Università
degli Studi
di Padova

IT+Robotics

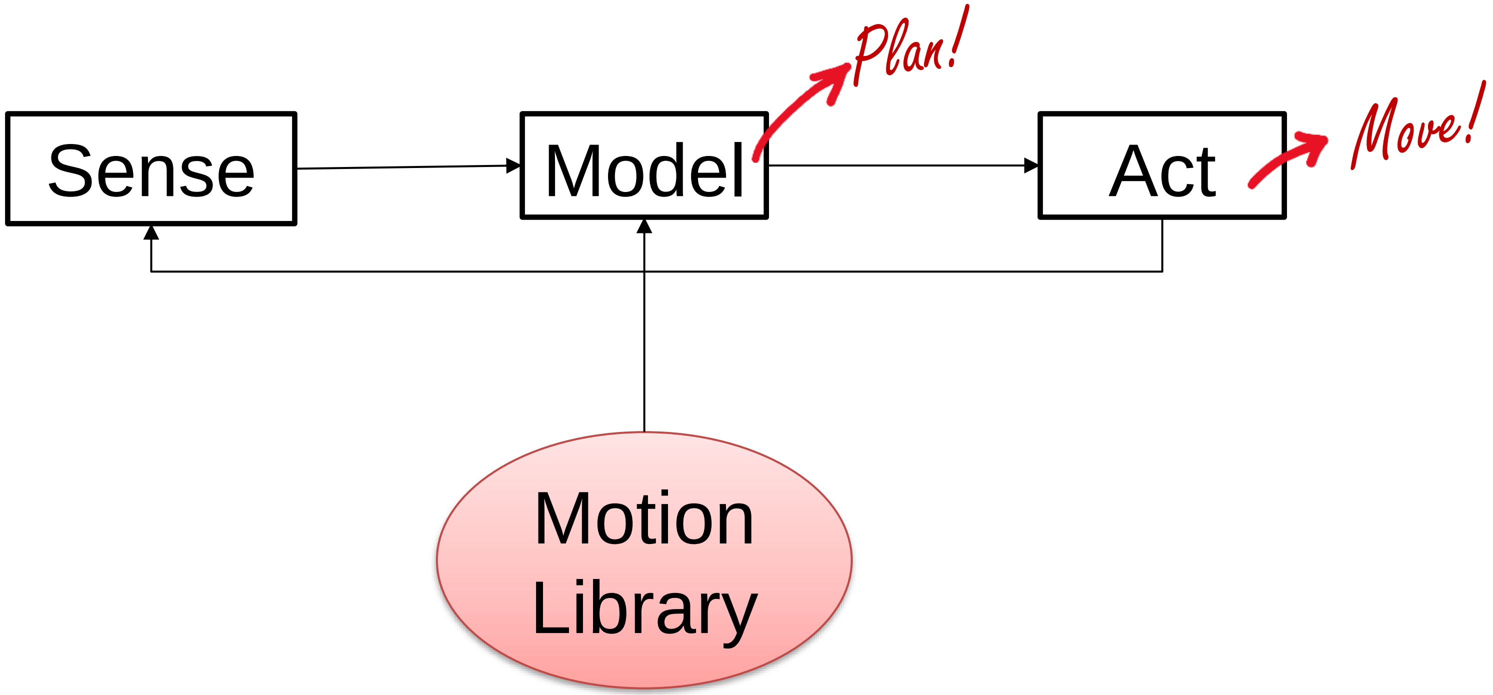
Motion Planning

Alberto Gottardi

27/11/2024

Special Thanks to Elisa Tosello

Paradigm: Sense-Model-Act

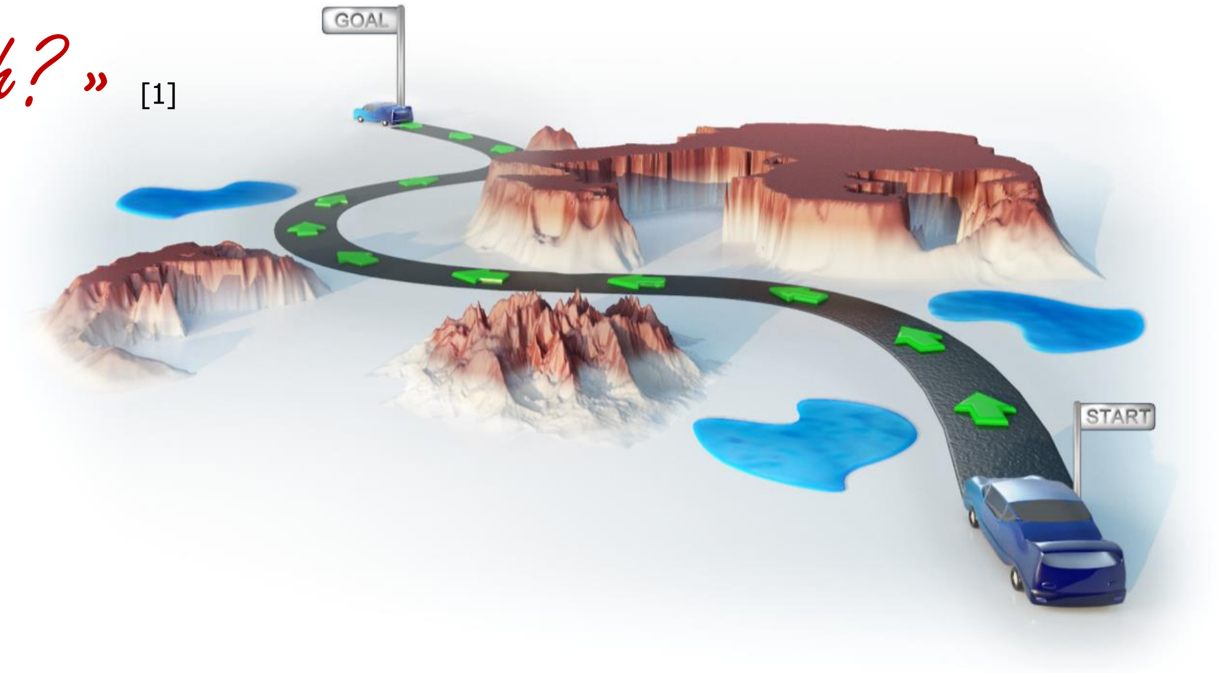


Motion Planner: Unformal Definition

Motion Planning is the ability for an agent to compute its own motion in order to achieve certain goals. All **automous robots** must have this ability.

Motion Planning: Foundamental Question

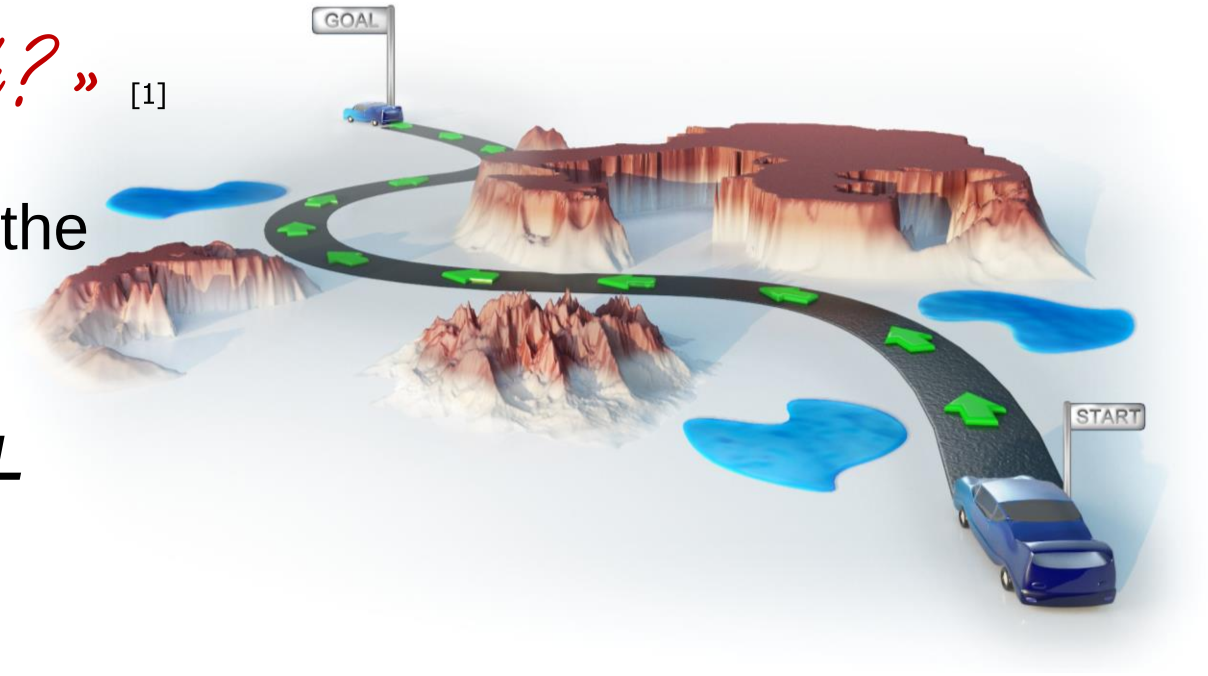
« Are two given points connected by a path? » [1]



[1] From Jean-Claude Latombe, CS326A, WEB Site, Stanford University, USA, 2004

Motion Planning: Foundamental Question

« Are two given points connected by a path? » [1]



Based on this question, we can define the **MP PROBLEM** as:

*Finding a path from **START** to **GOAL** without collisions*

[1] From Jean-Claude Latombe, CS326A, WEB Site, Stanford University, USA, 2004

Motion Planning: Basic Problem

Statement

Compute a **collision-free path** for a rigid or articulated object among static obstacles

Inputs:

- Geometry of moving object and obstacles
- Kinematics of moving object (degrees of freedom)
- Initial and goal configurations (placements)

Output:

Continuous **sequence of collision-free** object **configurations** connecting the initial and goal configurations

Definitions: Configuration Space

- **Workspace (W):** STATIC environment populated by obstacles ($W = R^N$, N = 2 or N = 3). In particular O C W (closet set) is the obstacle region.

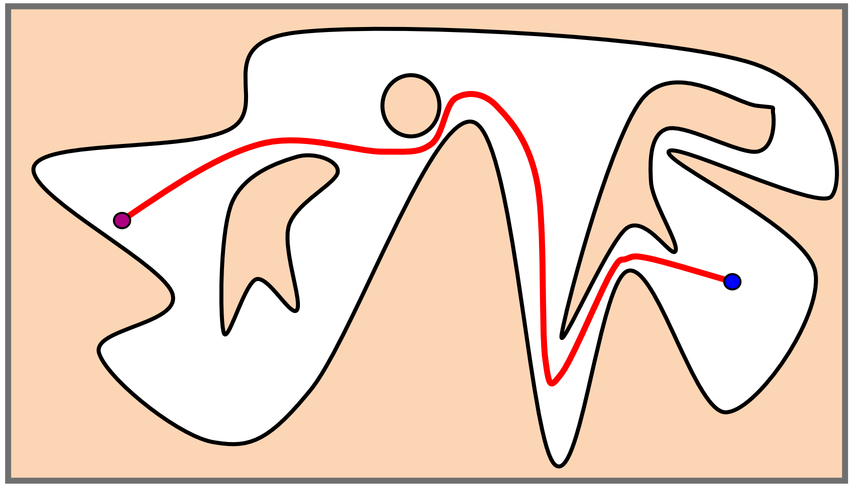
- **Robot configuration (R)** : poses of all points that compose a robot in a certain time according to a certain coordinate system

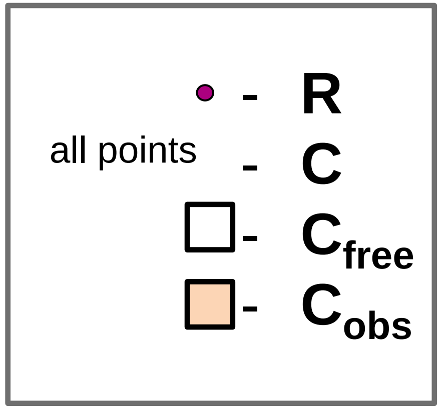
- **Configuration Space (C)**: set of all possible configurations q.

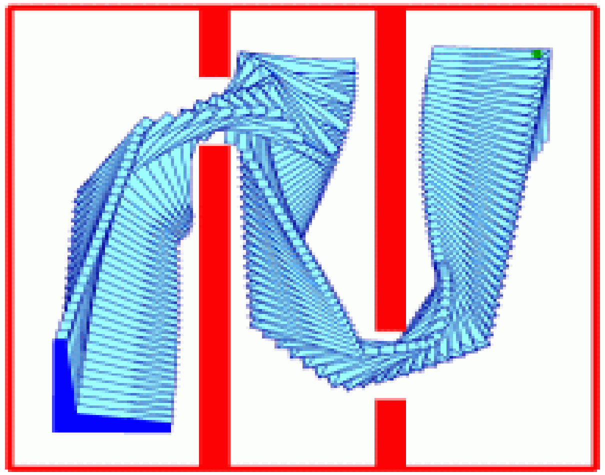
  - A configuration q is a complete specification of the location of every point on the robot geometry.
  - R(q) C W is the set of points occupied by the robot when at configuration q ∈ C

- **Collision Space (C_{obs})**: colliding configurations. $C_{obs} = \{q \in C \mid R(q) \cap O \neq \varnothing\}$

- **Free Space (C_{free})**: collision-free configurations. $C_{free} = C \setminus C_{obs}$

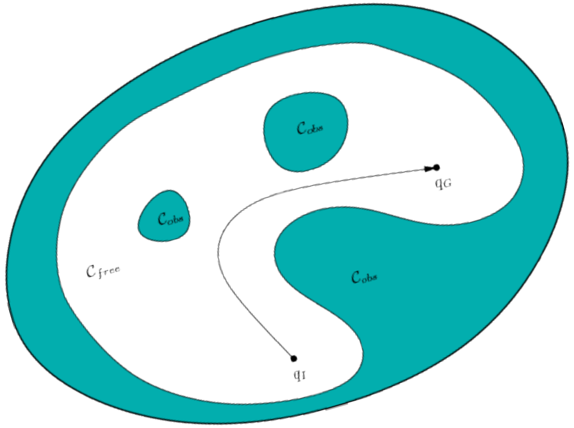






Statement

- Given
  - A workspace W, where either $W = R^2$ or $W = R^3$
  - An obstacle region $O \subset W$
  - A robot defined in W. Either a rigid body R or a collection of m links: A_1, A_2, …, A_m.
  - The configuration space C (C_{obs} and C_{free} are then defined)
  - An initial configuration $q_I \in C_{free}$
  - A goal configuration $q_G \in C_{free}$. The initial and goal configurations are often called a query (q_I, q_G)



Formal Definition

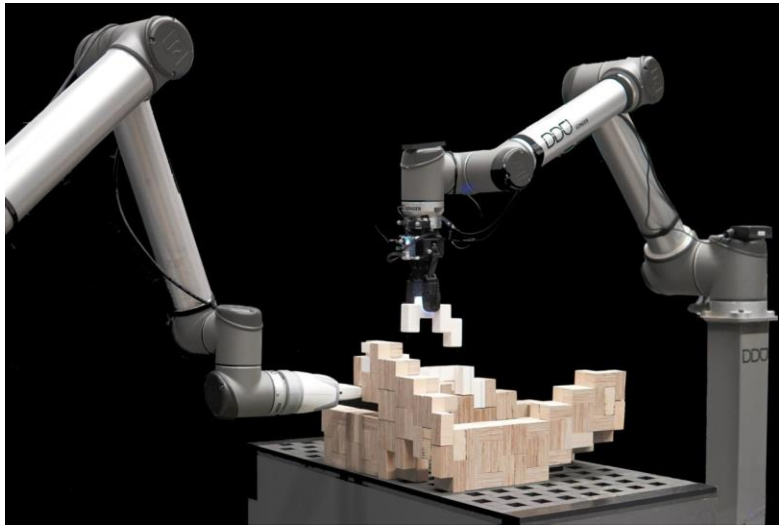
Compute a (continuous) path **$\tau:[0, 1] \rightarrow C_{free}$, such that $\tau(0) = q_I$ and $\tau(1) = q_G$**

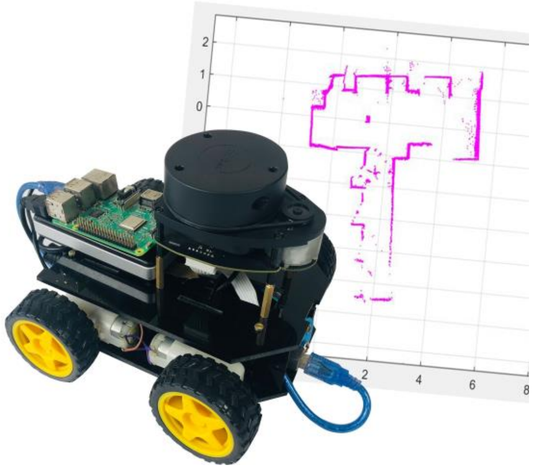
Compute motion strategies, e.g.:

- geometric paths
- time-parameterized trajectories
- sequence of sensor-based motion commands

To achieve high-level goals, e.g.:

- go to A without colliding with obstacles
- assemble product P
- build map of environment E
- find object O





Path Planning vs. Motion Planning

PATH PLANNING referes to the **purely geometric problem** of computing a collision-free path for a robot among static obstacles.

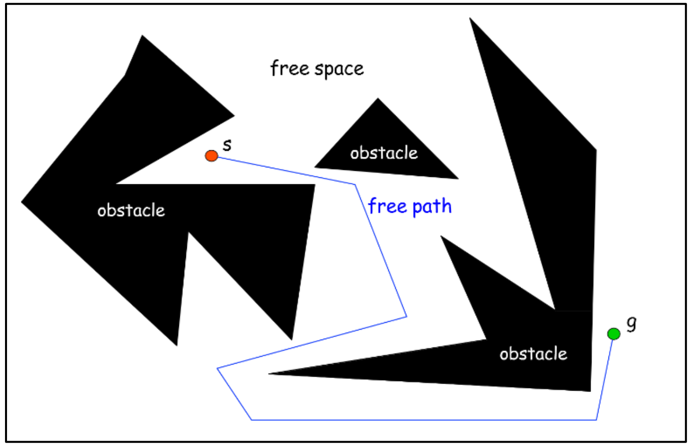
MOTION PLANNING is used for problems **involving time, dynamic constraints**, object coordination, sensory interaction, etc.

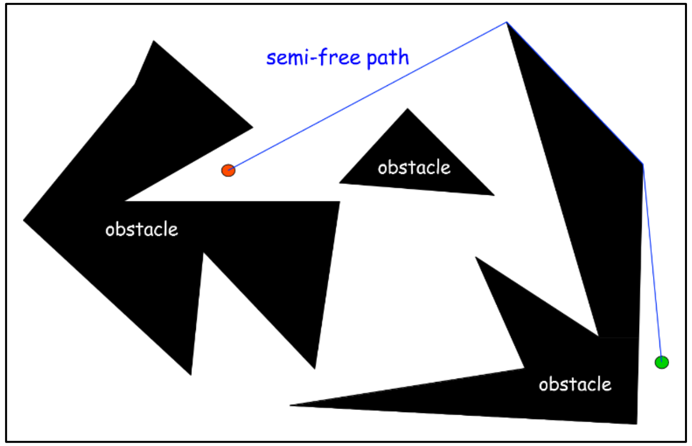
Path Planning vs. Motion Planning

PATH PLANNING referes to the **purely geometric problem** of computing a collision-free path for a robot among static obstacles.

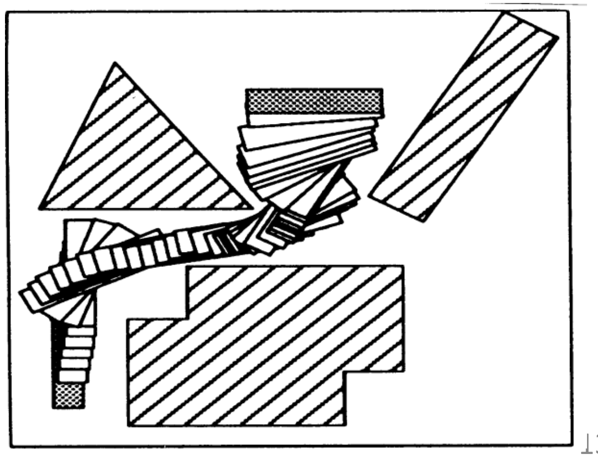
MOTION PLANNING is used for problems **involving time, dynamic constraints**, object coordination, sensory interaction, etc.

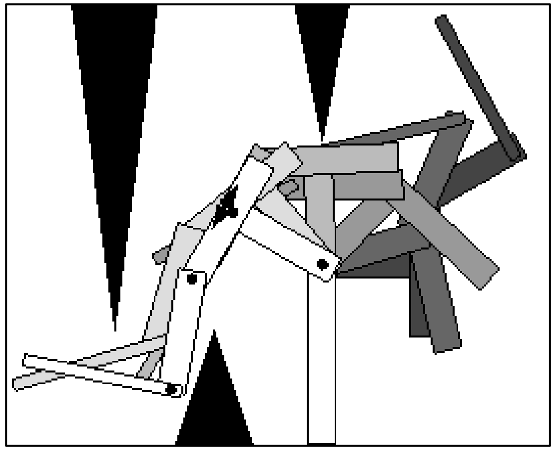
- **Path**: it is a geometric concept and stands for a line in a certain space (the space of Cartesian positions, the space of the orientations, the joint space,..) to be followed by the object whose motion has to be planned
- **Timing law**: it is the time dependence with which we want the robot to travel along the assigned path
- **Trajectory**: it is a path over which a timing law has been assigned. Actual output of the MP.





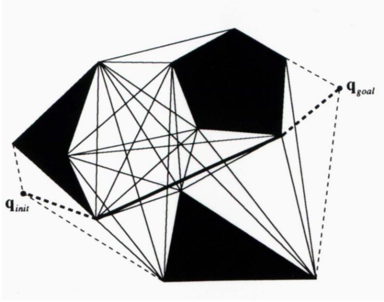
Robots have different shapes and kinematics. *How can we better define a path?*



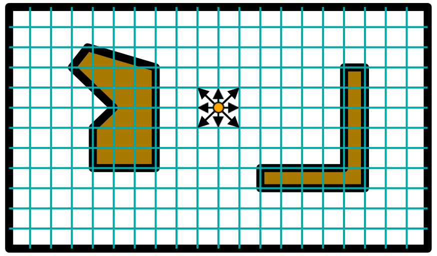


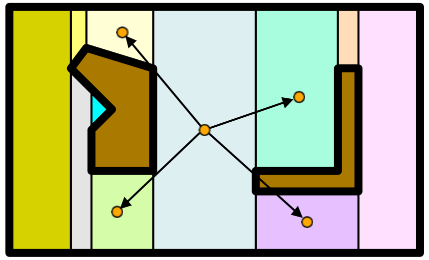
Classical Planning Approaches

Combinatorial Planning

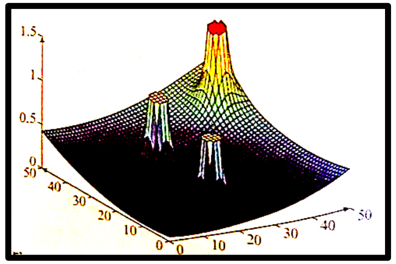


Sampling-based Planning



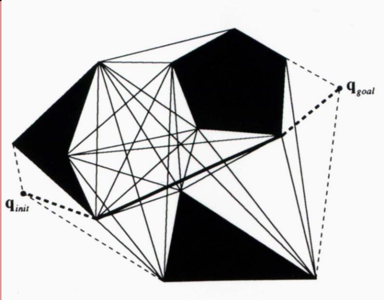


Artificial Potential Fields

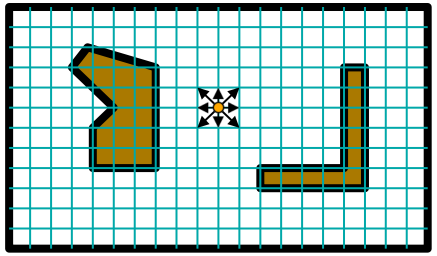


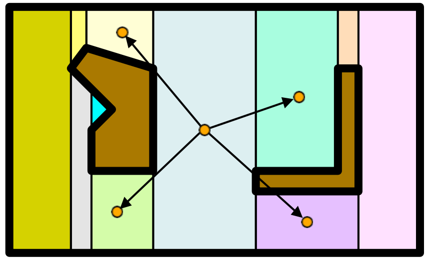
Classical Planning Approaches

Combinatorial Planning

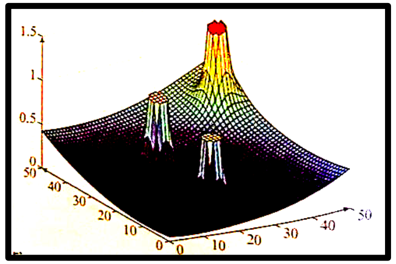


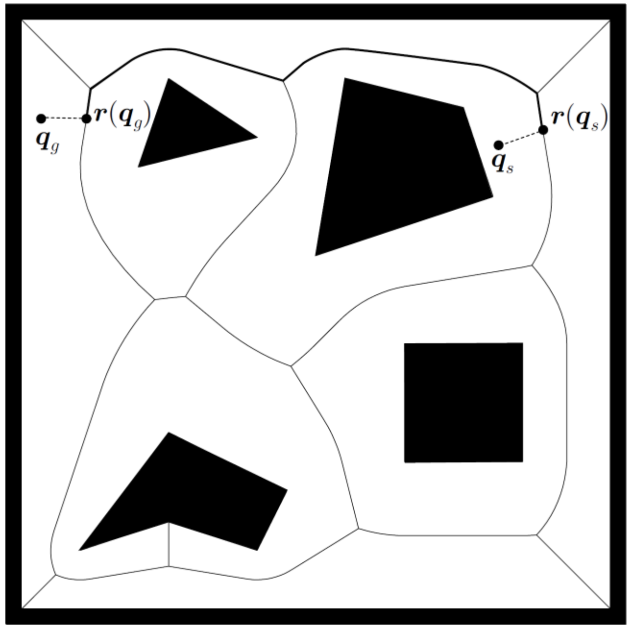
Sampling-based Planning



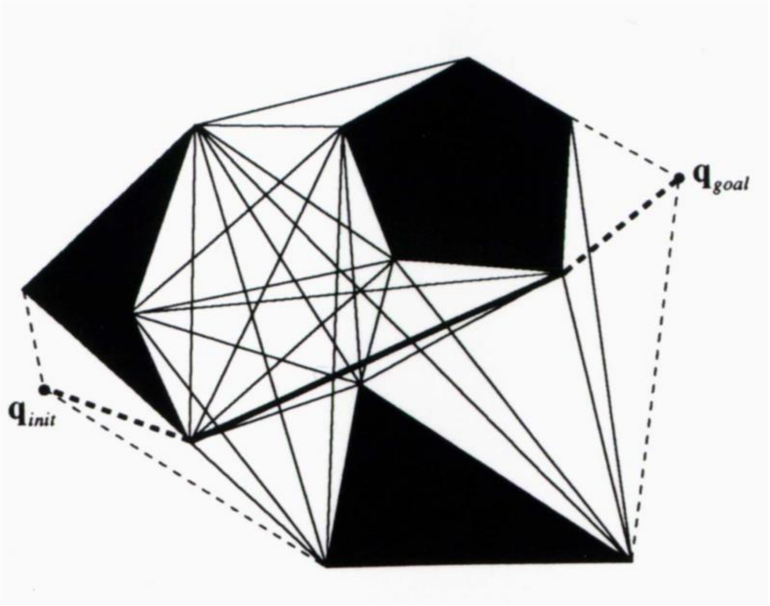


Artificial Potential Fields

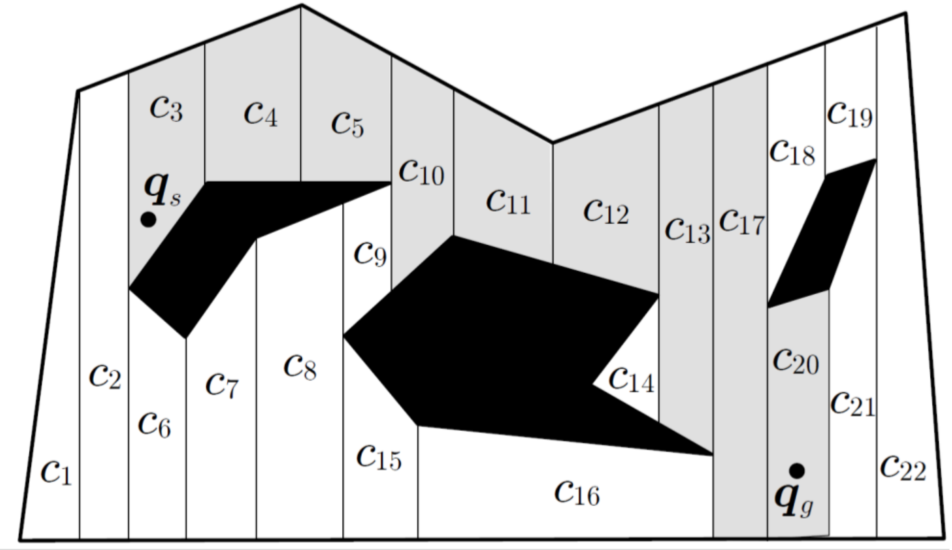




Generalized Voronoi Diagram



Visibility Graph

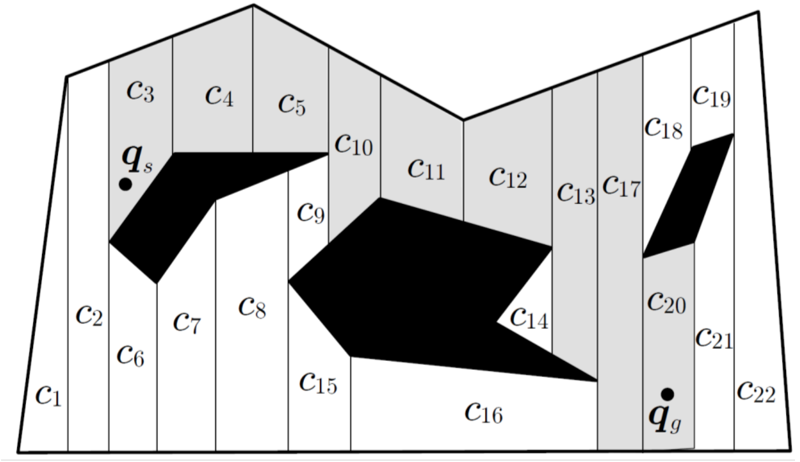


Exact Cell Decomposition

Source: Siciliano B., Sciavicco L. , Villani L. , Oriolo G. , Robotics: modelling, planning and control, Springer Science & Business Media, 2010

Exact Cell Decomposition

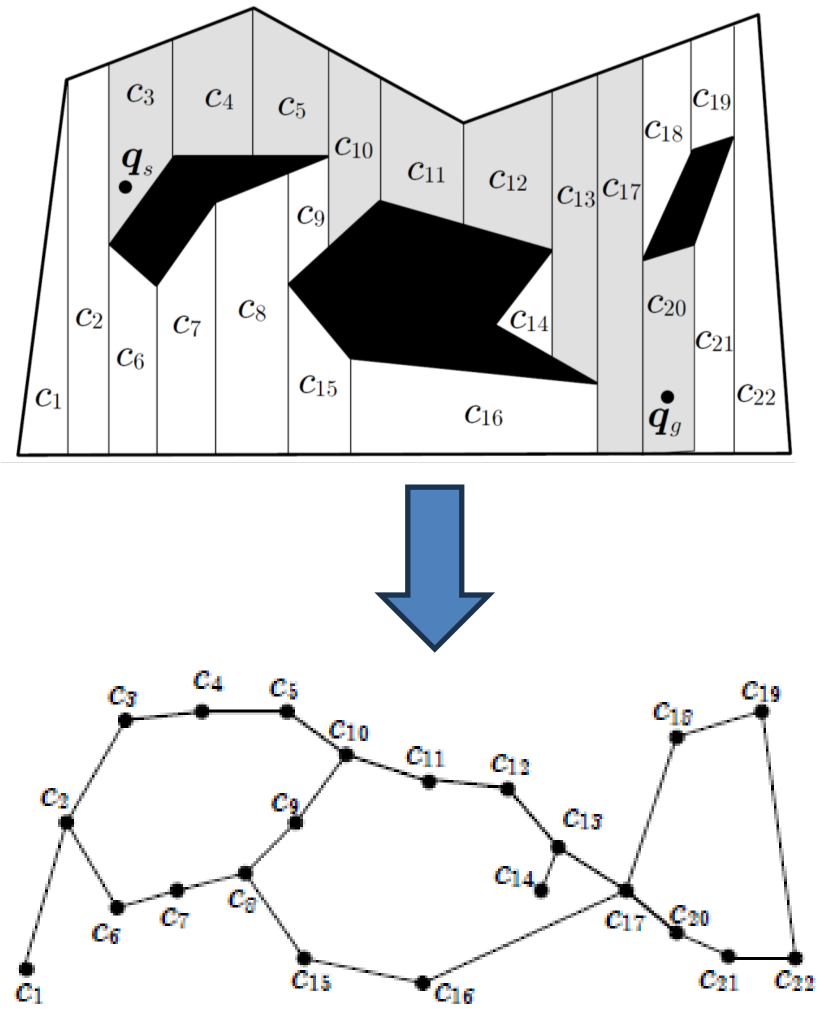
- **Idea**: decompose C_{free} into cells (typically convex polygons)
- Convexity guarantees that the line segments joining two configurations belonging to the same cell lies entirely in the cell itself, and therefore in C_{free}.



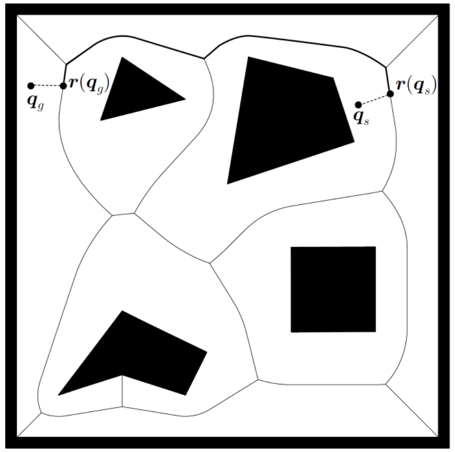
Source: Siciliano B., Sciavicco L. , Villani L. , Oriolo G. , Robotics: modelling, planning and control, Springer Science & Business Media, 2010

Exact Cell Decomposition

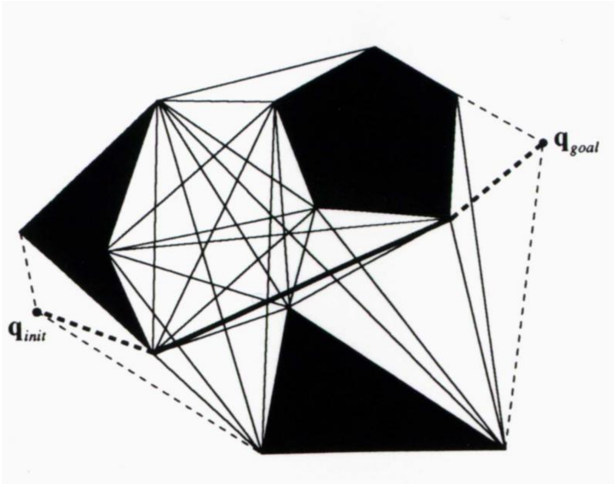
- **Idea**: decompose C_{free} into cells (typically convex polygons)
- Convexity guarantees that the line segments joining two configurations belonging to the same cell lies entirely in the cell itself, and therefore in C_{free}.
- After the decomposition, build the associated **connectivity graph C**
- Identify the cells c_s and c_g that contain $\mathbf{q_s}$ and $\mathbf{q_g}$
- Use a **graph search** algorithm to find a collision-free path quickly
- The algorithm has **complexity O(nlogn)**, where n is the #vertices using the **plane-sweep** algorithm



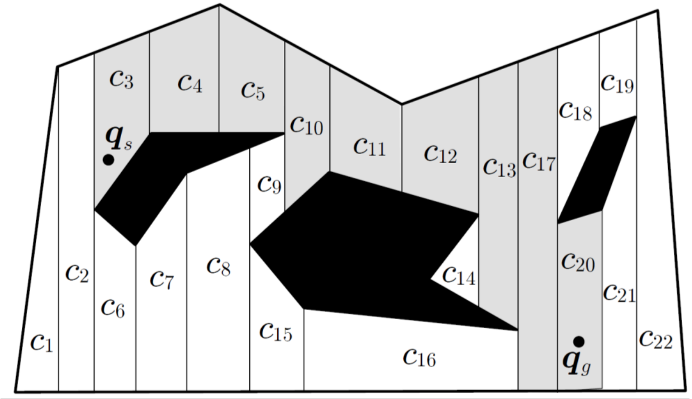
Source: Siciliano B., Sciavicco L. , Villani L. , Oriolo G. , Robotics: modelling, planning and control, Springer Science & Business Media, 2010



Generalized Voronoi Diagram



Visibility Graph

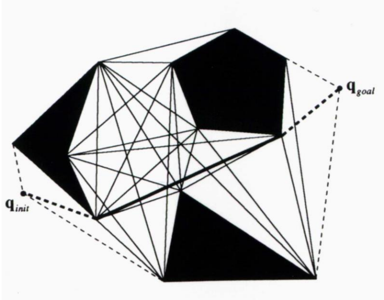


Exact Cell Decomposition

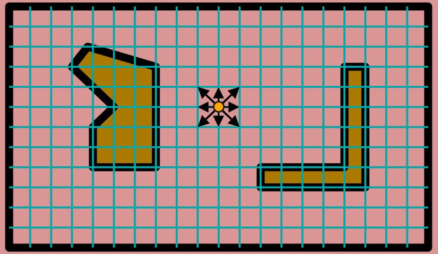
The combinatorial plannings are elegant and complete, but intractable when C-space dimensionality increases.

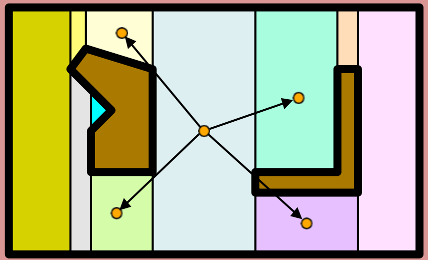
Source: Siciliano B., Sciavicco L. , Villani L. , Oriolo G. , Robotics: modelling, planning and control, Springer Science & Business Media, 2010

Combinatorial Planning

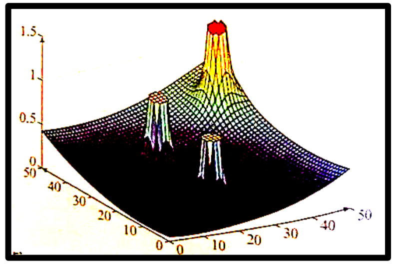


Sampling-based Planning





Artificial Potential Fields



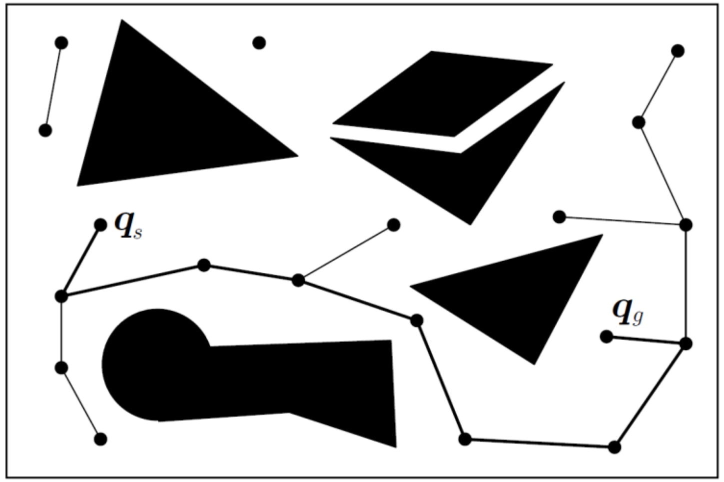
Classical Approaches: Sampling-based Planning

- Abandon the concept of explicitly describe and optimally explore C_{free} and C_{obs}, while rely on a sort of blind exploration of C_{free}, based on randomly sampling configuration points from C-space and connecting them to form a road map graph.

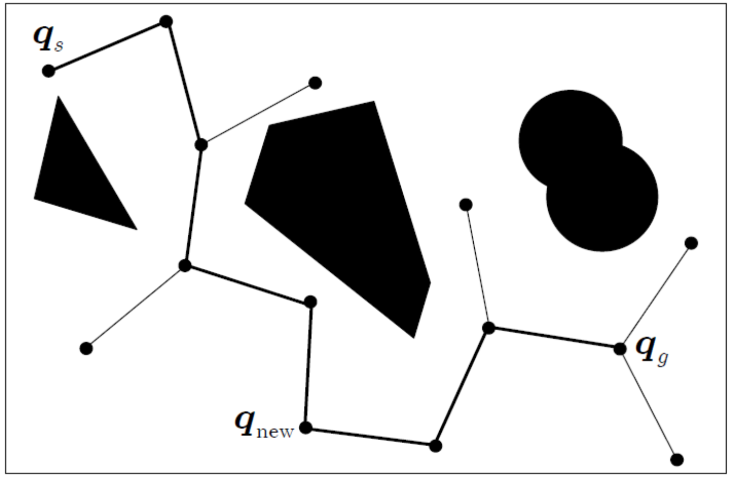
- This is realized by choosing at each iteration a sample configuration and checking if it entails a collision between the robot and the workspace obstacles. If the answer is affirmative, the sample is discarded. A configuration that does not cause a collision is instead added to the current roadmap and connected if possible to other already stored configurations.

- Deterministic vs. randomized approach to generate the sample

Probabilistic Roadmap (PRM)



Rapidly-exploring Random Tree (RRT)



Source: Siciliano B., Sciavicco L. , Villani L. , Oriolo G. , Robotics: modelling, planning and control, Springer Science & Business Media, 2010

Sampling-based planners provide a form of completeness

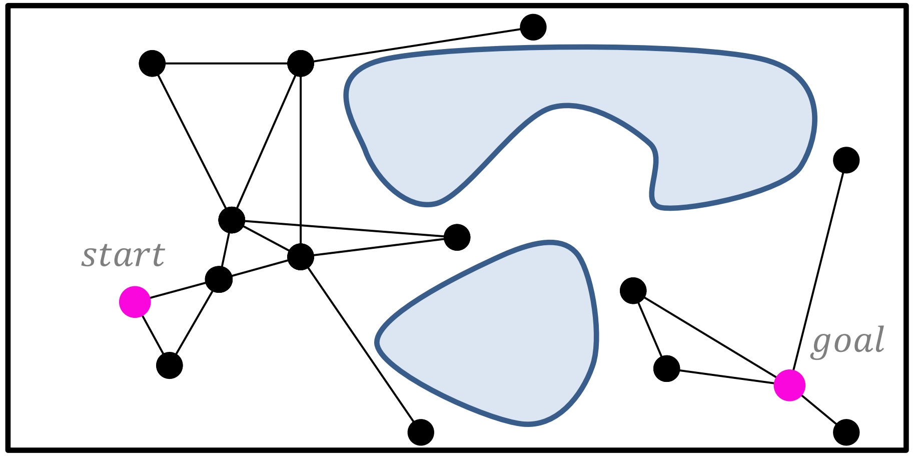
The probability of a planner finding a free path, if exists, tends to 1 as the execution time increases

Classical Approaches: Sampling-based Planning

PRM	PRM*	RRT	RRT*
RRTConnect	TRRT	BiTRRT	LazyRRT
KPIECE	SBL	EST	STRIDE

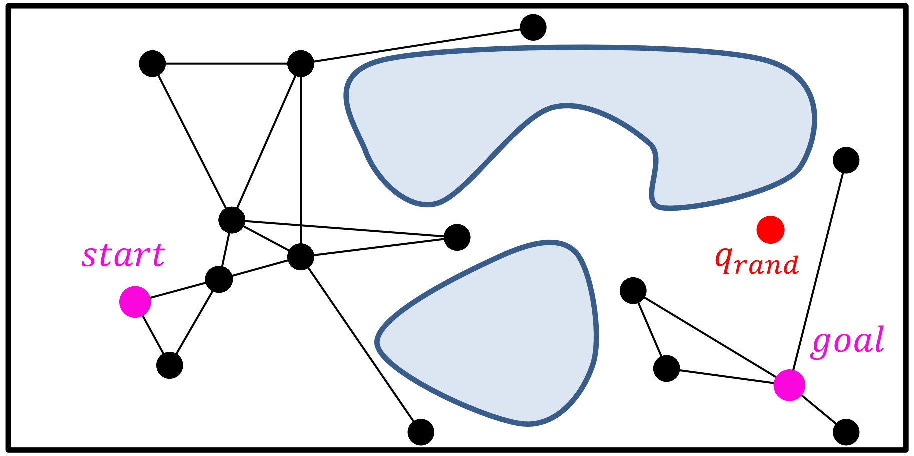
Probabilistic Roadmap
PRM

1) **LEARNING** → the algorithm expands the roadmap



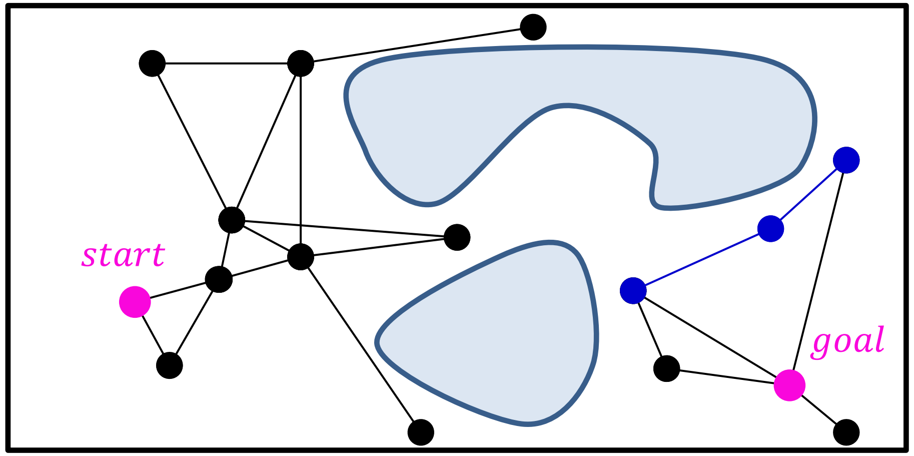
1) **LEARNING** → the algorithm expands the roadmap:

- Sample a random configuration in the free space (uniform probability distribution function)
- Connect the configuration to the k neighbors with distance < k (expansion strategy)



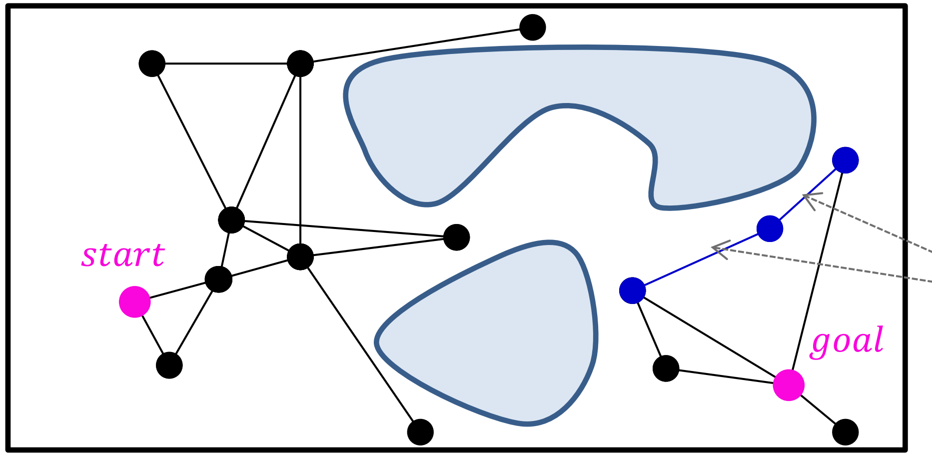
1) **LEARNING** → the algorithm expands the roadmap:

- Sample a random configuration in the free space (uniform probability distribution function)
- Connect the configuration to the k neighbors with distance < k (expansion strategy)



1) **LEARNING** → the algorithm expands the roadmap:

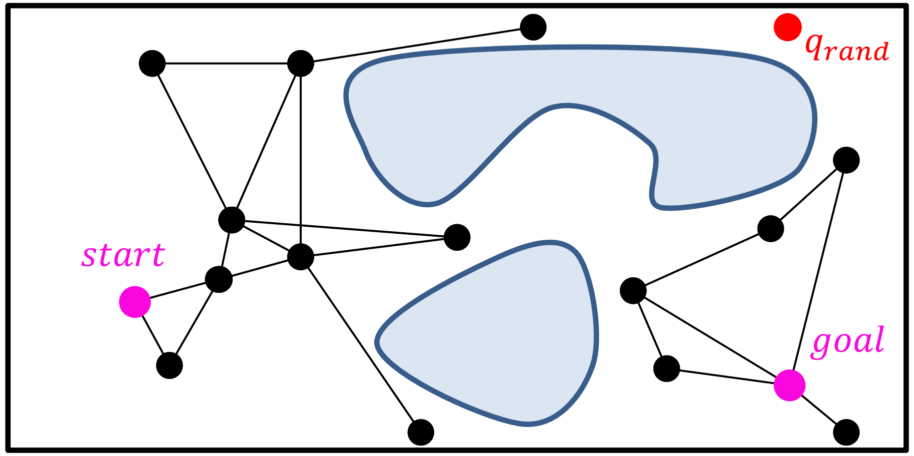
- Sample a random configuration in the free space (uniform probability distribution function)
- Connect the configuration to the k neighbors with distance < k (expansion strategy)



Parameter:
maximum number of edges
to be added to the roadmap

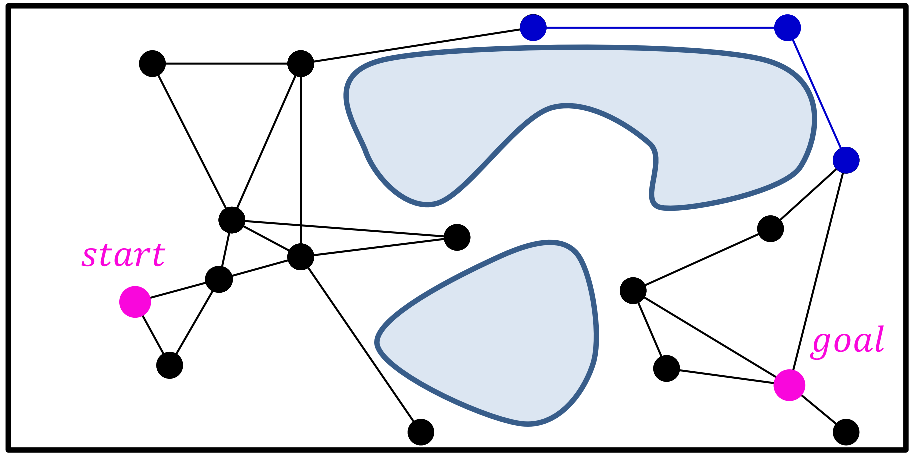
1) **LEARNING** → the algorithm expands the roadmap:

- Sample a random configuration in the free space (uniform probability distribution function)
- Connect the configuration to the k neighbors with distance < k (expansion strategy)



Probabilistic Roadmap (PRM)

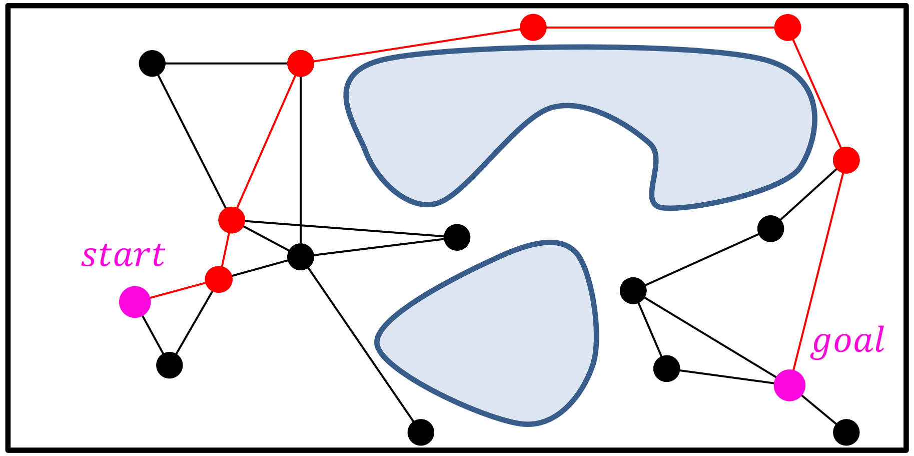
1) **LEARNING** → the algorithm expands the roadmap:
   - Sample a random configuration in the free space (uniform probability distribution function)
   - Connect the configuration to the k neighbors with distance < k (expansion strategy)

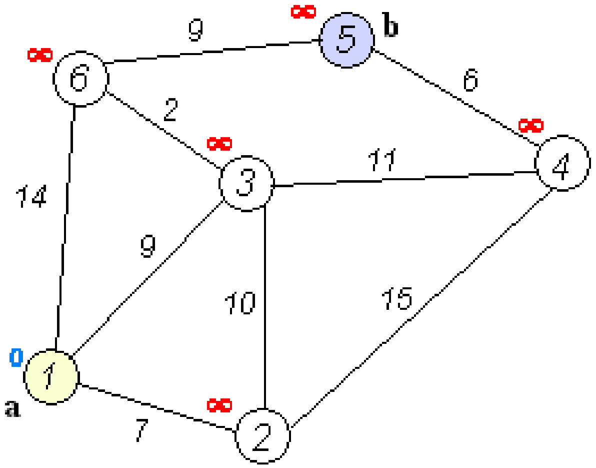


Probabilistic Roadmap (PRM)

2) SEARCH → the algorithm determines a solution through the roadmap:

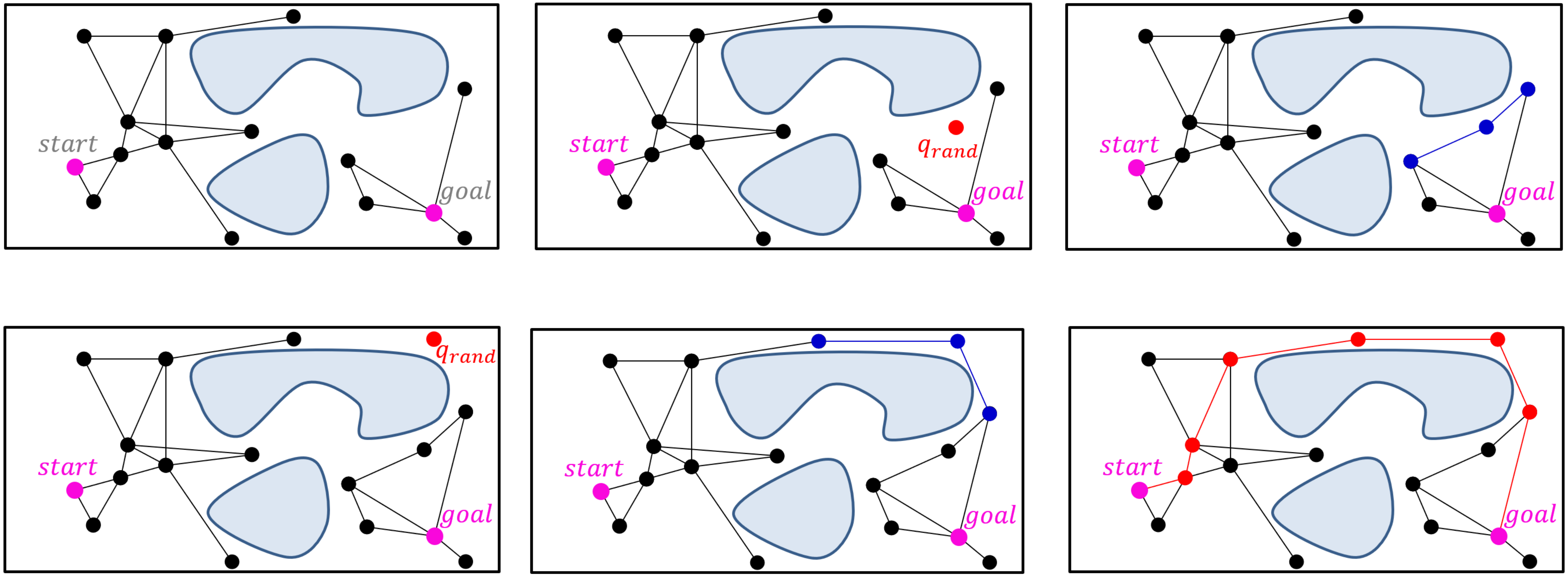
- Connect ***Start*** to ***Goal*** searching for the shortest path (use A* search or Dijkstra)





NOTE: the planner is probabilistic complete: increasing the number of the points, the probability of not finding the path goes to zero

Probabilistic Roadmap (PRM)



Probabilistic Roadmap (PRM)

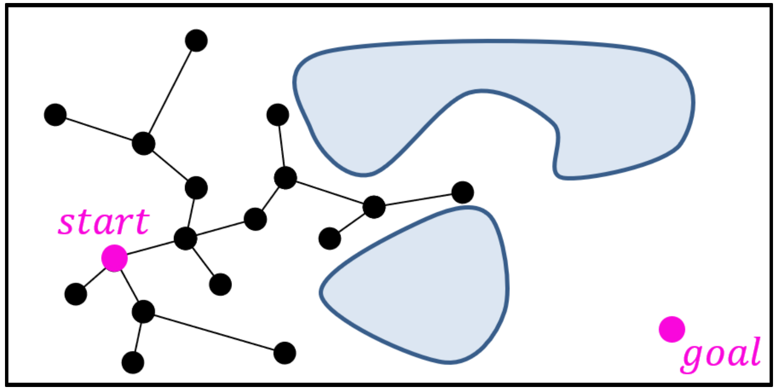
IDEA: Building a roadmap that connects *Start* and *Goal*.

PHASES:

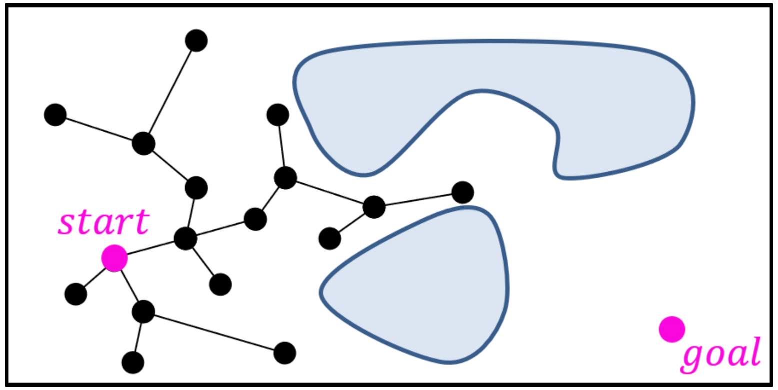
1) **LEARNING** → the algorithm expands the roadmap:
   - A random sample $\mathbf{q}_{rand}$ of the C-space is selected using a uniform probability distribution and tested for collision
   - If $\mathbf{q}_{rand}$ does not cause collisions it is added to a roadmap which is progressively being formed and connected (if possible) through free local paths to sufficiently "near" configurations already in the roadmap
   - The iterations terminate when either a maximum number of iterations has been reached or the number of connected components in the roadmap becomes smaller than a given threshold
2) **SEARCH** → the algorithm determines a solution through the roadmap:
   - Connect start and goal configurations to the roadmap
   - Search for a path using A* or Dijkstra

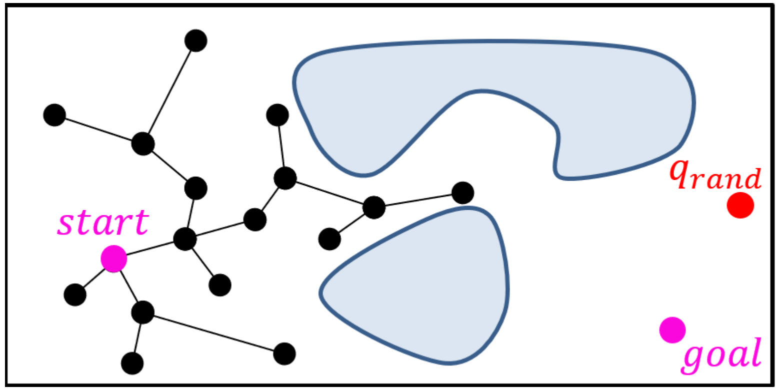
Rapidly-exploring Random Tree RRT

Rapidly-exploring Random Tree (RRT)

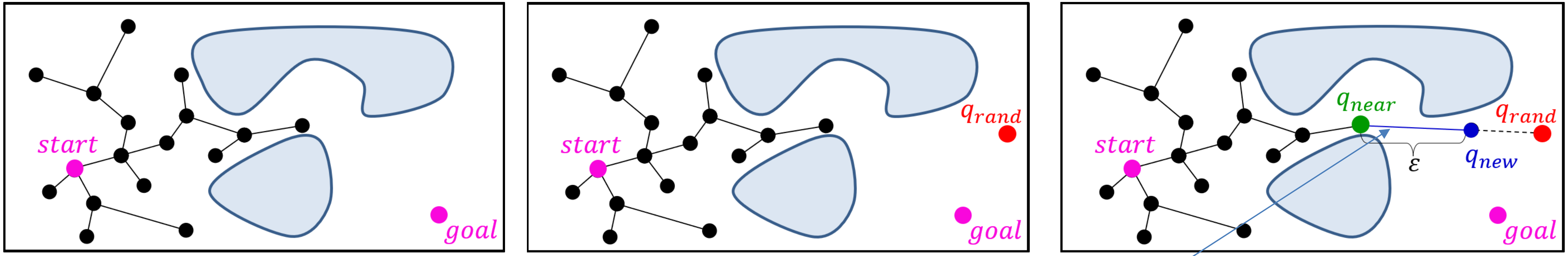


Rapidly-exploring Random Tree (RRT)



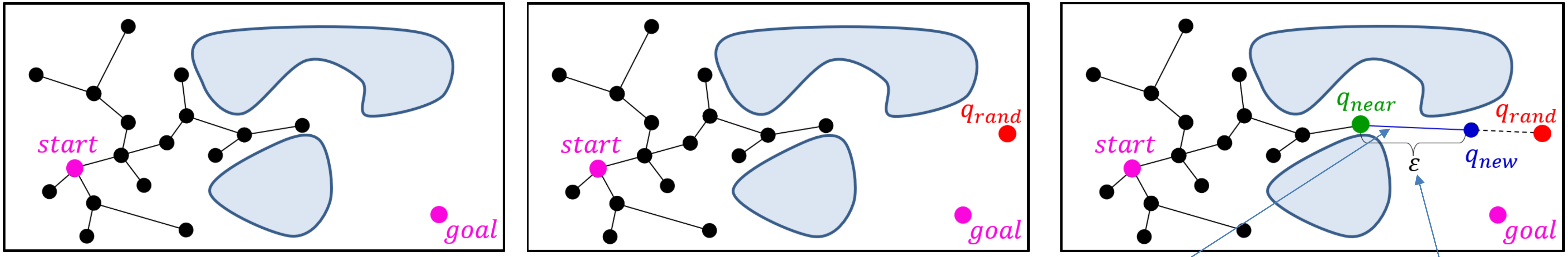


Rapidly-exploring Random Tree (RRT)



Expansion strategy towards q_{rand} led by a local planner (e.g., interpolation) starting from the nearest node (q_{near}).

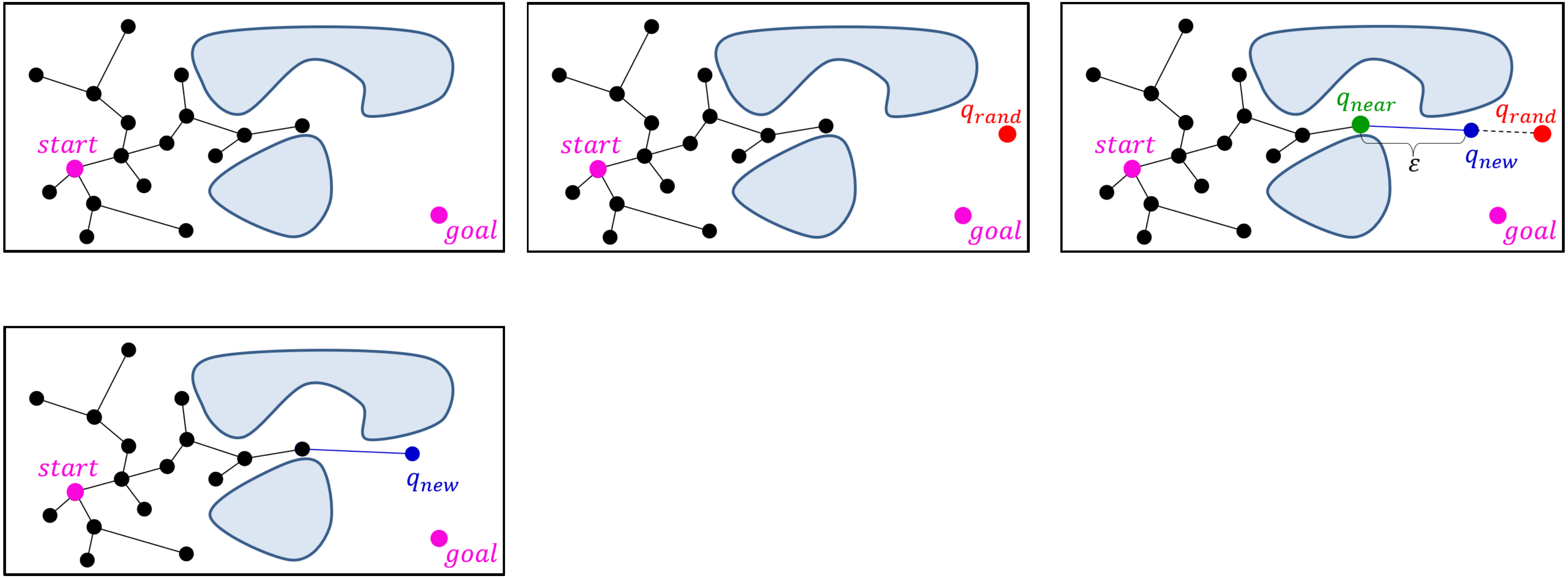
Rapidly-exploring Random Tree (RRT)



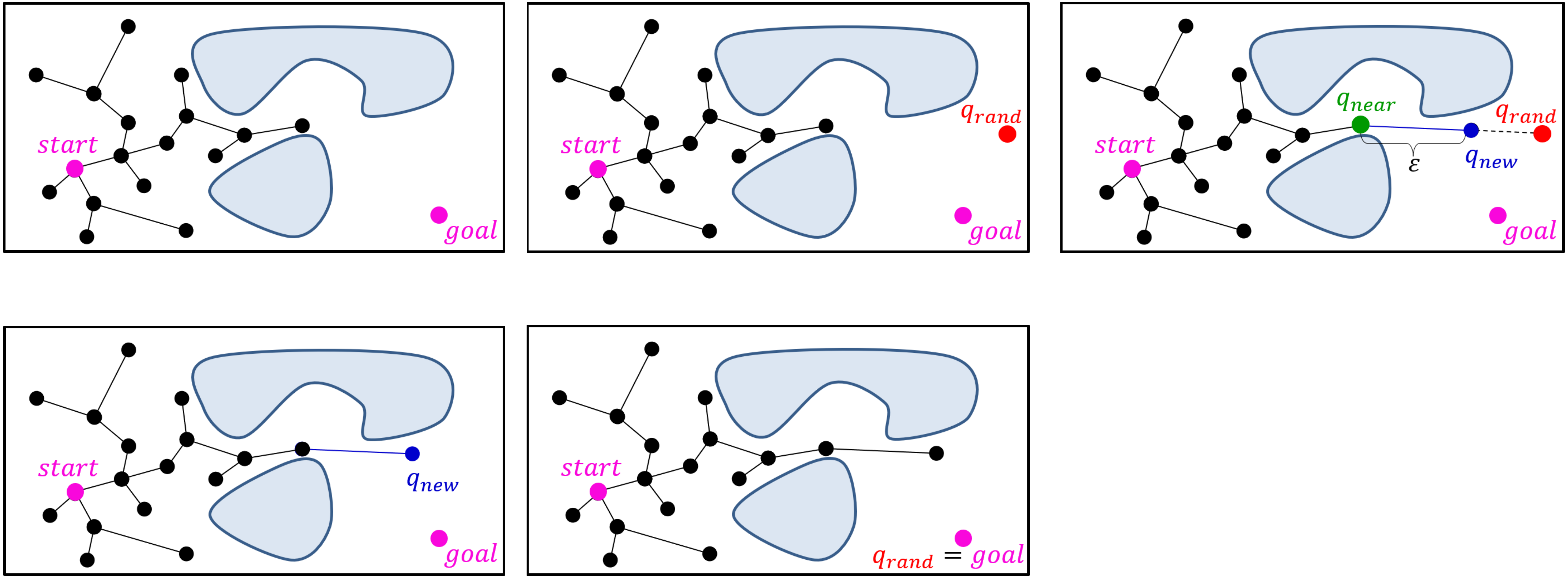
Expansion strategy towards q_{rand} led by a local planner (e.g., interpolation) starting from the nearest node (q_{near}).

maximum extension of the new edge (added by the local planner).

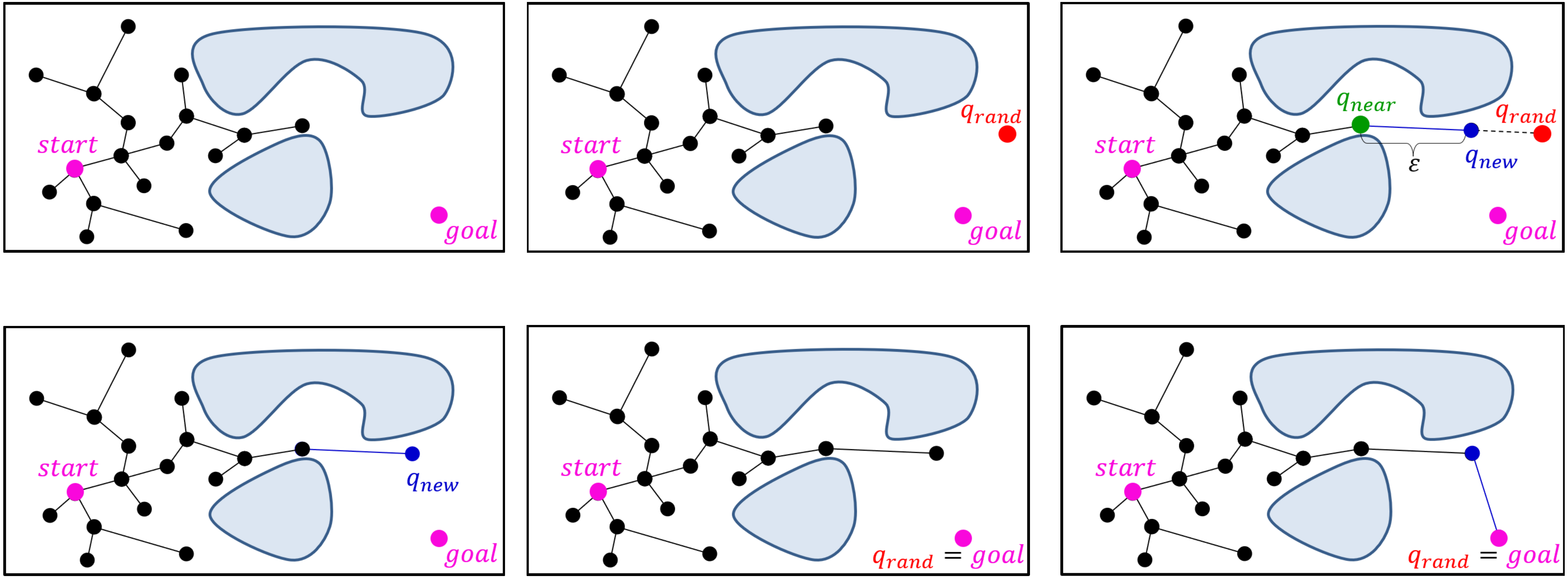
Rapidly-exploring Random Tree (RRT)



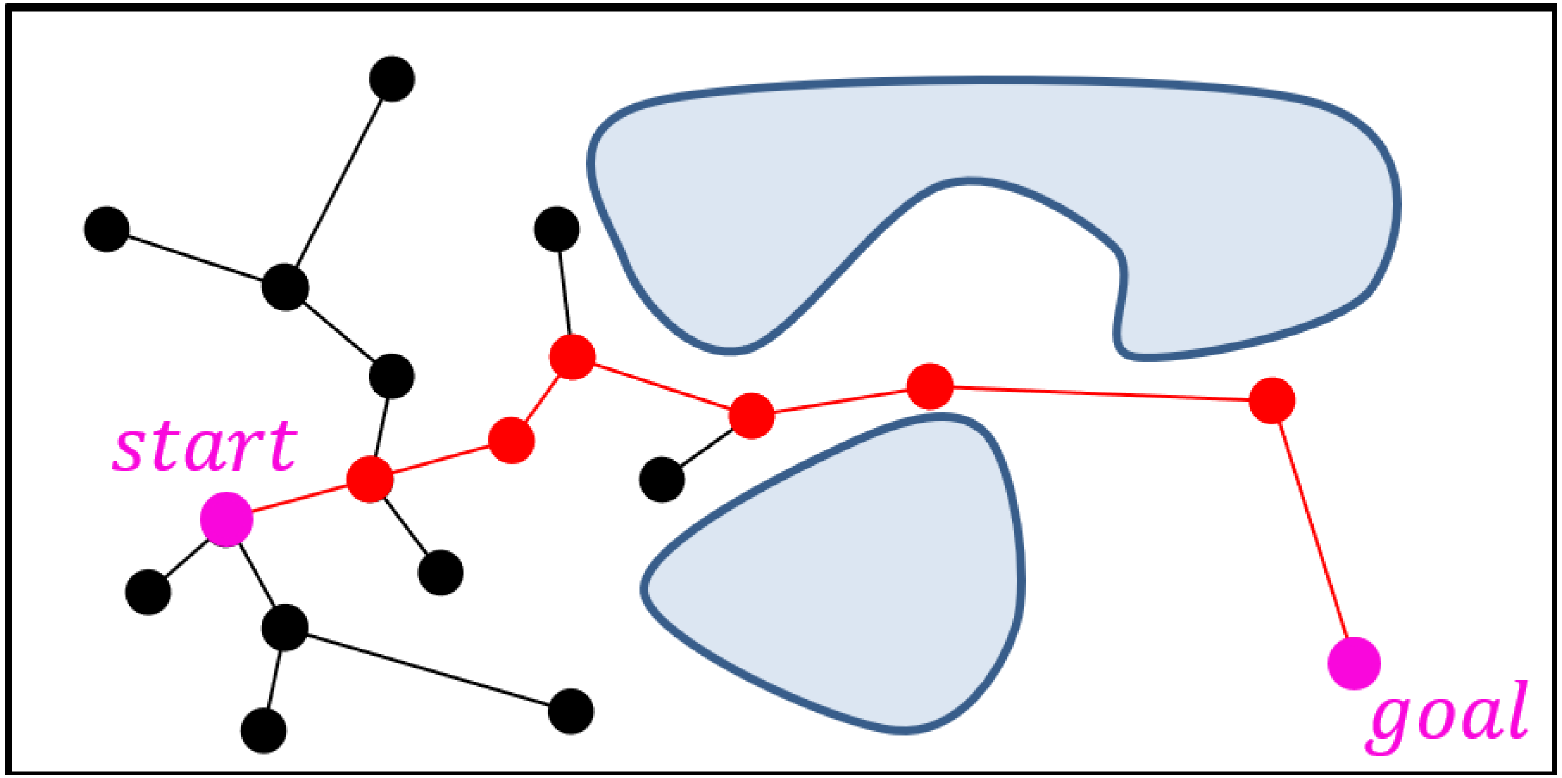
Rapidly-exploring Random Tree (RRT)



Rapidly-exploring Random Tree (RRT)



Rapidly-exploring Random Tree (RRT)

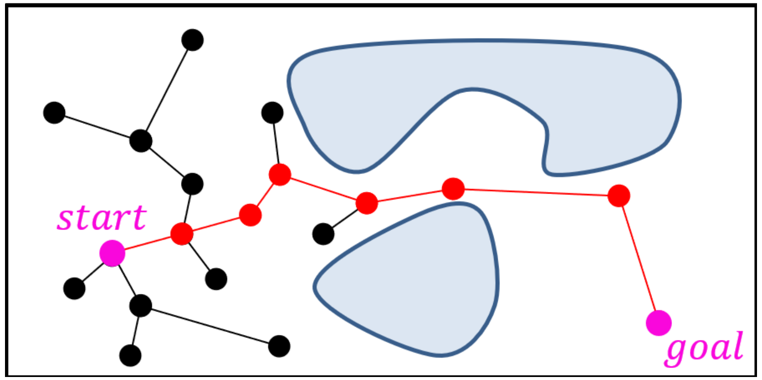


Rapidly-exploring Random Tree (RRT)

- A random sample $\mathbf{q}_{\mathbf{rand}}$ of the C-space is selected using a uniform probability distribution

- The configuration $\mathbf{q}_{\mathbf{near}}$ in the tree **T** (which is progressively formed) which is the closed one to $\mathbf{q}_{\mathbf{rand}}$ is found

- A new candidate configuration $\mathbf{q}_{\mathbf{new}}$ is produced on the segment joining $\mathbf{q}_{\mathbf{near}}$ to $\mathbf{q}_{\mathbf{rand}}$ at a predefined distance ϵ from $\mathbf{q}_{\mathbf{near}}$



- Check that $\mathbf{q}_{\mathbf{new}}$ and the segment $\mathbf{q}_{\mathbf{near}} - \mathbf{q}_{\mathbf{new}}$ belong to C_{free}

- If True => **T** is expanded by incorporating $\mathbf{q}_{\mathbf{new}}$ and the segment
- Otherwise, the configuration is discared

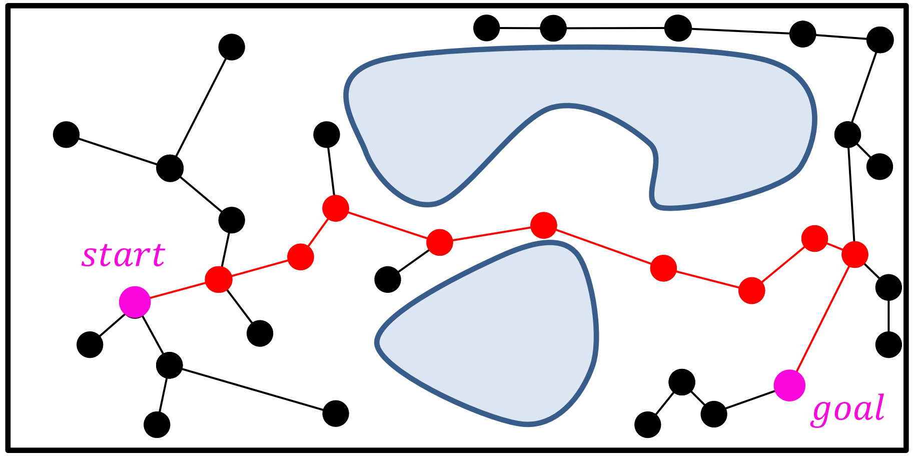
Rapidly-exploring Random Tree Connect
RRT-Connect

Rapidly-exploring Random Tree Connect (RRT-Connect)

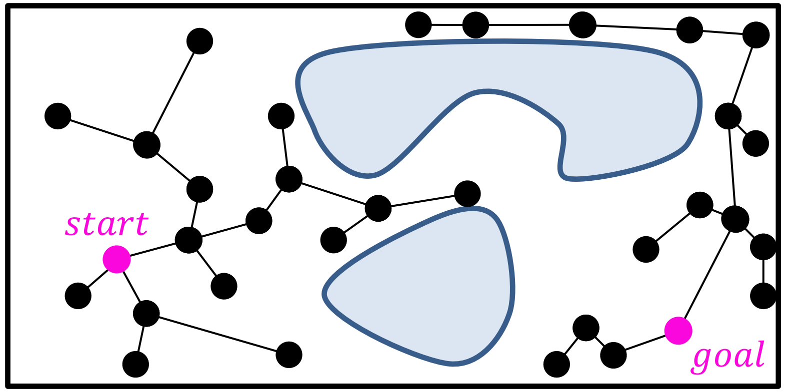
- Two trees are built: one from *Start* and one from *Goal*;

- Trees are alternately expanded;

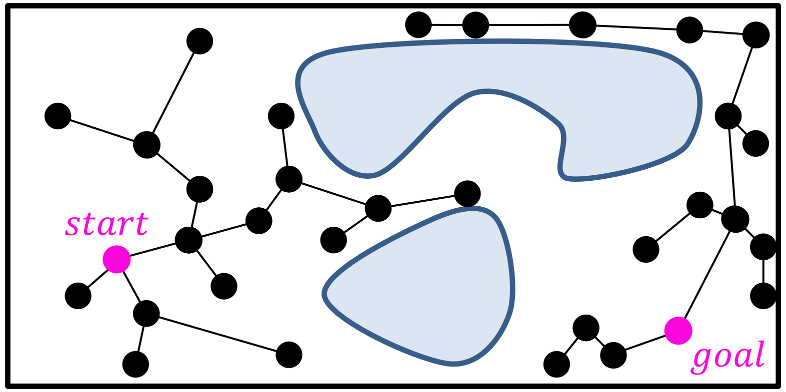
- After the expansion of a tree, we expand the other tree with the aim of creating a single connected component.

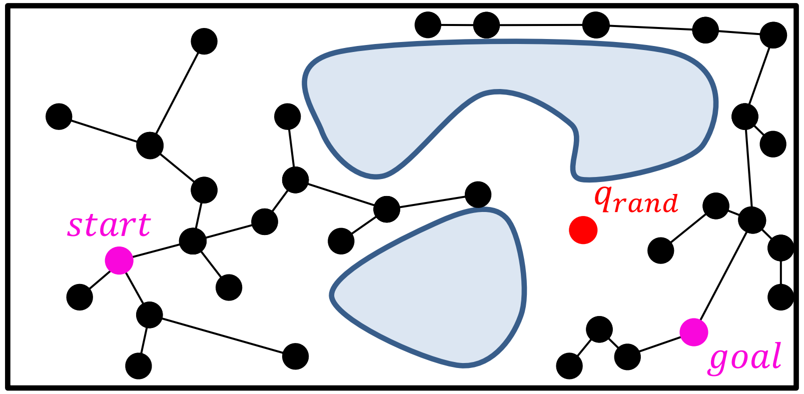


Rapidly-exploring Random Tree Connect (RRT-Connect)

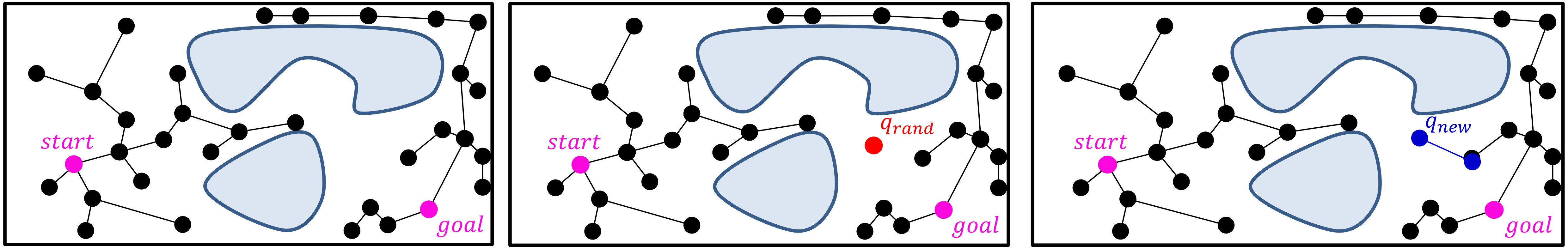


Rapidly-exploring Random Tree Connect (RRT-Connect)

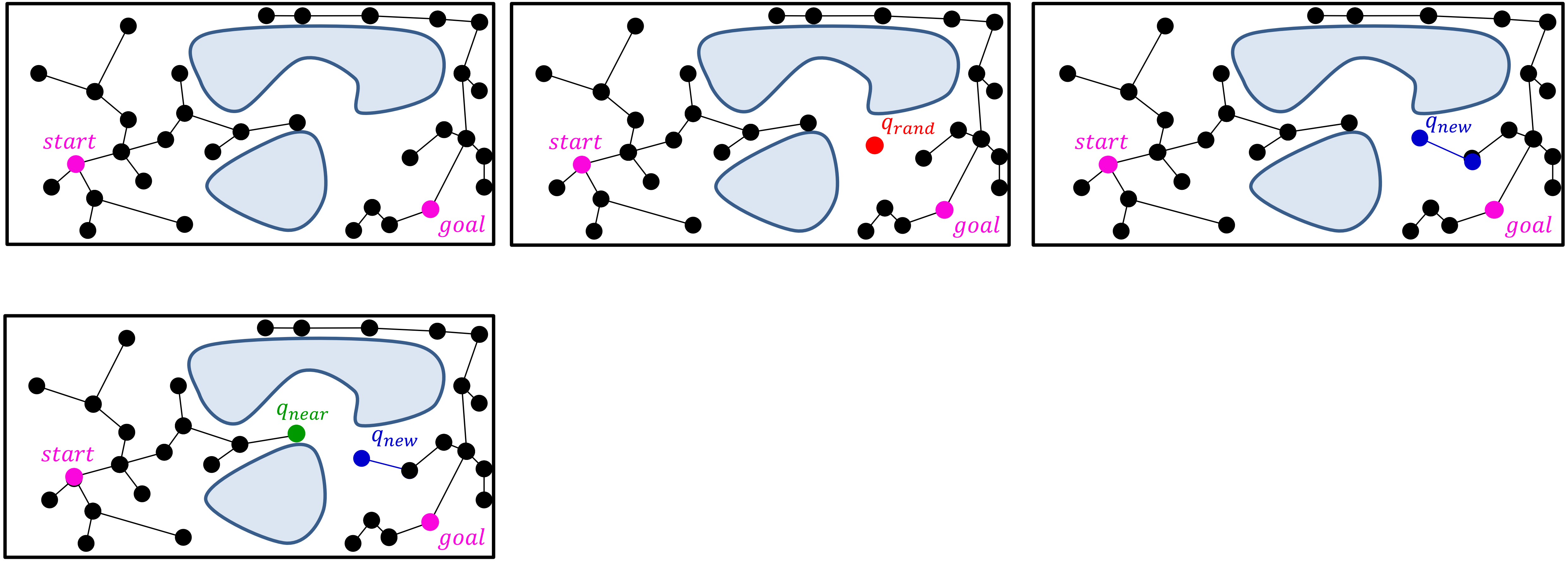




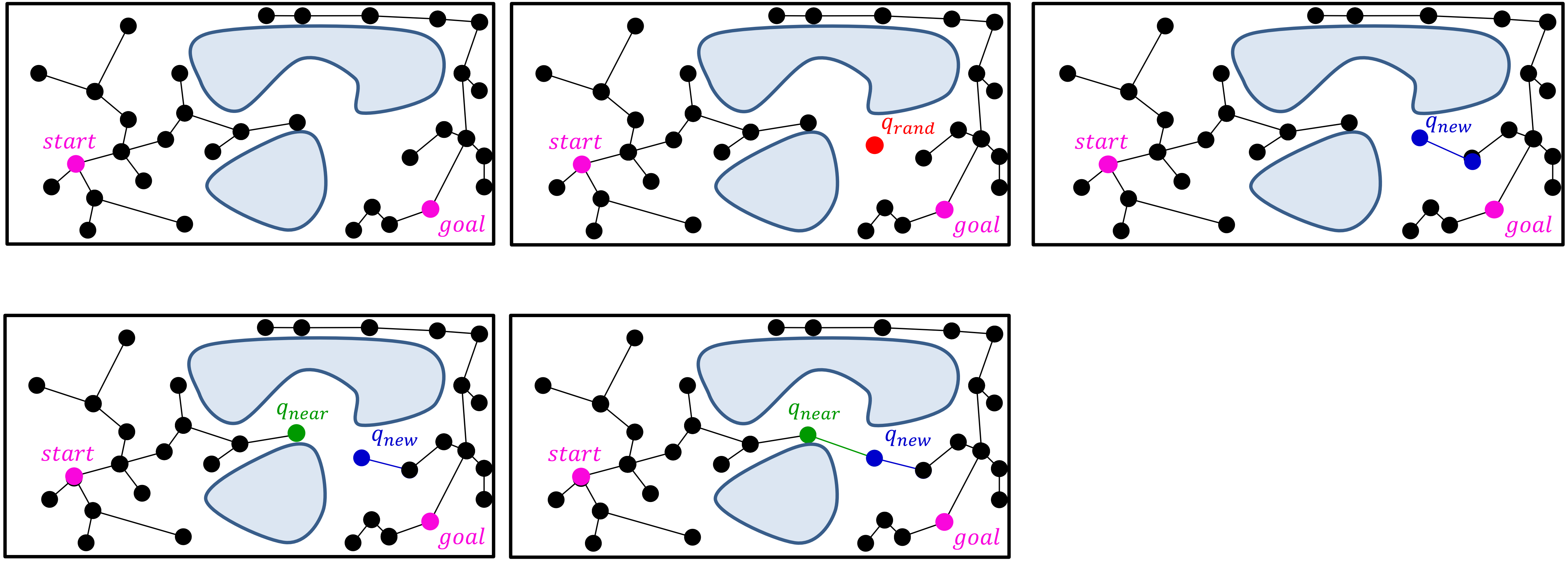
Rapidly-exploring Random Tree Connect (RRT-Connect)



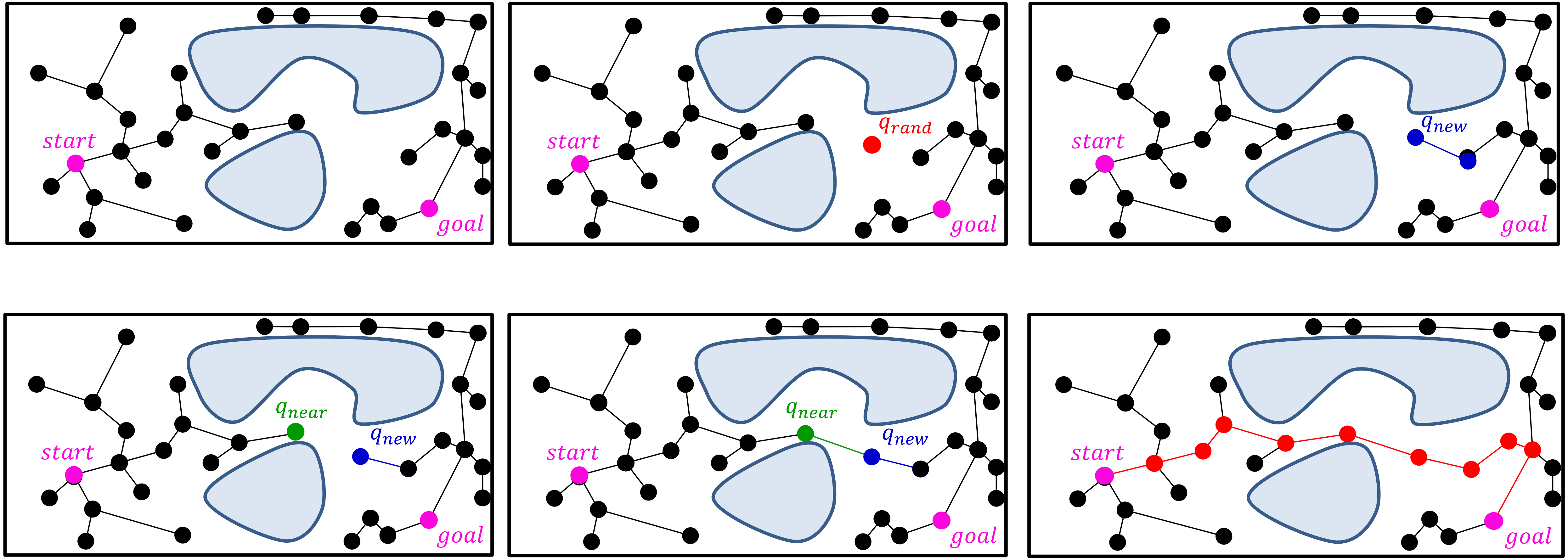
Rapidly-exploring Random Tree Connect (RRT-Connect)



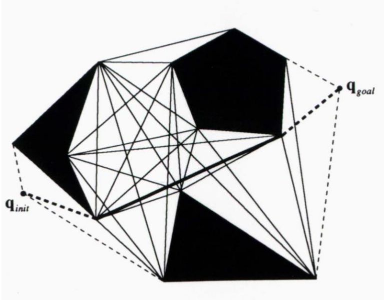
Rapidly-exploring Random Tree Connect (RRT-Connect)



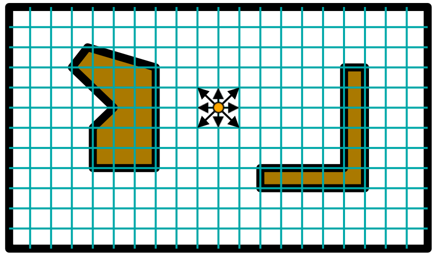
Rapidly-exploring Random Tree Connect (RRT-Connect)

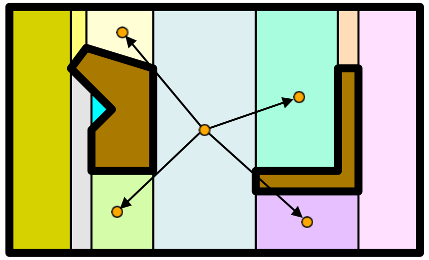


Combinatorial Planning

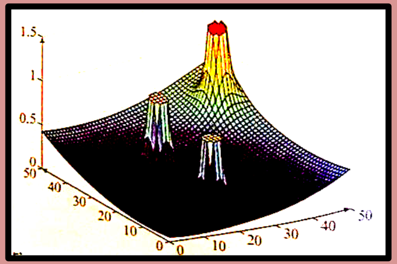


Sampling-based Planning

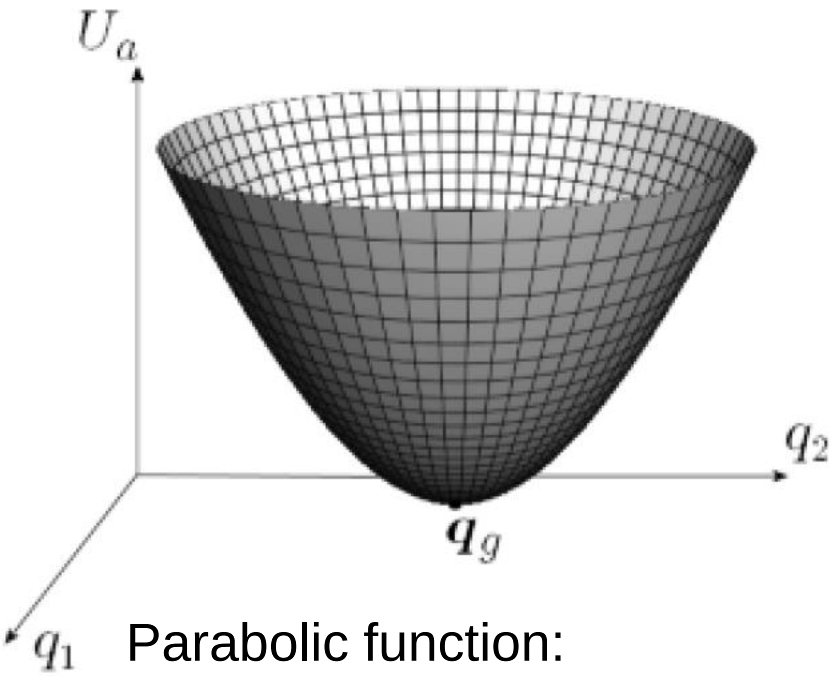




Artificial Potential Fields

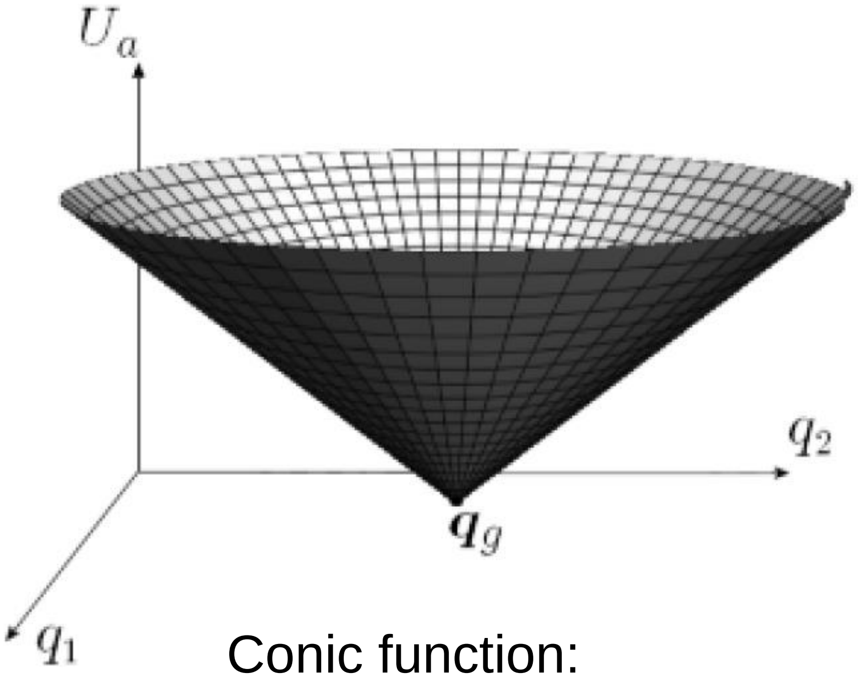


Artificial Potential Fields: Attractive Functions



Parabolic function:

$$U_a(\boldsymbol{q}) = \frac{1}{2}k_a\|\boldsymbol{e}(\boldsymbol{q})\|^2$$



Conic function:

$$U_a(\boldsymbol{q}) = k_a\|\boldsymbol{e}(\boldsymbol{q})\|$$

Error: $\boldsymbol{e}(\boldsymbol{q}) = \boldsymbol{q}_g - \boldsymbol{q}$

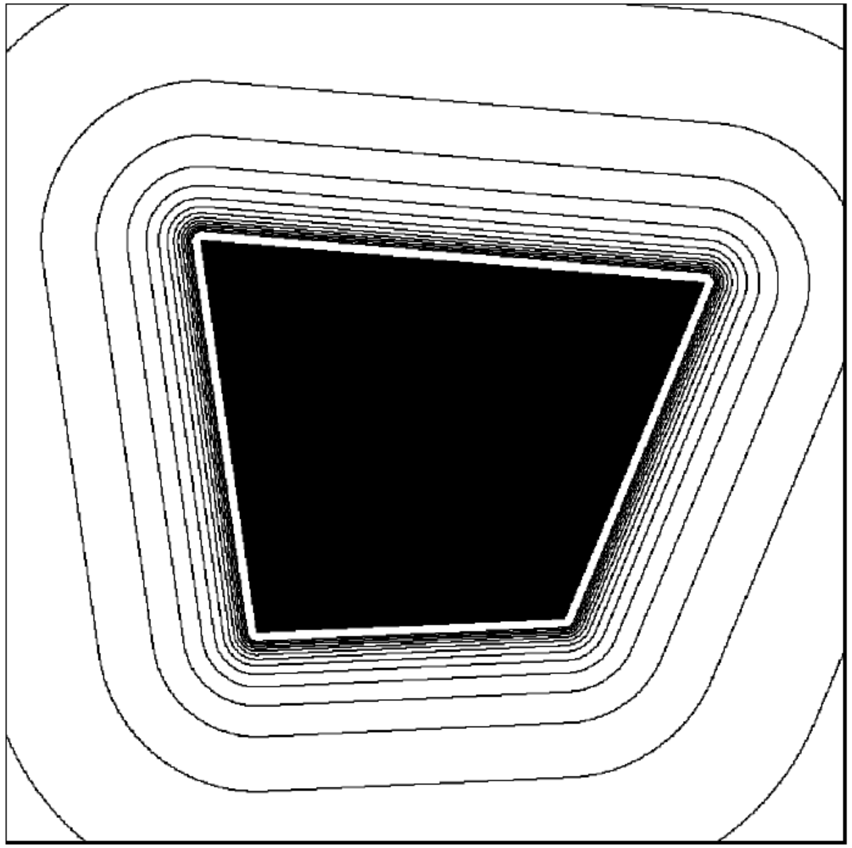
$K_a > 0$

Source: Siciliano B., Sciavicco L. , Villani L. , Oriolo G. , Robotics: modelling, planning and control, Springer Science & Business Media, 2010

Artificial Potential Fields: Repulsive Function

Distance: $\eta_i(\boldsymbol{q}) = min_{q' \in CO_i} \|\boldsymbol{q} - \boldsymbol{q}'\|$

$$U_{r,i}(\boldsymbol{q}) = \begin{cases} \frac{k_{r,i}}{\gamma}\left(\frac{1}{\eta_i(\boldsymbol{q})} - \frac{1}{\eta_{0,i}}\right), & \eta_i(\boldsymbol{q}) \leq \eta_{0,i} \\ 0, & \eta_i(\boldsymbol{q}) \geq \eta_{0,i} \end{cases}$$



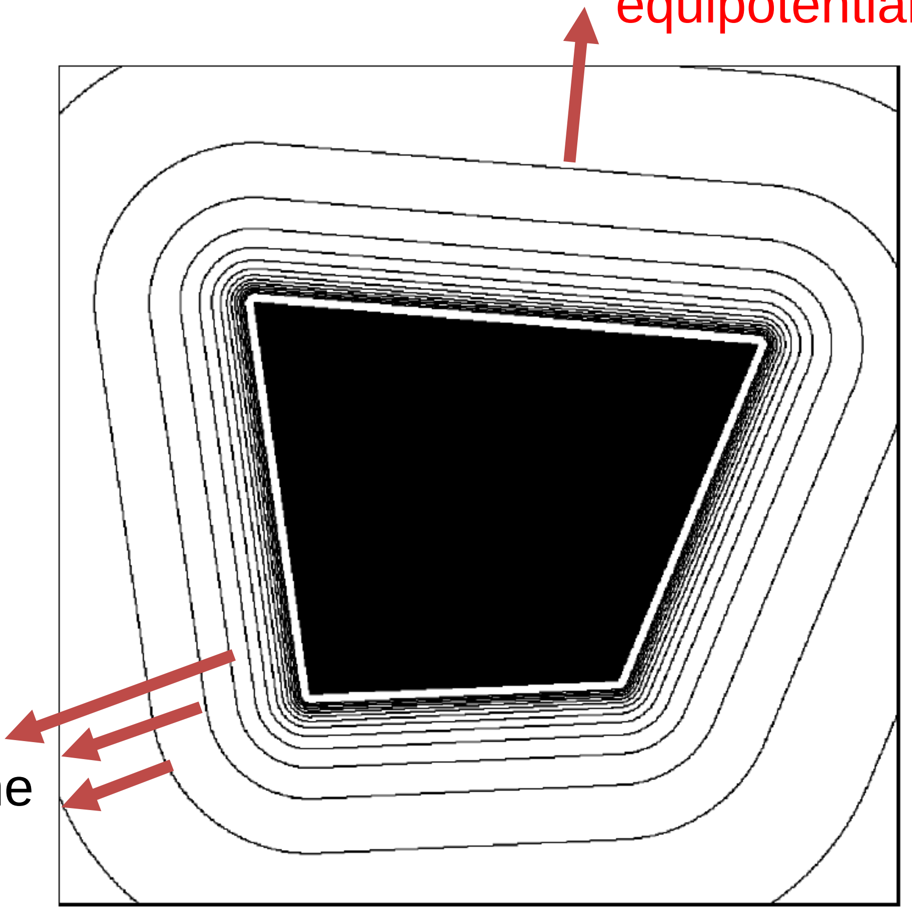
Source: Siciliano B., Sciavicco L. , Villani L. , Oriolo G. , Robotics: modelling, planning and control, Springer Science & Business Media, 2010

Artificial Potential Fields: Repulsive Function

Distance: $\eta_i(\boldsymbol{q}) = min_{q' \in CO_i} \|\boldsymbol{q} - \boldsymbol{q}'\|$

$$U_{r,i}(\boldsymbol{q}) = \begin{cases} \frac{k_{r,i}}{\gamma}\left(\frac{1}{\eta_i(\boldsymbol{q})} - \frac{1}{\eta_{0,i}}\right), & \eta_i(\boldsymbol{q}) \le \eta_{0,i} \\ 0, & \eta_i(\boldsymbol{q}) \ge \eta_{0,i} \end{cases}$$

Forces are orthogonal to equipotential contours



Forces increase approaching the boundary of CO_i

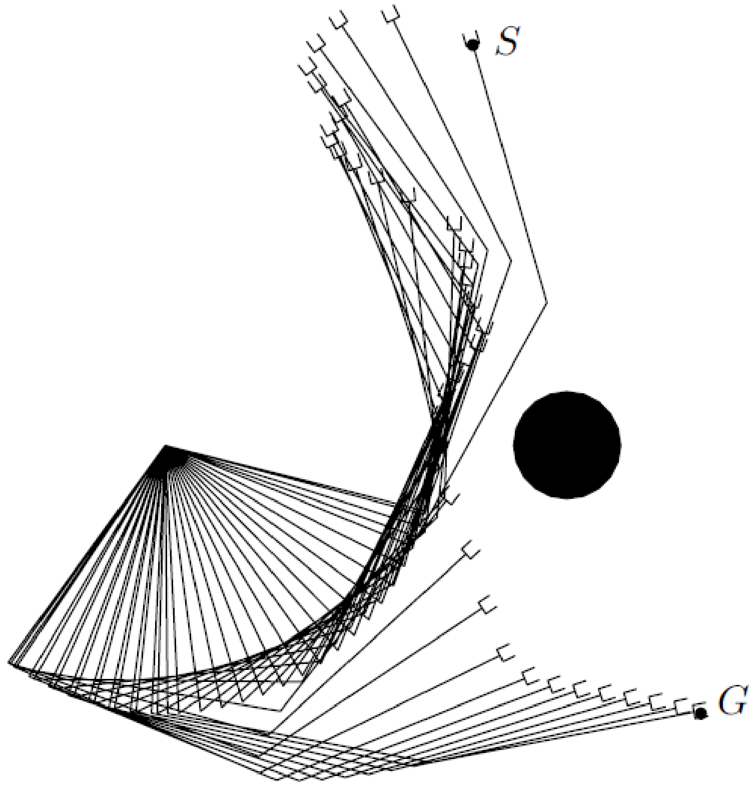
Source: Siciliano B., Sciavicco L. , Villani L. , Oriolo G. , Robotics: modelling, planning and control, Springer Science & Business Media, 2010

Artificial Potential Fields: Total Potential

Total Repulsive: $U_r(\boldsymbol{q}) = \sum_{i=1}^{p} U_{r,i}(\boldsymbol{q})$

Total Potential: $U_t(\boldsymbol{q}) = U_a(\boldsymbol{q}) + U_r(\boldsymbol{q})$

Total Force: $\mathbf{f}_{\mathrm{t}}(\boldsymbol{q}) = -\nabla U_t(\boldsymbol{q}) = \boldsymbol{f}_a(\boldsymbol{q}) + \sum_{i=1}^{p} \boldsymbol{f}_{r,i}(\boldsymbol{q})$



Source: Siciliano B., Sciavicco L. , Villani L. , Oriolo G. , Robotics: modelling, planning and control, Springer Science & Business Media, 2010

Three techniques to plan the robot motion:

1. Consider $\mathbf{f}_t$ as a generalized forces: $\boldsymbol{\tau} = \mathbf{f}_t(\mathbf{q})$

2. Consider $\mathbf{f}_t$ as a generalized accelerations: $\ddot{\boldsymbol{q}} = \mathbf{f}_t(\mathbf{q})$

3. Consider $\mathbf{f}_t$ as a generalized velocities: $\dot{\boldsymbol{q}} = \mathbf{f}_t(\mathbf{q})$

Three techniques to plan the robot motion:

1. Consider $\mathbf{f}_t$ as a generalized forces: $\boldsymbol{\tau} = \mathbf{f}_t(\mathbf{q})$
2. Consider $\mathbf{f}_t$ as a generalized accelerations: $\ddot{\boldsymbol{q}} = \mathbf{f}_t(\mathbf{q})$
3. Consider $\mathbf{f}_t$ as a generalized velocities: $\dot{\boldsymbol{q}} = \mathbf{f}_t(\mathbf{q})$

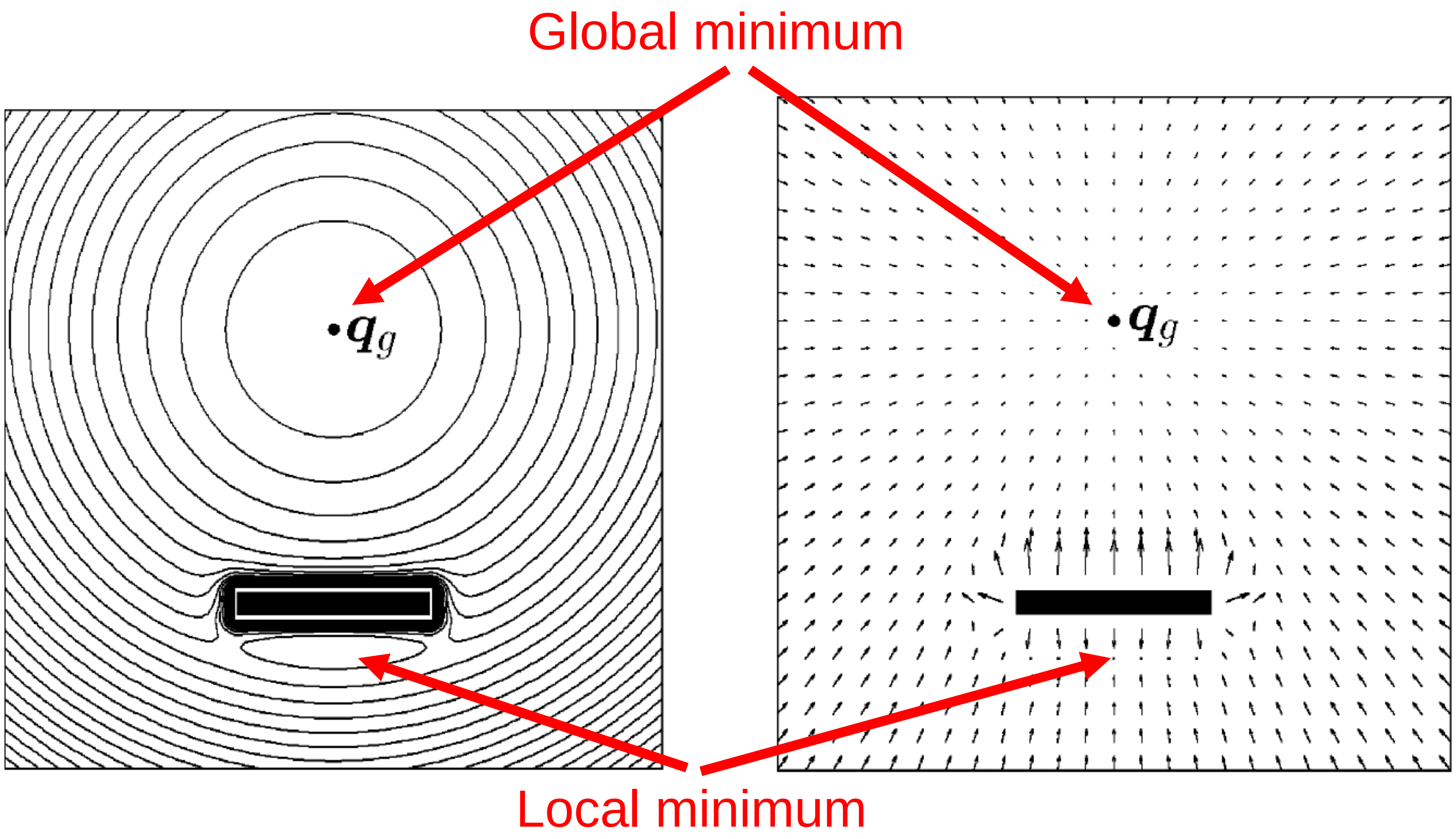
Offline planning

- The path is C are generated by numerical integration (Euler method)
  $\mathbf{q}_{k+1} = \mathbf{q}_k + T\mathbf{f}_t(\mathbf{q}_k)$

Online Planning

- Directly provides control inputs or reference velocities for low-level control

Artificial Potential Fields: Local Minima Problem



Source: Siciliano B., Sciavicco L. , Villani L. , Oriolo G. , Robotics: modelling, planning and control, Springer Science & Business Media, 2010

Artificial Potential Fields: Local Minima Problem

Local minima: $\mathbf{f}_{\mathrm{t}}(\boldsymbol{q}) = 0$

The Motion Planner based on APF is **not complete**, i.e. the path may not reach $\mathbf{q}_{\mathrm{g}}$ even if a solution exists.

Workarounds exists but keep in mind that APF are mainly used for **online** motion planning, where completeness may not be required

APF Local Minima Problem: Workaround 1

Best-first algorithm

- Discretization of C_{free} using a regular grid and associate to each free cell the value U_t at its centroid

- Build a tree T rooted at $\mathbf{q}_s$ and at each iteration: 1) Select the leaf with minimum value of U_t ; 2) Add as children its adjacent free cells that are not in T

- Planning stops when $\mathbf{q}_g$ is reached **(success)** or no further cells can be added to T (**failure**)

- Build a solution path by tracing back from $\mathbf{q}_g$ to $\mathbf{q}_s$

- Complexity is **esponential** in the C-dimension => applicable in low-dimensional spaces

APF Local Minima Problem: Workaround 2

Navigation function

- Build navigation functions, i.e. potentials without local minima

- If the C-obs are **star-shaped**, you can map *CO* to a collection of sphere via a **diffeomorphism** and build a potential in transformed space and map it back to C

- Alternative: define the potential as an **harmonic function** (solution of Laplace's equation)

- Both techniques require a **complete knowledge** of the environment. Hence the interest is more in theory than in practice

APF Local Minima Problem: Workaround 2

Numerical navigation function

- Applicable when C-space has low-dimensionality

- Represent C_{free} as a gridmap
- **Wavefront expansion:** assign potential 0 to the cell that contains the goal, 1 to cell adjacent to the 0-cell, 2 to cell adjacent to the 1-cell, etc.

APF Local Minima Problem: Workaround 2

Goal

2	1	2	3	4	5	6	7	8	9		19
1	0	1			6	7	8	9	10		18
2	1	2	3		7	8		10	11		17
3		3	4	5	6	7	8		12		16
4			5	6	7			12	13		15
5	6	7	6	7	8	9	10	11	12	13	14
6	7	8	7	8	9	10	11	12	13	14	15

Start

Solution path
steepest descent from the start

Open Motion Planning Library
OMPL

OMPL

- **OMPL**: **O**pen **M**otion **P**lanning **L**ibrary [3]

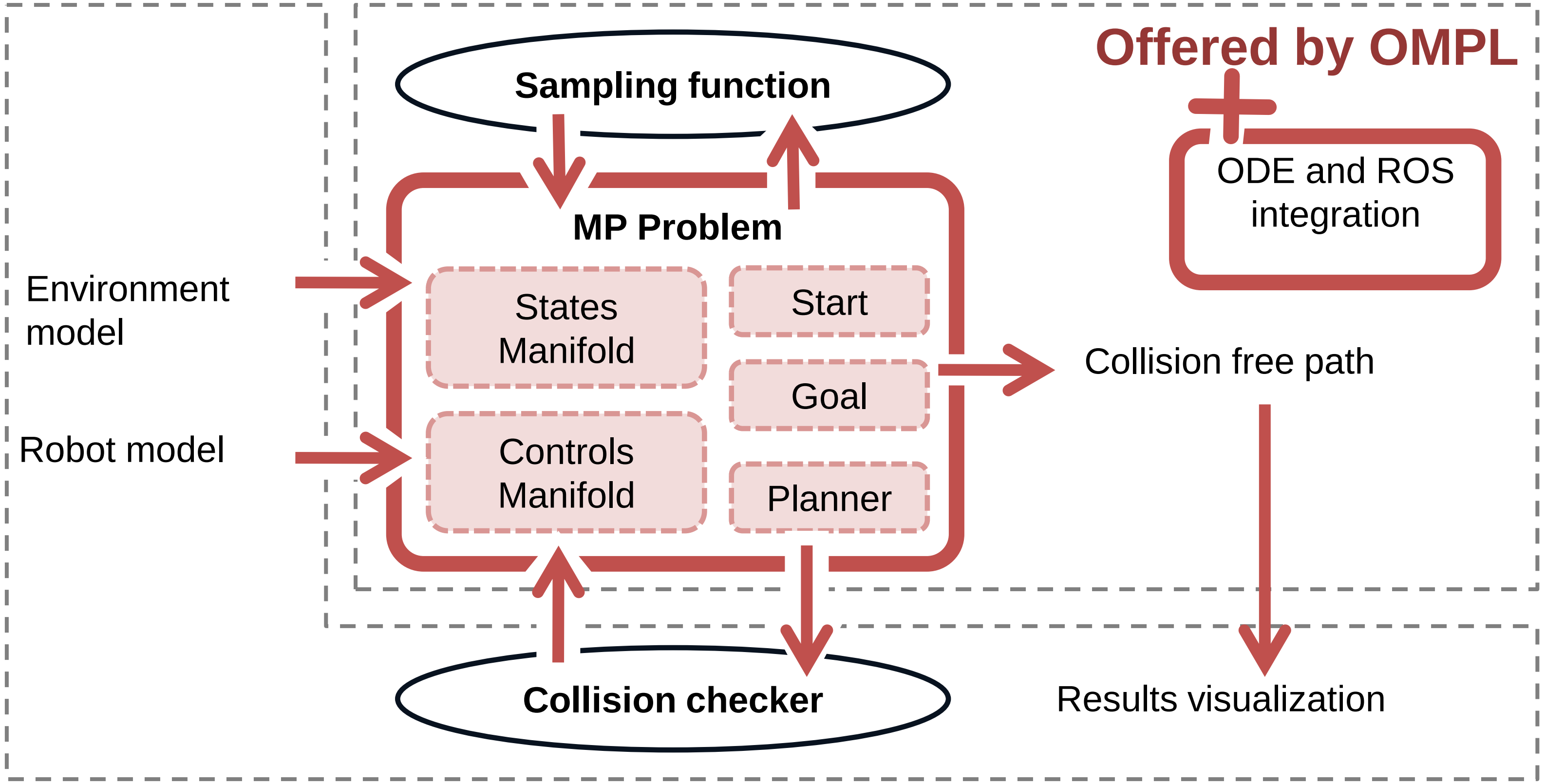
- Motion planning library that implements **Probabilistic sampling-based** Motion Planning algorithms

- Developed by Kavraki lab at Rice University (Houston, Texas)

- Link: https://ompl.kavrakilab.org/index.html

[3] Ioan A. Şucan, Mark Moll, Lydia Kavraki (December 2011). *The Open Motion Planning Library*. IEEE Robotics & Automation Magazine.

OMPL: Structure



UNIVERSITÀ DEGLI STUDI DI PADOVA

Thanks for the attention