Universidad Autónoma de Yucatán
Facultad de Matemáticas

Licenciatura:

Ingeniería en Computación

Tesis de Licenciatura

Detección de *Trypanosoma Cruzi* en Imágenes Obtenidas a partir de Muestras Sanguíneas

Roger David Soberanis Mukul

Asesores

Dr. Víctor Uc Cetina

Dr. Carlos Brito Loeza

Dr. Hugo Ruiz Piña

Mérida, Yucatán, México

Septiembre 2012

Dedicado a mis padres por su confianza y esfuerzo para sacarnos adelante.

A mis hermanas por su apoyo incondicional.

Agradecimientos

Quiero presentar mi más sincero agradecimiento a los profesores que me asesoraron durante el desarrollo de este proyecto. Al Dr. Víctor Uc, por sus enseñanzas en el campo de la inteligencia artificial, particularmente, en el aprendizaje automático, al Dr. Carlos Brito por su orientación en los temas de procesamiento de imágenes y al Dr. Hugo Ruiz, profesor investigador del centro de investigaciones regionales Dr. Hideyo Noguchi, por la información proporcionada y las imágenes para el desarrollo de este proyecto. Adicionalmente quiero agradecer al Dr. Arturo Espinosa por introducirme a los temas de visión computacional durante el curso de la licenciatura y que, junto con el Dr. Víctor Uc y el Dr. Carlos Brito, influyeron en mi decisión de llevar una formación en el área de sistemas inteligentes, lo cual hizo posible el desarrollo de esta tesis.

Resumen

La enfermedad de Chagas es una enfermedad crónica que afecta, principalmente, a la población latinoamericana. Esta enfermedad, es causada por el protozoario parásito *Trypanosoma cruzi*, el cual pasa su ciclo inicial de desarrollo en el torrente sanguíneo del portador. En este trabajo se presenta un proceso para la detección de *Trypanosoma cruzi* en imágenes obtenidas a partir de muestras sanguíneas tratadas con tinción Wright. Como punto inicial del proceso, se realiza una segmentación para separar las regiones teñidas del resto de la imagen para posteriormente realizar una clasificación binaria sobre los objetos segmentados, con el objetivo de discriminar los resultados de la segmentación como parásito o no parásito. La segmentación se compone de dos etapas, cada una de ellas realiza una segmentación de elementos que serán combinados para obtener el resultado final. Para la etapa de clasificación se realizaron pruebas con los algoritmos de AdaBoost y los k vecinos más cercanos.

Este trabajo es un producto colateral del proyecto de investigación "Diseño de un sistema de adquisición y análisis de imágenes digitales para su aplicación en el diagnóstico microscópico de la Enfermedad de Chagas" CONACYT SALUD-2008-113848, llevado a cabo en el centro de investigaciones regionales Dr. Hideyo Noguchi.

Índice

Lista de figuras

Lista de Tablas

1. Introducción

Los recientes estudios en el área de la inteligencia artificial han permitido el desarrollo de aplicaciones enfocadas al reconocimiento y clasificación de datos obtenidos de diversas fuentes de información, con la intención de crear aplicaciones computacionales capaces de tomar decisiones con base a la información disponible, de forma similar a como lo haría un ser humano.

Las imágenes proporcionan gran cantidad de información al ser humano, gracias al complejo sistema de visión que posee. Varios estudios se han enfocado en la posibilidad de que una computadora digital obtenga información relevante a partir de una imagen del entorno [19]. Con este objetivo se han desarrollado diversos algoritmos para el análisis y tratamiento de imágenes digitales. Entre las aplicaciones que se han investigado se encuentra el agrupamiento de los pixeles de la imagen de acuerdo a algún atributo en común, la reconstrucción tridimensional de escenas y el reconocimiento e identificación de objetos en la imagen.

El aprendizaje automático es un área perteneciente a la inteligencia artificial que se encarga del estudio y el diseño de métodos de aprendizaje para computadoras digitales. Entre los problemas que aborda se encuentra la clasificación de datos. En los problemas de clasificación, se posee un conjunto de datos los cuales pueden pertenecer a un conjunto

de clases. El objetivo de los algoritmos de clasificación es asignar el dato correcto a la clase correcta. La clasificación binaria es un caso especial en el cual sólo existen dos clases posibles. Los métodos de aprendizaje por refuerzo realizan el aprendizaje mediante un conjunto de entrenamiento compuesto por una colección de datos y un conjunto de etiquetas. Cada dato de la colección tiene asignada una única etiqueta que define su clase.

Los métodos de aprendizaje automático en conjunto con la información extraída de la imagen mediante visión computacional, permiten el aprendizaje de patrones en la imagen que posteriormente pueden ser empleados para el reconocimiento de objetos mediante la comparación de los datos de las imágenes con el patrón de datos aprendido.

La inteligencia artificial también presenta aplicaciones en el campo de la medicina. Entre éstas se encuentra el análisis de imágenes médicas como encefalogramas, imágenes de rayos x e imágenes de ultrasonido entre otras [19]. Estas aplicaciones buscan proporcionar herramientas que faciliten el análisis y diagnóstico por parte de los especialistas. Un tema interesante es el estudio de imágenes obtenidas mediante microscopía, es decir, mediante el análisis en microscopio de diversas muestras. Existen enfermedades causadas por microorganismos parásitos presentes en la sangre y una de las formas de diagnóstico es, precisamente, demostrar la presencia del parásito mediante el análisis microscópico.

La enfermedad producida por el parásito intracelular *Plasmodium vivax*, conocida como malaria, es un padecimiento que ha sido objeto de estudio por parte de los investigadores en inteligencia artificial. El género *Plasmodium* está compuesto por una familia de protozoarios parásitos transmitidos por mosquitos. A esta forma de transmisión mediante un agente biológico se le conoce como transmisión por vector. Los estudios presentan resultados de identificación de *Plasmodium* en imágenes obtenidas a partir de muestras sanguíneas [23][21][25]. Los principales procesos propuestos incluyen una etapa de segmen-

tación, la obtención de elementos descriptivos y la clasificación. La segmentación busca clasificar los pixeles de la imagen en regiones de interés y fondo.

En México y Latinoamérica en general está presente una enfermedad crónica degenerativa causada por el protozoario parásito *Trypanosoma cruzi*. Este padecimiento, conocido como Tripanosomiasis Americana o enfermedad de Chagas [4], es una enfermedad transmitida por vector y una vez adquirida presenta dos fases principales. La primera etapa, conocida como etapa aguda, inicia con la entrada de los parásitos en el torrente sanguíneo del portador. La segunda etapa es conocida como fase crónica y se caracteriza por su duración indeterminada. En esta segunda fase el parásito se reproduce en el tejido muscular, pudiendo afectar el corazón, lo que causaría la muerte del portador. Algunos autores identifican una fase intermedia, entre la fase aguda y la fase crónica, conocida como fase indeterminada, la cual se caracteriza por ser completamente asintomática. El diagnóstico durante la fase aguda puede realizarse mediante un análisis sanguíneo, en el cual se realiza una búsqueda del parásito en la muestra. A diferencia de los trabajos realizados con la detección de malaria, existen pocos trabajos referentes a la detección de Chagas en muestras sanguíneas.

Con base a lo anterior, los objetivos de esta tesis se enfocan al diseño o implementación de un clasificador binario que permita determinar la presencia de *Trypanosoma cruzi* en una imagen adquirida a partir de una muestra sanguínea. Las muestras fueron tratadas con tinción Wright [16]. Se busca probar la factibilidad de la detección automática o semiautomática de *Trypanosoma cruzi*, mediante la propuesta e implementación de un clasificador binario.

Como resultado de la investigación se tiene un clasificador divido en dos etapas: la etapa de segmentación y la etapa de discriminación. La primera de las etapas elimina los componentes innecesarios de la imagen y define una zona de búsqueda. La segunda etapa

tiene por objetivo clasificar los elementos segmentados, para determinar si corresponden al parásito. El método de segmentación propuesto se compone a su vez de dos etapas. Cada una de ellas son en sí un método de segmentación. El resultado final está formado por la combinación de los resultados de las dos etapas. En la discriminación se implementaron y compararon dos algoritmos de clasificación de uso común: AdaBoost y ls k vecinos más cercanos.

El documento está organizado de la siguiente manera: el capítulo 2 presenta una introducción a la Tripanosomiasis Americana, seguido de los temas relativos a visión computacional y aprendizaje automático que fueron empleados en el desarrollo del proyecto. También se incluye el estado del arte en el análisis de muestras microscópicas para la detección de microorganismos y conteo de células. El capítulo 3, define el problema de investigación que aborda este trabajo. En el capítulo 4 se describe el proceso propuesto para la detección de *Trypanosoma cruzi*. Las pruebas y resultados obtenidos se exponen en el capítulo 5. Finalmente, el capítulo 6 presenta las conclusiones finales y el trabajo a futuro como parte del seguimiento a este proyecto.

2. Marco Teórico

2.1. Antecedentes

Algunas enfermedades son causadas por agentes biológicos parásitos que entran al cuerpo humano, en algunas ocasiones, mediante un intermediario como los insectos. La enfermedad de Chagas presenta estas características. Antes de presentar los métodos de inteligencia artificial empleados es necesario dar una introducción a esta enfermedad.

2.1.1. Tripanosomiasis Americana o Enfermedad de Chagas

La Trypanosomiasis Americana, también conocida como enfermedad de Chagas, es una enfermedad crónica que afecta a la población de México, Centroamérica y Sudamérica. Es causada por el protozoario parásito *Trypanosoma cruzi* (Fig. 2.1). Este padecimiento, al igual que el agente causante de esta enfermedad, fueron descubiertos por el médico brasileño Carlos Chagas durante el período de 1909 a 1910 [4].

Trypanosoma cruzi es un protozoario con un único flagelo, el cual es empleado para la movilidad. Pertenece a la familia *Trypanosomatidae* y es un protozoario hemoflagelado que habita en la sangre de algunos mamíferos. Entre sus principales características se encuentra

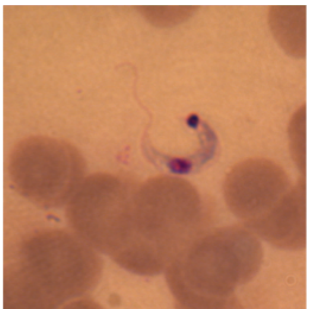

Figura 2.1: *Trypanosoma cruzi* en una muestra sanguínea.

la presencia de un cinetoplasto y un núcleo vesiculoso situado, generalmente, en la parte central del parásito (Fig. 2.2). El cinetoplasto es una estructura que contiene tres espirales de ADN y se encuentra alojada dentro de una expansión capsular de la mitocondria [2].

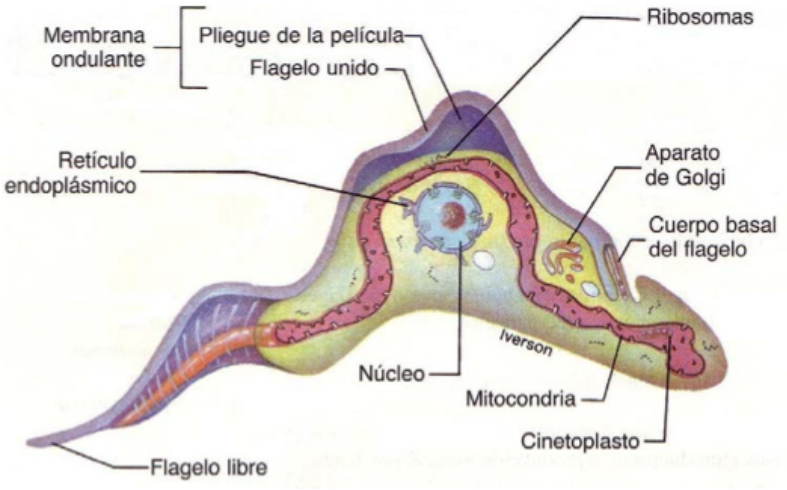

Figura 2.2: Estructura celular de *Trypanosoma cruzi*.

La enfermedad puede ser transmitida al ser humano de distintas maneras. La mas común y el mecanismo natural de transmisión ocurre a través de insectos como *Triatoma infestans* comúnmente conocido como chinche (Fig. 2.3). Este tipo de insectos son hematófagos. Cuando se alimentan de la sangre de una persona o de un animal infectado con

Figura 2.3: *Triatoma infestans*, insecto vector de *Trypanosoma cruzi*.

Trypanosoma cruzi, los parásitos pasan al tubo digestivo del insecto en donde se reproducen. El parásito es trasmitido a través de las heces del insecto, las cuales son depositadas en la piel del hospedero. Muchos triatominos defecan al momento de alimentarse. Los parásitos ingresan de las heces al torrente sanguíneo cuando el hospedero frota la herida causada por la picadura. Esta forma de transmisión realizada a través de un agente orgánico intermediario (en este caso *Triatoma infestans*), es conocida como transmisión por vector [3]. Existen otras formas de transmisión no vectoriales. Es posible que una madre portadora del protozoario trasmita la enfermedad a su bebé durante el embarazo o el parto. La enfermedad también puede ser trasmitida mediante la ingesta de alimentos contaminados con *Trypanosoma cruzi*. Otra forma de contagio es mediante la transfusión de sangre o transplante de órganos donados por una persona contagiada con la enfermedad [3].

Una vez que los parásitos ingresan al torrente sanguíneo, se pueden identificar tres fases de la enfermedad. La primera de éstas es la fase aguda, la cual da inicio al momento de producirse la infección por cualquiera de las formas antes mencionadas. Esta etapa dura entre 15 y 90 días y generalmente es asintomática, aunque un leve porcentaje de la

población presenta malestares durante esta etapa [3]. Uno de los signos de esta enfermedad es la inflamación del área por donde ingresaron los parásitos, conocido como chagoma y es consecuencia de la lesión ocasionada por el insecto al momento de alimentarse. Esta lesión desaparece durante el transcurso de la fase aguda. Cuando se presentan síntomas durante esta etapa, los más comunes son: decaimiento, dolor de cabeza, fiebre, escalofríos, pérdida de apetito e inflamación de los ganglios [28].

La fase aguda es seguida por la fase de latencia o fase indeterminada y se caracteriza por ser completamente asintomática. Durante esta etapa se puede detectar la enfermedad mediante el análisis de sangre. La duración de esta fase puede ser de varios años e incluso toda la vida del portador [3].

La etapa final de la enfermedad es conocida como fase crónica y es desarrollada por 2 ó 3 de cada 10 personas infectadas [3]. Esta etapa se manifiesta entre 20 y 30 años después de adquirir el parásito. Durante esta etapa el parásito se reproduce en el tejido muscular y es raro encontrarlo en la sangre. Las afectaciones más importantes de esta etapa son la cardiopatía chagásica y las afectaciones gastrointestinales. Una vez que los parásitos alcancen el músculo cardíaco, la infección crónica del músculo cardíaco conduce al fallo cardíaco y muerte del portador.

El diagnóstico de la enfermedad depende de la fase en la que ésta se encuentre. Demostrar la presencia del parásito constituye el diagnóstico certero, sin embargo esto sólo es posible durante la fase aguda. Los estudios generalmente incluyen el análisis microscópicos que buscan la presencia del parásito en la muestra tratada con técnicas de tinción como la tinción Wright [16] (la Fig. 2.1 fue obtenida utilizando esta técnica de tinción). Durante la fase crónica el diagnóstico se lleva acabo mediante la detección de anticuerpos anti T. cruzi [3]. La enfermedad es tratable con medicamentos durante la fase aguda. Los medicamentos empleados son efectivos en un 80 % de los casos [28]. No existe un tratamiento

efectivo para la fase crónica por lo que es importante un control médico periódico para evitar lesiones mayores a causa de este padecimiento.

2.2. Visión Computacional

En el desarrollo de este trabajo son de vital importancia dos áreas del conocimiento relacionadas con la inteligencia artificial. Una de ellas es la visión computacional. En este proyecto, la principal labor de los algoritmos de visión computacional es la de encontrar representaciones de la escena que posteriormente puedan ser interpretadas por los algoritmos de aprendizaje automático, así como limitar el espacio de búsqueda y eliminar la mayoría de los elementos carentes de interés en la imagen.

La visión computacional tiene por objetivo realizar decisiones útiles acerca de los objetos físicos y escenas reales, mediante el análisis de las imágenes adquiridas [19]. Difiere de las gráficas por computadora en el sentido de que este último intenta generar escenarios reales partiendo de datos. La visión computacional trata el problema opuesto, dado un conjunto de imágenes de escenas reales, intenta recuperar las propiedades de ésta, como lo son las formas, la iluminación y las distribuciones de color entre otras. Una imagen contiene información relevante y para un ser humano es una labor sencilla identificar los aspectos importantes dentro de la misma. Sin embargo, esto no es así para una computadora digital, la cual requiere de técnicas matemáticas para interpretar el contenido de la imagen.

La visión computacional tiene varias aplicaciones en diferentes áreas, las cuales incluyen el reconocimiento óptico de caracteres, el modelado y reconstrucción tridimensional, vigilancia y monitoreo, biometría y en el campo de la medicina con el análisis de imágenes médicas. Este último punto es el área de interés del presente trabajo.

2.2.1. Representación de una Imagen

Las imágenes digitales en escala de grises son representadas mediante arreglos bidimensionales discretos. Los objetos 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. Por lo tanto, una imagen es representada como una matriz:

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

de dimensiones $m \times n$ y donde cada entrada de la matriz $a_{i,j}$ representa un pixel de la imagen. Es común considerar a la imagen como una función bidimensional $I(x, y)$ donde x representa a las columnas y y a las filas de la imagen. En este sentido, $I(x, y)$ significa el

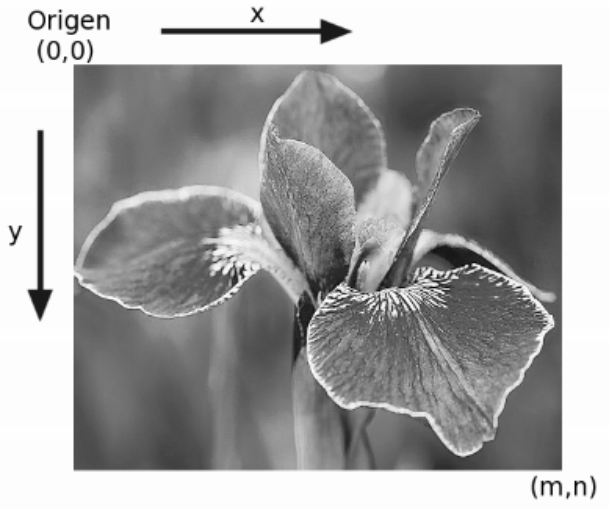

Figura 2.4: El origen de una imagen está situado en la esquina superior izquierda de la misma. Las flechas indican la dirección positiva de los ejes x, y.

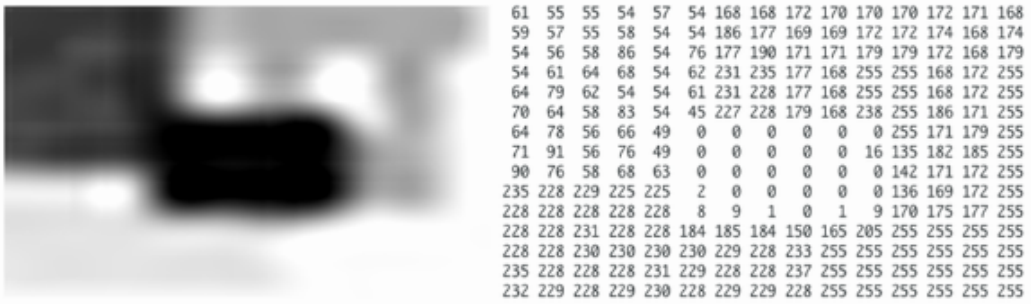

Figura 2.5: Los pixeles de una imagen en escala de grises son representados mediante valores enteros en el rango de 0 a 255. Donde 0 representa el nivel de gris negro y 255 el blanco.

valor del pixel ubicado en la posición (x, y) de la imagen, por ejemplo, la notación $I(0, 1)$ haría referencia al valor del pixel ubicado en la columna 0 y fila 1 de la imagen, lo cual equivale a la entrada $a_{1,0}$ de la matriz (2.1). Ambas coordenadas toman valores enteros con $0 \leq x < n$ y $0 \leq y < m$. El origen de la imagen, por lo general, se establece en la esquina superior izquierda de la misma [8]. El valor de los ejes crece conforme a las columnas y filas. Esto se ilustra en la Fig. 2.4.

Los pixeles de una imagen, o equivalentemente, las entradas de la matriz (2.1), son representadas como enteros positivos (o reales positivos dependiendo del problema a resolver), los cuales toman valores que van desde cero hasta 255, para el caso de una imagen en escala de grises. Para estas imágenes, el valor cero representa el color negro. Los pixeles son mas claros conforme mayor sea el valor de los mismos, teniendo un límite en 255, el cual representa el color blanco en la escala de grises (Fig. 2.5).

Los pixeles de una imagen a color son representados de una forma diferente. Durante el desarrollo de este proyecto se emplearon imágenes a color representadas en el espacio de colores RGB [24]. En esta representación, cada pixel es representado mediante tres cantidades (también llamados canales) que indican las intensidades de rojo, verde y azul (rgb por sus siglas en inglés) del color, el cual es el resultado de las contribuciones de estos

tres valores. Por lo tanto cada pixel $\boldsymbol{p}$ ubicado en la posición (x, y) de la imagen, puede verse como un vector de tres elementos:

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

donde cada elemento del vector puede 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 el color negro. De forma similar, si los tres canales son todos 255 obtendremos el color blanco.

Es pertinente mencionar algunas relaciones entre los pixeles de una imagen. Dada una imagen $I(x, y)$, un pixel $\boldsymbol{p}$ ubicado en la posición (x, y) de la imagen posee un vecindario alrededor de si mismo. Un vecindario es un conjunto de pixeles que rodean a $\boldsymbol{p}$. Se conoce como vecindario 4, al conjunto formado por los dos pixeles horizontales y dos pixeles verticales que rodean a $\boldsymbol{p}$ y se denota como $N_4(\boldsymbol{p})$. Este vecindario está formado por los pixeles que están arriba, abajo, izquierda y derecha de $\boldsymbol{p}$. Las coordenadas en la imagen de este vecindario respecto a $\boldsymbol{p}$ son:

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

Un segundo vecindario está formado por los cuatro pixeles diagonales inmediatos a $\boldsymbol{p}$, y es denotado por $N_D(\boldsymbol{p})$. La ubicación de estos pixeles respecto a $\boldsymbol{p}$ está dada por:

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

La unión de estos dos vecindarios forman un tercer conjunto de pixeles conocido como vecindario 8 el cual es denotado como $N_8(\boldsymbol{p})$. Si $\boldsymbol{p}$ se encuentra en el borde de la imagen, algunos pixeles pertenecientes a cada uno de estos vecindarios estarán fuera de la imagen,

es decir, sus coordenadas excederán las dimensiones de la imagen o serán menores que cero [8].

2.2.2. El Histograma de una Imagen

El histograma de una imagen está dado por el conteo de las incidencias de los valores de los pixeles de la imagen, dicho de otro modo, es la frecuencia con la que se repite cada pixel en la imagen. El conteo puede incluir a todos los valores posibles para un pixel, es decir, de 0 a 255 y es posible que existan valores que no se encuentren en ningún pixel de la imagen, lo cual indicaría una frecuencia de cero para dicho valor. También se puede establecer un conjunto de intervalos y realizar el conteo de acuerdo a la cantidad de pixeles de la imagen que se encuentran en cada intervalo. La Fig. 2.6 muestra una imagen en escala de grises con su respectivo histograma.

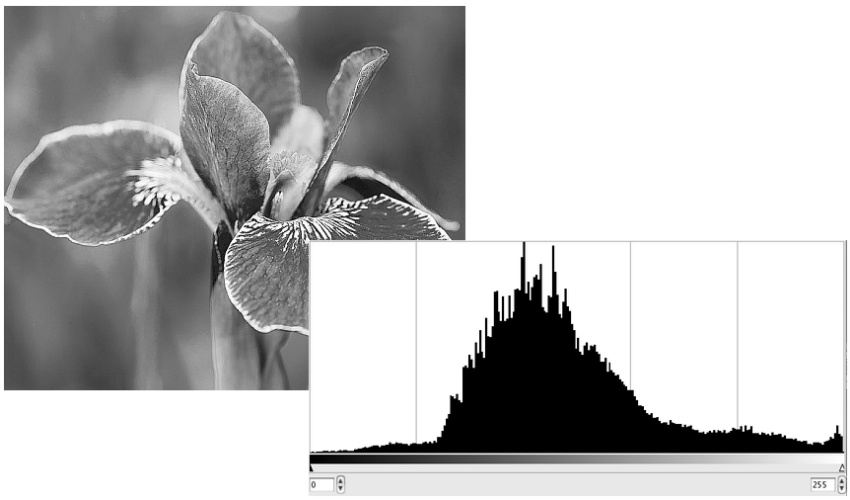

Figura 2.6: El histograma de una imagen obtenido con el programa de procesamiento de imágenes de GNU, GIMP (http://www.gimp.org).

Estas estructuras pueden ser empleadas para la identificación de puntos de interés en una imagen. También recolectan información útil y pueden ser usados como elementos característicos, en conjunto con un clasificador, para la detección de objetos en una imagen, comparando el histograma del objeto de interés con los histogramas obtenidos de las imágenes en donde se desea verificar la existencia de dicho objeto [1][7]. Un elemento característico es aquel que representa a un objeto o a un elemento particular de una imagen. Si los elementos característicos de dos objetos son semejantes entonces es muy probable que ambos objetos sean el mismo.

2.2.3. Filtros

Los filtros pueden ser útiles para eliminar el ruido en una imagen y darle una distribución más uniforme a la misma. Se pueden clasificar en filtros lineales y filtros no lineales. Para describir lo que son los filtros lineales es necesario definir lo que es una máscara y como éstas son aplicadas a la imagen. Las máscaras son empleadas en el procesamiento digital de imágenes y están relacionadas con el operador convolución. Una máscara, también conocida como kernel, se puede ver como un conjunto de pixeles, es decir, es una pequeña imagen en la que los valores de los pixeles son conocidos como pesos. La Fig. 2.7 presenta algunas máscaras que podrían ser aplicadas a una imagen [19]. Las máscaras tienen un

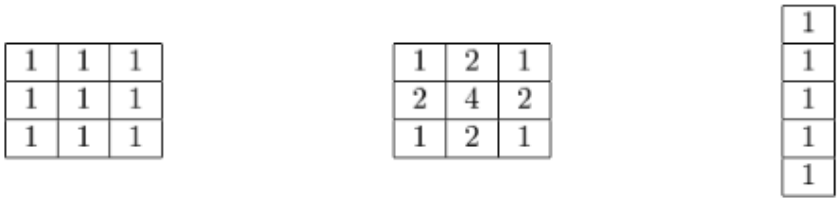

Figura 2.7: Ejemplos de máscaras que pueden ser aplicadas a una imagen. De izquierda a derecha las primeras dos máscaras son cuadradas y la tercera es rectangular.

origen. Para una máscara simétrica (como ambas máscaras de 3×3 elementos en la Fig. 2.7) se encuentra, comúnmente, en el centro de la misma. Por ejemplo, un kernel de 3×3 tendría su origen situado en la coordenada $(2, 2)$. Para las máscaras no simétricas el origen puede establecerse en cualquier elemento de la misma.

Los filtros modifican el valor de un pixel $\boldsymbol{p}$ de acuerdo al kernel proporcionado y al valor del vecindario que rodea a $\boldsymbol{p}$ y que entra en el rango de la máscara. Llamemos respuesta de $\boldsymbol{p}$ al filtro w al valor resultante de aplicar el filtro w sobre $\boldsymbol{p}$. Un filtro w es lineal si cumple con la condición de linealidad:

$$\begin{aligned} w(a+b) &= w(a) + w(b) \\ \alpha w(a) &= w(\alpha a) \end{aligned} \tag{2.5}$$

donde a y b son dos señales (podrían considerarse pixeles para este caso) y α un escalar cualquiera [8]. La respuesta de una imagen al filtro lineal se obtiene aplicando la máscara a todos los pixeles de la imagen. Para una imagen $I(x, y)$, la respuesta $G(x, y)$ a la máscara $w(s, t)$ de dimensiones $m \times n$ está dada por la convolución de la imagen con la máscara definida como:

$$G(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) I(x + s, y + t) \tag{2.6}$$

donde $a = (m-1)/2$ y $b = (n-1)/2$. Esta ecuación se aplica a todos los pixeles de I para obtener la respuesta G, la cual será otra imagen.

Los filtros no lineales son aquellos que no cumplen con la propiedad descrita por la ecuación (2.5) [8]. Como ejemplo de este tipo de filtros podemos mencionar al filtro de mediana. La mediana de un arreglo ordenado de N elementos, se define como el elemento en la posición $(N-1)/2$ si N es impar. Para el caso de arreglos con un número par de elementos se tendrán dos medianas, la primera en la posición $N/2$ y la segunda ubicada en la posición $N/2-1$. Para estos casos es posible tomar como única mediana el promedio de ambas como única mediana [19]. Para el filtro de mediana, se define una región o ventana

de búsqueda en lugar de una máscara. Esta ventana de búsqueda define el vecindario a considerar en el arreglo en donde se buscará la mediana. Esto indica que este vecindario será ordenado para posteriormente encontrar la mediana. Al igual que en las máscaras de los filtros lineales, esta ventana posee un origen el cual es, comúnmente, el centro de la misma. Un pixel de una imagen I ubicado en el origen de la ventana de búsqueda tendrá como respuesta a este filtro la mediana del vecindario contenido dentro de la ventana de búsqueda. La respuesta a este filtro se obtiene aplicando el mismo a cada pixel de la imagen. La Fig. 2.8 muestra el resultado de aplicar el filtro de mediana con una ventana de 5×5 a una imagen.

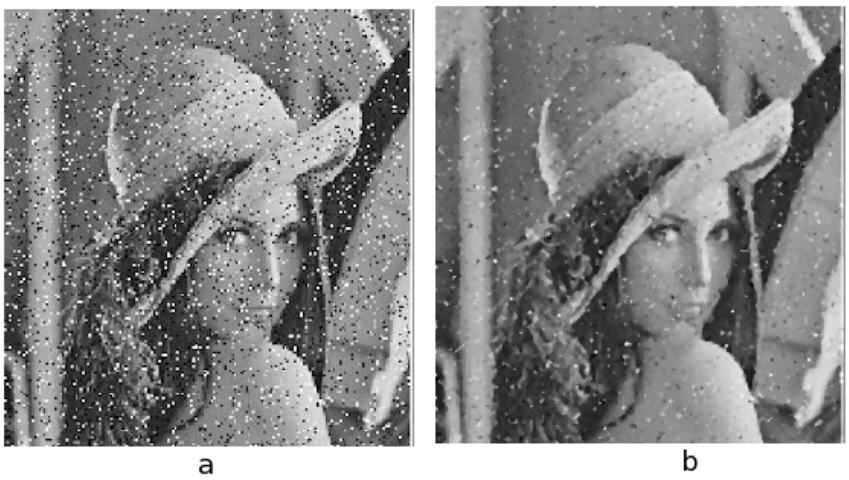

Figura 2.8: La imagen **b** es el resultado de aplicar un filtro mediana de 5×5 a la imagen **a**.

Una de la ventajas de un filtro de mediana es su tendencia a conservar los vértices de los objetos de la imagen, al mismo tiempo que suaviza regiones uniformes. Reduce el ruido de tipo sal y pimienta (pequeñas regiones blancas u oscuras como las presentes en la imagen **a** de la Fig. 2.8) y otros pequeños artefactos que pudieran presentarse en la imagen [19], entendiendo por artefactos, los objetos presentes en la imagen que no existen en la escena real de la cual la imagen fue adquirida.

2.2.4. Segmentación

La segmentación es el proceso de particionar una imagen en regiones. También puede entenderse como un problema de agrupamiento en el que se trata de recolectar aquellos datos presentes en la imagen que adquieren cierto sentido al agruparse, de acuerdo a un modelo previamente definido [7].

Una aplicación común de la segmentación es la umbralización. Suponiendo que la imagen está compuesta por objetos claros en un fondo oscuro de modo que los niveles de gris de los pixeles de los objetos y del fondo se encuentran agrupados en dos regiones dominantes, es posible extraer los objetos del fondo mediante la elección de un umbral T que separe ambas regiones, es decir, dada una imagen I, un pixel $\boldsymbol{p}$ ubicado en la posición (x, y) pertenece a un objeto en la imagen si $I(x, y) > T$, en otro caso, el pixel es considerado como parte del fondo. En general, una imagen umbralizada $G(x, y)$, es decir, el resultado de aplicar el proceso de umbralización a cada pixel de una imagen $I(x, y)$, está dada por:

$$g(x,y) = \begin{cases} 1 & \text{si } I(x,y) > T \\ 0 & \text{si } I(x,y) \leq T \end{cases} \tag{2.7}$$

Al final se obtendrá una imagen binaria g, en la cual los pixeles del fondo estarán representados por el valor cero mientras que los objetos tendrán el valor uno [7]. Una imagen binaria es aquella en la que los pixeles solo pueden tomar uno de dos valores posibles. La Fig. 2.9 muestra los resultados de aplicar este proceso de umbralización. La etapa de segmentación propuesta en este trabajo realiza un proceso similar, sin embargo realiza la umbralización de acuerdo al valor de dos canales de la imagen a color RGB. Esto es descrito con mayor detalle en el capítulo 4 de este documento.

Una forma de segmentación que resulta importante mencionar es la segmentación de color. En este caso, el principal objetivo es segmentar objetos que poseen un color en

Figura 2.9: Resultado de aplicar el proceso de umbralización a una imagen con un umbral de 180. Los objetos por debajo de este umbral son degradados a cero mientras que los que están por arriba son establecidos en uno.

particular, en una imagen RGB. Dado un conjunto de pixeles representativos del color de interés, es posible obtener un vector RGB promedio que representa el color que se desea segmentar. Cada pixel de la imagen es comparado con este modelo representativo para determinar si dicho pixel posee el color de interés. La comparación es realizada mediante alguna métrica de distancia como la distancia Euclidiana. Dados dos pixeles RGB $\boldsymbol{p}$, $\boldsymbol{q}$, ambos con la forma descrita por la ecuación (2.2), la distancia Euclidiana se define como:

$$D(\boldsymbol{p}, \boldsymbol{q}) = [(r_p - r_q)^2 + (g_p - g_q)^2 + (b_p - b_q)^2]^{1/2} \tag{2.8}$$

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 $\boldsymbol{q}$ [8]. La segmentación puede ser realizada de forma similar a la umbralización descrita anteriormente, calculando la distancia de los pixeles de la imagen con el modelo obtenido y comparando esta distancia con cierto umbral T, con la diferencia de que los valores por debajo del umbral serán considerados del color de interés (Fig. 2.10), como se muestra en la siguiente condición:

$$g(x, y) = \begin{cases} 1 & \text{si } D(\boldsymbol{p}, \boldsymbol{q}) < T \\ 0 & \text{si } D(\boldsymbol{p}, \boldsymbol{q}) \geq T \end{cases} \tag{2.9}$$

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

Figura 2.10: 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.

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. Éste es un problema de clasificación binaria, el cual es discutido en el apartado 2.3.1 de este capítulo.

2.2.5. Etiquetación

Dada una imagen binaria, posiblemente resultante de una umbralización o clasificación, es de interés saber que pixeles pertenecen a un mismo objeto en la imagen. La etiquetación resuelve este problema encontrando los conjuntos de pixeles que se encuentran conectados unos con otros en una imagen binaria. Recordemos que el valor v de un pixel $\boldsymbol{p}$ en una

imagen binaria sólo puede tomar dos valores posibles, comúnmente 0 ó 1. Suponiendo que B es una imagen binaria en la que $B(x, y) = B(x', y') = v$, con $v = 0$ o $v = 1$ son valores de dos pixeles en B. Se dice que el pixel (x, y) está conectado al pixel (x', y') si existe una secuencia de pixeles $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$ en donde $(x_0, y_0) = (x, y)$, $(x_n, y_n) = (x', y')$, $B(x_i, y_i) = v$ para $i = 0, 1, \ldots, n$ y (x_i, y_i) pertenece al vecindario 8 de (x_{i-1}, y_{i-1}) para $i = 0, 1, \ldots, n$. [19]. Dos pixeles $\boldsymbol{p}$ y $\boldsymbol{q}$ de una imagen binaria están conectados si existe una secuencia de pixeles vecinos, comenzando en $\boldsymbol{p}$ y terminando en $\boldsymbol{q}$, tal que cada pixel de la secuencia posee el mismo valor.

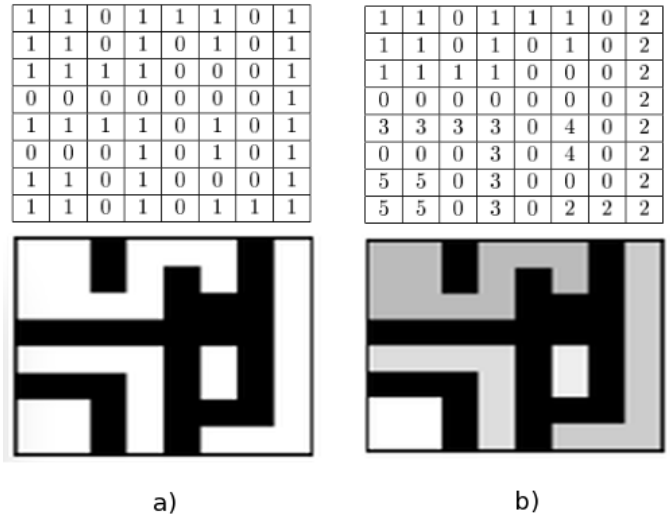

Figura 2.11: a) Una imagen binaria. La imagen superior muestra el valor de sus pixeles. b) Imagen etiquetada. La imagen superior muestra el valor de sus pixeles y permite apreciar como los pixeles conectados distintos de cero poseen la misma etiqueta.

Una imagen etiquetada de una imagen binaria B, es una imagen en la cual cada pixel $\boldsymbol{p}$ posee el mismo valor, o etiqueta, que sus respectivos pixeles conectados. Una etiqueta es un valor numérico único que identifica a cada conjunto de pixeles conectados. La etiquetación permite ubicar los objetos presentes en una imagen. Todos los pixeles de un mismo objeto

tendrán la misma etiqueta y objetos distintos tendrán etiquetas diferentes como valor de sus pixeles. Esto permite, entre otras cosas, contar el número de objetos en la imagen, ubicar la posición del objeto en la imagen y encontrar el número de pixeles que componen a un objeto. La Fig. 2.11 muestra el resultado de la etiquetación de una imagen binaria en la cual se puede observar como los pixeles conectados distintos de cero van adquiriendo una intensidad uniforme y diferente para cada grupo de pixeles.

2.2.6. La Imagen Integral

Algunos de los métodos de visión computacional requieren el cálculo de sumatorias sobre el valor de los pixeles en una región rectangular determinada. Viola y Jones [26] [27] proponen una representación intermedia de la imagen que permite calcular estas sumatorias de una forma rápida y eficiente. Esta representación es conocida como imagen integral.

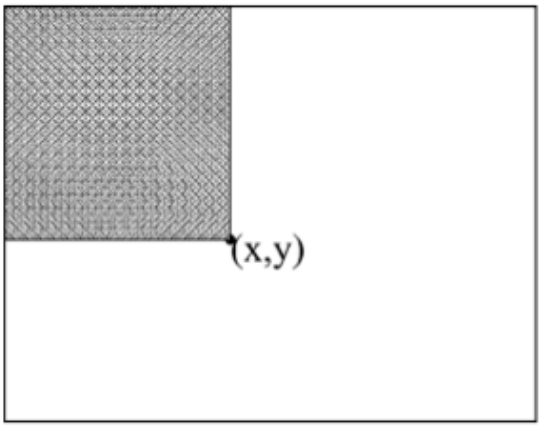

Figura 2.12: La imagen integral en la posición (x, y) es la sumatoria de todos pixeles dentro de la región sombreada.

La imagen integral en el punto (x, y) contiene la sumatoria de todos los pixeles que poseen coordenadas menores o iguales a (x, y) (Fig. 2.12). Formalmente, dada una imagen I en escala de grises, la imagen integral se define como:

$$II(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y') \tag{2.10}$$

donde $II(x, y)$ es el valor de la imagen integral en la posición (x, y) e $I(x', y')$ es valor del pixel en la posición (x', y') de la imagen original. La imagen integral acumula las sumatorias de los valores de los pixeles de la imagen hasta (x, y) y puede ser calculada haciendo uso de las siguientes fórmulas:

$$\begin{aligned} S(x, y) &= S(x, y-1) + I(x, y) \\ II(x, y) &= II(x-1, y) + S(x, y) \end{aligned} \tag{2.11}$$

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 $II(-1, y) = S(x, -1) = 0$. Esto permite calcular la imagen integral realizando únicamente un recorrido sobre la imagen original. Una vez calculada la imagen

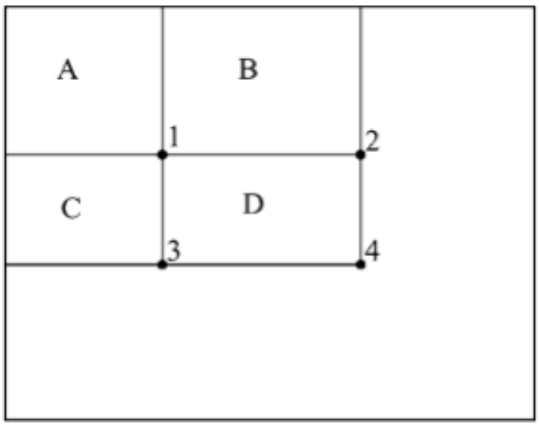

Figura 2.13: La sumatoria de los pixeles de una imagen, dentro de cualquier región rectangular puede ser calculada usando 4 referencias en la imagen integral.

integral, es posible realizar la suma de los pixeles dentro de una región rectangular de cualquier tamaño usando únicamente cuatro referencias correspondientes a las coordenadas de las esquinas de la región (Fig. 2.13).

Tomando como ejemplo la Fig. 2.13, la suma de los pixeles dentro de la región rectangular D estaría dada por la siguiente expresión:

$$4 + 1 - (2 + 3) \tag{2.12}$$

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.

2.3. Aprendizaje Automático

El aprendizaje automático es una rama de la inteligencia artificial que trata de los problemas de aprendizaje computacional, es decir, se encarga de diseñar métodos que le permiten a una computadora "aprender". Se dice que un programa computacional aprende si su desempeño en cierta tarea mejora conforme incrementa su experiencia, respecto a alguna medida de desempeño definida [11]. Por ejemplo, un programa puede tener la tarea de reconocer caracteres en un documento, la experiencia podría ser adquirida mediante un cierto número de ejemplos previamente clasificados y la medida de desempeño podría estar dada por el porcentaje de letras correctamente clasificadas [11]. El aprendizaje automático tiene varias aplicaciones entre las que se encuentran la traducción automática de documentos, la seguridad y control de acceso mediante el reconocimiento y clasificación de rostros y la detección de fraudes en tarjetas de crédito entre otras [20].

En el proceso de aprendizaje es posible identificar tres tipos de retroalimentación. La retroalimentación le permite al programa elegir las acciones que mejoran su desempeño dentro de un conjunto de acciones disponibles. El primero de estos casos está dado por los métodos de aprendizaje supervisado, en los cuales se realiza el aprendizaje de una función a partir de ejemplos de las entradas y salidas de la misma. La función será la encargada de definir las nuevas salidas para las nuevas entradas, es decir, decide que acciones realizar tomando en cuenta las entradas que recibe. En el aprendizaje supervisado el proceso de aprendizaje se realiza empleando ejemplos previamente conocidos. Considérese como ejemplo el problema de identificar dígitos en una imagen. Un método de aprendizaje supervisado emplearía una secuencia de imágenes que contengan un número junto con la información sobre que número es el que se encuentra en cada imagen. El aprendizaje no supervisado, por otra parte, emplea ejemplos (entradas) de los cuales no se tiene conocimiento de su identidad (salidas). Finalmente, las técnicas de aprendizaje por refuerzo realizan el aprendizaje basado en estímulos, por ejemplo, dando indicaciones sobre que comportamiento es deseable (salida) dada determinada situación (entrada) [18]. Cabe destacar que para este trabajo se emplearon métodos de aprendizaje supervisado, es decir, los algoritmos de aprendizaje fueron entrenados con un conjunto de entradas de las cuales se conocía previamente la salida.

2.3.1. Clasificación Binaria

Anteriormente, se había mencionado que el problema de segmentación puede verse como un problema de clasificación en el cual cada pixel de la imagen puede pertenecer a una de las clases disponibles. Para el caso de la umbralización el pixel puede pertenecer al fondo o a los objetos. La clasificación es un problema frecuente en el práctica en el cual

se tiene un conjunto de datos y se desea asignar cada elemento de este conjunto a una de dos o más clases disponibles.

Figura 2.14: Ejemplos de clasificación binaria. En ambas imágenes se tienen dos grupos disponibles (estrellas/rombos y estrellas/triángulos). El objetivo es encontrar una función que adecuadamente identifique los elementos de los grupos disponibles.

Cuando se tiene dos clases posibles se trata de un problema de clasificación binaria (Fig. 2.14). El problema se define como un conjunto de valores o instancias χ pertenecientes a un espacio de instancias $\boldsymbol{\chi}$ y un conjunto de etiquetas o clases $\gamma \in \{-1, +1\}$, en donde cada instancia está relacionada con una etiqueta, lo cual es denotado (χ, γ), $\chi \in \boldsymbol{\chi}$, $\gamma \in \{-1, +1\}$. Las instancias son las entradas del clasificador y las etiquetas, las cuales modelan las dos clases disponibles, son las salidas. En este trabajo fueron implementados clasificadores binarios entrenados mediante métodos de aprendizaje supervisado. Un conjunto de entrenamiento τ de m instancias para un clasificador binario es representado como $\tau = \{(\chi_i, \gamma_i)\}(i \in [1, 2, 3, \ldots, m])$, donde se conoce previamente, las salidas correspondientes para cada instancia del conjunto τ [20] [29].

2.3.2. El algoritmo de los K Vecinos más Cercanos

El algoritmo de los k vecinos más cercanos en un método de clasificación basado en la cercanía que tiene el elemento prueba z respecto a los elementos del conjunto de entrena-

miento τ. Para la clasificación se encuentran las k instancias del conjunto de entrenamiento que estén más cercanas al elemento a prueba z, y se asigna la clase γ que más veces se repita dentro de los k elementos más cercanos (Fig. 2.15). Es común que las instancias sean

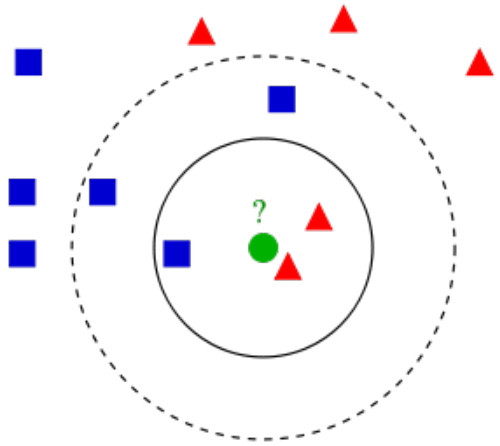

Figura 2.15: El algoritmo de los k vecinos más cercanos clasificará al círculo en la imagen según la clase de los elementos más cercanos. Con $k = 3$ el círculo sería clasificado como triángulo y con $k = 5$ se le asignaría la clase de los cuadrados.

vectores numéricos de n dimensiones, los cuales describen los componentes del problema a resolver. La medida de cercanía puede ser una métrica de distancia, como la distancia Euclidiana. Esta métrica fue presentada en la ecuación (2.8) para el caso de un vector de tres elementos (un pixel RGB). De forma general, la distancia Euclidiana entre dos vectores $\boldsymbol{x}$, $\boldsymbol{y}$ de n elementos está dada por:

$$D(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.13}$$

donde x_i y y_i son, en este caso, las i-ésimas entradas de los vectores $\boldsymbol{x}$ y $\boldsymbol{y}$ respectivamente. Otra métrica que es comúnmente empleada es la distancia de Manhattan definida como

$$D(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{n} \mid x_i - y_i \mid} \tag{2.14}$$

nuevamente, x_i y y_i definen las i-ésimas entradas de los vectores $\boldsymbol{x}$ y $\boldsymbol{y}$ respectivamente. Uno de los factores que influyen en el desempeño de este algoritmo es la selección del número k de elementos a comparar. Si el número de elementos es muy pequeño o muy grande es posible que realice una clasificación incorrecta (Fig. 2.15) [29]. El algoritmo de los k vecinos más cercanos se define a continuación.

Algoritmo 1 Algoritmo de los k vecinos más cercanos

Entrada: τ, el conjunto de instancias de entrenamiento, z, el elemento prueba que será clasificado y L, el conjunto de etiquetas disponible. Para la clasificación binaria $L = \{-1, +1\}$.

Salida: $c_z \in L$ la clase de z.

1: **para todo** instancia $\gamma \in D$ **hacer**

2: Calcular la distancia $D(z, \gamma)$ entre z y γ.

3: **fin para**

4: Seleccionar un subconjunto N de τ que contenga las k instancias del conjunto de entrenamiento más cercanas a z.

5: Establecer

$$c_z = \arg\max_{v \in L} \sum_{y \in N} Ind(v = clase(c_y))$$

donde Ind es una función indicadora que devuelve uno si la condición es verdadera y cero en cualquier otro caso.

La función indicadora devolverá uno cuando v sea igual a la clase de la instancia $y \in N$. De esta forma, la etiqueta que maximice la sumatoria será aquella que más se repita en el subconjunto N. La métrica de distancia puede ser alguna de las antes mencionadas u otra que mejor se ajuste al problema en cuestión.

2.3.3. Boosting y el Algoritmo 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 sólo es ligeramente menor que una decisión aleatoria. El *boosting* es un proceso iterativo que genera clasificadores

Algoritmo 2 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 2.

Algoritmo 3 El algoritmo AdaBoost.

Entrada: $\tau =$, 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 probabilidad de que el clasificador débil se equivoque al momento de clasificar las instancias de entrenamiento. De acuerdo con [29], 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.

$$\epsilon = E_{x \sim Ds}[\boldsymbol{y} \neq h_t(\boldsymbol{x})] \tag{2.15}$$

El Algoritmo 2 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 3.

2.4. Estado del Arte

En esta sección se presentan algunos de los trabajos que se han realizado para la detección de parásitos en la sangre. Se revisan los trabajos relativos a *Trypanosoma cruzi* y a *Plasmodium*, este segundo es el agente responsable de la malaria. También se presentan algunos trabajos realizados en el conteo de células, algunos de ellos relacionados con el análisis de sangre y el conteo de bacterias.

2.4.1. Diagnóstico de la Enfermedad de Chagas

Una revisión bibliográfica mostró que existen pocos trabajos realizados respecto a la detección de *Trypanosoma cruzi* en la sangre usando métodos de visión computacional. Entre los trabajos enfocados a la detección de este parásito es posible mencionar el realizado en [9]. El objetivo de este trabajo, como su autor describe, es el de crear un instrumento portable que permita la detección de la enfermedad con base al análisis sanguíneo. El dispositivo cuenta con un sistema óptico provisto de una cámara para la adquisición de las imágenes. El controlador de este dispositivo fue implementado con un arreglo de com-

puerta de campo programable (FPGA, por sus siglas en inglés), el cual es una arreglo de compuertas lógicas reconfigurable cuya programación puede realizarse mediante lenguajes de descripción de hardware.

El sistema es capaz de realizar un análisis de las imágenes para determinar la presencia de *Trypanosoma cruzi* en la muestra de sangre. Las imágenes son capturadas de un portaobjetos con una muestra de sangre, el cual es introducido en el dispositivo para iniciar el análisis. El método de análisis de imágenes propuesto está basado en el hecho de que el parásito es un ser vivo con capacidad de movimiento, gracias al flagelo que posee. Por lo tanto, *Trypanosoma cruzi* presentará un desplazamiento mayor que los componentes sanguíneos dentro de la muestra. Con base a lo anterior, el dispositivo recorre la muestra de forma automática posicionándose en una porción de la misma en cada paso del recorrido. Esta porción de la muestra es nombrada campo. Para cada campo del portaobjetos se captura una serie de imágenes, mismas que serán empleadas para detectar movimiento en ese campo. La detección de *Trypanosoma cruzi* se realiza de acuerdo al movimiento detectado en la muestra.

2.4.2. Detección de *Plasmodium*

Varios trabajos relacionados con la detección de parásitos sanguíneos están relacionados con la detección de *Plasmodium*. Algunas especies de este género son causantes de la malaria. Al igual que *Trypanosoma cruzi*, las especies del género *Plasmodium* causantes de la malaria son protozoarios parásitos, transmitidos al ser humano mediante vectores. A diferencia de *Trypanosoma cruzi*, los parásitos del género *Plasmodium* no tienen flagelo y el vector de transmisión son algunas especies de mosquito.

En [25] se presenta un método para la detección de parásitos de la malaria en imágenes de muestras sanguíneas tratadas con tinción giemsa. El proceso incluye una etapa de segmentación y una de clasificación. La segmentación es realizada mediante un clasificador Bayesiano de pixeles, el cual distingue entre los pixeles correspondientes a elementos teñidos y los pixeles que corresponden al fondo y otros elementos no teñidos. Las funciones de densidad de probabilidad de estas clases se estimaron con base al método del histograma, el cual emplea un histograma de tres dimensiones normalizado para que todos elementos sumen uno. El vector de características de los elementos teñidos es hallado para posteriormente emplear el método de los k vecinos más cercanos para realizar la clasificación final. Los vectores característicos que presentaron mejor desempeño fueron el correlograma, el cual cuenta las co-ocurrencias de dos pixeles que se encuentran a una distancia determinada uno del otro y que son de un color determinado en un conjunto de colores posibles. El histograma de color de la imagen fue el segundo vector característico en presentar un buen desempeño. La probabilidad de que un elemento clasificado como positivo a *Plasmodium*, empleando este método, sea realmente positivo es de 0.74, mientras la probabilidad de que un elemento clasificado como negativo sea realmente negativo es de 0.98.

Un método similar es propuesto en [21]. Al igual que en [25], el procedimiento comprende la segmentación de la imagen, la obtención de un vector de características y la clasificación de los elementos segmentados. El método de segmentación está compuesto por dos etapas. La primera de ellas realiza un análisis morfológico de los elementos de la imagen y como segunda etapa se tiene un proceso de umbralización. El vector de características propuesto toma elementos geométricos, entre ellos la forma y el tamaño del elemento segmentado, el color y la textura de estos elementos. La etapa de clasificación está compuesta por un árbol de clasificadores de dos etapas. La primera etapa determina la presencia de *Plasmodium* mientras que la segunda se encarga de estimar la especie a la que el parásito pertenece.

En [23], la segmentación es realizada mediante el procesamiento de una imagen binaria, resultado de un proceso de umbralización. Se emplea el algoritmo de transformación divisoria (*watershed transformation*) como principal método para separar los compuestos celulares. También fueron probados diversos vectores de características basados en forma, intensidad, textura e histogramas, siendo estos últimos los que presentaban mejor desempeño.

G. Diaz [6] propone un método de segmentación basado en el espacio de color normalizado RGB empleando el método de los k vecinos más cercanos como método de clasificación fondo/objeto. El vector de características fue formado a partir de cinco histogramas: el histograma de color, un histograma de nivel de saturación, histograma de escala de grises, histograma de textura Tamura y el histograma Sobel. La dimensionalidad de cada histograma fue reducida para formar un vector final de 25 características. En la etapa de clasificación se evaluaron dos clasificadores. Un clasificador basado en un percerptrón multicapa y un clasificador basado en máquinas de vectores de soporte (SVM, por sus siglas en inglés). El clasificador SVM fue el que presentó el mejor desempeño.

En [17] se emplean métodos de umbralización y métodos morfológicos para la segmentación de los eritrocitos y parásitos. El algoritmo busca identificar la presencia de *Plasmodium* y clasificarlo por especie. Para esto se propone dos vectores característicos. El primero se basa en elementos comúnmente usados en métodos de detección de *Plasmodium*, éstos son descriptores geométricos, atributos del color y textura en niveles de grises. El segundo vector de características propuesto está compuesto, principalmente, por características cuantitativas de las células afectadas por las especies del parásito. El clasificador posee dos etapas. La primera determina la presencia de malaria y la segunda determina la especie. Un clasificador basado en una red neuronal de retropropagación es el encargado de realizar la clasificación en ambas etapas.

Purwar [14] emplea métodos de minimización de energía para obtener bordes y posteriormente, considerando la forma casi circular de los glóbulos blancos, emplea la transformada de Hough para realizar el conteo de los glóbulos blancos. Para determinar la presencia del parásito en la imagen emplea el método de agrupación k medias, el cual agrupa los datos de acuerdo al valor de sus medias.

En [10] se aborda el tema de la segmentación de células rojas infectadas con malaria empleando el algoritmo de cortes normalizados, el cual es un método de segmentación no supervisado que modela la segmentación como un problema de partición de grafos. Este algoritmo fue probado en los espacios de color RGB, YCbCr, HSV y NTSC, presentando mejor desempeño en el espacio HSV, el cual es un modelo que representa los colores con base a su tono (H), saturación (S) y brillo (V) [24].

2.4.3. Detección y Conteo de Células

Finalmente, se presentan algunos trabajos relacionados con el análisis de imágenes microscópicas para la identificación y conteo de células sanguíneas y para el reconocimiento de bacterias en muestras de aguas residuales. Si bien estos trabajos no están enfocados a la detección de parásitos, tratan con el problema de detección y reconocimiento de células y microorganismos en una imagen microscópica.

El análisis de imágenes sanguíneas no sólo está enfocado a la detección de parásitos. Otros trabajos, como [13] y [5], abordan el problema de detección y reconocimiento de células sanguíneas, entre las que se encuentran los glóbulos rojos y los glóbulos blancos. En [13] se presenta un sistema para el reconocimiento, análisis y clasificación de células sanguíneas capaz de producir reportes sobre el conteo de las células identificadas. Las células son contadas con base a un portaobjetos con una cuadrícula de conteo. La cuadrícula

es detectada mediante detección de bordes y, posteriormente, detección de líneas. Para el conteo de células se realizan técnicas de contraste en el canal rojo de la imagen en una etapa de preprocesamiento. Se encuentra el contorno de los objetos de la imagen y se discrimina según el área de los contornos obtenidos. La discriminación célula/no-célula es realizada mediante métodos estadísticos que involucran la media y desviación estándar de los contornos de los objetos segmentados. En un proceso más complejo, el sistema clasifica las células blancas obtenidas, antes de contarlas. La segmentación se realiza mediante una umbralización en el histograma de color RGB de la imagen para posteriormente realizar una discriminación por contornos para determinar los elementos que corresponden a glóbulos blancos. La clasificación de estas células está a cargo de una red neuronal artificial.

En [5] se trata el problema de reconocimiento de leucocitos en imágenes tomadas a partir de muestras sanguíneas. El proceso se compone de tres etapas: segmentación, localización del vector de características y clasificación. El clasificador es implementado usando funciones de mezcla de Gaussianas para modelar las clases disponibles. Las funciones Gaussianas son obtenidas mediante el algoritmo de esperanza-maximización. Para inicializar este algoritmo se empleó el método de agrupación k medias. Para reducir la dimensionalidad de los datos se emplearon métodos de análisis de componente principal.

El trabajo desarrollado en [30] propone un método para el reconocimiento de bacterias en aguas residuales mediante el análisis de imágenes obtenidas por microscopía. Las etapas de segmentación están basadas en la detección de bordes en la imagen. El vector de características está dado por descriptores de forma basados en contorno. Al igual que en [5], el análisis de componente principal se emplea para reducir la dimensionalidad del vector de características. Para el reconocimiento se entrenó una red neuronal de retropropagación.

3. Formulación del Problema

En el capítulo anterior se describió la enfermedad de Chagas y los efectos que ésta puede tener sobre los humanos. Esta enfermedad, en la etapa crónica, puede ser mortal debido a los daños que ocasiona sobre el músculo cardíaco. Los estudios para la detección de la enfermedad en la etapa aguda involucran la búsqueda del parásito en una muestra de sangre analizada bajo el microscopio. Esta búsqueda es realizada de forma manual por un especialista, quien es capaz de reconocer al parásito en el plasma sanguíneo.

De igual forma, se presentaron algunas investigaciones relativas a la detección de parásitos causantes de la malaria en imágenes tratadas con muestras sanguíneas. Con todo esto surge la pregunta sobre la posibilidad de realizar un proceso similar al descrito en la detección de *Plasmodium*, pero ahora, enfocado a la detección de *Trypanosoma cruzi*, de forma que el diagnóstico de esta enfermedad se realice mediante el análisis de imágenes obtenidas a partir de muestras sanguíneas. En este contexto, la pregunta que este trabajo busca responder es la siguiente: ¿Es factible emplear métodos de visión computacional y aprendizaje automático para la detección de *Trypanosoma cruzi* en imágenes obtenidas a partir de muestras sanguíneas?

La revisión al estado del arte mostró que existen pocos trabajos realizados, específicamente, para la detección de *Trypanosoma cruzi* usando métodos de visión computacional.

Sin embargo, es posible que los procesos propuestos para la detección de *Plasmodium* se puedan emplear en la detección del parásito causante de la enfermedad de Chagas. En la detección de *Plasmodium*, como en otros trabajos de detección de objetos en imágenes, los principales pasos a seguir son la segmentación de las regiones de interés, la obtención de un vector de características descriptivas y la clasificación de las regiones a partir del vector de descriptores.

La detección automática o semiautomática de microorganismos en muestras sanguíneas es uno de los pasos a seguir en el prediagnóstico de enfermedades causadas por estos agentes infecciosos. Estos trabajos, junto con elementos de adquisición automática de imágenes microscópicas, contribuyen a la detección automática de microorganismos, permitiendo así analizar una mayor cantidad de imágenes en menor tiempo y reducir el espacio de búsqueda al conservar únicamente aquellas imágenes en donde se tiene la suficiente certeza de encontrar al parásito, lo cual deberá ser validado por el especialista para completar el diagnóstico. Este trabajo de tesis busca responder a la interrogante planteada en este capítulo y proponer un proceso de clasificación, que servirá como punto de partida para futuras investigaciones.

Cabe mencionar que esta tesis está enfocada a la detección del parásito en una imagen digital, es decir, se emplearán métodos de visión computacional para determinar si los pixeles de una subimagen en formato RGB $I_{sub}(x, y)$, obtenida de una imagen $I(x, y)$ en formato RGB, representan un protozoario parásito *Trypanosoma cruzi*.

4. Metodología

Esta sección describe el clasificador propuesto para la detección de *Trypanosoma cruzi* en imágenes de muestras sanguíneas tratadas con la tinción Wright , en las cuales es posible distinguir el núcleo y el cinetoplasto del parásito, los cuales quedan teñidos por el colorante (Fig. 4.2). Es posible distinguir dos etapas en el clasificador. La primera etapa segmenta la imagen para obtener aquellas regiones de interés en donde posteriormente se buscará la presencia del parásito. La segunda etapa es una fase de discriminación que busca clasificar los resultados de la segmentación como parásito o no parásito. A continuación se describe a detalle estas dos etapas. La Fig. 4.1 describe el proceso que el clasificador realiza para determinar la presencia de *Trypanosoma cruzi*.

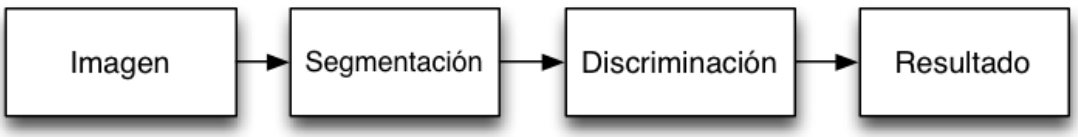

Figura 4.1: Proceso propuesto para la detección de *Trypanosoma cruzi*.

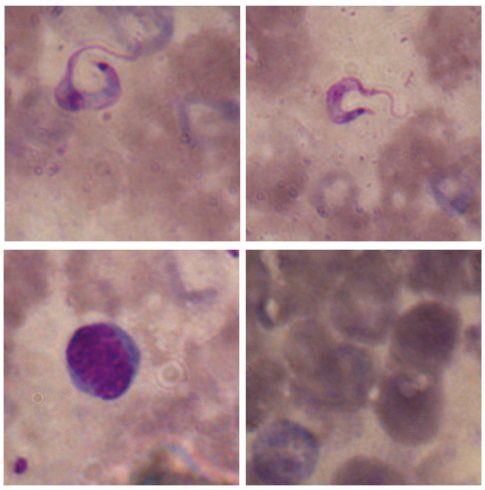

Figura 4.2: Las imágenes empleadas corresponden a muestras sanguíneas tratadas con tinción Wright . Las imágenes superiores presentan un parásito en la muestra en las cuales sobresale el núcleo al centro y el cinetoplasto en el extremo opuesto al flagelo.

4.1. Etapa de Segmentación

La segmentación es el proceso mediante el cual una imagen es dividida en varias regiones. Este proceso realiza un cambio en la representación de la imagen, permitiendo realizar un análisis con mayor facilidad [19]. Para los propósitos de este trabajo la imagen fue segmentada en dos regiones. La primera está conformada por el fondo, es decir, aquellos elementos de la imagen que no son de interés. Está compuesto principalmente por aquellos elementos no teñidos por el colorante. En contraste, la segunda región, nombrada como región de interés, está conformada por los elementos teñidos entre los cuales podría encontrarse el parásito. El método de segmentación desarrollado presenta varias etapas que se describen a continuación.

4.1.1. Diferencia de Canales

La segmentación empleada se compone de la intersección de los resultados de dos algoritmos de segmentación. El primero de estos algoritmos, nombrado como diferencia de canales, se basa en las propiedades que la imagen adquiere como consecuencia de la tinción realizada a la muestra.

El canal azul, en conjunto con el canal verde, aportan información relevante que permiten identificar una o ambas estructuras celulares presentes en el cuerpo del parásito, junto con algunos elementos de poco interés. En un pixel $\boldsymbol{p}$ perteneciente al cinetoplasto de *Trypanosoma cruzi* se observó que, en la mayoría de las veces, el canal azul presenta valores mayores comparado con el canal verde del mismo pixel. También se observó que esta propiedad es menos frecuente en los pixeles no teñidos correspondientes al fondo de la imagen. Por lo tanto, se decidió restar los canales azul y verde de un mismo pixel para compararlos y determinar si pertenecen a las estructuras celulares del parásito o si pertenecen al fondo de la imagen. Sea $\boldsymbol{p}$ un pixel representado por:

$$\boldsymbol{p} = \begin{bmatrix} r \\ g \\ b \end{bmatrix} \tag{4.1}$$

donde r, g y b representan los canales RGB del mismo. La primera segmentación es realizada comparando los canales verde (g) y azul (b) mediante la siguiente expresión:

$$b - g > T \tag{4.2}$$

donde T es un umbral que establece la diferencia mínima entre estos dos canales para que el pixel sea considerado como un pixel de interés. Dada una imagen $I(x, y)$, la regla de

segmentación está dada por:

$$G(x,y) = \begin{cases} 1 & \text{si } I(x,y)_b - I(x,y)_g > T \\ 0 & \text{de otra forma} \end{cases} \tag{4.3}$$

donde $G(x,y)$ es el valor del pixel en la posición (x,y) de la imagen de salida (imagen segmentada), $I(x,y)_b$, $I(x,y)_g$ son los valores de azul y verde del pixel ubicado en la posición (x,y) en la imagen de entrada I. El proceso es descrito en el Algoritmo 4. En este algoritmo se toma el valor absoluto de la diferencia, considerando la distancia que hay entre los canales G y B como regla de comparación.

Algoritmo 4 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 $\boldsymbol{p}$ tiene la forma descrita por la ecuación (4.1). **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 $G(x,y)$.

En esta primera etapa, se logra eliminar la mayor parte del fondo de la imagen. Entre los objetos segmentados, se encuentran glóbulos blancos y plaquetas, en adición a las estructuras celulares del parásito. La imagen central de la Fig. 4.3 muestra una imagen segmentada con este procedimiento. Al observar el comportamiento del algoritmo con diferentes umbrales, se decidió usar un valor de $T = 50$.

Las plaquetas y los elementos celulares del parásito son de menor tamaño comparados con los glóbulos blancos, que pueden ser eliminados mediante una etapa de discrimina-

Algoritmo 5 Etiquetación.

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

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

1: $etiqueta = 1$. // inicializar contador de etiquetas en 1.
2: **para todo** pixel $\boldsymbol{p}$ de la imagen $IB(x, y)$ en la posición (x, y) que sea diferente del fondo. **hacer**
3: Revisar el vecindario N_8 de $\boldsymbol{p}$.
4: **si** Algún vecino de $\boldsymbol{p}$ ya posee una etiqueta v **entonces**
5: Asignar $E(x, y) = v$.
6: **si no**
7: Asignar $E(x, y) = etiqueta$.
8: Incrementar *etiqueta* una unidad.
9: **fin si**
10: **fin para**
11: **devolver** La imagen etiquetada $E(x, y)$.

Figura 4.3: Resultados de la segmentación por diferencias de canales. Se observa que las estructuras celulares del parásito, al igual que los glóbulos blancos, fueron segmentadas. La tercera imagen es el resultado de aplicar la discriminación por área de la imagen central.

ción por áreas. Para calcular el área en pixeles de cada región segmentada se realiza una etiquetación a la imagen segmentada.

La etiquetación, como se describió anteriormente, es un método mediante el cual se asigna el mismo valor (etiqueta) a todos los pixeles de una misma región. Este valor es único entre las diferentes regiones. De esta forma se tiene el conocimiento sobre el número de regiones presentes. El Algoritmo 5 describe el proceso de etiquetación. Es común considerar en una imagen segmentada el valor cero como el fondo de la imagen y el valor uno como los objetos segmentados. Al final del proceso es necesario realizar una segunda revisión de las etiquetas asignadas para garantizar que todos los pixeles de un mismo objeto poseen la misma etiqueta. Antes de realizar el proceso de etiquetación se realiza un filtrado Gaussiano para suavizar la imagen segmentada y eliminar el ruido que pudo haberse obtenido durante la segmentación. El filtro Gaussiano es un filtro lineal que emplea un kernel de 3×3 en la que las entradas fueron obtenidas de una función Gaussiana.

Una vez calculada la imagen etiquetada, el área es hallada contando el número de pixeles etiquetados por cada región. El área de cada región es comparada con un umbral de área máxima. Si la región excede este umbral es eliminado del conjunto de regiones de interés y pasa a formar parte del fondo. También se determinó un umbral de área mínima para eliminar pequeños artefactos que lograron cumplir con el criterio de segmentación.

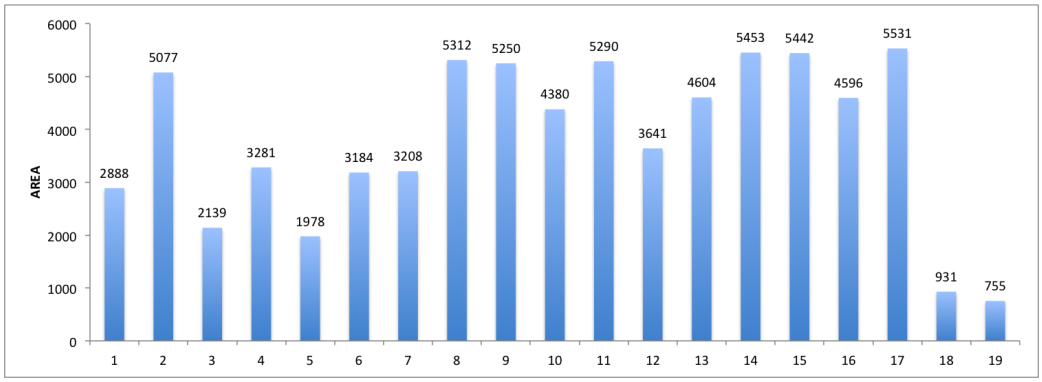

Figura 4.4: Áreas de las células sanguíneas.

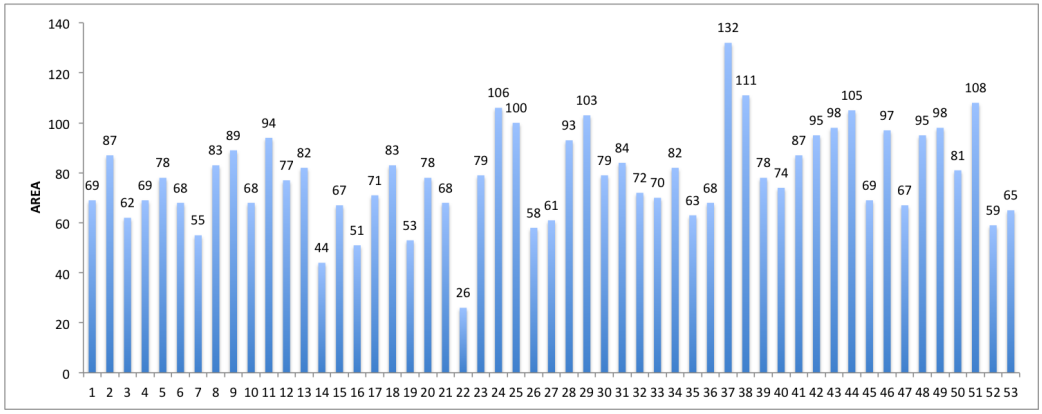

Figura 4.5: Áreas del cinetoplasto del parásito.

Los valores de área mínima y máxima establecidos fueron 45 y 400, respectivamente. Estos valores se determinaron de acuerdo a las gráficas de las Figuras 4.4, 4.5 y 4.6, en

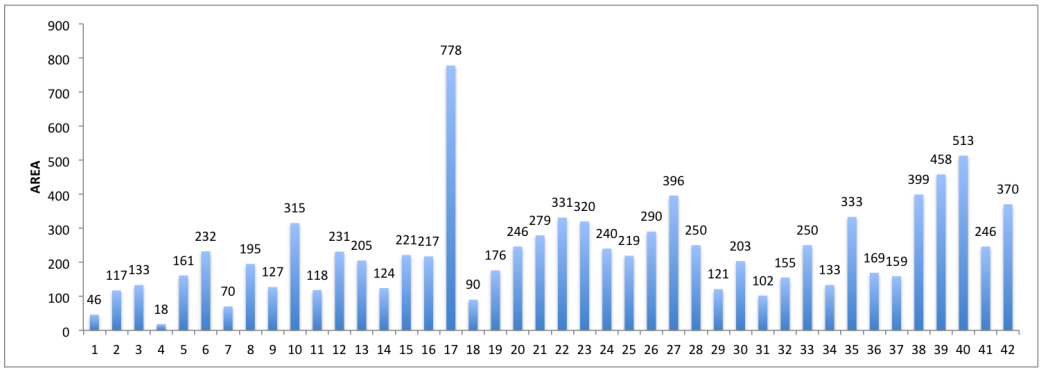

Figura 4.6: Áreas del núcleo del parásito.

donde se observa que los rangos del núcleo y cinetoplasto oscilan entre estos valores y la mayoría de los glóbulos blancos exceden este límite. La Fig. 4.3 muestra los resultados intermedios de este proceso hasta llegar a la discriminación por áreas.

En este punto es pertinente recalcar que esta segmentación únicamente encuentra una o ambas estructuras del parásito además de algunas plaquetas y otros artefactos. El resto del protozoario se pierde con el fondo, debido a que no cumple con los criterios de segmentación propuestos, como se puede apreciar en la Fig. 4.3.

4.1.2. Clasificación Gaussiana

El segundo método de segmentación empleado es un clasificador Gaussiano [12], el cual funciona como un clasificador binario en el que las clases se dividen en pixeles teñidos $\boldsymbol{p_s}$ y pixeles no teñidos $\boldsymbol{p_{ns}}$, donde los pixeles tienen la forma descrita en (4.1). Este clasificador actúa sobre cada uno de los pixeles de la imagen y les asigna una de las dos clases disponibles. El clasificador emplea un modelo basado en el vector de medias y la matriz de covarianza de ambas clases, por lo que es necesario encontrar estos parámetros antes de realizar la clasificación. Para realizar esta labor se obtuvieron dos conjuntos de

imágenes. El primer conjunto contenía muestras de pixeles teñidos y el segundo estaba formado por muestras de pixeles no teñidos. El vector de medias y matriz de covarianza se obtuvieron iterando sobre este conjunto de entrenamiento. Lo anterior hace que este sea un método de aprendizaje supervisado. El vector de medias está dado por:

$$\boldsymbol{\mu_p} = \begin{bmatrix} \frac{1}{n}\sum r \\ \frac{1}{n}\sum g \\ \frac{1}{n}\sum b \end{bmatrix} \tag{4.4}$$

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. La media fue calculada para ambos conjuntos.

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

$$cov_p = E[(\boldsymbol{p} - \boldsymbol{\mu_p})(\boldsymbol{p} - \boldsymbol{\mu_p})^t] \tag{4.5}$$

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. Al igual que la media, la covarianza fue calculada para ambos conjuntos de entrenamiento. Una vez obtenidos estos parámetros, la clasificación prosigue de la siguiente manera: primero, como una etapa de preprocesameinto, la imagen de entrada es sometida a un filtro de mediana para eliminar el efecto de ruido y crear una imagen con una distribución más uniforme. Posteriormente, cada pixel $\boldsymbol{p_u}$ de la imagen es sometido al siguiente criterio de clasificación:

$$[(\boldsymbol{p_u} - \boldsymbol{\mu_s})^t cov_s^{-1}(\boldsymbol{p_u} - \boldsymbol{\mu_s})] - [(\boldsymbol{p_u} - \boldsymbol{\mu_{ns}})^t cov_{ns}^{-1}(\boldsymbol{p_u} - \boldsymbol{\mu_{ns}})] > T \tag{4.6}$$

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 clasificador compara la distancia Mahalanobis entre $\boldsymbol{p_u}$ con la distribución de pixeles teñidos respecto a la distancia de Mahalanobis entre $\boldsymbol{p_u}$ y la distribución de pixeles no teñidos. En este sentido, T indica cuanto más cerca puede estar $\boldsymbol{p_u}$ a la distribución de no teñidos, en comparación con su distancia a la distribución de teñidos y aún así seguir considerando a $\boldsymbol{p_u}$ como un pixel teñido. Si la condición establecida por la ecuación (4.6) se cumple, entonces el pixel $\boldsymbol{p_u}$ es clasificado como no teñido (fondo). El Algoritmo 6 describe este proceso.

Algoritmo 6 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 $\boldsymbol{p}$ tiene la forma descrita por la ecuación (4.1). **hacer**

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

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

3: **fin para**

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

En este proyecto se empleó un umbral $T = 0$, con lo cual el pixel será asignado a la clase que esté más cercana a él. Como resultado de esta segmentación, se obtienen regiones de interés que contienen al parásito, además de glóbulos blancos, plaquetas y artefactos correspondientes a residuos del tinte dispersos en el fondo (Fig. 4.7). Este método segmenta una mayor área del parásito, sin embargo, también segmenta mayor número de plaquetas y artefactos. Debido a que varias de estas regiones presentan rangos de áreas similares al del

parásito, no es posible discriminar por área para eliminarlas. De igual forma, este método presenta sensibilidad a manchas del tinte en la imagen incrementando el área segmentada y por tanto, perdiendo el parásito entre la mancha de tinte.

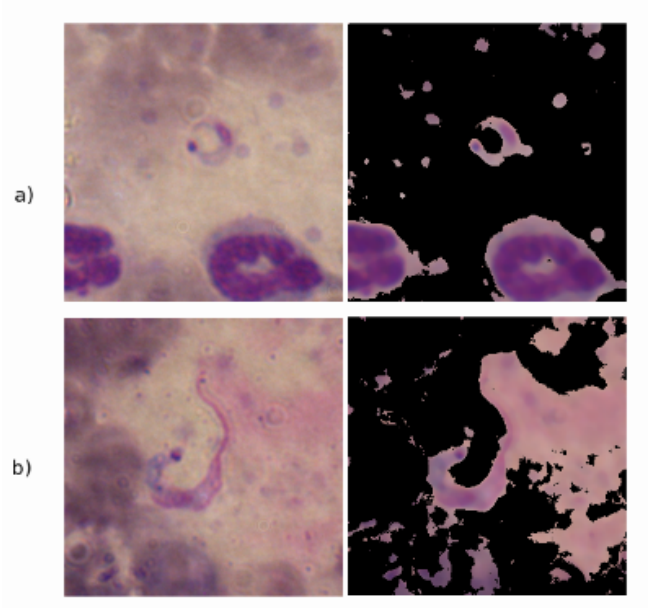

Figura 4.7: a) Imagen segmentada con el clasificador Gaussiano. b) el clasificador segmentó elementos no deseados, entre ellos el tinte disperso en la muestra. En ambas imágenes se conservó el color de los pixeles que cumplieron con los criterios de segmentación.

4.1.3. Intersección de Resultados

La etapa final combina los resultados de ambos procesos de segmentación para reducir el número de artefactos presentes en la segmentación final. El método de diferencia de canales encuentra únicamente las estructuras celulares del parásito junto con algunas plaquetas. El clasificador Gaussiano segmenta una mayor porción del parásito, pero incluye artefactos indeseables en los resultados. Sin embargo, la mayoría de estos artefactos no están presentes en los resultados de la primera segmentación. Por lo tanto, los resul-

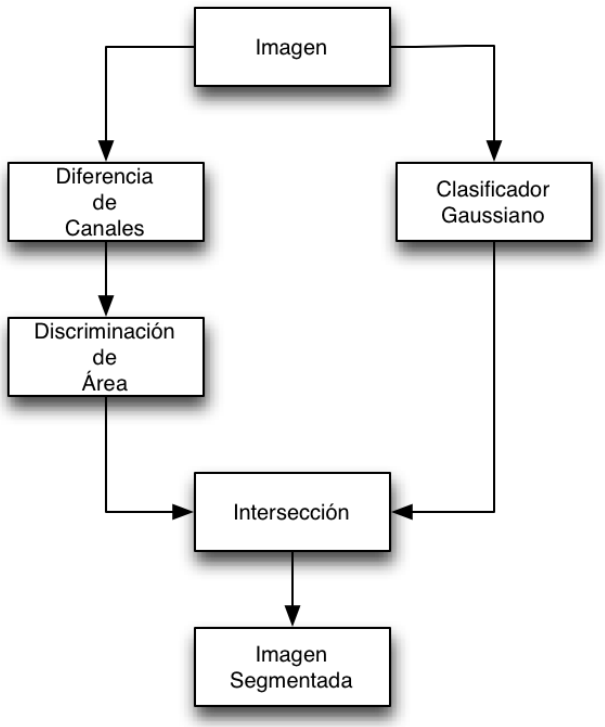

Figura 4.8: Etapas del proceso de segmentación propuesto. El resultado final está dado por la intersección de los resultados intermedios de la segmentación por diferencia de canales y la segmentación por clasificación Gaussiana.

tados del clasificador Gaussiano fueron discriminados de acuerdo al 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 intersección de los resultados de ambos métodos de segmentación. Este criterio se fundamenta en que las estructuras celulares (segmentadas por la diferencia de canales) se encuentran contenidas en el parásito (segmentado por el

clasificador Gaussiano). La Fig. 4.8 ilustra el proceso descrito en esta sección y la Fig. 4.9 presenta los resultados intermedios hasta llegar al resultado final.

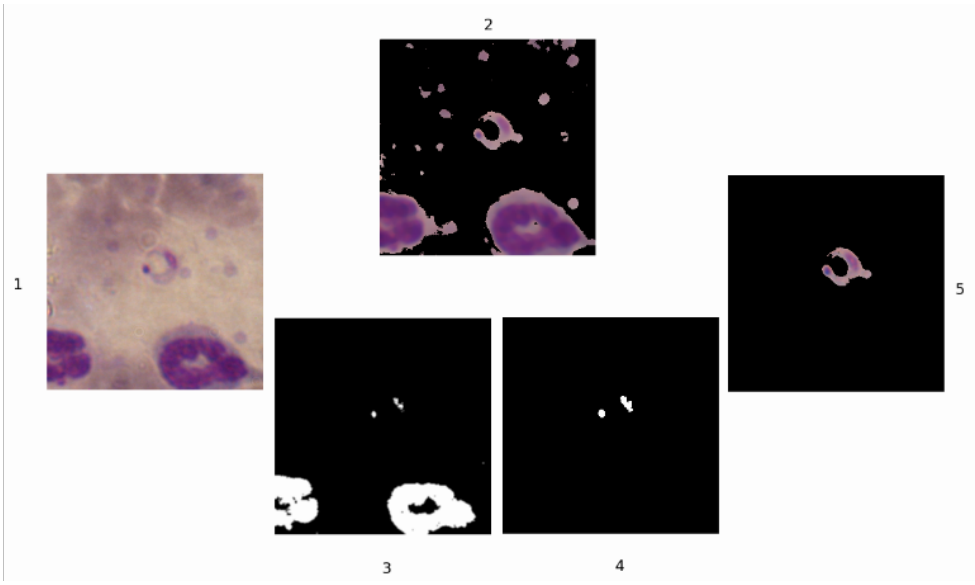

Figura 4.9: 1) La imagen de entrada. 2) Resultados del clasificador Gaussiano. 3) Resultados de la diferencia de canales. 4) Discriminación por áreas. 5) Imagen segmentada final formada por la intersección de los dos algoritmos de segmentación.

4.2. Etapa de Discriminación

La segunda etapa del proceso de detección propuesto consiste en un clasificador binario que permita realizar una discriminación final a los resultados obtenidos en la etapa de segmentación. La región de búsqueda queda definida por los pixeles pertenecientes a la intersección de las segmentaciones, es decir, la región de búsqueda está centrada en los pixeles segmentados tanto por la diferencia de canales, como por el clasificador Gaussiano.

Durante el proyecto se realizaron pruebas con dos clasificadores: los k vecinos más cercanos y AdaBoost. El algoritmo de los k vecinos más cercanos se describió en la sección 2.3.2. El histograma de la imagen en escala de grises se empleó como vector de características. Para convertir la imagen de color RGB a una imagen en escala de grises se emplearon los algoritmos disponibles en la biblioteca OpenCV (http://opencv.willowgarage.com), descrita brevemente en la sección 4.3 de este capítulo. Los pixeles en una imagen en escala de grises pueden tomar 256 valores posibles, por lo que el vector puede ser de 256 elementos. Sin embargo, para disminuir la cantidad de dimensiones de este vector se implementaron intervalos programables, de forma que cada elemento del histograma tendrá el conteo de los pixeles que entran en su rango respectivo. De acuerdo al algoritmo, se calcularon los histogramas de un conjunto de imágenes de entrenamiento y fueron almacenados para la posterior comparación de las imágenes que se clasificarán empleando el algoritmo descrito en la sección 2.3.2 con la distancia Euclidiana como métrica de comparación. Los histogramas fueron normalizados para que la suma de sus elementos sea uno. Los experimentos realizados se describen en el capítulo 5.

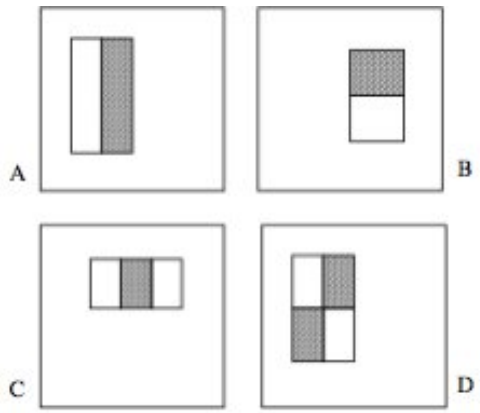

Figura 4.10: Ejemplos de características parecidas al tipo Haar. La respuesta a estas características está dada por la sumatoria de los pixeles dentro de la región clara menos la sumatoria de los pixeles dentro de la región oscura.

El algoritmo AdaBoost se describe en la sección 2.3.3. Como características descriptivas de la imagen se emplearon características parecidas al tipo Haar [27]. Éstas se muestran en la Fig. 4.10. Para calcular la respuesta de la región contenida dentro dentro de estas subventanas, se suman los pixeles contenidos dentro de la región blanca y el resultado se resta a la sumatoria de los pixeles contenidos dentro de la región oscura de la característica Haar. La ventana de trabajo, al igual que las características parecidas al tipo Haar, tiene un tamaño de 24×24 pixeles, por lo que los ejemplos de entrenamiento y los elementos a clasificar son redimensionados a estas medidas. Las respuestas pueden ser calculadas fácilmente empleando la imagen integral para realizar las sumatorias de forma rápida.

Los clasificadores débiles $h(X, f, p, \theta)$ que serán combinados para generar un clasificador robusto tienen la siguiente forma:

$$h(X, f, p, \theta) = \begin{cases} 1 & \text{si } pf(X) < p\theta \\ -1 & \text{de otra forma} \end{cases} \tag{4.7}$$

donde X es una subimagen de 24×24 en escala de grises, θ es un umbral de discriminación, f es una característica parecida al tipo Haar, $f(X)$ es la respuesta de la subimagen a esta característica y p puede ser 1 ó -1 e indica la dirección de la desigualdad. Es posible notar que si $p = 1$ la desigualdad será equivalente a evaluar $f(x) < \theta$ y si $p = -1$ la desigualdad a evaluar sería equivalente a $f(X) > \theta$. El algoritmo requiere de ejemplos positivos y negativos para entrenar el clasificador. Para generar un clasificador fuerte a partir de un conjunto de clasificadores débiles con la forma descrita por la ecuación 4.7 se implementó el procedimiento descrito por el Algoritmo 7 propuesto por Viola y Jones [27], el cual no difiere del algoritmo AdaBoost presentado en la sección 2.3.3.

Como se puede observar, el algoritmo AdaBoost requiere de un elemento de aprendizaje que genere el clasificador débil que minimice el error ponderado. Cada clasificador depende de tres variables: la característica Haar, el umbral θ y la paridad p. Las respuesta de la imagen a las características son valores numéricos. Supongamos que se tiene un número N

Algoritmo 7 Boosting empleando características parecidas al tipo Haar como clasificadores débiles.

Entrada: Conjunto de entrenamiento $(x_1, y_1) \ldots (x_n, y_n)$ donde $y_i = -1, 1$ para ejemplos negativos y positivos, respectivamente. Espacio F de características parecidas al tipo Haar que serán usadas como clasificadores débiles. Número de iteraciones It.

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

1: Inicializar la distribución $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ para $y_i = -1, 1$ los ejemplos negativos y positivos respectivamente, $w \in D$, donde m y l son el número de ejemplos negativos y positivos respectivamente.

2: **para** $t = 1, \ldots, It$ **hacer**

3: Seleccionar el mejor clasificador débil respecto al error ponderado:

$$e_t = \min_{f,p,\theta} \sum_i w_i Ind(y_i \neq h(x_i, f, p, \theta))$$

donde Ind es la función indicadora que devuelve 1 si la condición es verdadera y cero de otro modo y w_i, y_i, x_i son los valores de la distribución (peso), etiqueta y subimagen del i-ésimo ejemplo.

4: Se elige como clasificador débil $h_t(x) = h(x, f, p, \theta)$ la característica, umbral y paridad p que minimizaron el error ϵ_t.

5: $\alpha_t = 1/2 \ln(\frac{1-\epsilon_t}{\epsilon_t})$ Se determina el peso para el clasificador débil h_t.

6: $w_{t+1,i} = \frac{w_{t,i} \exp(-\alpha_t y_i h_t(\mathbf{x_i}))}{Z_i}$ Se actualiza la distribución de acuerdo al error ϵ_t. Z_i es un factor de normalización.

7: **fin para**

8: **devolver** $H(\boldsymbol{x}) = \text{sign}(\sum_{t=1}^{It} \alpha_t h_t(\boldsymbol{x}))$ donde sing(x) es la función signo que devuelve uno si $x \geq 0$ y menos uno si $x < 0$.

Algoritmo 8 Selección del clasificador débil.

Entrada: Espacio F de características parecidas al tipo Haar. Conjunto de entrenamiento τ. Vector de distribución D(pesos) de los ejemplos de entrenamiento.

Salida: h, θ, p, ϵ . Los parámetros del clasificador debil con el menor error.

1: **para todo** característica h en el espacio de características **hacer**

2: Calcular la respuesta de cada uno de los ejemplos de entrenamiento a la característica h.

3: Ordenar las respuestas de cada ejemplo.

4: **para todo** Elemento v en la lista ordenada de respuestas **hacer**

5: Se calculan cuatro sumas: La suma total T^+ de las distribuciones de los ejemplos positivos; La suma total T^- de las distribuciones de los ejemplo negativos; La sumatoria S^+ de las distribuciones de los ejemplo positivos cuyas respuestas están por debajo de v; La sumatoria S^- de las distribuciones de los ejemplo negativos cuyas respuestas están por debajo de v.

6: Tomar como error e para el clasificador h, con umbral v

$$e = \text{mín}(S^+ + (T^- - S^-), S^- + (T^+ - S^+)).$$

7: Si el primer término de la ecuación minimiza el error e tomar como paridad $p = -1$, en caso contrario tomar $p = 1$.

8: **fin para**

9: Tomar como umbral y paridad para el clasificador h, los elementos que corresponden al menor error e_{min} obtenido para h y hacer al error $\epsilon_h = e_{min}$.

