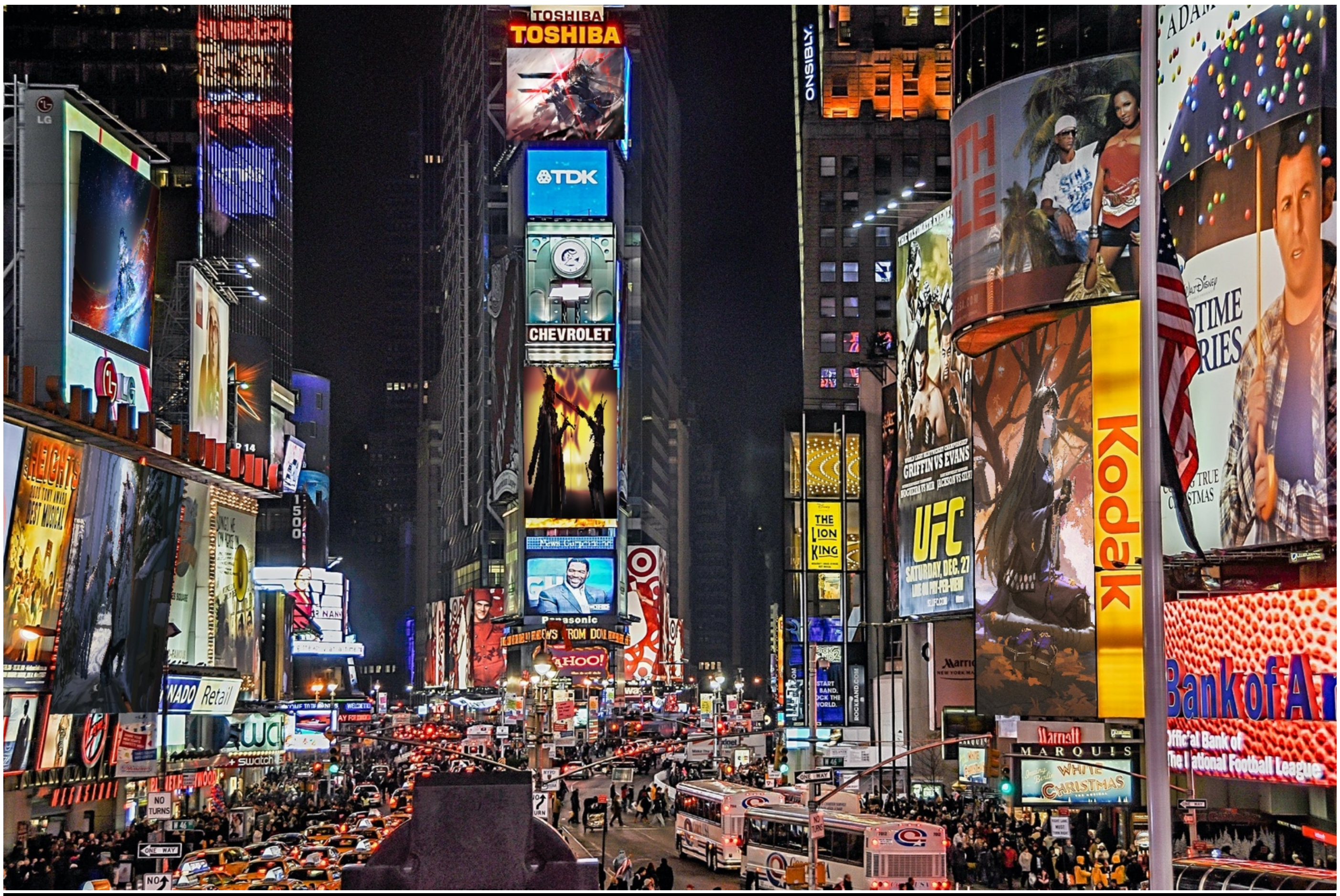
Computer Vision HW3 Report

