Computer Vision, Spring 2023 HW3

B11705009 An-Che, Liang

Part 1: Homography Estimation

Warped canvas:

Part 2: Marker-Based Planar AR

solve_homography(u, v)

In [ ]:

```python
import numpy as np
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
    A = np.zeros((2*N, 9))
    for i in range(N):
        ux, uy = u[i]
        vx, vy = v[i]
        A[2*i] = np.array([ux, uy, 1, 0, 0, 0, -ux*vx, -uy*vx, -vx])
        A[2*i+1] = np.array([0, 0, 0, ux, uy, 1, -ux*vy, -uy*vy, -vy])
    # TODO: 2.solve H with A
    U, S, VT = np.linalg.svd(A, full_matrices=True)
    V = VT.T
    h = VT[-1, :]/VT[-1, -1]
    H = np.reshape(h, (3, 3))
    return H
```

warping()

In [ ]:
```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, 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
    x_row = x.reshape(((xmax - xmin)*(ymax - ymin), 1))
    y_row = y.reshape(((xmax - xmin)*(ymax - ymin), 1))
    homogeneous_coords = np.concatenate((x_row, y_row, np.ones(
        ((xmax-xmin)*(ymax-ymin), 1))), axis=1)
    if direction == 'b':
        # TODO: 3.apply H_inv to the destination pixels and retrieve (u,v) pixels, th
        u = np.matmul(H_inv, homogeneous_coords.T).T
        u = np.divide(u, u[:, 2, None])
        ux = np.round(u[:, 0]).reshape(
            ((ymax - ymin), (xmax - xmin))).astype(np.int32)
        uy = np.round(u[:, 1]).reshape(
            ((ymax - ymin), (xmax - xmin))).astype(np.int32)
        # TODO: 4.calculate the mask of the transformed coordinate (should not exceed
        mask = ((0 < uy)*(uy < h_src))*((0 < ux)*(ux < w_src))
        # TODO: 5.sample the source image with the masked and reshaped transformed co
        dst[y[mask], x[mask]] = src[uy[mask], ux[mask]]
        # TODO: 6. assign to destination image with proper masking
        pass

    elif direction == 'f':
        # TODO: 3.apply H to the source pixels and retrieve (u,v) pixels, then reshap
        v = np.matmul(H, homogeneous_coords.T).T
        v = np.divide(v, v[:, 2, None])
        vx = np.round(v[:, 0].reshape(
            (ymax - ymin), (xmax - xmin))).astype(np.int32)
        vy = np.round(v[:, 1].reshape(
            (ymax-ymin), (xmax - xmin))).astype(np.int32)
        # TODO: 4.calculate the mask of the transformed coordinate (should not exceed

        # TODO: 5.filter the valid coordinates using previous obtained mask
        masked_vx = np.clip(vx, 0, w_dst-1)
        masked_vy = np.clip(vy, 0, h_dst-1)
        # TODO: 6. assign to destination image using advanced array indicing
        dst[masked_vy, masked_vx] = src

    return dst
```

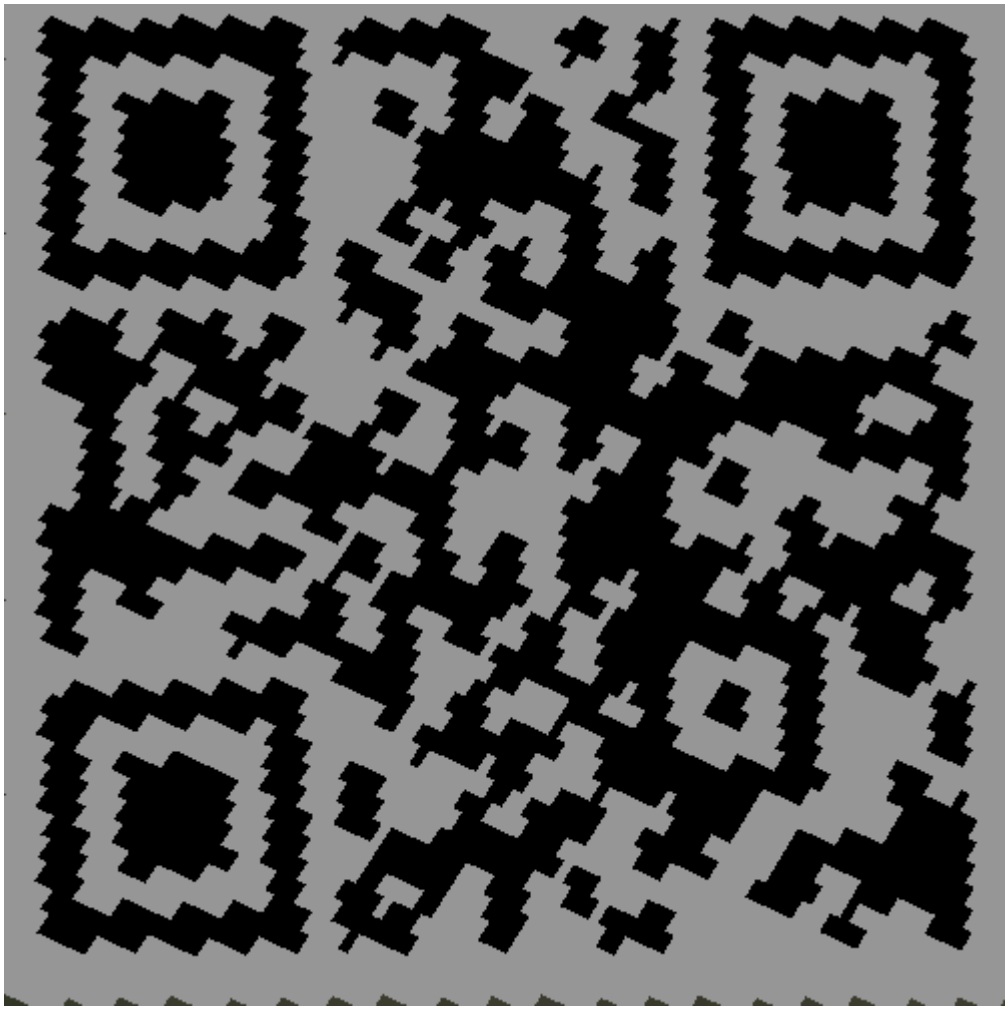
Interpolation method

For the sake of simplicity, I use the `np.round()` method to sample pixel value from the nearest integer coordinate.

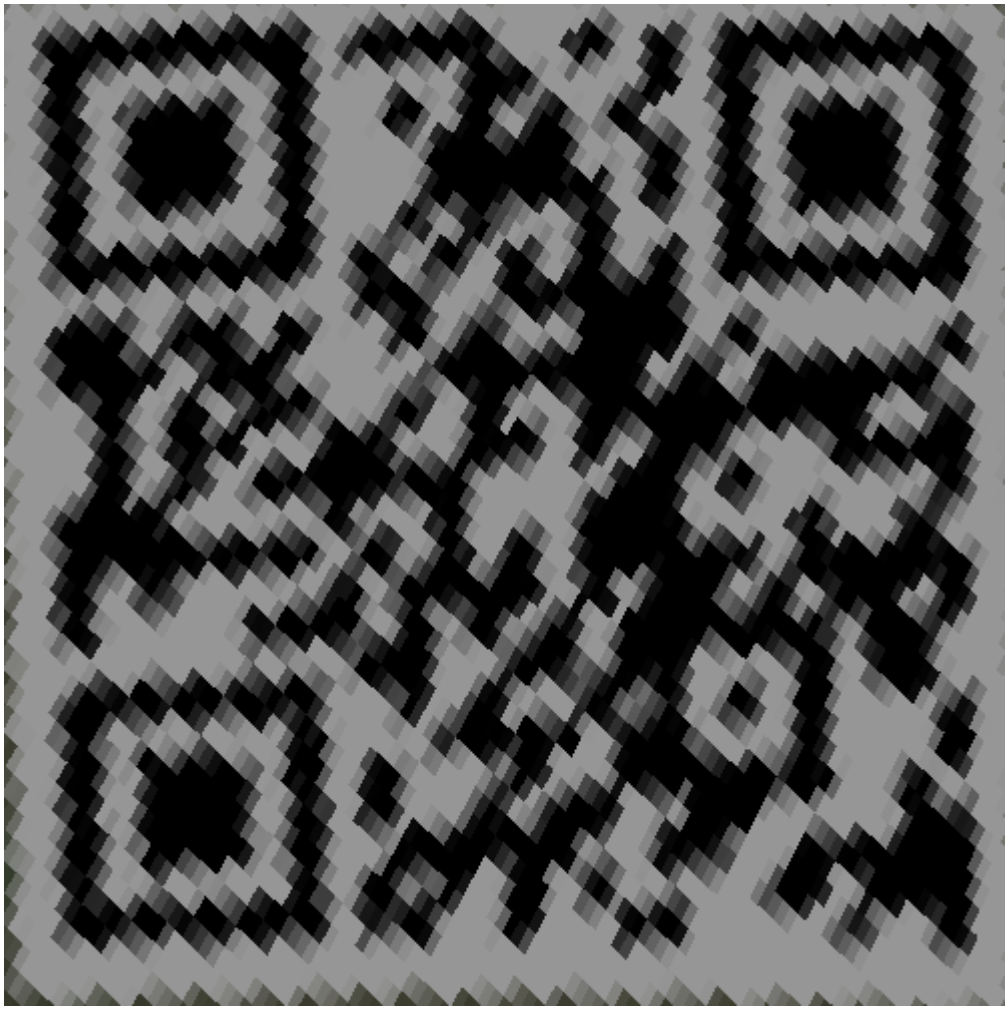
Part 3: Unwarp the secret

Unwarped images:

From `BL_secret1.png`:



From `BL_secret2.png`:



Warped result:

We can see that the two unwarped images are different. The first is sharper, with clear edges, while the second is blurred. I think the reason why is that we can see `BL_secret2.png` is distorted (the straight line on the building become curved line), and our unwarp function will try to retrieve the QR code by project it on a plane, which may cause the QR code we retrieve to be distorted as well.

Part 4: Panorama

Stitched panorama:

Can all consecutive images be stitched into a panorama?

False, by reordering the images, our program cannot generate the same panorama. The reason why is that in order to stitch two image together to form a part of the whole panorama, the two image must have some overlapping features, so we can match the two images on the same plane, but if the two consecutive images don't have overlapping features, then our algorithm will fail to stitch them together, thus explains the order of the images should not be changed randomly.