

ĐẠI HỌC QUỐC GIA  
THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN  
KHOA CÔNG NGHỆ THÔNG TIN



MTH00057 - Toán ứng dụng và thống kê cho Công  
nghệ thông tin

BÁO CÁO ĐỒ ÁN 1  
K-Means Compression Color

Họ tên  
Nguyễn Lê Hồ Anh Khoa

MSSV  
23127211

Giảng viên hướng dẫn  
Nguyễn Văn Quang Huy  
Trần Hà Sơn  
Nguyễn Đình Thúc  
Nguyễn Ngọc Toàn

Ngày 21 tháng 6 năm 2025

# Mục lục

<b>1</b>	<b>Thông tin sinh viên</b>	<b>3</b>
<b>2</b>	<b>Dánh giá</b>	<b>3</b>
2.1	Bảng tự đánh giá các yêu cầu đã hoàn thành . . . . .	3
2.2	Dánh giá tổng thể mức độ hoàn thành của bài nộp . . . . .	3
<b>3</b>	<b>Ý tưởng thực hiện</b>	<b>3</b>
3.1	Thuật toán K-Means clustering . . . . .	3
3.2	Mục tiêu của K-Means . . . . .	3
3.3	Ý tưởng cơ bản . . . . .	4
3.4	Mô tả thuật toán . . . . .	4
3.5	Ứng dụng thuật toán K-Means clustering trong giảm màu ảnh. . . . .	6
<b>4</b>	<b>Mô tả các hàm</b>	<b>7</b>
4.1	Các thư viện cần thiết . . . . .	7
4.2	Các hàm xử lý ảnh cơ bản . . . . .	7
4.2.1	( <code>read_img(img_path)</code> ) . . . . .	7
4.2.2	( <code>show_img(img_2d)</code> ) . . . . .	7
4.2.3	( <code>save_img(img_2d, img_path, export_type)</code> ) . . . . .	7
4.2.4	( <code>convert_img_to_1d(img_2d)</code> ) . . . . .	8
4.2.5	( <code>generate_2d_img(img_2d_shape, centroids, labels)</code> ) . . . . .	8
4.2.6	( <code>has_converged(old_centroids, new_centroids, tolerance=0.001)</code> )	8
<b>5</b>	<b>Kết quả</b>	<b>8</b>
5.1	Kết quả với <code>init_method = in_pixels</code> . . . . .	9
5.1.1	Kết quả với giá trị $K = 3$ . . . . .	9
5.1.2	Kết quả với giá trị $K = 5$ . . . . .	9
5.1.3	Kết quả với giá trị $K = 7$ . . . . .	9
5.1.4	Kết quả với giá trị $K = 9$ . . . . .	10
5.2	Kết quả với <code>init_method = random</code> . . . . .	10
5.2.1	Kết quả với giá trị $K = 3$ . . . . .	10
5.2.2	Kết quả với giá trị $K = 5$ . . . . .	10
5.2.3	Kết quả với giá trị $K = 7$ . . . . .	11
5.2.4	Kết quả với giá trị $K = 9$ . . . . .	11
5.3	Thời gian thực hiện . . . . .	11
<b>6</b>	<b>Nhận xét</b>	<b>13</b>
6.1	Chất lượng ảnh đầu ra . . . . .	13
6.2	Thời gian chạy của thuật toán . . . . .	13

6.3	Kết luận . . . . .	14
6.4	Mục tiêu của bài toán nén màu ảnh bằng K-Means . . . . .	14

# 1 Thông tin sinh viên

Họ và tên: Nguyễn Lê Hồ Anh Khoa. MSSV: 23127211. Lớp: 23CLC09

## 2 Đánh giá

### 2.1 Bảng tự đánh giá các yêu cầu đã hoàn thành

Bảng 1: Bảng tự đánh giá đồ án

STT	Yêu cầu	Mức độ hoàn thành
1	Đọc ảnh.	100%
2	Hiển thị ảnh.	100%
3	Lưu ảnh.	100%
4	Chuyển đổi ảnh từ kích thước 2D (height, width, channels) sang 1D (height × width, channels)	100%
5	Gom nhóm màu sử dụng K-Means.	100%
6	Tạo ảnh mới từ các màu trung tâm (từ K-Means).	100%
7	Cho phép nhập vào tên tập tin ảnh mỗi lần chương trình thực thi.	100%
8	Cho phép lưu ảnh với tối thiểu 2 định dạng là pdf và png.	100%
	<b>Tổng cộng</b>	<b>100%</b>

### 2.2 Đánh giá tổng thể mức độ hoàn thành của bài nộp

Bài nộp đã hoàn thành đầy đủ các yêu cầu đề ra trong bài tập. Tất cả các yêu cầu đều đã được cài đặt và kiểm thử thành công. Tổng thể, bài nộp đã hoàn thành 100% các yêu cầu đề ra.

## 3 Ý tưởng thực hiện

### 3.1 Thuật toán K-Means clustering.

### 3.2 Mục tiêu của K-Means

K-means clustering là một trong những thuật toán cơ bản nhất trong học không giám sát (Unsupervised learning). Thuật toán này dùng để phân dữ liệu đầu vào thành các cụm (cluster) khác nhau sao cho dữ liệu trong cùng một cụm có tính chất giống nhau [1]

### 3.3 Ý tưởng cơ bản

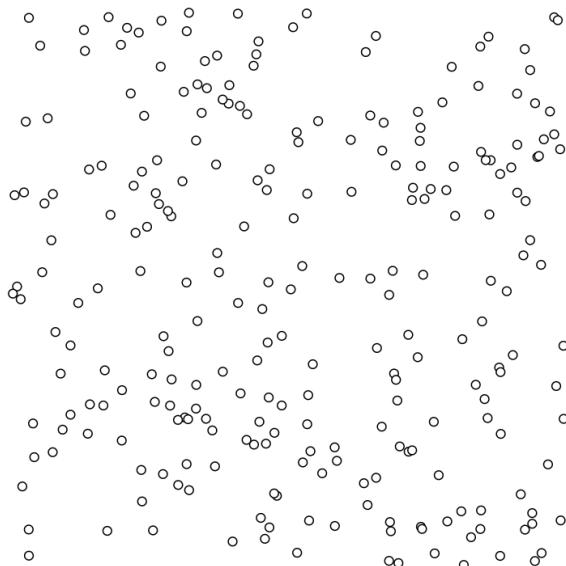
Thuật toán gồm 2 bước chính được lặp đi lặp lại cho đến khi đạt được kết quả tối ưu:

1. Gán mỗi điểm dữ liệu vào cụm gần nhất với nó.
2. Tính toán lại tâm của các cụm dựa trên các điểm dữ liệu đã được gán.
3. Lặp lại bước 1 và 2 cho đến khi không có sự thay đổi nào trong việc gán cụm hoặc đạt đến một số lần lặp tối đa.

### 3.4 Mô tả thuật toán

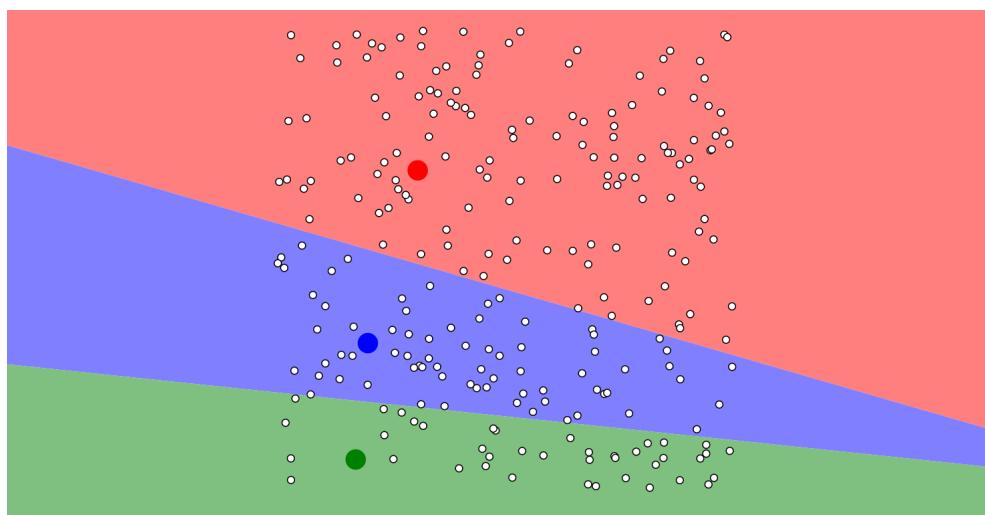
Ta có thể mô tả thuật toán K-Means clustering qua các bước sau:

1. Ta có tập dữ liệu ban đầu như sau:

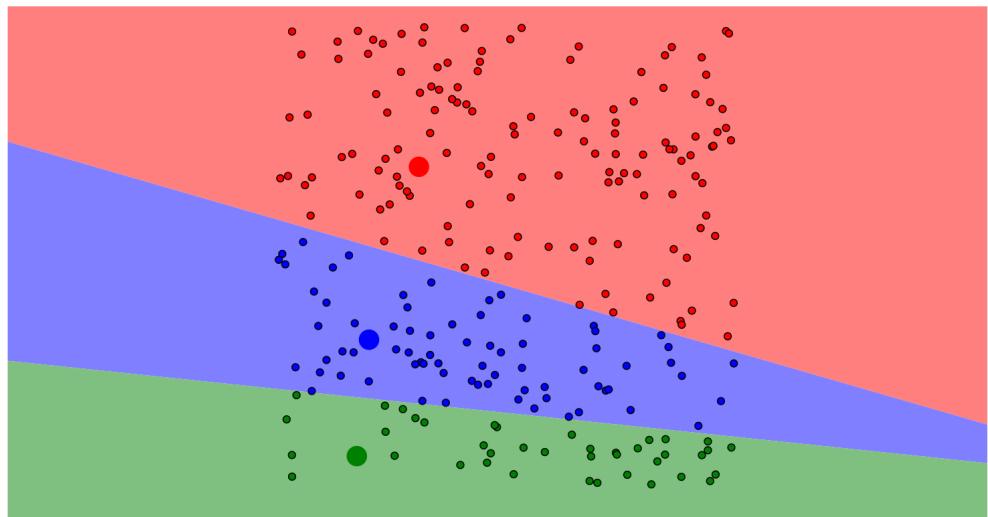


2. Chọn ngẫu nhiên  $K$  điểm làm tâm của các cụm phân loại dữ liệu vào từng nhóm.

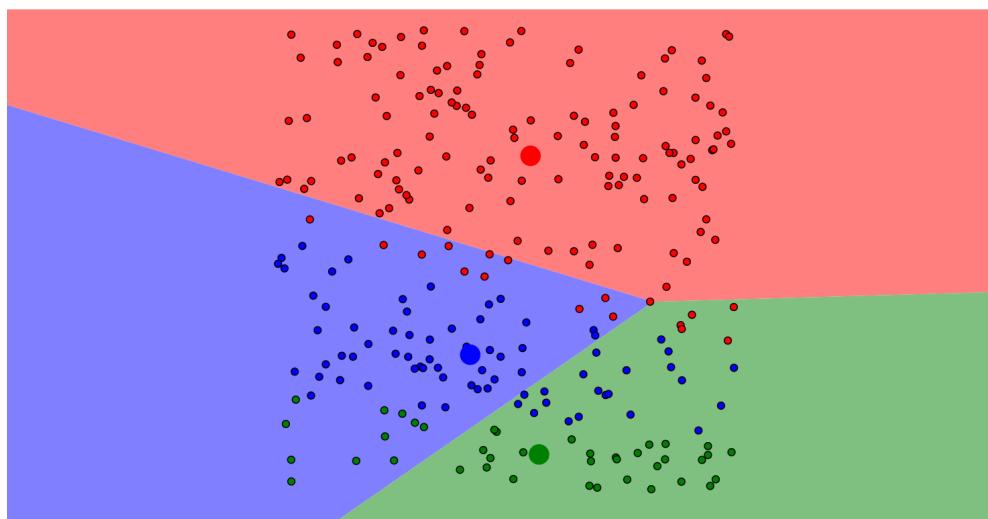
Giả sử ta chọn  $K = 3$ :



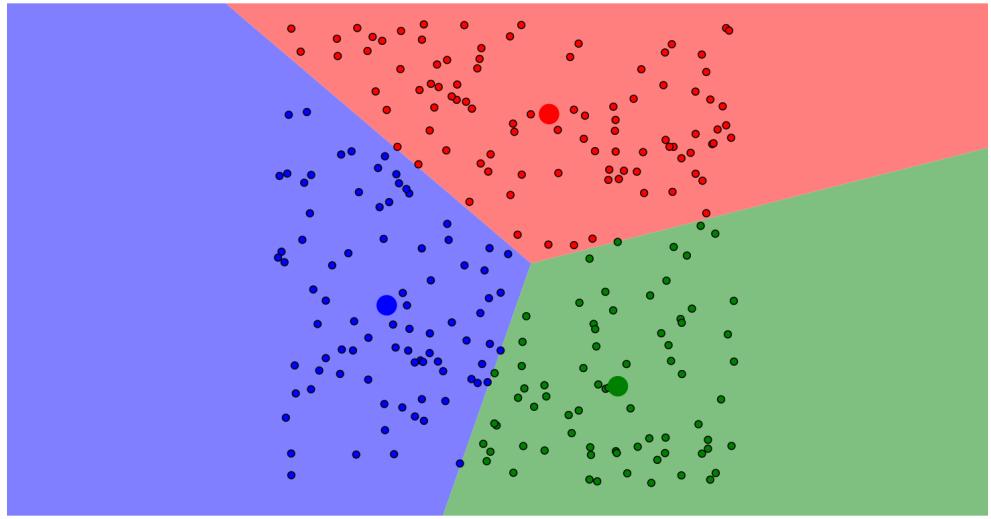
3. Gán mỗi điểm dữ liệu vào cụm gần nhất với nó. Khoảng cách giữa các điểm dữ liệu và tâm cụm được tính bằng khoảng cách Euclid:



4. Tính toán lại tâm của các cụm dựa trên các điểm dữ liệu đã được gán. Tâm cụm mới sẽ là trung bình của tất cả các điểm dữ liệu trong cụm đó:



5. Lặp lại bước 3 và 4 cho đến khi không có sự thay đổi nào trong việc gán cụm hoặc đạt đến một số lần lặp tối đa. Sau một vài lần lặp, ta có thể thu được kết quả cuối cùng như sau:



6. Kết quả cuối cùng là các cụm dữ liệu được phân loại rõ ràng, với mỗi điểm dữ liệu thuộc về một cụm duy nhất

### 3.5 Ứng dụng thuật toán K-Means clustering trong giảm màu ảnh.

Dựa vào ý tưởng cơ bản của thuật toán K-Means clustering bên trên, ta có thể áp dụng nó để giảm số lượng màu sắc có trong ảnh mà vẫn giữ được các đặc trưng của ảnh ban đầu.

1. Đầu tiên, ta sẽ đọc ảnh và chuyển đổi nó từ định dạng 2D (height, width, channels) sang định dạng 1D (height  $\times$  width, channels) để dễ dàng xử lý.
2. Tiếp theo, chọn số lượng màu sắc  $K$  mà ta muốn giữ lại trong ảnh.
3. Tạo ra  $K$  điểm centroids từ hình ảnh. Có 2 phương pháp để tạo ra các điểm này:
  - **random**: Chọn ngẫu nhiên  $K$  điểm centroids trong đoạn từ [0, 255] cho mỗi kênh màu (R, G, B).
  - **in\_pixels**: Chọn ngẫu nhiên  $K$  điểm centroids từ các màu sắc có trong ảnh.
4. Sử dụng thuật toán K-Means clustering để phân loại các điểm dữ liệu (màu sắc) vào các cụm dựa trên khoảng cách Euclid đến các điểm centroids.
5. Tính toán lại các điểm centroids dựa trên các điểm dữ liệu đã được phân loại.
6. Lặp lại bước 4 và 5 cho đến khi không có sự thay đổi nào trong việc gán cụm hoặc đạt đến một số lần lặp tối đa.
7. Sau khi thuật toán hội tụ, ta sẽ có các điểm centroids đại diện cho các màu sắc chính trong ảnh.
8. Cuối cùng, ta sẽ tạo ra một ảnh mới bằng cách thay thế mỗi điểm dữ liệu (màu sắc) trong ảnh gốc bằng màu sắc tương ứng với centroid gần nhất. Ảnh này sẽ có kích thước và bối cảnh giống như ảnh gốc nhưng chỉ sử dụng  $K$  màu sắc đã chọn.

- Kết quả là một ảnh đã được giảm màu sắc, với số lượng màu sắc giảm xuống còn  $K$  nhưng vẫn giữ được các đặc trưng chính của ảnh ban đầu. [2]

## 4 Mô tả các hàm

### 4.1 Các thư viện cần thiết

Trong đồ án này, các thư viện chính được sử dụng bao gồm:

- numpy**: Thư viện xử lý ma trận và mảng số học hiệu năng cao, cung cấp các phép tính vector hóa giúp tăng tốc độ tính toán trên dữ liệu lớn.
- PIL (Python Imaging Library)**: Thư viện xử lý ảnh, được sử dụng để đọc/ghi các định dạng ảnh phổ biến và thực hiện các thao tác cơ bản với ảnh.
- matplotlib**: Thư viện vẽ đồ thị, được dùng để hiển thị ảnh và biểu đồ đánh giá hiệu năng.
- Các thư viện hỗ trợ:
  - os**: Xử lý đường dẫn và thao tác file
  - time**: Đo thời gian thực thi

### 4.2 Các hàm xử lý ảnh cơ bản

#### 4.2.1 (`read_img(img_path)`)

Hàm `read_img` có nhiệm vụ đọc hình ảnh từ đường dẫn được truyền vào, sau đó chuyển đổi thành một mảng NumPy 2 chiều đại diện cho ảnh RGB. Điều này cho phép máy tính có thể xử lý hình ảnh dễ dàng hơn.

#### 4.2.2 (`show_img(img_2d)`)

Hàm `show_img` có chức năng hiển thị ảnh từ mảng numpy 2D sử dụng thư viện `matplotlib`. Để giúp hình ảnh hiển thị rõ ràng và không bị che bởi các trục tọa độ, lệnh `plt.axis('off')` được sử dụng để tắt hiển thị trục.

#### 4.2.3 (`save_img(img_2d, img_path, export_type)`)

Hàm `save_img` được dùng để lưu một hình ảnh từ mảng numpy 2D vào một đường dẫn cụ thể. Quá trình hoạt động như sau:

- Mảng ảnh 2D được chuyển thành ảnh sử dụng hàm `Image.fromarray()` của thư viện PIL.
- Tên tệp được ghép từ `img_path` và đuôi mở rộng `export_type`.
- Ảnh được lưu bằng hàm `save()`, hỗ trợ các định dạng như png, jpg, jpeg, pdf.

#### 4.2.4 (convert\_img\_to\_1d(img\_2d))

Hàm `convert_img_to_1d` chuyển đổi ảnh từ dạng 2D (height, width, channels) sang dạng 1D ( $height \times width, channels$ ) để phù hợp với việc xử lý của thuật toán K-Means. Quá trình chuyển đổi được thực hiện bằng hàm `reshape()`.

#### 4.2.5 (generate\_2d\_img(img\_2d\_shape, centroids, labels))

Sau khi phân cụm màu bằng K-means, ta cần chuyển kết quả thành ảnh để hiển thị và lưu trữ. Hàm `generate_2d_img` sẽ tái tạo lại ảnh 2D từ mảng 1D đã phân cụm. Kết quả nhận được sẽ là một ảnh 2D với kích thước ban đầu, trong đó mỗi pixel được gán màu tương ứng với tâm cụm gần nhất.

#### 4.2.6 (has\_converged(old\_centroids, new\_centroids, tolerance=0.001))

Hàm `has_converged` kiểm tra xem các centroid có thay đổi đáng kể sau mỗi vòng lặp hay không. Nếu độ thay đổi tương đối giữa các centroid mới và cũ nhỏ hơn ngưỡng `tolerance`, thuật toán sẽ kết thúc. Chi tiết như sau:

- Bước đầu tiên là tính toán độ chênh lệch tuyệt đối giữa các centroid cũ và mới bằng hàm `np.abs(new_centroids - old_centroids)`.
- Sau đó, các thay đổi này được chuẩn hóa để biết mức độ thay đổi tương đối so với giá trị ban đầu bằng cách chia cho giá trị tuyệt đối của các centroid cũ.
- Nếu sự thay đổi lớn nhất nhỏ hơn một ngưỡng nhất định (`tolerance`), ta xem như thuật toán đã hội tụ và có thể dừng lại.

Điều này giúp giảm thiểu số lần lặp không cần thiết, tiết kiệm thời gian tính toán. Hàm này được gọi trong vòng lặp chính của thuật toán K-means để kiểm tra điều kiện dừng. Việc so sánh độ chênh lệch của từng centroid thay vì tính tổng khoảng cách giúp tăng tốc độ tính toán, vì ta chỉ cần so sánh từng giá trị riêng lẻ thay vì tính tổng toàn bộ.

## 5 Kết quả

Tất cả các kết quả dưới đây đều được thực hiện với `max_iter = 100` với ảnh gốc có kích thước 1920px x 1080px với 236703 màu.

## 5.1 Kết quả với `init_method = in_pixels`

### 5.1.1 Kết quả với giá trị $K = 3$



(a) Ảnh gốc



(b) Kết quả

Hình 1: Kết quả với `init_method = in_pixels` và  $K = 3$

### 5.1.2 Kết quả với giá trị $K = 5$



(a) Ảnh gốc



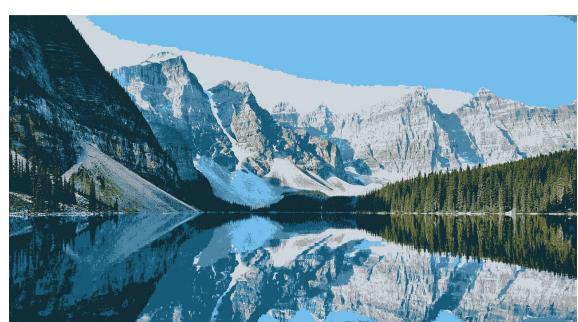
(b) Kết quả

Hình 2: Kết quả với `init_method = in_pixels` và  $K = 5$

### 5.1.3 Kết quả với giá trị $K = 7$



(a) Ảnh gốc



(b) Kết quả

Hình 3: Kết quả với `init_method = in_pixels` và  $K = 7$

#### 5.1.4 Kết quả với giá trị $K = 9$



(a) Ảnh gốc



(b) Kết quả

Hình 4: Kết quả với `init_method = in_pixels` và  $K = 9$

#### 5.2 Kết quả với `init_method = random`

##### 5.2.1 Kết quả với giá trị $K = 3$



(a) Ảnh gốc



(b) Kết quả

Hình 5: Kết quả với `init_method = random` và  $K = 3$

##### 5.2.2 Kết quả với giá trị $K = 5$



(a) Ảnh gốc



(b) Kết quả

Hình 6: Kết quả với `init_method = random` và  $K = 5$

### 5.2.3 Kết quả với giá trị $K = 7$



(a) Ảnh gốc



(b) Kết quả

Hình 7: Kết quả với `init_method = random` và  $K = 7$

### 5.2.4 Kết quả với giá trị $K = 9$



(a) Ảnh gốc



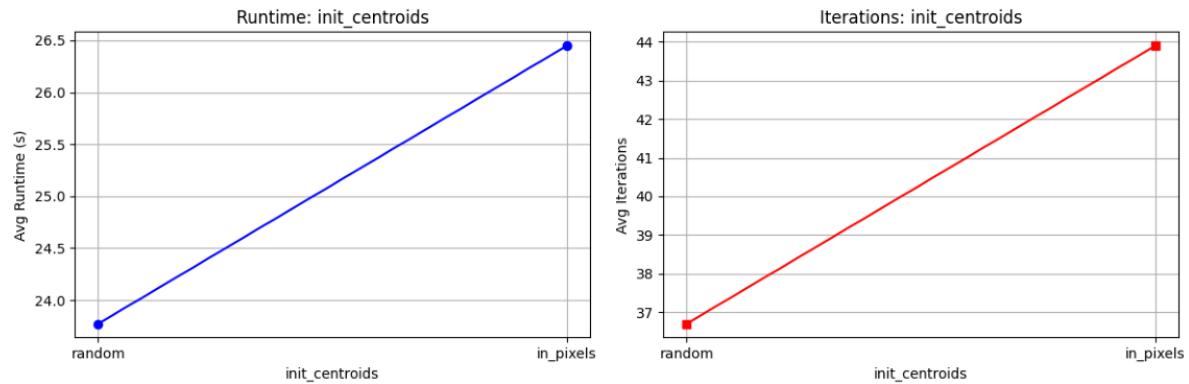
(b) Kết quả

Hình 8: Kết quả với `init_method = random` và  $K = 9$

## 5.3 Thời gian thực hiện

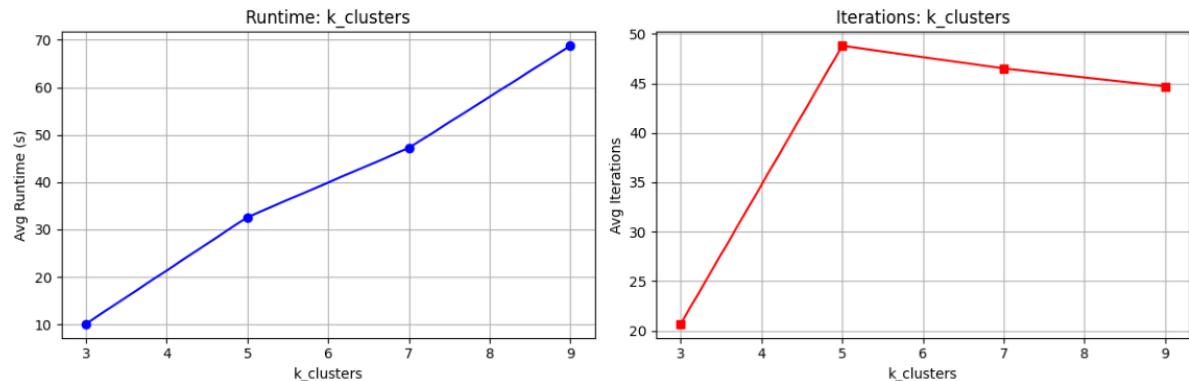
Để đánh giá thời gian thực hiện của thuật toán, ta sẽ đo thời gian thực hiện thuật toán với các phương pháp khởi tạo centroid khác nhau và các giá trị  $K$  khác nhau. Các kết quả được trình bày trong các hình bên dưới.

Performance Charts for init\_centroids



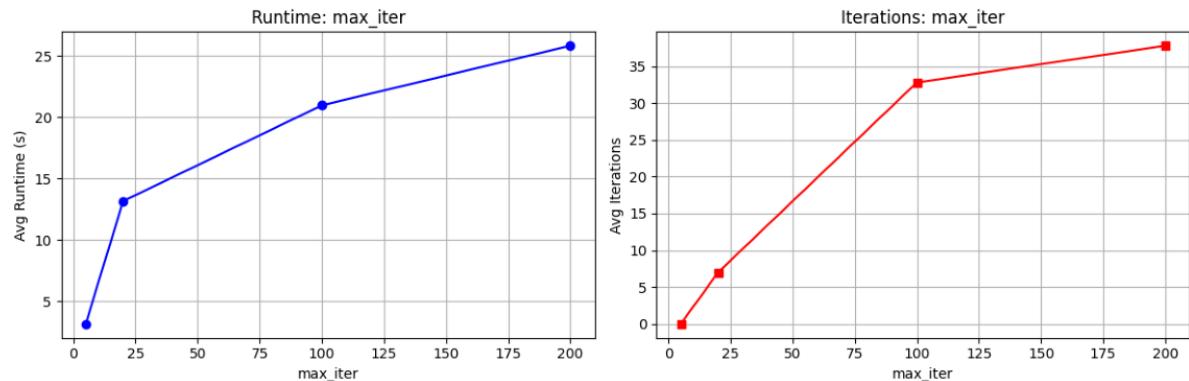
Hình 9: Thời gian thực hiện thuật toán với các phương pháp khởi tạo điểm centroid khác nhau với  $K = 5$  và  $\text{max\_iter} = 100$

Performance Charts for k\_clusters



Hình 10: Thời gian thực hiện thuật toán với các giá trị  $K$  khác nhau với  $\text{init\_method} = \text{in\_pixels}$  và  $\text{max\_iter} = 100$

Performance Charts for max\_iter



Hình 11: Thời gian thực hiện thuật toán với các giá trị  $\text{max\_iter}$  khác nhau với  $\text{init\_method} = \text{in\_pixels}$  và  $K = 5$

## 6 Nhận xét

### 6.1 Chất lượng ảnh đầu ra

Chất lượng ảnh của thuật toán K-Means phụ thuộc vào nhiều yếu tố như số lượng cụm  $K$ , phương pháp khởi tạo centroid, và số vòng lặp tối đa. Qua quá trình thử nghiệm (Có thể xem nhiều kết quả hơn tại [Link](#)), có thể rút ra một số nhận xét như sau:

- **Số lượng cụm  $K$ :** Khi  $K$  tăng, ảnh đầu ra giữ được nhiều chi tiết hơn. Ngược lại, nếu  $K$  quá nhỏ, ảnh sẽ bị mất nhiều chi tiết quan trọng. Qua thử nghiệm. Kết quả cho thấy với  $K = 3$ , ảnh đầu ra có vẻ mờ và mất nhiều chi tiết những vẫn giữ được bố cục hình ảnh. Khi tăng  $K$  lên 5, 7, và 9, ảnh trở nên sắc nét hơn và giữ được nhiều màu sắc gốc.
- **Phương pháp khởi tạo centroid:** Việc chọn phương pháp khởi tạo ảnh hưởng đến chất lượng ảnh đầu ra. Phương pháp `in_pixels` cho kết quả tốt hơn so với `random`, vì nó sử dụng các màu sắc có trong ảnh gốc làm điểm khởi đầu, giúp giảm thiểu hiện tượng mất màu.
- **Số vòng lặp tối đa `max_iter`:** Việc tăng số vòng lặp giúp thuật toán hội tụ tốt hơn, nhưng nếu quá cao có thể dẫn đến thời gian chạy lâu mà không cải thiện đáng kể chất lượng ảnh. Qua thử nghiệm, khi `max_iter` càng tăng từ 5 lên 100, chất lượng ảnh đầu ra càng tốt nhưng khi tăng lên 200, dường như chất lượng không thay đổi quá nhiều do thuật toán đã hội tụ.
- **Ảnh gốc:** Ảnh đầu vào có độ phân giải cao (1920x1080) và nhiều màu sắc, nên việc giảm màu sắc sẽ làm mất đi một số chi tiết. Tuy nhiên, thuật toán vẫn giữ được bố cục và các đặc trưng chính của ảnh.

### 6.2 Thời gian chạy của thuật toán

Thời gian thực thi của thuật toán K-Means phụ thuộc vào nhiều yếu tố khác nhau. Qua quá trình thử nghiệm và in ra các biểu đồ, có thể rút ra các nhận xét như sau:

- **Kích thước ảnh:** Thuật toán K-Means hoạt động trên từng điểm ảnh, nên khi ảnh đầu vào có kích thước lớn, số lượng pixel tăng lên rất nhiều. Điều này khiến thuật toán phải tính khoảng cách và cập nhật centroid nhiều lần hơn, dẫn đến thời gian xử lý kéo dài. Ở bài toán này, ảnh gốc có kích thước  $1920 \times 1080$  (tương đương khoảng 2 triệu điểm ảnh), thời gian xử lý trung bình dao động từ 10 đến 70 giây tùy thông số.
- **Phương pháp khởi tạo centroid (`init_centroids`):** Việc chọn centroid ban đầu ảnh hưởng đến cả tốc độ hội tụ và độ ổn định của kết quả. Hai phương pháp được so sánh là `random` và `in_pixels`:
  - Với `random`, thuật toán có xu hướng hội tụ nhanh hơn.
  - Với `in_pixels`, các centroid khởi tạo phân bố đa dạng hơn nhưng cần nhiều vòng lặp để điều chỉnh, do đó thời gian chạy tăng nhẹ.

Từ biểu đồ cho thấy `in_pixels` mất nhiều vòng lặp hơn và thời gian chạy lâu hơn

khoảng 2.5 giây so với random.

- **Số cụm K :** Nếu số lượng màu ban đầu trong ảnh lớn thì việc giảm số lượng về số màu sẽ phải mất nhiều thời gian hơn. Kết quả thử nghiệm cho thấy thời gian chạy tăng tuyến tính từ 10.0s ( K = 3 ) lên 69.0s ( K = 9 ).
- **Giới hạn số vòng lặp max\_iter:** Nếu tăng giá trị `max_iter`, thuật toán có cơ hội hội tụ tốt hơn. Từ biểu đồ, khi tăng `max_iter` từ 5 lên 100, thời gian chạy tăng đáng kể từ khoảng 3 giây lên hơn 20 giây. Nhưng sau đó, thời gian tăng không đáng kể, cho thấy thuật toán đã hội tụ gần như hoàn toàn. Điều này cho thấy việc chọn giá trị `max_iter` hợp lý là rất quan trọng để tránh lãng phí thời gian.
- **Cấu hình máy tính:** Tất cả thử nghiệm được thực hiện trên máy tính cá nhân sử dụng CPU Intel Core i7. Với các máy có cấu hình thấp hơn, thời gian thực thi có thể dài hơn đáng kể.

### 6.3 Kết luận

Qua quá trình thực hiện và phân tích thuật toán K-Means trong bài toán giảm màu ảnh, có thể rút ra một số điểm chính như sau:

- Thuật toán K-Means là một giải pháp hiệu quả cho bài toán giảm màu ảnh, có khả năng giảm đáng kể số lượng màu trong ảnh trong khi vẫn duy trì được các đặc trưng quan trọng của ảnh gốc.
- Hiệu quả của thuật toán phụ thuộc nhiều vào việc lựa chọn các tham số:
  - Số cụm K cần được cân nhắc dựa trên yêu cầu về chất lượng ảnh và thời gian xử lý
  - Phương pháp khởi tạo `in_pixels` cho kết quả tốt hơn dù tốn thời gian hơn `random`
  - Số vòng lặp `max_iter` khoảng 100 là đủ để đảm bảo hội tụ mà không tốn quá nhiều thời gian
- Thời gian thực thi của thuật toán tỷ lệ thuận với kích thước ảnh và số cụm K, do đó cần cân nhắc giữa chất lượng đầu ra và hiệu năng khi triển khai thực tế.
- Có thể cải thiện hiệu năng bằng cách:
  - Tối ưu hóa code thông qua vector hóa các phép tính
  - Sử dụng GPU để tăng tốc quá trình xử lý
  - Giảm kích thước ảnh đầu vào nếu không yêu cầu độ phân giải cao

### 6.4 Mục tiêu của bài toán nén màu ảnh bằng K-Means

Bài toán giảm màu ảnh bằng thuật toán K-Means Clustering là một phương pháp học không giám sát phổ biến trong xử lý ảnh và thị giác máy tính. Mục tiêu chính của bài toán bao gồm:

- **Nén ảnh (Image Compression):** Thay vì lưu ảnh dưới hàng trăm nghìn màu, ta chỉ giữ lại K màu đại diện. Điều này giúp giảm dung lượng ảnh mà vẫn giữ được nội dung thị giác chính.
- **Tối ưu hóa không gian lưu trữ:** Với ảnh đầu vào có thể chứa hàng trăm nghìn màu, việc thay thế bằng K màu giúp giảm số lượng bit cần thiết để mã hóa từng pixel.
- **Tiền xử lý cho các bài toán thị giác:** Ảnh đã được giảm màu sẽ đơn giản hóa cho các thuật toán như nhận dạng vật thể, phân đoạn ảnh hoặc học đặc trưng vì loại bỏ được nhiều màu.
- **Tăng tốc thuật toán học máy:** Trong một số mô hình học máy, đặc biệt là mạng nơ-ron xử lý ảnh, giảm số màu đầu vào giúp giảm số chiều của dữ liệu đầu vào, từ đó giảm thời gian huấn luyện và nhu cầu tài nguyên tính toán.
- **Ứng dụng trong thiết kế đồ họa:** Tạo ra các hiệu ứng ảnh như poster hóa, tranh vẽ kỹ thuật số hay ảnh nghệ thuật bằng cách giữ lại chỉ một số ít màu đặc trưng.
- **Trực quan hóa dữ liệu:** Giảm số màu giúp đơn giản hóa biểu diễn ảnh, giúp người dùng dễ dàng so sánh, phân tích sự khác biệt giữa các vùng ảnh.

Phương pháp này đặc biệt hữu ích trong môi trường có tài nguyên hạn chế như thiết bị IoT, ứng dụng di động, hệ thống truyền ảnh trực tuyến, hoặc nơi cần xử lý hàng loạt ảnh trong thời gian thực. [2]

## Tài liệu

- [1] Tiep Vu Huu, *K-means Clustering*, Jan 1, 2017, <https://machinelearningcoban.com/2017/01/01/kmeans>
- [2] GeekforGeeks, *Image compression using K-means clustering*, May 29, 2023, <https://www.geeksforgeeks.org/machine-learning-image-compression-using-k-means-clustering/>