Robotic Motion Planning: Bug Algorithms

Robotics Institute 16-735

http://voronoi.sbp.ri.cmu.edu/~motion

Howie Choset

http://voronoi.sbp.ri.cmu.edu/~choset

What's Special About Bugs

- Many planning algorithms assume global knowledge
- Bug algorithms assume only *local* knowledge of the environment and a global goal
- Bug behaviors are simple:
  - 1) Follow a wall (right or left)
  - 2) Move in a straight line toward goal
- Bug 1 and Bug 2 assume essentially tactile sensing (bump wall)
- Tangent Bug deals with finite distance sensing

Bug algorithms *

- Simple and intuitive
- Straightforward to implement
- Success guaranteed (when possible)
- Assumes perfect positioning and sensing
- Sensor based planning – has to be incremental and reactive

*Reference: Principles of Robot Motion. MIT Press. Howie Choset, Kevin Lynch, Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia Kavraki and Sebastian Thrun. Thanks to Howie Choset, CMU, for these slides

Bug algorithms

- Assumptions:
  - Point robot
  - Contact sensor (**Bug1,Bug2**) or finite range sensor (**Tangent Bug**)
  - Bounded environment
  - Robot position is perfectly known
  - Robot can measure the distance between two points

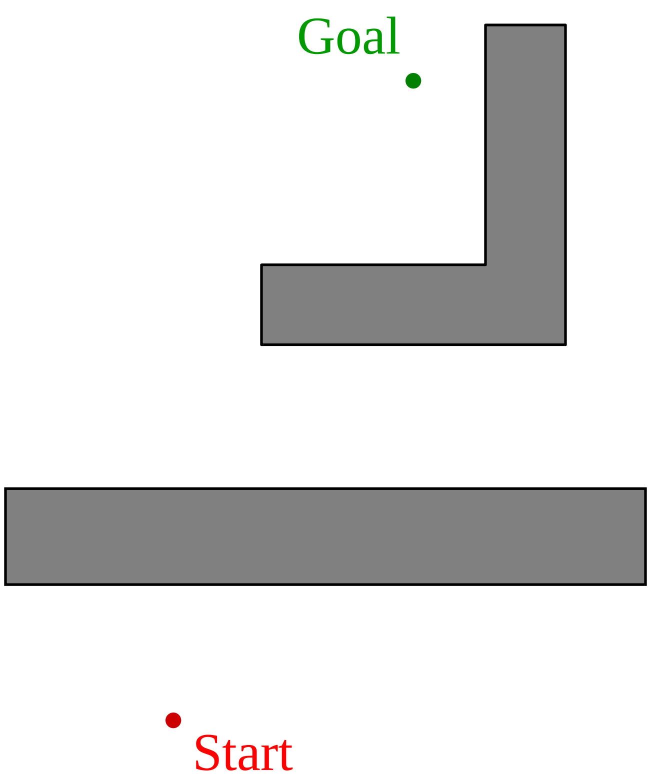
A Few General Concepts

- Workspace W
  - $\Re(2)$ or $\Re(3)$ depending on the robot
  - could be infinite (open) or bounded (closed/compact)
- Obstacle WO_i
- Free workspace $W_{free} = W \setminus \cup_i WO_i$

The *Bug* Algorithms

provable results...

Insect-inspired



- **known direction to goal**
  - **robot can measure distance d(x,y) between pts x and y**
- **otherwise local sensing**

  walls/obstacles & encoders
- ***reasonable* world**

  1) finitely many obstacles in any finite area

  2) a line will intersect an obstacle finitely many times

  3) Workspace is bounded

  $W \subset B_r(x),\ r < \infty$

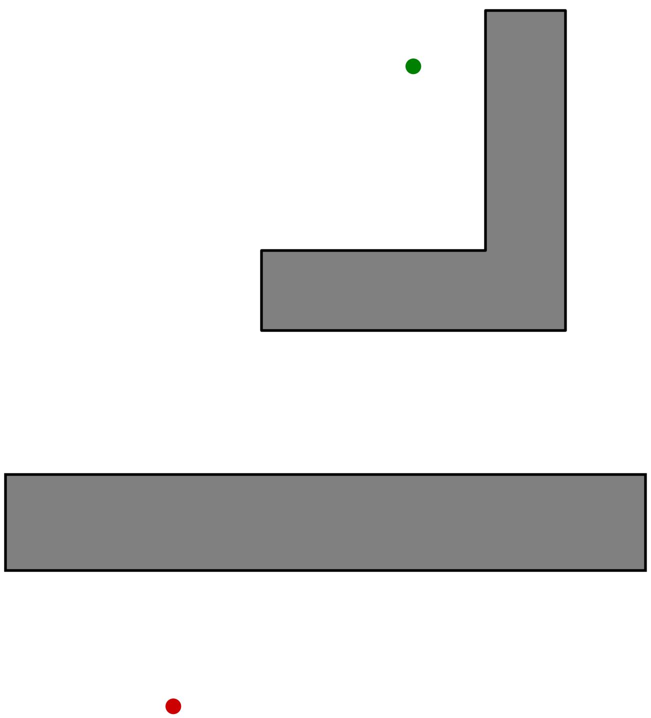
  $B_r(x) = \{ y \in \Re(2) \mid d(x,y) < r \}$

Buginner Strategy

"Bug 0" algorithm

- **known direction to goal**
- **otherwise local sensing**

  walls/obstacles & encoders



Some notation:

q_{start} and q_{goal}

"hit point" q^H_i
"leave point q^L_i

A *path* is a sequence of hit/leave pairs bounded by q_{start} and q_{goal}

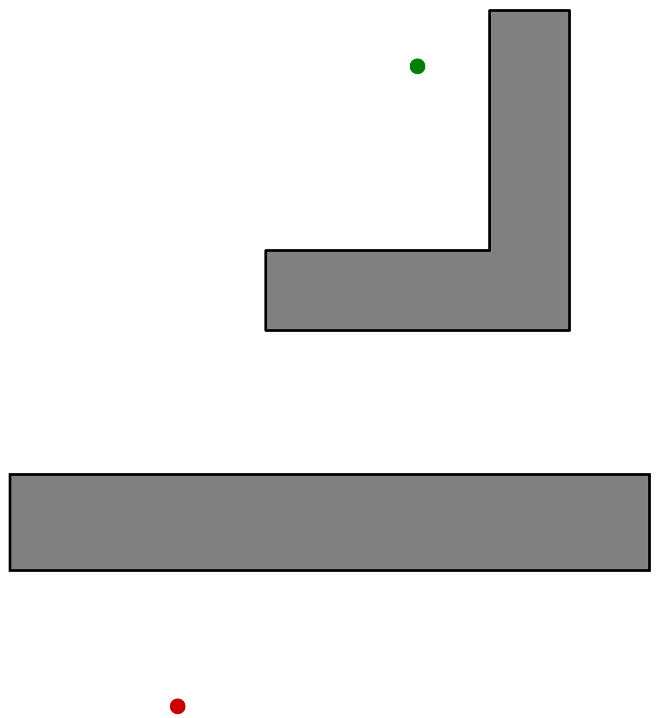
how ?

Buginner Strategy

"Bug 0" algorithm

- **known direction to goal**
- **otherwise local sensing**

  walls/obstacles & encoders



1) head toward goal

2) follow obstacles until you can head toward the goal again

3) continue

path ?

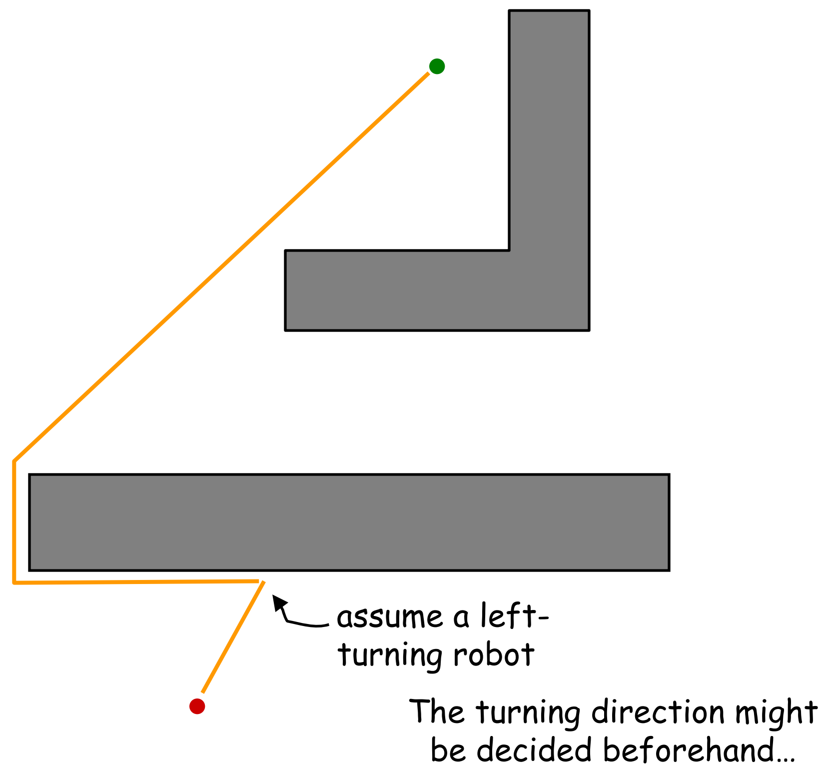
Buginner Strategy

"Bug 0" algorithm

1) head toward goal

2) follow obstacles until you can head toward the goal again

3) continue



OK ?

Bug Zapper

What map will foil Bug 0 ?

"Bug 0" algorithm

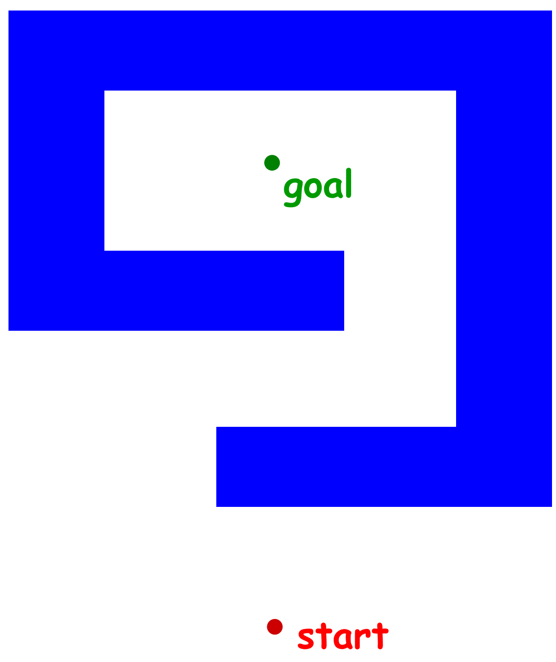
1) head toward goal

2) follow obstacles until you can head toward the goal again

3) continue

Bug Zapper

What map will foil Bug 0 ?



"Bug 0" algorithm

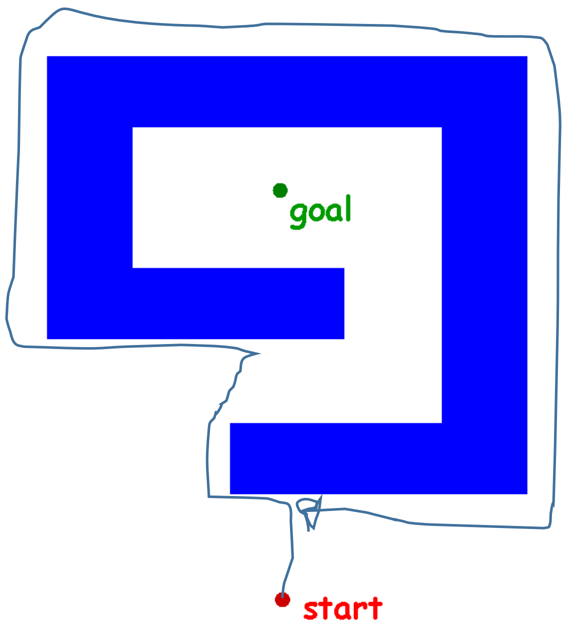
1) head toward goal

2) follow obstacles until you can head toward the goal again

3) continue

Bug Zapper

What map will foil Bug 0 ?



"Bug 0" algorithm

1) head toward goal

2) follow obstacles until you can head toward the goal again

3) continue

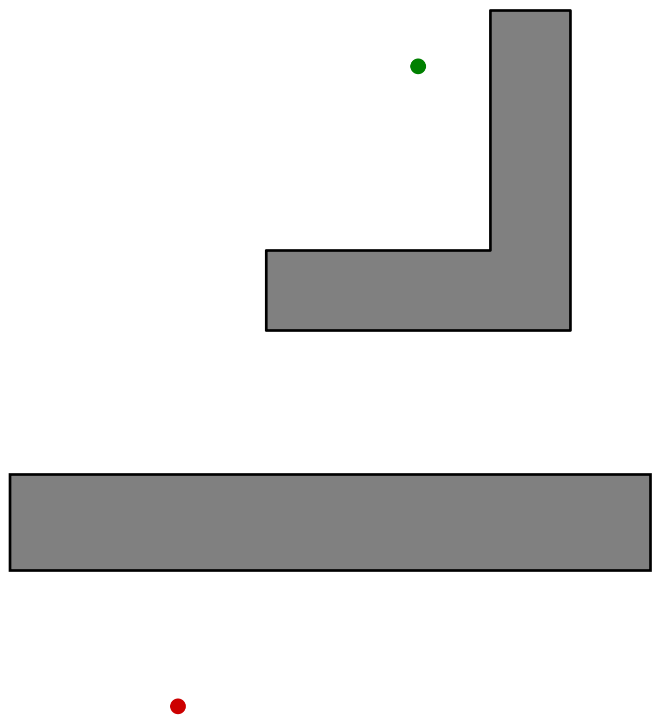
Bug 0 bugs out – endless loop!

A better bug?

But add <u>some</u> memory!

- known direction to goal
- otherwise local sensing

walls/obstacles & encoders

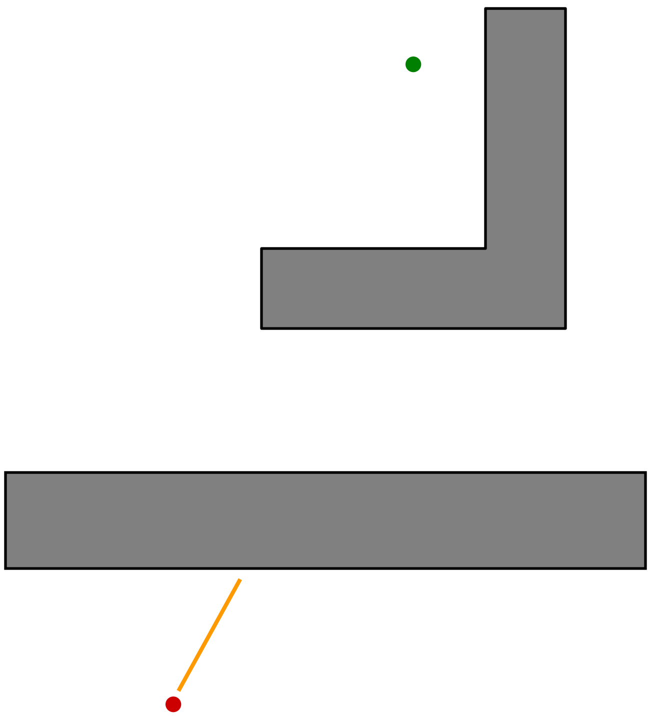


improvement ideas?

Bug 1

But some computing power!
- known direction to goal
- otherwise 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

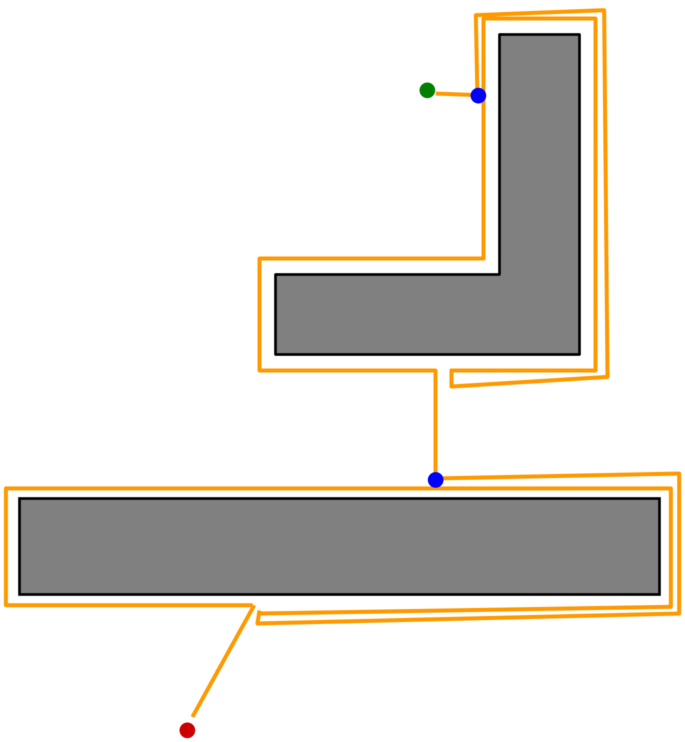
3) return to that closest point (by wall-following) and continue

Vladimir Lumelsky & Alexander Stepanov: Algorithmica 1987

Bug 1

But some computing power! { • known direction to goal • otherwise local sensing }

walls/obstacles & encoders



"Bug 1" algorithm

1) head toward goal

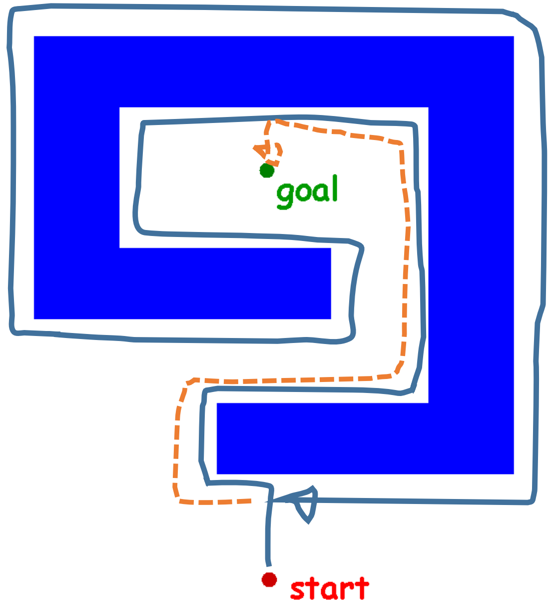
2) if an obstacle is encountered, circumnavigate it *and* remember how close you get to the goal

3) return to that closest point (by wall-following) and continue

Vladimir Lumelsky & Alexander Stepanov: Algorithmica 1987

Bug 1 works where Bug 0 fails!

What map will foil Bug 0 ?



"Bug 0" algorithm

1) head toward goal

2) follow obstacles until you can head toward the goal again

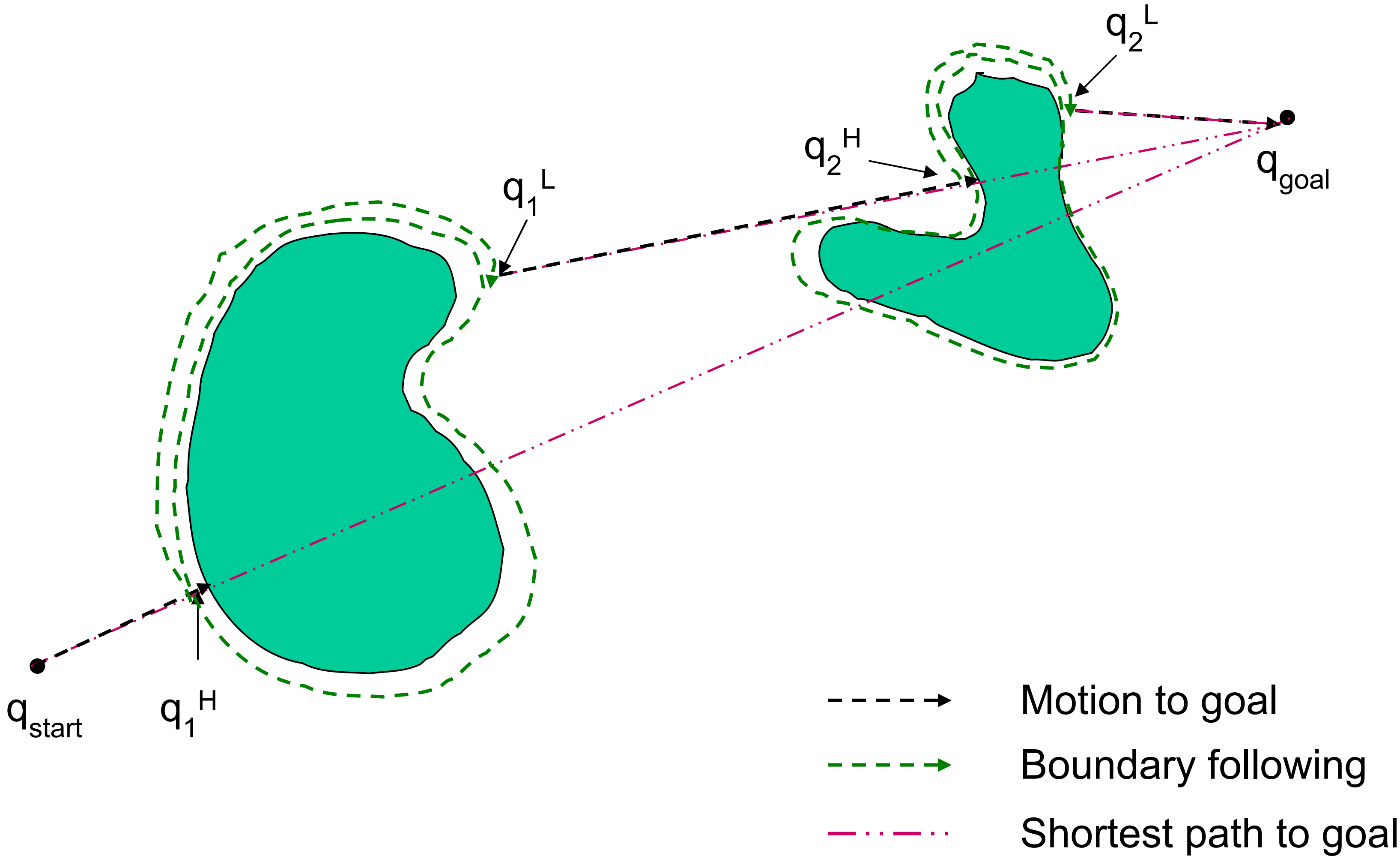
3) continue

1. Bug 1 circumnavigates obstacle
2. Remembers closest point
3. Returns to that point and heads to goal

BUG 1 More formally

- Let $q^L_0 = q_{start}$; i = 1
- repeat
  - repeat
    - from q^L_{i-1} move toward q_{goal}
  - until goal is reached or obstacle encountered at q^H_i
  - if goal is reached, exit
  - repeat
    - follow boundary recording pt q^L_i with shortest distance to goal
  - until q_{goal} is reached or q^H_i is re-encountered
  - if goal is reached, exit
  - Go to q^L_i
  - if move toward q_{goal} moves into obstacle
    - exit with failure
  - else
    - i=i+1
    - continue

Bug1 - example



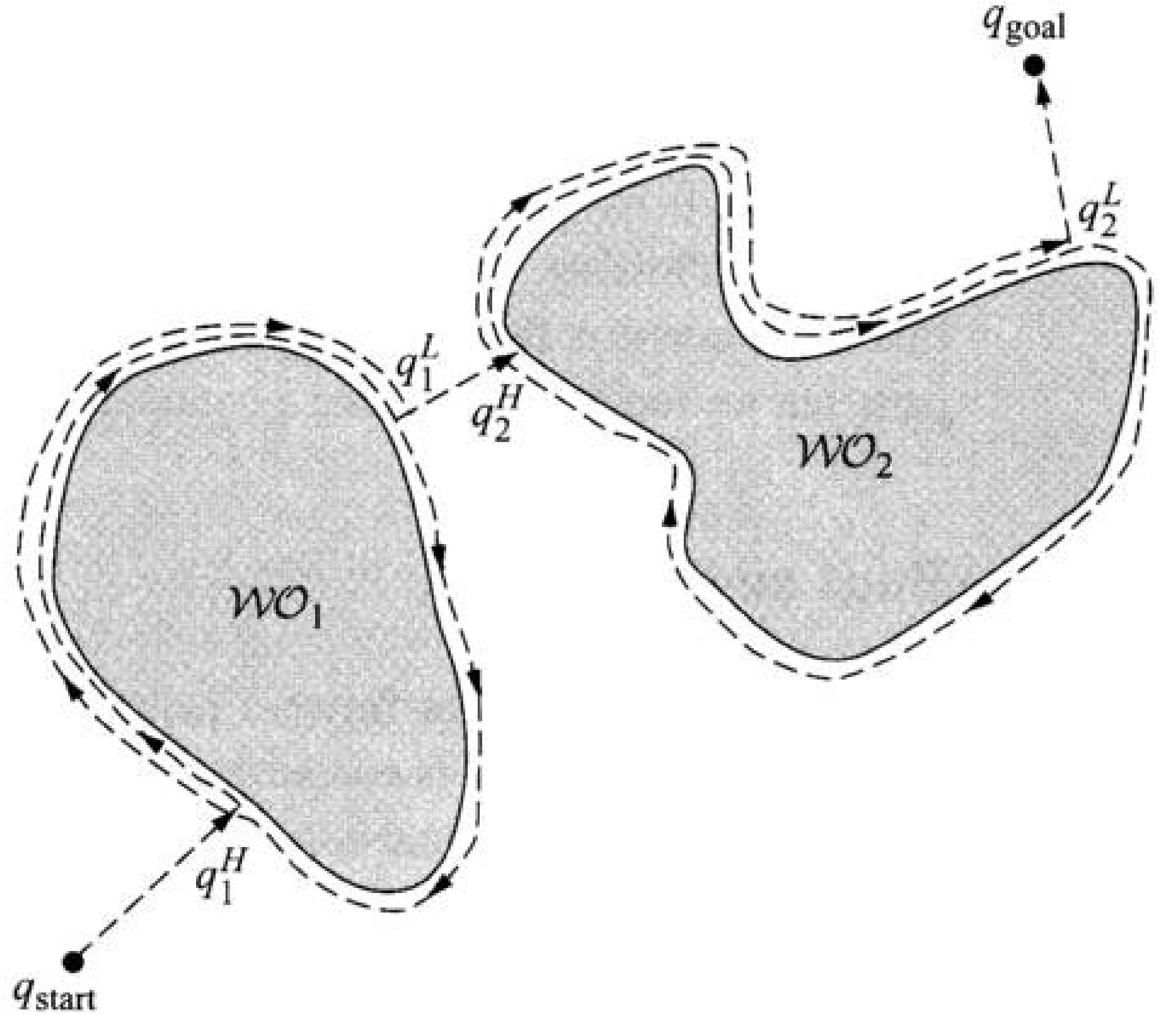


Figure 2.1 The Bug1 algorithm successfully finds the goal.

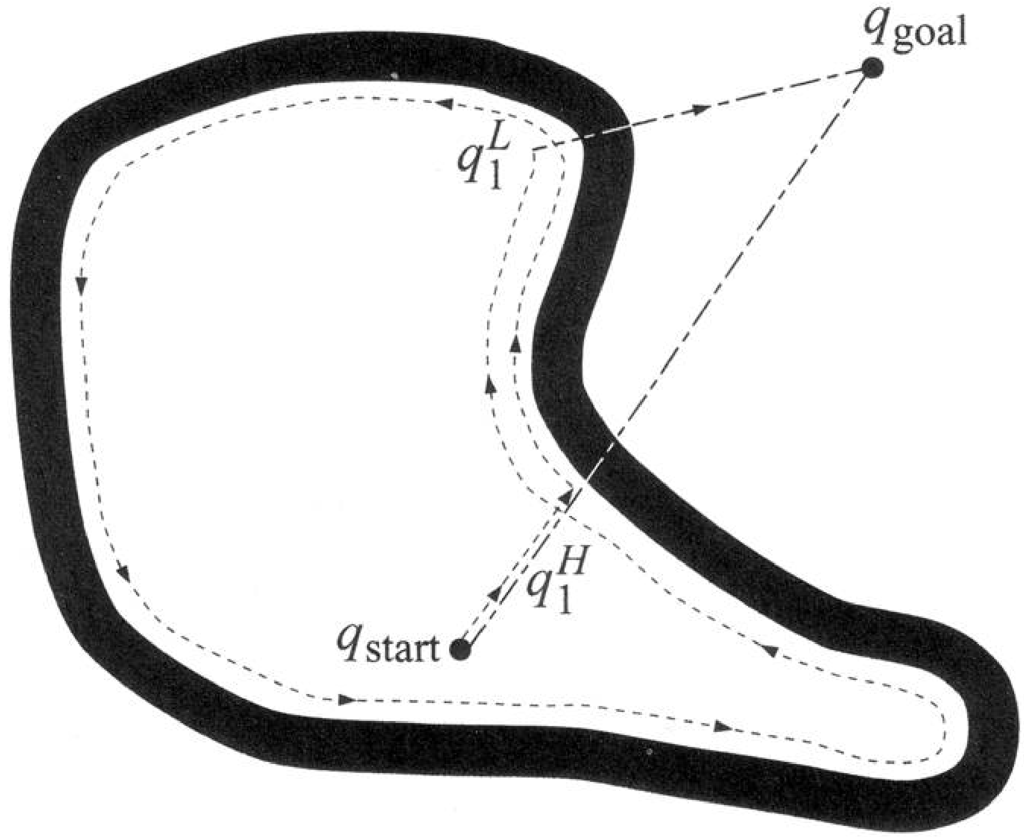
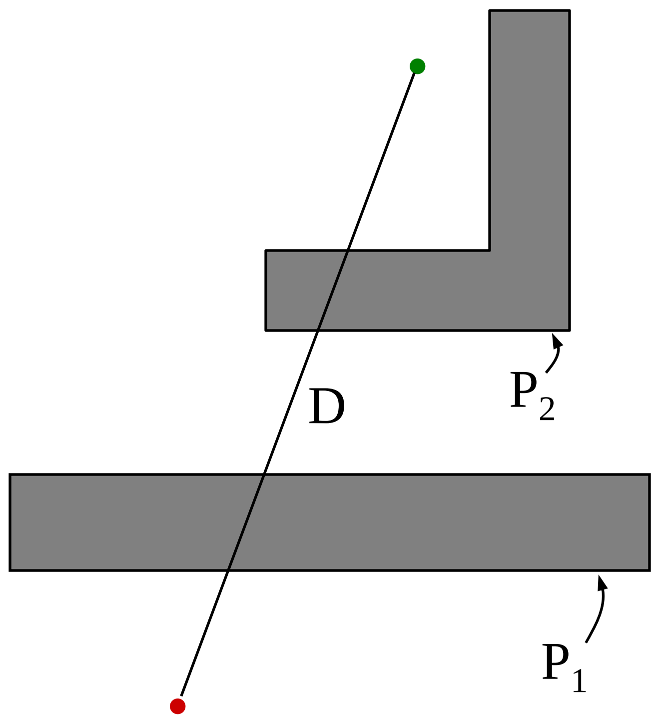


Figure 2.2 The Bug1 algorithm reports the goal is unreachable.

"Quiz" Bug 1 analysis

Bug 1: Path Bounds

What are upper/lower bounds on the path length that the robot takes?



D = straight-line distance from start to goal

P_i = perimeter of the *i* th obstacle

Lower bound:
What's the shortest distance it might travel?

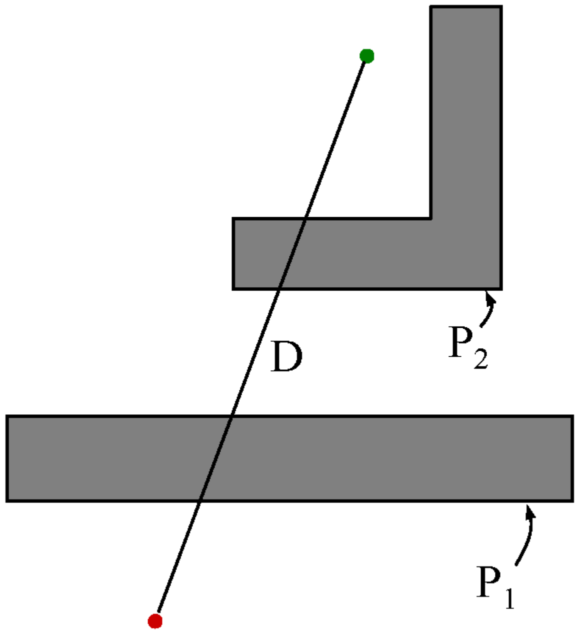
Upper bound:
What's the longest distance it might travel?

What is an environment where your upper bound is <u>required</u>?

"Quiz" Bug 1 analysis

Bug 1: Path Bounds

What are upper/lower bounds on the path length that the robot takes?



D = straight-line distance from start to goal

P_i = perimeter of the *i* th obstacle

Lower bound:
What's the shortest distance it might travel?

D

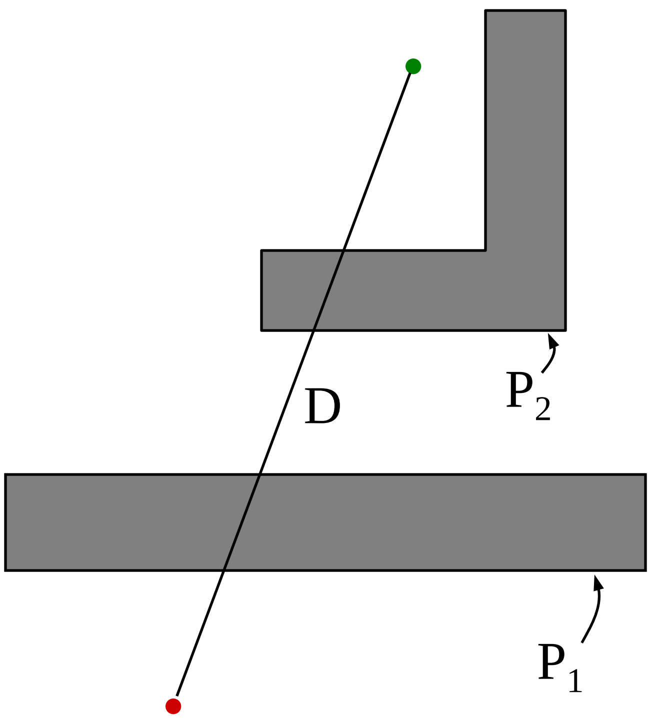
Upper bound:
What's the longest distance it might travel?

What is an environment where your upper bound is required?

"Quiz" Bug 1 analysis

Bug 1: Path Bounds

What are upper/lower bounds on the path length that the robot takes?



D = straight-line distance from start to goal

P_i = perimeter of the *i* th obstacle

Lower bound:
What's the shortest distance it might travel?

D

Upper bound:
What's the longest distance it might travel?

$$\mathbf{D + 1.5 \sum_i P_i}$$

What is an environment where your upper bound is required?

How Can We Show Completeness?

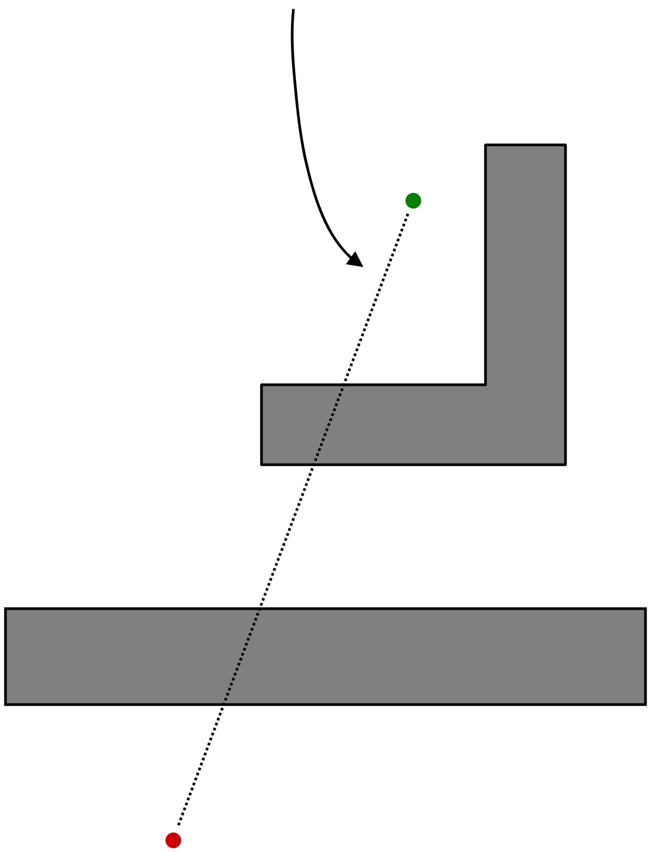
- An algorithm is *complete* if, in finite time, it finds a path if such a path exists or terminates with failure if it does not.

- Suppose BUG1 were incomplete
  - Therefore, there is a path from start to goal
    - By assumption, it is finite length, and intersects obstacles a finite number of times.
  - BUG1 does not find it
    - Either it terminates incorrectly, or, it spends an infinite amount of time
    - Suppose it never terminates
      - but each leave point is closer to the obstacle than corresponding hit point
      - Each hit point is closer than the last leave point
      - Thus, there are a finite number of hit/leave pairs; after exhausting them, the robot will proceed to the goal and terminate
    - Suppose it terminates (incorrectly)
    - Then, the closest point after a hit must be a leave where it would have to move into the obstacle
      - But, then line from robot to goal must intersect object even number of times (Jordan curve theorem)
      - But then there is another intersection point on the boundary closer to object. Since we assumed there is a path, we must have crossed this pt on boundary which contradicts the definition of a leave point.

Another step forward?

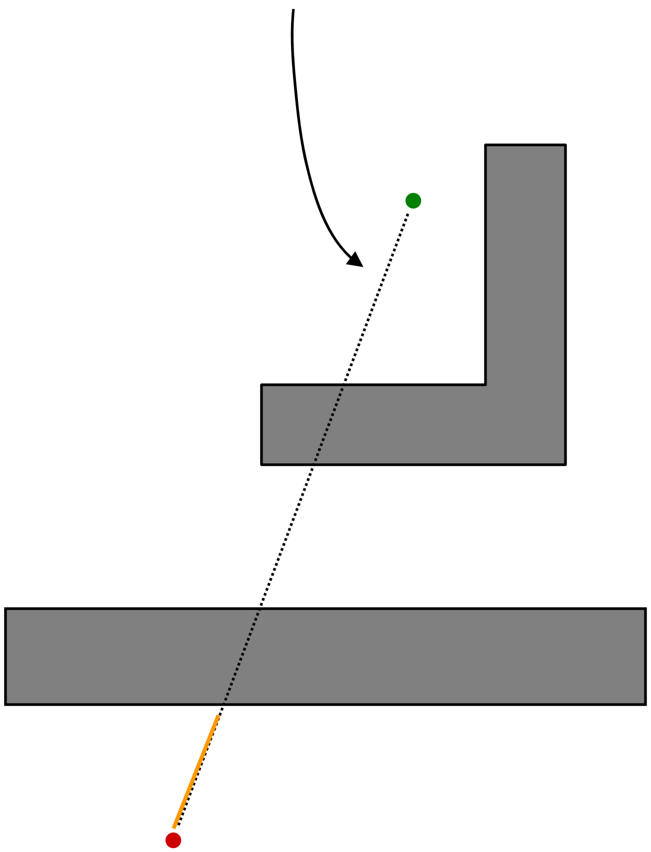
Call the line segment from the starting point to the goal the ***m-line***

"Bug 2" Algorithm



A better bug?

Call the line segment from the starting point to the goal the ***m-line***

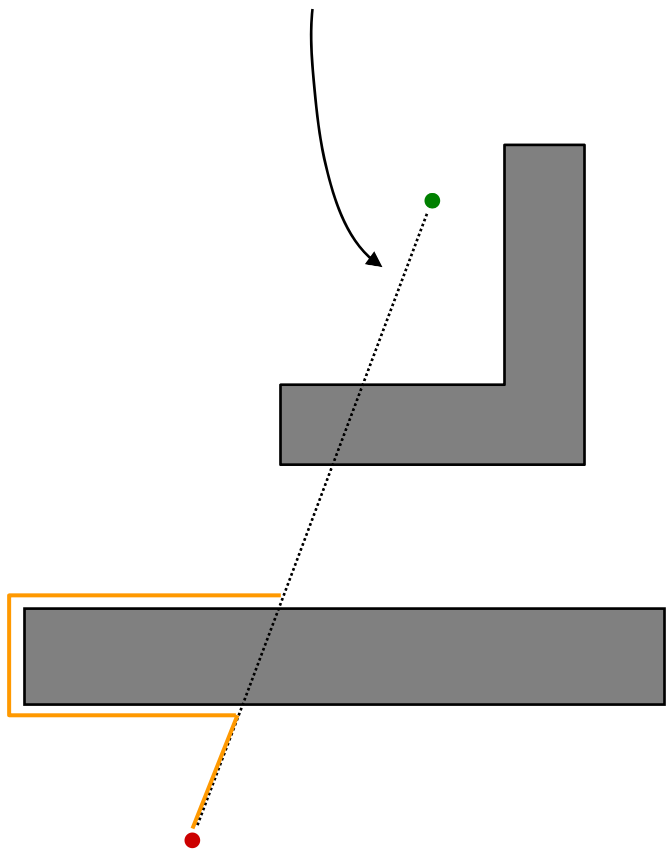


"Bug 2" Algorithm

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

A better bug?

Call the line segment from the starting point to the goal the ***m-line***

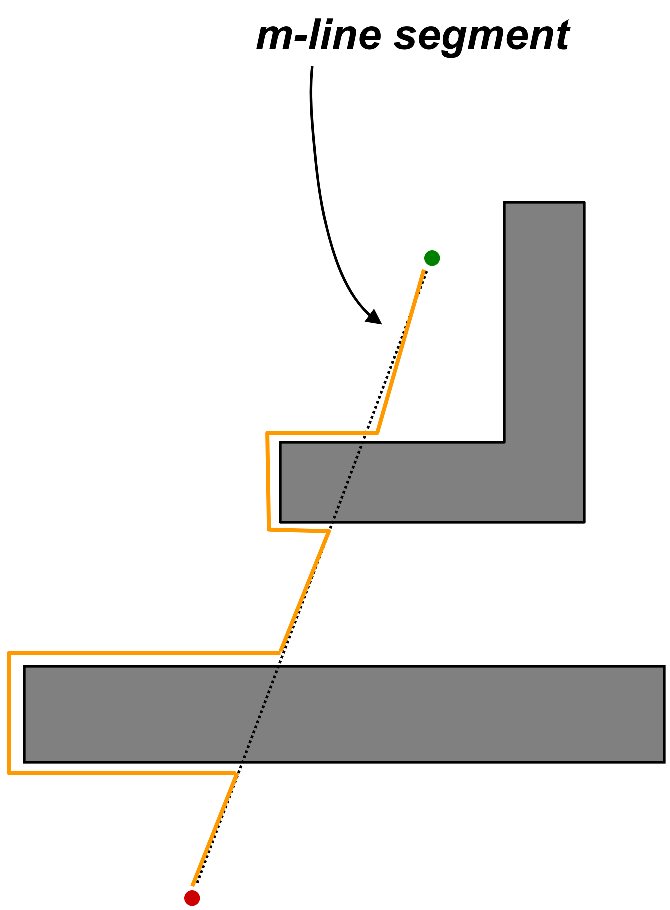


"Bug 2" Algorithm

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

2) if an obstacle is in the way, follow it until you encounter the m-line again.

A better bug?



"Bug 2" Algorithm

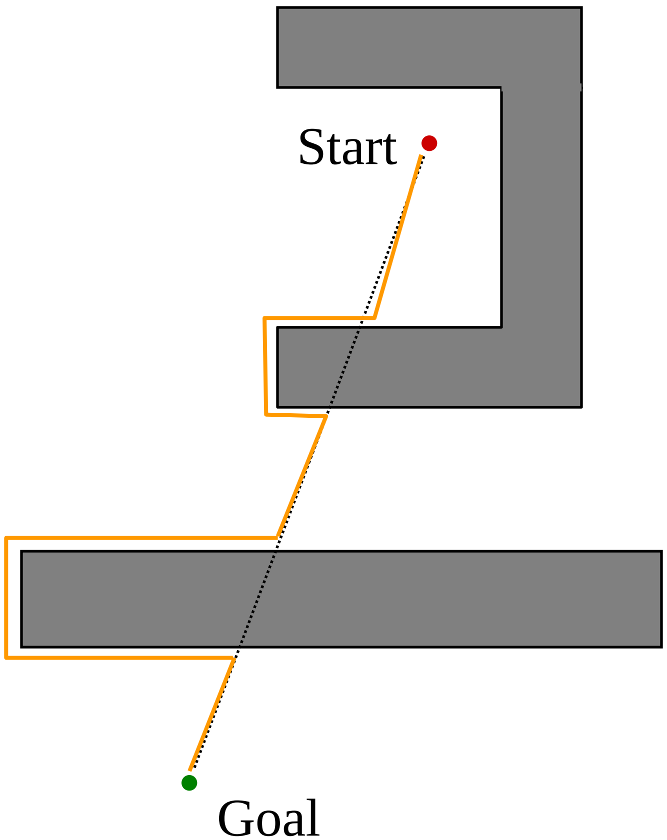
1) head toward goal on the *m-line*

2) if an obstacle is in the way, follow it until you encounter the m-line again.

3) Leave the obstacle and continue toward the goal

OK ?

A better bug?



"Bug 2" Algorithm

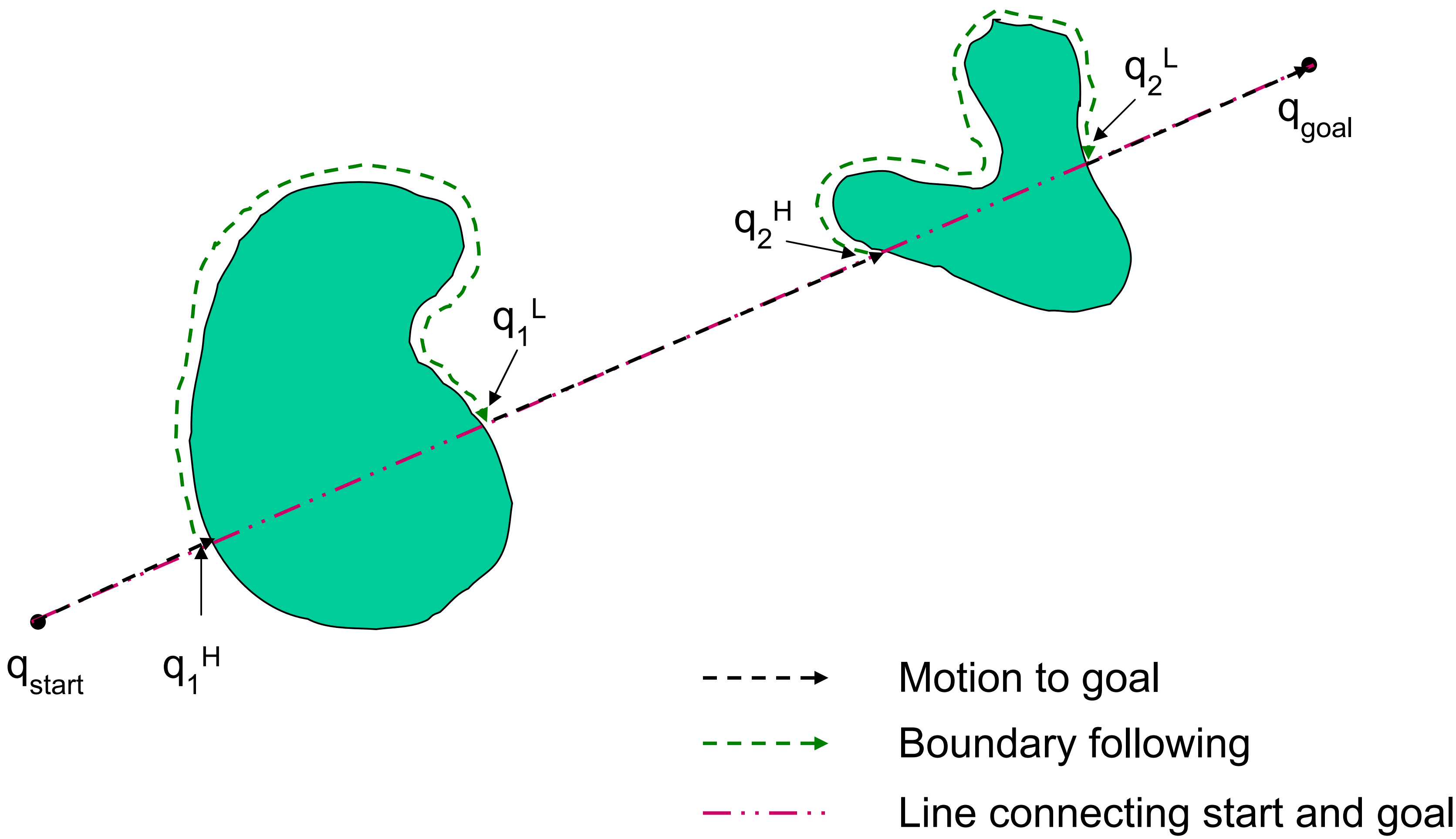
1) head toward goal on the *m-line*

2) if an obstacle is in the way, follow it until you encounter the m-line again ***closer to the goal***.

3) Leave the obstacle and continue toward the goal

Better or worse than Bug1?

Bug2 - example



Algorithm 2 Bug2 Algorithm

Input: A point robot with a tactile sensor

Output: A path to q_{goal} or a conclusion no such path exists

while True **do**
    repeat
        From q_{i-1}^{L}, move toward q_{goal} along m-line.
    until
    q_{goal} is reached **or**
    an obstacle is encountered at *hit point* q_i^H.
    Turn left (or right).
    repeat
        Follow boundary
    until
      q_{goal} is reached **or**
        q_i^H is re-encountered **or**
        m-line is re-encountered at a point m such that
          $m \neq q_i^H$ (robot did not reach the hit point),
          $d(m, q_{\text{goal}}) < d(m, q_i^H)$ (robot is closer), and
          if robot moves toward goal, it would not hit the obstacle
    if Goal is reached **then**
        Exit.
    end if
    if q_i^H is re-encountered **then**
        Conclude goal is unreachable
    end if
    Let $q_{i+1}^{L} = m$
    Increment i
end while

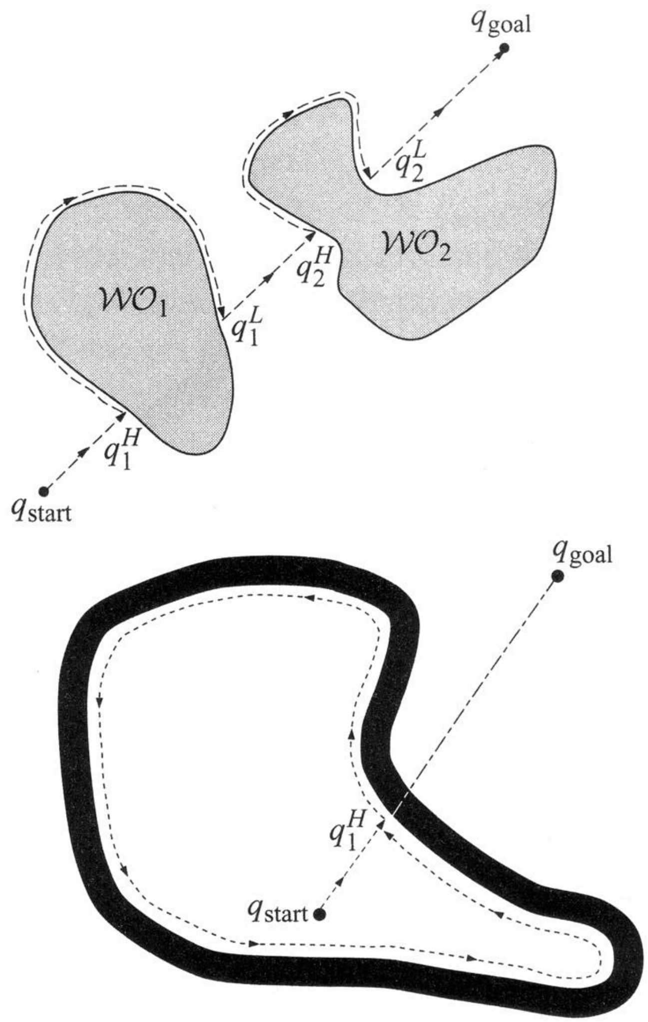


Figure 2.3 (Top) The Bug2 algorithm finds a path to the goal. (Bottom) The Bug2 algorithm reports failure.

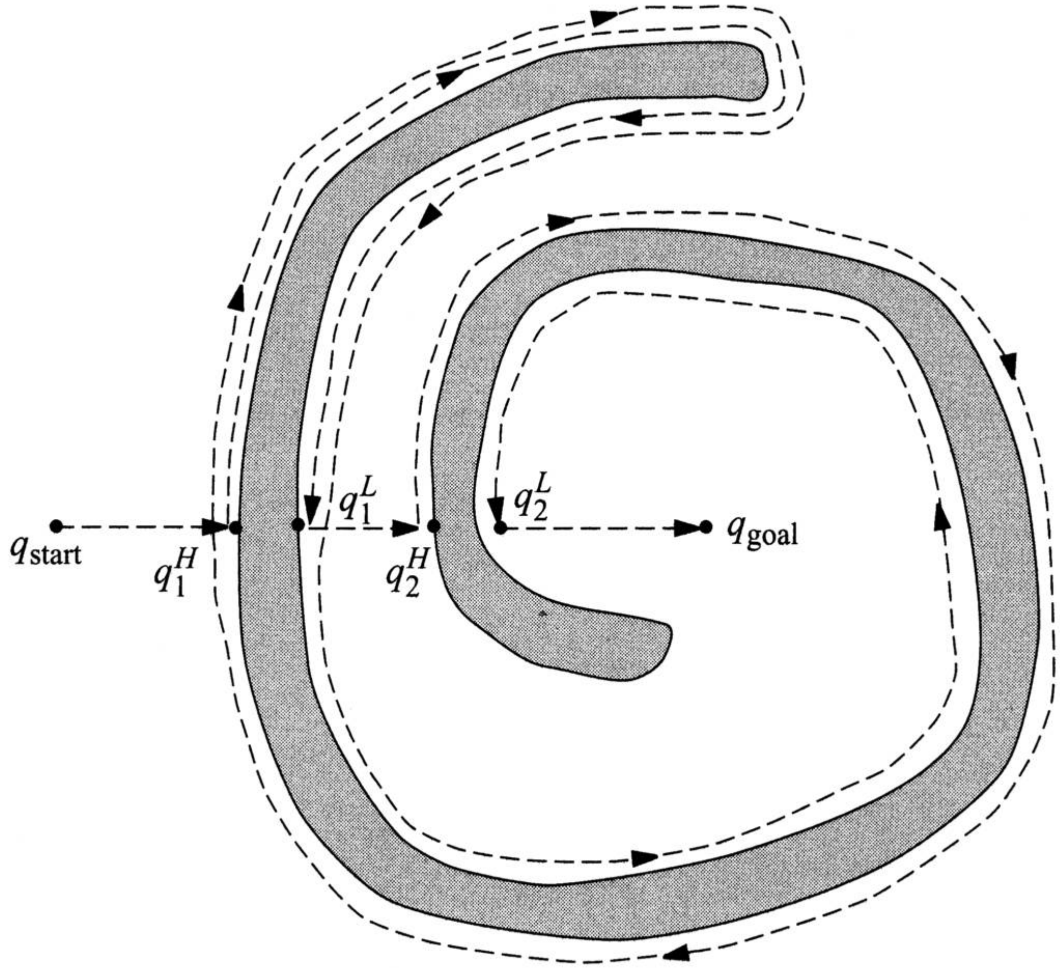


Figure 2.4 Bug2 Algorithm.

head-to-head comparison

or thorax-to-thorax, perhaps

Draw worlds in which Bug 2 does better than Bug 1 (and vice versa).

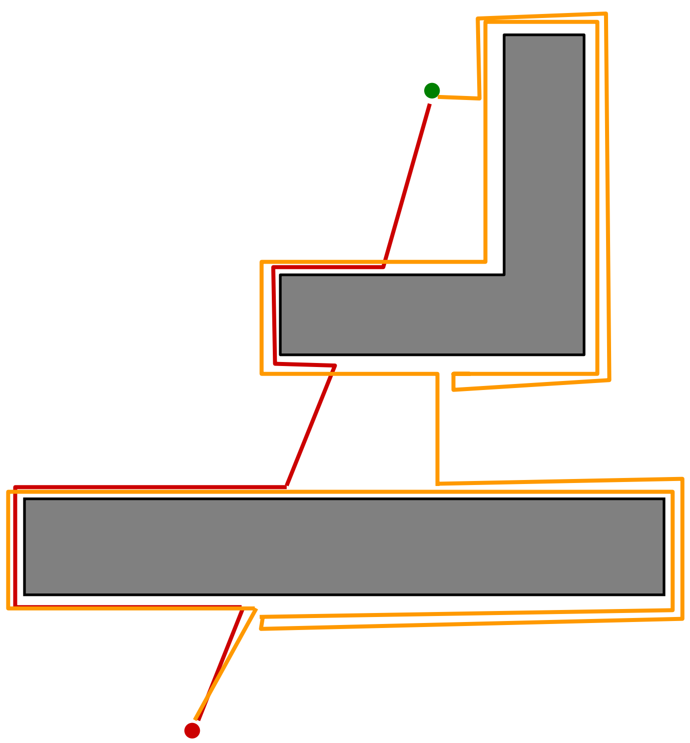
Bug 2 beats Bug 1	**Bug 1 beats Bug 2**

head-to-head comparison

or thorax-to-thorax, perhaps

Draw worlds in which Bug 2 does better than Bug 1 (and vice versa).

Bug 2 beats Bug 1



Bug 1 beats Bug 2

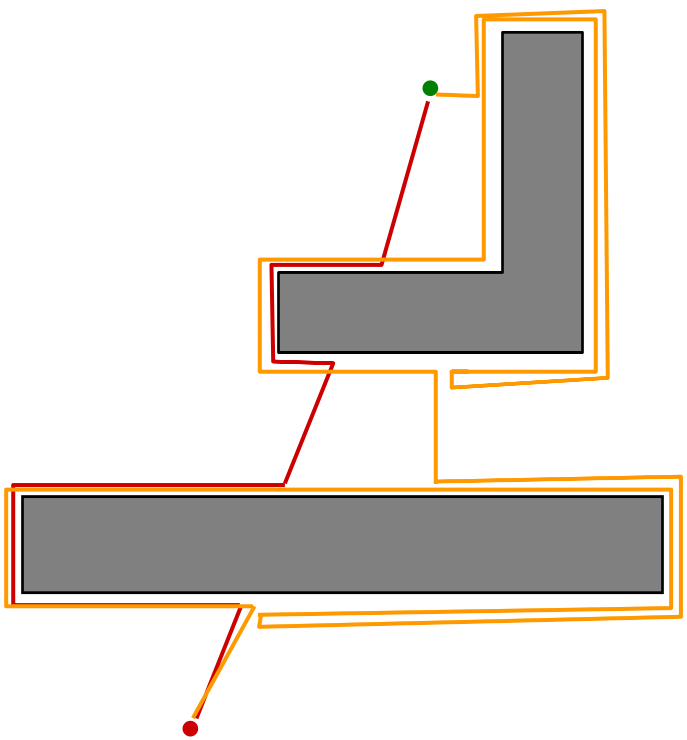
?

head-to-head comparison

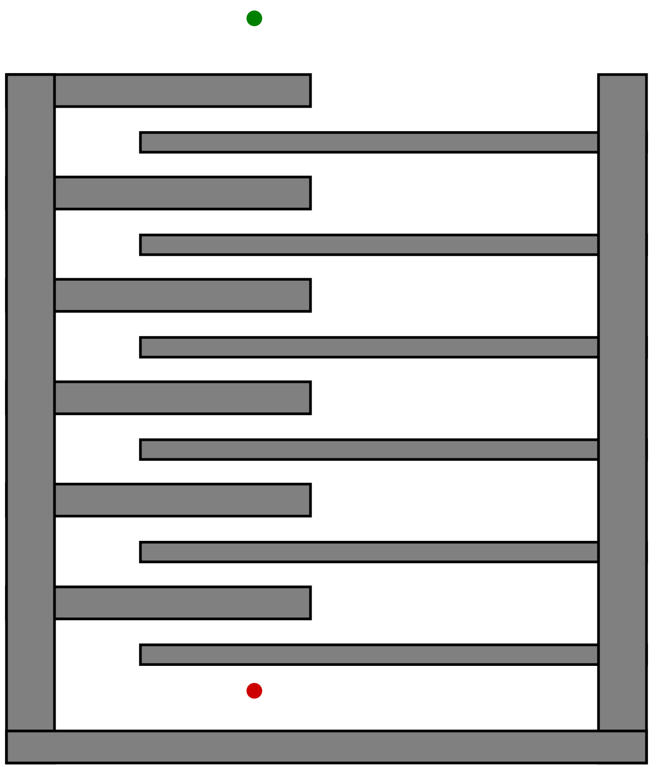
or thorax-to-thorax, perhaps

Draw worlds in which Bug 2 does better than Bug 1 (and vice versa).

Bug 2 beats Bug 1



Bug 1 beats Bug 2



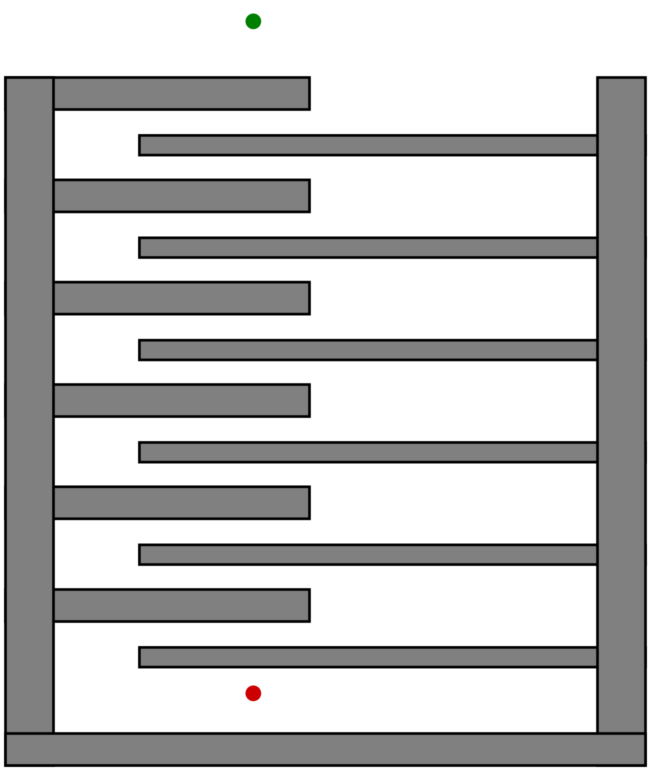
BUG 1 vs. BUG 2

- BUG 1 is an *exhaustive search algorithm*
  - it looks at all choices before commiting
- BUG 2 is a *greedy* algorithm
  - it takes the first thing that looks better
- In many cases, BUG 2 will outperform BUG 1, but
- BUG 1 has a more predictable performance overall

“Quiz” Bug 2 analysis

Bug 2: Path Bounds

What are upper/lower bounds on the path length that the robot takes?



D = straight-line distance from start to goal

P_i = perimeter of the *i* th obstacle

Lower bound:
What’s the shortest distance it might travel?

D

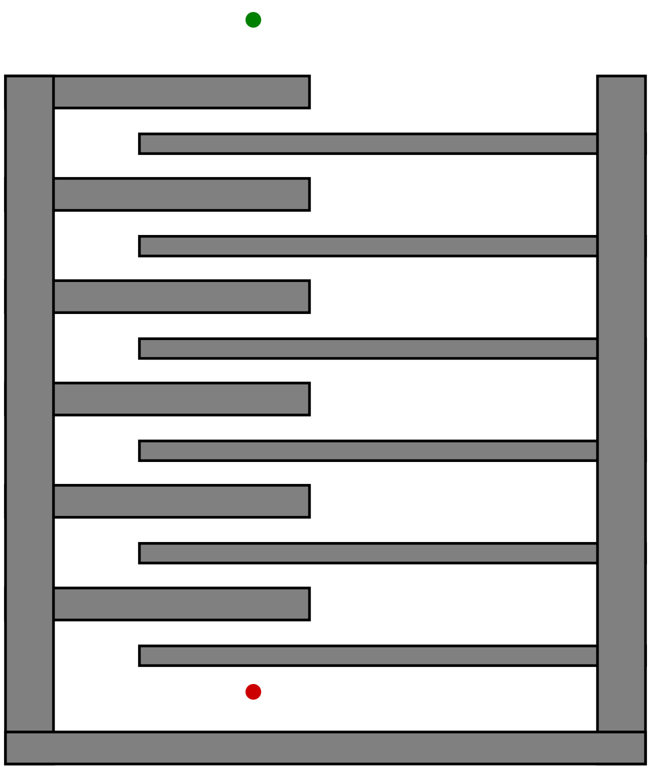
Upper bound:
What’s the longest distance it might travel?

What is an environment where your upper bound is required?

"Quiz" Bug 2 analysis

Bug 2: Path Bounds

What are upper/lower bounds on the path length that the robot takes?



D = straight-line distance from start to goal

P_i = perimeter of the *i* th obstacle

Lower bound: $\mathbf{D}$

What's the shortest distance it might travel?

Upper bound: $\mathbf{D} + \sum_i \frac{n_i}{2} P_i$

What's the longest distance it might travel?

$\mathbf{n_i}$ = # of m-line intersections of the *i* th obstacle

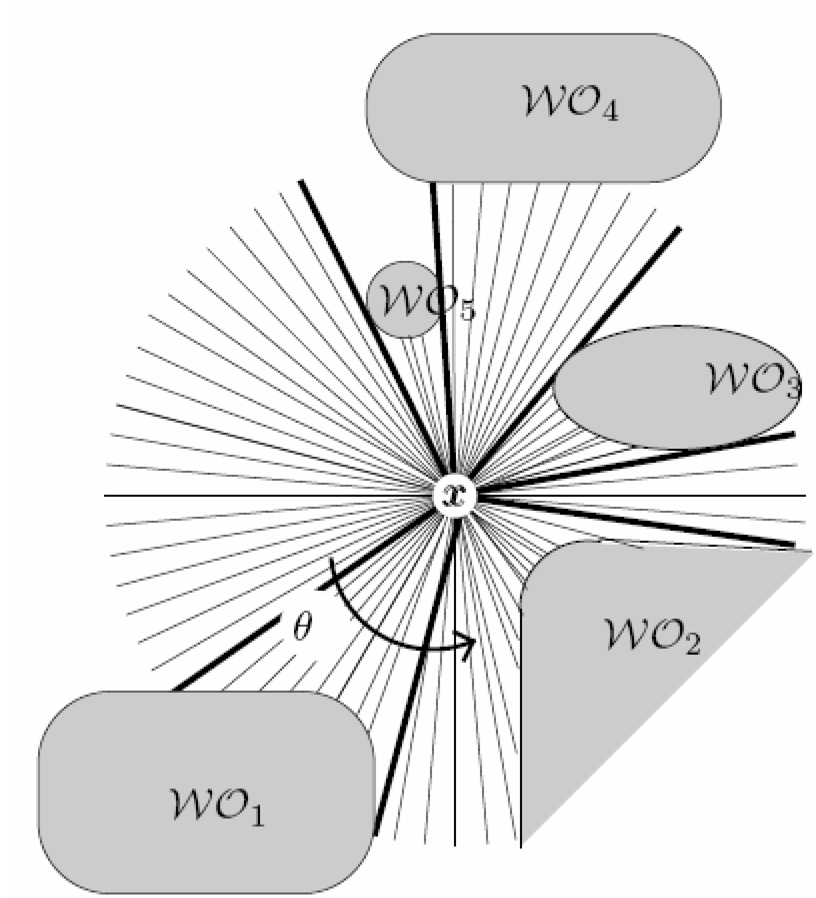
What is an environment where your upper bound is required?

A More Realistic Bug

- As presented: global beacons plus contact-based wall following
- The reality: we typically use some sort of range sensing device that lets us look ahead (but has finite resolution and is noisy).
- Let us assume we have a range sensor

Raw Distance Function

$$\rho : \mathbb{R}^2 \times S^1 \to \mathbb{R}$$

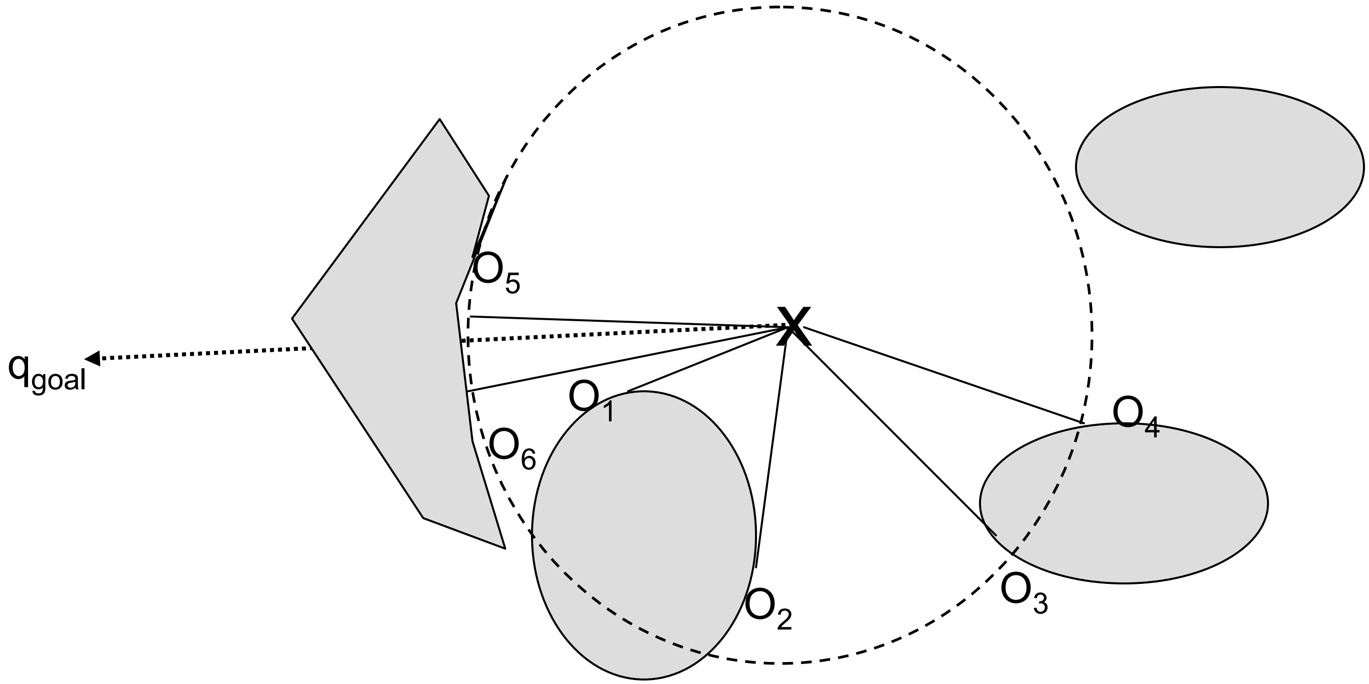


Saturated raw distance function

$$\rho_R(x,\theta) = \begin{cases} \rho(x,\theta), & \text{if } \rho(x,\theta) < R \\ \infty, & \text{otherwise.} \end{cases}$$

Tangent Bug

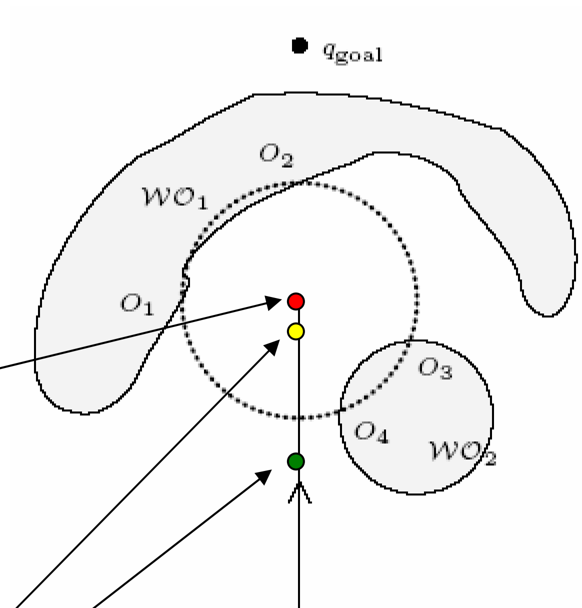
- Tangent Bug relies on finding endpoints of finite, conts segments of ρ_R



Problem: what if this distance starts to go up?
Ans: start to act like a BUG and follow boundary

`Behavior is a combination of moving to goal and obstacle following`

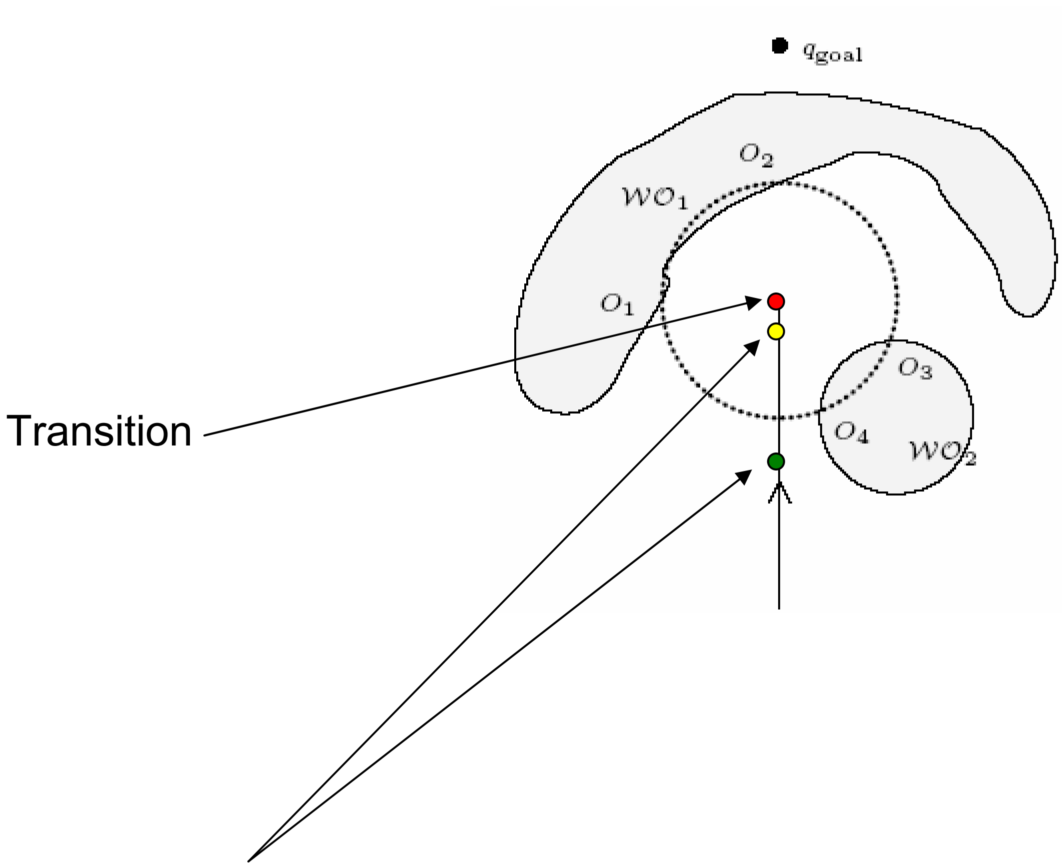
Motion-to-Goal Transition
from Moving Toward goal to “following obstacles”



Transition

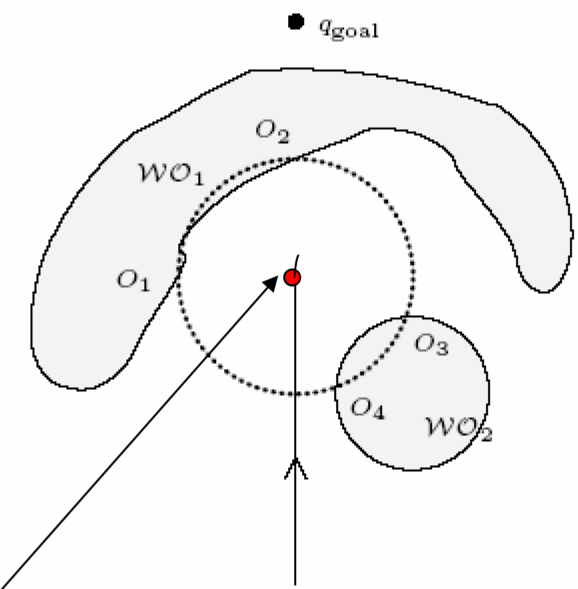
Currently, the motion-to-goal behavior “thinks” the robot can get to the goal

Motion-to-Goal Transition
From Moving Toward goal to “following obstacles”



Currently, the motion-to-goal behavior “thinks” the robot can get to the goal

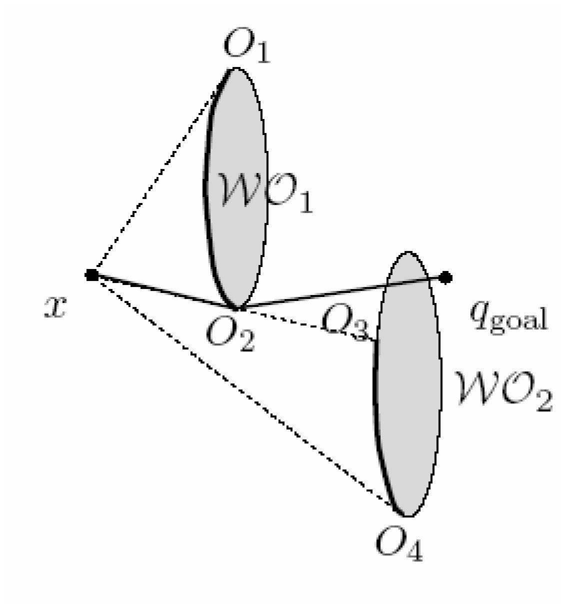
Motion-to-Goal Transition
Minimize Heuristic



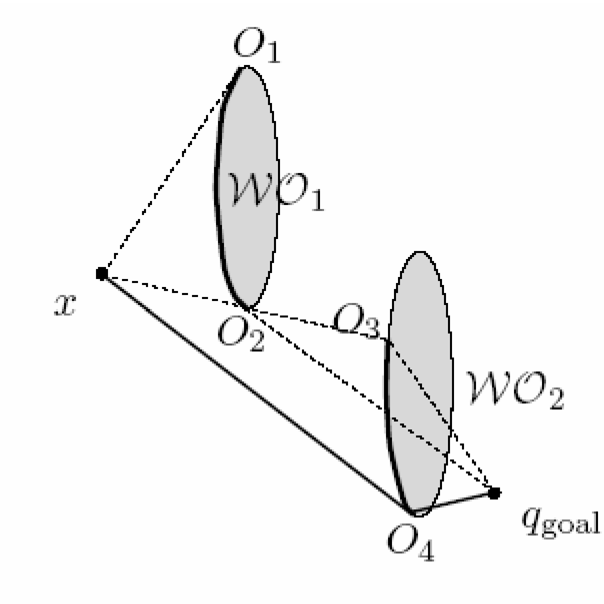
Now, it starts to see something --- what to do?
Ans: Choose the pt O_i that minimizes $d(x,O_i) + d(O_i,q_{goal})$

Minimize Heuristic Example

At x, robot knows only what it sees and where the goal is,



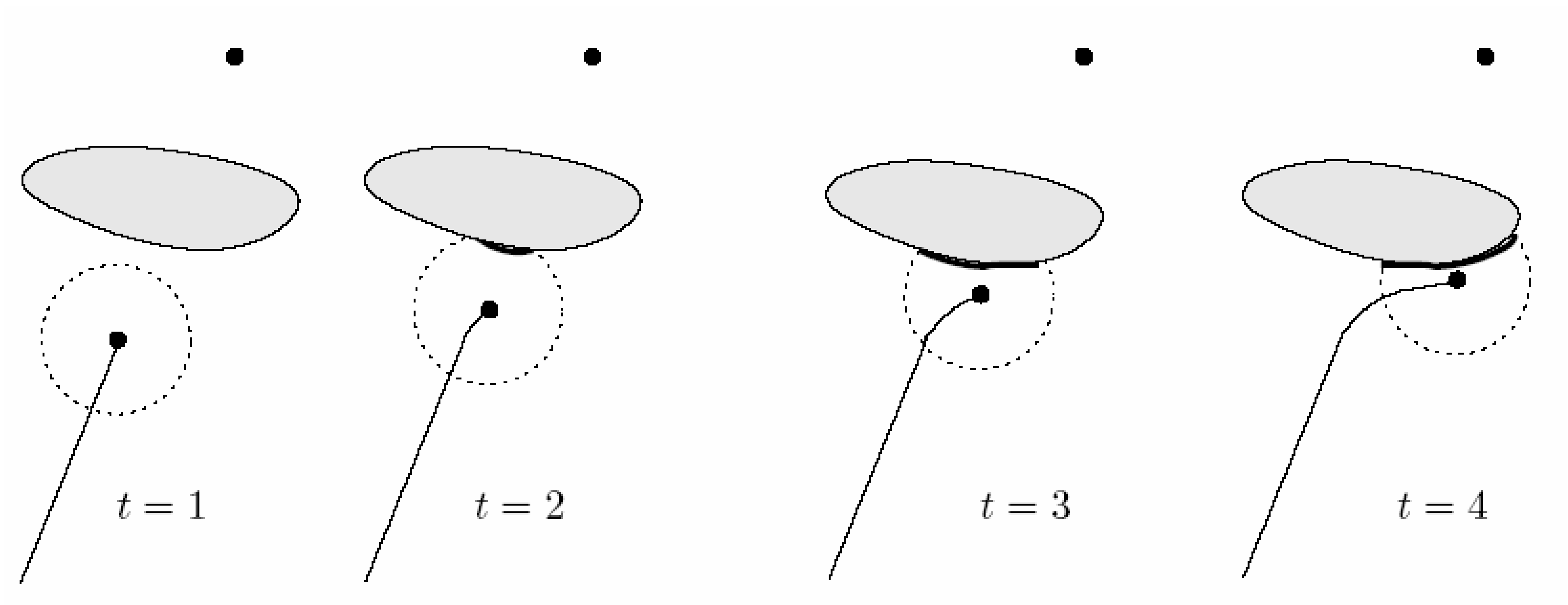
so moves toward O_2. Note the line connecting O_2 and goal pass through obstacle



so moves toward O_4. Note some "thinking" was involved and the line connecting O_4 and goal pass through obstacle

Choose the pt O_i that minimizes $d(x,O_i) + d(O_i,q_{goal})$

Motion To Goal Example



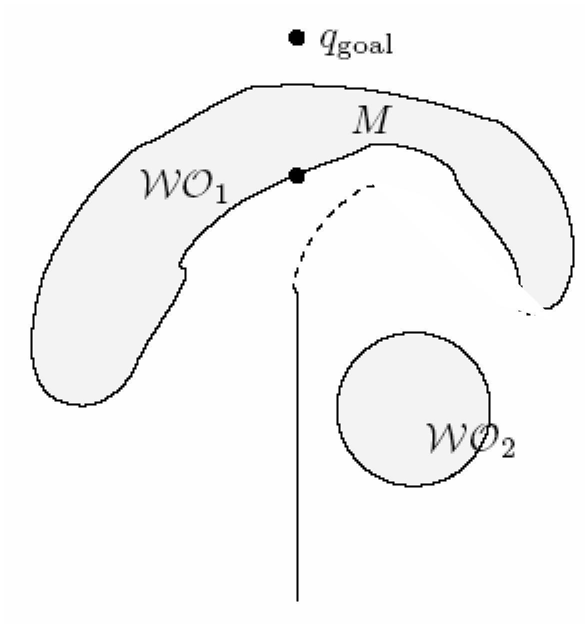
Choose the pt O_i that minimizes $d(x,O_i) + d(O_i,q_{goal})$

Transition *from* Motion-to-Goal

Choose the pt O_i that minimizes $d(x,O_i) + d(O_i,q_{goal})$

Problem: what if this distance starts to go up?

Ans: start to act like a BUG and follow boundary



M is the point on the "sensed" obstacle which has the shorted distance to the goal

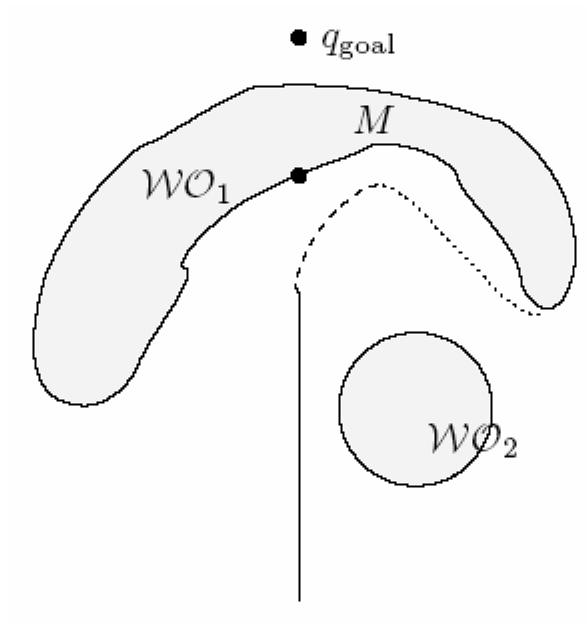
Followed obstacle: the obstacle that we are currently sensing

Blocking obstacle: the obstacle that intersects the segment $(1-\lambda)x + \lambda q_{\text{goal}} \quad \forall \lambda \in [0,1]$

They start as the same

Boundary Following

Move toward the O_i on the followed obstacle in the "chosen" direction



M is the point on the "sensed" obstacle which has the shorted distance to the goal

Followed obstacle: the obstacle that we are currently sensing

Blocking obstacle: the obstacle that intersects the segment

Maintain $d_{followed}$ and d_{reach}

They start as the same

$d_{followed}$ and d_{reach}

$d_{followed}$ is the shortest distance between the sensed boundary and the goal

- d_{reach} is the shortest distance between *blocking* obstacle and goal (or my distance to goal if no blocking obstacle visible)

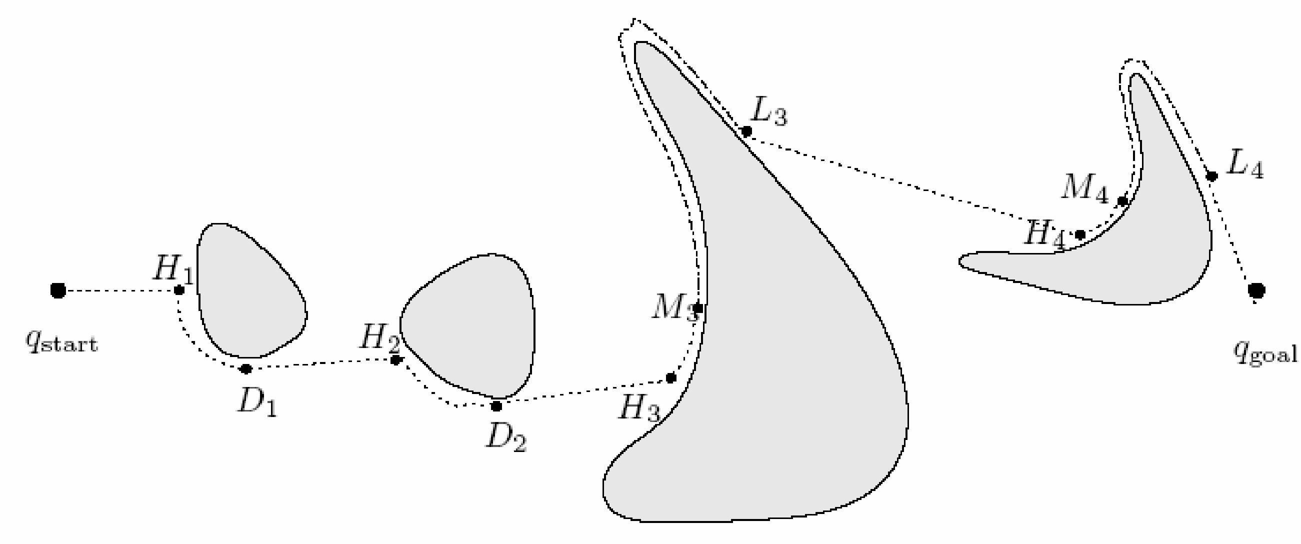
$$\Lambda = \{y \in \partial \mathcal{WO}_b : \lambda x + (1-\lambda) y \in \mathcal{Q}_{\text{free}} \quad \forall \lambda \in [0,1]\}.$$

$$d_{\text{reach}} = \min_{c \in \Lambda} d(q_{\text{goal}}, c)$$

- Terminate boundary following behavior when $d_{reach} < d_{followed}$

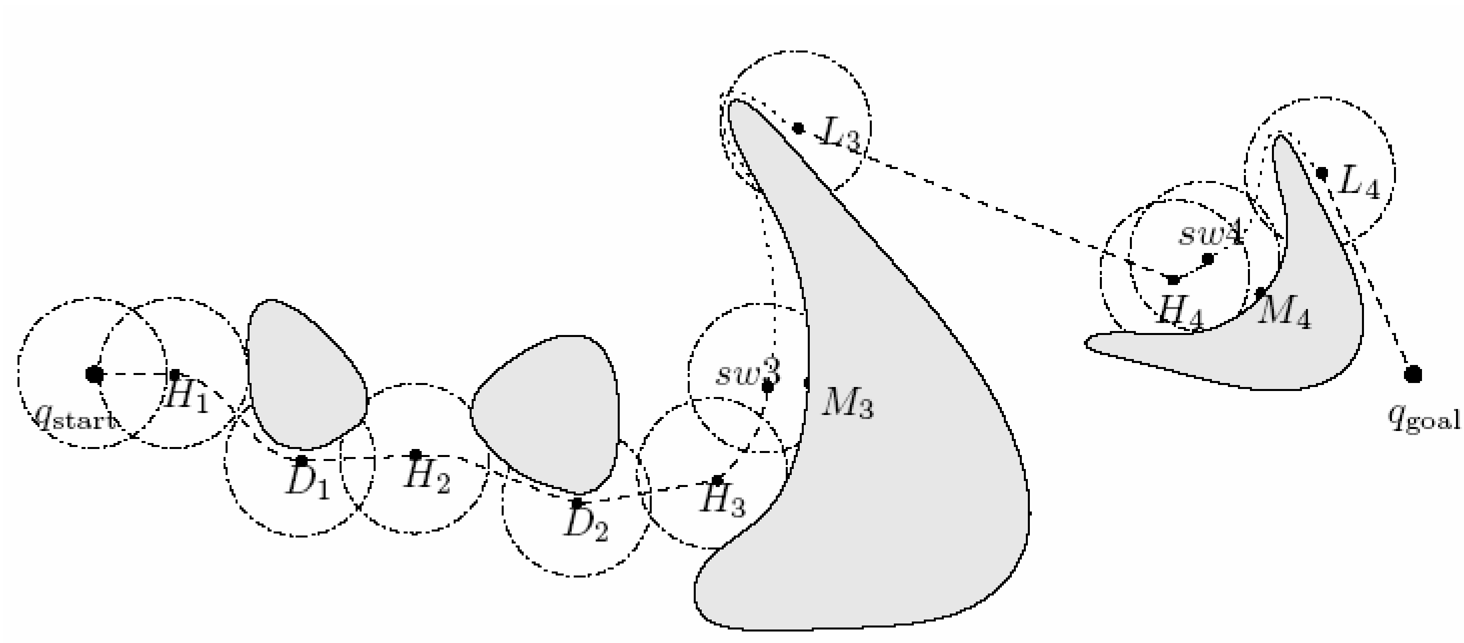
Note: d_followed = d_min, d_reach = d_leave in Chapter 2 Bug Algorithms text

Example: Zero Senor Range

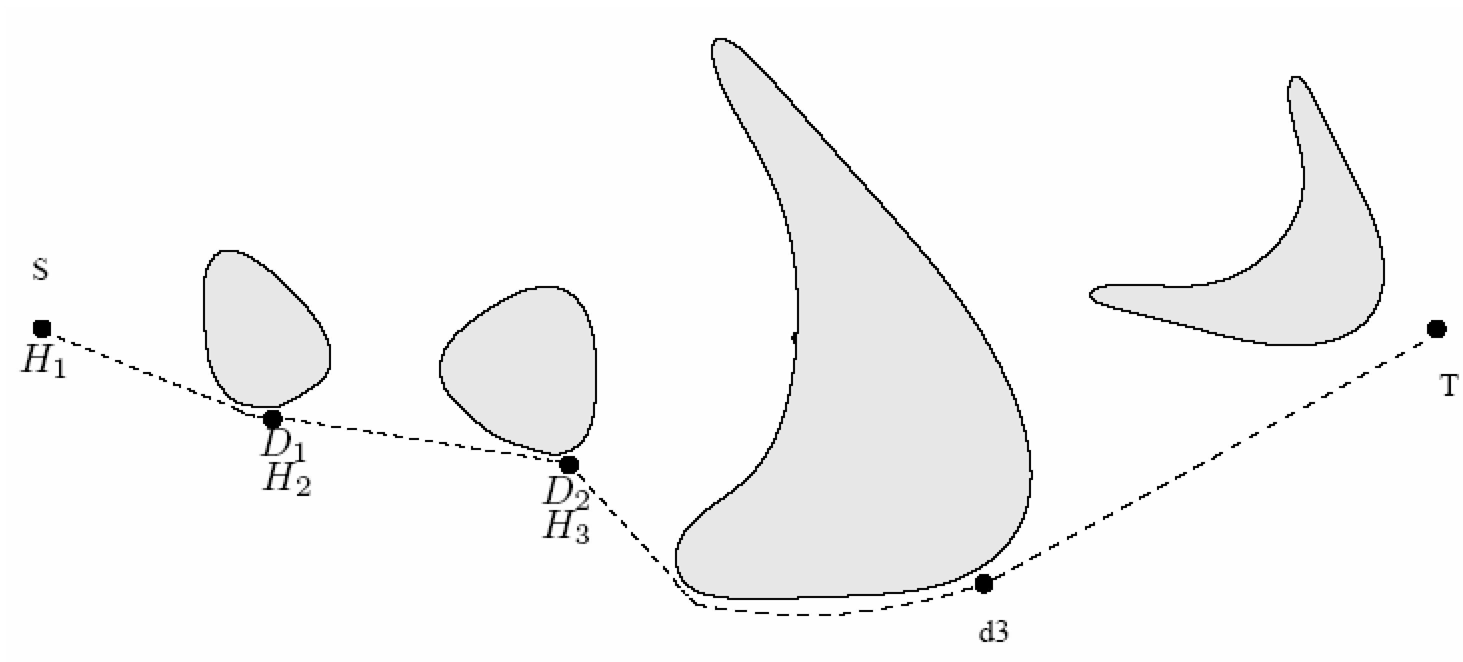


1. Robot moves toward goal until it hits obstacle 1 at H1
2. Pretend there is an infinitely small sensor range and the Oi which minimizes the heuristic is to the right
3. Keep following obstacle until robot can go toward goal again
4. Same situation with second obstacle
5. At third obstacle, the robot 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

Example: Finite Sensor Range

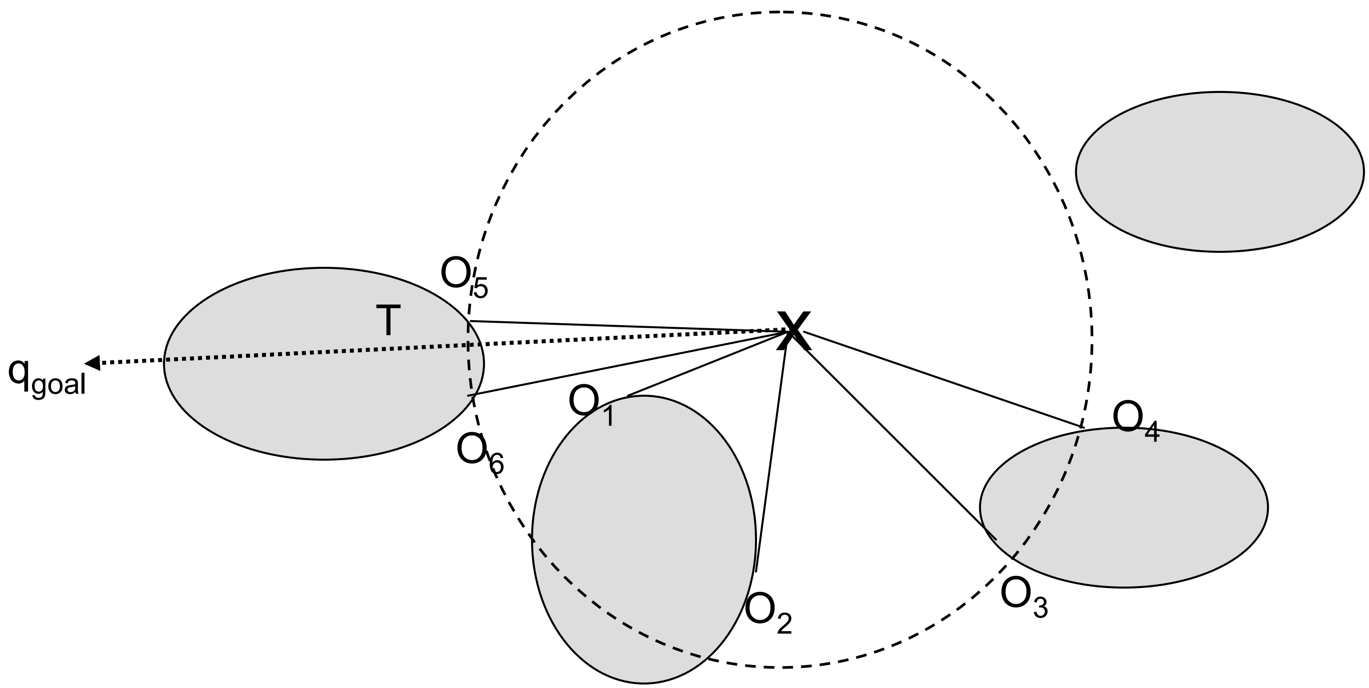


Example: Infinite Sensor Range



Tangent Bug

- Tangent Bug relies on finding endpoints of finite, conts segments of ρ_R



Now, it starts to see something --- what to do?
Ans: Choose the pt O_i that minimizes $d(x,O_i) + d(O_i,q_{goal})$
"Heuristic distance"

Tangent Bug Algorithm

1) repeat
   a) Compute continuous range segments in view
   b) Move toward $n \in \{T, O_i\}$ that minimizes $h(x,n) = d(x,n) + d(n,q_{goal})$

   until
   a) goal is encountered, or
   b) the value of h(x,n) begins to increase

2) follow boundary continuing in same direction as before repeating
   a) update $\{O_i\}$, d_{reach} and $d_{followed}$

   until
   a) goal is reached
   b) a complete cycle is performed (goal is unreachable)
   c) $d_{reach} < d_{followed}$

Note the same general proof reasoning as before applies, although the definition of hit and leave points is a little trickier. In the text,

o d_reach == d_leave and d_followed==d_min

The Basic Ideas

- A motion-to-goal behavior as long as way is clear or there is a visible obstacle boundary pt that decreases heuristic distance
- A boundary following behavior invoked when heuristic distance increases.
- A value $d_{followed}$ which is the shortest distance between the sensed boundary and the goal
- A value d_{reach} which is the shortest distance between *blocking* obstacle and goal (or my distance to goal if no blocking obstacle visible)
- Terminate boundary following behavior when $d_{reach} < d_{followed}$

Summary

- Bug 1: safe and reliable
- Bug 2: better in some cases; worse in others
- Bug 2 is greedy algorithm
- Tangent Bug: supports range sensing
- Sensors and control
  - should understand basic concepts and know what different sensors are - next lecture!