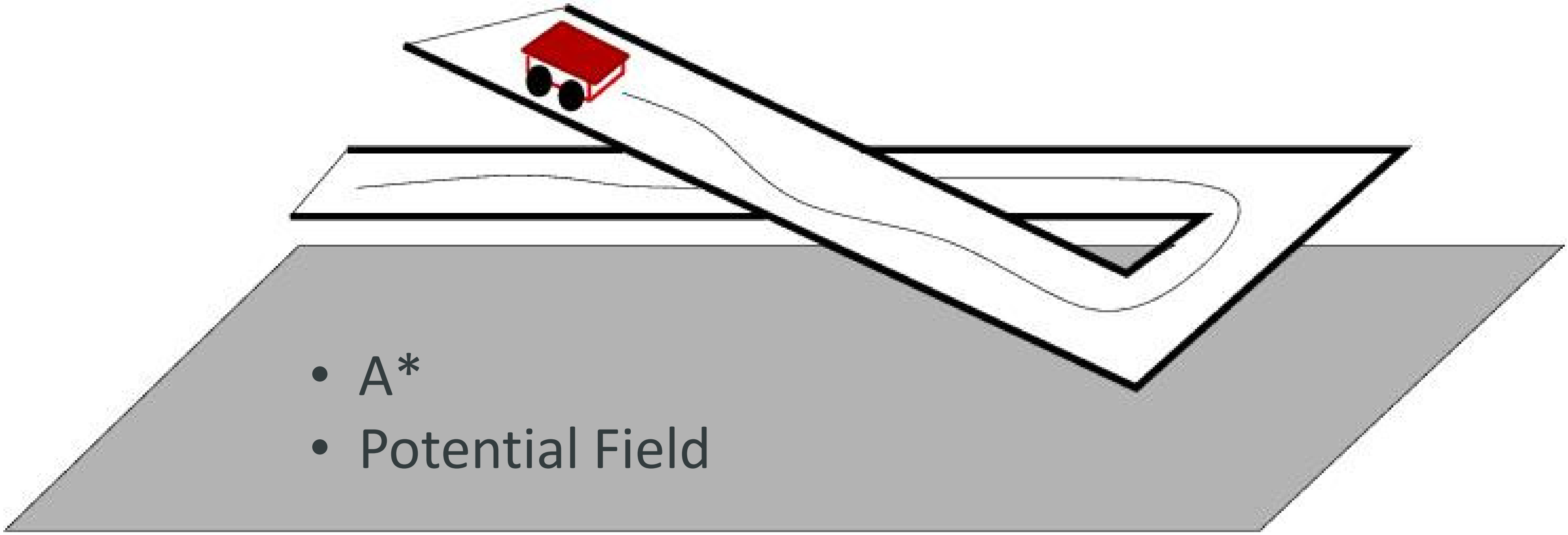
Lecture 8: ROBOT PATH PLANNING



- A*
- Potential Field

Robot Path Planning

GOALS

- collision-free trajectories
- robot should reach the goal location as fast as possible

requires an algorithm to find a suitable path between a start and target point whilst avoiding certain areas referred to as obstacles.

Department of Informatics
Centre for Robotics Research
KING'S College LONDON

Path planning algorithm - A*

- presented in 1968 by Hart *et al.*
- is admissible, i.e. guarantees to find the shortest path, if there is one
- is minimalistic, i.e. if compared to other admissible algorithms with the same knowledge of the search area, it will investigate the minimal amount of nodes necessary to find an optimal path.

Department of Informatics
Centre for Robotics Research
KING'S College LONDON

The A* Search

- A* is an algorithm that:
  – Uses heuristic to guide search
  – While ensuring that it will compute a path with minimum cost

"estimated cost"

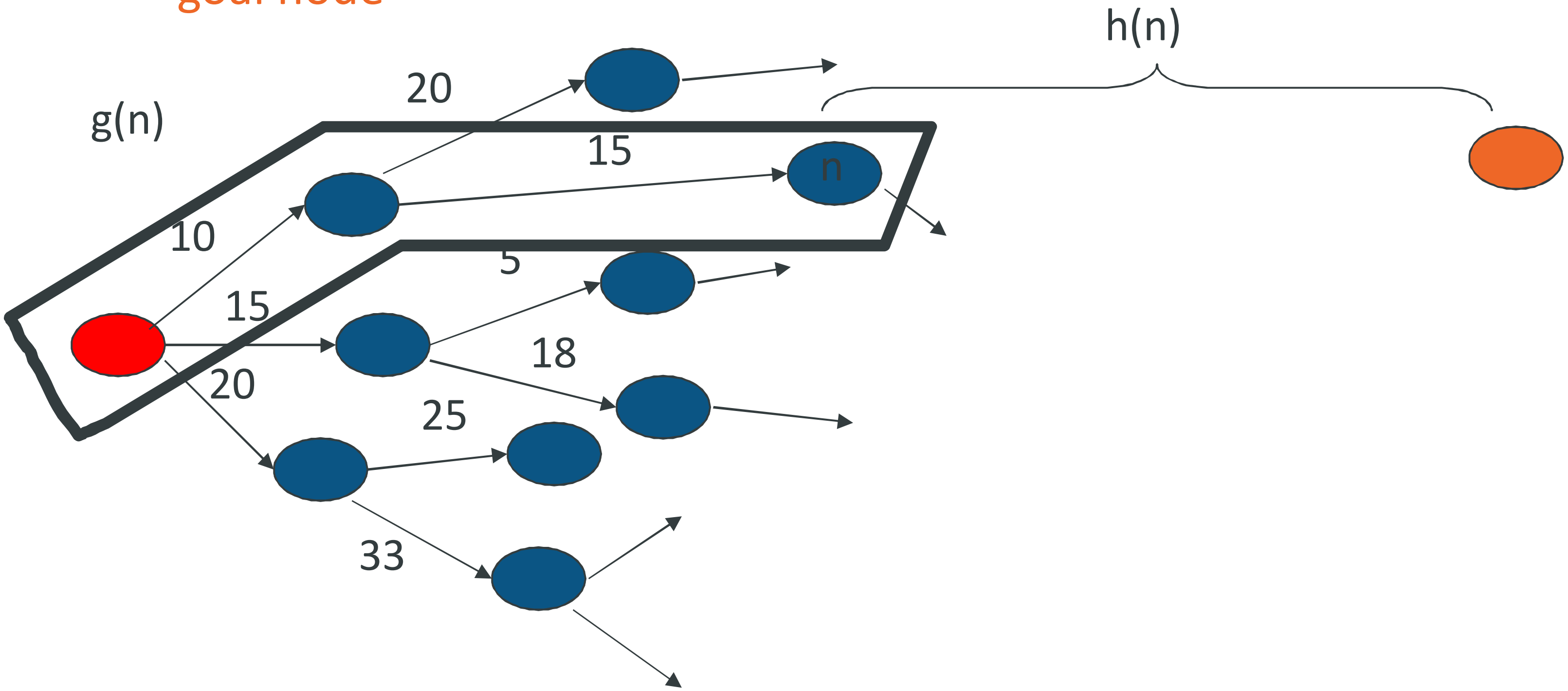
- A* computes the function $f(n) = g(n) + h(n)$

