```
clc;
clear;
close all;
% Robot Parameters
robotRadius = 1; % Radius of the robot
robotSpeed = 1; % Speed of the robot
% Workspace and Obstacles Setup
workspace = [0, 20, 0, 20]; % [x_min, x_max, y_min, y_max]
obstacles = [5, 5, 3; 15, 15, 4]; % [x_center, y_center, radius] for obstacles
% Start and Goal
start = [2, 2];
goal = [18, 18];
% RRT Parameters
maxNodes = 500; % Maximum number of nodes
stepSize = 1; % Step size between nodes
% Initialize RRT Tree
tree.nodes = start; % Start with the initial node
tree.parent = 0;
% Plot Environment
figure;
hold on;
axis([workspace(1) workspace(2) workspace(3) workspace(4)]);
axis equal;
grid on;
% Plot obstacles
theta = linspace(0, 2*pi, 100);
for i = 1:size(obstacles, 1)
    obstacle = obstacles(i, :);
    fill(obstacle(1) + obstacle(3)*cos(theta), obstacle(2) + obstacle(3)*sin(theta),
'r');
end
% Plot Start and Goal
plot(start(1), start(2), 'go', 'MarkerSize', 10, 'MarkerFaceColor', 'g');
plot(goal(1), goal(2), 'ro', 'MarkerSize', 10, 'MarkerFaceColor', 'r');
% RRT Algorithm Loop
for i = 1:maxNodes
    % Generate a random point in the workspace
    randPoint = [rand * (workspace(2) - workspace(1)) + workspace(1), ...
                 rand * (workspace(4) - workspace(3)) + workspace(3)];

    % Find the nearest node in the tree
    distances = sqrt((tree.nodes(:, 1) - randPoint(1)).^2 + (tree.nodes(:, 2) -
randPoint(2)).^2);
    [~, nearestIdx] = min(distances);
    nearestNode = tree.nodes(nearestIdx, :);

    % Calculate the new point (stepping towards the random point)
    direction = (randPoint - nearestNode) / norm(randPoint - nearestNode);
    newNode = nearestNode + direction * stepSize;

    % Check if the new node collides with any obstacle
    if ~isCollision(newNode, obstacles, robotRadius)
        tree.nodes = [tree.nodes; newNode]; % Add new node to the tree
        tree.parent = [tree.parent; nearestIdx]; % Record parent node

        % Plot the new edge
        plot([nearestNode(1), newNode(1)], [nearestNode(2), newNode(2)], 'b',
'LineWidth', 2);

        % Check if the new node is close enough to the goal
        if norm(newNode - goal) < stepSize
            disp('Goal reached!');
```

```
            break;
        end
    end
end
% Reconstruct the path from start to goal
path = goal;
currentNode = size(tree.nodes, 1);
while currentNode ~= 1
    parentNode = tree.parent(currentNode);
    path = [tree.nodes(parentNode, :); path];
    currentNode = parentNode;
end
% Plot the path
plot(path(:, 1), path(:, 2), 'k-', 'LineWidth', 3);
function collision = isCollision(point, obstacles, robotRadius)
    % Check if a point collides with any obstacle
    collision = false;
    for i = 1:size(obstacles, 1)
        dist = sqrt((point(1) - obstacles(i, 1))^2 + (point(2) - obstacles(i, 2))^2);
        if dist < obstacles(i, 3) + robotRadius
            collision = true;
            return;
        end
    end
end
```