Assignment 2 - PRM or RRT for finding a feasible path

Rapidly Expanding Random Tree

A Rapidly Exploring Random Tree (RRT) is a popular algorithm used in motion planning to efficiently search a high-dimensional space for a feasible path between a start and goal configuration.

The RRT algorithm works by incrementally growing a tree rooted at the start configuration. At each iteration, a new configuration is randomly sampled from the configuration space, and the nearest configuration in the tree is identified. A new node is added to the tree that is connected to the nearest node with a path that does not collide with obstacles in the environment. This process is repeated until the goal configuration is reached or a maximum number of iterations is reached.

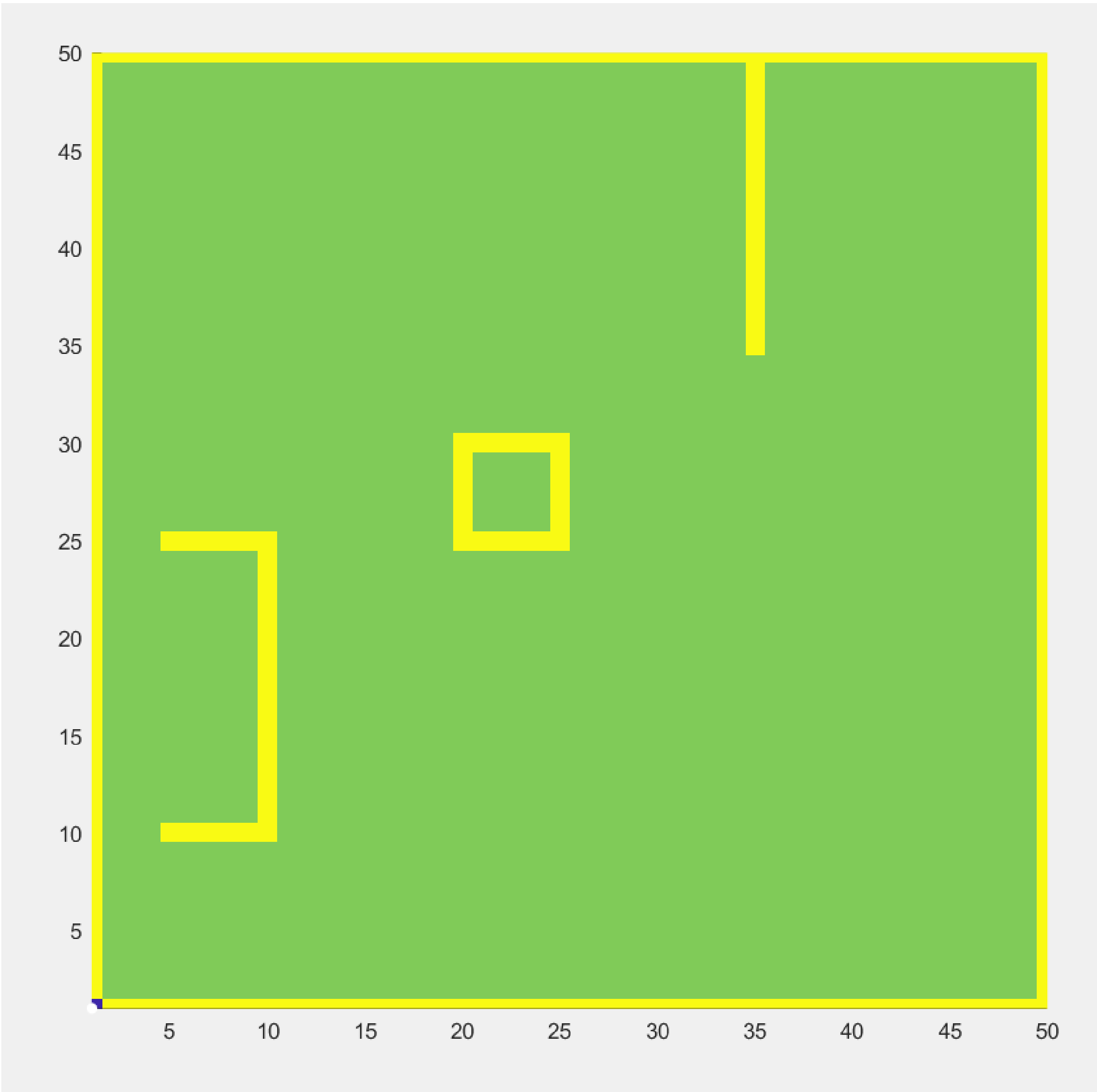
The "rapidly exploring" aspect of the RRT algorithm comes from the fact that the algorithm tends to expand the tree quickly in unexplored areas of the configuration space. The "random" aspect comes from the fact that new configurations are sampled randomly, which helps to avoid bias towards a particular direction or region of the configuration space.

Overall, the RRT algorithm is a powerful tool for solving complex motion planning problems in robotics and other related fields.

Simulations :-
https://drive.google.com/drive/folders/1K-cTFGKfTyilckYpuZG9DZQpK6Y0FIyx?usp=share_link

