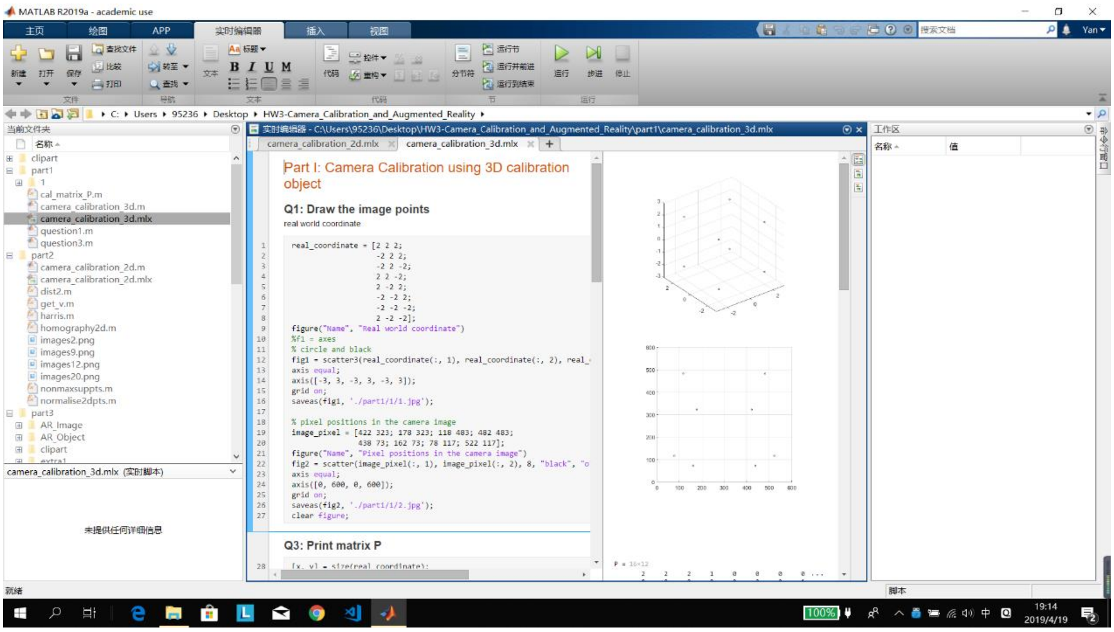
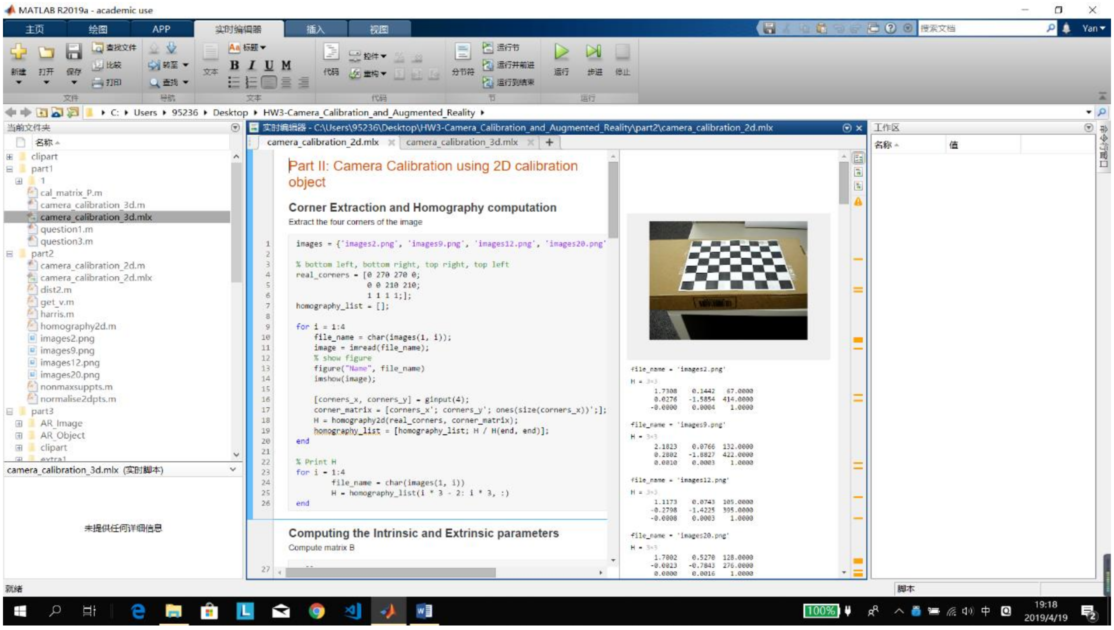
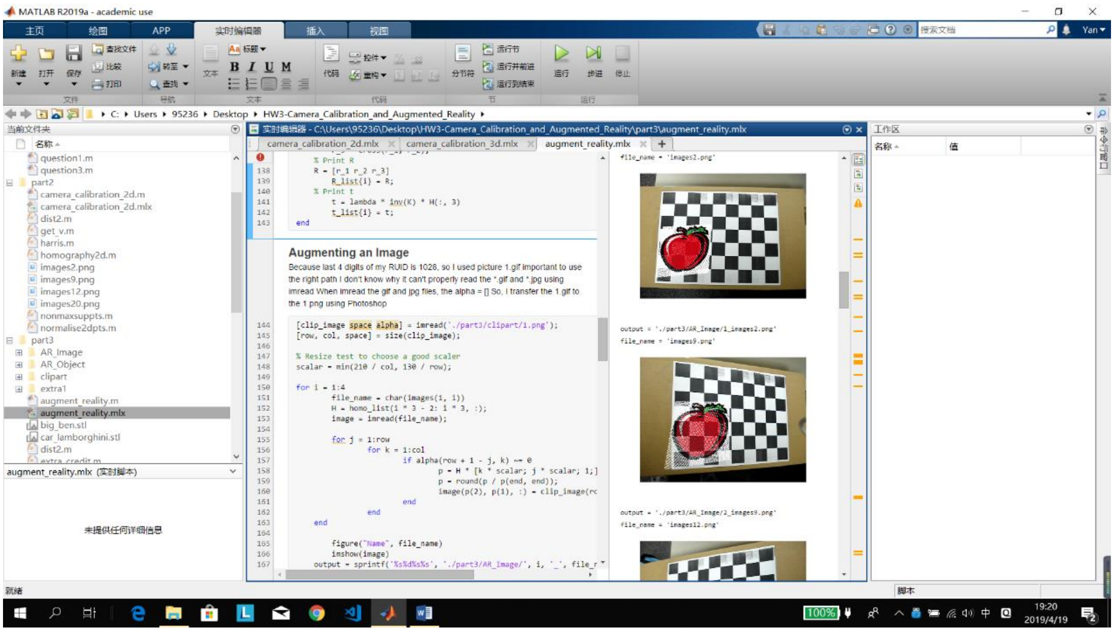
1 How to run

In part I, just run the camera_calibration_3d.m, the result Matlab publish file is camera_calibration_3d.mlx and it can be open in the matlab as **Live Script**.
In part I, just run the camera_calibration_2d.m, the result Matlab publish file is camera_calibration_2d.mlx and it can be open in the matlab as **Live Script**.
In part I, just run the augment_reality.m, the result Matlab publish file is augment_reality.mlx and it can be open in the matlab as **Live Script**.

All three **Live Scripts** will have views in Matlab like below.




MATLAB R2019a - academic use
Part II: Camera Calibration using 2D calibration object
Corner Extraction and Homography computation
Extract the four corners of the image
Computing the Intrinsic and Extrinsic parameters
Compute matrix B

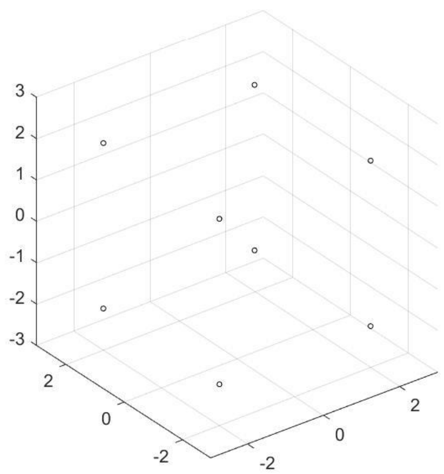

MATLAB R2019a - academic use
Augmenting an Image
Because last 4 digits of my RUID is 1026, so I used picture 1.gif important to use the right path I don't know why it can't properly read the *.gif and *.jpg using imread When imread the gif and jpg files, the alpha = [] So, I transfer the 1.gif to the 1.png using Photoshop

2 Part I: Camera Calibration using 3D calibration object

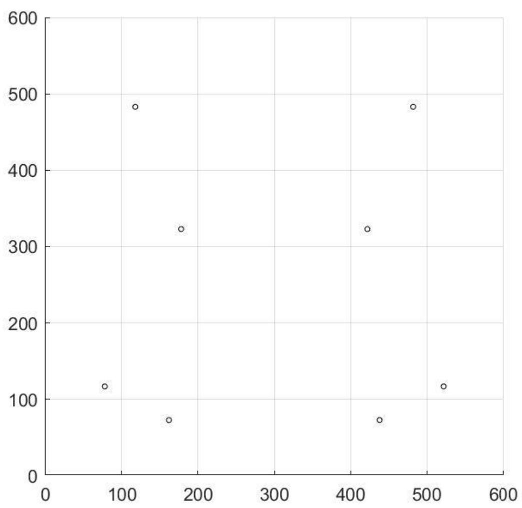
2.1 Q1: Draw the image points, using small circles for each image point.

I have written it in the part1/question1.m and also in the Q1 cell in camera_calibration_3d.m, the following are the results.

The real-world 3D coordinates.



The pixel positions in the camera image.



2.2 Q2, Q3, Q4, Q5 and Q6:

The function to generate matrix P is in cal_matrix_P.m
The following is the results.

Q3: Print matrix P

```
P = 16×12
     2     2     2     1     0     0     0     0  -844  -844  -844  -422
     0     0     0     0     2     2     2     1  -646  -646  -646  -323
    -2     2     2     1     0     0     0     0   356  -356  -356  -178
     0     0     0     0    -2     2     2     1   646  -646  -646  -323
    -2     2    -2     1     0     0     0     0   236  -236   236  -118
     0     0     0     0    -2     2    -2     1   966  -966   966  -483
     2     2    -2     1     0     0     0     0  -964  -964   964  -482
     0     0     0     0     2     2    -2     1  -966  -966   966  -483
     2    -2     2     1     0     0     0     0  -876   876  -876  -438
     0     0     0     0     2    -2     2     1  -146   146  -146   -73
    ⋮
```

Q4: Print matrix M

```
M = 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
```

Q5: Print the Euclidean coordinates of the camera center

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

Q6: Print matrix M'

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

2.3 Q7, Q8 and Q9:

The explanation is in the script.
For Q9,

```
K = [F_x s C_x;
     0 F_y C_y;
     0 C_y 1;]
Focal length in pixels = F
Optical center in pixels = C
```

The results are below.

Q7: Print R_x, theta_x, N

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

theta_x = -19.7812

N = 3×3
  734.6289   -0.0000  318.8125
    0.0009  734.0199  264.2723
    0.0000         0    1.0627
```

Q8: Calculate R_z and theta_z

n31 is small so no need for R_y rotation

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

theta_z = -7.2204e-05
```

Q9: Calculate K by M'/(R_x*R_y)

R_z is factorized out

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

fcoal_lengths = 1×2
  896.6817  896.1566

img_center = 1×2
  299.9999 -321.9619
```

3 Part II: Image Inpainting

3.1 Corner Extraction and Homography computation

When using function ginput, please click the image corners in the following order: Bottom left -> bottom right -> top right -> top left.

The results:

```
file_name = 'images2.png'
H = 3×3
    1.7308    0.1442   67.0000
    0.0276   -1.5854  414.0000
   -0.0000    0.0004    1.0000

file_name = 'images9.png'
H = 3×3
    2.1823    0.0766  132.0000
    0.2802   -1.8827  422.0000
    0.0010    0.0003    1.0000

file_name = 'images12.png'
H = 3×3
    1.1173    0.0743  105.0000
   -0.2798   -1.4225  395.0000
   -0.0008    0.0003    1.0000

file_name = 'images20.png'
H = 3×3
    1.7002    0.5270  128.0000
   -0.0023   -0.7843  276.0000
    0.0000    0.0016    1.0000
```

3.2 Computing the Intrinsic and Extrinsic parameters

Computing the Intrinsic and Extrinsic parameters

Compute matrix B

```
B = 3×3
   -0.0000    0.0000    0.0005
    0.0000   -0.0000    0.0003
    0.0005    0.0003   -1.0000


K = 3×3
  753.0497    0.7318  326.6904
         0  737.0727  220.5449
         0         0    1.0000


file_name = 'images2.png'
R = 3×3
    1.0007    0.0116    0.0008
    0.0169   -0.9858   -0.1675
   -0.0021    0.1674   -0.9868

t = 3×1
 -150.1320
  114.1799
  435.0308

R_T_R = 3×3
    1.0018   -0.0054    0.0000
   -0.0054    1.0000   -0.0000
    0.0000   -0.0000    1.0018

new_R = 3×3
    0.9999    0.0143    0.0008
    0.0143   -0.9858   -0.1674
   -0.0016    0.1674   -0.9859

new_R_T_R = 3×3
    1.0000    0.0000   -0.0000
    0.0000    1.0000    0.0000
   -0.0000    0.0000    1.0000


file_name = 'images9.png'
R = 3×3
    0.9285   -0.0116    0.3669
    0.0332   -0.9931   -0.1127
    0.3655    0.1168   -0.9216

t = 3×1
  -97.0781
  102.5232
  375.1063

R_T_R = 3×3
    0.9967   -0.0010    0.0000
   -0.0010    1.0000    0.0000
    0.0000    0.0000    0.9967

new_R = 3×3
    0.9300   -0.0111    0.3675
    0.0328   -0.9931   -0.1129
    0.3662    0.1170   -0.9232

new_R_T_R = 3×3
    1.0000    0.0000   -0.0000
    0.0000    1.0000    0.0000
   -0.0000    0.0000    1.0000

file_name = 'images12.png'
R = 3×3
    0.9110   -0.0094   -0.4209
   -0.0623   -0.9907   -0.1198
   -0.4163    0.1358   -0.9031

t = 3×1
 -145.0309
  116.5122
  492.2639

R_T_R = 3×3
    1.0072   -0.0034   -0.0000
   -0.0034    1.0000   -0.0000
   -0.0000   -0.0000    1.0071

new_R = 3×3
    0.9078   -0.0078   -0.4194
   -0.0637   -0.9908   -0.1194
   -0.4146    0.1351   -0.8999
```

```
new_R_T_R = 3×3
     1.0000   -0.0000   -0.0000
    -0.0000    1.0000    0.0000
    -0.0000    0.0000    1.0000

file_name = 'images20.png'
R = 3×3
     0.9998   -0.0013    0.0058
    -0.0057   -0.6905   -0.7231
     0.0143    0.7233   -0.6904

t = 3×1
  -117.5931
    33.5227
   445.5623

R_T_R = 3×3
     0.9998    0.0130   -0.0000
     0.0130    1.0000   -0.0000
    -0.0000   -0.0000    0.9996

new_R = 3×3
     1.0000   -0.0078    0.0058
    -0.0012   -0.6905   -0.7233
     0.0096    0.7232   -0.6905

new_R_T_R = 3×3
     1.0000    0.0000   -0.0000
     0.0000    1.0000    0.0000
    -0.0000    0.0000    1.0000
```

3.3 Improving accuracy

The figure 1 is painted using red circle, figure 2 is painted using yellow rhombus and figure 3 is painted using pinkish red cross.

The results are:

Improving accuracy

```
file_name = 'images2.png'
```

Figure 1 : Projected grid corners

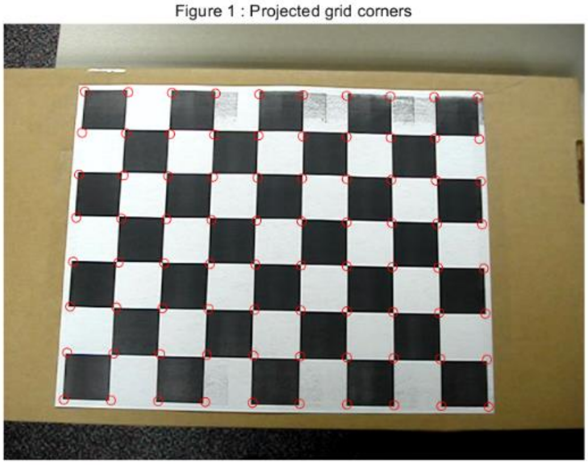


Figure 2 : Harris corners

Figure 3 : grid points

```
H = 3×3
      -0.9699   -0.0873  -35.2790
      -0.0176    0.8914 -230.2638
      -0.0000   -0.0002   -0.5556

err_reprojection = 0.2240

file_name = 'images9.png'
```

Figure 1 : Projected grid corners

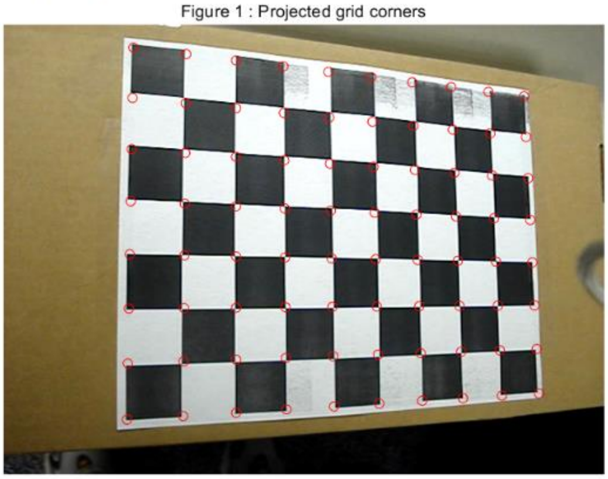
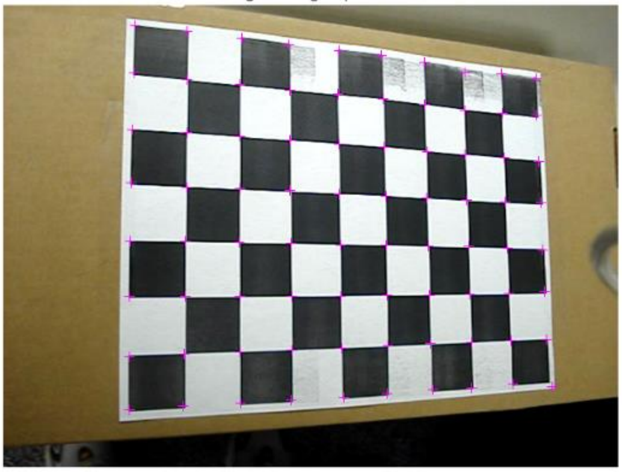


Figure 2 : Harris corners

Figure 3 : grid points



```
H = 3×3
    -1.0990   -0.0379  -63.1804
    -0.1489    0.9469 -208.3852
    -0.0005   -0.0001   -0.4909

err_reprojection = 0.2425
```

```
file_name = 'images12.png'
```

Figure 1 : Projected grid corners

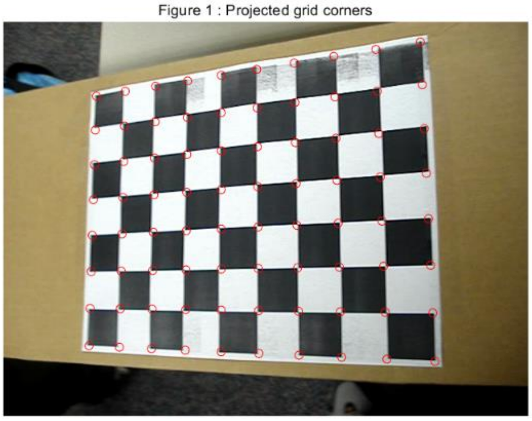
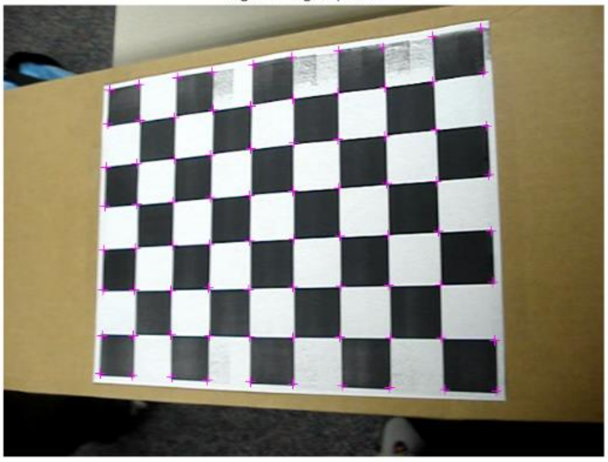


Figure 2 : Harris corners

Figure 3 : grid points



```
H = 3×3
      0.7133    0.0517   63.8469
     -0.1782   -0.9021  248.9692
     -0.0005    0.0002    0.6310

err_reprojection = 0.2570

file_name = 'images20.png'
```

Figure 1 : Projected grid corners

Figure 2 : Harris corners

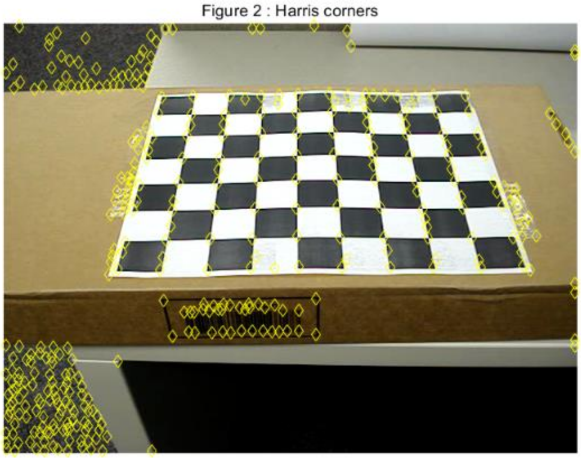


Figure 3 : grid points

```
H = 3×3
      0.8520    0.2672   63.4635
     -0.0072   -0.4013  139.6280
      0.0000    0.0008    0.5039

err_reprojection = 0.2116

B = 3×3
     -0.0000   -0.0000    0.0005
     -0.0000   -0.0000    0.0004
      0.0005    0.0004   -1.0000
```

```
K = 3×3
    725.8944   -1.7926  326.4675
           0  713.0109  232.8526
           0         0    1.0000


file_name = 'images2.png'
R = 3×3
      0.9958    0.0129    0.0043
      0.0179   -0.9862   -0.1645
      0.0014    0.1653   -0.9822

t = 3×1
     80.8809
    -11.6984
   -368.3006

file_name = 'images9.png'
R = 3×3
      0.9258   -0.0108    0.3789
      0.0277   -0.9947   -0.0991
      0.3780    0.1026   -0.9206

t = 3×1
     69.3634
    -10.0325
   -315.8543

file_name = 'images12.png'
R = 3×3
      0.9182   -0.0076   -0.4070
     -0.0558   -0.9915   -0.1164
     -0.4032    0.1302   -0.9108

t = 3×1
     92.3834
    -13.3620
   -420.6784

file_name = 'images20.png'
R = 3×3
      1.0014   -0.0036   -0.0044
     -0.0097   -0.7107   -0.7045
      0.0034    0.7035   -0.7118


t = 3×1
     84.0571
    -12.1578
   -382.7638
```

3.4 Can you suggest a way this can be done automatically (i.e without first letting the user manually select the 4 corners)?

From Harris Corner detector, we can get many corners including the ones we really want. First the pixel of the corner we want is dark, it's filters out many times. And the measure of the grid is bigger than the noises. Second, we already know the distance of these 4 points. Although the image is distorted, we can get use an approximate distance. Then we can use this approximate of grid distance to find the proper grid corners. From these grid corners, we get the four outward corners as the corner we want.

4 Part III: Augmented Reality 101

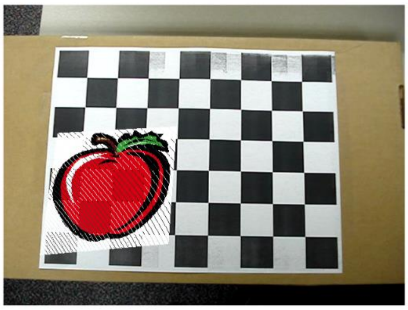
4.1 Augmenting an Image

Because last 4 digits of my RUID is 1028, so I used picture 1.png
The results is in AR_Image.

Augmenting an Image

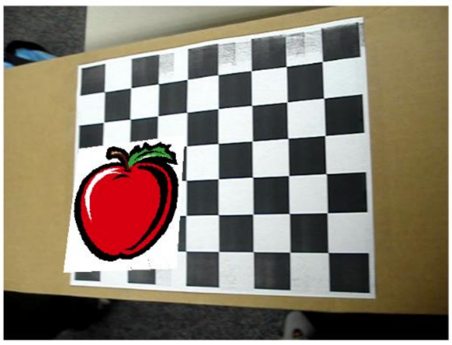
Because last 4 digits of my RUID is 1028, so I used picture 1.gif important to use the right path I don't know why it can't properly read the *.gif and *.jpg using imread When imread the gif and jpg files, the alpha = [] So, I transfer the 1.gif to the 1.png using Photoshop

```
file_name = 'images2.png'
```



```
output = './part3/AR_Image/1_images2.png'
file_name = 'images9.png'
```

```
output = './part3/AR_Image/2_images9.png'
file_name = 'images12.png'
```



```
output = './part3/AR_Image/3_images12.png'
file_name = 'images20.png'
```

```
output = './part3/AR_Image/4_images20.png'
```

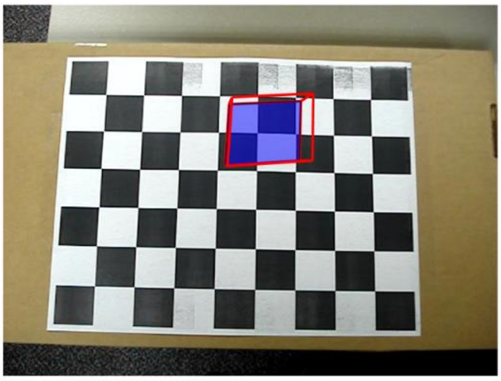
4.2 Augmenting an Object

The results is in AR_Object.

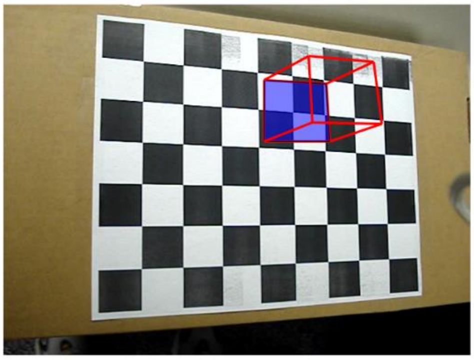
Augment Reality Object

Put object in the middle of the picture the object is 60 px * 60 px * 60 px

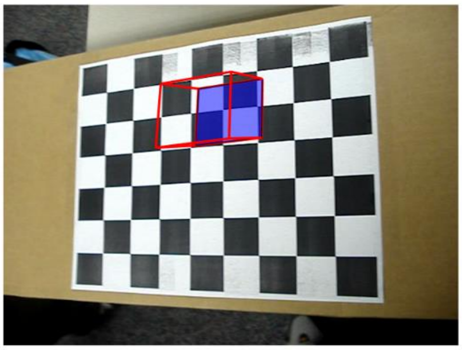
```
file_name = 'images2.png'
f1 =
  Figure (30: images2.png) - 属性:

      Number: 30
        Name: 'images2.png'
       Color: [0.9400 0.9400 0.9400]
    Position: [438.4000 311.4000 659.2000 474.4000]
       Units: 'pixels'

  显示 所有属性
output = './part3/AR_Object/1_images2.png'
```



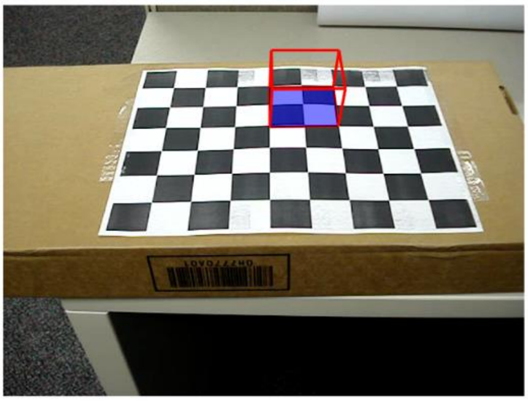
```
file_name = 'images9.png'
f1 =
  Figure (31: images9.png) - 属性:

      Number: 31
        Name: 'images9.png'
       Color: [0.9400 0.9400 0.9400]
    Position: [438.4000 311.4000 659.2000 474.4000]
       Units: 'pixels'

  显示 所有属性
output = './part3/AR_Object/2_images9.png'
```



```
file_name = 'images12.png'
f1 =
  Figure (32: images12.png) - 属性:

      Number: 32
        Name: 'images12.png'
       Color: [0.9400 0.9400 0.9400]
    Position: [438.4000 311.4000 659.2000 474.4000]
       Units: 'pixels'

  显示 所有属性
output = './part3/AR_Object/3_images12.png'
```



```
file_name = 'images20.png'
f1 =
  Figure (33: images20.png) - 属性:

      Number: 33
        Name: 'images20.png'
       Color: [0.9400 0.9400 0.9400]
    Position: [438.4000 311.4000 659.2000 474.4000]
       Units: 'pixels'

  显示 所有属性
output = './part3/AR_Object/4_images20.png'
```



5 Extra credit

5.1 Instead of augmenting a cube, augment a general mesh from a 3D file of your choice.

I choose a Lamborghini model because it is clear to see the difference and I download a script stlread.m to read the 3 dimension coordinates from the 3D **stl** file.
When read a stl file, we can get face-vertex normal vectors and then we can change it to xyz coordinates. Refer to
https://www.mathworks.com/help/matlab/ref/patch.html?searchHighlight=patch&s_tid=doc_srchtitle#burynyu-8

Extra Credit

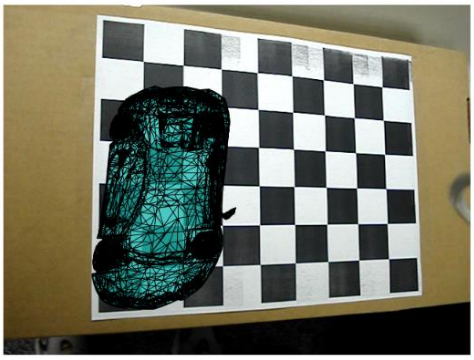
1: augment a general mesh from a 3D file.

*.stl can be opened using 3D printer or other Windows 10 default software. I choose stl file because we can easily get face-vertex vectors, and change them to xyz coordinates. Then use patch to draw a 3D model on our images.

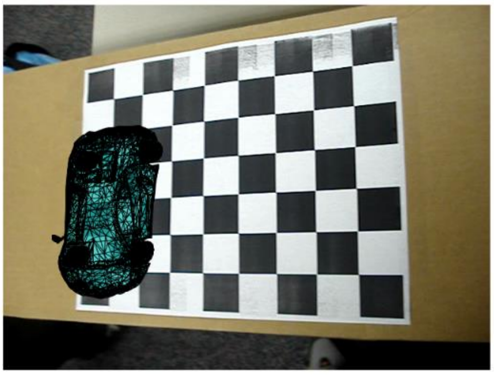
```
Title:
Num Facets: 32890
```

```
m = 3
n = 32890

file_name = 'images2.png'
f1 =
  Figure (34: images2.png) - 属性:

      Number: 34
        Name: 'images2.png'
       Color: [0.9400 0.9400 0.9400]
    Position: [438.4000 311.4000 659.2000 474.4000]
       Units: 'pixels'

  显示 所有属性
output = './part3/extra1/1car_lamborghiniimages2.png'
```

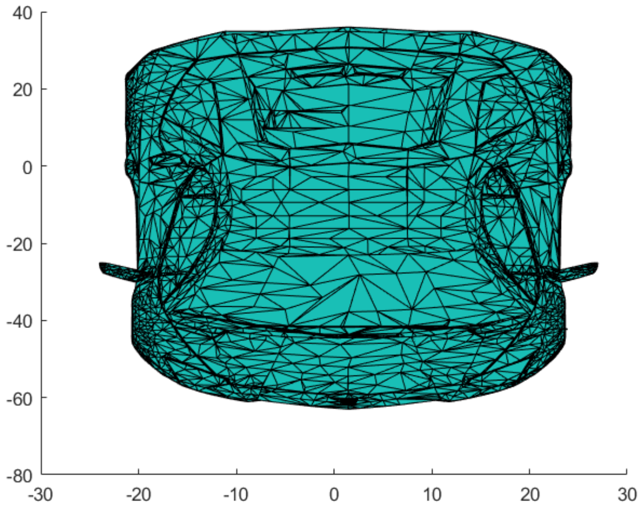
```
file_name = 'images9.png'
f1 =
  Figure (35: images9.png) - 属性:

      Number: 35
        Name: 'images9.png'
       Color: [0.9400 0.9400 0.9400]
    Position: [438.4000 311.4000 659.2000 474.4000]
       Units: 'pixels'

  显示 所有属性
output = './part3/extra1/2car_lamborghiniimages9.png'
```



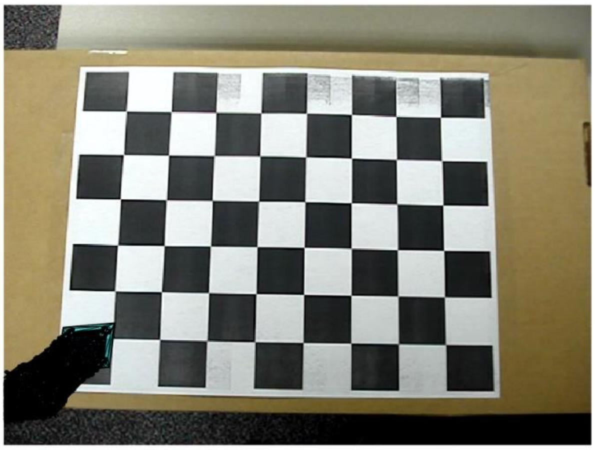
```
file_name = 'images12.png'
f1 =
  Figure (36: images12.png) - 属性:

      Number: 36
        Name: 'images12.png'
       Color: [0.9400 0.9400 0.9400]
    Position: [438.4000 311.4000 659.2000 474.4000]
       Units: 'pixels'

  显示 所有属性
output = './part3/extra1/3car_lamborghiniimages12.png'
```

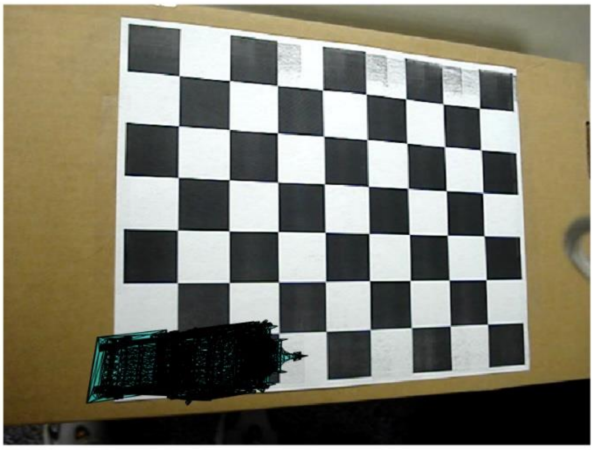


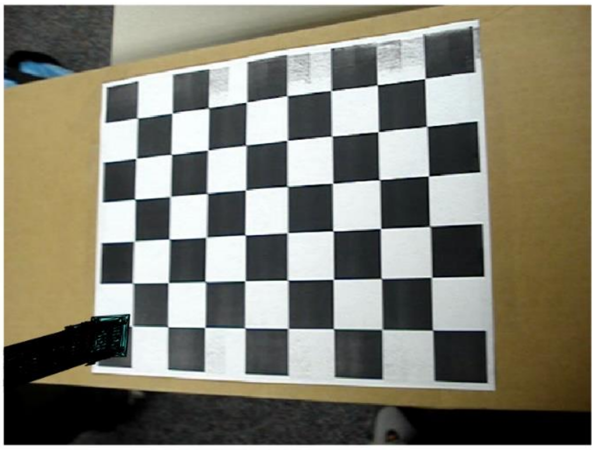
```
file_name = 'images20.png'
f1 =
  Figure (37: images20.png) - 属性:

      Number: 37
        Name: 'images20.png'
       Color: [0.9400 0.9400 0.9400]
    Position: [438.4000 311.4000 659.2000 474.4000]
       Units: 'pixels'

  显示 所有属性
output = './part3/extra1/4car_lamborghiniimages20.png'
```

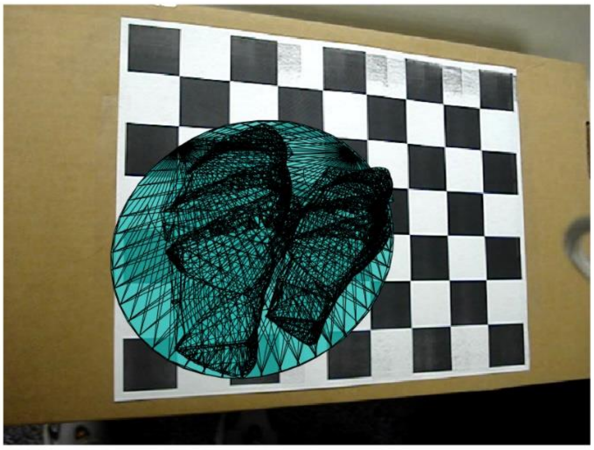


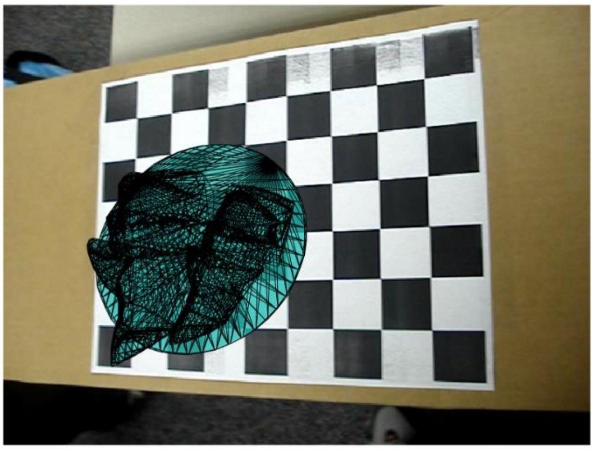
I also run other model as showing below: the Big Ben and the Sydney Opera House. (In extra1)











5.2 Can you find a way to estimate the intrinsic and extrinsic parameters from only two images of the grid.

Estimate from only two images from the chapter given. If n = 2, we can impose the skewless constraint = 0, i.e., [0, 1, 0, 0, 0, 0]b = 0. That means we assume the pixels construct a rectangle.
The function is in the extra_credit.m and I just use image2 and image9 for calculating.
The results are in the next page.

```
B = 3×3
    0.0000         0   -0.0005
         0    0.0000    0.0001
   -0.0005    0.0001    1.0000

K = 3×3
  833.3874         0  488.9995
         0  659.2114  -37.9520
         0         0    1.0000

file_name = 'images2.png'
R = 3×3
    0.0016   -0.0000    0.0000
   -0.0000   -0.0015   -0.0000
    0.0002    0.0005   -0.0000

t = 3×1
   -0.3840
    0.4942
    0.7421

file_name = 'images9.png'
R = 3×3
    0.0018   -0.0002    0.0000
    0.0004   -0.0019   -0.0000
    0.0012    0.0011   -0.0000

t = 3×1
   -0.3535
    0.5305
    0.7421
```