Lab 3 -Low level image processing

- Interaction with the keyboard and drawing simple primitives.
- Constructing and visualizing histograms.
- Contrast-Stretching and histogram equalization.
- Thresholding.

3.1 2.1 Drawing primitives

Compile and test the file **Aula03_ex_01.py.**

Analyze the code and verify how the keyboard is used to choose the type of primitive to be drawn.

Also analyze the functions that allow drawing some primitives: line segment, circle and rectangle

3.2 Drawing a grid upon an image

Create a new example (**Aula_03_ex_02.py**) that allows superimposing a grid (spacing of 20 pixels) upon an image read from file and displays the resulting image.

If the original image is a gray-level image, the grid should be white.

If the original image is a color (RGB) image, the grid should be gray.

Save the resulting image in a file using the imwrite function.

3.3 Constructing and visualizing the histogram of a gray-level image

Compile and test the file **Aula03_exe_03.py.**

Analyze the code, in particular the following steps:

1. Defining the features and computing the image histogram.
2. Computing image features from the histogram.
3. Creating and displaying an image representing the histogram. On this case the matplotlib is used. As an alternative OpenCV drawing functions could also be used.

Observe what happens when some histogram features are changed: for instance, size (**histSize**) and range of values (**range**).

Optional

Develop auxiliary functions that implement and display histograms stages.

3.4 Analyzing the histograms of different images

For some of the example images given, analyze their histograms.

Analyze the different features of the image histograms for the image set **ireland-06-*** and classify each one of those images.

3.5 Contrast-Stretching

Create a new example (**Aula_03_exe_05.py**) that allows applying the Contrast-Stretching operation to a given gray-level image.

The original image and the resulting image should be visualized, as well as the respective histograms.

To accomplish that:

1. Use the **minMaxLoc** function to determine the smallest and largest image intensity values.
2. Create a new image that uses the entire range of intensity values (from 0 to 255).

For each image pixel, the intensity of the corresponding pixel in the resulting image is given by:

$$final[x, y] = \frac{original[x, y] - \min}{\max - \min} \times 255$$

Optional

Apply the Contrast-Stretching operation to the **DETI.bmp** image and the **input.png** image.

Visualize the histograms of the different images. What differences do you notice?

3.6 Histogram-Equalization

Create a new example (**Aula_03_exe_06.py**) that allows applying the Histogram-Equalization operation to a given gray-level image, using the **equalizeHist** function.

The original image and the resulting image should be visualized, as well as the respective histograms.

Apply the Histogram-Equalization operation to the **TAC_PULMAO.bmp** image.

What is the difference between the histograms of the original image and the equalized image?
What does the Histogram-Equalization operation allow?

3.7 Histograms of RGB images

Create a new example (**Aula_03_exe_07.py**) that allows visualizing the histogram of each color component of an RGB image, as well as the histogram of the corresponding gray-level image.

Use the **split** function to get the intensity images for each one of the color components.

For some of the example RGB images given, analyze their histograms.

Visualize the resulting images and the respective histograms.