Corso di Robotica 2

Pianificazione del moto tra ostacoli

Metodi

Prof. Alessandro De Luca

DIPARTIMENTO DI INFORMATICA
E SISTEMISTICA ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Metodi di pianificazione

quattro grandi categorie di metodi

1. basati su una roadmap (mappa stradale) di C_{free}
   - grafo di visibilità (visibility graph)
   - ritrazione (diagrammi di Voronoi)
2. basati su una decomposizione in celle di C_{free}
   - esatta
   - approssimata
3. probabilistici (randomizzati) per elevate dimensioni
   - PRM
   - RRT bidirezionale
4. basati sull'uso di potenziali artificiali
   - nello spazio delle configurazioni
   - nello spazio di lavoro (per robot articolati)
   - adatti alla pianificazione in linea sensor-based

Valutazione dei metodi

- proposti in origine per il problema base di pianificazione
  - in particolare per $W = R^2$ e oggetti poligonali
- completezza
  - terminazione dell'algoritmo con una soluzione (se esiste) o riportando fallimento
- complessità
  - di spazio, di tempo
- flessibilità
  - possibilità di estendere l'algoritmo a situazioni più complesse
  - in particolare a $W = R^3$ e/o a robot articolati (elevate dimensioni di C)
- non tutti i metodi richiedono una costruzione esplicita dello spazio delle configurazioni libere

Complessità

- valutazione delle prestazioni di un algoritmo di soluzione
  - in condizioni ottimali, uguale alla complessità intrinseca del problema
- in funzione di una misura n della dimensione dell'input
  - n = {dimensione di C, numero dei vertici degli ostacoli, numero dei vincoli polinomiali che definiscono la forma degli ostacoli, …}
- complessità (tipicamente, nel caso peggiore)
  - temporale T(n): tempo necessario per l'esecuzione
  - spaziale S(n): memoria necessaria per l'esecuzione
- interessa il comportamento asintotico (per n grande)
  - O(f(n)) := insieme delle funzioni reali g(n) per cui $\exists\ \alpha$, n_0 tali che

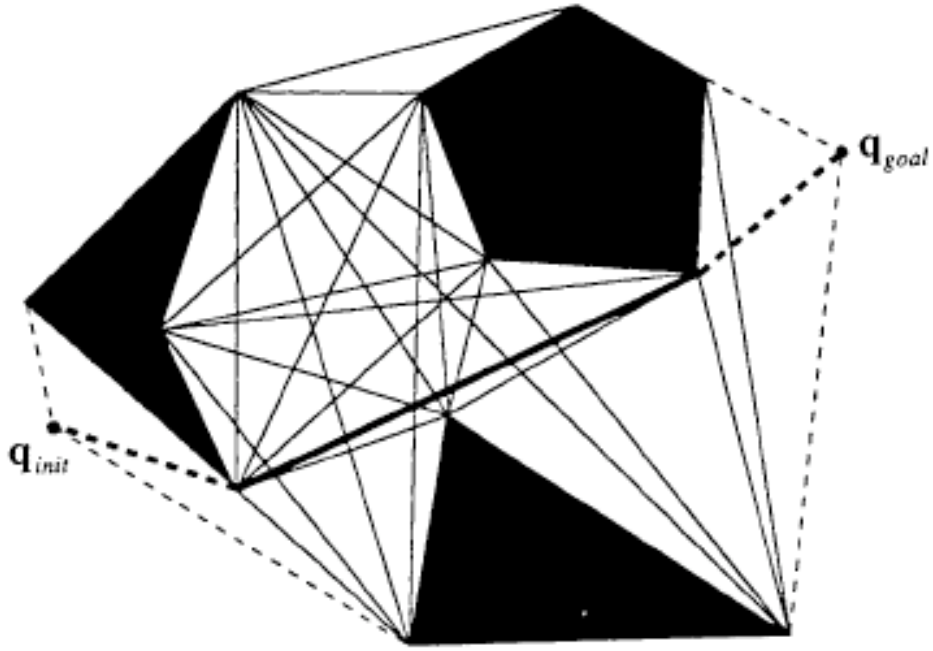
    $g(n) < \alpha f(n)$, per $n > n_0$
- algoritmi in tempo polinomiale [O(n), O(n log n), $O(n^2)$, …] sono realizzabili (anche per n grande) vs. tempo esponenziale …

Grafo di visibilità

- formulazione originale
  - C = R^2, CB regione poligonale
    - robot puntiforme in W = R^2
    - robot poligonale a orientamento costante in W = R^2
- teorema di riferimento
  - esiste un cammino semilibero tra ogni coppia di configurazioni (q_{init}, q_{goal}) se e solo se esiste una spezzata τ contenuta in cl(C_{free}), con estremi in q_{init} e q_{goal} e tale che i vertici di τ siano vertici di CB
- conseguenze
  - è sufficiente considerare solo l'insieme dei segmenti in cl(C_{free}) ("roadmap") che congiungono i vertici di CB ("visibili tra loro")
    - aggiungendo, per ogni "query" specifica, q_{init} e q_{goal} ai vertici
  - tale insieme contiene il cammino a lunghezza (euclidea) minima

Costruzione del Visibility Graph

- grafo VG(N,A) (non orientato) con
  - N = {vertici di CB, q_{init}, q_{goal}}
  - A = {segmenti tra nodi in N, o in C_{free} o lati di CB}



- la maggior parte del grafo si costruisce senza q_{init}, q_{goal} (rimane valido per "multiple queries" successive)
- per una data "query", si aggiungono gli archi visibili a partire da q_{init} e q_{goal}
- soluzione: ricerca di un cammino sul grafo VG(N,A) da q_{init} e q_{goal}

Algoritmo di pianificazione

1. costruzione di VG(N,A)
   - metodo "ingenuo": considerare tutte le coppie di nodi in N e verificare le intersezioni del segmento congiungente con i lati di CB… ⇒ complessità $O(n^3)$
   - con una opportuna struttura dati ⇒ $O(n^2)$
2. ricerca di un cammino minimo su VG(N,A)
   - pesi $c(N_i,N_j)$ degli archi dati dalle distanze euclidee tra i nodi
   - algoritmo di Dijkstra (non informato)
   - algoritmo A* informato dalla euristica $h(N_i)= \|q_i-q_{goal}\|$
     - sottostima sempre il "cost-to-go"
   - complessità $O(k \log n)$

nodi N_i e N_g

n = # vertici di CB, k = # archi di VG

Algoritmo A*

- ogni nodo N_i è memorizzato con una lista di nodi adiacenti $ADJ(N_i)$
- costruzione di un albero T dei cammini minimi (intermedi) correnti
- visita/marcatura e inserimento/prelievo dei nodi in una lista ordinata OPEN
- stima del costo di un cammino che passa per N_i: $f(N_i) = g(N_i) + h(N_i)$
- inizializzazione: T={N_s}, OPEN={N_g}

costo del cammino corrente da N_s a N_i in T

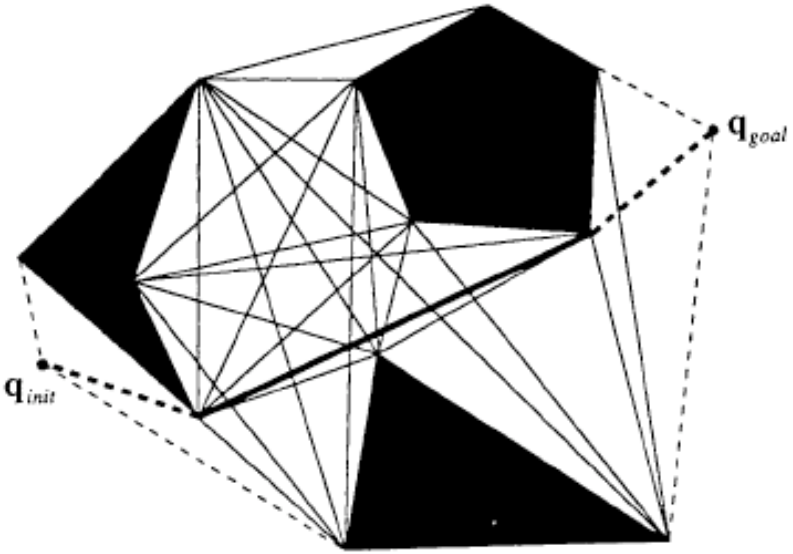
euristica sul cost-to-go

Algoritmo A^*

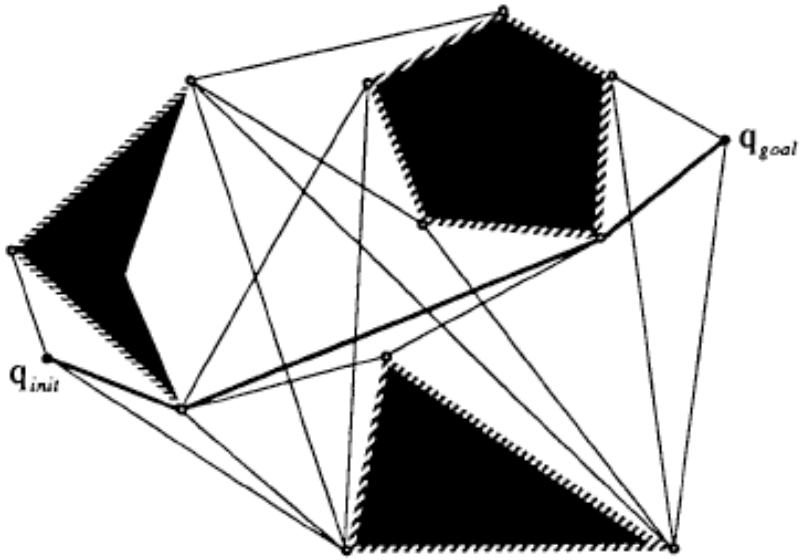
1 **repeat**
2 trova ed estrai il nodo N_{best} in OPEN
3 **if** $N_{\text{best}} = N_g$ **then** esci dal ciclo
4 **for** ogni nodo N_i in $\text{ADJ}(N_{\text{best}})$ **do**
5 **if** N_i è *non visitato* **then**
6 aggiungi N_i a T con un puntatore verso N_{best}
7 inserisci N_i in OPEN; marca N_i *visitato*
8 **else if** $g(N_{\text{best}}) + c(N_{\text{best}}, N_i) < g(N_i)$ **then**
9 ridirigi il puntatore di N_i in T verso N_{best}
10 se N_i non è in OPEN, inseriscilo; altrimenti aggiorna $f(N_i)$
11 **end if**
12 **until** OPEN è vuota

Grafo ridotto di visibilità

- alcuni archi non possono mai far parte di un cammino minimo: conviene eliminarli, riducendo la complessità della ricerca sul grafo
- nel grafo ridotto rimangono solo i segmenti tangenti
  - quelli appartenenti a rette che "non tagliano" i C-ostacoli a cui appartengono i loro estremi (nodi del grafo)



grafo completo

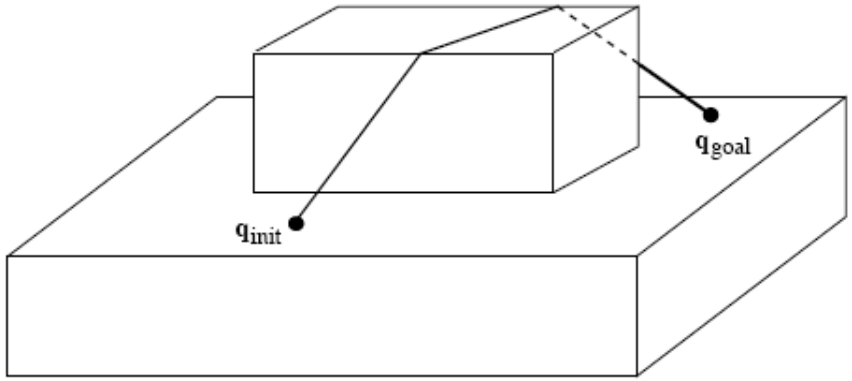


grafo ridotto

Estensioni

- per C-ostacoli in R^2 che sono poligoni generalizzati (con frontiera costituita di tratti rettilinei o circolari)
  - grafo di visibilità generalizzato
- per C-ostacoli poliedrali in R^3
  - si può lavorare in modo conservativo (tagliando eventuali soluzioni) solo con i vertici, ma il cammino trovato con A* non è detto sia minimo
  - i cammini ottimi passano infatti per gli spigoli (infiniti punti!)

    $\Rightarrow$ problema NP-hard



Metodi di ritrazione

- formulazione originale
  - C = R^2, CB regione poligonale
- idea
  - definire un'opportuna applicazione continua: $C_{free} \rightarrow$ R rete mono-dimensionale di curve in C_{free}
- ritrazione (in generale)
  - mappa suriettiva r: $X \rightarrow Y$ (spazi topologici), con $Y \subset X$, tale che $r(Y)=Y$
  - r preserva la connettività di X se, $\forall x \in X$, x e r(x) sono nella stessa componente connessa di X
- teorema di riferimento
  - sia r: $C_{free} \rightarrow$ R una ritrazione che preserva la connettività di C_{free}; esiste un cammino libero tra una coppia di configurazioni (q_{init}, q_{goal}) se e solo se esiste una cammino su R tra $r(q_{init})$ e $r(q_{goal})$

Diagramma di Voronoi

clearance

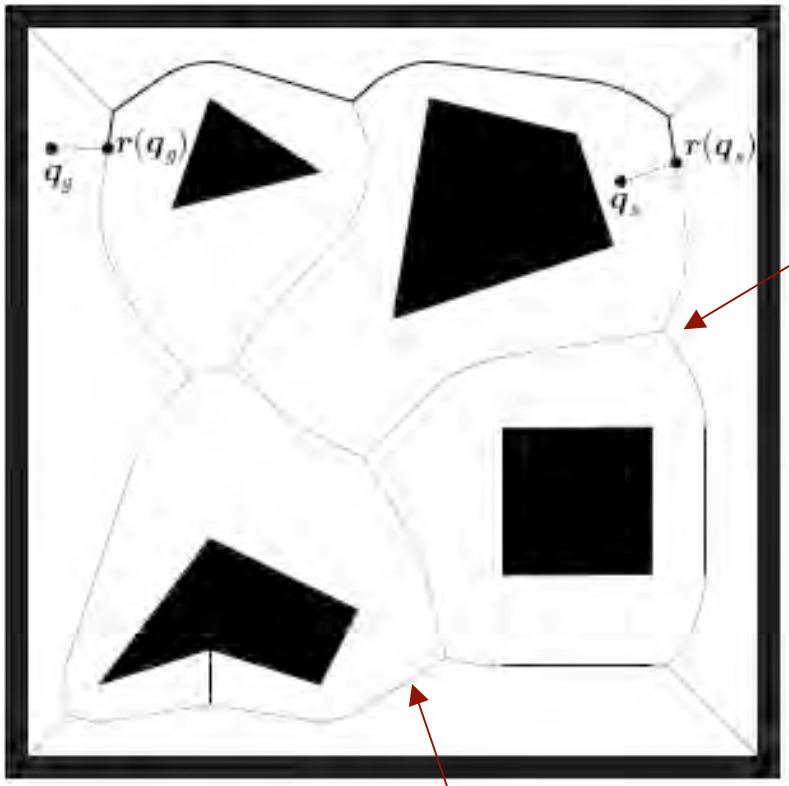
$$\gamma(q) = \min_{s \in \partial \mathcal{C}_{\text{free}}} \|q - s\|$$

insieme dei punti vicini (sulla frontiera di CB)

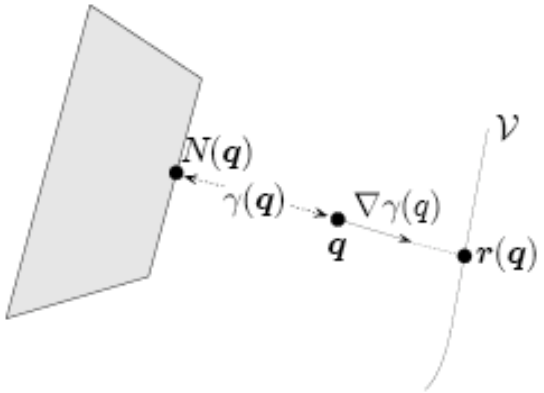
$$N(q) = \{s \in \partial \mathcal{C}_{\text{free}} : \|q - s\| = \gamma(q)\}$$

diagramma (generalizzato) di Voronoi

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



costituito da tratti rettilinei
(lato/lato o vertice/vertice)
o parabolici (lato/vertice)



funzione continua di ritrazione r(q)

rete di cammini a max clearance = più "sicuri"

Algoritmo di pianificazione

1. costruzione di $\mathcal{V}(\mathcal{C}_{\text{free}})$
   - metodo "ingenuo": considerare tutte le coppie (lato/lato), (vertice/vertice), (lato/vertice) --che sono $O(n^2)$, e calcolare l'intersezione di tutte le corrispondenti curve equidistanti ... $\Rightarrow$ complessità $O(n^4)$
   - poiché il # totale di archi di $\mathcal{V}(\mathcal{C}_{\text{free}})$ è O(n) $\Rightarrow$ è possibile scendere a O(n log n)
2. calcolo di $r(q_{init})$ e $r(q_{goal})$
   - usando il gradiente della clearance $\Rightarrow$ complessità O(n)
3. ricerca di un cammino su $\mathcal{V}(\mathcal{C}_{\text{free}})$ tra $r(q_{init})$ e $r(q_{goal})$
   - stesse considerazioni della ricerca sul grafo di visibilità
     - eccetto per la minimalità della lunghezza dei cammini
   - complessità O(n)

Decomposizione in celle

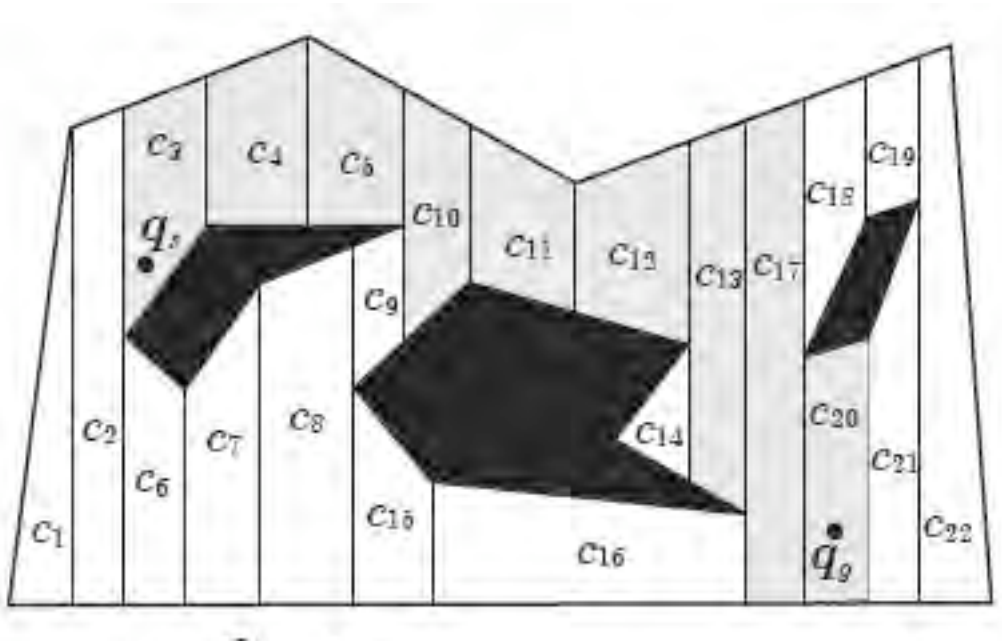
- decomporre C_{free} in regioni semplici (celle)
  - celle convesse: cammini interni lineari sempre ammissibili
  - passaggio tra celle contigue generabile immediatamente
- un grafo di connettività rappresenta la relazione di adiacenza tra celle (nodi del grafo)
- pianificazione di un canale
  - sequenza di celle adiacenti, con q_{init} e q_{goal} nelle due celle estreme
  - dato un canale, trovare un cammino libero è banale (il cammino non è unico, per cui si ha più flessibilità)
- decomposizione esatta
  - unione delle celle = C_{free} , le frontiere corrispondono a criticalità
- decomposizione approssimata
  - C_{free} approssimato (per difetto) da celle di forma predefinita

Decomposizione esatta in celle

- formulazione originale
  - C = R^2, CB regione poligonale
- decomposizione esatta con celle convesse poligonali (senza intersezioni)
  - in modo ottimo (minimo # di poligoni) ⇒ problema NP-completo
  - in modo efficiente con celle trapezoidali ottenute dall'algoritmo sweep-line ⇒ complessità O(n log n)

direzione di sweep

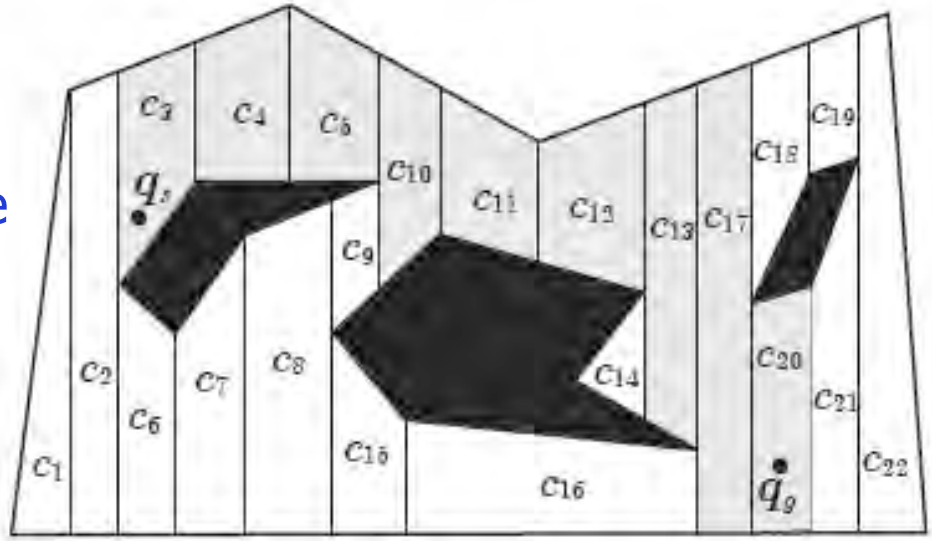
transizioni critiche: ogni volta che la sweep-line incontra un vertice di CB



... un canale di celle adiacenti (in grigio)

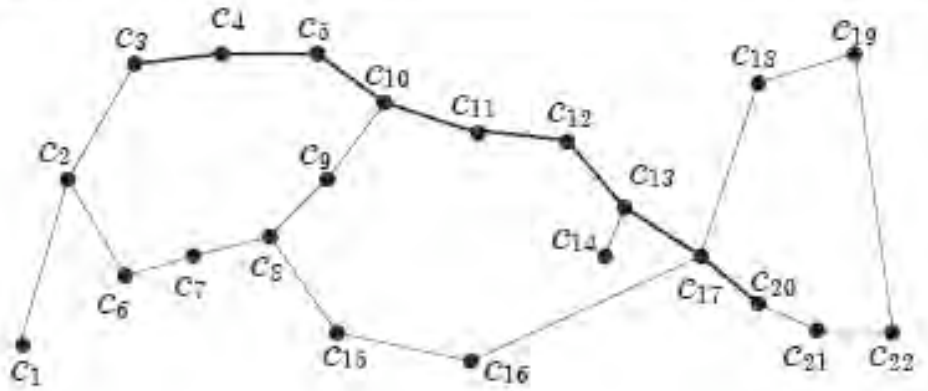
Canali e grafo di connettività

decomposizione esatta con sweep-line



grafo di connettività

nodi = solo le celle per la ricerca di un canale da q_s a q_g



transizione tra celle contigue nel punto intermedio del segmento comune

⇓

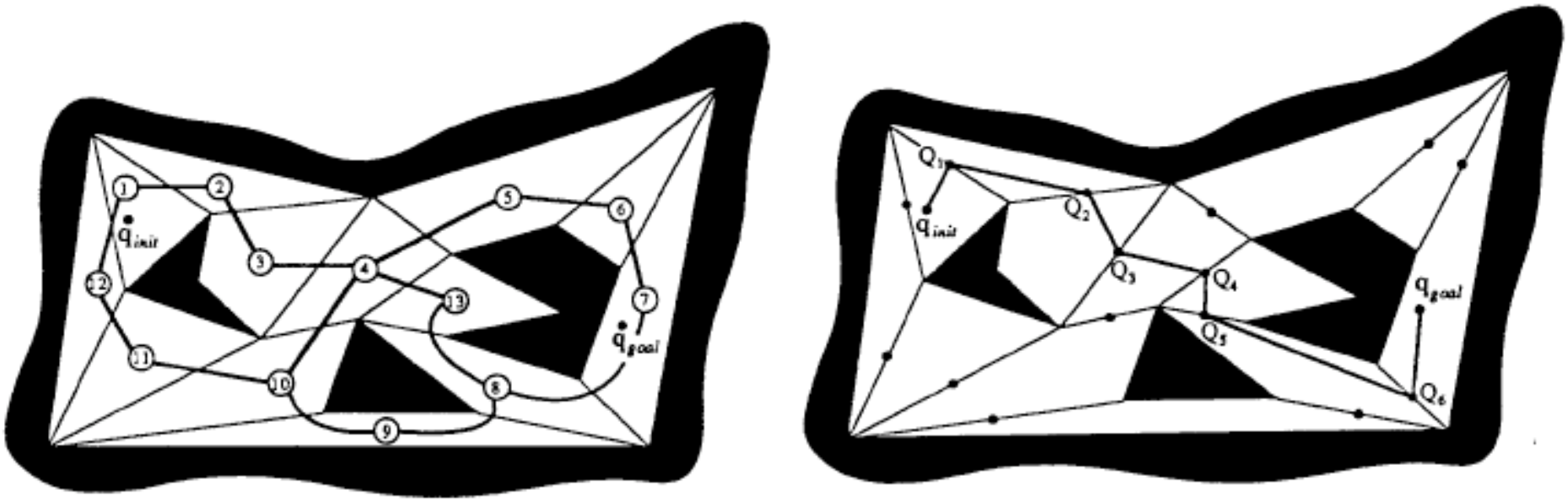
grafo modificato con nodi={centro celle, punti di transizione, q_s, q_g}
pesi archi= distanze tra i nodi

⇓

ricerca del cammino minimo con A*
⇒ O(n log n)

Esempio di pianificazione

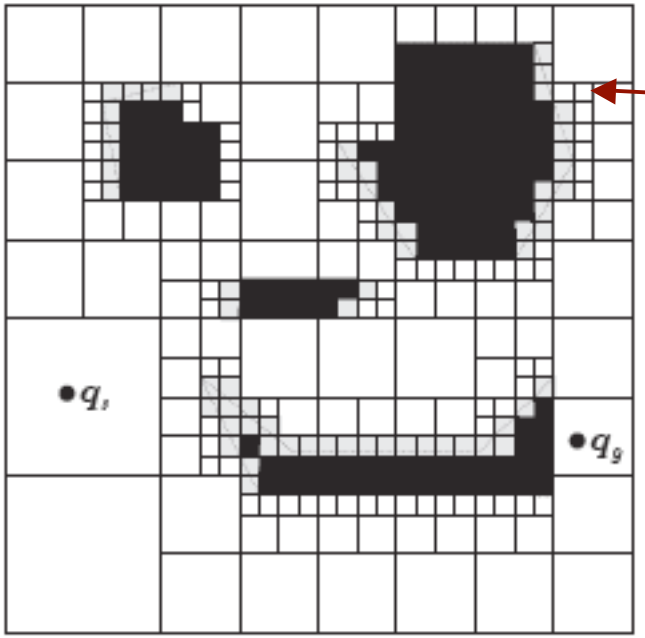
- con una decomposizione ottimale in celle poligonali
  - qui basta inserire nel grafo solo i nodi = {punti di transizione, q_s, q_q}



- il cammino finale può essere approssimato con uno più "smooth" (sfruttando i margini di sicurezza presenti)
- lunghezza finale del cammino: ≈ intermedia tra VG e Voronoi

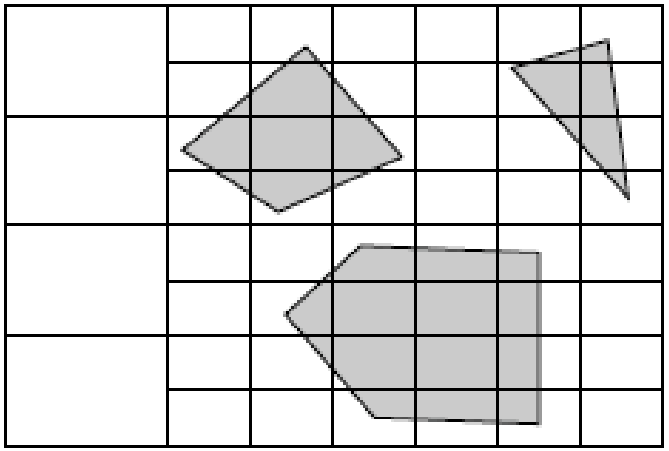
Decomposizione approssimata in celle

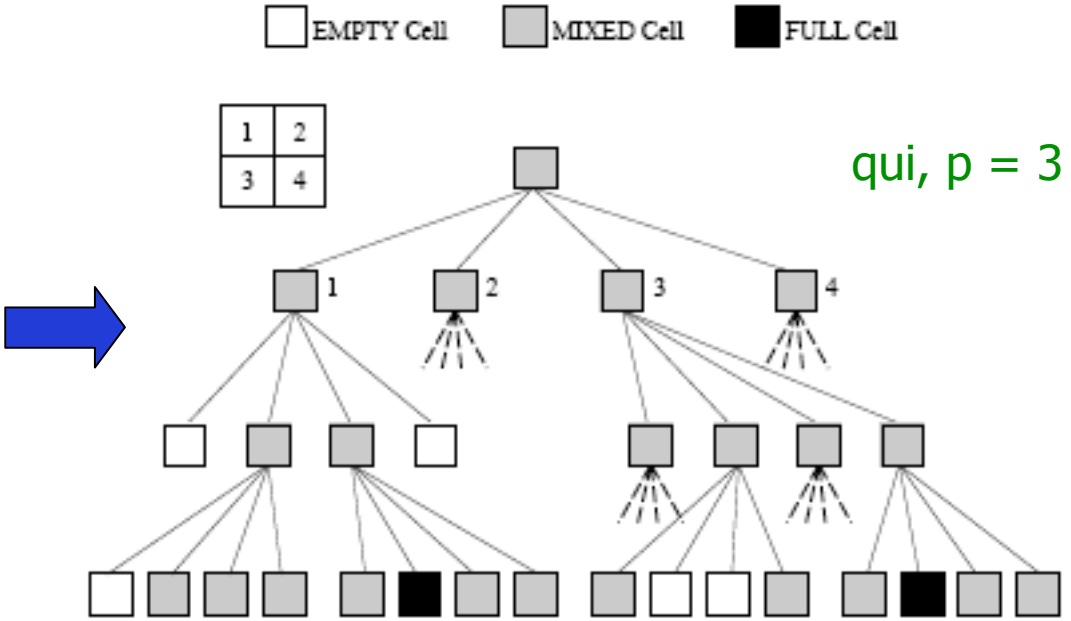
- formulazione originale: C = R^2, ma con regione CB qualsiasi (in principio)
- uso di celle di forma regolare (ipercubi) ma di dimensione non uniforme
- celle libere, occupate, miste (da decomporre in modo ricorsivo)
- struttura dati: quadtree in R^2 = albero di nodi (celle) con quattro foglie
  - la ricerca di cammini su un albero è più rapida
- ad un data profondità di decomposizione: unione celle libere $\subset C_{free}$



Esempio di quadtree

CB = poligoni in C = R^2

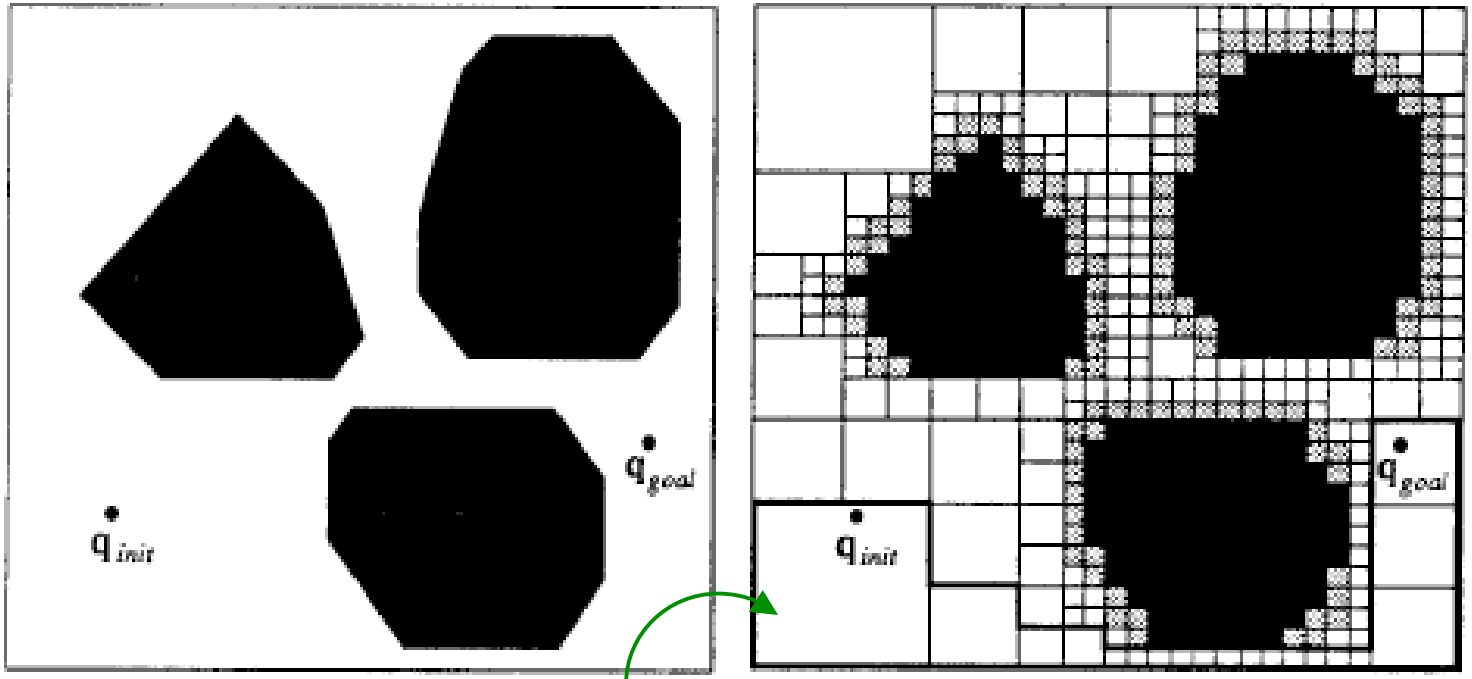




- albero in generale "sbilanciato", con un # di celle $\leq 2^{2p}$, dove p è la massima "profondità" esplorata
  - ci si può fermare ad una dimensione minima prefissata delle celle
  - per una query specifica, quando si trova un cammino di celle libere (empty) tra le due foglie contenenti q_{init} e q_{goal}

Esempio di pianificazione

- qui, n=2 (quadtree) con profondità p = 5



poi come per decomposizione esatta:
canale, grafo di connettività, …

- estendibile a $C=R^n$ (2^n-tree), ma complessità esponenziale (mai n > 4 o 5)
- metodo completo a livello della risoluzione scelta

Confronto tra pianificatori completi

metodo	C base	complessità di tempo	complessità di spazio	estensioni
grafo di visibilità	R^2	$O(n^2)$	$O(n^2)$	CB poligoni generalizzati, in $C = R^3$ con CB poliedri
diagramma di Voronoi generalizzato	R^2	$O(n \log n)$	$O(n)$	CB poligoni generalizzati, Voronoi 3D
decomposizione poligonale esatta	R^2	con sweep-line $O(n \log n)$	$O(n)$	in $C = R^3$ con CB poliedri

