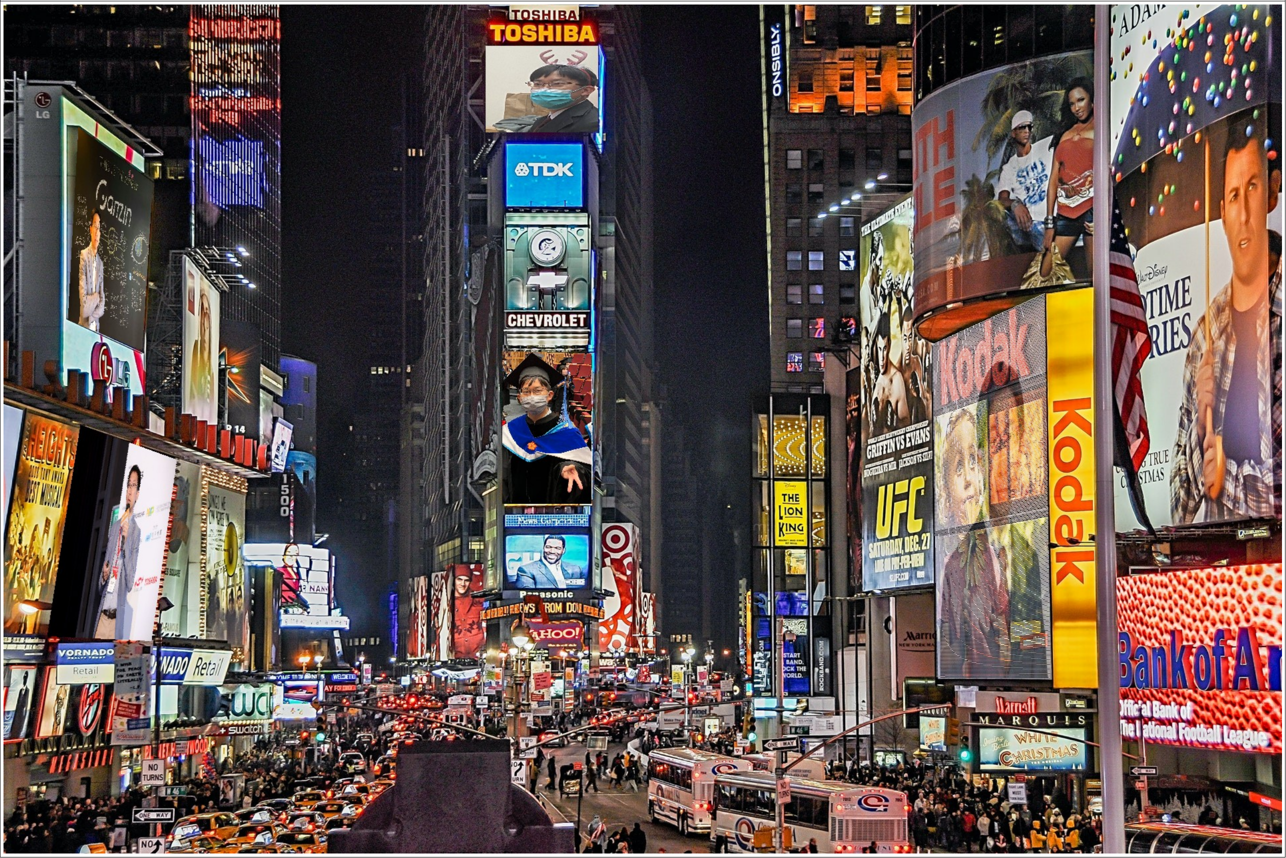
Computer Vision HW3 Report

Student ID: D11921B09
Name: 徐子程

Part 1.

Warped canvas
 TOSHIBA TDK CHEVROLET Kodak UFC THE LION KING Bank of A

Part 2.

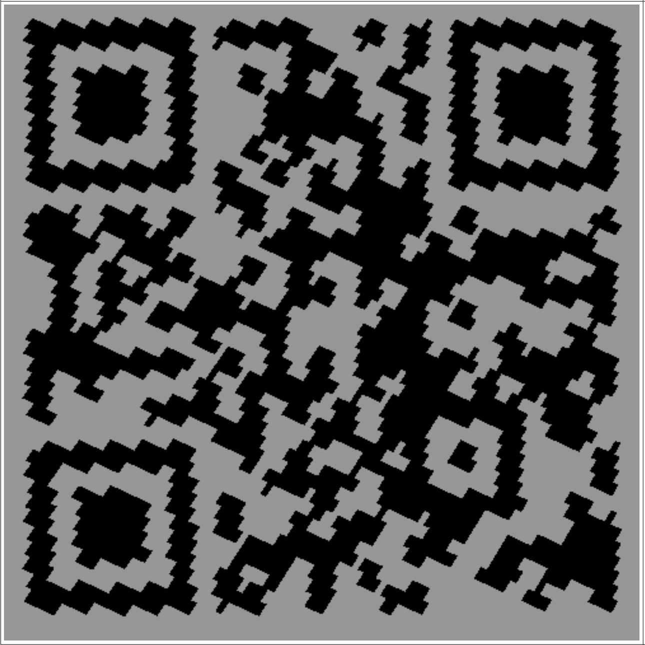
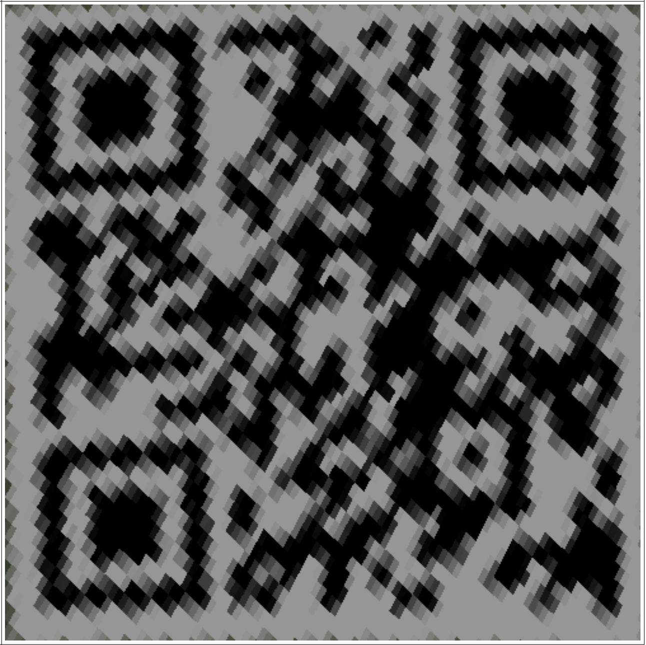
- Paste the function code `solve_homography(u, v)` & `warping( )` (both forward & backward)

```
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')

    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]]

    _, _, vt = np.linalg.svd(A)

    H = vt[-1].reshape(3, 3) # H is the last column of Vt

    return H
```

```
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)

    x = np.arange(xmin, xmax)
    y = np.arange(ymin, ymax)
    xv, yv = np.meshgrid(x, y)

    xv = xv.flatten()
    yv = yv.flatten()
    one_px = np.ones(xv.shape)

    new_px = np.array([xv, yv, one_px])

    if direction == 'b':
        new_src_px = np.dot(H_inv, new_px)
        new_src_px = (new_src_px/new_src_px[-1,:])

        mask = (new_src_px[0,:] >= 0) & (new_src_px[0,:] < w_src) & (new_src_px[1,:] >=
0) & (new_src_px[1,:] < h_src)

        val_src_x = new_src_px[0,:][mask].astype(int)
        val_src_y = new_src_px[1,:][mask].astype(int)
        val_dst_x = new_px[0,:][mask].astype(int)
        val_dst_y = new_px[1,:][mask].astype(int)

    elif direction == 'f':
        new_dst_px = np.dot(H, new_px)
        new_dst_px = (new_dst_px/new_dst_px[-1, :]).astype(int)

        mask = (new_dst_px[0,:] >= 0) & (new_dst_px[0,:] < w_dst) & (new_dst_px[1,:] >=
0) & (new_dst_px[1,:] < h_dst)

        val_src_x = new_px[0,:][mask].astype(int)
        val_src_y = new_px[1,:][mask].astype(int)
        val_dst_x = new_dst_px[0,:][mask].astype(int)
        val_dst_y = new_dst_px[1,:][mask].astype(int)

    dst[val_dst_y, val_dst_x] = src[val_src_y, val_src_x]

    return dst
```

- Briefly introduce the interpolation method you use:
    - I applied rounding to nearest neighbor method.

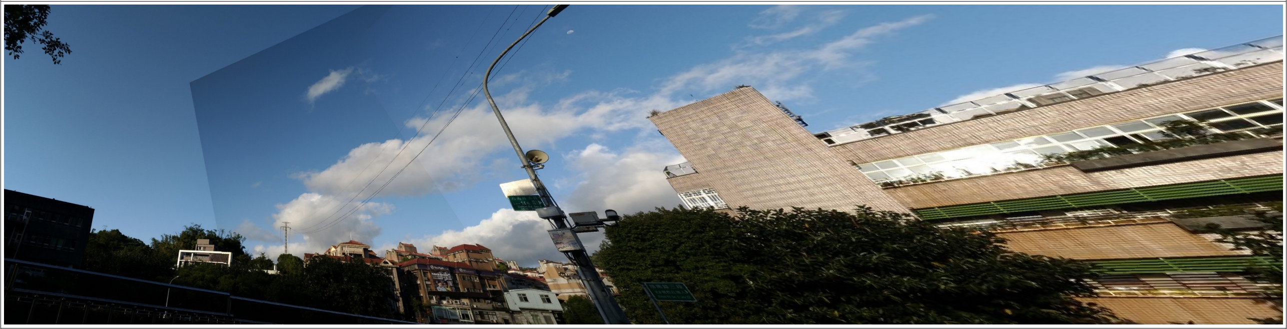
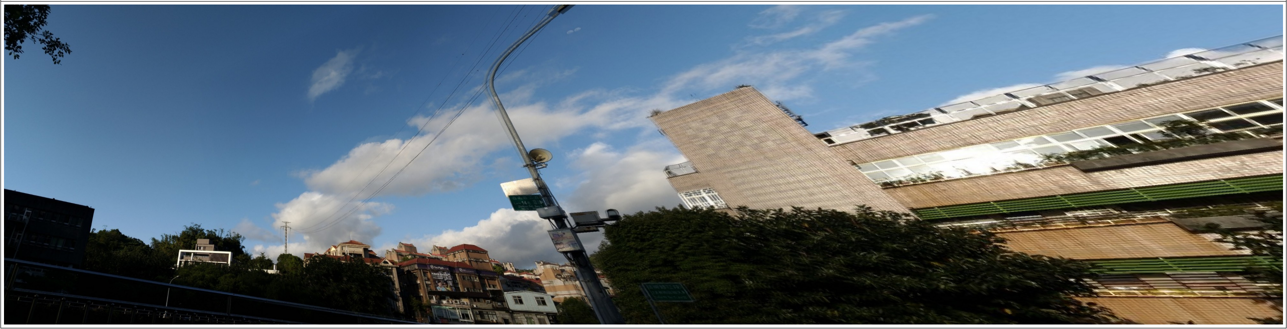
Part 3.

Warped image 1	Warped image 2
	

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

- The `BL_secret1` is undistorted image, while `BL_secret2` has some distortion effect (fisheye).
- The content in both warped QR codes are the same (https://qrgo.page.link/jc2Y9).
- The 2 warped results are different. Since `BL_secret2` has some non-linear distortion, but the solve_homography and the warping transformation are linear operations. Therefore, the transform cannot completely recovery the QR code (there are some blur at the edges).

Part 4.

Panorama without blending

Panorama with blending


• Can all consecutive images be stitched into a panorama?

- ➢ No. For successful stitching, the two consecutive images must have some overlapping parts for feature detection and matching.
- ➢ We implement the planar projection in this homework, the maximum range of images is 0~180°. If it exceed the range, we may need to apply other projection methods.

• [Bonus] Using blending techniques (e.g. alpha blending) to eliminate boundaries

We applied linear blending on image stitching with the following steps:

1. Record the positions of image 1 and 2 in the canvas respectively, then perform AND operation to find overlapping region. (Fig 1.)
2. Generate the alpha channel mask based on the gradient in the overlapping region.
3. Apply the alpha channel to the stitched image. (Fig 2.)

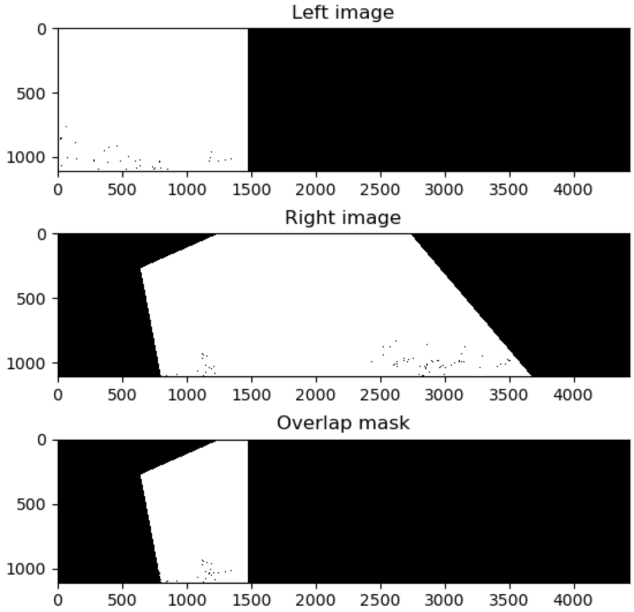


Fig. Image positions in canvas

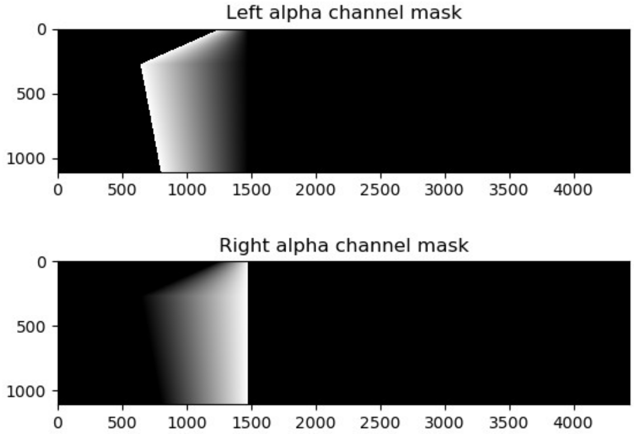


Fig 2. Alpha channel masks