Introducción al procesamiento de imágenes

Cynthia Cristal Quijas Flores - a01655996

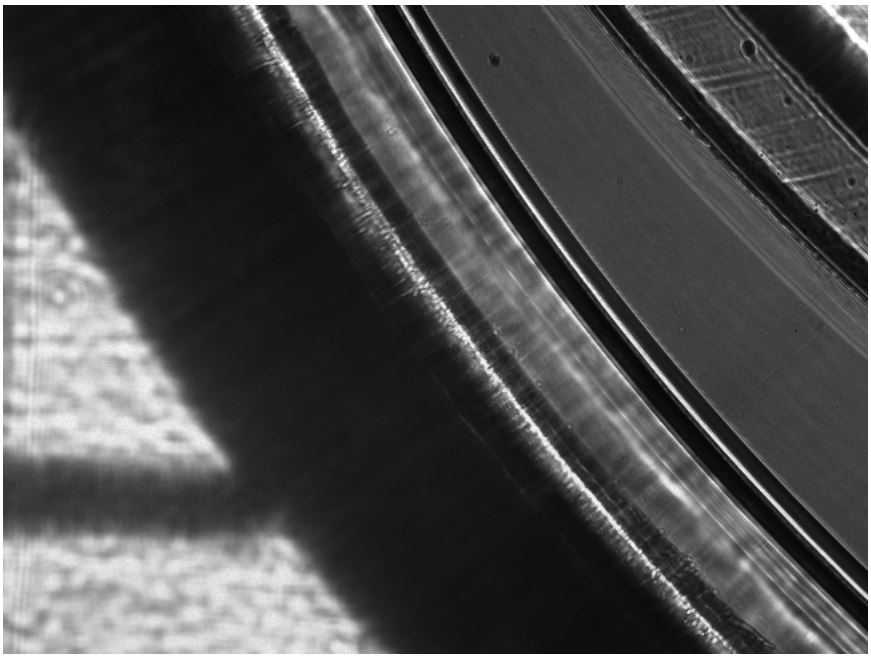
```
import cv2
```

```
image = cv2.imread('2024-08-13_08-25-25.805031_6b57d819-a9d1-4f46-88b0-1581d0de9316.png', -1)
```

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

Mostrar Imagen Original

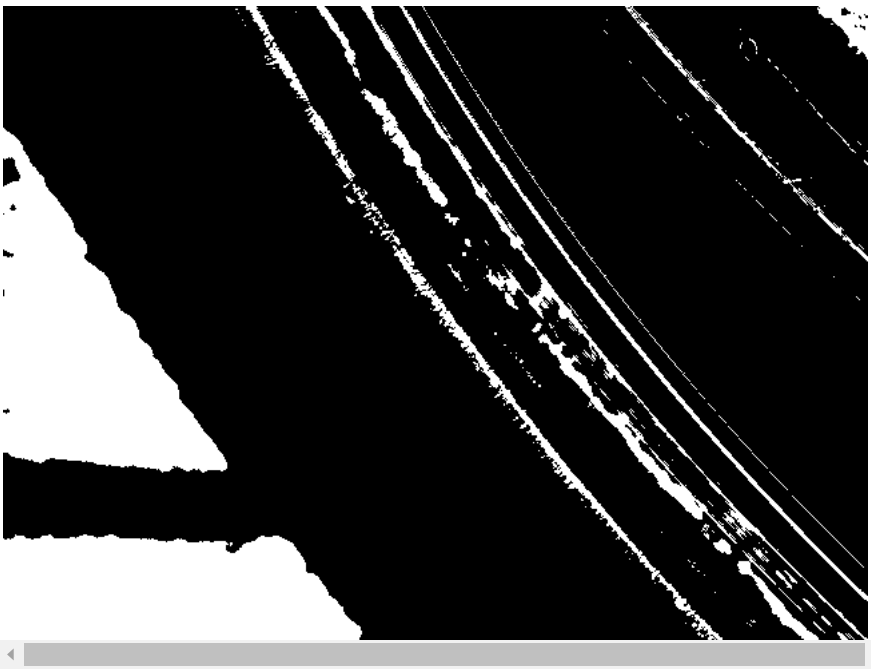
```
# !pip install opencv-python-headless

import cv2
from google.colab.patches import cv2_imshow

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



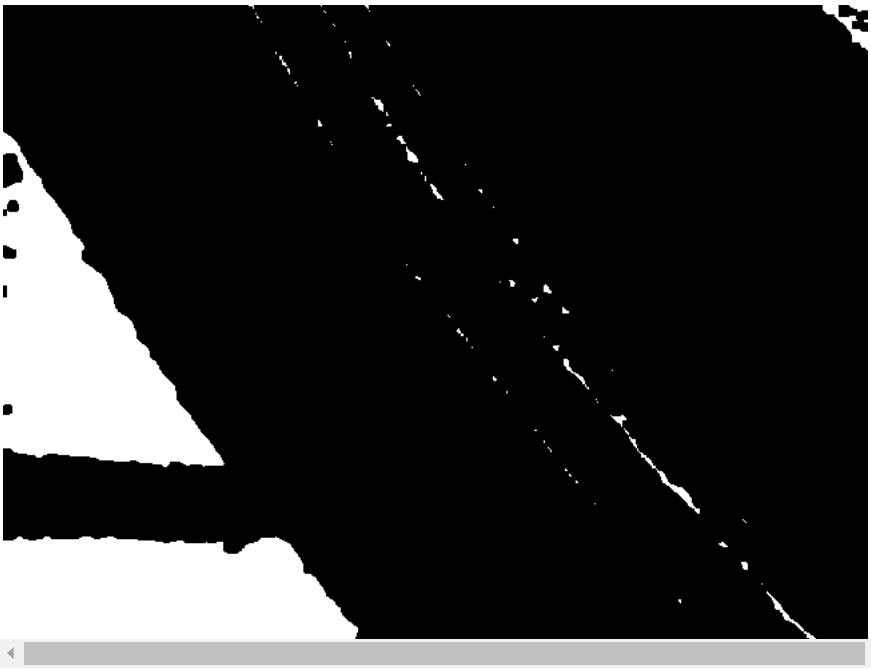
Redimensionzar la imagen y aplicaar threshold

```
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('2024-08-13_08-25-25.805031_6b57d819-a9d1-4f46-88b0-1581d0de9316.png', -1)

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

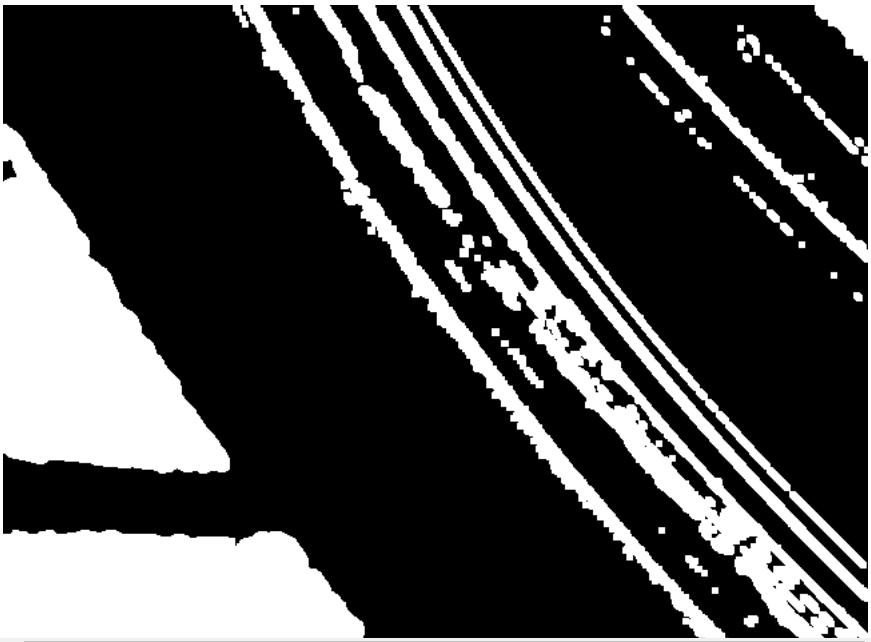
```
    img_resized = cv2.resize(image, None, fx=0.25, fy=0.25)

    # 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")
```