Consider the map attached as *image_map.mat*. The 0 value represents an obstacle, the 1 value is a free point in the partitioned map. To show the map, use the command *imshow(image_map)*. Implement yourself in a software program the RRT method for a point robot moving in the above map from $q_i = [20 \quad 20]$ to $q_f = [125 \quad 400]$. Select a suitable maximum number of iterations and show the obtained graph is a solution has been found. Otherwise, report a failure and increase the maximum number of iterations. Repeat the procedure for a finite number of times at your choice. [Hint: the robot is a point; therefore, the collision check algorithm is very simple: for a given point, you must just check whether the associated value of the map is 0 or 1.]

RTT can be schematized over a series of steps

- The first step is the generation of a random configuration q_{rand} with a uniform probability distribution. Algorithm ends when if the goal has been reached or if the maximum number of iterations is exceeded. Normalizing x between 0 and 428 and y between 0 and 171, it is sure that the q_{rand} generated is inside the domain

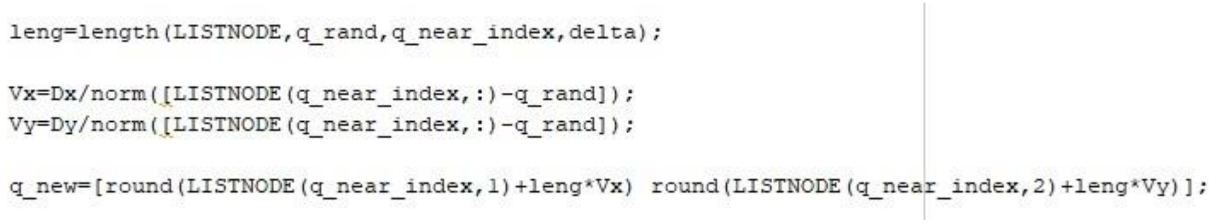
```
while (i<internal & found==0)

    x_rand=round(rand*428);
    y_rand=round(rand*171);

    q_rand=[x_rand y_rand];
```

- q_{rand} must be connected to the nearest node belonging to the network(q_{near}). An extremely high upper bound has been chosen, to be sure to find a better value.

```
%connection with the nearest node
function q_near_index=near(LISTNODE,q_rand,N)
        upper=10000;

        for j=1:N
            if(norm([LISTNODE(j,:)-q_rand])<upper)
                upper=norm([LISTNODE(j,:)-q_rand]);
                q_near_index=j;
            end
        end
end
```

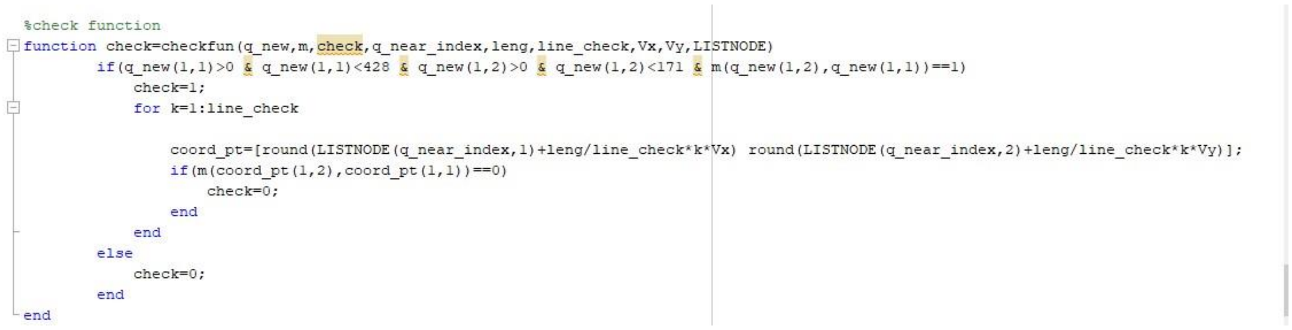
- Given a predeterminant distance δ; if the distance is longer than δ , this point is replaced with a point belonging to the segment which connects the 2 points but with distance δ

```
%calculation of the allowable distance
function leng = length(LISTNODE,q_rand,q_near_index,delta)
    if(norm([LISTNODE(q_near_index,:)-q_rand])<delta)
        leng=norm([LISTNODE(q_near_index,:)-q_rand]);
    else
        leng=delta;
    end
end
```



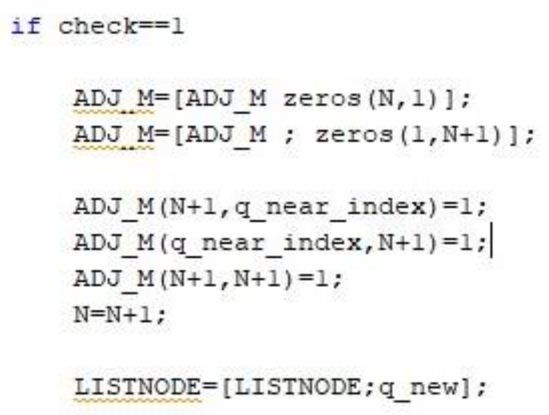
```
leng=length(LISTNODE,q_rand,q_near_index,delta);

Vx=Dx/norm([LISTNODE(q_near_index,:)-q_rand]);
Vy=Dy/norm([LISTNODE(q_near_index,:)-q_rand]);

q_new=[round(LISTNODE(q_near_index,1)+leng*Vx) round(LISTNODE(q_near_index,2)+leng*Vy)];
```

- To implement a collision-check between q_{near} and q_{new}



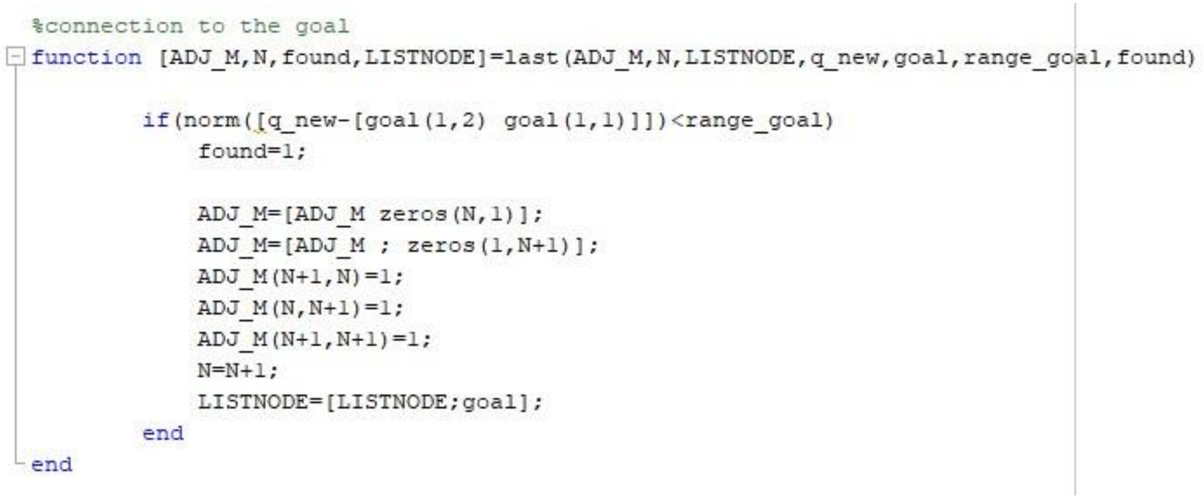
```
%check function
function check=checkfun(q_new,m,check,q_near_index,leng,line_check,Vx,Vy,LISTNODE)
        if(q_new(1,1)>0 & q_new(1,1)<428 & q_new(1,2)>0 & q_new(1,2)<171 & m(q_new(1,2),q_new(1,1))==1)
            check=1;
            for k=1:line_check

                coord_pt=[round(LISTNODE(q_near_index,1)+leng/line_check*k*Vx) round(LISTNODE(q_near_index,2)+leng/line_check*k*Vy)];
                if(m(coord_pt(1,2),coord_pt(1,1))==0)
                    check=0;
                end
            end
        else
            check=0;
        end
end
```

- If there are no collisions, the point must be added to the graph



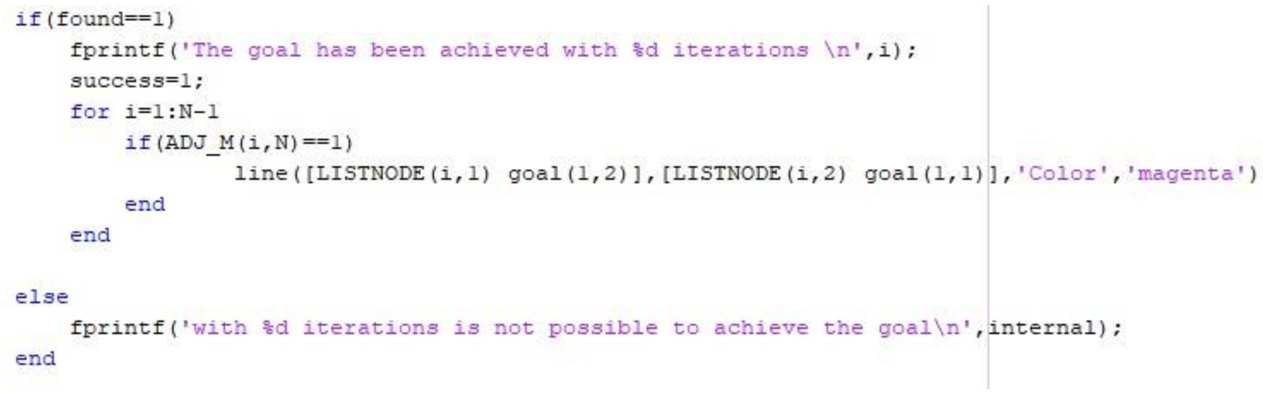
```
if check==1

    ADJ_M=[ADJ_M zeros(N,1)];
    ADJ_M=[ADJ_M ; zeros(1,N+1)];

    ADJ_M(N+1,q_near_index)=1;
    ADJ_M(q_near_index,N+1)=1;
    ADJ_M(N+1,N+1)=1;
    N=N+1;

    LISTNODE=[LISTNODE;q_new];
```

- In case q_{new} has a distance less than range_goal to the goal , it converges to the goal.



```
%connection to the goal
function [ADJ_M,N,found,LISTNODE]=last(ADJ_M,N,LISTNODE,q_new,goal,range_goal,found)

        if(norm([q_new-[goal(1,2) goal(1,1)]])<range_goal)
            found=1;

            ADJ_M=[ADJ_M zeros(N,1)];
            ADJ_M=[ADJ_M ; zeros(1,N+1)];
            ADJ_M(N+1,N)=1;
            ADJ_M(N,N+1)=1;
            ADJ_M(N+1,N+1)=1;
            N=N+1;
            LISTNODE=[LISTNODE;goal];
        end
end
```

- At the end of the simulation, the following messages are printed giving information on the status of the algorithm

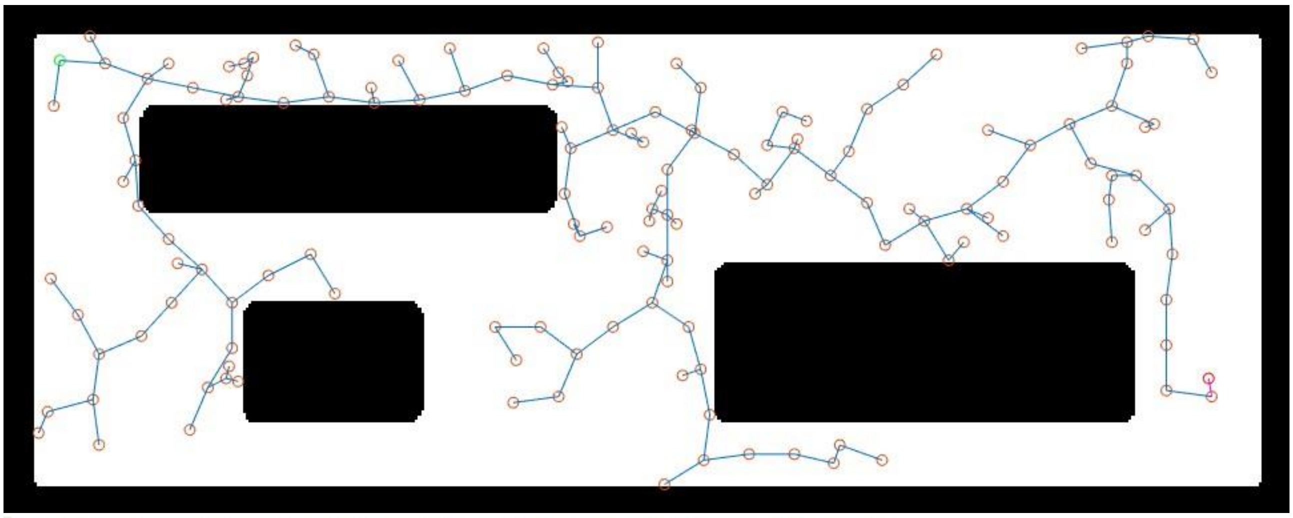


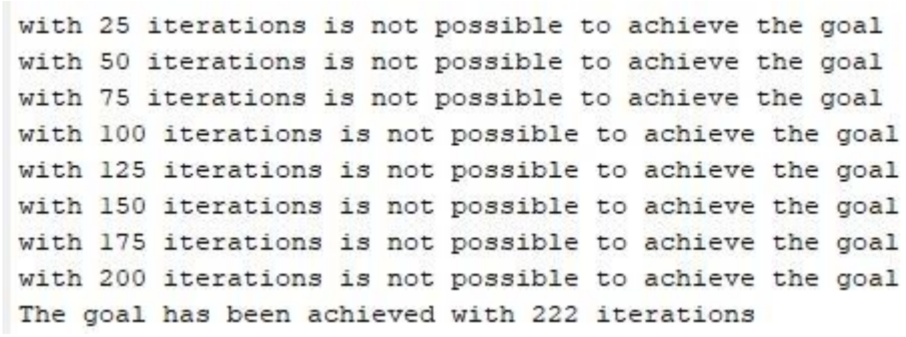
```
if(found==1)
    fprintf('The goal has been achieved with %d iterations \n',i);
    success=1;
    for i=1:N-1
        if(ADJ_M(i,N)==1)
                line([LISTNODE(i,1) goal(1,2)],[LISTNODE(i,2) goal(1,1)],'Color','magenta')
        end
    end

else
    fprintf('with %d iterations is not possible to achieve the goal\n',internal);
end
```

The variable “increase” represents the maximum number of times that the iteration increment mechanism can be done

Below, it is shown a simulation in case the goal has been reached and a simulation otherwise.

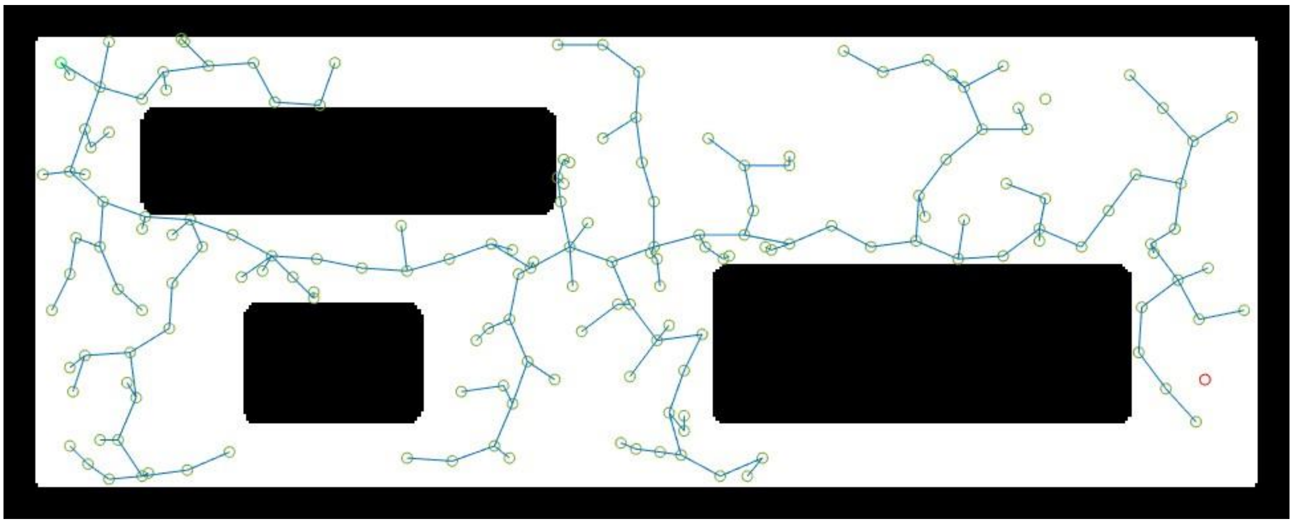
In case of success:

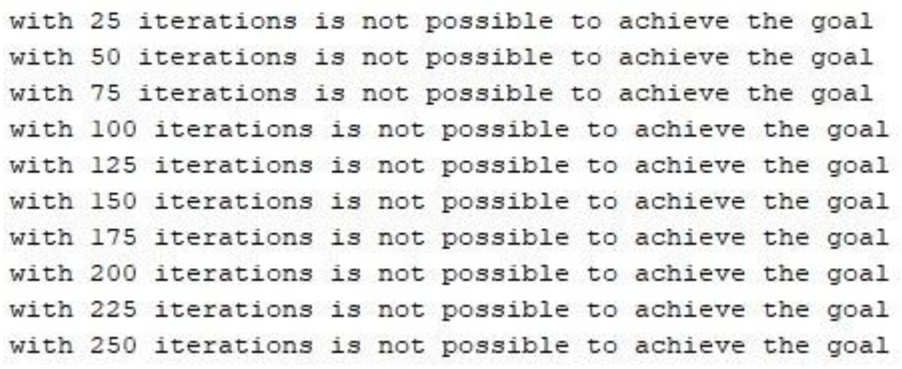




```
with 25 iterations is not possible to achieve the goal
with 50 iterations is not possible to achieve the goal
with 75 iterations is not possible to achieve the goal
with 100 iterations is not possible to achieve the goal
with 125 iterations is not possible to achieve the goal
with 150 iterations is not possible to achieve the goal
with 175 iterations is not possible to achieve the goal
with 200 iterations is not possible to achieve the goal
The goal has been achieved with 222 iterations
```

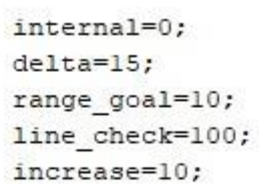
In case of failure:





```
with 25 iterations is not possible to achieve the goal
with 50 iterations is not possible to achieve the goal
with 75 iterations is not possible to achieve the goal
with 100 iterations is not possible to achieve the goal
with 125 iterations is not possible to achieve the goal
with 150 iterations is not possible to achieve the goal
with 175 iterations is not possible to achieve the goal
with 200 iterations is not possible to achieve the goal
with 225 iterations is not possible to achieve the goal
with 250 iterations is not possible to achieve the goal
```

The parameters chosen are:



```
internal=0;
delta=15;
range_goal=10;
line_check=100;
increase=10;
```

With a choice of a smaller δ, the number of iterations required to reach the goal would have been greater.

range_goal has been chosen smaller than δ to ensure that the RTT algorithm arrives sufficiently close to the goal.

Raising “increase” or the number of iterations, it is more likely to reach the goal

The file containing the algorithm is called: "RRT.m"