```
% Name      :    Rohan Singh Rajput
% Subject   :    Final Robotics Project EEL_5669
% Submitted :    15 - Dec - 2015
%-------------------------------------------------------------------------------------
% Objective :    Simulation of Adaptive 2.5D Visual Servoing of Kinematically Redundant Robot
%                Manipulators By : Y. Fang,  A. Behal, W. E. Dixon, and D. M. Dawson
%
% Tools     :    MATLAB(2015b), Statistical Tool Box, Machine Learning Tool
%                Box,Symbolic Tool Box, Simulink.
%
% Approch Taken: The Project is devided into two parts, first part discusses about the
%                Kinmetics of Robot manipulations the resepctive output has been
%                calculated on the basis of provided paper. The Error function of
%                translation and rotation are being plotted in kinematics analysis part.
%
%                For the Dynamics part the anlysis has been done using Peter Corke Robotics Tool Bo
x with the analysis of Puma 560 Robotics
%                Manuplator.
%
%----------------------------------------------------------------------------------------------
--------

%%%%*************Code Starts Here***************************************
clear;
close all;
clc;

%************Intialization of Pixels ***********************%%%%%%
p_star = [];
p = [];

p_star(:,1) =  [7.4 13 1]';
p_star(:,2) =  [12.4 13 1]';
p_star(:,3) =  [17.2 8 1]';
p_star(:,4) =  [22.2 8 1]';

p(:,1) = [-10.34 19.5 1]';
p(:,2) = [-9.94 29.5 1]';
p(:,3) = [0.46 39.5 1]';
p(:,4) = [0.86 49.5 1]';
%%%%************Intial Error Vector*************%%%%
ev = [];
ew = [];


for iterations=1:30

    %%%%%%%************Solve of Liner Algebric Equation For G Matrix**************%%%%
    syms g1 g2 g3 g4 g5 g6 g7 g8 a1 a2 a3 a4

    eq1 = a1*(p_star(1,1)*g1+p_star(2,1)*g2+p_star(3,1)*g3)-p(1,1);
    eq2 = a1*(p_star(1,1)*g4+p_star(2,1)*g5+p_star(3,1)*g6)-p(2,1);
    eq3 = a1*(p_star(1,1)*g7+p_star(2,1)*g8+p_star(3,1))-p(3,1);
    eq4 = a2*(p_star(1,2)*g1+p_star(2,2)*g2+p_star(2,3)*g3)-p(1,2);
    eq5 = a2*(p_star(1,2)*g4+p_star(2,2)*g4+p_star(2,3)*g6)-p(2,2);
```

```
    eq6 = a2*(p_star(1,2)*g7+p_star(2,2)*g8+p_star(2,3))-p(3,2);
    eq7 = a3*(p_star(1,3)*g1+p_star(2,3)*g2+p_star(3,3)*g3)-p(1,3);
    eq8 = a3*(p_star(1,3)*g4+p_star(2,3)*g5+p_star(3,3)*g6)-p(2,3);
    eq9 = a3*(p_star(1,3)*g7+p_star(2,3)*g8+p_star(3,3))-p(3,3);
    eq10 =a4*(p_star(1,4)*g1+p_star(2,4)*g2+p_star(3,4)*g3)-p(1,4);
    eq11 =a4*(p_star(1,4)*g4+p_star(2,4)*g5+p_star(3,4)*g6)-p(2,4);
    eq12 =a4*(p_star(1,4)*g7+p_star(2,4)*g8+p_star(3,4))-p(3,4);


    [a,b,c,d,e,f,g,h,i,j,k,l] = solve(eq1,eq2,eq3,eq4,eq5,eq6,eq7,eq8,eq9,g1,g2,g3,g4,g5,g6,g7,g8,a
1,a2,a3,a4);

    G = [a,b,c;d,e,f;g,h,1];
    G = vpa(G);
    alpha1 = i ; alpha2 = j ; alpha3 = k; alpha4 = l;



    %%%%%%%*************Camera Parameter********************%%%%%%%%%%


    al1=5;
    al2=0.2;
    al3=5;
    al4=2;
    al5=3;

    A_Matrix = [al1 al2 al4; 0 al3 al5; 0 0 1]

    %%%%%%%%%% Calculation of Euclidean Homogrophy***********************

    H = double(inv(A_Matrix)*G*A_Matrix);

    %%%%%%% Function for decoposition of Euclidean Homographic Matrix *****
    function [R,t,n,d] = homog(H)

    [u,d,v] = svd(H);
    d = diag(d);
    d = sort(d,'descend');


    dpr = d(2);
    ep = [1,0,1]';

    if d(1)~=d(2)~=d(3)
        x(1) = ep(1)*sqrt((d(1)^2 - d(2)^2)/(d(1)^2 - d(3)^2));
        x(2) = 0;
        x(3) = ep(3)*sqrt((d(2)^2 - d(3)^2)/(d(1)^2 - d(3)^2));

        if dpr > 0
            sth = ep(1)*ep(3)*(sqrt((d(1)^2)-(d(2)^2))*(d(2)^2)-(d(3)^2))/((d(1)+d(3))*d(2));
            cth = (d(2)^2 + (d(1)*d(3)))/((d(1)+d(3))*d(2));

            Rpr = [cth 0 -sth;0 1 0;sth 0 cth];
            tpr = (d(1)-d(3))*[x(1) 0 -x(3)]';
            npr = x'
        end
```

```
        if dpr<0

            sth = ep(1)*ep(3)*(sqrt((d(1)^2)-(d(2)^2))*(d(2)^2)-(d(3)^2))/((d(1)-d(3))*d(2));
            cth = (-d(2)^2 +(d(1)*d(3)))/((d(1)-d(3))*d(2));

            Rpr = [cth 0 -sth;0 1 0;sth 0 cth];
            tpr = (d(1)+d(3))*[x(1) 0 x(3)]';
            npr = x;
        end
    end


    if ((d(1) == d(2)) && ((d(1) == d(2))~= d(3))) || ((d(1) ~= d(2)) && (d(1) ~= d(2)) == d(3))
        if dpr>0

            npr = [0,0,1]';
            Rpr = eye(3);
            tpr = (d(3)-d(1))*npr;
        end

        if dpr<0
            npr = [0,0,1]';
            Rpr = [-1,0,0;0,-1,0;0,0,1];
            tpr = (d(3)+d(1))*npr;
        end
    end

    if d(1) == d(2) == d(3)
        if dpr>0
            Rpr = eye(3);
            tpr = 0;
            npr = [1,1,1]'
        end

        if dpr<0
            Rpr = -eye(3) + 2*(npr*npr');
            tpr = 2*dpr*npr;%n'
            npr = [1,1,1]';
        end
    end

    s = det(u)*det(v);
    d = s*dpr;
    n = v*npr;
    t = u*tpr;
    R = s*u*Rpr*v';
    %%%%%**************End of Function**************************************
    [R,xh,n_star] = homog(H);
    n_star_trans  = n_star';
    th = acos(0.5*(trace(R)-1));
    ux   = (R - R')/(2*sin(th));
    u     = [ux(3,2);ux(1,3);ux(2,1)];

    %%%%%%%%%%% Plot for Rotational Error****************
    ew(:,1) = u*th;
    new_ew(iterations,:) = ew;
```

```
    %%%%%%%%%%%*******Calcuating the Model M for the New Pixel Calculations*********
    for i = 1:size(p,2)
        m(:,i) = inv(A_Matrix)*p(:,i);
        me(1,i) = m(1,i);
        me(2,i) = m(2,i);
    end
    %%%%%%%%*********Calculation for M Star matrix*************************
    for k = 1:size(p_star,2)
        m_star(:,k) = inv(A_Matrix)*p_star(:,k);
        me_star(1,k) = m_star(1,k);
        me_star(2,k) = m_star(2,k);
    end
    n_trans = n_star_trans*R;
    Z_by_Z_Star= log(((1+(n_trans)*xh))*((n_star_trans)*m_star(:,1)))/((n_trans)*m(:,1));

    %%%%%%%%%%%%% Claculation of Translation Errors*************************
    ev(:,1) = [(me(1,1)-me_star(1,1)),(me(2,1)- me_star(2,1)),Z_by_Z_Star]';
    L_v = [-1,0,me(1,1);0,-1,me(1,2);0,0,1];
    L_v_w=[me(1,1)*me(2,1),-1-me(1,1).^2,me(2,1);
        1+me(2,1).^2,-(me(1,1)*me(2,1)),-me(1,1);
        -me(2,1),me(1,1),0];
    %%%%%%%%%% Provided Paramentes of Cmaera***************

    ko = 0.1;
      Tw = diag([0.5,0.5,0.5]);
    wc = -Tw*ew;
    Tv = diag([0.5,0.5,0.5]);

    %%%%%%%%%%% Derivative of d_star_cap******************%%%%%%%%%%%%%%%%
    syms d_star_cap_der(t)
    d_star_cap_der(t) = dsolve(diff(d_star_cap_der,t) == ko*((ev)')*L_v_w*(wc), d_star_cap_der(0) =
= -0.125);
    d_star_cap =vpa( d_star_cap_der(iterations));

    d_star=n_star_trans*m_star(:,1);
    gamma2=1/d_star;

    %%%%%%%%%**********Calculate vc*********************************
    vc=-(gamma2*inv(L_v))*((Tv*ev)+(d_star_cap*(L_v_w*wc)));
    %%%%%%%%%%%*********Updating Pixels again*******************
    for i=1:4
        m_expand(:,:,i)=[0,-m(3,i),m(2,i);m(3,i),0,-m(1,i);-m(2,i),m(1,i),0];
        m1_dot_bar(:,:,i)=-vc+m_expand(:,:,i)*wc;
        new_p(:,:,i)=A_Matrix*m1_dot_bar(:,:,i);
        p(:,i)=real(new_p(:,:,i));




    end




    end
%%%%%%%%%%%%%**********Dynamics Part****************%%%%%%%%%
```

```
q=[1.5708 -0.7854 3.1416 0 0.7854 0.4363]';
%%%%%%*************Using Peter Corke Tool BOx for Puma560**************
J = p560.jacob0(q);
K = diag{[200,200,200,200,200,200]}
qd = [vc(:,trail) ; wd(:,trail)]
pseudo_inv_j_dot=pinv(J)*qd;

%%%%%%%%%*************Calculating Coriolis Matrix************
C = p560.coriolis(q,qd)
M = p560.inertia(q);
G = p560.gravload(q);

%%%%%%%%%%%%*******Calculating Torque Value Got from Peter Corke********
yphi=M*(pseudo_inv_j_dot*vfd)+G+F;
vfd=(1/d_star)*ohma+ohmb;
ro=pseudo_inv_j_dot*vfd-q_dot;
torque= yphi + K*ro;
q_double_dot = inv(M)*(torque - C - G -F)

%%%%%%%%%%%%******Calculating q double integration*******************
sym q_double_dot(trial)

q_double_dot(trial) =  dsolve(diff(q_double_dot,t2) == inv(M)*(torque - C - G -F));

q = (q_double_dot(trail))';

end

%%%%%%%%%******************Plotting Results for all********************

%%%%%%********************Start ploting of Kinematics***************

%%%%%%%%%%%%%***********Plot for Translational Error**************
figure(1)
%%%%%%*******Error along X**************
subplot(3,1,1);
plot(t,ev(1,:)); ylim([-4 2])
title('Translational Errors(ev)')
xlabel('Steps');
ylabel('Error along X')

%%%%%%*******Error along Y**************
subplot(3,1,2);
plot(t,ev(2,:)); ylim([-.5 1.5]);
xlabel('Steps');
ylabel('Error along Y')

%%%%%%*******Error along Z**************

subplot(3,1,3);
plot(t,ev(3,:)); ylim([-1 0.5])
xlabel('Steps');
ylabel('Error along Z')

%%%%%%%%*******Plot for rotational Error**************
figure(2)
```

```
%%%%%*******Error along X**************

subplot(3,1,1);
plot(t,ew(1,:)); ylim([-0.05 0.15])
title('Rotational Errors(ew)')
xlabel('Steps');
ylabel('Error along X')

%%%%%*******Error along Y**************
subplot(3,1,2);
plot(t,ew(2,:)); ylim([-0.1 0.15]);
xlabel('Steps');
ylabel('Error along Y')

%%%%%*******Error along Z**************

subplot(3,1,3);
plot(t,ew(3,:)); ylim([-2 2])
xlabel('Steps');
ylabel('Error along Z')
%%%%%%%%%%%% Plot for d star cap derivative function***************
figure(3)
plot(t,deriv_d); ylim([-.25 0])
title('Derivative of d star cap')
xlabel('Steps');
ylabel('Values')

%%%%%%%%%%%%End of Kinematics Results**************************

%%%%%%%%************Start Plotting Dynamics******************

%%%%%%%**************Ploting Joint Torques of Robots*********

figure(4)

%%%%*********Joint 1********
subplot(3,2,1);
plot(t,torque(1,:)); ylim([-100 50])
xlabel('Steps');
ylabel('Joint Torque 1 Nm')
title('Control Torque Input')

%%%%%******** Joint 2 ************
subplot(3,2,2);
plot(t,torque(2,:)); ylim([-100 200]);
xlabel('Steps');
ylabel('Joint Torque 2 Nm')
title('Control Torque Input')

%%%%%******** Joint 3 ************

subplot(3,2,3);
plot(t,torque(3,:)); ylim([-60 20])
xlabel('Steps');
ylabel('Joint Torque 3 Nm')
```

```
%%%%%******** Joint 4 ************
subplot(3,2,4);
plot(t,torque(4,:)); ylim([-100 50])
xlabel('Steps');
ylabel('Joint Torque 4 Nm')

%%%%%******** Joint 5 ************

subplot(3,2,5);
plot(t,torque(5,:)); ylim([-100 200]);
xlabel('Steps');
ylabel('Joint Torque 5 Nm')

%%%%%******** Joint 6 ************

subplot(3,2,6);
plot(t,torque(3,:)); ylim([-60 20])
xlabel('Steps');
ylabel('Joint Torque 6 Nm')

%%%%%%%%***********End of plotting Dynamics********************

%%%%%%%%************End of Program******************************

%%%%%%%%%%%**********Plots are given below***********************
```

Published with MATLAB® R2015b