

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN ĐHQG-HCM

KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN 2 – Image Processing

Môn Học: Toán ứng dụng & thống kê

Giảng viên:

Trần Hà Sơn

Nguyễn Văn Quang Huy

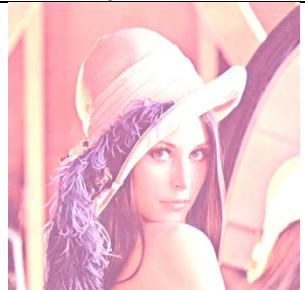
Nguyễn Đình Thúc

Nguyễn Ngọc Toàn

Sinh viên thực hiện:

Nguyễn Lê Hùng 22127135

I. Chức năng đã hoàn thành

0	Ảnh ban đầu		
STT	Chức năng	Ghi chú	Mức độ hoàn thành
1	Thay đổi độ sáng	Độ sáng 100	
2	Thay đổi độ tương phản	Độ tương phản +1.5	
3.1	Lật ảnh ngang		
3.2	Lật ảnh dọc		

4.1	RGB thành ảnh xám		
4.2	RGB thành ảnh sepia		
5.1	Làm mờ ảnh		
5.2	Làm sắc nét ảnh		
6	Cắt ảnh theo kích thước	250,250	

7.1	Cắt ảnh theo khung tròn		
7.2	Cắt ảnh theo khung elip		
8	Hàm main		100%
9	Phóng to/Thu nhỏ 2x	Hệ số = 2	100%

II. Ý tưởng thực hiện và mô tả các hàm

1. Thay đổi độ sáng cho ảnh:

-*Ý tưởng*: cộng thêm giá trị cho các phần tử của ma trận, có thể là dương hoặc âm, nếu là giá trị dương thì ảnh sẽ sáng hơn vì gần số biểu thị màu trắng là 255, nếu là giá trị âm thì ảnh sẽ tối đi vì gần với số biểu thị màu đen là 0.

-*Mô tả hàm chức năng*: `change_exposure(image, factor)`

+Hàm `change_exposure` nhận ảnh đầu vào được chuyển dưới dạng ma trận vector màu của ảnh bằng mảng NumPy chứa giá trị vector với kiểu dữ liệu ‘int16’ để đảm bảo rằng giá trị phơi sáng có thể chứa các giá trị lớn hơn 255 hoặc nhỏ hơn 0 nếu cần.

+Sau khi cộng giá trị ‘factor’ vào từng giá trị pixel của ảnh. Việc cộng này dẫn đến một sự thay đổi tuyến tính trong độ sáng của ảnh.

+Sau khi điều chỉnh giá trị pixel, hàm sử dụng ‘`np.clip`’ để đảm bảo rằng tất cả các giá trị pixel đều nằm trong khoảng từ 0 đến 255. Điều này là cần thiết vì như đã trình bày trong phần ý tưởng.

+Cuối cùng, kiểu dữ liệu của mảng được chuyển đổi về uint8 để phù hợp với định dạng của ảnh và đầu ra trả ảnh sau khi thực hiện chức năng dung hàm ‘`Image.fromarray`.’

2. Thay đổi độ tương phản cho ảnh:

-*Ý tưởng*: nhân thêm giá trị (factor) cho các phần tử trong ảnh. Độ tương phản của ảnh phản ánh sự khác biệt giữa các giá trị pixel sáng và tối. Khi tăng độ tương phản, các vùng sáng sẽ trở nên sáng hơn và các vùng tối sẽ trở nên tối hơn. Khi giảm độ tương phản, ảnh sẽ trở nên mờ nhạt hơn.

-*Mô tả hàm chức năng*: `change_contrast(image, factor)`:

+Tính giá trị trung bình của tất cả các pixel trong ảnh ‘`np.array(image)`’, chuyển đổi ảnh sang mảng NumPy

+ axis=(0, 1) chỉ định rằng phép tính trung bình sẽ được thực hiện dọc theo cả hai chiều của mảng ảnh, nghĩa là trung bình của toàn bộ ảnh.

+ ‘`keepdims=True`’ đảm bảo rằng mảng kết quả vẫn giữ nguyên số chiều như mảng đầu vào, để phép toán tiếp theo được thực hiện một cách chính xác.

+ Mỗi giá trị pixel trong mảng ảnh được điều chỉnh dựa trên công thức:

`(np_image - mean) * factor + mean.`

+ * factor: Nhân giá trị đã chuyển về trung tâm với hệ số tương phản. Nếu factor > 1, độ tương phản sẽ tăng; nếu factor < 1, độ tương phản sẽ giảm.

+Phương thức trả về giống hàm điều chỉnh độ sáng

3. Thay đổi độ tương phản cho ảnh

-*Ý tưởng:* Thay đổi thứ tự của các pixel trong mảng ảnh dọc theo chiều được chỉ định..

-*Mô tả hàm chức năng:* `flip_image(image, direction='horizontal')`:

- + Chuyển đổi ảnh sang mảng NumPy

- + Lật ảnh theo chiều ngang `np_image = np_image[:, ::-1]`

- + Lật ảnh theo chiều dọc `np_image = np_image[::-1, :]`

- + Phương thức trả về giống các hàm trước đó.

4. Chuyển đổi màu ảnh RGB

a) *Chuyển đổi ảnh RGB sang màu ảnh xám:*

-*Ý tưởng:* thay đổi các giá trị màu vector theo công thức màu chuyển thành ảnh xám.

-*Mô tả hàm chức năng:* `convert_to_grayscale(image)`:

- + Chuyển đổi ảnh sang mảng NumPy

- + Thực hiện phép nhân điểm (dot product) giữa các giá trị pixel của kênh màu và các trọng số [0.2989, 0.5870, 0.1140]. Các trọng số này tương ứng với độ nhạy cảm của mắt người đối với các màu đỏ, xanh lục và xanh lam. Kết quả là một mảng mới chứa các giá trị độ sáng của ảnh thang độ xám.[1]

- + Phương thức trả về giống các hàm trước đó.

b) *Chuyển đổi ảnh RGB sang màu ảnh Speia:*

-*Ý tưởng:* thay đổi các giá trị màu vector theo công thức màu chuyển thành màu ảnh Sepia

-*Mô tả hàm chức năng:* `convert_to_sepia(image)`:

- + Chuyển đổi ảnh sang mảng NumPy

- + ‘tr’, ‘tg’, và ‘tb’ lần lượt là các kênh màu đỏ, xanh lục và xanh lam của ảnh sepia. Chúng được tính toán bằng cách áp dụng các trọng số cụ thể cho các kênh màu đỏ, xanh lục và xanh lam của ảnh gốc. Thực hiện phép nhân điểm giữa các giá trị pixel của kênh màu tương tự hàm chuyển màu ảnh xám với giá trị được cho là màu của Sepia. [4]

- + Phương thức trả về giống các hàm trước đó.

5. Làm mờ/sắc nét ảnh:

a) *Làm mờ ảnh:*

-*Ý tưởng:* Hàm blur_image được thiết kế để làm mờ ảnh bằng cách sử dụng phương pháp Gaussian blur. Gaussian blur là một kỹ thuật làm mờ ảnh bằng cách áp dụng một hàm Gaussian[3] để tính toán trung bình trọng số của các pixel lân cận. Kết quả là các cạnh và chi tiết của ảnh bị làm mờ đi, tạo ra một hiệu ứng mượt mà hơn.

+ Công thức Gaussian Blur[3]

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

- Mô tả hàm chức năng: **blur_image(image)**

+ Chuyển đổi ảnh sang mảng NumPy với kiểu dữ liệu float32.

+ Tạo kernel Gaussian[7] đơn giản với một ma trận 5x5 chứa các giá trị 1/25.

Kernel này sẽ được sử dụng để làm mờ ảnh bằng cách tính trung bình trọng số của các pixel lân cận. Kernel Gaussian thường có dạng hình chuông, nhưng trong chương trình này, chúng ta sử dụng một kernel đơn giản để triển khai.

+ Ảnh được đệm **np.pad** thêm các pixel xung quanh để xử lý các pixel ở biên. mode='reflect' có nghĩa là các giá trị pixel ngoài biên sẽ được phản chiếu để làm đệm.

+ Khởi tạo mảng cho ảnh làm mờ '**blurred_image**' để lưu trữ kết quả của quá trình làm mờ

+ Áp dụng Kernel Gaussian cho từng kênh màu R,G,B. '**np.convolve**' được sử dụng để thực hiện phép tích chập (convolution) giữa mảng pixel của kênh màu và kernel. mode='same' đảm bảo rằng kết quả của phép tích chập có cùng kích thước với đầu vào.

+ Giới hạn giá trị pixel và chuyển đổi lại ảnh thành PIL.image với phương thức tương tự hàm chức năng chỉnh độ sáng.

b) *Làm sắc nét ảnh:*

- *Ý tưởng:* Tạo bộ lọc tăng cường sự khác biệt giữa các giá trị pixel liền kề, giúp làm nổi bật các cạnh và chi tiết của ảnh. Ý tưởng chính của hàm là áp dụng một kernel làm nét thông qua phép tích chập (convolution) để tính toán giá trị mới cho các pixel của ảnh.

- Mô tả hàm chức năng: **sharpen_image(image)**

+ Chuyển đổi ảnh sang mảng NumPy với kiểu dữ liệu float32.

+ Tạo kernel Gaussian[7] đơn giản với một ma trận 3x3 chứa các giá trị cụ thể.

Kernel này sẽ được sử dụng để làm nét ảnh bằng cách tăng cường sự khác biệt giữa các giá trị pixel liền kề. Giá trị trung tâm (5) đại diện cho trọng số của pixel trung tâm, trong khi các giá trị -1 đại diện cho trọng số của các pixel xung quanh.

+ Ảnh được đệm **np.pad** thêm các pixel xung quanh để xử lý các pixel ở biên.

mode='reflect' có nghĩa là các giá trị pixel ngoài biên sẽ được phản chiếu để làm đệm.

+ Khởi tạo mảng cho ảnh làm nét '**sharpened_image**' để lưu trữ kết quả của quá trình làm mờ

+ Áp dụng Kernel Gaussian cho từng kênh màu R,G,B. '**np.convolve**' được sử dụng để thực hiện phép tích chập (convolution) giữa mảng pixel của kênh màu và kernel. mode='same' đảm bảo rằng kết quả của phép tích chập có cùng kích thước với đầu vào.

+ Giới hạn giá trị pixel và chuyển đổi lại ảnh thành PIL.image với phương thức tương tự hàm chức năng chỉnh độ sáng.

6. Cắt ảnh theo kích thước (trung tâm)

- *Ý tưởng:* xác định tọa độ trung tâm của ảnh và sau đó cắt một phần của ảnh xung quanh tọa độ trung tâm đó với kích thước được xác định trước.

- *Mô tả hàm chức năng:* **crop_image_center(image, size)**

+ Chuyển đổi ảnh sang mảng NumPy

+ Xác định tọa độ trung tâm của ảnh

```
center_x, center_y = np_image.shape[1]//2, np_image.shape[0] // 2
```

Sử dụng phép chia lấy phần nguyên (//) để xác định tọa độ trung tâm của ảnh.

+ Cắt phần ảnh xung quanh trung tâm

```
cropped_image = np_image[center_y - half_size_y:center_y + half_size_y, center_x - half_size_x:center_x + half_size_x]
```

+ Phương thức trả về giống các hàm trước đó.

7.1 Cắt ảnh theo khung hình tròn

- *Ý tưởng:* Xác định tọa độ trung tâm và bán kính của hình tròn, sau đó áp dụng một mặt nạ để giữ lại các pixel nằm trong hình tròn và loại bỏ các pixel nằm ngoài.

+ Phương trình đường tròn hệ tọa độ Descartes[4]

$$(x - a)^2 + (y - b)^2 = r^2.$$

- *Mô tả hàm chức năng:* **crop_circular(image)**

+ Chuyển đổi ảnh sang mảng NumPy

+ Xác định chiều cao, chiều rộng và tọa độ trung tâm của ảnh

+ Xác định bán kính của hình tròn

+ Dùng hàm '**np.ogrid**' tạo ra các lưới tọa độ x và y với kích thước tương ứng với chiều cao chiều rộng của ảnh.

+ Tạo mặt nạ mask với công thức ở trên và áp dụng mặt nạ để loại bỏ các pixel

+ Phương thức trả về giống các hàm trước đó.

7.2 Cắt ảnh theo khung elip chéo nhau

-*Ý tưởng:* Viết hàm riêng nhằm tạo ra 1 mặt nạ với hai hình elip chéo nhau. Mặt nạ này sau đó có thể được sử dụng để cắt ảnh theo hình dạng của hai elip chéo nhau. Ý tưởng chính của hàm là xác định tọa độ trung tâm của ảnh và tạo ra các phương trình của hai hình elip, sau đó kết hợp chúng lại để tạo thành một mặt nạ hình elip kép.

