Universidad Autónoma de Yucatán
Facultad de Matemáticas

Maestría en Ciencias de la Computación

Tesis de Maestría

Algoritmos de segmentación de *Trypanosoma cruzi* en imágenes de muestras sanguíneas

Roger David Soberanis Mukul

Asesores

Dr. Víctor Uc Cetina

Dr. Carlos Brito Loeza

Mérida, Yucatán, México

Diciembre 2014

Dedicado a mis padres y hermanas.

Agradecimientos

Quiero presentar mis agradecimientos a los Drs. Víctor Uc y Carlos Brito, quienes me asesoraron durante el desarrollo de este proyecto. Al Dr. Hugo Ruiz, profesor investigador del Centro de Investigaciones Regionales Dr. Hideyo Noguchi, por las imágenes digitales para el desarrollo de este proyecto. A si mismo, presento mi agradecimiento a los profesores con los que tuve la oportunidad de cursar materias durante el transcurso de la maestría.

Resumen

La Tripanosomiasis Americana o Enfermedad de Chagas, es un padecimiento crónico presente en América Latina. Es causada por el protozoario parásito *Trypanosoma cruzi* el cual se introduce al cuerpo humano mediante la contaminación con el excremento u orina de insectos hematófagos de la subfamilia *Triatominae*. El diagnóstico se puede realizar mediante el análisis de muestras sanguíneas, debido a que el parásito pasa las primeras etapas de la enfermedad en el torrente sanguíneo del portador. En este trabajo, se comparan diversos algoritmos para la segmentación de *Trypanosoma cruzi* en imágenes de muestras sanguíneas. De forma similar, se propone un algoritmo para la segmentación de este parásito. El problema de segmentación se plantea como un problema de clasificación binaria sobre los pixeles de la imagen, para lo cual se propone el uso de algoritmos de clasificación junto con la información extraída a partir de conjuntos de pixeles conocidos como súper pixeles. Para esto, se evaluaron tres clasificadores: máquinas de vectores de apoyo, redes neuronales y AdaBoost. Como resultado de este trabajo, se presenta un algoritmo de segmentación basado en súper pixeles y se compara su desempeño en la segmentación de *Trypanosoma cruzi*, contra tres métodos de segmentación empleados en el estado del arte.

Índice

Lista de figuras

Lista de Tablas

1. Introducción

Muchos microorganismos son causa de diversas enfermedades que afectan al ser humano. La gente se encuentra expuesta al contagio de algunos de estos padecimientos, debido a la coexistencia entre las personas y estos agentes infecciosos. Los métodos de contagio son variados, pero muchos de estos agentes hacen uso de otros organismos biológicos como mecanismo de transmisión. La Enfermedad de Chagas es un ejemplo de este tipo de padecimientos.

Trypanosoma cruzi es un microorganismo parásito de la sangre responsable de la Enfermedad de Chagas. El mecanismo natural de transmisión de este parásito son los insectos hematófagos de la subfamilia *Triatominae*, quienes habitualmente se alimentan de la sangre de animales mamíferos incluyendo al ser humano. La enfermedad puede ser mortal en etapas tardías en las cuales el parásito se reproduce en tejido muscular, debido a los daños que comúnmente ocasiona al músculo cardiaco. En la etapa temprana o aguda se desarrolla en el torrente sanguíneo del portador, lo que permite su localización mediante el análisis de sangre al microscopio [6].

Actualmente, existen diversos estudios enfocados a mejorar las herramientas de diagnóstico de enfermedades. De esta forma existen trabajos enfocados al tratamiento de imágenes médicas obtenidas de diversos estudios, como encefalogramas, ultrasonidos y rayos

X. También, existen trabajos enfocados a la detección y reconocimiento de componentes sanguíneos, conteo de estos elementos y la detección de microorganismo en la sangre. En este último tema, podemos mencionar al parásito *Plasmodium* [21], agente biológico causante de la Malaria. La detección de este microorganismo en la sangre constituye la evidencia certera en el diagnóstico de la enfermedad. Lo mismo ocurre con la Enfermedad de Chagas. Observar *Trypanosoma cruzi* en una muestra sanguínea implica el potencial desarrollo de la enfermedad en los seres humanos [6].

Este trabajo se enfoca en una parte del proceso de detección de *Trypanosoma cruzi*: la segmentación. Las imágenes de muestras sanguíneas capturan diversos componentes sanguíneos. En una imagen podemos observar el conjunto de células sanguíneas y, en caso de contagio, es posible observar a *Trypanosoma cruzi* en la misma. Para realizar el proceso de detección, nos interesa eliminar de la imagen los elementos irrelevantes, como las células sanguíneas, e idealmente, conservar los elementos que pudieran representar al parásito. Este proceso de separación de zonas de interés y zonas irrelevantes es conocido como segmentación de imágenes [11].

En este trabajo, se propone el uso de clasificadores para discriminar los elementos de la imagen. Por lo tanto, las áreas del conocimiento que intervienen en esta labor son el procesamiento de imágenes y el aprendizaje automático. El procesamiento de imágenes se encarga de operar sobre los pixeles de una imagen digital, aplicando diversas operaciones para mejorar la calidad de la misma, de forma que permita una mejor apreciación visual por parte de un humano o facilite procesamientos futuros por parte de un algoritmo computacional.

Las imágenes proporcionan gran cantidad de información al ser humano. El sentido de la vista permite reconocer gran cantidad de objetos, así como realizar tareas complejas. Existen varios trabajos enfocados a replicar la capacidad visual humana, haciendo uso de

computadoras digitales. Estos trabajos se centran en tareas específicas como el reconocimiento de caracteres o la identificación de un objeto en específico dentro de la imagen. Los trabajos de detección de parásitos mencionados anteriormente, son otro ejemplo de estos temas. Como se describirá en capítulos posteriores, un proceso para la identificación de estos elementos en la imagen está constituido por la segmentación, seguida de la extracción de características y finaliza con la identificación o clasificación.

Los dos últimos pasos del proceso de identificación, entran dentro del objeto de estudio del aprendizaje automático, quien a su vez, forma parte del campo de la inteligencia artificial. El aprendizaje automático, se encarga del diseño y estudio de métodos de aprendizaje computacional, incluyendo los problemas de clasificación de la información [33]. El objetivo de la clasificación, es asignar a cada bloque de información disponible, una etiqueta o clase, dentro de un conjunto de dos o más clases. Tomemos como ejemplo la identificación de números en un documento. Los bloques de información podrían estar conformados por cada símbolo escrito dentro del documento. El clasificador tendría que escoger entre una de dos clases disponibles: dígito o letra. Cuando el problema solo dispone de dos clases posibles, se conoce como un problema de clasificación binaria [45].

Los clasificadores deben ser capaces de interpretar los bloques de información, de forma que la computadora pueda tomar una decisión respecto a estos. Lo anterior hace necesario disponer de una representación numérica de la información, para que la computadora pueda operar sobre la misma. Tomando de nuevo el ejemplo de reconocimiento de dígitos, supongamos que tenemos una imagen digital del documento. Entonces, la imagen digital contiene información sobre los símbolos escritos y es posible emplear los valores numéricos de los pixeles de la imagen para representar está información. Entonces, el clasificador tomaría estos valores para realizar su proceso de decisión. Otra opción es tomar valores más complejos, derivados de los pixeles de la imagen, como la media y varianza de los pixeles, valores de textura, bordes o gradientes. Estos elementos pueden considerarse como

valores representativos de la información contenida en la imagen. Posteriormente, pueden ser ordenados en un vector numérico y alimentar al clasificador. Este proceso de cálculo de valores representativos es conocido como extracción de características [14].

La propuesta presentada en este trabajo, contempla el uso de clasificadores binarios y presenta los métodos empleados para la extracción de características de la imagen, así como las características empleadas. Una forma de abordar el problema de clasificación es mediante la clasificación de los pixeles individuales de la imagen. Sin embargo, en este trabajo se propone emplear grupos de pixeles y realizar la extracción de características y la clasificación sobre estos grupos. El problema de segmentación de imágenes se plantea como un problema de clasificación binaria, donde cada grupo de pixeles es catalogado como de interés o irrelevante. Así mismo, se realizaron pruebas con tres clasificadores: máquinas de vectores de apoyo, redes neuronales y AdaBoost.

Entre los principales resultados de este trabajo se encuentran un conjunto de imágenes de *Trypanosoma cruzi* segmentadas de manera manual. De igual forma, se presenta un algoritmo de segmentación basado en súper pixeles y clasificadores, el cual es comparado con tres algoritmos del estado del arte, empleados para la segmentación de parásitos, particularmente, *Plasmodium*. Se compara el desempeño en la segmentación de *Trypanosoma cruzi* del algoritmo propuesto contra los algoritmos de segmentación Bayesiana, Gaussiana y con una propuesta para la segmentación de *Trypanosoma cruzi* realizada durante el trabajo de tesis de licenciatura presentado en [34].

El documento se organiza de la siguiente manera: el capítulo 2 presenta los conocimientos previos sobre la enfermedad de Chagas y sobre los métodos de aprendizaje automático y procesamiento de imágenes empleados en este trabajo, así como algunos trabajos realizados en el área de detección de parásitos en imágenes de muestras sanguíneas. El capítulo 3 describe el problema abordado en este trabajo. El proceso de segmentación se describe

a detalle en el capítulo 4. Finalmente, los capítulos 5 y 6 presentan los experimentos y resultados obtenidos y exponen los trabajos futuros que pueden derivarse de esta investigación.

2. Marco Teórico

2.1. Antecedentes

Existen diferentes padecimientos que afectan la salud humana y sus causas son diversas. Algunas enfermedades son causadas por agentes biológicos que se introducen al cuerpo humano con la ayuda de un tercer organismo. Un ejemplo de lo anterior son las enfermedades del dengue y la malaria , las cuales son transmitidas mediante algunas especies de mosquito. Este es el caso de la tripanosomiasis americana, una enfermedad que es transmitida por algunos insectos conocidos como chinche o pic. En este capítulo se realiza una descripción de esta enfermedad y se presentan los conocimiento previos que permitirán describir el trabajo realizado.

2.1.1. Tripanosomiasis Americana

La tripanosomiasis americana es una enfermedad crónica endémica de Latinoamérica, teniendo presencia en México. La causa de este padecimiento es debida al protozoario parásito *Trypanosoma cruzi* 2.1. Este padecimiento, al igual que el microorganismo responsable del mismo, fueron descubiertos por el médico brasileño Carlos Chagas [7] en el

periodo comprendido entre 1909 y 1910. Por este motivo, este padecimiento es también conocido como enfermedad de Chagas.

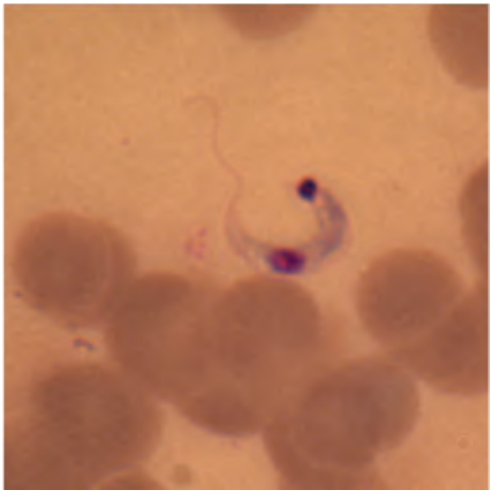

Figura 2.1: *Trypanosoma cruzi* en una muestra de sangre.

Trypanosoma cruzi pertenece a la familia *Trypanosomatidae* y es un protozoario hemoflagelado que habita en la sangre de algunos mamíferos. Posee un único flagelo que emplea para su movilidad al igual que un cinetoplasto y un núcleo vesiculoso situado en la parte central del parásito. El cineoplasto es una estructura que contiene tres espirales de ADN y se encuentra alojado dentro de una expansión capsular de la mitocondria [5].

Este padecimiento puede ser transmitido al ser humano de diversas maneras y la más común de ellas es mediante los mecanismos naturales de contagio. El vector natural de transmisión lo conforman los insectos hematófagos, particularmente, *Triatoma infestans* y *Triatoma dimidiata* el cual es conocido en el estado de Yucatán como chinche o pic (Fig. 2.2). Cuando estos insectos se alimentan de la sangre de una persona o animal infectado, los parásitos pasan al tubo digestivo del insecto donde se reproducen. Cuando *Triatoma infestans* se alimenta, en ocasiones también defecan. Si el insecto es portador de

Trypanosoma cruzi, estos parásitos estarán presentes en las heces del insecto. El ingreso del parásito al torrente sanguíneo ocurre al momento de frotar la herida causada por la picadura. A este método de transmisión realizado mediante un agente biológico externo, es conocido como transmisión por vector [6]. Otros medios de transmisión no vectoriales incluyen la transfusión de sangre y transplante de órganos donados por una persona infectada; Una madre infectada puede contagiar a su bebé durante el embarazo o el parto; También el contagio puede darse por la ingesta de alimentos contaminados [6].

Figura 2.2: *Triatoma dimidiata*, insecto responsable de la transmisión de la enfermedad de Chagas. Autor: Felipe Guhl

Una vez que los parásitos han ingresado al torrente sanguíneo, se pueden distinguir tres fases en la evolución de la enfermedad. La primera fase es conocida como fase o etapa aguda e inicia desde el momento que los parásitos ingresan al torrente sanguíneo. La duración de esta etapa es de 15 a 90 días y aunque en general es asintomática, un leve porcentaje de la población puede presentar decaimiento, fiebre, dolor de cabeza, escalofríos, pérdida de apetito e inflamación en los ganglios. Uno de los signos más notables de esta etapa es la

la inflamación de al área lesionada por el insecto al momento de alimentarse. Esta lesión, conocida como chagoma, desaparece durante el transcurso de la fase aguda [6]

Una vez concluida la fase aguda da inicio la fase indeterminada o fase de latencia caracterizada por la ausencia de síntomas. La duración de esta fase puede ser de varios años e incluso toda la vida del portador [6].

La fase final de esta enfermedad se denomina fase crónica y es alcanzada por 3 o 2 de cada 10 personas infectadas. Durante esta etapa el parásito pasa a reproducirse en el tejido muscular y es raro encontrarlo en la sangre. La fase crónica generalmente inicia entre 20 y 30 años después del ingreso de los parásitos al cuerpo. Las afectaciones más importantes de esta etapa son la cardiopatía chagásica y la afectaciones gastrointestinales. Una vez que los parásitos alcancen el músculo cardiaco la infección crónica de este último conduce al fallo cardiaco y muerte del portador [6].

Los métodos empleados para el diagnóstico de este padecimiento dependen de la fase en la que se encuentre. El diagnóstico certero consiste en demostrar la presencia del parásito. La observación de *Trypanosoma cruzi* durante el análisis sanguíneo en el microscopio presenta una prueba irrefutable de contagio. Sin embargo, este análisis solo puede ser realizado durante la fase aguda. Durante la fase crónica, el diagnóstico se puede realizar mediante la detección de anticuerpos anti T. cruzi [6]. La enfermedad es tratable con medicamentos durante la etapa aguda presentando efectividad en un 80 % de los casos [44]. No existe un tratamiento para la etapa crónica por lo que el control médico periódico es de vital importancia para evitar afectaciones mayores a causa de esta enfermedad.

2.2. Procesamiento de Imágenes Digitales

La segmentación requiere de la manipulación de la información contenida en una imagen de forma que sea posible reconocer patrones que permitan diferenciar los elementos de interés del resto de la imagen. La principal fuente de información son los valores de los pixeles de la imagen en los diferentes espacios de color disponibles. Para extraer información útil y segmentar la imagen, se combinan dos áreas del conocimiento relacionadas con la inteligencia artificial: el procesamiento de imágenes y el aprendizaje automático.

La frontera que divide a la visión computacional del procesamiento de imágenes en ocasiones no es muy clara. La visión computacional busca realizar decisiones útiles acerca de los objetos físicos y escenas reales contenidas en las imágenes digitales mediante el análisis de las mismas [32]. Por otra parte, el procesamiento de imágenes busca modificar la imagen con la finalidad de facilitar la búsqueda de información o mejorar la calidad de la misma. La segmentación presenta una forma de facilitar la búsqueda de información debido a que reduce la cantidad de pixeles dentro de la imagen. Aunque para esto es necesario que el proceso de segmentación tome decisiones acerca de la importancia de los pixeles. De acuerdo con [13], el procesamiento de imágenes digitales se refiere a los procesos que reciben y devuelven una imagen incluyendo a los procesos que extraen atributos de las mismas y reconocen objetos individuales dentro de ellas.

2.2.1. Representación de una Imagen

Para procesar la imagen es necesario tenerla almacenada en la memoria de la computadora mediante una representación discreta de la misma. Las imágenes digitales en escala de grises pueden ser representadas mediante arreglos bidimensionales discretos. Los ob-

jetos del mundo real son discretizados en una matriz de m filas por n columnas como resultado del proceso de muestreo realizado por los sensores de adquisición, obteniendo la siguiente representación matemática de la imagen:

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,n-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,0} & a_{m,1} & \cdots & a_{m-1,n-1} \end{bmatrix} \tag{2.1}$$

De esta forma, la matriz A representa una imagen de dimensiones $m \times n$ donde cada entrada $a_{i,j}$ es considerada como un pixel de la imagen. En ocasiones, resulta de utilidad considerar una imagen como una función de dos variables $I(x, y)$, donde x hace referencia a las columna y y a las filas de la imagen. La notación $I(x_i, y_i)$ hace referencia al valor del pixel en la posición (x_i, y_i) de la imagen. Como ejemplo de lo anterior, se tiene que la notación $I(0, 1)$ hace referencia al elemento $a_{1,0}$ dentro de la matriz A presentada en (2.1). El dominio de la función $I(x, y)$ está dado por las dimensiones de la imagen, tomado valores enteros de modo que $0 \leq x < n$ y $0 \leq y < m$. El origen de la imagen, por lo general, se establece en la esquina superior izquierda de la misma [13]. El valor de los ejes crece conforme a las columnas y filas de A. Esto se ilustra en la Fig. 2.3.

Cada uno de los pixeles de la imagen son representados como enteros positivos que, por lo general, toman valores entre cero y 255 para una imagen en escala de grises. El valor de cero representa el tono de gris negro. La intensidad o brillo del pixel aumenta conforme el valor se incrementa hasta ser completamente blanco cuando el pixel tiene por valor 255. La Fig 2.4 presenta una imagen en escala de grises junto con su representación matricial, donde es posible apreciar el cambio de valor de los pixeles respecto a su intensidad de brillo.

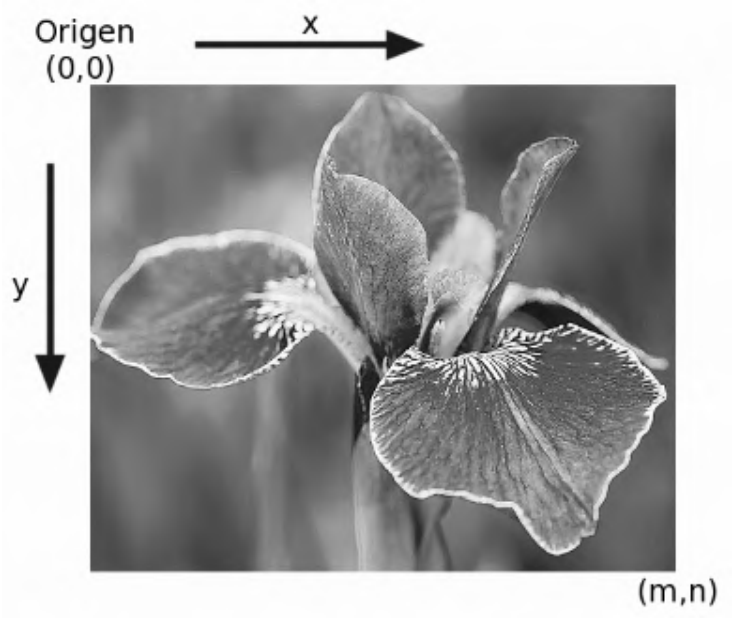

Figura 2.3: El origen de una imagen está situado en la esquina superior izquierda de la misma. Las flechas indican la dirección positiva de los ejes x, y.

Los pixeles de una imagen a color son representados de una forma diferente, por ejemplo, los pixeles de una imagen a color RGB [38] son representados por tres elementos. Las siglas RGB se refieren a que el color de un pixel se compone por la combinación de las intensidades de rojo (R), verde (G) y azul (B) del pixel. Un pixel $\boldsymbol{p}$ ubicado en la posición (x, y) de la imagen, puede verse como un vector de tres elementos:

$$\boldsymbol{p_{RGB}} = \begin{bmatrix} r \\ g \\ b \end{bmatrix} \tag{2.2}$$

Las entradas del vector anterior son conocidas como canales y pueden tomar valores desde 0 hasta 255. Un valor cero indica la completa ausencia del canal correspondiente y un valor 255 indica su intensidad máxima. Si todos los canales son cero obtendremos

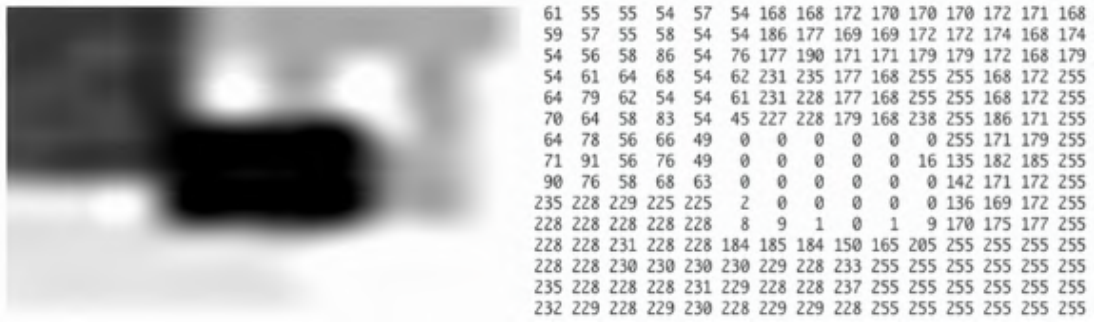

Figura 2.4: Los pixeles de una imagen en escala de grises son representados mediante valores enteros en el rango de 0 a 255. Donde 0 representa el nivel de gris negro y 255 el blanco.

el color negro. De forma similar, si los tres canales son todos 255 obtendremos el color blanco.

Existen diferentes relaciones entre los pixeles de una imagen respecto a la posición que ocupan dentro de la misma. Dada una imagen $I(x, y)$, un pixel $\boldsymbol{p}$ en la posición (x, y) posee un vecindario formado por los pixeles que lo rodean. En este sentido es posible distinguir diferentes vecindarios. Un vecindario 4 está formado por los dos pixeles horizontales y los dos pixeles verticales que rodean a $\boldsymbol{p}$, es decir, un vecindario 4, denotado como $N_4(\boldsymbol{p})$, está conformado por los pixeles en las siguientes posiciones respecto a (x, y):

$$(x-1, y), (x+1, y), (x, y-1), (x, y+1) \tag{2.3}$$

De forma similar, el vecindario $N_D(\boldsymbol{p})$ está formado por los cuatro pixeles diagonales que rodean a $\boldsymbol{p}$ en la posición (x, y), es decir:

$$(x-1, y-1), (x+1, y-1), (x-1, y+1), (x+1, y+1) \tag{2.4}$$

La unión de los vecindarios anteriores forman un tercer grupo conocido como vecindario 8 y simbolizado como $N_8(\boldsymbol{p})$. Este vecindario está formado por los 8 pixeles inmediatos que

rodean a $\boldsymbol{p}$. Cabe mencionar que si $\boldsymbol{p}$ se encuentra en el borde de la imagen es posible que algunos elementos de sus vecindarios queden fuera del dominio de la imagen [13].

2.2.2. El Histograma de una Imagen

El histograma de una imagen es un elemento de utilidad para analizar la distribución de las intensidades de los pixeles. El histograma de una imagen en escala de grises está conformado por el conteo de las incidencias de cada uno de los valores de los pixeles presentes en la imagen, es decir, un histograma representa la frecuencia con la que se repite cada pixel dentro de la imagen. El conteo puede considerar todos los posibles valores que puede tomar un pixel o establecer un conjunto de intervalos y contabilizar los pixeles que pertenecen a cada intervalo. La Fig 2.5 muestra una imagen en escala de grises y su histograma.

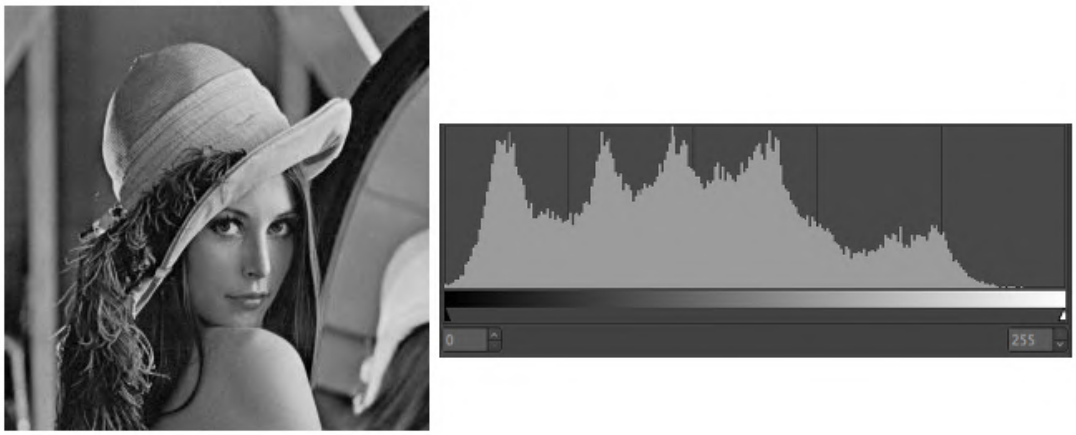

Figura 2.5: El histograma de una imagen obtenido con el programa de procesamiento de imágenes de GNU, GIMP.

El histograma presenta información de utilidad que puede ser empleada para la identificación de elementos en la imagen. Al comparar dos histogramas es posible deducir si

representan al mismo elemento o imagen [4][11]. Para realizar esta tarea se pueden emplear métricas de similitud entre los histogramas o entrenar un algoritmo de aprendizaje automático que permita discriminar los histogramas de diferentes elementos.

2.2.3. Filtros y Derivadas de la Imagen

Una imagen puede ser filtrada de dos maneras: en el dominio del espacio y en el dominio de la frecuencia. En este trabajo, cada vez que se mencione el tema de filtrado se hará referencia al filtrado en el dominio del espacio de la imagen, que opera directamente sobre los pixeles de la imagen [13].

El filtrado de imágenes es una operación que modifica el valor de un pixel $\boldsymbol{p}$ con base a una máscara proporcionada. Una máscara de filtrado, también conocida como kernel, es un arreglo bidimensional con valores reales conocidos como pesos. La Fig. 2.6 presenta algunos ejemplos de máscaras de convolución [32].

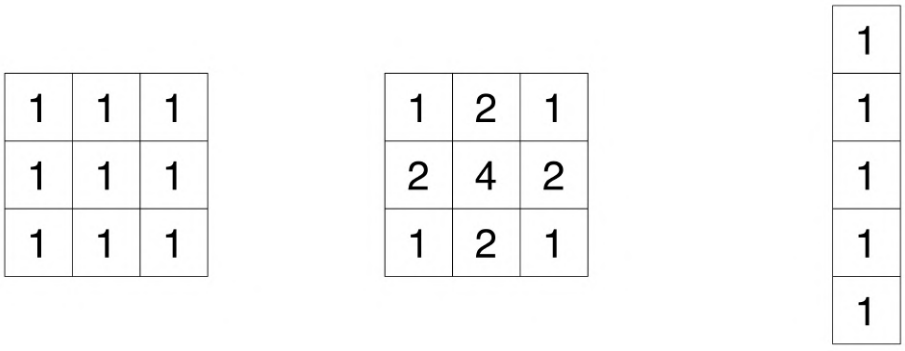

Figura 2.6: Ejemplos de máscaras que pueden ser aplicadas a una imagen. De izquierda a derecha las primeras dos máscaras son cuadradas y la tercera es rectangular.

Las máscaras tienen un origen que se encuentra en el centro de la misma para el caso de máscaras simétricas (máscaras de dimensiones impares). Por ejemplo, una máscara de

3×3 tendrá su origen en la posición $(1, 1)$ del arreglo (considerando que los índices inician en cero). En el caso de máscaras no simétricas, el origen puede ser situado en cualquier punto de la misma.

Para aplicar la máscara a la imagen, cada valor de esta será multiplicado por los elementos que componen un vecindario de un determinado pixel. El proceso de filtrado consiste en aplicar la operación anterior a cada pixel de la imagen. A esta operación de multiplicar los elementos de la máscara por los pixeles de la imagen se le conoce como convolución y se define de la siguiente manera, considerando una máscara simétrica A de dimensiones $m \times n$ y una imagen I de $M \times N$:

$$g(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} A(s, t) I(x + s, y + t) \tag{2.5}$$

donde A es la máscara de convolución, I es la imagen a filtrar, $a = (n-1)/2$ y $b = (m-1)/2$. Este proceso se repite para $y = 1, 2, 3, ...M - 1$ y $x = 1, 2, 3, ...N - 1$, de modo que se opere sobre cada pixel de la imagen.

La operación de filtrado de imágenes puede ser empleada para diversas aplicaciones, entre ellas la minimización de ruido dentro de la imagen y la aproximación de derivadas de la misma. La derivada de una imagen puede ser aproximada mediante las siguientes máscaras:

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.6}$$

La máscara anterior aproxima la derivada en la dirección x. La derivada en la dirección y puede ser aproximada mediante:

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{2.7}$$

La derivada puede ser usada para varias aplicaciones dentro del procesamiento de imágenes. La más común es para la detección de bordes en la imagen. Varios algoritmos de detección de bordes emplean diferentes máscaras para aproximar la derivada. Otras aplicaciones incluyen la detección de elementos descriptivos de la imagen. Estos elementos son conocidos como características y pueden ser empleados en labores de reconocimiento de patrones en imágenes. Este tema será retomado más adelante en este capítulo.

2.2.4. Segmentación de Imágenes

La segmentación de imágenes consiste en dividir la imagen en diferentes regiones mediante la creación de grupos de pixeles. Este problema trata de recolectar aquellos datos presentes en la imagen que adquieren sentido al agruparse, con base en un modelo previamente definido [11].

Una forma de segmentar una imagen es comparando el valor de cada pixel con un umbral predefinido. Este proceso es conocido como umbralización. Suponiendo que los pixeles de una imagen en escala de grises pueden agruparse en dos regiones con base al valor de sus intensidades, se define un umbral T que separe ambas regiones. De esta forma, un pixel p ubicado en la posición (x, y) dentro de una imagen I será considerado dentro de un grupo u otro de acuerdo a si $I(x, y) > T$. Es común emplear la segmentación para separar los elementos interés dentro de una imagen. Es común referirse a los elementos que no son de interés como el fondo de la imagen. Con base a lo anterior, un pixel que sobrepase el umbral puede ser considerado como parte de la región de interés mientras que en caso contrario, puede ser considerado como parte del fondo. En general, el proceso de umbralización puede definirse de la siguiente forma:

$$g(x, y) = \begin{cases} 1 & \text{si } I(x, y) > T \\ 0 & \text{si } I(x, y) \leq T \end{cases} \tag{2.8}$$

donde $g(x.y)$ es la imagen umbralizada resultante, $I(x, y)$ es la imagen de entrada y T es el umbral a considerar. Los pixeles de la imagen resultante g tendrán dominio en $0, 1$, donde, generalmente, el valor 1 se reserva para los elementos de interés [11]. Una imagen donde el dominio de los pixeles se compone de únicamente dos valores es conocida como imagen binaria. La Fig. 2.7 muestra el resultado de aplicar el proceso de umbralización a los pixeles de una imagen en escala de grises. Cabe mencionar que esta es una forma básica de segmentar una imagen. Es posible emplear formas más complejas en donde el umbral es comparado con una función de los pixeles de la imagen.

Figura 2.7: Resultado de aplicar el proceso de umbralización a una imagen con un umbral de 120. Los objetos por debajo de este umbral son degradados a cero mientras que los que están por arriba son establecidos en uno.

La segmentación no es exclusiva de las imágenes en escala de grises. La segmentación de imágenes a color se emplea para agrupar los pixeles que poseen algún atributo en particular relacionados con el color de los mismos. La segmentación de un color se puede realizar comparando un pixel determinado con una representación del color deseado. Un ejemplo de esta representación es la media, obtenida a partir de un conjunto muestra de pixeles del color a segmentar. La comparación puede ser realizada mediante alguna

métrica de similitud o distancia, e.g. la distancia euclidiana. Considerando dos pixeles $\boldsymbol{p}$ y $\boldsymbol{q}$ en el espacio de color RGB, la distancia euclidiana entre dos elementos está dada por la siguiente ecuación:

$$D(\boldsymbol{p}, \boldsymbol{q}) = [(r_p - r_q)^2 + (g_p - g_q)^2 + (b_p - b_q)^2]^{1/2} \tag{2.9}$$

donde r_p, g_p y b_q representan los canales rojo, verde y azul del pixel $\boldsymbol{p}$. De forma similar r_q, g_q y b_q representan los canales rojo, verde y azul del pixel $\boldsymbol{q}$ [13].

La segmentación de un color particular en la imagen se puede realizar mediante un procedimiento similar a la umbralización descrita anteriormente. La distancia entre un pixel determinado y la media (o cualquier otro modelo) del color deseado puede ser comparada con un umbral T, de forma que, si el umbral no es excedido, se considerará que el pixel es similar al color deseado. Los pixeles que sobrepasen este umbral serían considerados como parte del fondo (Fig. 2.8). Lo anterior es resumido por la siguiente igualdad:

$$g(x, y) = \begin{cases} 1 & \text{si } D(\boldsymbol{p}, \boldsymbol{q}) < T \\ 0 & \text{si } D(\boldsymbol{p}, \boldsymbol{q}) \geq T \end{cases} \tag{2.10}$$

donde $\boldsymbol{p}$ es un pixel RGB ubicado en la posición (x, y) de una imagen I y $\boldsymbol{q}$ es el modelo representativo del color de interés.

El problema de umbralización puede interpretarse como un problema de clasificación en donde los pixeles pueden ser asignados a la clase fondo o a la clase objeto, según el umbral de discriminación. En el caso de las imágenes a color, existen problemas similares en la que la imagen a color puede ser particionada en dos clases, donde cada pixel pertenece sólo a una de estas clases. Este es un problema de clasificación binaria, el cual es discutido en el apartado 2.3.1 de este capítulo.

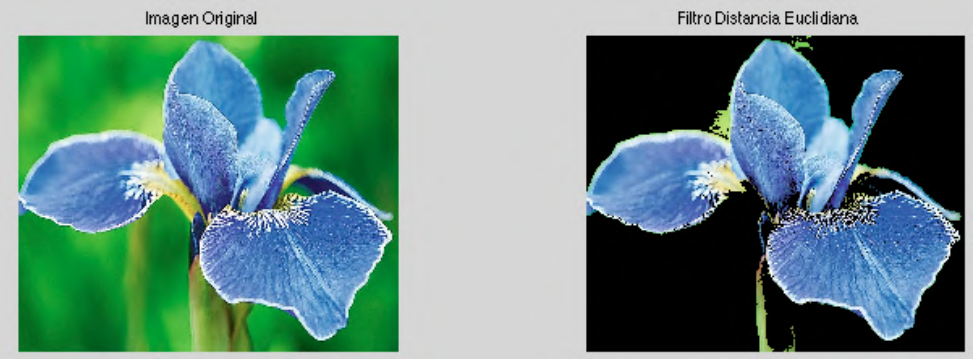

Figura 2.8: Segmentación del color azul usando la distancia euclidiana como métrica de distancia y un umbral de 0.5 para la comparación. Los pixeles cuya distancia excede el umbral fueron degradados al vector (0,0,0), los que cumplían la condición conservaron su valor.

2.2.5. La Imagen Integral

Algunos métodos para la extracción de información en imágenes requieren el cálculo de sumatorias sobre los valores de los pixeles contenidos en diferentes regiones rectangulares. Realizar la sumatoria de regiones puede ser una tarea tardada e ineficiente si existe traslape entre las regiones. Viola y Jones [40] [41] proponen una representación intermedia de la imagen que permite calcular estas sumatorias de una forma rápida y eficiente. Esta representación es conocida como la imagen integral.

La imagen integral en el punto (x, y) contiene la sumatoria de todos los pixeles contenidos por la región rectangular formada por el origen de la imagen y el punto en cuestión, como se ilustra en la Fig. 2.9. Formalmente, dada una imagen I en escala de grises, la imagen integral se define como:

$$II(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y') \tag{2.11}$$

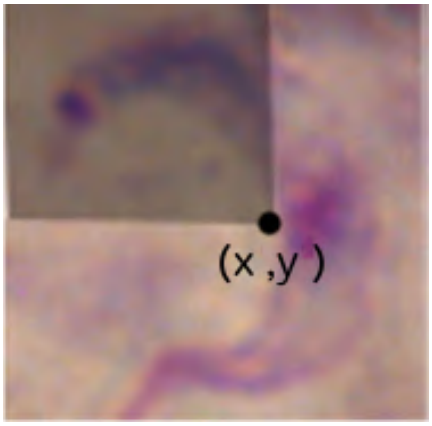

Figura 2.9: La imagen integral en la posición (x, y) es la sumatoria de todos pixeles dentro de la región sombreada.

donde $II(x, y)$ es el valor de la imagen integral en la posición (x, y) e $I(x', y')$ es el valor del pixel en la posición (x', y') de la imagen original. Esta representación puede ser calculada de forma eficiente empleando las siguientes dos fórmulas:

$$\begin{aligned} S(x, y) &= S(x, y-1) + I(x, y) \\ II(x, y) &= II(x-1, y) + S(x, y) \end{aligned} \tag{2.12}$$

donde $S(x, y)$ acumula la sumatoria de todos los pixeles de las filas menores o iguales a y y que están en la columna x, es decir, $S(x, y) = I(x, 0) + I(x, 1) + I(x, 2) + \ldots, +I(x, y)$. Se considera que $II(-1, y) = S(x, -1) = 0$. Lo anterior permite calcular la imagen integral realizando solo un recorrido sobre los pixeles de la imagen original.

Una vez calculada la imagen integral, es posible calcular las sumatorias de los pixeles de cualquier región rectangular en tiempo constante, accediendo únicamente a cuatro elementos de la imagen integral. Estos elementos coinciden con las esquinas que delimitan la región rectangular (Fig. 2.10).

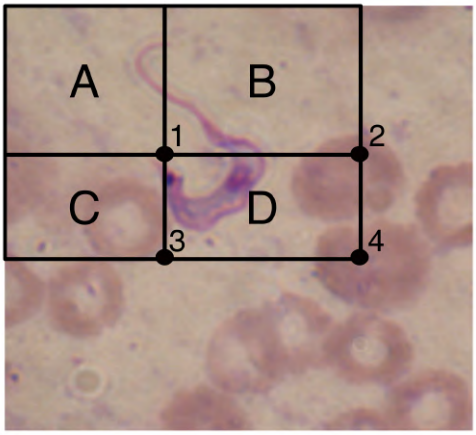

Figura 2.10: La sumatoria de los pixeles de una imagen, dentro de cualquier región rectangular puede ser calculada usando 4 referencias en la imagen integral.

Con base en la Fig. 2.10, la suma de los pixeles dentro de la región rectangular D está dada por la siguiente expresión:

$$4 + 1 - (2 + 3) \tag{2.13}$$

en donde el valor de la imagen integral en el punto 1 es igual a la suma de los pixeles dentro de la región A, el valor en el punto 2 está dado por $A + B$ y el valor en 3 y 4 está dado por $A + C$ y $A + C + B + D$ respectivamente.

2.2.6. Súper Pixeles SLIC

Los súper pixeles están formados por conjuntos de pixeles dentro de una imagen I. Estos conjuntos describen regiones continuas bien delimitadas que dividen la imagen. La principal motivación de emplear súper pixeles en la segmentación es la de extraer características descriptivas a partir de un conjunto de pixeles en lugar de emplear un pixel individual. Existen varios algoritmos orientados a la obtención de súper pixeles y en este trabajo se eligió el algoritmo SLIC (del inglés *simple linear iteractive clustering*) debido a

su fácil implementación y a la tendencia a preservar bordes en la imagen al momento de agrupar los pixeles.

Este algoritmo es presentado por R. Achanta en [2] y se basa en el algoritmo de agrupamiento k medias. Para generar los conjuntos de pixeles el algoritmo mide la similaridad respecto al color y la cercanía o proximidad entre dos pixeles de la imagen. Para medir la cercanía entre dos pixeles, cada uno se representa como un punto en un espacio penta-dimensional formado por la posición (x, y) del pixel en la imagen y los canales l, a, b de la imagen en formato CIELAB obteniendo el siguiente vector por cada pixel:

$$\boldsymbol{p}^5 = [l, a, b, x, y] \tag{2.14}$$

Una vez establecidos los elementos que definen un pixel en el espacio penta-dimensional, es necesario especificar una métrica de distancia que evalúe la similaridad de dos pixeles. Consideremos que el algoritmo toma como entrada el número K de súper pixeles deseados, donde cada súper pixel posee aproximadamente el mismo tamaño. Para una imagen de $m \times n$ con un total de $N = m * n$ pixeles, el tamaño aproximado de cada súper pixel es aproximadamente N/K pixeles. Con base a lo anterior, los centroides de cada súper pixel deben estar separados a intervalos de $S = \sqrt{N/K}$ en una cuadrícula. Tomando en cuenta lo anterior, para dos puntos $\boldsymbol{p}_k^5$, $\boldsymbol{p}_i^5$ en el espacio definido anteriormente, la métrica de distancia considera la similitud respecto al color mediante la siguiente expresión:

$$d_{lab}(\boldsymbol{p}_k^5, \boldsymbol{p}_i^5) = \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2} \tag{2.15}$$

Similarmente, la proximidad de los dos puntos es calculada mediante la siguiente igualdad:

$$d_{xy}(\boldsymbol{p}_k^5, \boldsymbol{p}_i^5) = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2} \tag{2.16}$$

Finalmente, la métrica de similitud empleada combina ambas distancias en la siguiente expresión:

$$D_s(\boldsymbol{p}_k^5, \boldsymbol{p}_i^5) = d_{lab}(\boldsymbol{p}_k^5, \boldsymbol{p}_i^5) + \frac{c}{S} d_{xy}(\boldsymbol{p}_k^5, \boldsymbol{p}_i^5) \tag{2.17}$$

donde c es un valor que controla que tan compacto es el súper pixel.

El algoritmo toma como entrada el número K de súper pixeles deseados, dentro del contexto de k medias, este valor equivale al número de grupos en los que se desea dividir la imagen. El algoritmo k medias inicializa un conjunto de k centroides aleatorios dentro de la imagen y posteriormente, para cada centroide, asigna los pixeles más cercanos realizando una búsqueda sobre toda la imagen. SLIC inicializa K centroides localizados a intervalos regulares y limita esta búsqueda a una región de $2S \times 2S$ alrededor del centroide, considerando que los pixeles de cada súper pixel estarán concentrados en un área de S^2 alrededor de su centroide.

El proceso de extracción de súper pixeles opera de la siguiente forma, primero, genera un conjunto de K centroides separados a una distancia regular de S en una cuadrícula. Posteriormente estos centroides son reubicados a la posición de menor gradiente dentro de una región de 3×3. El gradiente en un punto (x, y) de la imagen es calculado de la siguiente forma:

$$G(x, y) = \|I(x+1, y) - I(x-1, y)\|^2 + \|I(x, y+1) - I(x, y-1)\|^2 \tag{2.18}$$

donde $I(x, y)$ representa un pixel de la imagen en el espacio de color lab y $\|\cdot\|$ representa la norma L_2. El objetivo de esta etapa del algoritmo es evitar que los centroides queden definidos sobre un borde. Una vez inicializados los centroides, cada pixel es asignado al centroide más cercano con base a la métrica de distancia definida en (2.17). Una vez asignado cada pixel, los centroides son recalculados empleando el promedio de los pixeles asociados a este. Una vez que los centroides han sido actualizados, los pixeles son nuevamente asignados y este ciclo es repetido hasta alcanzar una condición de paro.

Una criterio de paro puede estar basado en el error de asignación, definido como la distancia L_1 entre los centroides previos y los recién calculados. Otra condición de paro que puede ser empleada es el número de ciclos de actualización de los centroides, a lo que

llamaremos iteraciones. Se puede terminar el proceso en el momento en que se alcance un número máximo de iteraciones indicado al inicio del algoritmo.

Algoritmo 1 El algoritmo SLIC

1: Inicializar los centros de los grupos $C_k = [l_k, a_k, b_k, x_k, y_x]$. $k = 1, 2, \ldots, K$.
2: Mover los centros a la posición del menor gradiente en un vecindario de 3×3.
3: Inicializar $etiqueta(\boldsymbol{p}) = -1$, $dist(\boldsymbol{p}) = \infty$ para cada pixel $\boldsymbol{p}$.
4: **repetir**
5: **para** cada centro C_k. **hacer**
6: **para** cada pixel $\boldsymbol{p}$ en una región de $2S \times 2S$ centrada en C_k. **hacer**
7: Calcular la distancia $d_{ck,\boldsymbol{p}^5}$ definida como la distancia D_s entre C_k y $\boldsymbol{p}$.
8: **si** $d_{ck,\boldsymbol{p}^5} < dist(\boldsymbol{p})$. **entonces**
9: Hacer $dist(\boldsymbol{p}) = d_{ck,\boldsymbol{p}^5}$ y $etiqueta(\boldsymbol{p}) = k$.
10: **fin si**
11: **fin para**
12: **fin para**
13: Actualizar los centros y calcular el error residual E.
14: **hasta que** $E \leq umbral$.
15: Asegurar la conectividad de cada súper pixel.

Los puntos principales de este proceso se presentan en el algoritmo 1. La Fig. 2.11 muestra el resultado de agrupar los pixeles de una imagen en 300 súper pixeles. En esta figura se han resaltado las fronteras entre cada súper pixel. Al final de este proceso, todos los pixeles dentro de cada súper pixel deben de estar conectados, sin embargo, es posible que esto no ocurra para algunos elementos. Por este motivo la etapa final de este algoritmo debe asegurar esta conectividad. Si bien el artículo original no presenta una forma de realizar esta etapa los detalles de la implementación se discutirán en el apartado 4.1 de este documento.

Figura 2.11: Izquierda: Imagen original. Derecha: La imagen dividida en 300 súper pixeles.

2.3. Aprendizaje Automático

El aprendizaje automático es una rama de la inteligencia artificial que estudia los problemas de aprendizaje computacional, es decir, se encarga del diseño y análisis de algoritmos que le permiten a una computadora aprender. Una computadora aprende si su desempeño en la solución de un problema o tarea mejora, a medida que su experiencia aumenta. Por ejemplo, en el problema de reconocimiento de caracteres, la experiencia puede ser adquirida mediante un conjunto de caracteres diseñado para el entrenamiento del algoritmo, mientras que el desempeño puede ser medido mediante el porcentaje de elementos reconocidos correctamente dentro de un segundo conjunto, diferente al conjunto de entrenamiento y comúnmente conocido como conjunto de prueba [22]. Entre las aplicaciones prácticas del aprendizaje automático se encuentra la traducción automática de documentos, la seguridad y control de acceso mediante el reconocimiento de patrones biométricos como el rostro o la huella digital, la detección de fraudes en tarjetas de crédito, entre otras [33].

Un factor que interviene en el proceso de aprendizaje es la retroalimentación. La retroalimentación le permite al programa elegir las acciones que mejoran su desempeño dentro de un conjunto de acciones disponibles. En la práctica, es posible identificar tres tipos de aprendizaje con base a la retroalimentación que reciben. El primero de estos es el aprendizaje supervisado, en cual realiza el aprendizaje de una función de discriminación con base a un conjunto de ejemplos de entrenamiento. Este conjunto está formado por ejemplos de entradas que la función analizará, junto con sus respectivas salidas, entendiendo estas últimas como el valor que se espera que la función devuelva al recibir dicha entrada. Luego se tiene el aprendizaje no supervisado, en el cual, los algoritmos realizan el aprendizaje a partir de ejemplos de entrenamiento de los cuales se desconoce la salida esperada. Finalmente, se tienen los métodos de aprendizaje por refuerzo quienes emplean estímulos para realizar el aprendizaje. Estos métodos emplean indicadores sobre que comportamiento o salida es aceptable para determinada situación o entrada [31].

Las entradas de estos métodos están formados por representaciones numéricas de los elementos que interfieren en el problema. Por ejemplo, supongamos un sistema de diagnóstico de determinada enfermedad. El sistema puede recibir como entrada un conjunto de síntomas y presentar un diagnóstico como salida, supongamos saludable o enfermo. Los datos de entrada pueden incluir la temperatura corporal, presión arterial o si el paciente presenta dolor de cabeza, entre otros parámetros. Toda la información anterior puede ser representada numéricamente y puede agruparse en forma vectorial, obteniendo vectores como $[34.5, 120, 90, 1]$, donde cada entrada representa los valores registrados de temperatura, presión (sistólica y diastólica) y un valor numérico que indique la presencia o ausencia de dolor, que en nuestro ejemplo el valor uno indica presencia y cero ausencia. Las entradas del vector anterior son conocidas como características o atributos y conforman los datos de entrada para los algoritmos de aprendizaje. Cuando la información proviene de imágenes es posible en emplear directamente el valor de los pixeles como características. Otra

opción, es el cálculo de nuevos atributos con base en los valores originales de los pixeles. El proceso de calcular nuevas características que describan la información contenida en el conjunto de datos original es conocido como extracción de características [14].

Durante este trabajo, se emplearon algoritmos de aprendizaje automático para la clasificación de características extraídas a partir de súper pixeles, con el objetivo de segmentar la imagen. Los métodos empleados en este trabajo pertenecen al área del aprendizaje supervisado y serán descritos en breve.

2.3.1. Clasificación Binaria

La clasificación es un problema que trata de la asignación correcta de un conjunto de clases o etiquetas, a un conjunto de datos. Un ejemplo de esto es la segmentación de imágenes. Suponiendo que se desea segmentar una imagen en dos regiones: fondo u objeto de interés. Este problema puede interpretarse en términos de la clasificación de los pixeles en una de las dos clases disponibles.

