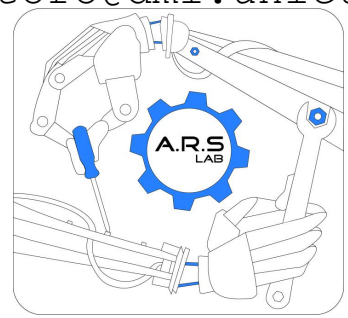
Navigation, Path Planning and Obstacle Avoidance

Corrado Santoro

ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
santoro@dmi.unict.it



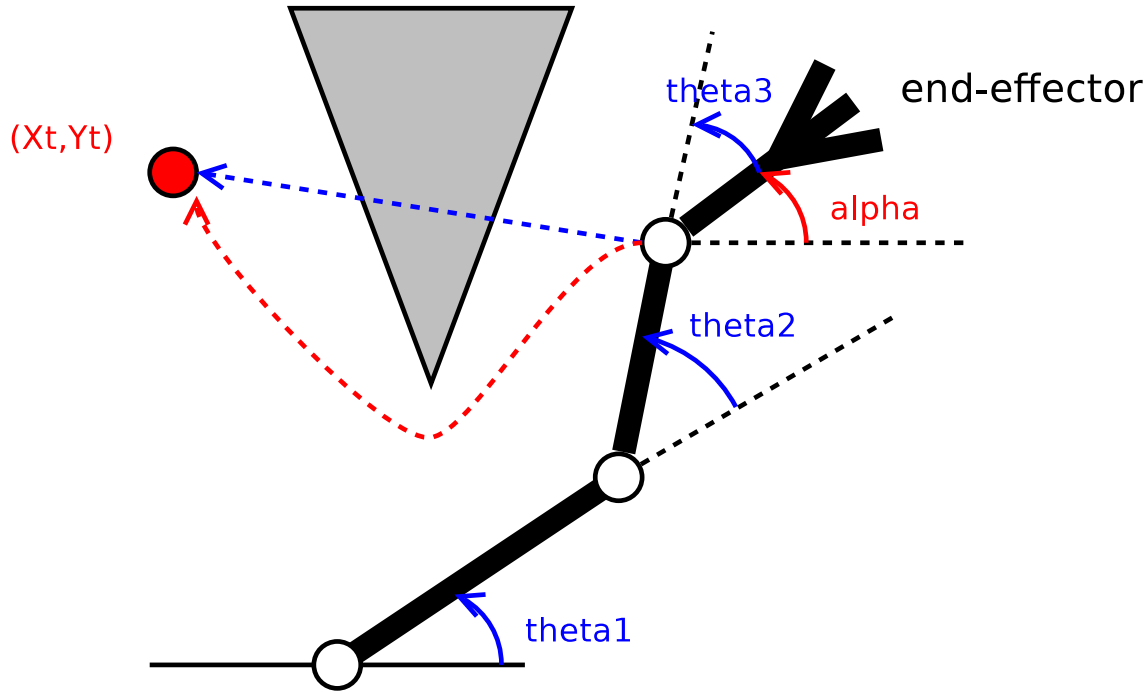
Robotic Systems

Kinematics, Position, Control

- Any robotic system is characterised by its own **kinematic model**
- The model, that is used to perform the **position control**, considers the use of
  - **controllable variables**, e.g. the speed of the wheels, etc.
  - **observable variables**, e.g. positions of the joints, etc.
  - **working variables**, the variables of the main reference system
- The model does not take into account the **real composition of the environment** and of the physical constraints that impede certain robot movements

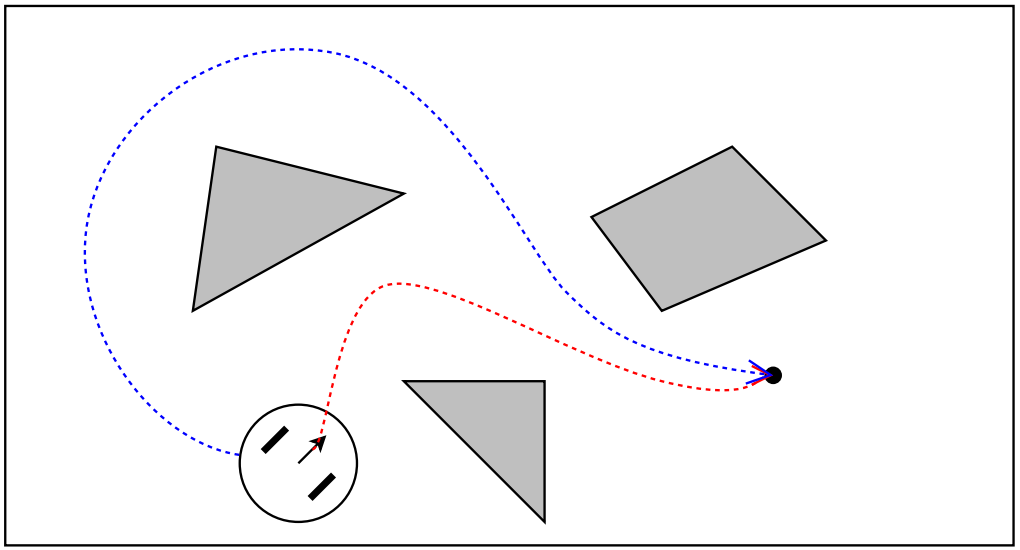
Environment and Constraints

- In the planar manipulator in figure, position (X_t, Y_t) is not directly reachable with a straight path (blue path) due to the presence of the obstacle
- The **red path** would allow the robot to overcome the obstacle and reach the position
- However, the red path, that is not "straight", must be properly **planned**



Environment and Constraints

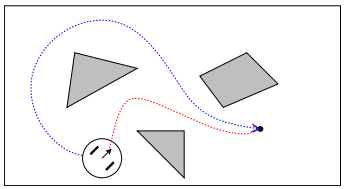
- In the mobile robot in figure, the destination point is not directly reachable with a straight path
- However, there are various paths that make the robot reach the target position
- Making the proper path is the aim of the **path planning** algorithms



Path Planning

Path Planning: what is needed

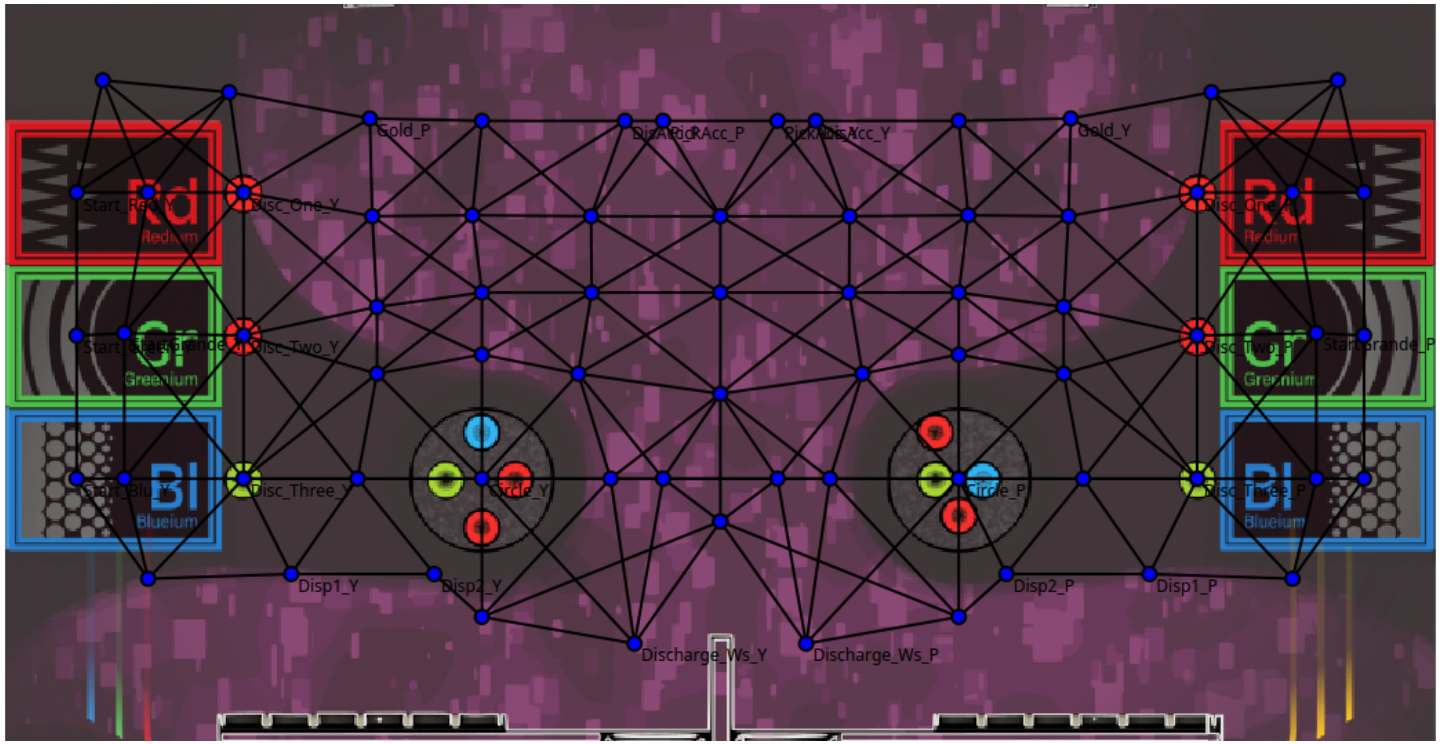
- **Environment model**: the environment in which the robot lives, with all its geometric characteristics and constraints must be modeled in the software; any planned path must be **inside the environment** and **it must not intersect** with the constraints
- **Robot model**: the robot cannot be considered as “point object”, but its physical nature with its dimenstions must be taken into account in the planning process; this must be done avoiding any **intersection** between the robot and one of the constraints
- **Choice of the trajectory**: a proper methodology must be used to make the algorithm derive the possible trajectories and select one of them on the basis of proper parameters, such as **minimal distance** or other



Fixed Graph

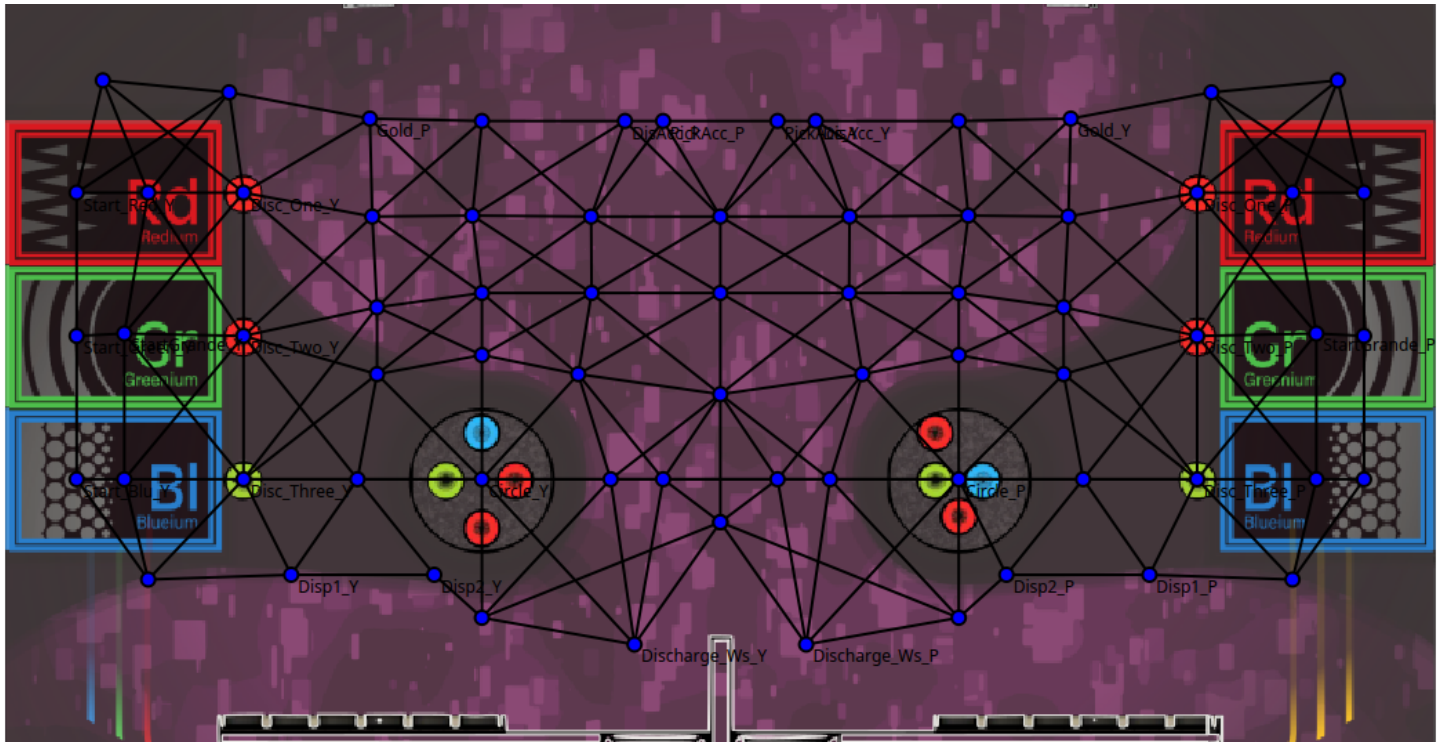
Fixed Graph (1)

- This algorithm consists in defining **a priori** a **graph** of possible paths
- The starting (START) and destination (END) points of the robot must always be represented by a **vertex** of the graph



Fixed Graph (2)

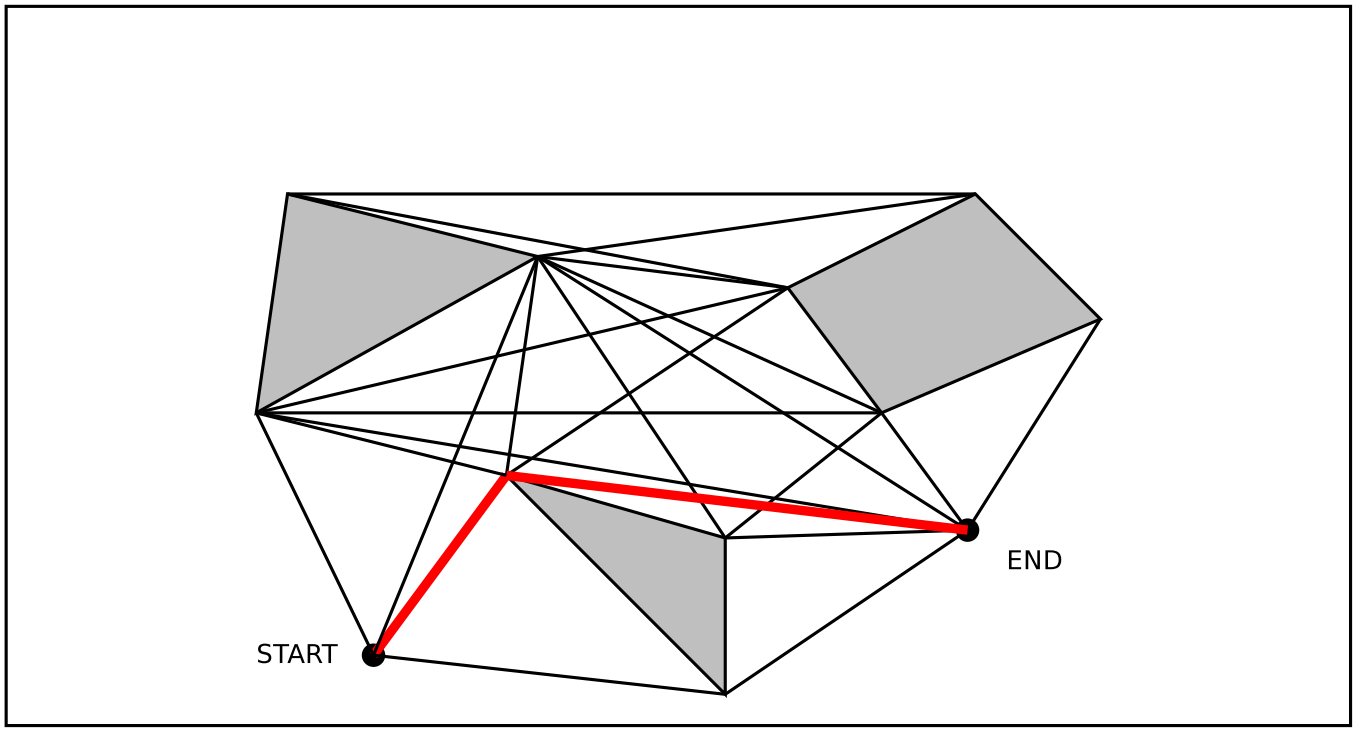
- A path from START to END is determined, on the graph, by means of the **Dijkstra algorithm** which finds the path with minimal distance
- The path computed is a list of **vertices** that represent the **line** to follow
- The path is then provided to the motion control module that implements the proper position control algorithm



Visibility Graph

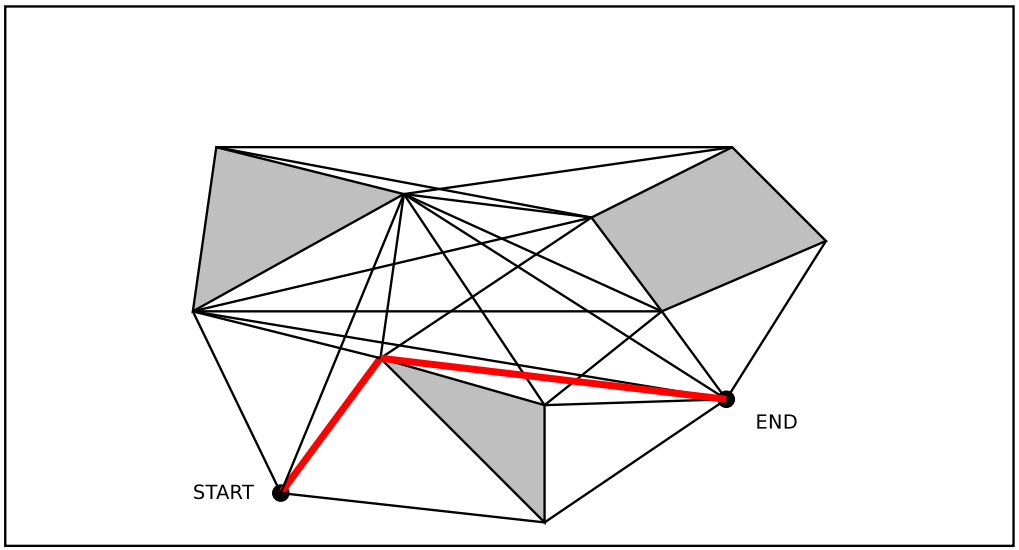
Visibility Graph (1)

- This method consists in modelling a **graph** that represents the **visibility** of all the vertices of the geometries of the constraints present in the environment
- Each **geometric vertex** (included START and END) is a **vertex of the graph**
- Two vertices are connected with an “edge” if they are **visible**
- In the resulting graph the Dijkstra algorithm is then run



Visibility Graph (2)

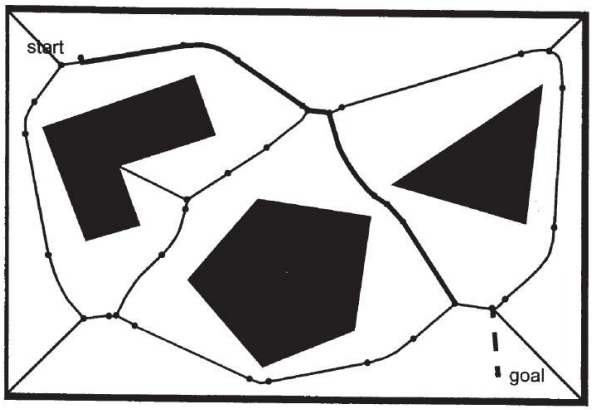
- **Advantages:** it is simple to implement and efficient
- **Disadvantages:**
  - Geometric constraints must be “enlarged” to take into account robot dimensions
  - The resulting path is in general “too near” to constraints (so it is not *safe*)
  - If the environment is dense of constraints, the complexity of the graph increases



Voronoi Diagrams

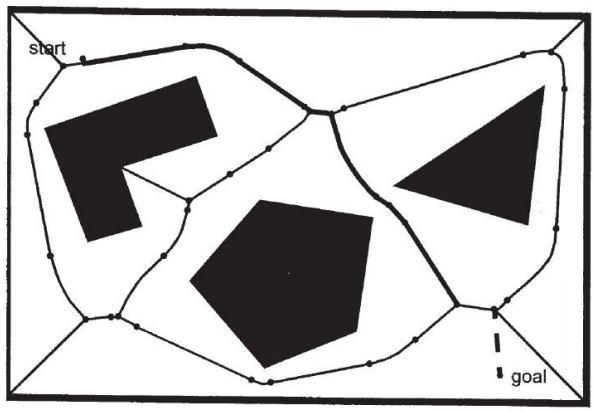
Voronoi Diagrams (1)

- **Voronoi diagrams** are defined as the geometric locus of the points that have maximal distance from obstacles
- They are build using a “tridimensional” approach:
  - For each point of the navigation space, the distance from the nearest constraint is determined
  - This distance is represented as a **height**
  - Points that have the same distance from one or more obstacles will form a **peak**
  - All peaks are linked together: the resulting geometries are the possible “roads” that the robot can follow



Voronoi Diagrams (2)

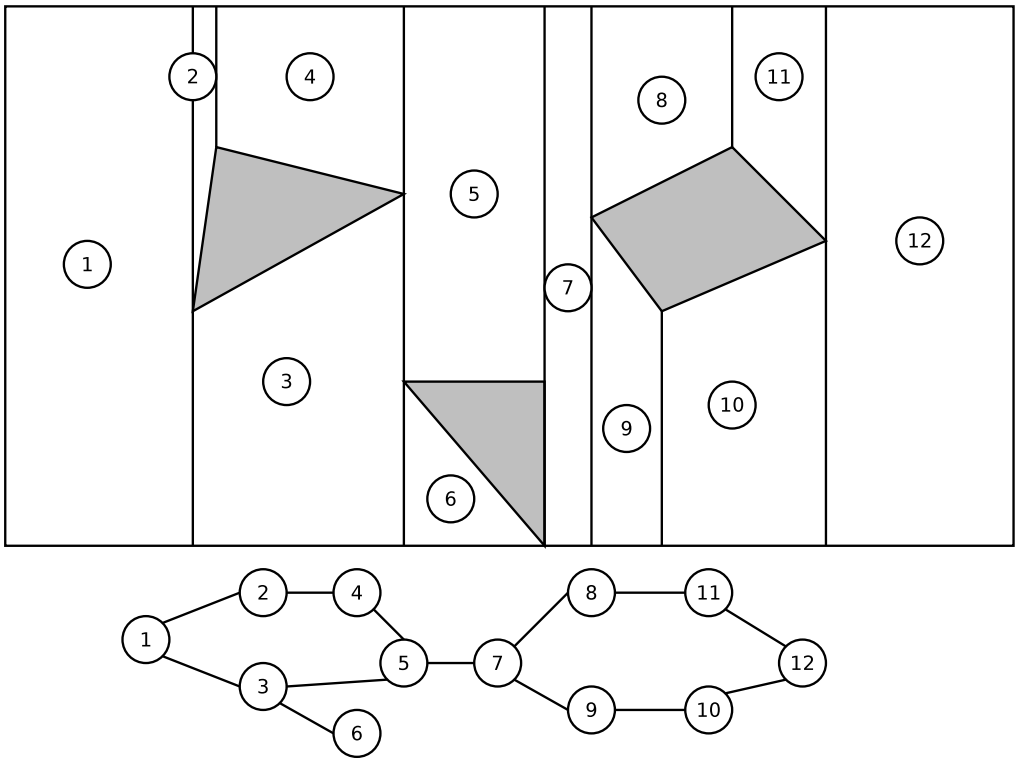
- **Advantages:**
  - Possible paths are limited and are more efficient than those of the visibility graph
  - The navigation is “safe” because it follows always points that are far from obstacles
- **Disadvantages:**
  - The construction of the Voronoi diagram is a hard task (from the computational point of view), but it is only done in the initial phase (unless the environment is dynamic)



Cell Decomposition

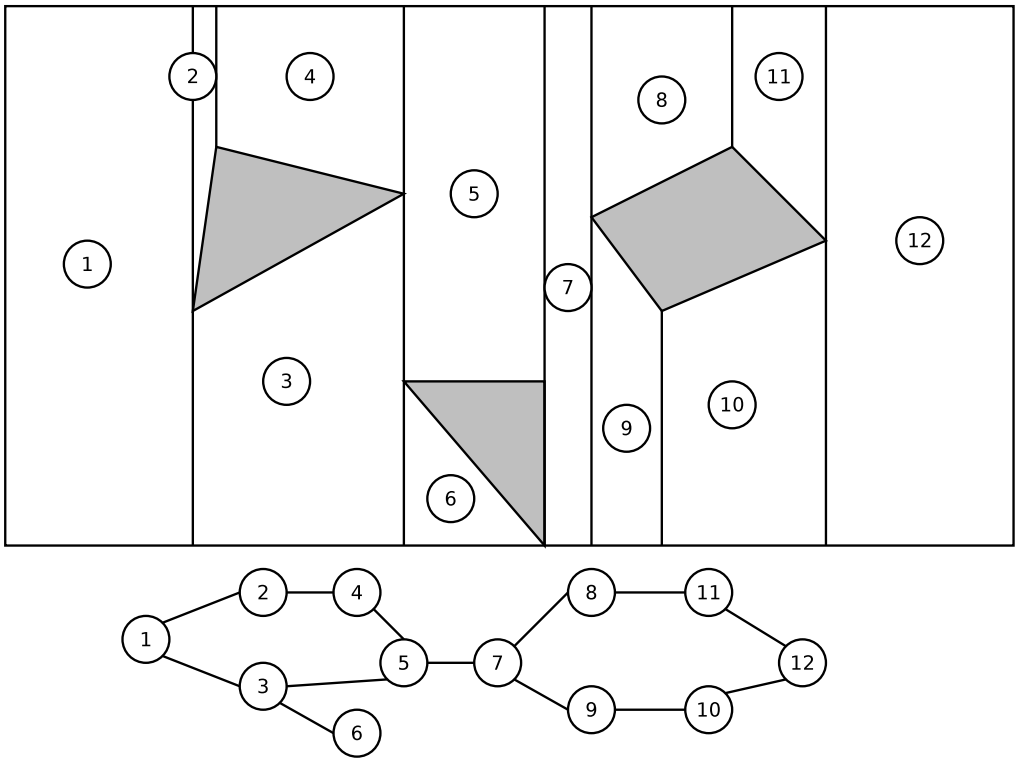
Cell Decomposition

- The whole navigation region is subdivided into **cells**
- For each cell a **navigation point** is identified
- A **graph** is built, cells are the vertices and the edges represent the fact that two cells are neighbours
- The Dijkstra algorithm is run the determine the minimum distance path



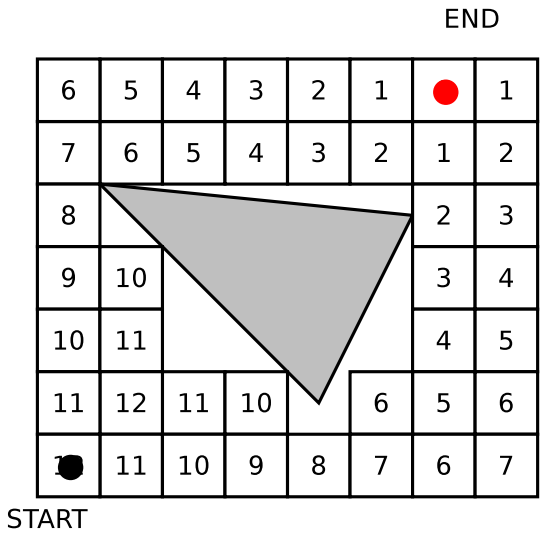
Cell Decomposition

- The algorithm is simple to implement and quite efficient
- The characteristics depend on the way in which the cells are subdivided:
  -
  - More cells $\Rightarrow$ more efficient path
  - Less cells $\Rightarrow$ more fast algorithm



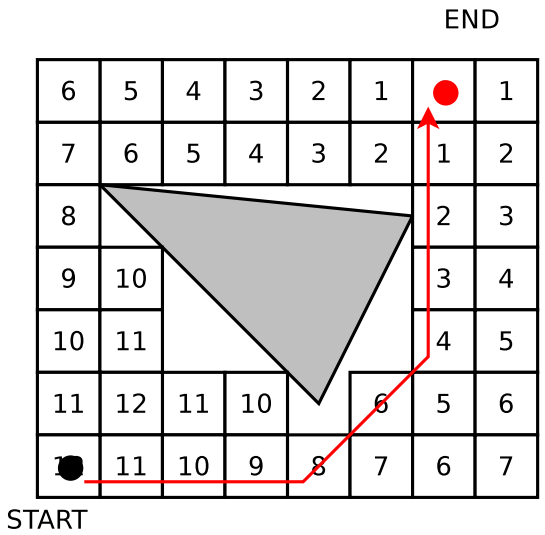
Fixed Cell Decomposition: NF1

- The whole region is subdivided in **fixed cells**
- Each cell is characterised by a **numeric mark**
- For each cell, the **navigation point** is its **center**
- We start from the END point that is marked with 0
- Each adjacent cell is marked with a value **incremented by 1**
- The algorithm proceeds until **all cells** are marked



Fixed Cell Decomposition: NF1

- To determine the path, the algorithm begins from START and selects the **adjacent cell with the minor mark value**
- The algorithm continues until the END point is reached



Potential Field

Potential Field (1)

- The whole navigation space is considered pervaded by a **potential field** made by:
  - **Attractive forces** towards the target
  - **Repulsive forces** that step away from constraints
- The **potential** of a point $q = (x, y)$ is defined as $U(q) = U_{att}(q) + U_{rep}(q)$
- The **motion vector** of a point $q = (x, y)$ is thus the **force** generated from potential gradient $F(q) = -\nabla U_{att}(q) - \nabla U_{rep}(q)$
- where the symbol ∇ is the gradient operator:

$$\nabla U(q) = \begin{bmatrix} \frac{\partial U}{\partial x} \\ \frac{\partial U}{\partial y} \end{bmatrix}$$

Potential Field (2)

- If q_{goal} is the **target point**, the attractive force can be modelled as:

$$F_{att}(q) = -k_{att}(q - q_{goal})$$

where k_{att} is a scalar that acts as a "weight"

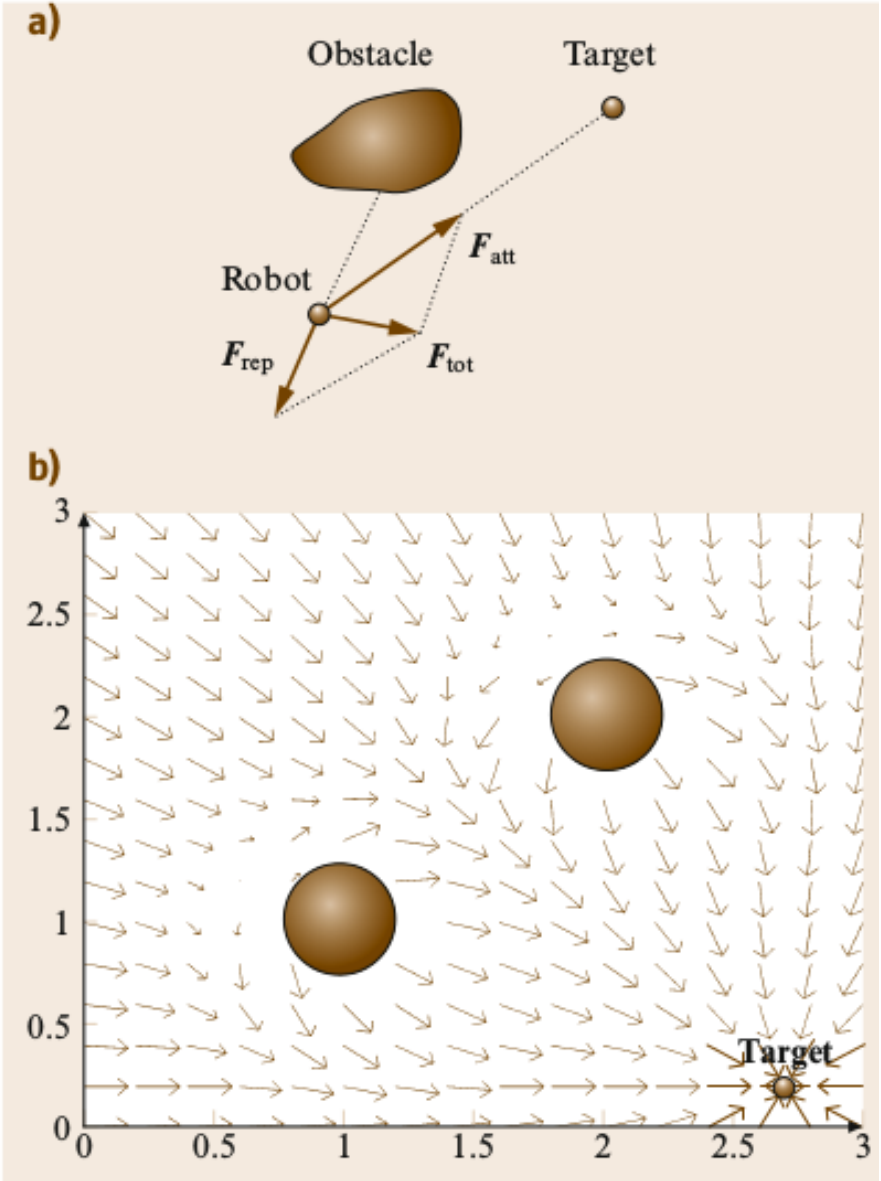
- repulsive force can be expressed as:

$$F_{rep}(q) = \begin{cases} k_{rep}\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)\frac{1}{\rho^2(q)}\frac{q - q_{obstacle}}{\rho(q)} & se \ \rho(q) \leq \rho_0 \\ 0 & se \ \rho(q) > \rho_0 \end{cases}$$

where

- $q_{obstacle}$ is the point of the obstacle with minimum distance
- $\rho(q)$ is the distance from an obstacle
- ρ_0 is distance threshold
- k_{rep} is a weight factor

Potential Field (3)



Potential Field (4)

- The approach consists in imposing to the robot the speeds v_x e v_y proportional to the components of the resulting force
- The example is a ball that is gradually guided towards the target point “coming down a hill” between the obstacles, however ...
- The algorithm is strongly dependent on the weight assigned to attractive and repulsive force k_{att}, k_{rep}
- The trend of the gradient is not linear; as soon as we approach an obstacle the contribution of the repulsive force is really strong, and this may cause sudden speed changes
- When there are obstacle near to one other, the algorithm can trigger undesirable oscillations
- The approach does not take into account the maneuverability of the robot

Obstacle Detection & Avoidance

Obstacle Detection & Avoidance

- The path planning implies to model the **environment characteristics** and to determine how to navigate in that environment
- The **Obstacle Avoidance** instead implies that, given a certain trajectory, an obstacle is identified and avoided with the object of **achieveing the target point in any case**
- The Obstacle Avoidance implies that, during the path, an unforeseen obstacle can be found; in this case we must:
  - **identify the presence of the obstacle**, an operation that is performed through proper sensors
  - **avoid the obstacle**, an operation that implies the use of proper algorithms

Obstacle Detection

Sensors for Obstacle Detection (1)

- Distance Sensors
  - Optical (light)
  - Acoustic (sound)
- Cameras (with proper computer vision algorithms, given that the obstacle has precise characteristics)
- Depth-cameras (stereoscopic vision)

Sensors for Obstacle Detection (2)

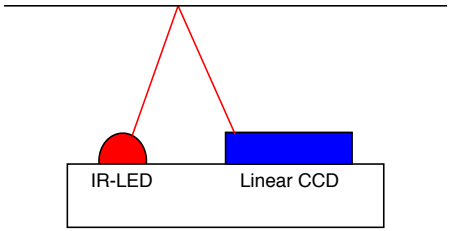
- **Acoustic Sensors**
  - They are based on the computation of the **time of flight** of an ultrasound acoustic signal
  - They give the **distance**, with the resolution of **centimeters**
  - They use proper “ultrasound capsules”
  - They have a wide **action cone**
  - They can be subject to **mutual interference** when, in the same environment, there are more sensors of the same type and directed towards one another

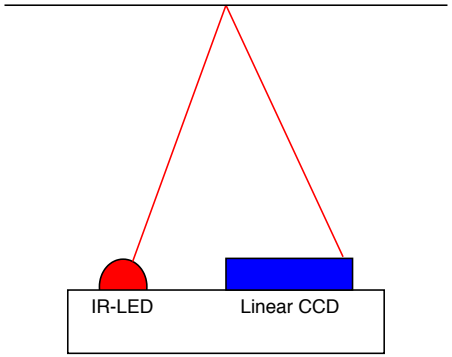
Sensors for Obstacle Detection (3)

- **Threshold Optical Sensors**
  - They are based on a light ray that is reflected by the obstacle
  - They determine the **presence of the obstacle** inside a certain tuning threshold
  - They are ON/OFF sensors
  - They use infrared or visibile (red) light
  - They have an **narrow action cone**
  - They are not subject to interference

Sensors for Obstacle Detection (4)

- **Optical Triangulation Sensors**
  - They uses a light ray (visible or infrared) whose reflection is detected by a linear CCD sensor
  - They measure the distance by identifying the triangle formed by the emitted ray, reflected ray and sensor
  - They have a **narrow action cone**
  - They are not subject to interference



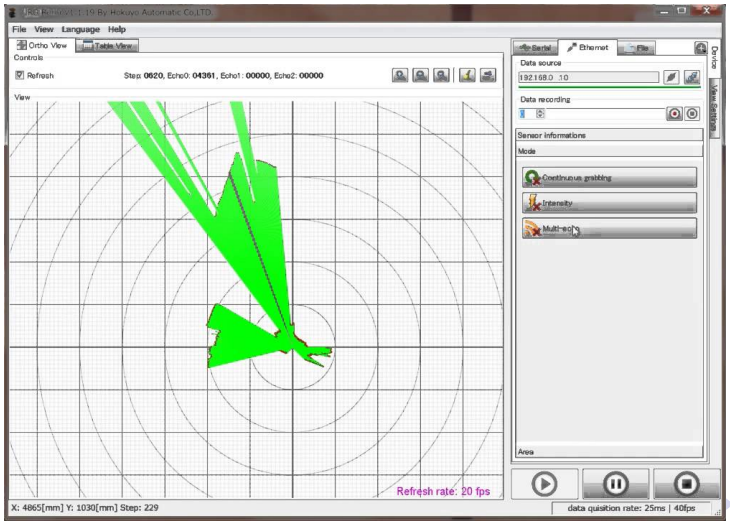


Sensors for Obstacle Detection (5)

- **Optical Sensors "time-of-flight"**
  - They use a infrared laser ray and determine the "time of flight" of the reflected ray
  - The revelator is made of phododiode called SPAD (Single Photon Avalance Diode)
  - They measure the distance with the resolution of the millimeter
  - They have a **narrow action cone**
  - They are sensible to light conditions
  - They are widely used in the smartphones

Sensors for Obstacle Detection (6)

- **LIDAR**
  - A **LIDAR** (**Laser Imaging Detection and Ranging**) is a sensor that, by means of the rotation of a laser and a “ToF” sensor, performs a 2D mapping of the environment
  - The sensor is made of a ToF mounted on a platform rotating at the speed of about 10 revolutions per second
  - The sensor measure the distance with the resolution of the millimeter
  - The sensor outputs a point cloud in **polar coordinates** centered in the sensor



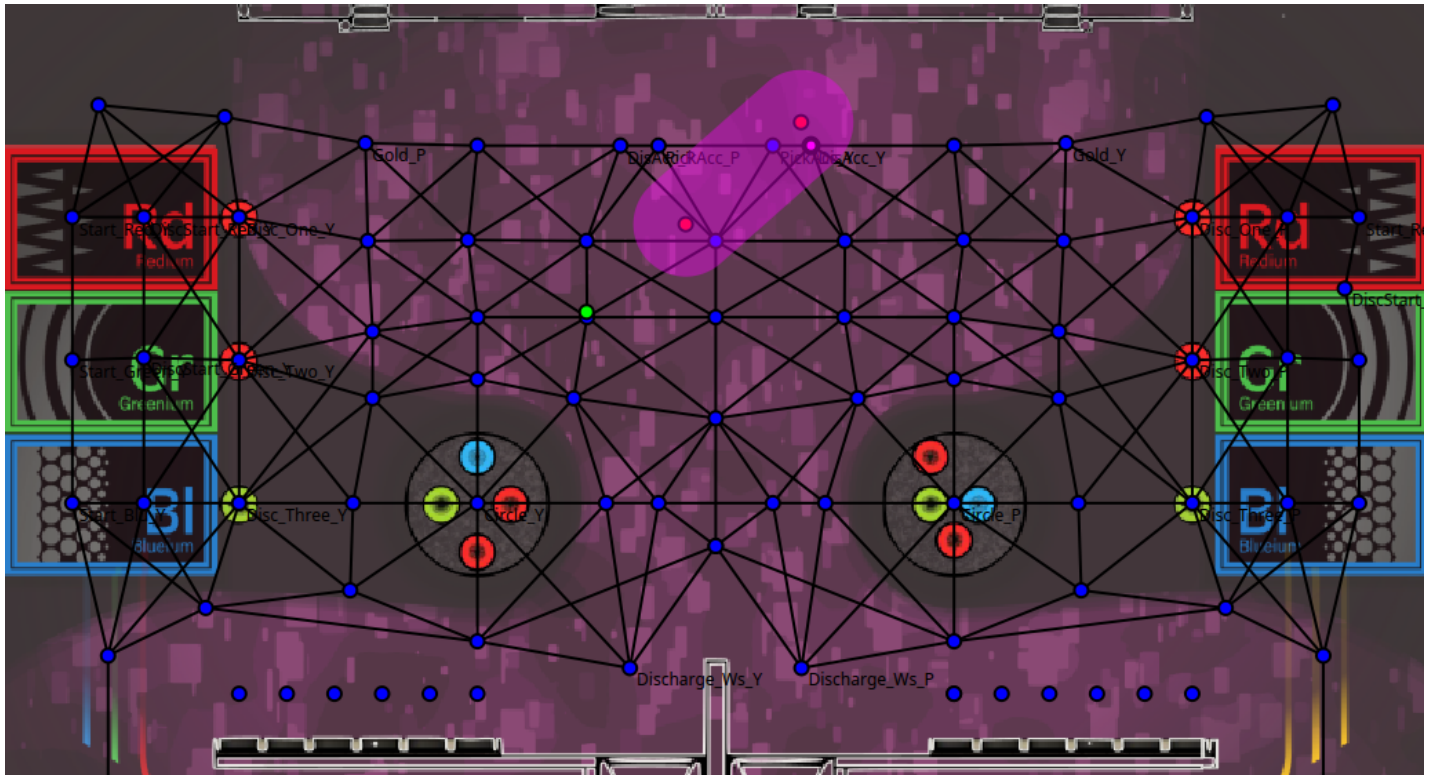
Obstacle Avoidance

Obstacle Avoidance

- When the **obstacle dimensions** are **known**, one of the previous path planning algorithms can be directly employed; in this case:
  - the obstacle must be inserted at runtime in the environment model
  - the path planning algorithm is then run
  - if the obstacle is **mobile**, a **decay factor** must me used to "remove" the obstacle from the point after a certain time

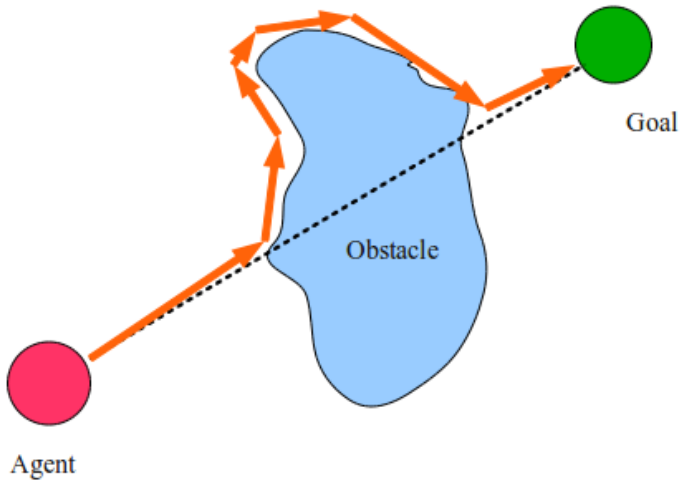
Fixed Graph and Obstacle Avoidance

- On a **fixed graph**, the **zone** in which the obstacle is supposed to be found is determined
- Edges and vertices that intersect that zone are temporarily removed from the graph
- The resulting graph is then used to plan a new path



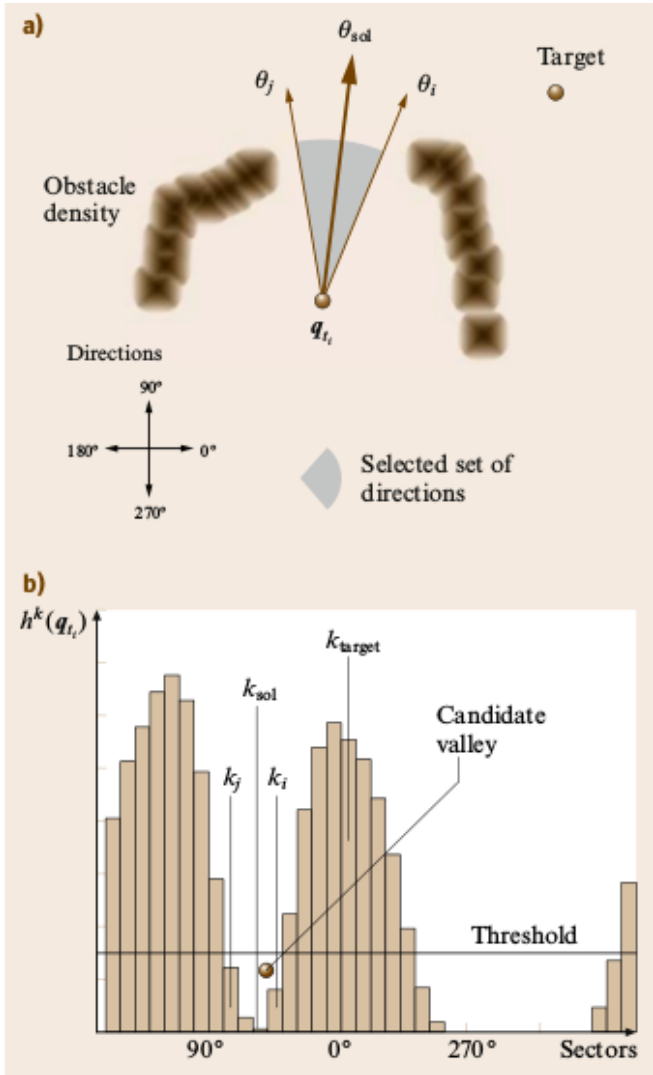
Bug Algorithm

- If the geometry of the obstacle is not know, one of most widely used approach is the **bug algorithm**
- It is based on the presence of 360 degree sensor (sonar, LIDAR, etc.)
- It consists in circumnavigation the obstacle in the direction of the target point, until the obstacle is overcome and the target is visible

