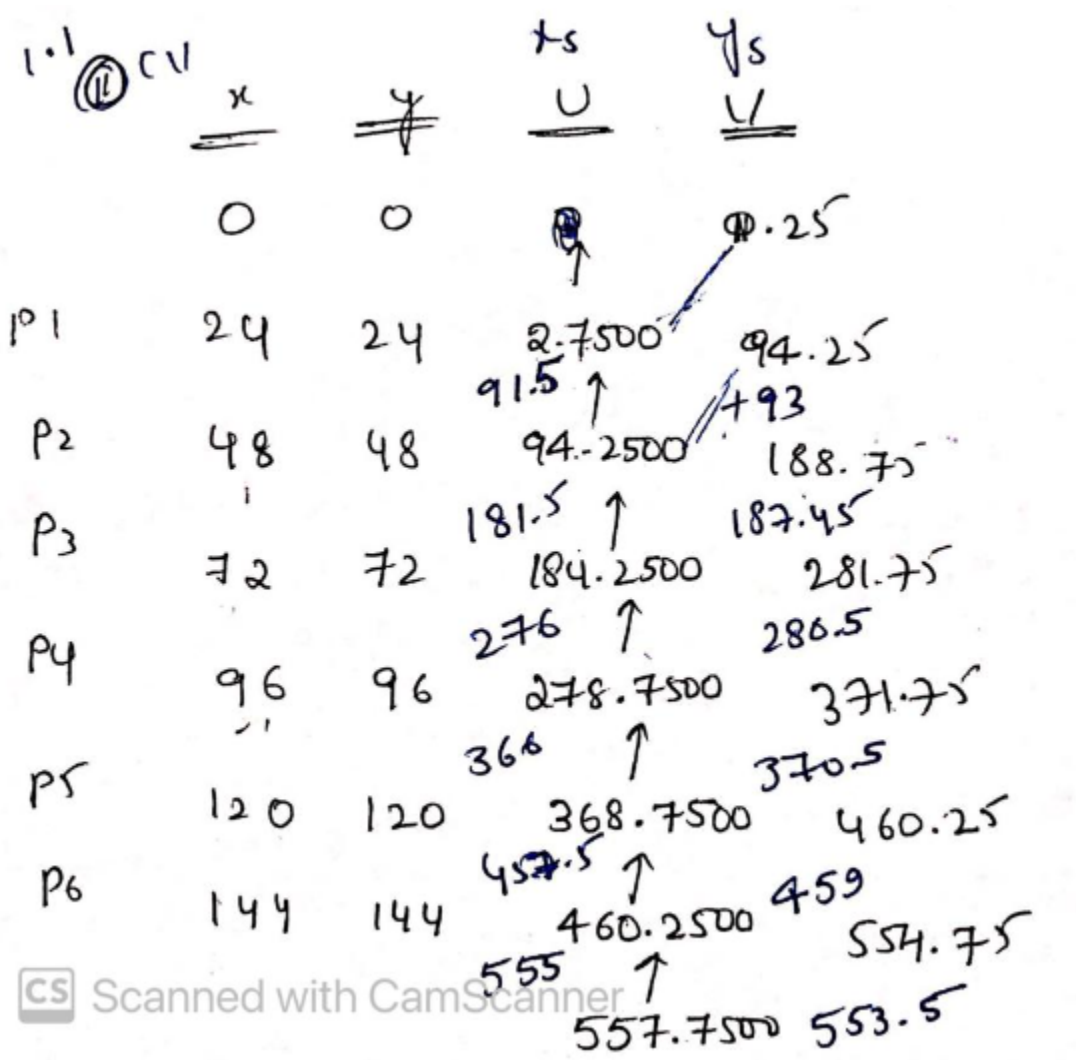
Assignment 1:

