Student: Steve Gillet
Course: Algorithmic Motion Planning – ASEN 52540-001 – Fall 2024
Instructor: Morteza Lahijanian
Teaching Assistant: Yusif Razzaq
Date: September 6, 2024

Homework 1 Assignment Submission

Exercise 1.

Draw the trajectories produced by Bug 1, Bug 2, and Tangent Bug (with unlimited radius) algorithms for a point robot in the workspace shown in Figure 1.

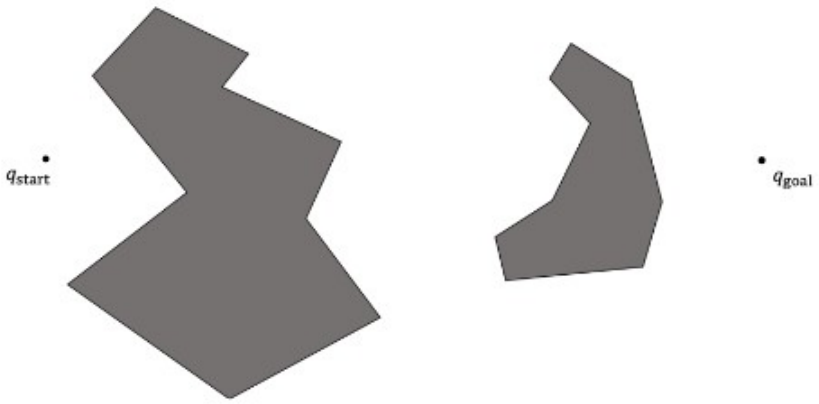


Figure 1: Simple environment.

Bug 1:

For Bug 1 (and for all of the bugs) I decided on a left-hand turn preference. You can see in the figure below that the robot had to circumnavigate both of the obstacles and probably came close to reaching the maximum distance for the algorithm (D + 3/2 Σ Pi) because it started on the far side of each obstacle and the closest points were on the opposite sides.

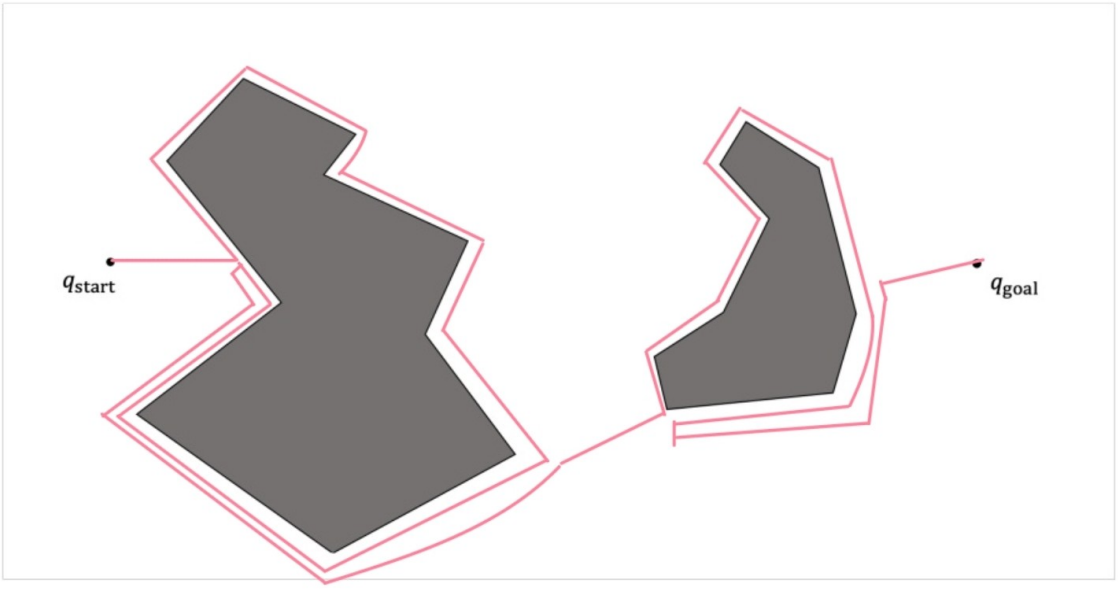


Figure 1: Simple environment.

