Robot Motion Control and Planning

http://www.ceng.metu.edu.tr/~saranli/courses/ceng786

Lecture 2 – Bug Algorithms

Uluç Saranlı

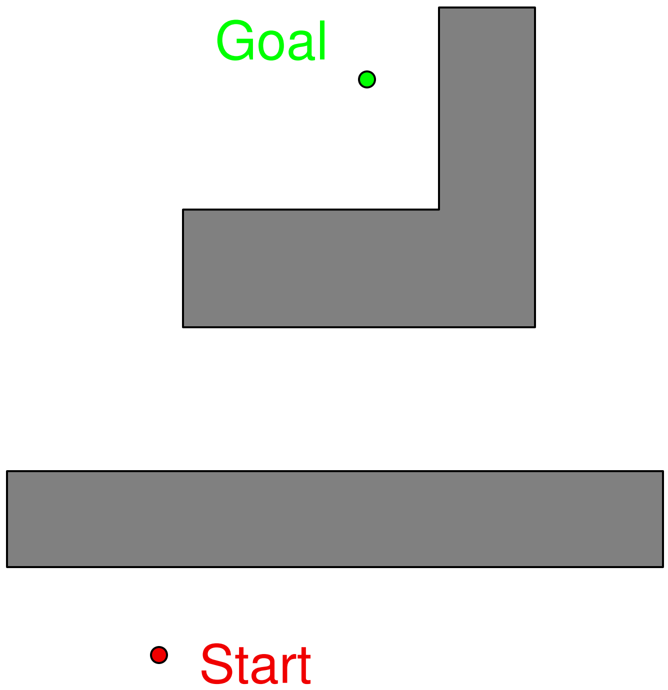
http://www.ceng.metu.edu.tr/~saranli

Bug Algorithms

- Point robot operating on the plane
  - Only local knowledge of the environment and a global goal
  - Bug1 and Bug2 assume tactile sensing
  - Tangent bug assumes finite distance sensing
- A few general concepts
  - Workspace W : $\mathbb{R}^2$ or $\mathbb{R}^3$depending on the robot. Could be infinite (open) or bounded (closed, compact)
  - Obstacles WO_i
  - Free workspace $W_{\mathtt{free}} := W - \bigcup_i WO_i$
- Representation: What is the current “situation”?

Bug Algorithms

Insect-inspired

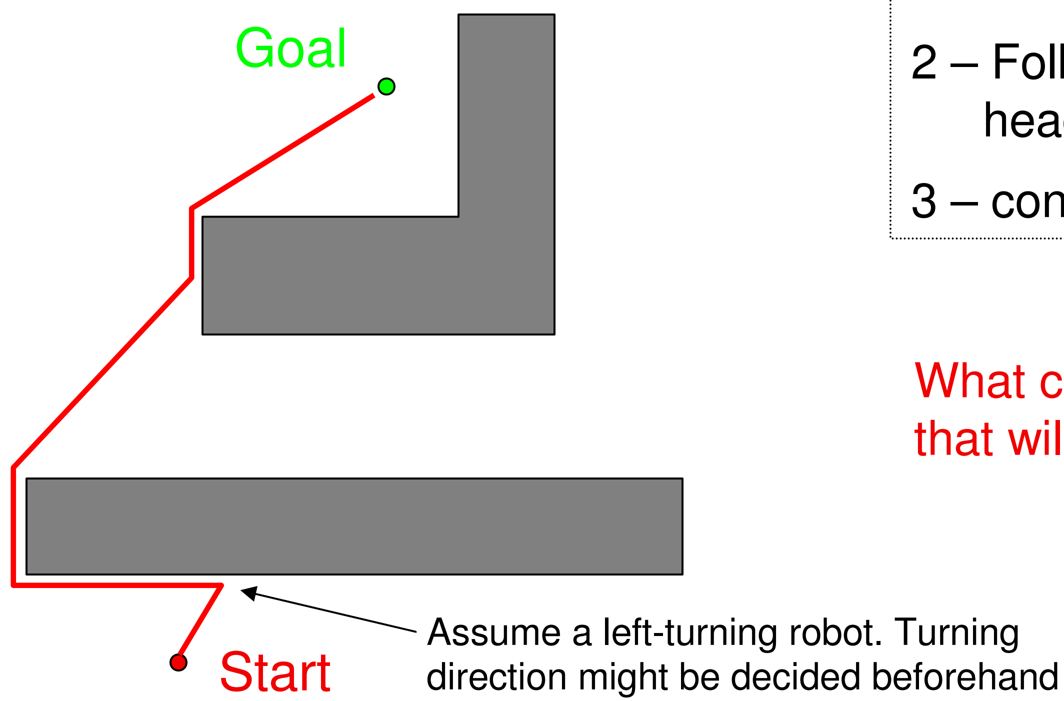


- **Known direction to goal**
  - robot can measure distance d(x,y) between pts x and y
- **Otherwise local sensing**
  - walls/obstacles and encoders
- **Reasonable world**
  - finitely many obstacles in any finite area
  - a line will intersect an obstacle finitely many times
  - Workspace is bounded

Beginner Strategy

"Bug 0" Algorithm

- **Known direction to goal**
- **Otherwise local sensing**
  - walls/obstacles and encoders



1 – Head toward goal

2 – Follow obstacles until you can head toward goal again

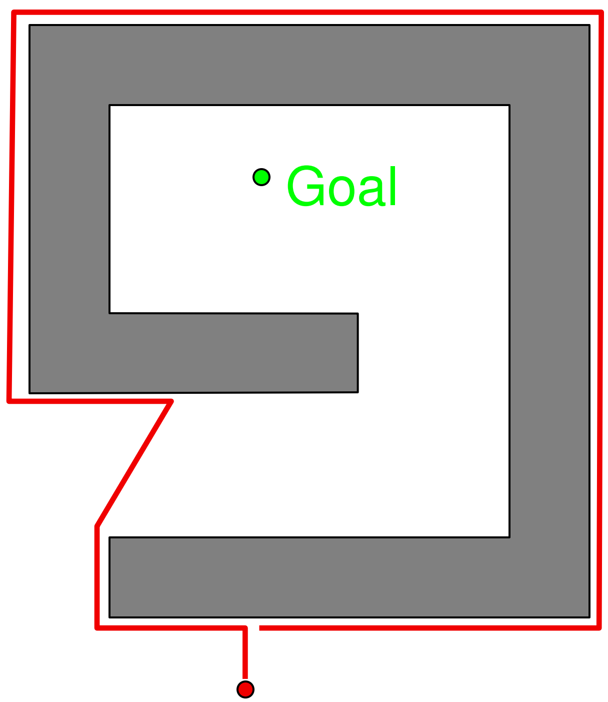
3 – continue

What can go wrong? Find a map that will foil Bug 0.

Bug Zapper

Bug 0 never reaches the goal!

Incomplete algorithm!



"Bug 0" Algorithm

1 – Head toward goal

2 – Follow obstacles until you can head toward goal again

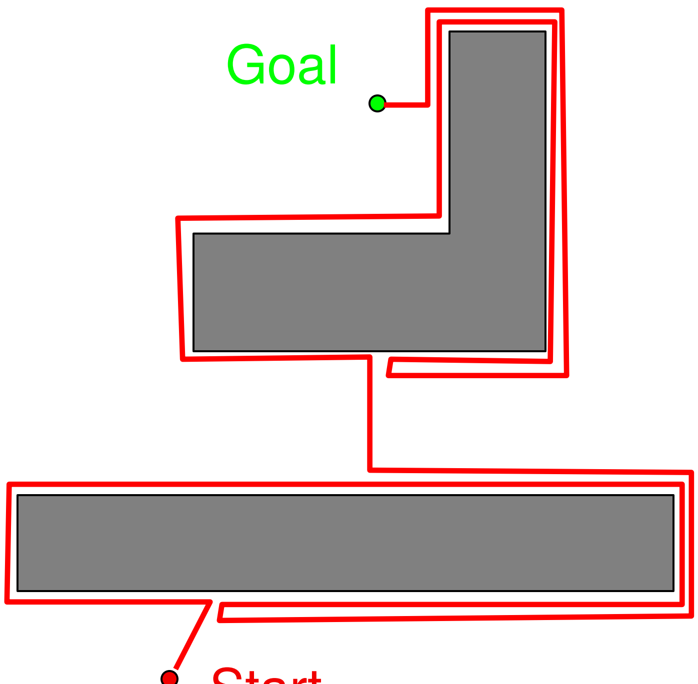
3 – continue

How can we improve Bug 0?

- Add memory
  - What information is available?
- Encoders
  - Keep track of robot's own motion

Bug 1

"Bug 1" Algorithm



- **Known direction to goal**
- **Otherwise local sensing**
  - walls/obstacles and encoders

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 and continue

- Takes longer to run

- Requires more computational effort

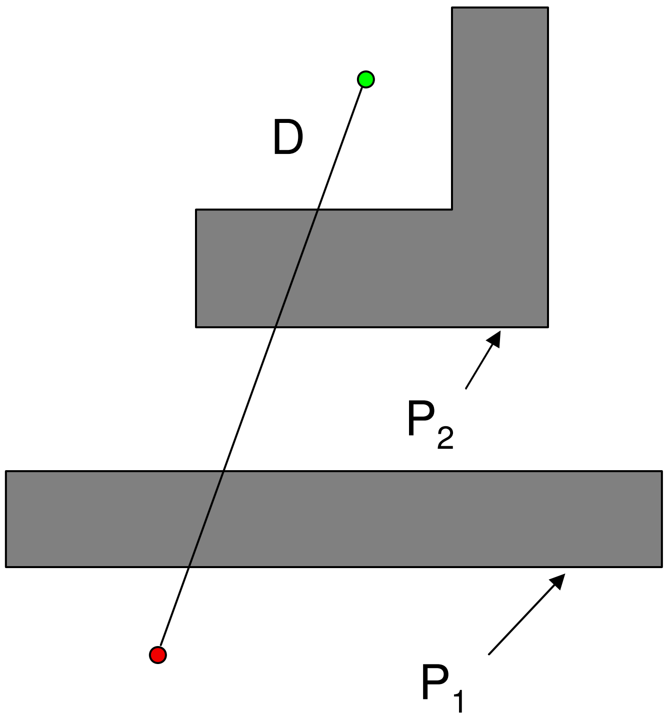
Vladimir Lumelsky & Alexander Stepanov: Algorithmica 1987

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 is the shortest distance it might travel?

$$D$$

Upper bound
what is the longest distance it might travel?

$$D + 1.5 \sum_i P_i$$

What is an environment where the upper bound is required?

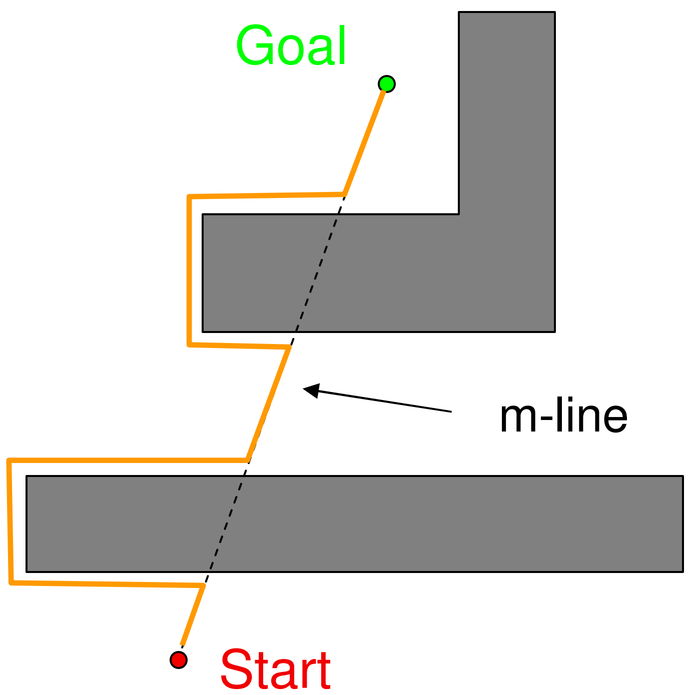
How Can We Show Completeness?

- 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.

A Better Bug?

"Bug 2" Algorithm



1 – Head toward goal on the m-line

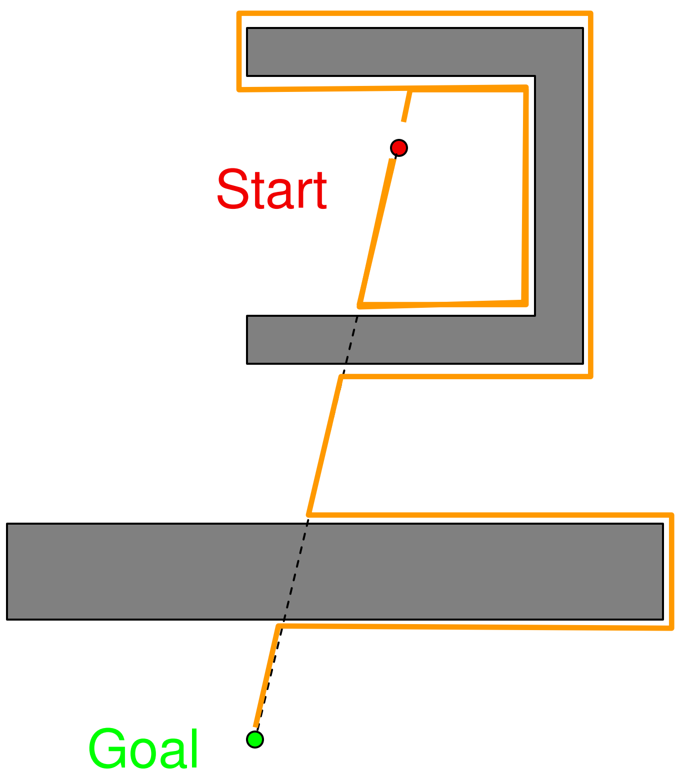
2 – If an obstacle is on the way, follow it until you hit the m-line again

3 – Leave the obstacle and continue toward the goal

What can go wrong? Find maps that will foil Bug 2.

A Better Bug?

Whoops! Infinite loop



1 – Head toward goal on the m-line

2 – If an obstacle is on the way, follow it until you hit the m-line again closer to the goal

3 – Leave the obstacle and continue toward the goal

Is this algorithm better or worse than Bug 1?

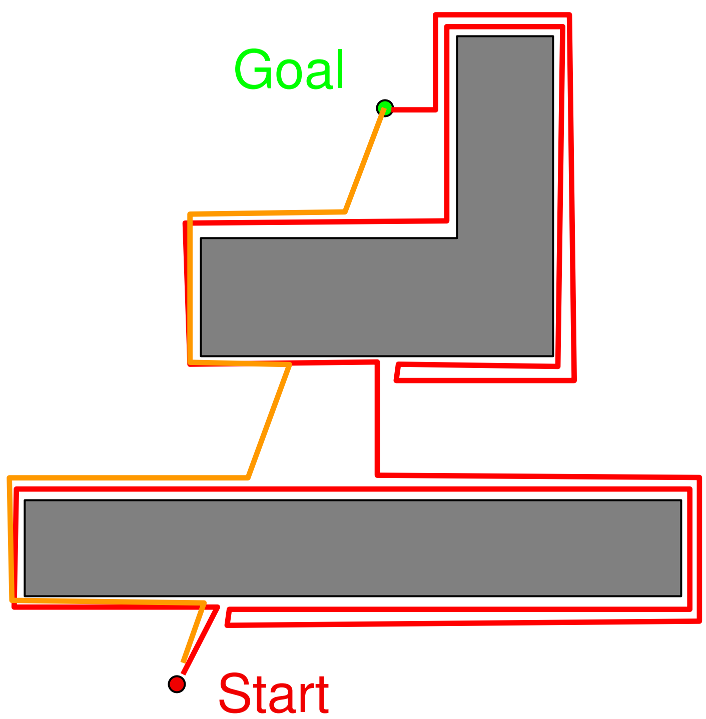
Bug 2 More formally

- 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

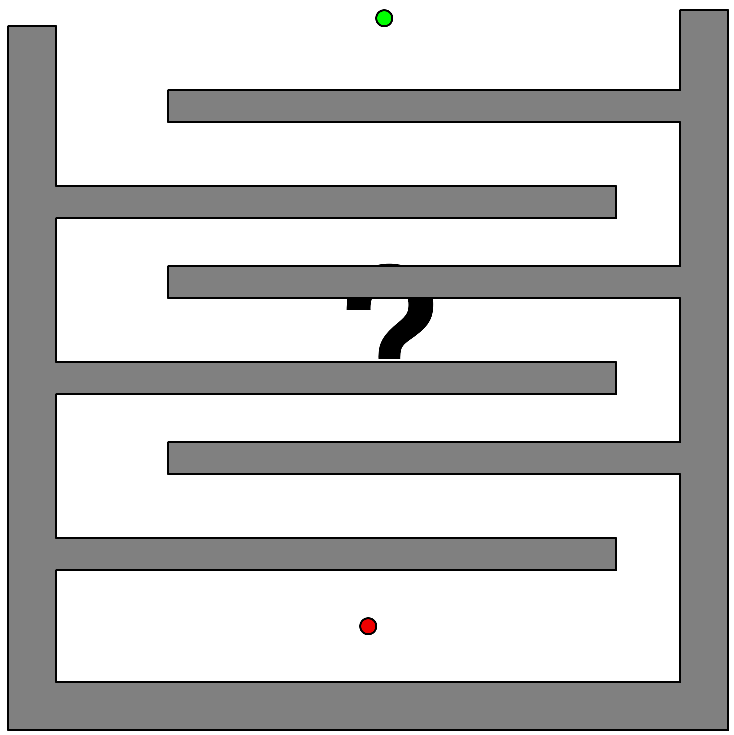
Head-to-head Comparison

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

Bug 2 beats Bug 1



Bug 1 beats Bug 2



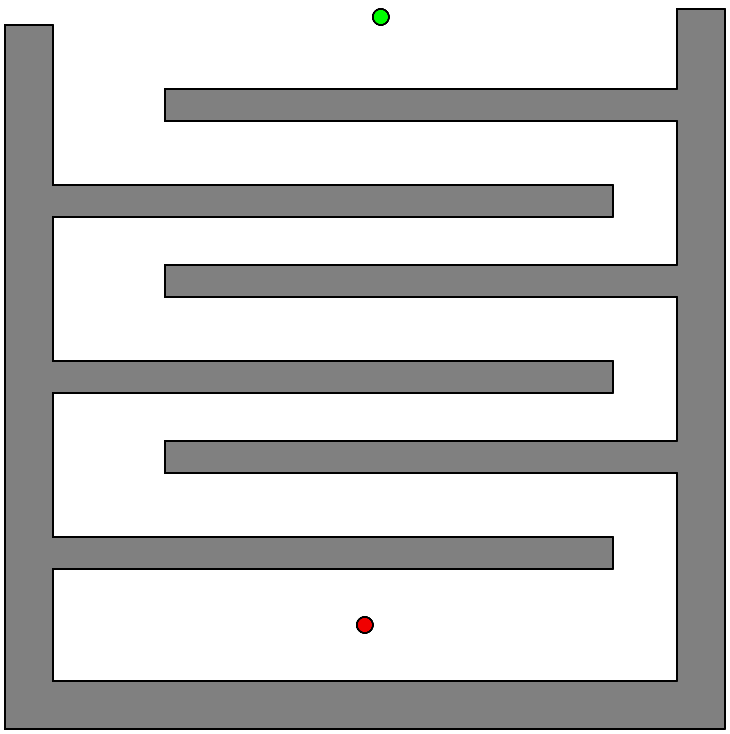
Bug 1 vs. Bug 2

- Bug 1 is an *exhaustive search algorithm*
  - it looks at all choices before committing
- 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

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 is the shortest distance it might travel?

$$D$$

Upper bound
what is the longest distance it might travel?

$$D + 1.5 \sum_i \frac{n_i}{2} P_i$$

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

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

A more realistic bug

- As presented: global beacons plus contact-based wall following

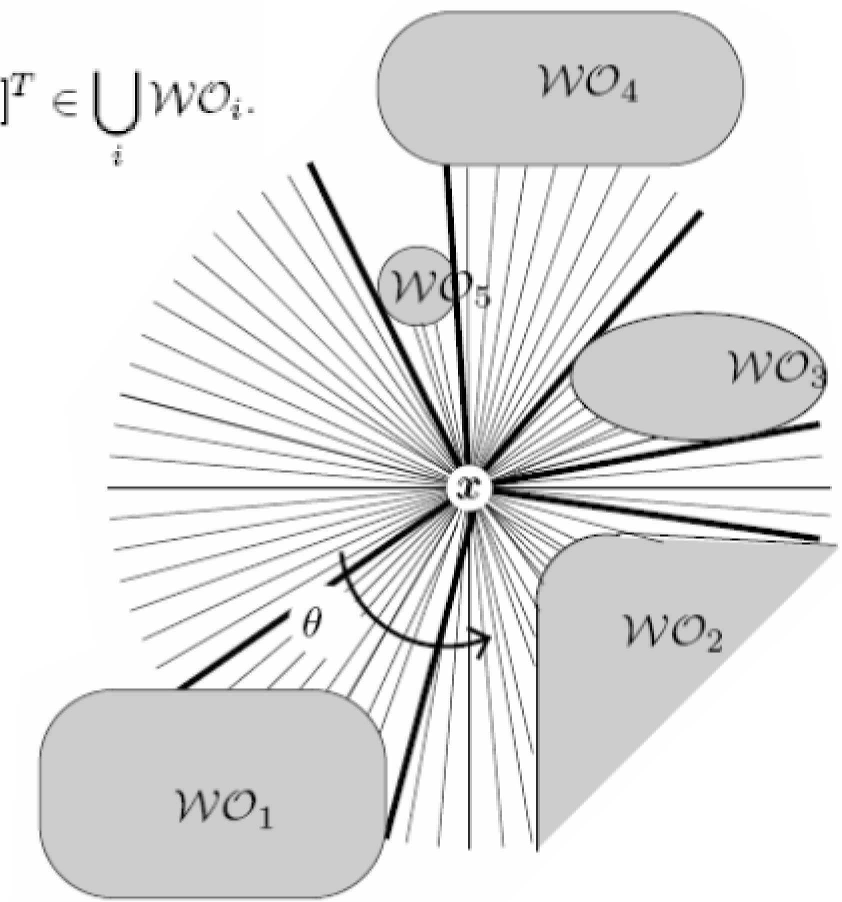
- The reality: we typically use some sort of range sensing device that lets us look ahead (but has finite resolution and is noisy)

- Now, let us assume we have a range sensor...

Raw Distance Function

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

$$\text{such that} \quad x + \lambda[\cos\theta, \sin\theta]^T \in \bigcup_i \mathcal{WO}_i.$$



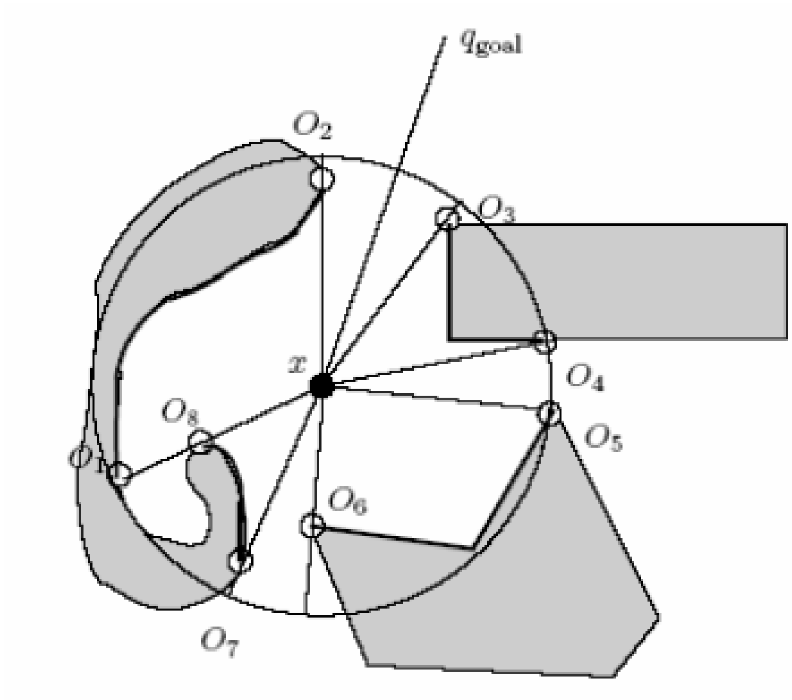
$$\rho : \mathbb{R}^2 \times S^1 \rightarrow \mathbb{R}$$

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 O_i of finite, continuous segments of ρ_R

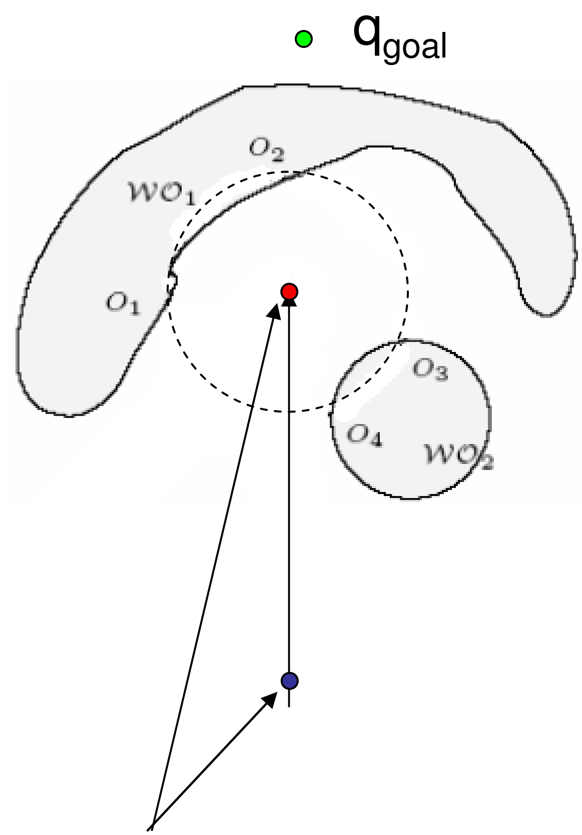


Tangent Bug Algorithm: Basic Ideas

- Motion-to-Goal (two variations)
  - Move towards the goal until an obstacle is sensed between the robot and the goal
  - Move towards the O_i that maximally decreases a heuristic distance, e.g. $d(x, O_i) + d(O_i, q_{goal})$

- Follow obstacle
  - Started if the robot cannot decrease the heuristic distance
  - Continuously moves towards the on the followed obstacle in the same direction as the previous motion-to-goal
  - Back to motion-to-goal when it is “better” to do so

Transition Among Motion-To-Goal



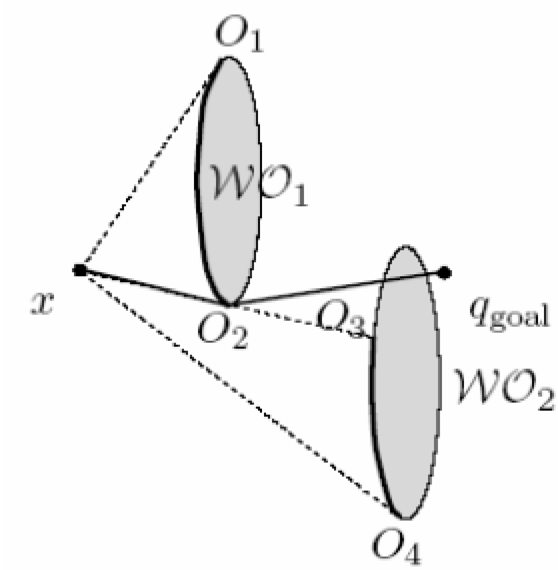
Currently, the motion-to-goal behavior "thinks" the robot can get to the goal

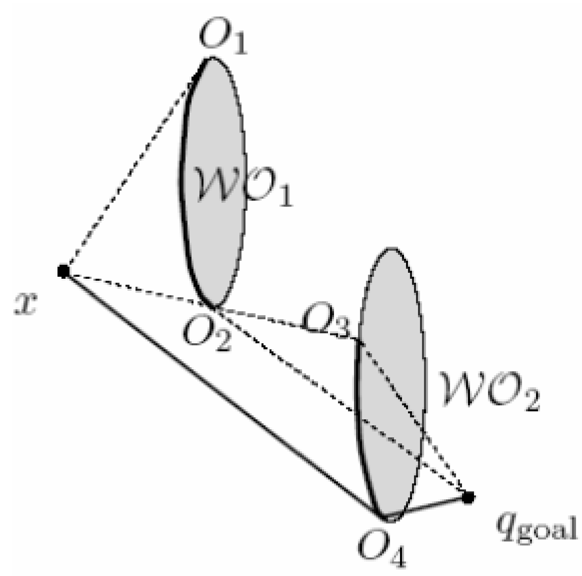
Now, it starts to "see" something on the way to the goal – what to do?

Ans: Start moving towards the O_i that minimizes $d(x, O_i) + d(O_i, q_{goal})$

Heuristic Example

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



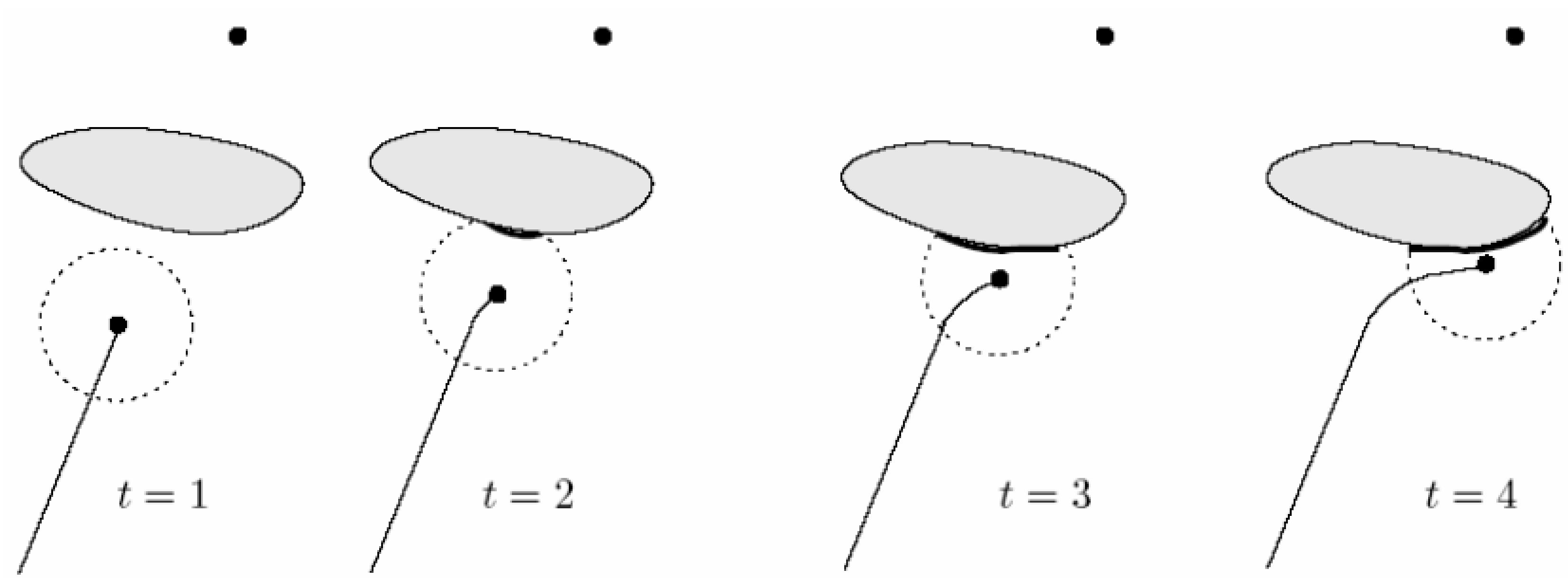


so it moves toward O_2. Note that the line connecting O_2 and goal passes through an obstacle

so it moves toward O_4. Note that some "thinking" was involved and the line connecting O_4 and the goal passes through an obstacle

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

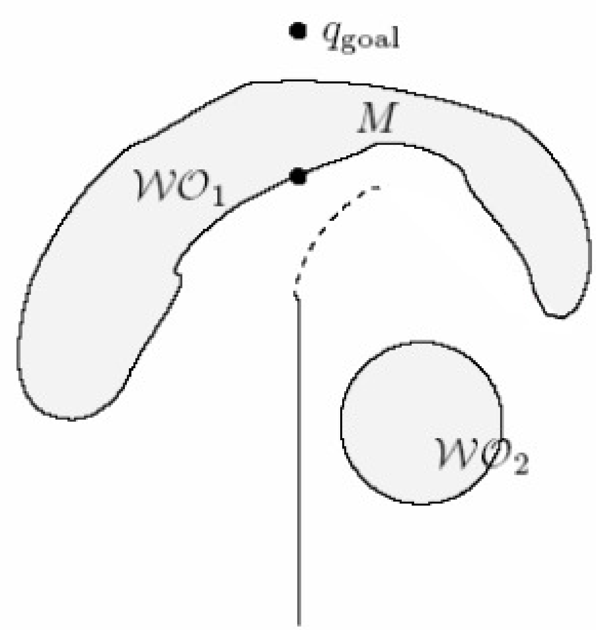
Motion-to-Goal Example



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

Transition From Motion-To-Goal

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

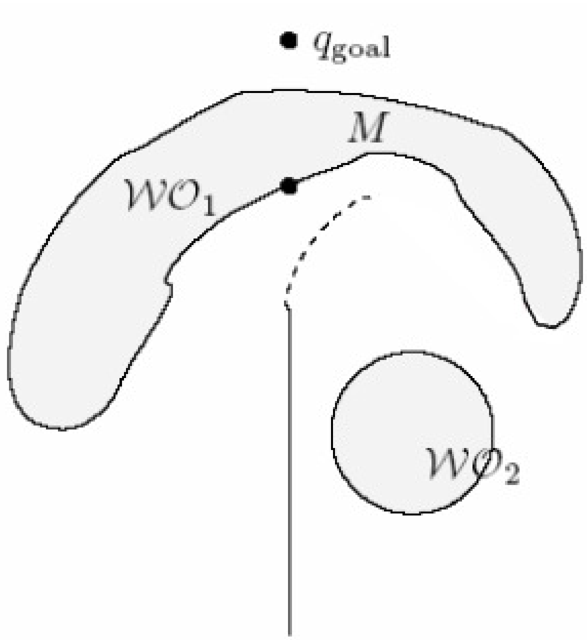


- Problem: What if this distance starts to go up?
- Answer: Start to act like a Bug and follow boundary!

- Some definitions:
  - M is the point on the “sensed” obstacle which has the shortest distance to the goal
  - Followed obstacle: the obstacle that we are currently sensing
  - Blocking obstacle: obstacle intersecting segment $(1-\lambda)x + \lambda q_{\text{goal}} \quad \forall \lambda \in [0, 1]$
  - The last two start as the same

Boundary following

- Move toward the Oi on the followed obstacle in the "chosen" direction while maintaining $d_{followed}$ and d_{reach}



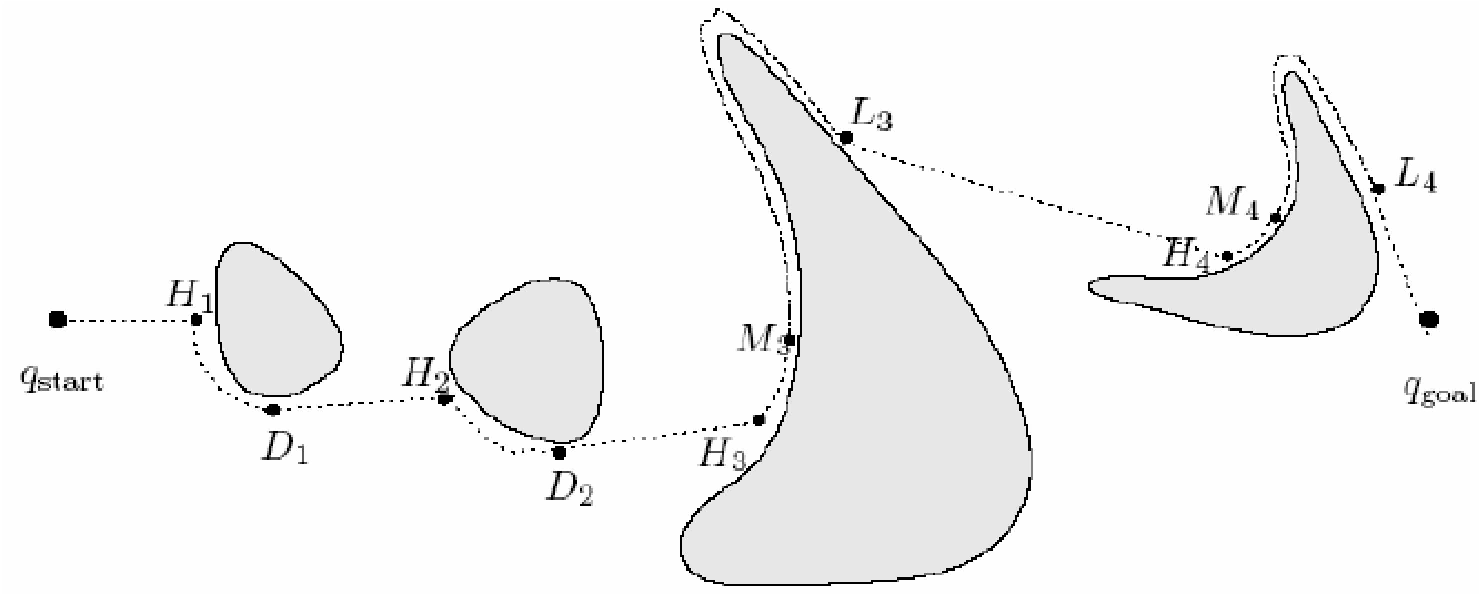
- $d_{followed}$ is the shortest distance between the *sensed* boundary and the goal
- d_{reach} is the shortest distance between *blocking* obstacle and goal (or my distance to goal if no blocking obstacle visible)

$$\Lambda := \{y \in \delta WO_b : \lambda x + (1-\lambda)y \in Q_{\text{free}} \quad \forall \lambda \in [0,1]$$
$$d_{\text{reach}} := \min_{c \in \Lambda} d(q_{\text{goal}}, c)$$

- Terminate when $d_{reach} < d_{followed}$

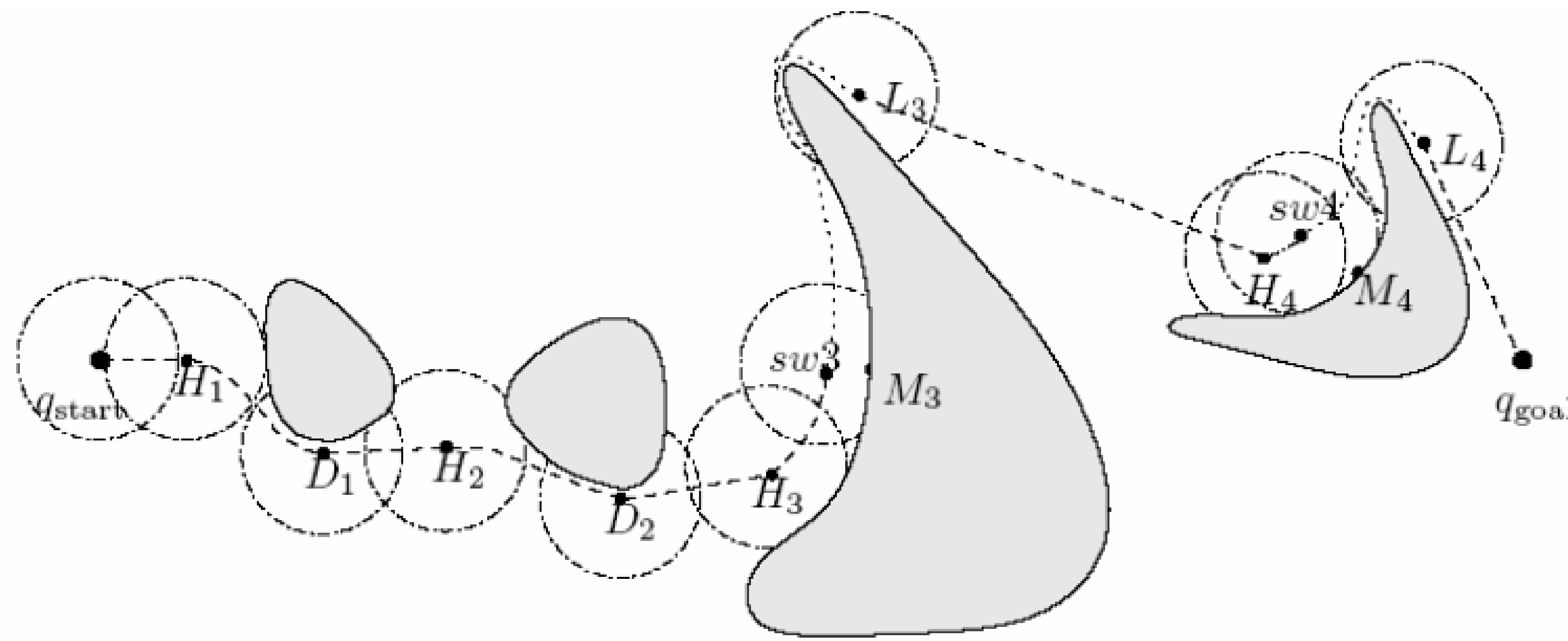
- Some definitions:
  - M is the point on the "sensed" obstacle which has the shortest distance to the goal
  - Followed obstacle: the obstacle that we are currently sensing
  - Blocking obstacle: obstacle intersecting segment $(1-\lambda)x + \lambda q_{\text{goal}} \quad \forall \lambda \in [0,1]$
  - The last two start as the same

Example: Zero Sensor Range

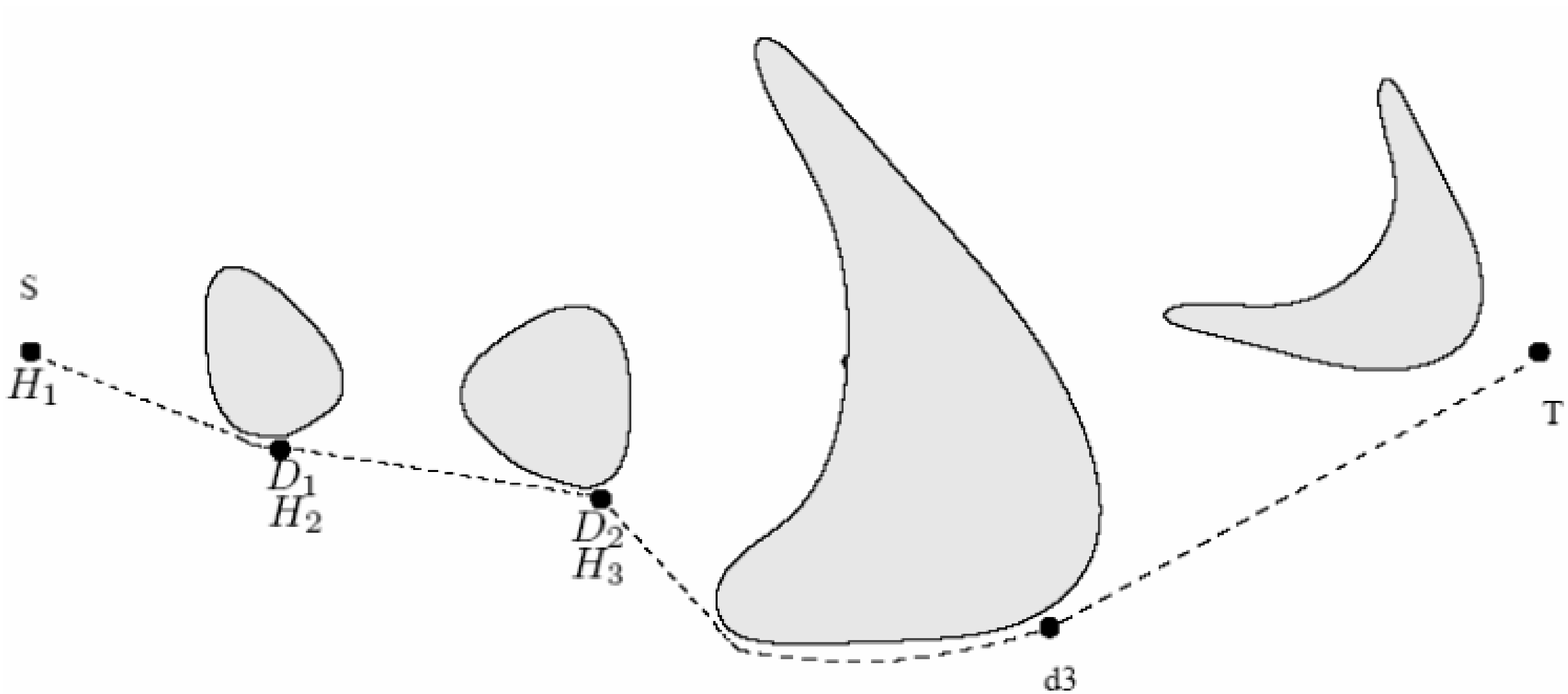


- Robot moves toward goal until it hits obstacle 1 at H_1
- Pretend there is an infinitely small sensor range and the O_i which minimizes the heuristic is to the right
- Keep following obstacle until robot can go toward obstacle again
- Same situation with second obstacle
- At third obstacle, the robot turned left until it could not increase heuristic
- $D_{followed}$ 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 Algorithm

- repeat
  - Compute continuous range segments in view
  - Move toward n in $\{T, O_i\}$ that minimizes $h(x,n) = d(x,n) + d(n,q_{goal})$ until
    - goal is encountered, or
    - the value of $h(x,n)$ begins to increase
- follow boundary continuing in same direction as before repeating
  - update $\{O_i\}$, d_{reach} and $d_{followed}$ until
    - goal is reached
    - a complete cycle is performed (goal is unreachable)
    - $d_{reach} < d_{followed}$

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

Implementing tangent Bug

- Basic problem: compute tangent to curve forming boundary of obstacle at any point, and drive the robot in that direction
- Let $D(x) = \min_c d(x,c) \quad c \in \bigcup_i WO_i$
- Let $G(x) = D(x) - W^*$ ← some safe following distance
- Note that $\nabla G(x)$ points radially away from the object
- Define $T(x) := (\nabla G(x))^\perp$ 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)

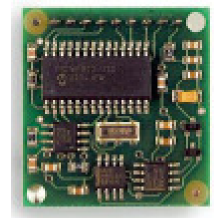
Sensors

Robots’ link to the external world...

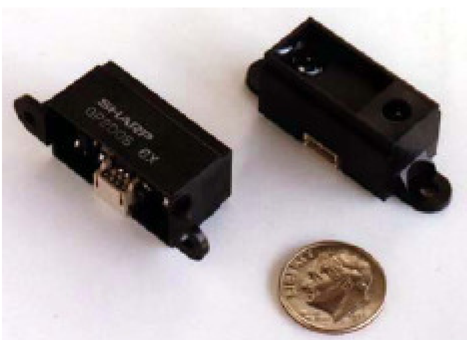
sonar rangefinder

gyro

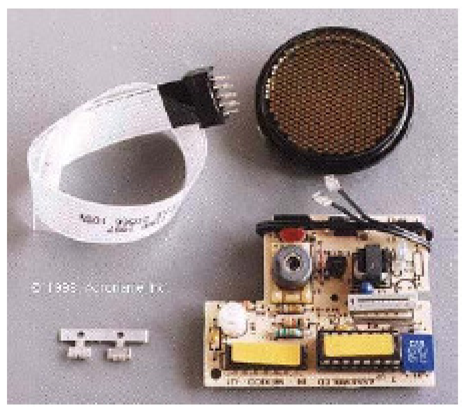
Sensors, sensors, sensors!
and tracking what is sensed: world models



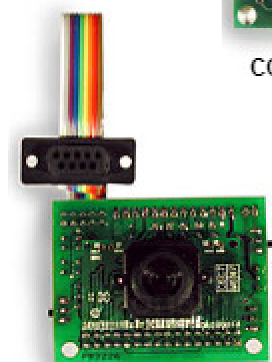
compass



IR rangefinder



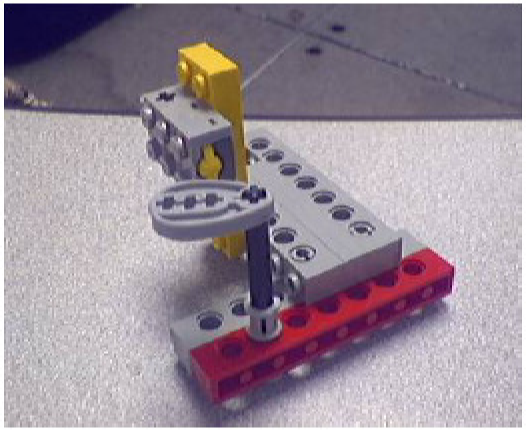
sonar rangefinder



CMU cam with on-board processing

odometry…

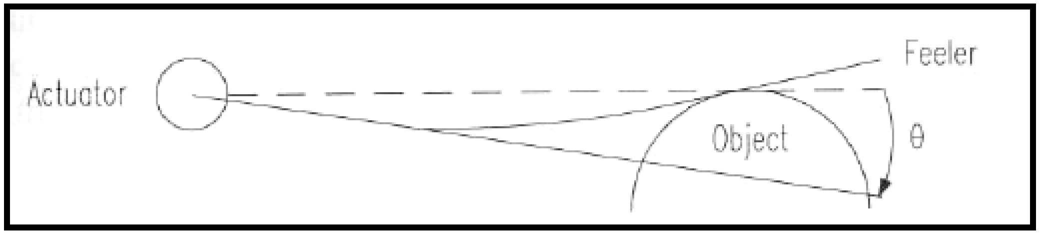
Tactile Sensors

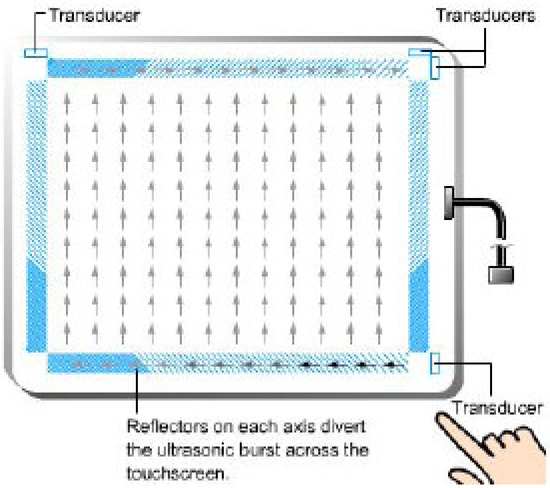


on/off switch

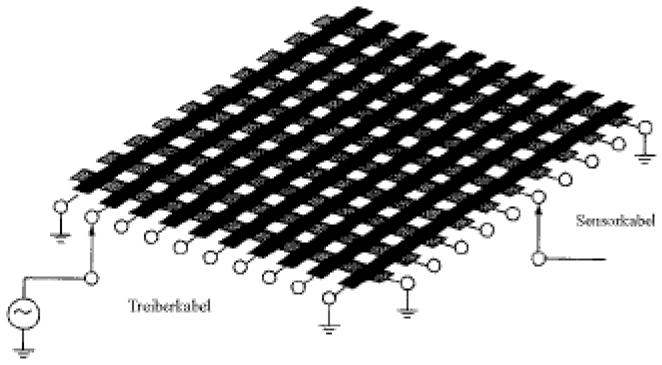
as a low resolution encoder

analog input: “Active antenna”





Surface acoustic waves



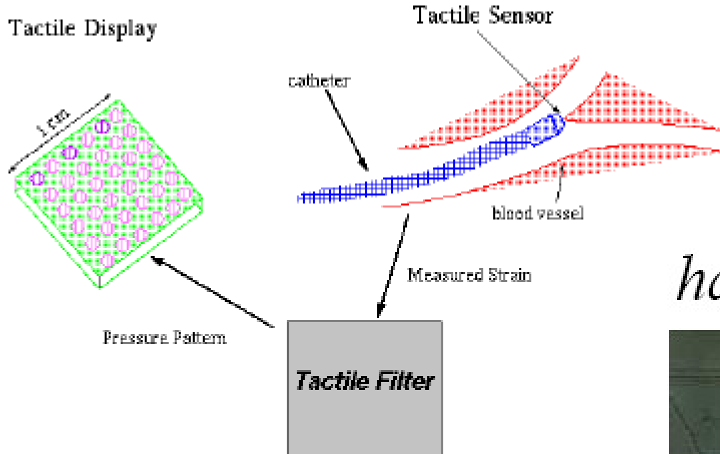
Capacitive array sensors

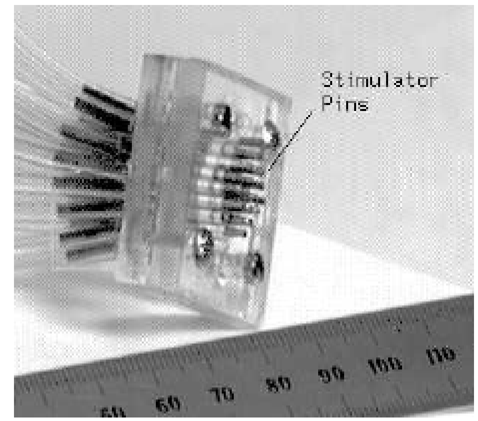
Resistive sensors

Tactile Applications

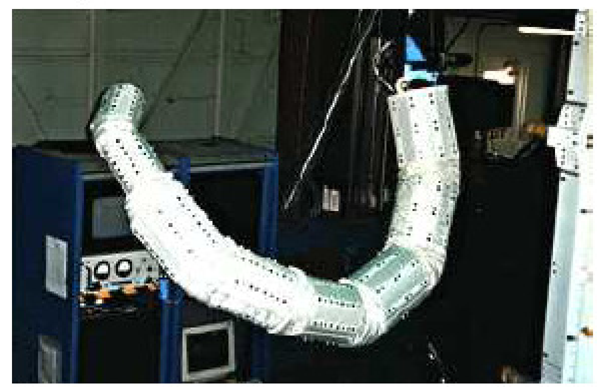
Medical teletaction interfaces

daVinci medical system





haptics



Robotic sensing
Merritt systems, FL

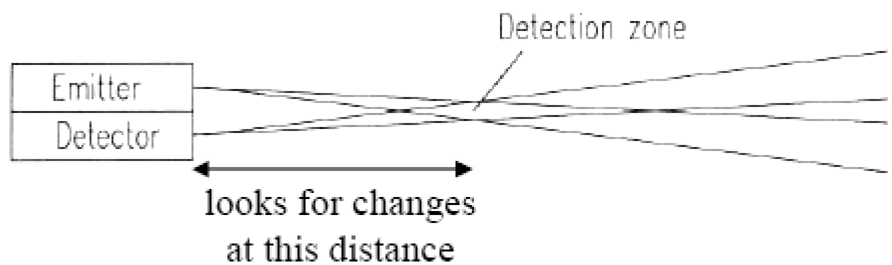
Infrared Sensors

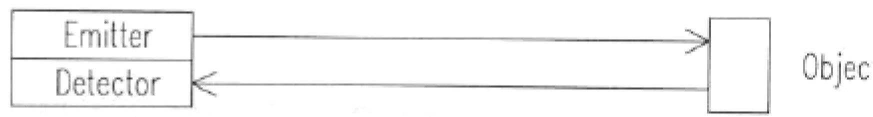
"Noncontact bump sensor"

IR emitter/detector pair

(1) sensing is based on light intensity.

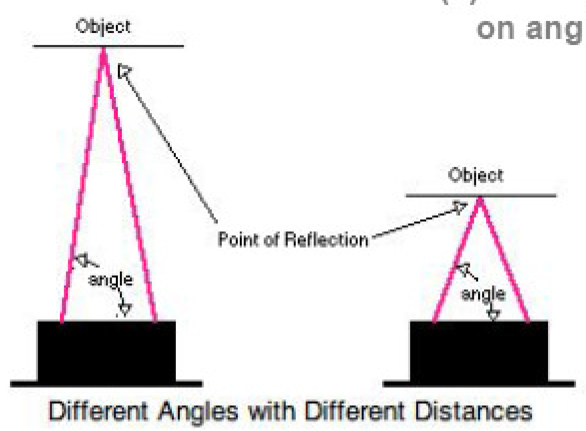
"object-sensing" IR





diffuse distance-sensing IR

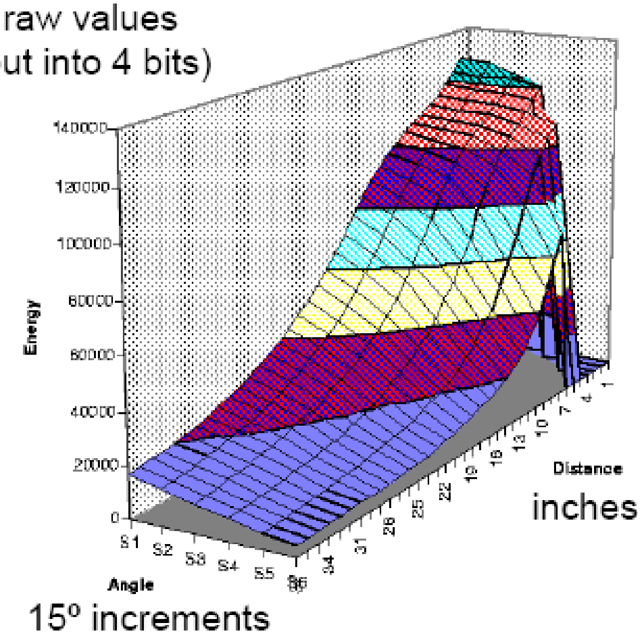
(2) sensing is based
on angle receved.



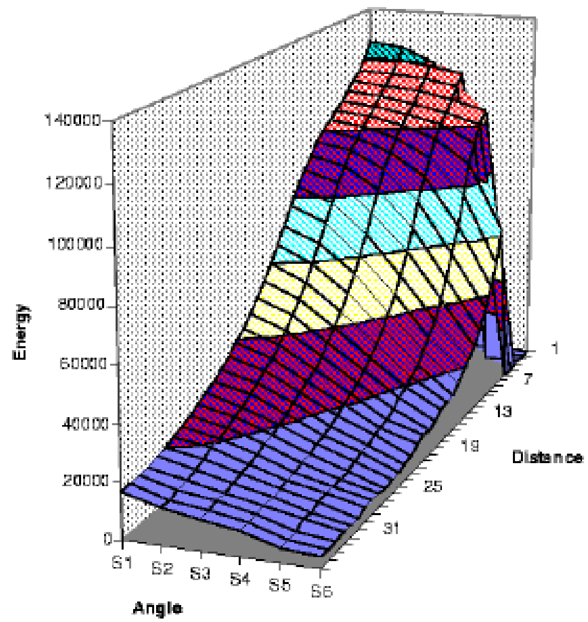
Infrared Calibration

The response to white copy paper
(a dull, reflective surface)

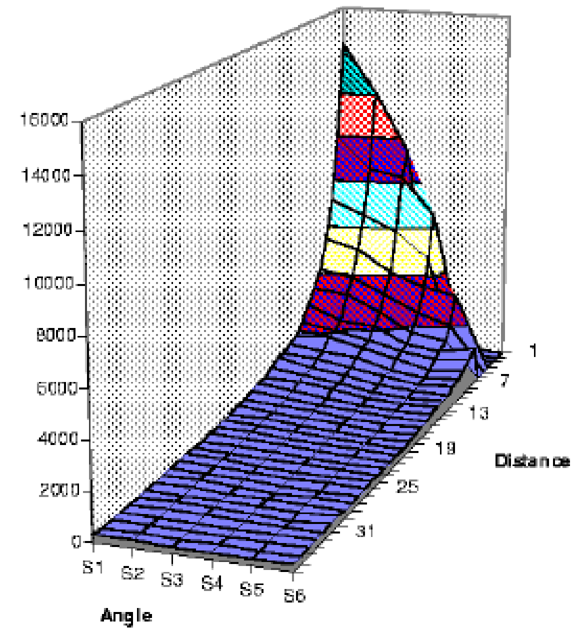
raw values
(put into 4 bits)



in the dark

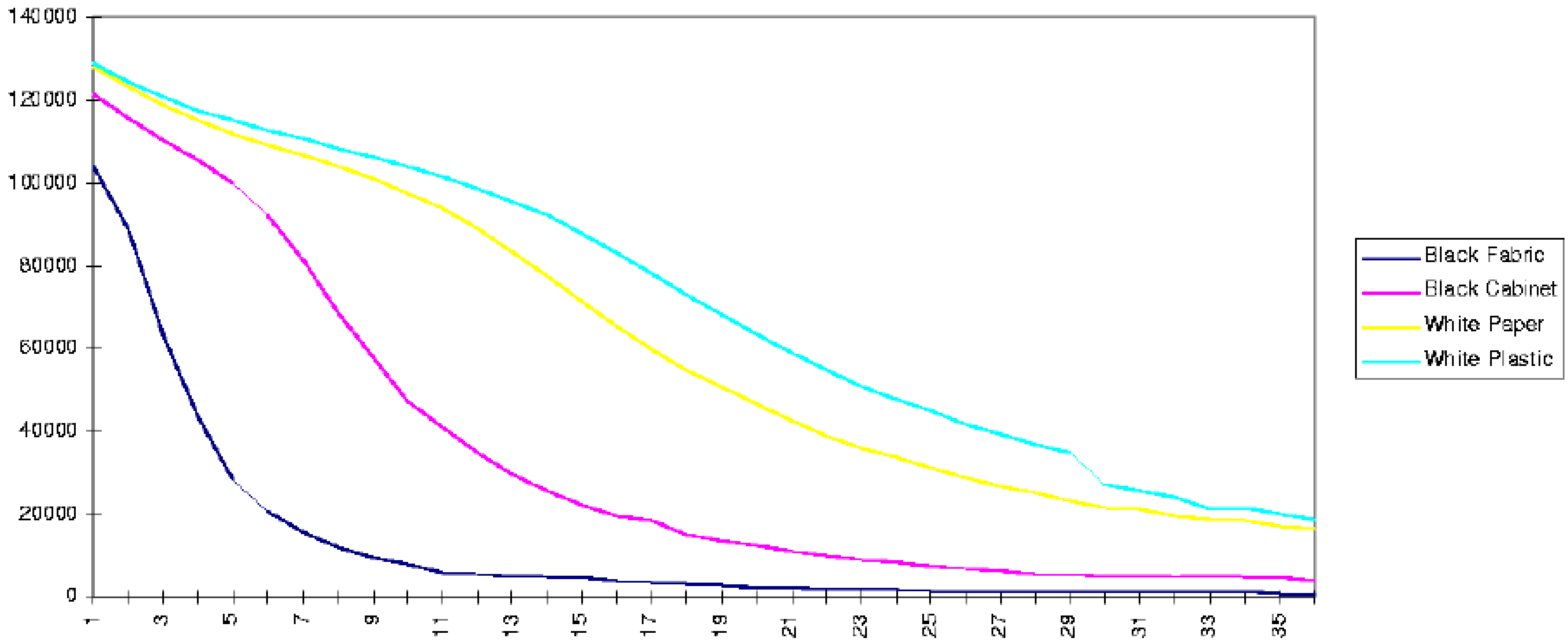


fluorescent light



incandescent light

Infrared Calibration



energy vs. distance for various materials

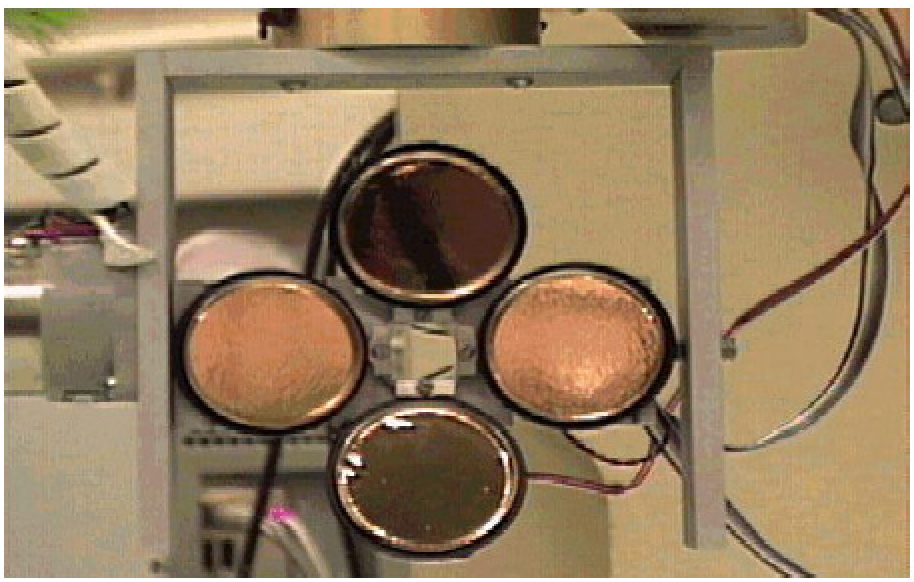
(the incident angle is 0º, or head-on)

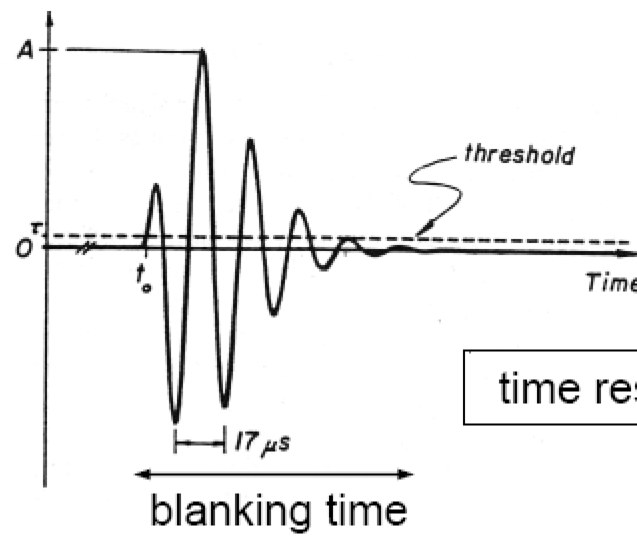
(with no ambient light)

Sonar Sensing

single-transducer sonar timeline

0	75μs		.5s
a "chirp" is emitted into the environment	typically when reverberations from the initial chirp have stopped	the transducer goes into "receiving" mode and awaits a signal... limiting range sensing	after a short time, the signal will be too weak to be detected

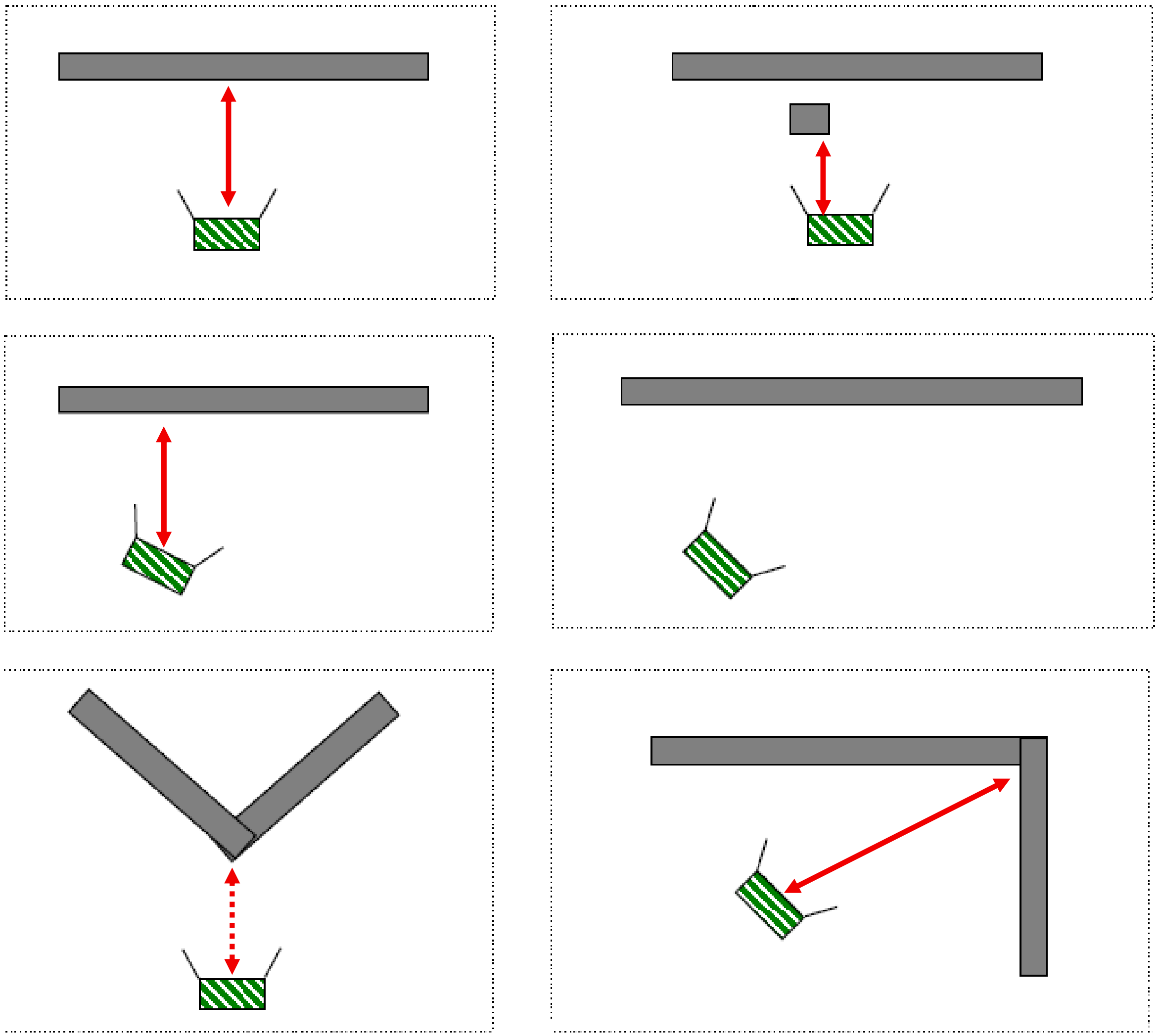




time response

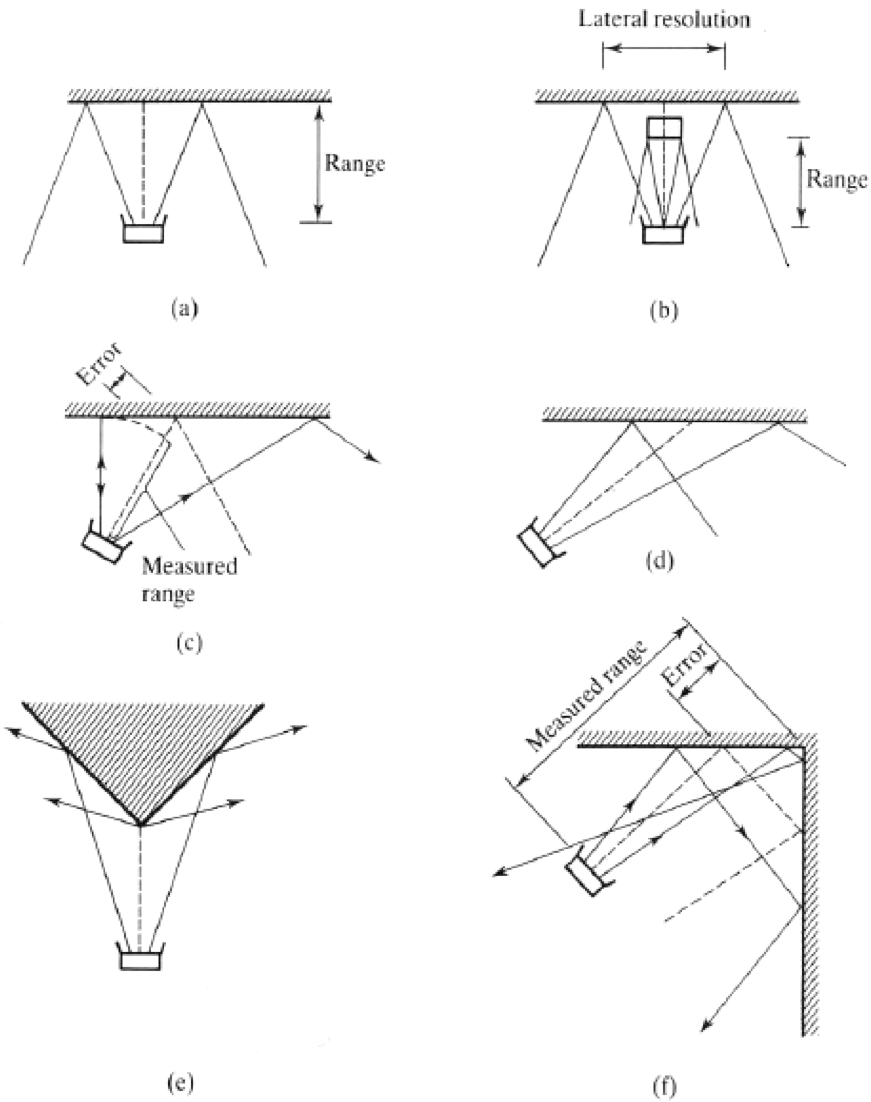
Polaroid sonar emitter/receivers

Sonar Effects



Draw the range reading that the sonar will return in each case…

Sonar Effects



(a) Sonar providing an accurate range measurement

(b-c) Lateral resolution is not very precise; the closest object in the beam's cone provides the response

(d) Specular reflections cause walls to disappear

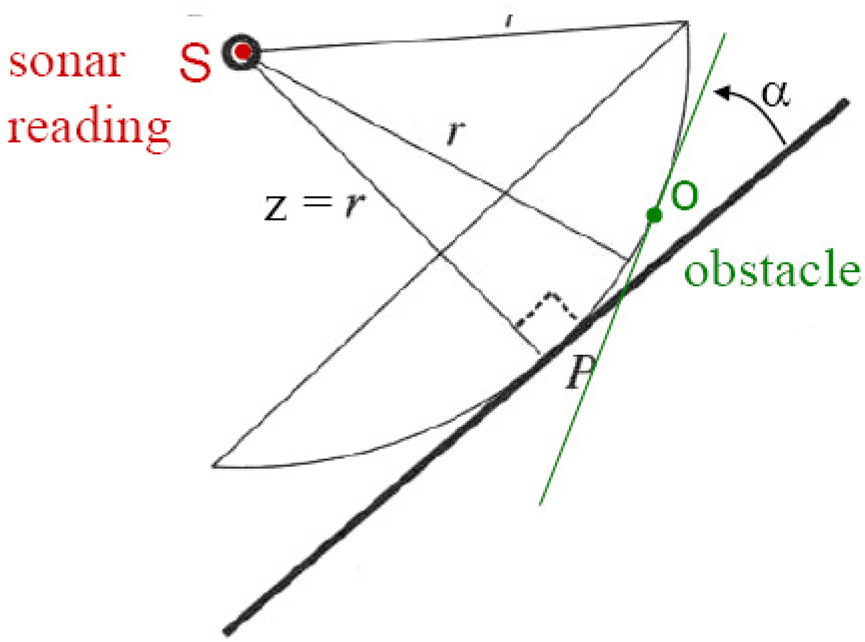
(e) Open corners produce a weak spherical wavefront

(f) Closed corners measure to the corner itself because of multiple reflections --> sonar ray tracing

Sonar Modeling

response model (Kuc)

$$h_R(t, z, a, \alpha) = \frac{2c\cos\alpha}{\pi a \sin\alpha}\sqrt{1 - \frac{c^2(t - 2z/c)^2}{a^2 \sin^2\alpha}}$$



- Models the response, h_R, with

  c = speed of sound

  a = diameter of sonar element

  t = time

  z = orthogonal distance

  α = angle of environment surface

- Then, allow uncertainty in the model to obtain a probability:

$$\boxed{p(S \mid o)}$$

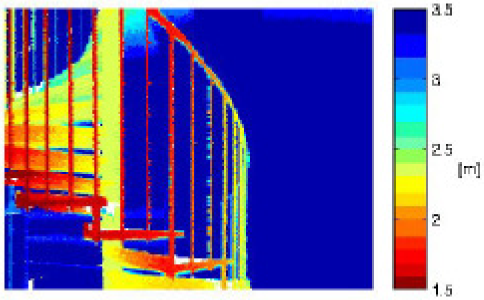
chance that the sonar reading is S, given an obstacle at location o

Laser Ranging

LIDAR

Sick Laser

Hoyuko laser



LIDAR map