

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**



**20127643 - TRƯỜNG GIA TIỄN**

**ĐỒ ÁN 2**

**| Đề tài |**

**IMAGE PROCESSING**

**| Giáo viên hướng dẫn |**

**Cô Phan Thị Phương Uyên**

**Thầy Vũ Quốc Hoàng**

**Thầy Nguyễn Văn Quang Huy**

**Môn: Toán ứng dụng cho công nghệ thông tin**

# I. MỤC LỤC

<b>I. MỤC LỤC .....</b>	<b>2</b>
<b>II. CÁC CHỨC NĂNG ĐÃ HOÀN THÀNH.....</b>	<b>3</b>
<b>III. MÔ TẢ CÁC HÀM .....</b>	<b>4</b>
1. Hàm thay đổi độ sáng .....	4
2. Hàm thay đổi độ tương phản.....	5
3. Hàm lật ảnh ngang – dọc .....	6
4. Hàm chuyển đổi ảnh RGB thành ảnh xám.....	6
5. Chồng 2 ảnh xám có cùng kích thước.....	7
6. Hàm làm mờ ảnh.....	8
7. Hàm cắt khung ảnh .....	9
a) Cắt theo khung tròn.....	9
b) Cắt theo khung ellipse chéo nhau .....	10
8. Khái quát về hàm main.....	12
<b>IV. HÌNH ẢNH KẾT QUẢ CỦA TỪNG CHỨC NĂNG .....</b>	<b>13</b>
1. Thay đổi độ sáng .....	13
2. Thay đổi độ tương phản.....	13
3. Lật ảnh.....	14
4. Chuyển RGB thành xám .....	15
5. Chồng 2 ảnh xám cùng kích cỡ.....	15
6. Làm mờ.....	16
7. Cắt khung tròn - ellipse.....	16
<b>V. TÀI LIỆU THAM KHẢO .....</b>	<b>17</b>

## II. CÁC CHỨC NĂNG ĐÃ HOÀN THÀNH

STT	TÊN CHỨC NĂNG	MỨC ĐỘ HOÀN THÀNH
1	Thay đổi độ sáng cho ảnh	100%
2	Thay đổi độ tương phản	100%
3	Lật ảnh (ngang – dọc)	100%
4	Chuyển đổi ảnh RGB thành ảnh xám	100%
5	Chồng 2 ảnh cùng kích thước (ảnh xám)	100%
6	Làm mờ ảnh	100%
7	Nâng cao – cắt ảnh theo khung hình tròn	100%
8	Nâng cao – cắt ảnh theo khung là 2 hình ellipse chéo nhau	90%

### III. MÔ TẢ CÁC HÀM

#### 1. Hàm thay đổi độ sáng

##### a) Ý tưởng

Thay đổi độ sáng về cơ bản là ta tăng - giảm giá trị RGB của mỗi pixel, nếu là tăng độ sáng thì ta tăng các giá trị RGB lên với ngưỡng cao nhất là 255 và nếu là giảm độ sáng thì giảm giá trị RGB của các pixel xuống với ngưỡng thấp nhất là 0. (1)

Với tăng - giảm độ sáng ta chỉ cần cộng ma trận hình ảnh ban đầu với ma trận chứa toàn giá trị mà ta muốn tăng giảm độ sáng. Sau đó ta xử lý các pixel có RGB vượt qua ngưỡng  $[0, 255]$ , như ta đã biết nếu RGB nếu bằng 0 thì sẽ có màu đen hoàn toàn và ngược lại nếu  $\geq 255$  thì sẽ có hiện tượng bị cháy sáng hay là quá trắng, vì vậy ta xử lý 2 trường hợp này bằng cách:

- Nếu  $> 255$  thì gán giá trị của R,G,B của pixel đó là 255
- Nếu  $< 0$  thì gán giá trị của R,G,B của pixel đó là 0

##### b) Input

- `imageArray`: ma trận ảnh đã được đổi sang `np.array`
- `percent`: Chỉ số độ sáng muốn thay đổi

##### c) Output

`newImageArray`: Ảnh đã được chỉnh lại độ sáng và được reshape lại thành 3D

##### d) Mô tả hàm

Trước tiên ta reshape lại ma trận ảnh từ 3D xuống thành 2D

Sau đó tạo một ma trận chứa có kích thước tương đương với ma trận ảnh 2D ở trên gọi là `alphaArray`, ma trận này có tất cả các phần tử đều có giá trị bằng nhau là giá trị độ sáng.

Tiếp đến ta cộng ma trận `alphaArray` này với ma trận ảnh 2D.

Sau đó đối với tất cả các phần tử/pixel trong ma trận ảnh 2D đã cộng thêm độ sáng, ta kiểm tra nếu RGB nào  $> 255$  hoặc  $< 0$  thì ta chỉnh lại

Cuối cùng reshape lại ma trận 2D thành 3D và trả về kết quả.

## 2. Hàm thay đổi độ tương phản

### a) Ý tưởng

Thay đổi độ tương phản (contrast) của một bức ảnh đơn giản là thay đổi giá trị RGB của các pixel với: Những điểm, nơi có độ sáng thấp thì ta làm cho thấp thêm còn những nơi có độ sáng cao thì ta làm cho nó sáng thêm.

Cách làm trên được thực hiện bằng cách sử dụng công thức:

$$\text{Factor} = 131 * (\text{contrast} + 127) / (127 * (131 - \text{contrast})) \quad (2)$$

Với công thức trên, factor là hệ số chỉnh độ tương phản, contrast là cấp độ tương phản mà người dùng muốn. (1)

Khi đã có hệ số chỉnh độ tương phản, đối với mọi pixel có trong ảnh, ta gán giá trị mới cho RGB của mỗi pixel theo công thức:  $R, G, B = \text{factor} * R, G, B + 127 * (1 - \text{factor})$ . (2)

Sau đó đối với từng pixel, kiểm tra giá trị RGB nếu  $> 255$  hoặc  $< 0$  thì hiệu chỉnh lại.

### b) Input

- `imageArray`: ma trận ảnh đã được đổi sang `np.array`
- `contrast`: cấp độ tương phản mà người dùng muốn

### c) Output

`newIMG`: ảnh 3D đã được điều chỉnh độ tương phản

### d) Mô tả hàm

Đầu tiên reshape lại mảng 3D thành mảng hình ảnh 2D.

Tính giá trị factor là độ hiệu chỉnh tương phản. (2)

Tạo một mảng hình ảnh mới bằng cách copy mảng cũ và gán từng phần tử của mảng mới với giá trị theo công thức đã nêu ở phần mô tả

Vì factor là kiểu float nên lúc này mảng có thể chứa toàn số float, ta chuyển về mảng về kiểu int bằng hàm `astype()`

Đối với từng phần tử của mảng, ta kiểm tra nếu  $RGB > 255$  thì cho bằng 255 và nếu  $RGB < 0$  thì cho bằng 0.

### 3. Hàm lật ảnh ngang – dọc

#### a) Ý tưởng

Lật ảnh ngang dọc về cơ bản chỉ là thay đổi vị trí của dòng, cột của ma trận ảnh.

Đối với lật ảnh ngang (horizon) thì ta đổi vị trí các cột, chuyển các cột ở vị trí đầu xuống vị trí cuối theo thứ tự và ngược lại.

Đối với lật ảnh dọc (vertical) thì ta đổi vị trí các dòng, chuyển các dòng ở vị trí đầu xuống vị trí cuối và ngược lại.

#### b) Input

- image: mảng hình ảnh 3D
- type (string): loại lật (horizon – lật ngang, vertical – lật dọc)

#### c) Output

Reversed\_arr: mảng 3D đã được lật theo trục người dùng chọn

#### d) Mô tả hàm

Ở đây ta sẽ sử dụng các hàm có sẵn trong thư viện **numpy** để viết hàm lật ảnh ngang dọc.

Đối với lật ảnh ngang (horizon) ta sử dụng **fliplr**, dịch ra theo nghĩa đen là lật trái phải. (3)

Đối với lật ảnh dọc (vertical) ta sử dụng **flipud**, dịch ra theo nghĩa đen là lật trên dưới. (4)

Các hàm trên cũng tương đương với việc sử dụng phương pháp xẻ mảng. (5)

### 4. Hàm chuyển đổi ảnh RGB thành ảnh xám

#### a) Ý tưởng

Ảnh màu xám có mỗi điểm ảnh chỉ gồm 2 giá trị khác với ảnh RGB mỗi điểm ảnh gồm 3 giá trị là RGB vì vậy việc chuyển đổi ảnh từ RGB sang xám đồng nghĩa với việc ta chuyển từ ma trận bậc 3 xuống bậc 2. (6)

Để chuyển đổi ảnh từ RGB sang xám ta sử dụng phương pháp luminosity thì theo công thức:

$$\text{New grayscale image} = (0.3 * R) + (0.59 * G) + (0.11 * B). \quad (7)$$

Nghĩa là ảnh mới sẽ bằng giá trị 3 giá trị RGB của ảnh màu cộng lại với nhau theo công thức trên. Nghĩa là ta lấy tất cả giá trị R nhân với 0.3 ta sẽ được một ma trận mới tương tự với G,B. Các ma trận này là ma trận bậc 2. Khi ta cộng các ma trận này lại, ta sẽ được ma trận ảnh xám mới. (7)

**b) Input**

imageArray: ma trận ảnh 3D

**c) Output**

greyscale: ma trận ảnh đã được chuyển thành ảnh 2D

**d) Mô tả hàm**

Đầu tiên, ta tách các giá trị r,g,b trong mảng hình RGB ra thành các ma trận 2D riêng biệt bằng phương pháp xẻ mảng. (5) (8)

Sau đó ta dùng công thức chuyển ảnh RGB thành xám theo phương pháp luminosity. Cuối cùng ta trả về mảng 2D xám cho người dùng.

## 5. Chồng 2 ảnh xám có cùng kích thước

**a) Ý tưởng**

Chồng 2 ảnh về cơ bản là cộng 2 ma trận ảnh này lại với nhau.

**b) Input**

img1: ma trận 3D ảnh thứ nhất

img2: ma trận 3D ảnh thứ hai

**c) Output**

newIMG: ảnh mới đã chồng 2 ảnh lại với nhau

**d) Mô tả hàm**

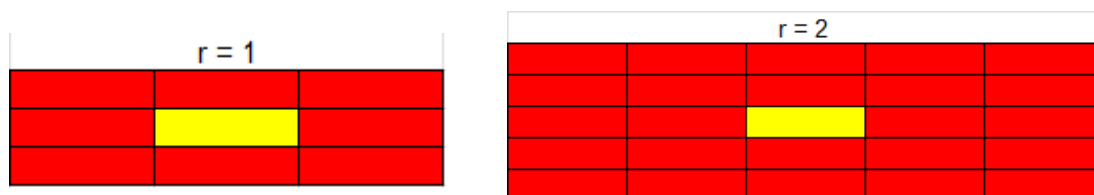
Chuyển 2 ảnh thành ảnh xám bằng hàm chuyển ảnh xám ở câu trên. Cộng 2 ma trận ảnh xám lại với nhau sau đó trả về cho người dùng ảnh 2D mới.

## 6. Hàm làm mờ ảnh

### a) Ý tưởng

Để làm mờ ảnh, ta sẽ sử dụng thuật toán box blur theo đó mỗi pixel trong hình ảnh kết quả có giá trị bằng giá trị trung bình của các pixel lân cận của nó trong hình ảnh đầu vào. Nghĩa là đối với từng pixel trong mảng hình ảnh cũ/đầu vào, ta sẽ gán nó với giá trị trung bình của toàn bộ các pixel xung quanh nó. (9)

Các pixel xung quanh pixel chính được tính theo một thông số được gọi là bán kính (radius), tức là lấy radius pixel xung quanh pixel chính. Ta có thể lấy một ví dụ:



Trên hình, màu đỏ tương đương với những pixel xung quanh và màu vàng chỉ pixel chính mà ta đang muốn gán giá trị trung bình. Như ta thấy, với  $r=1$  thì ta lấy 1 pixel tính từ pixel chính, và tương tự với  $r=2$  thì là 2 pixel. Và cũng thấy, việc lấy trung bình cũng giống như việc xét các ma trận vuông có bậc là  $(2 * r + 1)$ , các ma trận này được gọi là kernel.

Nhưng nếu xét như trên thì sẽ vướng phải trường hợp biên, đó là khi xét pixel chính thì các pixel xung quanh không thể tạo thành kernel. Vì vậy ta sẽ phải xử lý riêng trường hợp đó, có rất nhiều cách để xử lý trường hợp này, nhưng ở đây ta sẽ sử dụng phương pháp đơn giản nhất, đó là cho những pixel này thành màu đen  $\rightarrow \text{pixel} = (0,0,0)$

### b) Input

Img: mảng hình ảnh gốc

Radius: bán kính của kernel (tính từ pixel chính giữa/trung tâm,  $\text{kernel} = 2 * \text{radius} + 1$ )

### c) Output

newIMG: mảng hình ảnh 3D đã được làm mờ bằng thuật toán box blur

### d) Mô tả hàm

Tạo một mảng hình mới bằng cách copy mảng hình gốc.



Đối với từng pixel có trong mảng gốc, ta xét kiểm tra, nếu vị trí dòng / cột đó có các pixel xung quanh không thể tạo thành kernel, đó là khi  $\text{row} \text{ or } \text{col} < \text{radius} \text{ or } \text{row} + \text{radius} > \text{width}$  thì ta sẽ cho pixel chính thành màu đen. (9)

Còn nếu thỏa điều kiện có thể tạo thành kernel, ta sẽ cộng pixel đó vào trong biến tổng là sum, đồng thời tăng count lên 1.

Sau khi đã hoàn tất việc tính tổng tất cả các pixel xung quanh, ta lấy  $\text{sum} / \text{count}$  để lấy trung bình và lấy giá trị này gán vào pixel chính mà ta đang xét.

Nhưng đối với từng pixel, nếu ta xét các pixel xung quanh nó thì ta phải thêm 2 vòng lặp để xét từng dòng, từng cột trong phạm vi radius  $\rightarrow$  4 vòng for lồng nhau  $\rightarrow$  thuật toán sẽ bị chậm.

Trang amritamaz có đề xuất cách để tối ưu hóa thuật toán trên, đó là thay vì lồng 4 vòng lặp để lấy các điểm pixel xung quanh pixel chính thì ta chỉ cần xét và lấy trung bình của các pixel xung quanh theo 1 chiều là chiều ngang, sau đó làm tương tự với chiều dọc. Nghĩa là thêm một lần xét tất cả pixel có trong mảng nhưng theo chiều khác. (10)

Lúc này ta sẽ chỉ còn 3 vòng lặp lồng nhau nên độ phức tạp sẽ giảm xuống rõ rệt.

## 7. Hàm cắt khung ảnh

### Input:

- Img: mảng hình ảnh
- Type: loại khung cắt (cir - hình tròn, ell - hình ellipse)

### Output:

a: hình đã được cắt khung

#### a) Cắt theo khung tròn

##### a. Ý tưởng

Áp dụng phương trình đường tròn:

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

Với x, y là dòng/cột của pixel và a,b là dòng/cột của pixel trung tâm và R là bán kính đường tròn.

