Computer Vision HW3 Report

Student ID: R10522606
Name: 曾柏翔

Part 1. Homography Estimation

• Paste the function solve_homography( ).

```python
def solve_homography(u, v):
    N = u.shape[0]

    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, uy = u[:, 0].reshape((N, 1)), u[:, 1].reshape((N, 1))
    vx, vy = v[:, 0].reshape((N, 1)), v[:, 1].reshape((N, 1))

    A1 = np.concatenate(
        (ux, uy, np.ones((N, 1)), np.zeros((N, 3)), -np.multiply(ux, vx), -np.multiply(uy, vx), -vx), axis=1)
    A2 = 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.concatenate((A1, A2), axis=0)

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

    return H
```

• Paste your warped canvas.

Part 2. Marker-Based Planar AR

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

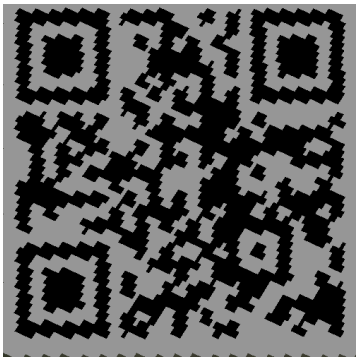
```python
def warping(src, dst, H, ymin, ymax, xmin, xmax, direction='b'):
    h_src, w_src, ch = src.shape
    h_dst, w_dst, ch = dst.shape
    H_inv = np.linalg.inv(H)

    # TODO: 1.meshgrid the (x,y) coordinate pairs
    xc, yc = np.meshgrid(np.arange(xmin, xmax, 1), np.arange(ymin, ymax, 1))
    dst_coor = np.stack((xc, yc), axis=-1)

    # TODO: 2.reshape the destination pixels as N x 3 homogeneous coordinate
    dst_coor = dst_coor.reshape((-1, 2))
    dst_coor = np.concatenate((dst_coor, np.ones((dst_coor.shape[0], 1))), axis=1)

    if direction == 'b':
        # TODO: 3.apply H_inv to the dest pixels and retrieve (u,v) pixels, then reshape to (ymax-ymin),(xmax-xmin)
        src_coor = np.dot(H_inv, dst_coor.T)
        src_coor = np.divide(src_coor, src_coor[-1, :])
        src_x = np.rint(src_coor[0, :].reshape((ymax - ymin, xmax - xmin))).astype(int)
        src_y = np.rint(src_coor[1, :].reshape((ymax - ymin, xmax - xmin))).astype(int)

        # TODO: 4.calculate the mask of the transformed coordinate (should not exceed the boundaries of source image)
        mask = ((0 < src_y) * (src_y < h_src)) * ((0 < src_x) * (src_x < w_src))

        # TODO: 5.sample the source image with the masked and reshaped transformed coordinates

        # TODO: 6. assign to destination image with proper masking
        dst[yc[mask], xc[mask]] = src[src_y[mask], src_x[mask]]

    elif direction == 'f':
        # TODO: 3.apply H to the source pixels and retrieve (u,v) pixels, then reshape to (ymax-ymin),(xmax-xmin)
        src_coor = np.dot(H, dst_coor.T)
        src_coor = np.divide(src_coor, src_coor[-1, :])
        dst_x = np.rint(src_coor[0, :].reshape(ymax - ymin, xmax - xmin)).astype(int)
        dst_y = np.rint(src_coor[1, :].reshape(ymax - ymin, xmax - xmin)).astype(int)

        # TODO: 4.calculate the mask of the transformed coordinate (should not exceed the boundaries of dst image)

        # TODO: 5.filter the valid coordinates using previous obtained mask

        # TODO: 6. assign to destination image using advanced array indexing
        dst[np.clip(dst_y, 0, h_dst - 1), np.clip(dst_x, 0, w_dst - 1)] = src

    return dst
```

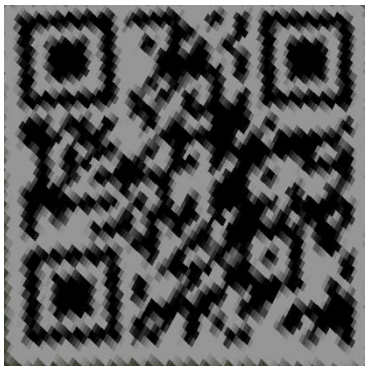
• Briefly introduce the interpolation method you use.

我使用了「最鄰近插值法」，如果變換座標不是整数，那麼就透過 numpy.rint()取最接近的整數，再利用 astype(int)轉換成整數。

Part 3. Unwarp the Secret

• Paste the 2 warped QR code and the link you find.





https://qrgo.page.link/jc2Y9 http://media.ee.ntu.edu.tw/courses/cv/21S/

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

兩張 QR code 大致上是一樣的，只有差在兩張的清晰程度不同，BL_secret1 比較清晰，BL_secret2 比較模糊，但皆能掃描出網站，都是 Spring 2021 CV 課程網站。

• If the results are the same, explain why. If the results are different, explain why.

可以看原始圖，BL_secret1 是屬於比較方正的 QR code，而 BL_secret2 則是有稍微曲線的感覺。因此在還原的時候，可能在座標轉換對應上會造成不精確，或是損失部分資訊。

Part 4. Panorama

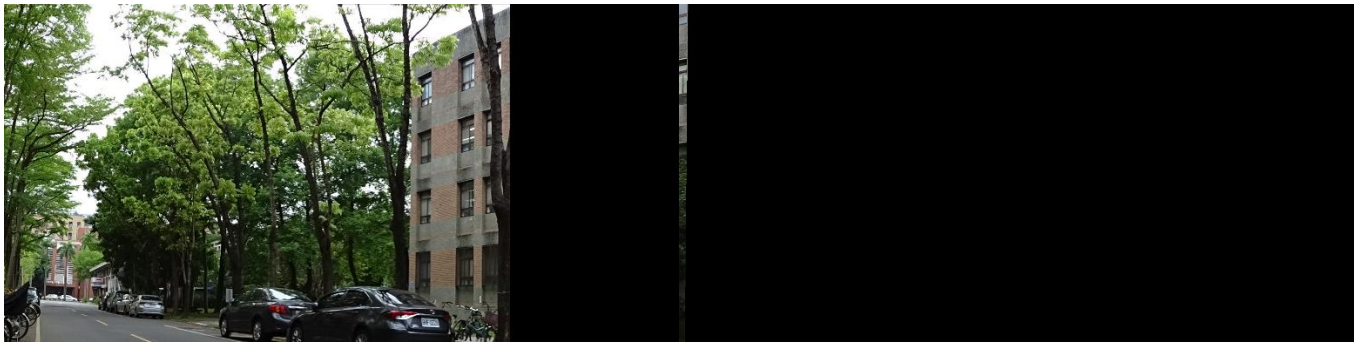
• Paste your stitched panorama.

• Can all consecutive images be stitched into a panorama?

這邊我做了幾個嘗試，首先是 hw3 所提供的三張圖片拼接，是能夠順利調整一些參數而達到完好的全景圖。

接著我嘗試了將最後一張複製一次後，進行 4 張圖拼接，結果依然能夠得到一張完好的全景圖。

再來我則使用了自行拍攝的照片來進行全景圖測試，結果出現了有部分圖像無法顯示的狀況。



最後我將原始圖 crop 後，來模擬拍照時若是有上下平移的狀況，結果可以明顯看到，雖然圖片連接處完美的接合了，但也造成圖片變形的狀況。

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

有些照片成功，有些失敗，推測原因是因為投影面的問題，由於投影面為一平面，因此若是連續照片超過 180°，則會造成無法投影至投影面上，而是投影到無窮遠處，進而產生失真的現象。

若是使用圓柱投影則可拓展到 360°，下圖為使用圓柱投影後，所改善的結果。