BỘ GIÁO DỤC VÀ ĐÀO TẠO
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

BÁO CÁO

TOÁN ỨNG DỤNG VÀ THỐNG KÊ CHO CÔNG NGHỆ THÔNG TIN – 22CLC03

LAB01

|Giáo viên hướng dẫn|

Phan Thị Phương Uyên

|Học viên|

Tôn Thất Minh Quân - 22127349

PHẦN 1: MỤC LỤC

NỘI DUNG

1. Ý tưởng thực hiện.

- Đọc và chuẩn bị dữ liệu:

Đọc màu từ ảnh của đường dẫn và chuyển sang mảng của Numpy.
Chuyển đổi mảng 2 chiều sang mảng 1 chiều, trong đó mỗi dòng là mỗi pixel, mỗi cột là mỗi kênh màu (R (R Red), G (Green), B (Blue)).

- Khởi tạo các Centroids ban đầu: Có 2 phương pháp

Chọn giá trị ngẫu nhiên cho mỗi kênh màu RGB từ 0 đến 255 của mỗi Centroid.
Chọn ngẫu nhiên không trùng nhau 10 pixel ở trên ảnh, và tính trung bình cho mỗi kênh màu và gán vào Centroid.

- Tìm các điểm hội tụ Centroid:

Gán nhãn: Tính toán khoảng cách giữa mỗi pixel với các centroids (sử dụng thuật toán Euclid).
Gán nhãn cho mỗi pixel với centroid gần nhất.

Cập nhật centroid:

Tính toán lại các centroid bằng cách lấy giá trị trung bình các pixel được gán với centroid đấy.
Lặp lại các bước trên đến khi Hội tụ hoặc chạm đến số lượng lặp tối đa.

- Điều kiện hội tụ:

Khác biệt giá trị của các centroid trong 3 lần lặp liên tiếp bé hơn giá trị 0,1.
Tính khoảng cách giữa mỗi centroid -> Lấy giá trị lớn nhất trong các khoảng cách so sánh với 0,1.
Nếu bé hơn trong 3 lần lặp liên tiếp thì sẽ tạm dừng thuật toán.

- Tạo ảnh mới từ Centroid và Label:

Thay thế các pixel với các centroid tương ứng với pixel ấy.
Đối với phương pháp chọn Centroid bằng cách gán giá trị ngẫu nhiên cho mỗi chiều màu sẽ xuất hiện tình trạng mất màu (không có pixel nào được gán với một centroid).
Để tránh tình trạng ấy, ta thực hiện so sánh số lượng giá trị khác nhau trong mảng labels với số lượng cluster (k_cluster).
Nếu ít hơn số lượng cluster: Thực hiện việc gán lại giá trị cho centroid.

2. Mô tả các hàm.

Hàm đọc ảnh:

Sử dụng Image trong thư viện PIL. Dùng hàm Image.open(path) để tải ảnh lên thông qua địa chỉ.

Img.convert('RGB') sẽ chuyển đổi ảnh sang chế độ màu RGB.

Sau đó chuyển ảnh sang mảng Numpy.

Hàm hiển thị ảnh:

Sử dụng thư viện matplotlib để hiển thị một ma trận dưới dạng ảnh.

```
plt.imshow(img_2d, cmap='gray')
```

Câu lệnh giấu trục toạ độ

```
plt.axis('off')
```

Hàm hiển thị ảnh:

```
plt.show()
```

Hàm lưu ảnh:

Sử dụng thư viện PIL
Câu lệnh tạo hình ảnh từ mảng 2 chiều.

```
img = Image.fromarray(img_2d)
```

Hàm để lưu ảnh dưới dạng jpg:

```
img.save(img_path)
```

Hàm để lưu ảnh dưới png:

```
img.save(img_path, "PNG")
```

Hàm chuyển đổi ảnh từ kích thước 2D(height, width, channels) sang 1D (height*width, channels):

Lấy kích thước của ảnh 2D(height, width, channels).

Img_1d = img_2d.reshape(height*width, channels) sẽ chuyển mảng theo kích thước tương ứng.

Hàm gom nhóm màu sử dụng K-means:

- Có 2 phương pháp chọn các giá trị cho centroid ứng với tham số đầu vào init_centroids:

Phương pháp 1: Init_centroids = 'random':

Chọn giá trị ngẫu nhiên cho mỗi kênh màu RGB từ 0 đến 255 của mỗi Centroid.
Với mỗi centroid thứ i, ta có câu lệnh:

```
centroids[i] = np.random.randint(0, 256, size=n, dtype=np.uint8)
```

Lựa chọn ngẫu nhiên 3 số từ 0 đến 255 dạng uint8.

Phương pháp 2: Init_centroids = 'in_pixels':

- Chọn ngẫu nhiên không trùng nhau 10 pixel ở trên ảnh, và tính trung bình cho mỗi kênh màu và gán vào Centroid.

Câu lệnh chọn ra 10 vị trí ngẫu nhiên không lặp lại từ 0 đến m – 1:

```
rand_indices = np.random.choice(m, size=10, replace=False)
```

Tính giá trị trung bình cho 10 giá trị lựa chọn ở trên và gán vào centroid thứ i.

```
centroids[i] = np.mean(img_1d[rand_indices], axis=0)
```

- axis = 0 có nghĩa tính giá trị trung bình theo trục 0 (trục hàng), vậy là tính trung bình theo mỗi chiều RGB. Giá trị trả về sẽ là 1 vector có kích thước là 3.

Tìm các điểm hội tụ:

Bằng cách gán nhãn cho mỗi pixel với centroid gần chúng nhất.
Câu lệnh tính khoảng cách mỗi pixel trong img_1d với tất cả các centroid trong centroids .
Sử dụng hàm np.linalg.norm sử dụng Euclide

```
distances = np.linalg.norm(img_1d[:, np.newaxis] - centroids, axis=2)
```

- Mảng img_1d có kích thước(height*width, channels) với height, width là độ cao và độ rộng của ảnh, channels là số chiều của ảnh (ở đây là 3 chiều RGB).
- img_1d[:, np.newaxis]: Thêm 1 chiều mới vào mảng, làm cho nó có kích thước (height*width, 1, channels)
- centroid là mảng có kích thước (k_cluster, channels) với k_cluster là số lượng cluster.

- Img_1d[:np.newaxis] – centroid: Phép trừ này sẽ thực hiện trên trục mới thêm vào của img_1d. Tạo ra mảng có kích thước (num_points, k_cluster, channels) sẽ tính khoảng cách Euclide mỗi pixel đến mỗi centroid và lưu vào trong mảng.
- Axis = 2 có nghĩa tính khoảng cách trên trục 2 của mảng 3 chiều tạo ra mảng 2 chiều có kích thước (height*width, k_clusters). Trong đó là khoảng cách của mỗi pixel tới mỗi centroid.

Np.argmin(distances, axis =1)

Trả về chỉ số của phần tử có giá trị nhỏ nhất của trục được chỉ định.
Axis = 1: chỉ định trục 1. Có nghĩa là mỗi hàng tương ứng với dữ liệu khoảng cách của 1 pixel, np.argmin sẽ tìm chỉ số giá trị nhỏ nhất trong hàng đó.
Kết quả trả về là mảng 1 chiều có kích thước height*width.

Cập nhật centroid:

Sau khi gán nhãn, ta sẽ cập nhật lại giá trị cho mỗi centroid với giá trị trung bình của tất cả dữ liệu thuộc mỗi centroid.

Câu lệnh trích xuất các pixel thuộc k:

cluster_points = img_1d[labels == k]

- labels là mảng 1 chiều chứa nhãn cho mỗi pixel. Mỗi phần tử đại diện cho centroid của mỗi pixel
- labels == k sẽ tạo ra 1 mảng boolean có cùng kích thước với mảng labels (height*width), trong đó, giá trị True nếu giá trị trong labels là k và False là ngược lại
- img_1d[labels == k] trả về các hàng của mảng img_1d mà tương ứng với các vị trí True trong mảng boolean.
- Trả về kết quả là cluster_points chứa tất cả các pixel thuộc centroid k.

Tiếp theo sẽ tính giá trị mới cho centroid:

centroids[k] = np.mean(cluster_points, axis=0)

- Hàm tính giá trị trung bình cho mỗi kênh màu. Trả về mảng có kích thước (channels).

Hàm tạo ảnh mới từ các màu trung tâm:

Câu lệnh tạo ma trận 0 để chứa các pixel mới:

new_img = np.zeros(img_2d_shape, dtype=np.uint8)

- Tạo ra mảng có kích thước (height, width, channels) với mỗi phần tử có kiểu dữ liệu uint8.