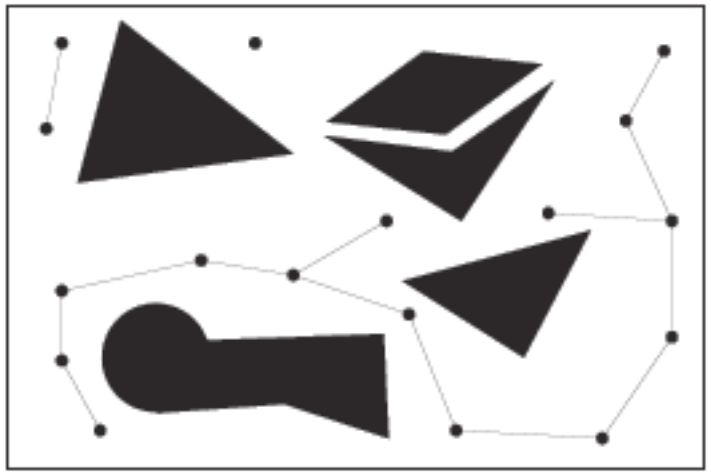
n = # vertici di CB

Metodi probabilistici

- per spazi delle configurazioni C di dimensioni elevate
  - gestiscono bene molti ostacoli, passaggi stretti, ...
- basati sul campionamento di C
  - deterministico su griglia regolare ⇒ non efficiente
  - randomizzato (con eventuale polarizzazione) ⇒ molto efficace!
- un campione scelto da C viene testato rispetto a collisioni
  - il "collision check" avviene sempre nel workspace ...
  - se il campione è ammissibile (ossia è in C_{free}) viene aggiunto ad un set utile, altrimenti è scartato
- si collegano i campioni utili ai precedenti
  - solo se possibile, usando cammini elementari liberi (check!)
  - costruzione incrementale di una roadmap a sufficiente connettività
- una miriade di versioni... (spesso con diverse euristiche)

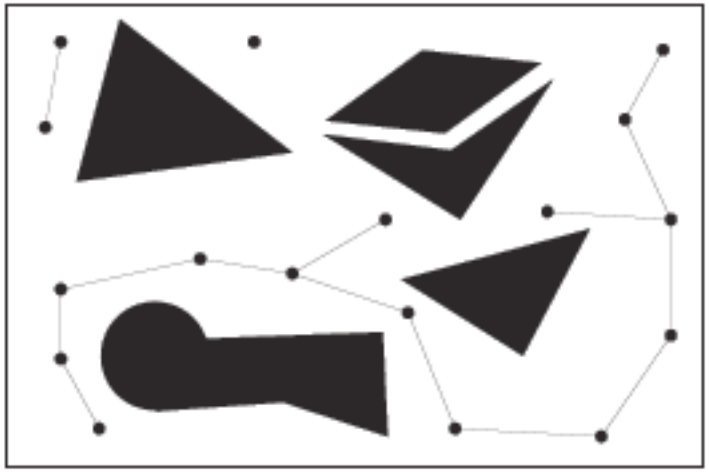
Probabilistic roadmap (PRM)

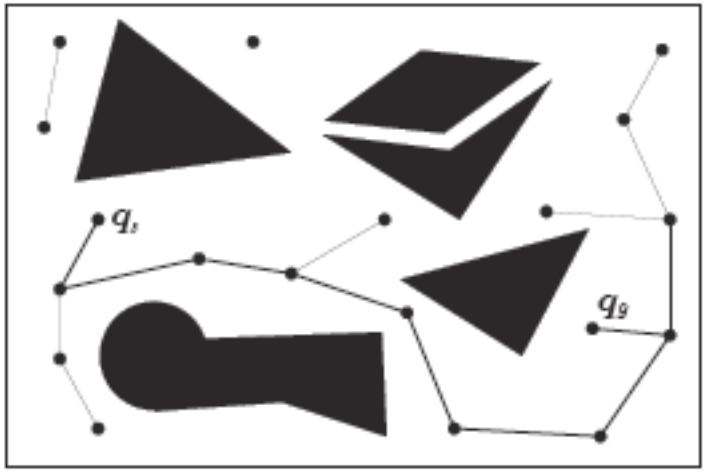
- inizializzazione della roadmap R(N,A)
  - A=∅; N={q_s,q_g} per una single query
  - A=∅; N=∅ prima di una serie di istanze del problema (multiple queries)
- iterazione generica
  - generare q_{rand} con distribuzione di probabilità uniforme in C
  - test di collisione per q_{rand}; se ok, q_{rand}→N; else loop
  - pianificatore locale: test di collisione su cammino elementare (ad es., rettilineo) tra q_{rand} e q_{near} ∈ N; se ok, arco (q_{near},q_{rand})→A; else loop
- criteri di arresto
  - # max iterazioni
  - # componenti connesse di N inferiore a limite assegnato
- varianti
  - distribuzione non uniforme, polarizzando la generazione verso regioni di C con pochi campioni (passaggi stretti)



Pianificazione con PRM

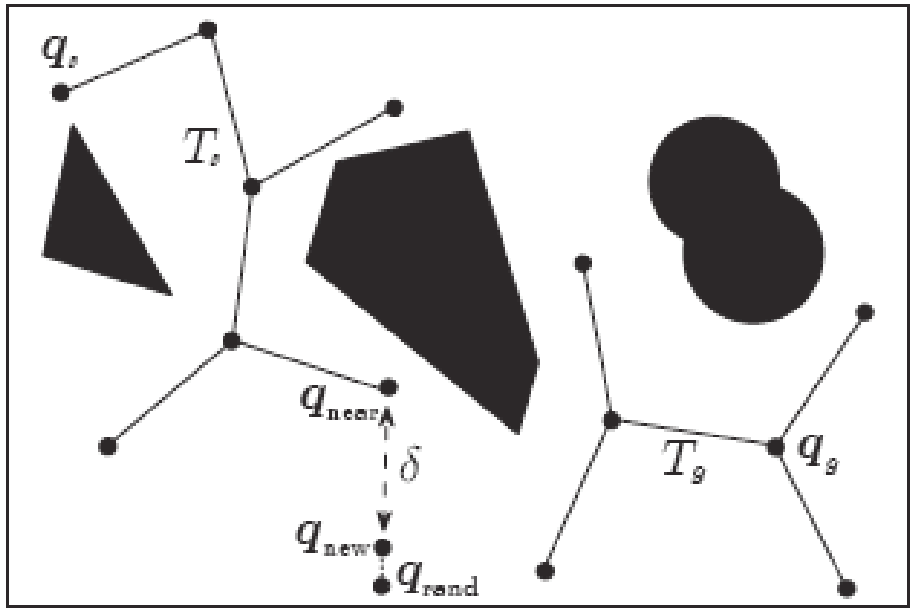
- single query
  - ricerca di un cammino tra q_s e q_g su R(N,A) (devono essere nella stessa componente connessa)
  - se non esiste, si continua ad "arricchire" R (fino a un limite di tempo)
- multiple query
  - come prima, avendo aggiunto e connesso la coppia q_s e q_g a R del caso
  - la roadmap "migliora con l'uso"
- strategia di arricchimento: cercare di connettere componenti diverse





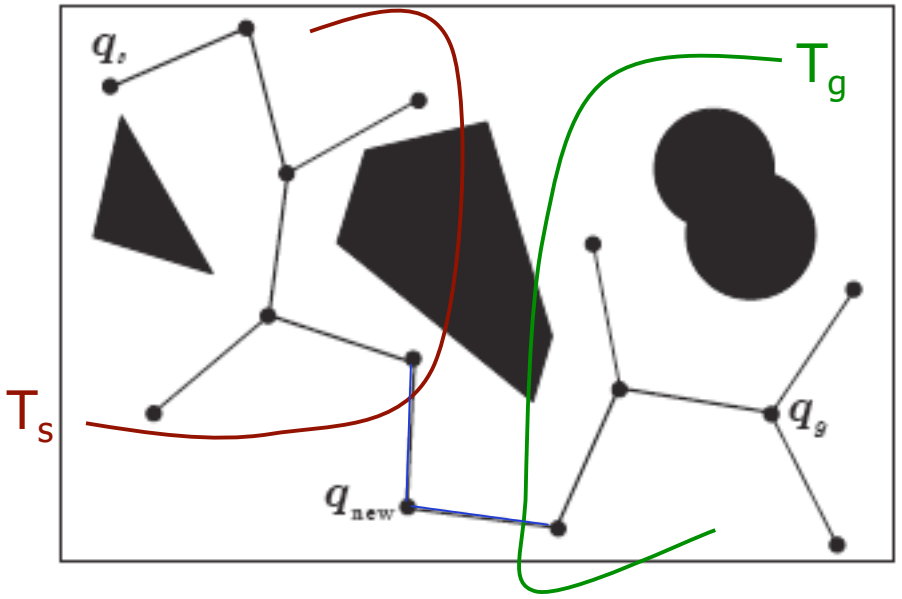
Algoritmo RRT

- Rapidly-exploring Random Tree T(N,A): è un metodo single-query
  - esplora solo le regioni di C utili per il singolo problema considerato
  - genera q_{rand} con distribuzione di probabilità uniforme in C (senza test!)
  - calcola $q_{new} = q_{near} + \delta\, q_{rand}$, con passo $\delta \leq 1$ prefissato (o variabile)
  - test di collisione su q_{new} e sul segmento aperto (q_{near}, q_{new}); se entrambi ok, T è espanso: $q_{new} \rightarrow N$, $(q_{near}, q_{new}) \rightarrow A$; else loop



RRT bidirezionale

- si velocizza la ricerca espandendo due alberi in parallelo
  - T_s (dallo start) e T_g (dal goal)
- si tenta di connettere i due alberi dopo un certo numero di iterazioni
  - test di collisione sul segmento tra q_{new} (generato da T_s) e $q_{near} \in T_g$; se ok, l'espansione connette i due alberi; else, si aggiunge a T_s solo l'eventuale tratto privo di collisioni
  - si ripete il tentativo scambiando i ruoli di T_s e T_g (bidirezionalità)
  - se dopo un certo numero di tentativi non si ottiene connessione, si ritorna alla fase di espansione in parallelo dei due alberi (ancora troppo lontani...)

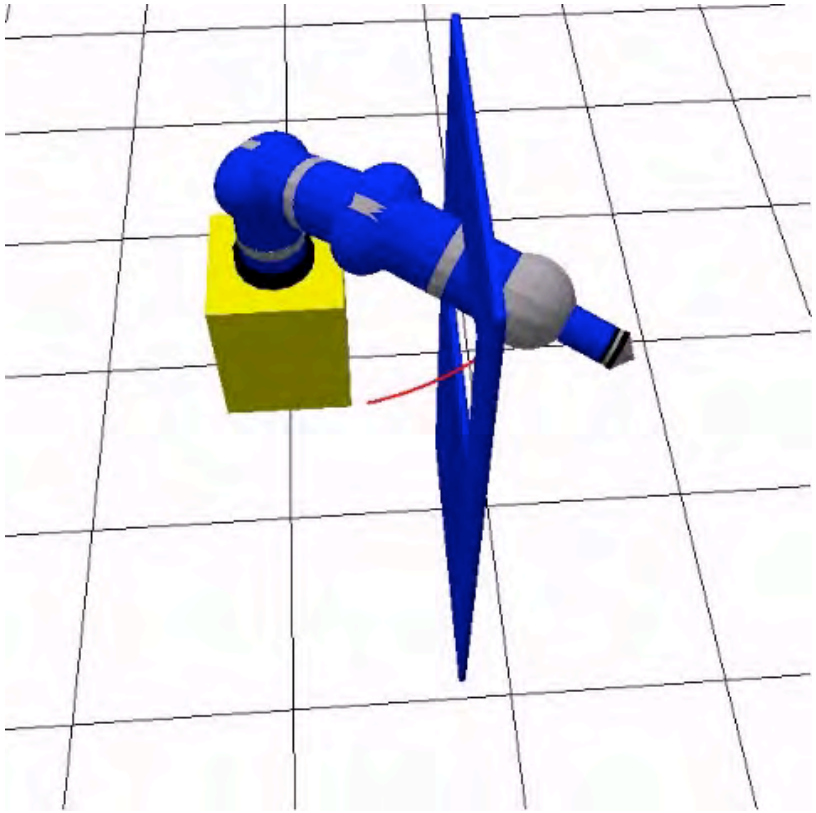


Commenti sui metodi probabilistici

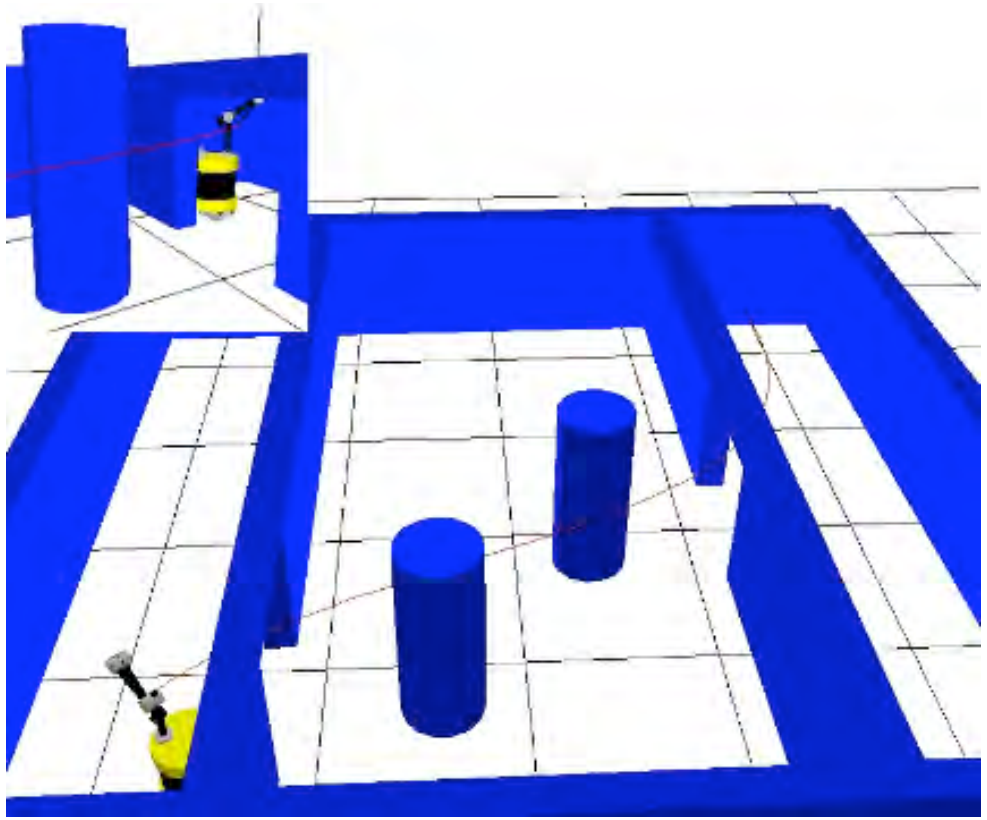
- facili da implementare
- veloci, scalabili
  - vari ordini di grandezza in meno dei metodi precedenti
- completi in senso probabilistico
  - la probabilità che un pianificatore trovi un cammino libero, se esiste, tende a 1 al crescere del tempo di esecuzione
- estendibili al caso di robot con vincoli
  - robot ridondanti rispetto al task (ad es., con organo terminale vincolato a un cammino assegnato)
  - robot mobili su ruote (con vincoli anolonomi da gestire a livello di pianificatore locale)
- adatti anche a task di esplorazione sensor-based di ambienti incogniti (costruzione di W_{free} o C_{free})

Video

pianificazione del moto con vincolo sul cammino dell'organo terminale



robot 7R (DLR-like)

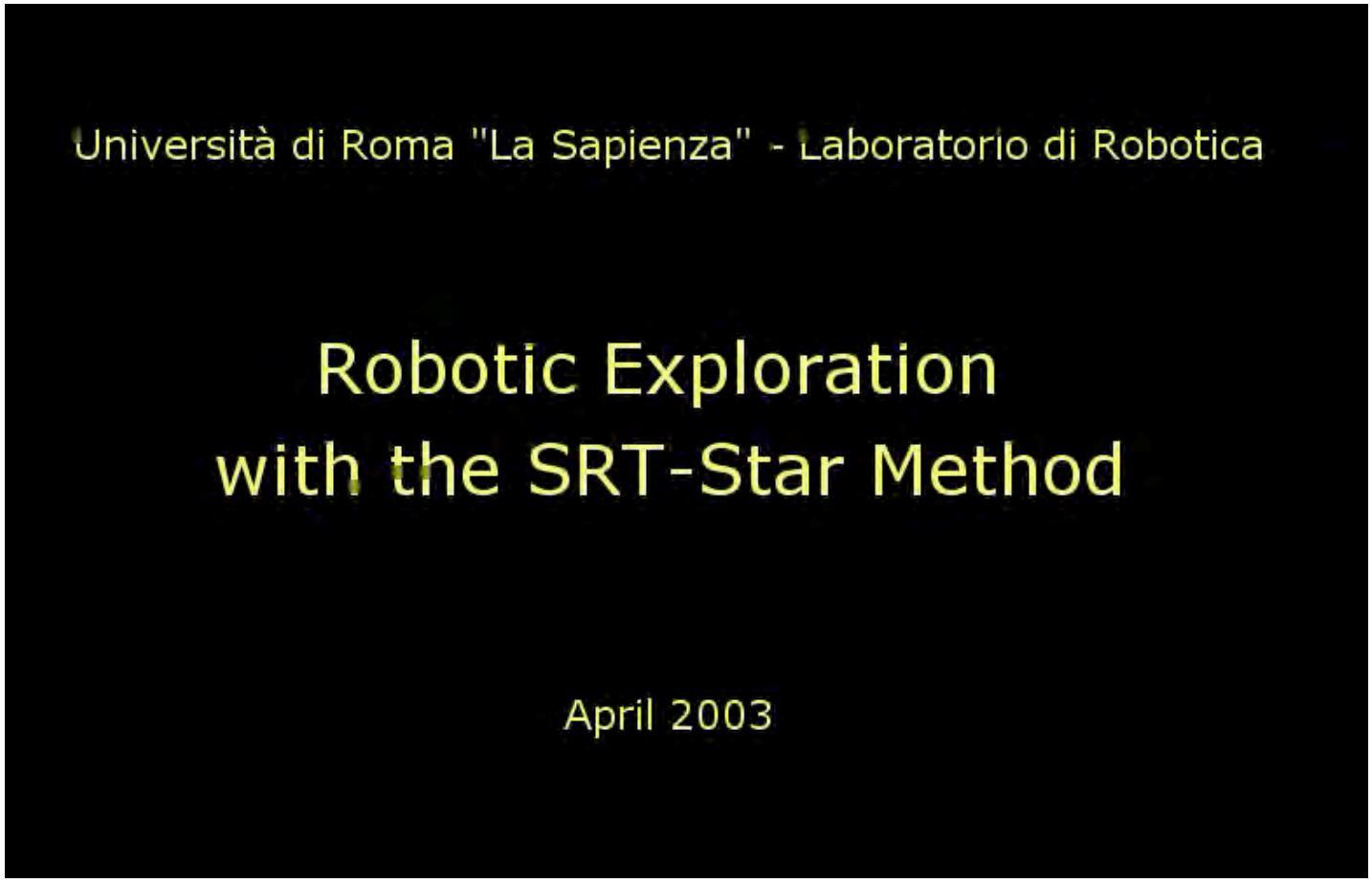


manipolatore mobile
(Nomad + 3R spaziale)

entrambi ridondanti per il task (di dimensione m=3)

Video

esplorazione randomizzata di ambiente ignoto



esperimenti con robot mobile Magellan con 16 ultrasuoni

Metodo dei potenziali artificiali

- i metodi visti finora sono adatti essenzialmente alla pianificazione fuori linea
  - con geometria e posa degli ostacoli nota e fissa
- un paradigma generale per la pianificazione in linea è basato sull'impiego di un insieme di forze artificiali che guidano il moto del robot
  - derivate in genere da un campo di potenziali artificiali
    - forza attrattiva verso il goal, forze repulsive dagli ostacoli
  - per ambienti noti, si usa in modo analogo ai metodi visti
    - l'essenza di C_{free} si riflette nella forma del potenziale totale
    - non è in genere un metodo completo
  - gestisce facilmente informazioni incrementali da sensori esterocettivi (di vario tipo) in ambienti incogniti
    - efficienti in real time e molto usati in pratica!

Potenziali artificiali

attrazione

potenziale attrattivo verso il goal

$$e = q_g - q$$

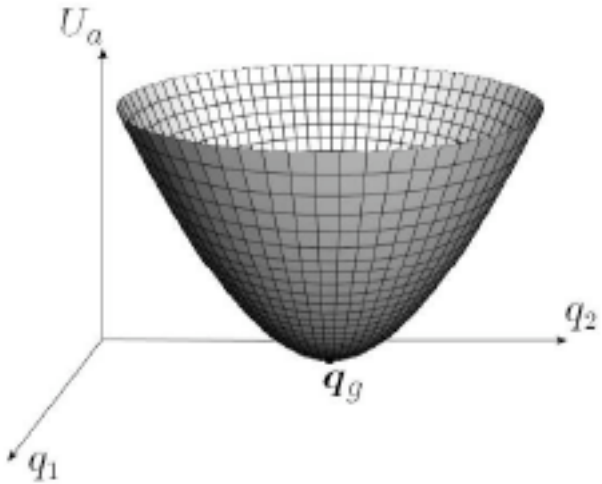
errore (nello spazio C)

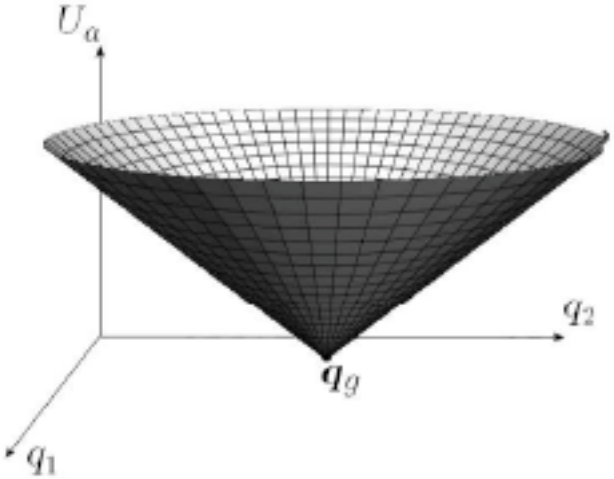
profilo parabolico

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

profilo conico

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





$$f_{a1}(q) = -\nabla U_{a1}(q) = k_a e(q)$$

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

è conveniente raccordare con continuità le forze dal profilo conico (lontano dal goal) a quello parabolico (vicino al goal)

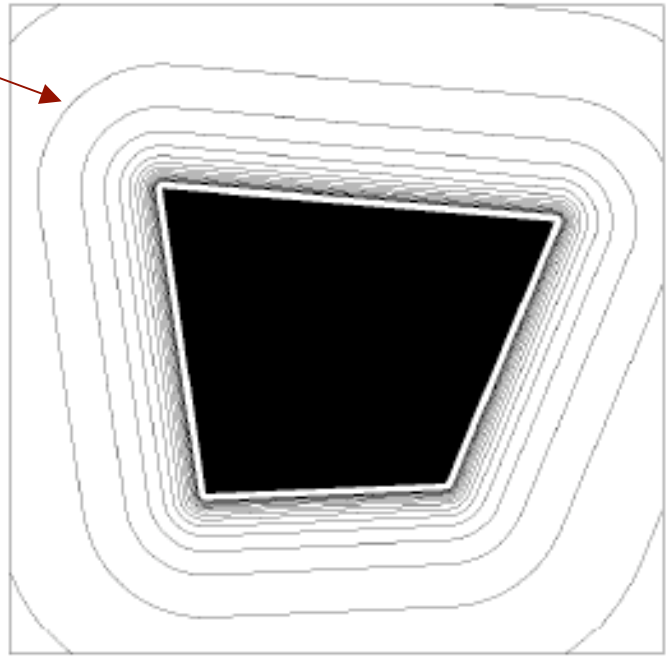
Potenziali artificiali

repulsione

potenziale repulsivo
uno per ogni ostacolo,
se sono tutti convessi;
else, uno per ogni
componente convessa CO_i
della regione CB

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

curve equipotenziali



continuo/differenziabile, con raggio di influenza

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

tipicamente $\gamma = 2$ (profilo iperbolico)
(o interi superiori, che aumentano la
rapidità di salita vicino all'ostacolo)

$$f_{r,i}(q) = -\nabla U_{r,i}(q) = \begin{cases} \dfrac{k_{r,i}}{\eta_i^2(q)}\left(\dfrac{1}{\eta_i(q)} - \dfrac{1}{\eta_{0,i}}\right)^{\gamma-1} \nabla\eta_i(q) & \text{se } \eta_i(q) \le \eta_{0,i} \\ 0 & \text{se } \eta_i(q) > \eta_{0,i} \end{cases}$$

per come
è costruita,
è continua
in tutto C_{free}

nota anche come FIRAS

Potenziale totale

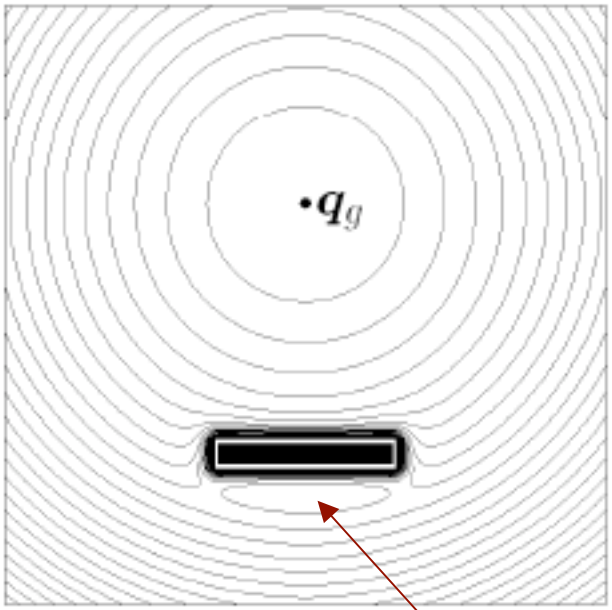
potenziale repulsivo da p ostacoli o loro componenti convesse

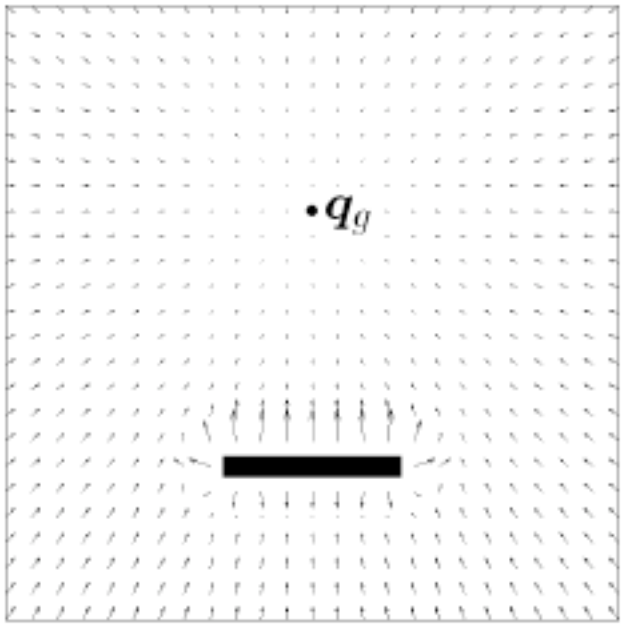
$$U_r(q) = \sum_{i=1}^{p} U_{r,i}(q) \qquad U_t(q) = U_a(q) + U_r(q)$$

potenziale totale

$$f_t(q) = -\nabla U_t(q) = f_a(q) + \sum_{i=1}^{p} f_{r,i}(q)$$

forza artificiale totale (anti-gradiente ossia massima discesa)

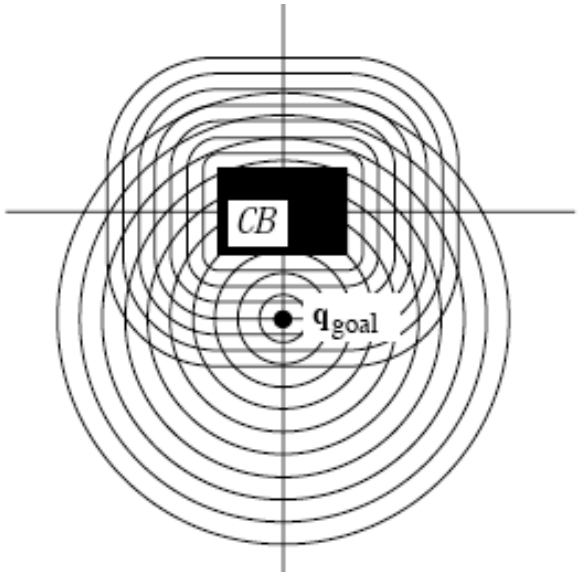


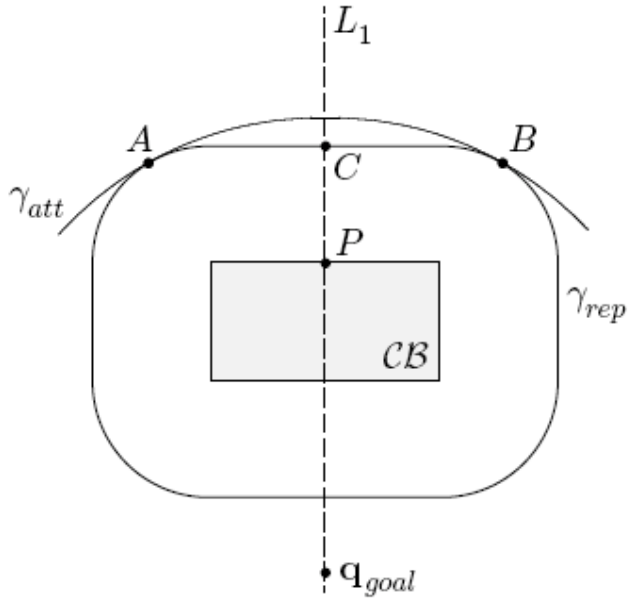


problema dei minimi locali ➡ metodo non completo

Insorgenza di minimi locali

- si ha in presenza di ostacoli con curve di livello del potenziale repulsivo a curvatura inferiore a quella delle curve di livello del potenziale attrattivo (circolari) ⇒ ad es., frontiere di CB lineari





1) lungo L_1P si ha un minimo (attrattivo decrescente, repulsivo crescente)
2) lungo AB si ha un minimo (potenziale totale uguale agli estremi A e B, repulsivo costante, attrattivo decrescente verso il centro C)

Pianificazione con i potenziali

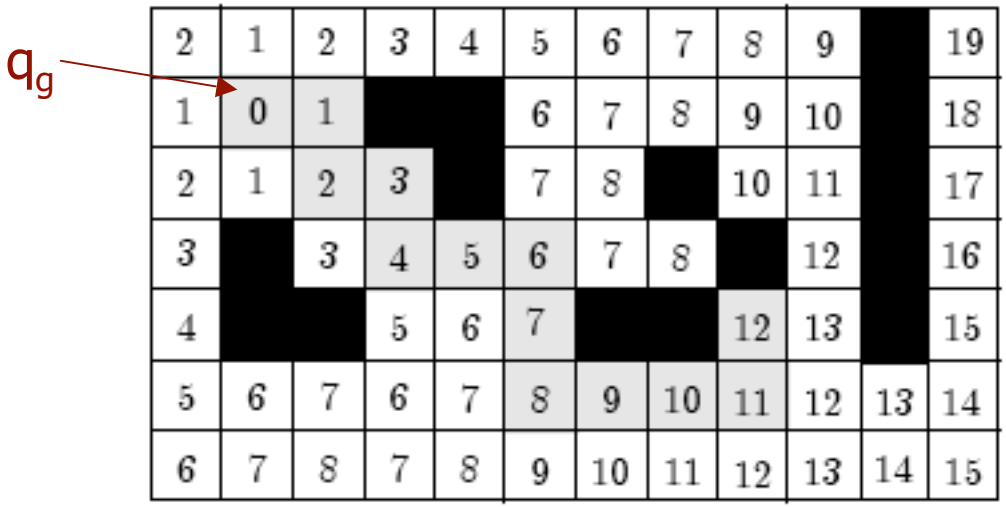
- uso del campo di forze artificiali
  - effettive forze generalizzate (con il modello dinamico): $\tau = f_t(q)$
    - forniscono il moto più fluido
  - accelerazioni (massa costante unitaria): $\ddot{q} = f_t(q)$
  - velocità (cinematico al primo ordine): $\dot{q} = f_t(q)$
    - la più usata, nella forma a tempo discreto $q_{k+1} = q_k + T f_t(q_k)$
- alternative nella pianificazione
  - fuori linea (con ambiente noto)
  - in linea (basato su informazioni sensoriali) in modo reattivo, di fatto uno schema in feedback!
- gestione dei minimi locali (per versione discretizzata del metodo)
  - si procede con un metodo best-first (gradiente discreto) e si "riempiono" le celle del bacino locale e/o si effettuano passi casuali (random walk) per uscire dai minimi

Funzioni di navigazione

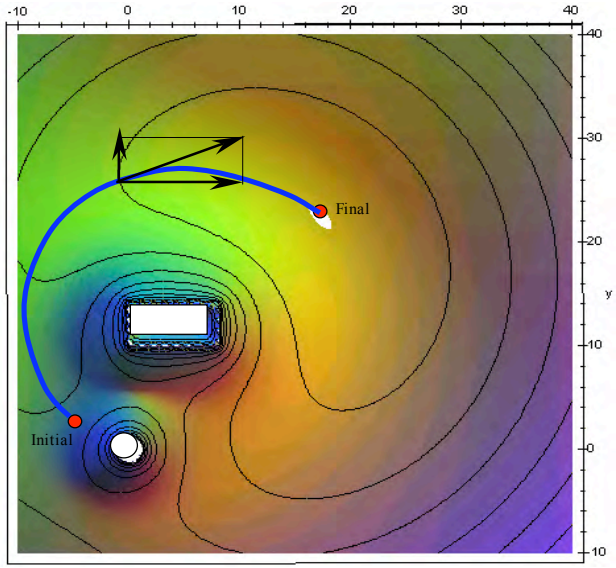
- sono campi potenziali privi di minimi locali
  - il goal (sempre noto a priori) è l'unico minimo = globale
  - ad es., approssimando i C-ostacoli per eccesso con (iper)sfere non si creano minimi (solo punti di sella, numericamente "instabili")
    - comunque non completezza dovuta alla riduzione di C_{free}
- diverse idee (prive di minimi o che riducono il problema)
  - uso di diffeomorfismi (fuori linea)
    - solo per C-ostacoli "stellati", trasformati in ipersfere
  - campi vorticosi in velocità (in linea)
    - al posto della repulsione, una forza tangenziale alle sue curve di isolivello (rimossa una volta superato l'ostacolo)
  - uso di geodesiche sul potenziale artificiale standard (fuori linea)
  - funzioni armoniche (fuori linea)
    - soluzioni di equazioni alle derivate parziali con condizioni al contorno
  - potenziali ellittici repulsivi che diventano presto isotropi/circolari (in linea)

Potenziali artificiali numerici

- versione pratica di funzione di navigazione priva di minimi
  - l'uso di una griglia discreta evita il carico computazionale tipico delle precedenti funzioni esatte di navigazione
  - si procede con una "wavefront expansion" dalla cella di goal, etichettata con 0, incrementando l'etichetta di una unità per le celle adiacenti non ancora etichettate…
  - si usa l'1-adiacenza ("Manhattan-distance")

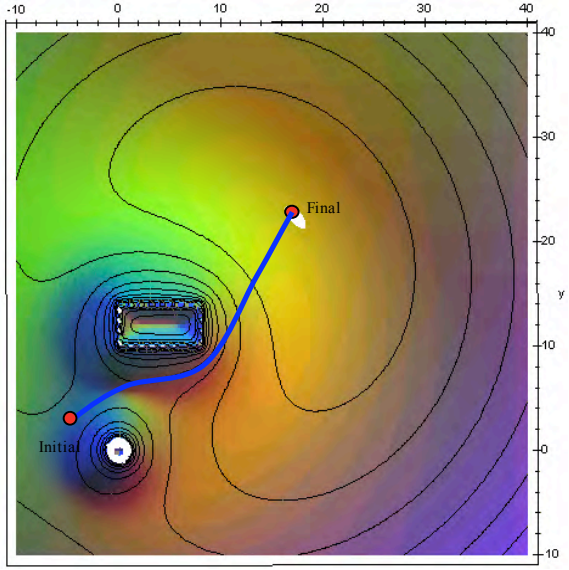


Esempi di altri metodi



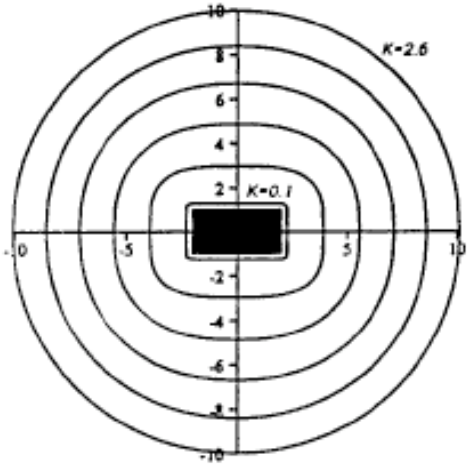
anti-gradiente di U_t

su un potenziale totale standard, cambiando la legge di moto



geodesica su U_t
curva a minima lunghezza nella metrica appropriata

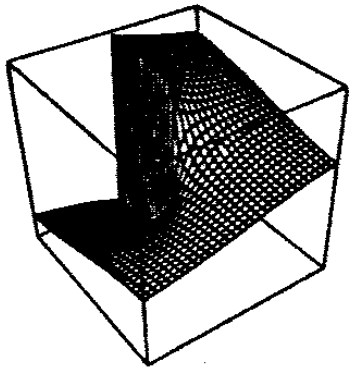
potenziale repulsivo ellittico (superquadric)
indipendente dalla forma dell'ostacolo a sufficiente distanza

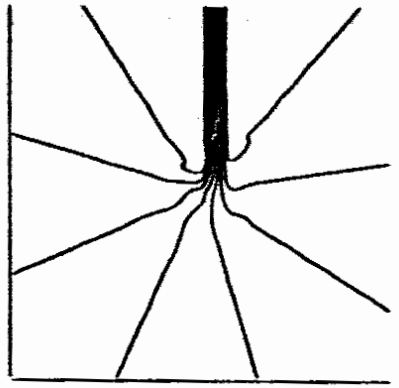


Campi vorticosi

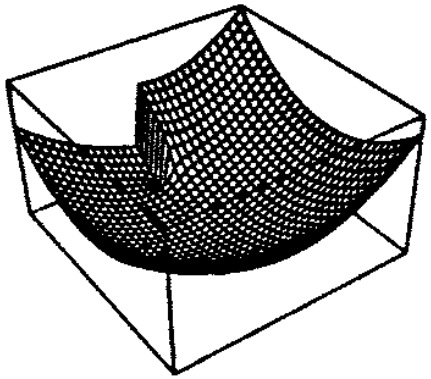
- si sostituisce al gradiente del potenziale ripulsivo, un campo di velocità che aggira l'ostacolo (simile ad un vortice nel moto "ostacolato" di un fluido)
  - per un ostacolo puntiforme nell'origine, il campo vorticoso di velocità (scelto qui con verso CCW) è
  - per ostacoli generali, si può costruire a partire dalle curve equipotenziali repulsive (moto CW/CCW lungo la tangente)

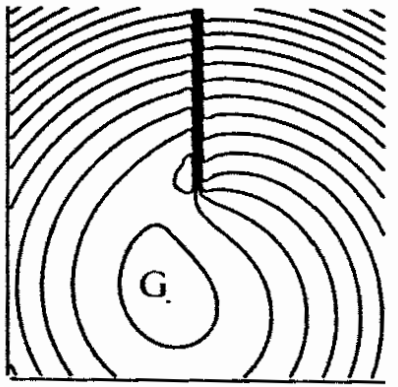
$$V(x,y) = \begin{pmatrix} \dfrac{x}{\sqrt{x^2+y^2}} \\ \dfrac{-y}{\sqrt{x^2+y^2}} \end{pmatrix}$$





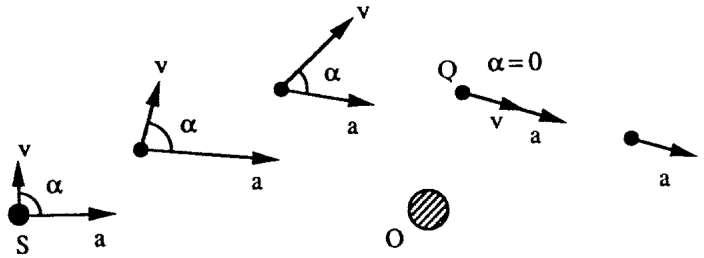
integrale del campo vorticoso di velocità e sue curve di isolivello (qui CW)





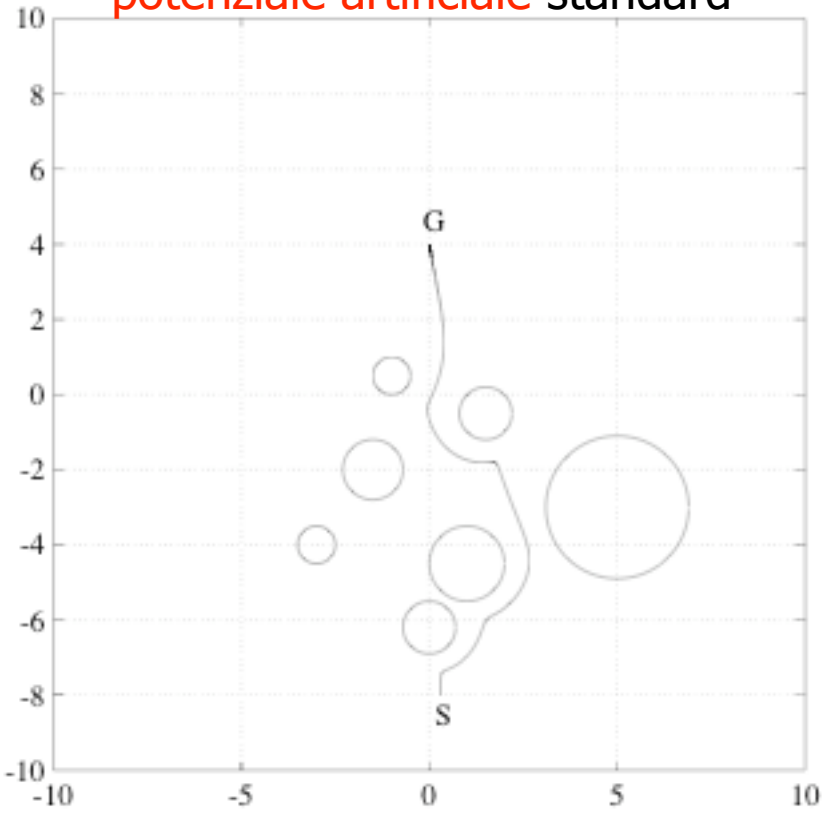
campo totale con attrazione al goal e sue curve di isolivello

strategia "a forbice" per la rimozione del vortice

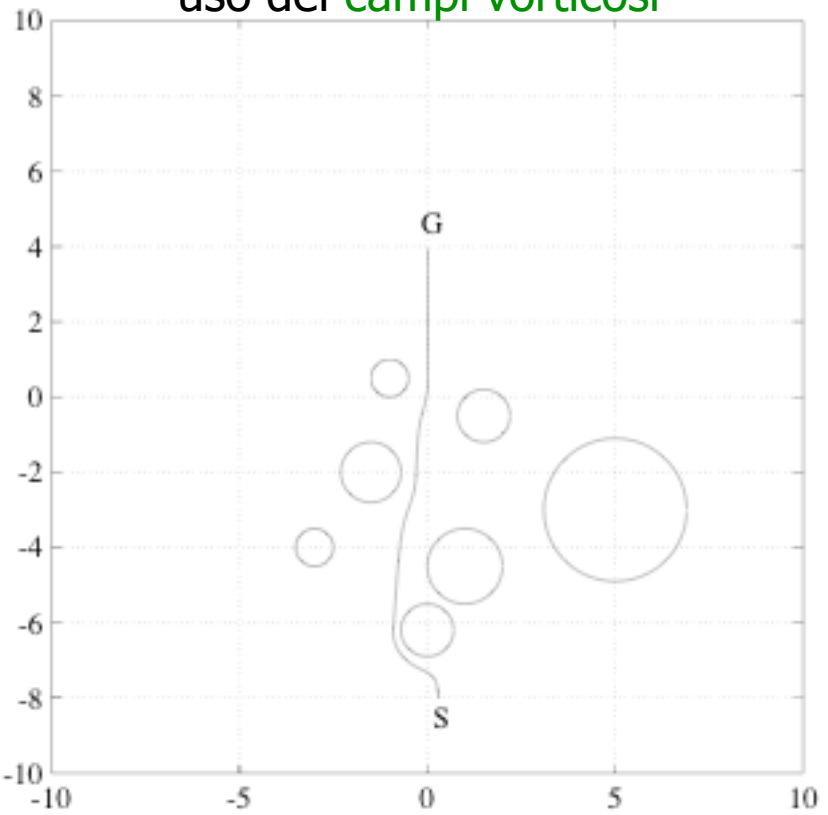


Potenziali vs. Vortici
robot mobile omnidirezionale

potenziale artificiale standard



uso dei campi vorticosi



robot puntiforme in $W=R^2$ o circolare tra ostacoli circolari in $C=R^2$

Potenziali vs. Vortici
robot uniciclo - 1

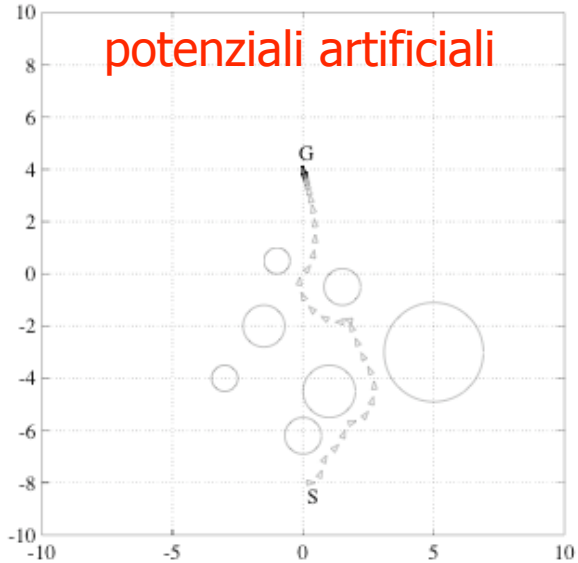
moto diverso per orientamento iniziale diverso: qui, $\theta(0)=0$

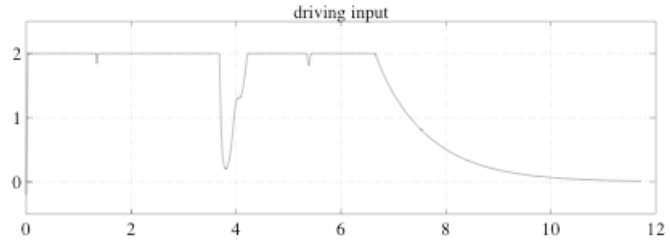
semplice legge di controllo

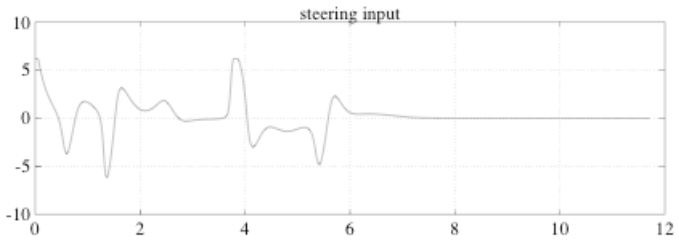
v esegue la proiezione del campo lungo θ

