LAB 02 – PROJECT 02: IMAGE PROCESSING

- Họ tên: Lê Yến Nhi

- MSSV: 19127498

- Đã hoàn thành: 100%

HÀM CHỨC NĂNG CHO 1, 2, 3, 5

`def truncate(val):` Dùng để giới hạn giá trị của màu trong khoảng [0, 255]

- Nếu giá trị đó âm, trả về 0.
- Nếu giá trị đó lớn hơn 255, trả về 255.
- Trả về chính nó nếu nằm trong khoảng từ [0, 255].

1. THAY ĐỔI ĐỘ SÁNG CHO ẢNH

1.1 Ý TƯỞNG THỰC HIỆN

- Chuyển đổi ảnh sang ma trận hai chiều bằng `np.array`.

- Mỗi phần tử của ma trận mang ba giá trị màu: đỏ, xanh lá, xanh dương (`red, green, blue`)

- Thay đổi độ sáng bằng cách duyệt qua từng phần tử của ma trận, cộng các giá trị màu của từng phần tử với cùng một số nguyên dương (`brightness = 40`), sau đó giới hạn giá trị của các màu bằng hàm `truncate.`

- Ta được ma trận mới, chuyển ma trận mới này thành hình ảnh bằng `Image.fromarray`.

1.2 KẾT QUẢ

2. THAY ĐỔI ĐỘ TƯƠNG PHẢN

2.1 Ý TƯỞNG THỰC HIỆN

- Chuyển đổi ảnh sang ma trận hai chiều bằng `np.array`.

- Mỗi phần tử của ma trận mang ba giá trị màu: đỏ, xanh lá, xanh dương (`red, green, blue`)

- Thay đổi độ tương phản bằng cách duyệt qua từng phần tử của ma trận, nhân các giá trị màu của từng phần tử với cùng một số nguyên dương (`alpha = 2`), sau đó giới hạn giá trị của các màu bằng hàm `truncate.`

- Ta được ma trận mới, chuyển ma trận mới này thành hình ảnh bằng `Image.fromarray`.

2.2 KẾT QUẢ

3. CHUYỂN ĐỔI ẢNH RGB THÀNH ẢNH XÁM

3.1 Ý TƯỞNG THỰC HIỆN

- Chuyển đổi ảnh sang ma trận hai chiều bằng `np.array`.

- Mỗi phần tử của ma trận mang ba giá trị màu: đỏ, xanh lá, xanh dương (`red, green, blue`)

- Chuyển đổi ảnh RGB thành ảnh xám bằng phương pháp ***Weighted method***

+ Do màu đỏ có nhiều bước sóng hơn trong cả ba màu, và màu xanh lá không chỉ có ít bước sóng hơn màu đỏ mà còn mang lại hiệu ứng dịu mắt hơn. Vì vậy ta phải giảm màu đỏ, tăng màu xanh lá cây, và đưa sự đóng góp của màu xanh lam vào giữa hai màu này.

+ Tạo biến `grayscale = 0.3 * red + 0.59 * green + 0.11 * blue`, làm tròn thành số nguyên không âm để thay đổi màu sắc.

+ Duyệt qua từng phần tử của ma trận, thay đổi các giá trị màu của từng phần tử bằng `grayscale`, sau đó giới hạn giá trị của các màu bằng hàm `truncate`

- Ta được ma trận mới, chuyển ma trận mới này thành hình ảnh bằng `Image.fromarray`.

3.2 KẾT QUẢ

4. LẬT ẢNH (NGANG – DỌC)

4.1 Ý TƯỞNG THỰC HIỆN

- Chuyển đổi ảnh sang ma trận hai chiều bằng `np.array`.

- Lật ảnh ngang – dọc bằng cách dùng `np.fliplr`, hoán vị lần lượt 2 cột ở phía trái và phải của ma trận cho nhau.

- Ta được ma trận mới, chuyển ma trận mới này thành hình ảnh bằng `Image.fromarray`.

4.2 KẾT QUẢ

5. CHỒNG HAI ẢNH CÙNG KÍCH THƯỚC

5.1 Ý TƯỞNG THỰC HIỆN

- Chuyển đổi hai ảnh màu sang hai ảnh xám bằng hàm

`convert_ColorImg_into_GrayscaleImg(image)`

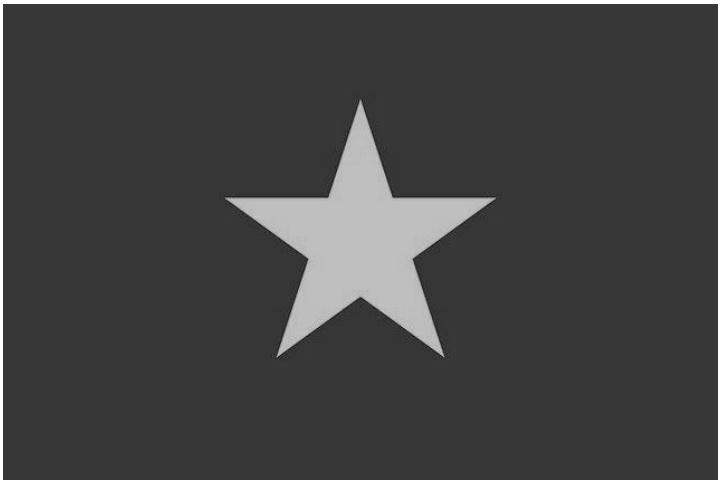
- Chuyển đổi hai ảnh xám sang hai ma trận hai chiều bằng `np.array`.

- Mỗi phần tử của ma trận mang ba giá trị màu: đỏ, xanh lá, xanh dương (`red, green, blue`)

- Chồng hai ảnh cùng kích thước bằng cách tạo một ma trận mới là tổng của hai ma trận ở trên. Duyệt qua từng phần tử của ma trận mới để giới hạn giá trị của các màu bằng hàm `truncate.`

- Chuyển ma trận mới này thành hình ảnh bằng `Image.fromarray`.

5.2 KẾT QUẢ



6. LÀM MỜ ẢNH

6.1 Ý TƯỞNG THỰC HIỆN

- Chuyển đổi ảnh sang ma trận hai chiều bằng `np.array`.

- Mỗi phần tử của ma trận mang ba giá trị màu: đỏ, xanh lá, xanh dương (`red, green, blue`)

- Làm mờ ảnh bằng phương pháp "**Gaussian blur 7 x 7**"

+ Ta duyệt qua những phần tử của ma trận – là điểm ảnh trung tâm của *ma trận lọc Gauss,* thay đổi các giá trị của các điểm ảnh này bằng hàm `averageColor(pixels,i,j,blurFactor)`.

+ Hàm `averageColor(pixels,i,j,blurFactor)` dùng để tìm giá trị của một điểm ảnh trung tâm. Giá trị này bằng trung bình cộng của các ô bao quanh nó. Số ô được xác định bằng công thức `(blurFactor * 2 + 1) ** 2`.

- Ta được ma trận mới, chuyển ma trận mới này thành hình ảnh bằng `Image.fromarray`.

6.2 KẾT QUẢ

THAM KHẢO:

https://www.tutorialspoint.com/dip/grayscale_to_rgb_conversion.htm
https://en.wikipedia.org/wiki/Kernel_(image_processing)