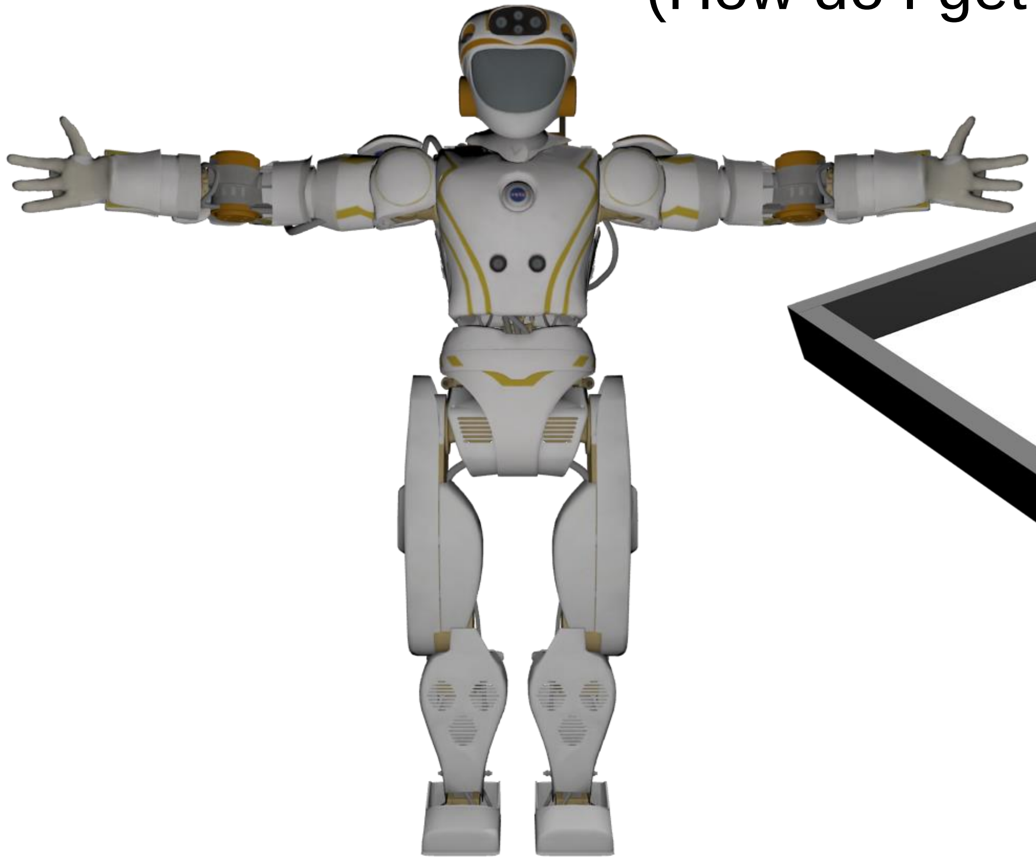
Real-world Multi-agent Systems

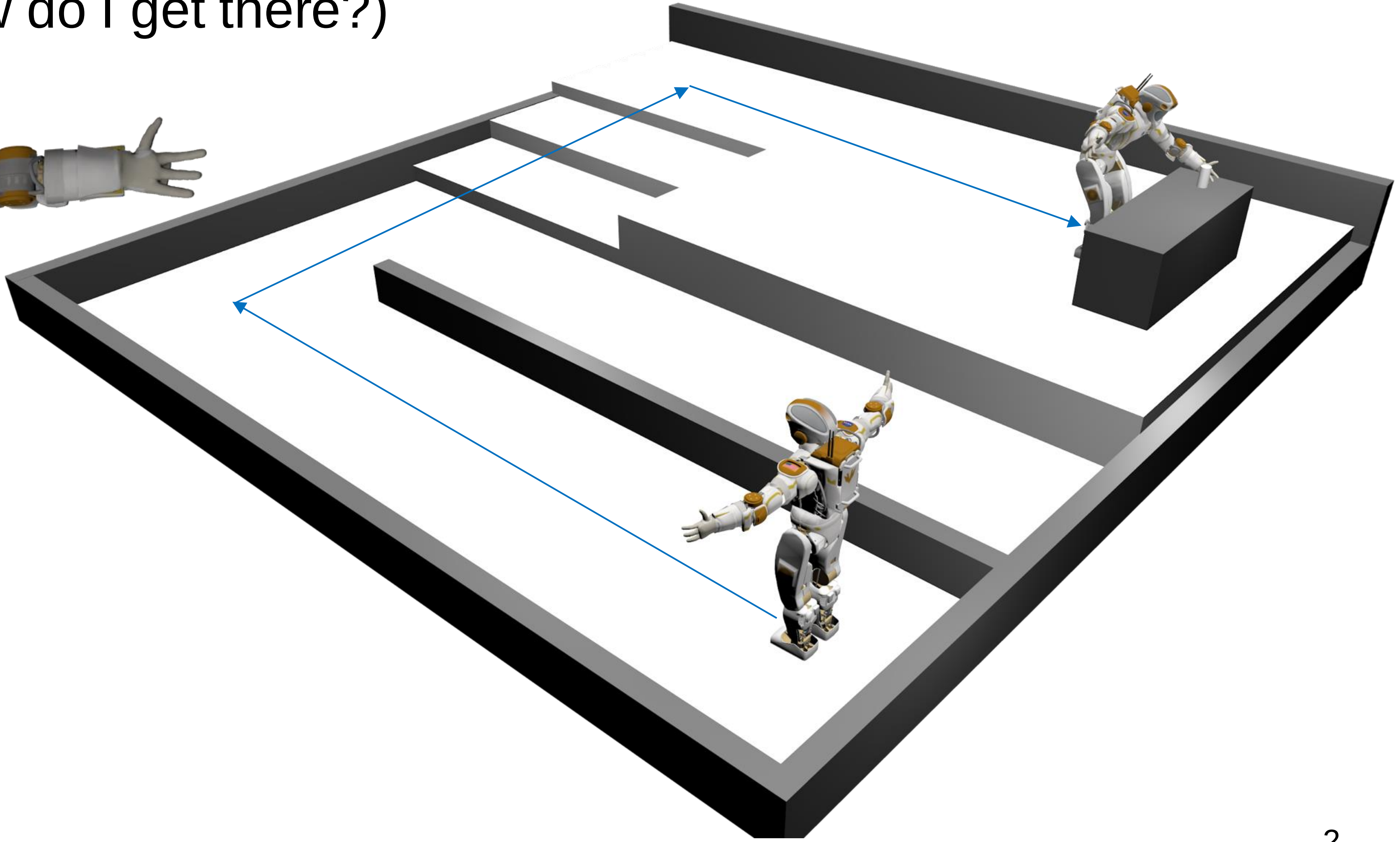
Motion planning

Akin Delibasi
Lecturer in Distributed Systems
Computer Science
University College London, UK

Motion Planning

Motion Planning
(How do I get there?)





Outline

- Motion planning:
  - ❑ Goals and challenges
  - ❑ Concepts and definitions
- Planning in continuous space: Potential Fields (PF)
- Sampling-based planning:
  - ❑ Rapidly exploring Random Tree (RRT): principle and code demo
  - ❑ Variants of RRT algorithms
  - ❑ Multi-Query Planner: Probabilistic Roadmap (PRM)
  - ❑ Grid-based search algorithms: A*

Motion Planning

Fundamental task: plan (collision-free) motions/paths for complex systems from a *start* to a *goal* position among a set of obstacles or within an environment.

- For mobile robots, it is also referred as a *path planning* problem.
- For articulated robots, eg, robot arms, legged robots, it is about finding *valid configurations.*

Definition: Motion Planning is a computational problem to find ***a sequence of valid configurations*** to move an object/agent from a *start* to a *goal* position.

Importance: crucial in fields, such as robotics, animation, and real life mobile, marine, aerial systems.

Motion Planning

Robot Motion – An Observation:

- Most of us have some prior exposure to equations of motion, e.g., Newton’s law f = ma
- Given a physical setup, and given all parameters, it is clear what it means to “compute forward” the motion of robot (“forward” is easy, the inverse is hard)
- The robot’s problem is the opposite one: define (ill-posed) requirements and compute actions to achieve complex goals

What is a Path Planning Problem?

- **Scenario**: a Mars Rover navigating the rocky and unpredictable terrain of Mars. The mission is to travel from its landing site to a distant geological point of interest.

- **Challenges**:
  - ❑ traverse this alien landscape while avoiding large rocks, steep slopes, and deep craters.
  - ❑ communication delay between Earth and Mars, rover must be able to make some decisions autonomously.

What is a Path Planning Problem?

- Elements of the problem:
  - ❑ Description of environment, e.g., a map
  - ❑ Positions of obstacles, terrain properties, etc.
  - ❑ Description of the robot and its capabilities, e.g., geometry of body, ability to move, etc.

- Problem: Given the above elements and start & goal points (sets), write a program to get from start to end
  - ❑ How to define feasible paths?
  - ❑ Can we have additional costs?

Principles of Motion Planning

- **Completeness**: A motion planning algorithm is complete, if it can find a path; if not, report that no path exists.

- **Optimality**: An optimal motion planning algorithm always finds the *best* path (as possible) according to a given cost function.

- **Computational Complexity**: The time it takes for the algorithm to find a path.

Concepts & theories: Configuration Space

Configuration is a key concept for motion planning: a complete specification of the position of every point in the robot geometry, or a configuration ***q***.

Define by ***A*** the complete description of the geometry of a robot, and by ***W*** a workspace. The goal of motion planning to find a collision-free path for ***A*** to move from an initial configuration to a goal configuration.

The configuration space or ***C-space***: the space of all possible configurations.

Concepts & theories

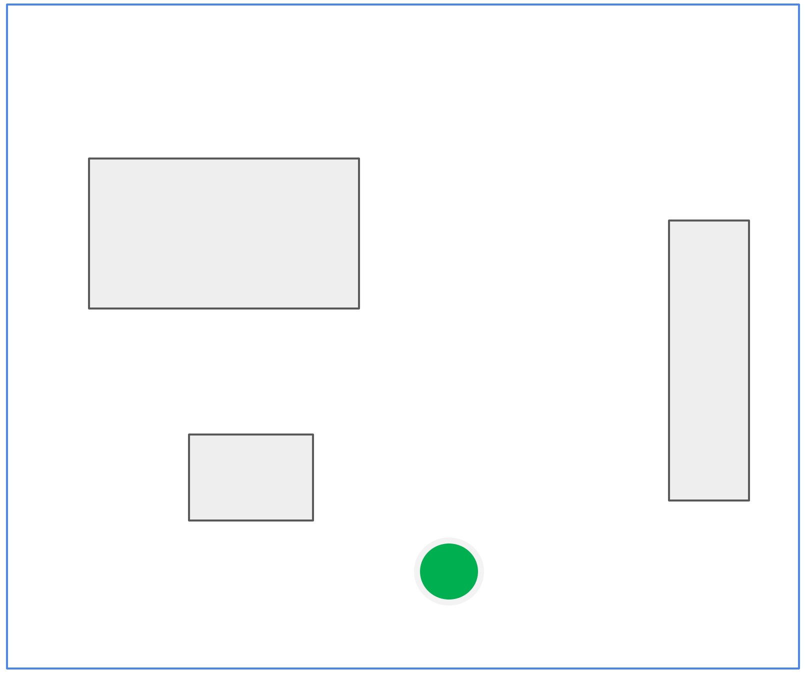
The C-space obstacle region, $\boldsymbol{C}_{\text{obs}}$, is defined as

$$\boldsymbol{C}_{\text{obs}} = \{\boldsymbol{q} \in \boldsymbol{C} \mid \boldsymbol{A}(\mathbf{q}) \cap \boldsymbol{O} \neq \emptyset\}$$

Since $\boldsymbol{O}$ and $\boldsymbol{A}(\mathbf{q})$ are closed sets in $\boldsymbol{W}$.

Thus, the free space that avoids collision is the set of configurations:

$$\boldsymbol{C}_{\text{free}} = \boldsymbol{C} \setminus \boldsymbol{C}_{\text{obs}}$$



Geometric Path Planning Problem

1. A workspace $\boldsymbol{W}$, where either $W = R^2$ or $W = R^3$.

2. An obstacle region $\boldsymbol{O} \subset \boldsymbol{W}$.

3. A robot defined in $\boldsymbol{W}$ by a rigid body $\boldsymbol{A}$ or a set of m links: $\boldsymbol{A_1}$, $\boldsymbol{A_2}$, ... $\boldsymbol{A_m}$.

4. The configuration space $\boldsymbol{C}$, $\boldsymbol{C}_{\text{free}}$, $\boldsymbol{C}_{\text{obs}}$.

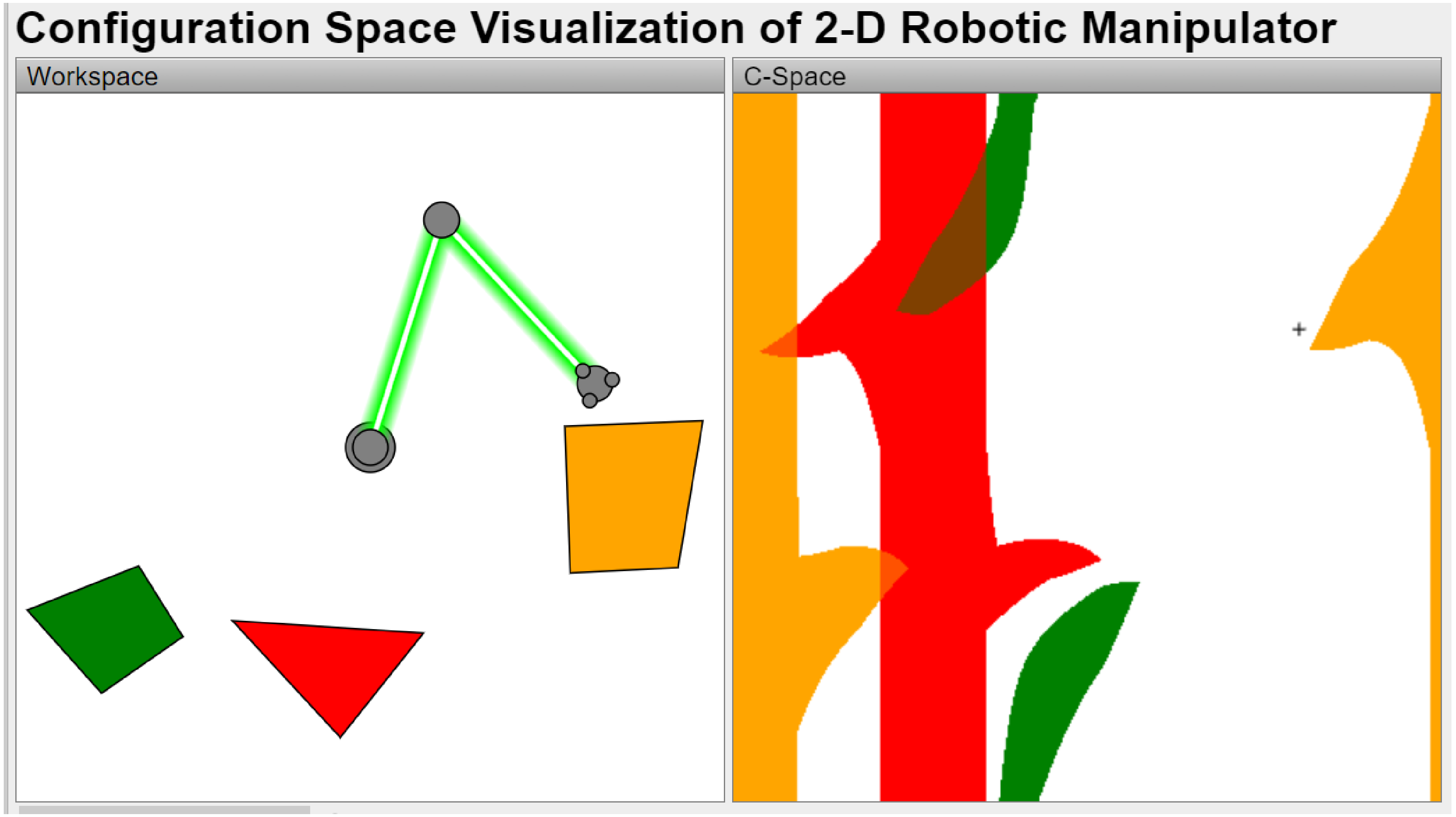
5. An initial configuration $\boldsymbol{q_I} \in \boldsymbol{C}_{\text{free}}$.

6. A goal configuration $\boldsymbol{q_G} \in \boldsymbol{C}_{\text{free}}$. The initial and goal configuration are often called a query $(\boldsymbol{q_I} ; \boldsymbol{q_G})$.

Compute a continuous path $\boldsymbol{\tau} : [0, 1]$ in $\boldsymbol{C}_{\text{free}}$ such that $\boldsymbol{\tau}(0)=\boldsymbol{q_I}$, $\boldsymbol{\tau}(1)=\boldsymbol{q_G}$.

Visualization of C-Space of 2-D Robot Arm

- Try it out by yourself: www.cs.unc.edu/~jeffi/c-space/robot.xhtml


Configuration Space Visualization of 2-D Robotic Manipulator
Workspace
C-Space

Planning methods

Types of Motion Planning

- Kinodynamic: These methods take into account both the robot’s kinematics and dynamics.
- Geometric-based: These methods consider the geometric representation of the robot and its environment.
- Decision-theoretic: decision theory to handle uncertainty in motion planning, factoring elements which are beyond geometry and kinodynamic.

Multi-Agent Path Planning: Multi-agent motion planning extends the principles of motion planning to multiple robots or agents. The goal is to plan paths for multiple agents from start to goal without collisions or violation of any specified constraints. (More details covered in lecture 6)

Types of Motion Planning

Paradigms

- Reactive Paradigm: simplified models, relies heavily on good sensing
- Probabilistic Robotics: seamless integration of models and sensing, inaccurate models, inaccurate sensors
- Hybrids: explicit model-based at higher levels, reactive at lower levels

  - Planning in continuous space: potential-field-based techniques
  - Sampling-Based Planning: RRT, PRM

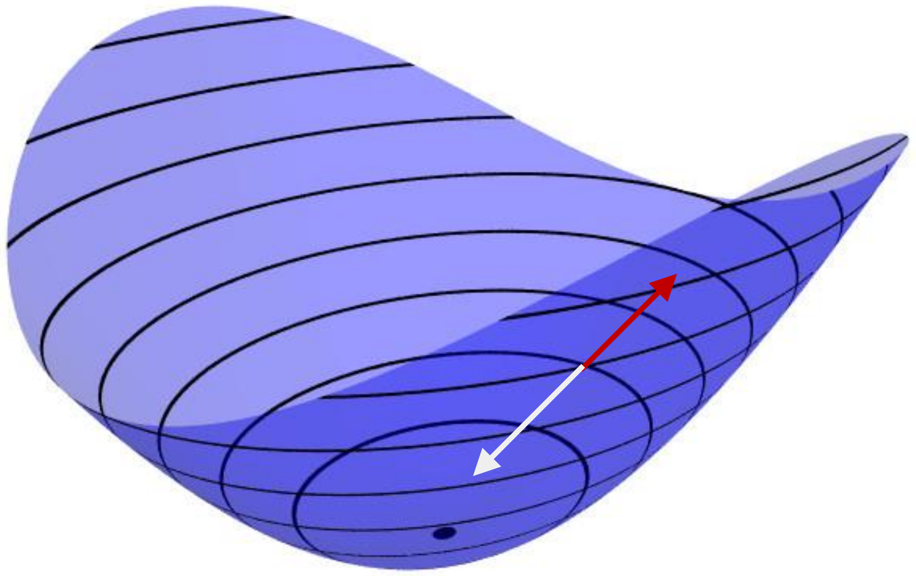
Artificial Potential Fields

Potential Fields

- ❑ Artificial potential field approach is originally proposed for collision avoidance. It constructs a differentiable real-valued function

  $\boldsymbol{U} : \boldsymbol{R}^m \rightarrow \boldsymbol{R}$

  called a potential function, which guides the motion of the moving object.



