INT3404E 20 - Image Processing: Homeworks 2

Nguyen Thi Thu Trang

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 Hàm tải ảnh

Hàm *load_image*() nhận đầu vào là đường dẫn của hình ảnh và sử dụng hàm *imread*() của thư viện OpenCV để đọc hình ảnh từ tệp. Sau đó, nó trả về hình ảnh đã được đọc.

```
# Load an image from file as function
def load_image(image_path):
    """
    Load an image from file, using OpenCV
    """
    img = cv2.imread(image_path)
    return img
```

2.2 Hàm hiển thị ảnh

Hàm *display_image*() nhận một hình ảnh làm đầu vào và sử dụng thư viện matplotlib để hiển thị hình ảnh. Hàm này sử dụng *plt.imshow*() để hiển thị hình ảnh với màu sắc chính xác bằng cách chuyển đổi từ không gian màu BGR sang RGB, sau đó đặt tiêu đề và tắt trục. Cuối cùng, nó sử dụng *plt.show*() để hiển thị hình ảnh ra màn hình.

```
# Display an image as function
def display_image(image, title="Image"):
    """
    Display an image using matplotlib. Rembember to use plt.show() to display the image
    """
    plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    plt.title(title)
    plt.axis('off')
    plt.show()
```

2.3 Hàm lưu ảnh

Hàm *save_image*() nhận một hình ảnh và đường dẫn đầu ra, sau đó sử dụng hàm *imwite*() của thư viện OpenCV để lưu hình ảnh đó vào tệp với đường dẫn đã được chỉ định.

```
# Save an image as function
def save_image(image, output_path):
    """
    Save an image to file using OpenCV
    """
    cv2.imwrite(output_path, image)
```

2.4 Hàm chuyển đổi thành ảnh xám

Hàm *grayscale_image*() nhận một hình ảnh và sử dụng OpenCV để chuyển đổi nó thành hình ảnh xám. Điều này được thực hiện bằng cách sử dụng *cv2.cvtColor*() để chuyển đổi không gian màu của hình ảnh từ BGR sang grayscale. Cuối cùng, hàm trả về hình ảnh xám đã được chuyển đổi.

```
# grayscale an image as function
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 = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    return img_gray
```

Figure 2: Gray image

2.5 Hàm lật ảnh

Hàm $flip\_image()$ nhận một hình ảnh và sử dụng OpenCV để lật nó theo chiều ngang. Điều này được thực hiện bằng cách sử dụng $cv2.flip()$ với tham số thứ hai là 1 để chỉ định lật theo chiều ngang. Cuối cùng, hàm trả về hình ảnh đã được lật.

```
# flip an image as function
def flip_image(image):
    """
    Flip an image horizontally using OpenCV
    """
    img_flipped = cv2.flip(image, 1)
    return img_flipped
```

Figure 3: Gray flipped image

2.6 Hàm xoay ảnh

Hàm $rotate\_image()$ nhận một hình ảnh và một góc quay, sau đó sử dụng OpenCV để xoay hình ảnh theo góc được chỉ định. Đầu tiên, nó tính toán ma trận quay M bằng cách sử dụng $cv2.getRotationMatrix2D()$. Sau đó, nó sử dụng $cv2.warpAffine()$ để áp dụng phép biến đổi xoay vào hình ảnh. Cuối cùng, hàm trả về hình ảnh đã được xoay.

```
# rotate an image as function
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)
    img_rotated = cv2.warpAffine(image, M, (cols, rows))
    return img_rotated
```

Figure 4: Gray rotated image

2.7 Hàm thực thi

```
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()
```