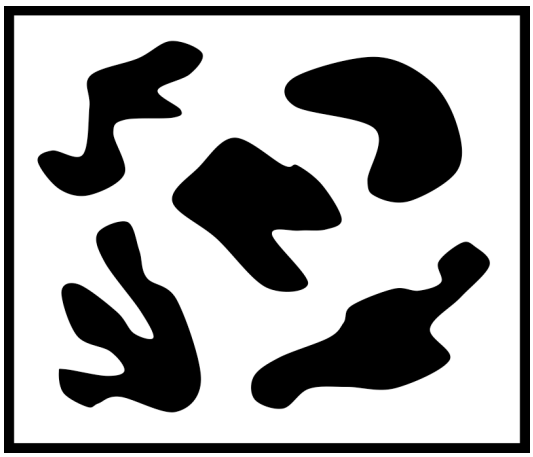
# Autonomous Robots
# Sampling-based algorithms – RRT

This document will guide you through the practical work related to path planning algorithms based on sampling-based algorithms. In particular, this practical exercise consists on programming the RRT (Rapidly-Exploring Random Trees) algorithm to solve a 2D path planning problem. All the code has to be programmed in Matlab.

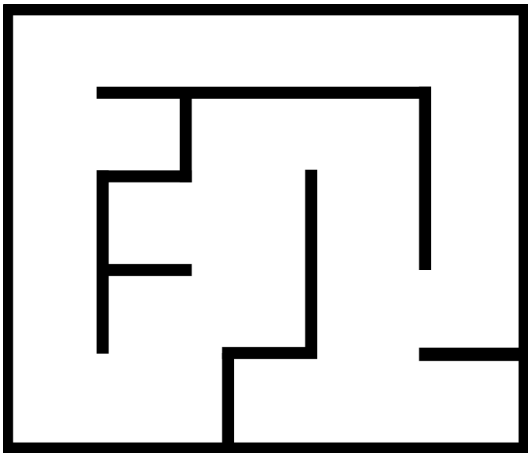
### 1. Environment

In order to have a convenient environment, we will use a discrete environment or map in which cells having a 0 will be free cells, and cells having a 1 will be occupied cells. Two environments are proposed to test your algorithm. You will find some files containing the values of the two maps. Check also the coordinates of the start and goal position below the figures.



Environment: map

q_start=[80, 70]; q_goal=[707, 615];



Environment: maze

q_start=[206, 198]; q_goal=[416, 612];

However, the vertex coordinates of the RRT algorithm will not be discrete but continuous. We will generate random values inside the range of the environment and the algorithm will build the tree with vertices having continuous coordinates. In order to determine if a vertex or edge belongs to the free space, the continuous coordinates will be rounded, and the rounded coordinate will be checked in the Matlab variable containing the map of the environment.

You can plot the previous maps by loading the data stored in the ".mat" files and then by using the following the commands:

```
colormap=[1 1 1; 0 0 0; 1 0 0; 0 1 0; 0 0 1];
imshow(uint8(map),colormap)
hold on
```

IMPORTANT: consider the columns of the "map" variable as the "x" axis and the rows as the "y" axis.

**2. RRT**

Program the RRT algorithm to build a tree that goes from the start position till the goal position and to generate a path that connects both vertices.

IMPORTANT: Do not use 2 trees (one centred in the start and another centred in the goal)! Use only one tree centred in the start position.

You must program a Matlab function that has the following definition. It is very important the input and output arguments have the same format.

```
function [vertices,edges,path]=rrt(map,q_start,q_goal,k,delta_q,p);
```

where:

map: matrix that you can obtain loading the ".mat" files.

q_start: coordinates x and y of the start position. You can find the coordinates below the figures of the environment in the previous page.

q_goal: coordinates x and y of the goal position. You can find the coordinates below the figures of the environment in the previous page.

k: maximum number of samples that will be considered to generate the tree, if the goal is not found before.

delta_q: distance between q_new and q_near.

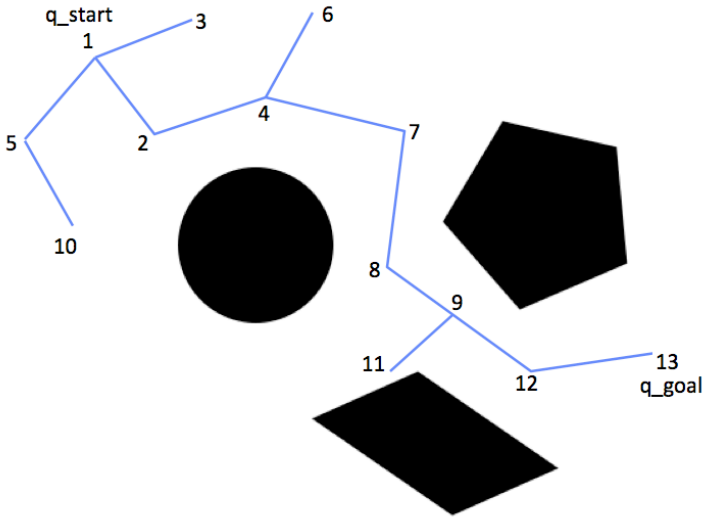
p: probability (between 0 and 1) of choosing q_goal as q_random.

vertices: list of x and y coordinates of the vertices. The first vertex will correspond to the start position and the last one will correspond to the goal position. The variable MUST have 2 columns for x and y coordinates and n rows (being n the number of vertices found in the tree).

edges: list of edges of the tree. There will be n-1 edges, stored in n-1 rows. Each edge MUST contain the index of the vertex (with respect "vertices" variable) and the index of the vertex that is one step to the root (or start position). Once the goal is found, the goal index will be stored in the last row, and will contain the goal index and its upper vertex in the direction to the start position. By following the vertices, the path to the goal can be extracted directly.

path: list of vertex indices from the start vertex (q_start) to the goal vertex (q_goal). In case no solution has been found, the list will be empty. The list MUST be represented as a row vector.

The following figure shows a simple example in which previous commented variables can be checked:




vertices= [
x1, y1
x2, y2
x3, y3
x4, y4
x5, y5
x6, y6
x7, y7
x8, y8
x9, y9
x10, y10
x11, y11
x12, y12
x13, y13];

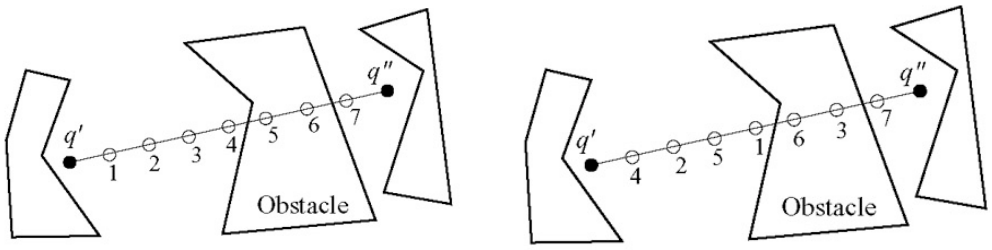
edges=[
2, 1
3, 1
4, 2
5, 1
6, 4
7, 4
8, 7
9, 8
10, 5
11, 9
12, 9
13, 12];

path=[ 13, 12, 9, 8, 7, 4, 2, 1];

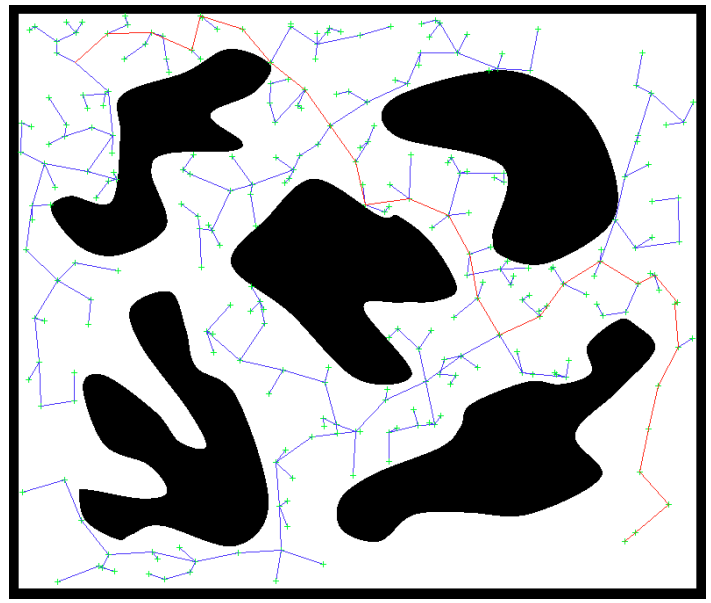
Your algorithm must perform the following things:

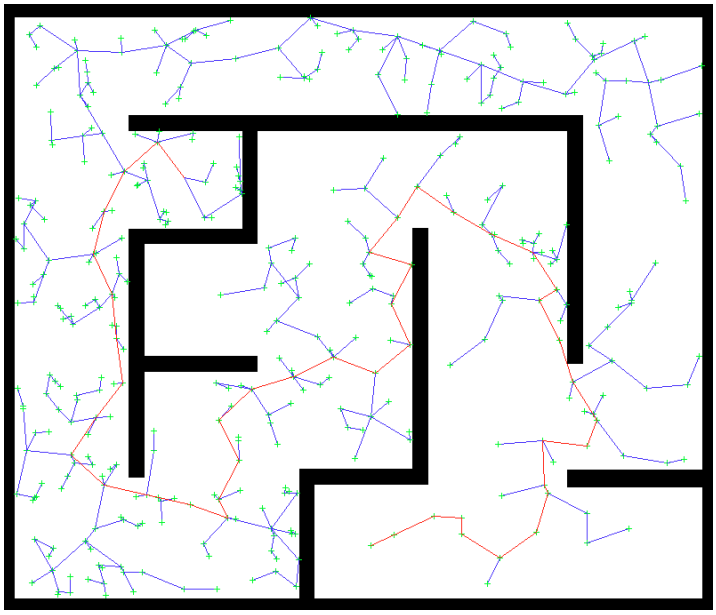
- Initialize the “vertices” variable with q_start
- Initialize the “edges” variable as empty
- For k samples repeat
    - With p probability use q_rand = q_goal
    - Otherwise generate q_rand in the dimensions of the map.
    - Find q_near from q_rand in “vertices”.
    - Generate q_new at delta_q distance from q_near in the direction to q_rand.
    - If q_new belongs to free space
        - If the edge between q_near and q_new belongs to free space
            - Add q_new in “vertices”
            - Add [index(q_new) index(q_near)] in “edges”
            - If q_new == q_goal
                - Fill “path” and stop break RRT function

In order to check if an edge belongs to the free space, use the incremental (left) or subdivision (right) strategies. You can use 10 intermediate points.




Using k=10000, delta_q=50 and p=0.3, you should obtain results such as the ones shown in the following 2 figures (the graphical representation must be done outside the function). You can see the tree in blue starting in q_start and in red the path that goes till q_goal.



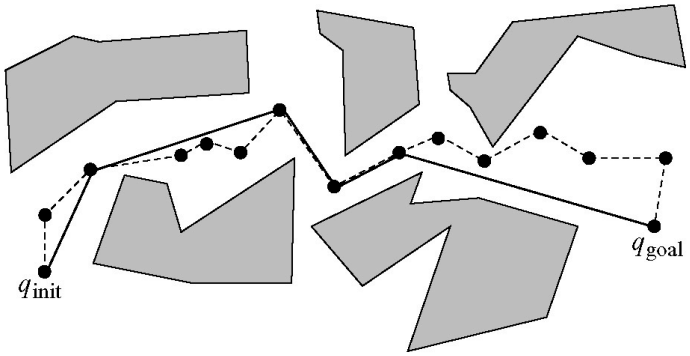


### 3. Smoothing

Once you finish the basic RRT algorithm, you must program a smoothing algorithm for obtaining a shorter and less noisy path.

We will use the greedy approach:

> *Connect q_goal from q_start, if it fails try from a closer position until it connects. Once q_goal is connected, start again with its directly connected position.*




You must program a Matlab function that has the following definition. It is very important the input and output arguments have the same format.

```
function [path_smooth]=smooth(map,path,vertices,delta);
```

where:

map: matrix that you can obtain loading the ".mat" files.

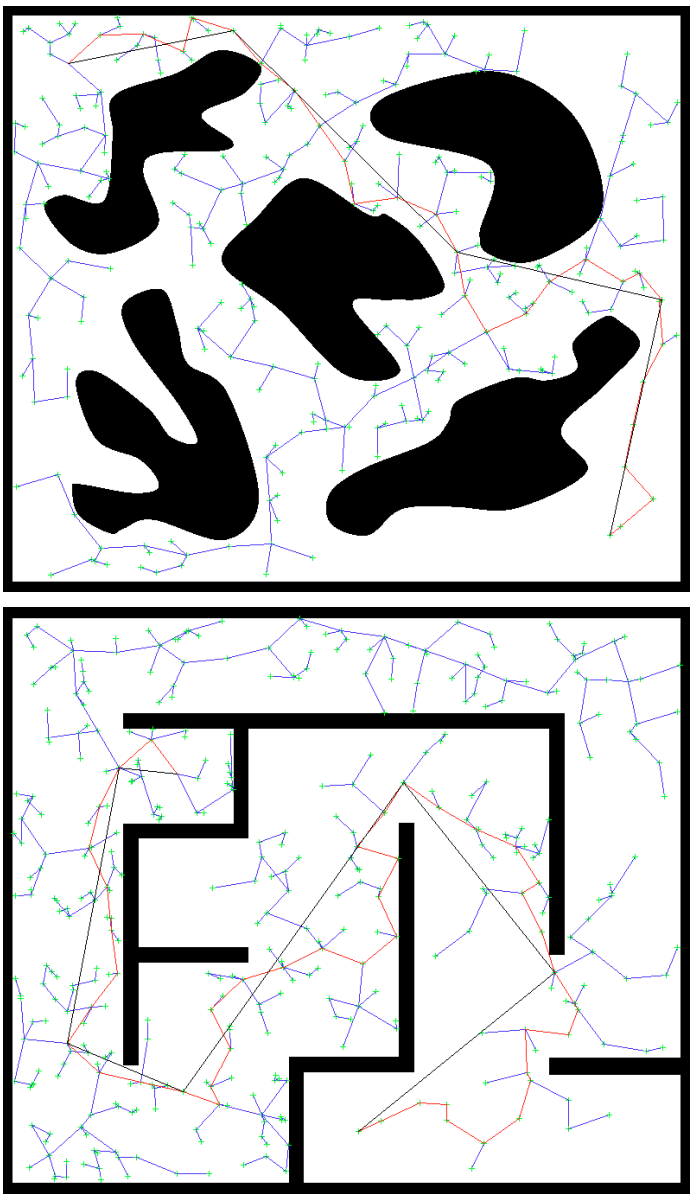
path: list of vertex indices from the start vertex (q_start) to the goal vertex (q_goal). The list MUST be represented as a row vector.

vertices: list of x and y coordinates of the vertices. The first vertex will correspond to the start position and the last one will correspond to the goal position. The variable MUST have 2 columns for x and y coordinates and n rows (being n the number of vertices found in the tree).

delta: incremental distance that will be used to check if direct connection between the vertices of the path is inside the free space. The edges will be divided obtaining several points, each of them separated this delta distance.

path_smooth: reduced list of vertex indices from the start vertex (q_start) to the goal vertex (q_goal) after applying the smoothing algorithm. The list MUST be represented as a row vector.

Using the results of the previous section, with a delta equal to 5, the smoothing algorithm generates the following paths (in black).



**5. Submission.**

**WORK TO DO:** Submit a report in pdf and the 2 Matlab files “RRT.m” and “smooth.m”. Explain in detail with graphical information, in the report, the work done in all sections. Explain also the problems you found. You might want to test your algorithm in other environments.

NOTE that only these 3 files are required and will be evaluated. Do not send more files.