ĐẠI HỌC QUỐC GIA TPHCM

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

Project 01: Color Compression

Môn học: Toán ứng dụng và thống kê cho CNTT

Sinh viên thực hiện:

Lý Anh Quân (22127344)

Giáo viên hướng dẫn:

ThS. Vũ Quốc Hoàng

ThS. Phan Thị Phương Uyên

ThS. Nguyễn Văn Quang Huy

ThS. Nguyễn Ngọc Toàn

Ngày 18 tháng 6 năm 2024

Mục lục

1 Ý tưởng thực hiện

Một bức ảnh trong máy tính có thể được lưu trữ dưới dạng ma trận của các điểm ảnh. Vì thế khi máy tính đọc một bức ảnh màu, tức là chúng đang đọc một ma trận điểm ảnh với kích thước của ảnh và kênh màu của một pixel. Trong thực tế mỗi điểm ảnh sẽ được biểu diễn bởi 3 giá trị RGB (Red, Green, Blue) trong khoảng [0,255]. [3]

Thuật toán `K-Means` là một kỹ thuật phân cụm dữ liệu không giám sát, dựa trên việc gom cụm các pixel lại với nhau, tạo thành một cluster. Sau đó tìm ra các tâm cụm (centroid) đại diện cho các nhóm màu sắc khác nhau trong hình ảnh.

Trong đồ án này, chúng ta cần tập trung vào việc giảm số lượng màu sắc của một bức ảnh thông qua thuật toán `K-Means`. Ở đây chúng ta sẽ không biết trước các pixel được gom cụm dựa vào tiêu chí nào mà thuật toán chỉ trả về k cluster. Vì thế ta có thể chọn ngẫu nhiên k để giảm số lượng màu của ảnh thành k cluster.
Các bước cơ bản để áp dụng thuật toán :[2]

- Bước 1: Nạp ảnh từ input, biến đổi bức ảnh màu thành ma trận với mỗi hàng là 1 vector tương ứng với 3 giá trị màu.
- Bước 2: Giảm số chiều của bức ảnh, sau đó chọn số clusters và thực hiện thuật toán `K-Means` với ma trận trên.
- Bước 3: Chúng ta sẽ có centroids và labels tương ứng cho mỗi clusters, giá trị của mỗi pixel sẽ được thay thế bằng giá trị của centroids tương ứng.
- Bước 4: Đưa ảnh về kích thước ban đầu sau đó lưu ảnh và hiển thị kết quả.

2 Mô tả hàm và thuật toán

2.1 Các thư viện sử dụng

- NumPy: Tính toán ma trận
- PIL: Đọc, ghi ảnh
- matplotlib: Hiển thị ảnh

2.2 Các hàm hỗ trợ

2.2.1 Hàm Đọc ảnh `read_Img()`

`Input:` img_path là đường dẫn tới tập tin ảnh (ảnh phải nằm chung folder với file `.ipynb`).
Sử dụng phương thức `Image.Open()` của thư viện PIL để mở ảnh.
Sau đó ta sẽ biến đổi ảnh thành 1 ma trận với giá trị mỗi hàng là 1 vector màu có 3 giá trị màu RGB bằng np.array() của thư viện `Numpy`

2.2.2 Hàm Hiển thị ảnh

Input: img_2d là ảnh ban đầu khi đã dùng hàm `read_Img()`
Sử dụng `plt.imshow()` và `plt.show()` để hiển thị ảnh gốc ban đầu

2.2.3 Hàm Lưu ảnh

Input: img_2d và img_path là ảnh và đường dẫn để lưu ảnh
Sử dụng `Image.fromarray()` của thư viện PIL để chuyển mảng img_2d thành một đối tượng ảnh và lưu vào đường dẫn tương ứng (với 2 định dạng là png và pdf)

2.2.4 Hàm Chuyển đổi ảnh từ kích thước 2D sang 1D

`Input:` img_2d là ảnh ban đầu khi đã dùng hàm `read_Img()`
Sử dụng `shape()` để lấy ra 3 giá trị tương ứng của mỗi hàng (heigh, width, channel). Và dùng `reshape()` để giảm số chiều của ma trận xuống thành (height * width, channel)

2.2.5 Hàm khởi tạo các centroids

`Input:`

- Ma trận img_1d chứa các vector màu.
- k_clusters là số lượng clusters muốn phân cụm.
- init_centroids là phương thức khởi tạo centroids.
  - 'random': chọn ra màu ngẫu nhiên từ 3 số trong khoảng [0, 255]
  - 'in_pixel': chọn ra màu ngẫu nhiên trong các vector màu từ ma trận img_1d
- Nếu phương thức khởi tạo là `random` thì ta sẽ dùng **np.random.randint** để chọn ra các giá trị ngẫu nhiên từ 0 đến 255 cho mảng centroids với kích thước là (k_cluster, channel). Để tránh trùng lặp trong lúc khởi tạo giá trị, ta có thể sử dụng `np.unique` để kiểm tra xem các centroids có bị trùng hay không, nếu có ta sẽ khởi tạo lại giá trị.
- Nếu phương thức khởi tạo là `in_pixel` thì ta sẽ dùng **np.random.choice** để chọn các chỉ số ngẫu nhiên từ các điểm ảnh có sẵn mà không có sự trùng lặp (replace=False).

`Output:` Centroids là các mảng centroids sau khi đã khởi tạo

2.2.6 K-Means

`Input:`

- Ma trận img_1d chứa các vector màu.
- k_clusters là số lượng clusters muốn phân cụm.
- max_inter là số lần thực hiện tính toán với centroids.
- init_centroids là phương thức khởi tạo centroids.
  - 'random': chọn ra màu ngẫu nhiên từ 3 số trong khoảng [0, 255]
  - 'in_pixel': chọn ra màu ngẫu nhiên trong các vector màu từ ma trận img_1d

`Output:`

- Centroids là kết quả cuối cùng của các mảng centroids sau khi đã thực hiện tính toán
- Labels chứa các nhãn để gán màu cho các pixel

Thực hiện thuật toán K-Means [3]

1. Khởi tạo k_cluster bất kỳ cho centroid bằng `init_centroids`
2. Lặp cho đến khi max_iter = 0 thì trả về centroids và labels
3. Tính distance = khoảng cách của các vector màu đến centroids
4. Gán nhãn labels cho các vector màu
5. cập nhật centroids bằng new_centroids = trung bình cộng của các điểm ảnh trong cùng 1 cluster
6. Nếu centroids = new_centroids thì dừng thuật toán và trả về centroids và labels
7. max_inter = max_inter - 1 và quay lại bước 2.

2.2.7 Hàm Tạo ảnh mới từ các centroids

[4] `Input:`

- img_2d_shape Một tuple chứa kích thước của ảnh gốc với định dạng (height, width, 3), với 3 là số lượng kênh màu RGB
- centroids là một mảng có kích thước (k_clusters, num_channels) lưu trữ các centroid của các cụm màu.
- labels Một mảng có kích thước (height * width) lưu trữ nhãn của từng điểm ảnh để xác định cụm màu mà điểm ảnh đó thuộc về.

Hàm thực hiện việc duyệt qua từng centroids và gán màu của centroids[i] cho các điểm ảnh có nhãn tương ứng labels = i trong mảng new_img

`Output:` new_img là ảnh mới từ các centroids và các nhãn màu [1]

3 Kết quả thực hiện

Hình 1: Hình ví dụ (1200*676 - updated 17/6/2024)

Kết quả thực hiện với hình 1

Bảng 1:

k_cluster	init_centroid	result
3	'random'	

Continued on next page

Bảng 1: (Continued)

5	'random'	
7	'random'	
3	'in_pixels'	
5	'in_pixels'	

Continued on next page

Bảng 1: (Continued)

7	'in_pixels'	

Hình 2: Hình ví dụ (1126 x 2000 - updated 17/06/2024)

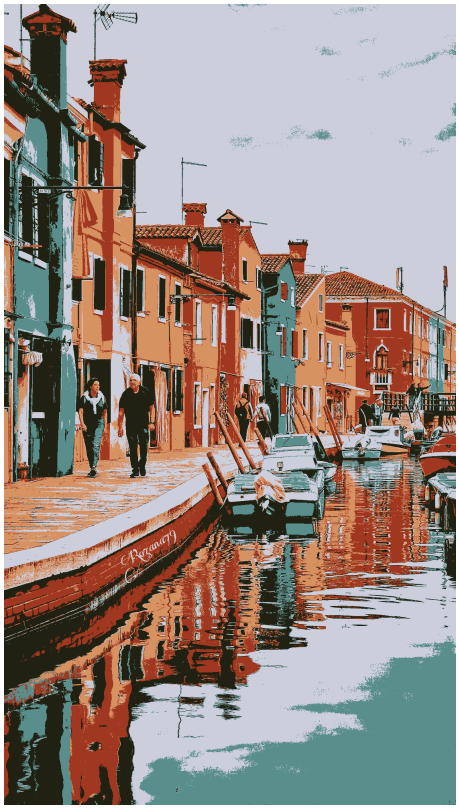
Kết quả thực hiện với Hình 2

Bảng 2:

k_cluster	init_centroid	result
3	'random'	

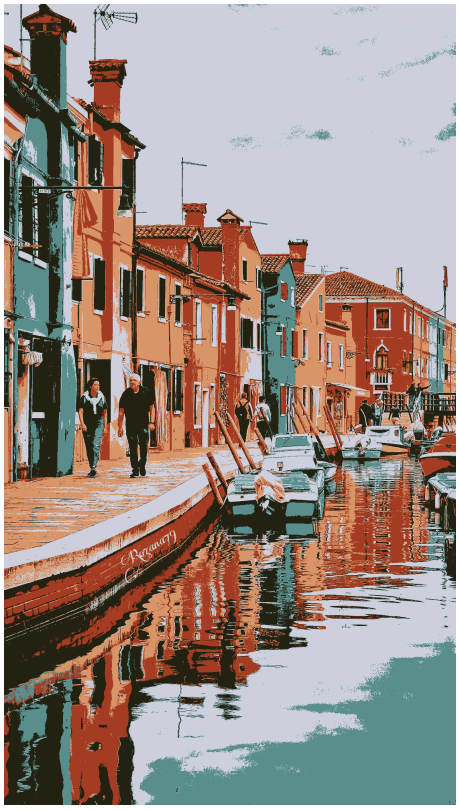
Continued on next page

Bảng 2: (Continued)

5	'random'	
7	'random'	

Continued on next page

Bảng 2: (Continued)

3	'in_pixels'	
5	'in_pixels'	

Continued on next page

Bảng 2: (Continued)

7	'in_pixels'	

4 Nhận xét

4.1 Nhận xét tổng quát

Về tổng thể, các kết quả sau 2 lần thử thể hiện đúng với yêu cầu. Nếu so kết quả với thuật toán `K-means` của thư viện **Scikit-learn** thì kết quả ra tương đối giống nhau.

Đối với các ảnh có kích thước nhỏ và ít màu sắc như hình 1 thường sẽ cho ra kết quả không tốt nếu số max_iter thấp. Khi ta cho số lượng cluster càng lớn thì kết quả xuất ra sẽ được chất lượng hơn đồng thời tốn nhiều tài nguyên và bộ nhớ để xử lý hơn.

4.2 Hạn chế

Do chương trình chỉ sử dụng đơn thuần thư viện Numpy nên việc tính toán trên ma trận có kích thước lớn lâu hơn so với việc sử dụng các thư viện khác (vd như Scipy) vì thế nên với các bức ảnh 4K kích thước lớn thì chương trình sẽ không chạy được.

Việc xác định trước K_cluster gây ra khá nhiều sai sót ở nhiều trường hợp. Trên thực tế, với nhiều bức ảnh khác nhau, ta không thể xác định được giá trị k này cho phù hợp nhất cho bức ảnh đó.

Kết quả tốt hay không sẽ phụ thuộc vào việc chọn ngẫu nhiên các centroids ban đầu, vì khi đó ta sẽ chọn ra được các màu hợp lý để tiếp tục cập nhật. Ngược lại, khi các centroid có giá trị gần với nhau thì kết quả xuất ra sẽ không được tốt.

Tài liệu

[1] Numpy document. 2008.

[2] Vũ Hữu Tiệp. K-means clustering. https://machinelearningcoban.com/2017/01/01/kmeans/, 2018.

[3] Phan Thị Phương uyên. Đồ án 1 - color compression.

[4] Nguyễn Đinh Quang Khánh. Project 01. 2022.