Robotics
Navigation I: Bug Algorithms

Admin

- Any?
- Exam?
- Paper?

Bug Algorithms

- Bug Algorithms
  - Behavioral roboticists love(d) insects
    - Simple behaviors – easy to implement
    - Complex emergent behaviors
  - So first navigation algorithm is based on insects too
  - Bug algorithms (supposedly) capture how a bug travels
    - Straight toward goals, moving only for obstacles
- Also – don’t need to know what the world is like
  - No path planning

Bug Algorithm assumptions

- These first two bug algorithms make the following assumptions
  - The robot has a sensor capable of detecting the goal from anywhere in the field of the robot
    - eg. has GPS sensor and GPS coordinates of goal while operating outside
  - The robot has a perfect touch sensor which can know when contact has been made with the obstacle
    - Simulate with proximity sensor
    - Talk about more complex tangent bug later.
  - Robot's 'world' is bounded

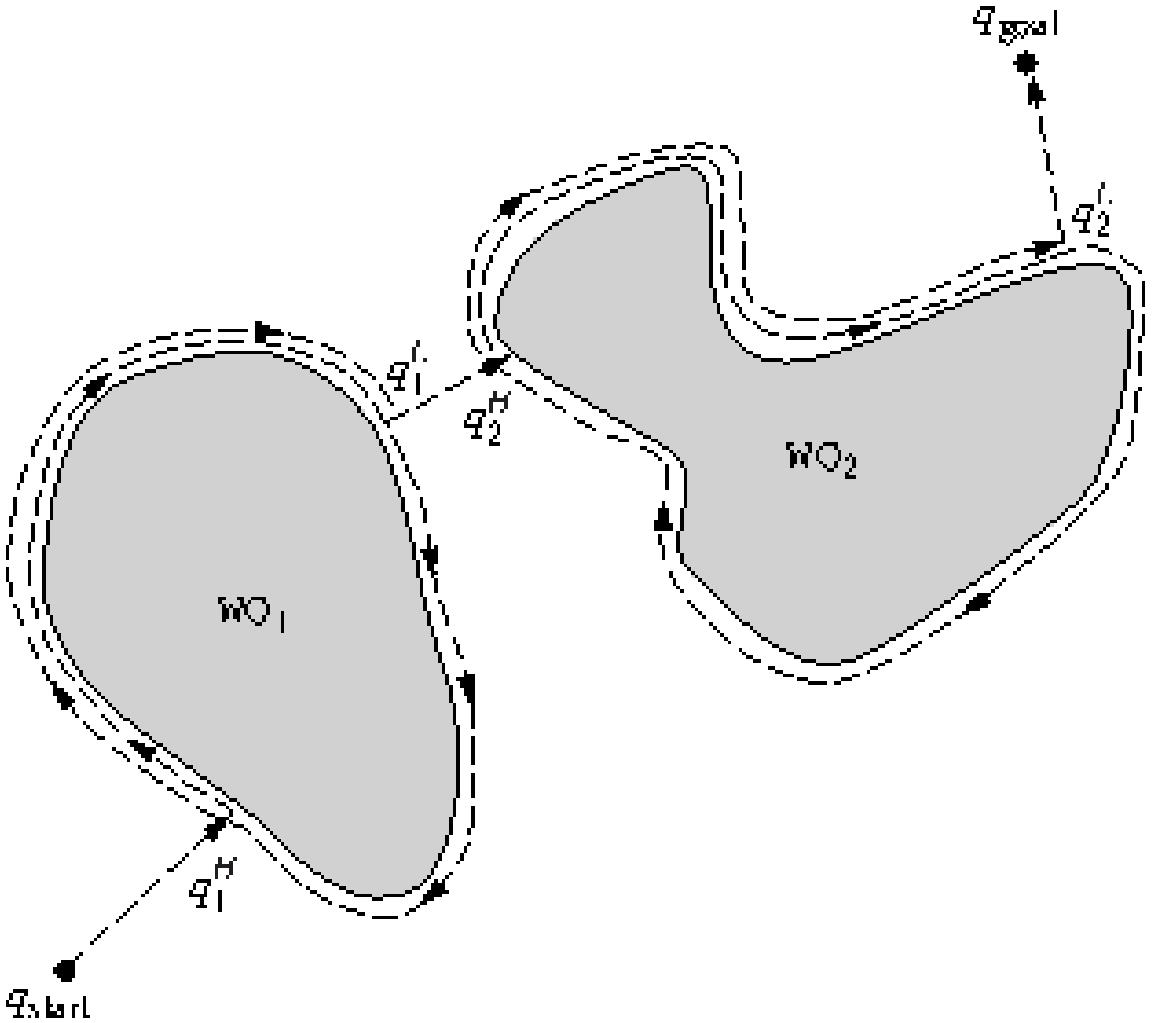
Bug1

- Bug1 Algorithm
  - Conservative Bug algorithm
  - Robot has two behaviors
    - Move to goal
    - Object boundary following
  - Move to goal
    - Bug travels from start straight toward goal
  -

Bug1 Object boundary following

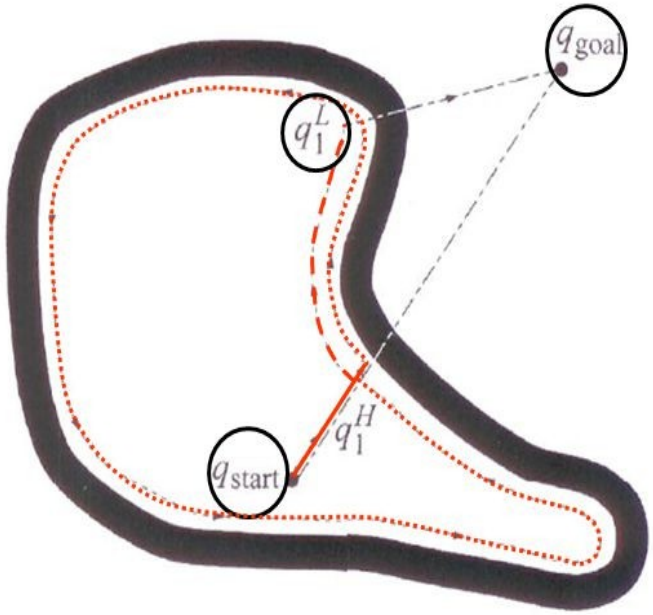
- Object Boundary following
  - If obstacle encountered circumnavigates entire obstacle.
  - Finds point on trip around object closest to goal and resumes moving to goal
  - If new line to goal intersects the current obstacle then there is no valid path to goal

Bug algorithm in picture



From Choset et al.

Bug 1 with unreachable Goal



The Bug1 algorithm reports the goal is unreachable.

Algorithm 1 Bug1 Algorithm from Original Book

Input: A point robot with a tactile sensor

Output: A path to the qgoal or a conclusion no such path exists

1: while Forever do

2: repeat

3: From q^L_{i-1} , move toward q_{goal} .

4: until q_{goal} is reached or an obstacle is encountered at q^H_i .

5: if Goal is reached then

6: Exit.

7: end if

8: repeat

9: Follow the obstacle boundary.

10: until q_{goal} is reached or q^H_i is re-encountered.

Determine the point q^L_i on the perimeter that has the shortest distance to the goal.

12: Go to q^L_i .

13: if the robot were to move toward the goal then

14: Conclude q_{goal} is not reachable and exit.

15: end if

16: end while

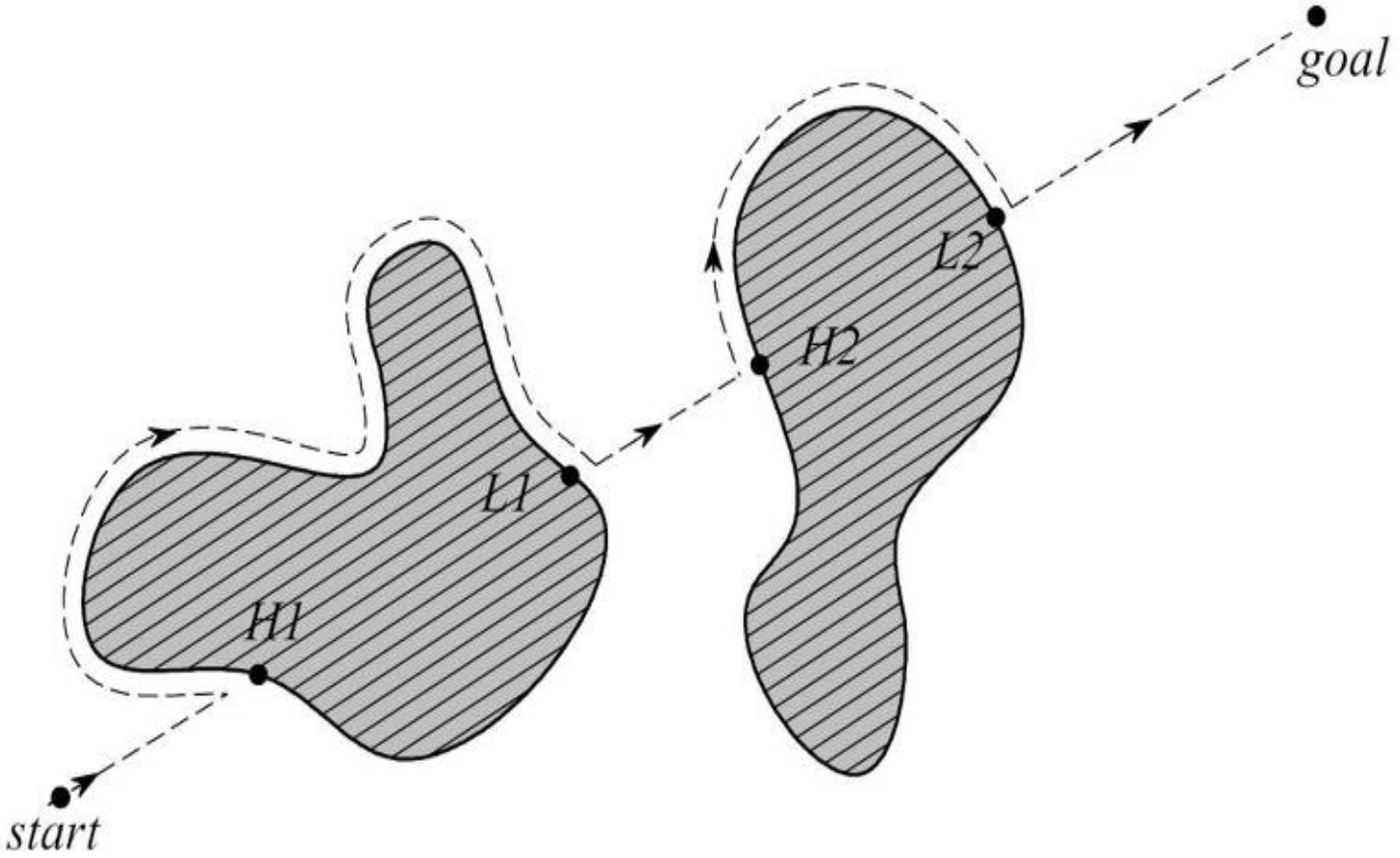
Choset's Adjustment

- Let qL0 = qstart; i = 1
- • repeat
- – repeat
- • from qLi-1 move toward qgoal
- – until goal is reached or obstacle encountered at qHi
- – if goal is reached, exit
- – repeat
- • follow boundary recording pt qLi with shortest distance to go
- – until qgoal is reached or qHi is re-encountered
- – if goal is reached, exit
- – Go to qLi
- – if move toward qgoal moves into obstacle
- • exit with failure
- – else
- • i=i+1
- • continue

Bug 2

- Basic Idea
  - Make line toward goal
  - Move along line
  - If obstacle
    - Move around obstacle till back on line
  - Move toward goal along line again
  -

Bug 2 in pictures



Bug2 doing poorly

Bug2

The “Pathological Case” where Bug2 is worse than bug1

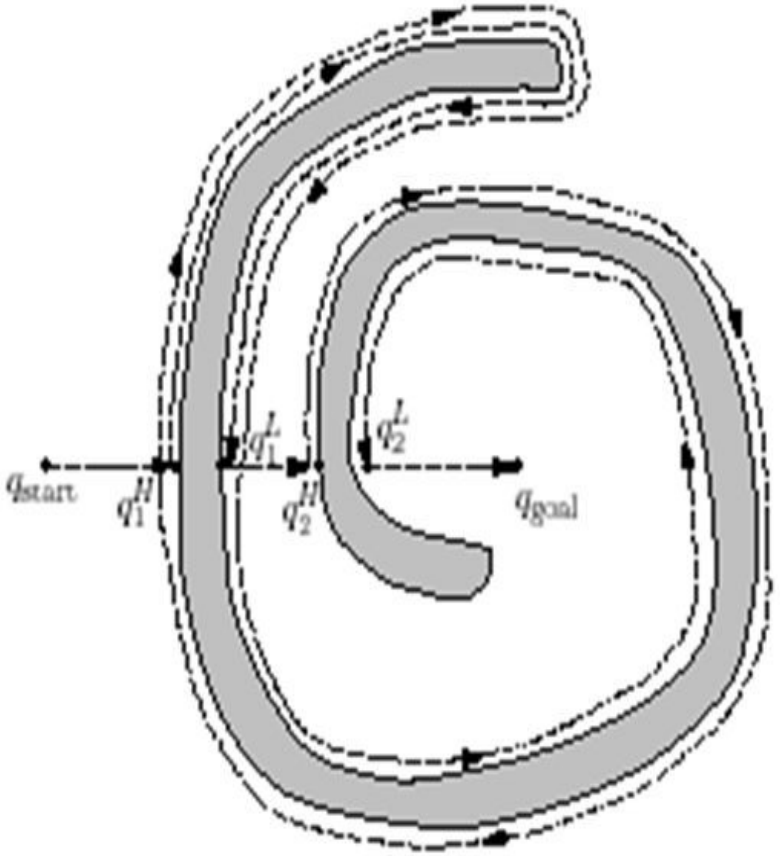


FIGURE 2.4. Bug2 Algorithm

Bug 2 Formal Algorithm

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

1: **while** True **do**
2: **repeat**
3: From q_{i-1}^{L}, move toward q_{goal} along m-line.
4: **until**
q_{goal} is reached **or**
an obstacle is encountered at *hit point* q_i^H.
5: Turn left (or right).
6: **repeat**
7: Follow boundary
8: **until**
9: q_{goal} is reached **or**
10: q_i^H is re-encountered **or**
11: m-line is re-encountered at a point m such that
12: $m \neq q_i^H$ (robot did not reach the hit point),
13: $d(m, q_{goal}) < d(m, q_i^H)$ (robot is closer), and
14: If robot moves toward goal, it would not hit the obstacle
15: Let $q_{i+1}^{L} = m$
16: Increment i
17: **end while**