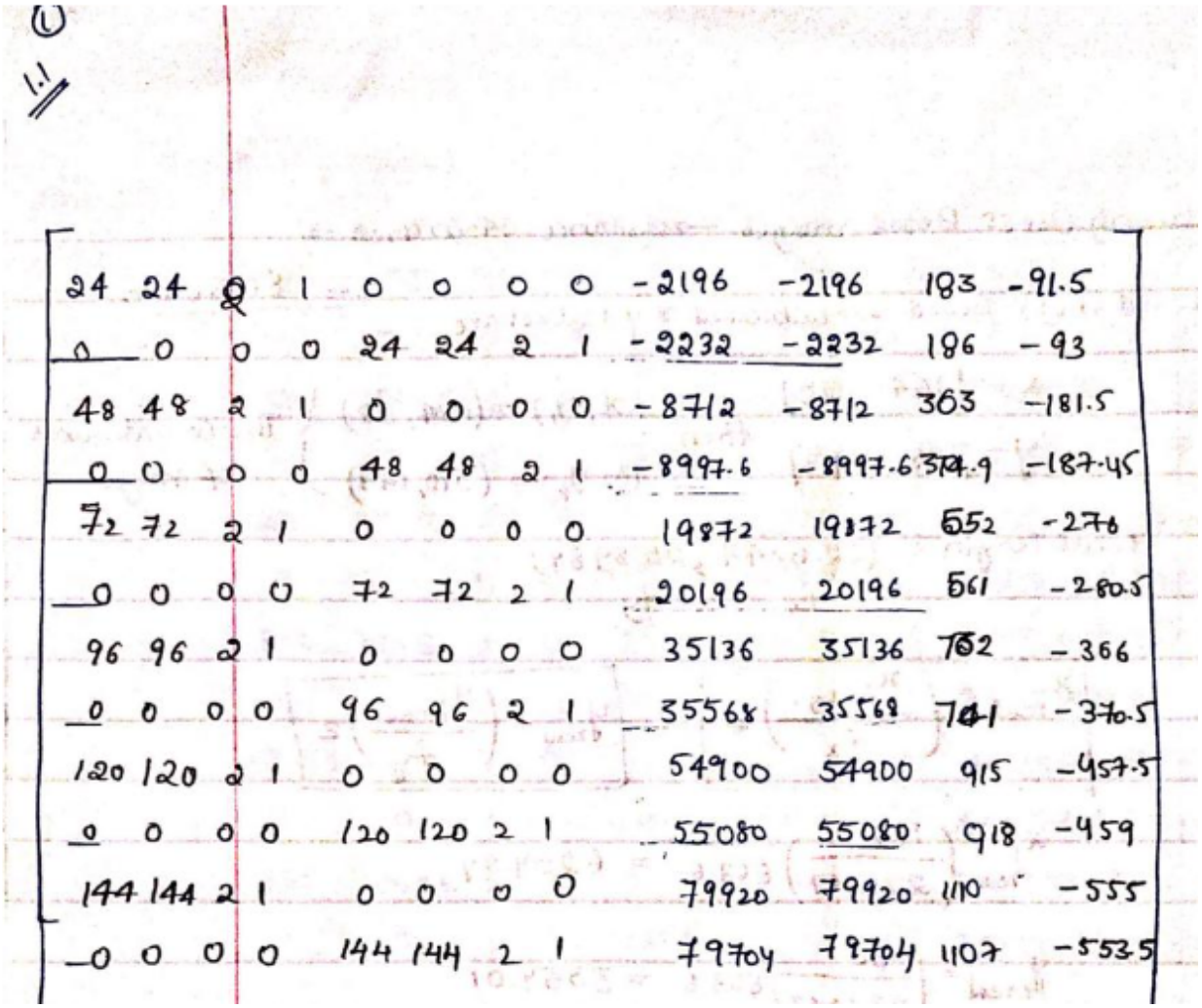
PART A: Theory

1. Point the camera to a chessboard pattern or any known set of reference points that lie on the same plane. Capture a series of 10 images by changing the orientation of the camera in each iteration. Select any 1 image, and using the image formation pipeline equation, set up the linear equations in matrix form and solve for intrinsic and extrinsic parameters (extrinsic for that particular orientation). You will need to make measurements of the actual 3D world points, and mark pixel coordinates.

I have noted the pixel values from the matlab using the ginput() function.



By using the reference equations we convert into array



24	24	2	1	0	0	0	0	-2196	-2196	183	-91.5
0	0	0	0	24	24	2	1	-2232	-2232	186	-93
48	48	2	1	0	0	0	0	-8712	-8712	363	-181.5
0	0	0	0	48	48	2	1	-8997.6	-8997.6	374.9	-187.45
72	72	2	1	0	0	0	0	19872	19872	552	-276
0	0	0	0	72	72	2	1	20196	20196	561	-280.5
96	96	2	1	0	0	0	0	35136	35136	732	-366
0	0	0	0	96	96	2	1	35568	35568	741	-370.5
120	120	2	1	0	0	0	0	54900	54900	915	-457.5
0	0	0	0	120	120	2	1	55080	55080	918	-459
144	144	2	1	0	0	0	0	79920	79920	1110	-555
0	0	0	0	144	144	2	1	79704	79704	1107	-553.5

