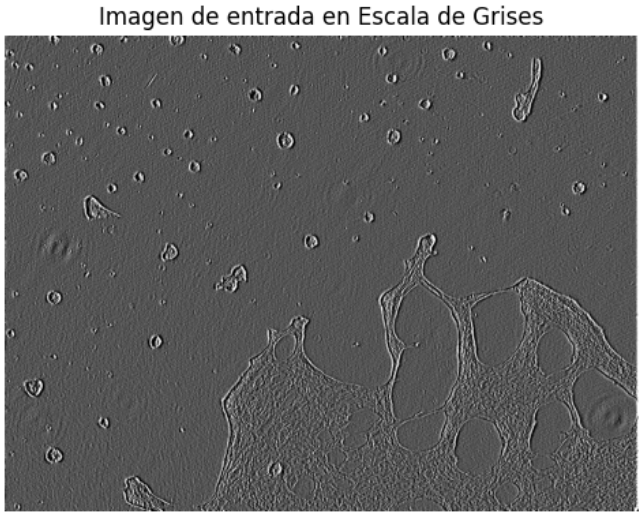
```
!pip install opencv-python-headless
```

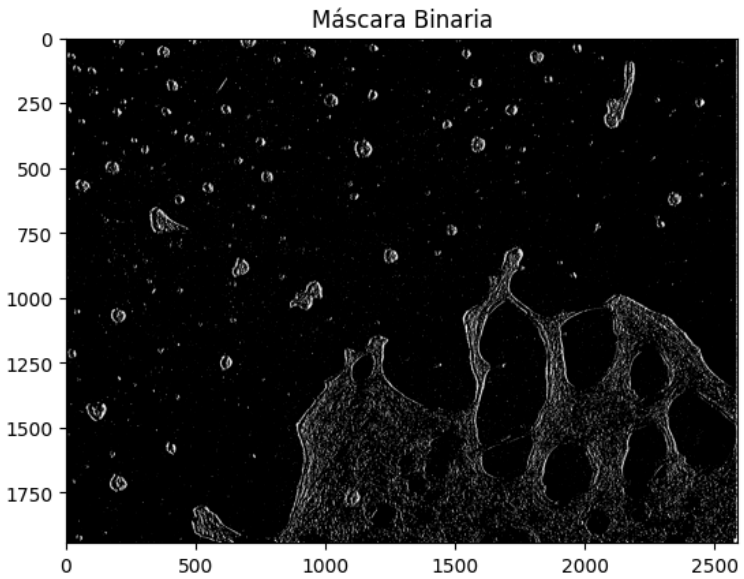
```
Requirement already satisfied: opencv-python-headless in /usr/local/lib/python3.10/dist-packages (4.10.0.84)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packages (from opencv-python-headless) (1.2
```

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

```
img = cv2.imread('/content/_DIC8.png', cv2.IMREAD_GRAYSCALE)
if img is not None:
    plt.figure(figsize=(6,6))
    plt.imshow(img, cmap='gray')
    plt.title('Imagen de entrada en Escala de Grises')
    plt.axis('off')
plt.show()
```

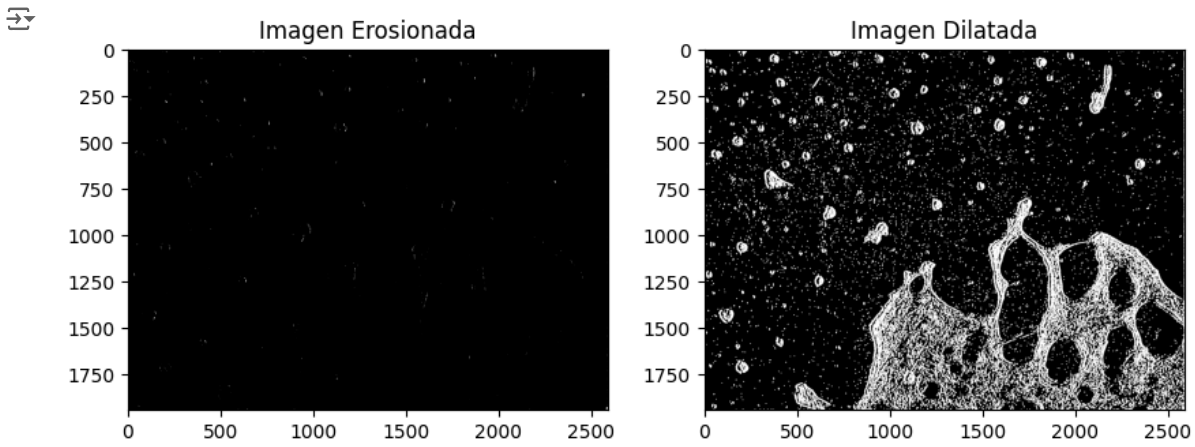



```
threshold_value = 128
_, binary_mask = cv2.threshold(img, threshold_value, 255, cv2.THRESH_BINARY)
plt.imshow(binary_mask, cmap='gray')
plt.title('Máscara Binaria')
plt.show()
```




```
kernel = np.ones((5, 5), np.uint8)
eroded_image = cv2.erode(binary_mask, kernel, iterations=1)
dilated_image = cv2.dilate(binary_mask, kernel, iterations=1)
```

```
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(eroded_image, cmap='gray')
plt.title('Imagen Erosionada')
plt.subplot(1, 2, 2)
plt.imshow(dilated_image, cmap='gray')
plt.title('Imagen Dilatada')
plt.show()
```



Parte 2

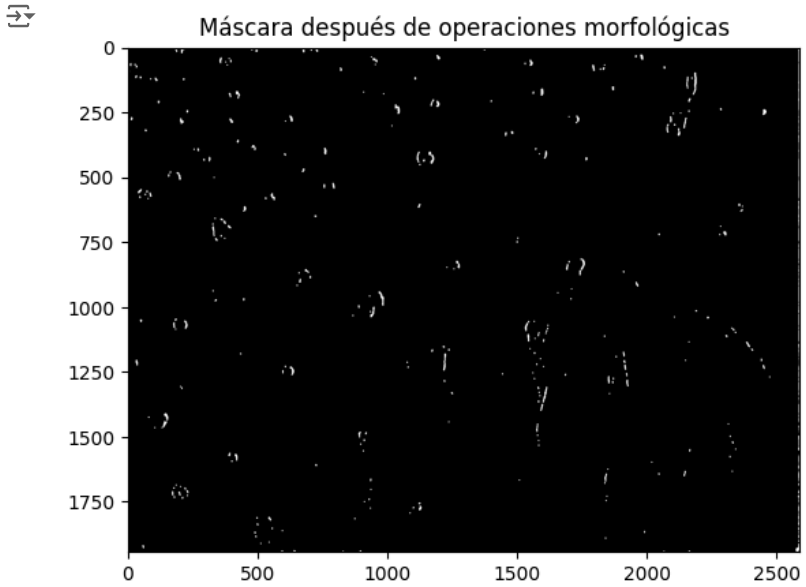
Generate create a dataframe with 2 columns and 10 rows Close

```
kernel = np.ones((5, 5), np.uint8)

# Aplicar apertura (erosión seguida de dilatación) para eliminar ruido
opening = cv2.morphologyEx(binary_mask, cv2.MORPH_OPEN, kernel)

# Aplicar cierre (dilatación seguida de erosión) para cerrar huecos en los blobs
closing = cv2.morphologyEx(opening, cv2.MORPH_CLOSE, kernel)

plt.imshow(closing, cmap='gray')
plt.title('Máscara después de operaciones morfológicas')
plt.show()
```

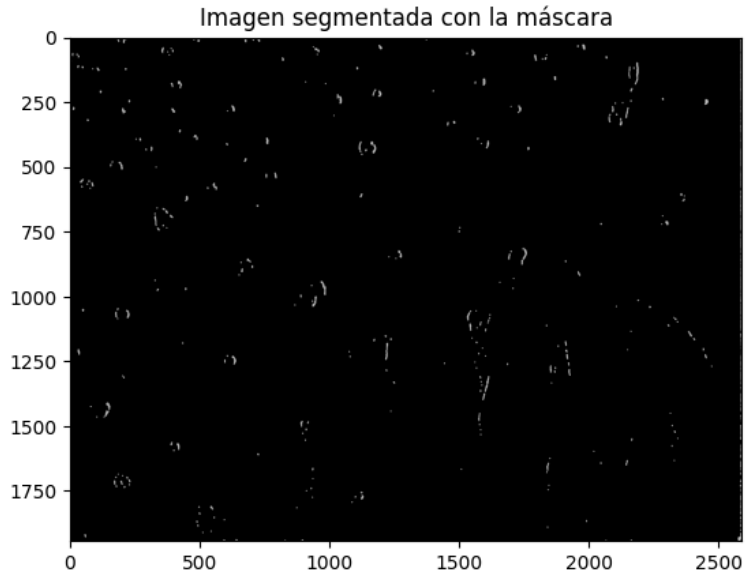



```
img_color = cv2.imread('/content/_DIC8.png')

masked_image = cv2.bitwise_and(img_color, img_color, mask=closing)

plt.imshow(cv2.cvtColor(masked_image, cv2.COLOR_BGR2RGB))  # Convertimos a RGB para visualización en Matplotlib
```

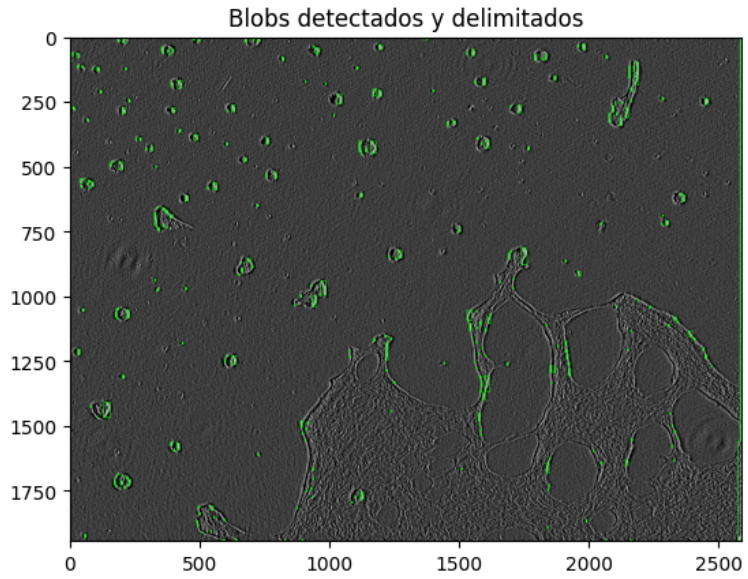
```
plt.title('Imagen segmentada con la máscara')
plt.show()
```




```
contours, _ = cv2.findContours(closing, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
widths = []
heights = []

for cnt in contours:
    x, y, w, h = cv2.boundingRect(cnt)
    widths.append(w)
    heights.append(h)
    cv2.rectangle(img_color, (x, y), (x + w, y + h), (0, 255, 0), 2)

plt.imshow(cv2.cvtColor(img_color, cv2.COLOR_BGR2RGB))
plt.title('Blobs detectados y delimitados')
plt.show()
```




```
import seaborn as sns

mean_width = np.mean(widths)
std_width = np.std(widths)
mean_height = np.mean(heights)
std_height = np.std(heights)

print(f"Ancho promedio: {mean_width:.2f}, Desviación estándar: {std_width:.2f}")
```

```
print(f"Alto promedio: {mean_height:.2f}, Desviación estándar: {std_height:.2f}")

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
sns.histplot(widths, kde=True, color='blue')
plt.title('Distribución de Anchos')

plt.subplot(1, 2, 2)
sns.histplot(heights, kde=True, color='red')
plt.title('Distribución de Altos')

plt.show()
```

```
Ancho promedio: 7.85, Desviación estándar: 3.71
Alto promedio: 19.47, Desviación estándar: 93.32
```