PIANIFICAZIONE DEL MOTO

Il problema canonico

Spazio delle configurazioni

Pianificazione mediante ritrazione

Pianificazione mediante decomposizione in celle

Pianificazione probabilistica

Pianificazione mediante potenziali artificiali

Il caso dei robot manipolatori

IL PROBLEMA CANONICO

- Pianificare un moto $\equiv$ decidere quale movimento deve effettuare il robot per eseguire un compito di trasferimento da una postura iniziale a una finale senza collidere con ostacoli
  - $\star$ pianificazione *autonoma*: sulla base di una descrizione ad alto livello del compito e di una rappresentazione dello spazio di lavoro (*fuori linea* o *in linea*)

- Problema canonico
  - $\star$ robot $\mathcal{B}$ in *spazio di lavoro* $\mathcal{W} = \mathbb{R}^N$ ($N = 2$ o 3), in *moto libero* (no vincoli cinematici)
  - $\star$ *ostacoli* $\mathcal{O}_1, \ldots, \mathcal{O}_p$

$$\Downarrow$$

- date posture iniziale e finale di $\mathcal{B}$ in $\mathcal{W}$, generare se esiste un *cammino* (sequenza continua di posture) che porti il robot dall'una all'altra e che sia privo di collisioni (inclusi i contatti) tra $\mathcal{B}$ e gli ostacoli $\mathcal{O}_1, \ldots, \mathcal{O}_p$
  - $\star$ $N = 2$: *piano movers' problem*
  - $\star$ $N = 3$: *generalized movers' problem*

- Situazioni reali
  - $\star$ ambienti non strutturati (ostacoli mobili)
  - $\star$ vincoli anolonomi
  - $\star$ contatti voluti (manipolazione, assemblaggio)

SPAZIO DELLE CONFIGURAZIONI

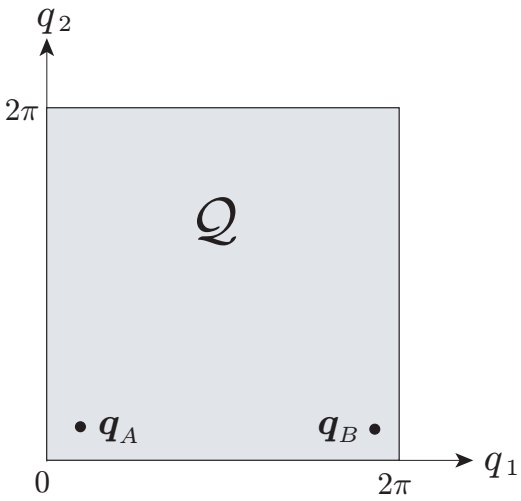
- Insieme delle configurazioni che il robot può assumere $\mathcal{C}$
- Configurazione $\boldsymbol{q} \in \mathbb{R}^n$
  - ⋆ robot mobile nel piano: $\mathcal{C} \equiv \mathbb{R}^2 \times SO(2)$ ($n = 3$)
  - ⋆ robot mobile nello spazio: $\mathcal{C} \equiv \mathbb{R}^3 \times SO(3)$ ($n = 6$)
  - ⋆ manipolatore planare a base fissa a n bracci (giunti elementari): $\mathcal{C} \subset (\mathbb{R}^2 \times SO(2))^n$ ($3n - 2n = n$)
  - ⋆ manipolatore spaziale a base fissa a n bracci (giunti elementari): $\mathcal{C} \subset (\mathbb{R}^3 \times SO(3))^n$ ($6n - 5n = n$)
  - ⋆ robot mobile con rimorchio nel piano: $\mathcal{C} \subset (\mathbb{R}^2 \times SO(2)) \times (\mathbb{R}^2 \times SO(2))$ (rimorchio connesso con giunto rotoidale $n = 4$: posizione+orientamento per motrice + orientamento per rimorchio)

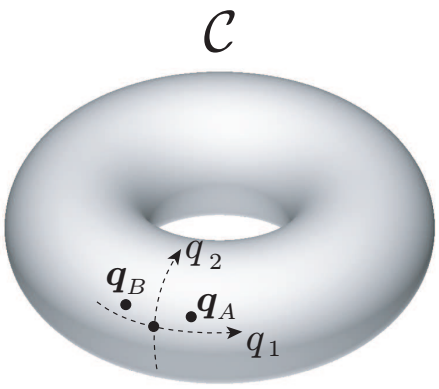
Esempio

- Manipolatore planare 2R

$$\mathcal{Q} = \{\boldsymbol{q} = (q_1, q_2) : q_1 \in [0, 2\pi), q_2 \in [0, 2\pi)\}$$

$$\mathcal{C} \equiv SO(2) \times SO(2)$$





Distanza

- $\mathcal{B} \subset \mathcal{W} \Rightarrow p(\boldsymbol{q})$: posizione del punto su $\mathcal{B}$
- Definizione

$$d_1(\boldsymbol{q}_1, \boldsymbol{q}_2) = \max_{p \in \mathcal{B}} \|p(\boldsymbol{q}_1) - p(\boldsymbol{q}_2)\|$$

- Norma euclidea

$$d_2(\boldsymbol{q}_1, \boldsymbol{q}_2) = \|\boldsymbol{q}_1 - \boldsymbol{q}_2\|$$

Ostacoli

⋆ necessità di costruire 'immagini' degli ostacoli in $\mathcal{C}$

- *$\mathcal{C}$–ostacolo*

$$\mathcal{CO}_i = \{\boldsymbol{q} \in \mathcal{C} : \mathcal{B}(\boldsymbol{q}) \cap \mathcal{O}_i \neq \emptyset\}$$

- *Regione dei $\mathcal{C}$–ostacoli*

$$\mathcal{CO} = \bigcup_{i=1}^{p} \mathcal{CO}_i$$

- *Spazio libero delle configurazioni*

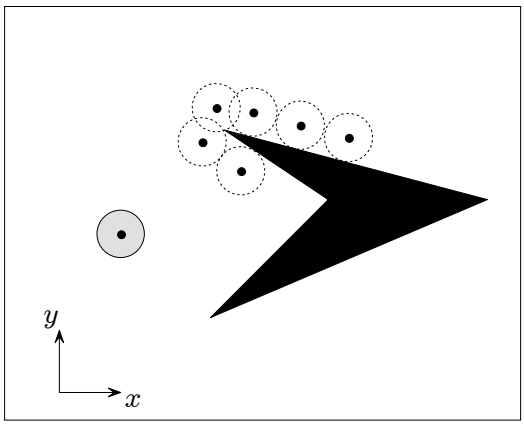
$$\mathcal{C}_{free} = \mathcal{C} - \mathcal{CO} = \{\boldsymbol{q} \in \mathcal{C} : \mathcal{B}(\boldsymbol{q}) \cap \left(\bigcup_{i=1}^{p} \mathcal{O}_i \right) = \emptyset\}$$

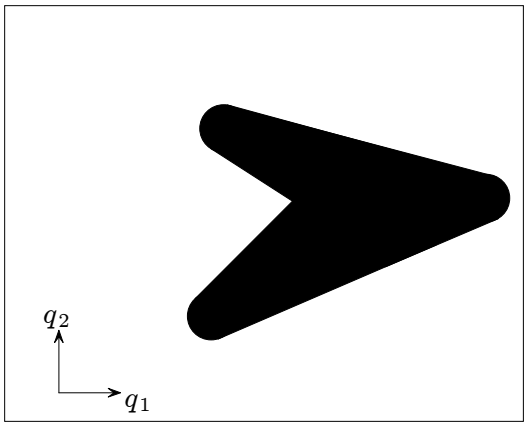
⋆ $\mathcal{C}_{free}$ può non essere connesso, a causa di occlusioni dovute ai $\mathcal{C}$–ostacoli

- Problema di generare un cammino libero tra $\boldsymbol{q}_s$ e $\boldsymbol{q}_g$

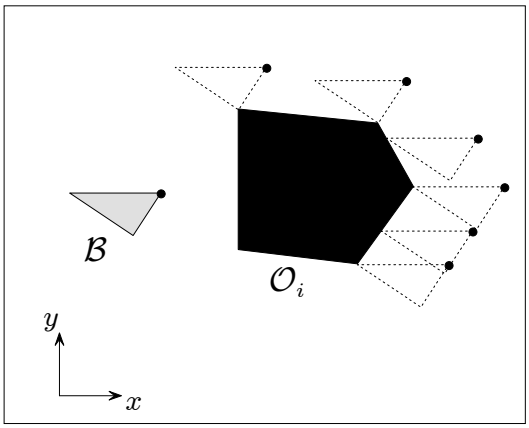
Esempi di ostacoli

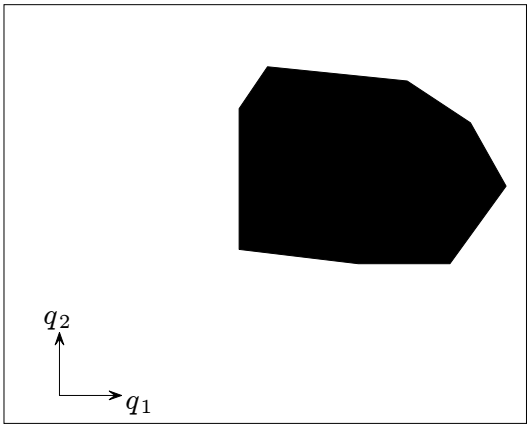
- Robot sfera in $\mathcal{W} = \mathbb{R}^N$



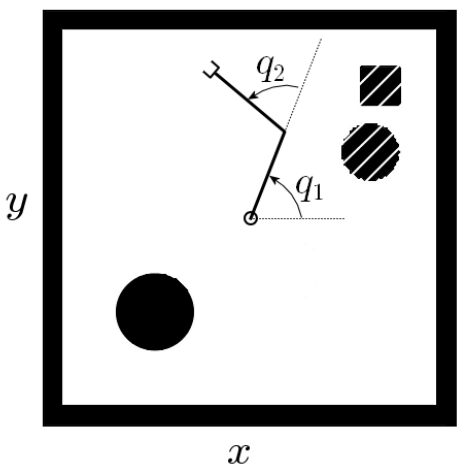


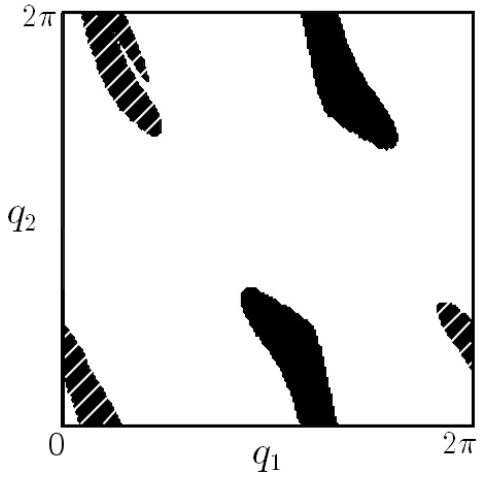
- Robot poliedrale libero di traslare (orientamento fisso) in $\mathbb{R}^N$

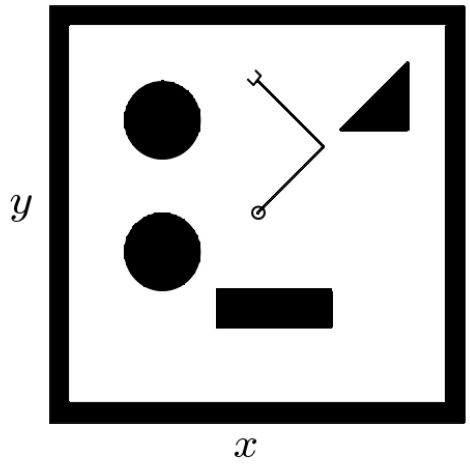


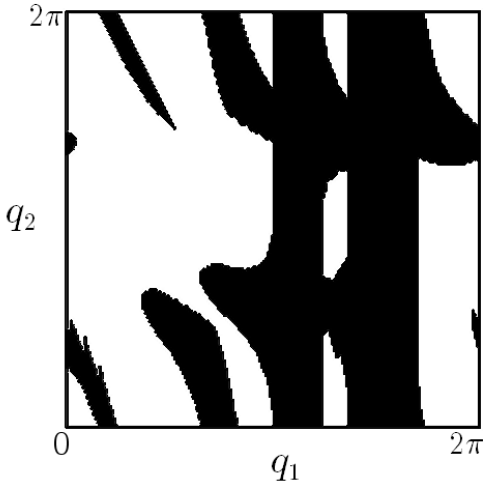


- Manipolatore planare (filiforme) 2R









- In generale, procedure complesse di generazione di $\mathcal{CO}$
  - ⋆ risoluzione della griglia di calcolo

PIANIFICAZIONE MEDIANTE RITRAZIONE

- Rappresentazione di $\mathcal{C}_{free}$ mediante una rete $\mathcal{R} \subset \mathcal{C}_{free}$ di cammini monodimensionali (*mappa stradale* o *roadmap*)

- La soluzione viene individuata connettendo (*ritraendo*) $\boldsymbol{q}_s$ e $\boldsymbol{q}_g$ a $\mathcal{R}$ e individuando su quest'ultima un cammino tra i punti di connessione

  ⋆ ipotesi $\mathcal{C}_{free}$: sottoinsieme limitato di $\mathcal{C} = \mathbb{R}^2$ di tipo *poligonale* (regione dei $\mathcal{C}$–ostacoli anch'essa poligonale)

- Minima distanza (*clearance*)

$$\gamma(\boldsymbol{q}) = \min_{\boldsymbol{s} \in \partial\mathcal{C}_{free}} \|\boldsymbol{q} - \boldsymbol{s}\|$$

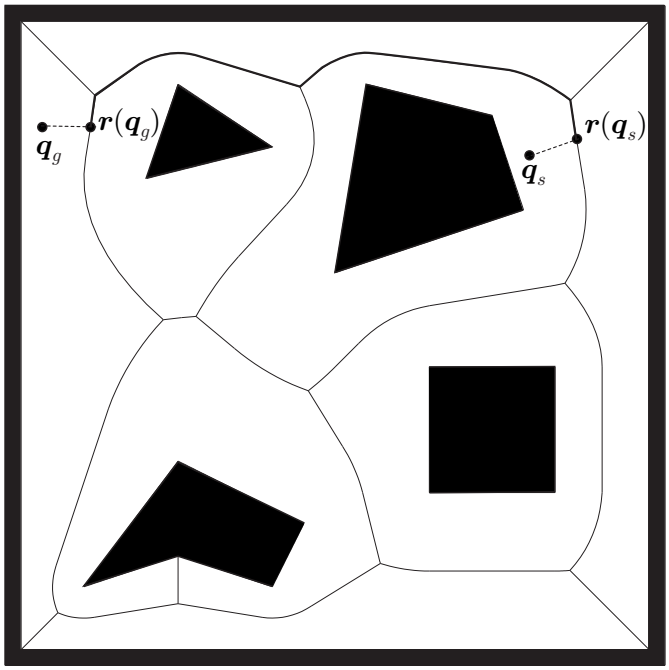
  ⋆ $\partial\mathcal{C}_{free}$: frontiera di $\mathcal{C}_{free}$

