Motion planning

Javier González Jiménez

Content

- Introduction
- Workspace and C-pace
- Global navigation
- Reactive navigation

Introduction

Now that the robot has a **map** and can estimate its **pose**, **what else is left?**

- How does it get to a destination? → Planning and Navigation
- How to integrate and manage all the robot functionalities (software)? → Robot control architecture

Introduction: Path planning vs Motion planning

Path planning:

- Find the path to go from a point A to a point B?
- Robot kinematics (velocities and accelerations) not considered

But robots are not able to move …

- in any direction (have Non-holonomic constraints): controllable degrees of freedom are less than the total degrees of freedom (e.g. a car)
- with any arbitrary velocity and acceleration

Motion planning takes all these things into account:

- Compute speed and turning commands to be sent to the robot (Non-holonomic constraints apply) to follow a desired *trajectory*
- Explicit consideration of time (*trajectory* instead of *path*)

Different concepts but sometimes both terms are used as synonymous!

Introduction

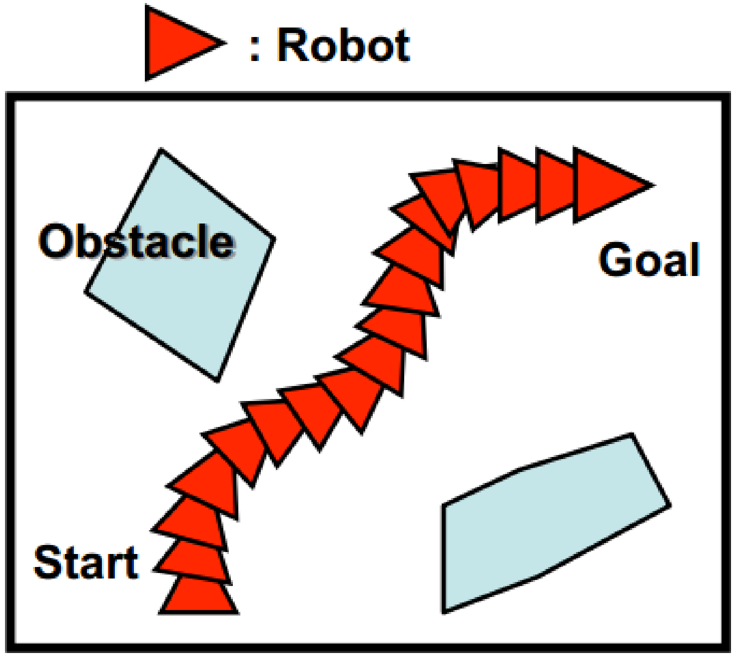
Motion planning problem:

Given:

Workspace
- World geometry
- Start and goal configuration

Robot model
- Robot’s geometry/kinematics

Compute:
- A collision-free, feasible path to the goal



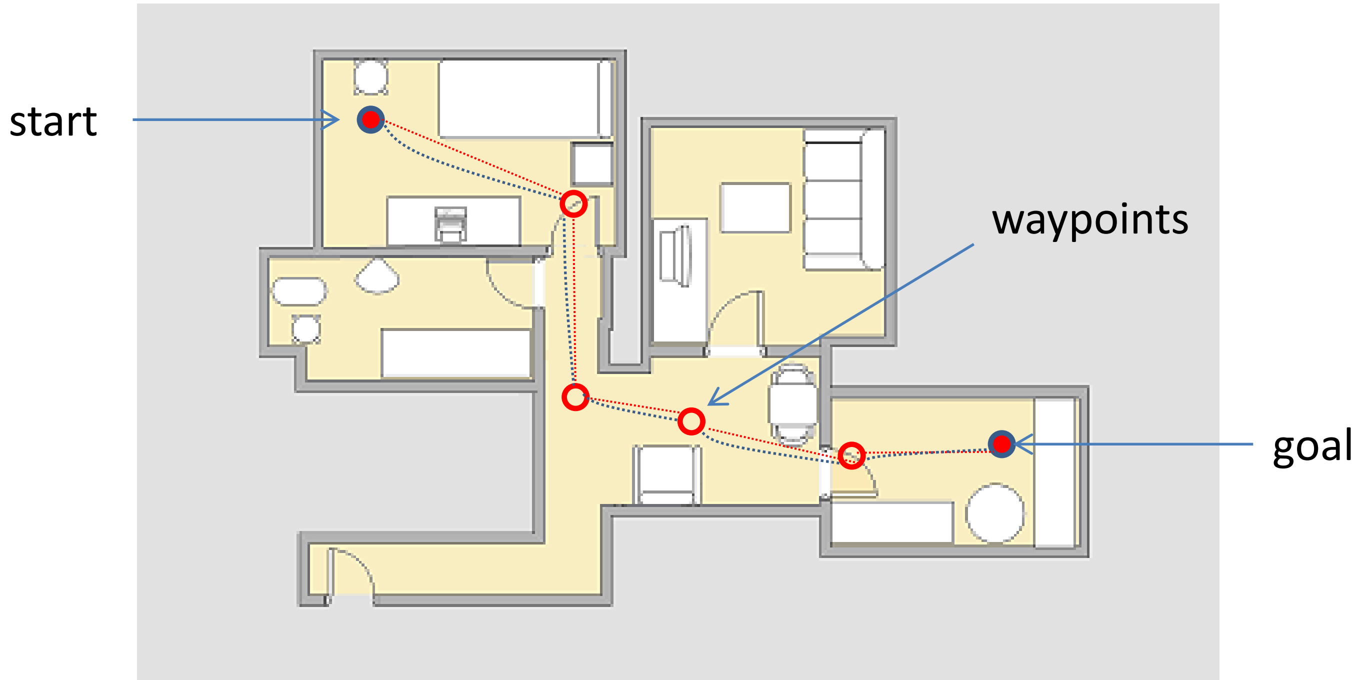
Images from K. Tsianos

Introduction

Usually, motion planning is decomposed in two problems:

1. *global* navigation: Find a sequence of waypoints between the start and goal
2. *local* reactive navigation: Navigate between consecutive waypoints while avoiding obstacles

This strategy is called **Planning with Roadmaps**



Introduction

Global navigation (or ***roadmap navigation***)

- A **RoadMap** is:
  - a kind of topological (not-detailed) route to the goal
  - represented by a sequence of waypoints that the robot needs to reach to get to its destination (goal)
- Input: the whole (known) map
- Technique: **search method** for finding an optimal roadmap to the goal, e.g. A*

Local navigation (or ***reactive navigation***)

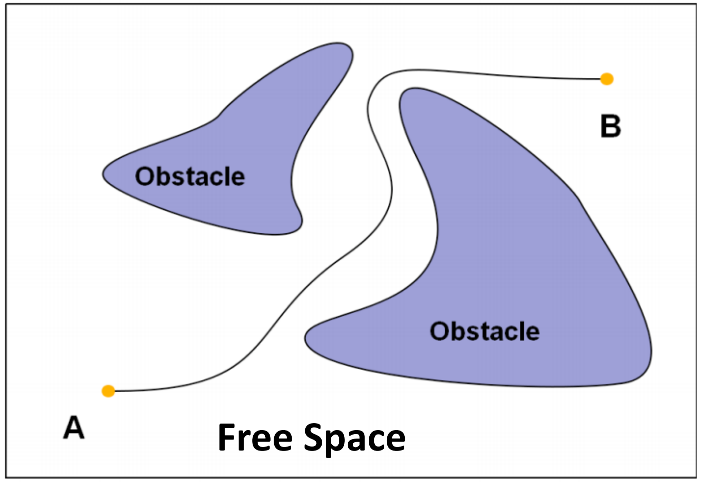
- High-frequency (real-time) generation of a local trajectory between waypoints
- Input: sensor current observation
- Techniques: virtual force field (VFF), vector field histogram (VFH), dynamic window, PT-space (UMA), …

Configuration space

The robot **workspace** consists of...

- Obstacles
  - Occupied spaces of the world
  - Robots can’t go there, neither pass through them
- Free Space
  - Unoccupied space (obstacle-free)
  - Robot, considered as a point, can go through it

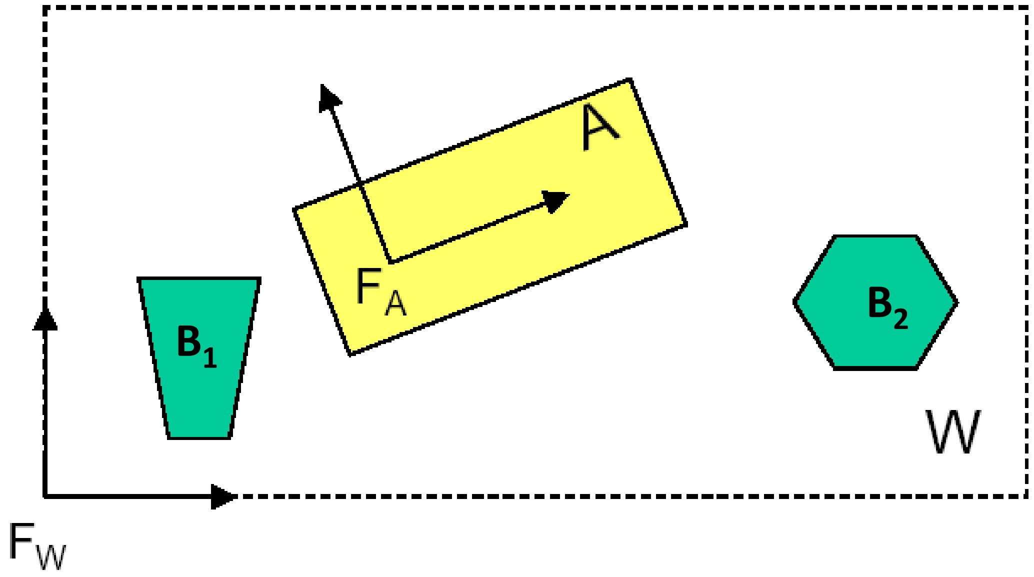
Workspace



But the robot is not a point!

Configuration Space (C-space):

- space of all possible robot poses **q** in a workspace **W**
- depends on the robot shape (**A**) and obstacles ($\mathbf{B_i}$)



A: robot shape

W: Workspace where robot moves

$\mathbf{B_1,\ldots B_m}$: obstacles in **W**

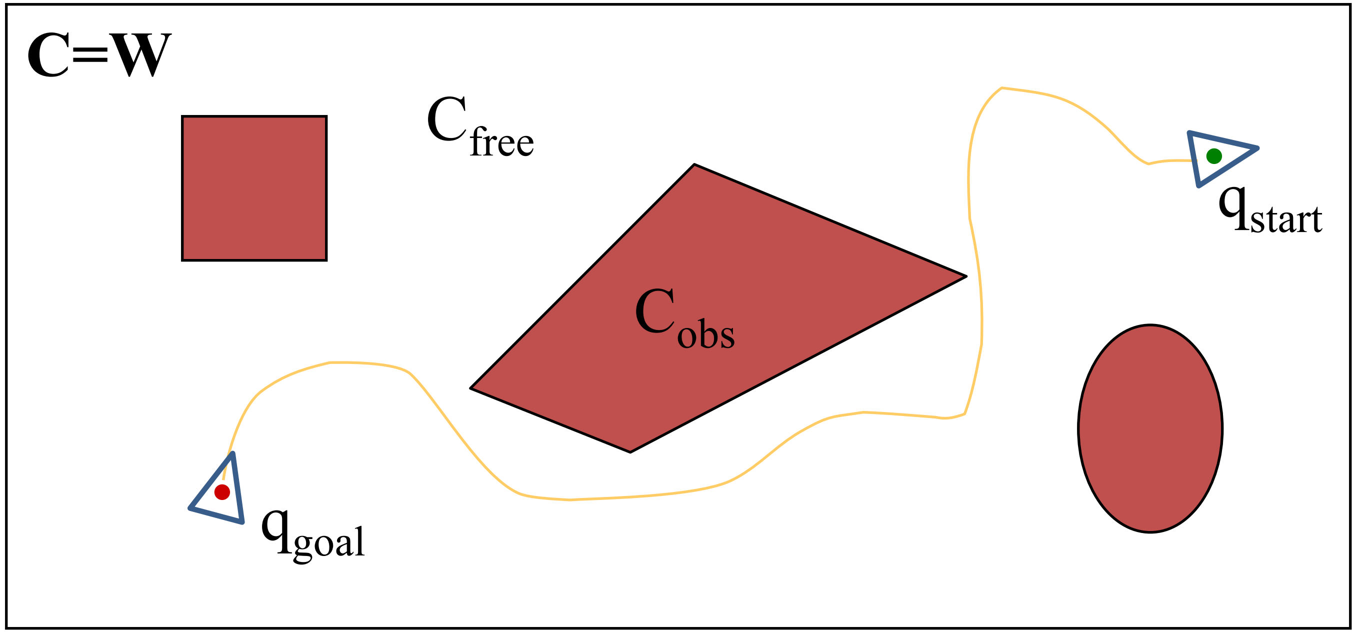
$\mathbf{F_W}$**:** world frame (fixed)

