ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

IDENTIFICACIÓN DE PARÁSITOS DE REPTILES USANDO TÉCNICAS DE PROCESAMIENTO DE IMÁGENES, VISIÓN POR COMPUTADOR Y APRENDIZAJE AUTOMÁTICO

SEGMENTACIÓN DE IMÁGENES DE PARÁSITOS DE REPTILES MEDIANTE TÉCNICAS DE PROCESAMIENTO DIGITAL DE IMÁGENES.

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TELECOMUNICACIONES

MARJORIE CHUULI RAMOS ANDAGOYA

marjorie.ramos@epn.edu.ec

DIRECTOR: Ing. VIVIANA CRISTINA PÁRRAGA VILLAMAR

viviana.parragav@epn.edu.ec

DMQ, agosto 2023

CERTIFICACIONES

Yo, Marjorie Chuuli Ramos Andagoya declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

MARJORIE CHUULI RAMOS ANDAGOYA

Certifico que el presente trabajo de integración curricular fue desarrollado por Marjorie Chuuli Ramos Andagoya, bajo mi supervisión.

ING. VIVIANA CRISTINA PÁRRAGA VILLAMAR
DIRECTORA

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

MARJORIE CHUULI RAMOS ANDAGOYA

ING. VIVIANA CRISTINA PÁRRAGA VILLAMAR

DEDICATORIA

Dedico este proyecto

A mi madre Carmen que en paz descanse que me cuida desde el cielo.

A mi padre Hugo y mis hermanos Ronald y Francisco que han sido mi pilar, permaneciendo a mi lado a cada instante.

Sin ellos nada sería posible.

A mis amigos.

AGRADECIMIENTO

A toda mi familia que pesar de la distancia he contado con su apoyo incondicional y me han forjado como persona.

A mis amigos universitarios que hemos compartido tantos años, esperando siempre la superación personal y académica de cada uno de ellos.

A mis mejores amigos que han estados en las buenas y en las malas especialmente Josselyn que ha pesar de los años desde el colegio nuestra amistad a perdurado con el tiempo, tal como Stefano, Jairo, Ítalo, Vanessa, Jessica y Maryori.

A mi primo más cercano Ramiro que fue el que me alentó a seguir adelante creyendo en mi siempre.

A mis mentores y maestros quienes a más de los conocimientos técnicos me enseñaron lecciones de vida.

Finalmente, a Viviana Párraga quien fue mi mentora con cariño y paciencia con quien realice este arduo trabajo.

ÍNDICE DE CONTENIDO

RESUMEN

El presente trabajo de integración implementa la segmentación de imágenes de parásitos de reptiles utilizando el software MATLAB mediante tres técnicas de segmentación. Inicialmente se realizó la extracción y análisis de la máscara de cada una de las imágenes, que se obtuvieron mediante una gran base de datos del proyecto de investigación "Identificación de parásitos con diferentes métodos coprológicos en muestras de reptiles en el Vivarium de Quito", que cuenta con 3616 Imágenes, su objetivo fue distinguir los parásitos gastroentéricos de los reptiles.

La herramienta Image Labeler, que se utilizó, fue añadida al catálogo de MATLAB en la versión 2022b, la cual facilita la adquisición de etiquetas que marcan o subrayan la región de interés (ROI). Con las máscaras correspondientes de cada imagen se dividió en dos grupos: 70% de entrenamiento que son aproximadamente 2532 imágenes y 30% de validación. Estas imágenes se escogen de forma aleatoria, es decir cada vez que se mande a correr el programa y/o código estás cambian.

Las técnicas implementadas fueron Otsuthresh, Activecontour e Imsegfmm que trabajan con imágenes 2D y cumplen con el propósito del entrenamiento de imágenes. Con las imágenes de entrenamiento y la aplicación de cada una de las técnicas se analizó la métrica de segmentación llamada "DICE", que es el coeficiente de similitud de Sørensen-Dice para segmentación de imágenes. Se obtuvo la mejor media mediante el entrenamiento previo para observar cuál de las tres técnicas obtiene la mejor eficiencia, es decir se evaluó la mejor precisión para la segmentación de imágenes.

PALABRAS CLAVE: Segmentación, Image Labeler, MATLAB, Implementación, DICE, Otsu, etiqueta.

ABSTRACT

The present integration work implements the segmentation of reptile parasite images using MATLAB software through three segmentation techniques. Initially, the extraction and analysis of the mask for each of the images were performed, which were obtained from a large database of the research project "Identification of parasites using different coprological methods in reptile samples at the Quito Vivarium." The project includes 3616 images, and its objective was to distinguish gastroenteric parasites in reptiles.

The Image Labeler tool, which was used, was added to the MATLAB catalog in version 2022b. This tool facilitates the acquisition of labels that mark or highlight the region of interest (ROI). The corresponding masks for each image were divided into two groups: 70% for training, which is approximately 2532 images, and 30% for validation. These images are randomly selected each time the program and/or code is run, meaning they change.

The implemented techniques were Otsuthresh, Activecontour, and Imsegfmm, which work with 2D images and serve the purpose of image training. Using the training images and the application of each method, the segmentation metric called "DICE" was analyzed. DICE is the Sørensen-Dice similarity coefficient for image segmentation. The best average was obtained through pre-training to observe which of the three techniques achieves the best efficiency. In other words, the best accuracy for image segmentation was evaluated.

KEYWORDS: Segmentation, Image Labeler, MATLAB, Implementation, DICE, Otsu, label.

1 INTRODUCCIÓN

El Ecuador, a pesar de ser un país poco extenso, es uno de los más megadiversos del mundo, ya que posee 498 especies de reptiles a la fecha de realización del presente plan de titulación. El país acoge el mayor número de especies de reptiles por unidad de área, que representa aproximadamente el 4.3% de la diversidad mundial, esto incluye 35 especies de tortugas, 5 cocodrilos y caimanes, 3 anfisbénidos, 208 lagartijas y 250 culebras. Además, se podría decir que este número de especies de reptiles aumente constantemente durante los próximos años [1].

La gran variedad de reptiles en el país y su constante descubrimiento ha dificultado la elaboración de una guía completa de las especies que se va renovando frecuentemente. Sin embargo, esta se va poniendo al día rápidamente tanto en número como en parasitología [2].

En la actualidad Ecuador, ha mejorado su conciencia ambiental y respeto a la vida debido a la implementación de políticas y regulaciones. Así se han creado centros de manejo de fauna silvestre para ayudar con la supervivencia de reptiles, que han salido de su ambiente natural [3].

Actualmente existe una amplia gama de parásitos gastrointestinales presentes en los reptiles que se han analizado, por lo que se debe tener un plan de desparasitación apropiado, y así evitar que en un futuro sean propenso a futuras patologías, además de implementar controles de sanidad para proporcionar las mejores condiciones en su ámbito [3].

Los parásitos constituyen una extensa gama tanto de protozoarios[1] y helmintos[2] que se localizan en el tracto digestivo, como se conoce existe gran diversidad de formas parasitarias dependiendo de su afinidad y de la especie [4]. Es fundamental el poder detectar a tiempo el tipo de parásito ya que conlleva que el reptil pueda ser curado a tiempo permitiendo su aislamiento. Al establecer medidas preventivas y de control se logra continuar su ciclo de vida con normalidad, además de no propagar la enfermedad a otros de su especie o en peores casos directamente a humanos [5].

[1] Los Protozoarios son microorganismos de una sola célula que forman parte del reino Protista, que destacan por su diversidad de formas, comportamientos y funciones en diversos ecosistemas.

[2] Los Helmintos son seres parasitarios multicelulares, que pueden vivir en el cuerpo humano u otros animales, provocando distintas enfermedades.

Al no contar con una investigación científica eficiente respecto a la parasitología de los reptiles se realiza una segmentación de cada una de las imágenes de las especies para poder obtener una base de datos útil para la utilización de próximos usuarios.

1.1 OBJETIVO GENERAL

Segmentar imágenes de parásitos de reptiles mediante la herramienta Image Labeler para comparar con técnicas de segmentación ya existentes en MATLAB.

1.2 OBJETIVOS ESPECÍFICOS

- Estudiar la teoría de la segmentación de imágenes
- Investigar los aspectos de la parasitología en reptiles.
- Segmentar imágenes que contienen parásitos de reptiles.
- Comprobar el funcionamiento y verificar los resultados obtenidos.

1.3 ALCANCE

A partir del proyecto de investigación, "Identificación de parásitos con diferentes métodos coprológicos en muestras de reptiles en el Vivarium de Quito" facilitado por la Médica Veterinaria Alejandra Núñez, quien proporcionó las imágenes de los parásitos con los que se trabajó en este proyecto de titulación identificándolos de acuerdo con su morfología mediante microscopio [6].

La colección de datos cuenta con un total de 3616 imágenes abarcando 42 diferentes tipos de parásitos. Sin embargo, el objetivo de este trabajo de titulación es emplear las imágenes de solamente 6 parásitos específicos debido a su gran cantidad y variedad como se muestra en la Figura 1.1.

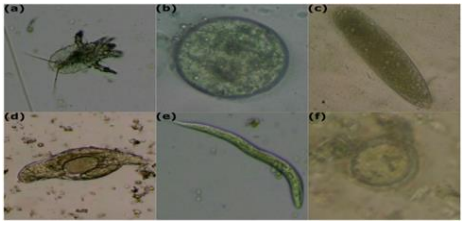

Figura 1.1 Ejemplos de cada parásito. a) Ophionyssus natricis; b) Strongyloides; c) Oxiurdo huevo; d) Taenia; e Blastocystis sp); f). Rhytidoides similis

En la Tabla 1.1 se puede observar la cantidad exacta de cada tipo de parásito con la que se trabajó.

Tabla 1. Número de imágenes por parásito [6].

Parásito	Número de imágenes
Acaro (Ophionyssus natricis)	245
Blastocystis sp	950
Oxiurdo huevo	345
Rhytidoides similis	1072
Strongyloides	648
Taenia	356
TOTAL	3616

Se llevo a cabo el procesamiento de imagen mediante segmentación, el cual implica en dividir la imagen en múltiples partes, caracterizando los pixeles de la misma, este proceso se realizó mediante la aplicación llamada Image Labeler que contiene la plataforma de programación MATLAB, donde se compararon tres técnicas de segmentación (Otsuthresh, Activecontour e Imsegfmm), utilizando métricas de segmentación apropiadas para evaluar las técnicas implementadas observando cual técnica fue la más óptima y eficiente.

1.4 MARCO TEÓRICO

1.4.1 PARÁSITOS EN REPTILES

Los parásitos son organismos que se desarrollan sobre o en el interior de otro organismo llamado huésped y se alimentan del huésped, es decir se aprovechan de sus nutrientes.

En el ámbito de la parasitología, no solo consiste en el estudio del parásito del animal en cautiverio, si no en numerosos factores que se relacionan con el parásito y su ciclo vital, es decir, tener en cuenta su comienzo, desarrollo y finalmente el exterminio de la parasitosis que ocasionan en los organismos pluricelulares [7].

Debido a la alta difusión de múltiples especies de parásitos en todo el mundo no se ha podido establecer un seguimiento preciso de los parásitos existentes en cuanto a cantidad, especie, morfología, etc. Pero se puede determinar lo extenso de la invasión del parásito por diferentes métodos de investigación de tipo cualitativos

El estudio de la parasitología en animales reptiles es un campo con poca exploración científica debido a varios factores como por ejemplo el fenómeno de helmintosis latente, que quiere decir que los helmintos aun sean muy jóvenes y no fueron detectados en el momento del análisis de laboratorio, también se pueden mencionar otros factores como el estado fisiológico del parásito o los fenómenos de resistencia adquiridos [7].

La necesidad de conocer el proceso invasivo de los parásitos es esencial ya que se puede conocer la capacidad invasiva que dicho organismo provocara en la naturaleza, además de su establecimiento, reproducción y como detalle prescindible evitar que estos parásitos lleguen al punto de propagarse hacia los humanos.

Al llevarse a cabo una investigación por Denom en 2007, en un zoológico en Estados Unidos, se observó que el 50% de los reptiles que fueron examinados, resultaron parasitados, por lo que se concluyó que esta fue la principal causa de muerte del 80% de los reptiles. Adicionalmente otras investigaciones han permitido establecer que los nematodos son los más prominentes en términos de representación, siendo los responsables de las lesiones ocasionadas y por los mismos figuran en el segundo lugar de muerte en animales reptiles que se mantienen o son criados en cautiverio [2].

En Ecuador, el análisis que se ha realizado para obtener detalles completos sobre los parásitos en reptiles es insuficiente y casi nulos. Es así como al conocer los estudios llamados “Determinación de la prevalencia y reinfestación de enteroparásitos en reptiles y aves silvestres del zoológico de Quito en Guayllabamba” y “Detección de parásitos del género Eimeria y Entamoeba invadens en la especie Iguana del parque histórico de Guayaquil y del parque seminario de Guayaquil” realizados por Castañeda en 2011 y Apolinario en 2011 respectivamente obtuvieron resultados negativos ya que en todas las especies analizadas más del 70% de cada una se encontraba parasitada, esto quiere decir, que no se ha tomado conciencia para tener entornos ecológicos óptimos para el correcto desarrollo y reproducción de los reptiles [2].

Al tener animales en estado de cautiverio refleja una decaída en sus defensas ya que el cambio en su entorno repercute en altos grados de improntación[3] que es un tipo de aprendizaje que ocurre en ciertos animales cuando se los extrae de su hábitat o se los aleja de los de su misma especie, esto ocasiona un estrés que provoca que sean más vulnerables a enfermedades, y por ende a contraer parásitos con facilidad [8].

1.4.1.1 Identificación de parásitos

En los reptiles hay varios tipos de parásitos y cada uno conlleva un tratamiento distinto, por lo tanto, primero se debe identificar la especie del parásito, y con el avance de la tecnología se puede realizar mediante aplicaciones de aprendizaje automático mediante técnicas de visión por computador [2].

Un diagnóstico idóneo depende de varios factores como la exposición a parásitos, edad del huésped, tratamiento previo, localización geográfica entre otros. La lectura correcta del resultado se verá influido por las formas de recolectar las muestras, así también como la elección de un correcto método de procesamiento [2].

Lo más habitual es usar el método directo que es la recolección de muestras de heces que puede ser recogida de múltiples maneras, en las cuales se realizan pruebas de concentración que pueden ser por flotación, que consiste en utilizar un medio líquido que tenga una suspensión más pesada que los parásitos en sí, los cuales suben a la superficie y facilita la observación microscópica, y por otro lado se tiene la sedimentación que se basa en la concentración de los componentes parasitarios por acción de la gravedad, o bien para una identificación del parásito más eficaz se realiza una combinación de ambas técnicas [9].

[3] La improntación se refiere al desarrollo de una marca perdurable o rastro en la mente, personalidad o conducta de un individuo o animal debido a una experiencia o influencia particular.

1.4.2 SEGMENTACIÓN DE IMÁGENES

La segmentación de imágenes constituye el paso fundamental para la implementación de sistemas automáticos ya que divide la imagen en varios segmentos u objetos, es decir, se clasifican por pixeles donde se asigna una etiqueta con el objetivo de obtener regiones de interés, al realizar este proceso se identificarán bordes, líneas o curvas.

Otra definición de segmentación la define como una clasificación de puntos, llamados pixeles, los cuales indican la categoría a la que pertenecen los diferentes puntos. Las características principales son: brillo, color, forma, textura, etc. [10].

La segmentación automática (asistida) es una tarea compleja del procesamiento de imágenes, pero es más rápida que la manual ya que esta conlleva tiempo, pero ambos tipos de segmentación deben cumplir con cuatro objetivos específicos para ser considerados efectivos y generales que son: definición de contornos, eliminación de ruido, independencia del umbral y métodos que no permitan la pérdida de datos [11].

Los algoritmos de segmentación se fundamentan en dos características que son: discontinuidad, similitud y conectividad. La discontinuidad se basa en cambios bruscos del nivel de gris donde se tiene detección de puntos aislados, líneas y bordes [10]. La similitud en cambio busca zonas con valores regulares del nivel de gris donde se valora el crecimiento de región, umbralización, división y fusión de regiones; estos dos tipos de segmentación son válidos tanto en imágenes estáticas con en las dinámicas (varia en el tiempo).

En la Figura 1.2 se muestra un ejemplo de segmentación.

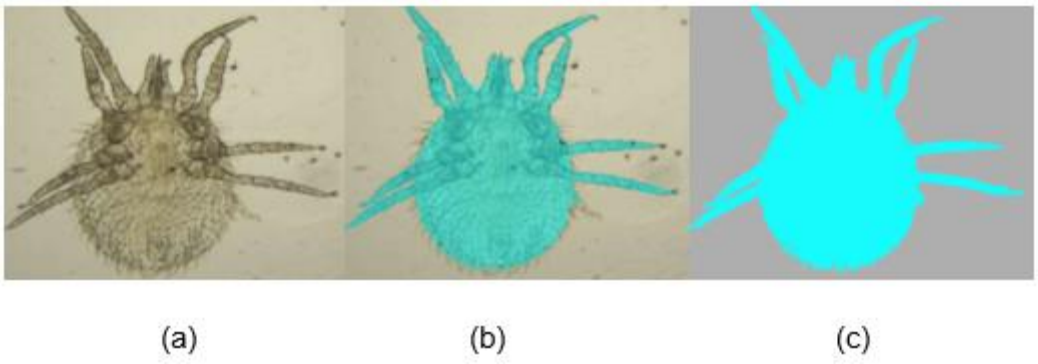

Figura 1.2 Ejemplos de segmentación de imagen: (a) Imagen original; (b) Imagen realizada en Imagen Labeler y (c) Máscara final obtenida.

1.4.2.1 Detección de puntos

Un punto aislado dentro de una figura tiene un valor de gris que difiere del valor o tono de los pixeles circundantes, es decir, una máscara (Laplaciano) la matriz para identificar un punto aislado es la siguiente:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Por lo tanto, se dice que un punto es aislado cuando el resultado de aplicar la máscara es mayor o igual que un cierto valor de umbral T [12].

1.4.2.2 Detección de líneas

Para detectar líneas se puede utilizar una máscara de Laplaciano análogamente, pero se desea detectar la línea en cuatro direcciones determinadas posibles: horizontal, vertical, de 45° y de -45° [12].

$$\begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix} \begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix} \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

1.4.3 MÉTODOS Y TÉCNICAS DE SEGMENTACIÓN

Existen diferentes métodos de segmentación como:

1.4.3.1 Segmentación binaria

Como su nombre lo indica solo tiene dos categorías frente y fondo, y su ejemplo más simple es la umbralización, que tiene el objetivo de alcanzar un valor de umbral que permita obtener una imagen en escala de grises, separando correctamente el fondo y el objeto que se desea separar, a todos los valores cuyo valor este por encima de ese valor se le asigna el nivel máximo de blanco y al resto se les asignara el color negro.

1.4.3.1.1 Técnicas de umbralización

Se tienen diferentes técnicas que difieren de acuerdo con la información que se desea rescatar de la imagen y el modo en la que se procesa la misma, entre las más importantes se tienen:

El método OTSU se destaca como uno de los más relevantes ya que altamente reconocido para el cálculo del umbral óptimo. Específicamente, emplea la varianza, que mide la dispersión de valores, para determinar el umbral de tal manera que la dispersión dentro de cada segmento sea mínima, pero al mismo tiempo la dispersión sea lo más alta posible entre segmentos diferentes. Para lograr esto, se calcula el cociente entre ambas varianzas y se busca un valor umbral que maximice esta proporción [13].

El método basado en histogramas brinda una información estadística que ofrece un umbral óptimo para la separación de las imágenes que representado por una curva que representa cada nivel de gris de la imagen, como desventaja es la presencia de ruido que toda imagen contiene.

El método por agrupamiento (Clustering) utiliza algoritmos de aprendizaje automático que consiste en dividir una imagen en N clústers que identifican similitudes de los pixeles en las imágenes y posteriormente etiquetarlas.[14].

El Método basado en los atributos de la imagen consiste en seleccionar valores de umbral, basados en las características buscando una media entre sus valores de similitud de la imagen original y la binarizada [15].

1.4.3.2 Segmentación asistida o semiautomática

La segmentación asistida de imágenes aun conlleva un gran reto dentro de las áreas que la investigan, debido a su alta complejidad y varias variables que se deben tomar en cuenta. Este reto ha conducido a realizar varios algoritmos tomando en cuenta múltiples aproximaciones semiautomáticas, pero aun en muchos escenarios se sigue utilizando la segmentación manual debido a que hay casos que se necesita la segmentación de una imagen con un fondo complejo. Los algoritmos más utilizados en este tipo de segmentación son Watershed (1979), GrabCut (2004) y métodos interactivos [16].

La segmentación basada en Watershed (Transformación divisoria) es una técnica que se basa en la complexión matemática, la cual extrae las fronteras de las regiones de interés, este clasifica los pixeles según su textura, gradiente de los niveles de gris y la proximidad espacial, es decir trabaja con imágenes en 3D [17].

El algoritmo BrabCut se basa en cortes de gráficos, es decir, el usuario delimita un cuadro en la imagen que desea segmentar y este algoritmo estima la distribución de color utilizando un modelo gaussiano. Por ende, el usuario puede corregir aún más las estimaciones señalando las regiones mal clasificadas y volviendo a ejecutar la optimización [18].

1.4.3.2.1 Propiedades para una segmentación asistida óptima

Las herramientas básicas para obtener un proceso correcto consisten en la visión asistida por el usuario, es decir, la interacción del individuo con la imagen, en la cual aplica varias técnicas esperando un resultado satisfactorio y la correlación basada en visión que se refiere a los métodos que el computador posee, con los cuales ayuda al usuario a que el resultado sea mejor. Por ejemplo, el sistema ICE (Edición Interactiva de Contornos) que representa los bordes de una figura y el usuario puede remover ciertas características a su conveniencia [16].

1.4.3.3 Segmentación basada en aprendizaje profundo

En este tipo de proceso de segmentación se incorporan las redes convoluciones (CNN-convolutional neural networks) las cuales han experimentado un rápido avance debido a la amplio predominio del aprendizaje profundo (Deep Learning) y ahora la gran mayoría de algoritmos, creados actualmente se basan en CNN, donde resuelven problemas como la segmentación semántica, instanciada y panóptica [19].

El concepto central al utilizar CNN no se trata únicamente de disminuir el volumen de la arquitectura, si no de desarrollar un modelo de aprendizaje adecuado para aprender a filtrar diversas características destacadas que posee la base de datos de imágenes con la que se trabaja, es decir, obtener un modelo automático que sea eficaz y eficiente, por lo tanto,

este nuevo modelo creado actualizará frecuentemente los coeficientes del Kernel para filtrar las imágenes que el modelo considere relevante [20].

Una de técnicas más utilizadas es la segmentación semántica que consiste en un algoritmo de Deep Learning que asocia una etiqueta a cada píxel de la imagen, es decir, puede reconocer diferentes categorías en una sola imagen como se muestra en la Figura 1.3.

Este tipo de segmentación se utiliza en diversos campos donde se requiere mapas con alta precisión como conducción autónoma, imágenes médicas, por satélite, entre otras [21].

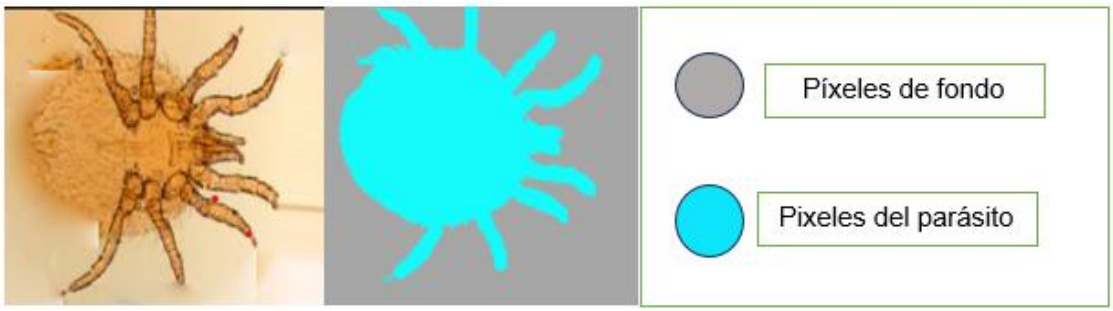

Figura 1.3 Ejemplos de segmentación semántica.

1.4.4 MÉTRICAS DE EVALUACIÓN DE ALGORITMOS PARA SEGMENTACIÓN

Las métricas de segmentación analizan si el modelo creado está aprendiendo correctamente o no. Habitualmente, cuando la exactitud da valores extremadamente inferiores a 1 o cercanos a 0, quiere decir que las métricas se encuentran en niveles bajos y es necesario continuar el proceso de entrenamiento para mejorar el resultado [22].

Existen varias métricas como:

1.4.4.1 Coeficiente de similitud de Sørensen-Dice

El coeficiente de Dice, es una medida estadística empleada para evaluar y comparar la similitud de dos conjuntos de muestras. La fórmula original de Sørensen está diseñada para ser utilizada en datos de presencia/ausencia, y se presenta en la ecuación (1.1).

$$\boldsymbol{S = \frac{2Z}{X+Y} = \frac{2\,.X \cap Y}{X+Y}} \tag{1.1}$$

Donde (*X*) y (*Y)* representa el número de especies en las muestras, mientas que (Z) es el número de especies compartidas por ambas muestras, finalmente (S) denota el cociente de similitud el cual oscila en un rango de 0 a 1 [23].

1.4.4.2 Coeficiente de precisión

El cálculo del porcentaje de predicción positiva realizada por los clasificadores que son correctos y se presenta en la ecuación (1.2).

$$\boldsymbol{P = \frac{TP}{TP+FP}} \tag{1.2}$$

El valor (TP) representa el número de clases positivas de la proporción de predicciones positivas correctamente y (FP) representa el número de clases negativas de la proporción de predicciones positivas correctamente [24].

1.4.4.3 Recall

El cálculo el porcentaje de patrones positivos que el clasificador detecta de forma correcta y se presenta en la ecuación (1.3).

$$\boldsymbol{Recall = \frac{TP}{TP+FN}} \tag{1.3}$$

Donde (TP) es el número de clases positivas que son predicciones incorrectas como negativas y (FN) es el número de clases negativas que son predicciones correctas como negativas [24].

1.4.4.4 Técnicas de segmentación en MATLAB

Se puede observar las técnicas y aplicaciones con las que cuenta MATLAB para la segmentación de imágenes como se observa en la Figura 1.4.

graythresh	Establecen umbrales de varios niveles en una imagen usando el método de Otsu
multithresh	
otsuthresh	
adaptthresh	Umbral adaptativo de la imagen utilizando estadísticas locales de primer orden
grayconnected	Seleccione una región de imagen con valores de gris similares utilizando la técnica de relleno por inundación
watershed	Transformada de Wateshed
activecontour	Utiliza técnicas de crecimiento regional de contornos activos
lazysnapping	Segmentación basada en gráficos
grabcut	Segmentación iterativa basada en gráficos
imseggeodesic	Segmentación de color basada en la distancia geodésica
imsegfmm	Segmentación binaria de imágenes utilizando el método de marcha rápida
gradientweight	Calcula los pesos de los píxeles de la imagen en función del gradiente
graydiffweight	Calcula los pesos de los píxeles de la imagen en función de la diferencia de intensidad de la escala de grises
imsegkmeans	Segmentación de imágenes basada en la agrupación K-medias
imsegkmeans3	Segmentación de volumen basada en agrupamiento de K-means
superpixels	Sobresegmentación de superpíxeles 2D de imágenes
superpixels3	Sobresegmentación de superpíxeles 3D de una imagen 3D

Figura 1.4 Técnicas de Segmentación en MATLAB [25].

1.4.5 HERRAMIENTA PARA LA SEGMENTACIÓN DE IMÁGENES

Las herramientas que realizan la segmentación asistida deben poseer varias propiedades algorítmicas que incluyen: simplicidad, es decir, que la interfaz debe ser sencilla e instintivo,

al mismo tiempo debe ser predecible, robusto en el cual se acepten imágenes ruidosas y aun así produzcan un buen resultado, inteligente donde el usuario no requiera realizar tantos pasos para realizar su tarea, rápido y manipulable [16].

Es importante señalar que los métodos primarios de segmentación de segmentación de imágenes se encuentren ya incorporados con bibliotecas como OpenCV o Scikit-Image, las cuales están accesibles para Python y se pueden instalar de manera sencilla mediante sistemas de administración de paquetes, como es el caso de pip.

También existen varios paquetes de software de código abierto como ITK, GIMP, VXL, ImageMagick, Fiji y también existen paquetes para fines académicos GemIdent, CVIPtools y MegaWave [19].

El articulo “Segmentación Asistida de Imágenes Usando Características de Textura” presenta una propuesta de un tipo de control de una brocha inteligente para poder segmentar imágenes basados en texturas inspirada en Smart Scissors (tijeras inteligentes) que trata de eliminar el exceso de datos que el usuario no necesita siguiendo el diagrama de flujo descrito en la Figura 1.5 [26].

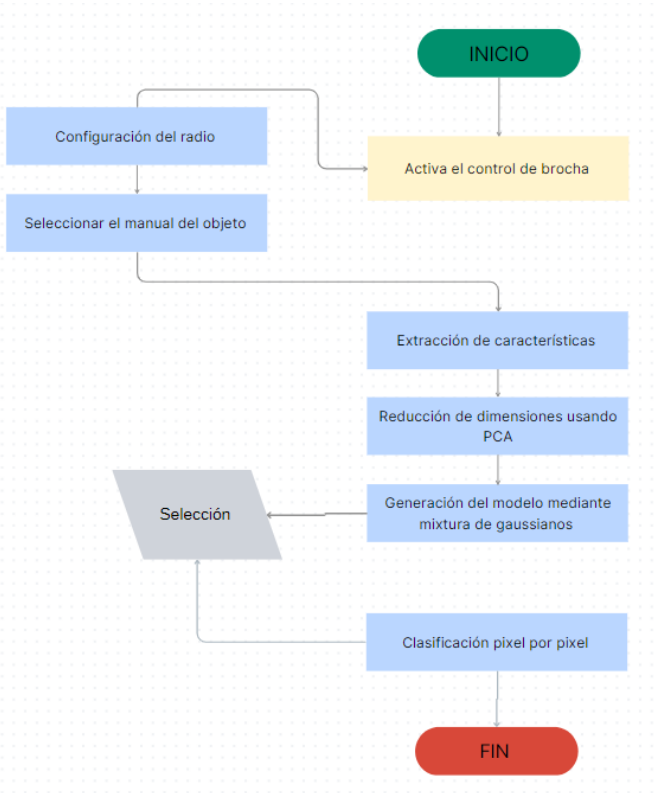

Figura 1.5 Flujo de usuario propuesto del control brocha inteligente [26].

Como se observa, existen varios software's en los cuales se puede segmentar imágenes, pero al realizar este trabajo de titulación en el software de MATLAB se desarrolla a profundidad las herramientas y Apps que se encuentren en esta plataforma.

1.4.5.1 Herramientas de MATLAB

En la actualidad el tratamiento de imágenes ya no se trata de un tema básico, ahora es fundamental para el entrenamiento de algoritmos que crean la facilidad de trabajar con grandes bases de datos, donde las imágenes pueden ser clasificadas y tratadas.

MATLAB proporciona un entorno de desarrollo integrado (IDE) con su propio lenguaje de programación, en esta plataforma se pueden realizar numerosas actividades, entre las más básicas se tienen: implementación de funciones y algoritmos, manipulación de matrices, interfaces de usuarios (GUI), se pueden ampliar fácilmente su capacidad mediante los TOOLBOXES y para el caso del procesamiento de imágenes se utiliza el Toolbox "Image Processing" [27].

La clasificación y procesamiento de imagen requiere una gran cantidad de recursos y programación, ya que se necesita una base de datos compacta, es decir maneja imágenes o documentos realmente pesados. Por ejemplo, se ha utilizado para publicidad, para conocer los gustos de los usuarios, también se utiliza en temas más relevantes como conocer si una persona tiene una enfermedad grave y que tan avanzada se encuentra la misma mediante inteligencia artificial, Deep Learning entre otras.

Las aplicaciones que utiliza MATLAB para el tratamiento de imágenes permiten mecanizar y automatizar los flujos de trabajos que se usan normalmente en el procesamiento de imágenes, lo cual permite segmentar imágenes, comparación de técnicas de segmentación, además trabaja con extensos paquetes de datos con los que se puede trabajar de forma interactiva, esto permite explorar imágenes, videos, creación de histogramas, explorar y manipular regiones de interés (ROI) [27].

1.4.5.2 Computer Vision Toolbox

Computer Vision Toolbox tiene la capacidad de realizar diversas funciones, algoritmos y/o aplicaciones, el cual diseña sistemas de procesamiento de imágenes 2D y 3D, además de video y visión por monitor, esta caja de herramientas es de código abierto e incluye muchas funciones útiles para extraer y manipular características de imágenes.

La caja de herramientas puede detectar, extraer y realizar una coincidencia de las características propias de las imágenes que se procesan, es decir automatiza el flujo de trabajo para el usuario, además se produce el etiquetado de las imágenes en tiempo real [28].

Además, uno de los aportes más importantes es que se puede realizar el entrenamiento mediante bases de datos utilizando algoritmos de Deep Learning como U-Net y E-CNN para segmentación Semántica y Aprendizaje automático como SSD, SCF y YOLO.

Computer Vision Toolbox ofrece algoritmos para segmentación analizando imágenes que son demasiado grandes; Existen modelos preentrenados que pueden detectar personas, animales, frutas y otros objetos comunes [29].

1.4.5.3 Image Labeler

La herramienta más relevante que se utilizó en este trabajo de integración fue Image Labeler, la cual permitió obtener las máscaras de las 3616 imágenes en las que se aplicaron las técnicas de segmentación.

Image Labeler es una aplicación que crea etiquetas reales de una manera didáctica e interactiva con lo cual crea regiones de interés (ROI), la APP posea una diversidad de formas (), las cuales se pueden aplicar en una imagen o un conjunto de imágenes 2D o 3D, también se puede aplicar a videos [30].

Se detalla explícitamente esta aplicación en el apartado 2.2.

2 METODOLOGÍA

En el siguiente capítulo se observa la implementación de las técnicas de segmentación analizando la métrica de comparación que existe en MATLAB. En la sección 2.1 se detalla cómo se obtuvo la base de datos con la que se trabaja de las 3616 imágenes. En la sección 2.2 se obtienen las máscaras de las imágenes mediante la aplicación Image Labeler que posee MATLAB con las cuales se crea una segunda base de datos con las que se realiza la comparación. A continuación, en la sección 2.3 se plantean las tres técnicas de segmentación que se utilizan con su debido entrenamiento. En la sección 2.4 se detalla la métrica de evaluación utilizada llamada DICE que es el coeficiente de similitud para determinar la eficiencia y precisión de cada una de las técnicas utilizadas.

2.1 BASE DE DATOS

Para el presente trabajo de integración curricular se utilizó la base de datos recolectada mediante el proyecto de investigación, “Identificación de parásitos con diferentes métodos coprológicos en muestras de reptiles en el Vivarium de Quito” facilitado por la Médica Veterinaria Alejandra Núñez, esta es de dominio público [6]. En la Figura 2.1 se pueden visualizar los parásitos de los reptiles con los que se trabajó

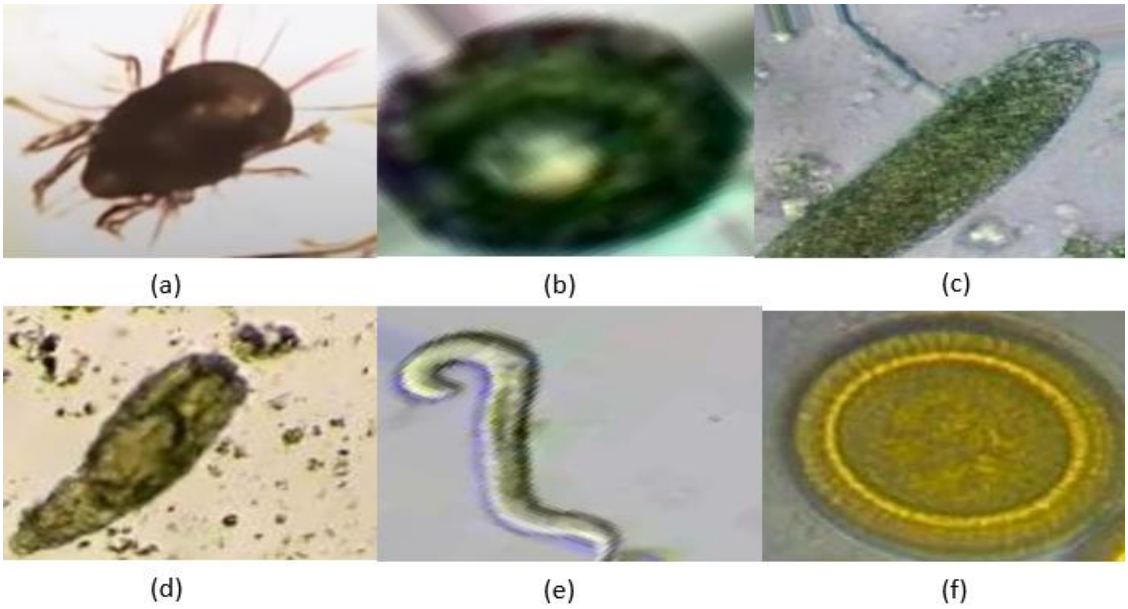

Figura 2.1 Ejemplos de cada parásito. a) Ácaro; b) Blastocystis sp; c) Oxiurdo huevo; d) Rhytidoides similis; e) Strongyloides; f) Taenia.

El conjunto de datos que se obtuvieron cuenta con 3616 imágenes de diferentes tipos de parásitos de reptiles. Para cumplir con la finalidad de este trabajo de titulación, se utilizan las imágenes de 6 parásitos como se detalla en la Tabla 2.1.

Tabla 2.1 Número de imágenes por parásito

Parásito	Acaro	Blastocystis sp	Oxiurdo huevo	Rhytidoides similis	Strongyloides	Taenia
Número de imágenes	245	950	345	1072	648	356
TOTAL	**3616**					

2.2 MÁSCARAS DE IMÁGENES MEDIANTE IMAGE LABELER

Para la obtención de las máscaras de las imágenes de los parásitos se utilizó la aplicación Image Labeler que posee MATLAB, esta aplicación ayudó a marcar las regiones de interés con las que se midió el coeficiente de similitud, comparando las máscaras obtenidas mediante la aplicación y las técnicas de segmentación que ya se encuentran predeterminadas en MATLAB, las cuales se detallan a profundidad en la sección 2.3. A continuación, se detalla cómo se obtuvieron las máscaras de cada imagen mediante un ejemplo de aplicación.

En la parte superior de MATLAB, en la pestaña "APPS" se puede encontrar "Image Labeler", se debe tomar en cuenta que esta aplicación solo se encuentra en las versiones recientes de MATLAB como se muestre en la Figura 2.2.

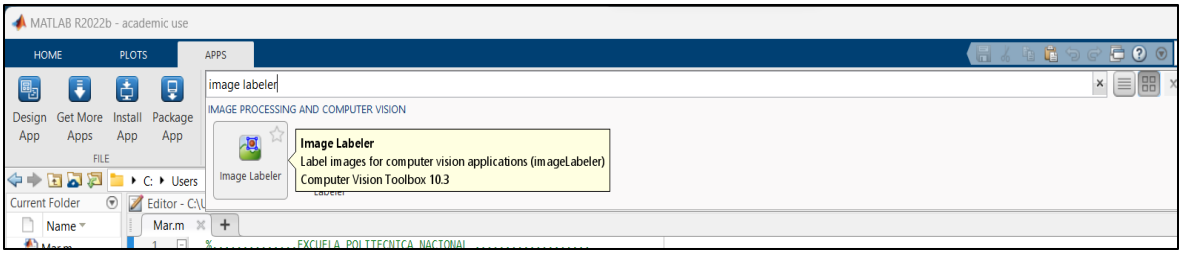

Figura 2.2 Ingreso a Image Labeler en MATLAB.

Al abrir Image Labeler se deben importan las imágenes que se desea obtener las máscaras en la aplicación como se observa en la Figura 2.3, es decir se agregan las imágenes deseadas puede ser una o un gran conjunto, se recomienda cargar entre 100 y 150 imágenes dependiendo de la capacidad del computador.

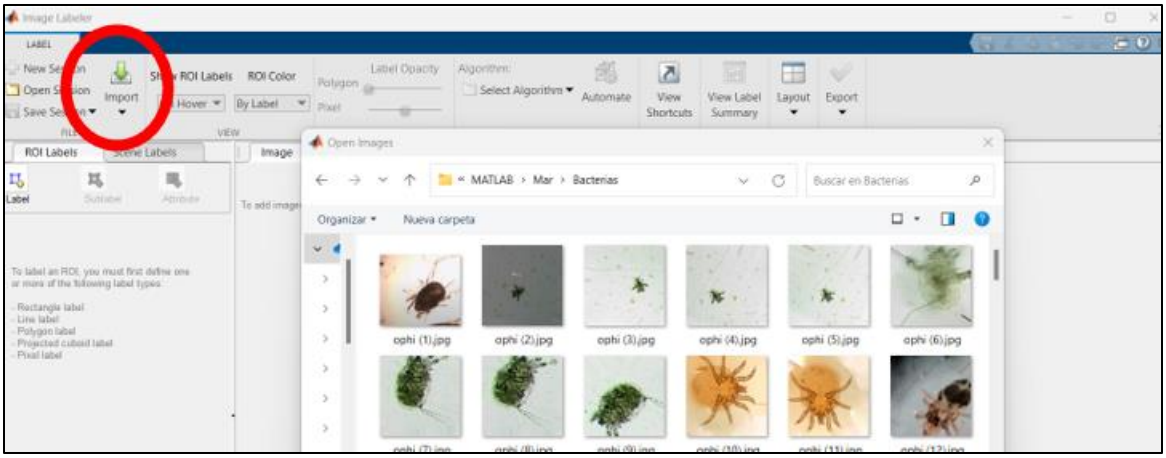

Figura 2.3 Proceso para importar imágenes.

Dependiendo del número de imágenes que se cargan la aplicación se demora antes de mostrar todas las imágenes en la parte inferior y la imagen con la que se trabaja aparece expandida en la ventana.

En la Figura 2.4 se observa que las imágenes han sido cargadas correctamente.

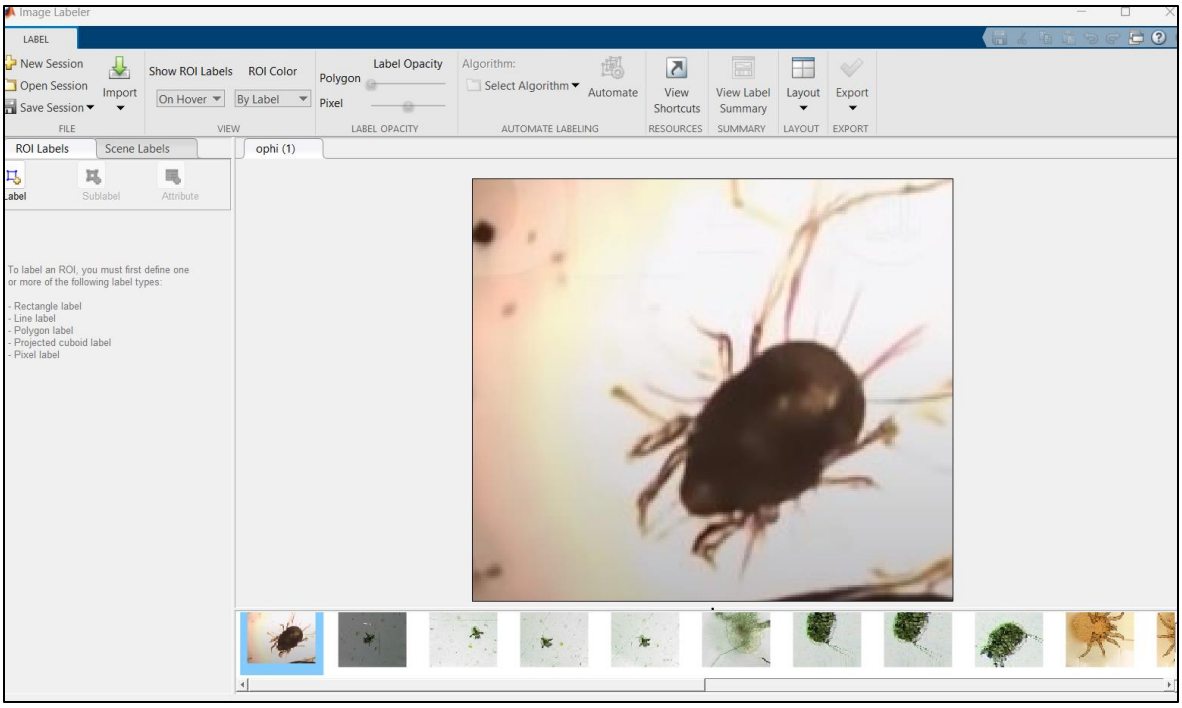

Figura 2.4 Visualización de las imágenes cargadas.

A continuación, se crean las etiquetas “Labels” que se encuentran en la parte izquierda, donde se le asigna un nombre (en este caso bact_mascara) y se tienen algunas opciones como rectángulo, línea, polígono, píxel y paralepípedo. Para este trabajo se utilizó la opción de Píxel, ya que las imágenes de los parásitos no tienen formas definidas y con esta opción el usuario puede seguir cualquier forma que desee y obtener una máscara del parásito más precisa.

En la Figura 2.5 se observa que se puede añadir las etiquetas necesarias en cualquier momento, variando el nombre y el color de preferencia.

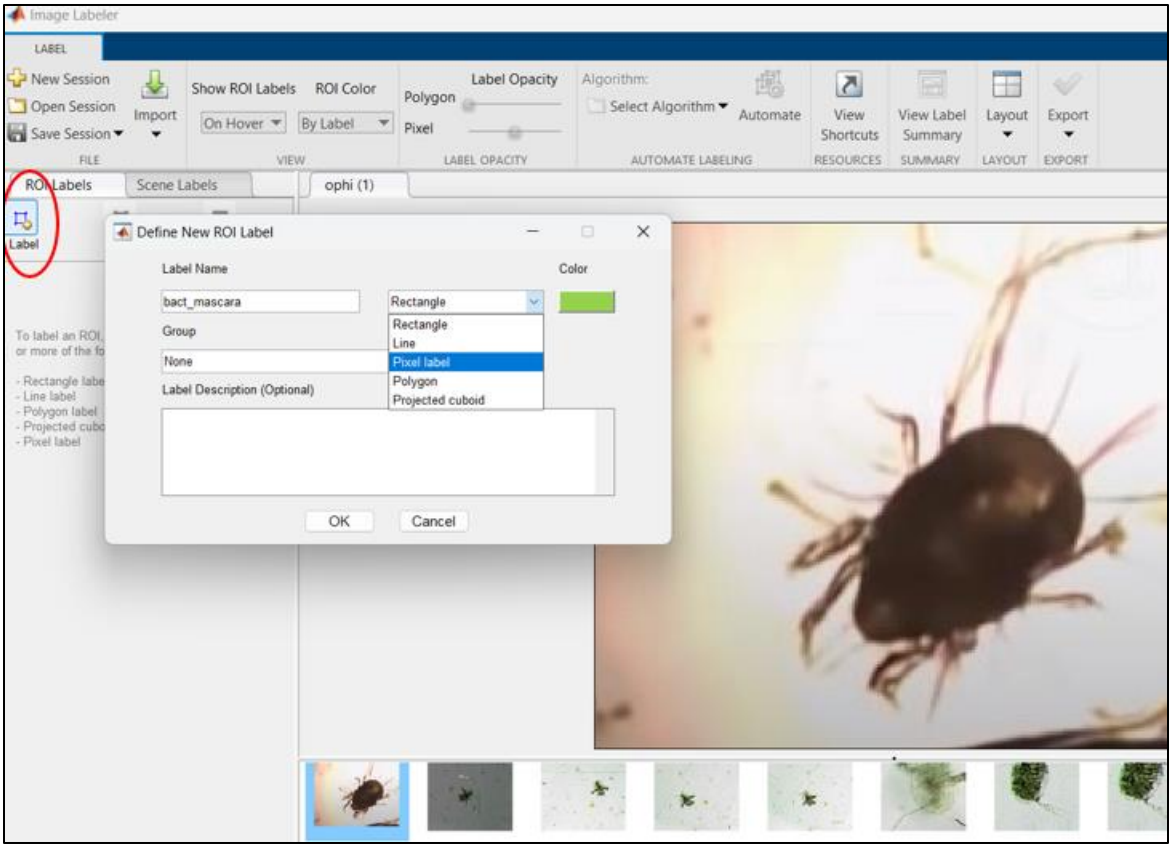

Figura 2.5 Etiquetado de las imágenes.

En la parte superior hay varias opciones para un correcto etiquetado, pero las más utilizadas fueron pincel (Brush), borrador (Erase), tamaño del pincel (Brush Size), relleno de inundación (Flood Fill) y Superpixel. Dependiendo de la imagen se marcó la zona de interés (ROI) usando la opción de etiquetado más adecuada.

Se utilizaron estas herramientas de etiquetado de forma interactiva como se puede observar en la Figura 2.6, donde se pinta la etiqueta encima de la imagen que se está trabajando.

Las herramientas que posee Image Labeler son muy similares a Microsoft Paint que es una plataforma conocida mundialmente. Este proceso se realizó con las 3616 imágenes de la base de datos con la que se trabajó.

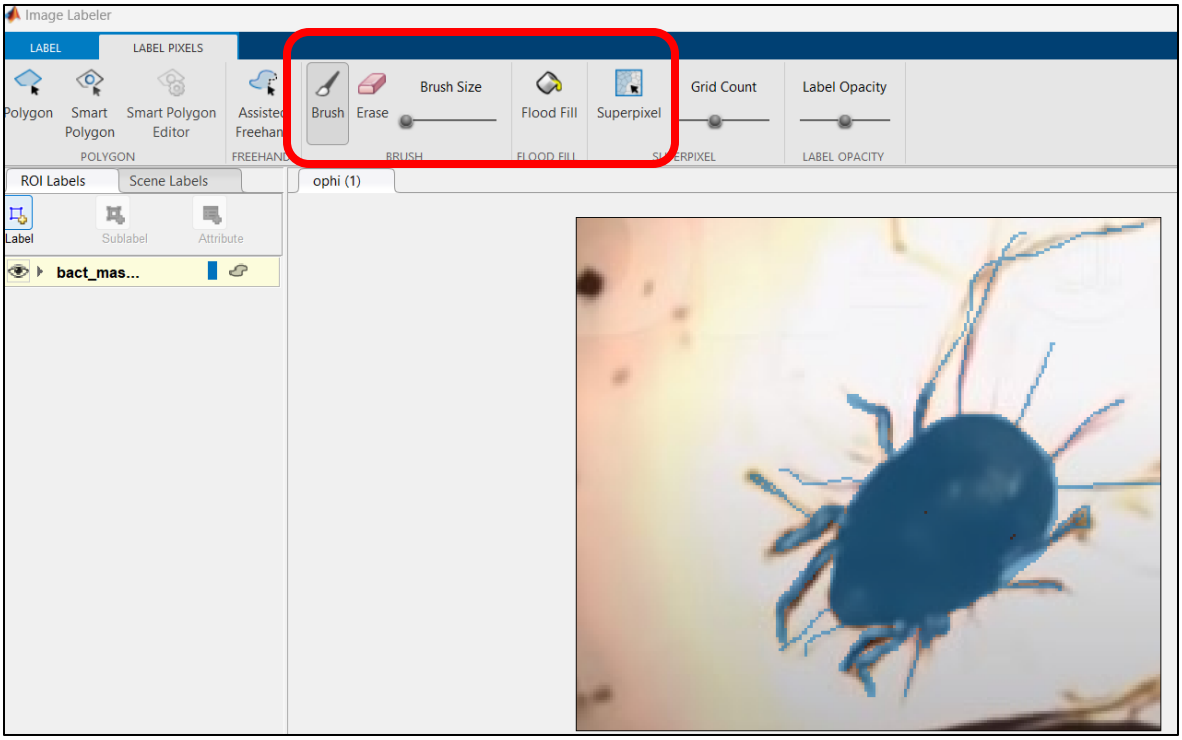

Figura 2.6 Herramientas de etiquetado para las imágenes.

Se recomienda guardar el progreso, ya que cuando se cargan demasiadas imágenes el programa suele fallar y se debe volver a empezar el trabajo, por lo tanto, si se guarda cada cierto tiempo Image Labeler guarda las etiquetas ya creadas como se observa en la Figura 2.7.

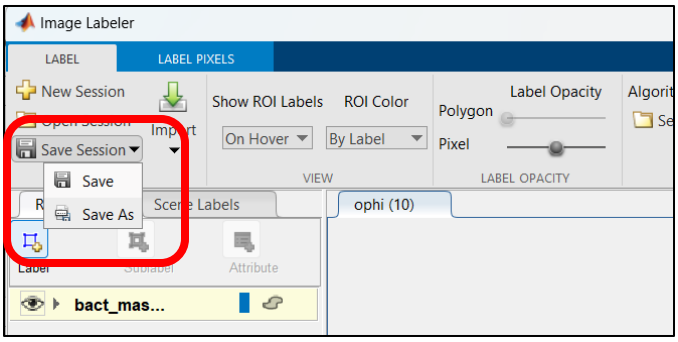

Figura 2.7 Guardar progreso de etiquetado.

Al etiquetar todas las imágenes, estas etiquetas deben exportarse NO solamente guardarse, así estas imágenes exportadas se utilizan para usarse en el entrenamiento de cualquier modelo como se muestra en la Figura 2.8.

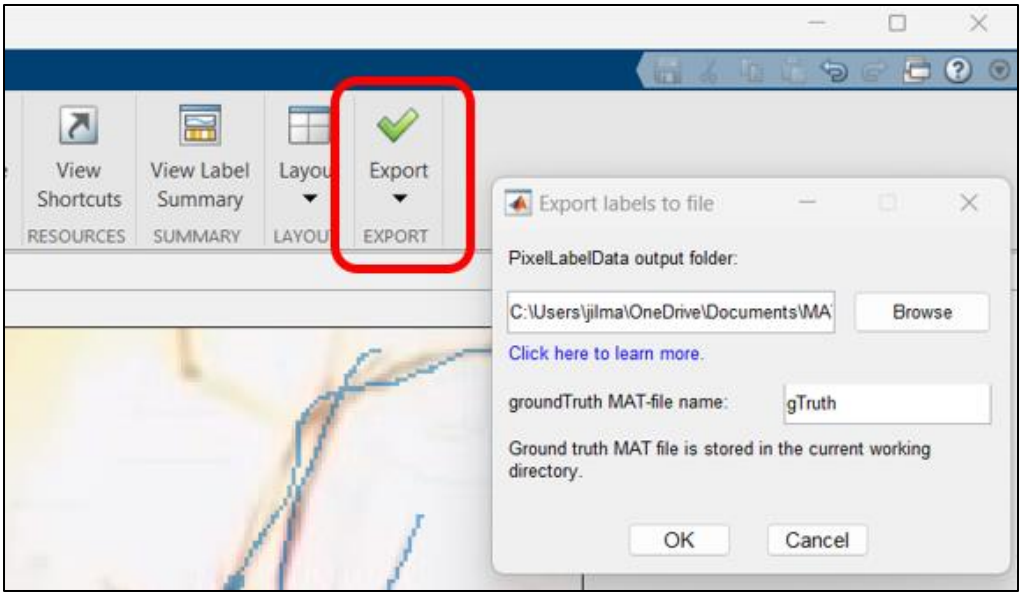

Figura 2.8 Exportar etiquetas

Finalmente, para obtener el resultado de las etiquetas, es decir la región de interés (ROI), se utiliza el código en MATLAB que se muestra en el ANEXO I.

La máscara obtenida para este ejemplo se puede observar en la Figura 2.9.

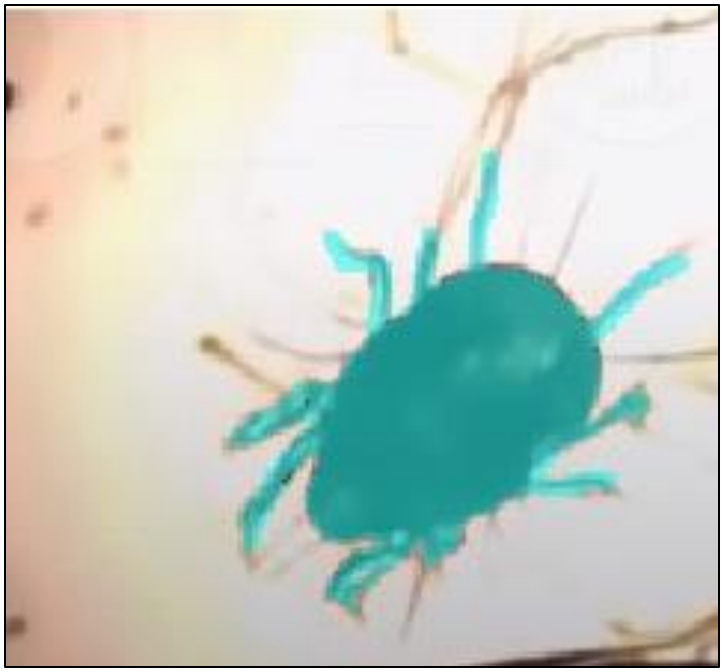

Figura 2.9 Máscara final obtenida.

2.3 APLICACIÓN DE LAS TÉCNICAS DE SEGMENTACIÓN

MATLAB cuenta como varias técnicas de segmentación como se pudo observar en la Figura 1.4, Pero para la implementación de este Trabajo de Integración Curricular se implementó tres técnicas que son Otsuthresh, Activecontour e Imsegfmm, ya que mediante pruebas realizadas con anterioridad son las únicas técnicas que permite variar los parámetros de evaluación para luego validar los resultados obtenidos del mismo.

2.3.1 Otsuthresh

La técnica de segmentación Otsuthresh tiene la siguiente sintaxis:

```
T = otsuthresh(counts)
```

Donde:

T utiliza el método para la segmentación utilizando la técnica Otsuthresh, el cual realiza umbrales de imágenes automáticamente, es decir, busca un valor ideal para la umbralización. Este usa la función “imbinarize” descrita por MATLAB que crea una imagen binaria que servirá para la comparación con las imágenes obtenidas en la sección 2.2,el parámetro “counts” muestra los recuentos de histogramas con números solamente positivos en vectores.

En la Figura 2.10 se muestra un ejemplo de aplicación de la técnica Otsuthresh, la foto fue tomada de la base de datos con la que se está trabajando y es un tipo de parásito Blastocystis sp.

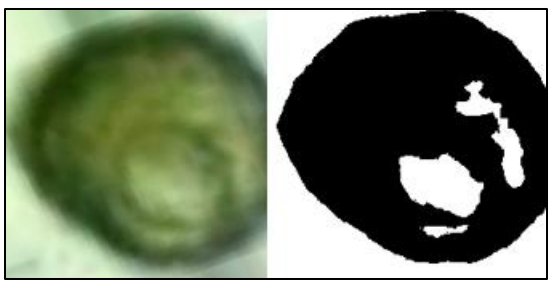

Figura 2.10 Ejemplo utilizando la técnica Otsuthresh

Esta técnica cuenta por defecto la función imhist(Imagen.L) descrita por MATLAB, que se utiliza para calcular el histograma, donde "L" es el valor a entrenar, el cual solicita la imagen original y un valor "L" que permite realizar la segmentación, por defecto este valor empieza en 3 pero se va variando hasta encontrar el mejor; hay imágenes que al aumentar este valor arroja una mejor comparación es por eso que en los resultados se seleccionó el mejor valor de "L" que es el valor óptimo para el mejor entrenamiento para esta técnica [13].

2.3.2 Activecontour

La segunda técnica de segmentación Activecontour tiene la siguiente sintaxis:

```
BW = activecontour(A,mask,n)
```

Como lo dice su nombre utiliza la técnica iterativa de segmentación mediante el crecimiento de contornos activos (serpientes), es decir especifica la curva original de la imagen tratada y con la función desarrolla esta curva hacia el límite de esta y así tener un máximo de iteraciones.

Es decir esta técnica segmenta la imagen "A" evolucionando el contorno durante un máximo de "n" iteraciones las cuales son controladas por la variable "mask", esta variable por defecto empieza en 100, si se aumenta este valor se obtiene mejores resultados, sin embargo el crecimiento debe ser considerable en este caso se asume un crecimiento de 100, para que mejores los resultados, mientras más crece se mejora, sin embargo llega el punto en que por más que crezca el resultado ya no mejora considerablemente y se mantiene constante [31].

En la Figura 2.11 se muestra un ejemplo de aplicación de la técnica Activecontour, la foto fue tomada de la base de datos con la que se está trabajando.

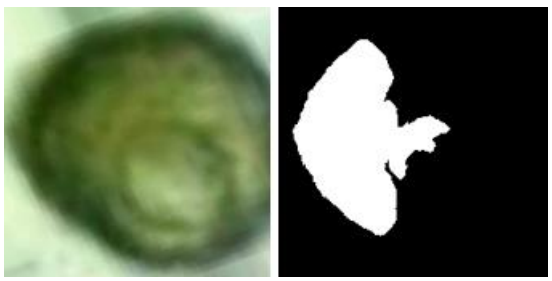

Figura 2.11 Ejemplo utilizando la técnica Activecontour.

2.3.3 Imsegfmm

La tercera técnica de segmentación Imsegfmm tiene la siguiente sintaxis:

```
BW = imsegfmm(W,mask,thresh)
```

Imsegfmm es una técnica de segmentación de imágenes binarias mediante el método de marcha rápida, el cual devuelve una imagen segmentada, donde la matriz "W" especifica los pesos para cada píxel, "mask" especifica la ubicación siendo una matriz lógica y "thresh" el nivel de umbralización.

Imsegfmm utiliza por defecto la función "graydiffweight" la cual calcula el peso de los píxeles de la imagen en función de la diferencia de intensidad en escala de grises, la función Imsegfmm mejora dependiendo del resultado de graydiffweight el cual varía dependiendo de cuanto cambie el valor de "L", es por eso por lo que esta es la variable que va a ser entrenada.

W = graydiffweight (i, mask, 'GrayDifferenceCutoff', L).

La variable "L" sirve para entrenar, es decir el valor optimizado, crecerá de 10 en 10 ya que si crece de forma más pequeña los cambios en el resultado final no tienen mayor diferencia [32].

En la Figura 2.12 se muestra un ejemplo de aplicación de la técnica Imsegfmm, la foto fue tomada de la base de datos con la que se está trabajando.

Figura 2.12 Ejemplo utilizando la técnica Imsegfmm.

2.4 VALIDACIÓN DE LA MÉTRICA DE SIMILITUD

Al realizar el entrenamiento de cada una de las técnicas vistas en la sección 2.3 se obtuvo el valor de entrenamiento óptimo el cual se utiliza para poder comparar el grado de similitud entre las imágenes obtenidas en la sección 2.2 con los de la sección 2.3.

La métrica clave que se analizó para realizar la comparación fue el coeficiente de similitud (DICE) que se detalla en la sección 1.4.4.1.

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 RESULTADOS

3.1.1 Creación de máscaras

Para crear las máscaras de imágenes se utiliza la aplicación Image Labeler de MATLAB utilizando el código mostrado a continuación.

```
for n = 1: Total_de_imagenes
    nombre = sprintf('imagen_original_%d.jpg', n);
    i = imread(['C:\Users\ rutaBDDLocal\imagen_original\',nombre]);
    nombre2 = sprintf('etiqueta_%d.png', n);
    x = imread(['C:\Users\rutaBDDLocal\etiquetas\PixelLabelData\',nombre2]);
    classNames = "bact";
    cmap = jet(numel(classNames));
    B = labeloverlay(i,x,'Transparency',0.5,'Colormap',cmap);
    figure('Name','Imagen_Labeler','NumberTitle','off'),imshow(B);
    nombre3 = sprintf('mascara_%d.jpg', n);
    saveas(gcf,['C:\Users\rutaBDDLocal\carpeta_destino\', nombre3]);
    close all;
end
```

Cada imagen tendrá un nombre especifico “imagen_original_%d.jpg”, donde “d” varía desde el 1 hasta el total de imágenes con las que se trabaja, inmediatamente se leen las imágenes originales desde la base de datos local, donde se debe tomar en cuenta que esta dirección cambiará dependiendo del computador de cada usuario. A continuación, se crea una función de MATLAB que define el color de la segmentación a realizarse.

Se utilizan las imágenes creadas en Imagen Labeler “etiquetas” y la variable “B” crea las figuras, al cumplirse la segmentación correcta se crean las imágenes finales, es decir las máscaras 'mascara_%d.jpg' donde de igual manera “d” varía desde el 1 al hasta el total de imágenes con las que se trabaja. S requiere la impresión de las imágenes en pantalla para poder capturarlas y guardarlas en la carpeta de destino, por lo cual esta carpeta será la nueva base de datos con la que se realiza posteriormente la comparación con las técnicas de segmentación

3.1.2 Función para Segmentación de Imágenes con Labeloverlay

La función que se presenta a continuación toma como parámetro de entrada "n", que representa el número de la imagen a segmentar. La función devuelve las variables "i" y "BL". En específico, la variable "i" corresponde a la imagen obtenida de nuestra base de datos, y es esta imagen la que se someterá al proceso de segmentación. Esta imagen se empleará tanto en el método "labeloverlay" como en las tres técnicas de segmentación, y así se envía como una variable común a fin de que todas las técnicas operen sobre la misma imagen.

```
function [BL,i] = nombre_de_funcion(n)
nombre = sprintf('imagen_original_%d.jpg', n);
i = imread(['C:\Users\rutaBDDLocal\imagen_original \',nombre]);
nombre2 = sprintf('etiqueta_%d.png', n);
x = imread(['C:\Users\rutaBDDLocal \etiquetas\PixelLabelData\',nombre2]);
classNames = "bact";
cmap = jet(numel(classNames));
B = labeloverlay(histeq(x),x,'Transparency',0.1,'Colormap',cmap);
BL = logical(B);
end
```

En cuanto a "BL", se trata de un conjunto de valores lógicos organizados en un arreglo. Esto se debe a que, para llevar a cabo la comparación entre las técnicas de segmentación en MATLAB y el enfoque "Labeloverlay", se utiliza la función "dice". Dicha función requiere que se le suministren arreglos binarios para llevar a cabo la comparativa. Por lo tanto, "B" representa el resultado de la segmentación mediante el uso de "Labeloverlay".

"Labeloverlay" permite obtener valores lógicos de las imágenes al trabajar con labeloverlay que fusiona la imagen original con la máscara que se creó en Image Labeler como se observa en la Figura 3.1.

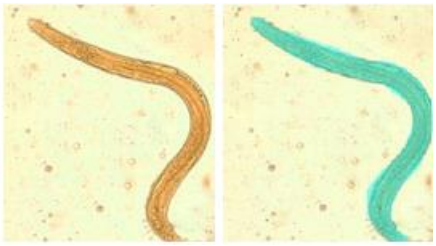

Figura 3.1 Resultado de aplicación de la función Labeloverlay.

3.1.3 Selección de forma aleatoria de imágenes

A continuación, se muestra el código para escoger de forma aleatoria, las imágenes que entran al grupo de entrenamiento y las que sirven para probar el entrenamiento.

```
function [A,B] = aleatorio(n)

n_entrenamiento = round(n*0.7);
n_pruebas = n - n_entrenamiento;
n_imagen = randperm(n);

for i = 1:n_entrenamiento
    A(i) = n_imagen(i);
end

for i = 1:n_pruebas
    B(i) = n_imagen(i+n_entrenamiento);
end
end
```

Como en el código anterior se crea una función del nombre que el usuario prefiera y “n” se refiere al número total de imágenes, por lo tanto, esta función devuelve el número de imágenes, las cuales deben sumar el 70% del total para el entrenamiento y el 30% para probar el resultado de este.

La variable “round” se utiliza para redondear el numero al entero de “n” más próximo ya que claramente se debe trabajar con números enteros, “A” continuación, se crea un vector con el subíndice aleatorio de imágenes que se usan para entrenamiento y se crea otro igualmente que se crea para pruebas de entrenamiento

Cada vez que se ejecute el código se escogen diferentes imágenes por lo tanto lo resultados siempre varían.

3.1.4 Resultados técnica Otsutresh

Para observar de mejor manera los resultados solo se trabajó con 100 imágenes en las tres técnicas de segmentación, pero en el ANEXO II se encuentra el entrenamiento con la base de datos completa de 3616 imágenes.

En la Figura 3.2 se puede observar los resultados obtenidos utilizando la técnica Otsuthresh.

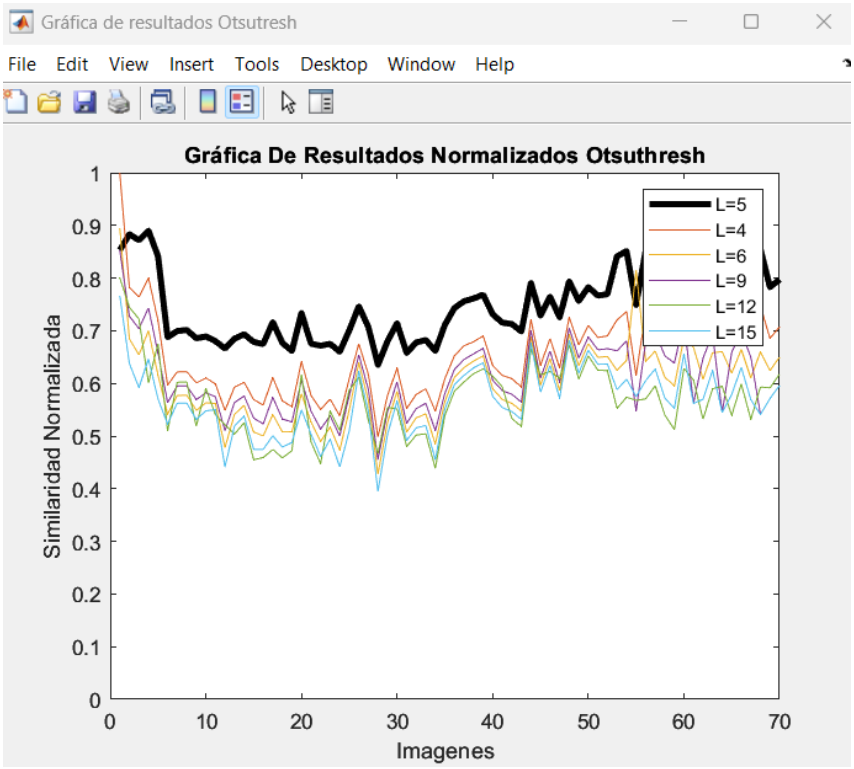

Figura 3.2 Resultados Normalizados de la técnica Otsutresh para entrenamiento

En la gráfica se observa en el eje “x” el número de imágenes con las que se realiza la comparación, mientras que en el eje “y” se observa el porcentaje normalizado de la similitud “L”, teniendo como referencia para la comparación la función Labeloverlay. Se utiliza la función “DICE” con los resultados de la segmentación usando la función Otsutresh.

similitud= DICE(BW1, BW2)

Este calcula el coeficiente de similitud de Sørensen-Dice entre imágenes binarias BW1 y BW2 [33].

La parte de código más relevante para realizar esta técnica de segmentación se muestra a continuación, cada función esta descrita detalladamente en el apartado 2.3.1.

```
I = imread('nombre_imagen.jpg');
[counts,x] = imhist(i,L);
T = otsuthresh(counts);
BW = imbinarize(i,T);
figure
imshow(BW)
```

Donde se va a entrenar el valor de "L" ya que la función Otsutresh por defecto trabaja con la función imhist, en la cual se solicita la imagen original y un valor "L" para realizar la segmentación, por defecto MATLAB tiene asignado el valor de 3, sin embargo, al aumentar este valor mejora la calidad de la segmentación, hay que tener en cuenta que se llega a un mejor resultado límite, es decir después de llegar a cierto valor la calidad de la segmentación ya no mejora como se observa en la Figura 3.3.

Al realizar el entrenamiento se obtiene que el mejor "L" tiene un valor de 5, la gráfica demuestra como al trabajar con el valor de 5 se obtiene la mejor similitud de todas las imágenes a pesar de que el valor de "L" aumente o disminuya.

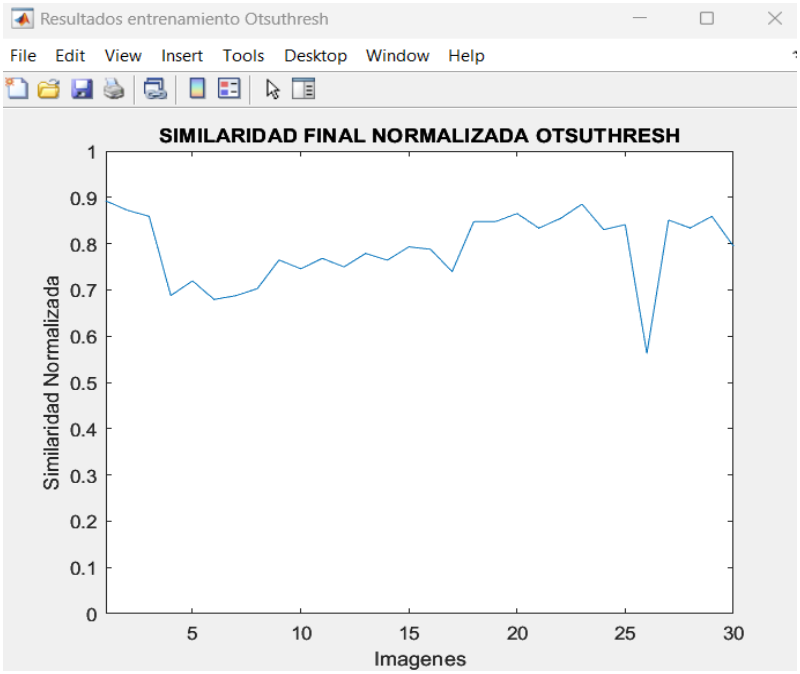

Figura 3.3 Resultados de Similaridad final Normalizado de la técnica Otsutresh para prueba de entrenamiento

En la imagen se observa la similitud con únicamente el valor óptimo con el que trabaja Otsuthresh, se puede observar que la similitud de la mayoría de las imágenes se encuentra sobre el 60%, teniendo la mayoría de similitud en el rango de 75% a 85%.

3.1.5 Resultados técnica Activecontour

El entrenamiento obtenido para la técnica Activecontour para 100 imágenes se observar en la Figura 3.4.

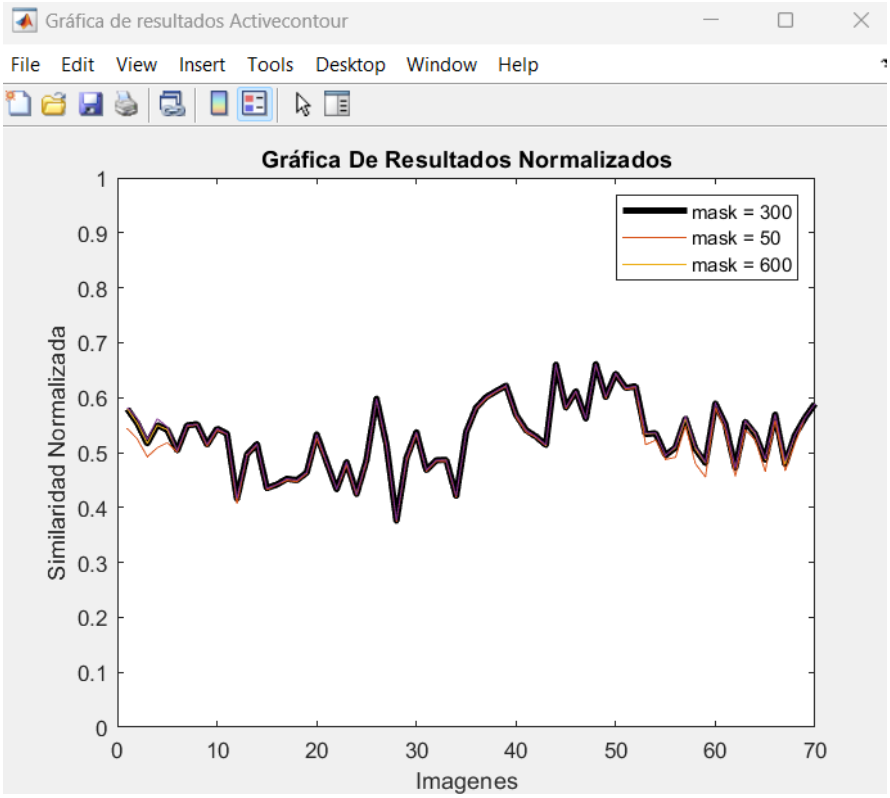

Figura 3.4 Resultados Normalizados de la técnica Activecontour para entrenamiento.

Para esta técnica de segmentación se utiliza contornos activos para segmentar una imagen en regiones de primer plano (objeto) y fondo. Este enfoque te permite delinear y separar las áreas de interés del fondo en la imagen que se requiere.

BW = activecontour(A,mask)

La imagen “A” es la imagen por segmentar y el parámetro "mask" es una imagen binaria que establece el estado inicial del contorno activo. Las fronteras de las regiones del objeto (píxeles blancos) en la máscara definen la posición inicial del contorno, que luego evoluciona con el objetivo de segmentar la imagen. El resultado es una imagen binaria de salida "BW" donde el primer plano está representado en blanco y el fondo en negro [31].

El argumento “mask” es el parámetro por entrenar en esta técnica de segmentación, tal cual sucede con el parámetro “L” para la técnica Otsuthresh.

Por defecto MATLAB posiciona este valor en 100, pero al entrenar las imágenes de la base de datos con la que se trabaja se observa que al variar el argumento en pasos de 100 se obtiene una variación, aunque no es muy prominente, por lo tanto, se obtuvo que para las 3616 imágenes con las que se trabaja el valor mas óptimo de entrenamiento fue de mask=300.

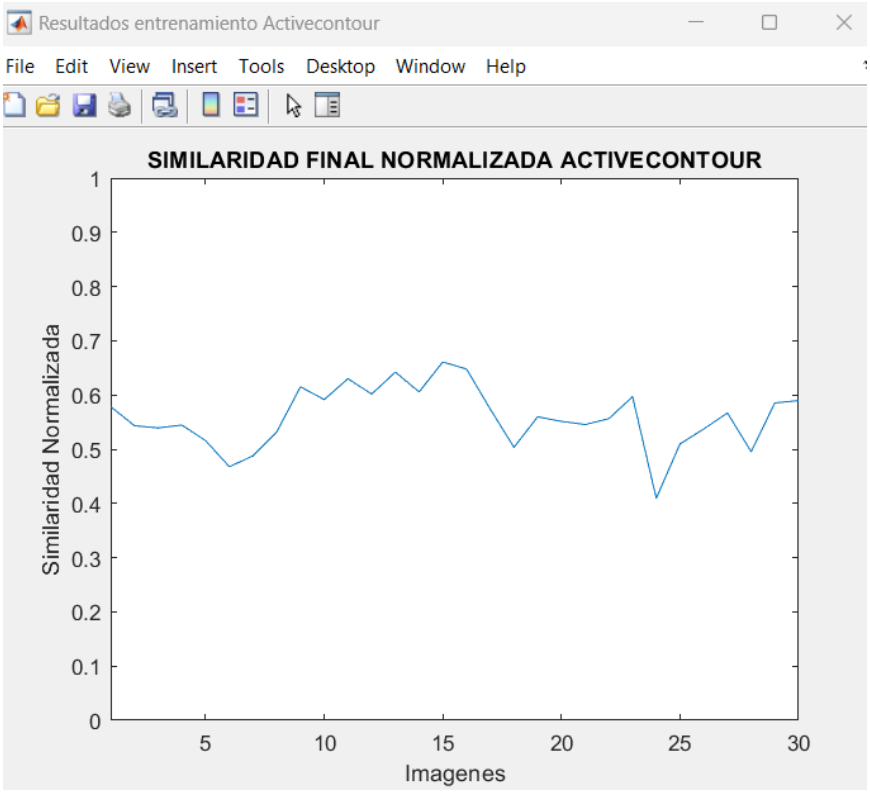

Figura 3.5 Resultados de Similaridad final Normalizado de la técnica Activecontour para prueba de entrenamiento.

La grafica muestra la similitud de la segmentación con el mejor valor de la opción “mask” con la que trabaja la función Activecontour, se observa que la mayoría de las imágenes tienen una similitud del 60%, la carga computacional para obtener los resultados es demasiado alta, es por eso por lo que se trabajó reduciendo el número de imágenes, como se puede observar los resultados no son los mejores si se toma en cuenta la carga computacional que consume esta función.

3.1.6 Resultados técnica Imsegfmm

Esta técnica usa de algoritmos y enfoques con el propósito de dividir los pixeles o áreas de una imagen en conjuntos lógicos y cohesivos, en la figura 3.6 se muestra el entrenamiento que se realizó mediante esta técnica con la base de datos de parásitos de reptiles con la que se trabaja actualmente.

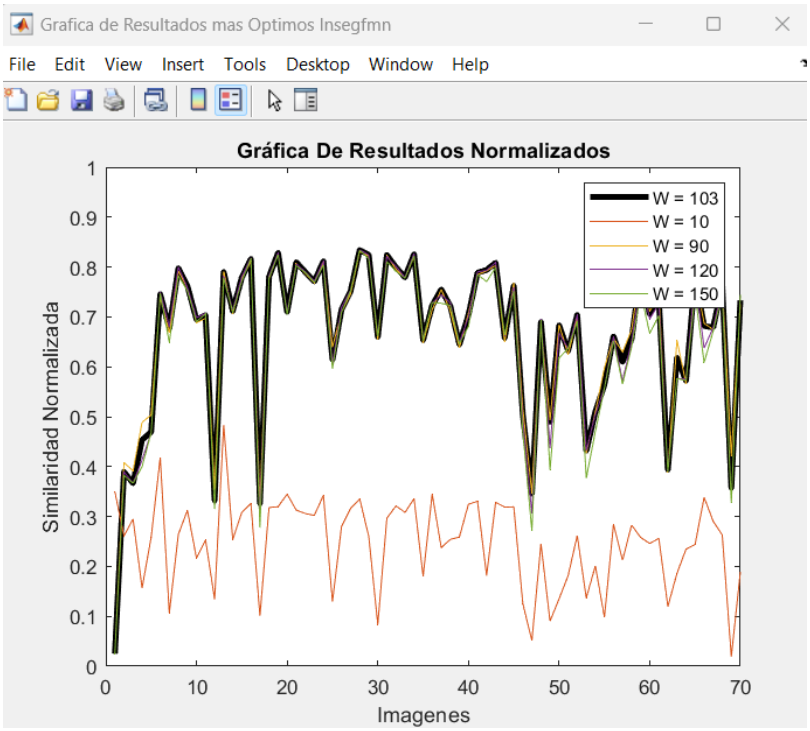

Figura 3.6 Resultados Normalizados de la técnica Imsegfmm para entrenamiento

Un ejemplo del uso de la función Imsegfmm se muestra a continuación:

```
thresh = 0.01;
[BW, D] = imsegfmm(W, mask, thresh);
figure
imshow(BW)
title('Imagen Segmentada')
```

Se puede observar que la función trabaja con tres argumentos, “mask” en este caso permite definir el contorno de la imagen, es decir su tamaño, debido a que las imágenes no eran todas del mismo tamaño se seleccionó un tamaño adecuado con la imagen más grande para no perder información con posibles recortes, el argumento “thresh” no cambia significativamente la segmentación por lo que no tiene caso variarlo, lo que se entrenó fue el valor “W” ya que se pudo observar que mientras crecía en pasos de 10 la segmentación mejoraba, hasta llegar a un límite después del cual los cambios eran despreciables.

En la imagen se observa que la mayoría de las imágenes tienen una similitud de entre el 70% y 80%, sin embargo, existen imágenes con una similitud notablemente baja, esto se debe al tamaño de la imagen, ya que algunas eran muy pequeñas y al tener un contorno fijo los errores en la segmentación crecían. La carga computacional es baja y como se observa los resultados en su mayoría son buenos como se muestra en la Figura 3.7.

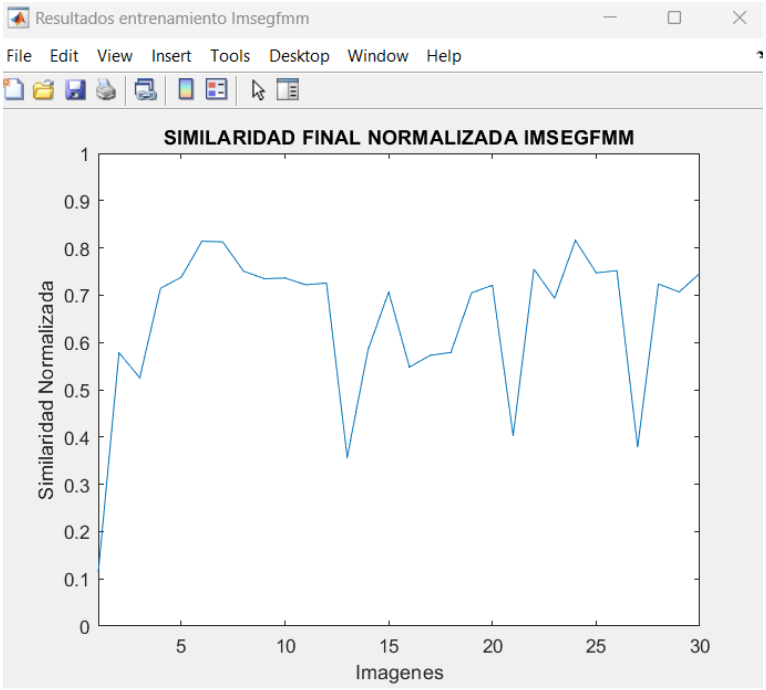

Figura 3.7 Resultados de Similaridad final Normalizado de la técnica Imsegfmm para prueba de entrenamiento.

Una vez obtenido el valor W óptimo se utiliza en las imágenes de prueba obteniendo la mayoría de las imágenes sobre el 65% de similitud, como se mencionó anteriormente existen ciertas imágenes con una similitud baja debido en gran medida a la diferencia de tamaño, aun así, los resultados obtenidos se encuentran sobre el 40%.

3.2 COMPARACIÓN DE LA MÉTRICA DICE FINAL

En la Figura 3.8 se muestra los resultados finales, obtenidos tras llevar a cabo el proceso de entrenamiento y las pruebas correspondientes. Estas pruebas se realizaron utilizando el 70% y 30 % respectivamente de la base de datos utilizada en este Trabajo de Integración Curricular.

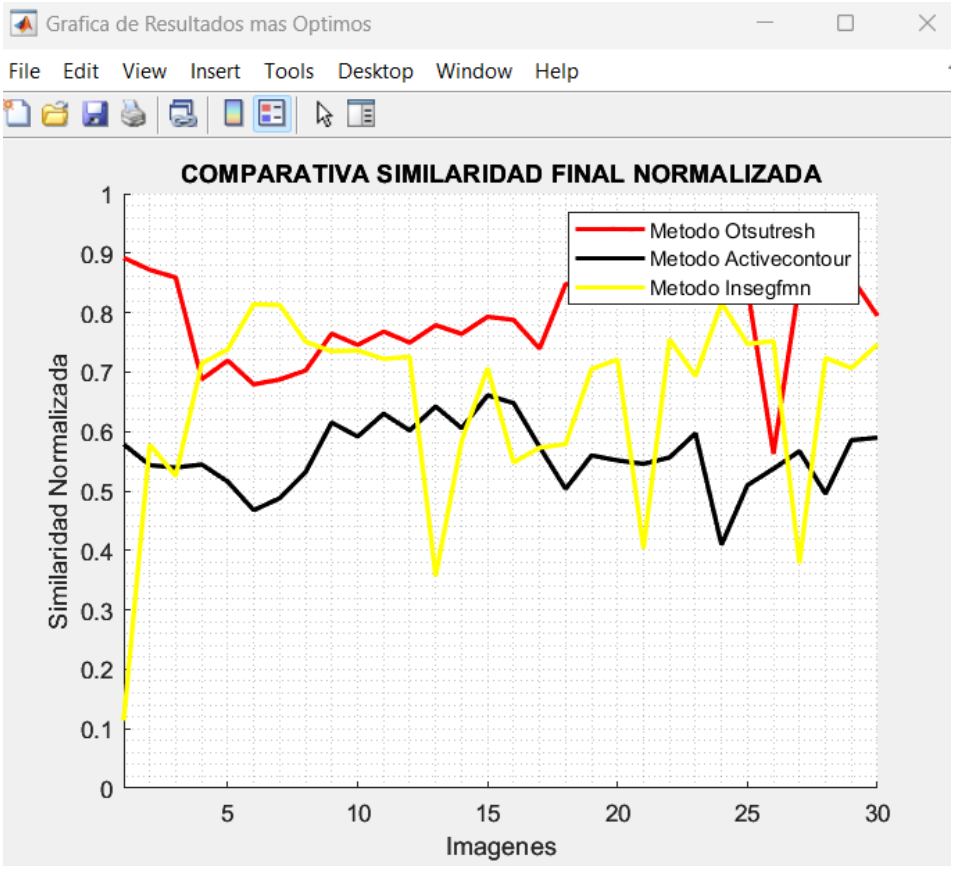

Figura 3.8 Resultados de comparación de Similaridad final Normalizada entre las tres técnicas de segmentación.

Se puede observar una comparación entre las tres técnicas, en los cuales Activecontour obtiene los peores resultados, esto sumado a la carga computacional queda como conclusión que no es recomendable para este tipo de imágenes.

Luego se encuentra la técnica Imsegfmm que obtiene buenos resultados, en tiempo de compilación tiene un tiempo igual al de Otsutresh, sin embargo, se puede observar como con Otsutresh se obtiene los mejores resultados, es decir funciona muy bien en la segmentación con un tiempo de compilación tolerable para trabajar con segmentación de imágenes, la similitud que se observa esta sobre el 70% aunque en una gran cantidad de imágenes tiene una similitud sobre el 80%.

Para finalizar se debe tener en cuenta que las imágenes segmentadas en este caso, son muy difíciles de diferenciar, las técnicas utilizadas buscan encontrar patrones para generar contornos que les permitan diferenciar objetos dentro de la imagen, si se analiza las imágenes que se encuentran en la base de datos, se puede observar cómo se mezclan los

colores dentro de las bacterias por lo que mediante algoritmos es muy difícil diferenciar contornos, la técnica basada en Labeloverlay es manejado directamente por el usuario, es decir, el usuario manualmente es el que realiza la segmentación, es por eso que se toma esta segmentación como la mejor y es la que sirve de base para comparar los resultados que se obtienen con las demás técnicas. La desventaja de esta técnica es que al ser manual si se trabaja con una gran cantidad de datos, realizar la segmentación puede tardar demasiado tiempo, por lo que con las técnicas anteriormente descritos se obtendrían resultados aceptables sin necesidad de realizar la segmentación de una imagen en una.

La Tabla 3,4, y 5 muestra los resultados obtenidos del promedio del DICE para las 3616 imágenes en cada técnica de segmentación, solamente se tomó el valor de 5 valores aleatorios ya que el tiempo de entrenamiento y la carga computacional son elevados.

Tabla 3. Resultados del DICE para la técnica Otsuthresh

Valor por entrenar "L"	DICE
3	0.8771
4	0.8002
5	0.9881
10	0.9395
12	0.7653

Tabla 4. Resultados del DICE para la Técnica Activecontour

Valor por entrenar "mask"	DICE
25	0.6532
50	0.5543
100	0.6543
300	0.7021
600	0.524

Tabla 5. Resultados del DICE para la Técnica lmsegfmm

Valor por entrenar "W"	DICE
10	0.4563
90	0.6501
103	0.9677
120	0.7643
150	0.8532

3.3 CONCLUSIONES

Al llevarse a cabo el proceso de entrenamiento utilizando el 70% de las imágenes y las pruebas con el 30%, se puede evidencia técnica de segmentación más eficiente para esta base de datos fue Otsuthresh, y se comprueba con el hecho de que obtuvo un DICE del 0.9881, lo que quiere decir que el grado de similitud mediante esta técnica es altamente efectivo.

El estudio sobre segmentación ha sido de vital importancia para poder comprender el procesamiento de imágenes, ya que es esencial para identificar y segregar las regiones de interés de una imagen (ROI), al dominar la teoría de segmentación se puede mejorar notablemente la eficiencia y especialmente la precisión para detección de objetos, evolucionando en diferentes campos como industria, medicina, etc.

La investigación de la parasitología en reptiles desempeñó un papel fundamental en este trabajo de investigación y en la mejora de las técnicas de segmentación de imágenes, ya que existe una conexión entre estos dos campos aparentemente diferentes pero esta relación ayuda a una interpretación y análisis de imágenes más precisos, ya que al estudiar los efectos de los parásitos en reptiles, se pueden identificar características específicas en las imágenes de estos animales que pudieron haber pasado inadvertidos, y así ayudar en campos como investigación bilógica, medicina veterinaria y conservación de la vida silvestre.

El presente proyecto puede ser la base aplicable a otro tipo de afecciones animales, y no se limita solo a parásitos, lo cual puede servir en otro tipo de enfermedades que aquejen a los animales en cautiverio, considerando que al ser escalable el mismo puede tener ramificaciones que permitan expandir su usa incluso fuera del ámbito veterinario.

Los resultados obtenidos a partir del modelo son positivos ya que la mayoría de los parásitos se detectaron de manera correcta y en los casos que no se detectaron (errores) se debió a la calidad de la imagen y/o una forma no entrenada de la imagen del parásito.

Los buenos resultados se basan en las imágenes que se ha utilizado para entrenar el sistema, por lo cual al aumentar el insumo de entrenamiento el proyecto puede usarse como propuesta para diferentes escenarios.

El código puede ser empleado (luego de una capacitación adecuada), en múltiples escenarios de detección y seguimiento de parásitos en diferentes tipos de animales, reduciendo la subjetividad del evaluador humano durante la evaluación médica y optimizando los tiempos de operación y/o monitoreo.

A partir de la recopilación de imágenes nuevas adjuntas a la ya existentes, se puede monitorear con mayor precisión la presencia o desarrollo de parásitos, detectar las enfermedades, estimar tiempos de recuperación, y aplicar tratamientos efectivos de desparasitación.

3.4 RECOMENDACIONES

Organizar, verificar, validar y depurar y verificar la base de datos para trabajar de una manera más eficiente ya que la base de datos obtenida al principio no tenía los nombres y números organizados, por lo tanto, si dos imágenes repiten su mismo nombre habrá contratiempos y la validación del código será ineficaz.

Utilizar las mismas dimensiones de las imágenes para que el espectador y/o usuario pueda apreciar de una manera detallada las imágenes y regiones de interés.

Verificar si las imágenes son 2D o 3D ya que algunas técnicas como: imsegkmeans3, superpixels3 solo trabajan en 3 dimensiones, así se podrá elegir la mejor manera para trabajar.

Se recomienda guardar el progreso ya que cuando se cargan demasiadas imágenes el programa suele fallar y se debe volver a empezar todo el trabajo, pero si se guarda cada cierto tiempo Image Labeler guarda y carga las imágenes y etiquetas ya creadas.

Para utilizar el código con una grande cantidad de datos es necesario contar con los recursos necesarios, es decir tener equipos con gran capacidad computacional, ya que, si no se cuenta con los mismo, el entrenamiento y ejecución del código tardará demasiado tiempo y es ineficaz porque tal vez el computar sufra daño.

Se debe tomar en cuenta que este tipo de trabajos deben ser aplicados en conjuntos de datos de gran volumen para que el entrenamiento sea eficiente, efectivo y óptimo.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] «Reptiles Ecuador». https://bioweb.bio/faunaweb/reptiliaweb/ (accedido 14 de agosto de 2023).

[2] M. E. Rodriguez, *Identificación de parásitos intestinales presentes en reptiles en cautiverio en dos centros de manejo de fauna silvestre*. Unuversidad Central del Ecuador, 2015.

[3] S. de S. Rodrigues *et al.*, «Avaliação coproparasitológica de Chelonoidis carbonaria, Spix, 1824 (Reptilia, Testudinidae) em cativeiro no Espírito Santo.».

[4] Most. M. Khatun, N. Begum, Md. A. A. Mamun, Md. M. H. Mondal, y Md. Shakif-Ul-Azam, «Coprological study of gastrointestinal parasites of captive animals at Rangpur Recreational Garden and Zoo in Bangladesh», *J. Threat. Taxa*, vol. 6, n.º 8, pp. 6142-6147, jul. 2014, doi: 10.11609/JoTT.o3093.6142-7.

[5] J. R. H. Ramírez, «Estudio patológico retrospectivo de mortalidad en reptiles del zoológico Jaime Duque entre el año 1991 y el 2006», 1991.

[6] K. A. N. Alverca, «“Identificación de parásitos con diferentes métodos coprológicos en muestras de reptiles en el vivarium de quito”», p. 93, 2021.

[7] Enrique Pardo Cobas y Buitrago Martha, «PARASITOLOGIA VETERINARIA», p. 125, jul. 2005.

[8] Corporación Autónoma Regional de Cundinamarca, «Graves consecuencias que sufre un animal silvestre cuando ha estado en cautiverio». https://www.car.gov.co/saladeprensa/estas-son-las-graves-consecuencias-que-sufre-un-animal-silvestre-cuando-ha-estado-en-cautiverio (accedido 14 de agosto de 2023).

[9] Dra. Hortensia Magaró *et al.*, «TÉCNICAS DE DIAGNÓSTICO PARASITOLÓGICO», p. 21.

[10] Dra. Nora La Serna Palomino y Lic. Ulises Román Concha1, «Técnicas de Segmentación en Procesamiento Digital de Imágenes», n.º 8, [En línea]. Disponible en: {nlasernap, uromanc}@unmsm.edu.pe

[11] D. Ortega y A. Iznaga Benítez, *Técnicas de Segmentación de Imágenes Médicas*. 2008.

[12] María Elena Cruz Meza, «Análisis de Imágenes».

[13] Nobuyuki Otsu: A threshold, «Selection method from grey level histograms». New York . , S.62–66. ISSN 1083-4419 de 1979.

[14] «La umbralización». https://www.lpi.tel.uva.es/~nacho/docencia/ing_ond_1/trabajos_03_04/sonificacion/cabroa_archivos/umbralizacion.html (accedido 14 de agosto de 2023).

[15] C. A. Cattaneo, L. I. Larcher, A. I. Ruggeri, A. C. Herrera, y M. Biasoni, «MÉTODOS DE UMBRALIZACIÓN DE IMÁGENES DIGITALES BASADOS EN ENTROPIA DE SHANNON Y OTROS», 2011.

[16] J. T. Pacheco, «Segmentación asistida usando bordes, textura y color», dic. 2007.

[17] Nora La Serna Palomino y Luzmila Pró Concepción, «Watershed: un algoritmo eficiente y flexible para segmentación de imágenes de geles 2-DE», vol. 7, dic. 2010.

[18] C. Rother, V. Kolmogorov, y A. Blake, «GrabCut: Extracción interactiva en primer plano mediante cortes de gráfico iterados , ACM Trans.», vol. 23, pp. 309-314, 2004.

[19] «Segmentación (procesamiento de imágenes)», *Wikipedia, la enciclopedia libre*. 12 de julio de 2023. Accedido: 14 de agosto de 2023. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Segmentaci%C3%B3n_(procesamiento_de_im%C3%A1genes)&oldid=152427375

[20] «3.16 - Segmentación de imágenes con Redes Convolucionales», *Codificando Bits*. https://www.codificandobits.com/curso/fundamentos-deep-learning-python/redes-convolucionales-16-segmentacion-imagenes/ (accedido 14 de agosto de 2023).

[21] «Segmentación semántica». https://la.mathworks.com/solutions/image-video-processing/semantic-segmentation.html (accedido 14 de agosto de 2023).

[22] J. M. Hidalgo, «CNN Segmentación Binaria», *Deepnote*. https://deepnote.com/@julianmelero_/CNN-Segmentacion-Binaria-15b6233b-bb10-4970-946c-2b12db61f12c (accedido 14 de agosto de 2023).

[23] C. I. Espinosa, *«Índices de Similitud» Similitud de Comunidades biológicas*. Accedido: 14 de agosto de 2023. [En línea]. Disponible en: https://ciespinosa.github.io/Similitud/%C3%ADndices-de-similitud.html

[24] J. V.L y R. Gopikakumari, «IEM: A New Image Enhancement Metric for Contrast and Sharpness Measurements», *Int. J. Comput. Appl.*, vol. 79, n.º 9, pp. 1-9, oct. 2013, doi: 10.5120/13766-1620.

[25] «Segmentación de imágenes - MATLAB & Simulink - MathWorks América Latina». https://la.mathworks.com/help/images/image-segmentation.html (accedido 21 de agosto de 2023).

[26] Andrés Felipe Ramírez Corrales, «Segmentación Asistida de Imágenes Usando Características de Textura», p. 10.

[27] A. M. Martínez, «Herramientas MATLAB para el tratamiento de imágenes».

[28] «Documentación de la caja de herramientas de visión artificial». https://www-mathworks-com.translate.goog/help/vision/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc (accedido 14 de agosto de 2023).

[29] «Machine Vision Toolbox», *Peter Corke*. https://petercorke.com/toolboxes/machine-vision-toolbox/ (accedido 14 de agosto de 2023).

[30] «Introducción al etiquetador de imágenes - MATLAB & Simulink». https://www-mathworks-com.translate.goog/help/vision/ug/get-started-with-the-image-labeler.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc (accedido 14 de agosto de 2023).

[31] «Segment image into foreground and background using active contours (snakes) region growing technique - MATLAB activecontour». https://www-mathworks-com.translate.goog/help/images/ref/activecontour.html?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc (accedido 14 de agosto de 2023).

[32] «Binary image segmentation using fast marching method - MATLAB imsegfmm». https://www-mathworks-com.translate.goog/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc&_x_tr_hist=true#d124e194260 (accedido 14 de agosto de 2023).

[33] «Coeficiente de similitud de Sørensen-Dice para segmentación de imágenes - MATLAB dice - MathWorks América Latina». https://la.mathworks.com/help/images/ref/dice.html#mw_3ecbfd75-2b6d-4464-936e-eb138389b8c9 (accedido 15 de agosto de 2023).

5 ANEXOS

5.1 ANEXO I: CÓDIGO IMPLEMENTADO

```
%% Funcio generacion de imagenes
function [BL,i] = love(n) %i es la imagen n = numero imagen
%   Funcion que permite obtener los valores logicos de las imagenes ...
%   cuando se trabaja con labeloverlay
nombre = sprintf('bact_%d.jpg', n);
%nombre = sprintf('ophi (%d).jpg', n);
i  =  imread(['C:\Users\  rutaBDDLocal\imagen_original\',nombre]);  %lee  las
imagenes originales
nombre2 = sprintf('Label_%d.png', n);
x = imread(['C:\Users\rutaBDDLocal \etiquetas\PixelLabelData\',nombre2]);
classNames = "bact";
cmap = jet(numel(classNames));
B = labeloverlay(histeq(x),x,'Transparency',0.1,'Colormap',cmap);
BL = logical(B); %La variable BL es la que se va a comparar, el metodo dice
compara binarios
```

```
%Mediante la funcion logical conviertimos B en un arreglo de valores
%lógicos entre 0 y 1
End
```

```
%% Funcio generacion de imagenes
function [BL,i] = love(n) %i es la imagen n = numero imagen
%   Funcion que permite obtener los valores logicos de las imagenes ...
%   cuando se trabaja con labeloverlay
nombre = sprintf('bact_%d.jpg', n);
%nombre = sprintf('ophi (%d).jpg', n);
i = imread(['C:\Users\ rutaBDDLocal\imagen_original\',nombre]); %lee las
imagenes originales
nombre2 = sprintf('Label_%d.png', n);
x = imread(['C:\Users\rutaBDDLocal \etiquetas\PixelLabelData\',nombre2]);
classNames = "bact";
cmap = jet(numel(classNames));
B = labeloverlay(histeq(x),x,'Transparency',0.1,'Colormap',cmap);
BL = logical(B); %La variable BL es la que se va a comparar, el metodo dice
compara binarios
%Mediante la funcion logical conviertimos B en un arreglo de valores
%lógicos entre 0 y 1
end
```

```
%% Valores para escoger imagenes de manera aleatoria
function [A,B] = aleatorio(n)
%Esta funcion devuelve el numero de la imagen, las cuales deben sumar el
%70% para entrenamiento y 30% para probar el resultado del entrenamiento

n_entrenamiento = round(n*0.7); %Calcula el 70% del numero de imagenes
n_pruebas = n - n_entrenamiento; %Calcula el 30% del numero de imagenes
%Valores aleatorios de imagenes
n_imagen = randperm(n);

%Crea el vector con el subindice aleatorio de imagenes que se usan para
%entrenamiento
for i = 1:n_entrenamiento
    A(i) = n_imagen(i);
end

%Crea el vector con el subindice aleatorio de imagenes que se usan para
%pruebas
for i = 1:n_pruebas
    B(i) = n_imagen(i+n_entrenamiento);
end
end
```

```
 %% Entrenamiento Otsutresh
r, clc, close all;
 %Automaticamente al azar se seleccionan el numero de imagenes para el
 %entrenamiento del modo Otsu
```

```
numero_imagenes = 3616; %Cambiar n
[A,B] =aleatorio(numero_imagenes);
a = sort(A); %Indices aleatorios ordenados de imagenes a entrenar
b = sort(B); %Indices aleatorios ordenados de imagenes a entrenar

for n = 1:length(A)
    [BL,i] = love(a(n)); %Funcion que llama en cada iteracion a las imagenes
creadas mediante ImagenLabel
    %Creo variables para entrenamiento
    %Las variables se iran actualizando cada vez que se cumpla la condicion
    %que se describe a continuacion, esto permite obtener los mejores
    %resultados
    sim2max = 0;
    k = 1;  %numero de comparaciones
    j = 1;
    l = 4; %Valor a entrenar ya que la la funcion Otsutresh por defecto
trabaja con la funcion imhist, en la cual
    %se solicita la imagen original y un valor l que le va a permitir
    %realizar la segmentacion, por defecto este valor es de 3 pero se lo
    %puede ir variando, hay imagenes que al aumentar este valor arroja una
    %mejor comparacion es por eso que se va a seleccionar el mejor valor
    lmax = 0;
    sim2 = 0;
    m = 1;
    %El lazo va a realizar quince comparaciones de resultados hasta
    %encontrar que por mas que se suba el valor de l ningun otro resultado
    %va a ser mejor
    while k <= 15

        %imhist: especifica el número de bins, l, utilizado para calcular el
histograma.
        counts = imhist(i,l); %Funcion imhist que solicita por defecto
Otsutresh para realizar la segmentacion de la imagen
        T = otsuthresh(counts); %tecnica usado para la segmentacion
        BO = imbinarize(i,T); %crea una imagen binaria a partir de la imagen
i para la comparacion con BL
        sim2(j) = dice(BL,BO); %Se crea un vector con el resultado de la
comparacion entre ambos resultados de segmentacion
        %Almaveno valores maximos dentro de vector lmax
        if sim2(j) >= sim2max
            sim2max = sim2(j); %Se crea el resultado que va a ser comparado
denro del vector
            %Con los mejores resultados y los va actualizando
            if sim2(j) == sim2max
                %Se almacena el l con el mejor resultado
                lmax(m) = l;
                m = m + 1;
            end
            k = 1; %si el siguiente valor de comparacion es mayor k se setea
            %nuevamente en uno para realizar las comparaciones nuevamente
        end
        %Las variables se van actualizando para cumplir las condiciones y
        %terminar el lazo
        j = j + 1; %Variable para vector con resultados a comparar

        %Si el siguiente valor de comparacion no es mayor k crece en
        %1 de esta forma si se compara 15 veces y ningun valor es mayor
        %al ya obtenido se termina el lazo
```

```
        k = k + 1;
        l = l + 1; %Variable l que va creciendo si la imagen va mejorando los
resulatados
    end

    l_final(n) = max(lmax); %Se crea un vecto unicamente con el valor de l
maximo, ya que
    %existen valores de l que arrojan el mismo resultado por eso se
    %almaacenan en el vector de resultados optimos, sin embargo se usa
    %unicamente el mayor ya que este asegura que no existe un valor mas
    %grande que me arroje un mejor resultado

end
%Se termina el entrenamiento, se saca la media de cada valor de l optimo
%obtenido para cada imagen y se la vuelve entero ya que la funcion imhist
%no funciona con decimales
l_opt = round(mean(l_final));
```

```
%% Gráfica de resultados Otsu
%Seleccionamos una sola vez los valores de l ya que existen muchas imagenes
%que repiten el mejor l en sus mejores resultados
final_comp = unique(l_final);
valores = length(final_comp);
%Demostrar porque l_opt es la mejor opcion

%Con todos los resultados que se obtuvo del entrenamiento se grafica cada
%imagen
for mg = 1:valores
    for n = 1:length(A)
        [BL,i] = love(a(n));
        %Se realiza todo el proceso Otsutresh con todos los valores l, es
        %decir con los l mas optimos que arrojo cada imagen
        counts = imhist(i,final_comp(mg));
        T = otsuthresh(counts);
        BX = imbinarize(i,T);
        similaridad(n) = dice(BL,BX);
    end
    valores_grafica(mg,:) = similaridad;
end

%Valor más óptimo
for n = 1:length(A)
    [BL,i] = love(a(n));
    %Se realiza todo el proceso Otsutresh pero esta vez se utiliza
    %unicamente el valor optimo de l que se obtuvo como la media de los
    %mejores l de cada imagen
    counts = imhist(i,l_opt);
    T = otsuthresh(counts);
    BX = imbinarize(i,T);
    similaridad_final(n) = dice(BL,BX);
end

%Gráfica de resultados

%En las graficas normalizadas, se demuestra como con el valor de l optimo
```

```
%la grafica posee una mayor area bajo la curva que con cualquier otro l

%Se realiza la grafica de todas las comparaciones obtenidas con el valor
%optimo de l para cada imagen
figure('Name','Gráfica de resultados Otsutresh','NumberTitle','off');
plot(1:length(A),similaridad_final,'black','LineWidth',3)

%Se realizan la grafica de los mejores l obtenidos de cada imagen en todas
%las imagenes
for n = 1:valores
    hold on;
    plot(1:length(A),valores_grafica(n,:))
end
title('Gráfica De Resultados Normalizados'),ylim([0.5 1])
legend('Valor Óptimo'),xlabel('Imagenes'),ylabel('Similaridad Normalizada');
```

```
%% Resultados entrenamiento Otsutresh

for n = 1:length(B)
    [BL,i] = love(b(n));
    %Proceso Otsutres unicamente usando el l optimo
    counts = imhist(i,l_opt);
    T = otsuthresh(counts);
    BX = imbinarize(i,T);
    similaridad_final_ots(n) = dice(BL,BX);
end

%Se realiza la grafica del resultado de las comparaciones que son arrojadas
%en porcentaje y son normalizadas

figure('Name','Resultados entrenamiento Otsu','NumberTitle','off');
plot(1:length(B),similaridad_final_ots); ylim([0,1]),xlim([1,length(B)])
title('SIMILARIDAD FINAL NORMALIZADA
OTSUTRESH'),xlabel('Imagenes'),ylabel('Similaridad Normalizada');
```

```
for n = 1:length(A)
    [BL,i] = love(a(n)); %Llamamos a la imagen y su valor binario para cada
iteracion
    %Variables solicitadas por defecto por la función activecontour
    mask = zeros(size(i));
    mask(1:end,1:end) = 1;
    %Se crea las variables de control del entrenamiento
    j = 1; %Variable para el vector donde se almacenan los mejores resultados
    k = 0; %Numero maximo de comparaciones
    sim2 = 0;
    sim2max = 0;
    mascara = 100; %Variable que se va a entrenar
    while k < 2 %El numero de comparaciones es bajo debido a la carga
computacional
        %tecnica Activecontour
        BA = activecontour(i,mask,mascara);
        sim2(j) = dice(BL,BA); %Se realiza la comparacion entre ambos
resultados
```

```
        %Se controla el valor de las comparaciones, siempre y cuando se
        %haya subido el valor de la mascara y el resultado no sea mejor el
        %valor de la comparacion aumenta
        if sim2(j) > sim2max
            k = k + 1;
        end
        mascara = mascara + 100; %La mascara va creciendo de 100 en 100 ya
que si se le aumenta
        %Valores pequeños los resultados no cambian
        j = j + 1;
    end
    mascara_max(n) = mascara; %Se crea un vetor con los resultados de la
mascara de cada
    %imagen mas optima
end

%Con las mejores msacaras de cada imagen se saca un promedio y se lo
%redondea ya que activecontour no funciona con decimales
mask_opt = round(mean(mascara_max));
```

```
%% Gráfica de resultados Activecontour

valores = [mask_opt-100 mask_opt mask_opt+100];

%Se realiza para cada imagen la grafica con los tres valores de msacaras
%definidos previamente
for mg = 1:length(valores)
    for n = 1:length(A)
        [BL,i] = love(a(n));
        %Se realiza todo el porces activecontour con los distintos valores
        %de mascara
        mask = zeros(size(i));
        mask(1:end,1:end) = 1;
        BA = activecontour(i,mask,valores(mg));
        similaridad_act(n) = dice(BL,BA);
    end
    valores_grafica(mg,:) = similaridad_act; %valores almacenados en matriz
para
    %posteriormente ser graficados con cada valor de mascara
end

%Valor más óptimo
%Se realiza el almacenamiento de datos unicamente con el valor de mascara
%mas optimo para compararlo con los otros valores
for n = 1:length(A)
    [BL,i] = love(a(n));
    %Se realiza todo el proceso activecontour
    mask = zeros(size(i));
    mask(1:end,1:end) = 1;
    BA = activecontour(i,mask,mask_opt);
    similaridad_act(n) = dice(BL,BA);
end

%Gráfica de resultados
%Se realiza la grafica de los resultados de la comparacion obtenidas con el
```

```
%valor mas optimo

figure('Name','Gráfica de resultados Activecontour','NumberTitle','off');
plot(1:length(A),similaridad_act,'black','LineWidth',3)

%Se realiza la grafica con todos los valores de las mejores mascaras de
%cada imagen
for n = 1:length(valores)
    hold on;
    plot(1:length(A),valores_grafica(n,:))
end
title('Gráfica De Resultados Normalizados'),ylim([0 1]);
legend('Valor Óptimo'),xlabel('Imagenes'),ylabel('Similaridad Normalizada');
```

```
%% Resultados entrenamiento Activecontour

%Se  realiza la comparacion de las imagenes usando unicamente el valor de
%la mascara mas optima dentro de la funcion acivecontour

for n = 1:length(B)
    [BL,i] = love(b(n));
    %tecnica activecontour
    mask = zeros(size(i));
    mask(1:end,1:end) = 1;
    BA = activecontour(i,mask,mask_opt);
    similaridad_final_act(n) = dice(BL,BA);
end

%Se grafica unicamente los resultados de la comparacion con las mascara mas
%optima del tecnica activecontour, los resultados son en porcentaje pero se
%encuentran normalizados

figure('Name','Resultados entrenamiento Activecontour','NumberTitle','off');
plot(1:length(B),similaridad_final_act),ylim([0 1]), xlim([1 length(B)]);
title('SIMILARIDAD FINAL NORMALIZADA
ACTIVECONTOUR'),xlabel('Imagenes'),ylabel('Similaridad Normalizada');
```

```
%% tecnica de Imsegfmm

%imsegfmn es una tecnica de segmentación de imágenes binarias mediante el
método de marcha rápida
for n = 1:length(A)
    [BL,i] = love(a(n));
    %Variables que utiliza la tecnica imsegfnm como requisitos para poder
    %leer adecuadamente la imagen para segmentarla
    mask = false(size(i));
    mask(170,70) = true;
    %Declaracion de variables para control de comparaciones y crecimiento
    %del vector donde se almacenan los resultados de cada comparacion
    sim2max = 0;
    k = 1; %Controla el numero de comparaciones
    j = 1;
```

```
    l = 10; %El valor optimizado, crecera de 10 en 10 ya que si crece de
forma mas pequeña los
    %cambios en el resultado final no tienen mayor diferencia
    lmax = 0; %Variable que sera optimizada como la mejor
    sim2 = 0;
    m = 1;
    while k <= 4
        %la tecnica imsegfcmn utiliza por defecto la funcion graydiffweight
        %la cual alcula los pesos de los píxeles de la imagen en función de
la diferencia
        %de intensidad en escala de grises, la funcion imsegfcmn mejora
        %dependiendo del resultado de graydiffweight el cual varia
        %dependiendo de cuanto cambie el valor de l, es por eso que esta es
        %la variable que va a ser entrenada
        W = graydiffweight(i, mask, 'GrayDifferenceCutoff', l);
        thresh = 0.01;
        %Funcion imsegfmn con las variables solicitadas
        [BW_I, D] = imsegfmm(W, mask, thresh);
        sim2(j) = dice(BL,BW_I); %Se realiza la comparacion y se almacena en
un vector
        %el cual va ir cambiando siempre y cuando el vlor de comparacion
        %sea mayor que el anterior
        if sim2(j) >= sim2max
            sim2max = sim2(j);
            if sim2(j) == sim2max
                lmax(m) = l;
                m = m + 1;
            end
            k = 1; %Si el siguiente valor crece la variable de comparacion se
setea nuevamente en 1
            %para comparar nuevamente desde el inicio
        end

        %Si el siguiente valor de comparacion no es mayor k crece en
        %1 de esta forma si se compara 15 veces y ningun valor es mayor
        %al ya obtenido se termina el lazo
        j = j + 1;
        k = k + 1;
        l = l + 10;

        %Debido a que las comparaciones creciendo el valor de l pueden ser
        %infinitas con cambios minimos, se pone una condicion de limite en
        %donde si llega a 80 el valor de l las comparaciones terminan
        if lmax > 80
            k = 5;
        end
    end
    l_final(n) = max(lmax); %los distinos valores de l almacenados, se
selleciona solo el
    %mayor ya que con esto se garantiza que no hay un l mas grande que me
    %de mejores resultados
end

%se saca un promedio de todos los valores de l obtenidos de cada imagen y
%se redondea para trabajar con enteros
l_opt = round(mean(l_final));
```

```
%% Gráfica de resultados Imsegfmm

%Los l optimos se repiten en diferentes imagenes por lo que aqui se
%selecciona una unica vez la l repetida y se la almacena
final_comp = unique(l_final);
valores = length(final_comp);

%se reailiza la grafica con todos los valores de l encontrados para todas
%las imagenes
for mg = 1:valores
    for n = 1:length(A)
        [BL,i] = love(a(n));
        %tecnica imsegfmnm completo
        mask = false(size(i));
        mask(170,70) = true;
        W = graydiffweight(i, mask, 'GrayDifferenceCutoff', final_comp(mg));
        thresh = 0.01;
        [BW_I, D] = imsegfmm(W, mask, thresh);
        similaridad(n) = dice(BL,BW_I);
    end
    valores_grafica(mg,:) = similaridad; %valores de similaridad de cada
imagen
    %almacenados para ser graficado
end

%Valor más óptimo
%Se realiza con el l mas optimo el proces do imsegfmn para obtener la
%similaridad y poder graficarla
for n = 1:length(A)
    [BL,i] = love(a(n));

    mask = false(size(i));
    mask(170,70) = true;
    W = graydiffweight(i, mask, 'GrayDifferenceCutoff', l_opt);
    thresh = 0.01;
    [BW_I, D] = imsegfmm(W, mask, thresh);
    similaridad_i(n) = dice(BL,BW_I);
end

%Gráfica de resultados
%Con los datos obtenidos con el l mas optimo se realiza la grafica
figure('Name','Grafica de Resultados mas Optimos
Insegfmn','NumberTitle','off');
plot(1:length(A),similaridad_i,'black','LineWidth',3)

%Se grafica la similaridad de todas las imagenes usando todos los l
for n = 1:valores
    hold on;
    plot(1:length(A),valores_grafica(n,:))
end
title('Gráfica De Resultados Normalizados'),ylim([0 1])
legend('Valor Óptimo'),xlabel('Imagenes'),ylabel('Similaridad Normalizada');
```

```
%% Resultados entrenamiento Imsegfmm

%Se utiliza el tecnica insegfmn unicamente con el l mas optimo

for n = 1:length(B)
    [BL,i] = love(b(n));
    %tecnica imsegfnm
    mask = false(size(i));
    mask(170,70) = true;
    W = graydiffweight(i, mask, 'GrayDifferenceCutoff', l_opt);
    thresh = 0.01;
    [BW_I, D] = imsegfmm(W, mask, thresh);
    similaridad_final_in(n) = dice(BL,BW_I);
end

%Grafica de similaridades con el mejor l
figure('Name','Resultados entrenamiento Imsegfmm','NumberTitle','off');
plot(1:length(B),similaridad_final_in); ylim([0,1]),xlim([1,length(B)])
title('SIMILARIDAD FINAL NORMALIZADA
IMSEGFMM'),xlabel('Imagenes'),ylabel('Similaridad Normalizada');
```

Código para comparación, obteniendo la técnica más optima

```
%% Grafica de resultados mas optimos
figure('Name','Grafica de Resultados mas Optimos','NumberTitle','off');
hold on
plot(1:length(B),similaridad_final_ots,'r','LineWidth',2)
plot(1:length(B),similaridad_final_act,'black','LineWidth',2)
plot(1:length(B),similaridad_final_in,'y','LineWidth',2)
hold off
legend('Metodo Otsutresh','Metodo Activecontour','Metodo
Insegfmn'),ylim([0,1]),xlim([1,length(B)])
title('COMPARATIVA SIMILARIDAD FINAL
NORMALIZADA'),xlabel('Imagenes'),ylabel('Similaridad Normalizada');
grid minor
```

5.2 ANEXO II. Métrica DICE utilizando toda la base de datos

A continuación, se muestran los resultados finales utilizando las 3616 imágenes de la base de datos.

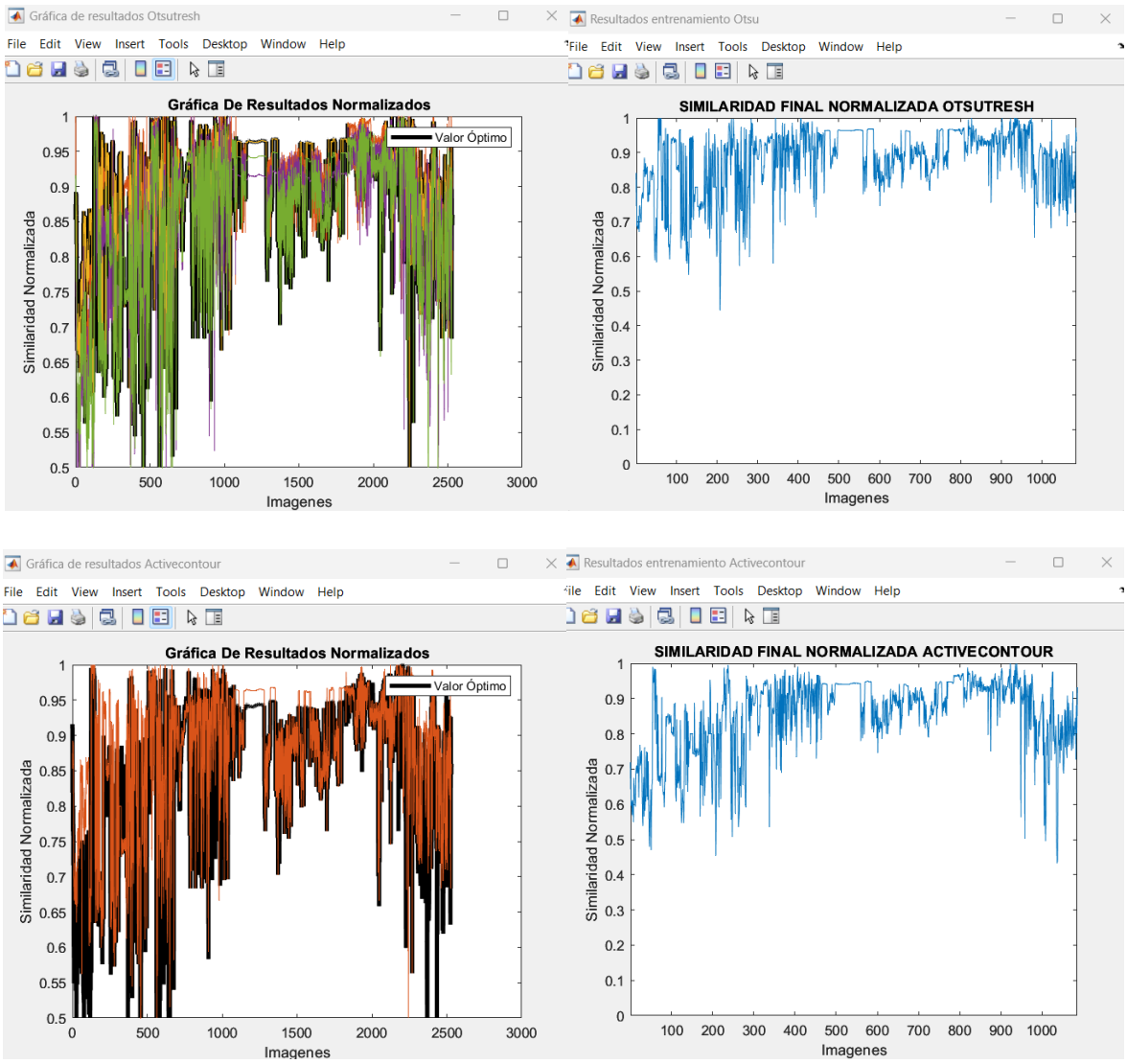

Figura 5.1 Resultados finales del entrenamiento y prueba de las técnicas de segmentación utilizadas