ĐẠI HỌC QUỐC GIA

THÀNH PHỐ HỒ CHÍ MINH

ĐẠI HỌC KHOA HỌC - TỰ NHIÊN

CÔNG NGHỆ THÔNG TIN

Project 01
Color Compression

Môn học : Toán ứng dụng và thống kê cho CNTT
Lớp : 23CLC02

Sinh viên thực hiện :
Nguyễn Thu Thảo (23127119)

Giảng viên :
Vũ Quốc Hoàng
Nguyễn Văn Quang Huy
Trần Thị Thảo Nhi

Mục lục

Danh sách hình vẽ

Danh sách bảng

I Ý tưởng thực hiện :

1 Tổng quan về đồ án :

Đồ án tập trung vào áp dụng thuật toán K-Means để giải quyết color quantization trong xử lý ảnh.
Mục tiêu làm giảm số lượng màu trong một bức ảnh xuống còn k màu cho trước, trong khi vẫn giữ được các đặc trưng màu sắc chính của ảnh gốc.

2 Input, output :

Input :

- File ảnh.
- Số nguyên k : Số màu còn lại sau khi xử lý ảnh.
- Thể loại file sẽ xuất sau khi xử lý.

Output :

- File ảnh đã xử lý còn k màu tùy theo định dạng đã nhập.

3 Mục tiêu :

- Cài đặt thuật toán K-Means sử dụng thư viện NumPy.
- Tìm ra k màu sắc chủ đạo trong ảnh gốc.
- Tạo ra một ảnh mới bằng cách thay thế mỗi pixel trong ảnh gốc bằng màu chủ đạo gần nhất với nó.
- Thể hiện và lưu ảnh sau khi thay đổi giá trị k.

4 Ý tưởng giải quyết :

- Chuyển ảnh thành mảng 2D các pixel.
- Biến đổi mảng thành mảng 1D.
- Sử dụng phương pháp K-Means để xử lý mảng 1D :
  1. Tạo các centroids màu dựa trên số màu còn lại sau khi xử lý.
  2. Đưa các pixel vào các centroids đó.
  3. Thay thế mỗi pixel bằng centroid tương ứng.
- Chuyển đổi lại thành mảng 2D.
- Chuyển đổi lại thành ảnh và thể hiện/lưu ảnh từ mảng 2D trên.

II Chi tiết thực hiện :

1 Cấu trúc chương trình :

- Hàm *readImg* : đọc ảnh và truyền dữ liệu vào mảng $[[N] \times [M], [RGB]]$ numpy
- Hàm *showImg* : Hiển thị hình ảnh từ mảng numpy.
- Hàm *printImg* : Lưu hình ảnh vào file từ mảng numpy.
- Hàm *convertArr1D* : chuyển đổi mảng 2D kích thước $[N] \times [M]$ thành mảng 1D kích thước $[N \times M]$.
- Hàm *convertArr2D* : chuyển đổi mảng 1D kích thước $[N \times M]$ thành mảng 2D kích thước $[N] \times [M]$ dựa trên kích cỡ mảng ban đầu.
- Hàm *getCentroids* : Khởi tạo centroids ban đầu bằng phương pháp *K-Means++* :
- Hàm *getLabels* : Gán nhãn cho mỗi điểm dữ liệu vào cluster gần nhất.
- Hàm *getNewCentroids* : Cập nhật lại centroids mới dựa trên các điểm dữ liệu trong cluster.
- Hàm *KMeans* : Thuật toán K-Means cho mảng 1D lấy từ *convertArr1D* dựa trên các hàm *getCentroids*, *getLabels*, *getNewCentroids* đã tạo trên.

2 Mô tả các hàm cốt lõi :

- Hàm *getCentroids* :
  1. Chọn centroids đầu tiên ngẫu nhiên từ tập hợp pixel.
  2. Tính khoảng cách bình phương nhỏ nhất từ mỗi pixel đến các centroids đã được chọn.
  3. Chọn centroids mới từ các pixel còn lại, tỉ lệ chọn một pixel tỉ lệ thuận với khoảng cách đã tính. Lặp lại cho đến khi đủ k centroids. $\rightarrow$ Tăng tính nhất quán mỗi lần khởi tạo
- Hàm *getLabels* :
  1. Tính khoảng cách Euclidean từ mỗi pixel đến tất cả k centroids.
  2. Trả về "nhãn" - centroids gần nhất của mỗi pixel.
- Hàm *getNewCentroids* :
  1. Tập hợp tất cả các pixel có cùng nhãn.
  2. Tính giá trị trung bình của các pixel trong từng centroids $\rightarrow$ centroids mới.
  3. Không có pixel trong centroids : chọn một pixel ngẫu nhiên để làm centroids mới.
- Hàm *KMeans* :
  1. Chuyển ảnh thành mảng 1D.
  2. Khởi tạo centroids bằng K-Means++.
  3. Lặp lại việc gán nhãn và cập nhật centroids cho đến khi hội tụ.

III Kết quả và kết luận :

1 Kết quả :

Drive link để xem hình ảnh rõ hơn.
Ảnh 1 : Test kết quả xử lý các lần chạy với ảnh có nhiều chi tiết nhỏ.

Hình 3.1.1 : Ảnh 1 : Photo by Eugenia Remark

Kích thước ảnh : 1814×2419 - 4.4 MB
Kết quả sau xử lý :

k	**k = 3**	**k = 5**	**k = 7**
Kết quả			
Thời gian (s)	t = 18.063	t = 88.94	t = 98.142
Dung lượng	550 KB	946 KB	1.15 MB

Bảng 3.1.1 : Kết quả sau khi xử lý Ảnh 1

Ảnh 2 : Test kết quả xử lý các lần chạy với ảnh các gradient màu.

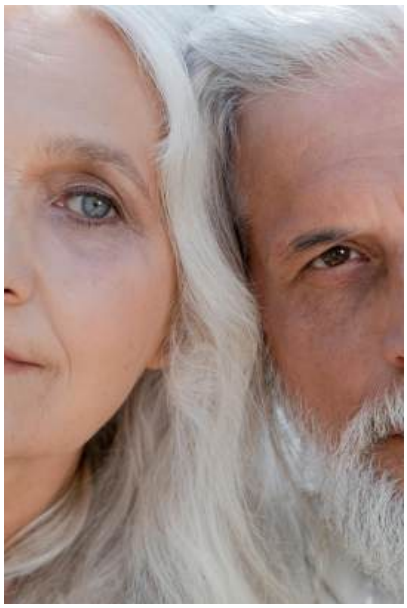
Hình 3.1.2 : Ảnh 2 : Photo by Edward Jenner

Kích thước ảnh : 2458 × 1638 - 3.5 MB
Kết quả sau xử lý :

k	**k = 3**	**k = 5**	**k = 7**
Kết quả			
Thời gian (s)	t = 9.099	t = 11.956	t = 53.651
Dung lượng	67.7 KB	73.6 KB	109 KB

Bảng 3.1.2 : Kết quả sau khi xử lý Ảnh 2

Ảnh 3 : Test kết quả xử lý các lần chạy với ảnh chân dung.



Hình 3.1.3 : Ảnh 3 : Photo by cottonbro studio

Kích thước ảnh : 1908 × 2862 - 3.45 MB
Kết quả sau xử lý :

k	**k = 3**	**k = 5**	**k = 7**
Kết quả			
Thời gian (s)	t = 27.568	t = 66.155	t = 183.422
Dung lượng	374 KB	533 KB	677 KB

Bảng 3.1.3 : Kết quả sau khi xử lý Ảnh 3

Ảnh 4 : Test kết quả xử lý các lần chạy với kích thước ảnh khác nhau.

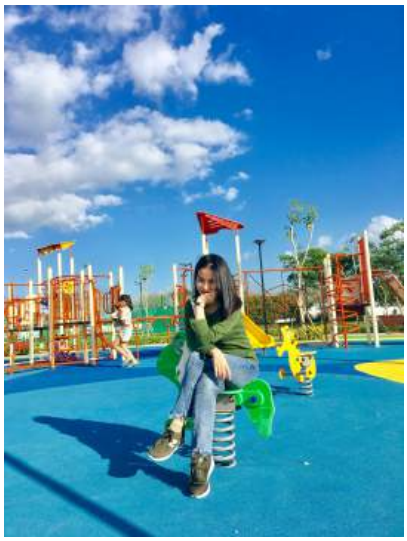
Hình 3.1.4 : Ảnh 4 : Photo by Alexander Grey

Kích thước ảnh : 2808 × 1872 - 4.36 MB
Kết quả sau xử lý :

k = 7	2808 × 1872 (100%)	1404 × 936 (50%)	297 × 618 (33%)
Kết quả			
Thời gian (s)	t = 73.652	t = 18.946	t = 6.791
Dung lượng	216 KB	78.3 KB	41.5 KB

Bảng 3.1.4 : Kết quả sau khi xử lý Ảnh 4

Ảnh 5 : Test kết quả xử lý các lần chạy khác nhau với cùng 1 k.



Hình 3.1.5 : Ảnh 5 : Photo by HY AAN

Kích thước ảnh : 1814 × 2419 - 4.43 MB
Kết quả sau xử lý :

k	**k = 7**	**k = 7**	**k = 7**
Kết quả			
Thời gian (s)	t = 98.804	t = 47.796	t = 115.799
Dung lượng	641 KB	610 KB	641 KB

Bảng 3.1.5 : Kết quả sau khi xử lý Ảnh 5

2 Kết luận :

- Thuật toán K-Means hoạt động khá hiệu quả trong việc tìm ra các màu sắc chủ đạo và thực hiện lượng tử hóa màu. Chương trình đạt được yêu cầu là giảm số lượng màu của một ảnh trong khi vẫn bảo toàn được các đặc tính màu sắc cơ bản.
- Giá trị k có ảnh hưởng trực tiếp đến chất lượng của ảnh kết quả. Khi k tăng, ảnh kết quả giữ lại được nhiều chi tiết và trông giống ảnh gốc hơn. Ngược lại, khi k giảm, ảnh bị đơn giản hóa mạnh mẽ.
- Giá trị k cũng ảnh hưởng đến thời gian chạy và dung lượng xuất ra. Khi k tăng, thời gian lâu hơn và dung lượng file xuất ra lớn. Khi k giảm, thời gian chạy nhanh hơn và dung lượng thấp hơn.
- Chưa tối ưu nhất, nhiều hình ảnh sau khi gộp nhóm bị xỉn màu.
- Đã nhất quán hơn nhưng mỗi lần chạy vẫn có khác biệt do khởi tạo bằng các pixel random.

IV Tài liệu tham khảo :

1. The Pillow development team, "Image Processing with Pillow (The Handbook)", https://pillow.readthedocs.io/en/latest/handbook/index.html, truy cập ngày 14/06/2025.
2. The Matplotlib development team, "pyplot summary", https://matplotlib.org/stable/api/pyplot_summary.html, truy cập ngày 14/06/2025.
3. Stack Overflow Community, "How do I convert a PIL Image into a NumPy array?", https://stackoverflow.com/questions/384759/how-do-i-convert-a-pil-image-into-a-numpy-array, truy cập ngày 15/06/2025.
4. GeeksforGeeks, "Python | Flatten a 2D numpy array into 1D array", https://www.geeksforgeeks.org/python/python-flatten-a-2d-numpy-array-into-1d-array/, truy cập ngày 15/06/2025.
5. Python Engineer, "K-means Clustering From Scratch In Python" [Video], https://youtu.be/lX-3nGHDhQg, truy cập ngày 15/06/2025.
6. w3resource, "NumPy: Save a NumPy array as an image file", https://www.w3resource.com/python-exercises/numpy/save-a-numpy-array-as-an-image-file-using-python.php, truy cập ngày 15/06/2025.
7. AskPython, "Extracting Filename from File Path in Python", https://www.askpython.com/python/examples/extract-filename-without-extension, truy cập ngày 22/06/2025.

V Acknowledgments :

Đồ án có sự giúp đỡ của bạn Phạm Cao Thu Hương (MSSV : 23127375) trong việc giải đáp các thắc mắc.
Ngoài ra, đồ án có sự hỗ trợ cùa deepseek trong việc tối ưu hóa và tăng tính nhất quán cũng như đảm bảo việc chỉ sử dụng thư viện numpy của hàm *getCentroids*, *getNewCentroids*, *KMeans*; và có sự hỗ trợ của GoogleAi Studio trong việc viết báo cáo phần 1. Ý tưởng thực hiện cũng như hỗ trợ dịch 1 số câu văn sang tiếng Việt ở phần 2, mục 2.2 Mô tả các hàm cốt lõi.