University of Science

Report

Lab01

Giảng viên: *Phạm Minh Hoàng*
Giảng viên: *Nguyễn Mạnh Hùng*
Giảng viên: *Lý Quốc Ngọc*

Sinh viên: *Huỳnh Cao Minh*
Date: *18/11/2023*

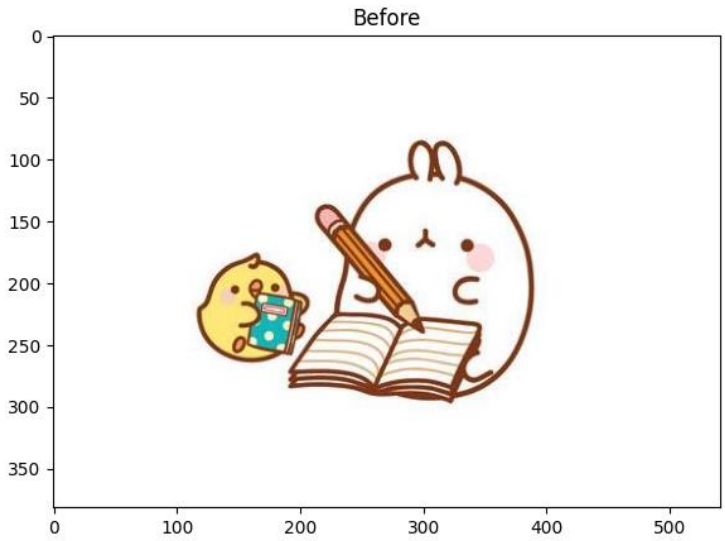
1. Bảng tiến độ thực hiện:

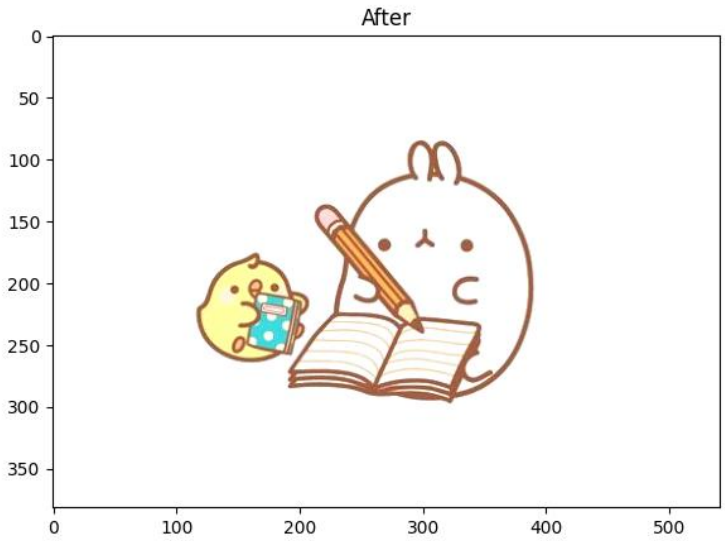
STT	Công việc thực hiện	Tiến độ hoàn thành
Transform colors and geometry		
1	Tăng/giảm độ sáng ảnh	100%
2	Tăng/giảm tương phản	100%
3	Lật ảnh ngang	100%
4	Lật ảnh dọc	100%
Image Smoothing and Blurring		
5	Làm sắc nét ảnh	100%
6	Làm mờ ảnh	100%

2. Nội dung thực hiện:

- **Tăng/giảm độ sáng ảnh**

  Hàm adjust_brightness được thiết kế để điều chỉnh độ sáng của một hình ảnh và so sánh hình ảnh gốc với hình ảnh đã điều chỉnh. Dưới đây là tóm tắt ngắn gọn về hàm này:
  Input (Đầu vào):
  img: Mảng NumPy biểu diễn hình ảnh gốc.
  img_file: Tên tệp hình ảnh (có thể không được sử dụng trong hàm).
  Output (Đầu ra):
  Không có giá trị trả về. Hàm này thực hiện việc hiển thị hình ảnh gốc và hình ảnh đã được điều chỉnh.
  Cách hoạt động:
  Người dùng sẽ được yêu cầu nhập vào một giá trị độ sáng muốn tăng thông qua hàm input.
  Hàm sử dụng thư viện NumPy để thực hiện điều chỉnh độ sáng bằng cách cộng giá trị độ sáng nhập vào với từng pixel của hình ảnh gốc.
  Kết quả sau khi điều chỉnh sẽ được kiểm tra và chuyển đổi về định dạng 8-bit không dấu (uint8).
  Hình ảnh gốc và hình ảnh đã được điều chỉnh được so sánh bằng cách gọi hàm compare.
  Ví dụ: Ảnh có tên đường dẫn "Test.jpg", với độ sáng được tăng là 40:





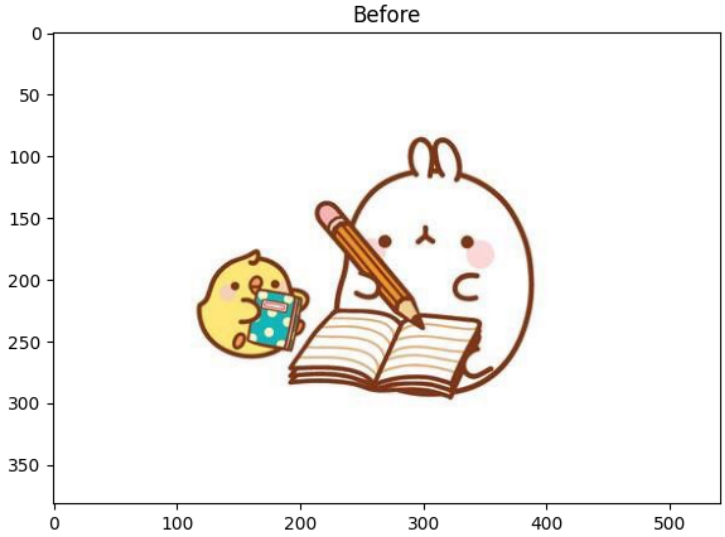
- **Tăng/giảm độ tương phản ảnh**

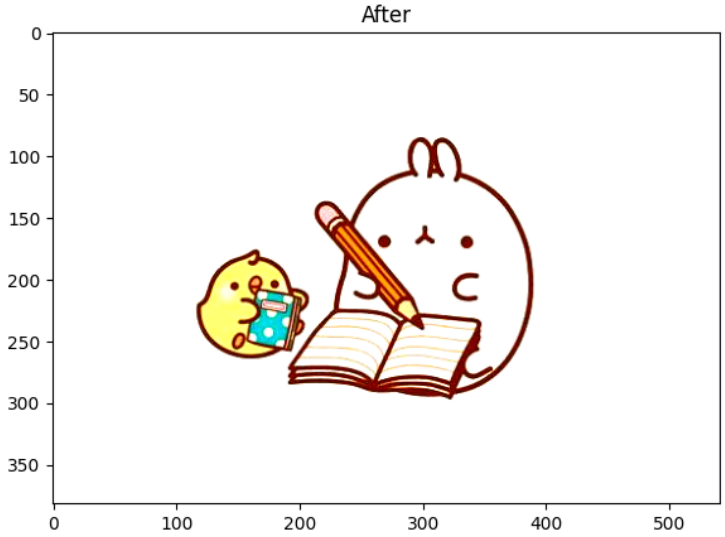
  Hàm adjust_contrast

Input: Hàm này nhận vào một ảnh (img) và một giá trị cons đại diện cho độ tương phản muốn điều chỉnh.
Output: Trả về một ảnh mới sau khi đã điều chỉnh độ tương phản.

Cách hoạt động:
np.clip(float(cons), -255, 255): Đảm bảo giá trị của cons nằm trong khoảng từ -255 đến 255.
factor = (259 * (cons + 255)) / (255 * (259 - cons)): Tính toán một hệ số dựa trên giá trị cons.
np.uint8(np.clip(factor * (img.astype(float) - 128) + 128, 0, 255)): Áp dụng hệ số này để điều chỉnh độ tương phản của ảnh (img). Cụ thể, giảm giá trị trung bình của pixel xuống mức 128, áp dụng hệ số và đảm bảo rằng giá trị pixel sau điều chỉnh nằm trong khoảng từ 0 đến 255.

Hàm adjust_contrast_running
Input: Hàm này nhận vào một ảnh (img) và tên file ảnh (img_file) từ đối số. Đồng thời, yêu cầu người dùng nhập giá trị độ tương phản muốn chỉnh sửa từ bàn phím.
Output: Không có giá trị trả về. Chỉ hiển thị và so sánh ảnh gốc và ảnh sau khi điều chỉnh tương phản.
Cách hoạt động:
Yêu cầu người dùng nhập giá trị độ tương phản (cons).
Gọi hàm adjust_contrast với giá trị cons nhập từ người dùng và ảnh img.
So sánh ảnh gốc (img) và ảnh sau khi điều chỉnh tương phản (res) bằng hàm compare
Ví dụ: Ảnh có tên đường dẫn “Test.jpg”, với độ tương phản được tăng là 70:

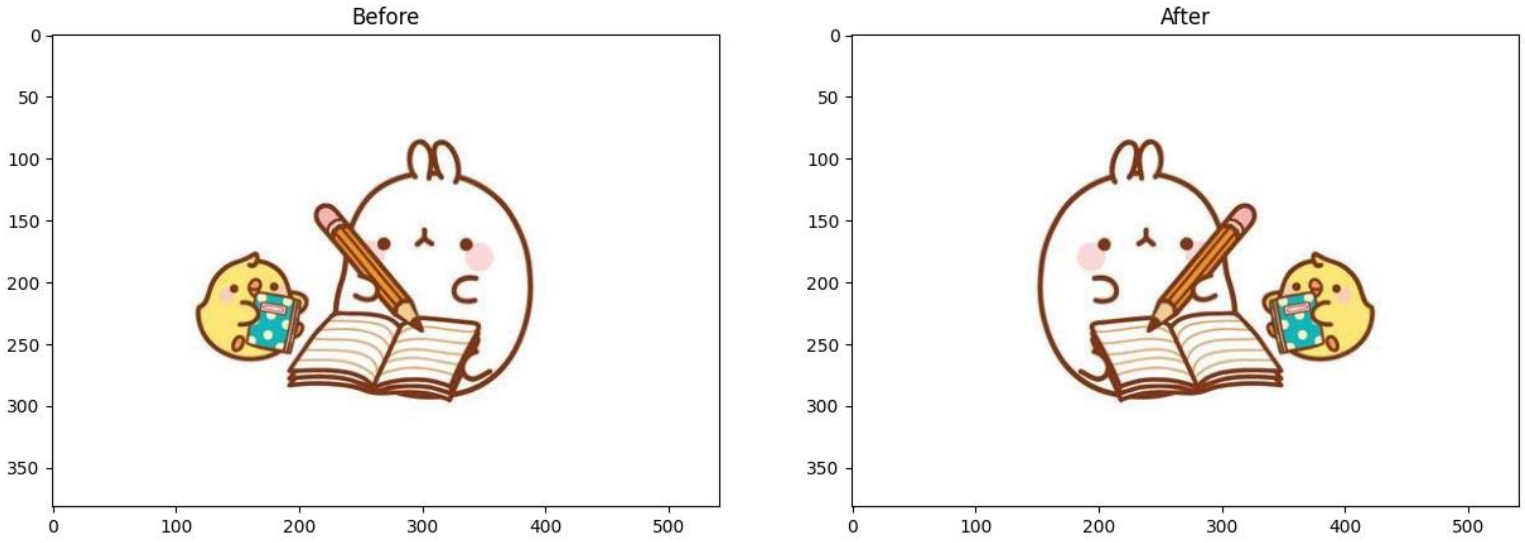




- **Lật ảnh ngang**

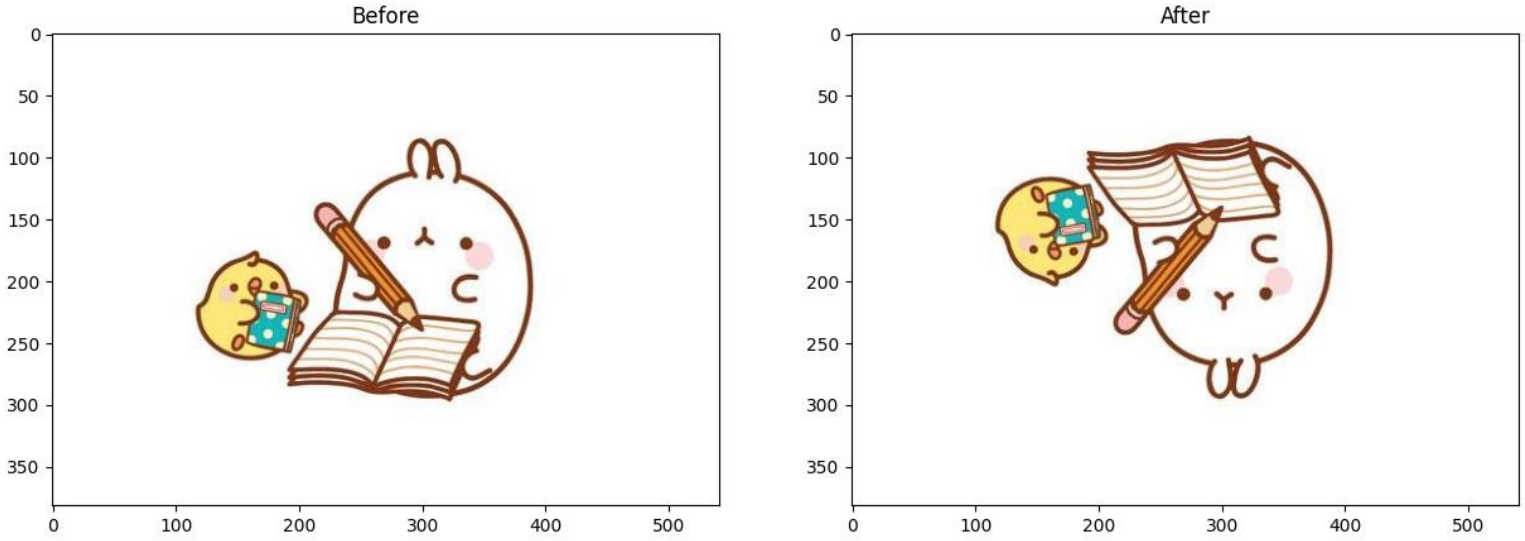
Hàm flip_image_horizontal(img):
Input: Một ma trận 2D (img) biểu diễn ảnh.
Output: Một ma trận 2D mới, được tạo ra bằng cách lật ngang (theo chiều ngang) ma trận đầu vào.
Cách hoạt động: Sử dụng biểu thức generator comprehension để tạo ma trận mới bằng cách lấy các giá trị từ ma trận đầu vào theo thứ tự ngược chiều ngang.
Hàm flip_image_horizontal_running(img, img_file):
Input:
img: Một ma trận 2D biểu diễn ảnh.
img_file: Tên của tệp ảnh
Output: Không có giá trị trả về (None).

Cách hoạt động: Gọi hàm flip_image_horizontal để lật ngang ma trận img, sau đó chuyển kết quả thành mảng NumPy (res)
Ví dụ: Ảnh có tên đường dẫn "Test.jpg":



- **Lật ảnh dọc**

Hàm flip_image_vertical nhận một danh sách img và trả về một danh sách mới là kết quả của việc đảo ngược các phần tử của img theo chiều dọc.
Hàm flip_image_vertical_running nhận hai tham số: img là danh sách cần đảo ngược và img_file là một tham số không được sử dụng trong hàm này. Hàm sử dụng flip_image_vertical để đảo ngược danh sách img, sau đó chuyển kết quả thành một mảng NumPy (res). Cuối cùng, nó gọi một hàm compare với hai tham số là img và res.
Ví dụ: Ảnh có tên đường dẫn "Test.jpg":



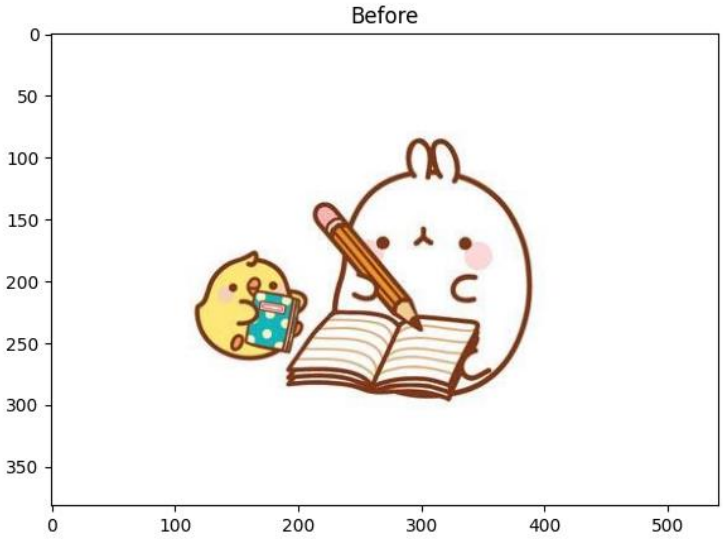
- **Làm mờ ảnh:**

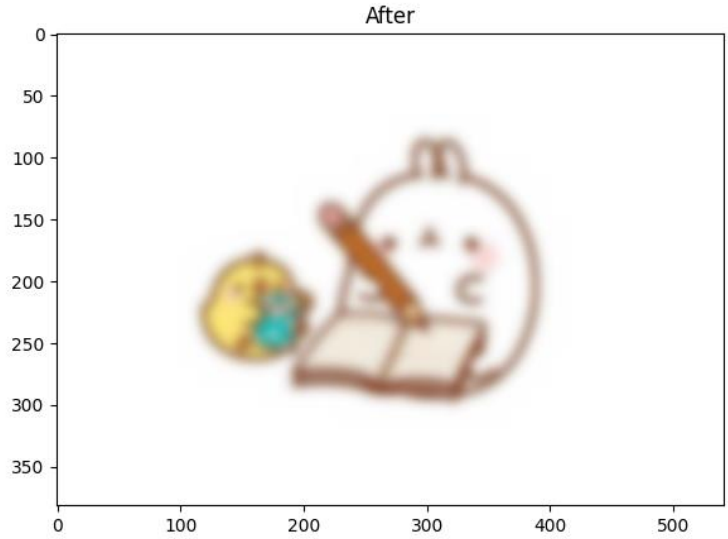
Hàm Gaussian_kernel(kernel_size, sigma):
Input: kernel_size là kích thước của ma trận lọc, sigma là tham số của hàm Gaussian.
Output: Ma trận lọc Gaussian 2D đã được chuẩn hóa.
Cách hoạt động:
Tạo một vector 1D từ -(kernel_size // 2) đến (kernel_size // 2).
Áp dụng hàm dnorm để tạo vector kernel 1D đã chuẩn hóa.
Tích hợp vector 1D để tạo ma trận 2D.
Chuẩn hóa ma trận 2D sao cho tổng các phần tử bằng 1.

Hàm dnorm(kernel_size, sigma):
Input: kernel_size là kích thước của vector lọc, sigma là tham số của hàm Gaussian.
Output: Vector 1D được tính theo hàm Gaussian.
Cách hoạt động: Tính toán giá trị của hàm Gaussian 1D cho mỗi phần tử trong vector.

Hàm Gaussian_blur_img(img, kernel_size):
Input: img là ảnh đầu vào, kernel_size là kích thước của ma trận lọc Gaussian.
Output: Ảnh đã được làm mờ.
Cách hoạt động:
Tạo ma trận lọc Gaussian bằng cách gọi hàm Gaussian_kernel.
Tạo một ảnh trắng blur_img để lưu kết quả.
Mở rộng ảnh đầu vào bằng cách thêm các hàng và cột vào biên theo kích thước của ma trận lọc.
Áp dụng lọc Gaussian trên từng kênh màu và từng pixel trong ảnh đầu vào.

Hàm blur_img_running(img, img_file):
Input: img là ảnh đầu vào, img_file là tên tệp ảnh
Output: Hiển thị ảnh đã được làm mờ.
Cách hoạt động: Nhận độ mờ muốn chỉnh sửa từ người dùng thông qua nhập từ bàn phím và gọi hàm Gaussian_blur_img để làm mờ ảnh với độ mờ đã chọn. Hiển thị ảnh gốc và ảnh đã làm mờ để so sánh.
Ví dụ: Ảnh có tên đường dẫn "Test.jpg", với độ mờ là 50:





- **Làm sắc nét ảnh:**

img: Một hình ảnh màu được biểu diễn dưới dạng mảng NumPy với kích thước (chiều cao, chiều rộng, 3), trong đó 3 đại diện cho các kênh màu RGB.
Output:
sharpened_color_image: Một phiên bản được làm sắc nét của hình ảnh đầu vào. Đây là một mảng NumPy có cùng hình dạng với đầu vào, biểu diễn hình ảnh màu được làm sắc nét.
Cách Hoạt Động của Hàm:
Hàm sharp_img nhận một hình ảnh màu img làm đầu vào.
Nó xác định một ma trận làm sắc nét kích thước 3x3.
Chuyển đổi hình ảnh màu đầu vào thành hình ảnh xám (gray_img) bằng cách lấy trung bình qua các kênh màu RGB.

Khởi tạo một mảng sharpened_image với giá trị zero, nơi sẽ lưu trữ hình ảnh xám được làm sắc nét.
Áp dụng ma trận làm sắc nét cho mỗi pixel của hình ảnh xám bằng cách sử dụng một vòng lặp lồng nhau, loại bỏ biên của hình ảnh.
Hình ảnh làm sắc nét kết quả được cắt giữ trong khoảng [0, 255] và chuyển đổi thành số nguyên 8 bit không dấu.
Hình ảnh màu cuối cùng được làm sắc nét (sharpened_color_image) được tạo ra bằng cách xếp ba bản sao của hình ảnh xám được làm sắc nét theo trục thứ ba để phù hợp với định dạng RGB.
Hàm trả về sharpened_color_image.
Ví dụ: Ảnh có tên đường dẫn “Test.jpg”: