# Table of Contents



```
% Name - Vasu Bansal
% Roll No. - 160776
% Course - ME766

clc;
clear;
% Length of links from user
l1 = input('Enter the length of link 1 : ');
l2 = input('Enter the length of link 2 : ');
origin = [0,0,0];
obstacle_ = initObstacles();
draw = drawObstacles(obstacle_);
% D-H notation parameters in form of vectors
theta1 = 0;
theta2 = 0;
theta3 = 0;
a = [0 0 l1];
alpha = [0 -90 0];
d = [0 0 0];
%--------------------------------------------------------------------------
figure(2); % Specifically for figure two
daspect([1 1 1]); % So that axis remain equal in size
axis([0 360 0 360 0 360]); % Set axis limits
startingAngle = 0;
endAngle = 360;
step = 20;
for i = startingAngle:step:endAngle
    theta1 = i*pi/180;
    for j = startingAngle:step:endAngle
        theta2 = j*pi/180;
        for k = startingAngle:step:endAngle
            theta3 = k*pi/180;
            theta = [theta1 theta2 theta3];
            T1 = [cos(theta(1)),              -sin(theta(1))
    ,0              , a(1);
                        sin(theta(1))*cos(alpha(1)),
 cos(theta(1))*cos(alpha(1)), -sin(alpha(1)), -sin(alpha(1))*d(1);
                        sin(theta(1))*sin(alpha(1)),
 cos(theta(1))*cos(alpha(1)), -cos(alpha(1)), -cos(alpha(1))*d(1);
                        0, 0, 0, 1];
```

```
            T2 = [cos(theta(2)),              -sin(theta(2))
   ,0              , a(2);
                        sin(theta(2))*cos(alpha(2)),
 cos(theta(2))*cos(alpha(2)), -sin(alpha(2)), -sin(alpha(2))*d(2);
                        sin(theta(2))*sin(alpha(2)),
 cos(theta(2))*cos(alpha(2)), -cos(alpha(2)), -cos(alpha(2))*d(2);
                        0, 0, 0, 1];
            T3 = [cos(theta(3)),              -sin(theta(3))
   ,0              , a(3);
                        sin(theta(3))*cos(alpha(3)),
 cos(theta(3))*cos(alpha(3)), -sin(alpha(3)), -sin(alpha(3))*d(3);
                        sin(theta(3))*sin(alpha(3)),
 cos(theta(3))*cos(alpha(3)), -cos(alpha(3)), -cos(alpha(3))*d(3);
                        0, 0, 0, 1];
            end
            T01 = T1*T2; % Transformatin matrices for links 1 and 2
            T02 = T01*T3;
            L1 = linspace(0,l1,20);
            L2 = linspace(0,l2,20);
            for n = 1:20
                X1(:,n) = T01*[L1(n) 0 0 1]'; % X1 and X2 are
 coordinates of links in fixed frame
                X2(:,n) = T02*[L2(n) 0 0 1]';
            end
            y = isCollision(X1, X2, obstacle_);
            figure(2);
            if(y==0)
                plot3(i,j,k,'o','color',[1 1 0],'MarkerSize',8);
                hold on;
            else
                plot3(i,j,k,'o','color',draw(:,y)','MarkerSize',8);
                hold on;
            end
            pause(0.000001);
            grid on;
        end
    end
end
xlabel('theta1');
ylabel('theta2');
zlabel('theta3');
%--------------------------------------------------------------------------
% This function was obtained from https://in.mathworks.com/
matlabcentral/fileexchange/25559-draw-cuboid?
focused=5171830&tab=function
function [CuboidHandle, verts, facs] = DrawCuboid(varargin)

% Draw Cuboid
% Draw a Cuboid using 8 rectangular faces. Places Cuboid Into Current
 Figure
%
% Inputs (SL, CV, EA, colr, alph)
% ----------------
% SL    - [X;Y;Z] Length of Cuboid Side (SL - SideLength)
```

```
% CV    - [X;Y;Z] Center of volume
% EA    - [Yaw(Z-axis);Pitch(y-axis);Roll(x-axis)] Euler/Rotation
 angles [radians]
% colr  - Color of cuboid; string (ex. 'r','b','g') or vector [R G B]
% alph  - Alpha transparency value of cuboid
%
% Outputs (CuboidHandle, verts, facs)
% ----------------
% CuboidHandle  = Handle for Patch Object
% verts         = 3x8 XYZ Vertices
% facs          = 6x4 Order of faces
%
```

