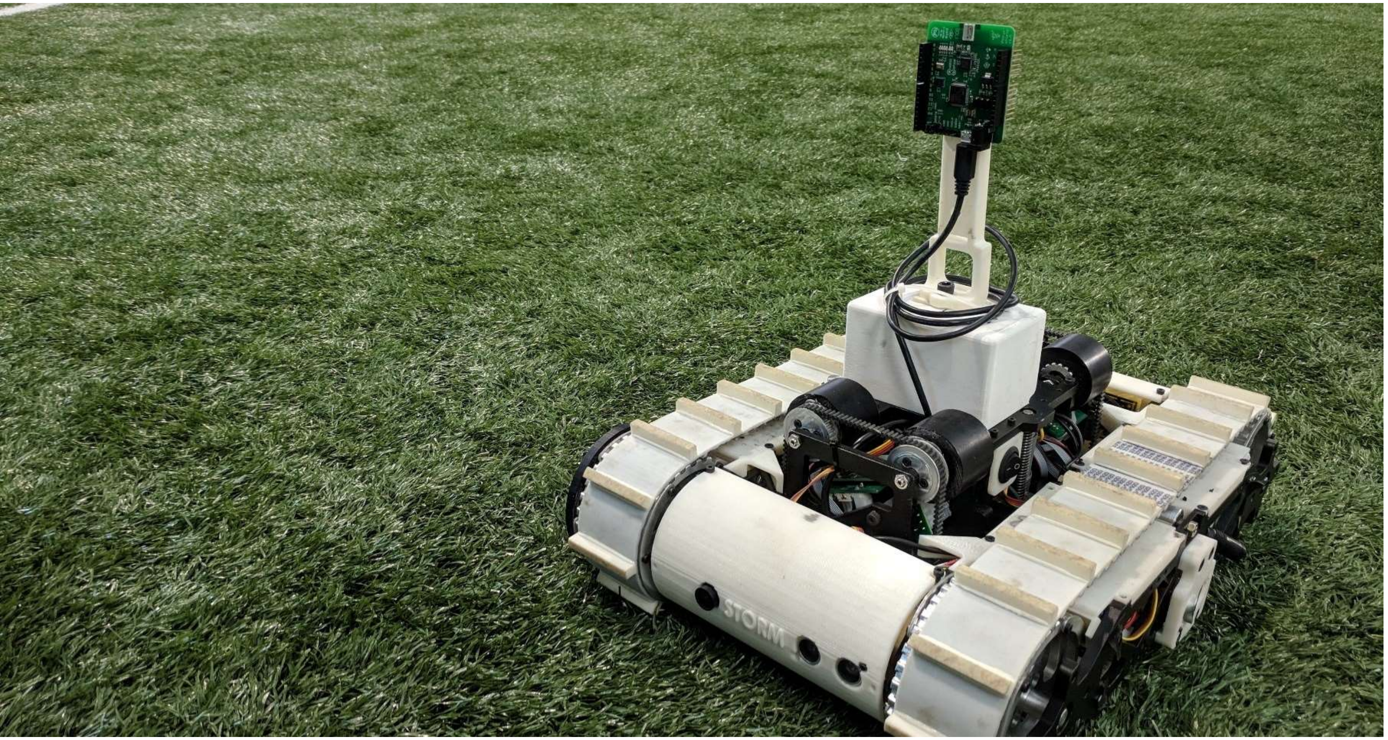


ED 5215

INTRODUCTION TO MOTION PLANNING

Instructor: Bijo Sebastian

MODULE - 1

Planning for:

- Point robot that can move in any direction
- Known environment with stationary obstacles
- Perfect sensing
- Perfect control

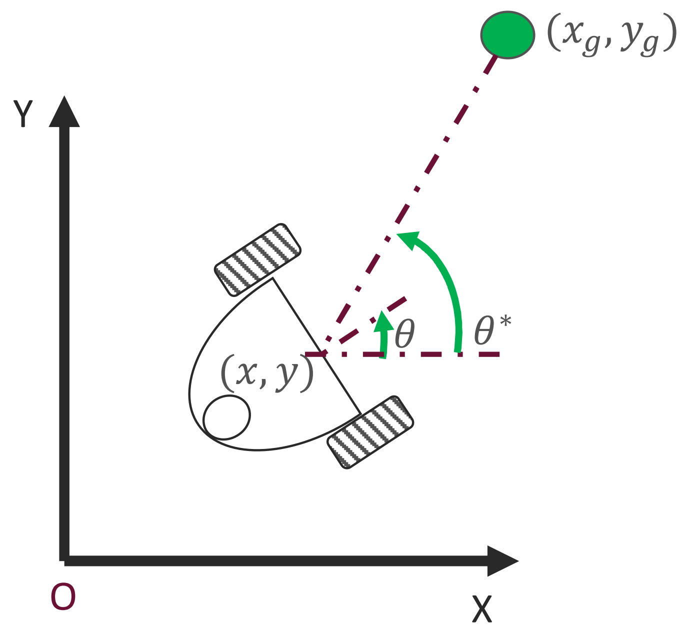
Approaches discussed:

Bug algorithms, Graph search, Depth-first, Breadth-first, Dijkstra, A*, Dynamic programming and Potential fields