$\mathbf{F_A}$**:** robot frame

Courtesy: John (Jizhong) Xiao
City College of New York

Configuration space

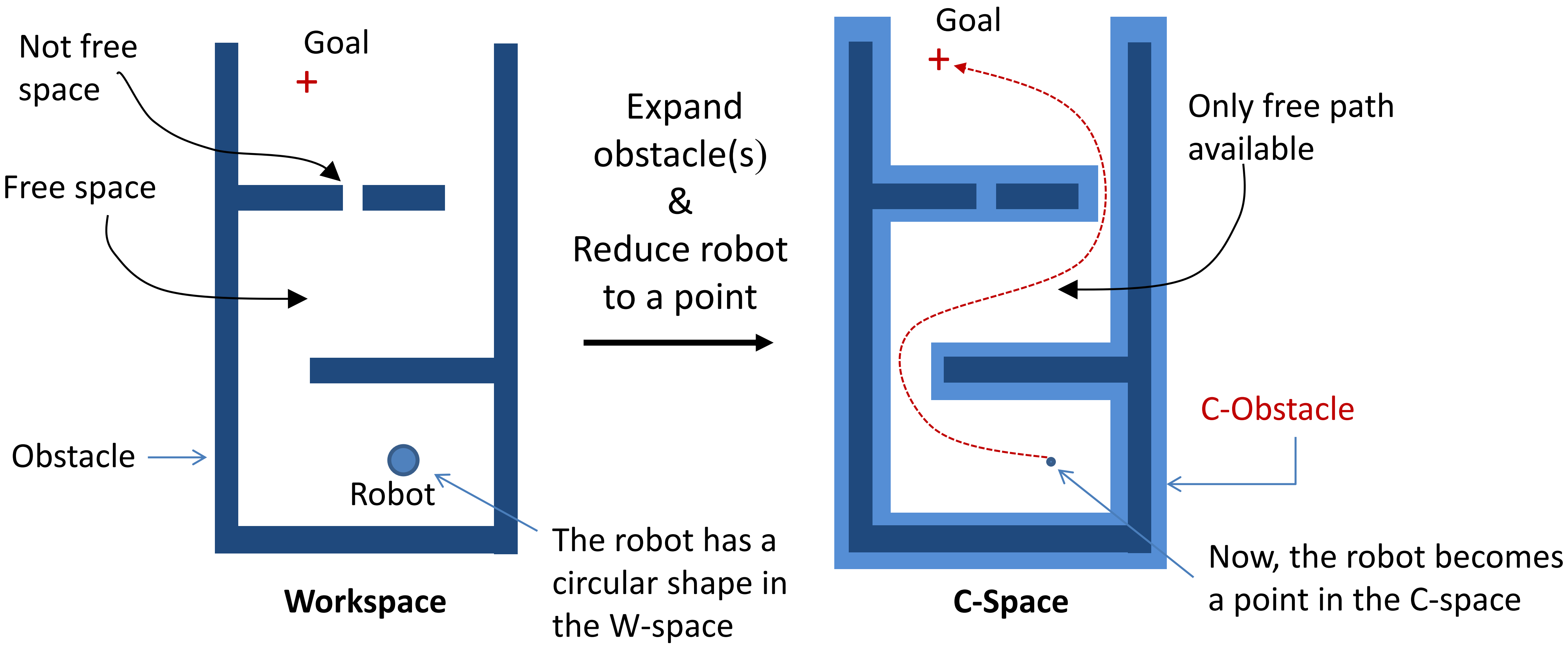
If the robot shape is a point (free-flying, no geometric constraints)
→ C-space = Workspace



Courtesy: John (Jizhong) Xiao
City College of New York

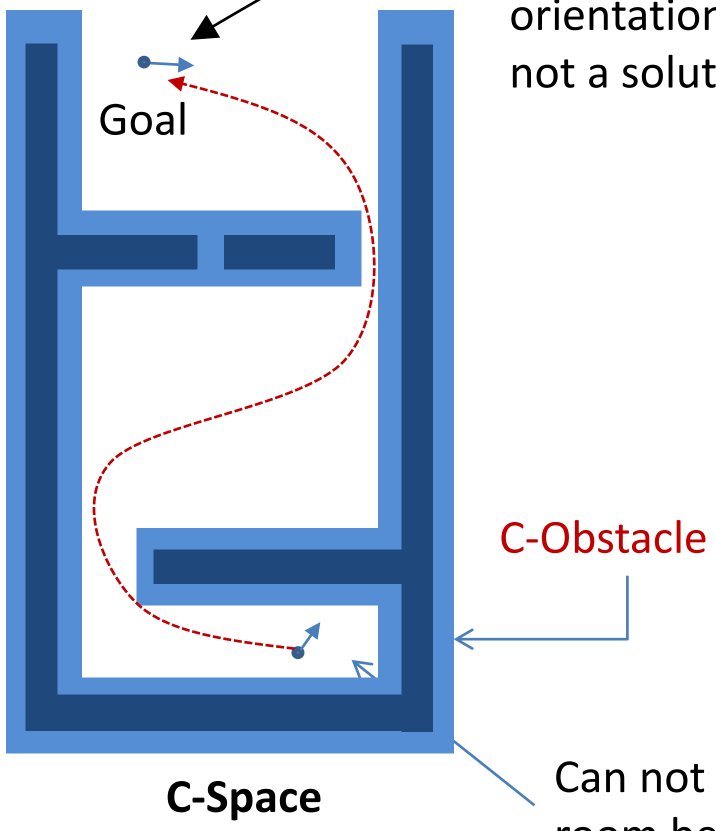
Configuration space

If the robot is not a point but has a circular shape



The robot orientation does not affect

Be aware of the robot orientation when planning in the C-space!



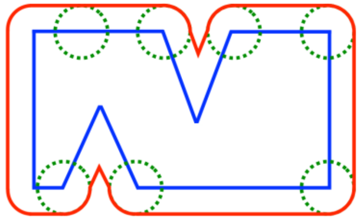
To end up at this orientation, this path is not a solution

Can not be a feasible path to get out of this first room because of the non-holonomic constraint

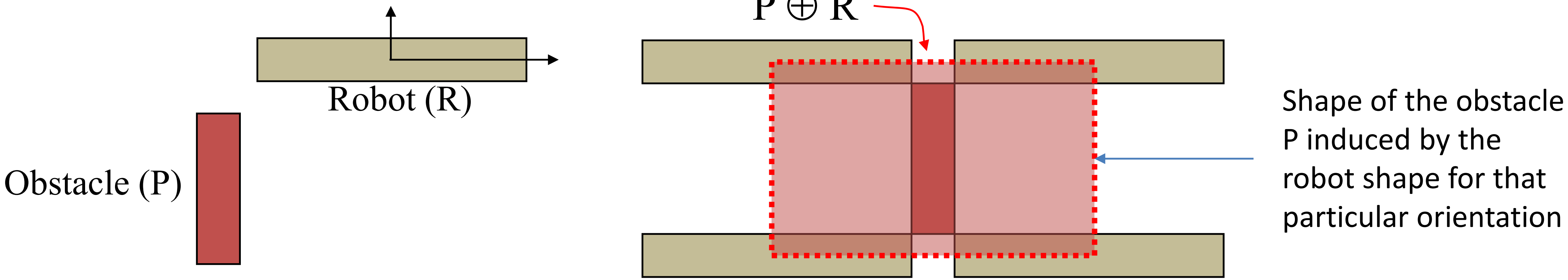
Configuration space

For robots with a polygonal shape …

- Obstacle must be expanded with such shape → depends on the robot orientation
- The expansion of one planar shape by another is done by *Minkowski sum* ⊕ (also known as dilation)



Example: Rectangular robot that **only translates**

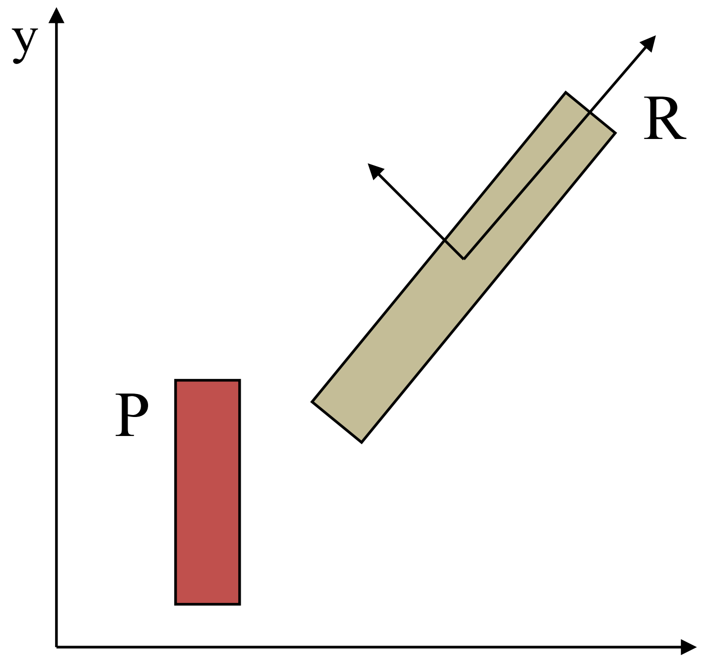


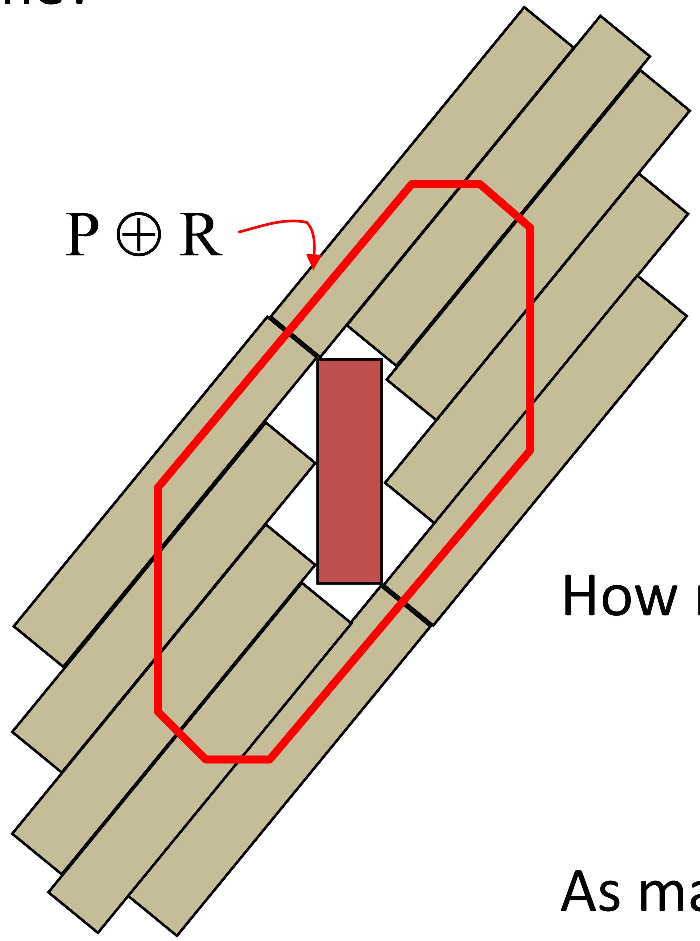
Courtesy: John (Jizhong) Xiao [City College of New York]

Configuration space

What would the C-obstacle be if the rectangular robot can ***translate*** *and* ***rotate*** in the plane?