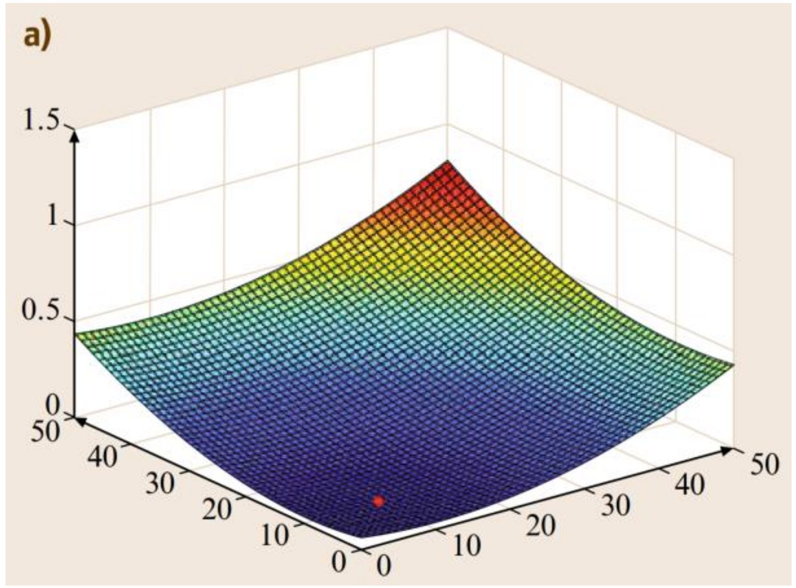
- ❑Treat the value as ‘energy’
- ❑Then, gradient is the vector,

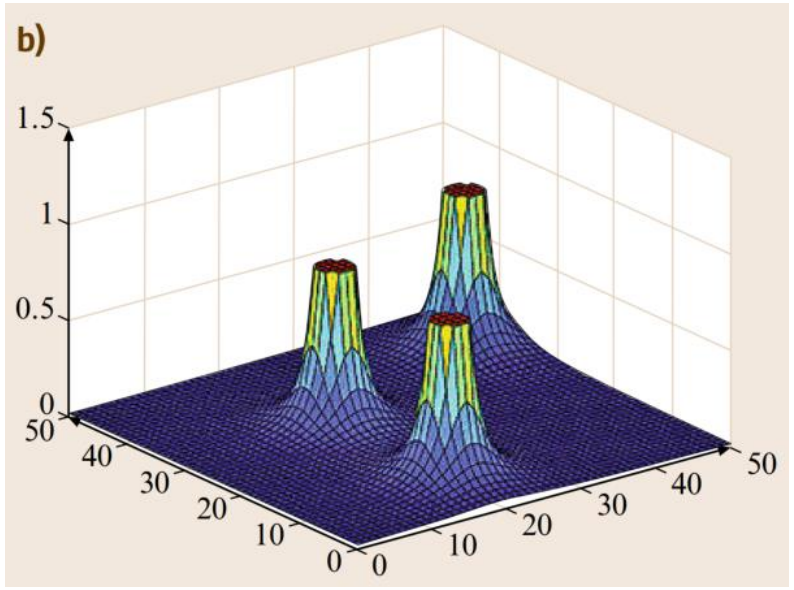
$$\nabla U(q) = DU(q)' = [\frac{\partial U}{\partial q_1}(q), \ldots, \frac{\partial U}{\partial q_m}(q)]'$$

Attractive and Repulsive Components

Potential field consists of:

1. an attractive component U_a, which pulls the robot towards the goal;
2. and a repulsive component U_r, which pushes the robot away from the obstacles.





(Picture source: Springer Handbook of Robotics ISBN 978-3-319-32552-1)

Attractive and Repulsive Components

Attractive Component:

$$\boldsymbol{U_a}(x) = \frac{1}{2} k_p (x - x_d)^2$$

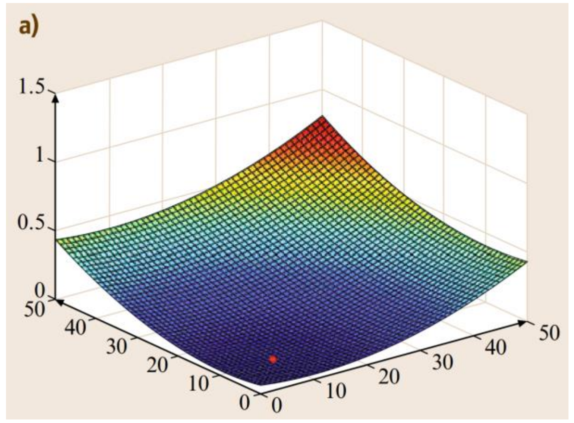
Repulsive Component:

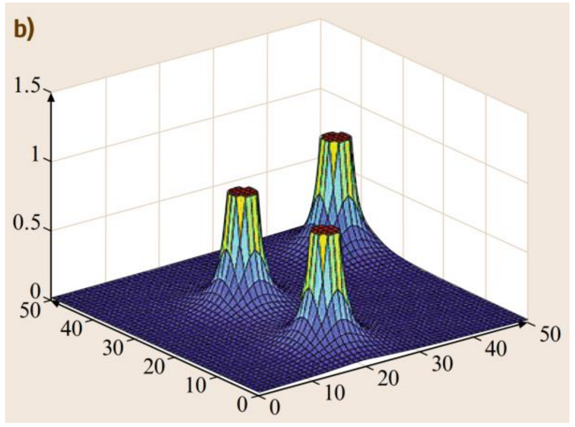
$$\boldsymbol{U_r}(x) = \frac{1}{2} \eta (1/\rho - 1/\rho_0)^2, \quad \text{if} \quad \rho \leq \rho_0$$

$\boldsymbol{U_r}(x) = 0$, otherwise

ρ : shortest distance to the obstacle

ρ_0: limit distance of the potential field influence



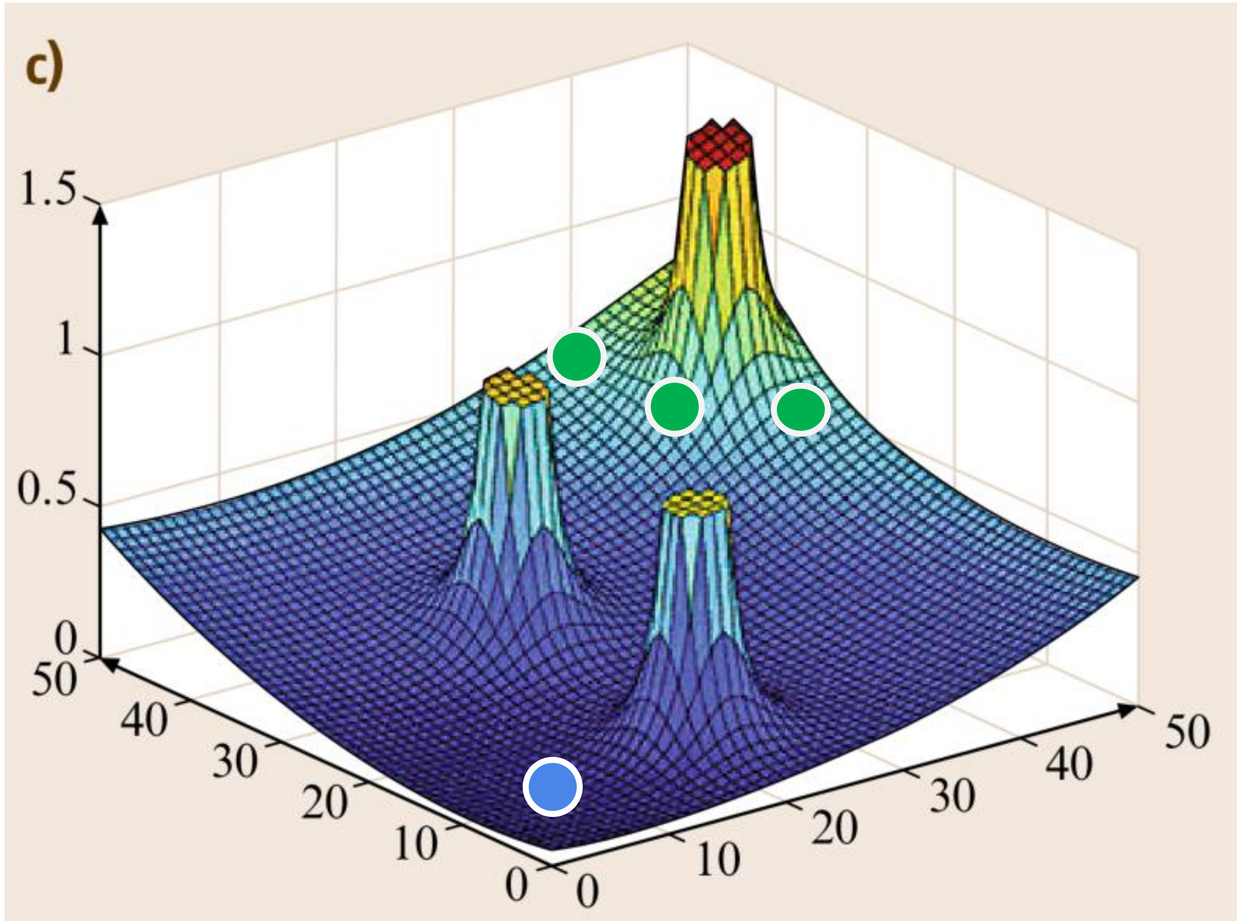


Potential Field: attractive + repulsive components

Online motion planning with PF

Input: Function, $\nabla U(q)$
Output: Sequence $[q(0), q(1), ...q(i)]$

```
1. $q(0) = q_{start}$
2. i = 0
3. while $\nabla U(q(i)) \neq 0$
4.     $q(i+1) = q(i) + \alpha(i)\nabla U(q(i))$
5.     $i = i + 1$
6. end while
```

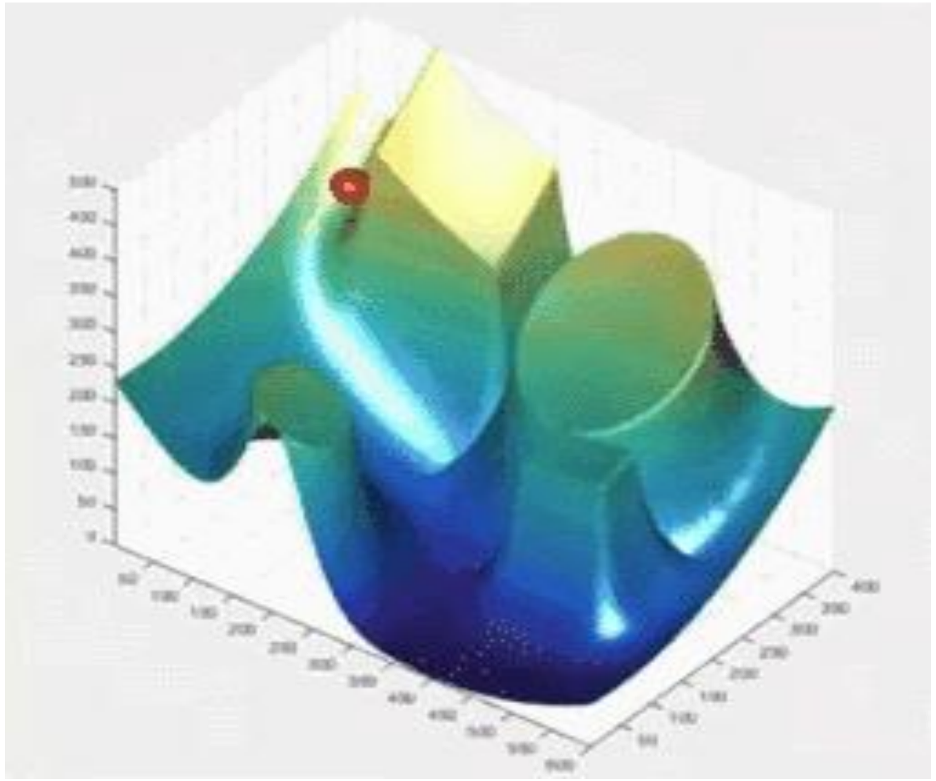


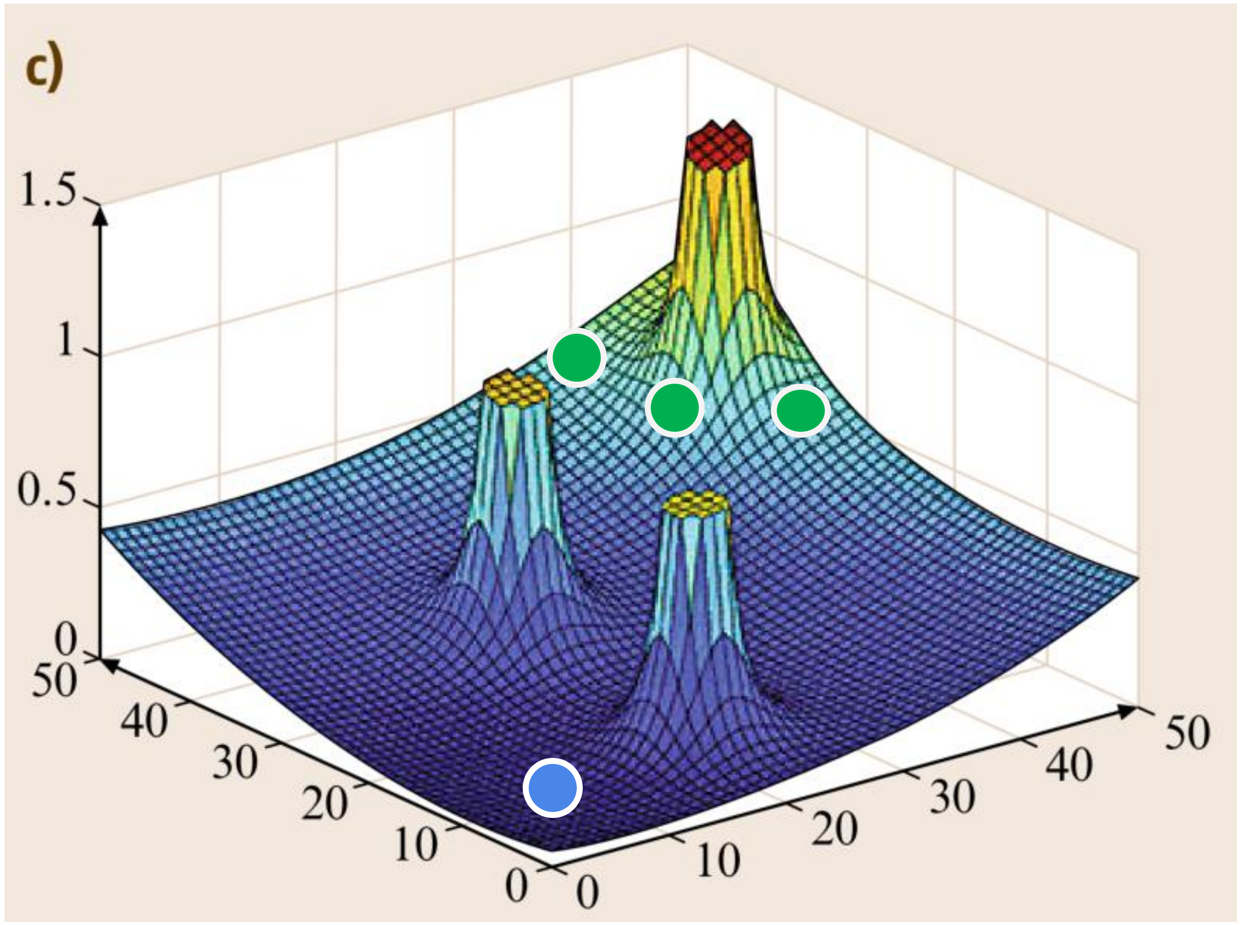
(Picture source: Springer Handbook of Robotics ISBN 978-3-319-32552-1)

Potential Field: attractive + repulsive components

What are the possible paths given different initial configurations?

Can you get any insight of possible drawbacks of this approach?

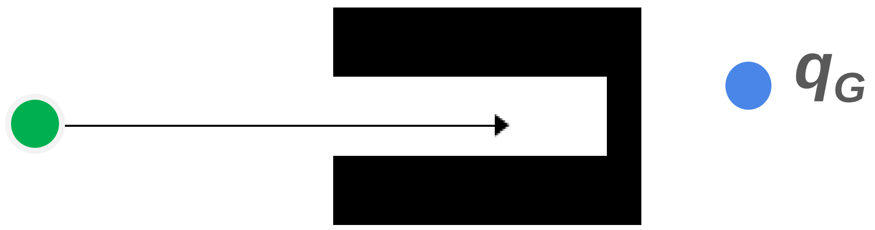


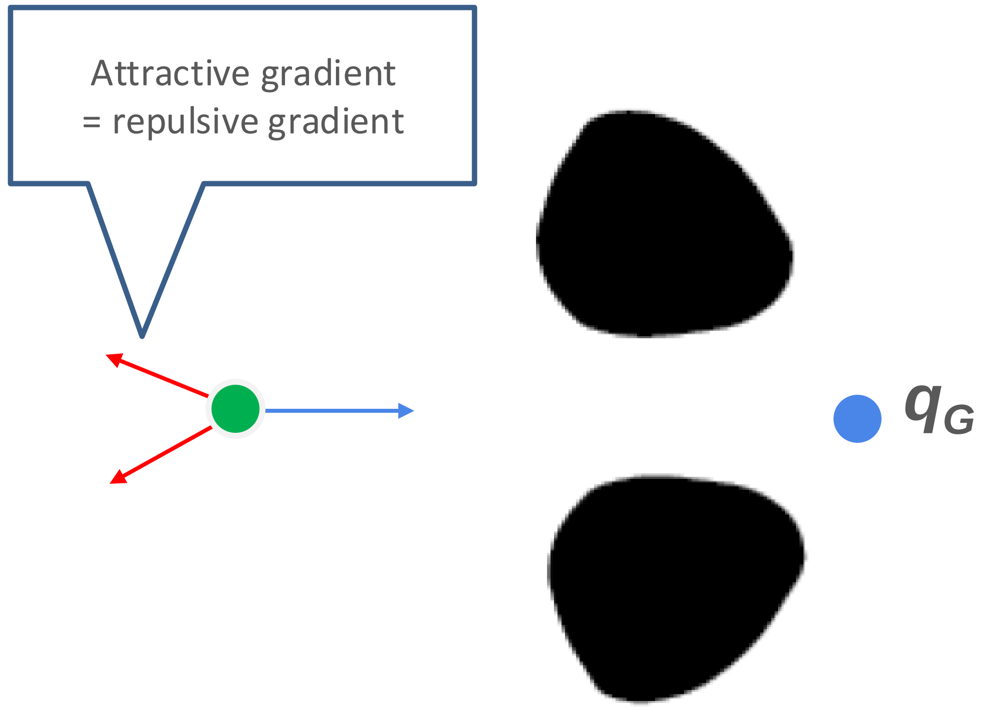


(Picture source: Springer Handbook of Robotics ISBN 978-3-319-32552-1)

Limitations: local minima

- ❑An issue with all gradient descent procedures: trapped in **local minima**.

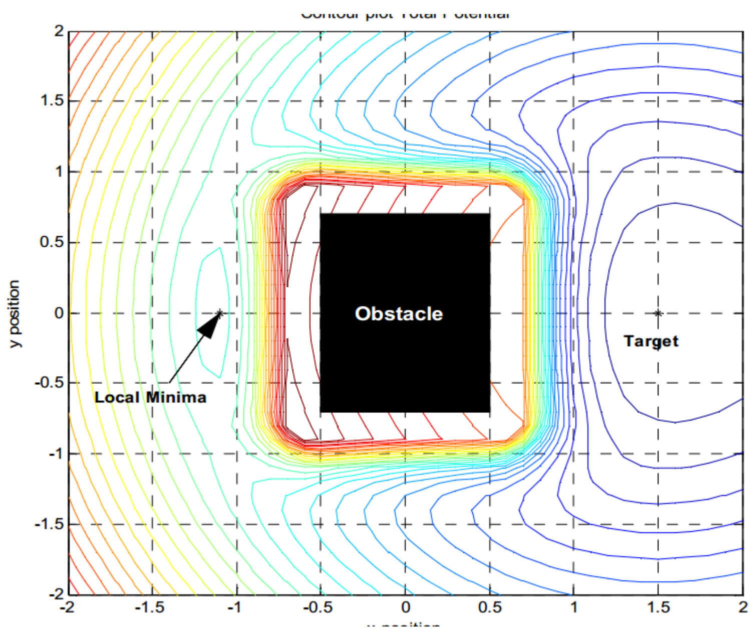


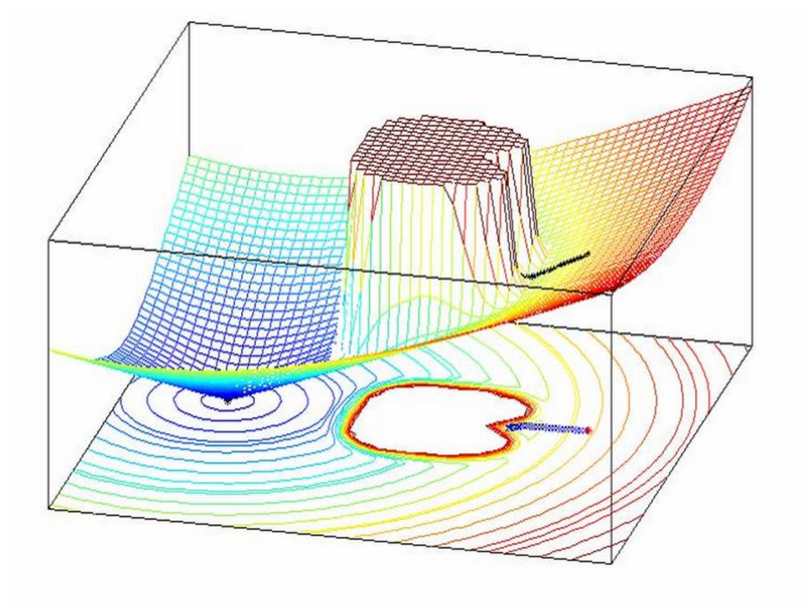


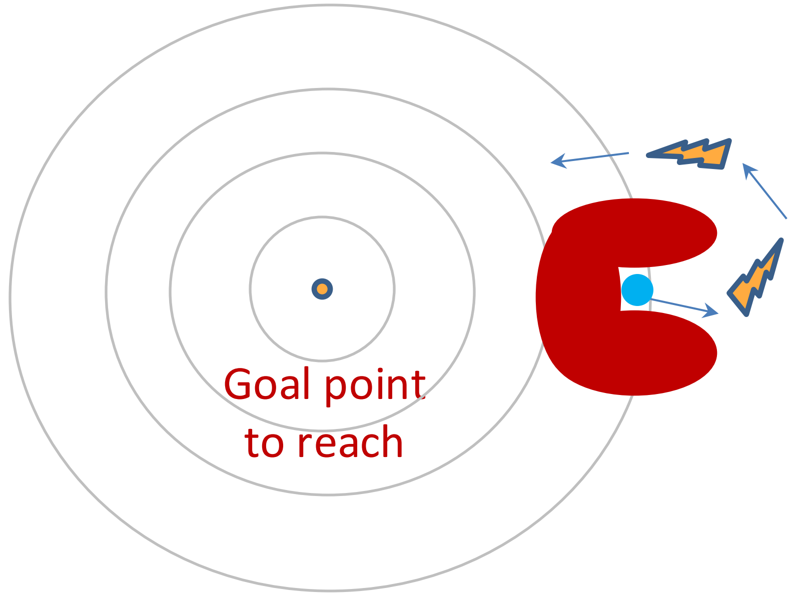
- ❑This issue is not limited to concave obstacles acting as traps, e.g., see the dual obstacles.

