HCM UNIVERSITY OF FALCUTY – FALCUTY OF INFORMATION TECHNOLOGY

Applied Mathematics and Statistics

PROJECT 02: Image Processing

Võ Duy Nhân – 20127056 – 20CLC11

4/7/2022

Teacher: Phan Thị Phương Uyên – Nguyễn Văn Quang Huy

Index

I. Information

- Student ID: 20127056
- Student name: Võ Duy Nhân
- Class: 20CLC11

II. Function completion

No	Function	Completion
1	Adjust brightness	100%
2	Adjust contrast	100%
3	Flip image	100%
4	Convert to grayscale	100%
5	Merge 1 image to another with same size	100%
6	Blur 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

```
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt

# Main function
fileName = input ("Enter name of filename")

img = Image.open(fileName)
img = np.asarray(img)

(h,w,c) = img.shape
img1d = np.reshape (img, [h*w,c])
```

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

```
# Tang do sang cho anh
def IncreaseBrightness (img1d, alpha):
    img1d = [[(255 if pixel[i] + alpha > 255 else 0 if  pixel[i] + alpha < 0 else  pixel[i] + alpha) for i in range(3)] for
    return img1d
```

- For more code detail, please check out my *20127056.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

```
# Tang do tuong phan cho anh
def IncreaseConstrast (img1d, alpha):
    img1d = [[(pixel[i] * alpha if pixel[i] * alpha < 255 else 255) for i in range(3)] for pixel in img1d]
    return np.array(img1d).astype(int)
```

- 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

```
# Lat anh ngang doc
def FlipImage (img, axis):
    if axis == 1: # Left-Right
        return np.fliplr(img)
    if axis == 2 : #Up - Down
        return np.flipud(img)
```

- 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

```
# Chuyen ve anh xam
def GrayScale (img1d) :
    img1d = [[ 0.3 * pixel[0] + 0.59 * pixel[1] + 0.11 * pixel[2] for _ in range(3)] for pixel in img1d]
    return np.array(img1d).astype(int)
```

- 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. Merge 1 image to another with same size

```
# Chong 2 anh cung kich thuoc
def MergeImage(img) :
    img = GrayScale (img) + img
    img = [ [(255 if pixel[i] > 255 else pixel[i]) for i in range(3)] for pixel in img]
    return img
```

- This function simply changes 1 img to grayscale and plus itself to grayscale version. That also means plussing 2 array with the same size.
- Here I also set the pixel value to 255 if it exceeds 255.

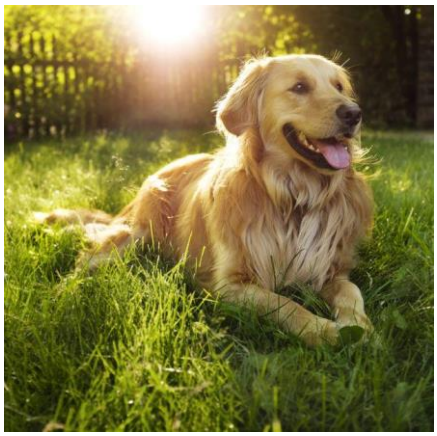
7. Blur image

```
blur = np.array([[0.0625, 0.125, 0.0625], [0.125, 0.25, 0.125], [0.0625, 0.125, 0.0625]])
```

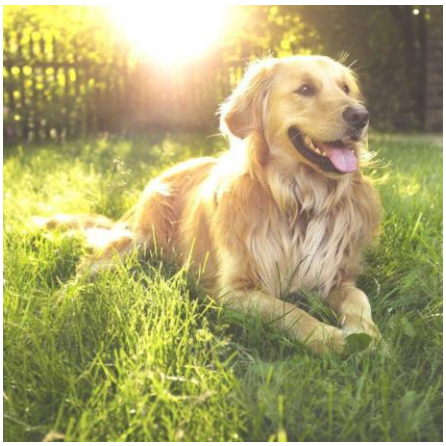
```
# Làm mờ ảnh
def GaussianBlur(img, blur) :
    (h,w,c) = img.shape
    imgRes = np.zeros ((h,w,c))

    rows = len(img)
    cols = len(img[0])

    for row in range(rows):
        for col in range(cols):
            if col - 1 >= 0 and col + 1 < cols and row - 1 >= 0 and row + 1 < rows:
                imgRes[row][col][0] =  np.sum(np.multiply(blur,img[row - 1 : row + 2, col - 1 : col + 2, 0]))
                imgRes[row][col][1] =  np.sum(np.multiply(blur,img[row - 1 : row + 2, col - 1 : col + 2, 1]))
                imgRes[row][col][2] =  np.sum(np.multiply(blur,img[row - 1 : row + 2, col - 1 : col + 2, 2]))
            else:
                imgRes[row][col] = img[row][col]
    return np.array(imgRes).astype(int)
```

- I prepare a 3x3 numpy array as Gaussian blur 3x3 described. 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.

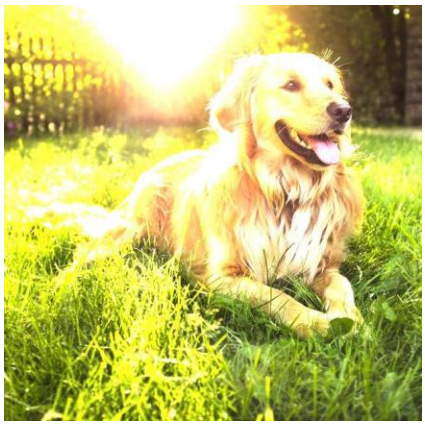
IV. Result image



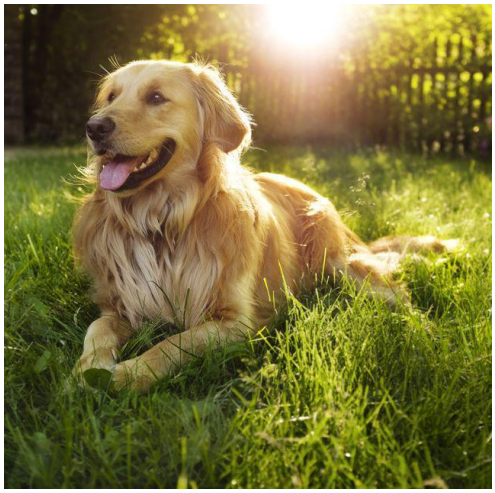
Original image with 640 x 635 pixel



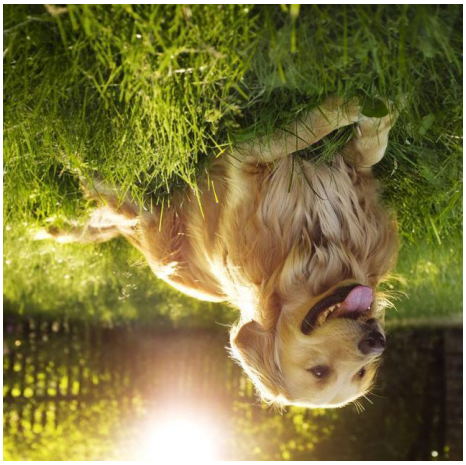
Adjusted brightness image with scale value 50



Adjusted contrast image with contrast value 2

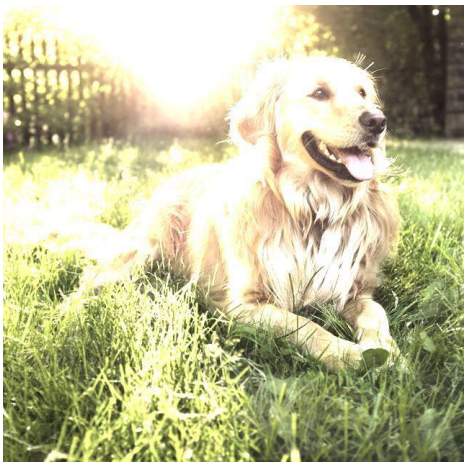


Flip image from left to right

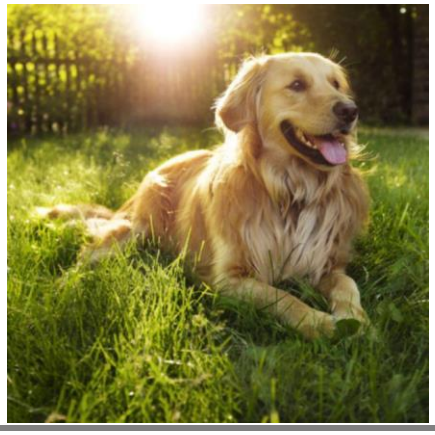


Flip image from up to down

Gray scale



Put the grayscale version on origin one



Blur 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/