ω allinea il robot lungo il campo

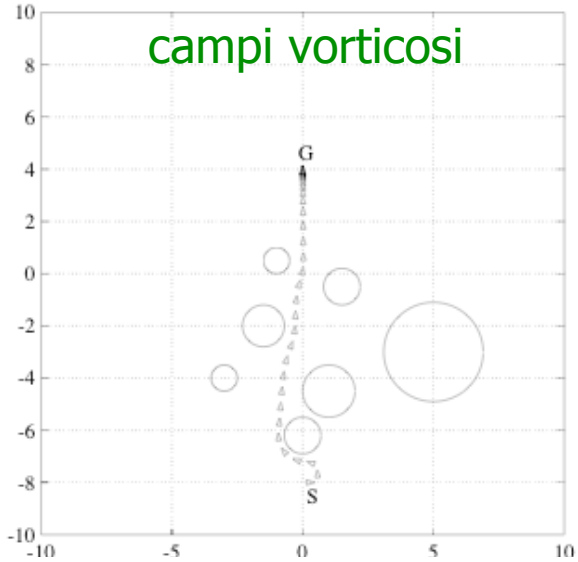
potenziali artificiali

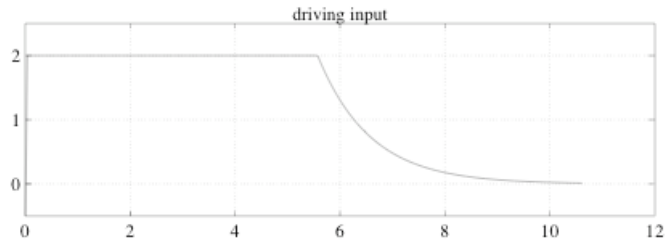


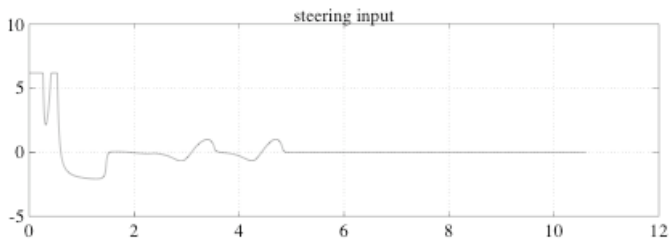




campi vorticosi

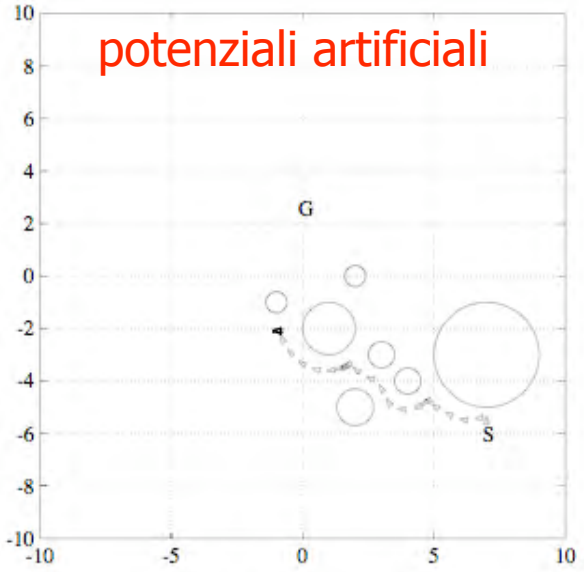


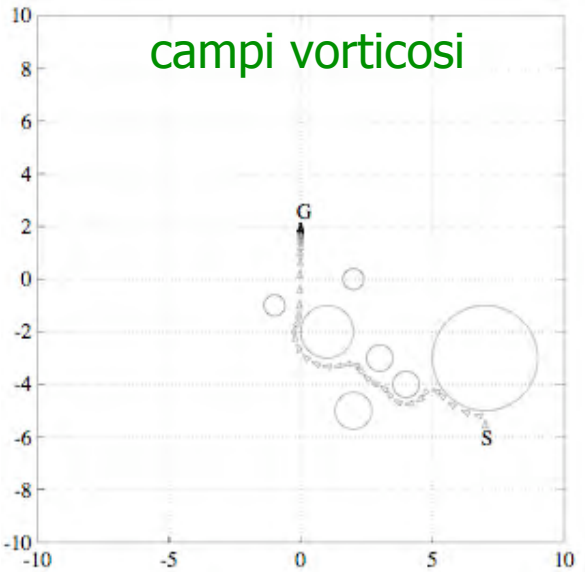




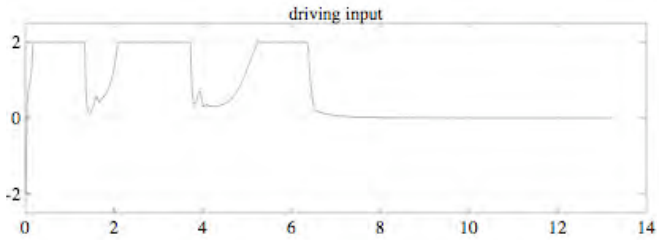
Potenziali vs. Vortici
robot uniciclo - 2

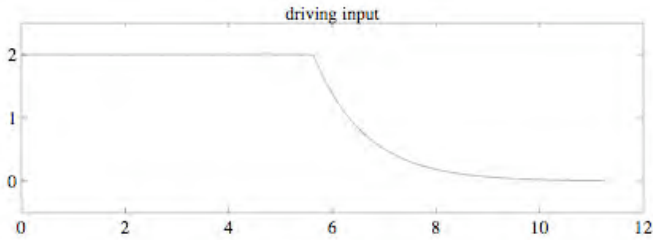
fallisce per un minimo locale

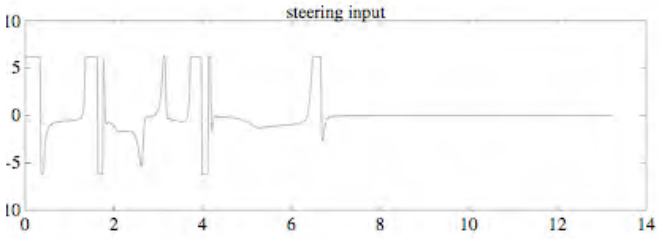


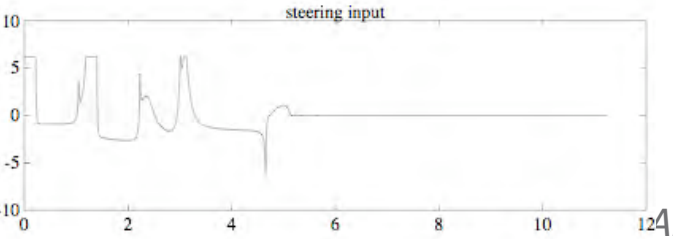


completa il task con successo









Caso dei robot manipolatori

uso P punti di controllo sulla struttura nello spazio di lavoro W

$$\dot{q} = -\sum_{i=1}^{P-1} J_i^T(q)\nabla U_r(p_i) - J_P^T(q)\nabla U_t(p_P)$$

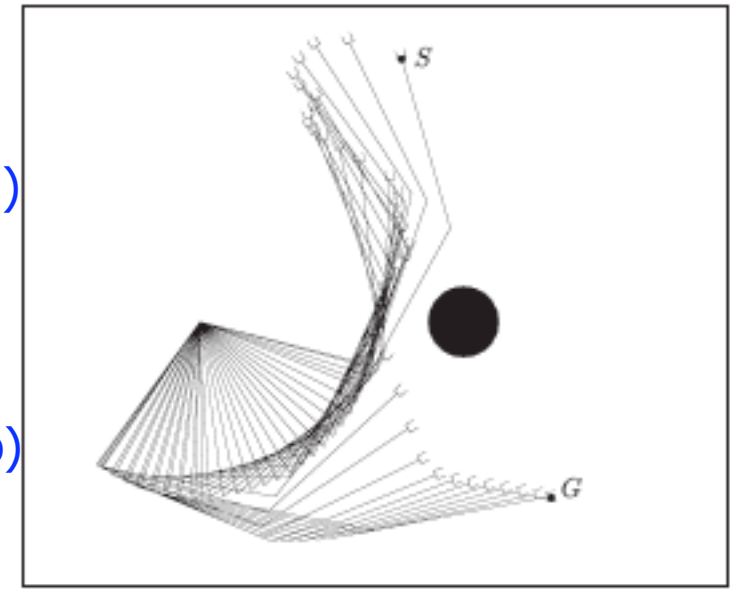
P-1 repulsivi 1 attrattivo (sull'E-E)

Jacobiani dei punti di controllo

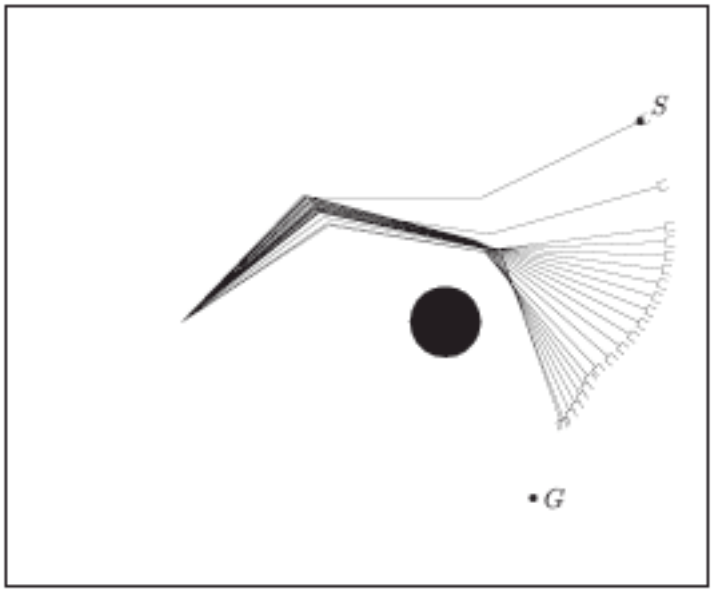
$$\nabla_q U(p_i) = \left(\frac{\partial U(p_i(q))}{\partial q}\right)^T = \left(\frac{\partial U(p_i)}{\partial p_i}\frac{\partial p_i}{\partial q}\right)^T = J_i^T(q)\nabla U(p_i)$$

C=R³ (tre giunti)

ma qui W=R² (nel piano)



successo

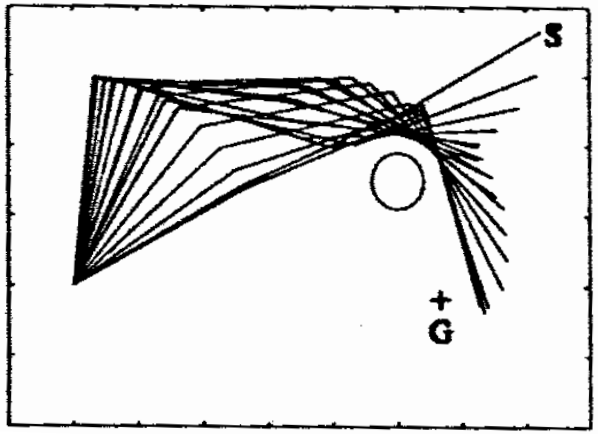


fallimento

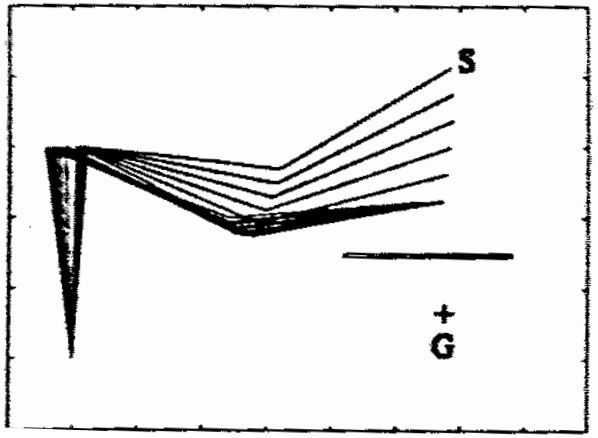
Potenziali vs. Vortici in $W=R^2$

manipolatore planare 3R

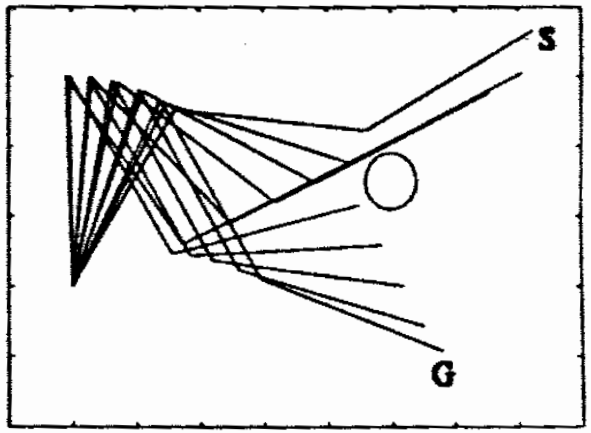
potenziali artificiali



fallimento in entrambi i casi



campi vorticosi



successo (grazie alla scelta CCW: il braccio si "ritrae" verso la base)