KHOA CÔNG NGHỆ THÔNG TIN

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN – ĐHQG TPHCM

TOÁN ỨNG DỤNG VÀ THỐNG KÊ CHO CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN 2: IMAGE PROCESSING

LỚP: 21CLC08

HỌ VÀ TÊN: NGÔ QUỐC QUÝ

MSSV: 21127679

Mục lục

I. Đánh giá độ hoàn thiện:

Chức năng	Mức độ hoàn thiện
Thay đổi độ sáng cho ảnh	Hoàn thành
Thay đổi độ tương phản	Hoàn thành
Lật ảnh (ngang – dọc)	Hoàn thành
Chuyển đổi ảnh RGB thành xám	Hoàn thành
Chuyển đổi ảnh RGB thành sep	Hoàn thành
Làm mờ	Hoàn thành
Làm sắc nét	Hoàn thành
Cắt ảnh theo kích thước (cắt ở trung tâm)	Hoàn thành
Cắt ảnh theo khung tròn	Hoàn thành
Khung là 2 hình ellip chéo nhau (nâng cao)	Chưa hoàn thành

II. Ý tưởng thực hiện, mô tả hàm:

1. Ý tưởng thực hiện:

- **change_brightness**: Đối với chức năng thay đổi độ sáng cho ảnh, ta cộng các màu trong mỗi pixel (red, green, blue) cho một số alpha bất kỳ (vd: alpha = 100), và sử dụng hàm **truncate** để giới hạn nếu chỉ số màu > 255 hoặc nhỏ < 0.
- **change_contrast:** Để độ tương phản của hình ảnh được xa nhau, cách dễ nhất đó là nhân các màu trong mỗi pixel cho một số scala bất kì để khoảng cách các màu xa nhau (scala nên nhỏ để độ tương hình ảnh không qua sáng vì bản chất giống với thay đổi độ sáng).
- **flip_photo_horizontal:** Đối với chức năng này, ta tạo thêm một hàm phụ **reverse_pixels** để hoán đổi vị trí các pixel, ta thực hiện hoán đổi các pixel theo từng cột (ứng với trục y).
- **flip_photo_vertical:** Chức năng này ta cũng sử dụng hàm **reverse_pixels** để hoán đổi vị trí các pixel, ta thực hiện hoán đổi các pixel theo từng hàng (ứng với trục x).
- **convert_RGB_to_greyscale:** Ta sử dụng công thức tính grey_scale sau đó gán lại cho từng pixel.
- **convert_RGB_to_sepia:** Ta sử dụng công thức tính mã màu sepia cho từng màu (red, green, blue) trong mỗi pixel sau đó gán lại các pixel mới cho ảnh.
- **blur_image:** Với mỗi pixel, ta thực hiện tính toán với 9 điểm xung quanh bao gồm cả pixel đó cho blur_kernel.
- **sharpen_image:** Tương tự với blur, ta thực hiện tính toán với 9 điểm xung quanh bao gồm cả pixel đó cho sharpen_kernel.

- **crop_image_in_center:** Chia hình ảnh thành các phần (4, 5, 6… phần), xác định kích thước mới của ảnh mới, sao đó gán các pixel ở trung tâm với ảnh mới.
- **crop_image_in_circle:** Tìm ra phương trình đường tròn, sau đó những pixel nào nhỏ hơn r^2 thì sẽ giữ lại, còn pixel nào nằm lớn hơn r^2 thì gán thành màu đen.

