Computer Vision HW3 Report

Student ID: R10921A36
Name: 石子仙

- Part 1 (2%)

1. Paste the function solve_homography( ) (1%)

```
def solve_homography(u, v):
    N = u.shape[0] # N=4
    H = None

    if v.shape[0] is not N:
        print('u and v should have the same size')
        return None
    if N < 4:
        print('At least 4 points should be given')

    uv = u*v
    uxvx = uv[:,0]
    uyvy = uv[:,1]

    uvexg = u*v[:,[1,0]]
    uxvy = uvexg[:,0]
    uyvx = uvexg[:,1] # shape is (4,)

    xeq = np.concatenate((u, np.ones(shape=(N, 1)), np.zeros(shape=(N, 3)), -1 * uxvx[...,
None], -1 * uyvx[..., None], -1 * (v[:,0])[..., None]), axis=1)
    yeq = np.concatenate((np.zeros(shape=(N, 3)), u, np.ones(shape=(N, 1)), -1 * uxvy[...,
None], -1 * uyvy[..., None], -1 * (v[:,1])[..., None]), axis=1)
    A = np.concatenate((xeq, yeq), axis=0)
    _, _, vt = np.linalg.svd(A)
    H = vt[-1, :]
    return H.reshape(3, 3)
```

2. Paste your warped canvas (1%)

- Part 2 (2%)

1. Paste the function code warping( ) (both forward & backward) (1%)

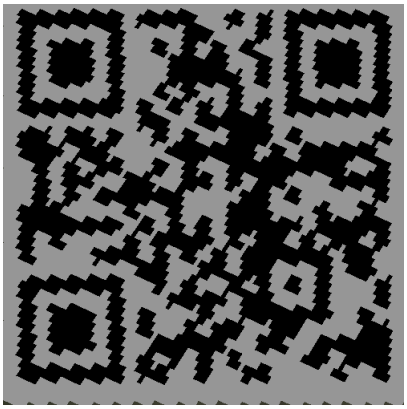
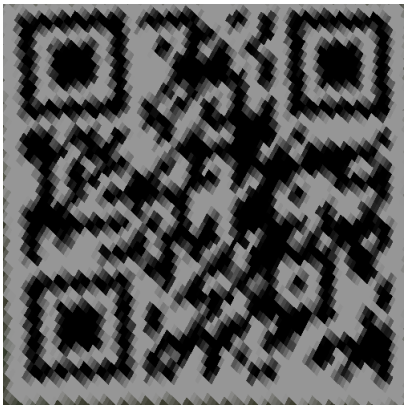
```
def warping(src, dst, H, ymin, ymax, xmin, xmax, direction='b'):
    h_src, w_src, ch = src.shape
    h_dst, w_dst, ch = dst.shape

    xv, yv = np.meshgrid(np.arange(xmin, xmax, 1), np.arange(ymin, ymax, 1))

    xv_flatten = xv.flatten()
    yv_flatten = yv.flatten()
    one_flatten = np.ones(shape=(xv_flatten.shape))

    mat = np.array([xv_flatten, yv_flatten, one_flatten])

    if direction == 'b':
        H_inv = np.linalg.inv(H)
        new_mat = np.dot(H_inv, mat)
        new_mat = new_mat / new_mat[-1, :]
        new_mat = np.round(new_mat)
        mask = (new_mat[0,:] >= 0) & (new_mat[0,:] < w_src) & (new_mat[1,:] >= 0) &
(new_mat[1,:] < h_src)

        valid_src_x = (new_mat[0,:].astype(int))[mask]
        valid_src_y = (new_mat[1,:].astype(int))[mask]
        valid_dst_x = (mat[0,:].astype(int))[mask]
        valid_dst_y = (mat[1,:].astype(int))[mask]

    elif direction == 'f':
        new_mat = np.dot(H, mat)
```

```
        new_mat = new_mat / new_mat[-1, :]
        new_mat = np.round(new_mat)
        mask = (new_mat[0,:] >= 0) & (new_mat[0,:] < w_dst) & (new_mat[1,:] >= 0) &
(new_mat[1,:] < h_dst)

        valid_src_x = (mat[0,:].astype(int))[mask]
        valid_src_y = (mat[1,:].astype(int))[mask]
        valid_dst_x = (new_mat[0,:].astype(int))[mask]
        valid_dst_y = (new_mat[1,:].astype(int))[mask]

    dst[valid_dst_y, valid_dst_x] = src[valid_src_y, valid_src_x]
    return dst
```

2. Briefly introduce the interpolation method you use (1%)
   我使用 nearest neighbor 方法，透過 np.round()四捨五入來找到最接近的點。

● Part 3 (8%)

1. Paste the 2 warped QR code and the link you find (1%)

sources	BL_secret1.png	BL_secret2.png
output		
link	http://media.ee.ntu.edu.tw/courses/cv/21S/	http://media.ee.ntu.edu.tw/courses/cv/21S/

2. Discuss the difference between 2 source images, are the warped results the same or different? (3%)
   第一張圖片沒有形變，第二章圖片有類似魚眼的形變效果。前者 warping 出來的 qr code 較清晰，第二章比較模糊，但兩個可以掃到相同 qr code。

3. If the results are the same, explain why. If the results are different, explain why. (4%)
   兩者結果不同，我認為是因為第二張經過像魚眼的轉換，無法用 perspective coordinate 解得很精確（perspective transform 不平行但也都是直線）。

● Part 4 (8%)

1. Paste your stitched panorama (1%)

2. Can all consecutive images be stitched into a panorama? (3%)
   No

3. If yes, explain your reason. If not, explain under what conditions will result in a failure? (4%)
   以我們程式實作的方式是無法達成，因為投影有分為平面投影、圓柱、圓錐等等（如下圖），程式使用的是平面投影，最多只能拍到 180 度內的範圍，超過的範圍就投不出來了。如果想要連接更多圖片，應該考慮先對圖片做其他投影處理（如圓柱投影），再接起來。另外”連續的”圖片若為→→→←←這樣拍（拍一拍又往回拍）這樣也會接不起來，需訂好一定順序與方向。

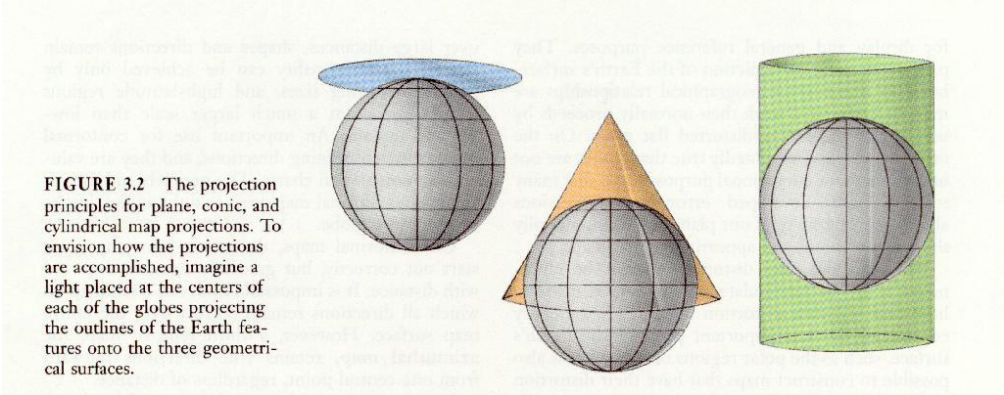


FIGURE 3.2 The projection principles for plane, conic, and cylindrical map projections. To envision how the projections are accomplished, imagine a light placed at the centers of each of the globes projecting the outlines of the Earth features onto the three geometrical surfaces.

Image source: https://faculty.kutztown.edu/courtney/blackboard/Physical/05Project/project.html