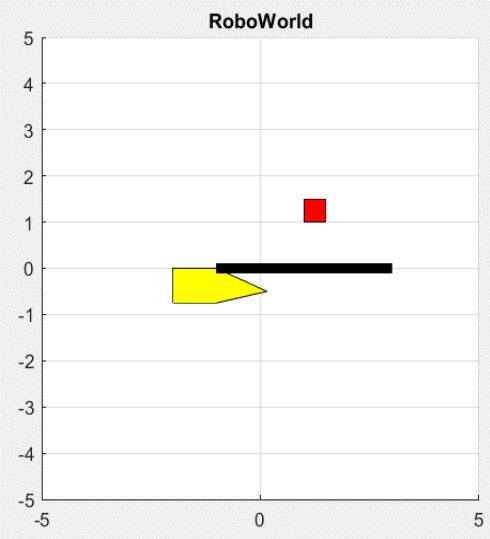
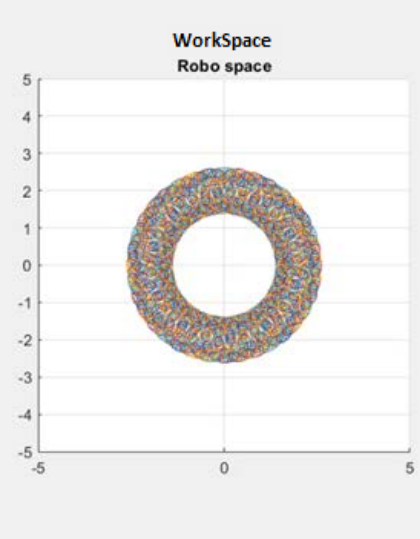
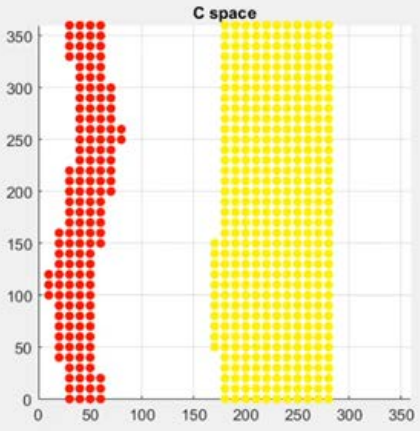
ROBOT MOTION PLANNING (ME510)

Assignment 1







ADITYA SHAH (Roll:2011mt02)

Department of Mechatronics

For the Academic Year 2021 - 2022

Code

```
clc;
clear;

obstacle1 = [1 1;1.5 1;1.5 1.5;1,1.5];
obstacle2 = [-2 0;-1 0;0.15 -0.5;-1 -0.75;-2 -0.75];
%theta1 = 20;
%theta2 = 30;


 x1 = 0;
 y1 = 0;
 ct=1;


 W1 = 1;%half width of link 1
 L1 = 0.1;%half thickness of link 1

 W2 = 0.5;%half width of link 2
 L2 = 0.1;%half thickness of link 2
 link1_geometry = [-W1, -L1; W1, -L1; W1, L1; -W1, L1];
 link2_geometry = [-W2, -L2; W2, -L2; W2, L2; -W2, L2];
 for theta1=0:10:360
     for theta2=0:10:360
 h1=subplot(1,3,1);
 title('RoboWorld');


 axis([-5 5 -5 5]);
 daspect([1 1 1]);
 grid on;
 hold on

 fill(obstacle1(:,1), obstacle1(:,2), 'r');
 fill(obstacle2(:,1), obstacle2(:,2), 'y');
 l1handle = fill(link1_geometry(:,1), link1_geometry(:,2), 'k');

 l2handle = fill(link2_geometry(:,1), link2_geometry(:,2), 'k');
 hold off


 pause(0.001)
 M(ct)=getframe(gcf);
 ct=ct+1;
 hold off


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

```
56.  y2 = y1+2*W1*sind(theta1);
57.
58.  X2=x2+W2*cosd(theta1+theta2);
59.  Y2=y2+W2*sind(theta1+theta2);
60.  set(l1handle, 'xdata', X1+rotated_l1_geometry(:,1),...
61.      'ydata', Y1+rotated_l1_geometry(:,2));
62.
63.
64.
65.  set(l2handle, 'xdata', X2+rotated_l2_geometry(:,1),...
66.      'ydata', Y2+rotated_l2_geometry(:,2));
67.  subplot(1,3,2);
68.  title('Robo-WorkSpace');
69.  xlabel('X ')
70.  ylabel('Y')
71.  hold on
72.  axis([-5 5 -5 5]);
73.  daspect([1 1 1]);
74.  grid on;
75.  scatter(X2,Y2)
76.  hbase.Visible = 'off';
77.
78.
79.  subplot(1,3,3);
80.  title('C-Space');
81.  xlabel('θ1(Degree)')
82.  ylabel('θ2(Degree)')
83.  hold on
84.  axis([0 360 0 360]);
85.  daspect([1 1 1]);
86.  grid on;
87.
88.  %red color ---------------------------------------------
89.  if (X2>0.65) && (X2<1.95) && (Y2>0.78) && (Y2<1.85)
90.
91.       scatter(theta1,theta2,[],'r','filled')
92.   end
93.   if (X1>0.75) && (X1<1.88) && (Y1>0.78) && (Y1<1.85)
94.       scatter(theta1,theta2,[],'r','filled')
95.   end
96.    if (((X2+X1)/2)>0.75) && (((X2+X1)/2)<1.85) && (((Y2+Y1)/2)>0.78) &&
   (((Y2+Y1)/2)<1.85)
97.
98.       scatter(theta1,theta2,[],'r','filled')
99.   end
100.  if ((X1/2)>0.75) && ((X1/2)<1.85) && ((Y1/2)>0.78) && ((Y1/2)<1.85)
101.      scatter(theta1,theta2,[],'r','filled')
102.  end
103.   if ((X1*0.75)>0.55) && ((X1*0.75)<1.95) && ((Y1*0.75)>0.78) &&
   ((Y1*0.75)<1.85)
104.      scatter(theta1,theta2,[],'r','filled')
105.   end
106.   if (((X2+X1)*0.75)>0.75) && (((X2+X1)*0.75)<1.95) &&
   (((Y2+Y1)*0.75)>0.78) && (((Y2+Y1)*0.75)<1.85)
107.      scatter(theta1,theta2,[],'r','filled')
108.  end
109.  %yellow color ----------------------------------------------
```