"actual cost"

➢ *g(n)* : cost of moving from the starting point to the node *n*
➢ *h(n)* : is an estimate of the cost of the distance between the node *n* and the goal point

The A* Search

- $f(n) = g(n) + h(n)$
  - $g(n)$ = “cost from the starting node to reach n”
  - $h(n)$ = “estimate of the cost of the cheapest path from n to the goal node”



Department of Informatics
Centre for Robotics Research
KING'S College LONDON

Heuristics

- technique designed for solving problem quickly when classic methods are too slow, or finding an approximate solution when classic methods fail to find an exact solution. It can be considered a shortcut.
- in A* to be admissible, the heuristic may never overestimate the cost of the optimal path from node n to the target node:

$$h(n) \leq \hat{h}(n) \text{ for all } n$$

where $h(n)$ is some estimate for the cost of the optimal path from node n to the target node and $\hat{h}(n)$ is the actual cost.

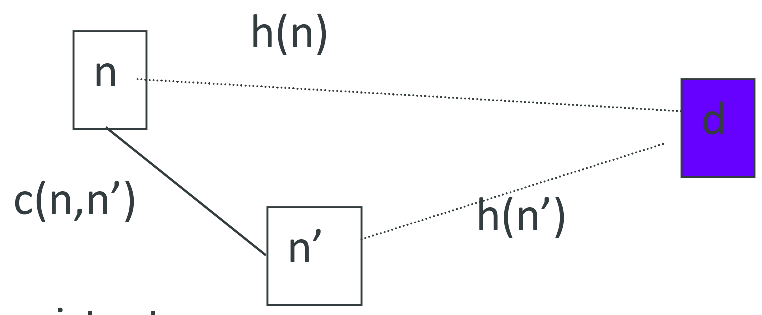
Department of Informatics
Centre for Robotics Research
KING'S College LONDON

Heuristics (cont.)

- A* generates an optimal solution if h(n) is an admissible heuristic and the search space is a tree:
  – h(n) is **admissible** if it never overestimates the cost to reach the destination node

- A* generates an optimal solution if h(n) is a consistent heuristic and the search space is a graph:
  – h(n) is **consistent** if for every node n and for every successor node n' of n:

$$h(n) \leq c(n,n') + h(n')$$



- If h(n) is consistent then h(n) is admissible
- Frequently when h(n) is admissible, it is also consistent

Department of Informatics
Centre for Robotics Research
KING'S College LONDON

Path planning algorithm – A* implementation

Variables and functions:

- *n* : node in the search space
- *close list*: contains all the nodes that have already been explored and the obstacles(added before starting the search)
- *open list*: contains all the nodes that still need to be checked
- $f(n)$=g(n)+$h(n)$

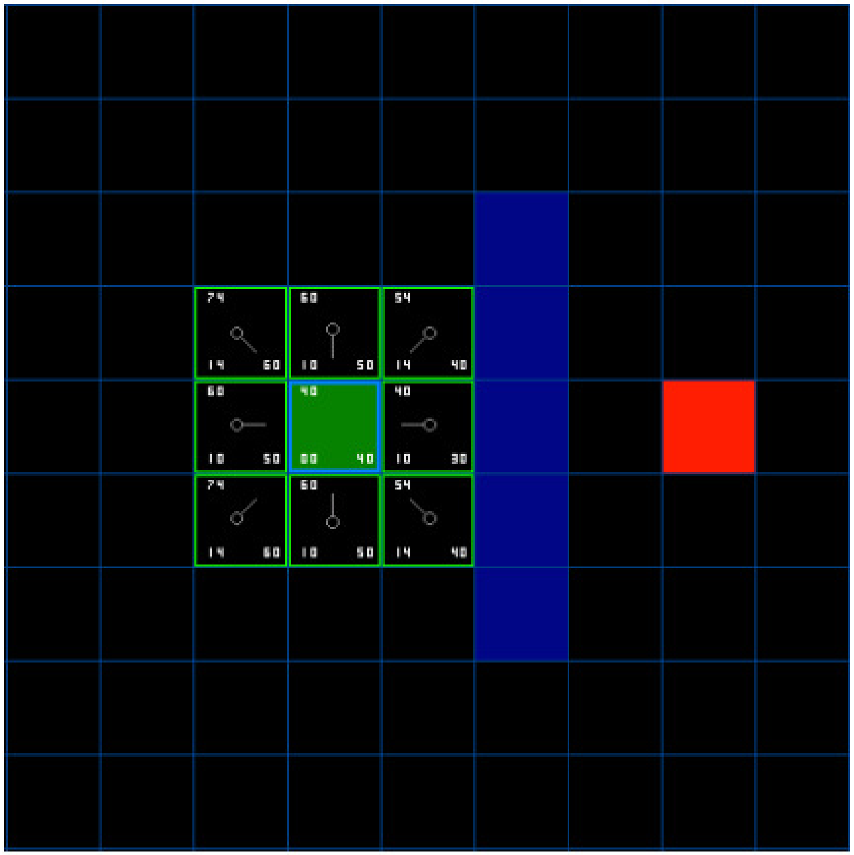
Department of Informatics
Centre for Robotics Research
KING'S College LONDON

