CS655000 Computer Vision Homework 2

108062586 楊子儀

Part 1. Camera Calibration

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

Chessboard 1	[[-1.20998210e-01 2.43149939e-02 3.70139071e-02 -2.80793151e-01] [6.73781113e-03 -2.00491352e-02 1.46634752e-01 -9.39465030e-01] [-1.65686547e-05 2.01612051e-04 1.76893633e-04 -3.10810712e-03]]
Chessboard 2	[[-1.11019282e-01 -3.84602121e-02 3.00658422e-02 -5.07214623e-01] [1.10579817e-02 -1.30618413e-02 1.32845786e-01 -8.42660119e-01] [-1.07631674e-04 1.49557766e-04 1.37512268e-04 -2.80725147e-03]]

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.

output.txt - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

```
************** pic 1 *****************
# --------- begin compute the matrix P --------- #
 ------------------ projection matrix ------------------
[[-1.20998210e-01  2.43149939e-02  3.70139071e-02 -2.80793151e-01]
 [ 6.73781113e-03 -2.00491352e-02  1.46634752e-01 -9.39465030e-01]
 [-1.65686547e-05  2.01612051e-04  1.76893633e-04 -3.10810712e-03]]
# --------- begin decompose P --------- #
 ------------------ K ------------------
[[441.79419439   1.97692207 186.31607276]
 [  0.         461.45465546 301.67533168]
 [  0.           0.           1.        ]]
 ------------------ R ------------------
[[-0.99360011 -0.10867566  0.0307965 ]
 [ 0.09464302 -0.65215767  0.7521523 ]
 [-0.06165647  0.75025328  0.65826932]]
 ------------------ t ------------------
[  2.51264954  -0.01472351 -11.56611198]
# --------- begin Reprojection --------- #
chessboard_1 error : 0.7071067811865476
************** pic 2 *****************
# --------- begin compute the matrix P --------- #
 ------------------ projection matrix ------------------
[[-1.11019282e-01 -3.84602121e-02  3.00658422e-02 -5.07214623e-01]
 [ 1.10579817e-02 -1.30618413e-02  1.32845786e-01 -8.42660119e-01]
 [-1.07631674e-04  1.49557766e-04  1.37512268e-04 -2.80725147e-03]]
# --------- begin decompose P --------- #
 ------------------ K ------------------
[[489.80440977  11.66276711 195.44556311]
 [  0.         507.47836703 286.10939159]
 [  0.           0.           1.        ]]
 ------------------ R ------------------
[[-0.80757977 -0.58968593  0.00924243]
 [ 0.35870077 -0.4786834   0.8013713 ]
 [-0.46813318  0.65048652  0.59809583]]
 ------------------ t ------------------
[  0.37612932  -0.33836009 -12.20985892]
# --------- begin Reprojection --------- #
chessboard_2 error : 0.9204467514322717
************** finish *****************
```

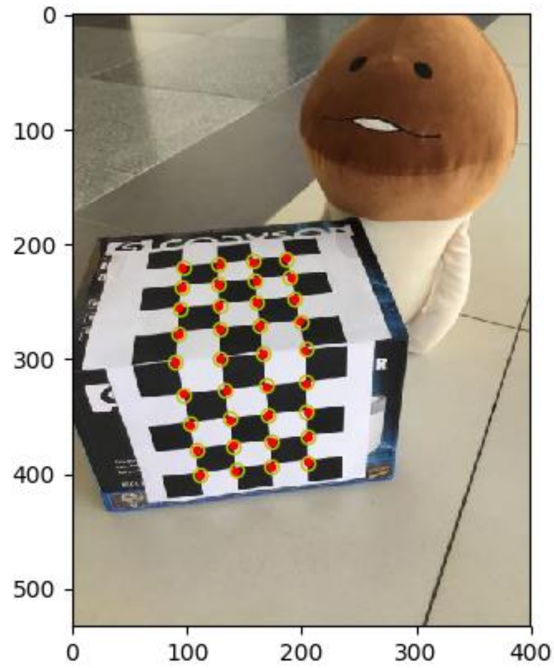
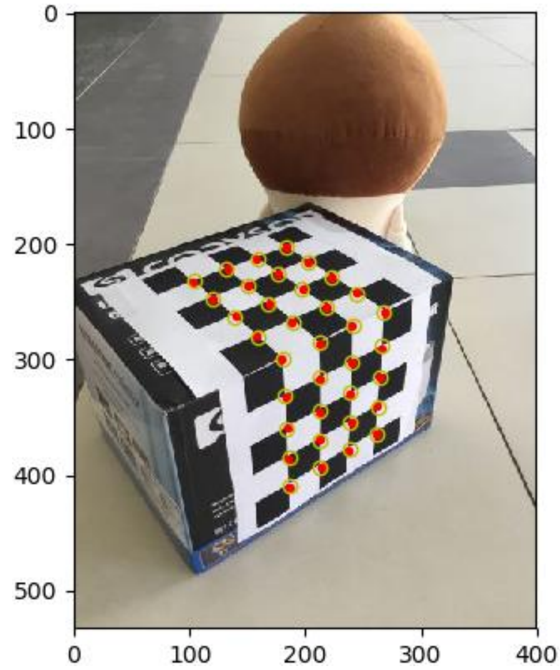
Chessboard 1 (K, R, t of pic 1)

Chessboard 2 (K, R, t of pic 2)

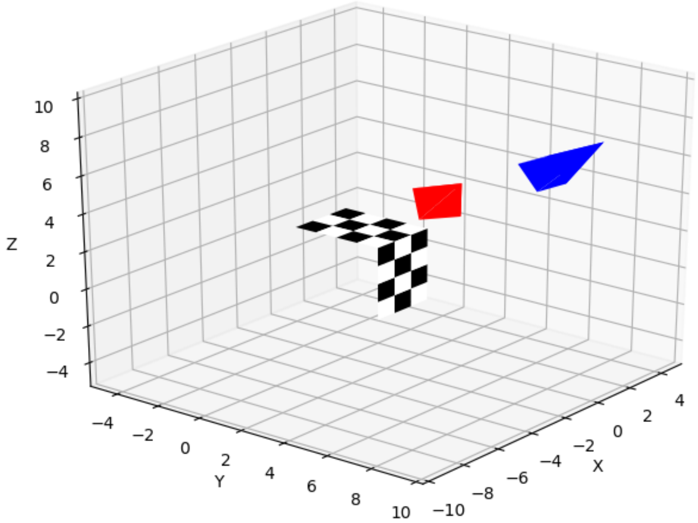
The decompose results show in the output.txt.

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

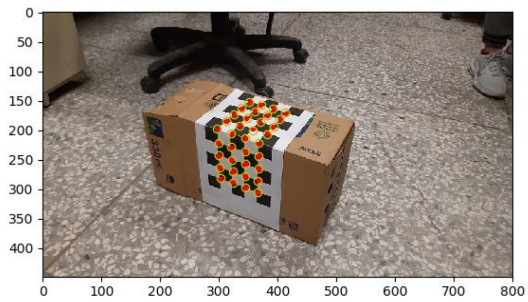
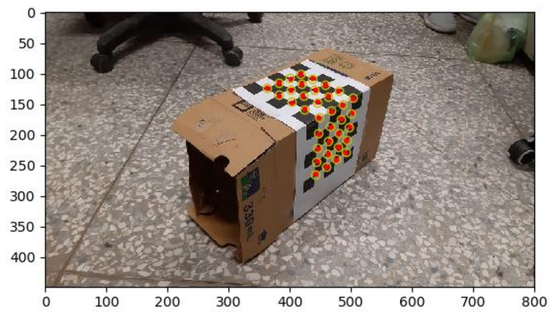
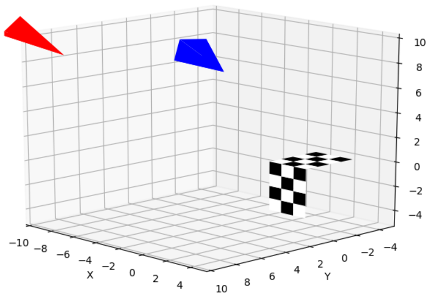
images) and compute the point re-projection root-mean-squared errors.

Chessboard 1	Chessboard 2
	
error : 0.7071067811865476	error : 0.9204467514322717

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

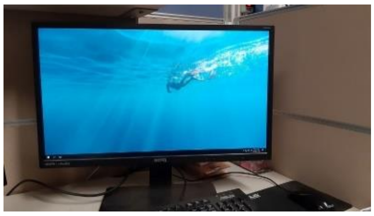
camera poses	
angle between the two camera pose vectors	Angle between two cameras: 24.4114323693213

e. **E. (Bonus) (10%) 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).**

Chessboard 1	Chessboard 2
	
	
Angle between two cameras: 41.3285572958312	

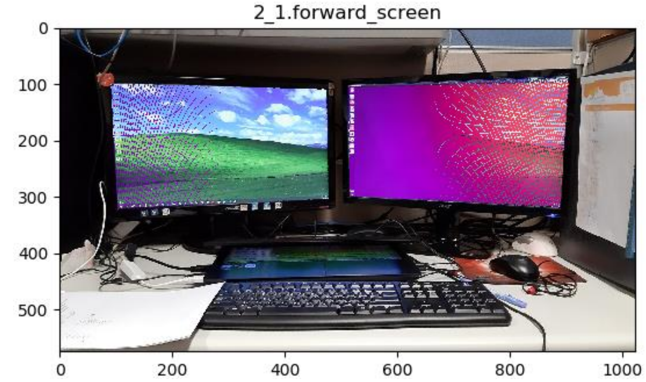
Part 2. Homography transformation

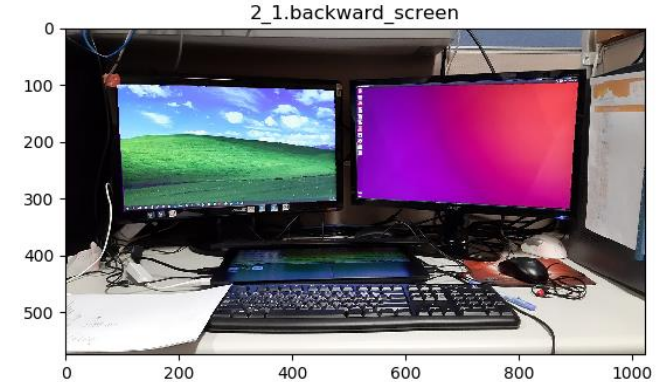
a. **Shoot three images A, B and C. Image A has to contain two objects. Image B and C should contain one object separately. Like the images shown above. (3 images)**

A	B	C
		

b. **Compute homography transformation between the two objects in image A. Use both backward and forward warping to switch them, like what example 1 shows. (2 images)**

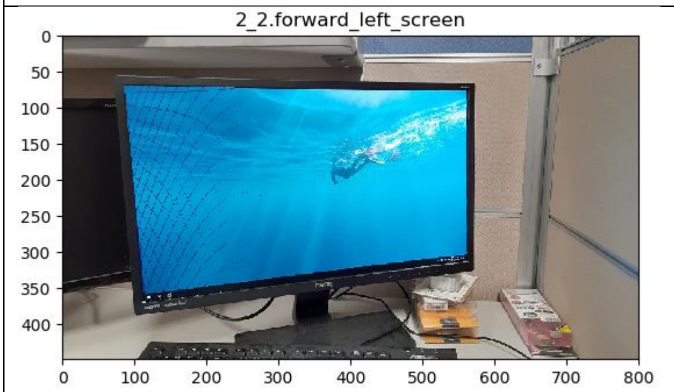
Forward warping	Backward warping

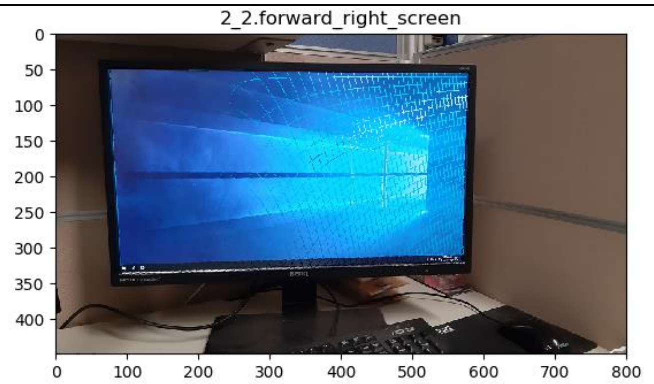




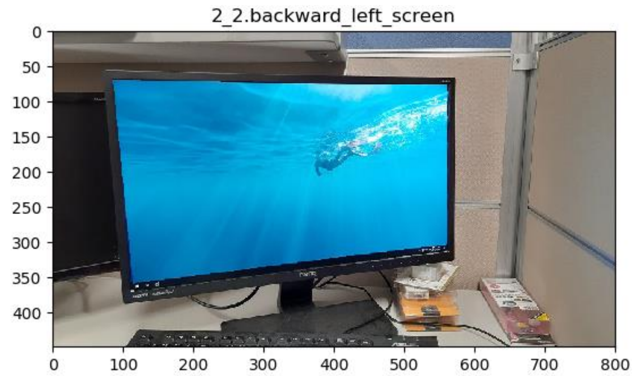
c. **Compute homography transformation between the object in image B and the object in image C. Use both backward and forward warping to switch them. Example 2 gives some illustration. (4 images)**

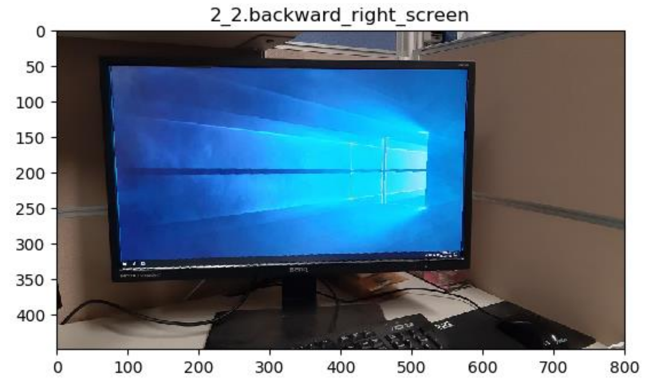
Forward warping





Backward warping





d. **Discuss the difference between forward and backward warping based on your results.**

使用Forward warping會無法完全填滿target的pixel位子。因為Forward是從source的pixel位置推算從target投影過來的pixel位置；另外因為投影計算的位置，會產生些許誤差，造成target位置無法被完全填滿。

Backward warping較能完全填滿target的pixel位子。因為Backward是直接從target的pixel位置推算從source投影過來的pixel位置，所以較能填滿。