Computer Vision HW3 Report

Student ID: R13943124
Name: 施伯儒

Part 1.

- **Paste your warped canvas**

Part 2.

- **Paste the function code *solve_homography(u, v)* & *warping( )* (both forward & backward)**

```
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
    """
    two equations:
        h11*ux + h12*uy + h13   + h21*0  + h22*0  + h23*0 -h31*ux*vx -h32*uy*vx -h33*vx = 0
        h11*0  + h12*0  + h13*0 + h21*ux + h22*uy + h23   -h31*ux*vy -h32*uy*vy -h33*vy = 0
    """
    A = []
    for i in range(N):
        A.append([u[i][0], u[i][1], 1, 0, 0, 0, -u[i][0]*v[i][0], -u[i][1]*v[i][0], -v[i][0]])
        A.append([0, 0, 0, u[i][0], u[i][1], 1, -u[i][0]*v[i][1], -u[i][1]*v[i][1], -v[i][1]])

    # TODO: 2.solve H with A
    """
        A*h = 0
        solve h --> finding NULL space of A
    """

    A = np.array(A)
    _, _, v_t = np.linalg.svd(A)
    H = v_t[-1].reshape(3,3)

    return H
```

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

    if direction == 'b':
        x = np.arange(0, w_dst, 1)
        y = np.arange(0, h_dst, 1)
        xx, yy = np.meshgrid(x, y)
        xx, yy = xx.flatten()[:, np.newaxis], yy.flatten()[:, np.newaxis]
        ones = np.ones((len(xx), 1))
        des_coor = np.concatenate((xx, yy, ones), axis=1).astype(np.int)
        Resource_pixel = H_inv.dot(des_coor.T).T # (N * 3)
        Resource_pixel[:, :2] = Resource_pixel[:, :2] / Resource_pixel[:, 2][:, np.newaxis]
        out_boundary = []

        if (Resource_pixel[:, 0] < 0).any():
            out_boundary += np.where(Resource_pixel[:, 0] < 0)[0].tolist()
        if (Resource_pixel[:, 1] < 0).any():
            out_boundary += np.where(Resource_pixel[:, 1] < 0)[0].tolist()
        if (Resource_pixel[:, 0] > w_src-1).any():
            out_boundary += np.where(Resource_pixel[:, 0] > (w_src -1))[0].tolist()
        if (Resource_pixel[:, 1] > h_src-1).any():
            out_boundary += np.where(Resource_pixel[:, 1] > (h_src - 1))[0].tolist()

        if len(out_boundary):
            Resource_pixel = np.delete(Resource_pixel, out_boundary, 0)
            des_coor = np.delete(des_coor, out_boundary, 0)

        tx = Resource_pixel[:, 0].astype(np.int)
        ty = Resource_pixel[:, 1].astype(np.int)
        dx = Resource_pixel[:, 0] - tx
        dy = Resource_pixel[:, 1] - ty
        ones = np.ones(len(dx)).astype(np.float)
        dst[des_coor[:, 1], des_coor[:, 0]] = ((((ones - dx) * (ones - dy))[:, np.newaxis] * src[ty, tx]) \
                                               + ((dx * (ones - dy))[:, np.newaxis] * src[ty, tx+1]) \
                                               + ((dx * dy)[:, np.newaxis] * src[ty+1, tx+1]) \
                                               + (((ones - dx) * dy)[:, np.newaxis] * src[ty+1, tx]))
```

```python
    elif direction == 'f':
        x = np.arange(0, w_src-1, 1)
        y = np.arange(0, h_src-1, 1)
        xx, yy = np.meshgrid(x, y)
        xx, yy = xx.flatten()[:, np.newaxis], yy.flatten()[:, np.newaxis]
        ones = np.ones((len(xx), 1))
        des_coor = np.concatenate((xx, yy, ones), axis=1).astype(np.int)
        Resource_pixel = H.dot(des_coor.T).T
        Resource_pixel[:, :2] = Resource_pixel[:, :2] / Resource_pixel[:, 2][:, np.newaxis]
        out_boundary = []

        if (Resource_pixel[:, 0] < 0).any():
            out_boundary += np.where(Resource_pixel[:, 0] < 0)[0].tolist()
        if (Resource_pixel[:, 1] < 0).any():
            out_boundary += np.where(Resource_pixel[:, 1] < 0)[0].tolist()
        if (Resource_pixel[:, 0] > w_dst-1).any():
            out_boundary += np.where(Resource_pixel[:, 0] > (w_dst -1))[0].tolist()
        if (Resource_pixel[:, 1] > h_dst-1).any():
            out_boundary += np.where(Resource_pixel[:, 0] > (h_dst - 1))[0].tolist()

        if len(out_boundary):
            Resource_pixel = np.delete(Resource_pixel, out_boundary, 0)
            des_coor = np.delete(des_coor, out_boundary, 0)

        tx = Resource_pixel[:, 0].astype(np.int)
        ty = Resource_pixel[:, 1].astype(np.int)
        dx = Resource_pixel[:, 0] - tx
        dy = Resource_pixel[:, 1] - ty
        ones = np.ones(len(dx)).astype(np.float)
        dst[ty, tx] = ((((ones - dx) * (ones - dy))[:, np.newaxis] * src[des_coor[:, 1], des_coor[:, 0]]) \
                                               + ((dx * (ones - dy))[:, np.newaxis] * src[des_coor[:, 1], des_coor[:, 0]+1]) \
                                               + ((dx * dy)[:, np.newaxis] * src[des_coor[:, 1]+1, des_coor[:, 0]+1]) \
                                               + (((ones - dx) * dy)[:, np.newaxis] * src[des_coor[:, 1]+1, des_coor[:, 0]]))

    return dst
```

- **Briefly introduce the interpolation method you use**

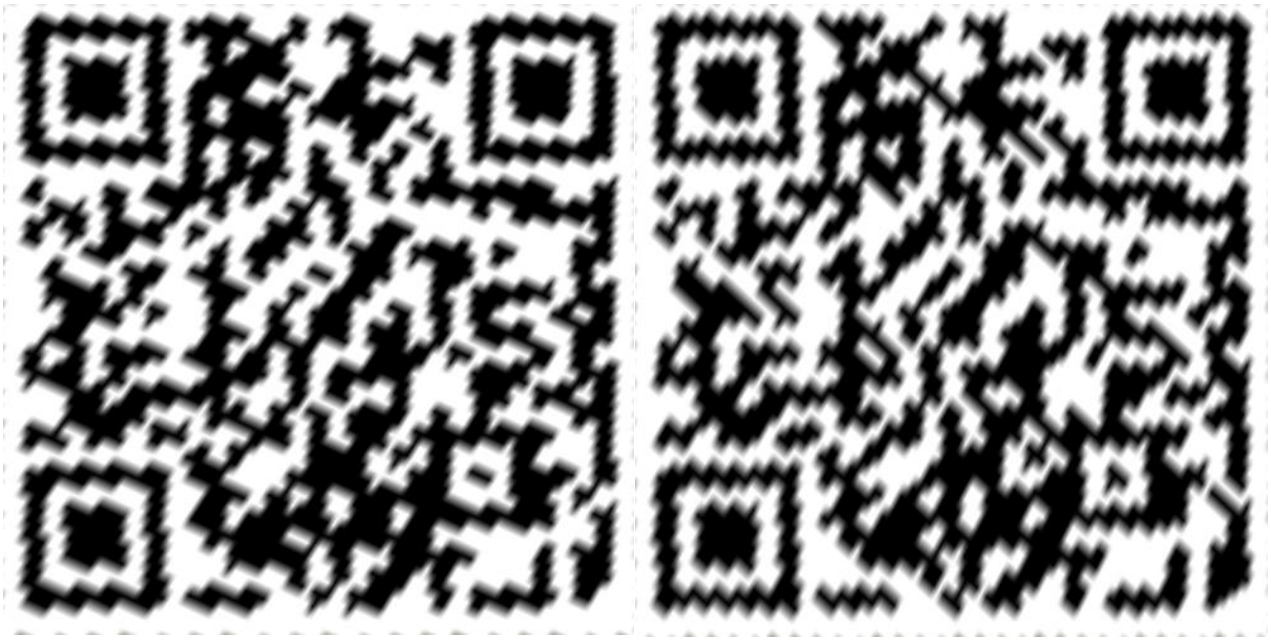
我使用的是 bilinear interpolation，如下所示:

```
dst[des_coor[:, 1], des_coor[:, 0]] = ((((ones - dx) * (ones - dy))[:, np.newaxis] * src[ty, tx]) \
                                     + ((dx * (ones - dy))[:, np.newaxis] * src[ty, tx+1]) \
                                     + ((dx * dy)[:, np.newaxis] * src[ty+1, tx+1]) \
                                     + (((ones - dx) * dy)[:, np.newaxis] * src[ty+1, tx]))
```

Part 3.

- **Paste the 2 warped images and the link you find**

Link: http://media.ee.ntu.edu.tw/courses/cv/25S/



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

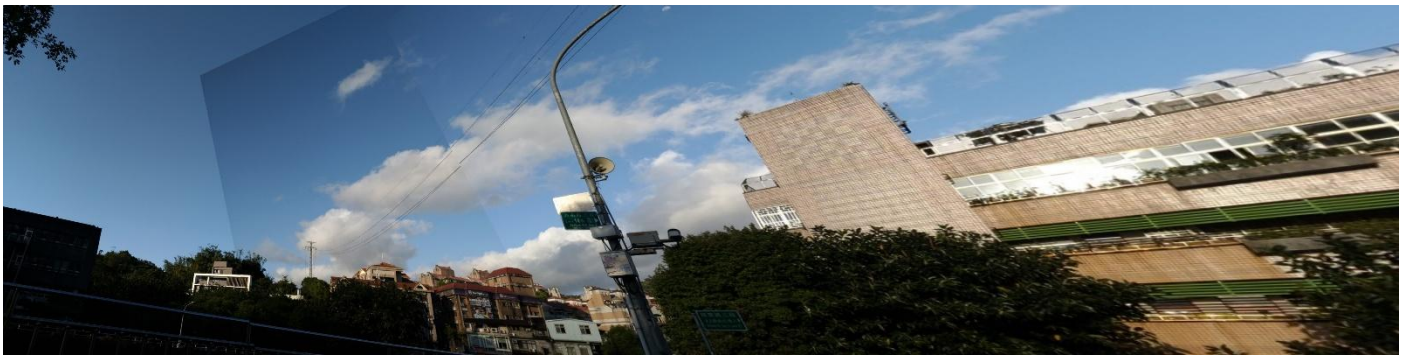
不太一樣，左邊比右邊清晰

- **If the results are the same, explain why. If the results are different, explain why?**

左圖的原始圖片是平面的，但是右圖的原始圖片有彎曲，造成資訊在 warping 時有經過壓縮，相較左圖完全平面而言會丟失一些資訊，所以比較模糊

Part 4.

- **Paste your stitched panorama**



- **Can all consecutive images be stitched into a panorama?**

不能

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

如果兩張圖差很多，以至於沒有任何重疊或相似的特徵，可能會找不到 matched points 去拼成 panorama