

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN**



BÁO CÁO

Toán Ứng Dụng Và Thống Kê Cho Công Nghệ Thông Tin

**Lab 02- Project 1
COLOR COMPRESSION**

Giảng viên lý thuyết: Vũ Quốc Hoàng

Giảng viên hướng dẫn thực hành: Phan Thị Phương Uyên, Nguyễn Văn Quang Huy, Ngô Đình Hy

Thành phố Hồ Chí Minh - 2023

MỤC LỤC

I.	Thông tin sinh viên:.....	2
II.	Ý tưởng thực hiện :.....	2
III.	Mô tả các hàm:.....	2
	1. Hàm main.....	2
	2. Cấu trúc hàm Hàm img2_1d.....	2
	3. Cấu trúc hàm hàm k_mean.....	3
	4. Cấu trúc hàm update_pixel.....	4
	5. Cấu trúc hàm save_pic.....	4
	6. Cấu trúc hàm ran_centroid.....	5
IV.	Kết quả.....	6
V.	Nhận xét kết quả.....	9
VI.	Các nguồn tài liệu tham khảo:.....	9

I. Thông tin sinh viên:

Tên	MSSV	Lớp
TÔ KHÁNH LINH	21127638	21CLC08

II. Ý tưởng thực hiện:

Dựa vào hướng dẫn của cô Phan Thị Phương Uyên:

Để mình nhắc lại cách làm luôn:

- Bước 1: Khởi tạo k centroids với 1 trong 2 tùy chọn 'random' hoặc 'in_pixels'

- Bước 2: Với mỗi điểm ảnh (vector màu) trong ảnh ban đầu:

 Tính khoảng cách đến k centroids

 Tìm centroid mà điểm ảnh đó gần nhất (khoảng cách đến centroid đó là ngắn nhất) - Đoạn này là tìm cụm màu cho điểm ảnh

- Bước 3: Sau khi đã tìm được các cụm màu cho tất cả điểm ảnh -> Cập nhật lại k centroids mới = trung bình cộng của các điểm ảnh (vector màu) đó thuộc về. Ví dụ, centroid 1 = mean(tất cả điểm ảnh thuộc nhóm màu 1).

- Bước 4: Kiểm tra điều kiện dừng:

 Nếu thỏa 1 trong 2 điều kiện sau thì dừng:

 1. Đạt max_iter

 2. Khi sự thay đổi của k centroids so với bước lần trước là không đáng kể (hội tụ)

 Ngược lại, chạy lại Bước 2 đến Bước 4

III. Mô tả các hàm:

● Hàm main:

- Nhập file input muốn cluster.
- Mở file ảnh và, lấy chiều rộng và chiều cao.
- Ta nén sáu hình nên dựa vào k_cluster đã được cho và init_centroid, với mỗi hình ta chuyển nó sang array, sau đó gọi kmeans(), khi xong ta gọi hàm update_pixel() để định dạng lại pixel về đúng vị trí của nó. Sau đó thêm nó vào mảng outfile (hình file output)

● Hàm img2_1d: Chuyển đổi ảnh thành ảnh 1 chiều

- Input: Tên file ảnh (filename), chiều cao của file (height), chiều rộng của file (width), channels của file (channels)
- Output: File dưới dạng mảng một chiều (img)
- Hàm này dùng để chuyển hình sang mảng một chiều để ta có thể thực hiện tính toán.

- **Hàm K_mean:**

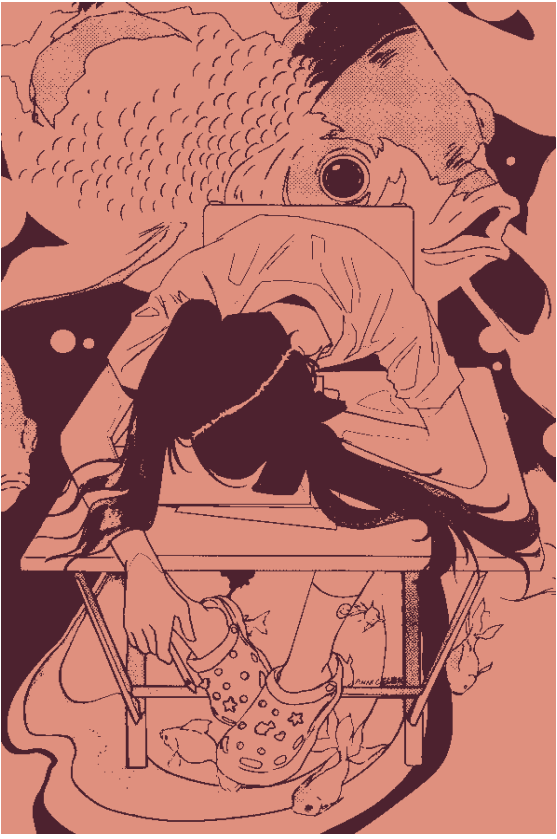

- Input: Mảng một chiều của ảnh (img_1d), Số màu (k_cluster), max_iter (Số vòng lặp), init_centroid (Cách phân bố centroid)
- Output: Mảng lưu trữ màu của centroid(centroids), Mảng lưu trữ cluster mà từng điểm ảnh thuộc về (labels)
- Gọi hàm random centroid ran_centroid()
- labels = [None for x in range(len(img_1d))]: Tạo mảng labels có cùng độ dài với img_1d, trong đó mỗi phần tử ban đầu đều được gán giá trị là None
- Ta tạo vòng lặp chạy max_iter lần:
 - + Tạo một bản sao của mảng centroids (centroidcpy)
 - + Mảng temp là một danh sách rỗng với k_clusters phần tử.
 - + Duyệt từng phần tử trong mảng img_1d
 - + distances=np.linalg.norm(centroids-img_1d[i], axis=1): Khoảng cách Euclidean từ điểm pixel đang xét đến các centroid là distance
 - + labels[i]=np.argmin(distances): Gán nhãn cho điểm dữ liệu đang xét bằng index (i) của centroid có khoảng cách gần nhất.
 - + temp[labels[i]].append(i): thêm index(i) của pixel đang xét vào temp với index là nhãn của pixel đó.
 - + for i, cluster in enumerate(temp): Duyệt qua từng cụm (cluster) và index(i) trong temp.

- + `mean = np.mean(img_1d[cluster], axis=0)` tính giá trị trung bình của các điểm dữ liệu trong cụm hiện tại.
- + `Centroids[i] = mean` cập nhật centroid của cụm hiện tại bằng giá trị trung bình (mean).
- + If `np.array_equal(centroids,centroidscopy)`: break kiểm tra xem centroids đã thay đổi hay không. Nếu nó giống centroids trước đó ta dừng vòng lặp.
- Sau khi vòng lặp kết thúc ta trả về centroids và labels
 - **Hàm `update_pixel`**: Cập nhật lại từng điểm ảnh tương ứng với nhóm mà nó thuộc về
- Input: Mảng một chiều của ảnh (`img_1d`), chiều cao của ảnh (`height`), chiều dài của ảnh (`width`), channel của ảnh (`channel`), centroids của ảnh (`centroid`), label của ảnh (`labels`)
- Output: mảng một chiều của ảnh (`img_1d`)
- Duyệt qua tất cả các điểm dữ liệu trong `img_1d`
- `img_1d[i] = centroids[labels[i]]`: gán giá trị của điểm thứ `i` trong `img_1d` bằng giá trị của centroid tương ứng với nhãn của nó trong `labels`.
- Biến mảng 1 chiều thành mảng 3 chiều.
- Trả về mảng `img_1d` đã được cập nhật giá trị pixel.
 - **Hàm `save_pic`**: Save hình dưới định dạng PNG hoặc PDF
- Input: Mảng của các hình muốn save, `init_centroid`, `k_cluster`
- Nhập định dạng (PNG hoặc PDF) nếu sai định dạng lập đến khi nhập đúng định dạng.
- Save hình và File có tên: “`initcentroid_kcluster.(png hoặc pdf)`”

- **Hàm `ran_centroid`:** Random centroid
 - Input: `k_clusters`, `img_1d`, `init_centroid`
 - Output: `centroid`
 - **Với `init_centroid` là ‘random’:**
 - + Ta liên tục random đến khi mảng centroid có chiều dài bằng `k_cluster`. Sử dụng hàm `np.random.randint(0,255,3)`, có nghĩa là random 3 phần tử từ 0 đến 255.
 - **Với `init_centroid` là ‘in_pixels’:**
 - + `pix=np.unique(img_1d, axis=0)` có nghĩa là `pix` chứa các giá trị pixel duy nhất trong mảng `img_1d`.
 - + `ind = np.random.choice(len(pix), size=k_clusters, replace=False)`: `ind` chứa `k_cluster` phần tử ngẫu nhiên của từ 0 đến kích cỡ của `pix`.
 - + `centroids = pix[ind].tolist()`: Thêm các `pix` thứ `ind` vào `centroid`

IV. Kết quả:



K	Random	In pixels
3		

5



7



V. Nhận xét kết quả:

- Hình có chiều cao: 960, chiều rộng: 596.
- Hình ảnh không còn như ban đầu mà có thay đổi và màu cũng trở nên đơn giản hơn nên ta có thể thấy thuật toán có hiệu quả. Nhìn chung thuật toán chạy khá tốt.
- Thời gian chạy thuật toán dưới 15 phút cho 6 hình.

VI. Tài liệu tham khảo:

- + <https://numpy.org/doc/stable/reference/random/generated/numpy.random.choice.html>
- + https://numpy.org/doc/stable/reference/generated/numpy.array_equal.html
- + <https://numpy.org/doc/stable/reference/random/generated/numpy.random.choice.html>
- + <https://numpy.org/doc/stable/reference/generated/numpy.unique.html>
- + <https://numpy.org/doc/stable/reference/generated/numpy.linalg.norm.html#numpy.linalg.norm>
- + <https://numpy.org/doc/stable/reference/generated/numpy.argmin.html>
- + <https://numpy.org/doc/stable/reference/generated/numpy.mean.html>
- + <https://numpy.org/doc/stable/reference/random/generated/numpy.random.choice.html>
- + <https://numpy.org/doc/stable/reference/random/generated/numpy.random.choice.html>
- + https://en.wikipedia.org/wiki/K-means_clustering
- + <https://www.youtube.com/watch?v=4b5d3muPQmA>
- + <https://www.youtube.com/watch?v=oG1t3qlzq14>
- + <https://www.youtube.com/watch?v=oG1t3qlzq14>

- + <https://stackoverflow.com/questions/1401712/how-can-the-euclidean-distance-be-calculated-with-numpy>
- + <https://github.com/kieuconghau/color-compression>:

Tham khảo cách dán nhãn label

HẾT