- Insieme dei punti *vicini* a $\boldsymbol{q}$

$$N(\boldsymbol{q}) = \{\boldsymbol{s} \in \partial\mathcal{C}_{free} : \|\boldsymbol{q} - \boldsymbol{s}\| = \gamma(\boldsymbol{q})\}$$

- Il *diagramma generalizzato di Voronoi* di $\mathcal{C}_{free}$ è l'insieme delle configurazioni che hanno più di un vicino:

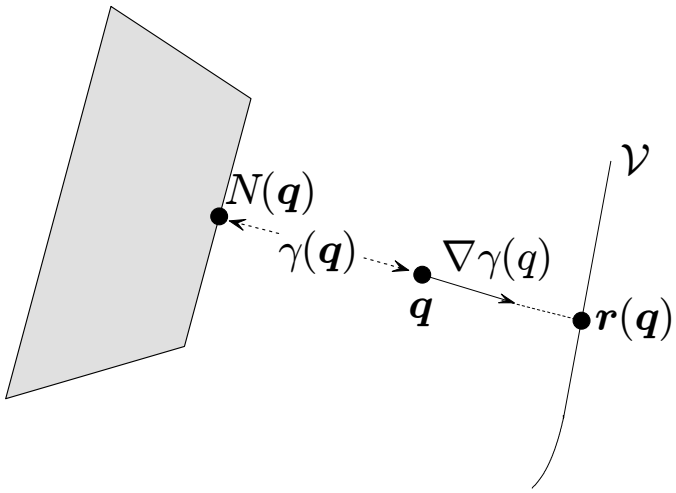
$$\mathcal{V}(\mathcal{C}_{free}) = \{\boldsymbol{q} \in \mathcal{C}_{free} : \text{card}(N(\boldsymbol{q})) > 1\}$$



  - ⋆ i suoi tratti elementari sono rettilinei (lato–lato, vertice–vertice) o parabolici (lato–vertice)
  - ⋆ può essere considerato come un *grafo* avente come *archi* i tratti elementari e come *nodi* gli estremi degli archi

- Roadmap naturale massimizzando localmente la clearance

- Procedura di ritrazione per connettere $\boldsymbol{q}$ a $\mathcal{V}(\mathcal{C}_{free})$



  - ⋆ segui $\nabla\gamma(\boldsymbol{q})$ sino alla prima intersezione $\boldsymbol{q}$ con $\mathcal{V}(\mathcal{C}_{free})$
  - ⋆ $\boldsymbol{r}(\cdot)$ preserva la connettività di $\mathcal{C}_{free}$ ($\boldsymbol{q}$ e $\boldsymbol{r}(\boldsymbol{q})$ appartengono sempre alla stessa componente connessa di $\mathcal{C}_{free}$)

$$\Downarrow$$

- Esiste un cammino libero da $\boldsymbol{q}_s$ a $\boldsymbol{q}_g$ se e solo se esiste un cammino libero tra $\boldsymbol{r}(\boldsymbol{q}_s)$ e $\boldsymbol{r}(\boldsymbol{q}_g)$

Algoritmo

1. Costruire il diagramma generalizzato di Voronoi $\mathcal{V}(\mathcal{C}_{free})$
2. Determinare le ritrazioni $\boldsymbol{r}(\boldsymbol{q}_s)$ e $\boldsymbol{r}(\boldsymbol{q}_g)$ su $\mathcal{V}(\mathcal{C}_{free})$
3. Ricercare su $\mathcal{V}(\mathcal{C}_{free})$ una sequenza di archi consecutivi tali che $\boldsymbol{r}(\boldsymbol{q}_s)$ appartenga al primo e $\boldsymbol{r}(\boldsymbol{q}_g)$ all'ultimo di essi
4. Se la ricerca ha successo, sostituire al primo arco della sequenza il suo sottoarco che origina in $\boldsymbol{r}(\boldsymbol{q}_s)$ e all'ultimo il suo sottoarco che termina in $\boldsymbol{r}(\boldsymbol{q}_g)$, e fornire in uscita il cammino costituito dal segmento di retta che unisce $\boldsymbol{q}_s$ a $\boldsymbol{r}(\boldsymbol{q}_s)$, dalla sequenza modificata di archi e dal segmento di retta che unisce $\boldsymbol{q}_g$ a $\boldsymbol{r}(\boldsymbol{q}_g)$; altrimenti, riportare un fallimento

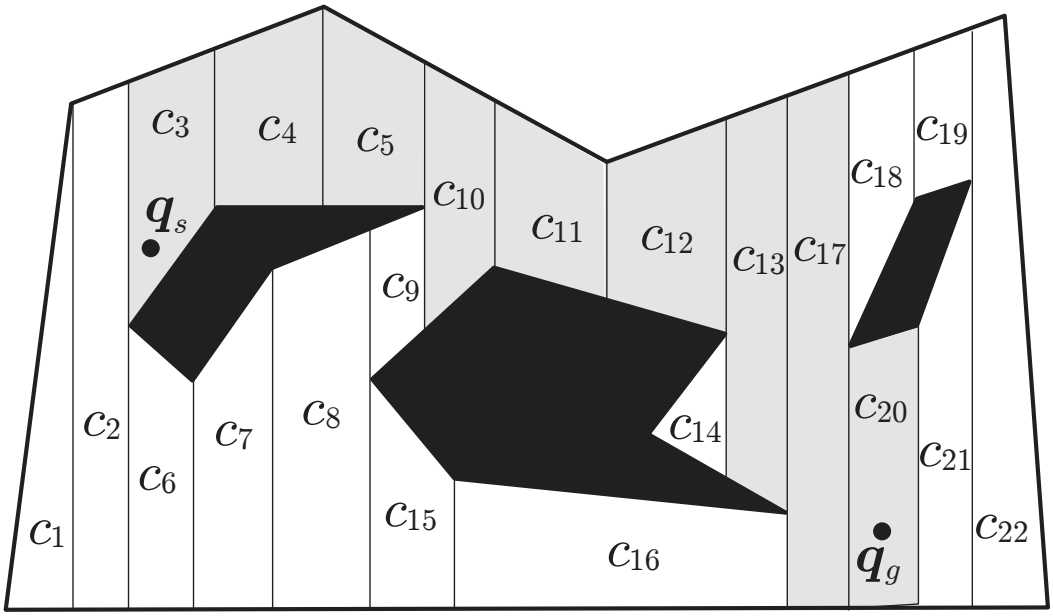
- Si utilizzano metodi di ricerca su grafo

PIANIFICAZIONE MEDIANTE DECOMPOSIZIONE IN CELLE

- Decomposizione di $\mathcal{C}_{free}$ in *celle*:
  - $\star$ è facile calcolare un cammino privo di collisioni tra due configurazioni nella stessa cella
  - $\star$ è facile calcolare un cammino privo di collisioni tra due celle adiacenti
- Calcolata una decomposizione in celle di $\mathcal{C}_{free}$, si cerca una sequenza di celle (*canale*) con $\boldsymbol{q}_s$ nella prima e $\boldsymbol{q}_g$ nell'ultima
- Si ottengono metodi differenti a seconda del tipo di celle usate per la decomposizione

