

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN 2: IMAGE PROCESSING

Môn: Toán ứng dụng và thống kê

Lớp: 22CLC03

Sinh viên:

Hoàng Bảo Khanh (22127183)

Giáo viên hướng dẫn:

ThS. Phan Thị Phương Uyên
ThS. Nguyễn Văn Quang Huy



Mục lục

1	Giới thiệu đồ án	3
2	Ý tưởng thực hiện	4
2.1	Đọc file và chuyển đổi ảnh sang ma trận	4
2.2	Thay đổi độ sáng của ảnh	4
2.3	Thay đổi độ tương phản của ảnh	4
2.4	Lật ảnh ngang	5
2.5	Lật ảnh dọc	5
2.6	Chuyển đổi RGB thành ảnh xám	5
2.7	Chuyển đổi RGB thành ảnh sepia	5
2.8	Làm mờ ảnh	6
2.9	Làm sắc nét ảnh	7
2.10	Cắt ảnh theo kích thước	7
2.11	Cắt ảnh theo khung tròn	7
2.12	Cắt ảnh theo khung elip	8
2.13	Phóng to, thu nhỏ ảnh	9
3	Mô tả các hàm	11
3.1	Hàm đọc file (<i>read_img</i>)	11
3.2	Hàm thể hiện hình ảnh (<i>show_img</i>)	11
3.3	Hàm lưu ảnh (<i>save_img</i>)	11
3.4	Hàm thực hiện thay đổi độ sáng (<i>changeBrightness</i>)	11
3.5	Hàm thực hiện thay đổi độ tương phản (<i>changeContrast</i>)	11
3.6	Hàm lật ảnh dọc (<i>flipImageVertically</i>)	11
3.7	Hàm lật ảnh ngang (<i>flipImageHorizontally</i>)	12
3.8	Hàm đổi ảnh sang ảnh xám (<i>grayScaleImage</i>)	12
3.9	Hàm đổi ảnh sang ảnh sepia (<i>sepiaScaleImage</i>)	12
3.10	Hàm làm mờ ảnh (<i>blur_image</i>)	12
3.11	Hàm làm sắc nét ảnh (<i>sharpen_image</i>)	12
3.12	Hàm cắt ảnh ở trung tâm (<i>cropCenter</i>)	13
3.13	Hàm cắt ảnh theo khung tròn (<i>cropCircular</i>)	13
3.14	Hàm cắt ảnh theo khung elip (<i>cropEllipses</i>)	13
3.15	Hàm phóng to, thu nhỏ ảnh (<i>zoom_image</i>)	14
3.16	Hàm thực hiện tất cả chức năng (<i>do_all</i>)	14
3.17	Hàm Main	14

4	Bảng đánh giá mức độ hoàn thành và hình ảnh kết quả	15
5	Tài liệu tham khảo	19

1 Giới thiệu đề án

Trong đề án lần này, em sẽ viết chương trình có khả năng xử lý ảnh với các chức năng cơ bản như:

- Thay đổi độ sáng cho ảnh
- Thay đổi độ tương phản
- Lật ảnh (ngang - dọc)
- Chuyển đổi ảnh RGB thành ảnh xám và ảnh sepia
- Làm mờ và làm sắc nét ảnh
- Cắt ảnh theo kích thước (cắt ở trung tâm)
- Cắt ảnh theo khung (khung tròn và khung 2 hình elip chéo nhau)

Để thực hiện các nội dung trên, em lập trình bằng ngôn ngữ python và sử dụng các thư viện chính như thư viện numpy, matplotlib, image.

2 Ý tưởng thực hiện

Dưới đây là những ý tưởng chính để em thực hiện các chức năng trong chương trình

2.1 Đọc file và chuyển đổi ảnh sang ma trận

Việc đầu tiên chương trình sẽ làm là yêu cầu người dùng nhập địa chỉ của ảnh. Sau đó chương trình sẽ đọc file ảnh và chuyển đổi ảnh sang ma trận hai chiều với các thông tin bao gồm:

- Độ rộng: là số pixels có được trên cùng 1 hàng
- Độ cao: là số pixels có được trên cùng 1 cột
- Số lượng kênh màu: đối với ảnh đầu vào là RGB thì số lượng kênh màu là 3, đối với ảnh trắng đen thì số lượng kênh màu là 2

2.2 Thay đổi độ sáng của ảnh

Để thay đổi độ sáng của ảnh, ý tưởng chính là sẽ lấy các giá trị RGB của từng pixel cộng với một giá trị độ sáng cho trước. Trong chương trình này, giá trị mặc định sẽ là 50.

2.3 Thay đổi độ tương phản của ảnh

Để thay đổi độ tương phản, ta cần phải tăng các khoảng cách giữa các điểm ảnh bên trong. Trong chương trình, mức độ tương phản mặc định sẽ bằng 50. Từ đó, chương trình trước tiên sẽ tính hệ số hiệu chỉnh độ tương phản dựa vào công thức sau:

$$F = \frac{259(C + 255)}{255(259 - C)}$$

Hình 1: Công thức tính hệ số hiệu chỉnh tương phản

Sau đó, để tính giá trị điểm ảnh mới, ta áp dụng biểu thức sau:

$$R' = F(R - 128) + 128$$

Hình 2: Công thức tính giá trị RGB mới

Với R' là giá trị Red mới, R là giá trị Red ban đầu và F là hệ số đã được tính ở trên. Tương tự, ta áp dụng cho giá trị Green và Blue. Từ đó ta có được ma trận điểm ảnh mới.

2.4 Lật ảnh ngang

Đối với lật ảnh theo chiều ngang, ta phải tìm cách lấy đối xứng các điểm ảnh qua trục thẳng đứng, các điểm ảnh ban đầu đang nằm bên trái sẽ được lấy đối xứng qua phía bên phải và ngược lại, các điểm ảnh ban đầu nằm bên phải sẽ được lấy đối xứng sang trái.

Do đó, trước tiên chương trình sẽ gọi ra một ma trận mới với cùng kích thước so với ma trận ban đầu và chương trình sẽ duyệt các điểm ảnh từ ma trận ban đầu theo thứ tự từ phải sang trái và gán vào ma trận mới.

2.5 Lật ảnh dọc

Tương tự với lật ảnh ngang, lật ảnh theo chiều dọc tức là lấy đối xứng các điểm ảnh qua trục ngang. Do đó, chương trình cũng sẽ gọi ra một ma trận mới có cùng kích thước và sẽ duyệt ma trận ban đầu các điểm ảnh từ dưới lên trên.

2.6 Chuyển đổi RGB thành ảnh xám

Để chuyển đổi ảnh màu sang ảnh xám, ý tưởng thực hiện phổ biến nhất sẽ là tính giá trị trung bình cộng giữa các giá trị RGB trong mỗi điểm ảnh. Tuy nhiên, việc tính trung bình cộng của rất nhiều vô số điểm ảnh sẽ tốn rất nhiều thời gian.

Do đó để khắc phục tình trạng trên, chương trình sẽ dùng Weight Method, tức lấy ma trận ban đầu nhân với một ma trận 1 chiều với các giá trị đặc biệt. Trong chương trình, ma trận đặc biệt này có dạng $[0.299, 0.587, 0.114]$. Kết quả của phép nhân trên sẽ là ma trận của ảnh đã được chuyển sang ảnh xám.