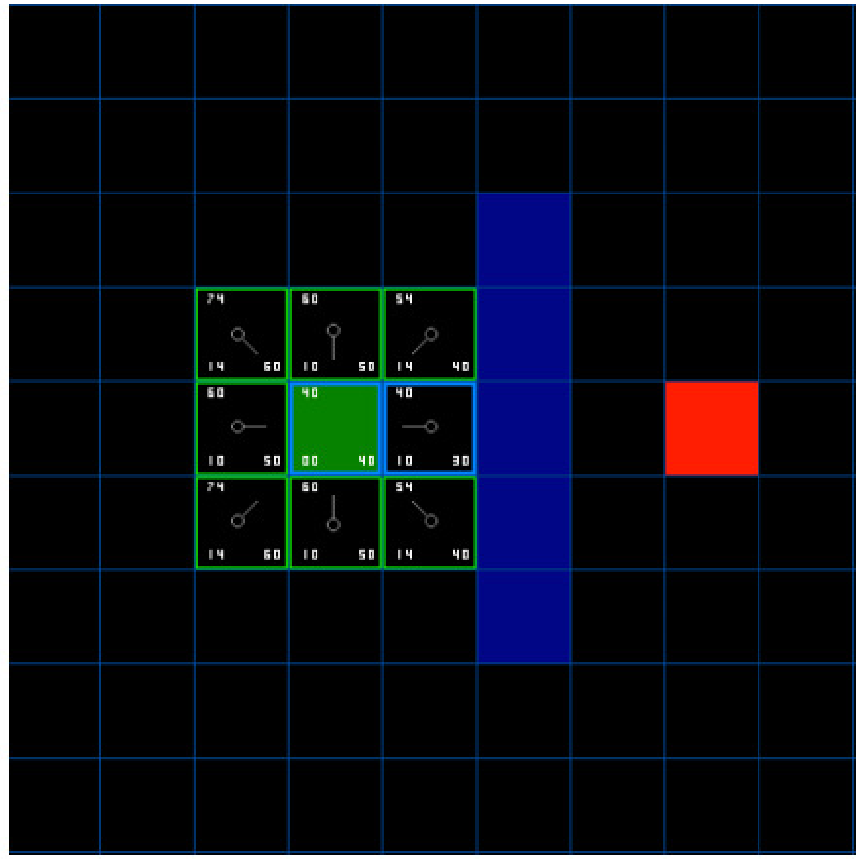
Path planning algorithm – A* implementation(cont.)

1. Set start node as current node and add it to the *closed list*
2. add all neighbours of the current node to *open list* along with *g(n)*, *h(n)*, *f(n)* and their parent node
3. if a neighbour is already in the open list, check if the *f(n)* score is lower and if so, update *g(n)*, *h(n)*, *f(n)* and its parent
4. choose the node with the smallest *f(n)* score in the *open list*, set it as the current node, remove it from the *open list* and add it to the *closed list*
5. repeat step 2 through 4, until the target node is added to the *closed list*, or until the *open list* is empty
6. if the target node is found, retrace the nodes by identifying their parents until the start node is found; this is the shortest path
7. if the *open list* is empty, no path to the target is possible

Department of Informatics
Centre for Robotics Research
KING'S College LONDON

Path planning algorithm – A* implementation





Department of Informatics
Centre for Robotics Research
KING'S College LONDON

Heuristics(cont.)

MANHATTAN

$$h(n) = |x_n - x_{goal}| + |y_n - y_{goal}|$$

x_n, x_{goal} ,y_n and y_{goal} are the node x-position, goal x-position, node y-position and goal y-position.

DIAGONAL

$$h_{manhattan}(n) = |x_n - x_{goal}| + |y_n - y_{goal}|$$

$$h_{diagonal}(n) = \min(|x_n - x_{goal}|\ ,\ |y_n - y_{goal}|)$$

$$h(n) = \sqrt{2}\ h_{diagonal}(n) + (h_{manhattan}(n) - 2\ h_{diagonal}(n))$$

Department of Informatics
Centre for Robotics Research
KING'S College LONDON

Heuristics(cont.)

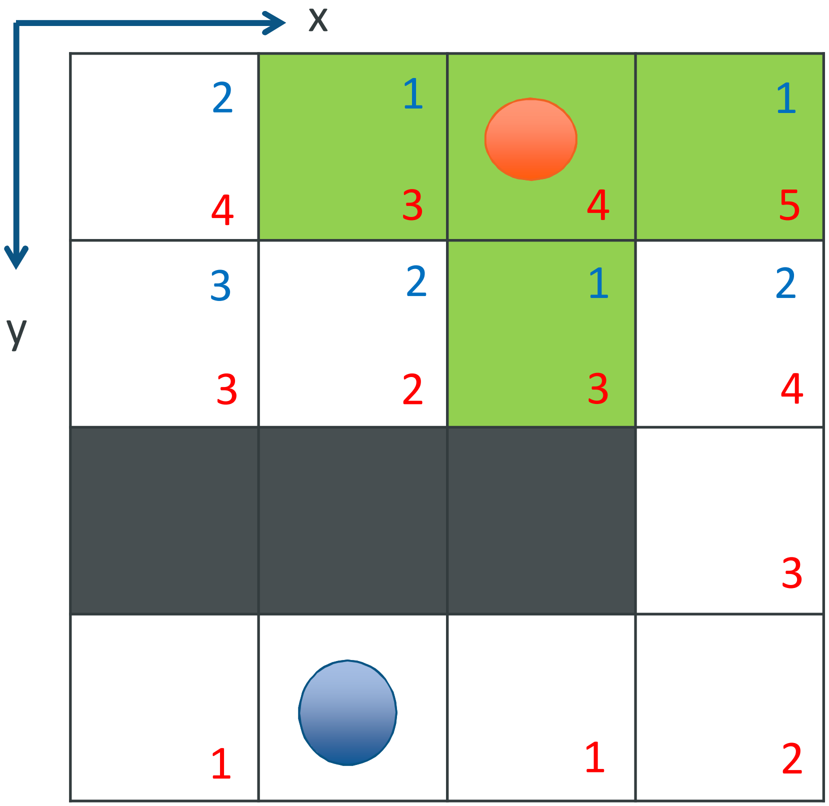
EUCLIDIAN

$$h(n) = \sqrt{(x_n - x_{goal})^2 + (y_n - y_{goal})^2}$$

Square root is relatively computational complex

Department of Informatics
Centre for Robotics Research
KING'S College LONDON

A*- Path Planning



(2,4)

