

**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 02: IMAGE PROCESSING**

**Môn học:** Toán ứng dụng và thống kê  
cho Công nghệ thông tin - 22CLC10

**Giáo viên hướng dẫn:**

Thầy Vũ Quốc Hoàng

Thầy Nguyễn Văn Quang Huy

Thầy Nguyễn Ngọc Toàn

Cô Phan Thị Phương Uyên

**Sinh viên thực hiện:**

22127440 Phan Võ Minh Tuệ

# MỤC LỤC

|  |           |
|--|-----------|
| <b>LỜI CẢM ƠN.....</b>                                 | <b>3</b>  |
| <b>TỔNG QUAN .....</b>                                 | <b>4</b>  |
| <b>1. Ý TƯỞNG THỰC HIỆN VÀ MÔ TẢ CÁC HÀM .....</b>     | <b>6</b>  |
| <b>1.1. Ý tưởng thực hiện.....</b>                     | <b>6</b>  |
| <b>1.2. Các thư viện được sử dụng.....</b>             | <b>7</b>  |
| <b>1.3. Mô tả các hàm.....</b>                         | <b>8</b>  |
| 1.3.1. Các hàm hỗ trợ.....                             | 8         |
| 1.3.2. Thay đổi độ sáng.....                           | 9         |
| 1.3.3. Thay đổi độ tương phản.....                     | 9         |
| 1.3.4. Lật ảnh .....                                   | 10        |
| 1.3.5. Chuyển đổi ảnh RGB thành ảnh xám/sepia.....     | 10        |
| 1.3.6. Làm mờ/sắc nét ảnh.....                         | 11        |
| 1.3.7. Cắt ảnh theo kích thước (cắt ở trung tâm) ..... | 11        |
| 1.3.8. Cắt ảnh theo khung.....                         | 12        |
| 1.3.9. Thu nhỏ, phóng to ảnh .....                     | 12        |
| <b>2. HÌNH ẢNH KẾT QUẢ.....</b>                        | <b>14</b> |
| <b>2.1. Môi trường thực thi.....</b>                   | <b>14</b> |
| <b>2.2. Thực thi chương trình.....</b>                 | <b>15</b> |
| 2.2.1. Test case 1 .....                               | 15        |
| 2.2.2. Test case 2 .....                               | 18        |
| 2.2.3. Test case 3 .....                               | 21        |
| <b>3. NHẬN XÉT KẾT QUẢ.....</b>                        | <b>24</b> |
| <b>TỔNG KẾT .....</b>                                  | <b>26</b> |
| <b>TÀI LIỆU THAM KHẢO .....</b>                        | <b>27</b> |

## LỜI CẢM ƠN

Trước hết, tôi xin bày tỏ lòng biết ơn sâu sắc đến quý thầy cô giáo hướng dẫn, đặc biệt là cô Phan Thị Phương Uyên, người đã trao cho tôi cơ hội thực hiện đề án này. Đây quả thực là một chủ đề nghiên cứu hết sức thú vị nhưng cũng mang nhiều ý nghĩa quan trọng đối với tôi trên con đường chinh phục bộ môn Khoa học máy tính sau này.

Tôi cũng muốn bày tỏ lòng biết ơn đối với Trường Đại học Khoa học Tự nhiên, Đại học quốc gia Thành phố Hồ Chí Minh đã hỗ trợ, cung cấp không gian nghiên cứu, học tập cho các sinh viên như chúng tôi. Ngoài ra, tôi cũng muốn gửi lời cảm ơn chân thành đến các bạn sinh viên lớp Toán ứng dụng và thống kê cho Công nghệ thông tin\_22CLC10 đã cung cấp thông tin, chia sẻ kinh nghiệm và tiếp thêm động lực cho tôi trong suốt thời gian thực hiện đề án lần này.

Cuối cùng, tôi không thể không nhắc đến gia đình yêu quý của mình, những người luôn ở bên cạnh, hỗ trợ và tạo điều kiện tốt nhất để tôi có thể tập trung vào việc học tập và nghiên cứu.

Lòng biết ơn của tôi cũng dành cho tất cả những ai đã giúp đỡ, trực tiếp hay gián tiếp, trong quá trình hoàn thành tiểu luận này.

**Phan Võ Minh Tuệ**

# TỔNG QUAN

**Đề án 02 - Image Processing** tập trung vào việc phát triển một vài tính năng cơ bản liên quan đến Computer Vision. Mã nguồn được viết bằng ngôn ngữ lập trình Python kết hợp với các thư viện cơ bản như *NumPy*, *PIL*, *matplotlib* để thực hiện các chức năng như đọc và hiển thị ảnh, áp dụng các thuật toán để xử lý dữ liệu theo yêu cầu.

