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 học: Toán ứng dụng và thống kê

GIÁO VIÊN HƯỚNG DẪN

Vũ Quốc Hoàng
Nguyễn Văn Quang Huy
Lê Thanh Tùng
Phan Thị Phương Uyên

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 học: Toán ứng dụng và thống kê

SINH VIÊN THỰC HIỆN

20127662 - Nguyễn Đình Văn

MỤC LỤC

Mục lục

THÔNG TIN THÀNH VIÊN

Mã số sinh viên	Họ và tên	Chú thích
20127662	Nguyễn Đình Văn	20127662@student.hcmus.edu.vn

CÁC CHỨC NĂNG ĐÃ HOÀN THÀNH

STT	Tên chức năng	Mức độ hoàn thành	Ghi chú
1	Thay đổi độ sáng cho ảnh	100%	
2	Thay đổi độ tương phản	100%	
3	Lật ảnh (ngang - dọc)	100%	
4	Chuyển đổi ảnh RGB thành ảnh xám	100%	
5	Chồng 2 ảnh cùng kích thước	100%	
6	Làm mờ ảnh	100%	
7	Viết hàm main xử lí	100%	
8	Khung hình tròn	100%	
9	Khung hình elip	90%	Đã in được khung elip nhưng hình ảnh có chất lượng kém so với ảnh gốc.

CÁC HÀM TRONG CHƯƠNG TRÌNH

1. Hàm thay đổi độ sáng – brightness (img, dosang)

- Ý tưởng: Chúng ta làm tăng giá trị của mỗi Pixel vì giá trị Pixel càng hướng tới 255 thì càng sáng.
- Input: img(Ma trận điểm ảnh) , dosang(Độ sáng)
- Output: Ma trận các điểm ảnh sau khi tăng Pixel
- Mô tả: Nhận ma trận điểm ảnh sau đó biến đổi thành ma trận 2D, tạo ma trận số tương ứng với ma trận ảnh vừa tạo, tiến hành cộng hai ma trận ta thu được ma trận điểm ảnh với độ sáng tăng (Lưu ý kết quả sau khi cộng nếu < 0 thì trả về 0, nếu > 255 thì trả về 255 - ở đây em dùng hàm np.clip() để đảm bảo điều kiên này). Sau đó ta thu được ma trận các điểm ảnh đã tăng độ sáng.

2. Hàm thay đổi độ tương phản – constrast (img, dotuongphan)

- Ý tưởng: Sử dụng công thức tăng độ tương phản *[1]* $F = \frac{259\,(C+255)}{255\,(259-C)}$ (Với C là độ tương phản) và $R, G, B = F(R, G, B - 128) + 128$ từ đó tính được hệ màu mới.
- Input: img (Ma trận điểm ảnh), dotuongphan (Độ tương phản)
- Output: Ma trận điểm ảnh sau khi tăng độ tương phản.
- Mô tả: Nhận vào ma trận điểm ảnh sau đó với mỗi Pixel ta tính công thức trên thu được điểm ảnh mới gán vào ảnh. Cứ lặp lại và ta thu được ma trận điểm ảnh tương phản.

3. Hàm lật ảnh (ngang – dọc) – flip(img, type)

- Ý tưởng: Sử dụng hàm trong numpy để tiến hành lật các cột/ hàng theo ý mình muốn. (np.flipud và np.fliplr) *[2]*.
- Input: img(Ma trận điểm ảnh), Type (Dựa vào type để xác định lật ảnh theo chiều ngang hoặc dọc).
- Output: Ma trận điểm sau khi lật.
- Mô tả: Nhận ma trận điểm ảnh và type sau đó dựa vào type xác định lật ảnh theo chiều ngang hoặc dọc. Sau khi lật ảnh thành công thì trả về ma trận ảnh đã lật.

4. Hàm chuyển đổi ảnh RGB thành ảnh xám – grayscale(img)

- Ý tưởng: Sử dụng các tính trung bình điểm ảnh để đưa về ảnh xám*[1]*.
- Input: img (Ma trận điểm ảnh)
- Output: Ma trận điểm ảnh sau khi tính trung bình mỗi điểm ảnh
- Mô tả: Nhận vào ma trận điểm ảnh, sau đó tính giá trị trung bình thu được ma trận trung bình. Sử dụng hàm concatenate của numpy nối thêm 2 lần tha thu được ma trận mỗi điểm ảnh đều bằng giá trị trung bình.

5. Hàm chồng hai ảnh cùng kích thước – concatenate(img1, img2)

- Ý tưởng: Chuyển hai ảnh về ảnh xám để đảm bảo điều kiện, sau đó cộng hai ma trận lại với nhau.
- Input: img1(Ma trận ảnh 1), img2(Ma trận ảnh 2).
- Output: Ma trận ảnh sau khi cộng img1 và img2
- Mô tả: Nhận vào ma trận 2 ảnh, sau đó ta chuyển hai ảnh về ảnh xám và cộng 2 ma trận lại với nhau thu được ảnh chồng 2 ảnh trên.

6. Hàm làm mờ ảnh – blur (img)

- Ý tưởng: Dựa vào thuật toán BoxBur *[3]*
- Input: img(Ma trận điểm ảnh)
- Output: Ma trận đã làm mờ.
- Mô tả: Nhận vào ma trận điểm ảnh. Theo thuật toán BoxBur thì ta nhận mỗi ô là ô trung tâm từ đó tính tổng xung quanh và nhân 1/9. Nhưng em thấy với việc làm vậy sẽ giảm số lượng màu cơ bản của ảnh nên em thực hiện việc phân cụm, mỗi cụm 9 ô, tính tổng 9 ô và gán lại giá trị này vào ô trung tâm và việc *1/9 sẽ làm giảm độ sáng của bức ảnh nên em bỏ qua bước này. Kết quả thu được ma trận các điểm ảnh đã giảm chất lượng.

7. Hàm main xử lí – main()

- Ý tưởng: Người dùng nhập vào tên ảnh, chuyển ảnh về ma trận và xử lí
- Input: Tên ảnh được xử lí
- Output: Ảnh đã được xử lí lưu về với tên và chức năng tương ứng
    - Ảnh tăng độ sáng: *image_brightness.png*
    - Ảnh tăng độ tương phản: *image_contrast.png*
    - Ảnh lật ngang: *image_flip_horizontal.png*
    - Ảnh lật dọc: *image_flip_vertical.png*
    - Ảnh xám: *image_grayscale.png*
    - Ảnh mờ: *image_blur.png*
    - Ảnh chồng 2 ảnh: *image_concatenate.png*
    - Ảnh khung tròn: *image_roundFrame.png*
    - Ảnh khung elip: *image_elipFrame.png*
- Mô tả: Người dùng nhập vào tên ảnh. In ra màn hình những lựa chọn cho người dùng(choose). Sau khi người dùng đã lựa chọn thì tiến hành thực hiện theo lưa chọn của người dùng. Xử lý hình ảnh sau đó xuất ra màn hình và lưu ảnh đã xử lí với tên tương ứng.

8. Hàm tạo khung hình tròn – roundFrame(img)

- Ý tưởng: Dựa vào phương trình hình tròn ($(x-a)^2-(y-b)^2=R^2$), tính toán các pixel nằm ngoài và trong đường tròn.
- Input: img(Ma trận ảnh)
- Output: Ma trận ảnh đã xử lí.
- Mô tả: Xác định tâm của hình tròn là (hàng/2, cột/ 2). Sau đó ta xác định bán kính, nếu số hàng < số cột thì R là khoảng cách từ tâm đến điểm chính giữa của hàng cuối cùng, ngược lại thì R là khoảng cách từ tâm đến điểm chính giữa của cột cuối cùng. Sau khi xác định O và R thì duyệt vị trí từng Pixel nếu > R (nằm ngoài đường tròn) thì gán bằng 0 (Màu đen). Và ta thu được ảnh có khung tròn.

9. Hàm tạo khung hình elip - elipFrame(img)

- Ý tưởng: Dựa vào phương trình elip ($\frac{x^2}{a^2}+\frac{y^2}{b^2}=1$) ta quay hình elip lần lượt 45' và -45' để thu được các vị trí nằm trong 2 hình elip. Từ đó xác định được các pixel nằm trong hai hình elip này.
- Input: img(ma trận điểm ảnh)
- Output: Ma trận ảnh đã xử lí.
- Mô tả: Đầu tiên ta xác định tâm elip, giống với tâm hình tròn. Ta vẽ được hình elip trên ảnh (Trục lớn 5/8 * độ dài ½ đường chéo, trục bé * độ dài ½ đường chéo - tỉ lệ này em xác định do vẽ phác thảo với các ảnh hình vuông và hình chữ nhật). Sau khi xác định được trục bé (2b) và lớn (2a) ta vẽ được hình elip. Dùng phép quay $\begin{cases} x' = x\cos(45) - y\sin(45) \\ y' = x\sin(45) = y\cos(45) \end{cases}$ (Ta thu được tọa độ khi dời hình elip một góc 45 – Trong bài code giá trị (+/ -) 0.7071067812 là giá trị của biểu thức Sin/Cos (+/ - 45')). Quay hình elip góc 45' ta thu được các điểm thuộc elip chéo (\), và quay hình elip góc – 45' ta thu được các điểm thuộc elip chéo (/). Từ đó các điểm nằm ngoài các điểm đã thu được chính là nằm ngoài khung elip ta tiến hành gán lại các điểm đó bằng 0 (màu đen). Và ta thu được ảnh có khung elip.

HÌNH ẢNH VÀ KẾT QUẢ

Ta có ảnh gốc:

Sau khi chạy thực nghiệm thu được kết quả như sau:

- Ảnh tăng độ sáng: *image_brightness.png*

- Ảnh tăng độ tương phản: *image_contrast.png*

- Ảnh lật ngang: *image_flip_horizontal.png*

- Ảnh lật dọc: *image_flip_vertical.png*

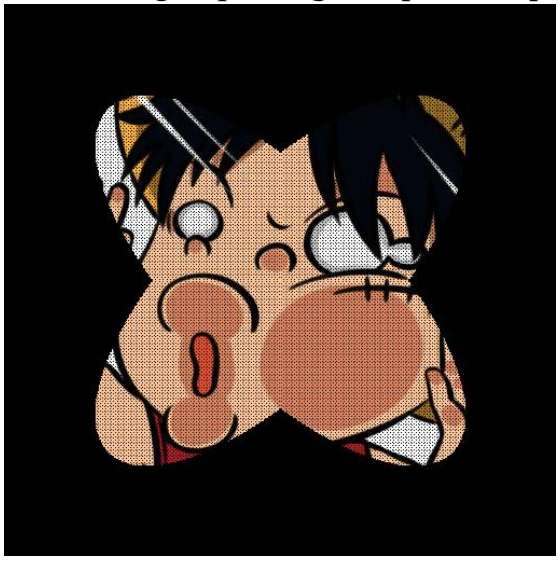
- Ảnh xám: *image_grayscale.png*

- Ảnh mờ: *image_blur.png*

- Ảnh chồng 2 ảnh: *image_concatenate.png*

- Ảnh khung tròn: *image_roundFrame.png*

- Ảnh khung elip: *image_elipFrame.png*



Nhận xét: Các yêu cầu xử lí ảnh đã hoàn thành được, đáp ứng hết các yêu cầu của đồ án. Tuy nhiên còn tính năng khung elip vì thuật toán chưa chặt chẽ nên hình ảnh chưa đều.

TÀI LIỆU THAM KHẢO

[1] https://koodibar.com/posts/xu-ly-hinh-anh-voi-python#3-thay-%C4%91%E1%BB%95i-%C4%91%E1%BB%99-s%C3%A1ng-brightness

[2] https://helpex.vn/question/cach-hieu-qua-nhat-de-dao-nguoc-mot-mang-numpy-6094a333f45eca37f4c03d3f

[3] Box blur - Wikipedia