2. Mô tả hàm:

- **truncate:**
  - Đầu vào: color_value (giá trị này là 1 mã trong bộ rgb)
  - Đầu ra: color_value
  - Mô tả:
    - Nếu color_value < 0 sẽ gán bằng 0.
    - Nếu color_value > 255 sẽ gán bằng 255.
- **reverse_pixels:**
  - Đầu vào:
    - current_coordinate (Tọa độ hiện tại)
    - reverse_coordinate (Tọa độ muốn hoán đổi vị trí)
    - image (Hình ảnh đang xử lí)
  - Đầu ra: Không có
  - Mô tả: Hàm thực hiện lấy mã màu của pixel tại current_coordinate và reverse_coordinate sau đó hoán đổi cho nhau.
- **change_brightness**:
  - Đầu vào: image (ảnh cần xử lí)
  - Đầu ra: new_image (kết quả)
  - Mô tả:
    - Thực hiện 2 vòng lặp duyệt theo trục y rồi đến x.
    - Trong vòng lặp thực hiện cộng mã màu với 1 số alpha (cụ thể trong bài sử dụng alpha = 100) và sử dụng hàm **truncate** để các mã màu không bị vượt quá giới hạn.
- **change_contrast:**
  - Đầu vào: image (ảnh cần xử lí)
  - Đầu ra: new_image (kết quả)
  - Mô tả:
    - Thực hiện 2 vòng lặp duyệt theo trục y rồi đến x.
    - Trong vòng lặp thực nhân các mã màu với 1 số scala (cụ thể trong bài sử dụng scala = 1.5) để khoảng cách giữa các màu rộng hơn và sử dụng hàm **truncate** để các mã màu không bị vượt quá giới hạn.
- **flip_photo_horizontal:**
  - Đầu vào: image (ảnh cần xử lí)
  - Đầu ra: new_image (kết quả)
  - Mô tả:
    - Thực hiện 2 vòng lặp duyệt theo trục y rồi đến x.

        - Chia đôi hình ảnh thành 2 nửa (dọc - trục y), sử dụng hàm **reverse_pixels** để hoán đổi vị trí tương ứng bên kia trục.
        - Trong vòng lặp sẽ có điều kiện dừng để đến ngay trục trung tâm của ảnh thì ảnh không bị hoán đổi về như ban đầu.
- **flip_photo_vertical:**
    - Đầu vào: image (ảnh cần xử lí)
    - Đầu ra: new_image (kết quả)
    - Mô tả:
        - Ngược lại với lật ảnh ngang, ta thực hiện 2 vòng lặp duyệt theo trục x rồi mới đến y.
        - Chia đôi hình ảnh thành 2 nửa (ngang – trục x), sử dụng hàm **reverse_pixels** để hoán đổi vị trí tương ứng bên kia trục.
        - Trong vòng lặp sẽ có điều kiện dừng để đến ngay trục trung tâm của ảnh thì ảnh không bị hoán đổi về như ban đầu.
- **convert_RGB_to_greyscale:**
    - Đầu vào: image (ảnh cần xử lí)
    - Đầu ra: new_image (kết quả)
    - Mô tả:
        - Thực hiện 2 vòng lặp duyệt theo trục y rồi đến trục x.
        - Lấy từng giá trị màu tính toán theo công thức grey_scale = 0.299 * red + 0.587 * green + 0.114 * blue, sau đó gán grey_scale vào pixel đang tính toán, thực hiện cho đến khi kết thúc vòng lặp.
- **convert_RGB_to_sepia:**
    - Đầu vào: image (ảnh cần xử lí)
    - Đầu ra: new_image (kết quả)
    - Mô tả:
        - Thực hiện 2 vòng lặp duyệt theo trục y rồi đến trục x.
        - Tính toán giá trị của từng pixel theo từng công thức:
            - new_red = red * 0.393 + green * 0.769 + blue * 0.189
            - new_green = red * 0.349 + green * 0.686 + blue * 0.168
            - new_blue = red * 0.272 + green * 0.534 + blue * 0.131
        - Gán các giá trị mới được tính toán vào từng pixel cho đến hết vòng lặp.
- **blur_image:**
    - Đầu vào: image (ảnh cần xử lí)
    - Đầu ra: new_image (kết quả)
    - Mô tả:
        - Thực hiện 2 vòng lặp duyệt theo trục y rồi đến trục x. (bỏ qua viền)
        - Với mỗi tọa độ (x, y) sẽ khởi tạo thêm các tọa độ xung quanh (x, y) (bao gồm 8 điểm).

