Computer Vision

Homework #3

經濟四 李品樺 B07303024

Part 1

```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
    u_x, u_y = u[:, 0], u[:, 1]
    v_x, v_y = v[:, 0], v[:, 1]
    u_x = np.reshape(u_x, (N, 1))
    v_x = np.reshape(v_x, (N, 1))
    u_y = np.reshape(u_y, (N, 1))
    v_y = np.reshape(v_y, (N, 1))

    ones = np.ones((N,1))
    zeros = np.zeros((N,3))
    first = np.concatenate((u_x, u_y, ones, zeros, -u_x*v_x, -u_y*v_x, -v_x), axis=1)
    second = np.concatenate((zeros, u_x, u_y, ones, -u_x*v_y, -u_y*v_y, -v_y), axis=1)
    A = np.vstack((first, second))

    # TODO: 2.solve H with A
    u, s, vt = np.linalg.svd(A)
    # normalize with the last row
    H = (vt[-1] / vt[-1, -1]).reshape(3,3)
    return H
```


TOSHIBA
TDK
CHEVROLET
Kodak
Kodak
UFC
THE LION KING
GRIFFIN VS EVANS
VORNADO
Retail
swatch
MARQUIS
WHITE CHRISTMAS
Panasonic
YAHOO!

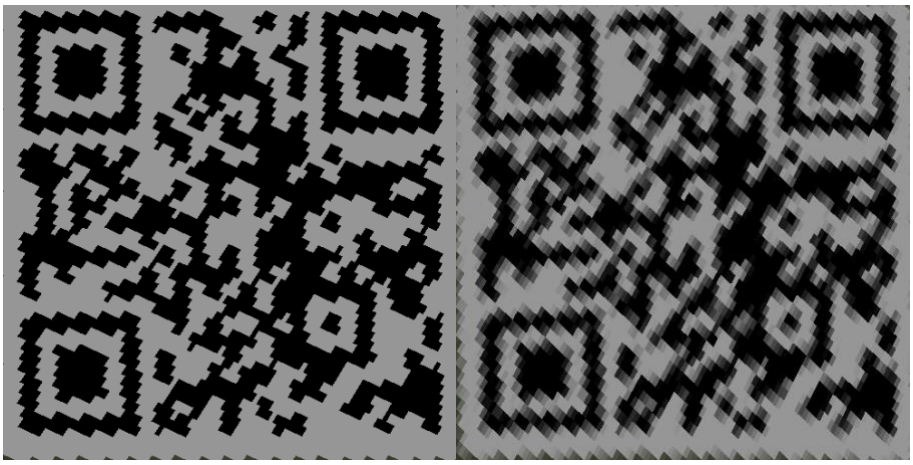
Part 2

- I use the method of nearest neighbor for interpolation. That is, we round the number to the nearest integer.

```python
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
    x = np.arange(xmin, xmax)
    y = np.arange(ymin, ymax)
    grid_x, grid_y = np.meshgrid(x, y)
    row_x, row_y = grid_x.reshape(1, -1), grid_y.reshape(1, -1)
    ones = np.ones(shape=(1, row_x.shape[1]))
    u = np.vstack((row_x, row_y, ones))

    # 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, u)
        v /= v[-1]  # normalize
        v = np.round(v).astype(int)
        # print(v)
        vx, vy = v[0, :], v[1, :]
        vy = vy.reshape(ymax-ymin, xmax-xmin)
        vx = vx.reshape(ymax-ymin, xmax-xmin)

        # TODO: 4.calculate the mask of the transformed coordinate (should not exceed the boundaries of source image)
        mask_y = ((vy >= np.zeros(vy.shape)) * (vy < h_src))
        mask_x = ((vx >= np.zeros(vx.shape)) * (vx < w_src))
        mask_final = mask_y * mask_x

        # TODO: 5.sample the source image with the masked and reshaped transformed coordinates
        output_y = vy * mask_final
        output_x = vx * mask_final
        grid_y *= mask_final
        grid_x *= mask_final
        # TODO: 6. assign to destination image with proper masking
        dst[grid_y, grid_x] = src[output_y, output_x]
```

```python
    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, u)
        v /= v[-1]  # normalize
        v = np.round(v).astype(int)
        vx, vy = v[0, :], v[1, :]
        vy = vy.reshape(ymax-ymin, xmax-xmin)
        vx = vx.reshape(ymax-ymin, xmax-xmin)
        # TODO: 4.calculate the mask of the transformed coordinate (should not exceed the boundaries of destination image)
        mask_y = ((vy >= np.zeros(vy.shape)) * (vy < h_dst))
        mask_x= ((vx >= np.zeros(vx.shape)) * (vx < w_dst))
        mask_final = mask_y * mask_x
        # TODO: 5.filter the valid coordinates using previous obtained mask
        output_y = vy * mask_final
        output_x = vx * mask_final
        # TODO: 6. assign to destination image using advanced array indicing
        dst[output_y, output_x] = src
    return dst
```

Part 3



- Link of both QR code leads to : [Computer Vision | Spring 2021 (ntu.edu.tw)](http://media.ee.ntu.edu.tw/courses/cv/21S/) (http://media.ee.ntu.edu.tw/courses/cv/21S/)
- The image of the building taken by smartphone does not have all straight lines on the celling (i.e. some straight lines are twisted), thus this source image is different from the one taken by camera.
- We can use the all() function provided by nparray to check the equality of the two images. That is, (output3_1==output3_2).all(), and the result is False. Thus, the warped results are the different.

```
(output3_1 == output3_2).all()
```

- The warped results are different due to the fact that the edges of the second input image are not straight lines, thus it is harder to warp to the image correctly, and the warped image is more blur.

Part 4

- Not all consecutive images can be stitched into a panorama
- If we want to get a projection of the panoramic sphere onto a flat surface, it is impossible to display panoramas wider or taller than 180 degrees in rectilinear projection since the two sides are already projected to infinity. As the projected angle increases, the image is more twisted. The suggested angle for projection is < 120°.

Reference: Panoramic Image Projections (cambridgeincolour.com) and Projections - PTGui Stitching Software