CSE 472 AR HW3 Report
Yakup Talha Yolcu
1801042609

Part 1:

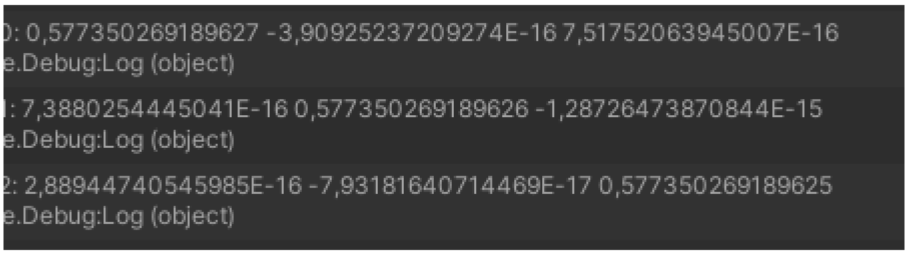
1.1 -> In this problem we are given a set of s and I matrices, we are to find corresponding homography matrix.

```
    public double[,] Smatrix = new double[4, 2] { { 1, 2 }, { 5, 3 }, {
4, 10 }, { 1, 6 } };
    public double[,] Imatrix = new double[4, 2] { { 1, 2 }, { 5, 3 }, {
4, 10 }, { 1, 6 } };
```

I gave 4 x y and 4 u v points. Because 4 points is sufficient to calculate homography matrix.

$$p_i = \begin{bmatrix} -x_i & -y_i & -1 & 0 & 0 & 0 & x_i x'_i & y_i x'_i & x'_i \\ 0 & 0 & 0 & -x_i & -y_i & -1 & x_i y'_i & y_i y'_i & y'_i \end{bmatrix}$$

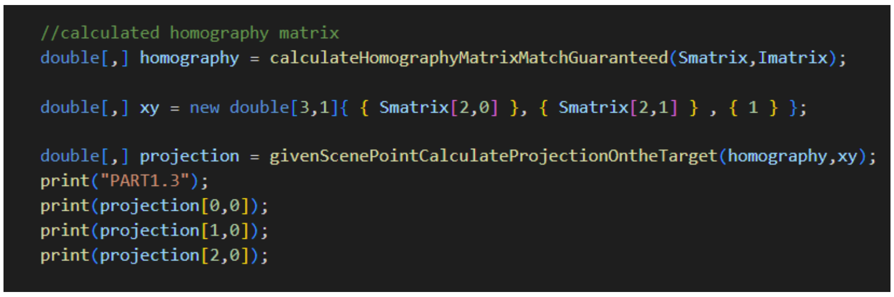
Then I created the pi matrix. Then created the 4 pi matrices according the given points
Then I solved the linear equation which PH = 0,
Found the homography matrix like that :



```
: 0,577350269189627 -3,90925237209274E-16 7,51752063945007E-16
e.Debug:Log (object)
: 7,3880254445041E-16 0,577350269189626 -1,28726473870844E-15
e.Debug:Log (object)
: 2,88944740545985E-16 -7,93181640714469E-17 0,577350269189625
e.Debug:Log (object)
```

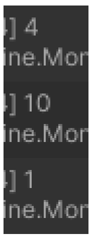
1.2 -> I did not implement this part

1.3 -> In this part we are given scene point which has xi and yi, we are expected to calculate image points (ui and vi)



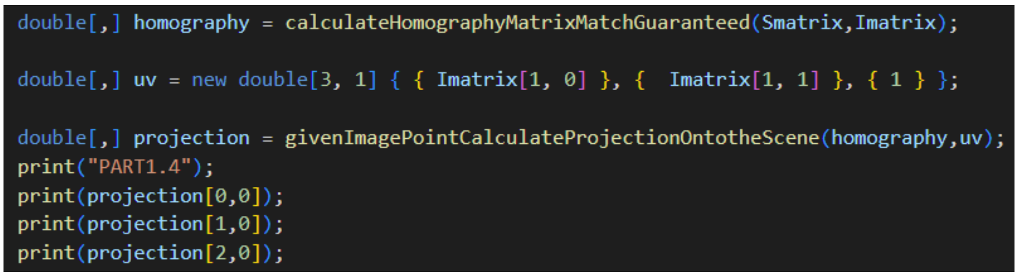
```
//calculated homography matrix
double[,] homography = calculateHomographyMatrixMatchGuaranteed(Smatrix,Imatrix);

double[,] xy = new double[3,1]{ { Smatrix[2,0] }, { Smatrix[2,1] } , { 1 } };

double[,] projection = givenScenePointCalculateProjectionOntheTarget(homography,xy);
print("PART1.3");
print(projection[0,0]);
print(projection[1,0]);
print(projection[2,0]);
```

Result :

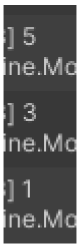


```
] 4
ne.Mor
] 10
ne.Mor
] 1
ne.Mor
```

1.4 -> In this part we are given image point (ui and vi) and homography, we are expected to calculate scene points