MOTIVATION

<u>Go-to-goal controller:</u>

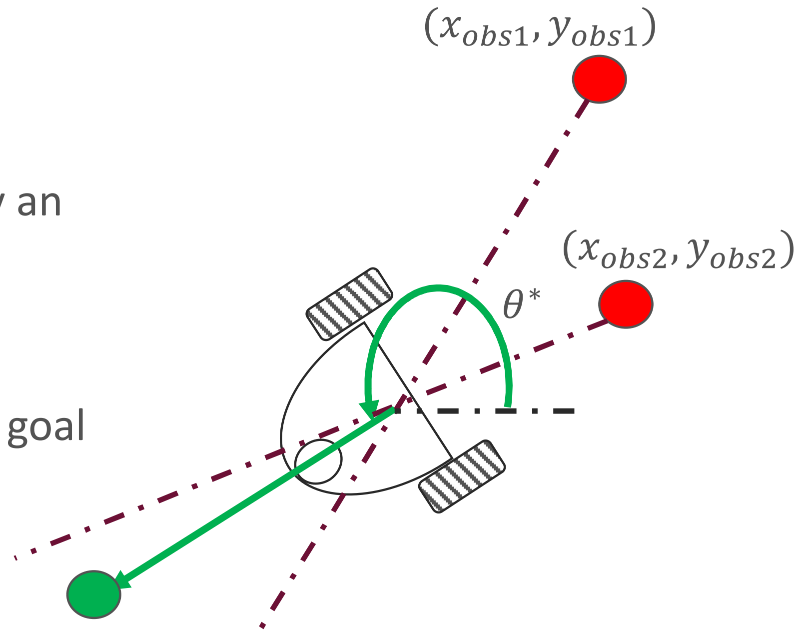
- Draw line connecting robot to goal
- Slope of line connecting robot to goal point → desired orientation θ^*, given by: $\theta^* = tan^{-1}(\frac{y_g - y}{x_g - x})$
- Error in orientation of robot: $e = \theta^* \ominus \theta \in [-\pi, \pi]$
- A PID controller on angular velocity
- Simple P controller on linear velocity



MOTIVATION

Avoid obstacle controller:

- Draw a line connecting the robot and obstacles
- Starting at robot's origin, extend the line away from the obstacle by an amount inversely proportional to the distance to obstacle
- Take a vector sum of the newly constructed lines , result is the new goal



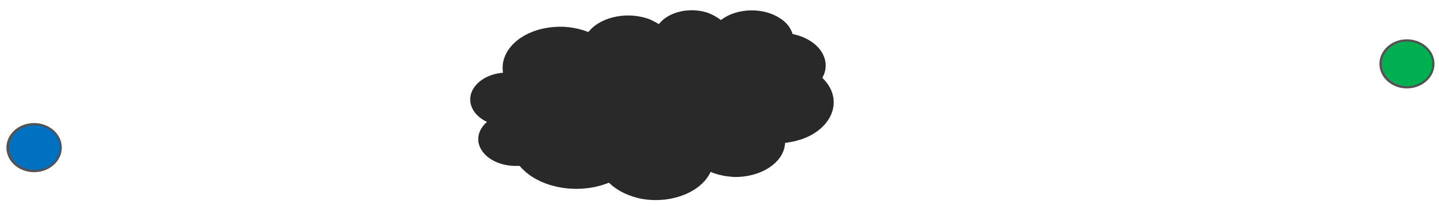
Question: Does this always work?

MOTIVATION

Let's simplify:

- Point robot in a 2D environment (Can move in any direction)
- Known environment with stationary obstacles
- Perfect sensing
- Perfect control
- Perfect localization

<u>Case : 1</u>



MOTIVATION

Let's simplify:

- Point robot in a 2D environment (Can move in any direction)
- Known environment with stationary obstacles
- Perfect sensing
- Perfect control
- Perfect localization

<u>Case : 2</u>

MOTIVATION

Robot behaviors:

- Go-to-goal
- Avoid-obstacle
- ?? (Can we add more behaviors ?)

Case : 2

MOTIVATION

Robot behaviors:

