University of Girona

Autonomous Robots

Lab 2

RRT Algorithm

Author:
Di MENG

Supervisor:
Dr. Marc CARRERAS

March 19, 2017

Universitat de Girona

1 Objective

The objective of this lab is to implement the Rapidly-exploring Random Tree algorithm(RRT) for robot path planning. With the provided map, start and goal positions to generate a random tree. Find an available path from the start point to final point which is not occupied with any obstacles and then reduce the unnecessary way-points to get a shorter smoothed path.

2 Introduction

RRT algorithm is one of the sampling-based techniques. The map is previously known. Its basic idea is to generate a space-filling tree inside the environment. Random samples drawn in the free space are used to construct the tree which has a probability to grow in the direction towards a specific point. It also has a bias to build the tree to the areas which is not aiming to the goal. That is for avoiding the branch of the tree stuck in a local area and never reaching to the goal.

RRT algorithm starts by taking account into the map of the environment, the starting and goal positions for robot. Then generate a random point in the free space which leads to a new node. This new node is considered effective when the connection between it and last node is in free space. If there is obstacle in the way, this node is not being considered any more and a new random point would be generated.

The tree is ended building when the connection between start point and goal point is finished. That means the node of the tree finally reach to the goal position so that the robot can go from start to the goal successfully.

3 Implementation of RRT algorithm

The map of the environment is provided as a matrix full of value 0 and 1 where 0 values represent the free space and 1 values represent the areas occupied with obstacles. The iteration of loop for sampling is limited. And the probability of considering the point leading the direction of a new node as a random point or the goal point is given. Also, a factor is provided to decide the step size for new node.

3.1 Generating the tree

Basically, my code did the following steps to generate the tree:

1. Initialize the vertices vector which contains all the nodes of the tree with the start point and create an empty array for saving edges, the lines between every nodes.

2. Generate a random value with uniform distribution between 0 and 1. If the random value is less than the given probability, consider the goal as new random point. On the contrary, sample a random point in the free space.

3. Generate a new node to the direction of the random point with a specific step size. This step size is fixed unless the distance from last point(start point in 1st iteration) to the random point is less than the given factor.

4. An edge is built from the last step. Check if this edge goes cross any obstacles, if not, that means this edge is effective and put the new node generated from last step in the vertices vector. Update the vertices vector.

5. Repeat the step 2-4 until the new node reaches to the goal position or the iteration is run out of limits.

In the end, we get a vertices vector containing all the nodes in the tree and a edge vector containing all the branches of the tree. By selecting the connected vertices shown in edge vector, we finally get the path which is a row vector containing the vertices that robot should reach each step.

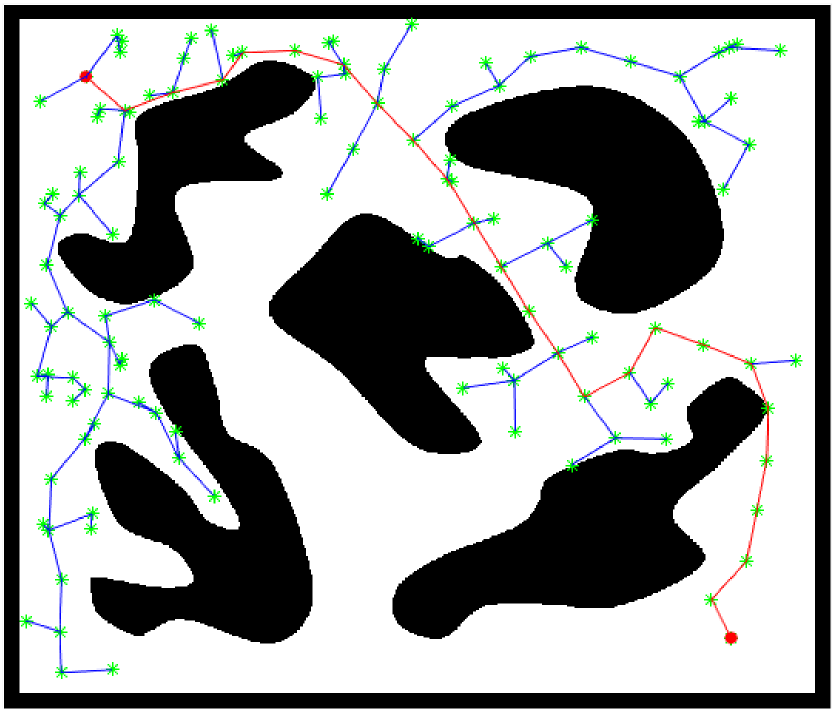

Figure 1: Path in map

The two red points are the start and goal points. The green points are all the nodes(vertices) in the tree, blue lines are edges. The red line is the generated path connected the start point and the end point.

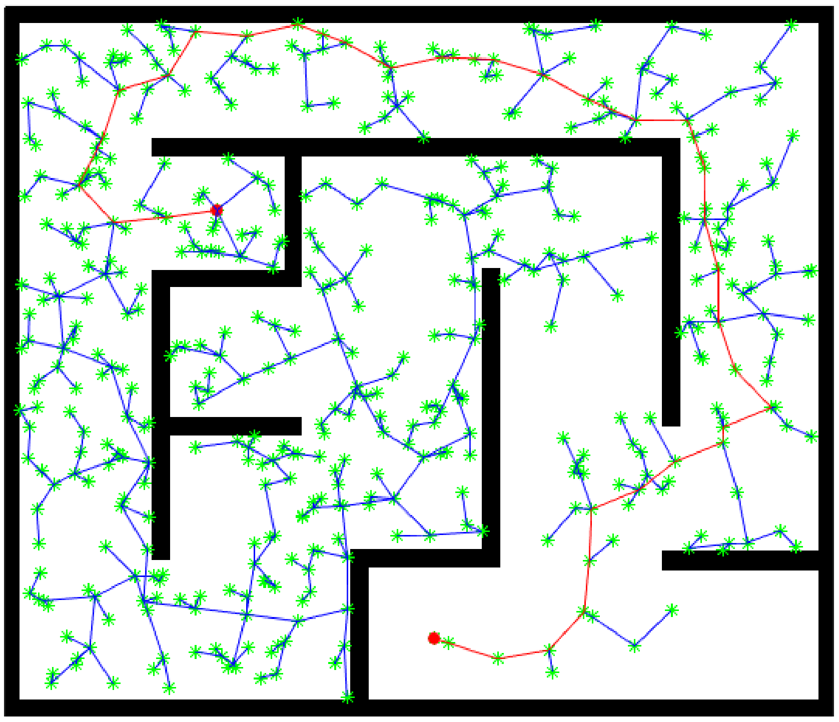

Figure 2: Path in maze

3.2 Smoothing the path

The path generated along the tree is always tortuous and there are many unnecessary nodes that robot has to reach. So, the path can be smoothed and simplified by reducing some unnecessary vertices in the path vector which are not affecting to reaching the goal. The smoothed path can still connect the start point to the goal point.

The code for smoothing the path follows the next steps:

1. Initialize the smoothed path with the goal point.
2. Take the first vertex as the starting point and the goal as the ending point.
3. Check if the starting point can connect to the ending point straight forward which means there's no occupied obstacles between.

4. If it fails, consider the next vertex(second vertex in 1st iteration) as the starting point and check again. Until the starting point reach the ending point, take that starting point as the ending point and put it in the smoothed path list.

5. Repeat the step 2-3 until the smoothed path has the first vertex in the list.

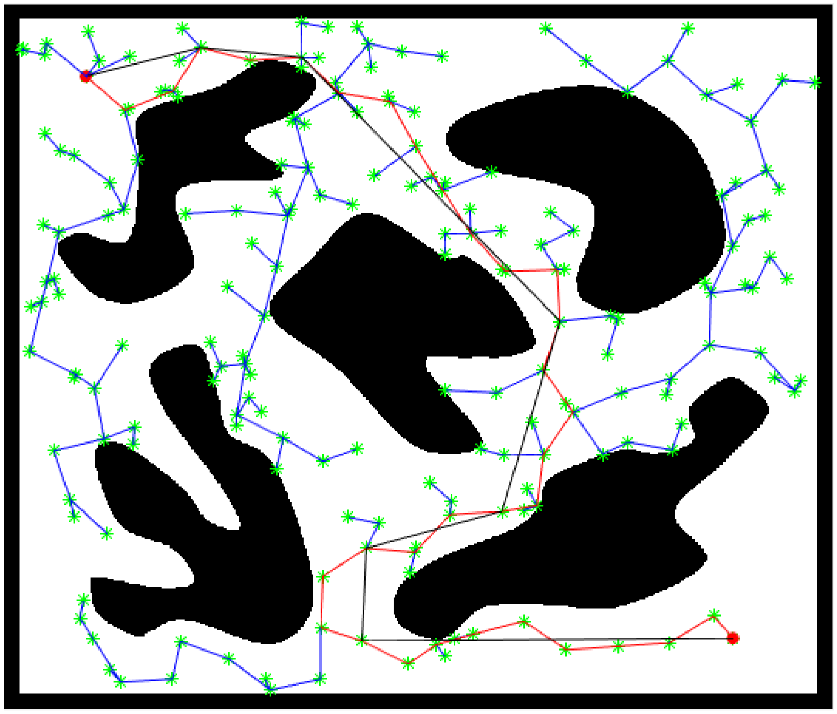

Figure 3: Smoothed path in map

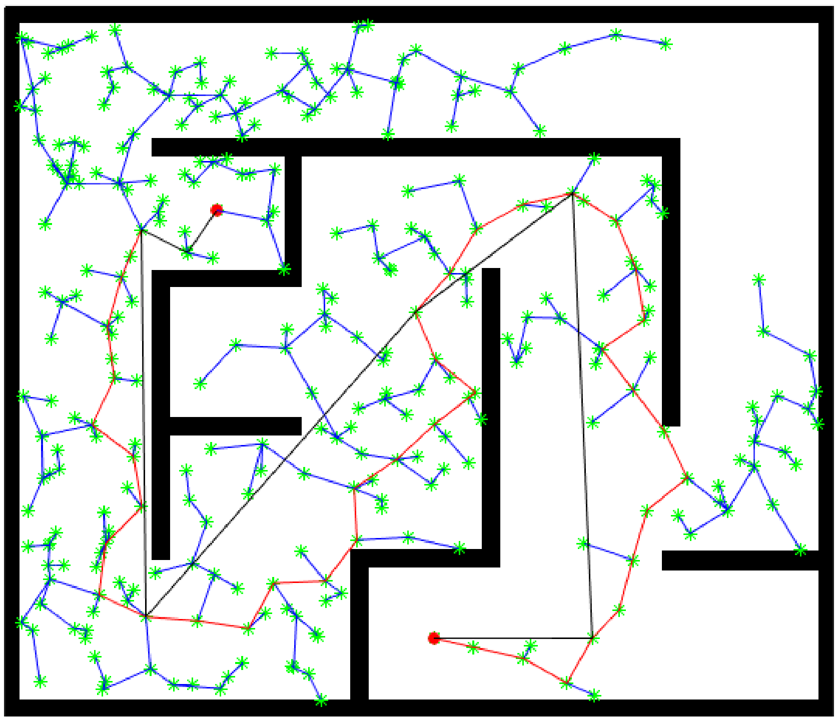

Figure 4: Smoothed path in maze

The black line is the smoothed path. Compared with the path along the tree, smoothed path is more straight forward and avoid many unnecessary distances that robot has to go.

4 Conclusion

In this lab assignment, a sampling-based path planning technique called Rapidly-exploring Random Tree algorithm is implemented in Matlab. A tree is generated in the free space of the map and a path going from start position to the goal position is found along the tree. Then an optimization is implemented for smoothing the path. Thus, with given map, a path can be always generated from the start to the goal based on this algorithm.