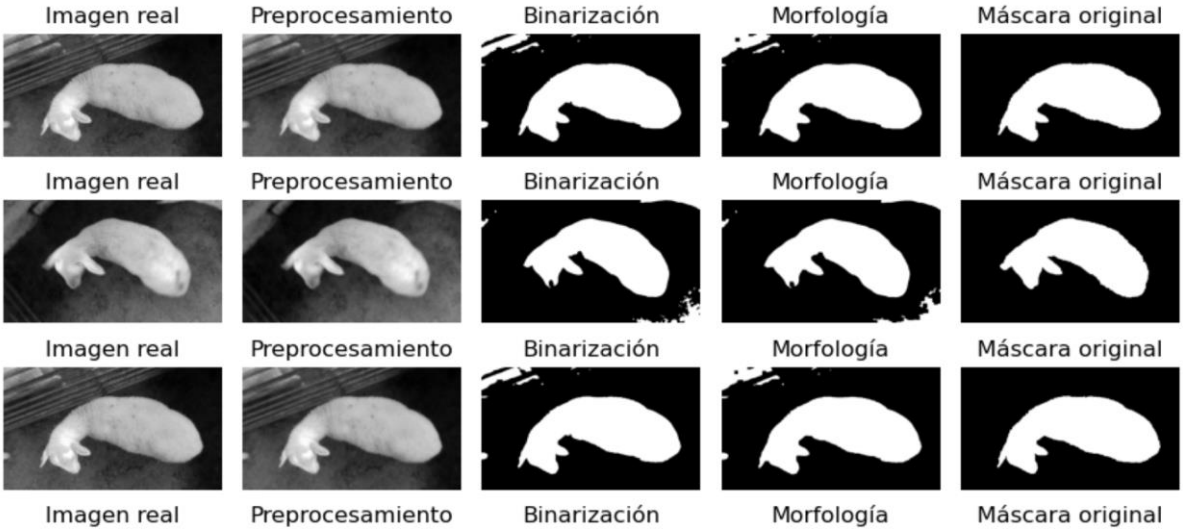
SEGMENTACIÓN DE OBJETOS



Aitor García Blanco

Índice

1.Introducción

En este proyecto se aborda el problema de la segmentación de objetos a través de técnicas clásicas de visión por computador, con el fin de delimitar e identificar objetos específicos dentro de una imagen. Para esto, se van a segmentar un conjunto de imágenes de corderos empleando las diferentes técnicas de procesamiento de imágenes vistas en la asignatura y expuestas posteriormente a modo resumen. El objetivo principal de la práctica es conseguir la mejor configuración de procesamiento para obtener las máscaras precisas que representan la ubicación de los corderos en las imágenes.

El proceso de segmentación incluye un preprocesamiento de la imagen para mejorar la calidad de esta, binarización para crear una representación en blanco y negro y, operaciones morfológicas para refinar las máscaras resultantes. Además, se evalúan las diferentes configuraciones de procesamiento, comparando sus resultados con las máscaras originales proporcionadas en el conjunto de datos.

En el siguiente repositorio se encuentra el proyecto en Python para esta práctica:

ULE-Informatica-2024-2025/vico24-AigarciabFabero: vico24-AigarciabFabero created by GitHub Classroom

2.Descripción del dataset

El dataset que se emplea para la segmentación de objetos, se puede encontrar en el siguiente repositorio: ULE-Informatica-2024-2025/VICO_lamb: Conjunto de datos de segmentación de corderos.

En el conjunto de datos de segmentación de Lamb, tenemos un conjunto de 24 imágenes (de corderos) a color y sus correspondientes máscaras.

Tal y como se analizan las imágenes en el archivo jupyter, podemos concluir que la dimensión de estas es de 848x480.

Dentro del estudio del dataset, se representan los histogramas para el conjunto de imágenes a color. Por un lado, en lecturas en escala de grises y por otro en escala de color; tal y como se puede ver a continuación.

