



Đại học Khoa học Tự nhiên – ĐHQG TP.HCM

Môn học
Toán ứng dụng và thống kê cho Công nghệ thông tin

Đồ án 2: Image Processing

Lớp: 21CLC05

Thông tin sinh viên:

21127608 Trần Trung Hiếu

Giảng viên phụ trách:

Thầy Vũ Quốc Hoàng

Thầy Nguyễn Văn Quang Huy

Thầy Lê Thanh Tùng

Cô Phan Thị Phương Uyên

Tp Hồ Chí Minh, tháng 7 năm 2023

Mục lục

THÔNG TIN CÁ NHÂN.....	3
THÔNG TIN ĐỒ ÁN	3
BÁO CÁO ĐỒ ÁN.....	4
I. Bảng đánh giá:.....	4
II. Hàm chức năng:.....	7
1. Hàm cơ bản:	7
2. Hàm thay đổi độ sáng cho ảnh:	8
3. Hàm thay đổi độ tương phản:.....	9
4. Hàm lật ảnh (ngang/dọc):	9
5. Chuyển đổi ảnh RGB thành ảnh xám/sepia:.....	10
6. Làm mờ/sắc nét ảnh:	13
7. Cắt ảnh theo kích thước (cắt ở trung tâm):	16
8. Cắt ảnh theo khung tròn:	17
9. Cắt ảnh theo khung là 2 hình elip:	18
10. Hàm main:.....	20
TÀI LIỆU THAM KHẢO.....	21
Thông tin máy tính thực hiện đồ án.....	21

THÔNG TIN CÁ NHÂN

Họ và Tên: Trần Trung Hiếu

Mã số sinh viên: 21127608

Lớp: 21CLC05

THÔNG TIN ĐỒ ÁN

Mã học phần: MTH00057

Tên học phần: Toán ứng dụng và thống kê cho công nghệ thông tin

Tên đồ án: Image Processing





BÁO CÁO ĐỒ ÁN





I. Bảng đánh giá:

Hình ảnh ban đầu: kích thước 512x512



Tên chức năng	Mức độ hoàn thành
1. Thay đổi độ sáng cho ảnh	

2. Thay đổi độ tương phản	
3. Lật ảnh ngang	
4. Lật ảnh dọc	
5. Chuyển đổi ảnh RGB thành ảnh xám	

<p>6. Chuyển đổi ảnh RGB thành ảnh sepia</p>	
<p>7. Làm mờ ảnh</p>	
<p>8. Làm sắc nét ảnh</p>	
<p>9. Cắt ảnh theo kích thước (cắt ở trung tâm)</p>	

10. Cắt ảnh theo khung tròn	
11. Cắt ảnh theo khung là 2 hình elip chéo nhau	

II. Hàm chức năng:

1. Hàm cơ bản:

i. Hàm khởi tạo hình ảnh:

- **Đầu vào:** **name** là tên của bức ảnh dùng cho thuật toán.
- **Đầu ra:** một mảng hình 3 chiều có **kích thước (x, y, 3)**
- **Mô tả hàm chức năng:** Mở hình ảnh bằng hàm `Image.open()` có trong thư viện `Image` thuộc `PIL` sau đó chuyển hình ảnh thành dạng mảng bằng hàm `np.array()`.

ii. Hàm lưu hình ảnh:

- **Đầu vào:** **img** là mảng 3 chiều của hình ảnh có **kích thước (x, y, 3)**, **name** là tên đầu vào của bức hình lúc đầu, **type_change** là chức năng xử lý ảnh tương ứng.
- **Mô tả hàm chức năng:** Lưu hình ảnh gồm tên ban đầu, chức năng xử lý ảnh và lưu ảnh ở dạng **png**.

iii. Hàm hiển thị hình ảnh:

- **Đầu vào:** `img_original` là tên của bức ảnh dùng cho thuật toán, `img_result` là một mảng hình 3 chiều có **kích thước (x, y, 3)** sau khi đã xử lý chức năng.
- **Mô tả hàm chức năng:** Sử dụng thư viện `matplotlib.pyplot`^[1] để hiển thị hình ảnh, ta dùng `plt.figure()` để chỉnh kích thước của khung ảnh hiển thị. Đầu tiên, ta mở ảnh ban đầu bằng hàm `Image.open()`, và chuyển đổi mảng ảnh (có **kiểu dữ liệu unit8**) kết quả thành một đối tượng hình ảnh bằng hàm `Image.fromarray()` và hiển thị hình ảnh bằng hàm `plt.subplot()` dùng tạo các ô trong bảng để có thể có hình ảnh ban đầu và kết quả trên cùng một hàng. Dùng hàm `plt.imshow()` để có thể hiển thị hình ảnh.

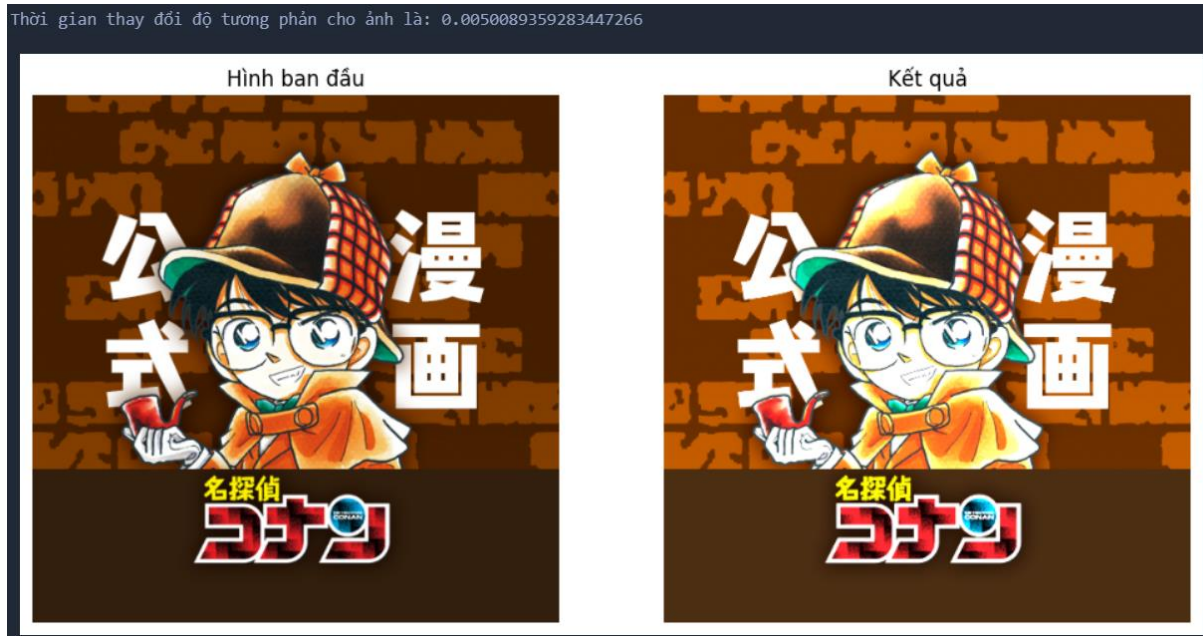
2. Hàm thay đổi độ sáng cho ảnh:



- **Đầu vào:** `img` là mảng 3 chiều của hình ảnh có **kích thước (x, y, 3)**
- **Đầu ra:** một mảng hình 3 chiều sau khi tăng độ sáng có **kích thước (x, y, 3)**
- **Ý tưởng:** Thay đổi độ sáng cho ảnh là thay đổi 1 lượng màu để ảnh có thể sáng hơn hoặc tối đi. Ta cộng mảng hình với 1 số anpha để có thể thay đổi độ sáng của bức hình. Nhưng phải giữ thành phần màu của hình trong giới hạn từ 0 đến 255
- **Mô tả hàm chức năng:** Cộng/trừ từng pixel của ảnh với trung bình của các pixel của ảnh bằng hàm `np.min()` (tương ứng cộng theo R, G, B) (nếu muốn tăng độ sáng của hình ảnh ta thực hiện phép cộng nếu muốn giảm độ sáng của hình ảnh ta dùng phép trừ). Dùng hàm `np.clip()`^[2] để giới hạn các phần tử trong mảng trong khoảng từ 0 đến 255 để đảm bảo các phần tử nằm trong giới hạn màu

RGB. Định dạng các phần tử của mảng là kiểu nguyên không dấu 8 bits để biểu diễn mã màu của các phần tử trong khoảng từ 0 đến 255.

3. Hàm thay đổi độ tương phản:



- **Đầu vào:** **img** là một mảng 3 chiều của hình ảnh có **kích thước** $(x, y, 3)$, **kernel** là mức độ độ tương phản muốn thay đổi cho hình.
- **Đầu ra:** một mảng hình 3 chiều sau khi tăng độ tương phản có **kích thước** $(x,y,3)$
- **Ý tưởng:** Thay đổi độ chênh lệch sáng tối của bức hình rõ hơn. Ta nhân ma trận hình ảnh với 1 số anpha để có thể thay đổi độ chênh lệch màu sáng và tối rõ ràng hơn (Nếu anpha > 1 độ tương phản tăng lên, nếu anpha < 1 độ tương phản sẽ giảm đi). Nhưng phải giữ thành phần màu của hình trong giới hạn từ 0 đến 255.
- **Mô tả hàm chức năng:** Nhân từng phần tử của mảng hình với hệ số **kernel** (nếu hệ số có giá trị từ 0 đến 1 thì độ tương phản sẽ giảm đi, còn nếu hệ số lớn hơn 1 thì độ tương phản tăng lên). Dùng hàm `np.clip()`^[2] hỗ trợ giới hạn các phần tử trong mảng trong khoảng từ 0 đến 255 để đảm bảo các phần tử nằm trong giới hạn màu RGB. Định dạng các phần tử của mảng là kiểu nguyên không dấu 8 bits để biểu diễn mã màu của các phần tử trong khoảng từ 0 đến 255.

4. Hàm lật ảnh (ngang/dọc):

Thời gian lật ảnh ngang là: 0.0
Thời gian lật ảnh dọc là: 0.0



- **Đầu vào:** `img` là một mảng 3 chiều của hình ảnh có **kích thước** $(x, y, 3)$
- **Đầu ra:** một mảng hình 3 chiều sau khi lật ảnh theo chiều ngang/dọc có **kích thước** $(x, y, 3)$
- **Ý tưởng:** Thực hiện đảo các cột/các dòng để có thể thay đổi lật ảnh theo chiều ngang/dọc
- **Mô tả hàm chức năng:** Sử dụng kỹ thuật **slicing**^[3] bằng cách dùng **step = - 1** của dòng hoặc cột của ảnh để có thể lật ngược ảnh.

5. Chuyển đổi ảnh RGB thành ảnh xám/sepia:

Thời gian chuyển ảnh RGB thành xám là: 0.004996776580810547
Thời gian chuyển ảnh RGB thành sepia là: 0.014034032821655273



i. Hàm chuyển đổi ảnh RGB sang ảnh xám:

- **Đầu vào:** **img** là một mảng **3 chiều** của hình ảnh có **kích thước (x, y, 3)**
- **Đầu ra:** một mảng hình **2 chiều** sau khi đổi ảnh RGB thành ảnh xám có **kích thước (x, y)**
- **Ý tưởng:** Đưa 1 pixel của hình ảnh từ 3 thành phần màu sang 1 thành phần màu. Ta thực hiện nhân tích vô hướng từng pixel (1 mảng có 3 phần tử) của ảnh là RGB theo công thức **pixel mới = (0.3*R + 0.59*G + 0.11*B)^[4]** để tạo ra một pixel mới (1 số) để tạo hình xám.

- **Mô tả hàm chức năng:** Nhân ma trận hình ảnh với ma trận **[0.3, 0.59, 0.11]**^[4] (Lấy từng pixel **nhân tích vô hướng** với mảng trên và trả về một số). Kiểm tra xem giá trị của từng pixel có vượt quá 255 không nếu có thì thay đổi điểm màu của pixel đó có giá trị 255. Định dạng các phần tử của mảng là kiểu nguyên không dấu 8 bits để biểu diễn mã màu của các phần tử trong khoảng từ 0 đến 256.

ii. Hàm chuyển ảnh RGB sang ảnh sepia:

- **Đầu vào:** **img** là một mảng 3 chiều của hình ảnh có **kích thước (x, y, 3)**
- **Đầu ra:** một mảng hình 3 chiều sau khi đổi ảnh RGB thành ảnh sepia có **kích thước (x, y, 3)**
- **Ý tưởng:** Nhân từng pixel (1 mảng có 3 phần tử) của ảnh RGB theo công thức

$$tr = 0.393R + 0.769G + 0.189B$$

$$tg = 0.349R + 0.686G + 0.168B$$

$$tb = 0.272R + 0.534G + 0.131B$$

^[5] để tạo hình sepia tương ứng với [tr, tg, tb] là pixel mới của ảnh tại vị trí tương ứng (1 mảng có 3 phần tử) (lấy R, G, B nhân với từng hệ số tương ứng có thể áp dụng nhân 2 ma trận với nhau)

- **Mô tả hàm chức năng:** Nhân ma trận hình ảnh với ma trận **[[0.393, 0.769, 0.189], [0.349, 0.686, 0.168], [0.272, 0.534, 0.131]]**^[5] (Lấy từng pixel nhân với từng mảng trên **sau khi chuyển vị** (ta sẽ nhân pixel đó với từng cột của ma trận trên sau khi chuyển vị sẽ được công thức tương ứng trên) và cộng các pixel đó lại theo dạng cột bằng hàm **np.dot()**^[7] (hàm thực hiện nhân 2 ma trận với nhau sau đó cộng theo cột của các pixel lại với nhau) trả về một pixel mới có 3 phần tử). Kiểm tra xem giá trị của từng pixel có vượt quá 255 không nếu có thì thay đổi điểm màu của pixel đó có giá trị 255. Định dạng các phần tử của mảng là kiểu nguyên không dấu 8 bits để biểu diễn mã màu của các phần tử trong khoảng từ 0 đến 256.

iii. Hàm in ảnh của ảnh xám:

- **Đầu vào:** **img_original** là tên của bức ảnh dùng cho thuật toán, **img_result** là một mảng hình 3 chiều có **kích thước (x, y, 3)** sau khi đã xử lý chức năng.
- **Mô tả hàm chức năng:** Sử dụng thư viện **matplotlib.pyplot**^[1] để hiển thị hình ảnh, ta dùng **plt.figure()** để chỉnh kích thước của khung ảnh hiển thị. Đầu tiên, ta mở ảnh ban đầu bằng hàm **Image.open()**, và chuyển đổi mảng ảnh (có **kiểu dữ liệu unit8**) kết quả thành một đối tượng hình ảnh bằng hàm **Image.fromarray()** và hiển thị hình ảnh bằng hàm **plt.subplot()** dùng tạo các ô

trong bảng để có thể có hình ảnh ban đầu và kết quả trên cùng một hàng. Dùng hàm plt.imshow() để có thể hiển thị hình ảnh.

- **Điểm khác biệt:** trong hàm plt.imshow() để hiển thị hình ảnh của kết quả ảnh xám cần dùng thêm cmap = 'gray' để có thể hiển thị hình ảnh dạng xám vì nếu không có thì cmap sẽ mặc định có giá trị viridis (một colormap chuẩn với dải màu từ xanh dương đến vàng. Nó hữu ích để hiển thị dữ liệu có giá trị tăng dần.)

6. Làm mờ/sắc nét ảnh:



i. Hàm làm mờ ảnh

- **Ý tưởng chức năng:** Ta lấy từng pixel đó trong bức hình (bỏ qua pixel ở biên của bức hình) và những pixel lân cận của pixel đó, ta được 1 mảng 3 chiều, ta nhân từng pixel của mảng đó với từng phần tử tương ứng trong mảng:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix},^{[5]}$$

sau đó ta lấy tổng các pixel vừa nhận được và nhân cho $\frac{1}{9}$ [5] ta được 1 pixel mới tương ứng với vị trí pixel ban đầu, thực hiện cho toàn bộ pixel trong hình (bỏ qua pixel ở biên của bức hình) ta được 1 hình sau khi làm mờ.

- **Hàm làm mờ 1 pixel:**

- + **Đầu vào:** **arr_pixel** là một mảng 3 chiều có **kích thước (3, 3, 3)** là mảng gồm pixel dùng để làm mờ và các pixel xung quanh pixel đó.
- + **Đầu ra:** một mảng 1 chiều là pixel đã làm mờ sau có thành phần màu là RGB.
- + **Ý tưởng:** Thực hiện thay đổi thành phần của pixel trong hình với pixel lân cận pixel đó (bên trái, bên phải, bên trên, bên dưới, chéo trên, chéo dưới) bằng cách tính tổng các thành phần màu R, G, B tương ứng. Ta nhân từng bộ pixel với mảng [5] (nhân từng pixel với 1) sau đó tính tổng của các pixel sau khi nhân ta được một pixel mới. Thực hiện toàn bộ pixel trên hình ta được hình ảnh đã làm mờ (trừ pixel ở biên).
- + **Mô tả hàm chức năng:** thay đổi mảng **arr_pixel** thành mảng 2 chiều có kích thước (x*y, 3) bằng hàm reshape() sau đó tính tổng các phần tử tương ứng theo R, G, B bằng hàm **np.sum()** có **axis = 0** (tính tổng theo cột) ta được 1 pixel mới là pixel cần tìm.

- **Hàm làm mờ hình:**

- + **Đầu vào:** **img** là một mảng 3 chiều của hình ảnh có **kích thước (x, y, 3)**
- + **Đầu ra:** một mảng hình 3 chiều là mảng của hình ảnh sau khi làm mờ ảnh có **kích thước (x, y, 3)**
- + **Ý tưởng:** Thực hiện thay đổi thành phần của từng pixel trong hình với pixel lân cận pixel đó (bên trái, bên phải, bên trên, bên dưới, chéo trên, chéo dưới) bằng cách tính trung bình cộng các thành phần màu R, G, B tương ứng.
- + **Mô tả hàm chức năng:** xác định vị trí của từng pixel bằng 2 vòng lặp (trừ những pixel nằm ở biên), sau đó ta làm mờ từng pixel bằng cách xác

định pixel đó và lân cận của pixel đó bằng **slicing** [chiều dài-1: chiều dài +2, chiều rộng-1: chiều rộng+2, :]. Dùng hàm làm mờ 1 pixel và ta có pixel tại vị trí tương ứng đã được làm mờ. Ta gán vào 1 mảng 3 chiều mới. Sau khi làm mờ toàn bộ hình ảnh. Sau khi thực hiện toàn bộ pixel ta nhân mảng kết quả với $\frac{1}{9}$ và định dạng các phần tử của mảng là kiểu nguyên không dấu 8 bits để biểu diễn mã màu của các phần tử trong khoảng từ 0 đến 256.

ii. Hàm làm rõ ảnh:

- **Ý tưởng chức năng:** Ta lấy từng pixel đó trong bức hình (bỏ qua pixel ở biên của bức hình) và những pixel lân cận của pixel đó, ta được 1 mảng 3 chiều, ta nhân từng pixel của mảng đó với từng phần tử tương ứng trong mảng:

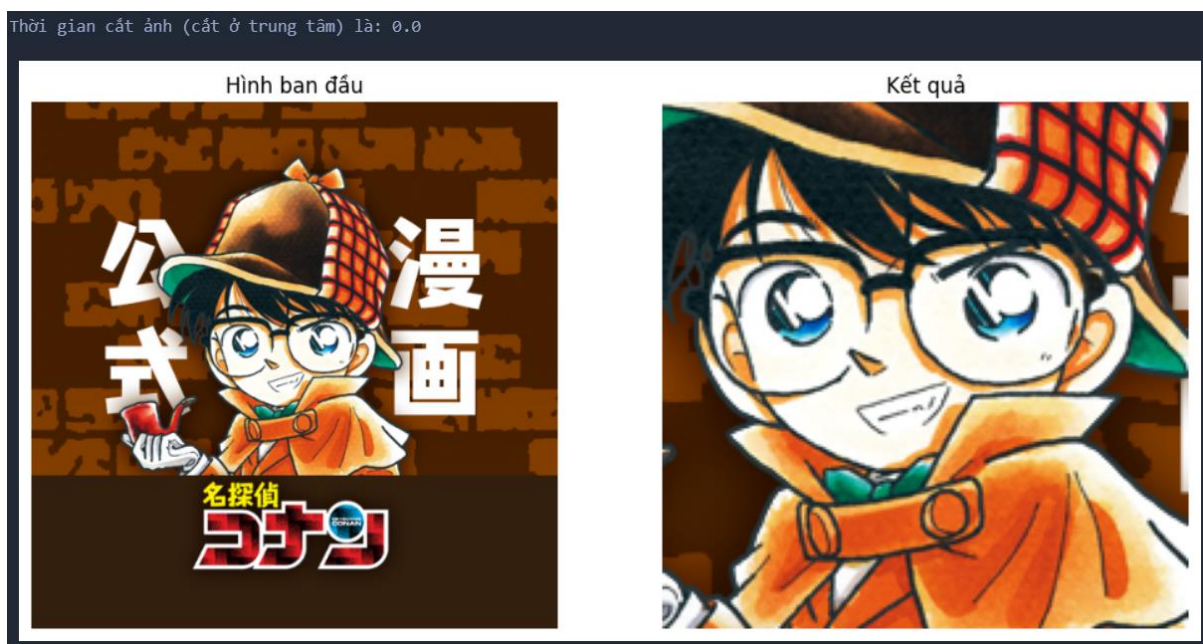
$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix},^{[6]}$$

sau đó ta lấy tổng các pixel vừa nhận ta được 1 pixel mới tương ứng với pixel ban đầu, thực hiện cho toàn bộ pixel trong hình (bỏ qua pixel ở biên của bức hình) ta được 1 hình sau khi làm rõ.

- **Hàm làm rõ 1 pixel:**
 - + **Đầu vào:** **arr_pixel** là một mảng 3 chiều có **kích thước (3, 3, 3)** là mảng gồm pixel dùng để làm rõ và các pixel xung quanh
 - + **Đầu ra:** một mảng 1 chiều (lần lượt là RGB) của pixel đã làm rõ
 - + **Ý tưởng:** Thực hiện thay đổi thành phần của pixel trong hình với pixel lân cận pixel đó (bên trái, bên phải, bên trên, bên dưới, chéo trên, chéo dưới) bằng cách tính tổng các thành phần màu R, G, B tương ứng với hệ số của ma trận. Ta nhân từng pixel với ma trận ^[5] (nhân từng pixel với hệ số có cùng vị trí) sau đó tính tổng của các pixel sau khi nhân ta được pixel mới.
 - + **Mô tả hàm chức năng:** thay đổi mảng **arr_pixel** thành mảng 2 chiều có kích thước (3*3, 3) bằng hàm reshape(). Thay đổi mảng kernel (mảng ^[6]) thành mảng 2 chiều có kích thước (9, 1). Sau đó lấy từng phần tử của mảng kernel đã chuyển vị nhân từng pixel của mảng arr_pixel tương ứng bằng hàm np.dot()^[7] (dùng phép tích 2 ma trận có kích thước (1,9) và (9,3)) ta được 1 mảng 2 chiều có 3 thành phần màu tương ứng với R G B sau đó ta chuyển thành mảng 1 chiều.
- **Hàm làm rõ hình:**

- + **Đầu vào:** **img** là một mảng 3 chiều của hình ảnh có **kích thước (x, y, 3)**
- + **Đầu ra:** một mảng hình 3 chiều là mảng của hình ảnh sau khi làm rõ ảnh có **kích thước (x, y, 3)**
- + **Ý tưởng:** Thực hiện thay đổi thành phần của từng pixel trong hình với pixel lân cận pixel đó (bên trái, bên phải, bên trên, bên dưới, chéo trên, chéo dưới) bằng cách nhân với mảng^[6] để có mảng pixel mới.
- + **Mô tả hàm chức năng:** xác định vị trí của từng pixel bằng 2 vòng lặp (trừ những pixel nằm ở biên), sau đó ta làm rõ từng pixel bằng cách xác định pixel đó và lân cận của pixel đó bằng **slicing** [chiều dài-1: chiều dài+2, chiều rộng-1: chiều rộng+2, :]. Dùng hàm làm rõ từng pixel và ta có pixel tại vị trí tương ứng đã được làm rõ. Ta gán vào một mảng 3 chiều mới. Sau khi làm rõ toàn bộ hình ảnh. Định dạng các phần tử của mảng là kiểu nguyên không dấu 8 bits để biểu diễn mã màu của các phần tử trong khoảng từ 0 đến 256.

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



- **Đầu vào:** **img** là một mảng 3 chiều của hình ảnh có **kích thước (x, y, 3)**
- **Đầu ra:** một mảng hình 3 chiều sau khi cắt hình ảnh vuông theo kích thước nhỏ hơn có (cắt ở trung tâm tám hình) có **kích thước (x, y, 3)**
- **Ý tưởng:** Xác định tâm của tám hình, từ tâm ta lấy ra 1 đoạn theo chiều dài, chiều rộng để có thể tạo ra hình mới sau khi cắt, còn những điểm không nằm trong đoạn đó sẽ là màu đen. Ta được 1 tám hình hình vuông sau khi cắt từ tâm.

- **Mô tả hàm chức năng:** Tìm tâm của hình bằng cách sử dụng hàm **shape** để có thể lấy kích thước của tấm hình và lưu vào `img_space`, sau đó lấy `img_space[0]//2` và `img_space[1]//2` tương ứng với việc lấy chiều dài và chiều rộng của hình sau đó chia cho 2 lấy nguyên để được tọa độ của tâm hình. Từ tâm tấm hình dùng kỹ thuật **slicing** để có thể lấy từ tâm của tấm hình qua trái, qua phải, lên trên, xuống dưới 1 đoạn `length` để có thể cắt tấm hình theo hình vuông có chiều dài và chiều rộng bức hình bằng **$2*length$** .

8. Cắt ảnh theo khung tròn:



- **Đầu vào:** **img** là một mảng 3 chiều của hình ảnh có **kích thước (x, y, 3)**
- **Đầu ra:** một mảng hình 3 chiều sau khi cắt theo khung hình tròn có **kích thước (x, y, 3)**
- **Ý tưởng:** Xác định tâm của tấm hình, xác định bán kính của hình tròn (bằng chiều dài nếu chiều dài tấm hình nhỏ hơn hoặc ngược lại bằng chiều rộng tấm hình nếu chiều rộng tấm hình nhỏ hơn). Xác định đường tròn của tấm hình đó, những điểm màu nằm trong và trên đường tròn thì giữ nguyên màu còn những điểm màu nằm ngoài đường tròn sẽ chuyển thành màu đen.

