Autonomous Robots
Lab 2 Report: Topological Maps -Visibility Graph

Mohammad Rami Koujan
M.Sc. VIBOT
University of Girona

March 29, 2016

1 Introduction and problem definition

The purpose of this lab assignment is to be familiar with robot path planning based on topological maps. In particular, plane sweep algorithm is a very powerful approach for building the visibility graph in a total time complexity of $n^2 * logn$. The idea behind algorithms of this type is to imagine that a line is swept or moved across the plane, stopping at some points. This algorithm can be summarized as follow:

- Input. A set N of polygonal obstacles and a point p that does not lie in the interior of any obstacle.
- Output. The set of all obstacle vertices visible from p.

```
1. Sort the obstacle vertices according to the clockwise angle that the half line from p to each vertex
   makes with the positive x-axis. In case of ties, vertices closer to p should come before vertices
   farther from p. Let $\alpha_1, . . . ,\alpha_n$ be the sorted list.
2. Let $\rho$ be the half-line parallel to the positive x-axis starting at p. Find the obstacle edges that are
   properly intersected by $\rho$, and store them in the S list in the order in which they are intersected
   by $\rho$.
3. $VG \longleftarrow \phi$
4. for $i \longleftarrow$ 1 to n
5.     do if VISIBLE($\alpha_i$) then Add $\alpha_i$ to $VG$.
6.     Insert into S the obstacle edges incident to $\alpha_i$ that lie on the clockwise side of the half-line
   from p to $\alpha_i$.
7.     Delete from S the obstacle edges incident to $\alpha_i$ that lie on the counter-clockwise side of the
   half-line from p to $\alpha_i$.
8. return $VG$
```

2 Rotational Plane Sweep algorithm Implementation

This section presents the different steps of implementation of the rotational plane sweep algorithm in Matlab program. Initially,a number of polygons has to be generated with a start and goal points. In order to do so, the function "ginput" is used to read the x and y coordinates of the chosen points on the opened figure. For Each polygon, all the points are selected by using the left click of the mouse except the last point which should be chosen by the right click, to indicate the end of a polygon. Furthermore, the start and goal points should be clicked first and last respectively. Figure 1 shows one possible example of generating polygons using the described script, where point 1 and 22 represent the start and goal points respectively. The Cartesian coordinates of each point are stored in another variable called "vertices" along with their polygon number in the third column, start point has 0 as its polygon number and last point has the final polygon number plus one.

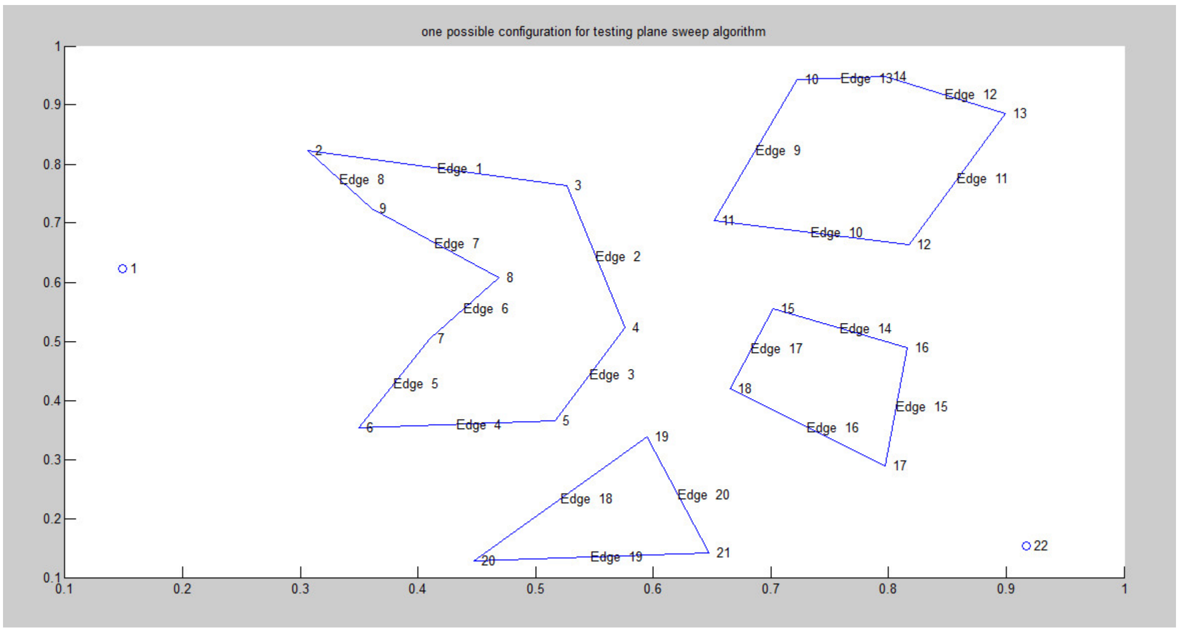

Figure 1: set of generated polygons

In the second step, the visibility of the starting point with respect to all the vertices and goal point should be computed. This is accomplished by applying the previously explained steps about the rotational plane sweep algorithm.

- The programmed function, in fact, starts by building the obstacles' edges list, OE, and plotting the environment. This is done by checking the third column of the vertices parameter sequentially to find connected edges taking into account that last vertex of each polygon is connected back to the first vertex of the same polygon. Consequently, OE variable contains the pairs of vertices that constitute edges. Therefore, the number of rows of this variable is equal to the total number of edges in the studied configuration and the number of columns equals to two.

- Thereafter, a loop iterates through each vertex doing three main tasks. First, calculating the angle between each vertex and the start one using their coordinates and the atan2() function and the result is stored in alpha list. Second, computing the distance between each edge and the start point and storing the result in a new column in OE variable. Third, finding all the edges that intersect the horizontal line going outwards from the start point and store the result in the S list. Finally, the s and alpha list are sorted in an ascending order.

- In the final stage of this program another "for" loop is used to go through each vertex in the alpha list to do the following:

    1. determining the visibility of the current vertex in alpha list: by making sure that the S list is empty, no edges blocking the visibility of the start point now, or the current alpha vertex is part of the first edge, the closest one, in the sorted S list or it is closer to the start point than the first edge. In the mentioned cases, the current alpha vertex is deemed visible to the start point. Otherwise, it is not visible.
    2. Updating the S list: This is performed by getting the two edges connected to the vertex under study and look for them in the S list. If any of them is inside S list, they will be deleted because they are passed by the half line emitting from the start point. Otherwise, the missing edge is added to this list.
    3. Sorting the updated S list: according to their distance from the start point.

Figure 2 shows an example of computing the visibility of the start node in two different environments. In figure 3 the algorithm is tested with a convex polygon and the visibility of the start and goal points is calculated.

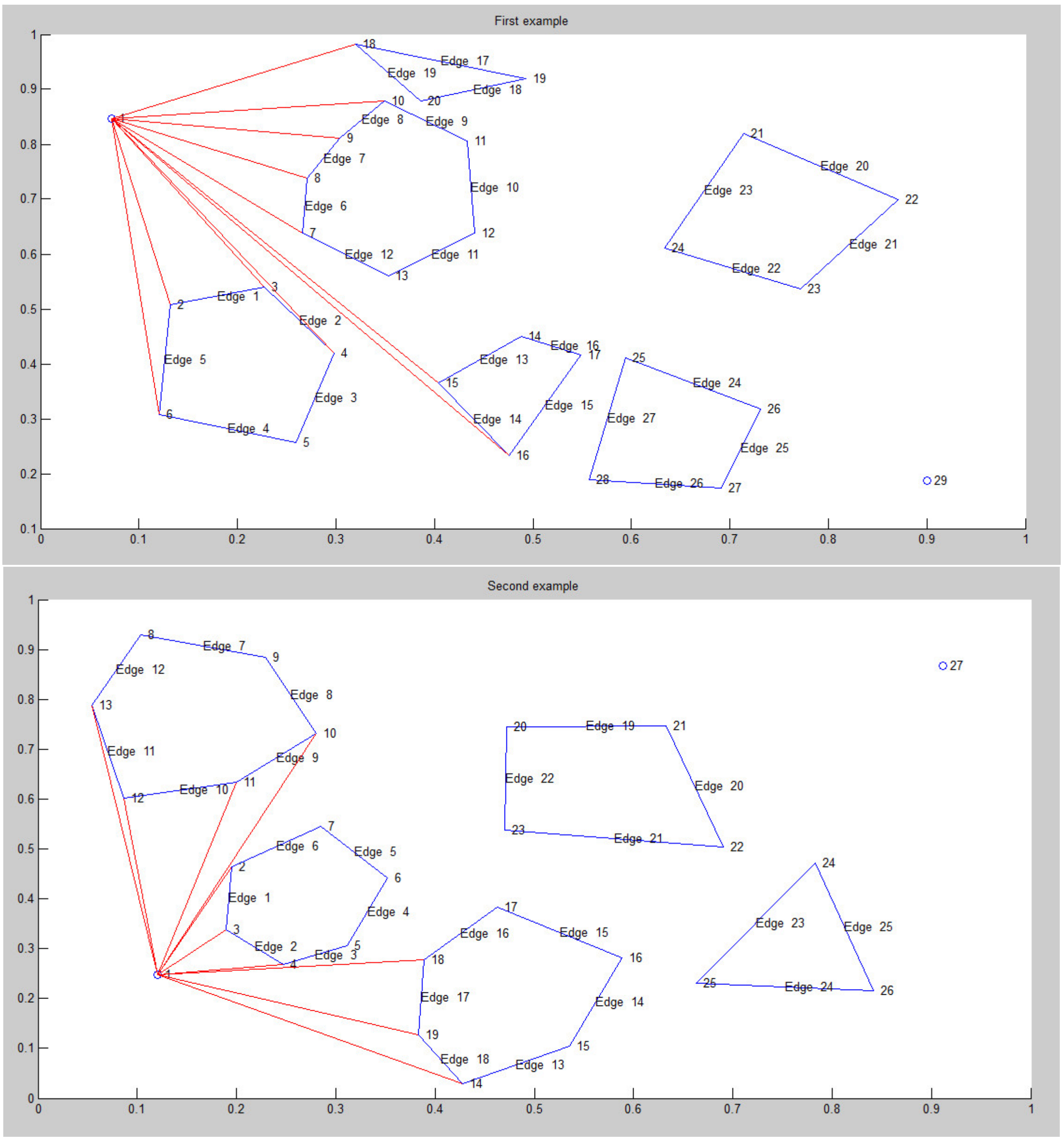

Figure 2: Finding visible nodes to the start one in two distinct environments

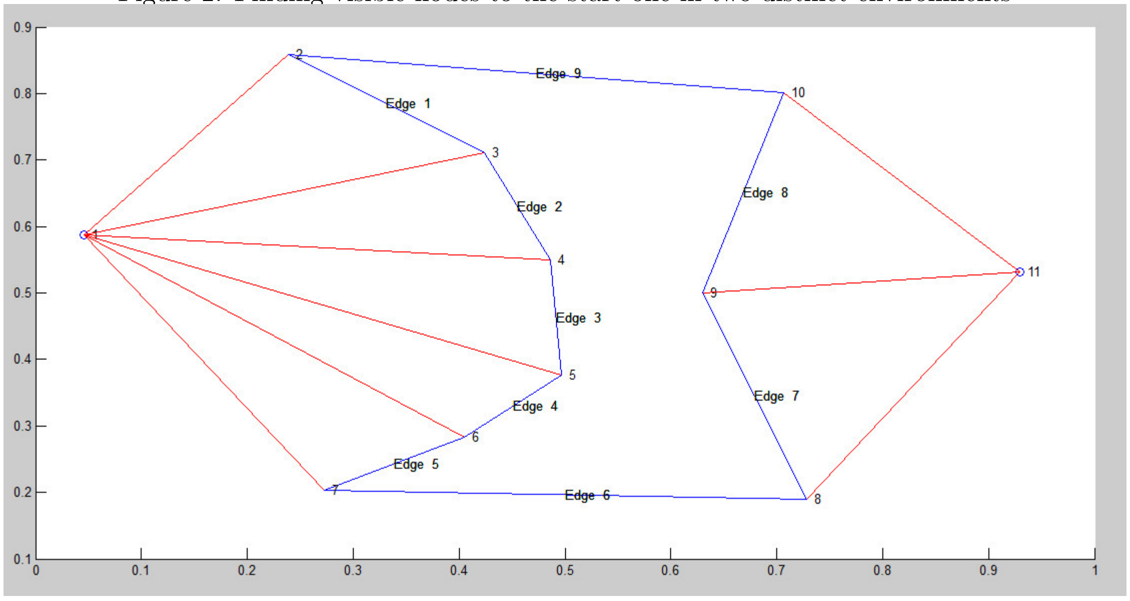

Figure 3: testing convex polygon for visibility by the start and goal points

In the final stage of this assignment the algorithm is applied in all vertices so that it generates the final visibility graph. TO perform this task, a number of changes to the last version of the program should be done:

1. putting the entire program in a "for" loop to repeat the procedure for every vertex stored in the "vertices" variable and for every iteration new vertex is chosen as a start point.
2. to take into account vertices of polygons when they are considered as start points, any vertex of the same polygon is not considered visible except if it is on the same edge as the start point or the line connecting the start and this vertex does not intersect their polygon.
3. edges that have the start point as their first or second vertex are not inserted in the S list.

Figure 4 shows the flowchart that explains the implemented rotational plane sweep algorithm in details.

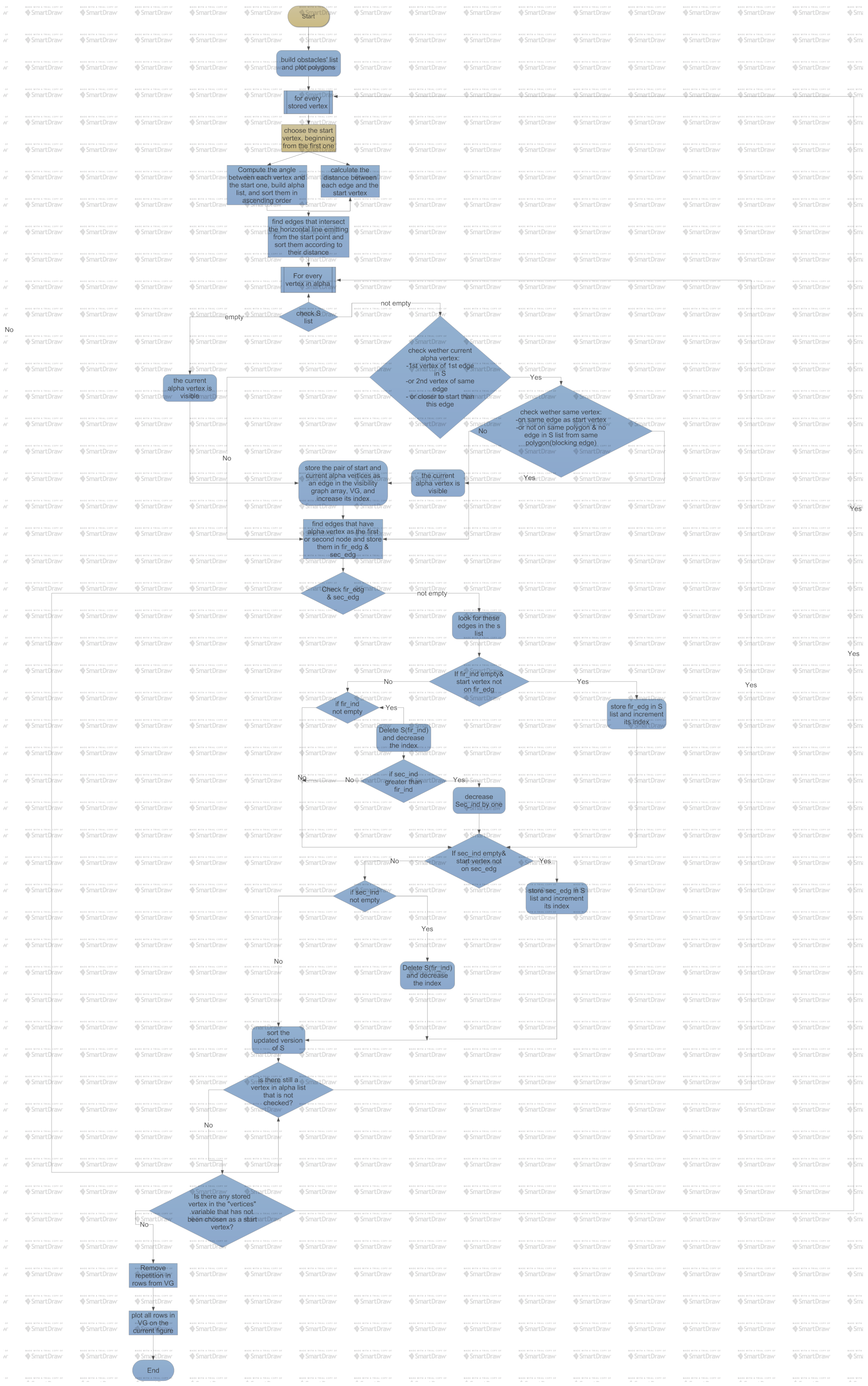
Start
build obstacles' list and plot polygons
for every stored vertex
choose the start vertex, beginning from the first one
Compute the angle between each vertex and the start one, build alpha list, and sort them in ascending order
calculate the distance between each edge and the start vertex
find edges that intersect the horizontal line emitting from the start point and sort them according to their distance
For every vertex in alpha
check S list
empty
not empty
check wether current alpha vertex: -1st vertex of 1st edge in S -or 2nd vertex of same edge - or closer to start than this edge
Yes
check wether same vertex: -on same edge as start vertex -or not on same polygon & no edge in S list from same polygon(blocking edge)
No
Yes
the current alpha vertex is visible
the current alpha vertex is visible
No
store the pair of start and current alpha vertices as an edge in the visibility graph array, VG, and increase its index
find edges that have alpha vertex as the first or second node and store them in fir_edg & sec_edg
Check fir_edg & sec_edg
not empty
look for these edges in the s list
If fir_ind empty& start vertex not on fir_edg
No
Yes
store fir_edg in S list and increment its index
if fir_ind not empty
Yes
Delete S(fir_ind) and decrease the index
if sec_ind greater than fir_ind
No
No
Yes
decrease Sec_ind by one
If sec_ind empty& start vertex not on sec_edg
No
Yes
store sec_edg in S list and increment its index
if sec_ind not empty
Yes
Delete S(fir_ind) and decrease the index
No
sort the updated version of S
is there still a vertex in alpha list that is not checked?
Yes
No
Is there any stored vertex in the "vertices" variable that has not been chosen as a start vertex?
Yes
No
Yes
Remove repetition in rows from VG
plot all rows in VG on the current figure
End
No

Figure 4: Flowchart of the implemented algorithm

The following figure shows the results of applying this algorithm on three different environments, the last one with convex polygons, to find their visibility graph.

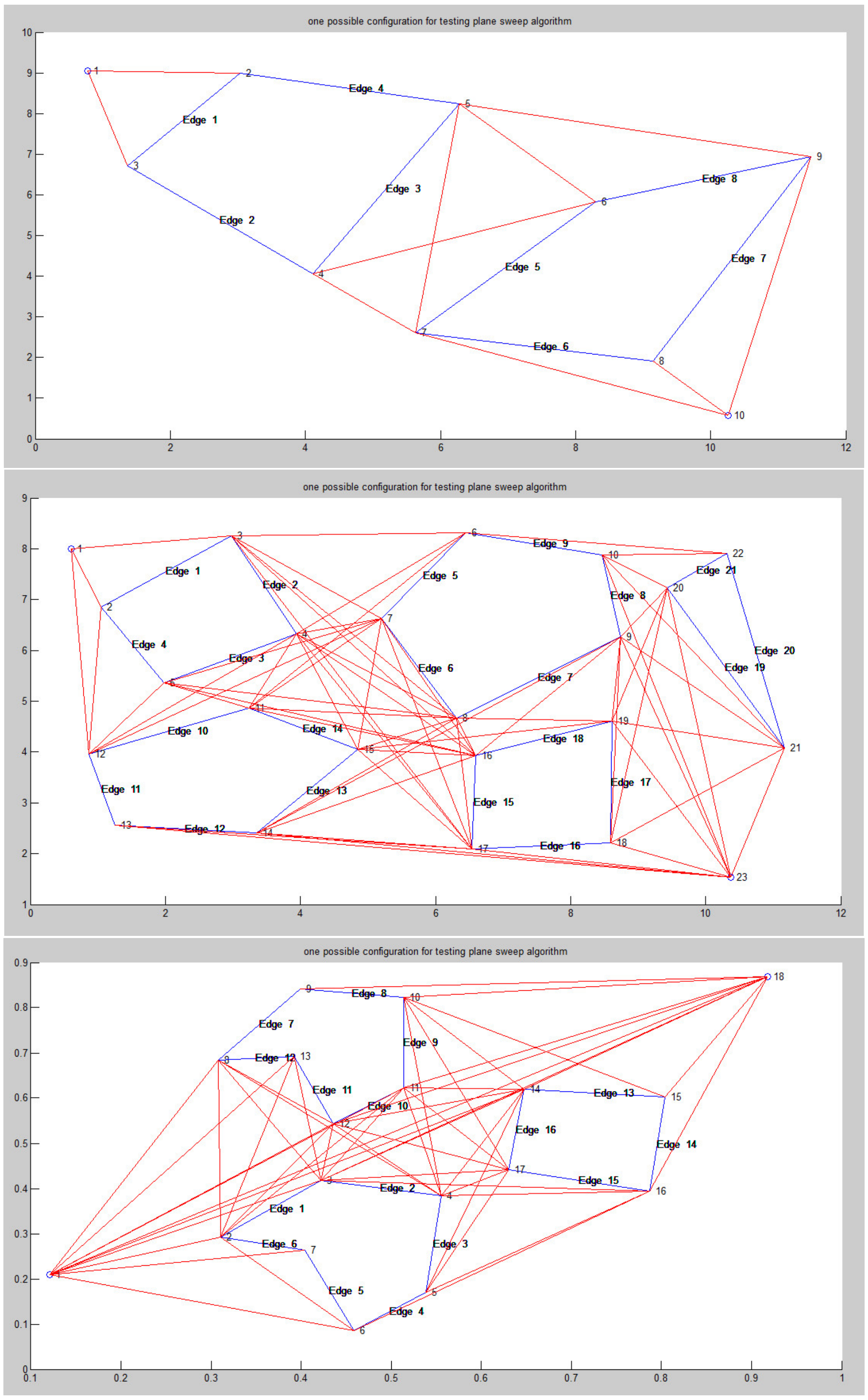

Figure 5: Computing the visibility graph of two environments

3 Conclusion

In this lab session, a path planning algorithm based on topological maps, RPS, that is used to build visibility graph is designed, analysed and implemented in Matlab program. The algorithm is tested on different maps taking into account convex polygons and giving as an output argument all visible vertices of this map.