Introduction to Autonomous and Intelligent Systems[1]

Obstacle Avoidance

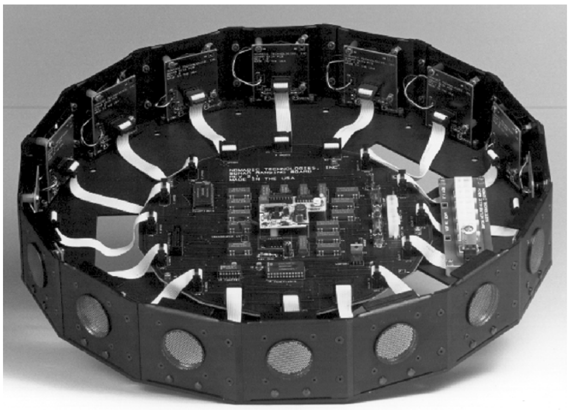
Justus Piater

Table of Contents

1. Range Sensors

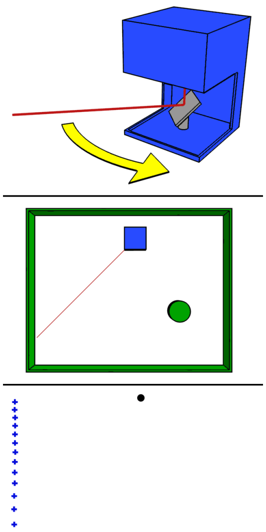
1.1. Sonar

- Time of flight
- Typically ultrasonic



1.2. Laser Range Finder

- Time of flight
- LIDAR (“Light detection and ranging”)

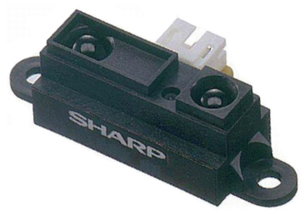


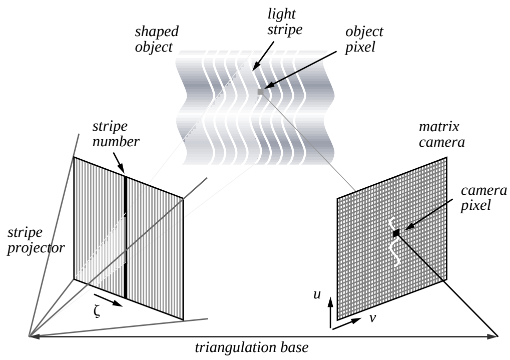
[1] https://iis.uibk.ac.at/courses/2019s/703030

1.3. ToF Cameras

1.4. Optical Triangulation

- 1D





- 2D: *structured light*

2. Obstacle Avoidance

2.1. Problem

Given:

- goal location in world coordinates
- robot pose in world coordinates
- range sensors

Sought:

- a (reactive) path to the goal in the presence of obstacles

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

16-735, Howie Choset with slides from G.D. Hager and Z. Dodds

A Few General Concepts

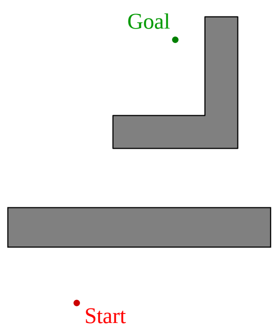
- Workspace W
  - $\Re(2)$ or $\Re(3)$ depending on the robot
  - could be infinite (open) or bounded (closed/compact)
- Obstacle WO_i
- Free workspace $W_{free} = W \setminus \cup_i WO_i$

16-735, Howie Choset with slides from G.D. Hager and Z. Dodds

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

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

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

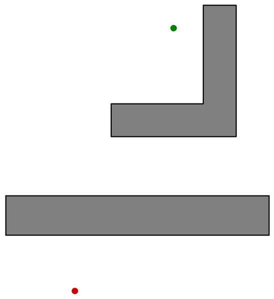
16-735, Howie Choset with slides from G.D. Hager and Z. Dodds

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}

16-735, Howie Choset with slides from G.D. Hager and Z. Dodds

how ?

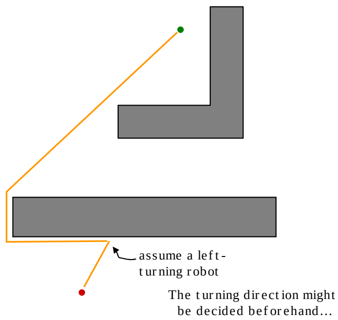
Buginner Strategy

"Bug 0" algorithm

1) head toward goal

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

3) continue



16-735, Howie Choset with slides from G.D. Hager and Z. Dodds

OK ?

Bug Zapper

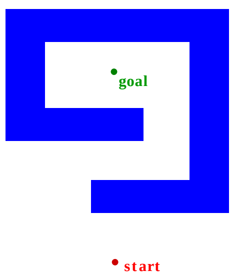
What map will foil Bug 0 ?

"Bug 0" algorithm

1) head toward goal

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

3) continue

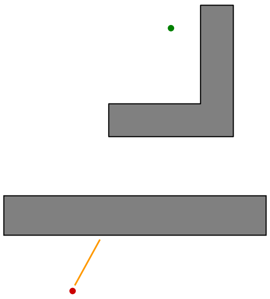


16-735, Howie Choset with slides from G.D. Hager and Z. Dodds

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

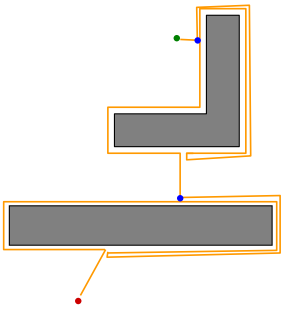
Vladimir Lumelsky & Alexander Stepanov: Algorithmica 1987

16-735, Howie Choset with slides from G.D. Hager and Z. Dodds

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

16-735, Howie Choset with slides from G.D. Hager and Z. Dodds

BUG 1 More formally

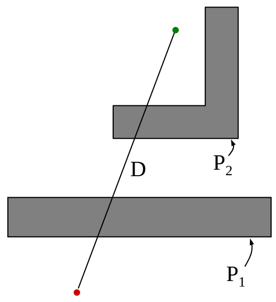
- Let $q^L_0 = q_{start}$; $i = 1$
- repeat
  - repeat
    - from q^L_{i-1} move toward q_{goal}
  - until goal is reached or obstacle encountered at q^H_i
  - if goal is reached, exit
  - repeat
    - follow boundary recording pt q^L_i with shortest distance to goal
  - until q_{goal} is reached or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

16-735, Howie Choset with slides from G.D. Hager and Z. Dodds

"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?

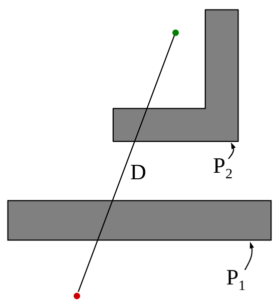
What is an environment where your upper bound is <u>required</u>?

16-735, Howie Choset with slides from G.D. Hager and Z. Dodds

"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?

16-735, Howie Choset with slides from G.D. Hager and Z. Dodds

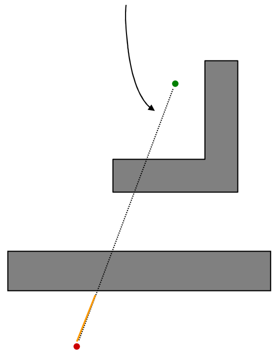
How Can We Show Completeness?

- 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, the 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.

16-735, Howie Choset with slides from G.D. Hager and Z. Dodds

A better bug?

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



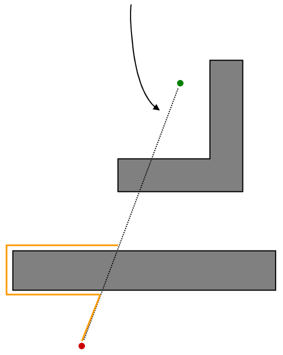
"Bug 2" Algorithm

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

16-735, Howie Choset with slides from G.D. Hager and Z. Dodds

A better bug?

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



"Bug 2" Algorithm

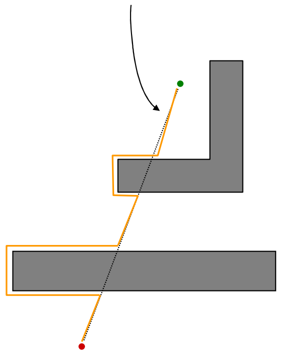
1) head toward goal on the *m-line*

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

16-735, Howie Choset with slides from G.D. Hager and Z. Dodds

A better bug?

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.

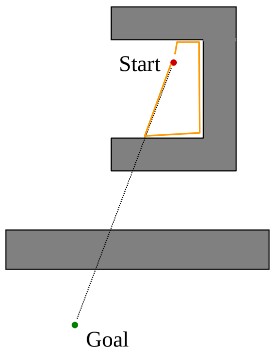
3) Leave the obstacle and continue toward the goal

16-735, Howie Choset with slides from G.D. Hager and Z. Dodds

OK ?

A better bug?

"Bug 2" Algorithm



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

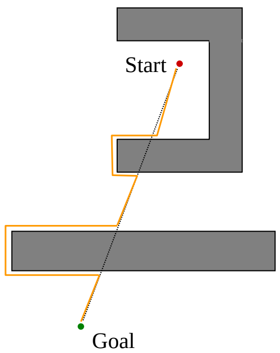
2) if an obstacle is in the way, follow it until you encounter the m-line again.

3) Leave the obstacle and continue toward the goal

16-735, Howie Choset with slides from G.D. Hager and Z. Dodds 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?

16-735, Howie Choset with slides from G.D. Hager and Z. Dodds

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

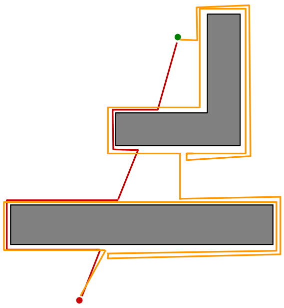
16-735, Howie Choset with slides from G.D. Hager and Z. Dodds

head-to-head comparison

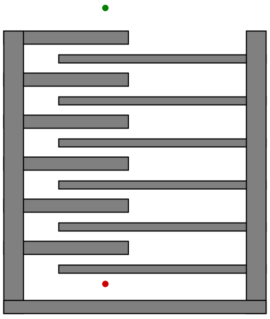
or thorax-to-thorax, perhaps

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

Bug 2 beats Bug 1



Bug 1 beats Bug 2



16-735, Howie Choset with slides from G.D. Hager and Z. Dodds

BUG 1 vs. BUG 2

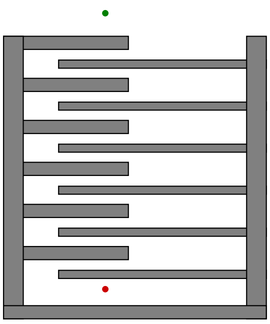
- BUG 1 is an *exhaustive search algorithm*
  - it looks at all choices before commiting
- BUG 2 is a *greedy* algorithm
  - it takes the first thing that looks better
- In many cases, BUG 2 will outperform BUG 1, but
- BUG 1 has a more predictable performance overall

16-735, Howie Choset with slides from G.D. Hager and Z. Dodds

"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?

$\mathbf{D}$

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

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

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

What is an environment where your upper bound is required?

16-735, Howie Choset with slides from G.D. Hager and Z. Dodds

To reach the upper bound, the robot needs to circumnavigate all obstacles (almost) completely for each hit point, minimally a sole obstacle once; see the looping obstacle.

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

16-735, Howie Choset with slides from G.D. Hager and Z. Dodds

2.2. Using A Range Sensor

- DistBug [Kamon and Rivlin 1997]

  Senses the distance F in freespace from the current location X to the nearest obstacle in the direction of the goal.

  Minor but effective extension of Bug2; allows the robot to leave the boundary early during wall following.

  Easy to implement.

- TangentBug [Kamon et al. 1996]

  Senses the distance to the nearest obstacle all around the robot.

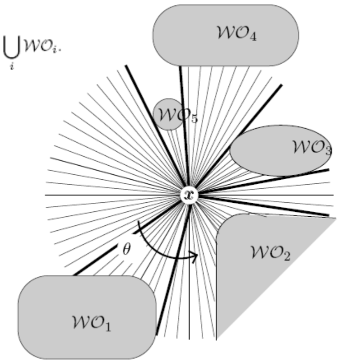
  State of the art; good approximations to globally optimal paths in many realistic scenarios.

  Slightly more involved to implement. (It is typically described in terms of the *local tangent graph*, but the following explanation succeeds without it.)

Both algorithms can handle zero, finite or infinite sensor ranges.

Raw Distance Function

$$\rho(x,\theta) = \min_{\lambda\in[0,\infty]} d(x, x+\lambda[\cos\theta,\sin\theta]^T),$$
$$\text{such that} \quad x+\lambda[\cos\theta,\sin\theta]^T \in \bigcup_i \mathcal{WO}_i.$$



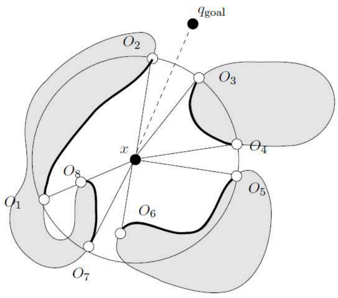
$$\rho: \mathbb{R}^2 \times S^1 \rightarrow \mathbb{R}$$

Saturated raw distance function

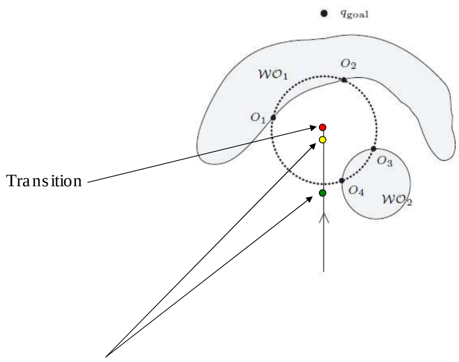
$$\rho_R(x,\theta) = \begin{cases} \rho(x,\theta), & \text{if } \rho(x,\theta) < R \\ \infty, & \text{otherwise.} \end{cases}$$

Intervals of Continuity

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



Motion-to-Goal Transition
from Moving Toward goal to "following obstacles"

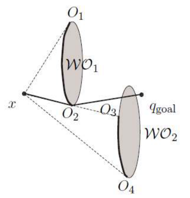


Currently, the motion-to-goal behavior "thinks" the robot can get to the goal

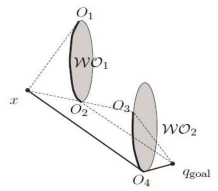
Now, it starts to see something --- what to do?
Ans: For any O_i such that $d(O_i,q_{goal}) < d(x,q_{goal})$,
choose the pt O_i that minimizes $d(x,O_i) + d(O_i,q_{goal})$

Minimize Heuristic Example

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



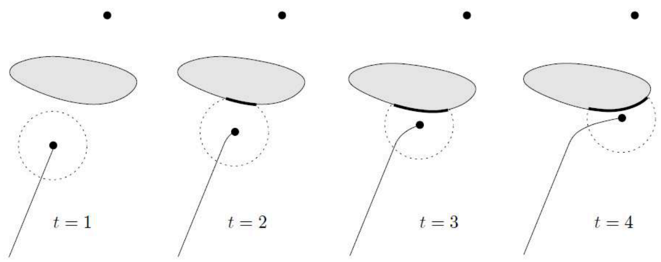
so moves toward O_2. Note the line connecting O_2 and goal pass through obstacle



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

For any O_i such that $d(O_i,q_{goal}) < d(x,q_{goal})$,
choose the pt O_i that minimizes $d(x,O_i) + d(O_i,q_{goal})$

Motion To Goal Example



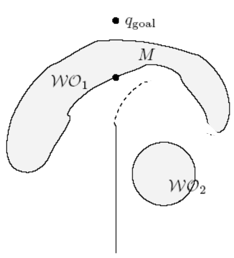
For any O_i such that $d(O_i,q_{goal}) < d(x,q_{goal})$,
choose the pt O_i that minimizes $d(x,O_i) + d(O_i,q_{goal})$

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 $(1-\lambda)x + \lambda q_{\mathrm{goal}} \quad \forall \lambda \in [0,1]$

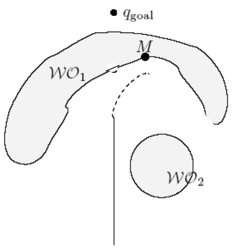
They start as the same

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 $(1-\lambda)x + \lambda q_{goal} \quad \forall \lambda \in [0,1]$

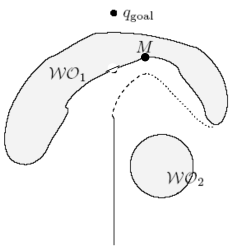
They start as the same

For any O_i such that $d(O_i,q_{goal}) < d(x,q_{goal})$,
choose the pt O_i that minimizes $d(x,O_i) + d(O_i,q_{goal})$

Boundary Following

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

Maintain d_{min} and d_{leave}



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

They start as the same

d_{min} and d_{leave}

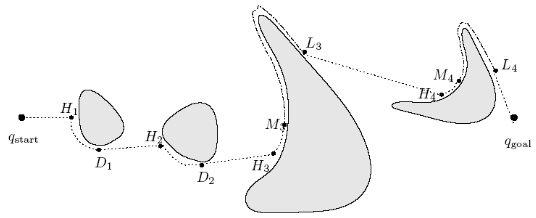
- d_{min} is the shortest distance, observed thus far, between the sensed boundary of the obstacle and the goal
- d_{leave} is the shortest distance between any point in the currently sensed environment and the goal

$$V_R(x) = \{y \in Q_{free} | d(x,y) < R \text{ and } \lambda x + (1-\lambda)y \in W_{free} \text{ for all } \lambda \in [0,1]\}$$

$$d_{leave}(x) = min_{y \in V(x)} d(x,y).$$

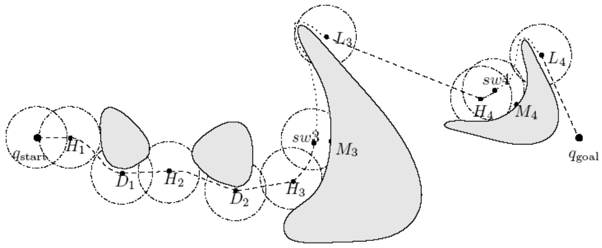
- Terminate boundary following behavior when $d_{leave} < d_{min}$
- Initialize with $x = q_{start}$ and $d_{leave} = d(q_{start}, q_{goal})$

Example: Zero Sensor Range

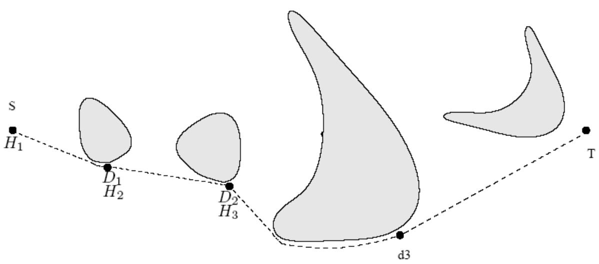


1. Robot moves toward goal until it hits obstacle 1 at H1
2. Pretend there is an infinitely small sensor range and the Oi which minimizes the heuristic is to the right
3. Keep following obstacle until robot can go toward obstacle again
4. Same situation with second obstacle
5. At third obstacle, the robot turned left until it could not increase heuristic
6. d_{min} is distance between M_3 and goal, d_{leave} is distance between robot and goal because sensing distance is zero

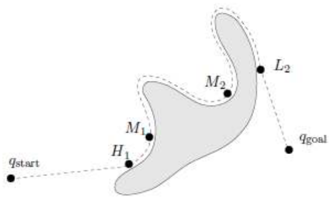
Example: Finite Sensor Range



Example: Infinite Sensor Range



d_{min} is constantly updated



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_{min} which is the shortest distance observed thus far between the sensed boundary of the obstacle and the goal
- A value d_{leave} which is the shortest distance between any point in the currently sensed environment and the goal
- Terminate boundary following behavior when $d_{leave} < d_{min}$

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_{leave} and d_{min}
   until
   a) goal is reached
   b) a complete cycle is performed (goal is unreachable)
   c) $d_{leave} < d_{min}$

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

Implementing Tangent Bug

- Basic problem: compute tangent to curve forming boundary of obstacle at any point, and drive the robot in that direction

- Let $D(x) = \min_c d(x,c) \quad c \in \cup_i WO_i$
- Let $G(x) = D(x) - W^*$ ← some safe following distance
- Note that $\nabla G(x)$ points radially away from the object
- Define $T(x) = (\nabla G(x))$ the tangent direction
  - in a real sensor (we'll talk about these) this is just the tangent to the array element with lowest reading

- We could just move in the direction T(x)
  - open-loop control

- Better is $\delta x = \mu\,(T(x) - \lambda\,(\nabla G(x))\,G(x))$
  - closed-loop control (predictor-corrector)

16-735, Howie Choset with slides from G.D. Hager and Z. Dodds

2.3. Remarks

- ***Contact sensors*** as a last resort: react if you bump into something.
- What if the obstacle moves?

3. References

3.1. References

I. Kamon, E. Rivlin, E. Rimon, "A new range-sensor based globally convergent navigation algorithm for mobile robots[1]". *IEEE International Conference on Robotics and Automation*, pp. 429–435, 1996.

I. Kamon, E. Rivlin, "Sensory-based motion planning with global proofs[2]". *IEEE Transactions on Robotics and Automation* 13(6), pp. 814–822, 1997.

V. Lumelsky, A. Stepanov, "Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape[3]". *Algorithmica* 2(4), pp. 403–430, 1987.

[1] http://www.cs.technion.ac.il/~ehudr/publications/pdf/KamonRR96i.pdf
[2] http://www.cs.technion.ac.il/~ehudr/publications/pdf/KamonR97a.pdf
[3] http://link.springer.com/content/pdf/10.1007%2FBF01840369.pdf