Introduction to Mobile Robotics

Riccardo Falconi

Dipartimento di Elettronica, Informatica e Sistemistica (DEIS)
Universita di Bologna

email: riccardo.falconi@unibo.it
Starting from the ’70s, early industrial robots
Used to replace men in case of hazardous/repetitivity/easy tasks

Typically used for tasks to be executed *in loco*
Workspaces were modeled considering *robots* capabilities and constraints
From Industrial Robotics to Mobile Robotics

Robotics for manipulation
Imitates the human arms (and hands) functionalities

Mobile Robotics
Himitates the animals *locomotion* principles

Definition
A mobile robot is an automatic machine able to move into the surrounding environment.
Typical Applications

**Indoor applications**

- Cleaning
- Service robotics (museums, shops, etc.)
- Surveillance
- Stockings in warehouses
- ...

**Outdoor applications**

- Military applications
- Demining
- Spatial and underwater exploration
- Civil protection and fire surveillance
- Automatic agricultural
- ...

Applicazioni *Outdoor*
How it all started...

- Mobile robots starting from 40s
- *Rockets V1 - V2* (inertial autopilots)

- Drive-by-wire
- Electric motors

- Remotely controlled tank
- 500 m far
Examples

AGV automatic warehouse

- Moves pallets from the loading bay to the depot area and vice versa
- Inside a warehouse there could be more than 30 AGVs

Automatic Guided Vehicle (AGV) Electric 80
Examples

Underwater robots
- Recover wreckage (Titanic)
- Pipes inspections
- Water pollution monitoring

Space applications
- Realized starting from 1965
- Highly autonomous
- Able to work without a supervisor
Examples

Agricultural robots (≈2015)

Mobile robots designed for agricultural purposes:

- structured environments (ad hoc orchards)
- large fields cultivated with cereals
Examples

Hazardous robots

Robot Pioneer

- Developed by Stanford University
- Chosen to explore Chernobyl nuclear plant
- First modular robot allowing to mount:
  - a robotic arm + a gripper
  - a stereo camera
Locomotion Principles

- Natural solutions are difficult to be replied from an engineering point of view
- The rolling is the most efficient solution, but it is not present in nature
- Most of the robotic systems move using wheels or tracks
- The walking is the natural best approximation of the rolling

Walking

Crawling

Running

Jumping
Locomotion Principles

Human walking
- It approximates rolling of a regular polygon
- Side of the polygon equal to the step of the person
- The shortest the step is, the better the rolling is approximated

Walking robots:
- can adapt to a variable environment
- move slow on flat surfaces
- overactuated

Robot with wheels/tracks:
- move faster on flat surfaces
- cannot adapt to a wide range of environmental conditions
Robot Modeling
Wheeled Mobile Robots - WMR

- The wheel is the most suitable solution adopted for many applications
- Three wheels are enough to ensure vehicle stability
- How many wheels? What kind of wheels?

**Fixed wheel**

**Centered orientable wheel**

**Castor wheel**

**Swedish wheel**
Space of the configurations

- The dimension of the space of the configurations is equal to the minimum number of parameters needed to localize the robot configuration.
- It depends on the structure of the considered robot.

Example: Unicycle

- $q = [x_r, y_r, \theta_r]^T \in \mathbb{R}^3$

Example: Bicycle

- $q = [x_r, y_r, \theta_r, \phi_r]^T \in \mathbb{R}^4$
**Anomalous constraints**

**Hypothesis**

- Each wheel rolls without sliding
- Each wheel introduces a nonholonomic constraint in the system (it does not allow the robot to move perpendicularly to the rolling direction)
- Wheels reduce the instantaneous robot mobility without reducing the maneuvering capabilities (e.g. parallel parking)

\[ v_t = \dot{\psi} r \]

Without constraint:

\[
\begin{align*}
\dot{x} &= v_t \cos \theta + v_n \cos \left( \theta + \frac{\pi}{2} \right) \\
\dot{y} &= v_t \sin \theta + v_n \sin \left( \theta + \frac{\pi}{2} \right)
\end{align*}
\]

As long as the sliding in the normal direction is not allowed \( v_n = 0 \)

\[
\begin{align*}
\dot{x} &= v_t \cos \theta \\
\dot{y} &= v_t \sin \theta
\end{align*}
\]

\[ \tan \partial \theta = \frac{\partial y}{\partial x} \iff \left[ \begin{array}{c} \dot{x} \sin \theta - \dot{y} \cos \theta = 0 \\
\text{Mobility constraint} \end{array} \right] \]
Pfaffian constraint

- For each wheel the constraint can be written in a vectorial form as: \( a(q) \dot{q} = 0 \)
- With \( N \) wheels, constraints can be expressed in matrix form as: \( A(q) \dot{q} = 0 \)
- A constraint that can be written as \( A(q) \dot{q} = 0 \) is called Pfaffian constraint

