

Tesi di Laurea in Ingegneria Informatica

**Path planning locale per robot mobili  
basato su switching tra potenziali artificiali**

Relatore:  
Prof. Luigi D'Alfonso

Candidato:  
Flavio Maiorana

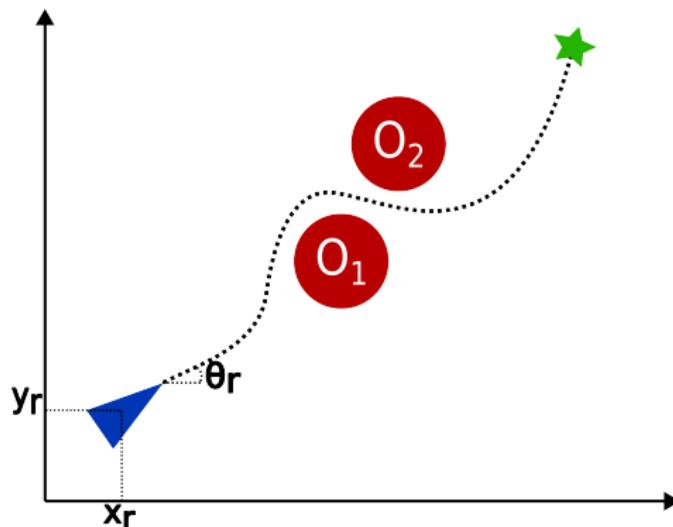
---

11 Luglio 2022

# Definizione del Problema

Path planning locale

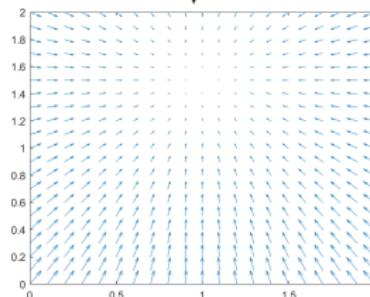
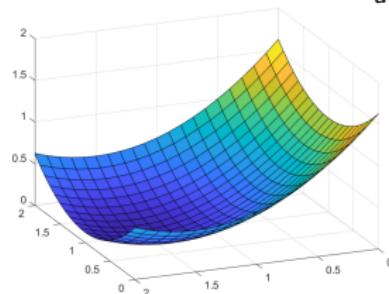
Guidare il robot mobile dalla sua **posizione iniziale** ad un **punto di goal**, evitando gli **ostacoli** non noti a priori



# Potenziali artificiali

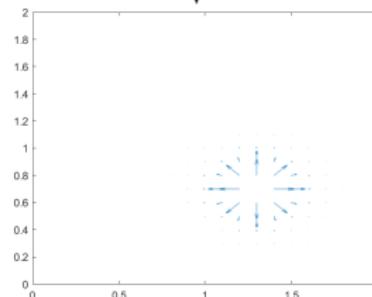
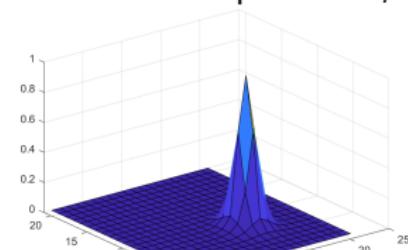
## Metodo classico

Potenziiale attrattivo  $U_a$



$$-\nabla U_a$$

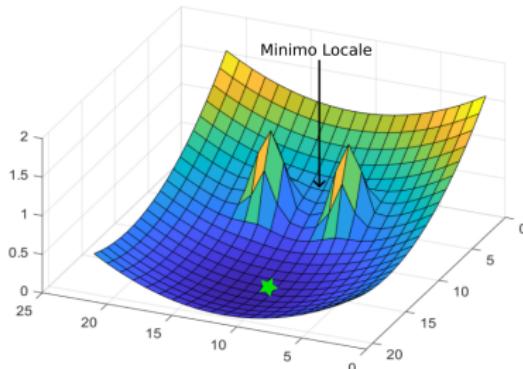
Potenziiale repulsivo  $U_r$



$$-\nabla U_r$$

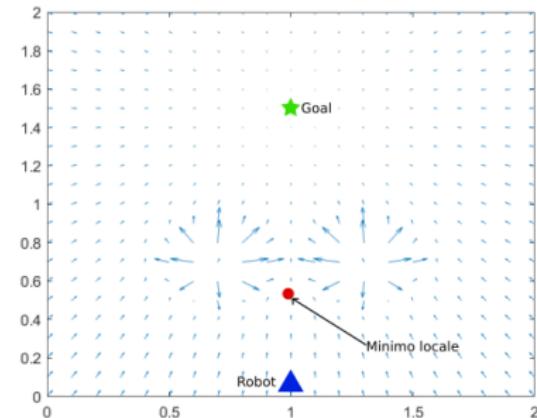
# Potenziali artificiali

Metodo classico



Potenziale totale

$$U_{tot} = U_a + U_r$$



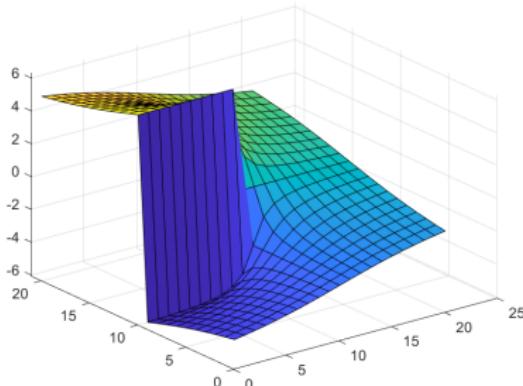
Antigradiente  $-\nabla U_{tot}$   
(Velocità di riferimento)

**La somma di potenziali causa il rischio di minimo locale**

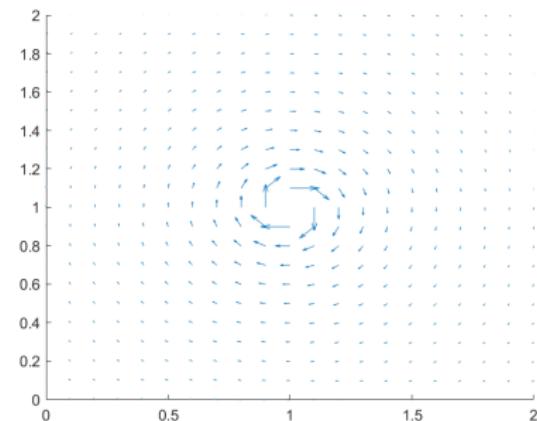
# Potenziali artificiali

## Potenziale bypassante

- Alternativa al potenziale repulsivo
- Antigradiente di forma circolare e non radiale
- Intensità aumenta con avvicinamento all'ostacolo



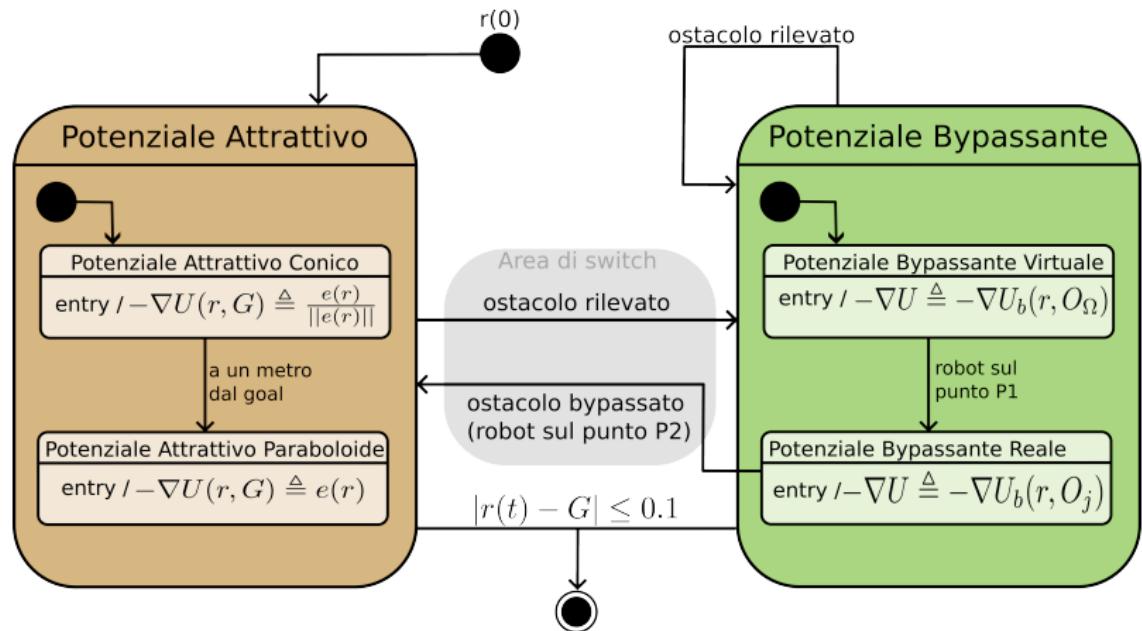
Potenziale totale  $U_b$



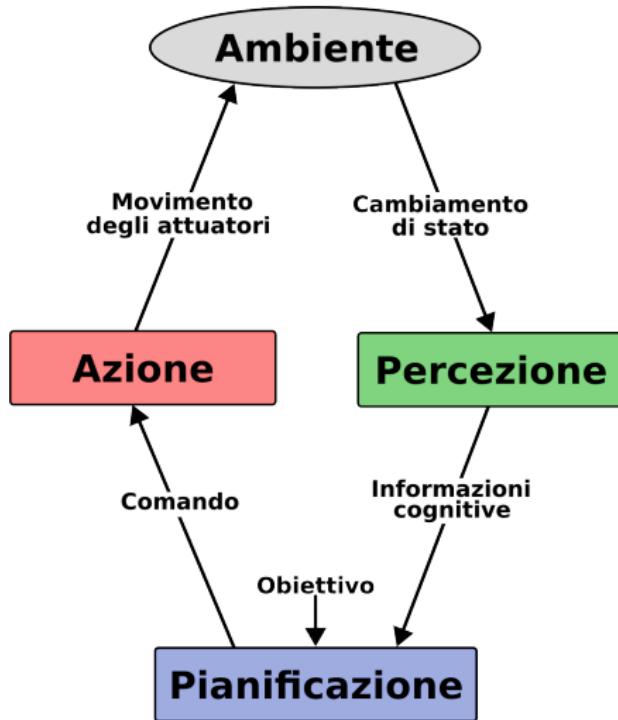
Antigradiente  $-\nabla U_b$

# Panoramica sull'algoritmo

Siano  $U_a$  e  $U_b$  il potenziale attrattivo e bypassante: in ogni  $t$  il robot é guidato da un solo potenziale tramite l'antigradiente  $-\nabla U$



# Architettura di navigazione



Entry point del modulo software

```
function start
error = norm([xr ,yr] -G);
while(error > 0.1)
    %Detected obstacle
    dObstacle = sense.scan();
    %New directive
    state.decision(dObstacle);
    %Commands to the actuators
    newPose = act.move(tspan);
    %Refreshing the error
    error = norm([xr ,yr] -G);
end
end
```

# Modulo di percezione



# Modulo di percezione

Punto di partenza

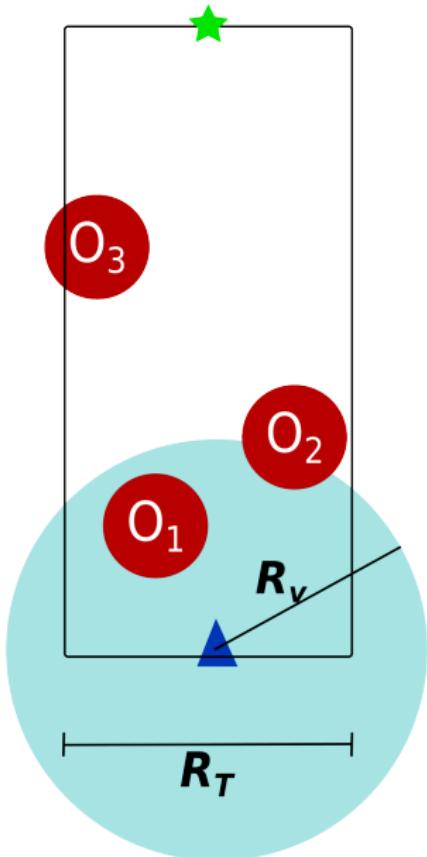
- Raggio di visione  $R_v$
- Tubo  $T$  verso il goal di larghezza  $R_T$
- Localizzazione sul centro dell'ostacolo



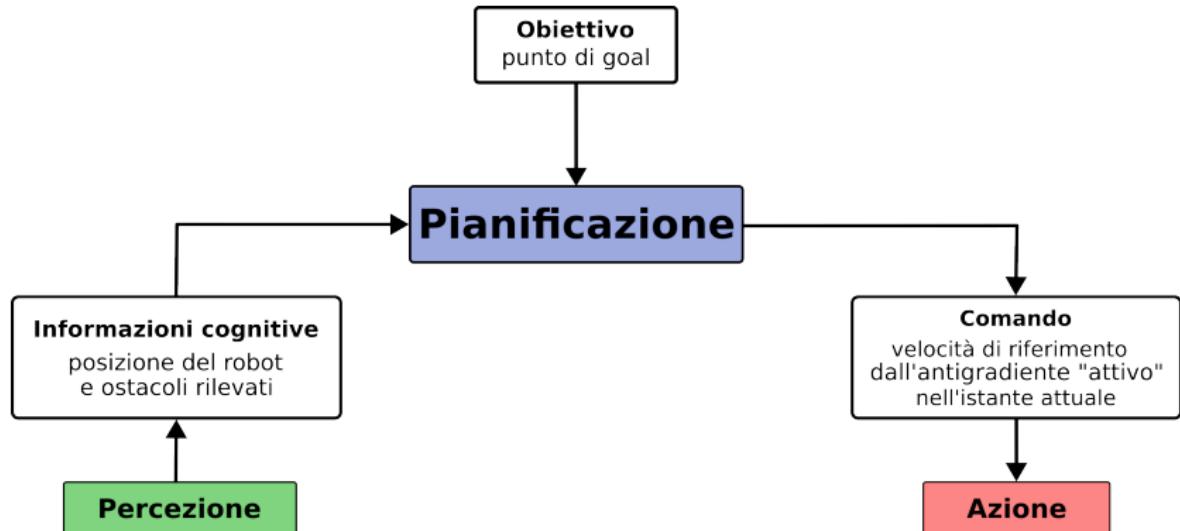
Un ostacolo risulta **rilevato** se valgono entrambe le condizioni

- $\|r(t) - O_j(t)\| \leq R_v$
- $O_j(t) \in T(r, G, R_T)$

L'ostacolo da bypassare è quello a distanza minima dal robot tra quelli rilevati

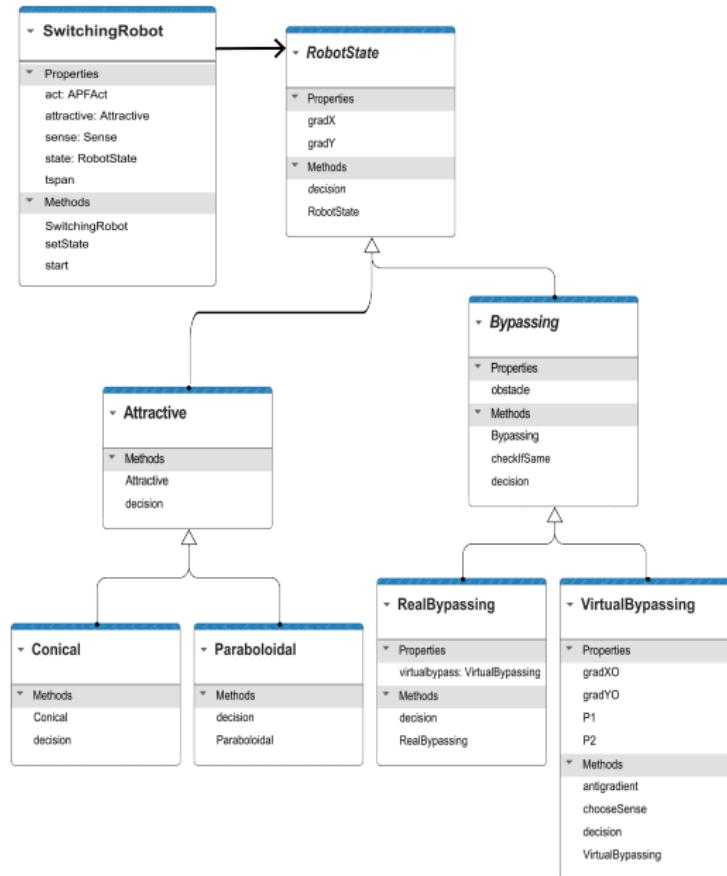


# Modulo di pianificazione



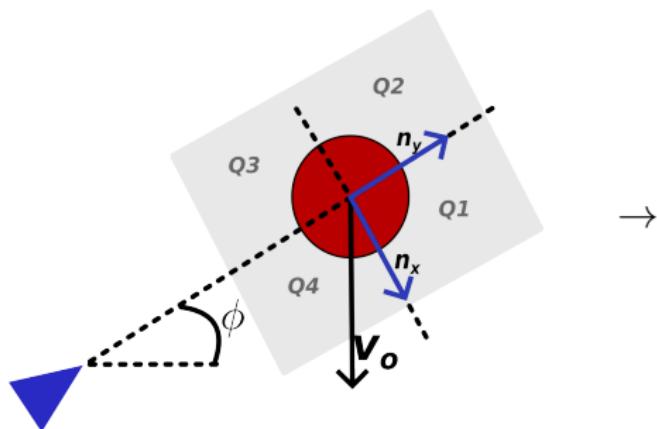
# Modulo di pianificazione

## Design pattern State

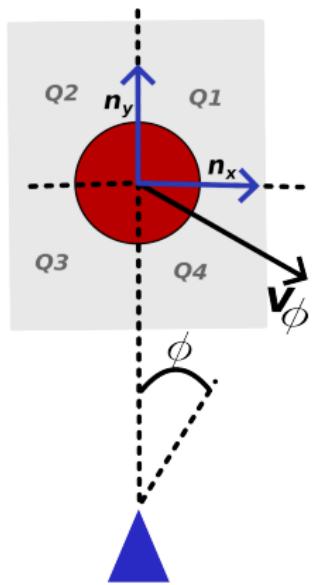


# Modulo di pianificazione

Switching : scelta del verso di bypassing



→

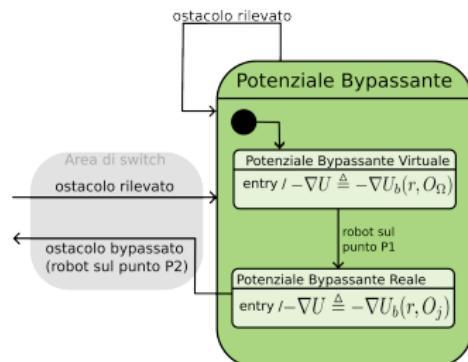
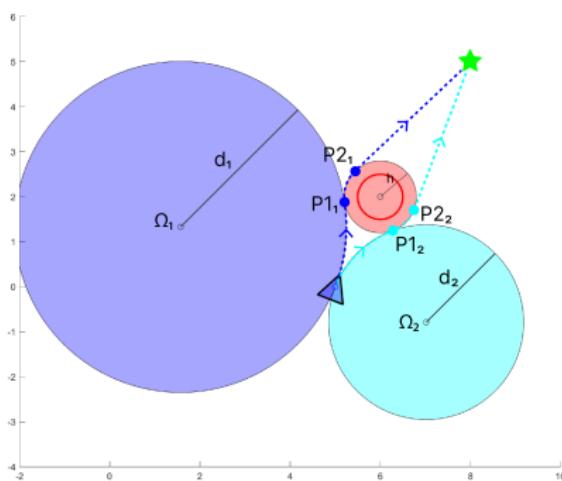


$$\begin{cases} \cos v_\phi \geq 0 \Rightarrow \text{orario} \\ \cos v_\phi < 0 \Rightarrow \text{antiorario} \end{cases}$$

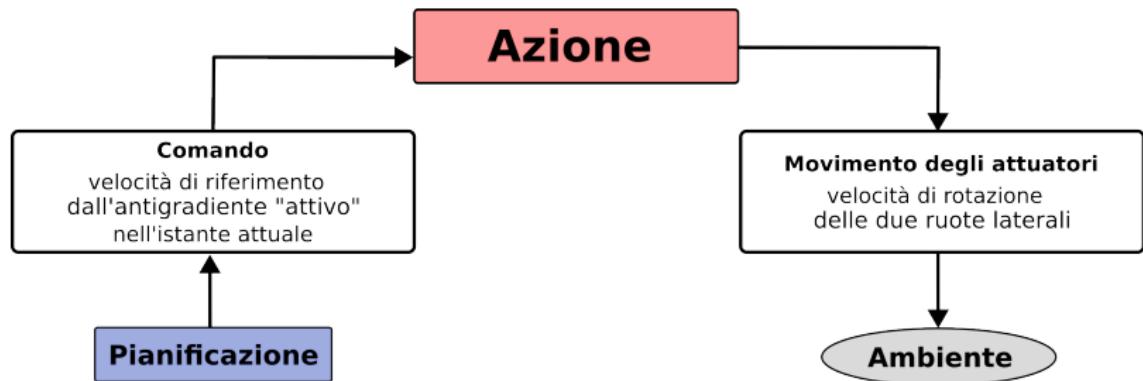
# Modulo di pianificazione

Switching: assenza di discontinuità

- $U_b$  calcolato nell'**istante**  $\tau$  di rilevamento dell'ostacolo
- Ausilio di un **ostacolo virtuale** con potenziale  $U_\Omega$
- **Condizioni di tangenza** tra le curve di livello di  $U_b$  e  $U_\Omega$



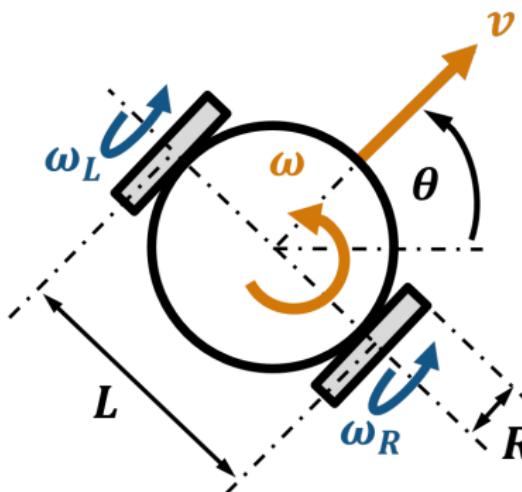
# Modulo di azione



# Modulo di azione

## Modello cinematico

$$\begin{cases} \dot{x}(t) = v(t) \cos(\theta(t)) \\ \dot{y}(t) = v(t) \sin(\theta(t)) \\ \dot{\theta}(t) = \omega(t) \end{cases} \Rightarrow \begin{cases} \dot{x}(t) = \frac{R}{2} (\omega_R(t) + \omega_L(t)) \cos(\theta(t)) \\ \dot{y}(t) = \frac{R}{2} (\omega_R(t) + \omega_L(t)) \sin(\theta(t)) \\ \dot{\theta}(t) = \frac{R}{L} (\omega_R(t) - \omega_L(t)) \end{cases}$$



Dettaglio: velocità delle ruote

$$\begin{bmatrix} \omega_R(t) \\ \omega_L(t) \end{bmatrix} = \begin{bmatrix} \frac{R}{2} & \frac{R}{2} \\ \frac{R}{L} & -\frac{R}{L} \end{bmatrix}^{-1} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix}$$

# Modulo di azione

## Legge di controllo

- Obiettivo: ottenere  $v(t)$  e  $\omega(t)$  (comandi di velocità)
- La velocità di riferimento è  $v_\nabla(t) \triangleq -\nabla U(r, G)$
- Sia  $M_v \triangleq \|v_\nabla(t)\|$  e  $\theta_\nabla \triangleq \angle v_\nabla(t)$

Tramite la **legge di controllo** nelle relazioni

$$v(t) = M_v \cos(\theta_\nabla(t) - \theta_r(t)) \quad (1)$$

$$\omega(t) = K_\omega(\theta_\nabla(t) - \theta_r(t)) \quad (2)$$

dove

$$K_\omega(t) \triangleq \begin{cases} \frac{\dot{\theta}_\nabla(t) + K_c |\theta_\nabla(t) - \theta_r(t)|^\nu \cdot sign(\theta_\nabla(t) - \theta_r(t))}{\theta_\nabla(t) - \theta_r(t)} & |\theta_\nabla(t) - \theta_r(t)| \geq \xi \\ 0 & \text{altrimenti} \end{cases}$$

si assicura che **la velocità del robot converga a  $v_\nabla(t)$  in t finito**

# Risultati

Cinque ostacoli in movimento

## Dati

- Assi cartesiani in metri
- $R_v = 1.5m$
- $R_T = 2m$
- $R_i = 0.5m \forall i$
- $r(0) = [5, 0]$
- $G = [8, 10]$

## Osservazioni

- Scelta verso cruciale
- Distanza di sicurezza  
dall'ostacolo proporzionale  
ad un "indice di invasività"
- Switching da bypassante di  
 $O_i$  a bypassante di  $O_j$

# Risultati

## Rischio minimo locale

### **Algoritmo con switching**

Il minimo locale non esiste e il robot passa con disinvoltura tra i due ostacoli

### **Algoritmo con metodo classico**

Il robot si ferma prima di attraversare il “varco” a causa di un annullamento di  $-\nabla U_{tot}$

# Risultati

Quindici ostacoli fermi

## Osservazioni

- L'algoritmo è performante anche con un elevato numero di ostacoli in proporzione alla dimensione dell'ambiente
- La traiettoria è ottima rispetto al tempo per raggiungere il goal e i vincoli dati dagli ostacoli presenti

# Conclusioni

## Riassunto

- Algoritmo di path planning basato su **switching** che **non causa rischio di minimi locali**
- Strategie di switching che assicurano il **bypass di ostacoli mobili** e **assenza di discontinuità** di velocità
- Legge di controllo in tempo finito per differential drive
- **Software modulare** e ausilio di design pattern

## Espansioni

- Ostacoli che cambiano velocità o di forma diversa
- Path planning multi agente
- Integrazione nel middleware ROS

“Il camminare presuppone che a ogni passo il mondo cambi in qualche suo aspetto e pure che qualcosa cambi in noi”

Italo Calvino