ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

HỌC PHẦN TOÁN ỨNG DỤNG VÀ THỐNG KÊ CHO CÔNG NGHỆ THÔNG TIN (MTH000057 – 21CLC01)

BÁO CÁO PROJECT 1

GIẢNG VIÊN LÝ THUYẾT
GIẢNG VIÊN THỰC HÀNH
SINH VIÊN THỰC HIỆN

- ✓ Nguyễn Đình Thức
- ✓ Nguyễn Văn Quang Huy
- ✓ Ngô Đình Hy
- ✓ Đinh Công Huy Hoàng - 21127507

MỤC LỤC

I. REDUCE COLOR IMAGE:

a. Ý tưởng thực hiện:

Sử dụng thuật toán K – means clustering để phân cụm các nhóm màu trong ảnh thành các cụm dữ liệu. Các dữ liệu được phân cụm có thể liên quan đến nhau, ở đây dùng phân cụm các màu có cùng nhóm với nhau trên bảng màu RGB. Từ đó ta thể chọn các màu đặc trưng cho từng nhóm màu được chia trong ảnh.

Cách thức thực hiện của thuật toán:

- Khởi tạo k_cluster điểm dữ liệu tạm thời được coi là tâm của các cụm dữ liệu đang xét.
- Với mỗi điểm dữ liệu, ta sẽ xác định 1 tâm cụm gần với điểm dữ liệu đang xét nhất.
- Sau khi tất cả điểm điểm dữ liệu đã có tâm, tính toán lại vị trí của tâm cụm mới và đảm bảo tâm cụm nằm chính giữa cụm dữ liệu.
- Thực hiện lặp đi lặp lại đến khi tâm cụm không đổi hoặc tâm tất cả các điểm dữ liệu không đổi.

b. Mô tả các hàm:

– Các hàm phụ:

```
def setup_Data(type, link, k_Cluster):
```

- **Tham số đầu vào:** loại tâm cụm muốn tạo *(type)*, link của ảnh *(link)*, số tâm cụm *(k_Cluster)*.
- **Mục đích:** chuẩn bị các dữ liệu đầu vào để chạy thuật toán.
- **Mô tả:** sử dụng hàm *Image.open* để mở ảnh và thực hiện xử lí và tạo các 3 – dim để dễ dàng thao tác và chỉnh sửa điểm ảnh.

```
def convert_backImage(data_Arr, central, labels):
```

- **Tham số đầu vào:** 3 – dim của ảnh *(data_Arr)*, list các tâm cụm *(central)*, nhãn dán ứng với tâm cụm *(labels)*.
- **Mục đích:** tạo ra một ảnh mới dựa trên các labels và central mới được tạo ra từ ảnh cũ sau khi chạy xong thuật toán, trả về ảnh.
- **Mô tả:** thực hiện chạy tất cả các điểm ảnh và gán lần lượt các điểm ảnh đã được tạo ra vào trong một 3 – dim mới để tạo ra một ảnh hoàn chỉnh.

```
def downloadFile(output, name, ext, image):
```

- **Tham số đầu vào:** địa chỉ lưu file *(output)*, tên file *(name)*, đuôi file *(ext)*, ảnh muốn lưu về *(image)*.
- **Mục đích:** lưu file về máy sau khi thực hiện chuyển đổi.

- **Mô tả:** cho người dùng nhập vào các dữ liệu cần thiết để lưu file. Sử dụng hàm save thực hiện lưu file về địa chỉ theo yêu cầu.

– Các hàm chính

```
def cal_Distance(temp, center):
```

- **Tham số đầu vào:** vị trí điểm ảnh đang xét *(temp)*, list các tâm cụm *(center)*.
- **Mục đích:** Tính toán khoảng cách từ các tâm cụm đến 1 điểm ảnh, trả về list khoảng cách Euclid giữa điểm đang xét và các tâm cụm.
- **Mô tả:** Lần lượt sử dụng hàm *np.linalg.norm* để tính toán khoảng cách 2 điểm ảnh. Sau đó thêm vào list khoảng cách của từ các tâm cụm đến điểm đang xét để trả về.

```
def create_Center(type,k_Cluster, data_Arr):
```

- **Tham số đầu vào:** loại tâm cụm muốn tạo *(type)*, số tâm cụm *(k_Cluster)*, 3 – dim của ảnh *(data_Arr)*
- **Mục đích:** tạo ra các tâm cụm ban đầu bằng phụ thuộc yêu cầu của người dùng muốn nhập.
- **Mô tả:** đối với loại tâm cụm là **'random'** là loại tâm cụm được tạo ngẫu nhiên không dựa trên một cơ sở nào, còn với tâm cụm là **'pixel'** là loại tâm cụm dựa trên ảnh có sẵn và từ đó tạo ra các loại tâm cụm.

```
def kmeans_Clus(data_Arr, central_Point, k_Cluster, max_Inter,
type):
```

- **Tham số đầu vào:** 3 – dim của ảnh *(data_Arr)*, list các tâm cụm *(central_Point)*, số lượng tâm cụm muốn tạo ra *(k_Cluster)*, số lần thực hiện lại (max_Inter), loại tâm cụm muốn tạo *(type)*.
- **Mục đích:** thực hiện thuật toán **K – means** để tìm ra tâm các cụm trong dữ liệu hình ảnh kiểm tra.
- **Mô tả:** tạo một ma trân labels tương ứng với ma trận ảnh. Xét từng điểm ảnh và dùng hàm cal_Distance để tính toán khoảng cách từ điểm ảnh đến các tâm cụm để rồi từ đó gán các giá trị 0, 1, 2,… tương ứng đối với index của tâm cụm gần với điểm ảnh đó nhất. Sau đó thực hiện tính toán lại tâm cụm dựa trên các labels tương ứng với từng điểm ảnh và thực hiện trong max_Inter lần. Sau khi chạy xong kiểm tra xem thuật toán có trả về giá trị Nan (not a number) hay không do sự trùng lập màu tâm cụm. Nếu có thì ta thực hiện lại thuật toán với vị trí tâm cụm lúc đầu khác nhau, nếu không trả về giá trị labels.

c. Kết quả thử nghiệm:

- **Ảnh gốc:**

Hình ảnh 1: Ảnh gốc (767 x 431)

- **Pixels – Random:**

- **3 cluster:**

Hình ảnh 2: Ảnh sau khi chạy lần lượt là Pixels và Random (ảnh được thu nhỏ một nửa bằng Word)

– **5 cluster:**

Hình ảnh 3: Ảnh sau khi chạy lần lượt là Pixels và Random (ảnh được thu nhỏ một nửa bằng Word)

– **7 cluster:**

Hình ảnh 4: Ảnh sau khi chạy lần lượt là Pixels và Random (ảnh được thu nhỏ một nửa bằng Word)

d. Nhận xét, đánh giá:

- Thuật toán **đơn giản và hiệu quả** ở hầu hết các trường hợp. Tuy nhiên khi hình ảnh có các cụm màu không quá rõ ràng và việc tạo tâm cụm ngẫu nhiên khoogn thật sự khác biệt sẽ dẫn đến trường hợp Nan – tâm cụm không có điểm ảnh nào.
- Do đó việc tạo tâm cụm dựa trên ***'pixels'*** tuy có thể màu gần sát với ảnh nhưng sẽ dễ dàng tạo đến trường hợp lỗi. Màu của ảnh được tạo bởi trường hợp ***'random'*** cũng **đẹp và chi tiết rõ nét hơn** do các cụm được phân chia rõ ràng.
- Việc gom cụm màu dựa trên thuật toán **K – means** sẽ làm cho ảnh màu không còn chính xác với ảnh gốc ban đầu. Nhưng bù lại sẽ tạo được **hiệu quả đồng nhất** của các vùng màu.
- Có sự **đánh đổi giữa tốc độ và chất lượng**, việc thực hiện tính toán càng lâu sẽ dẫn đến màu càng chính xác và các chi tiết càng rõ rệt hơn.

	Pixels	Random
Ưu điểm	Giúp đảm bảo các tâm cụm phản ánh đúng phân phối màu sắc giúp giữ lại các tính đặc trưng của ảnh giúp làm màu tự nhiên và chính xác hơn.	Tạo ra độ đa dạng và tính ngẫu nhiên cao cho kết quả. Giữ lại độ phong phú màu sắc trong ảnh.
Nhược điểm	Giảm tính ngẫu nhiên đa dạng của kết quả, dẫn đến việc không phân cụm tốt và mất mát chi tiết.	Kết quả giảm màu không phản ánh chính xác màu sắc của ảnh gốc. Tạo hiện tượng không tự nhiên nhất quán trong kết quả.

II. RESIZE IMAGE:

a. Ý tưởng thực hiện:

Dựa trên thuật toán Nearest Neighbor Isopolation để có thể tăng giảm kích cỡ (resize) và tăng giảm tỉ lệ của ảnh. Từ đó có thể thêm hoặc bớt các dòng điểm ảnh còn thiếu giúp kéo dài hoặc thu nhỏ lại kích thước của ảnh theo ý muốn.

Cách thực hiện thuật toán:

- Tính toán tỉ lệ giữa kích thước sau và trước khi thay đổi.
- Tạo lặp lại các pixel màu có sẵn ở ảnh cũ.
- Đối với mỗi pixel ảnh mới, tính toán index tương ứng giữa cũ và mới dựa trên hệ số tỉ lệ.
- Làm tròn đến giá trị nguyên gần nhất để có thể lấy ra pixel màu tương ứng.

b. Mô tả các hàm:

```
def resize_img(data_Arr,new_H,new_W):
```

Tham số đầu vào: 3 – dim của ảnh *(data_Arr)*, chiều cao mới *(new_H)*, chiều rộng mới *(new_W)*.

Mục đích: thay đổi kích cỡ hình ảnh theo ý muốn.

Mô tả: thực hiện tỉnh toán tỉ lệ giữa kích thước ảnh mới và ảnh cũ, sau đó ta tạo một 3 – dim tương ứng với kích thước mới và thực hiện gán tất cả các giá trị pixel tương ứng ở ảnh cũ sang ảnh mới theo tỉ lệ $(i_{old} * factor = i_{new})$ và sau đó thực hiện gán các giá trị còn thiếu bằng các dãy điểm ảnh gần nó nhát. Còn đối với việc thu nhỏ anh, ta thực hiện xóa pixel dựa trên index tương ứng mà không có sự chọn lọc.

c. Kết quả thử nghiệm:

– **Ảnh gốc:**

Hình ảnh 5: Ảnh gốc (767 x 431)

– **Ảnh sau khi thu nhỏ:**

Hình ảnh 6: Ảnh gốc sau khi được thu nhỏ bằng thuật toán (300 x 150)

- **Ảnh sau khi phóng to:**

Hình ảnh 7: Ảnh gốc sau khi được phóng to bằng thuật toán (1100 x 900)

d. Nhận xét, đánh giá:

- Thuật toán **đơn giản, dễ triển khai, không cần tính toán** phức tạp.
- Ảnh khi **phóng to** bằng thuật toán này sẽ được dublicate điểm ảnh từ những điểm gần nó do đó ảnh sẽ **không gây mờ và làm mất chi tiết** xuất hiện trong lúc thay đổi kích thước.
- Đối với việc **thu nhỏ ảnh**, việc xóa bỏ các vị trí điểm ảnh không có sự chọn lọc sẽ gây hiện tượng **mất chi tiết, điểm ảnh quan trọng.**
- Làm **giảm độ phân giải** của ảnh do các điểm ảnh bị dublicate lên.
- **Tốc độ xử lí cao, nhanh không** tốn quá nhiều thời gian cho việc xử lí.

III. THAM KHẢO:

- Machine Learning cơ bản **LINK**
- Github Phạm Đình Khanh **LINK**
- Image Processing NNI **LINK**
- ChatGPT **LINK**