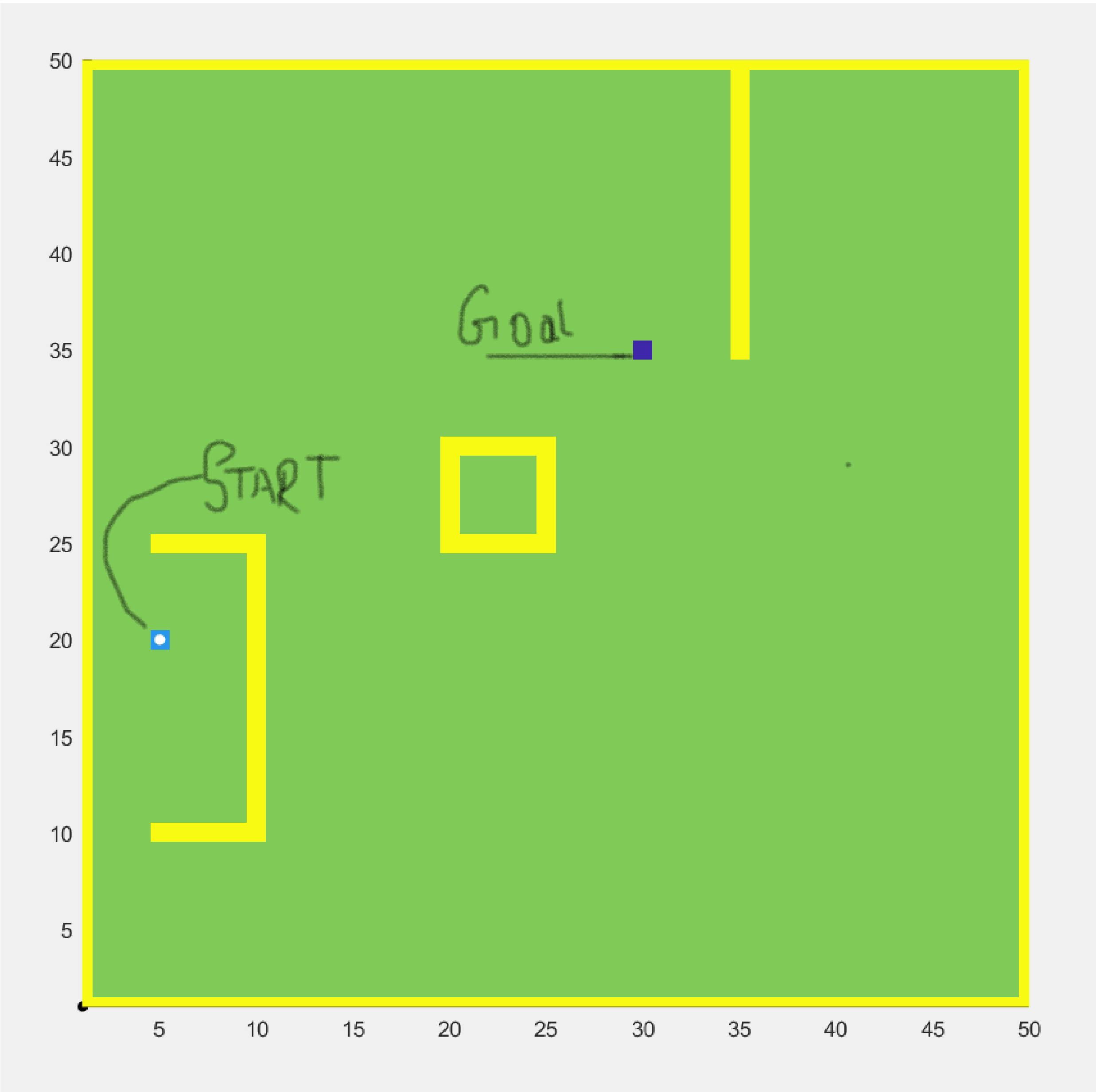
Assignment 2 - PRM or RRT for finding a feasible path

1. Generate a rectangular workspace with three obstacles



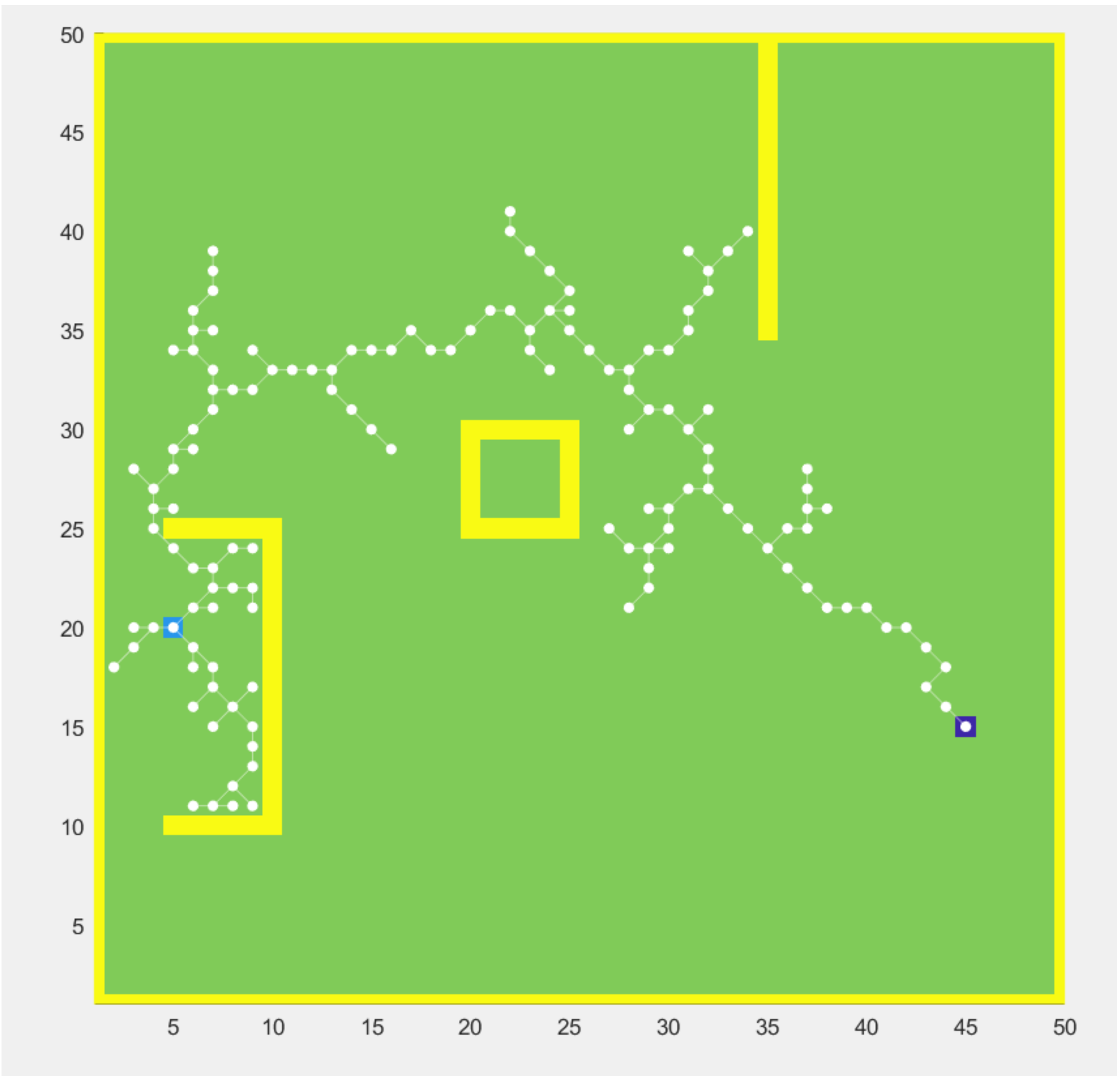
Assignment 2 - PRM or RRT for finding a feasible path

