L7-1: Sampled-based Motion Planning

Hao Su

Spring, 2021

Agenda

- Problem Formulation
- Probabilistic Roadmap Method (PRM)
- Rapidly-exploring Random Trees (RRT)

click to jump to the section.

Problem Formulation

Configuration Space

- Configuration space ($\mathcal{C}$-space) is a subset of $\mathbb{R}^n$ containing all possible states of the system(state space in RL).
- $\mathcal{C}_{free} \subseteq \mathcal{C}$ contains all valid states.
- $\mathcal{C}_{obs} \subseteq \mathcal{C}$ represents obstacles.
- Examples:
  - All valid poses of a robot.
  - All valid joint values of a robot.
  - ...

Motion Planning

- Problem:
  - Given a configuration space $\mathcal{C}_{free}$
  - Given start state q_{start} and goal state q_{goal} in $\mathcal{C}_{free}$
  - Calculate a sequence of actions that leads from start to goal
- Challenge:
  - Need to avoid obstacles
  - Long planning horizon
  - High-dimensional planning space

Motion Planning

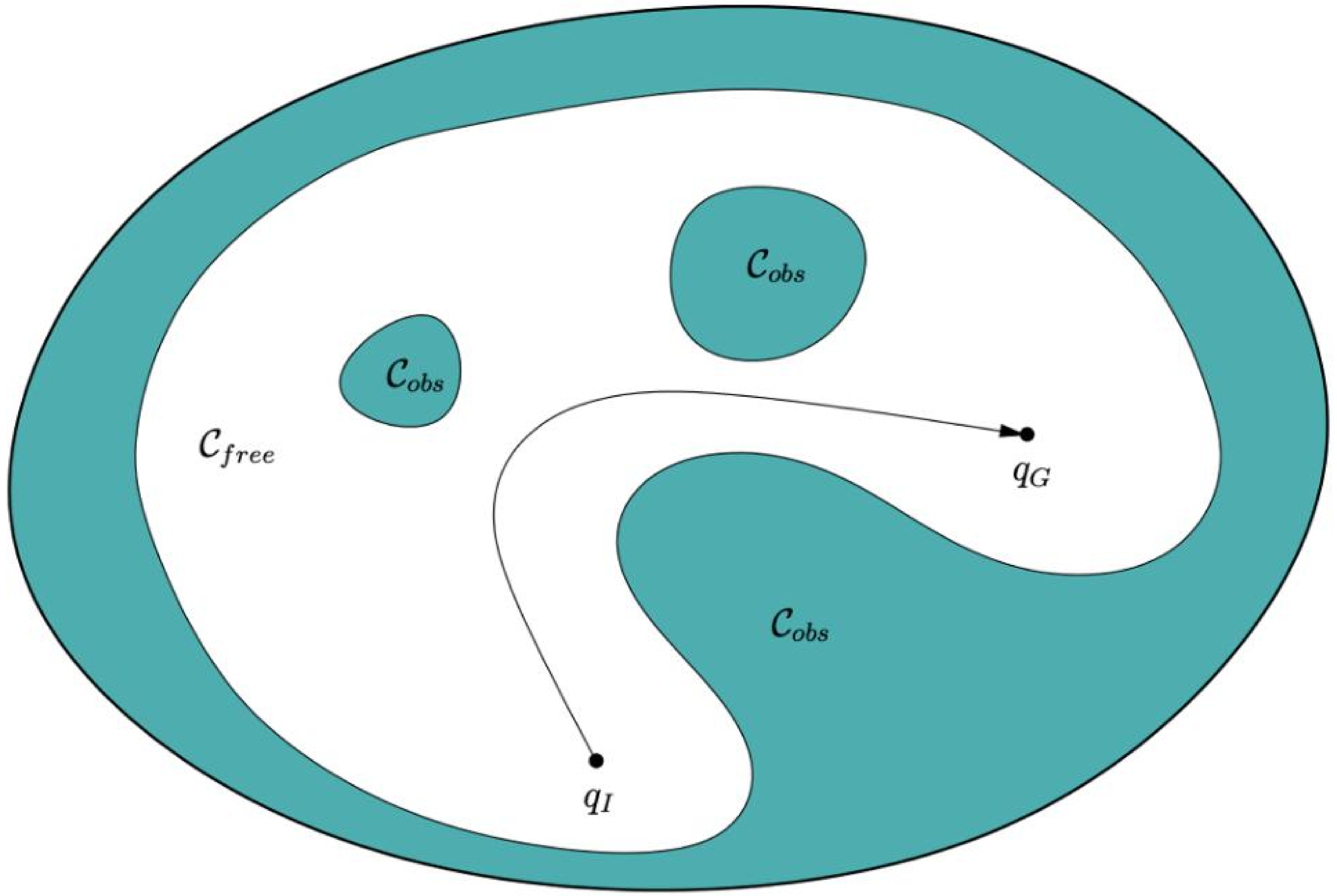
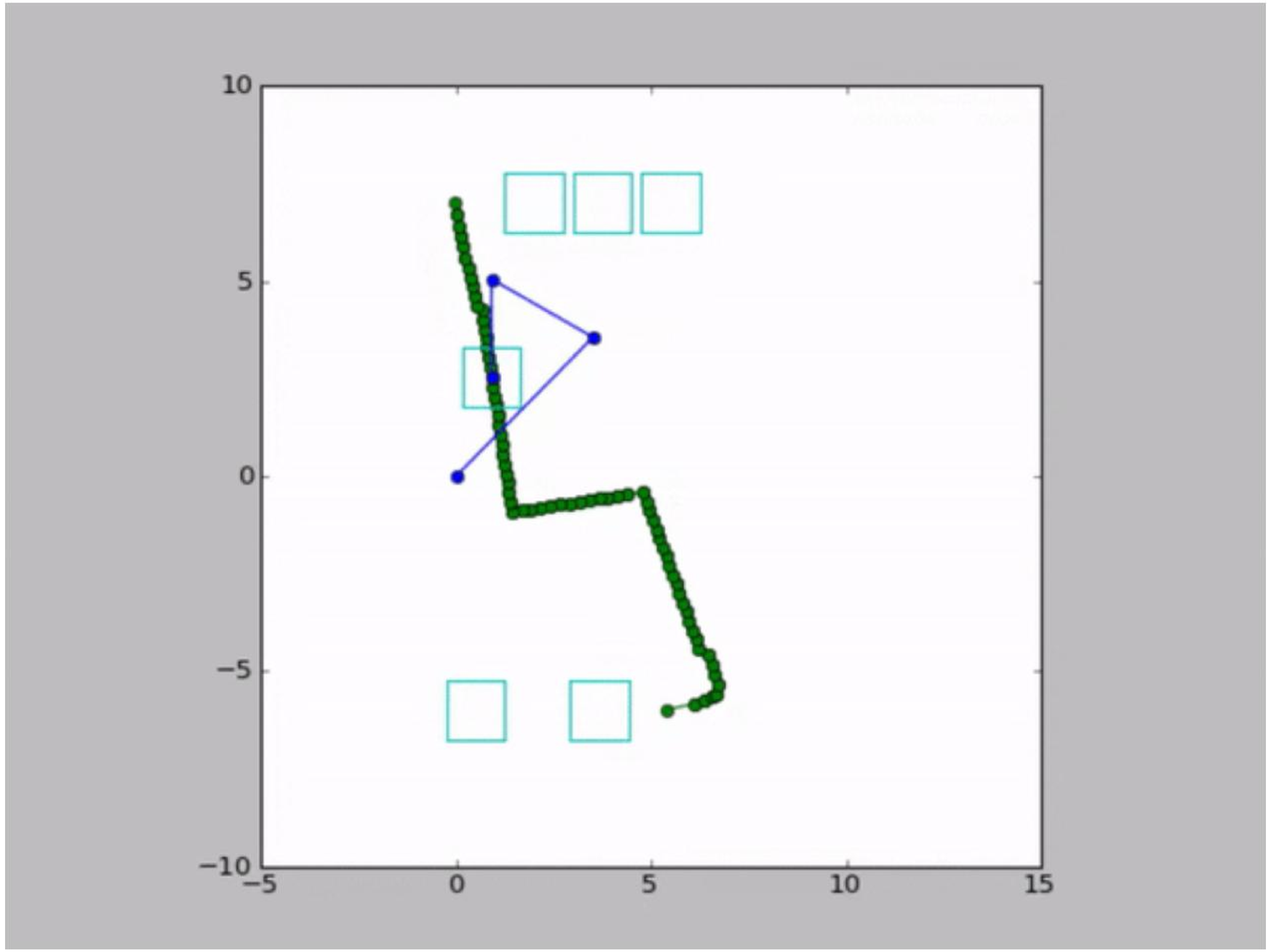


