

CS-417 INTRODUCTION TO ROBOTICS AND INTELLIGENT SYSTEMS

Motion Planning

Outline

- Path Planning
 - Visibility Graph
 - Bug Algorithms
 - Potential Fields
 - Skeletons/Voronoi Graphs
 - C-Space

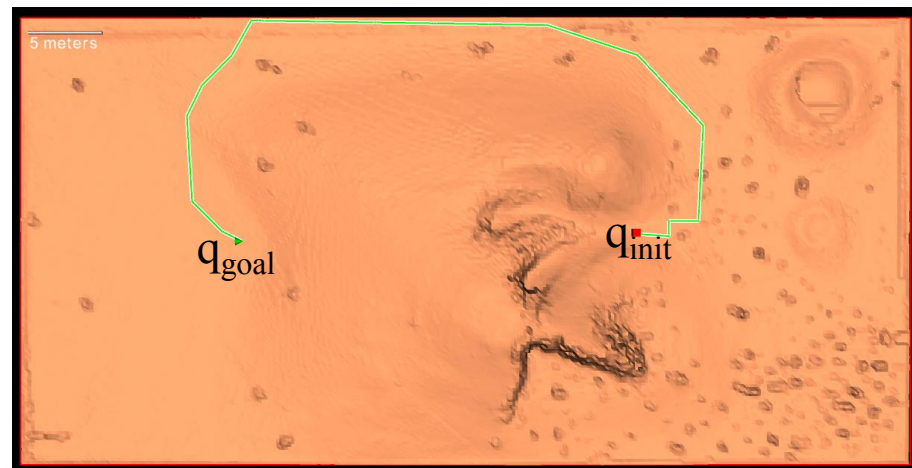
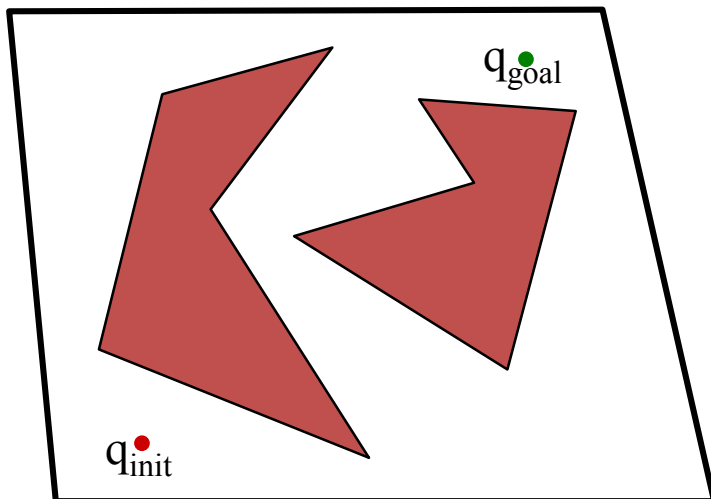


Motion Planning

- The ability to go from **A** to **B**
 - Known map – Off-line planning
 - Unknown Environment – Online planning
 - Static/Dynamic Environment

• q_{init}

• q_{goal}



Path Planning

World

Robot

Map



Path Planning

World

- Indoor/Outdoor
- 2D/2.5D/3D
- Static/Dynamic
- Known/Unknown
- Abstract (web)

Robot

Map



Path Planning

World

Robot

- Mobile
 - Indoor/Outdoor
 - Walking/Flying/Swimming
- Manipulator
- Humanoid
- Abstract

Map



Path Planning

World

Robot

Map

- Topological
- Metric
- Feature Based
- 1D, 2D, 2.5D, 3D



Path Planning

World

- Indoor/Outdoor
- 2D/2.5D/3D
- Static/Dynamic
- Known/Unknown
- Abstract (web)

Robot

- Mobile
 - Indoor/Outdoor
 - Walking/Flying/Swimming
- Manipulator
- Humanoid
- Abstract

