

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

**MÔN HỌC : Toán ứng dụng và thống kê cho
Công nghệ thông tin**



ĐỒ ÁN THỰC HÀNH 1
REPORT

Tài liệu này mô tả nội dung đồ án môn học cho môn học Toán ứng dụng và thống kê cho Công nghệ thông tin.

Giảng viên hướng dẫn

Phan Thị Phương Uyên

Nguyễn Văn Quang Huy

Sinh viên thực hiện :

Phan Trí Nguyên - 20127578

Thành phố Hồ Chí Minh, ngày 14 tháng 6 năm 2022

MỤC LỤC

1	Tổng quan.....	3
1.1	Thông tin sinh viên.....	3
1.2	Thông tin đồ án.....	3
2	Ý tưởng thực hiện và mô tả chi tiết các hàm	4
2.1	Ý tưởng thực hiện.....	4
2.2	Mô tả các hàm.....	7
3	Demo kết quả chạy chương trình ứng với từng số lượng màu.....	11
4	Nhận xét kết quả demo chạy chương trình	17
4.1	Đánh giá kết quả.....	17
4.2	Nhận xét về đồ án Color Compression.....	17
5	Tài liệu tham khảo	19

1

Tổng quan

1.1 Thông tin sinh viên

MSSV	Họ và tên	Email	Lớp
20127578	Phan Trí Nguyên	20127578@student.hcmus.edu.vn phantringuyen2002@gmail.com	20CLC05

1.2 Thông tin đồ án

Tên đồ án	Môi trường lập trình
Color Compression	Phần mềm: Jupyter Notebook, Anaconda , Python 3.10, PyCharm Community Edition 2022.1.2. Ngôn ngữ: Python.

Mô tả bài toán:

- Để lưu trữ một bức ảnh với kích thước lớn thì cần lưu trữ kích thước lưu trữ trong bộ nhớ PC tương ứng.
- Vì vậy chúng ta cần giảm bớt số lượng màu của ảnh về một số lượng màu chính nhất định đồng thời vẫn giữ được nội dung ảnh được bảo toàn nhất có thể.