Student ID: R10943131

Name: 周奕節

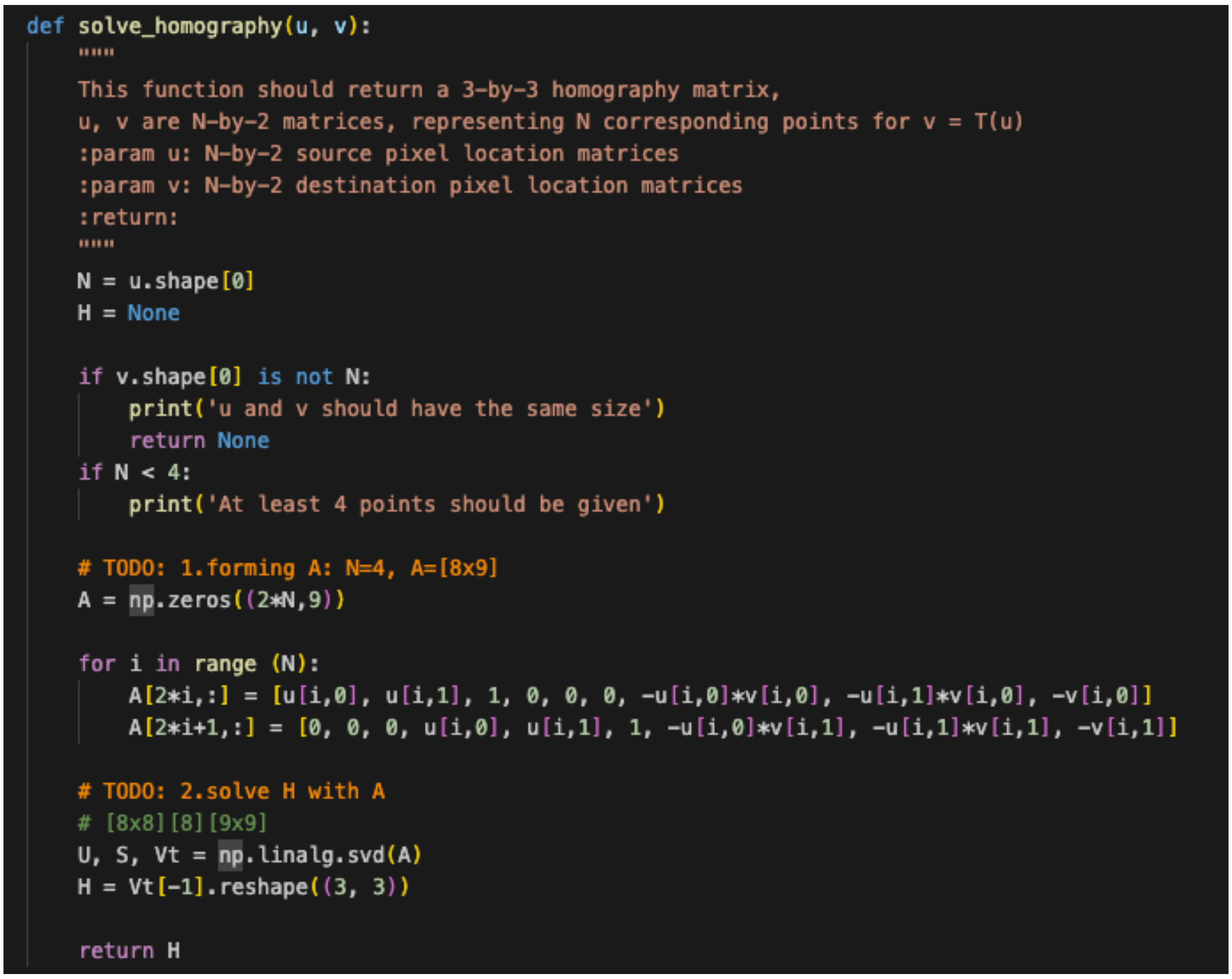
Part 1.

- **Paste your warped canvas**

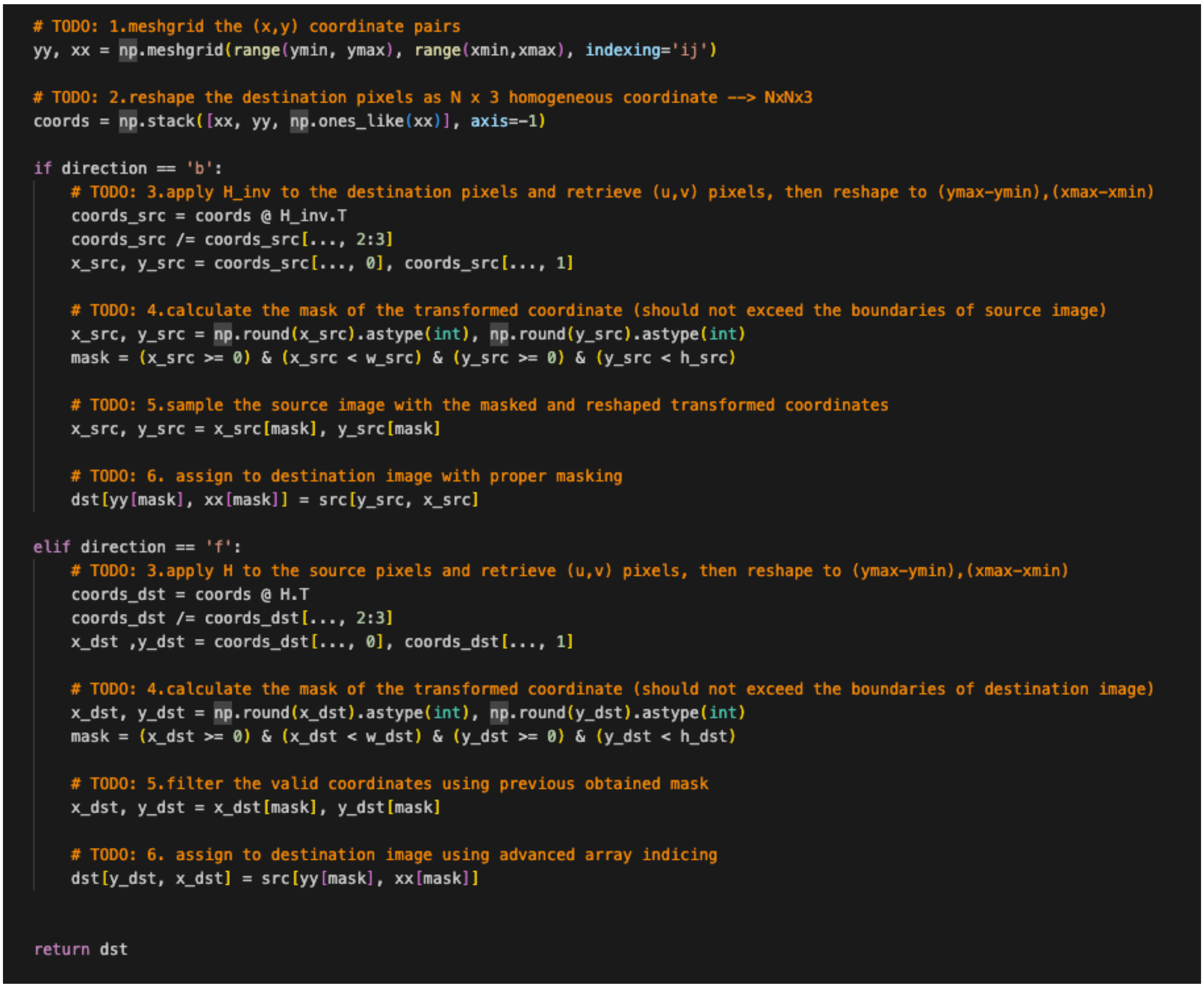


Part 2.

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



```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: N=4, A=[8x9]
    A = np.zeros((2*N,9))

    for i in range (N):
        A[2*i,:] = [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[2*i+1,:] = [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
    # [8x8][8][9x9]
    U, S, Vt = np.linalg.svd(A)
    H = Vt[-1].reshape((3, 3))

    return H
```



```python
# TODO: 1.meshgrid the (x,y) coordinate pairs
yy, xx = np.meshgrid(range(ymin, ymax), range(xmin,xmax), indexing='ij')

# TODO: 2.reshape the destination pixels as N x 3 homogeneous coordinate --> NxNx3
coords = np.stack([xx, yy, np.ones_like(xx)], 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)
    coords_src = coords @ H_inv.T
    coords_src /= coords_src[..., 2:3]
    x_src, y_src = coords_src[..., 0], coords_src[..., 1]

    # TODO: 4.calculate the mask of the transformed coordinate (should not exceed the boundaries of source image)
    x_src, y_src = np.round(x_src).astype(int), np.round(y_src).astype(int)
    mask = (x_src >= 0) & (x_src < w_src) & (y_src >= 0) & (y_src < h_src)

    # TODO: 5.sample the source image with the masked and reshaped transformed coordinates
    x_src, y_src = x_src[mask], y_src[mask]

    # TODO: 6. assign to destination image with proper masking
    dst[yy[mask], xx[mask]] = src[y_src, x_src]

elif direction == 'f':
    # TODO: 3.apply H to the source pixels and retrieve (u,v) pixels, then reshape to (ymax-ymin),(xmax-xmin)
    coords_dst = coords @ H.T
    coords_dst /= coords_dst[..., 2:3]
    x_dst ,y_dst = coords_dst[..., 0], coords_dst[..., 1]

    # TODO: 4.calculate the mask of the transformed coordinate (should not exceed the boundaries of destination image)
    x_dst, y_dst = np.round(x_dst).astype(int), np.round(y_dst).astype(int)
    mask = (x_dst >= 0) & (x_dst < w_dst) & (y_dst >= 0) & (y_dst < h_dst)

    # TODO: 5.filter the valid coordinates using previous obtained mask
    x_dst, y_dst = x_dst[mask], y_dst[mask]

    # TODO: 6. assign to destination image using advanced array indicing
    dst[y_dst, x_dst] = src[yy[mask], xx[mask]]


return dst
```

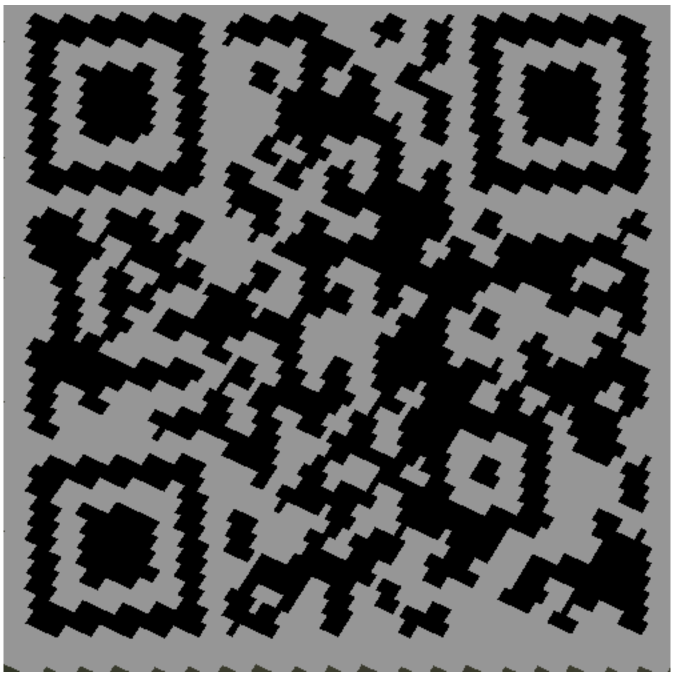
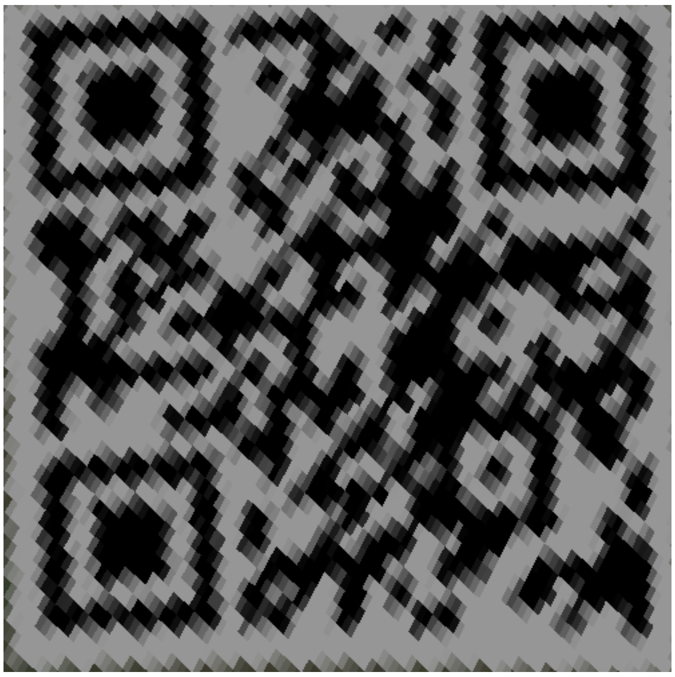
- **Briefly introduce the interpolation method you use**

