**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**LỚP: 21CLC07**

**Ảnh có chứa Phông chữ, biểu tượng, Đồ họa, Xanh điện

Mô tả được tạo tự động**

**BÁO CÁO PROJECT 1:**

**COLOR COMPRESSION**

Thông tin sinh viên thực hiện:

* Họ và Tên: Nguyễn Tấn Khiêm
* MSSV: 21127726

Giảng viên hướng dẫn:

• Giảng viên: Nguyễn Văn Quang Huy  
• Giảng viên: Ngô Đình Hy  
• Giảng viên: Trần Hà Sơn  
• Giảng viên: Nguyễn Đình Thúc

1. Đánh giá mức độ hoàn thành:

|  |  |  |
| --- | --- | --- |
| STT | Nội dung | Mức độ hoàn thành |
| 1 | Thay đổi độ sáng 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. | 100% |
| 7 | Cắt ảnh theo khung hình tròn. | 100% |
| 8 | Hàm main theo yêu cầu. | 100% |
| 9 | Cắt theo khung là hai hình elip chéo nhau. | 100% |

1. Miêu tả thuật toán:

* Đọc ảnh bằng thư viện PIL.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Hình : hàm đọc ảnh.

* Hàm lưu ảnh dưới dạng png:

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Hình : Hàm dùng để lưu ảnh dưới dạng png.

* Hàm tăng độ sáng: hàm sử dụng thư viện NumPy để chuyển đổi hình ảnh thành một mảng NumPy, và sau đó chuyển đổi các giá trị trong mảng này thành kiểu dữ liệu số nguyên. Tiếp theo, hàm sẽ thực hiện việc tăng độ sáng bằng cách cộng giá trị của tất cả các pixel trong mảng hình ảnh với một giá trị số nguyên được tính bằng cách nhân tỷ lệ scale với 255 (điều này được thực hiện để đảm bảo giá trị tăng không vượt quá phạm vi 0-255). Điều này làm tăng giá trị độ sáng của từng pixel trong hình ảnh. Sau khi tăng độ sáng, hàm kiểm tra và điều chỉnh các giá trị pixel nào lớn hơn 255 để đảm bảo rằng hình ảnh không vượt quá phạm vi 0-255 (giá trị tối đa của một pixel là 255). Cuối cùng, hàm chuyển đổi mảng NumPy đã tăng độ sáng trở lại thành kiểu dữ liệu np.uint8 (kiểu dữ liệu 8-bit không dấu) và trả về kết quả.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Hình : Hàm dùng để tăng độ sáng.

* Hàm tăng độ tương phản: Hàm này cũng sử dụng thư viện NumPy để chuyển đổi hình ảnh thành một mảng NumPy và chuyển đổi các giá trị trong mảng này thành kiểu dữ liệu số thực. Tiếp theo, hàm thực hiện việc tăng độ tương phản bằng cách nhân tất cả các pixel trong mảng hình ảnh với một giá trị được tính bằng cách bình phương tỷ lệ scale cộng với 1. Điều này làm tăng độ tương phản của từng pixel trong hình ảnh. Sau khi tăng độ tương phản, hàm kiểm tra và điều chỉnh các giá trị pixel nào lớn hơn 255 hoặc nhỏ hơn 0 để đảm bảo rằng hình ảnh không vượt quá phạm vi 0-255 (giá trị tối đa của một pixel là 255 và giá trị tối thiểu là 0). Cuối cùng, hàm chuyển đổi mảng NumPy đã tăng độ tương phản trở lại thành kiểu dữ liệu np.uint8 (kiểu dữ liệu 8-bit không dấu) và trả về kết quả.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Hình : Hàm tăng độ tương phản.

* Hàm lật ảnh ngang-dọc: Hàm này thực hiện lật ảnh theo chiều dọc (0) hoặc theo chiều ngang (1).  
  + Đầu tiên, hàm kiểm tra xem mode có phải là 0 hoặc 1 không. Nếu mode không nằm trong danh sách [0, 1], thì in thông báo lỗi "Mode is wrong value." và kết thúc chương trình.

+ Tiếp theo, hàm chuyển đổi hình ảnh thành một mảng NumPy bằng cách sử dụng hàm np.array(image).  
+ Sau đó, hàm lấy kích thước ban đầu của ảnh bằng cách sử dụng image\_array.shape, trong đó dim sẽ là một tuple (h, w) đại diện cho chiều cao và chiều rộng của hình ảnh.  
+ Tiếp theo, hàm thực hiện lật ảnh theo mode được chỉ định bằng cách sử dụng hàm np.flip(image\_array, mode). Nếu mode là 0, thì ảnh sẽ bị lật theo chiều dọc, nếu mode là 1, thì ảnh sẽ bị lật theo chiều ngang.  
+ Sau khi lật ảnh, mảng NumPy của ảnh lật sẽ được chuyển đổi trở lại thành một đối tượng PIL Image bằng cách sử dụng Image.fromarray(flipped\_image\_array).  
+ Cuối cùng, hàm trả về ảnh lật PIL Image đã được xử lý.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Hình : Hàm lật ảnh ngang-dọc theo mode.

* Hàm chỉnh ảnh xám: Hàm grey\_image nhận vào một hình ảnh và thực hiện chuyển đổi hình ảnh màu (RGB) sang hình ảnh xám:

+ Hàm chuyển đổi hình ảnh thành một mảng NumPy và chuyển đổi các giá trị trong mảng này thành kiểu dữ liệu số thực để thực hiện tính toán.  
+Tiếp theo, hàm sử dụng phép nhân ma trận (dot product) giữa mảng hình ảnh và một mảng có các hệ số alpha, beta và gamma để tính toán giá trị mới của hình ảnh xám. Mỗi điểm ảnh trong hình ảnh xám sẽ là kết quả của phép nhân các giá trị tương ứng trong hàng của mảng hình ảnh với hệ số tương ứng.  
+ Sau đó, hàm tạo một mảng mới gray\_image có cùng kích thước với hình ảnh ban đầu, và lặp lại giá trị tính toán được cho từng kênh màu để tạo thành một hình ảnh màu xám, trong đó tất cả các kênh màu có cùng giá trị.  
+ Cuối cùng, hàm chuyển đổi mảng NumPy của ảnh xám gray\_image trở lại thành kiểu dữ liệu np.uint8 (kiểu dữ liệu 8-bit không dấu) và trả về kết quả.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự động

Hình : Hàm chuyển ảnh xám.

* Hàm chuyển ảnh sepia: Hàm sepia\_image nhận vào một hình ảnh và thực hiện chuyển đổi màu sepia cho hình ảnh đó:   
  + Đầu tiên, hàm định nghĩa ma trận chuyển đổi màu sepia sepia\_matrix. Ma trận này chứa các hệ số màu để thực hiện chuyển đổi màu sepia.  
  + Tiếp theo, hàm chuyển đổi hình ảnh thành một mảng NumPy và chuyển đổi các giá trị trong mảng này thành kiểu dữ liệu số thực để thực hiện tính toán.  
  + Sau đó, hàm thực hiện phép nhân ma trận giữa sepia\_matrix và mảng hình ảnh để chuyển đổi màu. Kết quả là một mảng sepia\_image mới chứa các giá trị màu mới đã được chuyển đổi.  
  + Tiếp theo, hàm giới hạn giá trị của các kênh màu trong khoảng 0-255 bằng cách sử dụng hàm np.clip(sepia\_image, 0, 255). Điều này đảm bảo rằng các giá trị màu không vượt quá phạm vi cho phép (0 đến 255).  
  + Sau đó, mảng sepia\_image được chuyển đổi trở lại thành đối tượng PIL Image bằng cách sử dụng Image.fromarray(sepia\_image) và kiểu dữ liệu của nó được định dạng thành np.uint8 (kiểu dữ liệu 8-bit không dấu).  
  + Cuối cùng, hàm trả về hình ảnh sepia đã được xử lý.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Hình : Hàm chuyển ảnh thành màu sepia.

* Hàm thực hiện phép tích chập giữa một hình ảnh và một kernel (bộ lọc).

+ Hình ảnh đầu vào và kernel đều là các mảng NumPy. Hình ảnh có thể là hình ảnh màu hoặc hình ảnh xám, và kernel là một mảng bộ lọc có kích thước (kernelDim, kernelDim).

+ Đầu tiên, hàm tính toán `kernelDim` bằng cách lấy kích thước của kernel (số hàng hoặc số cột của kernel).

+ Sau đó, hàm tính toán kích thước phần đệm (padSize) dựa trên kích thước của kernel. Kích thước phần đệm được tính bằng phép chia nguyên của kernelDim cho 2.

+ Tiếp theo, hàm thực hiện phần đệm bằng cách sử dụng hàm `np.pad` để thêm các giá trị biên vào hình ảnh ban đầu. Giá trị biên được sao chép từ các biên của hình ảnh gốc, được xác định bằng cách sử dụng mode='edge'.

+ Sau khi hình ảnh đã được đệm, hàm tạo một mảng trống `image\_result` có kích thước giống với hình ảnh gốc. Mảng này sẽ chứa kết quả sau khi thực hiện phép tích chập.

+ Tiếp theo, hàm duyệt qua từng điểm ảnh trong hình ảnh gốc. Đối với mỗi điểm ảnh, hàm lấy một phần gồm kernelDim hàng và kernelDim cột từ hình ảnh đã đệm (image\_padded) bắt đầu từ vị trí (i, j). Sau đó, hàm thực hiện phép tích chập bằng cách nhân các giá trị trong phần đó với kernel và tính tổng.

+ Kết quả tính được cho mỗi điểm ảnh được lưu vào mảng `image\_result`. Cuối cùng, hàm trả về hình ảnh kết quả đã thực hiện phép tích chập. Mảng `image\_result` chứa hình ảnh sau khi áp dụng kernel để thực hiện phép tích chập.

+ Hàm này dùng để hổ trợ cho hàm tăng độ tương phản và làm mờ ảnh.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, phần mềm

Mô tả được tạo tự động

Hình : Hàm thực hiện phép tích chập giữa một hình ảnh và một kernel.

* Hàm tăng độ sắc nét:

+ Hàm `sharpen\_image` tăng độ sắc nét của hình ảnh bằng cách áp dụng bộ lọc sắc nét. Bộ lọc này tạo hiệu ứng sắc nét bằng cách làm nổi bật các biên cạnh trong hình ảnh. Hàm chấp nhận hình ảnh màu hoặc hình ảnh xám và trả về hình ảnh sau khi đã tăng độ sắc nét.

+ Để thực hiện việc tăng độ sắc nét, hàm sử dụng ma trận kernel có kích thước 3x3 và các giá trị phù hợp. Nếu hình ảnh là màu, hàm áp dụng bộ lọc sắc nét cho từng kênh màu (R, G, B) của hình ảnh. Nếu hình ảnh là xám, hàm áp dụng bộ lọc sắc nét trực tiếp trên hình ảnh.

+ Sau khi áp dụng bộ lọc, hàm giới hạn giá trị các pixel trong khoảng 0 đến 255 để đảm bảo rằng hình ảnh không vượt quá phạm vi màu chuẩn RGB. Cuối cùng, hàm trả về hình ảnh sau khi đã tăng độ sắc nét.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phông chữ

Mô tả được tạo tự động

Hình : Hàm tăng độ sắc nét.

* Hàm làm mờ ảnh: Hàm `blur\_image` thực hiện việc làm mờ ảnh bằng cách áp dụng bộ lọc làm mờ (blur filter) lên hình ảnh.

+ Đầu tiên, hàm xác định kích thước của kernel là 5x5 và tạo ma trận kernel với các giá trị bằng nhau, được tính dựa trên tham số `amount`. Khi `amount = 1.0`, tổng các phần tử trong kernel sẽ là 1, điều này sẽ tạo ra hiệu ứng làm mờ tổng thể hình ảnh.

+ Tiếp theo, hàm chuyển đổi hình ảnh thành một mảng NumPy bằng cách sử dụng `np.array(image)`.

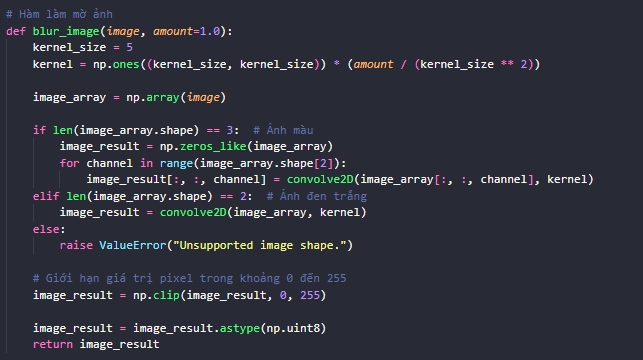
+ Sau đó, hàm kiểm tra hình ảnh là ảnh màu (hình ảnh có ba kênh màu) hay ảnh đen trắng (hình ảnh có một kênh màu). Tùy vào loại hình ảnh, hàm thực hiện phép tích chập với kernel làm mờ tương ứng.

+ Nếu hình ảnh là ảnh màu, hàm tạo một mảng kết quả `image\_result` có cùng kích thước với hình ảnh ban đầu và thực hiện phép tích chập với kernel làm mờ cho từng kênh màu (R, G, B) của hình ảnh.

+ Nếu hình ảnh là ảnh đen trắng, hàm thực hiện phép tích chập với kernel làm mờ trực tiếp trên hình ảnh.

+ Sau khi thực hiện phép tích chập, hàm giới hạn giá trị của các pixel trong khoảng 0-255 bằng cách sử dụng `np.clip(image\_result, 0, 255)`. Điều này đảm bảo rằng các giá trị màu không vượt quá phạm vi cho phép (0 đến 255).

+ Cuối cùng, hàm chuyển đổi mảng NumPy của ảnh kết quả `image\_result` thành kiểu dữ liệu `np.uint8` (kiểu dữ liệu 8-bit không dấu) và trả về kết quả.



Hình : Hàm làm mờ ảnh.

* Hàm cắt ảnh ở trung tâm: Hàm `crop\_and\_resize\_center\_image` nhận vào một hình ảnh và một tỷ lệ `scale` mặc định là 0.5. Mục đích của hàm là cắt và thay đổi kích thước hình ảnh sao cho chỉ giữ lại phần chính giữa của hình ảnh, và thay đổi kích thước của nó dựa trên tỷ lệ `scale`.

+ Đầu tiên, hàm chuyển đổi hình ảnh thành một mảng NumPy bằng cách sử dụng `np.array(image)`.

+ Sau đó, hàm tính toán chiều cao và chiều rộng của hình ảnh ban đầu.

+ Tiếp theo, hàm tính toán chiều cao và chiều rộng mới của hình ảnh dựa trên tỷ lệ `scale`.

+ Sau đó, hàm tính toán các chỉ số (`left`, `top`, `right`, `bottom`) để cắt phần chính giữa của hình ảnh. Phần chính giữa được tính bằng cách lấy tỷ lệ của kích thước mới so với kích thước ban đầu và áp dụng để cắt phần tương ứng.

+ Sau khi cắt phần chính giữa, hàm tạo một mảng mới `cropped\_image\_array` chỉ chứa phần hình ảnh đã cắt.

+ Cuối cùng, hàm chuyển đổi mảng NumPy của hình ảnh cắt `cropped\_image\_array` thành đối tượng PIL Image bằng cách sử dụng `Image.fromarray(cropped\_image\_array)` và trả về kết quả.



Hình : Hàm cắt ảnh ở trung tâm.

* Hàm cắt ảnh theo khung tròn: Hàm `circle\_frame` thực hiện việc cắt hình ảnh theo khung hình tròn:

+ Hàm chấp nhận hình ảnh đầu vào và một tham số `mode`, mặc định là 0. Tham số `mode` xác định cách tính bán kính của khung hình tròn cần cắt. Khi `mode = 0`, bán kính của khung là giá trị nhỏ nhất giữa chiều cao và chiều rộng của hình ảnh. Khi `mode = 1`, bán kính của khung là giá trị lớn nhất giữa chiều cao và chiều rộng của hình ảnh.

+ Đầu tiên, hàm chuyển đổi hình ảnh thành một mảng NumPy bằng cách sử dụng `np.array(image)`.

+ Sau đó, hàm tính toán tâm của hình ảnh (center) bằng cách lấy giữa các giá trị chiều cao và chiều rộng chia cho 2.

+ Tiếp theo, hàm tính toán bán kính của khung hình tròn (radius) dựa trên tham số `mode`.

+ Sau đó, hàm sử dụng `np.ogrid[:height, :width]` để tạo ra các mảng 2D tương ứng với tọa độ x và y của các điểm ảnh trong hình ảnh.

+ Tiếp theo, hàm tạo ra một mảng mask\_circle, trong đó các giá trị True đại diện cho các điểm ảnh nằm bên ngoài khung hình tròn (nằm ngoài bán kính) và các giá trị False đại diện cho các điểm ảnh nằm bên trong khung hình tròn (nằm trong bán kính).

+ Sau đó, hàm cắt bớt phần hình ảnh nằm bên ngoài khung hình tròn bằng cách đặt các giá trị của các điểm ảnh nằm ngoài khung (có giá trị True trong mảng mask\_circle) thành 0.

+ Cuối cùng, hàm chuyển đổi mảng NumPy của hình ảnh đã cắt `image\_array` thành đối tượng PIL Image bằng cách sử dụng `Image.fromarray(image\_array)` và trả về kết quả.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, phần mềm

Mô tả được tạo tự động

Hình : Hàm cắt ảnh theo khung hình tròn.

* Hàm cắt ảnh theo khung là hai hình Elip chéo nhau: Hàm `ellipse\_frame` thực hiện việc cắt hình ảnh theo hai hình elip chồng nhau và đặt ảnh chéo vào giữa ảnh gốc:

+ Hàm chấp nhận hình ảnh đầu vào và hai tham số `scale` và `angle`. Tham số `scale` xác định tỉ lệ kích thước của hình elip so với kích thước nhỏ nhất giữa chiều cao và chiều rộng của ảnh. Tham số `angle` xác định góc xoay của hình elip.

+ Đầu tiên, hàm chuyển đổi hình ảnh thành một mảng NumPy bằng cách sử dụng `np.array(image)`. Sau đó, hàm tính toán các thông số cần thiết để xoay hình elip dựa trên tham số `angle`.Tiếp theo, hàm xác định kích thước cạnh của hình elip (edge) dựa trên tham số `scale`.

+ Sau đó, hàm tạo hai mặt nạ elip (`mask\_ellipse1` và `mask\_ellipse2`) bằng cách sử dụng các phương trình elip và các thông số đã tính toán. Tiếp theo, hàm tính toán vị trí mới tương ứng với `x\_ellipse` và `y\_ellipse` sau khi xoay hình elip.

+ Sau đó, hàm chuyển đổi các mặt nạ elip đã tính toán thành mặt nạ mới sau khi xoay (`rotated\_mask\_ellipse`).

+ Tiếp theo, hàm tính toán vị trí để đưa hình elip chéo vào giữa ảnh gốc. Sau đó, hàm áp dụng mặt nạ đã tính toán với ảnh gốc để cắt hình ảnh theo hai hình elip.

+ Cuối cùng, hàm chuyển đổi mảng NumPy của ảnh đã cắt `image\_array` thành đối tượng PIL Image bằng cách sử dụng `Image.fromarray(image\_array)` và trả về kết quả.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự động

Hình : Hàm cắt ảnh theo khung hai hình elip chéo nhau.

1. Test case:

Ảnh có chứa văn bản, ảnh chụp màn hình, ảnh ghép tổng hợp

Mô tả được tạo tự động

Hình : Test case 1.

Ảnh có chứa văn bản, Mặt người, ảnh chụp màn hình, ảnh ghép tổng hợp

Mô tả được tạo tự động

Hình : Test case 2.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

Hình : Test case 3. Ảnh có chứa ảnh chụp màn hình, màu tím, hoa tím, văn bản

Mô tả được tạo tự động

Hình : Test case 4.

Ảnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự động

Hình : Test case 5.