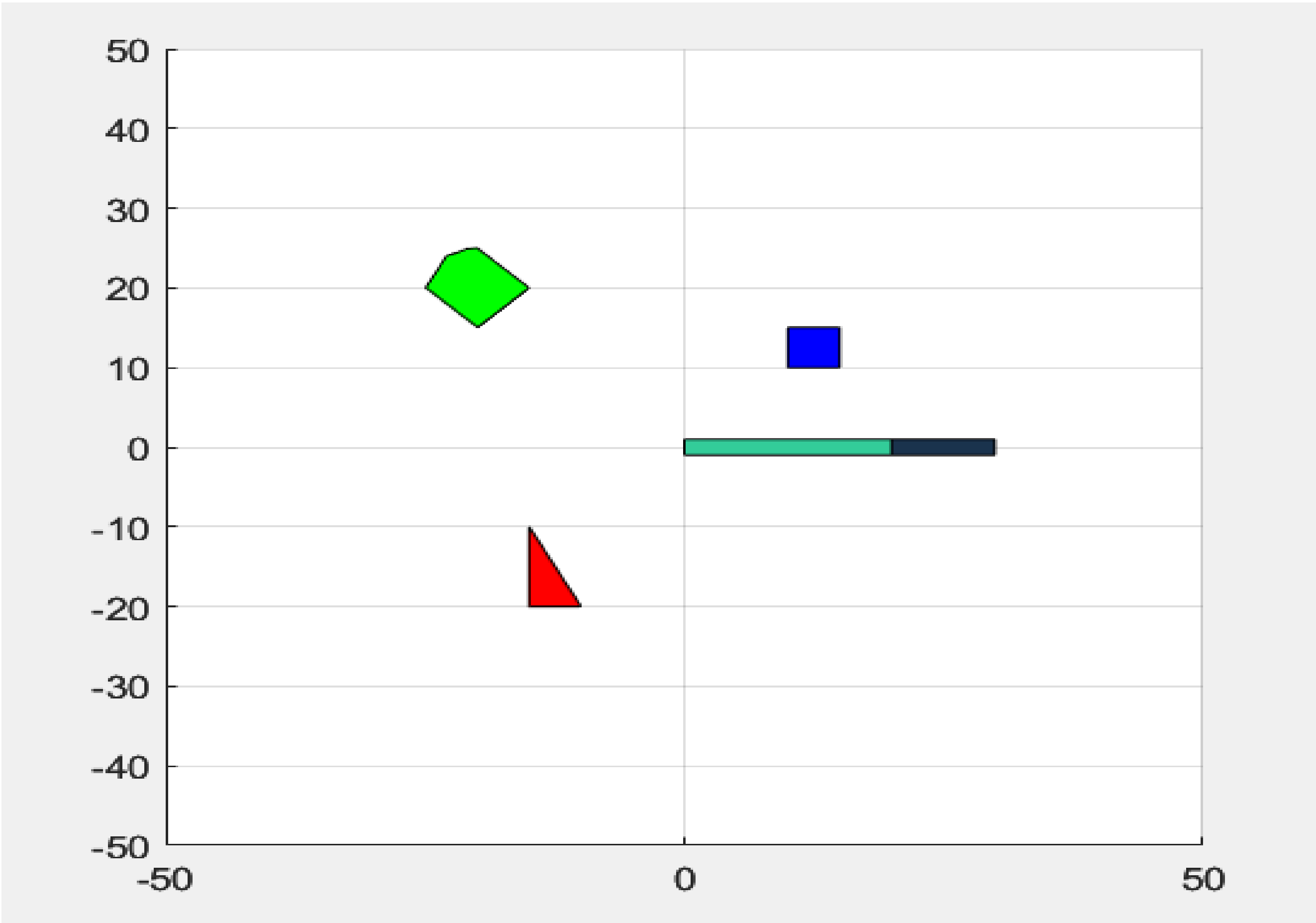
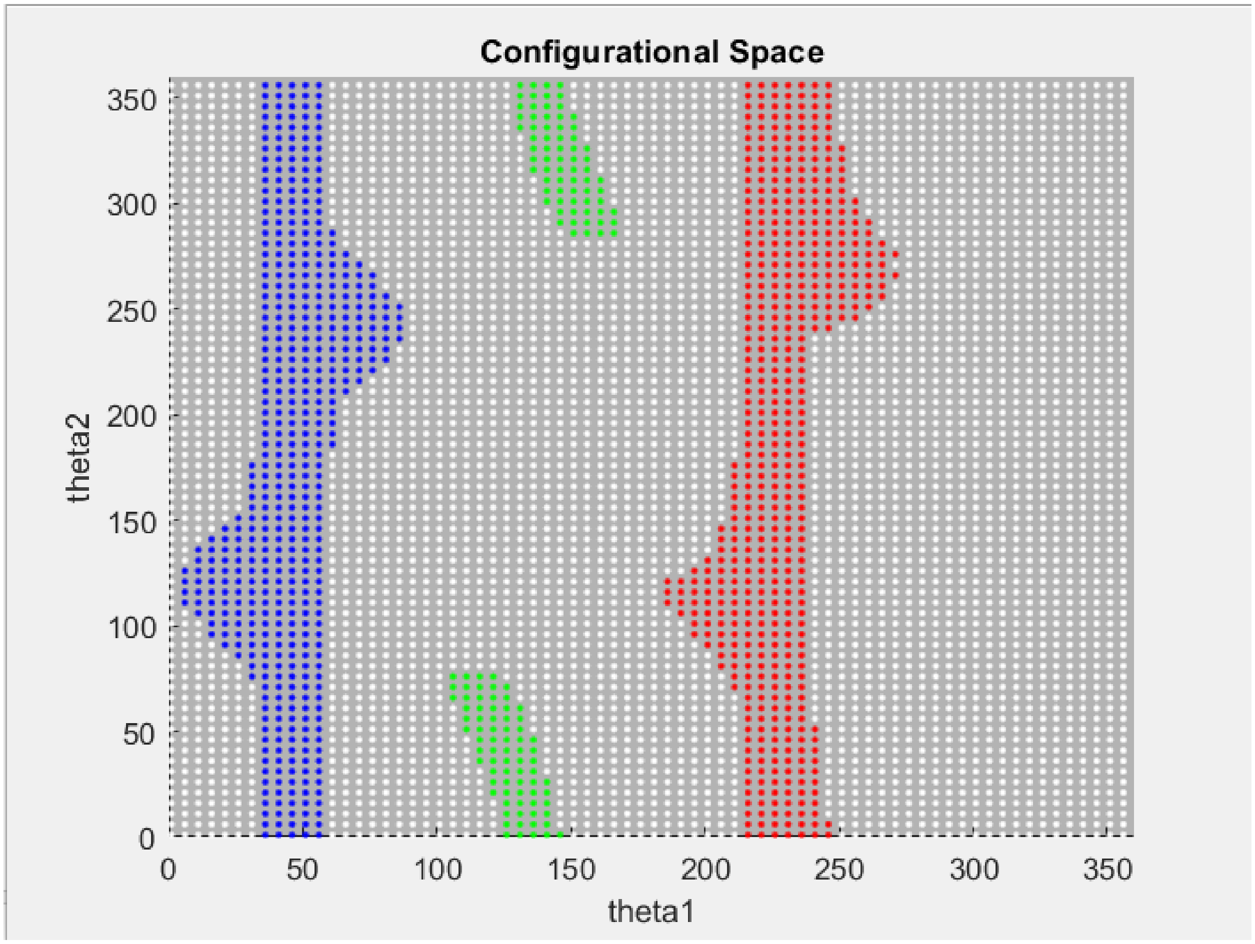
ME766A: ROBOT MOTION PLANNING



Cartesian space (x,y) showing the links, obstacles

Simulation
https://drive.google.com/file/d/18ZNLjrsiEnyCCKAS0kclkTwM5MI19B7T/view?usp=sharing



The C-space (theta1 verses theta2), showing the free space as white and obstacle space having the respective obstacle colour.

Simulation
https://drive.google.com/file/d/1Ls6T97HZYVHfR4SJQ0r4QYga3Rd5i3ZH/view?usp=sharing

ME766A: ROBOT MOTION PLANNING

This code is an implementation of a basic robot arm simulator with two links (represented as rectangles) and three obstacles (represented as polygons) in a 2D workspace.
The objective of this simulator is to check if any part of the robot arm intersects with any of the obstacles as it moves through the workspace. And Plot the C space.
The robot arm is defined by the lengths and widths of its two links, as well as the initial joint angles of the two links. These values can be changed in the code to modify the robot arm's configuration.
The robot arm is then rotated at each iteration of the nested loop by small increments, and the positions of the links are computed using trigonometric functions. The new positions of the links are plotted at each iteration of the loop, creating the impression of a moving robot arm.
The workspace is defined by the three obstacles, which are represented as polygons with a defined set of vertices.
The simulator checks if any part of the robot arm intersects with any of the obstacles using the **overlaps function**, which returns a logical value indicating whether or not two polygons overlap.
The configuration space of the robot arm is the space of all possible joint angles. The simulator checks if any part of the robot arm intersects with any of the obstacles at each configuration of the robot arm, and marks the corresponding point in a separate plot that shows the robot arm's configuration space.

```
% The obstacles are represented as polygons with their vertices
obstacle1 = [-15 -10;-10 -20;-15 -20]; %3 points traingle
obstacle2 = [10 10;15 10;15 15;10,15]; %4 points square
obstacle3 = [-20,25;-15,20;-20,15;-25,20;-23,24;-21,24.9]; % 6 points polygon
x1 = 0;  % coordinate of ground or
y1 = 0;  % or point at which one arm is fixed
% two geometries that represent links of a robot arm.
```

