Clab-3 Report

ENGN6528

Han Zhang
u7235649

21/05/2021

Task-1: 3D-2D Camera Calibration

1. **List** calibrate **function in your PDF file.**

```
def calibrate(im, XYZ, uv):
    n = uv.shape[0]
    A = []
    for i in range(n):
        x, y = uv[i]
        X, Y, Z = XYZ[i]
        A.append([X, Y, Z, 1, 0, 0, 0, 0, -x * X, -x *
Y, -x * Z, -x])
        A.append([0, 0, 0, 0, X, Y, Z, 1, -y * X, -y *
Y, -y * Z, -y])
    A = np.array(A)
    C = np.linalg.svd(A)[-1][-1]
    C = C / C[-1]
    C = C.reshape((3, -1))
    print("C:", C.shape, "\n", C)

    h, w, _ = np.array(im).shape
    addition = np.array([
        [0, 0, 0],
        [1, 0, 0],
        [0, 1, 0],
        [0, 0, 1]
    ])

    XYZ = np.vstack((XYZ, addition))
    print(n, n + 4)
    print("XYZ:", XYZ)

    XYZ_T = np.vstack((XYZ.T, np.ones((1, n + 4))))
    uv_new = np.dot(C, XYZ_T)
    uv_new_t = np.transpose(uv_new)

    x_new = np.empty((n + 4, 1))
    y_new = np.empty((n + 4, 1))
    for i in range(0, n + 4):
        x_new[i] = uv_new_t[i][0] / uv_new_t[i][2]
        y_new[i] = uv_new_t[i][1] / uv_new_t[i][2]

    uv_proj = np.hstack((x_new, y_new))
```

```
    x_proj = uv_proj[: n, 0]
    y_proj = uv_proj[: n, 1]

    plt.title("Projected points")
    plt.imshow(im)
    plt.xlim(xmin=0, xmax=w)
    plt.ylim(ymin=h, ymax=0)
    # draw points:
    for xx, yy in zip(x_proj, y_proj):
        plt.plot(xx, yy, ".", color='blue')

    # draw lines from the origin to the vanishing
points in the X, Y and Z directions:
    original_point_x = uv_proj[n, 0]
    original_point_y = uv_proj[n, 1]

    key_points_x = uv_proj[n + 1:, 0]
    key_points_y = uv_proj[n + 1:, 1]
    dir_x = key_points_x - original_point_x
    dir_y = key_points_y - original_point_y
    colors = ['r', 'g', 'b']
    start = [(original_point_x, original_point_y) for _
in range(3)]
    flag = True
    while flag:
        flag = False
        for i in range(3):
            start_x = start[i][0]
            start_y = start[i][1]
            kx = start_x + dir_x[i]
            ky = start_y + dir_y[i]
            c = colors[i]
            plt.plot([start_x, kx], [start_y, ky],
color=c)

            if 0 < kx < w and 0 < ky < h:
                print("Draw")
                flag = True
                start[i] = (kx, ky)
    plt.show()
    return C
```

2. **List the image you have chosen for your experiment, and display the image in your PDF file**

The image I use is stereo2012b.jpg shown as Fig. 1 below.