2.7 Chuyển đổi RGB thành ảnh sepia

Dựa vào ý tưởng dùng Weight Method, chương trình sẽ dùng phép toán như sau:

- $tr = 0.393R + 0.769G + 0.189B$ (tr là giá trị Red mới)

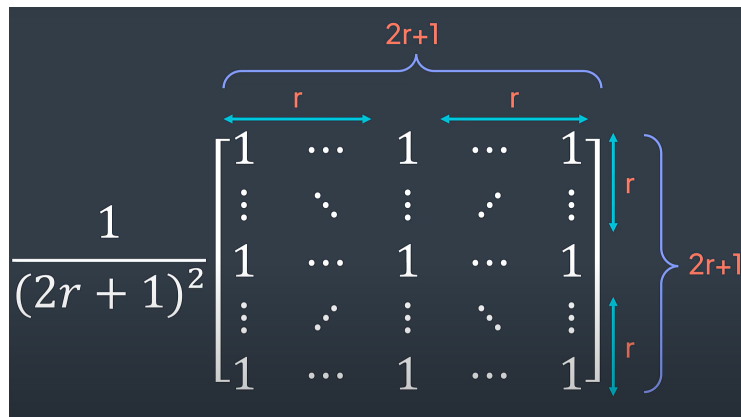
- $t_g = 0.349R + 0.686G + 0.168B$ (t_g là giá trị Green mới)
- $t_b = 0.272R + 0.534G + 0.131B$ (t_b là giá trị Blue mới)

Giả sử một điểm ảnh có các giá trị RGB lần lượt là 100, 150 và 200. Để tính t_r , ta sẽ lấy $0.393 \times 100 + 0.769 \times 150 + 0.189 \times 200 = 192.45$. Khi đó giá trị Red mới sẽ là 192 sau khi lấy phần nguyên. Tương tự, ta sẽ tính được t_g và t_b , tức giá trị mới của Green và Blue. Khi đó điểm ảnh đã được chuyển thành sepia thành công. Và để thực hiện phép nhân trên với toàn ma trận các điểm ảnh, trước tiên em sẽ tạo ra một ma trận 3×3 với các giá trị từ phép toán ở trên (với các cột là Red, Green và Blue). Sau đó, em sẽ lấy ma trận ban đầu nhân chuyển vị với ma trận vừa tạo thì ta sẽ có được kết quả.

2.8 Làm mờ ảnh

Về ý tưởng để cài đặt chức năng làm mờ ảnh, ban đầu hàm sẽ nhận giá trị đầu vào (r), giá trị này sẽ được dùng để tạo ma trận kernel có kích thước là $2r + 1$.

Về cách tạo ra ma trận kernel, trước tiên tạo ra ma trận toàn giá trị 1 với kích thước $2r + 1$, sau đó lấy kết quả trên chia với $(2r + 1)^2$.



Hình 3: Hình ảnh mô tả ma trận kernel (nguồn: [Inside code](#))

Trong chương trình, hàm sẽ làm mờ những điểm ảnh nằm ngoài khu vực biên. Ví dụ với $r = 2$, thì những điểm nằm trong viền ảnh gốc có độ dày bằng 2 thì vẫn giữ nguyên giá trị, các điểm ảnh khác sẽ được tiến hành làm mờ. Sau khi có được ma trận kernel, từng điểm ảnh trong ma trận mới sẽ được tính bằng tổng các tích của những điểm ảnh xung quanh với ma trận kernel. Từ đó ta có được ma trận điểm ảnh mới sau khi được làm mờ.

2.9 Làm sắc nét ảnh

Ý tưởng làm sắc nét ảnh gần như tương tự với làm mờ ảnh, tuy nhiên trong trường hợp này, chương trình sẽ tạo ra một kernel cố định có kích thước 3x3 (**Laplacian kernel**)

-1	-1	-1
-1	8	-1
-1	-1	-1

Hình 4: Kernel được sử dụng trong chức năng làm sắc nét ảnh

Sau khi tạo xong kernel, cách tính các điểm ảnh mới tương tự như chức năng làm mờ ảnh, giá trị các điểm ảnh bằng tổng của các tích của từng điểm ảnh ban đầu với ma trận kernel.

2.10 Cắt ảnh theo kích thước

Trong chương trình, chức năng cắt ảnh theo kích thước sẽ là cắt ảnh từ vị trí trung tâm với kích thước cho trước. Trong chương trình, kích thước mặc định để cắt sẽ là 250.

Trước tiên, ta phải tìm được vị trí của 4 đường thẳng giới hạn ảnh. Để tìm 4 đường trên, trước tiên chương trình sẽ tìm điểm ảnh có tọa độ trung tâm. Điểm ảnh trung tâm được tính bằng cách chiều cao và chiều rộng sẽ bằng một nửa của độ cao và độ rộng của ảnh. Từ đó, ta sẽ tính được như sau:

- Phương trình đường giới hạn trên và dưới sẽ được tính bằng cách lấy chiều cao của điểm ảnh trung tâm trừ và cộng với 250.
- Phương trình đường giới hạn trái và phải sẽ bằng chiều rộng của điểm ảnh trừ và cộng với 250.

Sau khi tìm được các đường thẳng trên, ta sẽ tạo một ma trận mới lấy các điểm ảnh thuộc vùng giới hạn bởi 4 đường trên.

2.11 Cắt ảnh theo khung tròn

Để cắt được ảnh theo khung tròn, ý tưởng chính là tìm được các điểm nằm trong phương trình đường tròn có tâm là điểm ảnh trung tâm.

Tâm của đường tròn sẽ là điểm ảnh trung tâm, do đó cách tìm tọa độ của điểm ảnh này tương tự như trong chức năng cắt ảnh theo kích thước. Bán kính của đường tròn sẽ tính bằng cách lấy giá trị nhỏ nhất giữa một nửa độ rộng và một nửa độ cao của ảnh.

Từ đó, áp dụng phương trình đường tròn, ta sẽ lấy các điểm ảnh phù hợp.

2.12 Cắt ảnh theo khung elip

Ở chức năng này, chương trình sẽ cắt theo khung với 2 elip chéo nhau. Trước tiên, chúng ta cần biết các khái niệm của phương trình elip. Phương trình elip bao gồm nửa trục lớn (a) và nửa trục bé (b) có phương trình tổng quát:

$$\frac{X^2}{a^2} + \frac{Y^2}{b^2} = 1$$

Trong chương trình, em sẽ mặc định giá trị a sẽ gấp đôi giá trị b . Tuy nhiên theo yêu cầu bài toán, elip trong hình nghiêng với một góc 45 độ, do đó, các giá trị x , y sẽ được cập nhật lại theo công thức như sau:

$$\begin{aligned}x &= (X - x_0) \cos \phi + (Y - y_0) \sin \phi \\y &= -(X - x_0) \sin \phi + (Y - y_0) \cos \phi\end{aligned}$$

Với x_0, y_0 lần lượt là tọa độ của tâm elip và X, Y là tọa độ ban đầu. Do đó, ứng với $\phi = 45$, ta sẽ có được giá trị x, y mới.

$$\begin{aligned}x &= (X - x_0) \frac{\sqrt{2}}{2} + (Y - y_0) \frac{\sqrt{2}}{2} \\y &= -(X - x_0) \frac{\sqrt{2}}{2} + (Y - y_0) \frac{\sqrt{2}}{2}\end{aligned}$$

Từ đó, ta có được hai phương trình elip sau:

$$\begin{aligned}\frac{x^2}{a^2} + \frac{y^2}{b^2} &= \frac{(X - Y)^2}{2a^2} + \frac{(X + Y)^2}{2b^2} = 1 \\ \frac{y^2}{a^2} + \frac{x^2}{b^2} &= \frac{(X + Y)^2}{2a^2} + \frac{(X - Y)^2}{2b^2} = 1\end{aligned}$$

Tuy nhiên, nếu chỉ tạo ra hai elip như vậy thì các elip trên sẽ không đảm bảo việc tiếp xúc với viền của ảnh ban đầu. Do đó để thực hiện yêu cầu trên, ta phải tìm được các giá trị biên tiếp xúc với các viền.

Dưới đây là cách tìm các đường tiếp xúc với elip tại các điểm biên. Ta có phương trình của 1 elip nghiêng với 1 góc 45 độ là:

$$\frac{(X - Y)^2}{2a^2} + \frac{(X + Y)^2}{2b^2} = 1$$

Áp dụng bất đẳng thức Cauchy - Schwarz, ta có:

$$\begin{aligned}\frac{(X - Y)^2}{2a^2} + \frac{(X + Y)^2}{2b^2} &\leq \frac{(X - Y + X + Y)^2}{2a^2 + 2b^2} = \frac{4X^2}{2a^2 + 2b^2} \\ \frac{(X - Y)^2}{2a^2} + \frac{(X + Y)^2}{2b^2} &\leq \frac{(-X + Y + X - Y)^2}{2a^2 + 2b^2} = \frac{4Y^2}{2a^2 + 2b^2}\end{aligned}$$

Từ đó ta sẽ có được:

$$\begin{aligned}1 &\leq \frac{4X^2}{2a^2 + 2b^2} \iff X \leq \sqrt{\frac{a^2 + b^2}{2}} \\ 1 &\leq \frac{4Y^2}{2a^2 + 2b^2} \iff Y \leq \sqrt{\frac{a^2 + b^2}{2}}\end{aligned}$$

Do đó, khi xét dấu bằng xảy ra và thay $a = 2b$, ta sẽ có được biểu thức:

$$a = \sqrt{\frac{2}{5}} \cdot 2X = \sqrt{\frac{2}{5}} \cdot width$$

$$a = \sqrt{\frac{2}{5}} \cdot 2Y = \sqrt{\frac{2}{5}} \cdot height$$

với width và height là độ rộng và chiều cao của bức ảnh

Khi đó, ta sẽ rút ra được biểu thức liên hệ giữa nửa trục lớn với độ rộng và chiều cao của bức ảnh:

$$a = \sqrt{\frac{2}{5}} \cdot \min(height, width)$$

Do đó ta sẽ tìm được độ dài nửa trục lớn và trục bé sao cho 2 elip tiếp xúc với viền của ảnh ban đầu.

2.13 Phóng to, thu nhỏ ảnh

Để thực hiện việc zoom ảnh bằng cách sử dụng NumPy, ý tưởng chính là tạo ra một phiên bản mới của ảnh với kích thước lớn hơn so với ảnh gốc và sao chép các giá trị pixel từ ảnh gốc vào các vị trí tương ứng trong ảnh mới theo hệ số zoom.

Điều này có thể được thực hiện bằng cách sử dụng phép nội suy (interpolation). Trước tiên, ta cần phải có được kích thước của ảnh ban đầu, đồng thời tính luôn kích thước của ảnh zoom bằng cách lấy kích thước ban đầu nhân với hệ số zoom. Trong chương trình sẽ mặc định khi phóng to, hệ số sẽ là 2 và khi thu nhỏ hệ số sẽ là 0.5.

Sau đó, chương trình sẽ thực hiện phép nội suy để tìm các giá trị pixel cho ma trận mới. Các ô tại vị trí i, j trong ma trận mới sẽ là điểm ảnh tại vị trí $i/zoom_factor, j/zoom_factor$ trong ma trận cũ. Khi đó các giá trị của pixel mới sẽ có cùng giá trị với pixel gần nhất trong ảnh gốc.

3 Mô tả các hàm

3.1 Hàm đọc file (*read_img*)

Hàm được sử dụng với mục đích đọc file ảnh thành ma trận hai chiều. Để thực hiện hàm, chương trình sử dụng hàm *open()* của thư viện Image và hàm sẽ trả về ma trận hai chiều sau khi đọc.

3.2 Hàm thể hiện hình ảnh (*show_img*)

Hàm sử dụng để biểu diễn ảnh qua hàm *imshow()* của thư viện *matplotlib.pyplot*, đồng thời hàm có sử dụng *plt.title()* để thể hiện tiêu đề của ảnh.

3.3 Hàm lưu ảnh (*save_img*)

Trước tiên, hàm sẽ thực hiện chuyển đổi ma trận hai chiều thành hình ảnh qua hàm *fromarray()* của thư viện Image. Sau đó, chương trình sẽ lưu ảnh dưới 2 dạng file png.

3.4 Hàm thực hiện thay đổi độ sáng (*changeBrightness*)

Tại hàm *changeBrightness*, hàm sẽ nhận đầu vào là ma trận ảnh hai chiều và giá trị độ sáng được cài đặt định là 50. Sau đó, hàm sẽ thực hiện cộng các giá trị của điểm với giá trị độ sáng và sử dụng hàm *np.clip* để đảm bảo các điểm ảnh có giá trị từ 0 đến 255.

3.5 Hàm thực hiện thay đổi độ tương phản (*changeContrast*)

Hàm sẽ nhận đầu vào là ma trận ảnh hai chiều và mức độ tương phản, đã được cài đặt định là 50. Từ đó, hàm sẽ tính toán giá trị *factor*, là hệ số hiệu chỉnh tương phản. Hàm sẽ sử dụng giá trị các điểm ảnh ban đầu, hệ số hiệu chỉnh tương phản và áp dụng công thức đã được đề cập ở phần ý tưởng, từ đó hàm sẽ trả về ma trận hai chiều sau khi thay đổi.

3.6 Hàm lật ảnh dọc (*flipImageVertically*)

Để ảnh theo chiều dọc mà tiết kiệm được thời gian chạy, hàm sẽ sử dụng phép slicing trong numpy. Cụ thể hơn phép slicing là *[:,-1, :]*, tức là tạo một ma trận mới bằng cách đảo ngược thứ tự các hàng của mảng ban đầu, thứ tự các cột giữ nguyên.

3.7 Hàm lật ảnh ngang (*flipImageHorizontally*)

Tương tự với hàm lật ảnh dọc, hàm lật ảnh ngang cũng sẽ sử dụng phép slicing, cụ thể phép slicing được sử dụng là `[:, ::-1]`. Khi đó, ma trận mới được tạo bằng cách đảo ngược thứ tự các cột, các hàng sẽ giữ nguyên.

3.8 Hàm đổi ảnh sang ảnh xám (*grayScaleImage*)

Trước khi thực hiện đổi ảnh, hàm sẽ tạo ma trận weight với các giá trị red là 0.299, green là 0.587 và blue là 0.114. Sau đó ma trận ảnh xám sẽ là kết quả của phép nhân ma trận giữa ma trận ban đầu và ma trận weight.

3.9 Hàm đổi ảnh sang ảnh sepia (*sepiaScaleImage*)

Trước tiên, hàm sẽ tạo ma trận weight 3x3 với các giá trị red, green, blue giống như em đã đề cập tại phần ý tưởng thực hiện hàm chuyển đổi sang ảnh sepia. Sau đó, ma trận ảnh sepia sẽ là kết quả của phép nhân hai ma trận ban đầu và chuyển vị của ma trận weight vừa tạo (vì khi chuyển vị ma trận weight thì phép nhân mới thực hiện đúng như cách tính toán đã đề cập tại phần ý tưởng).

3.10 Hàm làm mờ ảnh (*blur_image*)

Hàm làm mờ ảnh sẽ nhận giá trị đầu vào là một ma trận hai chiều và một giá trị r , được mặc định bằng 3. Từ giá trị r đó, trước tiên hàm sẽ tính ra được kích thước của ma trận kernel và tạo ma trận kernel bằng cách lấy ma trận toàn giá trị 1 chia với bình phương kích thước của kernel.

Sau khi có được ma trận kernel, hàm sẽ tiến hành quy trình làm mờ ảnh. Trong chương trình các điểm ảnh ở khu viền có độ dài bằng r ô sẽ không bị làm mờ. Các điểm ảnh còn lại sẽ bị làm mờ bằng cách tính tổng của các tích giữa ma trận kernel với các điểm ảnh xung quanh.

Sau khi thực hiện xong thuật toán, chương trình sử dụng hàm `np.clip` để đảm bảo các giá trị ô màu nằm trong khoảng từ 0 đến 255.

3.11 Hàm làm sắc nét ảnh (*sharpen_image*)

Đối với chức năng làm sắc nét ảnh, ma trận kernel sẽ được tạo mặc định bên trong hàm. Thuật toán thực hiện của hàm này tương tự với hàm làm mờ ảnh, tuy nhiên, trong quá trình khởi tạo, ma trận mới được cài đặt ở kiểu dữ liệu float để

tránh trường hợp các giá trị của pixel bị cắt bớt (overflow) hoặc làm tròn xuống 0 (underflow) khi sử dụng các kiểu dữ liệu như *np.uint8*.

3.12 Hàm cắt ảnh ở trung tâm (*cropCenter*)

Để cắt được ảnh ở trung tâm, ta phải tìm được 4 đường giới hạn ảnh. Do đó, hàm sẽ tìm 4 đường trên (top, bottom, left, right) bằng cách dựa vào tọa độ điểm ảnh trung tâm. Sau đó, hàm sẽ duyệt ma trận ảnh ban đầu trong vùng 4 đường trên thì ta sẽ có được ma trận kết quả sau khi cắt.

3.13 Hàm cắt ảnh theo khung tròn (*cropCircular*)

Để tìm đường khung tròn để cắt, trước tiên hàm sẽ tìm tọa độ của điểm ảnh trung tâm và bán kính của đường tròn trước. Chương trình có sử dụng hàm *np.ogrid* với mục đích để tạo các lưới tọa độ, với *y* sẽ chứa các tọa độ của hàng và *x* chứa các tọa độ của cột.

Sau đó hàm sẽ lấy các điểm ảnh nằm trong khu vực của đường tròn, tức sẽ thỏa bất phương trình sau:

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

Với *a*, *b* lần lượt là độ cao và độ rộng của điểm ảnh trung tâm và *R* là bán kính của đường tròn.

Các điểm nằm ngoài khu vực đường tròn sẽ trả về màu đen (tức mang giá trị là 0).

3.14 Hàm cắt ảnh theo khung elip (*cropEllipses*)

Trước khi thực hiện cắt ảnh, hàm trước tiên vẫn sẽ tìm tọa độ điểm ảnh trung tâm và độ cao, độ rộng của ảnh. Sau đó, hàm sẽ khởi tạo giá trị cho nửa trục lớn *a* và nửa trục bé *b* với $a = 2b$. Hàm sẽ gán giá trị cho *a* theo biểu thức đã được chứng minh ở phần ý tưởng.

$$a = \sqrt{\frac{2}{5}} \cdot \min(\text{height}, \text{width})$$

Ngoài ra, hàm có sử dụng *np.ogrid* với mục đích tạo các lưới tọa độ hàng và cột. Sau đó hàm chuyển đổi tọa độ ứng với elip nghiêng 1 góc 45 độ. Từ đó, hàm sẽ tạo ra được 2 vùng thỏa mãn phương trình elip. Và cuối cùng, hàm sẽ lấy những điểm ảnh nằm trong vùng cần tìm, các điểm còn lại sẽ trả về 0.

3.15 Hàm phóng to, thu nhỏ ảnh (*zoom_image*)

Hàm phóng to, thu nhỏ ảnh sẽ nhận giá trị *zoom_factor*. Đây là hệ số phóng, đối với phóng to thì hệ số mặc định là 2 và đối với thu nhỏ thì hệ số là 0.5.

Sau đó, hàm sẽ tạo kích thước mới dựa trên hệ số. Khi đó, ma trận kết quả sẽ được tính bằng phương pháp nội suy, và sẽ trả về ma trận ảnh hai chiều sau khi đã được phóng to hoặc thu nhỏ.

3.16 Hàm thực hiện tất cả chức năng (*do_all*)

Để thực hiện lựa chọn 0 trong hàm main, tức là thực hiện các chức năng cùng một lúc thì hàm *do_all* sẽ đảm nhiệm. Hàm sẽ tổng hợp tất cả các chức năng ở các hàm phía trên và hình ảnh kết quả sẽ được lưu lại. Ví dụ với hình ảnh có tên "cat.png", khi được thực hiện bởi chức năng làm mờ thì ảnh sẽ được lưu tên là "cat_blur.png". Hàm sẽ lưu tất cả các ảnh sau khi thực hiện tất cả chức năng.

3.17 Hàm Main

Tại hàm main, chương trình sẽ cài đặt các lựa chọn để người dùng có thể chọn các chức năng mà mình muốn:

- 0: Thực hiện tất cả
- 1: Điều chỉnh độ sáng
- 2: Điều chỉnh độ tương phản
- 3: Lật ảnh (dọc / ngang)
- 4: Chuyển đổi thành ảnh xám / ảnh sepia
- 5: Làm mờ / làm sắc nét ảnh
- 6: Cắt ảnh trung tâm
- 7: Cắt ảnh theo khung tròn / elip
- 8: Phóng to, thu nhỏ ảnh

Trong đó, đối với lựa chọn 3, 4, 5, 7 sẽ còn cho người dùng lựa chọn bên trong để thực hiện chức năng cụ thể.





4 Bảng đánh giá mức độ hoàn thành và hình ảnh kết quả





Trong đề án lần này, hình ảnh gốc mà em sử dụng là ảnh sau (kích thước 564 x 564):




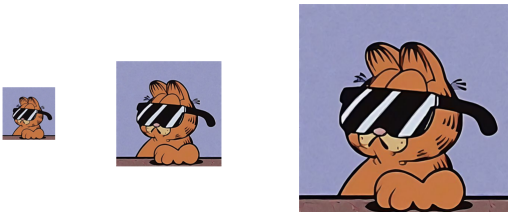


Hình 3: Hình ảnh gốc sử dụng (nguồn: [Pinterest](#))

Dưới đây là bảng đánh giá mức độ hoàn thành và kết quả của từng chức năng:

STT	Chức năng/Hàm	Mức độ hoàn thành	Ảnh kết quả
1	Thay đổi độ sáng	100%	
2	Thay đổi độ tương phản	100%	
3.1	Lật ảnh ngang	100%	
3.2	Lật ảnh dọc	100%	

STT	Chức năng/Hàm	Mức độ hoàn thành	Ảnh kết quả
4.1	RGB thành ảnh xám	100%	
4.2	RGB thành ảnh sepia	100%	
5.1	Làm mờ ảnh	100%	
5.2	Làm sắc nét ảnh	100%	

STT	Chức năng/Hàm	Mức độ hoàn thành	Ảnh kết quả
6	Cắt ảnh theo kích thước	100%	
7.1	Cắt ảnh theo khung tròn	100%	
7.2	Cắt ảnh theo khung elip	100%	
8	Hàm main	100%	
9	Phóng to/thu nhỏ 2x	100%	

Bảng 1: Bảng đánh giá và ảnh kết quả

5 Tài liệu tham khảo

[1]. tutorialspoint, GrayScale to RGB Conversion

https://www.tutorialspoint.com/dip/grayscale_to_rgb_conversion.htm, truy cập vào ngày 20/07/2024.

[2]. dyclassroom, How to convert a color image into sepia image

<https://dyclassroom.com/image-processing-project/how-to-convert-a-color-image-into-sepia-image>, truy cập vào ngày 20/07/2024.

[3]. Inside code, The algorithm to blur images (box blur)

<https://www.youtube.com/watch?v=4Eh0y3LHTNU>, truy cập vào ngày 22/07/2024.

[4]. tutorialspoint, Zooming Methods

https://www.tutorialspoint.com/dip/zooming_methods.html, truy cập vào ngày 30/07/2024.

[5]. NumPy, numpy.ogrid

<https://numpy.org/doc/stable/reference/generated/numpy.ogrid.html>, truy cập vào ngày 25/07/2024.

[6]. Fracis G.Loch, Image Processing Algorithms Part 5: Contrast Adjustment,

<https://www.dfstudios.co.uk/crypt/Online/56/imgproc5.html>, truy cập vào ngày 23/07/2024.

[7]. Laplacian, Laplacian of Gaussian,

<https://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm>, truy cập vào ngày 28/07/2024.