Nonholonomic constraint

- It is not fully integrable
- It cannot be written in the space of the configurations
- It doesn’t limit the mobility but only the instantaneous mobility of the robot

Allowed velocities

They can be generated by using a matrix \( G(q) \) such that:

\[
\text{Im}(G(q)) = \text{Ker}(A(q)), \quad \forall q \in \mathbb{C} \\
\mathbb{C} = \mathbb{R}^N
\]


Unicycle model

- **Constraint:**
  \[ \dot{x} \sin \theta - \dot{y} \cos \theta = [\sin \theta, -\cos \theta, 0] \dot{q} = 0 \]

- **Pfaffian form:**
  \[ A(q) \dot{q} = 0 \]
  \[ A(q) = [\sin \theta, -\cos \theta, 0] \]
  \[ q = [x, y, \theta]^T \]

- **Ker(A(q))**
  \[ \text{Ker}(A(q)) = \text{span} \left( \begin{bmatrix} \cos \theta \\ \sin \theta \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right) = \text{Im}(G(q)) \]

Unicycle kinematic model

- \[ \dot{q} = \begin{bmatrix} \cos \theta \\ \sin \theta \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega = \begin{bmatrix} \cos \theta & 0 \\ \sin \theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \]

- **v**: wheel linear velocity
- **\omega**: angular velocity
Unicycle model

- The real unicycle has statics stability problems
- Structures kinematically equivalent are used

**Synchronized Drive model**

- Parallel orientable and coupled wheels
- Inputs \([v, \omega]^T\)
- \([x, y, \theta]^T\) position and orientation.

**Differential Drive model**

- Two wheels independently driven
- A third wheel to ensure stability
- \([x, y, \theta]^T\) position of the center of the robot + its orientation.
Differential Drive unicycle model

- Definite
  \[
  \begin{align*}
  \omega_R &= \text{Right wheel velocity} \\
  \omega_L &= \text{Left wheel velocity} \\
  d &= \text{wheel axle} \\
  r &= \text{wheel radius}
  \end{align*}
  \]

Inputs \( \omega_R, \omega_L \)

\[
\begin{bmatrix}
  v \\
  \omega
\end{bmatrix} = \begin{bmatrix}
  \frac{r}{d} & \frac{r}{d} \\
  -\frac{r}{d} & -\frac{r}{d}
\end{bmatrix} \begin{bmatrix}
  \omega_R \\
  \omega_L
\end{bmatrix}
\]

State space model

\[
\dot{q} = \begin{bmatrix}
  \dot{x} \\
  \dot{y} \\
  \dot{\theta}
\end{bmatrix} = \begin{bmatrix}
  \cos \theta & 0 \\
  \sin \theta & 0 \\
  0 & 1
\end{bmatrix} \begin{bmatrix}
  \frac{r}{d} & \frac{r}{d} \\
  -\frac{r}{d} & -\frac{r}{d}
\end{bmatrix} \begin{bmatrix}
  \omega_R \\
  \omega_L
\end{bmatrix} = \begin{bmatrix}
  \frac{r \cos \theta}{d} & \frac{r \cos \theta}{d} \\
  \frac{r \sin \theta}{d} & \frac{r \sin \theta}{d}
\end{bmatrix} \begin{bmatrix}
  \omega_R \\
  \omega_L
\end{bmatrix}
\]
Mobile robot control
Actuators (DC motors, step-by-step motor, ...)  
End-Effector (gripper, hand, probe, etc.)  
Sensors  
  - Internal state (encoder, gyro, ...)  
  - Environmental (bumpers, rangefinders (infrared, ultrasound), laser, vision (mono, stereo), ...)  
Control  
  - Low level  
  - High level
Foundamentals

- High gain PI controllers are used to control robot motors such that it moves following the desired velocity profile.
- If the gain is high enough, the low level control *transforms* the robot in a kinematic system.
- Low level control is in charge of controlling the robot actuators following high level controller instructions.
Low level vs. High level control

Foundamentals
- The signal provided to the low level controller is computed by analyzing information gathered by sensors
- Control signals are velocities
- From a high level point-of-view the robot is considered as a kinematic system

High level
- it defines the robot *behaviour*
- it takes into account the kinematic model
- it is subject to wheels constraints
- it has to control a complex *non linear* system

Low level
- it controls the motors
- usually it is a PI control (*linear system*)
- it is not affected by wheels constraints
Elementary motion action

- **Point-to-point transfer**
  - Initial position
  - Final position

- **Trajectory Following**
  - Initial position
  - Parameter $s$
  - Reference WMR
  - Final position

- **Trajectory Tracking**
  - Initial position
  - Parameter $t$
  - Reference WMR at time $t$
  - Final position
Motion control

Two different problems

Configuration
Starting from an initial configuration \( q_0 = [x_0, y_0, \theta_0]^T \), the robot has to reach a predefined desired configuration \( q_d = [x_d, y_d, \theta_d]^T \).

More difficult but generic problem

Trajectory
Starting from an initial configuration \( q_0 = [x_0, y_0, \theta_0]^T \) that can be or not on the trajectory, the robot has to reply asymptotically a cartesian trajectory \( [x_d(t), y_d(t)]^T \):

- **Trajectory Following** if the problem is position-dependent (s)
- **Trajectory Tracking** if the problem is time-dependent (t)

Simpler and more practical problem
**Basic idea**

- To define a point $b$ out of the wheels axis
- To control the point $b$
- The point $b$ pulls the vehicle

\[
\begin{align*}
  x_b &= x_r + b \cos \theta_r, \quad b \neq 0 \\
  y_b &= y_r + b \sin \theta_r
\end{align*}
\]

- The point $b$ has no constraints and it can move instantaneously in all the directions
- The point $b$ can move sideways with respect to the forward direction

It is possible to define two inputs to control $v_{x,b}$ and $v_{y,b}$

\[
\begin{align*}
  \dot{x}_b &= v_{x,b} \\
  \dot{y}_b &= v_{y,b}
\end{align*}
\]
Trajectory following - I-O SFL

\[
\begin{cases}
\dot{x}_b = \dot{x}_r - b \cdot \omega \sin \theta_r = v \cos \theta_r - b \cdot \omega \sin \theta_r \\
\dot{y}_b = \dot{y}_r + b \cdot \omega \cos \theta_r = v \sin \theta_r + b \cdot \omega \cos \theta_r, \ b \neq 0 \\
\dot{\theta}_r = \omega
\end{cases}
\]

\[
\begin{bmatrix}
\dot{x}_b \\
\dot{y}_b
\end{bmatrix} =
\begin{bmatrix}
\cos \theta & -b \cdot \sin \theta_r \\
\sin \theta & b \cdot \cos \theta_r
\end{bmatrix}
\begin{bmatrix}
v \\
\omega
\end{bmatrix} = T(b, \theta_r)
\begin{bmatrix}
v \\
\omega
\end{bmatrix}
\]

It is possible to prove that
if \( b \neq 0 \), then \( \det(T(b, \theta_r)) \neq 0 \)

the matrix \( T(b, \theta_r) \) is always invertible

\[
\begin{bmatrix}
v \\
\omega
\end{bmatrix} = T(b, \theta_r)^{-1}
\begin{bmatrix}
\dot{x}_b \\
\dot{y}_b
\end{bmatrix} =
\begin{bmatrix}
\cos \theta & \sin \theta_r \\
-\frac{1}{b} \sin \theta & \frac{1}{b} \cos \theta_r
\end{bmatrix}
\begin{bmatrix}
\dot{x}_b \\
\dot{y}_b
\end{bmatrix}
\]
Trajectory following - I-O SFL

From the previous relationships the following linear system can be calculated

\[
\begin{align*}
\dot{x}_b &= v_{x,b} \\
\dot{y}_b &= v_{y,b} \\
\dot{\theta}_r &= \frac{1}{b} (v_{y,b} \cos \theta_r - v_{x,b} \sin \theta_r)
\end{align*}
\]

The x and y moving directions of the point \(b\) can be controlled independently by using the inputs \(v_{x,b}\) and \(v_{y,b}\)

Given a desired trajectory \((x_d(\cdot), y_d(\cdot))\) to be followed, it is possible to define \(v_{x,b}\) and \(v_{y,b}\) in order to ensure the asymptotic trajectory tracking by using the following linear control laws:

\[
\begin{align*}
\dot{x}_b &= v_{x,b} = \dot{x}_d + K_1 (x_d - x_b) \\
\dot{y}_b &= v_{y,b} = \dot{y}_d + K_2 (y_d - y_b)
\end{align*}
\]

Error dynamics

By defining \(\begin{cases} e_x = x_d - x_b \\ e_y = y_d - y_b \end{cases}\) \(\Rightarrow\) \(\begin{cases} \dot{e}_x + K_1 e_x = 0 \\ \dot{e}_y + K_2 e_y = 0 \end{cases}\) \(\iff\) \(\begin{cases} e_x \to 0 \\ e_y \to 0 \end{cases}\)
Once the inputs to control $b$ have been defined, it is possible to calculate the control inputs to be provided to the unicycle in order to move $b$ as desired.

\[
\begin{bmatrix}
\nu \\
\omega \\
\end{bmatrix} =
\begin{bmatrix}
\cos \theta & \sin \theta_r \\
-\frac{1}{b} \sin \theta & \frac{1}{b} \cos \theta_r \\
\end{bmatrix}
\begin{bmatrix}
\dot{x}_b \\
\dot{y}_b \\
\end{bmatrix}
\]
Interaction with the environment

Usually a robot has to move in an environment with static or moving obstacles. By exploiting its knowledge of the environment and by exploiting the data provided by the on-board sensors, the robot has to detect and avoid the obstacles in order to safety navigate the workspace, i.e. avoiding collisions.