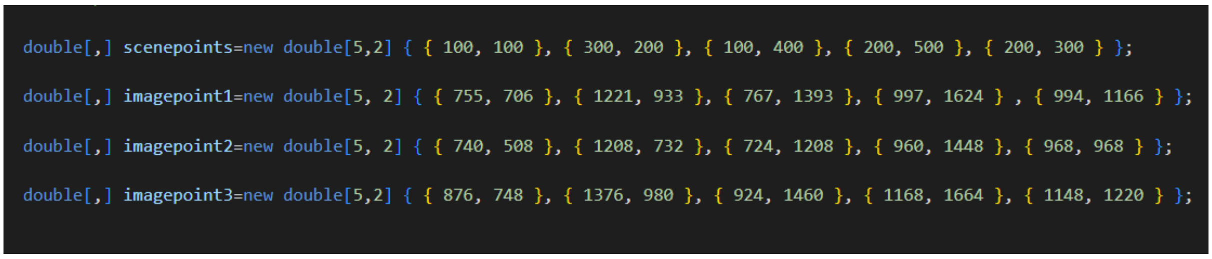


```
double[,] homography = calculateHomographyMatrixMatchGuaranteed(Smatrix,Imatrix);

double[,] uv = new double[3, 1] { { Imatrix[1, 0] }, {  Imatrix[1, 1] }, { 1 } };

double[,] projection = givenImagePointCalculateProjectionOntotheScene(homography,uv);
print("PART1.4");
print(projection[0,0]);
print(projection[1,0]);
print(projection[2,0]);
```



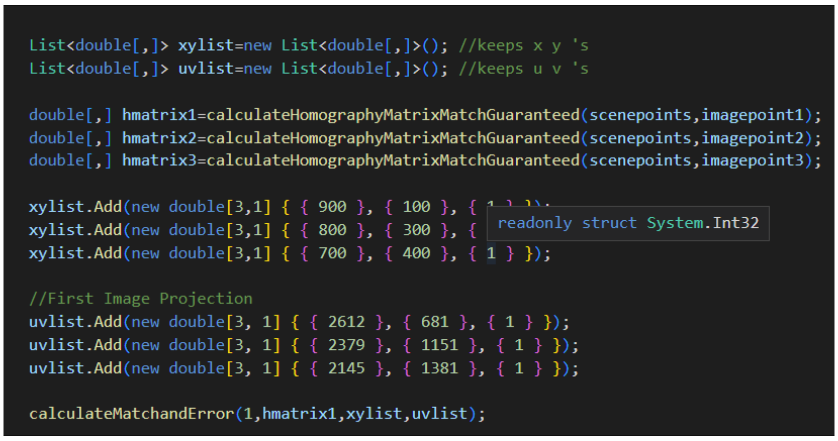
```
] 5
ine.Mo
] 3
ine.Mo
] 1
ine.Mo
```

1.5 -> We are to find 5 correspondences in 3 images manually.
My scene and image points :

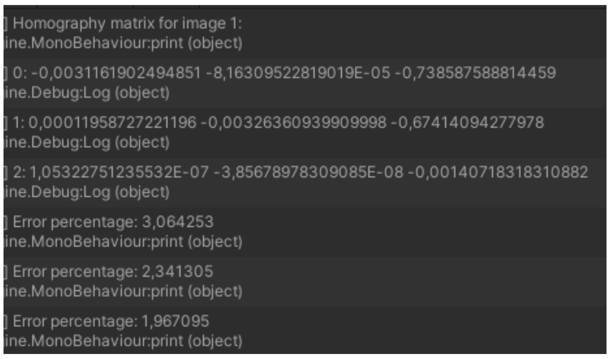


```
double[,] scenepoints=new double[5,2] { { 100, 100 }, { 300, 200 }, { 100, 400 }, { 200, 500 }, { 200, 300 } };

double[,] imagepoint1=new double[5, 2] { { 755, 706 }, { 1221, 933 }, { 767, 1393 }, { 997, 1624 } , { 994, 1166 } };

double[,] imagepoint2=new double[5, 2] { { 740, 508 }, { 1208, 732 }, { 724, 1208 }, { 960, 1448 }, { 968, 968 } };

double[,] imagepoint3=new double[5,2] { { 876, 748 }, { 1376, 980 }, { 924, 1460 }, { 1168, 1664 }, { 1148, 1220 } };
```

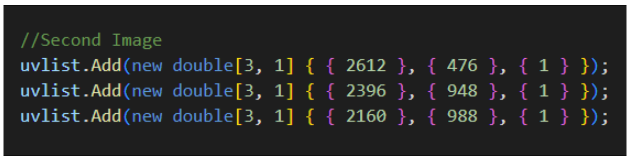
Calculating homography matrices for each image and calculating matches/projections and errors :



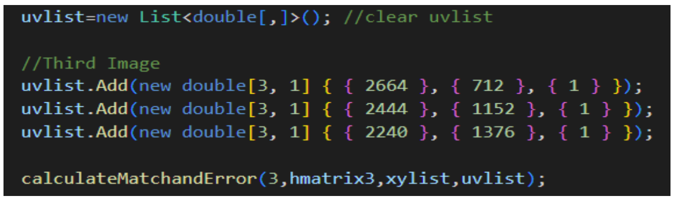
```
List<double[,]> xylist=new List<double[,]>(); //keeps x y 's
List<double[,]> uvlist=new List<double[,]>(); //keeps u v 's

double[,] hmatrix1=calculateHomographyMatrixMatchGuaranteed(scenepoints,imagepoint1);
double[,] hmatrix2=calculateHomographyMatrixMatchGuaranteed(scenepoints,imagepoint2);
double[,] hmatrix3=calculateHomographyMatrixMatchGuaranteed(scenepoints,imagepoint3);

xylist.Add(new double[3,1] { { 900 }, { 100 }, {
xylist.Add(new double[3,1] { { 800 }, { 300 }, {  readonly struct System.Int32
xylist.Add(new double[3,1] { { 700 }, { 400 }, { 1 } });

//First Image Projection
uvlist.Add(new double[3, 1] { { 2612 }, { 681 }, { 1 } });
uvlist.Add(new double[3, 1] { { 2379 }, { 1151 }, { 1 } });
uvlist.Add(new double[3, 1] { { 2145 }, { 1381 }, { 1 } });

calculateMatchandError(1,hmatrix1,xylist,uvlist);
```



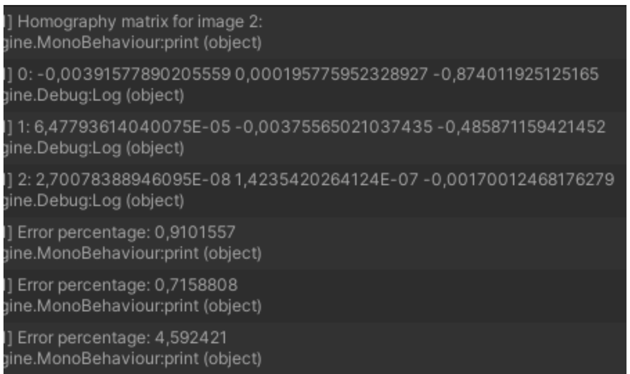
```
] Homography matrix for image 1:
ine.MonoBehaviour:print (object)
] 0: -0,0031161902494851 -8,16309522819019E-05 -0,738587588814459
ine.Debug:Log (object)
] 1: 0,00011958727221196 -0,00326360939909998 -0,67414094277978
ine.Debug:Log (object)
] 2: 1,05322751235532E-07 -3,85678978309085E-08 -0,00140718318310882
ine.Debug:Log (object)
] Error percentage: 3,064253
ine.MonoBehaviour:print (object)
] Error percentage: 2,341305
ine.MonoBehaviour:print (object)
] Error percentage: 1,967095
ine.MonoBehaviour:print (object)
```



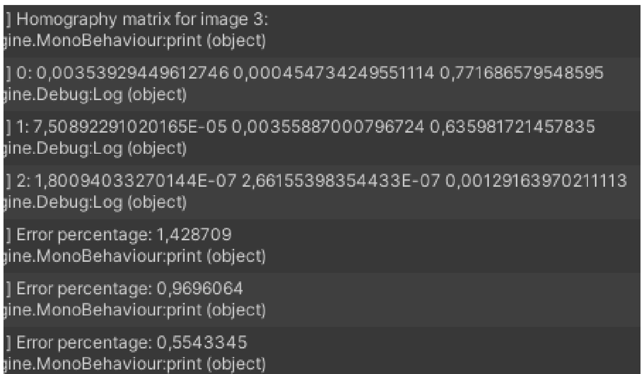
```
//Second Image
uvlist.Add(new double[3, 1] { { 2612 }, { 476 }, { 1 } });
uvlist.Add(new double[3, 1] { { 2396 }, { 948 }, { 1 } });
uvlist.Add(new double[3, 1] { { 2160 }, { 988 }, { 1 } });
```



```
uvlist=new List<double[,]>(); //clear uvlist

//Third Image
uvlist.Add(new double[3, 1] { { 2664 }, { 712 }, { 1 } });
uvlist.Add(new double[3, 1] { { 2444 }, { 1152 }, { 1 } });
uvlist.Add(new double[3, 1] { { 2240 }, { 1376 }, { 1 } });

calculateMatchandError(3,hmatrix3,xylist,uvlist);
```

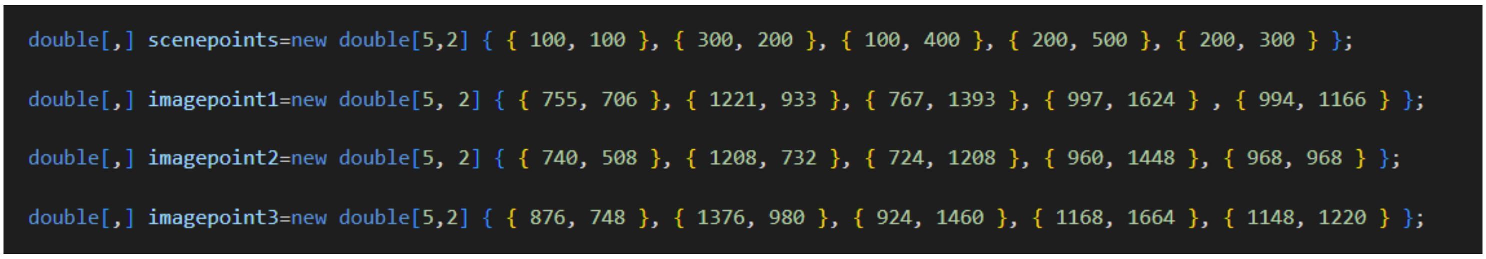


```
]] Homography matrix for image 2:
gine.MonoBehaviour:print (object)
]] 0: -0,00391577890205559 0,000195775952328927 -0,874011925125165
gine.Debug:Log (object)
]] 1: 6,47793614040075E-05 -0,00375565021037435 -0,485871159421452
gine.Debug:Log (object)
]] 2: 2,70078388946095E-08 1,4235420264124E-07 -0,00170012468176279
gine.Debug:Log (object)
]] Error percentage: 0,9101557
gine.MonoBehaviour:print (object)
]] Error percentage: 0,7158808
gine.MonoBehaviour:print (object)
]] Error percentage: 4,592421
gine.MonoBehaviour:print (object)
```



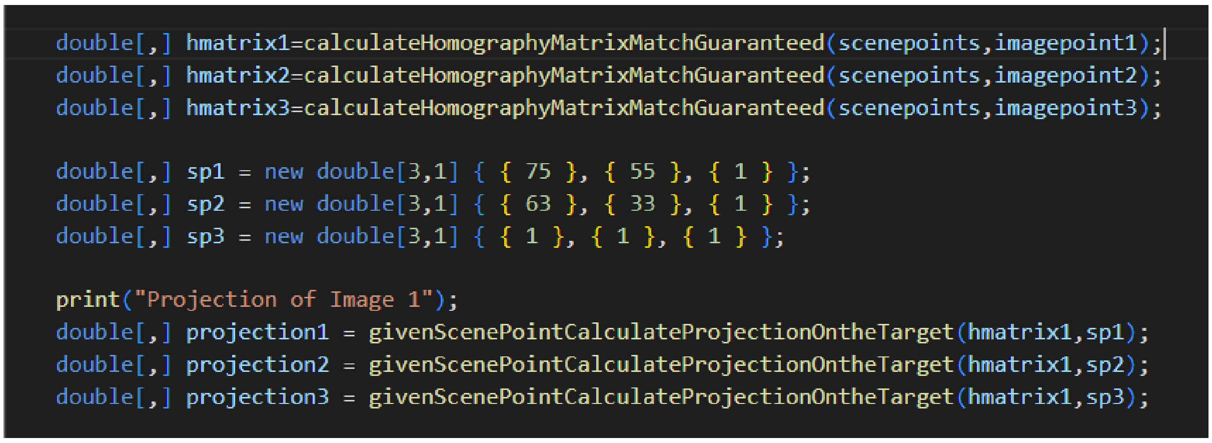
```
] Homography matrix for image 3:
gine.MonoBehaviour:print (object)
] 0: 0,00353929449612746 0,000454734249551114 0,771686579548595
gine.Debug:Log (object)
] 1: 7,50892291020165E-05 0,00355887000796724 0,635981721457835
gine.Debug:Log (object)
] 2: 1,80094033270144E-07 2,66155398354433E-07 0,00129163970211113
gine.Debug:Log (object)
] Error percentage: 1,428709
gine.MonoBehaviour:print (object)
] Error percentage: 0,9696064
gine.MonoBehaviour:print (object)
] Error percentage: 0,5543345
gine.MonoBehaviour:print (object)
```

1.6 -> We are given scene points and we are expected the calculate projection of the given scene points.

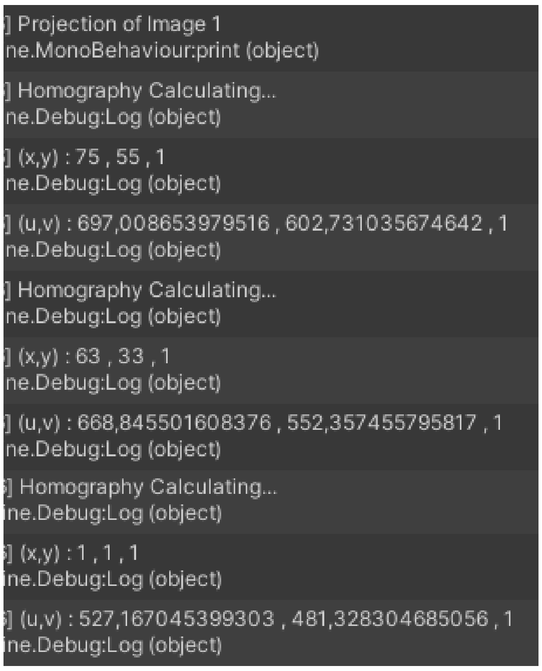


```
double[,] scenepoints=new double[5,2] { { 100, 100 }, { 300, 200 }, { 100, 400 }, { 200, 500 }, { 200, 300 } };

double[,] imagepoint1=new double[5, 2] { { 755, 706 }, { 1221, 933 }, { 767, 1393 }, { 997, 1624 } , { 994, 1166 } };

double[,] imagepoint2=new double[5, 2] { { 740, 508 }, { 1208, 732 }, { 724, 1208 }, { 960, 1448 }, { 968, 968 } };

double[,] imagepoint3=new double[5,2] { { 876, 748 }, { 1376, 980 }, { 924, 1460 }, { 1168, 1664 }, { 1148, 1220 } };
```

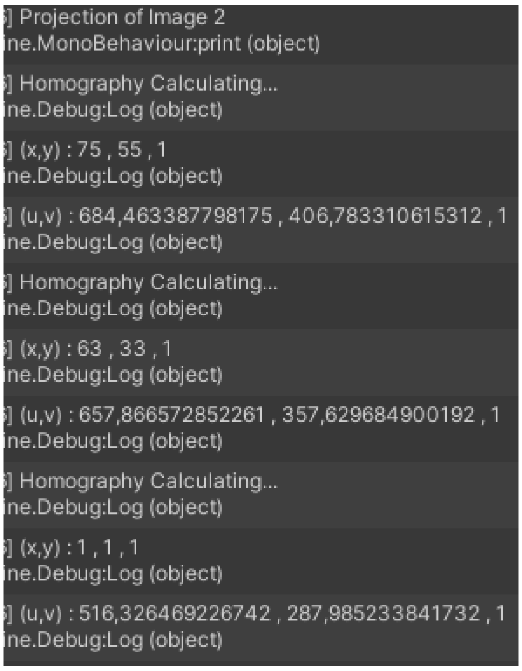
Calculate homographies :



