HCM UNIVERSITY OF FALCUTY – FALCUTY OF
INFORMATION TECHNOLOGY

Applied Mathematics and Statistics

PROJECT 02: Image Processing

Lê Quốc Trung – 20127369

Teacher: Phan Thị Phương Uyên – Nguyễn Văn Quang Huy

Index

I. Information

- Student ID: 20127369
- Student name: Lê Quốc Trung

II. Function completion

No	Function	Completion
1	Adjust brightness	100%
2	Adjust contrast	100%
3	Flip image	100%
4	Convert to grayscale and sepia	100%
5	Crop image	100%
6	Blur image and sharpen image	100%

III. Idea description

In this project, I use only libraries teacher provided such as: *Numpy*, *PIL* (*open()*, *save()* from *Image*), *Matplotlib* (*imshow()* from *pyplot*)

1. Open image

First, I get the name of the image file from the keyboard. Then I use the method *open()* from *Image* to open the image file and change it to PIL object. Numpy.asarray to change it to numpy array. And after that, I change the shape of the array to 2D array.

2. Adjust brightness

- For more code detail, please check out my *20127369.ipynb* file.
- For every number in *img1d* , that is R,G,B of pixel, I plus it with alpha (input from keyboard). Alpha might be negative or positive. Note that, if the number plus alpha exceed 255 then I set the value to 255, and if the sum is negative then I set it to 0

3. Adjust contrast

- For every number in *img1d* , that is R,G,B of pixel, I multiply it with alpha (input from keyboard). Alpha is a positive float. Note that, if the number multiply alpha exceed 255 then I set the value to 255
- In the end, I change all number in *img1d* to *int* type using *.astype(int)*

4. Flip image

- I change *img1d* to img with the original shape *[h,w,c]* and the axis is the horizontal or vertical axis input from the keyboard.
- The cool thing is that *Numpy* provides a function *fliplr*, which means flip left-right and *flipud*, which means flip up-down the array passed in parameter. The array can be any n-dimension.
- The reason I reshape *img1d* to the original image array is because whenever I flip left-right or up-down, it will flip the pixels themselves. That is cool!

5. Convert to grayscale and sepia

- For every pixel in *img1d* I change the value of R,G,B to the same value by using a part of each value and sum up to the final value.
- Weighted method says 30% of red, 59% of green, 11% of blue because red has more wavelength of all the three colors, and green is the color that has not only less wavelength than red color but also gives a more soothing effect to eyes.

6. Crop image from the center

7. Blur image and sharpen image

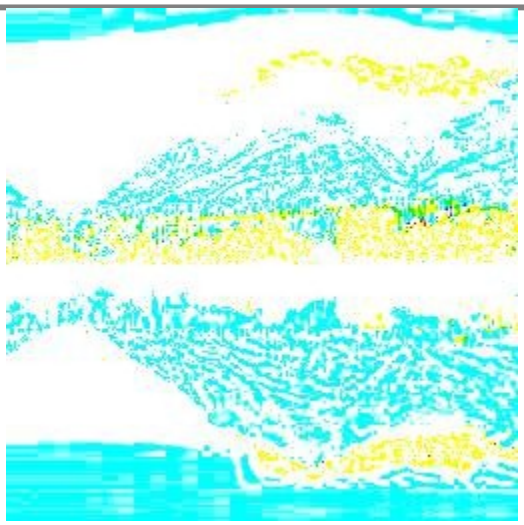
- I prepare 2 3x3 numpy arrays for 2 kernel for guassian blur and sharpen kernel. This is the kernel using for image processing.
- *img* is an array with 3-dimension as origin one.
- For every pixel I access in *img* I do blur for R,G,B correspondingly by np.multiply (multiply element in the same position of 2 matrix, not like matrix multiply matrix) of blur matrix with a 3x3 matrix of R (G/B), in which the central element is the one need to be blur. Then sum it up and assign to the pixel R (G/B) by np.sum
- For pixels that can not find the local neighbor pixel to make a 3x3 matrix I don’t change its RGB value.

IV. Result image

Original image:

Adjust brightness:

Adjust contrast:



Flip horizontal:

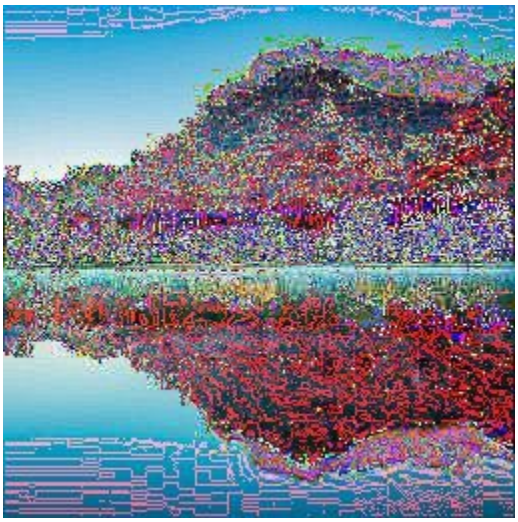
Flip vertical:

Gray image:

Sepia image:

Blur image:

Sharpen image:



V. Reference

https://numpy.org/doc/stable/reference/generated/numpy.flip.html

https://www.tutorialspoint.com/dip/grayscale_to_rgb_conversion.htm

https://en.wikipedia.org/wiki/Kernel_(image_processing)

https://betterdatascience.com/implement-convolutions-from-scratch-in-python/