Navigation problem

Given a starting and final configuration, the navigation problem deals with the possibility of finding a collision free path that can be followed by the robot in order to move from the starting to the final configuration.
Navigation: the Braitenberg Algorithm

- Valentino Braitenberg (Bolzano, 1924). Neuropsychiatric.
- A simple neural network elaborates the signals provided by the sensors on the right and left side of the robot and it applies the results directly to the motor.
The path planning based on the potential field approach is possible to plan the path even for fully actuated robots. Potential fields allow to generate the path incrementally by changing it every time an obstacle is detected.

**Potential function**

\[ U : \mathbb{R}^m \rightarrow \mathbb{R} \]

**Gradient of \( U \)**

\[
\nabla U(q) = \left[ \frac{\partial U}{\partial q_1}(q) \ldots \frac{\partial U}{\partial q_n}(q) \right]^T
\]

\[
q = \begin{bmatrix} q_1, & q_2 \end{bmatrix}
\]

\[
U(q) = \frac{1}{2} q^T q
\]

\[
\nabla U(q) = \begin{bmatrix} q_1, & q_2 \end{bmatrix}
\]

- \( U(q) \) is equivalent to an energy function defined over the space configuration
- \( \nabla U(\cdot) \) represents the forces generated by the potential field
- \( \nabla U(q) \) points in the \( U \) growing direction
Gradient descent

- Potential functions are like landscapes where the robot is immersed.
- The robot has to move from a *higher* point to a *lower* point.
- The robot has to follow a downhill path (negative gradient).

\[ \dot{q}(t) = -\nabla U(q) \]

The robot moves in a potential field defined as the sum of all the potential field present in the environment.
To create the potential field to drive a robot from $q_{\text{start}}$ to $q_{\text{goal}}$ while avoiding obstacles we have to consider two kind of potential fields

**Attractive Potential Field**

$$U_{\text{att}}(q)$$

- It has the minimum is in $q_{\text{goal}}$
- It has to attract the robot toward the goal position

**Repulsive Potential Field**

$$U_{\text{rep}} = \sum_{i=1}^{N_{\text{obst}}} U_{\text{rep},i}(q)$$

- It is calculated as the sum of all the repulsive potential field generated by each obstacle
- It has to push the robot away from the obstacles

**Potential field based control law**

$$\dot{q}(t) = -\nabla U(q), \quad \text{dove: } U(q) = U_{\text{att}}(q) + U_{\text{rep}}(q)$$
Navigation: Potential Fields

Attractive Potential Field

- Monotonically increasing with $\Delta(q, q_{\text{goal}})$
- Robot attracted in $\Delta(q, q_{\text{goal}})$
- $q_{\text{goal}}$ isolated minimum: no critical points

Example

$$U_{\text{att}} = \frac{1}{2} K_a \Delta(q, q_{\text{goal}})$$
$$\nabla U_{\text{att}} = \frac{1}{2} K_a \nabla \Delta^2(q, q_{\text{goal}}) = K_a (q - q_{\text{goal}})$$

- When the robot is far from the goal, it moves slowly
- When the robot is close to the goal, it moves fast
Navigation: Potential Fields

Repulsive Potential Field

- Given by the sum of the potential field associated to each obstacle
- The repulsive force increases as long as the robot moves closer to the obstacle
- The repulsive force is null after a certain predefined distance

Example

\[
U_{rep_i}(q) = \begin{cases} 
\frac{1}{2} K_{ri} \left( \frac{1}{d_i(q)} - \frac{1}{q^*} \right)^2 & d_i(q) \leq q^* \\
0 & d_i(q) > q^* 
\end{cases}
\]

\[
\nabla U_{rep_i}(q) = \begin{cases} 
K_{ri} \left( \frac{1}{q^*} - \frac{1}{d_i(q)} \right)^2 \nabla d_i(q) & d_i(q) \leq q^* \\
0 & d_i(q) > q^* 
\end{cases}
\]

- If \(d_i(q) = 0\), then \(\nabla U_{rep}(q) \to \infty\)

Local minima
The algorithm based on the gradient descent could be stuck in a *local minima*, that is a particular configuration where attractive and repulsive field are equal. This is not a *complete* planning method. Except some particular cases, the local minima are always present. Generically could be avoided by using convex obstacles.

**Possible solutions**

- *Best first* algorithm
- *Best first randomized* algorithm
- Navigation functions
- Adding noise
- *Escape window* algorithm