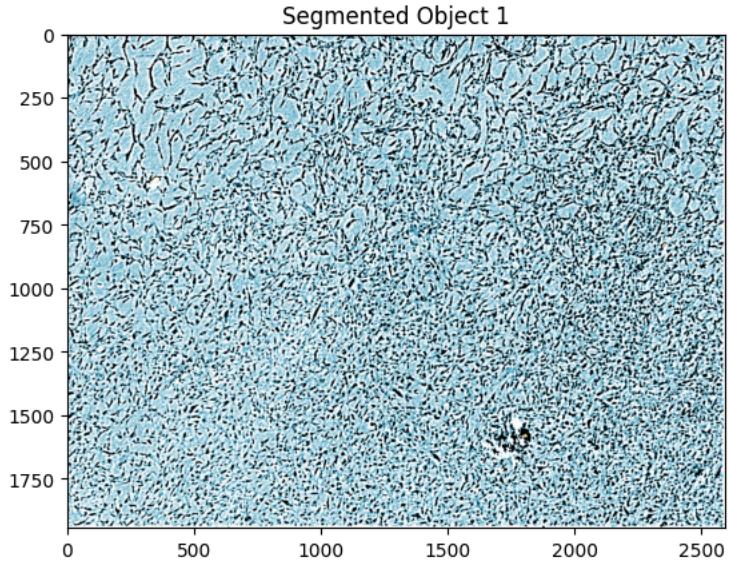
```
import cv2
import numpy as np
import os
import matplotlib.pyplot as plt
from scipy.stats import norm


from google.colab import files
uploaded = files.upload()
folder_path = '/content/temp_images'
os.makedirs(folder_path, exist_ok=True)

# Guardar las imágenes subidas en esa carpeta
for image_name, image_data in uploaded.items():
    with open(os.path.join(folder_path, image_name), 'wb') as f:
        f.write(image_data)
```

```
Elegir archivos Sin archivos seleccionados Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving _floorMask.png to _floorMask (1).png
Saving _Phi8_norm.png to _Phi8_norm (1).png
Saving _Phi8Color.png to _Phi8Color (1).png
Saving _validMask.png to _validMask (1).png
Saving d62536a3-1443-496e-a189-0c39acfa33eb_B1_02_05_floorMask.png to d62536a3-1443-496e-a189-0c39acfa33eb_B1_02_05_floorMas
Saving d62536a3-1443-496e-a189-0c39acfa33eb_B1_02_05_Phi8_norm.png to d62536a3-1443-496e-a189-0c39acfa33eb_B1_02_05_Phi8_nor
Saving d62536a3-1443-496e-a189-0c39acfa33eb_B1_02_05_Phi8.png to d62536a3-1443-496e-a189-0c39acfa33eb_B1_02_05_Phi8 (1).png
Saving d62536a3-1443-496e-a189-0c39acfa33eb_B1_02_05_Phi8Color.png to d62536a3-1443-496e-a189-0c39acfa33eb_B1_02_05_Phi8Colo
Saving d62536a3-1443-496e-a189-0c39acfa33eb_B1_02_05_validMask.png to d62536a3-1443-496e-a189-0c39acfa33eb_B1_02_05_validMas
```

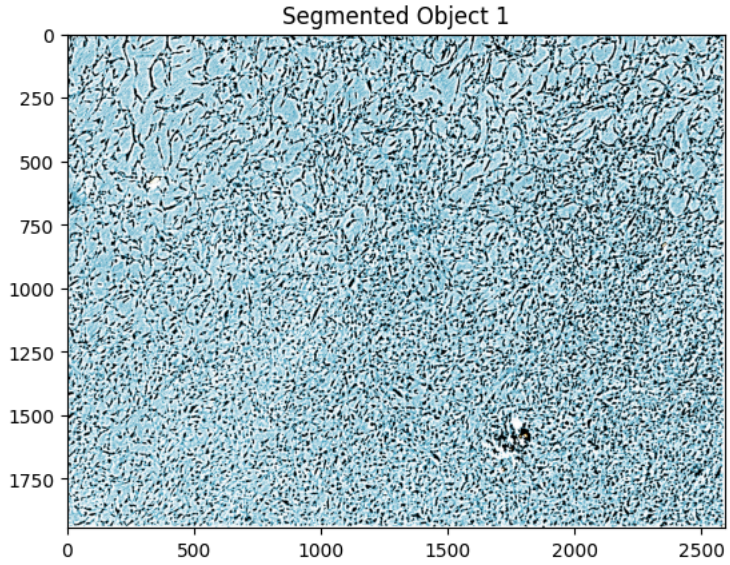
```
def process_image_pipeline(folder_path):
    images = [f for f in os.listdir(folder_path) if f.endswith(('.png', '.jpg', '.jpeg'))]
    images.sort()

    if not images:
        print("No se encontraron imágenes en la carpeta.")
        return

    print(f"Se encontraron {len(images)} imágenes.")

    widths = []
    heights = []

    for image_file in images:
        print(f"Procesando imagen: {image_file}")
        image_path = os.path.join(folder_path, image_file)
        original_image = cv2.imread(image_path)

        if original_image is None:
            print(f"No se pudo cargar la imagen {image_file}")
            continue

        grayscale_image = cv2.cvtColor(original_image, cv2.COLOR_BGR2GRAY)

        #binarizacion por thresholding
        _, binary_image = cv2.threshold(grayscale_image, 127, 255, cv2.THRESH_BINARY)

        #operaciones morfológicas
        kernel = np.ones((3, 3), np.uint8)
        binary_image = cv2.morphologyEx(binary_image, cv2.MORPH_CLOSE, kernel)

        #identificar blobs
        num_labels, labels, stats, centroids = cv2.connectedComponentsWithStats(binary_image)

        print(f"Se detectaron {num_labels} blobs en la imagen.")

        #informacion de blobs
        for i in range(1, num_labels):  #ignoramos el fondo
            x, y, w, h, area = stats[i]
            if area > 100:  #filtrar pequeños blobs
                widths.append(w)
                heights.append(h)
                #crear mascara binaria para este blob
                mask = np.zeros_like(grayscale_image)
                mask[labels == i] = 255

                #aplicar mascara a la original
                masked_image = cv2.bitwise_and(original_image, original_image, mask=mask)
```

```
                # Mostrar la imagen segmentada
                plt.imshow(cv2.cvtColor(masked_image, cv2.COLOR_BGR2RGB))
                plt.title(f"Segmented Object {i}")
                plt.show()

                 #analisis de ancho y alto (width and height)
    analyze_blob_distributions(widths, heights)
    return widths, heights


def analyze_blob_distributions(widths, heights):
    #calcular caracteristicas
    mean_width = np.mean(widths)
    std_width = np.std(widths)
    mean_height = np.mean(heights)
    std_height = np.std(heights)

    print(f"Mean Width: {mean_width}, Std Width: {std_width}")
    print(f"Mean Height: {mean_height}, Std Height: {std_height}")

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

    plt.subplot(1, 2, 1)
    plt.hist(widths, bins=20, color='blue', alpha=0.7, density=True)
    xmin, xmax = plt.xlim()
    x = np.linspace(xmin, xmax, 100)
    p = norm.pdf(x, mean_width, std_width)
    plt.plot(x, p, 'k', linewidth=2)
    plt.title("Width Distribution")

    plt.subplot(1, 2, 2)
    plt.hist(heights, bins=20, color='green', alpha=0.7, density=True)
    xmin, xmax = plt.xlim()
    x = np.linspace(xmin, xmax, 100)
    p = norm.pdf(x, mean_height, std_height)
    plt.plot(x, p, 'k', linewidth=2)
    plt.title("Height Distribution")

    plt.show()


process_image_pipeline('/content/')
```

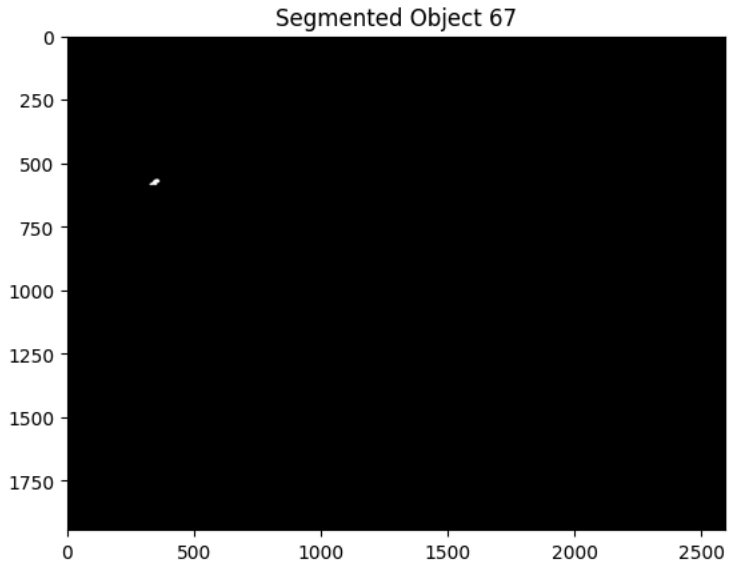
```
Se encontraron 18 imágenes.
Procesando imagen: _Phi8Color (1).png
Se detectaron 96 blobs en la imagen.
```

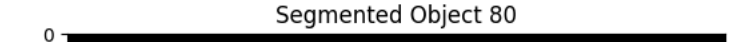


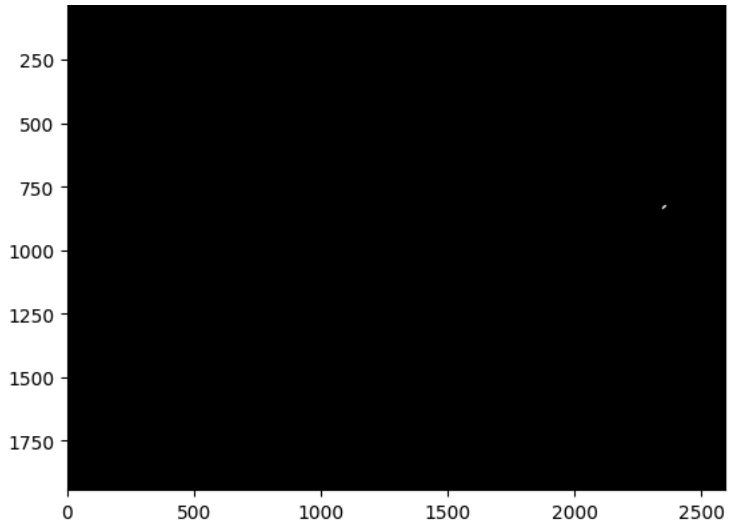

```
Procesando imagen: _Phi8Color.png
Se detectaron 96 blobs en la imagen.
```



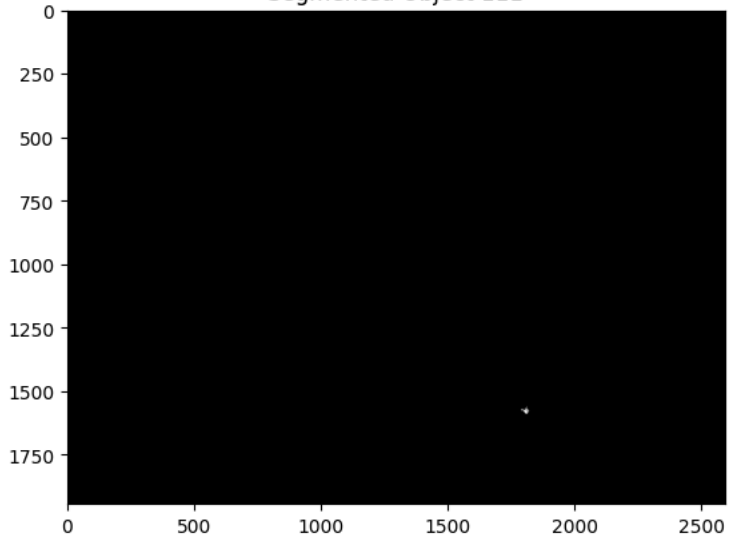

```
Procesando imagen: _Phi8_norm (1).png
Se detectaron 137 blobs en la imagen.
```



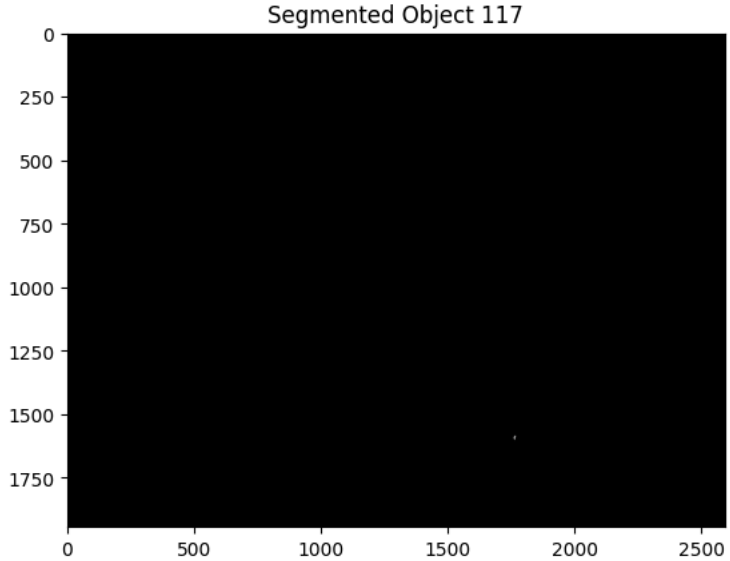




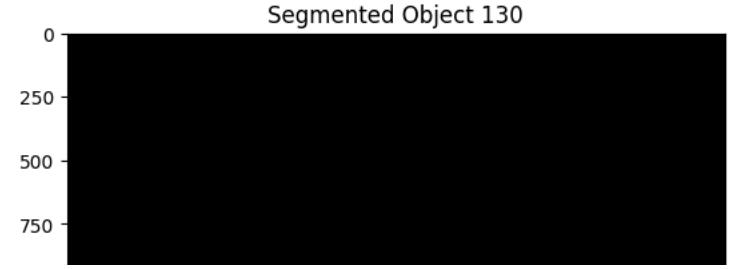
Segmented Object 111

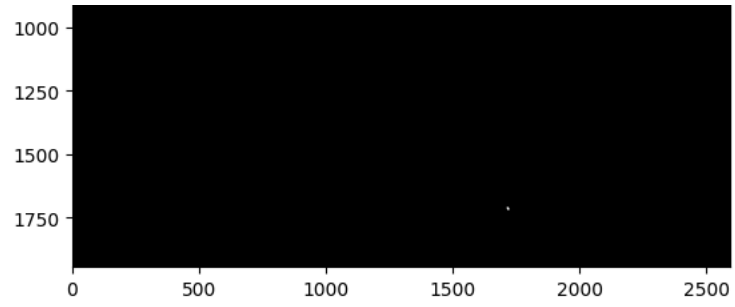


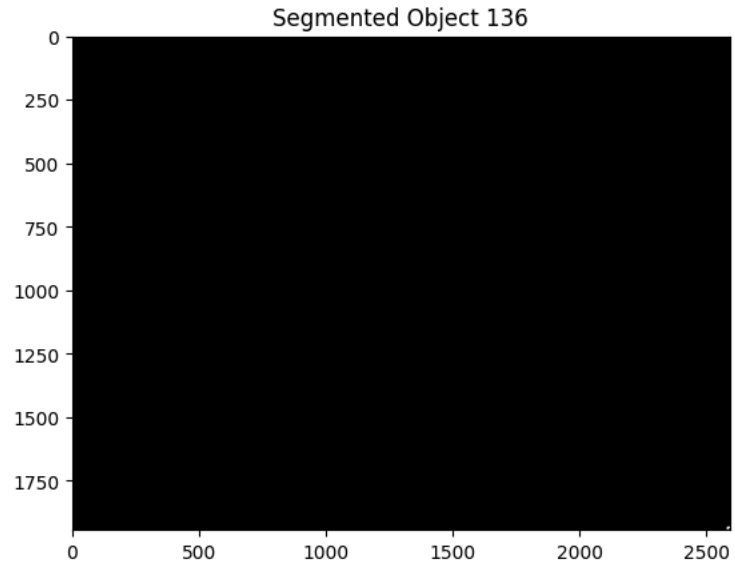
Segmented Object 117



Segmented Object 130








```
Procesando imagen: _Phi8_norm.png
Se detectaron 137 blobs en la imagen.
```