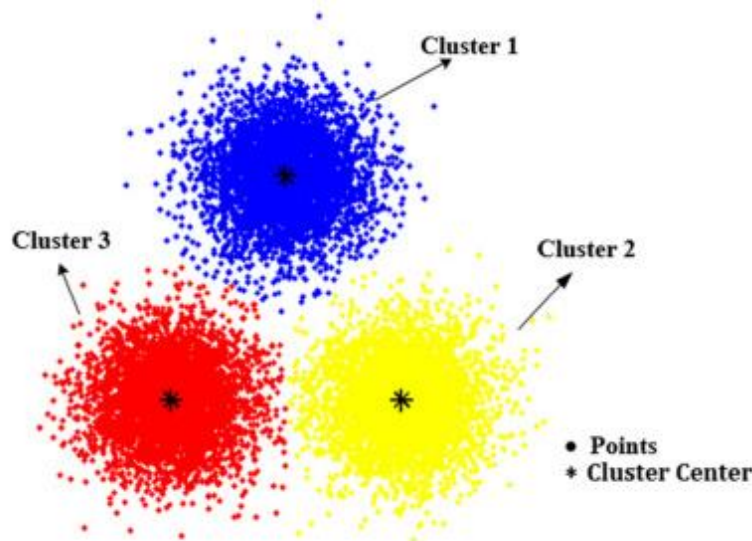
2

Ý tưởng thực hiện và mô tả chi tiết các hàm

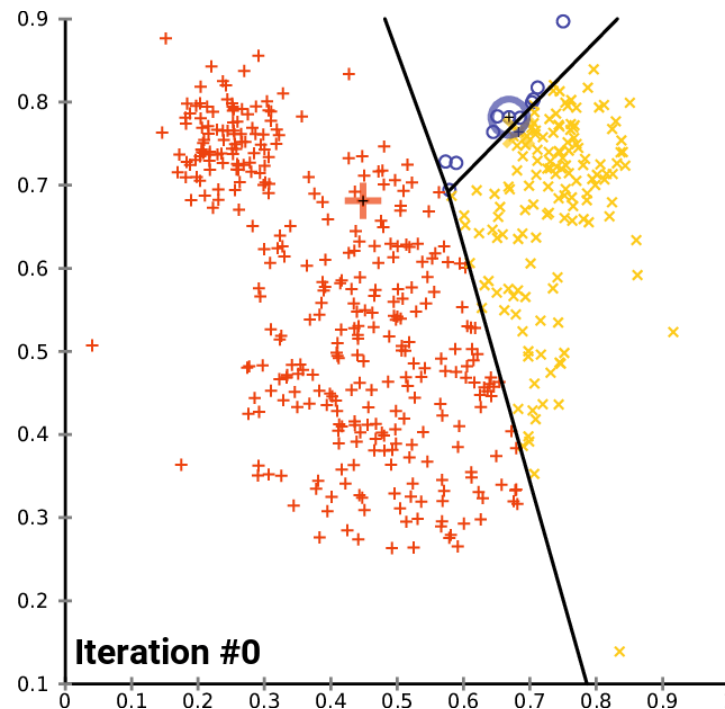
2.1

Ý tưởng thực hiện

- Bài toán yêu cầu việc gom các điểm ảnh tương đồng với nhau theo nhiều nhóm màu chủ đạo tương ứng.
- Từng điểm ảnh là một vector 3 chiều với 3 phần tử tương ứng với 3 màu gốc trong các mô hình ánh sáng bổ sung RGB (đỏ, xanh dương và xanh lá).



- Hay nói cách khác, để giải bài toán này, chúng ta sẽ thực hiện gom các vector tương đồng lại với nhau theo k cluster nhất định.
- Áp dụng thuật toán k-means sử dụng phương pháp tạo và cập nhật trung tâm để phân nhóm các điểm dữ liệu cho trước vào các nhóm khác nhau.



https://vi.wikipedia.org/wiki/Ph%C3%A2n_c%E1%BB%A5m_k-means

Phương pháp thực hiện như sau:

- Đầu tiên dữ liệu người dùng nhập vào là tên của bức ảnh, số lượng k cluster cần để xử lý ảnh, song chọn 1 trong 3 lựa chọn (.png .jpg người dùng nhập vào)
- Thực hiện tạo ra các điểm trung tâm ngẫu nhiên. Sau đó gán mỗi điểm trong tập dữ liệu vào trung tâm gần nó nhất. Mỗi điểm đó sẽ cập nhật lại trung tâm và tiếp tục lặp lại các bước đã kể trên.
- Tính toán và cập nhật lại các center mới cho mỗi cluster bằng cách dùng trung bình cộng của tất cả các điểm dữ liệu đã được gán vào cluster đó ở bước 2.
- Điều kiện dừng của thuật toán: Khi các trung tâm không thay đổi trong 2 vòng lặp kế tiếp nhau.



Ảnh ban đầu (171 KB)

Source: <https://www.pinterest.com/pin/302867143674250971/>



Ảnh sau khi giảm số lượng màu (163 KB)

2.2

Mô tả các hàm

Tên hàm	Mô tả
<pre>import matplotlib.pyplot as plt import numpy as np from PIL import Image</pre>	<p>Import các thư viện cần thiết trong đó:</p> <ul style="list-style-type: none"> - matplotlib.pyplot: hiển thị ảnh. - numpy: dùng cho tính toán trên ma trận. - PIL: đọc ảnh.
<pre>def euclid_distance(self, x1_y1, x2_y2):</pre>	<p>Tính khoảng cách các vector trên đường chéo ma trận (công thức Euclide).</p>
<pre>def determine_random_centroids(self):</pre>	<p>Xác định vị trí của điểm trung tâm của 1 khung màu</p>
<pre>determine_random_centroids.append((random_x, random_y))</pre>	<p>Thêm từng phần tử trong iterable vào list ban đầu với determine_random_centroids là list rỗng được khởi tạo trong hàm determine_random_centroids(self)</p>
<pre>random_x = np.random.rand() * self.img_1d.width random_y = np.random.rand() * self.img_1d.height</pre>	<p>Với giá trị ngẫu nhiên trong một hình dạng cho trước nhân với chiều dài (chiều cao) của các vector trong ảnh</p>
<pre>cluster_map = self.cluster_map for i in range(self.img_1d.width): for j in range(self.img_1d.height): distance = [] for m in range(self.k_clusters): in_pixels = self.img_1d_copy.getpixel(self.centroids[m]) point_pixel = self.img_1d_copy.getpixel((i,j)) euclid_dist = self.euclid_distance(in_pixels, point_pixel) distance.append(euclid_dist) index = distance.index(min(distance)) + 1 cluster_map[j][i] = index self.pixel_map[i,j] = self.img_1d_copy.getpixel(self.centroids[index - 1])</pre>	<p>Assign data points to clusters_Gán các điểm dữ liệu vào các cụm</p> <ol style="list-style-type: none"> 1. khai báo 1 ma trận (toàn là giá trị 1) 2. tạo biến không gian để xử lí list 2d <ol style="list-style-type: none"> a. khai báo 1 cái list rỗng b. trong khoảng k (cluster):

	<p>list những điểm trung tâm từ ảnh (vector) gốc sang ảnh kết quả (ảnh sau được xử lý giảm số lượng màu)</p> <p>list các điểm sang ảnh tạm</p> <p>khoảng cách từ mỗi điểm trung tâm đến các điểm dữ liệu trong ảnh</p> <p>thêm các giá trị khoảng cách vừa rồi vào cuối list rỗng vừa tạo</p> <p>c. kiểm tra các giá trị nhỏ nhất trong dist trên và thêm 1 giá trị cluster rỗng vì</p> <p>d. thực hiện gán các giá trị nhỏ nhất trong dist theo thứ tự vào cluster_map</p>
<pre>if (cluster_map == self.cluster_map).all(): self.convervation = True #mẫu chốt để kết thúc hàm self.cluster_map = cluster_map return cluster_map</pre>	<p>3. Check convergence: kiểm tra các điểm dữ liệu đến điểm trung tâm là gần nhất</p> <p>a. mẫu chốt để kết thúc hàm tạo các điểm trung tâm ứng với cluster, TRUE khi toàn bộ dữ liệu của ma trận cluster_map ở khởi tạo giống vs ma trận cluster</p> <p>4. Ngược lại thì gán ma trận cluster_map vào ma trận của hàm khởi tạo</p> <p>5. trả về giá trị cluster_map vừa xử lý xong</p>
<pre>def modify_new_map(self): ... def cluster(self): while self.convervation == False: self.modify_new_map() self.assign_data()</pre>	<p>Phân nhóm các điểm ảnh trong ma trận ảnh gốc.</p> <p>Xử lý để tính theo k cluster theo người dùng nhập vào. Cho đến khi điều kiện TRUE thì thoát hàm, khi đấy cho dừng thuật toán khi đạt được 1 kết</p>

	quả gần đúng và chấp nhận được.
<code>def __init__(self, img_path, k_clusters = 5):</code>	Tạo một hàm dựng khởi đầu cho class kmeans
<code>self.img_1d = PIL.Image.open(img_path)</code>	Thực hiện mở ảnh
<code>self.img_1d_copy = self.img_1d.copy()</code>	Tạo một ảnh tạm giống như ảnh gốc và trực tiếp xử lý độ phân giải trên ảnh tạm đó.
<code>self.pixel_map = self.img_1d.load()</code>	Để chuyển đổi từ một dictionary thành một JSON string.
<code>def choose_output_format(self, out):</code> <code>if out == 1:</code> <code>return str('.png')</code> <code>elif out == 2:</code> <code>return str('.jpg')</code>	Khi người dùng chọn tên đuôi file ảnh, thì hàm sẽ trả về tên đuôi file là .png hoặc .jpg tương ứng với giá trị 1 hoặc 2
<code>def main(self):</code>	Chương trình 'main' cho phép người dùng nhập vào tên tập tin ảnh mỗi lần chương trình thực thi
<code>image_path = input('Enter image file name: ')</code> <code>k_clusters = int(input("Enter number of clusters: "))</code>	Nhập tên file ảnh (VD: summer, picture, scene, family, ...) Nhập số lượng clusters k để xử lý ảnh
<code># Menu for users</code> <code>print("\ninput \"1\" for \".png\"")</code> <code>print("input \"2\" for \".jpg\"")</code> <code>print("input \"3\" for other types of image format (self-entered user)")</code>	Menu hướng dẫn người dùng
<code>out = int(input('Choose the output image saving format: '))</code> <code>if out == 1 or out == 2:</code> <code>image_path = image_path + str(self.choose_output_format(out))</code> <code>else:</code> <code>file_format = input('Enter the image saving format: ')</code> <code>image_path = image_path + str(file_format)</code>	Chọn (nhập) file định dạng nhập vào
<code>k_means = kmeans(image_path, k_clusters)</code> <code>k_means.cluster()</code>	Hàm xử lý ảnh bằng thuật toán K-Means
<code>print('\nThe image after being processed with Color Compression')</code> <code>plt.imshow(k_means.img_1d)</code>	Hàm in ra console hình ảnh sau khi được xử lý Color Compression

```
# Lưu ảnh ở định dạng tùy ý theo người dùng
print("\nsave image as \".png\" enter \"1\"")
print("save image as \".jpg\" enter \"2\"")
print("save image as \".pdf\" enter \"3\"")
choice = int(input('Choose the output image saving format: '))
if choice == 1:
    saved_output = '.png'
if choice == 2:
    saved_output = '.jpg'
if choice == 3:
    saved_output = '.pdf'

saved_img = str(input('Enter file image name: '))
saved_img = saved_img + saved_output

k_means.img_1d.save(saved_img)
print('\nOutput:\n', '-Image\'s name saved as ' + str(saved_img))
```

Chọn file định dạng để
lưu về máy (png, jpg hoặc
file pdf) và sau đấy lưu về
PC.

3

Demo kết quả chạy chương trình ứng với từng số lượng màu

- Đầu tiên thực hiện việc in ra màn hình file ảnh trong máy để đảm bảo được file ảnh có tồn tại và khi thực hiện Color Compression sẽ không ghi nhầm tên định dạng hoặc tên file.

```
Enter image file name (include image format): summer.png
```

```
<matplotlib.image.AxesImage at 0x24f11c46ee0>
```



Thực hiện với tên file là summer.png với image format là '1' ứng với .png và k cluster = 3, lưu ở định dạng file .jpg.

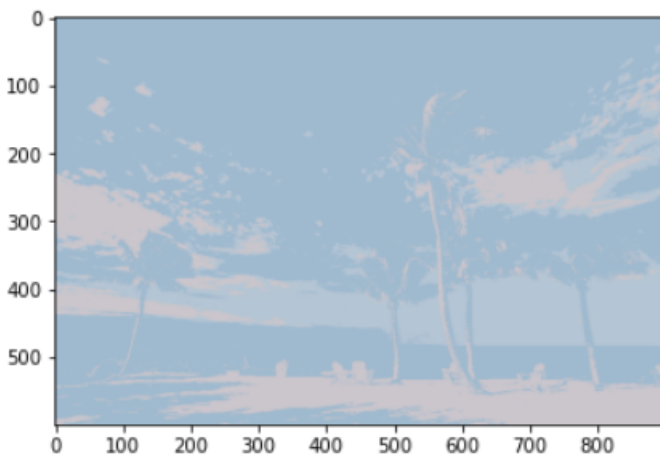
```
Enter image file name: summer
Enter number of clusters: 3
```

```
input "1" for ".png"
input "2" for ".jpg"
input "3" for other types of image format (self-entered user)
Choose the output image saving format: 1
```

```
Input:
-Image file name: summer.png
-Number of clusters: 3
```

The image after being processed with Color Compression

```
save image as ".png" enter "1"
save image as ".jpg" enter "2"
save image as ".pdf" enter "3"
Choose the output image saving format: 2
Enter file image name: summer
Image's name saved as summer.jpg
```



- Thực hiện với tên file là summer.png với image format là '1' ứng với .png và k cluster = 5, lưu ở định dạng file .png.

```
Enter image file name: summer
Enter number of clusters: 5
```

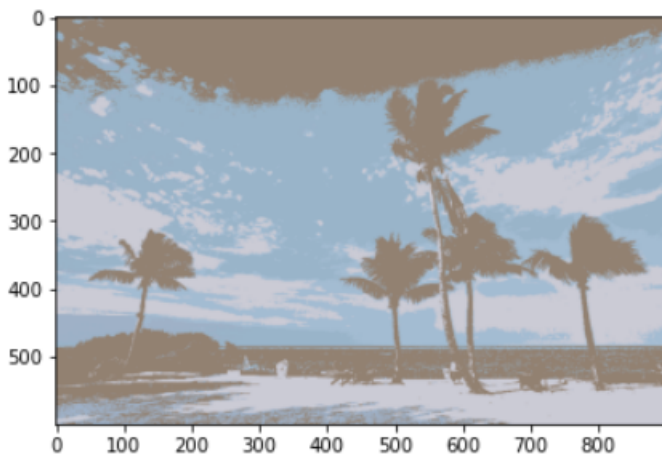
```
input "1" for ".png"
input "2" for ".jpg"
input "3" for other types of image format (self-entered user)
Choose the output image saving format: 1
```

Input:

```
-Image file name: summer.png
-Number of clusters: 5
```

The image after being processed with Color Compression

```
save image as ".png" enter "1"
save image as ".jpg" enter "2"
save image as ".pdf" enter "3"
Choose the output image saving format: 1
Enter file image name: picture
Image's name saved as picture.png
```



- Thực hiện với tên file là summer.png với image format là '1' ứng với .png và k cluster = 9, lưu ở định dạng file pdf.

```
Enter image file name: summer
Enter number of clusters: 9
```

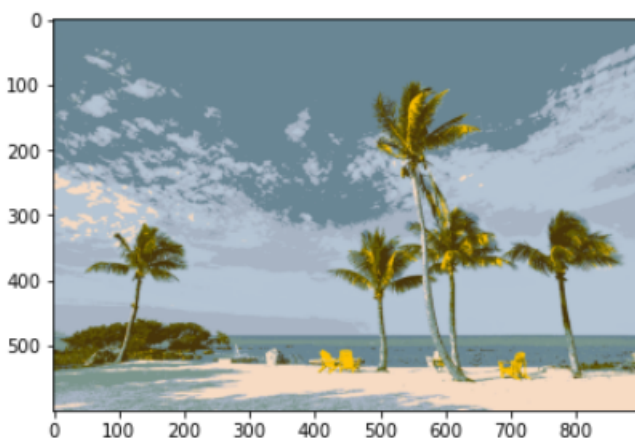
```
input "1" for ".png"
input "2" for ".jpg"
input "3" for other types of image format (self-entered user)
Choose the output image saving format: 1
```

```
Input:
-Image file name: summer.png
-Number of clusters: 9
```

The image after being processed with Color Compression

```
save image as ".png" enter "1"
save image as ".jpg" enter "2"
save image as ".pdf" enter "3"
Choose the output image saving format: 1
Enter file image name: summer9
```

```
Output:
-Image's name saved as summer9.png
```



Thực hiện với tên file là summer.png nhập vào file format và k cluster = 15, lưu ở định dạng file .png

```
Enter image file name: summer
Enter number of clusters: 15

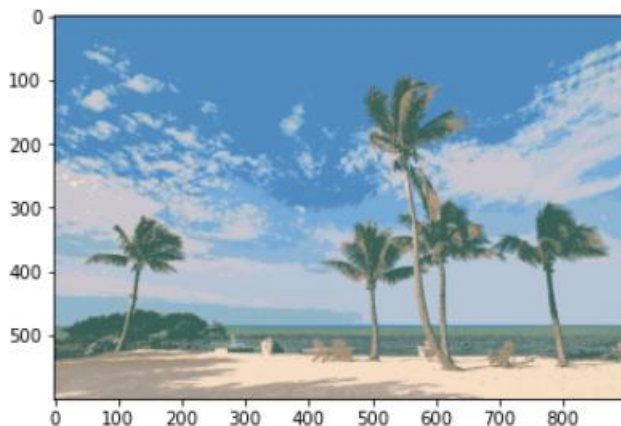
input "1" for ".png"
input "2" for ".jpg"
input "3" for other types of image format (self-entered user)
Choose the output image saving format: 3
Enter the image saving format: .png
```

Input:





- Image file name: summer.png
- Number of clusters: 15

The image after being processed with Color Compression

```
save image as ".png" enter "1"
save image as ".jpg" enter "2"
save image as ".pdf" enter "3"
Choose the output image saving format: 1
Enter file image name: picture
Image's name saved as picture.png
```



K – Clusters Ảnh sau khi nén ứng với k – clusters

3			
5			
9			
15			

4

Nhận xét kết quả demo chạy chương trình

4.1

Đánh giá kết quả

- Đánh giá mức độ hoàn thành của đồ án : 100%

4.2

Nhận xét về đồ án Color Compression

- Nếu k clusters càng lớn thì ta thấy màu sắc càng gần với màu sắc của bức ảnh ban đầu hơn do thuật toán K - means đã phân loại nhiều cluster màu hơn, còn nếu k clusters càng nhỏ thì màu sắc của nó càng xa so với màu sắc thực tế hơn.
- Và để thực hiện thuật toán K – means, điều hạn chế là người dùng cần biết trước số lượng k clusters để nhập vào, vì trên thực tế, việc xác định dữ liệu này đôi khi không khả thi.
- Ứng với từng đại lượng k clusters khác nhau, ta sẽ có được bức ảnh “compressed” khác nhau như:
 - ứng với $k = 3$, thuật toán đã phân loại bức ảnh thành 3 vùng miền màu khác nhau. Vì chỉ với 3 miền màu so với bức tranh gần trăm hoặc cả nghìn miền màu khác nhau nên ảnh khá mờ nhạt và một số đặc điểm trong ảnh khiến người dùng khó có thể phân biệt được.

- với $k = 5$, thuật toán đã có thể phân loại bức ảnh thành 5 vùng miền màu khác nhau. Do đó lúc này, bức ảnh sau khi được xử lý đã rõ nét hơn và người dùng đã có thể phân biệt được một số đặc điểm trong ảnh.
- Bên cạnh đó, ta cũng không thể không nhắc tới hàm khởi tạo giá trị centroids một cách random. Vì vậy, khi chạy chương trình với cùng 1 bức ảnh, cùng 1 k clusters nhưng khi chạy lần thứ hai, so với lần thứ nhất chạy, thì 2 bức ảnh lại có miền màu khác nhau. So với lần chạy chương trình đầu tiên thì lần chạy chương trình thứ hai có độ chính xác cao hơn, tính toán nhanh hơn và cho ra kết quả nếu giá trị miền màu random gần hơn so với các giá trị miền màu của bức ảnh.

5

Tài liệu tham khảo

<https://opg.optica.org/oe/fulltext.cfm?uri=oe-25-22-27570&id=375887>

https://vi.wikipedia.org/wiki/Ph%C3%A2n_c%E1%BB%A5m_k-means

[Machine Learning cơ bản \(machinelearningcoban.com\)](http://Machine Learning cơ bản (machinelearningcoban.com))

<https://nguyenvanhieu.vn/thuat-toan-phan-cum-k-means/#:~:text=Thu%E1%BA%ADt%20to%C3%A1n%20ph%C3%A2n%20c%E1%BB%A5m%20k%2Dmeans%20l%C3%A0%20m%E1%BB%99t%20ph%C6%B0%C6%A1ng%20ph%C3%A1p,s%E1%BB%AD%20thu%E1%BB%99c%20v%E1%BB%81%20nh%C3%B3m%20n%C3%A0o.>

<https://machinelearningcoban.com/2017/01/04/kmeans2/>