HW4 report

姓名：林子敬

學號：R09246003

Email：r09246003@ntu.edu.tw

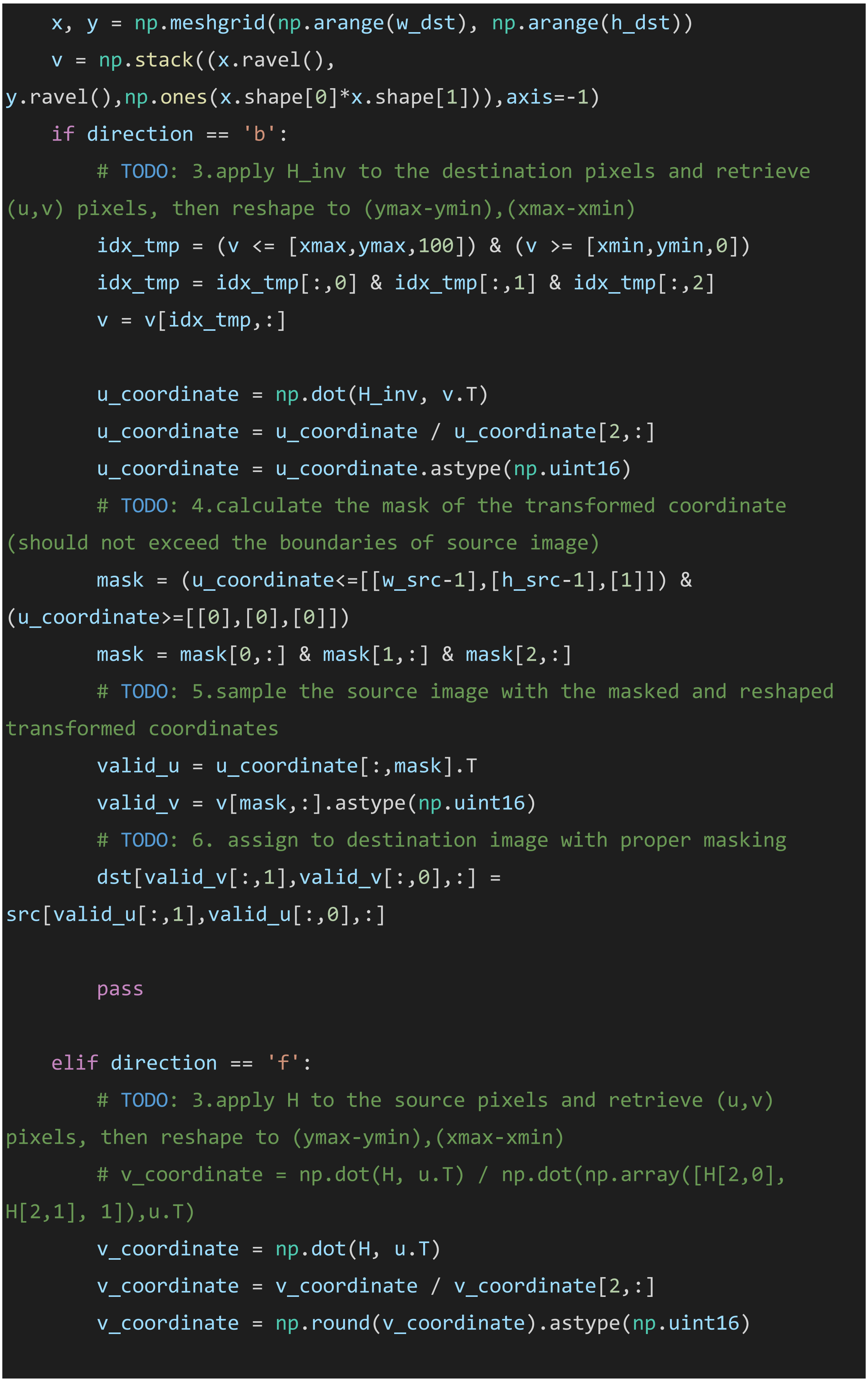
Part 1.

```
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
    u00, u01, u10, u11, u20, u21, u30, u31 = u[0,0], u[0,1], u[1,0], u[1,1], u[2,0], u[2,1], u[3,0], u[3,1]
    v00, v01, v10, v11, v20, v21, v30, v31 = v[0,0], v[0,1], v[1,0], v[1,1], v[2,0], v[2,1], v[3,0], v[3,1]
    A = np.array([[u00, u01, 1, 0, 0, 0, -u00*v00, -u01*v00],
        [0, 0, 0, u00, u01, 1, -u00*v01, -u01*v01],
        [u10, u11, 1, 0, 0, 0, -u10*v10, -u11*v10],
        [0, 0, 0, u10, u11, 1, -u10*v11, -u11*v11],
        [u20, u21, 1, 0, 0, 0, -u20*v20, -u21*v20],
        [0, 0, 0, u20, u21, 1, -u20*v21, -u21*v21],
        [u30, u31, 1, 0, 0, 0, -u30*v30, -u31*v30],
        [0, 0, 0, u30, u31, 1, -u30*v31, -u31*v31]])

    # TODO: 2.solve H with A
    x = np.linalg.solve(A, v.ravel())
    H = np.concatenate((x,np.array([1]))).reshape(3,3)
    return H
```

Part 2.

```
def warping(src, dst, H, ymin, ymax, xmin, xmax, direction='b'):
    """
    Perform forward/backward warpping without for loops. i.e.
    for all pixels in src(xmin~xmax, ymin~ymax),  warp to destination
          (xmin=0,ymin=0)  source                       destination
                         |--------|              |------------------------
-|
                         |        |              |                        |
                         |        |     warp     |                        |
    forward warp         |        |  ---------
>  |                        |
                         |        |              |                        |
                         |--------|              |------------------------
-|
                                 (xmax=w,ymax=h)

    for all pixels in dst(xmin~xmax, ymin~ymax),  sample from source
                            source                      destination
```

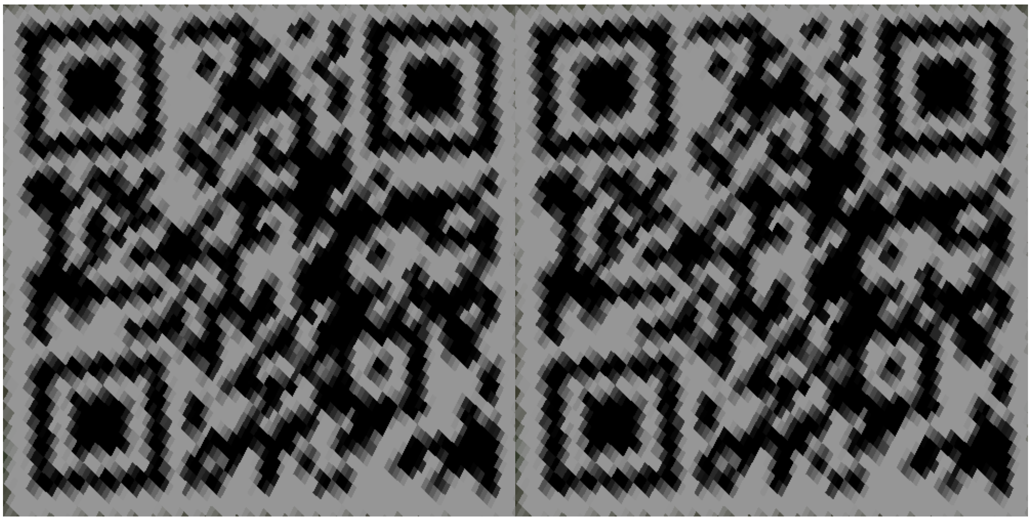
```
                        |--------|              |-----------------------
-|
                        |        |              |
(xmin,ymin)            |
                        |        |    warp      |          |--
|         |
    backward warp        |        |  <---------
-  |           |__|         |
                        |        |              |            (xmax,ymax)
|
                        |--------|              |-----------------------
-|

    :param src: source image
    :param dst: destination output image
    :param H:
    :param ymin: lower vertical bound of the destination(source, if
forward warp) pixel coordinate
    :param ymax: upper vertical bound of the destination(source, if
forward warp) pixel coordinate
    :param xmin: lower horizontal bound of the destination(source, if
forward warp) pixel coordinate
    :param xmax: upper horizontal bound of the destination(source, if
forward warp) pixel coordinate
    :param direction: indicates backward warping or forward warping
    :return: destination output image
    """

    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
    x, y = np.meshgrid(np.arange(w_src), np.arange(h_src))
    u = np.stack((x.ravel(),
y.ravel(),np.ones(x.shape[0]*x.shape[1])),axis=-1)
    # TODO: 2.reshape the destination pixels as N x 3 homogeneous
coordinate
```



```
    x, y = np.meshgrid(np.arange(w_dst), np.arange(h_dst))
    v = np.stack((x.ravel(),
y.ravel(),np.ones(x.shape[0]*x.shape[1])),axis=-1)
    if direction == 'b':
        # TODO: 3.apply H_inv to the destination pixels and retrieve
(u,v) pixels, then reshape to (ymax-ymin),(xmax-xmin)
        idx_tmp = (v <= [xmax,ymax,100]) & (v >= [xmin,ymin,0])
        idx_tmp = idx_tmp[:,0] & idx_tmp[:,1] & idx_tmp[:,2]
        v = v[idx_tmp,:]

        u_coordinate = np.dot(H_inv, v.T)
        u_coordinate = u_coordinate / u_coordinate[2,:]
        u_coordinate = u_coordinate.astype(np.uint16)
        # TODO: 4.calculate the mask of the transformed coordinate
(should not exceed the boundaries of source image)
        mask = (u_coordinate<=[[w_src-1],[h_src-1],[1]]) &
(u_coordinate>=[[0],[0],[0]])
        mask = mask[0,:] & mask[1,:] & mask[2,:]
        # TODO: 5.sample the source image with the masked and reshaped
transformed coordinates
        valid_u = u_coordinate[:,mask].T
        valid_v = v[mask,:].astype(np.uint16)
        # TODO: 6. assign to destination image with proper masking
        dst[valid_v[:,1],valid_v[:,0],:] =
src[valid_u[:,1],valid_u[:,0],:]

        pass

    elif direction == 'f':
        # TODO: 3.apply H to the source pixels and retrieve (u,v)
pixels, then reshape to (ymax-ymin),(xmax-xmin)
        # v_coordinate = np.dot(H, u.T) / np.dot(np.array([H[2,0],
H[2,1], 1]),u.T)
        v_coordinate = np.dot(H, u.T)
        v_coordinate = v_coordinate / v_coordinate[2,:]
        v_coordinate = np.round(v_coordinate).astype(np.uint16)
        # 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
        color = src.reshape((src.shape[0]*src.shape[1],3))
        dst[v_coordinate[1,:], v_coordinate[0,:], :] = color
        pass


    return dst
```

