Project 1: Tách ảnh xám từ ảnh màu và ngược lại

Nguyễn Trang Sỹ Lâm

Ngày 20 tháng 9 năm 2024

Môn học: Xử lý Ảnh số và Thị giác Máy tính
Giảng viên hướng dẫn: Võ Thanh Hùng

1 Giới thiệu

Trong bài toán này, chúng ta sẽ thực hiện hai nhiệm vụ chính:

- Tách ảnh xám từ ảnh màu.
- Kết hợp các kênh màu để tạo ảnh màu từ các ảnh xám.

Việc này có thể ứng dụng trong nhiều lĩnh vực như xử lý hình ảnh, nhận dạng đối tượng hoặc cải thiện chất lượng hình ảnh. Chúng ta sẽ sử dụng ngôn ngữ lập trình Python để hiện thực các bước này.

2 Công thức toán

2.1 Lý thuyết về tách các kênh màu từ ảnh màu

Khi tách các kênh màu từ ảnh màu (RGB), không có công thức toán học phức tạp. Mỗi pixel trong ảnh màu bao gồm ba giá trị: một giá trị cho kênh Đỏ (Red), một giá trị cho kênh Xanh Lá (Green), và một giá trị cho kênh Xanh Dương (Blue).

2.1.1 Cấu trúc của một pixel trong ảnh màu

Trong một ảnh màu, mỗi pixel được biểu diễn dưới dạng một bộ ba giá trị (R, G, B) như sau:

$$\text{Pixel}(x, y) = (R(x, y), G(x, y), B(x, y))$$

Với $R(x, y)$, $G(x, y)$, và $B(x, y)$ là giá trị của các kênh đỏ, xanh lá, và xanh dương tại tọa độ (x, y).

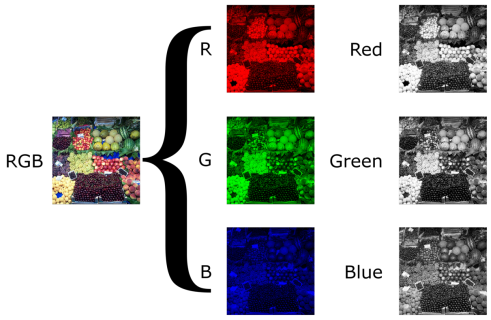
2.1.2 Tách kênh

Khi tách kênh màu, ta đơn giản lấy riêng từng kênh trong ba kênh đó để tạo ra ba ảnh riêng biệt, mỗi ảnh chỉ chứa giá trị của một kênh (Đỏ, Xanh Lá, Xanh Dương).

Ví dụ:

Kênh đỏ: $R(x, y)$, Kênh xanh lá: $G(x, y)$, Kênh xanh dương: $B(x, y)$

Các giá trị này được lưu dưới dạng ảnh xám (grayscale) tương ứng với mức độ sáng của từng kênh màu.



Hình 1: Thành phần RGB từ ba ảnh xám bởi Nevit Dilmen.

2.2 Tách ảnh xám

Một cách đơn giản để chuyển một mảng 3D ảnh màu sang mảng 2D ảnh xám là lấy giá trị trung bình của các kênh đỏ, xanh lá, và xanh dương để tạo ra giá trị ảnh xám cho mỗi pixel. Phương pháp này kết hợp độ sáng (luminance) do mỗi dải màu đóng góp để tạo ra một xấp xỉ hợp lý cho màu xám.

2.2.1 Nhận thức độ chói phụ thuộc vào kênh màu

Thực tế, mắt người nhạy cảm hơn với màu xanh lá so với màu xanh dương. Qua nhiều thử nghiệm, các nhà nghiên cứu đã cung cấp các trọng số khác nhau cho từng kênh màu để tính toán tổng độ chói chính xác hơn:

$$Y_{\text{linear}} = 0.2126R_{\text{linear}} + 0.7152G_{\text{linear}} + 0.0722B_{\text{linear}}$$

