

ĐẠI HỌC QUỐC GIA
THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN



MTH00057 - Toán ứng dụng và thống kê cho Công
nghệ thông tin

BÁO CÁO ĐỒ ÁN 2
Image Processing

Họ tên
Bùi Minh Duy

MSSV
23127040

Giảng viên hướng dẫn
Nguyễn Văn Quang Huy
Trần Hà Sơn
Nguyễn Đình Thúc
Nguyễn Ngọc Toàn

Ngày 27 tháng 7 năm 2025

Mục lục

1	Thông tin cá nhân	2
2	Các chức năng đã hoàn thành	2
3	Ý tưởng thực hiện và mô tả các hàm chức năng	3
3.1	Điều chỉnh độ sáng (Hàm <code>adjust_brightness</code>)	3
3.2	Điều chỉnh độ tương phản (Hàm <code>adjust_contrast</code>)	3
3.3	Lật ảnh (Hàm <code>flip_image</code>)	4
3.4	Chuyển ảnh RGB sang màu xám (Hàm <code>rgb_to_grayscale</code>)	4
3.5	Chuyển ảnh RGB sang màu sepia (Hàm <code>rgb_to_sepia</code>)	5
3.6	Làm mờ và sắc nét ảnh (Hàm <code>apply_filter</code>)	5
3.7	Cắt ảnh theo khung hình vuông ở trung tâm (Hàm <code>center_crop</code>)	7
3.8	Cắt ảnh theo khung hình tròn (Hàm <code>circular_crop</code>)	7
3.9	Cắt ảnh theo khung hai ellipse chéo nhau (Hàm <code>double_ellipse_crop</code>) . .	8
4	Kết quả thực nghiệm	11
4.1	Điều chỉnh độ sáng	11
4.2	Điều chỉnh độ tương phản	12
4.3	Lật ảnh	12
4.4	Chuyển đổi màu sắc	13
4.5	Làm mờ và sắc nét ảnh	13
4.6	Cắt ảnh theo khung hình vuông ở trung tâm	14
4.7	Cắt ảnh theo khung hình tròn	14
4.8	Cắt ảnh theo khung hai hình ellipse chéo nhau	15
4.9	Thời gian thực thi các chức năng	15
5	Acknowledgement	16

1 Thông tin cá nhân

- Họ và tên: Bùi Minh Duy
- MSSV: 23127040
- Lớp: 23CLC01

2 Các chức năng đã hoàn thành

Chức năng	Mức độ hoàn thành
Điều chỉnh độ sáng của ảnh	100%
Điều chỉnh độ tương phản của ảnh	100%
Chuyển ảnh RGB sang màu xám (grayscale)	100%
Chuyển ảnh RGB sang màu sepia	100%
Làm mờ ảnh	100%
Làm nét ảnh	100%
Lật ảnh theo chiều ngang và dọc	100%
Cắt ảnh theo khung hình vuông	100%
Cắt ảnh theo khung hình tròn	100%
Cắt ảnh theo khung 2 hình elip chéo nhau	100%

Bảng 1: Danh sách các chức năng và mức độ hoàn thành

3 Ý tưởng thực hiện và mô tả các hàm chức năng

3.1 Điều chỉnh độ sáng (Hàm `adjust_brightness`)

Ý tưởng thực hiện

Hàm `adjust_brightness` điều chỉnh độ sáng của ảnh bằng cách cộng trực tiếp một giá trị độ sáng (`value`) vào từng điểm ảnh. Đây là một cách điều chỉnh tuyến tính đơn giản nhưng hiệu quả để làm thay đổi độ sáng tổng thể của ảnh.

- Nếu `value > 0`, ảnh sẽ sáng hơn.
- Nếu `value < 0`, ảnh sẽ tối đi.
- Nếu `value = 0`, ảnh giữ nguyên độ sáng.

Mô tả chi tiết

Hàm nhận vào ảnh RGB đầu vào dưới dạng NumPy array và một giá trị số thực `value` đại diện cho mức độ điều chỉnh độ sáng.

Đầu tiên, ảnh được ép kiểu sang `float32` để đảm bảo phép cộng được thực hiện chính xác mà không bị tràn số. Sau đó, giá trị độ sáng được cộng trực tiếp vào từng phần tử (tức là từng kênh màu tại mỗi điểm ảnh) trong ảnh.

Kết quả thu được có thể chứa các giá trị vượt ngoài khoảng [0, 255], nên cần dùng hàm `clip` để giới hạn toàn bộ ảnh về trong khoảng hợp lệ này.