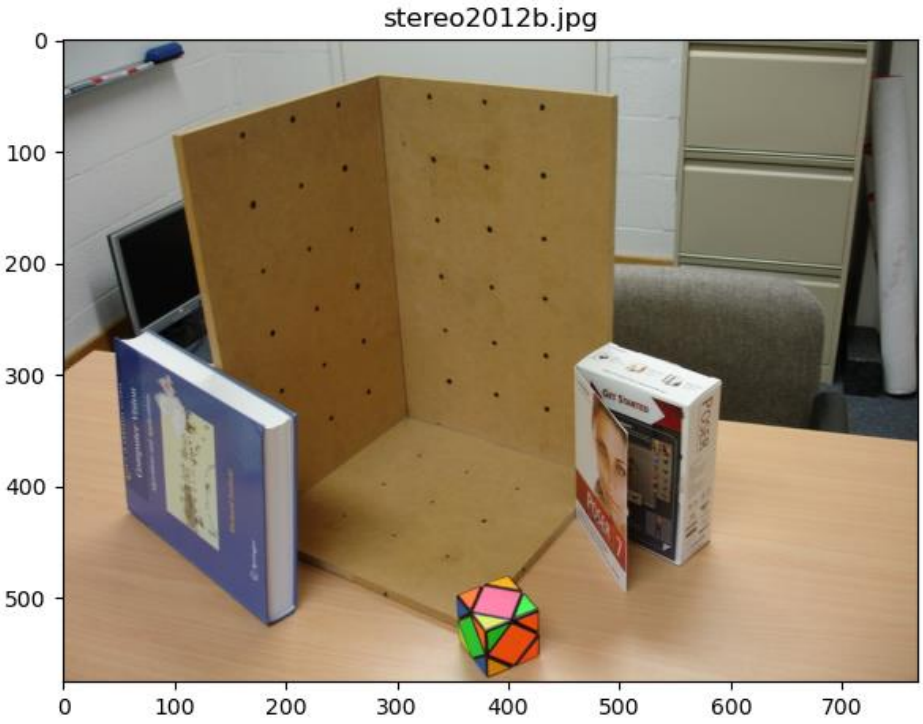

Fig. 1

3. **List the 3x4 camera calibration matrix P that you have calculated for the selected image. Visualize the projection of the XYZ coordinates back onto image using the calibration matrix P.**

The 2x4 camera calibration matrix P is

[[3.90215491e+00 -1.91264120e+00 -5.94827466e+00 3.06493892e+02]
[-4.07462784e-02 -7.35056908e+00 6.96569618e-01 3.35955247e+02]
[-5.05238264e-03 -4.67660748e-03 -6.56000251e-03 1.00000000e+00]]

The selected points on the image are shown as Fig. 2. The projected points as Fig.

3 that axis X, Y, Z are shown in red, green and blue.

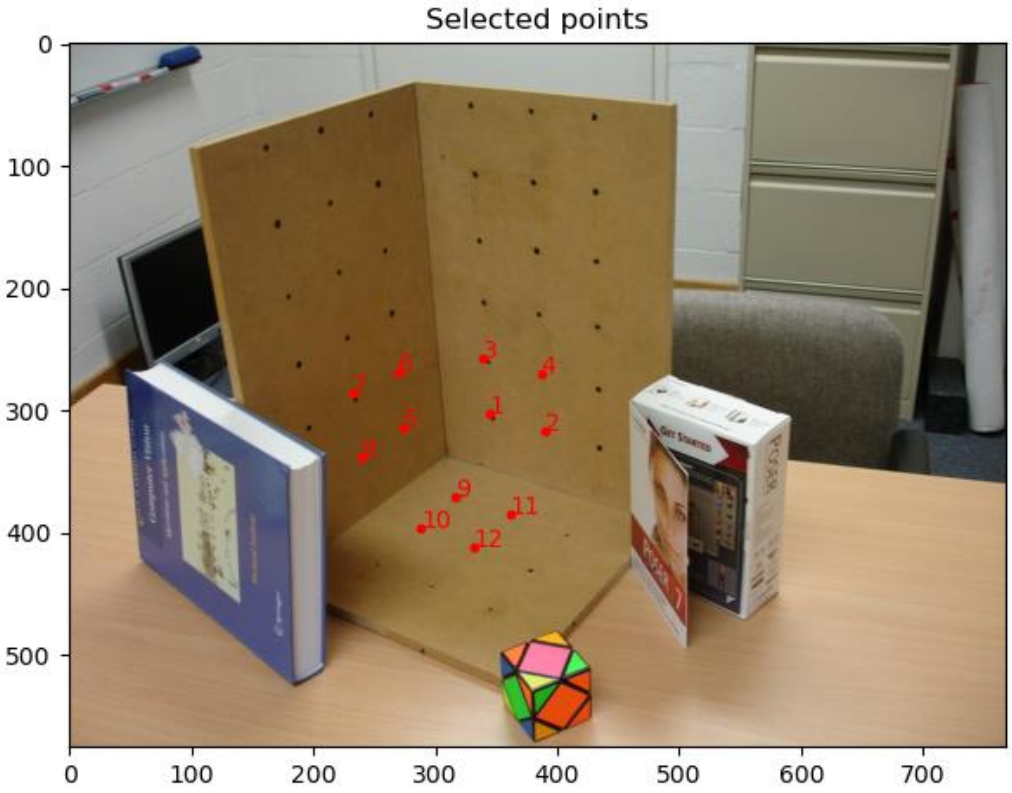

Fig. 2

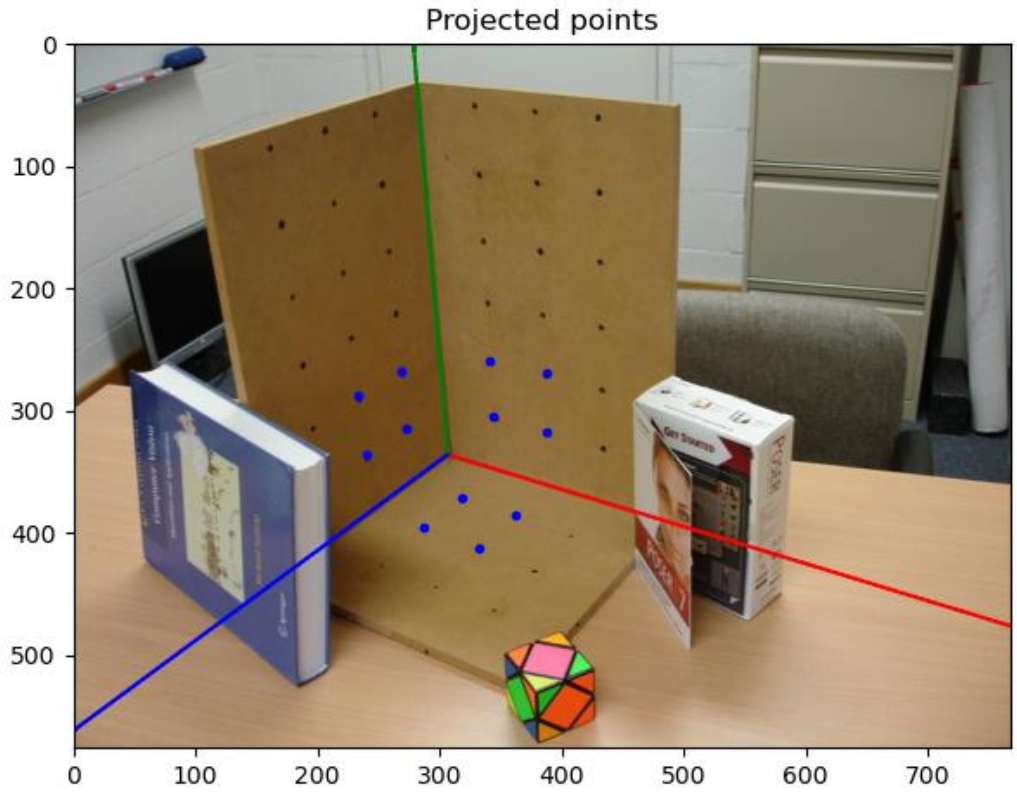

Fig. 3

4. **Decompose the P matrix into K, R, t by using the provided code. List the results, namely the K, R, t matrices.**

Matrix K:

[[708.84882905 6.00174414 312.39544625]
[0. 701.94222593 331.8788896]
[0. 0. 1.]].

Matrix R:

[[0.81095735 -0.05965338 -0.58205639]
[0.2450938 -0.86867254 0.43050789]
[-0.53129766 -0.49178195 -0.68983571]]

Matrix t:

[56.4346255 52.19267397 71.76613538]

5. (a) What is the focal length of the camera?

The focal length on X direction is round 708.85 and on Y direction is around 701.94.

(b) What is the pitch angle of the camera with respect to the X-Z plane in the world coordinate system?

Let the pitch angle is θ, according to matrix R, -sin(θ) = -0.58205639, then θ is around 0.9495 in arc degree (54.4047°).

6. Please resize your selected image using buitin function from matlab or python to (H/2, W/2) where H, and W denote the original size of your selected image.

(a) Please display your resized image in the report, list your calculated 3x4 camera calibration matrix P' and the decomposed K', R', t' in your PDF file.

Matrix P':

[[2.26903967e+00 -7.32820332e-01 -3.00145051e+00 1.53120668e+02]
[2.70730366e-01 -3.54182230e+00 5.31875426e-01 1.67357803e+02]
[-3.54723640e-03 -3.45146845e-03 -6.29501399e-03 1.00000000e+00]]

Matrix K':

[[430.8729327 -1.36301392 208.57909298]
[0. 431.21631029 123.45046911]
[0. 0. 1.]]

Matrix R':

[[0.87272716 -0.00659788 -0.48816368]
[0.20522181 -0.90231801 0.37908603]
[-0.44298004 -0.43102051 -0.7861234]]

Matrix t':

[66.70271583 65.19368555 85.52415462]

(b) Please analyze the differences between 1) K and K', 2) R and R', 3) t and t'. Please provide the reasoning when changes happened or there are no changes.

When projecting a point from 3D to 2D: $(x, y, z, 1) \rightarrow (u, v, S)$ the matrix can be represented as

$$\begin{pmatrix} a_x & 0 & u_0 \\ 0 & a_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} & R_{13} & T_x \\ R_{21} & R_{22} & R_{23} & T_y \\ R_{31} & R_{32} & R_{33} & T_z \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}.$$

After resizing the matrix is

$$\begin{pmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_x & 0 & u_0 \\ 0 & a_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} & R_{13} & T_x \\ R_{21} & R_{22} & R_{23} & T_y \\ R_{31} & R_{32} & R_{33} & T_z \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

which is equal to

$$\begin{pmatrix} 0.5a_x & 0 & 0.5u_0 \\ 0 & 0.5a_y & 0.5v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} & R_{13} & T_x \\ R_{21} & R_{22} & R_{23} & T_y \\ R_{31} & R_{32} & R_{33} & T_z \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}.$$

So, the values in matrix K except the last one will become 1/2 times, but the other matrixes R, t should not change.

In my result the values are not exactly satisfy the above law, this should because the error when remarking the point coordinates.

Task-2: Two-View DLT based homography estimation.

1. **List your source code for homography estimation function and display the two images and the location of six pairs of selected points.**

The homography estimation function *homography()*:

```
def homography(u2Trans, v2Trans, uBase, vBase):
    n = u2Trans.shape[0]
    A = []
    for i in range(n):
        a1 = [0, 0, 0, -uBase[i], -vBase[i], -1,
v2Trans[i] * uBase[i], v2Trans[i] * vBase[i],
v2Trans[i]]
        a2 = [uBase[i], vBase[i], 1, 0, 0, 0, -
u2Trans[i] * uBase[i], -u2Trans[i] * vBase[i], -
u2Trans[i]]
        A.append(a1)
        A.append(a2)
    A = np.array(A)
    H = np.linalg.svd(A)[-1][-1]
    H = H / H[-1]
    H = H.reshape((3, -1))
    return H
```

The image Left.jpg and the selected points are shown as Fig. 4. The image Right.jpg and the selected points are shown as Fig. 5.

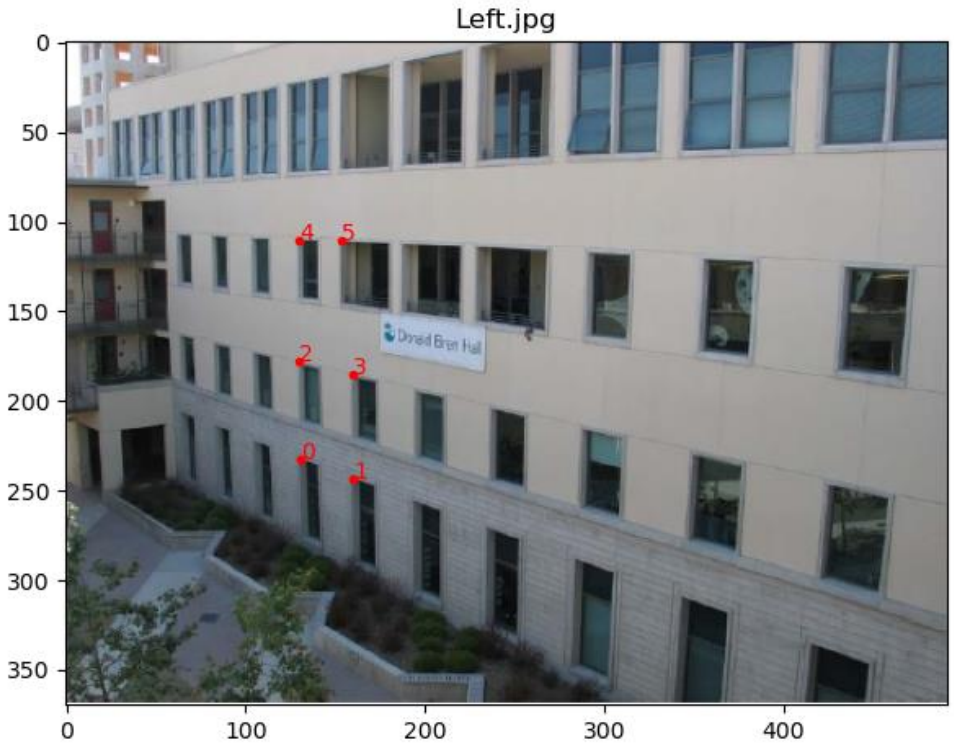
Left.jpg
0
50
100
150
200
250
300
350
0
100
200
300
400
4
5
2
3
0
1

Fig. 4

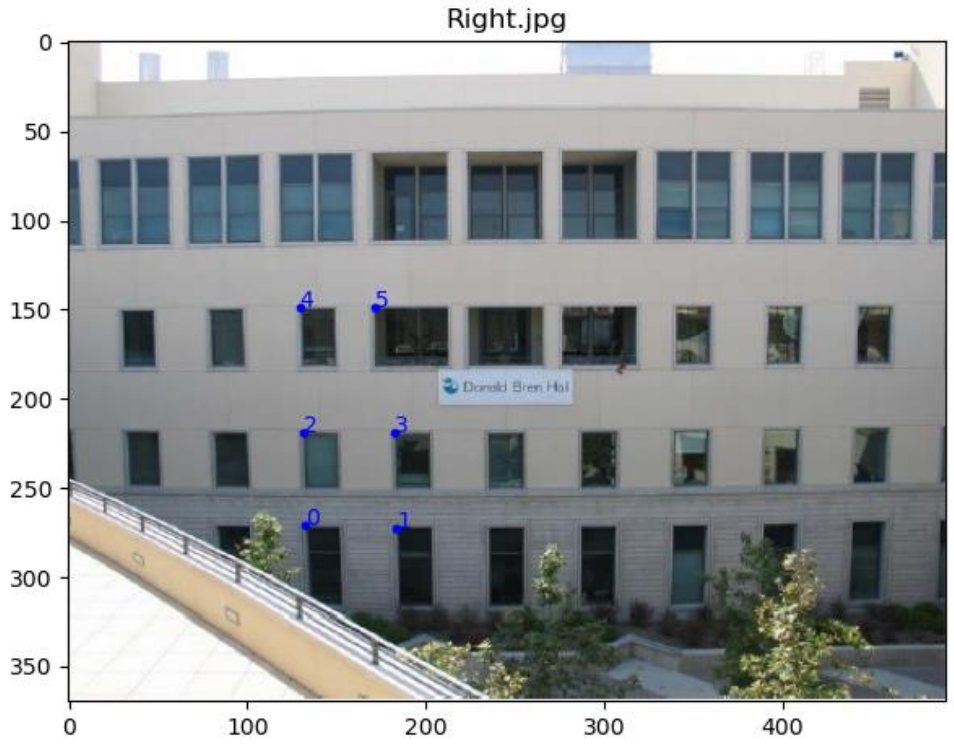
Right.jpg
0
50
100
150
200
250
300
350
0
100
200
300
400
4
5
2
3
0
1

Fig. 5

2. List the 3x3 camera homography matrix H that you have calculated.

The 3x3 camera homography matrix H:

[[3.49159709e+00 9.55220683e-02 -2.63451764e+02]
[5.11482719e-01 1.69131436e+00 -2.00283439e+01]
[3.88926025e-03 4.81957769e-04 1.00000000e+00]]

3. Warp the left image according to the calculated homography. Study the factors that affect the rectified results, e.g., the distance between the corresponding points, e.g the selected points and the warped ones.

The warped image of Left.jpg is shown as Fig. 6 below.

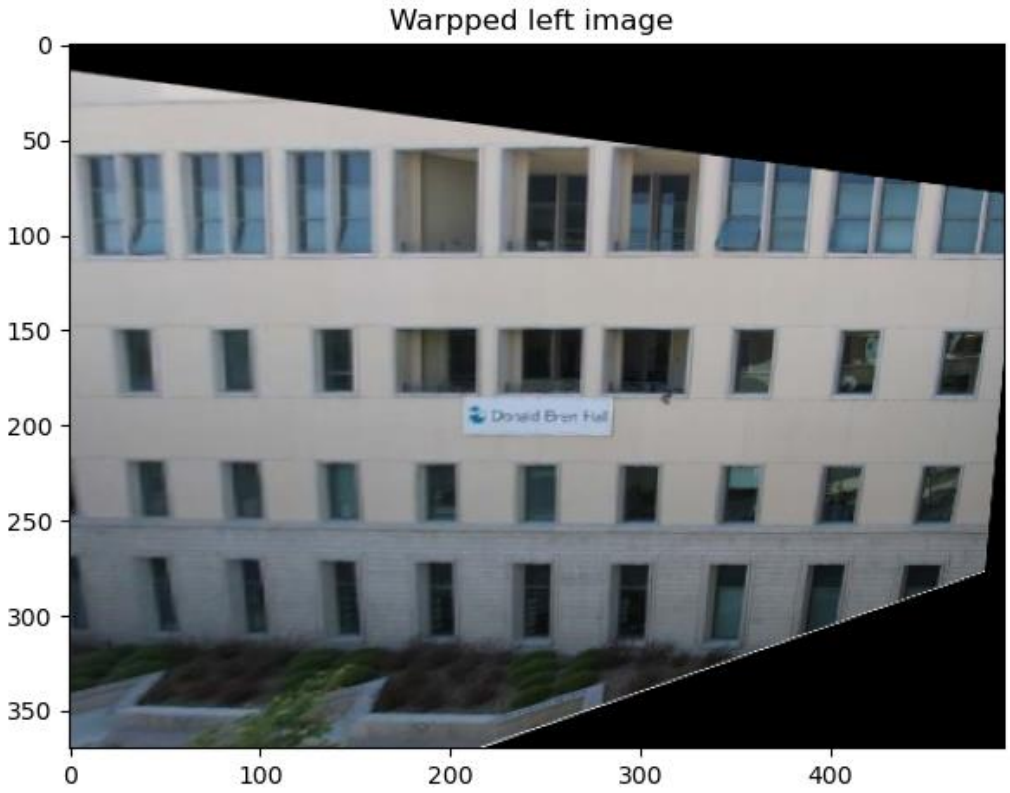

Fig. 6

Choosing a point farther away will improve the accuracy of the result, because the error in marking the point will have a smaller effect on the result.

The accuracy of the selected points between the base image and the warped image will also affect the result. The selected points in the base image and the warped image should be as close as possible in the real world.