University of Girona

Autonomous Robots

Lab 5

Graph Search – A* algorithm

Author:
Di MENG

Supervisor:
Dr. Marc Carreras

June 4, 2017

1 Objective

The objective of this lab is to grab the knowledge of A* algorithm which is a method of graph search. Given the start point, obstacles and end point, a visible graph is generated by the Rotational Plane Sweep(RPS) algorithm implemented in the last lab. A* algorithm is used to search the shortest path in the visible graph.

2 Introduction

The topological map is a simplified map only containing the relationship of points. It can be represented by a graph where nodes are real locations and the edges join the positions in free space. And the edges are including distances. The visible graph can be used to build up a topological map. It is a graph of intervisible locations, typically for a set of points and obstacles in the Euclidean plane. Each node in the graph represents a point location and each edge represents a visible connection between points.

A* search algorithm is a computer algorithm that is widely used in pathfinding and graph traversal, the process of plotting an efficiently directed path between multiple points. Thus, with the given visible graph which provides the locations of nodes and the euclidean distance between each two nodes. Graph search consists in generating a sequence of connected nodes that has minimum length. A* algorithm is a graph search algorithm that uses an heuristic to perfprm a better search.

3 Implementation of A* algorithm

The A* algorithm is implemented in the form of a function named "Astar". The input is the vertices representing the environment and edges from the visible graph. The output is the most preferable path and its cost. In this case, the minimum cost is the shortest distance.

function [path,minCost]=Astar(vertices, edges)

3.1 Environment configuration

The initial environment is including the start point, the goal and the obstacles represented by polygons. After applying RPS algorithm(function below), a visible graph is established. The graph contains vertices and edges which are the locations of the nodes and the lengths between each nodes for graph search.

function [edges]=RPS(vertices)

3.2 The cost of nodes in the graph

First of all, the costs of all the vertices are computed. Each node holds a cost which is the euclidean distance to the goal position. The start node and the goal node have zero cost. Once the node is explored, its cost is updated as the addition of its cost and the euclidean distance to the last nodes in the path.

3.3 Initializing the C list and O list

In the A* algorithm, there are close list(C list) and open list(O list) which contains the explored nodes and sorted unexplored nodes respectively. At the beginning, the path starts from the start point to the first vertex. The initial C list is [1, 0]. Since the current node is located at the first vertex, the available paths from first vertex are searched. For the resultant nodes, the costs of them are computed and sorted. In addition, the backpoint of the new explored nodes is saved. Thus, the O list contains the possible path nodes with costs and backpoints.

3.4 Exploring new nodes until the goal

Since we have the C list and O list, for the fist node in the O list which holds the least cost, we add it into the C list with its backpoint. Then for the end node in the C list, we search its following available nodes and add them into the O list. This operation is kept continuing until the goal node appearing to be the first node in the O list which means the path with minimum cost has been found. In the end, we add the goal node to the C list with its backpoint.

3.5 Obtaining the final path

Once we get the final version of C list, it also contains many useless node we explored. The final path is a sequence from the start node to the goal node. So from the end of the C list, we backsearch the sequential path to the beginning of the C list. In the end, a vector is obtained which contains the sequential nodes from the start to the goal.

4 Results from different environments

4.1 Small environment

The minimum cost is 13.3788, the elapsed time is 0.001897 seconds.

