ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO LAB 2
IMAGE PROCESSING

HỌ TÊN SINH VIÊN : **Võ Quốc Bình**

MÃ SỐ SINH VIÊN : **21127233**

LỚP : **21CLC08 - Toán ứng dụng và thống kê**

GIÁO VIÊN HƯỚNG DẪN : **Phan Thị Phương Uyên**

Thành phố Hồ Chí Minh, tháng 7 năm 2022

Mục lục

I. Thông tin sinh viên

- Họ tên sinh viên: Võ Quốc Bình
- Mã số sinh viên: 21127233
- File nộp bài:
    - File code: 21127233.ipynb
    - File báo cáo: 21127233.pdf

II. Mức độ hoàn thành

STT	Tên chức năng	Mức độ hoàn thành
1	Thay đổi độ sáng cho ảnh	100%
2	Thay đổi độ tương phản cho ảnh	100%
3	Lật ảnh	100%
4	Thay đổi màu cho ảnh	100%
5	Làm mờ/Làm sắc nét ảnh	100%
6	Cắt ảnh theo kích thước	100%
7	Cắt ảnh hình tròn	100%
8	Nâng cao: Cắt 2 hình ellipse chéo nhau	100%
9	Viết hàm main demo	100%

- Mức độ hoàn thành toàn đồ án: 100%

III. Ý tưởng thực hiện

1. Thay đổi độ sáng cho ảnh

- Ta tiến hành cộng cho tất cả các pixel một hằng số cho tất cả các pixel. Từ đó có thể dễ dàng thay đổi độ sáng cho các pixel đó.
- Ngoài ra ta còn phải giới hạng cho các pixel có giá trị sau khi cộng ở khoảng 0 đến 255.

2. Thay đổi độ tương phản

- Ta sử dụng công thức thay đổi độ tương phản theo công thức:

$$factor = \frac{259(constract + 255)}{255(259 - constract)}$$

Công thức được tham khảo tại đây

- Sau đó sử dụng công thức điều chỉnh hình ảnh với factor vừa có ở trên và giới hạn các pixel từ 0 đến 255 để có kết quả cuối cùng:

$$outputArray = factor(inputArray - 128) + 128$$

Công thức được tham khảo tại đây

3. Lật ảnh

a. Lật dọc

- Ta lật dọc bằng cách đổi ngược thứ tự sắp xếp của các hàng.

b. Lật ngang

- Ta lật dọc bằng cách đổi ngược thứ tự sắp xếp của các cột.

4. Chuyển ảnh màu RGB

a. Chuyển thành màu xám

- Ta thay đổi thành màu xám bằng cách thay đổi trọng số của các màu RGB tương ứng.
- Công thức : $grayImage = 0.299 * R + 0.587 * G + 0.114 * B$

b. Chuyển thành màu sepia

- Ta tạo ma trận có mảng màu sepia: $[[0.393, 0.769, 0.189], [0.349, 0.686, 0.168], [0.272, 0.534, 0.131]]$

- Sau đó nhân ma trận ảnh với ma trận sepia vừa tạo và clip giá trị lại từ 0 tới 255.

5. Làm mờ/Làm nét ảnh

a. Làm mờ

- Tạo ma trận blur_kernel: $[[[1],[2],[1]],$
  $[[2],[4],[2]],$
  $[[1],[2],[1]]]$
- Ta tạo thêm một ma trận viền có kích thước lớn hơn ma trận gốc 1 pixel ở viền.
- Sau đó ta nhân từng pixel với blur_kernel và sau đó xuất ra ma trận output_image.

b. Làm nét ảnh

- Ý tưởng tưởng tương tự làm mờ ảnh.
- Tạo ma trận sharpen_kernel: $[[[0]\ ,[-1],[0]],$
  $[[-1],[5]\ ,[-1]],$
  $[[0]\ ,[-1],[0]]]$
- Ta tạo thêm một ma trận viền có kích thước lớn hơn ma trận gốc 1 pixel ở viền
- Sau đó ta nhân từng pixel với sharpen_kernel và sau đó xuất ra ma trận output_image.
- Tại mỗi pixel ta clip ảnh trong khoảng từ 0 đến 255.

6. Cắt hình ở trung tâm

- Ta lấy cạnh nhỏ nhất của hình rồi chia đôi từ đó có được edge_size.
- Sau đó, ta tính toán tọa độ cắt(start_h, start_w) của điểm bắt đầu cắt để lấy phần cắt từ trung tâm của hình ảnh gốc. Điểm bắt đầu cắt (start_h, start_w) sẽ nằm ở giữa theo chiều dài và chiều rộng của hình ảnh.
- Cuối cùng ta được một hình vuông có độ dài mỗi cạnh bằng cạnh nhỏ nhất của hình gốc chia đôi.

