Introducción al procesamiento de imágenes

Dhali Tejeda - A00344820

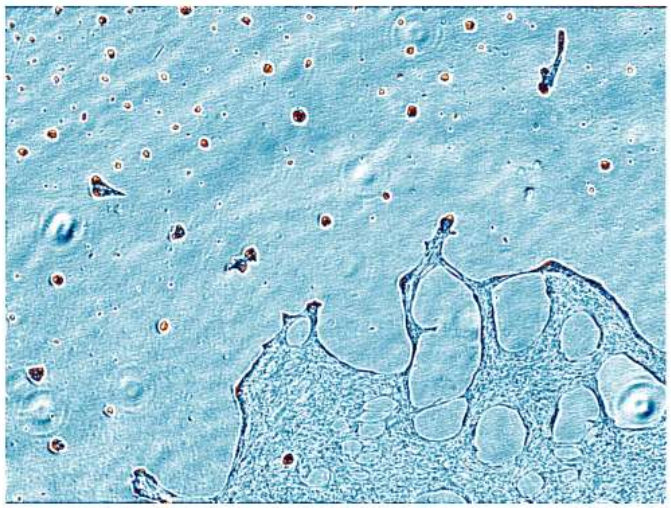
Instalación de OpenCV

```
!pip install opencv-python-headless
```

```
Requirement already satisfied: opencv-python-headless in /usr/local/lib/python3.10/dist-packages (4.10.0.84)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packages (from opencv-python-headless) (1.26.4)
```

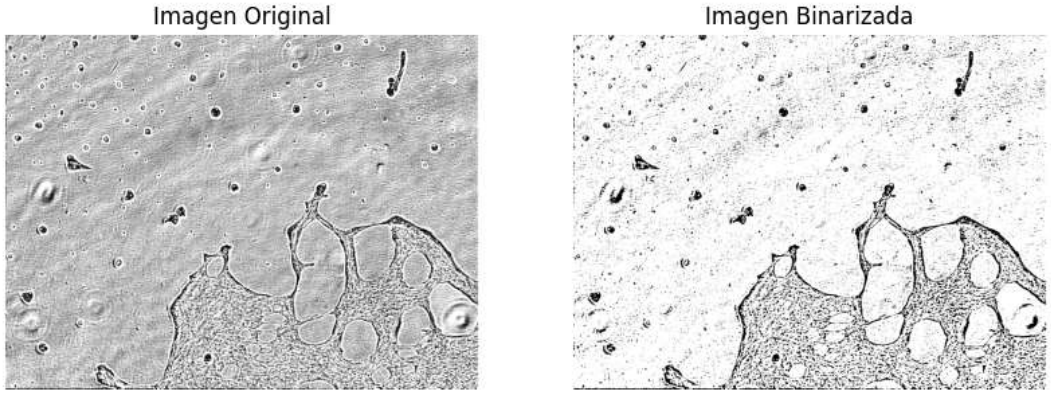
2. Construya un script que le permita visualizar las imágenes de entrada dentro de un ciclo.

```
import cv2
import matplotlib.pyplot as plt
import glob

# Cargar imágenes desde un directorio (modifica la ruta según tu archivo)
imagenes = glob.glob('/content/sample_data/88797675-e851-4bb9-93df-0d446823b511_A2_02_06_Phi8Color.png')

# Visualizar imágenes en un ciclo
for img_path in imagenes:
    # Leer la imagen
    img = cv2.imread(img_path, cv2.IMREAD_COLOR)

    # Convertir de BGR a RGB para visualizarla correctamente con matplotlib
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    # Mostrar la imagen
    plt.imshow(img_rgb)
    plt.axis('off')  # Para quitar los ejes
    plt.show()
```



3. Aplique la binarización por threshold sobre una muestra de sus imágenes de reto. El objetivo es que vea si puede segmentar elementos de interés.

```
# Leer nuestra imagen de ejemplo
img = cv2.imread('/content/sample_data/88797675-e851-4bb9-93df-0d446823b511_A2_02_06_Phi8Color.png', cv2.IMREAD_GRAYSCALE)

if img is None:
  print("Failed to load image. Check the file path.")
else:
  # Aplicar binarización por threshold
  _, img_bin = cv2.threshold(img, 128, 255, cv2.THRESH_BINARY)

  # Mostrar imagen original y binaria
  plt.figure(figsize=(10,5))

  plt.subplot(1, 2, 1)
  plt.title('Imagen Original')
  plt.imshow(img, cmap='gray')
  plt.axis('off')

  plt.subplot(1, 2, 2)
  plt.title('Imagen Binarizada')
  plt.imshow(img_bin, cmap='gray')
  plt.axis('off')

  plt.show()
```



4. Haga una experimentación sobre la máscara binaria obtenida en el punto 3, utilizando las funciones de erosión y dilatación y analice los resultados.

```
# Definir un kernel (matriz para la operación de erosión)
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))

# Aplicar erosión
img_eroded = cv2.erode(img_bin, kernel, iterations=1)

# Mostrar el resultado de la erosión
plt.imshow(img_eroded, cmap='gray')
plt.title('Imagen Erosionada')
plt.axis('off')
plt.show()

# Aplicar dilatación
img_dilated = cv2.dilate(img_bin, kernel, iterations=1)

# Mostrar el resultado de la dilatación
plt.imshow(img_dilated, cmap='gray')
plt.title('Imagen Dilatada')
plt.axis('off')
plt.show()
```

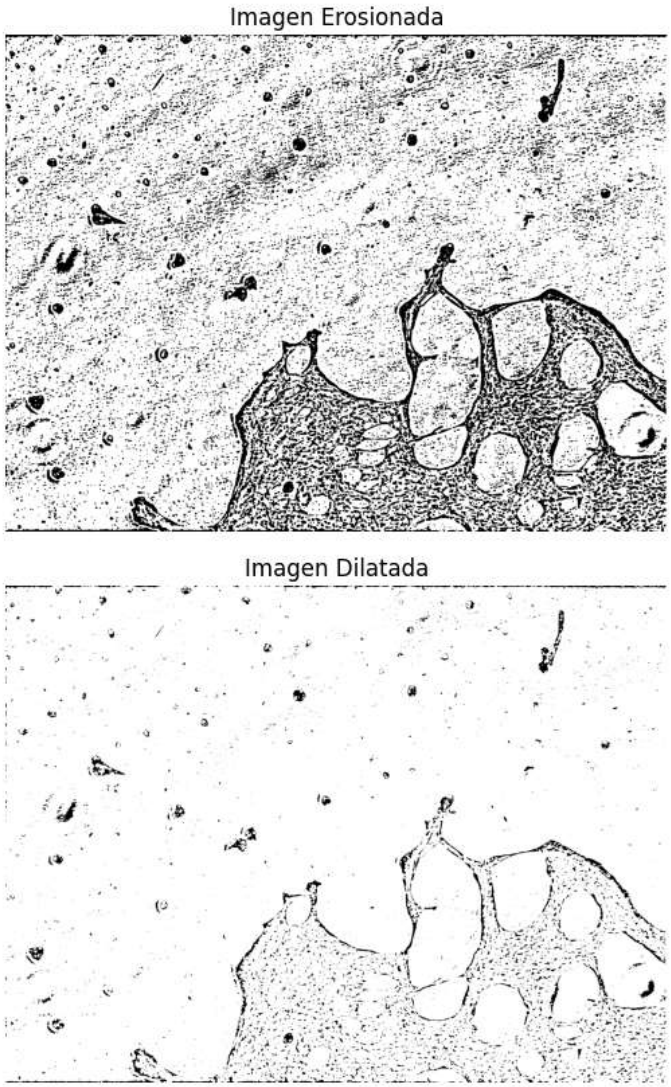


Imagen Erosionada
Imagen Dilatada