H(n) g(n)=1

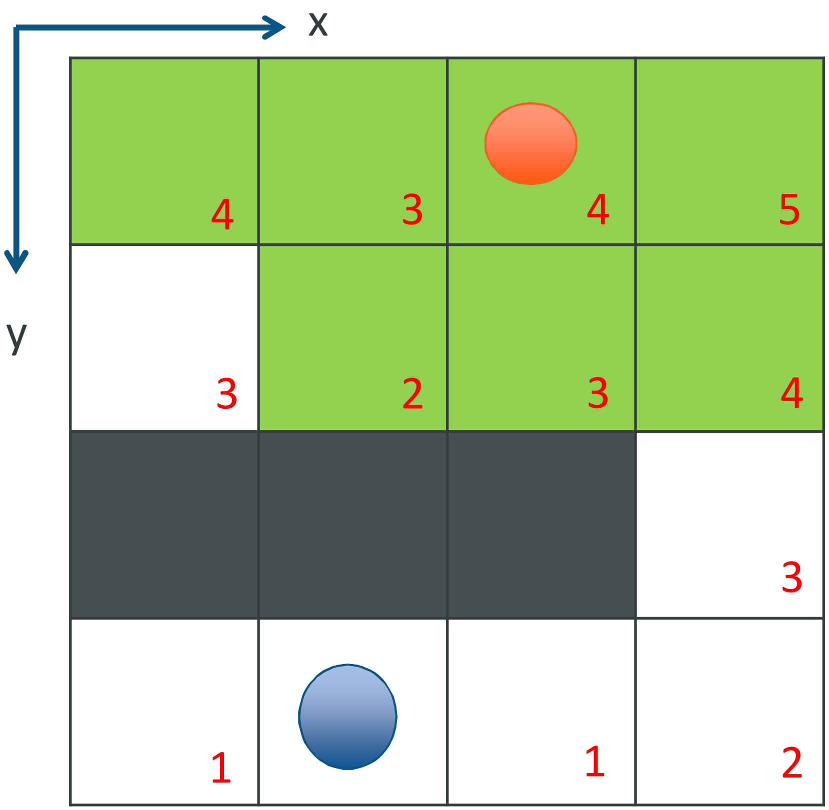
Close list
(1,3)
(2,3)
(3,3)
(3,1)

Starting node
(3,1)

Open list
(2,1)=4 (4,1)=6 (3,2)=4

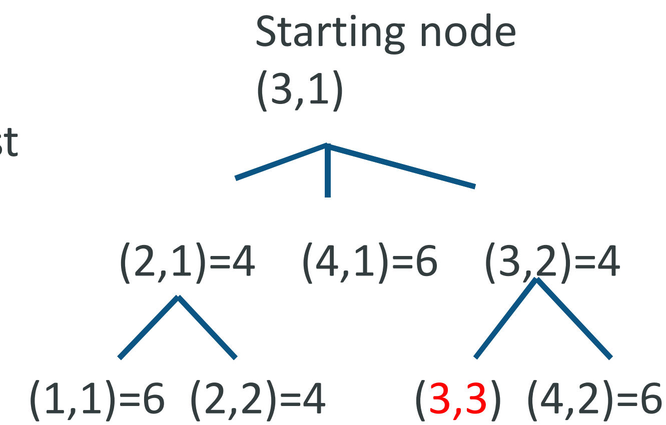
Department of Informatics
Centre for Robotics Research
KING'S College LONDON

A*- Path Planning



(2,4)

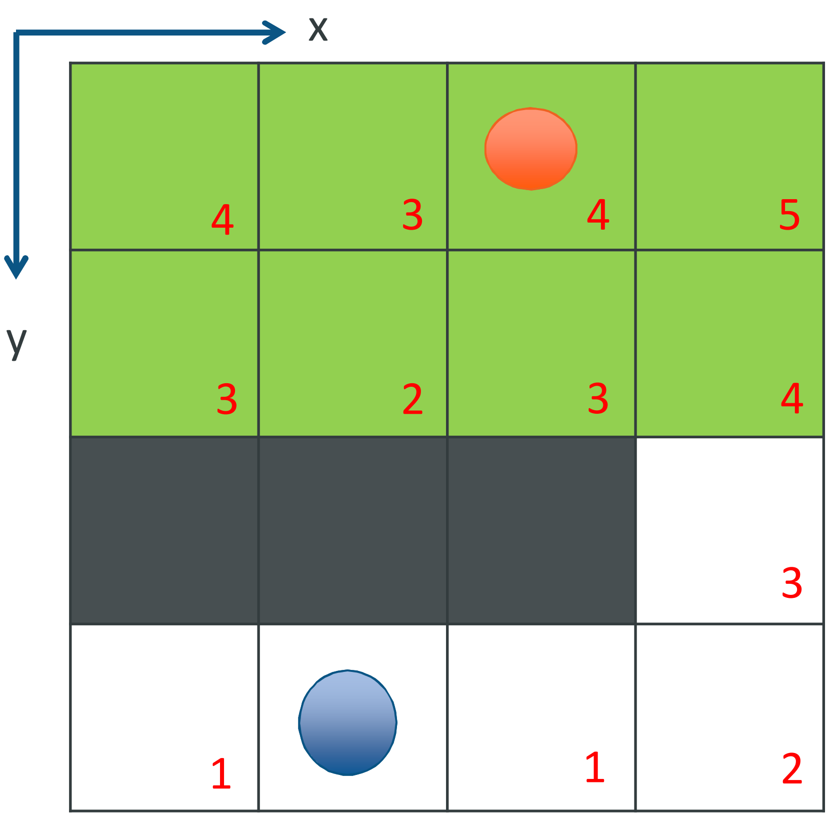
Close list
(1,3)
(2,3)
(3,3)
(3,1)
(2,1)
(3,2)



H(n) (Manhattan) g(n)=1

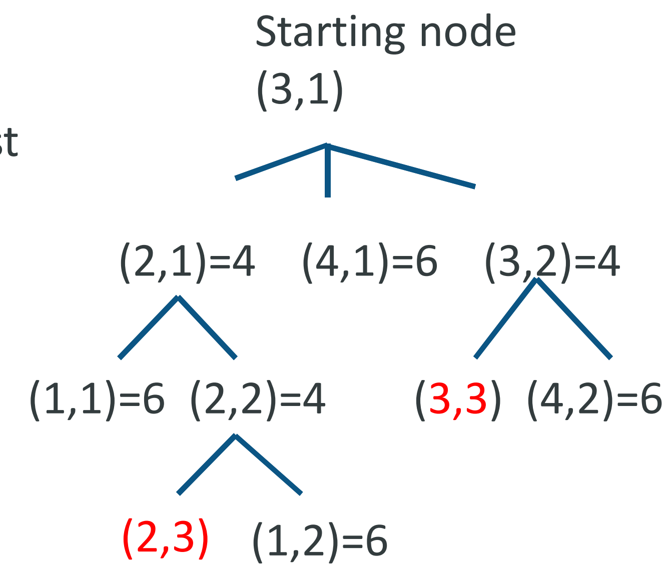
Department of Informatics
Centre for Robotics Research
KING'S College LONDON

A*- Path Planning



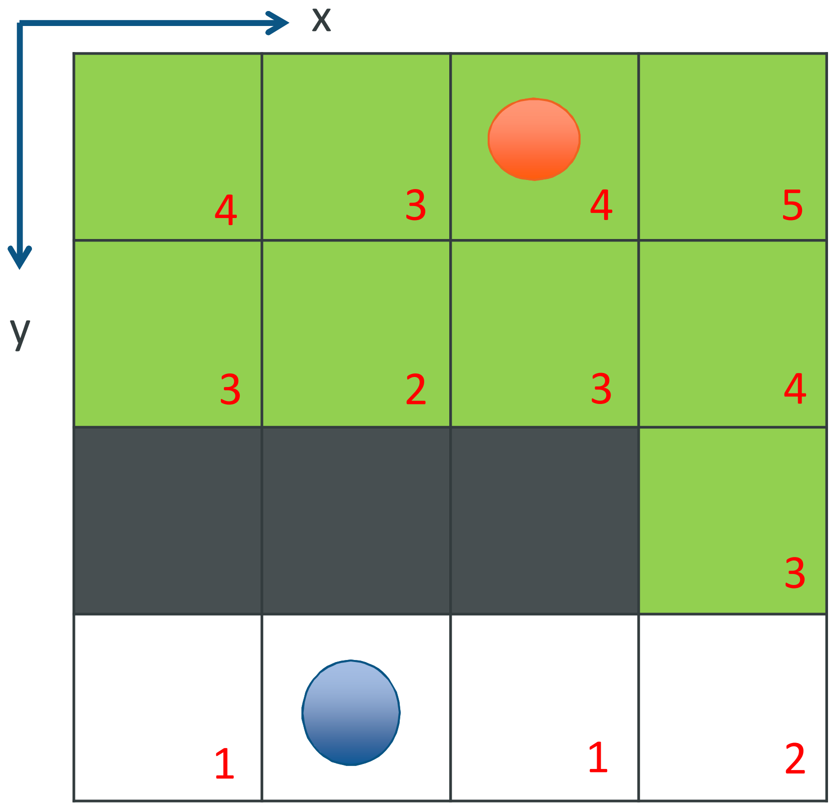
(2,4)

Close list
(1,3)
(2,3)
(3,3)
(3,1)
(2,1)
(3,2)
(2,2)



H(n) (Manhattan) g(n)=1

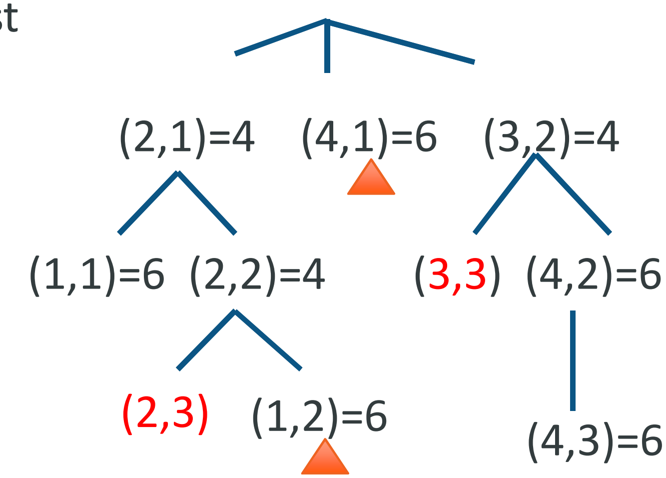
A*- Path Planning



(2,4)

Close list
(1,3)
(2,3)
(3,3)
(3,1)
(2,1)
(3,2)
(2,2)
(1,2)
(1,1)
(4,2)

