Laboratory Session 1

Computer Vision - Università Degli Studi di Genova

Justin Lee
Student ID: S4885003
leej1996@gmail.com

Steven Palma Morera
Student ID: S4882385
imstevenpm.study@gmail.com

***Abstract*—An algorithm of backward warping for translating, rotating and for a custom transforming two different source images using a bilinear transformation, were developed and tested successfully in MATLAB in order to get acquainted to image processing operations.**

***Index Terms*—backward warping, bilinear interpolation, image processing**

I. Introduction

The goal of this lab ([2]) is to implement some basic image processing operations in Matlab. Image warping and bilinear interpolation are used in order to compute a transformed image based on a source color image. The transformations to be done are translation, rotation, and a custom warping based off a MATLAB data file and greyscale for simplicity. The two sources images are displayed in Fig.1 and Fig.2.

Fig. 1. Test Image of Boccadasse. Source: [2]

Fig. 2. Test Image of a flower. Source: [2]

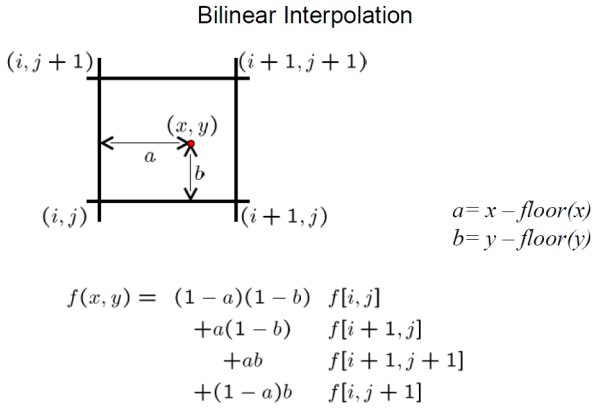

Fig. 3. Bilinear Interpolation. Source: [1]

II. Procedure

A. Bilinear Interpolation

The bilinear interpolation was executed by following a general formula that is described in Fig. 3. The reader can refer to the readme.txt document to find further information of the input and output arguments of each algorithm developed.

B. Backward warping - Translation

The translation was done by iterating through each pixel of the output image and subtracting the given translations from the x and y components respectively. This gives the coordinates in the source image for the desired intensity for the output image at its coordinates. Bilinear Interpolation was used to find the intensity values.

C. Backward warping - Rotation

Rotation was achieved by using an inverse rotation matrix to go from the coordinates of the output image to the desired coordinates on the input image. Bilinear interpolation was used to find the intensity values.

D. Backward warping - Custom transformation

A meshgrid was created for the image, and the custom transformation was loaded from the matlab data file and added to the meshgrid. This was then turned into the desired image. Notice that each source image needs a different data file because of its resolution.

III. Results

A. Backward warping - Translation

After running the 'translateimg.m' script in the MATLAB workspace as follows -using the flower image as the input- the Fig.4 was obtained.

translateimg(25.8,-36.3,'flower.jpg');,

Fig. 4. Flower image translated 25.8 px in the X-axis and -36.3 px in the Y-axis. Source: Own elaboration

After running again the same script as before but now with the following call -using the landscape image as the input-: the Fig.5 was obtained.

translateimg(-41.9,10.7,'boccadasse.jpg');,

B. Backward warping - Rotation

After running the 'rotateimg.m' script in the MATLAB workspace as follows -using the flower image as the input: the Fig.6 was obtained.

rotateimg(pi/4,'flower.jpg');,

After running again this same script but now with the following call -using the landscape image as the input-: the Fig.7 was obtained.

rotateimg(-1.812,'boccadasse.jpg');,

Fig. 5. Landscape image translated -41.9 px in the X-axis and 10.7 px in the Y-axis. Source: Own elaboration

Fig. 6. Flower image rotated 45 degrees along its center. Source: Own elaboration

Fig. 7. Landscape image rotated -103.8 degrees along its center. Source: Own elaboration

C. Backward warping - Custom Transformation

After executing the 'warpingimg.m' script in the MATLAB workspace as follows -using the flower image as the input-: the Fig.8 was obtained.

warpingimg('flower.jpg','flowerdata.mat');,

Fig. 8. Flower image after the custom transformation. Source: Own elaboration

After executing the same script as follows, but now using the landscape image as the input: the Fig.9 was obtained.

warpingimg('boccadasse.jpg','data.mat');,

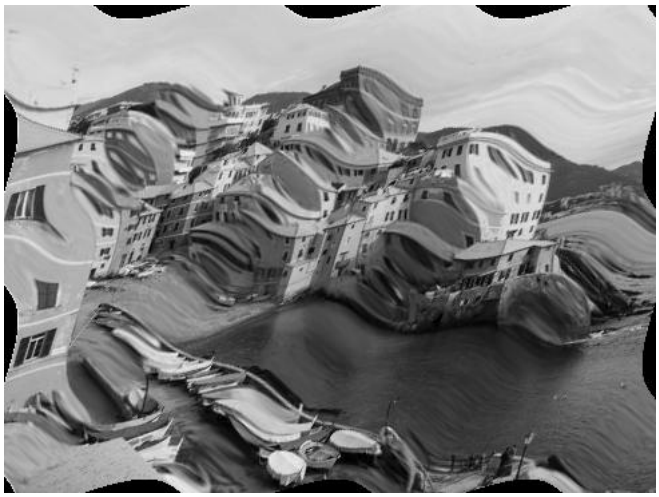

Fig. 9. Landscape image after the custom transformation. Source: Own elaboration

IV. Analysis of the results

A. Backward warping - Translation

Calling the 'translateimg.m' script like:

translateimg(25.8,-36.3,'flower.jpg');

it would mean that the desired output image should be moved 25.8 pixels on the X-axis and -36.3 pixels on the Y-axis. Considering that the origin of the reference system is located in the top left corner of each image, with the positive X-axis pointing right and the positive Y-axis pointing down, then a movement to the left-top corner should be expected. As can be appreciated in the Fig.4, this is exactly the result obtained. Same goes for the Fig.5 where, since the function was called like:

translateimg(-41.9,10.7,'boccadasse.jpg');,

a movement to the right-down corner should be the expected -and also in this case, the obtained- output.

Also it can be notice that the algorithm is robust enough to its inputs, since it properly handles positives and negatives values for the displacement, as well as integer and non-integer numbers. However, when the desired displacement of the image is a non-integer number, two things can be noticed. First, the Bilinear Interpolation algorithm is executed, since when performing the backward mapping to the original image, there is not a pixel with non-integer coordinates, so an interpolation should be used to get the intensity value of that point in the input image. Second, since there is not such thing as non-integer coordinates for a pixel, then if a non-integer displacement is desired, the output image will be rounded up to the closest integer. This last point means that, for example in the Fig.5 even if the desired displacement is (-41.9,10.7) the output image can be perceived as if it was moved (-42,11) pixels.

B. Backward warping - Rotation

By running the 'rotateimg.m' script like:

rotateimg(pi/4,'flower.jpg');

it would mean that the desired output image should be rotated 45 degrees along its center. Following the right-hand rule, from the direction of the X and Y axis explained before, the Z axis should be pointing inwards the image; therefore, the movement should be clockwise. Judging from Fig.6, this exact behaviour can be appreciated. And from Fig.7, where the script where called like:

rotateimg(-1.812,'boccadasse.jpg');,

it can be noticed that the ouput image was correctly rotated anticlockwise 103.8 degrees, as desired and expected.

Furthermore, saying that the image is rotated along its center is not completely true. If the resolution of the image is not square and odd, the image will be rotated along the closest left-down pixel to its center, this is because in this case there is not an actual middle pixel in the image so the algorithm developed chooses the said before. Therefore, if measured, the output image could be perceived as not entirely rotated as desired.

C. Backward warping - Custom Transformation

Each of the source images has its own data file to do the custom transformation. If the warping of the images was

done correctly, it would be shown that the images become distorted: certain sections become squished while others become stretched, creating a wavy pattern along the image. An example of this can be found in lab instructions. This is displayed very prominently in Fig. 9 and Fig. 8 and the images look very similar examples shown in the lab instructions.

D. Bilinear Interpolation

Both the backward warping for translation and rotation uses the Bilinear Interpolation algorithm to compute the intensity value of the corresponding input image point whenever that point is or not a real pixel in the input image, so an analysis must be performed in order to understand if the Bilinear Interpolation algorithm developed is properly working.

To evaluate the proper functioning of the bilinear interpolation algorithm implemented, the 'rotateimg.m' script was excecuted as follows:

rotateimg(0,'boccadasse.jpg');,

then the histogram of both the original image and the output image were registered, refer to Fig.10 and Fig.11 respectively.

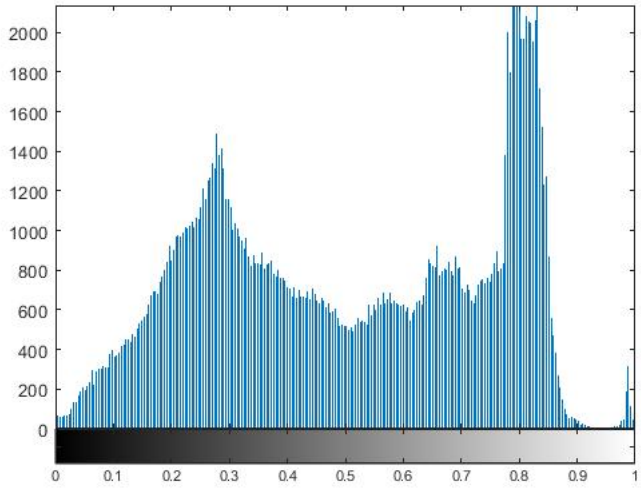

Fig. 10. Histogram of the original landscape image. Source: Own elaboration

Even if no rotation is performed, the intensity value of the pixels of the output image are generated by the bilinear interpolation algorithm implemented. It can be said that if the algorithm doesn't change significantly the intensity pixel values of the input image nor the distribution of them, then the algorithm is performing effectively. By looking at the Fig.10 and Fig.11 it is noticeable that the form of the histogram is still the same for both the source image and output image, meaning that the algorithm developed is working properly.

V. Conclusions

- All scripts developed execute their desired transformations correctly.
- The bilinear interpolation algorithm proves effective at interpolating values from surrounding pixels.

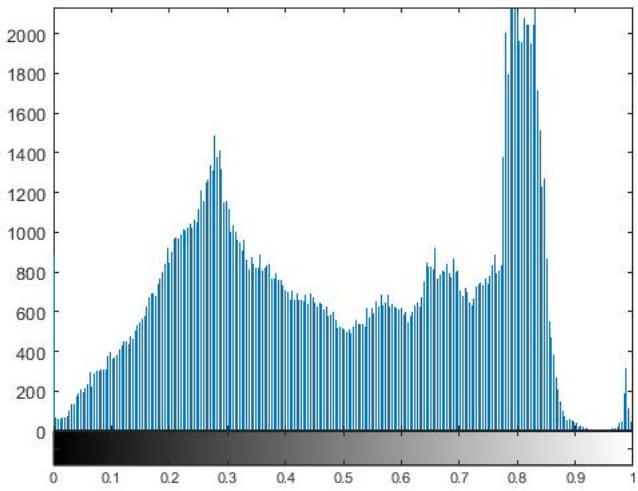

Fig. 11. Histogram of the landscape image rotated 0 degrees. Source: Own elaboration

References

[1] Solari, F. Part 1 - Image Processing Fundamentals, 2020. Retrieved from: https://2019.aulaweb.unige.it/pluginfile.php/209243/mod_resource/content/3/lecture1_image_fundamentals.pdf

[2] Solari, F. Lab Session n.1, 2020. Retrieved from: https://2019.aulaweb.unige.it/pluginfile.php/209245/mod_resource/content/2/Lab1.pdf