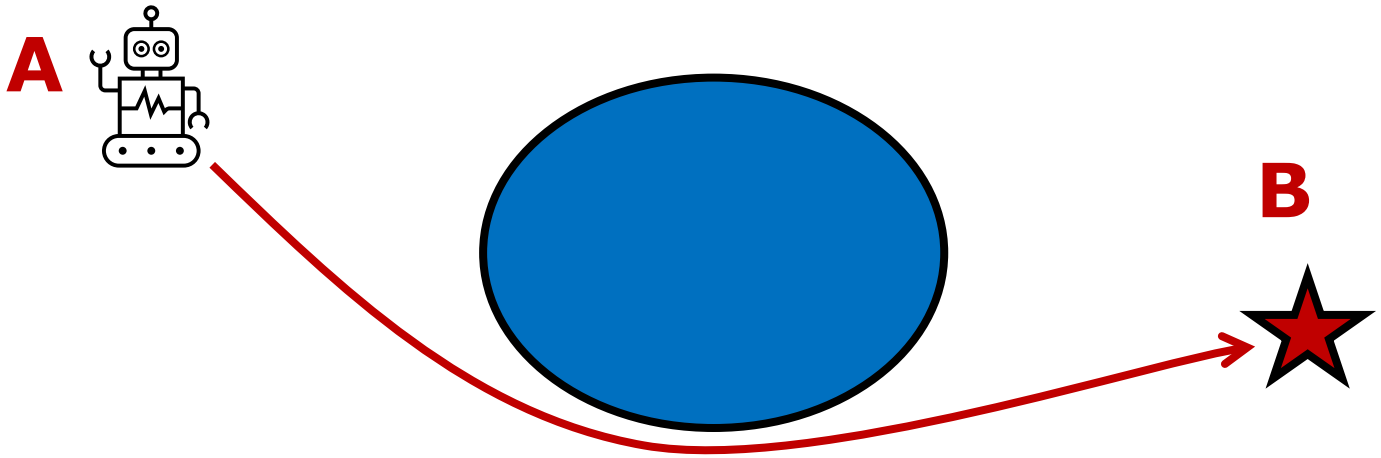
5 – Sampling-based Methods

Dr. Marija Popović

UNIVERSITÄT BONN

Review: What is Planning?

- Find a sequence of valid configurations to move a robot from point A to point B – *how*?
- **Classical path planning**: What is the shortest geometric path?



Review: Planning Methods

- Geometric
- Potential field
- Search-based
- Sampling-based
- Trajectory
- Bioinspired

Review: Planning Methods

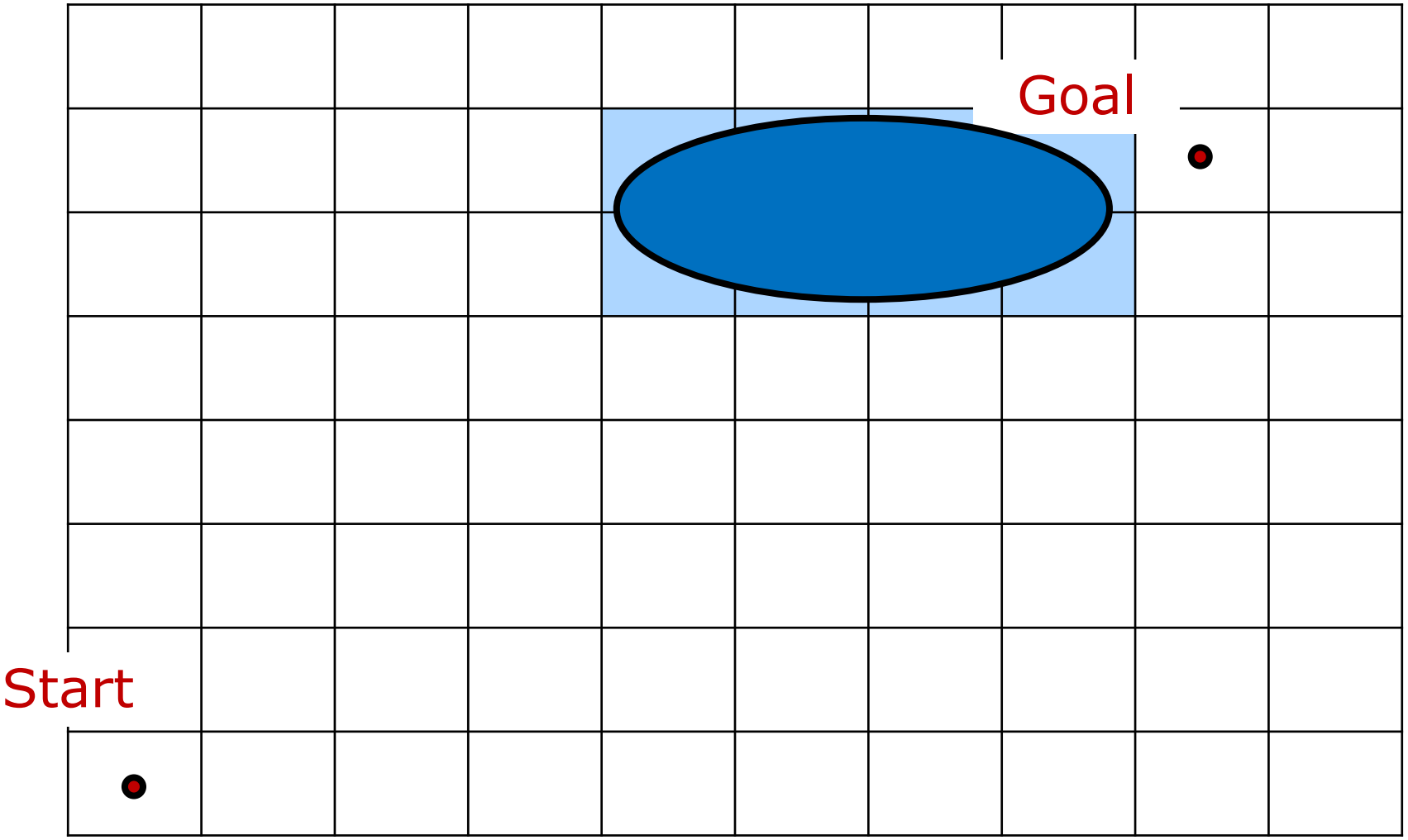
- Geometric
- Potential field
- Search-based
- Sampling-based
- Trajectory
- Bioinspired

Problem Statement

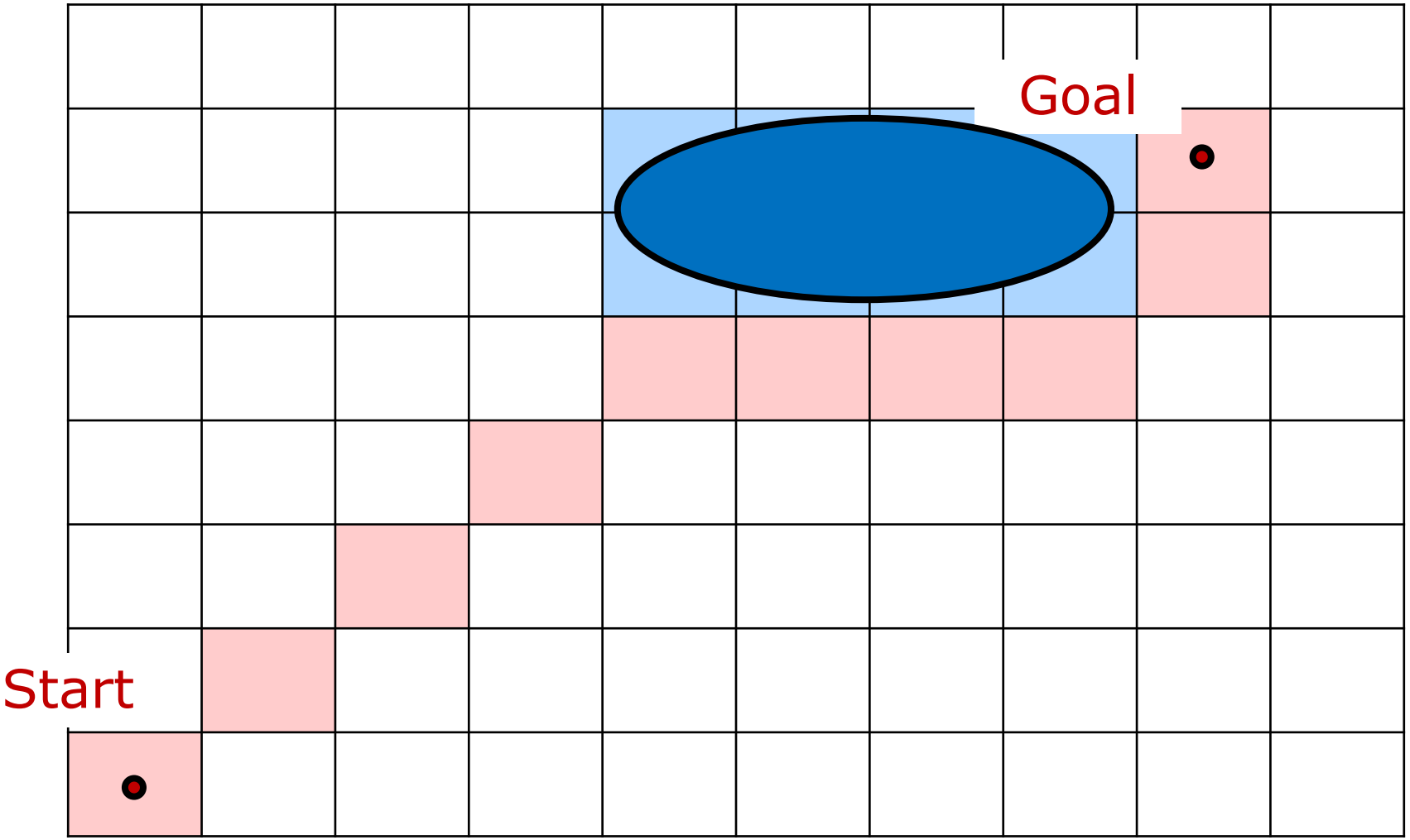
- **Given**:
  - Known environment
  - Robot model
  - Start and goal configurations
- Find a sequence of configurations to move the robot from start to goal
- Exclude uncertainty first

- **Sampling-based**: Sample to capture the connectivity of the free space

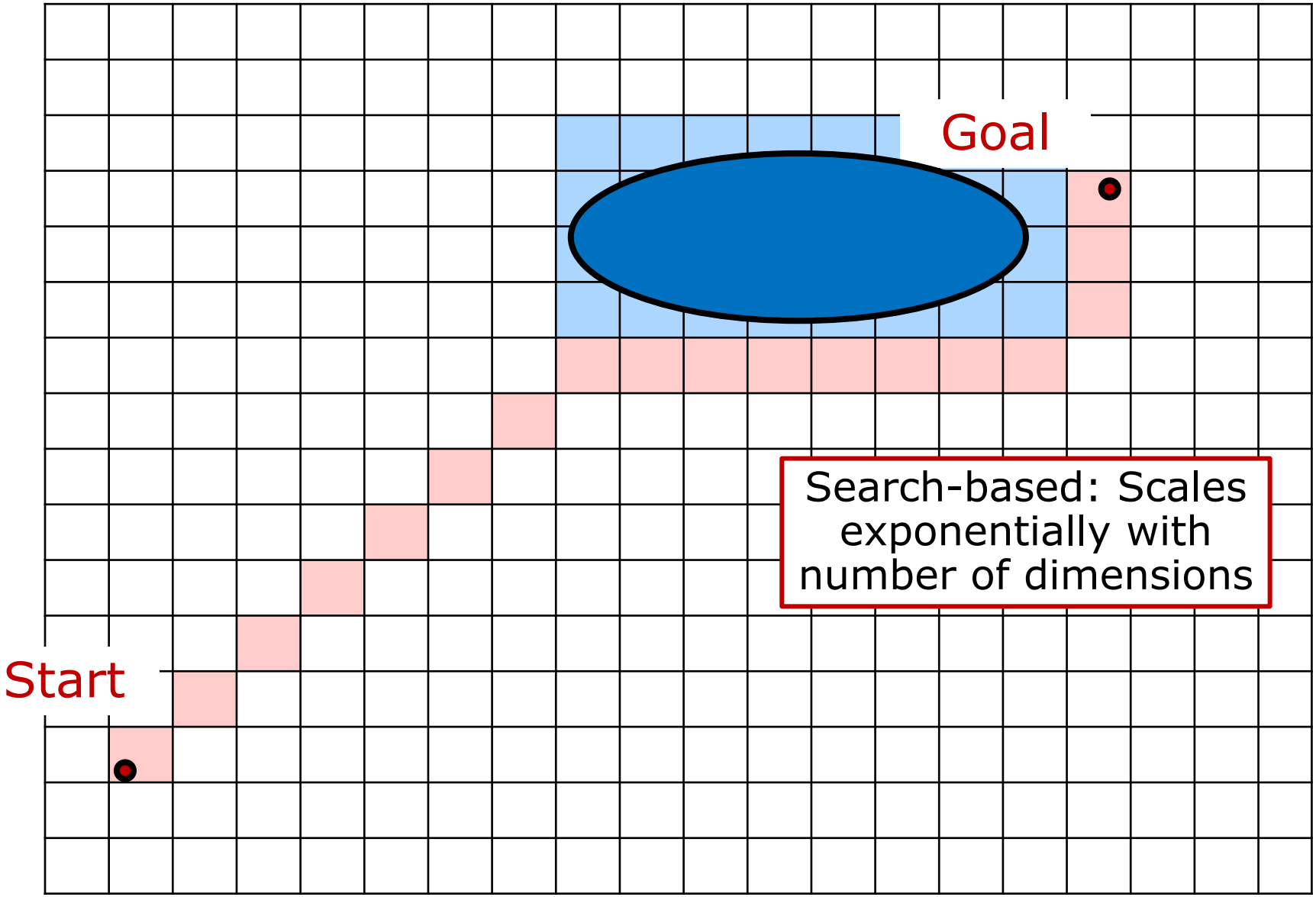
Motivation – Why Sample?



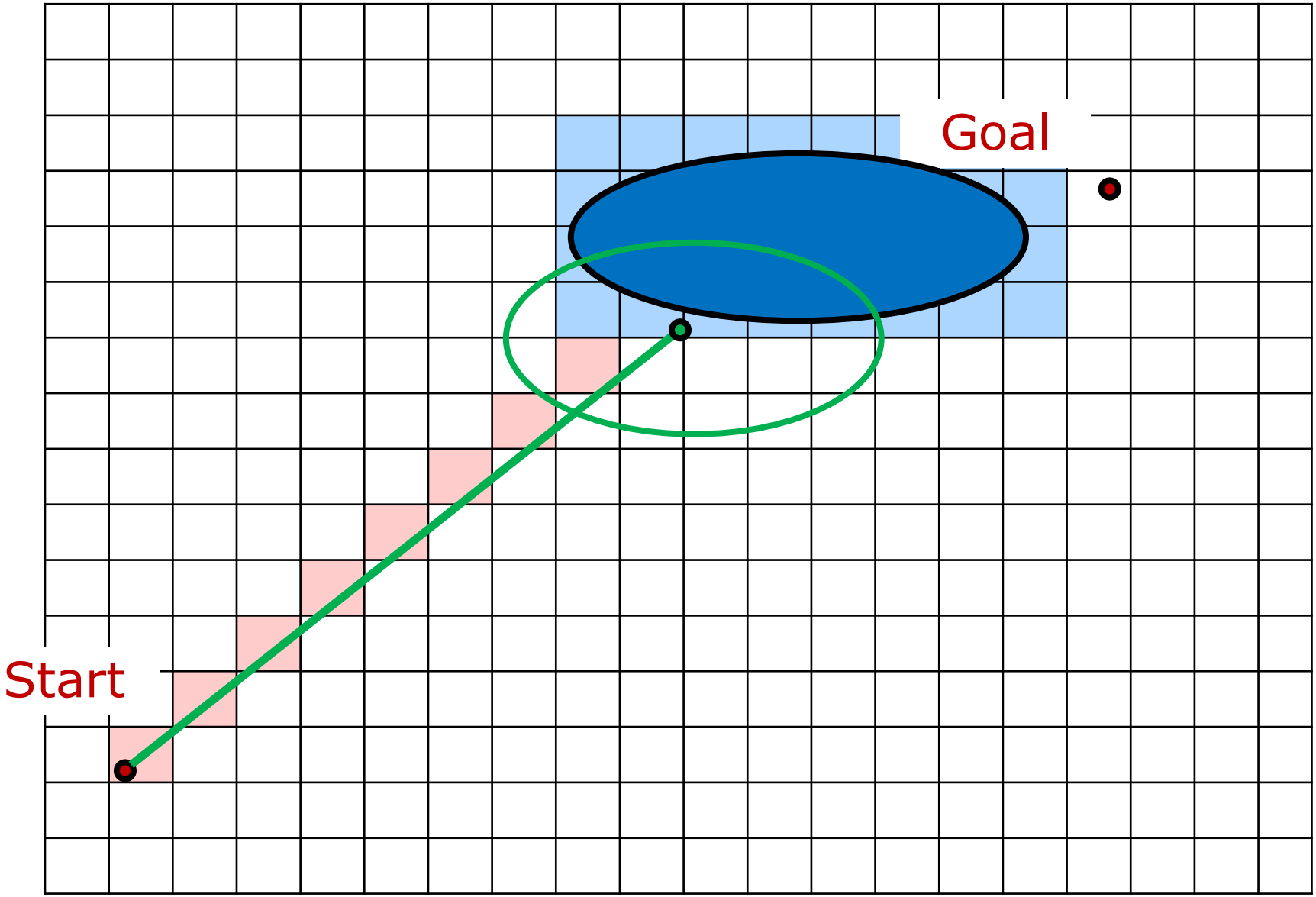
Motivation – Why Sample?



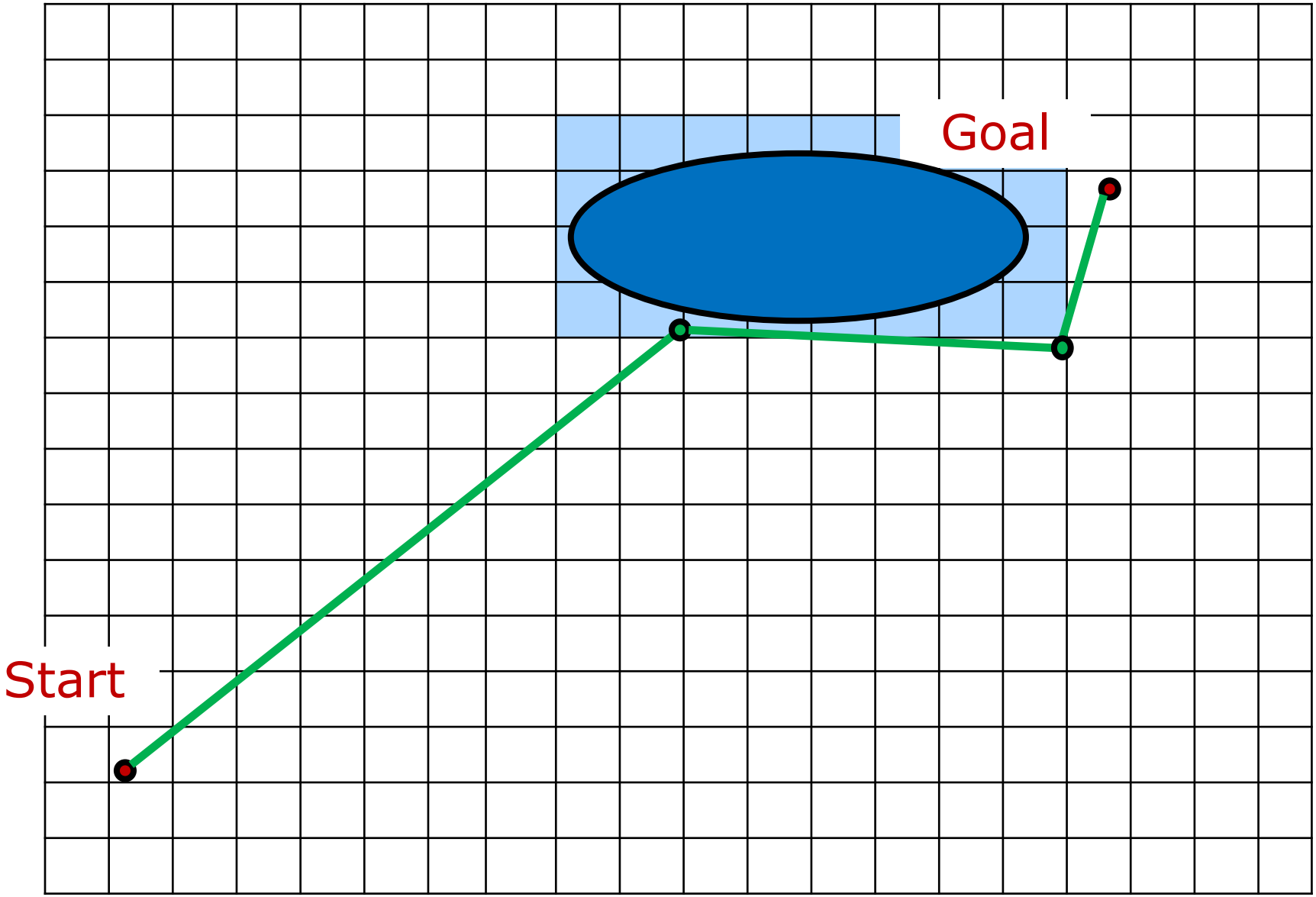
Motivation – Why Sample?



Motivation – Why Sample?



Motivation – Why Sample?

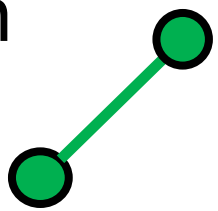


General Approach

1. Randomly sample configurations (nodes) in free space
2. Connect them to a roadmap (graph) to find a path of valid segments (edges)

- Sacrifice optimality/completeness for efficiency

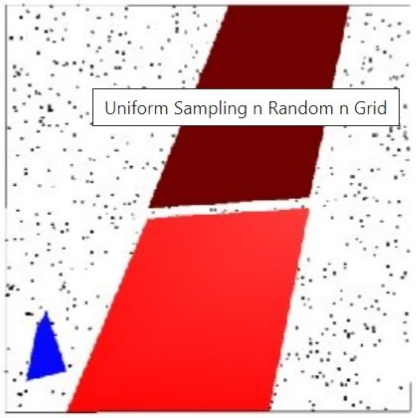
1. How to **sample**?
2. How to **collision-check** between samples?



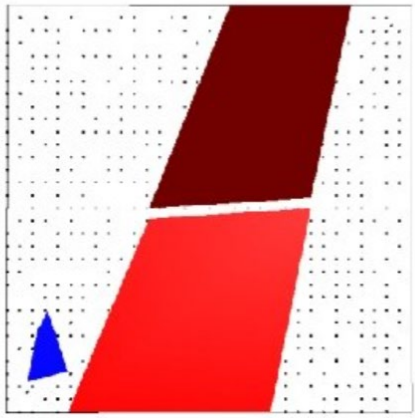
Sampling

- Most commonly used: **uniform random sampling**
- More advanced sampling techniques:

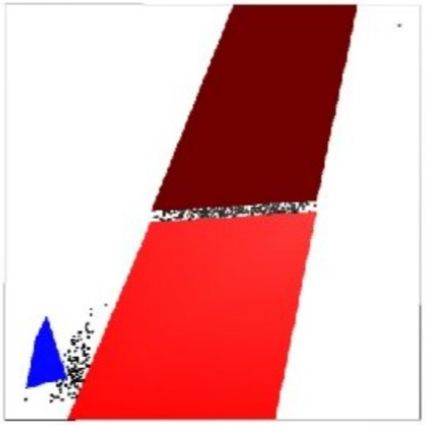
- Random



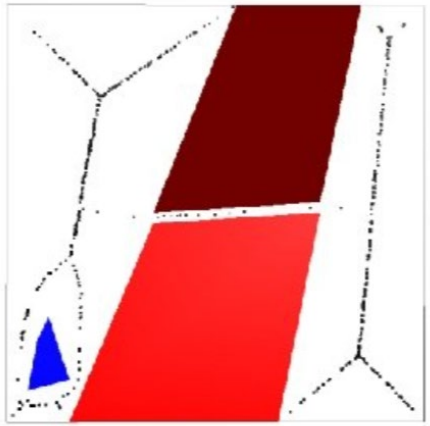
- Grid



- Bridge test



- Medial axis



Collision Checking

- "Black box" feasibility check

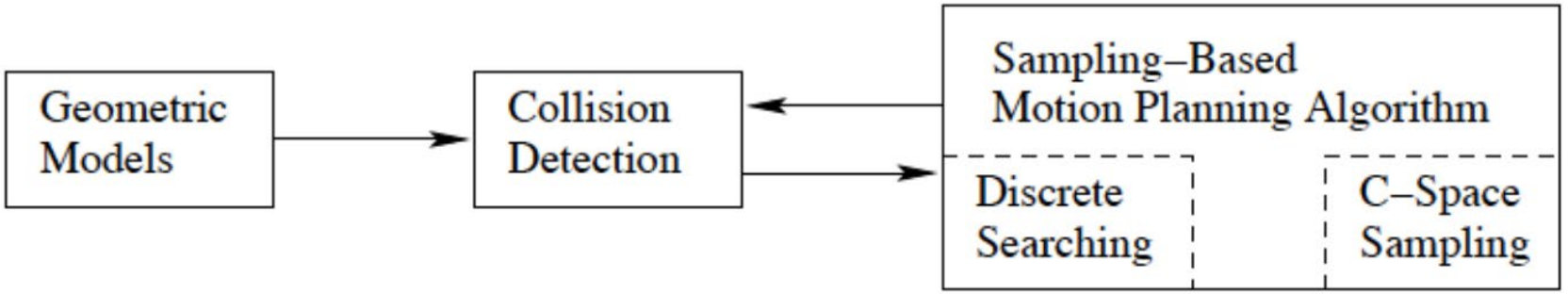
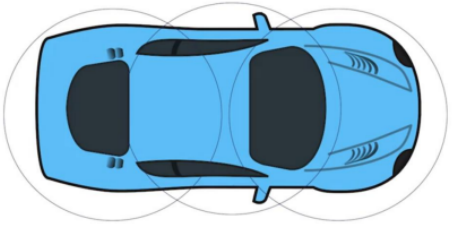


Figure 5.1: The sampling-based planning philosophy uses collision detection as a "black box" that separates the motion planning from the particular geometric and kinematic models. C-space sampling and discrete planning (i.e., searching) are performed.

LaValle, S. M., 2006. *Planning Algorithms*. Cambridge University Press.

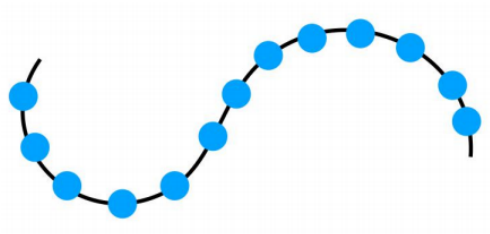
Collision Checking

- "Black box" feasibility check
- Explicit representation of obstacles
- Usually based on geometrical models



Collision checking between two objects:

- Approximate both with circles
- Check that distance between centers larger than radius of circles



Collision checking along a path:

- Sample points (i.e. poses) uniformly along the path (densely enough)
- Check collision for each pose

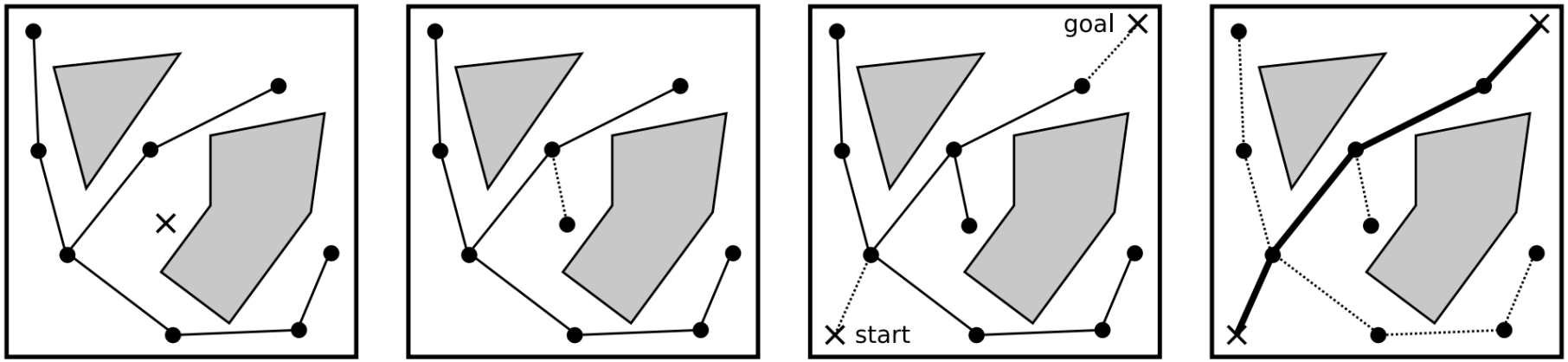
Criteria

- **Completeness**: The algorithm can always find a solution when a solution exists
  - Probabilistic completeness, rate of convergence
- **Optimality**: The solution is the best one of all possible solutions in terms of pre-defined cost
  - Asymptotic optimality
- **Complexity**: Time and space
  - Complexity as a function of the input
  - Complexity per sample

Multi-Query vs. Single-Query

- **Multi-query**: Roadmap-based, can solve many planning queries in the same environment.

(1) PRM Algorithm



(a) The *learning* phase: a random sample, denoted by ×, is generated

(b) A local planner is used to connect the new sample to nearby roadmap vertices.

(c) The *query* phase: the start and goal configurations are added to the roadmap.

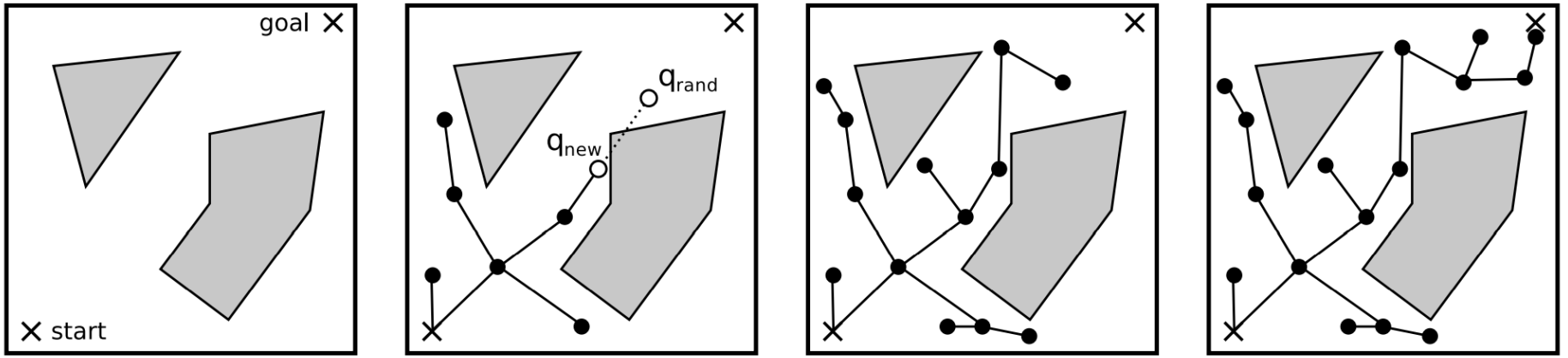
(d) A graph search algorithm is used to connect the start and goal through the roadmap.

Short, A. et al. *Recent progress on sampling based dynamic motion planning algorithms*. In: IEEE AIM.

Multi-Query vs. Single-Query

- **Single-query**: Incremental, build a roadmap/tree from the start configuration

(2) RRT Algorithm



(a) A tree is grown from the start configuration towards the goal.

(b) The planner generates a configuration q_{rand}, and grows from the nearest node towards it to create q_{new}.

(c) The tree rapidly explores the free space.

(d) The planner terminates when a node is close to the goal node. Common implementations will connect directly to the goal.

Short, A. et al. *Recent progress on sampling based dynamic motion planning algorithms*. In: IEEE AIM.

Multi-Query vs. Single-Query

Multi-query

```
GraphPlanner(q_start, q_goal)
    G.init(q_start, q_goal);
    while no path from q_start to q_goal do
        q_rand ← Sample();
        Q ← SelectNghbrs(G, q_rand)
        for all q_near ∈ Q do
            if Connect(q_near, q_rand) then
                G.Add(q_near, q_rand)
```

Single-query

```
TreePlanner(q_start, q_goal)
    T.init(q_start);
    while no path from q_start to q_goal do
        q_rand ← Sample();
        q_near ← Select(T, q_rand)
        q_new ← Extend(q_near, q_rand);
        if Connect(q_new, q_near) then
            T.Add(q_near, q_new)
```

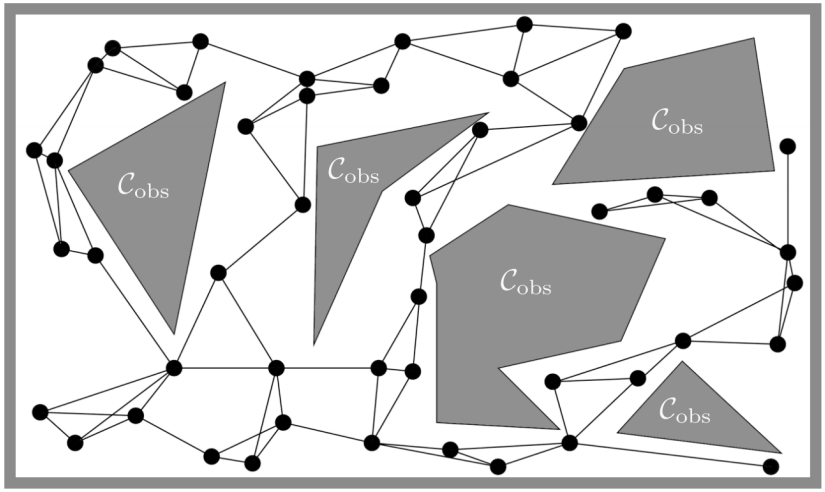
Probabilistic Roadmap (PRM)

PRM – Concept

- Two key steps:

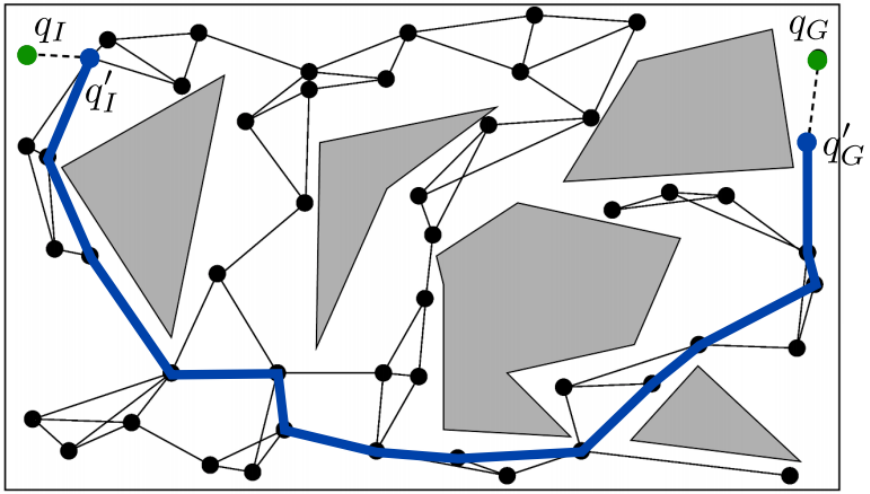
1. Learning phase

- Build a roadmap by sampling free space



2. Query phase

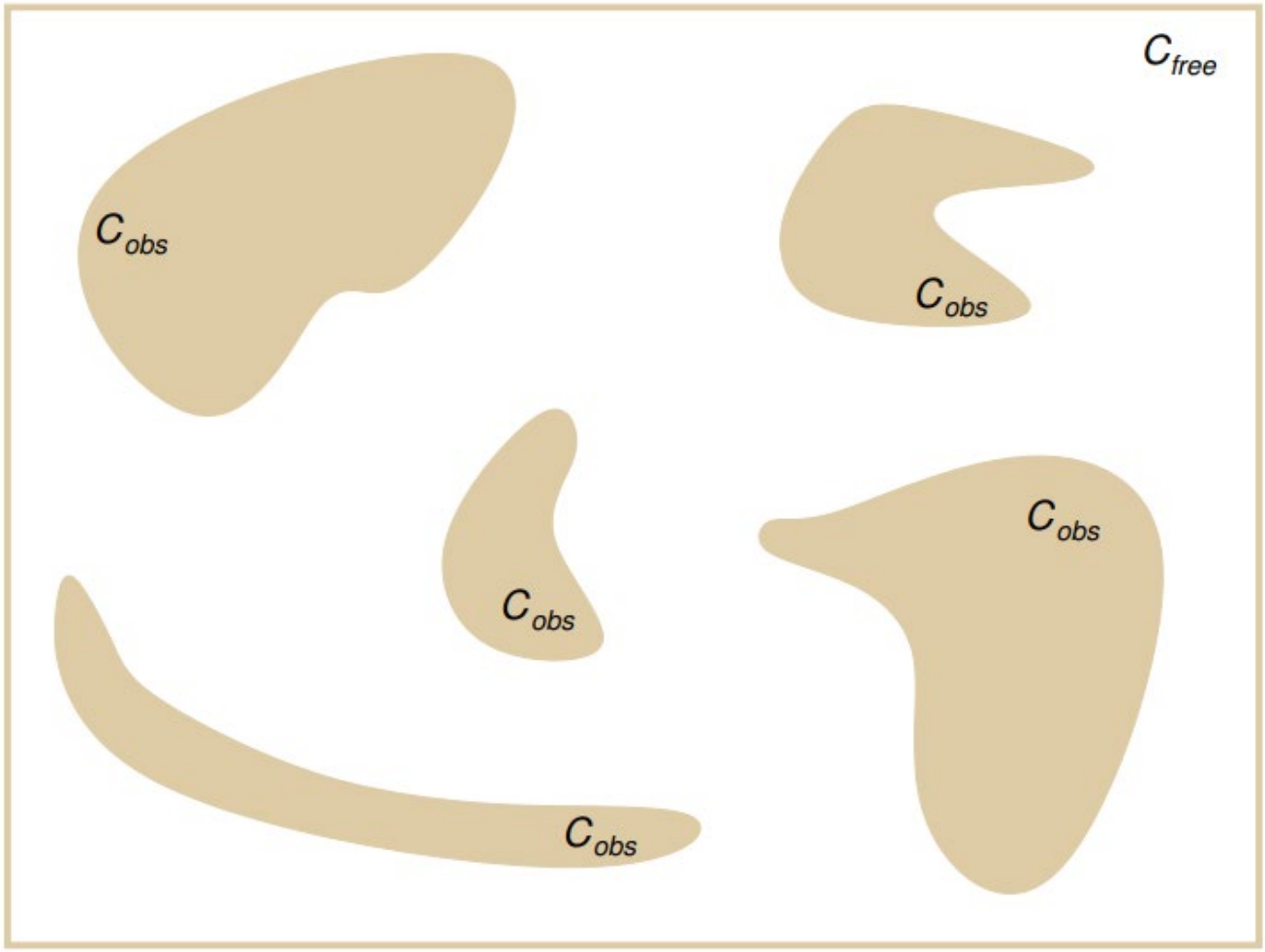
- Use search-based algorithm to find valid path



Kavraki, L. et al., 1996. *Probabilistic Roadmaps for Path Planning in High Dimensional Configuration Spaces*, In: IEEE Transactions on Robotics and Automation. 12(4):566–580.

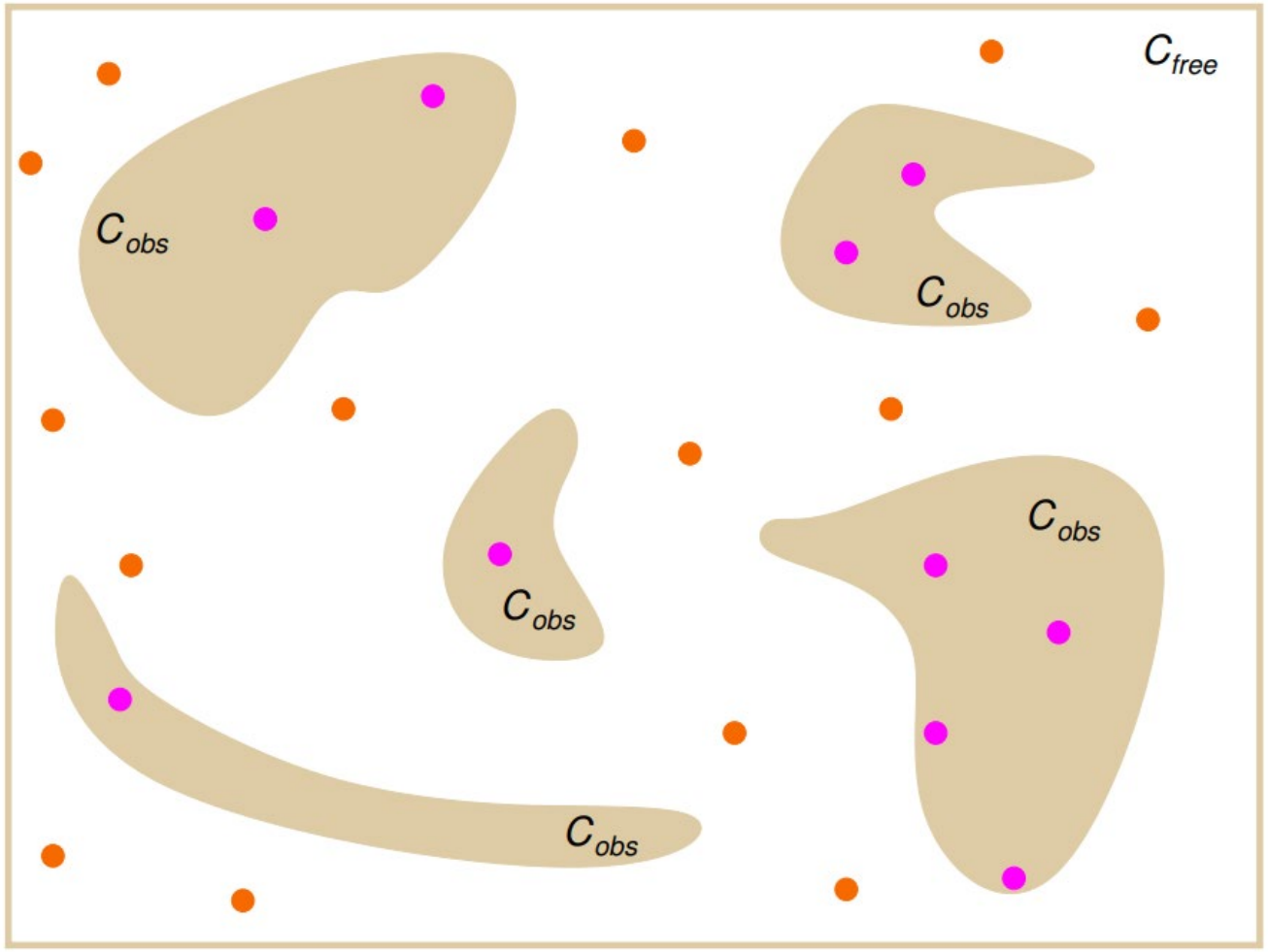
PRM – Learning Phase (1)

- Define free configuration space and obstacles



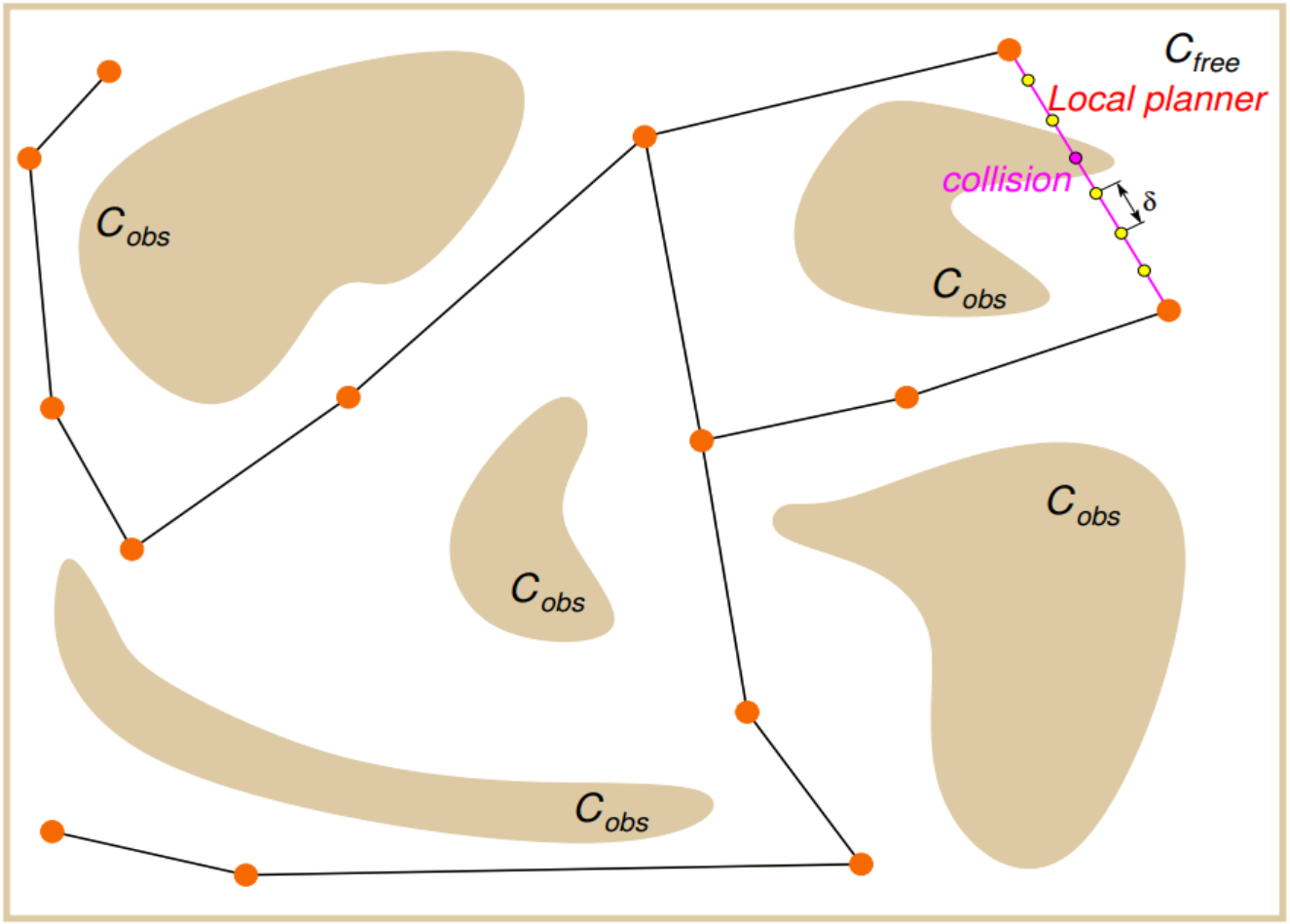
PRM – Learning Phase (1)

- Sample at random from configuration space



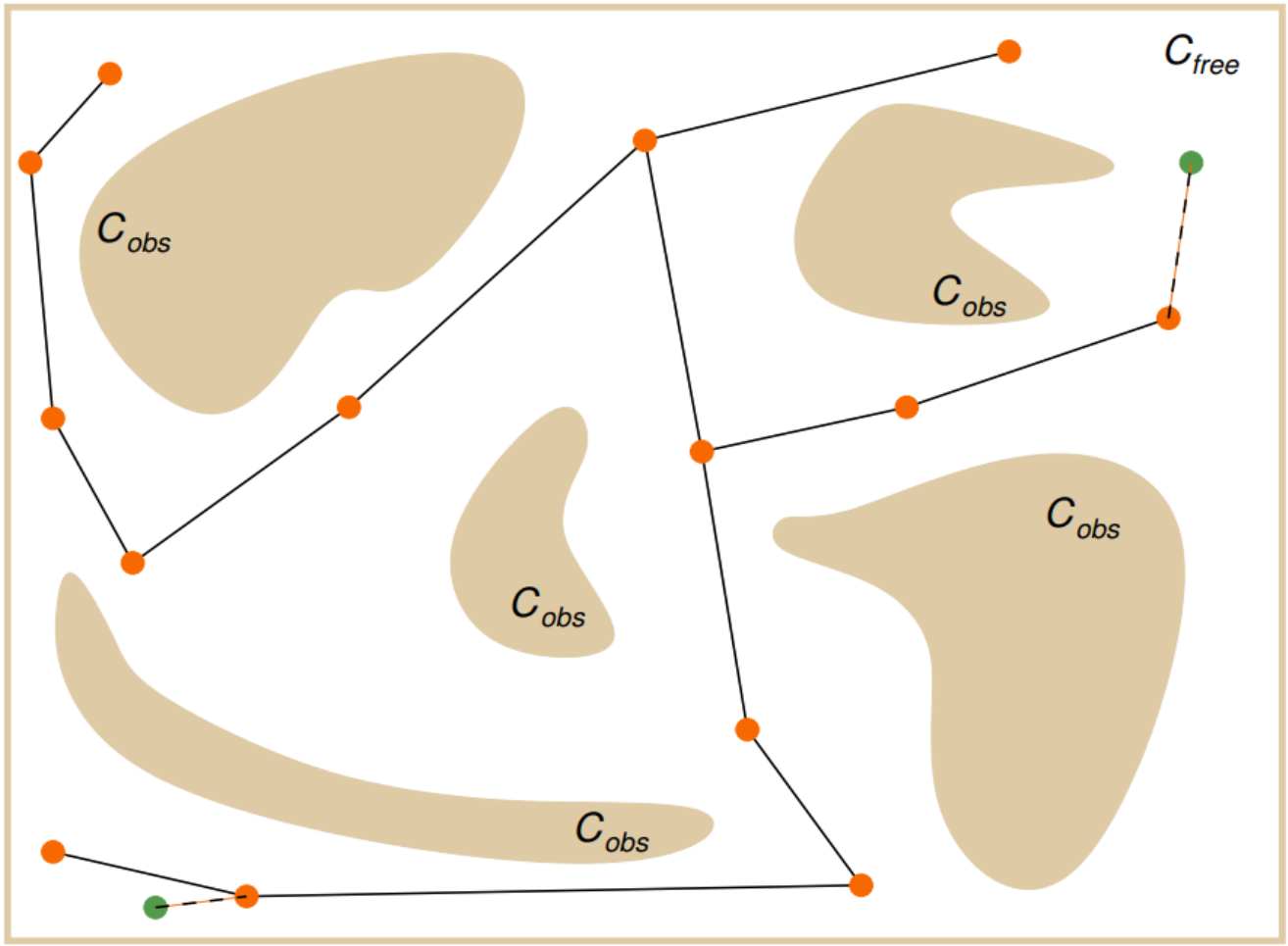
PRM – Learning Phase (1)

- Connect nearby points if a local planner guarantees collision-free paths



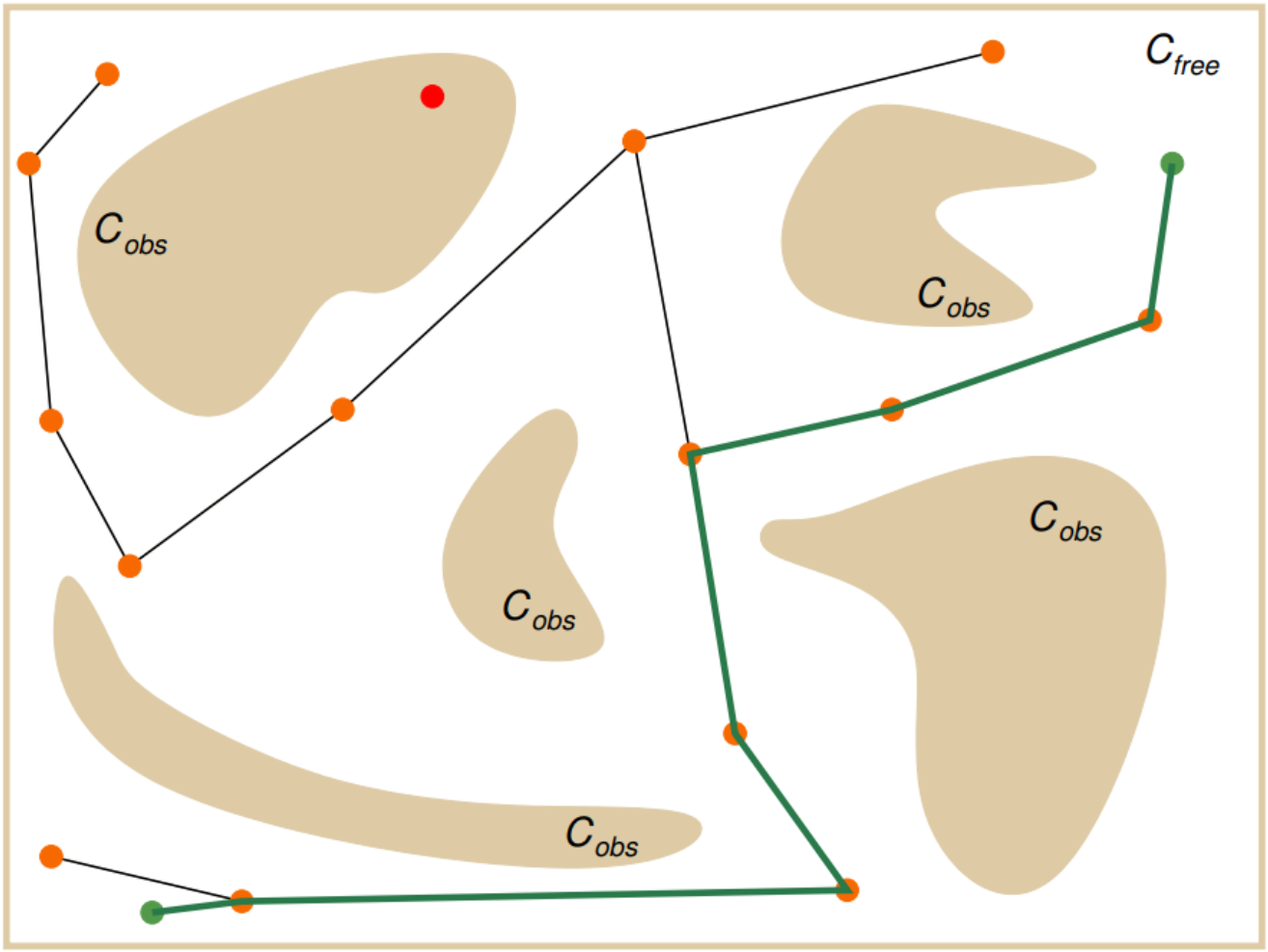
PRM – Query Phase (2)

- Query start and goal configurations



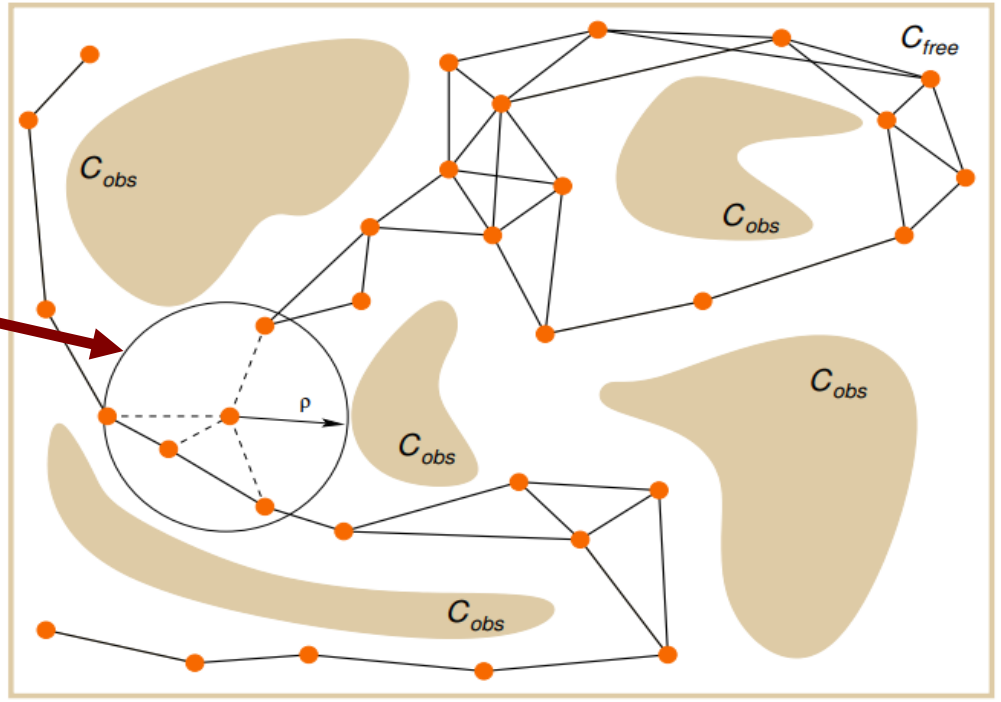
PRM – Query Phase (2)

- Apply graph search to find final path



PRM – Practical Implementation

Connect nodes within ball of a fixed radius ρ

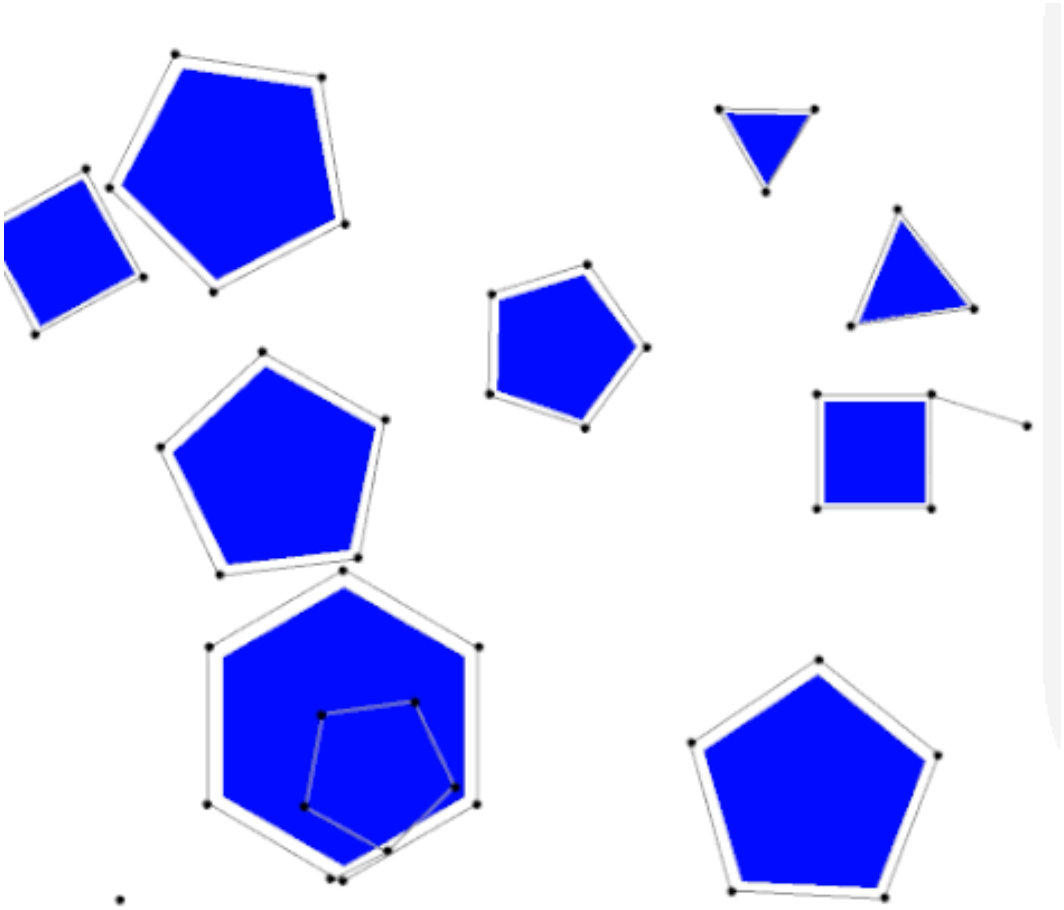


BUILD_ROADMAP

```
1  G.init(); i ← 0;
2  while i < N
3      if α(i) ∈ C_free then
4          G.add_vertex(α(i)); i ← i + 1;
5          for each q ∈ NEIGHBORHOOD(α(i),G)
6              if ((not G.same_component(α(i), q)) and CONNECT(α(i), q)) then
7                  G.add_edge(α(i), q);
```

https://cw.fel.cvut.cz/b181/_media/courses/b4m36uir/lectures/b4m36uir-lec05-slides.pdf
https://ethz.ch/content/dam/ethz/special-interest/mavt/robotics-n-intelligent-systems/asl-dam/documents/lectures/autonomous_mobile_robots/spring-2018/Planning_II.pdf

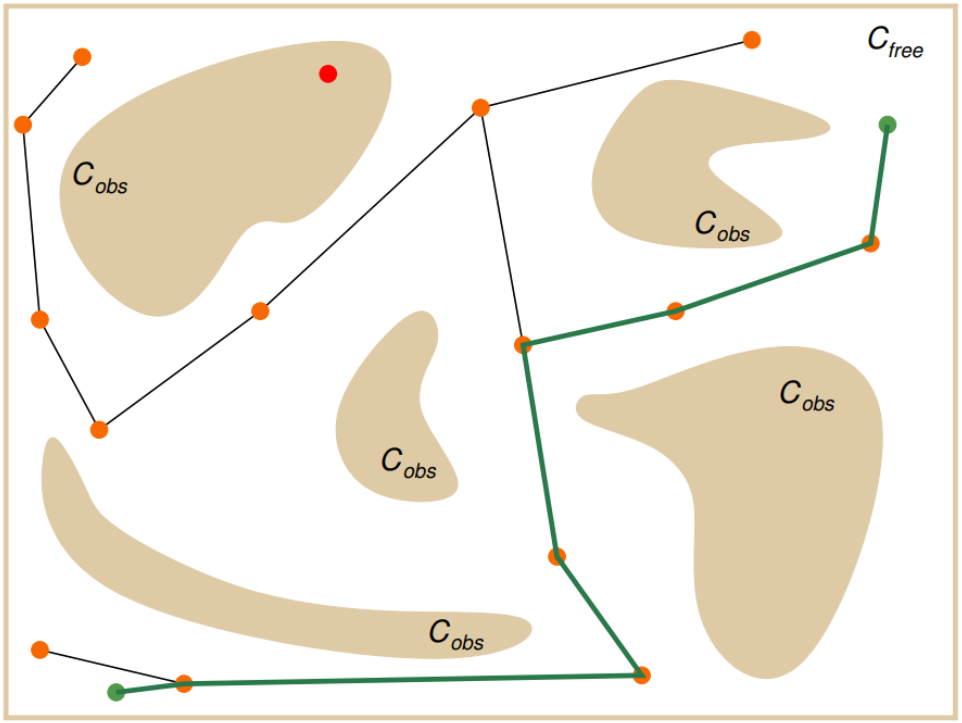
PRM – Example



https://upload.wikimedia.org/wikipedia/commons/9/99/PRM_with_Ob-maps.gif

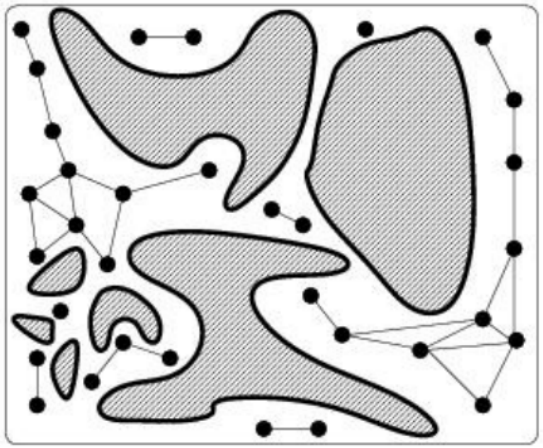
PRM – Discussion

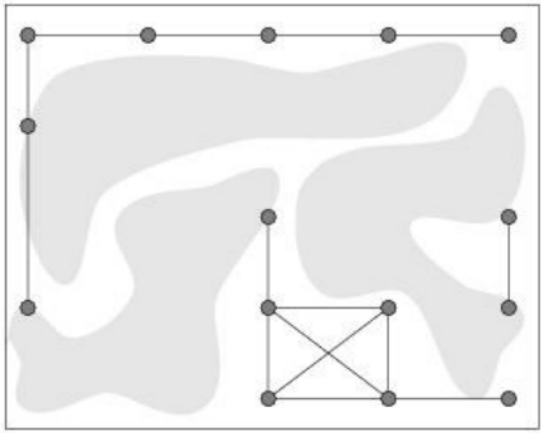
- **Advantages**:
  - Simple, easy to implement
  - Probabilistic completeness
  - Multi-query

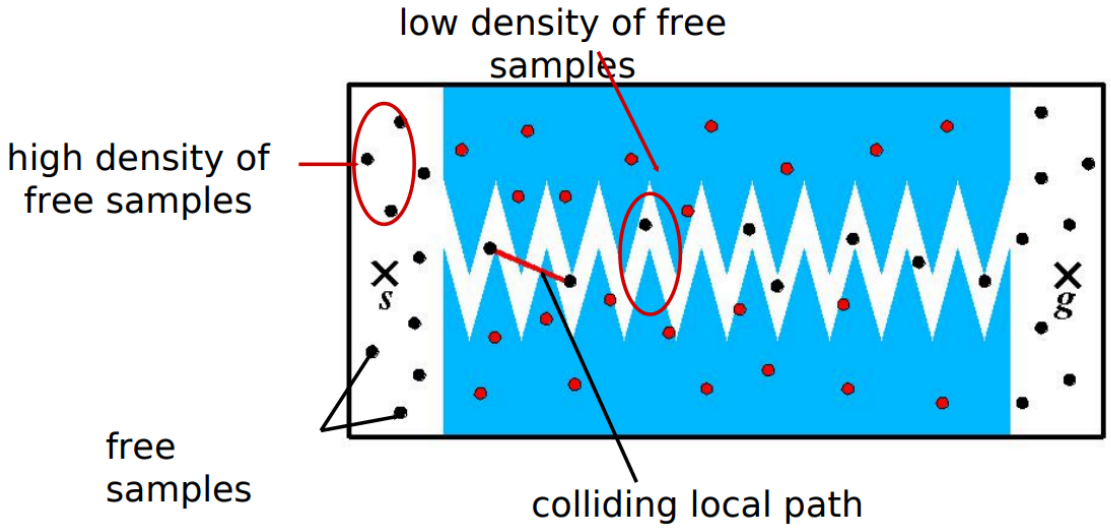


PRM – Discussion

- **Disadvantages**:
  - Narrow passages
  - Lack of connectivity







- Not suitable for dynamic environments

Tree-based Planners

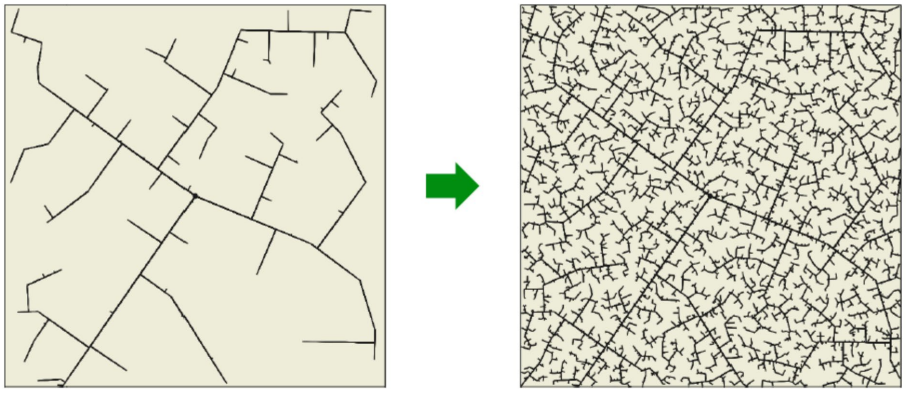
Tree-based Planners – Concept

- **General approach**:
  1. Grow a tree rooted at the starting node
  2. Randomly sample new nodes
  3. Connect new nodes to tree, if connecting path is collision-free
  4. Terminate until tree is sufficiently close to goal

- **Examples**:
  - RRT, RRT*, EST, C-FOREST, KPIECE, …

Rapidly Exploring Dense Tree (RDT)

- General family of tree-based planners
- Incrementally construct a search tree
- If sampling is random, RDT is a **rapidly exploring random tree (RRT)**

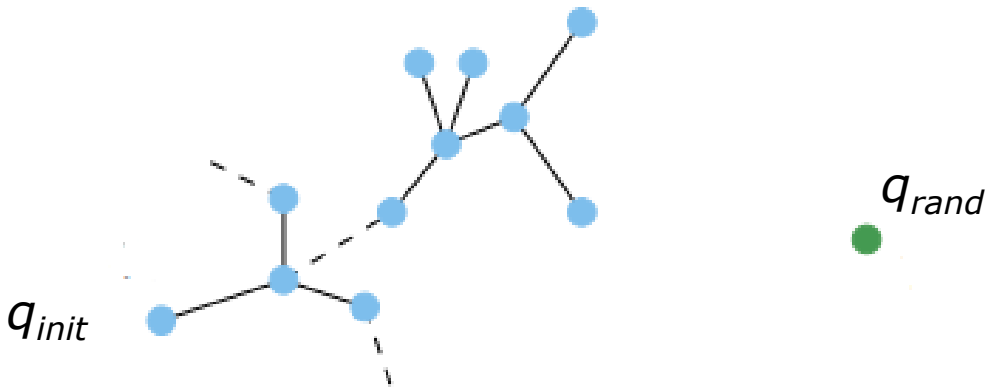


45 iterations 2345 iterations

LaValle, S. M., 1998. *Rapidly-exploring random trees: A new tool for path planning*, Technical Report. Computer Science Department, Iowa State University. TR 98-11.

RRT – Algorithm

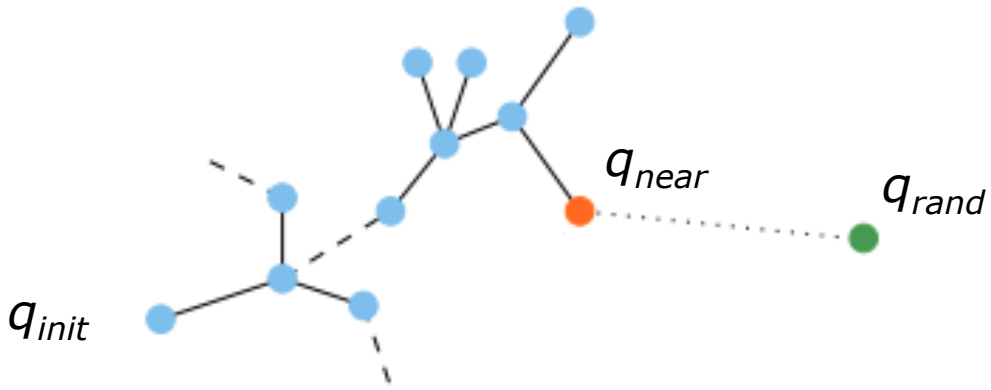
```
Algorithm BuildRRT
  Input: Initial configuration q_init, number of vertices in RRT K, incremental distance Δq)
  Output: RRT graph G

  G.init(q_init)
  for k = 1 to K
    q_rand ← RAND_CONF()
    q_near ← NEAREST_VERTEX(q_rand, G)
    q_new ← NEW_CONF(q_near, q_rand, Δq)
    G.add_vertex(q_new)
    G.add_edge(q_near, q_new)
  return G
```



https://github.com/VincentChen95/Robotic-Motion-Planning-A-Star-RRT-and-RRT-Star-Search

RRT – Algorithm

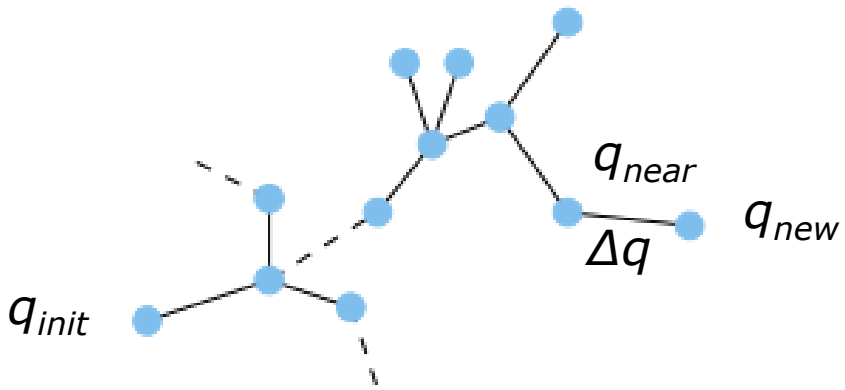
```
Algorithm BuildRRT
  Input: Initial configuration q_init, number of vertices in RRT K, incremental distance Δq)
  Output: RRT graph G

  G.init(q_init)
  for k = 1 to K
    q_rand ← RAND_CONF()
    q_near ← NEAREST_VERTEX(q_rand, G)
    q_new ← NEW_CONF(q_near, q_rand, Δq)
    G.add_vertex(q_new)
    G.add_edge(q_near, q_new)
  return G
```



https://github.com/VincentChen95/Robotic-Motion-Planning-A-Star-RRT-and-RRT-Star-Search

RRT – Algorithm

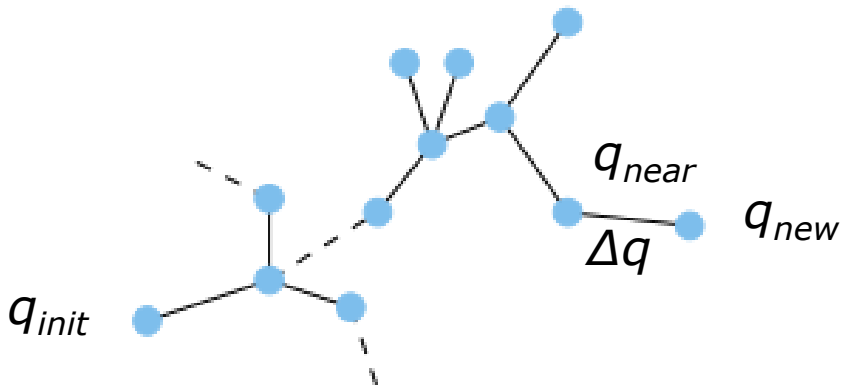
```
Algorithm BuildRRT
  Input: Initial configuration q_init, number of vertices in RRT K, incremental distance Δq)
  Output: RRT graph G

  G.init(q_init)
  for k = 1 to K
    q_rand ← RAND_CONF()
    q_near ← NEAREST VERTEX(q_rand, G)
    q_new ← NEW_CONF(q_near, q_rand, Δq)
    G.add_vertex(q_new)
    G.add_edge(q_near, q_new)
  return G
```



https://github.com/VincentChen95/Robotic-Motion-Planning-A-Star-RRT-and-RRT-Star-Search

RRT – Algorithm

```
Algorithm BuildRRT
  Input: Initial configuration q_init, number of vertices in RRT K, incremental distance Δq)
  Output: RRT graph G

  G.init(q_init)
  for k = 1 to K
    q_rand ← RAND_CONF()
    q_near ← NEAREST VERTEX(q_rand, G)
    q_new ← NEW_CONF(q_near, q_rand, Δq)
    G.add_vertex(q_new)
    G.add_edge(q_near, q_new)
  return G
```

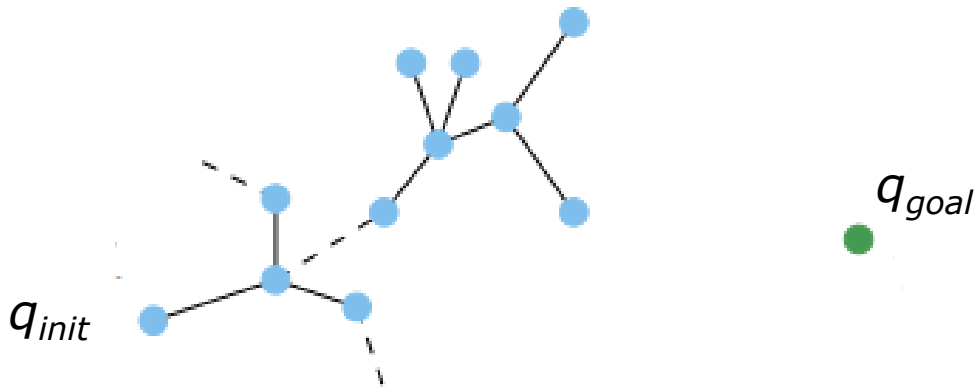


https://github.com/VincentChen95/Robotic-Motion-Planning-A-Star-RRT-and-RRT-Star-Search

RRT – Algorithm

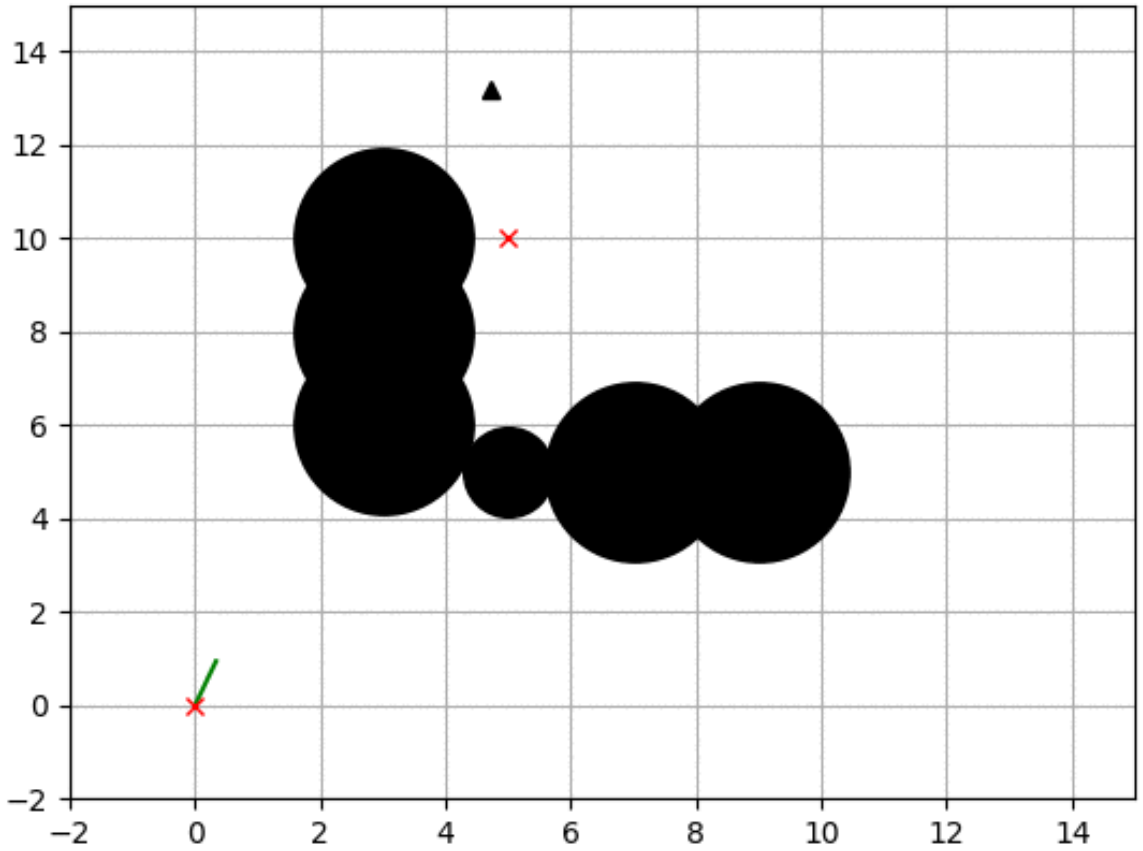
```
Algorithm BuildRRT
  Input: Initial configuration q_init, number of vertices in RRT K, incremental distance Δq)
  Output: RRT graph G

  G.init(q_init)
  for k = 1 to K
    q_rand ← RAND_CONF()
    q_near ← NEAREST_VERTEX(q_rand, G)
    q_new ← NEW_CONF(q_near, q_rand, Δq)
    G.add_vertex(q_new)
    G.add_edge(q_near, q_new)
  return G
```

Set $q_{rand} = q_{goal}$ some fraction of the time



https://github.com/VincentChen95/Robotic-Motion-Planning-A-Star-RRT-and-RRT-Star-Search

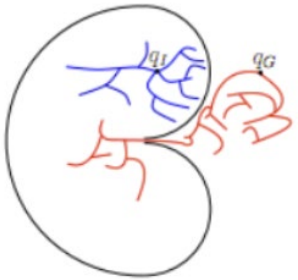
RRT – Example



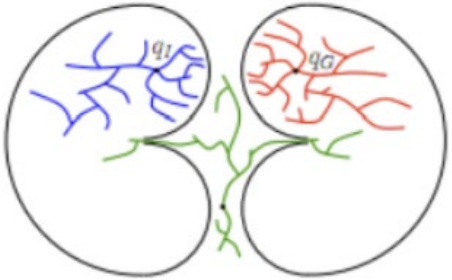
https://pythonrobotics.readthedocs.io/en/latest/modules/path_planning.html

RRT Variants

- **Uni-directional**: single tree
- **Bi-directional**: two trees from start and goal



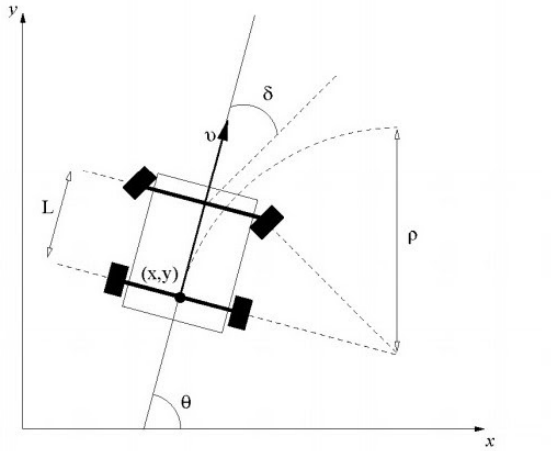
- **Multi-directional**: multiple trees (forests)



- **RRT-Connect**

RRT with Kinematic Constraints

- Car-like robot with kinematic constraints



$$\begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \frac{v}{L}\tan\delta \end{aligned}$$

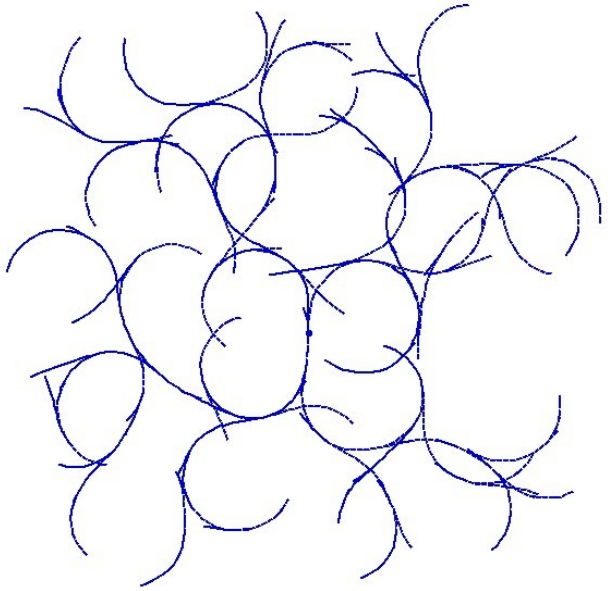
- RRT can directly consider such constraints

https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/

RRT with Kinematic Constraints

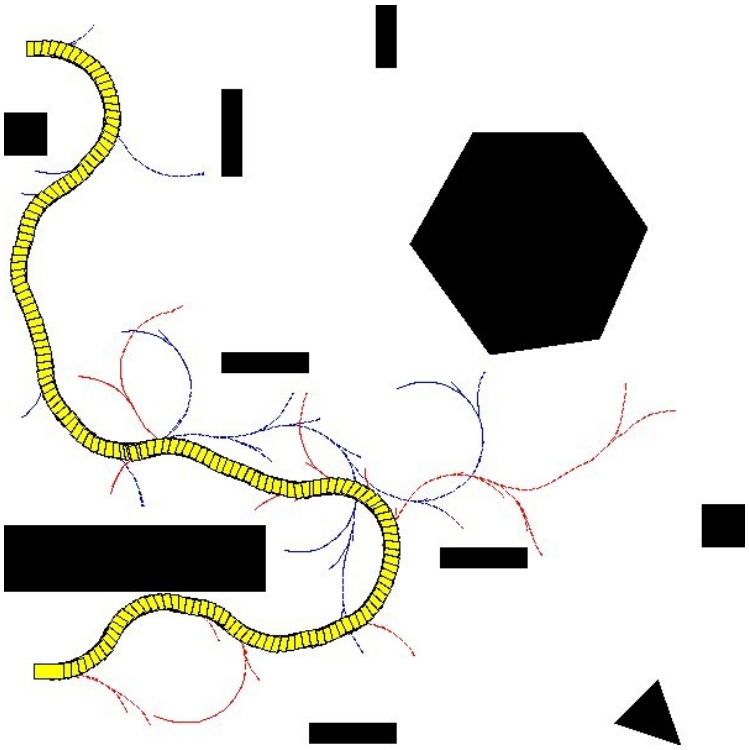
General approach:

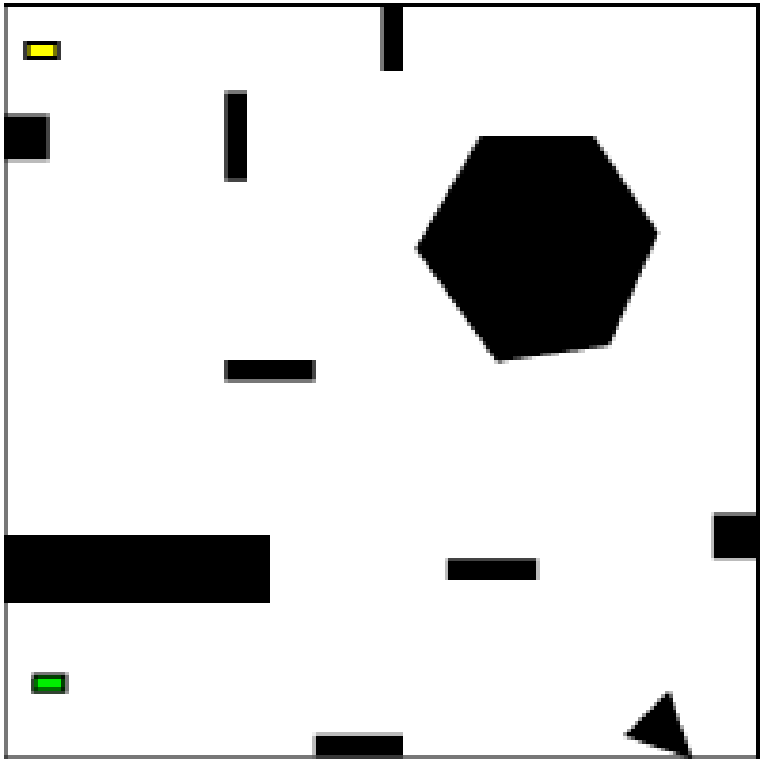
1. Select a configuration from the current tree
2. Sample control input and integrate system equation over a short period
3. If the motion is collision-free, add the endpoint and path to the tree



http://msl.cs.uiuc.edu/rrt/gallery_carsmooth.html

RRT with Kinematic Constraints



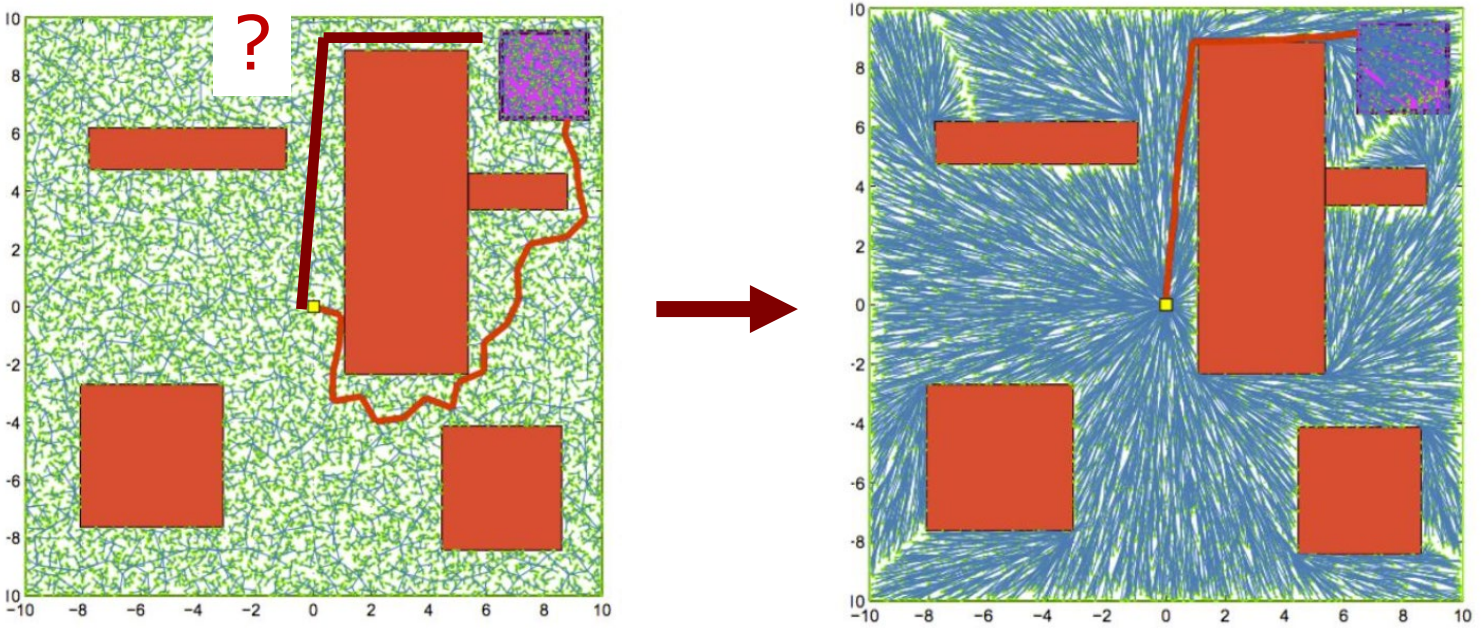


http://msl.cs.uiuc.edu/rrt/gallery_carsmooth.html

Asymptotically Optimal Planners

Asymptotic Optimality

- PRM, RRT are **probabilistically complete**
- However, they are not **asymptotically optimal**
  - Jagged paths
  - Cannot improve on solutions



https://www.programmersought.com/article/80634553105/

Asymptotic Optimality

- PRM, RRT are **probabilistically complete**
- However, they are not **asymptotically optimal**
  - Jagged paths
  - Cannot improve on solutions

- "**Star**" versions of algorithms:
  - Idea: Search near neighbours and rewire graph to find shortest path
  - RRG, PRM*, RRT*

Karaman, S. and Frazzoli, E., 2011. "Sampling-based algorithms for optimal motion planning", In: *The International Journal of Robotics Research.* 30(7):846-894.

Rapidly-exploring Random Graph (RRG)

- **Incremental** algorithm

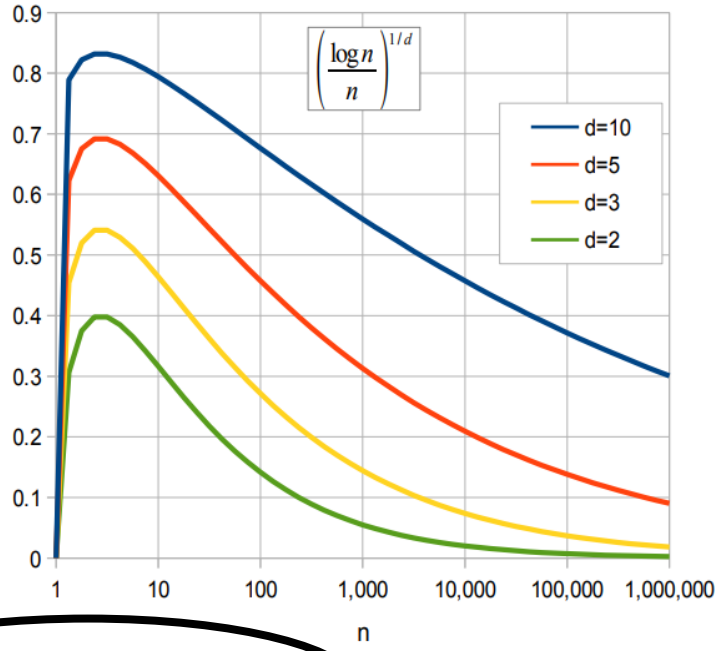
Algorithm : RRG

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$;
for $i = 1, \ldots, n$ **do**
  $x_{\text{rand}} \leftarrow \texttt{SampleFree}_i$;
  $x_{\text{nearest}} \leftarrow \texttt{Nearest}(G = (V, E), x_{\text{rand}})$;
  $x_{\text{new}} \leftarrow \texttt{Steer}(x_{\text{nearest}}, x_{\text{rand}})$;
  if $\texttt{ObtacleFree}(x_{\text{nearest}}, x_{\text{new}})$ **then**
    $X_{\text{near}} \leftarrow \texttt{Near}(G = (V, E), x_{\text{new}}, \min\{\gamma_{\text{RRG}}(\log(\text{card}\,(V))/\,\text{card}\,(V))^{1/d}, \eta\})$;
    $V \leftarrow V \cup \{x_{\text{new}}\}$; $E \leftarrow E \cup \{(x_{\text{nearest}}, x_{\text{new}}), (x_{\text{new}}, x_{\text{nearest}})\}$;
    foreach $x_{\text{near}} \in X_{\text{near}}$ **do**
      if $\texttt{CollisionFree}(x_{\text{near}}, x_{\text{new}})$ **then** $E \leftarrow E \cup \{(x_{\text{near}}, x_{\text{new}}), (x_{\text{new}}, x_{\text{near}})\}$
return $G = (V, E)$;

Karaman, S. and Frazzoli, E., 2011. "Sampling-based algorithms for optimal motion planning", In: *The IJRR.* 30(7):846-894.

Rapidly-exploring Random Graph (RRG)

- **Incremental** algorithm

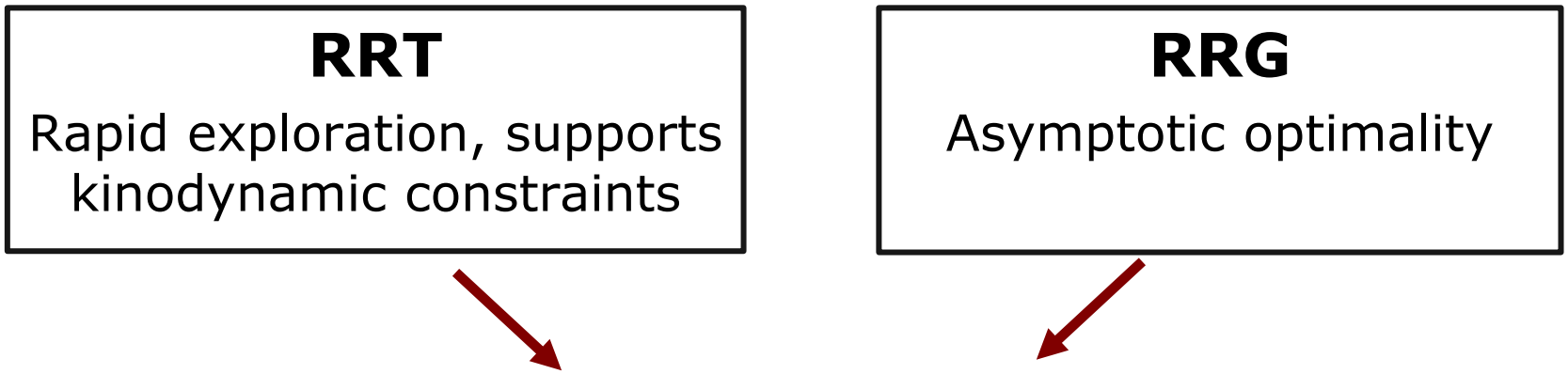


Algorithm : RRG

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$;
for $i = 1, \ldots, n$ **do**
  $x_{\text{rand}} \leftarrow \texttt{SampleFree}_i$;
  $x_{\text{nearest}} \leftarrow \texttt{Nearest}(G = (V, E), x_{\text{rand}})$;
  $x_{\text{new}} \leftarrow \texttt{Steer}(x_{\text{nearest}}, x_{\text{rand}})$;
  if $\texttt{ObtacleFree}(x_{\text{nearest}}, x_{\text{new}})$ **then**
    $X_{\text{near}} \leftarrow \texttt{Near}(G = (V, E), x_{\text{new}}, \min\{\gamma_{\text{RRG}} (\log(\text{card}\,(V))/\,\text{card}\,(V))^{1/d}, \eta\})$;
    $V \leftarrow V \cup \{x_{\text{new}}\}$; $E \leftarrow E \cup \{(x_{\text{nearest}}, x_{\text{new}}), (x_{\text{new}}, x_{\text{nearest}})\}$;
    foreach $x_{\text{near}} \in X_{\text{near}}$ **do**
      if $\texttt{CollisionFree}(x_{\text{near}}, x_{\text{new}})$ **then** $E \leftarrow E \cup \{(x_{\text{near}}, x_{\text{new}}), (x_{\text{new}}, x_{\text{near}})\}$
return $G = (V, E)$;

Karaman, S. and Frazzoli, E., 2011. "Sampling-based algorithms for optimal motion planning", In: *The IJRR.* 30(7):846-894.

RRT*



RRT*

RRG with no cycles

- Searches nearest neighbours like RRG
- "Rewires" tree as better paths are discovered

Karaman, S. and Frazzoli, E., 2011. "Sampling-based algorithms for optimal motion planning", In: *The IJRR*. 30(7):846-894.

RRT* - Rewiring Procedure

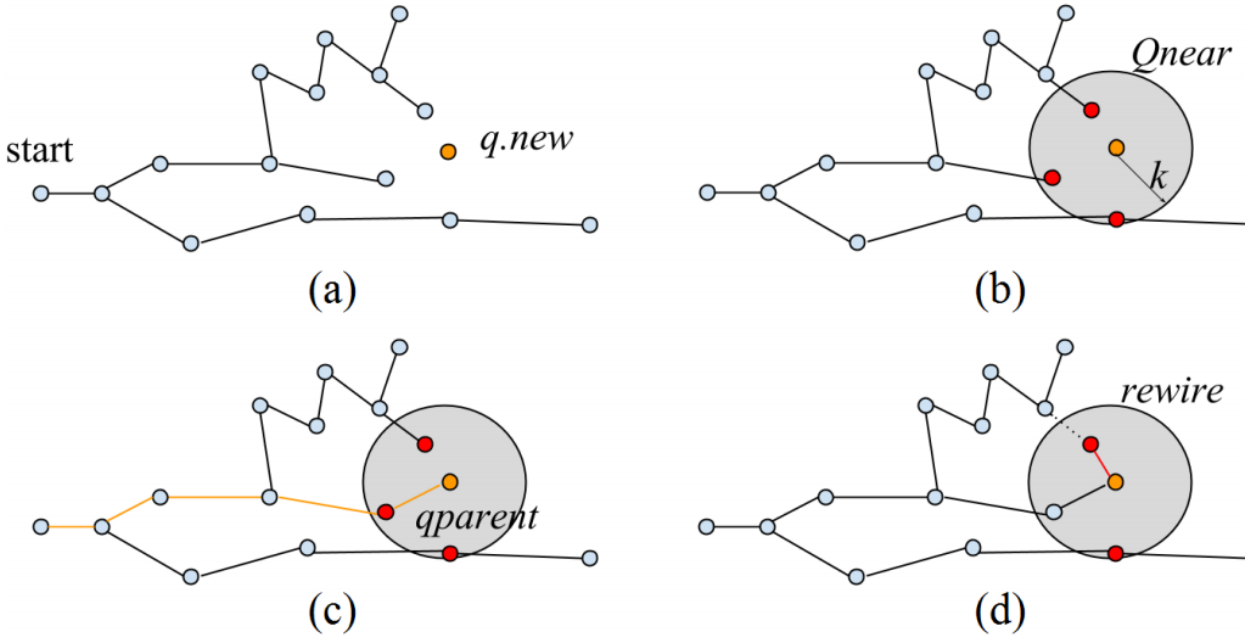
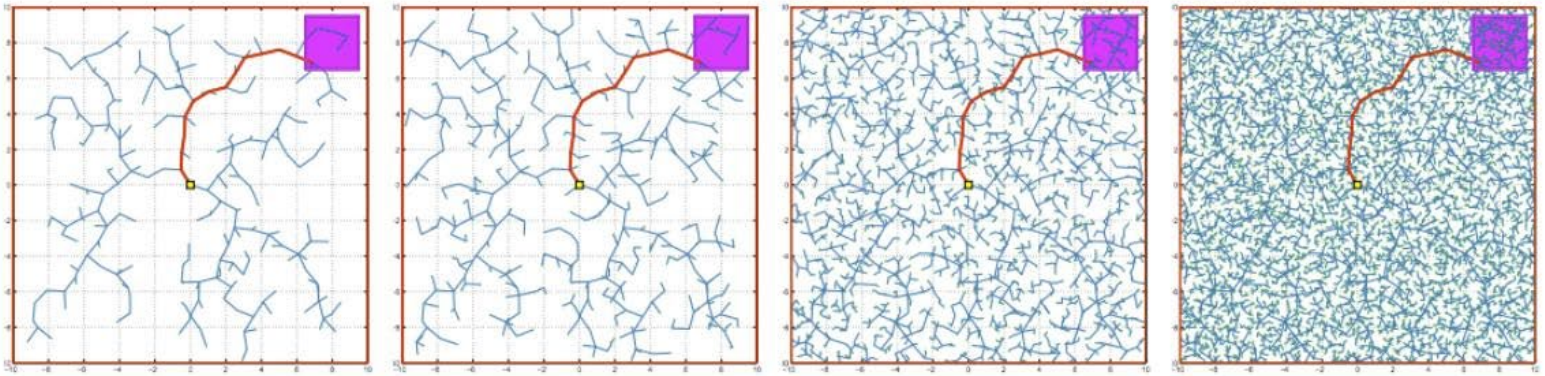


FIGURE 19. **Illustrating the operation of RRT* (a) A new random node, q_{new}, is selected, shown as orange node (b) Near vertices procedure returns a set, Q_{near}, of all nodes, shown as red nodes, within a certain distance of the new node (circular area shaded in grey) (c) q_{new} is connected to the node, q_{parent}, that has the shortest route to the start (shown as orange path) (d) The remaining nodes in Q_{near} are rewired through q_{new}, if it provides a shorter path towards the start.**

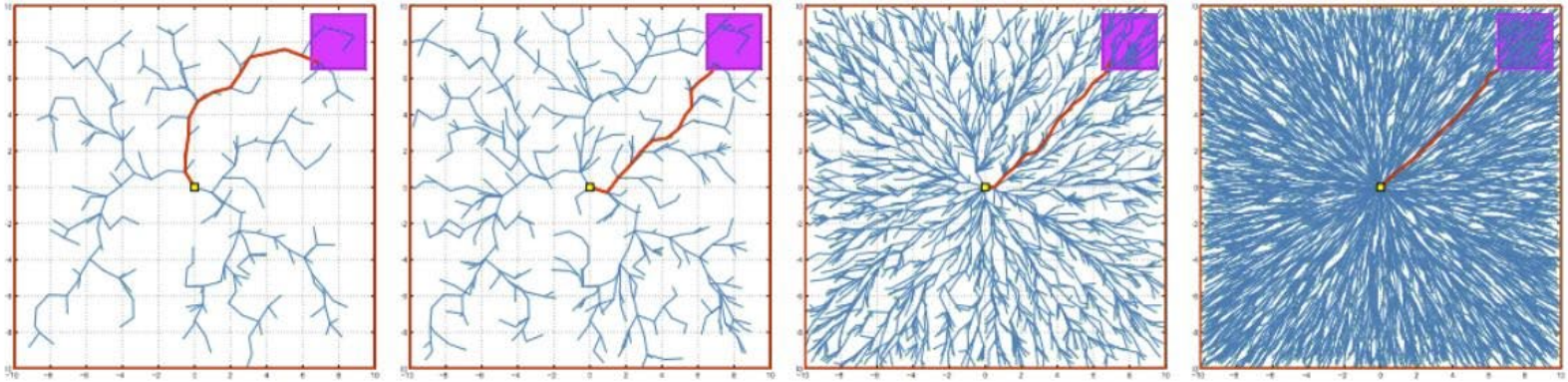
Elbanhawi, M. and Simic, M., 2014. "Sampling-Based Robot Motion Planning: A Review", In: *IEEE Access.* 2: 56-77.

RRT vs. RRT*

RRT



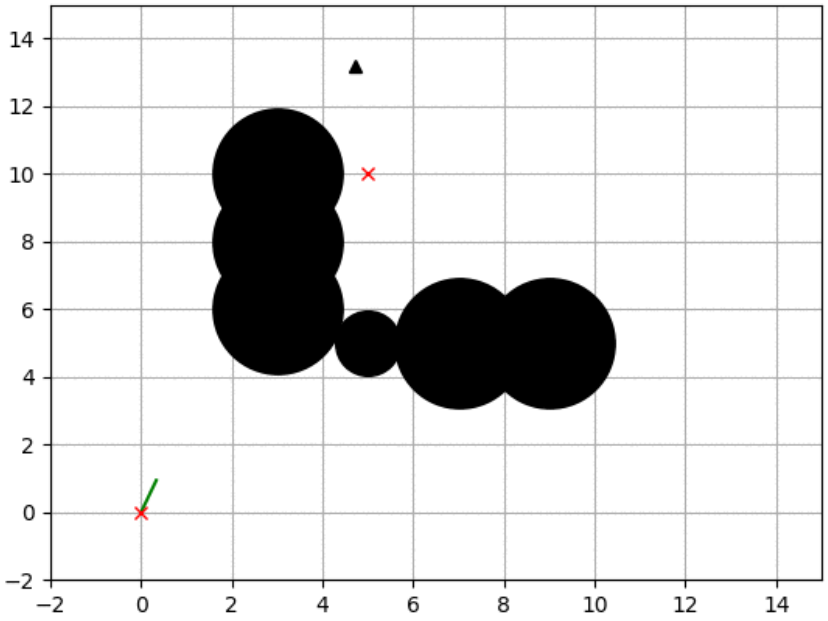
RRT*



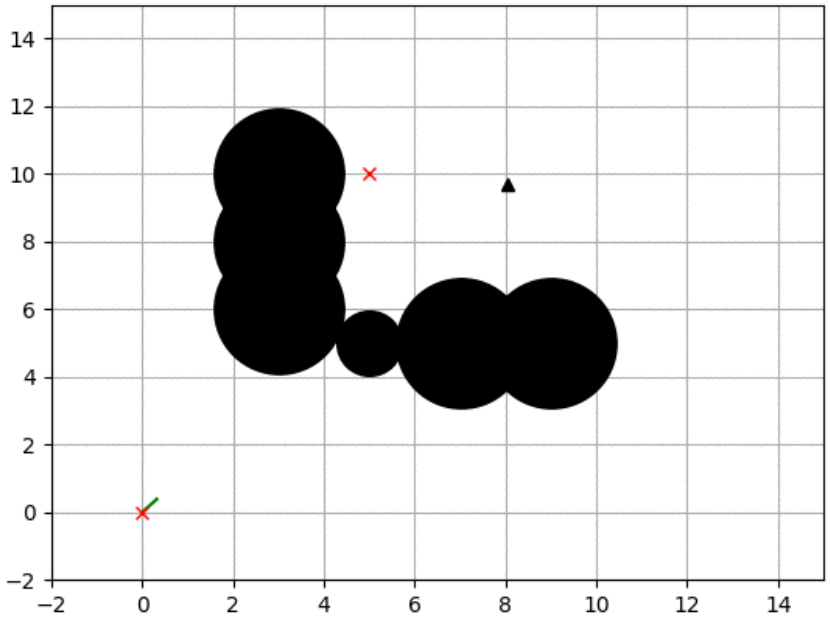
https://www.programmersought.com/article/80634553105/

RRT vs. RRT*

RRT



RRT*



https://pythonrobotics.readthedocs.io/en/latest/modules/path_planning.html

Overview of Methods

	Algorithm	Probabilistic Completeness	Asymptotic Optimality	Monotone Convergence	Time Complexity		Space Complexity
					Processing	Query	
Existing Algorithms	PRM	Yes	No	Yes	$O(n \log n)$	$O(n \log n)$	$O(n)$
	sPRM	Yes	Yes	Yes	$O(n^2)$	$O(n^2)$	$O(n^2)$
	k-sPRM	Conditional	No	No	$O(n \log n)$	$O(n \log n)$	$O(n)$
	RRT	Yes	No	Yes	$O(n \log n)$	$O(n)$	$O(n)$
Proposed Algorithms	PRM*	Yes	Yes	No	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
	k-PRM*						
	RRG	Yes	Yes	Yes	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
	k-RRG						
	RRT*	Yes	Yes	Yes	$O(n \log n)$	$O(n)$	$O(n)$
	k-RRT*						

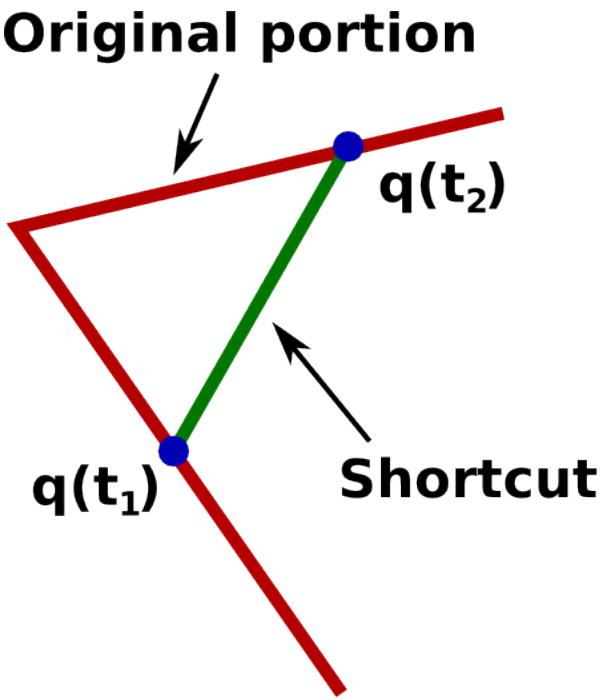
n: Number of samples for a fixed environment.

Karaman, S. and Frazzoli, E., 2011. "Sampling-based algorithms for optimal motion planning", In: *The IJRR*. 30(7):846-894.

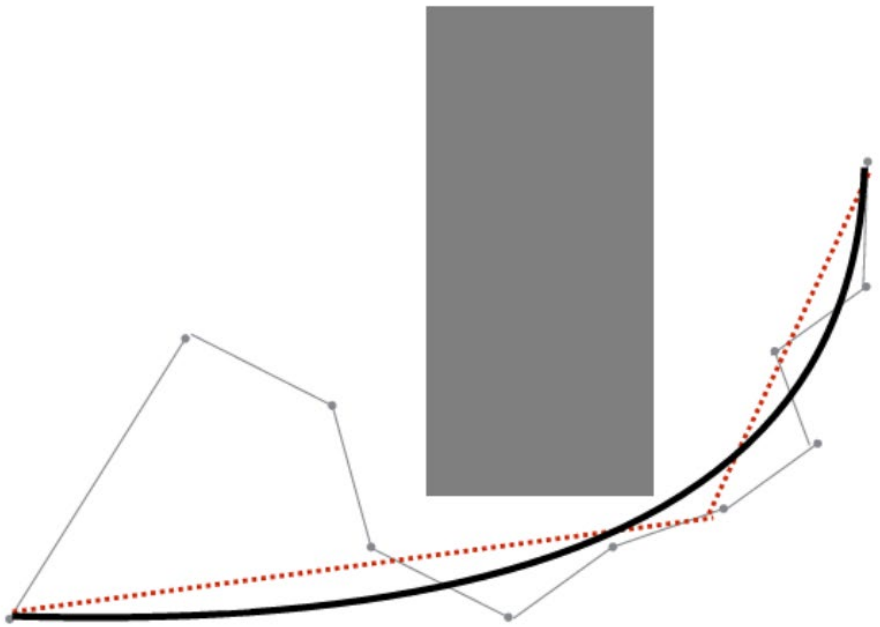
Post-Processing

- Suboptimal paths from sampling-based methods are often **post-processed**

Path shortcutting



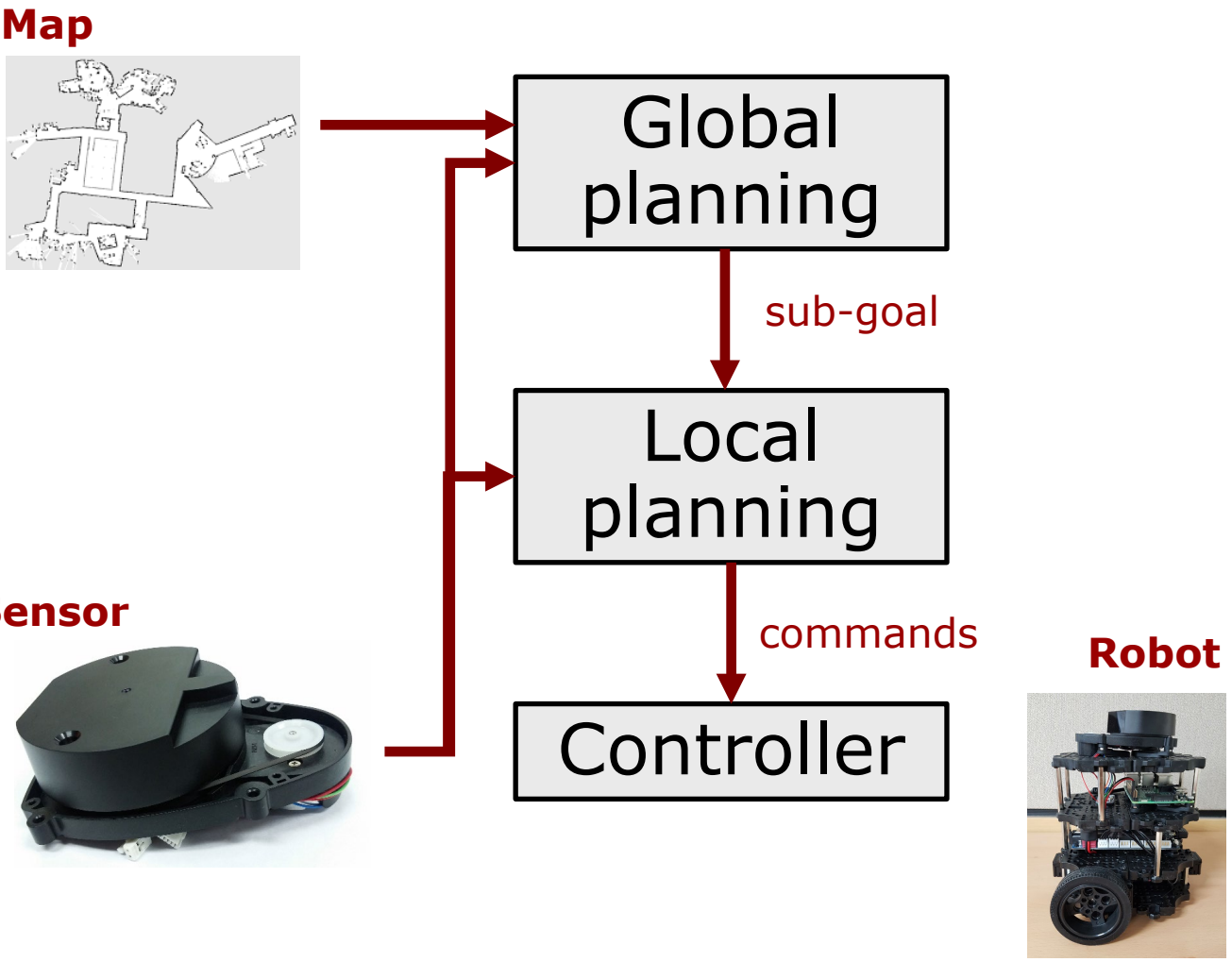
Path smoothing



https://crigroup.gitbooks.io/osrobotics/content/planning/post_processing.html

Karaman, S. and Frazzoli, E., 2011. "Sampling-based algorithms for optimal motion planning", In: *The IJRR*. 30(7):846-894.

Global vs. Local Planning



http://emanual.robotis.com
http://ais.informatik.uni-freiburg.de/teaching/ws13/mapping/pdf/slam10-gridmaps-4.pdf

Application – UAV

Oleynikova, H. et al., 2016. "Continuous-Time Trajectory Optimization for Online UAV Replanning", In: *The IEEE ICRA.*

Summary

- Overview of sampling-based methods
- Roadmap-based planners
  - PRM
- Tree-based planners
  - RDT, RRT
- Asymptotically optimal planners
  - RRG, RRT*
- Practical considerations

Further Reading

- Autonomous Navigation, Part 4: Path Planning with A* and RRT – YouTube
- kingston2018sampling-based-methods-for-motion-planning.pdf (kavrakilab.org)
- https://github.com/VincentChen95/Robotic-Motion-Planning-A-Star-RRT-and-RRT-Star-Search
- randomplanning.pdf (columbia.edu)
- Review paper:
  - Elbanhawi, M. and Simic, M., 2014. "Sampling-Based Robot Motion Planning: A Review", In: *IEEE Access.* 2: 56-77.
- Optimal sampling-based techniques:
  - Karaman, S. and Frazzoli, E., 2011. "Sampling-based algorithms for optimal motion planning", In: *The IJRR*. 30(7):846-894.
- Planning Algorithms – Steven M. LaValle (2006)
  - Planning Algorithms / Motion Planning (lavalle.pl)
  - Ch. 5

Thank you for your attention