- Go-to-goal
- Avoid-obstacle
- Wall following

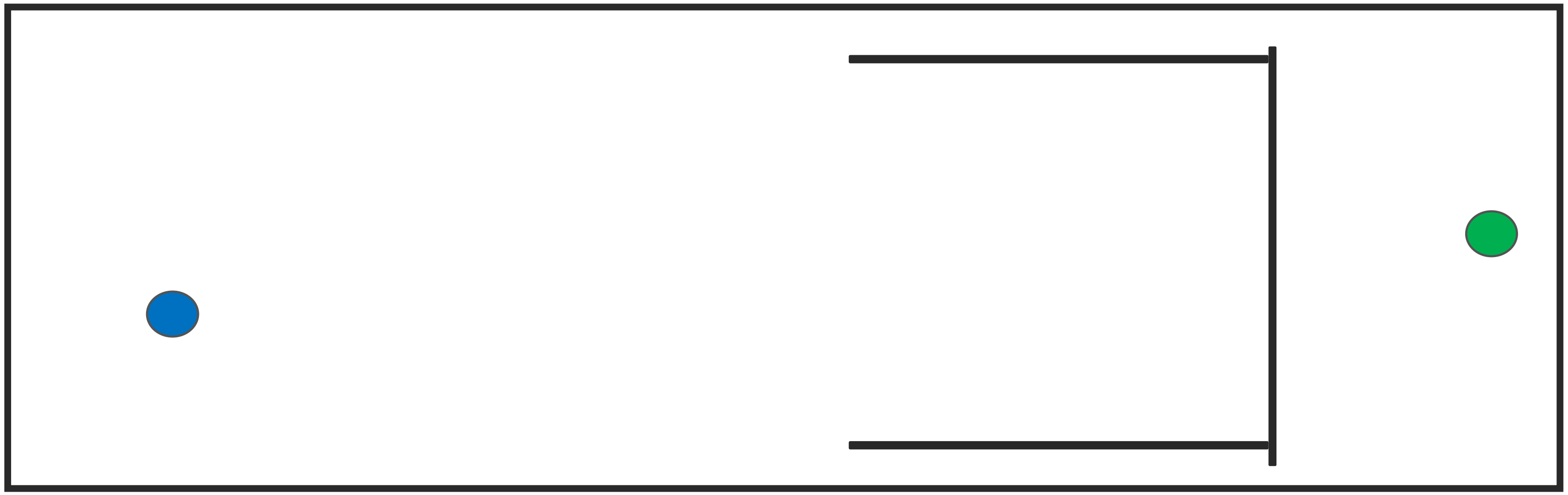
<u>Case : 2</u>

MOTIVATION

More assumptions (in addition to the ones stated before):

- Perfect “bump” sensor → Can follow any obstacle boundary
- Bounded 2D environment with finite number of obstacles
- Obstacles are well behaved → A line intersects an obstacle finitely many times

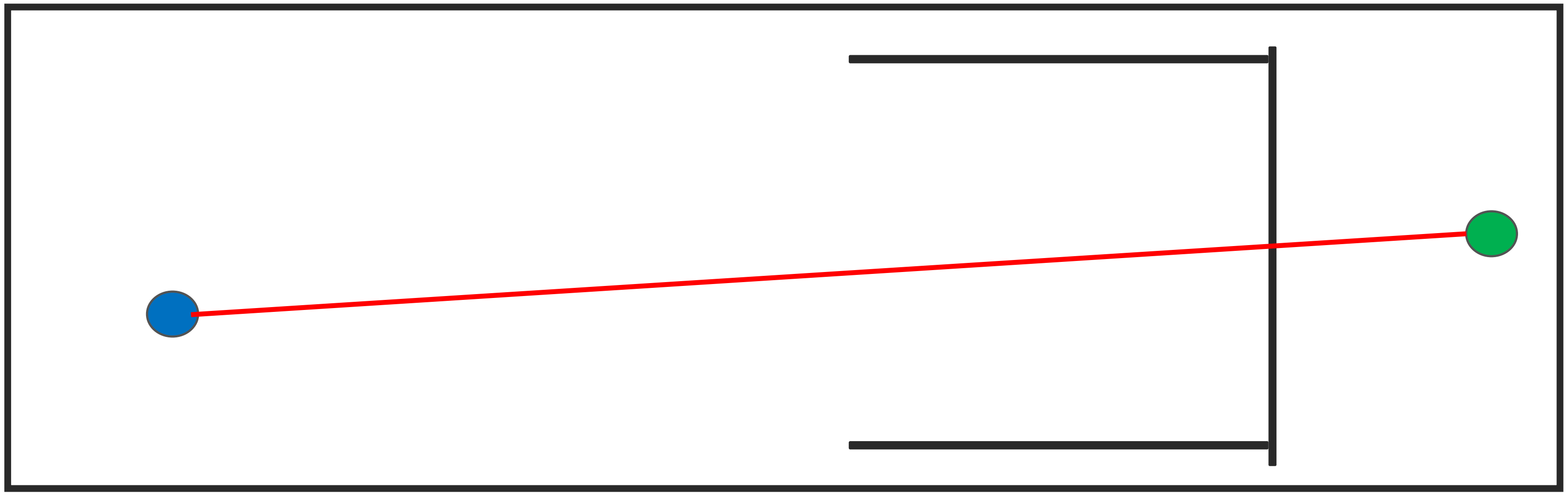
State an algorithm that allows you to successfully reach goal.



BUG-0 ALGORITHM

While not at goal:

- Move towards the goal in a straight line (m-line)
- If you hit an obstacle:
  - Follow its boundary clockwise till you reach the m-line again



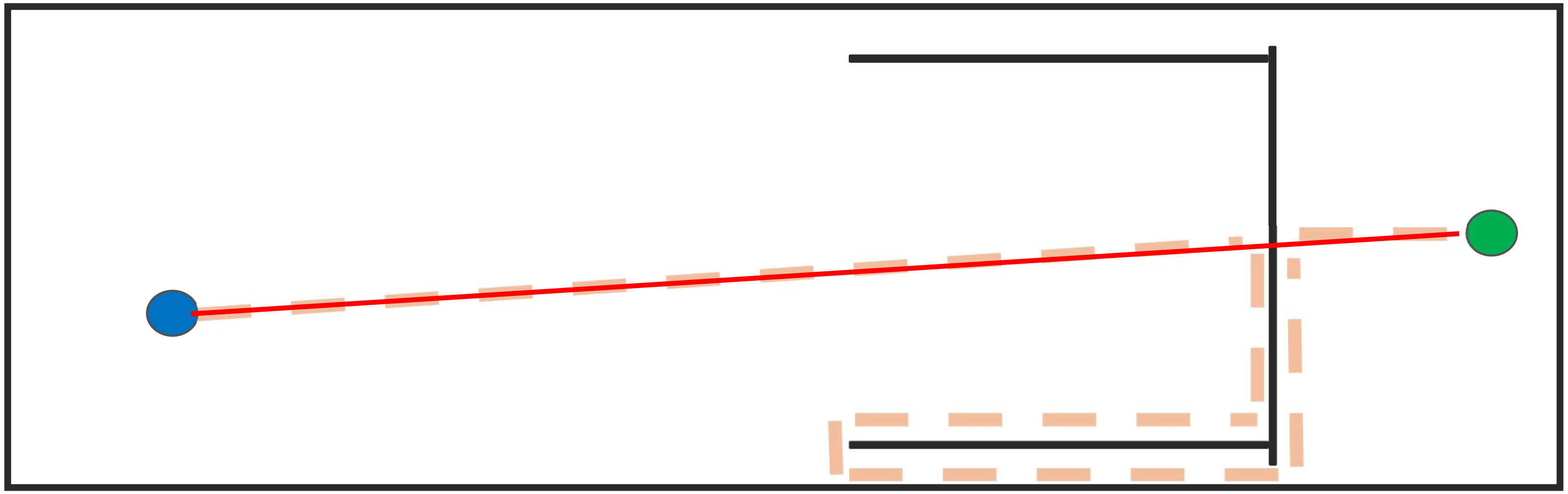
BUG-0 ALGORITHM

While not at goal:

Move towards the goal in a straight line (m-line)

If you hit an obstacle:

Follow its boundary clockwise till you reach the m-line again



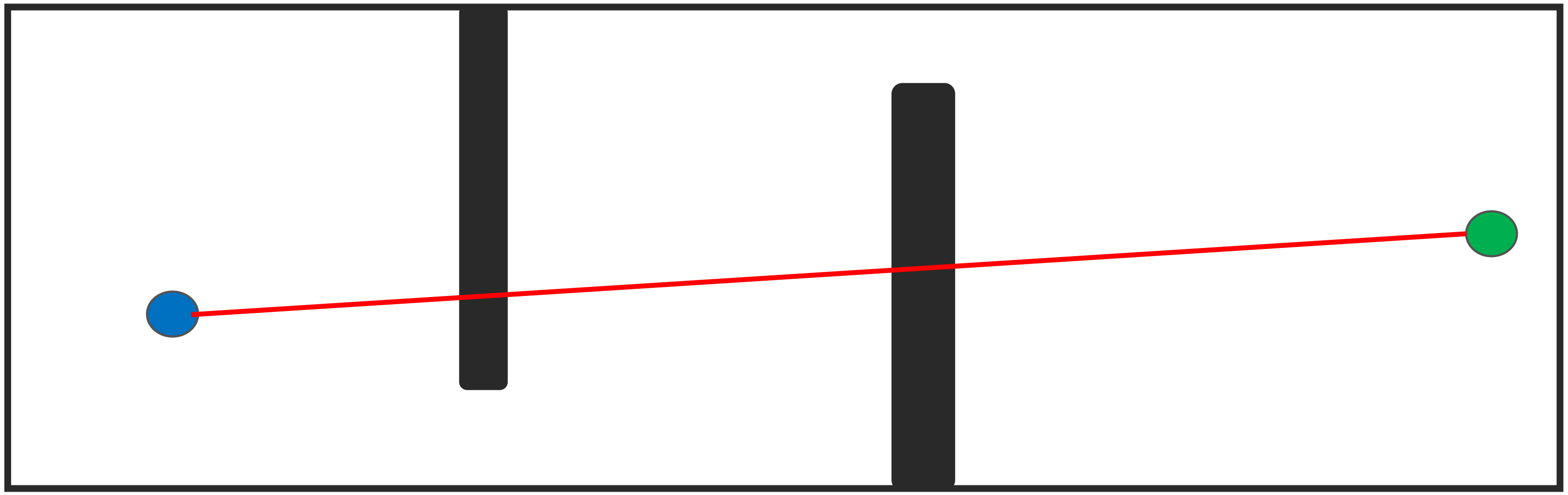
Question: Does Bug-0 always work?

BUG-0 ALGORITHM

Completeness: An algorithm is called complete if:

- it returns a feasible solution, if one exists;
- returns FAILURE in finite time, otherwise

Question: Is Bug-0 complete?

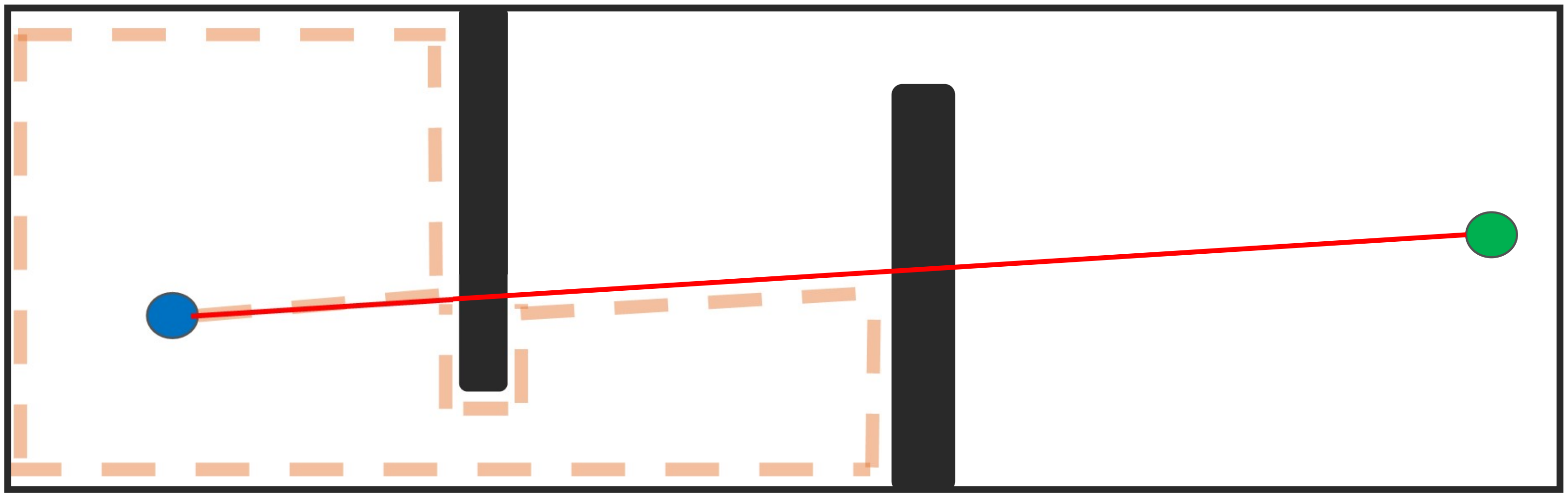


BUG-0 ALGORITHM

Completeness: An algorithm is called complete, if:

- it returns a feasible solution, if one exists;
- returns FAILURE in finite time, otherwise

Bug 0 is not complete



Question: How do we improve upon Bug-0 ?

BUG-0 ALGORITHM

Issue of Bug-0:

- No measure of progress → gets stuck in a loop

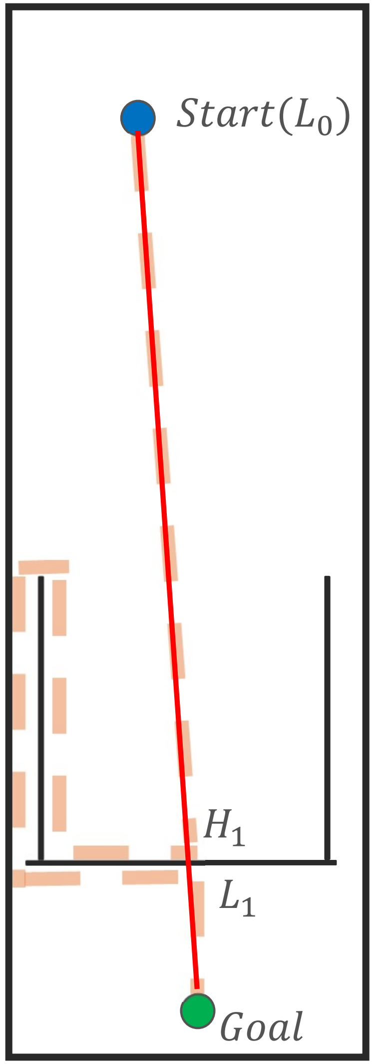
(This is a common problem for purely greedy/local approaches, as seen later)

Solution:

Add some global measure of progress → Bug 2

SOME DEFINITIONS

- For a scenario, we are given the start and goal points for the bug
- A "hit point" denoted by H_i (for $i = 1{,}2, \ldots\ldots$)is the point at which the bug hits/encounters and obstacle. There could be many hit points in a given scenario
- A "leave point" denoted by L_i (for $i = 0{,}1{,}2, \ldots\ldots$) is the point at which the bug leaves the boundary f an obstacle and continues to move toward the goal. The start point becomes L_0
- A path is a sequence of hit/leave pairs bounded by start and goal



BUG-2 ALGORITHM

Stay on the line connecting start to goal (m-line)

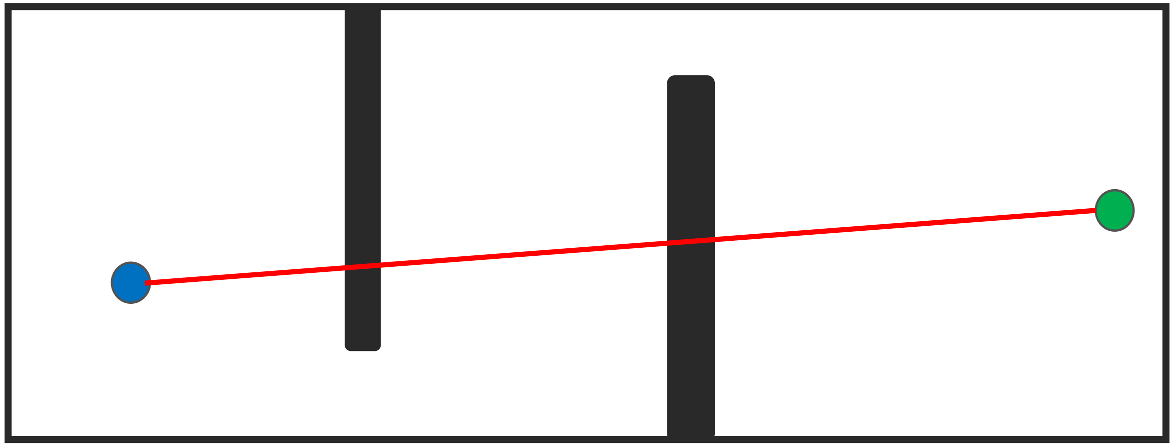
If hit an obstacle at a point (H_i):