```
110. if (X2>-2.4) && (X2<0.20) && (Y2>-0.785) && (Y2<0.28)
111.
112.      scatter(theta1,theta2,[],'y','filled')
113.  end
114.  if (X1>-2.4) && (X1<0.17) && (Y1>-0.785) && (Y1<0.14)
115.      scatter(theta1,theta2,[],'y','filled')
116.  end
117.   if (((X2+X1)/2)>-2.4) && (((X2+X1)/2)<0.17) && (((Y2+Y1)/2)>-0.785)
   && (((Y2+Y1)/2)<0.19)
118.
119.      scatter(theta1,theta2,[],'y','filled')
120.  end
121.  if ((X1/2)>-2.4) && ((X1/2)<0.17) && ((Y1/2)>-0.785) && ((Y1/2)<0.14)
122.      scatter(theta1,theta2,[],'y','filled')
123.  end
124.   if ((X1*0.75)>-2.4) && ((X1*0.75)<0.17) && ((Y1*0.75)>-0.785) &&
   ((Y1*0.75)<0.18)
125.      scatter(theta1,theta2,[],'y','filled')
126.   end
127.   if (((X2+X1)*0.75)>-2.4) && (((X2+X1)*0.75)<0.17) &&
   (((Y2+Y1)*0.75)>-0.785) && (((Y2+Y1)*0.75)<0.20)
128.      scatter(theta1,theta2,[],'y','filled')
129.   end
130.
131.     end
132.    cla(h1)
133. end
134.
```

Two-joint planar robot arm:

- If L1 and L2 are known then specifying θ_1 and θ_2 will determine all the points on the robot.
- Thus, configuration is

$q = (\theta_1 , \theta_2)$

Each θ_i correspond to a point on a unit circle S^1

So, Configuration space is $S^1 \times S^1 = T^1$ - 2D torus

The configuration space plotted in MATLAB as 2d plane ,can be visualized as 2D torus by folding the plane such that θ_1=0° meet with θ_1 =360° (horizontal folding)and similarly for θ_2

i.e folding vertically such that θ_2=0° meet with θ_2 =360°

because after 360°,angle just repeats as after 0°.

(360 ^{o} +θ = θ),which can be visualized in 2nd plot of Workspace, depicting the robot end effector workspace.

*For MATLAB coding we have use some offset w.r.t to both obstacles boundary, so the response is good (Conservative Approach),as we have used discreate angle with a step of 10^{o} and if it collide with Obstacles, it can be plotted in C-Space correctly as obstacles configuration space with corresponding colour of obstacles.

In this ,Obstacles created and filled with different colours to depict them easily in C-Space and to get Real World Animation, I have used the set handle command to get the arms at correct position of theta1 and theta2 using the rotation transformation matrix and giving an translation to (x1,y1) and (x2,y2) to get (X1,Y2) and (X2,Y2) for proper viewing ,so 1st arm is appended to origin and the 2nd arm is appended correctly to 1st one (as given is Cpace.m file) and then the end effector is being plotted using scatter command and when the two-joint planar robot arm overlap/collide with any obstacles ,it plot the corresponding theta1 and theta 2 in Cspace as Obstacles Configuration space with corresponding obstacles color, keeping the free space as white/no colour.

Because of step angle used (10^{o}) ,the workspace has some gaps in-between, but it should be continuous i.e. subset of 2-d Real Space(R^2).

One more Observation is that, viewing the robot in real space and finding the path can be hectic, but in C-Space ,we just need to find the connected path between start position and goal position, avoiding the obstacles Configuration space.

In C-Space, any two-point connected, even if they are neighbour, there lie many angles in between, which may not be connected. So, it's not feasible to find a path in C-Space as a straight line, because any two neighbouring pixels in C-Space is actually more curved rather then connected by a straight line.

So, C-Space is basically a Conceptual tool and it's help us in finding a path in a space, where Robot Configuration and Obstacles Configuration is very easy to visualize, but practically it's not feasible because of computational complexity and also to find every possibility of configuration it's not practical.

To View the Simulation Video, Plot and Code, click on the following link:

Assignment -1