BỘ GIÁO DỤC VÀ ĐÀO TẠO
ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

----------※※※----------

BÁO CÁO

Toán Ứng Dụng Và Thống Kê Cho Công Nghệ Thông Tin

Project 2
IMAGE PROCESSING

Giảng viên lý thuyết: Vũ Quốc Hoàng

Giảng viên hướng dẫn thực hành: Phan Thị Phương Uyên, Nguyễn Văn Quang Huy, Ngô Đình Hy

Thành phố Hồ Chí Minh - 2023

MỤC LỤC

I. Thông tin sinh viên:

Tên	MSSV	Lớp
TÔ KHÁNH LINH	21127638	21CLC08

II. Các chức năng đã hoàn thành

Số thứ tự	Chức năng	Mức độ hoàn thành
1	Thay đổi sáng, tương phản hình ảnh	100%
2	Thay đổi tương phản hình ảnh	100%
3	Lật ảnh (ngang - dọc)	100%
4	Chuyển đổi RGB thành ảnh xám và sepia	100%
5	Làm mờ và sắc nét ảnh	100%
6	Cắt ảnh theo kích thước (cắt ở trung tâm)	100%
7	Cắt ảnh theo khung hình tròn	100%
8	Viết hàm main	100%

Mức độ hoàn thành: 100%

III. Mô tả hàm chức năng và ý tưởng

1. Thay đổi độ sáng : change_brightness()

- **Ý tưởng thực hiện:** Ta cộng một số alpha vào từng pixel của từng kênh màu để tăng độ sáng cho ảnh. alpha âm thì hạ độ sáng còn alpha thì tăng độ sáng

nhưng alpha phải duy trì trong khoảng (-255;255) và pixel quá 255 thì lấy 255 dưới 0 thì lấy 0

- **Mô tả hàm:**
  - + **Input: image** = Hình ảnh đã được mở sẵn, **alpha** = Độ sáng thay đổi
  - + **Output:** Mảng 3 hình ảnh đã thay đổi độ sáng
- Cho người dùng nhập alpha sau đó dùng hàm np.clip để alpha chỉ nằm trong 0 đến 255 sau đó đổi ảnh sang dạng mảng.
- Tiếp theo, ta cộng một số alpha vào từng phần tử pixel của mảng.

```
    bright_image=img+alpha
```

- Sau đó dùng np.clip() để tránh ảnh ra khỏi vùng 0 đến 255.

```
    bright_limited=np.clip(bright_image,0,255)
```

- Sau đó ta trả về kết quả:

```
    return bright_limited.astype(np.uint8)
```

2. Thay đổi tương phản hình ảnh: change_contrast()

- **Ý tưởng:** của code là sử dụng một hệ số tương phản factor để thay đổi giá trị màu của từng pixel trong hình ảnh bằng công thức ở dưới. Hệ số tương phản này được tính toán dựa trên giá trị alpha đầu vào. Và dùng hệ số tương phản trên để tính từng pixel bằng công thức ở dưới.
- **Mô tả hàm:**
  - + Input: **image** = Hình ảnh đã được mở sẵn, **alpha** = Độ sáng tương phản
  - + Output: Mảng 3 hình ảnh đã thay đổi tương phản

- Ta tính factor từ alpha bằng công thức:

$$factor = \frac{259*(255+alpha)}{255*(259-alpha)}$$

```
    factor= (259*(255+alpha))/(255*(259-alpha))
```

- Sau đó ta cho pixel tương phản theo công thức:
  $pixel_{tương\ phản} = pixel_{hình\ gốc} * factor - 128 * factor + 128$

```
contrast=np.clip(img*float(factor)-128*float(factor)+128,0,2
55)
```

- Sau đó ta trả về kết quả:

```
return contrast.astype(np.uint8)
```

3. Lật ảnh (ngang - dọc): flip_img()

- **<u>Ý tưởng:</u>** Sử dụng hàm np.flip() để quay hình dọc và ngang.
- Với ngang axis=1, hàm np.flip() hoán đổi bằng cách hoán đổi cột i với cột width-1-i với i từ 0 đến width (width là chiều rộng ảnh).
- Với dọc axis=0, hàm np.flip()hoán đổi bằng cách hoán đổi cột height-1-i với i từ 0 đến height (height là chiều rộng ảnh).

a. Mô tả hàm:

+ Input: **img** = Hình ảnh đã được mở sẵn, **mode**= 0 hoặc 1 với (0 là dọc và 1 là ngang)
+ Output: Mảng 3 hình ảnh đã chuyển sang dọc hoặc ngang theo yêu cầu.