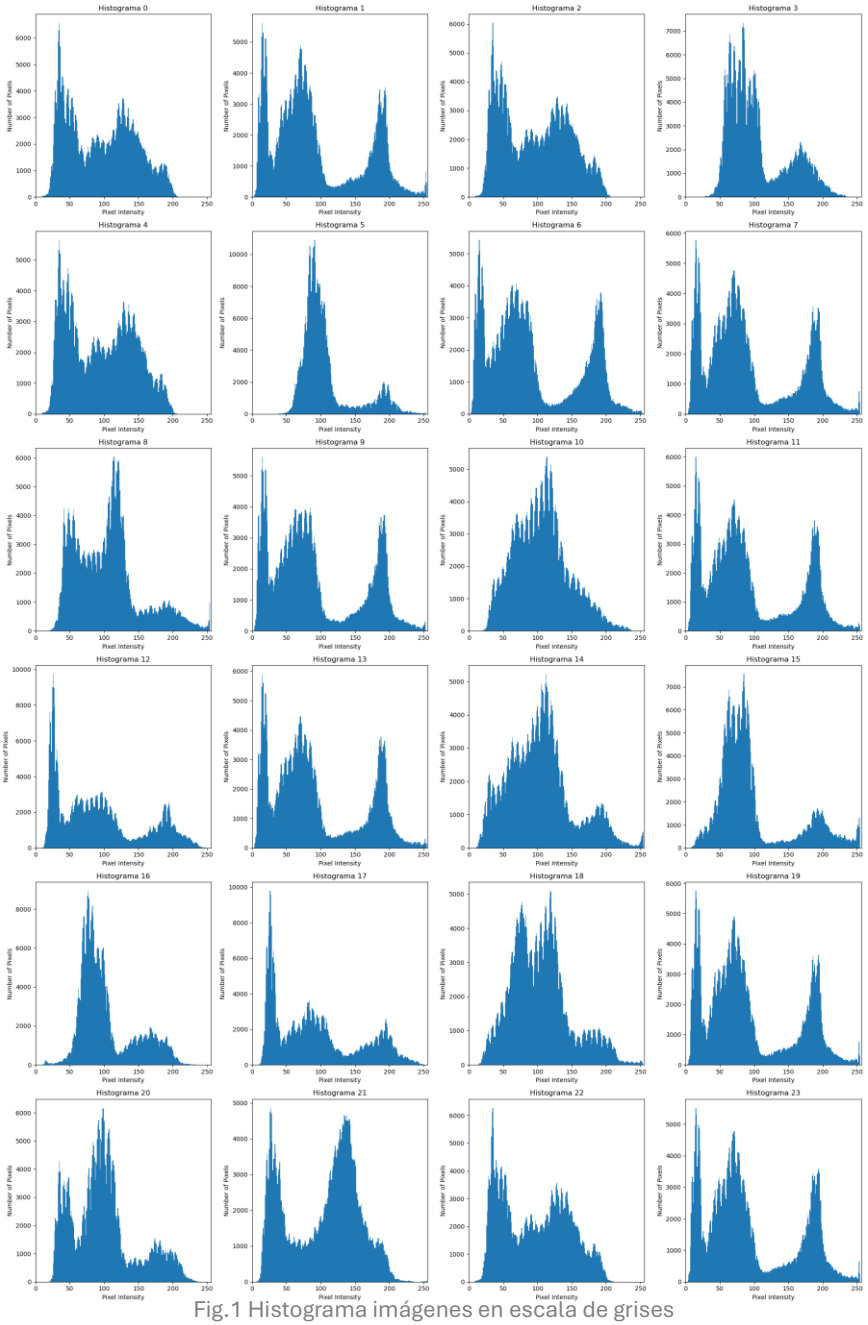


Fig.1 Histograma imágenes en escala de grises

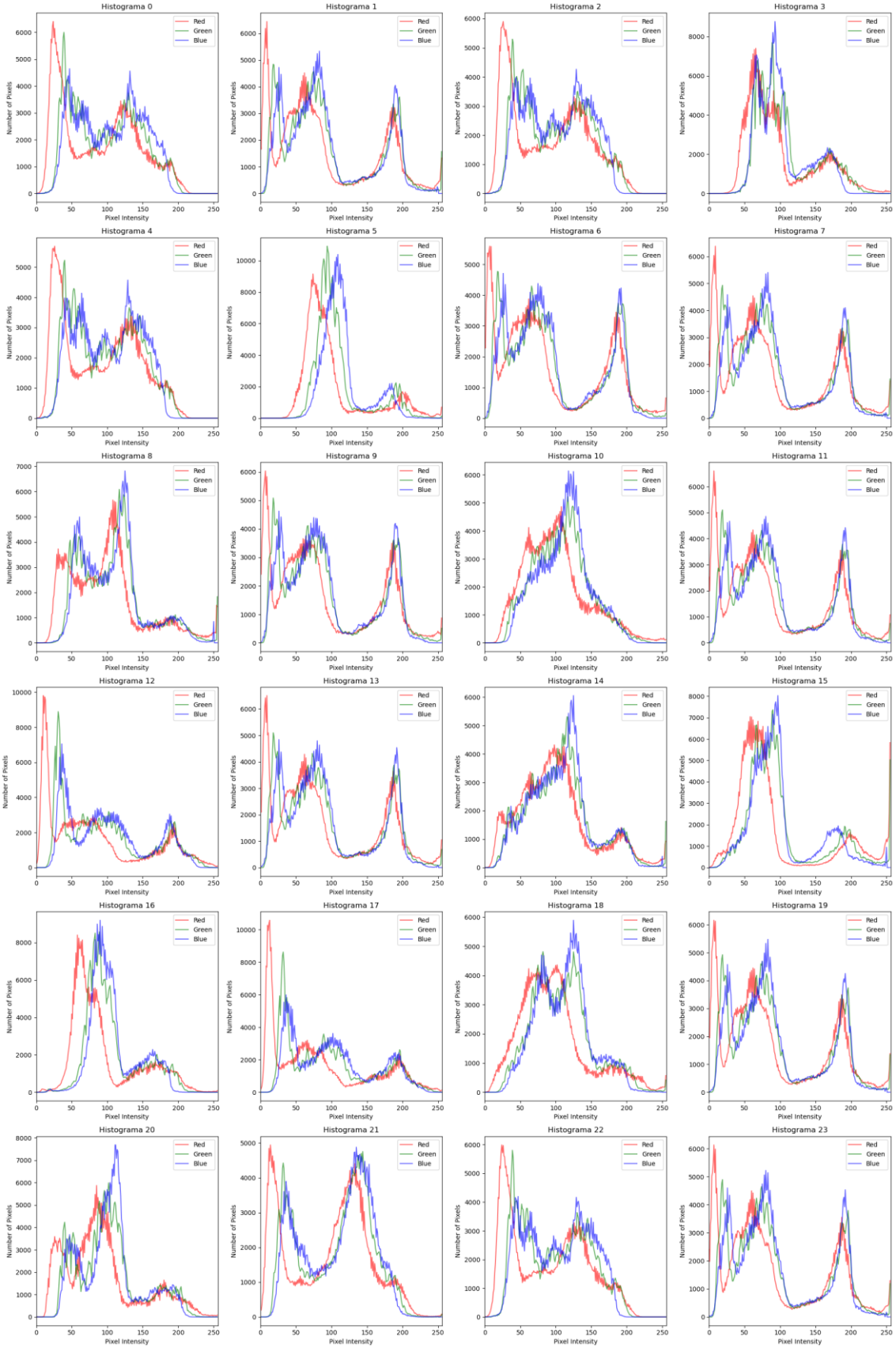


Fig.2 Histograma imágenes en escala de color

3.Fundamentos teóricos

El proceso de segmentación de los corderos en imágenes implica tres pasos principales: preprocesamiento, binarización y operaciones morfológicas. Cada uno de estos pasos se basa en fundamentos teóricos que permiten manipular y analizar las imágenes eficazmente.

A continuación, mostramos tres tablas a nivel aclarativo para conocer más en detalle cada uno de los diferentes procedimientos.

Para el preprocesamiento:

Operación	Nivel	Lineal	Tipo	Uso Principal
Ecualización de Histograma	Píxel	No	-	Mejora el contraste redistribuyendo los niveles de intensidad de manera uniforme
CLAHE	Vecino	No	-	Mejora el contraste localmente evitando la sobreexposición
Filtro de Caja	Vecino	Sí	Pasa-bajos	Suaviza la imagen, reduce el ruido eliminando detalles finos
Filtro Gaussiano	Vecino	Sí	Pasa-bajos	Suaviza la imagen, reduce el ruido preservando las estructuras de borde
Filtro Sobel	Vecino	Sí	Pasa-altos	Detecta bordes destacando cambios de intensidad en direcciones específicas (x o y)
Filtro Laplaciano	Vecino	Sí	Pasa-altos	Detecta bordes, mejora las transiciones rápidas de intensidad en todas direcciones
Filtro de Enfoque	Vecino	No	Pasa-altos	Mejora la nitidez destacando bordes y detalles
Filtro Mediana	Vecino	No	Pasa-bajos	Elimina ruido tipo *salt-pepper* preservando los bordes
Filtro Bilateral	Vecino	No	Pasa-bajos	Suaviza la imagen preservando los bordes, útil para reducción de ruido

Fig.3 Resumen procesos de preprocesamiento

Binarización:

Proceso	Descripción	Código de Ejemplo
Binarización Global	Divide la imagen en dos regiones: blanco y negro, utilizando un único valor de umbral. Es útil cuando la imagen tiene una iluminación uniforme, pero puede no ser adecuado para imágenes con iluminación variable.	`th, binary_mask = cv2.threshold(blurred, 118, 255, cv2.THRESH_BINARY)`
Umbralización Adaptativa	Calcula el umbral para cada píxel en función de una pequeña región a su alrededor. Se adapta a las condiciones de iluminación locales y produce mejores resultados para imágenes con iluminación variable. Se puede generar un umbral mediante la media del área vecina menos una constante (C), o una suma ponderada gaussiana de los valores del vecindario menos C.	`binary_mask = cv2.adaptiveThreshold(blurred,255,cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY,15,5) # binary_mask = cv2.adaptiveThreshold(blurred,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY,15,5)`
Método de Otsu	Encuentra el valor de umbral que minimiza la varianza ponderada dentro de la clase. Es eficaz para separar objetos del fondo cuando hay una clara diferencia en los niveles de intensidad.	`th, binary_mask = cv2.threshold(blurred, 118, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)`

Fig.4 Resumen procesos de binarización

Operaciones morfológicas:

Operación Morfológica	Descripción	Código de Ejemplo
Dilatación	Expande las regiones blancas en una imagen binaria. Aumenta el tamaño de los objetos y conecta regiones cercanas.	`binary_mask_morph = cv2.dilate(binary_mask, kernel, iterations=1)`
Erosión	Reduce las regiones blancas en una imagen binaria. Disminuye el tamaño de los objetos y elimina detalles pequeños o ruido.	`binary_mask_morph = cv2.erode(binary_mask, kernel, iterations=1)`
Apertura	Una erosión seguida de una dilatación con el mismo elemento estructurante. Elimina el ruido y los detalles pequeños, al tiempo que conserva la forma general de los objetos.	`binary_mask_morph = cv2.morphologyEx(binary_mask, cv2.MORPH_OPEN, kernel)`
Cierre	Una dilatación seguida de una erosión con el mismo elemento estructurante. Cierra pequeños agujeros en los objetos y suaviza los límites.	`binary_mask_morph = cv2.morphologyEx(binary_mask, cv2.MORPH_CLOSE, kernel)`
Gradiente Morfológico	La diferencia entre la imagen dilatada y la imagen erosionada. Resalta los bordes de los objetos.	`binary_mask_morph = cv2.subtract(binary_mask, cv2.erode(binary_mask, kernel, iterations=1))`
Umbralización	Aunque no es una operación morfológica en sí misma, a menudo se utiliza para binarizar imágenes antes de aplicar operaciones morfológicas. Se pueden utilizar métodos de umbralización global, adaptativa o el método de Otsu para binarizar la imagen.	`_, binary_mask_morph = cv2.threshold(binary_mask, 127, 255, cv2.THRESH_BINARY)`

Fig.5 Resumen operaciones morfológicas

A lo largo de los diferentes procesamientos, se ha seguido una estructura uniforme modular que se puede ver en el repositorio.

4.Primer procesamiento

En este primer procesamiento, se ha seleccionado el filtro bilateral con el fin de suavizar la imagen sin difundir los bordes, lo cual es crucial para tener una segmentación precisa (diámetro de área de los pixeles a considerar a filtrar = 9, la siga espacial y la sigma de intensidad toma un valor de 75; cuanto mayor es este, mayor suavizado sobre la imagen e intensidad). A nivel binarización, optamos por una binarización global basada en la técnica OSU que nos permite umbralizar automáticamente el umbral óptimo para separar los pixeles en dos clases (blanco y negro). Por último, con el fin de mejorar esta máscara binaria que obtenemos tras el proceso de binarización, se emplean las operaciones morfológicas de cierre (closing, con kernel de matrices de 14x14) para intentar cerrar pequeños agujeros dentro de los objetos y unir objetos cercanos. Las diferentes configuraciones de los parámetros para los diferentes procesos se recogen en el repositorio.

