Robotic Motion Planning: Bug Algorithms

(with some discussion on curve tracing and sensors)

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

- Tangent Bug deals with finite distance sensing

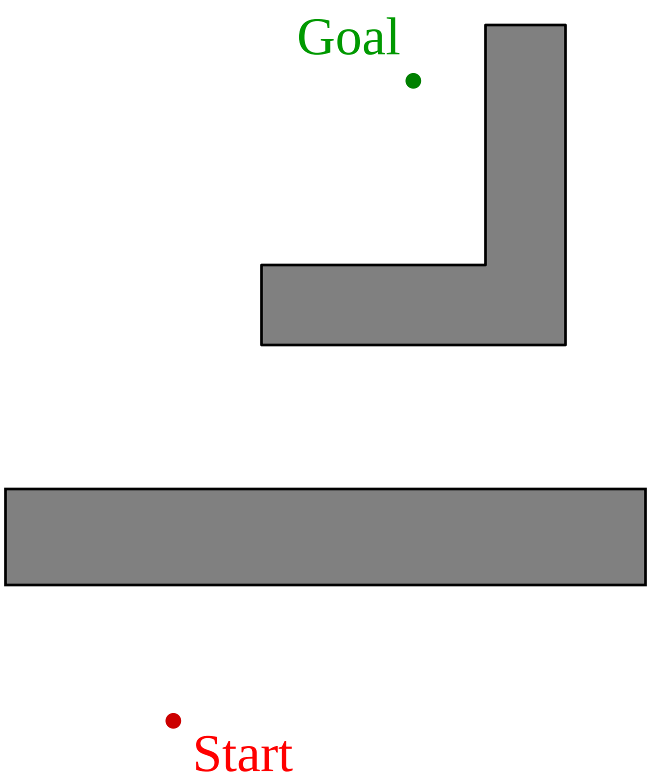
A Few General Concepts

- 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

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

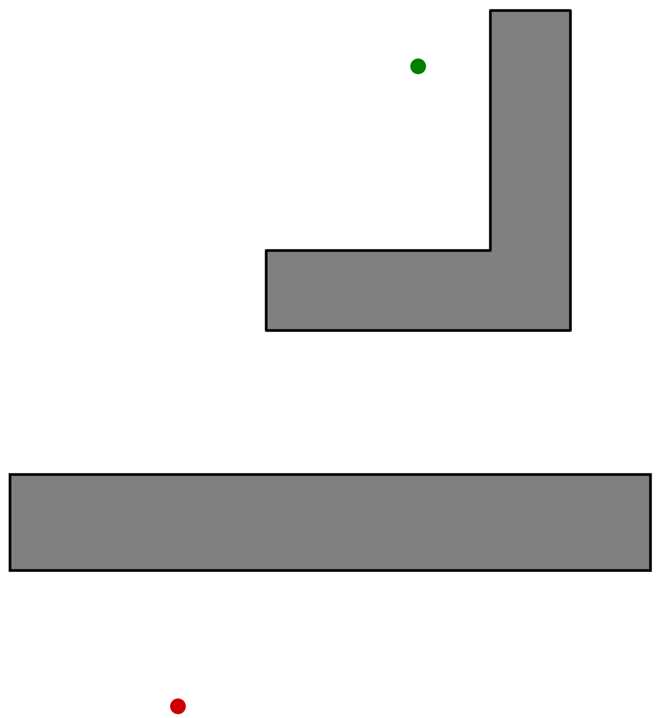
  $B_r(x) = \{\ y \in \Re(2)\ |\ d(x,y) < r\ \}$

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}

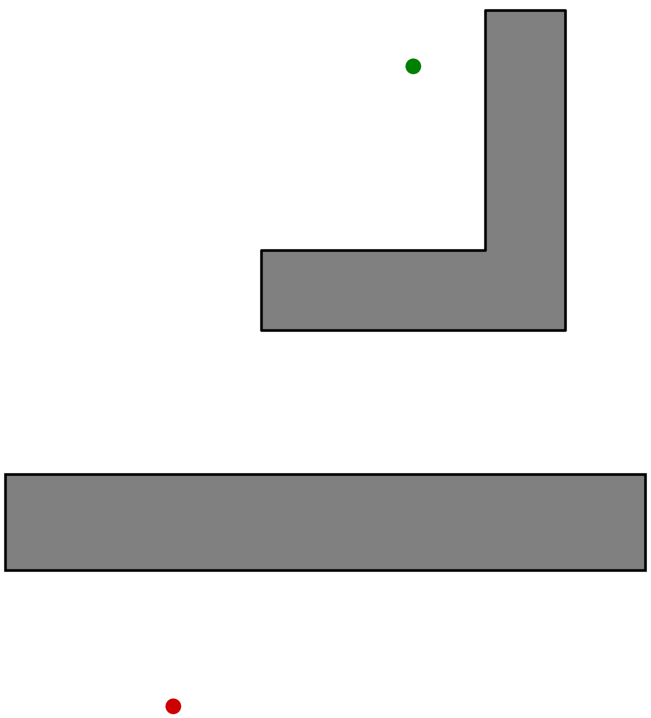
how ?

Buginner Strategy

"Bug 0" algorithm

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

  walls/obstacles & encoders



1) head toward goal

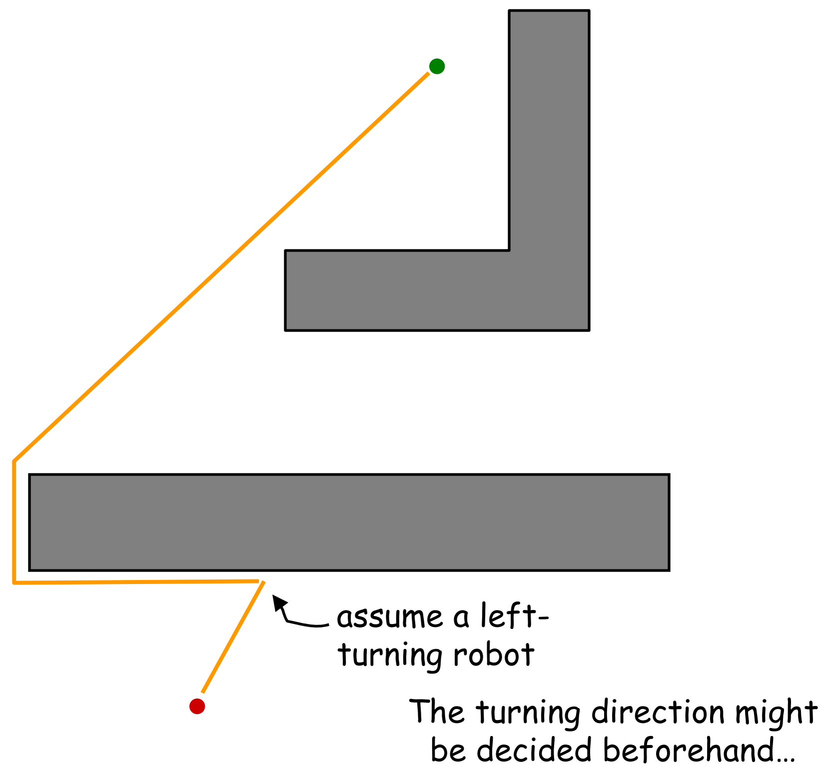
2) follow obstacles until you can head toward the goal again

