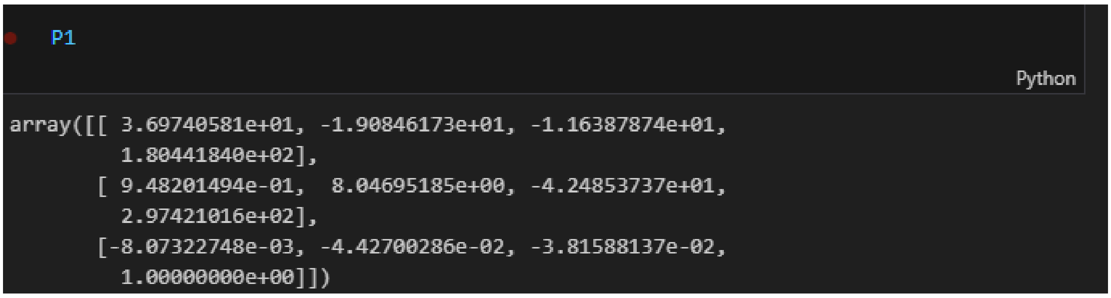
HW03

Camera Calibration

313581038 智能系統碩一 蒲品憶

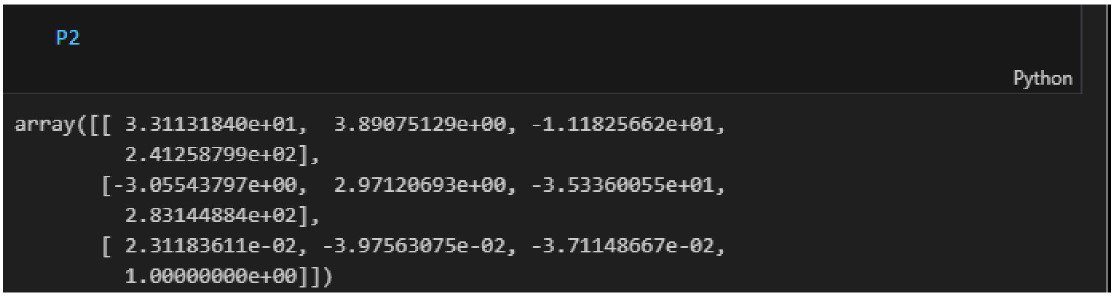
A. Compute the projection matrix from a set of 2D-3D point correspondences by using the least-squares (eigenvector) method for each image

根據 2D 和 3D 的點對應關係計算投影矩陣，可以使用最小二乘法解決這個問題。通過在投影矩陣中引入這些點，能夠估算投影矩陣。



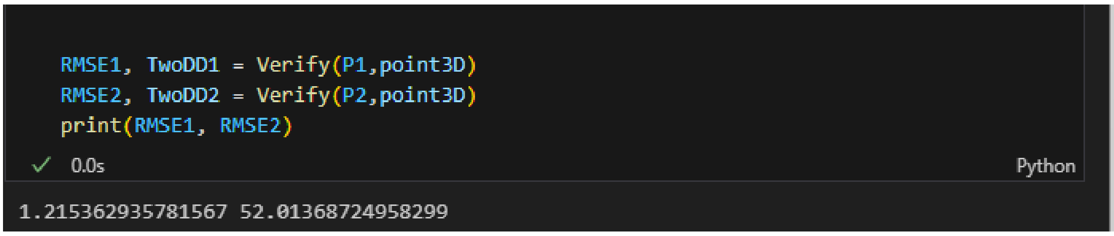
```
P1
```

```
array([[ 3.69740581e+01, -1.90846173e+01, -1.16387874e+01,
         1.80441840e+02],
       [ 9.48201494e-01,  8.04695185e+00, -4.24853737e+01,
         2.97421016e+02],
       [-8.07322748e-03, -4.42700286e-02, -3.81588137e-02,
         1.00000000e+00]])
```



```
P2
```

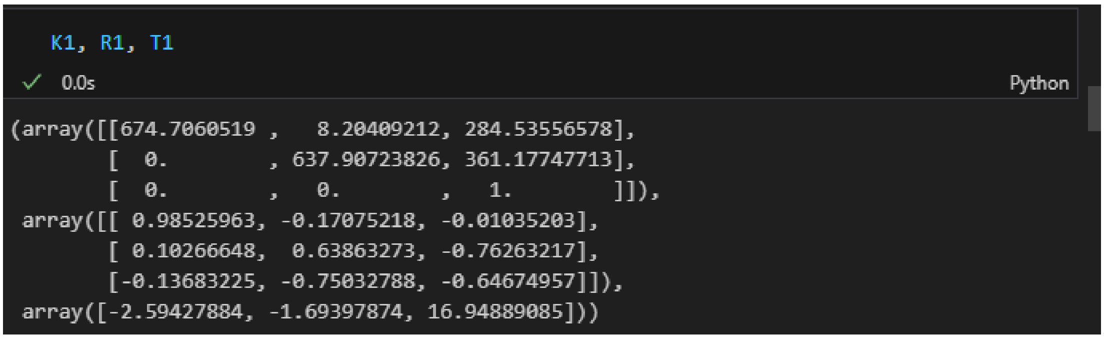
```
array([[ 3.31131840e+01,  3.89075129e+00, -1.11825662e+01,
         2.41258799e+02],
       [-3.05543797e+00,  2.97120693e+00, -3.53360055e+01,
         2.83144884e+02],
       [ 2.31183611e-02, -3.97563075e-02, -3.71148667e-02,
         1.00000000e+00]])
```



```
RMSE1, TwoDD1 = Verify(P1,point3D)
RMSE2, TwoDD2 = Verify(P2,point3D)
print(RMSE1, RMSE2)
```

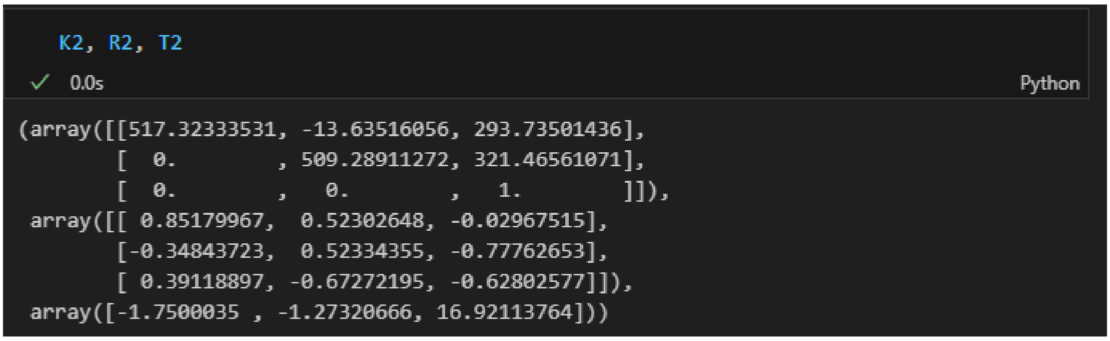
```
1.215362935781567 52.01368724958299
```

B. Decompose the two computed projection matrices from (A) into the camera intrinsic matrices K, rotation matrices R and translation vectors t by using the Gram-Schmidt process. Any QR decomposition functions are allowed. The bottom right corner of intrinsic matrix K should be normalized to 1. Also, the focal length in K should be positive



```
K1, R1, T1
```

```
(array([[674.7060519 ,   8.20409212, 284.53556578],
        [  0.        , 637.90723826, 361.17747713],
        [  0.        ,   0.        ,   1.        ]]),
 array([[ 0.98525963, -0.17075218, -0.01035203],
        [ 0.10266648,  0.63863273, -0.76263217],
        [-0.13683225, -0.75032788, -0.64674957]]),
 array([-2.59427884, -1.69397874, 16.94889085]))
```

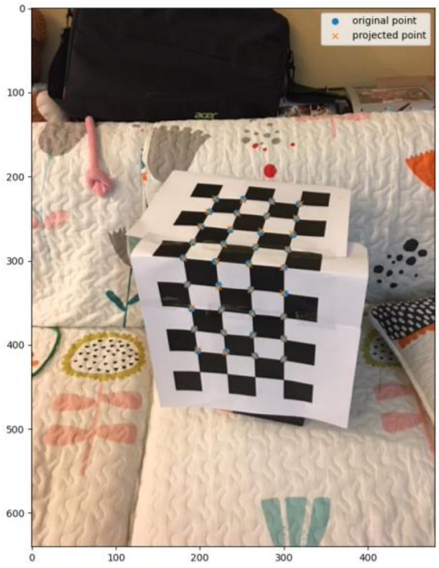


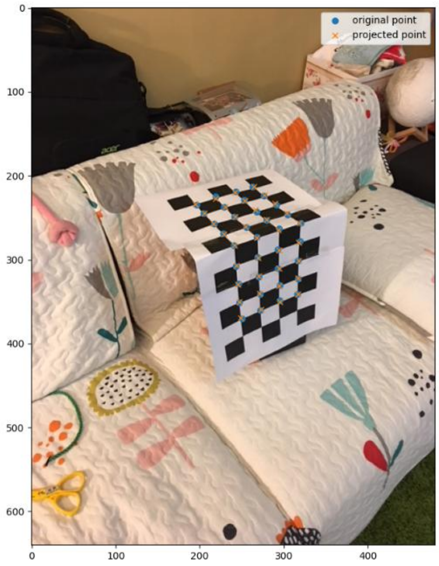
```
K2, R2, T2
```

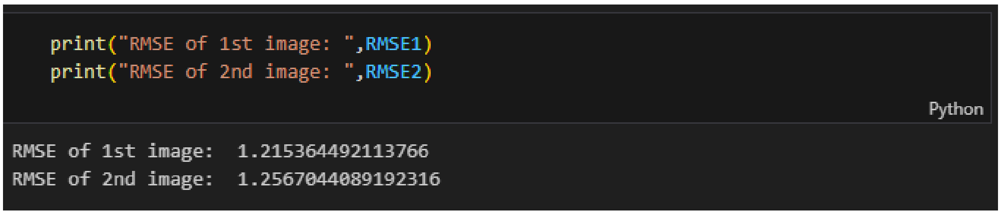
✓ 0.0s Python

```
(array([[517.32333531, -13.63516056, 293.73501436],
        [  0.        , 509.28911272, 321.46561071],
        [  0.        ,   0.        ,   1.        ]]),
 array([[ 0.85179967,  0.52302648, -0.02967515],
        [-0.34843723,  0.52334355, -0.77762653],
        [ 0.39118897, -0.67272195, -0.62802577]]),
 array([-1.7500035 , -1.27320666, 16.92113764]))
```

C. Re-project 2D points on each of the chessboard images by using the computed intrinsic matrix, rotation matrix and translation vector. Show the results (2 images) and compute the point re-projection root-mean-squared errors.





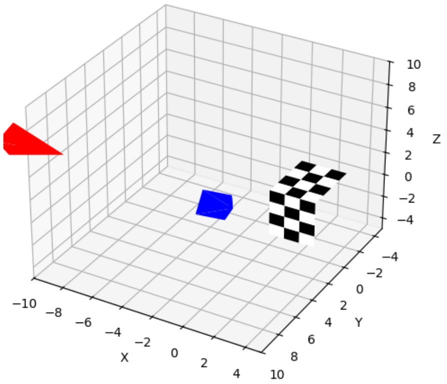


```
print("RMSE of 1st image: ",RMSE1)
print("RMSE of 2nd image: ",RMSE2)
```

Python

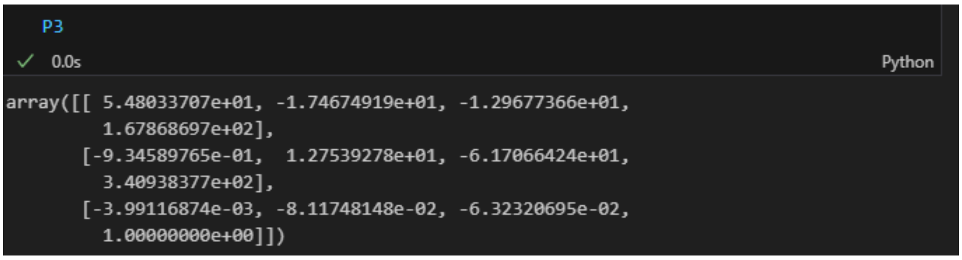
```
RMSE of 1st image:  1.215364492113766
RMSE of 2nd image:  1.2567044089192316
```

D. Plot camera poses for the computed extrinsic parameters (R, t) and then compute the angle between the two camera pose vectors.



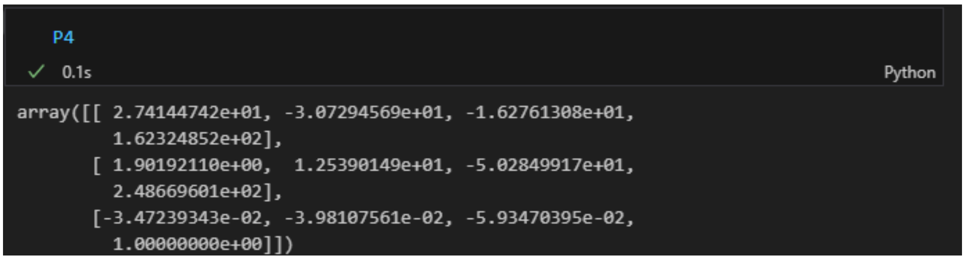
E. Print out two “chessboard.png” in the attached file and paste them on a box. Take two pictures from different angles. For each image, perform the steps above (A ~ D)

(a)



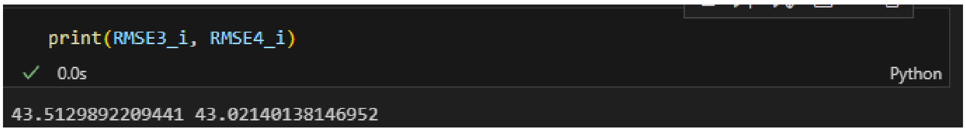
```
P3
```

```
array([[ 5.48033707e+01, -1.74674919e+01, -1.29677366e+01,
         1.67868697e+02],
       [-9.34589765e-01,  1.27539278e+01, -6.17066424e+01,
         3.40938377e+02],
       [-3.99116874e-03, -8.11748148e-02, -6.32320695e-02,
         1.00000000e+00]])
```



```
P4
```

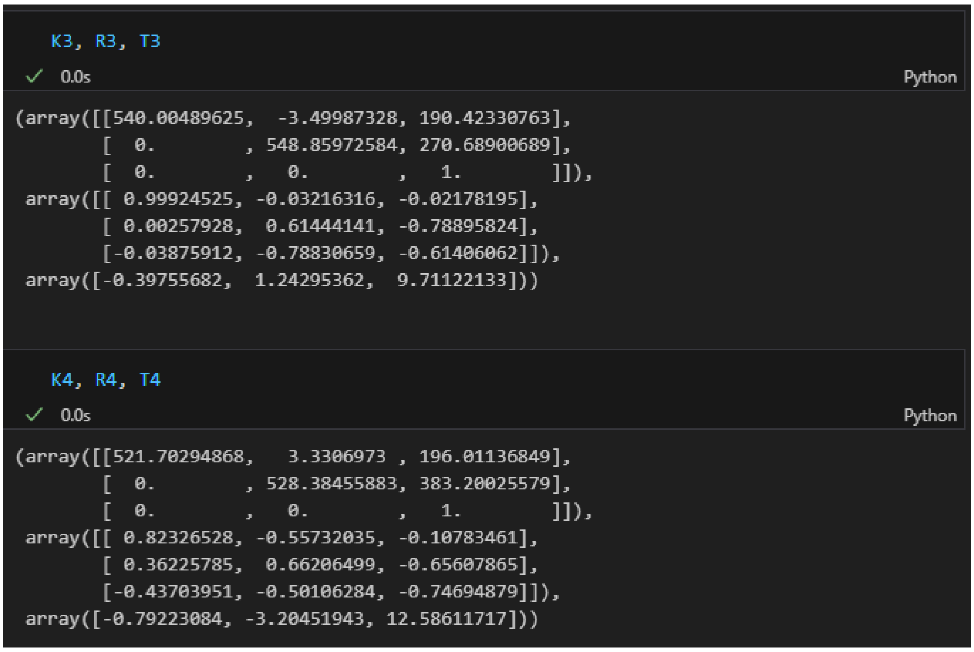
```
array([[ 2.74144742e+01, -3.07294569e+01, -1.62761308e+01,
         1.62324852e+02],
       [ 1.90192110e+00,  1.25390149e+01, -5.02849917e+01,
         2.48669601e+02],
       [-3.47239343e-02, -3.98107561e-02, -5.93470395e-02,
         1.00000000e+00]])
```



```
print(RMSE3_i, RMSE4_i)
```

```
43.5129892209441 43.02140138146952
```

(b)



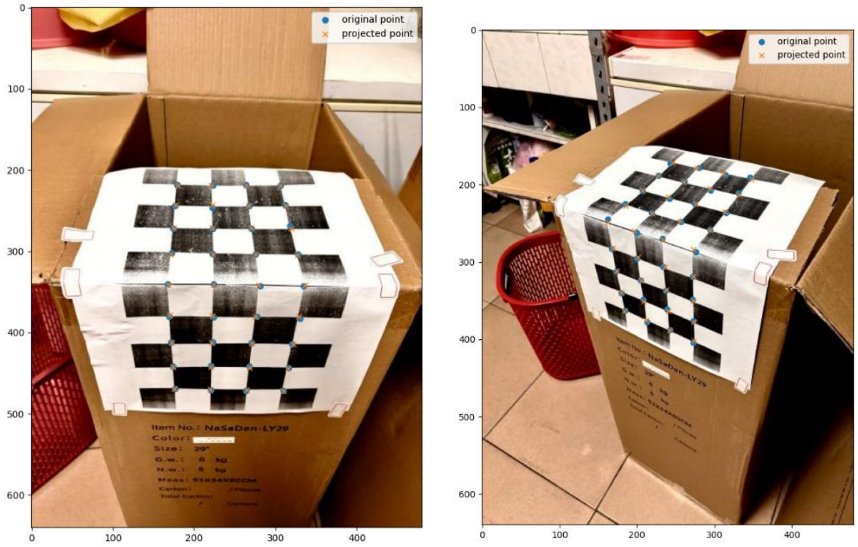
```
K3, R3, T3
```

```
(array([[540.00489625,  -3.49987328, 190.42330763],
        [  0.        , 548.85972584, 270.68900689],
        [  0.        ,   0.        ,   1.        ]]),
 array([[ 0.99924525, -0.03216316, -0.02178195],
        [ 0.00257928,  0.61444141, -0.78895824],
        [-0.03875912, -0.78830659, -0.61406062]]),
 array([-0.39755682,  1.24295362,  9.71122133]))
```

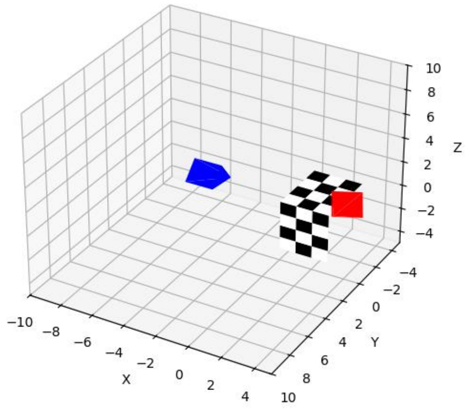
```
K4, R4, T4
```

```
(array([[521.70294868,   3.3306973 , 196.01136849],
        [  0.        , 528.38455883, 383.20025579],
        [  0.        ,   0.        ,   1.        ]]),
 array([[ 0.82326528, -0.55732035, -0.10783461],
        [ 0.36225785,  0.66206499, -0.65607865],
        [-0.43703951, -0.50106284, -0.74694879]]),
 array([-0.79223084, -3.20451943, 12.58611717]))
```

(c)

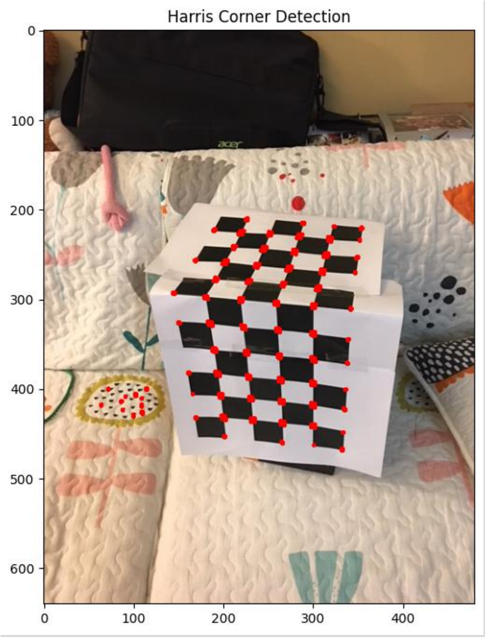


(d)



F. Instead of mark the 2D points by hand, you can find the 2D points in your images automatically by using corner detection, hough transform, etc.

從影像中自動提取 2D 點，而不再手動標註。這將大大提高處理效率，尤其是在圖像較多或不規則的情況下，能夠有效提高準確度和自動化程度。 我使用的方法是 cornerHarris。


Harris Corner Detection