MEAM 520
Lecture 12: Probabilistic Trajectory Planning

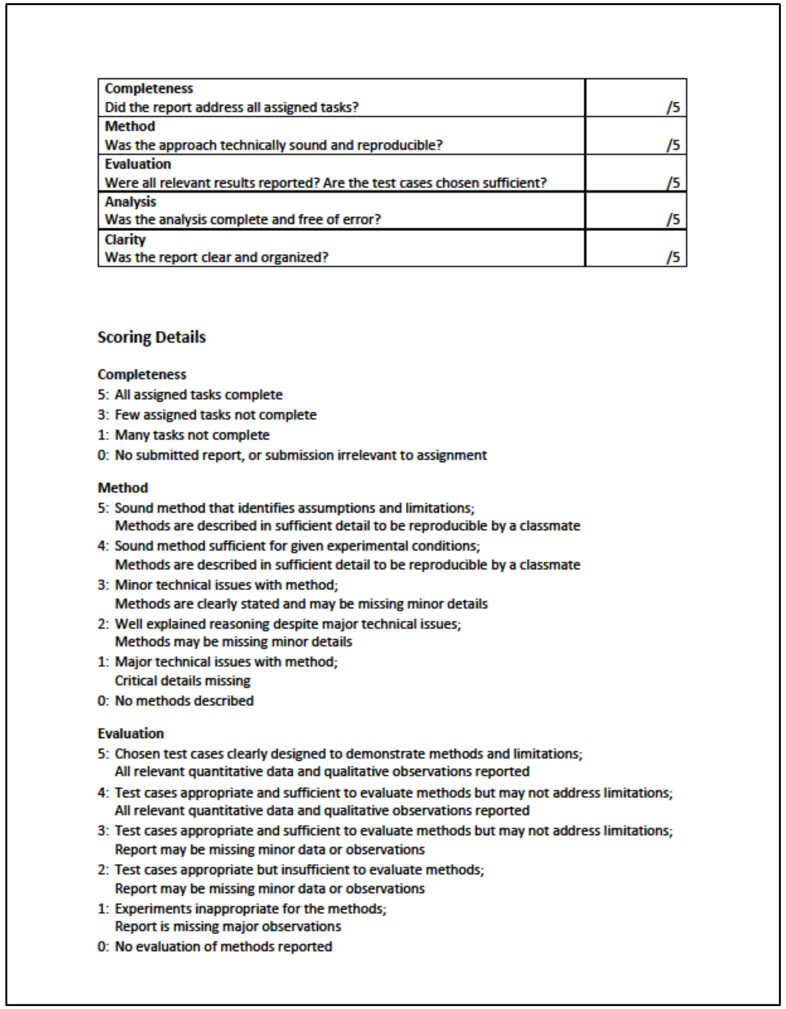
Cynthia Sung, Ph.D.

Mechanical Engineering & Applied Mechanics

University of Pennsylvania

Lab 1 grades posted

- Take a look at the comments left by the TA
- Grading is according to our lab report rubric
- Scoring is written so that 5=A, 4=B, 3=C, …
- Sample lab report posted on Canvas under Files > Example Lab Reports > Lab1_example.pdf
- If you believe an error has been made, post a regrade request on Piazza (not Canvas!)
- We will only consider data/information that is present in the report you submitted. No additional info please!



Completeness Did the report address all assigned tasks?	/5
Method Was the approach technically sound and reproducible?	/5
Evaluation Were all relevant results reported? Are the test cases chosen sufficient?	/5
Analysis Was the analysis complete and free of error?	/5
Clarity Was the report clear and organized?	/5

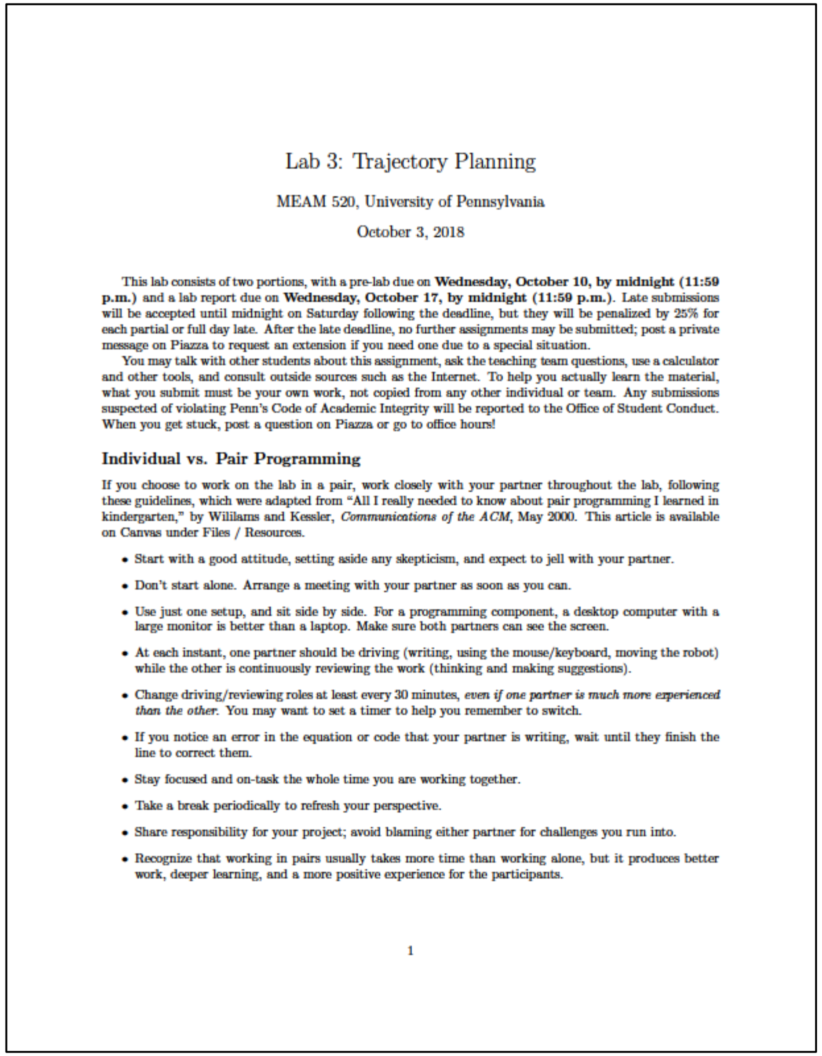
Scoring Details

Completeness
5: All assigned tasks complete
3: Few assigned tasks not complete
1: Many tasks not complete
0: No submitted report, or submission irrelevant to assignment

Method
5: Sound method that identifies assumptions and limitations;
Methods are described in sufficient detail to be reproducible by a classmate
4: Sound method sufficient for given experimental conditions;
Methods are described in sufficient detail to be reproducible by a classmate
3: Minor technical issues with method;
Methods are clearly stated and may be missing minor details
2: Well explained reasoning despite major technical issues;
Methods may be missing minor details
1: Major technical issues with method;
Critical details missing
0: No methods described

Evaluation
5: Chosen test cases clearly designed to demonstrate methods and limitations;
All relevant quantitative data and qualitative observations reported
4: Test cases appropriate and sufficient to evaluate methods but may not address limitations;
All relevant quantitative data and qualitative observations reported
3: Test cases appropriate and sufficient to evaluate methods but may not address limitations;
Report may be missing minor data or observations
2: Test cases appropriate but insufficient to evaluate methods;
Report may be missing minor data or observations
1: Experiments inappropriate for the methods;
Report is missing major observations
0: No evaluation of methods reported

Lab 3 is posted (due 10/17)



Lab 3: Trajectory Planning

MEAM 520, University of Pennsylvania

October 3, 2018

This lab consists of two portions, with a pre-lab due on **Wednesday, October 10, by midnight (11:59 p.m.)** and a lab report due on **Wednesday, October 17, by midnight (11:59 p.m.)**. Late submissions will be accepted until midnight on Saturday following the deadline, but they will be penalized by 25% for each partial or full day late. After the late deadline, no further assignments may be submitted; post a private message on Piazza to request an extension if you need one due to a special situation.

You may talk with other students about this assignment, ask the teaching team questions, use a calculator and other tools, and consult outside sources such as the Internet. To help you actually learn the material, what you submit must be your own work, not copied from any other individual or team. Any submissions suspected of violating Penn's Code of Academic Integrity will be reported to the Office of Student Conduct. When you get stuck, post a question on Piazza or go to office hours!

Individual vs. Pair Programming

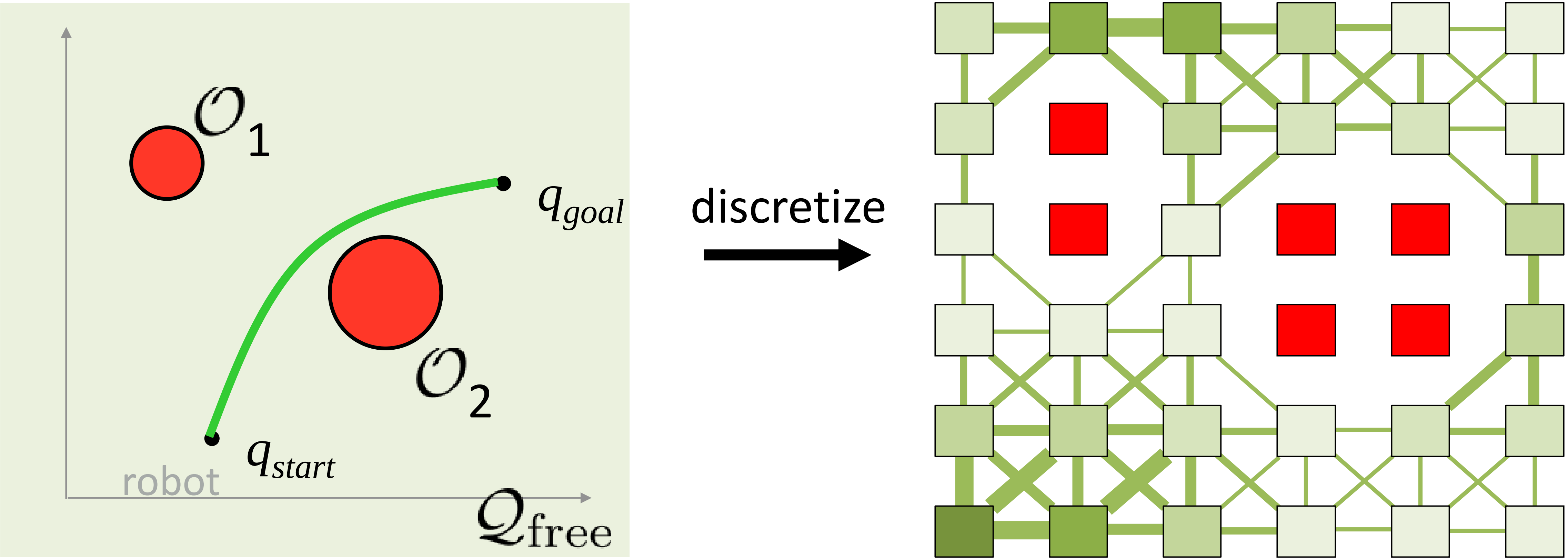
If you choose to work on the lab in a pair, work closely with your partner throughout the lab, following these guidelines, which were adapted from "All I really needed to know about pair programming I learned in kindergarten," by Wililams and Kessler, *Communications of the ACM*, May 2000. This article is available on Canvas under Files / Resources.

- Start with a good attitude, setting aside any skepticism, and expect to jell with your partner.
- Don't start alone. Arrange a meeting with your partner as soon as you can.
- Use just one setup, and sit side by side. For a programming component, a desktop computer with a large monitor is better than a laptop. Make sure both partners can see the screen.
- At each instant, one partner should be driving (writing, using the mouse/keyboard, moving the robot) while the other is continuously reviewing the work (thinking and making suggestions).
- Change driving/reviewing roles at least every 30 minutes, *even if one partner is much more experienced than the other.* You may want to set a timer to help you remember to switch.
- If you notice an error in the equation or code that your partner is writing, wait until they finish the line to correct them.
- Stay focused and on-task the whole time you are working together.
- Take a break periodically to refresh your perspective.
- Share responsibility for your project; avoid blaming either partner for challenges you run into.
- Recognize that working in pairs usually takes more time than working alone, but it produces better work, deeper learning, and a more positive experience for the participants.

1

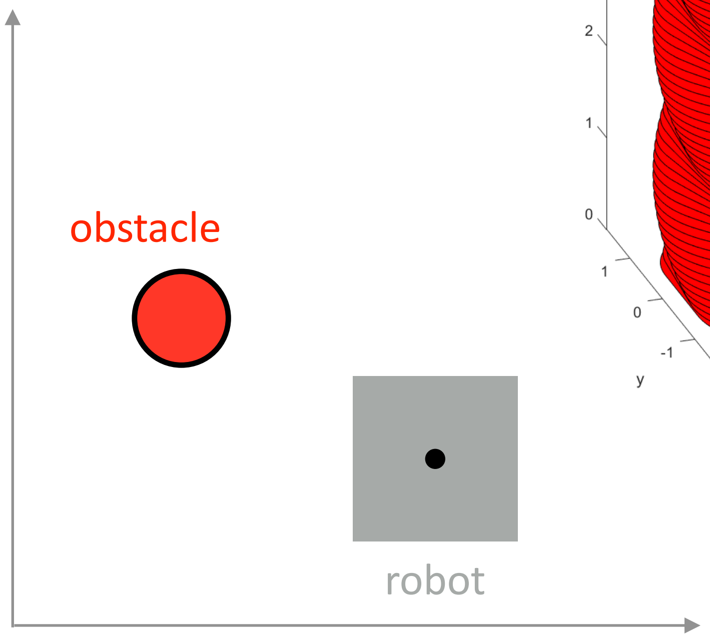
- Remember to create your lab group **BEFORE** turning in the report!
- Turn in your code as a zip and your report as a **separate** pdf
    - Turning in a single zip makes the grader's job more difficult
    - I've authorized them to take off points if you do not follow the submission instructions

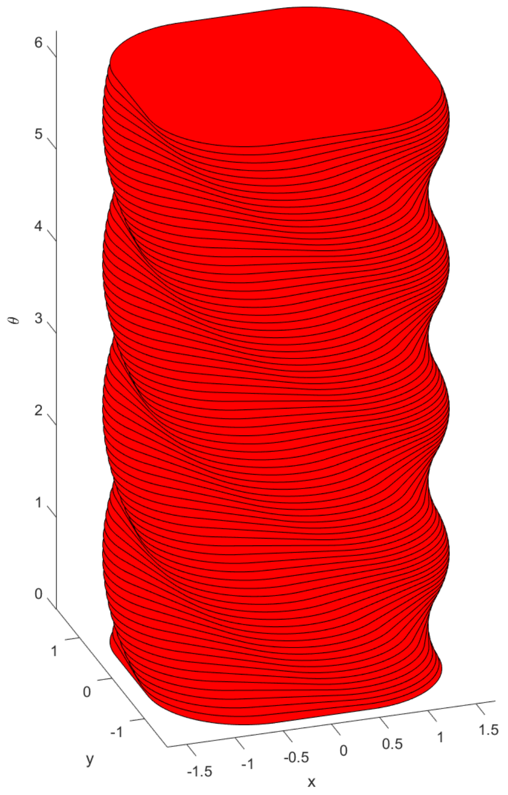
Last Time: Trajectory Planning



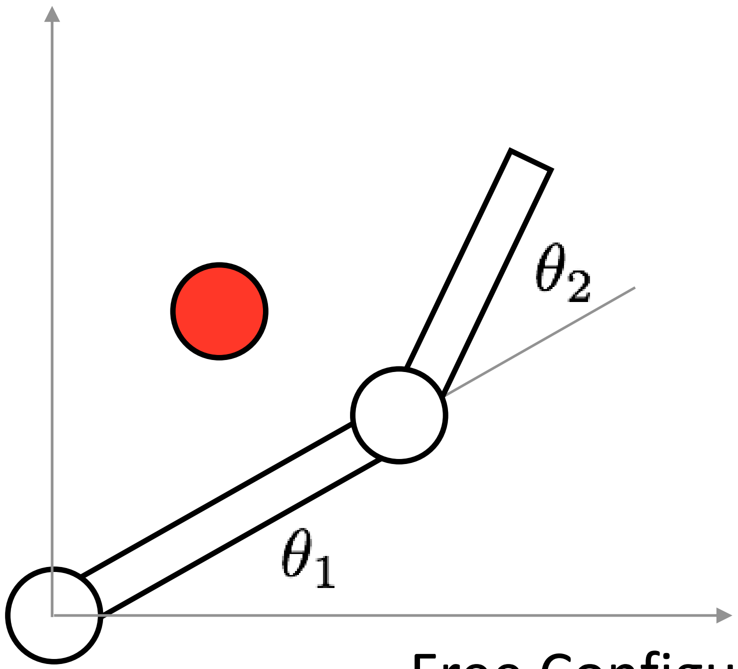
Path Planning is a **search**: BFS, Dijkstra, A*

Last Time: C-Space Obstacles

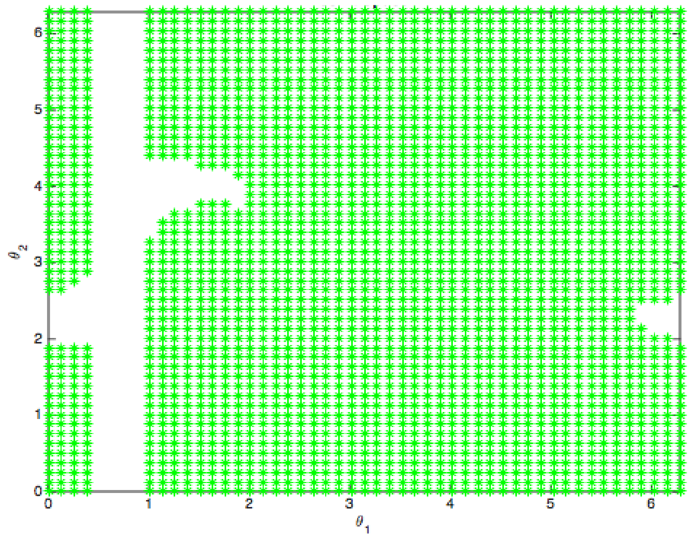




Obstacle

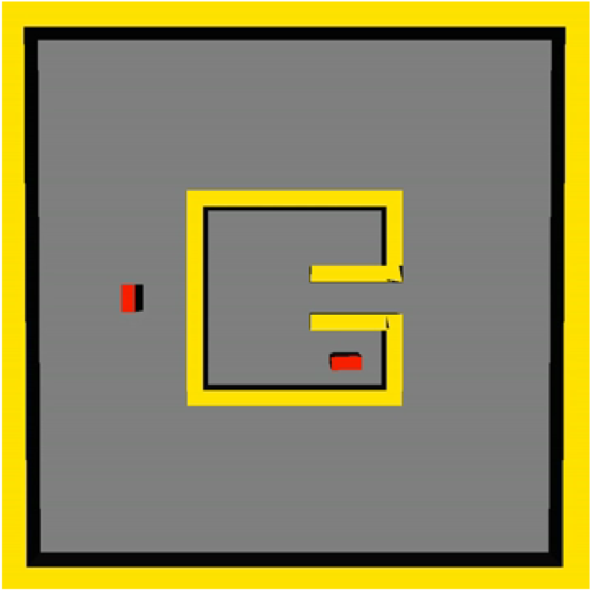


Free Configuration Space

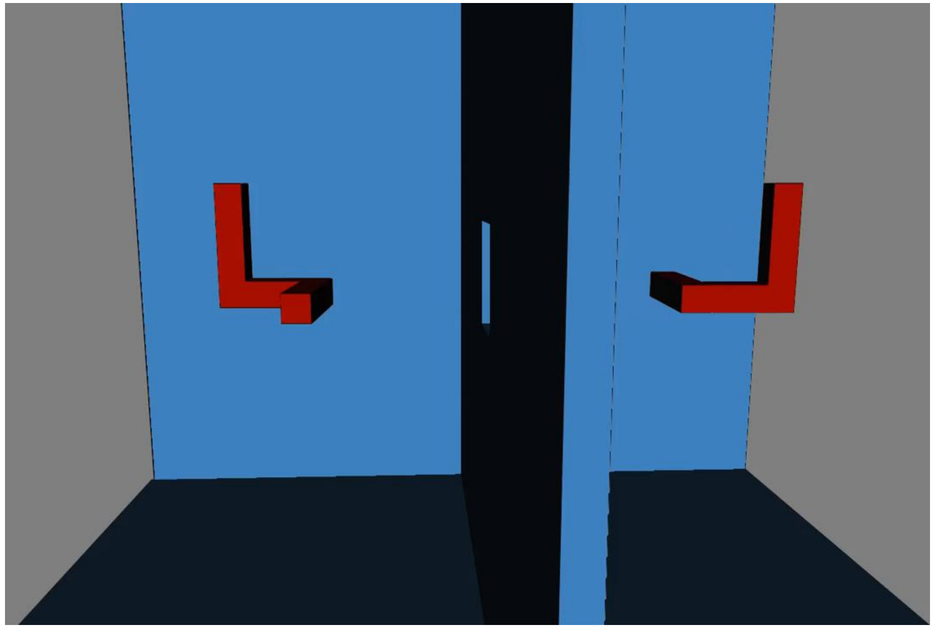


What makes planning hard?

https://www.youtube.com/watch?v=UTbiAu8IXas



https://vimeo.com/58686591



https://vimeo.com/58709589

Complex obstacles
Narrow corridors in the free C-space

CHALLENGE: Map out the free C-Space

Planning strategy

1. Convert your free C-space into a graph/roadmap Hard
2. Find a path from q_{start} to a node q_a that is in the roadmap Use Lecture 10
3. Find a path from q_{goal} to a node q_b that is in the roadmap Use Lecture 10
4. Search the roadmap for a path from q_a to q_b Use Lecture 11

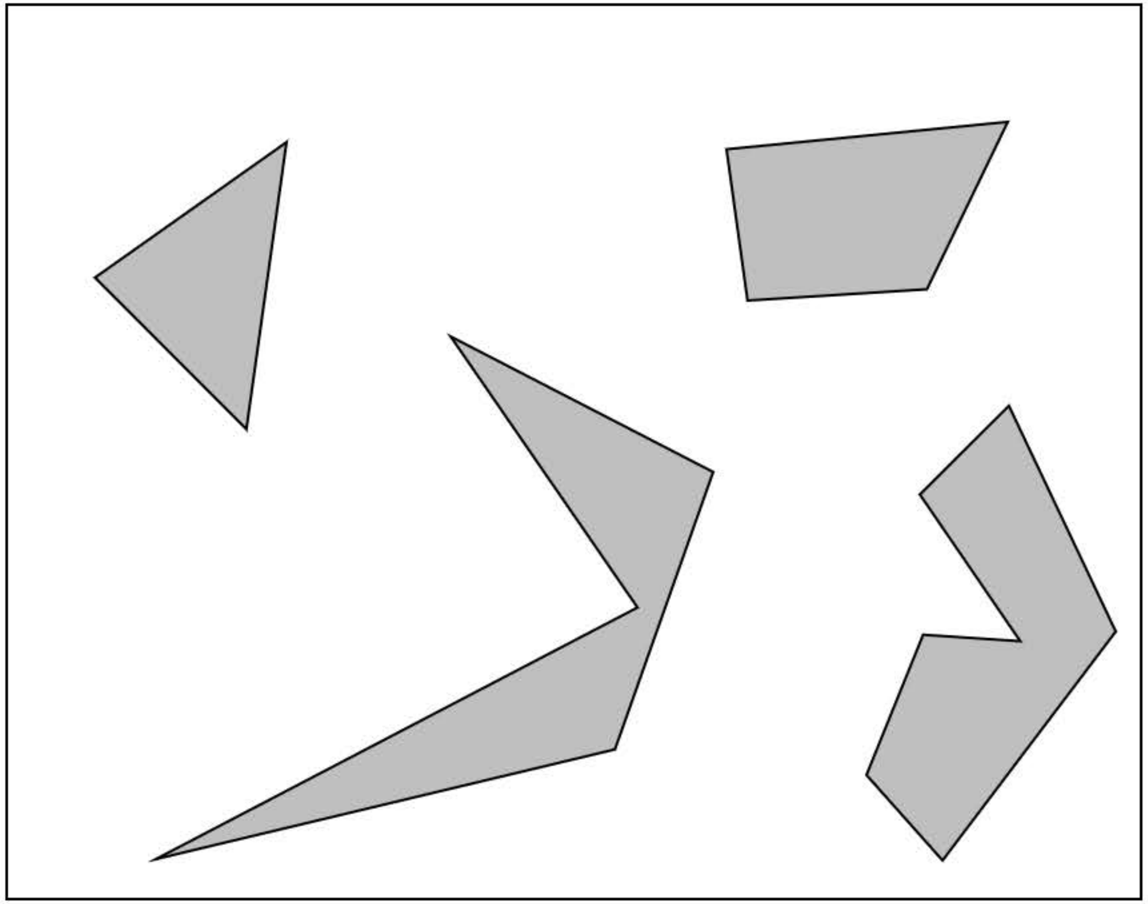
Probabilistic planners

Build a map of the free C-space using **sampling**

Are useful when it is difficult to describe the free C-space but easy to describe configurations in collision

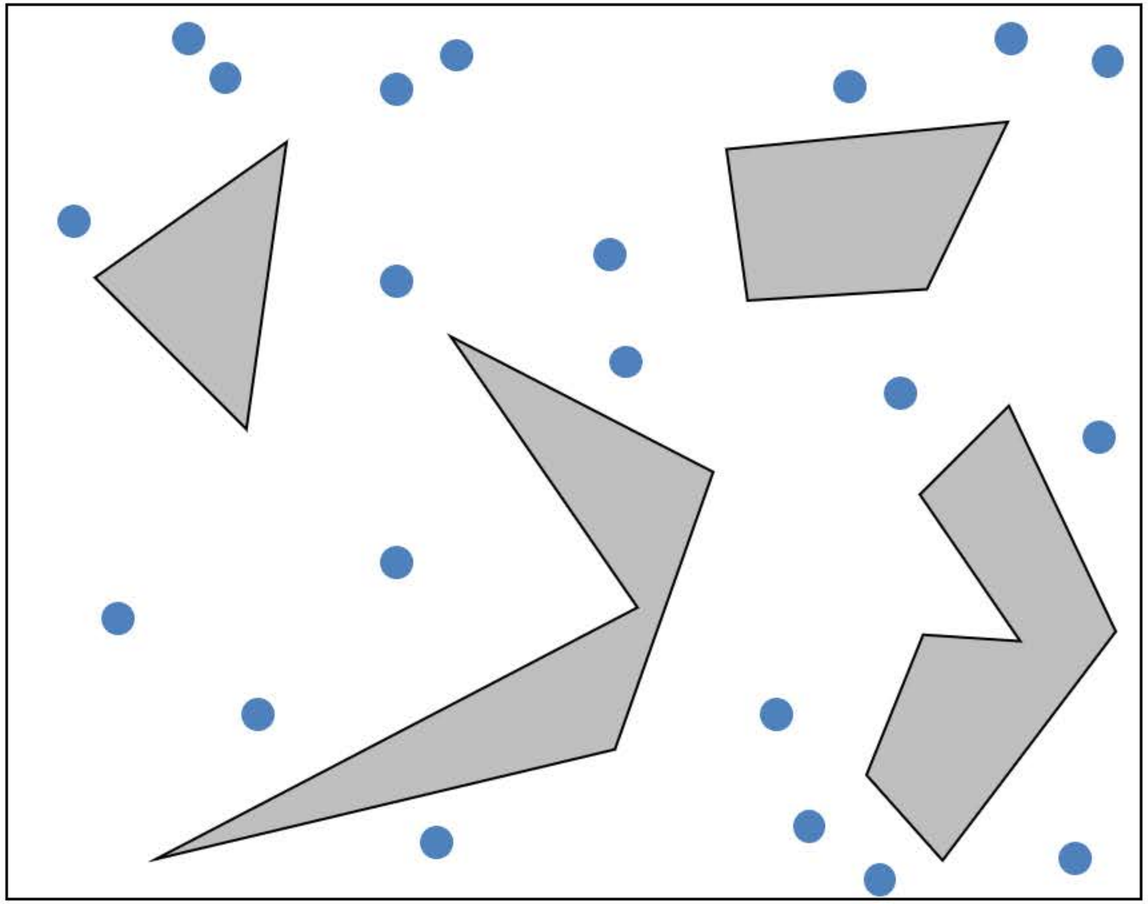
Are **probabilistically complete**

Probabilistic Roadmaps



Pseudocode:

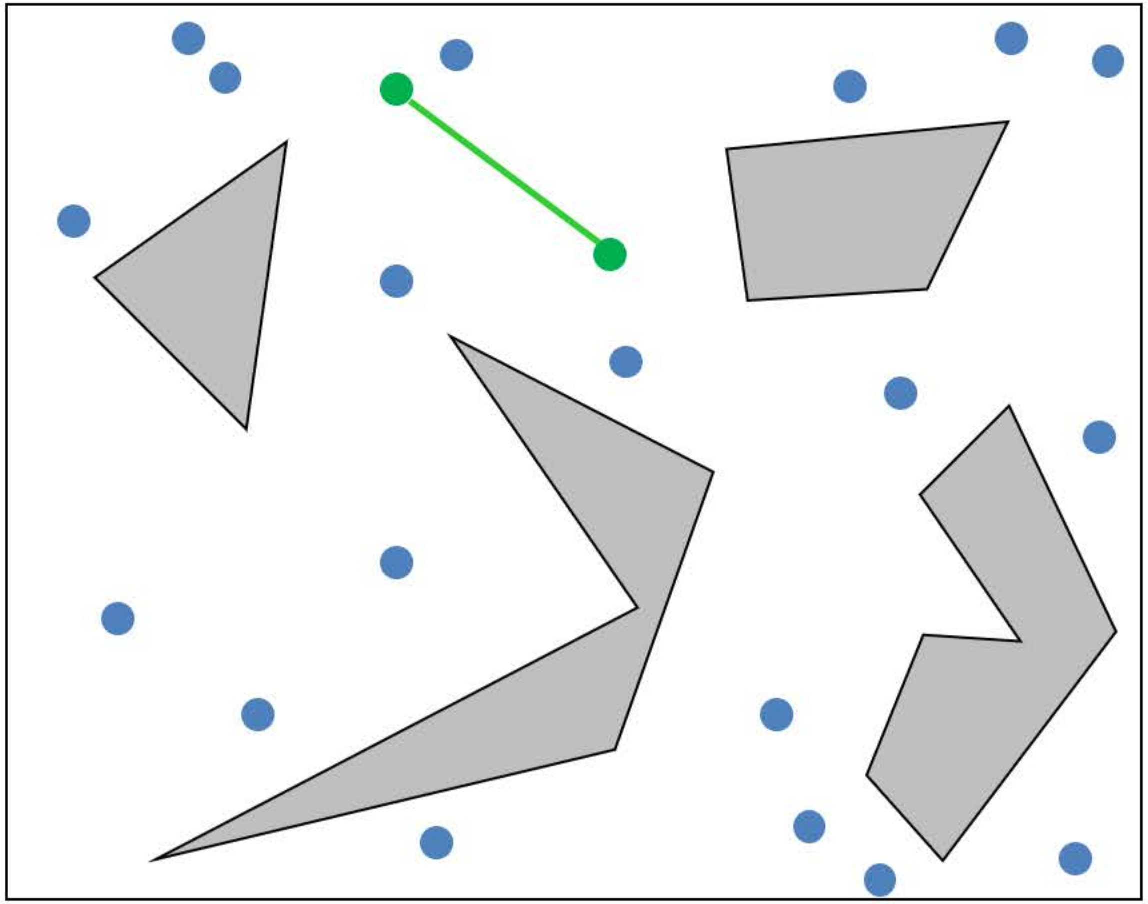
Probabilistic Roadmaps



Pseudocode:

V = Sample(n); E = {};

Probabilistic Roadmaps



Pseudocode:

V = Sample(n); E = {};

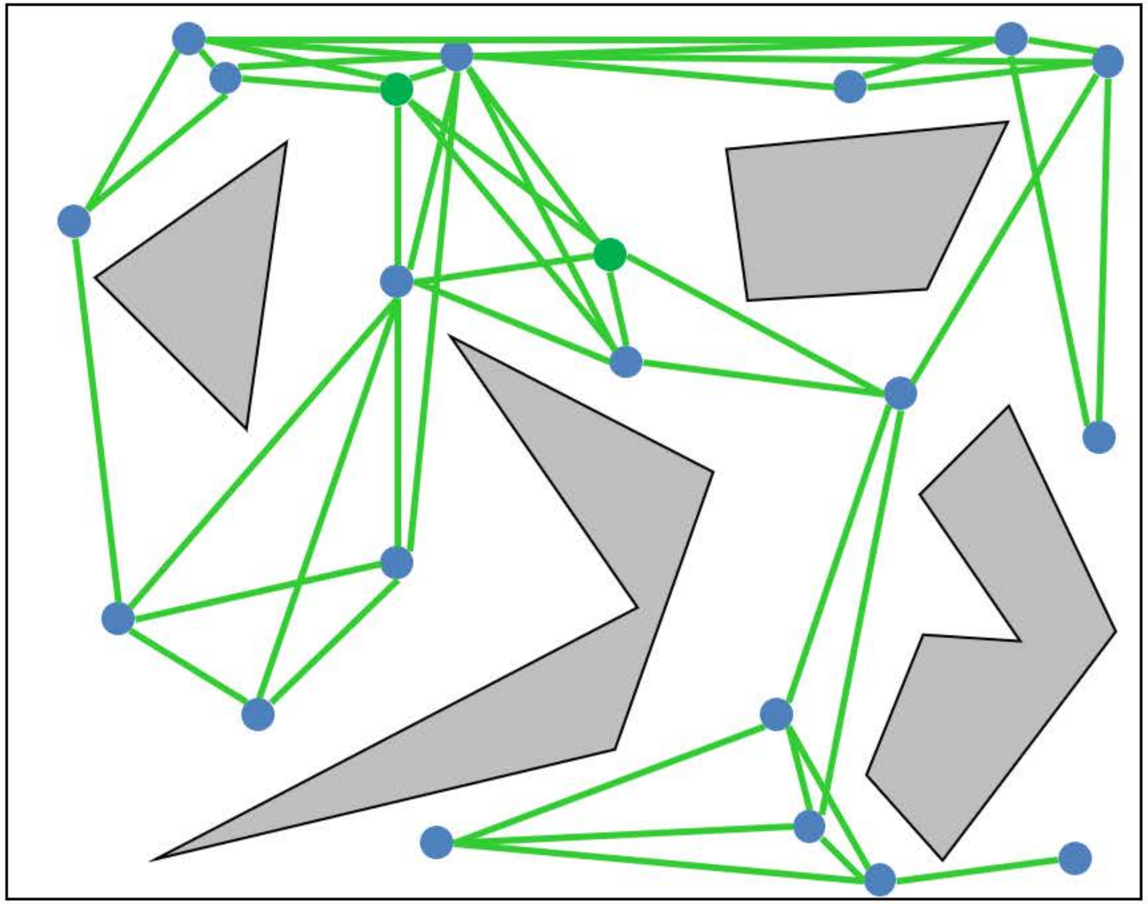
For all q ∈ V

  For all q' ∈ V\q

    If NOT collide(qq')

      E = E ∪ {(q,q')}

Probabilistic Roadmaps



Pseudocode:

V = Sample(n); E = {};

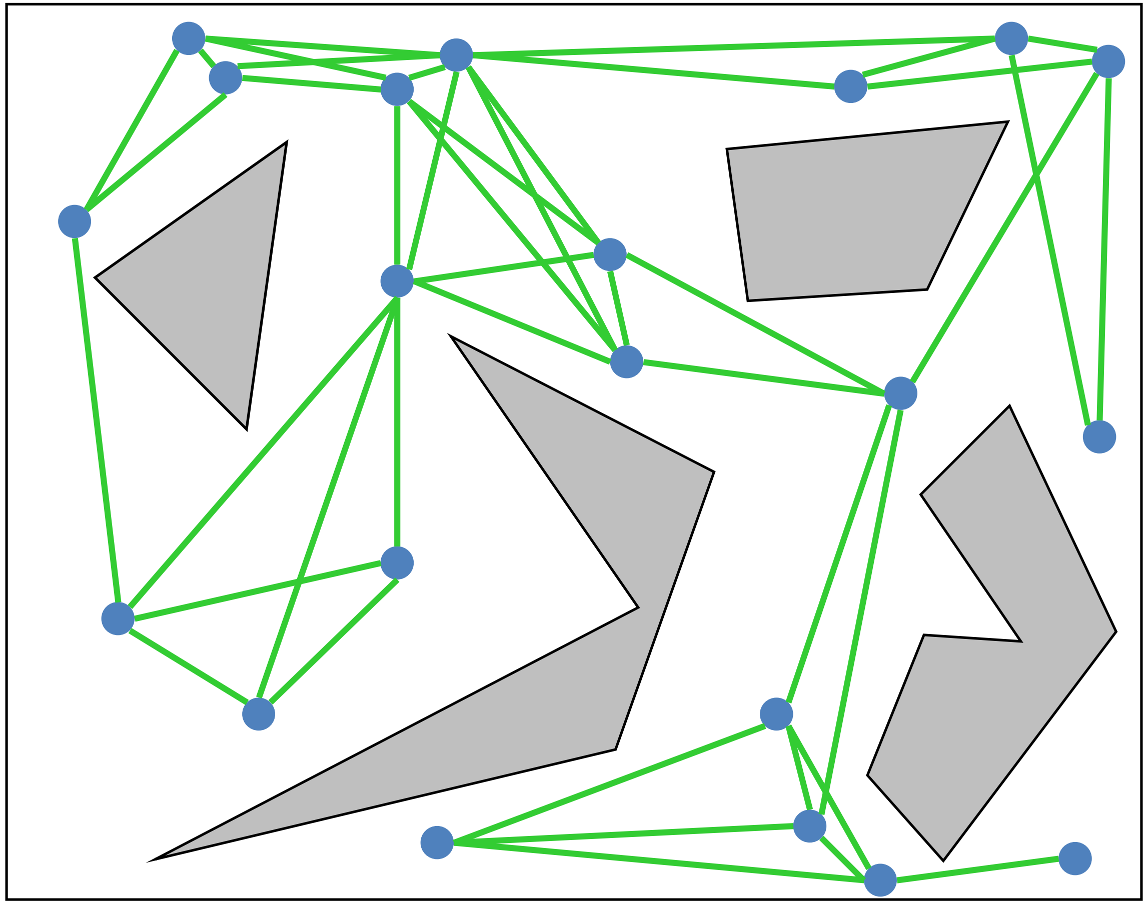
For all q ∈ V

  For all q' ∈ V\q

    If NOT collide(qq')

      E = E ∪ {(q,q')}

Probabilistic Roadmaps



Pseudocode:

V = Sample(n); E = {};

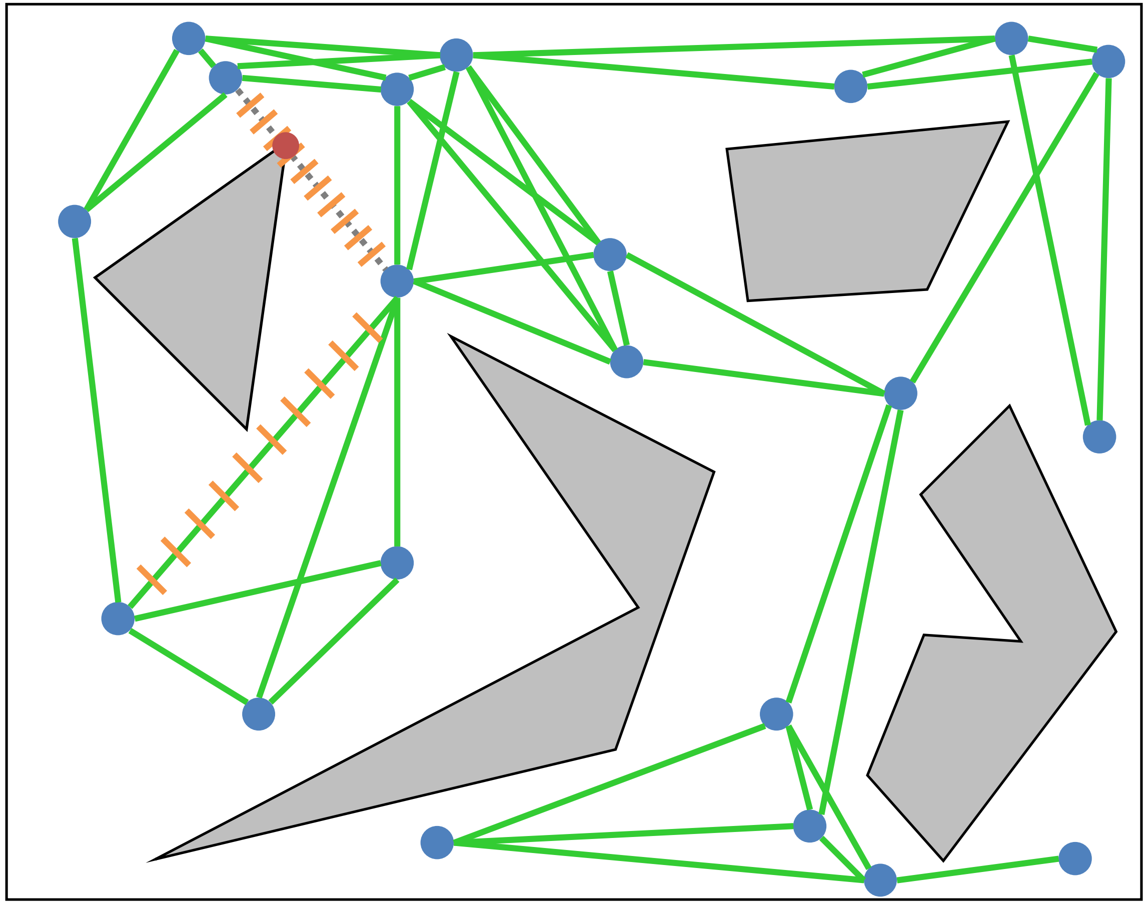
For all q ∈ V

| **For** all q' ∈ ~~V\q~~ $N_k(q)$

| | **If** NOT collide(qq')

| | | E = E ∪ {(q,q')}

Probabilistic Roadmaps



Pseudocode:

V = Sample(n); E = {};

For all q ∈ V

  For all q' ∈ ~~V\q~~ N_k(q)

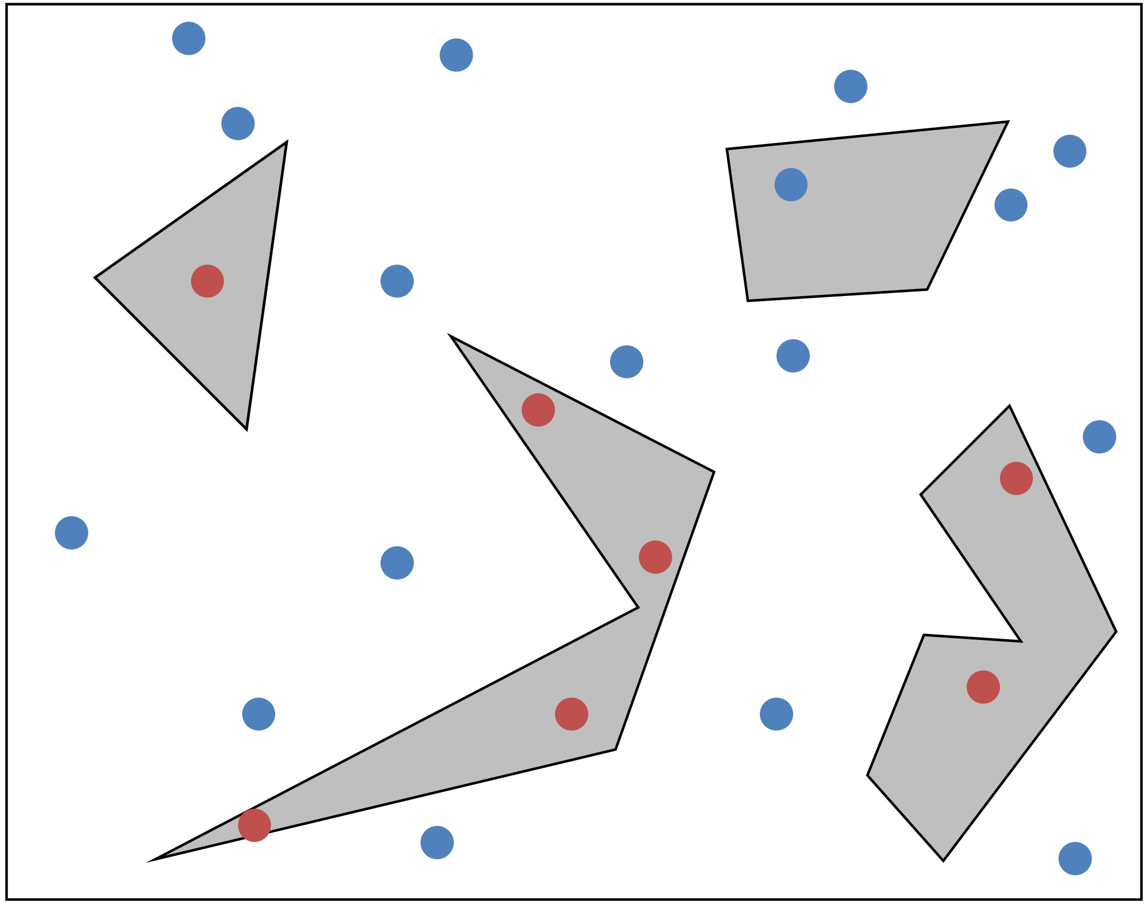
    If NOT collide(qq')

      E = E ∪ {(q,q')}

Lazy PRM: check collisions only when needed during search

Sampling Strategy

Q for you: How to uniformly sample orientations?



Basic PRM: uniform sampling

Sample(n):

V = {}

While |V|<n

Repeat

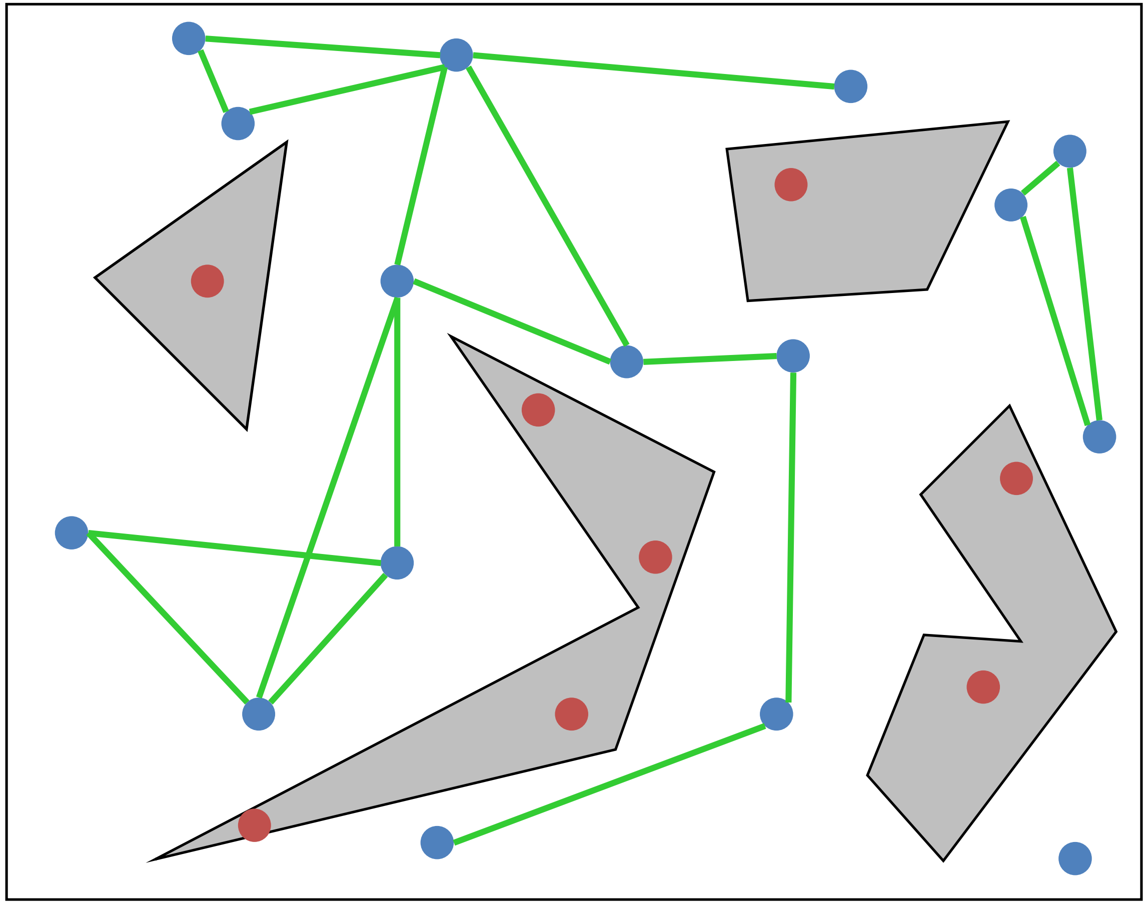
q = random configuration in Q

Until q is collision-free

V = V ∪ {q}

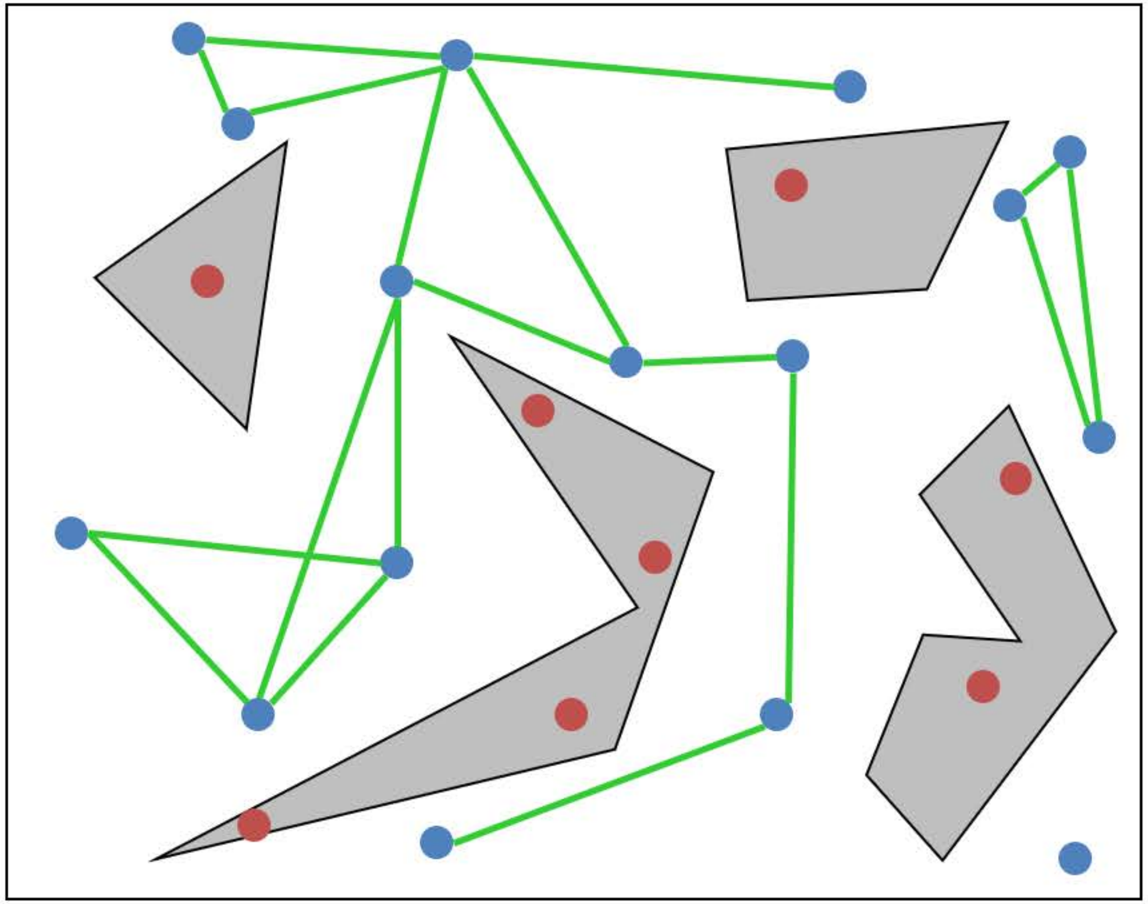
Produces a uniformly distributed sampling of the free space

Sampling Strategy



Success depends on n

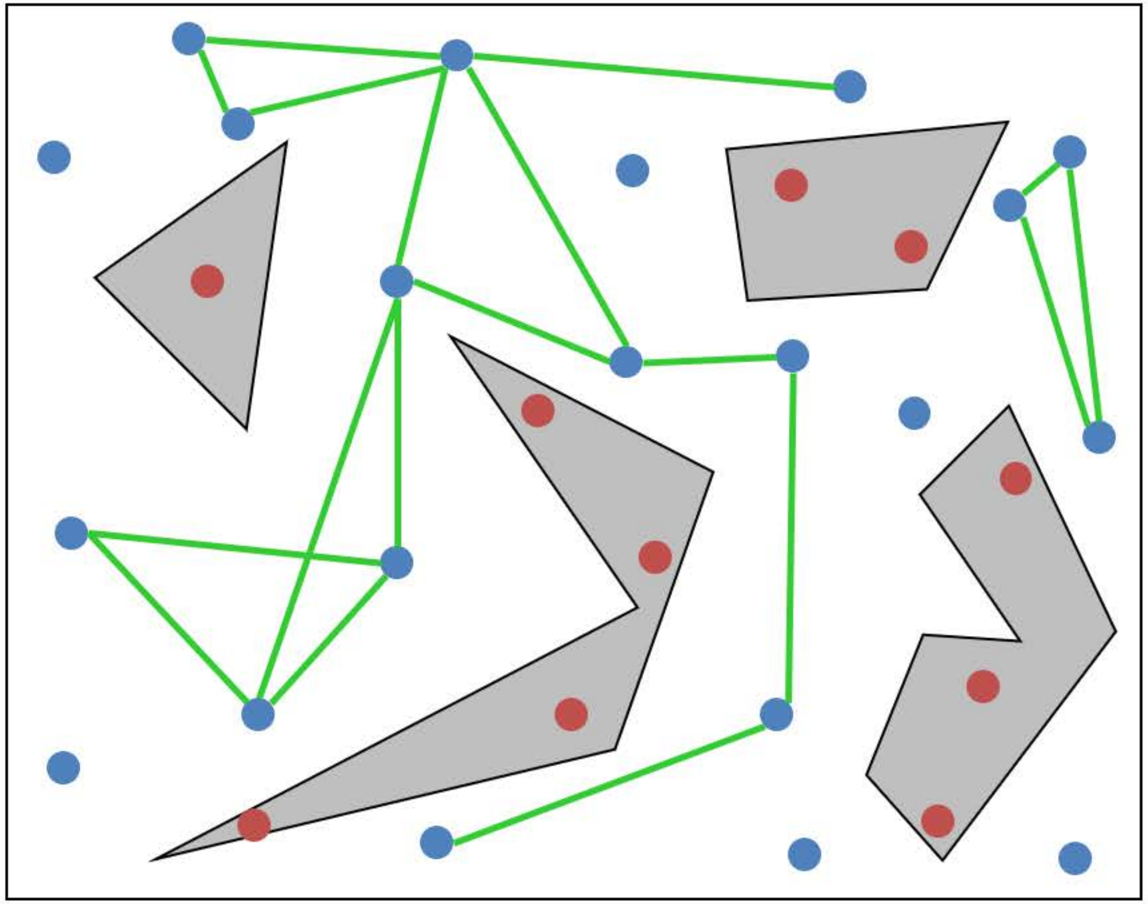
Sampling Strategy: Enhancement



Success depends on n

Add more random nodes

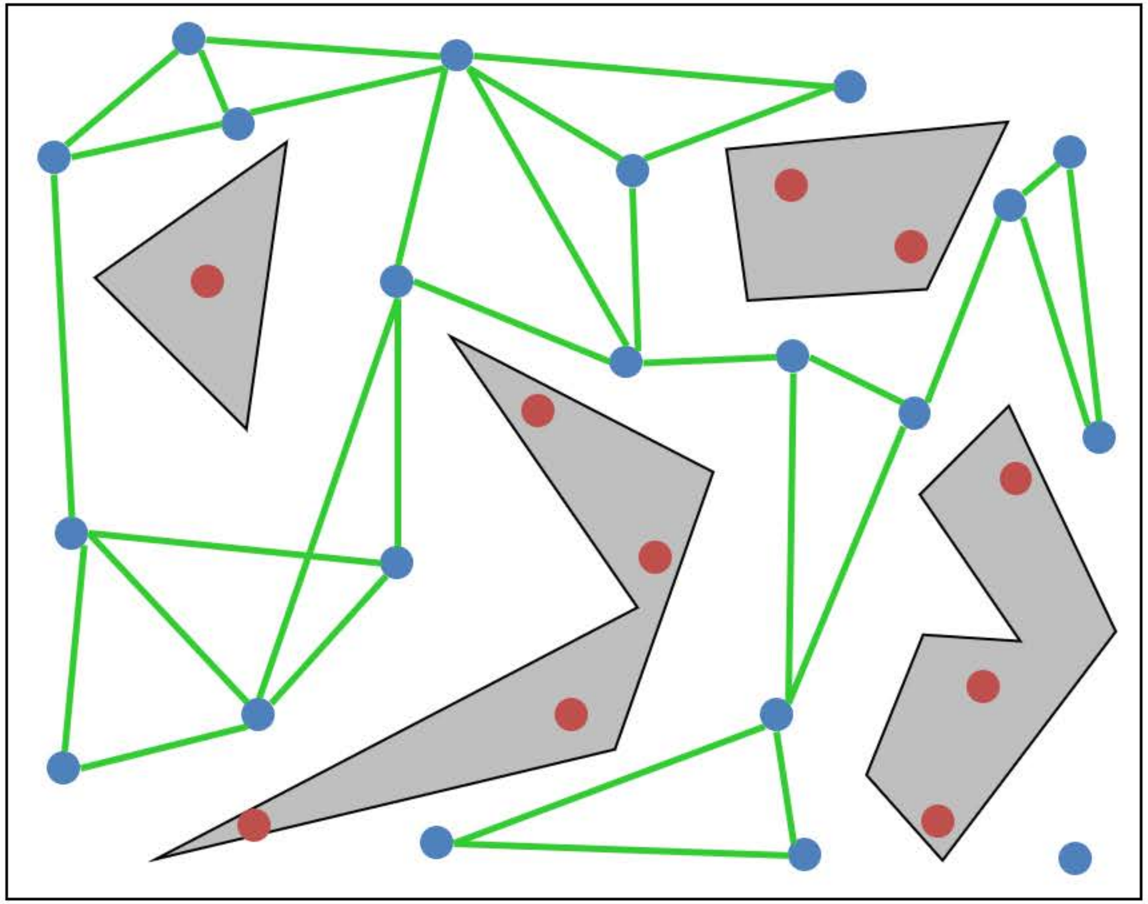
Sampling Strategy: Enhancement



Success depends on n

Add more random nodes

Sampling Strategy: Enhancement

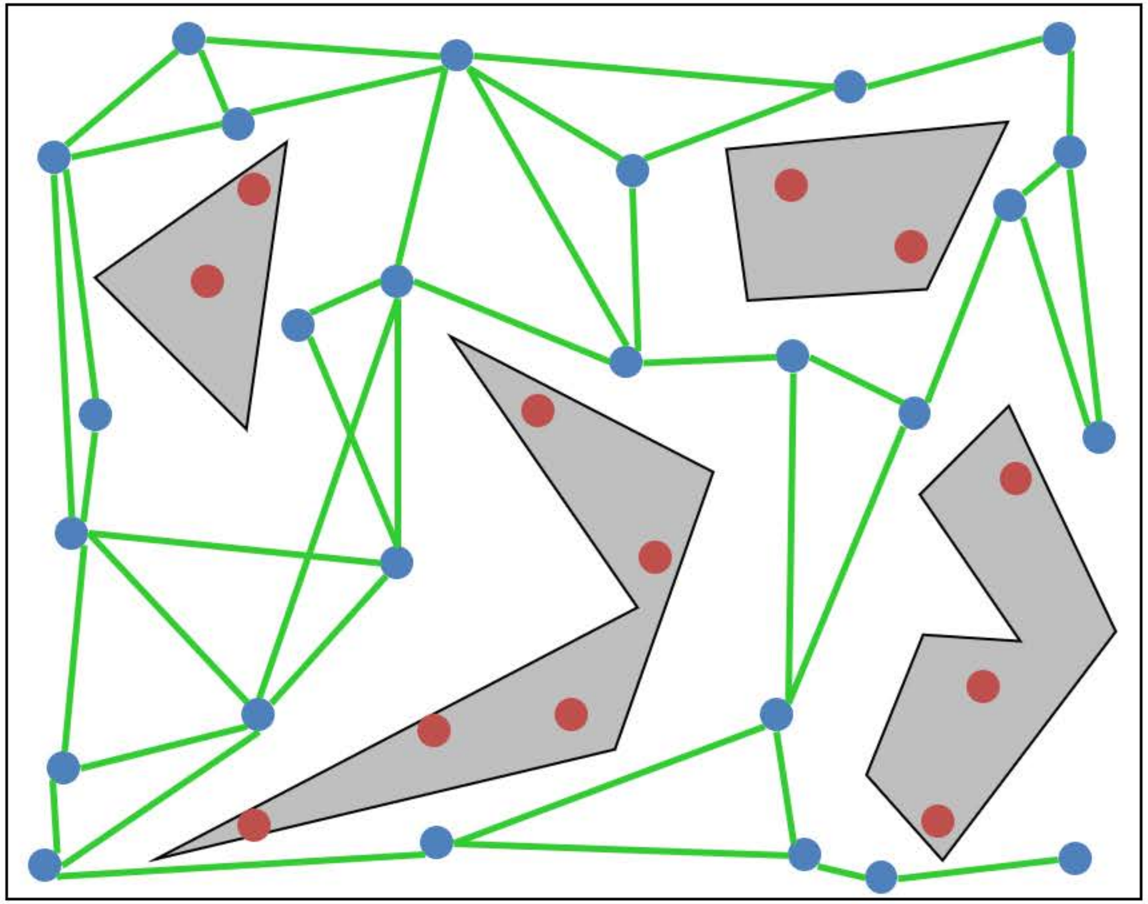


Success depends on n

Add more random nodes

Connect them to the existing roadmap

Sampling Strategy: Enhancement

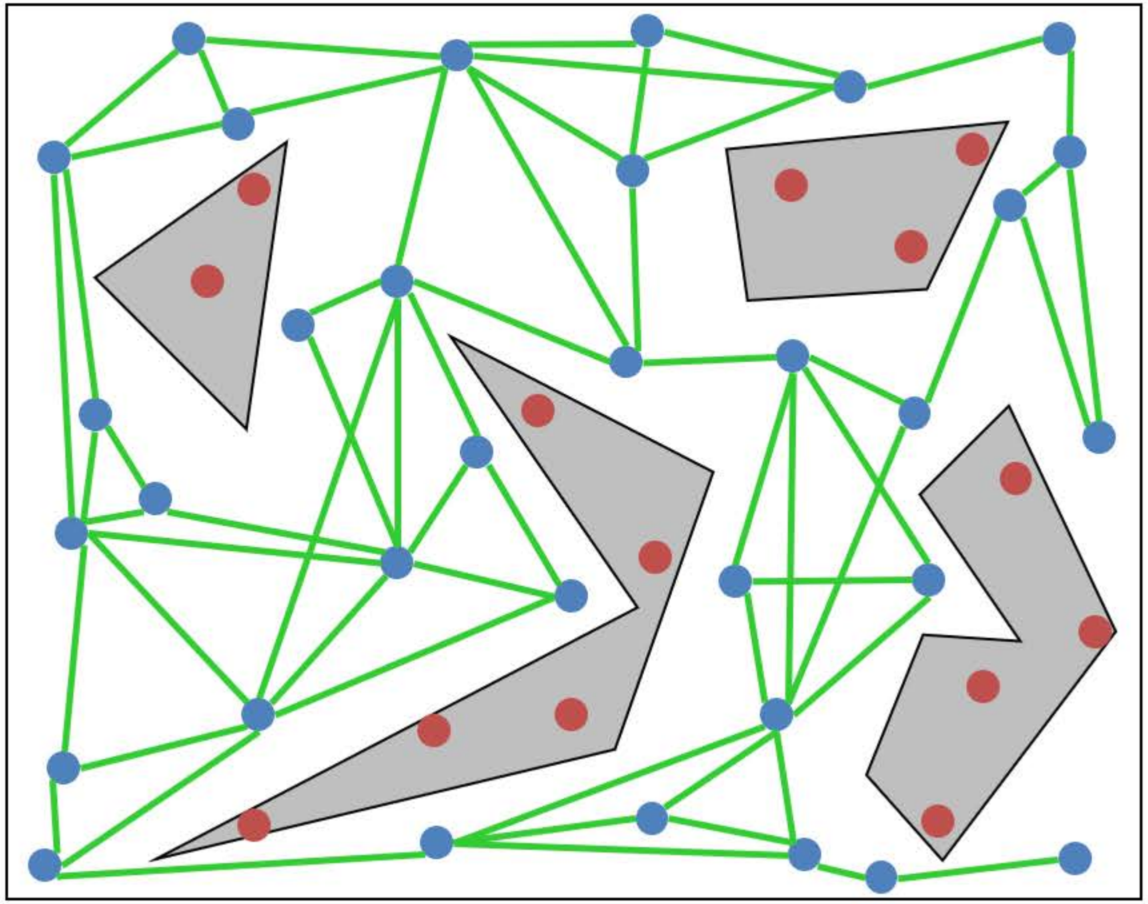


Success depends on n

Add more random nodes

Connect them to the existing roadmap

Sampling Strategy: Enhancement

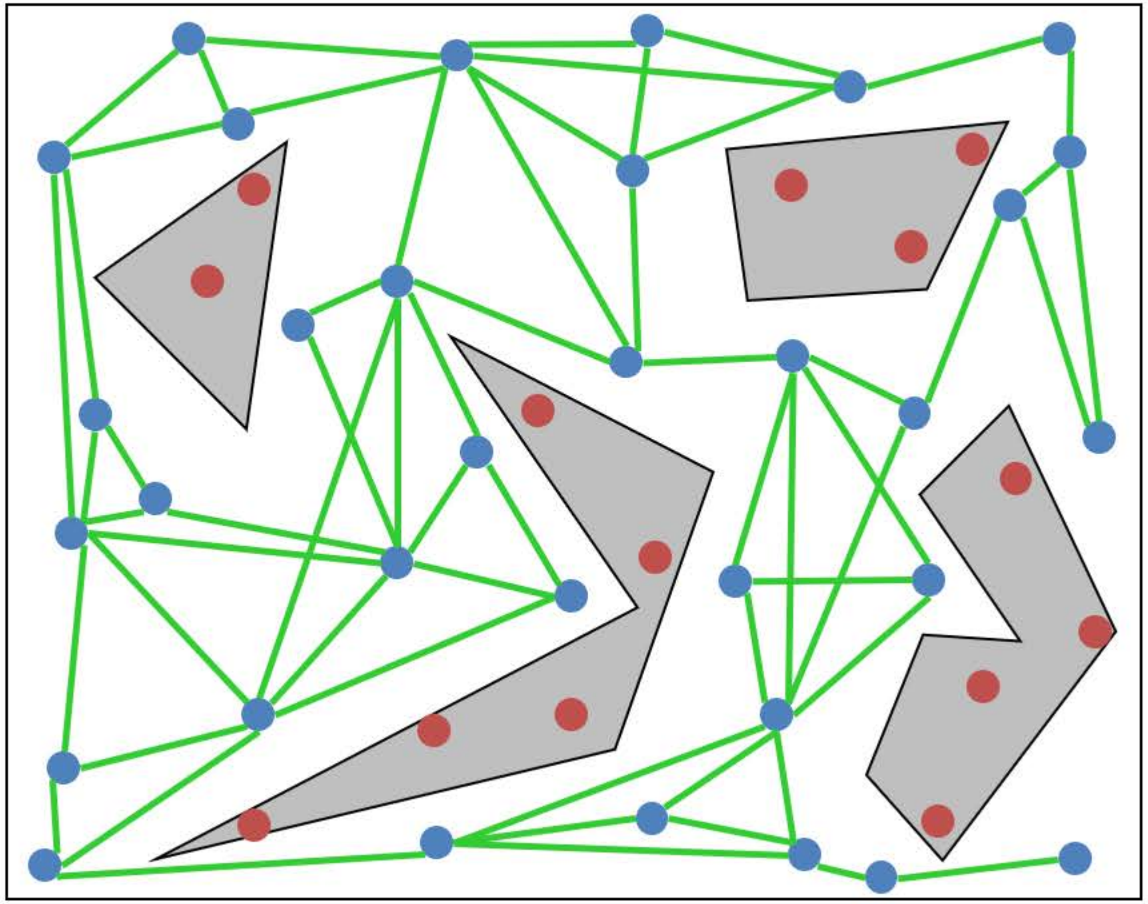


Success depends on n

Add more random nodes

Connect them to the existing roadmap

Sampling Strategy: Enhancement



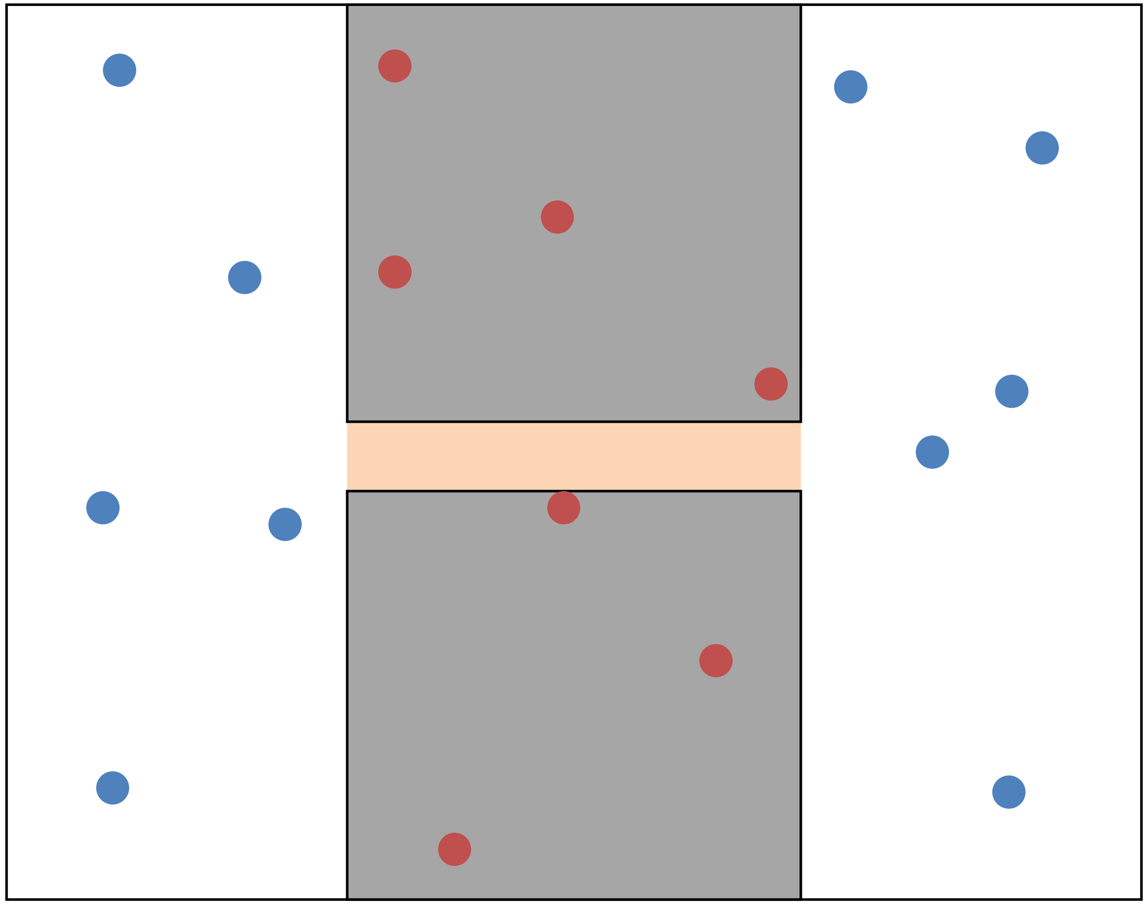
Success depends on n

Add more random nodes

Connect them to the existing roadmap

Probabilistic completeness: if a path exists, P(success)→1 as $n \to \infty$

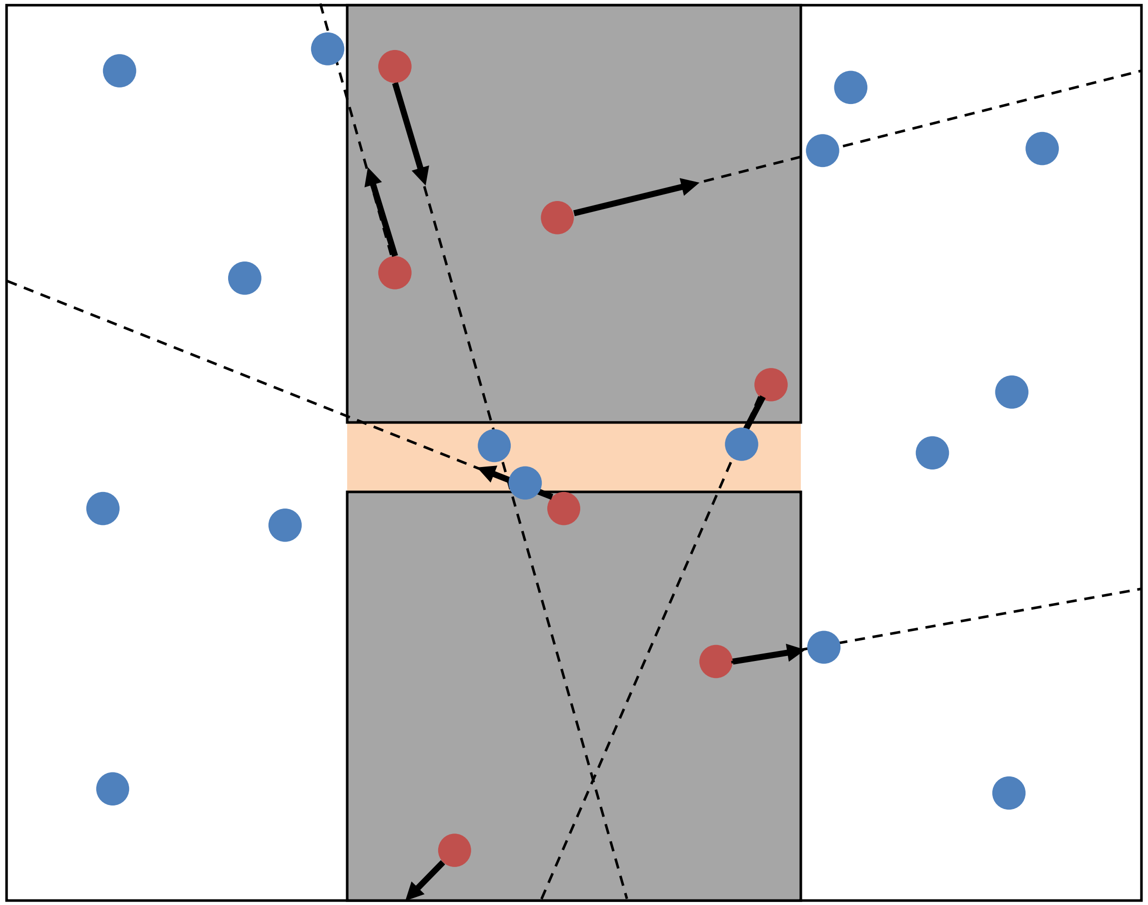
Sampling Strategy



#{samples} in a subset of C-space ~prop to volume of subset