For example: robot at 45 degrees





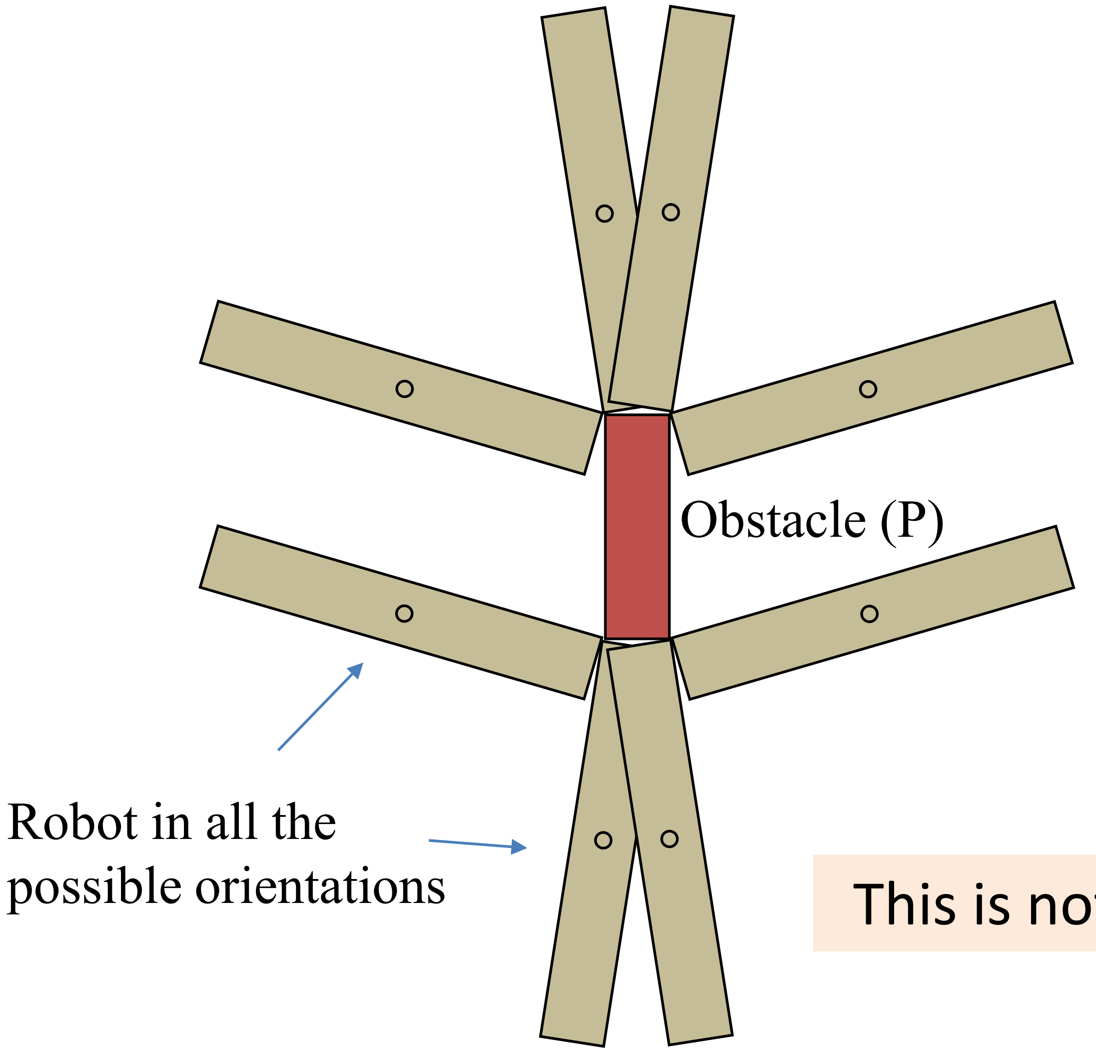
How many shapes does P ⊕R have?

As many as orientations

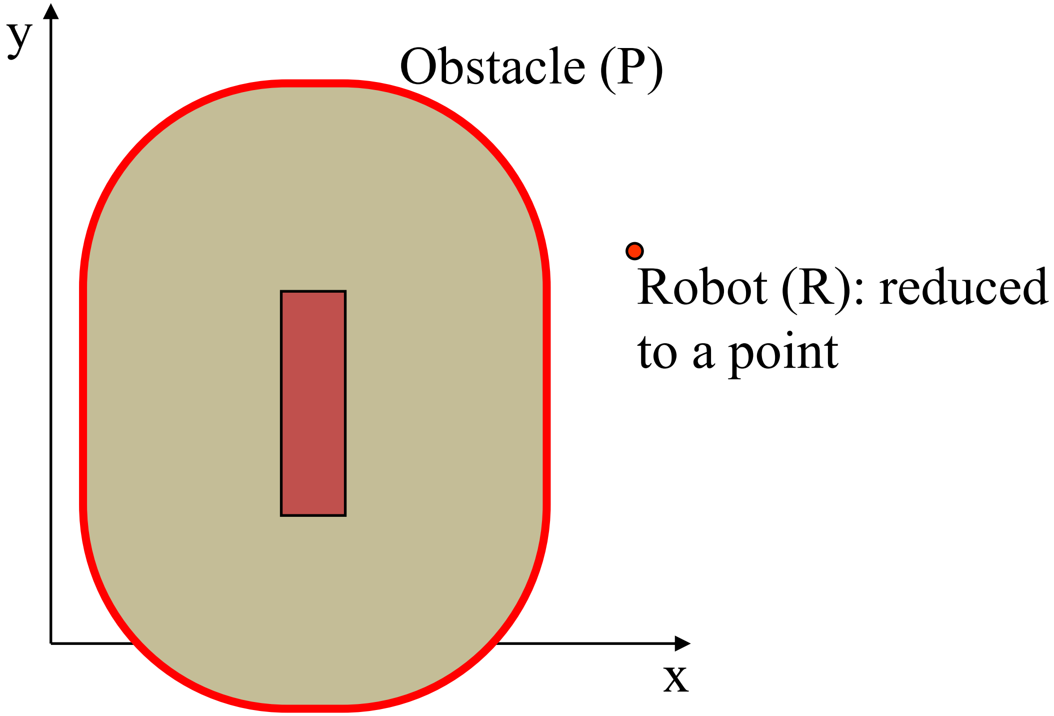
Courtesy: John (Jizhong) Xiao [City College of New York]

Configuration space

Robot can rotate in any orientation



Worst possible case

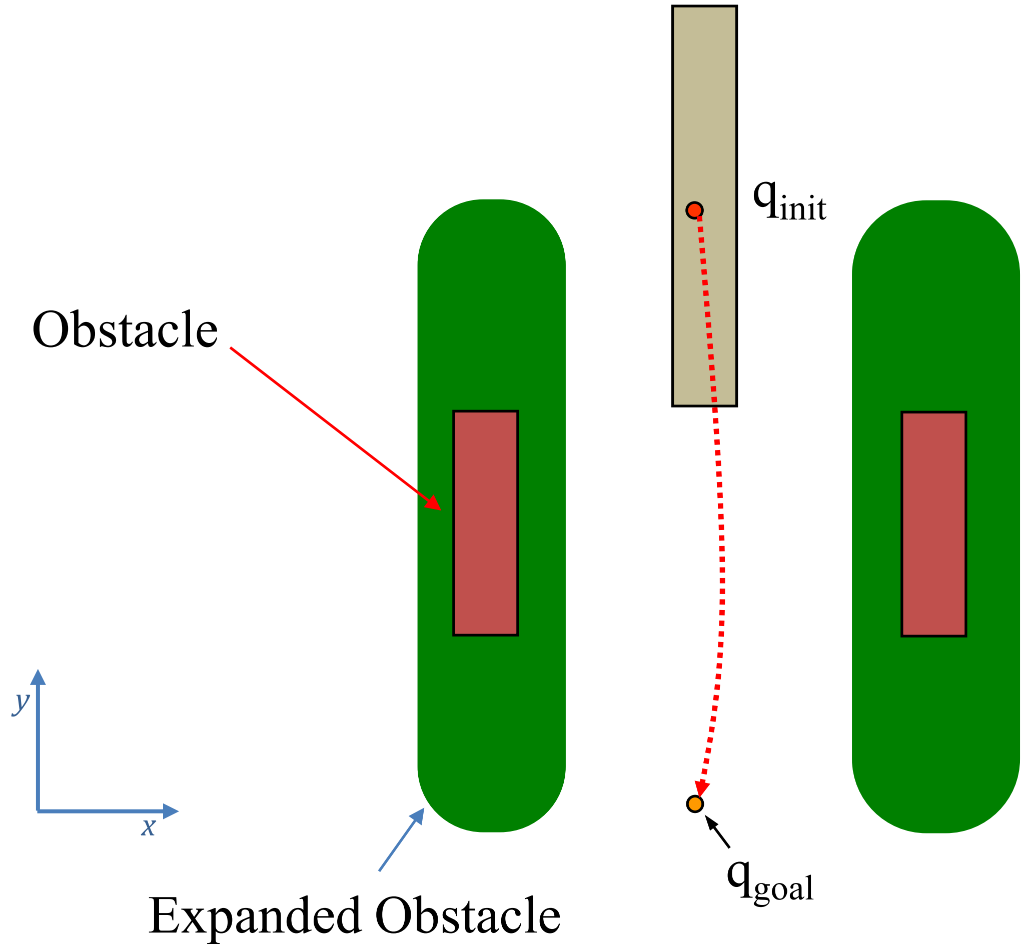


This is not a good space representation for planning, why?

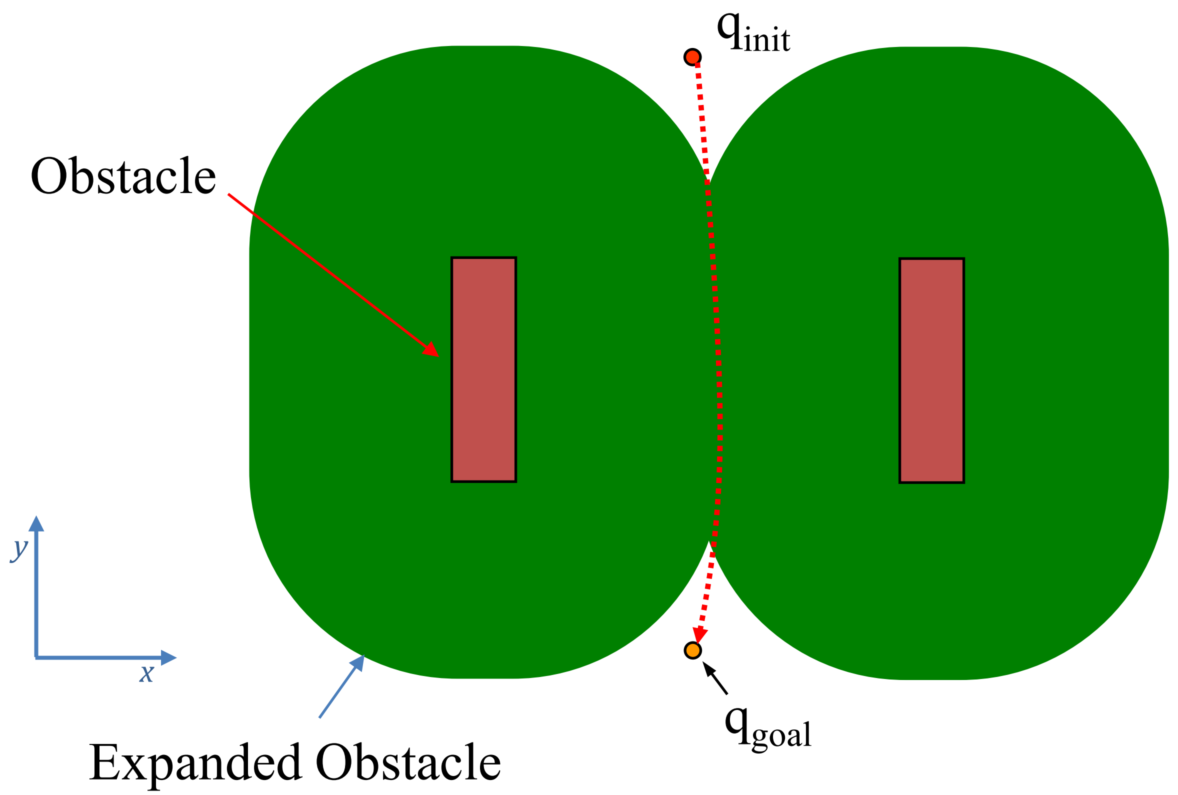
Courtesy: John (Jizhong) Xiao [City College of New York]

Why? Too conservative!

With the robot controlling its orientation there must be a free path!

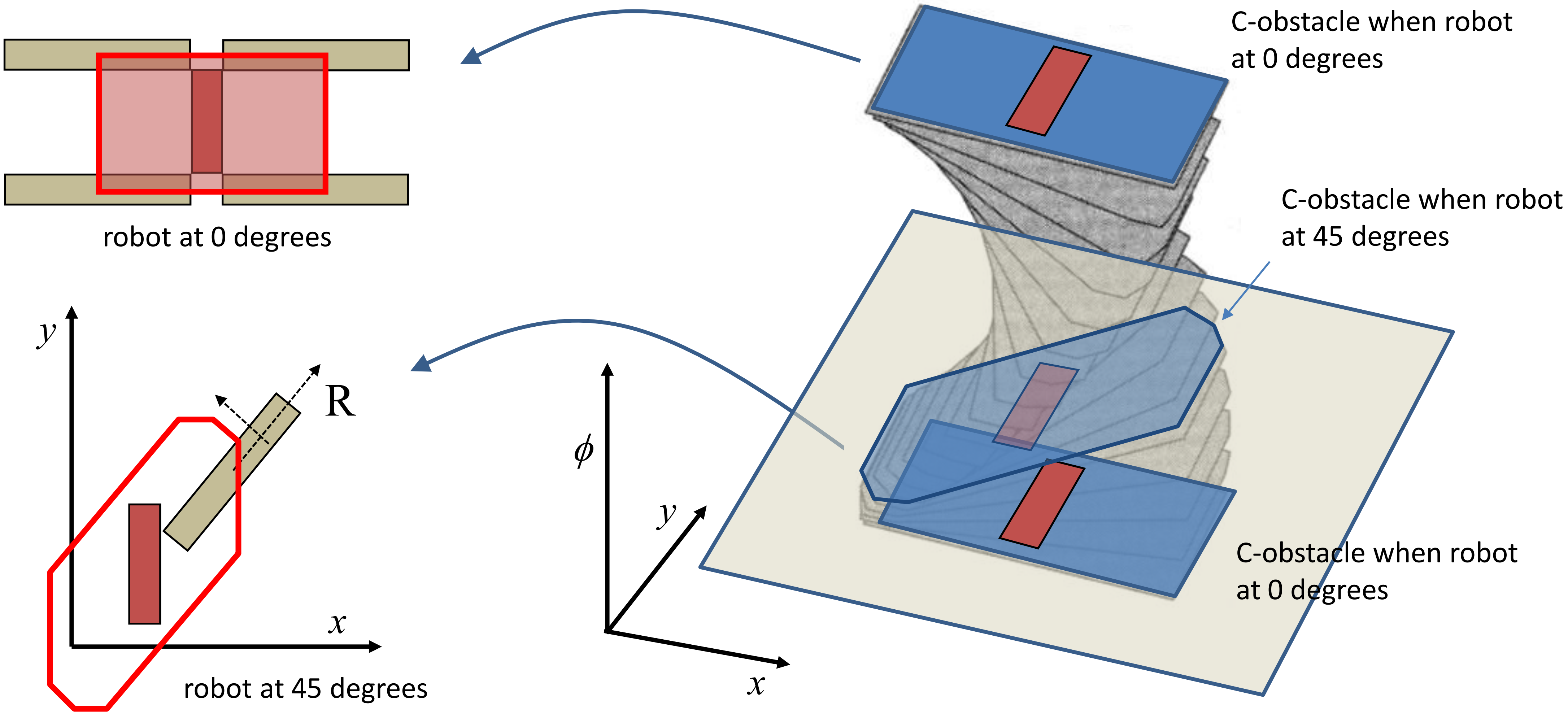


But not for the worst case of the C-space!



Courtesy: John (Jizhong) Xiao [City College of New York]

SOLUTION: C-space in 3D

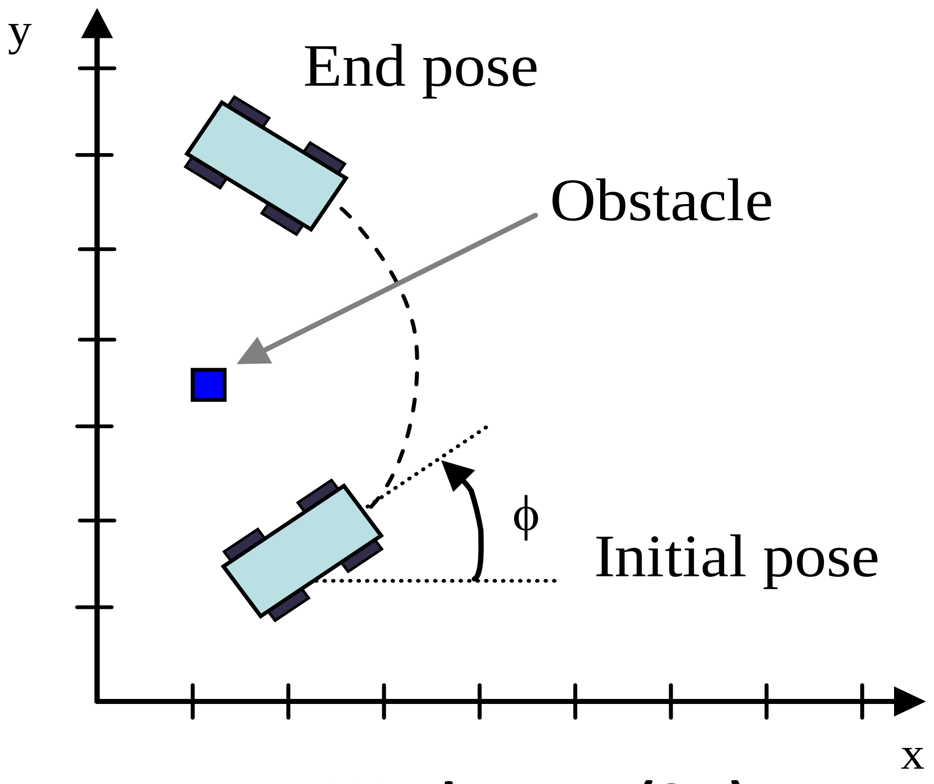


Obstacles becomes C-Obstacles (shape depends on the robot orientation)

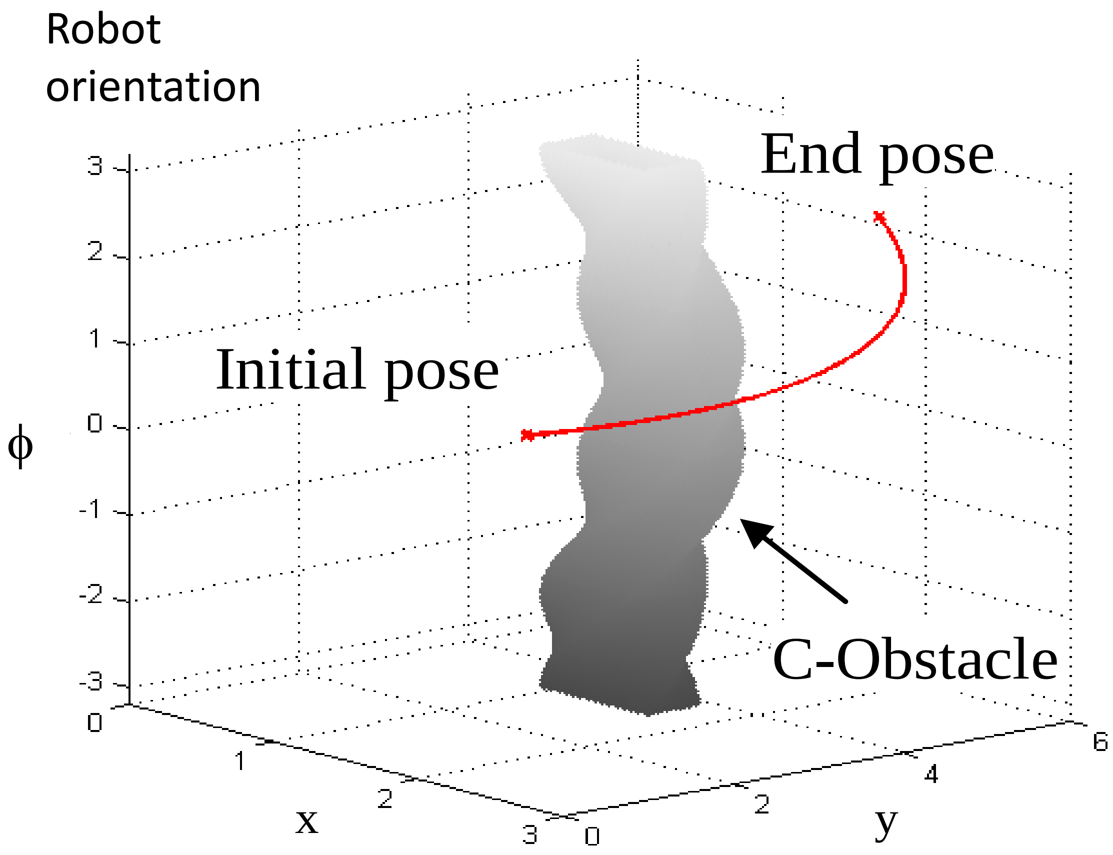
Courtesy: John (Jizhong) Xiao [City College of New York]

Configuration space (C-space)

Another example:



Workspace (2D)



C-Space (3D): robot becomes a point

Global navigation

Global Algorithms

Geometry-based algorithms :

- compute nodes and graph edges based on geometric constraints
- Techniques: Visibility graph, Voronoi Diagram, cell decomposition

Sampling-based algorithms:

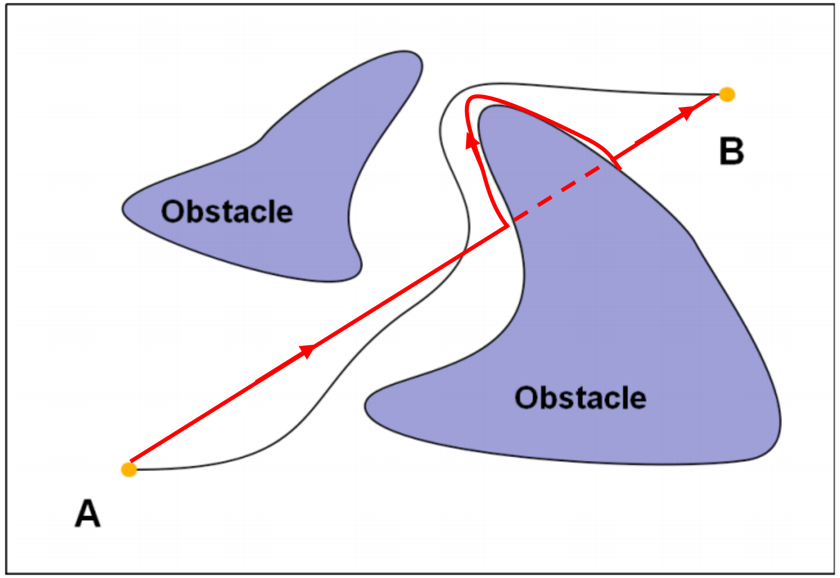
- select fixed or random robot configurations (poses) as nodes and interconnect them based on some constraints
- Techniques: Human-assisted, Probabilistic roadmap, Random Tree expansion

Global navigation: Geometry-based algorithms

Bug algorithm (sort of behavior-based navigation)

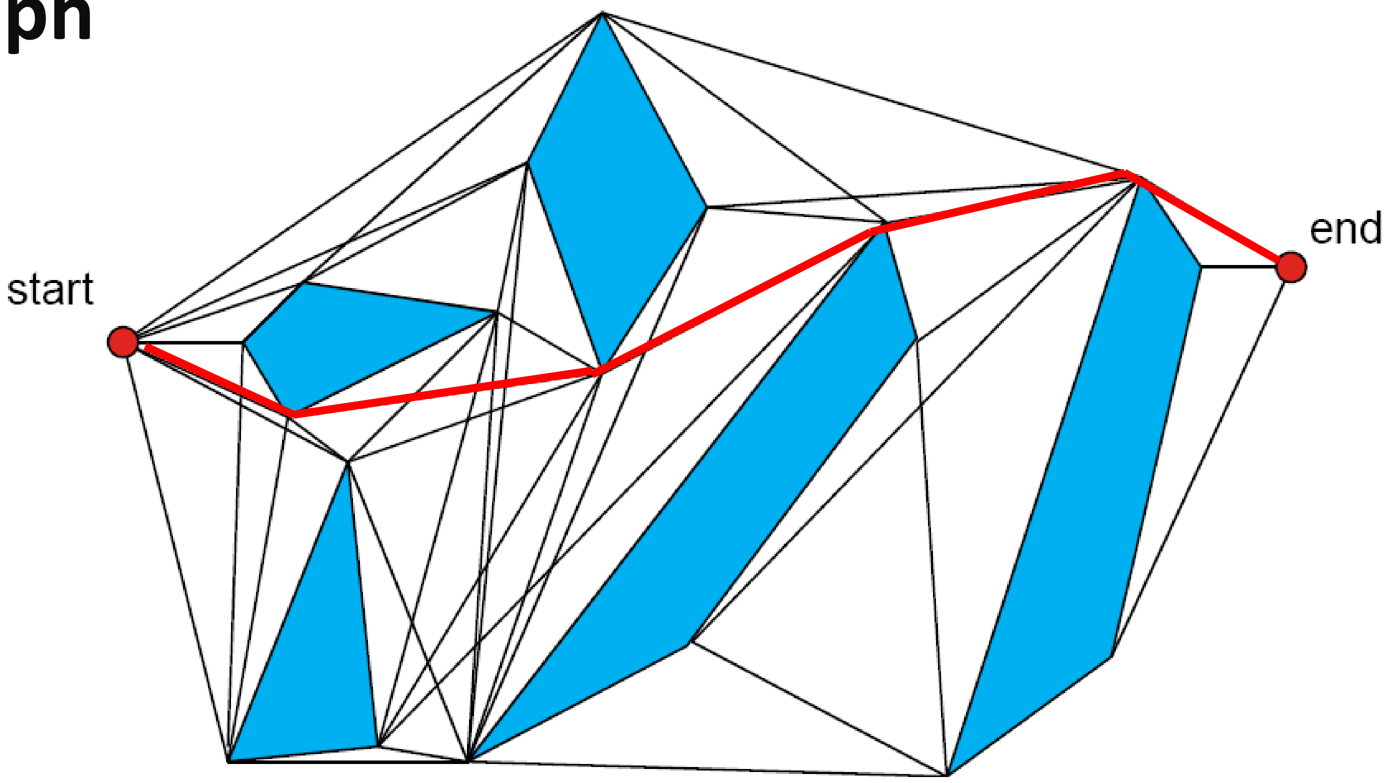
1. starting from A and given the coordinates of a goal pose B, draw a line AB (it may pass through obstacles that are known or are yet unknown)
2. move straight along this line until either the goal is reached or an obstacle is hit
3. on hitting an obstacle, follow the wall until AB is met
4. goto 2

Used by very simple (naïve) robots



Global navigation: Geometry-based algorithms

Visibility graph



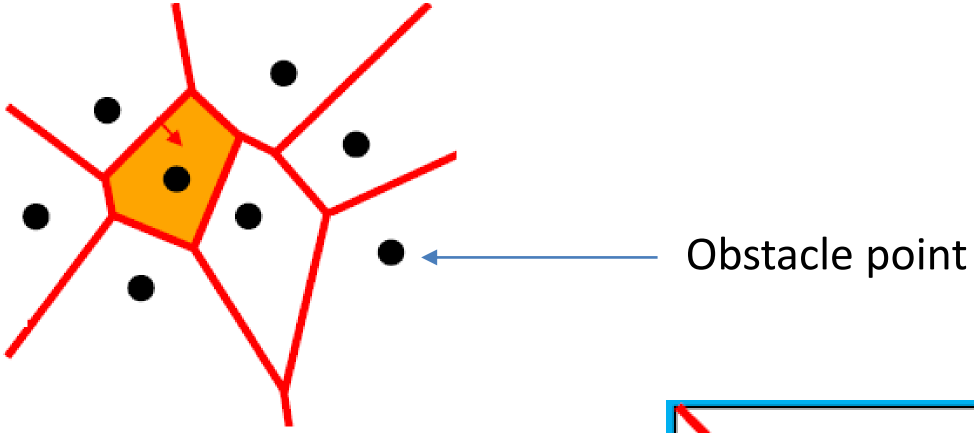
Algorithm

1. Obstacles represented by polygons
2. Connect vertices that are visible: Visibility graph
3. Find the shortest trajectory

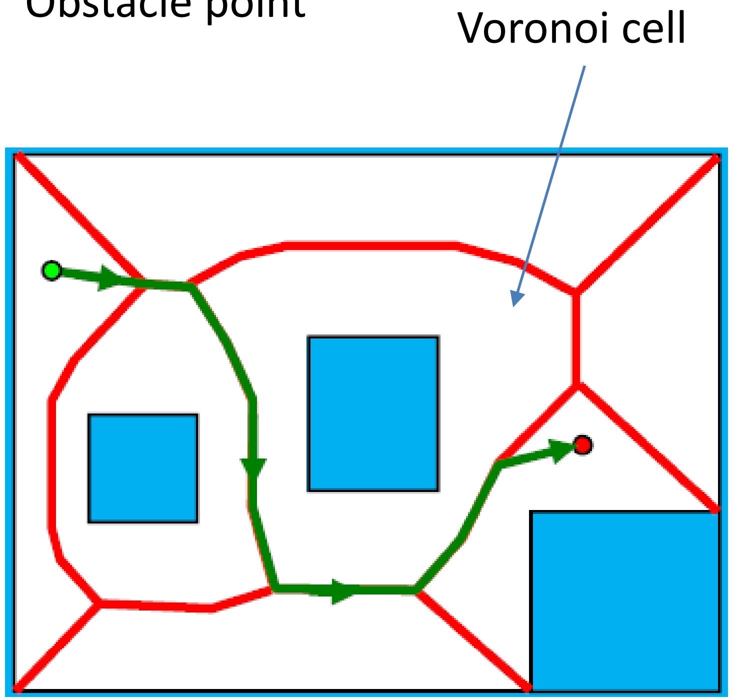
Global navigation: Geometry-based algorithms

Generalized Voronoi Diagram

- The **Voronoi diagram** is the space partition induced by Voronoi cells.



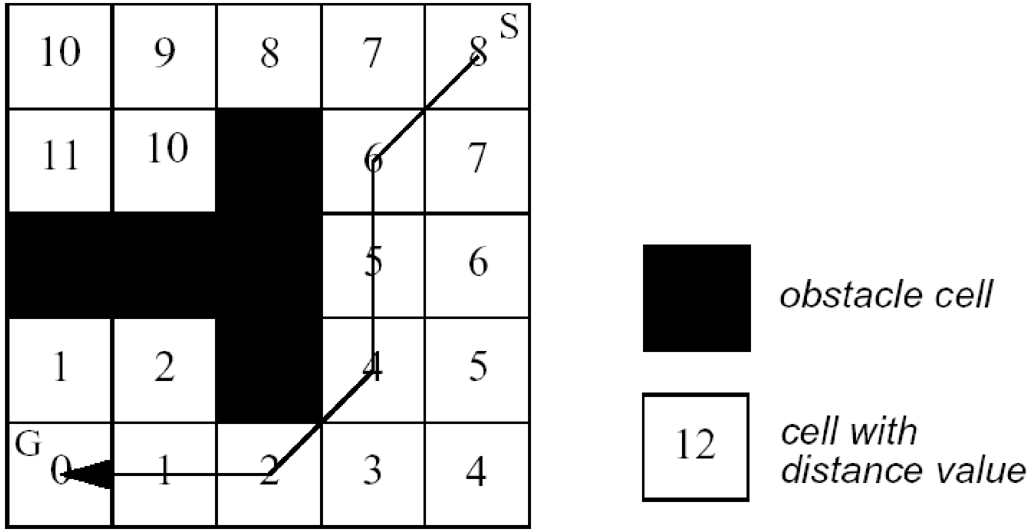
- Considering boundary points as obstacles, a robot would travel along the edges of a Voronoi diagram to keep maximum distance away from the obstacles.



Global navigation: Geometry-based algorithms

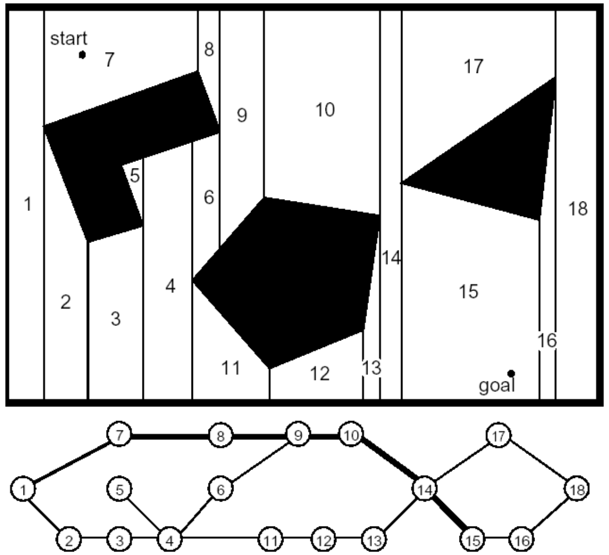
Cell decomposition

- Divide space into simple, connected regions called cells
- Determine which cells are adjacent and construct a connectivity graph (starting from the goal)
- Search for a minimum path in the connectivity graph to join them

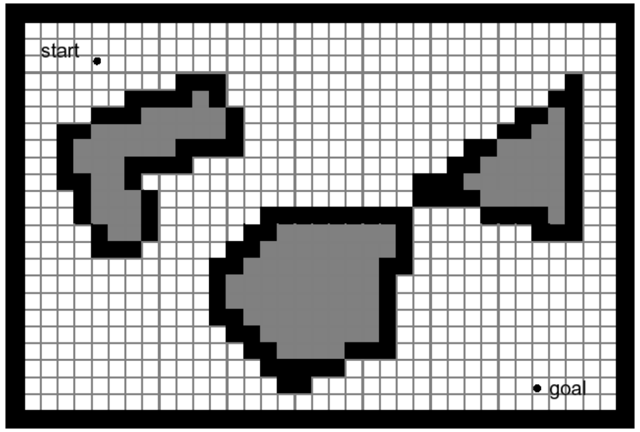


Global navigation: Geometry-based algorithms

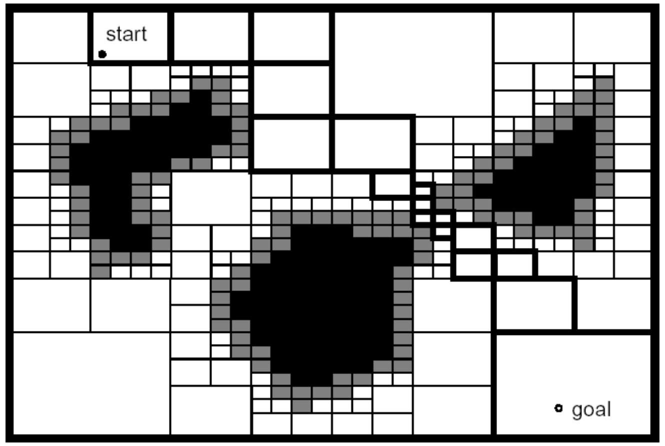
Different ways of Cell Decomposition



Exact



Approximate (using occupancy grid map)

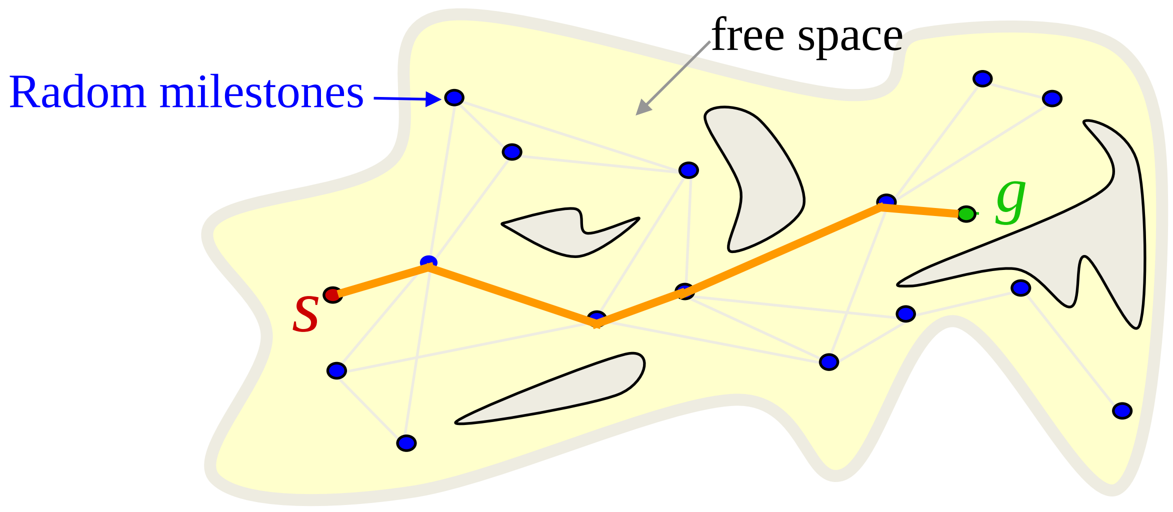


Adaptive

Global navigation

Sampling-based algorithms

Idea: Choose fixed or random valid robot positions (milestones) and interconnect them based on proximity to form a navigable path.



Different techniques to generate the milestones:

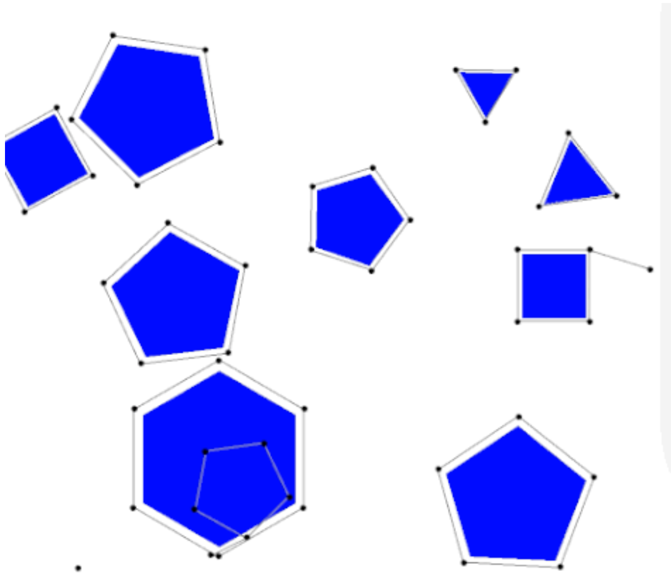
- Manually (a human decides where the milestones are)
- Probabilistic sampling
- Random Tree Expansion

Not covered here

Global navigation: Sampling-based algorithms

Probabilistic Road Maps (PRM)

- Generate random points (milestones) in free-space and connect those that represent valid short path lengths



Courtesy: Wikipedia

Kavraki; Svestka; Latombe; Overmars. (1996), "Probabilistic roadmaps for path planning in high-dimensional configuration spaces",

- Solution depends on how many nodes (n) are generated and how much connectivity (k) we want between nodes: typically, the **k nearest neighbors** or all neighbors less than some **predetermined distance**.

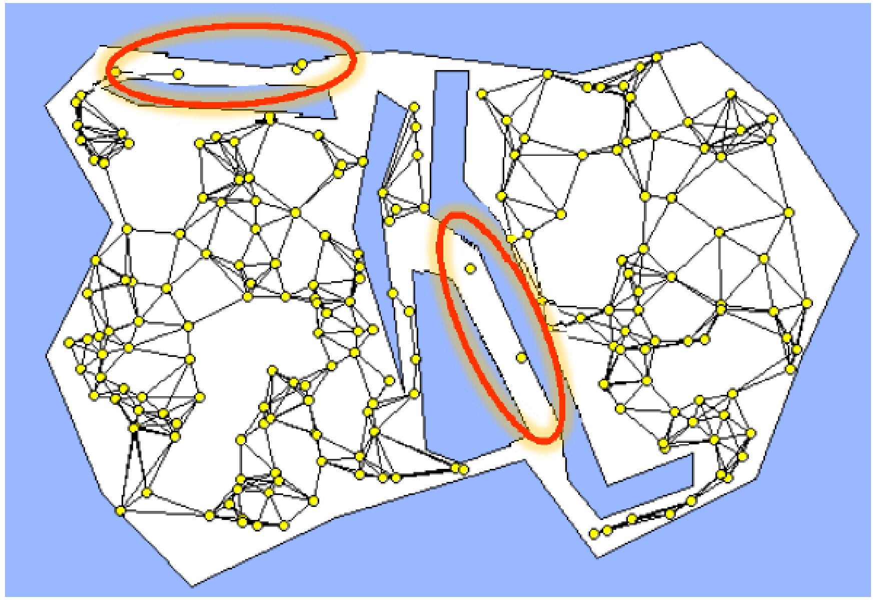
Global navigation: Sampling-based algorithms

Probabilistic Road Maps (PRM)

ALGORITHM

- Build graph
- Include start and goal in the graph
- Find optimal path in the graph from start to goal

Courtesy: Mark Lanthier
[scs.carleton.ca]

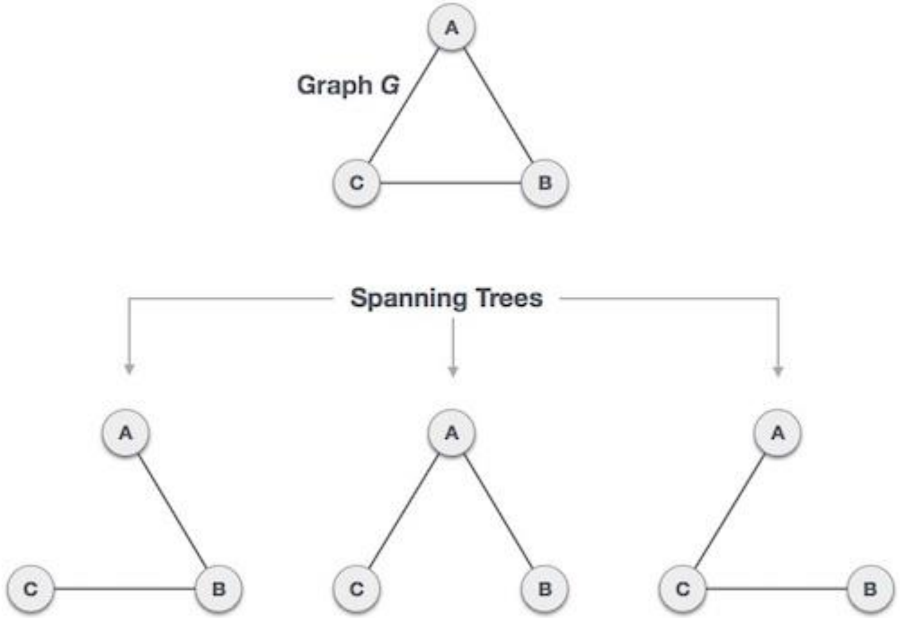


- Simple implementations can lead to not representative graphs (including disconnected graphs)
- Sophisticated solutions exist to generate optimal samples poses and connections between them

Optimal path search

Given a graph, a start and goal nodes, find out the best path according to a cost function

Spanning tree of the graph



A spanning tree

- is a subset of Graph G, which has all the vertices covered with minimum possible number of edges.
- does not have cycles and cannot be disconnected.

How to span the graph to a tree

- Breadth-First Search
- Depth-First Search
- A*

Global navigation

A* algorithm

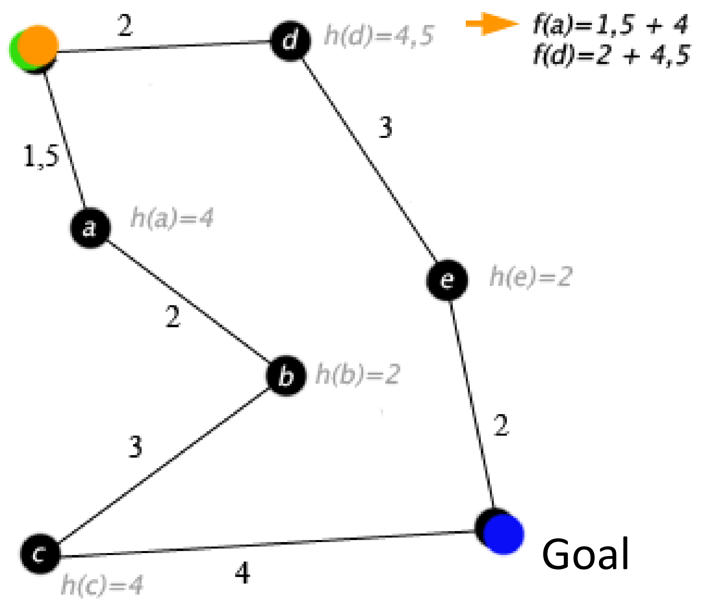
Search for a path between two nodes of a graph which minimizes a **cost function**

$$f(n) = g(n) + h(n)$$ n: node (of the graph)

g(n): Cost of the arc to go to node n (e.g. distance between nodes –milestones-)

h(n): heuristic to determine the order in which the search visits nodes

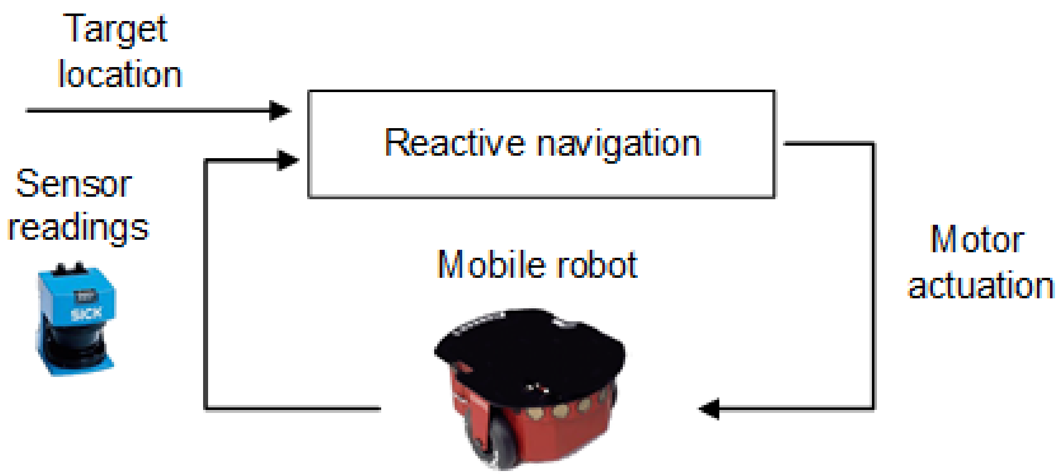
Start

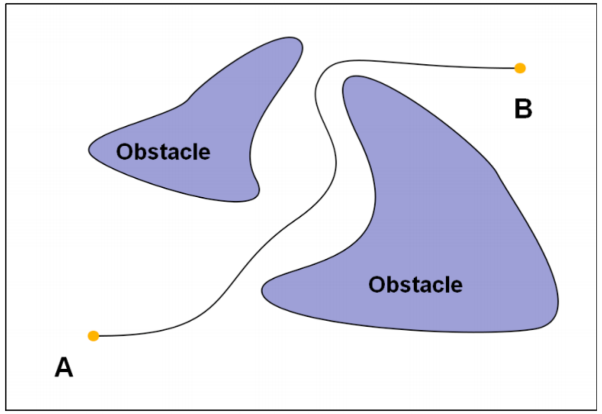


Goal

Reactive navigation

- **Objective**: move towards a local target location while avoiding obstacles
- **Input**: sensor data within a specific *look-ahead* plus target location
- **Output**: wheel motion commands
- No map and no memory of previous observations
- Must run very fast

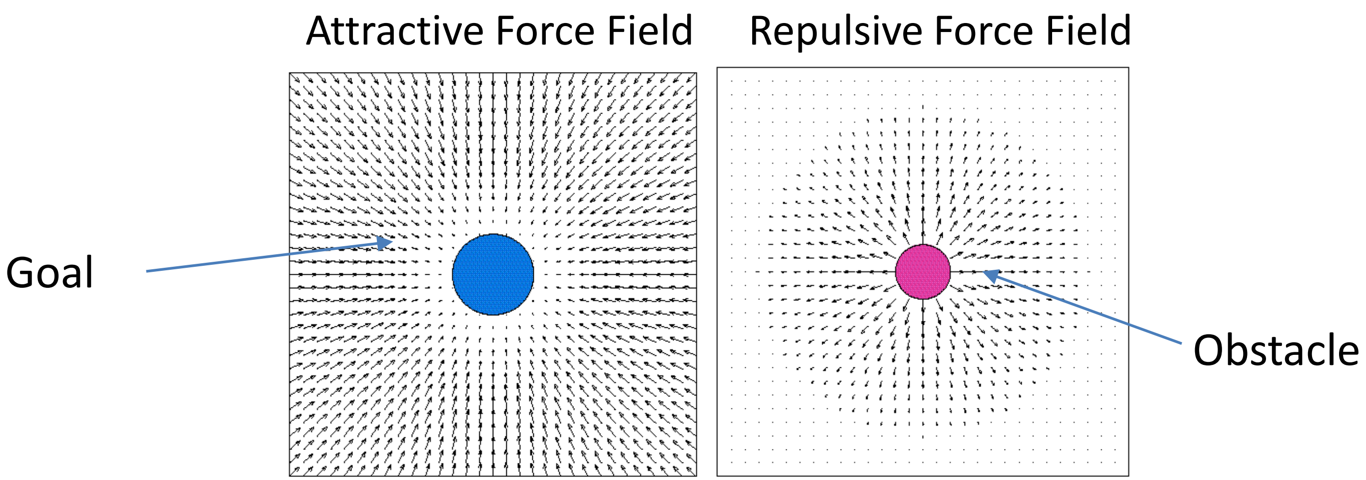


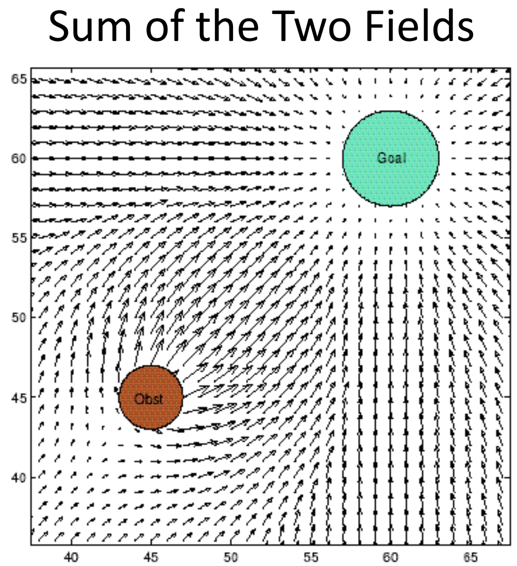


Potential fields

- Define a potential (energy) function over the free space with global **minimum at the goal** and a **maximum at obstacles**
- The robot moves to a lower energy configuration, similar to a ball rolling down a hill (gravitatory forces acting on it)
- To navigate, the robot applies a vector force proportional to the negated gradient of the potential field.

Forces generated by the potential field:





Potential fields

Algorithm:

1. Based on the observation, generates the potential energy function $U(p) = U_{att}(p) + U_{rep}(p)$ (must be differentiable)

   robot position

2. Force field $F(p)$ (there is a force vector in each 2D position)

Vector force proportional to minus the gradient of the potential field

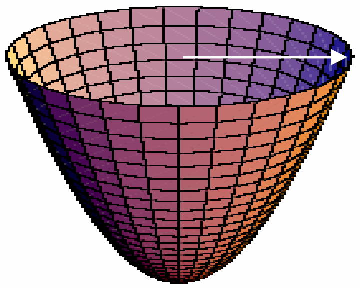
$$F(p) = -k\nabla U(p) = -k\nabla U_{att}(p) - k\nabla U_{rep}(p) = k\begin{bmatrix}\frac{\partial U}{\partial x}\\ \frac{\partial U}{\partial y}\end{bmatrix}$$

Gradient of the potential field

3. Set robot speed (v_x, v_y) proportional to the resulting force $F(p)$ generated by the field

Potential fields

Attractive Potential Field: Quadratic function representing the squared Euclidean distance to the goal d_{goal}



$$U_{att}(p) = \frac{1}{2} k_{att} d_{goal}^2(p) \qquad \text{with } d_{goal}^2(p) = \|p - p_{goal}\|^2$$

Robot position

Attractive force converges linearly towards $d_{goal}(p_{goal}) = 0$

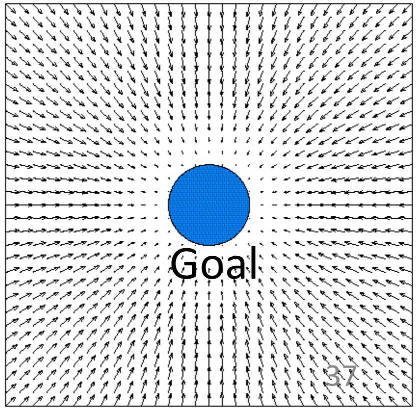
Gradient: $\left[\frac{\partial}{\partial x} \frac{\partial}{\partial y}\right]^T$

$$F_{att}(p) = -\nabla U_{att}(p)$$
$$= -k_{att} d_{goal}(p) \nabla d_{goal}(p)$$
$$= -k_{att} \underbrace{(p - p_{goal})}_{\text{vector to the goal}}$$

$$\nabla d_{goal} = \frac{p - p_{goal}}{d_{goal}}$$

Unit vector

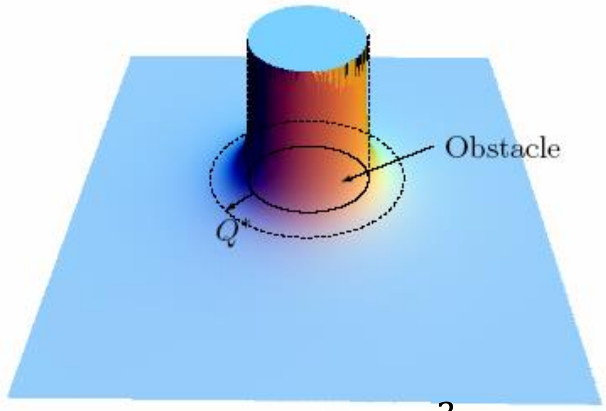
Attractive Force Field



Potential fields

Repulsive Potential Field

- Generate a barrier around obstacles
  - tends to infinity as p gets closer to the object (inversely proportional to $d(p)$)
  - not influence if very far from the obstacle ($d(p) > d_0$)

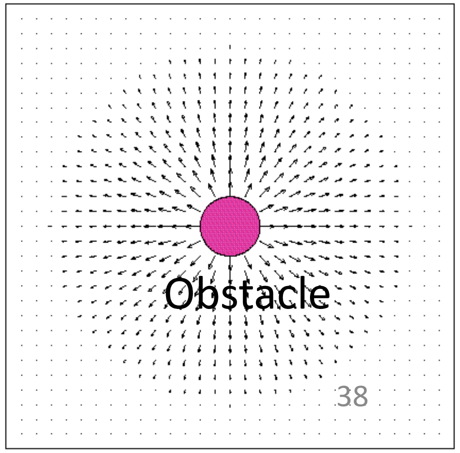


$d^2(p) = \left\| p - p_{obj} \right\|^2$

$d(p)$: distance to the object (e.g. each range from the laser scanner)

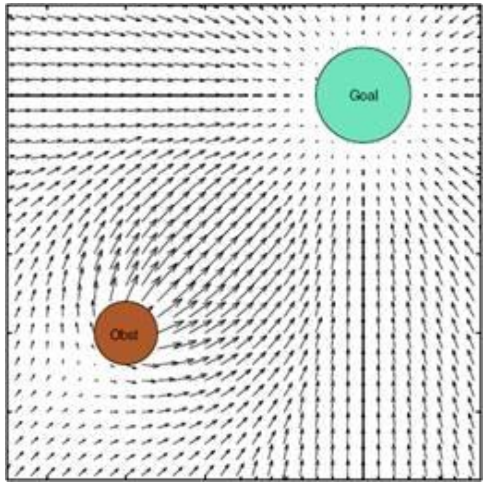
$$U_{rep}(p) = \begin{cases} \frac{1}{2} k_{rep} \left(\frac{1}{d(p)} - \frac{1}{d_0} \right)^2 & if\ d(p) \le d_0 \\ 0 & if\ d(p) > d_0 \end{cases}$$