Decomposizione esatta

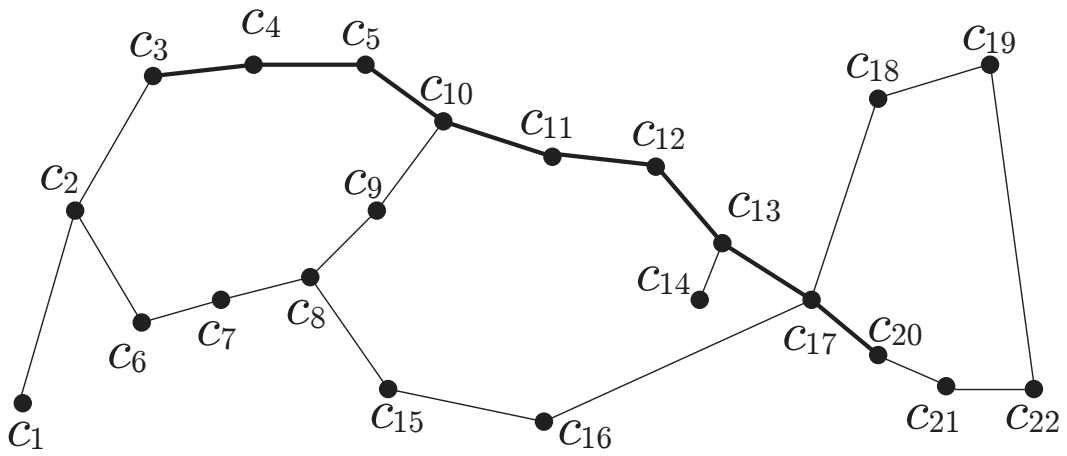
- Ipotesi
  - ⋆ $\mathcal{C}_{free}$ sottoinsieme limitato poligonale di $\mathcal{C} = \mathbb{R}^2$
- Decomposizione in celle attraverso poligoni convessi
  - ⋆ semplicità di transizione da una cella a quella adiacente (passando per il punto medio del segmento di frontiera comune)
- Algoritmo *sweep line* per la decomposizione di $\mathcal{C}_{free}$ in poligoni convessi (geometria computazionale)



- Retta non parallela ad alcun lato di $\partial\mathcal{C}_{free}$, a spazzare tutto $\mathcal{C}_{free}$
  - ⋆ ogni volta che si incontra un vertice di $\mathcal{C}_{free}$, due segmenti (*estensioni*) originano dal vertice
  - ⋆ una estensione in $\mathcal{C}_{free}$ è parte della frontiera della cella; il resto è costituito da lati di $\mathcal{C}_{free}$

$$\Downarrow$$

- Decomposizione trapezoidale



- Si costruisce il *grafo di connettività* C
  - ⋆ si individuano i nodi (celle) c_s e c_g cui appartengono $\boldsymbol{q}_s$ e $\boldsymbol{q}_g$
- La ricerca su grafo fornisce un *canale* tra c_s e c_g
  - ⋆ si estrae un cammino libero che unisce $\boldsymbol{q}_s$ a $\boldsymbol{q}_g$ via i punti medi dei segmenti di frontiera comune tra celle adiacenti

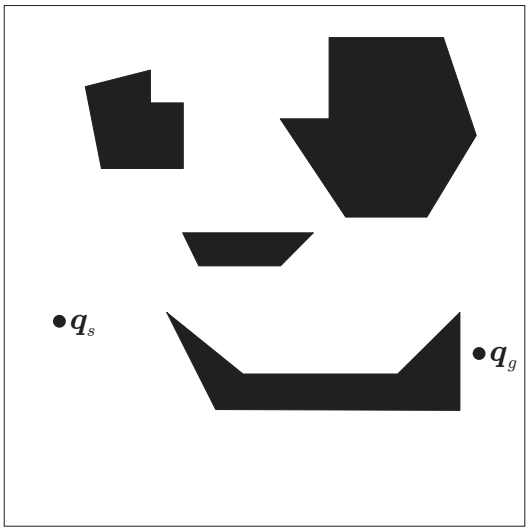
Algoritmo

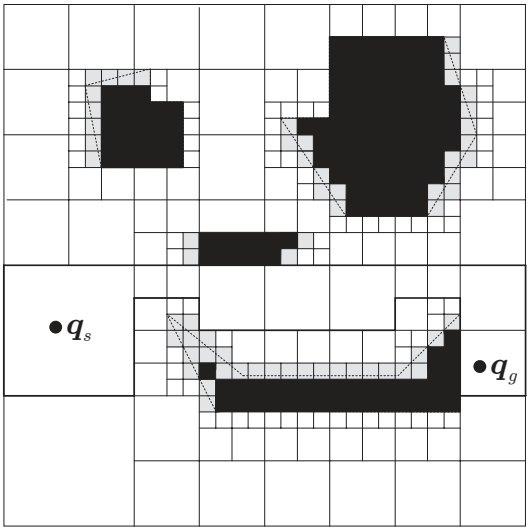
1. Generare una decomposizione poligonale convessa (trapezoidale) di $\mathcal{C}_{free}$
2. Costruire il grafo di connettività C associato alla decomposizione
3. Ricercare su C un canale di celle da c_s a c_g
4. Se la ricerca ha successo, estrarre un cammino privo di collisioni tra $\boldsymbol{q}_s$ e $\boldsymbol{q}_g$; altrimenti, riportare un fallimento

- Per la ricerca di un cammino *minimo*, si usa l'algoritmo di ricerca $A^\star$
- Il metodo è *multiple-query* (il grafo di connettività può essere riutilizzato)
- Il canale è più flessibile della roadmap poiché contiene una infinità di cammini (vincoli cinematici, ostacoli imprevisti)
- Possibilità di *curve fitting* tra spezzate
- Se $\mathcal{C}_{free}$ è un sottoinsieme limitato poliedrale di $\mathcal{C} = \mathbb{R}^3 \Rightarrow$ algoritmo *sweep-plane*
- Estensione a spazi di configurazione di dimensione arbitraria $\Rightarrow$ complessità esponenziale nella dimensione di $\mathcal{C}$

Decomposizione approssimata

- Ipotesi
  - ⋆ $\mathcal{C}_{free}$ sottoinsieme limitato poligonale di $\mathcal{C} = \mathbb{R}^2$
- Celle disgiunte ma di forma predefinita, per es. quadrati (approssimazione per difetto)
  - ⋆ la convessità garantisce la semplicità di pianificare in una cella e tra celle adiacenti
- Algoritmo ricorsivo per la decomposizione
  - ⋆ compromesso tra semplicità e accuratezza





- Dividere inizialmente $\mathcal{C}_{free}$ in 4 celle, classificandole:
  - ⋆ celle *libere*, il cui interno non ha intersezioni con la regione dei $\mathcal{C}$–ostacoli
  - ⋆ celle *occupate*, interamente contenute nella regione dei $\mathcal{C}$–ostacoli
  - ⋆ celle *miste*, né libere né occupate.