7. Cắt hình tròn

- Ta cắt thành hình tròn dựa trên cắt hình ở trung tâm(phần III.6). Với bán kính bằng cạnh của hình vuông chia đôi.
- Ta xác định tọa độ trung tâm hình(center).
- Tạo mask hình tròn: Ta tạo một mask (mặt nạ) có kích thước giống với ảnh cắt (cropped_image) và các giá trị trong mask được đặt là True tại các điểm nằm trong

vùng hình tròn, các điểm nằm ngoài vùng hình tròn được đặt là False với công thức đề xác định hình là: $(x - center[1])\text{^}2 + (y - center[0])\text{^}2 <= radius\text{^}2$

- Áp dụng mask lên ảnh cắt: Cuối cùng, ta áp dụng mask đã tạo lên ảnh cắt (cropped_image), giữ lại các điểm ảnh có giá trị True trong mask và đặt các điểm ảnh có giá trị False trong mask về giá trị 0.

8. Cắt 2 hình ellipse chéo nhau

- Tương tự như hình tròn, ta vẽ lần lượt 2 hình ellipse dựa trên nên hình vuông
- Hai trục chính và trục phụ được tính bằng công thức:

$$major\_axis = radius * 1.25$$

$$minor\_axis = radius * 0.671875$$

- Và sử dụng dụng ma trận mask để đánh dấu hình với điều kiện:

$$(distance\_1 + distance\_2) <= 1:$$

- Trong đó:

$$distance\_1 = ((x - center[0]) * np.sin(angle\_radians\_sub) + (y - center[1]) * np.\cos(angle\_radians\_sub)) ** 2/minor\_axis ** 2$$

$$distance\_2 = ((x - center[0]) * np.cos(angle\_radians\_sub) - (y - center[1]) * np.\sin(angle\_radians\_sub)) ** 2/major\_axis ** 2$$

Công thức được tham khảo ở đây

IV. Mô tả các hàm image processing

- Ta sử dụng 2 thư viện chính để viết chương trình này numpy(xử lí các phép tính trên ma trận) và PIL(để nhập xuất ảnh)

1. Hàm change_brightness

- Tham số đầu vào: image_array, brightness(mặc định là 25)
- Giá trị trả về: output_image
- Mô tả:
    - Tạo ma trận trả về giống với ma trận đầu vào
    - Cộng các pixel với giá trị brightness sau đó clip ma trận lại trong khoảng từ 0 đến 255

2. Hàm change_constract

- Tham số đầu vào: image_array, constract(mặc định là 25)
- Giá trị trả về: output_image
- Mô tả:
    - Tạo ma trận trả về giống với ma trận đầu vào
    - Tạo hằng số factor ứng với tham số constract đầu vào
    - Nhân hằng số factor vào ma trận với công thức:

$$outputArray = factor(inputArray - 128) + 128$$

    - Sau đó clip ma trận lại trong khoảng từ 0 đến 255

3. Hàm lật hình

a. Hàm flip_vertically

- Tham số đầu vào: image_array
- Giá trị trả về: output_image
- Mô tả:
    - Ta đổi ngược thứ tự các hàng với nhau

b. Hàm flip_horizontally

- Tham số đầu vào: image_array
- Giá trị trả về: output_image
- Mô tả:
    - Tả đổi ngược thứ tự các cột với nhau

4. Hàm chuyển màu RGB

a. Hàm change_to_grayscale

- Tham số đầu vào: image_array
- Giá trị trả về: gray_image
- Mô tả:
  - Tách các kênh màu gốc thành 3 màu riêng biệt
  - Tạo hằng số R, G, B với công thức đã nêu ở phần ý tưởng.
  - Cộng các kênh màu sau khi đã nhân với các hằng số R, G, B tương ứng với các kênh màu
  - Gán lại giá trị tại các pixel tương ứng

b. Hàm change_to_sepia

- Tham số đầu vào: image_array
- Giá trị trả về: sepia_image
- Mô tả:
  - Tạo ma trận màu sepia đã nêu ở phần ý tưởng
  - Nhân từng pixel ở ma trận màu gốc với ma trận màu sepia vừa tạo
  - Clip ma trận lại từ 0 đến 255 để bảo toàn màu

