Computer Vision HW3 Report

Student ID: R10521802
Name: 李睿莆

Part 1. (2%)

• Paste the function solve_homography( ) (1%)

Ans:

```python
def solve_homography(u, v):
    """
    This function should return a 3-by-3 homography matrix,
    u, v are N-by-2 matrices, representing N corresponding points for v = T(u)
    :param u: N-by-2 source pixel location matrices
    :param v: N-by-2 destination pixel location matrices
    :return:
    """
    N = u.shape[0]
    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')

    # TODO: 1.forming A
    ux = u[:,0].reshape((N,1))
    uy = u[:,1].reshape((N,1))
    vx = v[:,0].reshape((N,1))
    vy = v[:,1].reshape((N,1))
    vx_eq = np.concatenate((ux, uy, np.ones((N,1)), np.zeros((N,3)), -np.multiply(ux,vx), -np.multiply(uy,vx), -vx), axis=1)
    vy_eq = np.concatenate((np.zeros((N,3)), ux, uy, np.ones((N,1)), -np.multiply(ux,vy), -np.multiply(uy,vy), -vy), axis=1)
    A = np.stack((vx_eq, vy_eq), axis=1).reshape((2*N,9))

    # TODO: 2.solve H with A
    U, S, VT = np.linalg.svd(A)
    H = VT[-1,:].reshape((3,3))
    return H
```

• Paste your warped canvas (1%)

Ans:

圖 1、output1.png

Part 2. (2%)

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

Ans:

```python
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

    if direction == 'b':
        # TODO: 1.meshgrid the (x,y) coordinate pairs
        x, y = np.arange(xmin, w_dst), np.arange(ymin, h_dst)
        xx, yy = np.meshgrid(x, y)

        # TODO: 2.reshape the destination pixels as 3 x N homogeneous coordinate
        vx_row = xx.reshape((1, xx.size))
        vy_row = yy.reshape((1, yy.size))
        one_row = np.ones((1, xx.size))
        V = np.concatenate((vx_row, vy_row, one_row), axis=0)

        # TODO: 3.apply H_inv to the destination pixels and retrieve (u,v) pixels, then reshape to (ymax-ymin),(xmax-xmin)
        H_inv = np.linalg.inv(H)
        U = np.dot(H_inv, V)
        U /= U[2]
        ux = np.round(U[0]).reshape((h_dst-ymin, w_dst-xmin)).astype(np.int)
        uy = np.round(U[1]).reshape((h_dst-ymin, w_dst-xmin)).astype(np.int)

        # TODO: 4.calculate the mask of the transformed coordinate (should not exceed the boundaries of source image)
        # TODO: 5.sample the source image with the masked and reshaped transformed coordinates
        # TODO: 6. assign to destination image with proper masking
        mask = np.logical_and(np.logical_and(ux >= 0, ux < w_src), np.logical_and(uy >= 0, uy < h_src))
        dst[yy[mask], xx[mask]] = src[uy[mask], ux[mask]]

    elif direction == 'f':
        # TODO: 1.meshgrid the (x,y) coordinate pairs
        x, y = np.arange(xmin, xmax), np.arange(ymin, ymax)
        xx, yy = np.meshgrid(x, y)

        # TODO: 2.reshape the source pixels as 3 x N homogeneous coordinate
        ux_row = xx.reshape((1, xx.size))
        uy_row = yy.reshape((1, yy.size))
        one_row = np.ones((1, xx.size))
        U = np.concatenate((ux_row, uy_row, one_row), axis=0)

        # TODO: 3.apply H to the source pixels and retrieve (u,v) pixels, then reshape to (ymax-ymin),(xmax-xmin)
        V = np.dot(H, U)
        V /= V[2]
        vx = np.round(V[0].reshape((ymax-ymin, xmax-xmin))).astype(np.int)
        vy = np.round(V[1].reshape((ymax-ymin, xmax-xmin))).astype(np.int)

        # TODO: 4.calculate the mask of the transformed coordinate (should not exceed the boundaries of destination image)
        # TODO: 5.filter the valid coordinates using previous obtained mask
        # TODO: 6. assign to destination image using advanced array indicing
        mask = np.logical_and(np.logical_and(vx >= 0, vx < w_dst), np.logical_and(vy >= 0, vy < h_dst))
        dst[vy[mask], vx[mask]] = src[yy[mask], xx[mask]]

    return dst
```

• Briefly introduce the interpolation method you use (1%)

Ans:

在 backward 和 forward warp 中都以四捨五入並取至整數的方法(nearest neighbor)，解決圖片經過 Homography 產生 sub-pixel 的問題。

Part 3. (8%)

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

Ans:

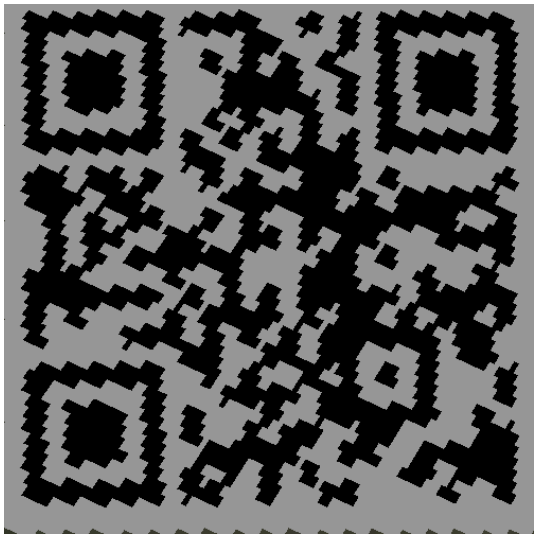

圖 2、output3_1.png

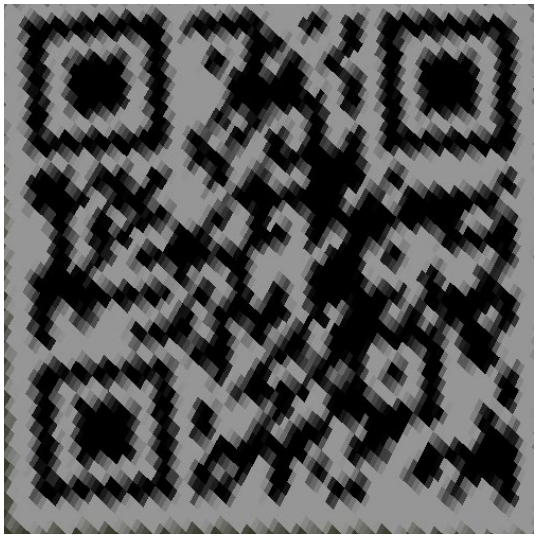

圖 3、output3_2.png

• Discuss the difference between 2 source images, are the warped results the same or different? (3%)

Ans:

圖 4、BL_secret1.png

圖 5、BL_secret1.png

圖 4 為取得圖 2 QR code 轉換的原圖，圖 5 為取得圖 3 QR code 轉換的原圖，可以發現圖 4 在拍攝時影像比例較正常，而圖 5 中有發生一些變形扭曲的現象，類似魚眼鏡頭的感覺。

以圖片轉換結果來說，圖 2 比圖 3 較清晰，用手機掃描也較快可以得到短網址，圖 3 可能由於較模糊的關係，手機要換一下角度才掃到網址，掃到的網址皆為 http://media.ee.ntu.edu.tw/courses/cv/21S/

• If the results are the same, explain why. If the results are different, explain why. (4%)

Ans:

圖 4 中的 QR code 沒有變形問題，較適合以四個角點框起來，因此轉換到 destination image(500x500)時影像清晰度較高，而圖 5 中的 QR code 有些微變形扭曲的情況，以四個角點包圍可能較無法涵蓋完整的 QR code，因此轉換後的影像有失真的問題且較為模糊。

Part 4. (8%)

• Paste your stitched panorama (1%)

Ans:

圖 6、output4.png

• Can all consecutive images be stitched into a panorama? (3%)

Ans:

在拼接影像的時候，影像像對之間需要有重疊的部分才能進行配準，但不是只要有重疊區域的影像就可以完整的進行拼接，通常拼接影像還會需要有相同的投影中心，且拍攝影像時盡量為遠景(沒有前景後景的差異)，使得影像可以視為平面等等。

• If yes, explain your reason. If not, explain under what conditions will result in a failure? (4%)

Ans:

圖 7、parallax error

圖 8、ghost artifacts

Image ref:

https://www.cs.jhu.edu/~misha/ReadingSeminar/Papers/Uyttendaele01.pdf

https://en.wikipedia.org/wiki/Image_stitching#Image_stitching_issues

若兩張影像不是在相同位置進行拍攝(如圖 7)，可能會因為視差(parallax)導致拼接的影像產生錯誤，而在近景攝影時，如室內攝影，近距離拍攝物體將使得視差的影響更大。另外，影像也會因拍攝時不同的曝光程度或是拍攝到非靜止的物件時(如圖 8)，使得在進行拼接時可能會產生偽影(Exposure Artifacts / Ghosting artifact)的現象。