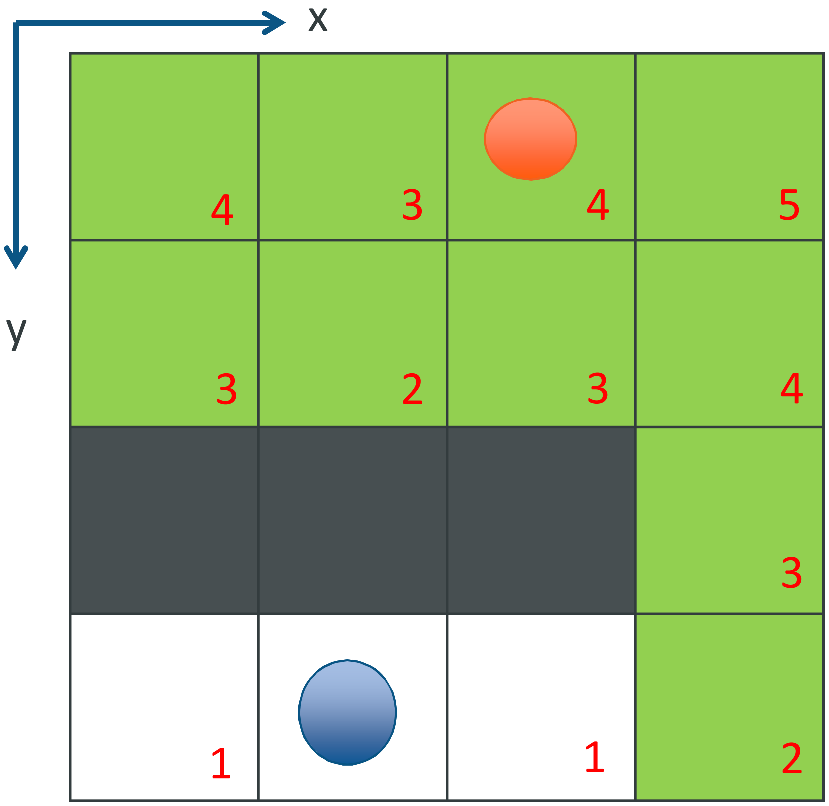
Starting node
(3,1)



H(n) (Manhattan) + g(n)=1

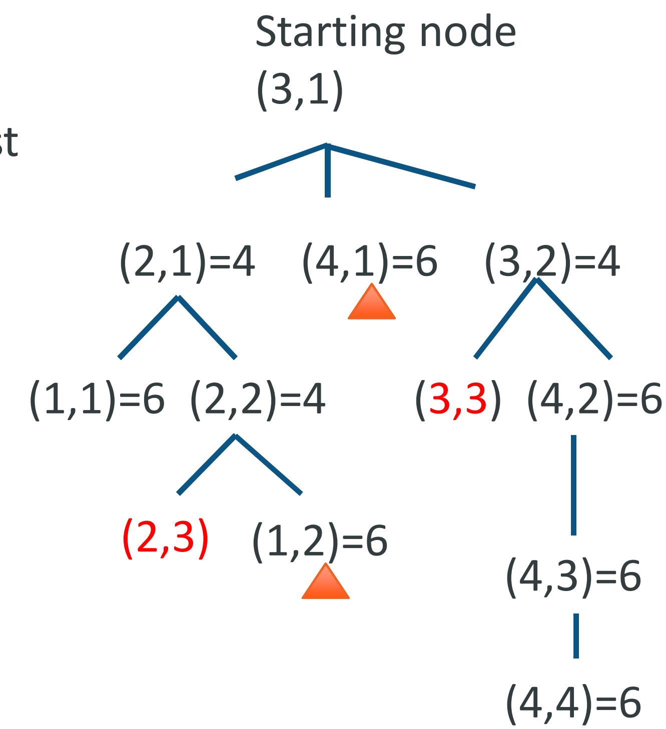
Department of Informatics
Centre for Robotics Research
KING'S College LONDON

A*- Path Planning



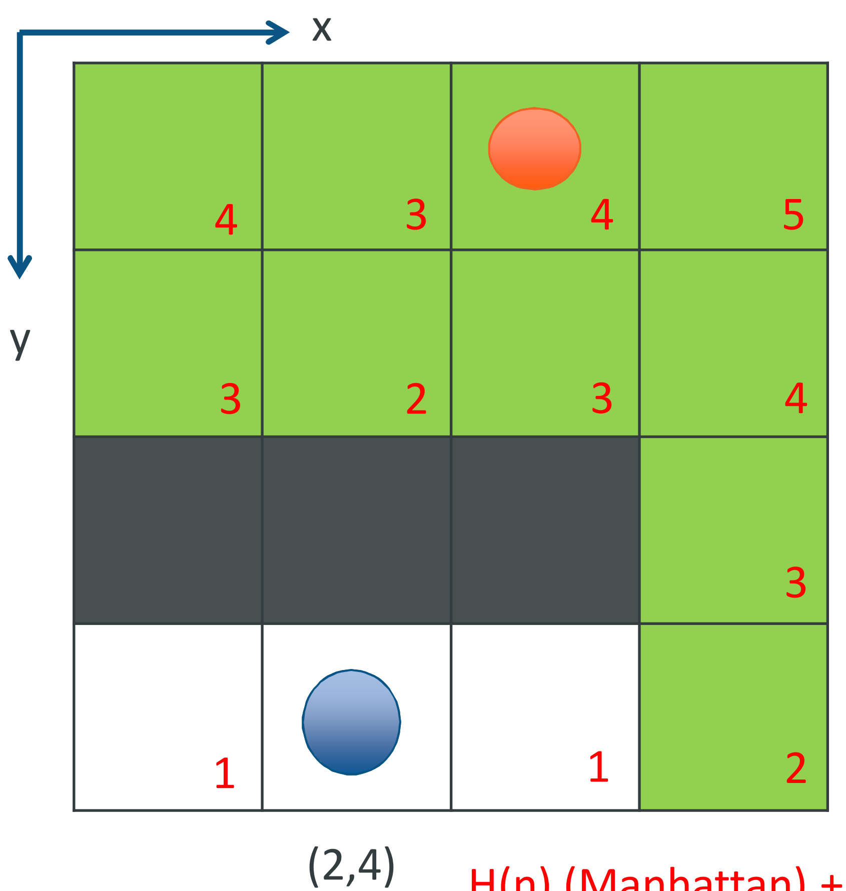
(2,4)

Close list
(1,3)
(2,3)
(3,3)
(3,1)
(2,1)
(3,2)
(2,2)
(1,2)
(1,1)
(4,2)
(4,3)