Here we are multiplying the transpose of matrix and matrix

In [26]:
```
matmul=np.dot(arr.T,arr)
matmul
```

Out[26]:
```
array([[ 5.24160000e+04,  5.24160000e+04,  1.00800000e+03,
         5.04000000e+02,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00, -2.33712000e+07,
        -2.33712000e+07,  4.01472000e+05, -2.00736000e+05],
       [ 5.24160000e+04,  5.24160000e+04,  1.00800000e+03,
         5.04000000e+02,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00, -2.33712000e+07,
        -2.33712000e+07,  4.01472000e+05, -2.00736000e+05],
       [ 1.00800000e+03,  1.00800000e+03,  2.40000000e+01,
         1.20000000e+01,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00, -4.01472000e+05,
        -4.01472000e+05,  7.71000000e+03, -3.85500000e+03],
       [ 5.04000000e+02,  5.04000000e+02,  1.20000000e+01,
         6.00000000e+00,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00, -2.00736000e+05,
        -2.00736000e+05,  3.85500000e+03, -1.92750000e+03],
       [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00,  5.24160000e+04,  5.24160000e+04,
         1.00800000e+03,  5.04000000e+02, -2.34410880e+07,
        -2.34410880e+07,  4.03555200e+05, -2.01777600e+05],
       [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00,  5.24160000e+04,  5.24160000e+04,
         1.00800000e+03,  5.04000000e+02, -2.34410880e+07,
        -2.34410880e+07,  4.03555200e+05, -2.01777600e+05],
       [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00,  1.00800000e+03,  1.00800000e+03,
         2.40000000e+01,  1.20000000e+01, -4.03556000e+05,
        -4.03556000e+05,  7.77580000e+03, -3.88790000e+03],
       [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00,  5.04000000e+02,  5.04000000e+02,
         1.20000000e+01,  6.00000000e+00, -2.01778000e+05,
        -2.01778000e+05,  3.88790000e+03, -1.94395000e+03],
       [-2.33712000e+07, -2.33712000e+07, -4.01472000e+05,
        -2.00736000e+05, -2.34410880e+07, -2.34410880e+07,
        -4.03556000e+05, -2.01778000e+05,  2.22568135e+10,
         2.22568135e+10, -3.59468034e+08,  1.79734017e+08],
       [-2.33712000e+07, -2.33712000e+07, -4.01472000e+05,
        -2.00736000e+05, -2.34410880e+07, -2.34410880e+07,
        -4.03556000e+05, -2.01778000e+05,  2.22568135e+10,
         2.22568135e+10, -3.59468034e+08,  1.79734017e+08],
       [ 4.01472000e+05,  4.01472000e+05,  7.71000000e+03,
         3.85500000e+03,  4.03555200e+05,  4.03555200e+05,
         7.77580000e+03,  3.88790000e+03, -3.59468034e+08,
        -3.59468034e+08,  6.18223201e+06, -3.09111600e+06],
       [-2.00736000e+05, -2.00736000e+05, -3.85500000e+03,
        -1.92750000e+03, -2.01777600e+05, -2.01777600e+05,
        -3.88790000e+03, -1.94395000e+03,  1.79734017e+08,
```

We get the parameter matrix by calculating the min values from the arr matrix

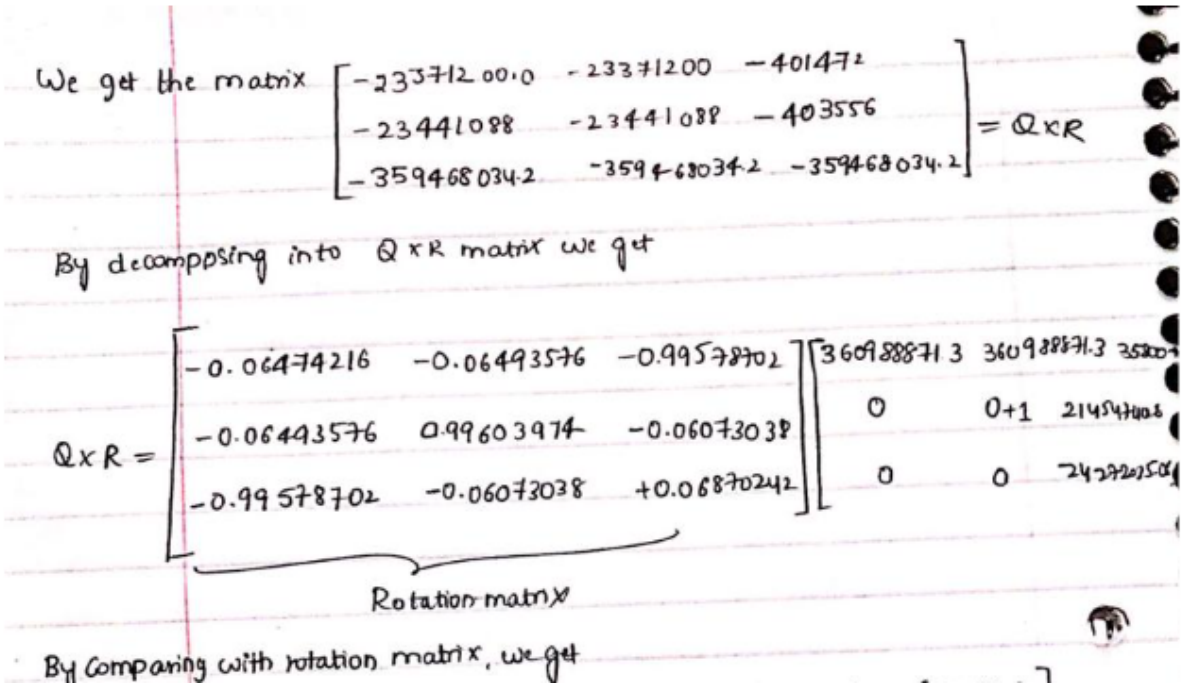
In [30]:
```
parameter=[]
for i in matmul:
    parameter.append(min(i))
```

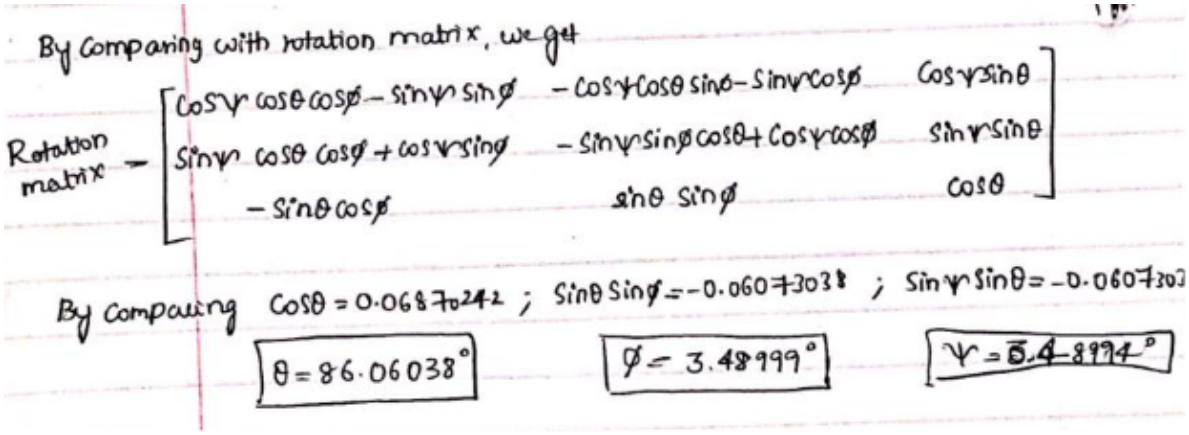
In [31]:
```
parameter
```

Out[31]:
```
[-23371200.0,
 -23371200.0,
 -401472.0,
 -200736.0,
 -23441088.0,
 -23441088.0,
 -403556.0,
 -201778.0,
 -359468034.2,
 -359468034.2,
 -359468034.2,
 -3091116.005]
```

Once you compute the Rotation matrix, you also need to compute the angles of rotation along each axis. Choose your order of rotation based on your experimentation setup.

