HO CHI MINH CITY NATIONAL UNIVERSITY
UNIVERSITY OF NATURE SCIENCE
FACULTY OF INFORMARTION TECHNOLOGY
---------------o0o---------------

PROJECT 1:

COLOR COMPRESSION

Lecturers:

Vũ Quốc Hoàng

Phan Thị Phương Uyên

Nguyễn Văn Quang Huy

Students:

Đặng Ngọc Tiến ID: 20127641

Contents

I. Introduction 1

1. Personal information: 1

II. Implementation idea and description 1

1. Implementation idea 1

2. Description 1

III. Demo and Comment results 3

IV. References 4

I. Introduction

1. Personal information:

Name: Đặng Ngọc Tiến

ID student: 20127641

II. Implementation idea and description

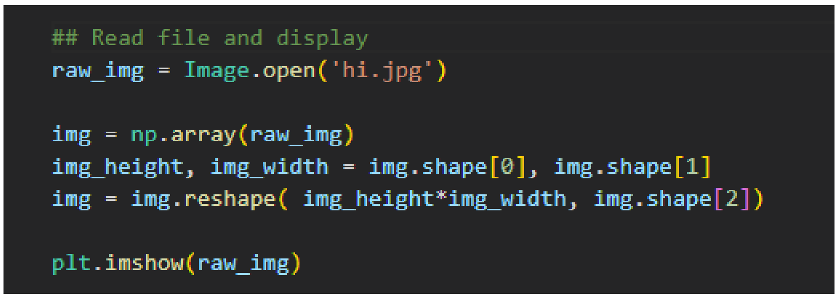
1. Implementation idea

A recipe for k-means

- Decide how many clusters you want, i.e., choose k
- Randomly assign a centroid to each of the k clusters
- Calculate the distance of all observation to each of the k centroids
- Assign observations to the closest centroid
- Find the new location of the centroid by taking the mean of all the observations in each cluster
- Repeat steps 3-5 until the centroids do not change position

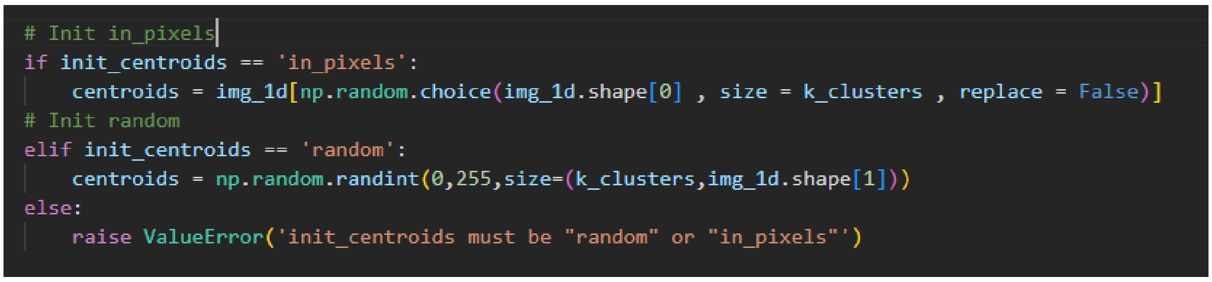
2. Description

- Use the PIL library to read the image, then reshape the image from a three-dimensional scalar into a 2-dimensional array of the form [height * width, n], where = 3

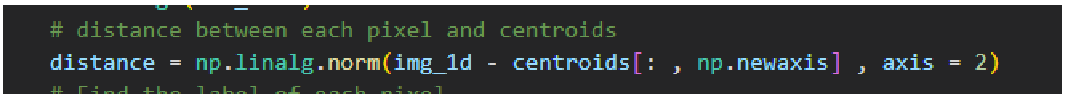


```
## Read file and display
raw_img = Image.open('hi.jpg')

img = np.array(raw_img)
img_height, img_width = img.shape[0], img.shape[1]
img = img.reshape( img_height*img_width, img.shape[2])

plt.imshow(raw_img)
```

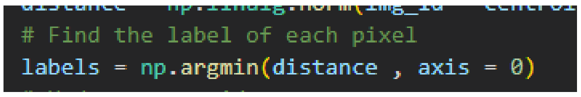
- For each parameter passed in init centroids is random in the range [0.255] or random pick from the color of the image, we perform initialization for centroids as k color points with k being the number of clusters.


```python
# Init in_pixels
if init_centroids == 'in_pixels':
    centroids = img_1d[np.random.choice(img_1d.shape[0] , size = k_clusters , replace = False)]
# Init random
elif init_centroids == 'random':
    centroids = np.random.randint(0,255,size=(k_clusters,img_1d.shape[1]))
else:
    raise ValueError('init_centroids must be "random" or "in_pixels"')
```

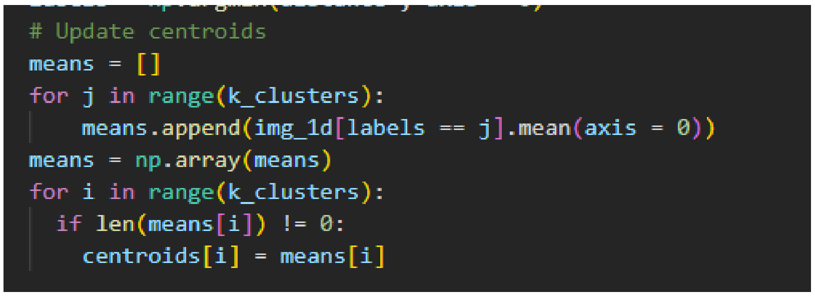
- Calculate the distance using np.linalg.norm in Numpy library


```python
# distance between each pixel and centroids
distance = np.linalg.norm(img_1d - centroids[: , np.newaxis] , axis = 2)
```

- Assign observations to the closest centroid using np.argmin in Numpy library


```python
# Find the label of each pixel
labels = np.argmin(distance , axis = 0)
```

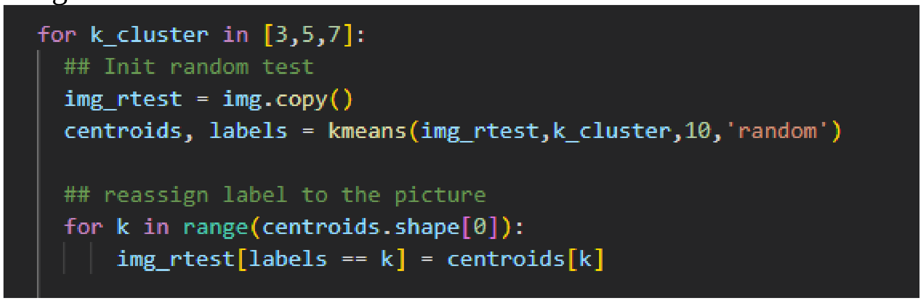
- Update centroid location.


```python
# Update centroids
means = []
for j in range(k_clusters):
    means.append(img_1d[labels == j].mean(axis = 0))
means = np.array(means)
for i in range(k_clusters):
  if len(means[i]) != 0:
    centroids[i] = means[i]
```

- And repeat by max_iter
- Return centroids and labels

Test program

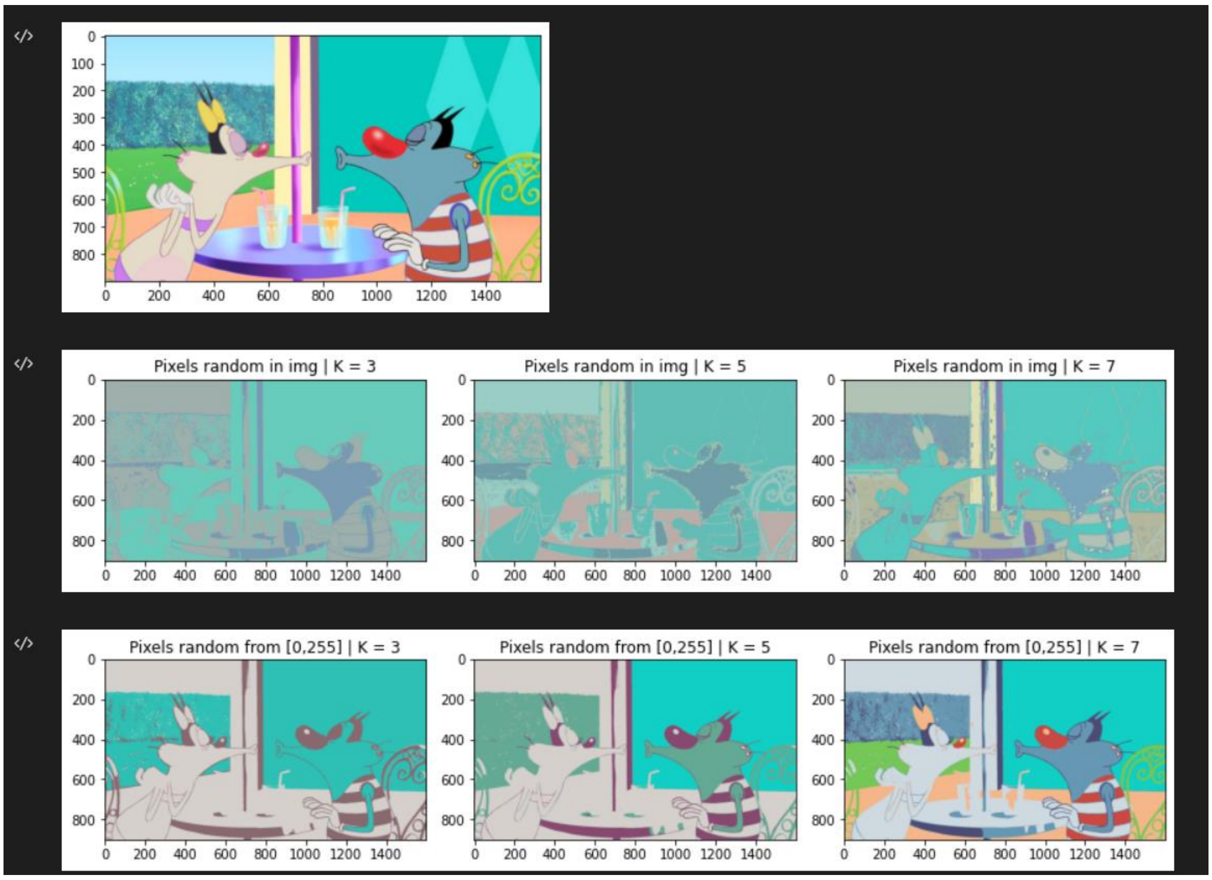
- After returning labels and centroids in kmean then we assign them back to the image


```python
for k_cluster in [3,5,7]:
  ## Init random test
  img_rtest = img.copy()
  centroids, labels = kmeans(img_rtest,k_cluster,10,'random')

  ## reassign label to the picture
  for k in range(centroids.shape[0]):
      img_rtest[labels == k] = centroids[k]
```

- And using matplotlib library to display img

III. Demo and Comment results

- **Test program:**


Pixels random in img | K = 3
Pixels random in img | K = 5
Pixels random in img | K = 7
Pixels random from [0,255] | K = 3
Pixels random from [0,255] | K = 5
Pixels random from [0,255] | K = 7

- **Comments:**
  - In general, the demo results give quite good results, if compared with sciket-learn's Kmean, we have approximately the same efficiency. However, the image compression performance for small images (compressed - reduced color by other software) or monochrome images often produces bad results with low max_iter.
  - In addition, if we plot the convergence of Kmean, when we init 'random' the convergence speed will be faster than the init taken from the point on the image in some cases. In case the image has many colors, taking random gives better performance.

IV. References

- Idea Kmean algorithm:

  K-means Clustering in Python: A Step-by-Step Guide (dominodatalab.com)

- Library:

  NumPy Documentation

  Matplotlib documentation — Matplotlib 3.5.2 documentation

  Pillow — Pillow (PIL Fork) 9.1.1 documentation