Robotica Probabilistica

Navigazione

Navigazione

- Evitare ostacoli (fissi, mobili)
- Pianificare il percorso o la traiettoria
- Esplorare

Path Planning

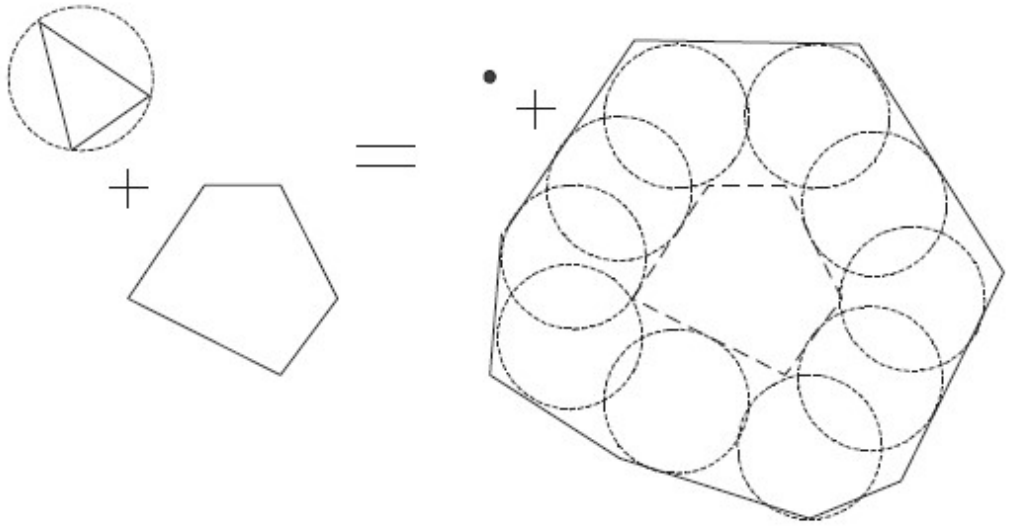
Latombe (1991):

"…eminently necessary since, by definition, a robot accomplishes tasks by moving in the real world."

- Traiettorie prive di collisioni
- Il robot dovrebbe raggiungere la locazione prima possibile

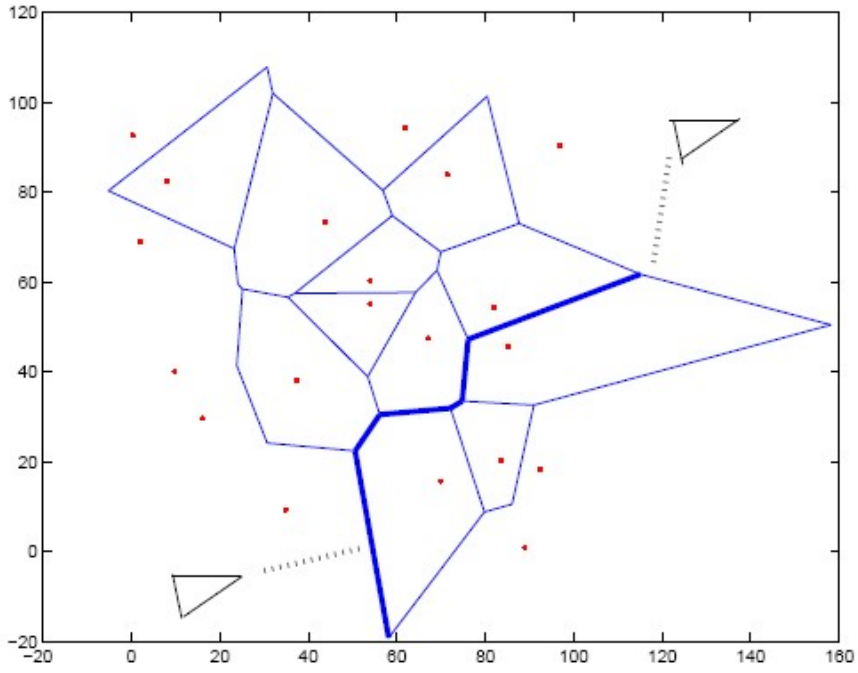
Ostacoli: Minkowski Sum

- Come rappresentare robot e ostacoli
- Forma ostacolo + forma del robot
- Robot approssimato da punto materiale
- Ostacolo aumentato
  - Somma A + B = {a+b | a in A, b in B} con A,B spazi vettoriali



Percorsi tra Ostacoli: Metodi di Voronoi

- Percorsi equidistanti tra i due punti-oggetto più vicini (diagrammi di Voronoi);
- Roadmap a partire dalla mappa:
  - Raggiungere il path vicino,
  - Path verso il goal,
  - Raggiungere il goal.



Percorsi tra Ostacoli: Metodi Bug

- Metodi Voroni richiedono mappa intera, percorsi equidistanti e lunghi.
- Metodo Bug1:
  - Vai da S a T in linea retta;
  - Se c'è ostacolo circumnavigalo finché non trovi punto di uscita.
- Non sempre un buon piano, ma garantito B se raggiungibile.

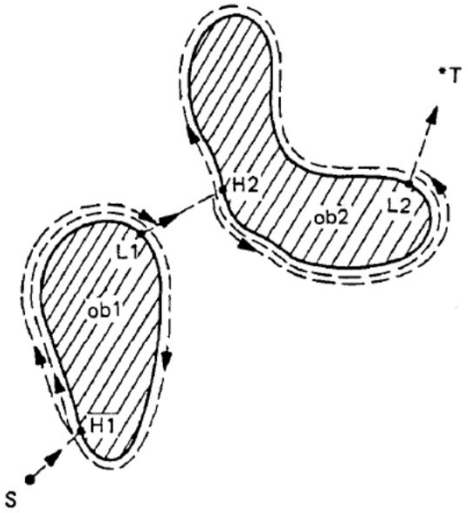
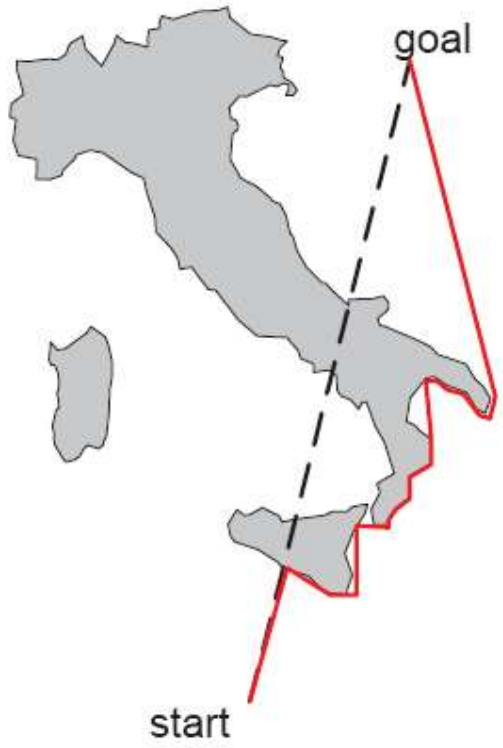


Fig. tratta da [Lumelsky Stepanov, 1987]

Path-Planning Strategies for a Point Mobile Automaton
Moving Amidst Unknown Obstacles of Arbitrary Shape, Lumelsky Stepanov, 1987

Percorsi tra Ostacoli: Metodi Bug

- Metodi voroni richiedono mappa intera, percorsi equidistanti e lunghi.
- Metodo Bug2:
  - Vai da A a B in linea retta;
  - Se c'e' ostacolo circumnavigalo finchè non incontri/vedi la linea AB.
  - Continua a seguire AB
- Migliore piano, nei casi semplici



Path-Planning Strategies for a Point Mobile Automaton
Moving Amidst Unknown Obstacles of Arbitrary Shape, Lumelsky Stepanov, 1987

Percorsi tra Ostacoli: Metodi Potenziali

- Goal basso potenziale, ostacoli alto potenziale;
- Ad ogni punto potenziale:

$$\mathbf{U}_\Sigma(\mathbf{x}) = \sum_{i=1:k} \mathbf{U}_{o,i}(\mathbf{x}) + \mathbf{U}_g(\mathbf{x})$$

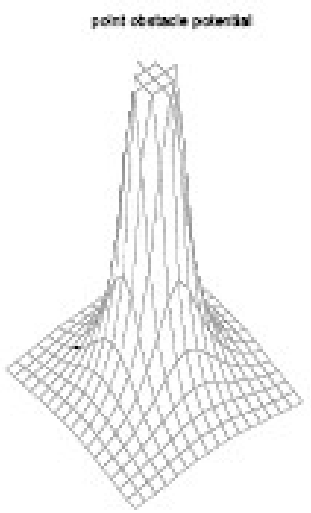
- Ad ogni punto forza:

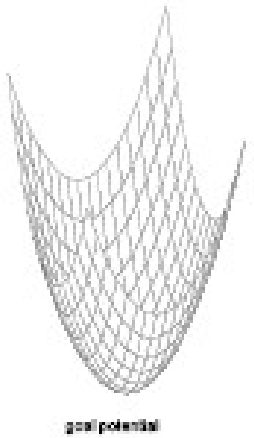
$$\begin{aligned}\mathbf{F}(\mathbf{x}) &= -\nabla \mathbf{U}_\Sigma(\mathbf{x}) \\ &= -\sum_{i=1:k} \nabla \mathbf{U}_{o,i}(\mathbf{x}) - \nabla \mathbf{U}_g(\mathbf{x})\end{aligned}$$

- Quale potenziale?

Percorsi tra Ostacoli: Metodi Potenziali

- Funzioni tipiche:





$$\mathbf{U}_{o,i}(\mathbf{x}) = \eta \begin{cases} \frac{1}{2}\left(\frac{1}{\rho(\mathbf{x})} - \frac{1}{\rho_0}\right)^2 & \forall \quad \rho(\mathbf{x}) \leq \rho_0 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{U}_g(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \mathbf{x}_g)^2$$

- Tra voroni e bug per vicinanza ad ostacoli
- Metodo locale, quindi problema minimi locali

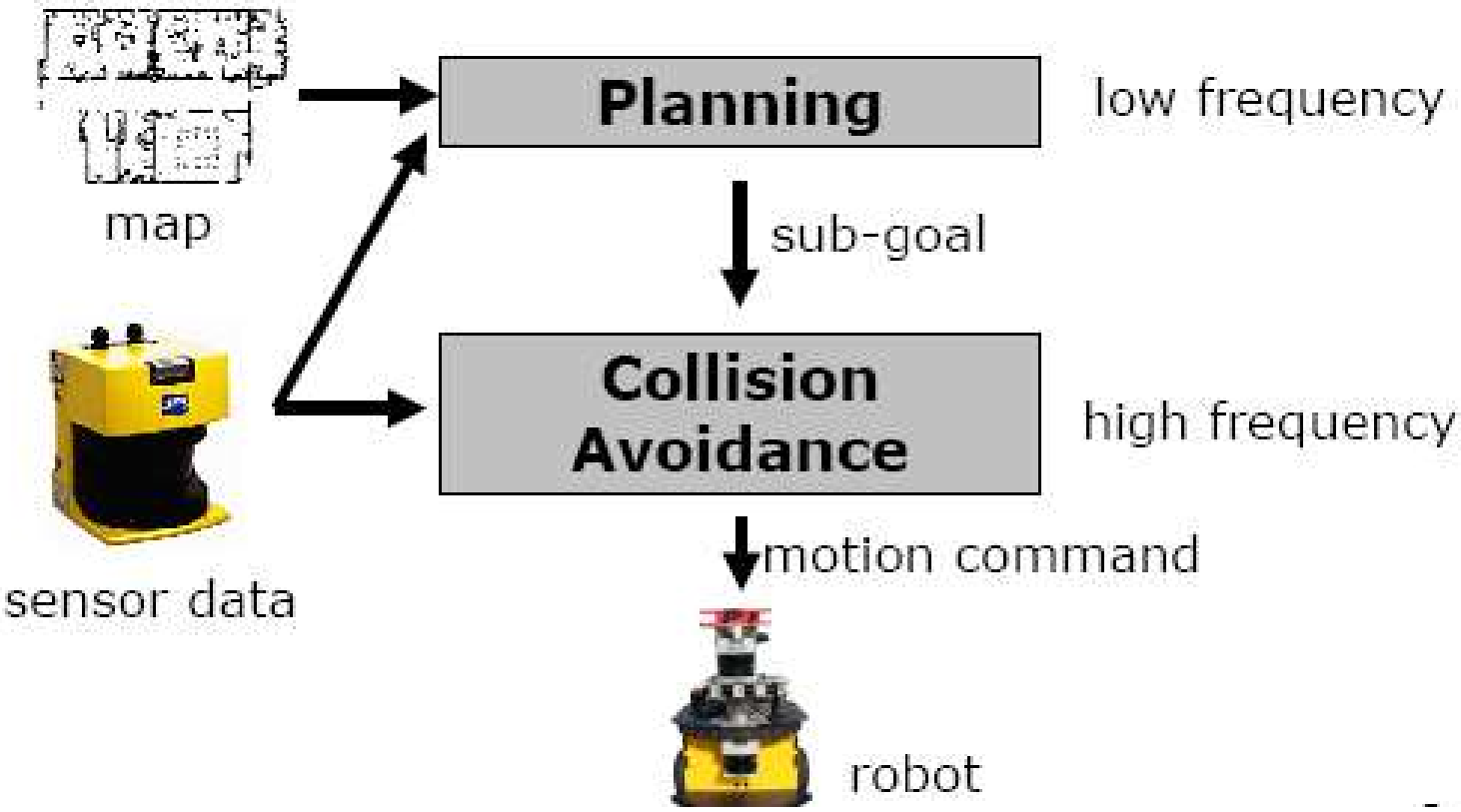
Path planning e Ambiente Dinamico

- Come reagire ad ostacoli imprevisti?
  - Efficacia
  - Affidabilità
- Approcci:
  - Dynamic Window Approaches
    [Simmons, 96], [Fox et al., 97], [Brock & Khatib, 99]
  - Grid map based Planning
    [Konolige, 00]
  - Nearness Diagram Navigation
    [Minguez at al., 2001, 2002]
  - Vector-Field-Histogram+
    [Ulrich & Borenstein, 98]
  - A*, D*, D* Lite, ARA*, …

Obiettivi in ambiente dinamico

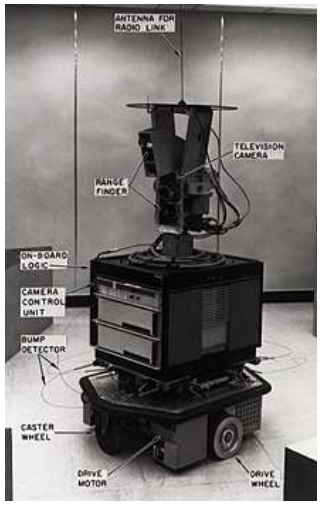
- Calcolo del path ottimo considerando incertezze potenziali nelle azioni
- Generare azioni rapidamente in caso di ostacoli imprevisti

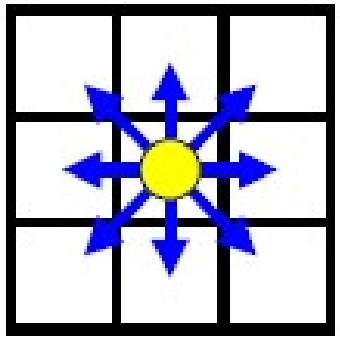
Approccio Classico



Path Planning su grid map

- What about using A* to plan the path of a robot?
- Finds the shortest path
- Requires a graph structure
- Limited number of edges
- In robotics: planning on a 2d occupancy grid map





Alberi di Ricerca

- Metodo:
  - Esplorazione dello spazi degli stati generando il successor degli stati già esplorati.
  - Ogni stato valutato: *è uno stato goal*?

Best-first search

- Idea: usare una funzione di valutazione *f(n)* per ogni nodo:
  - f(n) fornisce una stima del costo totale
  - → Espandi il nodo n con più piccolo f(n).

A* search

- Idea: Evita di espandere percorsi che sono già molto costosi
- Evaluation function $f(n) = g(n) + h(n)$
- $g(n)$ = costo fino ad ora per n
- $h(n)$ = stima del costo da n al goal
- $f(n)$ = stima del costo totale del path per n al goal
- Best First search ha $f(n)=h(n)$
- Uniform Cost search ha $f(n)=g(n)$

Euristica Ammissibile

- Una euristica *h(n)* è ammissibile se per ogni nodo *n*, *h(n)* ≤ $h^*(n)$, *dove* $h^*(n)$ è il costo reale per raggiungere il goal da *n*.
- Una euristica ammissibile non sovrastima il costo per raggiungere il goal, i.e., è ottimistica
- Esempio: $h_{SLD}(n)$ (non sovrastimata la reale distanza)
- Teorema: se *h(n) è* ammissibile, A* con `TREE-SEARCH` è ottimale

Dominanza

- Se $h_2(n) \geq h_1(n)$ per ogni n (entrambe ammissibili)
- allora h_2 domina h_1
- h_2 *è* migliore per la ricerca: è guarantito che espande un numero di nodi minore o uguale.

