Lab 2 - Rotational Plane Sweep Algorithm

Songyou Peng
psy920710@gmail.com

I. INTRODUCTION

This report will mainly discuss the implementation details of the rotational plane sweep (RPS) algorithm. RPS can help build a visibility graph of the environment, which can be utilised for path planning of robotics. Throughout the discussion, we will also talk about some problems we met and how we solved them.

II. IMPLEMENTATION

First, let us briefly explain the main idea of RPS. Starting with a certain vertex, a line is swept in the anti-clockwise direction. This line crosses the plane of the graph, pausing at the vertices of the polygon (obstacles) [1]. In this way, we can find the visibility of the current vertex to other vertices. If the line segment between the two vertices intersects in the edge of the obstacles or crosses the obstacles, we say two vertices are invisible, otherwise, we add the line segment to the visible list. In the following parts, we will discuss the details of our MATLAB implementation.

A. Obtain edges of all the polygon

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, but we need a few more steps to get the edges. Assuming we know a vertex v_1 with the coordinate (x_1, y_1) and next vertex $v_2 = (x_2, y_2)$ in the list. If v_2 belongs to the same polygon as v_1, we simply build the edge $[x_1, x_2, y_1, y_2]$. However, if v_2 belongs to the other polygon, which means v_1 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 $(n-1)$ to the v_1 index.

B. Calculate the angles of line segments

For each element in the vertices list, we need to compute angles of the line segments between the current vertex and all the following vertices (angle difference between the green lines and the red dash line in Fig. 1). 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.

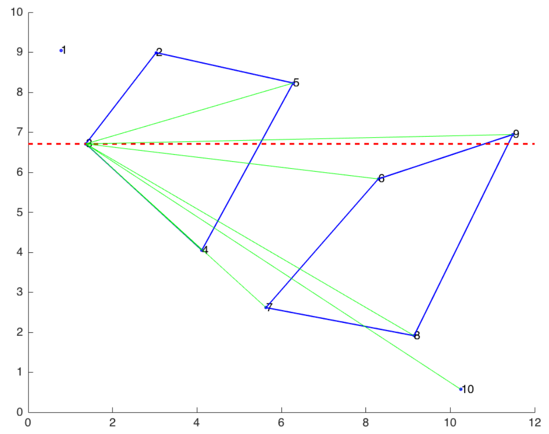


Fig. 1. Acquire angles of the line between the current vertex to the following vertices

C. Check the 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:

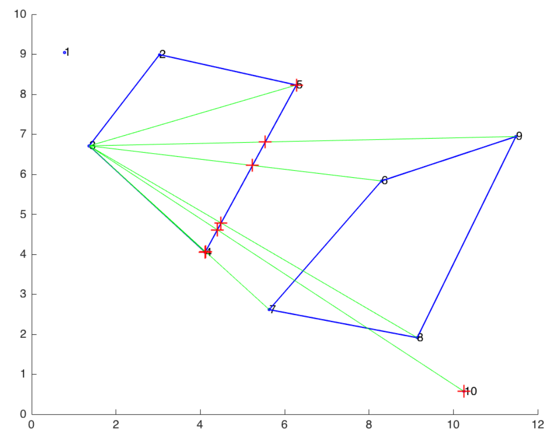


Fig. 2. Illustration for the intersection between line segment and polygon edges

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

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.

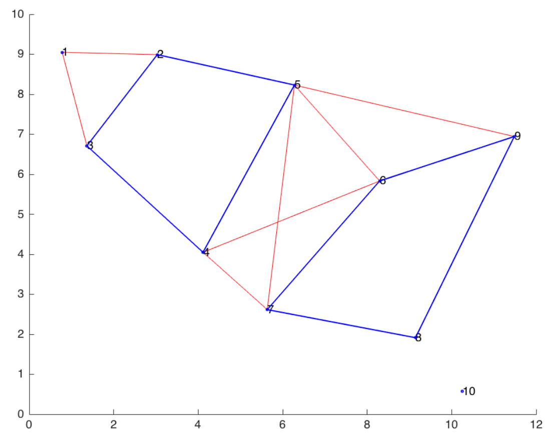
After eliminating the invalid cases, if we find there are two intersections between the line segment and the polygon edge, the line segment is exactly the polygon edge and we also add it to the edge list.

The last thing is to verify if the intersection is one of the vertices. If so, the goal point will not be connected and the graph we acquire is shown in Fig. 3(a). What we need to do is simply checking if the intersection is the current vertex while the connecting point is the goal point.

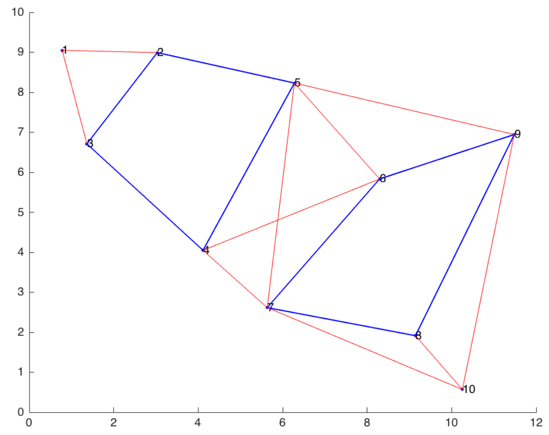
Fig. 3(b) and Fig. 4 are the final results of RPS algorithm in the environment of convex polygons.

D. *Non-convex polygon*

As mentioned in II-C, the way we deal with the situation that the line segment lies within the polygon is to check if two vertices belong to the same polygon. This works fine for the map



(a) Only check if two vertices belong to the same polygon



(b) Check the middle point position of the line segment

Fig. 3. Illustrations for checking the goal points separately

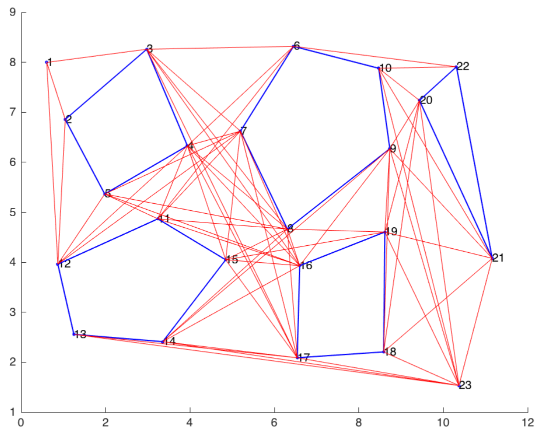
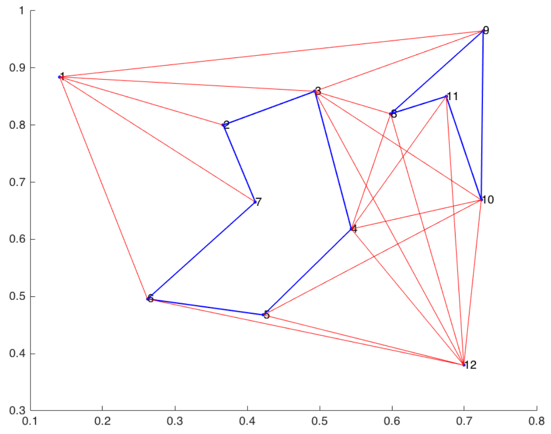
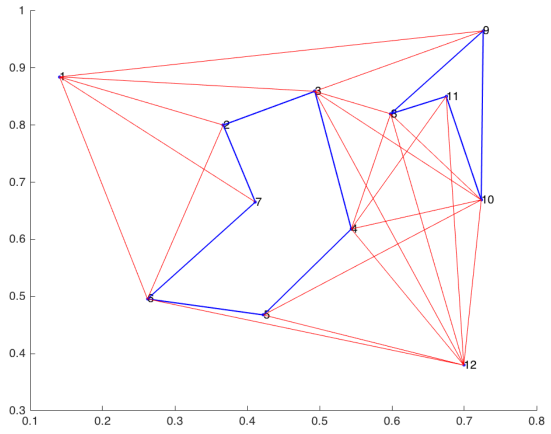


Fig. 4. The result for the big convex map

with all convex polygons, but may fail for non-convex polygons. In the Fig. 5, we can notice that actually vertex 2 and 6, 8 and 10 should be connected. However, as we only check if two vertices belong to the same polygon, which cannot deal with the non-convex cases, we come up with another idea. We can calculate the middle position of the line segment between the current vertex and the other vertex, and then use MATLAB function `inpolygon` to check if this point is in the polygon. If yes, we know the line segment lies within the polygon, otherwise it is a non-convex case and the edge is valid. The result of the new method is shown in the Fig. 5(b).



(a) Only check if two vertices belong to the same polygon



(b) Check the middle point position of the line segment

Fig. 5. Illustrations for the non-convex polygon results using two different methods

III. Conclusions

In this lab, we implemented the rotational plane sweep algorithm, which was simple but very useful. We tested the algorithm in both convex and non-convex environment and it worked well.

However, the RPS algorithm has one main limitation. Because the main step of the algorithm is to check if the intersections are the vertices, it is only suitable for the environment which has the polygon-like obstacles. For example, if the obstacles are similar to cylinders, the algorithm will not work.

References

[1] Choset, Howie M. Principles of robot motion: theory, algorithms, and implementation. MIT press, 2005.