

Navigation, Path Planning e Obstacle Avoidance

Corrado Santoro

ARSLAB - Autonomous and Robotic Systems Laboratory

Dipartimento di Matematica e Informatica - Università di Catania, Italy

santoro@dmi.unict.it

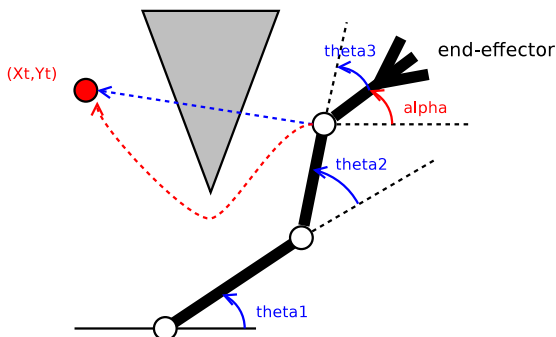


Programmazione Sistemi Robotici

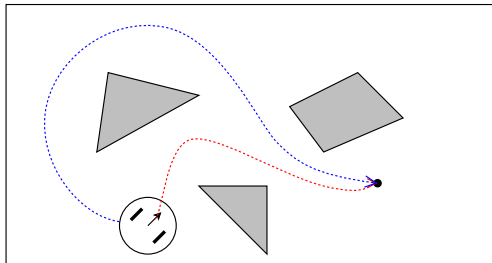
- Ogni sistema robotico è caratterizzato da un proprio **modello cinematico**
- Tale modello, che si utilizza per poter effettuare il **controllo in posizione**, prevede l'uso di
 - **grandezze controllabili**, es. velocità delle ruote etc.
 - **grandezze osservabili**, es. posizione delle ruote o dei giunti, etc.
 - **grandezze di lavoro**, cioè le variabili del sistema di riferimento "principale"
- Non si tiene tuttavia conto della **reale composizione dell'ambiente** e degli eventuali vincoli fisici che esso pone al movimento del sistema robotico

Ambiente e Vincoli

- Nel manipolatore planare in figura, il punto (X_t, Y_t) non è raggiungibile direttamente con un percorso rettilineo (**percorso blu**) a causa della presenza dell'ostacolo
- Il **percorso rosso** consentirebbe di superare l'ostacolo e raggiungere la posizione
- Tuttavia, il **percorso rosso**, non essendo geometricamente "rettilineo", andrebbe opportunamente **pianificato**



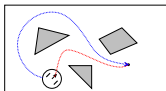
- Nel modello di robot mobile in figura, il punto di destinazione non è raggiungibile direttamente con un percorso rettilineo
- Tuttavia possono esistere svariati percorsi (due dei quali rappresentati in figura) che consentono al robot di raggiungere la posizione
- Stabilire il percorso è compito di un opportuno algoritmo di **path planning**



Path Planning

Path Planning: cosa serve

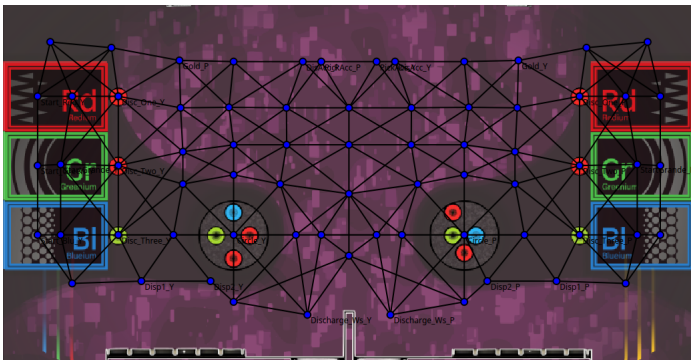
- **Modello dell'ambiente:** l'ambiente dove opera il robot, con le sue caratteristiche geometriche e con i suoi vincoli deve essere opportunamente modellato nel software; un qualunque percorso pianificato deve trovarsi sempre **dentro l'ambiente** e **non deve intersecarsi** con i vincoli
- **Modello del robot:** il robot non può essere trattato come “oggetto puntiforme”, ma la sua natura fisica e se le sue dimensioni devono essere tenute in conto nella pianificazione della traiettoria; questa deve poter essere seguita senza che il **robot possa intersecarsi** con i vincoli
- **Scelta della traiettoria:** occorre adottare una opportuna metodologia per la derivazione e la scelta della traiettoria, considerando opportuni parametri quali **distanza minima** o altro.



Fixed Graph

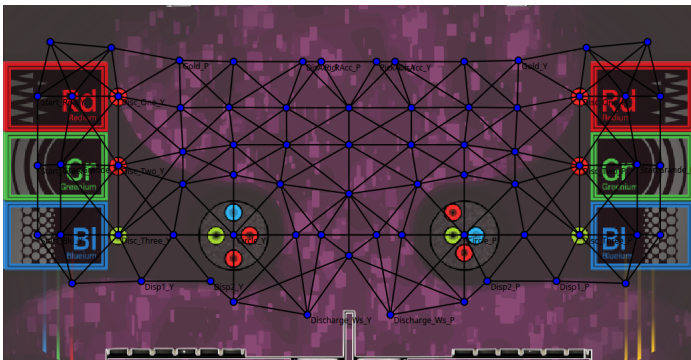
Fixed Graph (1)

- Questo metodo consiste nel definire **a priori** un **grafo** dei possibili percorsi
- La partenza (START) e la destinazione (END) del robot sono **sempre** rappresentati da un **vertice geometrico**
- Qualora il robot si trovi in una posizione che non corrisponde a un punto del grafo, si cerca il punto del grafo più vicino e lo si marca come START



Fixed Graph (2)

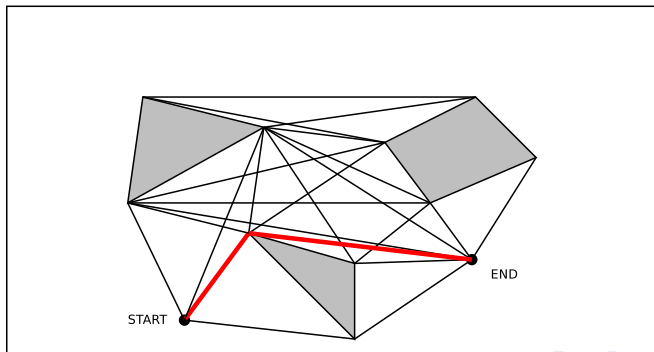
- Un percorso da START a END viene determinato usando, sul grafo, l'**algoritmo di Dijkstra** per il calcolo del percorso minimo
- Il percorso calcolato è dunque una lista di **punti** che rappresenta la **spezzata** da seguire
- Il percorso viene fornito al modulo di motion control che implementa l'algoritmo di controllo opportuno (esempio: controllo polare)



Visibility Graph

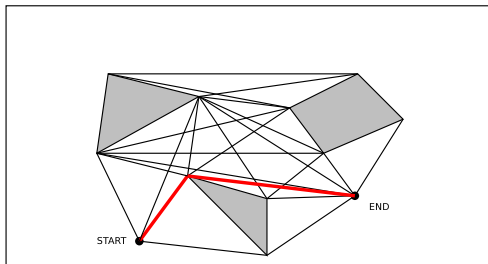
Visibility Graph (1)

- Questo metodo consiste nel modellare un **grafo** che rappresenta la **visibilità** di tutti i vertici delle geometrie dei vincoli presenti nell'ambiente
- Ogni **vertice geometrico** (inclusi i punti di START ed END) sono modellati con un **vertice nel grafo**
- Due vertici vengono collegati con un “edge” se essi sono **visibili**
- Sul grafo risultante, che include START ed END, viene avviato l'algoritmo di Dijkstra per il calcolo del percorso minimo



Visibility Graph (2)

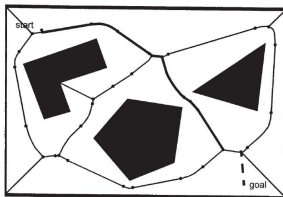
- **Vantaggi:** E' semplice ed efficiente da implementare
- **Svantaggi:**
 - I vincoli geometrici vanno “ingrossati” per tenere conto delle dimensioni del robot
 - Il path risultante passa “troppo vicino” ai vincoli (non è *safe*)
 - Se l'ambiente è denso di ostacoli, la complessità del grafo aumenta enormemente



Diagrammi di Voronoi

Voronoi Diagrams (1)

- I **diagrammi di Voronoi** sono definiti come il luogo geometrico dei punti che hanno distanza massima dagli ostacoli
- Essi si costruiscono usando un approccio tridimensionale:
 - Per ogni punto dello spazio di navigazione si determina la sua distanza dal vincolo più vicino
 - Tale distanza viene rappresentata come un'altezza nello spazio bidimensionale
 - I punti equidistanti da uno o più ostacoli formeranno un **picco (massimo)**
 - Si uniscono tutti i picchi tra loro: le geometrie risultanti sono le possibili "strade" che il robot può percorrere.



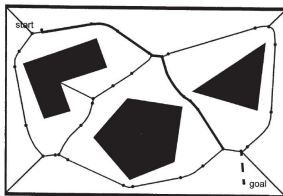
Voronoi Diagrams (2)

- **Vantaggi:**

- I possibili path sono minori e più efficienti rispetto al grafo di visibilità
- La navigazione è “safe” in quanto avviene sempre nei punti di maggiore distanza rispetto agli ostacoli.

- **Svantaggi:**

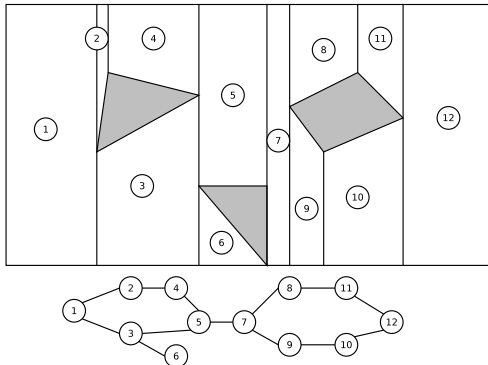
- La determinazione del diagramma è una procedura molto complessa (dal punto di vista computazionale), tuttavia essa viene avviata solo nella fase iniziale di costruzione



Cell Decomposition

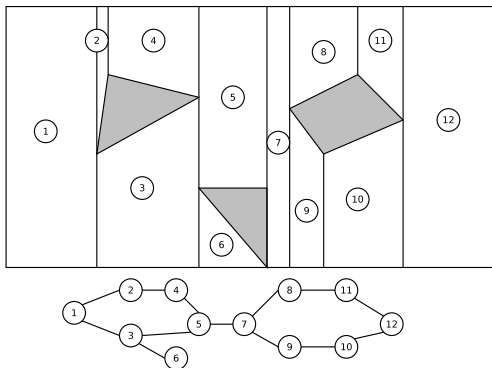
Cell Decomposition

- Si suddivide l'intera regione navigabile in un insieme di **celle**
- Per ogni cella si identifica un **punto di navigazione**
- Si costruisce un **grafo** in cui le celle sono rappresentate dai nodi e gli edge rappresentano il fatto che due celle siano confinanti
- Si fa girare l'algoritmo di Dijkstra per determinare il percorso minimo



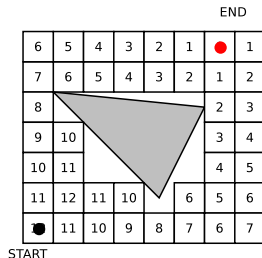
Cell Decomposition

- L'algoritmo è facilmente implementabile ed è abbastanza efficiente
- Le sue caratteristiche dipendono fortemente da come sono suddivise le celle:
 -
 - Più celle $\Rightarrow$ path più efficiente
 - Meno celle $\Rightarrow$ algoritmo più veloce



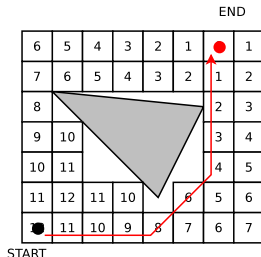
Fixed Cell Decomposition: NF1

- Si suddivide l'intera regione navigabile in un insieme di **celle di dimensioni fisse**
- Ogni cella è caratterizzata da una **marca numerica**
- Per ogni cella il **punto di navigazione** è il suo **centro**
- Si parte dal punto di arrivo (END), che si marca con il **valore "0"**
- Ogni cella adiacente si marca con un valore **incrementato di 1**
- Si procede fino a marcare **tutte le celle**



Fixed Cell Decomposition: NF1

- Per determinare il percorso, si parte dal punto di START e si sceglie la **cella adiacente con il valore di marca minore**
- Si procede fino ad arrivare al punto di END



Potential Field

Potential Field (1)

- Si modella lo spazio di navigazione come pervaso da un **campo di potenziale** composto da:
 - **Forze attrattive verso il target**
 - **Forze repulsive** che allontanano dai vincoli
- Il **potenziale** di un punto $q = (x, y)$ é definito come
$$U(q) = U_{att}(q) + U_{rep}(q)$$
- Il **vettore di moto** di un punto $q = (x, y)$ corrisponde alla **forza** generata dal potenziale $F(q) = -\nabla U_{att}(q) - \nabla U_{rep}(q)$
- dove il simbolo ∇ corrisponde all'operatore gradiente:

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

Potential Field (2)

- Se q_{goal} è il **punto target**, la forza attrattiva può essere modellata come:

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

dove k_{att} è uno scalare che agisce da “peso”

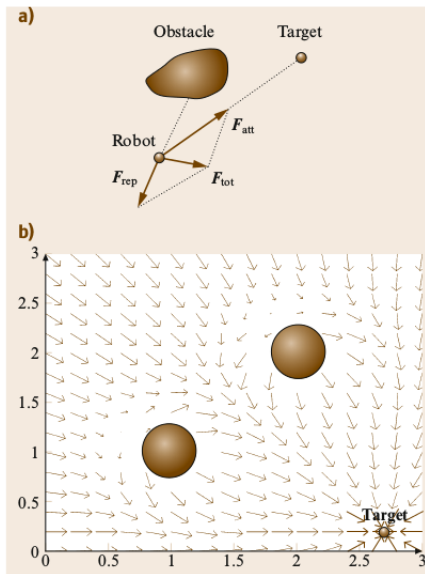
- la forza repulsiva può essere invece espressa come:

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

dove

- $q_{obstacle}$ è il punto dell'ostacolo a distanza minima
- $\rho(q)$ è la distanza minima da un ostacolo
- ρ_0 è una soglia di distanza
- k_{rep} è un fattore di peso

Potential Field (3)



Potential Field (4)

- Il metodo consiste nel imporre al robot delle velocità v_x e v_y proporzionali alle componenti della forza risultante.
- Il metodo consente di emulare il moto di una sfera che viene guidata gradualmente verso il punto target “scendendo da una collina” attraverso gli ostacoli; tuttavia ...
- L'algoritmo è fortemente dipendente dai pesi che si assegnano alla forza attrattiva e repulsiva k_{att} , k_{rep}
- L'andamento del gradiente non e' lineare; man mano che ci si avvicina agli ostacoli il contributo della repulsione diventa molto forte; questo può causare bruschi cambiamenti di velocità
- Nel caso in cui ci siano ostacoli vicini tra loro, il metodo può innescare sgradevoli oscillazioni nel moto
- Il metodo non tiene conto delle caratteristiche di manovrabilità specifiche del robot

Obstacle Detection & Avoidance

Obstacle Detection & Avoidance

- Il path planning consiste in una modellazione delle **caratteristiche dell'ambiente** e sulla determinazione della navigazione in tale ambiente
- L' **Obstacle Avoidance** implica invece che, data una traiettoria, venga identificata la presenza di un ostacolo ed esso venga evitato **posto che il target debba essere sempre raggiunto**
- L'obstacle avoidance presuppone che, durante il moto, un ostacolo non previsto possa trovarsi davanti al robot; in tal caso occorre:
 - **identificare la presenza dell'ostacolo**, operazione che si effettua tramite opportuni sensori
 - **evitare l'ostacolo**, operazione che comporta l'uso di opportuni algoritmi

Obstacle Detection

Sensori per l'identificazione dell'Ostacolo (1)

- **Sensori di distanza**
 - Ottici (luce)
 - Acustici (suono)
- **Telecamere** (con opportuni algoritmi di computer vision, qualora l'ostacolo abbia caratteristiche ben precise)
- **Telecamere “depth”** (visione stereoscopica)

Sensori per l'identificazione dell'Ostacolo (2)

● Sensori Acustici

- Sono basati sul calcolo del **tempo di volo** di un segnale acustico ad ultrasuoni
- Consentono di calcolare la **distanza dall'ostacolo**, con risoluzione (in genere) del **centimetro**
- Utilizzano opportune “capsule” ad ultrasuoni
- Hanno un **cono d'azione** ampio
- Sono soggetti a **interferenza mutua** qualora, in uno stesso ambiente, si utilizzino vari sensori di questo tipo e/o “puntati” l'uno verso l'altro



Sensori per l'identificazione dell'Ostacolo (3)

● Sensori Ottici a Soglia

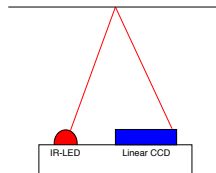
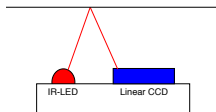
- Sono basati sull'identificazione di un raggio di luce che viene riflesso da un'ostacolo
- Consentono di determinare la **presenza dell'ostacolo** entro una certa soglia (regolabile)
- Sono sensori di tipo ON/OFF
- Utilizzano luce visibile (in genere rossa) o infrarosso
- Hanno un **cono d'azione molto stretto**
- Non sono soggetti a interferenza



Sensori per l'identificazione dell'Ostacolo (4)

● Sensori Ottici a Triangolazione

- Utilizzano un raggio di luce (visibile o infrarosso) la cui riflessione viene percepita da un sensore CCD lineare
- Misurano la distanza sulla base dell'identificazione del **triangolo** formato da raggio di luce emesso, raggio di luce riflesso e sensore
- Hanno un **cono d'azione abbastanza stretto**
- Non sono soggetti a interferenza



- **Sensori Ottici “time-of-flight”**

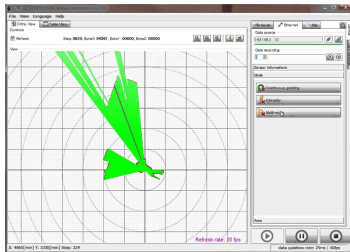
- Utilizzano un raggio di laser infrarosso e calcolano il “tempo di volo” del raggio riflesso
- Il rivelatore è costituito da un fotodiodo denominato SPAD (Single Photon Avalance Diode)
- Misurano la distanza con risoluzione del millimetro
- Hanno un **cono d'azione abbastanza stretto**
- Sono sensibili alle condizioni di illuminazione (in casi di forte illuminazione la distanza letta può essere falsata)
- Sono largamente utilizzati negli smartphone



Sensori per l'identificazione dell'Ostacolo (6)

● Sensori Ottici “LIDAR”

- Il **LIDAR** (**L**aser **I**maging **D**etection and **R**anging) è uno strumento che, attraverso la movimentazione di un laser e un sensore “ToF” permette una mappatura 2D dell'ambiente
- Il sensore è costituito da un ToF montato su una piattaforma rotante ad alta velocità (circa 10 giri/secondo)
- Misura la distanza con risoluzione del millimetro
- Produce in output un insieme di punti in **coordinate polari** (centrati sul sensore)

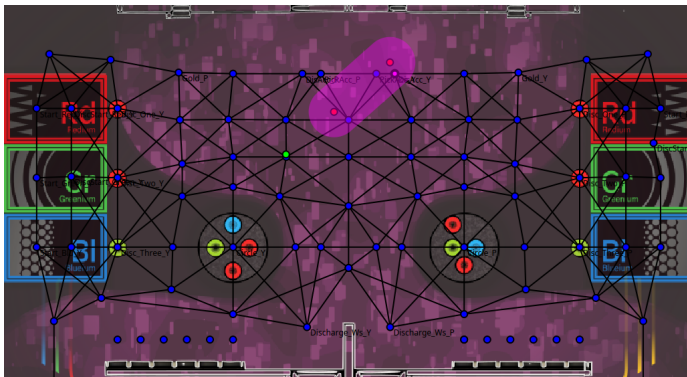


Obstacle Avoidance

- Se le **dimensioni dell'ostacolo** sono **note** (o determinabili a runtime), è possibile adottare uno degli algoritmi di path planning studiati; in tal caso occorre:
 - inserire l'ostacolo nel modello dell'ambiente
 - far girare l'algoritmo
 - se l'ostacolo è **mobile** utilizzare un **decay factor** per “far sparire” l'ostacolo dal punto dopo un po' di tempo

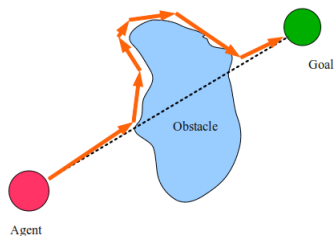
Fixed Graph e Obstacle Avoidance

- Su un **fixed graph** si **mappa** la zona dove potrebbe trovarsi l'ostacolo
- I nodi ed archi che intersecano l'ostacolo vengono (temporaneamente) rimossi dal grafo
- Il grafo risultante viene usato per determinare un nuovo percorso



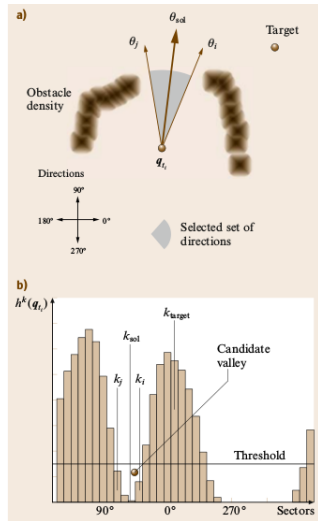
Bug Algorithm

- Se le dimensioni e la geometria dell'ostacolo **non sono note**, uno degli approcci può usati è il **bug algorithm**
- Si basa sulla presenza di un sensore a 360 gradi (sonar, LIDAR, etc.)
- Consiste nel circumnavigare l'ostacolo nella direzione del punto target fino a superarlo

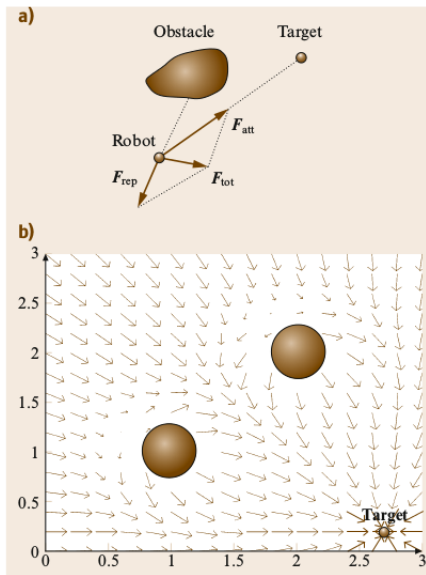


Vector Field Histogram

- Se le caratteristiche geometriche dell'ostacolo sono note, è possibile adottare il metodo **Vector Field Histogram**
- Si basa su due step:
 - 1 Determinazione di un diagramma polare della **densità** degli ostacoli (istogramma)
 - 2 Analisi dell'istogramma allo scopo di determinare il vettore di moto
- Si identificano le **valli** nell'istogramma, che corrispondono alle zone attraversabili dal robot
- Si sceglie la "valle migliore" per il raggiungimento del target
- Si determina, nella valle scelta, il vettore di moto da seguire considerando le dimensioni del robot

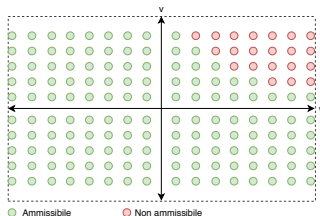
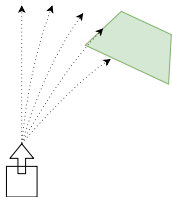


Potential Field



Dynamic Window Approach

- Il Dynamic Window Approach (DWA) consiste nel trattare il problema dell'obstacle avoidance nello spazio di moto del robot (v, ω)
- Data una posa del robot, ogni possibile coppia di valori (v_i, ω_i) genera una traiettoria circolare che incontra/non incontra l'ostacolo
- Tutte le coppie (v_i, ω_i) vengono riportate in un grafico
- Le coppie che portano ad una collisione sono marcate come **non ammissibili**
- Tra le coppie ammissibili viene scelta quella che
 - Porta al target nel minor tempo (massima velocità)
 - E' "raggiungibile" allo step successivo, date le caratteristiche di accelerazione del robot
 - Passa adeguatamente distante dall'ostacolo



Navigation, Path Planning e Obstacle Avoidance

Corrado Santoro

ARSLAB - Autonomous and Robotic Systems Laboratory

Dipartimento di Matematica e Informatica - Università di Catania, Italy

santoro@dmi.unict.it



Programmazione Sistemi Robotici