## Nội dung chi tiết

Báo cáo sẽ trình bày chi tiết về cách thức thực hiện từng chức năng, bao gồm:

- Đọc và hiển thị ảnh
- Chuyển đổi ảnh
- Thay đổi độ sáng cho ảnh
- Thay đổi độ tương phản
- Lật ảnh (ngang - dọc)
- Chuyển đổi ảnh RGB thành ảnh xám/sepia
- Làm mờ/sắc nét ảnh
- Cắt ảnh theo kích thước (cắt ở trung tâm)
- Cắt ảnh theo khung tròn/khung là 2 hình elip chéo nhau
- Thu nhỏ/phóng to ảnh
- Đánh giá kết quả

## Kết quả

Các kết quả thử nghiệm sẽ được minh họa bằng hình ảnh và biểu đồ, cùng với nhận xét về hiệu suất của các hàm được cài đặt nhằm xử lý ảnh cơ bản. Điều này cung cấp cái nhìn tổng quan và sâu sắc về khả năng và hiệu quả của các thuật toán xử lý ảnh được triển khai.

**Bảng đánh giá mức độ hoàn thành**

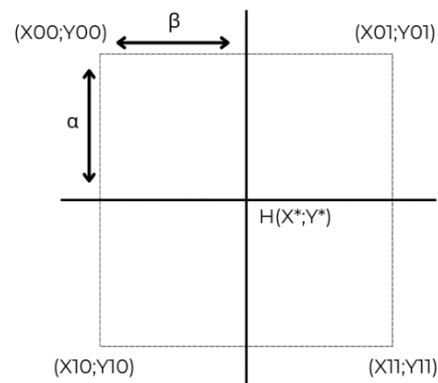
| <b>STT</b> | <b>Chức năng/Hàm</b>    | <b>Mức độ hoàn thành</b> |
|------------|-------------------------|--------------------------|
| 1          | Thay đổi độ sáng        | 100%                     |
| 2          | Thay đổi độ tương phản  | 100%                     |
| 3.1        | Lật ảnh ngang           | 100%                     |
| 3.2        | Lật ảnh dọc             | 100%                     |
| 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%                     |
| 6          | Cắt ảnh theo kích thước | 100%                     |
| 7.1        | Cắt ảnh theo khung tròn | 100%                     |
| 7.2        | Cắt ảnh theo khung elíp | 100%                     |
| 8          | Hàm main                | 100%                     |
| 9          | Phóng to/Thu nhỏ 2x     | 100%                     |

# 1. Ý TƯỞNG THỰC HIỆN VÀ MÔ TẢ CÁC HÀM

## 1.1. Ý tưởng thực hiện

❖ **Đồ án 02 - Image Processing** tập trung vào việc phát triển một chương trình xử lý ảnh đa năng với nhiều chức năng khác nhau. Đồ án này dựa trên nền tảng lý thuyết vững chắc về biểu diễn ảnh số dưới dạng ma trận các điểm ảnh, kết hợp với việc ứng dụng thành thạo thư viện NumPy và ứng dụng các phương pháp xử lý ảnh mạnh mẽ như:

- Nội suy song tuyến tính: là phương pháp tương tự như nội suy tuyến tính nhưng thay vì xét trên một chiều không gian, nội suy song tuyến tính có ý tưởng là xét ảnh hưởng của 4 nút lưới thân cận nhất lên điểm có tọa độ trung gian. Giả sử, ta có hệ trục tọa độ sau:



Khi đó, giá trị nội suy song tuyến tính được tính theo công thức sau:

$$H(X^*; Y^*) = (1-\alpha).(1-\beta).H(X_{00}; Y_{00}) + \alpha.(1-\beta).H(X_{10}; Y_{10}) \\ + (1-\alpha).\beta.H(X_{01}; Y_{01}) + \alpha.\beta.H(X_{11}; Y_{11})$$

- Áp dụng kernel tích chập: Kernel, còn được gọi là ma trận tích chập hoặc mask, là một ma trận nhỏ (thường có kích thước 3x3 hoặc 5x5) được sử dụng trong xử lý ảnh để thực hiện các phép biến đổi trên ảnh (làm mờ ảnh, phát hiện cạnh, tăng độ nét và giảm nhiễu). Kernel được áp dụng lên ảnh thông qua phép tích chập, di chuyển trên toàn bộ ảnh để tạo ra một ảnh mới với các đặc tính khác nhau tùy thuộc vào giá trị của kernel.
- Hiệu chỉnh màu sắc: Ứng dụng các công thức, bộ lọc để điều chỉnh các thông số màu sắc, cải thiện chất lượng và tính thẩm mỹ của hình ảnh.

❖ **Các bước thực hiện:**

- Đọc ảnh đầu vào và chuyển đổi thành mảng NumPy.
- Áp dụng các thuật toán xử lý ảnh tương ứng với từng chức năng.
- Hiển thị và lưu kết quả xử lý.
- Đo thời gian thực hiện của mỗi chức năng để đánh giá hiệu suất.

## 1.2. Các thư viện được sử dụng

❖ **NumPy** là một thư viện mạnh mẽ để xử lý các mảng và các phép toán số học.

Trong đoạn mã này, NumPy được sử dụng để:

- Chuyển đổi ảnh thành mảng đa chiều để dễ dàng xử lý.
- Thực hiện các phép toán ma trận hiệu quả trên dữ liệu ảnh.
- Tạo và áp dụng các kernel cho các chức năng như làm mờ dựa trên kernel Gaussian và làm sắc nét dựa trên kernel Laplacian.

❖ **PIL (Pillow)** là một thư viện để mở, xử lý và lưu trữ các định dạng hình ảnh khác nhau. Trong đoạn mã này, Pillow được sử dụng để:

- Đọc ảnh từ file.
- Lưu ảnh đã xử lý vào file.
- Chuyển đổi giữa các định dạng ảnh khác nhau.

❖ **Matplotlib** là một thư viện để tạo các biểu đồ và hình ảnh trực quan. Trong đoạn mã này, Matplotlib được sử dụng để:

- Hiển thị ảnh gốc và ảnh đã xử lý.
- Tạo các subplot để so sánh kết quả xử lý ảnh.
- Điều chỉnh các thuộc tính hiển thị như tiêu đề và tỷ lệ khung hình.

❖ **Timeit** là một thư viện để cung cấp các hàm liên quan đến thời gian. Trong đoạn mã này, timeit được sử dụng để đo thời gian thực hiện của các chức năng xử lý ảnh nhằm đánh giá hiệu suất thời gian.

❖ **OS** là một thư viện để cung cấp một cách thuận tiện để làm việc với hệ thống tệp và thư mục. Trong đoạn mã này, os được sử dụng để:

- Kiểm tra sự tồn tại của tệp hình ảnh đầu vào.
- Tạo các thư mục để lưu trữ các hình ảnh đã nén.

## 1.3. Mô tả các hàm

### 1.3.1. Các hàm hỗ trợ

#### ❖ Hàm `'read_img(img_path)'`

- Đầu vào:
  - `img_path: str` : đường dẫn đến ảnh.
- Đầu ra:
  - `img: numpy.ndarray` : mảng NumPy đại diện cho ảnh.
- Chức năng: Đọc hình ảnh từ đường dẫn và chuyển đổi nó thành mảng NumPy hai chiều để biểu diễn.
- Ý tưởng: Sử dụng thư viện PIL để mở và đọc hình ảnh, sau đó chuyển đổi nó thành mảng NumPy để dễ dàng xử lý bằng các thuật toán số.

#### ❖ Hàm `'save_and_show(img, save_path, title, cmap)'`

- Đầu vào:
  - `img: numpy.ndarray` : hình ảnh cần lưu và hiển thị.
  - `save_path: str` : đường dẫn để lưu hình ảnh.
  - `title: str` : tiêu đề cho hình ảnh được hiển thị.
  - `cmap: str` : bản đồ màu cho hiển thị hình ảnh.
- Đầu ra:
  - Lưu và hiển thị ảnh
- Chức năng: Lưu hình ảnh đã xử lý vào file và hiển thị nó.
- Ý tưởng: Sử dụng PIL để lưu hình ảnh và matplotlib để hiển thị hình ảnh với tiêu đề và colormap tùy chọn.

#### ❖ Hàm `'main()'`

- Mô tả chi tiết các bước:
  - Người dùng nhập đường dẫn hình ảnh
  - Vòng lặp thực hiện các chức năng:
    - Người dùng được chọn danh sách các chức năng xử lý hình ảnh cơ bản, bao gồm: thay đổi độ sáng, thay đổi độ tương phản, lật ảnh, chuyển đổi ảnh RGB thành ảnh grayscale/sepia, làm mờ hoặc sắc nét ảnh, cắt ảnh ở trung tâm, cắt ảnh theo khung, thu nhỏ hoặc phóng to.



- Người dùng nhập các kí tự khác được yêu cầu trong danh sách sẽ thoát khỏi vòng lặp.
- Chương trình xử lý và hiển thị kết quả ra màn hình cũng như lưu vào thư mục output với đường dẫn tương ứng.
- Thông báo lỗi tương ứng khi người dùng lựa chọn không hợp lệ và dừng chương trình.

### 1.3.2. Thay đổi độ sáng

- Tên hàm: `'adjust_brightness(img, alpha)'`
- Đầu vào:
  - `img: numpy.ndarray` : hình ảnh cần xử lý.
  - `alpha: float` : hệ số điều chỉnh độ sáng.
- Đầu ra:
  - `img: numpy.ndarray` : hình ảnh đã xử lý.
- Chức năng: Điều chỉnh độ sáng của hình ảnh.
- Ý tưởng: Thêm một giá trị alpha vào tất cả các pixel của hình ảnh, sau đó dùng `numpy.clip()` cắt giá trị để nằm trong khoảng [0, 255].
- Lưu ý: với alpha dương, ta tăng độ sáng cho bức ảnh; ngược lại, với alpha âm, ta giảm độ sáng cho bức ảnh.