# Number of inputs and assign defaults if not specified

Define Default Values

```
SL = [1;1;1];   CV = [0;0;0];   EA = [0;0;0];   colr = [0 1 0];   alph
 = 0.5;
switch nargin
    case 0
        % All Inputs Empty Using Default Values
    case 1
        SL = varargin{1};
    case 2
        SL = varargin{1};   CV = varargin{2};
    case 3
        SL = varargin{1};   CV = varargin{2};   EA = varargin{3};
    case 4
        SL = varargin{1};   CV = varargin{2};   EA = varargin{3};
 colr = varargin{4};
    case 5
        SL = varargin{1};   CV = varargin{2};   EA = varargin{3};
 colr = varargin{4};     alph = varargin{5};
    otherwise
        error('Invalid number of inputs')
end
```

# Form ZYX Rotation Matrix

[From Wikipedia Oct-11-2009 1:30 AM] http://en.wikipedia.org/wiki/Euler_Angles Calculate Sines and Cosines

```
c1 = cos(EA(1)); s1 = sin(EA(1));
c2 = cos(EA(2)); s2 = sin(EA(2));
c3 = cos(EA(3)); s3 = sin(EA(3));
% Calculate Matrix
R = [c1*c2           -c2*s1          s2
     c3*s1+c1*s2*s3  c1*c3-s1*s2*s3  -c2*s3
     s1*s3-c1*c3*s2  c3*s1*s2+c1*s3  c2*c3]';
```

# Create Vertices

```
x = 0.5*SL(1)*[-1 1 1 -1 -1 1 1 -1]';
y = 0.5*SL(2)*[1 1 1 1 -1 -1 -1 -1]';
z = 0.5*SL(3)*[-1 -1 1 1 1 1 -1 -1]';
```

# Create Faces

```
facs = [1 2 3 4
        5 6 7 8
        4 3 6 5
        3 2 7 6
        2 1 8 7
        1 4 5 8];
```

