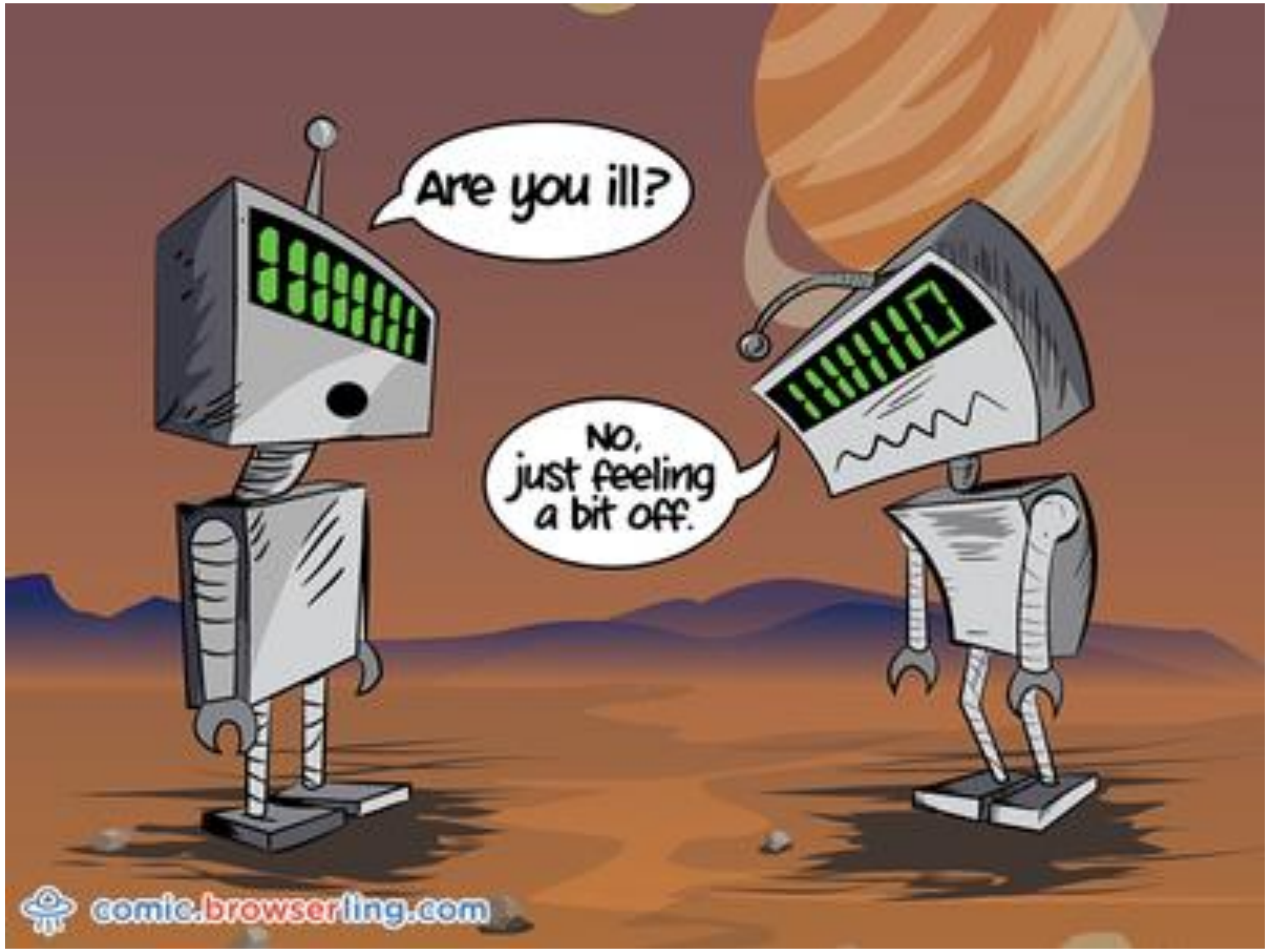
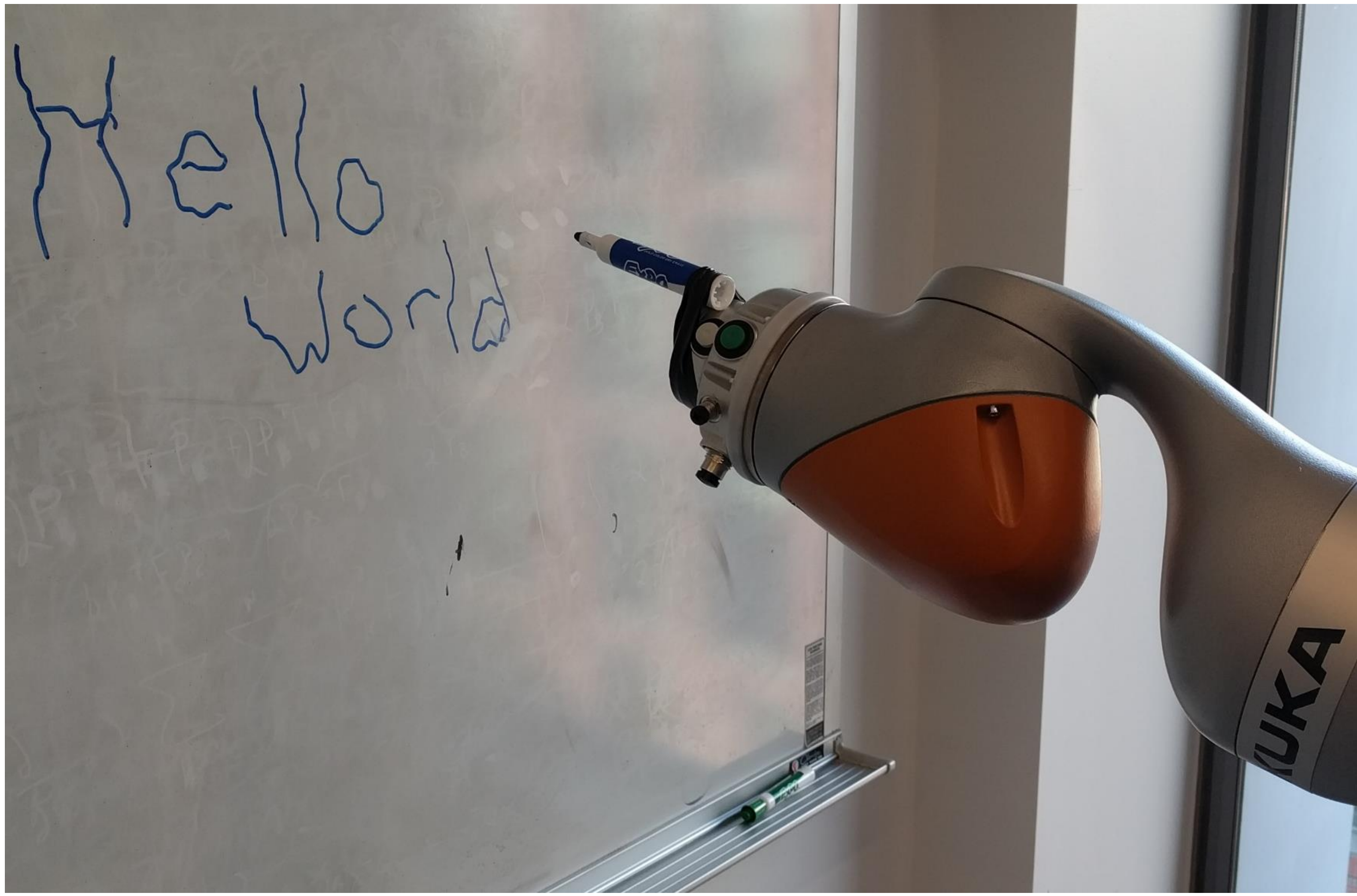
Robot Motion Planning



Brian Plancher
12/10/21

Hi I’m Brian!



Hi I'm Brian!

- **I'm obsessed with my dog Alvin**

Hi I’m Brian!

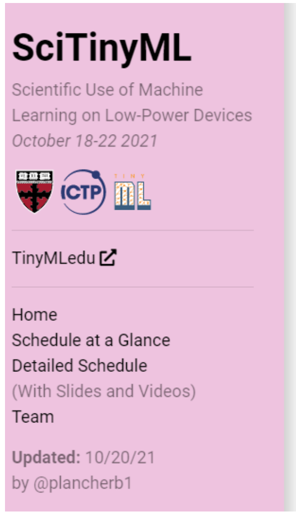
- I’m obsessed with my dog Alvin **and my daughter Tess**

Hi I’m Brian!

- I’m obsessed with my dog Alvin and my daughter Tess
- **I am passionate about teaching accessible, interdisciplinary hands-on, project-based courses**

Hi I’m Brian!

- I’m obsessed with my dog Alvin and my daughter Tess
- **I am passionate about teaching accessible, interdisciplinary hands-on, project-based courses**



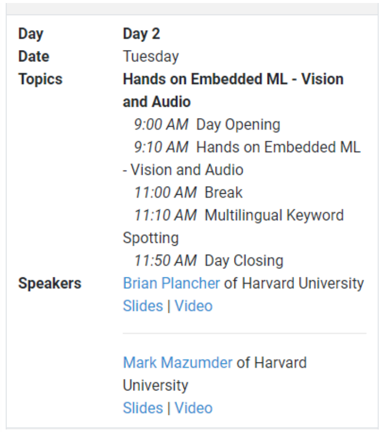
SciTinyML

Scientific Use of Machine Learning on Low-Power Devices
October 18-22 2021

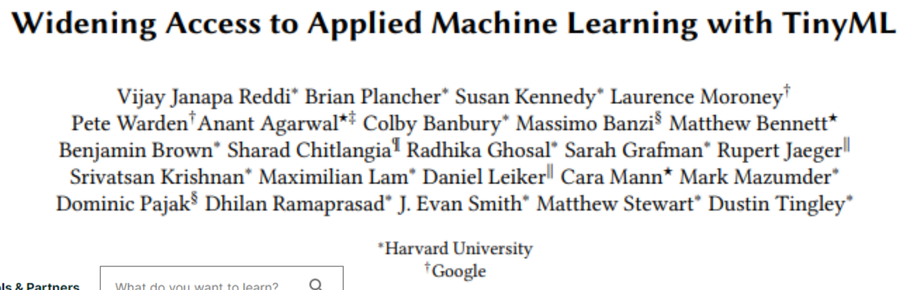
TinyMLedu

Home
Schedule at a Glance
Detailed Schedule
(With Slides and Videos)
Team

Updated: 10/20/21
by @plancherb1



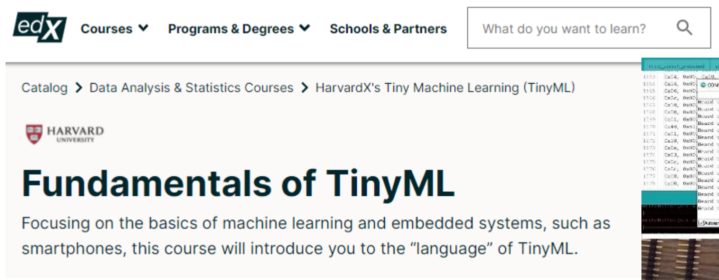
Day	**Day 2**
Date	Tuesday
Topics	**Hands on Embedded ML - Vision and Audio** *9:00 AM* Day Opening *9:10 AM* Hands on Embedded ML - Vision and Audio *11:00 AM* Break *11:10 AM* Multilingual Keyword Spotting *11:50 AM* Day Closing
Speakers	Brian Plancher of Harvard University Slides | Video Mark Mazumder of Harvard University Slides | Video



Widening Access to Applied Machine Learning with TinyML

Vijay Janapa Reddi* Brian Plancher* Susan Kennedy* Laurence Moroney†
Pete Warden† Anant Agarwal*‡ Colby Banbury* Massimo Banzi§ Matthew Bennett*
Benjamin Brown* Sharad Chitlangia¶ Radhika Ghosal* Sarah Grafman* Rupert Jaeger‖
Srivatsan Krishnan* Maximilian Lam* Daniel Leiker‖ Cara Mann* Mark Mazumder*
Dominic Pajak§ Dhilan Ramaprasad* J. Evan Smith* Matthew Stewart* Dustin Tingley*

*Harvard University
†Google



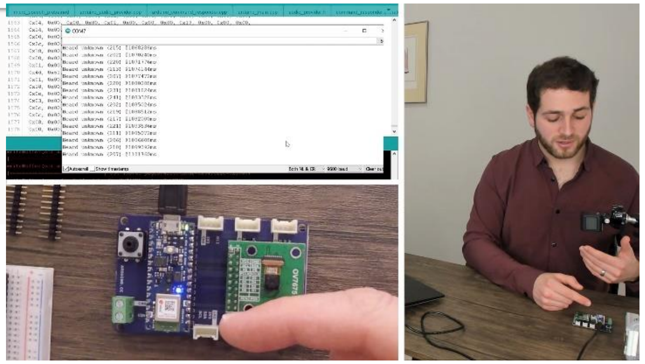
edX Courses Programs & Degrees Schools & Partners What do you want to learn?

Catalog > Data Analysis & Statistics Courses > HarvardX's Tiny Machine Learning (TinyML)

HARVARD UNIVERSITY

Fundamentals of TinyML

Focusing on the basics of machine learning and embedded systems, such as smartphones, this course will introduce you to the “language” of TinyML.



Hi I’m Brian!

- I’m obsessed with my dog Alvin and my daughter Tess
- I am passionate about teaching accessible, interdisciplinary hands-on, project-based courses
- **My research focuses on developing open-source planning and control algorithms that enable robots to operate in the real world and help people!**

How about you?

1. How many people in the audience have a degree in computer science? (e.g., Faculty, Staff)
2. How many people have taken a handful of CS courses? (e.g., Junior or Senior CS Majors)
3. How about one or two CS courses?
4. How many people have no CS background?

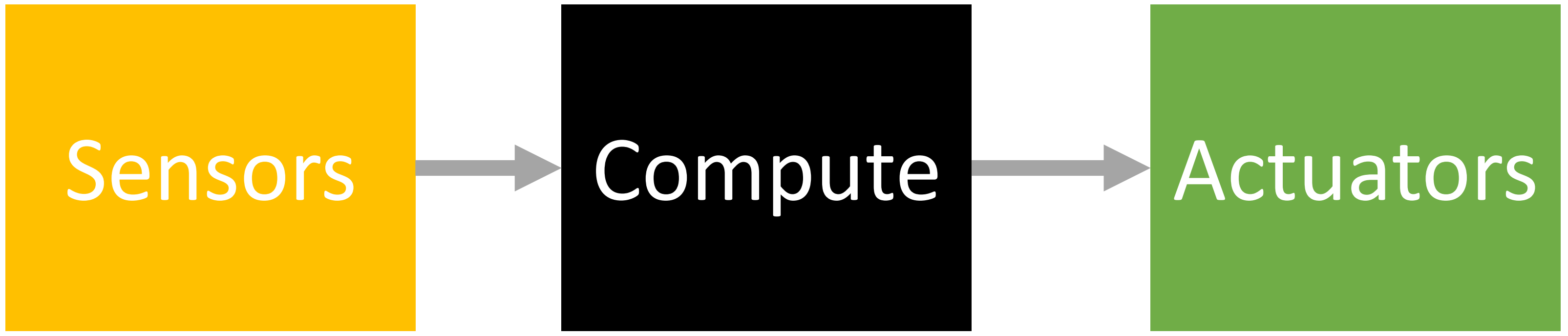
Learning Goals for Today

1. Learn some of the **language** of robotics
2. Understand the **importance of tradeoffs** in the selection of robotics algorithms for **real-world deployments**
3. Gain practice in exploring the **attributes of classes of algorithms** through an example

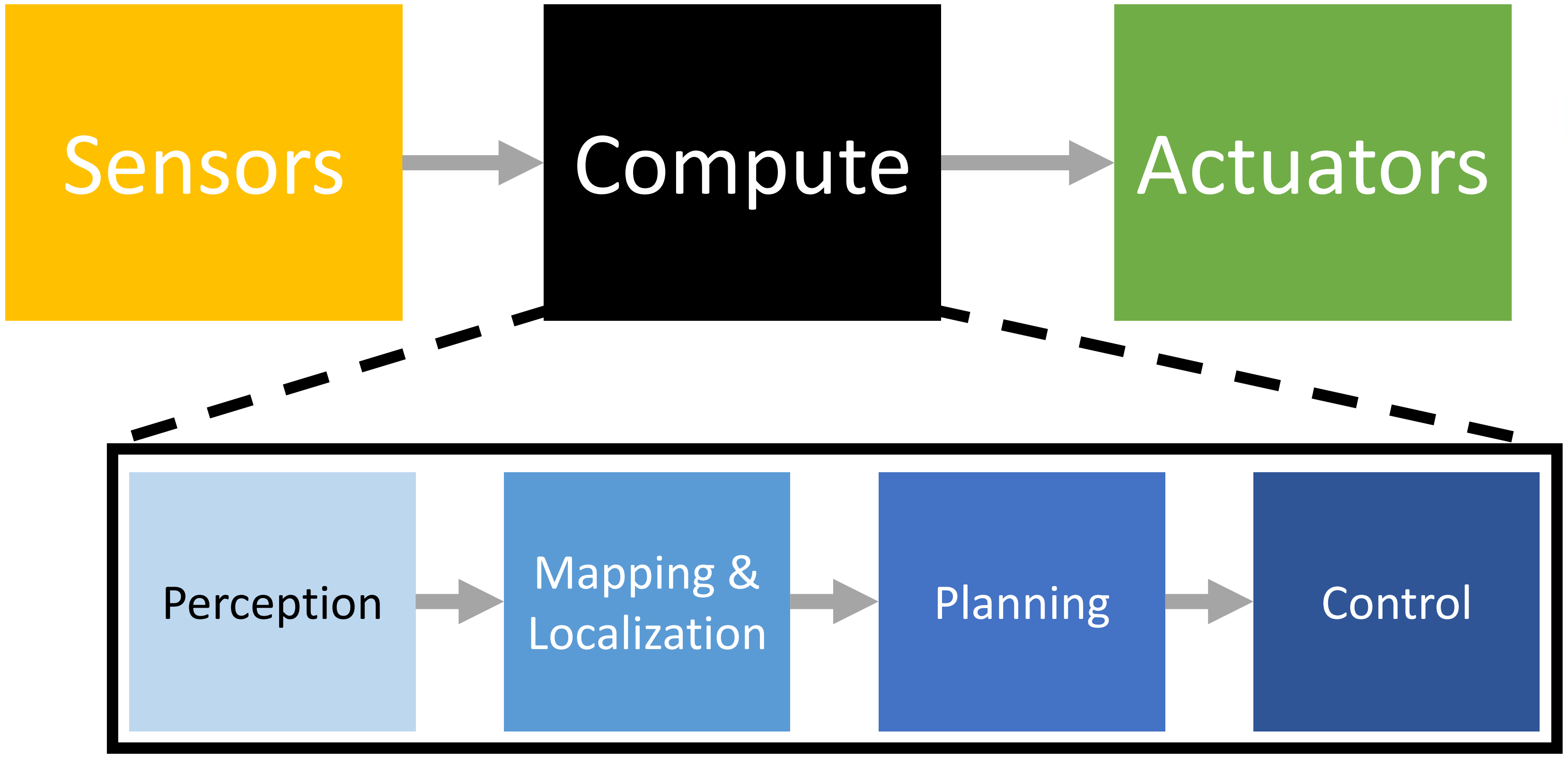
Learning Goals for Today

1. Learn some of the **language** of robotics
2. Understand the **importance of tradeoffs** in the selection of robotics algorithms for **real-world deployments**
3. Gain practice in exploring the **attributes of classes of algorithms** through an example

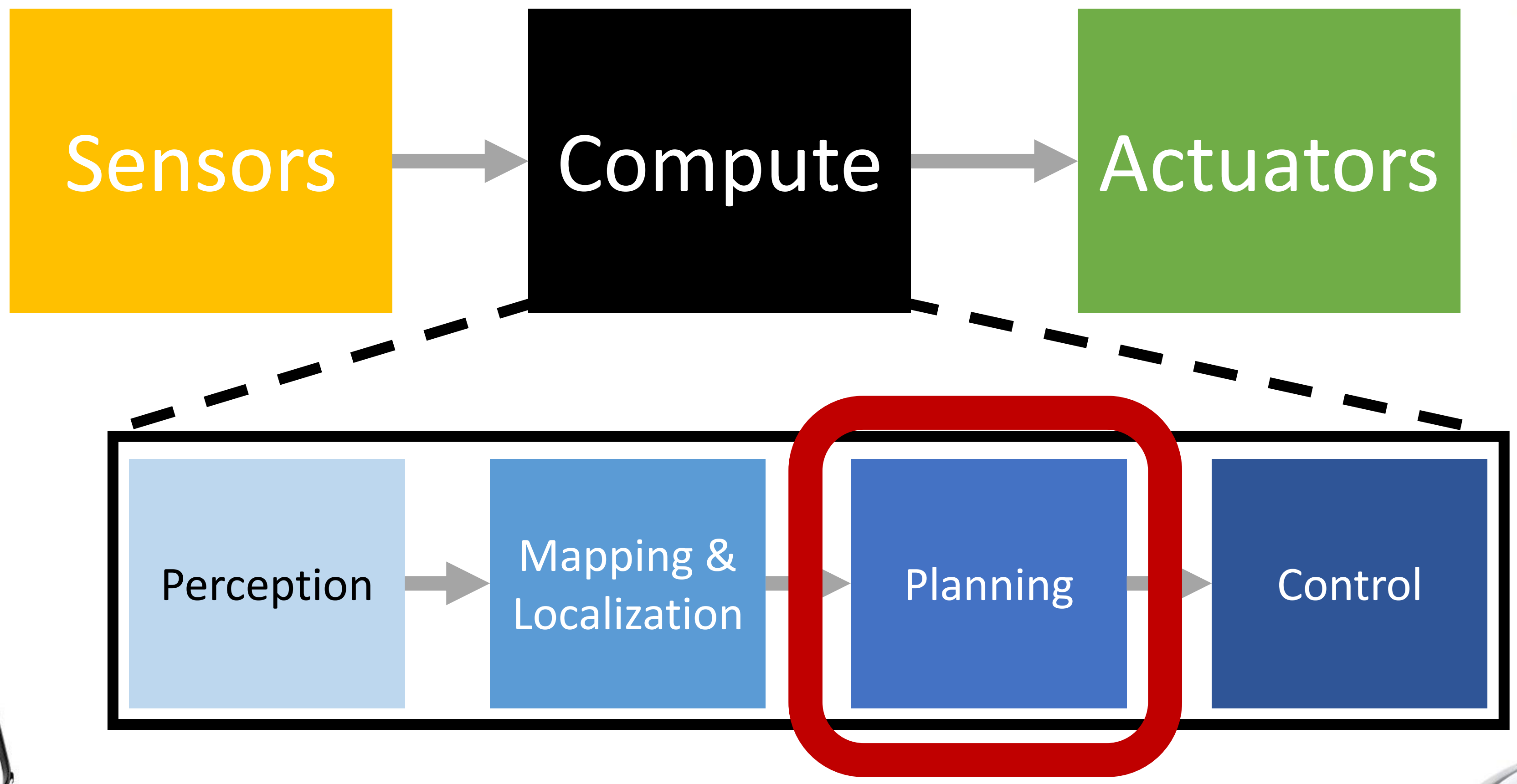
Robotics is a **BIG** space



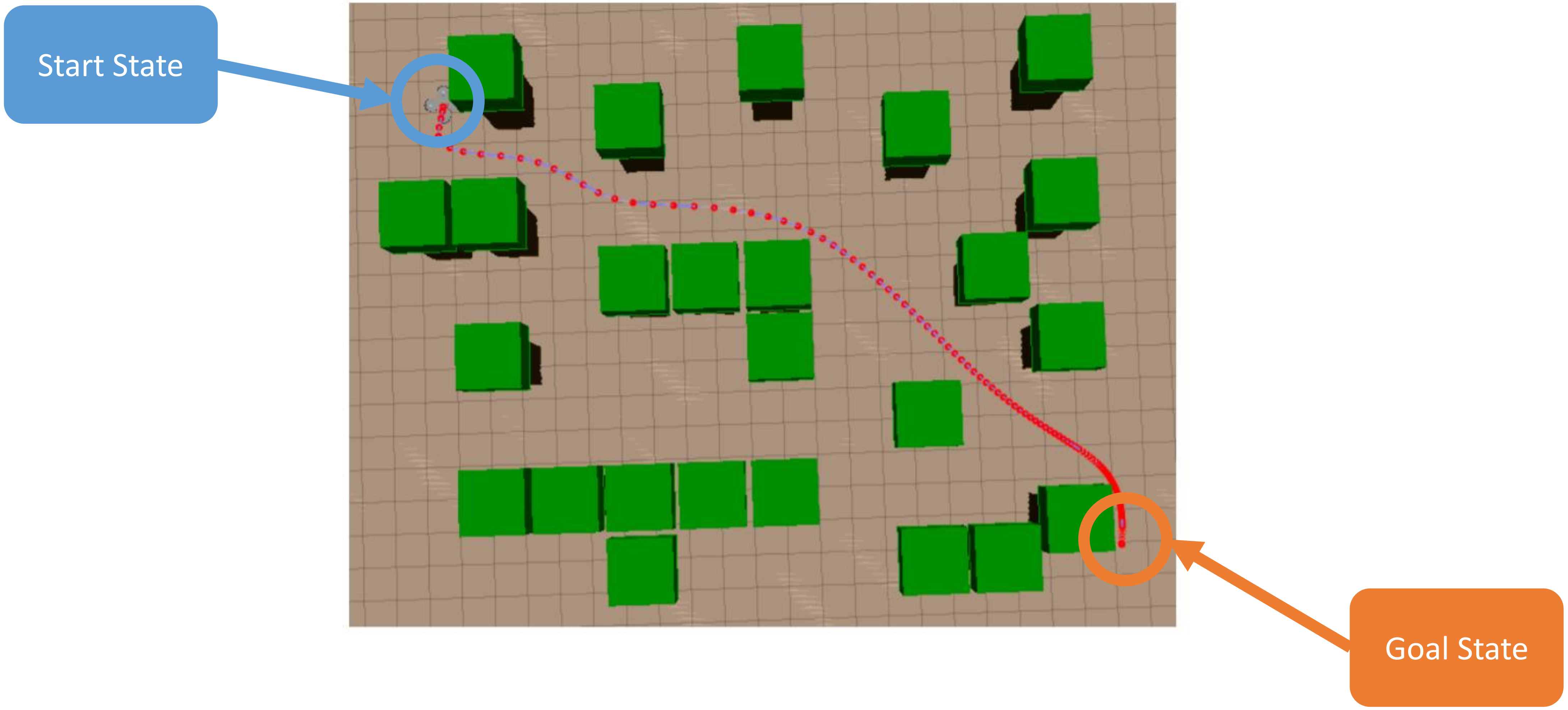
Robotics is a **BIG** space



Robotics is a **BIG** space



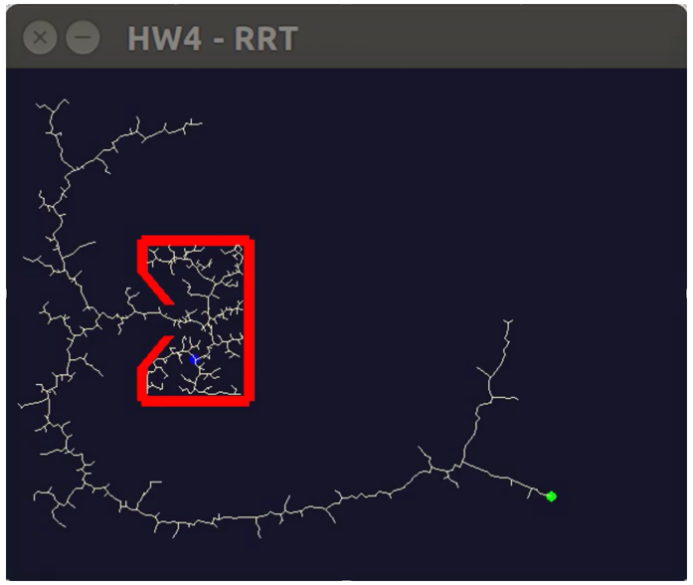
Planning is the process of computing an action plan for a robot based on given map of the world



Learning Goals for Today

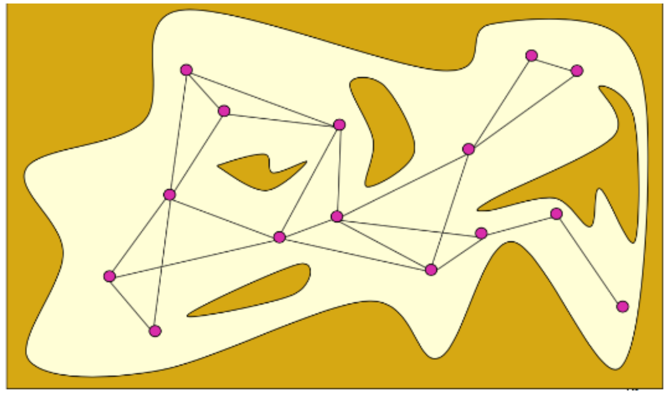
1. Learn some of the **language** of robotics
2. Understand the **importance of tradeoffs** in the selection of robotics algorithms for **real-world deployments**
3. Gain practice in exploring the **attributes of classes of algorithms** through an example

What types of algorithms can we use for planning?

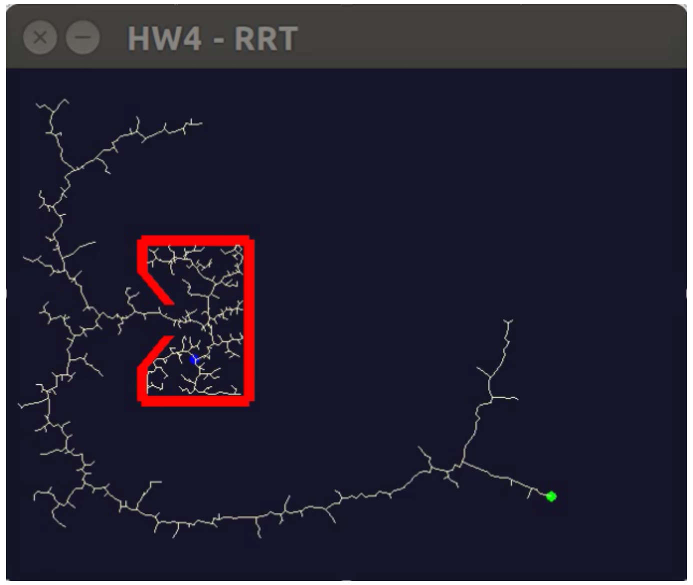


Random Search (RRT)

Random Search (PRM)

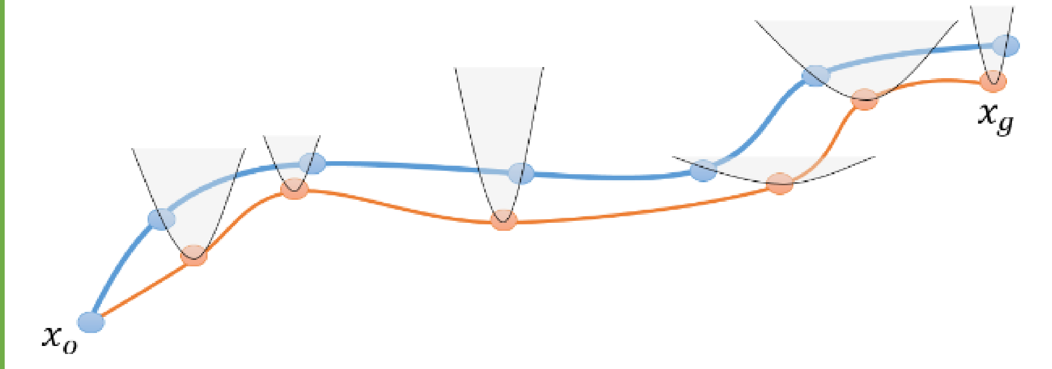


What types of algorithms can we use for planning?

