MỤC LỤC

I. Ý tưởng thực hiện và mô tả các hàm:

1. Ý tưởng thực hiện:

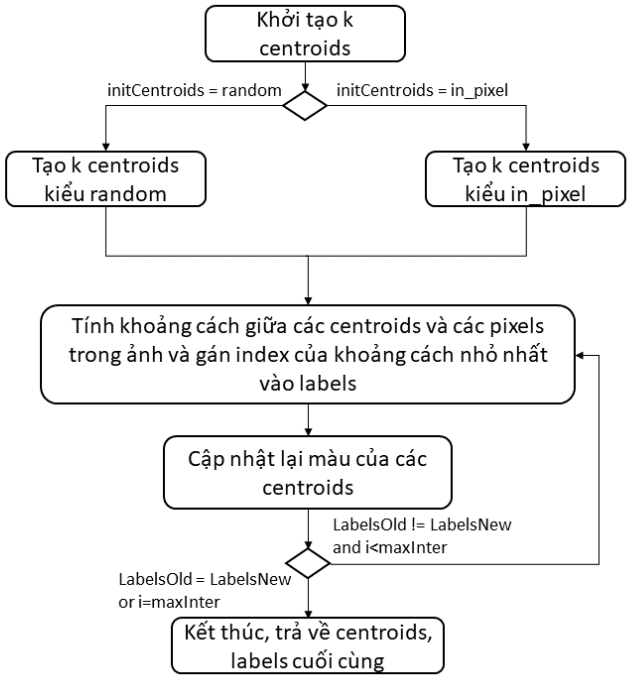
- Một bức ảnh được hiển thị dưới dạng mảng 2 chiều với mỗi phần tử là 1 vector không gian 3 chiều chứa các thông số R(Red), G(Green), B(Blue).
- Ý tưởng thực hiện thuật toán:

Bước 1: Chọn ra k centroids đầu tiên, với k là số lượng màu mà người dùng muốn ảnh được giảm xuống, k centroids đầu tiên sẽ được chọn ở 2 cách khác nhau đó là random hoặc in_pixel với random là chọn các màu ngẫu nhiên từ không gian màu RGB và in_pixel là không gian màu trong bức ảnh.

Bước 2: Tính khoảng cách giữa các pixel màu trong ảnh và các k centroids được chọn, trong bài thực hiện sử dụng thuật toán Euclidean để tính toán khoảng cách giữa các vector màu không gian 3 chiều, mục đích của việc làm này là để phân loại các màu trong ảnh ra k cụm khác nhau, các cụm sau khi đã xác định sẽ được lưu trữ trong 1 mảng 2 chiều (labels) bằng với mảng thể hiện bức ảnh, với mỗi phần tử sẽ đánh dấu cho tại pixel thứ [i,j] của bức ảnh sẽ gần với phần tử centroids nào nhất.

Bước 3: Cập nhật lại các centroids mới là trung bình cộng của các màu nằm trong một cụm centroids đã được lưu trữ ở trên (labels), cập nhật lại các labels mới, lặp lại số lần maxInter lần, nếu có 2 lần liên tiếp các labels giống nhau hoàn toàn, dừng lặp.

Bước 4: Tạo lại ảnh mới từ centroids cuối cùng và labels cuối cùng.



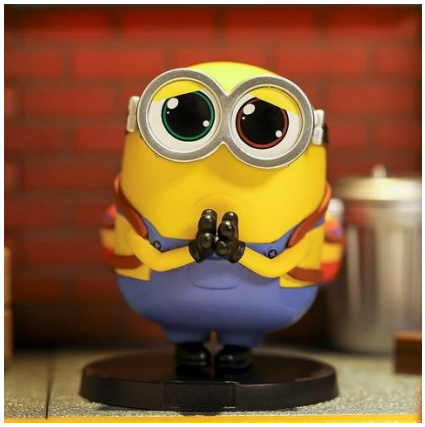
Hình I.1: Biểu đồ cho ý tưởng trên

2. Mô tả các hàm:

- Hàm **openImage()** và **imageToArray():** để mở ảnh và chuyển ảnh về dạng array để xử lý.
- Hàm **arrayCentroids():** khởi tạo màu theo kiểu "random" hoặc "in_pixel",
- Hàm **findNearestIndex():** tính khoảng cách Euclidean giữa từng pixel trong ảnh và k centroids đã tạo ra, sau đó dùng hàm np.argmin để lưu lại index của centroid gần nhất của từng điểm ảnh vào arrLabel, hàm trả về arrLabel.
- Hàm **updateCentroids():** cập nhật lại màu của từng centroids là giá trị trung bình cộng của các màu trong từng cụm centroids đã được xác định trong arrLabel (tức là các pixel có cùng index của centroid), giá trị trả về là centroids.
- Hàm **finalCentroids():** lặp lại quá trình tính khoảng cách và cập nhật centroids để có centroids và labels cuối cùng, giá trị trả về là centroids và labels.
- Hàm **colorLessImage():** tạo ảnh sau khi giảm màu từ centroids và labels.
- Hàm **kmeans():** ghép nối các hàm trên, chạy hàm **arrayCentroids()** để tạo centroids và hàm **finalCentroids()** để chạy thuật toán kmeans.
- Hàm **main():** hàm chính của chương trình, cho người dùng nhập giá trị đầu vào là tên của ảnh (ví dụ: test.png), tên ảnh sau khi giảm màu (ví dụ: testOut.png), số lượng màu mong muốn (ví dụ: 3), cách init centroids (random, in_pixel) maxInter sẽ dựa theo giá trị của kClusters và thay đổi số lần lặp bởi vì càng nhiều màu thì các quá trình tìm khoảng cách và cập nhật centroids sẽ mất thời gian, nhưng nếu càng ít màu mà số lần lặp ít thì màu sắc sẽ bị thiếu chi tiết, do đó maxInter sẽ được thay đổi: k càng lớn số lần lặp càng nhỏ và ngược lại. Sau đó sẽ in ra màn hình ảnh ban đầu người dùng nhập, thời gian chạy thuật toán kmeans và ảnh sau khi giảm màu, đồng thời cũng tạo ra một ảnh mới có tên ảnh sau giảm màu được nhập (testOut.png).

II. Hình ảnh kết quả với từng số lượng màu:

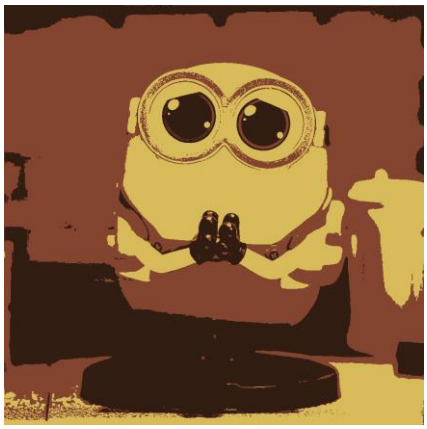
1. Hình 1: Kích thước 1000x1000 px



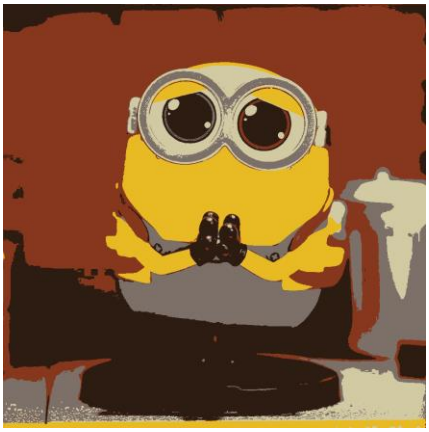
Kiểu init centroid: random

- Số lượng màu: 3

- Thời gian cho kết quả: 11.49s



- Số lượng màu: 5
- Thời gian cho kết quả: 15.36s

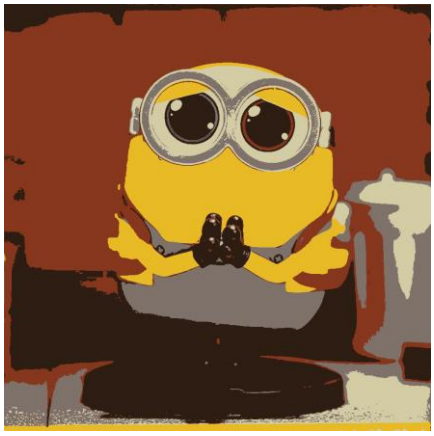


- Số lượng màu: 7
- Thời gian cho kết quả: 12.83s

Kiểu init centroid: in_pixel

- Số lượng màu: 3
- Thời gian cho kết quả: 6.43s

- Số lượng màu: 5
- Thời gian cho kết quả: 14.04s

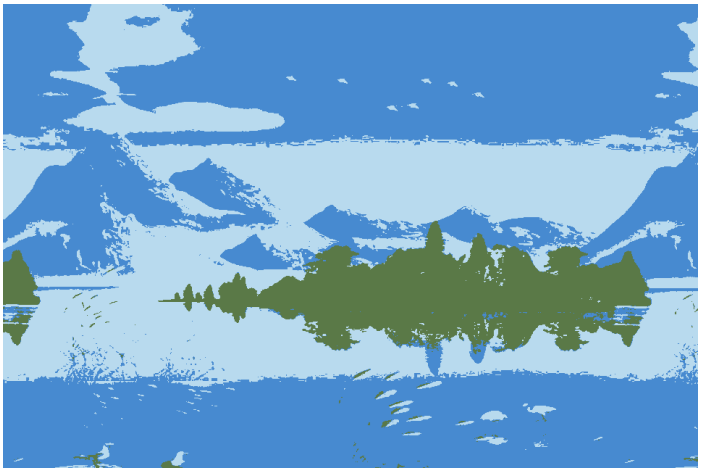


- Số lượng màu: 7
- Thời gian cho kết quả: 13.12s

2. Hình 2: Kích thước 640x960 px

Kiểu init centroid: random

- Số lượng màu: 3
- Thời gian cho kết quả: 5.51s

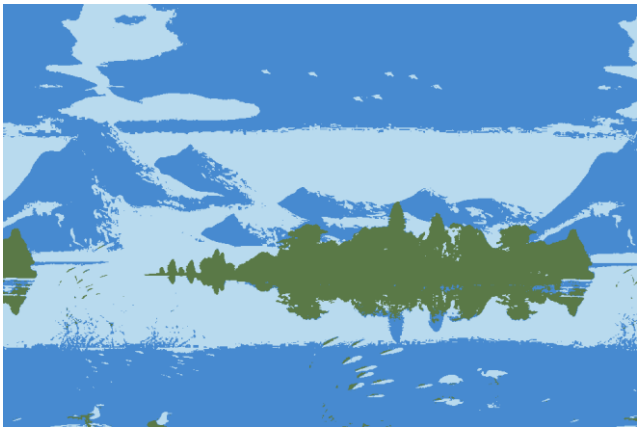


- Số lượng màu: 5
- Thời gian cho kết quả: 9.92s

- Số lượng màu: 7
- Thời gian cho kết quả: 7.77s

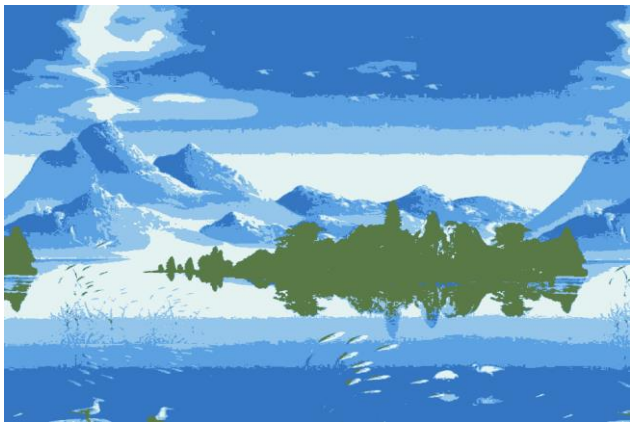
Kiểu init centroid: in_pixel

- Số lượng màu: 3
- Thời gian cho kết quả: 6.25s



- Số lượng màu: 5

- Thời gian cho kết quả: 9.48s



- Số lượng màu: 7
- Thời gian cho kết quả: 8.67s

3. Hình 3: Kích thước 1333x2000 px

Kiểu init centroid: random

- Số lượng màu: 3
- Thời gian cho kết quả: 36.42s

- Số lượng màu: 5
- Thời gian cho kết quả: 38.15s

- Số lượng màu: 7
- Thời gian cho kết quả: 37.87s

Kiểu init centroid: in_pixel

- Số lượng màu: 3
- Thời gian cho kết quả: 39.70s

- Số lượng màu: 5
- Thời gian cho kết quả: 45.08s

- Số lượng màu: 7
- Thời gian cho kết quả: 37.16s

III. Nhận xét các kết quả trên:

1. Nhận xét hình 1:

- *Về màu sắc:* với hình 1, trong ảnh có nhiều màu sắc, do đó cho dù để ở kiểu init centroids là random hay in_pixel thì ta cũng có thể dễ dàng phân cụm các màu sắc của ảnh và cho ra kết quả, bởi vì số lần lặp sẽ thay đổi dựa theo số lượng màu, cho nên màu sắc ở các lần k nhỏ cũng không bị sai lệch quá nhiều.
- *Về thời gian:* thời gian ở các lần chạy k=3, 5, 7 là gần bằng nhau bởi vì số lần lặp max inter được dựa vào k, k càng lớn lần lặp càng nhỏ, do đó thời gian cũng sẽ gần như giống nhau, kích thước hình ảnh cũng không quá lớn nên cũng không quá mất nhiều thời gian hoàn thành.

2. Nhận xét hình 2:

- *Về màu sắc:* với hình 2, trong ảnh là màu sắc thiên xanh, do đó khi để ở kiểu init là random, tùy vào lần random bất kì sẽ có trường hợp các màu sẽ chỉ gần với 1 centroids duy nhất, rõ ràng hơn tức là, giả sử random ra 3 màu: đỏ, vàng, xanh, thì các màu trong ảnh trên sẽ gần với màu xanh nhất, do đó toàn bộ ảnh sẽ bị biến thành màu xanh như ở lần chạy k=7 kiểu random ở trên. Ngược lại, khi init centroids là in_pixel, thì khi tạo ra các centroids ban đầu có thể tạo ra các centroids theo nhiều sắc độ của màu xanh, ví dụ: xanh đậm, xanh trung bình, xanh nhạt, do đó ta dễ dàng phân cụm các màu xanh trong ảnh mà tránh được trường hợp ảnh bị một màu như kiểu init centroids random.
- *Về thời gian:* thời gian ở các lần chạy k=3, 5, 7 là gần bằng nhau bởi vì số lần lặp max inter được dựa vào k, k càng lớn lần lặp càng nhỏ, do đó thời gian cũng sẽ gần như giống nhau, thời gian chạy sẽ nhanh hơn ảnh 1 do kích thước nhỏ hơn.

3. Nhận xét hình 3:

- *Về màu sắc:* với hình 3, trong ảnh có rất nhiều màu sắc, do đó khi init centroids kiểu random ta sẽ giảm thiểu tình trạng kết quả cho ra bé hơn k mong muốn.
- *Về thời gian:* thời gian ở các lần chạy k=3, 5, 7 là gần bằng nhau bởi vì số lần lặp max inter được dựa vào k, k càng lớn lần lặp càng nhỏ, tuy nhiên, do ảnh có độ phân giải lớn nên thời gian chạy khá là lâu so với hình 1 hay hình 2

4. Nhận xét chung:

- Thuật toán có thể giảm được màu ở ảnh, làm tốt cả 2 kiểu init centroids là random và in_pixel với ảnh có nhiều màu.
- Đối với ảnh có thiên hướng về một màu và toàn bộ ảnh chỉ là các sắc độ khác nhau của một màu, thuật toán hoạt động tốt ở kiểu init centroids là in_pixel.
- Về thời gian, khi chạy các k khác nhau do có sự thay đổi về max inter nên không bị thay đổi giữa các k quá nhiều.
- Thuật toán vẫn chưa thể tối ưu đối với các ảnh có kích thước lớn hơn >1000 px.

IV. Tài liệu tham khảo:

1. machinelearningcoban.com, *Bài 4: K-means Clustering (19.01.2017) tại:* https://machinelearningcoban.com/2017/01/01/kmeans/

2. blog.vietnamlab.vn, *Hiểu nhanh về thuật toán K-means Clustering - Bài toán phân cụm qua ví dụ đơn giản (26.01.2022) tại:* https://blog.vietnamlab.vn/untitled-11/

3. programiz.com, *NumPy Array Functions tại:* https://www.programiz.com/python-programming/numpy/array-functions

4. stackoverflow.com, *How do I convert a PIL Image into a NumPy array? tại:* https://stackoverflow.com/questions/384759/how-do-i-convert-a-pil-image-into-a-numpy-array

5. websitehcm.com, *numpy.reshape () trong Python (14.02.2022) tại:* https://websitehcm.com/numpy-reshape-trong-python/