UNIVERSITY OF SCIENCE
FALCUTY OF INFORMATION TECHNOLOGY

SUBJECT: APPLIED MATH AND STATISTICS

PROJECT 2: IMAGE PROCESSING
CLASS 20CLC11

Student: Mai Quý Trung
ID: 20127370

Lecturers: **VŨ QUỐC HOÀNG, NGUYỄN VĂN QUANG HUY, TRẦN THỊ THẢO NHI, PHAN THỊ PHƯƠNG UYÊN**

Table of Contents

I) Bảng công việc

Công việc	Hoàn thành	Ghi chú
Thay đổi độ sáng ảnh	100%	
Thay đổi độ tương phản ảnh	100%	
Lật ảnh ngang dọc	100%	
Chuyển đổi ảnh RGB về ảnh xám	100%	
Chồng 2 ảnh xám cùng kích thước	100%	
Làm mờ ảnh	100%	
Cắt ảnh theo hình tròn và elip	100%	

II) Tổng quan

1. Giới thiệu đồ án

- 1 bức ảnh có thể lưu trữ dưới ma trận của các điểm ảnh.
- Có nhiều loại ảnh được sử dụng trong thực tế: ảnh xám, ảnh màu, …
- Ảnh mầu tồn tại dưới nhiều dạng màu (màu rgb, rgba …) hoặc nhiều định dạng khác nhau (png, jpg …)
- Phổ biến nhất là ảnh RGB, trong đó mỗi điểm ảnh sẽ lưu trữ 3 thông tin kênh màu (mỗi kênh màu là 1 byte): R (red - đỏ), G (green – xanh là), B (blue – xanh dương)
- Như vậy số màu trong ảnh RGB là 256^3 tương ứng với gần 17 triệu màu.
- Trong đồ án 1, bạn đã được giới thiệu rằng ảnh được lưu trữ dưới dạng ma trận các điểm ảnh. Mỗi điểm ảnh có thể là một giá trị (ảnh xám) hoặc một vector (ảnh màu).

2. Yêu cầu đồ án

- Trong đồ án này, bạn được yêu cầu thực hiện các chức năng xử lý ảnh cơ bản sau:
  + Thay đổi độ sáng cho ảnh
  + Thay đổi độ tương phản
  + Lật ảnh (ngang - dọc)
  + Chuyển đổi ảnh RGB thành ảnh xám
  + Chồng 2 ảnh cùng kích thước (chỉ làm trên ảnh xám)
  + Làm mờ ảnh
  + Viết hàm main xử lý với các yêu cầu sau:
- Cho phép người dùng nhập vào tên tập tin ảnh mỗi khi hàm main được thực thi.
- Cho phép người dùng lựa chọn chức năng xử lý ảnh (từ 1 đến 6, đối với chức năng 4 cho phép lựa chọn giữa lật ngang hoặc lật dọc). Lựa chọn 0 cho phép thực hiện tất cả chức năng với tên file đầu ra tương ứng với từng chức năng.
- Nâng cao (không bắt buộc): Sinh viên thực cắt nội dung ảnh theo khung được áp lên, với khung là hình như hình tròn và 2 hình elip chéo nhau.

III) Chi tiết đồ án

1. Môi trường làm việc

- Ngôn ngữ sử dụng: Python (version 3.10.4)
- Text editor và trình biên dịch: Jupyter Notebook
- Source code: 20127370.ipynb

2. Ý tưởng

- Ở đồ án này, chúng ta sẽ làm việc trực tiếp trên 2 loại ảnh: ảnh màu RGB và ảnh xám.
- Đa số các yêu cầu của đồ án sẽ có 1 cách làm chung là đều tác động lên các pixel trong ma trận điểm ảnh, thay đổi các giá trị và chuyển đổi nhuần nhuyễn giữa pixel RGB và pixel ảnh xám.
- Với yêu cầu thay đổi độ sáng và độ tương phản, ta sẽ thay đổi giá trị của tất cả các pixel theo 1 công thức chung, 2 hàm này sẽ gần như tương tự nhau.
- Với yêu cầu lật ảnh ngang/dọc, chúng ta chỉ việc hoán đổi vị trí của các pixel đối xứng với nhau qua trục ngang/dọc.
- Với yêu cầu ảnh rgb chuyển sang xám, thay vì mỗi pixel là 1 mảng gồm 3 phần tử RGB thì ta biến chúng về còn 1 giá trị duy nhất với tỉ lệ giữa 3 màu theo quy ước.
- Với yêu cầu ghép 2 ảnh xám thành 1 cùng kích thước, ta chuyển đổi 2 ảnh gốc từ rgb sang xám rồi sau đó cứ mỗi pixel kết quả sẽ là trung bình cộng của cặp pixel tương ứng.
- Với yêu cầu làm mờ ảnh, ta sẽ sử dụng phương pháp Gaussian blur 3 x 3, gom nhóm lần lượt 9 pixel theo tỉ lệ nhất định.
- Cuối cùng, với yêu cầu nâng cao, ta cần xác định phương trình tổng quát trong mặt phẳng của hình tròn và hình elip. Sau đó, nếu các điểm ảnh nào nằm bên ngoài phương trình, ta sẽ gán giá trị pixel đó sang màu đen (tức là R = 0, G = 0, B = 0).

3. Chi tiết các hàm yêu cầu

Bước 1: Import thư viện

- Em sử dụng các thư viện như sau:
  + **matplotlib**: dùng **plt** để xuất ảnh dưới dạng figure và lưu lại dưới dạng png/pdf
  + **numpy**: thư viện dùng để xử lí các thông tin trên các ma trận n chiều và random choice
  + **PIL**: sử dụng **Image** từ thư viện này để đọc ảnh và chuyển chúng sang dạng ảnh RGB để thao tác trên ma trận
  + **math**: em sử dụng thư viện này để tính các hàm **cos()**, **sin()** và thao tác với số pi cho yêu cầu 7

Bước 2: Khởi tạo các giá trị ban đầu

- Hàm dưới đây sẽ hoạt động như sau:

+ Cho người dùng nhập vào tên file
+ Cho người dùng nhập vào định dạng file muốn xuất (png/pdf)
+ Sau đó hàm sẽ xử lí chuỗi lấy tên của file (không chứa extension) và định dạng file muốn xuất ra

Bước 3: Xử lí ảnh, chuyển đổi từ file raw sang các pixel màu

- Các bước hàm thực hiện như sau:
  + Hàm nhận input argument là filename từ người dùng nhập vào
  + Sau đó sử dụng thư viện **PIL** hàm **Image.open** và **convert('RGB')** để đọc ảnh và chuyển ảnh sang dạng ảnh RGB gồm 3 màu: R cho màu đỏ (Red), G cho màu xanh lá (Green), B cho màu xanh dương (Blue)
  + Tiếp theo ta sử dụng hàm **np.array** để chuyển ảnh RGB sang ma trận 3 chiều gồm: chiều cao, chiều rộng và mỗi pixel là 1 mảng màu tương ứng với 3 màu RGB
  + Chuyển đổi từ ma trận 2 chiều với các mảng màu rgb/rgba sang ma trận 1 chiều với hàm **reshape**
  + Sau đó lưu các thông số chiều cao, chiều rộng và số lượng màu của ảnh vào và trả về các giá trị đó

Bước 4: Viết hàm truncate

- Hàm có tác dụng tránh để giá trị của mỗi màu trong RGB bị vượt quá [0, 255] bằng cách lấy min của pixel và 255 sau đó lấy max của kết quả vừa ra với số 0:
- Giá trị của pixel phải nằm trong khoảng từ 0 đến 255.
- Nếu giá trị pixel lớn hơn 255, pixel đó sẽ được gán bằng 255.
- Nếu giá trị pixel nhỏ hơn 0, pixel đó sẽ được gán bằng 0.

Bước 5: Viết các hàm theo yêu cầu của đồ án.

1. ***Điều chỉnh độ sáng cho ảnh màu RGB***

- Hàm có tác dụng thay đổi độ sáng của ảnh.
- Tham số truyền vào gồm ma trận ảnh 1 chiều gồm các pixel màu và độ sáng muốn tăng/giảm (dưới dạng int).
- Với mỗi pixel màu là 1 mảng gồm 3 phần tử tương ứng 3 màu trong RGB, ta sẽ tịnh tiến giá trị của mỗi màu theo lượng sáng muốn tăng/giảm.
- Tuy nhiên, giá trị của mỗi màu sau khi tịnh tiến phải nằm trong khoảng [0, 255] nên chúng ta sẽ sử dụng hàm **truncate_pixel** để thực hiện điều đó.
- Đầu ra của hàm sẽ là ma trận các pixel sau khi đã được thay đổi giá trị độ sáng.

2. ***Điều chỉnh độ tương phản cho ảnh màu RGB***

- Hàm có tác dụng thay đổi độ tương phản của ảnh.

- Tham số truyền vào gồm ma trận ảnh 1 chiều gồm các pixel màu và độ tương phản muốn tăng/giảm (dưới dạng int).
- Đầu tiên, ta sẽ tạo 1 biến factor có giá trị như sau: factor = (259 * (contrast + 255)) / (255 * (259 - contrast)). Đây là giá trị tương phản.
- Sau đó, với mỗi giá trị trong mỗi pixel màu của ma trận ảnh, ta sẽ ánh xạ chúng qua 1 hàm như sau: factor * (color - 128) + 128 (với color là giá trị trong 3 màu RGB).
- Tuy nhiên, giá trị của mỗi màu sau khi ánh xạ phải nằm trong khoảng [0, 255] nên chúng ta sẽ sử dụng **hàm truncate_pixel** để thực hiện điều đó.
- Đầu ra của hàm sẽ là ma trận các pixel sau khi đã được thay đổi giá trị tương phản.

3. ***Lật ảnh ngang/dọc***

- Hàm có tác dụng lật ảnh ngang/dọc.
- Cách thực hiện hàm sẽ không quá cầu kì, 2 các lật ảnh ngang và dọc đều có nguyên lí như nhau: **vertical** để lật ảnh qua trục dọc còn **horizontal** để lật ảnh qua trục ngang.
- Tham số truyền vào sẽ là ma trận ảnh 2 chiều gồm các pixel màu, chiều cao ảnh, chiều rộng ảnh và cách lật ảnh (ngang/dọc)
- Với trục cho trước làm chuẩn, 2 pixel đối xứng nhau sẽ được đổi chỗ cho nhau (swap). Chúng ta sẽ làm như thế cho đến hết ma trận.
- Đầu ra của hàm là ma trận ảnh 2 chiều sau khi đã được lật ngang/dọc.

4. ***Chuyển ảnh màu RGB sang ảnh xám***

- Hàm có tác dụng chuyển ảnh màu RGB sang ảnh xám.
- Tham số truyền vào là ma trận ảnh 1 chiều gồm các pixel màu.
- Mỗi pixel của ảnh xám là 1 giá trị nằm trong khoảng [0, 255], còn mỗi pixel ảnh màu RGB là 1 mảng gồm 3 phần tử tưởng ứng với 3 màu red, green và blue.
- Mỗi giá trị pixel ảnh xám sẽ được định nghĩa như sau: xám = 0.3 * đỏ + 0.59 * xanh lá + 0.11 * xanh dương.
- Theo quy tắc trên, ta sẽ quét lần lượt từng pixel màu RGB và chuyển chúng về thành 1 giá trị pixel ảnh xám.
- Đầu ra của hàm là ma trận các pixel được chuyển từ ảnh màu RGB sang ảnh xám.

5. ***Ghép 2 ảnh xám có cùng kích cỡ***

- Hàm có tác dụng ghép 2 ảnh xám thành 1 (với cùng kích cỡ).
- Tham số truyền vào là 2 ma trận 1 chiều của 2 ảnh xám sẽ được ghép với nhau. Với cùng kích cỡ, cả 2 ma trận ảnh sẽ có cùng số lượng pixel.
- Chính vì vậy ta sẽ lần lượt lấy trung bình cộng giá trị của từng cặp pixel và làm như vậy cho đến hết ma trận.
- Đầu ra của hàm là 1 ma trận ảnh xám mới sau khi ghép 2 ma trận với mỗi pixel là giá trị trung bình của 2 pixel tương ứng.

6. ***Làm mờ ảnh RGB (Gaussian blur)***

- Hàm có tác dụng làm mờ ảnh RGB (theo phương pháp Gaussian blur).
- Tham số truyền vào là ma trận ảnh 2 chiều các pixel màu.
- Về nguyên lý làm mờ ảnh theo thuật toán Gaussian blur, cứ mỗi 9 pixel trong ma trận vuông ta sẽ có 1 bảng tỉ lệ tập trung màu như sau:

1 2 1
2 4 2
1 2 1

- Như biểu đồ trên, ma trận vuông 3x3 này sẽ tập trung màu nhiều nhất ở vùng trung tâm và thưa màu nhất ở 4 góc.
- Chính vì vậy, cứ mỗi 9 pixel như vậy ta sẽ lấy tổng các giá trị của mỗi pixel với tỉ lệ tương ứng với pixel đó.
- Tổng của kết quả trên chia cho 16 (1 + 2 + 1 + 2 + 4 + 2 + 1 + 2 + 1 = 16), như thế sẽ tạo ra 1 pixel mới từ 9 pixel trên.
- Hàm sẽ thực hiện tương tự vậy cho đến hết ma trận, đồng nghĩa với việc ta sẽ rút gọn 1 số lượng pixel nhất định.
- Đầu ra của hàm là ma trận mới sau khi được làm mờ.

7. ***Cắt ảnh màu RGB (circle/eclipse)***

- Hàm có tác dụng cắt ảnh màu RGB (hình tròn hoặc 2 hình elip chéo nhau).
- Tham số truyền vào là ma trận ảnh 1 chiều các pixel màu và hình dạng để crop (circle/eclipse).
- Đầu tiên, ta phải biết được phương trình tổng quát của đường tròn và đường eclipse trong mặt phẳng như sau:
  + *Phương trình đường tròn*: (x-a)^2 + (x-b)^2 - R^2 = 0
  + *Phương trình đường eclipse*: ((x.cos(a) + y.sin(a))^2 / a^2) + ((x.cos(a) - y.sin(a)^2 / b^2) - 1 = 0
- Từ 2 phương trình trên, chúng ta sẽ quét lần lượt các pixel ảnh, ta sẽ thực hiện cho đến hết ma trận.
- Ở vị trí pixel có phương trình > 0, điều đó có nghĩa là pixel đó nằm ngoài và cần được crop, chúng ta chỉ việc set pixel đó về màu đen (tức là R = 0, G = 0, B = 0).
- Đầu ra của hàm là ma trận ảnh sau khi đã được cắt theo hình dạng mong muốn (circle/eclipse).

Bước 6: Viết hàm main cho người dùng lựa chọn chức năng muốn sử dụng.

- Quy tắc chung của mỗi hàm phục vụ từng yêu cầu sẽ bao gồm 3 bước:
  + Cho người dùng nhập một hoặc nhiều file ảnh và sẽ có hàm import những tấm ảnh được chọn.
  + Ảnh hoặc các ảnh sau khi được import sẽ được xử lí và đưa và hàm thực hiện yêu cầu tương ứng.

+ Cuối cùng, kết quả sẽ được in ra màn hình cho người dùng và lưu lại dưới dạng file png/pdf.

- Hàm main sẽ cho người dùng chọn lựa yêu cầu mà họ muốn thực hiện, sau khi chọn 1 yêu cầu, hàm xử lí yêu cầu đó sẽ được thực hiện.
- Ngoài ra, chúng ta còn có thể chạy riêng từng yêu cầu bằng cách gọi từng hàm (đã bao gồm trong phần source code).

IV) Phân tích và thống kê

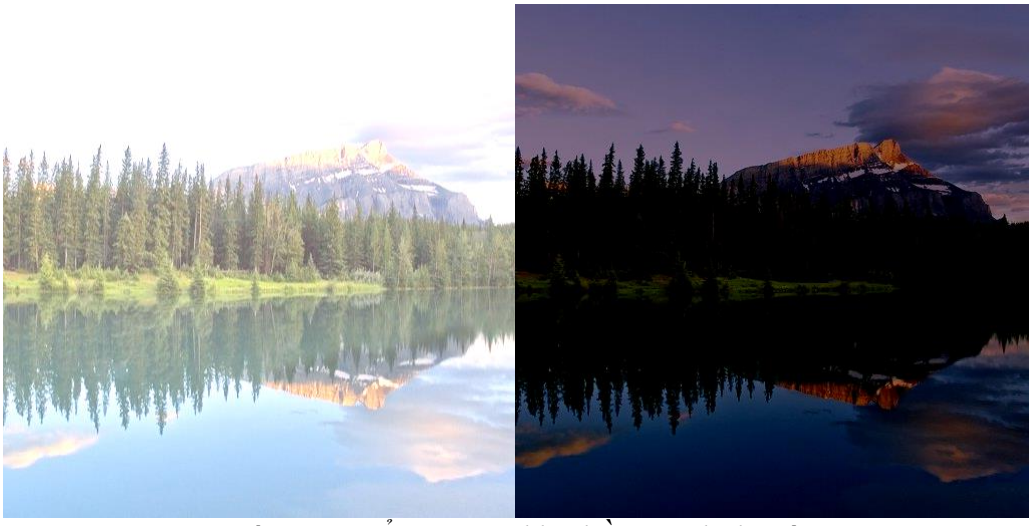
1. Thử nghiệm và kết quả

- Ảnh gốc thử nghiệm:

(Hình 0: Ảnh ban đầu)

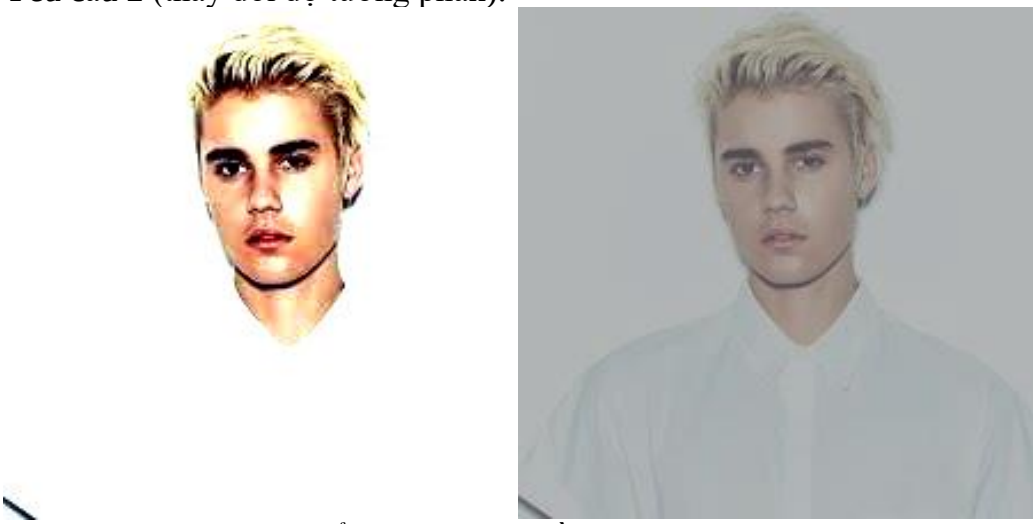
- Yêu cầu 1 (thay đổi độ sáng):

(Hình 1.1: Ảnh 1 sau khi điều chỉnh độ sáng)

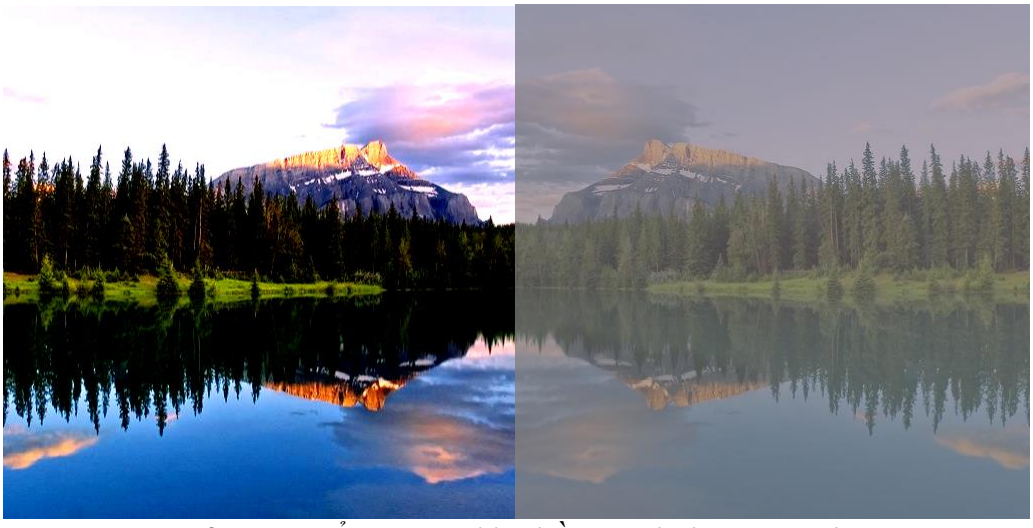


(Hình 1.2: Ảnh 2 sau khi điều chỉnh độ sáng)

- Yêu cầu 2 (thay đổi độ tương phản):



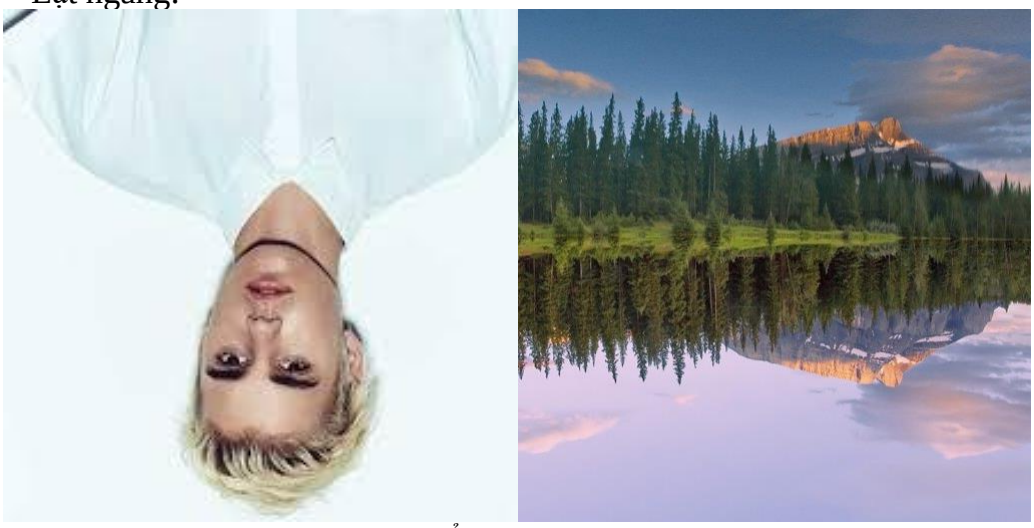
(Hình 2.1: Ảnh 1 sau khi điều chỉnh độ tương phản)



(Hình 2.2: Ảnh 2 sau khi điều chỉnh độ tương phản)

- Yêu cầu 3 (Lật ảnh ngang/dọc):

  + Lật ngang:

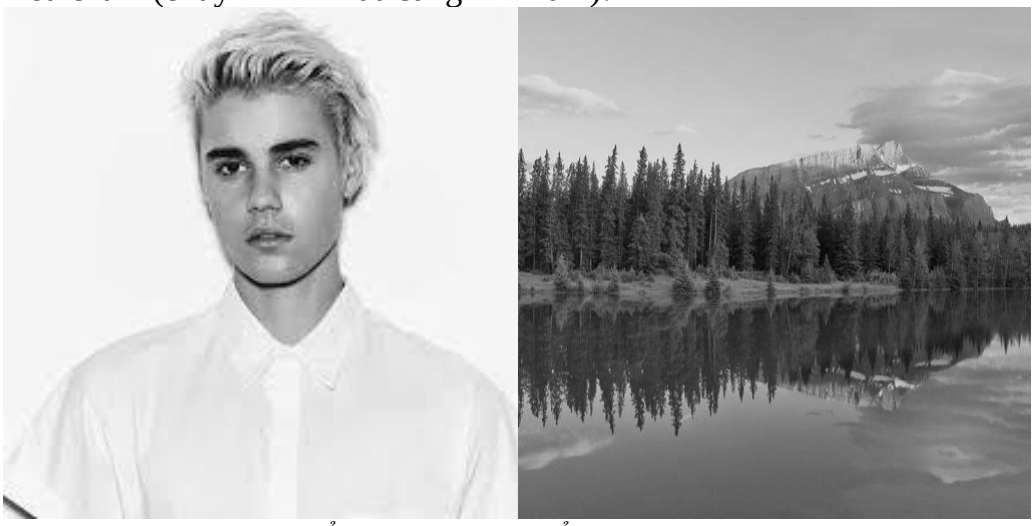


(Hình 4: Ảnh sau khi lật ngang)

  + Lật dọc:

(Hình 5: Ảnh sau khi lật dọc)

- Yêu cầu 4 (chuyển ảnh màu sang ảnh xám):



(Hình 6: Ảnh sau khi chuyển sang dạng ảnh xám)

- Yêu cầu 5 (ghép 2 ảnh xám cùng kích cỡ):

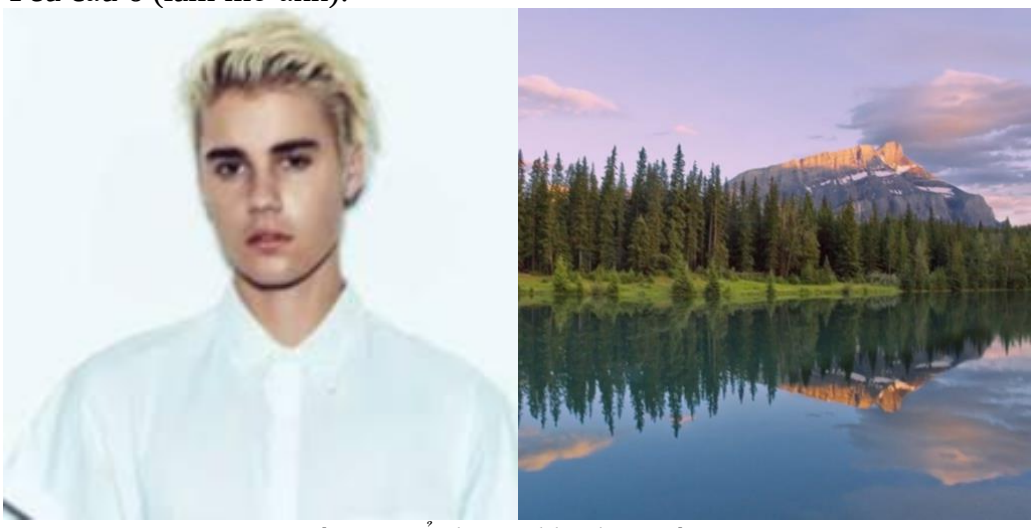
  + Ảnh gốc:

(Hình 7: Ảnh ban đầu có cùng kích thước)

+ Kết quả:

(Hình 8: Ảnh sau khi được ghép lại với nhau)

- Yêu cầu 6 (làm mờ ảnh):



(Hình 9: Ảnh sau khi được làm mờ)

- Yêu cầu nâng cao (cắt ảnh):

+ Cắt ảnh hình tròn:

(Hình 10: Ảnh sau khi cắt theo hình tròn)

+ Cắt ảnh 2 hình elip chéo nhau:

(Hình 11: Ảnh sau khi cắt theo 2 hình elip chéo nhau)

2. Phân tích và nhận xét

- Nhận xét chung: Qua việc sử dụng đúng phương pháp và áp dụng thuật toán tối ưu, hạn chế các vòng lặp, đa số các chức năng theo yêu cầu đồ án đều cho ra kết quả ảnh rất nhanh, hầu như các yêu cầu đều chạy dưới 10 giây với kích thước ảnh 512 x 512.
- Ở yêu cầu 5, ghép 2 ảnh xám có cùng kích thước, với mẫu thử nghiệm với kích thước 1920 x 1200, hàm chạy khoảng 35-40 giây, điều này càng cho thấy các hàm đều được tối ưu và có tốc độ chạy rất nhanh.
- Với hàm được cho là sẽ chạy lâu nhất, hàm làm mờ ảnh bằng Gaussian blur, cũng cho kết quả chạy không tồi, với thời gian chạy 15-17 giây kích thước ảnh 512 x

512, ngoài ra, so với hướng dẫn gốc của thuật toán, thay vì dùng 4 vòng lặp, em đã tinh chỉnh chỉ còn 2 vòng lặp, giúp tăng tốc độ chạy thuật toán.

- Ở yêu cầu nâng cao, không chỉ hình tròn và elip, nếu chúng ta biết thêm 1 số phương trình hình học khác, ta còn có thể cắt ảnh theo nhiều kiểu thú vị khác đa dạng và phong phú hơn.

V) Kết luận

Tóm gọn lại, đồ án 2 (xử lí ảnh) là 1 project vô cùng thú vị, áp dụng thực tế ma trận số học của toán vào các tấm ảnh tồn tại dưới dạng các pixel mà chúng ta thường thấy hằng ngày. Đồ án này giúp chúng ta thấy được một số thuật toán thú vị, đặc biệt là phương pháp Gaussian blur giúp làm mờ ảnh hay biết được cấu tạo của 1 giá trị ảnh xám đến từ sự kết hợp hài hoà giữa các tỉ lệ màu RGB. Thông qua đó, chúng ta sẽ có cái nhìn tổng quát hơn về toán học cũng như ứng dụng của nó trong việc lập trình, cụ thể là xử lý ảnh, điều đó dẫn đến sự phát triển của ngành thị giác máy tính trong xã hội hiện nay.

VI) Nguồn tham khảo

- **Algorithm for adjusting brightness and contrast of an RGB image:** **algorithms-for-adjusting-brightness-and-contrast-of-an-image**
- **General equation of a circle:** **khanacademy.org/circle-equation-review**
- **General equation of an ellipse:** **maa.org/EllipseGeneralEquation**
- **RGB to grayscale conversion:** **tutorialspoint.com/grayscale-to-rgb-conversion**
- **Gaussian blur:** **wikipedia.org/Gaussian-blur**