Upsampling de Imágenes

El objetivo de este notebook es implementar un código en Python para aumentar el tamaño de las imágenes mediante una técnica de interpolación y procesamiento de imágenes.

Explicación de la idea

Para aumentar el tamaño de una imagen, realizaremos un proceso de interpolación. Crearemos una nueva imagen vacía con un tamaño mayor al de la imagen original, determinado por el factor de escalado que reciba la función. Luego, copiaremos los píxeles de la imagen original en la nueva imagen, distribuyéndolos de manera proporcional al factor de escalado.

Una vez hecho esto, aplicaremos una convolución para suavizar los píxeles vacíos que quedan entre los originales, lo cual ayuda a reducir artefactos visuales. Finalmente, utilizaremos una transformación de la LUT para ajustar los colores de la imagen ampliada, de modo que se mantengan consistentes con los de la imagen original.

Código del proyecto

Primero, importamos las librerías necesarias:

In [11]:
```
import numpy as np
import cv2
from skimage.exposure import match_histograms
import matplotlib.pyplot as plt
import matplotlib
matplotlib.rcParams['figure.figsize'] = (20.0, 20.0)

images_path = './images/'
```

Ahora vamos a crear la función encargada de realizar todas las transformaciones sobre la imagen de entrada

In [14]:
```
def upsample(image, scale_factor, verbose=False):

    # Obtener dimensiones de la imagen original
    h, w = image.shape

    # Crear una nueva imagen de tamaño aumentado
    new_h = h * scale_factor
    new_w = w * scale_factor
    upsampled_image = np.zeros((new_h, new_w))

    # Asignar los píxeles de la imagen original a la nueva imagen
    for i in range(h):
        for j in range(w):
            upsampled_image[i * scale_factor, j * scale_factor] = image[i, j]

    # Crear un kernel de suavizado para la interpolación
    kernel = np.ones((scale_factor, scale_factor)) / (scale_factor ** 2)

    # Aplicar la convolución usando filter2D
    smoothed_image = cv2.filter2D(upsampled_image, -1, kernel)

    # Normalizar la imagen resultante
    normalized_image = cv2.normalize(smoothed_image, None, 0, 255, cv2.NORM_MINMAX)

    # Aplicamos la LUT de la imagen original a la aumentada para que mantenga los mismos colores
    out_image = match_histograms(normalized_image, image)

    if verbose:
        plt.subplot(231)
        plt.imshow(upsampled_image, cmap='gray')
        plt.title('Upsampled')

        plt.subplot(232)
        plt.imshow(normalized_image, cmap='gray')
        plt.title('Smoothed & Normalized')

        plt.subplot(233)
        plt.imshow(out_image, cmap='gray')
        plt.title('Result')

        plt.subplot(235)
        plt.hist(normalized_image.ravel(),256,[0, 255])
        plt.title('Smoothed & Normalized histogram')

        plt.subplot(236)
        plt.hist(out_image.ravel(),256,[0, 255])
        plt.title('Result histogram')

    return out_image
```

Finalmente, para poder probar la función anterior, vamos a cargar una imagen de ejemplo y aplicar el proceso de upsampling con las transformaciones correspondientes.

In [15]:
```
image = cv2.imread(images_path + 'lena.jpeg', 0)
scale_factor = 3

out_image = upsample(image, scale_factor, verbose=True)
```