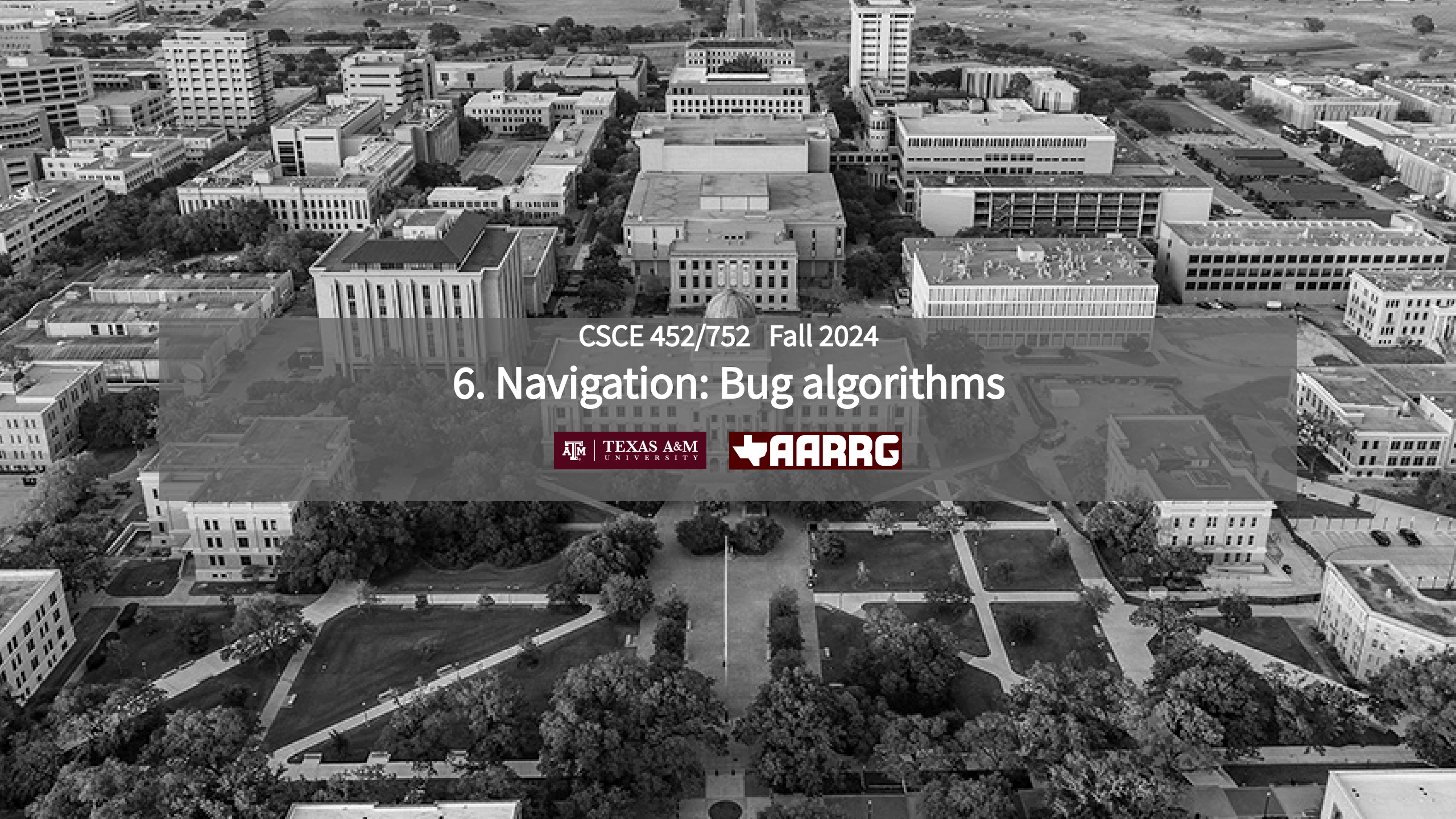


CSCE 452/752 Fall 2024

6. Navigation: Bug algorithms

TEXAS A&M UNIVERSITY

AARRG

Overview

Definition

Navigation is the process of moving a robot through its environment to a **goal state** while avoiding obstacles.

This is one of the fundamental problems for mobile robots.

We'll see several types of algorithms for this problem.

Key Question 1

How do each of these algorithms work?

Key Question 2

What are the differences in when each of these algorithms can be applied?

Bug algorithms

Introduction

Definition

Bug algorithms are a general class of navigation algorithms that do not require the robot to know the obstacles ahead of time.

Defining characteristic: Only **local** information about the obstacles is available.

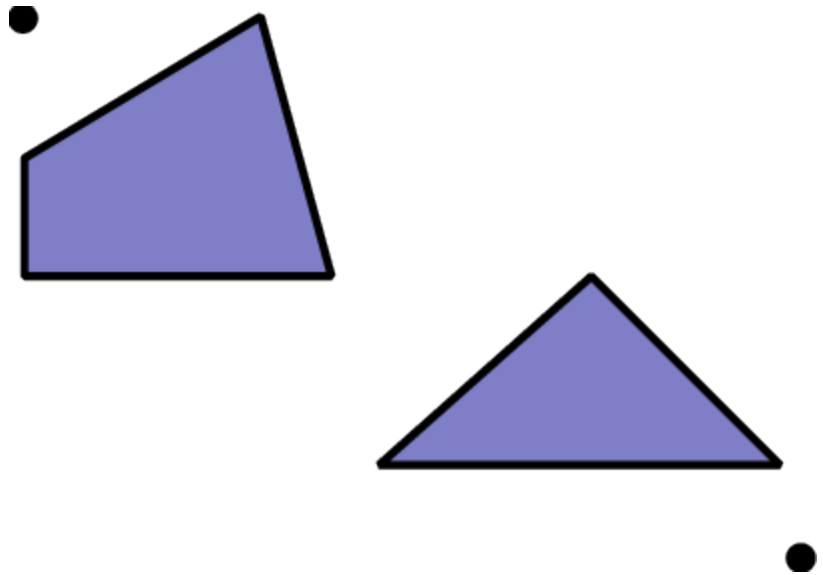
- No long-range sensing.
- No detailed representations of the obstacle shapes.

Motivating problem: Finding the Eiffel Tower

Sensor model

Bug algorithms assume that the robot has some combination of sensors that allow it to perform two basic actions:

1. Sense the direction and distance to the goal.
2. Follow nearby obstacle boundaries and measure the distance traveled.



An optimistic but incorrect algorithm

Here's a first crack at solving the problem:

- Move toward the goal until reaching an obstacle.
- Follow obstacle boundary until it's possible to move directly toward the goal.
- Repeat.

When does this fail?

What about an algorithm that can **guarantee** reaching the goal?

Online vs. Offline

Bug algorithms are **online planning** algorithms.

- Planning and execution are interleaved. The robot decides what to do “on-the-fly”.
- Contrast to: **Offline planning**, in which the robot forms a complete plan before moving.

Bug1

The simplest correct Bug algorithm is called Bug1.

- Move toward the goal until reaching an obstacle.
- Turn left and follow the obstacle boundary.
- Follow the entire boundary of that obstacle. Keep track of which point on the boundary is closest to the goal.
- Return to this point, following the obstacle boundary in whichever direction is shorter.
- Repeat until the robot reaches the goal.

Analysis of Bug1

How long a path does Bug1 generate?

Analysis of Bug1

How long a path does Bug1 generate?

- D: distance from start to goal
- M: number of obstacles
- p_i: perimeter of obstacle i

The path length is bounded by:

$$D + \frac{3}{2} \sum_{i=1}^{M} p_i$$

Hit points and leave points

As the robot follows an obstacle boundary, define:

- The **hit point** H is the point at which the robot reached the obstacle.
- The **leave point** L is the point at which the robot leaves the obstacle.

Bug2

Another, similar algorithm is called Bug2:

- Move toward the goal until reaching an obstacle.
- Follow this obstacle until it returns to the line connecting the start and goal positions.
  - If the robot can leave the obstacle toward the goal from this point, **and** it is closer to the goal than the hit point, then leave toward the goal.
  - Otherwise, continue around the obstacle.
- Repeat until the robot reaches the goal.

Analysis of Bug2

How long a path does Bug2 generate?

Analysis of Bug2

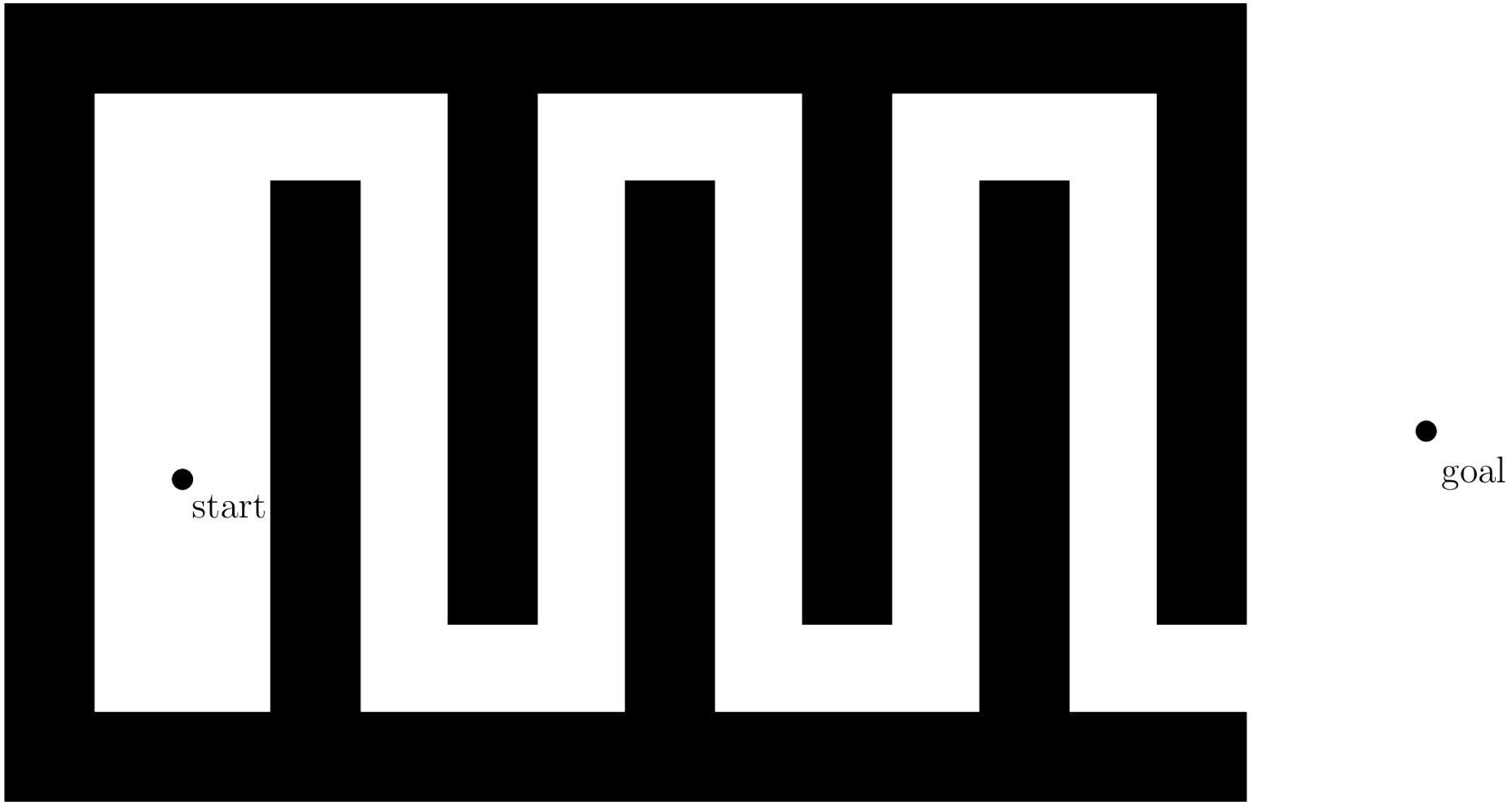
How long a path does Bug2 generate?

- D: distance from start to goal
- M: number of obstacles
- p_i: perimeter of obstacle i
- n_i: number of intersection points between obstacle i and the start-to-goal line.

The path length is bounded by:

$$D + \frac{1}{2} \sum_{i=1}^{M} n_i p_i$$

A bad case for Bug 2



Comparison between Bug1 and Bug2

Which algorithm is more “aggressive”? Which one is “conservative”?

Which algorithm provides better worst-case performance?

- Bug1 searches **exhaustively** for the best leave point.
- Bug2 is **opportunistic**, choosing the first suitable leave point it finds.

What if the goal cannot be reached?

What happens if there is no path to reach the goal?

Both algorithms can detect this.

- Bug1: When the closest point on an obstacle to the goal leads to an immediate collision.
- Bug2: When a complete cycle is made without finding a new leave point.