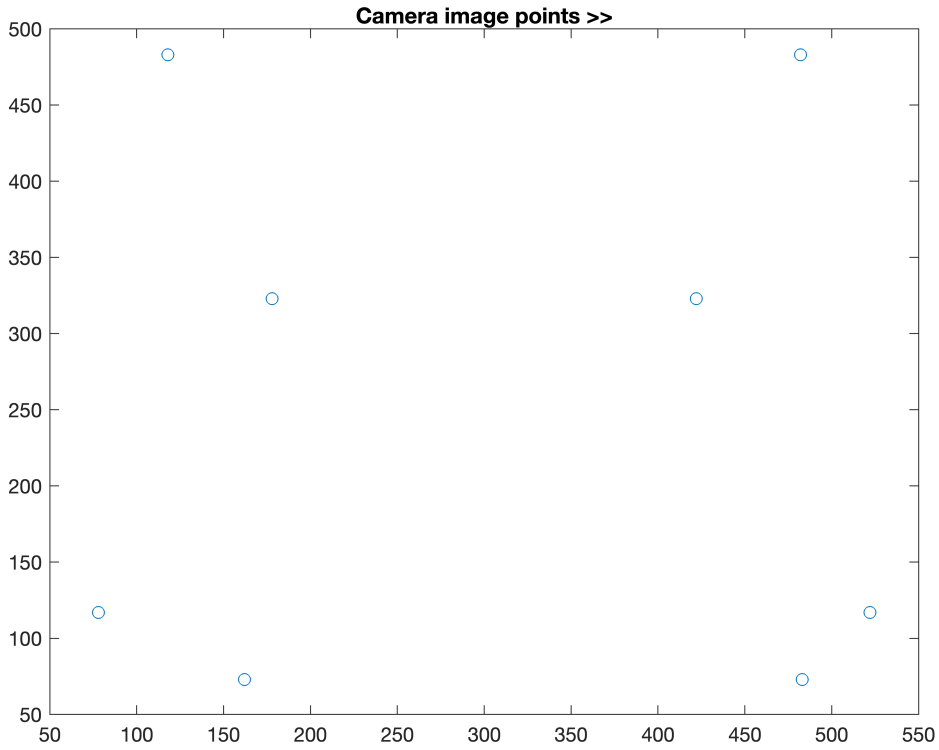
PART I: Camera Calibration using 3D calibration object

Q1. Draw the image points, using small circles for each image point

```
world_coordinates = [2 2 2; -2 2 2;-2 2 -2;2 2 -2;2 -2 2; -2 -2 2; -2 -2 -2; 2 -2 -2];
image_coordinates = [422 323; 178 323; 118 483; 482 483;483 73;162 73; 78 117;522 117]

plot(image_coordinates(:,1),image_coordinates(:,2),'o');
title("Camera image points >>")
```




```
p_matrix = [];
```

Q2 is at the bottom

Q3. Use this function to generate 2 rows of the matrix P for each cube corner and its image and obtain a matrix with 16 rows and 12 columns. Print matrix P.

```
for i = 1:8
    p_cap = transpose([world_coordinates(i,:) 1]);
    p_small = transpose(image_coordinates(i,:));
    rows = Prows(p_small, p_cap);
    p_matrix = [p_matrix; rows];
end
disp("P")
```

```
P
```

```
disp(p_matrix)
```

```
Columns 1 through 8

     2     2     2     1     0     0     0     0
     0     0     0     0     2     2     2     1
    -2     2     2     1     0     0     0     0
     0     0     0     0    -2     2     2     1
    -2     2    -2     1     0     0     0     0
     0     0     0     0    -2     2    -2     1
     2     2    -2     1     0     0     0     0
     0     0     0     0     2     2    -2     1
     2    -2     2     1     0     0     0     0
     0     0     0     0     2    -2     2     1
    -2    -2     2     1     0     0     0     0
     0     0     0     0    -2    -2     2     1
    -2    -2    -2     1     0     0     0     0
     0     0     0     0    -2    -2    -2     1
     2    -2    -2     1     0     0     0     0
     0     0     0     0     2    -2    -2     1

Columns 9 through 12

  -844  -844  -844  -422
  -646  -646  -646  -323
   356  -356  -356  -178
   646  -646  -646  -323
   236  -236   236  -118
   966  -966   966  -483
  -964  -964   964  -482
  -966  -966   966  -483
  -966   966  -966  -483
  -146   146  -146   -73
   324   324  -324  -162
   146   146  -146   -73
   156   156   156   -78
   234   234   234  -117
 -1044  1044  1044  -522
  -234   234   234  -117
```

Q4. Now we need to solve the system Pm =0. Find the singular value decomposition of matrix P using matlab svd function. The last column vector of V obtained by svd(P) should be the 12 elements in row order of the projection matrix that transformed the cube corner coordinates into their images. Print the matrix M

```
[U,sigma,V_transpose] = svd(p_matrix);

M(1,1:4) = V_transpose(1:4,end)';
M(2,1:4) = V_transpose(5:8,end)';
M(3,1:4) = V_transpose(9:12,end)';

disp("Matrix M");
```

```
Matrix M
```

```
disp(M);
```

```
    0.1959    0.0293    0.0771    0.7337
   -0.0002    0.2082   -0.0068    0.6109
   -0.0000    0.0001    0.0002    0.0024
```

Q5. Now we need to recover the translation vector which is a null vector of M. Find the singular value decomposition of matrix M=U ^V T . The 4 elements of the last column of V are the homogeneous coordinates of the position of the camera center of projection in the frame of reference of the cube (as in slide 36). Print the corresponding 3 Euclidean coordinates of the camera center in the frame of reference of the cube.

```
[u, s, v] = svd(M);
camera_center = v(:,end);
camera_center = camera_center/camera_center(end);
disp("Translation vector/camera center");
```

```
Translation vector/camera center
```

```
disp(camera_center(1:3));
```

```
    0.1293
   -3.2170
   -8.6275
```

Q6. Consider the 3x3 matrix M' composed of the first 3 columns of matrix M. Rescale the elements of this matrix so that its element m33 becomes equal to 1. Print matrix M' . Now let the rotation matrices be as defined in slide 38 where the axes e1, e2, e3 are the x, y, z axes respectively

```
m_dash = M(:,1:3);
m_dash = m_dash/m_dash(3,3);
disp("M");
```

```
M
```

```
disp(m_dash);
```

```
  843.7086  126.0766  331.8648
   -0.9935  896.5559  -29.3752
   -0.0040    0.5256    1.0000
```

Q7. We will perform the RQ factorization of M' in several steps. Note that the term at position (3, 2) would also be set to zero if the signs of cos(✓x)and sin(✓x)were reversed, but this would lead to finding a negative focal length for the camera. So we should choose the signs that leads to a positive focal length. Compute the angle ✓x of this rotation in degrees. Compute matrix N=M' ⇤Rx. Print Rx, ✓x and N.

```
cos=m_dash(3,3)/sqrt((m_dash(3,3)^2)+m_dash(3,2)^2);
sin=-m_dash(3,2)/sqrt((m_dash(3,3)^2)+m_dash(3,2)^2);
Rx=[1 0 0;0 cos -sin; 0 sin cos];
disp("Rx")
```

```
Rx
```

```
disp(Rx)
```

```
    1.0000         0         0
         0    0.8852    0.4652
         0   -0.4652    0.8852
```

```
theta=rad2deg(atan(sin/cos));
fprintf("ThetaX: %f \n", theta);
```

```
ThetaX: -27.724678
```

```
N=m_dash*Rx;
disp("N");
```

```
N
```

```
disp(N);
```

```
  843.7086  -42.7891  352.4183
   -0.9935  807.2913  391.0960
   -0.0040         0    1.1297
```

Q8. The element n31 of Nis small enough so that there is no need for a rotation Ry. However, element n21 is large and a rotation matrix Rz is needed to set it to zero. Compute the rotation matrix Rz using cosine and sine. Compute the rotation angle ✓z in degrees.

```
cosz=m_dash(2,2)/sqrt((m_dash(2,2)^2)+m_dash(2,1)^2);
sinz=-m_dash(2,1)/sqrt((m_dash(2,2)^2)+m_dash(2,1)^2);
theta_z=rad2deg(atan(sinz/cosz));
fprintf("theta_z: %f \n", theta_z);
```

```
theta_z: 0.063488
```

```
Rz=[cosz -sinz 0;sinz cosz 0; 0 0 1];
```

Q9. Since we factorized out Rz we can directly compute the calibration matrix K, how? Compute K and rescale so that its element K33 is set to 1. Print K. What are the focal lengths of the camera in pixels? What are the pixel coordinates of the image center of the camera?

```
RxRz=transpose(Rx)*transpose(Rz);
K=m_dash*inv(RxRz);
K=K/K(3,3);
disp("K");
```

```
K
```

```
disp(K);
```

```
  746.9672  -38.6092  311.5728
         0  714.6081  346.1952
   -0.0030    0.0000    1.0000
```

```
disp("----------------------------Intrinsic Parameters------------------------- \n")
```

```
----------------------------Intrinsic Parameters------------------------- \n
```

```
fprintf("Focal lengths--------------- \n");
```

```
Focal lengths---------------
```

```
fprintf("alpha: %f, beta: %f, gamma: %f \n", K(1,1),K(2,1),K(1,2));
```

```
alpha: 746.967211, beta: 0.000000, gamma: -38.609164
```

```
fprintf("Image centers--------------- \n");
```

```
Image centers---------------
```

```
fprintf("u0: %f, v0: %f \n", K(1,3), K(2,3))
```

```
u0: 311.572797, v0: 346.195211
```

Q2. Write a function that takes as argument the homogeneous coordinates of one cube corner and the homogeneous coordinates of its image, and returns 2 rows of the matrix

```
function y = Prows(uv,xyz1)
xyz1Transpose = transpose(xyz1);
uXxyz1Transpose = -uv(1)*xyz1Transpose;
vXxyz1Transpose = -uv(2)*xyz1Transpose;
Zeros = [0 0 0 0];
y = [xyz1Transpose Zeros uXxyz1Transpose;Zeros xyz1Transpose vXxyz1Transpose];
end
```