Cuối cùng, ảnh được chuyển ngược lại về kiểu `uint8` để sẵn sàng hiển thị hoặc lưu trữ.

3.2 Điều chỉnh độ tương phản (Hàm `adjust_contrast`)

Ý tưởng thực hiện

Hàm `adjust_contrast` cho phép điều chỉnh độ tương phản của một ảnh RGB bằng cách làm giãn hoặc co khoảng cách giữa các mức cường độ sáng so với mức sáng trung bình thực tế của ảnh. Việc thay đổi độ tương phản được điều khiển bằng hệ số `alpha`:

- Nếu `alpha > 1`, độ tương phản tăng: điểm ảnh sáng trở nên sáng hơn, điểm tối trở nên tối hơn.
- Nếu `0 < alpha < 1`, độ tương phản giảm: ảnh trở nên mờ nhạt hơn.
- Nếu `alpha = 1`, ảnh giữ nguyên.

Mô tả chi tiết

Hàm nhận đầu vào là một ảnh RGB dạng NumPy array và một hệ số điều chỉnh tương phản `alpha` (kiểu `float`).

Để điều chỉnh độ tương phản, mỗi giá trị pixel được dời gốc về mức sáng trung bình thực tế μ của toàn bộ ảnh, sau đó nhân với `alpha`, rồi cộng lại μ để đưa về không gian

cường độ ban đầu. Cụ thể, mỗi pixel được tính theo công thức:

$$\text{output}(x, y) = \alpha \cdot (\text{input}(x, y) - \mu) + \mu$$

Trong đó $\mu = \text{mean}(\text{input})$ là giá trị trung bình của toàn bộ ảnh (bao gồm cả ba kênh R, G, B).

3.3 Lật ảnh (Hàm `flip_image`)

Ý tưởng thực hiện

Hàm `flip_image` thực hiện phép đổi xứng ảnh theo chiều ngang hoặc chiều dọc. Tuỳ theo tham số đầu vào, hàm sẽ hoán đổi các hàng hoặc các cột của ma trận ảnh để tạo hiệu ứng lật gương tương ứng.

Mô tả chi tiết

Hàm nhận đầu vào là một ảnh RGB dưới dạng NumPy array và một chuỗi ký tự `direction` cho biết hướng lật mong muốn. Tham số `direction` có thể nhận một trong hai giá trị: “`horizontal`” (lật ngang) hoặc “`vertical`” (lật dọc).

Dựa vào giá trị của tham số này, ảnh sẽ được lật bằng cách sử dụng hàm `np.flip` của thư viện NumPy:

- Nếu `direction` là “`horizontal`”, ảnh được lật theo chiều ngang, tức là đảo ngược thứ tự các cột (trái – phải).
- Nếu `direction` là “`vertical`”, ảnh được lật theo chiều dọc, tức là đảo ngược thứ tự các hàng (trên – dưới).

Quá trình này không làm thay đổi nội dung pixel mà chỉ hoán đổi vị trí của chúng theo trực tương ứng.

3.4 Chuyển ảnh RGB sang màu xám (Hàm `rgb_to_grayscale`)

Ý tưởng thực hiện

Hàm `rgb_to_grayscale` chuyển đổi ảnh RGB màu sang ảnh xám (grayscale) bằng cách sử dụng các hệ số trọng số độ chói tiêu chuẩn. Mỗi điểm ảnh xám được tính bằng tổ hợp tuyến tính của ba kênh màu: đỏ (red), lục (green), và lam (blue), phản ánh mức độ sáng mà mắt người cảm nhận được.

Mô tả chi tiết

Hàm nhận đầu vào là một ảnh màu RGB dưới dạng NumPy array có kích thước $(H, W, 3)$, trong đó ba kênh tương ứng với màu Đỏ (R), Lục (G) và Lam (B).

Để chuyển đổi ảnh màu sang ảnh xám, hàm sử dụng công thức chuyển đổi độ chói theo tiêu chuẩn của ngành xử lý ảnh. Mỗi pixel được tính theo tổ hợp tuyến tính của ba kênh

màu như sau:

$$\text{gray}(x, y) = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

Công thức này phản ánh độ nhạy của mắt người với từng kênh màu: mắt nhạy với màu lục nhất, tiếp theo là đỏ và ít nhạy nhất với lam. Kết quả trả về là một ảnh xám 2D chỉ gồm một kênh độ sáng duy nhất.

3.5 Chuyển ảnh RGB sang màu sepia (Hàm `rgb_to_sepia`)

Ý tưởng thực hiện

Hàm `rgb_to_sepia` áp dụng bộ lọc màu sepia lên ảnh RGB để tạo hiệu ứng cổ điển. Hiệu ứng sepia được tạo ra bằng cách nhân mỗi điểm ảnh RGB với một ma trận biến đổi đặc trưng, chuyển đổi các kênh màu theo tỷ lệ nhằm tạo ra tông màu nâu ấm.

Mô tả chi tiết

Hàm nhận đầu vào là một ảnh RGB dưới dạng NumPy array với shape $(H, W, 3)$. Mỗi điểm ảnh sẽ được biến đổi theo hiệu ứng màu sepia bằng cách sử dụng một ma trận lọc sepia 3×3 có dạng:

$$\text{SepiaFilter} = \begin{bmatrix} 0.393 & 0.769 & 0.189 \\ 0.349 & 0.686 & 0.168 \\ 0.272 & 0.534 & 0.131 \end{bmatrix}$$

Mỗi dòng trong ma trận tương ứng với cách tính mới cho từng kênh màu đầu ra R', G', B' , dựa trên tổ hợp tuyến tính của các giá trị R, G, B ban đầu.

Quá trình chuyển đổi áp dụng phép nhân ma trận tại mỗi điểm ảnh theo công thức:

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \text{SepiaFilter} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

3.6 Làm mờ và sắc nét ảnh (Hàm `apply_filter`)

Ý tưởng thực hiện

Hàm `apply_filter` thực hiện áp dụng bộ lọc tích chập (convolution filter) lên ảnh RGB bằng cách quét qua từng điểm ảnh và tính toán giá trị mới dựa trên một kernel (ma trận lọc). Hai loại kernel được hỗ trợ:

- **blur:** Làm mờ ảnh bằng kernel trung bình kích thước 5×5 .
- **sharpen:** Làm nét ảnh bằng kernel Laplace 3×3 đã được điều chỉnh.

Việc lựa chọn bộ lọc làm mờ 5×5 thay vì 3×3 giúp làm mờ ảnh rõ rệt hơn và triệt tiêu nhiễu tốt hơn. So với Gaussian blur 5×5 , bộ lọc trung bình (box blur) có ưu điểm:

- Tính toán đơn giản hơn (không cần nhân với hệ số Gauss), do đó nhanh hơn.
- Vẫn mang lại hiệu ứng làm mờ đủ mạnh trong bối cảnh xử lý ảnh cơ bản.
- Giảm độ phức tạp trong khi vẫn đảm bảo hiệu quả trực quan.

Mô tả chi tiết

Đầu tiên, hàm kiểm tra loại kernel được chỉ định là `blur` hay `sharpen`:

- Nếu kernel là `blur`, một ma trận kernel kích thước 5×5 được khởi tạo, trong đó tất cả các phần tử đều mang giá trị bằng $\frac{1}{25}$. Ma trận này đóng vai trò làm mờ ảnh bằng cách tính trung bình cộng các giá trị trong vùng lân cận của mỗi điểm ảnh. Mỗi điểm ảnh đầu ra sẽ là trung bình của 25 điểm ảnh xung quanh trong ảnh gốc, giúp giảm nhiễu và làm mịn ảnh.
- Nếu kernel là `sharpen`, một ma trận kernel 3×3 được sử dụng với giá trị:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Kernel này làm tăng cường sự khác biệt giữa điểm ảnh trung tâm và các điểm ảnh lân cận, từ đó tăng độ sắc nét của ảnh. Trọng số lớn ở giữa (giá trị 5) kết hợp với các giá trị âm xung quanh giúp nhấn mạnh chi tiết.

Sau khi xác định kernel phù hợp, ảnh đầu vào được xử lý padding để đảm bảo kích thước đầu ra không thay đổi sau khi áp dụng bộ lọc:

- Gọi k_h và k_w lần lượt là chiều cao và chiều rộng của kernel. Khi đó, số lượng pixel cần padding theo chiều cao và chiều rộng được tính bằng:

$$pad_h = \left\lfloor \frac{k_h}{2} \right\rfloor, \quad pad_w = \left\lfloor \frac{k_w}{2} \right\rfloor$$

- Hàm `np.pad` được dùng để mở rộng ảnh đầu vào. Ảnh được chèn thêm các dải biên với giá trị lặp lại từ mép ảnh (chế độ `edge`), nhằm đảm bảo rằng các vùng biên vẫn có đủ lân cận để áp dụng kernel mà không gây lỗi truy xuất ngoài mảng.