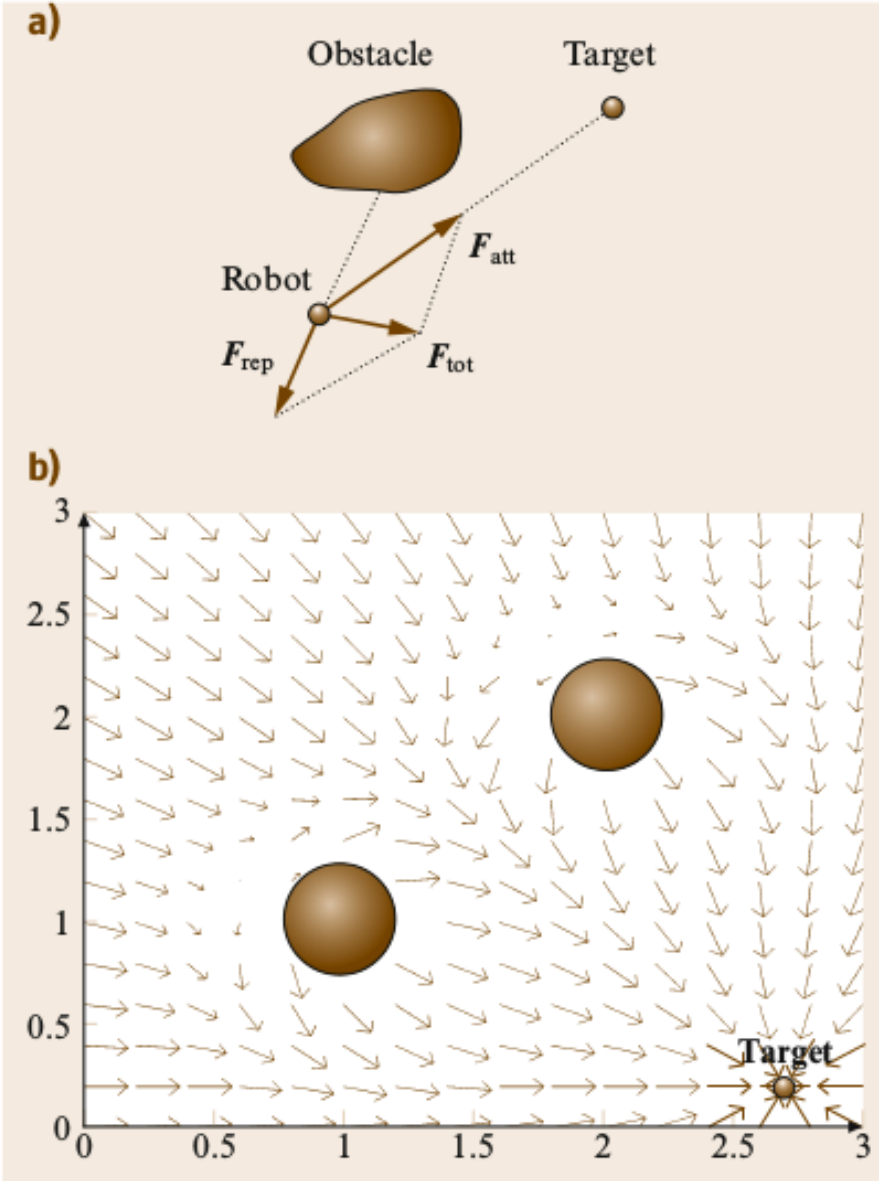


Vector Field Histogram

- When the geometry of the obstacle is know, the **Vector Field Histogram** is an approach that can be used
- It is based on two steps:
  1. Determination of the polar diagram of the **obstacle density** (histogram)
  2. Histogram analysis in order to determine the motion vector
- The **valleys** in the histogram are identified, they correspond to the zones that can be traversed
- The “best valley” to reach the target is selected
- In the selected valley, the motior vector is computed on the basis of the robot dimensions

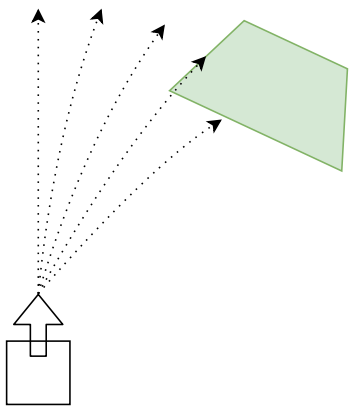


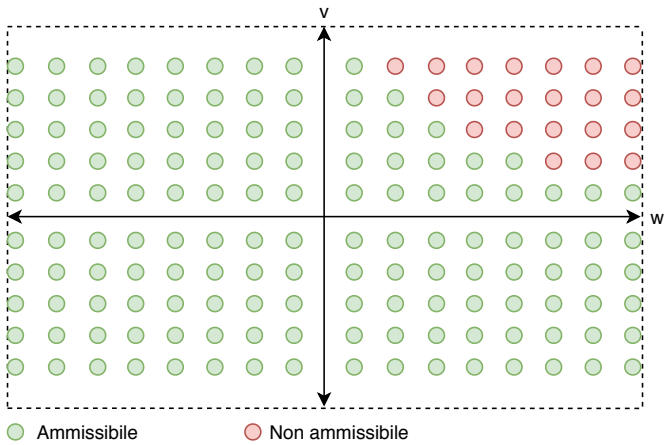
Potential Field



Dynamic Window Approach

- The Dynamic Window Approach (DWA) consists in treating the problem of obstacle avoidance in the space (v, ω) of the robot
- Given a robot pose, each tuple (v_i, ω_i) generates a circular trajectory that can meet or not the obstacle
- All the tuples (v_i, ω_i) are reported in a XY chart
- All the tuple that lead to a collision are marked as **non admissible**
- Among all admissible tuple, it is selected the one that:
  - leads to the target by using the highest speed
  - is "reachable", given the acceleration characteristics of the robot
  - drives the robot sufficiently far from the obstacle

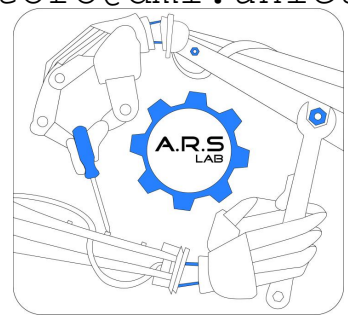




Navigation, Path Planning and Obstacle Avoidance

Corrado Santoro

ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
santoro@dmi.unict.it



Robotic Systems