Batuhan Tosun

2017401141

EE 576 MACHINE VISION | HOMEWORK 1

In this project, our goal is to find the homography map between a pair of images. The equation for the homography map is given by:

$$x' = Hx = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} x$$

where x is a pixel from the first image (source) and x' is the corresponding pixel in the second image (desired).

The first step in our project is to read a pair of consecutive images to a OpenCV Mat variable and display them side by side. To achieve this the main function (main.cpp) asks user to enter the file names of the two images (those files should be in the directory named "Data"). To display images side-by-side, first those images are first concatenated and then a window is opened to display those images using OpenCV's imshow function. While continue displaying the images, the program asks user to enter the number of input corresponding points, which is N. After that the program expects from user to choose N coordinates from both images (in total 2N points) by just clicking on the points with the left-button of the mouse (with a method named myCallBackFunc **[1]**). Those points (coordinates) are then stored in 2D vectors named d (for first image) and d_hat (for second image). After that the coordinates in those vectors are written into two separate txt files with names "img1.txt" and "img2.txt" to be used in my MATLAB file.

The MATLAB file findLinearSolution is designed to find a solution to the linear system:

$$Ah = a$$

where A is a 2Nx6 matrix, h = [h_{11} h_{12} h_{13} h_{21} h_{22} h_{23}], and a is 2Nx1 column vector. The solution to this linear system is done by using the Least Squares Formulation which results in:

$$h = (A^T A)^{-1} A^T a$$

After obtaining h with respect to the given coordinates and the formula above, the homography map H is constructed as well.

Continuing with our main program, the program asks user (with a method named homographyMapManuallyInput) to enter the elements of the manually calculated homography map row-by-row. Then this method returns the given homogprahy map in OpenCV Mat format.

The next step is to design a method named findHomographyMap that finds the homography map using built-in OpenCV function findHomography **[2]** with given vectors d and d_hat and returns the resulting homography matrix.

At the last step of our project, it is expected from us to apply both the manually computed and OpenCV computed homography transform on the first image and then display the second image, the first image transformed using manually computed H and first image transformed using OpenCV computed H side-by-side. And this is what the result of the program does.

To comment on the sensitivity of the results to N, homography transform is implemented on 3 different consecutive image pairs with 2 different N values (5 and 10). The first consecutive image pair is “boatB.png” (the first image) and “boatA.png” (the second image).

Fig1: Two consecutive images side-by-side (“boatB.png” on the left, “boatA.png” on the right)

First both the manually computed and OpenCV computed homography transforms are applied on the first image with N = 5. Then both the manually computed and OpenCV computed homography transforms are applied on the first image with N = 10.

Fig2: The second image (“boatA.png”), the first image transformed using OpenCV computed H with N=5 and the first image transformed using manually computed H with N=5

Fig3: The second image (“boatA.png”), the first image transformed using OpenCV computed H with N=5 and the first image transformed using manually computed H with N=10

From the figures Fig2 and Fig3, it can be observed that as we increase the number of input corresponding points (N), the resulting images after homography transforms (with both manually computed and OpenCV computed) becomes more similar to the second image (desired). Therefore, increasing N has a positive effect on the performance of the homogpraphy transform. Especially, the performance of the OpenCV computed H increases significantly (looking at the images at the middle).

The second consecutive image pair is “boatB.png” (the first image) and “boatA.png” (the second image).

Fig4: Two consecutive images side-by-side (“LePoint2A.png” on the left, “LePoint2B.png” on the right)

First both the manually computed and OpenCV computed homography transforms are applied on the first image with N = 5 and then with N=10

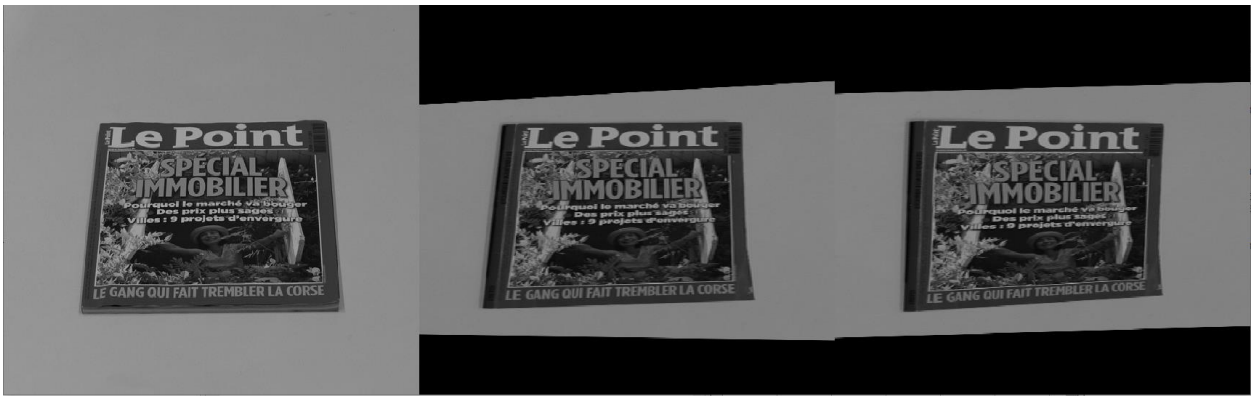

Fig5: The second image (“LePoint2B.png”), the first image transformed using OpenCV computed H with N=5 and the first image transformed using manually computed H with N=5

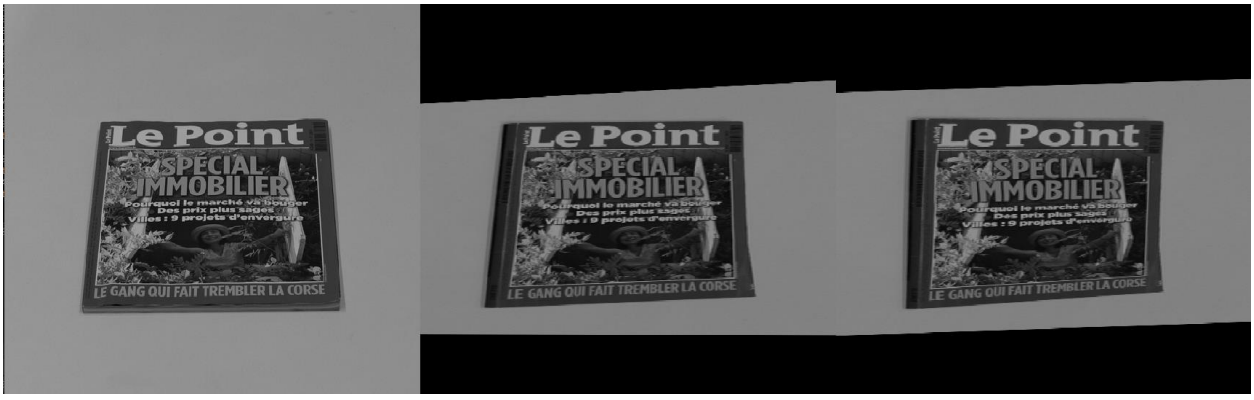

Fig6: The second image ("LePoint2B.png"), the first image transformed using OpenCV computed H with N=5 and the first image transformed using manually computed H with N=10

From the figures Fig5 and Fig3, again, it can be observed that as we increase the number of input corresponding points (N), the resulting images after homography transforms (with both manually computed and OpenCV computed) becomes more similar to the second image (desired). Therefore, increasing N has a positive effect on the performance of the homogpraphy transform. To see the effect of the N in this case, the smaller N should be chosen as 4 and the larger N as 12 since the results with both N's look pretty similar.

The last consecutive image pair is "boatB.png" (the first image) and "boatA.png" (the second image).

Fig7: Two consecutive images side-by-side ("LePoint1B.png" on the left, "LePoint1A.png" on the right)

First both the manually computed and OpenCV computed homography transforms are applied on the first image with N = 5 and then with N=10

Fig8: The second image ("LePoint1A.png"), the first image transformed using OpenCV computed H with N=5 and the first image transformed using manually computed H with N=5

Fig9: The second image ("LePoint1A.png"), the first image transformed using OpenCV computed H with N=5 and the first image transformed using manually computed H with N=10

From the figures Fig8 and Fig9, again, it can be observed that as we increase the number of input corresponding points (N), the resulting images after homography transforms (with both manually computed and OpenCV computed) becomes more similar to the second image (desired). Therefore, increasing N has a positive effect on the performance of the homogpraphy transform. To see the effect of the N in this case, the smaller N should be chosen as 4 and the larger N as 12 since the results with both N's look similar but still there is a little improvement on the transforms.

References

[1] https://www.opencv-srf.com/2011/11/mouse-events.html

[2] https://docs.opencv.org/4.x/d9/dab/tutorial_homography.html