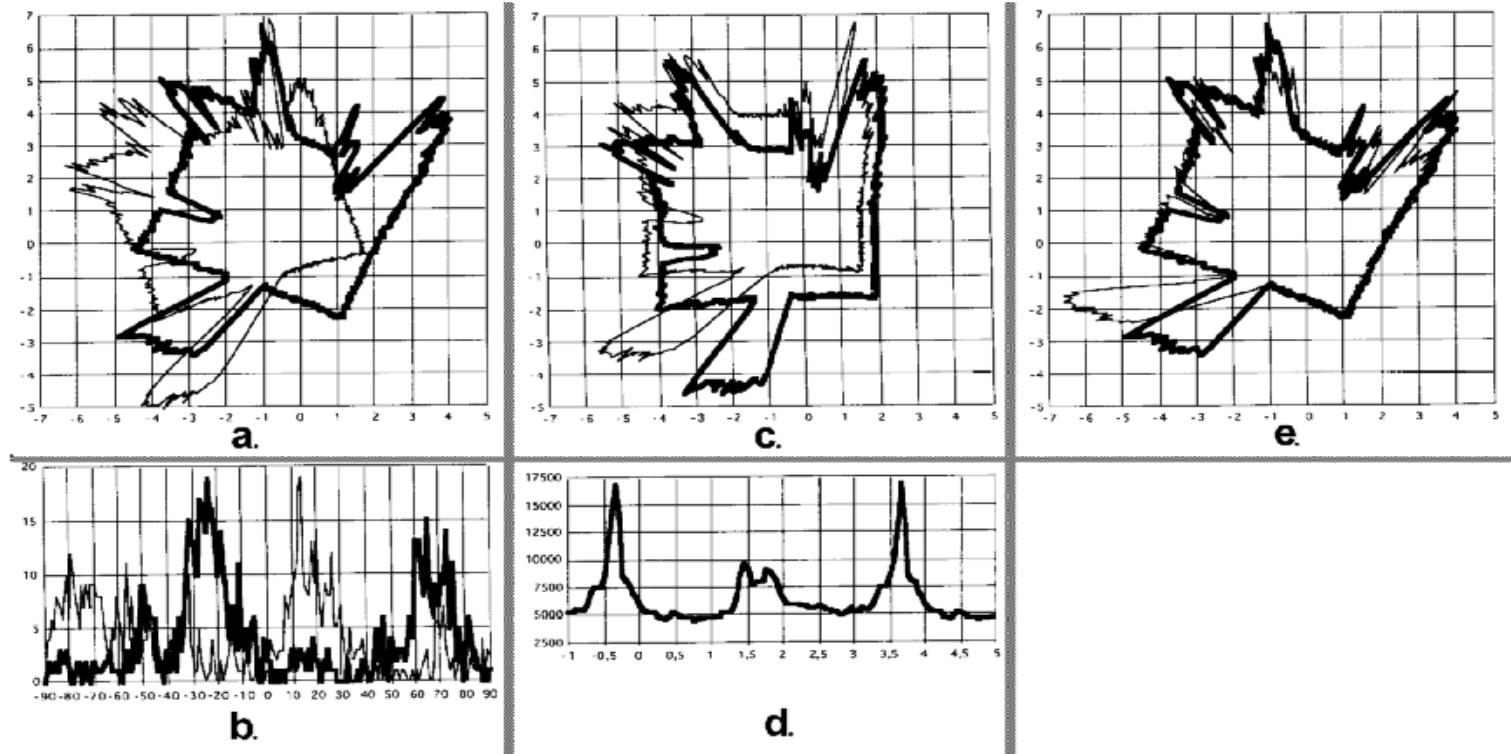
Laser Ranger



# Localization systems based on maps

Mapping techniques:

- Correlation



# Landmarks

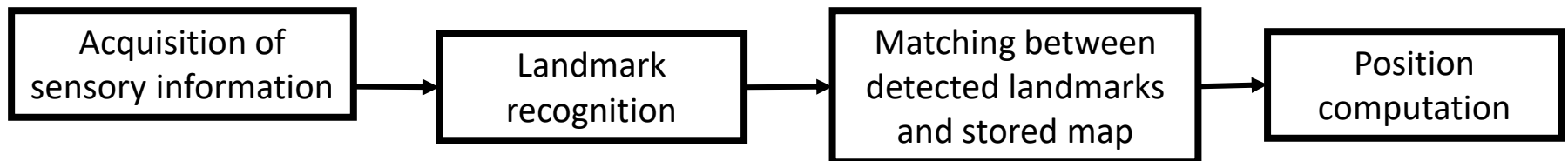
- Landmarks are characteristic shapes that the robot can recognize by using its sensory systems.
- Landmarks can be geometric shapes (e.g. boxes, lines, circles, ..) and they can contain additional information (e.g. bar-code).
- Landmarks are chosen so as to be easily recognised by the robot.
- The position and the characteristics of the landmarks need to be stored in the robot database.



# Landmarks

To simplify the problem, the current robot position estimated by odometry can be used.

Steps of the localization procedure:



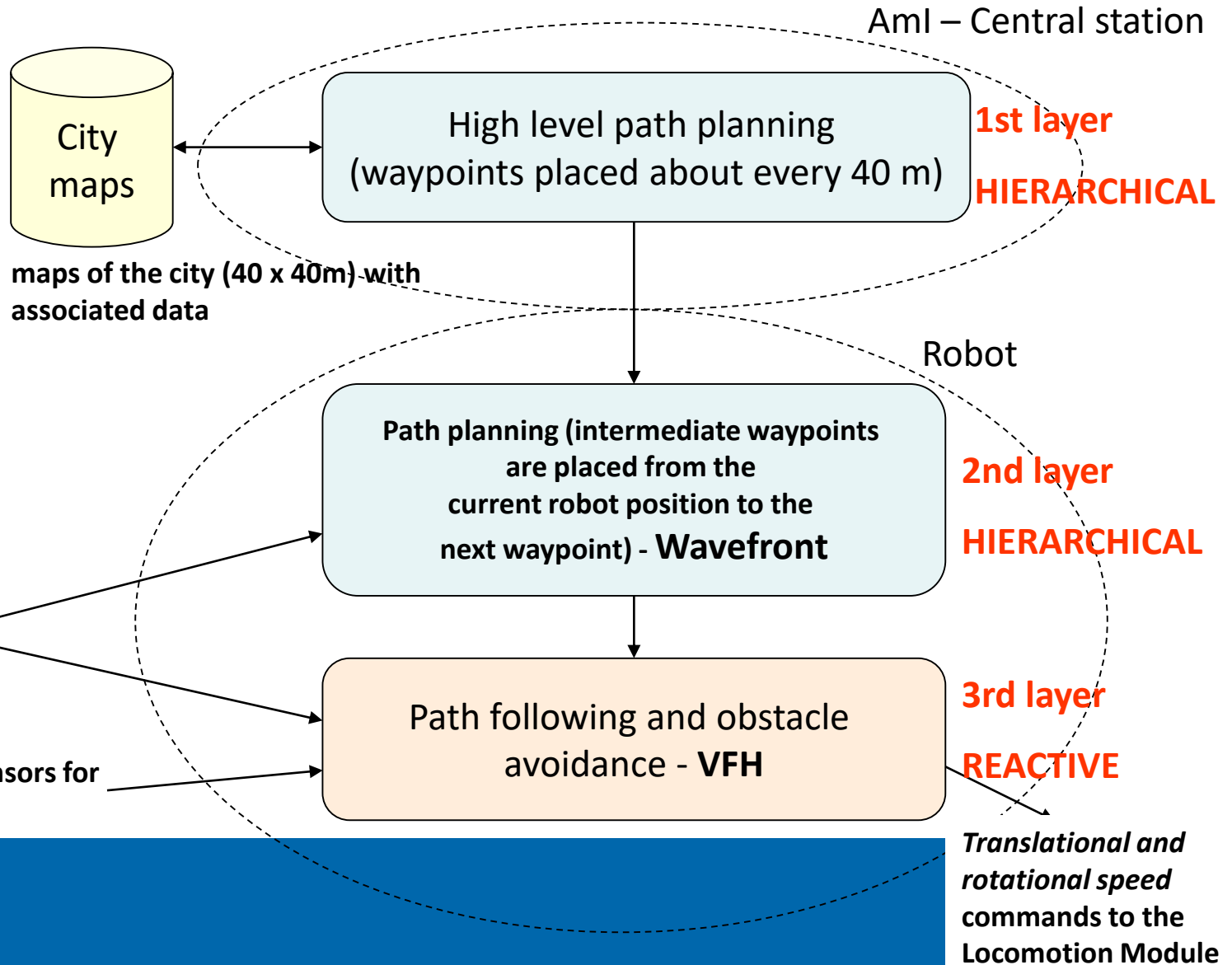
# Landmarks

- Landmarks can be divided in
  - **Natural Landmarks:** objects already present in the environment, with specific characteristics (e.g. lights, corridors, doors, etc.).
  - **Artificial Landmarks:** objects or markers purposively developed and placed in the environment to allow robot localization.



# DustCart hybrid architecture

3 layers:





# DustCart



<https://www.youtube.com/watch?v=gEy91PGGLR0>

<https://www.youtube.com/watch?v=ftouPdU1-Bo>