Bug 2:

Bug 2 had a much easier time of it and was able to get back on the m-line in both cases before circumnavigating half of the obstacles. I suspect that Bug 2 works better than Bug 1 in the vast majority of cases because the type of obstacle where Bug 1 is better doesn't seem as likely to occur, Bug 2 deals with more likely obstacles perhaps.

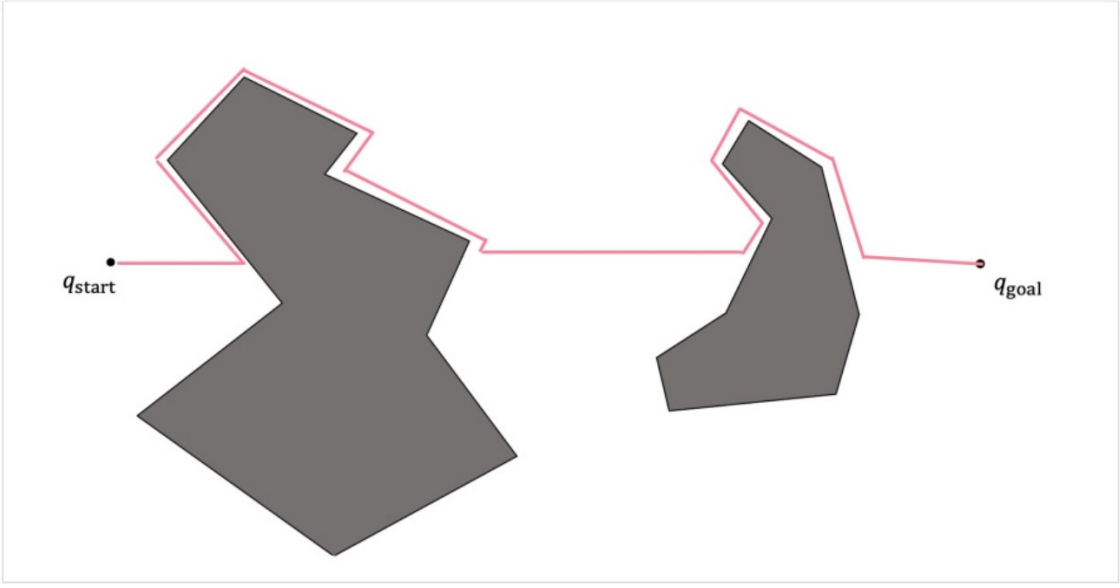


Figure 1: Simple environment.

Tangent Bug (Infinite Range):

Tangent Bug performed best of all because if it has infinite range it can effectively just see what point in the obstacle is closest to the goal and go straight there as soon as it comes into line of sight.