5. Hàm làm mờ ảnh/Hàm làm sắc nét ảnh

a. Hàm blur_image

- Tham số đầu vào: image array, iteration(giá trị mặc định đầu vào là 1)
- Giá trị trả về: output_image
- Mô tả:
  - Tạo ma trận blur_kernel với giá trị kernel tham khảo ở đây:

$$blur\_kernel = ([[[1],[2],[1]],\ [[2],[4],[2]],\ [[1],[2],[1]]]) \ / \ 16$$

  - Tạo ma trận padded_image với kích thước mỗi rộng hơn 1 pixel ở cạnh
  - Sau đó ta áp dụng blur_kernel lên từng điểm ảnh kết hợp với viền ở padded_image để tạo ra pixel sau khi blur.

b. Hàm sharpen_image

- Tham số đầu vào: image array, iteration(giá trị mặc định đầu vào là 1)
- Giá trị trả về: output_image
- Mô tả:
  - Ý tưởng tương tự hàm blur_image
  - Tạo ma trận sharpen_kernel với giá trị sharpen tham khảo ở đây:

$$sharpen\_kernel = ([[[0], [-1], [0]],$$
$$[[-1], [5], [-1]],$$
$$[[0], [-1], [0]]])$$

- Tạo ma trận padded_image với kích thước mỗi rộng hơn 1 pixel ở cạnh
- Sau đó ta áp dụng sharpen_kernel lên từng điểm ảnh kết hợp với viền ở padded_image để tạo ra pixel sau khi sharpen.
- Ngoài ra, ta cần clip lại pixel chạy từ 0 đến 255 để tránh bị lỗi ảnh

6. Hàm crop_center

- Tham số đầu vào: image_array
- Giá trị trả về: cropped image
- Mô tả:
  - Lấy cạnh nhỏ nhất của ảnh rồi chia đôi và gán vào edge_size
  - Lấy kích thước của ảnh hiện tại
  - Tính điểm bắt đầu của ảnh bằng công thức:

$$start\_h = (height - edge\_size) // 2$$
$$start\_w = (width - edge\_size) // 2$$

  - Sau đó crop ảnh chạy từ điểm bắt đầu chạy tới edge_size của mỗi chiều rộng và chiều cao

7. Hàm crop_circle

- Tham số đầu vào: image_array
- Giá trị trả về: cropped image
- Mô tả:
  - Tạo khung hình vuông với ý tưởng tại phần cắt trung tâm để tạo thành hình vuông.
  - Lấy tâm hình vuông và bán kính bằng nửa cạnh hình vuông
  - Áp dụng công thức ở phần III.7 để tạo mask đánh dấu giữa lại các pixel hình tròn còn lại thành màu đen.

8. Hàm crop_two_cross_ellipse

- Tham số đầu vào: image_array
- Giá trị trả về: cropped image

- Mô tả:
    - Tạo khung hình vuông với ý tưởng tại phần cắt trung tâm để tạo thành hình vuông.
    - Tính trục chính và trục phụ dựa vào công thức ở phần III.8
    - Áp dụng công thức ở phần III.8 để tạo mask đánh dấu giữa lại các pixel hình tròn còn lại thành màu đen.

9. Hàm save_image

- Tham số đầu vào: image_array, filename, name_process
- Giá trị trả về: Không
- Mô tả:
    - Ta lấy dạng file ảnh gốc và tên file ảnh từ filename
    - Chuyển ảnh dạng ma trận về thành hình
    - Tạo tên mới bằng cú pháp : tên file + "_" + name_process(tên process thực thi) + "."+ dạng file của ảnh gốc
    - Dùng hàm save của PIL đề lưu lại ảnh với tên là tên vừa tạo

10. Hàm main demo

- Mô tả:
    - Lấy tên file và tên option người dùng chọ
    - Dùng chuỗi lệnh if else để handle từng option người dùng chọn.

V. Nhận xét ảnh sau khi qua xử lí

- Ảnh đầu vào kích thước 512x512, filename = “ronaldo.jpg”

1. Thay đổi độ sáng cho ảnh

ronaldo_brightness.jpg

Nhận xét:

- Nếu brightness là số dương thì ảnh tiến gần về màu trắng.
- Nếu brightness bằng không thì ảnh không đổi màu.
- Nếu brightness là số âm thì ảnh tiến gần về màu đen

2. Thay đổi độ tương phản

ronaldo_constract.jpg

Nhận xét:

- Nếu giá trị factor nằm trong khoảng từ 0 đến 1 thì làm giảm độ tương phản của ảnh
- Nếu giá trị factor lớn hơn 1 thì làm tăng độ tương phản của ảnh
- Các vùng sáng sau khi đổi độ tương phản thì sáng hơn, các vùng tối thì tối hơn

3. Lật ảnh

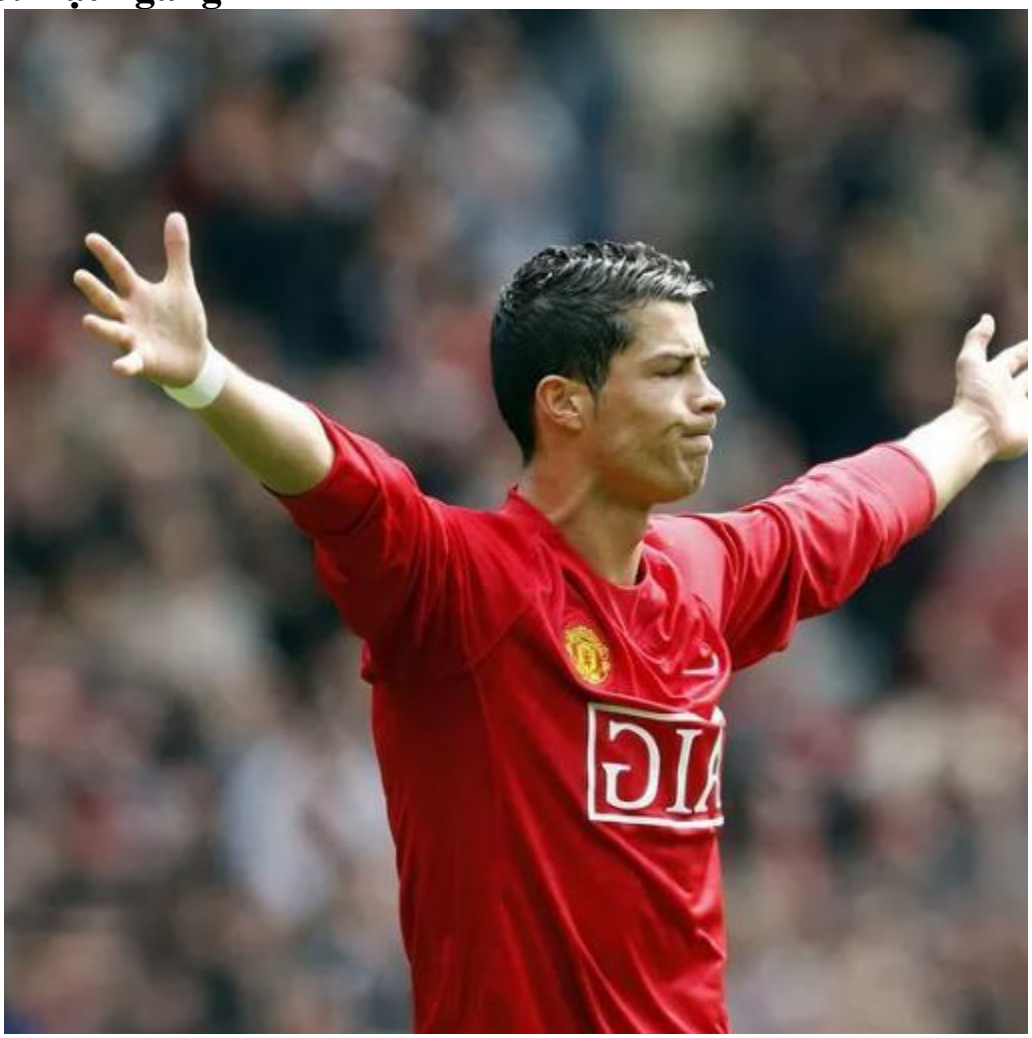
a. Lật dọc

ronaldo_vertical.jpg

Nhận xét:

- Ảnh bị lật dọc
- Nội dung ảnh không bị hỏng

b. Lật ngang



ronaldo_horizontal.jpg

Nhận xét:

- Ảnh bị lật ngang
- Nội dung ảnh không bị hỏng

4. Chuyển ảnh màu RGB

a. Chuyển thành màu xám

ronaldo_grayscale.jpg

Nhận xét:

- Ảnh đơn sắc xám
- Nội dung ảnh được giữ nguyên
- Kích thước ảnh giảm

b. Chuyển thành màu sepia

ronaldo_sepia.jpg

Nhận xét:

- Ảnh đơn sắc sepia
- Nội dung ảnh được giữ nguyên
- Kích thước ảnh giảm

5. Làm mờ/Làm nét ảnh

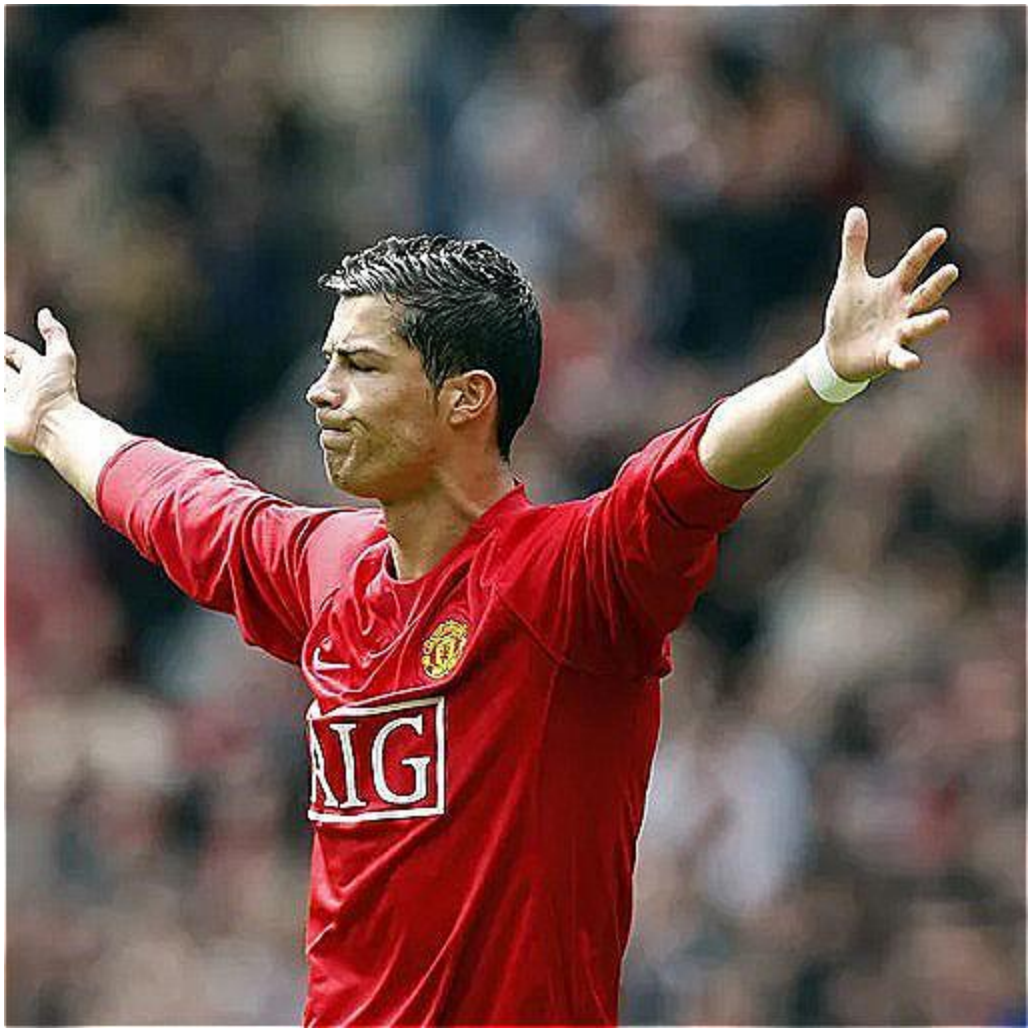
a. Làm mờ

ronaldo_blur.jpg

Nhận xét:

- Ảnh mờ hơn
- Nội dung ảnh được giữ nguyên
- Kích thước ảnh giảm

b. Làm nét ảnh



ronaldo_sharpen.jpg

Nhận xét:

- Các đường viền sắc nét hơn
- Một số chỗ có thể bị thay đổi thuộc tính màu
- Dễ bị nhiễu hình

6. Cắt hình ở trung tâm

ronaldo_crop_center.jpg

Nhận xét:

- Hình được cắt ở trung tâm hình gốc
- Hình vuông giúp giảm kích thước ảnh

7. Cắt hình tròn

ronaldo_circle.jpg

Nhận xét:

- Ảnh được cắt từ hình vuông
- Ảnh nằm ở tâm hình tròn
- Các phần ngoại phạm vi hình tròn bị cắt bỏ

8. Cắt 2 hình ellipse chéo nhau

ronaldo_ellipse_cross.jpg

Nhận xét:

- Ảnh được cắt từ hình vuông
- Ảnh nằm ở tâm 2 hình ellipse chéo
- Các phần ngoại phạm vi 2 hình ellipse bị cắt bỏ

VI. Tham khảo

- https://www.dfstudios.co.uk/articles/programming/image-programming-algorithms/image-processing-algorithms-part-5-contrast-adjustment/
- https://www.maa.org/external_archive/joma/Volume8/Kalman/General.html
- https://en.wikipedia.org/wiki/Kernel_(image_processing)