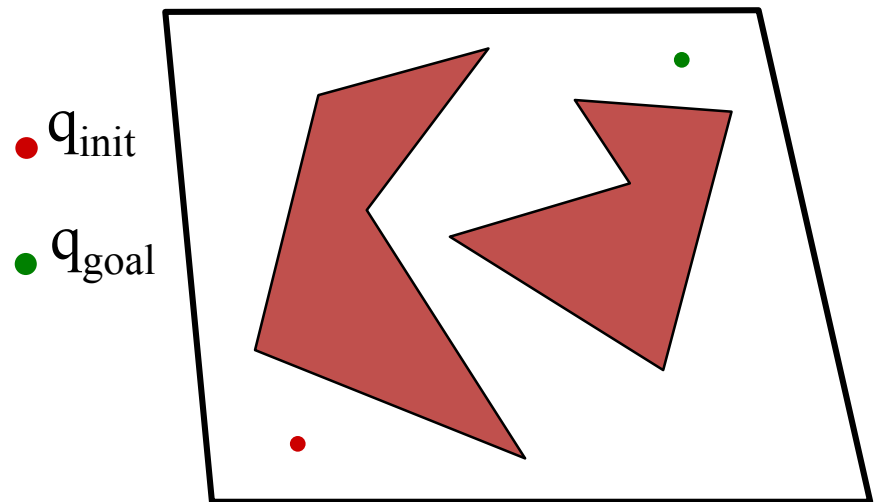
Map

- Topological
- Metric
- Feature Based
- 1D,2D,2.5D,3D



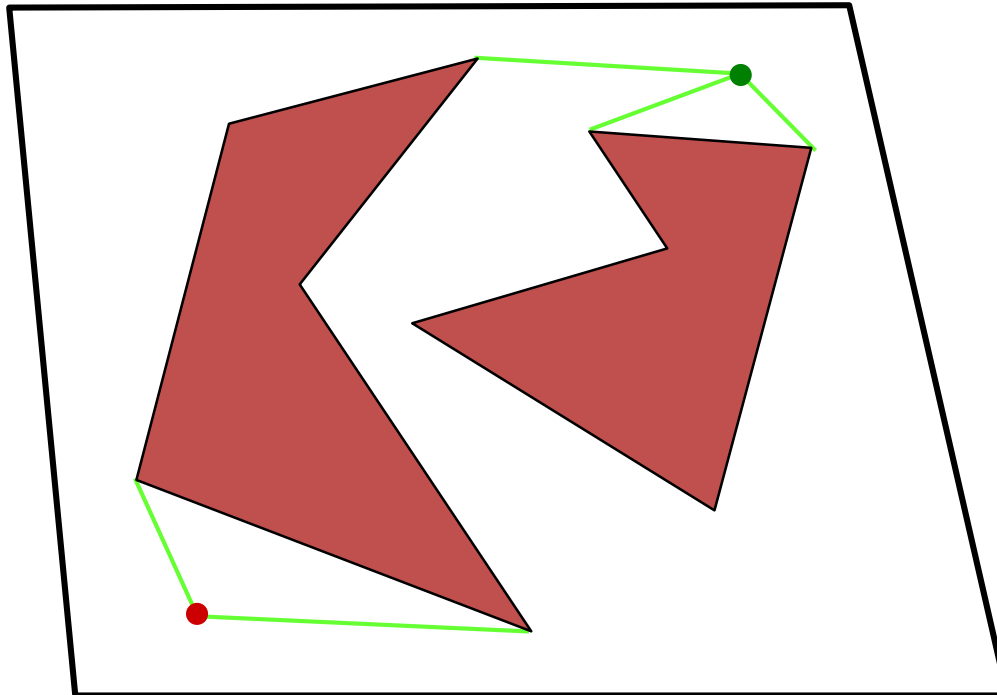
Path Planning: Assumptions

- Known Map
- Roadmaps (Graph representations)
- Polygonal Representation



Visibility Graph

- Connect Initial and goal locations with all the visible vertices

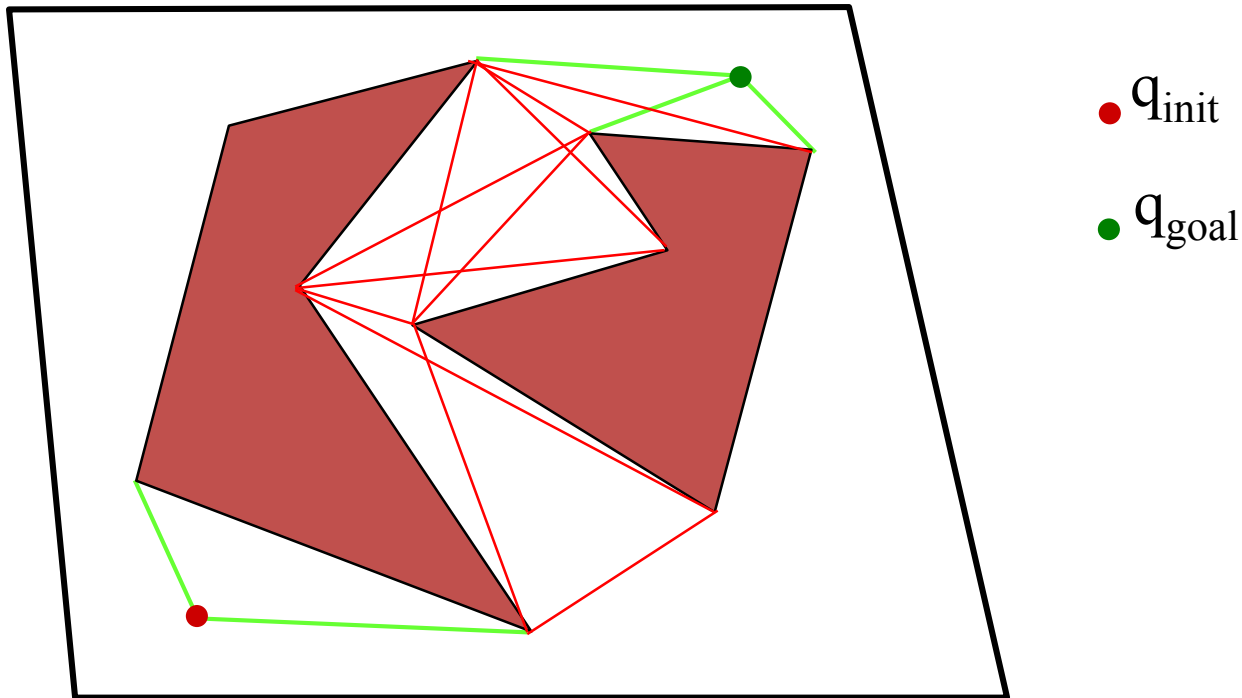


● q_{init}

● q_{goal}

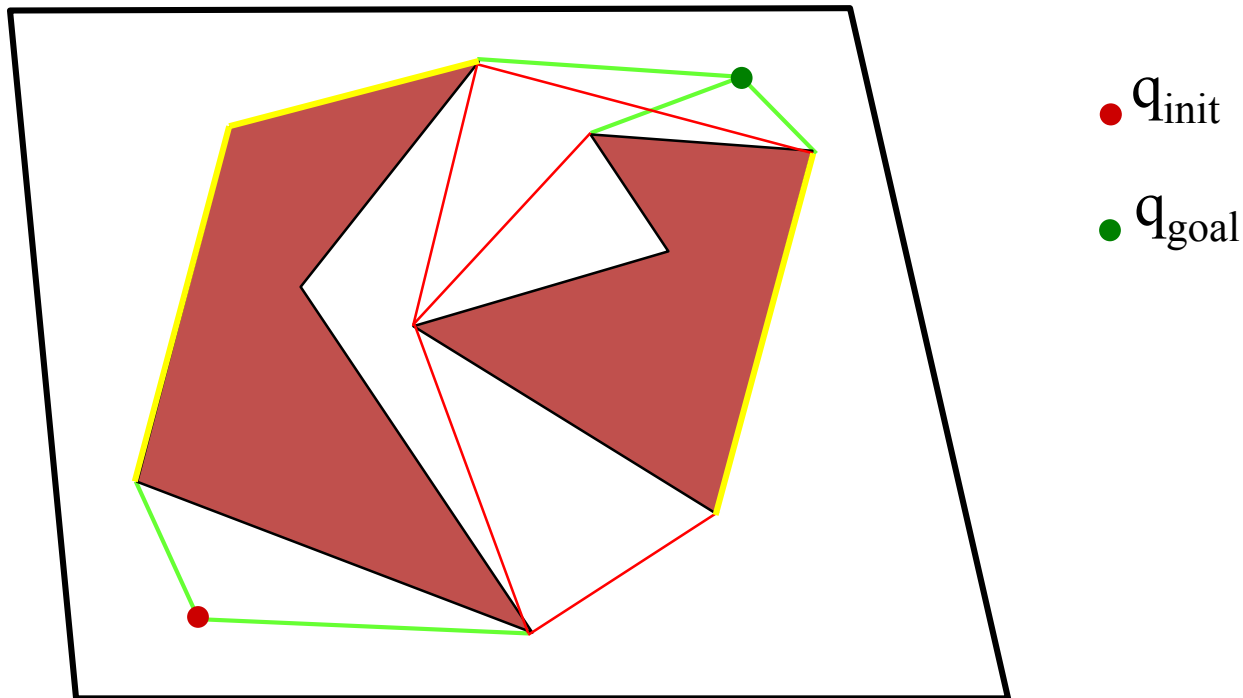
Visibility Graph

- Connect initial and goal locations with all the visible vertices
- Connect each obstacle vertex to every visible obstacle vertex



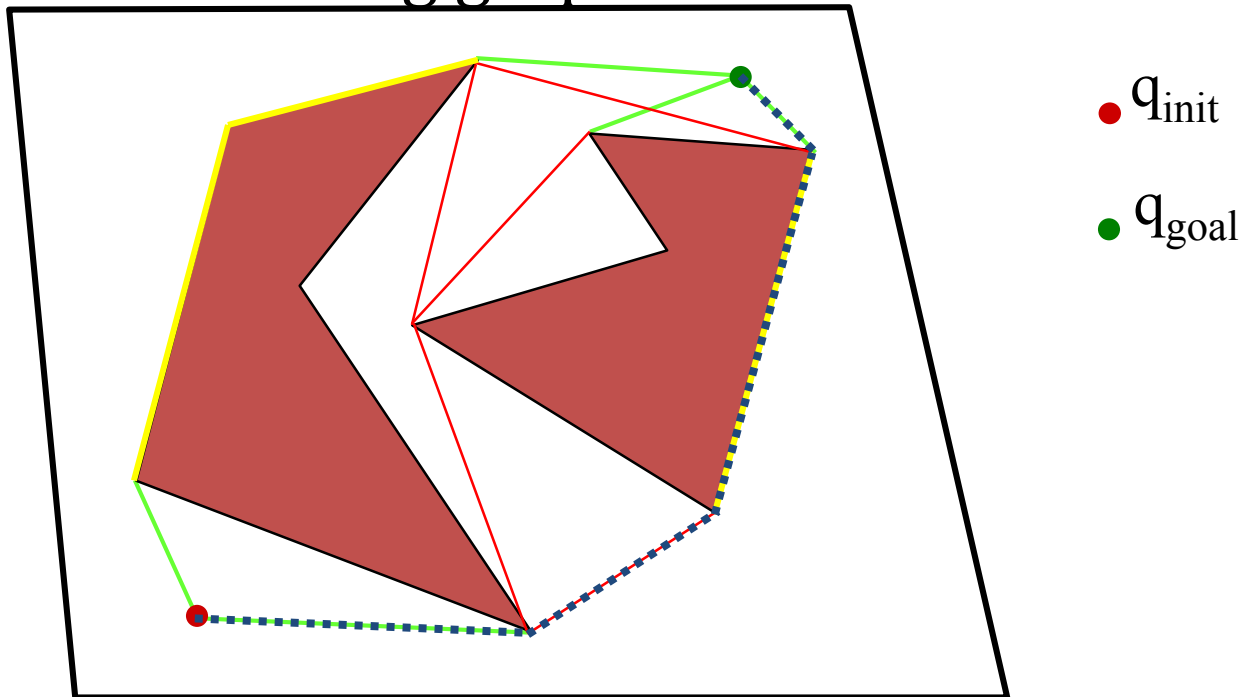
Visibility Graph

- Connect initial and goal locations with all the visible vertices
- Connect each obstacle vertex to every visible obstacle vertex
- Remove edges that intersect the interior of an obstacle



