Introducción al procesamiento de imágenes

Cynthia Cristal Quijas Flores - a01655996

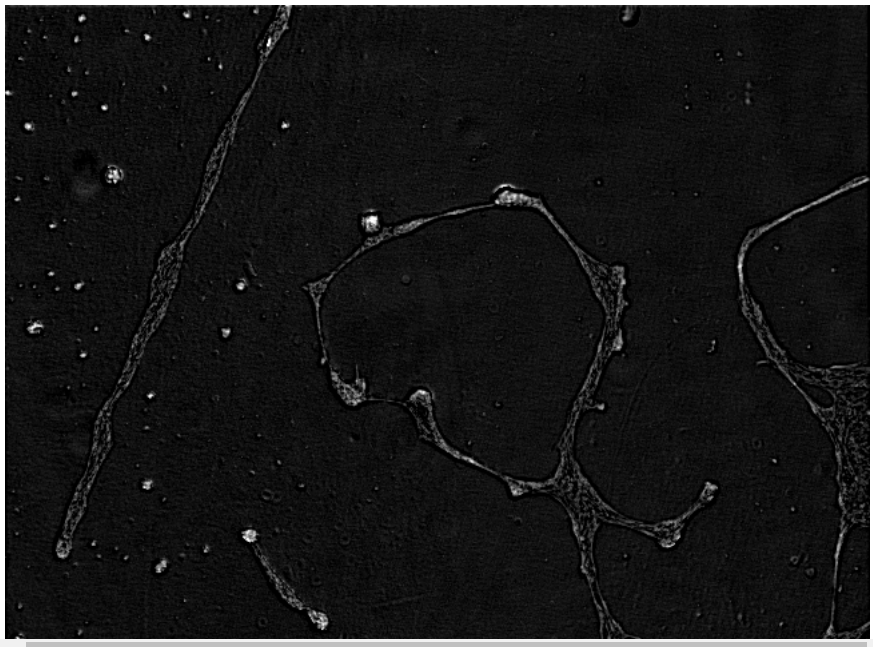
```
import cv2
```

```
image = cv2.imread('2024-08-15_21-23-02_A1_02_04__Phi8.png', -1)
```

```
img = cv2.resize(image, (0,0), fx=0.25, fy=.25)
```

Mostrar Imagen Original

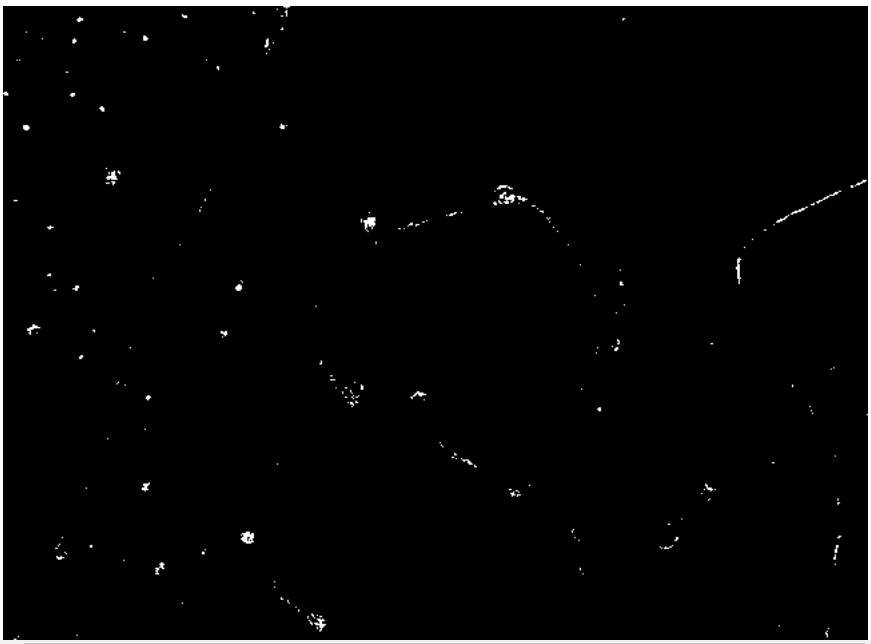
```
# !pip install opencv-python-headless

import cv2
from google.colab.patches import cv2_imshow

cv2_imshow(img) # Use cv2_imshow instead of cv2.imshow
cv2.waitKey(0)
cv2.destroyAllWindows()
```