# Rotate and Translate Vertices

```
verts = zeros(3,8);
for i = 1:8
    verts(1:3,i) = R*[x(i);y(i);z(i)]+R*CV;
end
```

# Draw Patch Object

```
CuboidHandle =
 patch('Faces',facs,'Vertices',verts','FaceColor',colr,'FaceAlpha',alph);
%view(3);

end
%--------------------------------------------------------------------------
function draw = drawObstacles(obstacle)
figure(1);
draw = [];
% The sphere function generates the x-, y-, and z- coordinates of a
 unit sphere for use with surf and mesh.
% sphere generates a sphere consisting of 20-by-20 faces.
% Reference - https://in.mathworks.com/help/matlab/ref/sphere.html

for i=1:obstacle.num
    if obstacle.type(i)==0 % If the obstacle is sphere
        [x,y,z] = sphere;
        x = x*obstacle.l(i);
        y = y*obstacle.l(i);
        z = z*obstacle.l(i);
        % surf command is used to plot in 3D
        hs=surf(x+obstacle.cx(i),y+obstacle.cy(i),z+obstacle.cz(i));
        hold on;
        color = [rand rand rand];
        set(hs,'FaceColor',color,'FaceAlpha',0.5,'EdgeColor','none');
        daspect([1 1 1]);
```

```
        axis([-20 20 -20 20 -20 20]);
    elseif (obstacle.type(i)==1) % If the obstacle to be drawn is
 Cuboid
        % This portion is already explained in DrawCuboid.m
        SL = [obstacle.l(i); obstacle.w(i); obstacle.h(i)];
        CV = [obstacle.cx(i); obstacle.cy(i); obstacle.cz(i)];
        color = [rand rand rand];
        alph = 0.5;
        x = 0.5*SL(1)*[-1 1 1 -1 -1 1 1 -1]';
        y = 0.5*SL(2)*[1 1 1 1 -1 -1 -1 -1]';
        z = 0.5*SL(3)*[-1 -1 1 1 1 1 -1 -1]';
        facs = [1 2 3 4
                5 6 7 8
                4 3 6 5
                3 2 7 6
                2 1 8 7
                1 4 5 8];
        verts = zeros(3,8);
        for j = 1:8
            verts(1:3,j) = [x(j);y(j);z(j)]+CV;
        end

 patch('Faces',facs,'Vertices',verts','FaceColor',color,'FaceAlpha',alph);
        hold on;
    end
    draw= [draw, color'];
end
xlabel('X')
ylabel('Y')
zlabel('Z')
end
%--------------------------------------------------------------------------
% The function drawObstacles is used to initiate the Obstacles
function obstacle = initObstacles()
    obstacle.num = input('Enter number of obstacles : ');
    for i=1:obstacle.num
        obstacle.type(i) = input('Enter 0 for sphere, 1 for cuboid :
 ');
        if(obstacle.type(i)==0) % Incase of sphere, length will be
 radius
                                % width, and height will be 0
            obstacle.l(i) = input('Enter radius for the sphere : ');
            obstacle.w(i)=0;
            obstacle.h(i)=0;
            obstacle.cx(i) = input('Enter the x-coordinate of center :
 ');
            obstacle.cy(i) = input('Enter the y-coordinate of center :
 ');
            obstacle.cz(i) = input('Enter the z-coordinate of center :
 ');
        elseif(obstacle.type(i)==1)
            obstacle.l(i) = input('Enter length of cuboid : ');
            obstacle.w(i) = input('Enter width of cuboid : ');
            obstacle.h(i) = input('Enter height of cuboid : ');
```

```
            obstacle.cx(i) = input('Enter the x-coordinate of center :
 ');
            obstacle.cy(i) = input('Enter the y-coordinate of center :
 ');
            obstacle.cz(i) = input('Enter the z-coordinate of center :
 ');
        end
    end
end
%-------------------------------------------------------------------------
function flag = isCollision(X1, X2, obstacle)
len = length(X1);
flag=0; % If there is a collision, then flag will be 1 or 2 depending
 on whether it's a sphere or cuboid
for i=1:obstacle.num
    if(flag~=0) % if flag is not equal to 0
        break;
    end
    for j = 1:len
        if obstacle.type(i)==0 % If the obstacle is a sphere
            dist1 = dista(X1, obstacle);
            dist2 = dista(X2, obstacle);
            if((dist1<=obstacle.l(i)) || (dist2<=obstacle.l(i)))
                flag=i;
                break;
            end
        end
        if obstacle.type(i)==1
            dist1(1) = abs(X1(1,j)-obstacle.cx(i));
            dist1(2) = abs(X1(2,j)-obstacle.cy(i));
            dist1(3) = abs(X1(3,j)-obstacle.cz(i));
            dist2(1) = abs(X2(1,j)-obstacle.cx(i));
            dist2(2) = abs(X2(2,j)-obstacle.cy(i));
            dist2(3) = abs(X2(3,j)-obstacle.cz(i));
            if(((dist1(1)<=obstacle.l(i)/2) &&
 (dist1(2)<=obstacle.w(i)/2) && (dist1(3)<=obstacle.h(i)/2)) ||
 ((dist2(2)<=obstacle.w(i)/2) &&(dist2(1)<=obstacle.l(i)/2) &&
 (dist2(3)<=obstacle.h(i)/2)))
                flag=i;
                break;
            end
        end
    end
end
end
function distance = dista(X, obstacle)
distance = sqrt((X(1,j)-obstacle.cx(i))^2+(X(2,j)-
obstacle.cy(i))^2+(X(3,j)-obstacle.cz(i))^2);
end
```

*Error using dbstatus*
*Error: File: G:\7th Sem\ME766\A2\A2.m Line: 68 Column: 1*
*Illegal use of reserved keyword "end".*

*Published with MATLAB® R2019a*

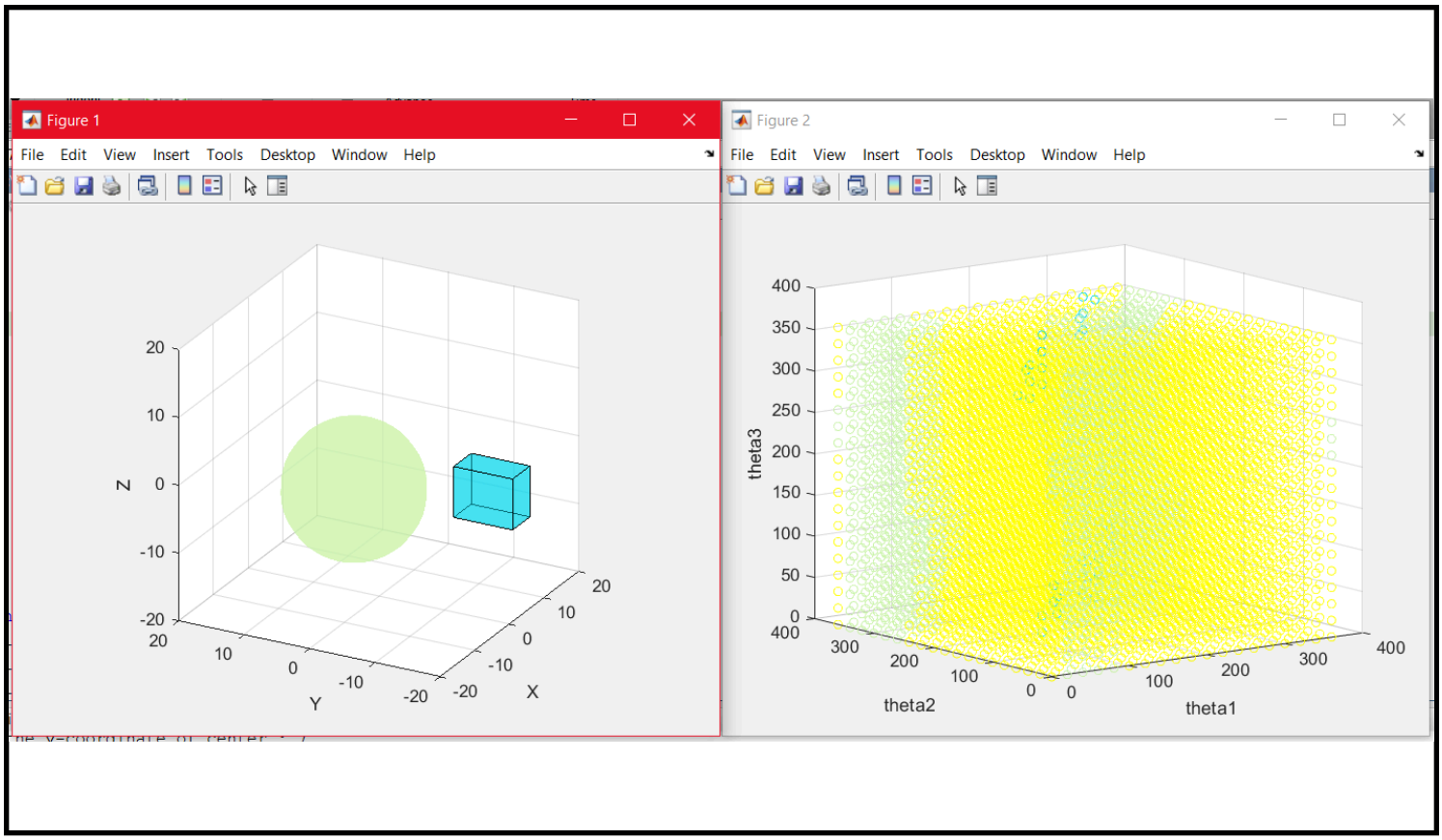


*Figure 1 - Shows workspace. Origin is at 0,0,0 and length of Link1 is 10 units and length of Link is 7.5 units*

*Figure 2 - Shows C space*