HO CHI MINH CITY NATIONAL UNIVERSITY
UNIVERSITY OF NATURE SCIENCE
FACULTY OF INFORMARTION TECHNOLOGY
----------o0o----------

PROJECT 02

IMAGE PROCESSING

Applied mathematics and statistics for information technology

Lecturers:

Ph.D Vũ Quốc Hoàng
Phan Thị Phương Uyên
Nguyễn Văn Quang Huy
Trần Thị Thảo Nhi

Students:

Đặng Ngọc Tiến ID: 20127641

Thành phố Hồ Chí Minh – 07/2022

Contents

Index

I. Introduction 3

1. Personal information: 3

II. Complete progress: 3

III. Implementation idea and description 3

1. Open the image and read the image 3
2. Adjust brightness 3
3. Adjust contrast 4
4. Flip the image 5
5. Convert RGB image to grayscale image 5
6. Blend 2 images of the same size 6
7. Blur the image 7

IV. Demo 9

1. The original image: 9
2. Adjust brightness 10
3. Adjust Contrast: 11
4. Flip image 12
5. Convert RGB image to grayscale image 13
6. Blend 2 images: 14
7. Blur the image: 15
8. Circular mask 16
9. Circular mask 17

V. References 18

I. Introduction

1. Personal information:

Name: Đặng Ngọc Tiến

ID student: 20127641

II. Complete progress:

Functions	Complete
Adjust Brightness	100%
Adjust Contrast	100%
Flip the image (Vertical / Horizontal)	100%
Convert RGB image to grayscale image	100%
Blend 2 images of the same size	100%
Blur the image	100%
Advanced	
Circular mask	100%
Ellipse mask	100%

III. Implementation idea and description

1. Open the image and read the image

Use the open() function in the PIL library to open the image. Then use the array() function in the Numpy library to convert the image to an array.

2. Adjust brightness

Implementation idea

- To increase the image brightness by 1 degree $k \in [-255; 255]$, we add each pixel by that k scalar.
- With pixels $v = [v_1\ v_2\ v_3]$, increase the brightness, the new pixels $v' = [v_1 + k\ \ v_2 + k\ \ v_3 + k]$
- In case the pixel after being added is overflowed from the RGB value range of $[0, 255]$, we force the image or scalar to int16. If after adding:
  - $v_i < 0$ thì $v_i = 0$

- $v_i > 255$ thì $v_i = 255$

Input: - img: input image with data type np.array

- brightness: is the brightness of the image.

Output: - output image with data type np.uint8 (int (0 to 255))

Description

- Convert scalar to np.ndarray and cast to int16 so you don't have to cast the full image (dtype = np.int16).
- Increase the image array to k degrees.
- Use the np.clip() function in the numpy library to limit the array elements to the range 0 to 255, because the color values in the image array are only within the range 0 to 255.

3. Adjust contrast

Implementation idea

- To increase the contrast by 1 degree $k \in [-255; 255]$, we multiply each pixel by the scalar factor.
- To calculate a contrast correction factor which is given by the following formula: $factor = \frac{259(k+255)}{255(259-k)}$
- In order for the algorithm to function correctly the value for the contrast correction factor (*F*) needs to be stored as a floating point number and not as an integer. The value *C* in the formula denotes the desired level of contrast.
- The next step is to perform the actual contrast adjustment itself. The following formula shows the adjustment in contrast being made to the red component of a colour: $R' = F(R - 128) + 128$
- In case the pixel after being added is overflowed from the RGB value range of $[0, 255]$, we force the image or scalar to int16. If after adding:
  - $v_i < 0$ thì $v_i = 0$
  - $v_i > 255$ thì $v_i = 255$

Input: - img: input image with data type np.array

- contrast: is the contrast of the image

Output: - output image with data type np.uint8 (int (0 to 255))

Description

- Calculate factor according to the above formula
- Convert image array to float
- Use the np.clip() function in the numpy library to limit the array elements to the range 0 to 255, because the color values in the image array are only within the range 0 to 255.

4. Flip the image

Implementation idea

The image is saved using a pixel-by-pixel matrix, so just flip the matrix in the desired direction and we will get the flipped image

Input: - img: input image with data type np.array

- direction: vertical or horizontal

Output: - output image with data type np.array

Description

- Use the np.fliplr() function in the Numpy library to flip the image horizontally
- Use the np.flipud() function in the Numpy library to flip the image vertical

5. Convert RGB image to grayscale image

Implementation idea

There are 2 methods to convert color image to grayscale image:

- Average method
- Weighted method

In this project i use weighted method.

Red color has more wavelength of all the three colors, and green is the color that has not only less wavelength then red color but also green is the color that gives more soothing effect to the eyes.

It means that we have to decrease the contribution of red color, and increase the contribution of the green color, and put blue color contribution in between these two.

Grayscale image = ((0.3 * R) + (0.59 * G) + (0.11 * B))

Input: img: input image with data type np.array

Output: output image with data type np.uint8 (int (0 to 255))

Description

- img[..., 3] to slide each pixel, then use np.dot in Numpy library to find the dot product of the pixel with the weights
- Because the gray image has only one color channel, to display it on a notebook, we use the parameter cmap='gray' of matplotlib.pyplot

6. Blend 2 images of the same size

Implementation idea

- When we talk about images, we know its all about the matrix either binary image(0, 1), gray scale image(0-255) or RGB image(255 255 255). So additions of the image is adding the numbers of two matrices
- But sometimes we do not want to perform simple addition in image, so in this case we have blending. This is also image addition, but different weights are given to images so that it gives a feeling of blending or transparency. Images are added as per the equation below :
    - g(x) = (1 - a)f(x) + af1(x)

Input:
- img1: input grayscale image with data type np.array
- img2: input grayscale image with data type np.array
- alpha: image transparency (0.0 to 1.0)

Output: - output image with data type np.uint8 (int (0 to 255))

Description

- Use addition to blend 2 images
- Use multiplication to adjust transparency
- Because of using gray image, must use cmap="gray" to display the results.

7. Blur the image

Implementation idea and description

There are two steps to this process:

- Create a Gaussian Kernel/Filter
- Perform Convolution and Average

Gaussian Kernel/Filter:

Create a function named, which takes mainly two parameters. The size of the kernel and the standard deviation.

Create a vector of equally spaced numbers using the size argument passed.

Now we will call the function which returns the density using the standard deviation.

Then we will create the outer product and normalize to make sure the center value is always 1.

The formula of a Gaussian function is:

$$f(x) = \frac{1}{\sqrt{2\pi}} . e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Perform Convolution

Convolution is the process of adding each element of the image to its local neighbors, weighted by the kernel. This is related to a form of mathematical convolution. The matrix operation being performed—convolution—is not traditional matrix multiplication.

Input: - img: input image with data type np.array

- Kernel_size

Output: - output image with data type np.uint8 (int (0 to 255))

Description

Gaussian Kernel/Filter:

- Dnorm() function calculates the gauss formula above
- Use np.linspace() in the numpy library to scale the values
- Use np.outer() in numpy library to convert 1-way kernel to 2-way kernel
- Normalize the kernel so that the sum of the elements in the kernel is equal to 1 by multiplying each element by the reciprocal of this sum

Perform Convolution and Average:

- np.lib.stride_tricks.as_strided in numpy library to get view from array with required size
- np.einsum in numpy library to multiply the kernel scalar with the view taken above
- np.dstack in library to merge color channels into an image

IV. Demo

1. The original image:

Input image name

Menu

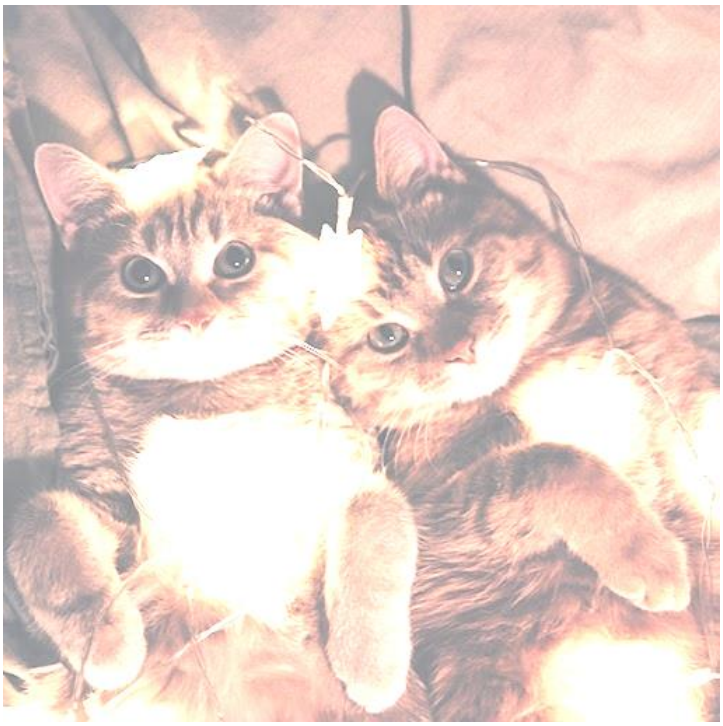
```
0. Do it all !
1. Adjust brightness
2. Adjust contrast
3. Flip image (vertical or horizontal)
4. Convert to grayscale
5. Blend images
6. Blur image
7. Circular mask
8. Elipse mask
9. Exit
```

Your choice.

2. Adjust brightness

- Input: brightness
- Brightness = -128

- Brightness = 128



3. Adjust Contrast:

Input: contraist

- Contraist = -128

- Contraist = 128

4. Flip image

Input:

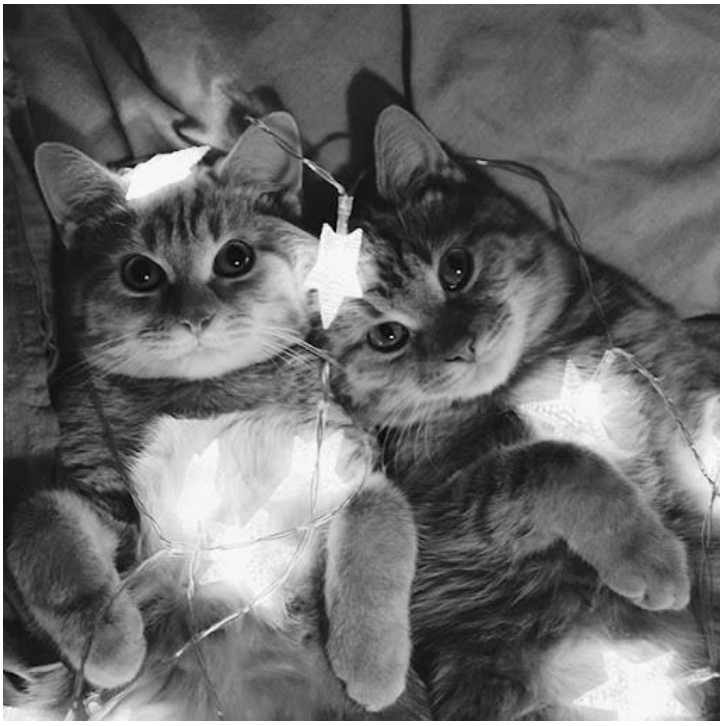
- 1. Vertical
- 2. Horizontal

Vertical

Horizontal

5. Convert RGB image to grayscale image

Output:



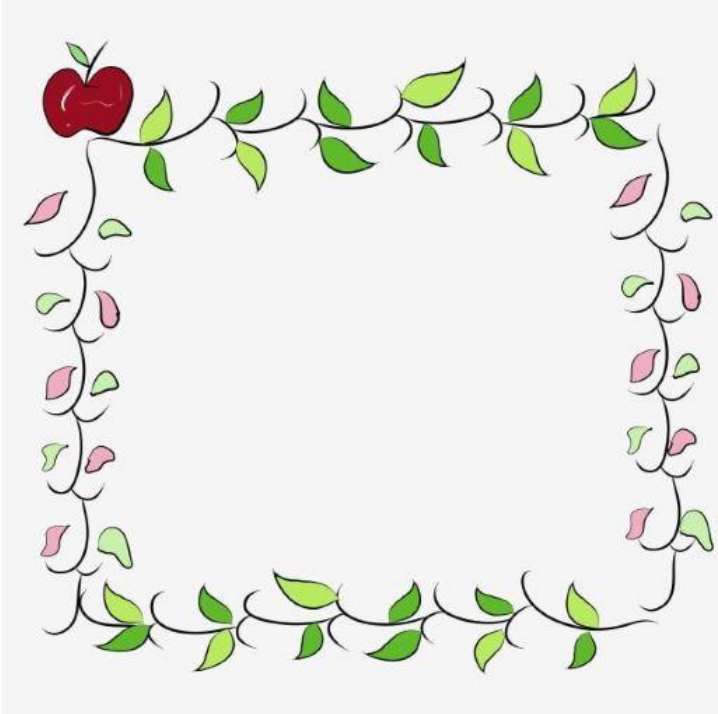
6. Blend 2 images:

Input: Name of 2 images.

Alpha = 0,5

Image 1:

Image 2:



Output:

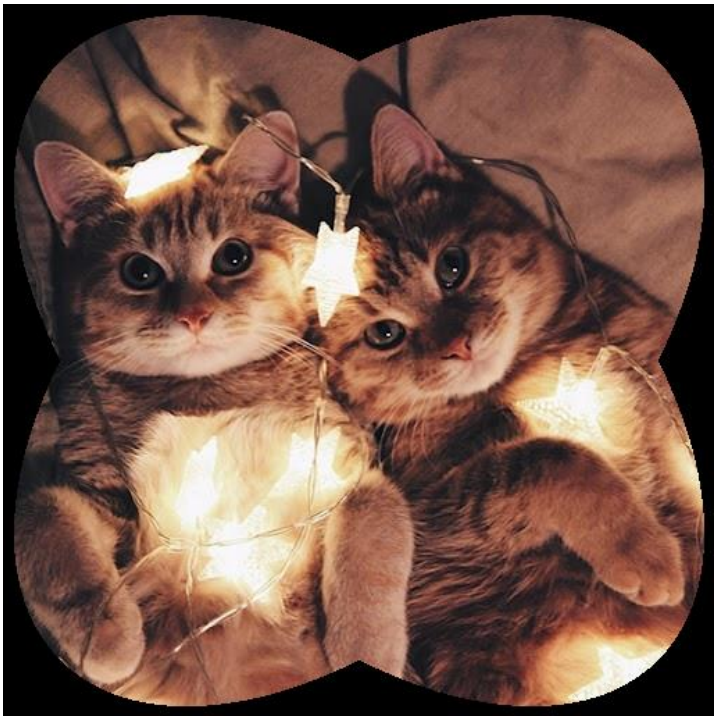
7. Blur the image:

Input: kernel size

Kernel size = 15

8. Circular mask

9. Circular mask



V. References

- Adjust brightness:

  Image Processing Algorithms Part 4: Brightness Adjustment | Dreamland Fantasy Studios (dfstudios.co.uk)

- Adjust contrast:

  Image Processing Algorithms Part 5: Contrast Adjustment | Dreamland Fantasy Studios (dfstudios.co.uk)

- Flip the image:

  numpy.flipud — NumPy v1.23 Manual

  numpy.fliplr — NumPy v1.23 Manual

- Convert RGB image to grayscale image

  Grayscale to RGB Conversion (tutorialspoint.com)

- Blend 2 images

  python - Image blending in numpy returns plain white image - Stack Overflow

- Blur the image

  Applying Gaussian Smoothing to an Image using Python from scratch - A Developer Diary

  Gaussian blur - Wikipedia

  numpy.einsum — NumPy v1.23 Manual

  numpy.dstack — NumPy v1.23 Manual

- Advanced

  [FIXED] How can I create a circular mask for a numpy array? ~ PythonFixing