- Costruire il grafo di connettività C associato al livello corrente
  - ⋆ nodi $\equiv$ celle libere e miste
  - ⋆ archi $\equiv$ congiungono nodi rappresentativi di celle adiacenti

- Cercare un cammino (*canale libero*) tra i nodi corrispondenti a $\boldsymbol{q}_s$ e $\boldsymbol{q}_g$ (via algoritmo $A^\star$); altrimenti fallimento
  - ⋆ se *canale misto*, decomporre ciascuna cella mista in 4 celle . . . sino a trovare un canale libero o una dimensione minima per le celle

- Riducendo la dimensione minima ammissibile, si trova una soluzione se esiste (metodo completo in risoluzione)

- Più semplice della decomposizione esatta

- Decomposizione ricorsiva usando *quadtrees*
  - ⋆ generalizzabile in $\mathbb{R}^3$ (octrees)
  - ⋆ complessità esponenziale (max $n = 4$)

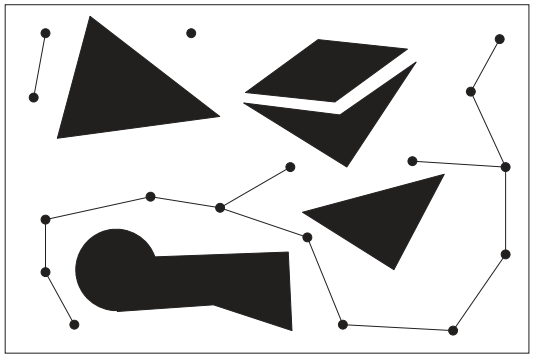
PIANIFICAZIONE PROBABILISTICA

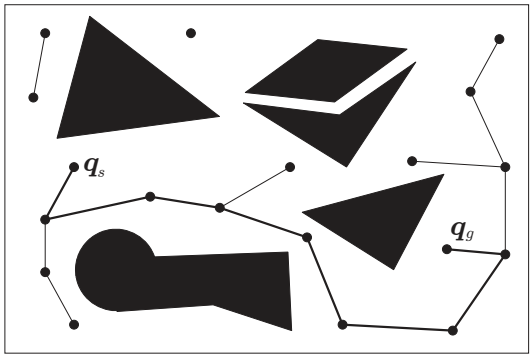
- Pianificatori *basati su campionamento*
  - ⋆ efficienti per pianificazione in spazi delle configurazioni a dimensione elevata

- Roadmap costruita attraverso un insieme finito di configurazioni appartenenti a $\mathcal{C}_{free}$ a connettività soddisfacente, quindi si itera
  - ⋆ approccio deterministico (griglia regolare sovrapposta a $\mathcal{C}$)
  - ⋆ approccio *randomizzato* (configurazioni campione generate in modo casuale)

Metodo Probabilistic Roadmap (PRM)

- Iterazione base per costruire la PRM
  - ⋆ generare $\boldsymbol{q}_{rand}$ in $\mathcal{C}$ secondo una *distribuzione di probabilità uniforme*
  - ⋆ test di collisione su $\mathcal{B}(\boldsymbol{q}_{rand})$
  - ⋆ se $\boldsymbol{q}_{rand} \in \mathcal{C}_{free}$, aggiungerla alla PRM; altrimenti, scartarla
  - ⋆ cercare lungo la PRM configurazioni sufficientemente vicine $\boldsymbol{q}_{near}$ (metrica euclidea o altre)
  - ⋆ se possibile, connettere $\boldsymbol{q}_{rand}$ a $\boldsymbol{q}_{near}$ con un cammino locale libero (*rettilineo*)





- Presenza di componenti della PRM non connesse tra di loro (una isolata)

- Non si calcolano i $\mathcal{C}$–ostacoli

- I passaggi stretti sono scarsamente campionati (distribuzioni diverse da quella uniforme)

- La generazione della PRM viene arrestata quando:
  - ⋆ si raggiunge un numero max di iterazioni
  - ⋆ il numero di componenti non connesse diventa inferiore a limite assegnato

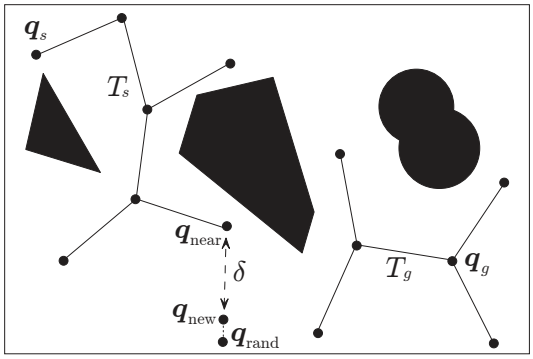
$$\Downarrow$$

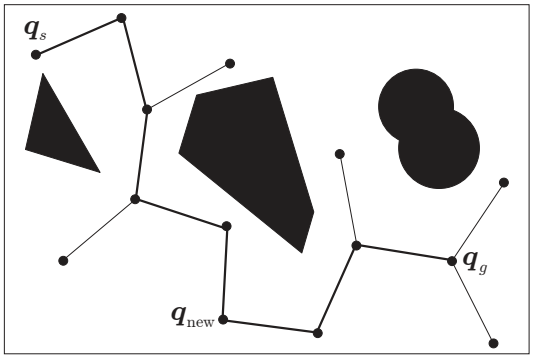
- Se $\boldsymbol{q}_s$ e $\boldsymbol{q}_g$ possono essere connesse alla stessa componente della PRM, trovare la soluzione con una ricerca su grafo; altrimenti, infittire la PRM attraverso più iterazioni

- Caratteristiche del metodo PRM
  - ⋆ intrinsecamente *multiple-query*
  - ⋆ semplice e veloce: il tempo per trovare una soluzione in spazi delle configurazioni di dimensione elevata ridotto di vari ordini di grandezza rispetto ai metodi precedenti
  - ⋆ è solo *completo in probabilità* $\equiv$ la probabilità di trovare una soluzione al problema di pianificazione, se questa esiste, tende a 1 al tendere del tempo di esecuzione a ∞ (se non esiste una soluzione, si arresta comunque dopo un numero max di iterazioni)

Metodo Rapidly-exploring Random Tree (RRT) bidirezionale

- Iterazione base per costruire un albero T
  - ⋆ T_s con radice in $\boldsymbol{q}_s$
  - ⋆ generare $\boldsymbol{q}_{rand}$ in $\mathcal{C}$ secondo una *distribuzione di probabilità uniforme*
  - ⋆ cercare l'albero per la configurazione più vicina $\boldsymbol{q}_{near}$ (metrica euclidea o altre)
  - ⋆ generare $\boldsymbol{q}_{new}$ a distanza δ da $\boldsymbol{q}_{near}$ nella direzione di $\boldsymbol{q}_{rand}$
  - ⋆ test di collisione su $\mathcal{B}(\boldsymbol{q}_{new})$ e il segmento da $\boldsymbol{q}_{near}$ a $\boldsymbol{q}_{new}$
  - ⋆ se il test è negativo, aggiungere $\boldsymbol{q}_{new}$ a T_s (espansione)
- Il processo di generazione di $\boldsymbol{q}_{new}$ risulta automaticamente polarizzato verso regioni di $\mathcal{C}_{free}$ ancora non visitate





- Introduzione di T_g con radice in $\boldsymbol{q}_g$
- Espansione dei due alberi, estendendo l'uno verso l'altro
  - ⋆ usare l'ultima $\boldsymbol{q}_{new}$ generata per T_s come $\boldsymbol{q}_{rand}$ per T_g sino a trovare un cammino libero, altrimenti si invertono i ruoli di T_s e T_g
- Caratteristiche del metodo RRT bidirezionale
  - ⋆ intrinsecamente *single-query*
  - ⋆ *completo in probabilità*
  - ⋆ passo δ variabile (variante *greedy*)
  - ⋆ può essere adattato a estensioni del problema di pianificazione canonico

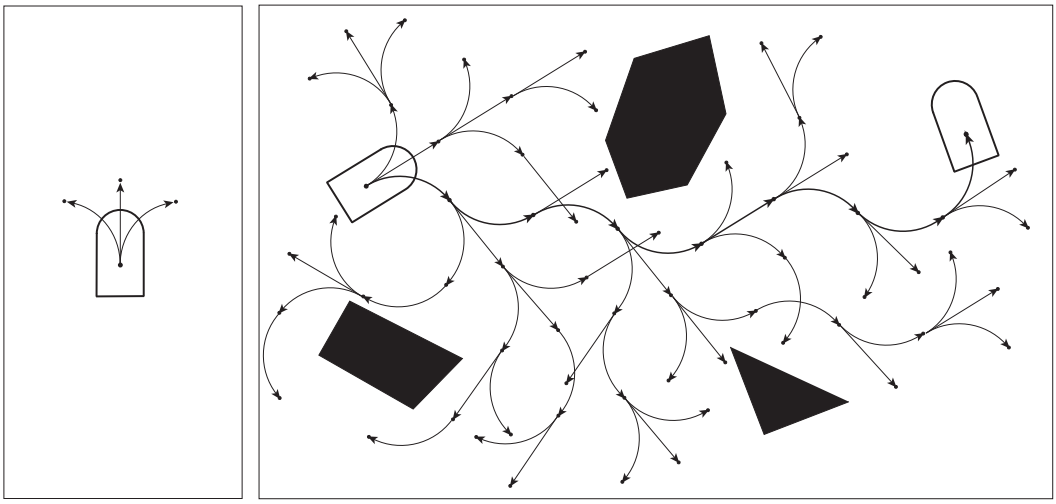
- Estensione ai robot anolonomi
  - ⋆ pianificazione del moto per un uniciclo in $\mathcal{C} = \mathbb{R}^2 \times SO(2)$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega$$

  - ⋆ cammini lineari in $\mathcal{C}$ come quelli impiegati per connettere $\boldsymbol{q}_{near}$ a $\boldsymbol{q}_{rand}$ non sono ammissibili
  - ⋆ è possibile impiegare primitive di moto per una scelta opportuna degli ingressi di velocità (*Dubins car*)

$$v = \bar{v} \qquad \omega = \{-\bar{\omega}, 0, \bar{\omega}\} \qquad t \in [0, \Delta]$$

    cui corrispondono tre cammini elementari



- ⋆ l'algoritmo è lo stesso con l'unica differenza che $\boldsymbol{q}_{new}$ viene generato da $\boldsymbol{q}_{near}$ scegliendo uno dei possibili cammini (in maniera randomizzata o come quello che porta l'uniciclo più vicino a $\boldsymbol{q}_{rand}$)
- ⋆ se è possibile raggiungere $\boldsymbol{q}_g$ da $\boldsymbol{q}_s$ con una concatenazione libera di primitive, la probabilità di trovare una soluzione tende a 1 al tendere del tempo a ∞

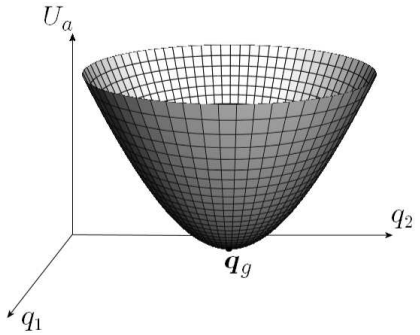
PIANIFICAZIONE MEDIANTE POTENZIALI ARTIFICIALI

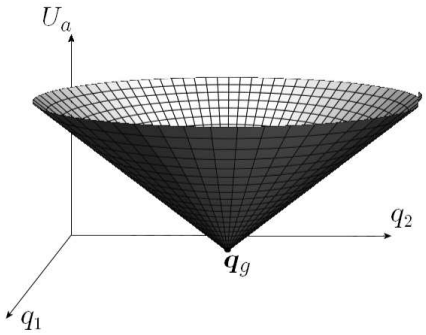
- I robot di servizio devono essere in grado di pianificare il moto *in linea*, utilizzando informazioni parziali sullo spazio di lavoro raccolte attraverso i sensori (caratteristiche di *autonomia*)
  - ⋆ informazioni sensoriali integrate in una mappa secondo un paradigma *sense–plan–move* (navigazione *deliberativa*)
  - ⋆ informazioni sensoriali impiegate per pianificare moti secondo un paradigma *stimulus-response* (navigazione *reattiva*)

- Campi di potenziali artificiali
  - ⋆ il punto che rappresenta il robot viene *attratto* (potenziale U_a) da $\boldsymbol{q}_g$ e *respinto* (potenziale U_r) dalla regione $\mathcal{CO}$
  - ⋆ pianificazione in modo incrementale: a ogni configurazione $\boldsymbol{q}$, la forza artificiale viene generata come $-\nabla(U_a(\boldsymbol{q}) + U_r(\boldsymbol{q}))$

Potenziale attrattivo

- Obiettivo: guidare il robot verso $\boldsymbol{q}_g$





- *Paraboloide*

$$U_{a1}(\boldsymbol{q}) = \frac{1}{2}\, k_a \boldsymbol{e}^T(\boldsymbol{q})\boldsymbol{e}(\boldsymbol{q}) = \frac{1}{2}\, k_a \|\boldsymbol{e}(\boldsymbol{q})\|^2$$

  - ⋆ forza lineare in $\boldsymbol{e} = \boldsymbol{q}_g - \boldsymbol{q}$

$$\boldsymbol{f}_{a1}(\boldsymbol{q}) = -\nabla U_{a1}(\boldsymbol{q}) = k_a \boldsymbol{e}(\boldsymbol{q})$$

- *Conica*

$$U_{a2}(\boldsymbol{q}) = k_a \|\boldsymbol{e}(\boldsymbol{q})\|$$

  - ⋆ forza costante

$$\boldsymbol{f}_{a2}(\boldsymbol{q}) = -\nabla U_{a2}(\boldsymbol{q}) = k_a \frac{\boldsymbol{e}(\boldsymbol{q})}{\|\boldsymbol{e}(\boldsymbol{q})\|}$$

- Soluzione conveniente

$$U_a(\boldsymbol{q}) = \begin{cases} \frac{1}{2}\, k_a \|\boldsymbol{e}(\boldsymbol{q})\|^2 & \|\boldsymbol{e}(\boldsymbol{q})\| \leq \rho \\ k_b \|\boldsymbol{e}(\boldsymbol{q})\| & \|\boldsymbol{e}(\boldsymbol{q})\| > \rho \end{cases}$$

  - ⋆ Continuità di $\boldsymbol{f}_a$ nella transizione

$$k_a \boldsymbol{e}(\boldsymbol{q}) = k_b \, \frac{\boldsymbol{e}(\boldsymbol{q})}{\|\boldsymbol{e}(\boldsymbol{q})\|} \qquad \|\boldsymbol{e}(\boldsymbol{q})\| = \rho$$

