










MỤC LỤC

I. Các chức năng đã hoàn thành:	3
II. Ý tưởng thực hiện và mô tả các hàm:	4
1. Thay đổi độ sáng cho ảnh:	4
2. Thay đổi độ tương phản cho ảnh:	5
3. Lật ảnh:	5
a) Lật ảnh ngang:	5
b) Lật ảnh dọc:	5
4. Chuyển đổi màu ảnh RGB:	5
a) Chuyển đổi ảnh RGB thành ảnh xám:	5
b) Chuyển đổi RGB thành ảnh sepia:	6
5. Làm mờ/sắc nét ảnh:	6
a) Làm mờ ảnh:	6
b) Làm sắc nét ảnh:	8
6. Cắt ảnh theo kích thước:	9
7. Cắt ảnh theo khung hình tròn:	10
8. Cắt ảnh theo khung elip chéo nhau:	11
9. Hàm main:	14
III. Hình ảnh kết quả với từng chức năng:	14
1. Hình 1: Kích thước 1000x1000 px	14
2. Hình 2: Kích thước 512x512 px	19
IV. Nhận xét các kết quả trên:	25
1. Nhận xét hình 1:	25
2. Nhận xét hình 2:	25
V. Tài liệu tham khảo:	26

I. Các chức năng đã hoàn thành:

0	Ảnh ban đầu		
STT	Chức năng	Ghi chú	Mức độ hoàn thành
1	Thay đổi độ sáng cho ảnh	Độ sáng +50	
2	Thay đổi độ tương phản cho ảnh	Độ tương phản +1.2	
3a	Lật ảnh ngang		
3b	Lật ảnh dọc		
4a	Chuyển đổi ảnh RGB thành ảnh xám		
4b	Chuyển đổi ảnh RGB thành ảnh sepia		

5a	Làm mờ ảnh	Sigma = 100, Kernel size = 21	
5b	Làm sắc nét ảnh	Sigma = 100, Kernel size = 21	
6	Cắt ảnh theo kích thước	Kích thước 50%	
7	Cắt ảnh theo khung hình tròn		
8	Cắt ảnh theo khung elip chéo nhau		

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 trong ma trận, có thể là âm hoặc dương, 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 số biểu thị màu đen là 0.
- *Mô tả hàm chức năng*: **exposure(image, num)**
 - + Hàm exposure với đầu vào *image* (ma trận vector màu của hình ảnh), *num* (giá trị để thay đổi độ sáng của ảnh).
 - + Hàm sẽ tạo ra một ma trận bằng với kích thước ma trận *image* (*arrExposure*), sau đó sẽ cộng ma trận *arrExposure* và *image* để thay đổi độ sáng theo mong muốn. Sử dụng hàm *np.clip* để ràng buộc giá trị trong khoảng [0, 255].

+ Đầu ra trả về ảnh sau khi thực hiện chức năng, dùng hàm *pil.Image.fromarray* để tạo ảnh mới từ ma trận đã tính.

2. Thay đổi độ tương phản cho ảnh:

- *Ý tưởng*: nhân thêm giá trị cho các phần tử trong ảnh, là số thập phân dương, nếu số lớn hơn 1 thì sẽ tăng độ tương phản vì các pixel màu sẽ có khoảng cách lớn hơn, nếu số bé hơn 1 thì sẽ giảm độ tương phản vì các pixel màu sẽ có khoảng cách nhỏ hơn.

- *Mô tả hàm chức năng*: **contrast(image, num)**

+ Hàm contrast với đầu vào *image* (ma trận vector màu của hình ảnh), *num* (giá trị để thay đổi độ tương phản của ảnh).

+ Hàm sẽ ép kiểu ma trận *image* về kiểu float để hỗ trợ nhân kiểu float về sau, sau đó sẽ nhân ma trận *image* với *num* để thay đổi độ tương phản theo mong muốn. Sử dụng hàm *np.clip* để ràng buộc giá trị trong khoảng [0, 255].

+ Đầu ra trả về ảnh sau khi thực hiện chức năng, dùng hàm *pil.Image.fromarray* để tạo ảnh mới từ ma trận đã tính.

3. Lật ảnh:

a) *Lật ảnh ngang*:

- *Ý tưởng*: thay đổi lật các giá trị ma trận theo chiều ngang.

- *Mô tả hàm chức năng*: **flip(image, type)**

+ Hàm flip với đầu vào *image* (ma trận vector màu của hình ảnh), *type* (kiểu lật ảnh: "0": lật ngang, "1": lật dọc).

+ Sử dụng hàm *np.flip* của thư viện numpy để lật ma trận theo chiều ngang (axis = 0)

b) *Lật ảnh dọc*:

- *Ý tưởng*: thay đổi lật các giá trị ma trận theo chiều dọc.

- *Mô tả hàm chức năng*: **flip(image, type)**

+ Hàm flip với đầu vào *image* (ma trận vector màu của hình ảnh), *type* (kiểu lật ảnh: "0": lật ngang, "1": lật dọc).

+ Sử dụng hàm *np.flip* của thư viện numpy để lật ma trận theo chiều dọc (axis = 1).

+ Đầu ra trả về ảnh sau khi thực hiện chức năng, dùng hàm *pil.Image.fromarray* để tạo ảnh mới từ ma trận đã tính.

4. Chuyển đổi màu ảnh RGB:

a) *Chuyển đổi ảnh RGB thành ảnh xám*:

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

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

- + Hàm `toGray` với đầu vào *image* (ma trận vector màu của hình ảnh).
- + Công thức chuyển ảnh từ màu RGB sang màu xám [3] là:

$$\text{Pixel} = 0.3 * R + 0.59 * G + 0.11 * B$$
- + Sử dụng hàm *np.dot* của thư viện *numpy* để tính tích vô hướng của vector (*image[:, :, :3]*) tại điểm ảnh với vector [0.3, 0.59, 0.11] tượng trưng cho công thức trên.
- + Đầu ra trả về ảnh sau khi thực hiện chức năng, dùng hàm *pil.Image.fromarray* để tạo ảnh mới từ ma trận đã tính.

b) Chuyển đổi RGB thành ảnh sepia:

- Ý tưởng: thay đổi các giá trị màu vector theo công thức màu chuyển thành ảnh sepia.
- Mô tả hàm chức năng: **sepia(image)**
 - + Hàm *sepia* với đầu vào *image* (ma trận vector màu của hình ảnh).
 - + Công thức chuyển ảnh từ màu RGB sang màu sepia [4] là:

$$\begin{aligned} \text{Pixel_R} &= 0.393 * R + 0.769 * G + 0.189 * B \\ \text{Pixel_G} &= 0.349 * R + 0.686 * G + 0.168 * B \\ \text{Pixel_B} &= 0.272 * R + 0.534 * G + 0.131 * B \end{aligned}$$
 - + Lấy các giá trị màu của các vector và gán cho giá trị màu mới tương ứng với công thức trên, sử dụng hàm *np.dot* để tính tích vô hướng của điểm ảnh với các vector tượng trưng cho công thức trên, dùng *np.clip* để ràng buộc giá trị trong khoảng [0, 255].
 - + Đầu ra trả về ảnh sau khi thực hiện chức năng, dùng hàm *pil.Image.fromarray* để tạo ảnh mới từ ma trận đã tính.

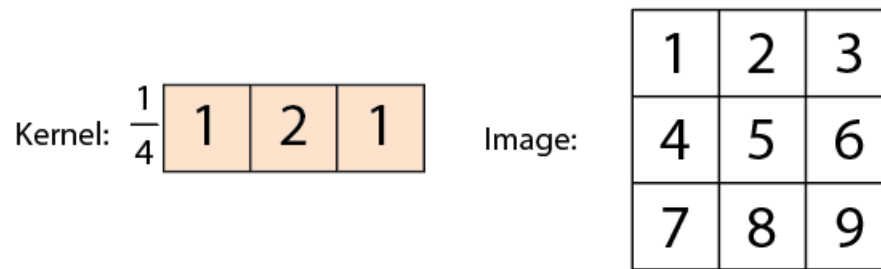
5. Làm mờ/sắc nét ảnh:

a) Làm mờ ảnh:

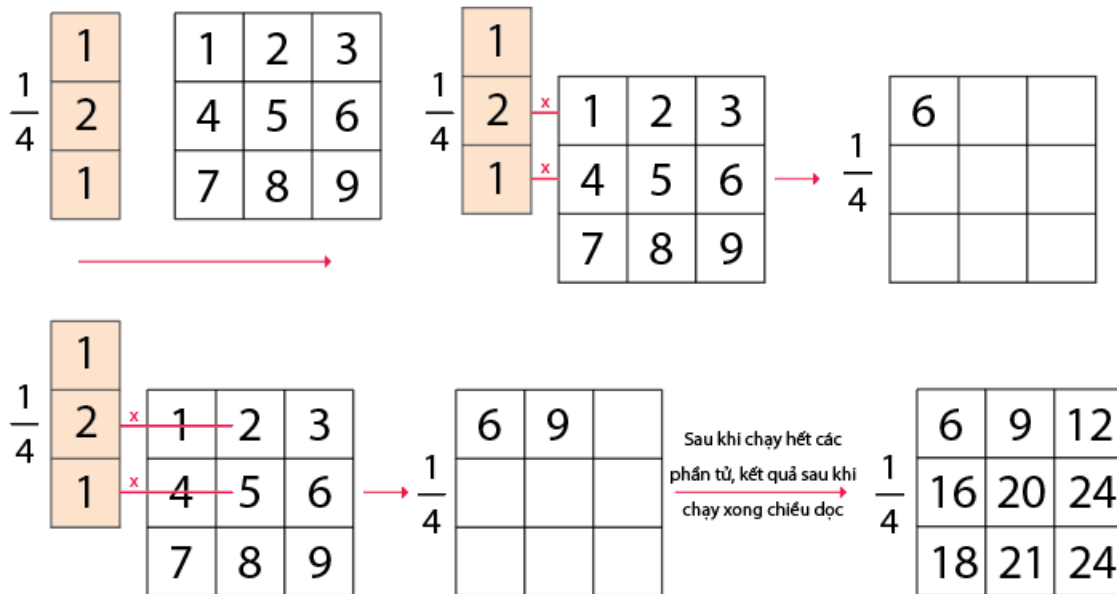
- Ý tưởng: trung hòa màu giữa pixel với các pixel xung quanh.
- + Công thức Gaussian Blur [2]:

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

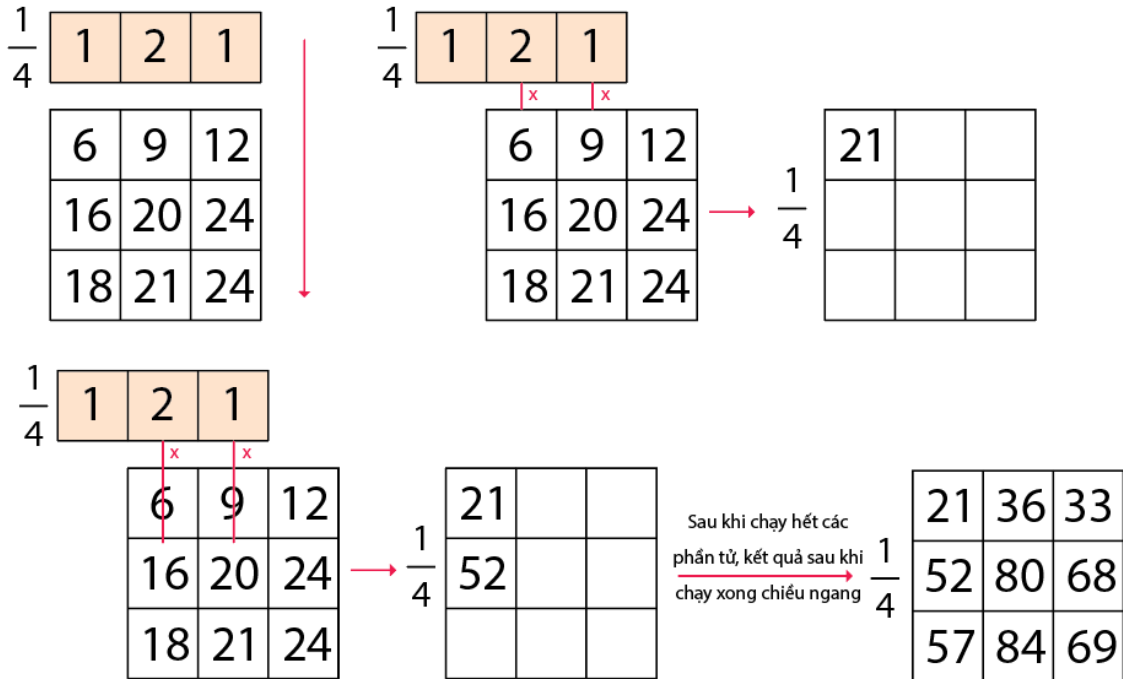
- + Kernel Gaussian Blur đảm bảo tổng các phần tử là bằng 1
- + Áp dụng convolution (tích chập) giữa ma trận ảnh và ma trận Gaussian kernel. Bước này tính toán giá trị mới cho mỗi pixel trong ảnh, dựa trên trung bình có trọng số của các pixel xung quanh (áp bộ lọc), trong bài ý tưởng là dành cho ma trận 1D [7]



Hình 5a.1: Ví dụ cho kernel 1D và hình ảnh



Hình 5a.2: Tính phép tích chập với chiều dọc (số trong hình vẽ là ví dụ minh họa, trên thực tế, $(1/4)$ sẽ được nhân vào các phần tử tạo thành ma trận ảnh mới)



Hình 5a.3: Tính phép tích chập với chiều ngang (tính tiếp với kết quả của chiều dọc, kết quả là ma trận ảnh sau khi được blur)

- Mô tả hàm chức năng: **gaussianKernel(sigma, size), blur(image, sigma, size)**
 - + Hàm gaussianKernel với đầu vào *sigma* (độ rộng của bộ lọc), *size* (kích thước kernel), dùng để tính Gaussian Kernel 1D theo công thức đã nêu ở trên, *np.linspace* để tạo mảng đối xứng.
 - + Hàm blur với đầu vào *image* (ma trận vector màu của hình ảnh), *sigma* (độ rộng của bộ lọc), *size* (kích thước kernel).
 - + Đầu tiên sẽ tạo Gaussian Kernel 1D, sau đó, dùng hàm *np.apply_along_axis* để chạy hàm tính tích chập *np.convolve* trên chiều dọc của ảnh, sau đó sẽ tính tích chập theo chiều ngang của kết quả ảnh vừa tính.
 - + Đầu ra trả về ảnh sau khi thực hiện chức năng, dùng hàm *pil.Image.fromarray* để tạo ảnh mới từ ma trận đã tính.

b) Làm sắc nét ảnh:

- Ý tưởng: làm tăng độ tách biệt màu giữa pixel với các pixel xung quanh.
 - + Kernel Sharpen được tạo ra từ Kernel Gaussian Blur, đảm bảo tổng các phần tử cũng bằng 1, với công thức [6]

$$2 * \text{identity} - \text{gaussianKernel} = \text{sharpenKernel}$$

Với: identity là ma trận toàn 0, giữa là số 1 có size bằng Gaussian Kernel

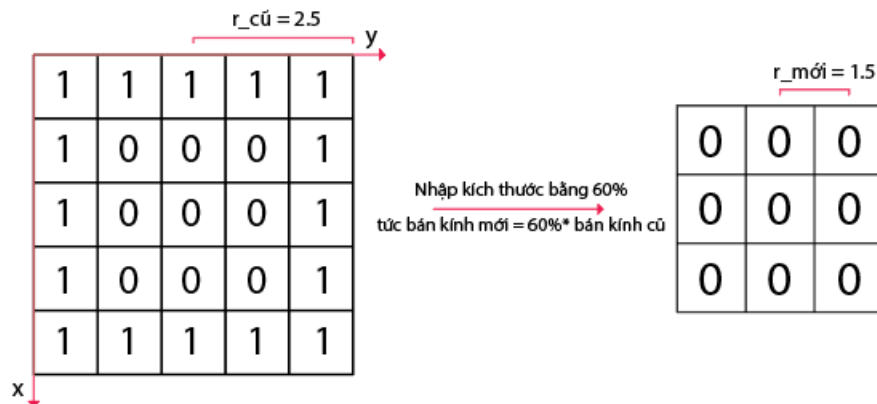
$$\text{VD: } 2 * \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix} - \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} -1 & 6 & -1 \\ 6 & 12 & -6 \\ -1 & 6 & -1 \end{bmatrix}$$

Hình 5b.1: Công thức tính Sharpen Kernel và ví dụ

- + Áp dụng convolution (tích chập) giữa ma trận ảnh và ma trận Sharpen kernel. Bước này tính toán giá trị mới cho mỗi pixel trong ảnh, dựa trên trung bình có trọng số của các pixel xung quanh (áp bộ lọc), trong bài ý tưởng là dành cho ma trận 1D, tương tự như đã trình bày cho quá trình Blur ở trên (hình 5a.1 đến 5a.3)
- Mô tả hàm chức năng: **sharpen(image, sigma, size)**
 - + Hàm sharpen với đầu vào *image* (ma trận vector màu của hình ảnh), *sigma* (độ rộng của bộ lọc), *size* (kích thước kernel).
 - + Đầu tiên sẽ tạo Gaussian Kernel 1D, sau đó, dùng hàm *np.zeros* để tạo ma trận 0 cùng kích thước Gaussian Kernel 1D (*identity*), đổi center của *identity* thành 1, tính toán Sharpen Kernel như công thức trình bày ở hình 5b.1, dùng hàm *np.apply_along_axis* để chạy hàm tính tích chập *np.convolve* trên chiều dọc của ảnh, sau đó sẽ tính tích chập theo chiều ngang của kết quả ảnh vừa tính.
 - + Đầu ra trả về ảnh sau khi thực hiện chức năng, dùng hàm *plt.imshow* để tạo ảnh mới từ ma trận đã tính.

6. Cắt ảnh theo kích thước:

- Ý tưởng: tách ma trận ở trung tâm.



Hình 6.1: Cắt ma trận theo kích thước

- Mô tả hàm chức năng: **squareCrop(image, r)**

+ Hàm squareCrop với đầu vào *image* (ma trận vector màu của hình ảnh), *r* (kích thước bán kính mới so với bán kính cũ, vd: nhập 50 tức là bán kính mới bằng 50% bán kính cũ).

+ Hàm sẽ tính bán kính mới dựa theo đầu vào, sau đó tạo ma trận mới, quét từ center của ma trận cũ [center_y-r mới:center_y+r mới] theo chiều dọc và [center_x-r mới, center_x+r mới] theo chiều ngang.

+ Đầu ra trả về ảnh sau khi thực hiện chức năng, dùng hàm *pil.Image.fromarray* để tạo ảnh mới từ ma trận đã tính.

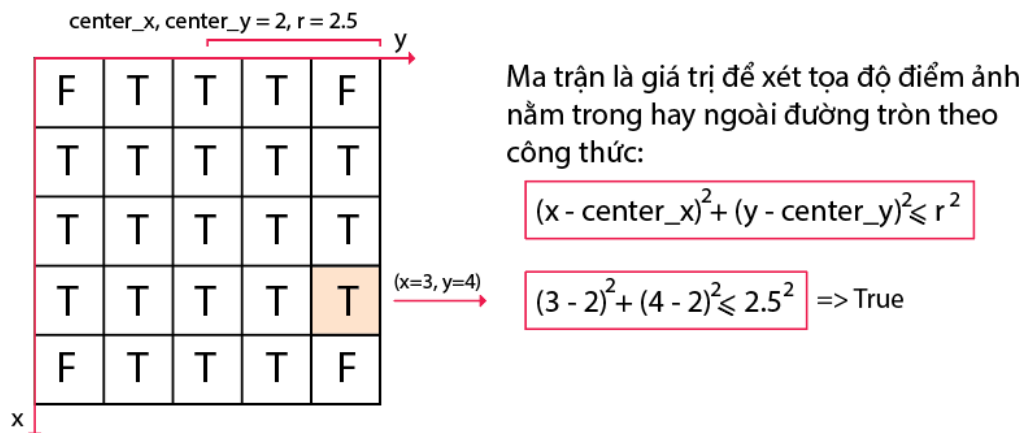
7. Cắt ảnh theo khung hình tròn:

- Ý tưởng: tạo ra ma trận trùng kích thước với ảnh chứa giá trị True, False tượng trưng cho trong và ngoài đường tròn rồi chuyển các pixel màu có giá trị False thành màu đen bằng cách dùng phép nhân element-wise multiplication giữa ma trận điểm ảnh và ma trận mask.

+ Công thức đường tròn [8]

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

Với (a, b) là tọa độ tâm, (x, y) là tọa độ các điểm trên đường tròn, r là bán kính.



Hình 7.1: Tạo ra ma trận trùng kích thước với ảnh chứa giá trị True, False tượng trưng cho trong và ngoài đường tròn (ma trận mask)

1	1	1	1	1		F	T	T	T	F		0	1	1	1	0
1	1	1	1	1		T	T	T	T	T		1	1	1	1	1
1	1	1	1	1	o	T	T	T	T	T	=	1	1	1	1	1
1	1	1	1	1		T	T	T	T	T		1	1	1	1	1
1	1	1	1	1		F	T	T	T	F		0	1	1	1	0

Hình 7.2: Phép *element – wise multiplication* giữa ma trận điểm ảnh và ma trận mask, tạo ma trận mới là ma trận ảnh đã cắt theo khung hình tròn

- Mô tả hàm chức năng: **circleCrop(image)**
 - + Hàm circleCrop với đầu vào *image* (ma trận vector màu của hình ảnh).
 - + Dùng hàm *np.mgrid* để tạo 2 ma trận chứa tọa độ x, y của điểm ảnh.
 - + Tạo ma trận mask theo công thức nêu trên (hình 7.1).
 - + Dùng hàm *np.multiply (*)* để tính phép nhân *element – wise* giữa ma trận điểm ảnh và ma trận mask để tạo ma trận mới là ma trận đã cắt theo khung hình tròn.
 - + Đầu ra trả về ảnh sau khi thực hiện chức năng, dùng hàm *pil.Image.fromarray* để tạo ảnh mới từ ma trận đã tính.

8. Cắt ảnh theo khung elip chéo nhau:

- Ý tưởng: tạo ra ma trận trùng kích thước với ảnh chứa giá trị True, False tương trưng cho trong và ngoài 2 hình elip chéo nhau rồi chuyển các pixel màu có giá trị False thành màu đen bằng cách dùng phép nhân *element-wise multiplication* giữa ma trận điểm ảnh và ma trận mask.
- + Định nghĩa elip

Cho 2 điểm F1 và F2 luôn cố định, với $F1 F2 = 2c$ ($c > 0$) và hằng số $a > c$, ta sẽ có

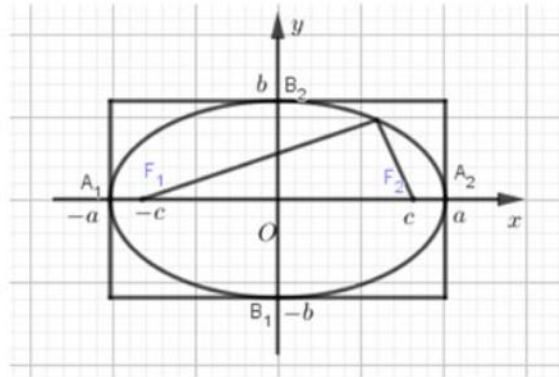
- tập hợp các điểm M thỏa mãn với $MF1 + MF2 = 2a$
- MF1, MF2 được gọi là bán kính qua tiêu
- Khoảng cách $F1 F2 = 2c$ là tiêu cự của (E)
- Các điểm F1, F2 là tiêu điểm của (E)

Hình 8.1: Định nghĩa elip [9]

Với $F_1(-c; 0)$, $F_2(c; 0)$:

$$M(x; y) \in (E) \Leftrightarrow \frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \quad (1) \text{ trong đó } b^2 = a^2 - c^2$$

(1) được gọi là phương trình chính tắc của (E)



Hình 8.2: Công thức elip [9]

+ Công thức đường elip [9]

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1.$$

Với (a, b) là bán kính đường chéo lớn, bán kính đường chéo nhỏ, (x, y) là tọa độ các điểm trên đường elip.

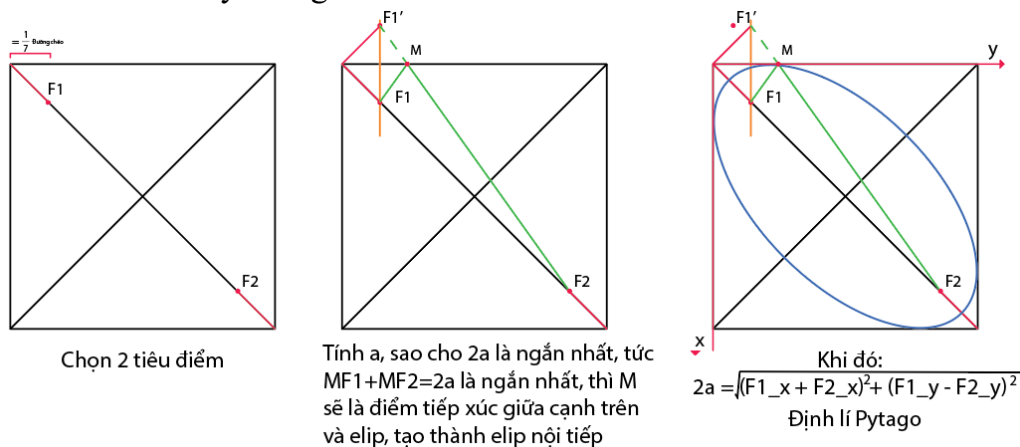
+ Công thức xoay trong hệ trục tọa độ [10]

$$x' = x \cos \alpha - y \sin \alpha$$

$$y' = x \sin \alpha + y \cos \alpha$$

Áp dụng để xoay elip sao cho 2 tiêu điểm nằm trên đường chéo của hình ảnh (góc quay bằng 45 độ)

+ Các bước ý tưởng như hình dưới:



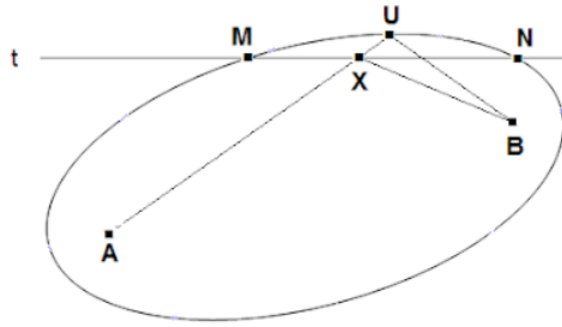
Hình 8.3: Ý tưởng thực hiện elip chéo nội tiếp

+ Chứng minh vì sao $MF_1 + MF_2$ ngắn nhất lại tạo thành elip nội tiếp:

Từ đây chúng ta kết luận rằng hình elip sẽ phải tiếp xúc với đường thẳng t tại điểm M . Vì sao vậy? Đó là bởi vì nếu hình elip mà không tiếp xúc với đường thẳng t thì đường elip sẽ cắt đường thẳng t tại hai điểm M và N . Chúng ta chỉ cần lấy một điểm X nằm bất kỳ giữa M và N như hình dưới đây thì chúng ta sẽ có

$$XA + XB < XA + XU + UB = AU + UB = \ell = MA + MB.$$

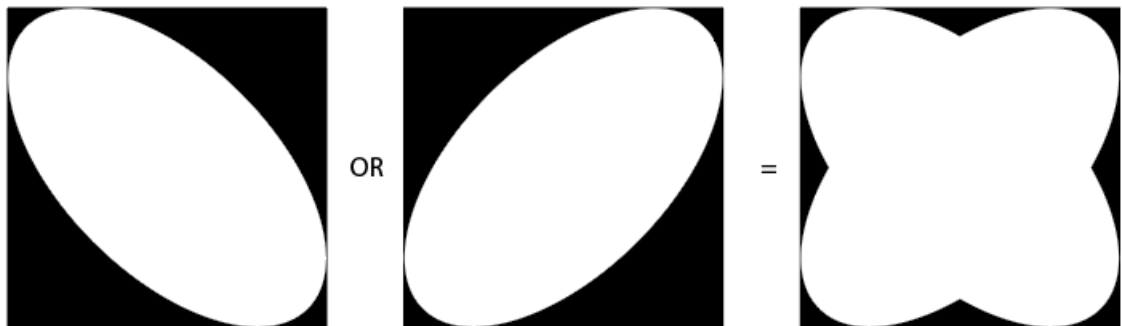
Như vậy $XA + XB < MA + MB$, mâu thuẫn với giả thiết rằng $MA + MB$ là nhỏ nhất.



nếu hình elip không tiếp xúc với t thì chúng ta sẽ tìm ra được một điểm X tốt hơn là điểm M

Hình 8.4: Chứng minh cho ý tưởng [5]

+ Sau khi tính được a, b để vẽ ra được elip, ta thực hiện phép xoay như công thức xoay trong hệ trục tọa độ đã nêu trên ta được ma trận mask left chứa các giá trị elip như hình 8.3, thực hiện phép lật ma trận maskElipLeft để tạo elip đối xứng (maskElipRight) sau đó dùng toán tử OR để tạo mask hình 2 elip chéo nhau.



Hình 8.5: Merge 2 elip bằng toán tử OR (vùng trắng biểu thị giá trị TRUE, đen là FALSE)

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

- + Hàm `elipCrop` với đầu vào *image* (ma trận vector màu của hình ảnh).
- + Dùng hàm `np.mgrid` để tạo 2 ma trận chứa tọa độ x, y của điểm ảnh.
- + Tạo ma trận `maskElipLeft` theo công thức đường elip và xoay trong hệ trục tọa độ
- + Dùng `np.flip` flip ma trận `maskElipLeft` thành `maskElipRight`.
- + Dùng `np.logical_or` để merge 2 elip (hình 8.5) tạo ma trận mask

+ Dùng hàm *np.multiply (*)* để tính phép nhân element – wise giữa ma trận điểm ảnh và ma trận mask để tạo ma trận mới là ma trận đã cắt theo khung hình tròn.

+ Đầu ra trả về ảnh sau khi thực hiện chức năng, dùng hàm *pil.Image.fromarray* để tạo ảnh mới từ ma trận đã tính.

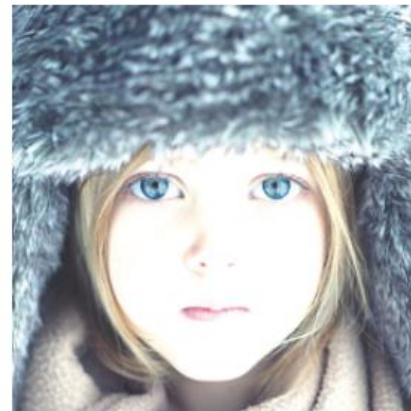
9. Hàm main:

- Chương trình sẽ cho chọn các chức năng theo yêu cầu từ 1-8 tương ứng (được nêu rõ khi chạy hàm main) và các giá trị đầu vào do người dùng nhập, đối với mục 0, chương trình sẽ thực hiện toàn bộ chức năng, 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), sau đó sẽ in ra màn hình thời gian chạy của chức năng tương ứng.
- Đối với chức năng số 5 (làm mờ/sắc nét), khi yêu cầu nhập *sigm*, *size*, người dùng cần nhập theo cú pháp: *sigma [khoảng trắng] size*.

III. Hình ảnh kết quả với từng chức năng:

1. Hình 1: Kích thước 1000x1000 px

- Đầu vào: *chandung.jpg*
- Chức năng: Thay đổi độ sáng cho ảnh.
- Đầu ra: *chandung_exposure.jpg*
- Thời gian: 0.048s



Độ sáng +50

- Đầu vào: *chandung.jpg*
- Chức năng: Lật ảnh ngang
- Đầu ra: *chandung_contrast.jpg*
- Thời gian: 0.05s



Độ tương phản +1.2

- Đầu vào: chandung.jpg
- Chức năng: Lật ảnh ngang
- Đầu ra: chandung_horizon.jpg
- Thời gian: 0.019s



Lật ảnh ngang

- Đầu vào: chandung.jpg
- Chức năng: Lật ảnh dọc
- Đầu ra: chandung_vertical.jpg
- Thời gian: 0.009s



Lật ảnh dọc

- Đầu vào: chandung.jpg
- Chức năng: Chuyển đổi ảnh RGB thành ảnh xám
- Đầu ra: chandung_gray.jpg
- Thời gian: 0.046s



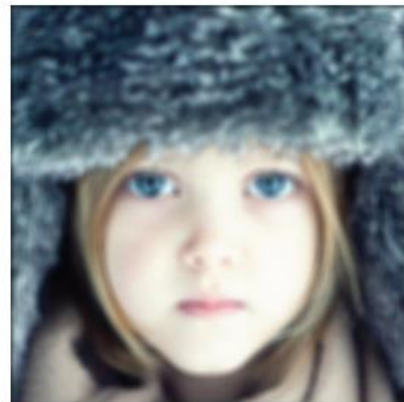
Ảnh xám

- Đầu vào: chandung.jpg
- Chức năng: Chuyển đổi ảnh RGB thành ảnh sepia
- Đầu ra: chandung_gray.jpg
- Thời gian: 0.14s



Ảnh sepia

- Đầu vào: chandung.jpg
- Chức năng: Làm mờ ảnh
- Đầu ra: chandung_blur.jpg
- Thời gian: 0.32s



Làm mờ ảnh

- Đầu vào: chandung.jpg
- Chức năng: Làm sắc nét ảnh
- Đầu ra: chandung_sharpen.jpg
- Thời gian: 0.25s



Làm sắc nét ảnh

- Đầu vào: chandung.jpg
- Chức năng: Cắt ảnh theo kích thước
- Đầu ra: chandung_square_crop.jpg
- Thời gian: 0.002s



Kích thước 50%

- Đầu vào: chandung.jpg
- Chức năng: Cắt ảnh theo khung hình tròn
- Đầu ra: chandung_circle_crop.jpg

- Thời gian: 0.07s



Khung tròn

- Đầu vào: chandung.jpg
- Chức năng: Cắt ảnh theo khung hình elip chéo nhau
- Đầu ra: chandung_elip_crop.jpg
- Thời gian: 0.18s



Khung elip chéo nhau

2. Hình 2: Kích thước 512x512 px

- Đầu vào: cat.jpg
- Chức năng: Thay đổi độ sáng cho ảnh.

- Đầu ra: cat_exposure.jpg
- Thời gian: 0.02s



Độ sáng +50

- Đầu vào: cat.jpg
- Chức năng: Thay đổi độ tương phản cho ảnh.
- Đầu ra: cat_contrast.jpg
- Thời gian: 0.02s



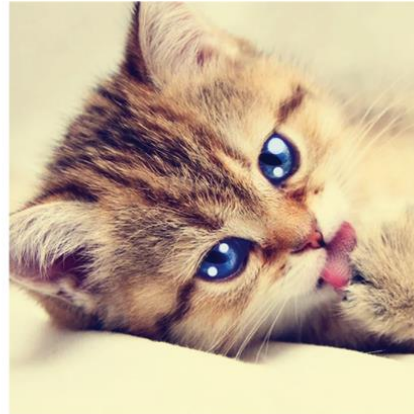
Độ tương phản +1.2

- Đầu vào: cat.jpg
- Chức năng: Lật ảnh ngang
- Đầu ra: cat_horizon.jpg
- Thời gian: 0.01s



Lật ảnh ngang

- Đầu vào: cat.jpg
- Chức năng: Lật ảnh dọc
- Đầu ra: cat_vertical.jpg
- Thời gian: 0.002s



Lật ảnh dọc

- Đầu vào: cat.jpg
- Chức năng: Chuyển đổi ảnh RGB thành ảnh xám
- Đầu ra: cat_gray.jpg
- Thời gian: 0.02s



Ảnh xám

- Đầu vào: cat.jpg
- Chức năng: Chuyển đổi ảnh RGB thành ảnh sepia
- Đầu ra: cat_sepia.jpg
- Thời gian: 0.03s



Ảnh sepia

- Đầu vào: cat.jpg
- Chức năng: Làm mờ ảnh
- Đầu ra: cat_blur.jpg
- Thời gian: 0.1s



Làm mờ ảnh

- Đầu vào: cat.jpg
- Chức năng: Làm sắc nét ảnh
- Đầu ra: cat_sharpen.jpg
- Thời gian: 0.07s



Làm sắc nét ảnh

- Đầu vào: cat.jpg
- Chức năng: Cắt ảnh theo kích thước
- Đầu ra: cat_square_crop.jpg
- Thời gian: 0.001s



Kích thước 50%

- Đầu vào: cat.jpg
- Chức năng: Cắt ảnh theo khung hình tròn
- Đầu ra: cat_circle_crop.jpg
- Thời gian: 0.01s



Khung tròn

- Đầu vào: cat.jpg
- Chức năng: Cắt ảnh theo khung hình elip chéo nhau
- Đầu ra: cat_elip_crop.jpg
- Thời gian: 0.04s



Khung elip chéo nhau

IV. Nhận xét các kết quả trên:

1. Nhận xét hình 1:

- 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.

2. Nhận xét hình 2:

- 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.

V. Tài liệu tham khảo:

- [1]. en.wikipedia.org, *Kernel (image processing)* (26.06.2023) tại: [https://en.wikipedia.org/wiki/Kernel_\(image_processing\)](https://en.wikipedia.org/wiki/Kernel_(image_processing))
- [2]. en.wikipedia.org, *Gaussian blur* (15.05.2023) tại: https://en.wikipedia.org/wiki/Gaussian_blur
- [3]. tranvantri.wordpress.com, *Từ RGB sang Đa mức xám (Grayscale)* (14.09.2014) tại: <https://tranvantri.wordpress.com/2014/09/14/tu-rgb-sang-da-muc-xam-grayscale/>
- [4]. dyclassroom.com, *How to convert a color image into sepia image* tại: <https://dyclassroom.com/image-processing-project/how-to-convert-a-color-image-into-sepia-image>
- [5]. vuontoanblog.blogspot.com, *Bài toán về tìm khoảng cách ngắn nhất và một tính chất của hình elíp* tại: <https://vuontoanblog.blogspot.com/2012/07/ellipse.html>
- [6]. blog.demofox.org, *Image Sharpening Convolution Kernels* (26.02.2022) tại: <https://blog.demofox.org/2022/02/26/image-sharpening-convolution-kernels/>
- [7]. towardsdatascience.com, *Types of Convolution Kernels : Simplified* (18.08.2019) tại: <https://towardsdatascience.com/types-of-convolution-kernels-simplified-f040cb307c37>
- [8]. vi.wikipedia.org, *Đường tròn* (30.03.2023) tại: https://vi.wikipedia.org/wiki/Đường_tròn
- [9]. kienthucviet.vn, *Định nghĩa , công thức Elip kèm bài tập (có đáp án)* tại: <https://www.kienthucviet.vn/blog/cong-thuc-elip/>
- [10]. vietjack.com, *Công thức về phép quay hay nhất* tại: <https://vietjack.com/tai-lieu-mon-toan/cong-thuc-ve-phep-quay-ctqt11.jsp>
- [11]. sharpsightlabs.com, *How to Use the Numpy Multiply Function* (18.10.2021) tại: <https://www.sharpsightlabs.com/blog/numpy-multiply/>