這裡我都直接把座標轉成 uint16 也就是無條件捨去，有點像是 nearesrt neighbor 但我是無條件捨去等於又犧牲一點精準度。

Part 3.



Link: http://media.ee.ntu.edu.tw/courses/cv/21S/

兩個 source images 的不同之處在於他們拍攝角度的不同。

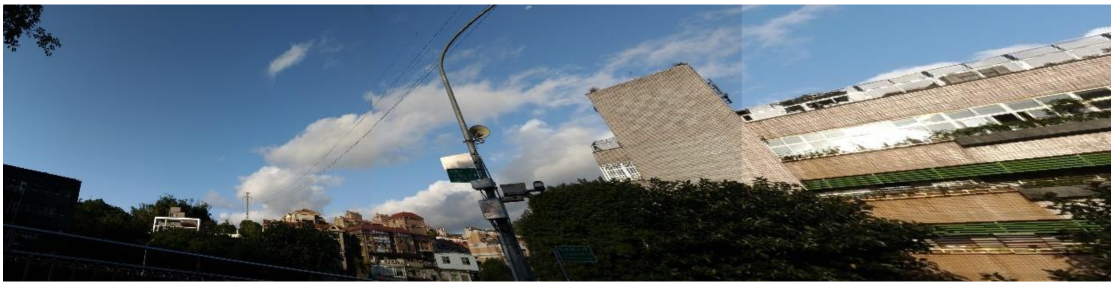
而兩個 warped result 經過

```
print(np.array_equal(output3_1, output3_2))
```

得到的結果是 True 可見他們是一樣的。

原因是他們都是從一個較小解析度的地方有點像是 upsampling 拿出來且我的插值方式是把座標無條件捨去到整數再取顏色，也就是因為這樣才會如上圖有點鋸齒狀的感覺。

Part 4.



不一定所有連續的影像都可以貼在一起，一個原因是如同這個例子影像間的亮暗程度不同使他們 match 的不好，或是不同的影像旋轉太嚴重等等，還有一個原因是要是拍攝時影像裡有會動的東西結果會有很多殘影。