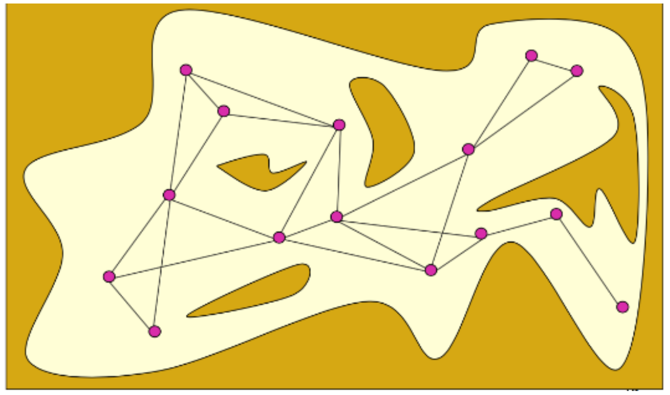


Random Search (RRT)

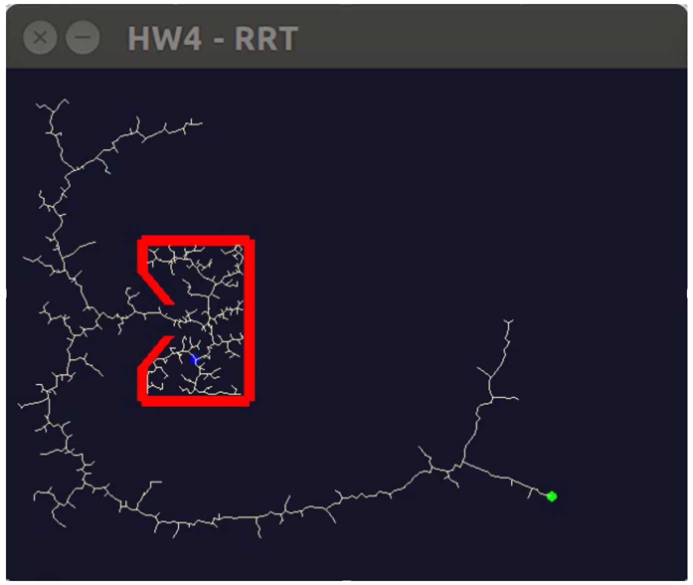
Optimal Local Search



Random Search (PRM)

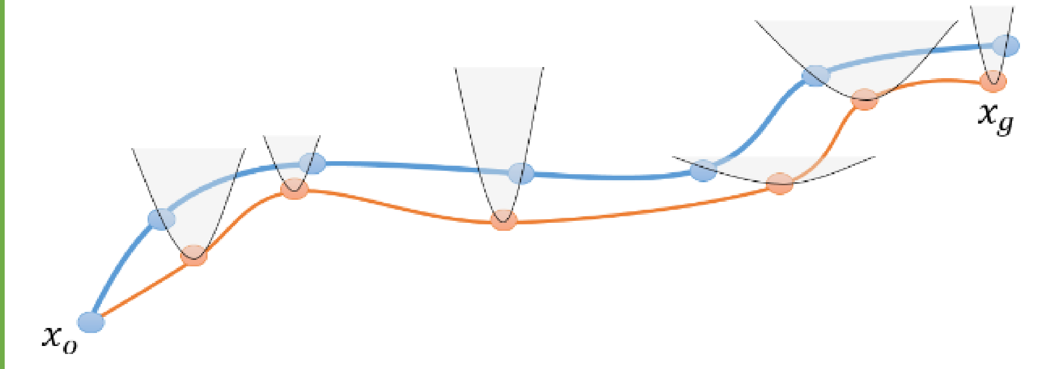


What types of algorithms can we use for planning?

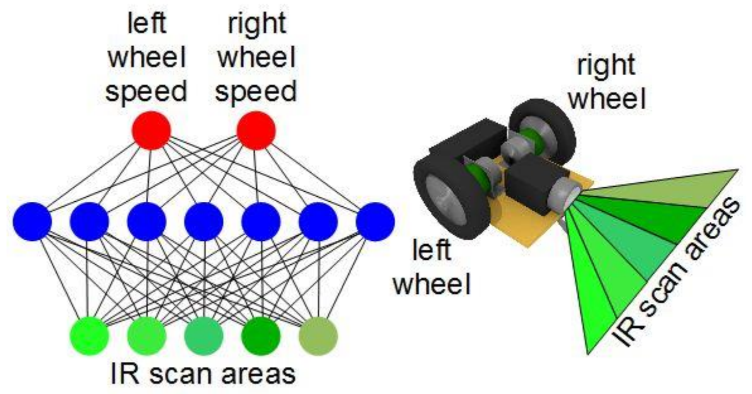


Random Search (RRT)

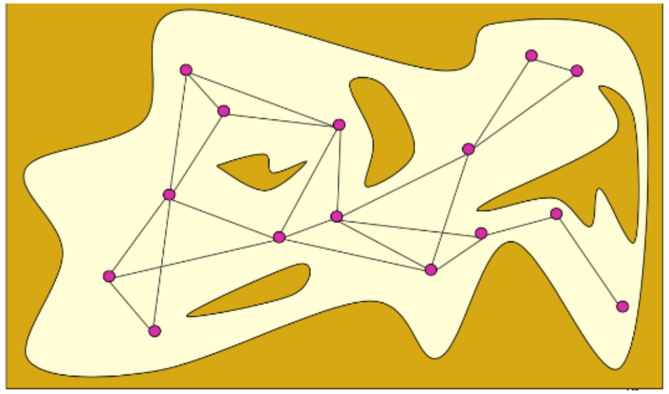
Optimal Local Search



Machine Learning



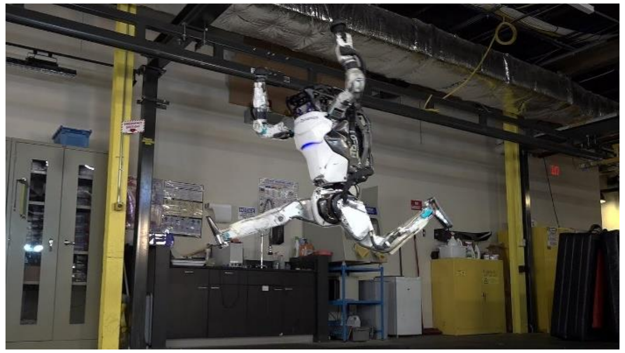
Random Search (PRM)



What types of algorithms can we use for planning?

Random Search (RRT)

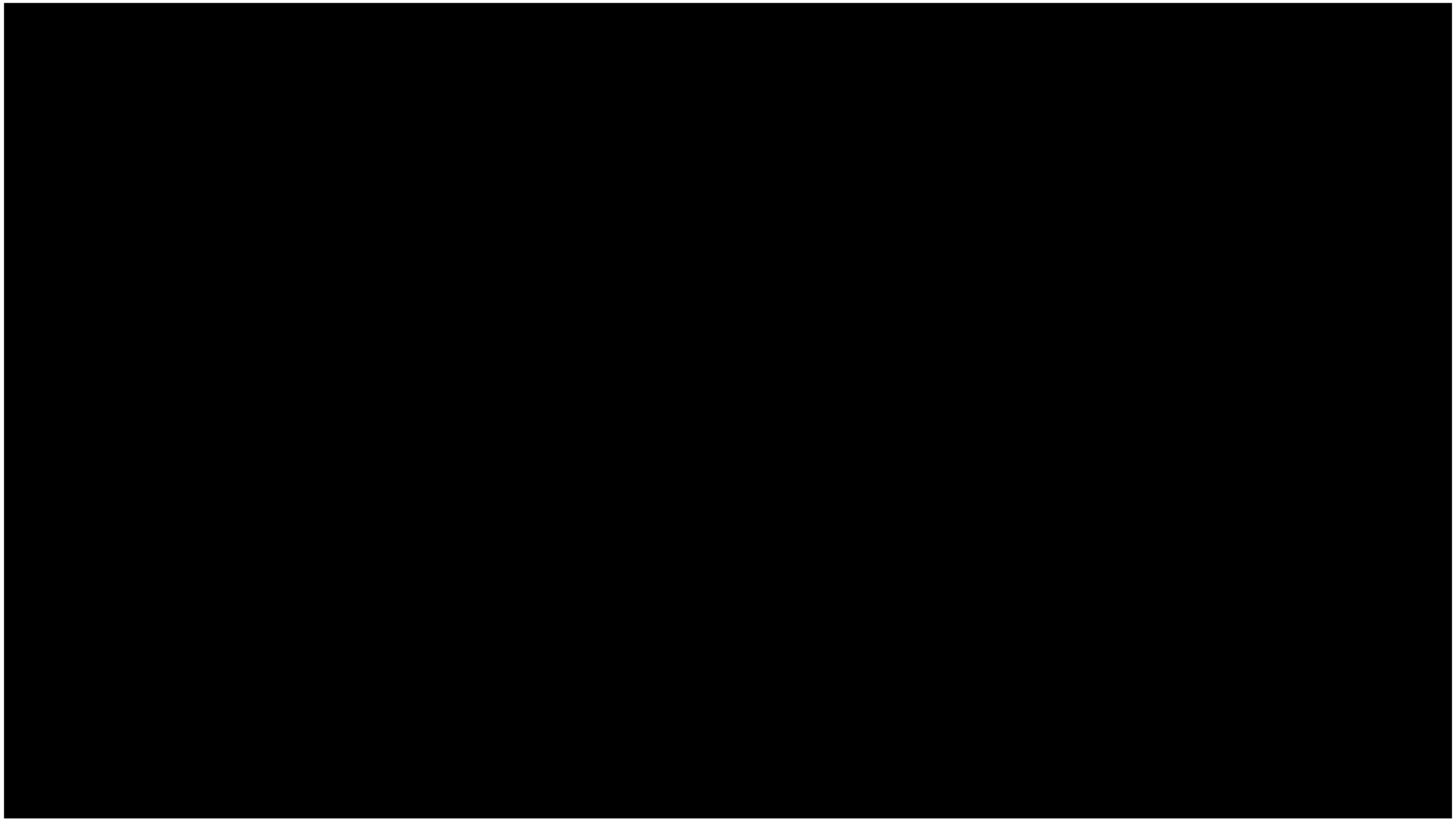
Optimal Local Search



Machine Learning

Random Search (PRM)

A case study on algorithm selection: CafeX



A case study on algorithm selection: CafeX

CafeX, the San Francisco based startup, has hired you to upgrade the motion planning software for their robot to make it faster without sacrificing coffee quality.

A case study on algorithm selection: CafeX

Take 2 minutes and consider: **Is the algorithm on the paper in front of you a good fit for the scenario below?**

CafeX, the San Francisco based startup, has hired you to upgrade the motion planning software for their robot to make it faster without sacrificing coffee quality.

A case study on algorithm selection: CafeX

Take another 2 minutes and turn to the **person next to you** and decide: **Which of your algorithms is a better fit?**

CafeX, the San Francisco based startup, has hired you to upgrade the motion planning software for their robot to make it faster without sacrificing coffee quality.

A case study on algorithm selection: CafeX

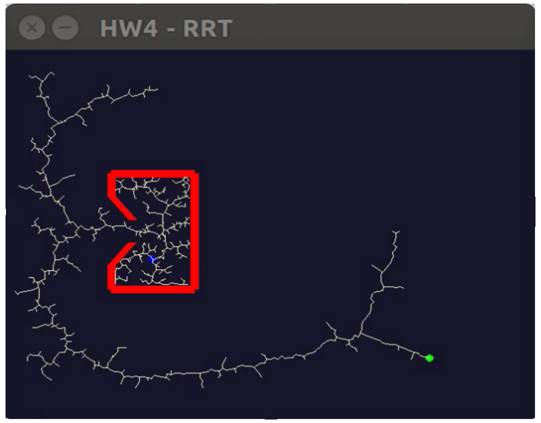
Take another 2 minutes and discuss with a **few pairs of people near you** and decide: **which of your algorithms is a better fit?**

CafeX, the San Francisco based startup, has hired you to upgrade the motion planning software for their robot to make it faster without sacrificing coffee quality.

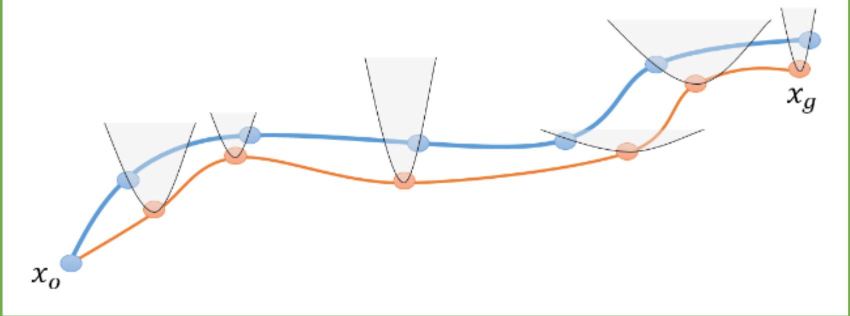
A case study on algorithm selection: CafeX

CafeX, the San Francisco based startup, has hired you to upgrade the motion planning software for their robot to make it faster without sacrificing coffee quality.

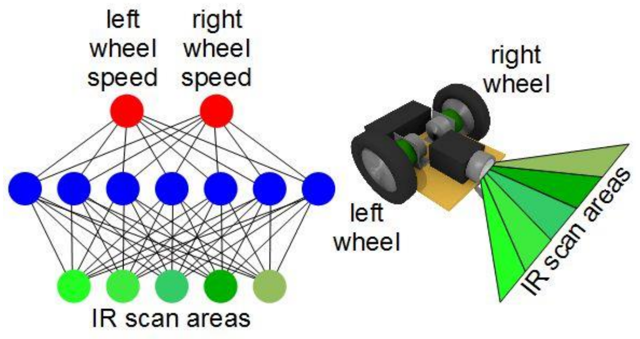
Random Search (RRT)



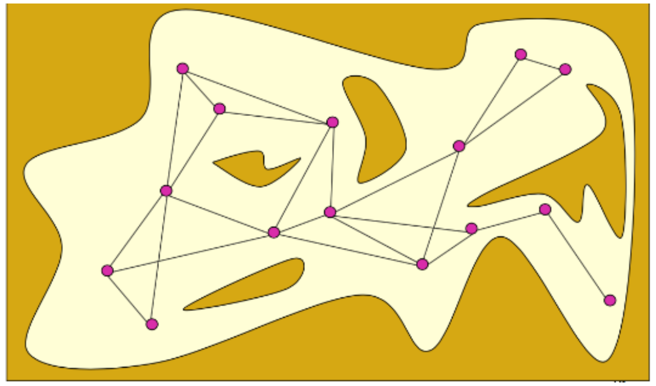
Optimal Local Search



Machine Learning



Random Search (PRM)



Learning Goals for Today

1. Learn some of the **language** of robotics
2. Understand the **importance of tradeoffs** in the selection of robotics algorithms for **real-world deployments**
3. Gain practice in exploring the **attributes of classes of algorithms** through an example

Robot Motion Planning with RRT

Naïve Random Search

Rapidly Exploring Random Trees (RRT)

Variants of RRT

Limitations of RRT

Robot Motion Planning with RRT

Naïve Random Search

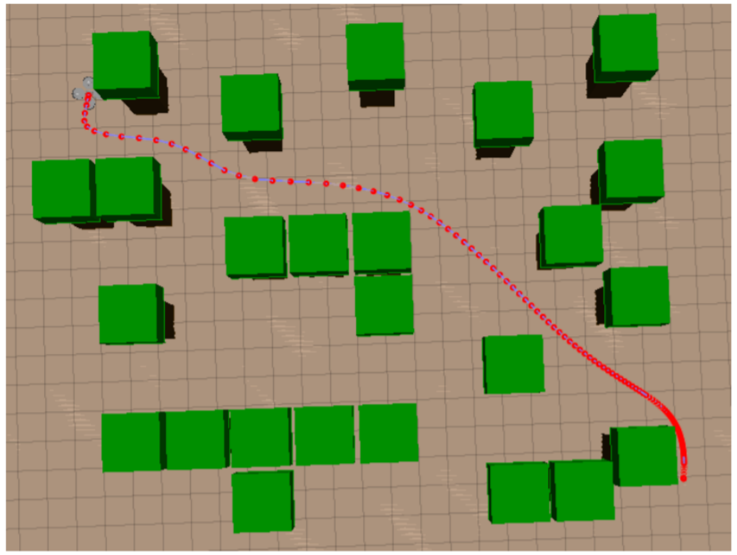
Rapidly Exploring Random Trees (RRT)

Variants of RRT

Limitations of RRT

A Naïve Random Search Based Approach

What if we **incrementally build up a graph** to explore our map and get from the start state to the goal state