can be arbitrarily longer

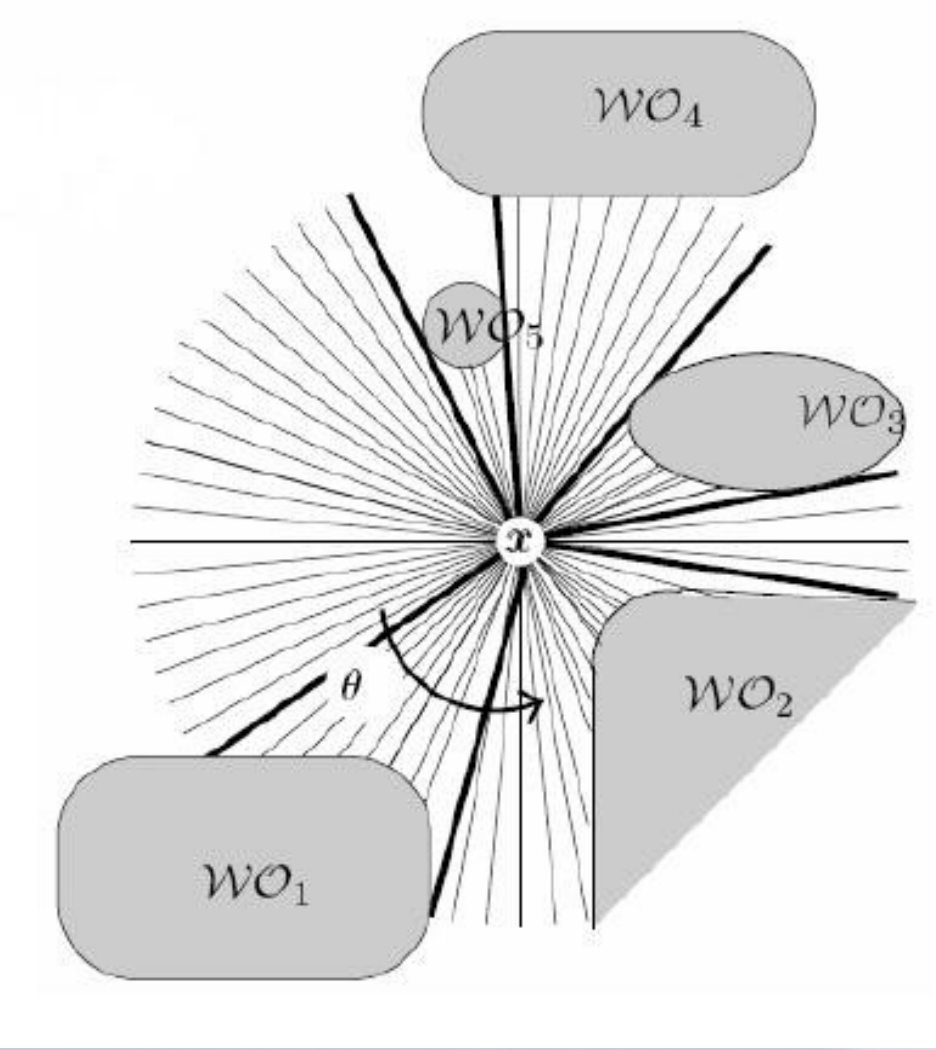
- Algorithm from Choset et al page 22

Tangent Bug

- Meant as improvement on Bug2
  - Shorter path to goal
  - Theory assumptions
    - Infinite range sensor with perfect accuracy.
  - In practice
    - Range sensor with good accuracy
    - Anything reading at the max range of the sensor is assumed to be a clear area for now

Find areas of discontinuity in depth

Each of the dark lines are areas of
Discontinuity. We need to care about those



Areas of discontinuity

- Limited by sensors
- Might find no discontinuity even when there is one if we reach max sensor range
-

Tangent algorithm

- Moves toward goal
  - Moving around obstacles as needed
  - Until goal is encountered or local minimum distance from goal encountered.
- Move around obstacle (in same direction)
  - Till goal reached
  - Or
  - Complete cycle around obstacle or
  - We have a clear path toward the goal again

Example path

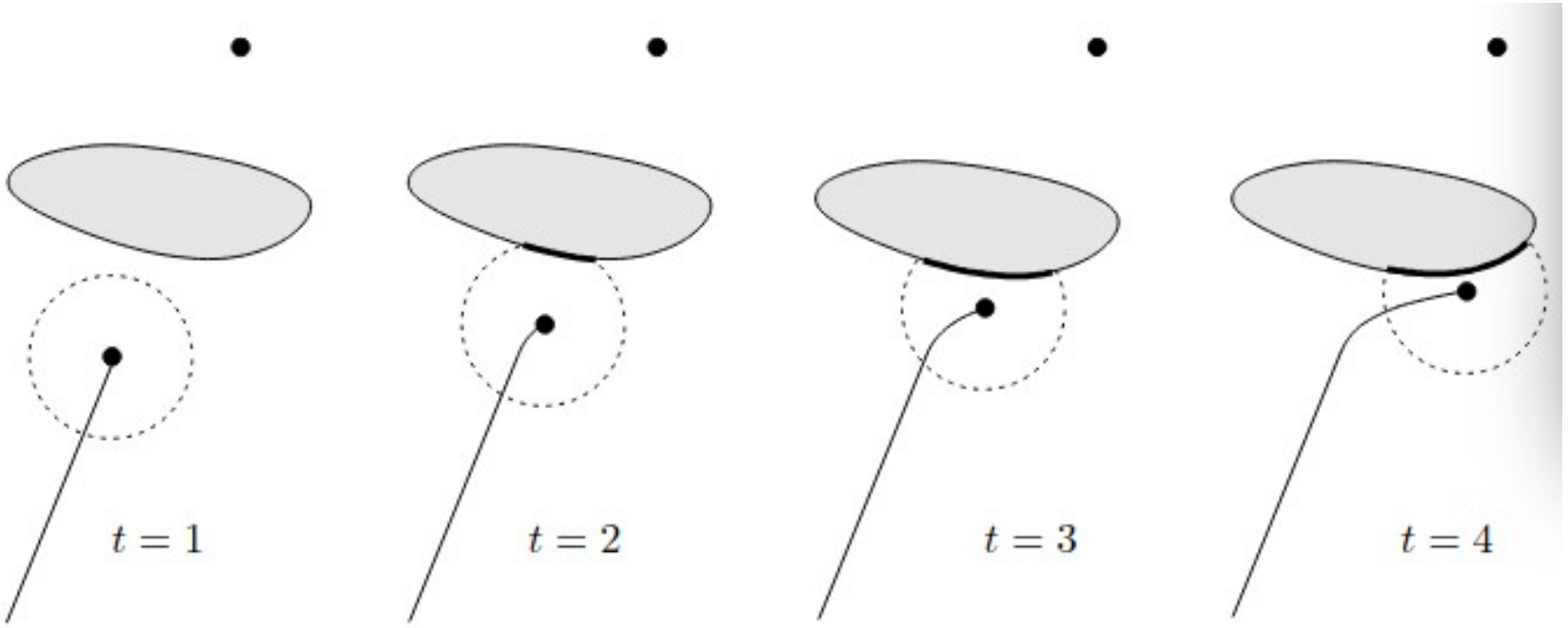


FIGURE 2.8. Demonstration of motion-to-goal behavior for a robot with a finite sensor range moving toward a goal which is "above" the light gray obstacle.

Two more examples

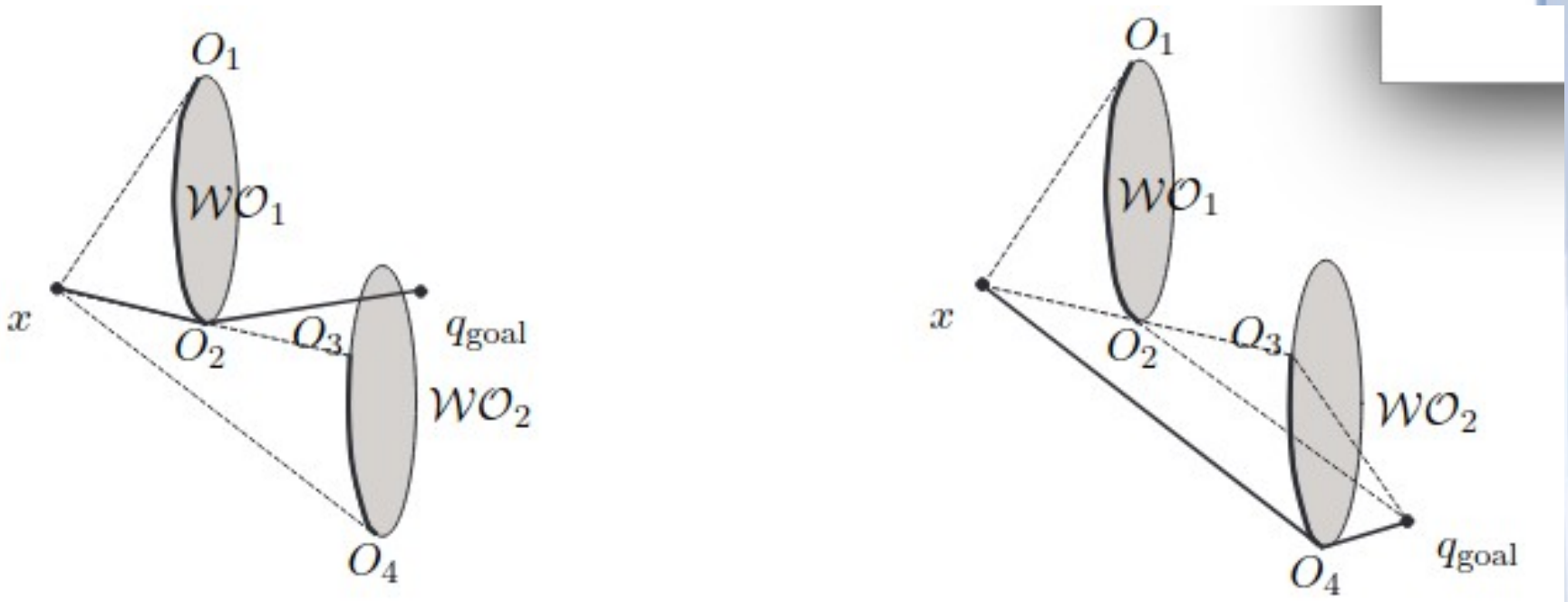


FIGURE 2.7. (Left) The planner selects O_2 as a subgoal for the robot. (Right) The planner selects O_4 as a subgoal for the robot. Note the line segment between O_4 and q_{goal} cuts through the obstacle.

Tangent Bug algo

Algorithm 3 Tangent Bug Algorithm

Input: A point robot with a range sensor

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

1: **while** True **do**

2: **repeat**

3: Continuously move toward the point $n \in \{T, O_i\}$ which minimizes $d(x, n) + d(n, q_{\text{goal}})$ where $d(n, q_{\text{goal}}) < d(x, q_{\text{goal}})$

4: **until**

- the goal is encountered **or**
- The direction that minimizes $d(x, n) + d(n, q_{\text{goal}})$ begins to increase $d(x, q_{\text{goal}})$, i.e., the robot detects a "local minimum" of $d(\cdot, q_{\text{goal}})$.

5: Choose a boundary following direction which continues in the same direction as the most recent motion-to-goal direction.

6: **repeat**

7: Continuously update d_{leave}, d_{min}, and $\{O_i\}$.

8: Continuously moves toward $n \in \{O_i\}$ that is in the chosen boundary direction.

9: **until**

- The goal is reached.
- The robot completes a cycle around the obstacle in which case the goal cannot be achieved.
- $d_{\text{leave}} < d_{\text{min}}$

10: **end while**

Tangent bug example 1

- Tangent bug with contact sensor only
- Figure 2.11 page 29

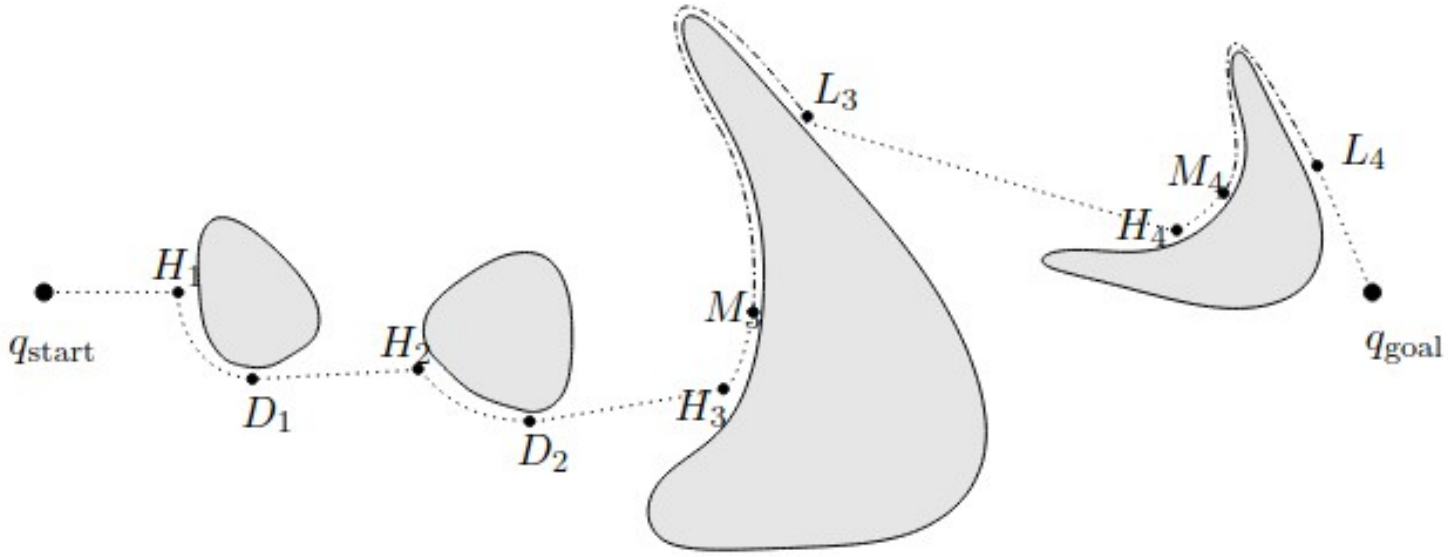
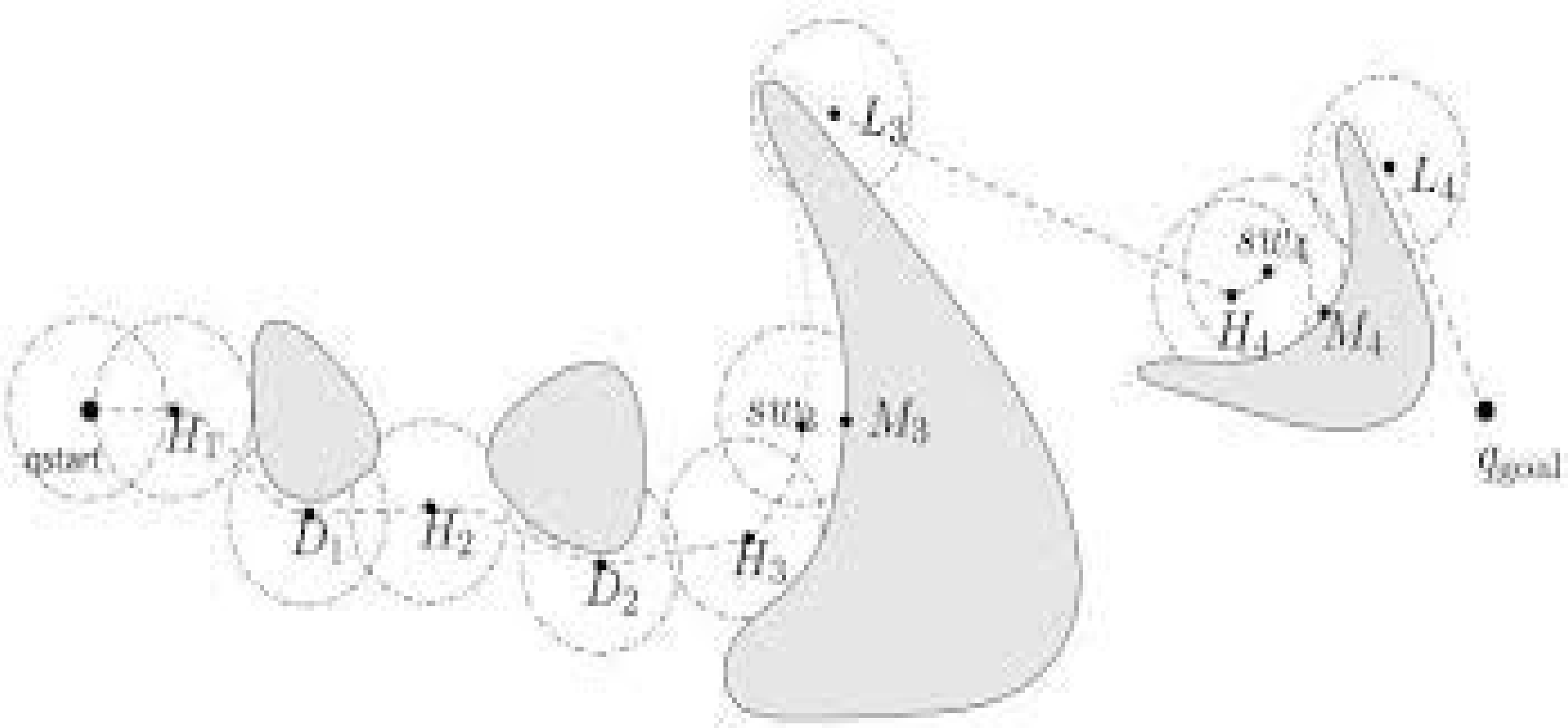