El problema de segmentación de pixeles en fondo/objeto de interés es un ejemplo de un problema particular de clasificación, conocido como clasificación binaria. De forma general, el problema de clasificación es binario cuando el conjunto de clases posibles posee solo dos elementos (Fig. 2.12).

En este problema intervienen varios elementos: un conjunto de características o instancias $\boldsymbol{\chi}$ pertenecientes a un espacio de instancias χ y un conjunto de etiquetas o clases $\gamma \in \{-1, +1\}$, en donde cada instancia está relacionada con una etiqueta, lo cual es denotado $(\boldsymbol{\chi}, \gamma)$, $\boldsymbol{\chi} \in \chi$, $\gamma \in \{-1, +1\}$. El clasificador recibe una instancia y trata de asignar una etiqueta, la cual representa la clase a la que pertenece dicha instancia. La clasificación es correcta si la etiqueta que devuelve coincide con la clase a la que la instancia

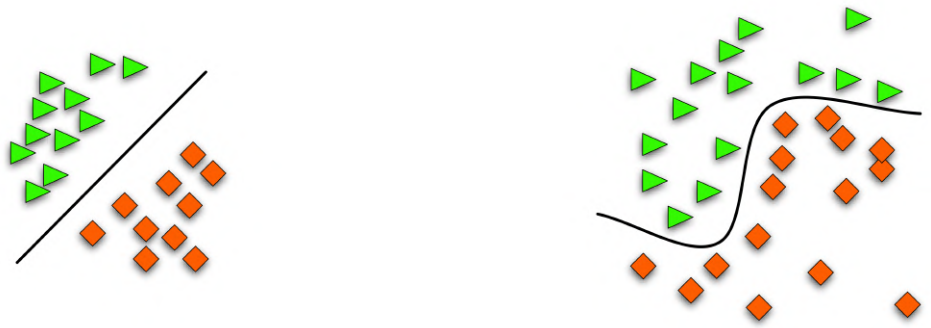

Figura 2.12: Ejemplos de clasificación binaria. En ambas imágenes se tienen dos grupos disponibles. El objetivo es encontrar una función que adecuadamente identifique los elementos de los grupos disponibles.

pertenece. En el entrenamiento supervisado de un clasificador intervienen conjuntos de entrenamiento. Un conjunto de entrenamiento τ de m instancias para un clasificador binario es representado como $\tau = \{(\boldsymbol{\chi}_i, \gamma_i)\}$, $i \in \{1, 2, 3, \ldots, m\}$, donde cada instancia $\boldsymbol{\chi}_i$ corresponde a la clase γ_i [33] [45].

2.3.2. Redes Neuronales y el Algoritmo de Retropropagación

Las redes neuronales artificiales tienen sus bases en la imitación de los sistemas biológicos formados por redes de neuronas interconectadas entre ellas. Una neurona artificial es modelada como una función de los datos de entrada y que entrega una salida que puede ser empleada como entrada para una nueva neurona o como una de las salidas finales de la red. Debido a lo anterior, los datos de entrada de determinada neurona pueden provenir del conjunto de datos que definen el problema o de la salida de otras neuronas de la red. Una red neuronal puede ser representada, de manera gráfica, mediante un grafo donde cada nodo representa una neurona y las aristas representan las conexiones entre las neuronas

(Fig. 2.13). Para este trabajo consideraremos redes neuronales representadas por grafos sin ciclos.

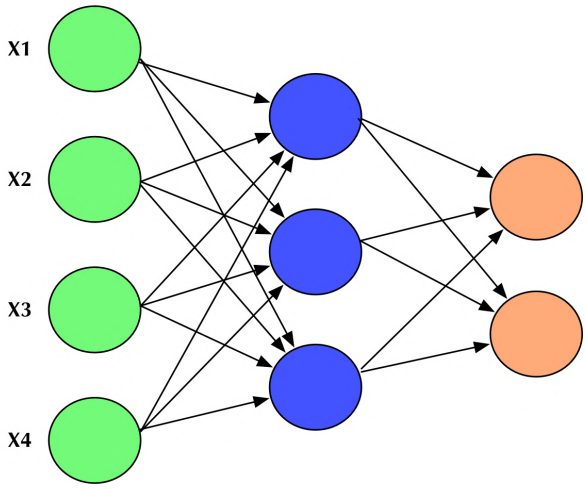

Figura 2.13: Representación gráfica de una red neuronal. Las neuronas en color verde son neuronas de entrada, las de color azul son neuronas ocultas y las de color naranja son neuronas de salida. La red posee una sola capa oculta.

Es posible distinguir tres tipos de neuronas en la red: las neuronas de entrada, quienes reciben la entrada directamente del conjunto de datos. Las neuronas ocultas que reciben su entrada de otras neuronas y sus salidas alimentan a otras neuronas. Finalmente, tenemos a las neuronas de salida que proporcionan la salida final de la red neuronal. Es posible organizar a las neuronas en capas, entiendo por capa, el conjunto de neuronas que recibe la misma información de entrada proveniente del mismo origen para cada neurona. En este sentido, es posible distinguir entre capa de entrada, capas ocultas y capa de salida. Lo anterior se presenta en la Fig. 2.13. El número de capas ocultas de una red neuronal puede variar, así como el número de neuronas por capa. Esto permite crear diferentes configuraciones en la estructura de la red.

Como se había mencionado al principio, cada neurona evalúa una función sobre su entrada. Un ejemplo de función es la siguiente:

$$o(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \text{si } w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n > 0 \\ -1 & \text{de otra forma} \end{cases} \tag{2.19}$$

donde cada w_i es un peso que la neurona asigna a cada entrada x_i y $-w_0$ es un umbral de decisión. La neurona definida mediante la función anterior es conocida como perceptrón. El aprendizaje de esta neurona consiste en encontrar el vector de pesos adecuado que permita clasificar correctamente las entradas.

Debido a que un perceptrón es definido mediante una función lineal de su entrada, un solo perceptrón es capaz de ajustarse a datos que son linealmente separables. Varias capas de neuronas lineales aún siguen generando funciones lineales [22]. Para extender las capacidades de un modelo de predicción basado en redes neuronales, es preciso considerar otro tipo de neurona. Otra función que se ha usado en el campo de las redes neuronales es la función sigmoidal. Una neurona con esta función es más adecuada que el perceptrón y es definida de la siguiente forma:

$$o(x_1, x_2, \ldots, x_n) = \frac{1}{1 + e^{-net}}, \text{ con } net = \sum_{i=0}^{n} w_i x_i \tag{2.20}$$

donde, para fines prácticos, $x_0 = 1$. De la ecuación anterior, para obtener la salida de esta neurona, primero se realiza la suma de las entradas ponderadas con los pesos de la neurona. Debido que w_0 no es multiplicado por ninguna entrada y simplemente es sumado, es común agregar el valor $x_0 = 1$ en los datos de entrada, para que multiplique a este peso sin alterar su valor. También, es prudente hacer notar que el número de pesos de una neurona es igual al número de entradas más uno. Una vez realizada esta suma, el resultado pasa a ser evaluado por la función sigmoidal. Nuevamente, el vector de pesos es empleado para definir a la neurona. Las neuronas de este tipo son empleadas para formar redes con una o más capas ocultas. Para que la red neuronal pueda clasificar correctamente

la información, es necesario emplear una metodología que encuentre los pesos adecuados para cada neurona de la red. En este trabajo se empleó el algoritmo de Retropropagación para entrenar la red.

Algoritmo 2 El algoritmo de Retropropagación

1: Inicializar los pesos de cada neurona empleando números pequeños generados de forma aleatoria (por ejemplo, entre -0.05 y 0.05).
2: Establecer una taza de aprendizaje η.
3: **repetir**
4: **para** Cada Ejemplo de entrenamiento. **hacer**
5: Propagar la entrada. Calcular la salida para cada neurona de la red iniciando por la capa de entrada.
6: Retropropagación del error:
7: **para** Cada neurona k en la capa de salida. **hacer**
8: Calcular el error: $\delta_k = o_k(1 - o_k)(\gamma_k - o_k))$.
9: **fin para**
10: **para** Cada capa oculta i. **hacer**
11: **para** Cada neurona h en la capa oculta i, iniciando con la última capa oculta. **hacer**
12: Calcular el error: $\delta_h = o_h(1 - o_h) \sum_{k \in \{\text{capa } i+1\}} w_{kh}\delta_k$.
13: **fin para**
14: **fin para**
15: Actualizar cada peso w_{ji} de la red neuronal: $w_{ji} = w_{ji} + \Delta w_{ji}$, donde $\Delta w_{ji} = \eta\delta_j x_{ji}$.
16: **fin para**
17: Actualizar los centros y calcular el error residual E.
18: **hasta que** Se alcance la condición de paro.

Consideremos que cada neurona posee un índice que la identifica dentro de cada capa de la red. Usaremos o_j para denotar la salida de la neurona j de determinada capa y δ_j para indicar su error. Usaremos la notación w_{ji} para indicar el peso que la neurona j tiene asignado a su entrada i. De forma similar, x_{ji} indica la entrada i de la neurona j. La entrada w_{j0} siempre será 1. Recordemos que para las neuronas ocultas y las neuronas de salida, esta $i-$ésima entrada está dada por la salida o_i de la neurona i de la capa anterior. Lo anterior se ejemplifica de manera gráfica en la Fig. 2.14.

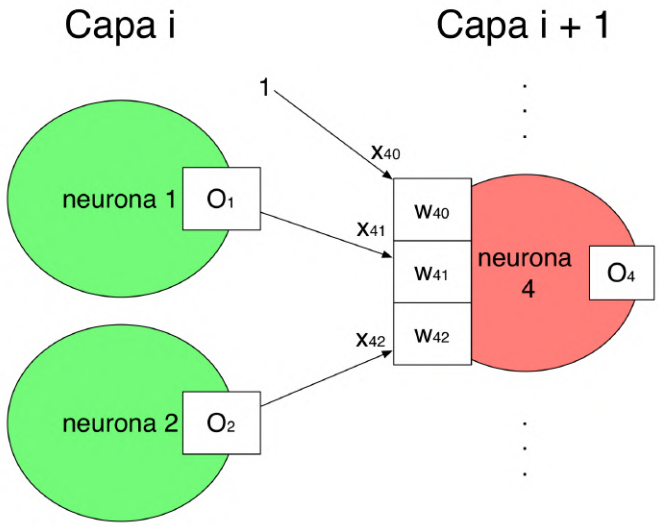

Figura 2.14: Un ejemplo de las relaciones entre las salidas, entradas y pesos de las neuronas de dos capas consecutivas.

El algoritmo de retropropagación corresponde a un método de aprendizaje supervisado, por lo que emplea un conjunto de entrenamiento para realizar el aprendizaje. Representaremos un elemento de este conjunto como $\{(\boldsymbol{\chi}, \boldsymbol{\gamma})\}$, donde $\boldsymbol{\chi}$ es el vector de características de entrada a la red y $\boldsymbol{\gamma}$ es la salida deseada para esta entrada. Al igual que la entrada, la salida deseada puede ser un vector. Por lo tanto, el número de neuronas de salida de la red debe ser igual al número de elementos de este vector. En este sentido, la notación γ_i

indica el i-ésimo componente de $\boldsymbol{\gamma}$ o, equivalentemente, la salida deseada para la i-ésima neurona de la capa de salida. Con lo anterior, el algoritmo 2 presenta el algoritmo de retropropagación para el entrenamiento de redes neuronales

2.3.3. AdaBoost

El *boosting* está formado por una familia de algoritmos que tiene por objetivo generar un clasificador robusto basado en clasificadores débiles. Suponiendo un problema de clasificación binaria en la que las instancias de entrenamiento y de prueba son obtenidas independientemente e idénticamente de acuerdo a una distribución Ds, un clasificador h es débil si su capacidad de discriminación bajo la distribución Ds solo es ligeramente mejor que una decisión aleatoria. El *boosting* es un proceso iterativo que genera clasificadores

Algoritmo 3 El proceso general de *boosting*.

Entrada: Ds, la distribución inicial de instancias, L un algoritmo base de aprendizaje, It el número de iteraciones.

Salida: $H(\boldsymbol{x})$, el clasificador fuerte.

1: $Ds_1 = Ds$ Se inicializa la distribución.
2: **para** $t = 1, \ldots, It$ **hacer**
3: Entrenar un clasificador débil h_t usando la distribución Ds_t y el algoritmo base de aprendizaje L.
4: Medir el error ϵ_t del clasificador h_t.
5: Crear una nueva distribución Ds_{t+1} ajustando la distribución Ds_t de acuerdo al error ϵ_t.
6: **fin para**
7: **devolver** $H(\boldsymbol{x})$ como una combinación de los clasificadores h_t obtenidos.

débiles en cada iteración. Cada clasificador débil h_i se enfoca en los errores de clasificación

de su predecesor h_{i-1}. La distribución de las instancias es modificada en cada iteración de forma que los errores cometidos por el clasificador débil anterior a la iteración actual son puestos en evidencia. El nuevo clasificador se entrena de acuerdo a esta nueva distribución. Al final del proceso, los clasificadores débiles obtenidos se combinan para obtener un clasificador fuerte con un error de clasificación muy pequeño. Este proceso se describe en el Algoritmo 3.

Algoritmo 4 El algoritmo AdaBoost.

Entrada: τ, conjunto de m instancias de entrenamiento, L un algoritmo base de aprendizaje, It el número de iteraciones.

Salida: $H(\boldsymbol{x})$, el clasificador fuerte.

1: $Ds_1 = 1/m$ //Se inicializa la distribución.
2: **para** $t = 1, \ldots, It$ **hacer**
3: $h_t = L(\tau, Ds_t)$ //Entrenar un clasificador débil h_t usando la distribución Ds_t, el conjunto de entrenamiento y el algoritmo base de aprendizaje L.
4: $e_t = Pr_{x \sim Ds_t, y} Ind(\boldsymbol{y} \neq h_t(\boldsymbol{x}))$ //Medir el error ϵ_t del clasificador h_t.
5: **si** $\epsilon > 0.5$ **entonces**
6: terminar.
7: **fin si**
8: $\alpha_t = 1/2 \ln(\frac{1-\epsilon_t}{\epsilon_t})$ //Se determina el peso para el clasificador débil h_t
9: $Ds_{t+1}(i) = \frac{Ds_t(i) \exp(-\alpha_t y_i h_i(\mathbf{x_i}))}{Z_i}$ Crear una nueva distribución Ds_{t+1} ajustando la distribución Ds_t de acuerdo al error ϵ_t. Z_i es un factor de normalización.
10: **fin para**
11: **devolver** $H(\boldsymbol{x}) = \text{sign}(\sum_{t=1}^{It} \alpha_t h_t(\boldsymbol{x}))$.

El error es medido de acuerdo a una función indicadora $Ind(\boldsymbol{y} \neq h_t(\boldsymbol{x}))$ la cual devuelve uno cuando el clasificador débil h_t asigna una etiqueta incorrecta a una instancia de entrenamiento y cero en otro caso. Por lo tanto, el error es una medida de probabi-

lidad de que el clasificador débil se equivoque al momento de clasificar las instancias de entrenamiento. De acuerdo con [45], el error del clasificador h_t fue tomado en este trabajo como la esperanza de que el ejemplo $\boldsymbol{x}$ sea incorrectamente clasificado. La esperanza es calculada respecto a la distribución Ds_t.

$$\epsilon = E_{x \sim Ds}[\boldsymbol{y} \neq h_t(\boldsymbol{x})] \tag{2.21}$$

El Algoritmo 3 presenta un panorama general del proceso de *boosting*. AdaBoost es una instancia de este proceso general, en el que se definen los mecanismos para crear la nueva distribución y la forma de combinar los clasificadores débiles para generar el clasificador final. El método AdaBoost se presenta en el Algoritmo 4.

2.3.4. Máquinas de Vectores de Apoyo

Máquinas de vectores de apoyo o máquinas de soporte vectorial (SVM, por sus siglas en inglés), es un algoritmo de aprendizaje supervisado que se encarga de entrenar un clasificador a partir de un conjunto de entrenamiento $\tau = \{(\boldsymbol{\chi}_i, \gamma_i)\}$, $i = 1, 2, 3, \ldots, n$ con $\boldsymbol{\chi}_i$ un descriptor representativo de los datos y γ_i la clase o tipo a la que pertenece el $i-$ésimo descriptor. Para un problema de clasificación binaria, $\gamma_i \in \{-1, 1\}$. Como se había discutido anteriormente, el objetivo de un clasificador es el de asignar la clase correcta a futuras observaciones del problema, posiblemente ajenos al conjunto de entrenamiento. SVM encuentra el hiperplano óptimo que separa los conjuntos de datos procedentes de cada clase en el problema de clasificación binario. La clasificación se realiza de acuerdo a la siguiente expresión:

$$h_{w,b}(\boldsymbol{\chi}_i) = f(w^T \boldsymbol{\chi}_i + b) \tag{2.22}$$

con $h_{w,b}(\boldsymbol{\chi}_i)$ la hipótesis del clasificador sobre la clase a la que pertenece el descriptor $\boldsymbol{\chi}_i$, w un vector de pesos, b el bias del hiperplano, $\boldsymbol{\chi}_i$ un vector descriptor de los datos a clasificar y $f(z)$ una función tal que $f(z) = 1$ para valores de z mayores o iguales a cero y $f(z) = -1$ en cualquier otro caso [45]. Para esto, las máquinas de vectores de apoyo buscan obtener solución al siguiente problema de optimización:

$$\begin{aligned} &\min_{w,b,\xi} = \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \\ &\text{s. t. } \gamma_i (w^T \phi(\boldsymbol{\chi}_i) + b) \geq 1 - \xi_i, \\ &\xi_i > 0 \end{aligned} \tag{2.23}$$

donde C es el parámetro de penalización.

Para realizar el análisis de patrones se utilizan kernels que mapean los datos a un espacio dimensional más alto para poder encontrar el hiperplano que pueda separar las muestras. La función kernel $K(\cdot)$ se encuentra definida como

$$K(\boldsymbol{\chi}_i, \boldsymbol{\chi}_j) \equiv \phi(\boldsymbol{\chi}_i)^T \phi(\boldsymbol{\chi}_j) \tag{2.24}$$

Para el análisis de patrones, se encuentran definidos 3 kernels básicos: lineal (2.25), polinomial (2.26) y la función de base radial (2.27).

$$K(\boldsymbol{\chi}_i, \boldsymbol{\chi}_j) = \boldsymbol{\chi}_i^T \boldsymbol{\chi}_j \tag{2.25}$$

$$K(\boldsymbol{\chi}_i, \boldsymbol{\chi}_j) = (\rho \boldsymbol{\chi}_i^T \boldsymbol{\chi}_j + r)^d, \rho > 0 \tag{2.26}$$

$$K(\boldsymbol{\chi}_i, \boldsymbol{\chi}_j) = \exp(-\rho \|\boldsymbol{\chi}_i - \boldsymbol{\chi}_j\|^2), \rho > 0 \tag{2.27}$$

La selección de los parámetros γ, r y d de los kernels puede realizarse mediante un proceso de optimización de validación cruzada. Durante este proceso, también se pueden incluir valores del parámetro de penalización, con el fin de determinar su valor óptimo.

2.4. Estado del Arte

Esta sección presenta un resumen de algunos trabajos realizados en el ámbito de segmentación y detección de parásitos sanguíneos, considerando *Trypanosoma cruzi* y *Plasmodium*, el primero responsable de la enfermedad de Chagas y el segundo la causa de la enfermedad conocida como malaria. De forma similar, se mencionan algunos trabajos realizados para resolver problemas relativos a la segmentación y conteo de células sanguíneas.

2.4.1. Detección de *Trypanosoma cruzi*

El número de trabajos que tratan la segmentación y detección de *Trypanosoma cruzi* por medio de métodos de visión computacional es menor, comparado con los de *Plasmodium*. Uno de los trabajos realizados es presentado por Luján [19], donde se describe el desarrollo de un instrumento portable para la toma de muestras sanguíneas y detección de *Trypanosoma cruzi*. El principal objetivo de este trabajo es el de crear un instrumento portable de bajo costo que permita la detección del padecimiento. El proceso de detección del parásito se realiza con base en el movimiento del mismo, calculado con base en un conjunto de imágenes de la muestra. Un proceso de detección similar también se presenta en [28].

En [29], se propone un proceso de detección basado en el movimiento del parásito empleando un microscopio holográfico digital para la adquisición de imágenes.

En [23] se presenta un proceso de segmentación basado en la combinación de los resultados de dos algoritmos de segmentación, el primero es un clasificador Gaussiano entrenado con base en un conjunto de pixeles de entrenamiento y el segundo se obtiene de la resta de los canales verde y azul de los pixeles de la imagen en el espacio de color RGB.

2.4.2. Detección de *Plasmodium*

Existen varios trabajos orientados hacia la detección de *Plasmodium*, el agente biológico causante de la malaria. *Plasmodium*, es un protozoario parásito que, a diferencia de *Trypanosoma cruzi*, carece de flagelo y su principal vector de transmisión son algunas especies de mosquito.

En [39], se presenta un proceso para la detección de *Plasmodium* en imágenes de muestras sanguíneas, tratadas con tinción Giemsa. El proceso se divide en una etapa de segmentación y una de clasificación. La segmentación se realiza mediante un clasificador bayesiano que actúa sobre los pixeles de la imagen. Las distribuciones de probabilidad fueron estimadas empleando el método del histograma, empleando un histograma de tres dimensiones normalizado de forma que la suma de elementos sea igual a uno. La clasificación final se realizó con el método de los k vecinos más cercanos. En la segunda etapa se evaluaron como vectores de características el correlograma, formado mediante el conteo de las co-ocurrencias de dos pixeles que se encuentran a una distancia determinada y poseen un determinado color dentro de un conjunto de colores posibles. Este vector presentó el mejor desempeño respecto a un segundo vector formado por el histograma de color de la imagen.

En [35], se propone un proceso de detección de *Plasmodium* compuesto por etapas de segmentación, extracción de características y la clasificación de los elementos segmentados. El proceso de segmentación se compone de dos etapas: inicialmente, se realiza un análisis morfológico de los elementos presentes en la imagen y, posteriormente, se realiza un proceso de umbralización. El vector de características está conformado por características geométricas, incluyendo la forma y tamaño del elemento segmentado, el color y la textura de estos mismos elementos. Para la etapa de clasificación, se empleó un árbol de

clasificación de dos etapas. La primera etapa determina la presencia del parásito mientras que la segunda determina la especie a la que el parásito pertenece.

En [37] se realiza un proceso de segmentación empleando una imagen binaria obtenida mediante un proceso de umbralización. Se emplea el algoritmo de transformación divisoria para separar los compuestos celulares. Como vectores de características, se realizaron pruebas con características basadas en forma, intensidad, textura e histograma, siendo estos últimos los que presentaron el mejor desempeño.

El proceso de segmentación propuesto en [10] se basa en el espacio de color RGB normalizado y empleando el algoritmo de los k vecinos más cercanos para diferenciar entre los objetos de interés y el fondo de la imagen. El vector de características se compone de la información de cinco histogramas: el histograma de color, un histograma del nivel de saturación, un histograma de la escala de grises, histograma de textura de Tamura y el histograma de nivel de saturación. La dimensionalidad de cada histograma fue reducida para formar un vector final de 25 características. Para clasificar los elementos segmentados se empleó un clasificador basado en perceptrón multicapa comparado con un clasificador basado en máquinas de vectores de soporte (SVM por sus siglas en inglés). El mejor desempeño fue alcanzado mediante el clasificador SVM.

En [30] se emplean métodos de umbralización y métodos morfológicos para la segmentación de los eritrocitos y parásitos. El objetivo del proceso es identificar la presencia de *Plasmodium* y clasificarlo por especie. Se emplean dos vectores de características: el primero se basa en elementos comúnmente empleados en la detección de *Plasmodium* incluyendo descriptores geométricos, atributos de color y textura en niveles de grises. El segundo vector se compone de características cuantitativas presentadas por las células afectadas por el parásito. La clasificación se compone de dos etapas: primero, se determina la presencia de malaria y, posteriormente, se determina la especie a la que pertenece. Para ambas eta-

pas se empleó un clasificador basado en una red neuronal entrenada con el algoritmo de retropropagación.

Purwar [26] emplea métodos de minimización de energía para detectar bordes. Posteriormente, considerando la forma casi circular de los glóbulos blancos, se emplea la transformada de Hough para realizar el conteo de glóbulos blancos. La detección de *Plasmodium* se realiza empleando el algoritmo de agrupamiento k medias.

Mandal [20], realiza la segmentación de células rojas infectadas con malaria empleando del algoritmo de cortes normalizados. Este algoritmo, es un método de segmentación no supervisado que modela la segmentación como un problema de partición de grafos. El algoritmo se probó en los espacios de color RGB, YCbCr, HSV y NTSC, obteniendo mejores resultados con el espacio HSV.

2.4.3. Detección y Conteo de Células

Los trabajos realizados en el área de procesamiento de imágenes obtenidas de muestras sanguíneas, incluyen la segmentación de células sanguíneas para su identificación y conteo. De forma similar, el tratamiento de imágenes microscópicas, no necesariamente obtenidas de muestras sanguíneas, se ha empleado para el reconocimiento y conteo de bacterias. Algunos trabajos presentados en esta ámbito se presentan en esta sección.

Los trabajos presentados en [25] [9] abordan el problema de detección y reconocimiento de células sanguíneas, considerando glóbulos rojos y blancos. En el primero, se presenta un sistema para el reconocimiento, análisis y clasificación de células sanguíneas. El sistema es capaz de producir reportes acerca de los resultados obtenidos. El conteo se realiza con base en un portaobjetos con una cuadrícula de conteo. La cuadrícula es detectada a partir de la detección de bordes y lineas. Posteriormente, para realizar el conteo, se emplean técnicas

de contraste en el canal rojo de la imagen. Posteriormente, se encuentra el contorno de los objetos de la imagen y se discrimina según el área de los contornos obtenidos. Para discriminar entre células sanguíneas y elementos carentes de interés, se emplearon métodos estocásticos basados en la media y la desviación estándar de los contornos obtenidos de los objetos segmentados. El sistema también clasifica las células blancas obtenidas antes de contarlas. La segmentación se realiza mediante mediante un proceso de umbralización sobre el histograma de color RGB. Posteriormente, realiza una discriminación por contornos para determinar los elementos que corresponden a glóbulos bancos. La clasificación se realiza mediante una red neuronal artificial.

El trabajo presentado en [9] aborda el problema de reconocimiento de leucocitos en imágenes de muestras sanguíneas. La clasificación es implementada empleando funciones de mezcla de Gaussianas para modelar las clases disponibles. Las funciones Gaussianas son estimadas mediante el algoritmo de esperanza-maximización. Este algoritmo es inicializado empleando el algoritmo k medias. La dimensionalidad de los datos es reducida empleando el método de análisis de componente principal.

En [46] se propone un método para el reconocimiento de bacterias en aguas residuales. Las imágenes corresponden a muestras de agua analizadas con el microscopio. La etapa de segmentación se realiza mediante la detección de bordes en la imagen. Para la detección, se empleó un vector de características compuesto por descriptores de forma basados en contorno. La dimensionalidad del vector de características es disminuida empleando el análisis de componente principal. Finalmente, el proceso de reconocimiento se realiza mediante una red neuronal entrenada con el algoritmo de retropropagación.

3. Formulación del Problema

El capítulo anterior describió la Trypanosomiasis Americana o Enfermedad de Chagas, al igual que los efectos que esta produce en el ser humano. Una vez desarrollada la etapa crónica de la enfermedad sobreviene un riesgo de mortalidad debido a las complicaciones que esta enfermedad puede producir sobre el músculo cardiaco y sistema digestivo. La detección oportuna en la etapa aguda puede asegurar el tratamiento y la prevención de los efectos adversos de este padecimiento. El diagnóstico certero está dado por la evidencia visual del parásito en la sangre. Lo anterior implica el análisis de muestras sanguíneas en el microscopio, lo cual solo se puede realizar durante la etapa aguda, que es cuando el parásito se encuentra en el torrente sanguíneo. Esta labor es realizada por un especialista con el conocimiento necesario para identificar al parásito en la muestra.

En este mismo capítulo se presentaron algunos trabajos enfocados en la detección de células y parásitos sanguíneos. Varios trabajos se enfocan a la detección de *Plasmodium* en imágenes de muestras sanguíneas, con el objetivo de apoyar a la detección oportuna de la Malaria, enfermedad causada por este parásito. En las metodologías propuestas, es posible distinguir tres etapas: segmentación, extracción de características y detección. La segmentación se encarga de reducir los elementos de la imagen, eliminando los pixeles que representan objetos de poco interés, mientras conserva los pixeles que podrían pertenecer, en este caso, a *Plasmodium*. Las características son una representación numérica de los

resultados de la segmentación. La detección se encarga de realizar la discriminación final sobre las características obtenidas a partir de los objetos segmentados.

En este sentido, en este trabajo se plantea abordar a detalle la primera etapa del proceso de detección, pero esta vez, enfocado a *Trypanosoma cruzi*. La segmentación de *Trypanosoma cruzi* permitirá reducir la cantidad de elementos de la imagen que serán analizados por las etapas subsecuentes. Al igual que en los trabajos con *Plasmodium*, las imágenes son adquiridas mediante el análisis en microscopio de muestras de sangre tratadas con técnicas de tinción.

En el problema de segmentación de *Trypanosoma cruzi*, se propone emplear un procedimiento basado en clasificadores. En este esquema, se propone generar un conjunto de características a partir de los pixeles de la imagen para posteriormente clasificarlos en una de dos clases posibles: objeto de interés o fondo. Si bien un solo pixel puede no ser suficiente para extraer un conjunto de características, la propuesta planteada contempla el uso de grupos de pixeles para la extracción, de forma que las características capturen la relación entre los pixeles y sus vecinos de acuerdo al grupo al que pertenezcan.

Trabajos futuros pueden contemplar problemas específicos del proceso de detección, que complementen el proceso de segmentación propuesto. Lo anterior permitirá la detección automática de *Trypanosoma cruzi*, contribuyendo de esta forma, con el diagnóstico asistido por computadora de esta enfermedad y a su atención oportuna.

4. Metodología

En este capítulo se presenta el procedimiento empleado para segmentar la imagen. Como se había mencionado anteriormente, el proceso calcula un conjunto de súper pixeles en la imagen. Posteriormente, las características son extraídas con base en estos súper pixeles para posteriormente, clasificarlos como fondo o región de interés.

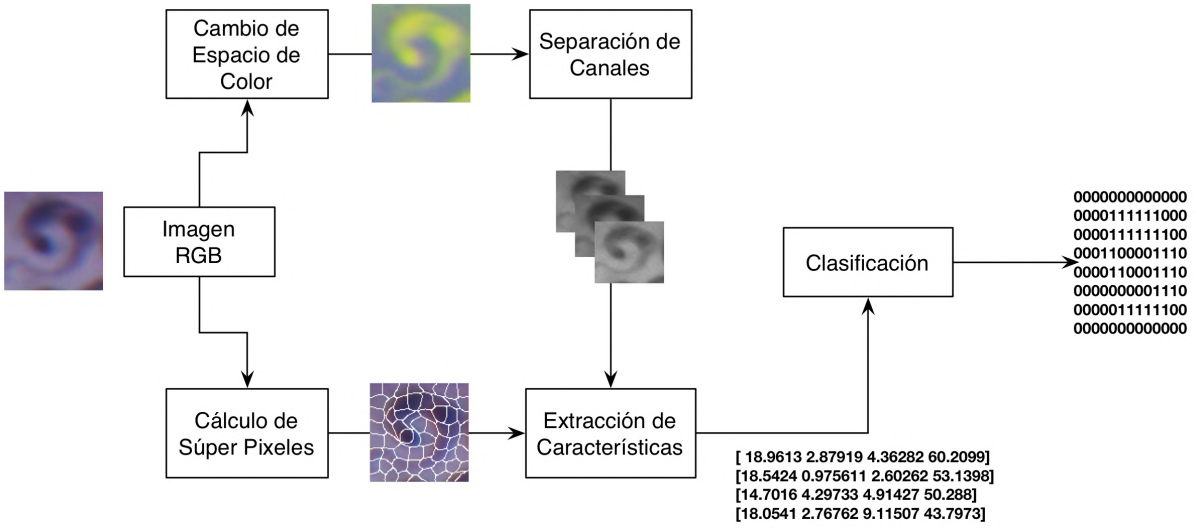

Figura 4.1: Proceso general empleado para la segmentación de imágenes.

Se realizaron pruebas con diversas características y con tres clasificadores. Los clasificadores empleados están basados en redes neuronales entrenadas con el algoritmo de

retropropagación, máquinas de vectores de soporte y un ensamble de perceptrones entrenado con el algoritmo de *boosting* AdaBoost. Las características empleadas están basadas en texturas de Tamura y medidas estadísticas como la media y la varianza de los pixeles que forman el súper pixel. Estas características serán explicadas a mayor detalle en este capítulo. La Fig. 4.1 presenta un panorama general del proceso de segmentación empleado.

4.1. Cómputo de Súper Pixeles

Los súper pixeles fueron discutidos en la sección 2.2.6, en esta sección se discutirán detalles de su implementación y la forma en que fueron empleados. Por comodidad, repetimos el algoritmo 1 de extracción:

1: Inicializar los centros de los grupos $C_k = [l_k, a_k, b_k, x_k, y_x]$. k = 1,2,..., K.
2: Mover los centros a la posición del menor gradiente en un vecindario de 3×3.
3: Inicializar $etiqueta(\boldsymbol{p}) = -1$, $dist(\boldsymbol{p}) = \infty$ para cada pixel $\boldsymbol{p}$.
4: **repetir**
5: **para** cada centro C_k. **hacer**
6: **para** cada pixel $\boldsymbol{p}$ en una región de $2S \times 2S$ centrada en C_k. **hacer**
7: Calcular la distancia $d_{ck,\boldsymbol{p}^5}$ definida como la distancia D_s entre C_k y $\boldsymbol{p}$.
8: **si** $d_{ck,\boldsymbol{p}^5} < dist(\boldsymbol{p})$. **entonces**
9: Hacer $dist(\boldsymbol{p}) = d_{ck,\boldsymbol{p}^5}$ y $etiqueta(\boldsymbol{p}) = k$.
10: **fin si**
11: **fin para**
12: **fin para**
13: Actualizar los centros y calcular el error residual E.
14: **hasta que** $E \leq umbral$.
15: Asegurar la conectividad de cada súper pixel.

Es conveniente repasar algunos detalles de este algoritmo, el cual está basado en el algoritmo de agrupamiento k medias. Las entradas son la imagen con un número total de N pixeles y el número K de súper pixeles que se desea generar. Cada pixel p de la imagen es representado mediante un vector penta-dimensional $\boldsymbol{p}^5$ formado por los componentes lab del pixel en formato CIELAB y su posición (x, y) dentro de la imagen. La métrica de distancia D_s se define de la siguiente forma:

$$D_s(\boldsymbol{p}_k^5, \boldsymbol{p}_i^5) = d_{lab}(\boldsymbol{p}_k^5, \boldsymbol{p}_i^5) + \frac{c}{S} d_{xy}(\boldsymbol{p}_k^5, \boldsymbol{p}_i^5)$$

donde $d_{lab}(\boldsymbol{p}_k^5, \boldsymbol{p}_i^5)$ es la distancia euclidiana entre $\boldsymbol{p}_k^5$ y $\boldsymbol{p}_i^5$ respecto al color CIELAB, $d_{xy}(\boldsymbol{p}_k^5, \boldsymbol{p}_i^5)$ es la distancia euclidiana respecto a su posición en la imagen, $S = \sqrt{N/K}$ y c es un parámetro configurable.

Si bien en [2] se describe todo lo anterior, no presenta una forma de asegurar la conectividad de los súper pixeles, como menciona en el último paso del algoritmo. Sin embargo, una implementación de este algoritmo está disponible en [1] en donde se incluye una metodología para asegurar la conectividad. Es preciso mencionar que durante este trabajo de tesis, se desarrolló una implementación propia de este algoritmo, excepto en esta última etapa de conectividad, la cual fue obtenida como una adaptación del proceso presentado en [1]. Este proceso de conectividad se describe en el algoritmo 5.

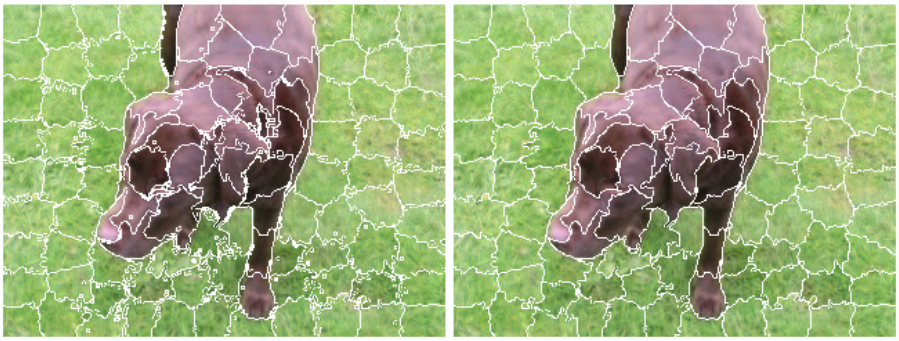

Figura 4.2: Resultados antes (izquierda) y después (derecha) de aplicar el algoritmo 5.

Algoritmo 5 Asegurar Conectividad

1: Inicializar $etiquetaNueva(p) = -1$ para cada pixel $\boldsymbol{p}$ de la imagen.
2: Inicializar $etiqueta = 0$ y $etiquetaAdj = 0$.
3: **para** cada pixel p de la imagen. **hacer**
4: **si** $etiquetaNueva(\boldsymbol{p})$ es igual a -1. **entonces**
5: **para** Cada vecino $\boldsymbol{p'}$ en el vecindario $N(4)$ de $\boldsymbol{p}$. **hacer**
6: **si** $etiquetaNueva(\boldsymbol{p'})$ es diferente de -1. **entonces**
7: Hacer $etiquetaAdj = etiquetaNueva(\boldsymbol{p'})$.
8: **fin si**
9: **fin para**
10: Realizamos un recorrido iniciando en $\boldsymbol{p}$.
11: **para** cada pixel $\boldsymbol{p'}$ en el recorrido. **hacer**
12: visitamos cada vecino $\boldsymbol{p''}$ dentro del vecindario $N(4)$ de $\boldsymbol{p'}$.
13: **si** $etiquetaNueva(\boldsymbol{p''})$ es igual -1. **entonces**
14: Agregamos a $\boldsymbol{p''}$ al recorrido.
15: Hacemos $etiquetaNueva(\boldsymbol{p''}) = etiqueta$.
16: **fin si**
17: **fin para**
18: **si** El número final de pixeles recorridos es menor que determinado valor. **entonces**
19: Hacer $etiquetaNueva(\boldsymbol{p'}) = etiquetaAdj$ para cada pixel $\boldsymbol{p'}$ en el recorrido anterior.
20: Disminuir $etiqueta$ en una unidad.
21: **fin si**
22: Hacemos $etiqueta = etiqueta + 1$.
23: **fin si**
24: **fin para**

Este algoritmo complementa el proceso de extracción de súper pixeles descrito anteriormente. La Fig. 4.2 presenta las diferencias en los resultados antes y después de aplicar el algoritmo de conectividad.

4.2. Extracción de Características

La extracción de características se realiza sobre cada súper pixel generado empleando cada canal de la imagen a color, es decir, la imagen a color es separada con base en sus canales. Por ejemplo, para el caso del espacio de color RGB, la imagen original es separada en tres imágenes de un solo canal. La primera imagen contiene los valores del canal rojo de cada pixel, la segunda contiene los valores del canal verde y la tercera del azul. Las características son entonces extraídas de cada imagen con base a los conjuntos de pixeles contenidos en un súper pixel. Cada característica empleada se obtiene con base a imágenes en escala de gris y el mismo grupo de características es extraído a cada canal de la imagen. El vector final se obtiene agrupando las características extraídas a cada canal. Este proceso se ilustra en la Fig. 4.3.

Para esta parte del proceso se consideraron otros espacios de color ademas del espacio RGB. Particularmente, se emplearon los espacios de color YCbCr y HSV. El espacio de color RGB es un modelo orientado a hardware diseñado para ser empleado en los monitores a color. El espacio de canal HSV es nombrado así por las siglas de *Hue* (matiz), *Saturation* (saturación) y *Value* (valor). El canal H representa un atributo de color; la saturación, o canal S, describe la cantidad de brillo o luz blanca relativo a la matiz (H). Mientras el canal H describe únicamente color, el canal S representa el grado de "pureza" de este color; finalmente, el canal V describe la cantidad de brillo en la imagen. El espacio de color YCbCr es una representación de color de tres canales orientado a hardware. Este

espacio separa la luminancia o brillo de la crominancia. El canal Y representa este valor de brillo mientras que los canales Cb y Cr representan la crominancia en azul y en rojo respectivamente [20].

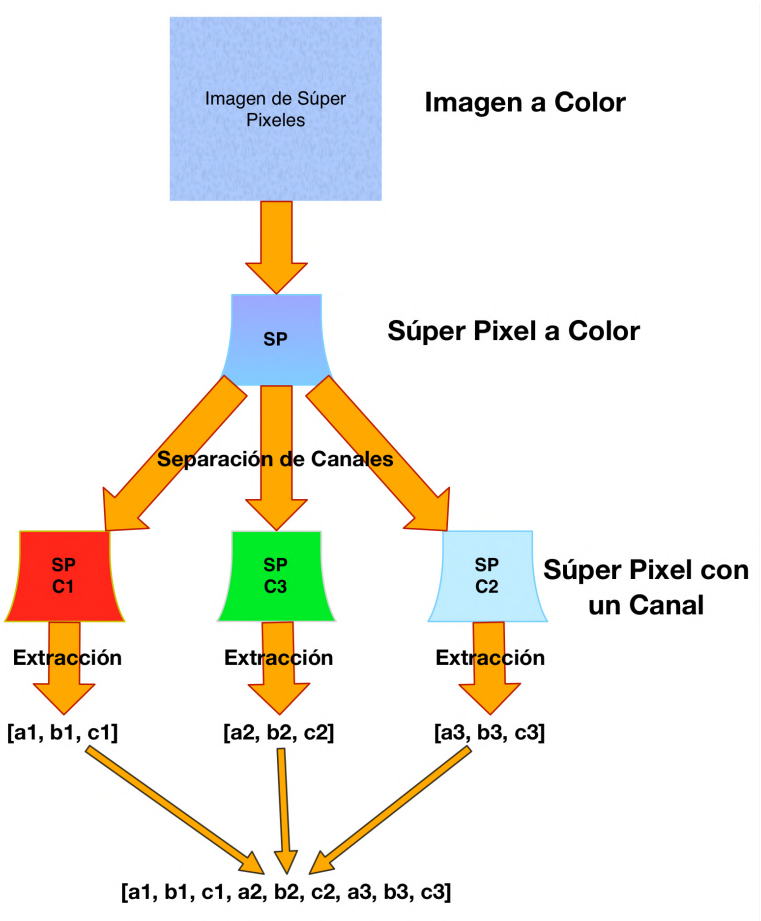

Figura 4.3: Proceso de extracción de características de la imagen.

Adicionalmente a los espacios mencionados anteriormente, se consideró un canal formado por la diferencia de los canales azul y verde de la imagen en el espacio RGB. La principal motivación de lo anterior fue una investigación previa donde se mostraba que es-

tos canales almacenaban información sobre los elementos teñidos pertenecientes al parásito [34].

Las características extraídas se basan en la textura, la media y la varianza de cada súper pixel. Las características basadas en textura se han empleado en problemas de detección y reconocimiento de objetos en imágenes y en problemas de recuperación de imágenes [15] [16], incluyendo la detección de *Plasmodium* y la recuperación de imágenes médicas [21] [35] [18]. Las características de textura de Tamura se basan en la percepción humana de la textura y han sido empleadas en los problemas antes mencionados [15] [18] [43] [10].

En este proyecto se emplearon las características de textura de Tamura para representar la información contenida en un súper pixel. Tamura propuso seis características para describir la textura, de las cuales las primeras tres se encuentran fuertemente correlacionadas con la percepción humana de textura, mientras que las tres restantes se encuentran fuertemente correlacionadas con las tres primeras, aportando menor información adicional [10] [15] [12]. Las características empleadas corresponden a la aspereza, el contraste y la direccionalidad de la imagen.

4.2.1. Aspereza

La aspereza es una característica relacionada con la escala y describe el tamaño más grande en el cual la textura existe [15]. Es calculada a partir de diferencias entre los promedios de dos vecindarios disjuntos de pixeles, es decir, que no poseen pixeles en común. Un vecindario es descrito como una región cuadrada de tamaño $2^k \times 2^k$, para diferentes valores de k y centrado en el punto (x, y) de la imagen.

El procedimiento para calcular la aspereza de la imagen es el siguiente: primero es necesario calcular los promedios de los vecindarios para distintos valores de k, esto se

realiza de la siguiente forma:

$$A_k(x,y) = \sum_{i=x-2^{k-1}}^{x+2^{k-1}-1} \sum_{j=y-2^{k-1}}^{y+2^{k-1}-1} \frac{I(i,j)}{2^{2k}} \tag{4.1}$$

con $I(x,y)$ una imagen en escala de grises, A es una matriz que almacena el resultado obtenido en cada punto. Al final de proceso se obtendrán un número k de matrices A.

Posteriormente, el proceso continúa con el cálculo de diferencias entre dos promedios obtenidos de vecindarios disjuntos. Estos vecindarios deben estar localizados en lados opuestos con referencia a la dirección vertical y horizontal. Los promedios de los vecindarios son obtenidos con base a las matrices calculadas en el paso anterior. La diferencia se obtiene, para la dirección horizontal, con la siguiente igualdad:

$$E_{k,h}(x,y) = |A_k(x+2^{k-1},y) - A_k(x-2^{k-1},y)| \tag{4.2}$$

de forma similar, para la dirección vertical tenemos:

$$E_{k,v}(x,y) = |A_k(x,y+2^{k-1}) - A_k(x,y-2^{k-1})| \tag{4.3}$$

Una vez calculadas las diferencias, se encuentra el valor k que maximiza la diferencia en el punto (x,y) en cualquier dirección. Lo anterior puede ser calculado de la siguiente forma: primero, para cada valor k, se encuentra la dirección que maximiza la diferencia en el punto (x,y) y se almacena el valor obtenido, es decir:

$$c_k(x,y) = \max[E_{k,h}(x,y), E_{k,v}(x,y)] \tag{4.4}$$

Posteriormente, para cada punto (x,y), se determina el valor de k que maximiza $c_k(x,y)$ y se almacena de la siguiente forma:

$$S_{best}(x,y) = 2^{k_{best}(x,y)} \tag{4.5}$$

donde

$$k_{best}(x,y) = \arg\max_{k} c_k(x,y) \tag{4.6}$$

Una vez calculada la matriz S_{best}, la aspereza f_{crs} de la imagen se calcula como el promedio de los valores óptimos almacenados en esta matriz. En las primeras etapas de este proceso es necesario calcular sumatorias sobre los pixeles de diferentes regiones cuadradas, esto con la finalidad de obtener los promedios de estas regiones. Para agilizar el cálculo de estas sumas es posible calcular la imagen integral de I antes de iniciar el proceso. El proceso de obtención de la imagen integral fue discutido en la sección 2.2.5 de este documento.

4.2.2. Contraste

El contraste mide en que medida varían los niveles de gris de la imagen, es decir, el rango dinámico de los niveles de gris, junto con la forma en que su distribución se encuentra sesgada hacia el negro y blanco, descrito como la polarización de la distribución de blanco y negro [16] [12].

El contraste es calculado con base en la varianza y la curtosis del histograma de la imagen en escala de grises. El histograma representa la frecuencia con la que cada valor de gris se repite en la imagen y fue presentado en la sección 2.2.2. Para calcular el contraste de la imagen, el histograma fue normalizado de forma que la suma de sus valores sea igual a uno. El histograma normalizado H_{norm} de la imagen puede ser definido de la siguiente manera:

$$H_{norm}[v] = \frac{\sum_{y=0}^{m-1}\sum_{x=0}^{n-1} Ind(I(x,y)=v)}{(m)(n)} \tag{4.7}$$

donde $v \in \{0, 1, 2, \ldots, v_{max}\}$ un valor de la escala de grises de la imagen , con v_{max} el máximo valor posible, I es una imagen en escala de grises de dimensiones $m \times n$ e $Ind(\cdot)$ es una función que devuelve uno cuando su argumento es verdadero y cero en otro caso. Una vez calculado el histograma, la varianza σ^2 de la imagen es calculada mediante la siguiente expresión:

$$\sigma^2 = \sum_{v=0}^{v_{max}} (v - \mu)^2 H_{norm}[v] \tag{4.8}$$

con μ la media de la distribución de pixeles de la imagen obtenida de la siguiente forma [42]:

$$\mu = \sum_{v=0}^{v_{max}} v H_{norm}[v] \tag{4.9}$$

La curtosis α_4 se calcula mediante:

$$\alpha_4 = \frac{\mu_4}{\sigma^4} \tag{4.10}$$

con μ_4 el cuarto momento centrado en la media, definido como:

$$\mu_4 = \sum_{v=0}^{v_{max}} (v - \mu)^4 H_{norm}[v] \tag{4.11}$$

Con base en lo anterior, el contraste f_{con}de la imagen es obtenido empleando la siguiente expresión [15]:

$$f_{con} = \frac{\sigma}{\mu_4^n} \tag{4.12}$$

De acuerdo con la literatura, el valor de $n = 1/4$ es recomendado como el mejor valor para discriminar texturas [15] [12].

4.2.3. Direccionalidad

La direccionalidad representa una propiedad global sobre una región y describe el grado global de direccionalidad. Esta característica es medida empleando la distribución de bordes locales orientados contra sus ángulos direccionales [43] [12].

Para calcular la direccionalidad, es necesario obtener la intensidad de borde $e(x, y)$ y el ángulo direccional $\alpha(x, y)$ en el punto (x, y) de la imagen. Estos valores son calculados mediante las siguientes igualdades:

$$e(x, y) = 0.5(|\Delta_x(x, y)|+|\Delta_y(x, y)|) \tag{4.13}$$

$$\alpha(x, y) = \tan^{-1}(\frac{\Delta_y(x, y)}{\Delta_x(x, y)}) \tag{4.14}$$

En las fórmulas anteriores, Δ_x y Δ_y pueden entenderse como aproximaciones a la derivada de la imagen en el punto (x, y) y son obtenidas mediante la convolución de la imagen, para el caso de Δ_x, con la máscara:

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{4.15}$$

y para Δ_y con:

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{4.16}$$

Una vez calculados estos dos valores, se construye un histograma H_{dir} empleando los valores de $\alpha(x, y)$. Al igual que el histograma de la imagen en escala de grises, H_{dir} se obtiene con base en la frecuencia con la que cada valor cuantificado de $\alpha(x, y)$ se repite en la imagen. Sin embargo, para que un valor sea contabilizado, es necesario que su intensidad de borde $e(x, y)$ asociado sea mayor que un umbral predefinido [43]. Lo anterior se puede expresar de la siguiente forma:

$$H_{dir}[\theta] = \sum_{i=0}^{n-1} Ind(Q(\alpha(x, y)) = \theta) Ind(e(x, y) > T) \tag{4.17}$$

Nuevamente, $Ind(\cdot)$ es un indicador que retorna uno cuando su argumento es verdadero y cero en otro caso; θ es el ángulo cuantificado; Q es una función que expresa el proceso de cuantificación de $\alpha(x, y)$ y T es el umbral para los valores de intensidad de borde.

Para cuantificar los ángulos definidos en $\alpha(x, y)$, se emplearon 12 intervalos en el histograma, donde cada intervalo representa un ángulo $\theta \in \{-75, -60, -45, \ldots, 75, 90\}$ medido en grados. Para realizar el proceso de cuantificación, el valor de $\alpha(x, y)$ para un determinado punto (x, y) es asociado al ángulo θ más cercano, es decir:

$$Q(\alpha(x, y)) = \arg\min_{\theta} |\alpha(x, y) - \theta| \tag{4.18}$$

Una vez elaborado el histograma, es necesario encontrar los valles y los picos presentes en él. Consideraremos un pico como un valor en el histograma cuya frecuencia es mayor que su predecesor y sucesor. De forma similar, consideraremos un valle como el valor en el histograma con menor frecuencia que se encuentre entre dos picos (Fig. 4.4).

Para encontrar estos puntos en el histograma realizamos restas entre dos valores consecutivos del mismo. Consideraremos como un pico el valor del histograma donde las restas respecto a su antecesor sean positivas y las de su sucesor sean negativas. En un valle las

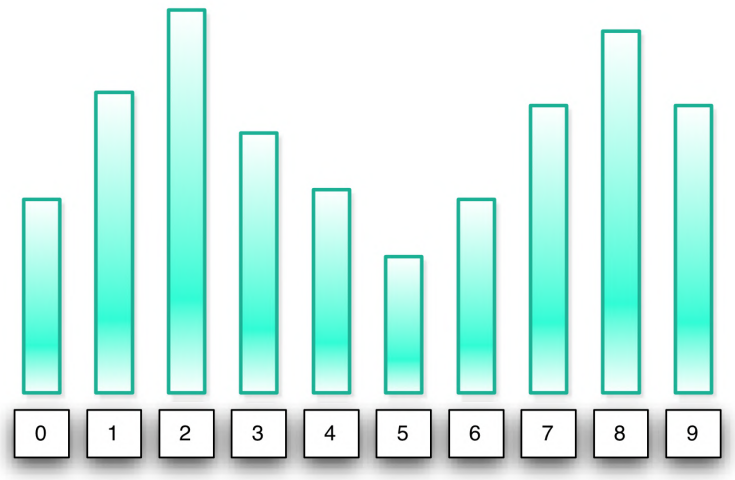

Figura 4.4: En este histograma existen picos en los valores 2 y 8. Existe un valle en el valor 5.

restas de su antecesor son negativas mientras que las de su sucesor son positivas. Lo anterior es una versión discreta de la derivada para encontrar puntos máximos y mínimos en una función. En [17] proponen considerar la naturaleza periódica de los ángulos en la búsqueda de picos y valles. En este sentido, los bordes orientados en 90 grados son iguales a los -90 grados. Por lo tanto, se consideró indexar el histograma de forma circular. Por ejemplo, en la Fig. 4.4 el valor que sigue a 9 sería 0. En este sentido, en esta figura existiría un valle en 0 y en 5.

Consideremos que existen un número de n_p picos en el histograma con ϕ_i la posición angular del i-ésimo pico. También, llamemos v_i al conjunto de valores angulares comprendidos entre el valle anterior y el valle posterior al pico ϕ_i. Con base en lo anterior, la direccionalidad es calculada mediante:

$$F_{dir} = \sum_{i=1}^{n_p} \sum_{\theta \in v_i} (\theta - \phi_i)^2 H_{dir}(\theta) \tag{4.19}$$

Como se había mencionado anteriormente, estas características son calculadas sobre el subconjunto de pixeles de la imagen delimitado por un súper pixel.

4.3. Clasificación y Segmentación

El proceso de clasificación emplea el vector de atributos para decidir si el súper pixel asociado pertenece o no a la región de interés y de esta forma segmentar la imagen. En este trabajo se evaluaron tres clasificadores: Máquinas de vectores de apoyo, Redes neuronales y un ensamble de neuronas perceptrón entrenado con AdaBoost. Adicionalmente a estos clasificadores, también se compararon tres algoritmos empleados en la segmentación de imágenes: Naive Bayes, Clasificación Gaussiana y el procedimiento de segmentación de *Trypanosoma cruzi* propuesto en [34].

4.3.1. AdaBoost y Perceptrón

El algoritmo de aprendizaje AdaBoost fue descrito anteriormente en la sección 2.3.3, de forma similar, la neurona artificial perceptrón se presentó en la sección 2.3.2. Recordemos que, por una parte, una neurona perceptrón evalúa una función lineal sobre un conjunto de características de entrada $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$. Esta función se define de la siguiente manera:

$$o(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \text{si } w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n > 0 \\ -1 & \text{de otra forma} \end{cases} \tag{4.20}$$

donde cada w_i es un valor real que representa el peso asignado a cada característica. Diferentes pesos definen diferentes neuronas y la labor del proceso de aprendizaje es encontrar el conjunto de pesos que maximice la capacidad predictiva de la neurona.

Por otra parte, el algoritmo AdaBoost construye un clasificador robusto a partir de la suma ponderada de los resultados de un conjunto de clasificadores débiles. En este algoritmo se pueden distinguir dos procesos: la selección o entrenamiento de cada clasificador débil y la ponderación de los mismos. El algoritmo describe como ponderar cada clasificador débil, sin embargo, el proceso de selección depende del clasificador a emplear. En esta sección se describe el proceso empleado para generar un ensamble de neuronas perceptrón empleando el algoritmo AdaBoost.

Para iniciar, describiremos un procedimiento para encontrar los pesos de la neurona y, posteriormente, explicaremos la modificación realizada a este proceso para que pueda ser empleado junto con AdaBoost. En el capítulo cuatro de [22], se presenta un algoritmo de aprendizaje para perceptrón basado en el método del gradiente descendente. Este proceso, realiza una búsqueda de los pesos que mejor se ajustan a un conjunto de entrenamiento.

Para medir el desempeño de la neurona, consideremos el error cuadrático medio para un perceptrón definido mediante su vector de pesos $\boldsymbol{w}$:

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{d\in D}(t_d - o_d)^2 \tag{4.21}$$

donde D es el conjunto de ejemplos de entrenamiento, t_d es la etiqueta verdadera asignada al ejemplo d y o_d es la etiqueta estimada obtenida mediante la salida del perceptrón. El método del gradiente descendente, determina un vector de pesos que minimice este error. Para esto, inicia con un vector arbitrario que posteriormente es modificado en pequeños pasos, de forma que el error decrezca. El gradiente de una función indica la dirección

de mayor crecimiento, por lo tanto, incrementar el vector de pesos con base en el valor ponderado del negativo del gradiente del error producirá el efecto de minimización deseado [22]. Para esto, es necesario calcular el gradiente del error respecto a los pesos, el cual se define de la siguiente forma:

$$\nabla E(\boldsymbol{w}) = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n}\right] \tag{4.22}$$

Para obtener el gradiente, es necesario calcular la derivada del error respecto cada peso $w_i \in \boldsymbol{w}$, esto se obtiene de la siguiente forma:

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i}\frac{1}{2}\sum_{d\in D}(t_d - o_d)^2$$

Aplicando las propiedades de la derivada tenemos:

$$\frac{\partial E}{\partial w_i} = \frac{1}{2}\sum_{d\in D}\frac{\partial}{\partial w_i}(t_d - o_d)^2$$

Con base en la regla de la cadena, derivamos la expresión anterior obteniendo:

$$\frac{1}{2}\sum_{d\in D}\frac{\partial}{\partial w_i}(t_d - o_d)^2 = \frac{1}{2}\sum_{d\in D}2(t_d - o_d)\frac{\partial}{\partial w_i}(t_d - o_d) \tag{4.23}$$

Consideremos, ahora, la derivada parcial a la derecha de la ecuación anterior:

$$\frac{\partial}{\partial w_i}(t_d - o_d) = \frac{\partial}{\partial w_i}t_d - \frac{\partial}{\partial w_i}o_d$$

Considerando la función de evaluación de perceptrón (4.20), reescribimos el lado derecho de la expresión anterior como:

$$\frac{\partial}{\partial w_i}t_d - \left[\frac{\partial}{\partial w_i}w_0x_0 + \frac{\partial}{\partial w_i}w_1x_{1d} + \cdots + \frac{\partial}{\partial w_i}w_ix_{id} + \cdots + \frac{\partial}{\partial w_i}w_nx_{nd}\right]$$

con $x_0 = 1$ y x_{jd} el valor de la $j-$ésima característica del ejemplo d . Al derivar la expresión anterior respecto a w_i obtenemos:

$$\frac{\partial}{\partial w_i}(t_d - o_d) = -x_{id} \tag{4.24}$$

Sustituyendo la expresión anterior en 4.23, obtenemos:

$$\frac{\partial E}{\partial w_i} = -\sum_{d \in D}(t_d - o_d)x_{id} \tag{4.25}$$

La regla de actualización para los pesos del perceptrón es entonces:

$$w_i = w_i + \Delta w_i \tag{4.26}$$

con $\Delta w_i = \eta \sum_{d \in D}(t_d - o_d)x_{id}$, donde η es un factor de ponderación para el gradiente conocido como constante de aprendizaje.

AdaBoost requiere entrenar el clasificador con base a la distribución Ds_t de los ejemplos de entrenamiento. Para considerar el efecto de la distribución sobre el error de entrenamiento, reescribimos el error de la siguiente forma:

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{d \in D} Pr(d)(t_d - o_d)^2 \tag{4.27}$$

con $Pr(d)$ la probabilidad correspondiente para el ejemplo d, definida por la distribución Ds_t. Siguiendo un procedimiento similar al anterior, calculamos el gradiente de este error:

$$\frac{\partial}{\partial w_i}\frac{1}{2}\sum_{d \in D} Pr(d)(t_d - o_d)^2 = \sum_{d \in D} Pr(d)(t_d - o_d)\frac{\partial}{\partial w_i}(t_d - o_d)$$

Sustituyendo 4.24 en la ecuación anterior obtenemos:

$$\frac{\partial E}{\partial w_i} = -\sum_{d \in D} Pr(d)(t_d - o_d)x_{id} \tag{4.28}$$

La expresión anterior define el gradiente que se debe restar a los pesos para actualizar su valor. El valor de Δw_i en la regla de actualización 4.26, queda definida ahora en términos de este nuevo gradiente. Empleando esta expresión, es posible entrenar el clasificador considerando el conjunto de entrenamiento y la distribución de este conjunto, ambos valores proporcionados por el algoritmo AdaBoost.

Tomando en cuenta lo anterior, se describe el algoritmo para entrenar el clasificador débil basado en perceptrón:

Algoritmo 6 Proceso de aprendizaje para perceptrón

Entrada: Ds, la distribución inicial de instancias (para usar junto con AdaBoost), D, conjunto de ejemplos de entrenamiento.

Salida: $h(\boldsymbol{x})$, el clasificador débil.

1: Inicializar cada peso w_i con un valor aleatorio.
2: **repetir**
3: Inicializar cada Δ_j en cero.
4: **para** cada peso w_i de le neurona. **hacer**
5: Calcular $\Delta w_i = \Delta w_i - \eta \frac{\partial E}{\partial w_i}$ Usando la ecuación 4.25 para un entrenamiento habitual o 4.28 para su uso con AdaBoost.
6: **fin para**
7: **para** Cada peso w_i. **hacer**
8: $w_i = w_i + \Delta w_i$.
9: **fin para**
10: **hasta que** La condición de paro sea alcanzada.
11: **devolver** $h(\boldsymbol{x})$ un clasificador basado en la neurona perceptrón.

4.3.2. Naive Bayes y Clasificación Gaussiana

El clasificador *Naive* Bayes emplea el teorema de Bayes para decidir la clase a la cual pertenece determinado elemento. Dado un vector de atributos $\boldsymbol{x} = \{x_1, x_2, \ldots, x_D\}$ y la variable clase c, donde $\text{dom}(c) = \{0, 1\}$ y $\text{dom}(x_i) = \{0, 1, 2 \ldots, S\}, x_i \in \boldsymbol{x}$, *Naive* Bayes asume que los atributos son independientes entre ellos dada la clase c y, por lo tanto, la distribución conjunta de $\boldsymbol{x}$ y c está dada por:

$$p(\boldsymbol{x}, c) = p(c) \prod_{i=1}^{D} p(x_i|c) \tag{4.29}$$

La clasificación de un nuevo elemento $\boldsymbol{x_{new}}$ se realiza comparando las probabilidades de que pertenezca a alguna de las clases, de modo que clase$(\boldsymbol{x_{new}}) = 1$ si

$$p(c = 1|\boldsymbol{x_{new}}) > p(c = 0|\boldsymbol{x_{new}}) \tag{4.30}$$

donde $(c|\boldsymbol{x_{new}})$ está dada por la regla de Bayes:

$$p(c|\boldsymbol{x_{new}}) = \frac{p(\boldsymbol{x_{new}}|c)p(c)}{p(\boldsymbol{x_{new}})} = \frac{p(\boldsymbol{x_{new}}|c)p(c)}{\Sigma_c p(\boldsymbol{x_{new}}|c)p(c)} \tag{4.31}$$

Por lo tanto, para poder usar el clasificador es necesario aprender las tablas de probabilidad para $p(\boldsymbol{x}|c)$ y $p(c)$, con $\boldsymbol{x} = \{x_1, x_2, \ldots, x_D\}$. Considerando la independencia condicional entre los elementos de $\boldsymbol{x}$ tenemos:

$$p(\boldsymbol{x}|c) = p(x_1|c)p(x_2|c)p(x_3|c) \ldots p(x_D|c) \tag{4.32}$$

Dado un conjunto de datos de entrenamiento Dat $= (\boldsymbol{x}^n, c^n), n = 1, 2, 3, \ldots, N$, con $\text{dom}(c) = \{0, 1\}$ y $\text{dom}(x_i) = \{0, 1, 2 \ldots, S\}$, las tablas de probabilidad para cada $p(x_i|c)$ pueden ser calculadas de la siguiente manera [3]:

$$p(x_i = s|c) = \frac{\Sigma_n \text{I}[x_i^n = s]\text{I}[c^n = c]}{\Sigma_{s',n'} \text{I}[x_i^{n'} = s']\text{I}[c^{n'} = c]} \tag{4.33}$$

Donde I$[arg]$ es una función indicadora que devuelve uno si la expresión arg es verdadera y x_i^n es el i-ésimo elemento del vector de atributos $\boldsymbol{x^n}$. De la expresión anterior se tiene que las tablas de probabilidad se obtienen contando el número de veces que el elemento x_i adquiere el valor s dentro de clase c. Posteriormente, este valor es normalizado usando el total de elementos dentro la clase c. Lo anterior es con base a los elementos del conjunto de entrenamiento. La probabilidad $p(c)$ se calcula de forma similar, dividiendo el total de elementos dentro de la clase c entre el total de elementos del conjunto de entrenamiento Dat.

El clasificador Gaussiano [24] actúa como un clasificador binario sobre los pixeles de la imagen. Para este problema de segmentación las clases se dividen en pixeles teñidos $\boldsymbol{p_s}$ y pixeles no teñidos $\boldsymbol{p_{ns}}$, donde los pixeles se encuentran en el espacio de color RGB. Emplea un modelo basado en el vector media y la matriz de covarianza obtenida de un conjunto de pixeles de cada clase. Lo anterior sugiere que es necesario tener un conjunto de entrenamiento conformado por elementos teñidos y no teñidos, de forma que estos parámetros puedan ser estimados. El vector de medias está dado por:

$$\boldsymbol{\mu_p} = \begin{bmatrix} \frac{1}{n}\sum r \\ \frac{1}{n}\sum g \\ \frac{1}{n}\sum b \end{bmatrix} \tag{4.34}$$

es decir, por la sumatoria de los canales r, g y b de los pixeles contenidos en el conjunto de entrenamiento divido entre el total de pixeles del conjunto, denominado como n en la igualdad anterior.

Una vez calculado el vector de medias, la matriz de covarianza es calculada mediante la siguiente igualdad:

$$cov_p = E[(\boldsymbol{p} - \boldsymbol{\mu_p})(\boldsymbol{p} - \boldsymbol{\mu_p})^t] \tag{4.35}$$

donde E representa la esperanza, $\boldsymbol{\mu_p}$ se refiere al vector media de la distribución de pixeles y el subíndice t se refiere a la transpuesta de $(\boldsymbol{p} - \boldsymbol{\mu_p})$. Recordemos que el pixel $\boldsymbol{p}$ se puede ver como un vector de tres dimensiones, por lo que la matriz de covarianza será una matriz de 3×3.

Para determinar si un pixel determinado $\boldsymbol{p_u}$ pertenece a una u otra clase, se somete a la siguiente regla de clasificación:

$$[(\boldsymbol{p_u} - \boldsymbol{\mu_s})^t cov_s^{-1}(\boldsymbol{p_u} - \boldsymbol{\mu_s})] - [(\boldsymbol{p_u} - \boldsymbol{\mu_{ns}})^t cov_{ns}^{-1}(\boldsymbol{p_u} - \boldsymbol{\mu_{ns}})] > T \tag{4.36}$$

donde $\boldsymbol{\mu_s}$ y cov_s^{-1} son el vector media y la matriz de covarianza inversa obtenida de las muestras teñidas, $\boldsymbol{\mu_{ns}}$ y cov_{ns}^{-1} son el vector media y matriz de covarianza inversa obtenidos del conjunto de muestras no teñidas, T es un umbral de comparación y $\boldsymbol{p_u}$ es el pixel a clasificar. El umbral T indica cuanto más cerca puede estar $\boldsymbol{p_u}$ a la distribución de no teñidos y aún así seguir siendo considerado como un pixel teñido. Si la condición establecida por la ecuación (4.36) se cumple, entonces el pixel $\boldsymbol{p_u}$ es clasificado como no teñido. El Algoritmo 7 describe este proceso.

Algoritmo 7 Segmentación por clasificación Gaussiana.

Entrada: Imagen $I(x, y)$ tomada de una muestra sanguínea en formato RGB. Matriz de covarianza cov_s y vector de medias $\boldsymbol{\mu_s}$ de la clase teñida. Matriz de covarianza cov_{ns} y vector de medias $\boldsymbol{\mu_{ns}}$ de la clase no teñida.

Salida: Imagen binaria segmentada $G(x, y)$.

1: **para todo** pixel $\boldsymbol{p}$ de la imagen $I(x, y)$ en la posición (x, y), donde $\boldsymbol{p}$ es un pixel en formato RGB. **hacer**

2: Asignar un valor a $G(x, y)$ en la posición (x, y) de acuerdo a la siguiente regla

$$G(x,y) = \begin{cases} 0 & \text{si } [(\boldsymbol{p}-\boldsymbol{\mu_s})^t cov_s^{-1}(\boldsymbol{p}-\boldsymbol{\mu_s})] - [(\boldsymbol{p}-\boldsymbol{\mu_{ns}})^t cov_{ns}^{-1}(\boldsymbol{p}-\boldsymbol{\mu_{ns}})] > T \\ 1 & \text{de otra forma} \end{cases}$$

3: **fin para**

4: **devolver** La imagen segmentada $G(x, y)$.

4.3.3. Segmentación de *Trypanosoma cruzi*

El proceso de segmentación propuesto en [34], combina los resultados de dos algoritmos de segmentación para obtener el resultado final. Uno de los algoritmos es el clasificador Gaussiano descrito en la sección anterior, mientras que el segundo es una propuesta basada en la umbralización de la imagen resultante de restar los canales azul y verde de la imagen original en formato RGB.

Este proceso de resta, nombrado como diferencia de canales, se basa en las propiedades que presenta la imagen debido al efecto del colorante empleado al momento de teñir la muestra. En un pixel $\boldsymbol{p}$ perteneciente al cinetoplasto de *Trypanosoma cruzi* el canal azul presenta valores mayores comparado con el canal verde del mismo pixel. El algoritmo de segmentación por diferencia de canales es el siguiente:

Algoritmo 8 Diferencia de canales.

Entrada: Imagen $I(x, y)$ tomada de una muestra sanguínea en formato RGB.

Salida: Imagen binaria segmentada $G(x, y)$.

1: **para todo** pixel $\boldsymbol{p}$ de la imagen $I(x, y)$ en la posición (x, y), donde $\boldsymbol{p}$ se encuentra en formato RGB. **hacer**

2: Asignar un valor a $G(x, y)$ en la posición (x, y) de acuerdo a la siguiente regla:

$$G(x,y) = \begin{cases} 1 & \text{si } \mid b - g \mid > T \\ 0 & \text{de otra forma} \end{cases}$$

3: **fin para**

4: **devolver** La imagen segmentada $G(x, y)$.

Una vez realizado este proceso, prosigue la eliminación de objetos que fueron segmentados, pero que carecen de interés. Debido que esta parte del proceso tiene por objetivo identificar las estructuras celulares que pertenecen al parásito (núcleo y cinetoplasto), los objetos de gran tamaño pueden ser despreciados. Para esto se calcula el área de los objetos segmentados mediante un proceso de etiquetación. Este método asigna el mismo valor (etiqueta) a todos los pixeles de una misma región. De modo similar, diferentes regiones poseen diferentes etiquetas. El área puede ser calculada contando el número de pixeles con la misma etiqueta. El algoritmo 9 presenta un proceso de etiquetación.

La última etapa del proceso de segmentación de *Trypanosoma cruzi* es la intersección de los resultados del clasificador Gaussiano y la diferencia de canales. Para esto, los resultados son tomados a partir de los resultados obtenidos por clasificación Gaussiana , siempre que cumplan con el siguiente criterio: si la región segmentada por el clasificador Gaussiano contiene un elemento obtenido de los resultados de la segmentación por diferencia de canales, la región es considerada como una región de interés, es decir, se toman como regiones de interés los resultados del clasificador Gaussiano que pertenezcan a la

Algoritmo 9 Etiquetación.

Entrada: Imagen binaria $IB(x, y)$ resultado de la segmentación.

Salida: Imagen etiquetada $E(x, y)$.

$etiqueta = 1$. // inicializar contador de etiquetas en 1.

para todo pixel $\boldsymbol{p}$ de la imagen $IB(x, y)$ en la posición (x, y) que sea diferente del fondo. **hacer**

Revisar el vecindario N_8 de $\boldsymbol{p}$.

si Algún vecino de $\boldsymbol{p}$ ya posee una etiqueta v **entonces**

Asignar $E(x, y) = v$.

si no

Asignar $E(x, y) = etiqueta$.

Incrementar *etiqueta* una unidad.

fin si

fin para

devolver La imagen etiquetada $E(x, y)$.

intersección de los resultados de ambos métodos de segmentación [23] [34]. La Fig. 4.5 ilustra el proceso descrito.

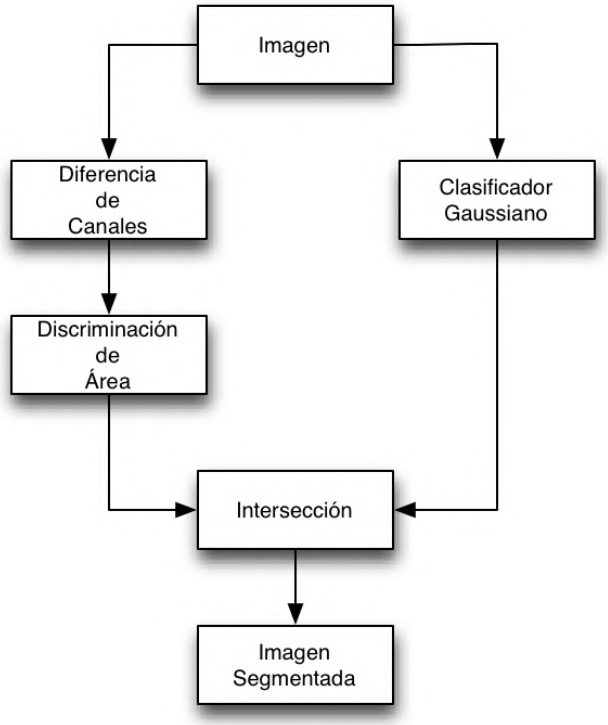

Figura 4.5: Etapas del proceso de segmentación de *Trypanosoma cruzi*. El resultado final está dado por la intersección de los resultados intermedios de la segmentación por diferencia de canales y la segmentación por clasificación Gaussiana.