        - Thực hiện tính tổng các điểm trên sau đó chia cho 9 và gán vào cho (x, y), thực hiện đến hết vòng lặp (ở đây do tất cả phần tử trong kernel đều bằng 1 nên ta thực hiện tính tổng sau đó chia cho 9)
- **sharpen_image:**
    - Đầu vào: image (ảnh cần xử lí)
    - Đầu ra: new_image (kết quả)
    - Mô tả:
        - Tương tự các bước với **blur_image**.
        - Hàm sử dụng vòng for để duyệt các tọa độ xung quanh (x,y) kiểm tra điều kiện để lấy được alpha ứng với các phân tử trong kernel, sau đó thực hiện nhân vô hướng và tính tổng tương tự với **blur_image**, sau đó gán các giá trị màu đó cho từng pixel, và thực hiện tương tự đến hết vòng lặp.
- **crop_image_in_center:**
    - Đầu vào: image (ảnh cần xử lí)
    - Đầu ra: new_image (kết quả)
    - Mô tả:
        - Chia hình ảnh ban đầu ra thành 4 phần theo chiều cao, và tương tự với chiều ngang.
        - Gán khoảng cách bằng 1 phần
        - Tạo 1 chiều cao và 1 chiều rộng mới bằng 2 lần khoảng cách.
        - Thực hiện 2 vòng for theo y rồi đến x, để duyệt và lấy từng phần tử ở trung tâm và gán vào ảnh mới.
- **crop_image_in_circle:**
    - Đầu vào: image (ảnh cần xử lí)
    - Đầu ra: new_image (kết quả)
    - Mô tả:
        - Thực hiện tính toán bán kính và tâm I của đường tròn
        - Duyệt qua 2 vòng lặp và kiểm tra điều kiện nếu tọa độ nhỏ hơn bình phương bán kính thì được giữ lại, ngược lại, nếu lớn hơn sẽ gán màu đen vào những pixel đó.
- **input_image:**
    - Đầu vào: Không có đầu vào.
    - Đầu ra:
        - image (hình ảnh cần xử lí)
        - name (tên hình ảnh cần xử lí)
    - Mô tả: Thực hiện đọc hình ảnh bằng Image.open() và hình ảnh sang RGB và trả về.
- **output:**
    - Đầu vào:
        - image_result (kết quả sau khi xử lí hình ảnh bằng các chức năng)
        - function (tên chức năng)

        - name (tên ảnh)
    - Đầu ra: Không có đầu ra.
    - Mô tả:
        - Thực hiện chuyển đổi kết quả thành ma trận 2 chiều, sau đó reshape lại thành (w, g, (RGB))
        - Xử lí chuỗi để tạo ra tên ảnh mới
        - Lưu hình ảnh dưới định dạng ‘{name}_{function}.png’.
- **menu:**
    - Đầu vào:
        - image (Hình ảnh cần xử lí)
        - name (Tên hình ảnh)
    - Đầu ra: Không có đầu ra.
    - Mô tả:
        - In bảng mô tả cách chọn chức năng trong màn hình console.
        - Yêu cầu nhập từ 0 – 7 cho từng chức năng tương ứng.
        - Kiểm tra xem người dùng muốn thực hiện chức năng nào và gọi hàm **output** để xuất ảnh sau khi xử lí.

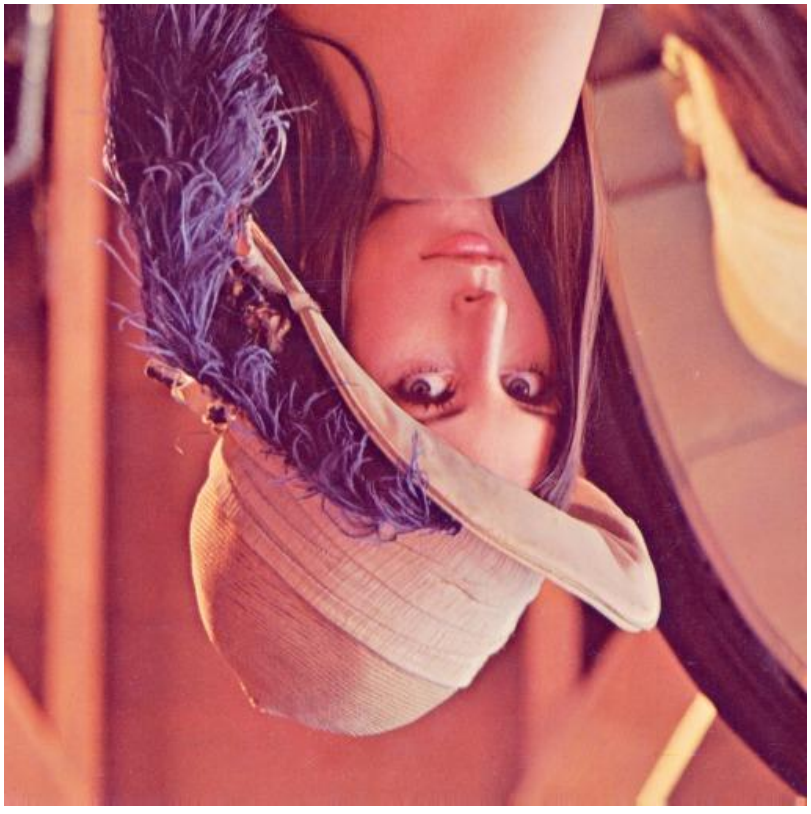
III. Hình ảnh kết quả của từng chức năng:

Ảnh gốc

Thay đổi độ sáng

Thay đổi độ tương phản

Lật ảnh ngang



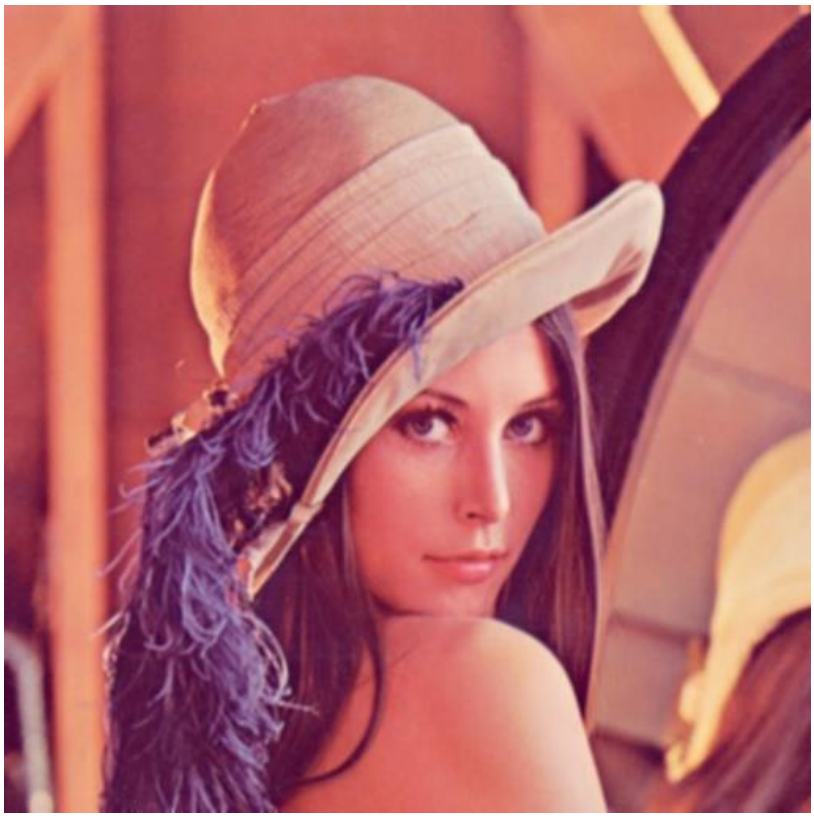
Lật ảnh dọc

Chuyển ảnh RGB thành xám

Chuyển ảnh RGB thành sepia

Làm mờ ảnh xám

Làm sắc nét ảnh xám



Làm mờ ảnh màu

Làm sắc nét ảnh màu

Cắt ảnh ở trung tâm

Cắt ảnh theo khung tròn

IV. Tài liệu tham khảo:

Baeldung: https://www.baeldung.com/cs/convert-rgb-to-grayscale

Yusuf Shakeel: https://www.youtube.com/watch?v=Q5Xbj3jlMsc&t=278s

Wikipedia: **https://en.wikipedia.org/wiki/Kernel_(image_processing)**