Relaxed problems

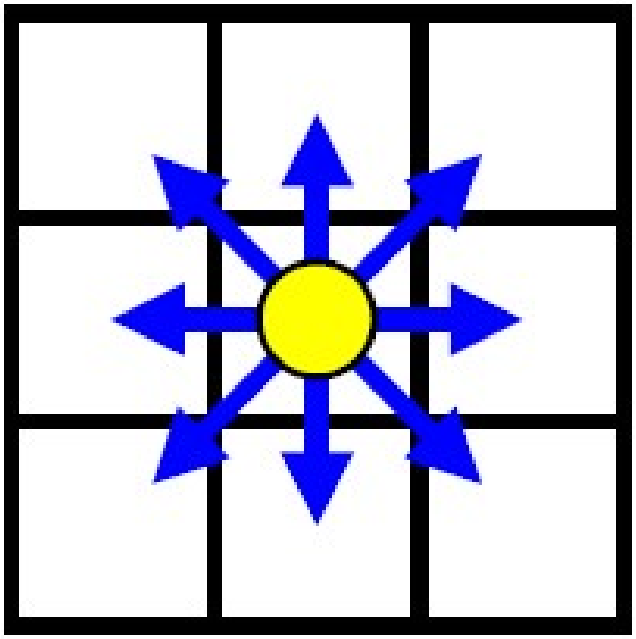
- A problem with fewer restrictions on the actions is called a relaxed problem
- The cost of an optimal solution to a relaxed problem is an admissible heuristic for the original problem
- If the rules of the 8-puzzle are relaxed so that a tile can move anywhere, then $h_1(n)$ gives the shortest solution
- If the rules are relaxed so that a tile can move to any adjacent square, then $h_2(n)$ gives the shortest solution

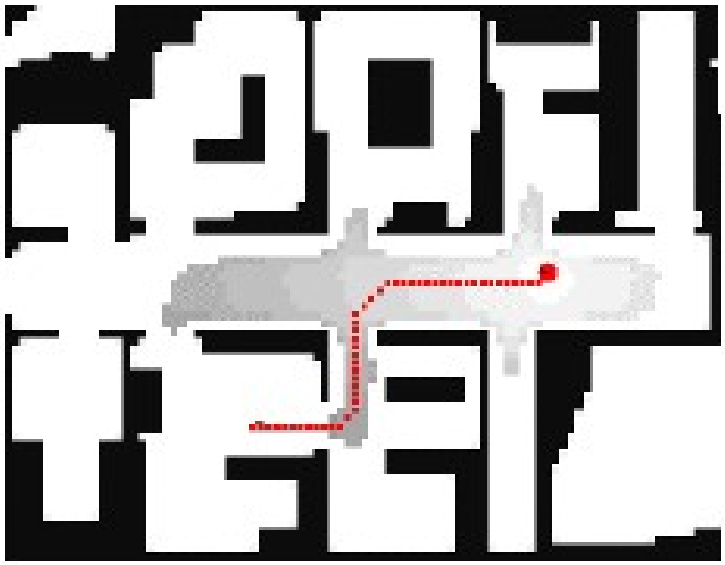
Propertà di A*

- Teminazione e complezza
  - Su grafi finiti ed archi non negativi
- Tempo:
  - Esponenziale: b^d dove b è il branching e d la profondità
  - Polinomiale se: $|h(n) - h^*(n)| \leq O(\log h^*(n))$
- Ottimale
- Ottimamente efficiente: (non c'è altro algoritmo che con la stessa euristica può espandere meno nodi)

Path Planning nella grid map

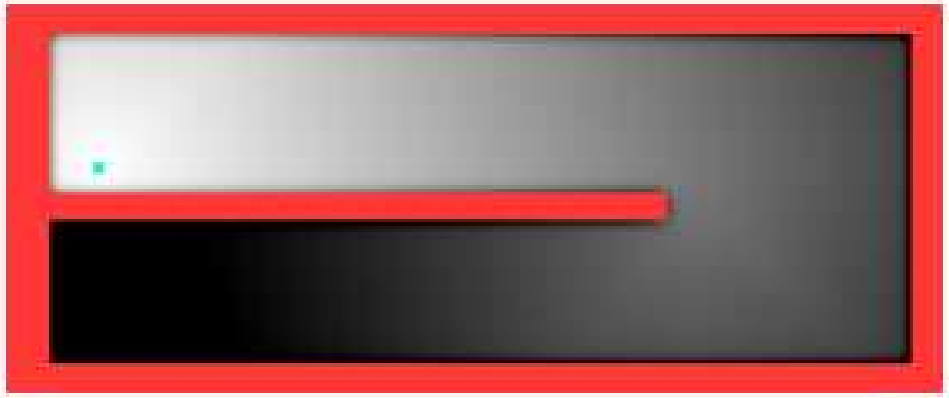
L’algoritmo di A* può essere utilizzato nella grid map (occupancy grid)





Value Iteration

- To compute the shortest path from every state to one goal state, use (deterministic) value iteration.
- Very similar to Dijkstra's Algorithm.
- Such a cost distribution is the optimal heuristic for A*.