  Simply use np.round() to round the elements in the array to the nearest integer.

Part 3.

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

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

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

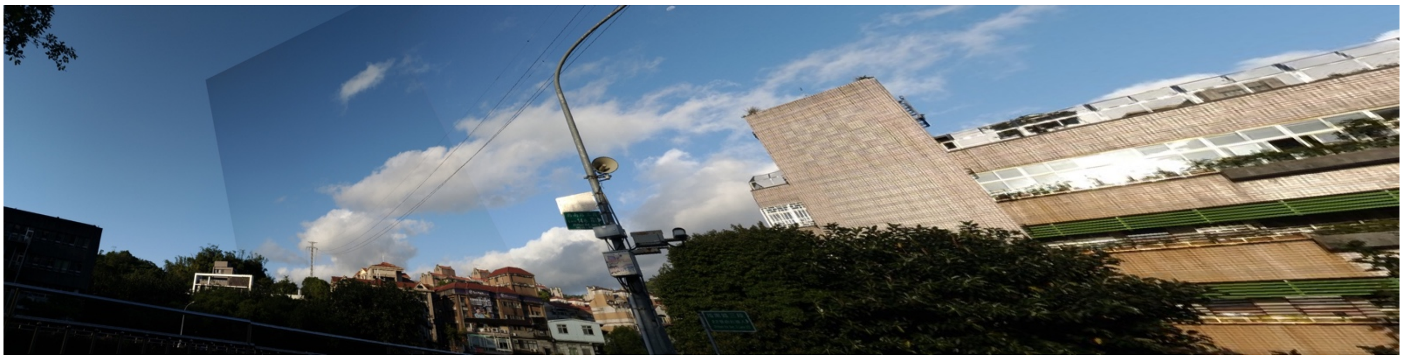
The main difference between the 2 source images is that the second image is a little curved, and it's taken from smartphone.
The warped results are different, although both of them half same shape, the first one is more clear and the second one is more blurred.

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

The results are different due to the second source images have lens distortion which warped the QR code, and the distortion occurred when taking wide-angle photos using a smartphone. Because warping and unwarping images can introduce blurriness, so it's less sharp after warping back to the original shape.

Part 4.

- **Paste your stitched panorama**



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

No

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

We may fail to stitch consecutive images into a panorama when consecutive photos have insufficient overlap or the camera isn't rotated around its optical center.