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

Assignment 5
Morphology and Image Description

Problem Statement

In this assignment, you must implement an information retrieval system based on mathematical morphology and image description. Read the instructions for each step. Use python with the **numpy** and **imageio** libraries.

Your program must read the following parameters:
Parameters input:

- **Index** of a query image ∈ $[0, B - 1]$
- **Q value** for GLCM
  - Reference pixel is $[x, y] = [0, 0]$
  - Neighbor pixel is:
    i. 2D coordinate $[x, y]$ for the 8-neighborhood
    ii. Where, $x \subset \{-1, 0, 1\}$ and $y \subset \{-1, 0, 1\}$
- **Parameter F:** (1-Opening, 2-Closing)
- **Parameter T:** threshold for image binarization
- **B** number, representing the total number of images in the dataset
- The **B image names** to be read

Part 1 - Morphological Image Processing

Morphological processing can enhance or mitigate patterns related to relationships among neighbor pixels in images. In this assignment, you will implement these techniques to pre-process a dataset of images before applying texture analysis.

Step-by-Step:

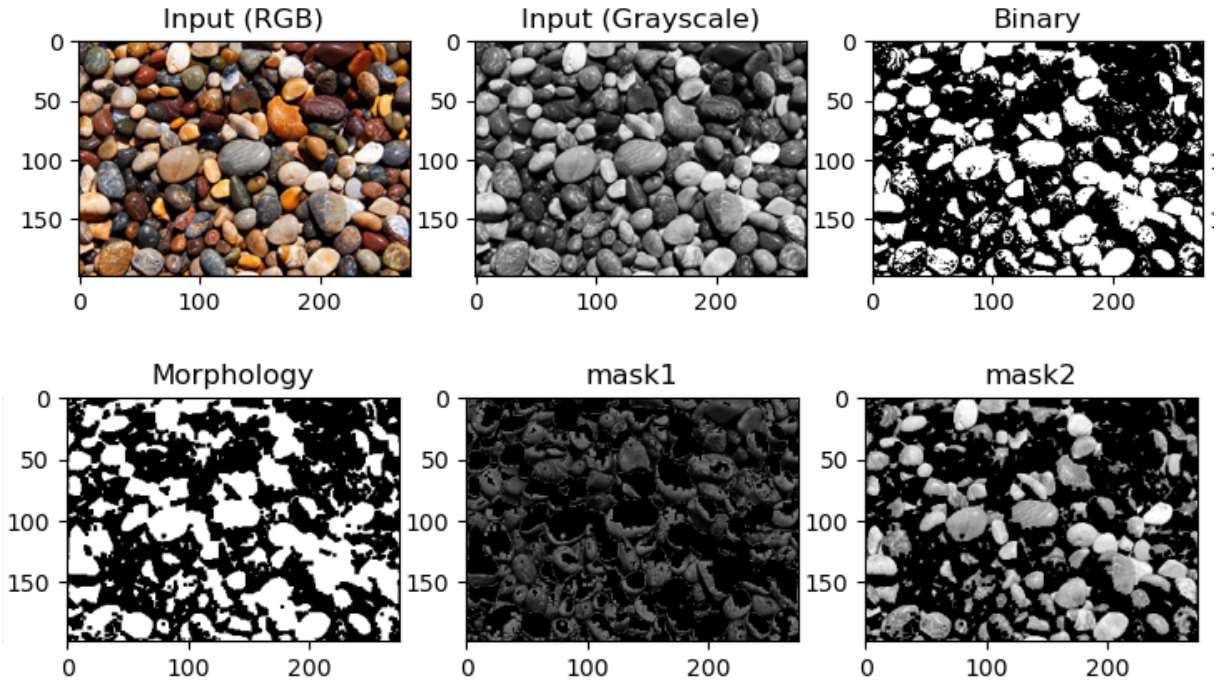
- **Generate a grayscale image (Igray)** using the Luminance weights.
- **Binarize the grayscale image** according to the T value.
  i. img[img < T] = 0
  ii. img[img >= T] = 1
- **Apply the morphology function** defined by F to obtain an image **(Imorph)**
- **Obtain 2 segmented masks** from the binary images.
  - mask1 is composed by the **Igray** pixels where the **Imorph** is 0
  - mask2 is composed by the **Igray** pixels where the **Imorph** is 1

Given B colored images as input, you need to convert them to grayscale using the Luminance weights:

$$GrayImage_{MxN} = ((0.299*Red) + (0.587*Green) + (0.114*Blue))$$

After that, you must perform a limiarization using the given threshold T and perform the morphological processing required. To do so, you need to implement the **Erosion** and **Dilation** algorithms and perform the **Opening** (Erosion + Dilation) whether F = 1, or perform **Closing** (Dilation + Erosion) whether F = 2. Now you must create two new images called **mask1** and **mask2** from the **Imorph** image, that is, **mask1** is composed by the **Igray** pixels where the **Imorph** is 0 while **mask2** is composed by the **Igray** pixels where the **Imorph** is 1. Figure 1 below presents an example of these steps applied to process the images.

Figure 1. Example.



Obs.: you can consider a filter 3x3 to perform the morphological manipulations.
Obs.: you can create an image MxN initialized with zeros for the masks and replace only the pixels given by the limiarization.

Part 2 - Information Retrieval

Understanding data relationships based on their intrinsic patterns is one of the main tasks of a great data scientist. Different types of data require different approaches to capture patterns. However, the goal is the same: use the intrinsic patterns explored to find useful information. In this task, you are going to

implement GLCM with Haralick descriptors in order to understand texture patterns in images and observe their relationships by performing queries by similarity.

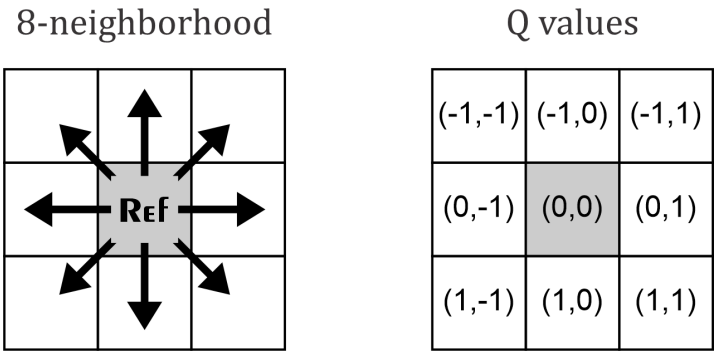
Step-by-Step:

- **Feature extraction**
  For all images in the dataset:
  - Create a co-occurrence matrix with probabilities for image **mask1**
  - Create a co-occurrence matrix with probabilities for image **mask2**
  - Compute Haralick descriptors for image **mask1**
  - Compute Haralick descriptors for image **mask2**
  - Concatenate the **mask1_descriptors** with **mask2_descriptors**
- **Performing queries**
  i. Compute the similarity between the query image and all the dataset.
  ii. Show a similarity ranking with the B most similar images.

Once we have all the images in the dataset pre-processed, we can start to extract features in order to identify texture patterns. Your first task is to compute the Co-Occurrence matrix, which is necessary to calculate Haralick descriptors. To generate the probabilities matrix, you need to normalize the Co-Occurrence matrix in which its sum must be equal to 1.

The Q value is a coordinate of the neighbor pixel in the 8-neighborhood of the reference pixel [0,0], as Figure 2 shows.

Figure 2. Neighborhood to consider in the Co-Occurrence matrix analysis.



Obs.: compute the Co-Occurrence matrix only for the pixels in the range [1, M-1] and [1, N-1], where the MxN is the size of the image. This process is required only to facilitate your implementation, to dismiss the necessity of considering different iterations depending on the Q value. Consider an initial matrix initialized with zeros of size $\mathbf{(Max+1)^2}$, defined as follows: **np.zeros([Max+1, Max+1])** , where **Max** is the **np.max(MaskN)**.

Once you have the probabilities matrices (for **mask1** and **mask2**), we will now calculate the descriptors as an array of 8 dimensions.

Below are the Haralick descriptors you must calculate:

Auto correlation	$\sum_{i=0}^{M}\sum_{j=0}^{N}(i \cdot j)\,p(i,j)$	Entropy	$-\sum_{i=0}^{M}\sum_{j=0}^{N}p(i,j)\,log(p(i,j))$ Consider only probabilities > 0
Contrast	$\sum_{i=0}^{M}\sum_{j=0}^{N}(i-j)^2\,p(i,j)$	Homogeneity	$\sum_{i=0}^{M}\sum_{j=0}^{N}\frac{p(i,j)}{1+(i-j)^2}$
Dissimilarity	$\sum_{i=0}^{M}\sum_{j=0}^{N}\lvert i-j\rvert\,p(i,j)$	Inverse difference	$\sum_{i=0}^{M}\sum_{j=0}^{N}\frac{p(i,j)}{1+\lvert i-j\rvert}$
Energy	$\sum_{i=0}^{M}\sum_{j=0}^{N}p(i,j)^2$	Maximum probability	$max(p(i,j))$

OBS.: Here, M is equal to N, once it is the size of the probability matrix (Max+1, Max+1).

At this moment, your implementation is supposed to have one 16D array for <u>each</u> image in the dataset as follows.

mask1_descriptor = [auto_correlation, contrast, dissimilarity, energy, entropy, homogeneity, inverse_difference, maximum_probability]
mask2_descriptor = [auto_correlation, contrast, dissimilarity, energy, entropy, homogeneity, inverse_difference, maximum_probability]

all_descriptors = np.concatenate((**mask1_descriptor** , **mask2_descriptor**), axis=None)

Compute the similarity between the **query image** and **all the images** in the dataset using the **euclidean similarity** to compare their **all_descriptors** array. Normalize the similarity matrix in the [0, 1] range. Print the B most similar images to the query formatted exactly like the output example.

Obs.: Remember that distance and similarity are reversed. While distance 1 between two images means they are totally different, similarity 1 between them means they are exactly the same. Thus, if you calculate using the euclidean distance, it is necessary to apply (1 - distances) to obtain the similarities.

Input and Output

Example of input:
Query index, Q [x,y], F, T, B, B images.

Input	0 0 1 1 128 3 pedras1.png grama1.png pedras2.png

Example of output:
Ranking presenting the B most similar images.

Output	Query: pedras1.png Ranking: (0) pedras1.png (1) pedras2.png (2) grama1.png

Once your result will be compared directly with our result, it is necessary to print the output exactly as the output example. Firstly, print "Query: " followed by the query image. In the next line you must print "Ranking:" followed by B lines containing the most similar images in descending order.

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: