Clab-3 Report

ENGN4528

Yixi Rao

U6826541

1. List calibrate function in your PDF file. [3 marks]

```python
def calibrate(im, XYZ, uv):
    #*---------------------------------------------plotting----------------------------------------
    N     = XYZ.shape[0]
    S, T_ = Normalize(XYZ, im.shape[0], im.shape[1])

    h_XYZ = np.hstack((XYZ, np.ones(N).reshape(-1, 1)))
    h_uv  = np.hstack((uv,  np.ones(N).reshape(-1, 1)))
    n_XYZ = h_XYZ @ S.T #
    n_uv  = h_uv  @ T_.T #
    A     = None

    for i in range(N):
        Xi          = n_XYZ[i]
        xi, yi ,wi = n_uv[i]
        Ai          = np.vstack((np.hstack((np.zeros(4), -1 * wi * Xi,   yi * Xi     )),
                                 np.hstack((wi * Xi,     np.zeros(4),    -1 * xi * Xi))
                                ))

        A           = np.vstack((A, Ai)) if i != 0 else Ai

    _, _, v   = np.linalg.svd(A)
    C         = (v[-1] / np.sum(v[-1])).reshape(3, 4) #
    P_denorm = np.linalg.inv(T_) @ C @ S
```

```python
def Normalize(XYZ, h, w):
    lambdas, V = np.linalg.eig((XYZ.T - np.mean(XYZ, axis=0).reshape(-1, 1)) @ (XYZ.T - np.mean(XYZ,
                   axis=0).reshape(-1, 1)).T)
    S    = np.vstack((np.hstack((V @ np.diag(1 / lambdas) @ np.linalg.inv(V), -1 * V @ np.diag(1 /
         lambdas) @ np.linalg.inv(V) @ np.mean(XYZ, axis=0).reshape(-1, 1))),
                     np.hstack((np.zeros(3), np.ones(1)))
                    ))
    T_   = np.linalg.inv(np.array([[w + h, 0,     w / 2],
                                   [0,     w + h, h / 2],
                                   [0,     0,         1]]))
    return (S, T_)
```

The calibrate function first converts the XYZ and uv, which are a N x 3 array of XYZ coordinates of the calibration target points and N x 2 array of the image coordinates of the calibration target points, to homogeneous XYZ, uv. Then, XYZ and uz are normalized by using the S_{norm} and T_{norm} (the normalization method I used is the same as the method introduced in lecture slide). The function then performs the DLT algorithm and calculates a 12 x 12 A matrix using the XYZ and uz, the final C (normalized) is gained by using the SVD method, and then denormalize it using the S_{norm} and T_{norm} to gain the final result.

2. List the image you have chosen for your experiment, and display the image in your PDF file. [0.5 mark]

The image I choose is stereo2012a.jpg.

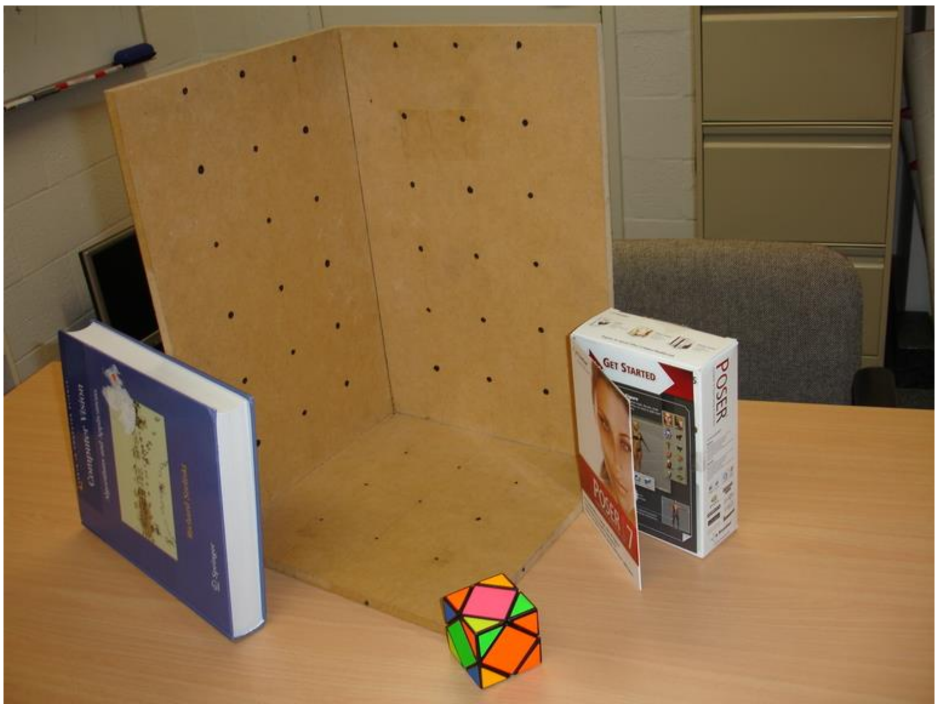
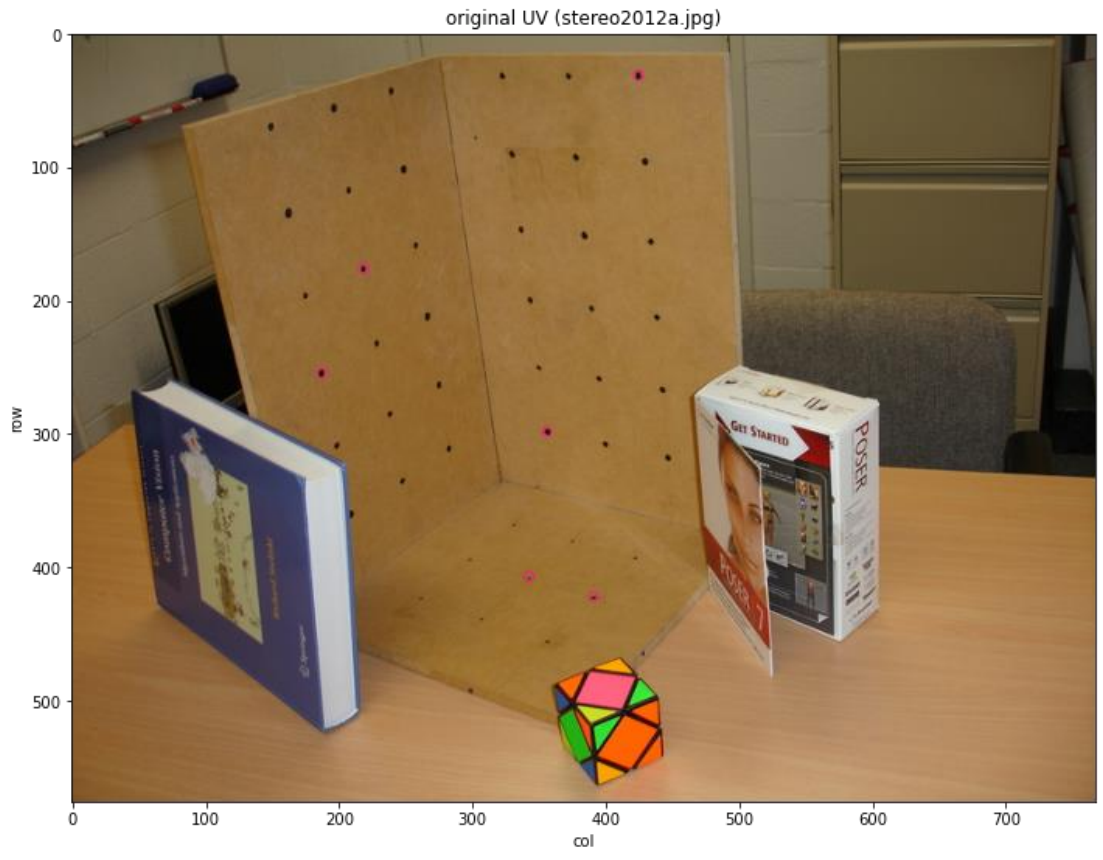


Figure 1.1 stereo2012a.jpg

3. List the 3x4 camera calibration matrix P that you have calculated for the selected image. Please visualize the projection of the XYZ coordinates back onto image using the calibration matrix P and report the reprojection error

$$\begin{bmatrix} -0.448695 & 0.224832 & 0.682142 & -35.507107 \\ 0.013278 & 0.807537 & -0.120159 & -36.779717 \\ 0.000496 & 0.00039 & 0.000675 & -0.110368 \end{bmatrix}$$

figure 1.2 matrix P



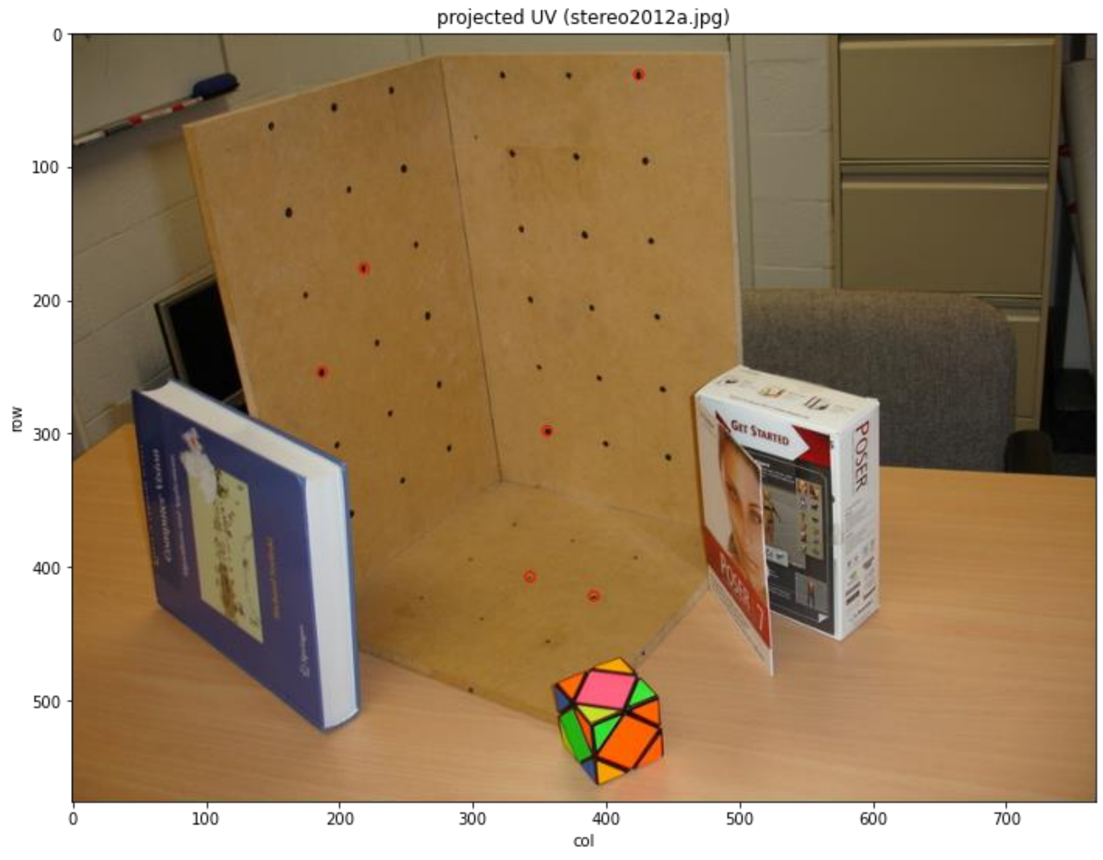


Figure 1.3 projection of the XYZ coordinates back onto image using the calibration matrix P

Figure 1.3 shows the original uv coordinates (pink circle) and the projected uv coordinates (red circle) using the calibration matrix. We can see that the calibration image projects the original XYZ coordinates onto the same places where the original uv located in the image. the MSE is 0.4049678700008122.

4. Decompose the P matrix into K, R, t, such that P = K[R|t], by using the following provided code. List the
results, namely the K, R, t matrices, in your PDF file.

$$\begin{bmatrix} 833.483403 & 2.882374 & 381.353613 \\ 0.0 & 837.789408 & 281.297979 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$$

Figure 1.4 matrix K

$$\begin{bmatrix} 0.827736 & -0.095874 & -0.552866 \\ 0.163113 & -0.901634 & 0.400564 \\ -0.536886 & -0.421741 & -0.730677 \end{bmatrix}$$

Figure 1.5 matrix R

$$\begin{bmatrix} 70.020851 & 56.234781 & 79.575377 \end{bmatrix}$$

Figure 1.6 matrix t

Using the vgg_KR_from_P function, we can get the K, R, t results, which are shown in figure 1.4 ,1.5, 1.6.

5. Please answer the following questions:
- What is the focal length (in the unit of pixel) of the camera?
- What is the pitch angle of the camera with respect to the X-Z plane in the world coordinate system?
- What is the camera center coordinate in the XYZ coordinate system

1. the focal length (in the unit of pixel) of the camera is $f_x = 834, f_y = 838\ (in\ the\ unit\ of\ pixel)$.

2. the pitch angle is -51.03553328361391 (degree). This is calculated from the R matrix, we know that $R = R_x R_y R_z$ and assume that

$R_x(\alpha), R_y(\beta), R_z(\gamma)$ as defined in lecture slide. We then know $R[0,2] = \sin(\beta)$ by matrix multiplication. β can be gained by using the $\text{atan}()$ function.

3. the camera center is $(70.02085112,\ 56.23478138,\ 79.57537657)$. Form the lecture slides we know that $P = K[R|-RC] = [P_1, P_2, P_3, P_4]$, So $PC = 0$. We then can use the SVD to calculate the camera center coordinate.

6. Please resize your selected image using builtin function from matlab or python to (H/2, W/2) where H, and W denote the original size of your selected image. Using the interface function, to find the uv coordinates in the resized 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.

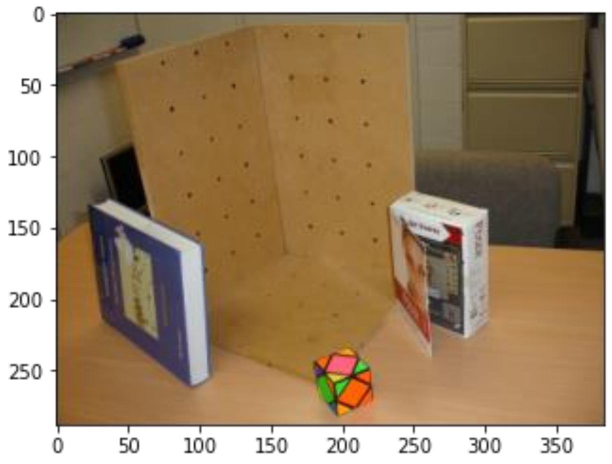


Figure 1.7 resized stereo2012a.jpg

$$\begin{bmatrix} -0.253729 & 0.107417 & 0.365652 & -18.963235 \\ 0.007081 & 0.432844 & -0.075249 & -19.746926 \\ 0.000519 & 0.000343 & 0.000677 & -0.118476 \end{bmatrix}$$

Figure 1.8 P' matrix

$$\begin{bmatrix} 464.187167 & -2.699607 & 180.560377 \\ 0.0 & 462.816258 & 119.546039 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$$

Figure 1.9 K' matrix

$$\begin{bmatrix} 0.815076 & -0.112149 & -0.568395 \\ 0.129232 & -0.921171 & 0.367073 \\ -0.564756 & -0.372647 & -0.736332 \end{bmatrix}$$

Figure 1.10 R' matrix

$$\begin{bmatrix} 75.67142 & 59.489785 & 86.894316 \end{bmatrix}$$

Figure 1.11 t' matrix

The figure 1.7 shows the resized image of stereo2012a.jpg, I resize it by using the cv2.resize() function. And using the vgg_KR_from_P function to obtain the K, R, t matrix, they are shown in figure 1.9, 1.10. 1.11.

b. Please analyses 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.

1) $\boldsymbol{K}$ and $\boldsymbol{K'}$

We can observe that the focal length f_x and f_y of matrix K is nearly equal to twice focal length f_x' and f_y' of matrix K'. This is because the image size is halved, which means the uv is halved but XYZ is not changed. Matrix A is computed by

$$A_i = \begin{bmatrix} 0^T & -w_i * x_i & y_i * x_i \\ w_i * x_i & 0^T & -x_i * x_i \end{bmatrix},\ A_i' = \begin{bmatrix} 0^T & -\frac{1}{2} w_i * X_i & \frac{1}{2} y_i * X_i \\ \frac{1}{2} w_i * x_i & 0^T & -\frac{1}{2} x_i * X_i \end{bmatrix} = \frac{1}{2} A_i$$

So $f_x, f_y = 2 * f_x', 2 * f_y'$

CV2 assume that the original point of the image coordinate is in the top-left corner of the image. So, the original principal point image coordinate is $(381, 281)$, which is the center of the image, so the principal point of the resized image is also should be located in the center of the resized image and it is $(181, 120)$, which is nearly the half of $(381, 281)$.

2) $\boldsymbol{R}$ and $\boldsymbol{R'}$, $\boldsymbol{t}$ and $\boldsymbol{t'}$

We can observe that R and t are not change, this is because these are the extrinsic camera parameters, their function is translating the world coordinate to the camera coordinate. As we only change the image size and we are not changing the camera and the world coordinate system, so the extrinsic camera parameters should not change. There are some differences because we digitize the image twice using the ginput manually, we cannot guarantee that we select the exact same points again.

c. Let us check the focal length (f and f') (in pixel unit) and the principal points extracted from K and K', respectively. Please discuss their relationship between (f and f') and its connection to the image size of the original image and the one after resizing.

Form the matrix K and k', the relationship between f and f' is $f = 2 * f'$, and the relationship between p and p' is $p = 2 * p'$. We know that $\frac{f_x}{Z} = \frac{x_i}{X_c}$ and $\frac{f_y}{Z} = \frac{y_i}{Y_c}$, now we assume that the scaling of original image and resized image is c, for example, c can be $\frac{1}{2}$, which means $\frac{img}{img'} = \frac{1}{c} = 2$. We can conclude that the relationship between f and f' is $\frac{f}{f'} = \frac{1}{c}$, this is because

$$f_x' = \frac{x_i \ * \ C}{X_c} * Z, \qquad f_y' = \frac{y_i \ * \ C}{X_c} * Z$$

the relationship between p and p' is $\frac{p}{p'} = \frac{1}{c}$, this is because the principle point should be the center coordinate of the image, so when the image is scaled, the principle points are also scaled at the same ratio.

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 (namely, plotted those points on images). Explain what you have done for the homogrqaphy and what is shown.

```python
def homography(u2Trans, v2Trans, uBase, vBase):
    N      = uBase.shape[0]
    T1     = T_norm(370, 492)
    T2     = T_norm(370, 492)

    uvBase     = np.hstack((uBase, vBase))
    uv2Trans   = np.hstack((u2Trans, v2Trans))

    h_uvBase   = np.hstack((uvBase,   np.ones(N).reshape(-1, 1)))
    h_uv2Trans = np.hstack((uv2Trans, np.ones(N).reshape(-1, 1)))

    n_uvBase   = h_uvBase   @ T1.T #
    n_uv2Trans = h_uv2Trans @ T2.T #

    A = None
    for i in range(N):
        Xi         = n_uvBase[i]
        xi, yi ,wi = n_uv2Trans[i]
        Ai         = np.vstack((np.hstack((np.zeros(3), -1 * wi * Xi,   yi * Xi     )),
                                np.hstack((wi * Xi,     np.zeros(3),    -1 * xi * Xi))
                                ))
        A          = np.vstack((A, Ai)) if i != 0 else Ai

    _, _, v  = np.linalg.svd(A)
    H = (v[-1] / np.sum(v[-1])).reshape(3, 3)
    H = np.linalg.inv(T2) @ H @ T1
    return H
```

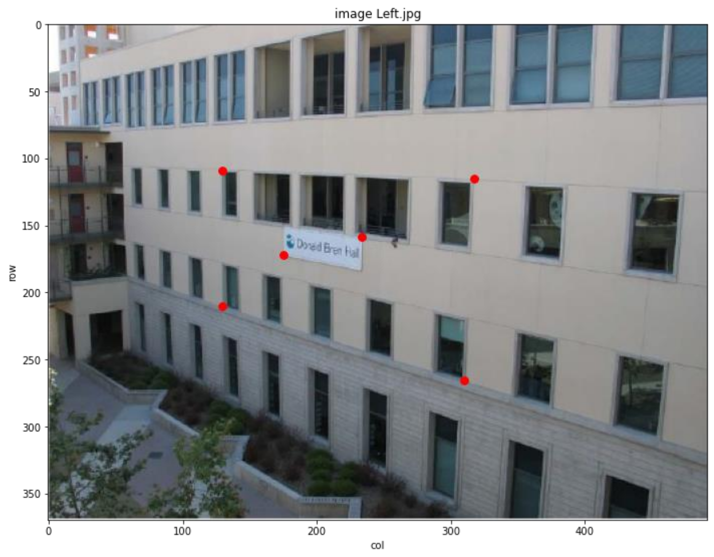


Figure 2.1 image left.jpg

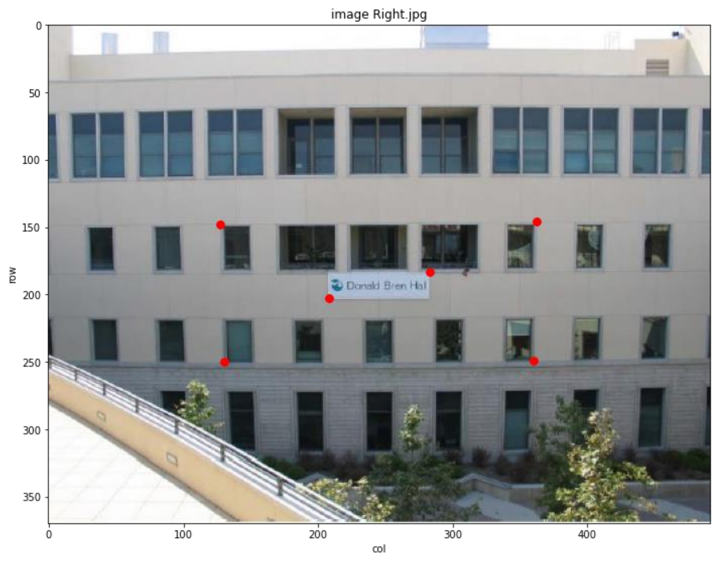


Figure 2.2 image Right.jpg

Figure 2.1 and 2.2 shows six pairs of corresponding points I selected from the Left image and Right image. We can observe that the left image is taken from a side view, the six points form a parallelogram, and the right image is taken from a front view, the six points form a rectangle.

For the homography estimation function, I first combine all the u coordinates and the v coordinates together into uv pairs by using the $p.hstack((u,v))$, then I turn all the uv pairs into homogeneous uv pairs by using $np.hstack((uvBase, np.ones(N).reshape(-1,1)))$. To get a better performance, I decide to normalized the uv pairs by using the T_{norm} matrix defined in the lecture slide. For the DLT algorithm, the A_i is computed for each correspondence (x_i, X_i), and I only take the first two rows. The final matrix A is gained by assembling all the

A_is together to form a 12 x 9 matrix A. It is hard to get the over-determined solution, so I decide to use SVD, and take the right-most column of V as the final solution. Besides, the final homography matrix are normalized to ensure that it sums to 1.

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

$$\begin{bmatrix} 0.360944 & 0.002849 & -26.905881 \\ 0.059985 & 0.159802 & -1.969633 \\ 0.000451 & -1.0 \cdot 10^{-5} & 0.100748 \end{bmatrix}$$

Figure 2.3 matrix H

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.

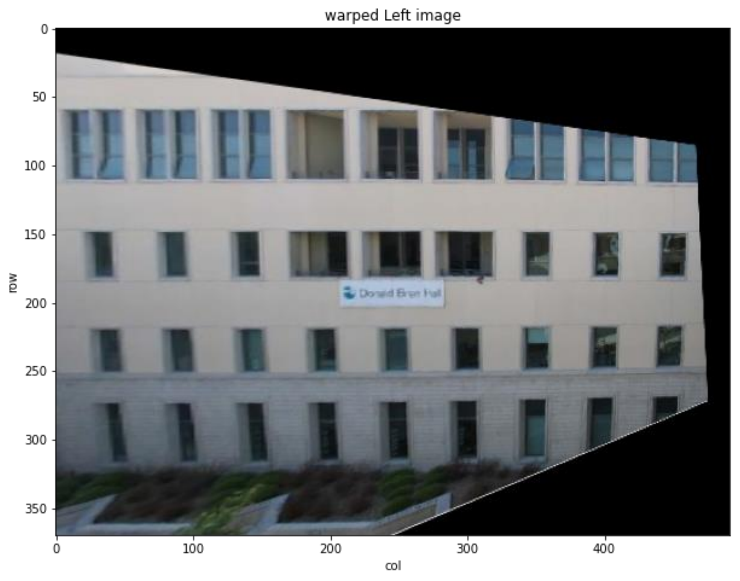
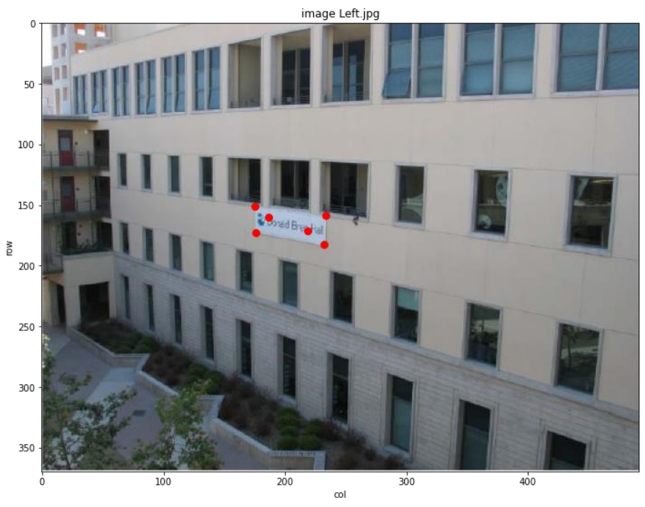


Figure 2.3 warped left image (from the original selection of points)

Figure 2.3 shows the warped left image using H, we can see that the majority of the building in the image are transformed to the front view, especially for the second and third row of windows, which are aligned uniformly and reasonably. For the other parts of the image, such as the ground are distorted massively, this is because the six selected pairs of point only form a plane for the building not for the ground.

The factors that affect the rectified results can be the distance between the corresponding points, the number of points that we selected, and how we select the points.

For the distance between the corresponding points, I found that if the six corresponding points are selected very close will make the warped image very distorted and fuzzy, for example, I selected some very close six points shown in figure 2.4.



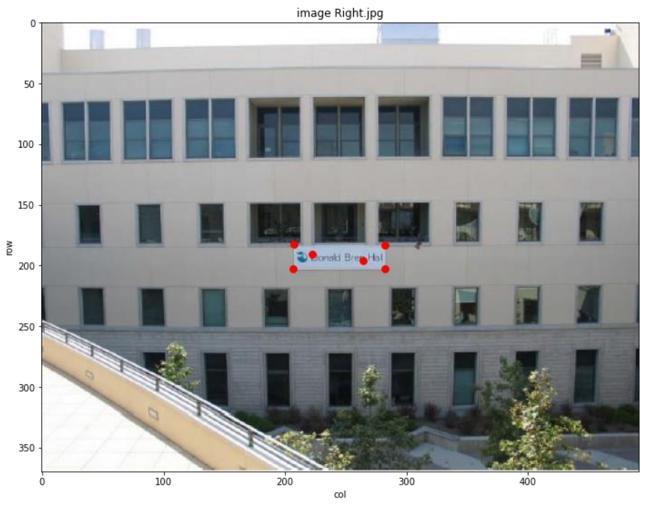


figure 2.4 selection of points are close

we will get a very fuzzy and distorted warped image and only the area covered by the points is very clear and correct. Figure 2.5 shows this result. If the six corresponding points are selected far away from each other, then we will get a better result and a clearer image. I think the reason is more areas covered by the points for the calculation of the matrix H will provide more information about the plane. And therefore, provide a decent warped image.

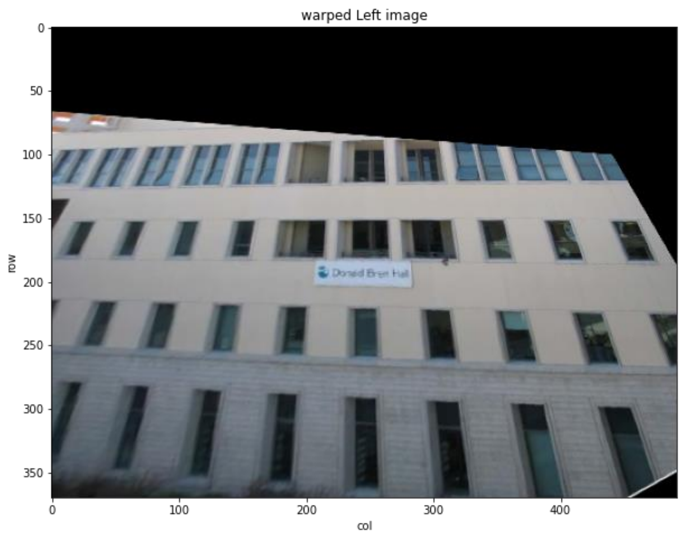


figure 2.5 warped image of selection of points are close

For the pattern of selection of points, the six points we select should form a coplanar plane such as a parallelogram, the reason is we want to implement the image homography transformation. If the points we select are a line, the H matrix will only do line homography transformation, so the warped image will be very bad. Figure 2.7 shows this result.

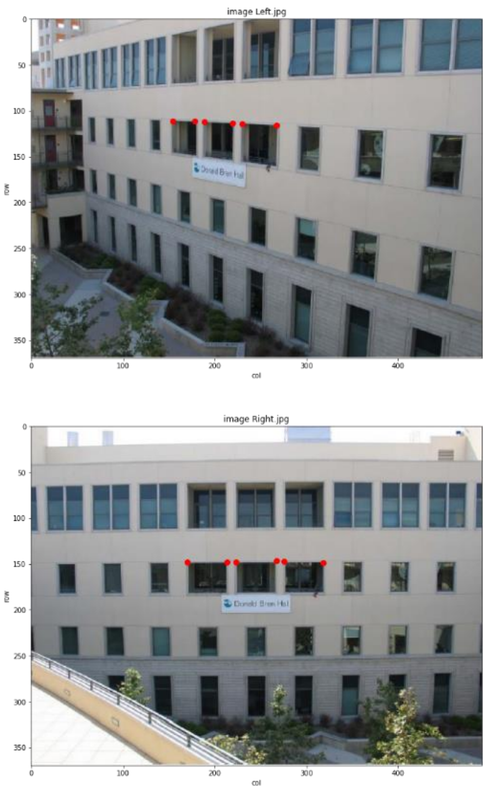


figure 2.6 selection of points form a line

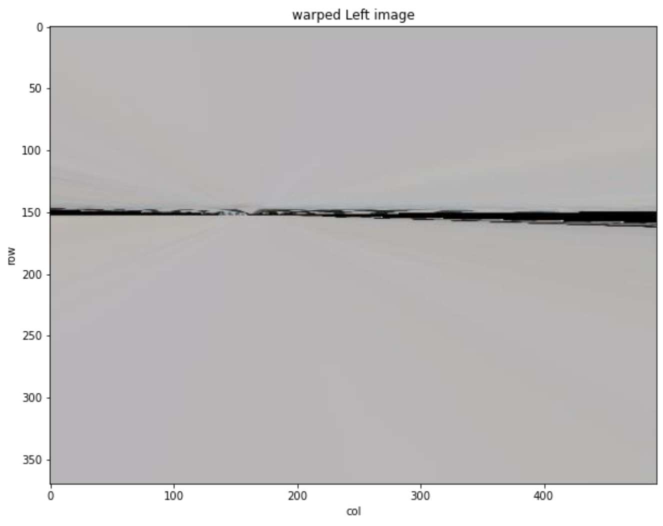


figure 2.7 warped image (selection of points form a line)

figure 2.6 shows the points I select from a line, figure 2.7 shows the warped image using the matrix H calculated using the points in figure 2.6. we can observe that, the warped result are a line, the performance is very bad.

For the number of points that we selected, it is obvious that if we have more points for the calculation of the matrix H, the warped image will perform better. In an extreme case, if we have all the corresponding points of the building in the left image and the right image, the calculated matrix H will be very accurate and provide a perfect warped image. On the other hand, if we use total 8 points, 4 points of it are used to form a plane in the building, others are used to form a plane in the ground, the resulted warped image will solve the problem of having bad performance of the ground.