TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

BÁO CÁO PROJECT 2

IMAGE PROCESSING

Sinh viên thực hiện

Họ và tên: Lê Mỹ Khánh Quỳnh
Mssv: 21127681
Lớp: 21CLC01

HỌC PHẦN: TOÁN ỨNG DỤNG VÀ THỐNG KÊ CHO CÔNG NGHỆ THÔNG TIN

MỤC LỤC

A. ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH

STT	Công việc	Mức độ hoàn thành
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 / sepia	100%
5	Làm mờ / sắc nét ảnh	100%
6	Cắt ảnh theo kích thước (ở trung tâm)	100%
7	Cắt ảnh theo khung hình tròn	100%
8	Cắt ảnh theo khung 2 elip chép	100%
9	Cài đặt hàm main theo yêu cầu	100%
10	Đảm bảo thời gian chạy các chức năng trong mức chấp nhận được	100%

B. MÔ TẢ CÁC CHỨC NĂNG, KẾT QUẢ THỬ NGHIỆM VÀ NHẬN XÉT

Hình anh 1, hình ảnh gốc thử nghiệm(512 x 512)

1. Thay đổi độ sáng cho ảnh

❖ Function: ***change_brightness(img,brigtness)***

- **Tham số đầu vào*:*** data array của ảnh, mức thay đổi độ sáng.

- **Ý tưởng:** thay đổi độ sáng cho ảnh bằng cách tăng/giảm giá trị các phần tử trong kênh màu.
- **Mô tả**: Cộng từng phần tử trong data array của ảnh gốc với tham số mức thay đổi độ sáng truyền vào (*brightness*), tham số có thể là số dương (tăng độ sáng) hoặc số âm (giảm độ sáng). Sau đó, các phần tử được clip lại trong khoảng [0,255] và trả về.
- **Kết quả:**

Hinh anh 2, tăng độ sáng +100

Hinh anh 3, giảm độ sáng -100

- **Nhận xét:** Đơn giản, dễ cài đặt, hiệu quả đem lại ổn định, phủ đều lên toàn bộ ảnh. Tuy nhiên độ sáng quá cao hoặc quá thấp có thể gây mất mát chi tiết và không phản ánh đủ thông tin của ảnh, cần lưu ý khi lựa chọn thông số tăng giảm.

2. Thay đổi độ tương phản cho ảnh

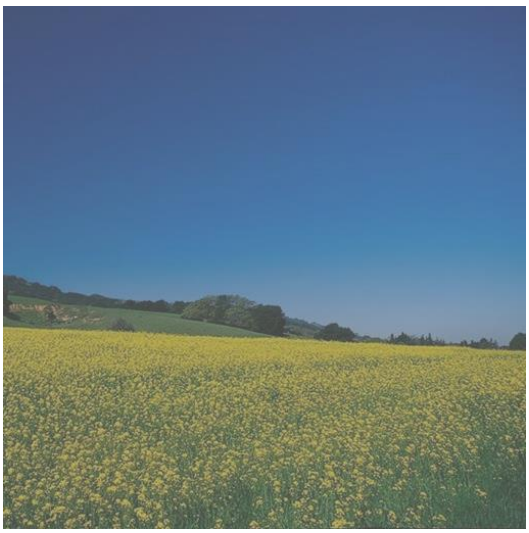
❖ Function: ***change_contrast(img,contrast)***

- **Tham số đầu vào:** data array của ảnh, mức tương phản mong muốn
- **Ý tưởng:** Thay đổi độ tương phản bằng cách sử dụng công thức hiệu chỉnh thông số tương phản.
- **Mô tả:** Với tham số đầu vào về mức tương phản mong muốn của ảnh (contrast), hệ số tương phản được tính theo công thức:

$F = \frac{259(contrast+255)}{255(259-contrast)}$. Dựa vào hệ số tương phản F vừa tính được, độ tương phản sẽ được hiệu chỉnh theo công thức F(R/G/B - 128)+128. Cuối cùng, các giá trị được clip lại trong đoạn [0,255] và trả về.

- **Kết quả:**

Hinh anh 4, tăng độ tương phản lên giá trị 100



Hinh anh 5, giảm độ tương giảm xuống giá trị -100

- **Nhận xét:** Điều chỉnh độ tương phản của ảnh phủ đều lên toàn bộ ảnh, tuy nhiên nếu mức contrast quá cao hoặc quá thấp sẽ dễ gây mất chi tiết ảnh, cần lưu ý khi lựa chọn contrast.