Assunzioni tipiche in A* path planning

- A robot is assumed to be localized.
- Often a robot has to compute a path based on an occupancy grid.
- Often the correct motion commands are executed (but no perfect map).

Is this always true?

Problemi

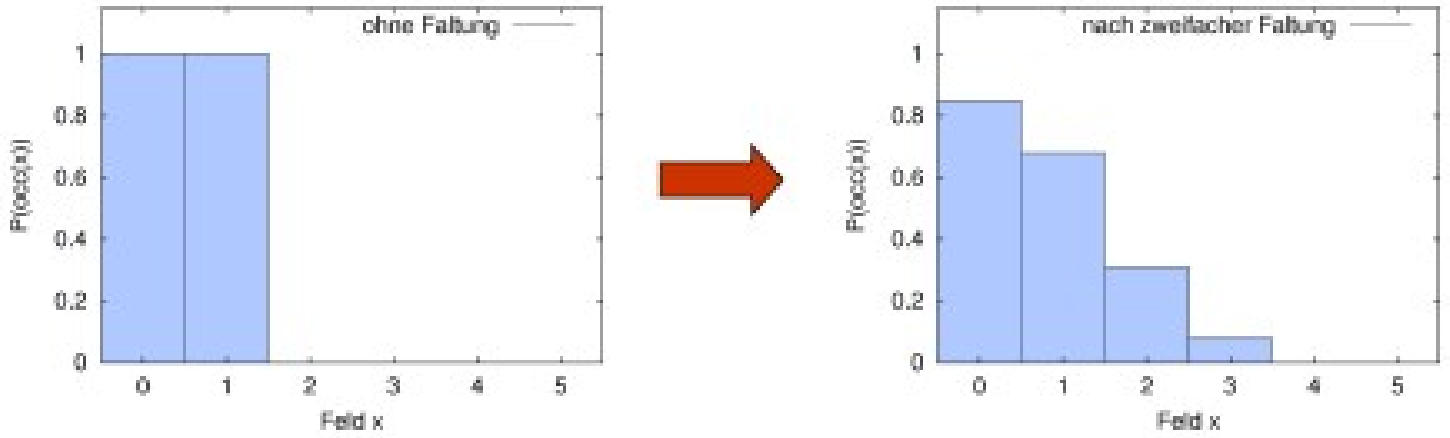
- What if the robot is slightly delocalized?
- Moving on the shortest path guides often the robot on a trajectory close to obstacles.
- Trajectory aligned to the grid structure.

Possibile Soluzione: convoluzione della grid map

- Convolution blurs the map.
- Obstacles are assumed to be bigger than in reality.
- Perform an A* search in such a convolved map.
- Robots increases distance to obstacles and moves on a short path!

Esempio

- 1-d environment, cells c_0, …, c_5



- Cells before and after 2 convolution runs.

Convoluzione

- Consider an occupancy map. Than the convolution is defined as:

$$P(occ_{x_i,y}) = \frac{1}{4} \cdot P(occ_{x_{i-1},y}) + \frac{1}{2} \cdot P(occ_{x_i,y}) + \frac{1}{4} \cdot P(occ_{x_{i+1},y})$$

$$P(occ_{x_0,y}) = \frac{2}{3} \cdot P(occ_{x_0,y}) + \frac{1}{3} \cdot P(occ_{x_1,y})$$

$$P(occ_{x_{n-1},y}) = \frac{1}{3} \cdot P(occ_{x_{n-2},y}) + \frac{2}{3} \cdot P(occ_{x_{n-1},y})$$

- This is done for each row and each column of the map.
- "Gaussian blur"

Applicazione di A*

- The costs are a product of path length and occupancy probability of the cells.
- Cells with higher probability (e.g., caused by convolution) are avoided by the robot.
- Thus, it keeps distance to obstacles.
- This technique is fast and quite reliable.

Navigazione Costale

- Non il path più breve, ma quello che minimizza la probabilità di perdersi
- Considerare: lunghezza path e infromazione persa (es. Minerva Robot)
- Entropy Map: