UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTE OF COMPUTING

INTRODUCTION TO DIGITAL IMAGE PROCESSING
MC920 / MO443

HOMEWORK 4

STUDENT: Gian Franco Joel Condori Luna
RA: 234826

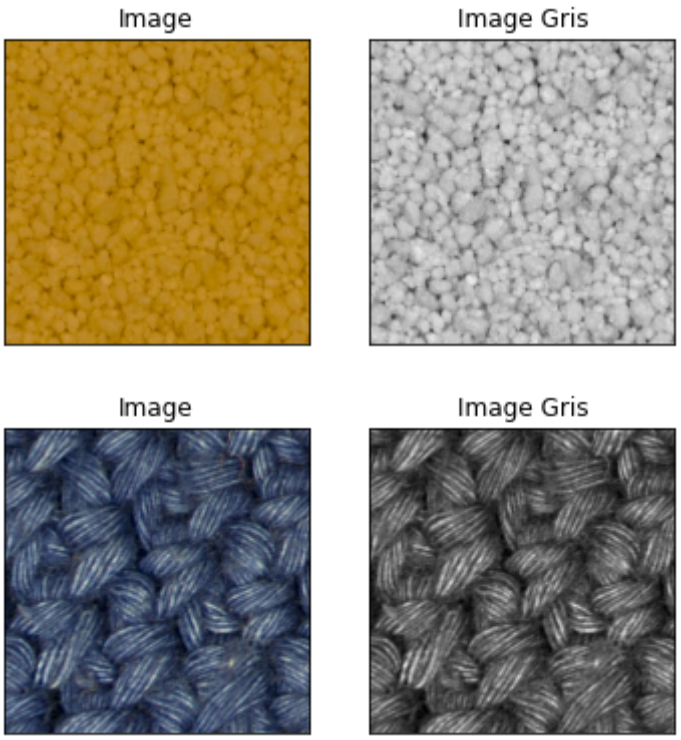
1.- Problem specification:

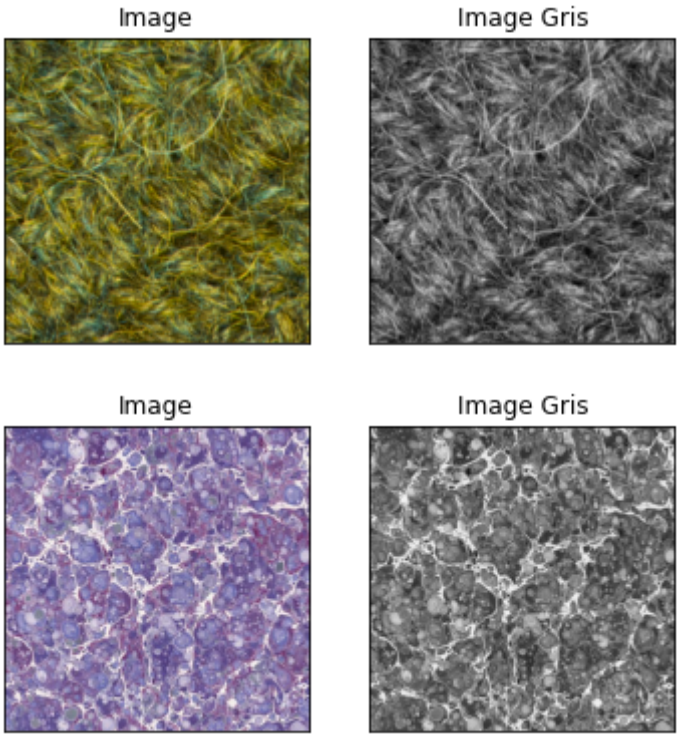
Texture is one of the main pieces of information to describe an image. The objective of this work is to analyze and compare texture images by means of local binary patterns (LBP) and co-occurrence matrices (GLCM). The main steps are described in the following sections.

1.1.- Color Transformation:

In order to convert the images to gray, the python library "skimage" was used.

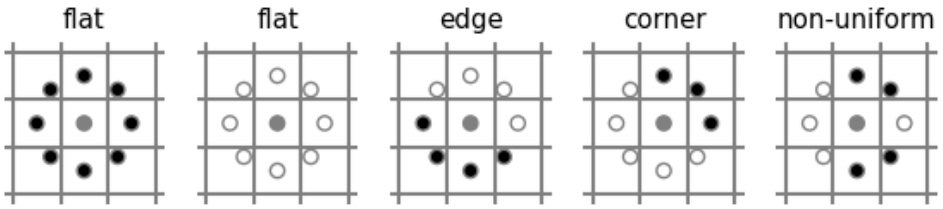
```
image1 = io.imread('/content/drive/My
Drive/imagenes/imagens_textura/textura1.png')
image1_gris = color.rgb2gray(image1)*255
```





1.2.- Local Binary Patterns

LBP looks at points surrounding a central point and tests whether the surrounding points are greater than or less than the central point (i.e. gives a binary result).[1]



The figure above shows example results with black (or white) representing pixels that are less (or more) intense than the central pixel. When surrounding pixels are all black or all white, then that image region is flat (i.e. featureless). Groups of continuous black or white pixels are considered “uniform” patterns that can be interpreted as corners or edges. If pixels switch back-and-forth between black and white pixels, the pattern is considered “non-uniform”.

When using LBP to detect texture, you measure a collection of LBPs over an image patch and look at the distribution of these LBPs. Lets apply LBP to a brick texture.

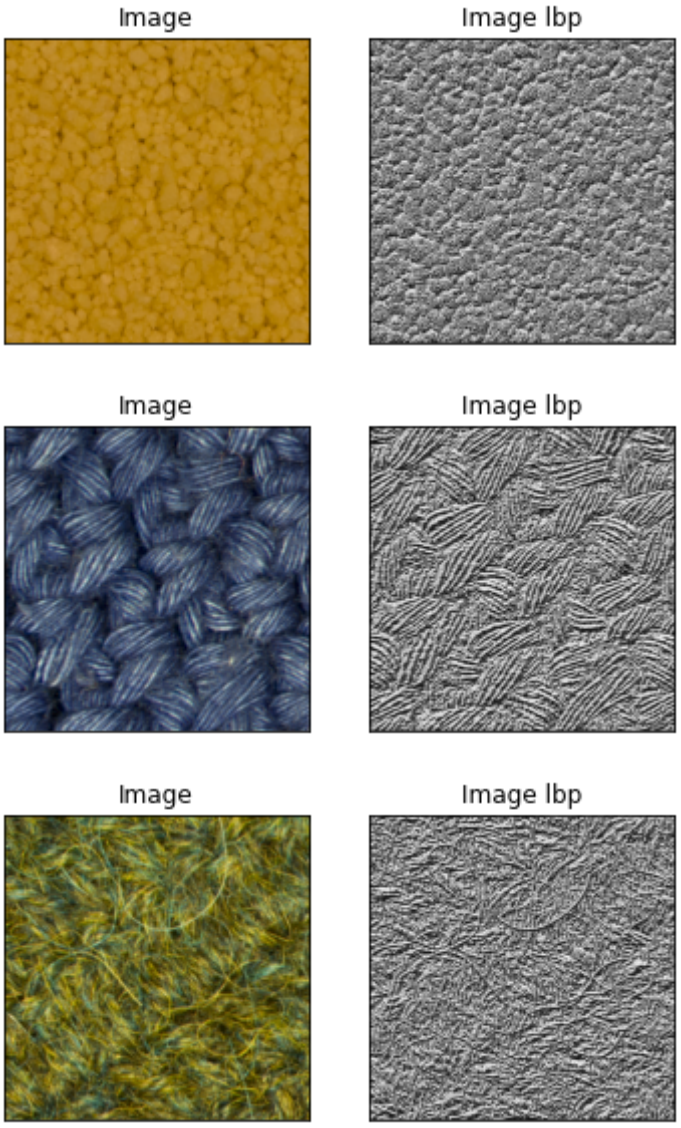
We use the function “local_binary_pattern” from the python library "skimage":

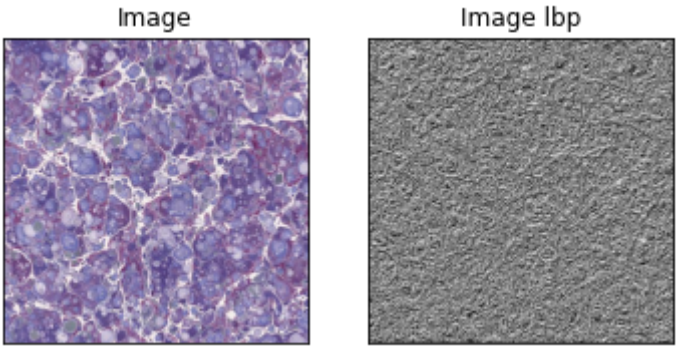
```
local_binary_pattern(double[:, ::1] image,int P, float R, char method)
```

Parameters

- image : (N, M) double array Graylevel image.
- P : Number of circularly symmetric neighbour set points (quantization of the angular space).
- R : Radius of circle (spatial resolution of the operator).
- method : {'D', 'R', 'U', 'N', 'V'}. Method to determine the pattern.
  * 'D': 'default'
  * 'R': 'ror'
  * 'U': 'uniform'
  * 'N': 'nri_uniform'
  * 'V': 'var'

```
lbp1 = local_binary_pattern(image1_gris, 8, 1)
```

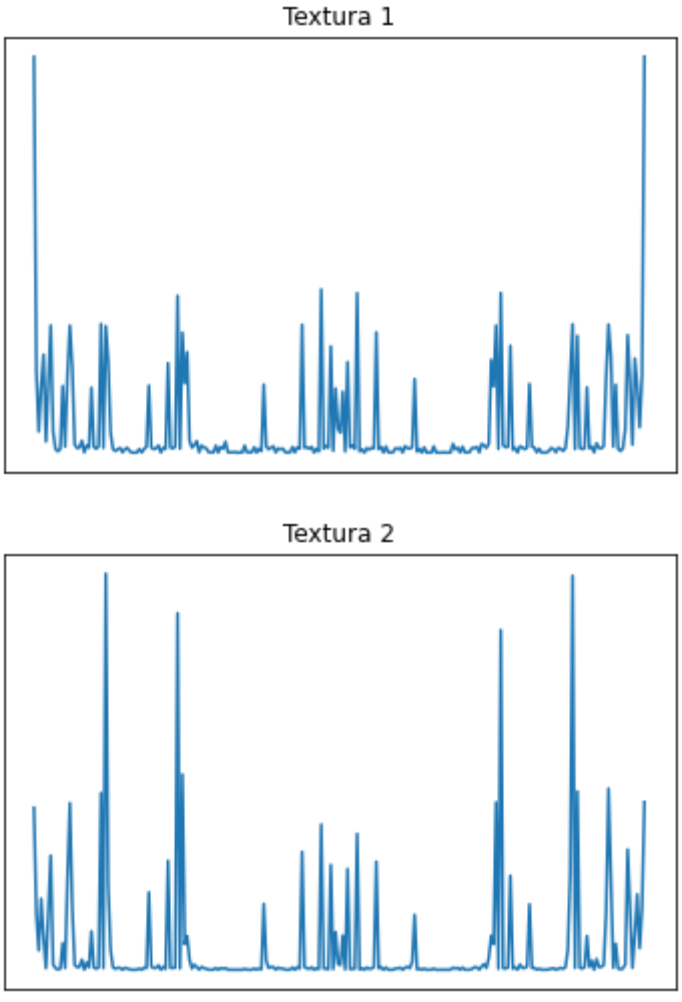


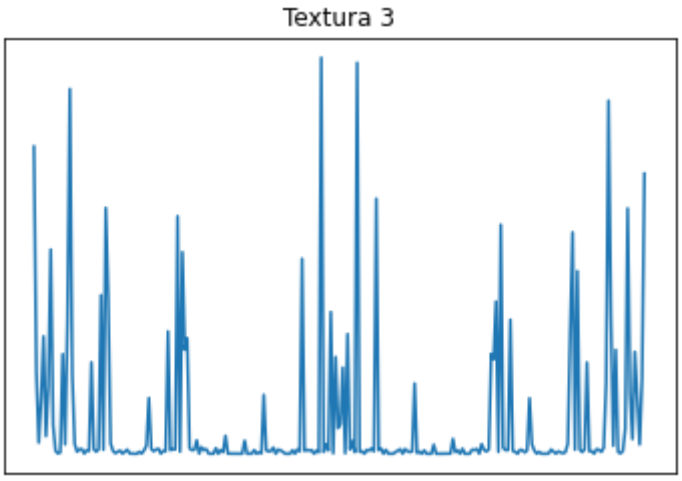


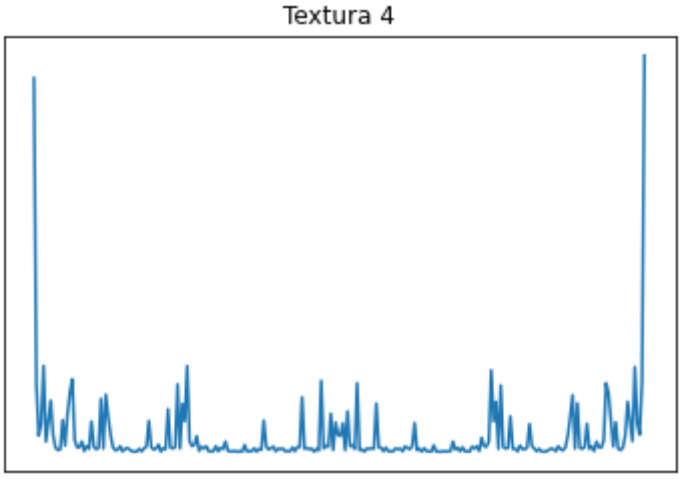
1.3.- Comparison between texture images:

a) **Histograms:**
we use the "histogram" function of numpy. [2]

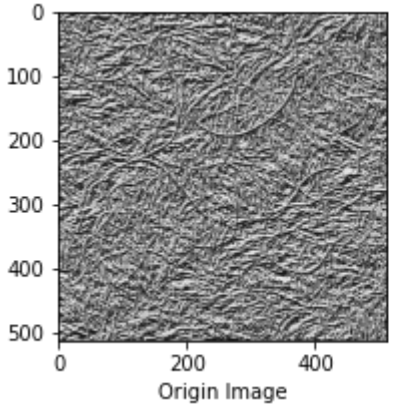
```
histogram, points = np.histogram(lbp4, bins=np.arange(2**8 + 1),
density=True)
```

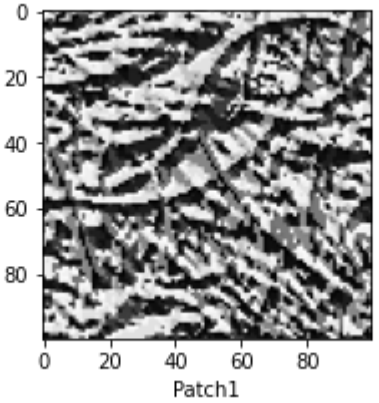


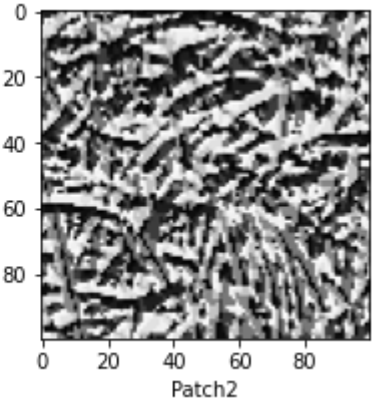


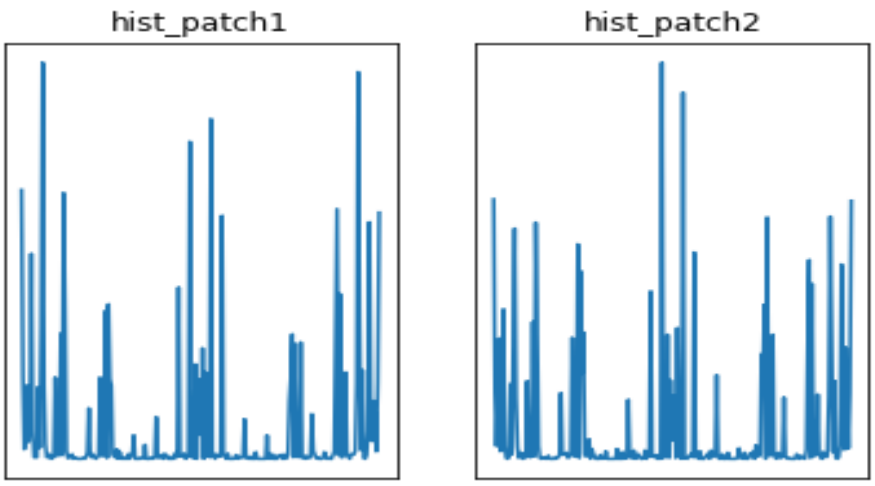


In order to make a comparison we will obtain two parts of the same image (patch1 and patch2), in this case of “textura3.png”(origin image)









- **Euclidean distance:**

```
from scipy.spatial.distance import euclidean
euclidean(hist_patch1, hist_patch2)

#RPTA: 0.04611897657147219
```

- **Bhattacharyya**

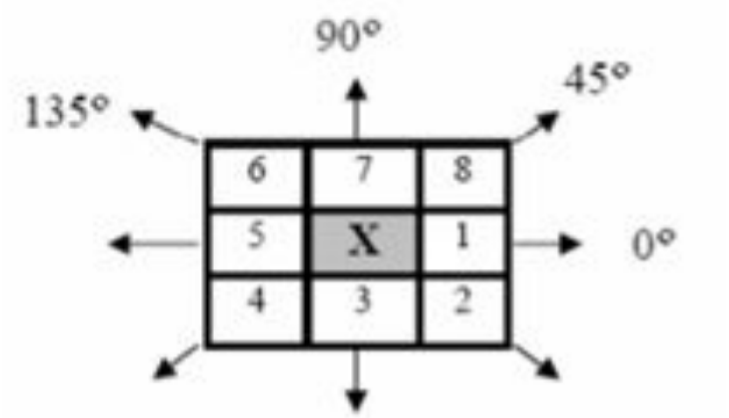
```
from dictances import bhattacharyya
bhattacharyya(dict(enumerate(hist_patch1)),
dict(enumerate(hist_patch2)))

#RPTA: 0.018576697105584232
```

b) Co-occurrence matrix:

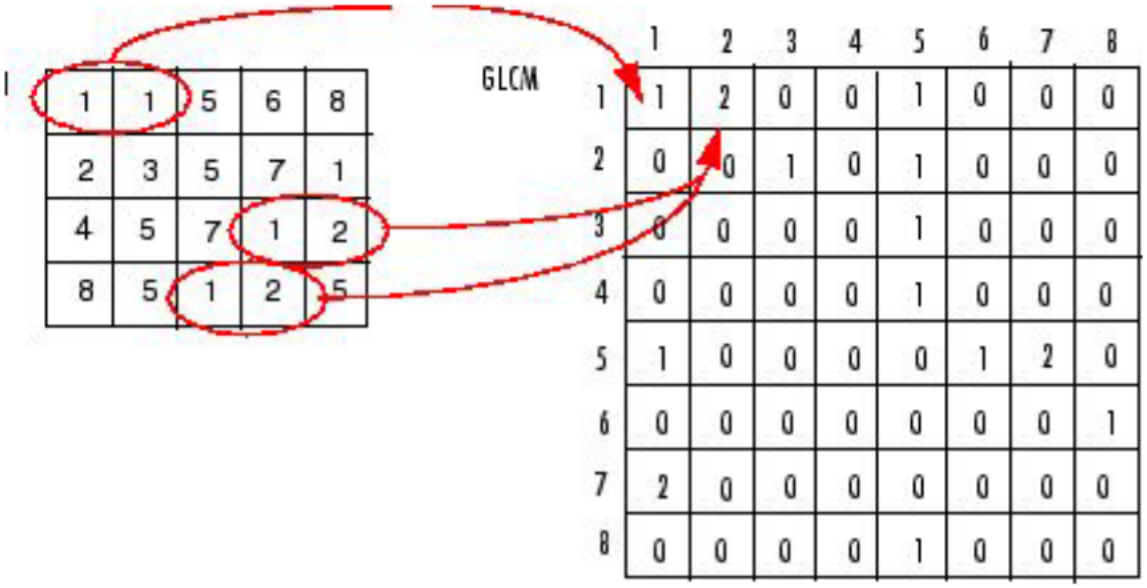
To explain what the co-occurrence matrix is and what its usefulness is, we must refer to Rob ert Haralick's 1973 article: Textural features for image classification. In this text the author clearly explains what is the need to create a statistical model that is really capable of describing a texture through its characteristics and also presents us with the co-occurrence matrix as the basis of said model.

Therefore, we will define the co-occurrence matrix as the one that describes the frequency of a gray level that appears in a specific spatial relationship with another gray value, within the area of a given window.[3].

To calculate a co-occurrence matrix, we will first be given directions, which can be any of the 8 basic directions (north, south, east, west and the 4 diagonals).[4]



We will also be given a distance, which in this case means how many different positions there are in the selected direction between the reference pixel and the neighbor.[4]



To calculate the co-occurrence matrix we are going to use the python library "`skimage`" [5], It contains two functions that will help us: "greycomatrix" and "greycoprops".

GREAYCOMATRIX: A grey level co-occurrence matrix is a histogram of co-occurring greyscale values at a given offset over an image. [6]

GREYCOPROPS: Compute a feature of a grey level co-occurrence matrix to serve as a compact summary of the matrix. The properties are computed as follows: contrast, dissimilarity, homogeneity, ASM, energy and correlation. [7]

In order to use these functions we have to convert the image to gray and then to integer.

```
#Read image
image = io.imread('/content/drive/My
Drive/imagenes/imagens_textura/textura2.png')
image_gris = color.rgb2gray(image)*255
image_int = image_gris.astype("uint8")
```

Image

Image Gris

Image Integer

```
#Full image co-occurrence matrix
GLCM = greycomatrix(image_int, [1], [0, np.pi/4, np.pi/2, 3*np.pi/4])
GLCM
```

array([[[[3, 1, 1, 1]],

[[2, 1, 4, 4]],

[[0, 1, 1, 2]],

...,

[[0, 0, 0, 0]],

[[0, 0, 0, 0]],

[[0, 0, 0, 0]]],

[[[2, 2, 3, 1]],

[[21, 14, 17, 14]],

[[10, 12, 16, 9]],

...,

[[0, 0, 0, 0]],

[[0, 0, 0, 0]],

[[0, 0, 0, 0]]],

[[[2, 3, 0, 1]],

[[17, 19, 18, 9]],

[[28, 19, 37, 27]],

...,

[[0, 0, 0, 0]],

[[0, 0, 0, 0]],

[[0, 0, 0, 0]]],

...,

[[0, 0, 0, 0]],

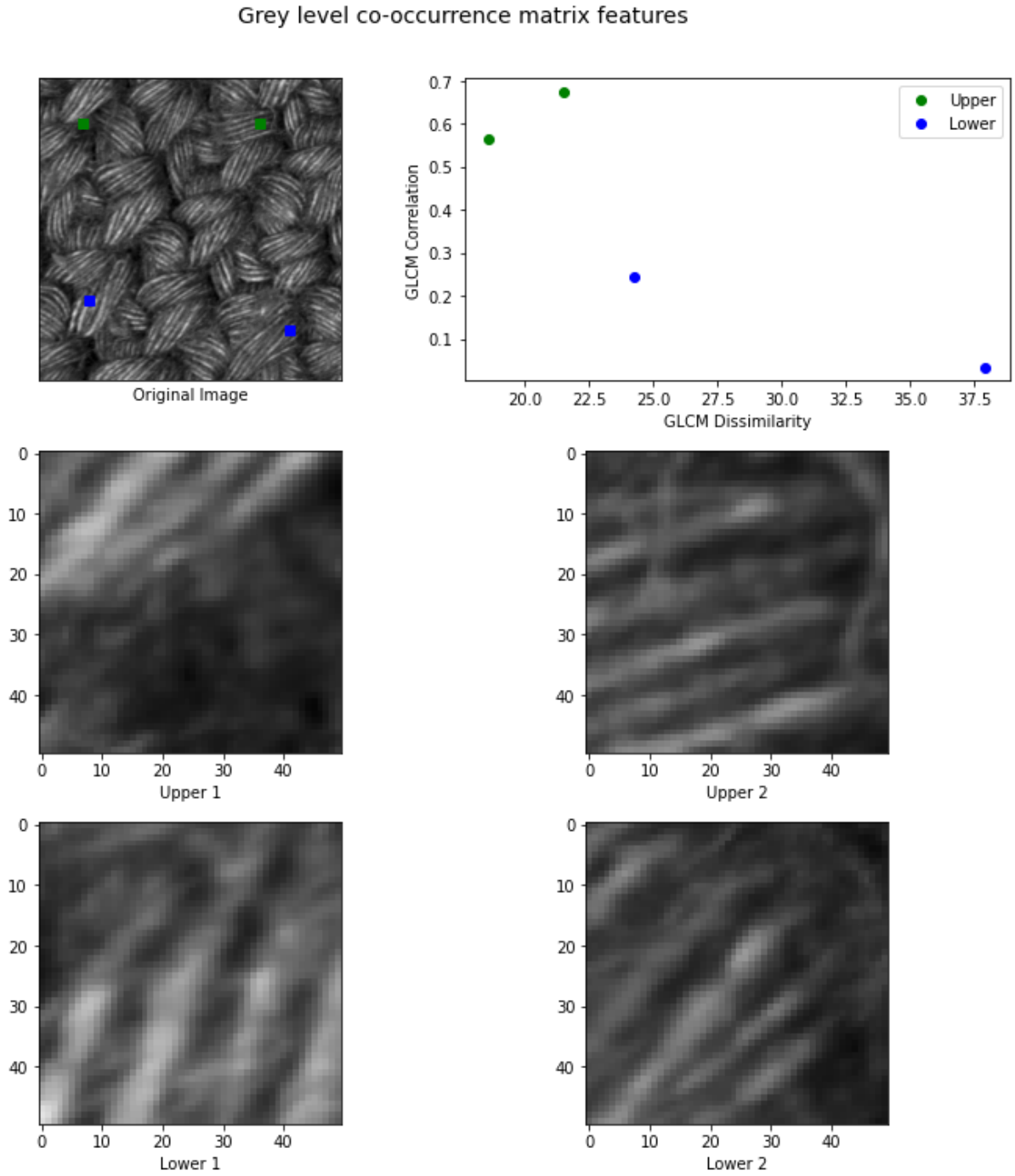
[[0, 0, 0, 0]],

[[0, 0, 0, 0]]]], dtype=uint32)

```
#Prop co-ocurrence matrix
dissimilarity_full= greycoprops(GLCM, 'dissimilarity')[0, 0]
correlation_full= greycoprops(GLCM, 'correlation')[0, 0]
homogeneity_full= greycoprops(GLCM, 'homogeneity')[0, 0]
energy_full= greycoprops(GLCM, 'energy')[0, 0]
contrast_full= greycoprops(GLCM, 'contrast')[0, 0]
print("dissimilarity full image: ", dissimilarity_full)
print("correlation full image: ", correlation_full)
print("homogeneity full image: ", homogeneity_full)
print("energy full image: ", energy_full)
print("contrast full image: ", contrast_full)
```

dissimilarity full image: 6.901804825097846
correlation full image: 0.9566178142682213
homogeneity full image: 0.1532335224413614
energy full image: 0.019117895580601826
contrast full image: 86.27265395670253

Now we are going to compare the two ends of the image, upper end and lower end, each with two points, in total 4 points of the image:



	dissimilarity	correlation	homogeneity	energy	contrast
Upper 1:	21.53066666666666	0.673857443843553	0.06584176020040879	0.020696513692222307	950.078222222222
Upper 2:	18.60177777777777	0.5653613500568522	0.05304320982225391	0.020294133434365987	552.5119999999997
Lower 1:	37.93688888888887	0.03464692945526327	0.025653944892864065	0.01744409058381365	2248.2595555555545
Lower 2:	24.27333333333333	0.24363516774560712	0.05019540526789443	0.02037185179798371	1002.5595555555552

This helps us to be able to compare how similar the textures of the same image are, we can appreciate that no matter how much parts of the same image are, there are parts that are totally different when comparing between: `dissimilarity vs correlation`

REFERENCES

1.-https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_local_binary_pattern.html
2.- https://numpy.org/doc/stable/reference/generated/numpy.histogram.html
3.- M.Presutti. La matriz de co-ocurrencia en la clasificación multiespectral. Universidad Nacional de La Plata, Agosto, 2004.
4.-https://eprints.ucm.es/id/eprint/62039/1/ZHU_CHEN_TFG2020-JoaquinBarrioLottmannXueboZhuChen_4398577_776647045.pdf
5.- https://scikit-image.org/
6.- https://www.kite.com/python/docs/skimage.feature.greycomatrix
7.- https://www.kite.com/python/docs/skimage.feature.greycoprops