Assignment 1: Bug Algorithms

Graduate Student: Tze-Yuan Cheng

1.(5pts) Bug 2 meets only a finite number of obstacles. Moreover, the only obstacles that can be met are those that intersect the straight-line segment from init to goal

Proof: For Bug 2 Algorithm, only the obstacle that intersect the m-line can be met by Bug 2, and all the line segments from **Li** to **Hi+1 (i = 0,1,….)** are within the circle of radius **d(init, goal)** centered at goal since each hitting point **Hi** is closer than the last leaving point **Li-1**. Based on the assumption that any finite disc can intersect only a finite number of obstacles, and there are only a finite number of obstacle in the environment, we can conclude that Bug 2 meets only a finite number of obstacles.

2.(8pts) Bug2 will pass any point of the i-th obstacle boundary at most $n_i/2$ times, where n_i is the number of intersections between the straight line (init, goal) and the i-th obstacle

Proof: There are two arguments to prove the Lemma:

Argument A: In Bug 2 Algorithm, for each obstacle i, after hitting each hit point H, the Bug 2 will walk around the obstacle until reaching the leave point L, which is on the intersections between the straight line and obstacle. Therefore, all Hj and Lj appear in pairs.

Argument B: Under the environment assumption, since obstacles are of finite thickness, the distance from Hj to goal : d(Hj), is less than the distance from Lj to goal : d(Lj), and the after leaving Lj, the Bug 2 will walk on the straight line (int, goal) until hitting the next hit point Hj+1, so d(Lj)> d(Hj+1). Therefore, we will have the consecutive inequalities:

$$d(Hj) > d(Lj) > d(Hj+1) > d(Lj+1) > d(Hj+2) > d(Lj+2) \ldots\ldots$$

Therefore, based on **B**, we can infer that any point on the i-th obstacle can be defined as H or L for only once, and according to **A**, H & L will appear in pairs, and when each pair of H and L appears, any point on the obstacle will be passed at most once. Thus, we can conclude that Bug 2 will pass any point of the i-th obstacle boundary at most $n_i/2$ times, where n_i is the number of intersections between the straight line (init, goal) and the i-th obstacle.

My Bug Algorithm

A.(8pts) Explain the general idea and the basic steps of your algorithm:

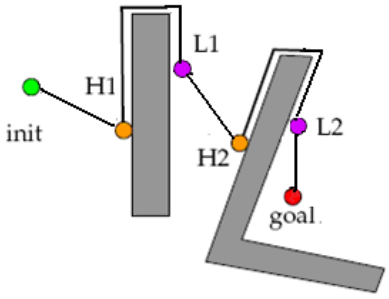
My Bug Algorithm can be viewed as the intermediate version between Bug 0 and Bug 2 algorithms. At the beginning, the bug heads toward the goal until the sensor reports contact with an obstacle; upon hitting the obstacle at point Hi, the bug will **remember** the distance from **Hi** to goal : **d(Hi, goal)**, and then follow the obstacle until it finds a leaving point **Q**, at which the bug can head toward the goal without hindrance, and the distance from **Q** to goal **d(Q, goal)** has to be shorter than **d(Hi, goal),** and then take **Q** as **Li** to heads toward the goal. The Pseudo-code is summarized as follows:

My Bug Pseudo-code

1: L0 ← init; i ← 1
2: **loop**
3: **repeat** move on a straight line from Li-1 to goal
4: **until** goal is reached or obstacle is encountered at Hi
5: **if** goal is reached **then** exit with success
6: **else** remember the distance **d(Hi, goal)**
7: **repeat** follow boundary **until**
- Q is reached, at which:
A. bug can heads toward the goal AND B. d(Q, goal) < d(Hi, goal)
8: Li ← Q
9: **if** move on straight line from Li toward goal moves into obstacle then exit with failure
10: **else** i ← i + 1

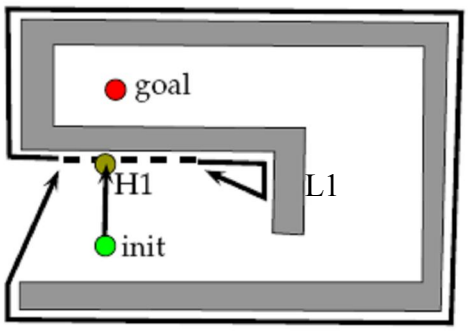
B.(2pts) Draw a simple scene and trace your algorithm:

As shown in the following scene, the bug first move toward the goal, as it hits the first obstacle at H1, **d(H1,goal)** is remembered, and then it begins to follow the boundary until a “free” leave point L1 of **d(Li, goal)** shorter than **d(Hi, goal)** is found, and it again leaves the obstacle at L1 in the direction of goal until hitting the second obstacle.

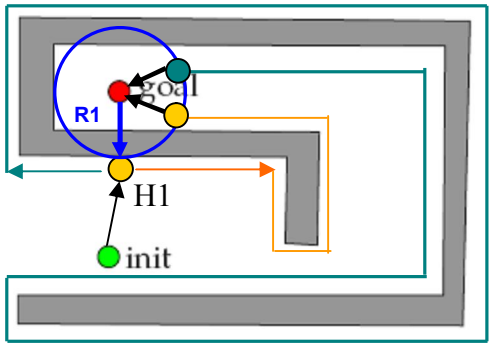


C. (4pts) Draw a scene where your algorithm does better than Bug0 (if you cannot, then you need to redesign your algorithm):

In the following scene, when using **Bug 0,** the bug will be trapped within the corner once it finds a free leave point L1, and it will never find any other leave point to reach the goal no matter it turns left or right at H1:



As shown in the following figure, using **My Bug** algorithm in the same scene, however, the bug will first remember the distance R1= d(H1, goal), and no mater turning left (orange path) or right (blue path), it will keep following the boundary until a “free point” of shorter distance to goal than d(H1, goal) is found to reach the goal; in other words, a free leave point within the circle of radius R1 will be found as a leave point, and the bug will not be trapped at any point.



D. (3pts) What are the strengths and weaknesses of your algorithm in comparison to the other Bug algorithms?

Strength: Generally, since the leave point only needs to have shorter distance than that at the hit point, My Bug can always find the leave point before finishing a complete circle, and therefore spend less time on searching for leave point than Bug 1. As compared with Bug 2, since the leave point and the moving direction of My Bug is not limited to the m-line, the total path length of My Bug to reach the goal is always shorter than Bug 2.

Weaknesses: My Bug is also a greedy algorithm, and the leave point found is not necessarily the closest point to the goal. Therefore, it is not guaranteed that My Bug can find shortest path toward the goal after leaving the obstacle.