- ❑There are solutions such as adding random walks to get out of local minima, but this problem is generally better resolved by sampling based methods.

Limitations: local minima



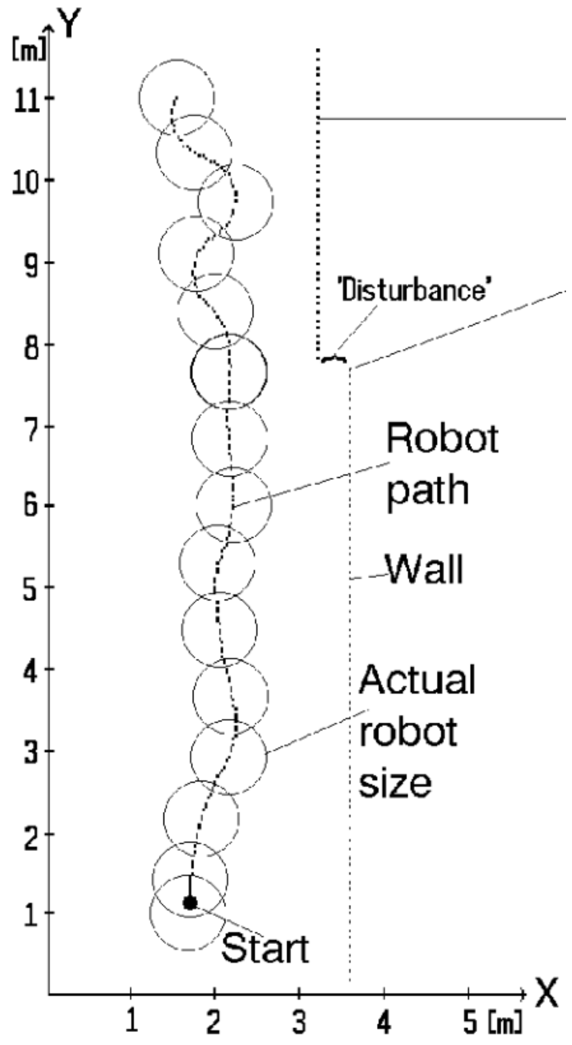


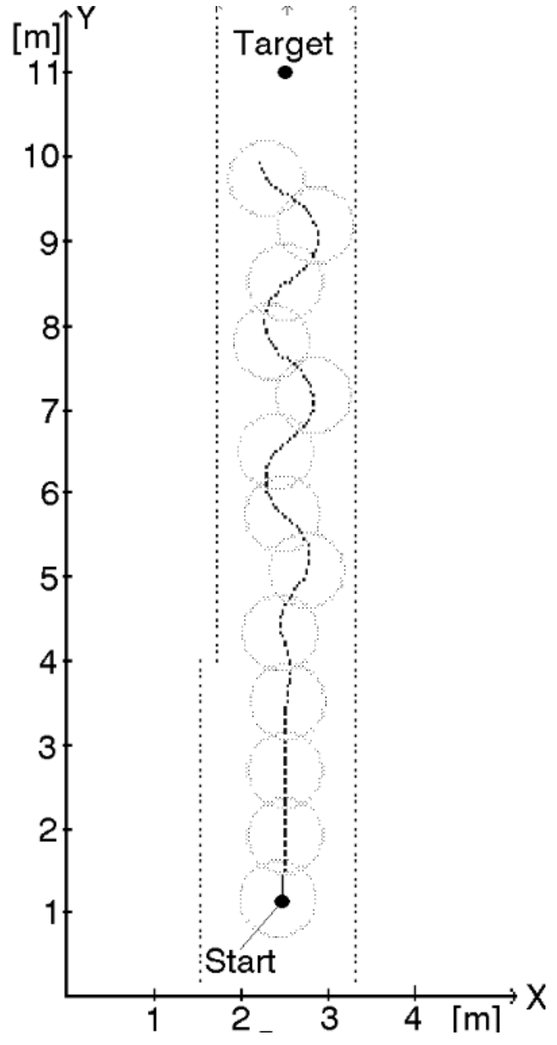


Limitations: unstable oscillation

Oscillations caused by disturbances or the discontinuity of the obstacles.

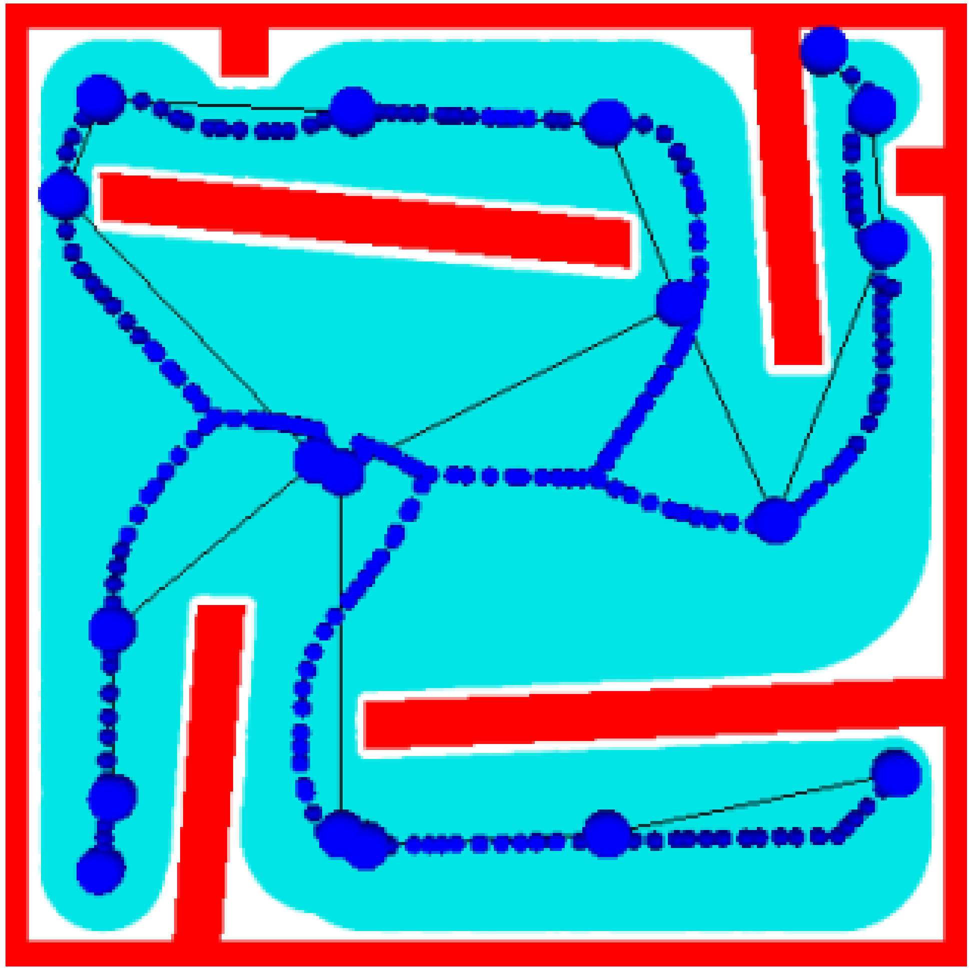
Oscillations in narrow passages: oscillation motion occurs when the robot is traveling in a narrow passage with high speed, because the robot receives repulsive forces from both side of the wall.





An example of improved potential field

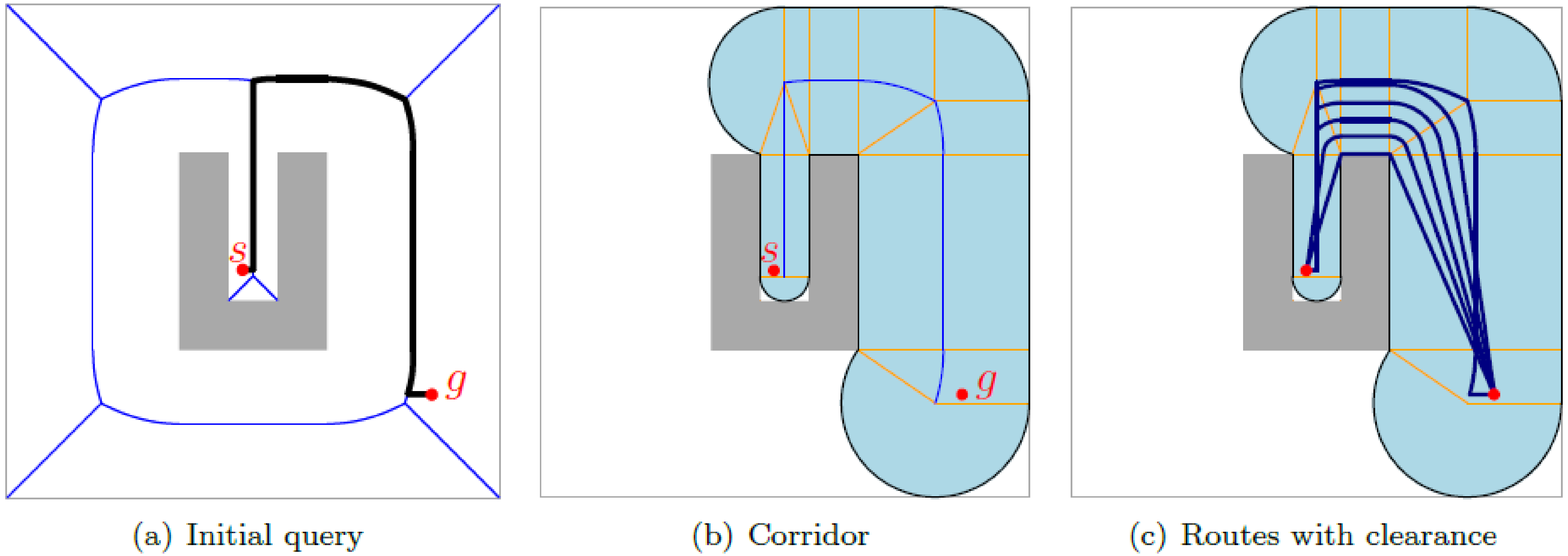
Corridor Map Method (CMM) creates collision-free corridors around static obstacles in the environment. These corridors are a network of safe paths, or key via points which the robot can follow. The corridors are typically computed algorithmically based on the layout of the environment and the locations of obstacles.



[The Corridor Map Method: Real-Time High-Quality Path Planning (uu.nl)]

Corridor Map Method (CMM)

Corridors: a network of safe paths, key via points which the robot can follow.



(a) Initial query (b) Corridor (c) Routes with clearance

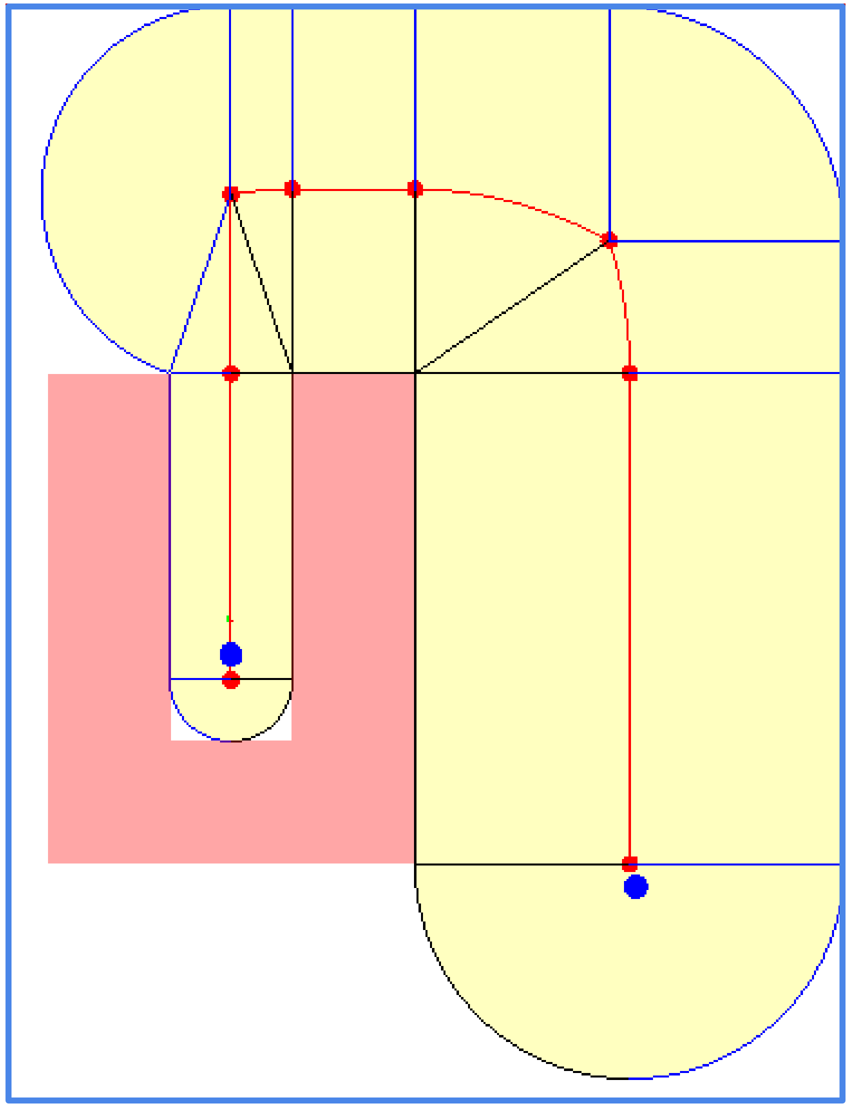
[The Explicit Corridor Map: A Medial Axis-Based Navigation Mesh for Multi-Layered Environments]

An example of improved potential field

Indicative Route Method (IRM) uses a control path to guide towards the target.

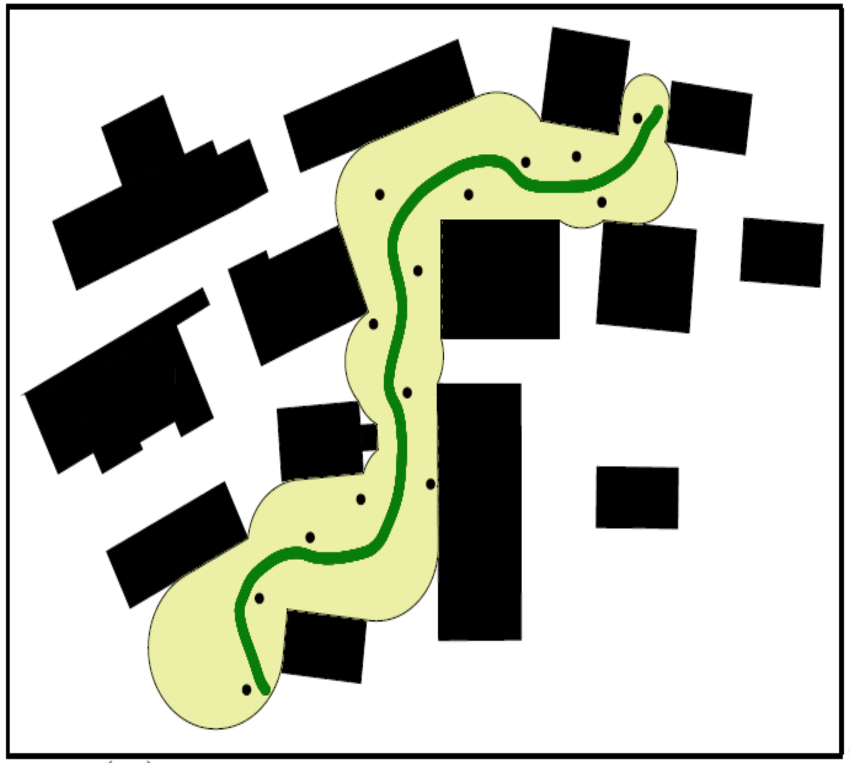
- ❑ This control path can be *manually* designed or *automatically computed* by a higher-level path planning approach.
- ❑ The control path essentially acts as a sequence of key via points that guide the robot towards the target.

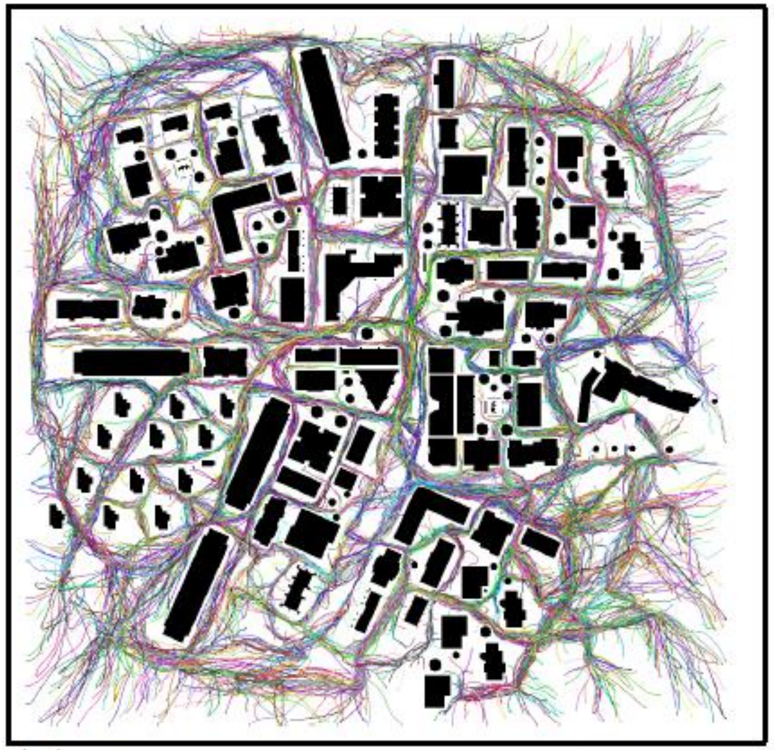
I. Karamouzas, R. Geraerts, M. Overmars. Indicative Routes for Path Planning and Crowd Simulation. In The Fourth International Conference on the Foundations of Digital Games (FDG'09) , pp. 113-120, 2009.



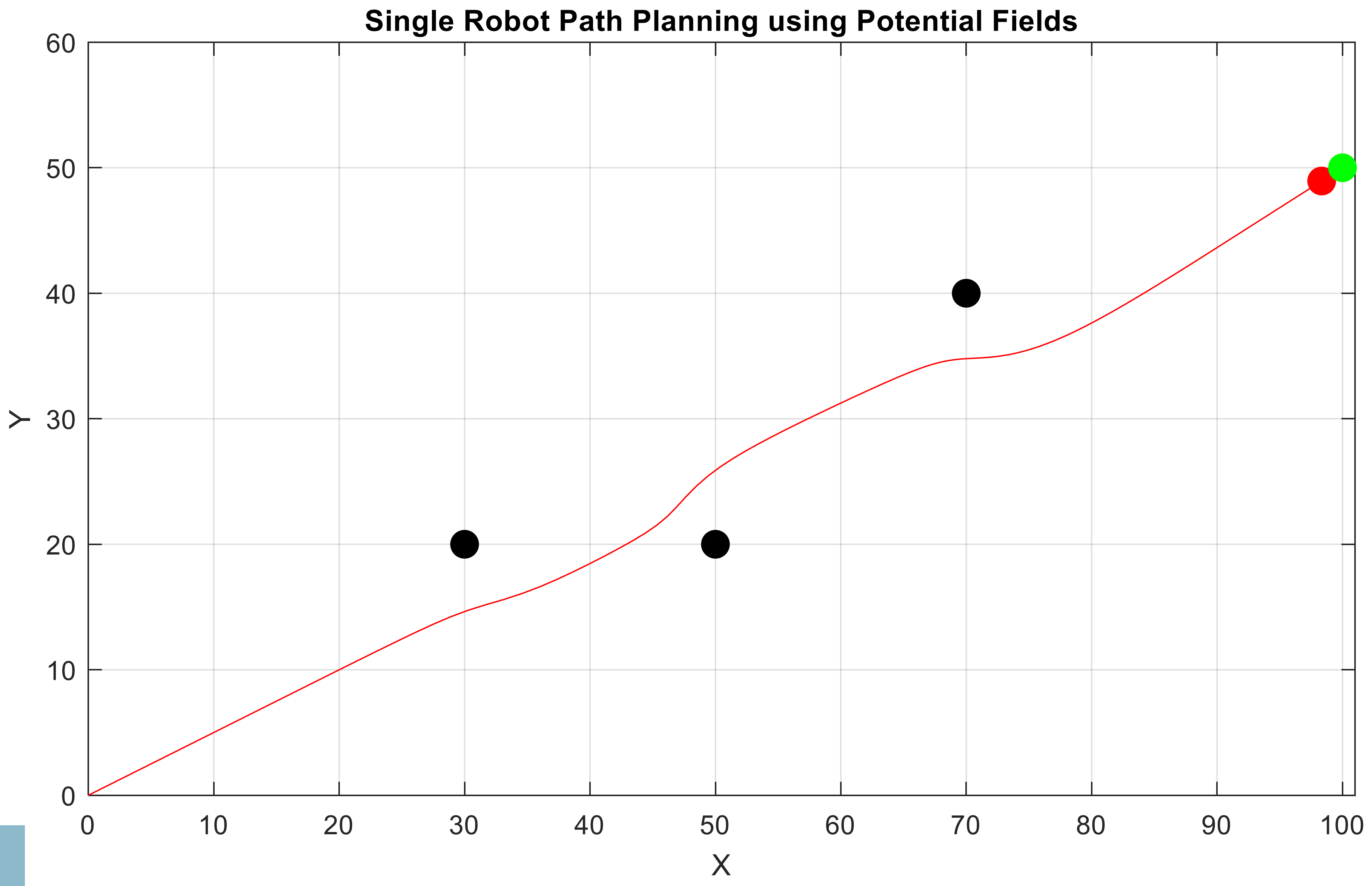
Indicative Routes for Path Planning

(a) 3D Model

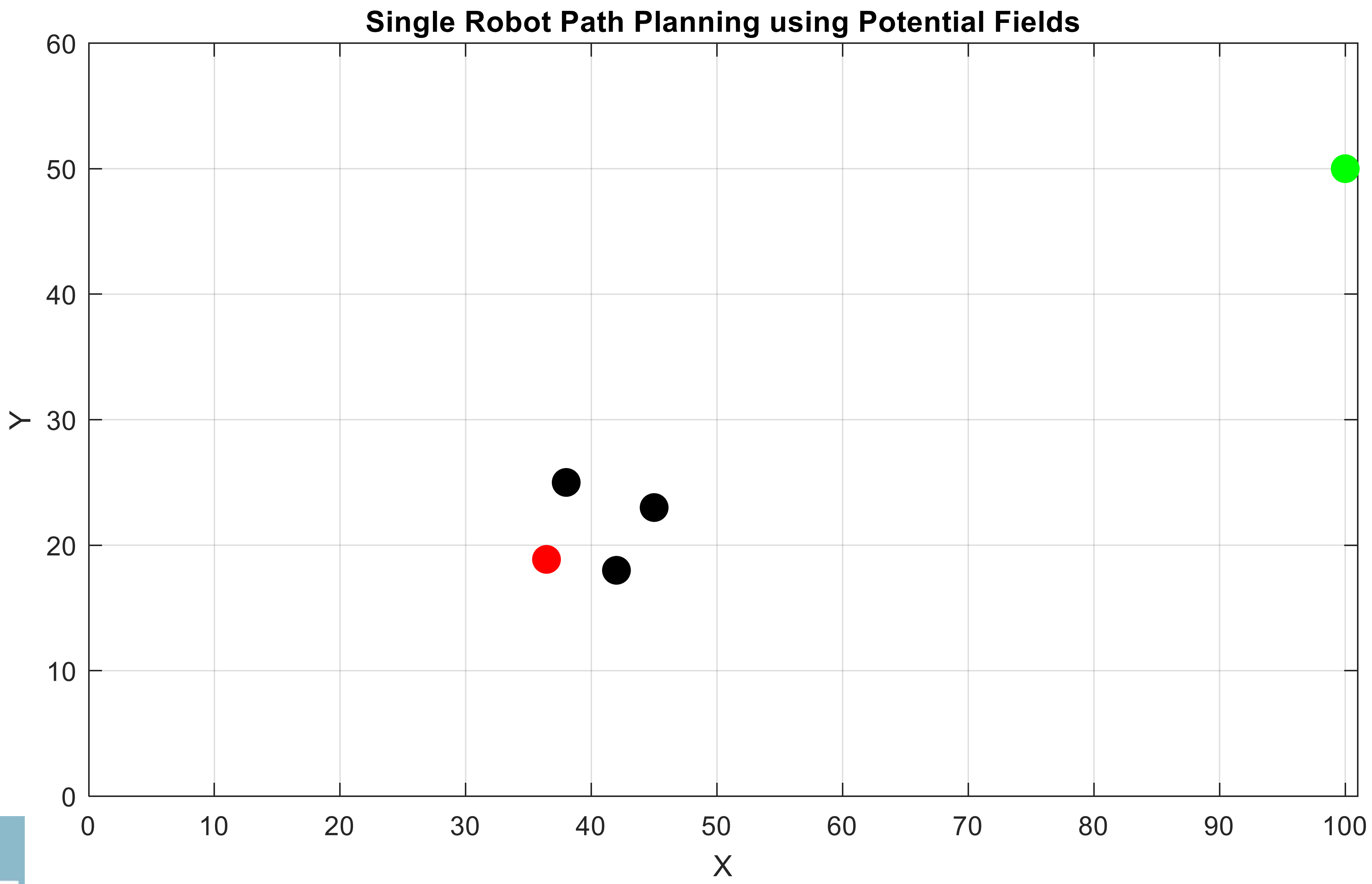

(b) Footprint


(c) 2000 paths generated by the IRM

Case study


Single Robot Path Planning using Potential Fields
60
50
40
30
20
10
0
Y
0
10
20
30
40
50
60
70
80
90
100
X

How to get out of the trap?



Outline

- Motion planning:
  - ❑ Goals and challenges
  - ❑ Concepts and definitions
- Continuous planning: Potential Fields (PF)
- Sampling-based planning:
  - ❑ Rapidly exploring Random Tree (RRT): principle and code demo
  - ❑ Variants of RRT algorithms
  - ❑ Multi-Query Planner: Probabilistic Roadmap (PRM)
  - ❑ Grid-based search algorithms: A*

RRT

Sampling-based Planners

- A standard for sampling-based planners is to provide a complete/feasible solution if a solution path indeed exists, then the planner will eventually find it.
- Sampling-based approach is not meant to search for the optimal solution, rather it is sub-optimal.

Common categories for the sampling-based planners are:

- Single-Query Planners: incremental search; a single set of initial-goal is given to the planning algorithm;
- Multi-Query Planners: mapping the connectivity of C-free.

Single-Query Planners

- This class of methods probe and search the continuous C-space by: 1. extending tree data structures initialized at these known configurations; 2. connecting them eventually.
- RRT is a single-query planner.

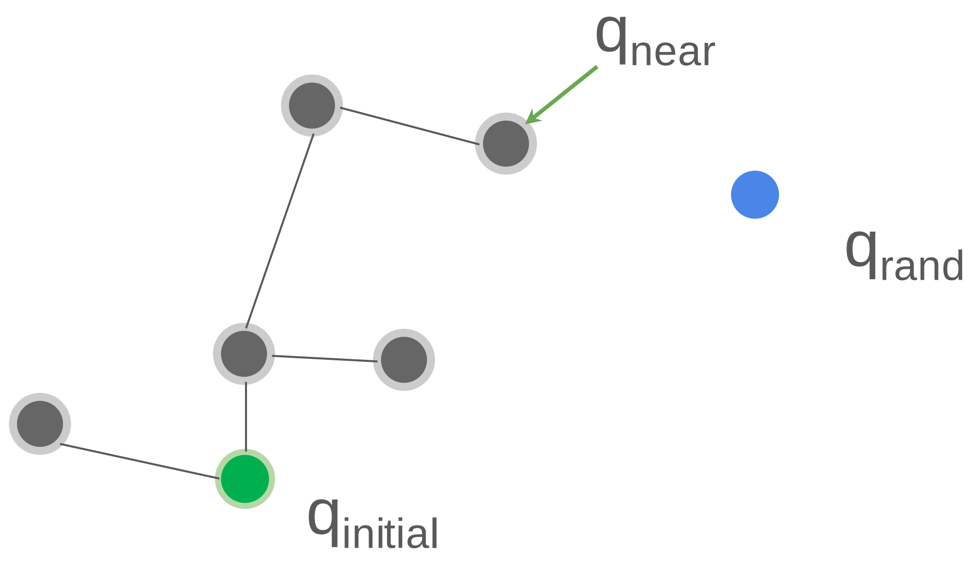
Picture by Javed Hossain

Basic RRT algorithm

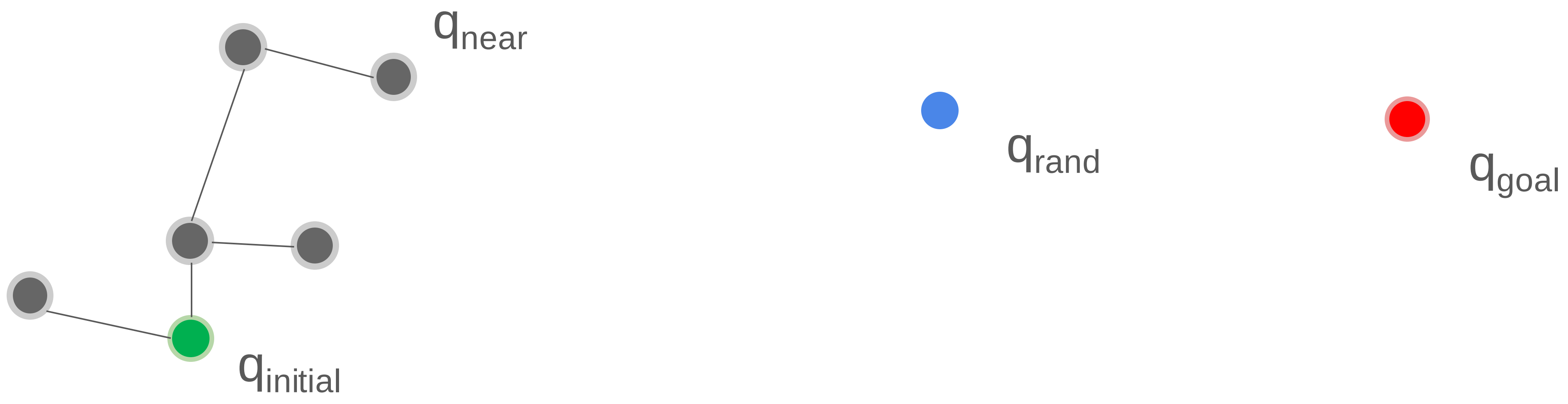
Pseudo code

Algorithm 1 BUILD_RRT(q_{init})

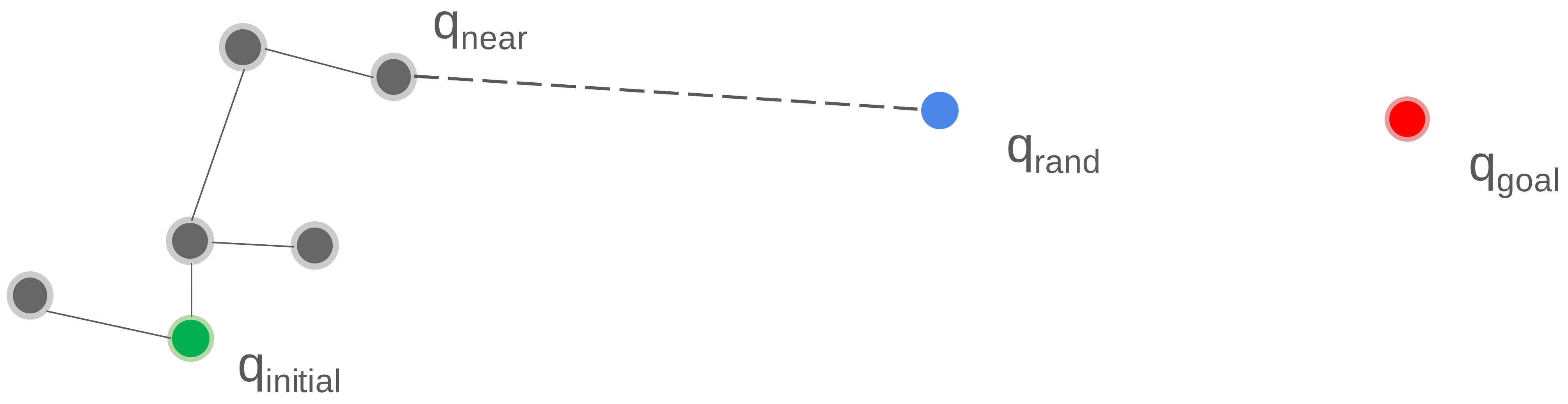
```
$\mathcal{T}$.init($q_{init}$);
for $k = 1$ to $K$ do
    $q_{rand} \leftarrow$ RANDOM_CONFIG();
    $q_{near} \leftarrow$ NEAREST_NEIGHBOR($q_{rand}, \mathcal{T}$);
    if edge_valid($q_{rand}, q_{near}$) then
        $\mathcal{T}$.add_vertex($q_{rand}$);
        $\mathcal{T}$.add_edge($q_{near}, q_{rand}$);
        if close_to_goal($q_{rand}$) then
            return SUCCESS
return FAILURE
```



Basic RRT algorithm

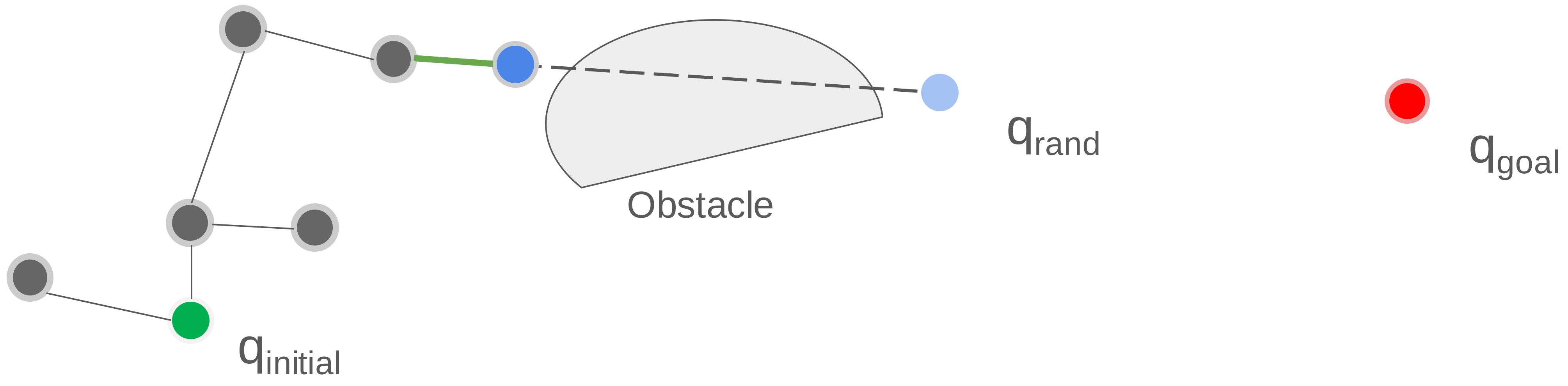


Basic RRT algorithm



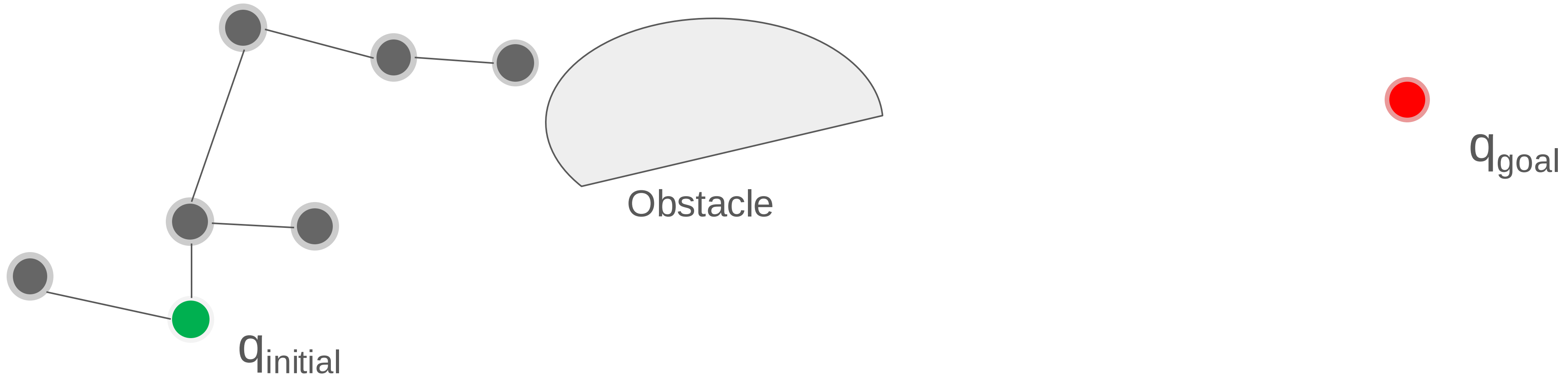
Basic RRT algorithm

- If there is an obstacle lying between qnear and qrand, the edge travels up to the obstacle boundary, as far as allowed by the collision check algorithm.



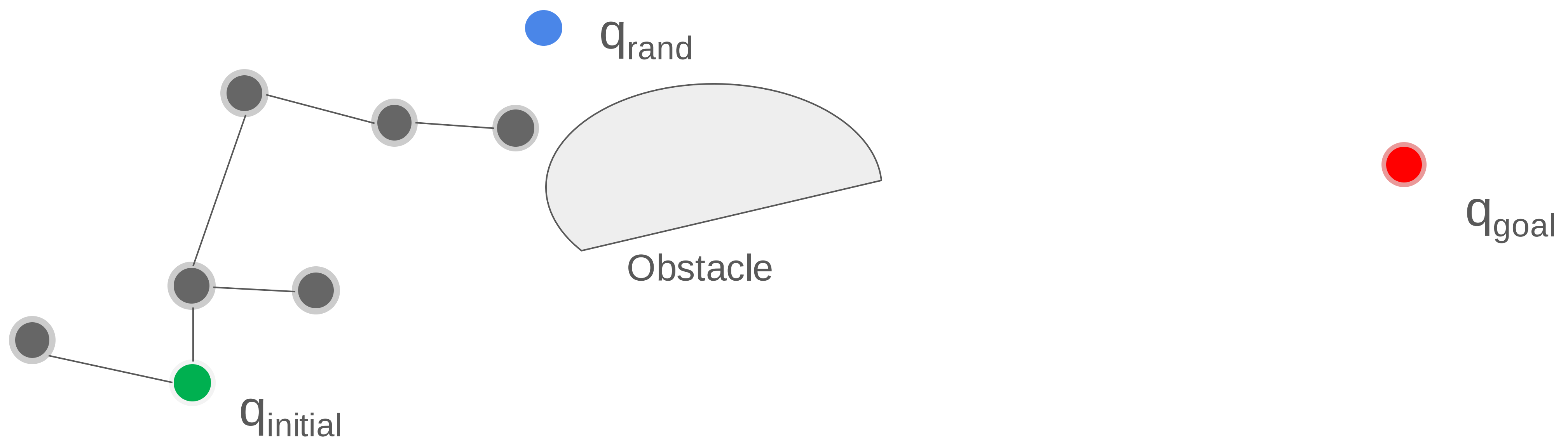
Basic RRT algorithm

- If there is an obstacle lying between qnear and qrand, the edge travels up to the obstacle boundary, as far as allowed by the collision check algorithm.

