

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN  
ĐẠI HỌC QUỐC GIA TP.HCM**



**BÁO CÁO ĐỒ ÁN  
TOÁN ỨNG DỤNG VÀ THỐNG KÊ CHO  
CÔNG NGHỆ THÔNG TIN**

**Họ và tên sinh viên:** Nguyễn Thuận Phát

**MSSV:** 21127665

**Giảng viên lý thuyết:** Vũ Quốc Hoàng

**Giảng viên thực hành:**

- Phan Thị Phương Uyên
- Nguyễn Văn Quang Huy

## Mục lục

1. Mức độ hoàn thành .....	2
2. Ý tưởng & mô tả các hàm .....	2
2.1 Hàm loadImage .....	2
2.2 Hàm saveImage .....	2
2.3 Hàm changeBrightness .....	2
2.4 Hàm changeContrast .....	3
2.5 Hàm flip .....	3
2.6 Hàm convertGS .....	4
2.7 Hàm blur .....	4
2.8 Hàm sharpen .....	5
2.9 Hàm crop .....	5
2.10 Hàm circle .....	6
2.11 Hàm main .....	7
3. Hình ảnh kết quả .....	7
4. Tham khảo .....	9

## 1. Mức độ hoàn thành

STT	Tên chức năng	Tiến độ
1	Thay đổi độ sáng	Hoàn thành
2	Thay đổi độ tương phản	Hoàn thành
3	Lật ảnh ngang – dọc	Hoàn thành
4	Chuyển đổi ảnh RGB thành ảnh xám/sepia	Hoàn thành
5	Làm mờ/sắc nét ảnh	Hoàn thành
6	Cắt ảnh theo kích thước (cắt ở trung tâm)	Hoàn thành
7	Cắt ảnh theo khung hình tròn	Hoàn thành
8	Viết hàm main xử lí	Hoàn thành

## 2. Ý tưởng & mô tả các hàm

### 2.1 Hàm loadImage

-Chức năng: hàm mở ảnh và chuyển thành numpy array rồi trả về.

-Hàm có 1 tham số là tên/đường dẫn của ảnh cần mở.

### 2.2 Hàm saveImage

-Chức năng: hàm lưu ảnh sau khi đã xử lí xong, ảnh sẽ được lưu chung thư mục với ảnh gốc.

-Hàm có 3 tham số: fileName là tên/đường dẫn của ảnh gốc, image là 1 numpy array chứa ảnh cần lưu, method là chức năng đã thực hiện với ảnh gốc.

-Hàm sẽ tìm vị trí của dấu “.” cuối cùng (trước định dạng file như png, jpg) và tách chuỗi fileName ra tại vị trí đó và chèn vào tên phương thức. Sau đó hàm sử dụng save của thư viện PIL để lưu ảnh.

### 2.3 Hàm changeBrightness

-Chức năng: là hàm xử lí cho chức năng thay đổi độ sáng của chương trình.

-Ý tưởng thực hiện: cộng/trừ ba giá trị RGB của mỗi điểm ảnh trong ảnh với một lượng alpha để tăng/giảm độ sáng của ảnh.

-Hàm có 2 tham số: data là ảnh cần xử lí chứa trong 1 numpy array, alpha là giá trị để cộng vào 3 thuộc tính RGB của mỗi điểm ảnh(nếu muốn giảm độ sáng thì để giá trị của alpha âm).

-Khi thực thi, hàm sẽ chuyển data thành mảng 1 chiều để dễ duyệt(có lưu lại chiều dài và chiều rộng). Chạy vòng lặp duyệt hết mảng data rồi với mỗi điểm ảnh trong data, ta cộng thêm một lượng alpha sau đó lưu vào mảng result. Trong lúc chạy vòng lặp nếu có giá trị nào đó của điểm ảnh sau khi cộng alpha vượt quá 255 thì sẽ set lại thành 255, tương tự nếu nhỏ hơn 0 sẽ set lại thành 0. Sau khi chạy xong vòng lặp, ta có mảng result là mảng một chiều chứa các điểm ảnh sau khi thay đổi độ sáng, nhờ vào chiều dài và chiều rộng lưu trước đó để chuyển lại thành ảnh như ban đầu rồi trả về.

## 2.4 Hàm changeContrast

-Chức năng: là hàm xử lý cho chức năng thay đổi độ tương phản của chương trình.

-Ý tưởng thực hiện: nhân ba giá trị RGB của mỗi điểm ảnh trong ảnh với một lượng alpha để tăng/giảm độ tương phản của ảnh.

-Hàm có 2 tham số: data là ảnh cần xử lý chứa trong 1 numpy array, alpha là hệ số để nhân vào 3 thuộc tính RGB của mỗi điểm ảnh(nếu muốn tăng tương phản thì  $\alpha > 1$ , giảm tương phản thì  $0 < \alpha < 1$ ).

-Khi thực thi, hàm sẽ chuyển data thành mảng 1 chiều để dễ duyệt(có lưu lại chiều dài và chiều rộng). Chạy vòng lặp duyệt hết mảng data rồi với mỗi điểm ảnh trong data, ta nhân thêm một lượng alpha sau đó lưu vào mảng result. Trong lúc chạy vòng lặp nếu có giá trị nào đó của điểm ảnh sau khi nhân alpha vượt quá 255 thì sẽ set lại thành 255. Sau khi chạy xong vòng lặp, ta có mảng result là mảng một chiều chứa các điểm ảnh sau khi thay đổi độ tương phản, nhờ vào chiều dài và chiều rộng lưu trước đó để chuyển lại thành ảnh như ban đầu rồi trả về.

## 2.5 Hàm flip

-Chức năng: là hàm xử lý chức năng lật ảnh ngang – dọc của chương trình.

-Ý tưởng thực hiện:

- Dọc: đảo ngược thứ tự các dòng trong mảng ảnh gốc, vậy là ta thu được mảng kết quả là ảnh đã lật theo chiều dọc.
- Ngang: đảo ngược thứ tự các cột trong mảng ảnh gốc, vậy là ta thu được mảng kết quả là ảnh đã lật theo chiều ngang.

-Hàm có 2 tham số: data là ảnh cần xử lý dưới dạng numpy array, direction là chiều lật(1 là lật dọc, 2 là lật ngang).

-Đối với lật dọc: hàm duyệt từng dòng trong mảng data từ trên xuống rồi chèn vào đầu mảng result.

-Đối với lật ngang: hàm duyệt các pixel từ trên xuống dưới từ trái qua phải. Các pixel trên cùng 1 dòng sẽ chèn vào mảng tạm row. Mỗi pixel khi được duyệt sẽ chèn vào vị trí  $\text{index} = 0$  của mảng row. Sau khi duyệt hết 1 dòng thì append mảng row vào mảng result.

## 2.6 Hàm convertGS

-Chức năng: là hàm xử lý chức năng chuyển đổi ảnh RGB thành ảnh xám/sepia.

-Ý tưởng thực hiện:

- Chuyển thành ảnh xám: chuyển từ dùng 3 giá trị để biểu diễn màu điểm ảnh sang dùng 1 giá trị để biểu diễn màu điểm ảnh, giá trị này là trung bình cộng của ba giá trị màu.
- Chuyển sang sepia: 3 giá trị màu của mỗi điểm ảnh trong ảnh gốc đổi thành giá trị mới, ba giá trị này được tính theo ba giá trị cũ. Công thức như sau:

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

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

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

với  $tr, tg, tb$  là 3 giá trị mới,  $R, G, B$  là 3 giá trị cũ

-Hàm có 2 tham số: data là ảnh cần xử lý dưới dạng numpy array, type là kiểu ảnh muốn chuyển thành(1 là chuyển thành ảnh xám, 2 là chuyển thành ảnh sepia).

-Đối với chuyển thành ảnh xám: hàm duyệt các pixel từ trên xuống dưới từ trái qua phải. Các pixel trên cùng 1 dòng sẽ được tính giá trị trung bình và giá trị này được chèn vào mảng tạm row. Sau khi duyệt hết 1 dòng thì append mảng row vào mảng result.

-Đối với chuyển thành ảnh sepia: hàm duyệt các pixel từ trên xuống dưới từ trái qua phải. Các pixel trên cùng sau khi tính giá trị mới sẽ được chèn vào mảng tạm row. Sau khi duyệt hết 1 dòng thì append mảng row vào mảng result.

## 2.7 Hàm blur

-Chức năng: là hàm xử lý chức năng làm mờ ảnh.

-Ý tưởng thực hiện: mỗi điểm ảnh trong ảnh gốc sẽ nhận giá trị mới, giá trị này bằng trung bình cộng của các điểm ảnh xung quanh và chính điểm ảnh đó.

-Hàm có 1 tham số : data là ảnh cần xử lý dưới dạng numpy array.

-Khi thực thi, hàm sao chép lại mảng data cho vào mảng copy. Tạo thêm “viền” các pixel [0, 0, 0] cho mảng copy. Sau khi thực hiện xong, mảng copy có hình dạng như sau:

[0, 0, 0]	[0, 0, 0]	[0, 0, 0]	[0, 0, 0]	[0, 0, 0]	[0, 0, 0]
[0, 0, 0]	[85, 2, 12]	[85, 2, 12]	[255, 255, 255]	[22, 25, 255]	[0, 0, 0]
[0, 0, 0]	[22, 25, 255]	[255, 65, 15]	[61, 27, 187]	[85, 2, 12]	[0, 0, 0]
[0, 0, 0]	[255, 255, 255]	[255, 255, 255]	[255, 255, 255]	[255, 255, 255]	[0, 0, 0]
[0, 0, 0]	[85, 2, 12]	[61, 27, 187]	[61, 27, 187]	[255, 255, 255]	[0, 0, 0]
[0, 0, 0]	[255, 65, 15]	[255, 255, 255]	[255, 255, 255]	[22, 25, 255]	[0, 0, 0]
[0, 0, 0]	[255, 255, 255]	[22, 25, 255]	[61, 27, 187]	[22, 25, 255]	[0, 0, 0]
[0, 0, 0]	[61, 27, 187]	[255, 255, 255]	[255, 255, 255]	[255, 65, 15]	[0, 0, 0]
[0, 0, 0]	[85, 2, 12]	[255, 255, 255]	[85, 2, 12]	[255, 255, 255]	[0, 0, 0]
[0, 0, 0]	[0, 0, 0]	[0, 0, 0]	[0, 0, 0]	[0, 0, 0]	[0, 0, 0]

Các pixel màu xanh dương là viền mới được thêm  
 Các pixel màu còn lại là pixel trong ảnh gốc

Hình 1 Mô tả mảng copy sau khi thêm viền

-Vì các pixel ở góc và cạnh viền chỉ có lần lượt 3 và 5 pixel xung quanh nên việc thêm viền như vậy là để các pixel ở các góc và cạnh viền của ảnh vẫn có đủ 8 pixel xung quanh như các pixel ở giữa để lấy ra tính toán. Sau đó duyệt lần lượt từng pixel của ảnh gốc trong mảng copy(màu nền khác xanh dương), lấy 8 giá trị của pixel xung quanh rồi tính trung bình. Khi tính trung bình, giá trị của pixel sau khi cộng với các giá trị của các pixel xung quanh sẽ chia cho một số n khác nhau(thêm viền để khi lấy giá trị của 8 pixel xung quanh không bị lỗi vượt quá giới hạn mảng đối với các pixel ngoài rìa), n là 9 nếu pixel không nằm ở cạnh viền hoặc góc (màu nền trắng ở ảnh trên), n là 6 nếu pixel đó nằm ở cạnh viền(màu nền vàng ở ảnh trên), n là 4 nếu pixel đó nằm ở đỉnh góc(màu nền xanh lá ở ảnh trên). Pixel sau khi tính giá trị mới sẽ thêm vào mảng result rồi trả về.

## 2.8 Hàm sharpen

-Chức năng: là hàm xử lý chức năng làm sắc nét ảnh.

-Ý tưởng thực hiện: mỗi điểm ảnh trong ảnh gốc sẽ nhận giá trị mới, giá trị này bằng giá trị của chính điểm ảnh đó nhân 5 trừ cho giá trị của các pixel trên, dưới, trái, phải.

-Hàm có 1 tham số : data là ảnh cần xử lý dưới dạng numpy array.

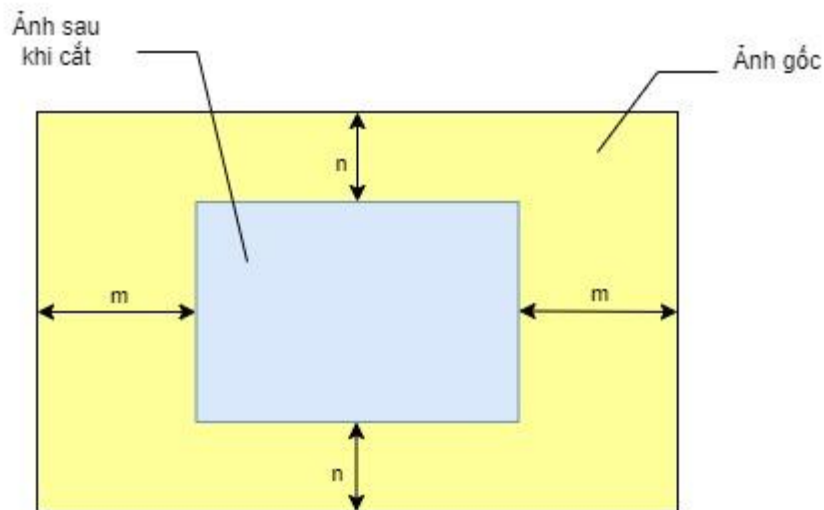
-Khi thực thi, hàm cũng sao chép lại mảng data cho vào mảng copy và thêm viền các pixel [0, 0, 0] giống như hàm blur với mục đích tương tự. Hàm duyệt từng pixel của ảnh gốc trong mảng copy rồi tính giá trị mới theo công thức đã ghi ở trên rồi lưu vào mảng result rồi trả về.

## 2.9 Hàm crop

-Chức năng: là hàm xử lý chức năng cắt ảnh ở trung tâm cho chương trình.

-Ý tưởng thực hiện: ảnh sau khi cắt sẽ nhỏ hơn ảnh gốc, tính khoảng chênh lệch n, m (đơn vị pixels) giữa các kích thước mới và cũ, khi chạy vòng for cộng thêm các khoảng này vào sẽ ra chính xác vị trí của các pixel của ảnh sau khi cắt.

-Hàm có 3 tham số: data là ảnh cần xử lý dưới dạng numpy array, height là chiều dọc của ảnh sau khi cắt, width là chiều ngang của ảnh sau khi cắt.



Hình 2 Minh họa ý tưởng

-Khi thực thi, đầu tiên hàm sẽ kiểm tra xem 2 kích thước của ảnh sau khi cắt (tạm gọi height theo chiều dọc, width theo chiều ngang) có hợp lệ hay không(phải nhỏ hơn hoặc bằng kích thước gốc), nếu không sẽ trả về thông báo không hợp lệ. Sau đó tính hai khoảng chênh lệch n (theo chiều dọc), m (theo chiều ngang) như trên ảnh minh họa với công thức như sau:

$$n = (\text{chiều dọc gốc} - \text{chiều dọc mới}) / 2$$

$$m = (\text{chiều ngang gốc} - \text{chiều ngang mới}) / 2$$

(làm tròn xuống)

-Sau khi tính hai khoảng chênh lệch ta chạy 2 vòng for lồng nhau để duyệt các pixel trong mảng data. Vòng for ở ngoài cho biến x chạy trong đoạn từ 0 cho đến height, vòng for ở trong cho biến y chạy từ đoạn từ 0 đến width. Với mỗi lần lặp của vòng for ở trong, ta lưu `data[x + n][y + m]`(đây chính xác là pixels của ảnh cần cắt) vào mảng result. Sau khi chạy xong vòng lặp ta được mảng result là ảnh sau khi cắt.

## 2.10 Hàm circle

-Chức năng là: là hàm xử lý chức năng cắt ảnh hình tròn(đường kính hình tròn là chiều ngắn hơn của ảnh).

-Ý tưởng thực hiện: tính tọa độ tâm ảnh cũng là tâm hình tròn rồi tính khoảng cách đến tâm của từng điểm ảnh, nếu khoảng cách nhỏ hơn hoặc bằng bán kính thì thêm vào mảng kết quả rồi trả về.







-Hàm có 1 tham số là ảnh cần xử lý dưới dạng numpy array.

-Khi thực thi, hàm sẽ tính tọa độ tâm hình tròn bằng cách lấy 2 kích thước chia 2, tính bán kính bằng cách lấy kích thước nhỏ hơn chia 2. Chạy vòng for duyệt từng điểm ảnh trong mảng data, tính khoảng cách của điểm ảnh đến tâm, nếu nhỏ hơn hoặc bằng bán kính thì lưu điểm ảnh đó vào mảng result, còn nếu lớn hơn bán kính thì lưu điểm ảnh màu đen [0, 0, 0] vào mảng result (tạo thành vùng đen ngoài hình tròn). Sau đó hàm trả về mảng result là ảnh kết quả cần tìm.









### 2.11 Hàm main







-Chức năng: là hàm cho người dùng chọn lựa và thực thi các chức năng của chương trình

## 3. Hình ảnh kết quả

Tên chức năng	Ảnh gốc	Ảnh kết quả
Thay đổi độ sáng		
Thay đổi độ tương phản		
Lật ảnh ngang		



Lật ảnh dọc		
Chuyển đổi ảnh RGB thành ảnh xám		
Chuyển đổi ảnh RGB thành ảnh sepia		
Làm mờ ảnh		

Làm sắc nét ảnh		
Cắt ảnh theo kích thước (cắt ở trung tâm)		
Cắt ảnh theo khung hình tròn		

#### 4. Tham khảo

- pil.save(): <https://www.educative.io/answers/how-to-control-the-brightness-of-an-image-in-python>
- python str.split(): [https://www.w3schools.com/python/ref\\_string\\_split.asp](https://www.w3schools.com/python/ref_string_split.asp)
- numpy deep copy: <https://www.delftstack.com/howto/numpy/python-numpy-deep-copy/#:~:text=NumPy%20Deep%20Copy%20With%20the%20copy.,->

[deepcopy\(\)%20Function&text=A%20shallow%20copy%20means%20the,array%20into%20the%20copied%20array.](#)

- chuyển ảnh thành sepia: <https://dyclassroom.com/image-processing-project/how-to-convert-a-color-image-into-sepia-image>