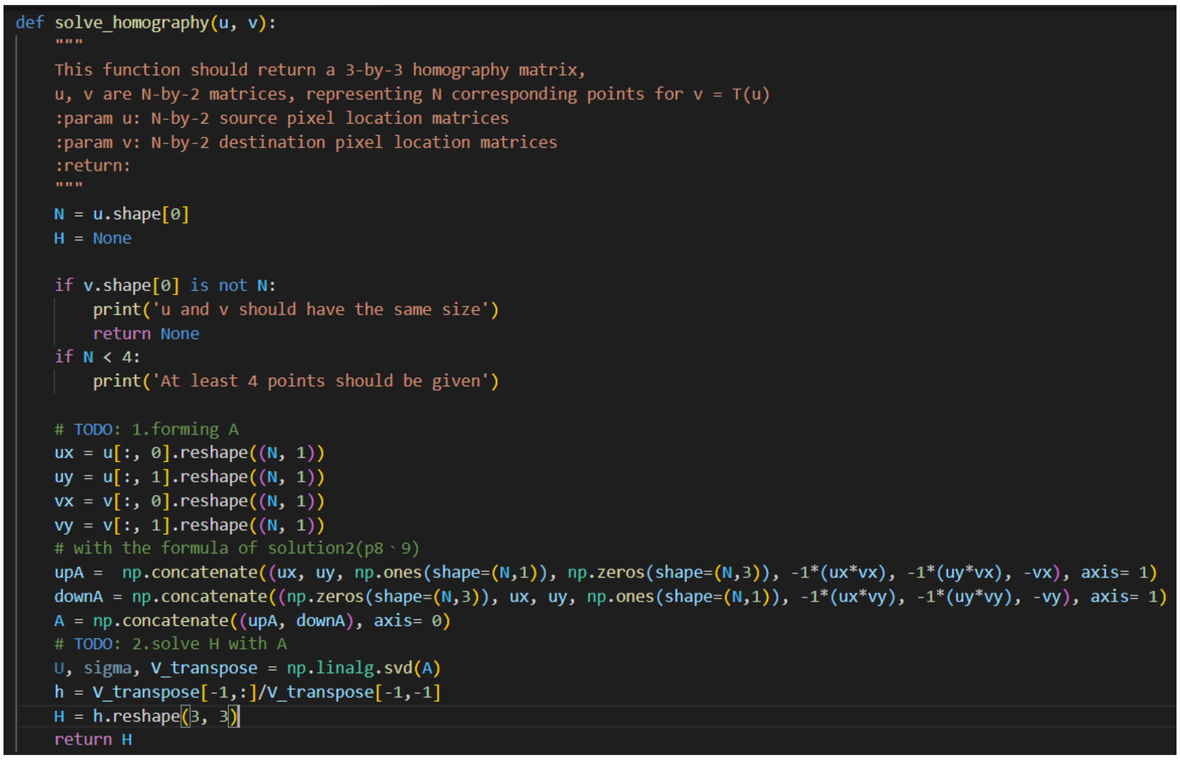
電腦視覺 HW3-Report

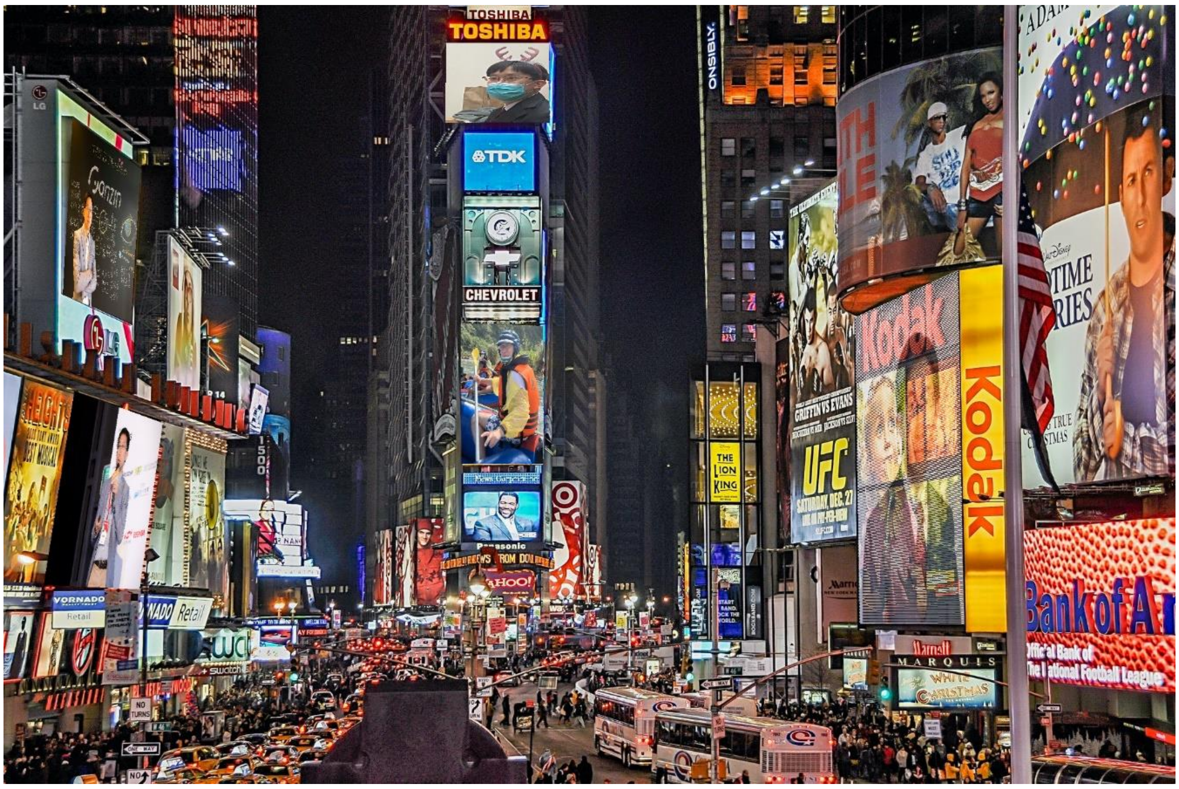
R10522811 機械研一　簡永展

Part 1: Homography estimation

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

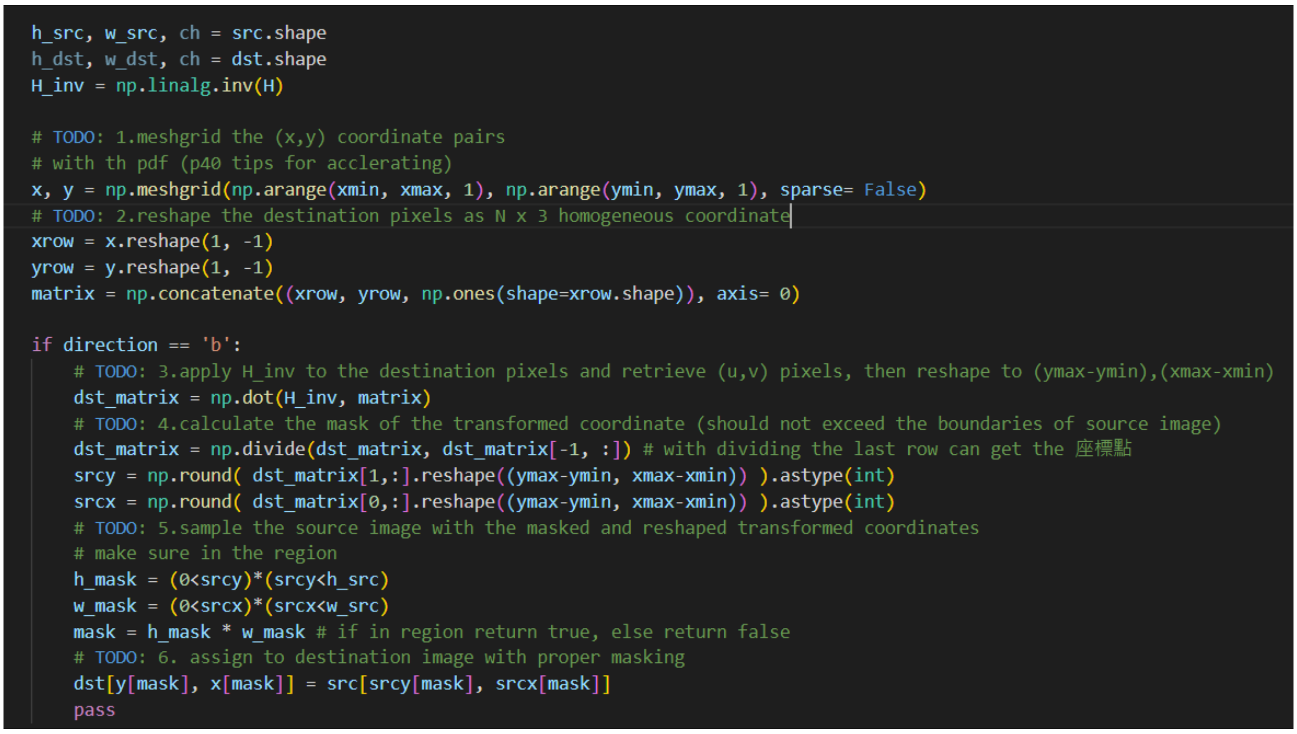


```python
def solve_homography(u, v):
    """
    This function should return a 3-by-3 homography matrix,
    u, v are N-by-2 matrices, representing N corresponding points for v = T(u)
    :param u: N-by-2 source pixel location matrices
    :param v: N-by-2 destination pixel location matrices
    :return:
    """
    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
    ux = u[:, 0].reshape((N, 1))
    uy = u[:, 1].reshape((N, 1))
    vx = v[:, 0].reshape((N, 1))
    vy = v[:, 1].reshape((N, 1))
    # with the formula of solution2(p8、9)
    upA =  np.concatenate((ux, uy, np.ones(shape=(N,1)), np.zeros(shape=(N,3)), -1*(ux*vx), -1*(uy*vx), -vx), axis= 1)
    downA = np.concatenate((np.zeros(shape=(N,3)), ux, uy, np.ones(shape=(N,1)), -1*(ux*vy), -1*(uy*vy), -vy), axis= 1)
    A = np.concatenate((upA, downA), axis= 0)
    # TODO: 2.solve H with A
    U, sigma, V_transpose = np.linalg.svd(A)
    h = V_transpose[-1,:]/V_transpose[-1,-1]
    H = h.reshape(3, 3)
    return H
```

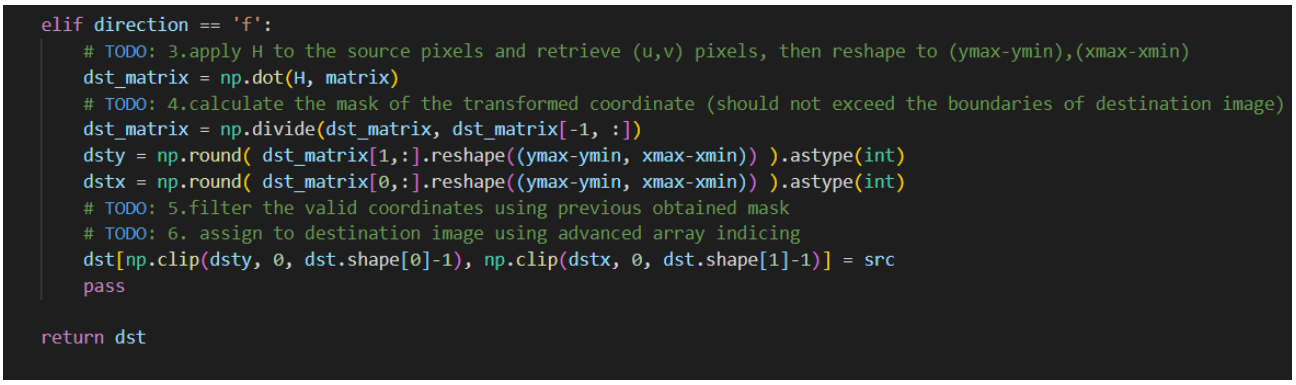


Part 2: Marker-Based Planar AR

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



```python
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
# with th pdf (p40 tips for acclerating)
x, y = np.meshgrid(np.arange(xmin, xmax, 1), np.arange(ymin, ymax, 1), sparse= False)
# TODO: 2.reshape the destination pixels as N x 3 homogeneous coordinate
xrow = x.reshape(1, -1)
yrow = y.reshape(1, -1)
matrix = np.concatenate((xrow, yrow, np.ones(shape=xrow.shape)), axis= 0)

if direction == 'b':
    # TODO: 3.apply H_inv to the destination pixels and retrieve (u,v) pixels, then reshape to (ymax-ymin),(xmax-xmin)
    dst_matrix = np.dot(H_inv, matrix)
    # TODO: 4.calculate the mask of the transformed coordinate (should not exceed the boundaries of source image)
    dst_matrix = np.divide(dst_matrix, dst_matrix[-1, :]) # with dividing the last row can get the 座標點
    srcy = np.round( dst_matrix[1,:].reshape((ymax-ymin, xmax-xmin)) ).astype(int)
    srcx = np.round( dst_matrix[0,:].reshape((ymax-ymin, xmax-xmin)) ).astype(int)
    # TODO: 5.sample the source image with the masked and reshaped transformed coordinates
    # make sure in the region
    h_mask = (0<srcy)*(srcy<h_src)
    w_mask = (0<srcx)*(srcx<w_src)
    mask = h_mask * w_mask # if in region return true, else return false
    # TODO: 6. assign to destination image with proper masking
    dst[y[mask], x[mask]] = src[srcy[mask], srcx[mask]]
    pass
```



```python
elif direction == 'f':
    # TODO: 3.apply H to the source pixels and retrieve (u,v) pixels, then reshape to (ymax-ymin),(xmax-xmin)
    dst_matrix = np.dot(H, matrix)
    # TODO: 4.calculate the mask of the transformed coordinate (should not exceed the boundaries of destination image)
    dst_matrix = np.divide(dst_matrix, dst_matrix[-1, :])
    dsty = np.round( dst_matrix[1,:].reshape((ymax-ymin, xmax-xmin)) ).astype(int)
    dstx = np.round( dst_matrix[0,:].reshape((ymax-ymin, xmax-xmin)) ).astype(int)
    # 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
    pass

return dst
```

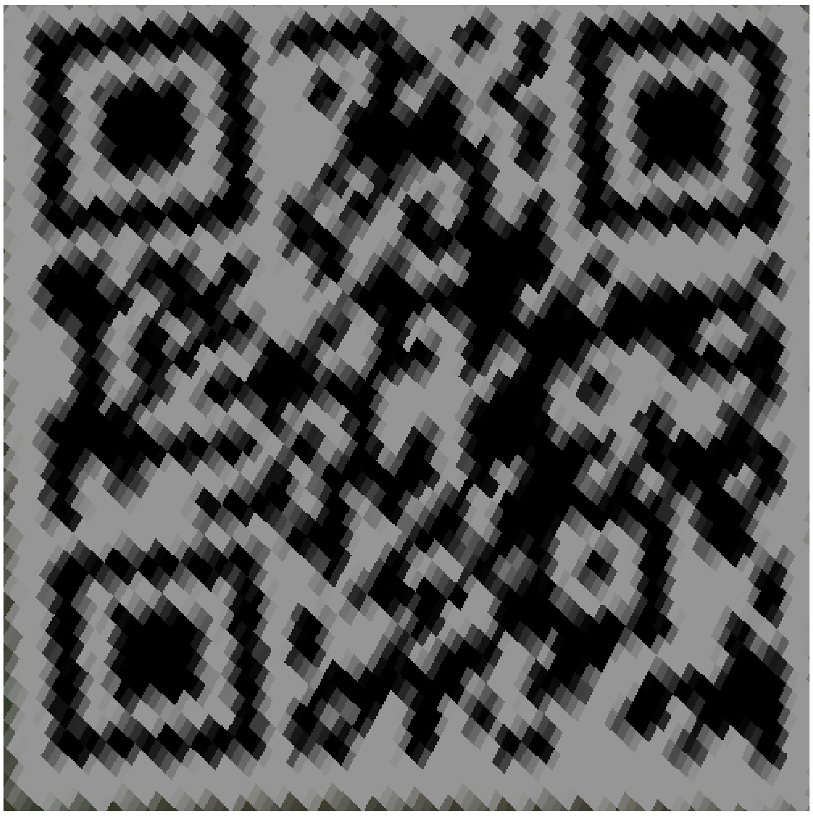
Briefly introduce the interpolation method you use:

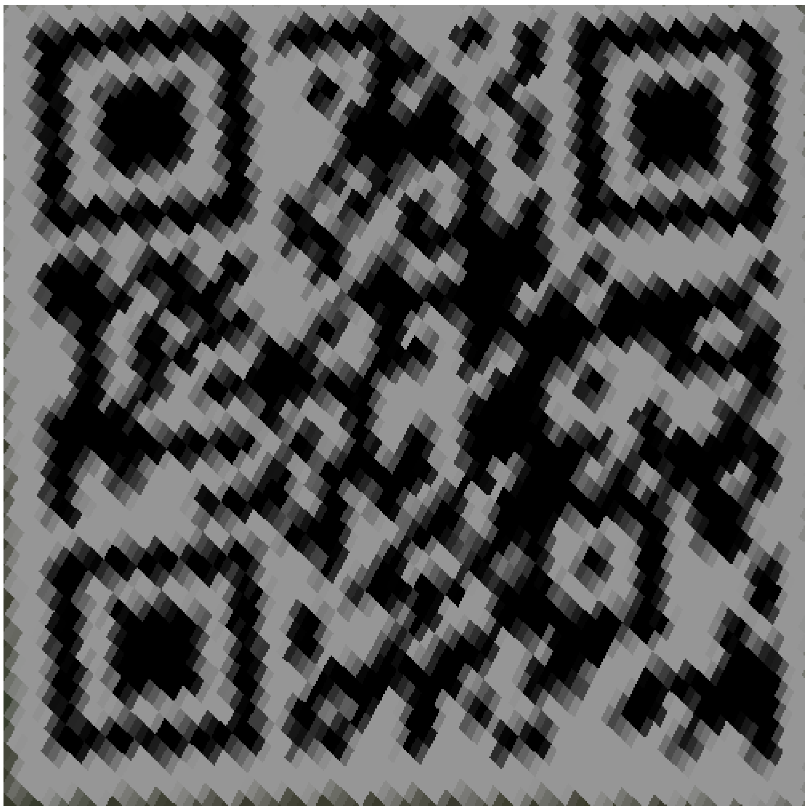
Ans: Forward warping: As you can see, I use the nearest neighbor as my interpolation. I use np.round to take the nearest integer.

Backward warping: As you can see, I just easily use the mask to make sure the region after back warping will match my original shape.

Part 3: Unwarp the secret

Paste the 2 warped images


output3_1.png

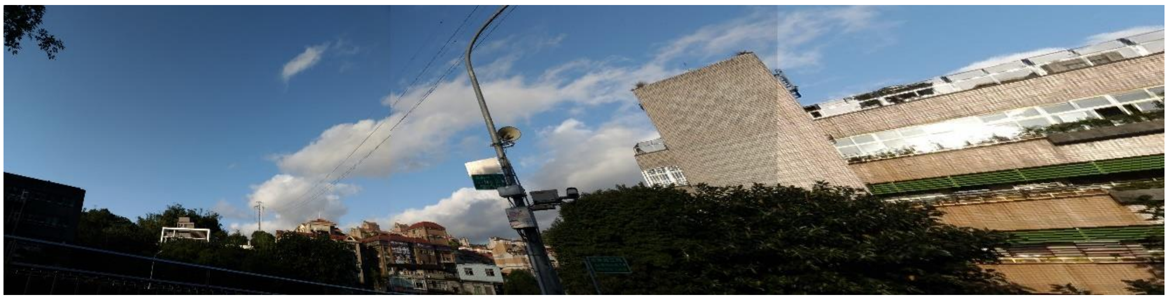

output3_2.png

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

Ans: These two results are the same. Both of these images direct to the same website, which is http://media.ee.ntu.edu.tw/courses/cv/21S/.
I think the first image is more square, which is suitable to do reproduce it; however, the second image is more rectangle, which may loss some information during the back warping. Therefore, first image is clearer than second one.

Part4: Panorama

Paste your stitched panorama



Can all consecutive images be stitched into a panorama?
If yes, explain your reason. If not, explain under what conditions will result in a failure?

Ans: In this work, all these images can be stitched into a panorama; however, it is easy to see that these images can't be stitched one hundred percent perfect. It will lose some information during generating the panorama.

In any case of continuous images: if the panorama is projected to a plane, the rotation angle cannot exceed 180 degrees. If the panorama is projected onto a cylindrical surface, the rotation angle cannot exceed 360 degrees.
Therefore, when the panorama is projected to the plane, when the rotation angle is 180 degrees, the two sides happen to be projected to two infinite distances, which the physical limit has been reached. Therefore, it is impossible to exceed 180 degrees.