El resultado de este procedimiento para las diferentes métricas:

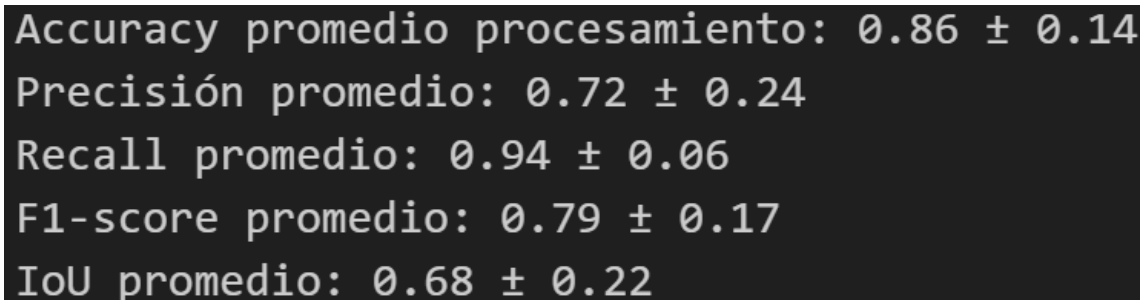


Fig.6 Métricas primer procesamiento

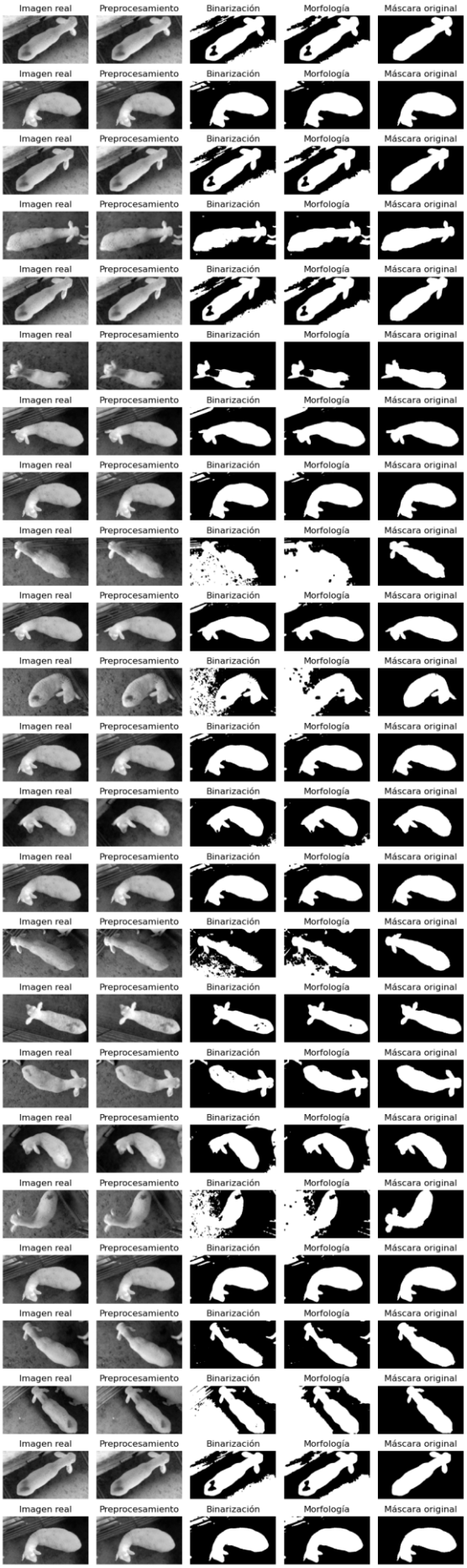


Fig.7 Resultados primer procesamiento

5.Segundo procesamiento

En este segundo procedimiento se ha empleado a nivel preprocesamiento un filtro 2D para suavizar la imagen (El kernel empleado es un filtro de suavizado que da más peso a los píxeles centrales y menos a los píxeles periféricos “np.array([[1/16, 2/16, 1/16], [2/16, 4/16, 2/16], [1/16, 2/16, 1/16]])”). En el proceso de binarización, se opta por una binarización global, pero en este caso, no empleamos la técnica OSU sino un umbral fijo (118 como parámetro). Para mejorar la mascara binaria se usan operaciones morfológicas (con kernel 15x15) mediante técnicas de tipo abierto que permiten eliminar pequeños objetos y ruido.

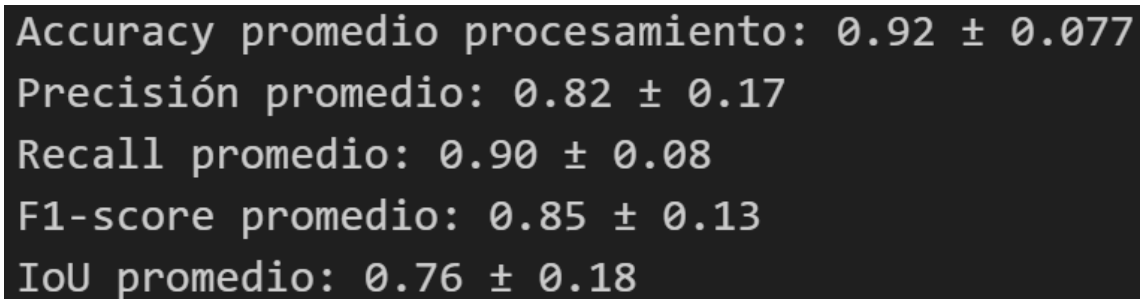


Fig.8 Métricas segundo procesamiento

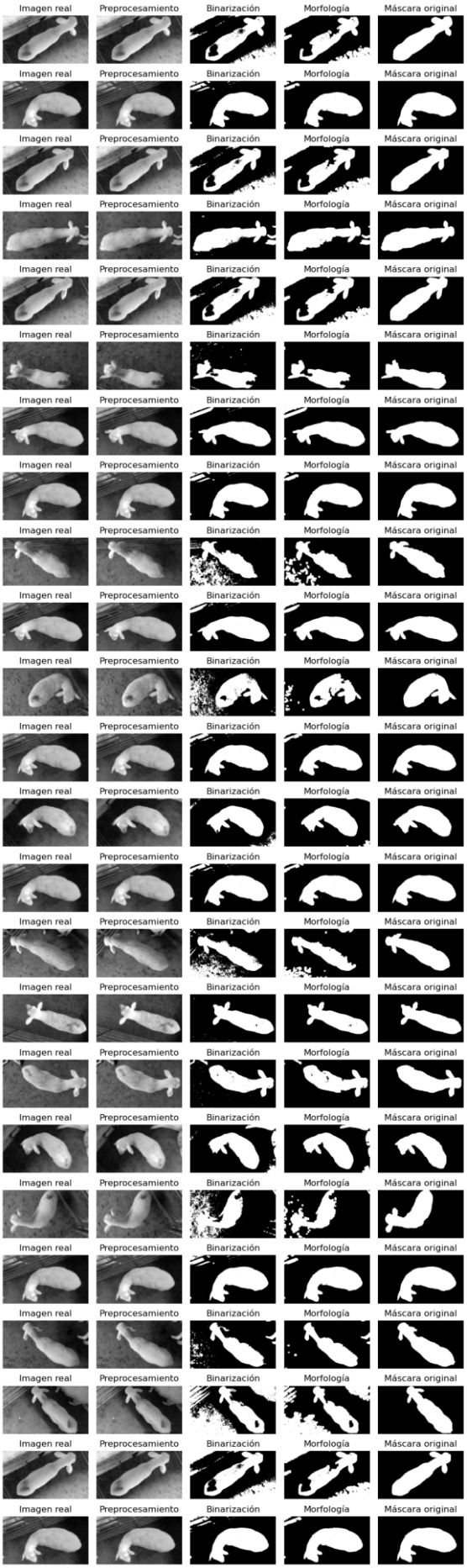


Fig.9 Resultados segundo procesamiento

6.Tercer procesamiento

En este segundo procedimiento se ha empleado a nivel preprocesamiento un filtro de mediana para suavizar la imagen (el número de pixeles que se considera para calcular la mediana se establece en 7). En el proceso de binarización optamos por una binarización global, con un umbral fijo (118). Para mejorar la máscara binaria, se emplean operaciones morfológicas mediante técnicas de tipo abierto que permiten eliminar pequeños objetos y ruido (kernel 14x14).