We divide the parameter matrix into Q and R matrices using the qr decomposition and then we get the rotation matrix and the angles w.r.t x,y,z axes





2. Select any pair of images from the set in problem 1 above. Compute the homography between those two images.

Source image

Destination image

$$\begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{x}_d \\ \tilde{y}_d \\ \tilde{z}_d \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix}$$

(3×3 matrix)

- 9 unknowns
- 8 degrees of freedom
- min pair of matching points → 4

$$\frac{\tilde{x}_d}{\tilde{z}_d} = \frac{h_{11}x_s + h_{12}y_s + h_{13}}{h_{31}x_s + h_{32}y_s + h_{33}}, \quad \frac{\tilde{y}_d}{\tilde{z}_d} = \frac{h_{21}x_s + h_{22}y_s + h_{23}}{h_{31}x_s + h_{32}y_s + h_{33}}$$

$$\begin{bmatrix} x_s & y_s & 1 & 0 & 0 & 0 & -x_d y_s & -x_d y_s & -x_d \\ 0 & 0 & 0 & x_s & y_s & 1 & -y_d x_s & -y_d y_s & -y_d \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Then the matrix becomes

$$\begin{bmatrix} x_s & y_s & 1 & 0 & 0 & 0 & -x_d x_s & -x_d y_s & -x_d \\ 0 & 0 & 0 & x_s & y_s & 1 & -y_d x_s & -y_d y_s & -y_d \\ & & & & \vdots & & & & \\ x_s & y_s & 1 & 0 & 0 & 0 & -x_d x_s & -x_d y_s & -x_d \\ 0 & 0 & 0 & x_s & y_s & 1 & -y_d x_s & -y_d y_s & -y_d \\ x_s & y_s & 1 & 0 & 0 & 0 & -x_d x_s & -x_d y_s & -x_d \\ 0 & 0 & 0 & x_s & y_s & 1 & -y_d x_s & y_d y_s & -y_d \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

(A) known (h) unknown

$Ah = 0$ and $\|h\|^2 = 1$ and $\min_h \|Ah\|^2$ such that $\|h\|^2 = 1$

$\|Ah\|^2 = (Ah)^T(Ah) = h^T A^T A h$ and $\|h\|^2 = h^T h = 1$

$\min_h (h^T A^T A h)$ such that $h^T h = 1$

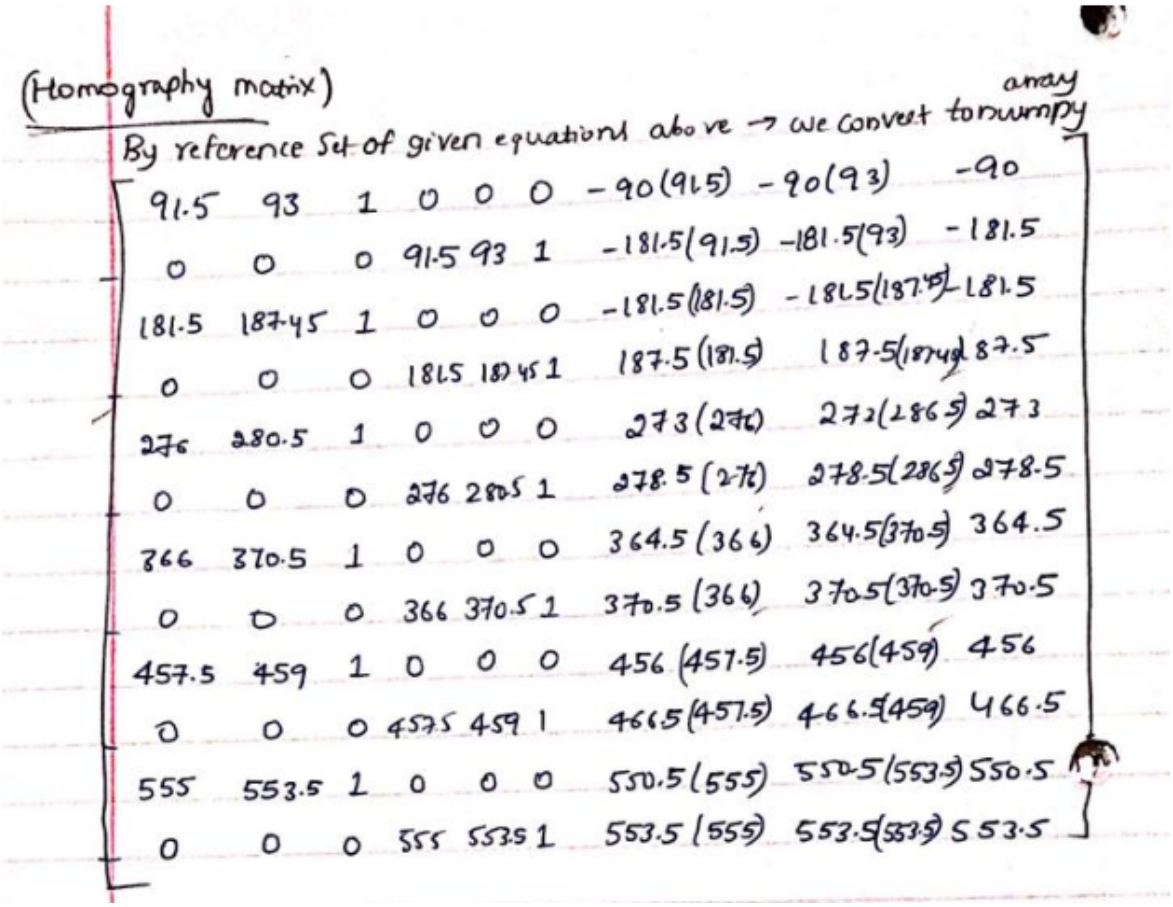
Noting the pixel values from the matlab for the image.