3. Lật ảnh (ngang-dọc)

❖ Function: ***flip_image(img,type)***

- **Tham số đầu vào:** data array của ảnh, loại (ngang/dọc)
- **Ý tưởng:** lật các phần tử của data array của ảnh gốc theo trục để thu được hình ảnh lật theo yêu cầu.
- **Mô tả:** Sử dụng hàm np.flipud để lật các phần tử của mảng theo trục dọc và sử dụng hàm np.fliplr để lật các phần tử mảng theo trục ngang.
- **Kết quả:**

Hinh anh 6, lật dọc

Hinh anh 7, lật ngang

- **Nhận xét:** Do sử dụng cách lật các phần tử của mảng numpy theo trục nên có thể bảo toàn hình gốc khi biến đổi và hình ảnh thu được cũng đảm bảo phản ánh đủ và đúng thông tin, hoạt động nhanh chóng và hiệu quả.

4. Chuyển đổi ảnh RGB sang ảnh xám/sepia

❖ Function: ***rgb_to_grayscale(img)***

- **Tham số đầu vào:** data array của ảnh.
- **Ý tưởng:** nhân giá trị của từng phần tử trong kênh màu với hệ số theo công thức để thu được ảnh xám.
- **Mô tả:** Đối với từng giá trị R,G,B của mỗi pixel, ta nhân lần lượt với các hệ số (0.2989, 0.5870, 0.1140) để thu được ảnh xám.
- **Kết quả:**

Hinh anh 8, chuyển sang ảnh xám

- **Nhận xét:** Hình ảnh được chuyển đổi chính xác sang gam màu xám, hình ảnh xám tuy giúp tiết kiệm vùng nhớ do chỉ chứa 1 kênh màu so với RGB nhưng phản ánh thiếu thông tin màu sắc và chi tiết.

❖ Function: ***rgb_to_sepia(img)***

- **Tham số đầu vào:** data array của ảnh.
- **Ý tưởng:** Thay đổi giá trị từng phần tử trong kênh màu theo công thức để thu được hình ảnh sepia.
- **Mô tả:** Đối với từng giá trị R,G,B của mỗi pixel, ta tính toán lần lượt với công thức như sau:

R_new = 0.393*r+0.769*g+0.189*b
G_new = 0.349*r+0.686*g+0.168*b
B_new = 0.272*r+0.534*g+0.131*b

- **Kết quả:**

Hinh anh 9, chuyển sang ảnh sepia

- **Nhận xét:** Hình ảnh được chuyển đổi sang ảnh sepia nhanh chóng và hiệu quả. Ảnh sepia tạo hiệu ứng cổ điển và tăng tính nghệ thuật nhưng thiếu tính thực tế và không thể phản ánh thông tin đầy đủ như ảnh RGB.

5. Làm mờ / sắc nét ảnh

a. Làm mờ

- **Ý tưởng:** Sử dụng thuật toán box blur để làm mờ hình ảnh.
  ❖ Function: ***create_summed_area_table(img)***
- **Tham số đầu vào:** data array của ảnh.
- **Mô tả:** Bảng summed-area chứa các phần tử với giá trị của từng phần tử là tổng các giá trị từ phần tử đầu tiên đến phần tử hiện tại tương ứng của mảng tham số đầu vào. Để tính ra giá trị của phần tử tại vị trí x,y của summed-area table (sat), ta lấy giá trị tương ứng từ mảng gốc + giá trị phía trên nó trong sat + giá trị bên trái nó trong sat – giá trị nằm ở góc chéo trái trên nó trong sat.
  ❖ Function: ***box_blur(img,radius)***
- **Tham số đầu vào:** data array của ảnh, bán kính của kernel
- **Mô tả**: Thay đổi giá trị của từng phần tử bằng giá trị trung bình của các phần tử lân cận nó nằm trong kernel (kích thước kernel theo yêu cầu người dùng) để làm mờ ảnh.

- **Kết quả:**

Hinh anh 10, làm mờ

- **Nhận xét:** Thuật toán box blur dễ triển khai và hiệu quả, kết hợp với summed-area table để tối ưu hóa nên hoạt động nhanh chóng. Hình ảnh kết quả cho ra có hiệu ứng mờ phủ đều, thời gian ổn định. Tuy nhiên, hình ảnh mờ có thể gây mất chi tiết, nếu kernel có kích thước quá lớn thì hình ảnh bị làm mờ cao sẽ không thể phản ánh đủ và chính xác thông tin.

b. Làm sắc nét

- **Ý tưởng: :** sử dụng laplacian filter để làm sắc nét hình ảnh.
  ❖ Function: ***laplacian_sharpen(img):***
- **Tham số đầu vào:** data array của ảnh.
- **Mục đích:** Làm sắc nét ảnh bằng laplacian filter
- **Mô tả:** Tính chập từng phần tử của ảnh với kernel là laplacian filter
- **Kết quả:**

Hinh anh 11, làm sắc nét

- **Nhận xét:** Làm nổi bật các biên cạnh và chi tiết giúp hình ảnh trở nên sắc nét. Các đối tượng ảnh trở nên rõ ràng hơn, tuy nhiên việc dùng laplacian filter để tăng độ nét có thể gây nhiễu và làm nổi bật một vài đặc trưng không mong muốn.

