INT3404E 20- Image Processing Homework 1

Lê Thị Hải Anh - 22028162

March 2024

1 Bài toán

Hình 1: Original image

Một bức ảnh có thể được biểu diễn dưới dạng một mảng NumPy của các "pixel", với kích thước H × W × C, trong đó H là chiều cao, W là chiều rộng và C là số kênh màu. Hình 1 minh họa hệ tọa độ. Gốc tọa độ nằm ở góc trên bên trái và chiều đầu tiên chỉ định hướng Y (hàng), trong khi chiều thứ hai chỉ định chiều X (cột). Thông thường, chúng ta sẽ sử dụng một bức ảnh với các kênh màu đại diện cho mức đỏ, xanh lá cây và xanh dương của mỗi pixel, được gọi theo cách viết tắt là RGB. Giá trị cho mỗi kênh dao động từ 0 (tối nhất) đến 255 (sáng nhất). Tuy nhiên, khi tải một ảnh thông qua Matplotlib, phạm vi này sẽ được tỷ lệ từ 0 (tối nhất) đến 1 (sáng nhất) thay vì là một số nguyên, và sẽ là một số thực.

Viết mã Python để tải một bức ảnh, thực hiện một số thao tác trên ảnh và trực quan hóa các hiệu ứng của chúng.

2 Báo cáo kết quả

Về tổng quan, bài toán được giải quyết bởi ngôn ngữ Python cùng với các thư viện OpenCV, MathPlotLib và Numpy để xử lý hình ảnh.

```
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np
```

Với hàm thực thi được trình bày ở mục 2.3, hình ảnh kết quả của các hàm xử lý (trình bày ở mục 2.2) sẽ được đưa ra.

2.1 Các hàm cơ bản

2.1.1 Hàm tải ảnh

Sử dụng hàm *imread*() của thư viện OpenCV để đọc ảnh

```
def load_image(image_path):
    return cv.imread(image_path)
```

2.1.2 Hàm hiển thị ảnh

Khi sử dụng OpenCV để đọc ảnh, nó sẽ đọc ảnh dưới dạng BGR (Blue-Green-Red), trong khi Matplotlib hiển thị ảnh dưới dạng RGB (Red-Green-Blue). Nên cần chuyển đổi không gian màu của ảnh từ BGR sang RGB trước khi hiển thị nó bằng Matplotlib.

```
def display_image(image, title="Image"):
    image_rgb = cv.cvtColor(image, cv.COLOR_BGR2RGB)

    plt.imshow(image_rgb)
    plt.title(title)
    plt.axis('off')
    plt.show()
```

2.1.3 Hàm lưu ảnh

Sử dụng hàm *imwite*() của thư viện OpenCV để lưu ảnh

```
def save_image(image, output_path):
    cv.imwrite(output_path, image)
```

2.2 Các hàm xử lý ảnh

2.2.1 Hàm chuyển sang ảnh xám

Chuyển đổi một hình ảnh thành hình ảnh xám. Chuyển đổi hình ảnh gốc thành hình ảnh xám. Trong hình ảnh xám, giá trị pixel của 3 kênh sẽ giống nhau cho một tọa độ X, Y cụ thể. Phương trình cho giá trị pixel [1] được cho bởi: $p = 0.299R + 0.587G + 0.114B$ Trong đó R, G, B là các giá trị cho mỗi kênh tương ứng. Chúng ta sẽ thực hiện điều này bằng cách tạo một mảng gọi là *img_gray* có cùng hình dạng như img. Kết quả thực thi hàm này thể hiện qua Hình 2.

```
def grayscale_image(image):
    height, width = image.shape[:2]
    img_gray = np.zeros((height, width), dtype=np.uint8)

    for i in range(height):
        for j in range(width):
            R, G, B = image[i, j]
            p = 0.299 * R + 0.587 * G + 0.114 * B
            img_gray[i, j] = p

    return img_gray
```

Hình 2: Gray image

Hình 3: Gray flipped image

2.2.2 Hàm lật ảnh

Sử dụng hàm *flip*() của thư viện OpenCV để lật ảnh. Kết quả thực thi hàm này được thể hiện qua Hình 3.

```
def flip_image(image):
    flipped_img = cv.flip(image, 1)
    return flipped_img
```

2.2.3 Hàm xoay ảnh

Để xoay ảnh, cần xác định tâm xoay và tạo ma trận quay sử dụng hàm getRotationMatrix2D(). Sau đó, ma trận này được áp dụng vào ảnh thông qua hàm warpAffine() của thư viện OpenCV, kết quả sau khi thực thi hàm này được minh họa qua Hình 4.

```
def rotate_image(image, angle):
    height, width = image.shape[:2]
    center = (width / 2, height / 2)
    rotation_matrix = cv.getRotationMatrix2D(center, angle, 1)
    rotated_image = cv.warpAffine(image, rotation_matrix, (width, height))
    return rotated_image
```

Hình 4: Gray rotated image

2.3 Hàm thực thi

```
if __name__ == "__main__":
    i# Load an image from file
    img = load_image("hw1/images/uet.png")

    # Display the image
    display_image(img, "Original Image")

    # Convert the image to grayscale
    img_gray = grayscale_image(img)

    # Display the grayscale image
    display_image(img_gray, "Grayscale Image")

    # Save the grayscale image
    save_image(img_gray, "hw1/images/gray.jpg")

    # Flip the grayscale image
    img_gray_flipped = flip_image(img_gray)

    # Display the flipped grayscale image
    display_image(img_gray_flipped, "Flipped Grayscale Image")

    # Save the rotated grayscale image
    save_image(img_gray_flipped, "hw1/images/gray_flipped.jpg")

    # Rotate the grayscale image
    img_gray_rotated = rotate_image(img_gray, 45)

    # Display the rotated grayscale image
    display_image(img_gray_rotated, "Rotated Grayscale Image")

    # Save the rotated grayscale image
    save_image(img_gray_rotated, "hw1/images/gray_rotated.jpg")

    # Show the images
    plt.show()
```