HW 3 Report

陳翰雯 - B08902092

Part 1: Homography Estimation

```
def solve_homography(u, v):

    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
    A = []
    for i in range(N):
        u_x, u_y = u[i]
        v_x, v_y = v[i]

        fir = np.array([u_x, u_y, 1, 0, 0, 0, -u_x*v_x, -u_y*v_x, -v_x])
        sec = np.array([0, 0, 0, u_x, u_y, 1, -u_x*v_y, -u_y*v_y, -v_y])

        A.append(fir)
        A.append(sec)
    A = np.array(A)

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

    # Normalize
    H = (1/H.item(8)) * H

    return H
```

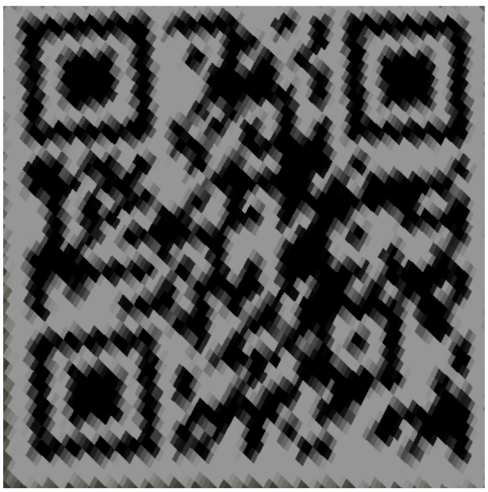
Part 2: Marker-Based Planar AR

```
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
    range_x = np.arange(xmin, xmax, 1)
    range_y = np.arange(ymin, ymax, 1)
    ones = np.ones((ymax-ymin, xmax-xmin))

    x,y = np.meshgrid(range_x, range_y)

    # TODO: 2.reshape the destination pixels as N x 3 homogeneous coordinate
    coordinates = np.stack((x.ravel(),y.ravel(),ones.ravel())) # shape : 3 x N

    if direction == 'b':
        # TODO: 3.apply H_inv to the destination pixels and retrieve (u,v) pixels, then reshape to (ymax-ymin),(xmax-xmin)
        new_pixels = np.matmul(H_inv, coordinates)
        u = np.round((new_pixels[0] / new_pixels[2]).reshape((ymax-ymin, xmax-xmin))).astype(int)
        v = np.round((new_pixels[1] / new_pixels[2]).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 = np.ones(shape=(ymax-ymin, xmax-xmin), dtype=np.bool)
        u_mask = np.where(np.logical_and(u>=0, u<w_src), True, False)
        v_mask = np.where(np.logical_and(v>=0, v<h_src), True, False)

        # TODO: 5.sample the source image with the masked and reshaped transformed coordinates
        mask = np.logical_and(np.logical_and(u_mask, v_mask), mask)

        # TODO: 6. assign to destination image with proper masking
        dst[y[mask], x[mask]] = src[v[mask], u[mask]]

        pass

    elif direction == 'f':
        # TODO: 3.apply H to the source pixels and retrieve (u,v) pixels, then reshape to (ymax-ymin),(xmax-xmin)
```

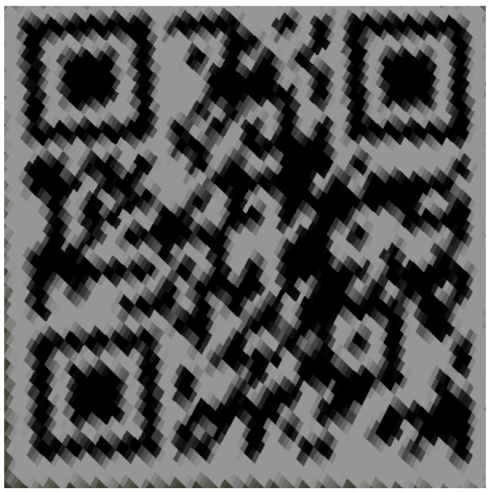
```
        new_pixels = np.matmul(H, coordinates)
        u = np.round((new_pixels[0] / new_pixels[2]).reshape((ymax-ymin, xmax-xmin))).astype(int)
        v = np.round((new_pixels[1] / new_pixels[2]).reshape((ymax-ymin, xmax-xmin))).astype(int)

        # TODO: 4.calculate the mask of the transformed coordinate (should not exceed the boundaries of destination image)
        mask = np.ones(shape=(ymax-ymin, xmax-xmin), dtype=np.bool)
        u_mask = np.where(np.logical_and(u>=0, u<w_dst), True, False)
        v_mask = np.where(np.logical_and(v>=0, v<h_dst), True, False)

        # TODO: 5.filter the valid coordinates using previous obtained mask
        mask = np.logical_and(np.logical_and(u_mask, v_mask), mask)

        # TODO: 6. assign to destination image using advanced array indicing
        dst[v[mask], u[mask]] = src[y[mask], x[mask]]

        pass

    return dst
```

The interpolation method used here is nearest neighbor. The result of matrix multiplication between the homography matrix H and the coordinates are rounded off. In this case the rounded coordinates are then the nearest neighbor of the originally obtained coordinates.

Part 3: Unwarp the secret



output3_1.png



output3_2.png

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

In the 2 source images, the very first visible difference is that second image (BL_secret2.png) is bigger or zoomed. However, if we look at it clearly, we could see that the lines of the building in the second image are not straight. This means, it is possible that the second image is a result of warping image 1.

The results are the same and this could by the assumption I made in the previous explanation that the second image is a result of warping image 1. Since the second image is just a result of warping, hence, both images could be regarded as same image. Moreover, in both images, we could see that both barcodes are complete and can be seen clearly.

The resulting images may be different if the picture is taken from another position or angle. This is because, based on the angle of the camera, the information of the barcode received may be different, it may be incomplete. Hence, may result in different output images.

Part 4: Panorama

Yes, it is because consecutive images will share same features and also their correspondences, hence, it is possible to stitch them together by finding its homography and do warping. Stitching images will fail if they share no same features.