Sau bước padding, hàm thực hiện thao tác tích chập (convolution) theo từng điểm ảnh như sau:

- Duyệt qua từng pixel tại vị trí (i, j) trong ảnh gốc (chưa padding), và từng kênh màu (R, G, B).
- Tại mỗi vị trí, trích xuất một vùng con từ ảnh đã được padding, với kích thước đúng bằng kích thước kernel, bắt đầu từ vị trí tương ứng với (i, j) .
- Vùng con này sau đó được nhân từng phần tử với kernel tương ứng, sau đó tính tổng toàn bộ các tích này để ra giá trị điểm ảnh mới tại (i, j) cho từng kênh màu.
- Kết quả này được gán vào ma trận ảnh đầu ra tại vị trí tương ứng.

Quá trình trên được lặp lại cho toàn bộ ảnh và tất cả các kênh màu, cho ra ảnh đã được xử lý bằng bộ lọc làm mờ hoặc làm nét theo yêu cầu đầu vào.

3.7 Cắt ảnh theo khung hình vuông ở trung tâm (Hàm center_crop)

Ý tưởng thực hiện

Hàm `center_crop` cắt một vùng vuông có kích thước xác định từ chính giữa ảnh đầu vào. Vị trí bắt đầu cắt được tính dựa trên khoảng cách từ tâm ảnh đến các cạnh, đảm bảo vùng trích xuất được căn giữa một cách đối xứng.

Mô tả chi tiết

- Ảnh đầu vào là một ảnh màu RGB hình vuông, được biểu diễn dưới dạng mảng NumPy với shape là $(H, W, 3)$, trong đó H là chiều cao cũng như chiều rộng của ảnh.
- Đầu vào thứ hai là một số nguyên S biểu diễn kích thước cạnh của vùng ảnh vuông cần cắt ra từ trung tâm ảnh gốc.
- Do ảnh là hình vuông, ta chỉ cần tính một giá trị để xác định vị trí bắt đầu cắt theo cả chiều cao và chiều rộng:

$$\text{start} = \left\lfloor \frac{H - S}{2} \right\rfloor$$

Giá trị này đảm bảo vùng ảnh cần cắt nằm chính giữa ảnh gốc.

- Sau khi có chỉ số bắt đầu, vùng ảnh kết quả được trích xuất bằng cách lấy các hàng và cột từ chỉ số `start` đến `start + S`, cụ thể:

$$\text{img_cropped} = \text{img}[\text{start} : \text{start} + S, \text{start} : \text{start} + S]$$

Vùng ảnh thu được có kích thước $S \times S$, vẫn giữ nguyên 3 kênh màu RGB.

3.8 Cắt ảnh theo khung hình tròn (Hàm circular_crop)

Ý tưởng thực hiện

Hàm `circular_crop` tạo mặt nạ là một hình tròn nội tiếp ảnh vuông đầu vào, với tâm và bán kính xác định theo kích thước ảnh. Ảnh đầu ra chỉ giữ lại các điểm ảnh nằm trong vòng tròn này, các điểm ngoài vùng được gán màu đen.

Mô tả chi tiết

- Ảnh đầu vào là một ảnh màu RGB hình vuông có shape $(H, W, 3)$, được biểu diễn dưới dạng mảng NumPy.
- Đầu tiên, lấy chiều cao và chiều rộng của ảnh (cùng bằng H) và sinh ra hai ma trận chỉ số tọa độ X và Y lần lượt đại diện cho trực hoành (cột) và trực tung (hàng) trên toàn bộ ảnh:

$$Y, X = \text{np.ogrid}[0 : H, 0 : W]$$

Đây là hai mảng giúp xác định vị trí của từng pixel trong ảnh.

- Tính toán tâm của ảnh bằng:

$$\text{center} = \left\lfloor \frac{H}{2} \right\rfloor$$

và lấy bán kính bằng với khoảng cách từ tâm đến mép ảnh:

$$\text{radius} = \text{center}$$

Đây chính là bán kính của hình tròn nội tiếp trong ảnh vuông.

- Sử dụng công thức của hình tròn để tạo một mặt nạ nhị phân `mask`, đánh dấu các điểm ảnh nào nằm bên trong hình tròn:

$$\text{mask} = (X - \text{center})^2 + (Y - \text{center})^2 \leq \text{radius}^2$$

- Khởi tạo một mảng ảnh mới cùng kích thước với ảnh gốc và có toàn bộ giá trị bằng 0 (tức là màu đen):

$$\text{output} = \text{np.zeros_like}(\text{img})$$

- Dựa vào mặt nạ, sao chép các pixel thuộc vùng hình tròn từ ảnh gốc sang ảnh đầu ra:

$$\text{output}[\text{mask}] = \text{img}[\text{mask}]$$

- Kết quả trả về là ảnh mới chỉ hiển thị vùng hình tròn nằm giữa ảnh gốc, còn các vùng nằm ngoài hình tròn được đặt thành màu đen.

