Motion Planning Project 1

MAE 195 Introduction to Robot Motion Planning and Navigation

Alex Nguyen, Chris Kangkorn, Colin Nisbet

Department of Mechanical and Aerospace Engineering
University of California Irvine
Instructor: Solmaz S. Kia
May 28th, 2021

Contents

1 Introduction

The goal of this project was to enable students to implement two basic motion planning algorithms: Bug 1 and the Decomposition and Search Method algorithms [1]. Here, we assume the robot to be a point-mass which traverses the environment shown in Figure 1. The devised algorithms are able to find a path from start to goal for any *randomly* initialized start and goal points in the free work space.

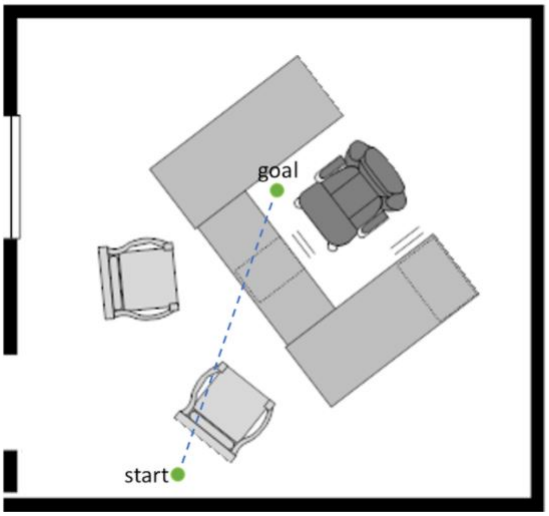


Figure 1: Environment with a line from a start point to a goal point.

The obstacles (i.e., chairs and desk) are approximated to be simple polygons with a finite number of edges and vertices. The polygon representation of the obstacles are used to create the simulated workspace. Also, an animation of the robot's trajectory path from one realization of a randomized start and goal point is shown for each respective motion planning algorithm.

2 Part 1: Bug 1 Algorithm

2.1 Background

The guide to simulating the Bug 1 algorithm was shown in E1.8 in [1]. The textbook outlines two different pseudocodes for implementing the sensor-based algorithm. An overview of the high-level Bug 1 algorithm and a detailed BugBase algorithm is shown in Figure 2.

The Bug 1 algorithm was developed by following the BugBase algorithm which contains detailed steps required to execute the Bug 1 algorithm. Although, a separate function was developed for circumnavigating obstacles since this pseudocode does not contain any logic for getting around obstacles.

Bug 1 algorithm

```
1: while not at goal :
2:     move towards the goal
3:     if hit an obstacle :
4:         circumnavigate it (moving to the left or right is unimportant). While circumnavigating, store in memory the minimum distance from the obstacle boundary to the goal
5:         follow the boundary back to the boundary point with minimum distance to the goal
```

(a) High Level Overview

BugBase algorithm

Input: Two locations `start` and `goal` in W_{free}, a list of polygonal obstacles `obstaclesList`, and a length `step-size`

Output: A sequence, denoted `path`, of points from `start` to the first obstacle between `start` and `goal` (or from `start` to `goal` if no obstacle lies between them). Successive points are separated by no more than `step-size`.

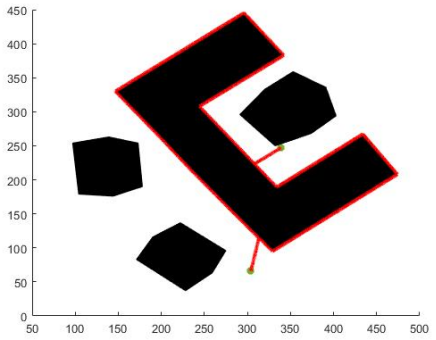
```
1: current-position := start
2: path := [start]
3: while distance(current-position, goal) > step-size :
4:     compute candidate-current-position by taking a step of length step-size towards goal
5:     compute distance from candidate-current-position to each obstacle
6:     if distance from candidate-current-position to closest obstacle < tolerance :
7:         return "Failure: There is an obstacle lying between the start and goal" and path
8:     current-position := candidate-current-position
9:     path := [path, current-position]
10: path := [path, goal]
11: return "Success" and path
```

