Autonomous Robotics
Lab 4 - Topological Mapping - Visibility Graph

Anirudh Puligandla

May 1, 2017

1 Introduction

The rotation plane sweep algorithm is a path planning algorithm based on topological maps. It is one of the most widely used algorithm for path planning of autonomous robots. The following sections explain the algorithm, its implementation in MATLAB and, lastly, the results on a few sample environments.

2 Environment

A proper environment, with well defined obstacles and boundaries is necessary to test the algorithm. Different cases were considered while building environments based on different shapes of polygons. The coordinates of the vertices of the obstacles were defined in a variable *vertices*. The variable was defined in the same format as mentioned in the instructions. The following figure shows a couple of examples for the environment.

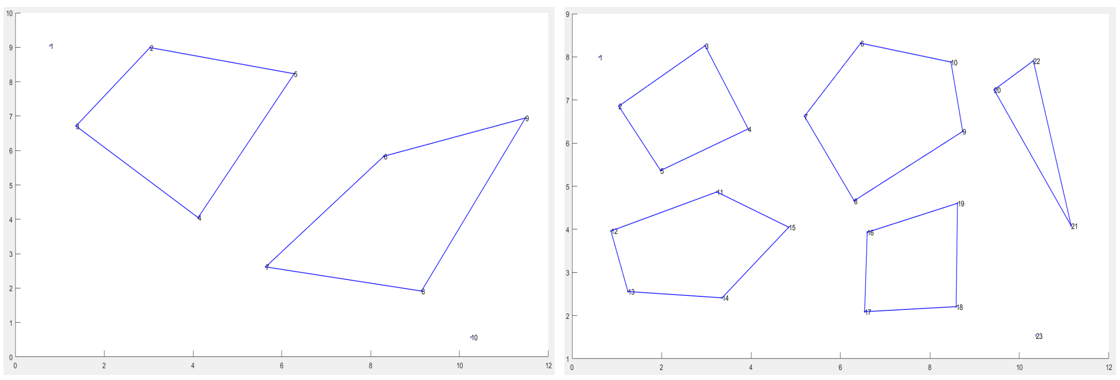

Figure 1: Different environments for algorithm testing

3 Rotational Plane Sweep (RPS) algorithm

The basic principle of the algorithm is to output all the visible vertices form each of the vertices. A list of visible edges, from the start position, is output in the variable $edges$. The instructions given in the lab manual were followed while building the algorithm. The attached $RPS.m$ file contains a function, with the same format as proposed in the instructions. The following subsections describe, in detail, the implementation of the algorithm.

3.1 Compute Edges of all polygons

The first step is to obtain all the edges of polygons in the graph. We already know the coordinates of all the vertices in order. Assuming we know a vertex $v1$ with the coordinate $(x1, y1)$ and next vertex $v2 = (x2, y2)$ in the list. If $v2$ belongs to the same polygon as $v1$, we simply build the edge $[x1, x2, y1, y2]$. However, if $v2$ belongs to the other polygon, which means $v1$ is the last element of the polygon, we need to connect it to the first one of the polygon. Our approach is finding the number n of vertices of the current polygon and then figuring out the index of the first element of the polygon by subtracting $(n1)$ to the $v1$ index.

3.2 Angles between horizontal line and vertices

For each element in the vertices list, we need to compute angles of the line segments between the current vertex and all the vertices. Because we rotate our sweep line in the anti-clockwise direction, we need to sort the angles in the ascending order so that we know the pausing order of the line segment. Figure 2(a) shows an example of the angle computations.

3.3 Intersections

This part is the core of the algorithm, which decides whether or not to put the edge between two vertices into the visible list. Once we acquire the line segment between the current vertex and the other vertex, we can check its intersection with all the polygon edges one by one. If there is no intersection, which means the two lines have no connection, we simply continue to check next polygon edge. If the intersection between two lines exists, there are two invalid cases:

- Intersect on the polygon edge line, not vertex (line across 3 and 6 in Fig. 2 (b)).
- Intersect on a vertex, but the line segment lies in the polygon (line across 3 and 5 in Fig. 2(b)).

For the first case, we simply check if the intersection point is in the given vertices list. For the second one, at first, we check if the intersection vertex belongs to the same polygon as the current vertex. If yes, we think the line segment lies in the polygon. After eliminating the invalid cases, if we find there are two intersections between the line segment and the polygon edge, the line segment is exactly the polygon edge and we also add it to the edge list.

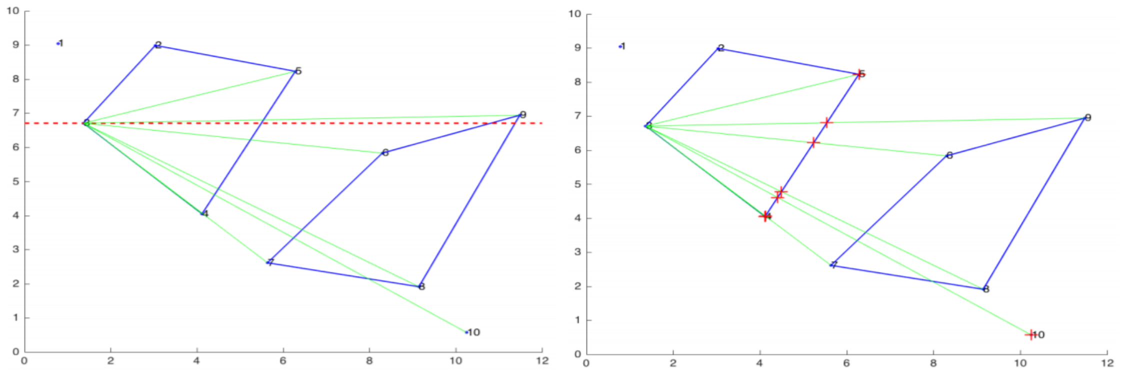

Figure 2: From Left: (a) angle computation example; (b)Invalid intersections

4 Testing

4.1 Start position from all the vertices

The algorithm was first supposed to be tested to compute the visibility of the starting point with respect to all the vertices and goal point. Figure 3(a) shows the result of the RPS algorithm for the same. In the figure, we can see that the start point is visible from only 3 vertices of the polygon. *Point*6 in the figure shows the goal point, while *point*1 is the start position

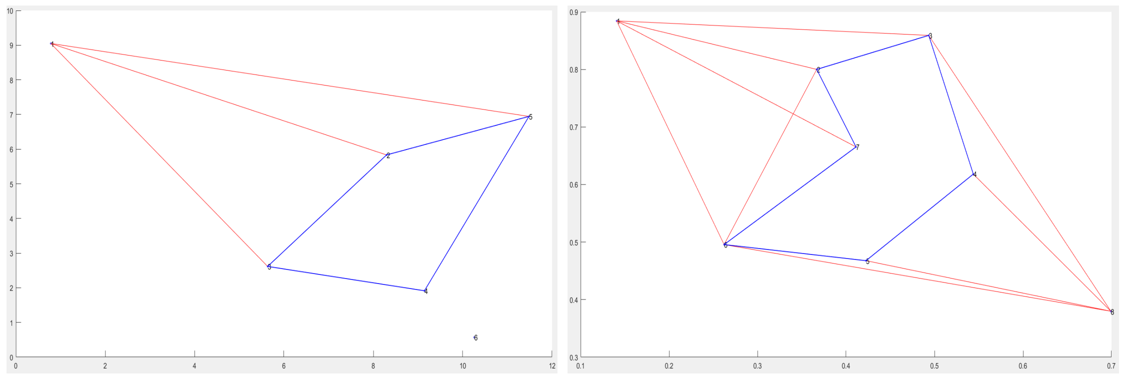

Figure 3: From Left: (a) visibility of starting point from all the vertices; (b)Incorrect result for concave polygon

4.2 For a polygon vertex

Next, the algorithm was supposed to be tested with polygons having concave edges. Figure 4(a) shows the result for concave polygons. There was a **problem** for this part (shown in Figure 3(b)). The algorithm also connects the vertices from the same polygon and considers them as an edge. The problem was rectified by considering the edges between the current vertex and the vertices that are not adjacent to the current vertex as invalid.

4.3 For all vertices

Lastly, the algorithm was to be tested for all the vertices of all the polygons for a given environment. An environment consisting of 2 polygons was provided in the instructions for testing purposes. The algorithm successfully computed all the visible edges from each of the vertices and the start and goal points. Figure 4(b) shows the result for the provided environment. Figure 5 below, shows the result for a larger environment consisting of 6 polygons, that was provided with the instructions.

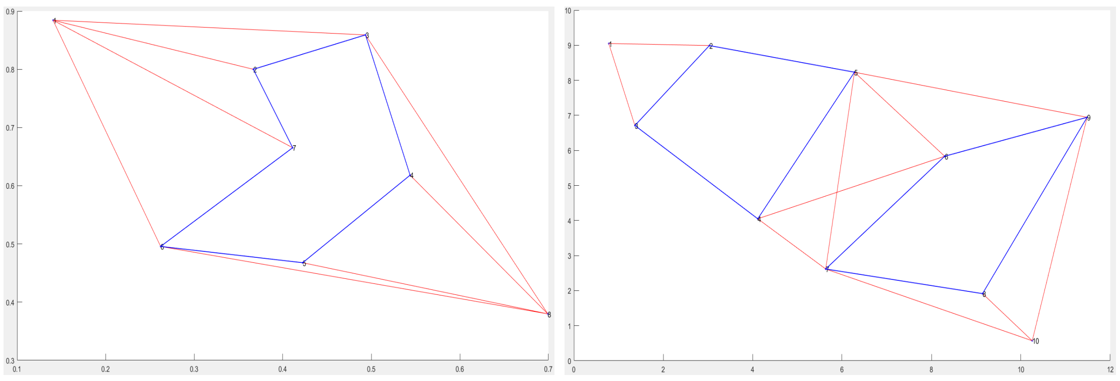

Figure 4: From Left: (a) corrected result for concave polygon; (b)result for given environment consisting of two polygons

5 Conclusion

The RPS algorithm was successfully implemented and tested for the provided environments, while considering different cases for convex or concave polygons, and the results were clearly depicted. The algorithm was implemented in the same format as mentioned in the instructions.

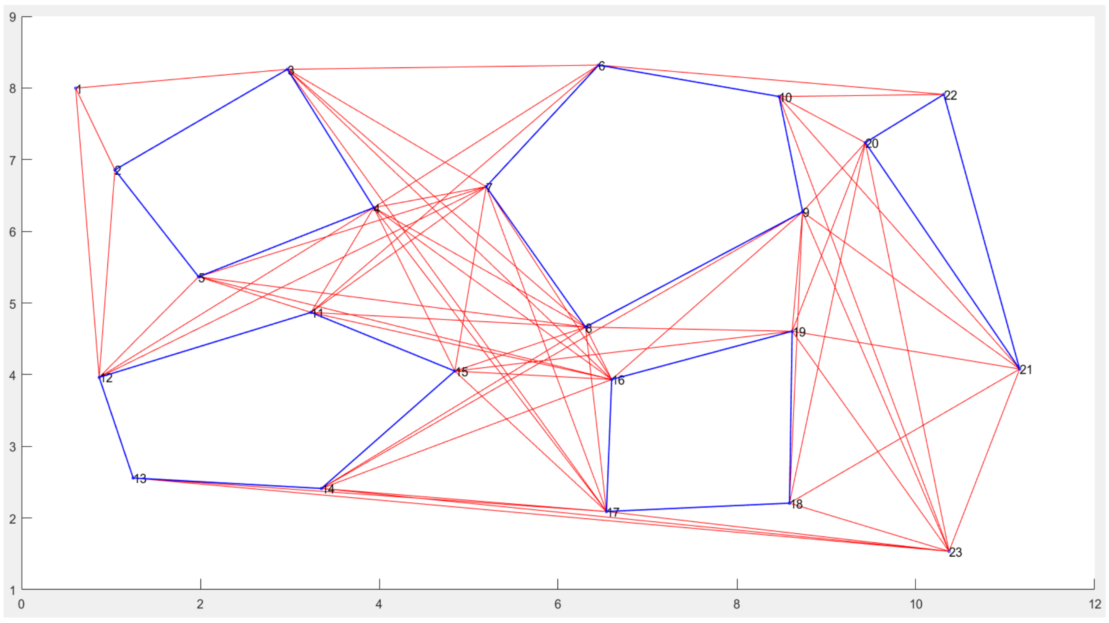

Figure 5: Result for larger environment