- Sử dụng hàm np.flip với 0 hoặc 1 với (0 là dọc và 1 là ngang):

```
      flip_img=np.flip(img,mode)
```

- Sau đó trả về kết quả:

```
return flip_img.astype(np.uint8)
```

4. Chuyển đổi RGB thành ảnh xám và sepia:

a. Chuyển RGB thành ảnh xám: gray_scale()

- Lấy giá trị pixel trung bình của 3 kênh màu để lấy

pixel của kênh grayscale.

- Kết hợp độ sáng và độ chói mà từng kênh màu đóng góp để tạo thành một màu xám hoàn chỉnh.
- **Mô tả hàm:**
  + Input: **img** = Hình ảnh đã được mở sẵn.
  + Output: Mảng 3 chiều của hình ảnh sau khi được đổi thành grayscale.
- Tính trung bình của 3 kênh màu và cho nó vào mảng gray_img (1 chiều vì grayscale chỉ có một kênh màu biểu thị độ xám).

```
    gray_img=np.mean(img,axis=2)
```

- Trả về kết quả gray_img:

```
    return gray_img.astype(np.uint8)
```

b. Chuyển ảnh sang ảnh sepia: sepia()

- **Ý tưởng:** Đổi các chiều RGB lần lượt thành như sau ta được hình một hình có filter sepia:

  newRed = 0.393R + 0.769G + 0.189B
  newGreen = 0.349R + 0.686G + 0.168B
  newBlue = 0.272R + 0.534G + 0.131B

- Với newRed là kênh màu 1, newGreen là kênh màu 2, newBlue là kênh màu 3.
- **Mô tả hàm:**
  + **Input: img** = Hình ảnh đã được mở sẵn.
  + **Output:** Mảng 3 chiều của hình ảnh sau khi được đổi thành sephia.
- R,G,B lần lượt là kênh màu 1,2,3:

```
    R,G,B=sepia_img[:,:,0],sepia_img[:,:,1],sepia_img[:,:,2]
```

- Áp dụng công thức trên cho từng kênh màu:

```
    sepia_img[:, :, 0]=0.393*R+0.769*G+0.189*B
    sepia_img[:, :, 1]=0.349*R+0.686*G+0.168*B
    sepia_img[:, :, 2]=0.272*R+0.534*G+0.131*B
```

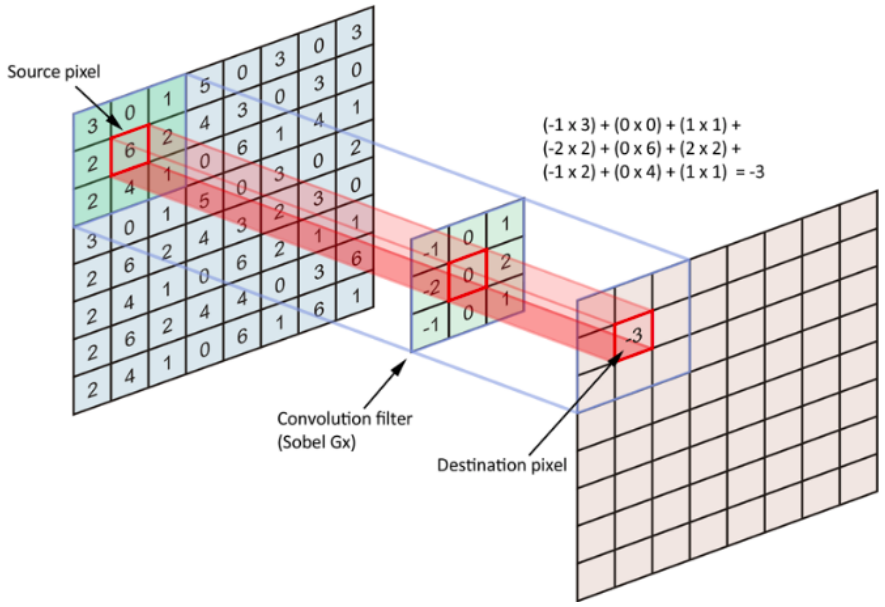
- Trả về kết quả:

```
return sepia_img.astype(np.uint8)
```

5. Làm mờ và sắc nét ảnh

a. Hàm chập (convolution): convolve()

