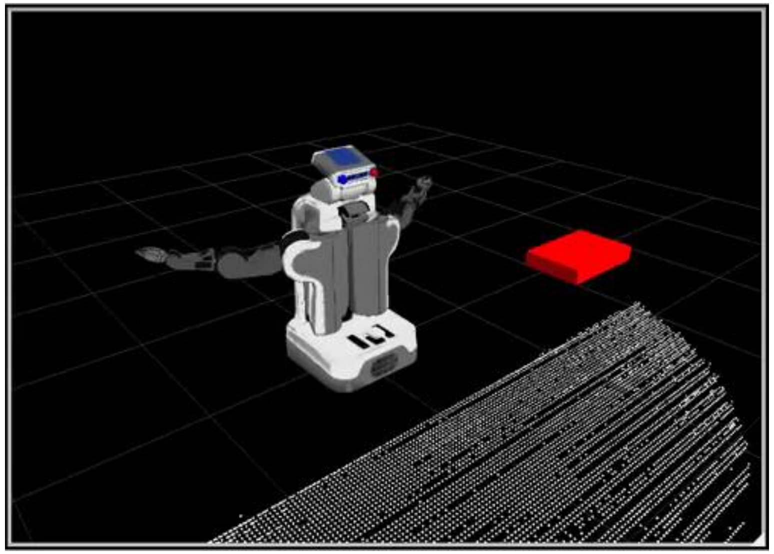
Robot Planning



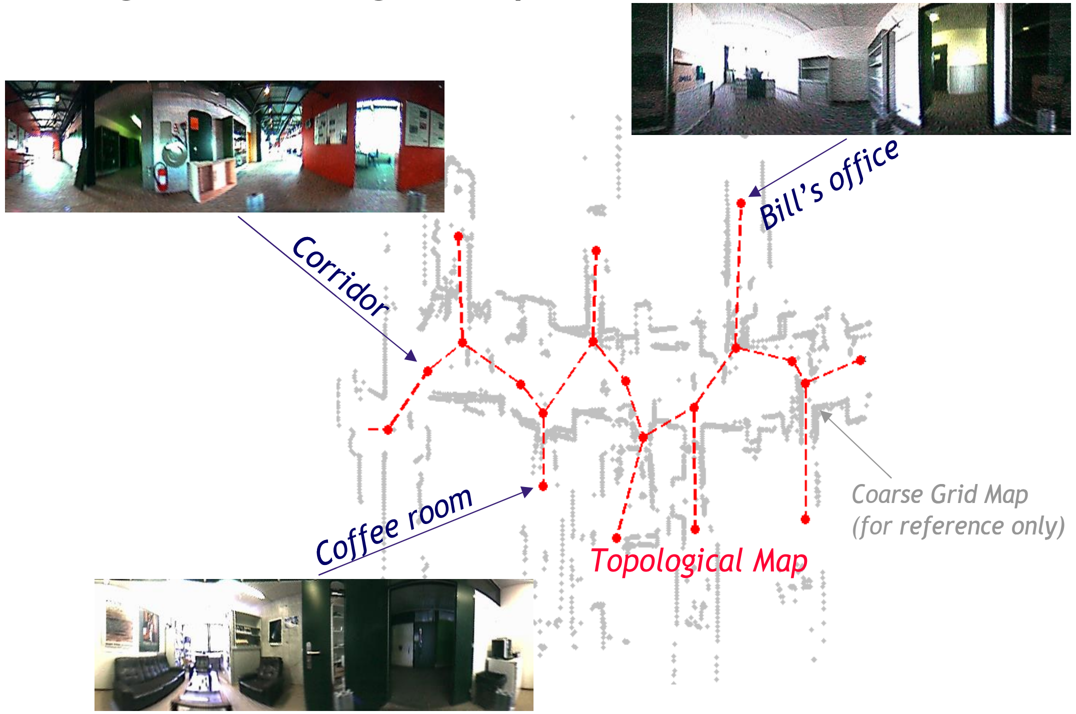
Ok now we’re more realistic…

… however, still autonomous robots are more complex than this

- The robots have to autonomously find their way through..
  - Robotic Manipulators need:
    - To solve the inverse kinematics and dynamics problems
    - Find the correct trajectories to avoid obstacles
    - …
  - Mobile Robots need to use path planning to navigate the environment

The Planning Problem (case of Mobile Robots 1/2)

- The problem: find a path in the work space (physical space) from the initial position to the goal position avoiding all collisions with the obstacles
- Assumption: there exists a good enough map of the environment for navigation.
  - Topological
  - Metric
  - Hybrid methods



The Planning Problem (case of Mobile Robots 2/2)

- We can generally distinguish between
  - (global) path planning and
  - (local) obstacle avoidance.

- First step:
  - Transformation of the map into a representation useful for planning
  - This step is planner-dependent

- Second step:
  - Plan a path on the transformed map

- Third step:
  - Send motion commands to controller
  - This step is planner-dependent (e.g. Model based feed forward, path following

Planning Algorithms: Potential Fields

Goal: Attractive Force

Large Distance --> Large Force

Model as Spring

Hooke's Law

$F = -k X$

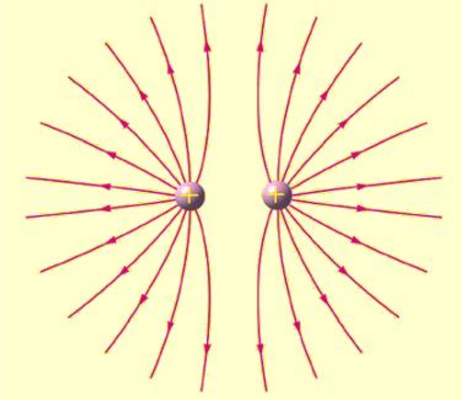
Obstacles: Repulsive Force

Small Distance --> Large Force

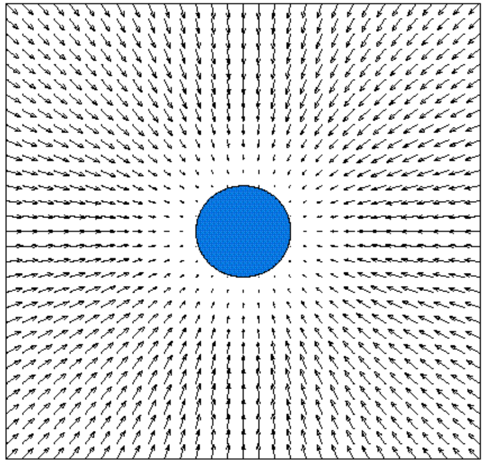
Model as Electrically Charged Particles

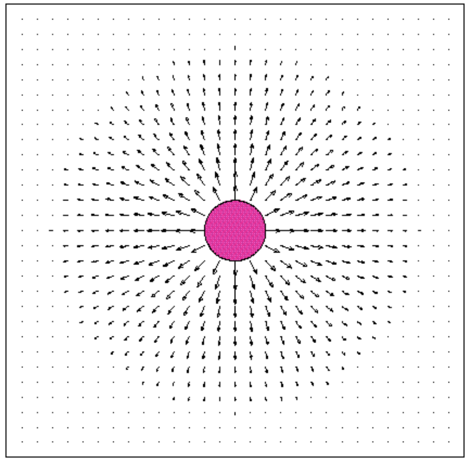
Coulomb's law

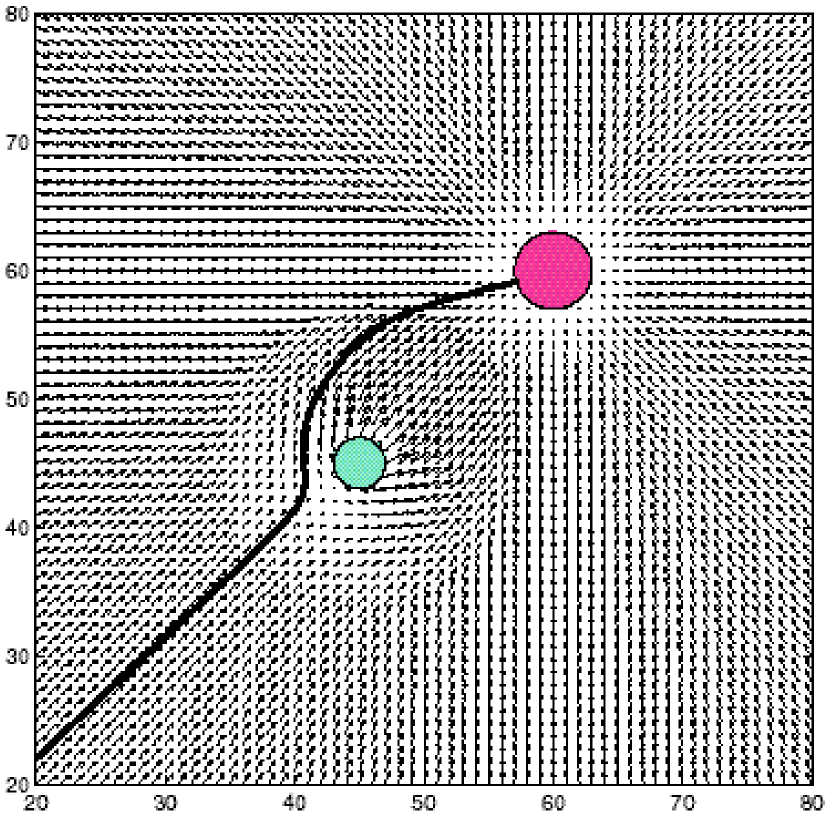
$F = k\, q_1 q_2 / r^2$



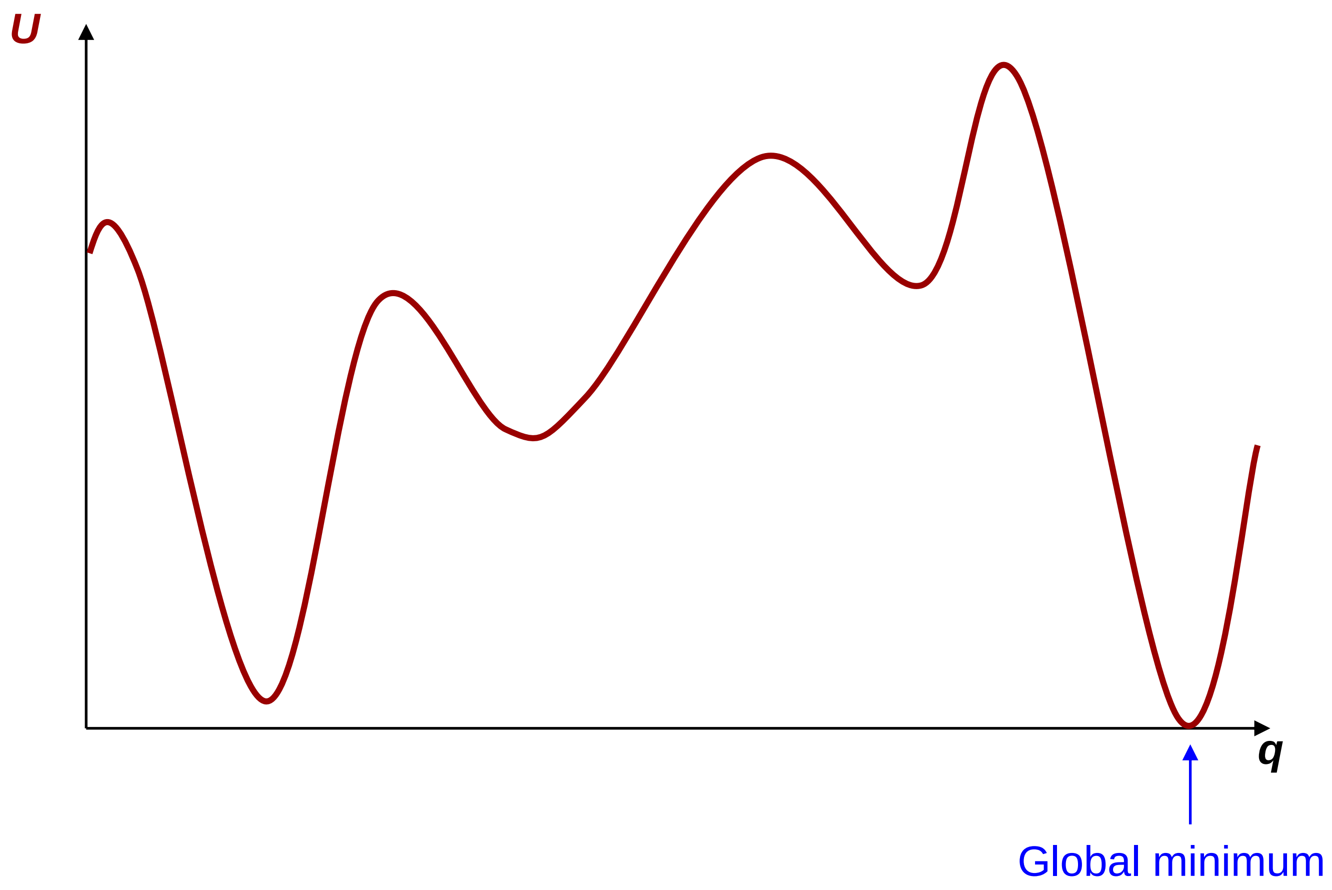
Planning Algorithms: Potential Fields







Bad Potential Field



Sampling-based Path Planning (or Randomized graph search)

- When the state space is large complete solutions are often infeasible.
- In practice, most algorithms are only resolution complete, i.e., only complete if the resolution is ne-grained enough
- Sampling-based planners create possible paths by randomly adding points to a tree until some solution is found

RRT

- RRT is a good example of a Sampling-based algorithm:

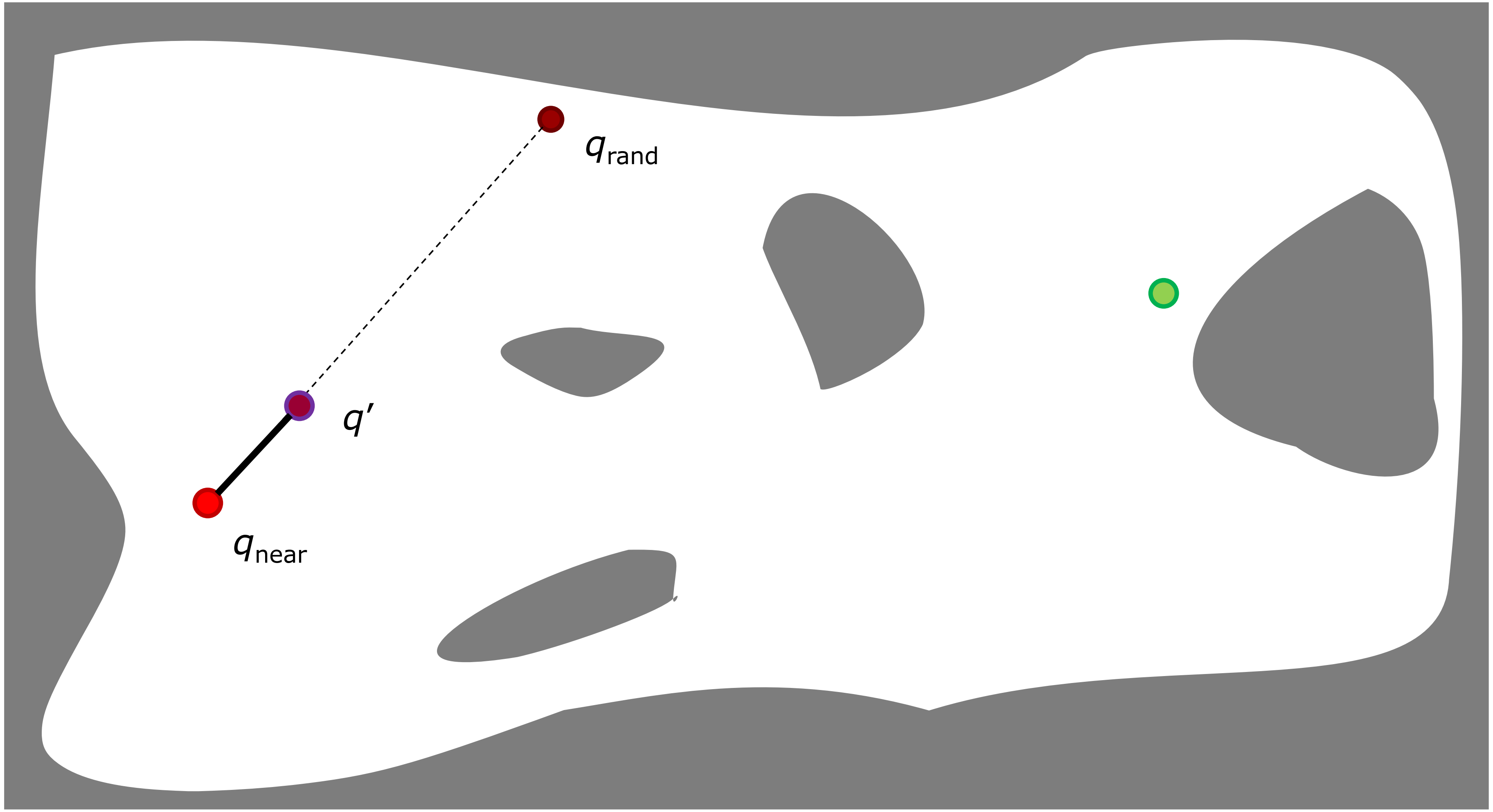
$\mathrm{RRT}(q_0)$

```
G.init(q_0);
for i = 1 to k do
    q_n ← NEAREST(S, α(i));
    q_s ← STOPPING-CONFIGURATION(q_n, α(i));
    if q_s ≠ q_n then
        G.add_vertex(q_s);
        G.add_edge(q_n, q_s);
```

- Several additional algorithms are worth exploring
  - RRT*
  - Informed RRT
  - …

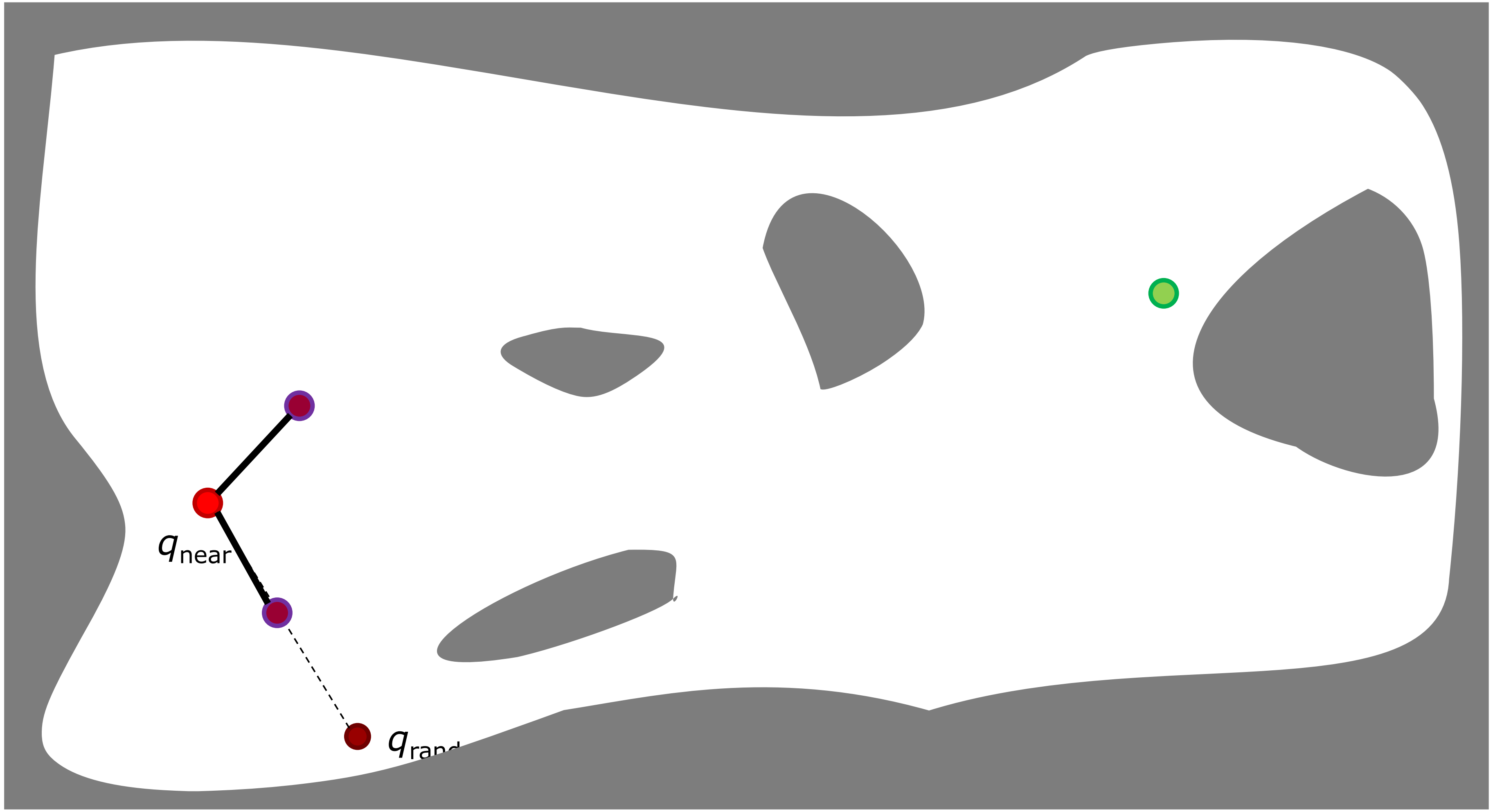
RRT

- Rapidly-exploring random trees



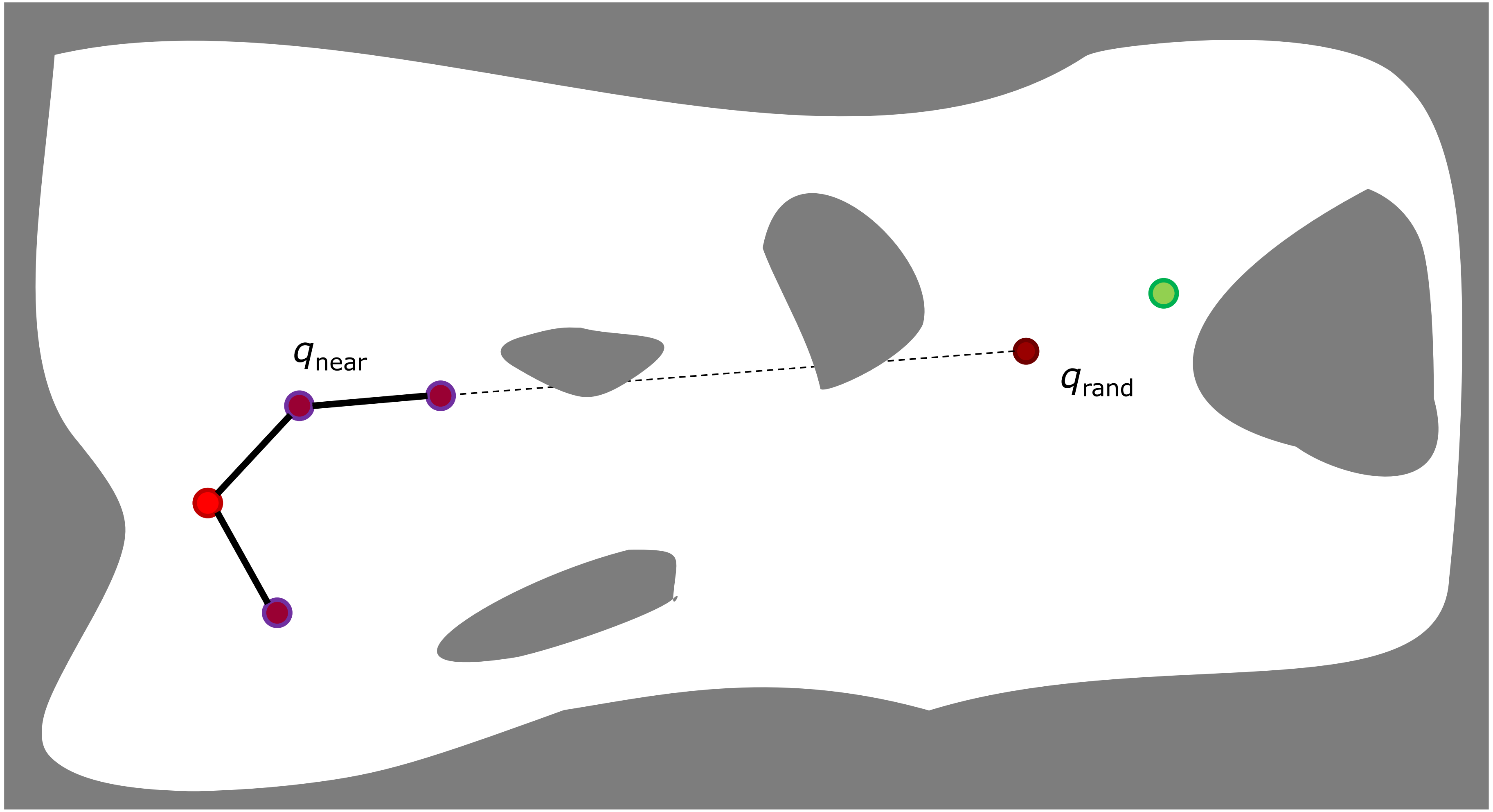
RRT

- Rapidly-exploring random trees



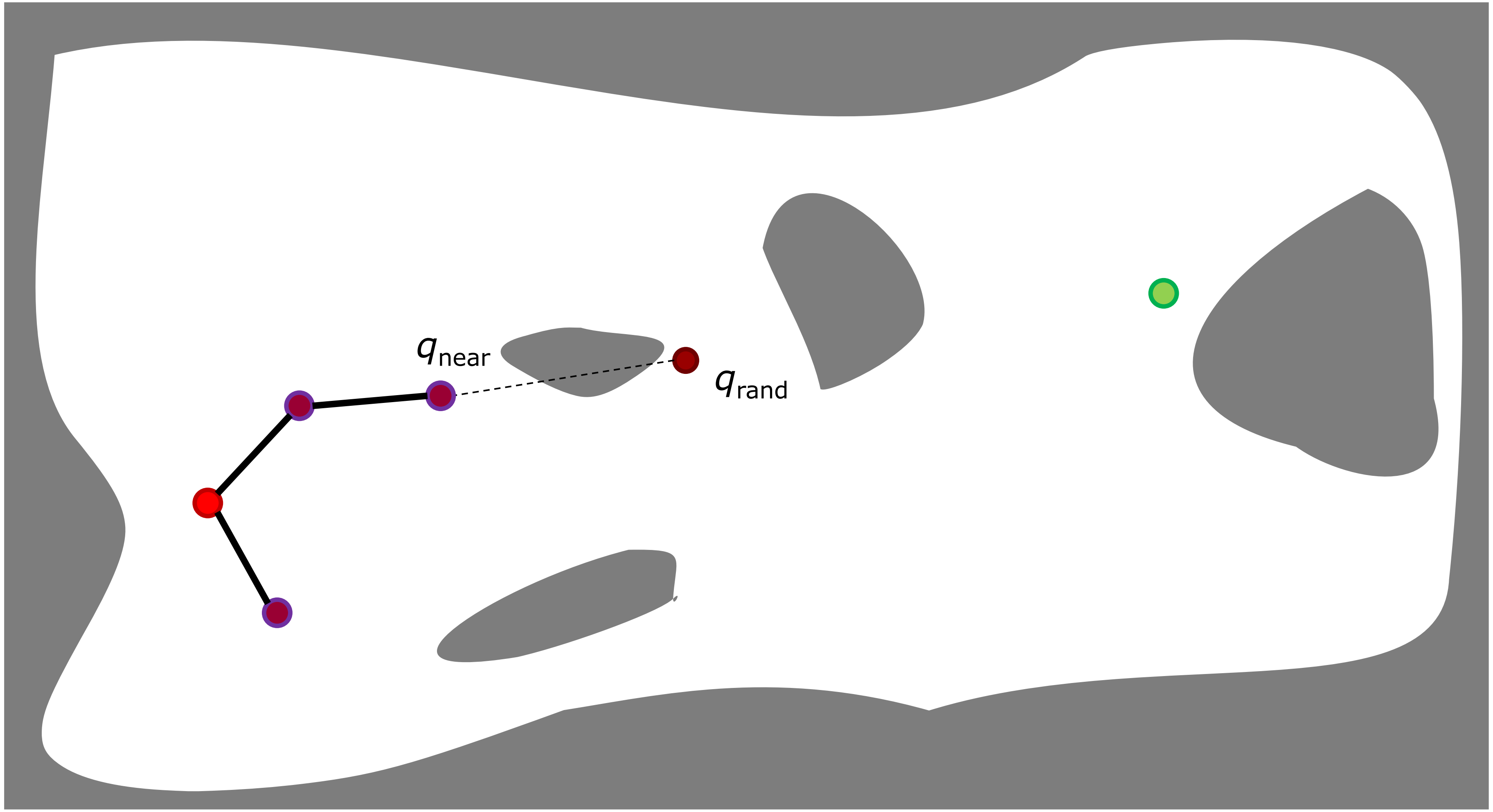
RRT

- Rapidly-exploring random trees



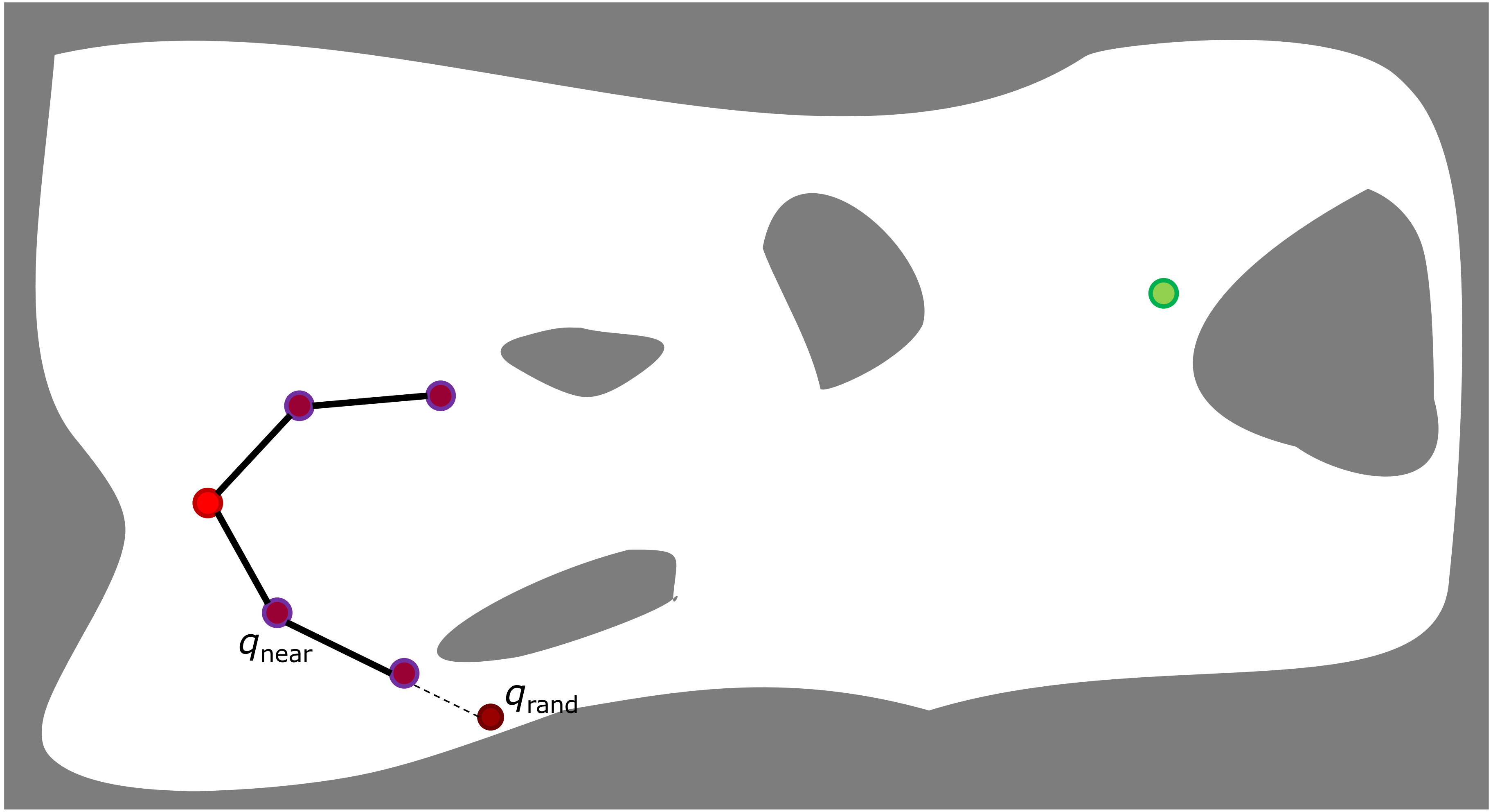
RRT

- Rapidly-exploring random trees



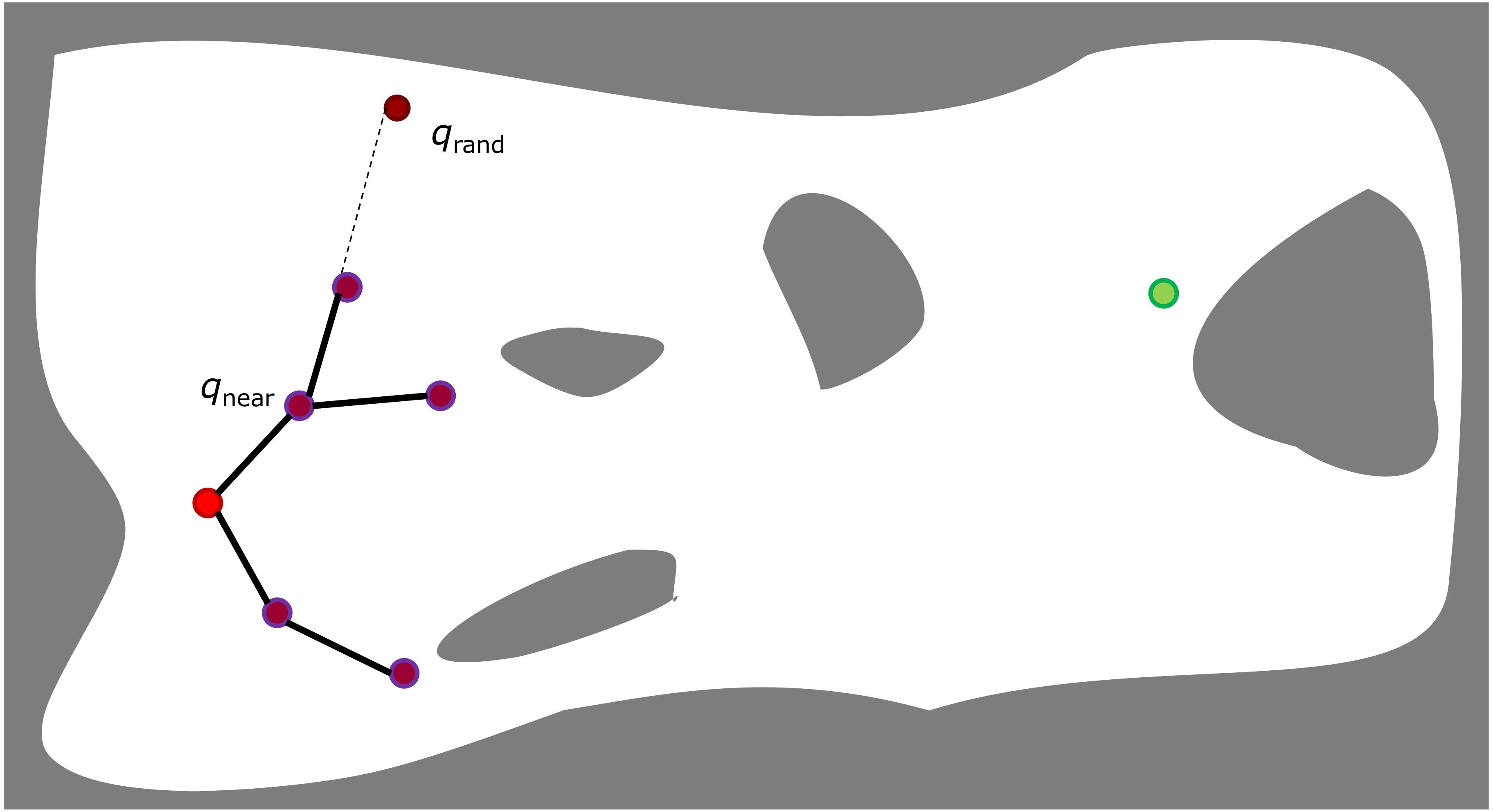
RRT:

- Rapidly-exploring random trees



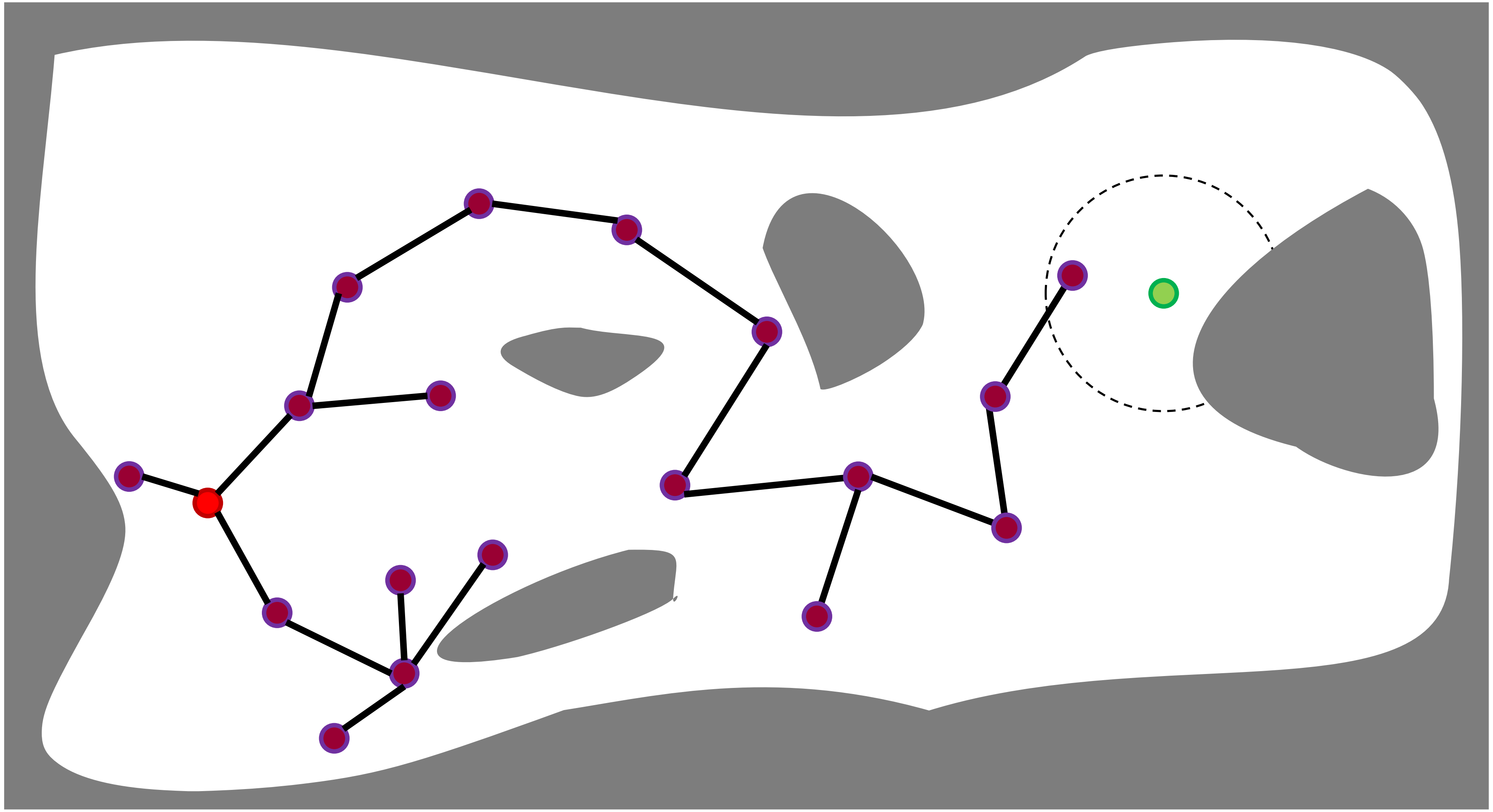
RRT

- Rapidly-exploring random trees



RRT

- Rapidly-exploring random trees



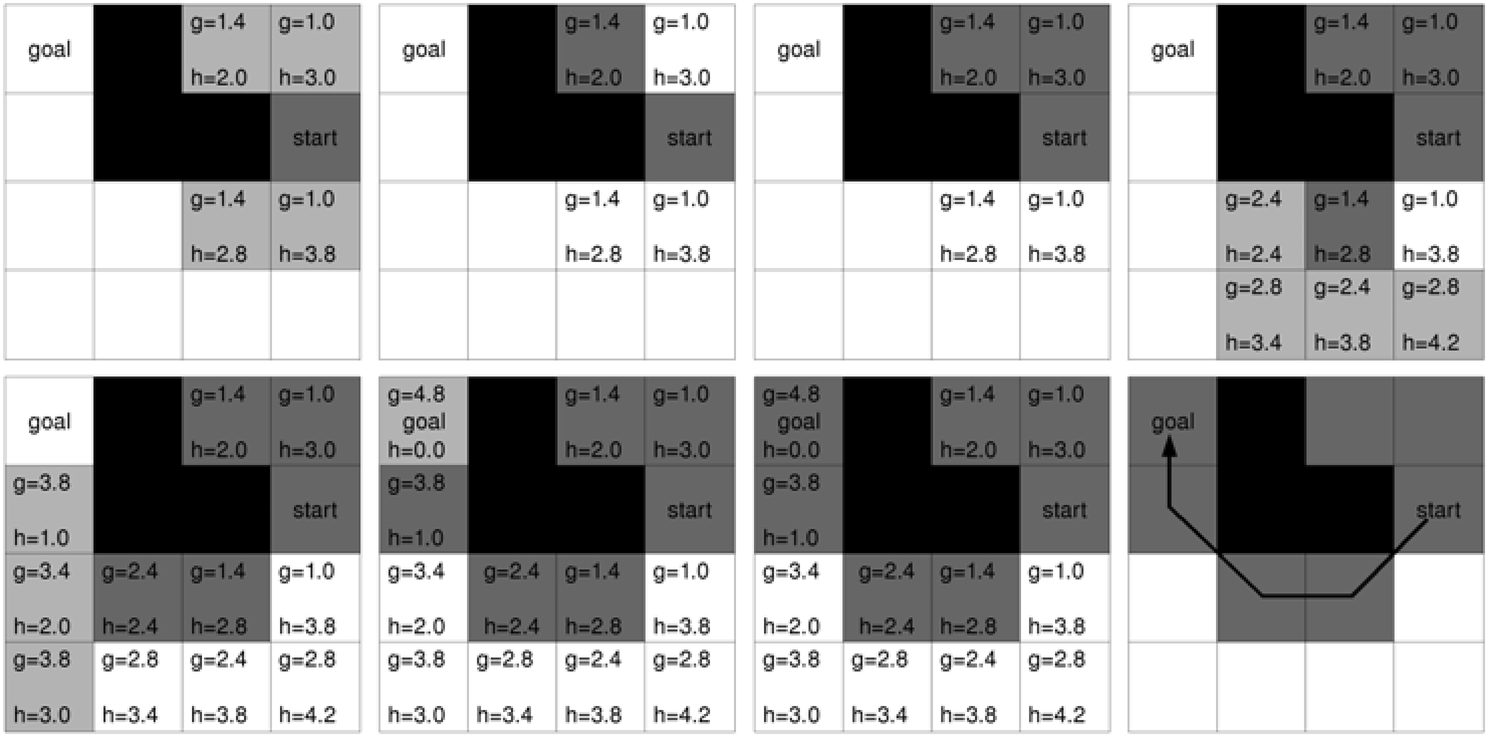
Forward Search Agorithms

- Forward Search Methods:
  - Breadth first
  - Dijkstra’s algorithm
  - A∗

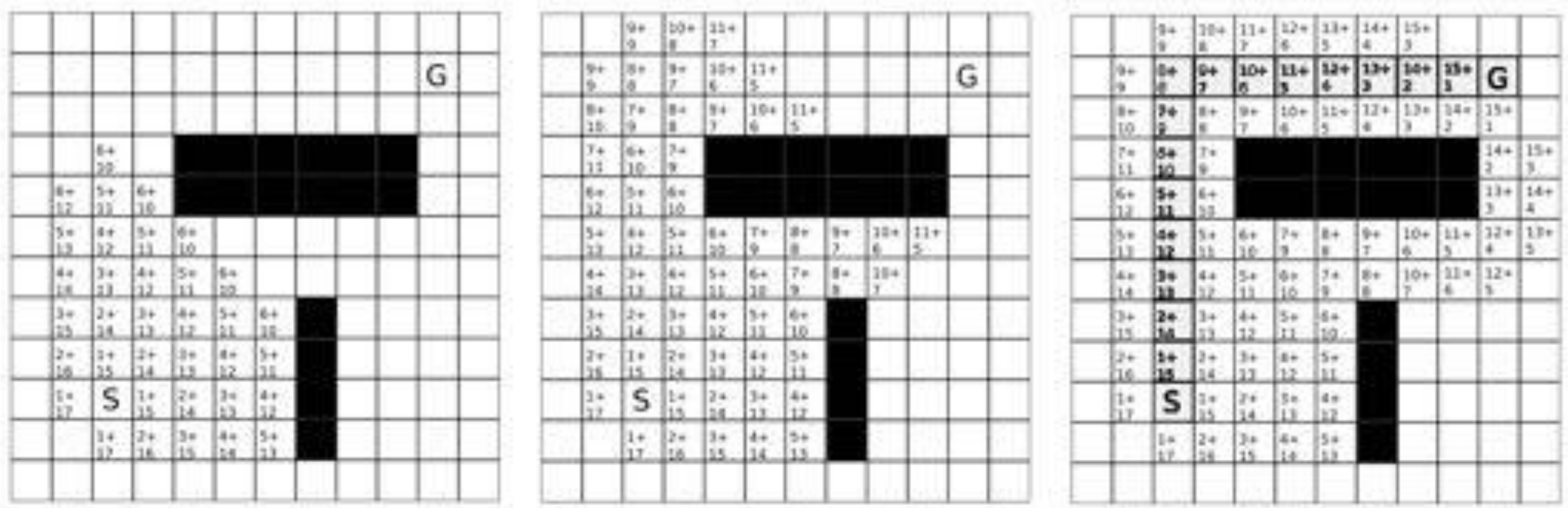
```
FORWARD_SEARCH
Q.Insert(x_I) and mark x_I as visited
while Q not empty do
    x ← Q.GetFirst()
    if x ∈ X_G
        return SUCCESS
    forall u ∈ U(x)
        x′ ← f(x, u)
        if x′ not visited
            Mark x′ as visited
            Q.Insert(x′)
        else
            Resolve duplicate x′
return FAILURE
```

Planning Algorithms: A* Search

- Similar to Dijkstra's algorithm, except that it uses a heuristic function h(n)
- $f(n) = g(n) + \varepsilon\, h(n)$



Planning Algorithms: A*



Software for Autonomous Systems

SFfAS-31391:

Learning ROS Transforms (TF), Robot Visualization (RVIZ) and Simulation (Gazebo)

Lecturer, Course Coordinator: Evangelos Boukas—PhD