4.4. La Biblioteca OpenCV

OpenCV (Open Computer Vision) es un biblioteca para el desarrollo de aplicaciones de visión computacional desarrollada por Intel^{TM}que permite la manipulación de imágenes así como el procesamiento de las mismas. Es capaz de manipular varios formatos, entre ellos,

jpg, png, pgm, tiff entre otros. También puede obtener imágenes desde una video cámara o desde una fuente de video AVI. Posee un amplio repertorio de funciones que implementan varios de los algoritmos empleados en la visión computacional y algunos algoritmos de aprendizaje automático. Esta biblioteca está desarrollada en lenguaje de programación C y C++ y está disponible para los sistemas operativos Microsoft Windows, GNU/Linux y Apple Mac OS.

Durante este trabajo, los algoritmos fueron implementados con la ayuda de esta biblioteca. Permite trabajar con matrices de diferentes tipos de datos, entre las que se encuentran datos de 8, 32 y 64 bits. Las imágenes son representadas mediante una estructura de C, que contienen entre sus elementos, la profundidad de la imagen, la cantidad de filas y columnas y el valor de los pixeles, los cuales siempre están almacenados en un arreglo sin importar si se trata de una imagen en escala de grises o RGB. Para el caso de una imagen RGB, tres elementos consecutivos del arreglo corresponden a un solo pixel, por lo que la imagen debe ser recorrida con saltos de tres posiciones por pixel.

OpenCV posee funciones de filtrado de imágenes y todos los filtros empleados en este proyecto se realizaron mediante las implementaciones disponibles en esta biblioteca. El resto de los algoritmos se desarrollaron manipulando las matrices e imágenes con las funciones de OpenCV. A pesar de que OpenCV implementa AdaBoost junto con otros algoritmos, se decidió realizar implementaciones propias de ellas para tener mayor control en estos procesos y en sus parámetros.

5. Experimentos y Resultados

En este capítulo se describirán tanto las pruebas realizadas como los resultados obtenidos de las mismas. Se realizaron comparaciones del desempeño de los tres clasificadores descritos en capítulos anteriores junto con tres diferentes combinaciones de características. Para definir las tres características a emplear se midió el desempeño del clasificador de máquinas de vectores de apoyo con 11 combinaciones diferentes de vectores de características. Posteriormente, estos clasificadores junto con los descriptores son comparados con los métodos de segmentación basados en Naive Bayes, Clasificación Gaussiana y la combinación de este último con la diferencia de canales.

5.1. Validación Cruzada de k Iteraciones

La validación cruzada es un método estadístico empleado para la evaluación y comparación de clasificadores. Este método divide el conjunto de datos en varios subconjuntos de entrenamiento y de prueba, posteriormente cada uno de estos subconjuntos se emplean para evaluar el clasificador.

Una de las formas básicas de validación cruzada es la validación cruzada de k iteraciones. Este método divide el conjunto de datos en k subconjuntos, cada uno con aproxima-

damente el mismo número de elementos. Los datos son asignados a cada subconjunto de forma aleatoria. Para realizar la validación, se utilizan $k-1$ subconjuntos para realizar el entrenamiento del clasificador y, posteriormente, el subconjunto restante es utilizado para validar el modelo. El proceso es repetido k veces usando un conjunto de validación diferente en cada caso, por lo que cada subconjunto es usado como conjunto de prueba durante las diferentes iteraciones de la validación cruzada. Durante cada iteración es posible obtener una medida de validación para evaluar el desempeño a lo largo de las etapas de validación. Como medida adicional, es posible realizar múltiples episodios de validación cruzada de k iteraciones, organizando los datos de diferentes formas en los k subconjuntos de cada episodio.

Para evaluar el desempeño de los algoritmos se empleó el algoritmo de validación cruzada de 5 iteraciones, es decir, se obtuvieron 5 subconjuntos del total de imágenes disponibles [27].

5.2. Sensibilidad y Especificidad

En cada iteración de la validación cruzada se calcularon dos valores de referencia: la sensibilidad y la especificidad [36], para verificar el desempeño del clasificador binario.

La sensibilidad se define como la probabilidad de que el clasificador considere un ejemplo como positivo dado que el ejemplo es realmente positivo. Este valor está definido por:

$$sensibilidad = \frac{VP}{VP + FN} \tag{5.1}$$

donde VP representa a los verdaderos positivos, es decir, los ejemplos clasificados como positivos, siendo realmente positivos; FN representa a los ejemplos clasificados como negativos a pesar de ser positivos, lo cual es nombrado como falsos negativos.

La especificidad, por otra parte, se refiere a la probabilidad de que un ejemplo sea considerado como negativo, dado que es realmente negativo. La especificidad es calculada mediante:

$$especificidad = \frac{VN}{VN + FP} \tag{5.2}$$

en este caso, VN, los verdaderos negativos, son ejemplos clasificados como negativos siendo realmente negativos, y los falsos positivos FP, son ejemplos negativos erróneamente clasificados como positivos [36].

5.3. Evaluación de los Algoritmos

Para entrenar los algoritmos se empleó un conjunto de imágenes de *Trypanosoma cruzi*, segmentadas de forma manual. En total se generaron 900 imágenes con parásito. Similarmente, se obtuvo un conjunto de imágenes de muestras sanguíneas sin parásito, empleados como ejemplos negativos para el entrenamiento. Las imágenes empleadas para el entrenamiento se muestran en la Fig. 5.1.

Figura 5.1: Arriba: imágenes positivas junto a su segmentación manual. Abajo: imágenes negativas.

En la Fig. 5.1 puede observarse una imagen binaria correspondiente a la segmentación manual. El principal objetivo de esta imagen es el de distinguir los súper pixeles que definen al parásito del resto de los pixeles de la imagen, de forma que en el entrenamiento se empleen las muestras adecuadas.

Se realizaron pruebas con 11 vectores de características formados por diferentes combinaciones de las características descritas en la sección 4.2, así como por los espacios de color RGB, YCbCr (también conocido como YCC) y HSV. Para identificar las características empleadas dentro de cada vector consideraremos la siguiente simbología:

- Aspereza (a).
- Contraste (cn).
- Direccionalidad (d).
- Curtosis (ct).
- Media (m)
- Varianza (v).

Para indicar los canales de color para los cuales se calculan estas características, indicaremos las iniciales del mismo: YCC, RGB y HSV. En caso de incluir la diferencia de los canales azul y verde en el espacio RGB, emplearemos el símbolo dif. Con base a lo anterior, el código

$$(a, cn, d)_{RGB,Dif}$$

se interpreta como el vector de características formado por la aspereza, el contraste y la direccionalidad calculados para cada canal R, G, B y Dif. En este caso, se tienen cuatro canales y tres características por canal, formando un vector de 12 elementos.

Para definir los tres vectores de descriptores a emplear, se realizaron pruebas preliminares con el algoritmo de clasificación máquinas de vectores de apoyo (SVM, por sus siglas en inglés). Para esto, se utilizó la implementación disponible en la biblioteca LIBSVM [8], empleando el algoritmo validación cruzada disponible en la misma. Para cada imagen de entrenamiento, se aplicó el proceso de extracción de características basado en súper pixeles. Se tomaron 2000 súper pixeles para evaluar estos vectores. Los resultados se muestran en la tabla 5.1.

Tabla 5.1: Desempeño de los vectores de características empleando SVM como clasificador.

Vector	No. de Características	Exactitud
$(as, cn, d)_{RGB}$	9	76.75
$(as, cn)_{RGB,dif}$	8	77.25
$(as, cn, d, ct, m, v)_{RGB,dif}$	24	88.8
$(as, cn, d, ct, m, v)_{YCC}$	18	89.65
$(as, cn, d, ct, m, v)_{HSV}$	18	86.45
$(as, cn, d, ct, m, v)_{RGB,dif,YCC}$	42	90.75
$(m, v)_{RGB}$	6	88.25
$(ct, m, v)_{RGB,dif}$	12	90.75
$(ct, m, v)_{YCC}$	9	88.9
$(ct, m, v)_{RGB,dif,YCC}$	21	89.9
$(ct, m, v)_{HSV}$	9	86.1

La exactitud se define como el total de elementos correctamente clasificados dividido entre el total de elementos empleados para la prueba. Durante la validación cruzada, se empleó el kernel función base radial y se realizó una búsqueda de parámetros, observando el desempeño del clasificador con durante las etapas de la validación cruzada y variando los parámetros del clasificador. Los resultados mostrados en la tabla corresponden a los

mejores parámetros encontrados para cada vector. Para el caso del kernel función base radial, los parámetros están definidos por el parámetro ρ y la constante de regularización C (ver sección 2.3.4).

Como puede observarse en la tabla 5.1, los tres vectores con mejor desempeño son $(as, cn, d, ct, m, v)_{RGB,dif,YCC}$, compuesto por 42 características y con parámetros $c = 91.773135873$ y $\rho = 0.00003009743$; el vector $(ct, m, v)_{RGB,dif}$, con 12 elementos y parámetros $c = 26.9086852881$ y $\rho = 0.00002478797$; finalmente, el vector $(ct, m, v)_{RGB,dif,YCC}$, de 21 elementos y con parámetros $c = 29.0406129701$ y $\rho = 0.00004530406$.

Una vez definidos estos elementos, se empleó una implementación propia del proceso de validación cruzada de 5 iteraciones, empleando el mismo conjunto de entrenamiento, para evaluar el desempeño de los clasificadores basados en redes neuronales, entrenados con el algoritmo de retropropagación y el ensamble de perceptrón entrenado con el algoritmo AdaBoost.

Tabla 5.2: Desempeño de la red neuronal entrenada con el algoritmo de retropropagación como clasificador.

Vector	Tamaño	Exactitud	Sensibilidad	Especificidad
$(as, cn, d, ct, m, v)_{RGB,dif,YCC}$	42	90.15	91.89	88.29
$(ct, m, v)_{RGB,dif}$	12	83.8	87.28	80.41
$(ct, m, v)_{RGB,dif,YCC}$	21	87.75	88.75	86.72

Al evaluar el desempeño de la red neuronal de retropropagación, se realizaron varios ensayos de validación cruzada cada una con diferentes topologías de red y con diferentes constantes de aprendizaje. En cada iteración de la validación cruzada, se calculó la exactitud, la sensibilidad y la especificidad. Al finalizar las 5 iteraciones, en cada ocasión, se promediaron las medidas de desempeño obtenidas y se tomó este valor como referencia

para la comparación. Los resultados de evaluar la red neuronal entrenada con el algoritmo de retropropagación se presentan en la tabla 5.2. La tabla presenta los promedios obtenidos. El mejor desempeño se obtuvo con el vector de 42 características, empleando una constante de aprendizaje de 0.0001 y una topología de una capa oculta con 28 neuronas.

El mismo proceso se realizó para evaluar el ensamble de clasificadores basado en perceptrón. Los parámetros que fueron refinados fueron la constate de aprendizaje y el número de neuronas del ensamble. Los resultados se resumen en la tabla 5.3. El mejor desempeño fue obtenido empleando el vector de 42 elementos con una constante de aprendizaje de 0.004 y un total de 32 neuronas en el ensamble.

Tabla 5.3: Desempeño del ensamble de perceptrón entrenado con AdaBoost.

Vector	Tamaño	Exactitud	Sensibilidad	Especificidad
$(as, cn, d, ct, m, v)_{RGB,dif,YCC}$	42	83.4	84.61	82.22
$(ct, m, v)_{RGB,dif}$	12	81.25	80.67	82.09
$(ct, m, v)_{RGB,dif,YCC}$	21	80.2	85.31	75.55

Comparando la exactitud de los clasificadores, se decidió emplear SVM como clasificador para realizar las pruebas y comparaciones con los algoritmos de segmentación del estado del arte. Para decidir el vector de características a emplear, se realizó la evaluación de los vectores mediante el algoritmo de validación cruzada de cinco iteraciones sobre un conjunto de 2000 súper pixeles, empleando una implementación propia en lugar de la versión incluida en LIBSVM. Esta implementación es la misma que se empleó al momento de realizar las pruebas con AdaBoost y la red neuronal. El rendimiento fue evaluado para los dos vectores con mejor exactitud obtenidos en las pruebas iniciales. Los resultados se presentan en la tabla 5.4, donde se puede observar que el vector con mejor desempeño

es el de 42 características, de la misma forma, la red neuronal parece presentar mejor desempeño que SVM al ser probado bajo las mismas condiciones.

Tabla 5.4: Desempeño del clasificador SVM, empleando una versión propia de validación cruzada.

Vector	Tamaño	Exactitud	Sensibilidad	Especificidad
$(as, cn, d, ct, m, v)_{RGB,dif,YCC}$	42	88.2	87.05	88.91
$(ct, m, v)_{RGB,dif}$	12	86.0	85.4	86.41

Con base en la tabla 5.4 y 5.2, los experimentos posteriores se realizaron con el vector de 42 características. Una vez seleccionado el vector de características, se realizó el entrenamiento de tres clasificadores SVM empleando este vector y tres diferentes tamaños de súper pixel. Los tamaños empleados fueron 50, 100 y 150 pixeles por cada súper pixel. Lo anterior con el objetivo de evaluar el desempeño del clasificador con base en diferentes tamaños de súper pixel. Como se verá a continuación, un tamaño de 150 pixeles por súper pixel ofrece mejores resultados visuales, a pesar de que, en promedio, un tamaño de 100 pixeles por súper pixel presenta menor error. Con base en lo anterior y en los resultados de las tablas 5.4 y 5.2, se decidió incluir a la red neuronal en las comparaciones, realizando el entrenamiento con el mismo conjunto que SVM y empleando súper pixeles de tamaño 150. Para realizar el entrenamiento de los clasificadores SVM y retropropagación, se emplearon $17,000$ súper pixeles positivos y $17,000$ negativos para un tamaño de 150 pixeles por súper pixel. Los súper pixeles para realizar el entrenamiento fueron obtenidos con base en 800 imágenes del total de 900 disponibles con parásito.

Para comparar los algoritmos de segmentación, se emplearon 10 imágenes de prueba las cuales fueron sometidas a los diferentes procesos de segmentación. Estas 10 imágenes fueron elegidas de forma aleatoria del conjunto de 100 imágenes que no fueron empleadas para generar los súper pixeles de entrenamiento, por lo que estás imágenes de prueba no fueron

observadas por los clasificadores durante el entrenamiento. Para evaluar el desempeño, se compararon los resultados obtenidos por cada clasificador, contra la segmentación manual realizada, empleando para esta labor, el error cuadrático medio como medida de exactitud. La tabla 5.5, muestra los resultados de esta comparación. En esta tabla, BP hace referencia a la red neuronal entrenada con el algoritmo de retropropagación, SP50, SP100 y SP150 se refieren a los clasificadores SVM empleando un tamaño de súper pixel de 50, 100 y 150, respectivamente, Gauss se refiere al clasificador Gaussiano, Bayes hace referencia a la segmentación Bayesiana y Alg2012, se refiere al algoritmo propuesto en [34].

Tabla 5.5: Comparación de los resultados de la segmentación, basado en el error cuadrático medio.

Muestra	BP	SP50	SP100	SP150	Gauss	Bayes	Alg2012
T1	0.345358	0.334952	0.286507	0.344317	0.26847	0.330443	0.434617
T2	0.501434	0.219093	0.20172	0.251982	0.160735	0.222297	0.462979
T3	0.335502	0.204844	0.221038	0.223668	0.185052	0.220346	0.169965
T4	0.425794	0.30262	0.261683	0.265427	0.168579	0.281246	0.143702
T5	0.270535	0.237645	0.218081	0.203238	0.139147	0.27998	0.562995
T6	0.386961	0.288897	0.216581	0.211286	0.205077	0.246348	0.205077
T7	0.325937	0.273438	0.279062	0.247031	0.2075	0.257969	0.208281
T8	0.224206	0.178005	0.153203	0.168509	0.103316	0.337302	0.492489
T9	0.4221	0.28486	0.276923	0.264957	0.330281	0.376068	0.519658
T10	0.372671	0.221739	0.148654	0.181781	0.0886128	0.224845	0.59234
Promedio	0.361	0.2546	0.22635	0.23622	0.18568	0.2777	0.3792

Los resultados descritos en la tabla 5.5 se presentan de manera gráfica en la Fig. 5.2. En esta figura se presentan los resultados de cada clasificador, agrupados respecto a las imágenes de pruebas. Se presenta el error cuadrático obtenido mediante una gráfica de

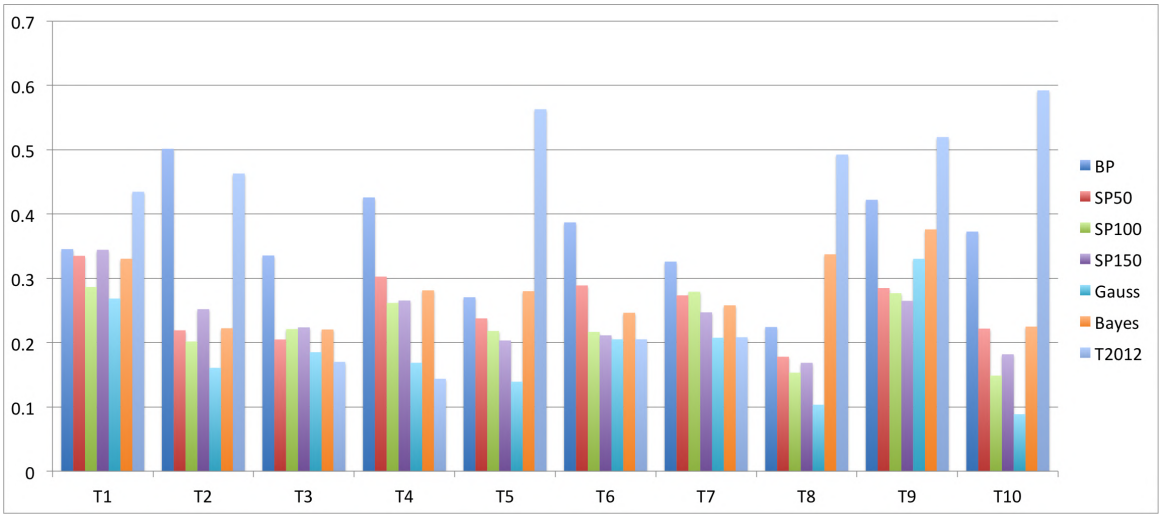

Figura 5.2: Comparación gráfica del desempeño de los clasificadores en un conjunto de 10 imágenes. En el eje de las x tenemos el identificador de la imagen y en el eje y tenemos el error cuadrático medio obtenido. En este sentido, los clasificadores con mejor desempeño presentan barras más cortas en la gráfica.

barras. Con base en los resultados, se puede observar que el clasificador Gaussiano presenta el menor error respecto a la segmentación manual. Seguido de este se encuentran los métodos de segmentación basados en súper pixeles con SVM. Dentro de este grupo es posible observar que los tamaños de súper pixel que presentan menor error son 100 y 150, respectivamente. Los resultados obtenidos con estos algoritmos pueden observarse en las Fig. 5.3 a 5.12, las cuales muestran las 10 imágenes $(T1, \ldots, T10)$ de prueba. En estas figuras se presentan las imágenes de muestra a color junto con su segmentación manual, posteriormente, se presentan los resultados obtenidos con cada uno de los clasificadores. Con base en estas imágenes se puede observar que, en algunas ocasiones, el clasificador propuesto en [34] no logra segmentar ningún elemento. Esto es debido a la forma de operar de este algoritmo. Debido a que combina los resultados de la segmentación Gaussiana y de la diferencia de canales, la segmentación se realizará con éxito si la diferencia de canales

logra segmentar las estructuras celulares de los parásitos. El resultado se obtiene considerando los elementos del clasificador Gaussiano que contienen alguna de las estructuras segmentadas por la diferencia de canales, por lo que si la diferencia no logra segmentar las estructuras, ningún elemento resultante de la segmentación Gaussiana será considerada. Por otra parte, la segmentación emplea el área para discriminar los elementos que son demasiado grandes o demasiado pequeños. Esto hace que la segmentación sea sensible al tamaño de la imagen, evitando que algunos elementos sean segmentados de forma correcta.

La red neuronal presenta algunos problemas al momento de segmentar la imagen, sin embargo, había presentado un mejor desempeño en las etapas de entrenamiento. Es posible que esto se deba a que las imágenes empleadas para estas pruebas fueron obtenidas a partir de un conjunto diferente al empleado durante el entrenamiento, por lo que la red neuronal estaría presentando problemas en su capacidad predictiva ante nuevos elementos para este problema.

Posteriormente, se procedió a realizar la segmentación de una imagen compleja, la cual contiene al parásito rodeado de plasma y células sanguíneas. Al igual que las imágenes anteriores, estas nuevas imágenes fueron segmentadas empleando los clasificadores presentados en la tabla 5.5, a excepción de la red neuronal, debido a que en las imágenes $T1, T2, \ldots, T10$ no presentó resultados visuales adecuados. No se dispone de una segmentación manual para estas imágenes. Algunos resultados se presentan en las Fig. 5.13 y Fig. 5.14. Respecto a la segmentación basada en súper pixeles, una apreciación visual muestra un mejor desempeño empleando un número de 150 pixeles por súper pixel. Comparado con el resto de los algoritmos, se observa similitud entre los resultados de la segmentación basada en súper pixeles y la segmentación Gaussiana.

Comparando la segmentación por súper pixeles y la segmentación Gaussiana, se puede observar que el uso de súper pixeles logra separar la estructura del parásito del resto de la

imagen, lo cual permitiría discriminar los elementos segmentados del resto de la imagen mediante una segunda etapa que refine la segmentación, o bien, mediante el reconocimiento sobre los elementos segmentados. Como ejemplo de lo anterior las Figs. 5.15 y 5.16 presentan los resultados de aplicar una umbralización de área de los objetos segmentados por el clasificador SVM con súper pixeles de tamaño 150 y por el clasificador Gaussiano. Con esto se eliminan los objetos que son más grandes que cierto valor y los objetos más pequeños que un segundo umbral. Es pertinente mencionar que este proceso no es automático. Los umbrales fueron definidos de manera manual, probando con diferentes valores hasta llegar al resultado deseado para cada imagen segmentada. Los resultados de la Fig. 5.15 son similares, sin embargo, los resultados presentados en la Fig. 5.16 muestran que el algoritmo basado en súper pixeles logró aislar al parásito del resto de la imagen, lo que permitió un mejor resultado al momento de discriminar los objetos por área. Sin embargo, a pesar de que este aislamiento es frecuente, no ocurre en todos los casos. El uso de umbrales de área fijos no es adecuado, debido a que el área del parásito segmentado puede variar dependiendo del tamaño de la imagen y pueden ser empleados clasificadores que operen sobre los diferentes objetos segmentados, por lo que trabajos futuros pueden contemplar el uso de umbrales de área que se ajusten según el tamaño de la imagen.

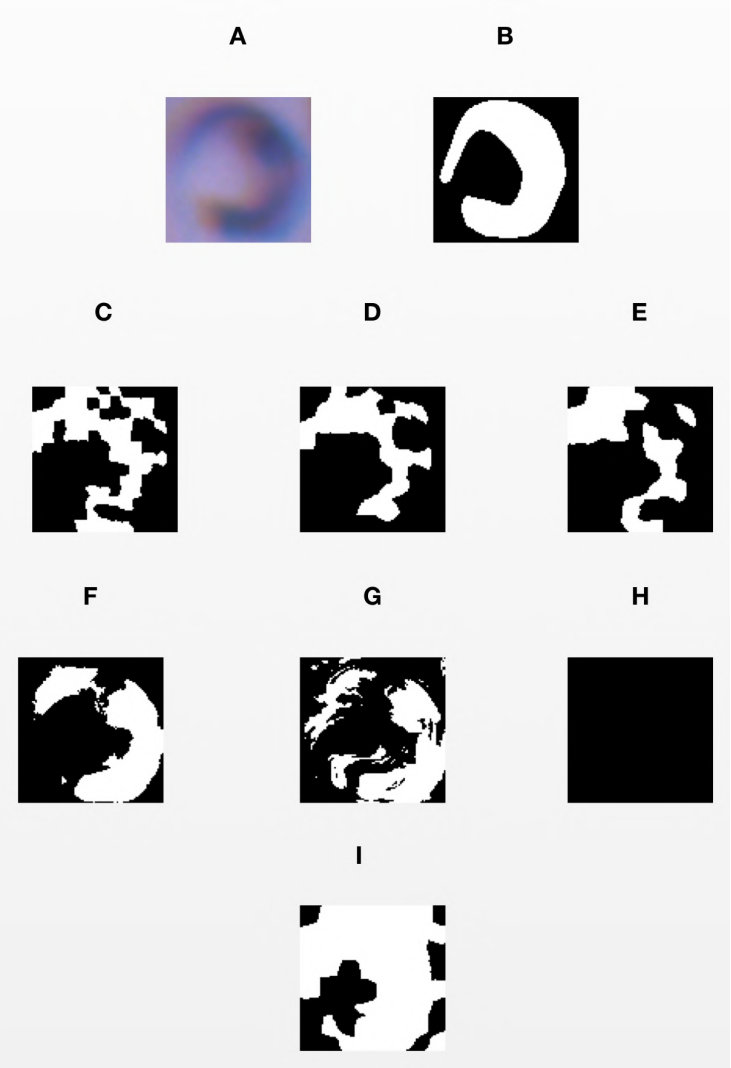

Figura 5.3: Ejemplo T1: A y B son la imagen original y la segmentación manual. C, D y E, son los resultados de SVM con tamaños de 50, 100 y 150 para los súper pixeles. F, G, H e I, son los resultados de la clasificación Gaussiana, Bayesiana, el algoritmo propuesto en [34] y la clasificación empleando redes neuronales, respectivamente.

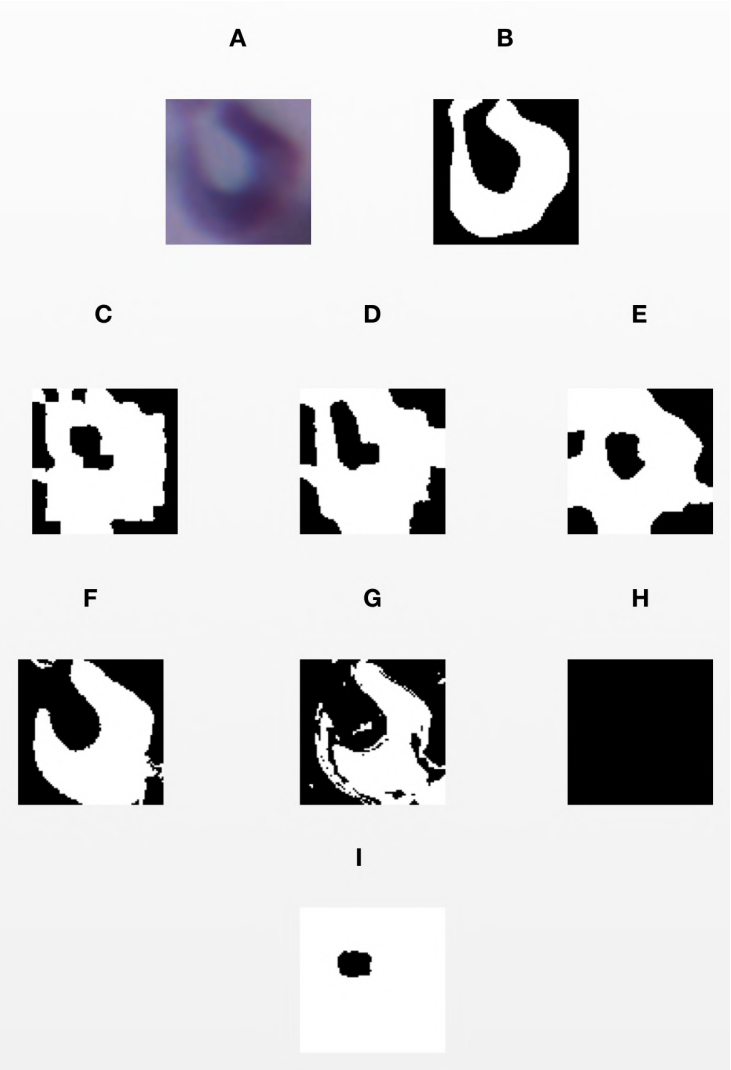

Figura 5.4: Ejemplo T2: A y B son la imagen original y la segmentación manual. C, D y E, son los resultados de SVM con tamaños de 50, 100 y 150 para los súper pixeles. F, G y H e I, son los resultados de la clasificación Gaussiana, Bayesiana, el algoritmo propuesto en [34] y la clasificación empleando redes neuronales, respectivamente.

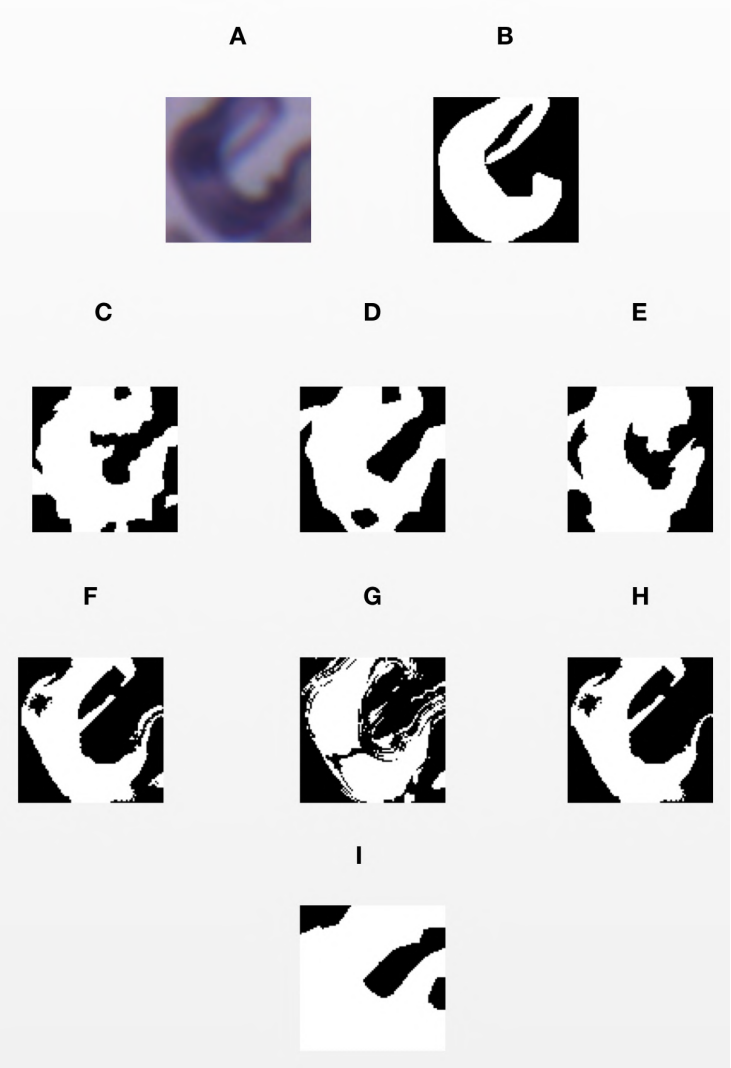

Figura 5.5: Ejemplo T3: A y B son la imagen original y la segmentación manual. C, D y E, son los resultados de SVM con tamaños de 50, 100 y 150 para los súper pixeles. F, G y H e I, son los resultados de la clasificación Gaussiana, Bayesiana, el algoritmo propuesto en [34] y la clasificación empleando redes neuronales, respectivamente.

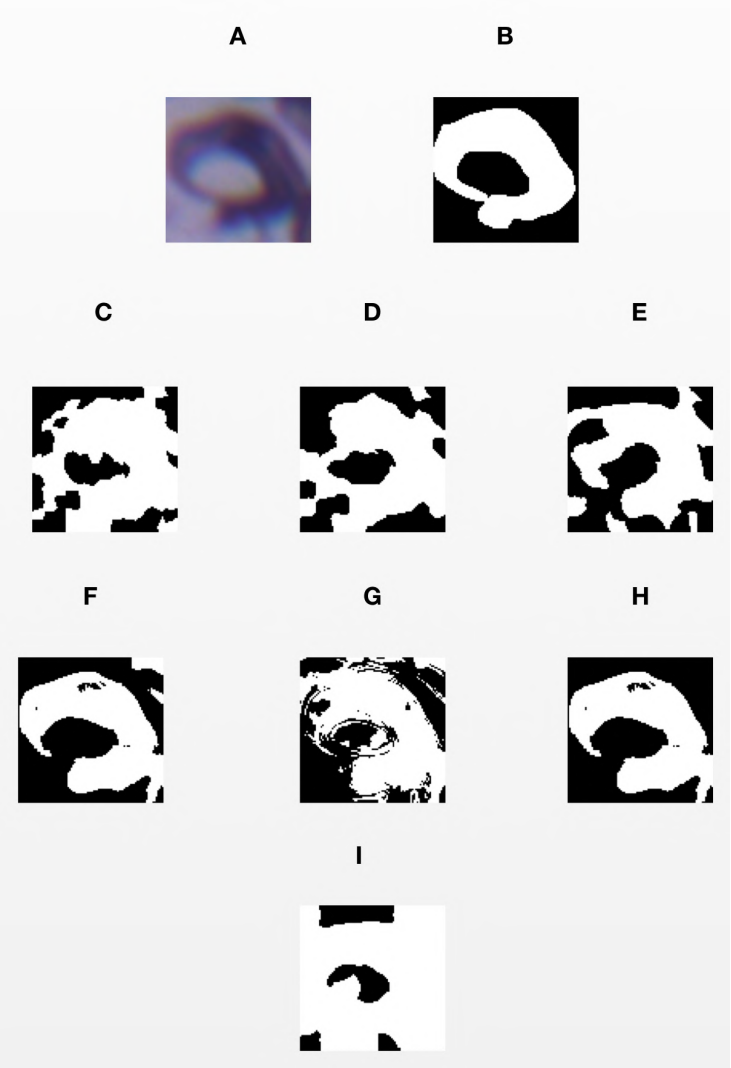

Figura 5.6: Ejemplo T4: A y B son la imagen original y la segmentación manual. C, D y E, son los resultados de SVM con tamaños de 50, 100 y 150 para los súper pixeles. F, G y H e I, son los resultados de la clasificación Gaussiana, Bayesiana, el algoritmo propuesto en [34] y la clasificación empleando redes neuronales, respectivamente.

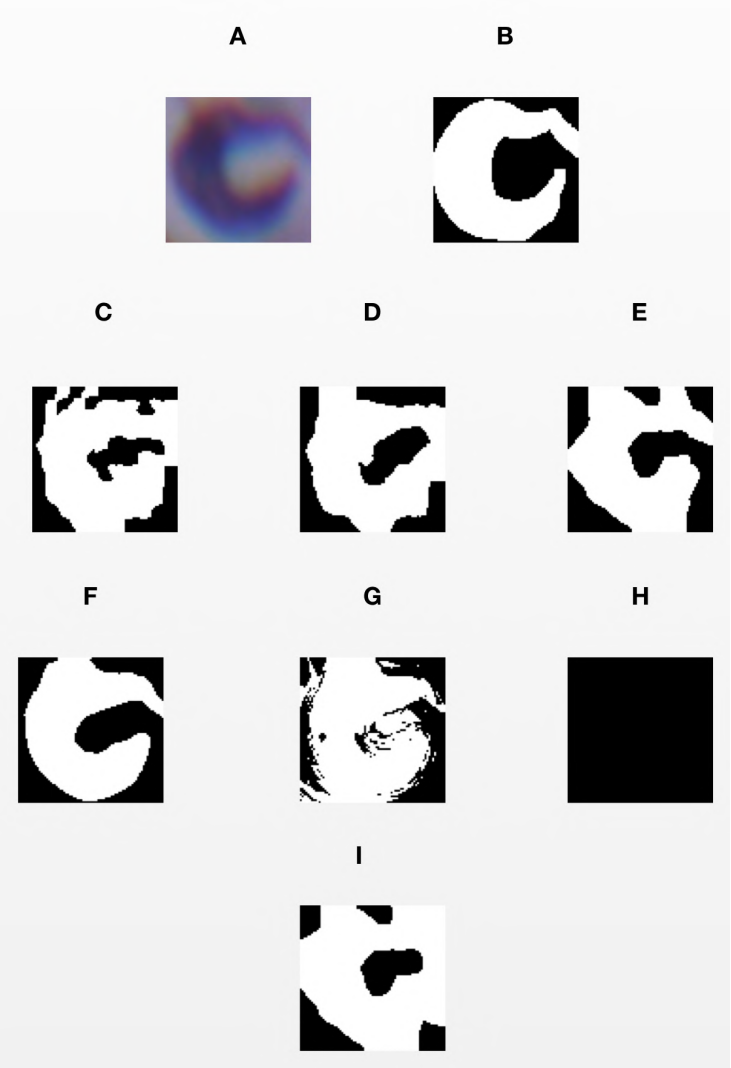

Figura 5.7: Ejemplo T5: A y B son la imagen original y la segmentación manual. C, D y E, son los resultados de SVM con tamaños de 50, 100 y 150 para los súper pixeles. F, G y H e I, son los resultados de la clasificación Gaussiana, Bayesiana, el algoritmo propuesto en [34] y la clasificación empleando redes neuronales, respectivamente.

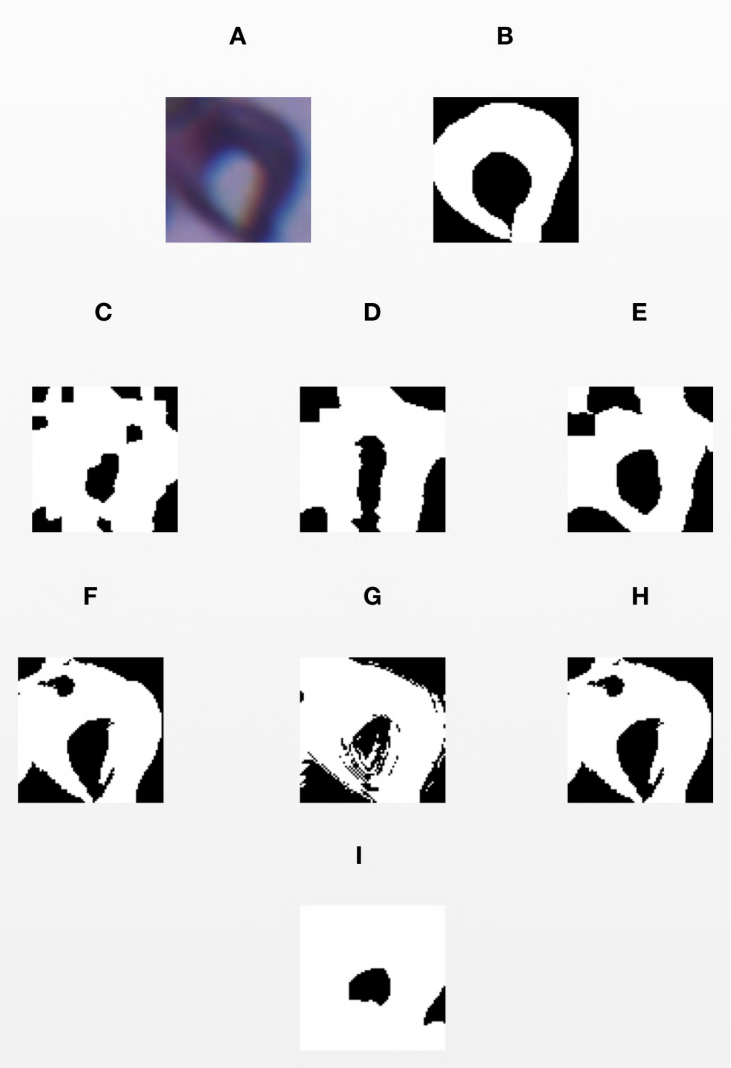

Figura 5.8: Ejemplo T6: A y B son la imagen original y la segmentación manual. C, D y E, son los resultados de SVM con tamaños de 50, 100 y 150 para los súper pixeles. F, G y H e I, son los resultados de la clasificación Gaussiana, Bayesiana, el algoritmo propuesto en [34] y la clasificación empleando redes neuronales, respectivamente.

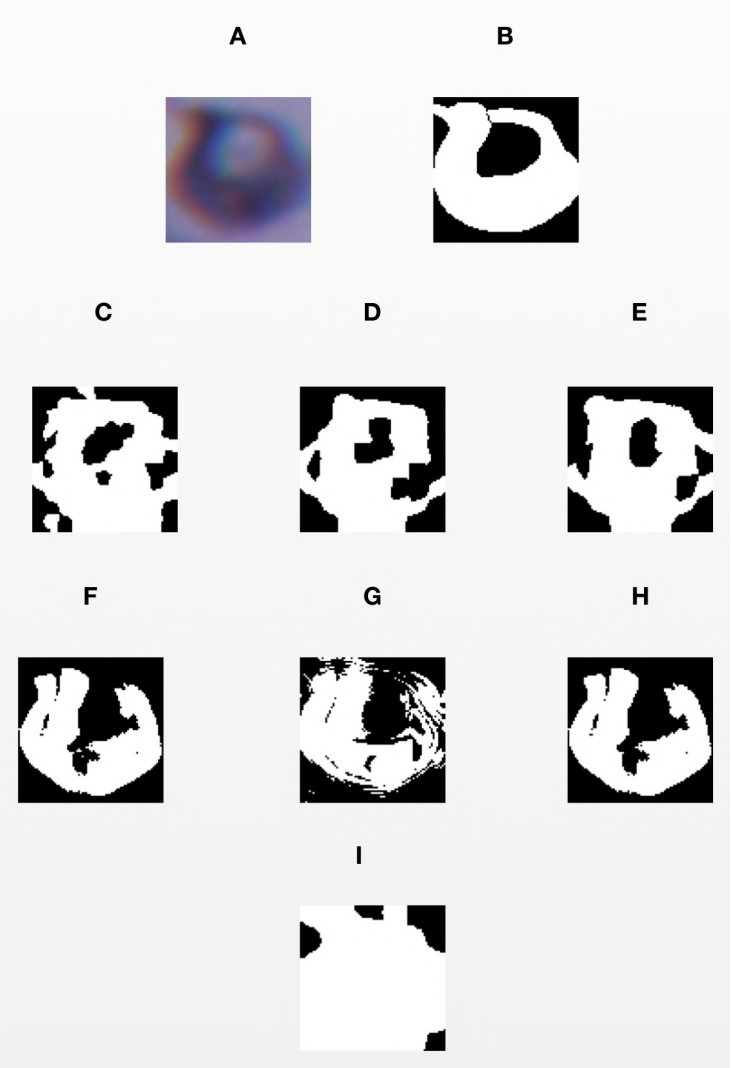

Figura 5.9: Ejemplo T7: A y B son la imagen original y la segmentación manual. C, D y E, son los resultados de SVM con tamaños de 50, 100 y 150 para los súper pixeles. F, G y H e I, son los resultados de la clasificación Gaussiana, Bayesiana, el algoritmo propuesto en [34] y la clasificación empleando redes neuronales, respectivamente.

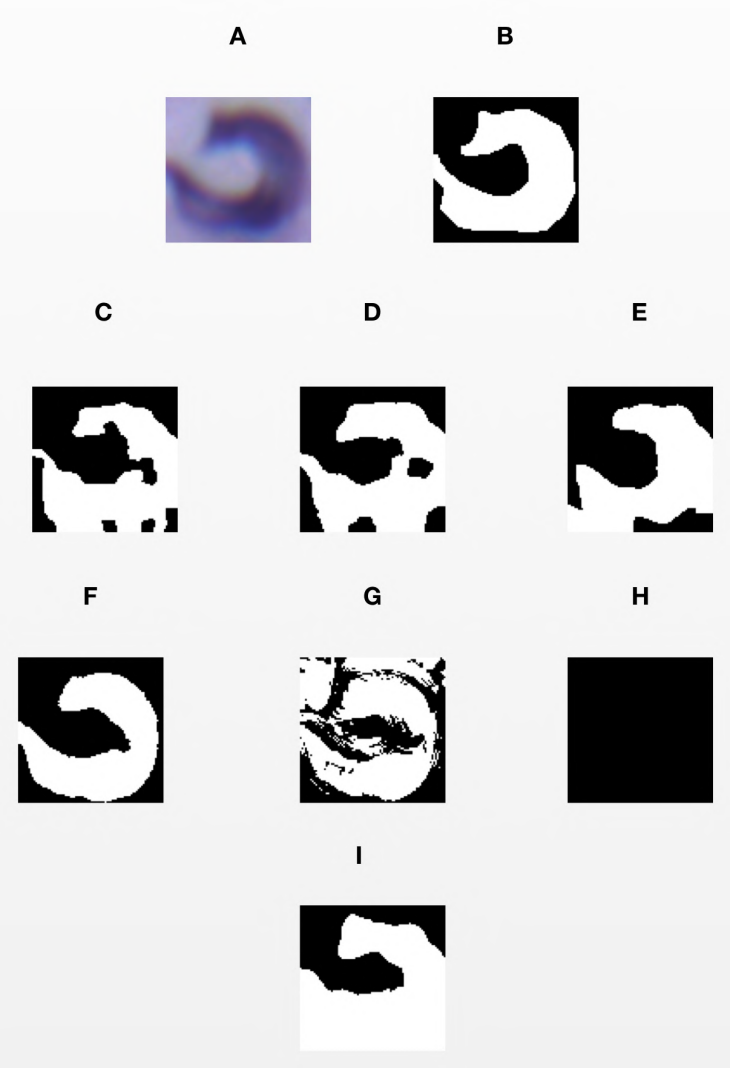

Figura 5.10: Ejemplo T8: A y B son la imagen original y la segmentación manual. C, D y E, son los resultados de SVM con tamaños de 50, 100 y 150 para los súper pixeles. F, G y H e I, son los resultados de la clasificación Gaussiana, Bayesiana, el algoritmo propuesto en [34] y la clasificación empleando redes neuronales, respectivamente.

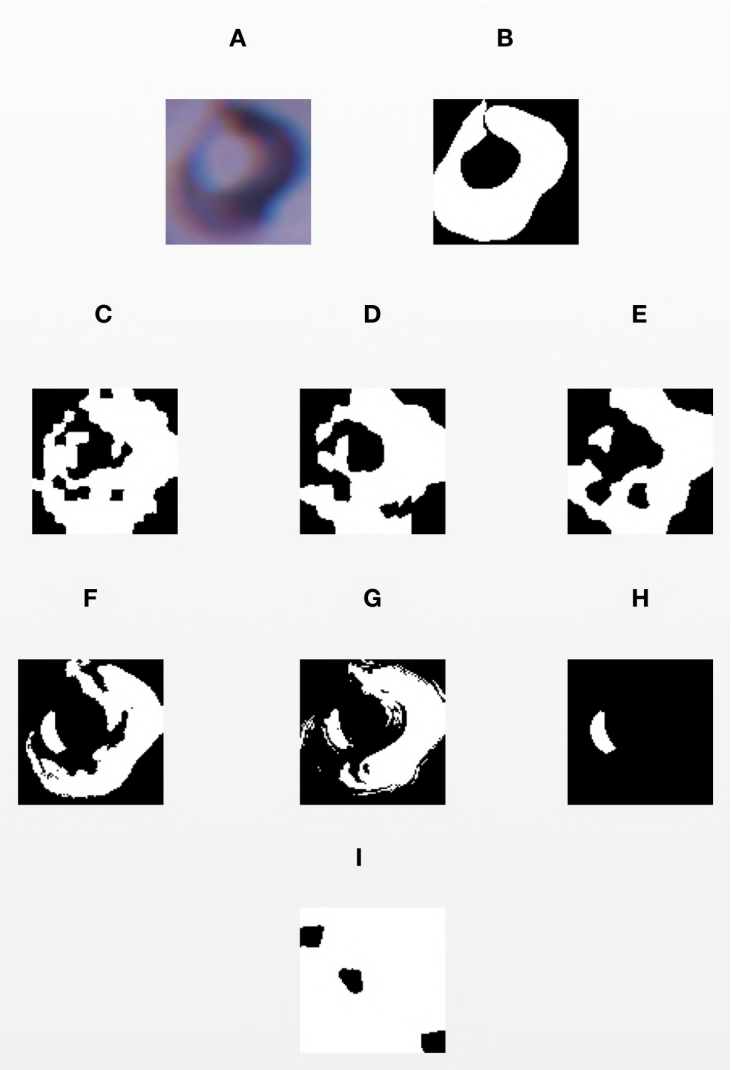

Figura 5.11: Ejemplo T9: A y B son la imagen original y la segmentación manual. C, D y E, son los resultados de SVM con tamaños de 50, 100 y 150 para los súper pixeles. F, G y H e I, son los resultados de la clasificación Gaussiana, Bayesiana, el algoritmo propuesto en [34] y la clasificación empleando redes neuronales, respectivamente.

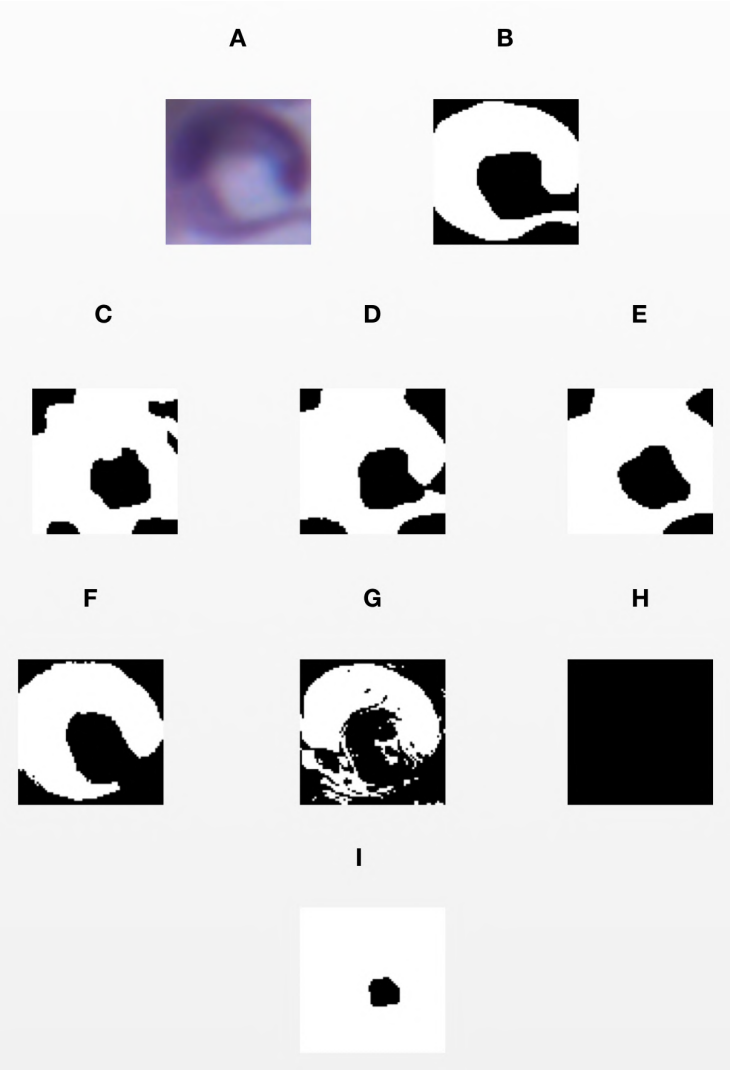

Figura 5.12: Ejemplo T10: A y B son la imagen original y la segmentación manual. C, D y E, son los resultados de SVM con tamaños de 50, 100 y 150 para los súper pixeles. F, G y H e I, son los resultados de la clasificación Gaussiana, Bayesiana, el algoritmo propuesto en [34] y la clasificación empleando redes neuronales, respectivamente.

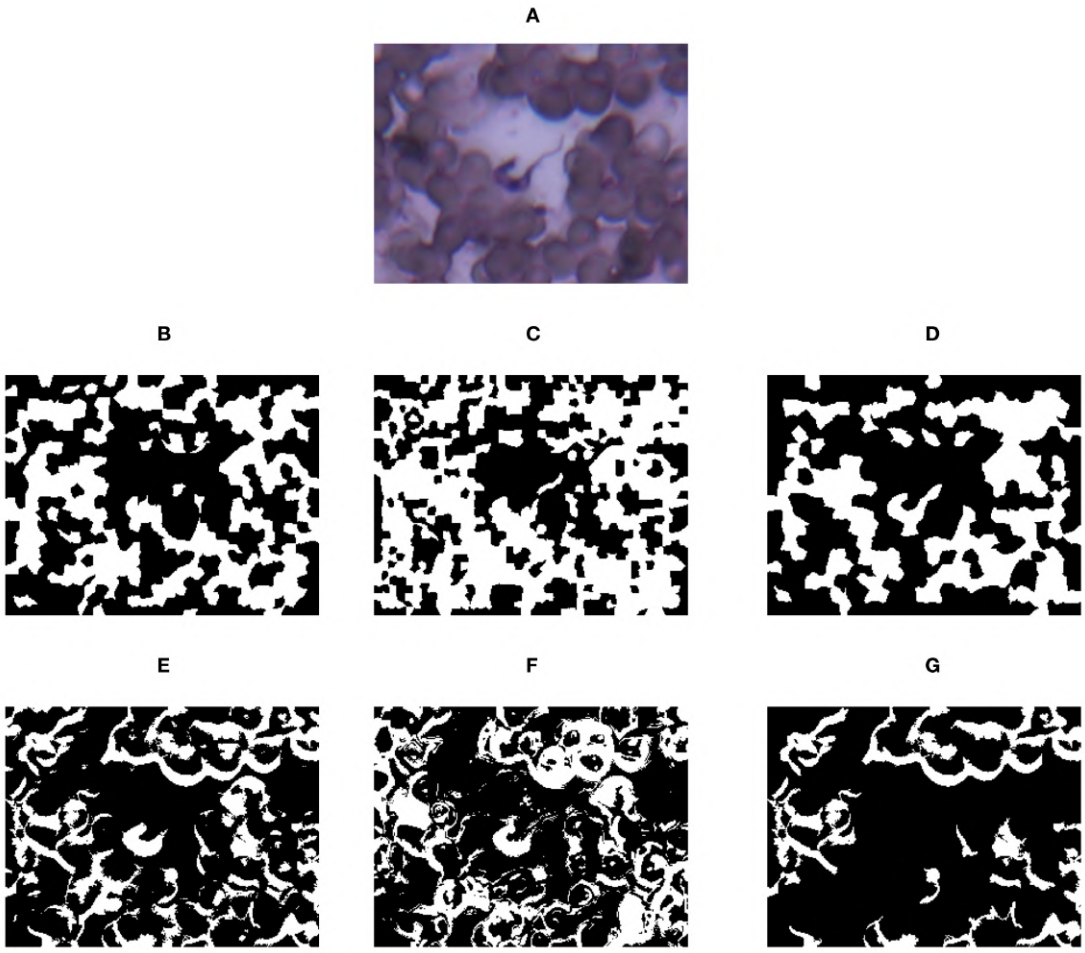

Figura 5.13: Segmentación de una muestra: A es la imagen original. B, C y D, son los resultados de SVM con tamaños de 50, 100 y 150 para los súper pixeles. E, F y G, son los resultados de la clasificación Gaussiana, Bayesiana y el algoritmo propuesto en [34], respectivamente.

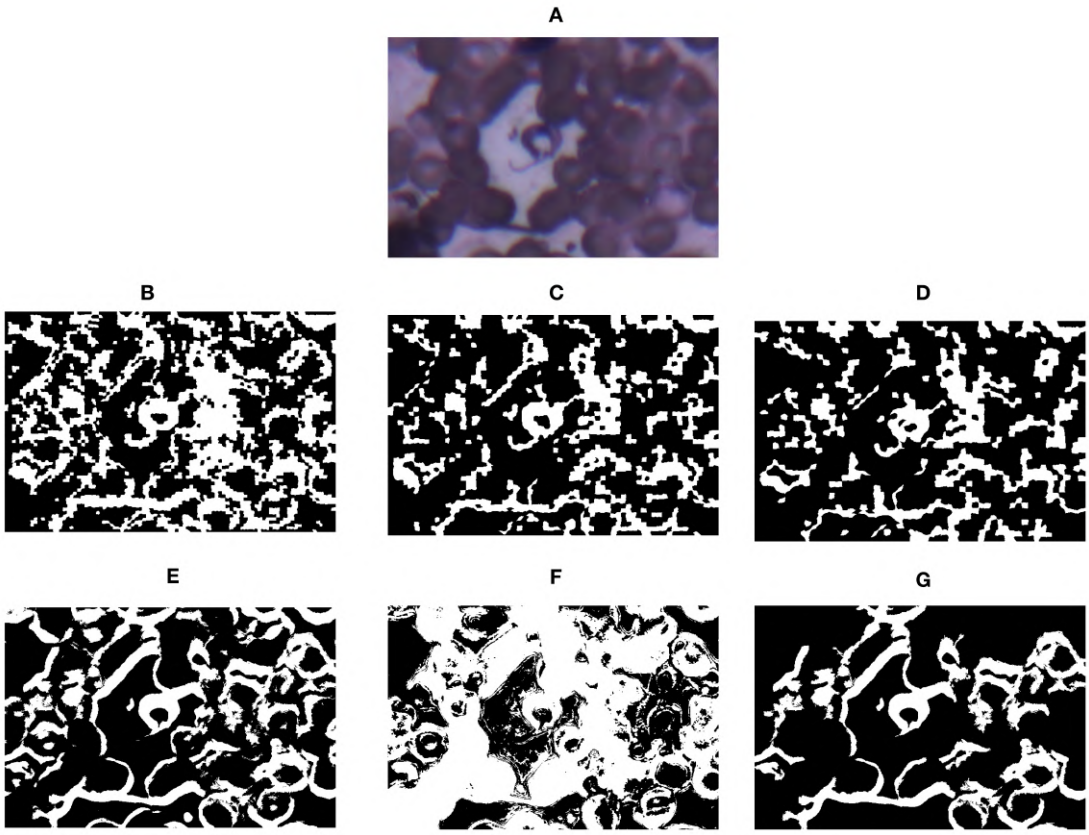

Figura 5.14: Segmentación de una muestra: A es la imagen original. B, C y D, son los resultados de SVM con tamaños de 50, 100 y 150 para los súper pixeles. E, F y G, son los resultados de la clasificación Gaussiana, Bayesiana y el algoritmo propuesto en [34], respectivamente.

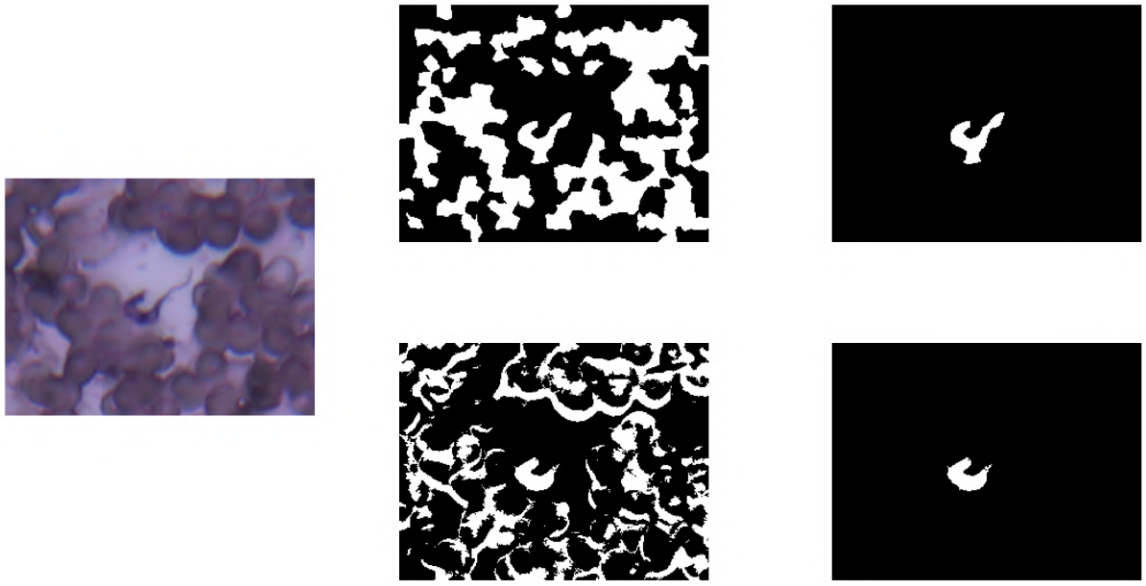

Figura 5.15: Discriminación por área de la Fig. 5.13. Las imágenes superiores fueron obtenidas con base en el clasificador de súper pixeles, empleando un tamaño de 150 pixeles por súper pixel. Las imágenes inferiores corresponden a resultados del clasificador Gaussiano. Los umbrales fueron establecidos de manera manual de acuerdo a la imagen.

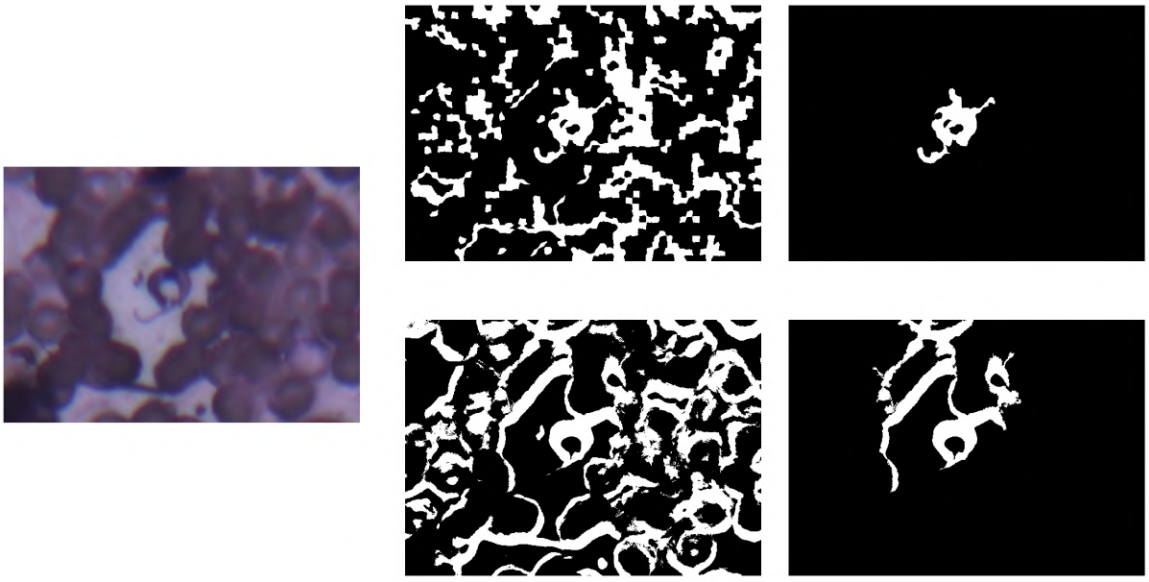

Figura 5.16: Discriminación por área de la Fig. 5.14. Las imágenes superiores fueron obtenidas con base en el clasificador de súper pixeles, empleando un tamaño de 150 pixeles por súper pixel. Las imágenes inferiores corresponden a resultados del clasificador Gaussiano. Los umbrales fueron establecidos de manera manual de acuerdo a la imagen.

6. Conclusiones y Trabajo Futuro

6.1. Conclusiones

El uso de súper pixeles y clasificadores para la segmentación de imágenes de *Trypanosoma cruzi*, permite aislar el objeto de interés del resto de los objetos de la imagen, en algunos casos en los que los clasificadores basados en la teoría Bayesiana presentan dificultades. Aislar el objeto de interés permite un mejor resultado al momento de evaluar cada objeto segmentado dentro de la misma imagen. Si bien este trabajo no incluye un proceso de evaluación, presenta un ejemplo mediante la discriminación manual de área.

Entre las ventajas que se pudieron observar al momento de trabajar con súper pixeles podemos mencionar su capacidad de adaptarse a los contornos de la imagen. Esto permite una diferenciación entre los súper pixeles contenidos por el objeto de interés y aquello que se encuentran externos al borde del mismo. Sin embargo, debido a que un súper pixel se encuentra compuesto por varios pixeles, todos estos pasarán a formar parte del objeto de interés en caso de que el súper pixel sea clasificado como positivo, aunque solo una parte de este súper pixel pertenezca al objeto de interés. Para el caso de segmentación de *Trypanosoma cruzi*, esto provoca un borde distorsionado al momento de realizar la segmentación.

El uso de un grupo de pixeles para representar una muestra, en vez de un pixel individual, permite el cálculo de características complejas, que capturan la relación entre los grupos de pixeles que pertenecen a determinado objeto en la imagen.

Entre las principales desventajas del uso de la segmentación basada en la clasificación de súper pixeles, se encuentran el tiempo de cálculo de los súper pixeles y el tiempo de entrenamiento del clasificador. Sin embargo, una vez entrenado el clasificador, por lo general el proceso de clasificación no es tardado. En este sentido, existe cierta preferencia al clasificador Gaussiano, debido a su buen desempeño y menor tiempo de ejecución.

Respecto a la comparación de los clasificadores empleados en este trabajo, se concluye que el algoritmo SVM presenta el mejor rendimiento, seguido de la red neuronal. El tamaño de los súper pixeles influye en la clasificación final. Un tamaño muy pequeño involucra una mayor cantidad de elementos no deseados en la imagen. Un tamaño mayor ocasiona mayor irregularidad en los bordes de los objetos segmentados. En ambos extremos es posible que el objeto de interés se conecte con otros objetos segmentados que son irrelevantes. Lo anterior dificultaría el proceso de detección o refinamiento por parte de etapas posteriores. Un tamaño adecuado puede ser encontrado mediante el trabajo experimental con imágenes de entrenamiento. En este proyecto se encontró que el proceso de segmentación mejora al emplear un tamaño aproximado de 150 pixeles por súper pixel. Sin embargo, es posible que esto pueda variar dependiendo del problema a tratar y del tamaño de la imagen.

Las contribuciones de este trabajo incluyen un proceso de segmentación basado en la clasificación súper pixeles, el cual puede ser empleado en diversos problemas; El uso de esta metodología como un componente en el proceso de segmentación de *Trypanosoma cruzi* y la comparación de diversos algoritmos de segmentación que pueden emplearse en este problema. Derivado de esto, se tiene un conjunto de imágenes de *Trypanosoma*

cruzi junto a su segmentación manual, que permitirá el desarrollo y evaluación de futuras investigaciones en el área.

6.2. Trabajo Futuro

Siguiendo la linea de segmentación de imágenes basado en súper pixeles, los trabajos a futuro incluyen la evaluación con diferentes características. Considerando la desventaja del tiempo de cálculo de la imagen de súper pixeles, es pertinente el estudio de metodologías de optimización para este proceso, incluyendo el uso de tarjetas gráficas para cómputo en paralelo de estos elementos en los puntos donde sea pertinente. De forma similar, es posible definir el tamaño de los súper pixeles mediante dos formas, la primera es indicando directamente el número de pixeles por súper pixel. La segunda forma es indicando el número de súper pixeles en los que se desea dividir la imagen. Esto puede ser sensible al tamaño de la imagen. Indicar un tamaño fijo de pixeles por súper pixel puede presentar algunos problemas en imágenes muy grandes, debido a la posibilidad de que el número de pixeles elegido no sea significativo dado el tamaño de la imagen. Estudios futuros pueden contemplar metodologías para determinar la mejor forma de elegir el tamaño de los súper pixeles con base en las características de la imagen a evaluar. Lo anterior es de especial importancia, debido a que los estudios de búsqueda y confirmación de presencia del parásito se realizan a diferentes aumentos del microscopio, realizando el primero con un aumento de 40×, mientras que el segundo se realiza a 100×, teniendo como consecuencia la aparición del parásito a diferentes escalas.

En el marco general del proceso de detección de *Trypanosoma cruzi*, futuros trabajos estarían enfocados en las siguientes etapas del proceso, considerando la refinación de la

segmentación mediante la discriminación de los objetos segmentados, empleando métodos de reconocimiento entrenados de acuerdo al objeto de interés.

Bibliografía

[1] Slic-superpixels, 2013. URL https://github.com/PSMM/SLIC-Superpixels.

[2] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274 – 2282, May 2012.

[3] D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[4] G. Bradski and A. Kaehler. *Learning OpenCV*. O' Reilly, 2008.

[5] T. C. Bravo. Trypanosoma cruzi: Historia natural y diagnóstico de la enfermedad de chagas. *Revista Mexicana de Patología Clínica*, pages 205–219, 2004.

[6] *Guía para la atención del paciente infectado con Trypanosoma Cruzi (Enfermedad de Chagas)*. Centro Nacional de Diagnóstico e Investigación de Endemoepidemias, Noviembre 2006.

[7] C. Chagas. Nova tripanozomiaze humana. *Estudos sobre a morfologia e o ciclo evolutivo do Schizotrypanum cruzi n. gen. n. sp., agente etiológico de nova entidade mórbida do homen. Mem*, pages 159–218, 1909.

[8] C.-C. Chang and C.-J. Lin. Libsvm : a library for support vector machines, 2011. URL http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[9] M. C. Colunga, O. S. Siordia, and S. J. Maybank. Leukocyte recognition using em-algorithm. In *MICAI 2009: Advances in Artificial Intelligence*, pages 545–555. Springer Berlin Heidelberg, 2009.

[10] G. Díaz, F. González, and E. Romero. A semi-automatic method for quantification and classification of erythrocytes infected with malaria parasites in microscopic images. *Journal in Biomedical Informatic*, pages 296–307, 2009.

[11] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2003.

[12] G. Gimel'farb. Cbir: Texture features, 2014. URL http://www.cs.auckland.ac.nz/courses/compsci708s1c/lectures/Glect-html/topic4c708FSC.htm.

[13] R. Gonzalez and R. Woods. *Digital Image Processing*. Prentice Hall, 2002.

[14] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh. *Feature Extraction Foundations and Applications*. Springer, 2006.

[15] P. Howarth and S. Rüger. Evaluation of texture features for content-based image retrieval. In *Proceedings of the International Conference on Image and Video Retrieval, Springer-Verlag*, 2004.

[16] P. Howarth and S. Rüger. Robust texture features for still-image retrieval. In *IEE Proceedings Vision, Image and Signal Processing*, 2005.

[17] M. M. Islam, D. Zhang, and G. Lu. A geometric method to compute directionality features for texture images. In *2008 IEEE International Conference on Multimedia and Expo*, pages 1521 – 1524. IEEE, 2008.

[18] Y. Jaganathan and I. Vennila. A hybrid approach based medical image retrieval system using feature optimized classification similarity framewor. *American Journal of Applied Sciences*, 2013.

[19] E. Luján. Un sistema dedicado para detectar trypanosoma cruzi en sangre, 2011.

[20] S. Mandal, A. Kumar, J. Chatterjee, M. M, and A. K. Ray. Segmentation of blood smear images using normalized cuts for detection of malarial parasites. In *Annual IEEE India Conference (INDICON)*, 2010.

[21] D. M. Memeu, K. A. Kaduki, A. C. K. Mjomba, N. S. Muriuki, and L. Gitonga. Detection of plasmodium parasites from images of thin blood smears. *Open Journal of Clinical Diagnostics*, 2013.

[22] T. Mitchell. *Machine Learning.* McGraw-Hil, 1997.

[23] R. S. Mukul, V. U. Cetina, C. B. Loeza, and H. R. Piña. An automatic algorithm for the detection of trypanosoma cruzi parasites in blood sample images. *Computer Methods and Programs in Biomedicine*, 112:633–639, 2013.

[24] S. L. Phung, A. Bouzerdoum, and D. Chai. Skin segmentation using color pixel classification: analysis and comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):148–154, January 2005.

[25] G. Priyankara, O. Seneviratne, R. Silva, W. Soysa, and C. De Silva. An extensible computer vision application for blood cell recognition and analysis. 2006.

[26] Y. Purwar, S. Shah, G. Clarke, A. Almugairi, and A. Muehlenbachs. Automated and unsupervised detection of malarial parasites in microscopic images. *Malaria Journal*, 2011.

[27] P. Refaeilzadeh, L. Tang, and H. Liu. Cross-validation. *Encyclopedia of Database Systems*, 2009.

[28] E. A. G. Romero, L. Alvarez, and M. A. Basombrio. Optical detection of trypanosoma-cruzi in blod samples, for diagnosis purpose. *5th Iberoamerican Meeting on Optics and 8th Latin American Meeting on Optics, Lasers, and Their Applications*, 2004.

[29] G. Romero, A. Monaldi, and E. Alanís. Digital holographic microscopy for detection of trypanosoma cruzi parasites in fresh blood mounts. *Optics Communications*, 2012.

[30] N. Ross, C. Pritchard, D. Rubin, and A. Dusé. Automated image processing method for the diagnosis and classification of malaria on thin blood smears. *Med Biol Eng Comput.*, pages 427–436, 2006.

[31] S. Russell and P. Nowig. *Artificial Intelligene a Modern Approach.* Prentice Hall, 2 edition, 2003.

[32] L. Saphiro and G. Stockman. *Computer Vision.* Prentice Hall, 2000.

[33] A. Smola and S. Vishwanathan. *Introduction to Machine Learning.* Cambridge University Press, 2008.

[34] R. Soberanis. *Detección de Trypanosoma Cruzi en Imágenes Obtenidas a partir de Muestras Sanguíneas.* Tesis de licenciatura, Universidad Autónoma de Yucatán, 2012.

[35] J. Soni and N. Mishra. Advanced image analysis based system for automatic detection of malarial parasite in blood images. In *International Conference on Advanced Computing, Communication and Networks*, pages 129–137, 2011.

[36] S. Spitalnic. Test properties i: Sensitivity, especificity, and predictive values. September 2004.

[37] V. Špringl. Automatic malaria diagnosis through microscopy imaging, 2009.

[38] R. Szeliski. *Computer Vision: Algorithms and Applications.* Springer, September 2010.

[39] F. B. Tek, A. G. Dempster, and I. Kale. Malaria parasite detection in peripheral blood images. In *Proceedings of the British Machine Conference*, 2006.

[40] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION*, 2001.

[41] P. Viola and M. Jones. Robust real-time face detection. *International Journal of Computer Vision*, pages 137–154, 2004.

[42] D. D. Wackerly, W. Mendenhall, and R. L. Scheaffer. *Estadística Matemática con Aplicaciones*. MATH. Thomson, 2002.

[43] B. Wang, H. Bao, S. Yang, and H. Lou. Crowd density estimation based on texture feature extraction. *Journal of Multimedia*, 2013.

[44] WHO. Control of chagas disease: second report of the who expert committee. Technical report, World Health Organization (WHO) Technical Report Series, 2002.

[45] X. Wu and V. Kumar. *The Top Ten Algorithms in Data Mining*. Taylor & Francis Group, 2009.

[46] L. Xiaojuan and C. Cunshe. A novel wastewater bacteria recognition method based on microscopic image analysis. In *Proceedings of the 7th WSEAS International Conference on CIRCUITS, SYSTEMS, ELECTRONICS, CONTROL and SIGNAL PROCESSING*, 2008.