3.9 Cắt ảnh theo khung hai ellipse chéo nhau (Hàm `double_ellipse_crop`)

Ý tưởng thực hiện

Hàm `double_ellipse_crop` tạo mặt nạ gồm hai ellipse đồng tâm, xoay lần lượt $+45^\circ$ và -45° , sao cho mỗi ellipse đều được nội tiếp trong ảnh vuông. Tỷ lệ giữa hai bán trục ellipse có thể điều chỉnh được thông qua tham số `ratio` với $0 < \text{ratio} < 1$. Kết quả là ảnh chỉ giữ lại phần hợp giữa hai ellipse này, còn lại được tô màu đen.

Mô tả chi tiết

- Ảnh đầu vào là một ảnh RGB hình vuông có kích thước $(H, W, 3)$, biểu diễn bằng mảng NumPy.
- Xác định kích thước và tọa độ tâm ảnh:

$$H = W = \text{img_array.shape}[0], \quad C_x = C_y = \left\lfloor \frac{H}{2} \right\rfloor$$

- Sinh lưới tọa độ X, Y biểu diễn vị trí từng điểm ảnh:

$$Y, X = \text{np.ogrid}[0 : H, 0 : W]$$

- Bán trục lớn a và bán trục nhỏ b được tính theo công thức:

$$a = \frac{H}{\sqrt{2(1 + \text{ratio}^2)}}, \quad b = a \cdot \text{ratio}$$

- Định nghĩa ellipse đầu tiên bằng cách xoay ảnh một góc $+45^\circ$. Tính tọa độ đã xoay:

$$\begin{aligned} x_1 &= (X - C_x) \cdot \cos(45^\circ) + (Y - C_y) \cdot \sin(45^\circ) \\ y_1 &= -(X - C_x) \cdot \sin(45^\circ) + (Y - C_y) \cdot \cos(45^\circ) \end{aligned}$$

Sau đó tạo mặt nạ:

$$\text{mask}_1 = \left(\frac{x_1}{a}\right)^2 + \left(\frac{y_1}{b}\right)^2 \leq 1$$

- Tương tự, ellipse thứ hai được xoay góc -45° :

$$\begin{aligned} x_2 &= (X - C_x) \cdot \cos(-45^\circ) + (Y - C_y) \cdot \sin(-45^\circ) \\ y_2 &= -(X - C_x) \cdot \sin(-45^\circ) + (Y - C_y) \cdot \cos(-45^\circ) \end{aligned}$$

$$\text{mask}_2 = \left(\frac{x_2}{a}\right)^2 + \left(\frac{y_2}{b}\right)^2 \leq 1$$

- Kết hợp hai mặt nạ để tạo vùng hợp:

$$\text{mask} = \text{mask}_1 \vee \text{mask}_2$$

- Khởi tạo ảnh đầu ra toàn màu đen:

$$\text{output} = \text{np.zeros_like}(\text{img_array})$$

- Sao chép các điểm ảnh từ ảnh gốc theo mặt nạ:

$$\text{output}[\text{mask}] = \text{img_array}[\text{mask}]$$

- Kết quả là ảnh chỉ giữ lại phần hình học là hợp của hai ellipse xoay đối xứng.

Giải thích công thức tính a và b

Để mỗi ellipse xoay 45° được nội tiếp ảnh vuông kích thước $H \times H$, ta cần đảm bảo rằng: hình chữ nhật bao ngoài của ellipse sau khi xoay phải vừa khít với ảnh.

Giả sử ellipse có bán trục lớn a , bán trục nhỏ b , trước khi xoay ellipse nằm dọc trục tung và ngang trục hoành.

Khi xoay ellipse một góc $\theta = 45^\circ$, hình chiếu (projection) của ellipse theo hai trục x và y sẽ có chiều dài lần lượt là:

$$L_x = a \cdot |\cos \theta| + b \cdot |\sin \theta|, \quad L_y = a \cdot |\sin \theta| + b \cdot |\cos \theta|$$

Với $\theta = 45^\circ$, ta có $\cos \theta = \sin \theta = \frac{\sqrt{2}}{2}$, nên:

$$L_x = L_y = \frac{\sqrt{2}}{2}(a + b)$$

Để ellipse không vượt ra ngoài khung ảnh vuông có cạnh H , ta yêu cầu:

$$\frac{\sqrt{2}}{2}(a + b) \leq \frac{H}{2}$$

(Vì tâm ellipse nằm ở giữa ảnh, nên biên trái và phải mỗi bên chỉ có $\frac{H}{2}$ để chứa ellipse)

Giải bất đẳng thức:

$$a + b \leq \frac{H}{\sqrt{2}}$$

Giả sử ta đặt tỉ lệ giữa hai bán trục là $\text{ratio} = \frac{b}{a} \Rightarrow b = a \cdot \text{ratio}$, ta thê vào:

$$a + a \cdot \text{ratio} \leq \frac{H}{\sqrt{2}} \Rightarrow a(1 + \text{ratio}) \leq \frac{H}{\sqrt{2}}$$

Tuy nhiên, để công thức tổng quát hơn và chính xác khi ellipse xoay bất kỳ góc, ta dùng bình phương tổng:

$$L_x^2 + L_y^2 = (a^2 + b^2)$$

\Rightarrow Chiều dài đường chéo hình chữ nhật chứa ellipse (sau xoay) = $\sqrt{a^2 + b^2}$

Yêu cầu ellipse nằm gọn trong ảnh vuông có đường chéo là H , ta cần:

$$\sqrt{a^2 + b^2} \leq \frac{H}{\sqrt{2}} \Rightarrow a^2 + b^2 \leq \left(\frac{H}{\sqrt{2}}\right)^2 = \frac{H^2}{2}$$

Thay $b = a \cdot \text{ratio}$ vào:

$$\begin{aligned} a^2 + a^2 \cdot \text{ratio}^2 &\leq \frac{H^2}{2} \\ \Rightarrow a^2(1 + \text{ratio}^2) &\leq \frac{H^2}{2} \\ \Rightarrow a^2 &\leq \frac{H^2}{2(1 + \text{ratio}^2)} \\ \Rightarrow a &\leq \frac{H}{\sqrt{2(1 + \text{ratio}^2)}} \end{aligned}$$

Vì ta muốn dùng ellipse lớn nhất nội tiếp ảnh, nên chọn dấu bằng:

$$a = \frac{H}{\sqrt{2(1 + \text{ratio}^2)}}, \quad b = a \cdot \text{ratio}$$

Điều này đảm bảo: **dù ellipse xoay $\pm 45^\circ$, vẫn luôn nằm trong ảnh.**

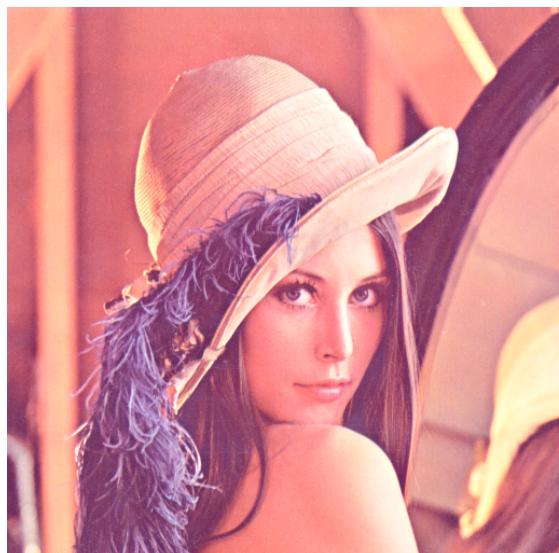
4 Kết quả thực nghiệm

Trong phần này, ta trình bày kết quả thực hiện các chức năng xử lý ảnh trên ảnh mẫu Lena, với kích thước 512×512 pixels. Các chức năng xử lý ảnh bao gồm điều chỉnh độ sáng, độ tương phản, lật ảnh, chuyển đổi màu sắc, làm mờ/sắc nét ảnh, và cắt ảnh. Mỗi chức năng sẽ được trình bày kèm theo hình ảnh kết quả.