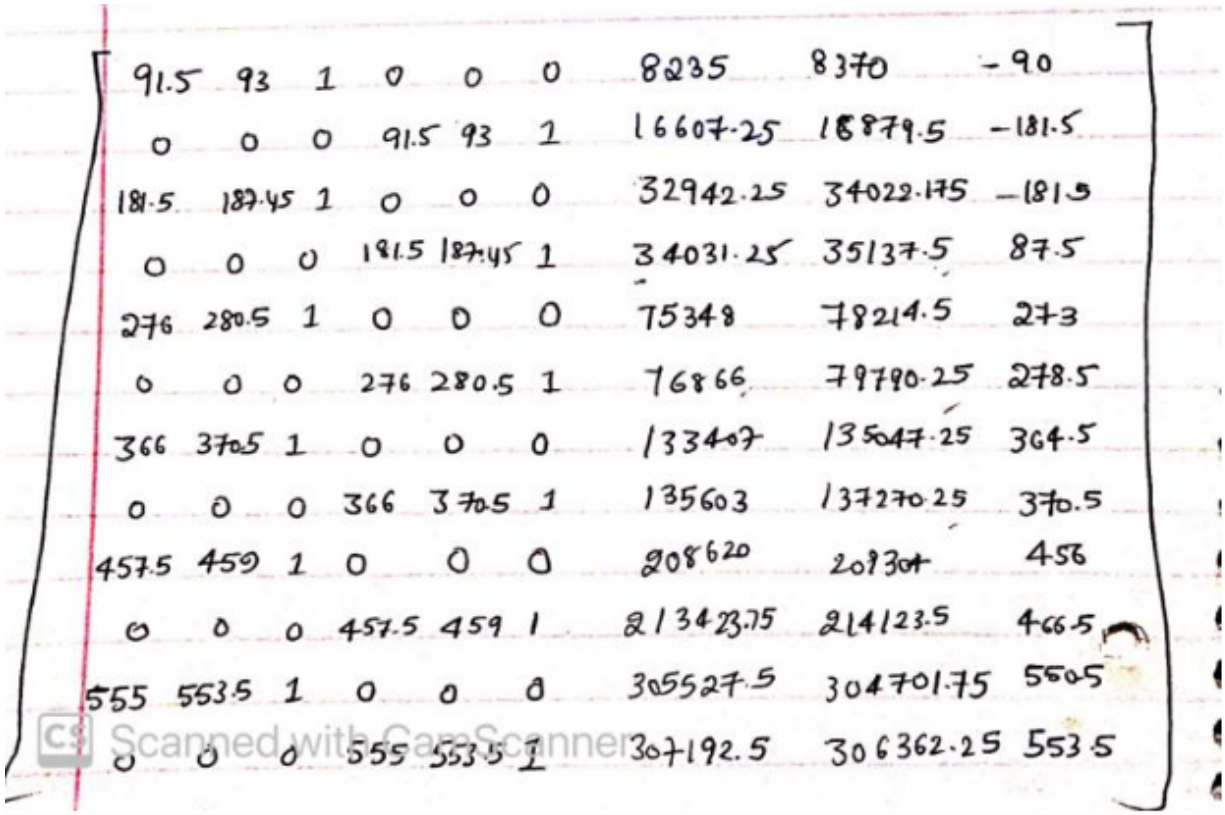
1.2 Homography

Other image (destination image co-ordinates)

