

# ENGN4528 Computer Vision – 2021

## Computer-Lab 3 (C-Lab3)

Hongxiang Zhang(u7101924)  
<sup>1</sup>Australian National University

May 21, 2021

### 1 Task-1

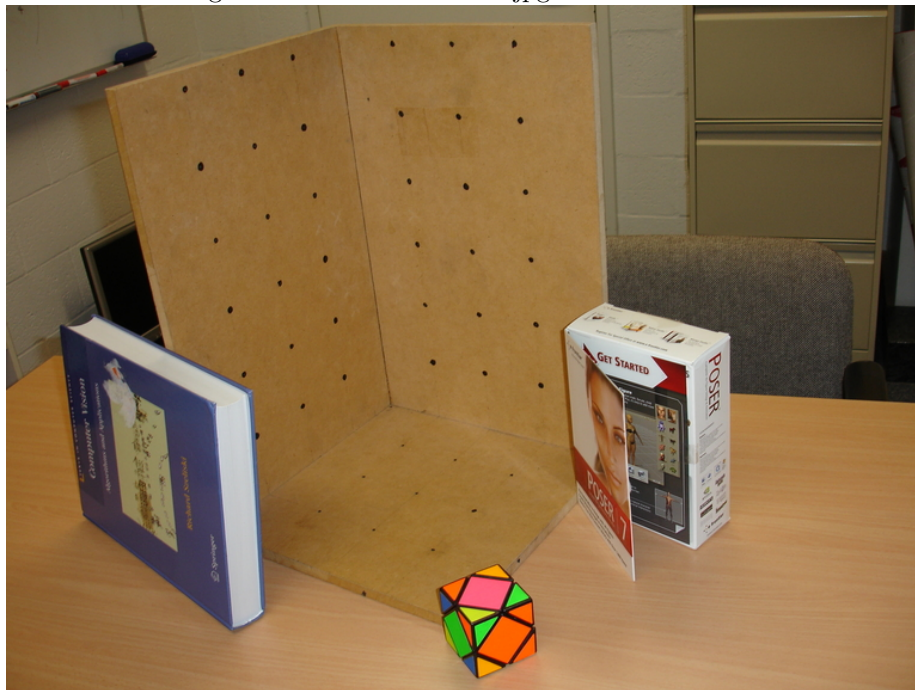
#### 1.1

```
1 def calibrate(im, XYZ, uv):
2     A = np.zeros((2*(len(XYZ)),12))
3     for i in range(A.shape[0]): #for each x_i compute A_i
4         if i%2==0:
5             tmp = np.array(XYZ[i//2])
6             A[i,4:8] = (-1*tmp).copy()
7             tmp2 = uv[i//2][1]*tmp
8             A[i,8:] = tmp2.copy()
9         else:
10            A[i,:4] = tmp.copy()
11            tmp2=(-1*tmp).copy()
12            tmp2 = uv[i//2][0]*tmp2
13            A[i,8:] = tmp2.copy()
14        u,s,v = np.linalg.svd(A) #obtain SVD of A
15        C = v[-1]
16        C = C.reshape(3,4)
17        C = np.linalg.inv(T_norm)@C@S_norm #denormalized
18        #print(C)
19        return C
```

For calibrate function, there are mainly 4 steps. First, compute  $A_i$  for each point. Second, combine all  $A_i$  matrices to one  $A$  matrix. Third, solve the SVD of  $A$  to get  $C$ . Fourth, denormalized  $C$  and reshape.

## 1.2

Answer: The image I choose is stereo2012a.jpg

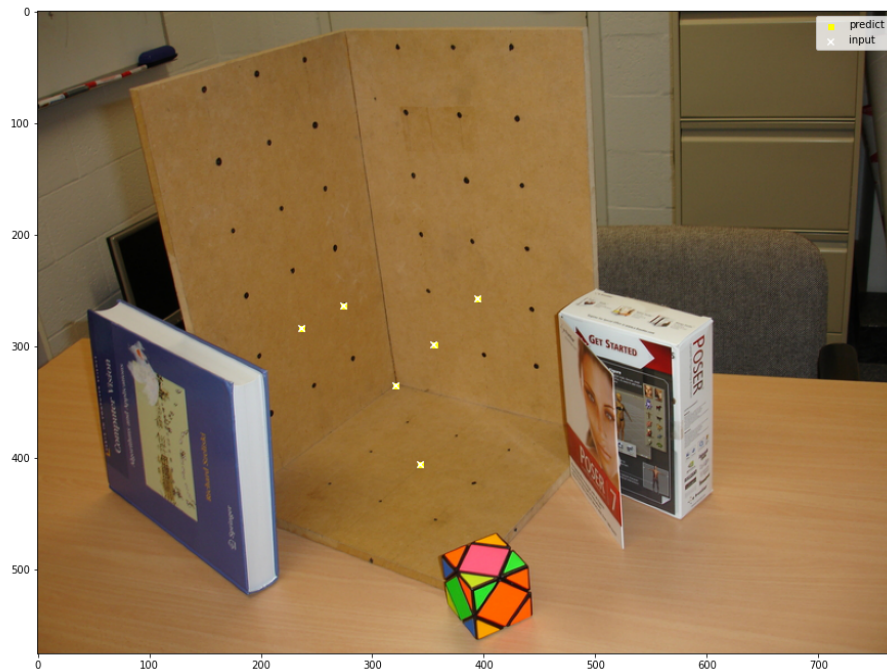


## 1.3

Answer: matrix P:

$$\begin{bmatrix} -1.69926167e+00 & 3.21130575e-01 & 1.40613179e+00 & -1.03915956e+01 \\ -4.35367606e-01 & 1.87301768e+00 & -1.08674559e+00 & 1.55109554e+01 \\ 1.32685851e-03 & 1.21078926e-03 & 2.41171036e-03 & -3.15885150e-01 \end{bmatrix}$$

Matrix P is shown above. For the image below, the white cross label indicated the XYZ coordinate I choose, Yellow square label indicated the coordinate which calculates by matrix P.



## 1.4

Answer:

Using vgg\_KR\_from\_P.py to decompose the P matrix got matrix K,R and t.

Matrix K:

```
[[ 721.7261638   -4.53488275  168.67624332]
 [   -0.         727.26733597 -102.92671541]
 [   -0.         -0.          1.          ]]
```

matrix R:

```
[[ 0.88693   -0.05960165 -0.45804239]
 [ 0.13662497 -0.91341695  0.38340982]
 [-0.44123554 -0.40263769 -0.80199382]]
```

matrix t:

```
[62.77183476 48.22114397 72.23511016]
```

## 1.5

Answer:

There are two directions of focal length, for x axis is  $k_{00}$  in matrix K, for y axis is  $K_{11}$  in matrix K.

Focal length of the camera is  $f_x = 721.7261638$ ,  $f_y = 727.26733597$

For pitch angle to the X-Z plane,  $\theta_y$  is the angle we want. The equation is  $\theta_y = \text{atan2}(-R_{31}, \sqrt{r_{32}^2 + r_{33}^2})$ .

pitch angle = -26.182739995507493

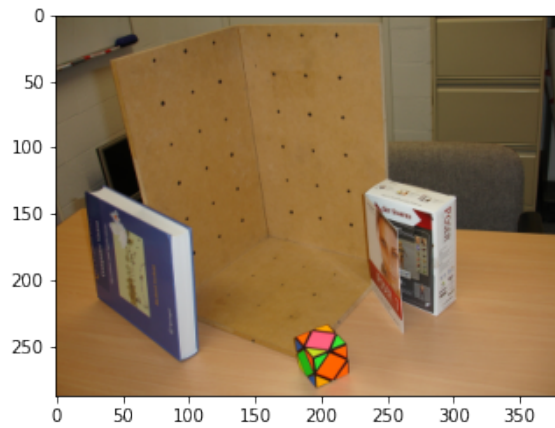
## 1.6

uv coordinate after resize:

[[160.5, 167.5], [177.5, 149. ], [137. , 131.5], [171.5, 203. ], [197. , 128.5],  
[118. , 142. ]]

### 1.6.1 a

Answer: Resized image:



matrix K':

```
[[360.8630819   -2.26744137   84.33812166]
 [ -0.          363.63366798  -51.46335771]
 [ -0.          -0.           1.           ]]
```

matrix R':

```
[[ 0.88693    -0.05960165 -0.45804239]
 [ 0.13662497 -0.91341695  0.38340982]
 [-0.44123554 -0.40263769 -0.80199382]]
```

matrix t':

```
[62.77183476 48.22114397 72.23511016]
```

### 1.6.2 b

(1)K and K'

From the matrix illustrate, K' is 1/2 of K matrix. Matrix K and K' represent the intrinsic parameters matrix which is the camera parameters. what we do is resize the image to 1/2 of the original image, which means one pixel represents more view than the original one. In this case, K\_02, K\_12 is half of the origin K\_02, K\_12, which indicates the cords of principal points. Also, the focal length of x, y-axis, and skew in the y-axis corresponding to K\_00, K\_11, and K\_01 respectively are half of the origin.

(2)R and R'

From the matrix illustrate, R' did not change. R and R' matrix represent the rotation matrix. The new image just resizes the length of the original image, which means the rotation matrix did not change.

(3)t and t'

From the matrix illustrate, t' did not change. t and t' represent the traslation. For resized image and origin image, the translation did not change. So the Translation matrix t and t' will not change.

## 2

### 2.1

Answer: For homography estimation function, There are mainly 4 steps. First, compute A.i for each point. Second, combine all A.i to a matrix A. Third, using SVD to solve A got matrix H. Fourth, denormalized H and reshape.

```
1
2 def homography(u2Trans, v2Trans, uBase, vBase):
3     A = np.zeros((2*(len(vbase)),9))
4     for i in range(A.shape[0]): #compute A.i for each point.
5         if i%2==0:
6             tmp = np.array(uBase[i//2])
7             A[i,3:6] = (-1*tmp).copy()
8             A[i,5]=-1
```

```

9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26

    tmp2 = u2Trans[i//2][1]*tmp
    A[i,6:9] = tmp2.copy()
    A[i,8] = u2Trans[i//2][1]

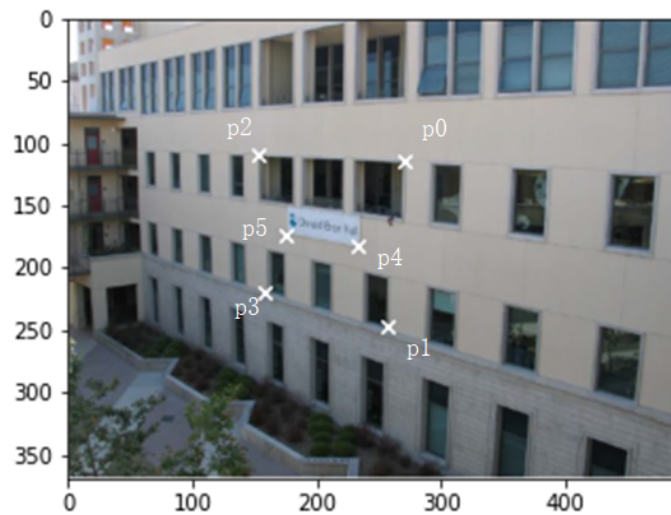
else:
    A[i,:3] = tmp.copy()
    A[i,2] = 1

    tmp2=(-1*tmp).copy()
    tmp2 = u2Trans[i//2][0]*tmp2
    A[i,6:9] = tmp2.copy()
    A[i,8] = -1*u2Trans[i//2][0]

u,s,v = np.linalg.svd(A) #SVD
C = v[-1]
C = C.reshape(3,3)
C = np.linalg.inv(T_norm)@C@T_norm #denormalized
return C

```

The image below shows the location of six pairs of selected points.



In the below image. The white cross label indicate the six selected points, and yellow square label indicate the predicted point after calculate with matrix H.



## 2.2

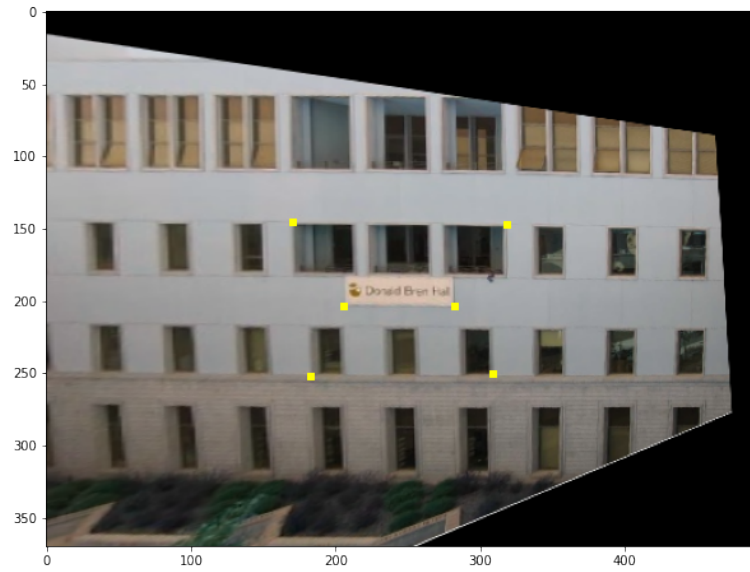
Answer: Using DLT algorithm to calculate matrix H. H shown below

$$\begin{bmatrix} 6.73269082e-01, & 4.92558408e-04, & -4.77381102e+01, \\ 1.14736644e-01, & 3.06259861e-01, & -3.90225861e+00, \\ 8.36792499e-04, & -3.85192620e-05, & 2.00796437e-01 \end{bmatrix}$$

## 2.3

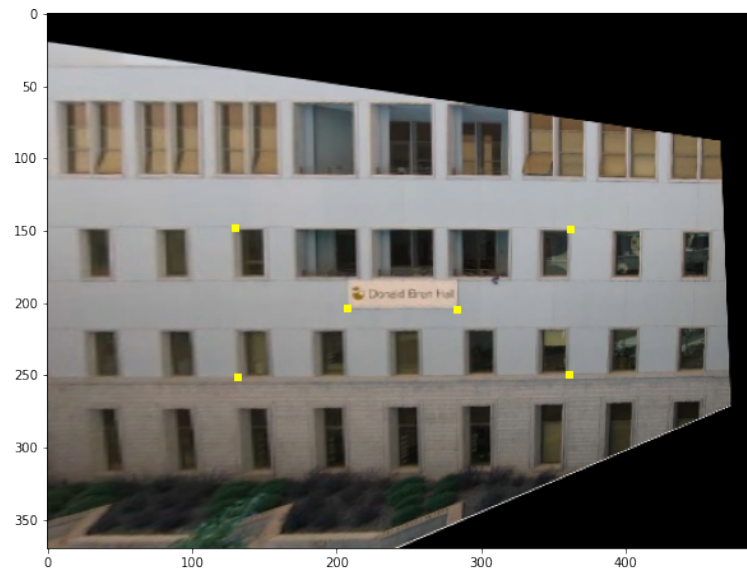
Answer: Distance for each point in group one is

$$[0.8552535889510352, 0.9874512075298797, 1.5357492316427845, 1.7922351117891206, 1.0245671826245484, 2.275832997005989]$$



Distance for each point in group two is  
[0.2822897315796688 , 1.1773835682240874 , 0.6670013865970991 , 1.1337963387135694  
, 0.04076848450723069 , 0.7024475138312357]





As the two images show, the performance of the two images is similar. But the two different matrix show, Those selected points are more sparse have a lower distance than those are dense. In this case, the more sparse selected point has a better performance.

Points