INT3404E 20 - Image Processing: Homeworks 2

Lê Xuân Hùng - 22028172

1 Đề bài

Figure 1: Ảnh gốc

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

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):
    """
    Load an image from file, using OpenCV
    """
    image = cv2.imread(image_path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    return image / 255.0
```

2.1.2 Hàm hiển thị ảnh

```
def display_image(image, title="Image"):
    """
    Display an image using matplotlib. Rembember to use plt.show() to display the image
    """
    plt.figure(figsize=(8, 6))
    plt.imshow(image)
    plt.title(title)
    plt.axis('off')
```

2.1.3 Hàm lưu ảnh

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

```
def save_image(image, output_path):
    """
    Save an image to file using OpenCV
    """
    image = (image * 255).astype(np.uint8)
    cv2.imwrite(output_path, cv2.cvtColor(image, cv2.COLOR_RGB2BGR))
```

2.2 Các hàm xử lý - biến đổi ảnh

Figure 2: Gray image

Figure 3: Gray flipped image

Figure 4: Gray rotated image

2.2.1 Hàm chuyển sang ảnh xám

Quá trình chuyển đổi ảnh màu thành ảnh xám là biến đổi ảnh gốc sang dạng không màu. Trong ảnh xám, giá trị điểm ảnh của các kênh màu sẽ trùng nhau tại một vị trí (x, y) nhất định. Công thức tính toán giá trị điểm ảnh như sau: $p = 0.299R + 0.587G + 0.114B$
Trong đó, R, G, B tương ứng là giá trị của các thành phần màu đỏ, xanh lá cây và xanh lam. Để thực hiện, ta sẽ khởi tạo một mảng mới img_gray có kích thước tương đương ảnh nguồn img. Kết quả thu được sau khi áp dụng phép biến đổi được minh họa qua Hình 2.

```
def grayscale_image(image):
    """
    Convert an image to grayscale.
    Convert the original image to a grayscale image.
    In a grayscale image, the pixel value of the 3 channels will be the same for a particular
    X, Y coordinate.
    The equation for the pixel value [1] is given by:
        p = 0.299R + 0.587G + 0.114B
    Where the R, G, B are the values for each of the corresponding channels. We will do this by
    creating an array called img_gray with the same shape as img
    """
    img_gray = np.dot(image, [0.299, 0.587, 0.114])
    return img_gray
```

2.2.2 Hàm lật ảnh

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

```
def flip_image(image):
    """
    Flip an image horizontally using OpenCV
    """
    return cv2.flip(image, 1)
```

2.2.3 Hàm xoay ảnh

Để thực hiện phép xoay ảnh, ta cần xác định trung tâm quay và tính toán ma trận xoay bằng cách sử dụng hàm $getRotationMatrix2D()$. Tiếp theo, ma trận xoay này sẽ được áp dụng lên ảnh nguồn thông qua việc gọi hàm $warpAffine()$ của thư viện OpenCV, kết quả sau khi thực hiện được minh họa ở Hình 4.

```
def rotate_image(image, angle):
    """
    Rotate an image using OpenCV. The angle is in degrees
    """
    rows, cols = image.shape[:2]
    M = cv2.getRotationMatrix2D((cols / 2, rows / 2), angle, 1)
    return cv2.warpAffine(image, M, (cols, rows))
```

2.3 Hàm thực thi

```python
if __name__ == "__main__":
    # Load an image from file
    img = load_image("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, "lena_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 flipped grayscale image
    save_image(img_gray_flipped, "lena_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, "lena_gray_rotated.jpg")

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