	U (Pixels) / 2.75	V (Pixels) / 92.75	x_d (diff) U	y_d (diff) V
point 1	92.75	184.25	90	181.5
point 2	184.25	280.25	181.5	187.5
point 3	275.75	370.25	273	278.5
point 4	367.25	463.25	364.5	370.5
point 5	458.75	559.25	456	466.5
point 6	553.25	646.25	550.5	553.5

We convert the values into the numpy array by using standard set of reference equations





91.5	93	1	0	0	0	8235	8370	-90
0	0	0	91.5	93	1	16607.25	16879.5	-181.5
181.5	187.45	1	0	0	0	32942.25	34022.175	-181.5
0	0	0	181.5	187.45	1	34031.25	35137.5	87.5
276	280.5	1	0	0	0	75348	78214.5	273
0	0	0	276	280.5	1	76866	79790.25	278.5
366	3705	1	0	0	0	133407	135047.25	364.5
0	0	0	366	3705	1	135603	137270.25	370.5
457.5	459	1	0	0	0	208620	209304	456
0	0	0	457.5	459	1	213423.75	214123.5	466.5
555	553.5	1	0	0	0	305527.5	304701.75	5505
0	0	0	555	553.5	1	307192.5	306362.25	553.5

In [12]:
```
A=[[78,73.5,1,0,0,0,-5967,-5622.75,-76.5],
   [0,0,0,109.5,101.75,1,-11169,-10378.5,-102],
  [144,132,1,0,0,0,-20520,-18810,-142.5],
   [0,0,0,174,162,1,-28536,--26568,-164],
  [276,280.5,1,0,0,0,-75348,-78214.5,-273],
  [0,0,0,276,280.5,1,76866,79790.25,278.5]]
```

In [32]:
```
mat=[[78,73.5,1,0,0,0,-5967,-5622.75,-76.5],
   [0,0,0,109.5,101.75,1,-11169,-10378.5,-102],
  [144,132,1,0,0,0,-20520,-18810,-142.5],
   [0,0,0,174,162,1,-28536,--26568,-164],
  [276,280.5,1,0,0,0,-75348,-78214.5,-273],
  [0,0,0,276,280.5,1,76866,79790.25,278.5]]
```

In [33]:
```
mat=np.array(mat)
```

In [37]:
```
matrix= np.dot(mat.T,mat)
```

```
In [38]: matrix

Out[38]: array([[ 1.02996000e+05,  1.02159000e+05,  4.98000000e+02,
                  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
                 -2.42163540e+07, -2.47344165e+07, -1.01835000e+05],
                [ 1.02159000e+05,  1.01506500e+05,  4.86000000e+02,
                  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
                 -2.42823285e+07, -2.48353594e+07, -1.01009250e+05],
                [ 4.98000000e+02,  4.86000000e+02,  3.00000000e+00,
                  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
                 -1.01835000e+05, -1.02647250e+05, -4.92000000e+02],
                [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
                  1.18442250e+05,  1.16747625e+05,  5.59500000e+02,
                  1.50267465e+07,  2.55084952e+07,  3.71610000e+04],
                [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
                  1.16747625e+05,  1.15277312e+05,  5.44250000e+02,
                  1.58016352e+07,  2.56291688e+07,  4.11727500e+04],
                [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
                  5.59500000e+02,  5.44250000e+02,  3.00000000e+00,
                  3.71610000e+04,  9.59797500e+04,  1.25000000e+01],
                [-2.42163540e+07, -2.42823285e+07, -1.01835000e+05,
```

We get the Homographic matrix as below

```
In [39]: Homograp=[]
         for i in Res:
             Homograp.append(min(i))

In [40]: Homograp

Out[40]: [-24734416.5,
          -24835359.375,
          -102647.25,
          0.0,
          0.0,
          0.0,
          -24282328.5,
          -24835359.375,
          -101835.0]
```

PART B: MATLAB Prototyping

3. Write a MATLAB script to find the real world dimensions (e.g. diameter of a ball, side length of a cube) of an object using perspective projection equations. Validate using an experiment where you image an object using your camera from a specific distance (choose any distance but ensure you are able to measure it accurately) between the object and camera.

MATLAB's *ginput()* function to record the pixel coordinates of the object's end points

```
I = imread('image1.png');
imshow(I);
[x y] = ginput(2);
I have used the above function to calculate the pixel values.
```

(3)

Capturing chess Board images → distance 34 inches →

$Z = 863.6$ mm

captured chess Board → calibrated x, y values are

$x = [164 \quad 210]$

$y = [88 \quad 143]$

then $(x_1, y_1) = (164, 88)$, $(x_2, y_2) = (210, 143)$ } The co-ordinates of image

focal length = $(22.6247, 25.8989)$
$f_x \quad f_y$

$$x_{real} = \left(\frac{x_{image}}{f_x}\right) z \qquad y_{real} = \left(\frac{y_{image}}{f_y}\right) z$$

$$x_{1\,real} = \left(\frac{164}{22.6247}\right) 863.6 = 6259.99$$

$$y_{1\,real} = \left(\frac{88}{22.6247}\right) 863.6 = 3359.01$$

$$x_{2\,real} = \left(\frac{210}{25.8989}\right) 863.6 = 7002.45956$$

$$y_2 \text{ real} = \left(\frac{143}{25.8989}\right) 863.6 = 4768.34151$$

We calculate euclidean dist

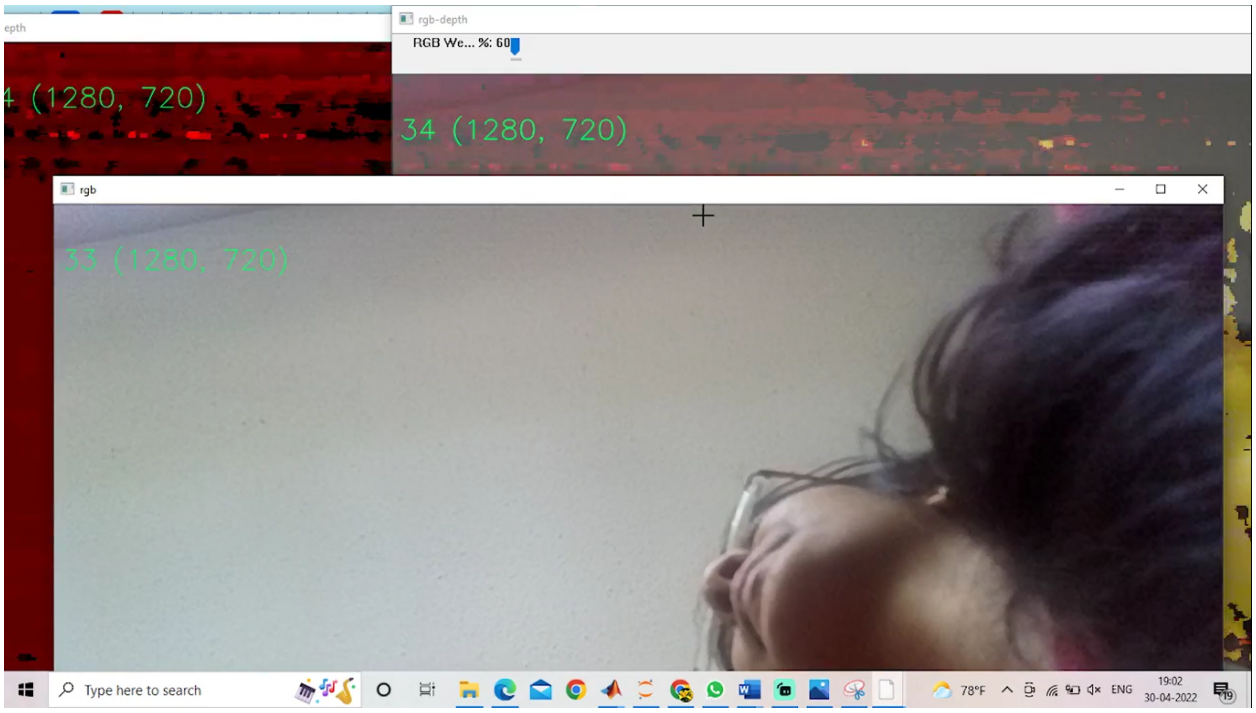
$$= \sqrt{(7002.45956 - 6259.99)^2 + (4768.34151 - 3359.)^2}$$

$$= \sqrt{674041}$$

$$= 821 \text{ mm}$$

The distances are almost equal.

5. We got the following image when we run that RGB stream from the mono camera and a depth map stream from the stereo cameras:
Yes, it is feasible.



The frame rate and the resolution are given and The above shown are rgb,rgb depth and depth
The video link is given below:
https://drive.google.com/file/d/15TpAQPqmpDjYLqVUDiWhvflEcTethtfi/view?usp=sharing

6. Run the camera calibration tutorial. Compare the output with answers from Part A and Matlab calibration exercise

The calibrated parameters in the matlab are as follows: