TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

Báo cáo đồ án 1: Color Compression

Thông tin sinh viên:

Họ và tên: Tô Quốc Thanh

MSSV: 22127388

Lớp: 22CLC10

Mục lục

I. Ý tưởng thực hiện 3
II. Mô tả các hàm 3
III. Hình ảnh kết quả 5
IV. Nhận xét 6
V. Tài liệu tham khảo 6

I. Ý tưởng thực hiện

Dựa trên các prototype hàm được cung cấp trước từ giảng viên và ý tưởng của thuật toán K-Means [1], ta thực hiện đồ án này theo các bước sau:
-*Bước 1*: Import các thư viện cần thiết (numpy và matplotlib).
-*Bước 2*: Đọc hình ảnh từ thiết bị, chuyển từ mảng 2D thành mảng 1D chứa các list là mã màu của mỗi pixel.
-*Bước 3*: Áp dụng thuật toán K-Means với mảng vừa tạo được. Tiến hành phân cụm các pixel và tìm ra các màu trung tâm, đánh dấu nhãn cho từng pixel (sẽ trình bày rõ hơn trong phần mô tả hàm kmeas ở phần 2).
-*Bước 4*: Từ kết quả ở Bước 3, ta tạo hình ảnh mới bằng cách với mỗi pixel, ta tìm màu sắc của nó thông qua label của pixel, trong đó label là index trong mảng màu chủ đạo.

II. Mô tả các hàm

- ***read_img(img_path):*** Hàm đọc hình ảnh từ thiết bị [2.1]

Dựa vào hàm imread(img_path) có sẵn trong thư viện matplotlib, ta chỉ cần truyền vào một string ứng với đường dẫn của hình ảnh: img_path để đọc hình ảnh từ thiết bị.

- ***show_img(img_2d):*** Hàm hiển thị hình ảnh [2.2]

Hình ảnh mà ta đọc được ở trên là một mảng 2D chứa toàn bộ thông tin về bức ảnh (chiều cao x chiều rộng x kênh màu), ta dựa vào hàm imshow(array) có sẵn trong thư viện matplotlib để hiển thị hình ảnh.

- ***save_img(img_2d, img_path):*** Hàm lưu hình ảnh [2.3]

Tương tự như 2 hàm nói trên, thư viện matplotlib cũng có thư viện hỗ trợ việc lưu hình ảnh, đó là hàm imsave(img_path, arr) trong đó img_path là đường dẫn để lưu ảnh, arr tương ứng với mảng img_2d mà ta tạo ra.

- ***convert_img_to_1d(img_2d):*** Hàm chuyển từ mảng 2D sang 1D

```
def convert_img_to_1d(img_2d):
    new_img = [elements for rows in img_2d for elements in rows]
    return new_img
```

Bức hình gốc được tạo nên từ nhiều pixel là một mảng 2D, mỗi pixel lại chứa một mã màu RGB, vì thế để chuyển từ mảng 2D thành 1D, ta đơn giản là lấy từng pixel của mỗi hàng trong ảnh tạo thành 1 list, với mỗi phần tử là mã màu RGB của pixel tương ứng.

- ***kmeans(img_1d, k_clusters, max_iter, init_centroids):*** Thuật toán K-Means

Áp dụng thuật toán K-Means với mảng 1D mà ta tạo được, tuy nhiên mảng 1D là là list chứa các list, nhiều thao tác không được hỗ trợ, vì thế ta cần chuyển đổi nó thành mảng chứa các vector trong numpy. Ở đây ta không cần cứng nhắc với kiểu dữ liệu uint8 như lưu màu RGB thông thường, chỉ cần trả về dạng uint8 trước khi save hình ảnh là được.

Với k_clusters là số lượng màu mới của bức ảnh, max_iter là số lần lặp tối đa khi tiến hành phân cụm và init_centroids là cách thức khởi tạo mảng centroids.

Trước tiên, tùy theo giá trị của chuỗi init_centroids mà người dùng nhập vào, ta tạo random một số màu sắc hoặc chọn các màu sắc từ các pixel của hình ảnh. Tiếp theo, với mỗi pixel ta tìm xem nó gần nhất với màu nào trong mảng vừa tạo (nghĩa là có sự tương đồng về màu sắc nhất) và gán nhãn (label) cho nó, ta gọi hành động này là phân cụm.

Việc phân cụm sẽ được lặp đi lặp lại cho đến khi không có pixel nào thay đổi nhãn hoặc hết số lần lặp quy định. Sau khi kết thúc thuật toán, ta sẽ nhận được các màu chủ đạo của hình ảnh là trung bình cộng mã màu của các pixel có cùng label, và mảng labels chứa thông tin về label ứng với từng pixel.

Sẽ có trường hợp các màu được khởi tạo lúc đầu không thuộc các màu trung tâm, vì các màu này có tần suất xuất hiện thấp, có thể được xem đây là một điểm ngoại lai khi phân cụm, trường hợp này ta có thể trực tiếp bỏ qua kênh màu này, hoặc gán cho nó một giá trị bất kì, trong đồ án này em gán cho nó là giá trị trung bình của tất cả pixel, vì trái ngược với màu đang xét là ngoại vi, thì trung bình của các pixel là trung tâm của các màu, khiến cho kênh màu được chọn trở nên tương đồng hơn với các màu của các pixel xung quanh. Điều này có thể làm thuật toán chạy chậm hơn khi phải duyệt qua mảng các pixel, nhưng tỉ lệ xảy ra trường hợp đang xét là rất thấp. Giả sử với bức hình cơ bản có 1920 x 1080 pixel, tỉ lệ chọn được 1 kênh màu ngoại lai (nghĩa là không có màu nào gần giống nó) là 1/ (1920 x 1080), tỉ lệ tồn tại của nó cũng rất thấp khi hầu hết các bức ảnh đều có rất nhiều màu, mỗi pixel có thể có đến $256^3 \approx 1.7 \times 10^7$ màu (theo chuẩn RGB) [3].

- ***generate_2d_img(img_2d_shape, centroids, labels):*** Hàm tạo hình ảnh từ mảng 2D sau khi kết thúc thuật toán K-Means

Từ 2 mảng centroids và labels tìm được ở trên, ta tạo hình ảnh mới bằng cách với mỗi pixel, ta tìm màu sắc của nó thông qua label của pixel, trong đó label là index trong mảng màu chủ đạo.
Trước khi trả về hình ảnh mới, phải ép kiểu về uint8 để đảm bảo các pixel có giá trị màu từ 0 đến 255, vì đây là tham số đầu vào mà hàm imsave(img_path, arr) của thư viện matplotlib yêu cầu mà ta đã nói ở trên.

- ***is_valid_image_extension(file_name):*** Hàm kiểm tra phần mở rộng của tên file có hợp lệ không

Một số phần mở rộng mà chương trình hỗ trợ thuộc về các loại file hình ảnh phổ biến như: '.png', '.jpg', '.jpeg','.pdf',...
Nếu phần mở rộng không thuộc các loại file trên, chương trình không cho phép người dùng nhập hay lưu ảnh.

III. Hình ảnh kết quả

Hình ảnh gốc:

Centroids_init: Random
Đi từ trái sang phải, chỉ số k_cluster tương ứng của từng ảnh lần lượt là 3, 5, 7

Centroids_init: In pixel

Đi từ trái sang phải, chỉ số k_cluster tương ứng của từng ảnh lần lượt là 3, 5, 7

IV. Nhận xét

Thuật toán K-Means giúp giảm số màu của một bức ảnh, từ hàng ngàn, thậm chí là hàng chục ngàn màu sắc xuống còn 3 – 7 màu sắc (hoặc một số k_cluster tùy chọn). Tuy ảnh mới không được chi tiết như bức ảnh ban đầu, tuy nhiên nhìn chung bố cục của hình ảnh vẫn được giữ nguyên.

Với số k càng cao, bức hình sẽ càng có nhiều màu sắc, do đó độ chi tiết và tương đồng với hình ảnh gốc càng cao. Dĩ nhiên, thời gian tính toán, xử lý của thuật toán K-Means cũng cao hơn khi phải phân chia các pixel vào nhiều cụm màu hơn.

Tốc độ thuật toán bị ảnh hưởng khá nhiều bởi thao tác khởi tạo centroids, nếu các màu được tạo gần với các màu trung tâm, thuật toán phân cụm sẽ sớm kết thúc. Ngược lại sẽ phải thực hiện phân cụm nhiều lần hơn hoặc đến khi hết số lần lặp quy định.

V. Tài liệu tham khảo

[1] K means Clustering - Introduction - GeeksforGeeks

[2] Working with Images in Python using Matplotlib - GeeksforGeeks

[3] Project1- Applied Mathematics and Statistics

Special thanks to ChatGPT and Gemini for helping and providing information.