+ Phương trình hình elip có trục chính nằm ngang và trục phụ nằm dọc trong hệ tọa độ Descartes với tâm tại (x_0, y_0) là[5]

$$\frac{(x - x_0)^2}{a^2} + \frac{(y - y_0)^2}{b^2} = 1 .$$

-*Mô tả hàm chức năng: `crop_elliptical_mask(image)`*

- + Xác định lưới tọa độ y và x. với hàm '`np.ogrid`'
- + Xác định trục lớn, trục nhỏ của elip trong đó 'a' và 'b' là các trục chính, trục phụ của hình elip, được xác định dựa trên kích thước của ảnh và hệ số tỉ lệ 125%
- + Tạo mặt nạ cho hình elip thứ nhất, 'rotate1' là góc xoay 45 độ ($\pi/4$ radian).

```
rotate1 = np.pi / 4
dist_from_F1 = ((X - center[0]) * np.cos(rotate1) + (Y - center[1]) * np.sin(rotate1))**2 / a
dist_from_F2 = ((X - center[0]) * np.sin(rotate1) - (Y - center[1]) * np.cos(rotate1))**2 / b
ellipse_mask1 = dist_from_F1 + dist_from_F2 <= 1
```

+ Tạo mặt nạ cho hình elip thứ hai

```
rotate2 = -np.pi / 4
dist_from_F3 = ((X - center[0]) * np.cos(rotate2) + (Y - center[1]) * np.sin(rotate2))**2 / a
dist_from_F4 = ((X - center[0]) * np.sin(rotate2) - (Y - center[1]) * np.cos(rotate2))**2 / b
ellipse_mask2 = dist_from_F3 + dist_from_F4 <= 1
```

+ Kết hợp hai mặt nạ sử dụng phép cộng

-*Mô tả hàm chức năng: `crop_double_elliptical(image)`*

- + Chuyển đổi ảnh sang mảng NumPy
- + Tạo mặt nạ hình kép sử dụng hàm `crop_elliptical_mask(image)`

```
h, w = np_image.shape[:2]
ellipse_mask = create_elliptical_mask(h, w)
```

+ Tạo mặt nạ mask với công thức ở trên và áp dụng mặt nạ để loại bỏ các pixel

+ Phương thức trả về giống các hàm trước đó.

9. Phóng to/thu nhỏ ảnh:

-*Ý tưởng:* Ý tưởng chính của hàm là sử dụng phương pháp nội suy bilinear để thay đổi kích thước của ảnh theo một hệ số cho trước. [6]

-*Mô tả hàm chức năng: `bilinear_interpolate(image, x, y)`*

- + Tính toán các giá trị pixel xung quoanh (4 pixel lân cận)
- + Sử dụng các trọng số để nội suy tuyến tính giá trị của pixel mới.

```
def bilinear_interpolate(image, x, y):
    x0 = int(np.floor(x))
    x1 = min(x0 + 1, image.shape[1] - 1)
    y0 = int(np.floor(y))
    y1 = min(y0 + 1, image.shape[0] - 1)

    Ia = image[y0, x0]
    Ib = image[y1, x0]
    Ic = image[y0, x1]
    Id = image[y1, x1]

    wa = (x1 - x) * (y1 - y)
    wb = (x1 - x) * (y - y0)
    wc = (x - x0) * (y1 - y)
    wd = (x - x0) * (y - y0)

    return wa * Ia + wb * Ib + wc * Ic + wd * Id
```

- Mô tả hàm chức năng: `resize_image_bilinear(image, new_width, new_height)`

- + Xác định tỷ lệ giữa kích thước gốc và kích thước mới.
- + Sử dụng hàm `bilinear_interpolate` để nội suy giá trị của từng pixel mới dựa trên vị trí tương ứng trong ảnh gốc.

```
def resize_image_bilinear(image, new_width, new_height):
    src_height, src_width, num_channels = image.shape
    resized_image = np.zeros((new_height, new_width, num_channels), dtype=np.uint8)

    x_ratio = src_width / new_width
    y_ratio = src_height / new_height

    for y in range(new_height):
        for x in range(new_width):
            src_x = x * x_ratio
            src_y = y * y_ratio
            for c in range(num_channels):
                resized_image[y, x, c] = bilinear_interpolate(image[:, :, c], src_x, src_y)

    return resized_image
```

a) *Phóng to ảnh:*

- *Ý tưởng:* Ý tưởng chính của hàm là sử dụng phương pháp nội suy đã cài đặt ở trên để thay đổi kích thước của ảnh theo một hệ số cho trước.

- *Mô tả hàm chức năng: `zoom_in(image, factor)`*

+ Chuyển đổi ảnh sang mảng NumPy.

+ Tính toán kích thước mới của ảnh sau khi phóng to sử dụng phép nhân với hệ số
`new_h, new_w = int(h * factor), int(w * factor)`

+ Sử dụng hàm '`resize_image_bilinear`' để thay đổi kích thước ảnh

+ Phương thức trả về giống các hàm trước đó

b) *Thu nhỏ ảnh:*

- *Ý tưởng:* Ý tưởng chính của hàm là sử dụng phương pháp nội suy (interpolation) để thay đổi kích thước của ảnh theo một hệ số cho trước.

- *Mô tả hàm chức năng: `zoom_out(image, factor)`*

+ Chuyển đổi ảnh sang mảng NumPy.

+ Tính toán kích thước mới của ảnh sau khi phóng nhỏ sử dụng phép chia với hệ số
`new_h, new_w = int(h / factor), int(w / factor)`

+ Sử dụng hàm '`resize_image_bilinear`' để thay đổi kích thước ảnh

+ Phương thức trả về giống các hàm trước đó

8. Hàm main

- Chương trình sẽ cho chọn các chức năng theo yêu cầu từ 1-11 tương ứng (được
nêu rõ khi chạy hàm main) và các giá trị đầu vào do người dùng nhập. Đối với lựa chọn
0, chương trình sẽ thực hiện toàn bộ chức năng trong chương trình, với đầu vào được mặc
định như mục I, đầu ra là tên tương ứng như mục III. Sau khi chọn chức năng, chương
trình sẽ hỏi tiếp tục thực hiện (chọn 0), hay dừng lại (chọn 1).

III. Hình ảnh kết quả với từng chức năng

1. Hình 2: Kích thước 728 x 410 pixels

- Input: test2.png
- Chức năng thay đổi độ sáng cho ảnh
- Đầu ra: test2_exposure.png



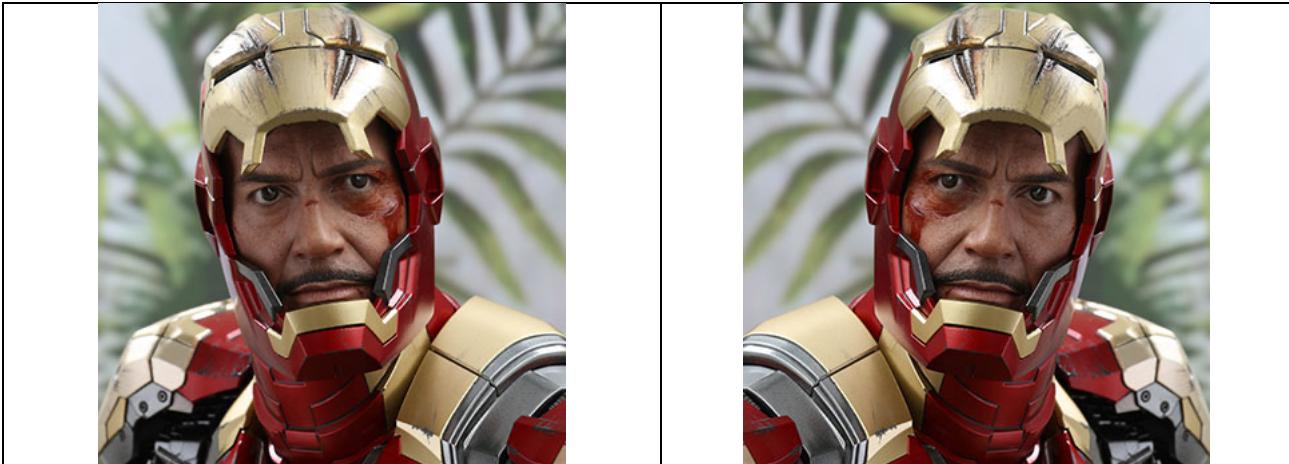
Độ sáng +100

- Input: test2.png
- Chức năng thay đổi độ tương phản cho ảnh
- Đầu ra: test2_contrast.png

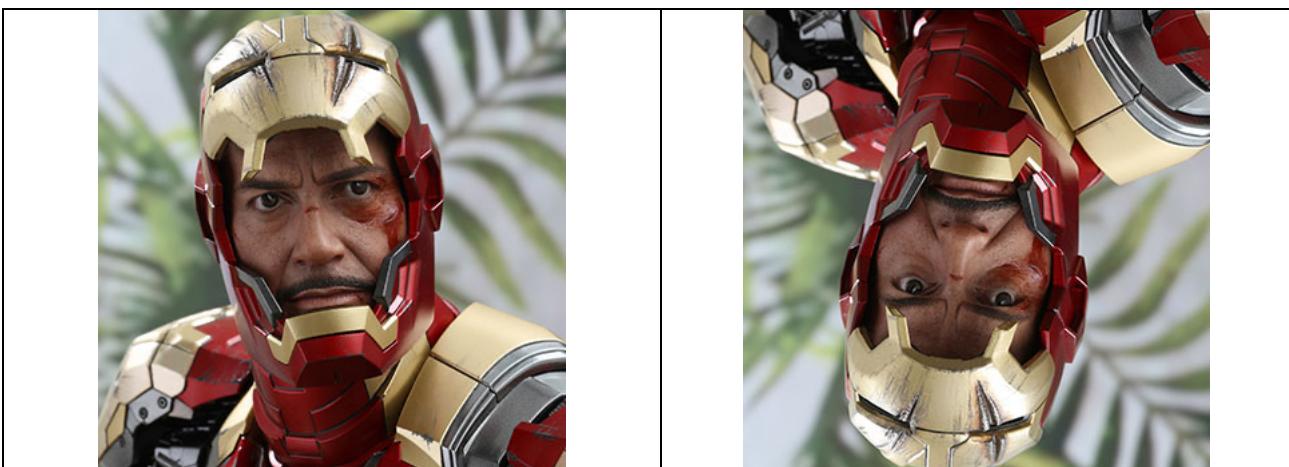


Độ tương phản +1.5

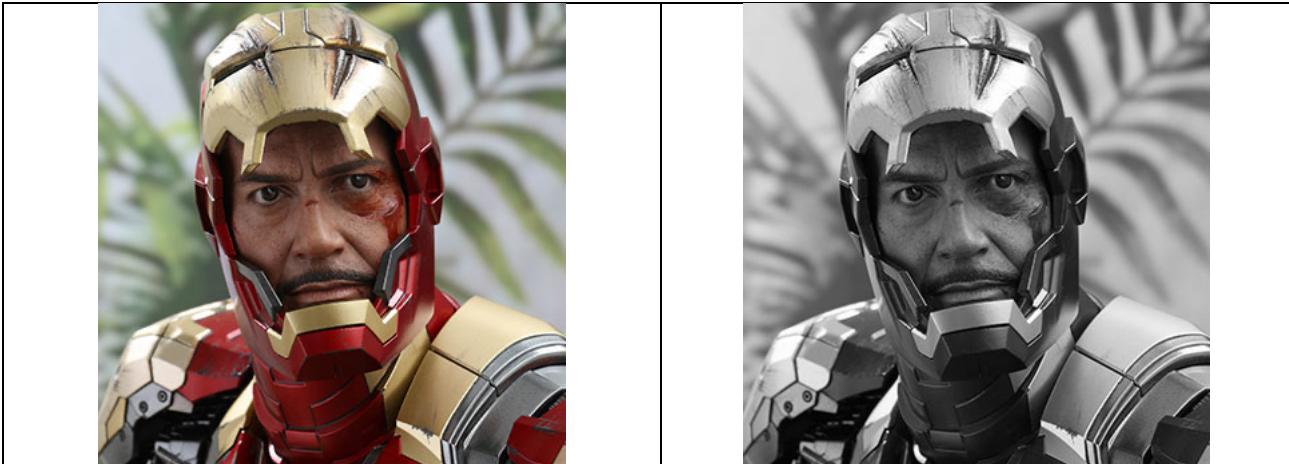
- Input: test2.png
- Chức năng: Lật ngang ảnh
- Đầu ra: test2_flip_horizontal.png



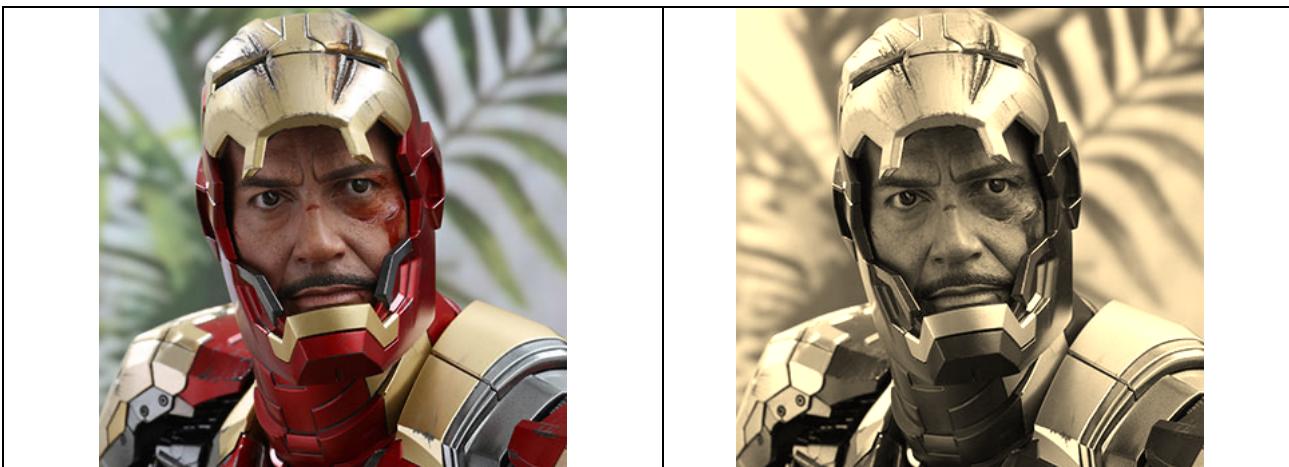
- Input: test2.png
- Chức năng: Lật ngang dọc
- Đầu ra: test2_flip_vertical.png



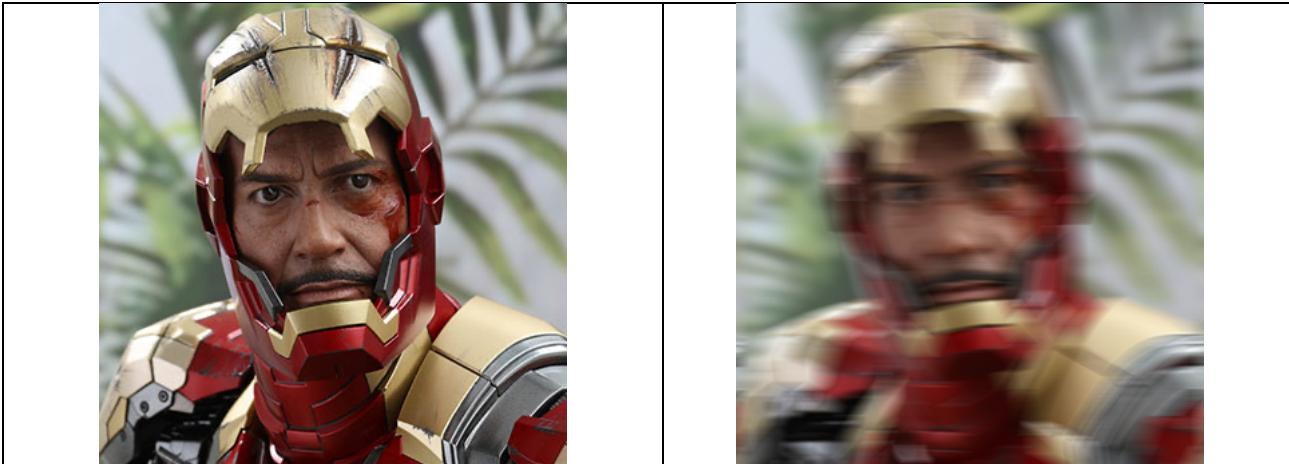
- Input: test2.png
- Chức năng: Chuyển màu ảnh RGB sang màu xám
- Đầu ra: test2_grayscale.png



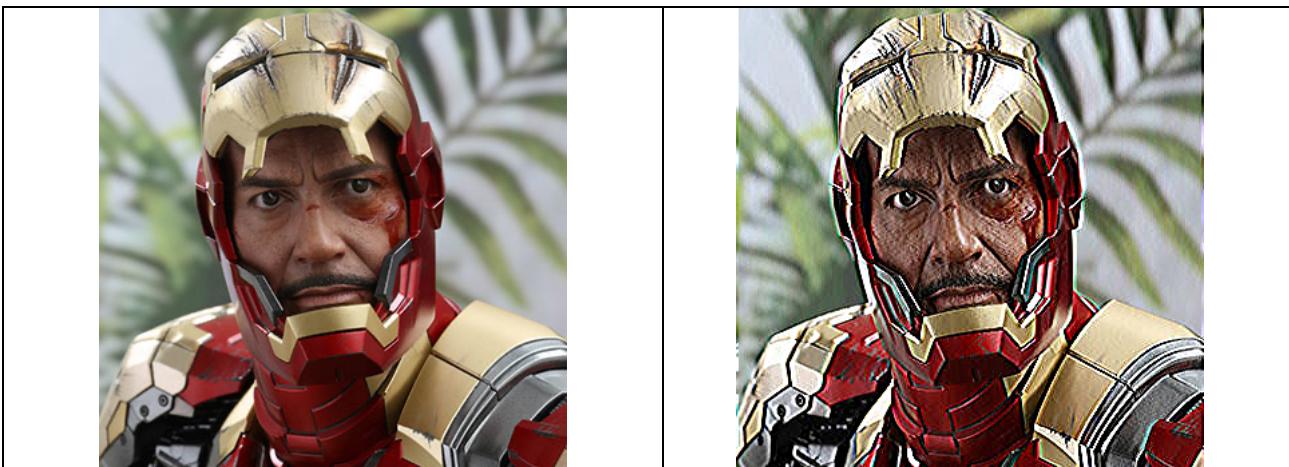
- Input: test2.png
- Chức năng: Chuyển màu ảnh RGB sang màu sepia
- Đầu ra: test2_sepia.png



- Input: test2.png
- Chức năng: Làm mờ ảnh
- Đầu ra: test2_blur.png



- Input: test2.png
- Chức năng: Làm sắc nét ảnh
- Đầu ra: test2_sharpen.png



- Input: test2.png
- Chức năng: Cắt ảnh theo kích thước (trung tâm)
- Đầu ra: test2_crop_center.png



- Input: test2.png
- Chức năng: Cắt ảnh theo khung hình tròn
- Đầu ra: test2_sepia.png



- Input: test2.png
- Chức năng: Cắt ảnh theo khung 2 elip chéo nhau
- Đầu ra: test2_crop_double_elliptical.png



- Input: test2.png
- Chức năng: Zoom in/out
- Đầu ra: test2_zoom_in.png / _zoom_out.png



IV. Nhận xét kết quả trên

- **Thay đổi độ sáng cho ảnh:** Thực hiện tốt trong thời gian cho phép.
- **Thay đổi độ tương phản cho ảnh:** Thực hiện tốt trong thời gian cho phép.
- **Lật ảnh ngang:** Thực hiện tốt trong thời gian cho phép.
- **Lật ảnh dọc:** Thực hiện tốt trong thời gian cho phép.
- **Chuyển đổi ảnh RGB thành ảnh xám:** Thực hiện tốt trong thời gian cho phép.
- **Chuyển đổi ảnh RGB thành ảnh sepia:** Thực hiện tốt trong thời gian cho phép.
- **Làm mờ ảnh:** Thực hiện tốt tuy nhiên vẫn chưa thể tối ưu được thực hiện padding cho phần viền không bị đen.
- **Làm sắc nét ảnh:** Thực hiện tốt tuy nhiên vẫn chưa thể tối ưu được thực hiện padding cho phần viền không bị trắng
- **Cắt ảnh theo kích thước:** Thực hiện tốt trong thời gian cho phép.
- **Cắt ảnh theo khung hình tròn:** Thực hiện tốt trong thời gian cho phép.
- **Cắt ảnh theo khung hình elip chéo nhau:** Thực hiện tốt trong thời gian cho phép.
- **Phóng to ảnh:** Thực hiện tốt trong thời gian cho phép với điều kiện hệ số đầu vào không quá 4
- **Thu nhỏ ảnh:** Thực hiện tốt trong thời gian cho phép.

V. Tài liệu tham khảo

REFERENCES

- [1] Từ RGB sang Đa mức xám (Grayscale) <https://tranvantri.wordpress.com/2014/09/14/tu-rbg-sang-da-muc-xam-grayscale/>
- [2] Processing an image to sepia tone in Python
<https://stackoverflow.com/questions/36434905/processing-an-image-to-sepia-tone-in-python>
- [3]Hàm Gaussian Blur https://en.wikipedia.org/wiki/Gaussian_blur
- [4]Đường tròn https://vi.wikipedia.org/wiki/%C4%90%C6%B0%C1%BB%C9Dng_tr%C3%B2n
- [5]Elip <https://vi.wikipedia.org/wiki/Elip>
- [6]Phương pháp nội suy Bilinear <https://viblo.asia/p/upscale-anh-voi-mot-mang-cnn-don-gian-Az45b04gZxY>
- [7] Kernel (image processing) [https://en.wikipedia.org/wiki/Kernel_\(image_processing\)](https://en.wikipedia.org/wiki/Kernel_(image_processing))

*****HẾT*****