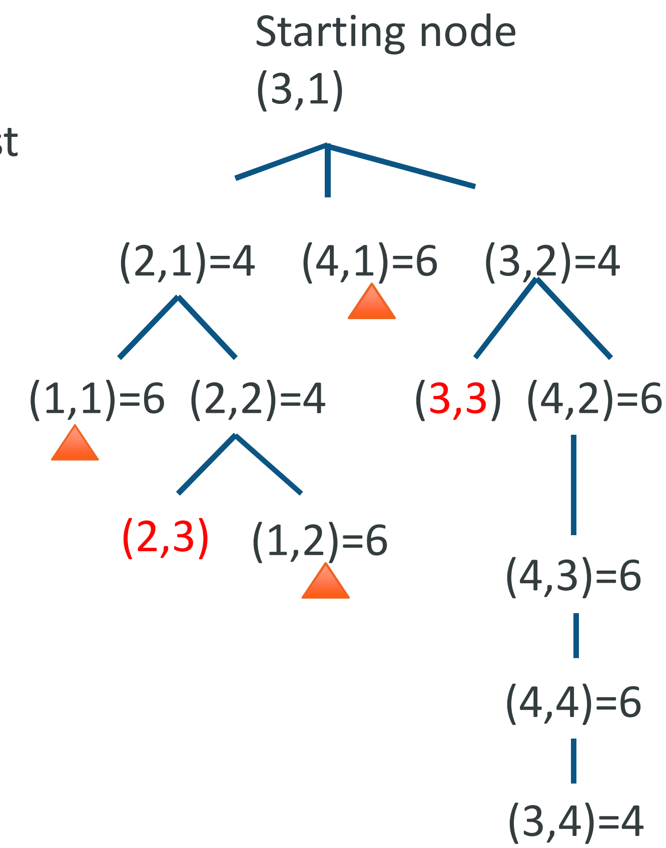


H(n) (Manhattan) + g(n)=1

A*- Path Planning



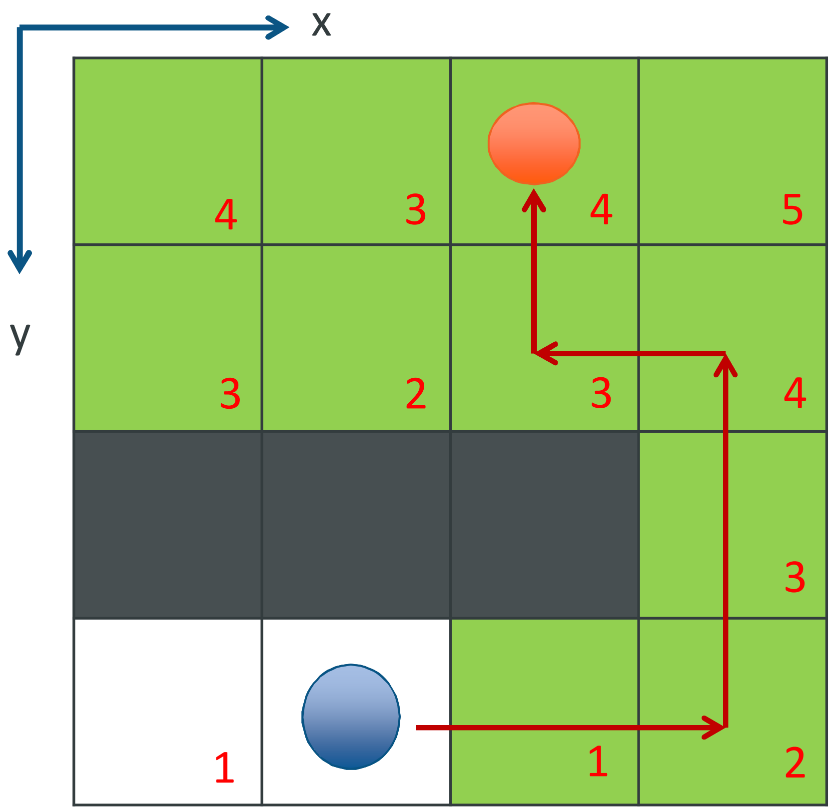
Close list
(1,3)
(2,3)
(3,3)
(3,1)
(2,1)
(3,2)
(2,2)
(1,2)
(1,1)
(4,2)
(4,3)
(4,4)



H(n) (Manhattan) + g(n)=1

Department of Informatics
Centre for Robotics Research
KING'S College LONDON

A*- Path Planning



(2,4)

Close list	Current node	Open list
(1,3)	(3,4)=2	(2,4)=goal
(2,3)		
(3,3)		
(3,1)		
(2,1)		
(3,2)		
(2,2)		
(1,2)		
(1,1)		
(4,2)		
(4,3)		
(4,4)		

H(n) (Manhattan) + g(n)=1

Department of Informatics
Centre for Robotics Research
KING'S College LONDON

Potential Fields

- **Initially proposed for real-time collision avoidance [Khatib 1986].**
- **A potential field is a scalar function over the free space.**
- **To navigate, the robot applies a force proportional to the negated gradient of the potential field.**
- **A** navigation function **is an ideal potential field that**
  - **has global minimum at the goal**
  - **has no local minima**
  - **grows to infinity near obstacles**
  - **is smooth**

Attractive & Repulsive Fields

$$F_{att} = -k_{att}(x - x_{goal})$$

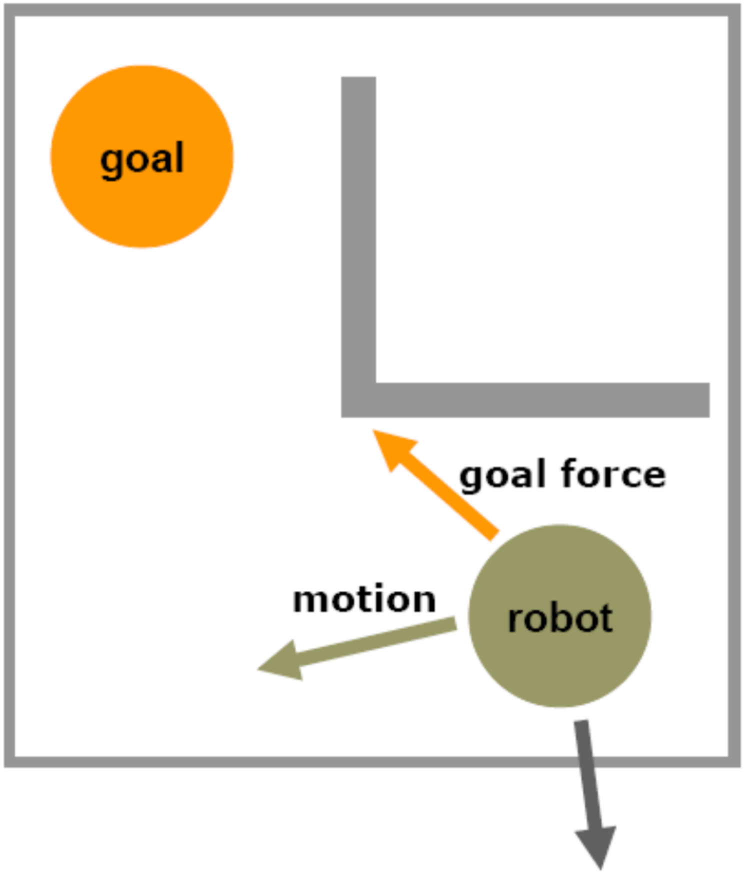
$$F_{rep} = \begin{cases} k_{rep}\left(\frac{1}{\rho} - \frac{1}{\rho_0}\right)\frac{1}{\rho^2}\frac{\partial\rho}{\partial x} & \text{if } \rho \le \rho_0, \\ 0 & \text{if } \rho > \rho_0 \end{cases}$$