- **Ý tưởng**: Sử dụng phép tính chập (Convolution) với một ma trận kernel Gaussian blur và Gaussian sharp để làm mờ và làm sắc ảnh.
- Để áp dụng Convolution ta tạo một mảng padding để thêm padding [0, 0, 0] vào xung quanh ma trận ảnh.
- Xét mỗi pixel trong ma trận ảnh lấy các 8 pixel xung quanh ma trận (bao gồm padding) theo dạng (3x3) rồi nhân với kernel, sau đó cộng hết các giá trị đã nhân lại ta được một pixel làm mờ. Cụ thể như hình dưới đây:



- Sau khi làm mờ tất cả các pixel, ta trả về mảng đó.
- **Mô tả hàm:**
  + **Input: img** = mảng hình ảnh **filter=** kernel áp dụng, **dim**= số chiều của mảng.
  + **Output:** Mảng 3 chiều của hình ảnh sau khi được áp dụng kernel (filter).

- Nếu ma trận này có 3 kênh màu RGB:
  + Tạo một mảng padding toàn ma trận không với kênh màu tương ứng hình và kích thước chiều

dài và chiều rộng đều lớn hơn ma trận gốc 2.Sau đó padding xung quanh ma trận đó mảng [0,0,0] với độ dày 1. Ở trong lớp đó là pixel gốc của ảnh:

```
     padding=np.zeros((img.shape[0]+2,img.shape[1]+2,
img.shape[2]))
    padding= np.pad(img, ((1, 1), (1, 1), (0, 0)))
```

+ Sau đó tiến hành thực hiện convolution bằng cách xét từng pixel của padding lấy pixel 3x3 xung quanh pixel đang xét nhân với filter (kernel), Sau đó cộng lại tất cả tích vừa tính.:

```
        for i in range(img.shape[0]):
                for j in range(img.shape[1]):

multiply=np.multiply(padding[i:i+3,j:j+3],filter)
                    column=np.sum(multiply,axis=1)
                    pixel=np.sum(column,axis=0)
                    img[i,j]=pixel
```

- **Nếu đây là ma trận của ảnh grayscale:**
  + Ta tạo một ma trận padding trong đó ta đặt 0 xung quanh ma trận gốc:

```
    padding=np.zeros((img.shape[0]+2,img.shape[1]+2))
    padding[1:-1,1:-1]=img
```

  + Sau đó tiến hành thực hiện convolution bằng cách xét từng pixel của padding lấy pixel 3x3 xung quanh pixel đang xét nhân với filter (kernel), Sau đó cộng lại tất cả tích vừa tính:

```
        for i in range(img.shape[0]):
                for j in range(img.shape[1]):
                    multiply =
np.sum(padding[i:i+3,j:j+3] * filter)
                    img[i,j]=multiply
```

- Giới hạn pixel chỉ ở (0,255) và trả về kết quả img:

```
    img=np.clip(img, 0, 255)
    return img
```

b. Hàm làm mờ ảnh: blur()

- **Ý tưởng:** Ta lấy ma trận Gaussian 3x3 theo yêu cầu tương ứng ở đây là ma trận Gaussian 3x3 blur rồi truyền vào hàm convolution.
- **Mô tả hàm:**
  + **Input: img** = Hình ảnh đã được mở sẵn,
  + **Output:** Mảng 3 chiều của hình ảnh sau khi được làm mờ.
- Khởi tạo kernel với mảng RGB:

```
        kernel = np.array([[[1], [2], [1]],
                           [[2], [4], [2]],
                           [[1], [2], [1]]]) / 16
```

- Khởi tạo kernel với mảng grayscale:

```
kernel = np.array([[1, 2, 1],
                   [2, 4, 2],
                   [1, 2, 1]]) / 16
```

- Chuyển img sang mảng 3 chiều, sau đó chuyển mảng sang int để tránh tràn số. Tiếp theo, ta truyền mảng img và kernel vào hàm convole trả về là blur_img:

```
    img=img_3d(img)
    img=img.astype(int)
    blur_img=convolve(img,kernel)
```

- Trả về kết quả:

```
return blur_img.astype(np.uint8)
```

c. Hàm làm sắc nét ảnh: sharp()

- **Ý tưởng:** Ta lấy ma trận Gaussian 3x3 theo yêu cầu tương ứng ở đây là ma trận Gaussian 3x3 sharp rồi truyền vào hàm convolution.
- **Mô tả hàm:**
  + **Input: img** = Hình ảnh đã được mở sẵn,
  + **Output:** Mảng 3 chiều của hình ảnh sau khi được làm sắc nét.
- Khởi tạo kernel với mảng RGB:

```
        kernel = np.array([[[0], [-1], [0]],
                           [[-1],[5], [-1]],
                           [[0], [-1], [0]]])
```

- Khởi tạo kernel với mảng grayscale:

```
kernel = np.array([[0, -1, 0],
                   [-1,5, -1],
                   [0, -1, 0]])
```

- Chuyển img sang mảng 3 chiều, sau đó chuyển mảng sang int để tránh tràn số. Tiếp theo, ta truyền mảng img và kernel vào hàm convole trả về là sharp_img:

```
    img=img_3d(img)
    img=img.astype(int)
    sharp_img=convolve(img,kernel)
```

- Trả về kết quả:

```
    return sharp_img.astype(np.uint8)
```

6. Cắt ảnh theo kích thước (cắt ở trung tâm): crop_center()

- **Ý tưởng:** Tìm ra trung tâm của ảnh sau đó, cắt ảnh bằng cách sửa chiều dài của ảnh bắt đầu từ chiều dài trừ kích thước cắt ảnh đến chiều dài cộng kích thước cắt ảnh, tương tự với chiều rộng.
- Lưu ý: ta sẽ chia kích thước cắt ảnh dài rộng thành 2 để sau khi ảnh hoàn thành, ảnh sẽ có chiều dài và rộng bằng kích thước ta nhập vào.
- **Mô tả:**
  - + **Input: img** = Hình ảnh đã được mở sẵn, **cropX**= Kích thước chiều rộng muốn cắt, **cropY**= kích thước chiều dài muốn cắt.
  - + **Output:** Mảng 3 chiều của hình ảnh sau khi được đã cắt.
- Tìm trung tâm chiều dài và chiều rộng của ảnh bằng cách lấy phần nguyên của chiều dài và rộng chia hai:

```
    h1=h//2
    w1=w//2
```

- Sau đó chia chiều rộng và chiều dài ảnh (sizeX, size

Y)sau khi cắt ra làm hai:

```
    cropX=cropX//2
    cropY=cropY//2
```

- Nếu chiều rộng và dài của ảnh sau khi cắt mà lớn hoặc nhỏ hơn 0 và lớn hơn ảnh gốc ta báo lỗi và trả về ảnh thường gốc:

```
    if(h1-cropY<=0 or h1+cropY>h or w1-cropX<=0 or
w1+cropX>w):
        print("Out of size! Return original pic.")
        return img.astype(np.uint8)
```

- Ảnh được cắt sẽ có pixel từ trung tâm - kích thước X,Y cắt đến trung tâm + kích thước cắt X,Y.

```
    center_img=img[h1-cropY:h1+cropY,w1-cropX:w1+cropX,:]
```

- Trả về kết quả:

```
return center_img.astype(np.uint8)
```

7. Cắt ảnh theo khung hình tròn: crop_circle()

- **<u>Ý tưởng:</u>** Tính khoảng cách từ từng điểm vị trí pixel đến tâm. Nếu lớn hơn hoặc bằng bán kính điểm đó sẽ có giá trị bằng 0 nghĩa là pixel đó màu đen.
- **Mô tả hàm:**
  - + **Input: img** = Hình ảnh đã được mở sẵn.
  - + **Ouput:** Mảng 3 chiều của hình ảnh sau khi được cắt theo khung tròn
- Tính trung tâm của ảnh:

```
    cen_h=h//2
    cen_w=w//2
```

- Tính khoảng cách euclidean từ tâm đến từng vị trí x và y của pixel:

```
    euclidean_dis=(y-cen_h)**2+(x-cen_w)**2
```

- Lấy từng điểm nằm ngoài hình tròn tức những điểm có khoảng cách đến tâm lớn hơn hoặc bằng bán kính:

```
    circle=(euclidean_dis>=r)
```

- Những điểm có chỉ số tại những điểm nằm ngoài

bán kính sẽ bằng 0 tức là pixel thành màu đen:

```
        img[circle]=0
```

- Trả về kết quả:

```
    return img.astype(np.uint8)
```

8. Hàm main:

- Hệ thống sẽ yêu cầu người dùng nhập tên file, sau đó nhập dạng file của output.
-
- Sau đó hệ thống yêu cầu người dùng nhập option từ 0-7 lựa chọn

a. Option 1:

- Option 1 thay đổi độ sáng
- Đầu tiên người dùng nhập một số alpha với điều kiện (-255<alpha<255)
- Sau đó gọi hàm change_brightness để thay đổi độ sáng truyền vào hình ảnh đã mở sẵn và số alpha. Trả về bright_image
- Mở bright_image. Sau đó save hình đã thay đổi độ sáng, dưới tên hợp lệ.

b. Option 2:

- Option 2 thay đổi độ tương phản
- Đầu tiên người dùng nhập một số alpha với điều kiện (-255<alpha<255)
- Sau đó gọi hàm change_contrast để thay đổi độ tương phản truyền vào hình ảnh đã mở sẵn và số alpha. Trả về contrast_image
- Mở contrast_image. Sau đó save hình đã thay đổi độ sáng, dưới tên hợp lệ.

c. Option 3:

- Option 3 thay đổi chiều ngang hoặc chiều dọc của hình ảnh

- Đầu tiên ta nhập mode, mode = 1 là quay ngang, mode =0 là quay dọc
- Sau đó gọi hàm flip_img và truyền img và mode vào. Trả về flip_img
- Sau đó save hình về với tên hợp lệ.

d. Option 4:

- Option 4 chuyển hình sang hình xám, hình sepia.
- Truyền img vào hàm gray_scale() để truyển hình sang hình xám trả về gray_image.
- Truyền img vào hàm sepia() để chuyển hình sang sepia trả về sephia_image
- Với hình xám vì hình chỉ có 1 kênh màu nên ta không mở dưới dạng RGB còn sephia thì mở như bình thường.
- Save hai hình về dưới tên hợp lệ

e. Option 5:

- Option 5 làm mờ hình và làm hình sắc nét.
- Gọi hàm blur() để làm mờ hình trả về blur_img
- Gọi hàm sharp() để làm sắc nét hình trả về sharp_img.
- Save hai hình về dưới dạng RGB và dưới tên hợp lệ.

f. Option 6:

- Option 6 dùng để cắt hình.
- Ta nhập chiều rộng ảnh (sizeX)và chiều dài (sizeY) ảnh sau khi cắt
- Sau đó gọi hàm crop_center truyền img và sizeX và sizeY, trả về crop_img.
- Sau đó lưu hình về với tên hợp lệ.

g. Option 7:

- Option 7 dùng để cắt hình dạng hình tròn

- Gọi hàm crop_circle() truyền vào img, trả về circle_img.
- Sau đó lưu hình về dưới dạng tên hợp lệ.

h. Option 0:

- Option 0 làm tất cả các option trên từ (0-7) riêng option 4 ta làm cả hai dọc và ngang.

IV. Kết quả hàm

Đầu tiên ta có ảnh gốc:

a.jpg

1. Thay đổi độ sáng ảnh (alpha =50):

a_bright.jpg

2. Thay đổi độ tương phản (alpha=50):

a_contrast.jpg

3. Quay ngang và dọc ảnh:

a_horizontal.jpg

a_vertical.jpg

4. Ảnh xám và sepia:

- Ảnh xám:

a_gray.jpg

- Ảnh sepia:

a_sepia.jpg

5. Ảnh mờ và sắc nét:

- Với ảnh RGB:
  + Mờ:

a_blur.jpg

+ Sắc nét:

a_sharp.jpg

- Với ảnh grayscale:
  + Mờ:

  + Sắc nét:

6. Cắt ảnh từ trung tâm:
   - Cắt ảnh từ trung tâm:

+ Chiều rộng: 200 chiều dài:150

a_crop.jpg

7. Cắt ảnh hình tròn:
   - Cắt ảnh hình tròn

a_circle.jpg

V. Tài liệu tham khảo

- Brightness, Contrast: Algorithms for Adjusting Brightness and Contrast of an Image – The IE Blog (nitk.ac.in)
- Grayscale: https://e2eml.school/convert_rgb_to_grayscale.html
- Sephia: https://dyclassroom.com/image-processing-project/how-to-convert-a-color-image-into-sepia-image
- Blur, Sharp:

#004 CNN Padding - Master Data Science (datahacker.rs)
Bài 5: Giới thiệu về xử lý ảnh | Deep Learning cơ bản (nttuan8.com)
https://www.youtube.com/watch?v=KuXjwB4LzSA&t=288s
Kernel (image processing) - Wikipedia

- Sepia: https://dyclassroom.com/image-processing-project/how-to-convert-a-color-image-into-sepia-image
- Circle mask: https://itecnote.com/tecnote/python-how-to-create-a-circular-mask-for-a-numpy-array/