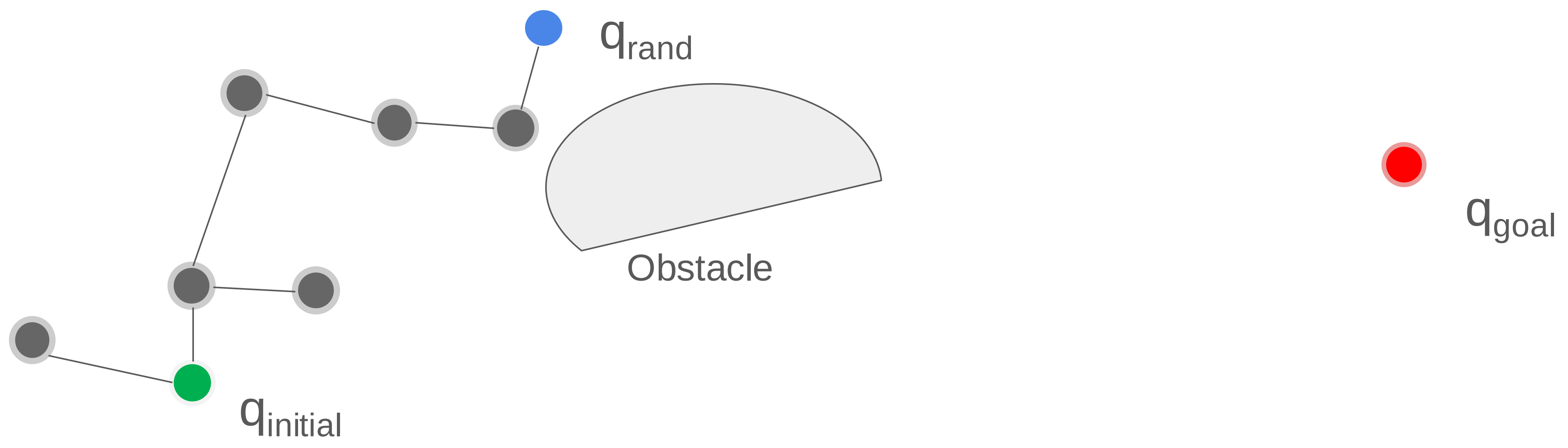


Basic RRT algorithm

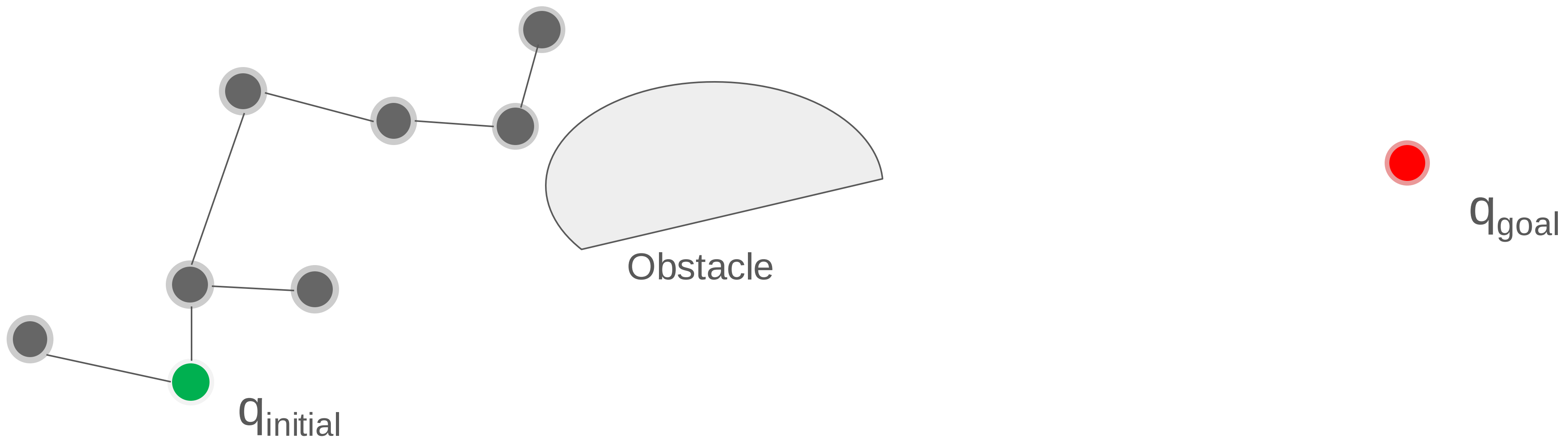
- If there is an obstacle lying between qnear and qrand, the edge travels up to the obstacle boundary, as far as allowed by the collision check algorithm.



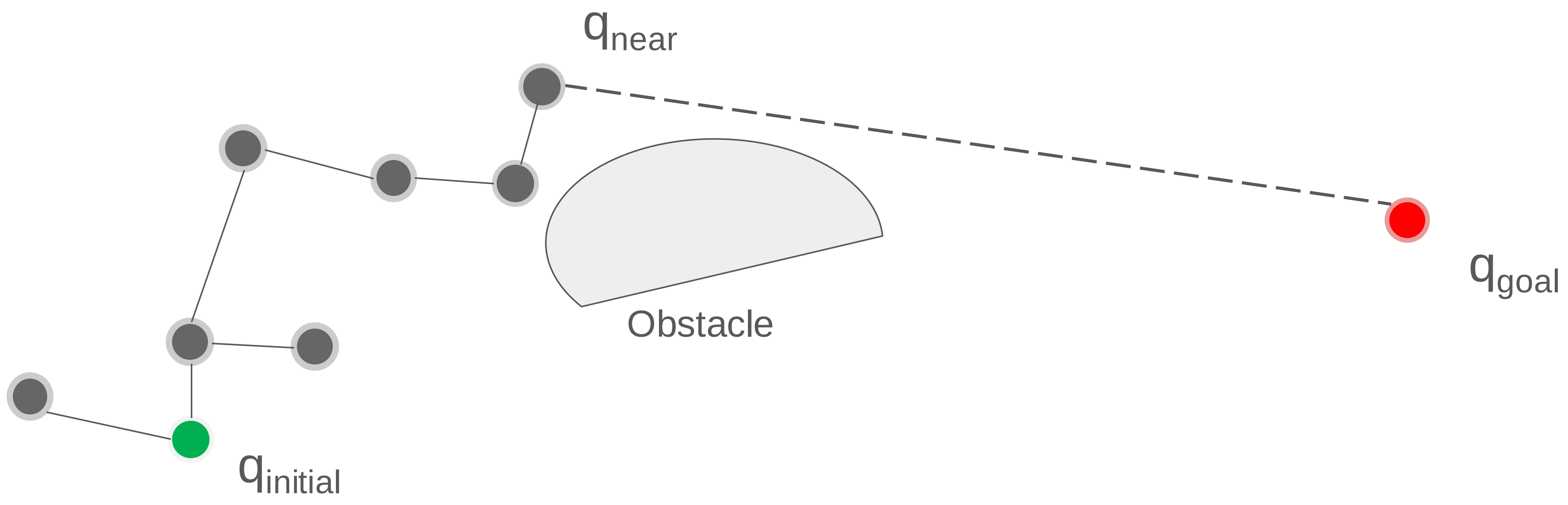
Basic RRT algorithm



Basic RRT algorithm

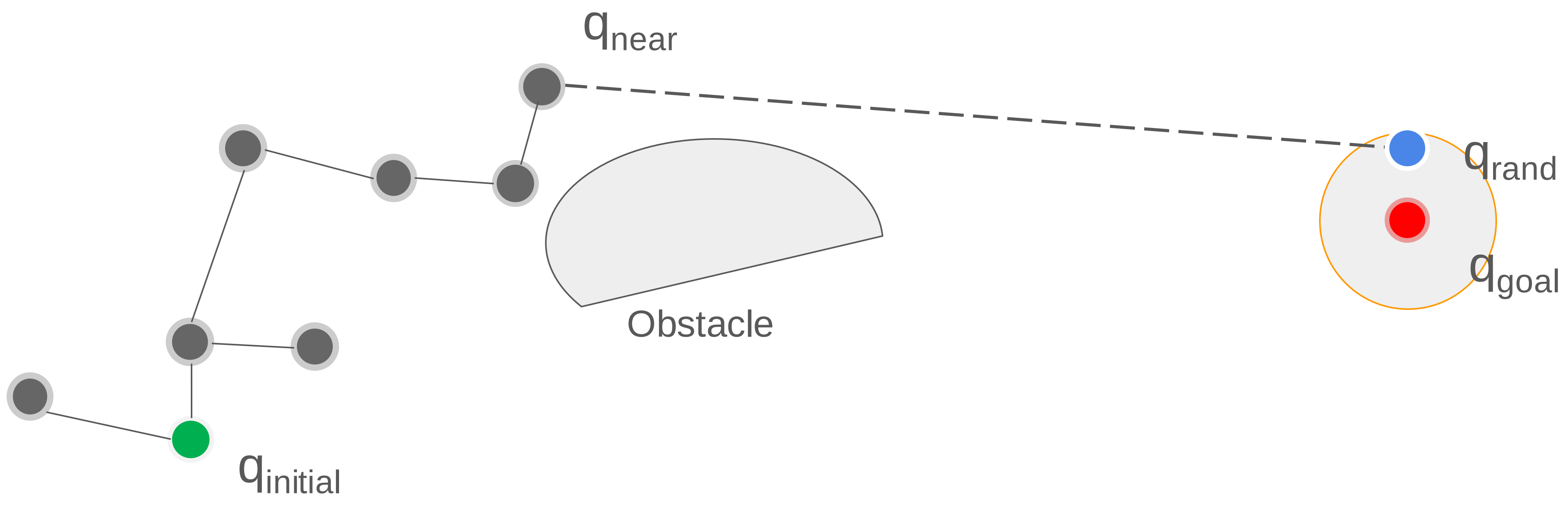


Basic RRT algorithm



To enable fast convergence, force $q_{rand}=q_{goal}$ for every *n*th iteration.

Basic RRT algorithm



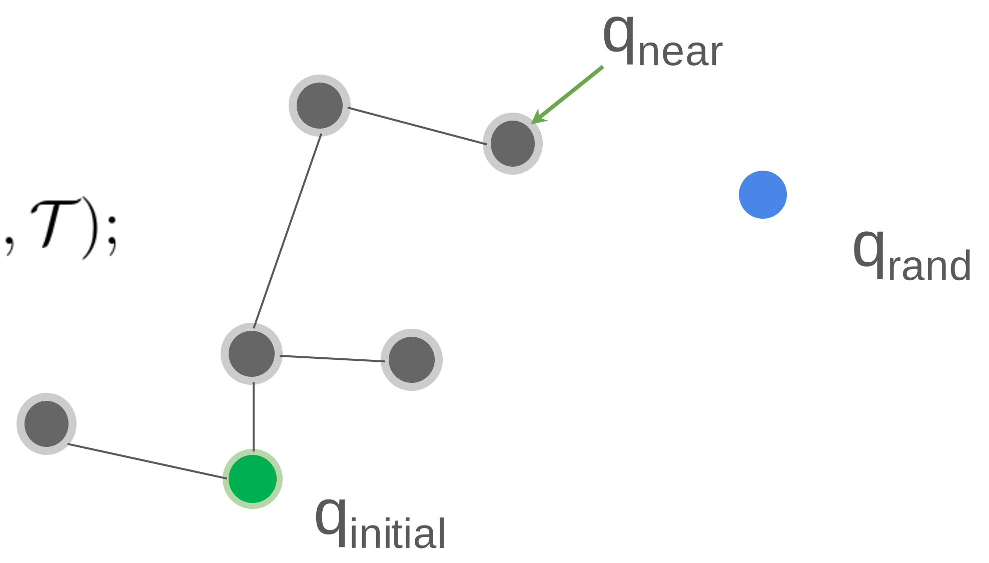
To enable fast convergence, define a collision free region, the search succeeds if a collision free pair (q_{rand} , q_{near}) falls into this safe region.

Basic RRT algorithm

Pseudo code

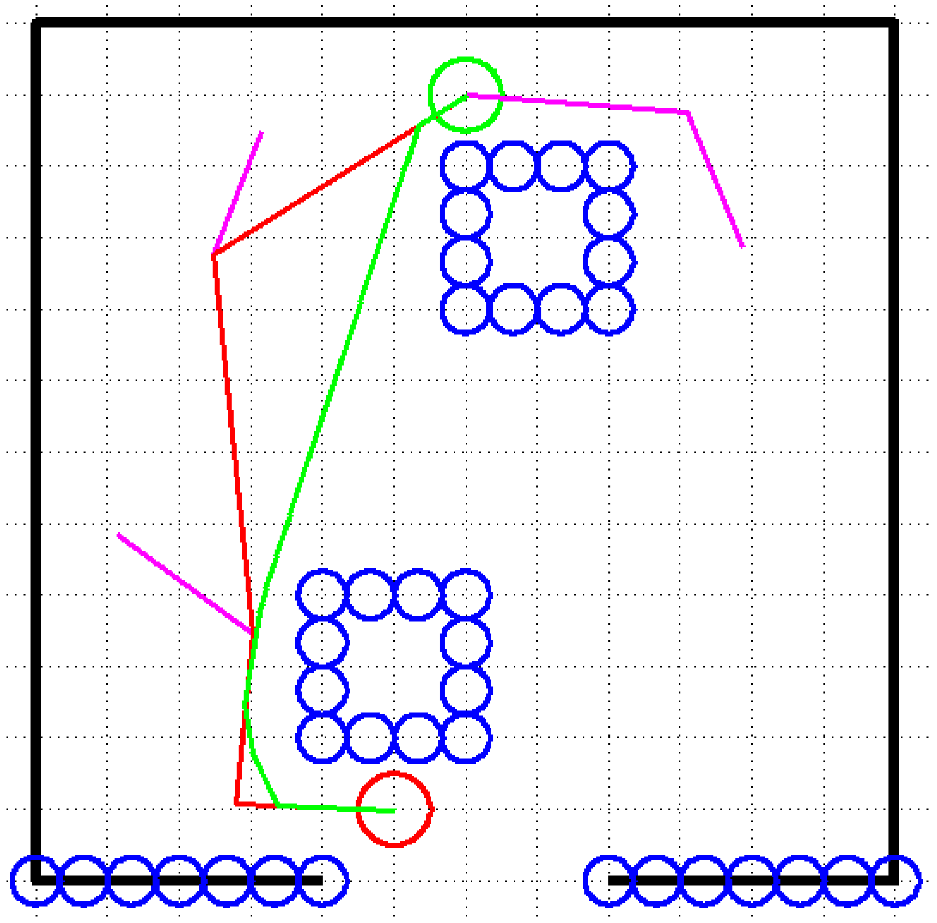
Algorithm 1 BUILD_RRT(q_{init})

```
$\mathcal{T}$.init($q_{init}$);
for $k = 1$ to $K$ do
    $q_{rand} \leftarrow$ RANDOM_CONFIG();
    $q_{near} \leftarrow$ NEAREST_NEIGHBOR($q_{rand}, \mathcal{T}$);
    if edge_valid($q_{rand}, q_{near}$) then
        $\mathcal{T}$.add_vertex($q_{rand}$);
        $\mathcal{T}$.add_edge($q_{near}, q_{rand}$);
        if close_to_goal($q_{rand}$) then
            return SUCCESS
return FAILURE
```



RRT Example

- In this example, number of iterations is 21.

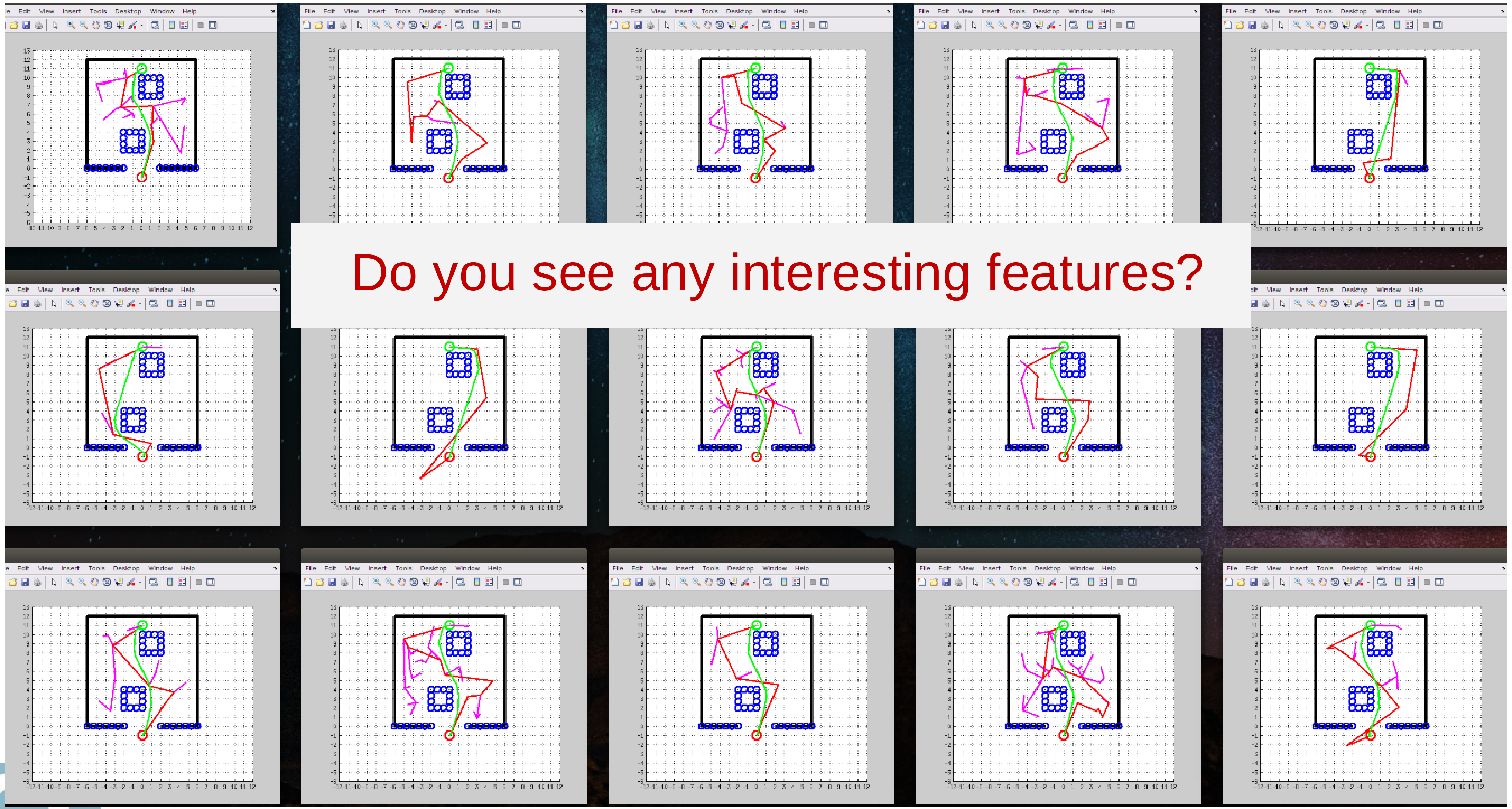


🟢 : initial

🔴 : goal

Code source: University of Illinois at Urbana-Champaign,Control Systems Laboratory for use in GE423 - Mechatronics and research projects: http://coecsl.ece.illinois.edu/ge423/spring13/RickRekoskeAvoid/rrt.html

Same initial & final positions for 15 runs

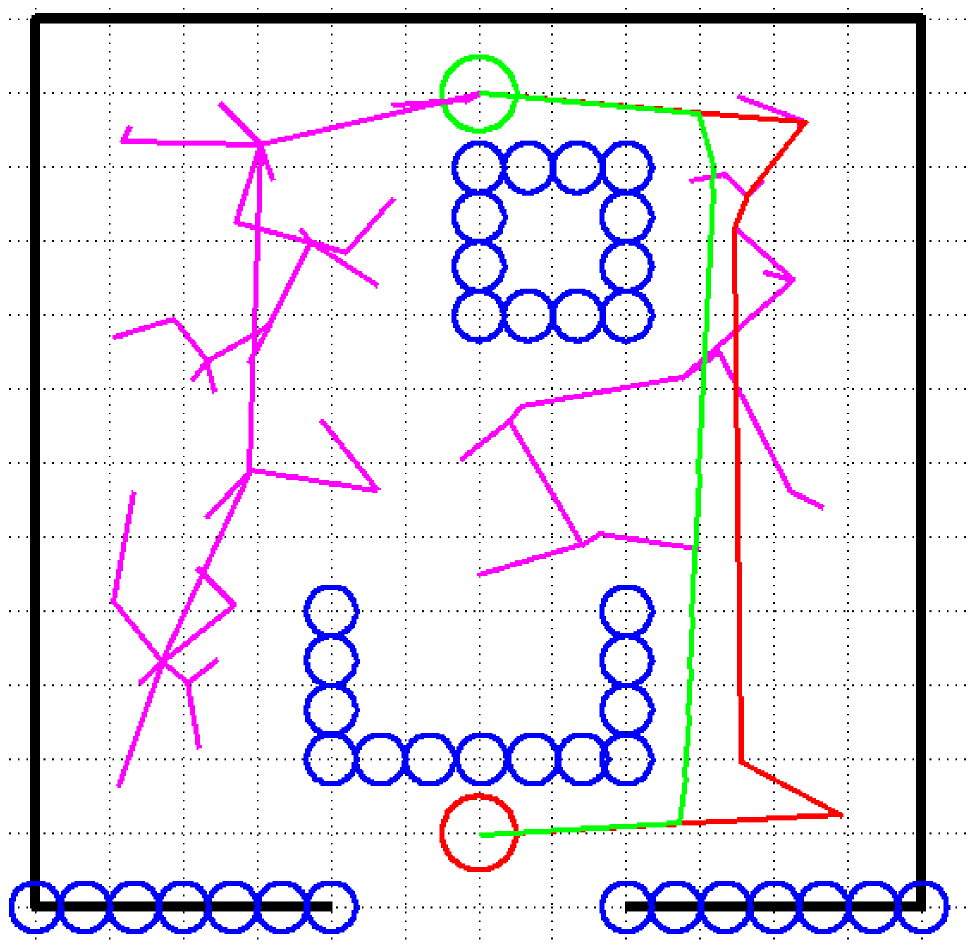


Do you see any interesting features?

RRT Example

- We create a local minima problem encountered in the potential field approach.

- In this example, number of iterations is 114.

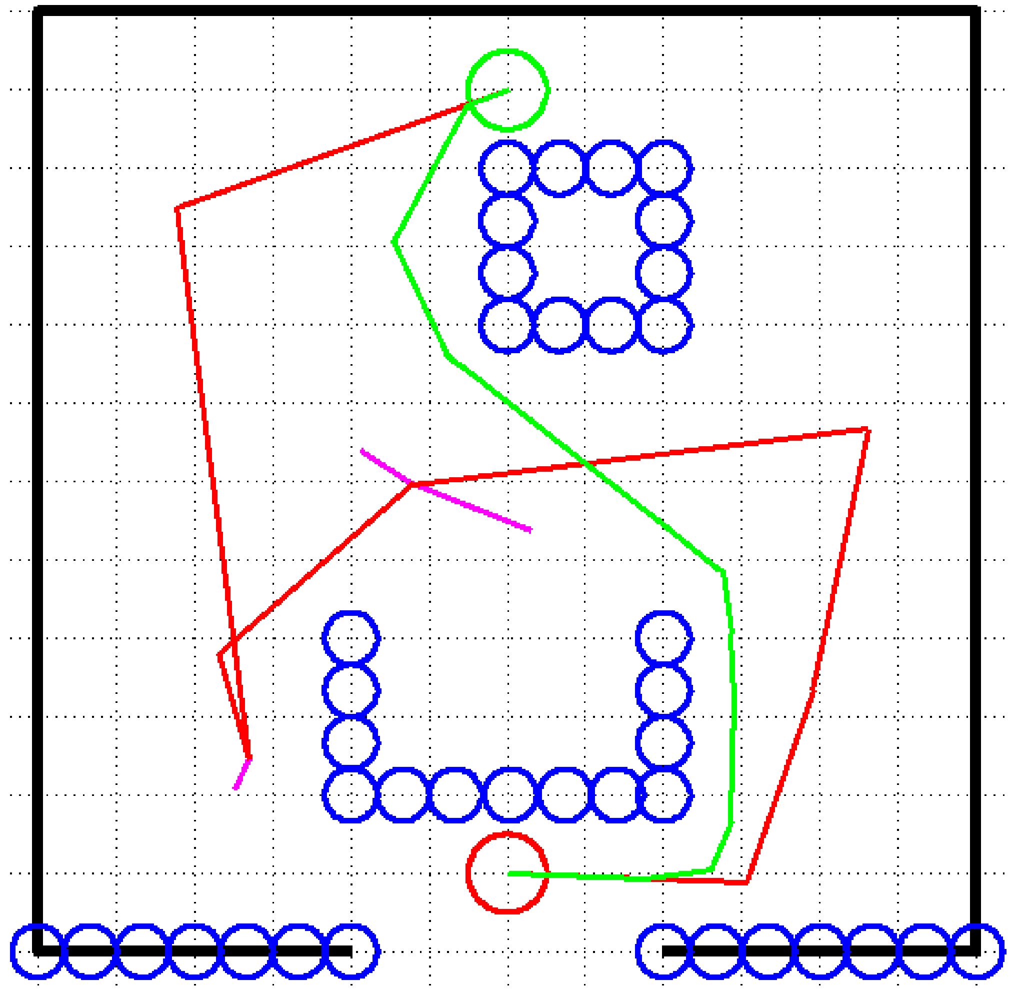


RRT Example

- We create a local minima problem encountered in the potential field approach.

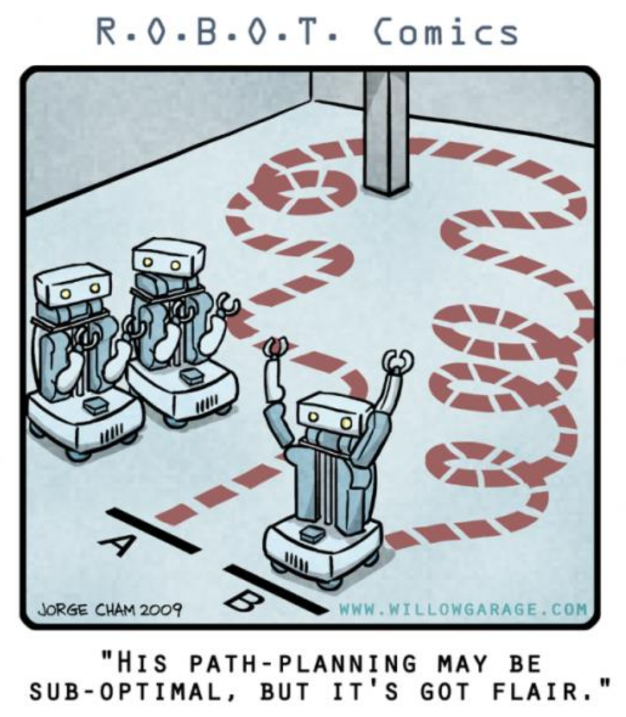
- In this example, number of iterations is 29.

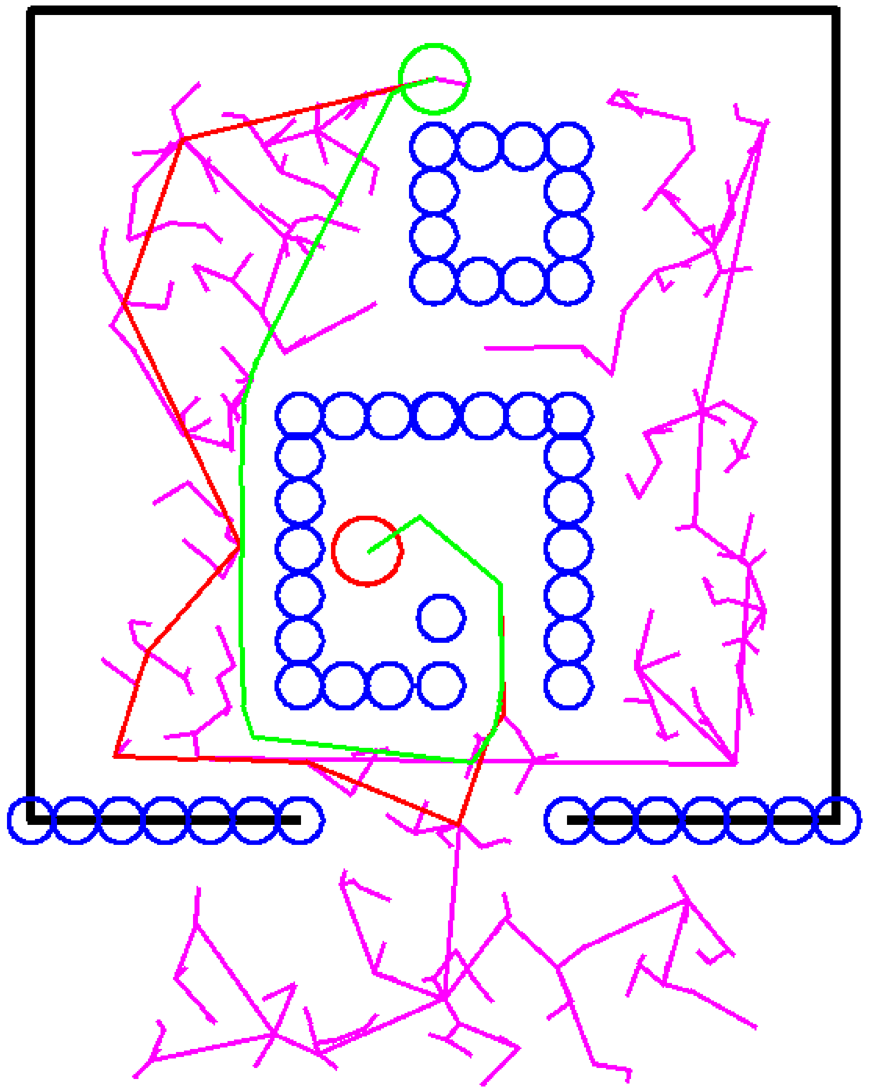
Number of iterations varies!



RRT Example

- A trap, a local minima problem encountered in the potential field.
- Number of iterations: 569.



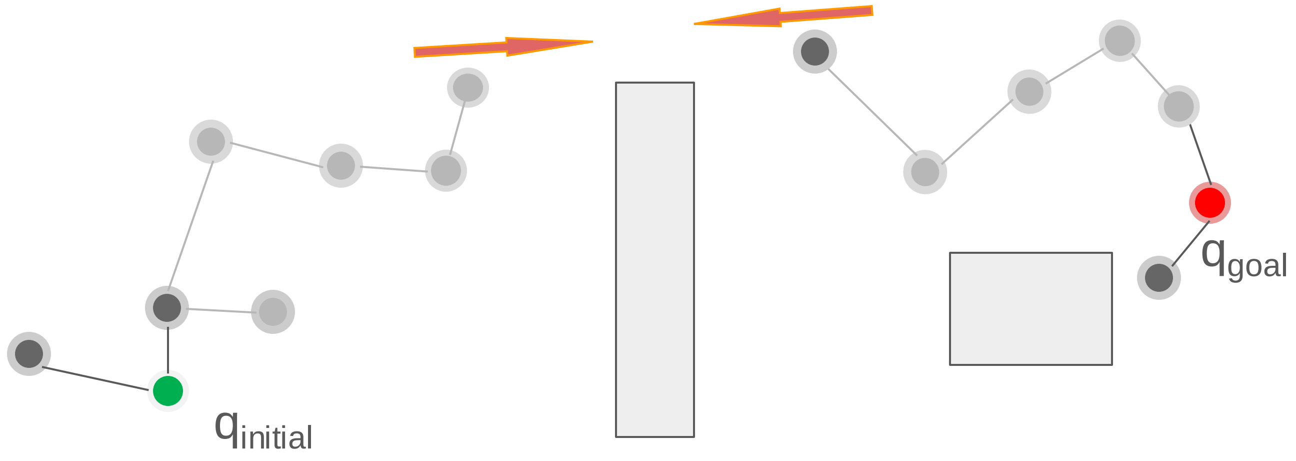


Variants of RRT algorithms

- RRT-Connect: grow and connect two RRTs
- Rapidly-exploring random graph (RRG) and RRT*: a variant of RRT that converges towards an optimal solution
- RRT*-Smart:
- A*-RRT and A*-RRT*
- RRT*FN: RRT* with a fixed number of nodes, which randomly removes a leaf node in the tree in every iteration
- RRT*-AR: sampling-based alternate routes planning

RRT-Connect: grow and connect two RRTs

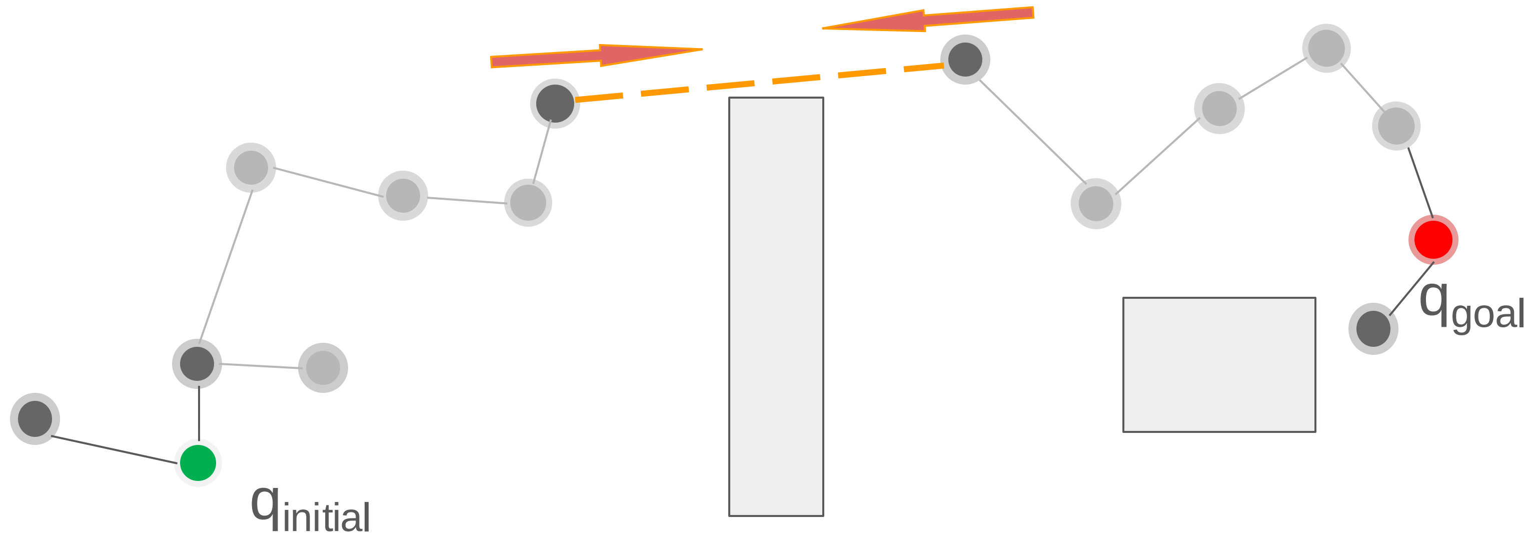
- A simple greedy heuristic that aggressively tries to connect two trees, one from the initial configuration and the other from the goal.



- The idea of constructing search trees from the initial and goal configurations comes from classical AI bi-directional search.

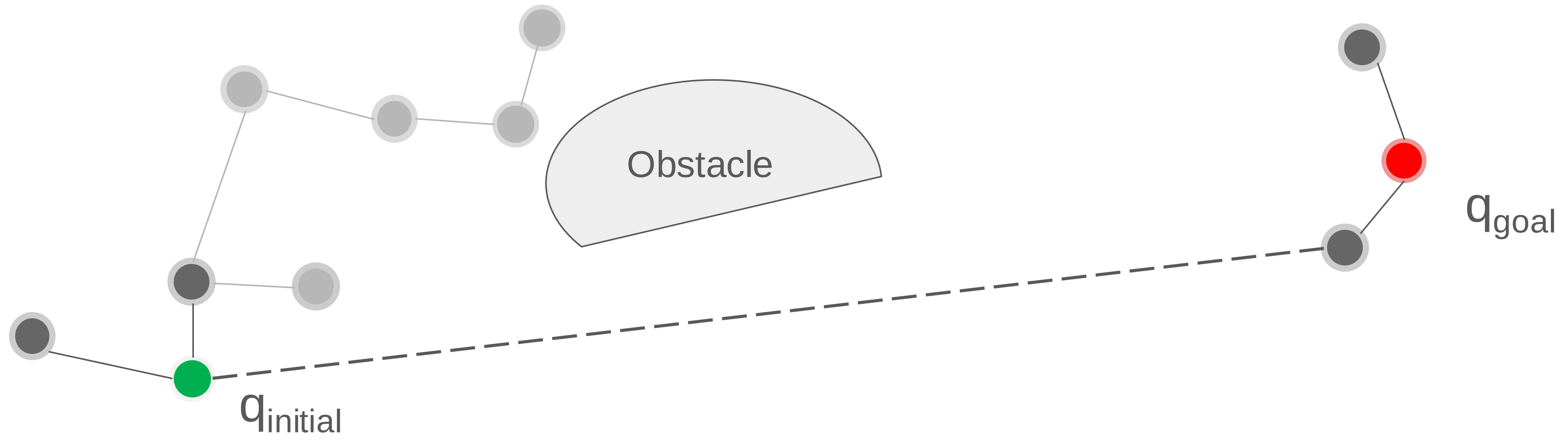
RRT-Connect

- This approach finds a graph that covers the space nicely that is independent of the query, ie RRT-Connect is still single query planning.



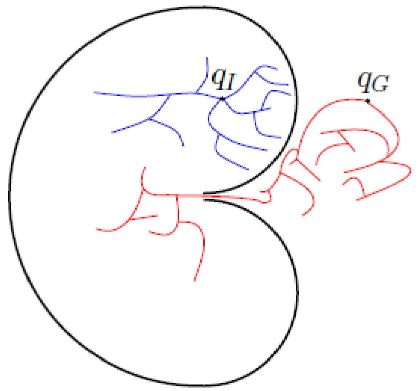
James Kuffner,, Steven LaValle, “RRT-connect: An efficient approach to single-query path planning”, IEEE International Conference on Robotics and Automation (ICRA) 2000.

RRT-Connect

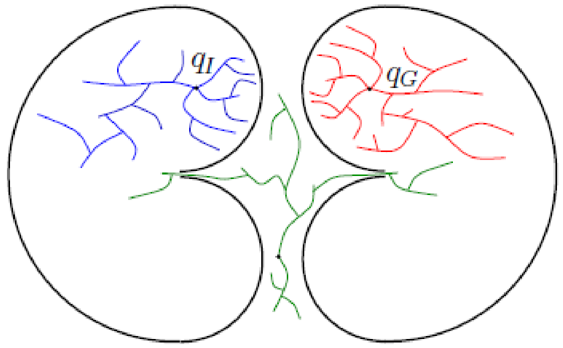


Bi-directional search

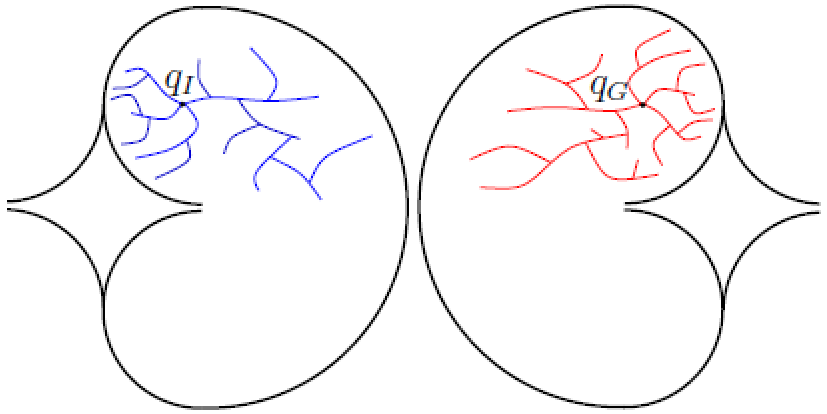
Extensions of RRT algorithms



Bi-directional search



Multi-directional search



Some problems may still be too hard!

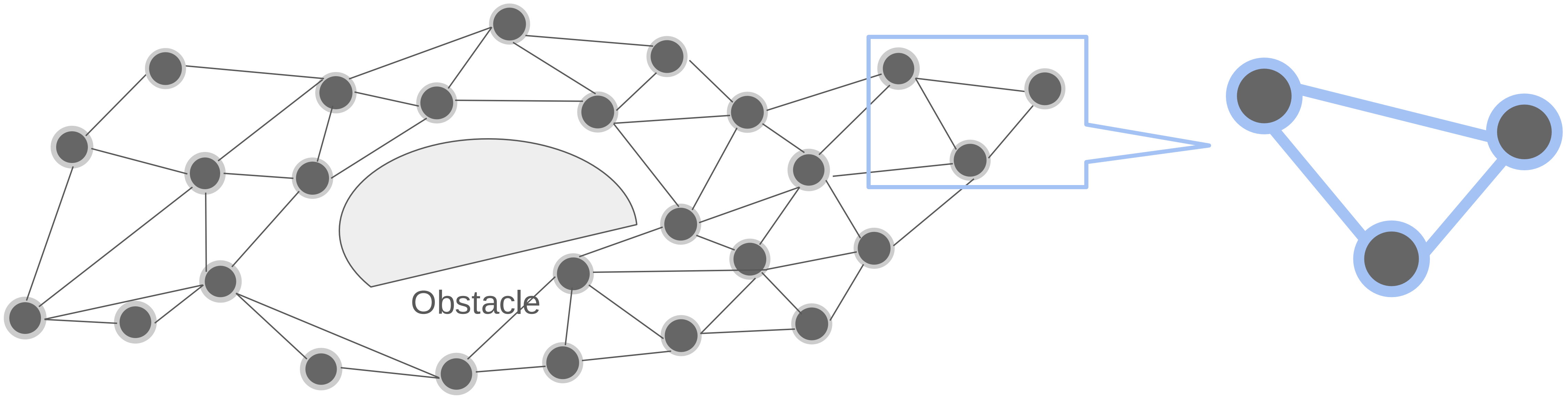
Outline

- Motion planning: goals and challenges
- An intuitive approach: Potential Fields
- Rapidly exploring Random Tree (RRT): principle and code demo
- Variants of RRT algorithms
- Multi-Query Planner: Probabilistic Roadmap (PRM)
- Grid-based search algorithms: A* Search

Probabilistic roadmap (PRM)

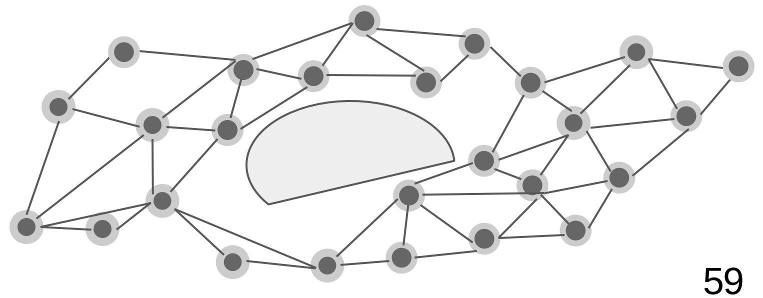
PRM: basics

- Vertex: a node or point
- Edge: connection between nodes
- Graph: a data structure that contains a set of vertices and edges.



Probabilistic Roadmap Construction

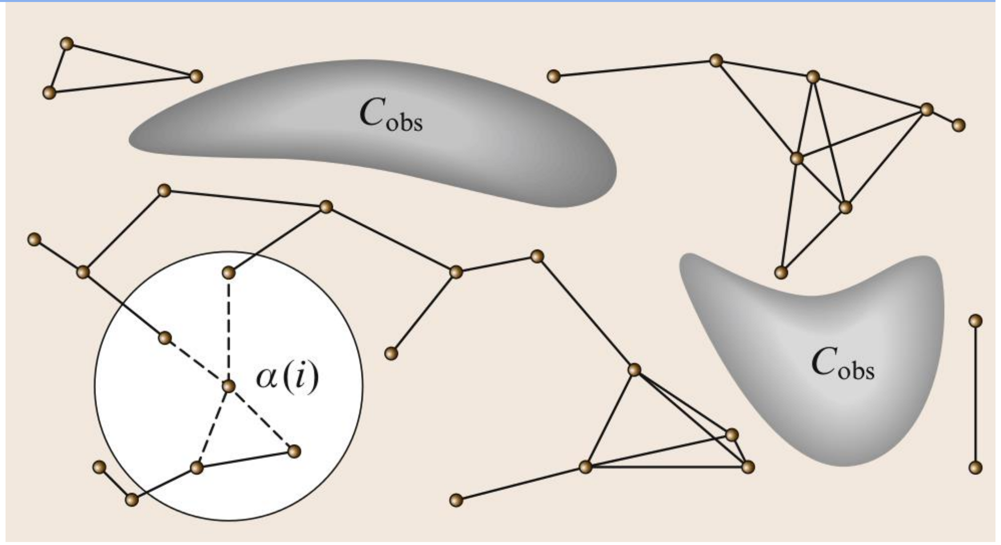
- Major steps (visualise in previous slide)
- Sampled configurations are tested for collisions, typically in workspace (may also do **C-space**, but more expensive)
- Collision-free configurations are retained as “milestones”
- Each milestone is linked by straight paths to *k* nearest neighbours (local interpolation)
- Retain only *collision free* links, which form a graph – include start and end poses as milestones
- Use **A*** or other graph traversal algorithm to find path



PRM: pseudo code

N: number of nodes to include in the roadmap

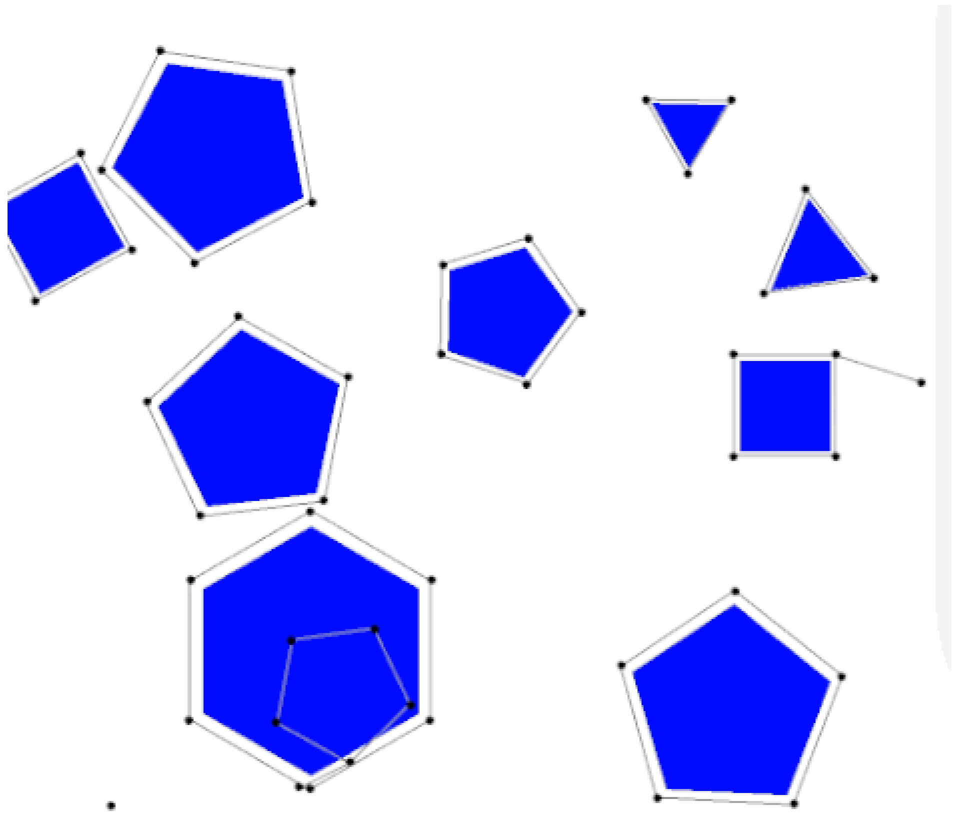
```
1:G.init(); i=0;
2:while i < N do
3:  if α(i) ∈ $C_{free}$ then
4:          G.add_ vertex(α(i)); i= i + 1;
5:          for q ∈ NEIGHBORHOOD(α(i),G) do
6:              if CONNECT (α(i),q) then
7:                  G.add_ edge (α(i),q);
8:              end if
9:          end for
10:  end if
11:end while
```



α(i) is connected to all its neighbors.

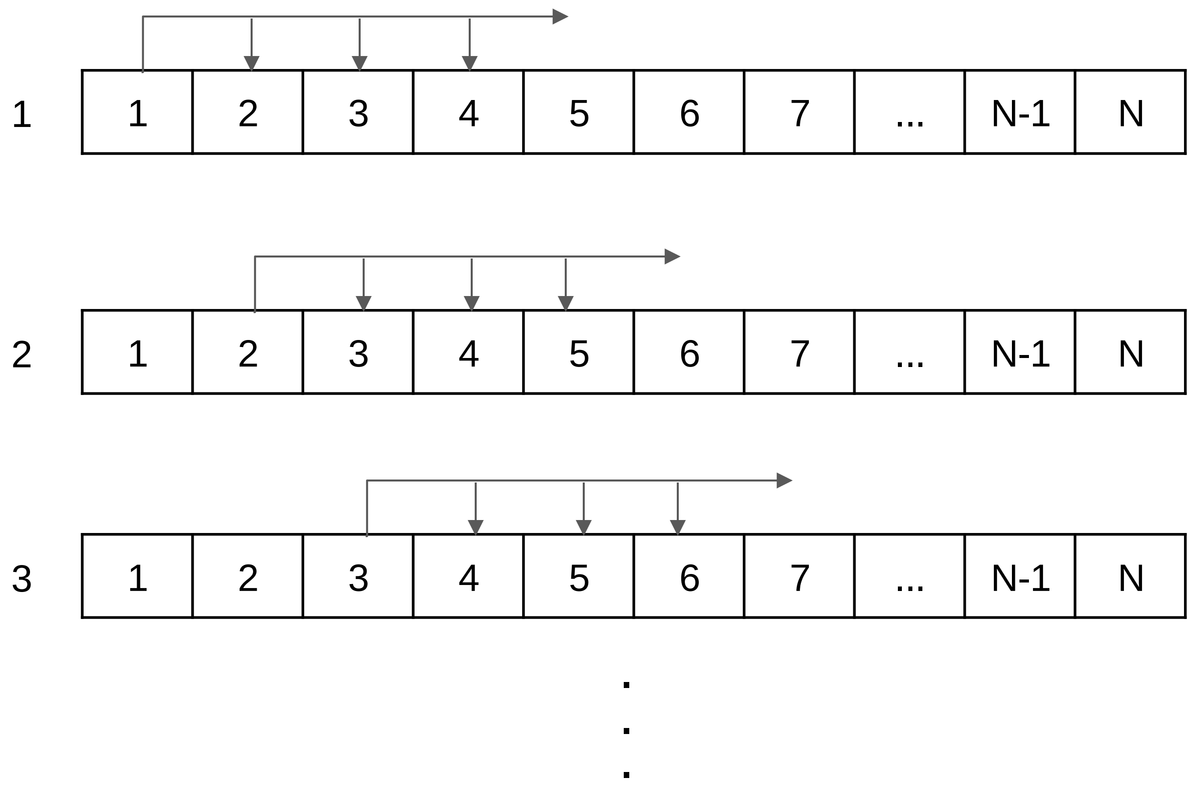
Probabilistic Roadmaps

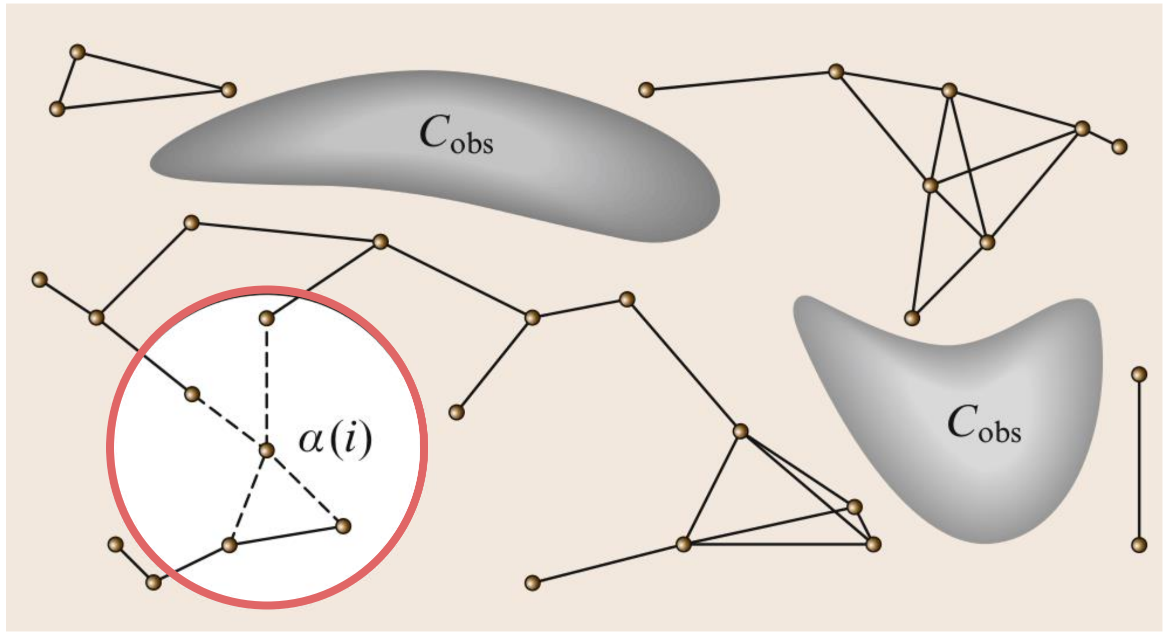
It is probabilistic because the nodes are spread randomly over the **C-space**, and the collision detector removes those in collision with obstacles, leaving only the those that are inside the **C_{free}** space.



Author of picture: Eric O. Scott

PRM: connectivity & dimensionality



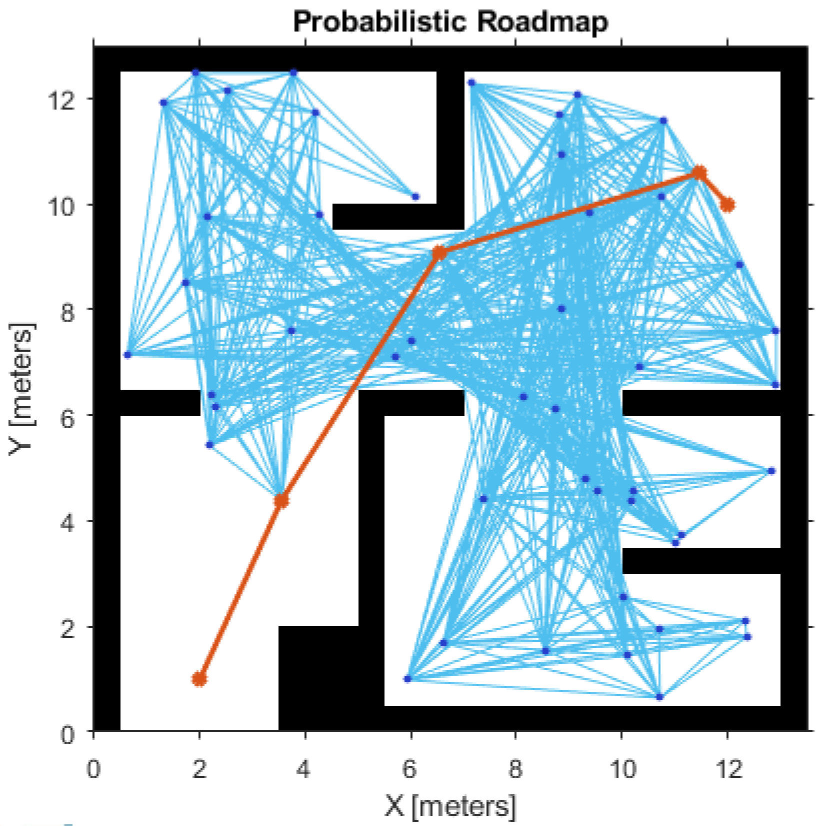


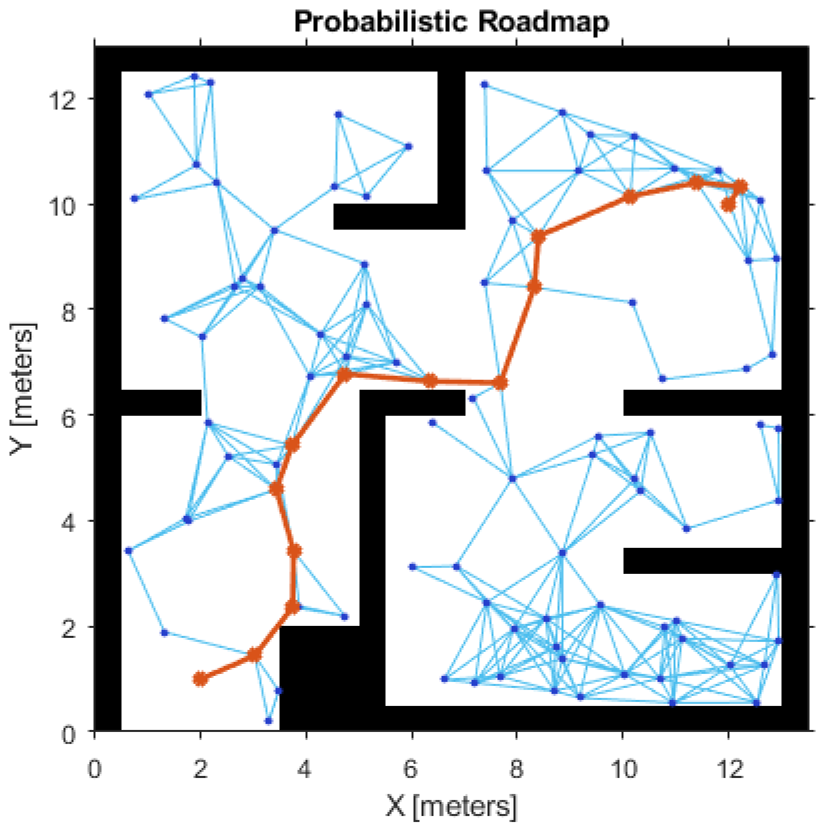
Number of collision checks:

$\sum i = \frac{N \cdot (N+1)}{2}, i = 1, \ldots, N$ The dimensionality increases quadratically.

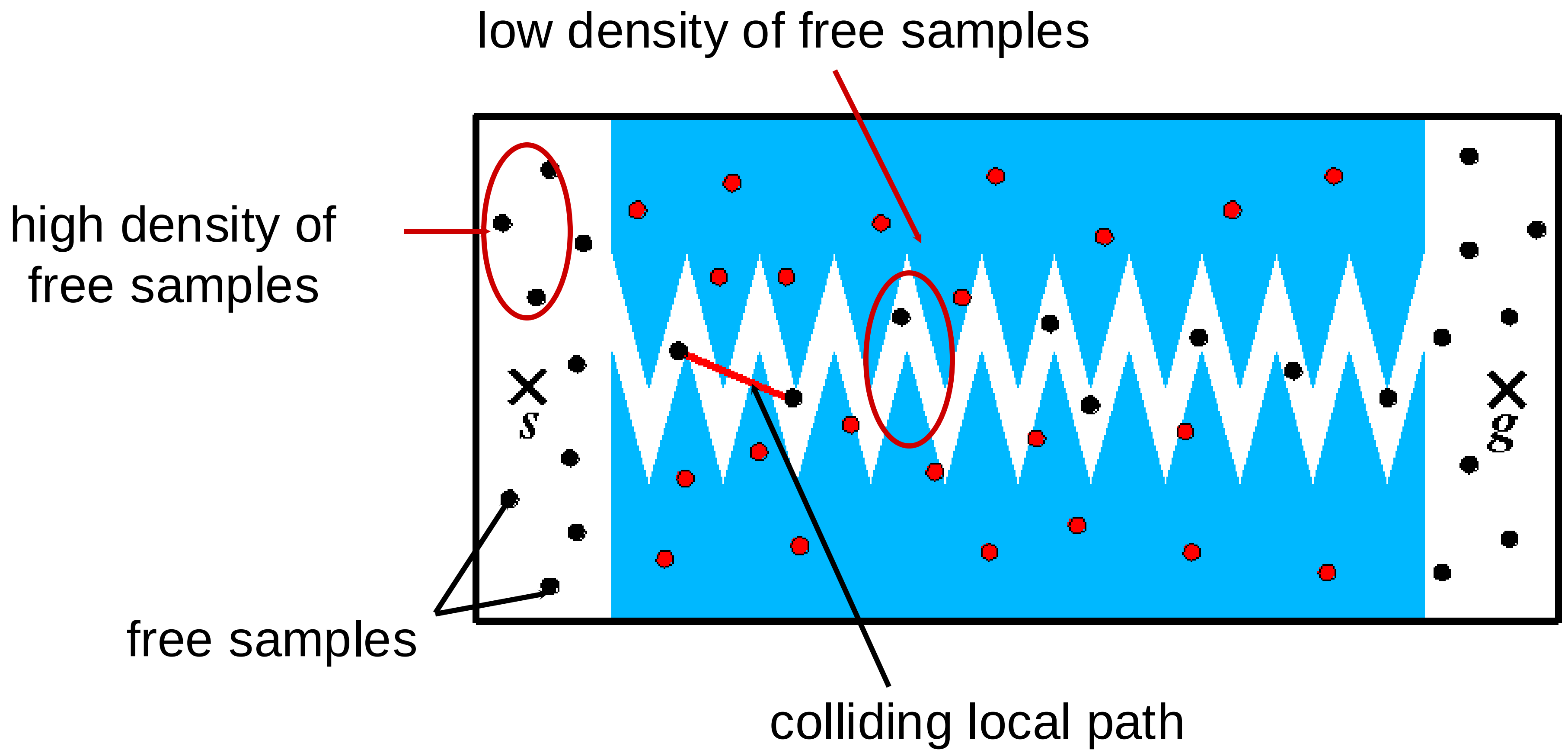
Probabilistic Roadmaps

- The neighborhood distance for connecting each collision-free node is a tunable parameter. A smaller connection distance reduces the number of connections, resulting in a simpler map (see more here)