FIGURE 2.10. The path generated by Tangent Bug with zero sensor range. The dashed lines correspond to the motion - to - goal behavior and the dotted lines correspond to boundary-following.

Tangent Bug example 2

- Tangent Bug with limited distance sensor
-



Navigation algorithms and practice

- Several of these algorithms make assuptions
  - Bug algo – how does it find the SG line again?
  - How does robot know where on map it is?
  - Good odometry?
    - How was your odometry for the lab?

Landmark Navigation

- Navigate using landmarks in the world
  - Have a map of the world with land mark positions
  - What landmarks?
    - Buildings, mountains, lakes etc for people.
      - Require scene understanding/object identifications.
      - If encounter landmark from one direction, can robot identify it as same object if “seen” from other direction?
    - Perception only landmarks
      - Position of sun, heat source, sound from a particular direction, etc.
      - Florescent paper
      - Eliminate understanding stage of robot landmark navigations.

Perceptual Landmarks

- Must be
  - Visible from many positions
  - Recognizable under many conditions (viewing angles, lighting etc.
  - Have a known location
    - Either stationary or move in known, predictable manner (eg the sun)

Perceptual landmarks II

- First two points require
  - generalized internal representation of landmark
  - Raw sensor data not so useful even when using canonical representation.
    - Use neural net?
- Pitfall
  - Perceptual aliasing.

Landmark navigation is for the bees

- ... and the ants
  - Take reference sensor reading (retina image)
  - When ready to navigate to landmark
    - Try to match current sensor image to reference image
    - Move to reduce difference between the two
    - Draw ant experiment to illustrate

Canonical routes

- Establish well known path from start to goal
  - Use combination of dead reconing and sensor id of path to navigate it
    - Wood ant paths
    - Human roads
      - What other canonical routes do we use?

When navigating

- Use as much sensor information as you can
  - Limitations
    - Sensors
    - Processing power
    - Program quality.
  - Eg
    - Odometry to dead reckoning
    - Landmark identification
    - Together help reinforce each other.

Navigation I and II Summary

- Looked at ways of representing and navigating space
- All were somewhat idealized
  - Each with their own tradeoffs
  - Each better for some robots and some environments than others.
-

Reading Assignment