(b) Detailed Overview

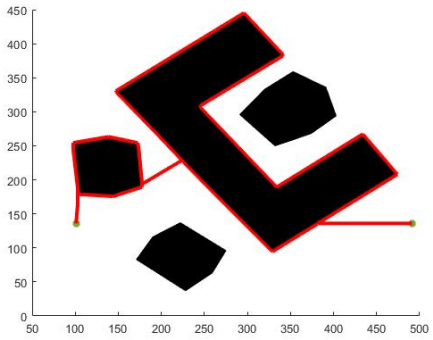
Figure 2: Bug 1 algorithm pseudocode

2.2 Simulated Results

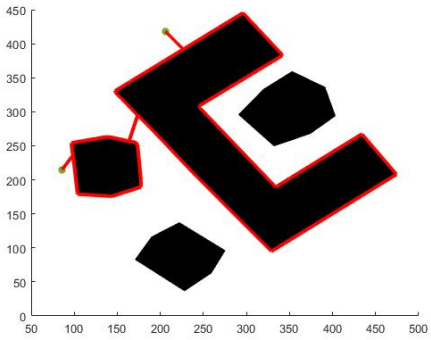
The navigated path of the point mass robot shown in Figure 3 represents three different realizations of a randomized start point and goal point for our motion planning solution. An animation for the path shown in Figure 3(c) is included with the report.



(a) Randomized path 1



(b) Randomized path 2



(c) Randomized path 3

Figure 3: Examples of the Bug 1 algorithm performing for randomized start and goal points

3 Part 2: Decomposition and Search Method

3.1 Background

The guide to simulating the decomposition and search method in chapter 2 [1] is outlined as follows.

sweeping trapezoidation algorithm

Input: a polygon possibly with polygonal holes
Output: a set of disjoint trapezoids, whose union equals the polygon
1: initialize an empty list $\mathcal{T}$ of trapezoids
2: order all vertices (of the obstacles and of the workspace) horizontally from left to right
3: **for** each vertex selected in a left-to-right sweeping order :
4: extend vertical segments upwards and downwards from the vertex until they intersect an obstacle or the workspace boundary
5: add to $\mathcal{T}$ the new trapezoids, if any, generated by these segment(s)

(a) Sweeping Trapezoidation Algorithm

roadmap-from-decomposition algorithm

Input: the trapezoidation of a polygon (possibly with holes)
Output: a roadmap
1: label the center of each trapezoid with the symbol $\diamond$
2: label the midpoint of each vertical separating segment with the symbol $\bullet$
3: **for** each trapezoid :
4: connect the center to all the midpoints in the trapezoid
5: **return** the roadmap consisting of centers and connections between them through midpoints

(b) Roadmap-from-Decomposition Algorithm

planning-via-decomposition+search algorithm

Input: free workspace W_{free}, start point p_{start} and goal point p_{goal}
Output: a path from p_{start} to p_{goal} if it exists, otherwise a failure notice. Either outcome is obtained in finite time.
1: compute a decomposition of W_{free} and the corresponding roadmap
2: in the decomposition, find the start trapezoid Δ_{start} containing p_{start} and the goal trapezoid Δ_{goal} containing p_{goal}
3: in the roadmap, search for a path from Δ_{start} to Δ_{goal}
4: **if** no path exists from Δ_{start} to Δ_{goal} :
5: **return** *failure* notice
6: **else**
7: **return** path by concatenating:
the segment from p_{start} to the center of Δ_{start},
the path from the Δ_{start} to Δ_{goal}, and
the segment from the center of Δ_{goal} to p_{goal}.

(c) Planning-via-Decomposition-and-Search Algorithm

Figure 4: Decomposition and Search algorithm pseudocode

First, the workspace must be decomposed into convex subsets, i.e., trapezoidation of a polygon by decomposing it into a collection of trapezoids. The pseudocode for this algorithm is shown in Figure 4(a). The decomposition requires classifying each polygon vertex as one of the following types shown in Figure 5.

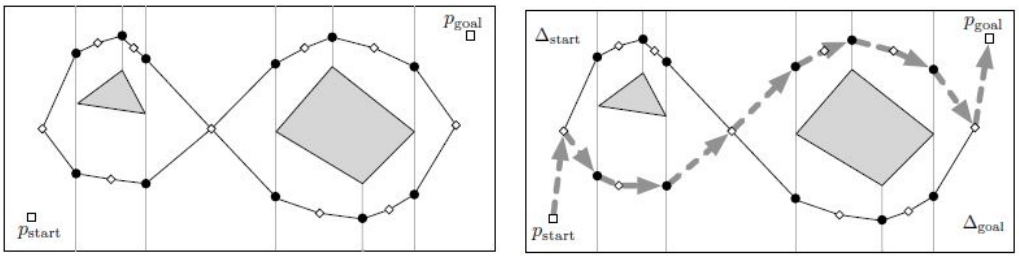
Vertex Type	Vertex as Endpoint of Two Segments	Vertex as Convex or Non-Convex	Example
(i)	left/left	convex	p_6 and p_8
(ii)	left/left	non-convex	p_3
(iii)	right/right	convex	p_2 and p_4
(iv)	right/right	non-convex	p_7
(v)	left/right	convex	p_1
(vi)	left/right	non-convex	p_5

Figure 5: Trapezoidal decomposition algorithm vertex types

Second, the decomposed workspace must be transformed into a roadmap. Here, we assume that we can compute an easy-to-navigate *roadmap*. The pseudocode for this algorithm is shown in Figure 4(b). The obtained roadmap will contain (1) a collection of center points (one for each trapezoid) and (2) a collection of paths connecting center points (each being composed of 2 segments, connecting a center to a midpoint and the same endpoint to a distinct center). An example of this roadmap in the free workspace can be shown in Figure 6(a).

Finally, we consider the motion planning problem in the free workspace derived from the decomposed convex subsets and roadmap. The solution to the problem will be found by utilizing the planning-via-decomposition-and-search algorithm. The pseudocode for this algorithm is shown in Figure 4. An example of this motion planning solution from a roadmap is shown in Figure 6(b).

In general, this method will not find the shortest path in terms of distance, but it will find the shortest path in terms of number of edges since the edges are not weighted.



(a) Roadmap in free workspace

(b) Roadmap solution via decomposition-and-search algorithm

Figure 6: An example roadmap for a free workspace with a motion planning solution

By means of decomposition, we have transformed a continuous planning problem into a discrete planning problem. If a solution exists, we can find a path in the free workspace!

3.2 Simulated Results

There are two parts to finding our motion planning solution. First, we must discretize our workspace then form a road map. Secondly, we must apply the decomposition and search algorithm to find a path which minimizes the number of nodes in the workspace.

The trapezoidal decomposition algorithm produced the disjointed polygons shown in Figure 7. Notice, the union of all the colored polygons defines the free workspace for our environment.

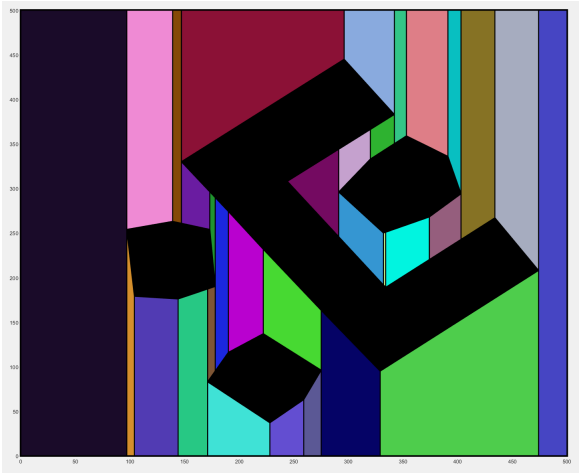


Figure 7: Trapezoidal decomposition of workspace

The roadmap algorithm utilized our previous trapezoidal decomposition allowed us to construct by constructing the roadmap based on all trapezoid centroids and line midpoints as nodes. The roadmap for our problem can be seen in Figure 8 where the adjacency list was constructed based on the numbered nodes. Keep in mind, the key to finding our motion planning solution is to ensure we have a well defined adjacency list.

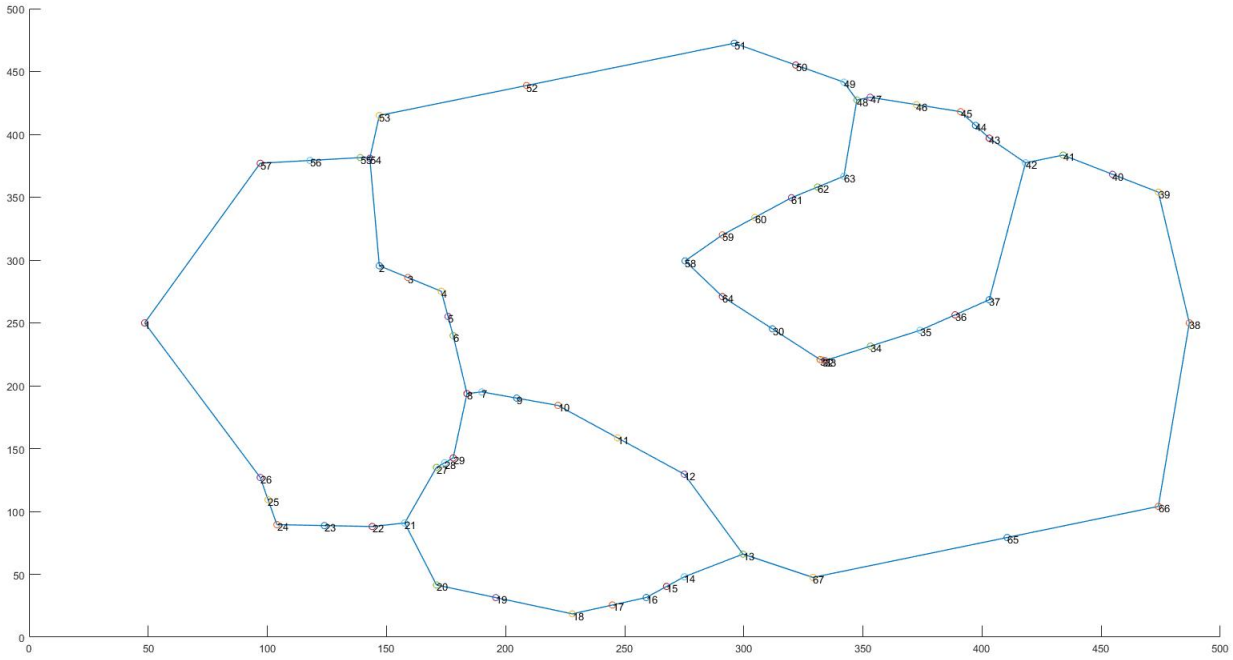
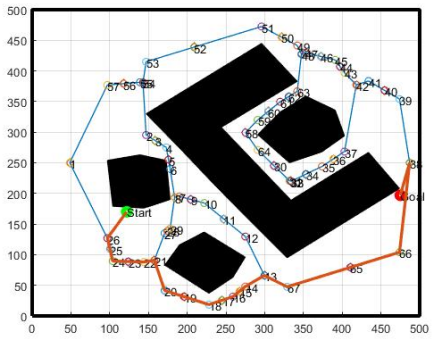
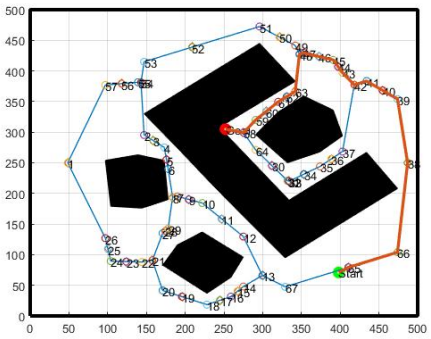


Figure 8: Roadmap including labeled nodes

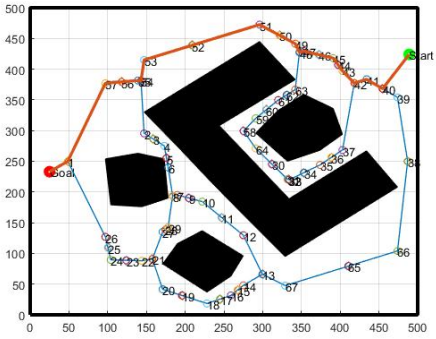
Finally, we are ready to apply the decomposition and search algorithm to find the motion planning solution. The resulting paths for a point mass robot is shown in Figure 9. These paths represent three different realizations of a randomized start point and goal point for our motion planning solution. An animation for the path shown in Figure 9 is included with the report.



(a) Randomized path 1



(b) Randomized path 2



(c) Randomized path 3

Figure 9: Examples of the Decomposition and Search algorithm performing for randomized start and goal points

4 Conclusion

Overall, this was an informative project which helped develop our robotic motion planning intuition and coding skills. One major improvement for this algorithm is to follow Dijkstra's algorithm and assigns weights to all the edges which will find the shortest path from start to goal. In the future, we may find ourselves applying these algorithms in real-time which will raise the question: Will the experiment match the simulated results?

References

[1] F. Bullo and S. L. Smith, *Lectures on Robotic Planning and Kinematics*, 2020.