ME766A: ROBOT MOTION PLANNING

```
% The links of the% robot arm are represented as rectangles with specified
dimensions.
%diemnsion of link; one being thinner than other
% for clear visulation of difference
W1 = 10;
L1 = 1;
W2 = 5;
L2 = 1;
link1_geometry = [-W1, -L1; W1, -L1; W1, L1; -W1, L1];
link2_geometry = [-W2, -L2; W2, -L2; W2, L2; -W2, L2];
% The code first sets up the plot by specifying the axis limits and
% aspect ratio and adding a grid
axis([-50 50 -50 50]);
daspect([10 10 10]);
grid on;
hold on;
% It then adds the obstacles and
% links to the plot and saves handles to these objects for later use.
fill(obstacle1(:,1), obstacle1(:,2), 'r');
fill(obstacle2(:,1), obstacle2(:,2), 'b');
fill(obstacle3(:,1), obstacle3(:,2), 'g');
l1handle = fill(link1_geometry(:,1), link1_geometry(:,2), [.2 .8 .6]);
l2handle = fill(link2_geometry(:,1), link2_geometry(:,2), [.1 .2 .3]);
ob1 = polyshape(obstacle1(:,1),obstacle1(:,2));
ob2 = polyshape(obstacle2(:,1),obstacle2(:,2));
ob3 = polyshape(obstacle3(:,1),obstacle3(:,2));
hold off;
% The code then enters a nested loop that iterates over all possible
% pairs of joint angles for the robot arm. For each pair of joint
% angles, the code computes the positions of the links and updates
% the plot accordingly.
th1 = 0; % initial angle of arm 1
th2 = 0;  % initial angle of arm 2
for theta1 = th1:5:360+th1
   for theta2 = th2:5:360+th2
        %rotate link about origin
       rotated_l1_geometry = link1_geometry*[cosd(theta1) sind(theta1);...
           -sind(theta1), cosd(theta1)];
       rotated_l2_geometry = link2_geometry*[cosd(theta2+theta1)
sind(theta2+theta1);...
           -sind(theta2+theta1), cosd(theta2+theta1)];
       X1=x1+W1*cosd(theta1);
       Y1=y1+W1*sind(theta1);
       set(l1handle, 'xdata', X1+rotated_l1_geometry(:,1),...
           'ydata', Y1+rotated_l1_geometry(:,2));
       l1 = polyshape(X1+rotated_l1_geometry(:,1),Y1+rotated_l1_geometry(:,2));
%for Overlap chaecking

       x2 = x1+2*W1*cosd(theta1);
```

```
        y2 = y1+2*W1*sind(theta1);
        X2=x2+W2*cosd(theta1+theta2);
        Y2=y2+W2*sind(theta1+theta2);
        set(l2handle, 'xdata', X2+rotated_l2_geometry(:,1),...
            'ydata', Y2+rotated_l2_geometry(:,2));
        l2 = polyshape(X2+rotated_l2_geometry(:,1),
Y2+rotated_l2_geometry(:,2));     %for Overlap chaecking

%It also computes the position of the end
% effector of the robot arm and plots it in a separate window,
% creating an animation of the end effector's motion.
% The code then checks if any part of the robot arm
% intersects with any of the obstacles. This is done
% by creating polygon objects for the links and obstacles
% using the polyshape function, and then using the overlaps
% function to check if the polygons overlap.
% If any part of the robot arm intersects with
% an obstacle, the code marks the corresponding point
% in a separate plot that shows the robot arm's configuration
% space, which is the space of possible joint angles.
        %Checking obstacle intersection
        counter = 0;
        if overlaps(l1,ob1) || overlaps(l2,ob1)
            counter = 1;
        elseif overlaps(l1,ob2) || overlaps(l2,ob2)
            counter = 2;
        elseif overlaps(l1,ob3) || overlaps(l2,ob3)
            counter = 3;
        end
        counter;

% Points that correspond to collisions are marked in red, while points
% that correspond to non-collisions are marked in blue.
% The code then repeats this process for all possible
% joint angles, effectively creating a map of the robot
% arm's configuration space and identifying regions where
% the robot arm would collide with obstacles%
        %Configuration Space
        figure(3)
        set(gca,'Color',[0.7 0.7 0.7])
        title('Configurational Space')
        xlabel('theta1')
        ylabel('theta2')
        x = theta1+1-th1;
        y = theta2+1-th2;
        hold on
        if counter == 1
            plot(x,y,'r.')
        elseif counter== 2
```

```
            plot(x,y,'b.')
        elseif counter== 3
            plot(x,y,'g.')
        else
            plot(x,y,'w.')
        end
        axis([0 360 0 360])
        [theta1,theta2]
        drawnow  limitrate
    end
end
% Overall, the code provides a visualization
% and analysis of the robot arm's workspace and
% configuration space, and can be used to help design and optimize robot arm
movements.
```