

- **Environment 3.** One obstacle with center $c_1 = [0.2, 0.35]^T$ and radius $r_1 = 0.2$. A second obstacle with center $c_2 = [0.5, 0.3]^T$ and radius $r_2 = 0.2$. A third obstacle with center $c_3 = [0.7, 0.5]^T$ and radius $r_3 = 0.2$.

```
clear
close all

% Environment initiation
theta_start.coord = [0; 0];
theta_goal = [1; 1];

% Obstacle Parameters

% First obstacle center and radius
center = [0.2; 0.35];
radius = 0.2;

% Second obstacle center and radius
center2 = [0.5; 0.3];
radius2 = 0.2;

% Third obstacle center and radius
center3 = [0.7; 0.5];
radius3 = 0.2;

% Specifying parameters
epsilon = 0.1;
delta = 0.1;
N = 1000;

% Visualizing the environment
figure
grid on
hold on
axis([0, 1, 0, 1])
axis equal
viscircles(center', radius, 'Color', [0.5, 0.5, 0.5]);
viscircles(center2', radius2, 'Color', [0.7, 0.3, 0.3]); % Visualizing the second
obstacle
viscircles(center3', radius3, 'Color', [0.3, 0.5, 0.7]); % Visualizing the third
obstacle
plot(0, 0, 'ko', 'MarkerFaceColor', 'k')
plot(1, 1, 'ko', 'MarkerFaceColor', 'k')

% Initializing the tree
theta_start.parent = 0;
G(1) = theta_start;
```

```

for idx = 1:N

    % Stop if the last node in G is close to theta_goal
    if norm(G(end).coord - theta_goal) < epsilon
        break
    end

    % Sample random joint position
    theta_rand = rand(2,1);
    % % theta_rand = theta_goal;
    % % if rand() < 0.8
    % %     theta_rand = rand(2,1);
    % % end

    % Find node in G nearest to theta_rand
    min_dist = inf;
    theta_near_index = 0;

    for jdx = 1:length(G)
        coord = G(jdx).coord;
        dist = norm(theta_rand - coord);
        if dist < min_dist
            min_dist = dist;
            theta_near_index = jdx;
        end
    end

    theta_near = G(theta_near_index);

    % take a step from theta_near towards theta_rand
    vec_to_rand = theta_rand - theta_near.coord;
    dist_to_rand = norm(vec_to_rand);
    if dist_to_rand < delta
        theta_new.coord = theta_rand;
    else
        theta_new.coord = theta_near.coord + delta * ...
            vec_to_rand/dist_to_rand;
    end

    % Check if theta_new is collision free with all obstacles
    dist_to_obs1 = norm(theta_new.coord - center);
    dist_to_obs2 = norm(theta_new.coord - center2);
    dist_to_obs3 = norm(theta_new.coord - center3);
    if dist_to_obs1 < radius || dist_to_obs2 < radius2 || dist_to_obs3 < radius3
        continue
    end

    % If collision free, add theta_new to tree with parent theta_near
    theta_new.parent = theta_near_index;
    G = [G, theta_new];
end

```

```

% Plot node and edge
plot(theta_new.coord(1), theta_new.coord(2), 'o', 'Color', [0.5, 0.5, 0.5], ...
'MarkerFaceColor', [0.5, 0.5, 0.5])
line([theta_near.coord(1), theta_new.coord(1)], [theta_near.coord(2), ...
theta_new.coord(2)], 'Color', 'k', 'LineWidth', 2);
drawnow

end

% Work backwards from the final node to the root of the tree
child_theta = G(end);
while child_theta.parent ~= 0

    parent_theta_index = child_theta.parent;
    parent_theta = G(parent_theta_index);
    line([child_theta.coord(1), parent_theta.coord(1)], ...
        [child_theta.coord(2), parent_theta.coord(2)], ...
        'Color', [1, 0.5, 0], 'LineWidth', 3);
    child_theta = parent_theta;
end

```

