

ĐẠI HỌC QUỐC GIA TP HCM

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BỘ MÔN CÔNG NGHỆ TRI THỨC



Báo cáo Đồ án Thực hành

Đề tài: Color Compression

Môn học: Toán Ứng dụng và Thống kê cho Công nghệ Thông tin

Sinh viên thực hiện:

Trà, Hoàng Anh (21127453)

Giáo viên hướng dẫn:

ThS. Vũ Quốc Hoàng

ThS. Nguyễn Văn Quang Huy

CN. Ngô Đình Hy

ThS. Phan Thị Phương Uyên

Ngày 16 tháng 7 năm 2023

LỜI MỞ ĐẦU

Color compression là một kỹ thuật giảm số lượng màu sắc trong một hình ảnh, giúp tiết kiệm dung lượng lưu trữ và truyền tải. Color compression có nhiều ứng dụng trong đồ họa máy tính, xử lý ảnh, truyền thông và nén dữ liệu.

Trong báo cáo này, em xin trình bày sơ lược về thuật toán K-Means, một phương pháp color compression phổ biến, hiệu quả và cách ứng dụng thuật toán K-Means vào bài toán Color compression. Thuật toán K-Means là một kỹ thuật phân cụm dữ liệu không giám sát, dựa trên việc tìm ra các tâm cụm (centroid) đại diện cho các nhóm màu sắc khác nhau trong hình ảnh. Mục tiêu của thuật toán K-Means là tối thiểu hóa tổng bình phương khoảng cách giữa các điểm dữ liệu và tâm cụm tương ứng của chúng. Đồng thời, báo cáo này cũng đánh giá ảnh hưởng của số lượng cụm (k) và phương thức khởi tạo cụm đến chất lượng hình ảnh.

Do thời gian và trình độ chuyên môn còn hạn chế nên đồ án này không thể tránh khỏi những thiếu sót. Em rất mong nhận được nhiều ý kiến đóng góp từ các thầy, cô giáo và bạn bè để đồ án ngày càng hoàn thiện hơn!

Sinh viên thực hiện:
Trà, Hoàng Anh (21127453)

Mục lục

1	Ý tưởng thực hiện	3
2	Mô tả hàm và thuật toán	3
2.1	Các thư viện sử dụng	3
2.2	Các hàm hỗ trợ	4
2.2.1	Hàm đọc ảnh <code>readImg()</code>	4
2.2.2	Hàm biến đổi <code>reshape_img()</code>	4
2.2.3	Hàm kiểm tra điều kiện dừng <code>cmp()</code>	4
2.2.4	Hàm vận hành và xuất kết quả <code>showResult()</code>	4
2.3	Thuật toán K-means	4
3	Kết quả thực hiện	5
4	Nhận xét	8
4.1	Nhận xét chung	8
4.2	Hạn chế nói chung	8
	Tài liệu tham khảo	9

1 Ý tưởng thực hiện



Hình 1: Hình ví dụ 450x250 (updated 16/7/2023)

Cho bức ảnh như trong Hình 1. Đây là một bức ảnh màu, mỗi điểm ảnh sẽ được biểu diễn bởi ba giá trị tương ứng với màu Red, Green, và Blue. Mỗi giá trị này cũng là một số tự nhiên không vượt quá 255. Nếu xem mỗi pixel là một điểm dữ liệu mô tả bởi một vector màu ba chiều chứa các giá trị này, sau đó áp dụng thuật toán K-Means, ta cũng có thể phân các pixel ảnh thành các cluster và có được kết quả như mong muốn.

Như vậy, ta có thể đưa ra ý tưởng thực hiện [3] như sau:

1. Nạp ảnh, biến đổi bức ảnh thành 1 ma trận mà mỗi hàng là 1 vector màu với 3 giá trị màu.
2. Chọn số clusters và thực hiện thuật toán K-Means với ma trận trên.
3. Sau khi tìm được các clusters và centroid cho mỗi cluster, giá trị của mỗi pixel được thay bằng giá trị của centroid tương ứng.
4. Đưa ảnh về kích thước ban đầu và xuất ra kết quả cuối cùng.

2 Mô tả hàm và thuật toán

2.1 Các thư viện sử dụng

- PIL: Đọc và ghi ảnh.
- NumPy: Tính toán ma trận.

- `matplotlib`: Hiển thị hình ảnh.

2.2 Các hàm hỗ trợ

2.2.1 Hàm đọc ảnh `readImg()`

Input: `img_path` là đường dẫn tới tập tin ảnh

Sử dụng phương thức `Image.open()` trong thư viện PIL để mở ảnh.

2.2.2 Hàm biến đổi `reshape_img()`

Input: `raw_img` là ảnh sau khi mở bằng hàm `readImg()`

Biến đổi bức ảnh thành 1 ma trận mà mỗi hàng là 1 vector màu với 3 giá trị màu bằng `shape()` và `reshape()`

2.2.3 Hàm kiểm tra điều kiện dừng `cmp()`

Input: `minus_centroid` là hiệu số giữa centroids cũ và centroids mới

Output: True hoặc False.

Nếu giá trị tuyệt đối của các hiệu số thành phần trong `minus_centroid` không lớn hơn 1 thì trả về True.

Ngược lại, trả về False.

2.2.4 Hàm vận hành và xuất kết quả `showResult()`

Thực hiện các thao tác để xuất ra kết quả cuối cùng:

- Biến đổi bức ảnh thành 1 ma trận mà mỗi hàng là 1 vector màu với 3 giá trị màu bằng hàm `reshape_img()`.
- Thực hiện thuật toán K-Means với ma trận trên.
- Thay đổi giá trị của các pixel bằng giá trị của các centroid tương ứng.
- Đưa kết quả về kích thước ban đầu bằng `reshape`.
- Xuất ảnh gốc và ảnh kết quả[4].

2.3 Thuật toán K-means

Input:

- Ma trận `img_1d` chứa các vector màu.
- `k_clusters` là số lượng clusters.
- `max_inter` là số lần thực hiện tính toán với centroids.
- `init_centroids` là phương thức khởi tạo centroids.
 - 'random': chọn màu ngẫu nhiên bằng cách random 3 số từ `[0, 255]`.

- 'in_pixels': chọn màu từ các vector màu trong ma trận img_1d.

Output:


- centroids là mảng các centroids cuối cùng sau khi thực hiện thuật toán.
- labels chứa các nhãn để gán màu của các pixels.






Thực hiện thuật toán[2][3]:

1. centroids = Khởi tạo k_clusters centroid cho img_1d bằng phương pháp init_centroids
2. Nếu max_iter = 0 thì dừng thuật toán và trả về centroids, labels
3. distance = Khoảng cách từ các vector màu tới các centroids[1]
4. Gán nhãn cho các vector màu vào labels[1]
5. new_centroids = k_clusters centroid mới, mỗi centroid mới là trung bình cộng của tất cả các điểm ảnh cùng cluster[1]
6. Nếu centroids ã new_centroids thì dừng thuật toán và trả về centroids, labels
7. centroids = new_centroids
8. max_iter = max_iter - 1 và quay lại bước 2.

3 Kết quả thực hiện

Kết quả thực hiện với hình 1

k_clusters	init_centroids	Kết quả
3	random	





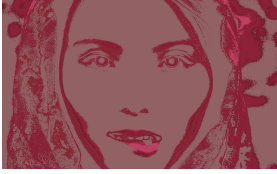
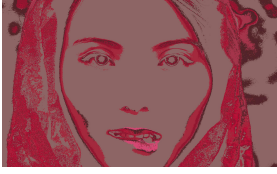
k_clusters	init_centroids	Kết quả
5	random	
7	random	
3	in_pixels	
5	in_pixels	
7	in_pixels	

k_clusters	init_centroids	Kết quả
------------	----------------	---------



Hình 2: Hình ví dụ 2048x1289 (updated 16/7/2023)

Kết quả thực hiện với hình 2

k_clusters	init_centroids	Kết quả
3	random	
5	random	
7	random	
3	in_pixels	
5	in_pixels	
7	in_pixels	

4 Nhận xét

4.1 Nhận xét chung

Nhìn chung, kết quả thu được sau 2 lần thử khá tốt, nếu so với thuật toán Kmeans của `scikit-learning` thì ta có hiệu quả xấp xỉ nhau.

Tuy nhiên, hiệu suất nén ảnh đối với ảnh có kích thước nhỏ (được nén - giảm màu bằng phần mềm khác) như Hình 1 hoặc ảnh đơn sắc thường cho kết quả không tốt với `max_iter` thấp. Rõ ràng, số lượng `k_clusters` càng nhiều thì chất lượng kết quả càng tốt.

Ngoài ra, nếu chúng ta vẽ đồ thị quá trình vận hành của Kmeans, khi chúng ta khởi tạo 'random' thì tốc độ cho ra kết quả sẽ nhanh hơn và kết quả cũng khả thi hơn so với khởi tạo 'in_pixels' trong một số trường hợp.

Trường hợp ảnh có kích thước lớn và nhiều màu thì lấy ngẫu nhiên sẽ cho hiệu quả tốt hơn theo Bảng 2.

4.2 Hạn chế nói chung

- Số lượng `k_clusters` phải được xác định trước. Việc xác định trước 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 này cho phù hợp nhất đối với bức ảnh đó.
- Kết quả cuối cùng phụ thuộc rất nhiều vào bước khởi tạo các centroids ban đầu. Bởi nếu centroids ban đầu được khởi tạo hợp lý thì sẽ cho ra kết quả rất tốt. Tuy nhiên, nếu các centroids có giá trị như nhau thì khi trải qua thuật toán sẽ cho ra một kết quả có "1 màu" duy nhất.

Từ 2 Bảng 1 và 2 cũng thấy, khi khởi tạo bằng "in_pixels" thì kết quả cuối cùng cũng gặp nhiều vấn đề hơn.

Tài liệu

- [1] NumPy Developers. Numpy reference. 2008.
- [2] Vũ Hữu Tiếp. Machine learning cơ bản. *machinelearningcoban.com*, III(10):110–126, 2018.
- [3] ThS. Phan Thị Phương Uyên. Hướng dẫn thực hiện Đồ án 1: Color compression. 2023.
- [4] Đặng Ngọc Tiến. Project 1: Color compression. 2022.