Follow the boundary until you reach either:

- goal → TERMINATE
- H_i → declare FAILURE
- m-line again → check if we are closer:

  If yes, then move along m-line

  Else continue



BUG-2 ALGORITHM

Stay on the line connecting start to goal (m-line)

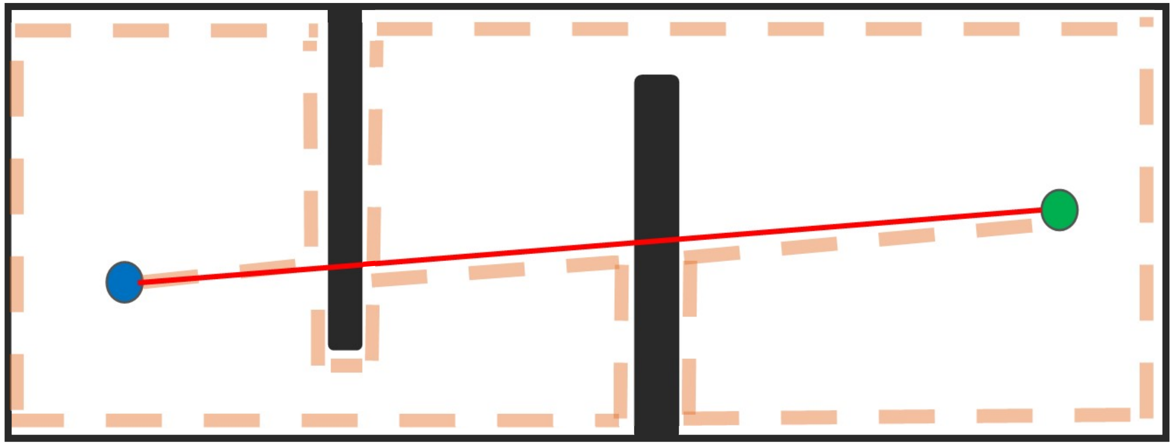
If hit an obstacle at a point (H_i):

Follow the boundary until you reach either:

- goal → TERMINATE
- H_i → declare FAILURE
- m-line again → check if we are closer:

  If yes, then move along m-line

  Else continue



BUG-2 ALGORITHM (FORMALLY)

Initialize: $L_0 = start, i = 1$

While not at goal:

Move towards the goal in a straight line (m-line)

If you encounter an obstacle O_i, store the point, H_i:

Follow the boundary clockwise until you reach either:

goal → TERMINATE

H_i → declare FAILURE

m-line again → check if we are closer.:

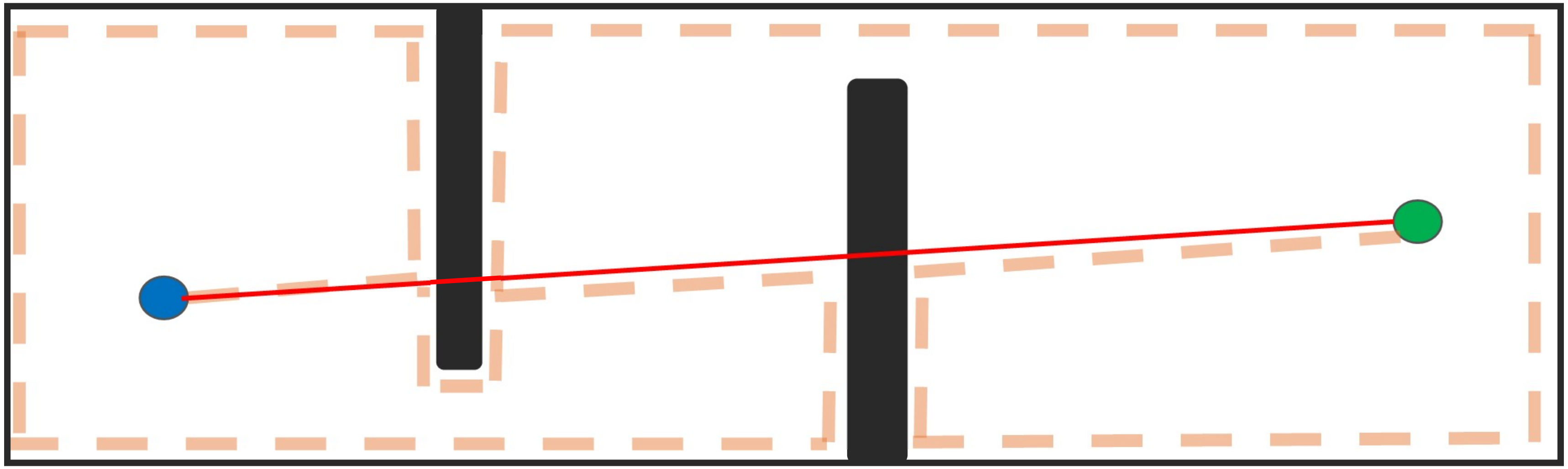
If yes then store the point L_i, set $i = i + 1$, then move along m-line

Else continue following the obstacle boundary

BUG-2 ALGORITHM

Bug 2 is complete

- If the goal is reachable, it guarantees that the robot reaches the goal
- If the goal is not reachable, it reports failure in finite time



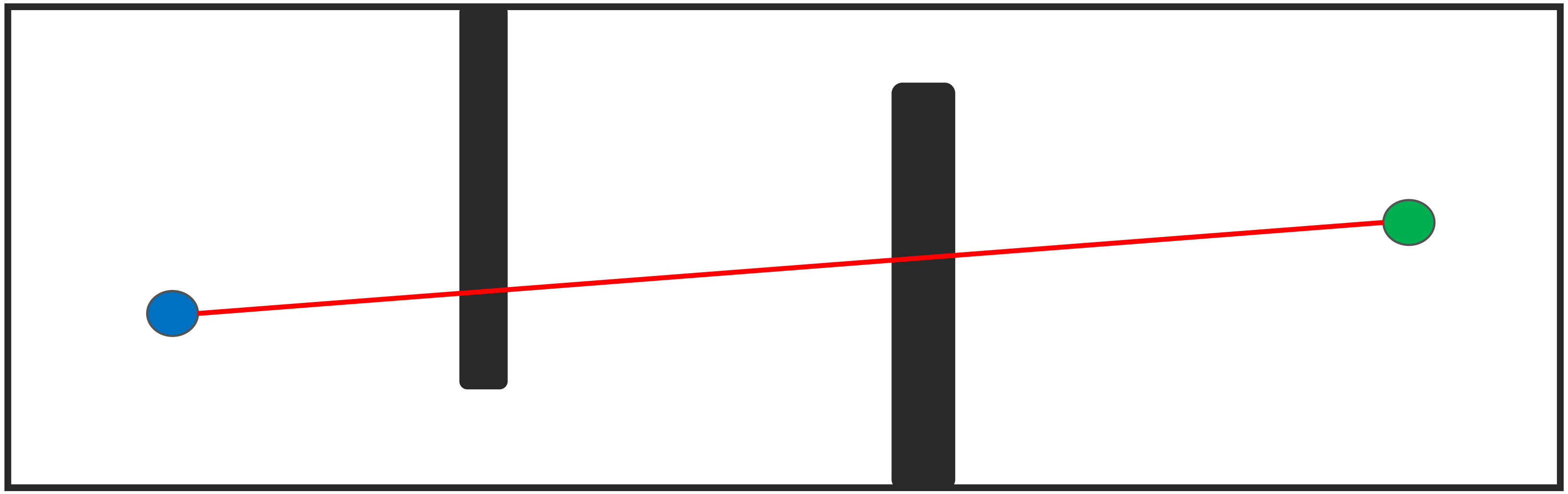
BUG-1 ALGORITHM

Move towards the goal

- If an obstacle O_i is encountered at location H_i:

    Follow the boundary of O_i clockwise until H_i is revisited

    Move to the point (along shortest direction) on O_i that is closest to the goal (call this point L_i)



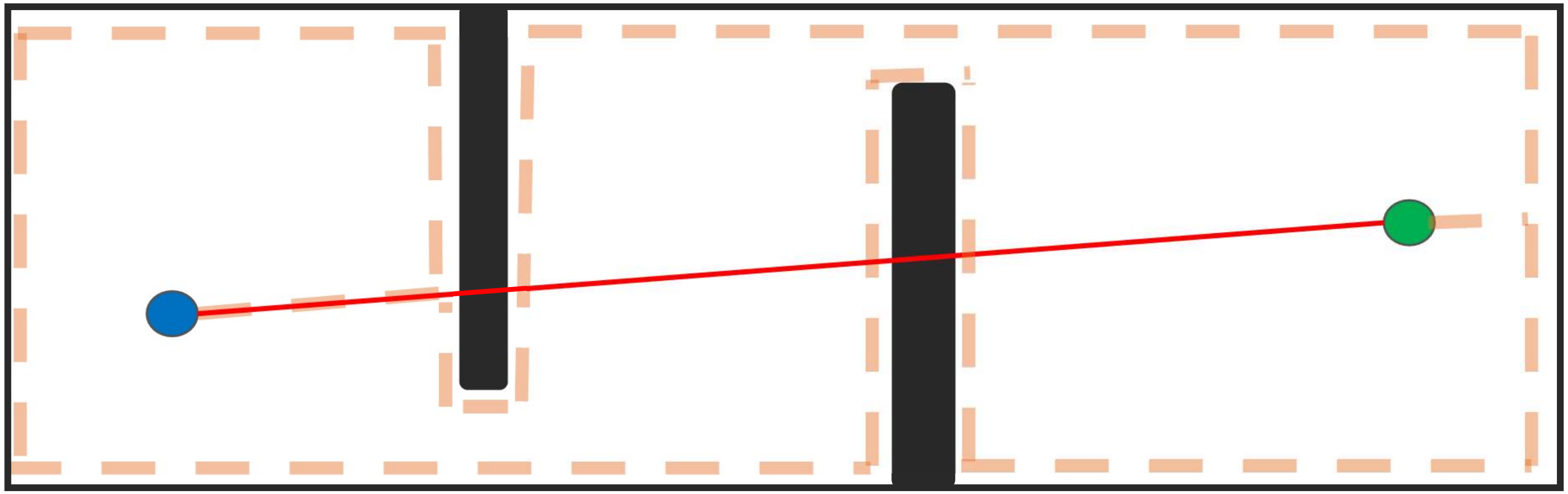
BUG-1 ALGORITHM

Move towards the goal

- If an obstacle O_i is encountered at location H_i:

  Follow the boundary of O_i clockwise until H_i is revisited

  Move to the point (along shortest direction) on O_i that is closest to the goal (call this point L_i)



BUG-1 ALGORITHM (FORMALLY)

Initialize: $L_0 = start, i = 1$

While not at goal:

Move towards the goal in a straight line (m-line)

If an obstacle O_i is encountered at location H_i:

Follow the boundary of O_i clockwise until H_i is revisited

Move to the point (L_i) on O_i that is closest to the goal, along shortest direction

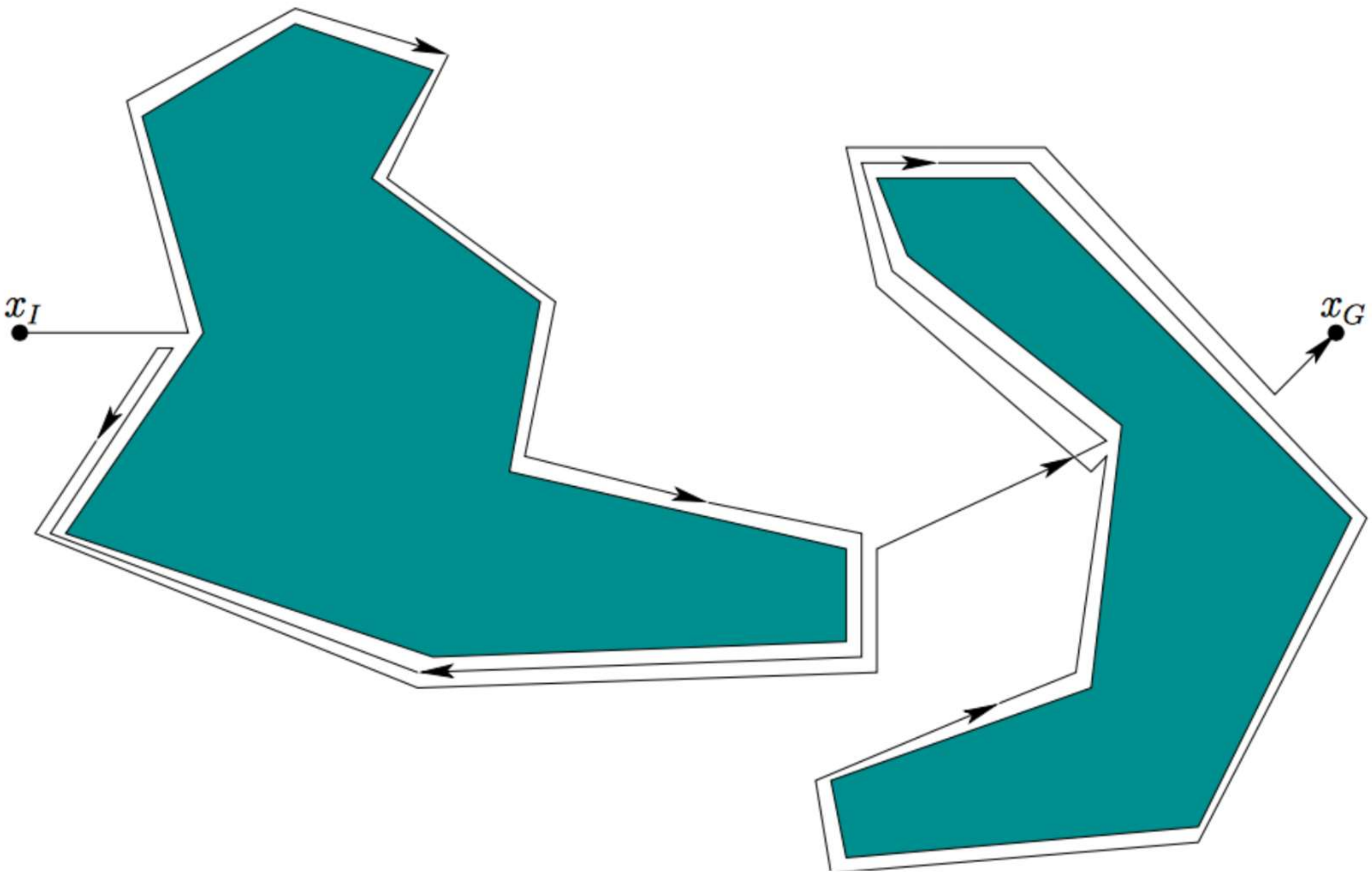
From L_i try to move towards goal

If this brings you closer to obstacle

exit with failure

else, $i = i + 1$ and continue the while loop

BUG-1 ALGORITHM



An illustration of Bug 1 strategy. Ch 12. Planning algorithms , S.M. LaValle

BUG ALGORITHMS

Questions:

1. Is Bug-1 Complete?
2. Is Bug-2 better than Bug-1 in all possible scenarios? Can you come up with some example scenarios that prove otherwise?

BUG ALGORITHMS

Questions:

1. Is Bug-1 Complete? Yes, refer to original paper for detailed explanation
2. Is Bug-2 better than Bug-1 in all possible scenarios? Can you come up with some example scenarios that prove otherwise?

<u>Additional reading:</u>

- Chapter 12.3.3 of “Planning Algorithms” book by S. M. LaValle
- The original paper that discussed Bug algorithms:

Lumelsky, Vladimir J., and Alexander A. Stepanov. "Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape." Algorithmica 2.1-4 (1987): 403-430