```
%==========================
%  Name:                hw4_4.m
%
%  Author:              Kairi Kozuma
%
%==========================

% Transformation matrix for left and right cameras
R_WL = [0.913545457642601,-0.063627629171822,0.401729040058774;0.287606238475951,0.799453749866612,-0.527405302792764;-0.287606238475951,0.597348496800
T_WL = [-8.659258262890683;2.169872981077807;4.830127018922193];
R_WR = [0.994521895368273,-0.016351854232753,0.103241544429788;0.073912785203567,0.808411029059454,-0.583959337863936;-0.073912785203567,0.588391217607
T_WR = [10.659258262890683;5.830127018922193;1.169872981077807];

PsiR = [400,0,320;0,400,240;0,0,1];
PsiL = PsiR;

% a) Find in world coordinates

% Rays
pa_rLeft = [302,492;236,193];
pa_rRight = [279,372;235,194];

% Use function
[p1wL, p1wR, alpha1L, alpha1R] = findWorldPoint(R_WL, T_WL, PsiL, R_WR, T_WR, PsiR, pa_rLeft, pa_rRight);
disp('a)');
fprintf('World points obtained from left camera:\n');
disp(p1wL);
fprintf('With the following alpha values:\n');
disp(alpha1L);

fprintf('World points obtained from right camera:\n');
disp(p1wR);
fprintf('With the following alpha values:\n');
disp(alpha1R);

% b) Find depth along optical axis for each camera

% Transform to left and right camera frames
G_LW = [R_WL',-R_WL' * T_WL;0,0,0,1];
G_RW = [R_WR',-R_WR' * T_WR;0,0,0,1];

% Homogenous form
p1L = G_LW * [p1wL;ones(1,2)];
p1R = G_RW * [p1wR;ones(1,2)];

disp('b)');
disp('Depths for points with respect to left camera:');
disp(p1L(3,:));

disp('Depths for points with respect to right camera:');
disp(p1R(3,:));

% c) Find points relative to left camera frame
pc_rRight = [293,382;203,250];
pc_rLeft = [206,299;204,249];

% Homogenous form
G_WR = [R_WR,T_WR;0,0,0,1];
G_LR = G_LW * G_WR;
R_LR = G_LR(1:3,1:3);
T_LR = G_LR(1:3,4);
[p3wL, p3wR, alpha3L, alpha3R] = findWorldPoint(eye(3,3), zeros(3,1), PsiL, R_LR, T_LR, PsiR, pc_rLeft, pc_rRight);

disp('c)');
disp('Point in left camera frame');
disp(p3wL);

%============================ findWorldPoint ==============================
%
%  function [pwL, pwR, alphaL, alphaR] = findWorldPoint(R_WL, T_WL, PsiL, R_WR, T_WR, PsiR, raysL, raysR)
%
%
%  INPUT:
%       R_WL = rotation matrix for left camera
%       T_WL = translation vector for left camera
%       PsiL = camera matrix for left camera
%       R_WR = rotation matrix for right camera
%       T_WR = translation vector for right camera
%       PsiR = camera matrix for right camera
%       raysL = rays in left camera
%       raysR = rays in right camera
%
%  OUTPUT:
%       pwL = points in world obtained from left camera
%       pwR = points in world obtained from right camera
%       alphaL = alpha values for pwL
%       alphaR = alpha values for pwR
%
%============================ findWorldPoint ==============================
function [pwL, pwR, alphaL, alphaR] = findWorldPoint(R_WL, T_WL, PsiL, R_WR, T_WR, PsiR, raysL, raysR)
```

```
% Homogenous Form
raysL = [raysL;ones(1,length(raysL))];
raysR = [raysR;ones(1,length(raysR))];

n = size(raysL,2);

% b vector
b = T_WR - T_WL;

alphaL = zeros(1, n);
alphaR = zeros(1, n);
pwL = zeros(3,n);
pwR = zeros(3,n);

% Iterate through points
for i=1:n

    % Get ray from both cameras
    rayL = raysL(:,i);
    rayR = raysR(:,i);

    % A matrix
    A = [R_WL * (PsiL \ rayL), - R_WR * (PsiR \ rayR)];

    % Calculate alpha
    alpha = (A \ b);
    aL = alpha(1);
    aR = alpha(2);

    % Store alpha values
    alphaL(i) = aL;
    alphaR(i) = aR;

    b2 = A * alpha;

    pwL(:,i) = aL * R_WL * (PsiL \ rayL) + T_WL;
    pwR(:,i) = aR * R_WR * (PsiR \ rayR) + T_WR;

end

end
```

```
a)
World points obtained from left camera:
   10.7439   17.7004
  -27.2817  -14.1866
   45.4139   23.0636

With the following alpha values:
   53.7103   32.8662

World points obtained from right camera:
   10.7421   17.7009
  -27.2421  -14.2160
   45.4435   23.0364

With the following alpha values:
   54.9702   30.0399

b)
Depths for points with respect to left camera:
   53.7103   32.8662

Depths for points with respect to right camera:
   54.9702   30.0399

c)
Point in left camera frame
  -51.2820  -10.2043
  -16.1943    4.3733
  179.9368  194.3672
```

Published with MATLAB® R2016b