```
close all
clear all


%Assume SI Units for everything
%Run Robotics Toolbox Lab Computer
%run ~/Desktop/rvctools/startup_rvc.m

%Run Robotics Toolbox Personal Computer
%run C:\Users\Dustan\Desktop\rvctools\startup_rvc.m

%% Create Robot and Obstacle
%Original Robot
L(1) = Link('d',4,'a',0,'alpha',pi/2,'offset',pi/2);
L(2) = Link('d',0,'a',0,'alpha',pi/2,'offset',pi/2);
L(3) = Link('d',sqrt(16.25),'a',0,'alpha',pi/2);
L(4) = Link('d',0,'a',2,'alpha',pi/2,'offset',pi/2);
bot = SerialLink(L, 'name', 'Dustan');

%define position of robot joint 2
[H,var] = bot.fkine([0 0 0 0]);
x_top = var(1:3,4,1);

%Obstacle
obs_pos = [0,3,2];
obs_rad = 1;

%determine distance between joint 2 and obstacle center
x_top = var(1:3,4,1);
dist = sqrt((x_top(1) - obs_pos(1))^2 + (x_top(2) - obs_pos(2))^2 +...
    (x_top(3) - obs_pos(3))^2);

%Robot with a DoF added
L(1) = Link('d',4,'a',0,'alpha',pi/2,'offset',pi/2);
L(2) = Link('d',0,'a',0,'alpha',pi/2,'offset',pi/2);
L(3) = Link('d',dist, 'a', 0, 'alpha',-pi/2,'offset',-pi/2);
L(4) = Link('d',0,'a',sqrt(16.25)-dist,'alpha',-pi/2,'offset',-pi/2);
L(5) = Link('d',0,'a',2,'alpha',pi/2);

bot = SerialLink(L, 'name', 'Dustan');

%% Calculate Path in Joint Space
q_i = [0, 5*pi/3, 0, 0, 5*pi/3];
x_f = [0;2;4];
q = calc_q(q_i, x_f, obs_pos, obs_rad, bot);

%% Create Animation
figure(1)
bot.plot(q_i)
view(170,-10)
hold on
[x, y, z] = sphere;
x = x*obs_rad;
y = y*obs_rad;
z = z*obs_rad;
surf(x+obs_pos(1),y+obs_pos(2),z+obs_pos(3));
pause(5)
for i = 1:length(q)
```

```matlab
    bot.animate(q(i,:));
end

%% Print Final Position
H_f = bot.fkine(q(end,:));
x_f_ik = H_f(1:3,4)

%the final position obtained was [-0.0149, 2.0982, 3.9903] - this could
%have easily been more accurate simply by setting a smaller inverse
%kinematics tolerance, but this made the animation take significantly
%longer due to a higher number of q_k_pl_1 joint angles.


function q = calc_q(q_i, x_f, obs_pos, obs_rad, bot)
%calc_q Generate a joint space path while avoiding an obstacle
%   q_i = initial configuration [q1, q2, q3, q4]
%   x_f = final position goal in task space [x,y,z]'
%   obs_pos = obstacle position in task space [x,y,z]'
%   obs_rad = obstace radius in meters
%   bot = Serial link robot using robotics toolbox
H = bot.fkine(q_i);
x = H(1:3,4);
J = bot.jacob0(q_i);
Ja = J(1:3,:);
K = 0.01*eye(3);
qk = q_i;
q = [qk];
while sum((x-x_f).^2) > 1e-2,
    qdot = Ja.'*K*(x_f-x);
    q_k_pl_1 = qk + qdot.';
    [H, all] = bot.fkine(q_k_pl_1);
    %position of joint 3 which is at the same radius as the obstacle
    x_c = all(1:3,4,3);
    %distance between the obstacle center and joint 3
    int_dist = sqrt(sum((x_c' - obs_pos).^2));
    %adjust q_k_pl_1 to not hit obstacle assuming link radius of 0.75
    while int_dist < obs_rad+0.75,
        q_k_pl_1(1) = q_k_pl_1(1) + 0.001;
        [H,all] = bot.fkine(q_k_pl_1);
        x_c = all(1:3,4,3);
        int_dist = sqrt(sum((x_c' - obs_pos).^2));
    end
    x = H(1:3,4);
    J = bot.jacob0(q_k_pl_1);
    Ja = J(1:3,:);
    qk = q_k_pl_1;
    q(end+1,:) = qk;
end

end
```