Hình 1: Ảnh mẫu Lena

4.1 Điều chỉnh độ sáng



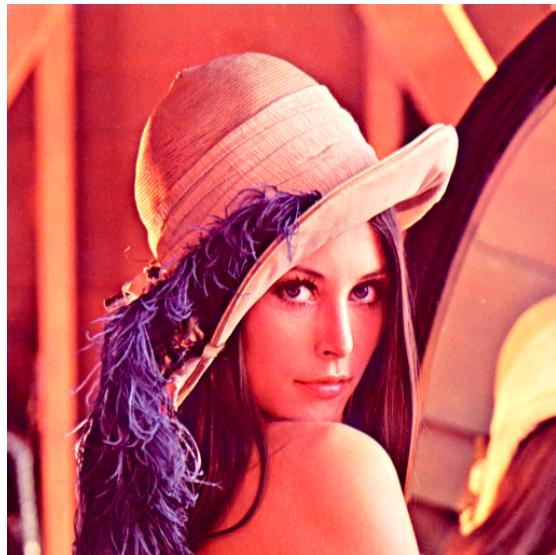
(a) Tăng sáng (+40)



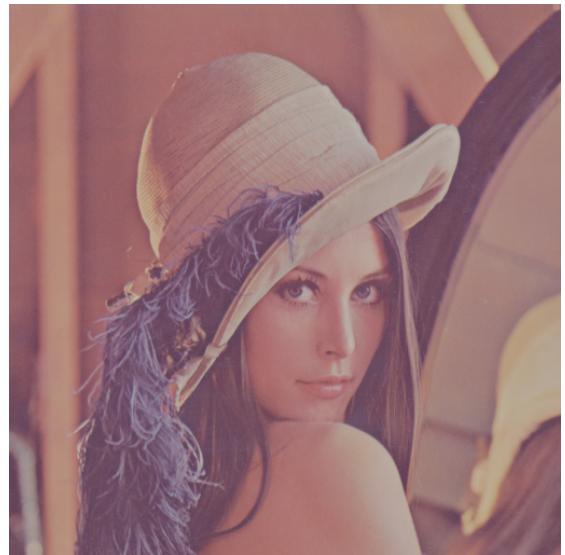
(b) Giảm sáng (-40)

Hình 2: Ảnh sau khi điều chỉnh độ sáng

4.2 Điều chỉnh độ tương phản



(a) Tăng tương phản (1.5)



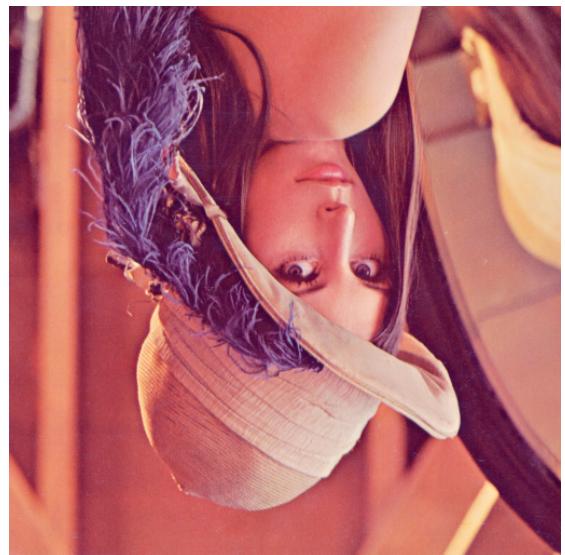
(b) Giảm tương phản (0.5)

Hình 3: Ảnh sau khi điều chỉnh độ tương phản

4.3 Lật ảnh



(a) Lật ngang



(b) Lật dọc

Hình 4: Kết quả lật ảnh

4.4 Chuyển đổi màu sắc



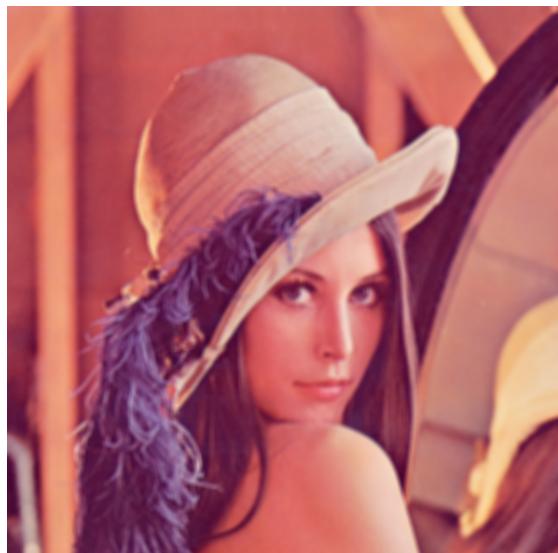
(a) Ảnh xám



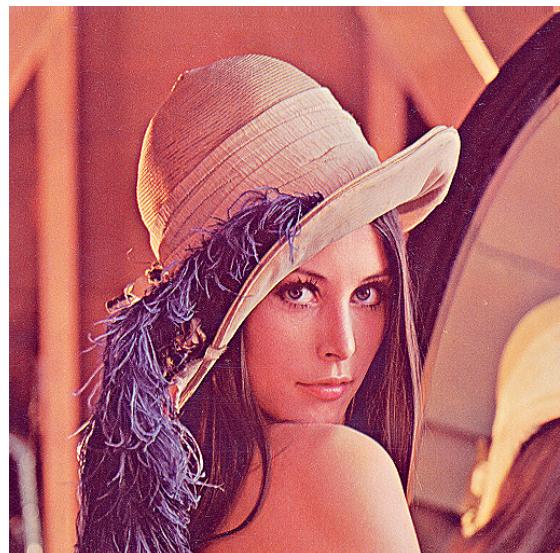
(b) Ảnh sepia

Hình 5: Ảnh sau khi chuyển đổi màu sắc

4.5 Làm mờ và sắc nét ảnh



(a) Làm mờ (Blur)



(b) Làm sắc nét (Sharpen)

Hình 6: Ảnh sau khi áp dụng các bộ lọc

4.6 Cắt ảnh theo khung hình vuông ở trung tâm



Hình 7: Ảnh cắt ở trung tâm (size = 256×256)

4.7 Cắt ảnh theo khung hình tròn



Hình 8: Ảnh cắt theo khung hình tròn

4.8 Cắt ảnh theo khung hai hình ellipse chéo nhau



Hình 9: Ảnh cắt theo khung 2 hình ellipse chéo nhau (ratio = 0.6)

4.9 Thời gian thực thi các chức năng

Bảng 2: Thời gian thực thi của từng chức năng xử lý ảnh

Chức năng	Thời gian
Điều chỉnh độ sáng	1.8ms
Điều chỉnh tương phản	4.4ms
Lật ảnh ngang	< 1ms
Lật ảnh dọc	< 1ms
Chuyển ảnh RGB sang màu xám	4.3ms
Chuyển ảnh RGB sang màu sepia	7.5ms
Làm mờ ảnh	2760.7ms
Làm sắc nét ảnh	2635.1ms
Cắt ảnh theo khung hình vuông ở trung tâm	< 1ms
Cắt ảnh theo khung hình tròn	6.9ms
Cắt ảnh theo khung hai hình ellipse chéo nhau	15ms

5 Acknowledgement

Trong quá trình thực hiện báo cáo và xây dựng hệ thống xử lý ảnh, ChatGPT đã hỗ trợ sinh viên:

- Đề xuất và soạn thảo template báo cáo LaTeX một cách rõ ràng và nhất quán.
- Viết `docstring` cho các hàm xử lý ảnh, giúp chuẩn hóa tài liệu và dễ dàng bảo trì mã nguồn.
- So sánh các loại kernel làm mờ phổ biến (Gaussian 3×3 , Gaussian 5×5 , Box Blur $5 \times 5, \dots$) và đề xuất lựa chọn kernel phù hợp giữa hiệu quả làm mờ và độ phức tạp tính toán.
- Đề xuất ý tưởng và chứng minh công thức tính hai bán kính của elip cho tính năng cắt ảnh theo khung 2 hình elip chéo nhau (`double_ellipse_crop`).
- Hỗ trợ viết hàm đo thời gian thực thi (`benchmark`) cho từng hàm xử lý ảnh, giúp đánh giá hiệu suất và tối ưu hóa.

Tài liệu

- [1] Wikipedia, *Kernel (image processing)*, truy cập ngày 22/07/2025,
[https://en.wikipedia.org/wiki/Kernel_\(image_processing\)](https://en.wikipedia.org/wiki/Kernel_(image_processing))
- [2] dyclassroom.com, *How to convert a color image into sepia image*, truy cập ngày 20/07/2025,
<https://dyclassroom.com/image-processing-project/how-to-convert-a-color-image-into-sepia-image>
- [3] MathWorks, *Convert RGB image or colormap to grayscale*, truy cập ngày 20/07/2025,
<https://www.mathworks.com/help/matlab/ref/rgb2gray.html>
- [4] Gonzalez, R. C. & Woods, R. E., *Digital Image Processing*, 4th Edition, Chương 3: Intensity Transformations and Spatial Filtering, truy cập ngày 18/07/2025,
https://www.c172.org/090imagePLib/books/Gonzales_Woods-Digital.Image.Processing.4th.Edition.pdf