HW3 - Writeup

Computer Science and Engineering
2017-18538
Hwang Sunyoung

Part 1

H = compute_h(p1, p2)

p1 and p2 is N×2 matrices of corresponded $(x,y)^T$ coordinates between two images.

Suppose p1 is $(x_1,\ y_1)$, $(x_2,\ y_2)$, $(x_3,\ y_3)$, $(x_4,\ y_4)$ and p2 is $(x'_1,\ y'_1)$, $(x'_2,\ y'_2)$, $(x'_3,\ y'_3)$, $(x'_4,\ y'_4)$

for each correspondence point, we can write 2×9 matrices such as

$$p_i = \begin{bmatrix} -x_i & -y_i & -1 & 0 & 0 & 0 & x_i x'_i & y_i x'_i & x'_i \\ 0 & 0 & 0 & -x_i & -y_i & -1 & x_i y'_i & y_i y'_i & y'_i \end{bmatrix}$$

To compute PH = 0, we can matrix multiplication like that:

$$PH = \begin{bmatrix} -x_1 & -y_1 & -1 & 0 & 0 & 0 & x_1 x'_1 & y_1 x'_1 & x'_1 \\ 0 & 0 & 0 & -x_1 & -y_1 & -1 & x_1 y'_1 & y_1 y'_1 & y'_1 \\ -x_2 & -y_2 & -1 & 0 & 0 & 0 & x_2 x'_2 & y_2 x'_2 & x'_2 \\ 0 & 0 & 0 & -x_2 & -y_2 & -1 & x_2 y'_2 & y_2 y'_2 & y'_2 \\ -x_3 & -y_3 & -1 & 0 & 0 & 0 & x_3 x'_3 & y_3 x'_3 & x'_3 \\ 0 & 0 & 0 & -x_3 & -y_3 & -1 & x_3 y'_3 & y_3 y'_3 & y'_3 \\ -x_4 & -y_4 & -1 & 0 & 0 & 0 & x_4 x'_4 & y_4 x'_4 & x'_4 \\ 0 & 0 & 0 & -x_4 & -y_4 & -1 & x_4 y'_4 & y_4 y'_4 & y'_4 \end{bmatrix} \begin{bmatrix} h1 \\ h2 \\ h3 \\ h4 \\ h5 \\ h6 \\ h7 \\ h8 \\ h9 \end{bmatrix} = 0$$

To get H, compute SVD $P = USV^T$ and get last singular vector of V^T as H. Then reshape H to 3×3 matrix.

H = compute_h_norm(p1, p2)

I expressed normalization as a matrix. A shape of normalization matrix is equal to that of p1. p1 is divided by 1600 and p2 is divided by 1200. Then compute H with normalized p1 and p2.

Part 2

p_in, p_ref = set_cor_mosaic()

```
p_in = np.array([[1283, 418],
                 [1444, 404],
                 [1283, 511],
                 [1446, 506],
                 [1239, 541],
                 [1296, 543],
                 [1255, 964],
                 [1284, 967],
                 [1334, 919],
                 [1461, 585],
                 [1466, 723]])
p_ref = np.array([[535, 423],
                  [678, 424],
                  [536, 515],
                  [679, 513],
                  [493, 544],
                  [548, 545],
                  [507, 948],
                  [536, 951],
                  [583, 898],
                  [691, 584],
                  [694, 711]])
```

p_in and p_ref is N by 2 matrices of corresponded $(x, y)^T$ coordinates between two images.

igs_warp, igs_merge = warp_image(igs_in, igs_ref, H)

This function warp igs_in to view of igs_ref. First, compute corresponding coordinate in igs_in to view of igs_ref. And for each pixel, compute np.linalg.solve(A, b) to get t matrix for affine transformation. Then we can get igs_warp image. Then arrange igs_warp and igs_ref to igs_merge in proper position for panorama image.

Part 3

c_in, c_ref = set_cor_rec()

```
c_in = np.array([[1064, 166],
                 [1403, 127],
                 [1047, 869],
                 [1397, 888]])

c_ref = np.array([[0, 0],
                  [200, 0],
                  [0, 400],
                  [200, 400]])
```

c_in and c_ref is N by 2 matrices of corresponded $(x, y)^T$ coordinates between two images.

igs_rec = rectify(igs, p1, p2)

To normalize and compute H, normalize p1 and p2 and apply compute_h function. Then produce igs_rec, this method is equal to **warp_image.** Difference is normalizing factor, 1920 and 1056, that is size of igs.