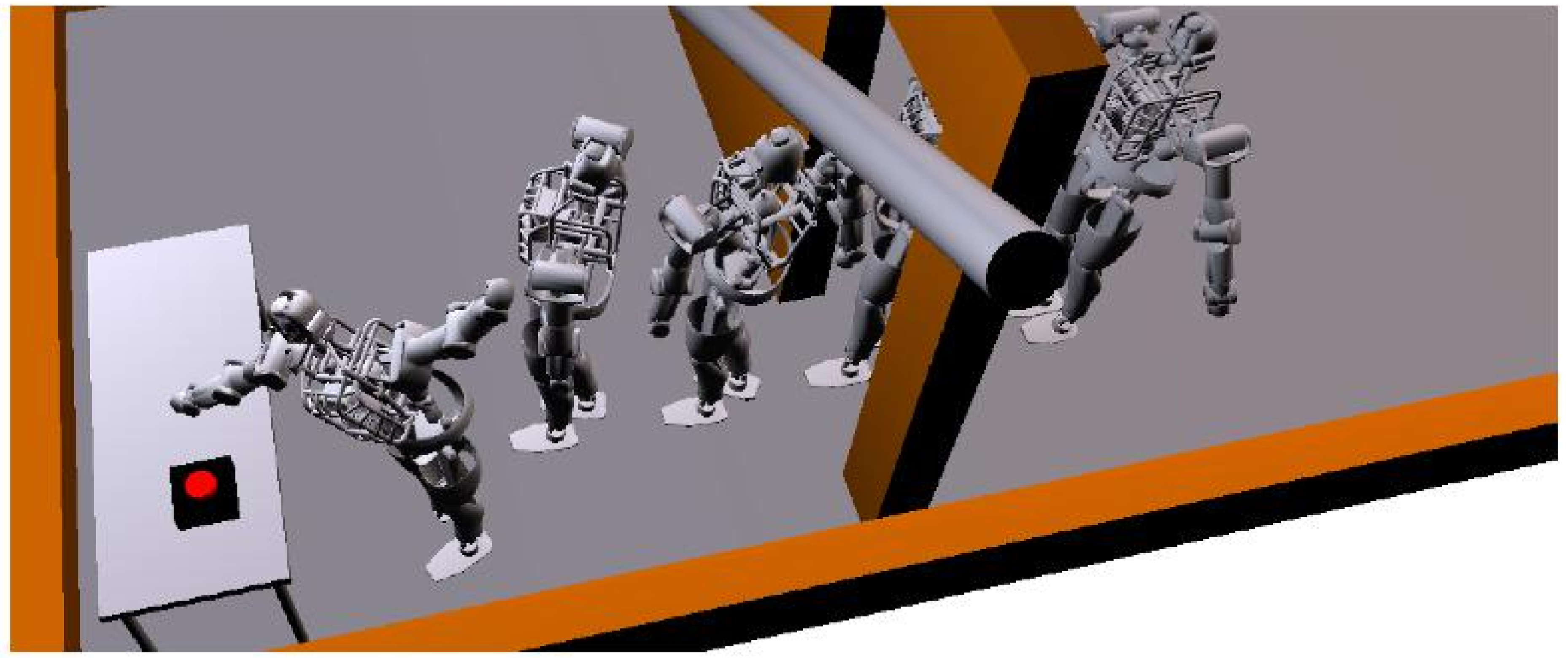
Figure 4.11: The basic motion planning problem is conceptually very simple using C-space ideas. The task is to find a path from q_I to q_G in $\mathcal{C}_{free}$. The entire blob represents $\mathcal{C} = \mathcal{C}_{free} \cup \mathcal{C}_{obs}$.

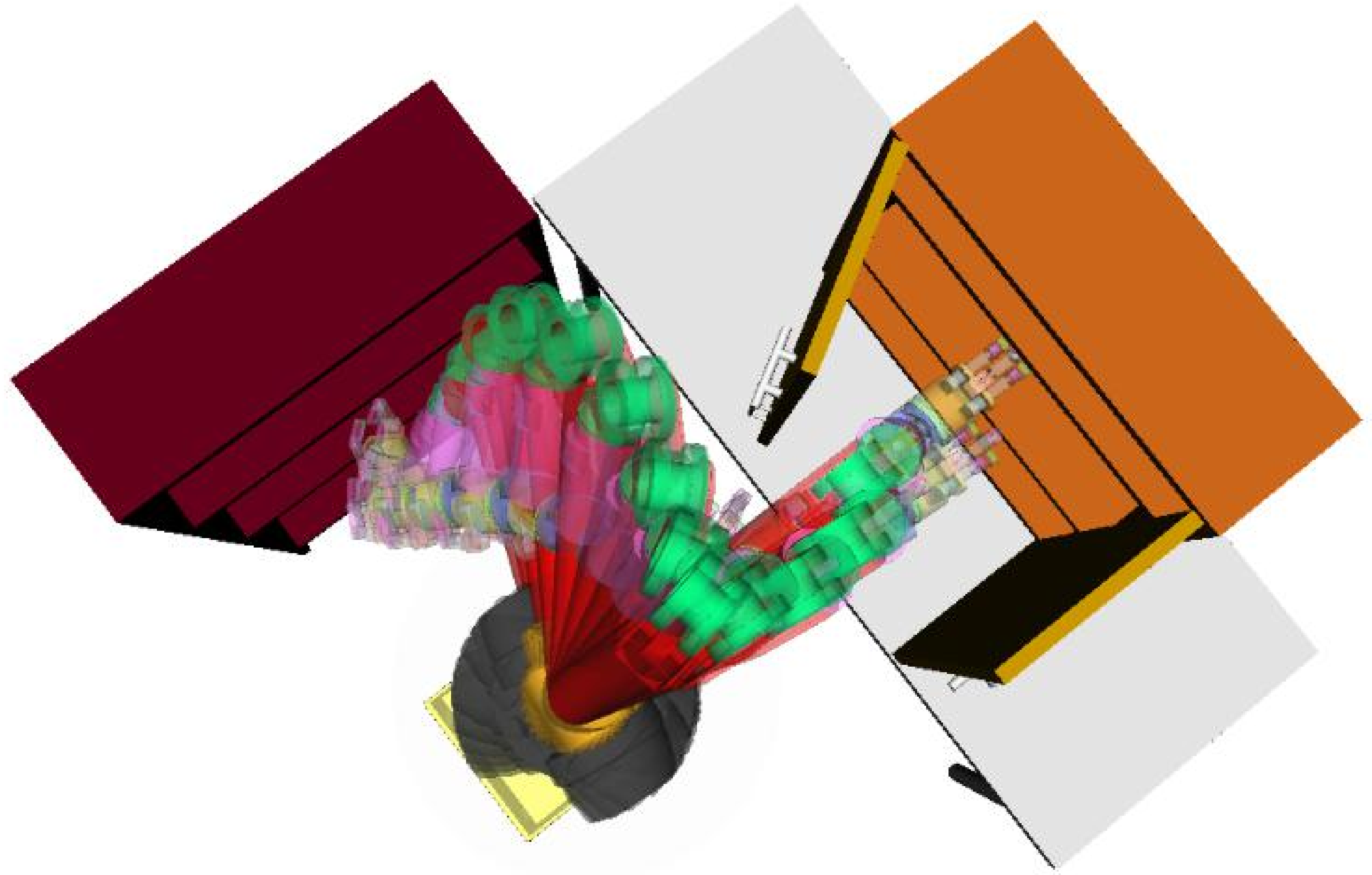
LaValle, Steven M. Planning algorithms. Cambridge university press, 2006.

Examples



Examples





- *Ratliff N, Zucker M, Bagnell J A, et al. CHOMP: Gradient optimization techniques for efficient motion planning, ICRA 2009*
- *Schulman, John, et al. Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization, RSS 2013*

Sample-based Algorithm

- The key idea is to explore a smaller subset of possibilities randomly without exhaustively exploring all possibilities.
- Pros:
  - Probabilistically complete
  - Solve the problem after knowing partial of $\mathcal{C}_{free}$
  - Apply easily to high-dimensional $\mathcal{C}$-space
- Cons:
  - Requires to find path between two close points
  - Does not work well when the connection of $\mathcal{C}_{free}$ is bad
  - Never optimal

Probabilistic Roadmap Method (PRM)

Probabilistic Roadmap(PRM)

- The algorithm contains two stages:
  - Map construction phase
    - Randomly sample states in C_{free}
    - Connect every sampled state to its neighbors
    - Connect the start and goal state to the graph
- Query phase
  - Run path finding algorithms like Dijkstra

Kavraki, Lydia E., et al. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces." IEEE transactions on Robotics and Automation 12.4 (1996): 566-580.

Rejection Sampling

- Aim to sample uniformly in $\mathcal{C}_{free}$.
- Method
  - Sample uniformly over $\mathcal{C}$.
  - Reject the sample not in the feasible area.

Pipeline

Input: n: number of sampled nodes in the roadmap, k: number of closest neighbours to examine for each configuration, q_{start}, q_{goal}.

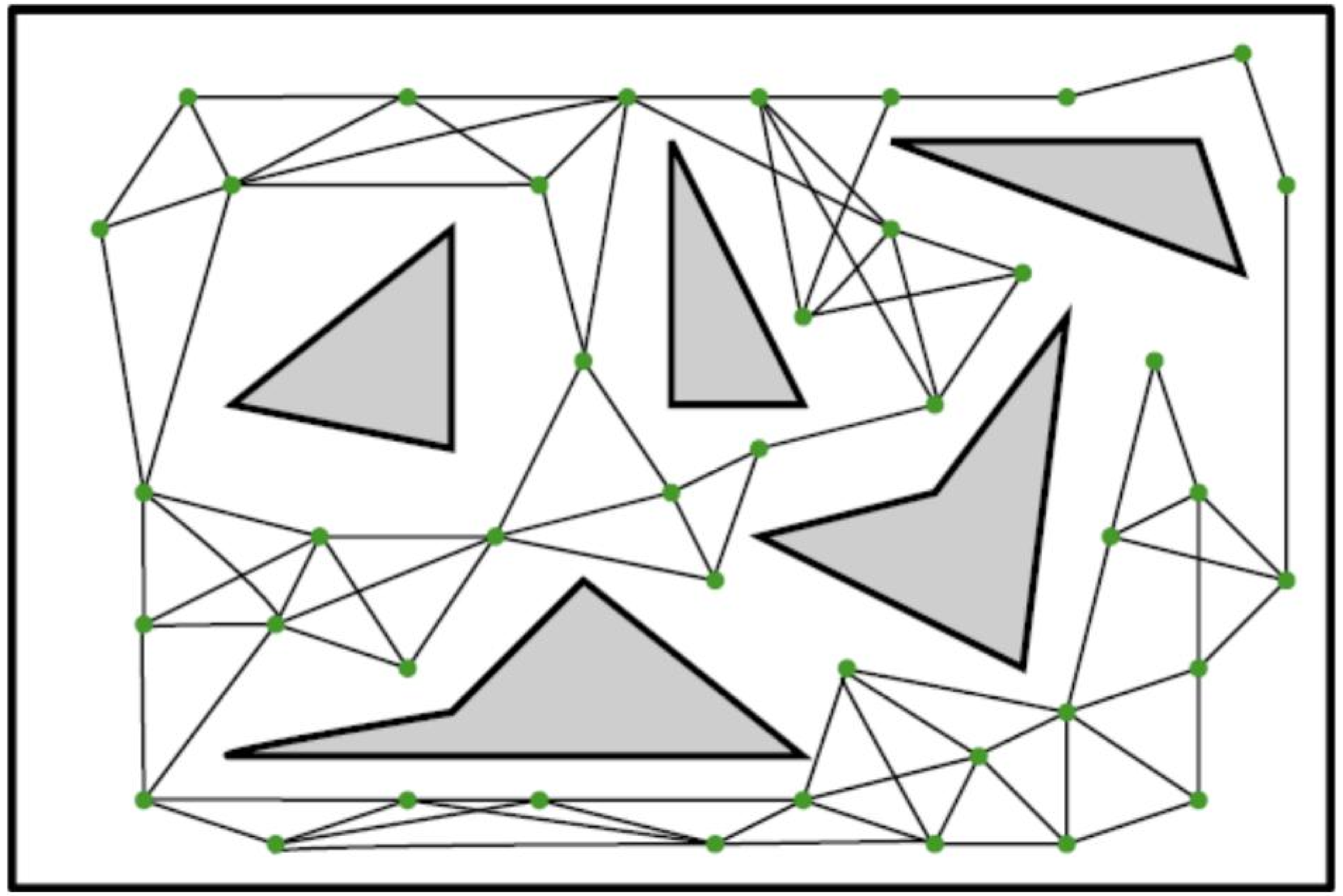
$V \leftarrow \{q_{start}, q_{goal}\}$;
$E \leftarrow \emptyset$;
while $|V| < n$ **do**
    repeat
        $q \leftarrow$ a random configuration in C.
    until q *is in* C_{free};
end
foreach $q \in V$ **do**
    $N_q \leftarrow$ the k closest neighbours of q chosen from V according to a distance function;
    foreach $q' \in N_q$ **do**
        if $(q, q') \notin E$ *and* $(q, q') \in C_{free}$ **then**
            $E \leftarrow E \cup \{(q, q')\}$
        end
    end
end
Find a path from q_{start} to q_{goal} with Dijkstra algorithm;

Challenges

- Connect neighboring points:
  - In general it requires solving dynamics
- Collision checking:
  - It takes a lot of time to check if the edges are in the configuration space.

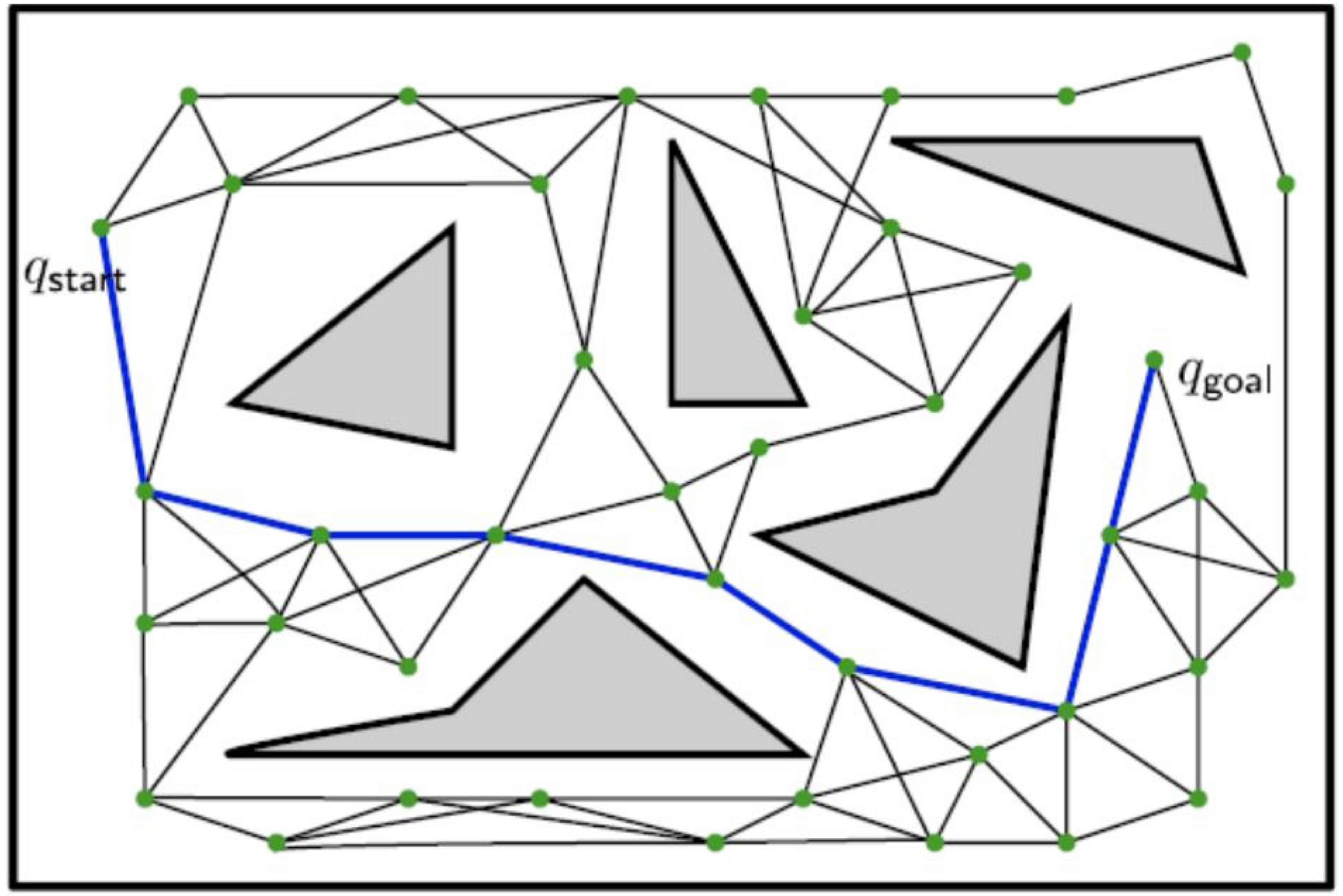
Example

PRM generates a graph $G = (V, E)$ such that every edge is in the configuration space without colliding with obstacles.



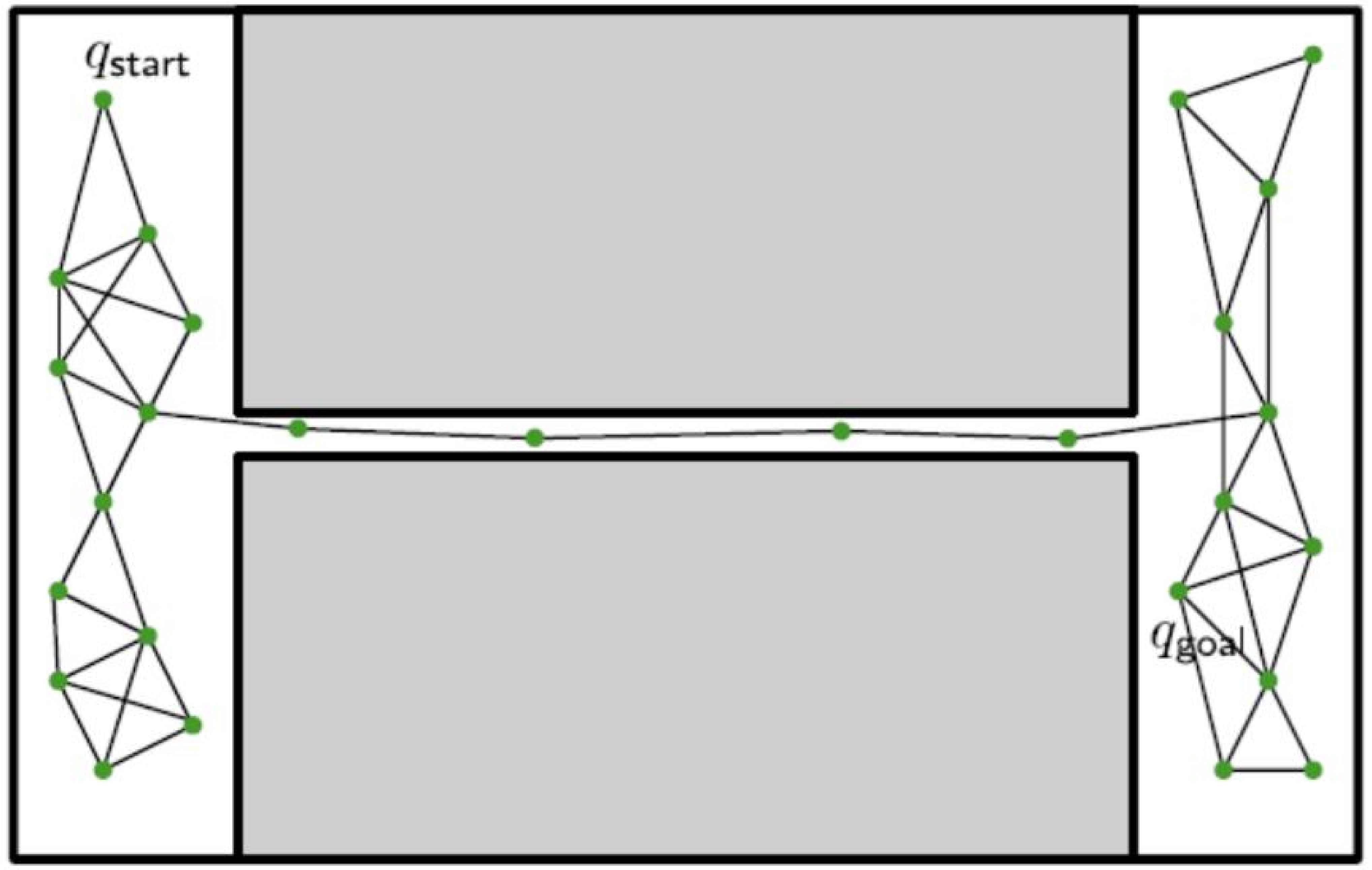
Example

Find the path from start state q_{start} to goal state q_{goal}



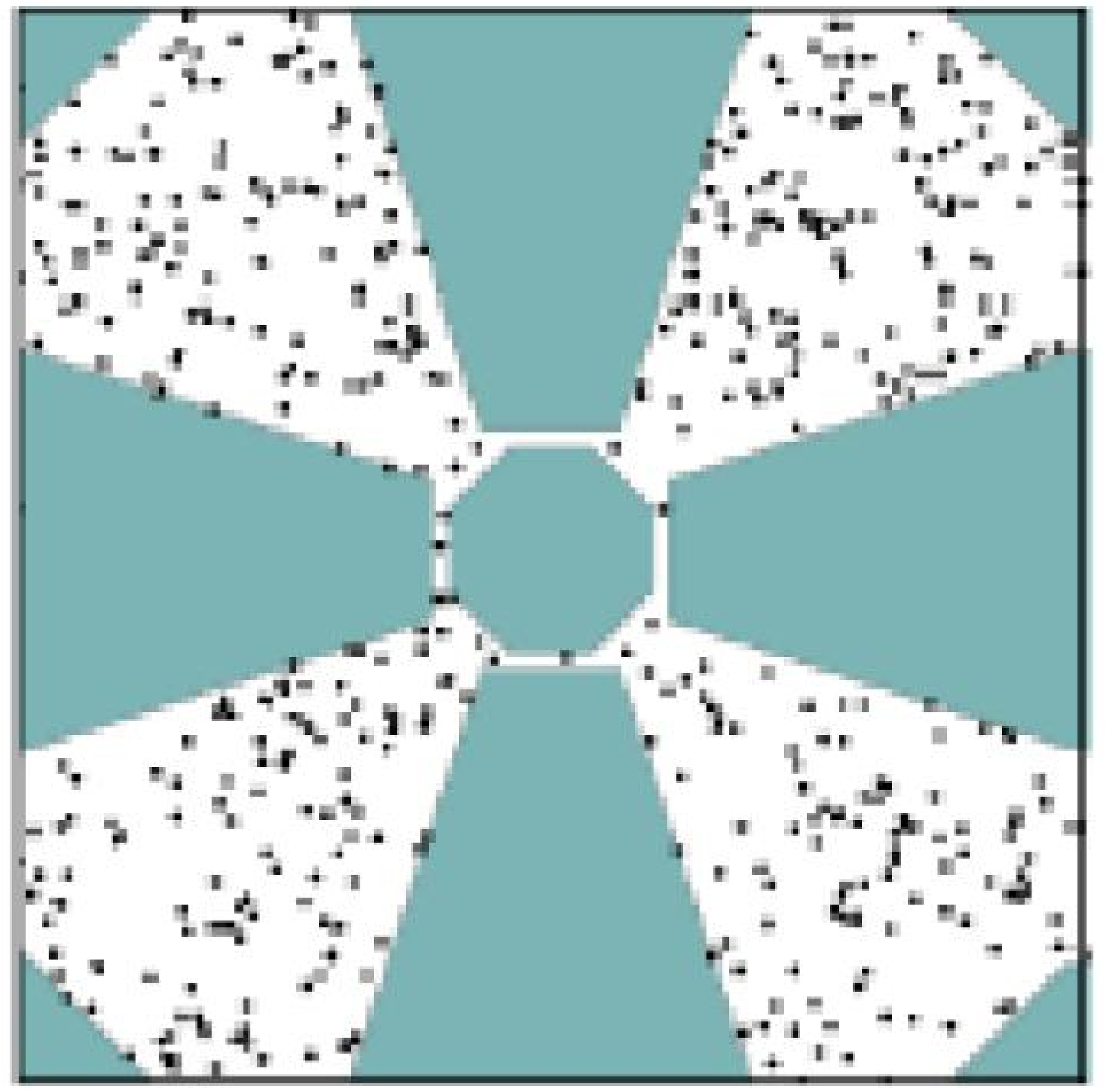
Limitations: Narrow Passages

It is unlikely to sample the points in the narrow bridge

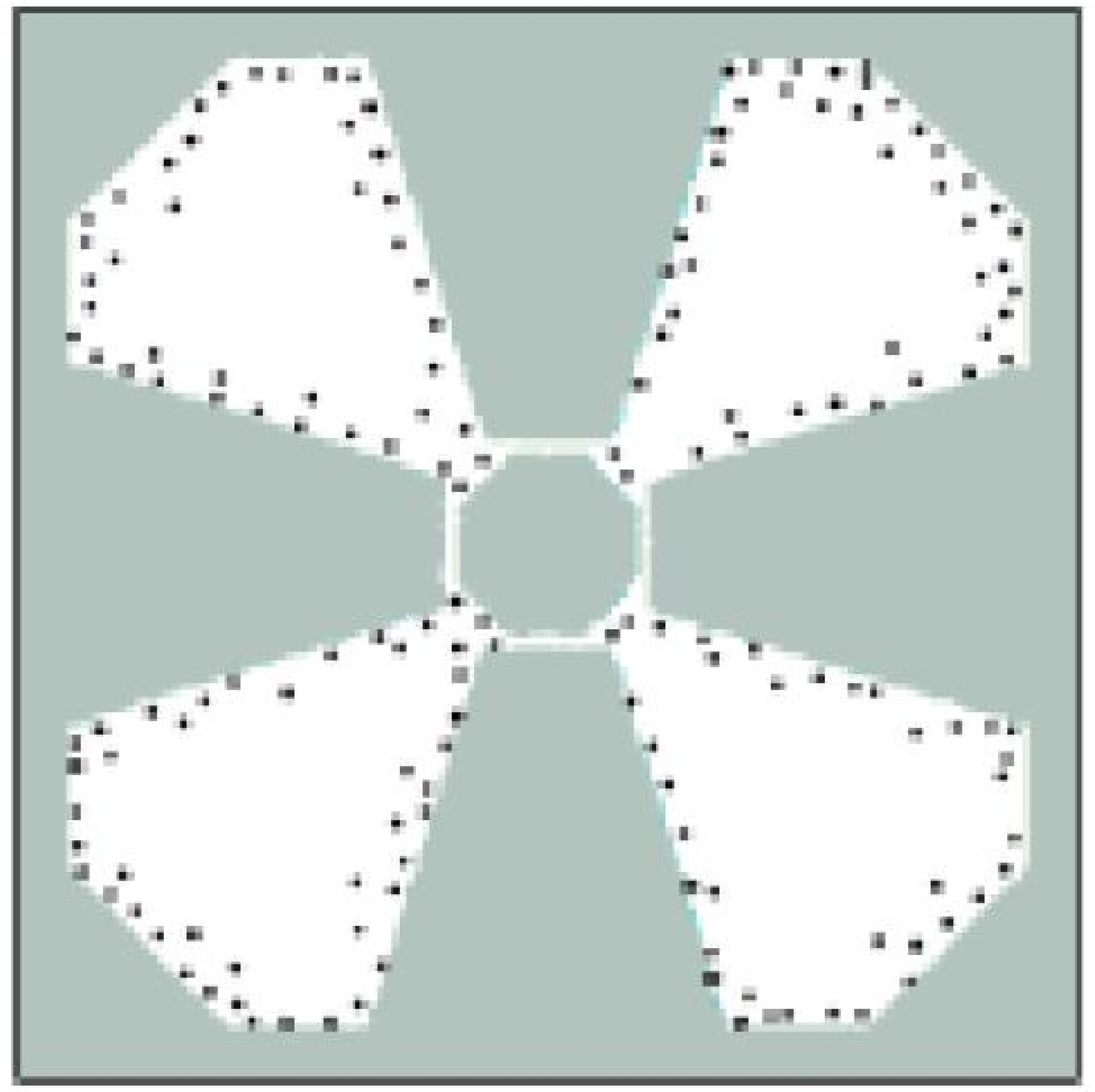


Gaussian Sampling

- Generate one sample q_1 uniformly in the configuration space
- Generate another sample q_2 from a Gaussian distribution $\mathcal{N}(q_1, \sigma^2)$
- If $q_1 \in \mathcal{C}_{free}$ and $q_2 \neq \mathcal{C}_{free}$ then add q_1



Uniform sampling

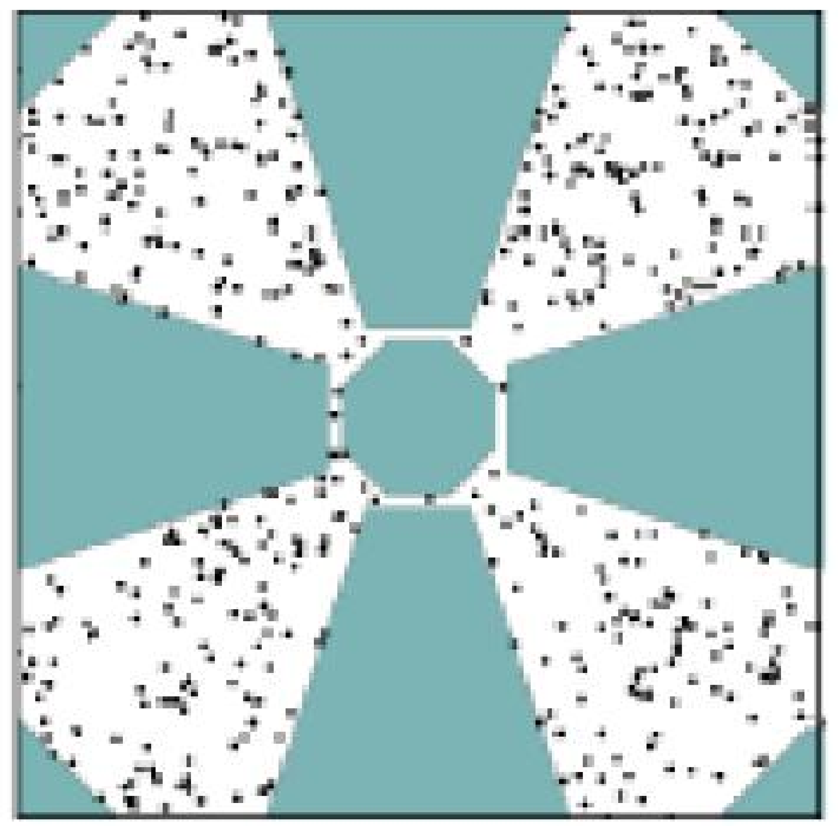


Gaussian sampling

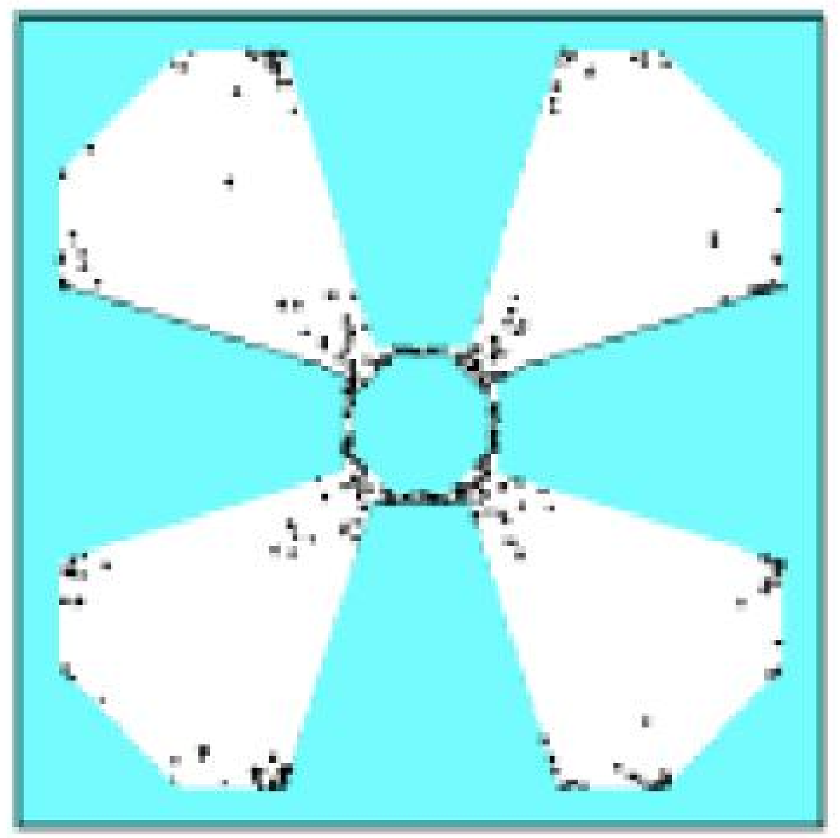
Read by Yourself

Bridge Sampling

- Generate one sample q_1 uniformly in the configuration space
- Generate another sample q_2 from a Gaussian distribution $\mathcal{N}(q_1, \sigma^2)$
- $q_3 = \frac{q_1+q_2}{2}$
- If q_1, q_2 are not in $\mathcal{C}_{free}$ then add q_3



Uniform sampling



Bridge sampling

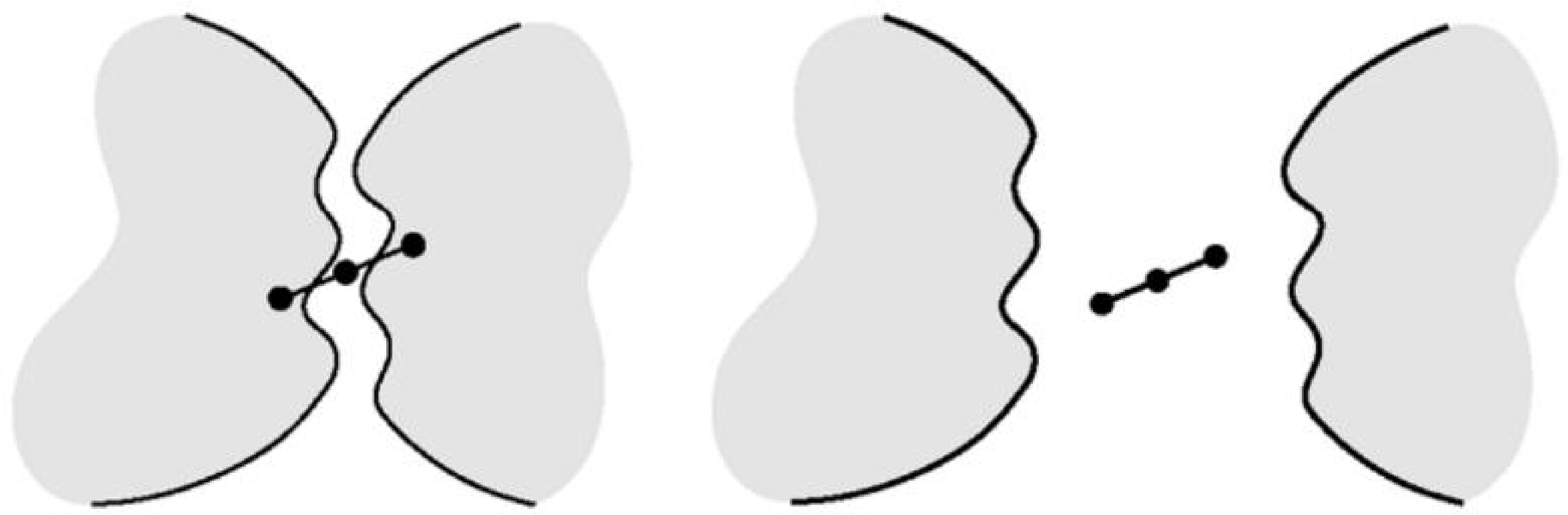


Fig. 2. Building short bridges is much easier in narrow passages (left) than in wide-open free space (right).

Read by Yourself

Rapidly-exploring Random Trees (RRT)

Rapidly-exploring Random Tree(RRT)

- RRT grows a tree rooted at the start state by using random samples from configuration space.
- As each sample is drawn, a connection is attempted between it and the nearest state in the tree. If the connection is in the configuration space, this results in a new state in the tree.