Mô tả hàm chức năng: Xác định chiều cao và chiều rộng của bức hình bằng **shape** để xác định tâm của hình tròn và bán kính của hình tròn bằng độ dài của cạnh nhỏ hơn chia 2 (lấy phần nguyên). Lấy vị trí của từng pixel bằng cách tạo hai mảng `x` và `y` chứa tọa độ các pixel trên ảnh theo trục `x` và `y` bằng hàm **`np.ogrid()`**^[9] và tạo mảng **mask** bool chứa giá trị `True/False` thỏa điều kiện pixel nằm trong hay ngoài hình tròn (`True` nếu tọa độ điểm đó nằm trong và

trên đường tròn và False nếu nằm ngoài hình tròn) bằng phương trình đường tròn: $(x-x_0)+(y-y_0) = R^2$ Dùng mảng mask kiểm tra những điểm nằm ngoài hình tròn thì sẽ thay đổi thành phần màu thành màu đen (có giá trị bằng 0). Định dạng các phần tử của mảng là kiểu nguyên không dấu 8 bits để biểu diễn mã màu của các phần tử trong khoảng từ 0 đến 256.

9. Cắt ảnh theo khung là 2 hình elip:



- **Đầu vào:** **img** là một mảng 3 chiều của hình ảnh có **kích thước (x, y, 3)**
- **Đầu ra:** một mảng hình 3 chiều sau cắt hình theo khung có hình ellips có **kích thước (x, y, 3)**
- **Ý tưởng:** Xác định tâm của tấm hình ellips là tâm của tấm hình, xác định góc quay của 2 hình ellips để có trục chính và trục phụ trùng với 2 đường chéo của bức hình. Xác định trục lớn, trục bé của 2 hình ellips sau khi xoay và xác định 2 hình ellips đồng tâm đó, những điểm màu nằm trong và trên đường cong của hình ellips thì giữ nguyên màu còn những điểm màu nằm ngoài sẽ chuyển thành màu đen.
- **Mô tả hàm chức năng:** Chứng minh toán học ta có chiều trục lớn có hình ellips bằng $\frac{7R \cdot \sin(\theta)}{4}$ và trục bé của hình ellips bằng $R \cdot \cos(\theta)$ (trong đó θ là góc qua của hình ellips đứng xoay trái để được hình ellips nằm chéo có trục lớn dọc theo đường chéo chính bức hình). Tạo ra các mảng một chiều có kích thước bằng với kích thước của `img_space[0]` và `img_space[1]`. Tìm bán kính $R1 = \text{img_space}[0] // 2$ và $R2 = \text{img_space}[1] // 2$ (chia kích thước ảnh

cho 2 lấy nguyên). Tìm bán kính R nhỏ hơn. Tạo hai mảng x và y chứa tọa độ các pixel trên ảnh theo trục Oxy bằng hàm **np.ogrid()**^[9]. Sau khi có mảng tọa độ của các pixel ta thực hiện chiếu tọa độ của hình đó lên 2 trục tọa độ mới có trục tọa độ trùng với đường chéo của bức hình (xoay trái 1 góc theta và xoay phải 1 góc -theta) (theta = 45°) theo công thức^[8]:

$$x_new = x * \cos(\theta) - y * \sin(\theta)$$

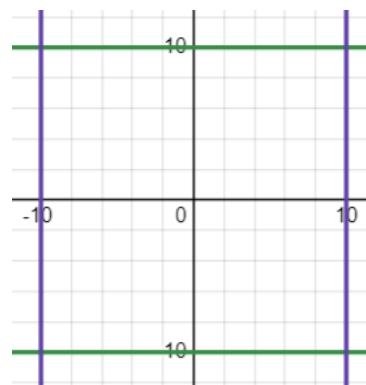
$$y_new = x * \sin(\theta) + y * \cos(\theta)$$

và

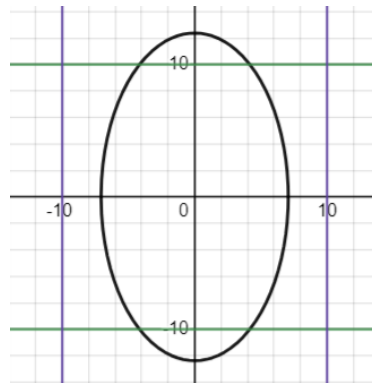
$$x_new = x * \cos(-\theta) - y * \sin(-\theta)$$

$$y_new = x * \sin(-\theta) + y * \cos(-\theta)$$

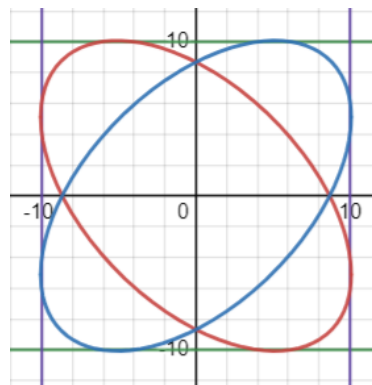
Sau khi tìm được tọa độ trên trục chéo tương ứng ta tạo mảng **mask** bool chứa giá trị True/False thỏa điều kiện nằm trong hay ngoài hình ellips (True nếu tọa độ điểm đó nằm trong và trên đường ellips và False nếu nằm ngoài hình ellips) bằng phương trình ellips $\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2$. Dùng mảng mask kiểm tra những điểm nằm ngoài ellips thì sẽ thay đổi thành phần màu thành màu đen (có giá trị bằng 0). Định dạng các phần tử của mảng là kiểu nguyên không dấu 8 bits để biểu diễn mã màu của các phần tử trong khoảng từ 0 đến 256.



Xác định tâm của tam hình (coi như là điểm O(0, 0)) và gắn vào trục tọa độ Oxy



Xác định hình ellips ban đầu sau khi có tâm $O(0, 0)$, trục lớn bằng $\frac{7R \cdot \sin(\theta)}{4}$ và trục bé bằng $R \cdot \cos(\theta)$



Sau khi thực hiện xoay hình ellips trên 1 góc 45° và -45° ta được 2 hình ellips đồng tâm có trục lớn và trục bé tương ứng trùng với 2 đường chéo của hình vuông

10. Hàm main:

- Nhập tên của tám hình cần thực hiện chức năng (bức hình đầu vào phải có 3 kênh màu có kích thước $(x, y, 3)$).
- Nhập chức năng muốn xử lý hình ảnh:
 - + 0: thực hiện tất cả chức năng
 - + 1: thực hiện làm sáng hình
 - + 2: thực hiện thay đổi độ tương phản cho hình
 - + 3: thực hiện lật ngang/dọc bức hình
 - + 4: thực hiện chuyển ảnh RGB sang ảnh xám/sepia
 - + 5: thực hiện làm mờ/làm sắc nét ảnh
 - + 6: thực hiện cắt ảnh ở trung tâm

- + 7: thực hiện cắt ảnh theo khung tròn
- + 8: thực hiện cắt ảnh theo 2 khung ellips
- Sau khi thực hiện chức năng sẽ hiện thị thời gian thực hiện chức năng và hình ảnh trước và sau khi thực hiện chức năng

TÀI LIỆU THAM KHẢO

- [1] Tham khảo thư viện matplotlib.pyplot [matplotlib.pyplot.imshow — Matplotlib 3.7.2 documentation](https://matplotlib.org/3.7.2/documentation)
- [2] Tham khảo hàm np.clip() [numpy.clip\(\) in Python - GeeksforGeeks](https://www.geeksforgeeks.org/numpy-clip-in-python/)
- [3] Tham khảo slicing trong python [Python Slicing – How to Slice an Array and What Does \[::-1\] Mean? \(freecodecamp.org\)](https://www.freecodecamp.org/python-slicing/)
- [4] Tham khảo tỷ lệ màu chuyển ảnh RGB sang ảnh xám [Grayscale to RGB Conversion | Tutorialspoint](https://www.tutorialspoint.com/python/convert-rgb-to-grayscale-image.htm)
- [5] Tham khảo tỷ lệ màu chuyển ảnh RGB sang ảnh sepia [How to convert a color image into sepia image - Image Processing Project - dyclassroom | Have fun learning :-\)](https://www.dyclassroom.com/learn/python/image-processing/convert-rgb-to-sepia-image/)
- [6] Tham khảo tỷ lệ để làm ảnh mờ [Box blur - Wikipedia](https://en.wikipedia.org/wiki/Box_blur)
- [7] Tham khảo np.dot() [numpy.dot\(\) in Python - GeeksforGeeks](https://www.geeksforgeeks.org/numpy-dot-in-python/)
- [8] Tham khảo xoay hình ellips [geometry - What is the general equation of the ellipse that is not in the origin and rotated by an angle? - Mathematics Stack Exchange](https://math.stackexchange.com/questions/1111111/geometry-what-is-the-general-equation-of-the-ellipse-that-is-not-in-the-origin-and-rotated-by-an-angle/)
- [9] Tham khảo np.ogrid() [numpy.ogrid — NumPy v1.25 Manual](https://numpy.org/doc/1.25/manual/)

Thông tin máy tính thực hiện đồ án

Hệ điều hành: Windows 10 Home Single

Bộ xử lý: Core i7-1165G7 @ 2.80Hz (8 CPUs) ~ 2.8GHz

Bộ nhớ: 16GB