ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN 1: COLOR COMPRESSION

Môn học: Toán ứng dụng và thống kê

Lớp: 22CLC03

Sinh viên thực hiện:

Hoàng Bảo Khanh (22127183)

Giáo viên hướng dẫn:

Ths. Phan Thị Phương Uyên

Ths. Nguyễn Văn Quang Huy

Ngày 19/06/2024

MỤC LỤC:

1. Giới thiệu đồ án:

Một bức ảnh có thể lưu trữ dưới ma trận các điểm ảnh. Có nhiều loại ảnh được sử dụng trong thực tế như ảnh xám, ảnh màu…

Tuy nhiên, số lượng màu lưu trữ trong ảnh khá lớn, khi lưu ảnh sẽ tốn chi phí lưu trữ. Do đó bài toán đặt ra là giảm số lượng màu để biểu diễn ảnh sao cho nội ảnh được bảo toàn nhất có thể.

Để thực hiện nội dung yêu cầu trên, em cài đặt chương trình giảm số lượng màu cho ảnh bằng cách sử dụng thuật toán K-Means.

2. Ý tưởng thực hiện

a. Đọc file và chuyển đổi ma trận

Để thực hiện được việc giảm số lượng màu trong ảnh, em cài đặt hàm để đọc file ảnh. Sau đó từ file ảnh, chương trình sẽ chuyển đổi sang ma trận hai chiều với các kích thước bao gồm:

- Độ rộng: là số pixels có được nằm trên cùng 1 hàng ngang
- Độ cao: là số pixels có được nằm trên cùng 1 hàng dọc

Trong đó, mỗi điểm ảnh sẽ bao gồm 3 giá trị tương ứng với 3 màu RGB.

Sau đó, để có thể thực hiện thuật toán K-Means, chương trình sẽ chuyển đổi ma trận 2 chiều sang ma trận 1 chiều và số lượng kênh màu sẽ là 3 (đó là các kênh màu RGB).

b. Thuật toán K-Means

Trước khi chạy thuật toán K-Means, trước tiên chương trình phải tạo các centroids, là k giá trị pixels được chọn làm giá trị trung tâm với k là số lượng màu của ảnh dùng cho thuật toán (*k_clusters*). Để tạo centroids, chương trình sẽ có 3 sự lựa chọn chính:

- Chế độ ‘random’: nếu lựa chọn tạo centroids ngẫu nhiên, chương trình sẽ tạo ngẫu nhiên các giá trị màu từ 0 cho đến 255 với kích thước ứng với k màu đã chọn trước đó *(k_clusters*).
- Chế độ ‘in_pixels’: chương trình sẽ chọn là điểm ảnh ngẫu nhiên trong ảnh để làm centroids.

Sau khi lựa chọn centroids, chương trình sẽ tính khoảng cách từ các điểm ảnh đến centroids, và giá trị labels sẽ lưu các vị trí của các giá trị centroid mà điểm ảnh có khoảng cách gần nhất.

Sau đó, từng giá trị centroid sẽ được cập nhật lại bằng tính trung vị trên dòng của các điểm ảnh thuộc nhóm centroid tương ứng.

Để kết thúc vòng lặp sớm, chương trình sẽ kiểm tra tính hội tụ giữa centroids cũ và mới. Nếu không có nhiều sự khác nhau giữa hai centroids trên thì thuật toán K-Means sẽ dừng.

c. Biểu diễn ảnh và lưu ảnh

Trước khi biểu diễn ảnh, chương trình sẽ chuyển đổi mảng một chiều sang mảng hai chiều. Chương trình sẽ lưu ảnh sau khi chuyển đổi từ ma trận hai chiều sang hình ảnh và sẽ lưu hai file ảnh dưới dạng file pdf và file png.

3. Mô tả các hàm

Trong chương trình sẽ bao gồm các hàm chính sau:

a. *Hàm read_img*:

Hàm được sử dụng với mục đích đọc file ảnh thành ma trận hai chiều. Để thực hiện hàm, chương trình sử dụng hàm ***open()*** của thư viện Image và hàm sẽ trả về ma trận hai chiều sau khi đọc.

b. *Hàm show_img*:

Hàm sử dụng để biểu diễn ảnh qua hàm ***imshow()*** của thư viện matplotlib.pyplot

c. *Hàm save_img*:

Trước tiên, hàm sẽ thực hiện chuyển đổi ma trận hai chiều thành hình ảnh qua hàm ***fromarray()*** của thư viện Image. Sau đó, chương trình sẽ lưu ảnh dưới 2 dạng file pdf và file png.

d. *Hàm convert_img_to_1d*:

Hàm chuyển đổi ma trận hai chiều thành mảng một chiều để có thể sử dụng cho thuật toán K-Means. Khi đó mảng một chiều sẽ có kích thước bằng độ rộng nhân với độ cao của ma trận ban đầu.

e. *Hàm kmeans*:

Trước tiên, hàm sẽ thực hiện việc tạo centroids. Việc khởi tạo centroids sẽ do hàm initialize_centroids đảm nhiệm với các lựa chọn thiết lập centroids tương ứng (random hoặc in_pixels).

Sau đó, hàm sẽ chạy trong vòng lặp với số lần đã được xác định sau khi người dùng nhập biến max_iters (số vòng lặp tối đa có thể chạy). Bên trong vòng lặp, hàm sẽ khởi tạo labels để lưu các chỉ số của các giá trị centroid mà các điểm ảnh có khoảng cách nhỏ nhất đến centroid đã chọn.

Ngoài ra, chương trình còn kiểm tra xem liệu còn các centroid nào không được sử dụng trong labels hay không. Nếu có, hàm sẽ khởi tạo giá trị centroids và cho vòng lặp chạy lại từ đầu. Còn nếu không, chương trình sẽ khởi tạo centroids mới với các giá trị toàn 0 và thực hiện gán các giá trị trung vị của các điểm ảnh vào các centroid tương ứng.

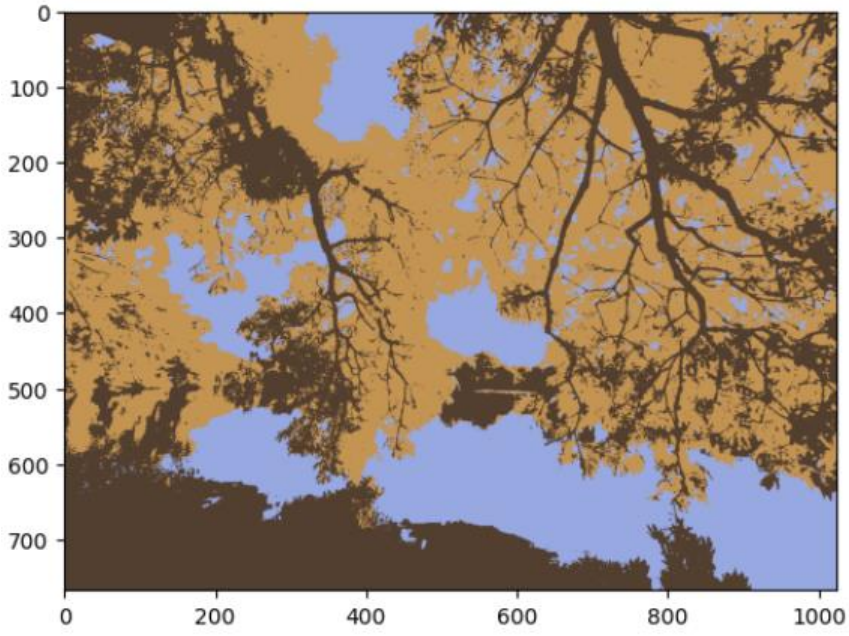
Chương trình còn kiểm tra tính hội tụ của centroids cũ và centroids vừa tạo. Nếu giá trị của hai mảng centroids trên gần bằng nhau với sai số đã cho thì thuật toán sẽ dừng và trả về giá trị centroids và labels cuối cùng.

4. Kết quả thực nghiệm

Tại mục này chúng ta sẽ thử nghiệm xem chương trình sẽ chạy như thế nào với số lượng màu khác nhau. Do đó, chương trình cho phép người dùng nhập số lượng màu sử dụng cho thuật toán và gán chúng vào biến k_clusters. Tại lần thử nghiệm này, số vòng lặp tối đa được chọn đều là 100 và centroids được chọn ở chế độ 'random'. Hình ảnh gốc mà em chọn thử nghiệm sẽ là hình sau:

Hình 1: Ảnh gốc được chọn làm thử nghiệm (nguồn: Google)

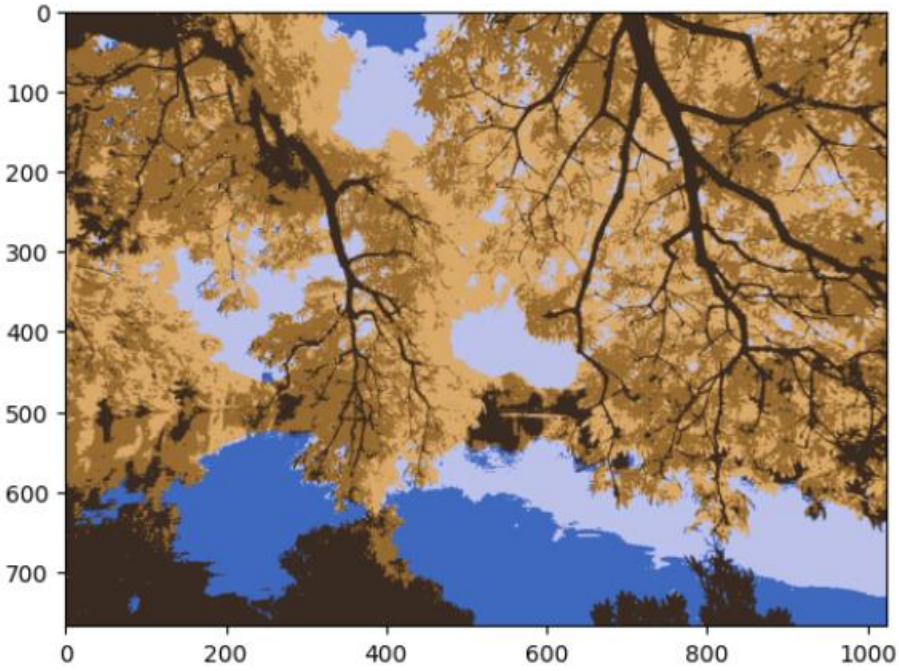
- Trường hợp số lượng màu là 3 (k_clusters = 3):



Hình 2: Kết quả sau khi chạy với số lượng màu là 3

Từ kết quả, ta thấy bức hình không rõ ràng so với ảnh gốc, với ba màu chủ yếu là nâu, xanh biển và vàng. Thời gian trung bình khi chạy cả chương trình là 18.37 giây.

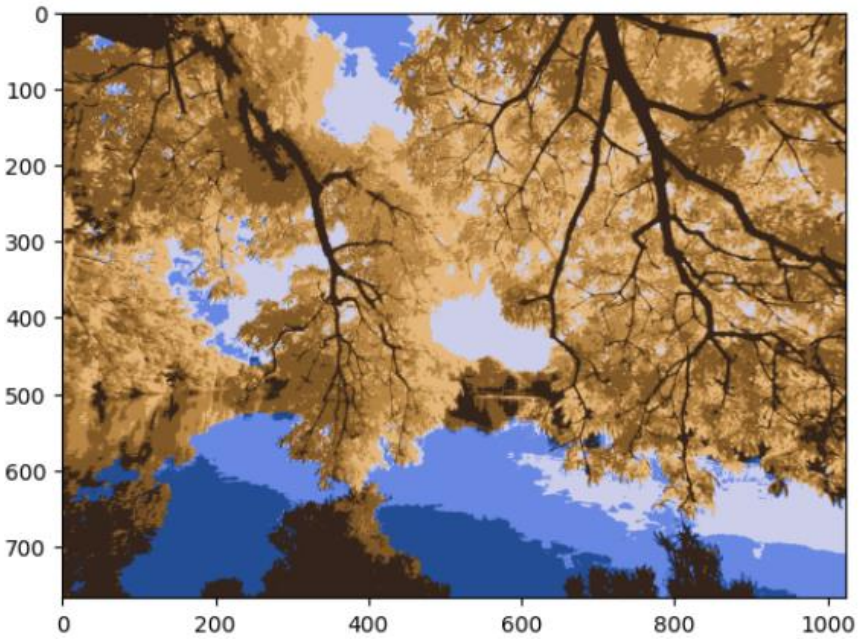
- Trường hợp số lượng màu là 5 (k_clusters = 5)



Hình 3: Kết quả sau khi chạy với số lượng màu là 5

Từ kết quả, ta thấy hình ảnh trên rõ ràng hơn so với hình ở trường hợp đầu. Thời gian chạy trung bình ở trường hợp này là 25.47 giây.

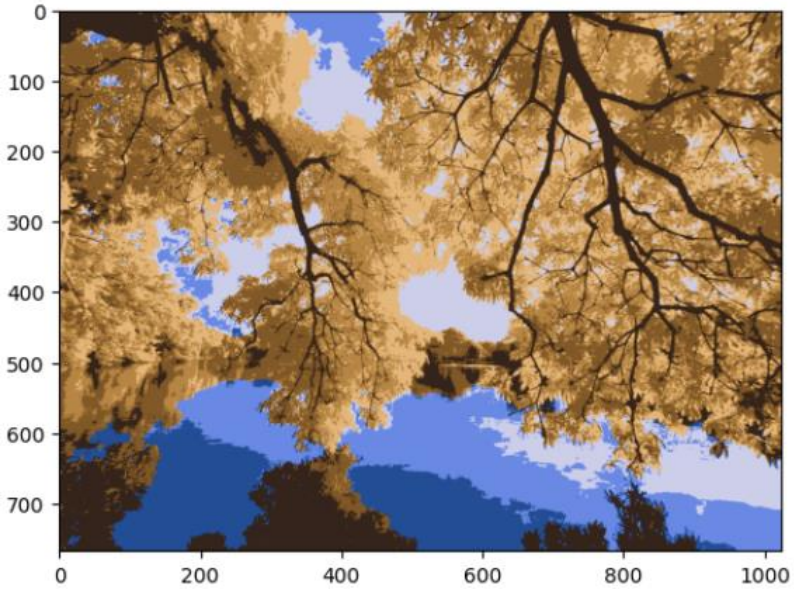
- Trường hợp số lượng màu là 7 (k_clusters = 7)



Hình 4: Kết quả sau khi chạy với số lượng màu là 7

Với trường hợp trên, hình ảnh đã được biểu diễn rõ nét và cụ thể hơn nhiều so với hai trường hợp đầu. Thời gian chạy trung bình là 35.3 giây.

- Đối với phương pháp ‘in_pixels’:



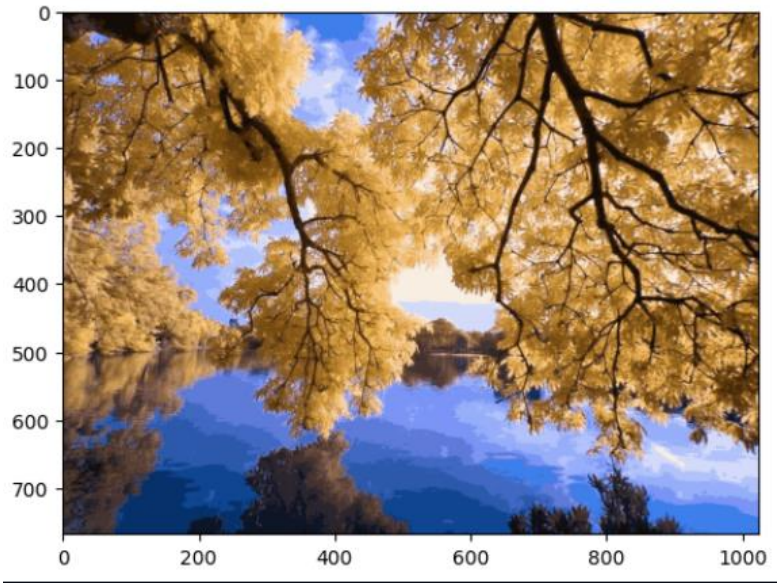
Hình 5: Kết quả với số lượng màu là 7 khi sử dụng phương pháp in_pixels

Từ kết quả trên ta có thể thấy được không có nhiều sự khác nhau giữa hai phương pháp random và in_pixels. Thời gian chạy ở trường hợp trên là 39.1 giây.

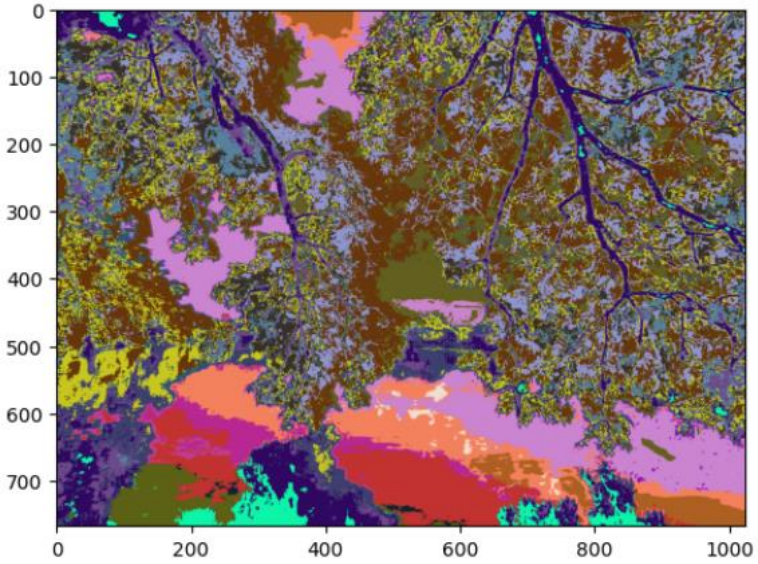
5. Nhận xét:

Từ các trường hợp trên, ta rút ra được các nhận xét như sau:

- Với số lượng màu càng nhiều thì thời gian chạy chương trình càng lâu. Tuy nhiên chất lượng hình ảnh sẽ rõ ràng hơn.
- Đối với cả hai phương pháp chọn centroids là random và in_pixels, kết quả hình ảnh đưa ra ở cả hai trường hợp khá là tương đồng và sự khác nhau không đáng kể.
- Khi thử nghiệm thêm trường hợp số lượng màu là 50 với cùng tối đa 100 vòng lặp, thì phương pháp in_pixels cho ra kết quả chính xác hơn rất nhiều so với phương pháp random. Điều đó cho thấy việc khởi tạo centroids bằng phương pháp in_pixels là chính xác hơn so với phương pháp random.



Hình 6: Kết quả của phương pháp in_pixels với số lượng màu là 50



Hình 7: Kết quả của phương pháp random với số lượng màu là 50

6. Tài liệu tham khảo

[1] K-means clustering, *https://www.youtube.com/watch?v=4b5d3muPQmA* , truy cập vào ngày 15/06/2024

[2] Bài 4: K-means clustering, Vu, *https://machinelearningcoban.com/2017/01/01/kmeans/*, truy cập ngày 15/06/2024.

[3] numpy.allclose, *https://numpy.org/doc/stable/reference/generated/numpy.allclose.html* , truy cập ngày 17/06/2024.

[4] Pulkit Sharma, Introduction to K-Means Clustering Algorithm, *https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/*, 24/04/2024.

[5] numpy.where, *https://numpy.org/doc/stable/reference/generated/numpy.where.html* , truy cập ngày 18/06/2024

[6] Imshow in Python, *https://plotly.com/python/imshow/* , truy cập ngày 18/06/2024