ĐẠI HỌC QUỐC GIA

THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

BÁO CÁO ĐỒ ÁN 1

COLOR - COMPRESSION

MÔN: TOÁN ỨNG DỤNG VÀ THỐNG KÊ CHO CÔNG NGHỆ THÔNG TIN

Sinh viên : Nguyễn Hoài Mẫn

MSSV : 20127561

LỚP : 20CLC05

Khoa : Công nghệ thông tin

TP. Hồ Chí Minh, ngày 18 tháng 6 năm 2022

MỤC LỤC

I. Ý tưởng thực hiện và mô tả các hàm: 1
1. Ý tưởng thực hiện 1
2. Mô tả các hàm 1
2.1 Hàm khởi tạo centroids 1
2.2 Hàm trả về index khoảng cách gần nhất từ centriods .. 1
2.3 Hàm cập nhật lại vị trí của các centroids 2
2.4 Hàm chạy thuật toán Kmeans 2
2.5 Hàm compress Image 3
2.6 Hàm nhập thông tin đầu vào 3
2.7 Hàm main 4
II. Hình ảnh với từng k_clusters: 6
1. Với k_cluster = 3, max interator = 50: 6
2. Với k_cluster = 5, max interator = 50: 7
3. Với k_cluster = 7, max interator = 50: 7
4. Nhận xét: 7
III. TÀI LIỆU THAM KHẢO 8

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

1. Ý tưởng thực hiện

B1: Khởi tạo K điểm dữ liệu trong bộ dữ liệu và tạm thời coi nó là tâm của các cụm dữ liệu của chúng ta.

B2: Với mỗi điểm dữ liệu trong bộ dữ liệu, tâm cụm của nó sẽ được xác định là 1 trong K tâm cụm gần nó nhất.

B3: Sau khi tất cả các điểm dữ liệu đã có tâm, tính toán lại vị trí của tâm cụm để đảm bảo tâm của cụm nằm ở chính giữa cụm.

B4: Bước 2 và bước 3 sẽ được lặp đi lặp lại cho tới khi vị trí của tâm cụm không thay đổi hoặc tâm của tất cả các điểm dữ liệu không thay đổi.

2. Mô tả các hàm

2.1 Hàm khởi tạo centroids

```
#Khởi tạo centroids giá trị ngẫu nhiên
def InitCentroids(self,img):
    m,n = img.shape
    centriod = np.zeros((self.k_clusters,n))
    for i in range(self.k_clusters):
        centriod[i] = img[np.random.randint(0,m+1),:]
    return centriod
```

- Hàm khởi tạo giá trị centroids ngẫu nhiên.

2.2 Hàm trả về index khoảng cách gần nhất từ centriods

```
# Hàm trả về index khoảng cách gần nhất từ centriods
def Idx_pixels(self,img,centriods):
    m,n = img.shape
    l = len(centriods)
    idx =  np.zeros((m,1))
    for i in range(m):
        D = np.zeros((1,self.k_clusters))
        #Với mỗi pixel, lấy khoảng cách tối thiểu (khoảng cách Euclide) và gắn index pixel đó
        for j in range(l):
            D[:,j] = np.sqrt(np.sum(np.power((img[i,:]-centriods[j,:]),2)))#tính toán khoảng cách
        # return index of the closest center
        idx[i] = np.argmin(D) + 1
    return idx
```

- Với mỗi điểm dữ liệu trong tập dữ liệu, tâm cụm của nó sẽ là 1 trong số k_cluster tâm cụm gần với nó nhất. Hàm trên sẽ tính toán khoảng cách giữa 2 điểm bằng cách sử dụng Euclidean distance rồi trả về index khoảng cách gần nhất từ centriods.

2.3 Hàm cập nhật lại vị trí của các centroids

```
#Cập nhật lại vị trí của các centroids
def UpdateCentroids(self,img,idx):
    m,n = img.shape
    centroids = np.zeros((self.k_clusters,n))
    count = np.zeros((self.k_clusters,1))
    # Tính toán centroid mới bằng cách sử dụng giá trị trung bình trên các hàng với mỗi cụm
    for i in range(m):
        index = int(idx[i]-1)
        centroids[index,:] += img[i,:]
        count[index]+=1
    return centroids/count
```

- Tính toán lại tọa độ của mỗi tâm cụm được thực hiện bằng cách lấy trung bình cộng tọa độ của tất cả các điểm dữ liệu của cụm. Sau khi tính toán xong, vị trí mới của tâm cụm sẽ nằm chính giữa cụm của nó.

2.4 Hàm chạy thuật toán Kmeans

```
def RunKmeans(self,img,centroid,max_iter):
    idx = self.Idx_pixels(img,centroid)
    # Run K-means
    for i in range(max_iter):
        # Tính toán lại centroids and idx dựa vào clusters hiện t
        centroid =  self.UpdateCentroids(img,idx)
        idx = self.Idx_pixels(img,centroid)
    return centroid,idx
```

- Các thông số truyền vào hàm là img, centroid, max_iter. Sau số vòng lặp là max_iter hàm sẽ xử lý và trả về centroid và idx.

- Hàm sẽ tìm bộ centroid mới cho đến khi idx của các điểm ảnh không có sự thay đổi nữa hoặc đạt tới giới hạn lặp.

2.5 Hàm compress Image

```
def CompressedImage(img, k_clusters, max_iter):
    kmeans = Kmeans(k_clusters, max_iter)
    #Khởi tạo centroids
    initial_centriod = kmeans.InitCentroids(img)
    # run kmeans
    compressed_Centriod,conpressed_idx = kmeans.RunKmeans(img,initial_centriod,max_iter)
    img_Compressed = img.copy()
    #Trả về chế độ xem của mảng ban đầu
    for i in range(1,k_clusters+1):
        img_Compressed[(conpressed_idx == i).ravel(),:] = compressed_Centriod[i-1]
    return img_Compressed
```

- Hàm dùng để gom nhóm các điểm ảnh của bức tranh về k_cluster cụm, sau đó nén lại thành ma trận ảnh.

2.6 Hàm nhập thông tin đầu vào

```
def Input():
    s = input("import your image file: ")
    r = os.path.exists(s)
    while True:
        if r == True:
            break
        s = input('File does not exist! Please re-enter:')
        r = os.path.exists(s)
    img = mpimg.imread(s)
    imgplot = plt.imshow(img)
    plt.show()

    max_iter = int(input("import max iterator: "))
    while max_iter < 0:
        if max_iter > 0:
            break
        max_iter = int(input('Max iterator must be greater than 0! Please re-enter:'))

    k_clusters = int(input("import k_clusters: "))
    while k_clusters < 0:
        if k_clusters > 0:
            break
        k_clusters = int(input('k must be greater than 0! Please re-enter:'))
    return img, max_iter, k_clusters
```

- Hàm dùng để nhập và kiểm tra thông tin đầu vào gồm img, k_clusters và max iter, nếu nhập sai sẽ yêu cầu nhập lại.

- Các điều kiện nhập:

+ Image phải tồn tại.

+ Max_iter phải lớn hơn 0.

+ K_cluster phải lớn hơn 0.

2.7 Hàm main

```
def main():
    A, max_iter, k_clusters = Input()
    img = (A/255).reshape(A.shape[0]*A.shape[1],3)
    img_Compressed = CompressedImage(img,k_clusters,max_iter)
    img_Compressed = img_Compressed.reshape(A.shape[0],A.shape[1],3)
    plt.imshow(img_Compressed)

    print("Choice 1. If you want to save as png format")
    print("Choice 2. If you want to save as pdf format")
    print("Choice 3. If you don't want to save")
    choice = int(input("Enter your choice: "))
    while (True):
        if choice == 1:
            temp = input("Enter the name of the file you want to save: ")
            plt.imsave(temp +'.png',img_Compressed )
            print("save successfully")
            break
        if choice == 2:
            temp = input("Enter the name of the file you want to save: ")
            plt.imsave(temp +'.pdf',img_Compressed )
            print("save successfully")
            break
        if choice == 3:
            break
        print("1. If you want to save as png format")
        print("2. If you want to save as pdf format")
        print("3. If you don't want to save")
        choice = int(input("Syntax error! Please re-enter: "))

    print("successful conversion!")
```

- Hàm dùng để thực thi các hàm cần thiết để chạy chương trình

- Hàm cho phép lựu chọn export ảnh theo ý muốn của người dùng

+ Lưu theo định dang png.

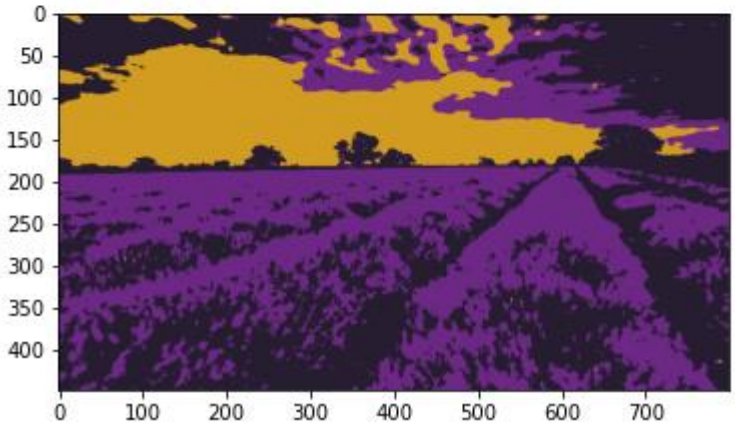
+ Lưu theo định dang pdf.

+ Không thực hiện lưu.

* Kết quả sau khi chạy hàm main:

```
import your image file: test.jpg
```


```
import max iterator: 50
import k_clusters: 3
Choice 1. If you want to save as png format
Choice 2. If you want to save as pdf format
Choice 3. If you don't want to save
Enter your choice: 3
successful conversion!
```



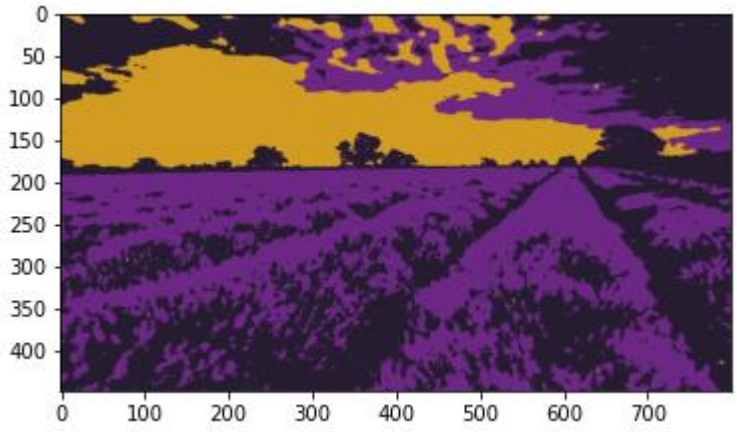
* Các file lưu:

test1.pdf	6/19/2022 1:40 PM	WPS PDF Docume...	65 KB
test1.png	6/19/2022 5:16 PM	PNG File	33 KB

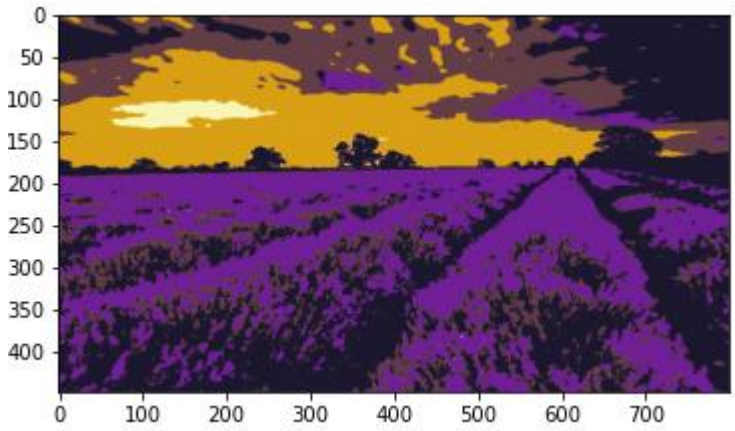
II. Hình ảnh với từng k_clusters:

* Hình ảnh ban đầu:

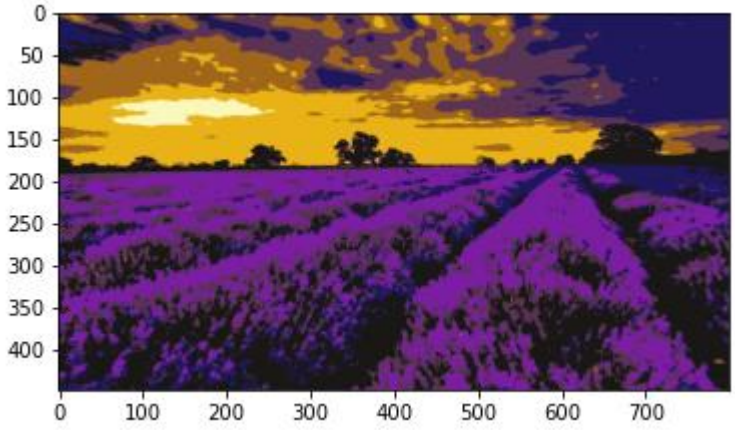
1. Với k_cluster = 3, max interator = 50:



2. Với k_cluster = 5, max interator = 50:



3. Với k_cluster = 7, max interator = 50:



4. Nhận xét:

- Các ảnh đều đã thực hiện thành công giảm số lượng màu về k_cluster tương ứng với 3, 5, 7.
- Với việc phân thành k_cluster cho bức ảnh, ta đã được các bức ảnh với số lượng màu khác nhau nhưng vẫn giữ được độ tương quan so với ảnh gốc.
- Ứng với số k_cluster càng lớn thì ảnh thể hiện càng được nhiều màu sắc hơn và theo đó thì kích thước bức ảnh cũng càng lớn hơn.

III. TÀI LIỆU THAM KHẢO

1. https://nguyenvanhieu.vn/thuat-toan-phan-cum-k-means/

2. http://ungdung.khoa-hnvd.com/Hoc_thuat/KMeans.html

3. https://www.youtube.com/watch?v=RaTve-ddaps&ab_channel=stepbystepdatascience

4. https://www.youtube.com/watch?v=8mUi6U_5ZCg&t=723s&ab_channel=stepbystepdatascience

5. https://www.youtube.com/watch?v=B08yMWVGWpk