電腦視覺-HW3 Report

R08921053 電機丙研二 梁峻瑋

#Part1: Homography estimation

1. Paste the function code solve_homography (u, v) & your warped canvas:

```
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
    ux = u[:,0].reshape((N,1))
    uy = u[:,1].reshape((N,1))
    vx = v[:,0].reshape((N,1))
    vy = v[:,1].reshape((N,1))

    upA   = np.concatenate( (ux, uy, np.ones((N,1)), np.zeros((N,3)), -1*np.multiply(ux,vx), -1*np.multiply(uy,vx), -1*vx), axis=1 );
    downA = np.concatenate( (np.zeros((N,3)), ux, uy, np.ones((N,1)), -1*np.multiply(ux, vy), -1*np.multiply(uy,vy), -1*vy), axis=1 );
    A     = np.concatenate( (upA, downA), axis=0 );

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

#Part2: Marker Based Planar AR

Paste the function code warping( ) (both forward & backward)

```
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
    # TODO: 2.reshape the destination pixels as N x 3 homogeneous coordinate
    xc, yc = np.meshgrid(np.arange(xmin, xmax, 1), np.arange(ymin, ymax, 1), sparse = False)
    xrow = xc.reshape(( 1,(xmax-xmin)*(ymax-ymin) ))
    yrow = yc.reshape(( 1,(xmax-xmin)*(ymax-ymin) ))
    onerow =  np.ones(( 1,(xmax-xmin)*(ymax-ymin) ))
    M = np.concatenate((xrow, yrow, onerow), axis = 0)

    if direction == 'b':
        # TODO: 3.apply H_inv to the destination pixels and retrieve (u,v) pixels, then resha
        Mbar = np.dot(H_inv,M)
        Mbar = np.divide(Mbar, Mbar[-1,:])
        srcy = np.round( Mbar[1,:].reshape((ymax-ymin, xmax-xmin)) ).astype(int)
        srcx = np.round( Mbar[0,:].reshape((ymax-ymin, xmax-xmin)) ).astype(int)

        # TODO: 4.calculate the mask of the transformed coordinate (should not exceed the bou
        # TODO: 5.sample the source image with the masked and reshaped transformed coordinate
        h_mask = (0<srcy)*(srcy<h_src)
        w_mask = (0<srcx)*(srcx<w_src)
        mask   = h_mask*w_mask

        # TODO: 6. assign to destination image with proper masking
        dst[yc[mask], xc[mask]] = src[srcy[mask], srcx[mask]]

    elif direction == 'f':
        # TODO: 3.apply H to the source pixels and retrieve (u,v) pixels, then reshape to (ym
        Mbar = np.dot(H,M)
        Mbar = np.divide(Mbar, Mbar[-1,:])
        dsty = np.round(Mbar[1,:].reshape(ymax-ymin,xmax-xmin)).astype(int)
        dstx = np.round(Mbar[0,:].reshape(ymax-ymin,xmax-xmin)).astype(int)

        # TODO: 4.calculate the mask of the transformed coordinate (should not exceed the bou
        # TODO: 5.filter the valid coordinates using previous obtained mask
        # TODO: 6. assign to destination image using advanced array indicing
        dst[np.clip(dsty, 0, dst.shape[0]-1), np.clip(dstx, 0, dst.shape[1]-1)] = src

    return dst
```

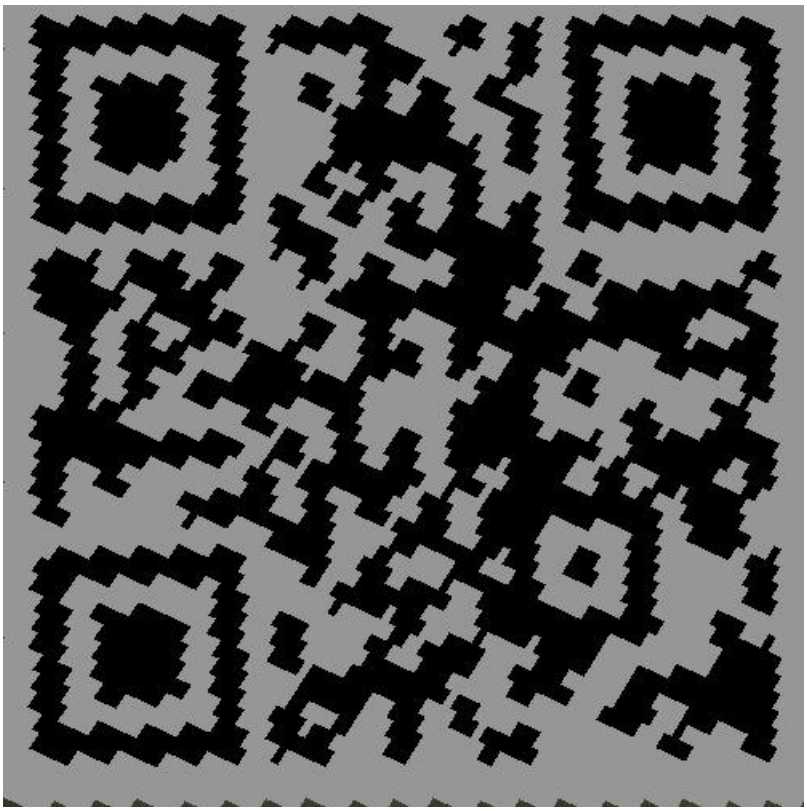
2. Briefly introduce the interpolation method you use

關於 forward 部分，我採取的是 nearest neighbor，也就是用 np.round().astype(int)，取到最接近的整數.

關於 backward 部分，基本上只需要用 mask 將多餘的部分去除就好.

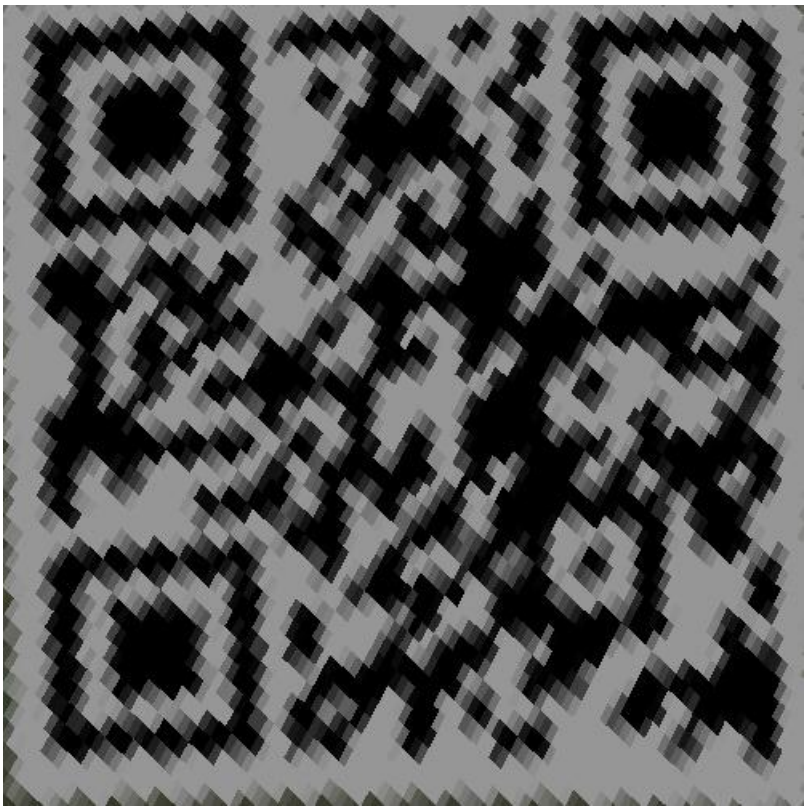
#Part3: Unwarp the secret

1. Paste the 2 warped images



output3_1.png

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



output3_2.png

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

這兩張轉換後圖片顯然是上圖比較清晰，下圖比較模糊！當然，兩者都可以掃描出相同的結果: http://media.ee.ntu.edu.tw/courses/cv/21S/
(也就是我們電腦視覺課的網站首頁)

歸咎其原因，可能是因為第一張源圖的 QR-code 比較方正，長寬比例比較接近，因此很適合還原回來！然而第二張源圖的 QR-code 相當狹長，在轉換的過程中可能損失資訊，因此在相同大小下，會相對模糊.

Part4: Panorama

1. Paste your stitched panorama

2. Can all consecutive images be stitched into a panorama?

以這次作業的三張圖:可以連接，如同上圖所示.
雖然連接處不可能 100%完美，會損失資訊，但肉眼能辨識出三張圖被連接起來.

以任意情況下的連續圖: 如果把全景圖投射到平面，那旋轉角度不能超過 180 度;
如果把全景圖投射到圓柱面，那旋轉角度不能超過 360 度，如底下這個網站所解釋的: [Projections - PTGui Stitching Software]

以全景圖投射到平面的狀況，由於旋轉角 180 度時，兩側恰巧分別投射到兩個無窮遠處，已經到達物理極限! 因此不可能再超過 180 度.

[Bonus] Using homography to produce a “more than 2 images panorama” (3%)
我的程式實作方式並不局限於 2 張或 3 張的圖片，因此可以連接更多圖片！例如，如果我重複利用 frame3，當作 frame4，就可以產生四張圖片的全景圖!!