2.2.2 Nén gamma

Để mắt nhận ra sự khác biệt nhỏ khi độ chói thấp, nhưng lại ít nhạy cảm hơn với độ chói cao, ta áp dụng nén gamma, nén lại các giá trị ở dải thấp hơn và giãn rộng ở dải cao hơn. Trước khi tính toán độ chói grayscale, cần phải áp dụng giải nén gamma (gamma expansion):

$$C_{\text{linear}} = \begin{cases} \frac{C_{\text{srgb}}}{12.92}, & C_{\text{srgb}} \leq 0.04045 \\ \left(\frac{C_{\text{srgb}}+0.055}{1.055}\right)^{2.4}, & C_{\text{srgb}} > 0.04045 \end{cases}$$

2.2.3 Xấp xỉ tuyến tính

Việc nén và giải nén gamma có thể tốn kém. Trong các tình huống cần tốc độ xử lý nhanh, phương pháp xấp xỉ tuyến tính đơn giản hơn có thể được sử dụng:

$$Y_{\text{approx}} = 0.2989R + 0.5870G + 0.1140B$$

Phương pháp này được sử dụng rộng rãi trong các thư viện xử lý ảnh như OpenCV và Pillow.

3 Hiện thực

Chúng ta sẽ sử dụng thư viện Pillow (PIL) và NumPy để thực hiện các bước xử lý ảnh.

3.1 Tách ảnh xám từ ảnh màu

```
from PIL import Image
import numpy as np

# Load the color image
img_color = np.array(Image.open('color_image.jpg'))

# Convert the color image to grayscale using weighted sum
img_gray = 0.2989 * img_color[:,:,0] +
0.5870 * img_color[:,:,1] +
0.1140 * img_color[:,:,2]

# Save the grayscale image
```

```
Image.fromarray(img_gray.astype(np.uint8)).save('gray_image.jpg')
```

3.2 Tách các kênh màu từ ảnh màu

```
# Load the color image
img_color = np.array(Image.open('color_image.jpg'))

# Separate the color channels
R = img_color[:,:,0]
G = img_color[:,:,1]
B = img_color[:,:,2]

# Save each channel as a grayscale image
Image.fromarray(R).save('gray_R.jpg')
Image.fromarray(G).save('gray_G.jpg')
Image.fromarray(B).save('gray_B.jpg')
```

3.3 Tạo ảnh màu từ các channel xám

```
# Load the grayscale images
gray_R = np.array(Image.open('gray_R.jpg'))
gray_G = np.array(Image.open('gray_G.jpg'))
gray_B = np.array(Image.open('gray_B.jpg'))

# Stack the grayscale images into a color image
img_color_reconstructed = np.stack([gray_R, gray_G, gray_B], axis=-1)

# Save the reconstructed color image
Image.fromarray(img_color_reconstructed.astype(np.uint8))
.save('color_image_reconstructed.jpg')
```

4 Kết quả và Thảo luận

4.1 Tách ảnh xám từ ảnh màu

Hình dưới đây là kết quả của quá trình chuyển ảnh màu sang ảnh xám:

Hình 2: Ảnh màu gốc.

Hình 3: Ảnh xám tách từ ảnh màu.

4.2 Tách các kênh màu từ ảnh màu

Sau khi tách các kênh màu từ ảnh màu ban đầu, ta có các ảnh xám tương ứng với từng kênh màu:

Hình 4: Kênh đỏ dưới dạng ảnh xám.

Hình 5: Kênh xanh lá dưới dạng ảnh xám.

Hình 6: Kênh xanh dương dưới dạng ảnh xám.

4.3 Tạo ảnh màu từ các kênh xám

Sau khi kết hợp các kênh xám để tái tạo lại ảnh màu, ta thu được hình ảnh sau:

Hình 7: Ảnh màu được tái tạo từ các kênh xám.

5 Kết luận

Chúng ta đã thực hiện thành công hai nhiệm vụ chính của bài toán: tách ảnh xám từ ảnh màu, và tái tạo lại ảnh màu từ các kênh xám. Công thức luma theo tiêu chuẩn ITU-R BT.601-2 đã được áp dụng để chuyển đổi ảnh màu sang ảnh xám, giúp thể hiện độ sáng của hình ảnh theo cách mắt người cảm nhận.

6 Code đầy đủ

Mã nguồn đầy đủ có thể truy cập tại GitHub: ComputerVisionAssignment.