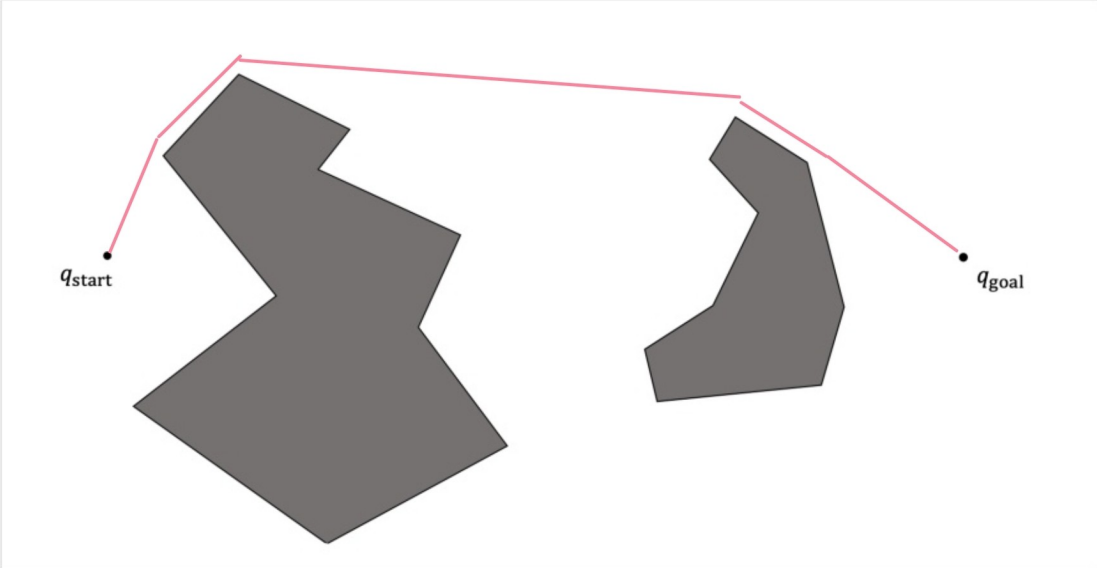


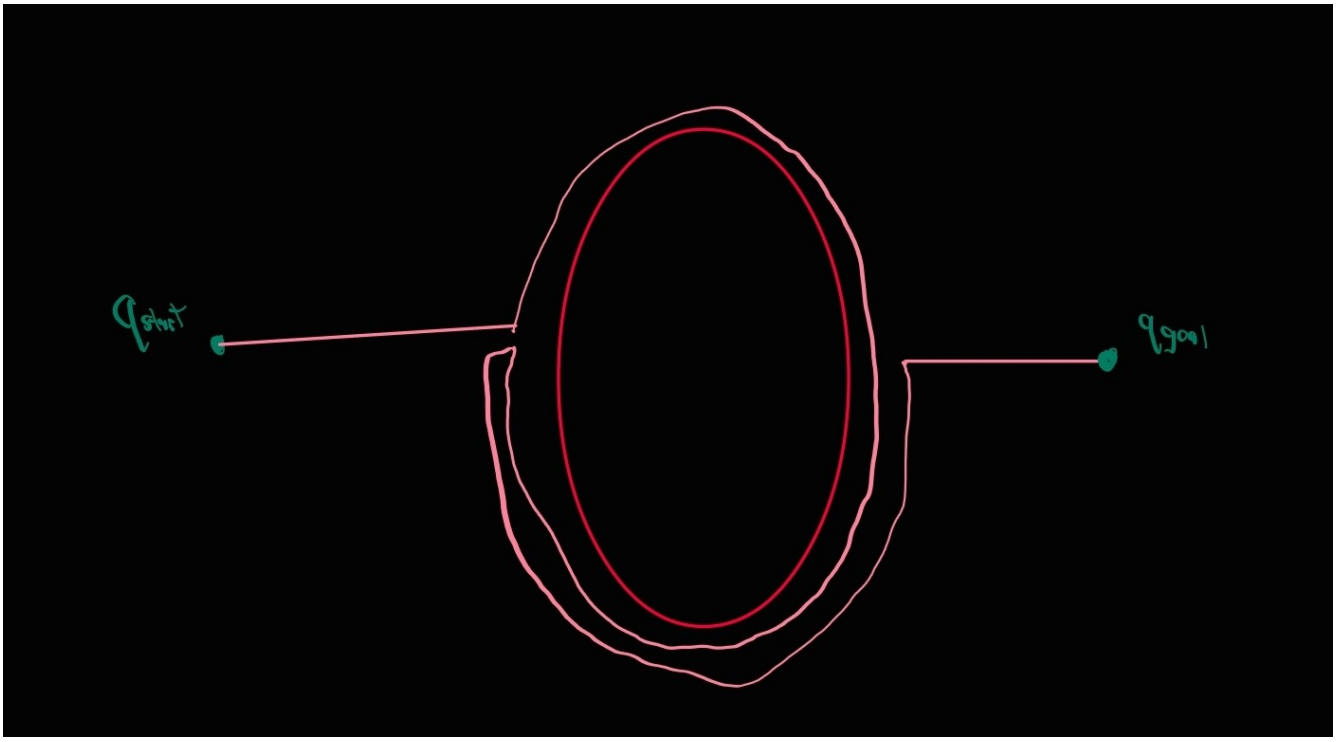
Figure 1: Simple environment.