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

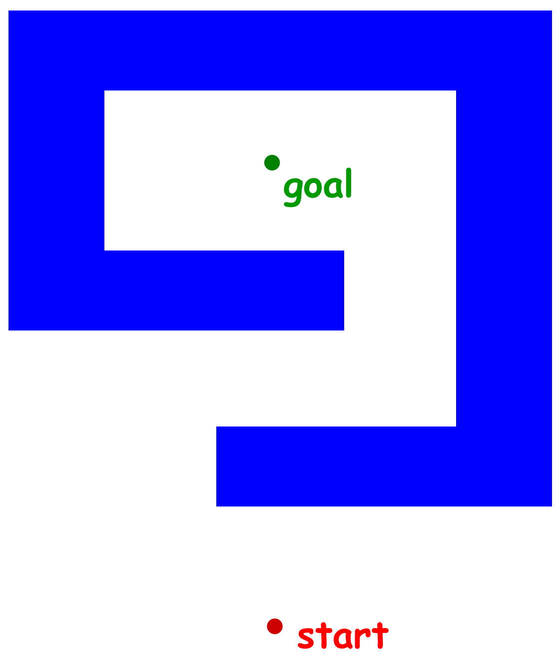
1) head toward goal

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

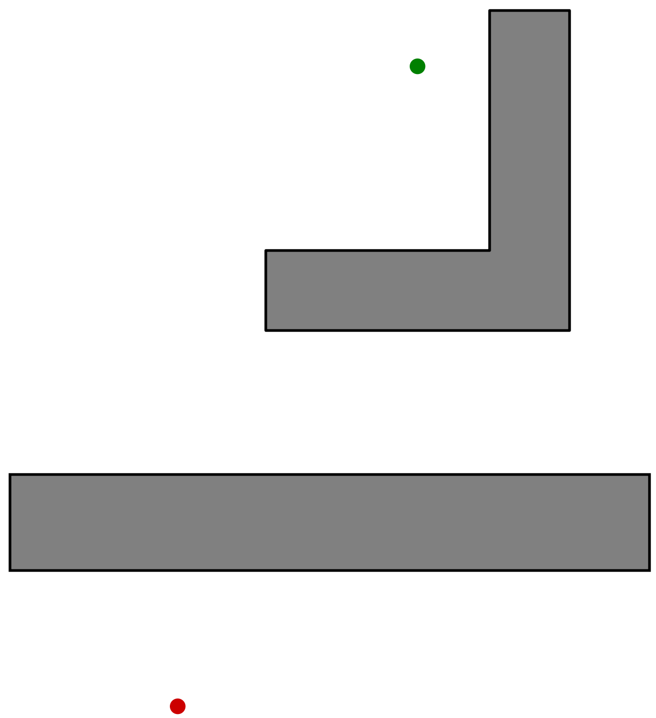
3) continue

A better bug?

But add some memory!

- known direction to goal
- otherwise local sensing

walls/obstacles & encoders

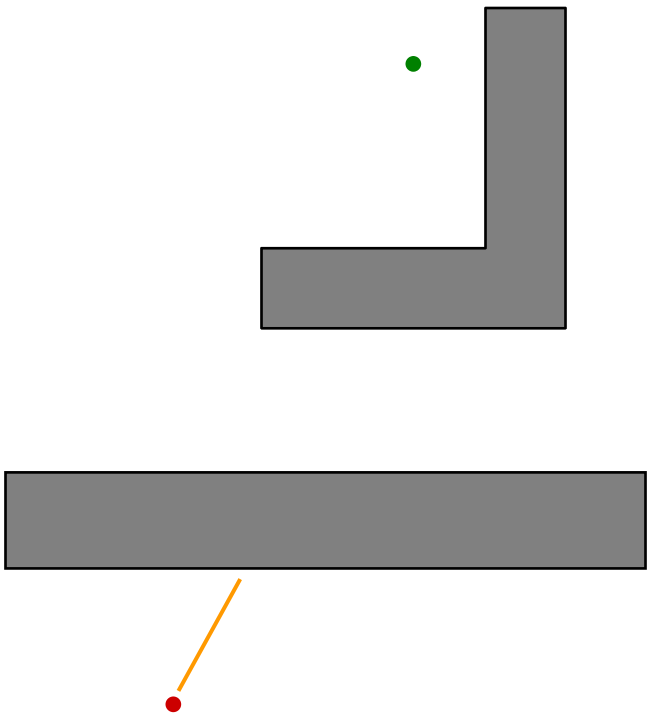


improvement ideas?

Bug 1

But <u>some</u> 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

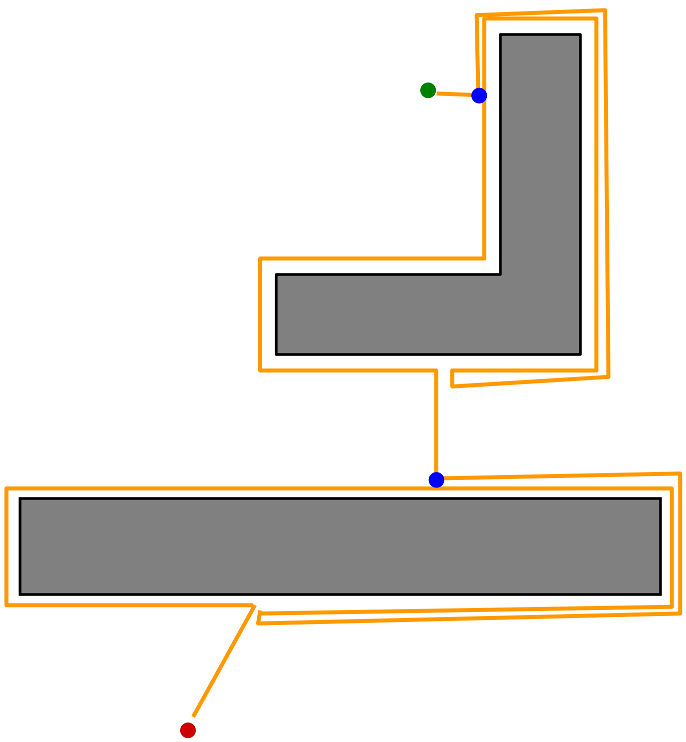
3) return to that closest point (by wall-following) and continue

Vladimir Lumelsky & Alexander Stepanov: <u>Algorithmica</u> 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

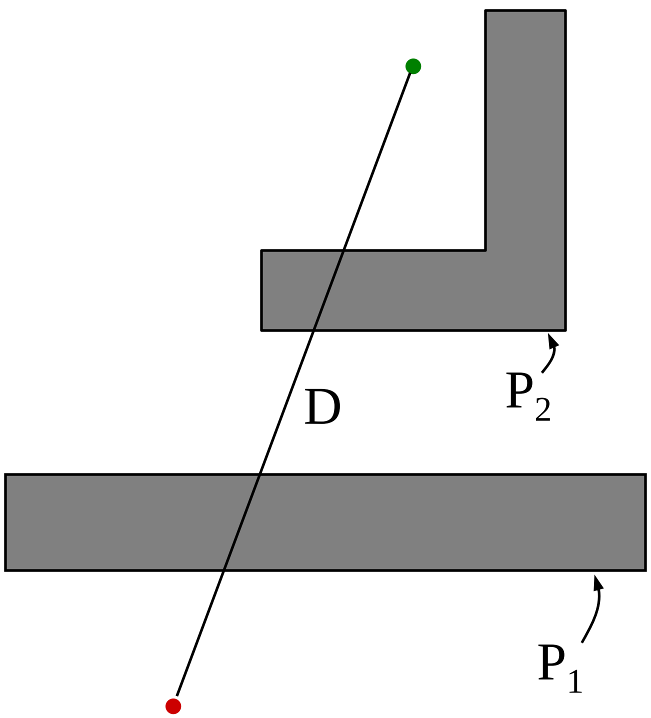
BUG 1 More formally

- 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

"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:

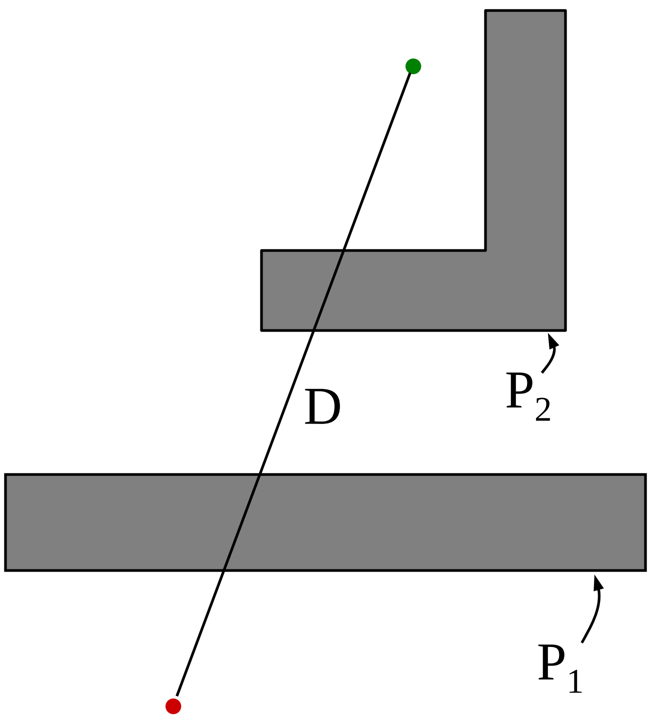
What's the longest distance it might travel?

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?

$\mathbf{D}$

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?

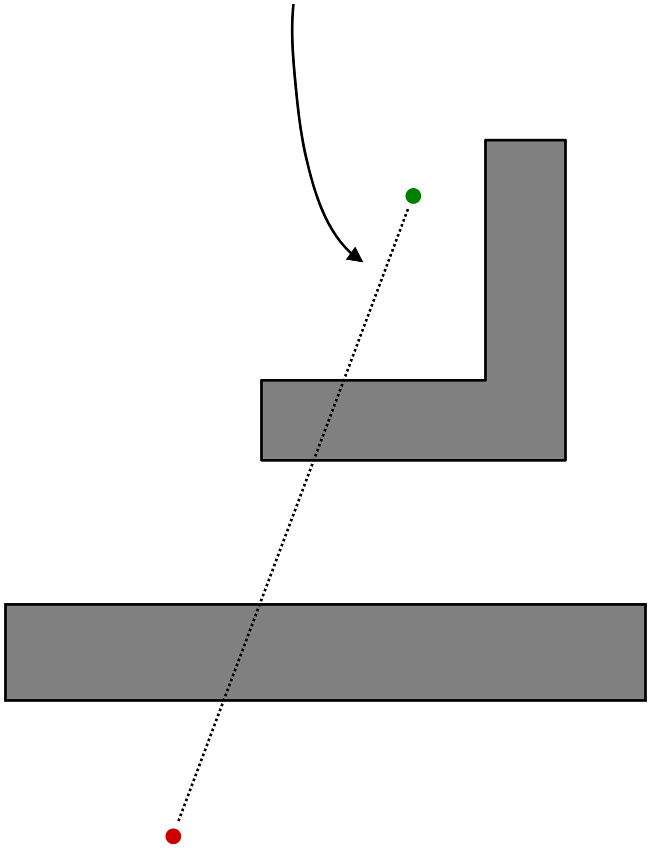
- An algorithm is *complete* if, in finite time, it finds a path if such a path exists or terminates with failure if it does not.

- 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?

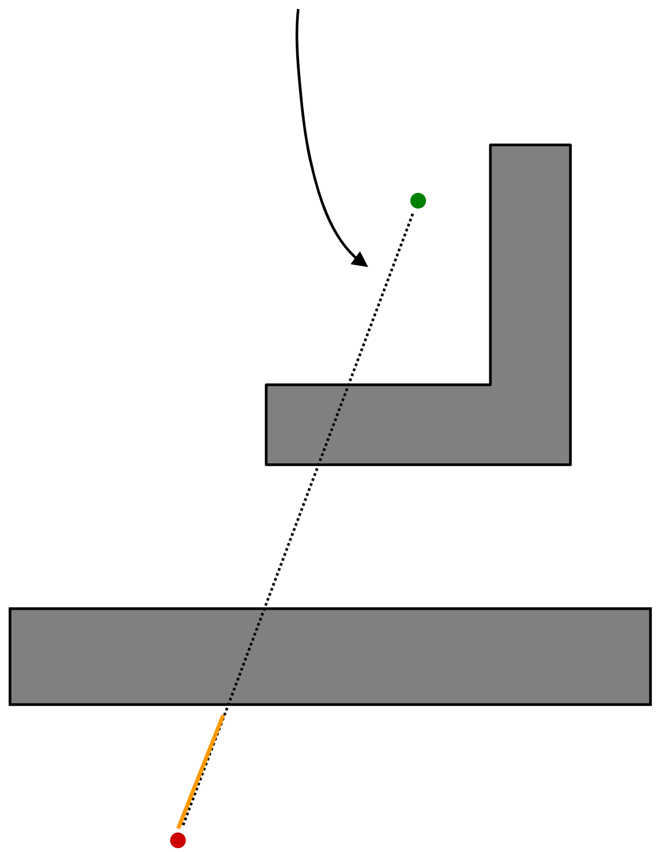
Call the line from the starting point to the goal the ***m-line***

"Bug 2" Algorithm



A better bug?

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

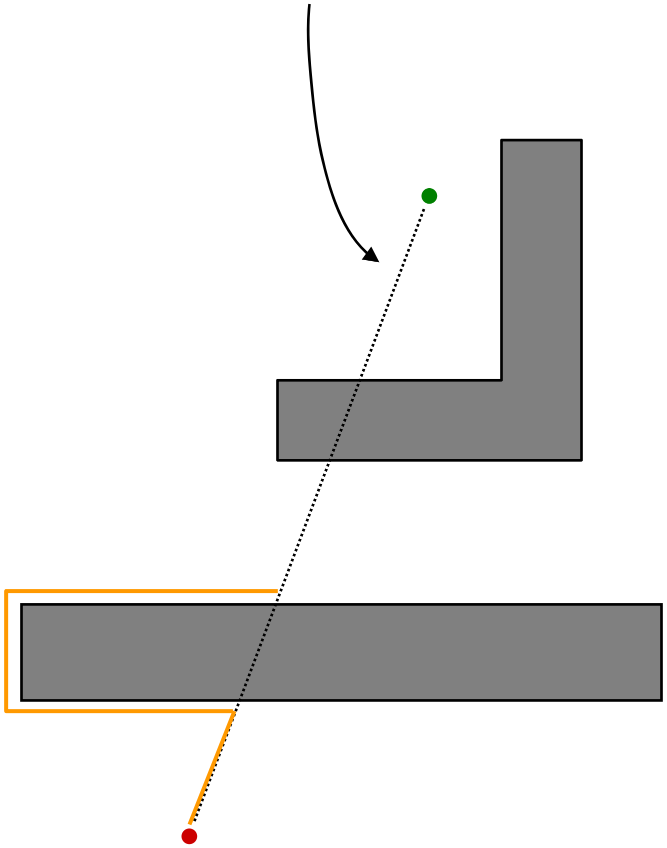


"Bug 2" Algorithm

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

A better bug?

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

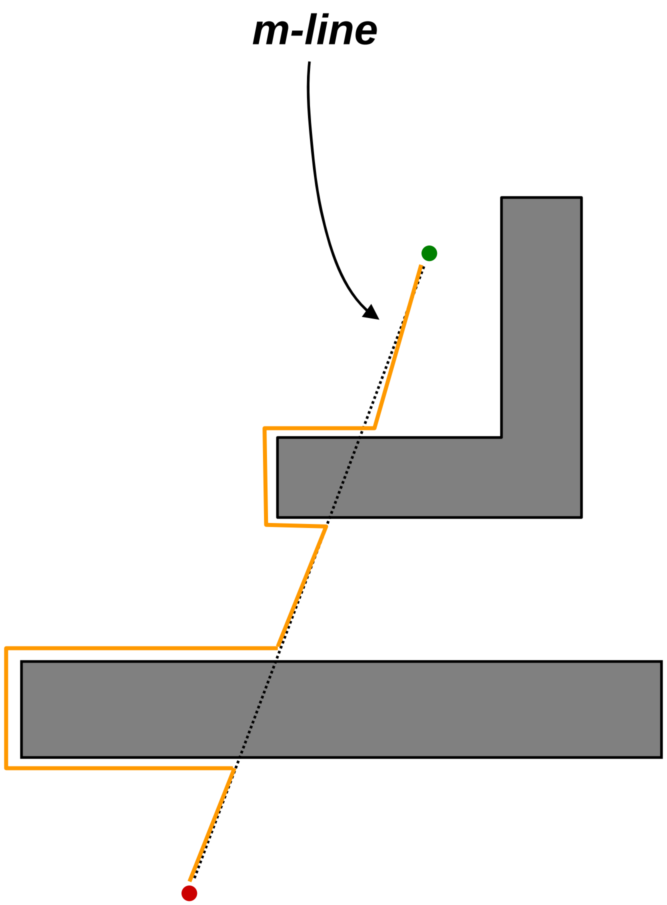


"Bug 2" Algorithm

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

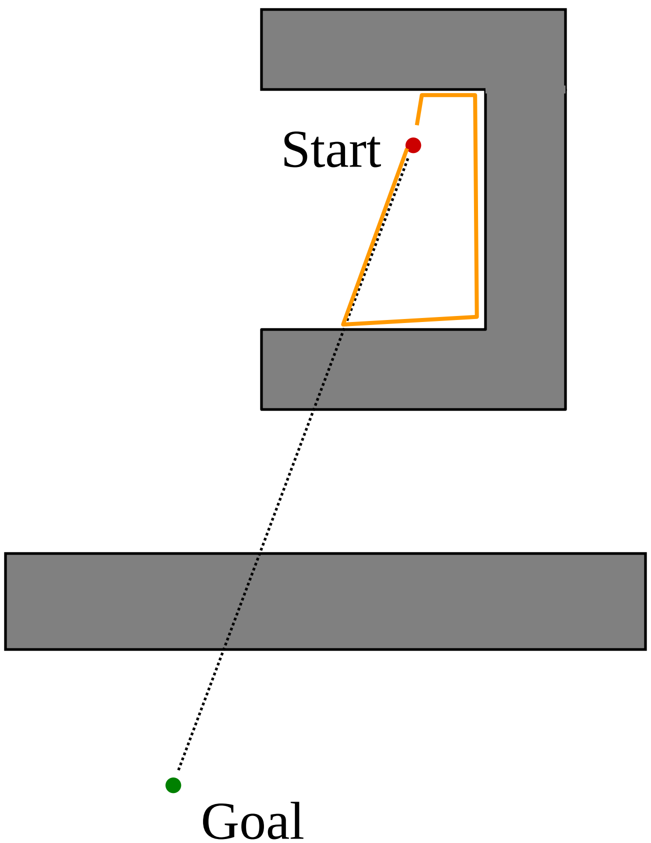
1) head toward goal on the *m-line*

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

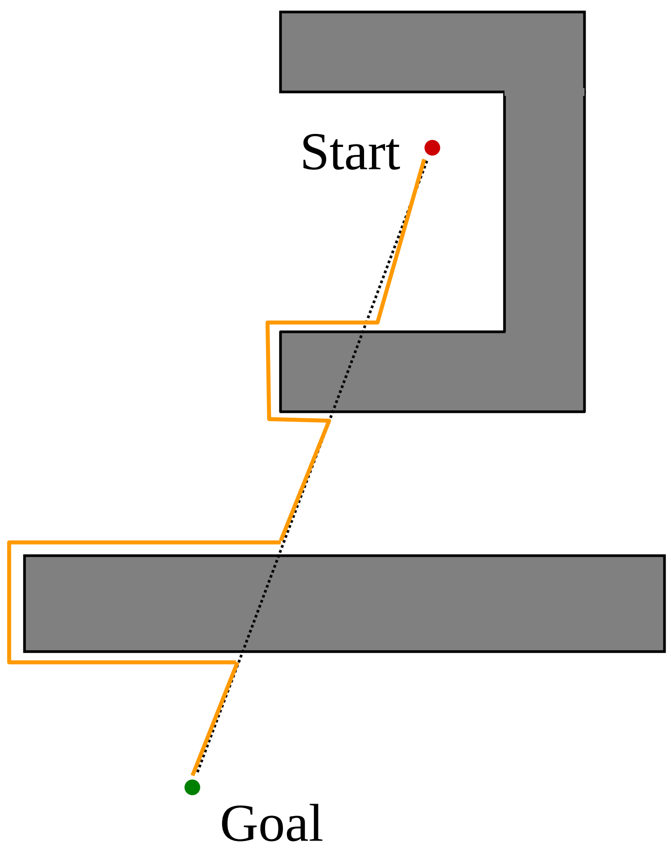
1) head toward goal on the *m-line*

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

3) Leave the obstacle and continue toward the goal

NO! How do we fix this?

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?

BUG 2 More formally

- Let $q^L_0 = q_{start;}$ i = 1
- repeat
  - repeat
    - from q^L_{i-1} move toward q_{goal} along the m-line
  - until goal is reached or obstacle encountered at q^H_i
  - if goal is reached, exit
  - repeat
    - follow boundary
  - until q_{goal} is reached or q^H_i is re-encountered or m-line is re-encountered, x is not q^H_i, $d(x,q_{goal}) < d(q^H_i,q_{goal})$ and way to goal is unimpeded
  - if goal is reached, exit
  - if q^H_i is reached, return failure
  - else
    - $q^L_i = m$
    - i=i+1
    - continue

head-to-head comparison

or thorax-to-thorax, perhaps

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

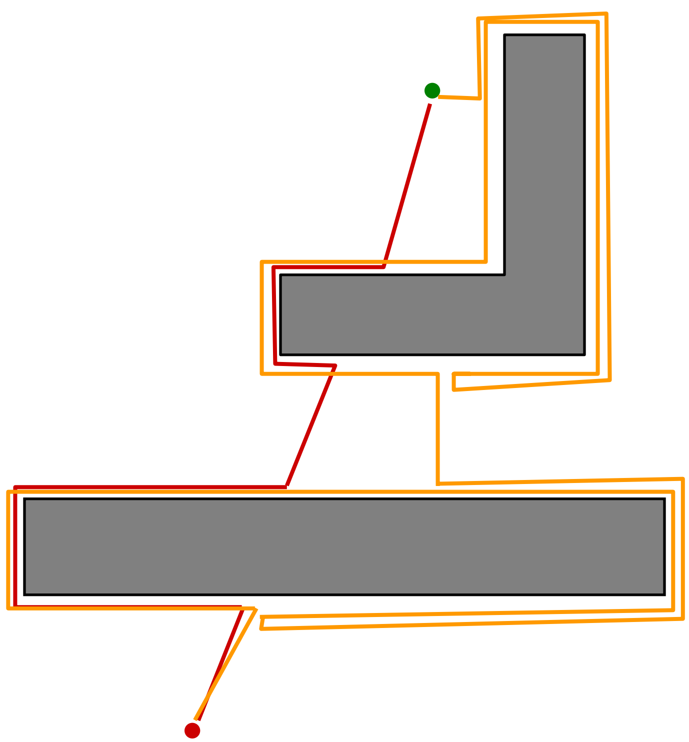
Bug 2 beats Bug 1	Bug 1 beats Bug 2

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

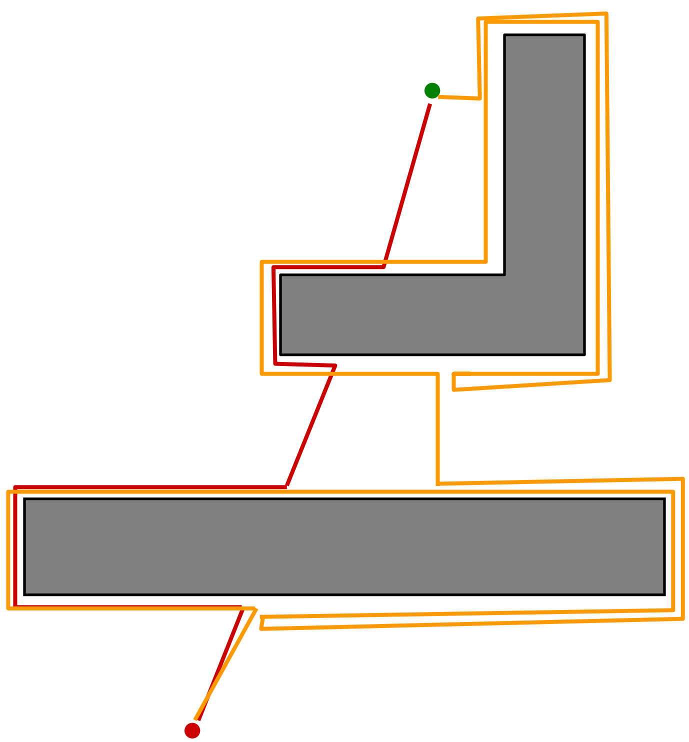
?

head-to-head comparison

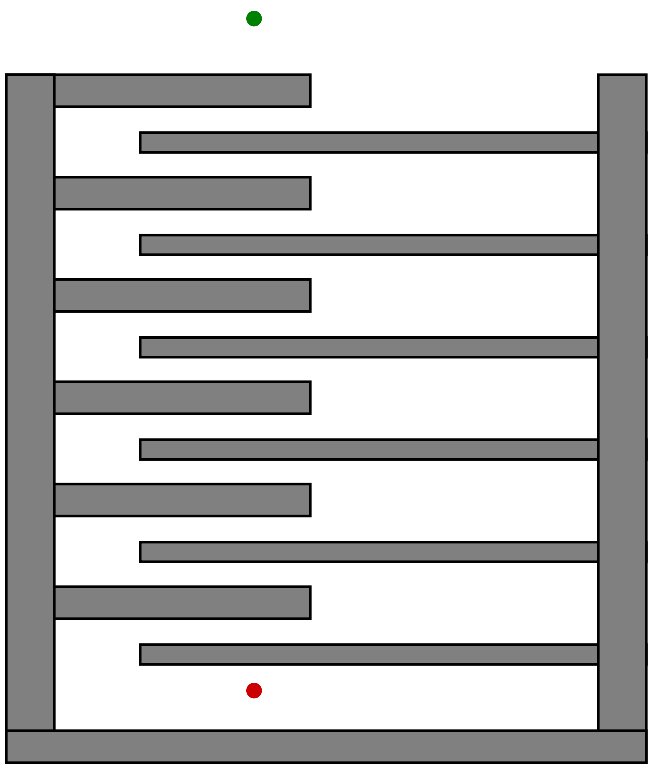
or thorax-to-thorax, perhaps

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

Bug 2 beats Bug 1



Bug 1 beats Bug 2



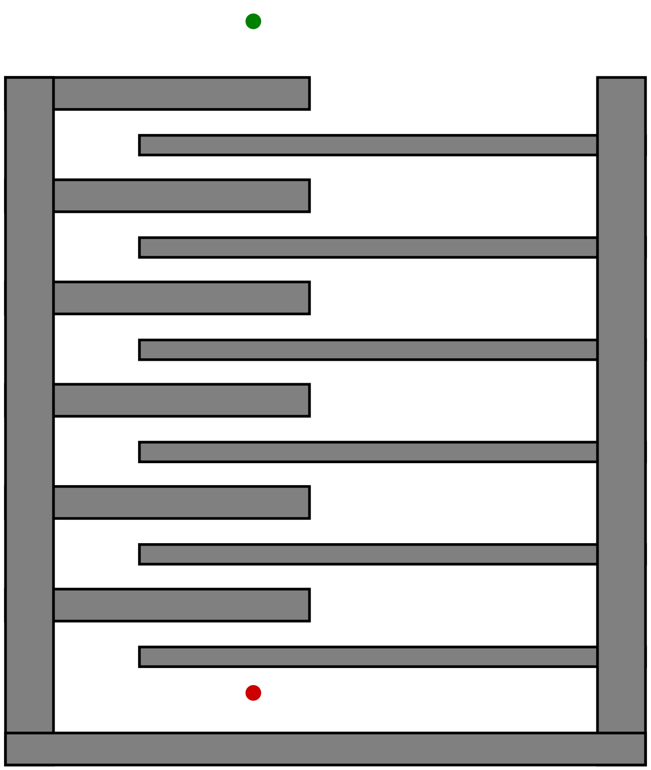
BUG 1 vs. BUG 2

- 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

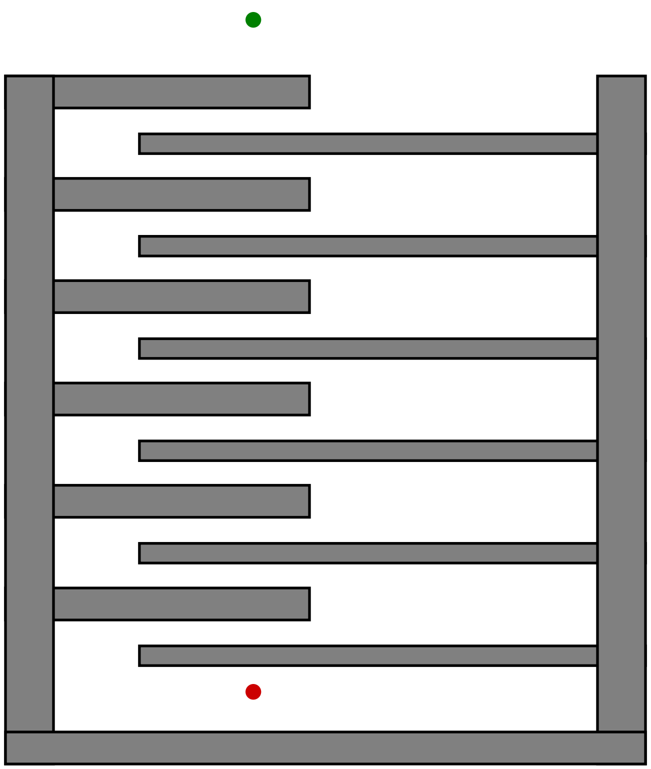
Upper bound:
What's the longest distance it might travel?

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:
What's the shortest distance it might travel?

$$D$$

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

$$D + \sum_i \frac{n_i}{2} P_i$$

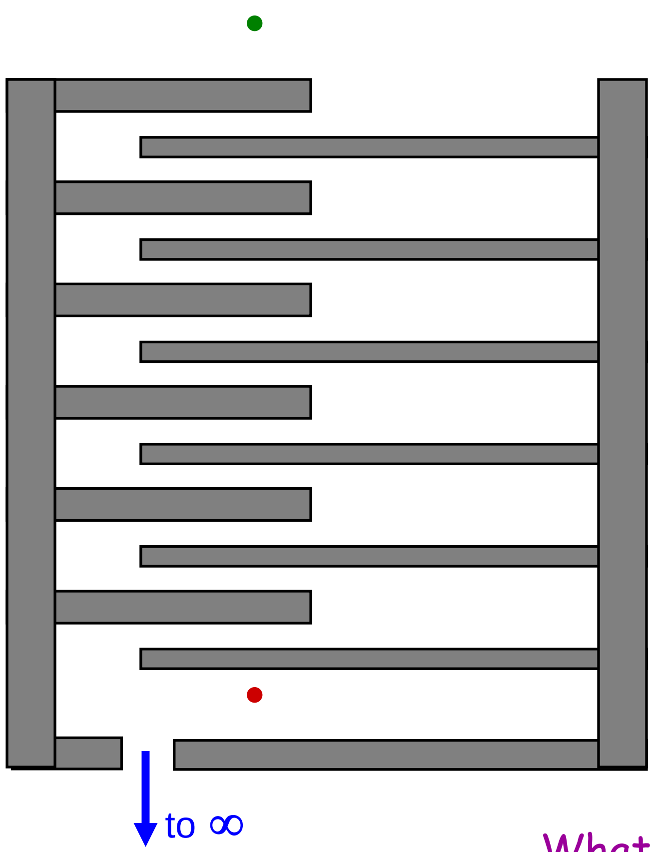
n_i = # of s-line intersections of the *i* th obstacle

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: What's the shortest distance it might travel? **D**

Upper bound: What's the longest distance it might travel? $D + \sum_i \frac{n_i}{2} P_i$

n_i = # of s-line intersections of the *i* th obstacle

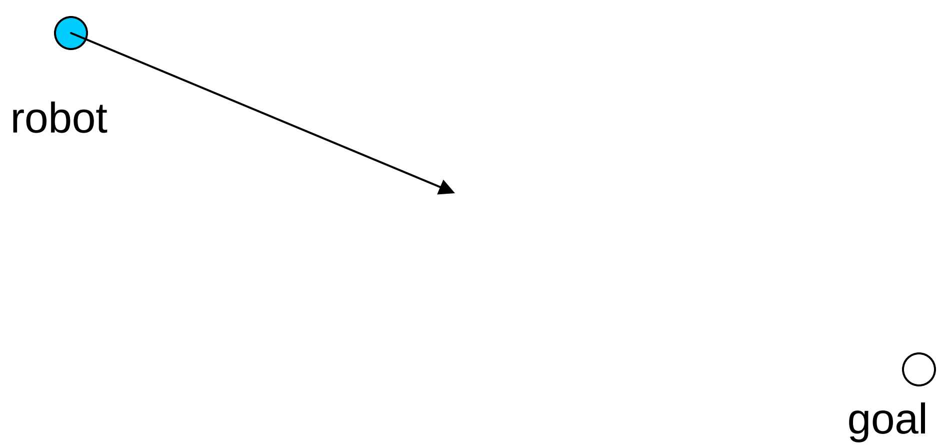
What is an environment where your upper bound is required?

A More Realistic Bug

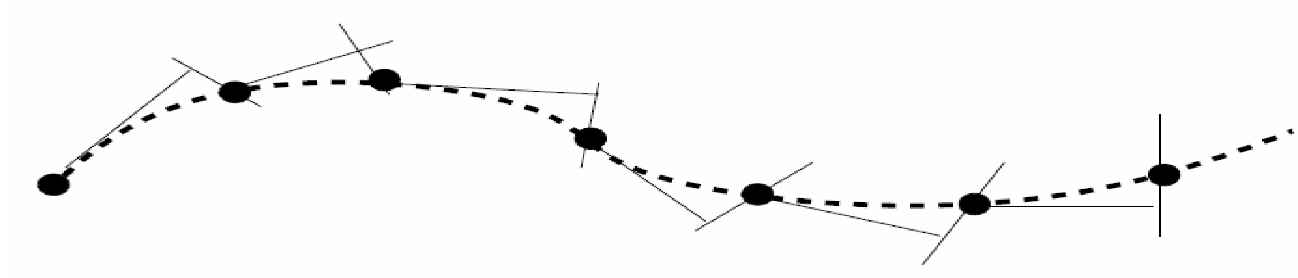
- As presented: global beacons plus contact-based wall following
- The reality: we typically use some sort of range sensing device that lets us look ahead (but has finite resolution and is noisy).
- Let us assume we have a range sensor
- distance fn: $\rho(x,\theta) = \min_{\lambda>=0} d(x, x+\lambda[c_\theta,s_\theta])$
  s.t. $x+\lambda[c_\theta,s_\theta]) \in \cup_i WO_i$
- Note we write $\rho: \Re(2) \times S(1) \rightarrow \Re$
  - what is S(1) ?
- Saturated distance: $\rho_R(x,\theta) = \rho(x,\theta)$ if $\rho(x,\theta) < R$, else ∞

Move to Goal

- Distance $d(a,b) = ((a_x - b_x)^2 + (a_y - b_y)^2)^{1/2}$
- Gradient descent of d(a,b), i.e., decrease distance to the goal



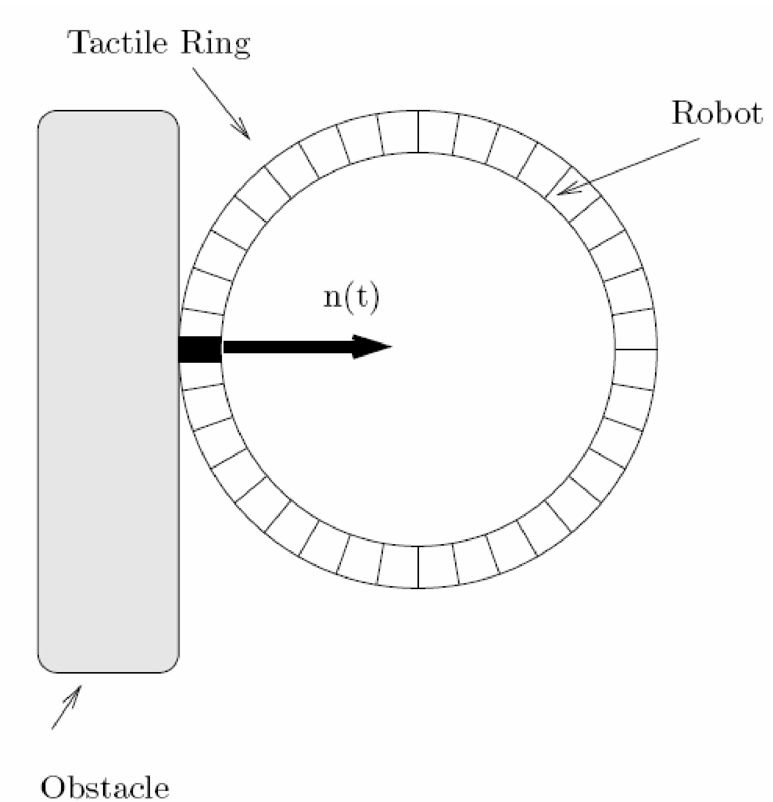
Circumnavigating Obstacles: Curve Tracing

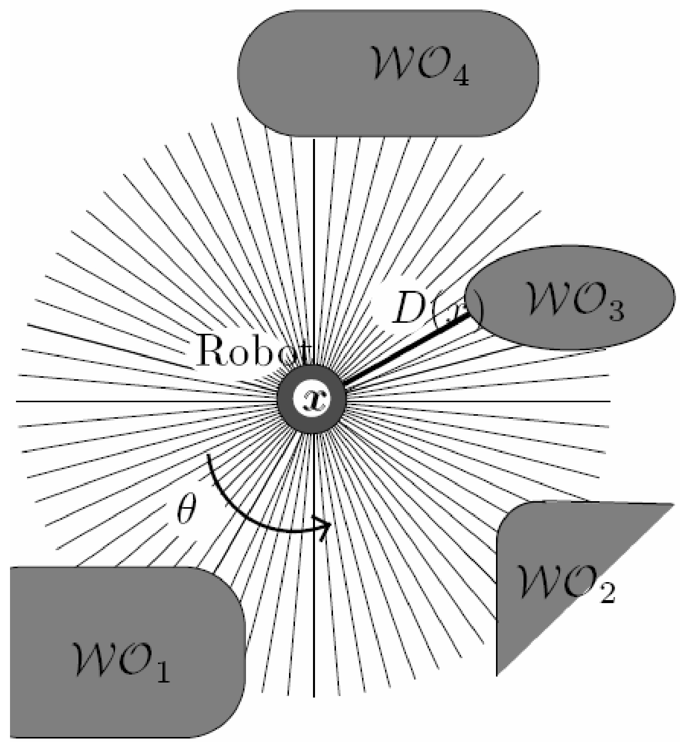


Predict: Tangent

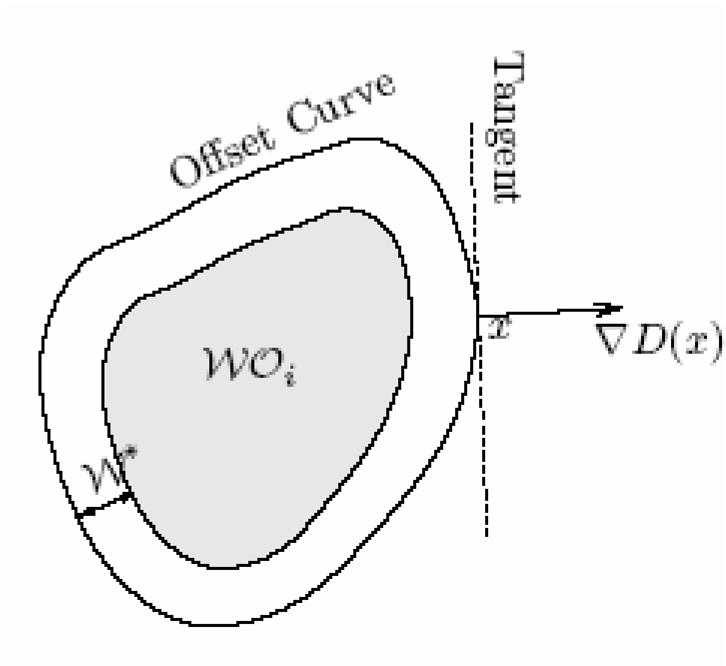
Correct: Something else

Normal (and hence Tangent) to Obstacle





Circumnavigate Obstacles: Boundary Following



Safety distance W*

D(x) = min d(x,c)

Normal is parallel to ∇D(x)

Increase/Decrease/Same

Tangent is orthogonal to both

$\dot{c}(t) = v \quad v$ is in $(n(c(t))^{\perp}$

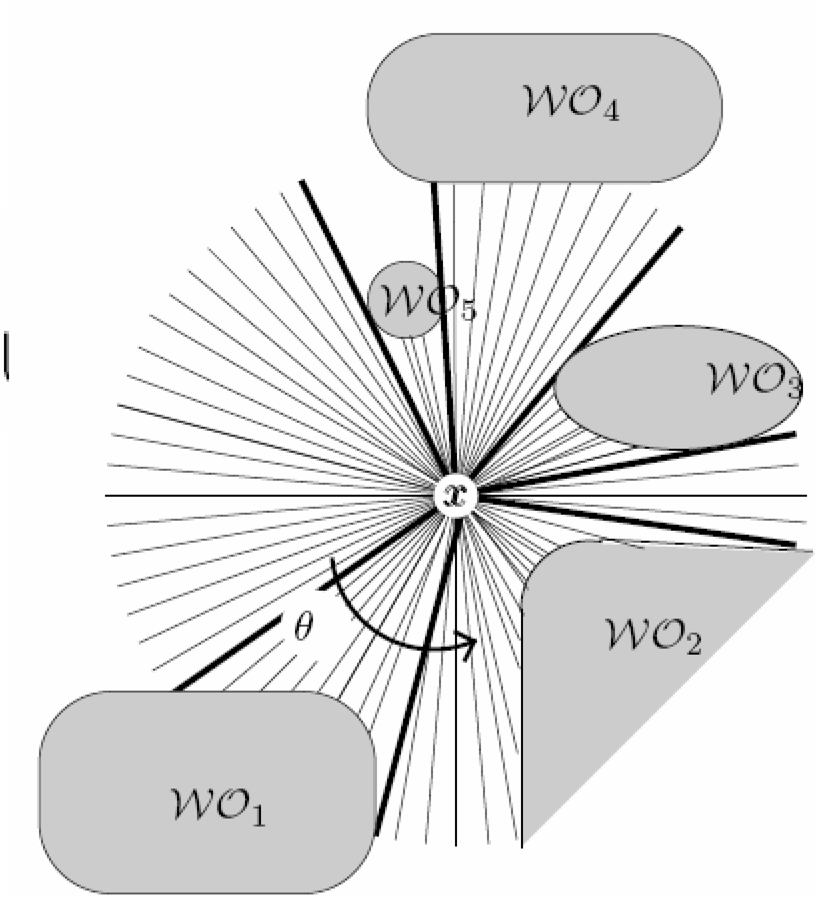
Raw Distance Function

$$\rho(x,\theta) = \min_{\lambda \in [0,\infty]} d(x, x + \lambda[\cos\theta, \sin\theta]^T),$$

$$\text{such that} \quad x + \lambda[\cos\theta, \sin\theta]^T \in \bigcup$$

Saturated raw distance function

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



Implicit Function Theorem

G(x) = D(x) – W*

Roots of G(x) trace the offset curve

DG(x) = DD(x), which is like a gradient in Euclidean spaces

Null of DG(x) is tangent, hence perp of DD(x) is too

THEOREM D.1.1 (Implicit Function Theorem) *Let $f : \mathbb{R}^m \times \mathbb{R}^n \rightarrow \mathbb{R}^n$ be a smooth vector-valued function, $f(x, y)$. Assume that $D_y f(x_0, y_0)$ is invertible for some $x_0 \in \mathbb{R}^m$, $y_0 \in \mathbb{R}^n$. Then there exist neighborhoods X_0 of x_0 and Z_0 of $f(x_0, y_0)$ and a unique, smooth map $g : X_0 \times Z_0 \rightarrow \mathbb{R}^n$ such that*

$$f(x, g(x, z)) = z$$

for all $x \in X_0$, $z \in Z_0$.

Correction

THEOREM D.2.1 (Newton-Raphson Convergence Theorem) *Let* $f : \mathbb{R}^n \to \mathbb{R}^n$ *and* $f(y^*) = 0$. *For some* $\rho > 0$, *let* f *satisfy*

- $Df(y^*)$ *is nonsingular with bounded inverse, i.e.,* $\|(Df(y^*))^{-1}\| \leq \beta$
- $\|Df(x) - Df(y)\| \leq \gamma \|x - y\|$ *for all* $x, y \in B_\rho(y^*)$, *where* $\gamma \leq \frac{2}{\rho\beta}$

Now consider the sequence $\{y^h\}$ *defined by*

$$y^{h+1} = y^h - (Df(y^h))^{-1} f(y^h),$$

for any $y^0 \in B_\rho(y^*)$. *Then* $y^h \in B_\rho(y^*)$ *for all* $h > 0$, *and the sequence* $\{y^h\}$ *quadratically converges onto* y^*, *i.e.,*

$$\|y^{h+1} - y^*\| \leq a \|y^h - y^*\|^2$$

where $a = \frac{\beta\gamma}{2(1-\rho\beta\gamma)} < \frac{1}{\rho}$.