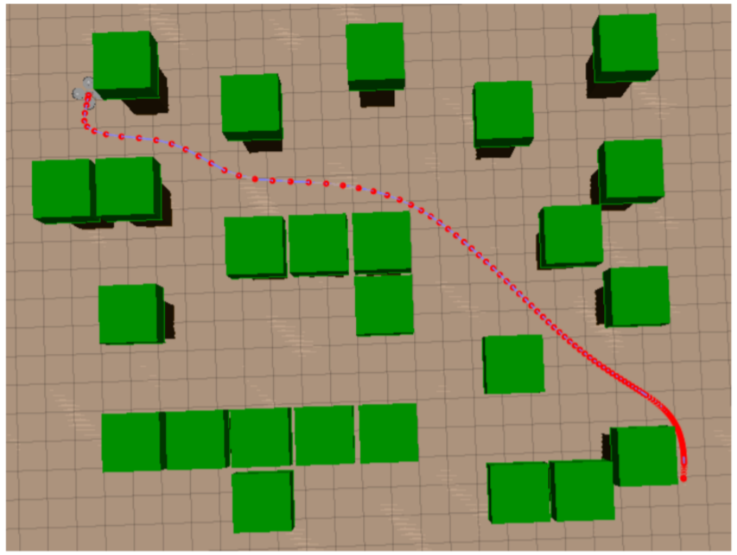


A Naïve Random Search Based Approach

What if we **incrementally build up a graph** to explore our map and get from the start state to the goal state

Algorithm (input: s_0, s_{goal}, initial state graph G)

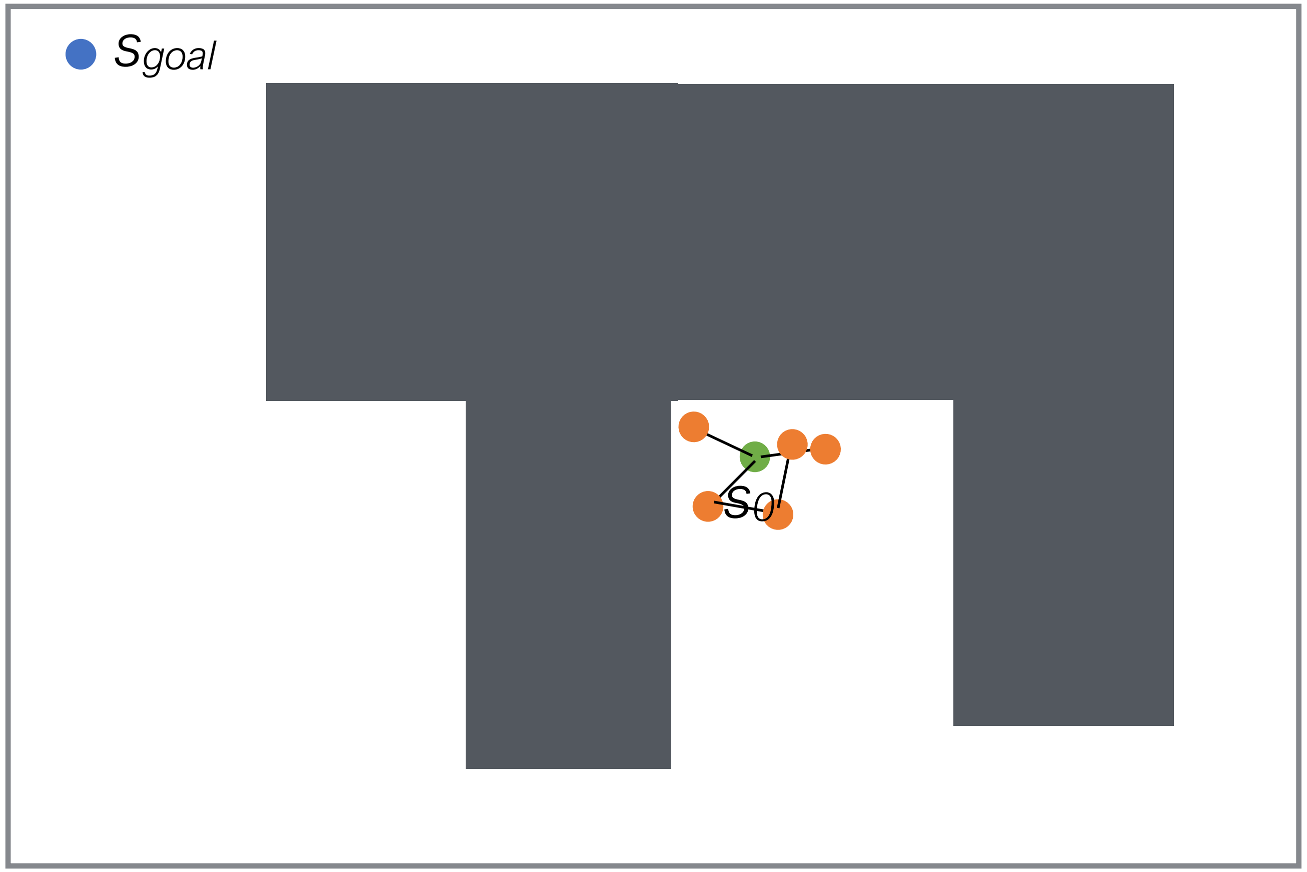
- Pick a random state $s \in G$
- Apply random action a
- Add resulting state s' to G
- Repeat until G has a path from s_0 to s_{goal}



A Naïve Random Search Based Approach

Algorithm (input: s_0, s_{goal}, initial state graph G)

- Pick a random state $s \in G$
- Apply random action a
- Add resulting state s' to G
- Repeat until G has a path from s_0 to s_{goal}



A Naïve Random Search Based Approach

What if we **incrementally build up a graph** to explore our map and get from the start state to the goal state

Algorithm (input: s_0, s_{goal}, initial state graph G)

- Pick a random state $s \in G$
- Apply random action a
- Add resulting state s' to G
- Repeat until G has a path from s_0 to s_{goal}

Probabilistically complete: As iterations go to infinity, probability that G contains a solution goes to 1!

A Naïve Random Search Based Approach

What if we **incrementally build up a graph** to explore our map and get from the start state to the goal state

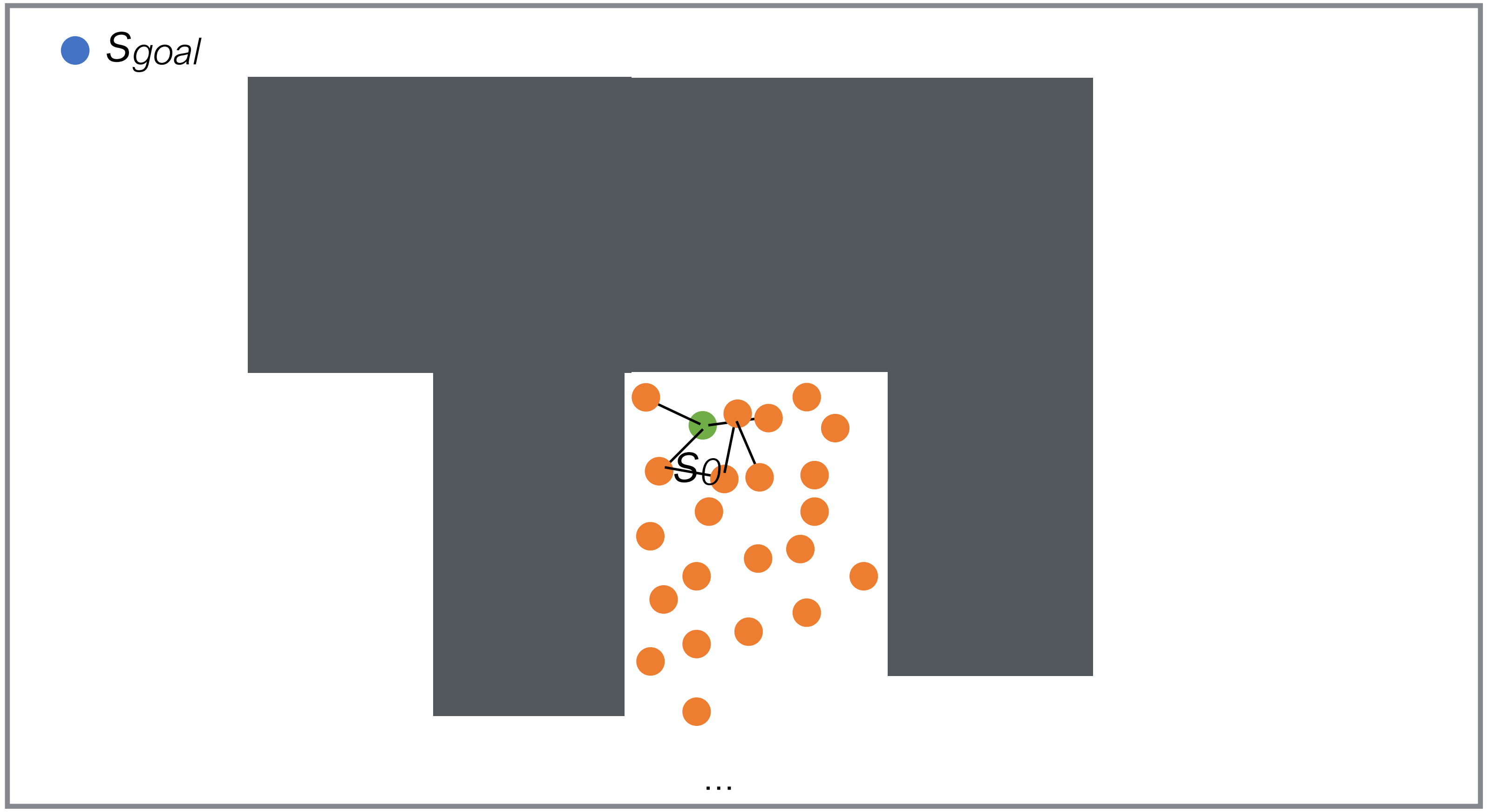
Algorithm (input: s_0, s_{goal}, initial state graph G)

- Pick a random state $s \in G$
- Apply random action a
- Add resulting state s' to G
- Repeat until G has a path from s_0 to s_{goal}

Probabilistically complete: As iterations go to infinity, probability that G contains a solution goes to 1!

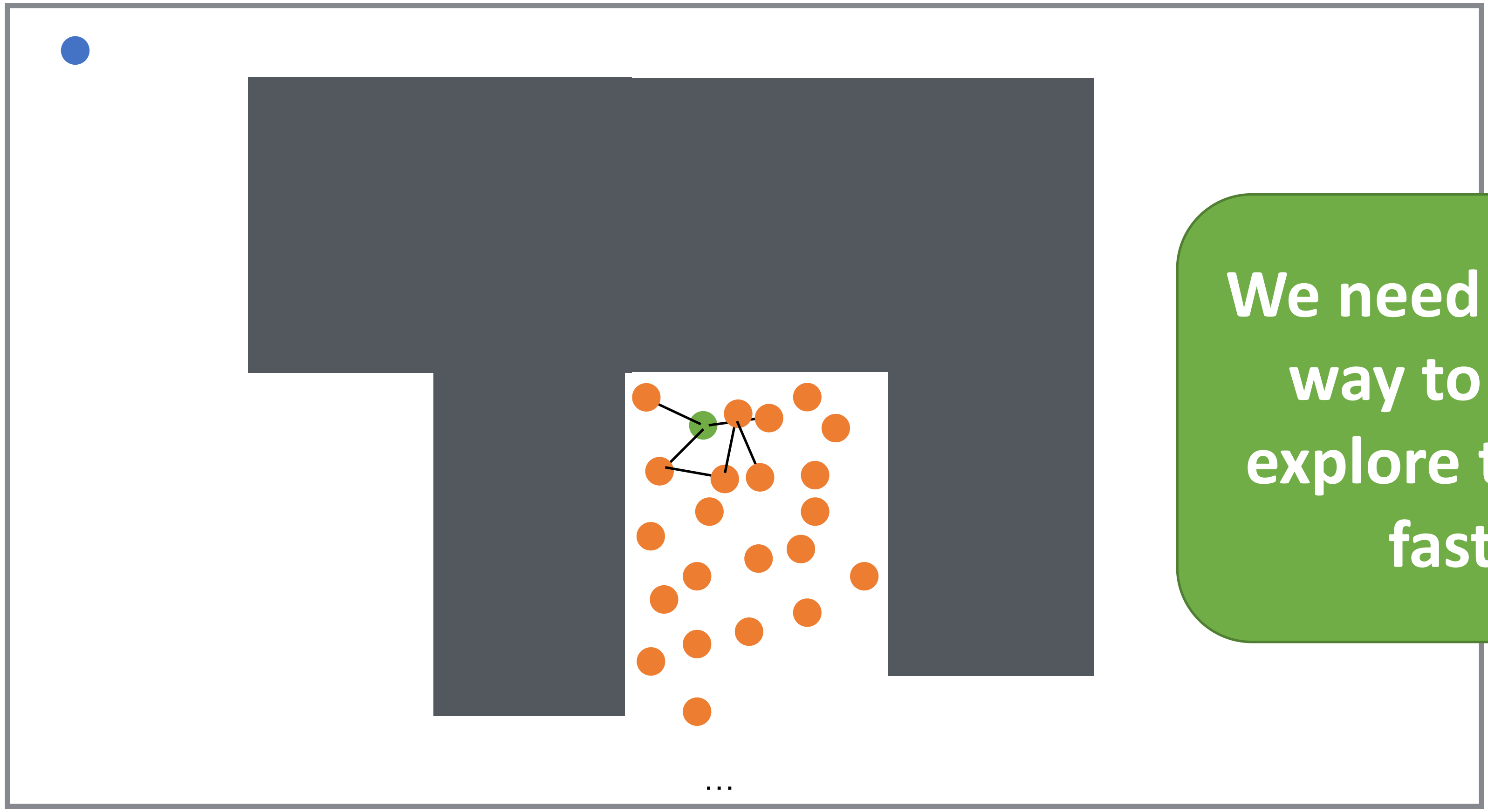
Q: What's the problem with this?

A Naïve Random Search Based Approach



Lots of samples close to your initial state —> slow!

A Naïve Random Search Based Approach



We need to find a way to better explore the map faster!

Lots of samples close to your initial state —> slow!

Robot Motion Planning with RRT

Naïve Random Search

Rapidly Exploring Random Trees (RRT)

Variants of RRT

Limitations of RRT

Rapidly Exploring Random Trees (RRTs)

Naïve (s_0, s_{goal}, initial state graph G)

- Pick a random state $s \in G$
- **Apply random action** a
- Add resulting state s' to G
- Repeat until G has a path from s_0 to s_{goal}

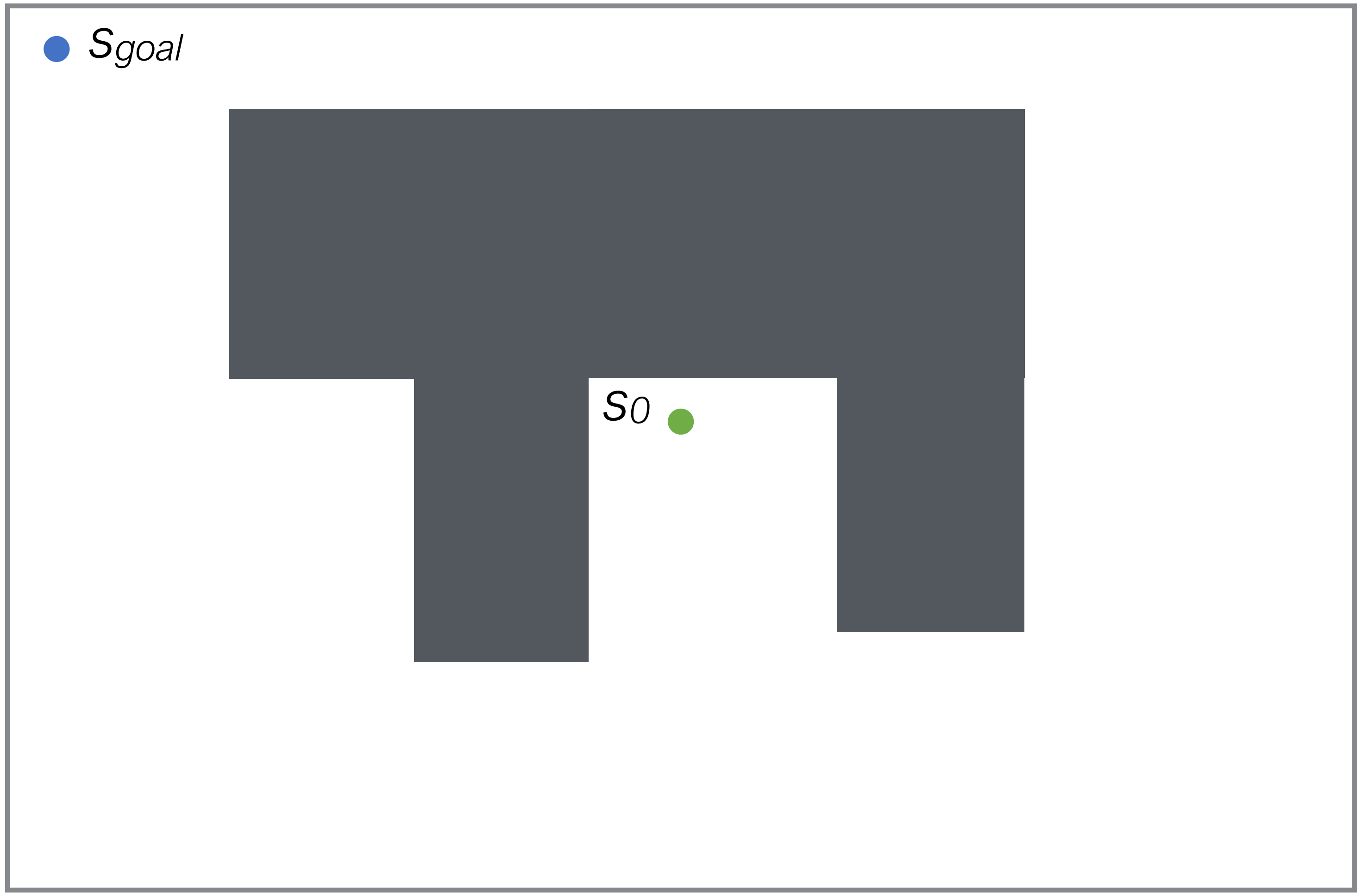
RRT (s_0, s_{goal}, initial state tree T)

- Sample a random state $s \in S$
- **Find closest state** $s_c \in T$
- **Extend** s_c **toward** s
- Add resulting state s' to T
- Repeat until T contains a path from s_0 to s_{goal}

Rapidly Exploring Random Trees

Algorithm (**input: s_0, s_{goal}, initial state tree T**)

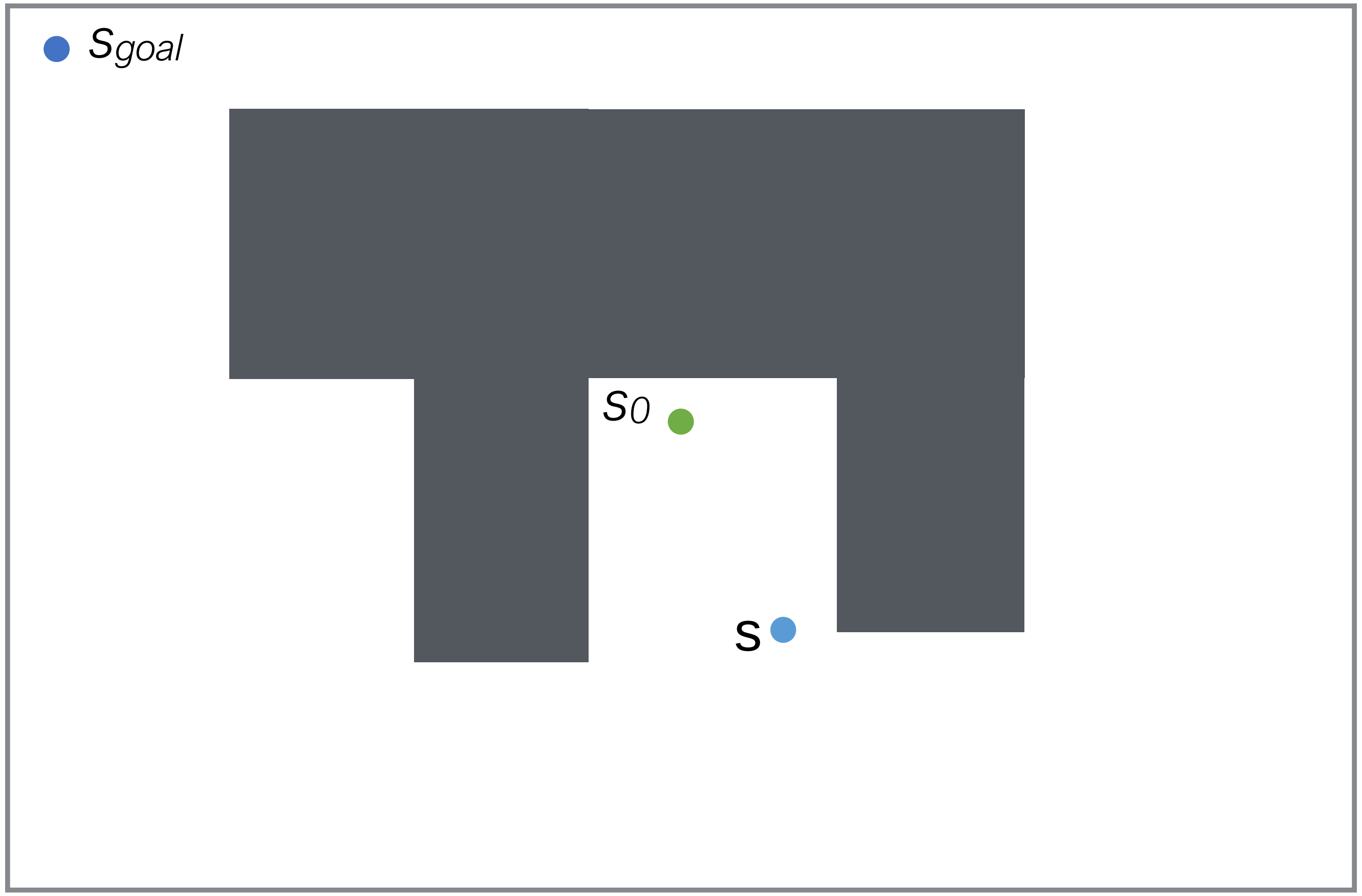
- Sample a random state $s \in S$
- Find closest state $s_c \in T$
- Extend s_c toward s
- Add resulting state s' to T
- Repeat until T contains a path from s_0 to s_{goal}



Rapidly Exploring Random Trees

Algorithm (input: s_0, s_{goal}, initial state tree T)

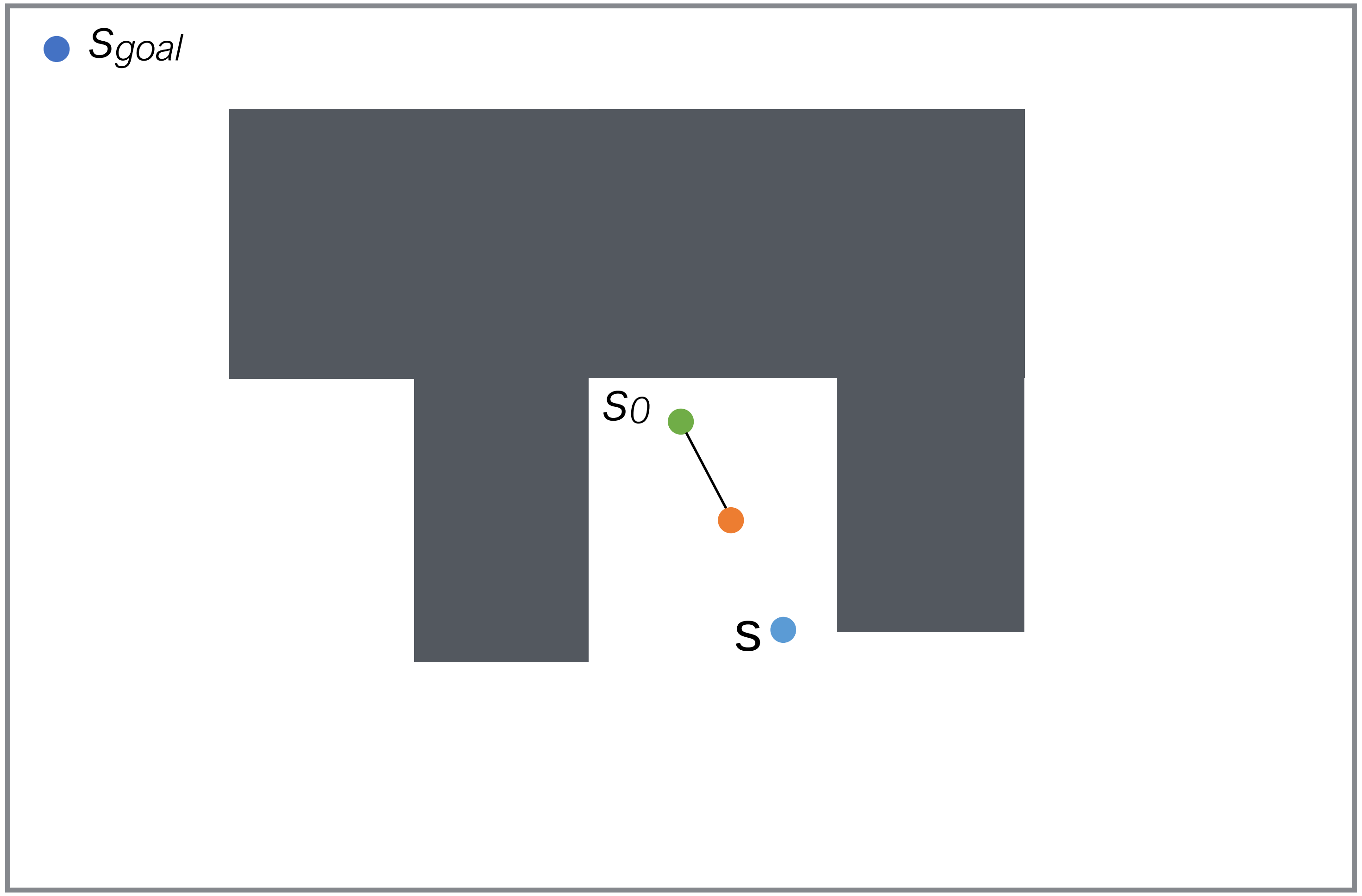
- **Sample a random state s ∈ S**
- Find closest state $s_c \in T$
- Extend s_c toward s
- Add resulting state s' to T
- Repeat until T contains a path from s_0 to s_{goal}



Rapidly Exploring Random Trees

Algorithm (input: s_0, s_{goal}, initial state tree T)

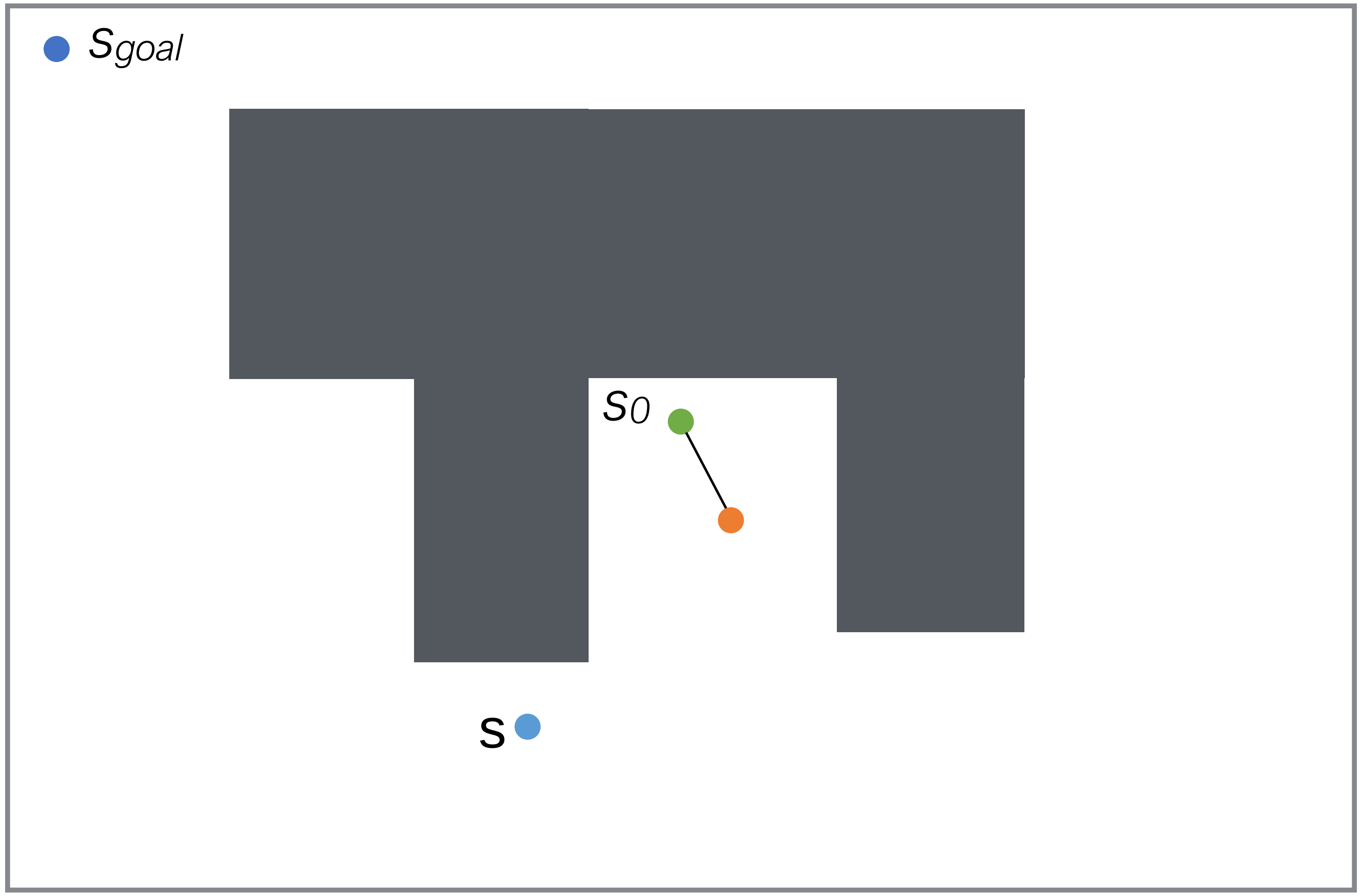
- Sample a random state $s \in S$
- **Find closest state $s_c \in T$**
- **Extend s_c toward s**
- **Add resulting state s' to T**
- Repeat until T contains a path from s_0 to s_{goal}



Rapidly Exploring Random Trees

Algorithm (input: s_0, s_{goal}, initial state tree T)

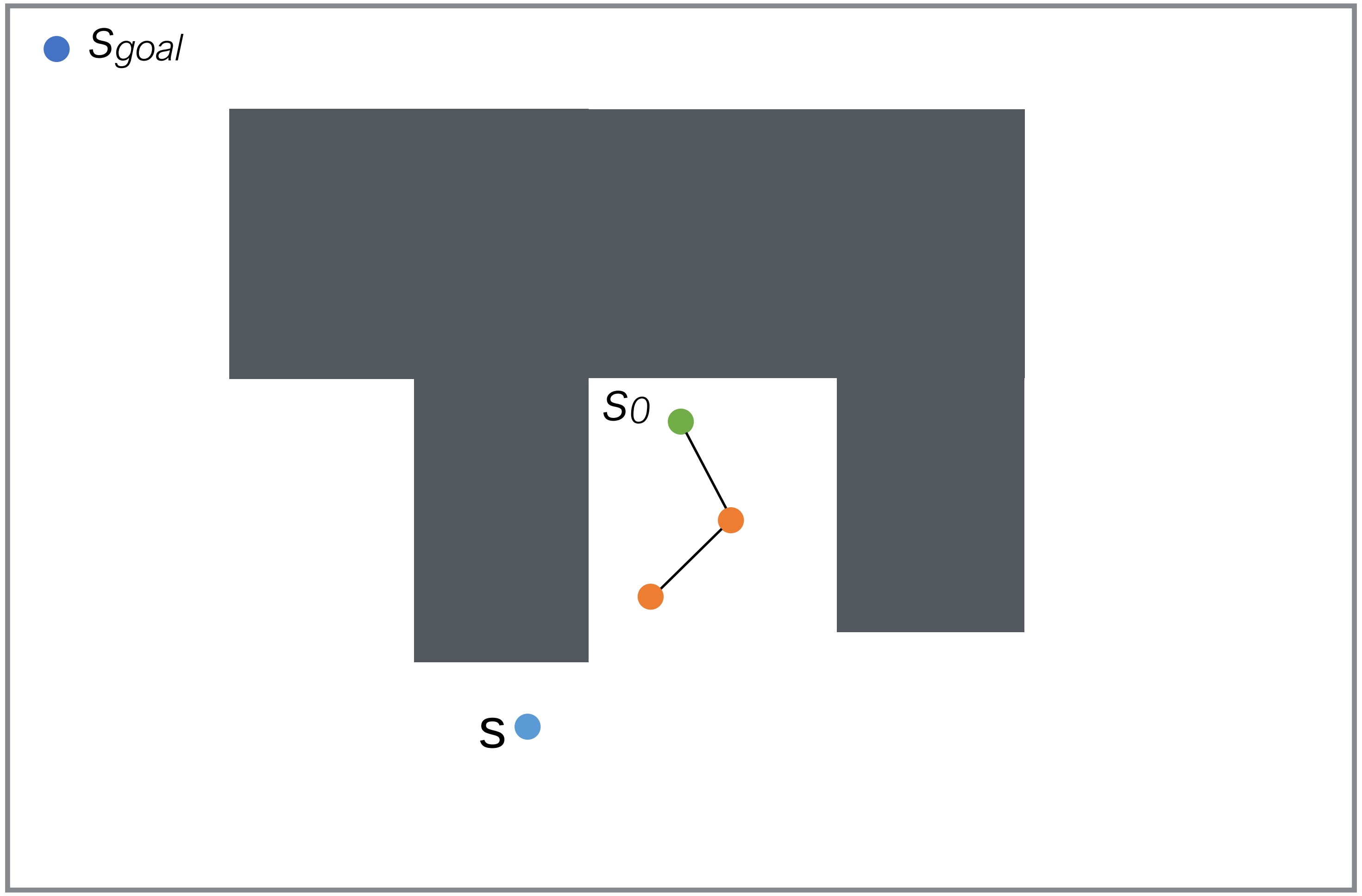
- **Sample a random state s ∈ S**
- Find closest state $s_c \in T$
- Extend s_c toward s
- Add resulting state s' to T
- Repeat until T contains a path from s_0 to s_{goal}



Rapidly Exploring Random Trees

Algorithm (input: s_0, s_{goal}, initial state tree T)

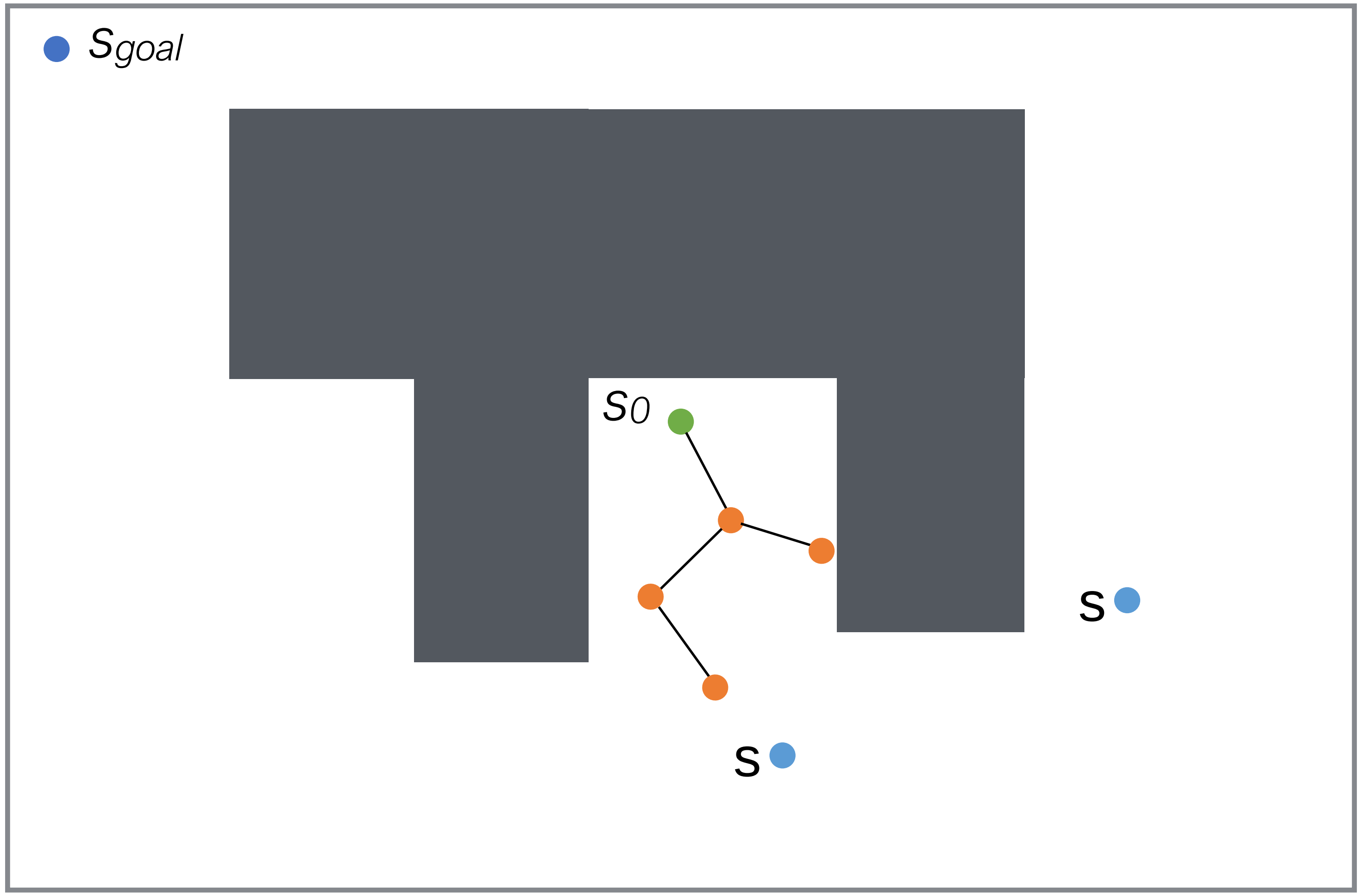
- Sample a random state $s \in S$
- **Find closest state $s_c \in T$**
- **Extend s_c toward s**
- **Add resulting state s' to T**
- Repeat until T contains a path from s_0 to s_{goal}



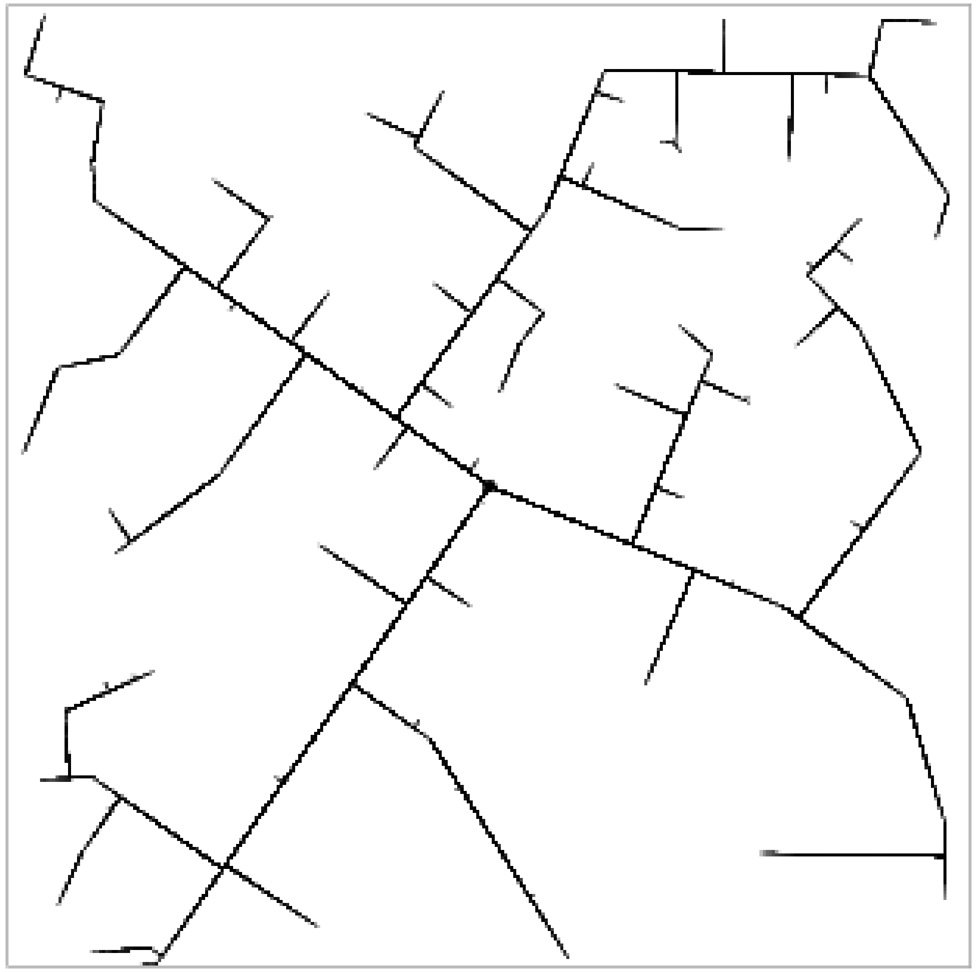
Rapidly Exploring Random Trees

Algorithm (input: s_0, s_{goal}, initial state tree T)

- Sample a random state $s \in S$
- Find closest state $s_c \in T$
- Extend s_c toward s
- Add resulting state s' to T
- **Repeat until T contains a path from s_0 to s_{goal}**



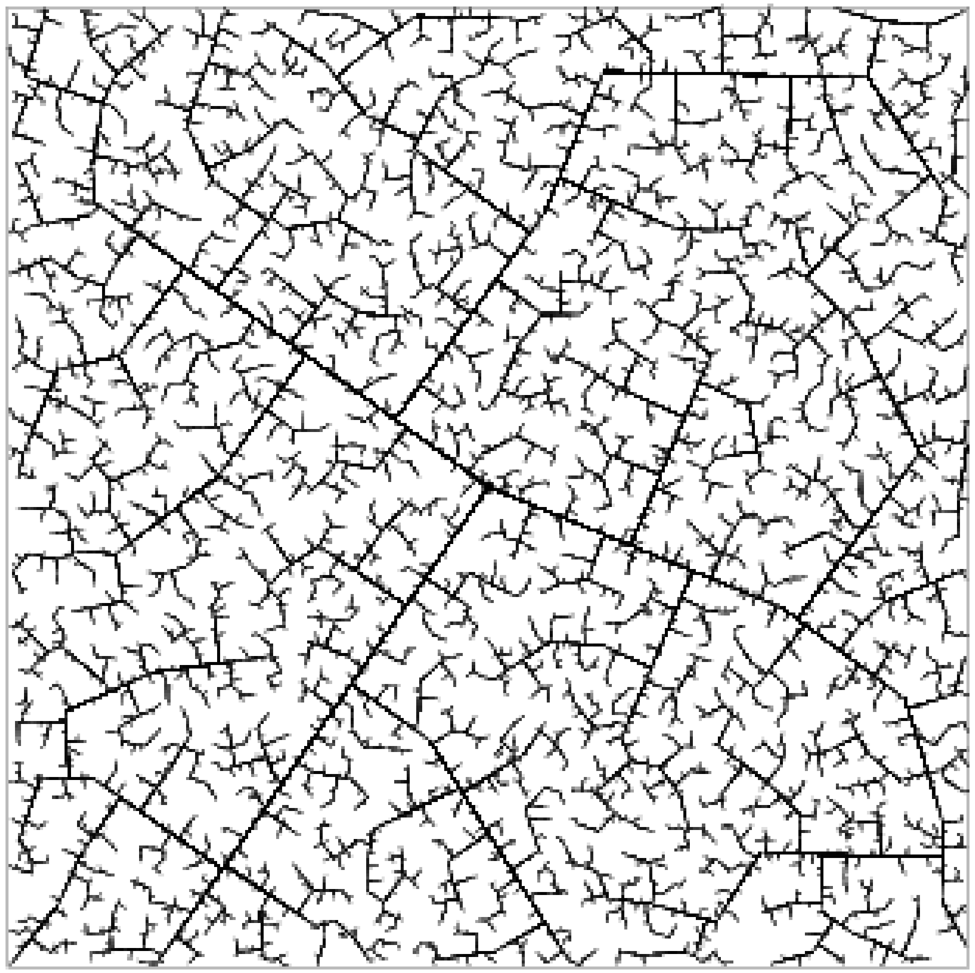
Rapidly Exploring Random Trees (RRTs)


45 iterations

RRT (s_0, s_{goal}, initial state tree T)

- Sample a random state $s \in S$
- **Find closest state $s_c \in T$**
- **Extend s_c toward s**
- Add resulting state s' to T
- Repeat until T contains a path from s_0 to s_{goal}

Rapidly Exploring Random Trees (RRTs)


2345 iterations

RRT (s_0, s_{goal}, initial state tree T)

- Sample a random state $s \in S$
- **Find closest state $s_c \in T$**
- **Extend s_c toward s**
- Add resulting state s' to T
- Repeat until T contains a path from s_0 to s_{goal}

Properties of RRT

Key idea: **random sampling** will naturally **encourage exploration**

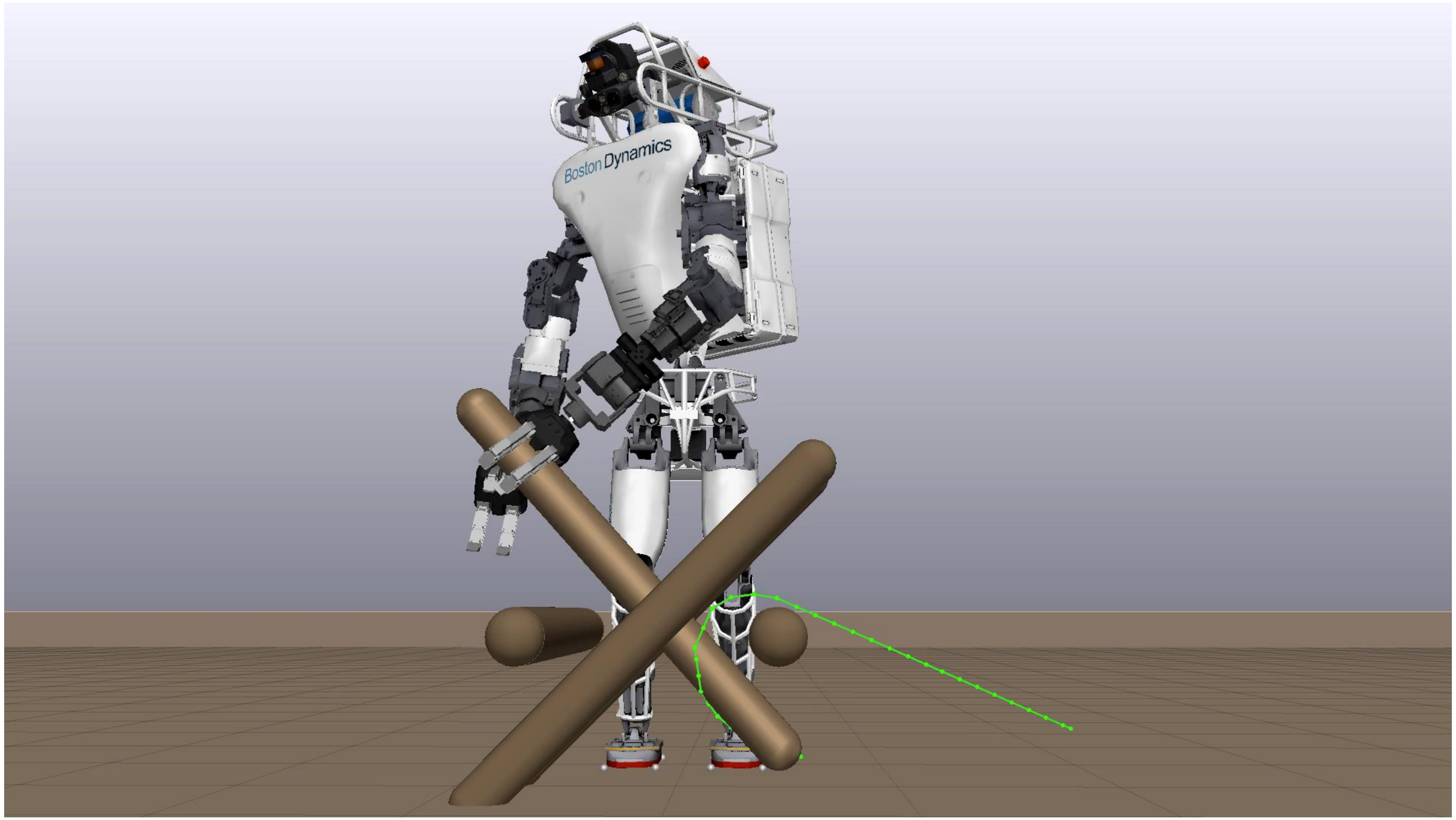
Properties of RRT

Key idea: **random sampling** will naturally **encourage exploration**

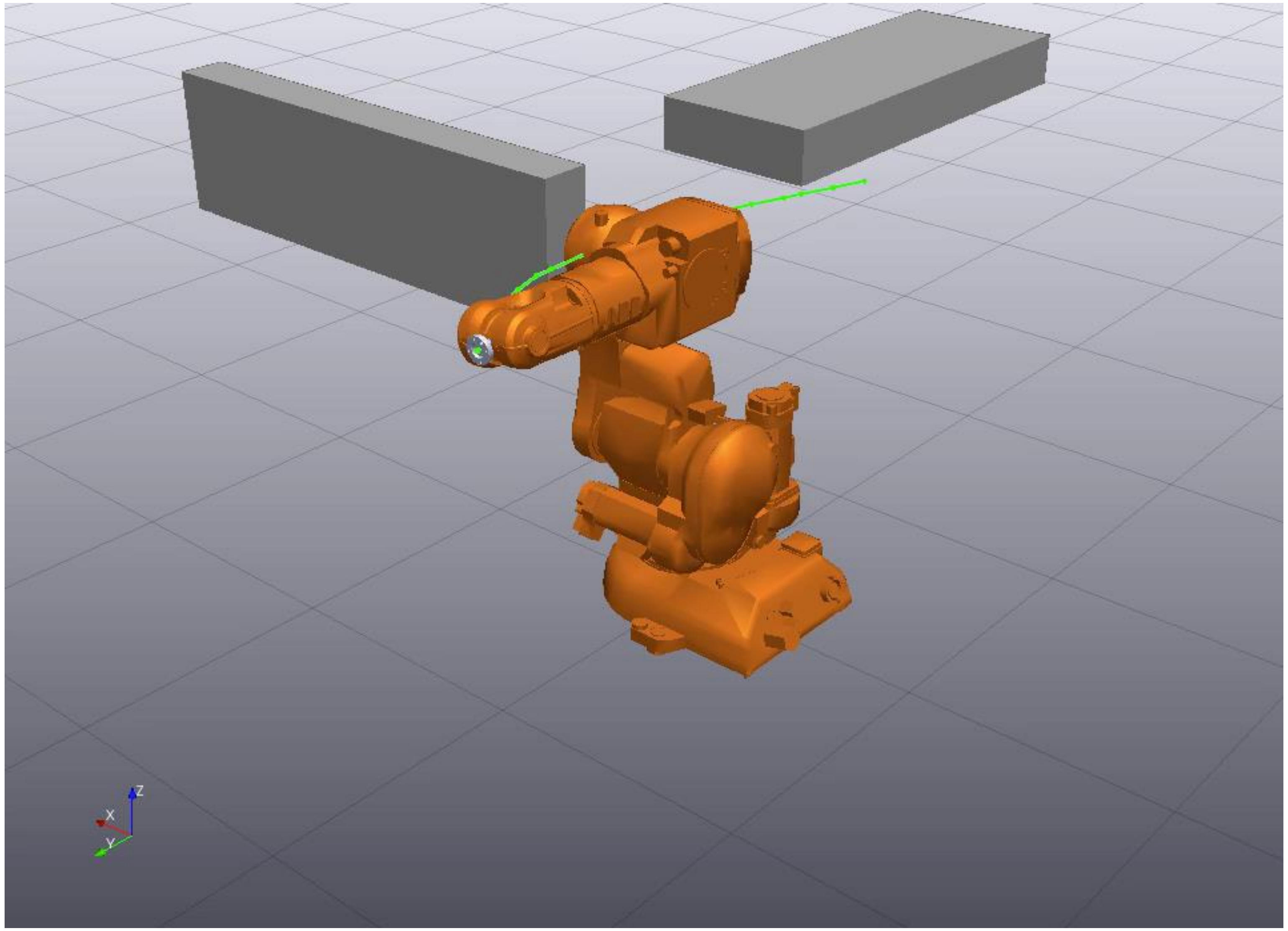
RRT is **probabilistically complete**!

- If there’s a solution it will find it eventually
- Can still be slow for some problems, but it is faster than naive action sampling approach

RRT often works really well in practice



RRT often works really well in practice



Robot Motion Planning with RRT

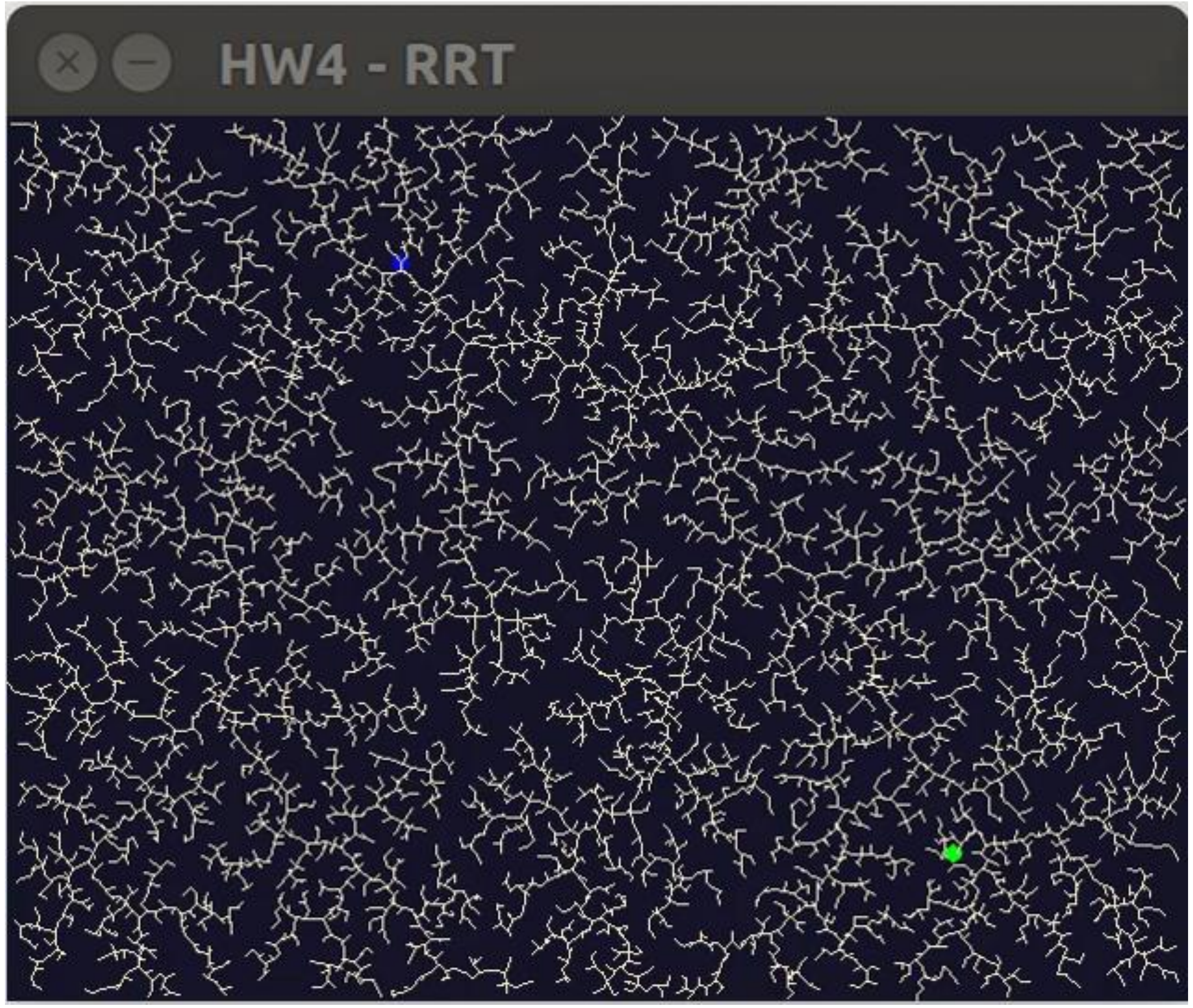
Rapidly Exploring Random Trees – Variants

Standard RRT (input: s_0, s_{goal}, initial state tree T)

- Sample a random state $s \in S$
- Find closest state $s_c \in T$
- Extend s_c toward s
- Add resulting state s' to T
- Repeat until T contains a path from s_0 to s_{goal}

Rapidly Exploring Random Trees – Variants

Rapidly Exploring Random Trees – Variants



Q: What can we change to make this better?

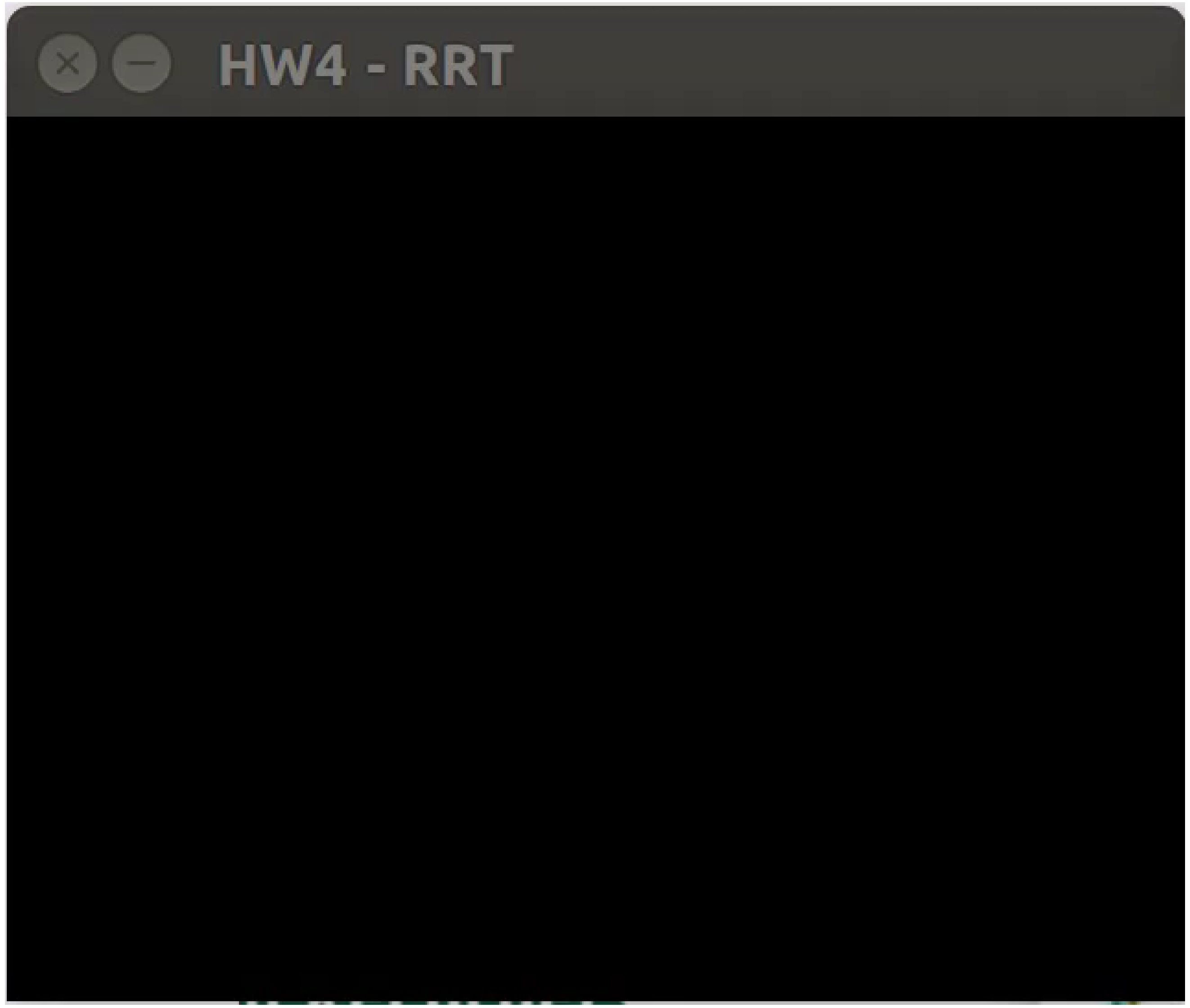
Rapidly Exploring Random Trees – Variants

RRT with Goal Directed Sampling (input: s_0, s_{goal}, initial state tree T)

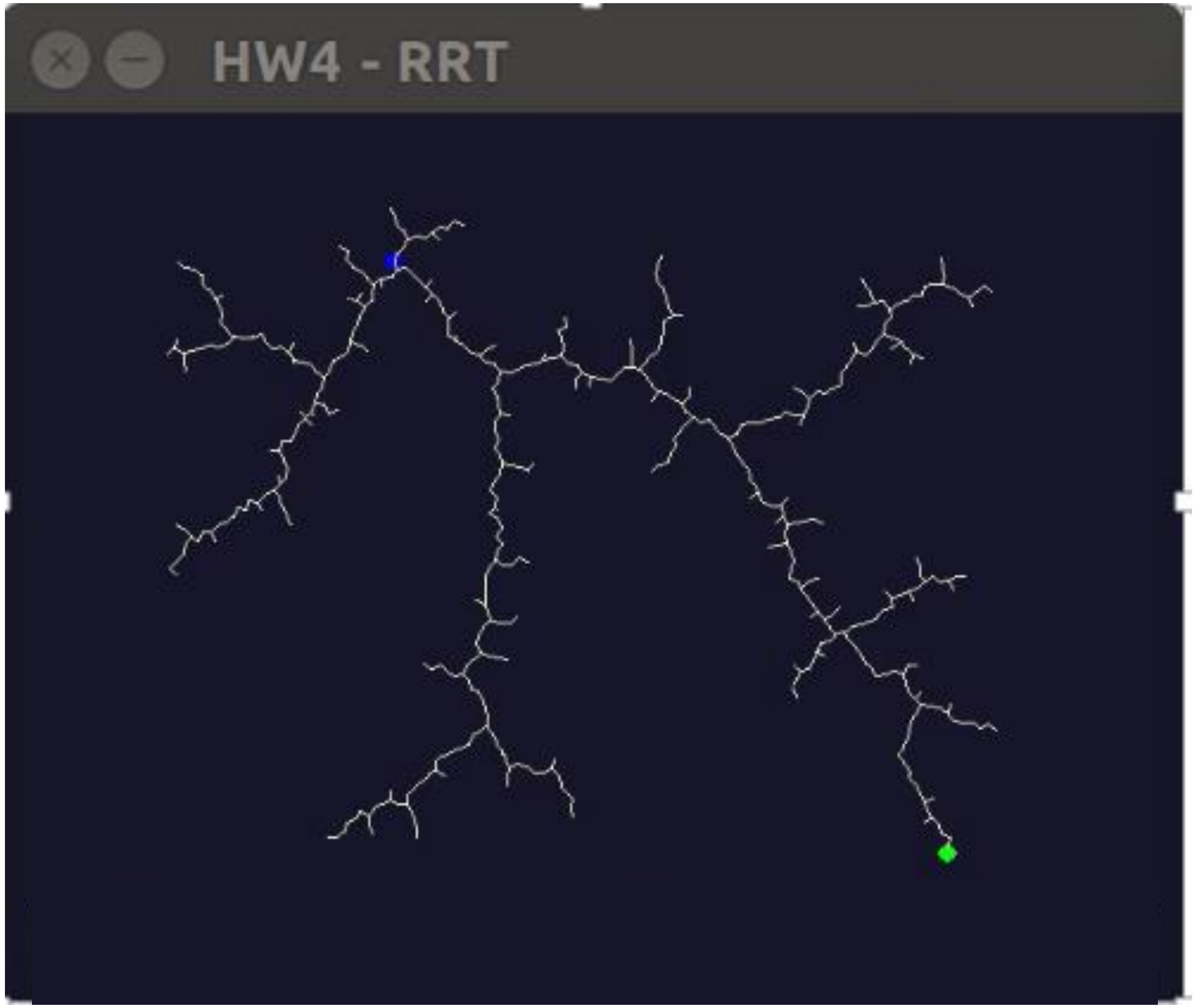
- Sample a random state $s \in S$ **with probability $(1-p)$ and with probability p sample the goal**
- Find closest state $s_c \in T$
- Extend s_c toward s
- Add resulting state s' to T
- Repeat until T contains a path from s_0 to s_{goal}

Intuition: instead of "stumbling" upon the solution, bias the tree growth in the goal direction

Rapidly Exploring Random Trees – Variants



Rapidly Exploring Random Trees – Variants



Of course again we have a tradeoff in exploration vs. goal direction!

Rapidly Exploring Random Trees – Variants

Goal Directed RRT

RRT*

Bidirectional RRT

GPU-RRT*

LQR-RRT*

Rapidly Exploring Random Trees – Variants

Goal Directed RRT

PQ-RRT*

RRT*

RRT-Connect

Kinodynamic RRT*

Bidirectional RRT

Informed RRT*

RRT^X

LBT-RRT

LQR-RRT*

GPU-RRT*

T-RRT

Particle RRT

RRT* Smart

Robot Motion Planning with RRT

Naïve Random Search

Rapidly Exploring Random Trees (RRT)

Variants of RRT

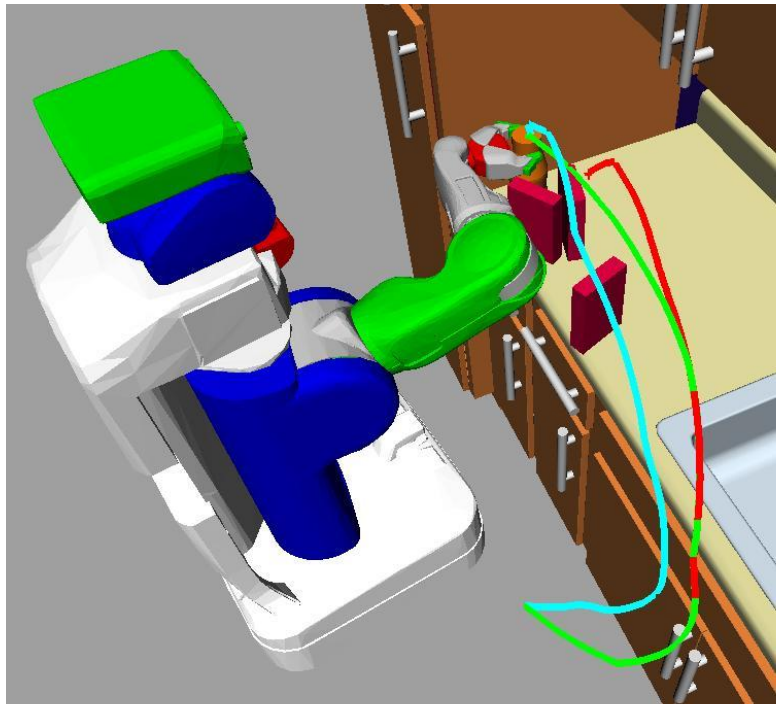
Limitations of RRT

Sometimes Paths are Weird (Not Optimal)

Another tradeoff!

Configuration Space (aka where we plan)

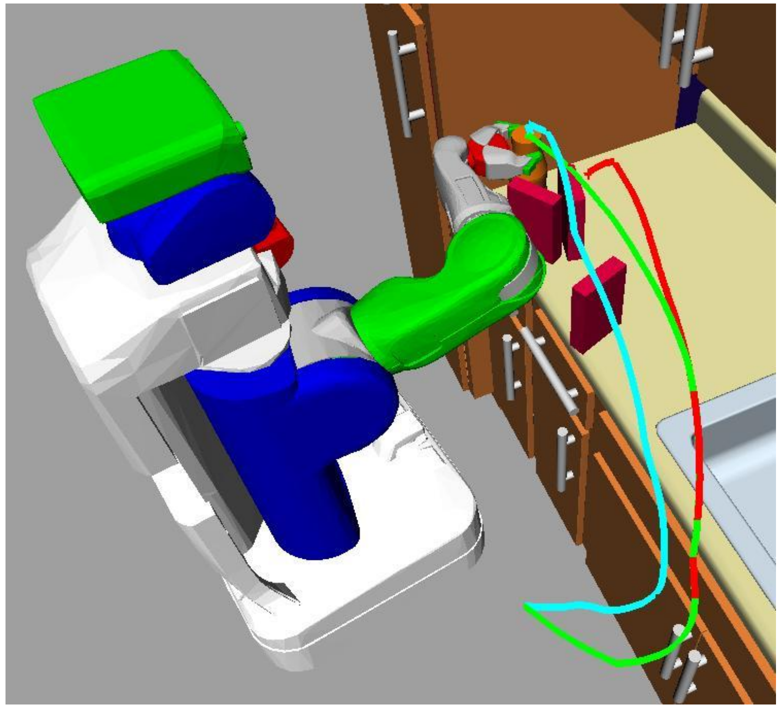
- So far we have been exploring RRT in 2D but robots don’t exist in a 2D world!



Configuration Space (aka where we plan)

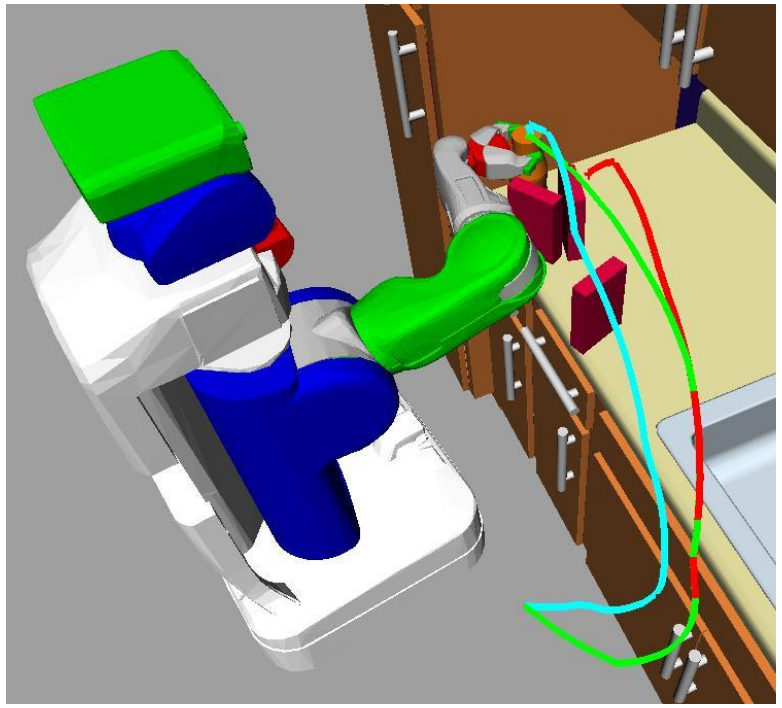
- So far we have been exploring RRT in 2D but robots don't exist in a 2D world!

- Task space: the 6D workspace of the robot
  - E.g., the pose (x,y,z,roll,pitch,yaw) of the robot's hand or an object



Configuration Space (aka where we plan)

- So far we have been exploring RRT in 2D but robots don't exist in a 2D world!
- Task space: the 6D workspace of the robot
  - E.g., the pose (x,y,z,roll,pitch,yaw) of the robot's hand or an object
- Configuration space: the *n*-dimensional space of joint angles + robot world position
  - Vector



Planning (in configuration space) is hard!

How many dimensions is the configuration space for Atlas?

Planning (in configuration space) is hard!

How many dimensions is the configuration space for Atlas?

(2 ankles + 2 knees + 2 hips (in 2 directions) + torso + 2 shoulders (in 2 directions) + 2 elbows + 2 wrists + 6dof pose of com) = ~ 24 variables

Planning (in configuration space) is hard!

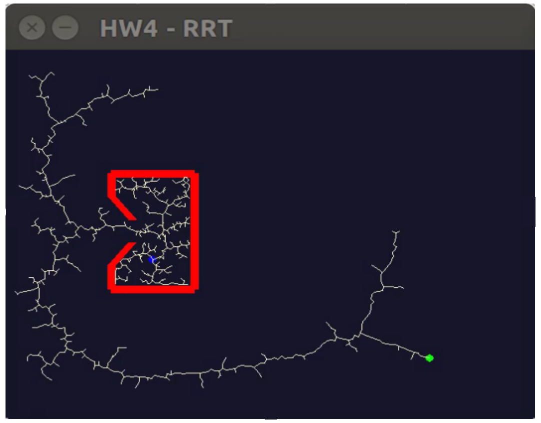
Sampling in 24+ dimensions can be very slow!

(2 ankles + 2 knees + 2 hips (in 2 directions) + torso + 2 shoulders (in 2 directions) + 2 elbows + 2 wrists + 6dof pose of com) = ~ 24 variables

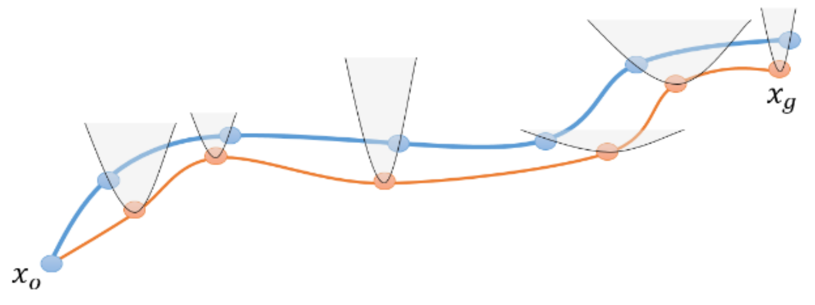
Remember its all about the **tradeoffs**!

CafeX, the San Francisco based startup, has hired you to upgrade the motion planning software for their robot to make it faster without sacrificing coffee quality.

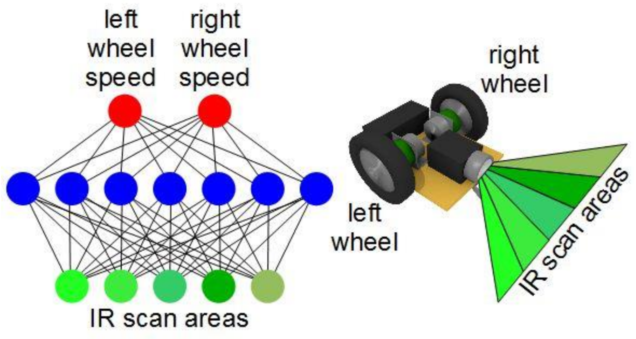
Random Search (RRT)



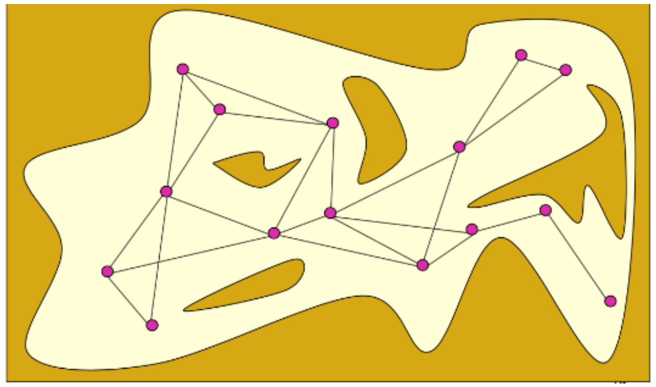
Optimal Local Search



Machine Learning



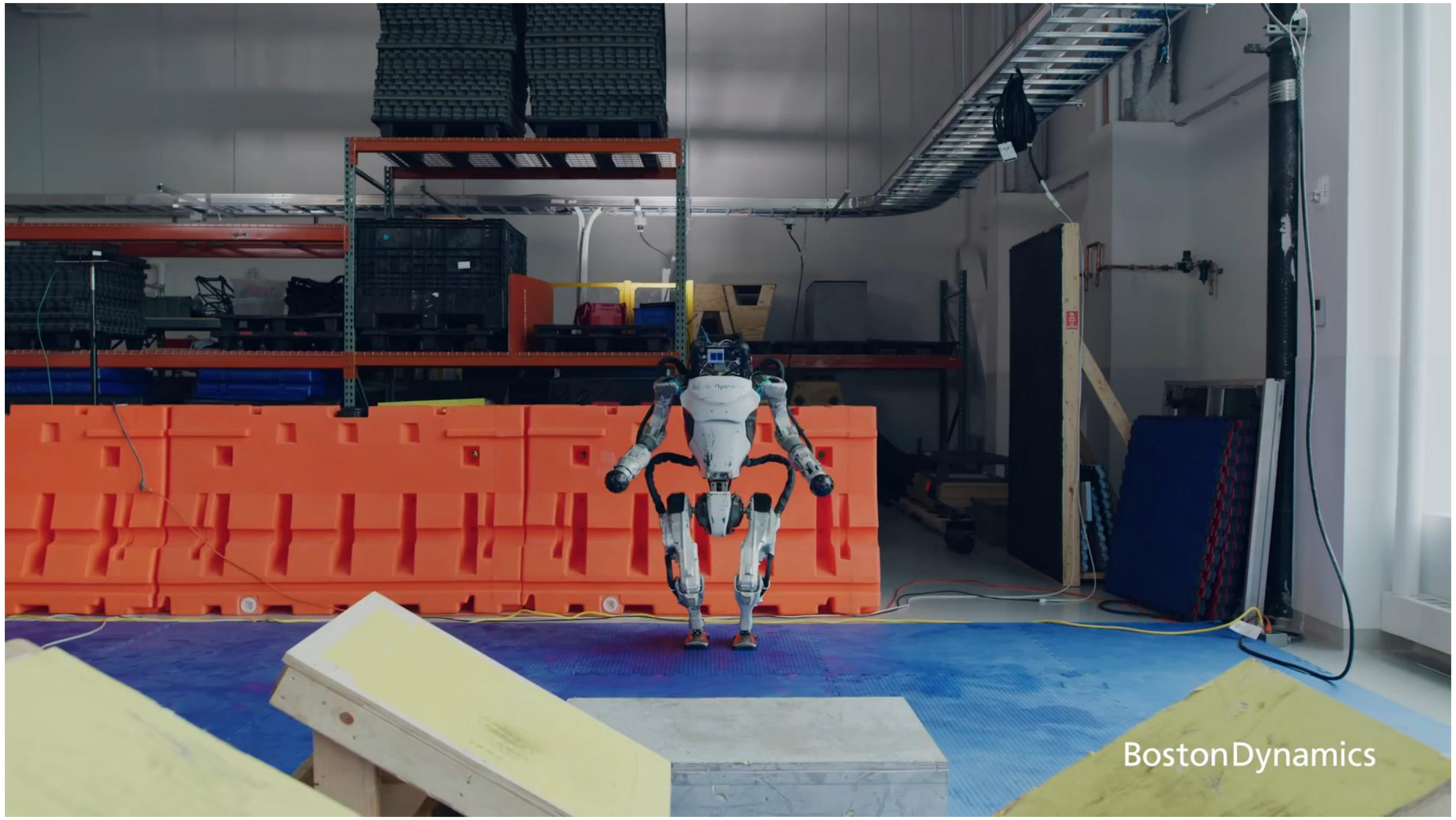
Random Search (PRM)



Learning Goals for Today

1. Learn some of the **language** of robotics
2. Understand the **importance of tradeoffs** in the selection of robotics algorithms for **real-world deployments**
3. Gain practice in exploring the **attributes of classes of algorithms** through an example

Thank You! Questions? I’d love your feedback!



Feedback Link: https://bit.ly/Brian-Simmons-21