```
%We follow the instructions in section 2.4.4 of the Camera Calibration
%chapter by Zhengyou Zhang

%Since we have 4 computed homographies, we can stack n equations to
 get Vb=0
V = [];
for i = 1:4
    %Assigns the i(th) Homography to the variable H
    H = eval(['H' num2str(i)]);

    %Let the i-th coloumn vector of H(homography) be h(i) =
    %[h(i1),h(i2),h(i3)]'
    h1 = H(:,1);
    h2 = H(:,2);
    h3 = H(:,3);

    %Using v(ij) = [hi(1)*hj(1), hi(1)*hj(2)+hi(2)*hj(1), hi(2)*hj(2),
 hi(3)*hj(1)+hi(1)*hj(3), hi(3)*hj(2)+hi(2)*hj(3),
    %hi(3)*hj(3)]'
    v11 = [h1(1)*h1(1), h1(1)*h1(2)+h1(2)*h1(1), h1(2)*h1(2),
 h1(3)*h1(1)+h1(1)*h1(3), h1(3)*h1(2)+h1(2)*h1(3), h1(3)*h1(3)]';
    v12 = [h1(1)*h2(1), h1(1)*h2(2)+h1(2)*h2(1), h1(2)*h2(2),
 h1(3)*h2(1)+h1(1)*h2(3), h1(3)*h2(2)+h1(2)*h2(3), h1(3)*h2(3)]';
    v22 = [h2(1)*h2(1), h2(1)*h2(2)+h2(2)*h2(1), h2(2)*h2(2),
 h2(3)*h2(1)+h2(1)*h2(3), h2(3)*h2(2)+h2(2)*h2(3), h2(3)*h2(3)]';

    %We stack the 'n' equations for 'n' images to obtain V
    V = [V; v12'; (v11-v22)'];
end

%We now need to solve for Vb = 0
%The solution for Vb = 0 from the chapter is said to be well known as
 the
%eigenvector of V'.V associated with the smallest
 eigenvalue(equivalently,
%the right singular vector of V associated with the smallest singular
%value).
%We acheive this by by applying the singular value decomposition on V.
[U, Sigma, V_transpose] = svd(V);

b = V_transpose(:,end);

%Once 'b' is estimated, we can now compute all the camera instrinsic
%parameters by first describing the matrix "B".

%We know that B is symmetric and is defined by a 6D vector as follows,
% b = [B11, B12, B22, B13, B23, B33]'
% Where,
% B = [ B11,    B12,    B13]
%     [ B12,    B22,    B23]
%     [ B13,    B23,    B33]
B11 = b(1);
```

```
B12 = b(2);
B22 = b(3);
B13 = b(4);
B23 = b(5);
B33 = b(6);

B = [B11, B12, B13; B12, B22, B23; B13, B23, B33];

%Display the computed matrix B
disp("Matrix B >>");
disp(B);

%We can now calculate the intrinsic parameters using the equations
%mentioned in Page 21 of Camera Calibration chapter.
v0 = (B12*B13 - B11*B23)/(B11*B22 - B12^2);
lambda = B33 - (B13^2 + v0*(B12*B13-B11*B23))/B11;
alpha = sqrt(lambda/B11);
beta = sqrt(lambda*B11/(B11*B22-B12^2));
gamma = -B12*alpha^2*beta/lambda;
u0 = gamma*v0/alpha - B13*alpha^2/lambda;

%Therefore our intrinsic matrix A can be defined as follows,
A = [alpha, gamma, u0; 0, beta, v0; 0, 0, 1];

%Display the computed intrinsic parameters matrix K
disp("Intrinsic Parameters matrix >>");
disp(A);

%Now once we have computed the matrix B and the intrinsic parameters
 of the
%camera.We can now compute the R and t values for each image.
%We follow the steps mentioned in page 21.

files = ["images2", "images9", "images12", "images20"];

for i = 1:4
    %Assigns the i(th) Homography to the variable H
    H = eval(['H' num2str(i)]);

    %Let the i-th coloumn vector of H(homography) be h(i) =
    %[h(i1),h(i2),h(i3)]'
    h1 = H(:,1);
    h2 = H(:,2);
    h3 = H(:,3);

    %INV(A)*b can be slower and lass accurate than A\b. Consider using
 A\b for
    %INV(A)*b or b\A for b*INV(A).
    lambda_r = 1/ norm(A\h1);
    r1 = lambda_r*(A\h1);
    r2 = lambda_r*(A\h2);
    r3 = cross(r1,r2);
    t  = lambda_r*(A\h3);
```

```
    R = [r1, r2, r3];

    disp(["Rotation Matrix for >> " files(i)]);
    disp(R);
    disp(["Translation vector for >> " files(i)]);
    disp(t);

    %Now we need to confirm that our rotation matrix is in fact a
 rotation matrix.
    %We print R'*R to check if it is an identity matrix.
    disp(["Transpose(R)*R for >> " files(i)]);
    disp(R'*R);

    %We notice that it is not an identity matrix, hence we can enforce
 R to be
    %a rotation matrix by SVD decomposition of R by setting the
 singular values to ones
    % U*Sigma*V_T = svd(R), new rotation matrix will be U*V_T
    [U, Sigma, V_transpose] = svd(R);
    R_new = U*V_transpose;

    disp(["New rotation matrix for >> " files(i)]);
    disp(R_new);
    disp(["Transpose(R_new)*R_new for >> " files(i)]);
    disp(R_new'*R_new);

end

Matrix B >>
   -0.0000    0.0000    0.0005
    0.0000   -0.0000    0.0004
    0.0005    0.0004   -1.0000

Intrinsic Parameters matrix >>
  728.2247    2.4379  325.4991
         0  711.3290  234.1077
         0         0    1.0000

    "Rotation Matrix for >> "    "images2"

   -0.9998   -0.0125    0.0126
   -0.0180    0.9874   -0.1640
   -0.0097   -0.1641   -0.9874

    "Translation vector for >> "    "images2"

  148.8465
 -105.2400
 -419.5551

    "Transpose(R)*R for >> "    "images2"

    1.0000   -0.0037   -0.0000
   -0.0037    1.0020    0.0000
```

```
  -0.0000    0.0000    1.0020

   "New rotation matrix for >> "    "images2"

  -0.3516   -0.8032    0.4809
  -0.6120    0.5860    0.5312
  -0.7084   -0.1075   -0.6975

   "Transpose(R_new)*R_new for >> "    "images2"

   1.0000   -0.0000   -0.0000
  -0.0000    1.0000   -0.0000
  -0.0000   -0.0000    1.0000

   "Rotation Matrix for >> "    "images9"

   0.9273   -0.0065    0.3751
   0.0299   -0.9979   -0.0864
   0.3732    0.0906   -0.9251

   "Translation vector for >> "    "images9"

 -96.0230
  95.0853
 361.8431

   "Transpose(R)*R for >> "    "images9"

   1.0000   -0.0020    0.0000
  -0.0020    1.0040    0.0000
   0.0000    0.0000    1.0040

   "New rotation matrix for >> "    "images9"

  -0.2035   -0.8525   -0.4816
   0.4400    0.3598   -0.8228
   0.8747   -0.3793    0.3018

   "Transpose(R_new)*R_new for >> "    "images9"

   1.0000   -0.0000    0.0000
  -0.0000    1.0000    0.0000
   0.0000    0.0000    1.0000

   "Rotation Matrix for >> "    "images12"

   0.9199   -0.0064   -0.3910
  -0.0503   -0.9890   -0.1118
  -0.3890    0.1243   -0.9101

   "Translation vector for >> "    "images12"

 -145.1721
  105.4850
```

```
  477.3528

    "Transpose(R)*R for >> "    "images12"

    1.0000   -0.0045    0.0000
   -0.0045    0.9936    0.0000
    0.0000    0.0000    0.9936

    "New rotation matrix for >> "    "images12"

    0.9081   -0.4178   -0.0271
    0.4142    0.9060   -0.0874
    0.0610    0.0682    0.9958

    "Transpose(R_new)*R_new for >> "    "images12"

    1.0000    0.0000    0.0000
    0.0000    1.0000   -0.0000
    0.0000   -0.0000    1.0000

    "Rotation Matrix for >> "    "images20"

    0.9998   -0.0000    0.0027
   -0.0120   -0.7081   -0.7052
    0.0157    0.7053   -0.7080

    "Translation vector for >> "    "images20"

  -117.0827
    24.9967
   432.8085

    "Transpose(R)*R for >> "    "images20"

    1.0000    0.0195    0.0000
    0.0195    0.9990    0.0000
    0.0000    0.0000    0.9986

    "New rotation matrix for >> "    "images20"

    0.5100    0.7048    0.4931
   -0.8492    0.5040    0.1580
   -0.1372   -0.4993    0.8555

    "Transpose(R_new)*R_new for >> "    "images20"

    1.0000    0.0000   -0.0000
    0.0000    1.0000    0.0000
   -0.0000    0.0000    1.0000
```

Published with MATLAB® R2019a