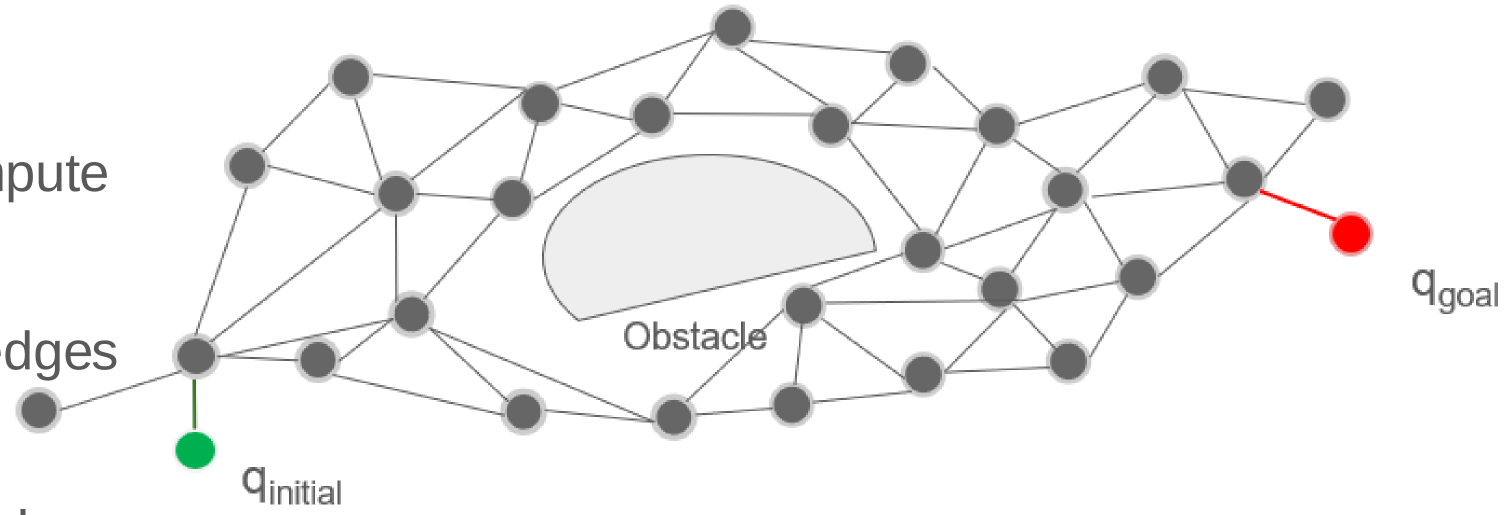
What can Go Wrong? Narrow Passages



It is difficult to capture the free space associated with the narrow passages, because in the narrow passage, volume associated with the free space is relatively low.

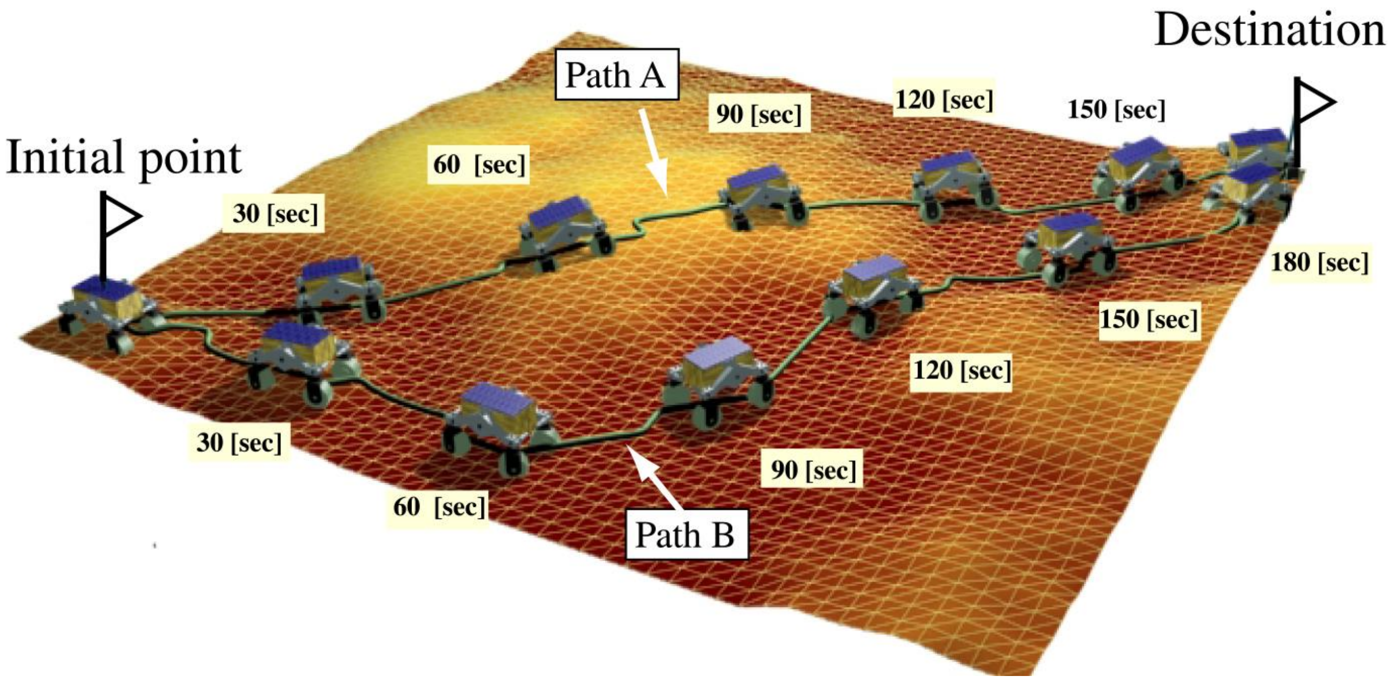
Probabilistic Roadmaps

- Exploit the observation: it is cheap to check if a single robot configuration is in free space or not
- This assumption requires collision detection (feasibility check)
- So, by coarse sampling, we can compute nodes of a roadmap in free space
- Fine sampling (local planner) picks edges connecting nodes
- At ‘query’ time, connect start and goal points to nearest node in roadmap and perform path planning as before



Search Algorithms

Optimality of motion planning

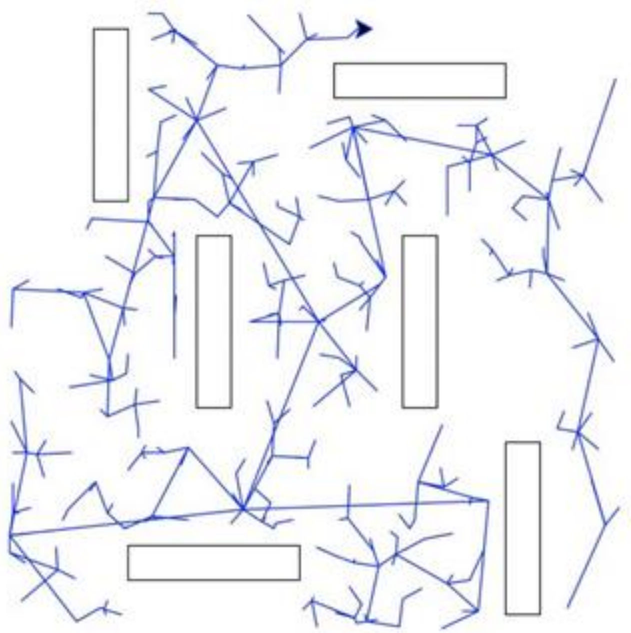


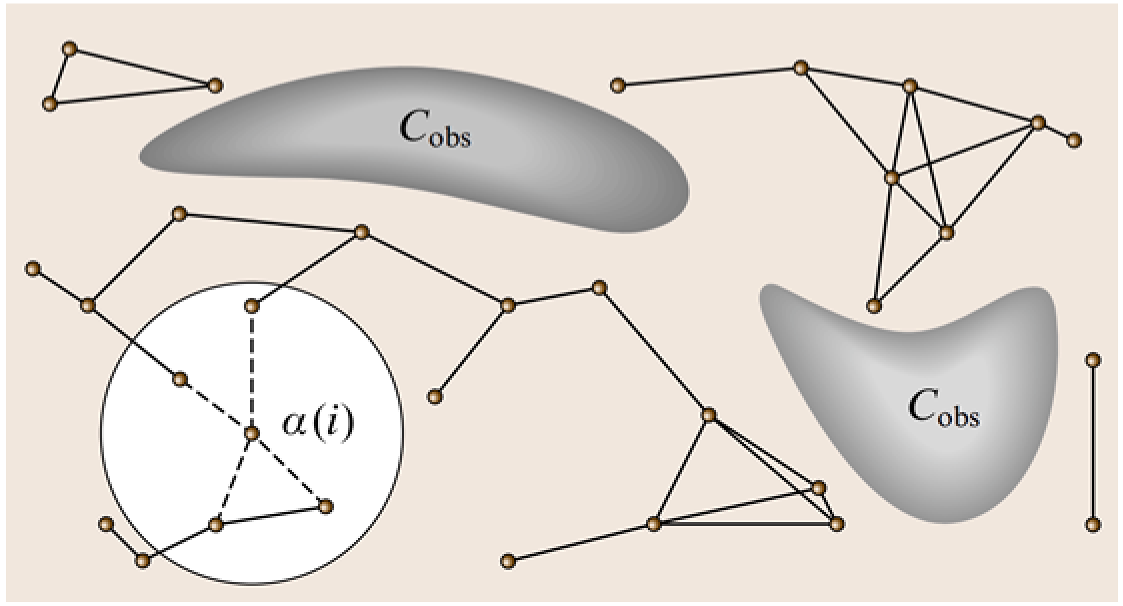
Sub-optimal → Optimal

Discretizations of C-Space

- Sampling-based planning uses collision-detection to probe and incrementally search the Cspace for a solution, eg Probabilistic Roadmaps (PRM) or Rapidly Exploring Random Trees (RRT).

They are probabilistically complete!

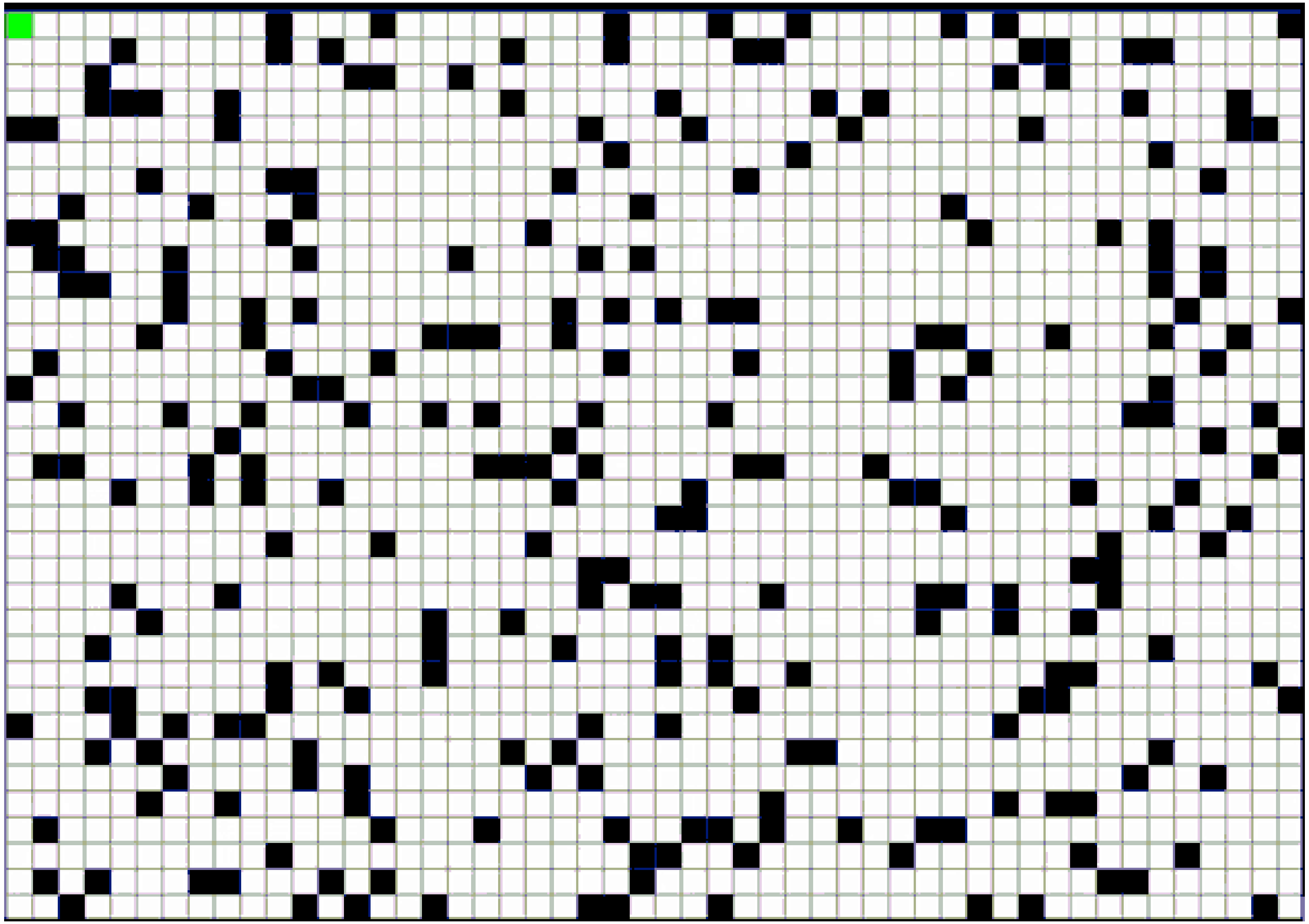




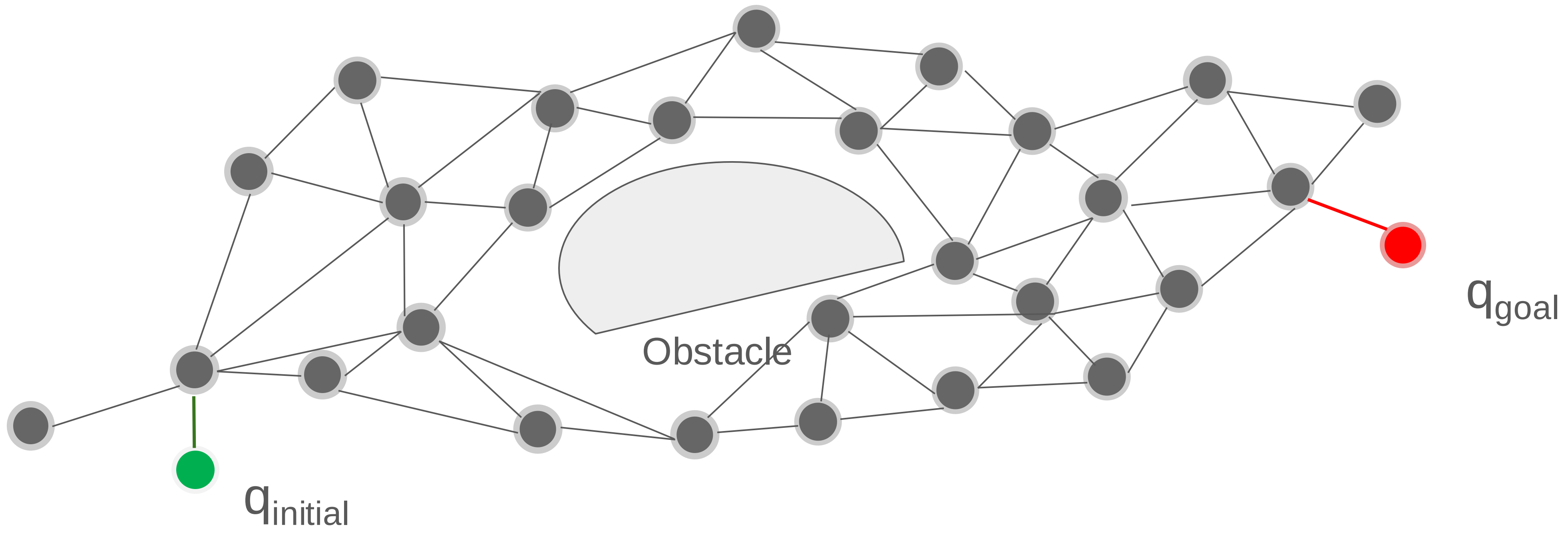
A* search algorithm

- The A* search algorithm: find the shortest path between two points. It begins at a starting point and calculates a ‘**cost**’ for all neighboring points.
- This total cost is a combination of the actual distance from the start point + an estimate of the distance to the goal (known as a ‘heuristic’).
- A* algorithm selects the node with the *lowest cost*. If a cheaper path to a previously seen node is found, its cost is updated. This process continues until the destination is reached.
- A* prioritizes nodes that are likely to lead to the goal, at a lower cost, without visiting all the nodes or searching all the possible paths. That is why by using ‘heuristic’, A* is faster than algorithms that search all the nodes.

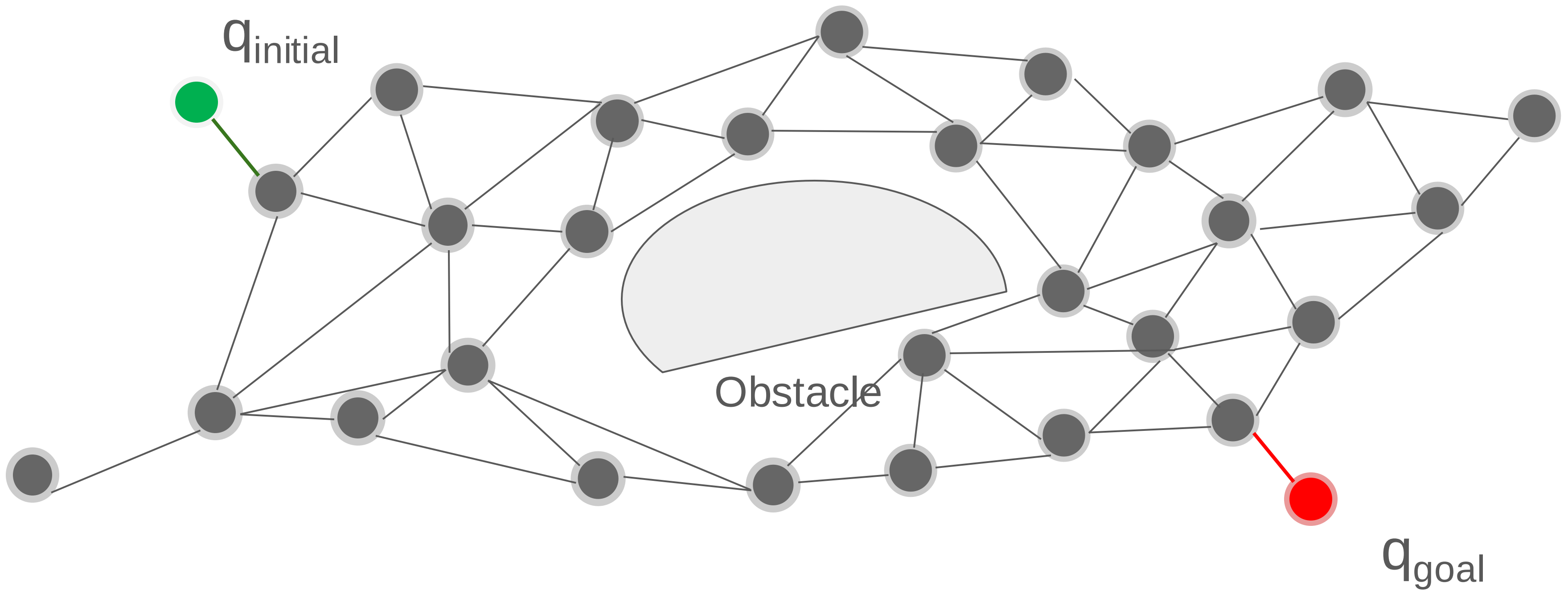
A* search algorithm



A* search algorithm for PRM



A* search algorithm for PRM



This example also shows the nature of multi-query planning, PRM can be used for different pairs of start-goal positions.

Take away messages

- Configuration space or ***C-space,*** C-free motion planning
- Planning in continuous space: Potential Fields (PF)
- Sampling-based planning:
  - ❑ Rapidly exploring Random Tree (RRT)
  - ❑ Probabilistic Roadmap (PRM)
  - ❑ How to use A*