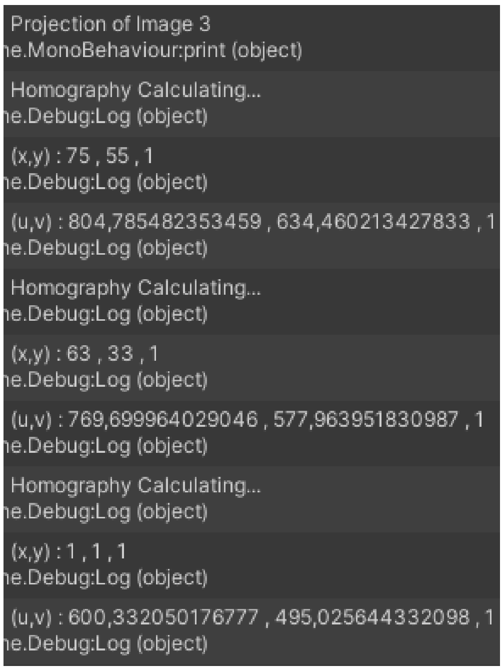
```
    double[,] hmatrix1=calculateHomographyMatrixMatchGuaranteed(scenepoints,imagepoint1);
    double[,] hmatrix2=calculateHomographyMatrixMatchGuaranteed(scenepoints,imagepoint2);
    double[,] hmatrix3=calculateHomographyMatrixMatchGuaranteed(scenepoints,imagepoint3);

    double[,] sp1 = new double[3,1] { { 75 }, { 55 }, { 1 } };
    double[,] sp2 = new double[3,1] { { 63 }, { 33 }, { 1 } };
    double[,] sp3 = new double[3,1] { { 1 }, { 1 }, { 1 } };

    print("Projection of Image 1");
    double[,] projection1 = givenScenePointCalculateProjectionOntheTarget(hmatrix1,sp1);
    double[,] projection2 = givenScenePointCalculateProjectionOntheTarget(hmatrix1,sp2);
    double[,] projection3 = givenScenePointCalculateProjectionOntheTarget(hmatrix1,sp3);
```



```
] Projection of Image 1
ne.MonoBehaviour:print (object)
] Homography Calculating...
ne.Debug:Log (object)
] (x,y) : 75 , 55 , 1
ne.Debug:Log (object)
] (u,v) : 697,008653979516 , 602,731035674642 , 1
ne.Debug:Log (object)
] Homography Calculating...
ne.Debug:Log (object)
] (x,y) : 63 , 33 , 1
ne.Debug:Log (object)
] (u,v) : 668,845501608376 , 552,357455795817 , 1
ne.Debug:Log (object)
] Homography Calculating...
ine.Debug:Log (object)
] (x,y) : 1 , 1 , 1
ine.Debug:Log (object)
] (u,v) : 527,167045399303 , 481,328304685056 , 1
ine.Debug:Log (object)
```

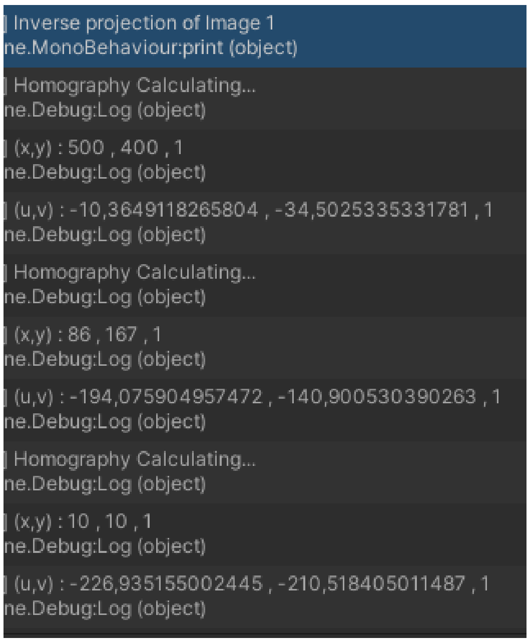


```
] Projection of Image 2
ine.MonoBehaviour:print (object)
] Homography Calculating...
ine.Debug:Log (object)
] (x,y) : 75 , 55 , 1
ine.Debug:Log (object)
] (u,v) : 684,463387798175 , 406,783310615312 , 1
ine.Debug:Log (object)
] Homography Calculating...
ine.Debug:Log (object)
] (x,y) : 63 , 33 , 1
ine.Debug:Log (object)
] (u,v) : 657,866572852261 , 357,629684900192 , 1
ine.Debug:Log (object)
] Homography Calculating...
ine.Debug:Log (object)
] (x,y) : 1 , 1 , 1
ine.Debug:Log (object)
] (u,v) : 516,326469226742 , 287,985233841732 , 1
ine.Debug:Log (object)
```

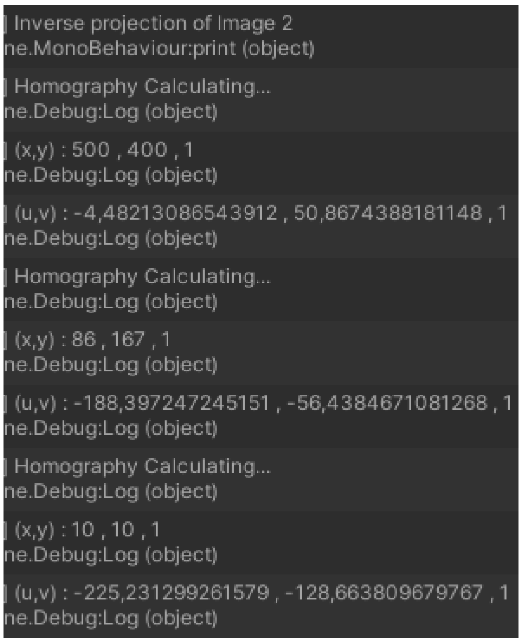


```
Projection of Image 3
e.MonoBehaviour:print (object)
Homography Calculating...
e.Debug:Log (object)
(x,y) : 75 , 55 , 1
e.Debug:Log (object)
(u,v) : 804,785482353459 , 634,460213427833 , 1
e.Debug:Log (object)
Homography Calculating...
e.Debug:Log (object)
(x,y) : 63 , 33 , 1
e.Debug:Log (object)
(u,v) : 769,699964029046 , 577,963951830987 , 1
e.Debug:Log (object)
Homography Calculating...
e.Debug:Log (object)
(x,y) : 1 , 1 , 1
e.Debug:Log (object)
(u,v) : 600,332050176777 , 495,025644332098 , 1
e.Debug:Log (object)
```

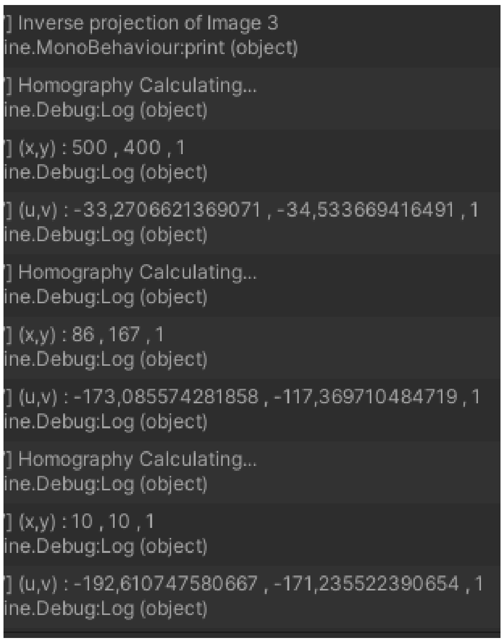
1.7 -> We are given image points and we are expected to calculate scene points



```
] Inverse projection of Image 1
ne.MonoBehaviour:print (object)
] Homography Calculating...
ne.Debug:Log (object)
] (x,y) : 500 , 400 , 1
ne.Debug:Log (object)
] (u,v) : -10,3649118265804 , -34,5025335331781 , 1
ne.Debug:Log (object)
] Homography Calculating...
ne.Debug:Log (object)
] (x,y) : 86 , 167 , 1
ne.Debug:Log (object)
] (u,v) : -194,075904957472 , -140,900530390263 , 1
ne.Debug:Log (object)
] Homography Calculating...
ne.Debug:Log (object)
] (x,y) : 10 , 10 , 1
ne.Debug:Log (object)
] (u,v) : -226,935155002445 , -210,518405011487 , 1
ne.Debug:Log (object)
```



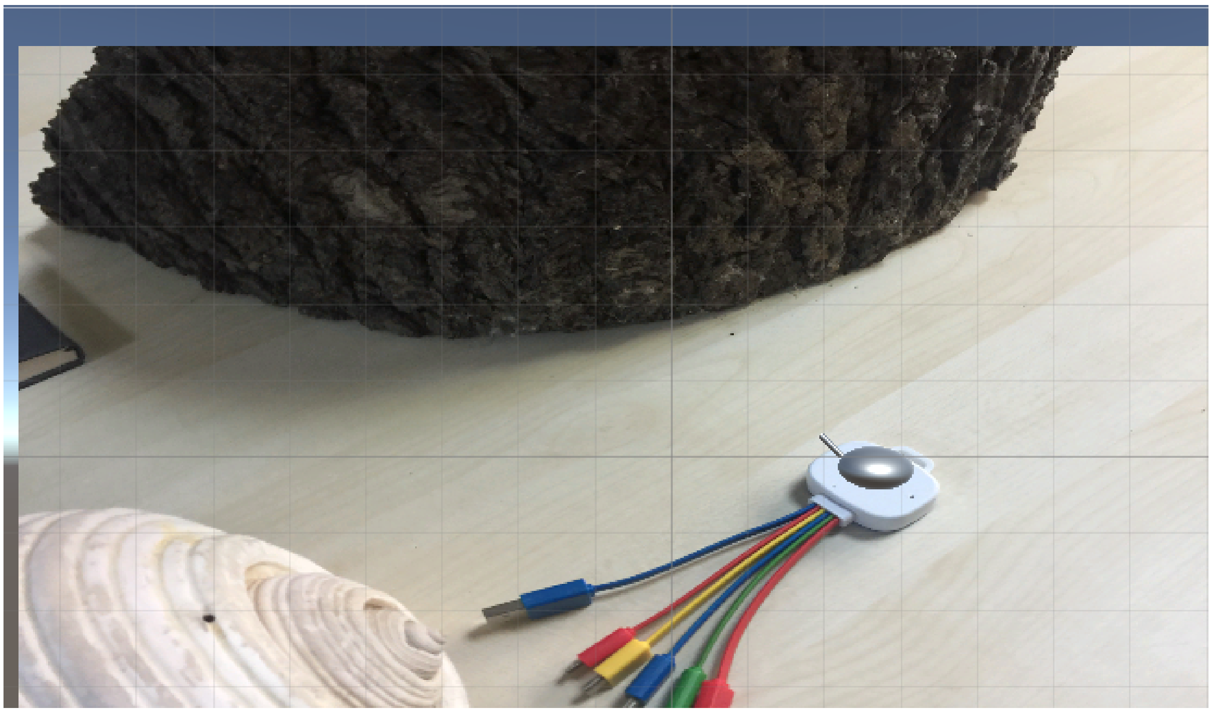
```
] Inverse projection of Image 2
ne.MonoBehaviour:print (object)
] Homography Calculating...
ne.Debug:Log (object)
] (x,y) : 500 , 400 , 1
ne.Debug:Log (object)
] (u,v) : -4,48213086543912 , 50,8674388181148 , 1
ne.Debug:Log (object)
] Homography Calculating...
ne.Debug:Log (object)
] (x,y) : 86 , 167 , 1
ne.Debug:Log (object)
] (u,v) : -188,397247245151 , -56,4384671081268 , 1
ne.Debug:Log (object)
] Homography Calculating...
ne.Debug:Log (object)
] (x,y) : 10 , 10 , 1
ne.Debug:Log (object)
] (u,v) : -225,231299261579 , -128,663809679767 , 1
ne.Debug:Log (object)
```



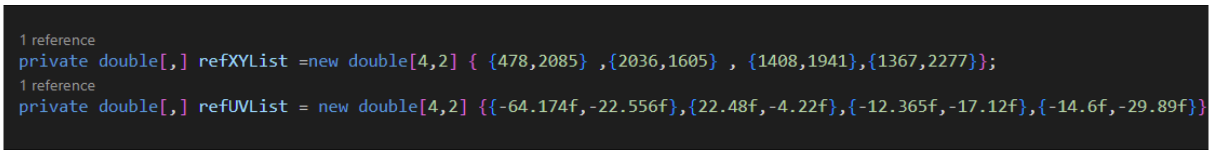
```
] Inverse projection of Image 3
ine.MonoBehaviour:print (object)
] Homography Calculating...
ine.Debug:Log (object)
] (x,y) : 500 , 400 , 1
ine.Debug:Log (object)
] (u,v) : -33,2706621369071 , -34,533669416491 , 1
ine.Debug:Log (object)
] Homography Calculating...
ine.Debug:Log (object)
] (x,y) : 86 , 167 , 1
ine.Debug:Log (object)
] (u,v) : -173,085574281858 , -117,369710484719 , 1
ine.Debug:Log (object)
] Homography Calculating...
ine.Debug:Log (object)
] (x,y) : 10 , 10 , 1
ine.Debug:Log (object)
] (u,v) : -192,610747580667 , -171,235522390654 , 1
ine.Debug:Log (object)
```

Part 2

In this part we have a teapot and selected reference image. Then we are expected to apply this projection to other 18 images respectively. Object should be stayed on the same point.
My reference image

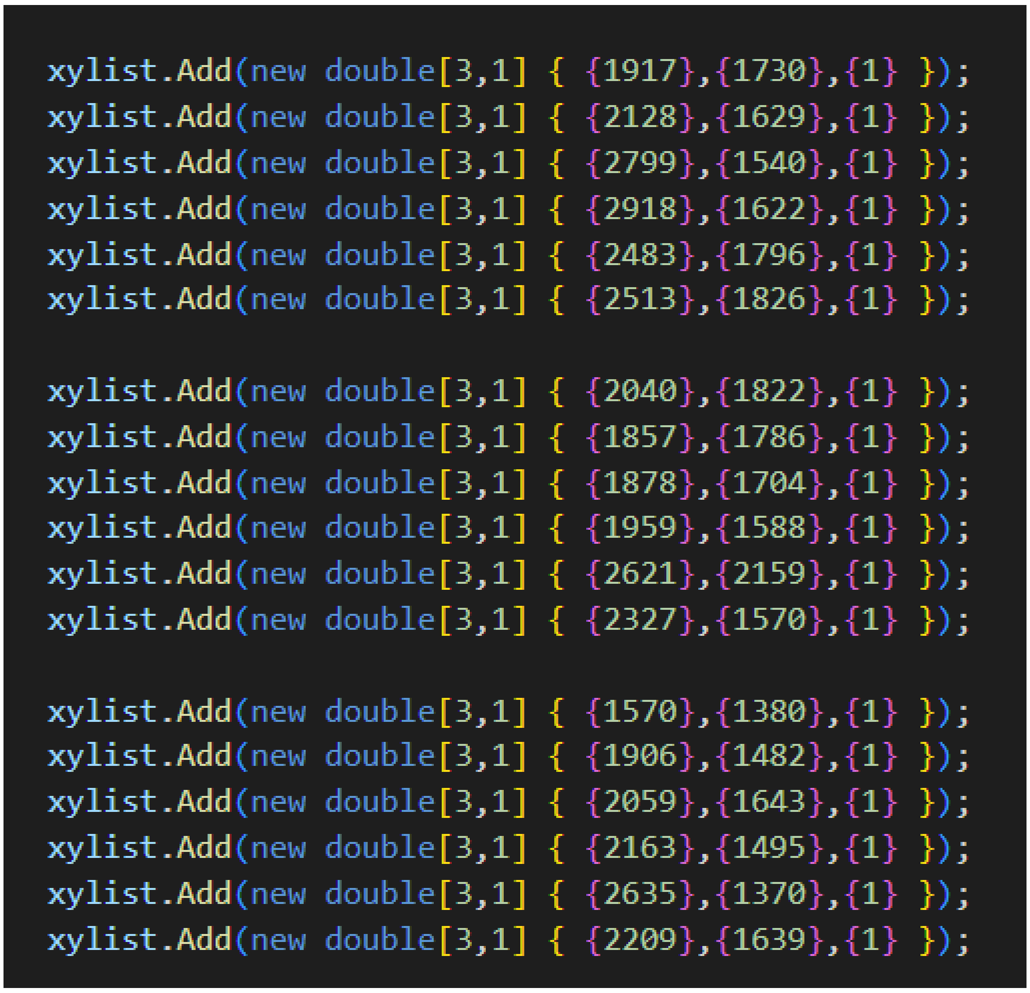


I determined as the 4 XY and 4 UV points by using paint and a sphere in the Unity



```
1 reference
private double[,] refXYList =new double[4,2] { {478,2085} ,{2036,1605} , {1408,1941},{1367,2277}};
1 reference
private double[,] refUVList = new double[4,2] {{-64.174f,-22.556f},{22.48f,-4.22f},{-12.365f,-17.12f},{-14.6f,-29.89f}}
```

Then I take the pixels of the same spot.

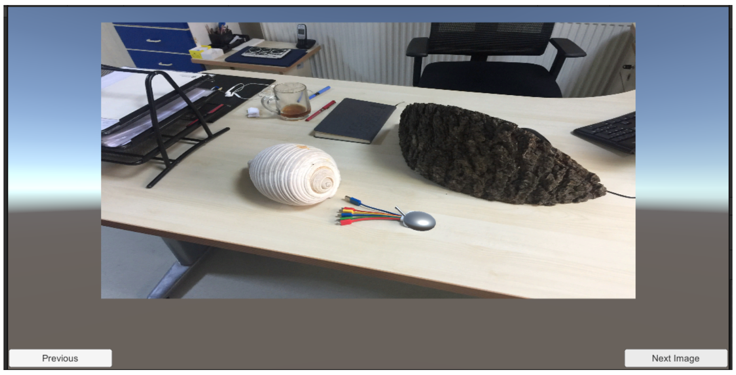


```
xylist.Add(new double[3,1] { {1917},{1730},{1} });
xylist.Add(new double[3,1] { {2128},{1629},{1} });
xylist.Add(new double[3,1] { {2799},{1540},{1} });
xylist.Add(new double[3,1] { {2918},{1622},{1} });
xylist.Add(new double[3,1] { {2483},{1796},{1} });
xylist.Add(new double[3,1] { {2513},{1826},{1} });

xylist.Add(new double[3,1] { {2040},{1822},{1} });
xylist.Add(new double[3,1] { {1857},{1786},{1} });
xylist.Add(new double[3,1] { {1878},{1704},{1} });
xylist.Add(new double[3,1] { {1959},{1588},{1} });
xylist.Add(new double[3,1] { {2621},{2159},{1} });
xylist.Add(new double[3,1] { {2327},{1570},{1} });

xylist.Add(new double[3,1] { {1570},{1380},{1} });
xylist.Add(new double[3,1] { {1906},{1482},{1} });
xylist.Add(new double[3,1] { {2059},{1643},{1} });
xylist.Add(new double[3,1] { {2163},{1495},{1} });
xylist.Add(new double[3,1] { {2635},{1370},{1} });
xylist.Add(new double[3,1] { {2209},{1639},{1} });
```

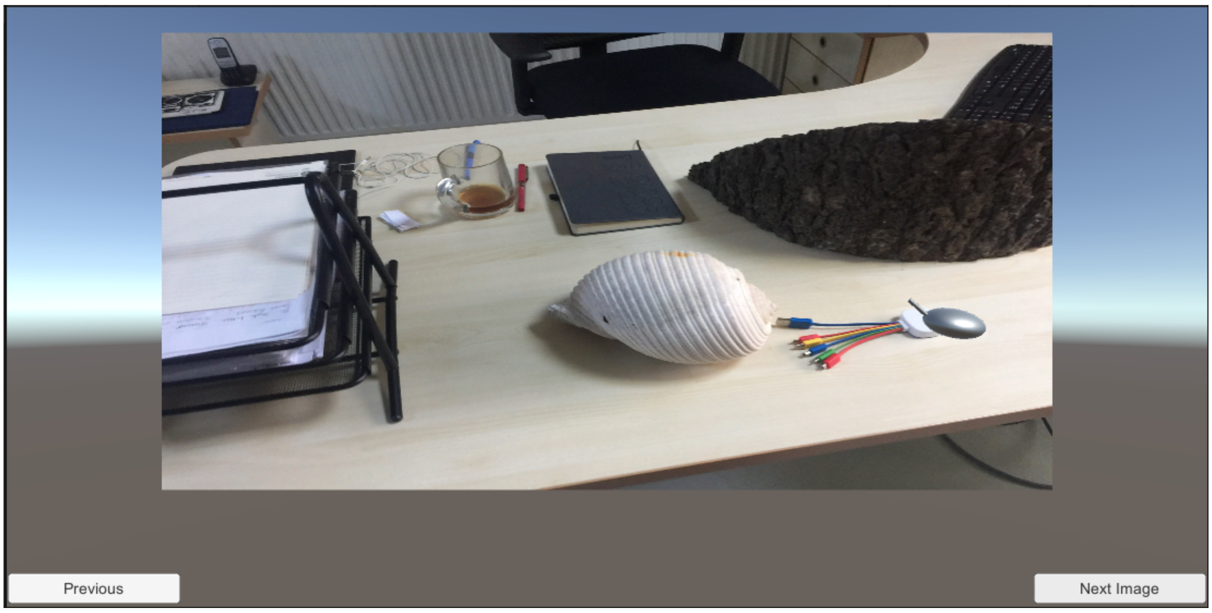
Then I calculate the homography matrix at the start of the program

```
homography_matrix=calculateHomographyMatrix(refXYList,refUVList);
```

My program is like :



I have a menu going backward and forward in the images in order. On each change, I calculate the U and V points by using homography matrix what I calculated before.


Previous
Next Image