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

❖ Function: ***crop_center(img,crop_height,crop_width)***

- **Tham số đầu vào:** data array của ảnh, chiều cao của ảnh sau cắt, chiều rộng ảnh sau cắt.
- **Ý tưởng:** Tạo mảng con chứa các pixel nằm trong kích thước ảnh muốn crop, tình từ vị trí trung tâm.
- **Mô tả:** Dựa vào kích thước (dài,rộng) từ tham số đầu vào, ảnh mới được cắt từ vùng trung tâm của ảnh gốc theo kích thước tương ứng bằng cách tạo mảng con chứa các pixel tương ứng.
- **Kết quả:**

Hình ảnh 12, cắt ở trung tâm với kích thước 250x300

- **Nhận xét:** Do sử dụng mảng con để sao chép các pixels trong kích thước cần thiết nên hình ảnh cắt vẫn đảm bảo đủ thông tin và chi tiết của ảnh gốc, không gây mất mát cho ảnh gốc sau khi cắt.

7. Cắt ảnh theo khung hình tròn

❖ Function: ***crop_circle_frame(img,radius)***

- **Tham số đầu vào:** data array của ảnh, bán kính của hình tròn
- **Ý tưởng:** Giữ các pixels nằm trong bán kính hình tròn, các phần tử nằm ngoài sẽ bị loại bỏ.
- **Mô tả:** Tính khoảng cách từng pixel tới tâm hình tròn, nếu pixel nằm ngoài bán kính thì che lấp đi. Cuối cùng, thu được hình ảnh được crop theo khung tròn với bán kính theo yêu cầu.
- **Kết quả:**

Hình ảnh 13, cắt theo khung tròn bán kính 200

- **Nhận xét:** Bản chất chỉ là che lấp các pixels nằm ngoài vùng hình tròn nên không thay đổi nội dung hay thông tin ảnh, kết quả thu được đạt yêu cầu.

8. Cắt ảnh theo khung 2 elips chéo

- **Ý tưởng:** Tạo ra 2 mask elip chéo và apply mask vào ảnh gốc để thu được hình ảnh theo yêu cầu.
  - ❖ Function:***create_rotated_elip_mask(img,rotation_angle,major_axis,minor_axis)***
- **Tham số đầu vào:** data array của ảnh, góc quay, trục lớn, trục bé
- **Mô tả:** Dựa vào góc quay $\propto$ được truyền ở tham số, phép quay được tính bằng công thức $x = x \times \cos \propto + y \times \sin \propto$ và $y = x \times \sin \propto + y \times \cos \propto$. Để tạo mask cho elip chéo, ta cần kiểm tra mỗi pixel có nằm trong elip hay không, điều này được thể hiện bằng biểu thức $\left(\frac{x}{major_{axis}}\right) ** 2 + \left(\frac{y}{minor_{axis}}\right) ** 2$ và so sánh nó với 1. Nếu giá trị này <=1 thì pixel nằm trong elip mask và ngược lại.
  - ❖ Function: ***crop_2elips_frame(img,major_axis,minor_axis)***
- **Tham số đầu vào:** data array của ảnh, trục lớn, trục bé
- **Mô tả:** Tạo 2 mask elip chéo nhau bằng cách tạo 1 mask quay 45 độ và 1 mask quay 135 độ. Kết hợp 2 mask lại với nhau, những điểm

đều thuộc 1 trong 2 mask sẽ mang chân trị 1, ngược lại là 0. Duyệt qua từng pixel xem pixel đó có nằm trong khung 2 elip không, nếu không thì thực hiện làm đen để che lấp pixel đó đi.

- **Kết quả:**

Hinh anh 14, được cắt theo khung 2 elip có trục lớn là 300, trục nhỏ là 150

- **Nhận xét:** Tương tự với cắt khung hình tròn, cắt theo khung 2 elip chéo chỉ đơn giản là tạo ra 2 mask elip chéo nhau và apply lên hình ảnh gốc, bản chất vẫn là che lấp pixel không thuộc khung elip nên hình ảnh được giữ nguyên thông tin và nội dung.

C. NGUỒN THAM KHẢO

- Tham khảo về thuật toán box blur và summed-area table: **link1**, **link2**, **link3**
- Công thức thay đổi độ tương phản**:** **link**
- Công thức chuyển đổi ảnh RGB sang ảnh xám, sepia: **link1**, **link2**
- Tham khảo về laplacian filter để làm nét ảnh: **link**
- **ChatGPT**