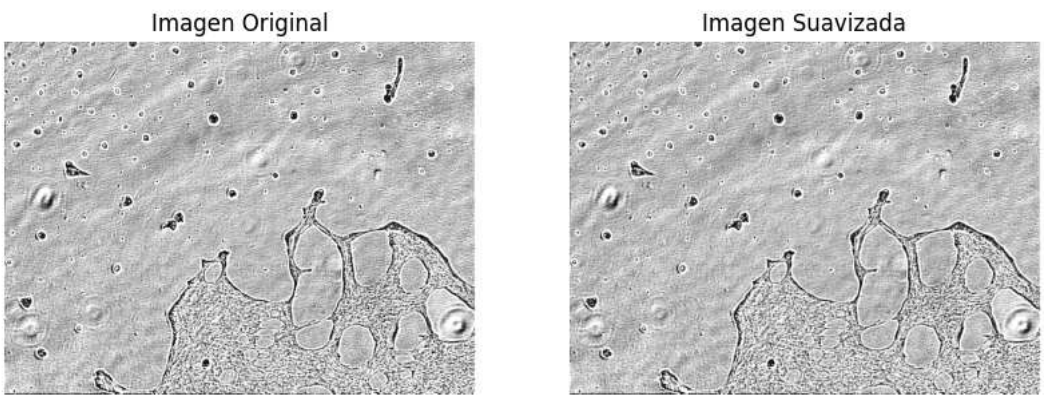
Segmentación de imágenes

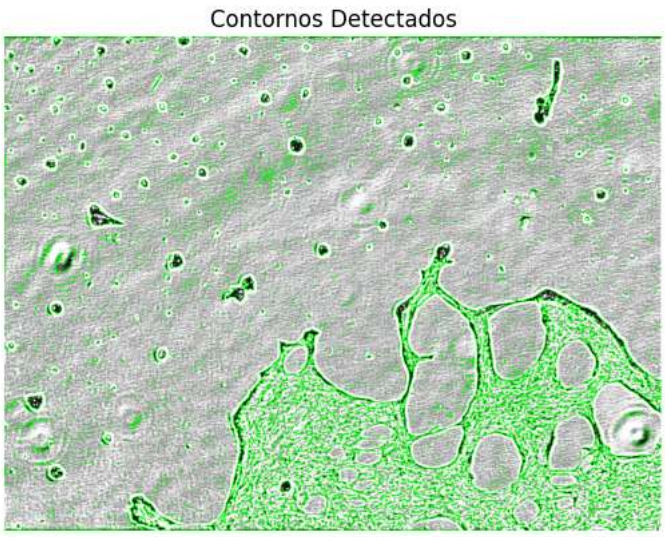
Dhali Tejeda - A00344820

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Cargar la imagen en escala de grises
img = cv2.imread('/content/sample_data/88797675-e851-4bb9-93df-0d446823b511_A2_02_06_Phi8Color.png', cv2.IMREAD_GRAYSCALE)

# Aplicar un filtro gaussiano para suavizar la imagen
blurred_img = cv2.GaussianBlur(img, (5, 5), 0)

# Mostrar la imagen original y suavizada
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(img, cmap='gray')
plt.title('Imagen Original')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(blurred_img, cmap='gray')
plt.title('Imagen Suavizada')
plt.axis('off')
plt.show()
```



Encontrar contornos en la imagen

```
# Binarización por threshold
_, img_bin = cv2.threshold(blurred_img, 128, 255, cv2.THRESH_BINARY_INV)

# Encontrar los contornos
contours, _ = cv2.findContours(img_bin, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# Dibujar los contornos sobre la imagen original
img_contours = cv2.cvtColor(img, cv2.COLOR_GRAY2BGR)
cv2.drawContours(img_contours, contours, -1, (0, 255, 0), 2)

# Mostrar los contornos
plt.imshow(img_contours)
plt.title('Contornos Detectados')
plt.axis('off')
plt.show()
```

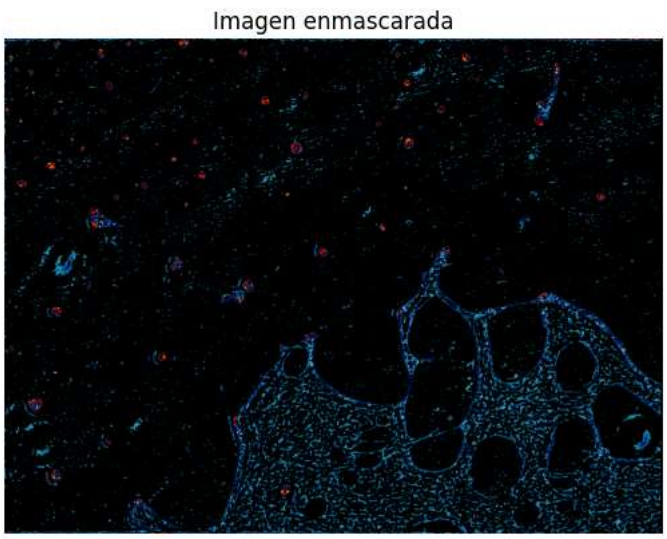


Aplicar las operaciones morfológicas a la imagen

