TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG - HCM
KHOA CÔNG NGHỆ THÔNG TIN

---o0o---

BÁO CÁO ĐỒ ÁN THỰC HÀNH

CORLOR COMPRESSION

Môn : **Toán ứng dụng và thống kê cho công nghệ thông tin**

Giảng Viên : **Phan Thị Phương Uyên**

Lớp : **21CLC05**

Tên : **Huỳnh Minh Quang**

MSSV : **21127149**

Thành phố Hồ Chí Minh, ngày 15 tháng 07 năm 2023

MỤC LỤC

1. Ý tưởng thực hiện

- Chương trình này được viết để giảm số lượng màu trong một tệp hình ảnh sử dụng thuật toán K-Means. Ý tưởng chính của chương trình là tìm các cụm màu chính trong hình ảnh và thay thế mỗi pixel trong hình ảnh bằng một màu trong cụm màu đó. Các bước thực hiện chương trình bao gồm:
    - Đọc tệp hình ảnh đầu vào sử dụng thư viện PIL.
    - Chuyển đổi hình ảnh thành một mảng numpy để thực hiện các phép tính và xử lý dữ liệu.
    - Thực hiện K-Means clustering trên mảng hình ảnh 1D để tìm các điểm trung tâm của các cụm màu.
    - Gán nhãn cho từng pixel trong hình ảnh dựa trên các điểm trung tâm đã tìm được.
    - Thay thế mỗi pixel trong hình ảnh bằng một màu từ cụm màu tương ứng.
    - Lưu hình ảnh đã giảm số lượng màu vào tệp mới.
    - Hiển thị hình ảnh gốc và hình ảnh đã giảm số lượng màu sử dụng thư viện matplotlib.

2. Mô tả các hàm

- **Hàm initial_Centroids(img, k_clusters, initCentroids):** Hàm này được sử dụng để khởi tạo các điểm trung tâm cho các cụm màu ban đầu. Có hai cách khởi tạo được hỗ trợ trong code này:

```python
# Initialize centroids
def initial_Centroids(img, k_clusters, initCentroids):
    if initCentroids == 'random':
        # Random initialization
        return np.random.randint(0, 256, size=(k_clusters, img.shape[1]))
    elif initCentroids == 'in_pixels':
        # Initialize centroids from random pixels in the image
        rand_indices = np.random.choice(img.shape[0], size=k_clusters, replace=False)
        return img[rand_indices]
    else:
        return None
```

    - Nếu initCentroids là "random", các điểm trung tâm sẽ được khởi tạo ngẫu nhiên trong khoảng từ 0 đến 255 cho mỗi kênh màu.
    - Nếu initCentroids là "in_pixels", các điểm trung tâm sẽ được khởi tạo từ các điểm

ảnh ngẫu nhiên trong hình ảnh ban đầu.

- **Hàm update_Centroids(img, k_clusters, labels, old_centroids):** Hàm này được sử dụng để cập nhật các điểm trung tâm của các cụm màu. Với mỗi cụm màu, hàm tính trung bình của các điểm ảnh thuộc cụm đó và cập nhật điểm trung tâm tương ứng.

```
# Update centroids
def update_Cetroids(img, k_clusters, labels, old_centroids):
    centroids=old_centroids
    for i in range(k_clusters):
        mask = labels == i
        if np.any(mask):
            centroids[i] = np.mean(img[mask], axis=0)
        return centroids
```

- **Hàm label_Pixels(img, centroids):** Hàm này được sử dụng để gán nhãn cho từng điểm ảnh trong hình ảnh ban đầu dựa trên các điểm trung tâm của các cụm màu. Hàm tính khoảng cách giữa mỗi điểm ảnh và các điểm trung tâm, sau đó gán nhãn cho điểm ảnh bằng cụm màu gần nhất.

```
def label_Pixels(img, centroids):
    # Assign labels to pixels
    distances = np.linalg.norm(img[:, np.newaxis] - centroids, axis=-1)
    return np.argmin(distances, axis=1)
```

- **Hàm kmeans(img_1d, k_clusters, max_iter, initCentroids):** Hàm này thực hiện thuật toán K-Means clustering để giảm số lượng màu trong hình ảnh. Hàm nhận đầu vào là một mảng 1D của hình ảnh (img_1d), số lượng cụm (k_clusters), số lần lặp tối đa (max_iter) và cách khởi tạo điểm trung tâm (initCentroids). Trong hàm này, các bước cơ bản của thuật toán K-Means được thực hiện:

```python
def kmeans(img_1d, k_clusters, max_iter, initCentroids):
    # Initialize cluster centroids and label
    centroids=initial_Centroids(img_1d, k_clusters, initCentroids)
    labels = label_Pixels(img_1d, centroids)
    # Run K-means
    for _ in range(max_iter):
        old_centroids=centroids.copy()
        centroids=update_Cetroids(img_1d, k_clusters, labels, old_centroids)
    # Check convergence
        if np.allclose(old_centroids, centroids, rtol=1e-3, equal_nan=False):
            break
    return centroids, labels
```

- Khởi tạo các điểm trung tâm ban đầu.
- Gán nhãn cho từng điểm ảnh dựa trên các điểm trung tâm.
- Cập nhật các điểm trung tâm bằng cách tính trung bình của các điểm ảnh thuộc cụm tương ứng.
- Kiểm tra điều kiện dừng để xác định khi nào thuật toán đã hội tụ.

- **Hàm image_Processing(image_File_Name, k):** Hàm này thực hiện quá trình xử lý hình ảnh. Đầu tiên, hình ảnh được đọc từ tệp được chỉ định (image_File_Name). Sau đó, hình ảnh được chuyển đổi thành một mảng numpy và được chuyển thành một mảng 2D để thực hiện K-Means clustering. Hàm cũng khởi tạo giá trị cho các tham số như số lần lặp tối đa và cách khởi tạo điểm trung tâm. Kết quả cuối cùng là hình ảnh gốc và hình ảnh đã giảm số lượng màu.

```python
def image_Processing(image_File_Name, k):
    # Load the image
    image = Image.open(image_File_Name)
    # Convert the image to a numpy array
    img_array = np.array(image)

    # Reshape the array to a 2D matrix
    img_2d = img_array.reshape(-1, img_array.shape[-1])
    # Init value
    max_iter = 1000
    initCentroids = 'random'

    centroids, labels = kmeans(img_2d, k, max_iter, initCentroids)
    # Update the pixel values based on the cluster centroids
    reduced_img_2d = centroids[labels]

    # Reshape the reduced image back to its original shape
    reduced_img_array = reduced_img_2d.reshape(img_array.shape)

    # Create a PIL image from the reduced image array
    reduce_Image=Image.fromarray(reduced_img_array.astype(np.uint8))
    return image, reduce_Image
```

- **Hàm Display(image, reduced_image, k):** Hàm này được sử dụng để hiển thị hình ảnh gốc và hình ảnh đã giảm số lượng màu. Hai hình ảnh được hiển thị cùng một lúc bên cạnh nhau trong một cửa sổ đồ họa. Điều này giúp chúng ta so sánh hình ảnh gốc và hình ảnh đã giảm số lượng màu với số lượng màu đã được chỉ định (k).

```python
# Display the original and reduced images
def Display(image, reduced_image, k):
    fig, axes = plt.subplots(1, 2, figsize=(10, 5))
    axes[0].imshow(image)
    axes[0].set_title("Original Image")
    axes[0].axis("off")
    axes[1].imshow(reduced_image)
    axes[1].set_title("Reduced Image (k = {})".format(k))
    axes[1].axis("off")
    plt.show()
```

- **Hàm Save_img(reduce_Image):** Hàm này được sử dụng để lưu hình ảnh đã giảm số lượng màu vào tệp mới. Người dùng được yêu cầu nhập định dạng đầu ra (png hoặc pdf) và hình ảnh được lưu với tên tệp là "reduced_image" và định dạng tương ứng.

```python
# Save the reduced image
def Save_img( reduce_Image):
    # Prompt the user to choose the output format
    output_format = input("Enter the output format (png/pdf): ")
    fileName = "reduced_."+ output_format
    reduce_Image.save(fileName)
```

- **Hàm main():** Đây là hàm chính của chương trình. Trong hàm này, người dùng được yêu cầu nhập tên tệp hình ảnh và số lượng màu cần giảm. Sau đó, chương trình thực hiện quá trình giảm số lượng màu và hiển thị hình ảnh gốc và hình ảnh đã giảm số lượng màu.

```python
# Main
if __name__ == '__main__':
    # Prompt the user to enter the image file name
    input_File_Name= input("Enter the image file name: ")
    # Prompt the user to enter the
    k = int(input("Enter the number of colors to reduce to: "))

    init_Image, reduce_Image=image_Processing(input_File_Name, k)

    Display(init_Image, reduce_Image, k)

    Save_img( reduce_Image)
```

3. Hình ảnh kết quả

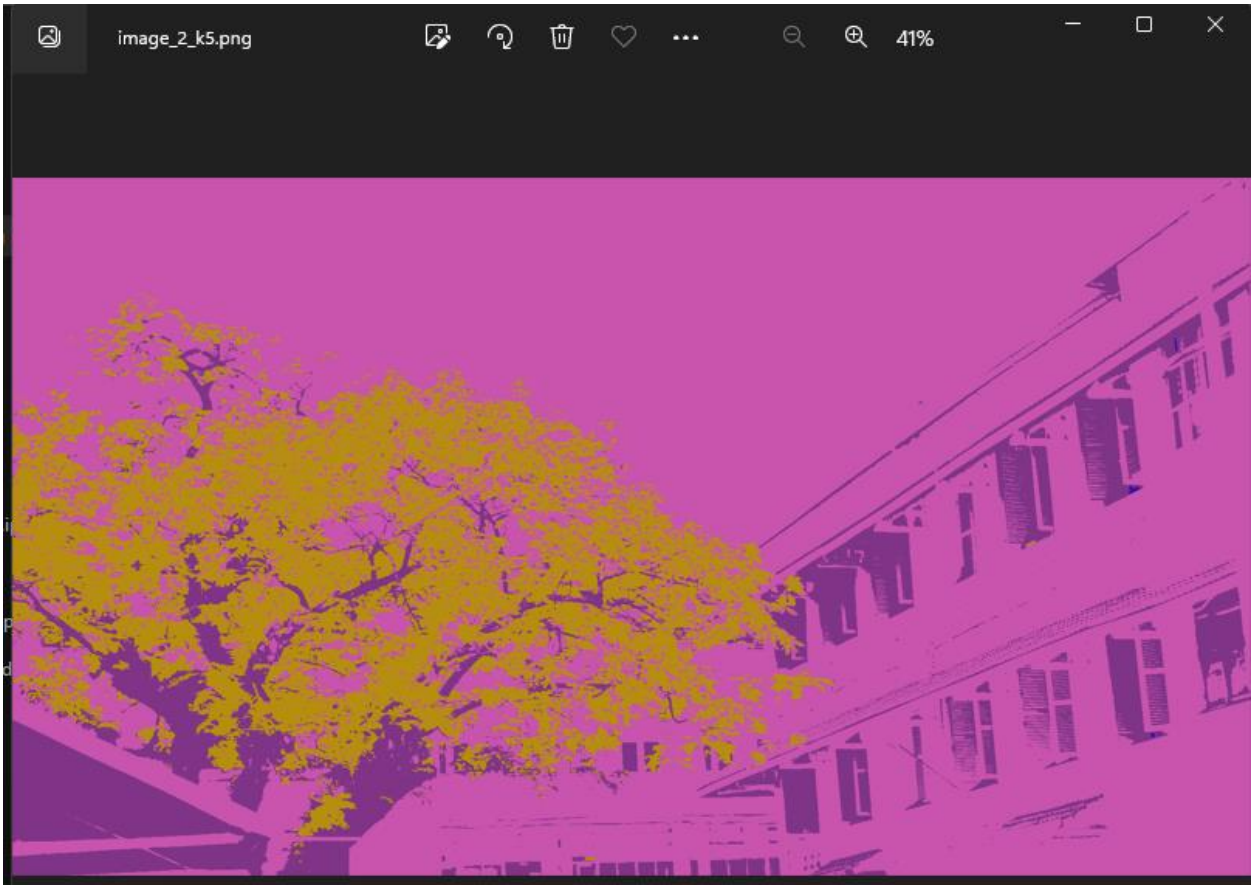
- **Hình ảnh đã giảm số lượng màu với k = 3:**

- **Hình ảnh đã giảm số lượng màu với k = 5:**

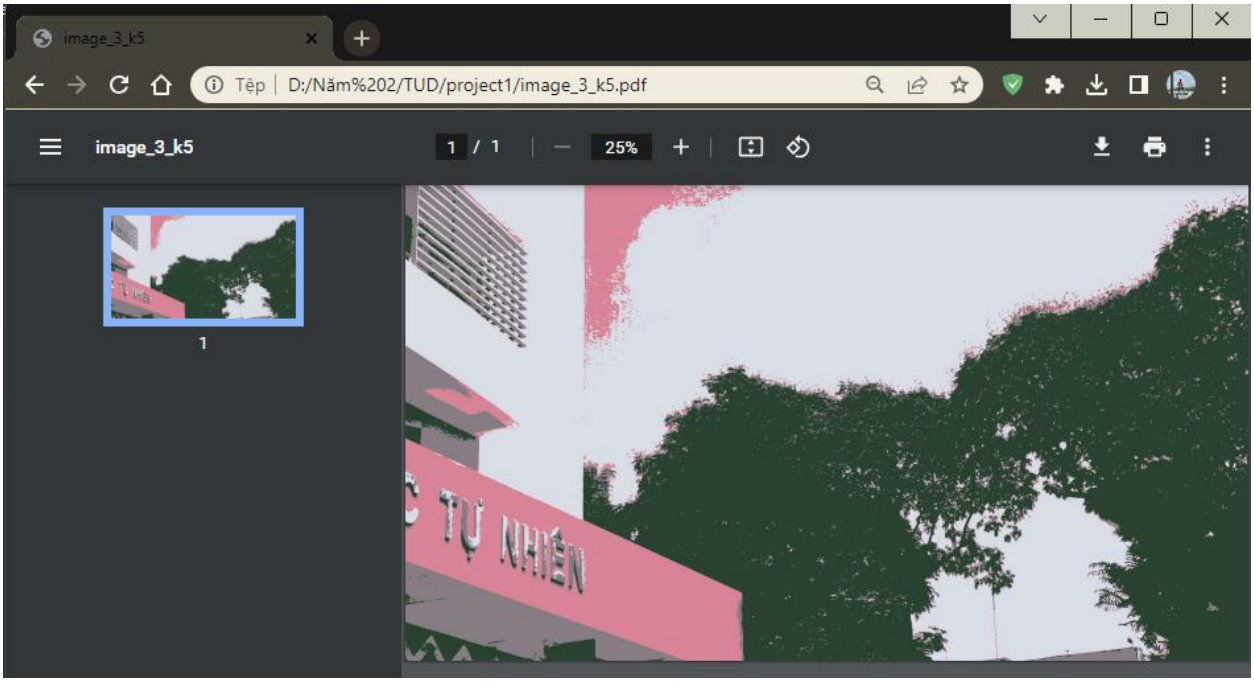
- **Hình ảnh đã giảm số lượng màu với k = 7:**

- **Hình ảnh đã giảm số lượng màu với k = 11:**

- **Hình ảnh lưu dạng .png:**



- **Hình ảnh lưu dạng .pdf:**



4. Nhận xét kết quả

- Sau khi chạy chương trình với các giá trị k khác nhau, chúng ta có thể nhận xét về kết quả như sau:
    - Với số lượng màu k = 3, hình ảnh đã giảm số lượng màu trông rất đơn giản và chỉ sử dụng một số màu cơ bản để biểu diễn hình ảnh. Điều này có thể tạo ra hiệu ứng một hình ảnh tương tự với hình ảnh dạng hoạt hình hoặc hình ảnh mang tính biểu tượng cao.
    - Với số lượng màu k = 5, hình ảnh đã giảm số lượng màu vẫn giữ được nhiều chi tiết hơn so với k = 3. Điều này làm cho hình ảnh trông tự nhiên hơn và vẫn giữ được một số đặc điểm của hình ảnh gốc.
    - Với số lượng màu k = 7, hình ảnh đã giảm số lượng màu có thể tái tạo được nhiều chi tiết hơn và gần gũi với hình ảnh gốc hơn. Điều này làm cho hình ảnh trông phong phú hơn với nhiều màu sắc khác nhau.
- **Nhận xét:** Qua việc thay đổi số lượng màu, chúng ta có thể thấy rằng số lượng màu ảnh hưởng trực tiếp đến diện mạo và cảm nhận của hình ảnh. Giảm số lượng màu có thể làm mất một số thông tin màu sắc và chi tiết của hình ảnh gốc, nhưng đồng thời cũng có thể tạo ra hiệu ứng đặc biệt và làm nổi bật các yếu tố khác của hình ảnh.

5. Tài liệu tham khảo

- https://blog.luyencode.net/
- https://www.geeksforgeeks.org/
- https://www.kaggle.com/

___Hết___