Segmentación de imágenes

October 13, 2024

Instrucciones

El objetivo | Crear un pipeline que a partir de las imágenes de reto obtenga información relevante de las zonas a identificar, así como algunas características generales para su modelado.

Pueden utilizar los resultados del ejercicio "Introducción al procesamiento de imágenes" Obtenga la segmentación de objetos de interés en las imágenes de reto. 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. 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.

[22]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import os
```

[23]:
```
def plot_steps(binary_image, refined_mask, contours, image_with_contours):
    plt.figure(figsize=(12, 6))

    plt.subplot(1, 3, 1)
    plt.imshow(binary_image, cmap='gray')
    plt.title('Imagen binarizada')

    plt.subplot(1, 3, 2)
    plt.imshow(refined_mask, cmap='gray')
    plt.title('Máscara refinada')

    plt.subplot(1, 3, 3)
    plt.imshow(image_with_contours)
    plt.title('Contornos detectados')

    plt.tight_layout()
    plt.show()
```

[24]:
```
def process_image(image_path):
    binary_image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
```

```
    # Paso 1: Refinar la máscara binaria usando operaciones morfológicas
    kernel = np.ones((5, 5), np.uint8)
    refined_mask = cv2.morphologyEx(binary_image, cv2.MORPH_CLOSE, kernel)

    # Paso 2: Eliminar ruido pequeño con un filtro de contornos pequeños
    min_contour_area = 50  # Ajusta este valor según sea necesario
    contours, _ = cv2.findContours(refined_mask, cv2.RETR_EXTERNAL, cv2.
↪CHAIN_APPROX_SIMPLE)

    # Crear una copia de la imagen original para dibujar los contornos
    image_with_contours = np.zeros_like(binary_image)

    # Inicializar listas para almacenar los tamaños (anchos y altos)
    widths = []
    heights = []

    # Iterar sobre los contornos para extraer los tamaños y filtrar contornos␣
↪pequeños
    for contour in contours:
        area = cv2.contourArea(contour)
        if area >= min_contour_area:  # Filtrar contornos pequeños
            x, y, w, h = cv2.boundingRect(contour)
            widths.append(w)
            heights.append(h)
            cv2.drawContours(image_with_contours, [contour], -1, (255, 255,␣
↪255), 2)

    # Graficar el resultado de la máscara refinada y los contornos
    plot_steps(binary_image, refined_mask, contours, image_with_contours)

    return widths, heights
```

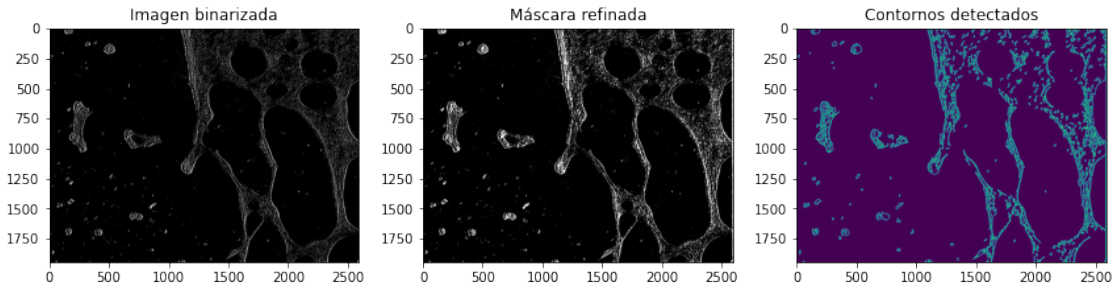
[25]:
```
def process_images_in_directory(directory_path):
    all_widths = []
    all_heights = []


    for filename in os.listdir(directory_path):
        if filename.endswith(".jpg") or filename.endswith(".png"):
            image_path = os.path.join(directory_path, filename)
            print(f"Procesando imagen: {image_path}")

            widths, heights = process_image(image_path)
            all_widths.extend(widths)
            all_heights.extend(heights)

    return all_widths, all_heights
```

[26]:
```
def plot_distributions(widths, heights):
    #  distribuciones de ancho y alto
    plt.figure(figsize=(12, 6))

    plt.subplot(1, 2, 1)
    plt.hist(widths, bins=20, color='blue', alpha=0.7)
    plt.title('Distribución de Anchos')
    plt.xlabel('Ancho')
    plt.ylabel('Frecuencia')

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

    plt.tight_layout()
    plt.show()
```

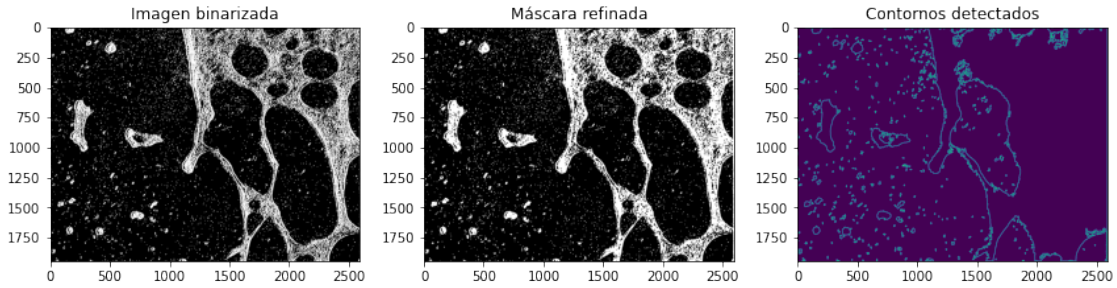
[27]:
```
# Función principal
def main():
    directory_path = 'saved_images'
    all_widths, all_heights = process_images_in_directory(directory_path)

    # Estadisticas
    print(f"Ancho promedio: {np.mean(all_widths):.2f}, Desviación estándar: {np.
 ↪std(all_widths):.2f}")
    print(f"Alto promedio: {np.mean(all_heights):.2f}, Desviación estándar: {np.
 ↪std(all_heights):.2f}")

    plot_distributions(all_widths, all_heights)
```

[28]:
```
main()
```

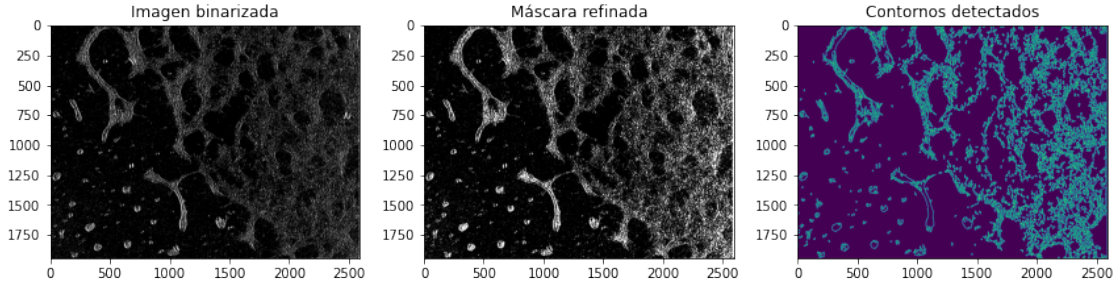
```
Procesando imagen: saved_images\0.28890384546380754binary_image.png
```



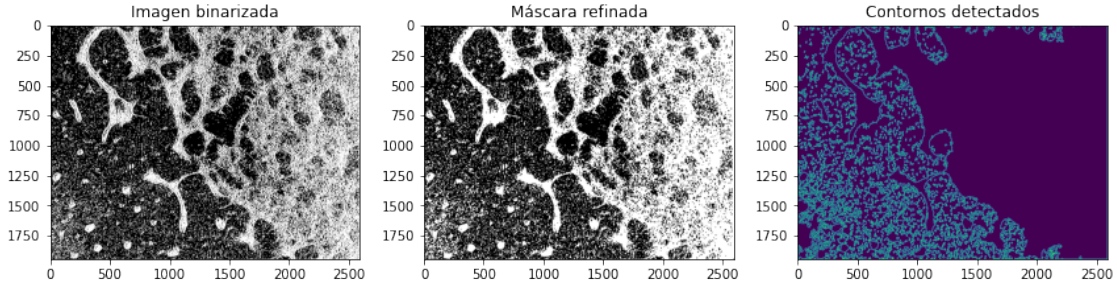

```
Procesando imagen: saved_images\0.28890384546380754dilated_image.png
```



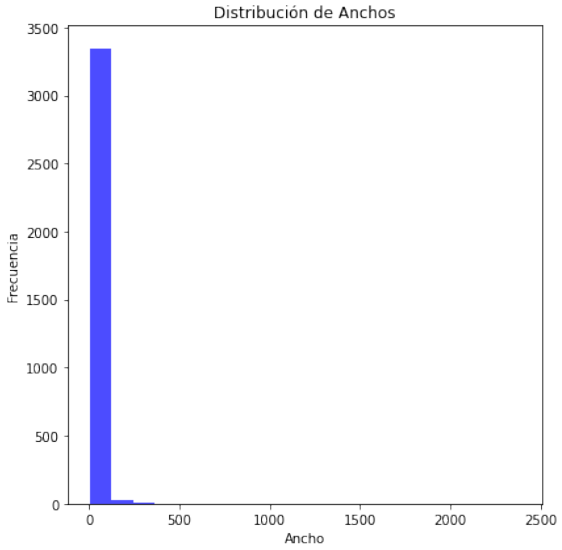
Procesando imagen: saved_images\0.47512360688417155binary_image.png

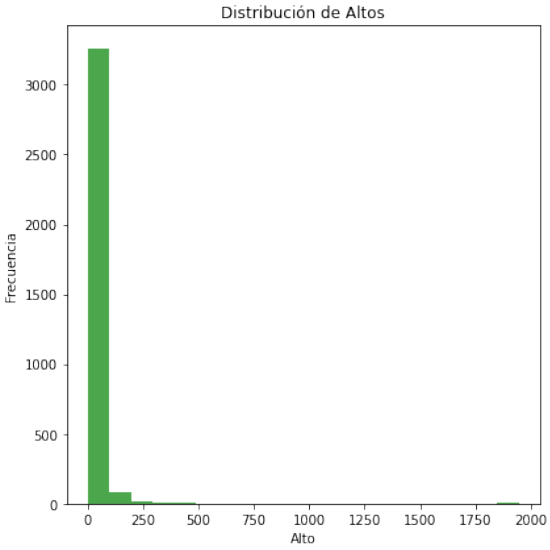


Procesando imagen: saved_images\0.47512360688417155dilated_image.png



Ancho promedio: 23.76, Desviación estándar: 53.88
Alto promedio: 34.50, Desviación estándar: 78.37


Distribución de Anchos
Frecuencia
Ancho
0
500
1000
1500
2000
2500
3000
3500


Distribución de Altos
Frecuencia
Alto
0
250
500
750
1000
1250
1500
1750
2000
2500
3000