ovvero $k_b = \rho \, k_a$

Potenziale repulsivo

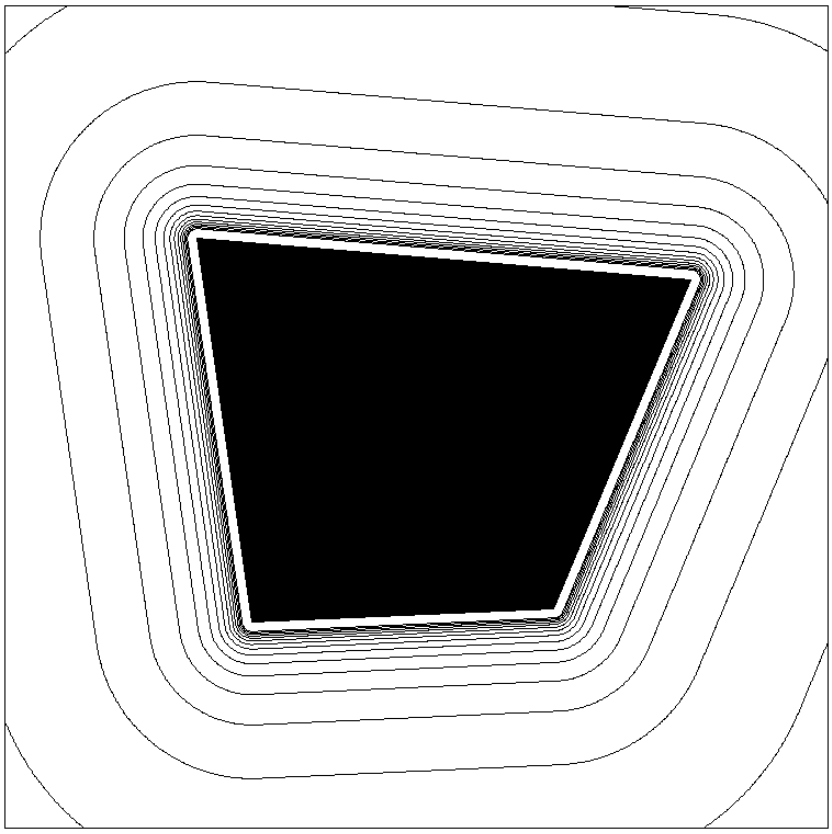
- Obiettivo: tenere il robot lontano da $\mathcal{CO}$
- Ipotesi
  - ⋆ $\mathcal{CO}$ decomposta in componenti convesse $\mathcal{CO}_i$
- Per ogni $\mathcal{CO}_i$, si definisce un potenziale repulsivo

$$U_{r,i}(\boldsymbol{q}) = \begin{cases} \dfrac{k_{r,i}}{\gamma}\left(\dfrac{1}{\eta_i(\boldsymbol{q})} - \dfrac{1}{\eta_{0,i}}\right)^{\gamma} & \eta_i(\boldsymbol{q}) \leq \eta_{0,i} \\ 0 & \eta_i(\boldsymbol{q}) > \eta_{0,i} \end{cases}$$

ove $k_{r,i} > 0$, $\gamma = 2, 3, \ldots$ e

$$\eta_i(\boldsymbol{q}) = \min_{\boldsymbol{q}' \in \mathcal{CO}_i} \|\boldsymbol{q} - \boldsymbol{q}'\|$$

⋆ $\eta_{0,i}$ è il *raggio di influenza* di $\mathcal{CO}_i$



$\star$ $k_r = 1, \gamma = 2$

- Forza repulsiva

$$\begin{aligned}\boldsymbol{f}_{r,i}(\boldsymbol{q}) &= -\nabla U_{r,i}(\boldsymbol{q}) \\ &= \begin{cases} \dfrac{k_{r,i}}{\eta_i^2(\boldsymbol{q})}\left(\dfrac{1}{\eta_i(\boldsymbol{q})}-\dfrac{1}{\eta_{0,i}}\right)^{\gamma-1}\nabla\eta_i(\boldsymbol{q}) & \eta_i(\boldsymbol{q}) \leq \eta_{0,i} \\ 0 & \eta_i(\boldsymbol{q}) > \eta_{0,i} \end{cases}\end{aligned}$$

  - ⋆ $\boldsymbol{f}_{r,i}$ è ortogonale al contorno equipotenziale passante per $\boldsymbol{q}$ e punta nella direzione che si allontana dall'ostacolo
  - ⋆ $\boldsymbol{f}_{r,i}$ è continua ovunque grazie alla decomposizione convessa di $\mathcal{CO}$

- Potenziale repulsivo aggregato di $\mathcal{CO}$

$$U_r(\boldsymbol{q}) = \sum_{i=1}^{p} U_{r,i}(\boldsymbol{q})$$

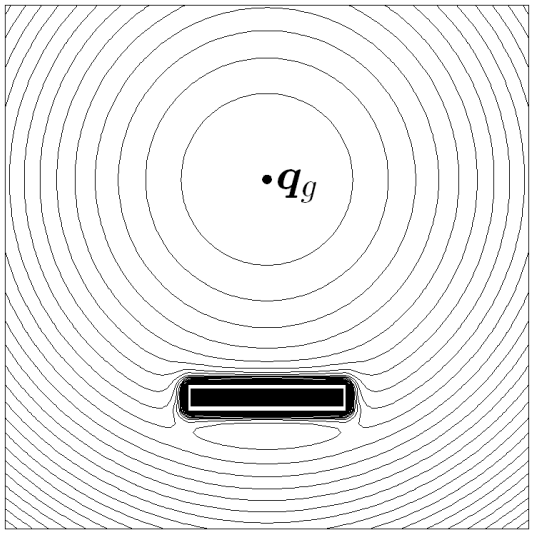
Potenziale totale

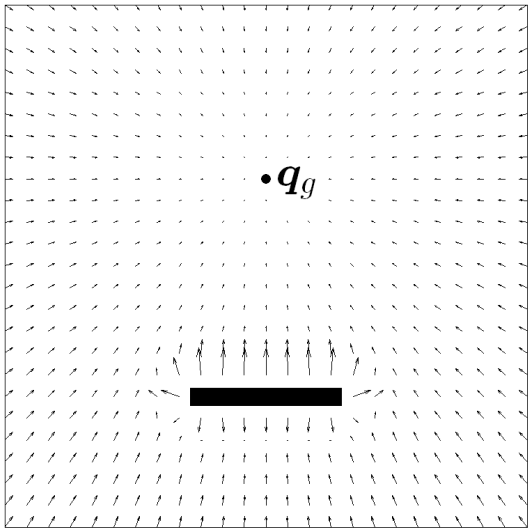
- Sovrapposizione

$$U_t(\boldsymbol{q}) = U_a(\boldsymbol{q}) + U_r(\boldsymbol{q})$$

- Campo di forza

$$\boldsymbol{f}_t(\boldsymbol{q}) = -\nabla U_t(\boldsymbol{q}) = \boldsymbol{f}_a(\boldsymbol{q}) + \sum_{i=1}^{p} \boldsymbol{f}_{r,i}(\boldsymbol{q})$$





⋆ *minimi locali*

Tecniche di pianificazione

1. $\boldsymbol{\tau} = \boldsymbol{f}_t(\boldsymbol{q})$

- ⋆ comandi diretti in coppia al robot

2. $\ddot{\boldsymbol{q}} = \boldsymbol{f}_t(\boldsymbol{q})$

- ⋆ necessario risolvere la dinamica inversa

3. $\dot{\boldsymbol{q}} = \boldsymbol{f}_t(\boldsymbol{q})$

- ⋆ controllo cinematico

- La **1.** genera i movimenti più dolci, mentre la **3.** è più veloce (indipendentemente dalla dinamica del robot) per correggere il moto; la **2.** fornisce risultati intermedi

- Solo la **3.** garantisce (in assenza di minimi locali) la stabilità asintotica di $\boldsymbol{q}_g$; smorzamento in velocità necessario per le **1.** e **2.**

- Pianificazione fuori linea
  - ⋆ cammini generati integrando numericamente ... con riferimento alla **3.** (*algoritmo di massima discesa*)

$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k + T\boldsymbol{f}_t(\boldsymbol{q}_k)$$

Il problema dei minimi locali

- Se un cammino pianificato entra nel bacino di attrazione di un *minimo locale* $\boldsymbol{q}_m$ di U_t, la pianificazione si blocca $\boldsymbol{f}_t(\boldsymbol{q}_m) = -\nabla U_t(\boldsymbol{q}_m) = \mathbf{0}$

- Il potenziale totale presenta inevitabilmente minimi locali $\Longrightarrow$ metodi *non completi* ($\boldsymbol{q}_g$ può non essere raggiunta anche quando una soluzione al problema esiste)

  - $\star$ esistono rimedi, seppure la completezza non sia tipicamente richiesta per la pianificazione in linea

- Algoritmo best-first
  - ⋆ costruire una rappresentazione discretizzata (per difetto) di $\mathcal{C}_{free}$ tramite una griglia regolare e associare a ogni cella libera della cella il valore di U_t al suo punto centrale
  - ⋆ costruire l'albero T con radice in $\boldsymbol{q}_s$: a ogni iterazione, selezionare la foglia di T con il valore minimo di U_t e aggiungere come successori le celle adiacenti a essa che non sono in T
  - ⋆ arrestarsi quando si raggiunge $\boldsymbol{q}_g$ o riportare il fallimento
  - ⋆ nel caso di successo, costruire il cammino soluzione risalendo l'albero da $\boldsymbol{q}_g$ a $\boldsymbol{q}_s$

- Caratteristiche
  - ⋆ evolve come versione discretizzata su griglia del metodo di massima discesa sino a incontrare un minimo locale
  - ⋆ in un minimo locale, "riempie" il bacino di attrazione sino a trovare una via d'uscita
  - ⋆ è completo in risoluzione
  - ⋆ la sua complessità è esponenziale nella dimensione di $\mathcal{C}$ ($n \leq 3$)
  - ⋆ l'efficienza migliora se si alternano passi casuali (*random walks*) alle iterazioni per riempire il bacino (*randomized best-first*)

- Funzioni di navigazione
  - ⋆ i cammini generati dall'algoritmo best-first sono tutt'altro che efficienti (minimi locali non evitati)
  - ⋆ si costruiscono dei potenziali artificiali privi di minimi locali (*funzioni di navigazione*)
  - ⋆ se i $\mathcal{C}$–ostacoli sono *stellati*, si può costruire un diffeomorfismo che trasformi $\mathcal{CO}$ in una collezione di sfere, generare il potenziale nello spazio trasformato, e antitrasformarlo per ottenere un potenziale privo di minimi locali in $\mathcal{C}$
  - ⋆ una possibilità alternativa è quella di definire il potenziale come una *funzione armonica*

- Funzione *numerica* di navigazione

2	1	2	3	4	5	6	7	8	9		19
1	0	1			6	7	8	9	10		18
2	1	2	3		7	8		10	11		17
3		3	4	5	6	7	8		12		16
4			5	6	7			12	13		15
5	6	7	6	7	8	9	10	11	12	13	14
6	7	8	7	8	9	10	11	12	13	14	15

  - ⋆ *wavefront expansion algorithm*

IL CASO DEI ROBOT MANIPOLATORI

- La complessità della pianificazione del moto è notevole, a causa della dimensione elevata dello spazio delle configurazioni ($n \geq 4$)
  - ⋆ manipolatore antropomorfo a 6 gradi di libertà: sostituire il polso sferico (e l'organo terminale) con il volume 'spazzato' (approssimazione per eccesso dell'ingombro)
- La costruzione e la forma di $\mathcal{CO}$ sono complicate dalla presenza dei giunti rotoidali
- Pianificazione fuori linea
  - ⋆ i metodi probabilistici forniscono le prestazioni migliori (test di collisione onerosi)
- Pianificazione in linea
  - ⋆ adattamento del metodo dei potenziali artificiali

Potenziali artificiali

- Per evitare il calcolo di $\mathcal{CO}$ e lavorare in uno spazio a dimensione ridotta, si costruisce il potenziale in $\mathcal{W} = \mathbb{R}^N$ (anziché in $\mathcal{C}$) e viene fatto agire su un insieme di *punti di controllo* distribuiti lungo il robot
  - ⋆ un punto per braccio $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{P-1}$ sui quali agisce il potenziale repulsivo U_r
  - ⋆ un punto sull'organo terminale $\boldsymbol{p}_P$ su cui agisce il potenziale totale $U_t = U_a + U_r$

1. Si impongono ai giunti del robot le forze generalizzate risultanti dai vari campi di forza

$$\boldsymbol{\tau} = -\sum_{i=1}^{P-1} \boldsymbol{J}_i^T(\boldsymbol{q}) \nabla U_r(\boldsymbol{p}_i) - \boldsymbol{J}_P^T(\boldsymbol{q}) \nabla U_t(\boldsymbol{p}_P)$$

⋆ $\boldsymbol{J}_i(\boldsymbol{q}), i = 1, \ldots, P$: Jacobiano relativo al punto di controllo

2. Pianificazione puramente cinematica

$$\dot{\boldsymbol{q}} = -\sum_{i=1}^{P-1} \boldsymbol{J}_i^T(\boldsymbol{q}) \nabla U_r(\boldsymbol{p}_i) - \boldsymbol{J}_P^T(\boldsymbol{q}) \nabla U_t(\boldsymbol{p}_P)$$

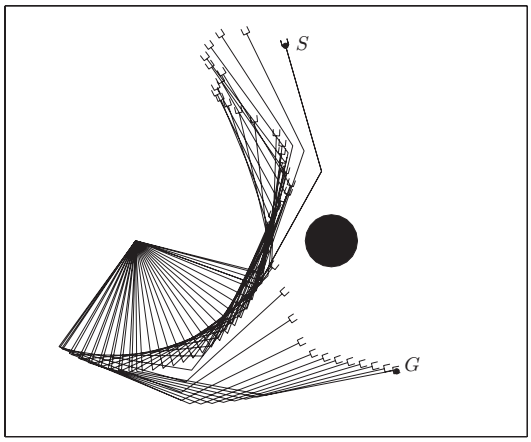
⋆ velocità di giunto come riferimento per gli anelli di controllo di basso livello

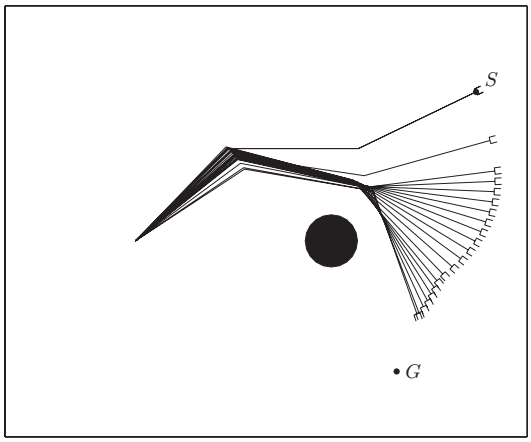
- La **2.** corrisponde a un passo di minimizzazione basata su gradiente di un potenziale combinato in $\mathcal{W}$

$$\nabla_{\boldsymbol{q}} U(\boldsymbol{p}_i) = \left(\frac{\partial U(\boldsymbol{p}_i(\boldsymbol{q}))}{\partial \boldsymbol{q}} \right)^T = \left(\frac{\partial U(\boldsymbol{p}_i)}{\partial \boldsymbol{p}_i} \frac{\partial \boldsymbol{p}_i}{\partial \boldsymbol{q}} \right)^T$$
$$= \boldsymbol{J}_i^T(\boldsymbol{q}) \nabla U(\boldsymbol{p}_i)$$

- La **1.** genera movimenti più dolci, mentre la **2.** è più veloce (indipendentemente dalla dinamica del robot) per effettuare correzioni del moto

- Entrambe possono bloccarsi in corrispondenza di equilibri tra forze puramente repulsive e forze attrattive–repulsive (anche se U_t non è in un minimo locale)





  - ⋆ il metodo dovrebbe essere utilizzato in congiunzione con un algoritmo best-first