Extend Operation

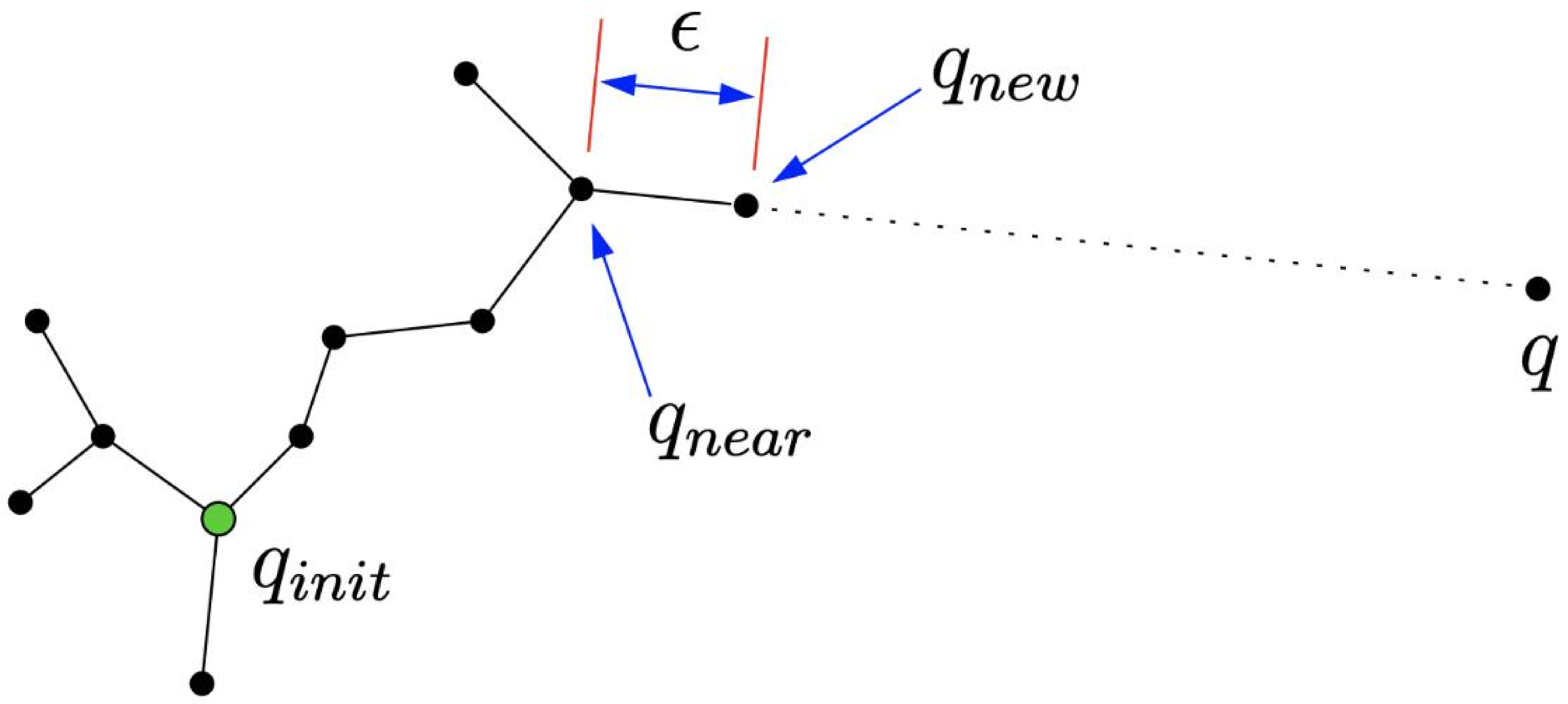


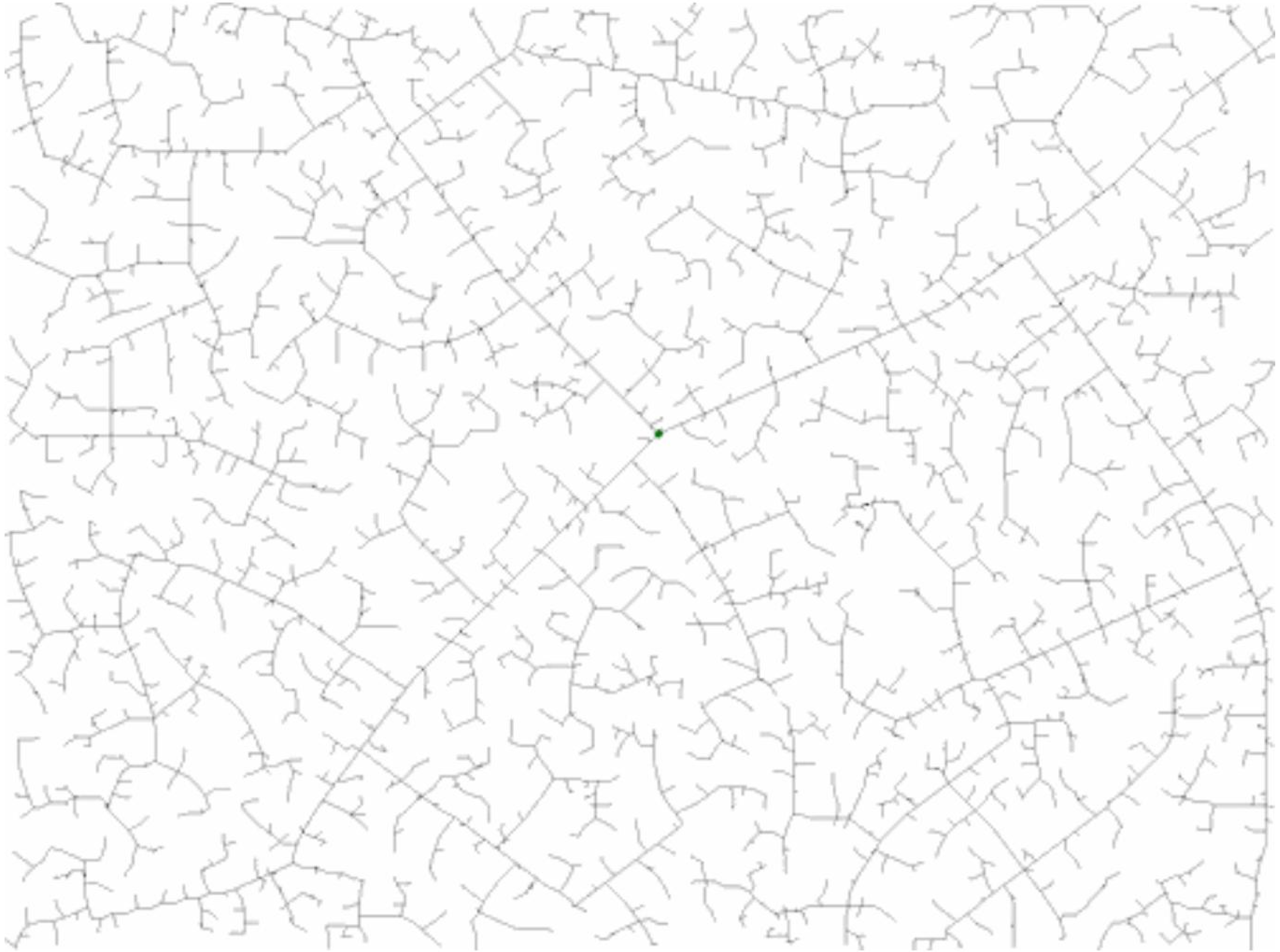
Figure 3: The EXTEND operation.

Pipeline

Input: n: number of sampled nodes in the tree, ϵ is the stepsize, β is the probability of sampling q_{goal}, q_{start}, q_{goal}.

$V \leftarrow \{q_{start}\}$;
$E \leftarrow \emptyset$;
for $i = 1 \rightarrow n$ **do**
  if *rand*$(0, 1)$ < β **then**
    $q_{target} \rightarrow q_{goal}$
  end
  else
    $q_{target} \rightarrow$ uniformly random sample from C_{free}
  end
  $q_{near} \rightarrow$ nearest neighbor of q_{target} in V ;
  $q_{new} \rightarrow q_{near} + \frac{\epsilon}{|q_{near} - q_{target}|}(q_{near} - q_{target})$;
  if $q_{new} \in C_{free}$ *and* $(q_{near}, q_{new}) \in C_{free}$ **then**
    $V \rightarrow V \cup \{q_{new}\}$;
    $E \rightarrow E \cup \{(q_{near}, q_{new})\}$;
  end
end
Find a path from q_{start} to q_{goal} with Dijkstra algorithm;

Examples



Challenges

- Find nearest neighbor in the tree
  - We need to support online quick query
  - Examples: KD Trees
- Need to choose a good ϵ to expand the tree efficiently
  - Large ϵ: hard to generate new samples
  - Small ϵ: too many samples in the tree

RRT-Connect

- Grow two trees starting from q_{start} and q_{start} respectively instead of just one.
- Grow the trees towards each other rather than random configurations
- Use stronger greediness by growing the tree with multiple epsilon steps instead of a single one.

Kuffner, James J., and Steven M. LaValle. "RRT-connect: An efficient approach to single-query path planning." Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065). Vol. 2. IEEE, 2000.

Pseudo Code

1 of 1

Automatic Zoom

RANDOM_CONFIG(q_{goal})

if $rand(0,1) < \beta$ **then**
  return q_{goal}
else
  return random sample from C_{free}
end if

EXTEND($\mathcal{T}, q$)

$q_{near} \leftarrow$ NEAREST_NEIGHBOR($\mathcal{T}, q$)
if $|q_{near} - q| < \epsilon$ **then**
  $q_{new} \leftarrow q$
else
  $q_{new} \leftarrow q_{near} + \frac{\epsilon}{|q - q_{near}|}(q - q_{near})$
end if
if $q_{new} \in C_{free}$ **and** $(q_{near}, q_{new}) \in C_{free}$ **then**
  $\mathcal{T}$.add_vertex(q_{new})
  $\mathcal{T}$.add_edge((q_{near}, q_{new}))
  if $q_{new} = q$ **then**
    return *Reached*
  else
    return *Advanced*
  end if
else
  return *Trapped*
end if

CONNECT($\mathcal{T}, q$)

repeat
  $S \leftarrow$ EXTEND($\mathcal{T}, q$)