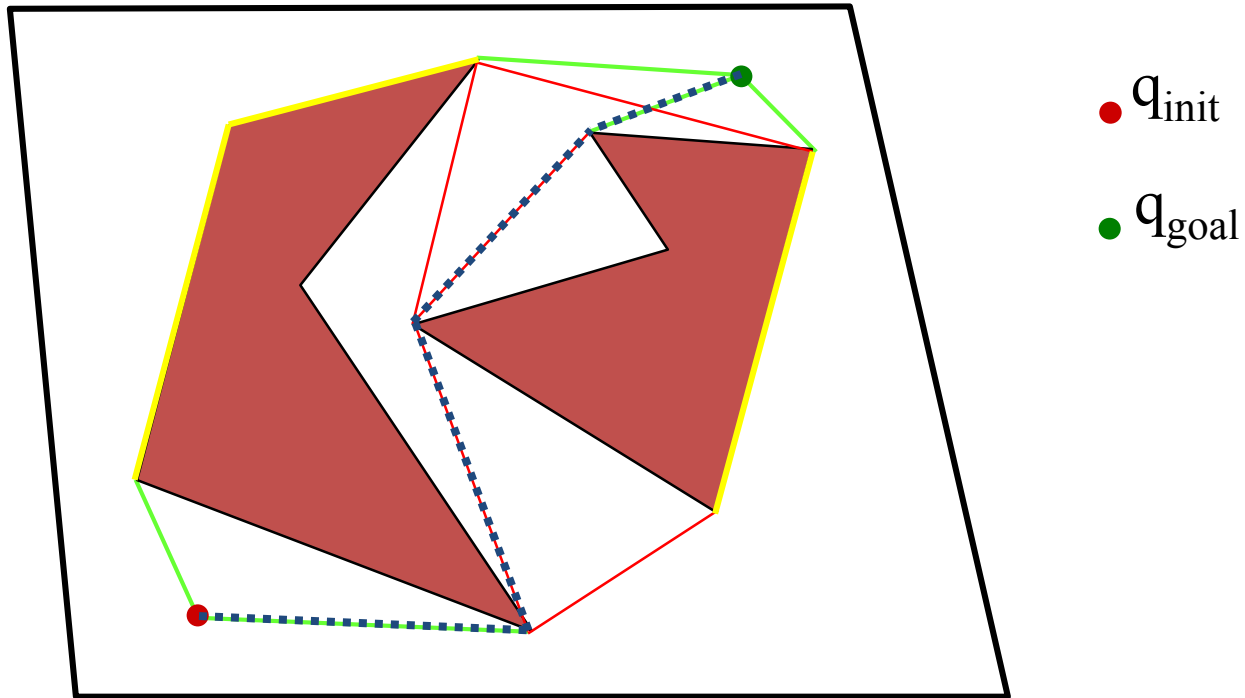
Visibility Graph

- Connect initial and goal locations with all the visible vertices
- Connect each obstacle vertex to every visible obstacle vertex
- Remove edges that intersect the interior of an obstacle
- Plan on the resulting graph



Visibility Graph

- An alternative path
- Alternative name: “Rubber band algorithm”



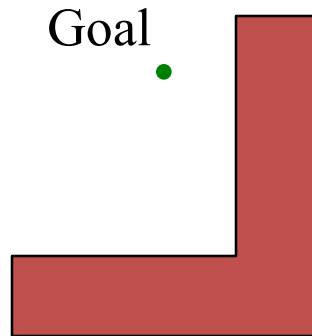
Major Fault

- Point robot
- Path planning like that guarantees to hit the obstacles



Limited-knowledge path planning

- Path planning with limited knowledge
 - Insect-inspired “bug” algorithms



- known direction to goal
- otherwise local sensing
 - walls/obstacles encoders
- “reasonable” world
 1. finitely many obstacles in any finite disc
 2. a line will intersect an obstacle finitely many times

• Start



Not truly modeling bugs...

Insects do use several cues for navigation:

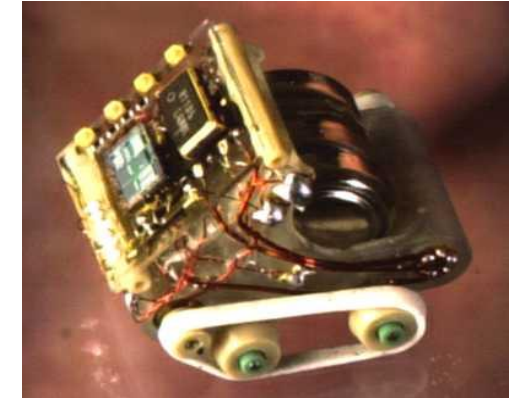


they're not ears...

visual landmarks

polarized light

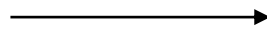
chemical sensing



neither are the current
bug-sized robots

Other animals use information from

magnetic fields

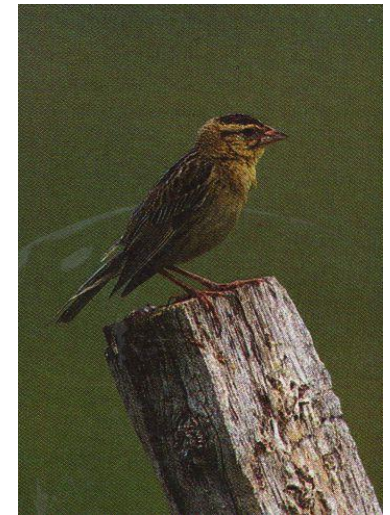


electric currents

temperature



bacteria



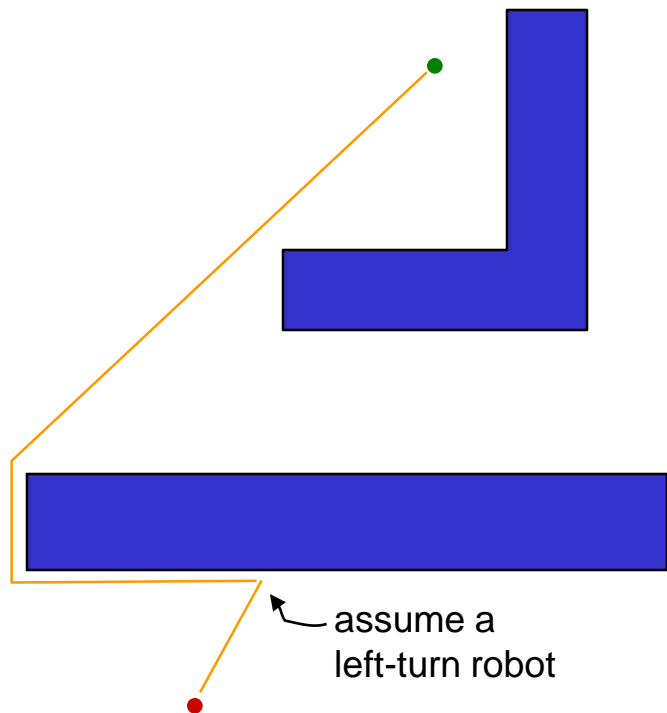
migrating bobolink¹⁷



Bug Strategy

Insect-inspired “bug” algorithms

- known direction to goal •
- otherwise only local sensing
walls/obstacles encoders



“Bug 0” algorithm

- 1) head toward goal
- 2) follow obstacles until you can head toward the goal again
- 3) continue

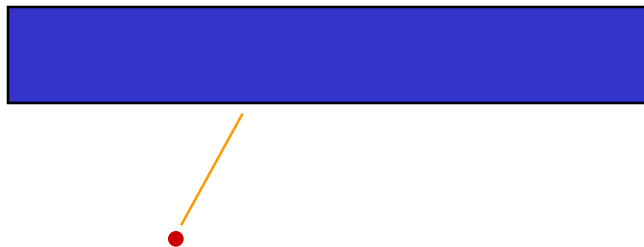
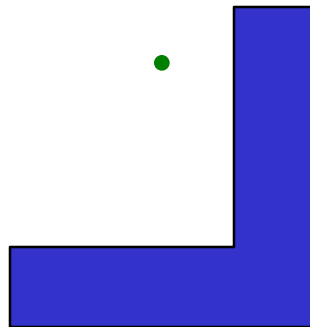
Does It Work?



Bug 1

Insect-inspired “bug” algorithms

- known direction to goal
- otherwise only local sensing
walls/obstacles encoders



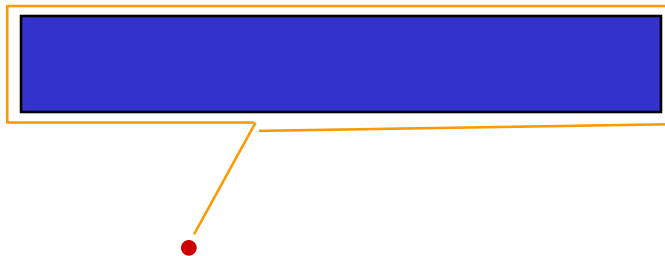
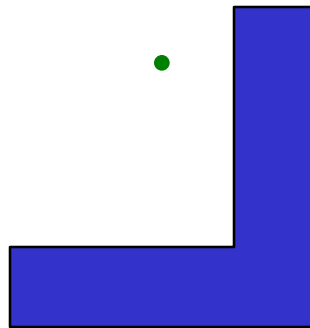
“Bug 1” algorithm

1) head toward goal

Bug 1

Insect-inspired “bug” algorithms

- known direction to goal
- otherwise only local sensing
walls/obstacles encoders



“Bug 1” algorithm

- 1) head toward goal
- 2) if an obstacle is encountered, circumnavigate it *and* remember how close you get to the goal

Bug 1 analysis

Distance Traveled

What are bounds on the path length that the robot takes?

Available Information:

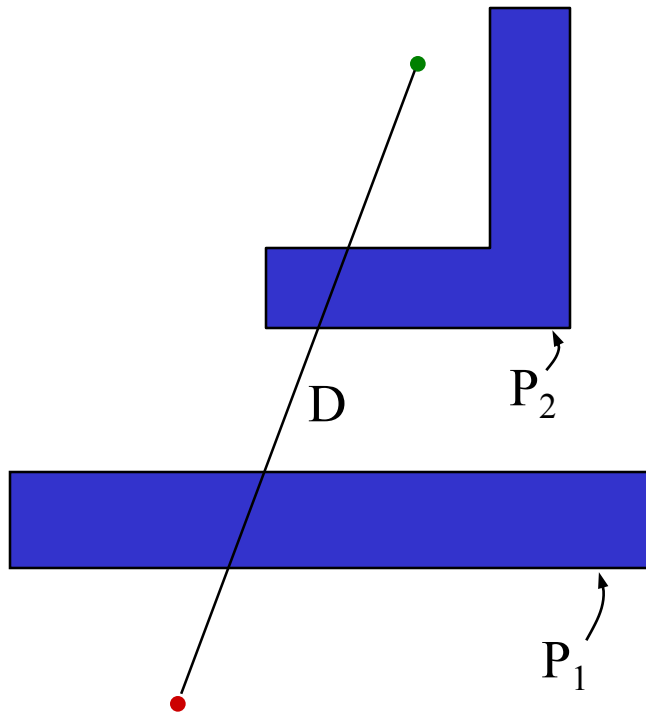
D = straight-line distance from start to goal

P_i = perimeter of the i th obstacle

Lower and upper bounds?

Lower bound:

Upper bound:



Bug 1 analysis

Distance Traveled

What are bounds on the path length that the robot takes?

Available Information:

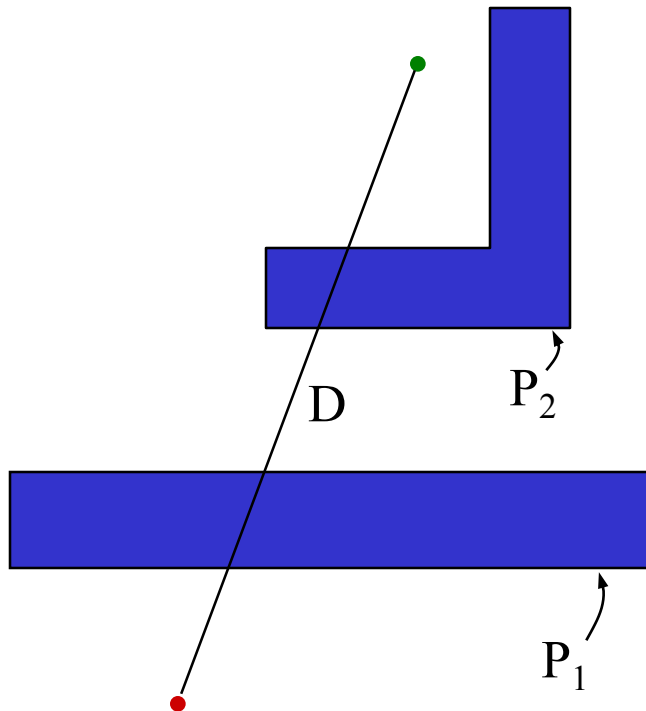
D = straight-line distance from start to goal

P_i = perimeter of the i th obstacle

Lower and upper bounds?

Lower bound: D

Upper bound:



Bug 1 analysis

Distance Traveled

What are bounds on the path length that the robot takes?

Available Information:

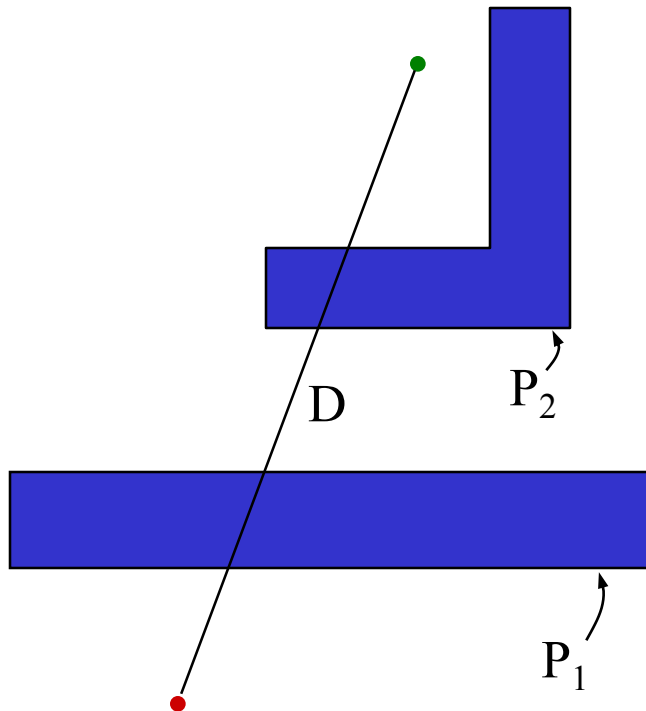
D = straight-line distance from start to goal

P_i = perimeter of the i th obstacle

Lower and upper bounds?

Lower bound: D

Upper bound: $D + 1.5 \sum P_i$



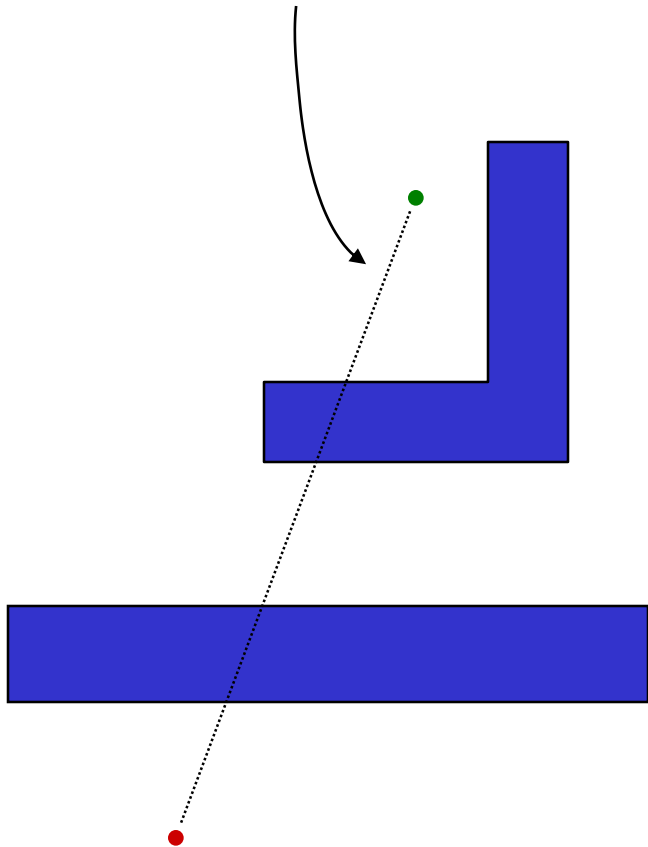
Bug Mapping



A better bug?

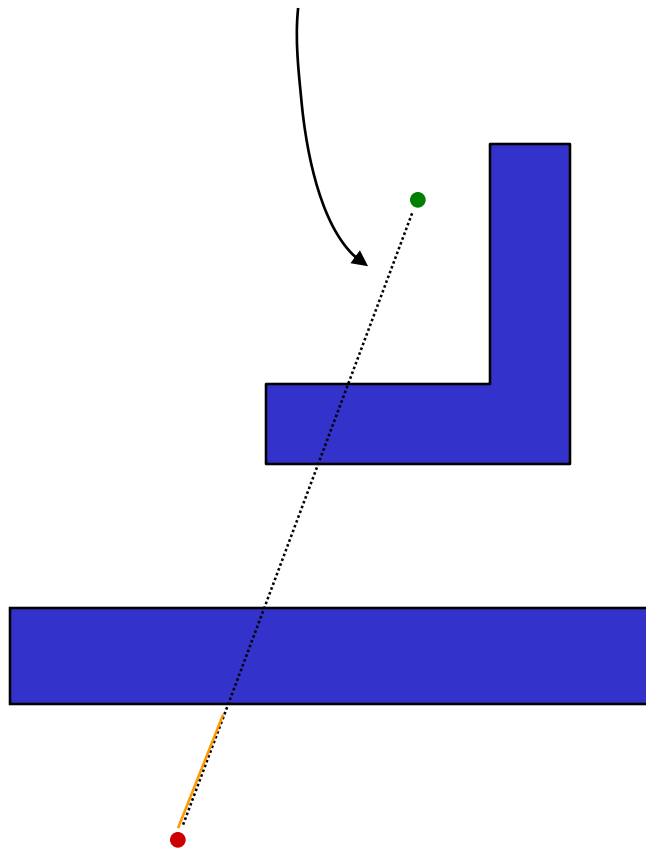
Call the line from the starting point to the goal the *s-line*

“Bug 2” algorithm



A better bug?

Call the line from the starting point to the goal the *s-line*

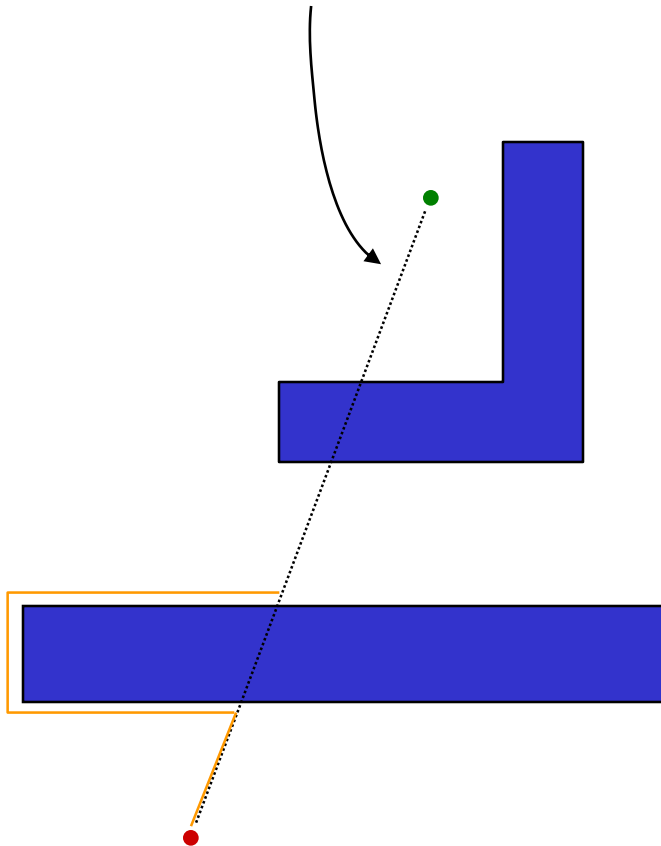


“Bug 2” algorithm

1) head toward goal on the *s-line*

A better bug?

Call the line from the starting point to the goal the *s-line*



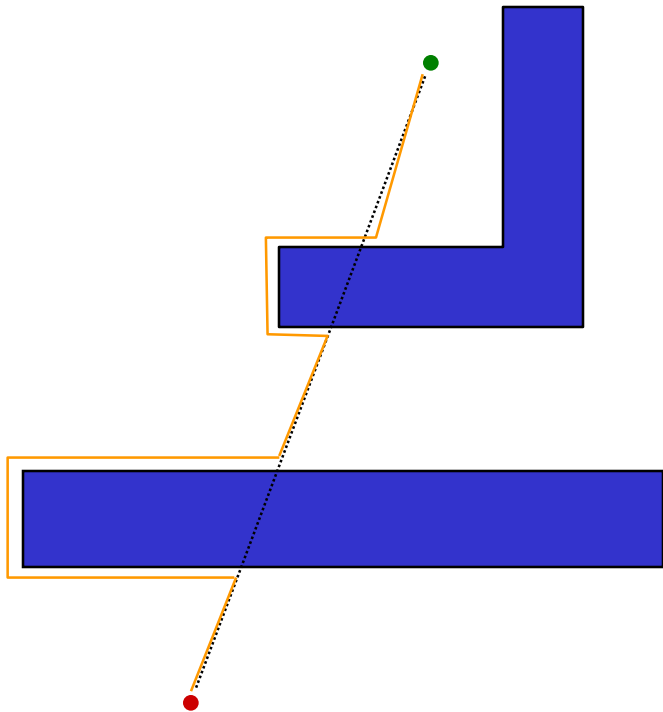
“Bug 2” algorithm

- 1) head toward goal on the *s-line*
- 2) if an obstacle is in the way, follow it until encountering the *s-line* again.

A better bug?

“Bug 2” algorithm

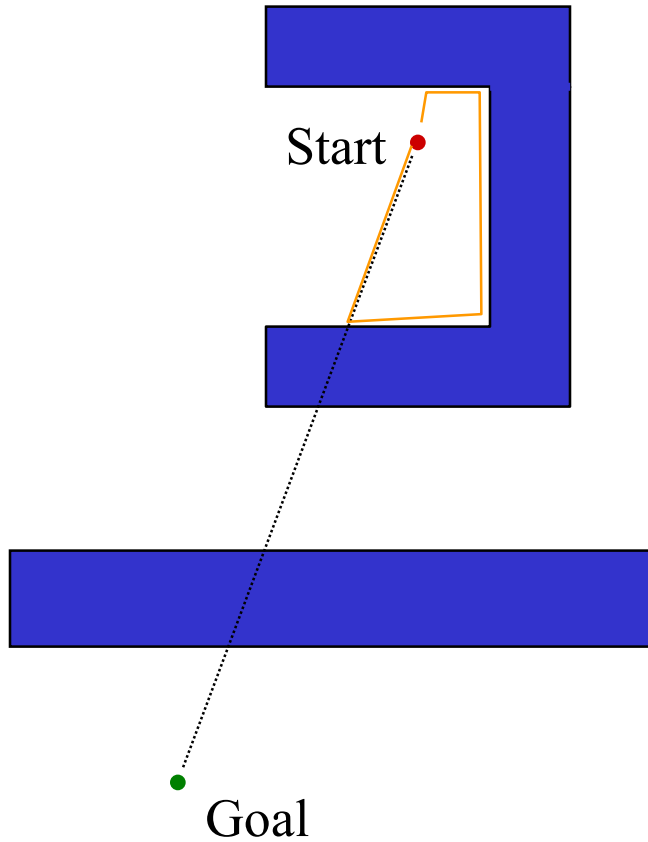
s-line



- 1) head toward goal on the *s-line*
- 2) if an obstacle is in the way, follow it until encountering the *s-line* again.
- 3) Leave the obstacle and continue toward the goal

A better bug?

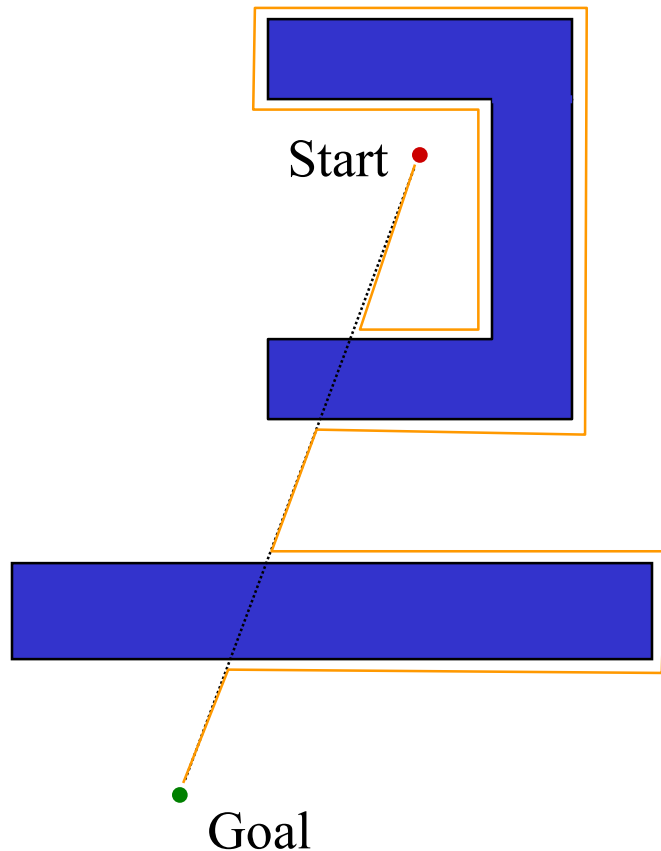
“Bug 2” algorithm



- 1) head toward goal on the *s-line*
- 2) if an obstacle is in the way, follow it until encountering the *s-line* again *closer to the goal*.
- 3) Leave the obstacle and continue toward the goal

Bug 2 analysis

Distance Traveled



What are bounds on the path length that the robot takes?

