Try to code the assignment by yourself. Plagiarism is not tolerated

Assignment 6
Color Image Processing and Segmentation

Problem Statement

In this assignment, you have to implement a K-means algorithm for color image segmentation. Read the instructions for each step. Use python with the **numpy**, **random** and **imageio** libraries.

Your program must have the following steps:

1. **Parameters input:**
   a. Filename for the input image ($\mathbf{I}$)
   b. Filename for the reference image ($\mathbf{R}$)
   c. Option for pixel attributes:
      1 - (R, G, B)
      2 - (R, G, B, x, y)
      3 - (luminance)
      4 - (luminance, x, y)
   d. Number of clusters ($\mathbf{k}$)
   e. Number of iterations ($\mathbf{n}$)
   f. Seed ($\mathbf{S}$) to be used for the random centroids choice
2. **Generate an output image** ($\hat{\mathbf{I}}$) according to the option for feature extraction.
3. **Compare the output image** ($\hat{\mathbf{I}}$) **with the original one** ($\mathbf{R}$).

K-means algorithm

K-means is a clustering algorithm based on the concept of similarity. The main idea is to group similar items according to their attributes. Each cluster will be a labeled region on the image, and it is represented by a centroid, which is a point in the attribute space that is calculated by computing the mean of all points in the cluster.

The main implementation **STEPS** are represented below:

1. **Input:**
   a. **k** number of clusters
   b. Dataset **D** with **r** attributes (in this context, objects are pixels and their color and coordinates are attributes)
   c. **n** number of iterations
2. Initialize the **k** cluster centroids by selecting **k** examples from **D**
3. Repeat until **n**
   a. Assign each example (pixel) to the cluster relative to the centroid with the smallest distance to the pixel.
   b. Update the clusters by re-calculating the centroids, that is, the average vector considering all objects in each cluster. Note that the centroid does not necessarily coincide with an object of the cluster.
   c. Return a set that indicates the cluster of each object.

Additional instructions and observations:

1. The parameter **k** refers to the total number of clusters that will be discovered considering all image pixels. This is also the number of distinct labeled regions in the output image; You should use:
   a. random.seed(S)
   b. ids = np.sort(random.sample(range(0, m*n), k))
      i. to generate an index set that determines the position of the initial centroids in the dataset;
2. The parameter **n** refers to the number of internal iterations of K-means.
3. Use Euclidean distance for step **3.a** of K-means.

Attributes:

For a $\mathbf{I}_{m,n,3}$ RGB input image, in which, for example **I** (x, y, c) represents the pixel at coordinate x, y and color channel $C \in [0, 1, 2]$ associated respectively to R, G and B. There are four options to be considered as attribute spaces to perform the clustering-based segmentation. The first two use the actual RGB values, while the remaining ones use a linear combination of the color channels:

- **R, G, B:** a 3D array
  $\rightarrow [I(x,y,0),\ I(x,y,1),\ I(x,y,2)]$
- **R, G, B, x, y:** a 5D array
  $\rightarrow [I(x,y,0),\ I(x,y,1),\ I(x,y,2),\ x,\ y]$
- **Luminance:** a 1D array
  $\rightarrow [0.299 \cdot I(x,y,0) + 0.587 \cdot I(x,y,1) + 0.114 \cdot I(x,y,2)]$
- **Luminance, x, y:** a 3D array
  $\rightarrow [0.299 \cdot I(x,y,0) + 0.587 \cdot I(x,y,1) + 0.114 \cdot I(x,y,2),\ x,\ y]$

Note that Luminance is an RGB-to-grayscale transformation that combines the three color matrices into a single matrix of luminance levels.

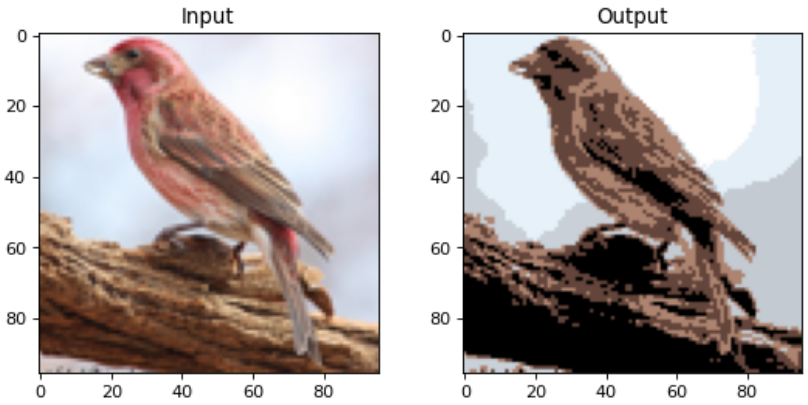
Segmented Image:

The segmented image $\mathbf{f}$ is a $m\,x\,n$ image with labels defined by $f\,(x,y) \in [1,\,k]$, that is, each pixel receives a label relative to the cluster to which it belongs.

After obtaining the clustered image $\mathbf{f}$, you must generate the output image $\hat{\mathbf{I}}$ by using the centroids of each cluster as the pixel information. Whether the function is 1 or 2, you must create $\hat{\mathbf{I}}$ as a 3D array, that is, an RGB image. However, if the function is 3 or 4, you must create $\hat{\mathbf{I}}$ as a 1D array, that is, a grayscale image where each pixel is the luminance value.

It will be obtained as a result image composed of $\mathbf{k}$ variations of pixel intensities, which were computed by the final centroids of each cluster (see Fig. 1).

Figure 1. Example of a segmented image using K-means.



Comparison with original image

After generating the output image $\hat{\mathbf{I}}$, compare it with the reference image $\mathbf{R}$ using RMSE. Since $\mathbf{R}$ has values between 0 and 255, you must normalize $\hat{\mathbf{I}}$ so that it also has values in the same range in the **uint8** format.

When the images $\mathbf{R}$ and $\hat{\mathbf{I}}$ are composed only by 1 channel (luminance), you can compute the RMSE as follows:

$$\text{RMSE} = \sqrt{\frac{1}{MN}\Sigma\Sigma(R(i,j) - \hat{I}\,(i,j))^2}$$

Where M x N is the size of the image.

However, when the images $\mathbf{R}$ and $\hat{\mathbf{I}}$ are composed of 3 channels (R, G, B), you must compute the RMSE for each channel separately (as presented in the equation above) and calculate the average among them later. For example:

$$RMSE_{FINAL} = (RMSE_{RED} + RMSE_{GREEN} + RMSE_{BLUE}) / 3$$

Input and Output

Example of input:
Input image ($\mathbf{I}$), Reference image ($\mathbf{R}$), choice of function (1), number of clusters (5), total number of iterations (10) and seed (42).

Input	image_1.png image_1_ref1.png 1 5 10 42

Example of output:
RMSE value in float format with 4 decimal places.

Output	0.4873

Submission

Submit your source code using the Run.Codes (only the .py file)

1. **Comment your code.** Use a header with name, USP number, course code, year/semester and the title of the assignment. A penalty on the grading will be applied if your code is missing the header and comments.
2. **Organize your code in programming functions.** Use one function per method.

Contact

If you have any questions, contact us by sending an email following the five steps below:

1st step: Include **BOTH** emails, sherlon@usp.br **and** messias@ifsc.usp.br.
2nd step: Include the subject **exactly** like this:
Subject: "**[Digital Image Processing 2022 | sem1] - Assignment 1**"
Do not change the initial part (**black**).
Replace the final part with the topic you are interested in (**red**).
3rd step: Add your personal information to help us find your submissions in Run.Codes and E-Disciplinas quickly.
4th step: Formulate your question in detail. Include your implementation and/or screenshots if necessary.
5th step: Send email and wait. We will respond as soon as possible.

Example of Email: