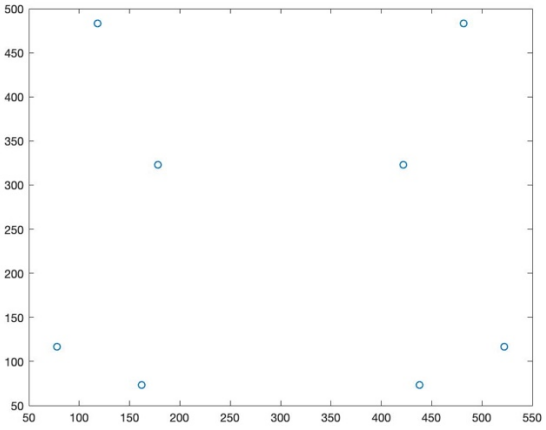
Assignment 3
Camera Calibration and Augmented Reality
bf289

- **Part1 Camera Calibration using 3D calibration object**



1.
```
function [P_matrix] = get_P_matrix(cube_points, image_points)
    P_matrix = [];
    [rows_n col_n] = size(cube_points);
    for i = 1 : rows_n
        object_point = cube_points(i , :);
        image_point = image_points(i , :);
        rows = GetRows(object_point , image_point);
        P_matrix = [P_matrix ; rows];
    end

    display(P_matrix);
end

function [rows] = GetRows(cube_points, image_points)
    %convert object_point to homogeneous coordinates
    cube_points = [cube_points 1];
    u = image_points(1);
    v = image_points(2);
    zeroes = [0 0 0 0];
    row1 = [cube_points zeroes cube_points.*(-u)];
    row2 = [zeroes cube_points cube_points.*(-v)];
    rows = [row1; row2];
```
2.
```
end
```

3.

```
P_matrix =

  Columns 1 through 9

         2         2         2         1         0         0         0         0      -844
         0         0         0         0         2         2         2         1      -646
        -2         2         2         1         0         0         0         0       356
         0         0         0         0        -2         2         2         1       646
        -2         2        -2         1         0         0         0         0       236
         0         0         0         0        -2         2        -2         1       966
         2         2        -2         1         0         0         0         0      -964
         0         0         0         0         2         2        -2         1      -966
         2        -2         2         1         0         0         0         0      -876
         0         0         0         0         2        -2         2         1      -146
        -2        -2         2         1         0         0         0         0       324
         0         0         0         0        -2        -2         2         1       146
        -2        -2        -2         1         0         0         0         0       156
         0         0         0         0        -2        -2        -2         1       234
         2        -2        -2         1         0         0         0         0     -1044
         0         0         0         0         2        -2        -2         1      -234

  Columns 10 through 12

      -844        -844        -422
      -646        -646        -323
      -356        -356        -178
      -646        -646        -323
      -236         236        -118
      -966         966        -483
      -964         964        -482
      -966         966        -483
       876        -876        -438
       146        -146         -73
       324        -324        -162
       146        -146         -73
       156         156         -78
       234         234        -117
      1044        1044        -522
       234         234        -117
```

4.

```
M_matrix =

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

5.

```
camera_center =

   -0.0000   -2.9912   -8.2695
```

6.

```
M_quote =

  734.6289  107.8955  299.9999
    0.0009  780.1442    0.2641
    0.0000    0.3597    1.0000
```

```
R_x =

    1.0000         0         0
         0    0.9410    0.3384
         0   -0.3384    0.9410


Theta_x =

  -19.7812


N_matrix =

  734.6289   -0.0000  318.8125
    0.0009  734.0199  264.2723
    0.0000         0    1.0627
```

7.

```
R_z =

    1.0000    0.0000         0
   -0.0000    1.0000         0
         0         0    1.0000


Theta_z =

  -7.2204e-05
```

8.

```
K_matrix =

  691.2797    0.0009  299.9999
    0.0000  690.7067  248.6780
    0.0000    0.0000    1.0000

focal_length_x = 691.2797
focal_length_y = 690.7067
image center = (u, v) = (299.9999, 248.678)
```

9.

- **Part2 Camera Calibration using 2D calibration object**
  1. Corner Extraction and Homography computation

```
images2.png, H is
    0.9655   -0.0835   54.2980
    0.0172    0.8809   43.4384
   -0.0000   -0.0002    0.6033

images9.png, H is
   -1.0919    0.0345  -71.6419
   -0.1477   -0.9309  -10.3829
   -0.0005    0.0001   -0.5191

images12.png, H is
    0.7037   -0.0492   75.9062
   -0.1758    0.8856   59.9260
   -0.0005   -0.0002    0.6658

images20.png, H is
   -0.8559    0.2712 -122.6998
    0.0113   -0.3995  -57.9033
    0.0000    0.0008   -0.6893
```

2. Computing the Intrinsic and Extrinsic parameters

B_matrix	B_matrix = −0.0000 0.0000 0.0005 0.0000 −0.0000 0.0004 0.0005 0.0004 −1.0000
v_0	v_0 = 230.3654
lambda	lambda = −0.7692
alpha	alpha = 723.3788
beta	beta = 702.7882
gamma	gamma = 0.6580
u_0	u_0 = 317.6932
A_matrix	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 = 1.0000 0.0061 −0.0000 0.0061 1.0000 −0.0000 −0.0000 −0.0000 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 = 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 =

    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 =

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

```
image : images2.png

R_modified =

    0.9998   -0.0180    0.0027
    0.0173    0.9864    0.1636
   -0.0056   -0.1635    0.9865


R_T_R =

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

```
image : images9.png

R_modified =

   -0.9241   -0.0134   -0.3818
   -0.0286   -0.9941    0.1042
   -0.3810    0.1072    0.9183


R_T_R =

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

```
image : images12.png

R_modified =

    0.9112    0.0030    0.4119
   -0.0549    0.9920    0.1141
   -0.4082   -0.1266    0.9041


R_T_R =

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

```
image : images20.png

R_modified =

   -0.9999    0.0041    0.0150
    0.0077   -0.7097    0.7044
    0.0135    0.7045    0.7096


R_T_R =

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

3. Improving accuracy

   Projected grid corners

Figure 1: Projected grid corners

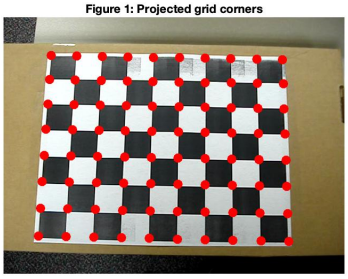


Figure 2: Projected grid corners

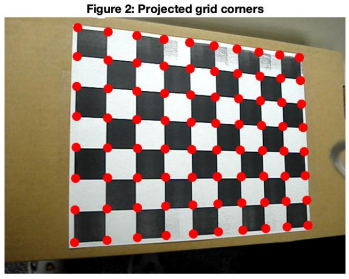


Figure 3: Projected grid corners

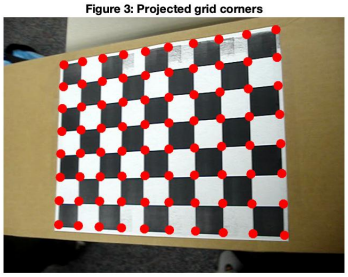
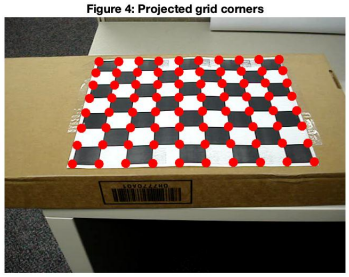


Figure 4: Projected grid corners



Harris corners

Figure 1: Harris corners

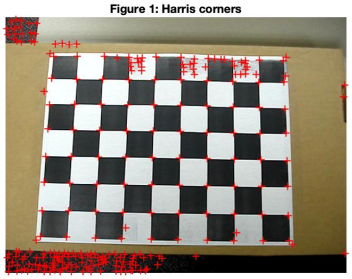


Figure 2: Harris corners

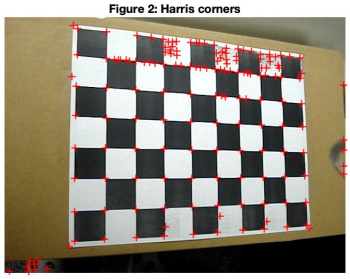


Figure 3: Harris corners

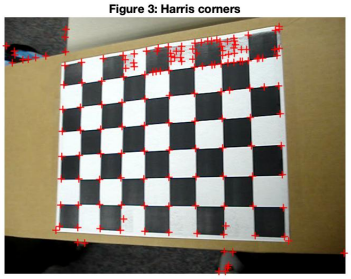
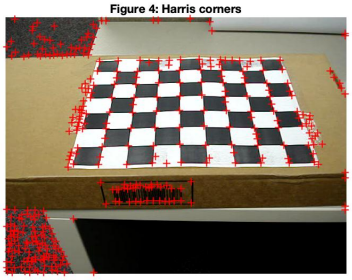


Figure 4: Harris corners



Closest harris corners

Figure1: grid points

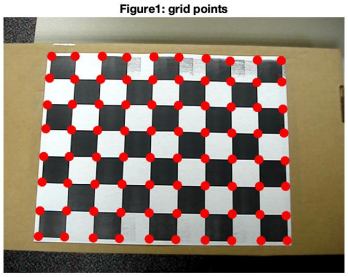


Figure2: grid points

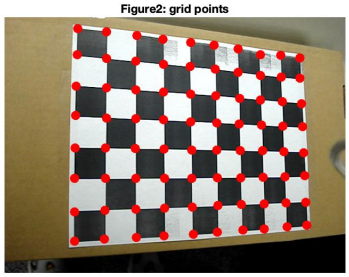


Figure3: grid points

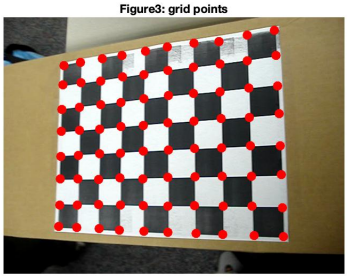
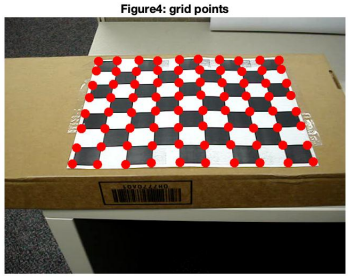


Figure4: grid points



H matrix

```
images2.png, H is
   -0.9608    0.0840  -54.6288
   -0.0172   -0.8786  -44.6587
   -0.0000    0.0002   -0.6014

images9.png, H is
    1.0827   -0.0321   71.7175
    0.1440    0.9321   11.6294
    0.0005   -0.0001    0.5200

images12.png, H is
   -0.7016    0.0488  -75.4923
    0.1760   -0.8856  -60.6018
    0.0005    0.0002   -0.6648

images20.png, H is
   -0.8627    0.2665 -121.5009
    0.0057   -0.4000  -56.9806
   -0.0000    0.0008   -0.6851
```

K matrix

```
K_matrix =

  735.3424    0.4787  324.3213
         0  717.0152  233.3936
         0         0    1.0000
```

R and T matrix

```
image : images2.png

R =

    0.0010   -0.0000   -0.0000
    0.0000    0.0010    0.0000
    0.0000   -0.0002    0.0000


T =

   -0.1468
   -0.1027
    0.4625
```

```
image : images9.png

R =

   -0.0010   -0.0000   -0.0000
   -0.0000   -0.0010    0.0000
   -0.0004    0.0001    0.0000


T =

    0.1013
    0.1177
   -0.3999
```

```
image : images12.png

R =

   1.0e-03 *

    0.9152    0.0063    0.0004
   -0.0549    0.9927    0.0001
   -0.4113   -0.1315    0.0009


T =

   -0.1465
   -0.1014
    0.5113
```

```
image : images20.png

R =

   1.0e-03 *

    0.8994    0.0003    0.0000
   -0.0082    0.6353    0.0006
    0.0065   -0.6336    0.0006


T =

   -0.1052
   -0.1104
    0.5268
```

Err_reprojection

```
image : images2.png
Sum err_reprojection between the new corner and approx corner is
  155.9689

Average err_reprojection between the new corner and approx corner is
    1.9496

Sum err_reprojection between the new corner and correct corner is
  143.9755

Average err_reprojection between the new corner and correct corner is
    1.7997
```

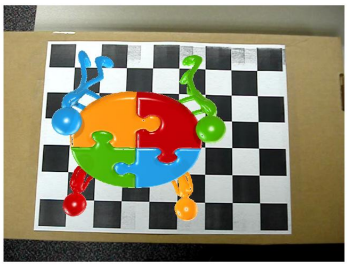
```
image : images9.png
Sum err_reprojection between the new corner and approx corner is
  156.8206

Average err_reprojection between the new corner and approx corner is
    1.9603

Sum err_reprojection between the new corner and correct corner is
  149.1836

Average err_reprojection between the new corner and correct corner is
    1.8648
```

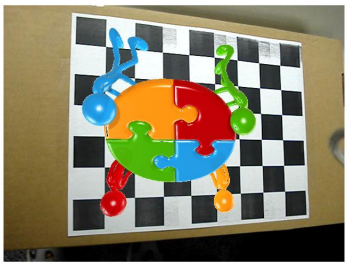
```
image : images12.png
Sum err_reprojection between the new corner and approx corner is
  184.9190

Average err_reprojection between the new corner and approx corner is
    2.3115

Sum err_reprojection between the new corner and correct corner is
  166.4552

Average err_reprojection between the new corner and correct corner is
    2.0807
```

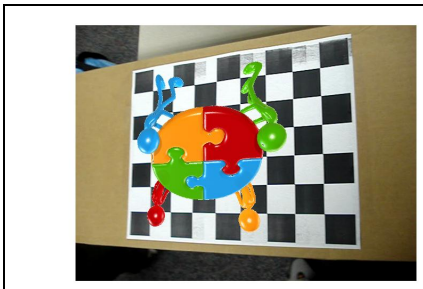
```
image : images20.png
Sum err_reprojection between the new corner and approx corner is
   90.2832

Average err_reprojection between the new corner and approx corner is
    1.1285

Sum err_reprojection between the new corner and correct corner is
  135.8244

Average err_reprojection between the new corner and correct corner is
    1.6978
```

- # Part3 Augmented Reality 101

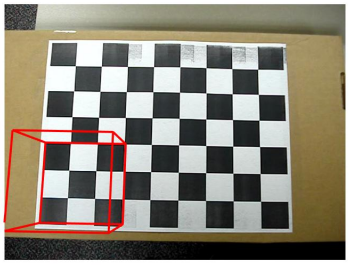
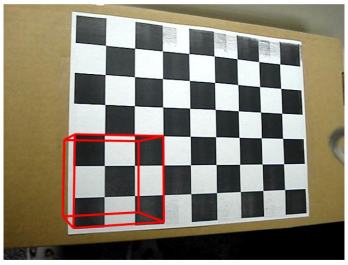
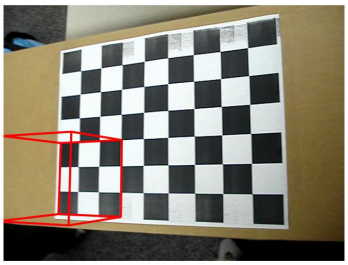
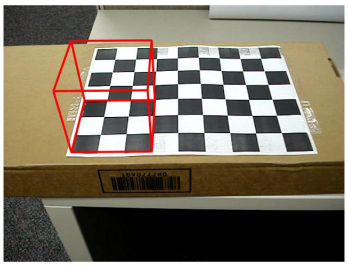
Augment an Image








Augment an Object

- ## Extra Credit

We could estimate the intrinsic and extrinsic parameters from only two images of the grid as Zhang described in the section 2.4.9:

Let (R_s, t_s) be the rigid transformation between the two cameras such that

$$(R', t') = (R, t) \circ (R_s, t_s)$$

or more precisely $R'=RR_s$ and $t'=Rt_s+t$.

Stereo calibration is then to solve A,A',k1,k2,k1',k2',$\{(R_i,t_i) \mid i = 1,\ldots,n\}$, and ($R_s$, t_s) by minimizing the following functions:

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\left[\delta_{ij}\left\|m_{ij}-\breve{m}(A,k_1,k_2,R_i,t_i,M_j)\right\|^2+\delta'_{ij}\left\|m'_{ij}-\breve{m}(A',k'_1,k'_2,R'_i,t'_i,M_j)\right\|^2\right]$$

subject to

$$R'_i = R_i R_s and t'_i = R_i t_s + t_i$$

Obviously, it is a nonlinear optimization problem. To obtain the initial guess, we first run single-camera calibration independently for each camera, and compute R_s through SVD from $R'_i = R_i R_s$ and t_s through least-squares from $t'_i = R_i t_s + t_i$.