Available Information:

D = straight-line distance from start to goal

P_i = perimeter of the i th obstacle

N_i = number of s-line intersections with the i th obstacle

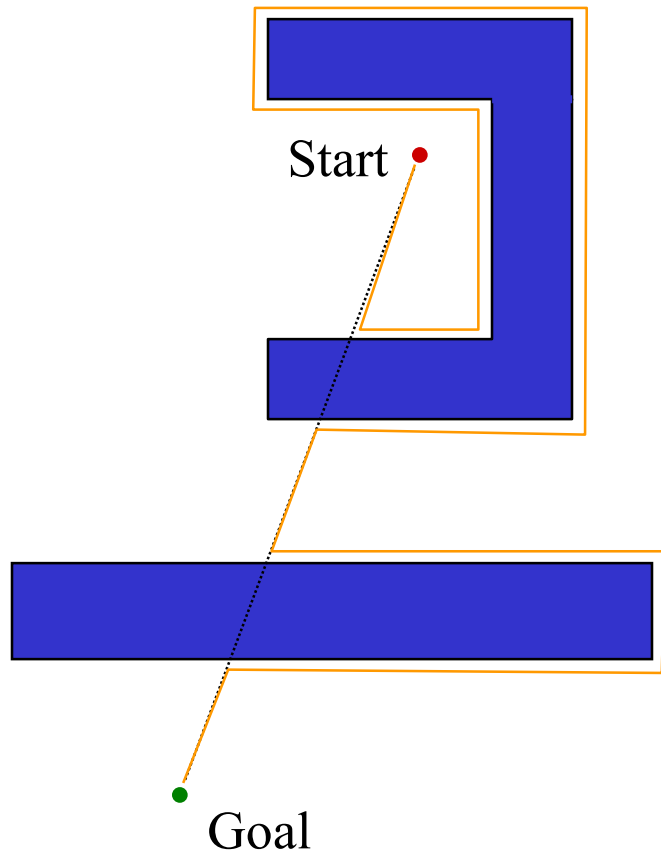
Lower and upper bounds?

Lower bound:

Upper bound:

Bug 2 analysis

Distance Traveled



What are bounds on the path length that the robot takes?

Available Information:

D = straight-line distance from start to goal

P_i = perimeter of the i th obstacle

N_i = number of s-line intersections with the i th obstacle

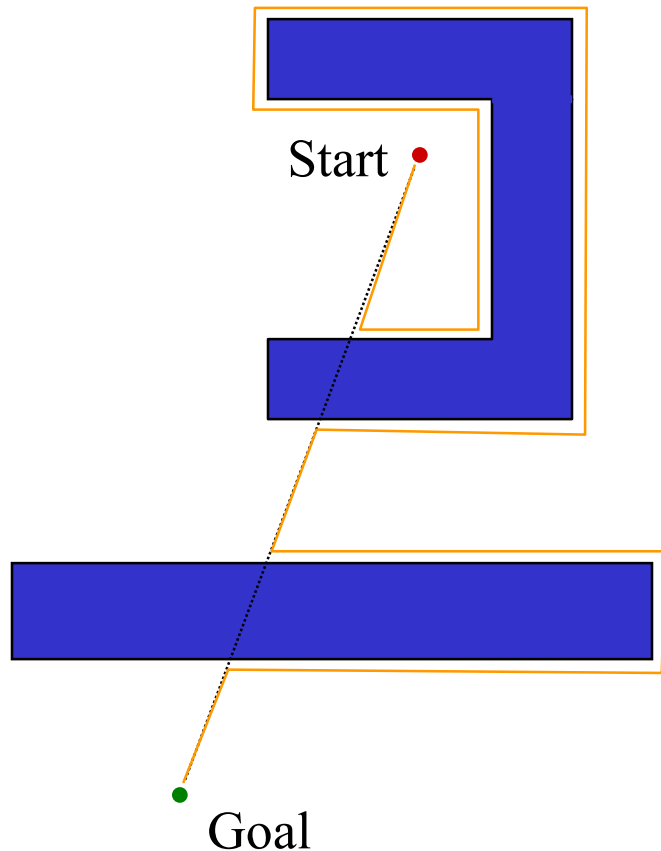
Lower and upper bounds?

Lower bound: D

Upper bound:

Bug 2 analysis

Distance Traveled



What are bounds on the path length that the robot takes?

Available Information:

D = straight-line distance from start to goal

P_i = perimeter of the i th obstacle

N_i = number of s-line intersections with the i th obstacle

Lower and upper bounds?

Lower bound: D

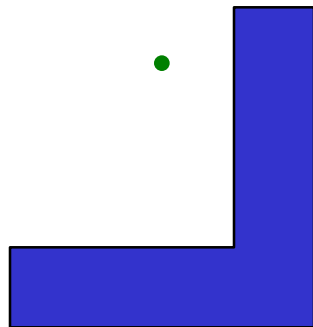
Upper bound: $D + 0.5 \sum_i N_i P_i$

head-to-head comparison

or thorax-to-thorax, perhaps

What are worlds in which Bug 2 does better than Bug 1 (and vice versa) ?

Bug 2 beats Bug 1



Bug 1 beats Bug 2

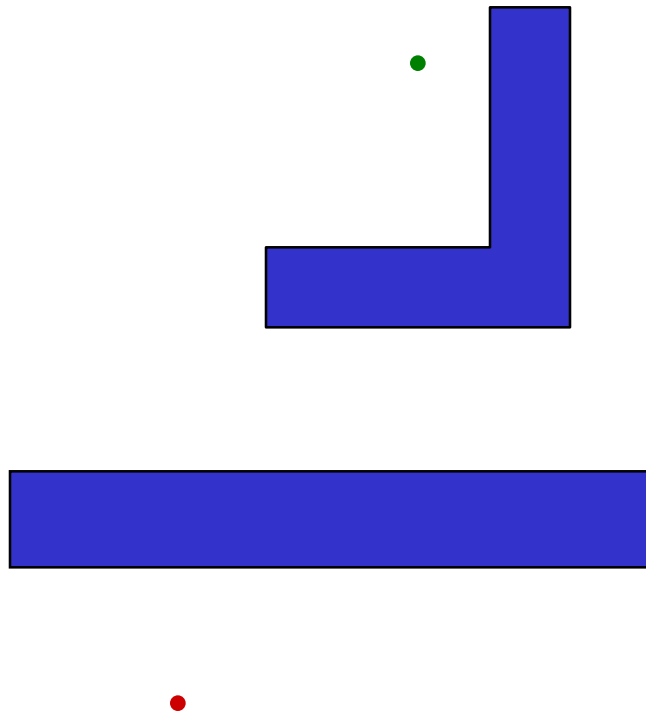


head-to-head comparison

or thorax-to-thorax, perhaps

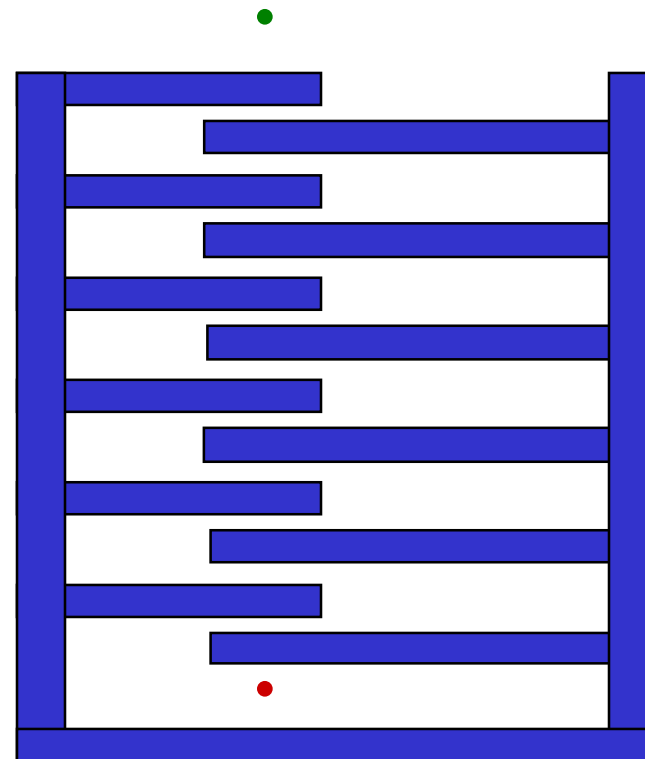
What are worlds in which Bug 2 does better than Bug 1 (and vice versa) ?

Bug 2 beats Bug 1



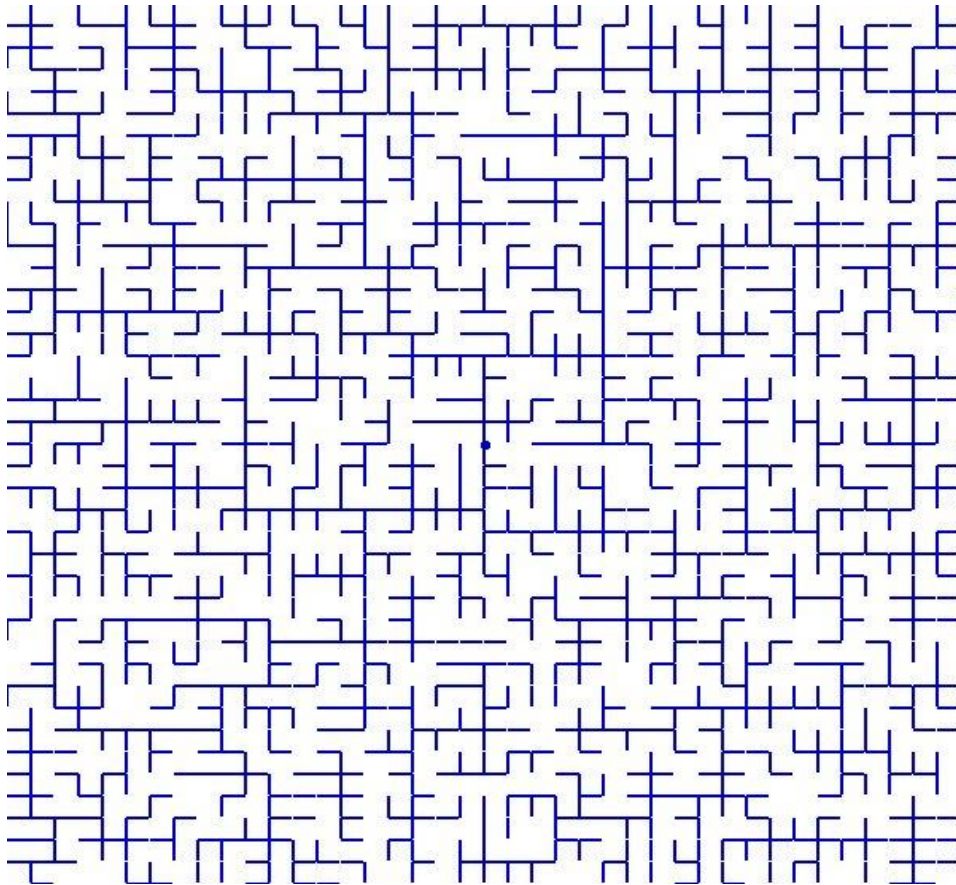
Bug 1 beats Bug 2

“zipper world”



Other bug-like algorithms

The Pledge maze-solving algorithm

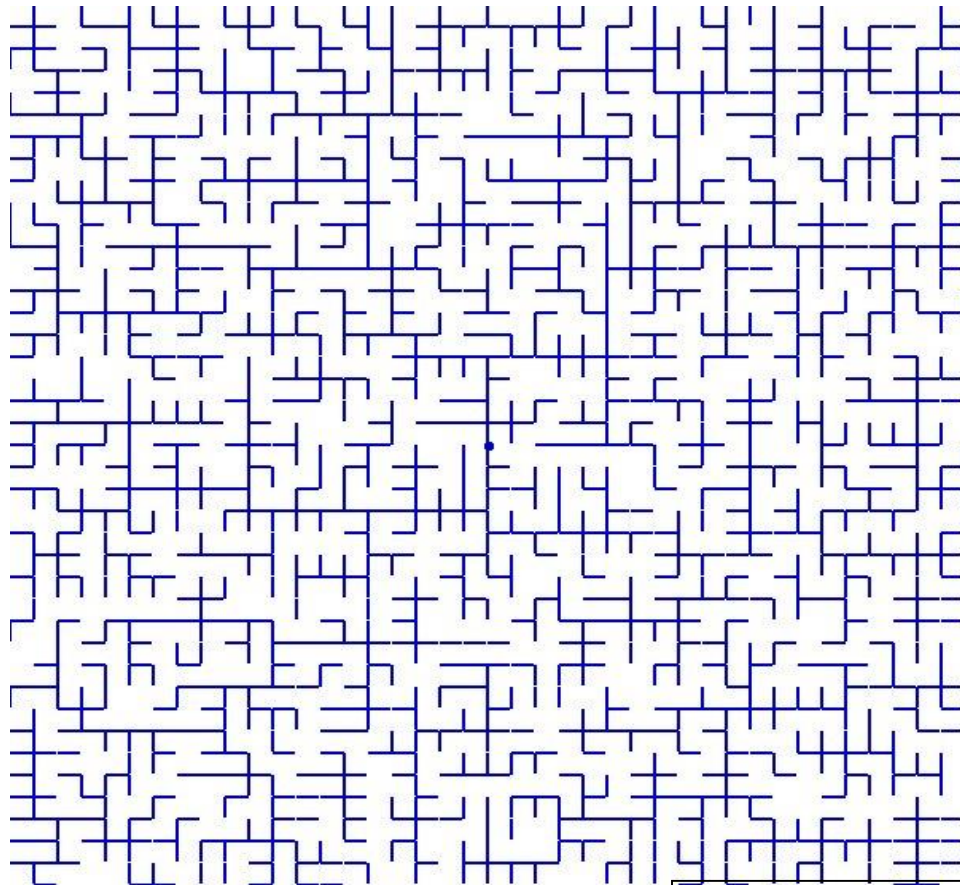


1. Go to a wall
2. Keep the wall on your right
3. Continue until out of the maze



Other bug-like algorithms

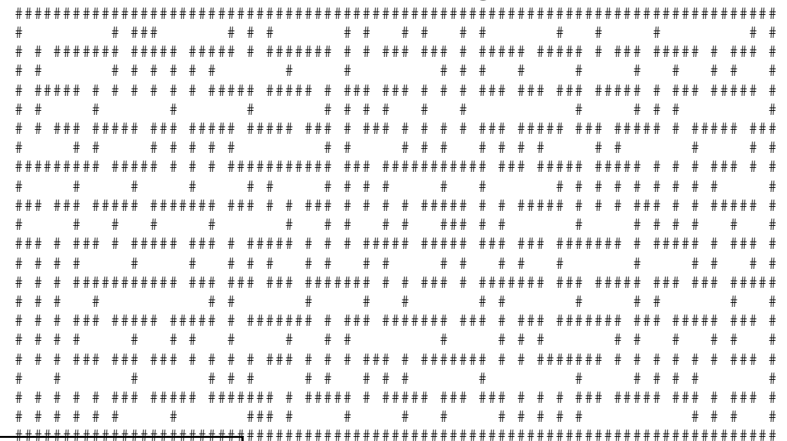
The Pledge maze-solving algorithm



- 1) Go to a wall
- 2) Keep the wall on your right
- 3) Continue until out of the maze

```
int a[1817];main(z,p,q,r){for(p=80;q+p-80;p=2*a[p])
for(z=9;z--;)q=3&(r=time(0)+r*57)/7,q=q?q-1?q-2?1-p%79?-
1:0:p%79-77?1:0:p<1659?79:0:p>158?-
79:0,q?!a[p+q*2]?a[p+=a[p+=q]=q]=q:0:0;for(;q++-
1817);printf(q%79?"%c":"%c\n", "#"[!a[q-1]]);}
```