k_{att}, k_{rep} : positive scaling factors

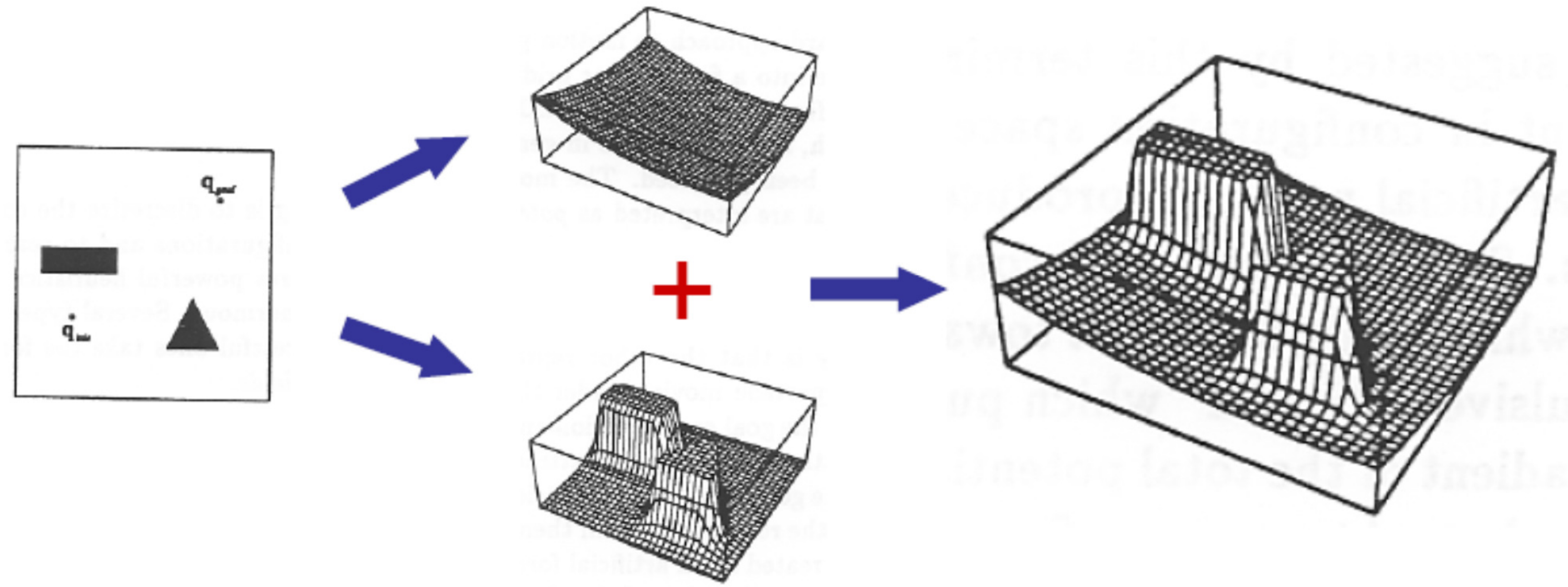
x : position of the robot

ρ : distance to the obstacle

ρ_0 : distance of influence



How Does It Work?

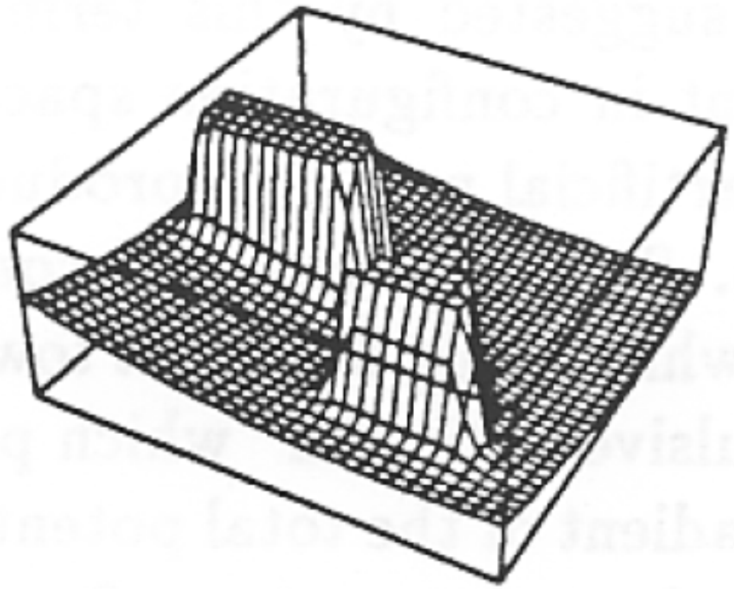


Algorithm Outline

- Place a regular grid G over the configuration space
- Compute the potential field over G
- Search G using a best-first algorithm-such as A* with potential field as the heuristic function

Local Minima

- **What can we do?**
  - **Escape from local minima by taking random walks**
  - **Build an ideal potential field – navigation function – that does not have local minima**



Department of Informatics
Centre for Robotics Research
King's College London