Part1:Homography estimation

Part2: Marker-Based Planar A

```
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]

    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 = None
    for i in range(N):
        ui = u[i]
        vi = v[i]
        tmp = np.array([[ui[0], ui[1], 1, 0, 0, 0,
-vi[0] * ui[0], -vi[0] * ui[1]], \
                [0, 0, 0, ui[0], ui[1], 1, -vi[1] *
ui[0], -vi[1] * ui[1]]])
        if i == 0:
            A = tmp
            # A = A[:,np.newaxis]
        else:
            A = np.concatenate((A,tmp),axis=0)

    A = np.array(A)
    b = np.array(v).reshape(-1,1)
    # TODO: 2.solve H with A
    if (np.linalg.det(A)):

        H = np.dot(np.linalg.inv(A), b)
    else:
        H = np.dot(np.linalg.pinv(A), b)
    H = np.append(H,1).reshape(3,3)
    return H


def warping(src, dst, H, ymin, ymax, xmin, xmax,
direction='b',use_mask = True):

    mask = None
    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 = np.linspace(xmin, xmax,xmax-xmin)
    y = np.linspace(ymin, ymax,ymax-ymin)
```

```
    mesh = np.array(np.meshgrid(x,y)).reshape(2,-1)
    mesh =
np.concatenate((mesh,np.ones(shape=(1,mesh.shape[1]))),
axis=0)
    # TODO: 2.reshape the destination pixels as N x 3
homogeneous coordinate

    if direction == 'b':

        # TODO: 3.apply H_inv to the destination pixels
and retrieve (u,v) pixels, then reshape to
(ymax-ymin),(xmax-xmin)
        v = np.dot(H_inv, mesh).T
        v = np.divide(v[:, :2], np.tile(v[:, 2], (2,
1)).T)
        v = v.reshape((ymax - ymin), (xmax - xmin), -1)
        v = np.round(v).astype(int)
        # TODO: 4.calculate the mask of the transformed
coordinate (should not exceed the boundaries of source
image)

        # TODO: 5.sample the source image with the
masked and reshaped transformed coordinates
        valid1 = np.logical_and(0 < v[:, :, 0], v[:, :,
0] < w_src)
        valid2 = np.logical_and(0 < v[:, :, 1], v[:, :,
1] < h_src)
        valid = np.logical_and(valid1, valid2)

        # TODO: 6. assign to destination image with
proper masking

        v = v[valid]
        if use_mask == True:
            mask1 = np.zeros(dst.shape[:2])
            mask2 = cv2.cvtColor(dst,
cv2.COLOR_BGR2GRAY)
            mask1[ymin:ymax, xmin:xmax][valid] = 1
```

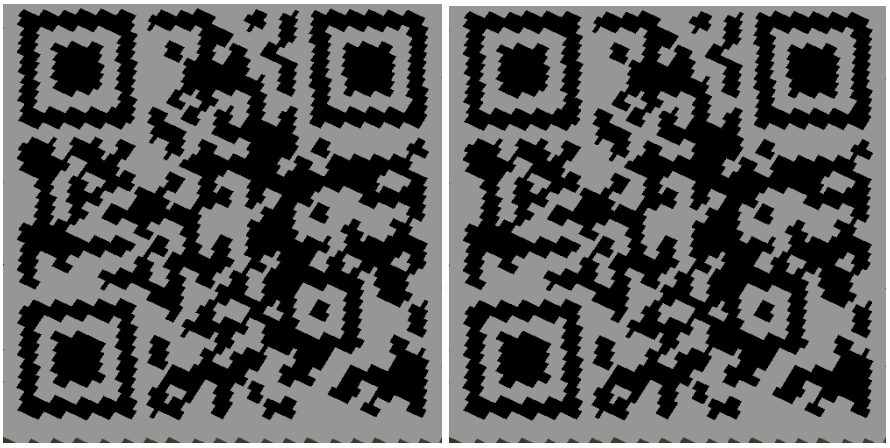
```
            mask = np.logical_and(mask1,
mask2).reshape(mask1.shape)
        dst[ymin:ymax, xmin:xmax][valid] = src[v[...,
1], v[..., 0]]


        return dst, mask
    elif direction == 'f':
        # TODO: 3.apply H to the source pixels and
retrieve (u,v) pixels, then reshape to
(ymax-ymin),(xmax-xmin)
        v = np.dot(H,mesh).T
        v = np.divide(v[:,:2],np.tile(v[:,2],(2,1)).T)
        v = v.reshape((ymax-ymin),(xmax-xmin),-1)
        v= np.round(v).astype(int)
        # 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
        valid1 = np.logical_and(0 < v[:, :, 0] , v[:, :,
0]<w_dst)
        valid2 = np.logical_and(0 < v[:, :, 1] , v[:, :,
1]<h_dst)
        valid = np.logical_and(valid1,valid2)
        v =
np.multiply(np.tile(valid,2).reshape(v.shape),v)
        src =
np.multiply(np.tile(valid,3).reshape(src.shape),src)
        # TODO: 6. assign to destination image using
advanced array indicing
        dst[v[..., 1], v[..., 0]] = src
    return dst
```

interpolation 使用nearest neighbor

Part3: Unwarp the secret



第二張原圖有點畸變第一張則正常，兩張影像warp輸出的結果是一樣的，都是本課程的網頁。雖然第二張原始影像是變型，但經過warping後會將目標範圍的影像拉平，所以結果會與第一張影像的結果一致。

Part4: Panorama

無法將所有影像放入Panorama的影像，由於部份區塊會被切掉

最後，我有嘗試做blending，但是結果不如理想。我是在做warping時就找到重疊的區域，並將重疊區域回傳。接下來，將重疊的部份做linear blending。