2. Mark the start point and the goal point in the workspace.



Assignment 2 - PRM or RRT for finding a feasible path

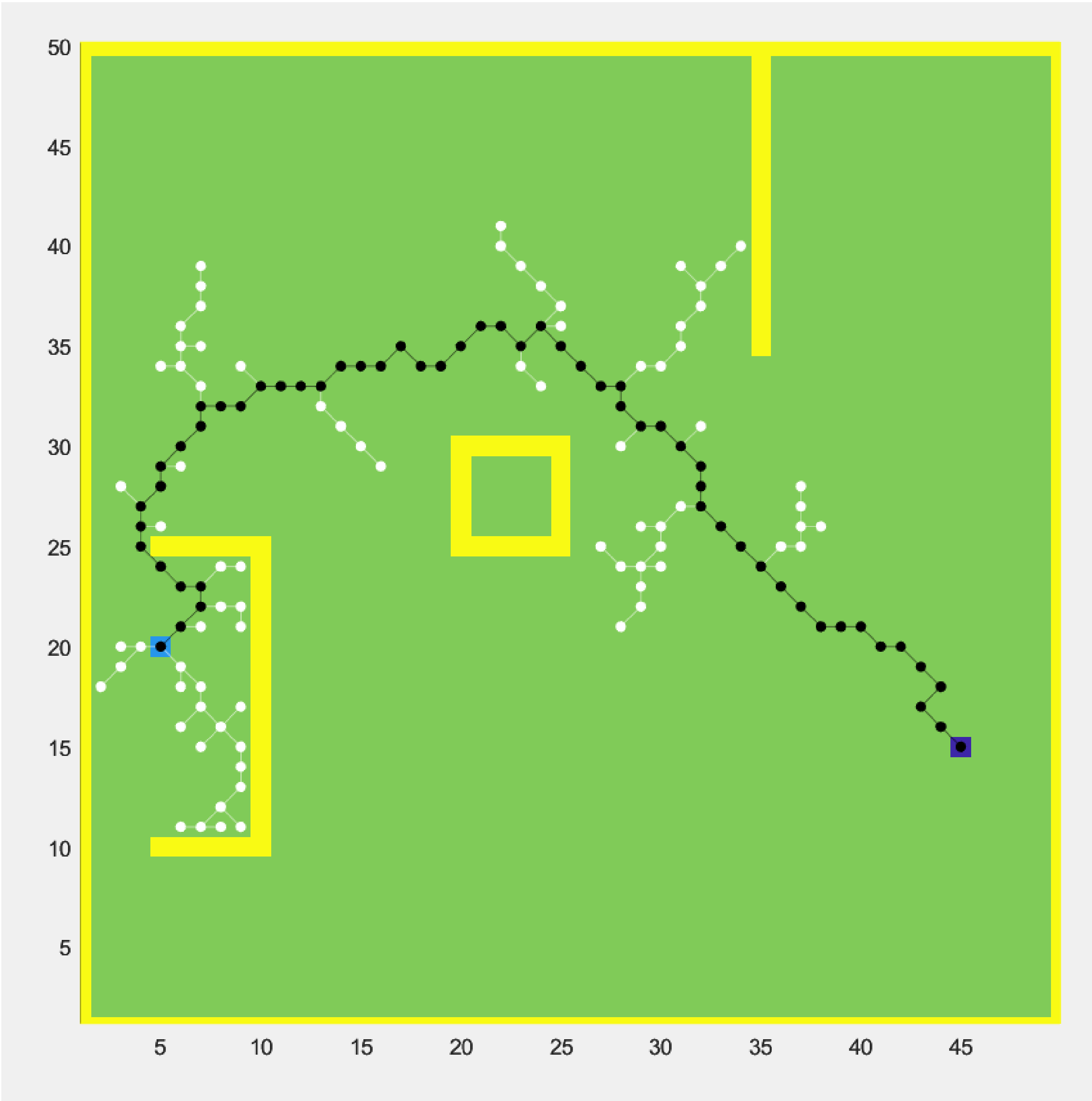
3. Generate feasible points (generate a set of random points from the start point and connect it to a tree forming branches. From the start point the next feasible points (one or more) may be made at a random direction with a fixed step length. Generate the roadmap in the form of a tree with branches. Find a path from the start point to the goal point)



Simulation :-
https://drive.google.com/file/d/1lhVhaE2XOY2DNIuYYRNSoLq3fRmEHZmf/view?usp=share_link

Assignment 2 - PRM or RRT for finding a feasible path

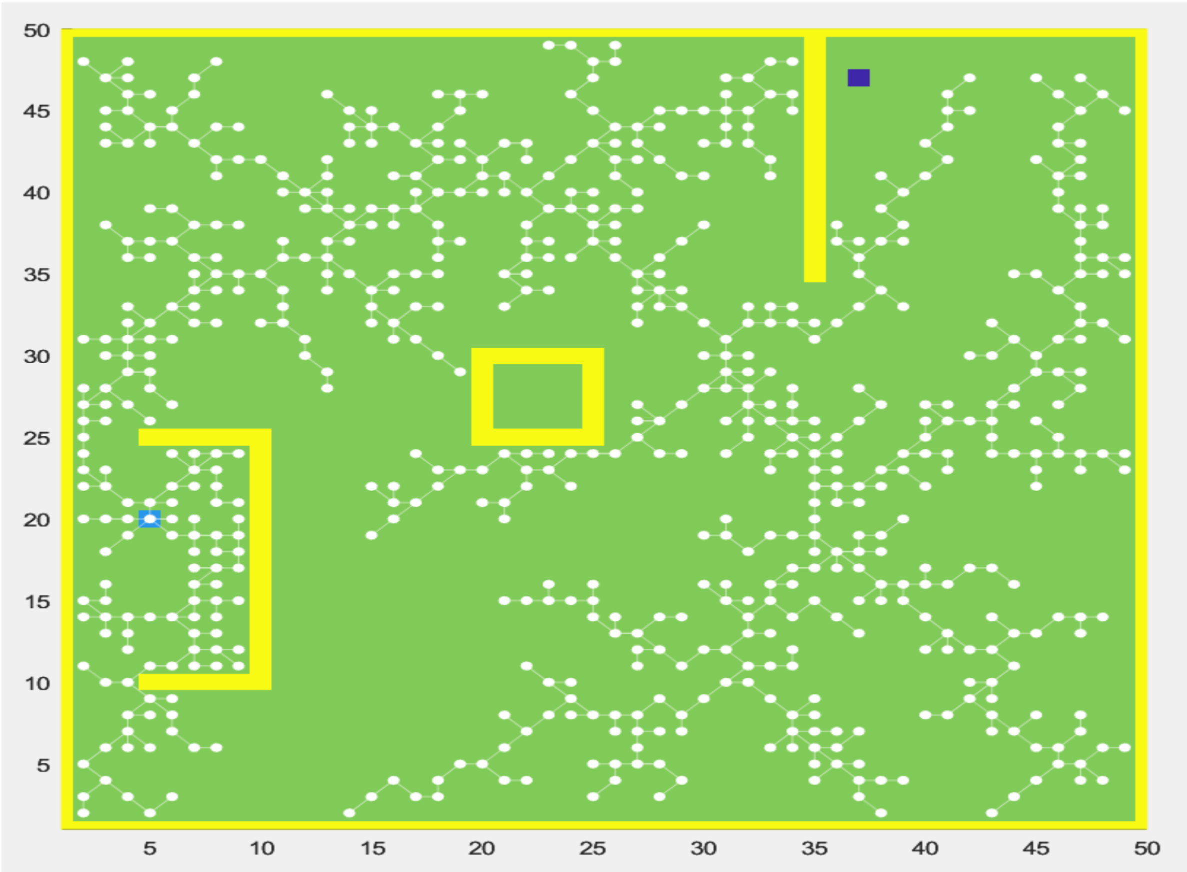
4. Road map generated, Final path from initial point to goal point.

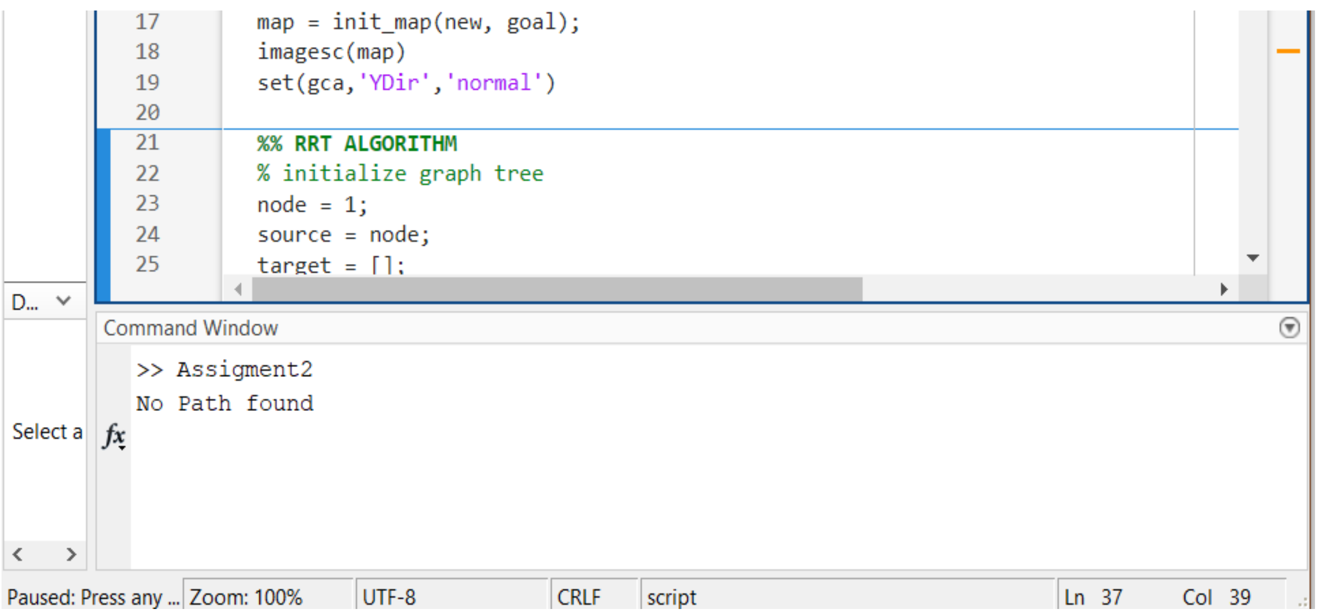


Simulation :-
https://drive.google.com/file/d/19i5aGWF1cxL01UNcjT7fW96ZdRYPFQRO/view?usp=share_link

