scc0251/5830— Prof. Moacir Ponti
Teaching Assistant: Aline Becher

Assignment 4

Try to code the assignment by yourself. Plagiarism will not be tolerated

Colour image processing and segmentation

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 I;
   b) Filename for the reference image 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 k:
   e) Number of iterations: n
   f) Seed S to be used for the random centroids choice.
2. **Generate an output image ($\hat{I}$)** according to the option for feature extraction.
3. **Compare the output image $\hat{I}$ with reference image R.**

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 is going to be a labelled 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.

K-means algorithm

1: **Input:** k: Number of clusters;
2: D: dataset with r attributes (in our case, objects are pixels and their colour and coordinates attributes);
3: n: Number of iterations;
4: **do:**
5: Initialize the k cluster centroids by selecting k examples from D;

6: **repeat** until n
7: Assign each example (pixel) to the cluster relative to the centroid with the smallest distance to the pixel.
8: 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 coincides with an object of the cluster.
9: **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 are going to be discovered considering all image pixels. This is also the number of distinct labelled regions in the output image; You should use
   random.seed(S)
   ids = np.sort(random.sample(range(0, m*n), k))
   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 6 of K-means.

Attributes: For a $I_{m,n,3}$ RGB input image, in which, for example $I(x, y, c)$ represents the pixel at coordinate x, y and colour 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 a linear combination of the colour channels:

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

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

Segmented image: segmented image $\hat{I}$ is a $m \times n$ image with labels defined by $\hat{I}(x, y) \in \{1, \ldots k\}$, that is, each pixel receives a label relative to the cluster that it belongs to.

Comparison with original image

After processing the image and obtaining $\hat{I}$, it should be normalized between 0 and 255 so that it also has values in the same range in the *uint8* format.
Your program must compare the generated image $\hat{I}$ with a reference image R. This comparison must use the root mean squared error (RMSE). Print this error in the screen,rounding to 4 decimal places.

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

where $M \times N$ is the size of the image.

Attention: To calculate the RMSE between 2 images `G` and `H`, use `G.astype(float)` and `H.astype(float)` to avoid overflow.

Input and output

Example of input:
image.png
image-ref.npy
2
5
10
55

Example of output: RMSE value in float format with 4 decimal places
2.8527

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/semestre 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.