Rishika Bhanushali
rb1182

CS534 Computer Vision Assignment-3

Camera Calibration and Augmented Reality

Table of Contents

TASK 1: Camera Calibration using 3D calibration object

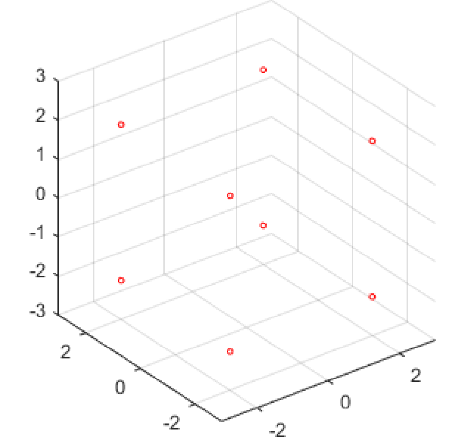
This involved camera calibration for a Robot vehicle.

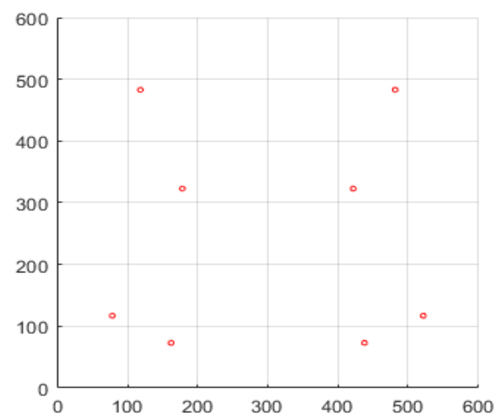
1.1 Outputs

Rishika Bhanushali
rb1182

1.1.1 Drawing Image Points

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





1.1.2 Matrix P

Write a Matlab 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 P (slide 30 of the Camera Calibration pdf document). This matrix P will be used to compute the 12 elements of the projection matrix M such that λpi = MPi

Code: get_P_matrix.m

1.1.3 Print Matrix P

Use this Matlab 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.

*Size(P)=16*12*

```
P_matrix = 2×12
     2     2     2     1     0     0     0     0  -844  -844  -844  -422
     0     0     0     0     2     2     2     1  -646  -646  -646  -323

P_matrix = 2×12
    -2     2     2     1     0     0     0     0   356  -356  -356  -178
     0     0     0     0    -2     2     2     1   646  -646  -646  -323

P_matrix = 2×12
    -2     2    -2     1     0     0     0     0   236  -236   236  -118
     0     0     0     0    -2     2    -2     1   966  -966   966  -483

P_matrix = 2×12
     2     2    -2     1     0     0     0     0  -964  -964   964  -482
     0     0     0     0     2     2    -2     1  -966  -966   966  -483

P_matrix = 2×12
     2    -2     2     1     0     0     0     0  -876   876  -876  -438
     0     0     0     0     2    -2     2     1  -146   146  -146   -73

P_matrix = 2×12
    -2    -2     2     1     0     0     0     0   324   324  -324  -162
     0     0     0     0    -2    -2     2     1   146   146  -146   -73

P_matrix = 2×12
    -2    -2    -2     1     0     0     0     0   156   156   156   -78
     0     0     0     0    -2    -2    -2     1   234   234   234  -117

P_matrix = 2×12
         2        -2        -2         1         0         0         0         0     -1044      1044      1044      -522
         0         0         0         0         2        -2        -2         1      -234       234       234      -117
```

Rishika Bhanushali
rb1182

1.1.4 Print Matrix M

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.

Q4: Print matrix M

```
M = get_M_matrix(P);
```

```
M_matrix = 3×4
    0.1925    0.0283    0.0786    0.7346
    0.0000    0.2044    0.0001    0.6120
    0.0000    0.0001    0.0003    0.0024
```

```
disp(M)
```

```
    0.1925    0.0283    0.0786    0.7346
    0.0000    0.2044    0.0001    0.6120
    0.0000    0.0001    0.0003    0.0024
```

1.1.5 Translational vector recovery

Now we need to recover the translation vector which is a null vector of M. Find the singular value decomposition of matrix $M = U\Sigma V^T$. 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.

Q5: Translational vector recovery

```
[m_U, m_Sigma, m_V_T] = svd(M);
t = m_V_T(:, end);
center = t(1: end - 1) / t(end)
```

```
center = 3×1
   -0.0000
   -2.9912
   -8.2695
```

```
disp(center)
```

```
   -0.0000
   -2.9912
   -8.2695
```

1.1.6 Print Matrix M’

onsider the 3x3 matrix M' composed of the rst 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 dened in slide 38 where the axes e1, e2, e3 are the x, y, z axes respectively. Therefore M' can be written as M0 = KRT z RT y RT

Q6: Print matrix M'

```
% Rescaling the elements of this matrix so that its element m33 becomes equal to 1
M_dash = M(:, 1:3) /  M(3, 3) %3*4
```

```
M_dash = 3×3
    734.6289  107.8955  299.9999
      0.0009  780.1442    0.2641
      0.0000    0.3597    1.0000
```

1.1.7 Rotation Matrix Rx

```
[R_x, Theta_x, N_matrix] = get_Rx_Thetax_N(M_dash);
```

```
R_x = 3×3
      1.0000         0         0
           0    0.9410    0.3384
           0   -0.3384    0.9410

Theta_x = -19.7812
N_matrix = 3×3
    734.6289   -0.0000  318.8125
      0.0009  734.0199  264.2723
      0.0000         0    1.0627
```

1.1.8 Rotation angle Theta

```
[R_z, Theta_z] = get_Rz_Thetaz(N_matrix);
```

```
R_z = 3×3
      1.0000    0.0000         0
     -0.0000    1.0000         0
           0         0    1.0000

Theta_z = -7.2204e-05
```

1.1.9 Compute and Print Matrix K

```
K = M_dash / (R_x * R_y);

% Print K
K = K / K(3,3)
```

```
K = 3×3
     896.6817  247.8504  299.9999
       0.0011  896.1566 -321.9619
       0.0000    0.8262    1.0000
```

```
focal_lengths = [K(1, 1), K(2, 2)]
```

```
focal_lengths = 1×2
     896.6817  896.1566
```

```
image_center = [K(1, 3), K(2, 3)]
```

```
image_center = 1×2
     299.9999 -321.9619
```

TASK 2: Camera Calibration using 2D calibration object

Executed separate code files for all the subfunctions

2.1 Outputs

2.1.1 Corner Extraction and Homography computation

images2.png, H is	images9.png, H is	images12.png, H is	images20.png, H is
Column 1	Column 1	Column 1	Column 1
0.9655 0.0172 -0.0000	-1.0919 -0.1477 -0.0005	0.7037 -0.1758 -0.0005	-0.8559 0.0113 0.0000
Column 2	Column 2	Column 2	Column 2
-0.0835 0.8809 -0.0002	0.0345 -0.9309 0.0001	-0.0492 0.8856 -0.0002	0.2712 -0.3995 0.0008
Column 3	Column 3	Column 3	Column 3
54.2980 43.4384 0.6033	-71.6419 -10.3829 -0.5191	75.9062 59.9260 0.6658	-122.6998 -57.9033 -0.6893

2.1.2 Computing Intrinsic and Extrinsic parameters

```
B_matrix =

   -0.0000    0.0000    0.0005
    0.0000   -0.0000    0.0004
    0.0005    0.0004   -1.0000
```

v_0 = 230.3654

lambda = -0.7692

alpha =723.3788

beta = 702.7882

gamma =0.6580

u_0 =317.6932

```
A_matrix =

  723.3788    0.6580  317.6932
         0  702.7882  230.3654
         0         0    1.0000
```

```
image = images2.png

R_matrix =

    0.9998   -0.0149    0.0027
    0.0203    0.9864    0.1636
   -0.0061   -0.1636    0.9865


T_matrix =

 -141.5014
 -101.3666
  449.8412


R_T_R_matrix =

    1.0000    0.0061   -0.0000
    0.0061    1.0000         0
   -0.0000         0    1.0000
```

```
image = images9.png

R_matrix =

   -0.9241   -0.0119   -0.3818
   -0.0271   -0.9941    0.1042
   -0.3811    0.1078    0.9183


T_matrix =

   93.1975
  112.4254
 -375.5904


R_T_R_matrix =

    1.0000   -0.0031    0.0000
   -0.0031    1.0000   -0.0000
    0.0000   -0.0000    1.0000
```

```
image = images12.png

R_matrix =

    0.9112    0.0035    0.4119
   -0.0544    0.9919    0.1141
   -0.4083   -0.1268    0.9041


T_matrix =

 -141.1402
 -100.1742
  501.5568


R_T_R_matrix =

    1.0000    0.0010    0.0000
    0.0010    1.0000    0.0000
    0.0000    0.0000    1.0000
```

```
image = images20.png

R_matrix =

   -0.9999    0.0071    0.0150
    0.0098   -0.7097    0.7044
    0.0114    0.7044    0.7096


T_matrix =

  111.9242
  120.8244
 -580.1506


R_T_R_matrix =

    1.0000   -0.0060    0.0000
   -0.0060    1.0000    0.0000
    0.0000    0.0000    1.0000
```

2.1.3 Projected Grid corners, improving accuracy

I) Projected Grid Corners

Figure 1: Projected grid corners

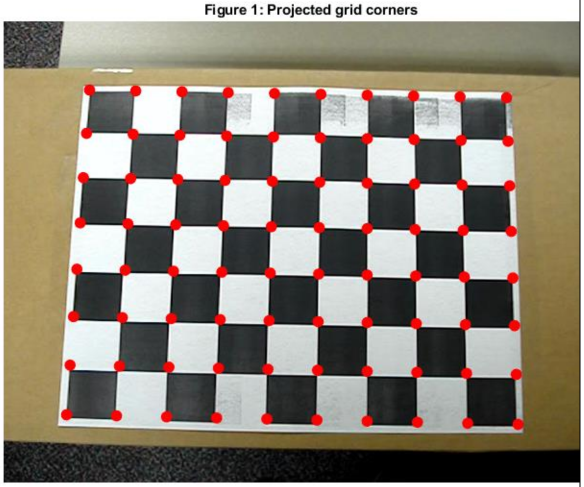


Figure 2: Projected grid corners

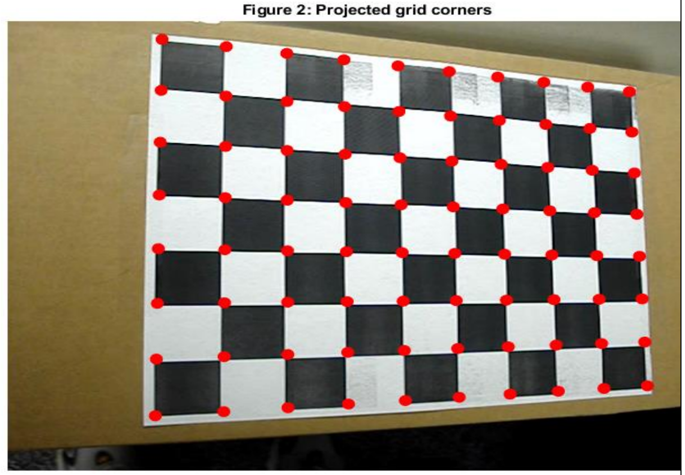


Figure 3: Projected grid corners

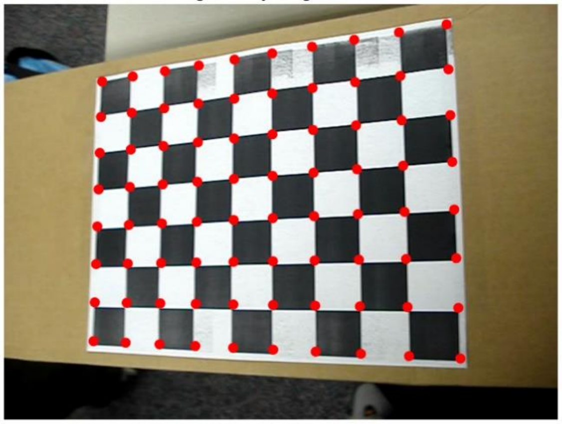


Figure 4: Projected grid corners

2)Harris corners

Figure 1: Harris corners

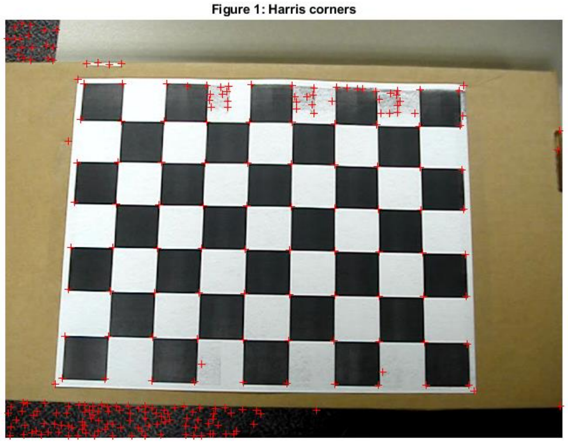


Figure 3: Harris corners

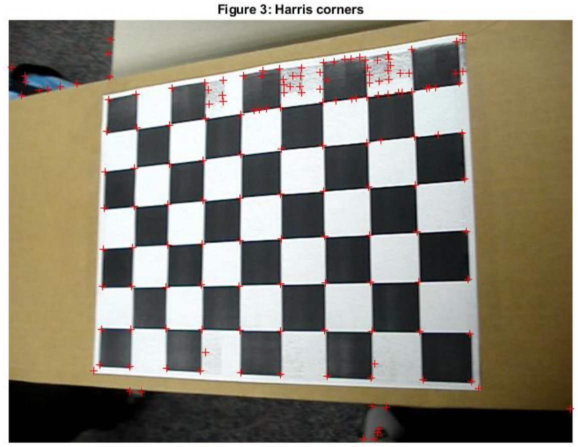


Figure 2: Harris corners

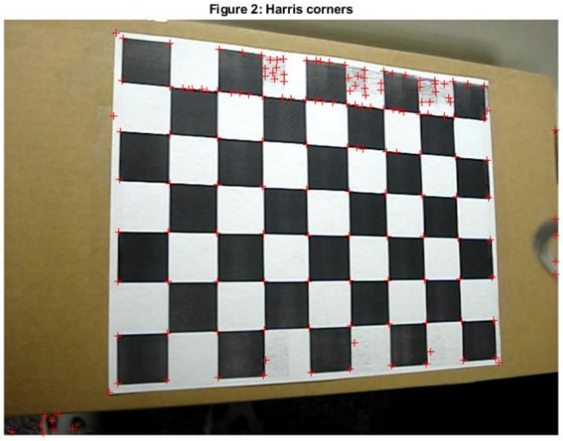


Figure 4: Harris corners

3) Grid Points:

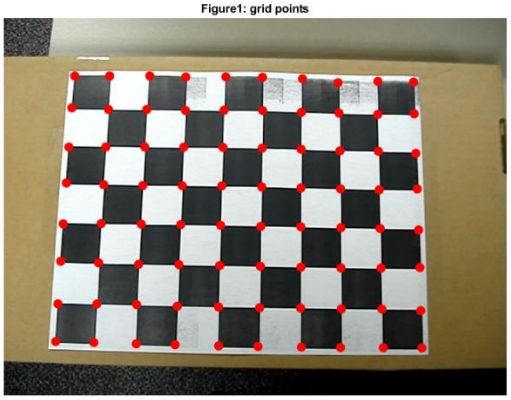


Figure1: grid points

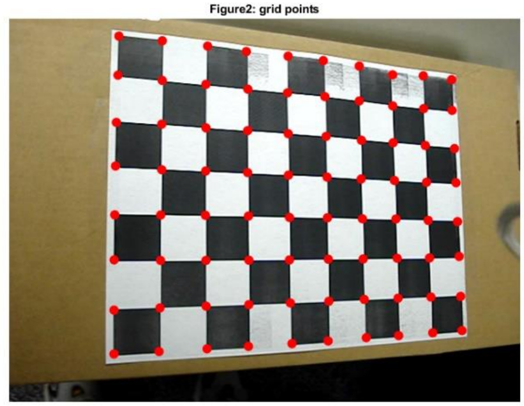


Figure2: grid points

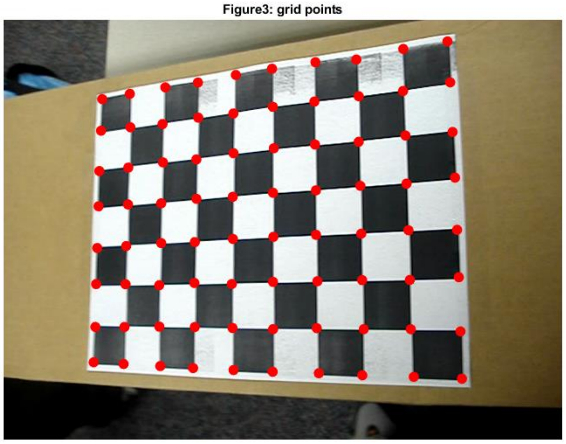


Figure3: grid points

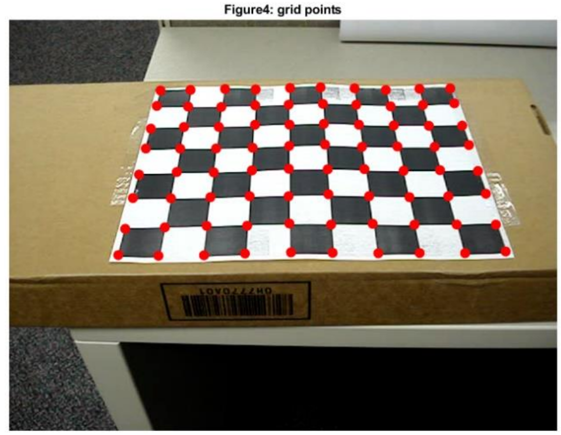


Figure4: grid points

2.1.4 Updated H, K matrices

```
images2.png, H is
   -0.9699    0.0873  -53.6111
   -0.0176   -0.8914  -43.0744
   -0.0000    0.0002   -0.6022
```

```
images9.png, H is
   -1.1014    0.0357  -70.6884
   -0.1500   -0.9475   -9.3528
   -0.0005    0.0001   -0.5191
```

```
images12.png, H is
   -0.7133    0.0517  -74.6938
    0.1782   -0.9021  -59.5219
    0.0005    0.0002   -0.6674
```

```
images20.png, H is
    0.8520   -0.2672  119.5679
   -0.0072    0.4013   55.3491
    0.0000   -0.0008    0.6767
```

Rishika Bhanushali
rb1182

```
image : images2.png

R =

   -0.9998    0.0142   -0.0043
   -0.0180   -0.9864    0.1640
   -0.0014    0.1640    0.9864


T =

  147.8365
  102.9665
 -446.8938
```

```
image : images9.png

R =

   -0.9243   -0.0094   -0.3815
   -0.0261   -0.9952    0.0933
   -0.3807    0.0970    0.9197


T =

   99.1289
  114.2336
 -373.3122
```

```
image : images12.png

R =

   -0.9153   -0.0063    0.4028
    0.0546   -0.9916    0.1158
    0.3991    0.1292    0.9079


T =

  148.3980
  101.9572
 -496.2113
```

```
image : images20.png

R =

    0.9999    0.0035    0.0044
   -0.0097    0.7141    0.7000
    0.0034   -0.7000    0.7141


T =

 -120.2781
 -124.0846
  574.6801

  721.3977   -0.6624  327.5222
         0  707.9265  234.6418
         0         0    1.0000
```

```
K_matrix =

  721.3977   -0.6624  327.5222
         0  707.9265  234.6418
         0         0    1.0000
```

2.1.5 Err_Reprojection

(initial average is before imporvement)

(the last average is after improvement)

Rishika Bhanushali
rb1182

```
image : images2.png
Sum err_reprojection:new corner-- approximate corner is=
  155.9689

Average err_reprojection:new corner-- approximate corner is=
    1.9496

Sum err_reprojection:new corner-- correct corner is=
  143.9755

Average err_reprojection:new corner-- correct corner is=
    1.7997
```

```
image : images9.png
Sum err_reprojection:new corner-- approximate corner is=
  156.8206

Average err_reprojection:new corner-- approximate corner is=
    1.9603

Sum err_reprojection:new corner-- correct corner is=
  149.1836

Average err_reprojection:new corner-- correct corner is=
    1.8648

image : images12.png
```

```
image : images12.png
Sum err_reprojection:new corner-- approximate corner is=
  184.9190

Average err_reprojection:new corner-- approximate corner is=
    2.3115

Sum err_reprojection:new corner-- correct corner is=
  166.4552

Average err_reprojection:new corner-- correct corner is=
    2.0807
```

```
image : images20.png
Sum err_reprojection:new corner-- approximate corner is=
   90.2832

Average err_reprojection:new corner-- approximate corner is=
    1.1285

Sum err_reprojection:new corner-- correct corner is=
  135.8244

Average err_reprojection:new corner-- correct corner is=
    1.6978
```

2.1.6 Can this be done automatically?

This can be done automatically using Harris corner detector. a corner can be interpreted as the junction of two edges, where an edge is a sudden change in image brightness. In our case, the points we are interested in are dark so out of the many points reported by harris detector, the number of points reduce. Now, we also know the approximate distance between the points, hence detection of the four cornens becomes even easier.

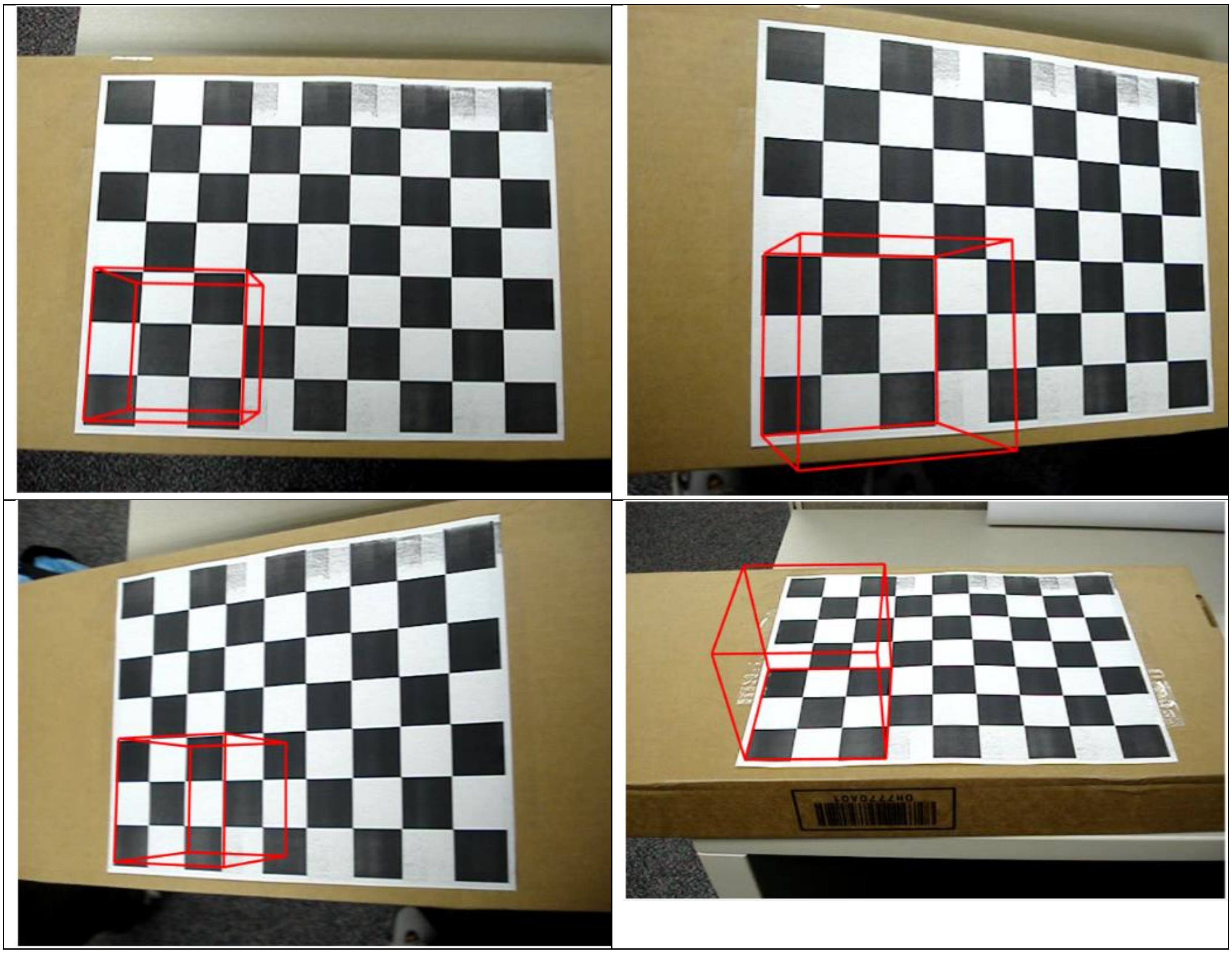
TASK 3 Augmented Reality 101

3.1 Outputs

Output: 3.1.1 Image Augmentation

My RUID is 204003359 → So, I used 9.jpg as my clipart image

Output: 3.1.2 Object Augmentation



References

1) https://github.com/sumitd-archives/Augmented_Reality
2) [Computer Vision: Algorithms and Applications] by Richard Szeliski

Rishika Bhanushali
rb1182

3) Calibration by Viewing a Plane from Unknown Orientations - Zhang, ICCV99

Directions to execute the code

Execute the below files to run the code.

The folder Structure of the training and testing images is as follows:

Assignment 3

```
├── ...
├──Part 1                                        # Folder
│   ├── part1.mlx                                # code-matlab file(main)
│   ├── Other support functions                  # matlab files
│   ├── Output

├──Part 2                                        # Folder
│   ├── part2.m                                  # code-matlab file(main)
│   ├── Other support functions                  # matlab files
│   ├── Output                                   # Folder
├──Part 3                                        # Folder
│   ├── part 3.mlx                               # code-matlab file(main)
│   ├── Other support functions                  # matlab files
│   ├── Output                                   # Folder
```