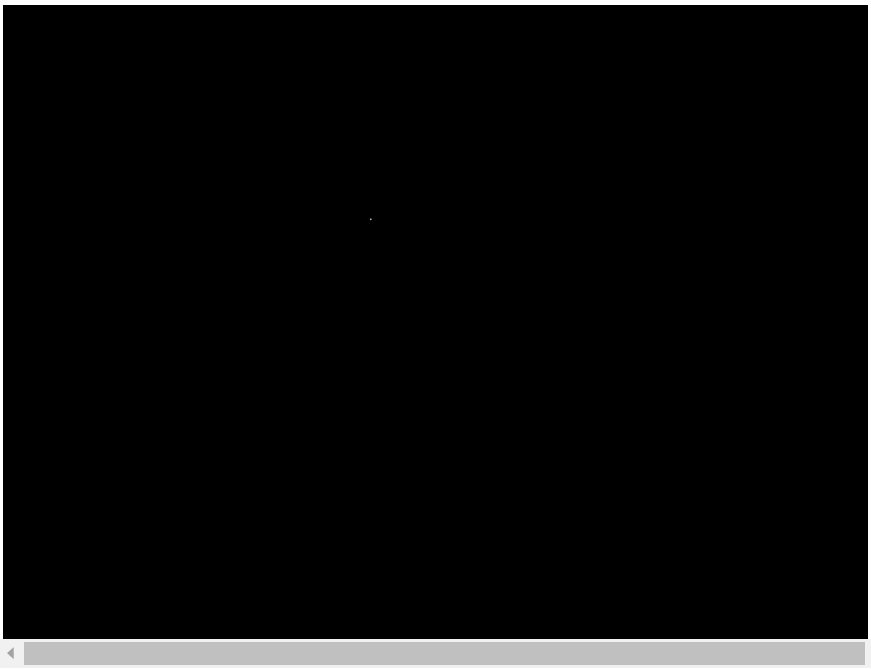
Redimensionzar la imagen y aplicaar threshold

```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

# Función para mostrar la imagen (cv2_imshow si estás en Colab)
def mostrar_imagen(imagen, titulo="Imagen"):
    if 'google.colab' in str(get_ipython()):
        cv2_imshow(imagen)  # Google Colab
    else:
        cv2.imshow(titulo, imagen)
        cv2.waitKey(0)
        cv2.destroyAllWindows()

# Cargar la imagen
image = cv2.imread('2024-08-15_21-23-02_A1_02_04__Phi8.png', -1)

if image is None:
    print("Error: Could not load image. Please check the file path."
else:
    # Redimensionar la imagen.
```

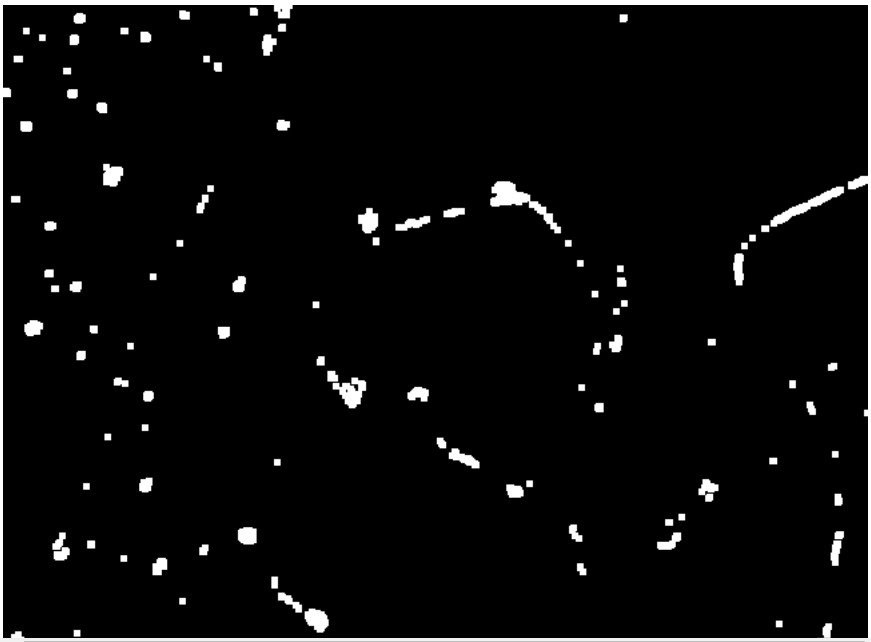
```
img_resized = cv2.resize(image, None, fx=0.25, fy=0.25)

# Mostrar la imagen original
# mostrar_imagen(img_resized, "Imagen Original")

# Paso 1: Aplicar binarización por threshold
if len(img_resized.shape) == 3: # Check if image is color
    gray = cv2.cvtColor(img_resized, cv2.COLOR_BGR2GRAY)
else: # Image is already grayscale
    gray = img_resized
_, binary = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)

# Mostrar la imagen binarizada
mostrar_imagen(binary, "Imagen Binarizada")
```



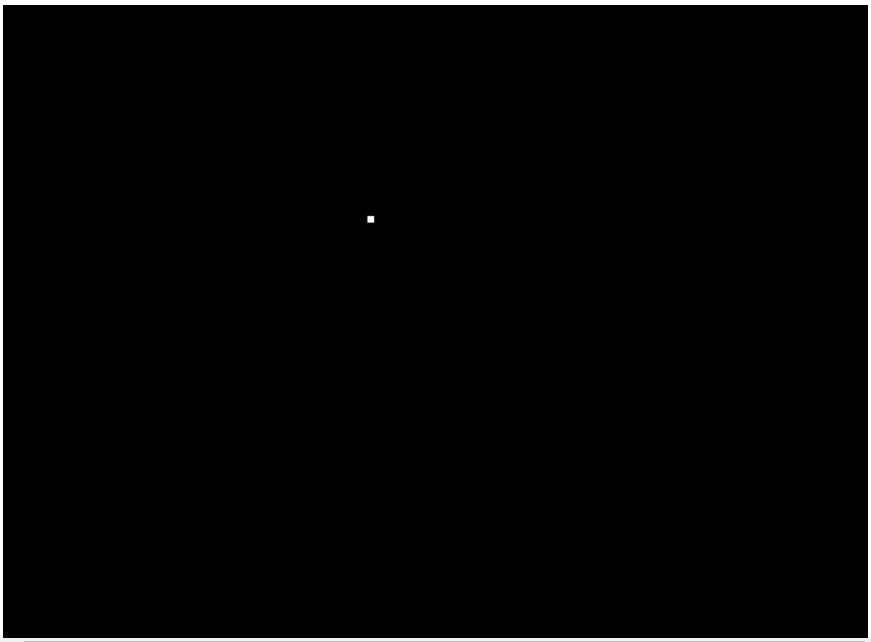
```
# Paso 2: Aplicar operaciones morfológicas (Erosión y Dilatación)
# Kernel para las operaciones morfológicas
kernel = np.ones((5,5), np.uint8)

# Erosión
erosion = cv2.erode(binary, kernel, iterations=1)
mostrar_imagen(erosion, "Erosión")
```



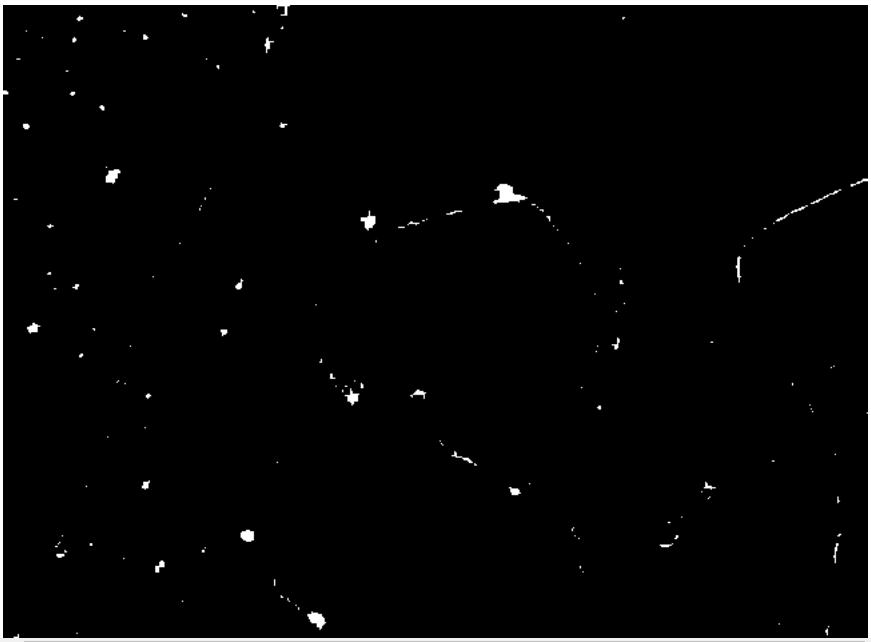
```
# Dilatación
dilation = cv2.dilate(binary, kernel, iterations=1)
mostrar_imagen(dilation, "Dilatación")
```



```
# Paso 3: Combinar las operaciones
# Erosión seguida de dilatación (Apertura)
opening = cv2.morphologyEx(binary, cv2.MORPH_OPEN, kernel)
mostrar_imagen(opening, "Apertura")
```

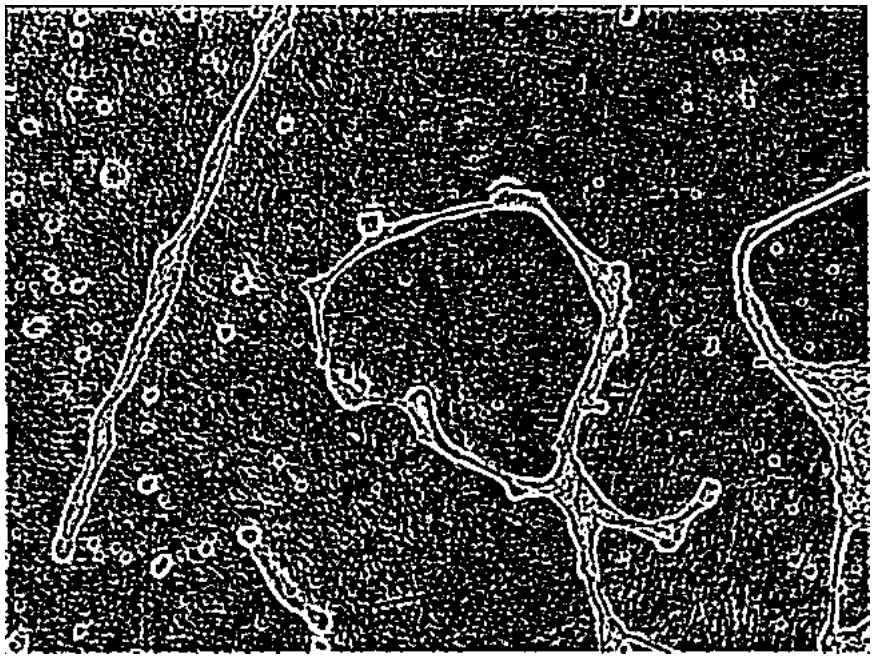


```
# Dilatación seguida de erosión (Cierre)
closing = cv2.morphologyEx(binary, cv2.MORPH_CLOSE, kernel)
mostrar_imagen(closing, "Cierre")
```



Obtenga la segmentación de objetos de interés en las imágenes de reto.

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

# Función para mostrar la imagen
def mostrar_imagen(imagen, titulo="Imagen"):
    if 'google.colab' in str(get_ipython()):
        from google.colab.patches import cv2_imshow
        cv2_imshow(imagen)
    else:
        cv2.imshow(titulo, imagen)
        cv2.waitKey(0)
        cv2.destroyAllWindows()

# Cargar la imagen de reto
image = cv2.imread('2024-08-15_21-23-02_A1_02_04__Phi8.png', -1)

if image is None:
    print("Error: Could not load image. Please check the file path.")
else:
    # Redimensionar la imagen si es necesario (opcional)
    img_resized = cv2.resize(image, None, fx=0.25, fy=0.25)

    # Paso 1: Preprocesamiento (convertir a escala de grises)
    if len(img_resized.shape) == 3:  # Si la imagen es a color
        gray = cv2.cvtColor(img_resized, cv2.COLOR_BGR2GRAY)
    else:  # Si la imagen ya está en escala de grises
        gray = img_resized

    # Aplicar un filtro Gaussiano para suavizar y reducir ruido
    blurred = cv2.GaussianBlur(gray, (5, 5), 0)

    # Paso 2: Segmentación inicial con umbralización adaptativa
    # Umbral adaptativo para mejorar la segmentación en imágenes con iluminación variable
    binary = cv2.adaptiveThreshold(blurred, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                   cv2.THRESH_BINARY_INV, 11, 2)

    # Mostrar la imagen binarizada
    mostrar_imagen(binary, "Umbralización Adaptativa")


```

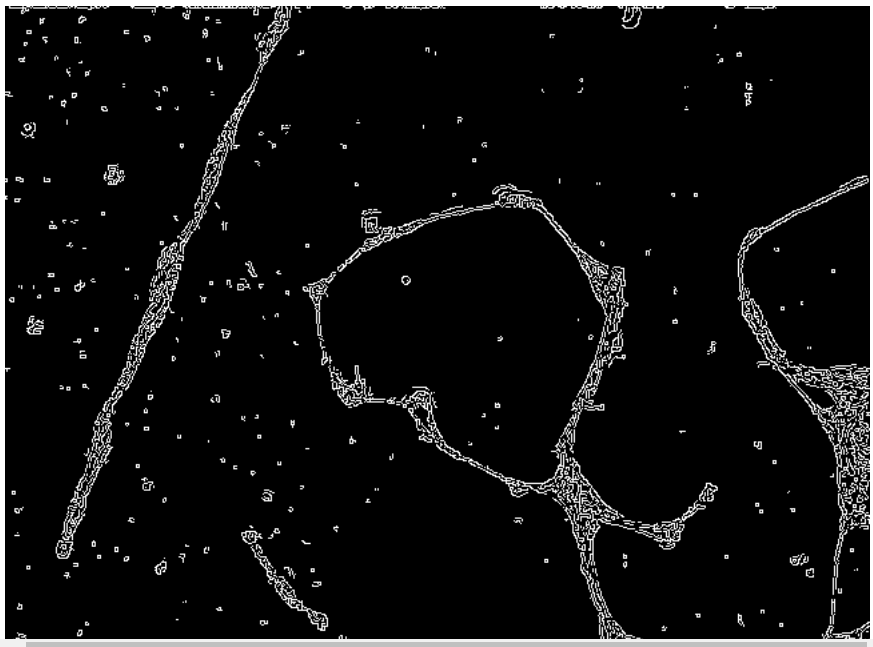


A partir de los blobs obtenidos en la máscara binaria, aplique filtros y/o operaciones morfológicas para el refinamiento de los mismos, en caso de ser necesario.

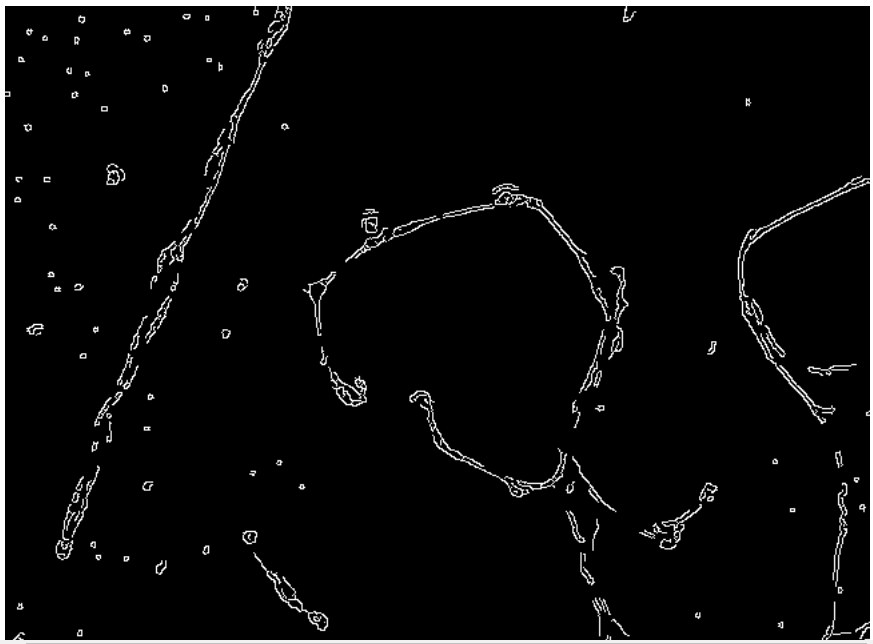
```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

# Función para mostrar la imagen (cv2_imshow si estás en Colab)
def mostrar_imagen(imagen, titulo="Imagen"):
    if 'google.colab' in str(get_ipython()):
        cv2_imshow(imagen)  # Google Colab
    else:
        cv2.imshow(titulo, imagen)
        cv2.waitKey(0)
        cv2.destroyAllWindows()

# Cargar la imagen
image = cv2.imread('2024-08-15_21-23-02_A1_02_04__Phi8.png', -1)

if image is None:
    print("Error: Could not load image. Please check the file path.")
else:
    # Redimensionar la imagen.
    img_resized = cv2.resize(image, None, fx=0.25, fy=0.25)

    # Paso 1: Aplicar binarización por threshold
    if len(img_resized.shape) == 3: # Check if image is color
        gray = cv2.cvtColor(img_resized, cv2.COLOR_BGR2GRAY)
    else: # Image is already grayscale
        gray = img_resized
    _, binary = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)

    # Mostrar la imagen binarizada
    mostrar_imagen(binary, "Imagen Binarizada")

    # Paso 2: Aplicar operaciones morfológicas (Erosión y Dilatación)
    kernel = np.ones((5,5), np.uint8)
    erosion = cv2.erode(binary, kernel, iterations=1)
    mostrar_imagen(erosion, "Erosión")
    dilation = cv2.dilate(binary, kernel, iterations=1)
    mostrar_imagen(dilation, "Dilatación")
    opening = cv2.morphologyEx(binary, cv2.MORPH_OPEN, kernel)
    mostrar_imagen(opening, "Apertura")
    closing = cv2.morphologyEx(binary, cv2.MORPH_CLOSE, kernel)
    mostrar_imagen(closing, "Cierre")

    # Paso 3: Aplicar filtros avanzados

    # 3.1: Hysteresis Thresholding (utilizado dentro del detector de bordes Canny)
```

```
47      edges_canny = cv2.Canny(gray, 100, 200)  # Umbrales de histéresis
48
49      # 3.2: Aplicar un filtro Gaussiano para suavizar la imagen
50      gaussian_blur = cv2.GaussianBlur(gray, (5, 5), 0)
51
52      # 3.3: Aplicar detector de bordes Canny sobre la imagen suavizada
53      edges_gaussian_canny = cv2.Canny(gaussian_blur, 100, 200)
54
55      # 3.4: Refinamiento final usando Cierre Morfológico sobre los bordes detectado
56      closing_edges = cv2.morphologyEx(edges_gaussian_canny, cv2.MORPH_CLOSE, kernel
57
```

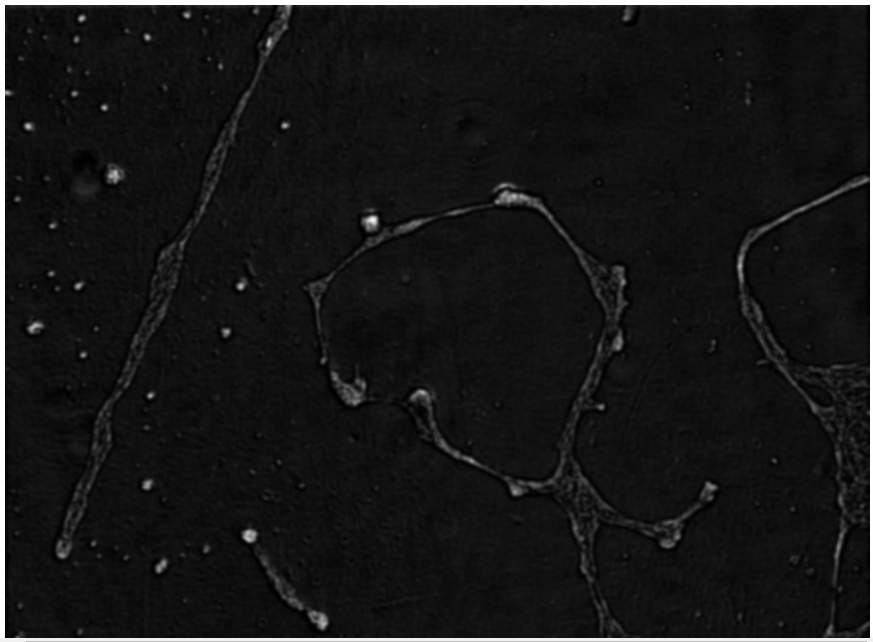
```
mostrar_imagen(edges_canny, "Bordes con Canny")
```



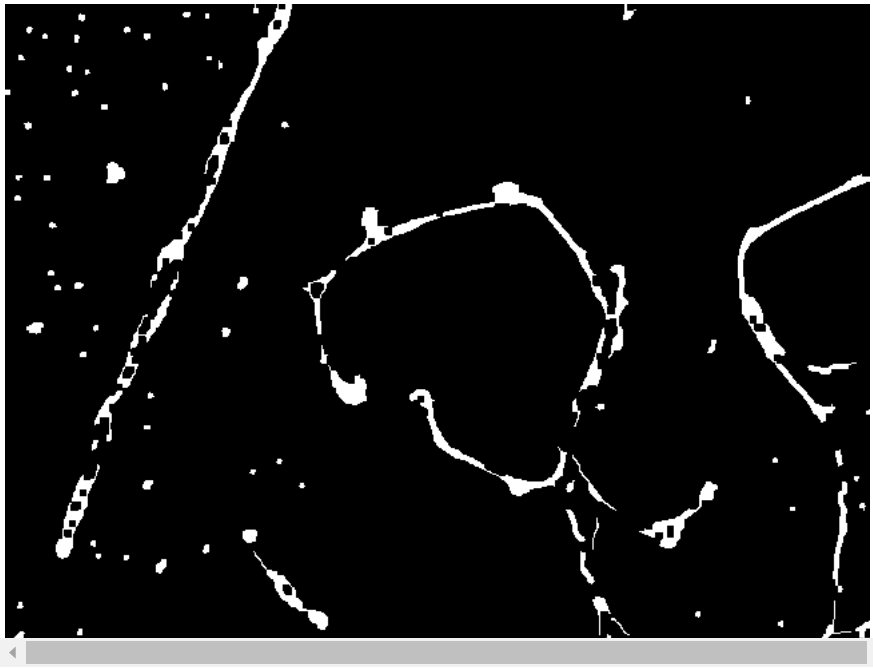
```
mostrar_imagen(edges_gaussian_canny, "Bordes Canny sobre Imagen Suavizada")
```



```
mostrar_imagen(gaussian_blur, "Suavizado con Filtro Gaussiano")
```



```
mostrar_imagen(closing_edges, "Refinamiento con Cierre Morfológico sobre Bordes")
```



A partir de las máscaras obtenidas, obtenga las muestras sobre las imágenes reales (usando “image masking” o una técnica similar). Después obtenga las distribuciones de ancho y alto de cada crop, especifique el tamaño de la muestra, los estimadores de las distribuciones y visualizaciones.

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

# Función para mostrar la imagen
def mostrar_imagen(imagen, titulo="Imagen"):
    if 'google.colab' in str(get_ipython()):
        from google.colab.patches import cv2_imshow
        cv2_imshow(imagen)
    else:
        cv2.imshow(titulo, imagen)
        cv2.waitKey(0)
        cv2.destroyAllWindows()

```