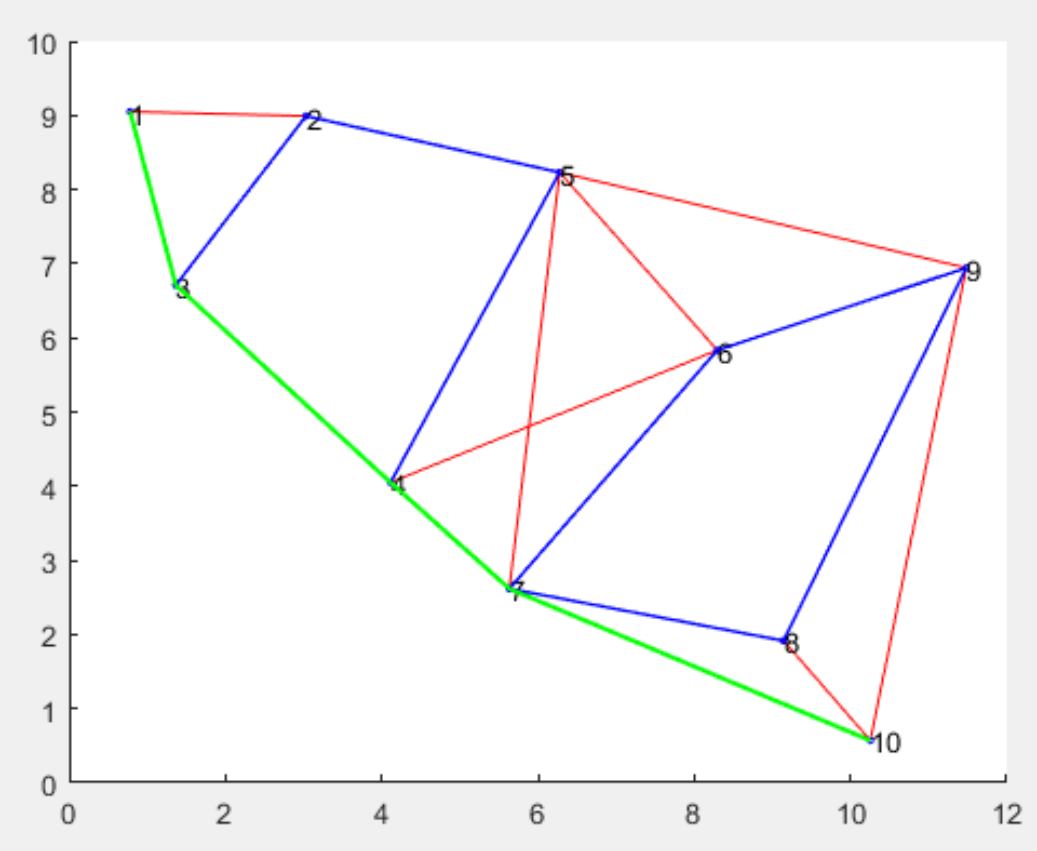

Figure 1: Shortest path in green in small environment

4.2 Convex polygon

The minimum cost is 0.7935, the elapsed time is 0.001996 seconds.

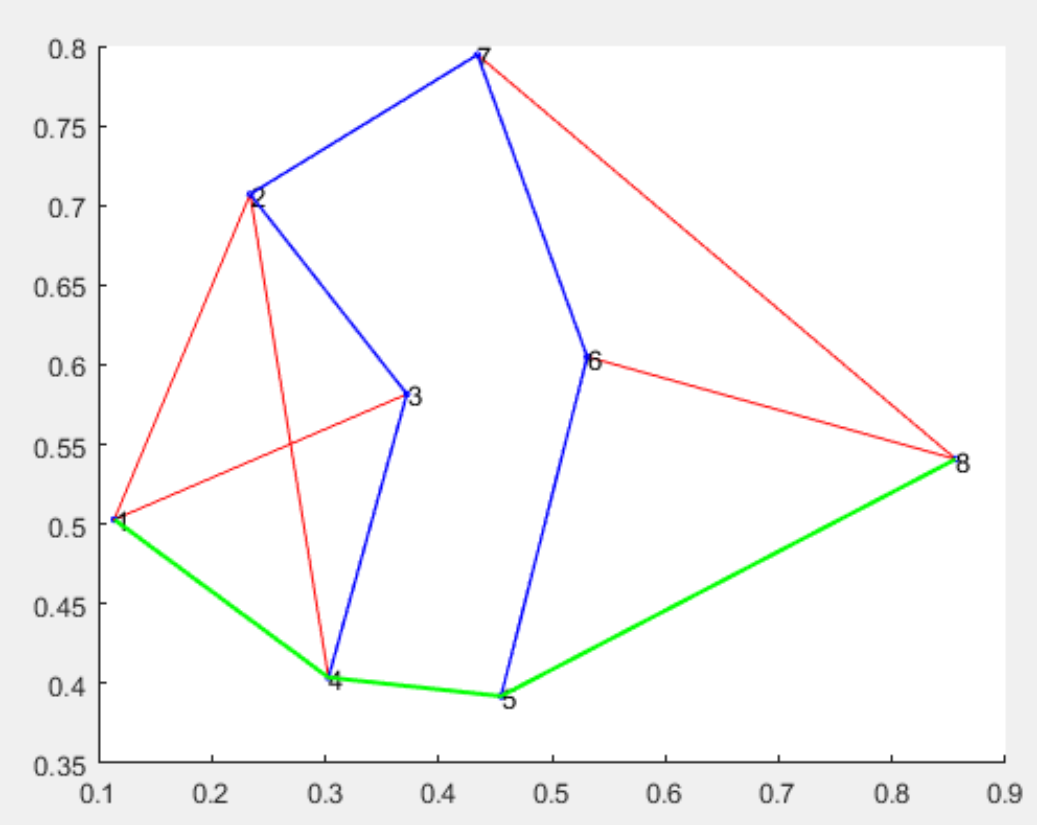

Figure 2: Shortest path in green in convex environment

4.3 Big environment

The minimum cost is 12.6005, the elapsed time is 0.003306 seconds.

5 Conclusion

In this lab assignment, the shortest path was obtained by A* algorithm. This algorithm was implemented in Matlab and tested in different environments. The results showed proper performance. Compared with many graph search

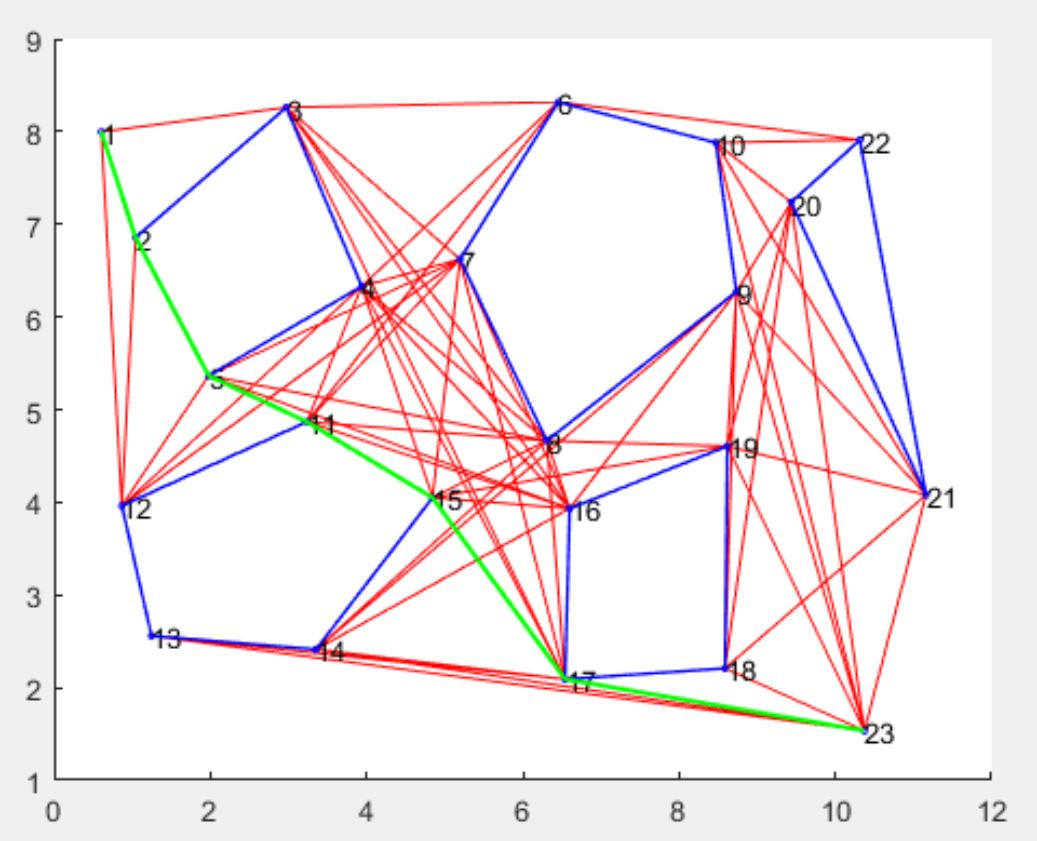

Figure 3: Shortest path in green in big environment

algorithms which consumes much time, A* algorithm is very fast by using heuristic.