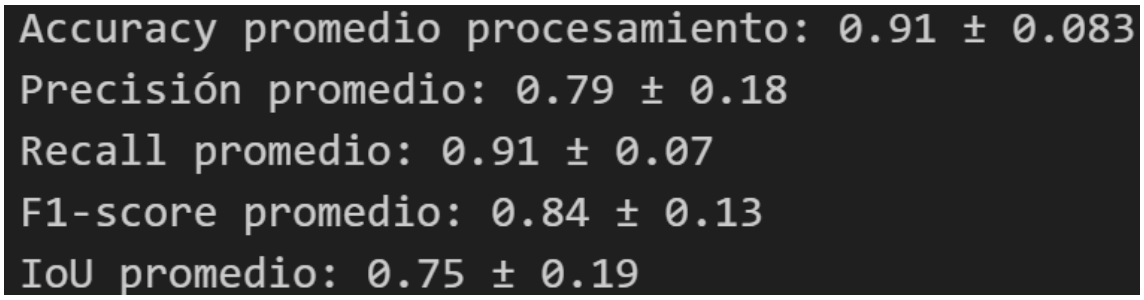


Fig.10 Métricas tercer procesamiento

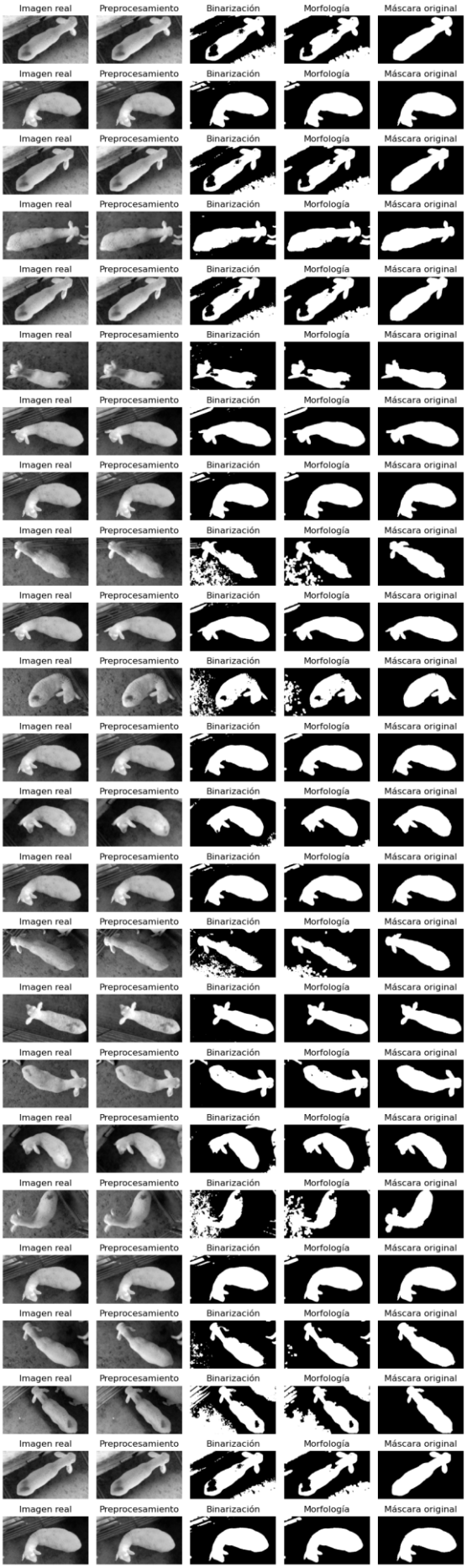


Fig.11 Resultados tercer procesamiento

7.Conclusión

En los tres procedimientos se han obtenido resultados muy similares si nos basamos en todas las métricas empleadas. Además, somos conscientes de la dificultad a la hora de procesar algunas imágenes debido al ruido de la imagen o la fusión del objeto a segmentar con el fondo debido a la aproximación de colores. Por tanto, podemos concluir que los resultados obtenidos para los diferentes procedimientos son más que satisfactorios.

Cabe destacar que no se ha empleado a nivel preprocesamiento ningún filtro de tipo “pasa alta”, esto se debe a que se centraría en los bordes o cambios de intensidad y no en las imágenes como tal; dando lugar a muchos falsos negativos.