Assignment 2 - PRM or RRT for finding a feasible path

5. Failed cases , with suitable object position or shapes

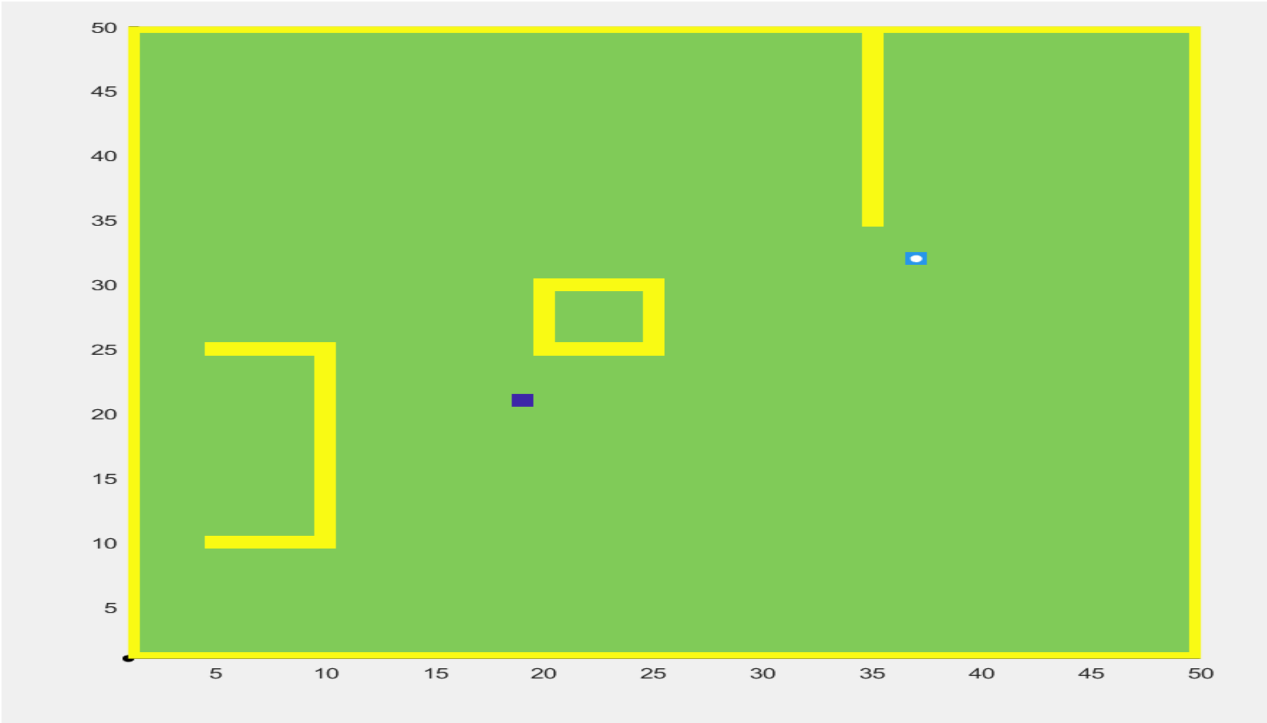


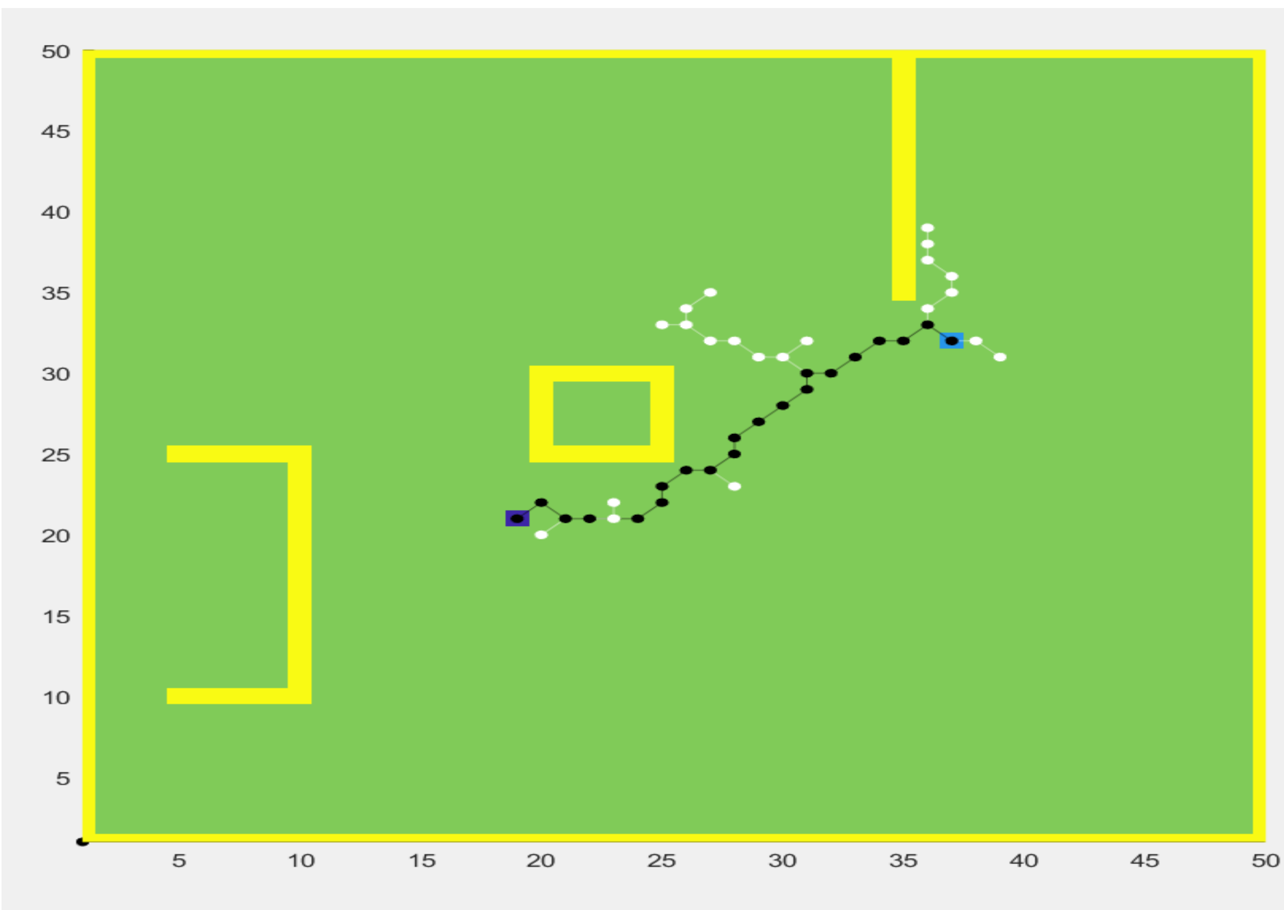


Simulation :-
https://drive.google.com/file/d/1cDZrQj3YeevnUdBdODGFpx1IDm-1Qf11/view?usp=share_link

Assignment 2 - PRM or RRT for finding a feasible path

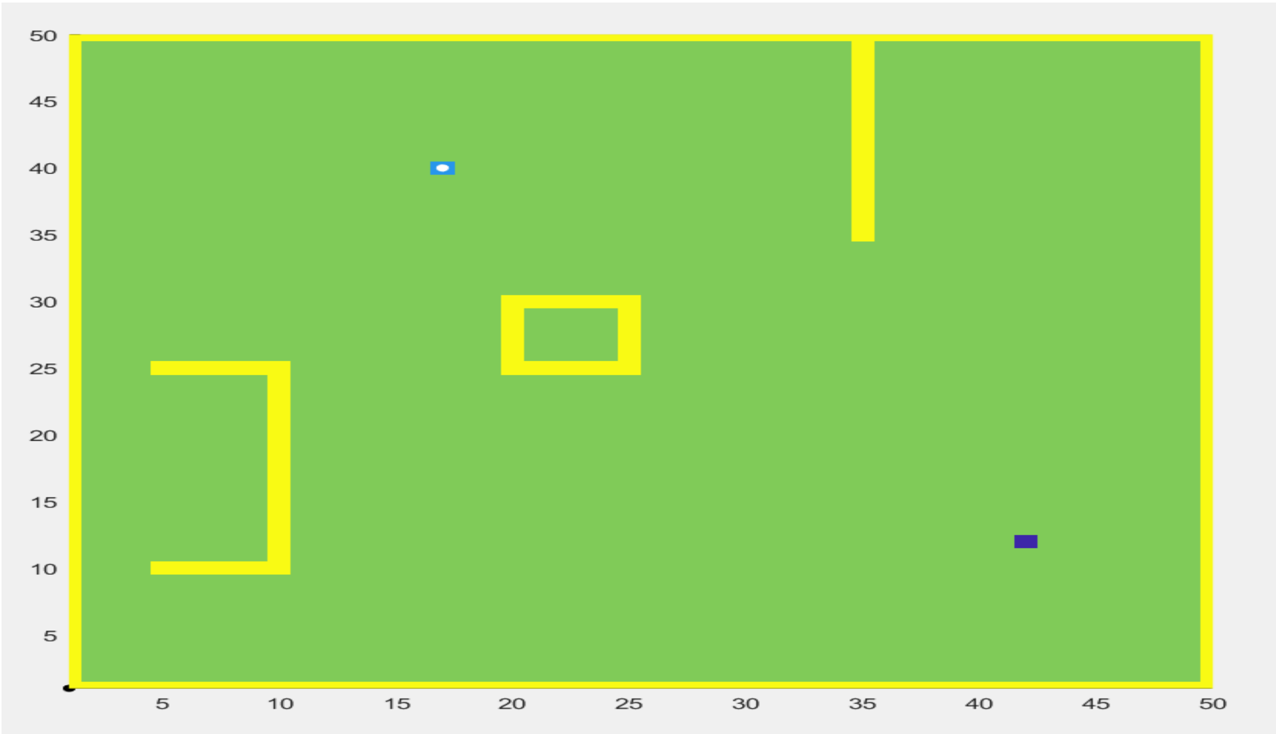
Multiple images showing different start and goal points

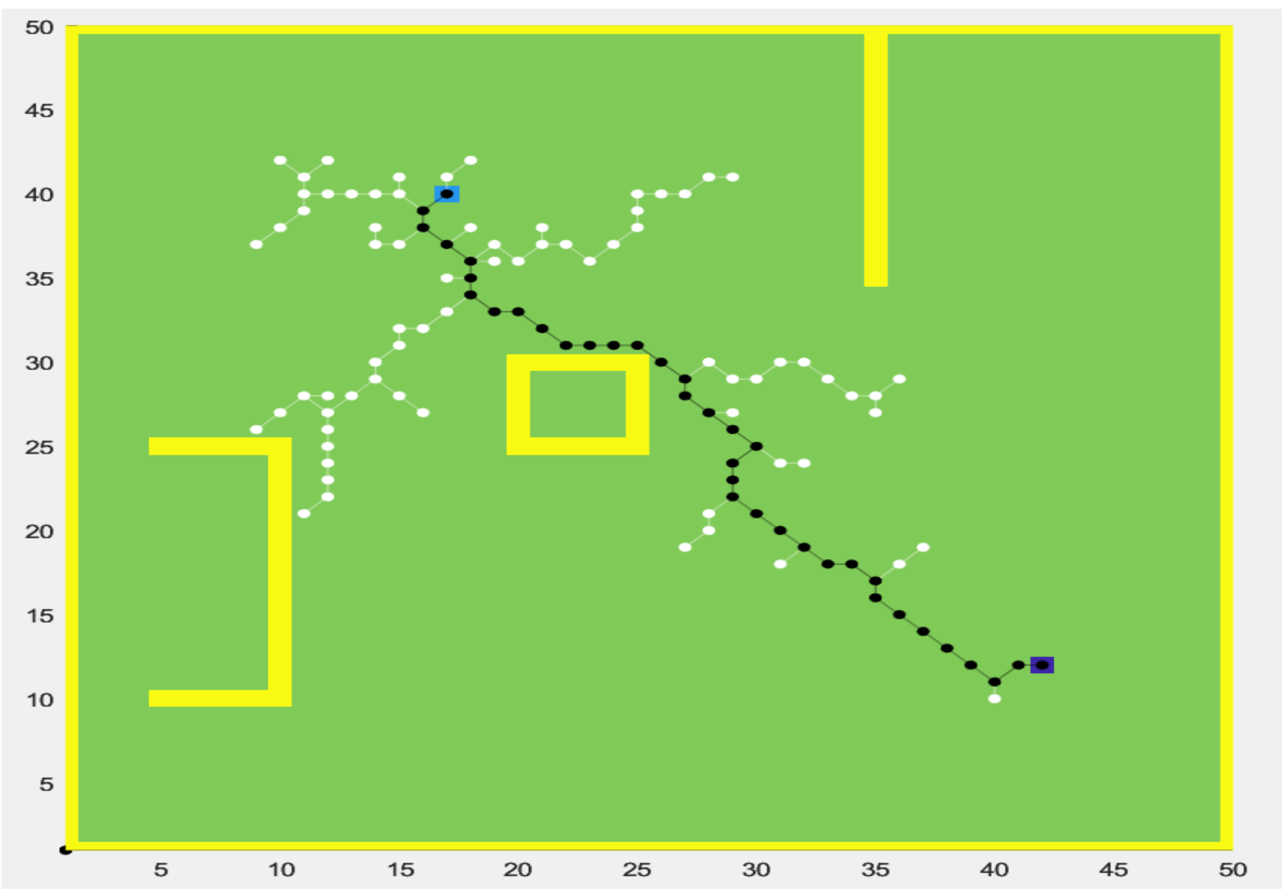




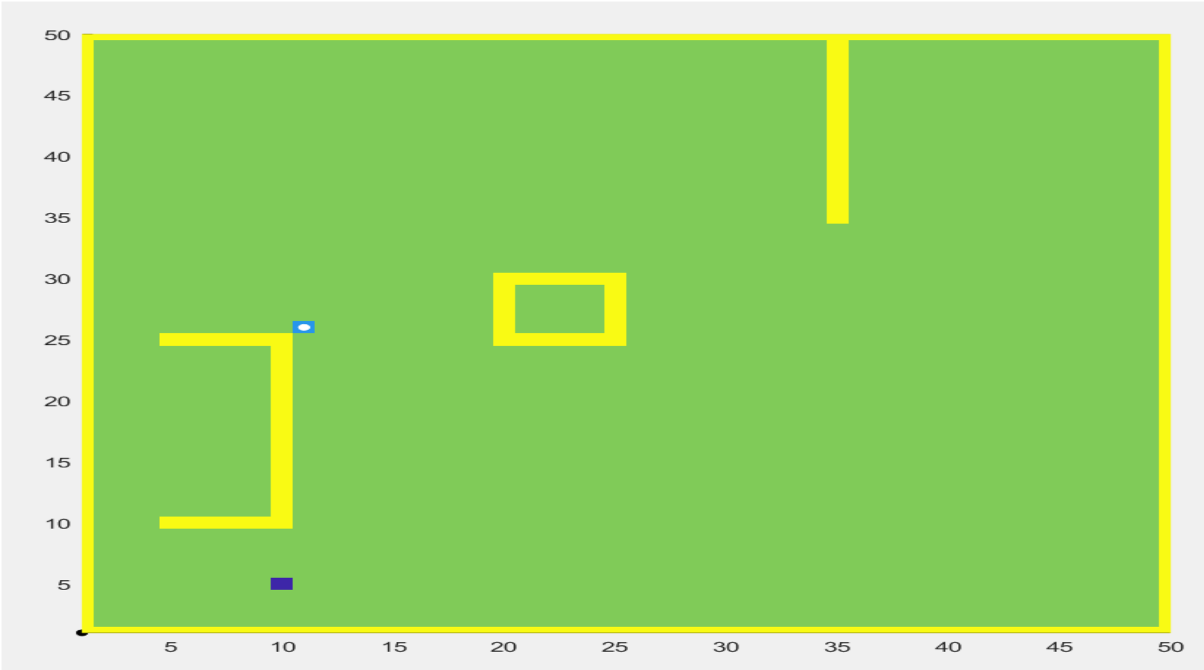
Assignment 2 - PRM or RRT for finding a feasible path

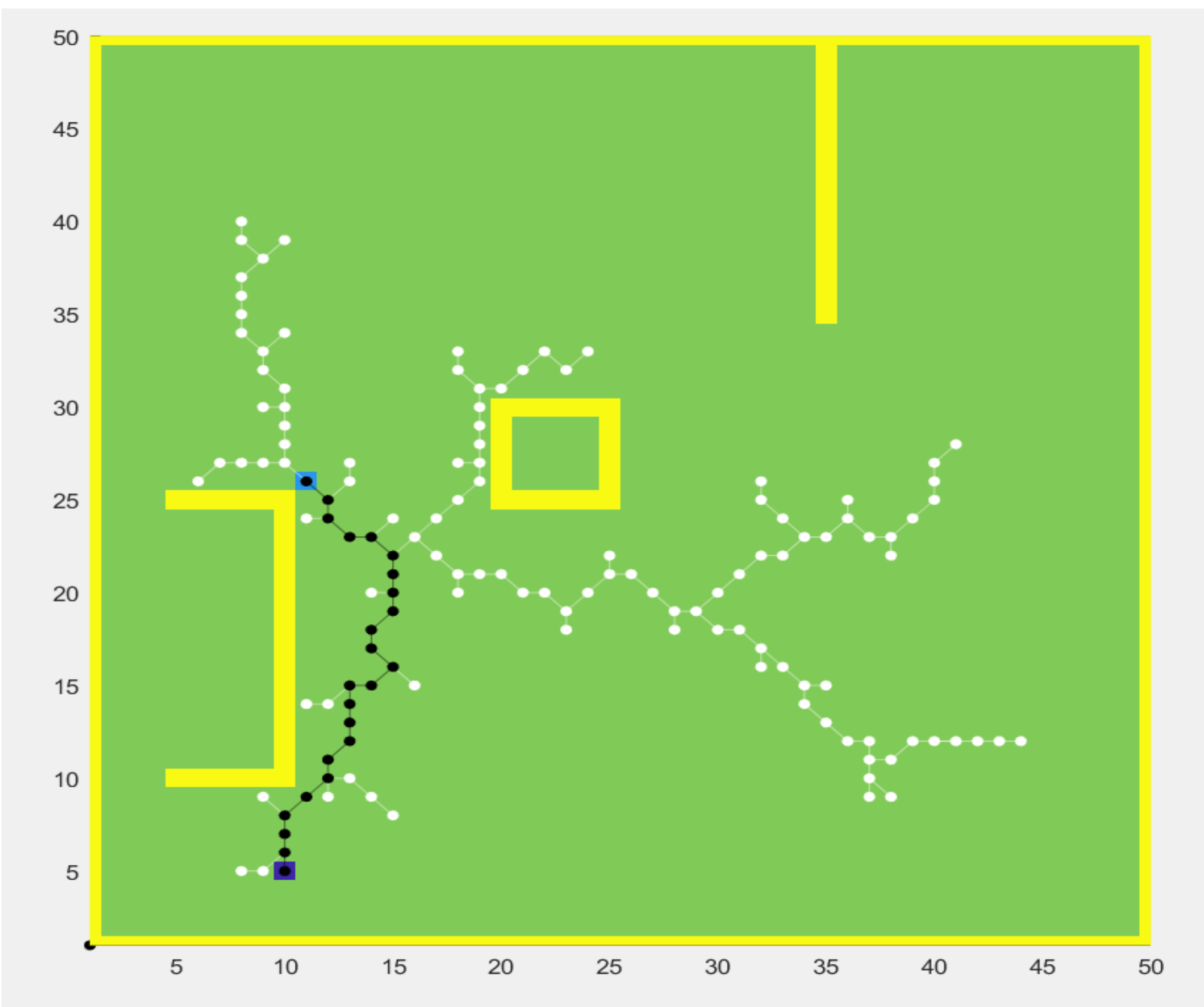
Multiple images showing different start and goal points





Assignment 2 - PRM or RRT for finding a feasible path





Assignment 2 - PRM or RRT for finding a feasible path

The full program

```
%import mathematic and static toolbox of matlab to run this code
%% intilizing
% two ways to set goal point or starting point
% 1st way :-random generating two points for goal and start
% Generate 2 random integers between 1 and 50
random_numbers = randi([5, 45], 1, 2);
random_numbers2 = randi([5, 45], 1, 2);
% randomly allocating robot's starting state and goal state
%new =  random_numbers;
%goal = random_numbers2;
%second way manually specify the cordinates of two points
% set here robot's starting state and goal state
new =[20,5]; % space is of 50-50 so x/y cordinate must be less than that
goal=[15,45];% space is of 50-50 so x/y cordinate must be less than that
% initialize map
hold on
map = init_map(new, goal);
imagesc(map)
set(gca,'YDir','normal')
%% RRT ALGORITHM
% initialize graph tree
node = 1;
source = node;
target = [];
nodes_x(node) = new(1);
nodes_y(node) = new(2);
rrt_graph = graph(source,target);
rrt_plot = plot(rrt_graph, 'w','XData', nodes_y, 'YData',
nodes_x,'NodeLabel',{});
%pause      %%uncomment it if you want to to see just the obstacle
iterator = 1;
%  stopping condition :- goal reached or enough time taken
while (any(new ~= goal) || iterator==1600)
   iterator = iterator + 1;

   % select direction state
   x_rand = select_state(goal,0.9,1);
   % select nearest neighbor to this current random state
   for node = 1:node
       near(node) =
pdist([nodes_x(node),nodes_y(node);x_rand(1),x_rand(2)],'euclidean');
   end
   [dist, nearest_node] = min(near);
   % state of the nearest neighbor
   x_near = [nodes_x(nearest_node), nodes_y(nearest_node)];
```

Assignment 2 - PRM or RRT for finding a feasible path

```
    % move towards x_rand position
    new = x_near + move(x_near,x_rand);
    % check if position is occupied
    if map(new(1), new(2)) ~= 1
        % check if the node already exists
        exists_node = false;
        for i=1:node
            if new(1) == nodes_x(node) && new(2) == nodes_y(node)
                exists_node = true;
                break
            end
        end
        if exists_node == false
            % add current state as a node to the graph
            node = node + 1;
            rrt_graph = addnode(rrt_graph,1);
            rrt_graph = addedge(rrt_graph,nearest_node,node);
            nodes_x(node) = new(1);
            nodes_y(node) = new(2);
        end
        if iterator == 1599
            disp('No Path found')
            pause
            break;
        end
    end

    delete(rrt_plot)
    rrt_plot = plot(rrt_graph, 'w','XData', nodes_y, 'YData',
nodes_x,'NodeLabel',{}, 'LineWidth', 0.5000, 'MarkerSize', 4);
    grid on
    pbaspect([1 1 1])
    xlim([1 50])
    ylim([1 50])
    pause(0.01)
end
hold off
pause
% finding shortest path
spath = shortestpath(rrt_graph,1,node);
highlight(rrt_plot,spath,'NodeColor','k','EdgeColor','k');
%% AUXILIARY FUNCTIONS
function x = select_state(x_goal,epsilon,dist)
    if rand<epsilon
        if dist == 1
            % from a uniform distribution
            x = [randi([1,50]), randi([1,50])];
        elseif dist == 2
            x(1) = random('Normal',25,7.5);
```

```
            x(2) = random('Normal',25,7.5);
            for i=1:2
               if x(i) < 1
                  x(i) = 1;
               elseif x(i) >50
                   x(i) = 50;
               end
            end
        elseif dist == 3
            x(1) = random('Rayleigh',x_goal(1));
            x(2) = random('Rayleigh',x_goal(2));
            for i=1:2
               if x(i) < 1
                  x(i) = 1;
               elseif x(i) >50
                   x(i) = 50;
               end
            end
        end
    else
        x = x_goal;
    end
end
function angle = find_orientation(source,target)
   target(1) = target(1)-source(1);
   target(2) = target(2)-source(2);
   angle = atan2(target(1),target(2));
   if angle < 0
       angle = angle + 2*pi;
   end
end
function delta = move(source,target)
   angle = find_orientation(source,target);
   delta(1) = sin(angle);
   delta(2) = cos(angle);
   for i = 1:2
       if 0 < delta(i) && delta(i) < 0.3535
           delta(i) = 0;
       elseif 0.3535 <= delta(i) && delta(i) < 1
           delta(i) = 1;
       elseif -0.3535 < delta(i) && delta(i) < 0
           delta(i) = 0;
       elseif -1 < delta(i) && delta(i) <= -0.3535
           delta(i) = -1;
       end
   end
end
function map = init_map(map_source, map_target)
   % free unnocuppied map
```

Assignment 2 - PRM or RRT for finding a feasible path

```
   map_x = 50;
   map_y = 50;
   for i = 1:map_x
       for j = 1:map_y
           map(i,j) = 0;
      end
   end

% adding 3 obstacles 1st is a 3 line cage
   for i = 10:25
       for j = 10:10
           map(i,j) = 1;
       end
   end
   for i = 25:25
       for j = 5:10
           map(i,j) = 1;
       end
   end

   for i = 10:10
       for j = 5:10
           map(i,j) = 1;
       end
   end
% 2nd obstacle is line obstacles
   for i = 35:50
       for j = 35:35
           map(i,j) = 1;
       end
   end
% 3rd obstacle is a rectangular obstacle
   for i = 25:30
       for j = 25:25
           map(i,j) = 1;
       end
   end

   for i = 25:25
       for j = 20:25
           map(i,j) = 1;
       end
   end
   for i = 30:30
       for j = 20:25
           map(i,j) = 1;
       end
   end
   for i= 25:30
```

Assignment 2 - PRM or RRT for finding a feasible path

```
        for j=20:20
            map(i,j) = 1;
        end
    end
% Walls bounding will be defiedn same as obtacle
    for i = 1:50
        for j = [1,50]
            map(i,j) = 1;
        end
    end
    for i = [1,50]
        for j = 1:50
            map(i,j) = 1;
        end
    end

    map(map_source(1),map_source(2)) = -1;
    map(map_target(1),map_target(2)) = -2;
end
```