Exercise 2.

Construct an example for which the upper bound of the traveled path for Bug 1 is obtained. How does Bug 2 perform in this example?

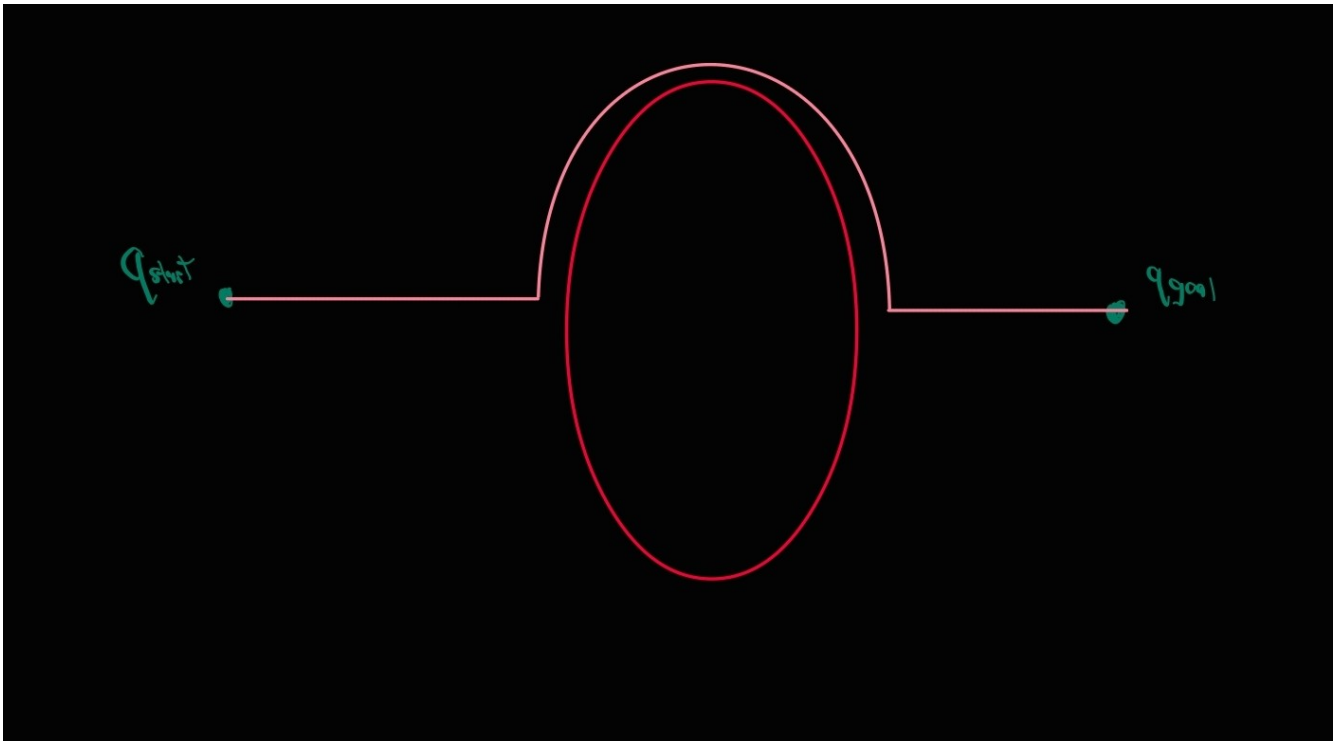
Bug 1:

The most obvious case in which Bug 1 will do bad is when the m-line crosses through the middle of a roughly symmetrical object. The robot will have to circumnavigate the entire obstacle and then go all the way back to the other side in order to continue. This contributes to the notion that Bug 1 suffers more with more simple obstacles.



Bug 2:

Bug 2 does an entire object perimeter less travel.



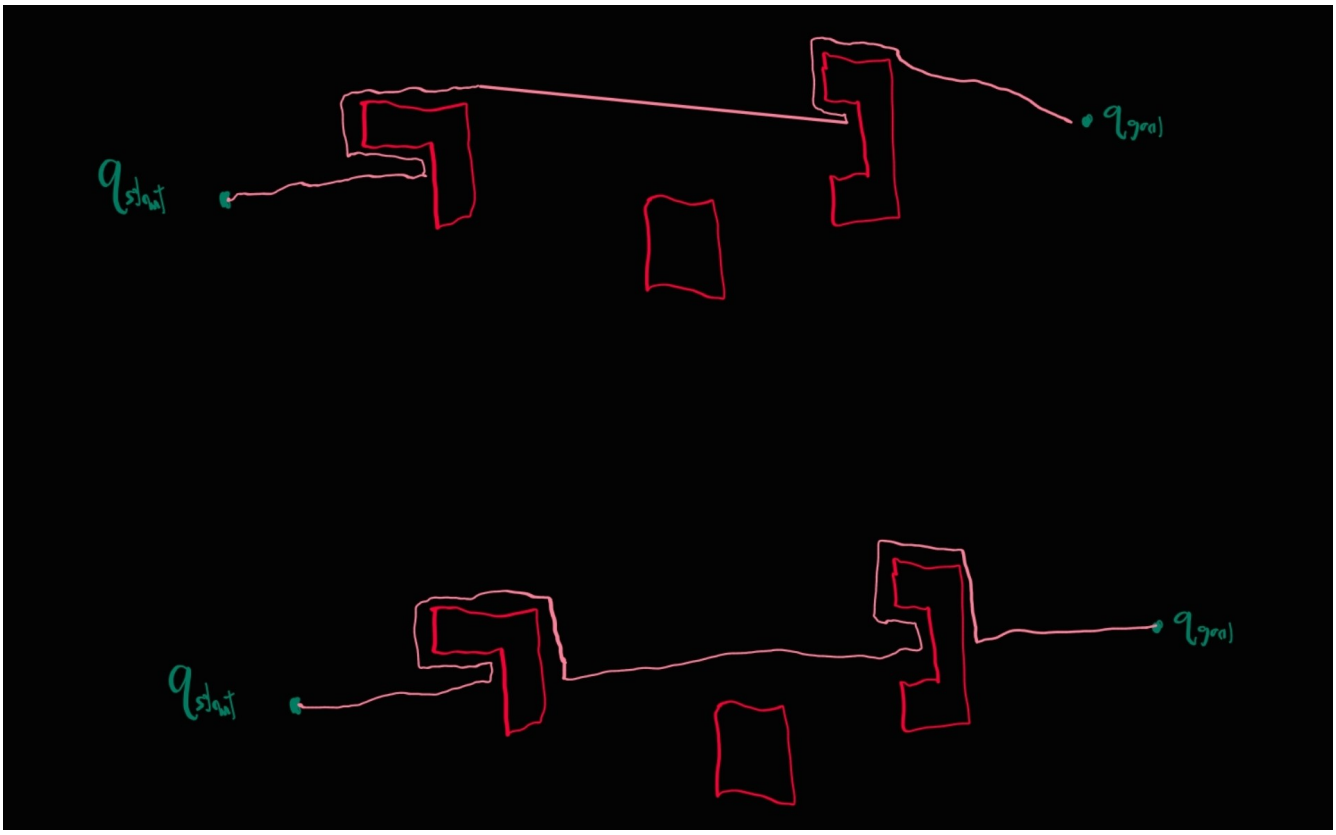
Exercise 3.

What is the difference between the Tangent Bug algorithm with zero range detector and Bug 2? Draw examples.

Tangent Bug (Zero Range) v. Bug 2:

Tangent Bug with zero range and Bug 2 don’t actually seem that different when you see the paths they take it’s just that Bug 2 leaves the obstacle when it gets back to the m-line whereas Tangent Bug leaves the obstacle when it can move towards the goal again or, if it had to do boundary-following, when it can move towards the goal again and it is closer to the goal than it was when it started the boundary-following. This probably reduces the distances traveled in most cases since it doesn’t have to do quite as much of the perimeter of the obstacles.

In the figure below you can see the Tangent Bug with zero range on top and the Bug 2 on the bottom and the little bit of difference between the two paths, the Tangent Bug being a bit shorter.



Exercise 4.

Consider a point robot at q_{start} with the goal of reaching q_{goal} in workspace W which consists of a set of obstacles $WO = \cup^{n}_{i=1} W\ O_i$, where $W\ O_i$ for all $i \in \{1, 2, \ldots, m\}$ $(m < n)$ is within the radius of $d(q_{start}, q_{goal})$ from q_{goal} and the rest of the obstacles are outside of this radius. What is the maximum number of obstacles the robot will encounter if it uses BUG 1 algorithm? Justify your answer.

Max m Obstacles:

Bug 1 algorithm will at most encounter m number of obstacles in this scenario. Conceivably all m of the obstacles within the radius $d(q_{start}, q_{goal})$ could be between the robot and the q_{goal} but there's no reason why a robot following Bug 1 would go outside of the radius and hit other obstacles out there. Even if an obstacle took the robot out of the radius (an obstacle that was partially inside and outside of the radius), the algorithm would still dictate that the robot follow the perimeter of that obstacle until it got back to the closest point to the q_{goal} which would necessarily have to be within the radius. There is nothing within that algorithm that would dictate the robot leaving that radius except following the perimeter of an obstacle but the algorithm also dictates that the robot would not leave that obstacle and therefore could not contact obstacles outside of the radius.

Exercise 5.

Is Bug 2 algorithm complete? Show a counter example or a proof.

Complete Proof:

I will go about proving that Bug 2 is complete using a similar method as Bug 1 was proven in class and that I've copied below:

"

Prove Bug 1 is complete:

• Suppose BUG1 were incomplete (proof by contradiction)

• 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 spends an infinite amount of time, or it terminates incorrectly

• Suppose it never terminates

• But each leave point is closer to the goal 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)

• Then there must be another intersection point on the boundary closer to object.

Since we assumed there is a path, we must have crossed this point on boundary

which contradicts the definition of a leave point.

“[1]

Using a similar tract I will prove by contradiction and suppose that Bug 2 is incomplete. If so, there is a path from start to goal with finite length, intersecting obstacles a finite number of times that Bug 2 does not find. Either it spends an infinite amount of time or it terminates incorrectly.

Never terminates: There are a finite number of points where the m-line passes through the obstacles therefore there must be a finite number of hit and leave points, since the algorithm dictates that the robot not leave from the same leave point twice the robot must eventually either leave and reach the goal or return to the hit point and terminate with failure.

Terminate incorrectly: In order to terminate incorrectly the robot must reach the hit point and terminate despite the fact that there was a leave point that it should have left from to get to the goal, however with a finite obstacle with finite intersections of the m-line, the robot must either hit all of the possible leave points or go back to the hit point and terminate correctly.

1 From Lecture 2 Bug Algorithms, Morteza Lahijanian