Tạo hình ảnh mới dựa trên centroids:

new_img = centroids[labels].reshape((height, width, 3))

- labels là mảng 1 chiều chứa nhãn cho mỗi pixel.
- centroids[labels]: thay thế mỗi nhãn trong labels bằng giá trị các centroid tương ứng. Tạo ra mảng 2 chiều chứa giá trị các điểm ảnh.
- Hàm reshape sẽ biến mảng 3 chiều thành mảng 2 chiều tức là hình ảnh mới có cùng kích thước với hình ảnh gốc nhưng các điểm ảnh đã được thay thế.

Hàm dừng sớm có điều kiện:

Câu lệnh tạo hàm chứa giá trị centroid trong vòng lặp trước.

pre_centroids = centroids.copy()

Hàm copy() sẽ giúp chúng ta thay đổi giá trị trên centroid mà không ảnh hưởng đến pre_centroid.

Tạo 1 biến đếm times = 0 để đếm số lượng vòng lặp mà các sự khác nhau giữa các giá trị centroid dưới giá trị cho phép (ở đây là 0,1).

centroid_shift = np.linalg.norm(centroids - pre_centroids, axis=1)

Tương tự như câu lệnh trên, hàm sẽ tính khoảng cách Euclide giữa các centroid tương ứng và lưu vào mảng 1 chiều có kích thước k_clusters.

Câu lệnh điều kiện so sánh giá trị lớn nhất của mảng (khoảng cách lớn nhất giữa các centroid tương ứng) mới tìm được.

If np.max(centroid_shift) < 0.1

Nếu thoả mãn thì biến đếm times sẽ tăng lên 1. Lúc đó sẽ kiểm tra giá trị biến đếm nếu lớn hơn 3 thì vòng lặp sẽ kết thúc.

Hàm tránh trường hợp mất màu khi dùng phương pháp chọn centroid 'random':

- Ý tưởng: Sau khi gán nhãn cho mỗi pixel, thì ta sẽ đếm số lượng các giá trị khác nhau và đem so sánh với k_cluster:
+ Nếu bằng nhau thì không có trường hợp mất màu.
+ Nếu bé hơn thì có xảy ra và buộc chương trình chạy lại đoạn code tìm centroid.
Câu lệnh điều kiện:

if len(np.unique(labels)) < k_clusters:

Hàm np.unique sẽ tạo ra mảng chứa các giá trị khác nhau trong labels. Hàm len() sẽ đếm số lượng phần tử trong labels.

Hàm main:

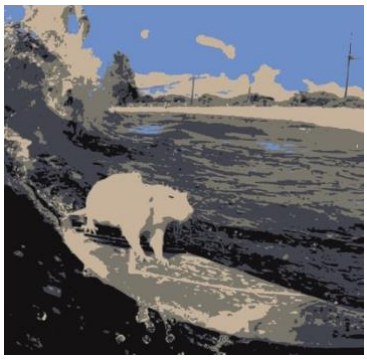
Hàm sẽ cho người dùng nhập địa chỉ hình ảnh khi chương trình thực thi.
Chương trình lưu hình ảnh mới tạo dưới dạng png và jpg.

Kiểm tra hiệu quả của chương trình:

Hình ảnh được kiểm tra:

Hình ảnh với k = 3:

Hình ảnh với k = 5



Hình ảnh với k = 7:

Nhận xét:

- Thời gian thực thi của thuật toán phụ thuộc vào số lượng pixel và số lượng cluster. Số lượng càng lớn thì thời gian thực thi càng nhiều.
- Kích thước file: Việc nén ảnh bằng K-means có thể giảm kích thước ảnh so với ảnh gốc nếu số lượng k_cluster càng nhỏ
- Nếu chọn được số k_cluster thì ta sẽ được ảnh nén với độ chi tiết tốt mà không gì với ảnh gốc. Nếu chọn k_cluster không phù hợp thì chất lượng hình ảnh giảm đi đáng kể, mất đi tính thẩm mỹ.

THAM KHẢO

- Adith Bharadwaj, “Image compression using K-means clustering”, Geeksforgeeks, 29/5/2023, https://www.geeksforgeeks.org/image-compression-using-k-means-clustering/, 21:00 19/6/2024