### 1.3.3. Thay đổi độ tương phản

- Tên hàm: `'adjust_contrast(img, alpha)'`
- Đầu vào:
  - `img: numpy.ndarray` : hình ảnh cần xử lý.
  - `alpha: float` : hệ số điều chỉnh độ tương phản.
- Đầu ra:
  - `img: numpy.ndarray` : hình ảnh đã xử lý.
- Chức năng: Điều chỉnh độ tương phản của hình ảnh.
- Ý tưởng: Tính giá trị trung bình của ảnh. Sau đó áp dụng công thức điều chỉnh độ tương phản trên từng pixel như sau:

$$\text{new\_pixel} = (\text{old\_pixel} - \text{mean}) * \text{alpha} + \text{mean}$$

Sau đó, ta dùng `numpy.clip()` cắt giá trị để nằm trong khoảng [0, 255].

- Lưu ý:

- $\alpha > 1$ : tăng độ tương phản.
- $0 < \alpha < 1$ : giảm độ tương phản.

#### 1.3.4. Lật ảnh

- Tên hàm: `'flip_img(img, direction)'`
- Đầu vào:
  - `img: numpy.ndarray` : hình ảnh cần xử lý.
  - `direction` : chiều lật ảnh (ngang hoặc dọc).
- Đầu ra:
  - `img: numpy.ndarray` : hình ảnh đã xử lý.
- Chức năng: Lật hình ảnh theo chiều ngang hoặc dọc.
- Giải thích: Sử dụng hàm NumPy để lật ảnh:
  - `numpy.fliplr()` cho lật ngang
  - `numpy.flipud()` cho lật dọc

#### 1.3.5. Chuyển đổi ảnh RGB thành ảnh xám/sepia

- Tên hàm: `'convert_to_grayscale_or_sepia(img, mode)'`
- Đầu vào:
  - `img: numpy.ndarray` : hình ảnh cần xử lý.
  - `mode: str` : chế độ xử lý (grayscale hoặc sepia).
- Đầu ra:
  - `img: numpy.ndarray` : hình ảnh đã xử lý.
- Chức năng: Chuyển đổi hình ảnh sang thang độ xám hoặc màu sepia.
- Giải thích:
  - Chế độ grayscale:
    - Áp dụng phương pháp luminosity cho mỗi pixel:
 
$$Y = 0.299R + 0.587G + 0.114B$$
    - Giới hạn giá trị pixel trong khoảng  $[0, 255]$ .
  - Chế độ sepia:
    - Sử dụng ma trận chuyển đổi cho mỗi kênh màu.
    - Giới hạn giá trị pixel trong khoảng  $[0, 255]$ .

### 1.3.6. Làm mờ/sắc nét ảnh

#### ❖ Hàm '*blur\_or\_sharpen\_img(img, mode)*'

- Đầu vào:
  - *img: numpy.ndarray* : hình ảnh cần xử lý.
  - *mode: str* : chế độ xử lý (blur hoặc sharpen).
- Đầu ra:
  - *img: numpy.ndarray* : hình ảnh đã xử lý.
- Chức năng: Làm mờ hoặc làm sắc nét hình ảnh.
- Giải thích:
  - Cung cấp kernel phù hợp cho từng chế độ xử lý:
    - Làm mờ ảnh: *Kernel Gaussian Blur 5x5*.
    - Làm sắc nét ảnh: *Kernel Laplacian*.
  - Gọi hàm tích chập '*convolve*' để nhận ma trận ảnh kết quả.

#### ❖ Hàm '*convolve(img, kernel)*'

- Đầu vào:
  - *img: numpy.ndarray* : hình ảnh cần xử lý.
  - *kernel: numpy.ndarray* : kernel.
- Đầu ra:
  - *img: numpy.ndarray* : hình ảnh đã xử lý.
- Chức năng: Thuật toán áp dụng tích chập cho hình ảnh và kernel.
- Giải thích:
  - Thêm padding cho ảnh bằng cách sao chép các pixel biên.
  - Sử dụng kỹ thuật "sliding window" để áp dụng kernel lên từng vùng của ảnh.
  - Thực hiện phép nhân ma trận element-wise (phép nhân từng phần tử) và tính tổng.
  - Giới hạn giá trị pixel trong khoảng [0, 255].

### 1.3.7. Cắt ảnh theo kích thước (cắt ở trung tâm)

- Tên hàm: '*crop\_center(img, crop\_size)*'
- Đầu vào:
  - *img: numpy.ndarray* : hình ảnh cần xử lý.

- *crop\_size: int* : kích thước ảnh vuông muốn cắt.
- Đầu ra:
  - *img: numpy.ndarray* : hình ảnh đã xử lý.
- Chức năng: Cắt ảnh thành hình vuông từ trung tâm.
- Giải thích:
  - Tính toán tọa độ bốn đỉnh của hình vuông.
  - Cắt hình vuông từ tọa độ bốn đỉnh trong mảng hình ảnh.

#### 1.3.8. Cắt ảnh theo khung

##### ❖ Hàm '*crop\_circle(img)*'

- Đầu vào:
  - *img: numpy.ndarray* : hình ảnh cần xử lý.
- Đầu ra:
  - *img: numpy.ndarray* : mảng NumPy đại diện cho ảnh.
- Chức năng: Cắt ảnh thành hình tròn.
- Ý tưởng:
  - Tạo mask hình tròn bằng cách sử dụng phương trình đường tròn.
  - Áp dụng mặt nạ lên ảnh gốc, đặt các pixel ngoài vùng là 0 (đen).

##### ❖ Hàm '*crop\_crossed\_ellipses(img)*'

- Đầu vào:
  - *img: numpy.ndarray* : hình ảnh cần xử lý.
- Đầu ra:
  - *img: numpy.ndarray* : mảng NumPy đại diện cho ảnh.
- Chức năng: Cắt ảnh thành hình hai ellipse giao nhau.
- Ý tưởng:
  - Tạo hai mặt nạ ellipse xoay 45 độ và -45 độ
  - Kết hợp hai mặt nạ bằng phép OR
  - Áp dụng mặt nạ lên ảnh gốc, đặt các pixel ngoài vùng là 0 (đen).

#### 1.3.9. Thu nhỏ, phóng to ảnh

- Tên hàm: '*zoom\_img(img, zoom\_factor)*'
- Đầu vào:
  - *img: numpy.ndarray* : hình ảnh cần xử lý.

- *zoom\_factor: float* : hệ số zoom.
- Đầu ra:
  - *img: numpy.ndarray* : hình ảnh đã xử lý.
- Chức năng: Phóng to hoặc thu nhỏ hình ảnh.
- Giải thích: Áp dụng thuật toán Bilinear Interpolation (Nội suy song tuyến tính). Ý tưởng chính:
  - Tạo ảnh mới với kích thước được điều chỉnh theo hệ số zoom.
  - Với mỗi pixel trong ảnh mới, tính toán vị trí tương ứng trong ảnh gốc.
  - Sử dụng 4 pixel lân cận trong ảnh gốc để tính giá trị cho pixel mới bằng công thức được đề cập ở phần ***Ý tưởng thực hiện (trang 5)***.

## 2. HÌNH ẢNH KẾT QUẢ

### 2.1. Môi trường thực thi

- Môi trường thực thi: Anaconda
- Ngôn ngữ lập trình: Python version 3.11.5
- Hướng dẫn thực thi mã nguồn
  - Tải môi trường Anaconda và file mã nguồn Jupyter Notebook.
  - Mở IDE Visual Studio Code (VS Code xanh).
  - Chọn kernel Python 3.11.5 (Anaconda).
  - Chạy chương trình và nhập vào các thông tin cần thiết như đường dẫn đến tệp tin ảnh cần xử lý, chọn các chức năng cần xử lý và thực hiện theo hướng dẫn.
  - Xem và lưu kết quả.

## 2.2. Thực thi chương trình






### 2.2.1. Test case 1

Ảnh gốc







- Loại: ảnh hoạt hình
- Kích thước: 512x512
- Dung lượng: 35.8 KB

| Chức năng   | Hiệu suất  | Kết quả   |
|---|--|---|
| Thay đổi độ sáng: <ul style="list-style-type: none"><li>• <math>\alpha = 100</math></li></ul>       | <ul style="list-style-type: none"><li>- Kích thước: 512x512</li><li>- Dung lượng: 34.1 KB</li><li>- Thời gian xử lý: 0.04487560s</li></ul> | A cartoon illustration of a boy with a large orange mouth and a yellow background, similar to the original image but with increased brightness. |
| Thay đổi độ tương phản: <ul style="list-style-type: none"><li>• <math>\alpha = 2.5</math></li></ul> | <ul style="list-style-type: none"><li>- Kích thước: 512x512</li><li>- Dung lượng: 46.9 KB</li><li>- Thời gian xử lý: 0.03201360s</li></ul> | A cartoon illustration of a boy with a large orange mouth and a yellow background, similar to the original image but with increased contrast.   |

|  |  |   |
|--|--|---|
| <p>Lật ảnh:</p> <ul style="list-style-type: none"> <li>• direct. = horizontal</li> </ul> | <ul style="list-style-type: none"> <li>- Kích thước: 512x512</li> <li>- Dung lượng: 38 KB</li> <li>- Thời gian xử lý: 0.00030520s</li> </ul>   |    |
| <p>Chuyển ảnh RGB sang ảnh grayscale</p>   | <ul style="list-style-type: none"> <li>- Kích thước: 512x512</li> <li>- Dung lượng: 31.7 KB</li> <li>- Thời gian xử lý: 0.02044850s</li> </ul> |    |
| <p>Chuyển ảnh RGB sang ảnh sepia</p>   | <ul style="list-style-type: none"> <li>- Kích thước: 512x512</li> <li>- Dung lượng: 37.7 KB</li> <li>- Thời gian xử lý: 0.03127520s</li> </ul> |   |
| <p>Làm mờ</p>  | <ul style="list-style-type: none"> <li>- Kích thước: 512x512</li> <li>- Dung lượng: 31.8 KB</li> <li>- Thời gian xử lý: 0.08694860s</li> </ul> |  |
| <p>Làm sắc nét</p>   | <ul style="list-style-type: none"> <li>- Kích thước: 512x512</li> <li>- Dung lượng: 51.2 KB</li> <li>- Thời gian xử lý: 0.02777010s</li> </ul> |  |



|   |   |   |
|---|---|---|
| <p>Cắt ảnh (trung tâm)</p> <ul style="list-style-type: none"> <li>size = 256</li> </ul> | <ul style="list-style-type: none"> <li>Kích thước: 256x256</li> <li>Dung lượng: 10.4 KB</li> <li>Thời gian xử lý: 0.00008480s</li> </ul>  |    |
| <p>Cắt ảnh khung tròn</p>   | <ul style="list-style-type: none"> <li>Kích thước: 512x512</li> <li>Dung lượng: 41 KB</li> <li>Thời gian xử lý: 0.00499880s</li> </ul>    |    |
| <p>Cắt ảnh khung 2 elip chéo</p>  | <ul style="list-style-type: none"> <li>Kích thước: 512x512</li> <li>Dung lượng: 43.5 KB</li> <li>Thời gian xử lý: 0.02225280s</li> </ul>  |   |
| <p>Zoom ảnh</p> <ul style="list-style-type: none"> <li>factor = 2.0</li> </ul>          | <ul style="list-style-type: none"> <li>Kích thước: 1024x1024</li> <li>Dung lượng: 92.2 KB</li> <li>Thời gian xử lý: 0.3997990s</li> </ul> |  |






### 2.2.2. Test case 2

#### Ảnh gốc



- Loại: ảnh chân dung
- Kích thước: 512x512
- Dung lượng: 39.1 KB

| Chức năng   | Hiệu suất  | Kết quả |
|---|--|---------|
| Thay đổi độ sáng: <ul style="list-style-type: none"> <li>• <math>\alpha = -100</math></li> </ul>      | <ul style="list-style-type: none"> <li>- Kích thước: 512x512</li> <li>- Dung lượng: 10.3 KB</li> <li>- Thời gian xử lý: 0.02654520s</li> </ul> |         |
| Thay đổi độ tương phản: <ul style="list-style-type: none"> <li>• <math>\alpha = 0.5</math></li> </ul> | <ul style="list-style-type: none"> <li>- Kích thước: 512x512</li> <li>- Dung lượng: 14.2 KB</li> <li>- Thời gian xử lý: 0.03627020s</li> </ul> |         |
| Lật ảnh: <ul style="list-style-type: none"> <li>• direct. = vertical</li> </ul>                       | <ul style="list-style-type: none"> <li>- Kích thước: 512x512</li> <li>- Dung lượng: 19.8 KB</li> <li>- Thời gian xử lý: 0.00062890s</li> </ul> |         |

|  |  |   |
|--|--|---|
| Chuyển ảnh RGB sang ảnh grayscale  | <ul style="list-style-type: none"> <li>- Kích thước: 512x512</li> <li>- Dung lượng: 17.5 KB</li> <li>- Thời gian xử lý: 0.01477850s</li> </ul> |    |
| Chuyển ảnh RGB sang ảnh sepia  | <ul style="list-style-type: none"> <li>- Kích thước: 512x512</li> <li>- Dung lượng: 21.3 KB</li> <li>- Thời gian xử lý: 0.03772430s</li> </ul> |    |
| Làm mờ   | <ul style="list-style-type: none"> <li>- Kích thước: 512x512</li> <li>- Dung lượng: 16.1 KB</li> <li>- Thời gian xử lý: 0.11288010s</li> </ul> |   |
| Làm sắc nét  | <ul style="list-style-type: none"> <li>- Kích thước: 512x512</li> <li>- Dung lượng: 31.1 KB</li> <li>- Thời gian xử lý: 0.02659520s</li> </ul> |  |
| Cắt ảnh (trung tâm) <ul style="list-style-type: none"> <li>• size = 200</li> </ul> | <ul style="list-style-type: none"> <li>- Kích thước: 200x200</li> <li>- Dung lượng: 5.12 KB</li> <li>- Thời gian xử lý: 0.00008360s</li> </ul> |  |

|   |  |  |
|---|--|--|
| Cắt ảnh khung tròn  | <ul style="list-style-type: none"> <li>- Kích thước: 512x512</li> <li>- Dung lượng: 19.8 KB</li> <li>- Thời gian xử lý: 0.00534200s</li> </ul> |   |
| Cắt ảnh khung 2 elip chéo   | <ul style="list-style-type: none"> <li>- Kích thước: 512x512</li> <li>- Dung lượng: 20.9 KB</li> <li>- Thời gian xử lý: 0.02220940s</li> </ul> |   |
| Zoom ảnh <ul style="list-style-type: none"> <li>• factor = 0.5</li> </ul> | <ul style="list-style-type: none"> <li>- Kích thước: 256x256</li> <li>- Dung lượng: 7.19 KB</li> <li>- Thời gian xử lý: 0.02054890s</li> </ul> |  |

### 2.2.3. Test case 3






#### Ảnh gốc






- Loại: ảnh phong cảnh
- Kích thước: 1920x1080
- Dung lượng: 264 KB

| Chức năng   | Hiệu suất  | Kết quả |
|---|--|---------|
| Thay đổi độ sáng: <ul style="list-style-type: none"> <li>• <math>\alpha = 20</math></li> </ul>        | - Kích thước: 1920x1080<br>- Dung lượng: 256 KB<br>- Thời gian xử lý: 0.1042451s |         |
| Thay đổi độ tương phản: <ul style="list-style-type: none"> <li>• <math>\alpha = 1.2</math></li> </ul> | - Kích thước: 1920x1080<br>- Dung lượng: 266 KB<br>- Thời gian xử lý: 0.2522850s |         |
| Lật ảnh: <ul style="list-style-type: none"> <li>• direct. = vertical</li> </ul>                       | - Kích thước: 1920x1080<br>- Dung lượng: 256 KB<br>- Thời gian xử lý: 0.0028790s |         |



|   |   |  |
|---|---|--|
| <p>Chuyển ảnh RGB sang ảnh grayscale</p>  | <p>- Kích thước: 1920x1080</p> <p>- Dung lượng: 213 KB</p> <p>- Thời gian xử lý: 0.1501709s</p> |    |
| <p>Chuyển ảnh RGB sang ảnh sepia</p>  | <p>- Kích thước: 1920x1080</p> <p>- Dung lượng: 249 KB</p> <p>- Thời gian xử lý: 0.3790328s</p> |    |
| <p>Làm mờ</p>   | <p>- Kích thước: 1920x1080</p> <p>- Dung lượng: 217 KB</p> <p>- Thời gian xử lý: 1.1503056s</p> |   |
| <p>Làm sắc nét</p>  | <p>- Kích thước: 1920x1080</p> <p>- Dung lượng: 329 KB</p> <p>- Thời gian xử lý: 0.3163480s</p> |  |
| <p>Cắt ảnh (trung tâm)</p> <ul style="list-style-type: none"> <li>size = 600</li> </ul> | <p>- Kích thước: 600x600</p> <p>- Dung lượng: 48.9 KB</p> <p>- Thời gian xử lý: 0.0007679s</p>  |  |

|   |   |   |
|---|---|---|
| Cắt ảnh khung tròn  | <ul style="list-style-type: none"> <li>- Kích thước: 1920x1080</li> <li>- Dung lượng: 144 KB</li> <li>- Thời gian xử lý: 0.0660538s</li> </ul>  |   |
| Cắt ảnh khung 2 elip chéo   | <ul style="list-style-type: none"> <li>- Kích thước: 1920x1080</li> <li>- Dung lượng: 156 KB</li> <li>- Thời gian xử lý: 0.1469203s</li> </ul>  |   |
| Zoom ảnh <ul style="list-style-type: none"> <li>• factor = 3.0</li> </ul> | <ul style="list-style-type: none"> <li>- Kích thước: 5760x3240</li> <li>- Dung lượng: 1.09 MB</li> <li>- Thời gian xử lý: 13.965082s</li> </ul> |  |

### 3. NHẬN XÉT KẾT QUẢ

Dựa trên kết quả thu được từ phần trên, chúng ta có thể rút ra các nhận xét về chất lượng hình ảnh và thời gian xử lý của các chức năng xử lý ảnh như sau:

#### 1. Hiệu suất xử lý:

- Thời gian xử lý phụ thuộc vào kích thước ảnh và độ phức tạp của thao tác. Ảnh có kích thước lớn hơn (như test case 3 với 1920x1080 pixels) thường có thời gian xử lý lâu hơn so với ảnh nhỏ hơn.
- Các thao tác đơn giản như lật ảnh và cắt ảnh (trung tâm) có thời gian xử lý rất nhanh, thường dưới 0.001 giây.
- Các thao tác phức tạp hơn như làm mờ, làm sắc nét và chuyển đổi màu sắc (sepia, grayscale) có thời gian xử lý lâu hơn.
- Thao tác zoom ảnh, đặc biệt là khi phóng to (factor > 1), có thời gian xử lý lâu nhất trong tất cả các thao tác, đặc biệt là với ảnh kích thước lớn như test case 3 (13.965082s).

#### 2. Dung lượng ảnh:

- Hầu hết các thao tác không làm thay đổi đáng kể dung lượng ảnh, ngoại trừ một số trường hợp cụ thể:
  - Làm sắc nét thường làm tăng dung lượng ảnh.
  - Chuyển sang ảnh grayscale thường làm giảm dung lượng.
  - Zoom ảnh với factor > 1 làm tăng đáng kể dung lượng ảnh.

#### 3. Kích thước ảnh:

- Hầu hết các thao tác giữ nguyên kích thước ảnh, ngoại trừ:
  - Cắt ảnh (trung tâm) làm giảm kích thước ảnh.
  - Zoom ảnh thay đổi kích thước tỷ lệ thuận với hệ số zoom.

#### 4. Đặc điểm theo loại ảnh:

- Ảnh hoạt hình (test case 1) có xu hướng có dung lượng nhỏ hơn so với ảnh chân dung hoặc phong cảnh cùng kích thước.
- Ảnh chân dung (test case 2) có sự thay đổi dung lượng đáng kể khi áp dụng các hiệu ứng như thay đổi độ sáng và độ tương phản.



- Ảnh phong cảnh (test case 3) có dung lượng lớn nhất và thời gian xử lý lâu nhất do kích thước lớn.

5. Hiệu quả của các thao tác:

- Thay đổi độ sáng và độ tương phản có tác động rõ rệt đến dung lượng ảnh, đặc biệt là với ảnh chân dung.
- Cắt ảnh khung tròn và khung 2 elip chéo không làm giảm kích thước ảnh nhưng có thể làm biến động nhẹ (tăng hoặc giảm) dung lượng do thêm thông tin về vùng trong suốt.

## TỔNG KẾT

**Đồ án 02 - Image Processing** đã thành công trong việc phát triển một chương trình xử lý ảnh đa năng với nhiều chức năng khác nhau. Dựa trên nền tảng lý thuyết về biểu diễn ảnh số dưới dạng ma trận các điểm ảnh, kết hợp với việc ứng dụng thư viện NumPy và các phương pháp xử lý ảnh, chúng tôi đã đạt được các kết quả sau:

- ❖ Phát triển thành công các chức năng xử lý ảnh cơ bản:
  - Thay đổi độ sáng và độ tương phản
  - Lật ảnh theo chiều ngang hoặc dọc
  - Chuyển đổi ảnh RGB thành ảnh xám hoặc sepia
  - Làm mờ hoặc sắc nét ảnh
  - Cắt ảnh theo kích thước hoặc theo khung (hình tròn, ellipse giao nhau)
  - Thu nhỏ hoặc phóng to ảnh
- ❖ Ứng dụng thành công các kỹ thuật xử lý ảnh nâng cao:
  - Nội suy song tuyến tính trong việc thu phóng ảnh
  - Áp dụng kernel tích chập cho việc làm mờ và sắc nét ảnh
  - Sử dụng các phương pháp toán học để tạo mặt nạ cho việc cắt ảnh theo khung đặc biệt
- ❖ Tối ưu hóa hiệu suất xử lý:
  - Sử dụng thư viện NumPy để thực hiện các phép toán ma trận hiệu quả
  - Áp dụng các thuật toán tối ưu cho từng chức năng xử lý ảnh

Tuy nhiên, vẫn còn một số hạn chế và hướng phát triển trong tương lai:

- ❖ Cải thiện hiệu suất:
  - Tìm hiểu thêm FFT (fast Fourier transform) để tăng tốc độ xử lý thuật toán tích chập.
  - Làm ảnh trở nên đẹp và mềm mại hơn dựa trên histogram.
- ❖ Nâng cao giao diện:
  - Phát triển giao diện đồ họa cho phép người dùng tương tác trực tiếp với ảnh.
  - Tích hợp chức năng xem trước kết quả xử lý trong thời gian thực.

## TÀI LIỆU THAM KHẢO

- [1] NumPy documentation, <https://numpy.org/doc/>, ngày truy cập: 22/7/2024
- [2] Kernel (image processing), [https://en.wikipedia.org/wiki/Kernel\\_\(image\\_processing\)](https://en.wikipedia.org/wiki/Kernel_(image_processing)), ngày truy cập: 22/7/2024
- [3] Dynamsoft, Image Processing 101 Chapter 1.3: Color Space Conversion, <https://www.dynamsoft.com/blog/insights/image-processing/image-processing-101-color-space-conversion/>, ngày truy cập: 22/7/2024
- [4] 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>, ngày truy cập: 22/7/2024
- [5] Hình ảnh:
- <http://www.inanhducanh.com/loi-co-ban-thuong-gap-khi-chup-anh-chan-dung-va-cach-su-ly.html>, ngày truy cập: 22/7/2024
  - <https://wall.alphacoders.com/big.php?i=1081449>, ngày truy cập: 22/7/2024
  - <https://www.vietnamfineart.com.vn/anh-anime/anh-hoat-hinh-anime/>, ngày truy cập: 22/7/2024