Đối với công thức trên, ta dễ dàng xác định được pixel nào không nằm trong đường tròn, đó là vé bên trái  $>$  bán kính bình phương. Và đó là những pixel mà ta sẽ tô đen

Việc quan trọng là xác định được tâm và bán kính của đường tròn, như ta thấy, đối với một bức ảnh thì pixel trung tâm chính là tâm đường tròn, vị trí của pixel này trong ma trận có thể được xác định là  $[\text{int}(\text{height}/2)][\text{int}(\text{width}/2)]$  và bán kính của đường tròn ta sẽ lấy là  $\min(\text{height}/2, \text{width}/2)$ .

Việc sau đó chỉ là xét tất cả pixel có trong ảnh, xem pixel nào không nằm trong đường tròn trên.

## b. Mô tả

Mô tả chung chưa xét type là hình tròn hay ellipse

Đầu tiên khi chưa xét type, ta tìm pixel trung tâm của hình lấy  $\text{height}/2$ ,  $\text{width}/2$  thì ra dòng và cột của pixel trung tâm.

Tiếp đến là tính độ dài đường chéo (sử dụng cho việc cắt khung ellipse), ta dùng pythagore  $\rightarrow$  đường chéo  $= \sqrt{\text{width}^2 + \text{height}^2}$

Sau đó với từng pixel, ta thế giá trị dòng, cột của pixel đó vào phương trình đường tròn

nếu  $>$  bán kính<sup>2</sup>  $\rightarrow$  nằm ngoài đường tròn  $\rightarrow$  tô đen pixel đó.

## b) Cắt theo khung ellipse chéo nhau

### a. Ý tưởng

Tương tự đối với công thức cắt khung hình tròn, ta sử dụng phương trình ellipse tổng quát với góc quay:

$$\left( \frac{\cos(\theta)(x - x_c) + \sin(\theta)(y - y_c)}{a} \right)^2 + \left( \frac{\cos(\theta)(y - y_c) - \sin(\theta)(x - x_c)}{b} \right)^2 = 1$$

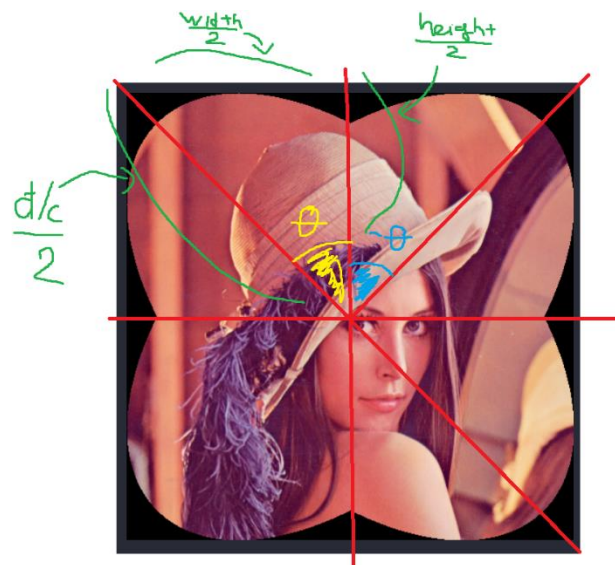
Với:

- a, b: trục chính trục phụ
- $x_c, y_c$ : tọa độ tâm (dòng/cột pixel trung tâm)
- $\cos, \sin$  của góc quay

Do khung của ta là 2 hình ellipse chồng lên nhau nên đối với mỗi pixel, ta sẽ xét một lúc cả 2 hình, nếu pixel không nằm trong cả 2 hình thì ta sẽ bôi đen nó. Điều kiện để một tọa độ (pixel) không nằm trong ellipse là vế phải của phương trình trên  $> 1$ .

Việc xác định tâm của mảng hình ta đã nói ở trên, với trục chính và phụ của hình ellipse, em không tìm được cách nào chính xác để xác định đúng nhất vì thế nên sẽ dựa trên đường chéo (đã nói cách tìm ở trên) để ước lượng khoảng chừng.

Còn việc xác định các góc quay thì ta sử dụng lượng giác trong tam giác vuông từ đó xác định được cos, sin chính xác của góc quay sang phải và sau đó thì dựa trên công thức lượng giá  $\cos(x) = \cos(-x)$  và  $\sin(-x) = -\sin(x)$  để ta có được sin, cos của góc còn lại.



Sau đó ta xét tất cả các pixel có trong mảng ảnh, nếu pixel nào không nằm trong 2 hình ellipse trên thì ta gán cho nó bằng (0,0,0).

### b. Mô tả hàm

Ta sẽ hardcode cho trục chính và trục phụ của ellipse như sau:

- Trục chính(major axis) =  $\text{đ/c}(\text{diagonal}) / 2.5$
- Trục phụ (minor axis) =  $\text{majorAxis} / 2$

Sau đó dùng công thức lượng giác trong tam giác để tính cos, sin của góc quay.

Đối với mỗi pixel, ta sẽ xét nó có nằm trong cả 2 hình ellipse đã được xoay hay không, nếu không thì gán cho giá trị rgb của pixel đó là (0,0,0).

## 8. Khái quát về hàm main









Hàm main là một menu, đầu tiên người dùng nhập tên ảnh đang muốn xử lý.

Sau đó nhập vào choice, là lựa chọn chức năng mà người dùng muốn sử dụng. choice bao gồm:


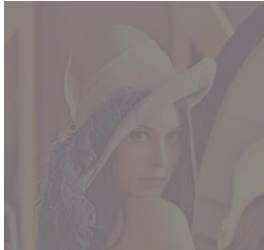


0. Thực hiện hết tất cả các chức năng có sẵn và lưu ảnh theo tên, chức năng này còn bao gồm việc chọn:
  - a. Chọn “0” là sử dụng các tham số default đã được nhập sẵn cho toàn bộ tất cả các chức năng.
  - b. Chọn “1” là người dùng phải tự nhập tất cả mọi thông số cho tất cả các hàm chức năng (ví dụ: đối với thay đổi độ sáng phải nhập thông số độ sáng, ..)
1. Thay đổi độ sáng, có nhập thông số độ sáng muốn thay đổi
2. Thay đổi độ tương phản, có nhập thông số độ tương phản muốn thay đổi
3. Lật ảnh ngang dọc, nhập chuỗi “horizon” để lật ngang, “vertical” để lật dọc
4. Chuyển ảnh RGB thành ảnh xám
5. Chồng 2 ảnh màu xám cùng kích cỡ, nhập tên ảnh thứ 2 để chồng
6. Làm mờ ảnh, nhập thông số bán kính (radius với  $\text{kernel} = 2 * \text{radius} + 1$ )
7. Đóng khung cho ảnh, nhập loại khung (cir – hình tròn, ell – hình ellipse chồng)

## IV. HÌNH ẢNH KẾT QUẢ CỦA TỪNG CHỨC NĂNG

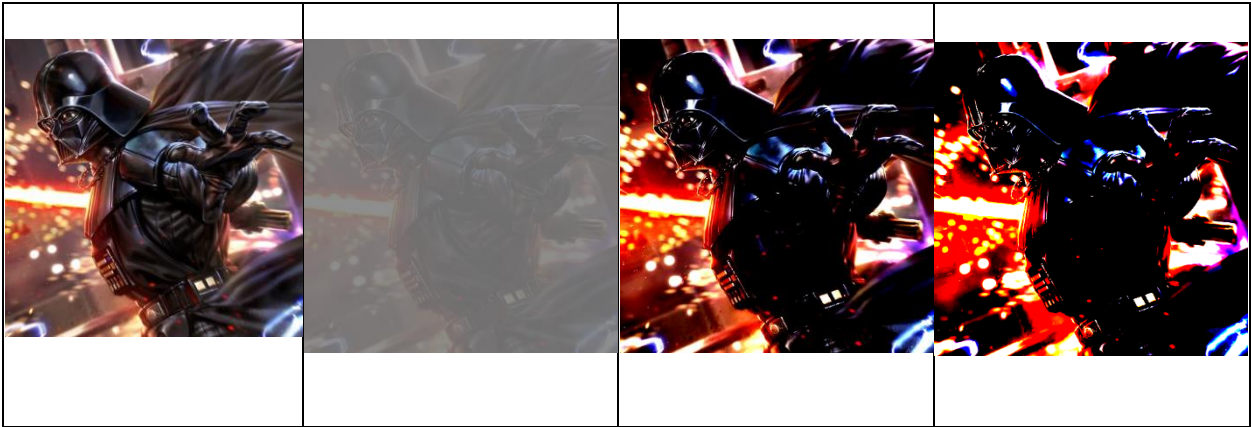
### 1. Thay đổi độ sáng

ẢNH GỐC	-100	50	100
			
			


### 2. Thay đổi độ tương phản

Ảnh gốc	-100	50	100
			











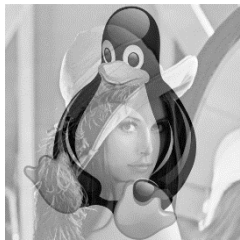

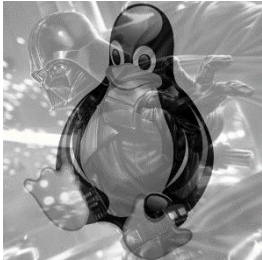
3. Lật ảnh

ẢNH GỐC	LẬT NGANG	LẬT DỌC
		
		

#### 4. Chuyển RGB thành xám

ẢNH GỐC	ẢNH ĐÃ CHUYỂN THÀNH XÁM
	
	







#### 5. Chồng 2 ảnh xám cùng kích cỡ

ẢNH GỐC 1	ẢNH GỐC 2	ẢNH ĐÃ CHỒNG
		
		

## 6. Làm mờ

ẢNH GỐC	$r = 1 (3 \times 3)$	$r = 5 (11 \times 11)$	$r = 20 (41 \times 41)$
	 time: 5.4s	 time: 12.1s	 time: 33.2s
	 time: 5.5s	 time: 11.9s	 time: 31.3s

## 7. Cắt khung tròn - ellipse

ẢNH GỐC	KHUNG TRÒN	KHUNG ELLIPSE
		
		



## V. TÀI LIỆU THAM KHẢO

1. [Online] <https://ie.nitk.ac.in/blog/2020/01/19/algorithms-for-adjusting-brightness-and-contrast-of-an-image/#:~:text=To%20change%20the%20brightness%20of,every%20pixel%20of%20the%20image..>
2. Stackoverflow. *how-do-i-increase-the-contrast-of-an-image-in-python-opencv*. [Online] <https://stackoverflow.com/questions/39308030/how-do-i-increase-the-contrast-of-an-image-in-python-opencv>.
3. numpy. [Online] <https://numpy.org/doc/stable/reference/generated/numpy.flipplr.html>.
4. numpy. [Online] <https://numpy.org/doc/stable/reference/generated/numpy.flipud.html#numpy.flipud>.
5. stackoverflow. [Online] <https://stackoverflow.com/questions/41309201/efficient-transformations-of-3d-numpy-arrays>.
6. wikipedia. [Online] <https://en.wikipedia.org/wiki/Grayscale>.
7. tutorialspoint. [Online] [https://www.tutorialspoint.com/dip/grayscale\\_to\\_rgb\\_conversion.htm](https://www.tutorialspoint.com/dip/grayscale_to_rgb_conversion.htm).
8. stackoverflow. *How can I convert an RGB image into grayscale in Python?* [Online] <https://stackoverflow.com/questions/12201577/how-can-i-convert-an-rgb-image-into-grayscale-in-python>.
9. wikipedia. [Online] [https://en.wikipedia.org/wiki/Box\\_blur](https://en.wikipedia.org/wiki/Box_blur).
10. mazumdar, amrita. amritamaz. [Online] <http://amritamaz.net/blog/understanding-box-blur>.
11. [Online] <https://thptchuyenlamson.vn/phuong-trinh-duong-tron/>.