IOCCC random maze generator



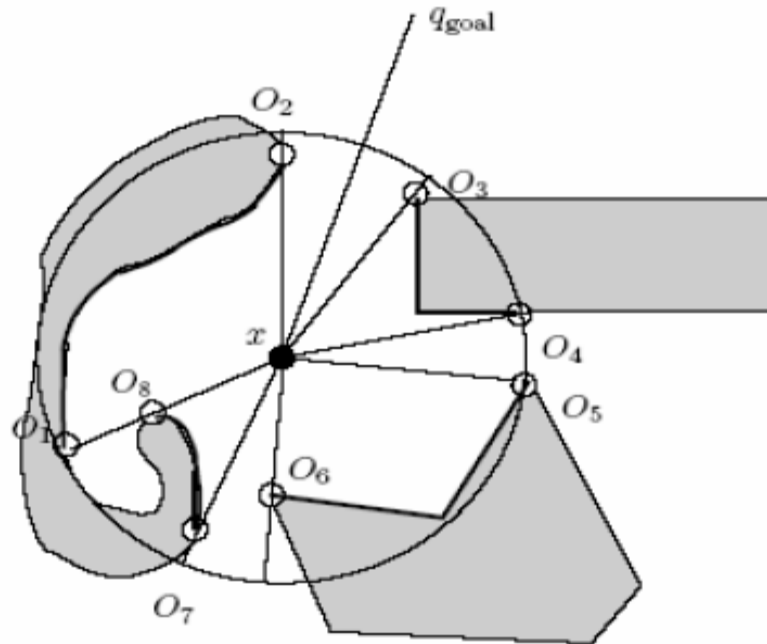
discretized RRT

mazes of unusual origin

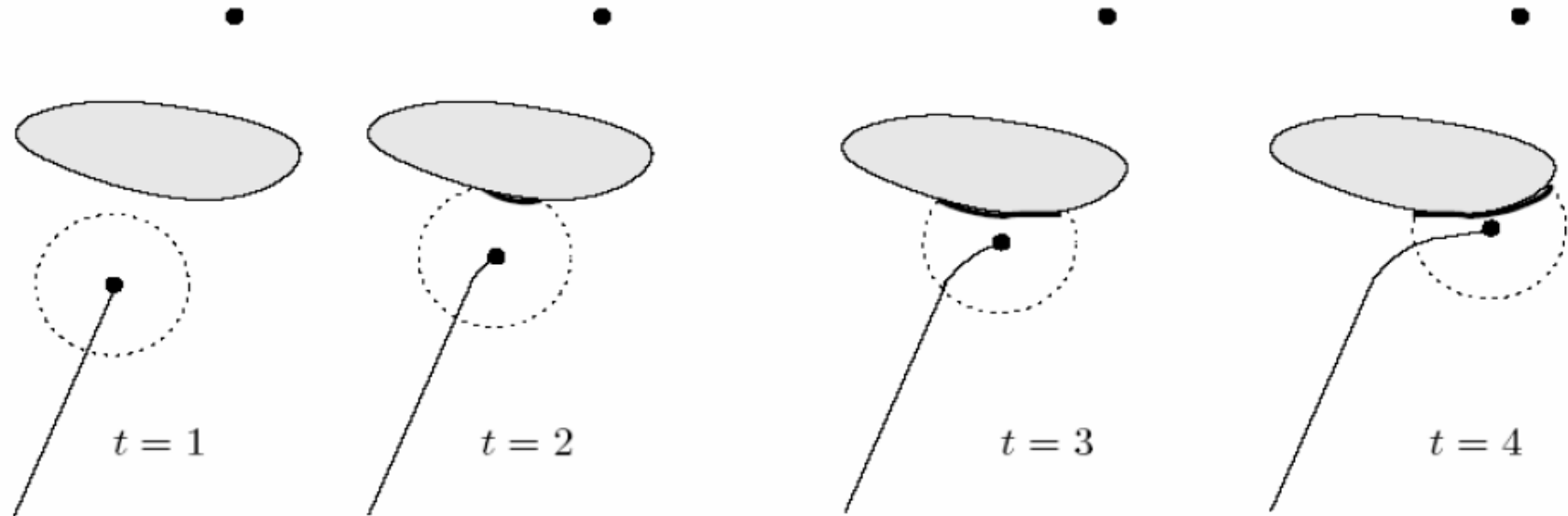


Tangent Bug

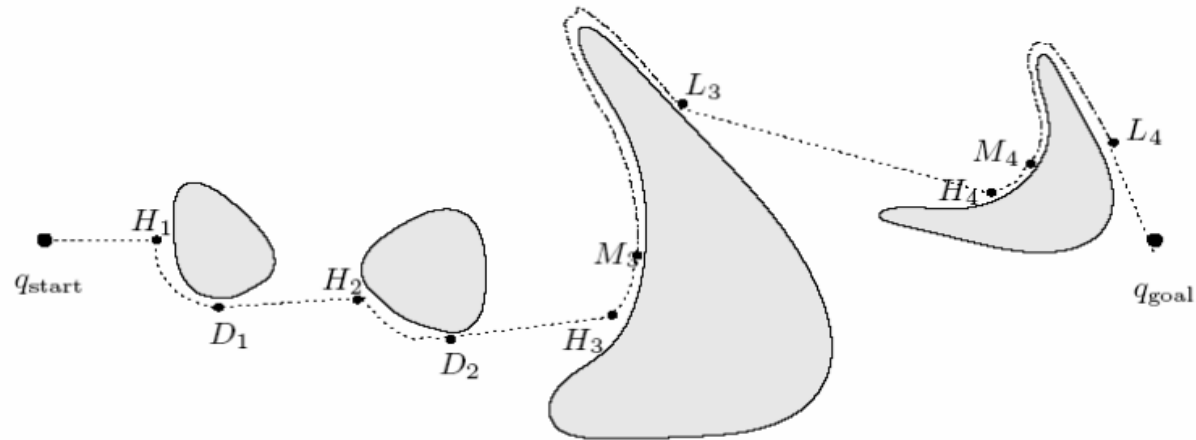
- Limited Range Sensor
- Tangent Bug relies on finding endpoints of finite, continues segments of the obstacles



Tangent Bug

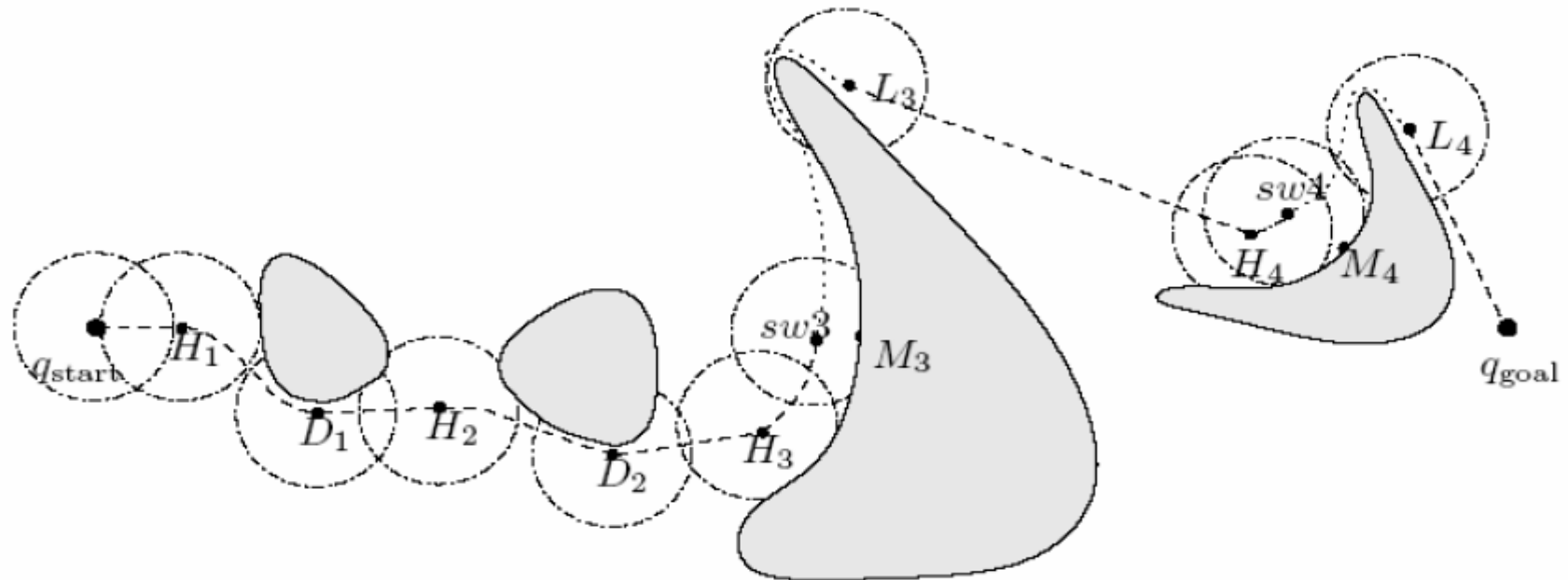


Contact Sensor Tangent Bug



1. Robot moves toward goal until it hits obstacle 1 at H_1
2. Pretend there is an infinitely small sensor range and the direction which minimizes the heuristic is to the right
3. Keep following obstacle until robot can go toward obstacle again
4. Same situation with second obstacle
5. At third obstacle, the robot turned left until it could not increase heuristic
6. $D_{followed}$ is distance between M_3 and goal, d_{reach} is distance between robot and goal because sensing distance is zero

Limited Sensor Range Tangent-Bug



Infinite Sensor Range Tangent Bug