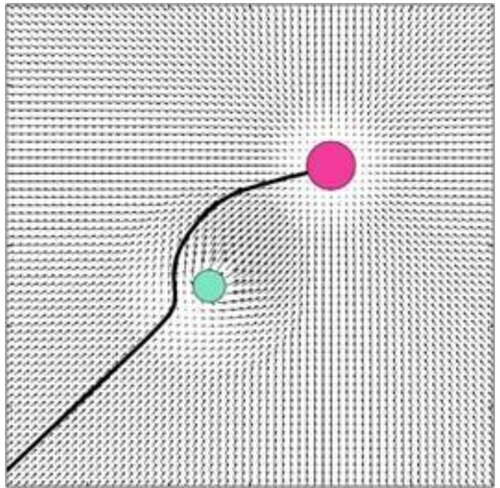
$$F_{rep}(p) = -\nabla U_{rep}(p) = \begin{cases} k_{rep} \left(\frac{1}{d(p)} - \frac{1}{d_0} \right) \frac{1}{d^2(p)} \frac{p - p_{obj}}{d(p)} & if\ d(p) \le d_0 \\ 0 & if\ d(p) > d_0 \end{cases}$$



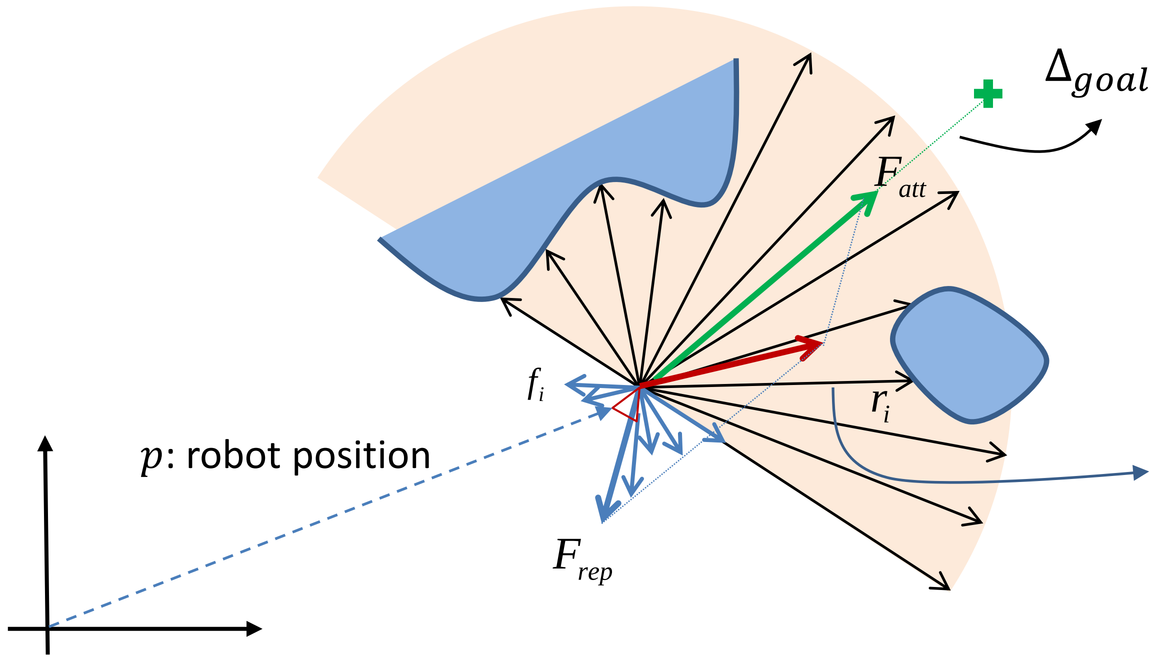
Potential fields for a laser scan

- Input: distance to obstacles (laser scan) and target
- Output: Velocity to the wheels (proportional to the force)
- Problems:
  - represents the robot as a free-flying point and does not take the vehicle's shape and kinematics into consideration
  - can be trapped into local minima





Potential fields for a laser scan



$\Delta_{goal} = p - p_{goal}$ (Goal vector)

$\Delta_i = p - p_i$

$\|\Delta_i\| = r_i$

Repulsive force

Unit vector: force direction

$$f_i = \begin{cases} \left(\frac{1}{r_i} - \frac{1}{r_{\max}}\right)\frac{1}{r_i^2}\frac{\Delta_i}{r_i} & \text{if } r_i < r_{\max} \\ 0 & \text{if } r_i \geq r_{\max} \end{cases}$$

$$F_{rep} = k_{rep} \sum_i f_i$$

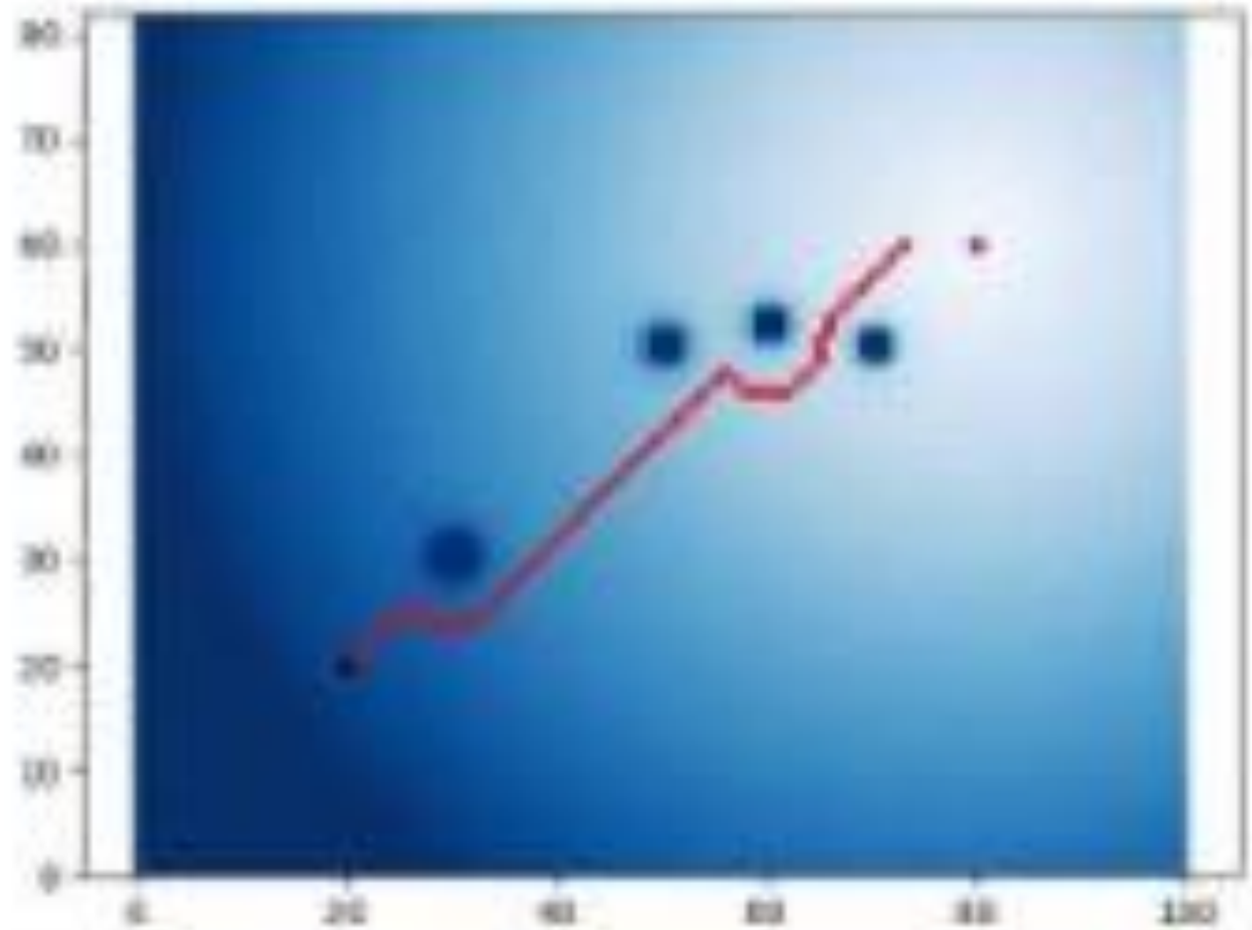
Attractive force

$$\boxed{F_{att} = -k_{att} \Delta_{goal}}$$

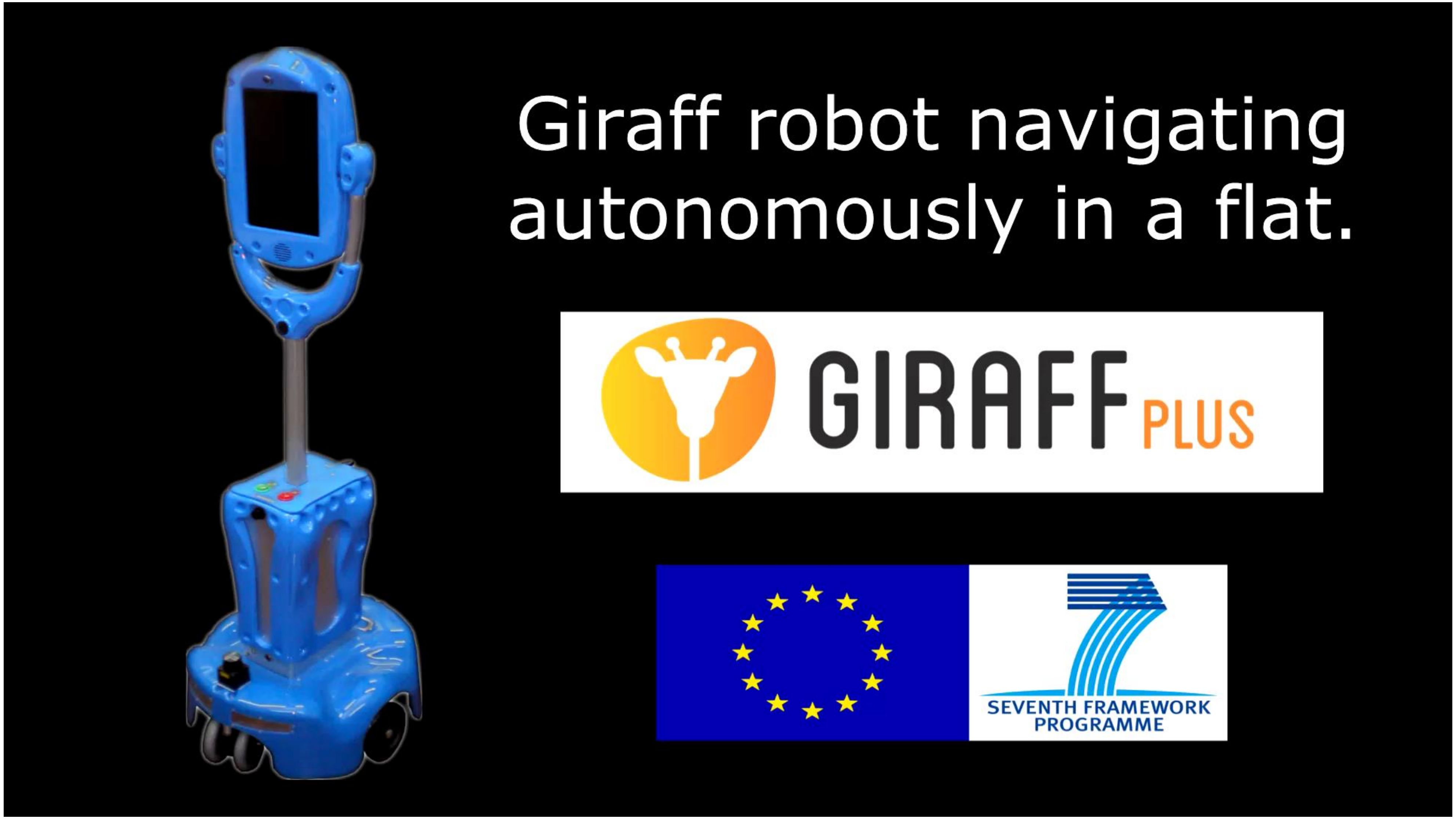
Resulting total force

$$F_{total} = F_{att} + F_{rep}$$

Potential fields



Roald map + Reactive navigation



The end