```
# Definir un kernel para las operaciones morfológicas
kernel = np.ones((3, 3), np.uint8)

# Aplicar dilatación para unir regiones
img_dilated = cv2.dilate(img_bin, kernel, iterations=2)

# Aplicar erosión para eliminar ruido
img_eroded = cv2.erode(img_dilated, kernel, iterations=2)

plt.imshow(img_eroded, cmap='gray')
plt.title('Resultado después de operaciones morfológicas')
plt.axis('off')
plt.show()
```

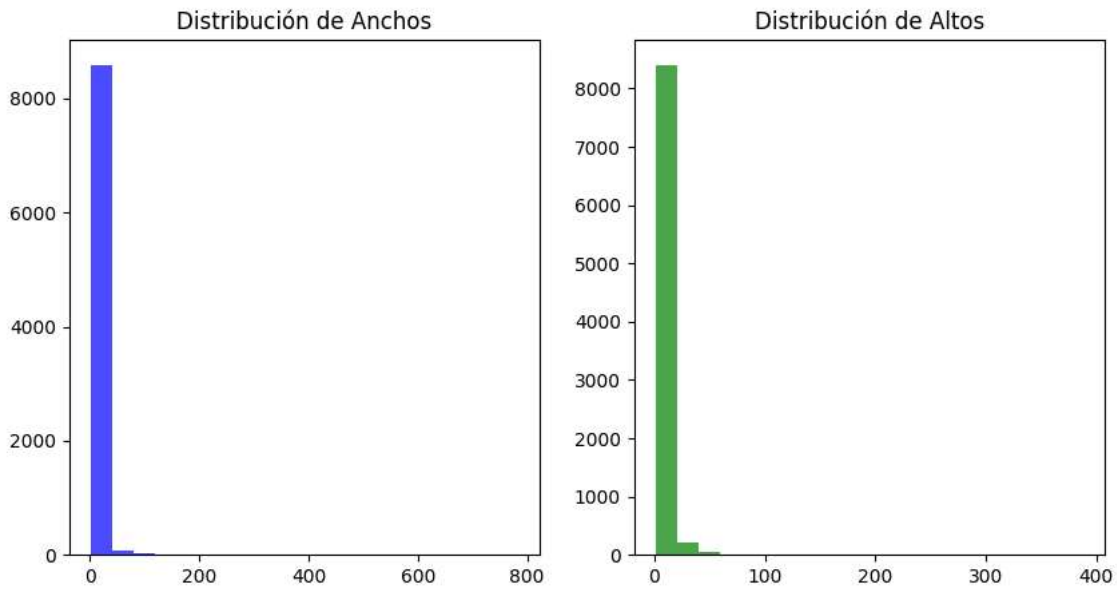
Crear una copia de la imagen original en color para hacer el masking

```
img_color = cv2.imread('/content/sample_data/88797675-e851-4bb9-93df-0d446823b511_A2_02_06_Phi8Color.png')

# Aplicar la máscara binaria a la imagen original
masked_img = cv2.bitwise_and(img_color, img_color, mask=img_eroded)

plt.imshow(cv2.cvtColor(masked_img, cv2.COLOR_BGR2RGB))
plt.title('Imagen enmascarada')
plt.axis('off')
plt.show()
```



Obtener las distribuciones de ancho y alto de cada crop

```
heights = []
widths = []

for contour in contours:
    x, y, w, h = cv2.boundingRect(contour)
    widths.append(w)
    heights.append(h)

plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.hist(widths, bins=20, color='blue', alpha=0.7)
plt.title('Distribución de Anchos')

plt.subplot(1, 2, 2)
plt.hist(heights, bins=20, color='green', alpha=0.7)
plt.title('Distribución de Altos')

plt.show()
```



Calcular los estimadores de las distribuciones y visualizaciones.

```
import numpy as np

# Calcular los estimadores de las distribuciones
mean_width = np.mean(widths)
mean_height = np.mean(heights)

std_width = np.std(widths)
std_height = np.std(heights)

print(f"Media de los anchos: {mean_width}, Desviación estándar: {std_width}")
print(f"Media de los altos: {mean_height}, Desviación estándar: {std_height}")
```

```
Media de los anchos: 6.912484208108419, Desviación estándar: 16.882195794776614
Media de los altos: 5.5623061904215, Desviación estándar: 11.84328271454853
```