Narrow-passage problem: unlikely to have samples in a passage

Sampling Strategy



#{samples} in a subset of C-space ~prop to volume of subset

Narrow-passage problem: unlikely to have samples in a passage

Strategy: Sample near obstacles

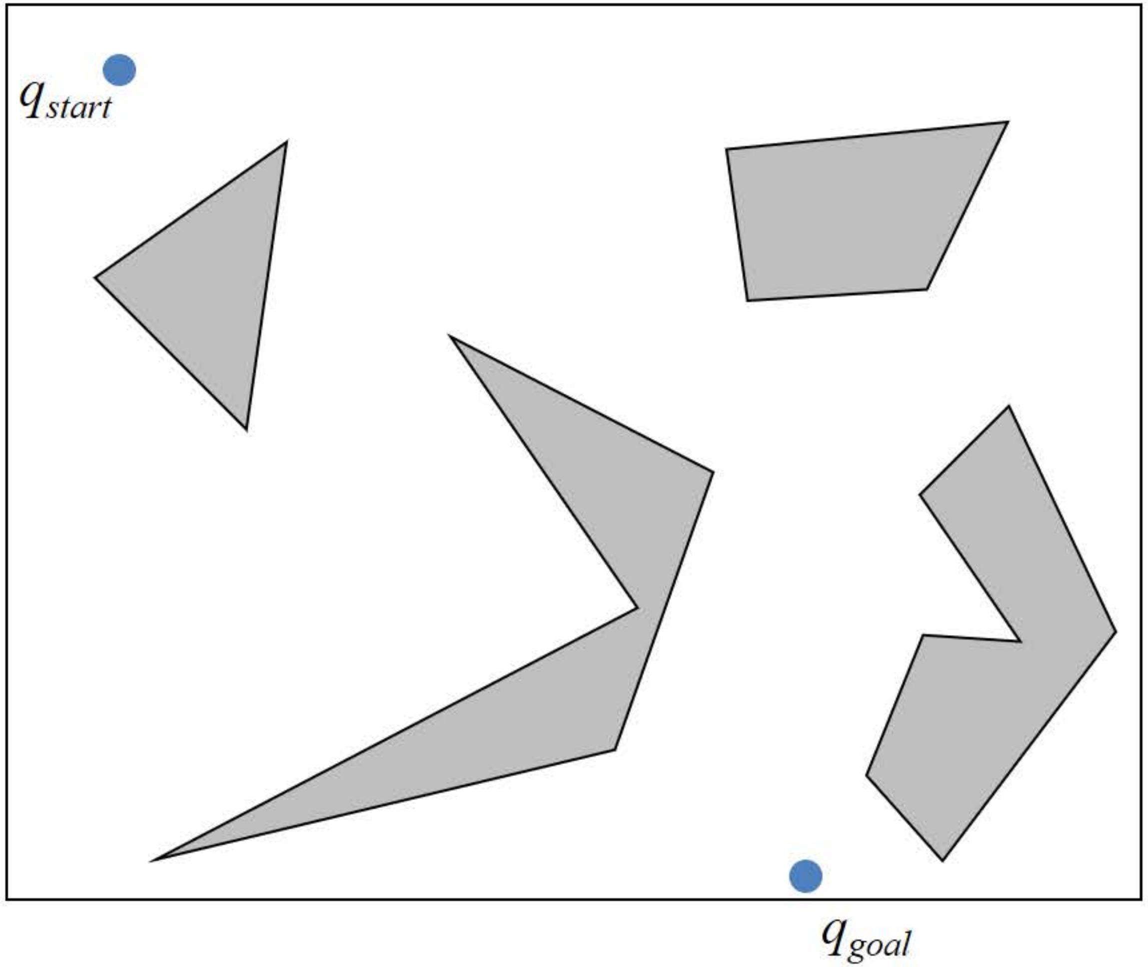
PRM is a multi-query planner.

The goal is to create a roadmap of the free C-space and perform multiple searches very fast

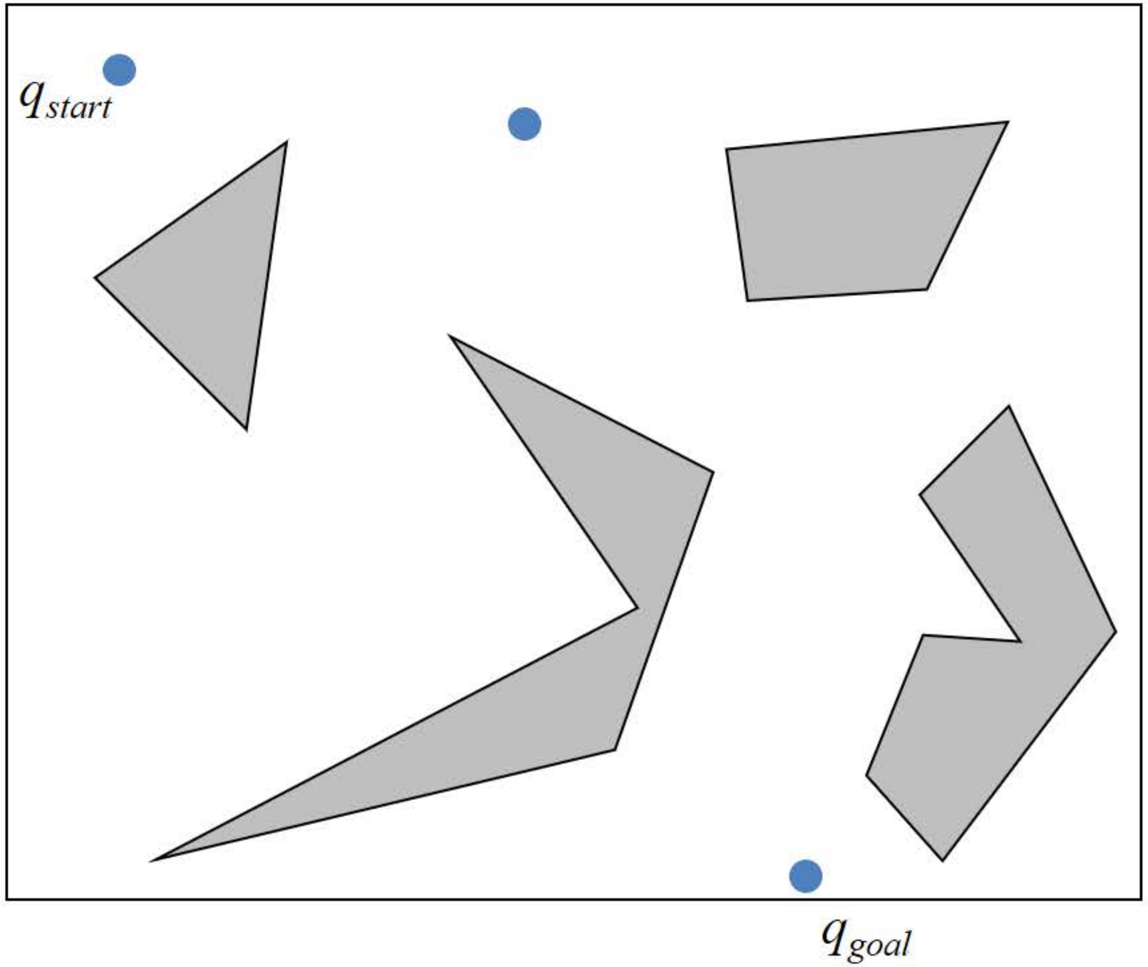
Rapidly-exploring Random Trees (RRTs) build the roadmap incrementally for single-query search

Rapidly-exploring Random Trees (RRTs)

$T_{start} = (q_{start}, \emptyset)$, $T_{goal} = (q_{goal}, \emptyset)$



Rapidly-exploring Random Trees (RRTs)

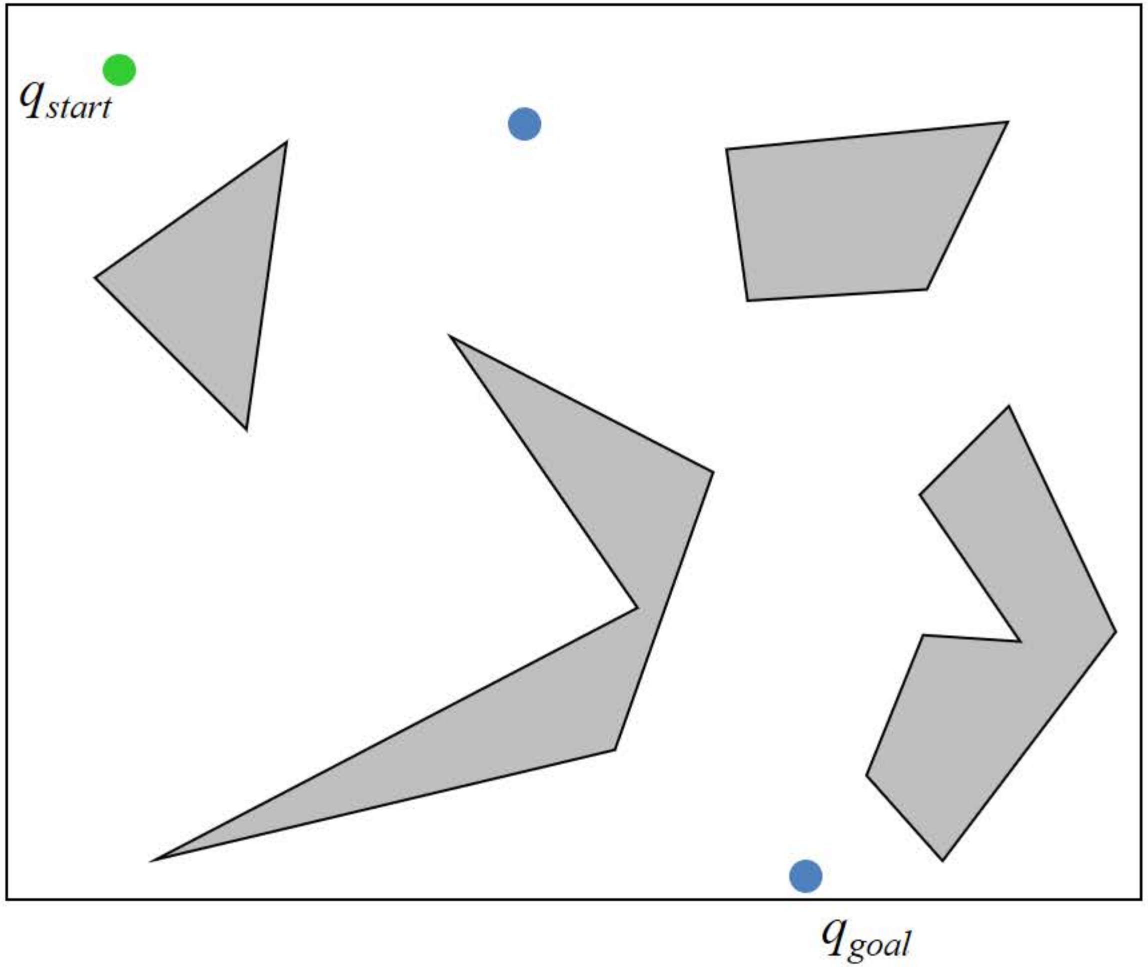


$T_{start} = (q_{start}, \emptyset)$, $T_{goal} = (q_{goal}, \emptyset)$

For i = 1 to n_{iter}

q = random configuration in Q_{free}

Rapidly-exploring Random Trees (RRTs)

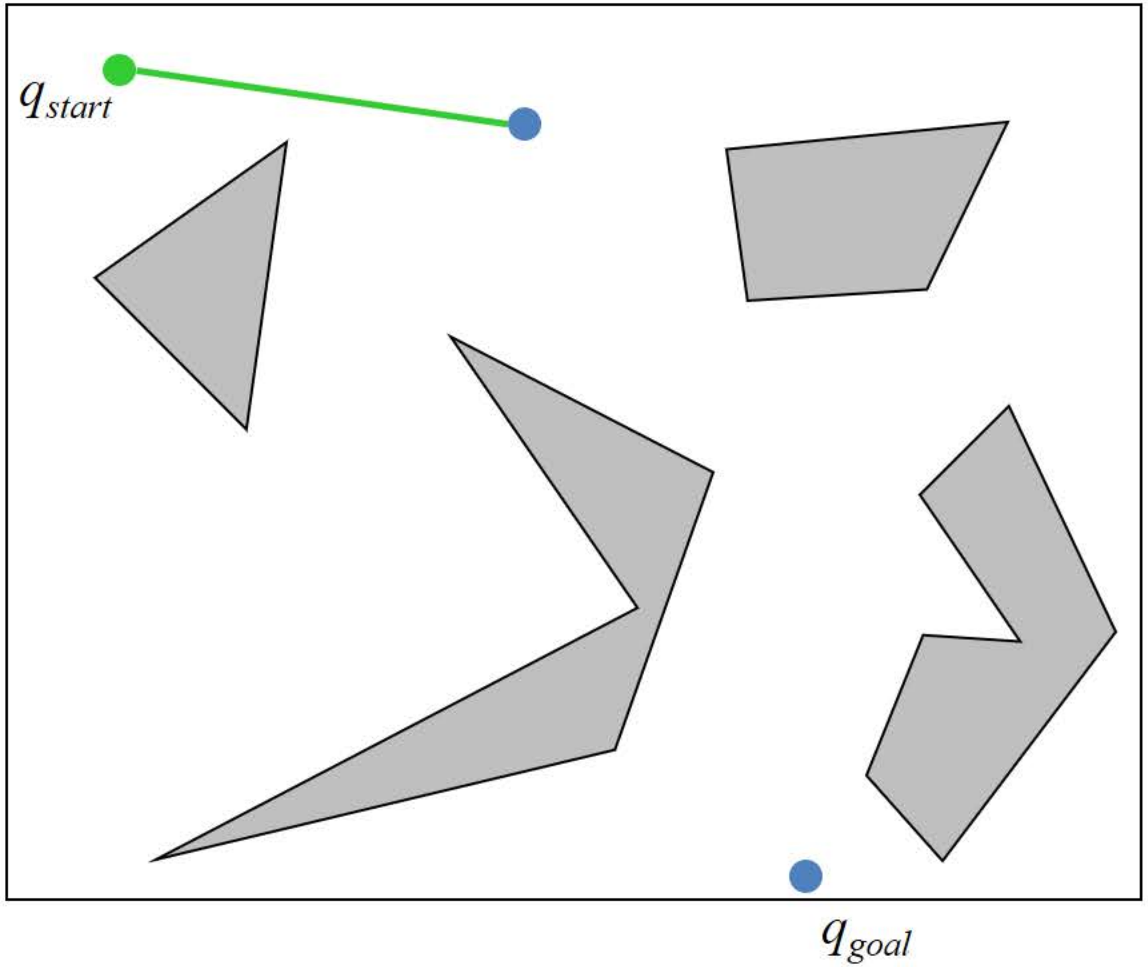


$T_{start} = (q_{start}, \emptyset)$, $T_{goal} = (q_{goal}, \emptyset)$

For $i = 1$ to n_{iter}

- q = random configuration in Q_{free}
- q_a = closest node in T_{start}

Rapidly-exploring Random Trees (RRTs)

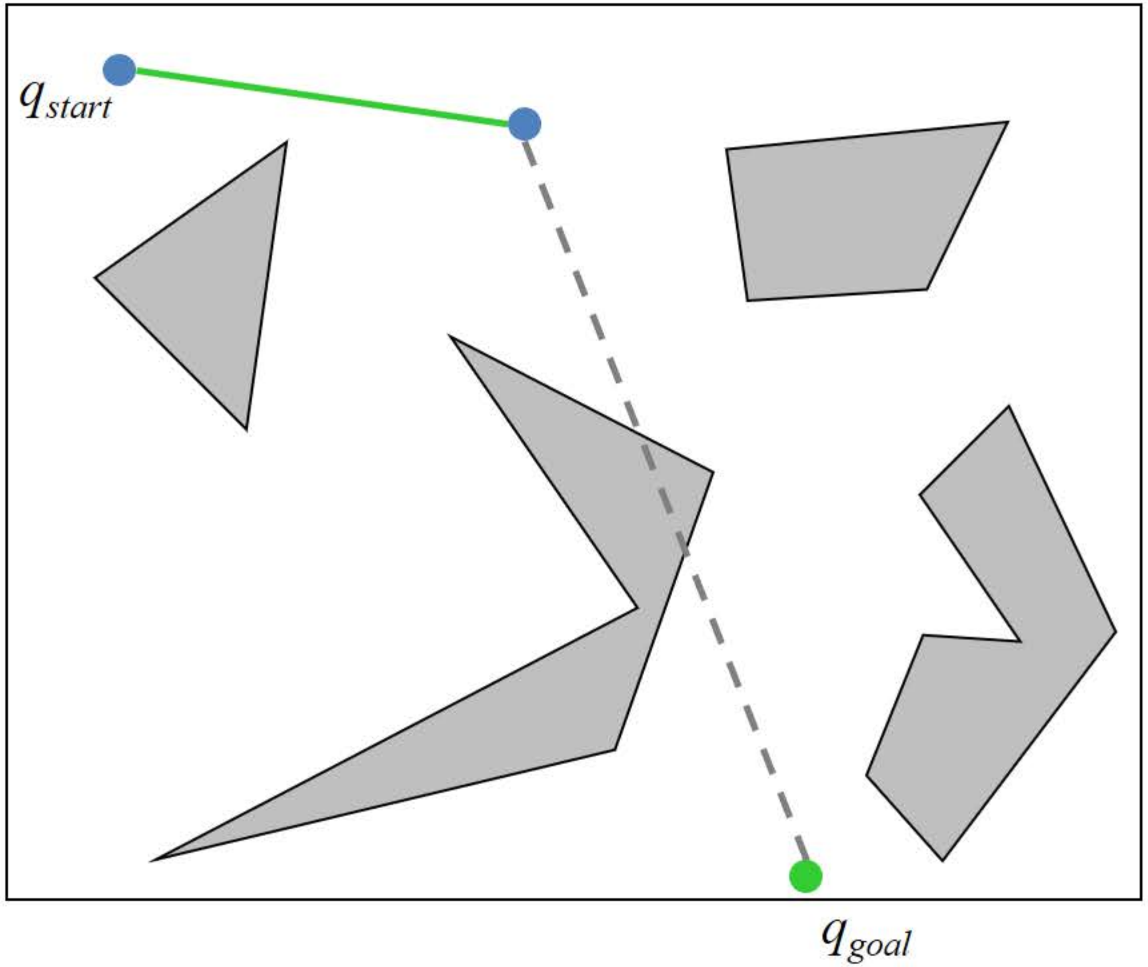


$T_{start} = (q_{start}, \emptyset)$, $T_{goal} = (q_{goal}, \emptyset)$

For $i = 1$ to n_{iter}

- q = random configuration in Q_{free}
- q_a = closest node in T_{start}
- **If** NOT collide(qq_a)
  - Add (q, q_a) to T_{start}

Rapidly-exploring Random Trees (RRTs)

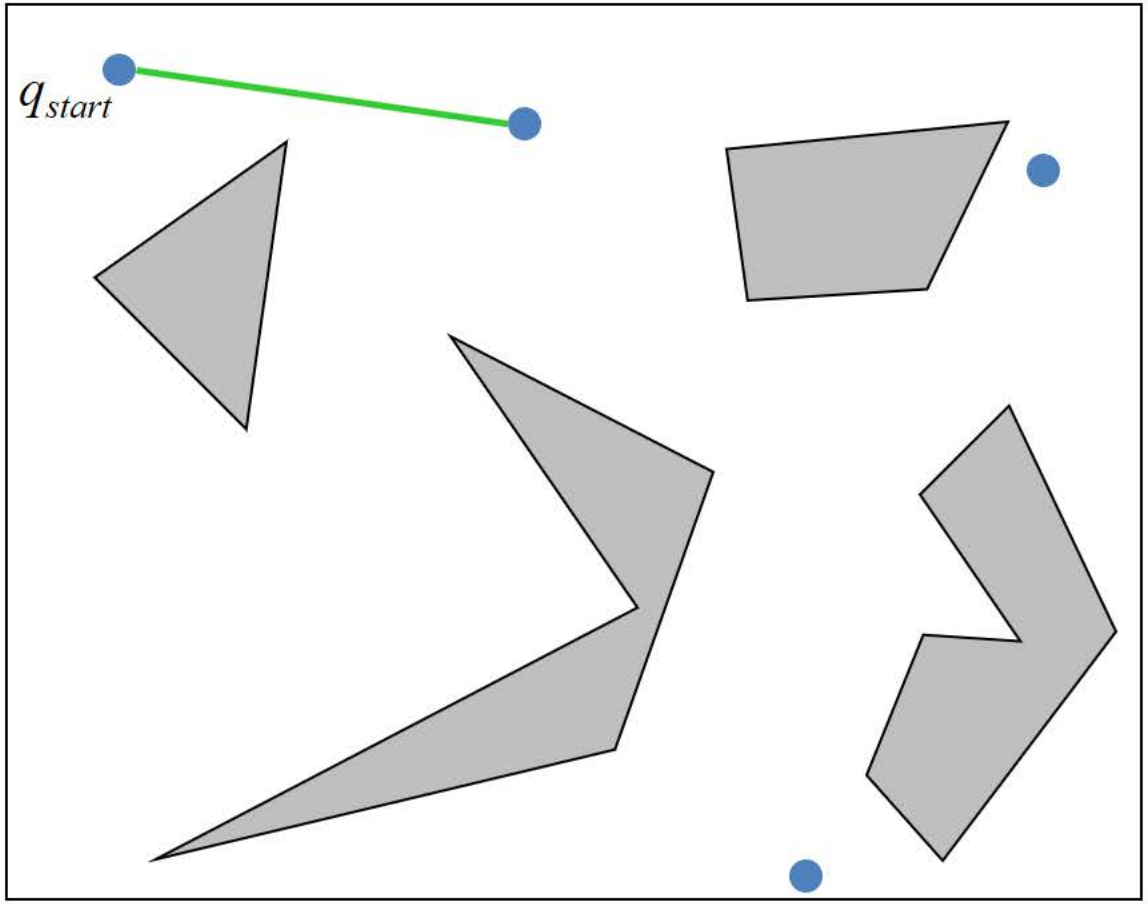


$T_{start} = (q_{start}, \emptyset)$, $T_{goal} = (q_{goal}, \emptyset)$

For i = 1 to n_{iter}

- q = random configuration in Q_{free}
- q_a = closest node in T_{start}
- **If** NOT collide(qq_a)
  - Add (q,q_a) to T_{start}
- q_b = closest node in T_{goal}
- **If** NOT collide(qq_b)

Rapidly-exploring Random Trees (RRTs)

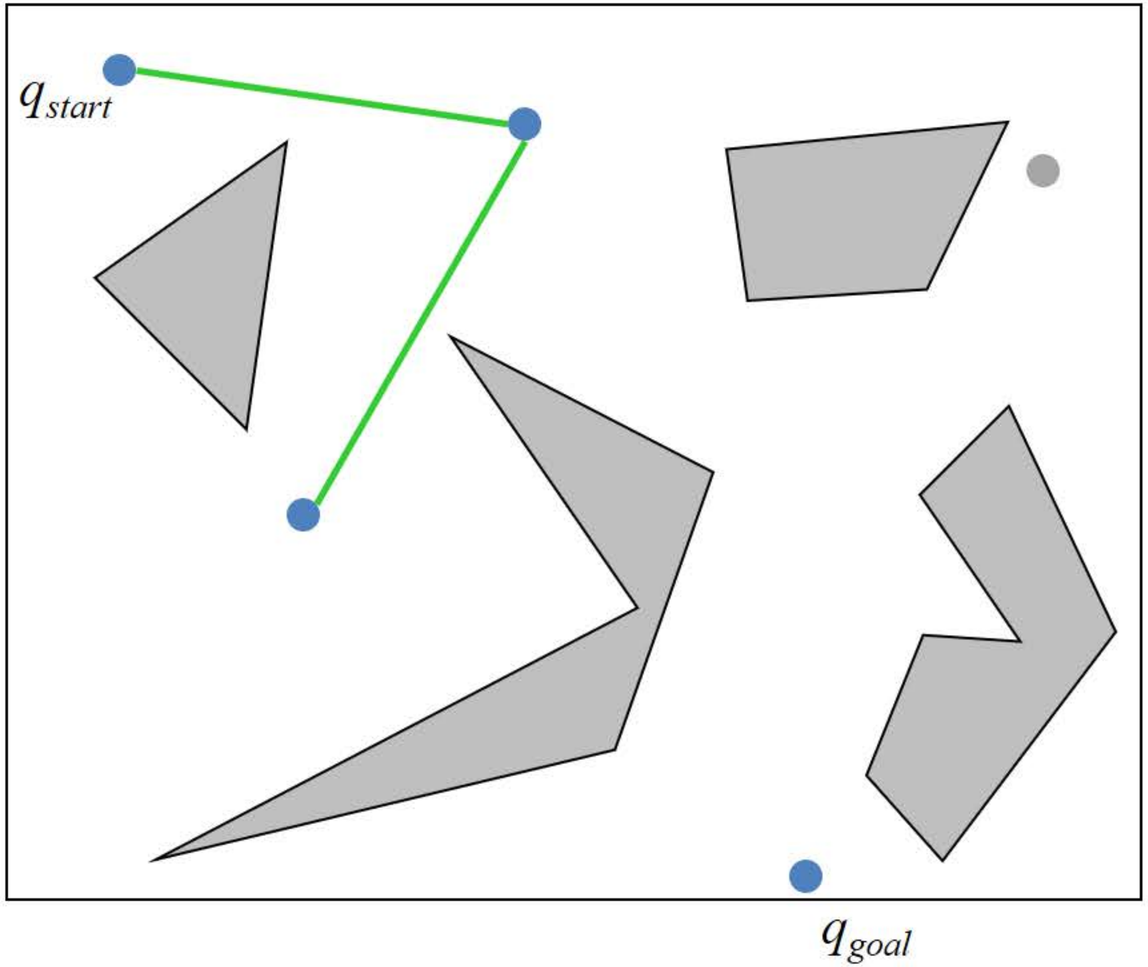


$T_{start} = (q_{start}, \emptyset)$, $T_{goal} = (q_{goal}, \emptyset)$

For i = 1 to n_{iter}

- q = random configuration in Q_{free}
- q_a = closest node in T_{start}
- **If** NOT collide(qq_a)
  - Add (q, q_a) to T_{start}
- q_b = closest node in T_{goal}
- **If** NOT collide(qq_b)
  - Add (q, q_b) to T_{goal}
- **If** q connected to T_{start} and T_{goal}
  - break

Rapidly-exploring Random Trees (RRTs)

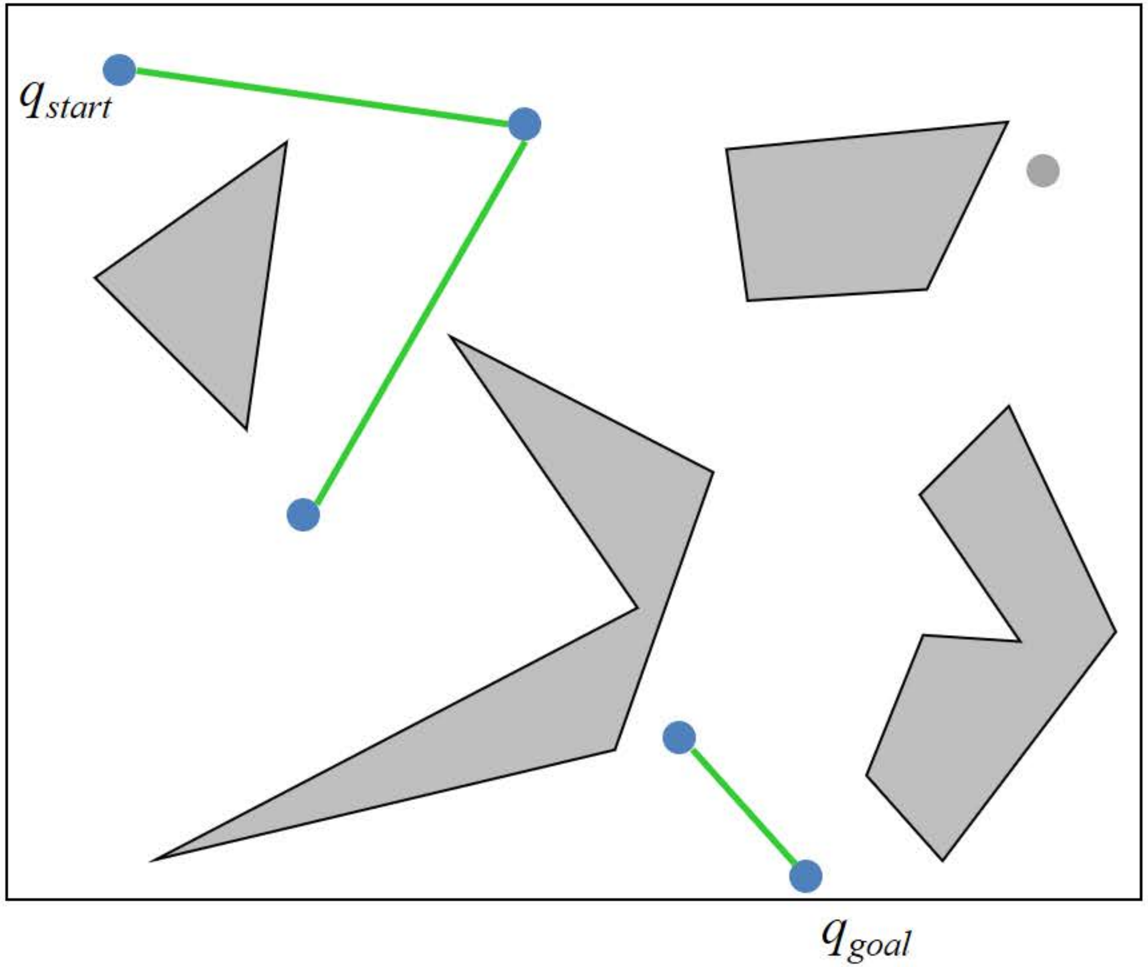


$T_{start} = (q_{start}, \emptyset)$, $T_{goal} = (q_{goal}, \emptyset)$

For i = 1 to n_{iter}

- q = random configuration in Q_{free}
- q_a = closest node in T_{start}
- **If** NOT collide(qq_a)
  - Add (q,q_a) to T_{start}
- q_b = closest node in T_{goal}
- **If** NOT collide(qq_b)
  - Add (q,q_b) to T_{goal}
- **If** q connected to T_{start} and T_{goal}
  - break

Rapidly-exploring Random Trees (RRTs)

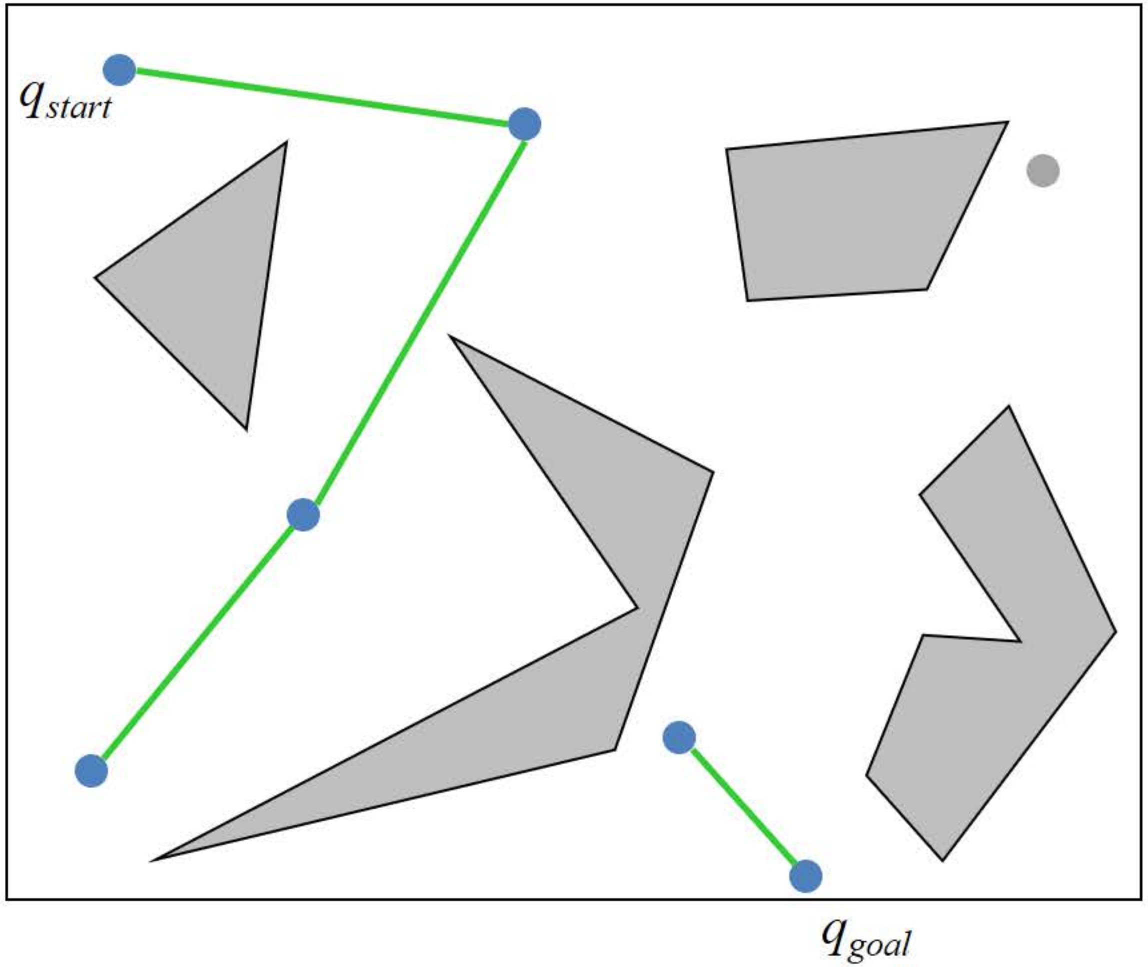


$T_{start} = (q_{start}, \emptyset)$, $T_{goal} = (q_{goal}, \emptyset)$

For i = 1 to n_{iter}

- q = random configuration in Q_{free}
- q_a = closest node in T_{start}
- **If** NOT collide(qq_a)
  - Add (q,q_a) to T_{start}
- q_b = closest node in T_{goal}
- **If** NOT collide(qq_b)
  - Add (q,q_b) to T_{goal}
- **If** q connected to T_{start} and T_{goal}
  - break

Rapidly-exploring Random Trees (RRTs)

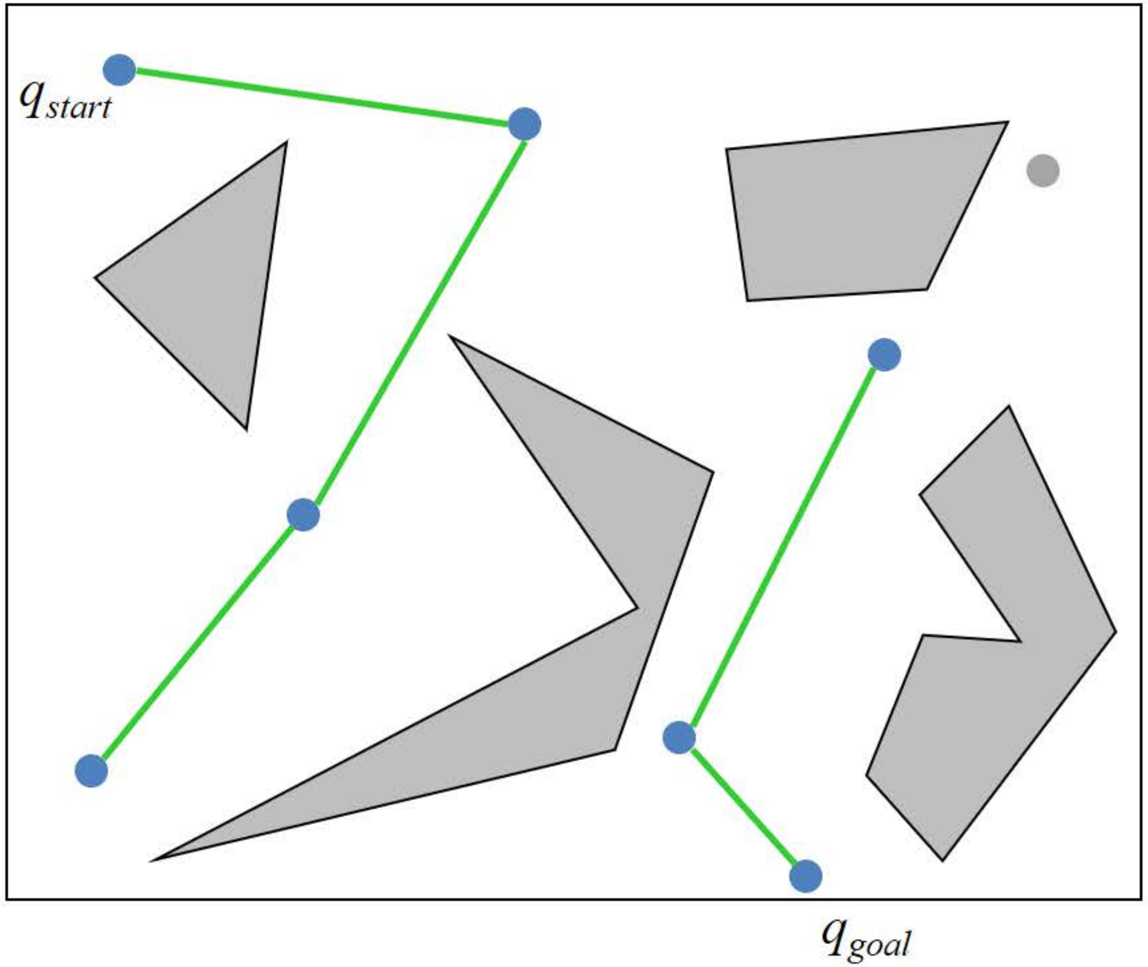


$T_{start} = (q_{start}, \emptyset)$, $T_{goal} = (q_{goal}, \emptyset)$

For i = 1 to n_{iter}

- q = random configuration in Q_{free}
- q_a = closest node in T_{start}
- **If** NOT collide(qq_a)
  - Add (q,q_a) to T_{start}
- q_b = closest node in T_{goal}
- **If** NOT collide(qq_b)
  - Add (q,q_b) to T_{goal}
- **If** q connected to T_{start} and T_{goal}
  - break

Rapidly-exploring Random Trees (RRTs)

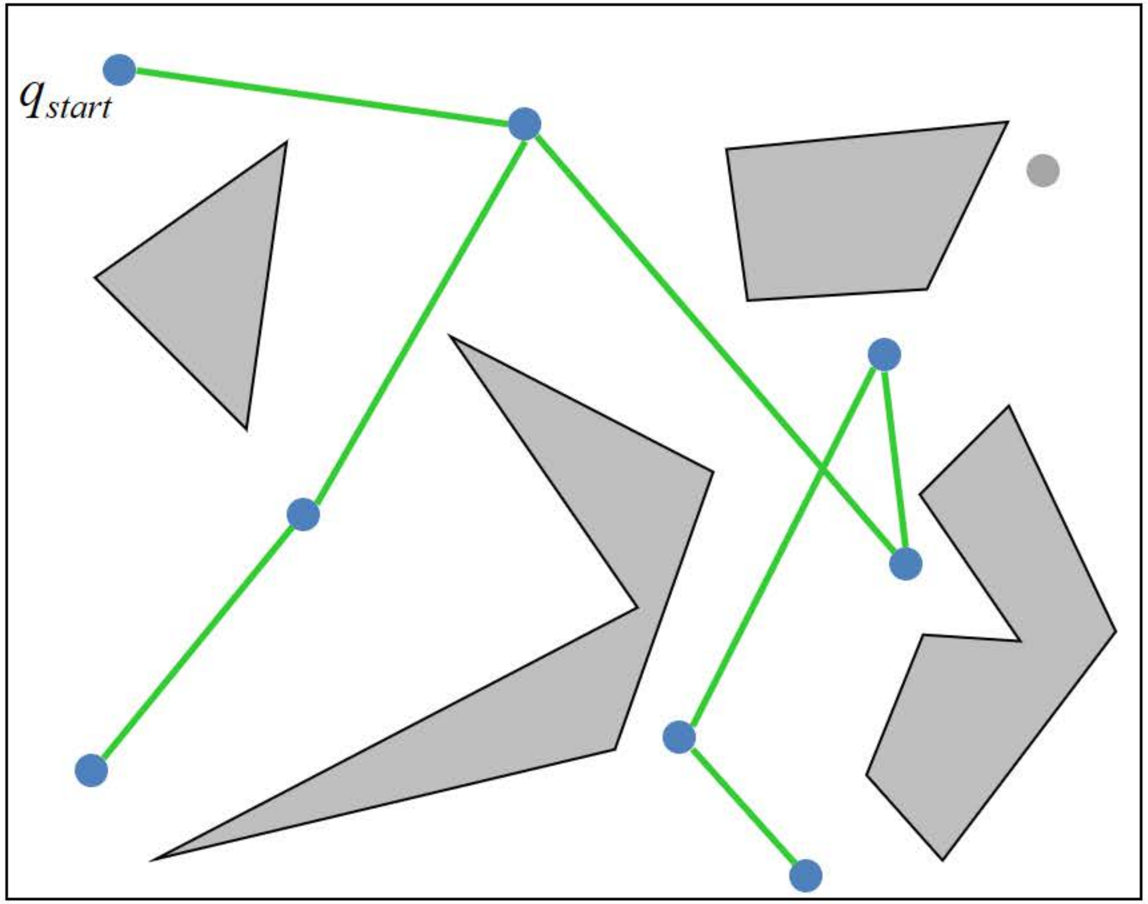


$T_{start} = (q_{start}, \emptyset)$, $T_{goal} = (q_{goal}, \emptyset)$

For i = 1 to n_{iter}

- q = random configuration in Q_{free}
- q_a = closest node in T_{start}
- **If** NOT collide(qq_a)
  - Add (q, q_a) to T_{start}
- q_b = closest node in T_{goal}
- **If** NOT collide(qq_b)
  - Add (q, q_b) to T_{goal}
- **If** q connected to T_{start} and T_{goal}
  - break

Rapidly-exploring Random Trees (RRTs)

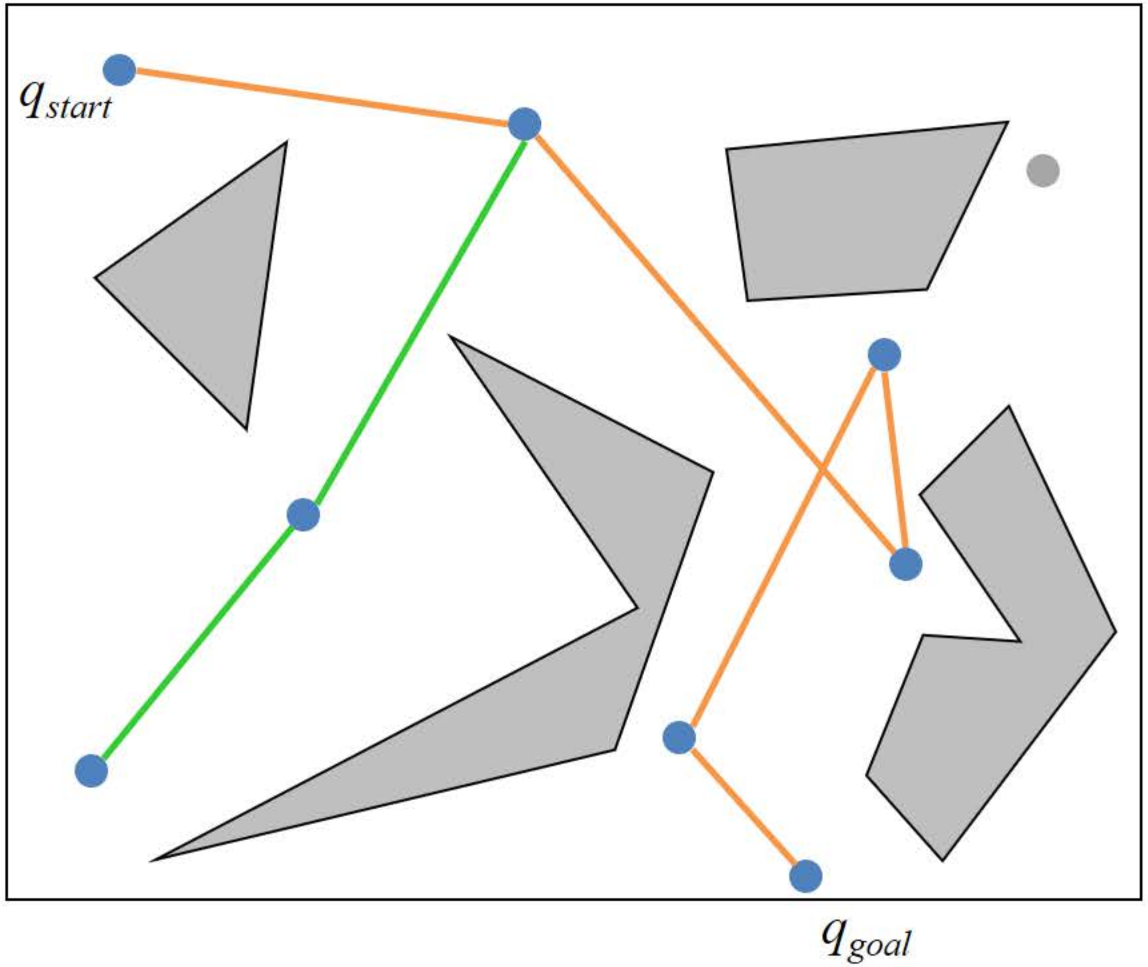


$T_{start} = (q_{start}, \emptyset)$, $T_{goal} = (q_{goal}, \emptyset)$

For i = 1 to n_{iter}

- q = random configuration in Q_{free}
- q_a = closest node in T_{start}
- **If** NOT collide(qq_a)
  - Add (q,q_a) to T_{start}
- q_b = closest node in T_{goal}
- **If** NOT collide(qq_b)
  - Add (q,q_b) to T_{goal}
- **If** q connected to T_{start} and T_{goal}
  - break

Rapidly-exploring Random Trees (RRTs)

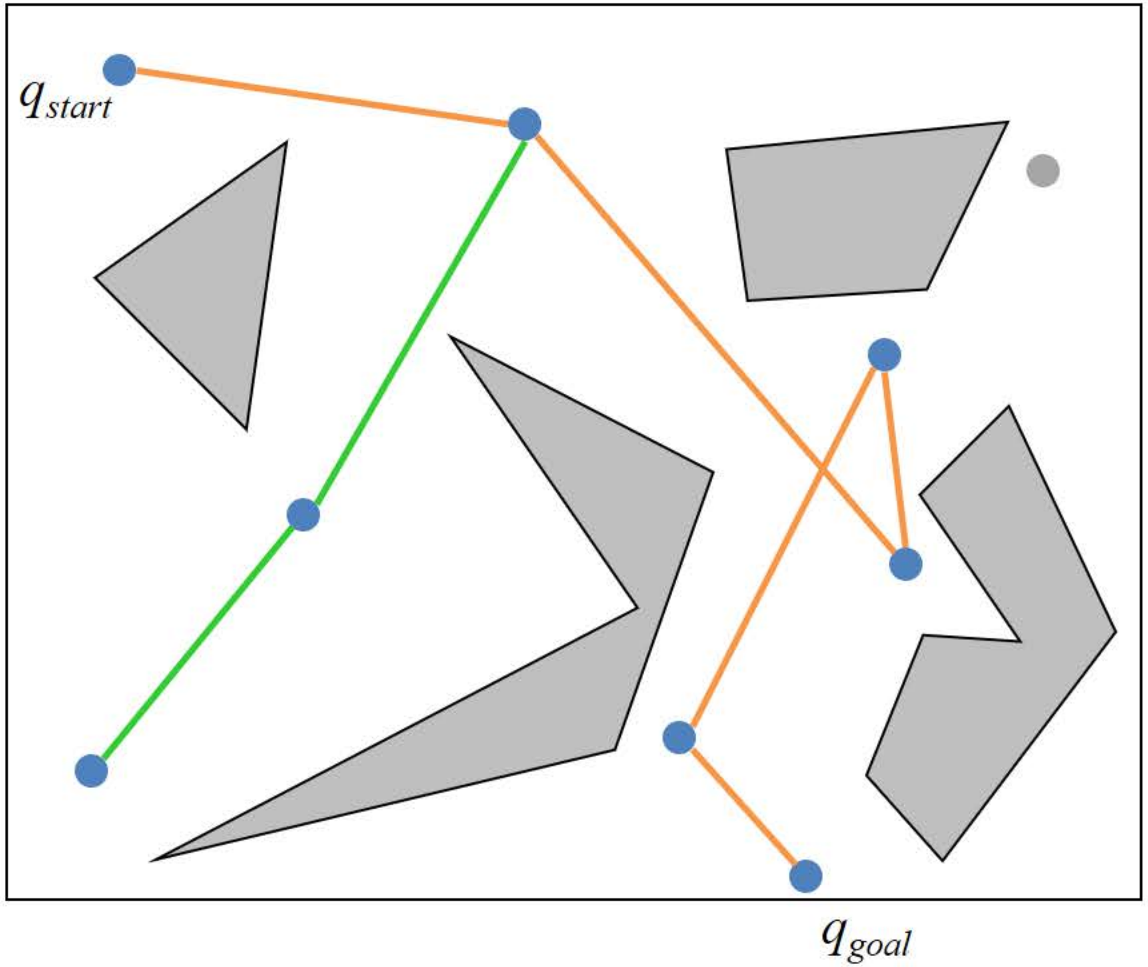


$T_{start} = (q_{start}, \emptyset)$, $T_{goal} = (q_{goal}, \emptyset)$

For i = 1 to n_{iter}

- q = random configuration in Q_{free}
- q_a = closest node in T_{start}
- **If** NOT collide(qq_a)
  - Add (q, q_a) to T_{start}
- q_b = closest node in T_{goal}
- **If** NOT collide(qq_b)
  - Add (q, q_b) to T_{goal}
- **If** q connected to T_{start} and T_{goal}
  - break

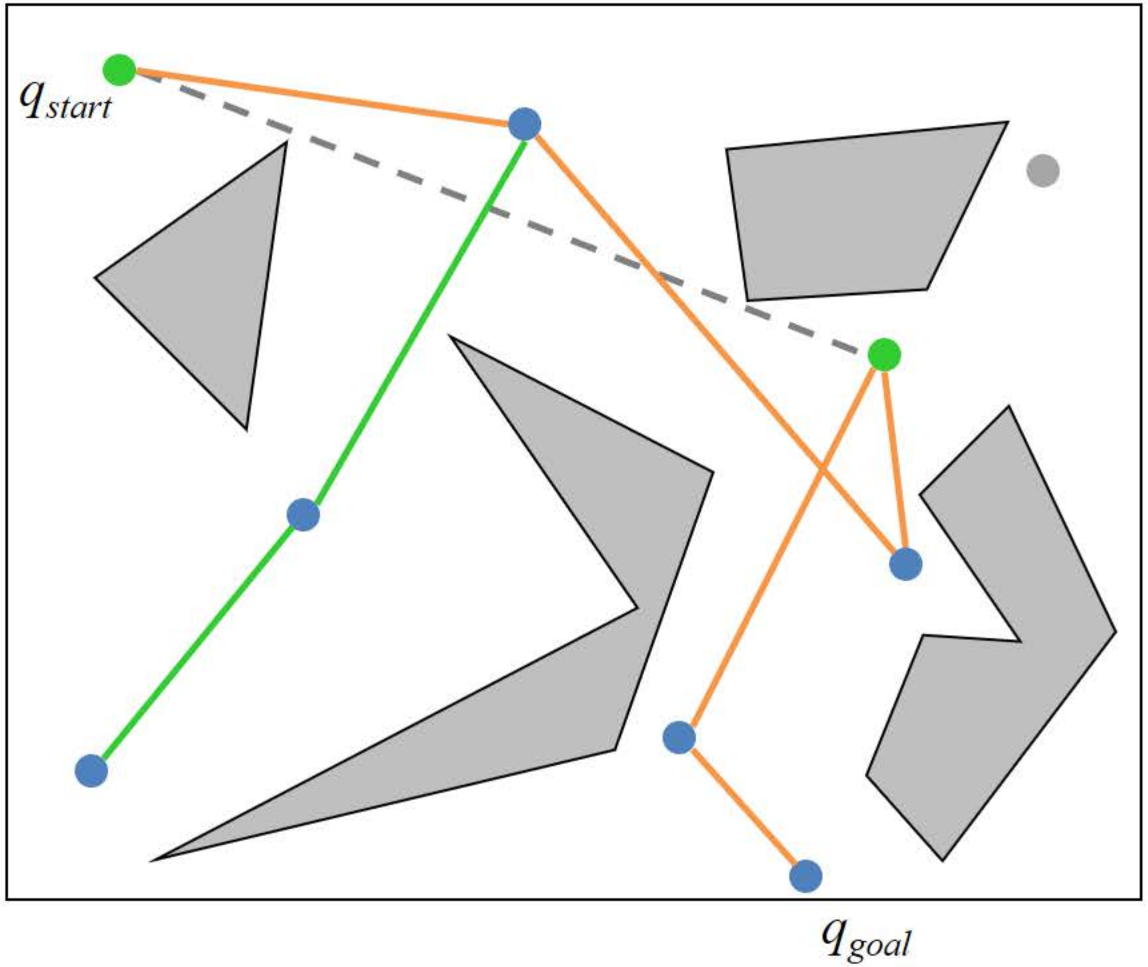
Post-Processing: Greedy



Choose two random nodes

Try to connect them together

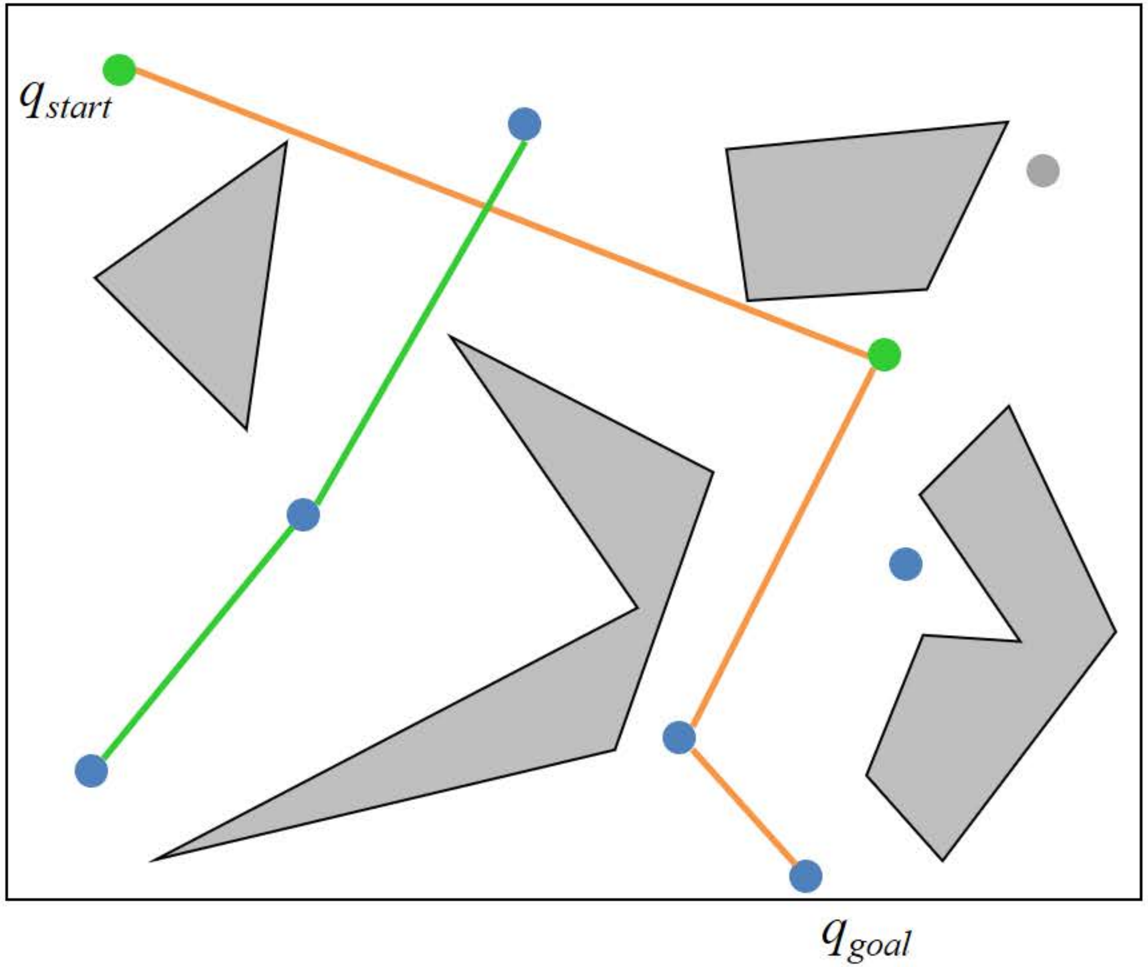
Post-Processing: Greedy



Choose two random nodes

Try to connect them together

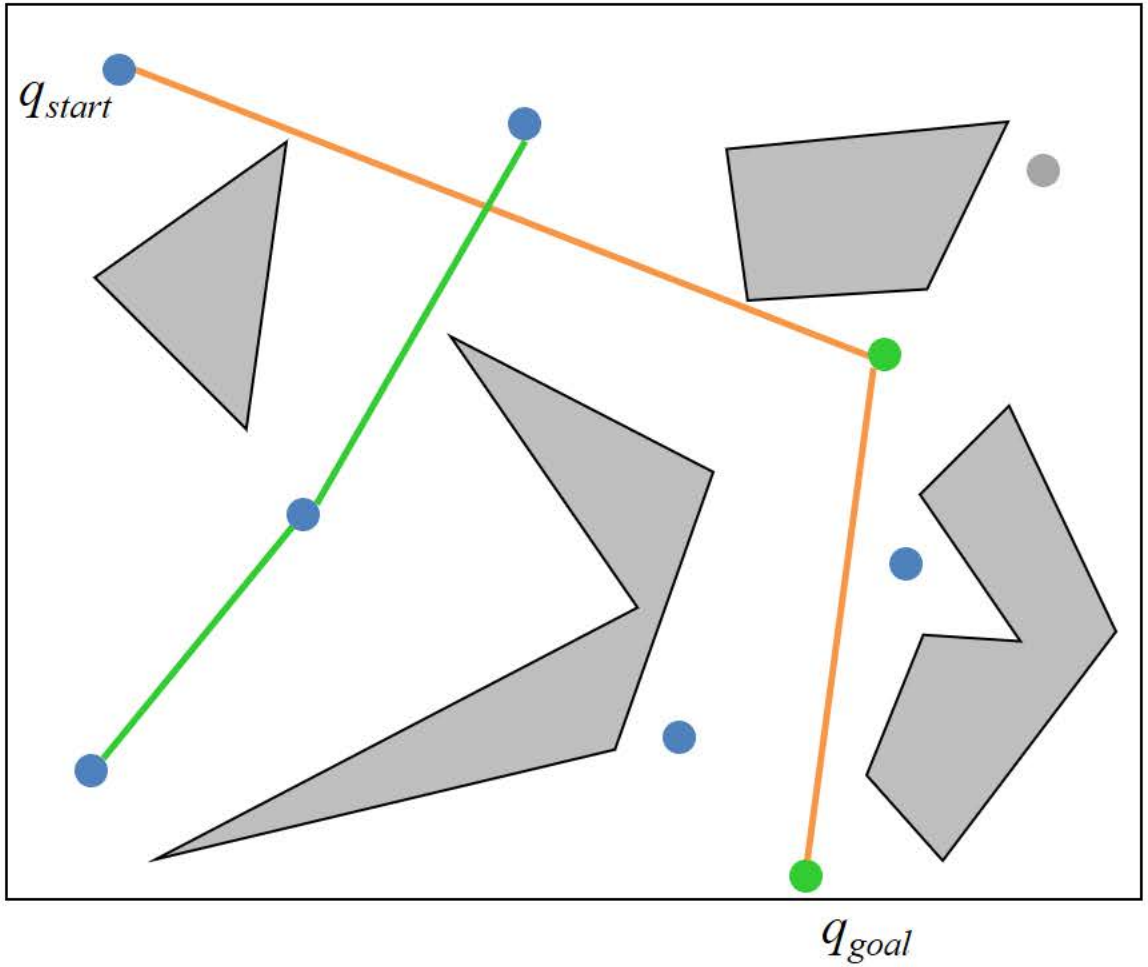
Post-Processing: Greedy



Choose two random nodes

Try to connect them together

Post-Processing: Greedy



Choose two random nodes

Try to connect them together

Common Variants

- Move in the direction of q by a max step size
- Connect to multiple neighbors: RRG (G=graph)
- Add a heuristic to bias sampling: Informed RRT
- Check collisions only once trees are joined: Lazy RRT

Kinodynamic Planning

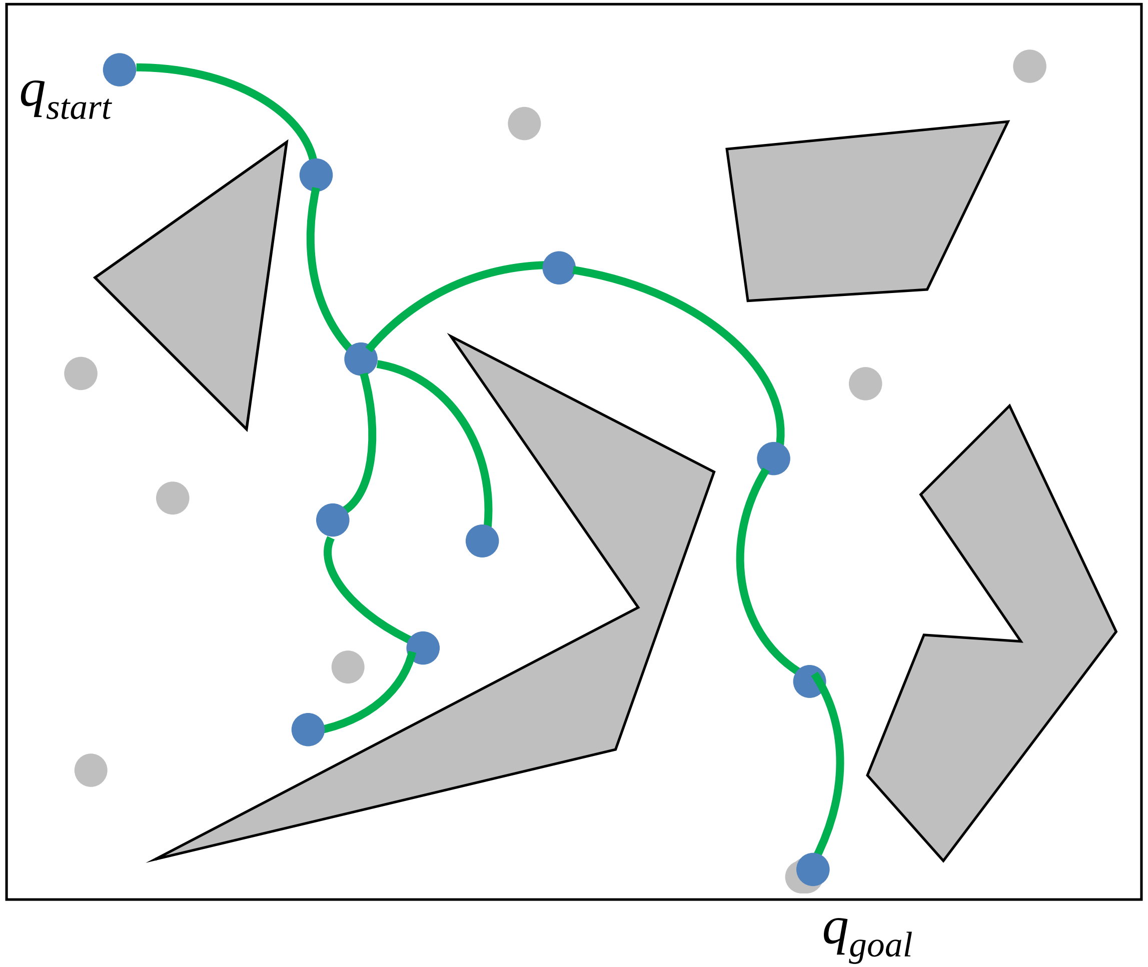
Kinodynamic planning requires velocity, acceleration, and force/torque bounds to be satisfied in addition to any task/kinematics constraints.

Grid-based search methods have some issues with this.

We have to be able to create a graph based on **achievable** motions.

RRT was originally designed to do kinodynamic planning.

Previously: Rapidly-exploring Random Trees (RRTs)

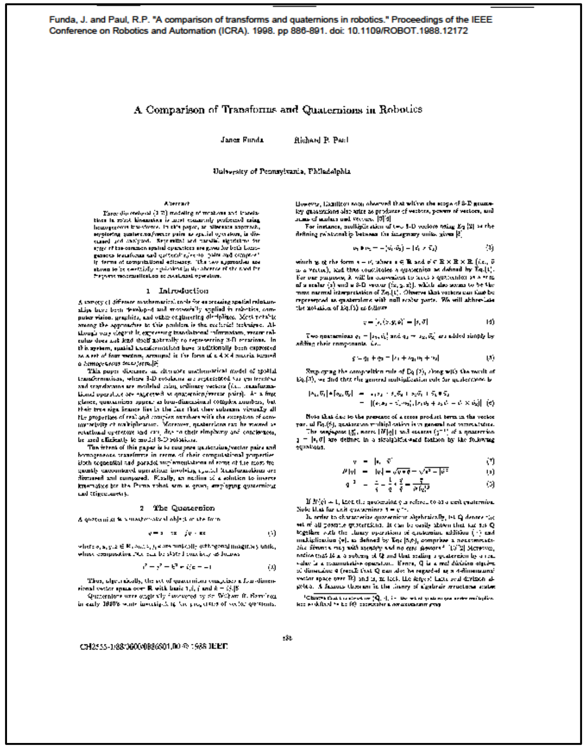


$T_{start} = (q_{start}, \emptyset)$, $T_{goal} = (q_{goal}, \emptyset)$

For i = 1 to n_{iter}

- q = random configuration in Q_{free}
- q_a = closest node in T_{start}
- **If** NOT collide(qq_a')
  - Add (q,q_a') to T_{start}
- q_b = closest node in T_{goal}
- **If** NOT collide(qq_b')
  - Add (q,q_b') to T_{goal}
- **If** q_a' within ε distance to q_{goal}
  - break

Next time: POMDPs!


Funda, J. and Paul, R.P. "A comparison of transforms and quaternions in robotics." Proceedings of the IEEE Conference on Robotics and Automation (ICRA). 1998. pp 886-891. doi: 10.1109/ROBOT.1988.12172

A Comparison of Transforms and Quaternions in Robotics

James Funda Richard P. Paul

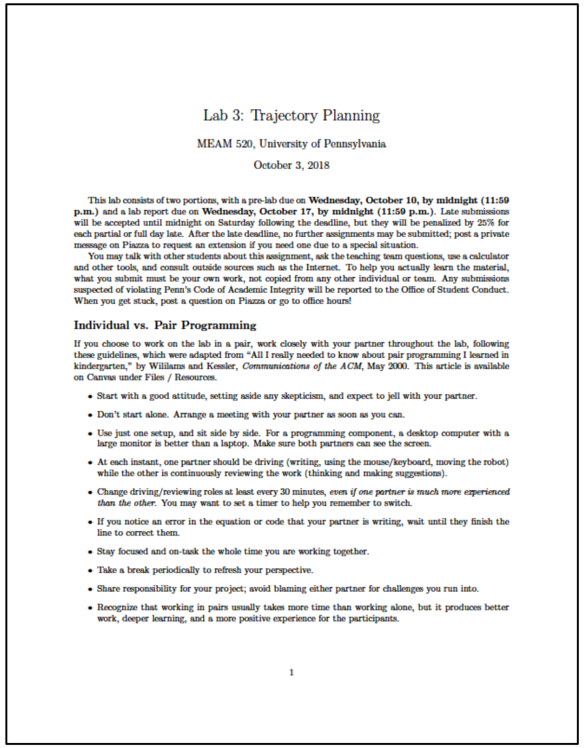
University of Pennsylvania, Philadelphia

Abstract

1 Introduction

2 The Quaternion

“Grasping POMDPs”
(ICRA 2007)


Lab 3: Trajectory Planning

MEAM 520, University of Pennsylvania

October 3, 2018

This lab consists of two portions, with a pre-lab due on **Wednesday, October 10, by midnight (11:59 p.m.)** and a lab report due on **Wednesday, October 17, by midnight (11:59 p.m.)**. Late submissions will be accepted until midnight on Saturday following the deadline, but they will be penalized by 25% for each partial or full day late. After the late deadline, no further assignments may be submitted; post a private message on Piazza to request an extension if you need one due to a special situation.

You may talk with other students about this assignment, ask the teaching team questions, use a calculator and other tools, and consult outside sources such as the Internet. To help you actually learn the material, what you submit must be your own work, not copied from any other individual or team. Any submissions suspected of violating Penn's Code of Academic Integrity will be reported to the Office of Student Conduct. When you get stuck, post a question on Piazza or go to office hours!

Individual vs. Pair Programming

If you choose to work on the lab in a pair, work closely with your partner throughout the lab, following these guidelines, which were adapted from “All I really needed to know about pair programming I learned in kindergarten,” by Wililams and Kessler, *Communications of the ACM*, May 2000. This article is available on Canvas under Files / Resources.

- Start with a good attitude, setting aside any skepticism, and expect to jell with your partner.
- Don't start alone. Arrange a meeting with your partner as soon as you can.
- Use just one setup, and sit side by side. For a programming component, a desktop computer with a large monitor is better than a laptop. Make sure both partners can see the screen.
- At each instant, one partner should be driving (writing, using the mouse/keyboard, moving the robot) while the other is continuously reviewing the work (thinking and making suggestions).
- Change driving/reviewing roles at least every 30 minutes, *even if one partner is much more experienced than the other*. You may want to set a timer to help you remember to switch.
- If you notice an error in the equation or code that your partner is writing, wait until they finish the line to correct them.
- Stay focused and on-task the whole time you are working together.
- Take a break periodically to refresh your perspective.
- Share responsibility for your project; avoid blaming either partner for challenges you run into.
- Recognize that working in pairs usually takes more time than working alone, but it produces better work, deeper learning, and a more positive experience for the participants.

1

Lab 3: Inverse Kinematics due 10/17

- You can now do all the tasks