10: **fin para**

11: Elegir como clasificador debil de error mínimo, a la característica h con umbral θ y paridad p que tenga el menor error ϵ_h.

12: **devolver** la característica h, el umbral θ, la paridad p y el error mínimo ϵ_h obtenido.

de imágenes de entrenamiento. Es posible ver que para cada característica Haar diferente, existen N umbrales posibles. La explicación de esto es la siguiente: Es posible encontrar la respuesta de cada ejemplo a una característica en particular. Si ordenamos estas respuestas, es posible seleccionar un umbral entre cualquiera dos respuestas consecutivas y se obtendría el conjunto de ejemplos divididos en dos clases. Ahora, dado que se tiene N respuestas, es posible escoger N umbrales. El mejor umbral para una característica determinada será aquel que divida, de la mejor forma posible, las respuestas de los ejemplo positivos y las respuestas de los ejemplo negativos. Viola y Jones proponen en [27] una forma para encontrar la paridad, el umbral y el error de cada característica. Este proceso se describe en el Algoritmo 8. Este algoritmo asume que las respuestas fueron ordenadas en orden ascendente.

El método propuesto por Viola y Jones entrena, en realidad, una cascada de clasificadores, donde cada elemento de la cascada es entrenada con los resultados de las etapas anteriores empleando AdaBoost. El clasificador en cascada está diseñado para eliminar en las primeras etapas a los elementos con menor probabilidad de pertenecer a la clase positiva mientras se concentra en aquellos elementos que son más prometedores. Durante este proyecto sólo se implementó el algoritmo de entrenamiento de los nodos de la cascada, lo cual sería equivalente a entrenar una cascada con un solo nodo.

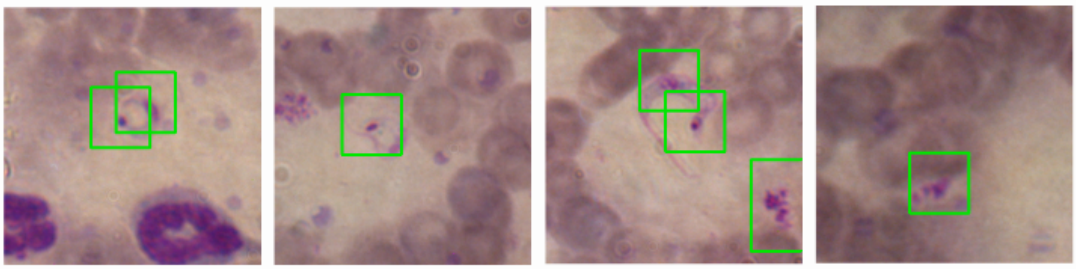

Figura 4.11: Algunos resultados de la etapa de discriminación. Se puede observar dos falsos positivos en la tercera y cuarta imagen.

Finalmente, la Fig. 4.11 presenta algunos resultados obtenidos empleando las etapas de segmentación y discriminación sobre la imagen para obtener el resultado final. El desempeño de los clasificadores descritos en esta sección (los k vecinos más cercanos y AdaBoost) como etapa de discriminación es evaluado individualmente en el capítulo 5.

4.3. 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 compara el desempeño del algoritmo AdaBoost contra el desempeño de una clasificación realizada mediante los k vecinos más cercanos. Para la realización de esta etapa se utilizaron 120 imágenes, de las cuales 60 corresponden a parásitos y 60 imágenes a fondo, glóbulos blancos y plaquetas. Se empleó el método de validación cruzada para verificar el desempeño de los algoritmos.

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 aproximadamente 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 10 iteraciones, es decir, se obtuvieron 10 subconjuntos del total de imágenes disponibles. Para cada una de las 60 imágenes positivas se obtuvo la subimagen que contiene al parásito. Para las imágenes negativas se obtuvieron 60 subimágenes que contenían, fondo, glóbulos blancos y plaquetas entre otros artefactos ajenos al parásito, entendiéndose por fondo, aquellos elementos de la imagen que no contenían ninguna célula u otro artefacto y entendiendo por artefactos, los objetos presentes en la imagen que no existen en la escena real de la cual la imagen fue adquirida. En total se obtuvieron 120 subimágenes de diversos tamaños. La Fig. 5.1 muestra algunas de las subimágenes obtenidas.

Se realizaron 3 episodios de la validación cruzada de 10 iteraciones con cada algoritmo, empleando las 120 subimágenes obtenidas para generar 10 subconjuntos de 12 imágenes en cada episodio. Cada uno de los 10 subconjuntos generados en los primeros dos episodios (referidos como episodio A y episodio B) contenían un número aleatorio de muestras positivas y negativas, es decir, el primer subconjunto podría contener 7 muestras positivas y 5 negativas mientras que el segundo subconjunto podría tener 3 muestras positivas y 9 negativas. El tercer episodio (Episodio C) contenía 10 subconjuntos compuestos por 12 subimágenes cada uno, de las cuales 6 correspondían a muestras positivas mientras que 6 correspondían a muestras negativas, seleccionadas de forma aleatoria en cada caso [15]

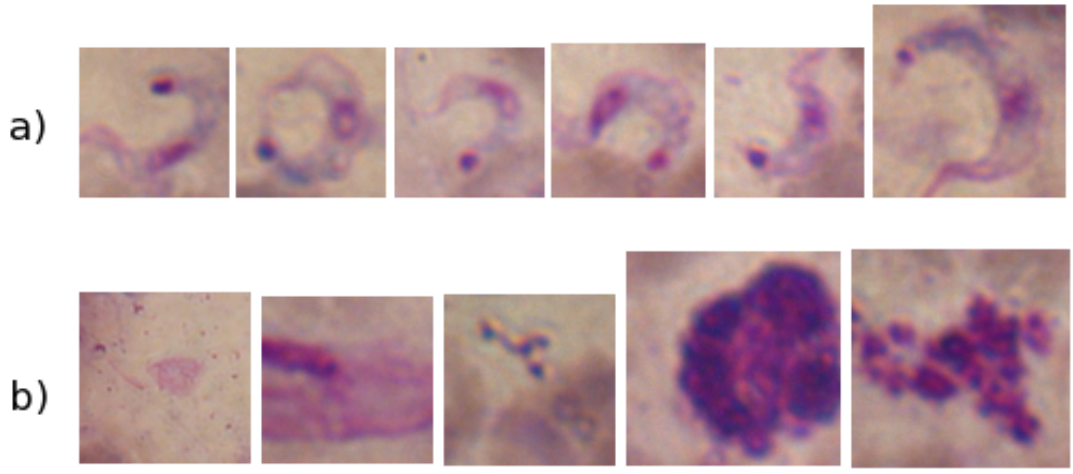

Figura 5.1: a) Ejemplos de subimágenes positivas y b) ejemplos de subimágenes negativas usadas en la validación cruzada. En total fueron empleadas 120 imágenes.

5.2. Sensibilidad y Especificidad

En cada iteración de la validación cruzada se calcularon dos valores de referencia: la sensibilidad y la especificidad [22], 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 esta 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 [22].

5.3. Etapa de Segmentación

El clasificador se compone de una etapa de segmentación y una etapa de discriminación. La segmentación representa el primer filtro que busca eliminar la mayoría de los elementos carentes de interés y conservar aquellos pixeles que pertenezcan a elementos teñidos por el colorante, entre los cuales se encuentra el parásito.

Para el desarrollo de las pruebas con los clasificadores se implementó el método de segmentación descrito en la sección 4.1 en el lenguaje de programación orientado a objetos C++. El umbral de la resta de canales fue establecido en 50, es decir, el segmentador basado en diferencia de canales únicamente deja pasar a aquellos pixeles cuya distancia entre sus canales verde y azul sea al menos de 50 unidades. Las pruebas realizadas mostraron que umbrales superiores a éste incrementan el número de artefactos. De acuerdo a la sección 4.1.1, el área mínima para que un objeto sea considerado de interés se estableció en 45 y el área máxima en 400, por lo que cualquier objeto que no se encuentre dentro de este rango será despreciado. El umbral del clasificador Gaussiano, descrito en la sección 4.1.2, se estableció en cero. Con esto, cada pixel es clasificado positivamente si su distancia de Mahalanobis a la distribución positiva es estrictamente menor que su distancia a la distribución negativa.

La Fig. 5.2 muestra algunos resultados del proceso de segmentación aplicado sobre las subimágenes que son utilizadas en la validación cruzada. Los algoritmos de clasificación operan según los resultados de esta etapa. Una vez segmentada la imagen se evalúa el resultado para determinar si existe algún objeto presente en ella. La forma de hacer esto es mediante etiquetación. Si el número de etiquetas es igual a cero, se considera que la imagen no contiene ningún elemento de interés y es descartada. En caso contrario, la imagen es evaluada por la siguiente etapa para determinar la identidad del objeto presente en ella. Las siguientes secciones describen los resultados obtenidos con dos clasificadores empleados en la etapa de discriminación.

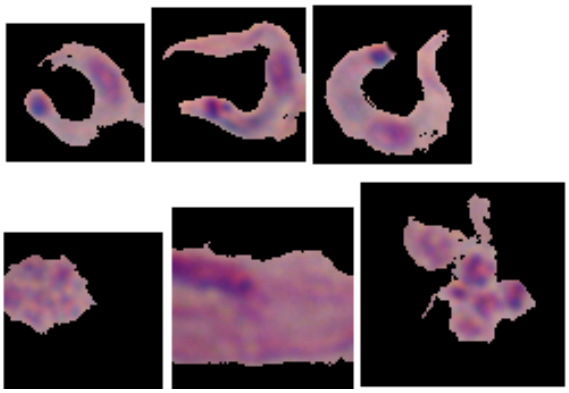

Figura 5.2: Muestras segmentadas, las imágenes superiores muestran al parásito segmentado y las inferiores corresponden a plaquetas y otros artefactos.

5.4. Etapa de Discriminación

Los resultados de la segmentación definen regiones de búsqueda que serán evaluados por la etapa de discriminación. Esta etapa busca clasificar los elementos de interés para determinar si la región analizada corresponde o no a un parásito. Como se ha mencionado

en secciones anteriores, se implementaron dos algoritmos para realizar la discriminación, estos son: los k vecinos más cercanos y AdaBoost. En esta sección se presentan los resultados obtenidos con cada uno de estos algoritmos como etapa de discriminación del proceso de clasificador propuesto. Los resultados presentados en esta sección fueron obtenidos aplicando, a cada imagen de prueba, ambas etapas del clasificador propuesto: segmentación y discriminación.

5.4.1. Los K Vecinos más Cercanos

El método de los k vecinos más cercanos realiza la discriminación de acuerdo a la clase a la que pertenezcan los vecinos mayoritarios del elemento a clasificar. Este algoritmo es descrito con mayor detalle en la sección 2.3.2. El algoritmo es usado en la etapa de discriminación de dos formas diferentes, las cuales son explicadas a continuación.

Las primeras pruebas se realizaron entrenando el algoritmo con imágenes segmentadas. Antes de calcular y almacenar los histogramas, las imágenes de entrenamiento fueron segmentadas con el método propuesto. Para realizar la comparación con el conjunto prueba se emplearon los resultados directos de la segmentación. Para entrenar el algoritmo se emplearon 108 subimágenes elegidas de forma aleatoria. El número de barras o clases del histograma fue establecido en 20 y se consideró un número de 50 vecinos, con lo cual se obtuvo un buen desempeño al repetir distintos períodos de validación cruzada con diferentes números. Las Tablas 5.1 y 5.2 muestran la sensibilidad y especificidad obtenidas durante las 10 iteraciones (It en la tabla) de los 3 episodios de la validación cruzada realizada. En este punto es pertinente recordar que los conjuntos de prueba de los episodios uno y dos contienen un número aleatorio de imágenes positivas y negativas formando 12 en total, mientras que los ejemplos de prueba del episodio 3 contienen exactamente 6 imágenes por cada clase.

Tabla 5.1: Sensibilidad del algoritmo de clasificación con los k vecinos más cercanos como método de discriminación y empleando imágenes segmentadas para el entrenamiento.

	It 1	It 2	It 3	It 4	It 5	It 6	It 7	It 8	It 9	It 10	Media
Episodio 1	0.875	1	1	1	1	1	1	1	1	1	0.987
Episodio 2	1	1	1	0.83	1	1	1	1	1	1	0.983
Episodio 3	1	0.83	1	1	1	1	1	1	1	1	0.983

Tabla 5.2: Especificidad del algoritmo de clasificación con los k vecinos más cercanos como método de discriminación y empleando imágenes segmentadas para el entrenamiento.

	It 1	It 2	It 3	It 4	It 5	It 6	It 7	It 8	It 9	It 10	Media
Episodio 1	0.8	1	1	0.57	0.778	0.75	0.75	0.71	0.833	0.88	0.808
Episodio 2	0.5	0.667	1	0.83	1	0.75	0.667	0.625	1	0.875	0.792
Episodio 3	0.5	0.83	0.83	0.667	1	0.83	0.83	1	0.83	0.667	0.8

En la segunda prueba, el algoritmo fue entrenado con las subimágenes sin segmentar. La clasificación se realizó usando la imagen de entrada, es decir, una vez que la segmentación determinaba la presencia de un elemento de interés en la subimagen, la misma subimagen original de entrada era pasada para su clasificación mediante los k vecinos más cercanos. En esta versión del algoritmo, los ejemplos de entrenamiento no sólo contenían información sobre la distribución de pixeles del parásito, como ocurría en la primera prueba, sino que también incluía información de los pixeles que rodean al mismo. Se emplearon los mismos parámetros que en la primera prueba. Los resultados sobre la sensibilidad en cada iteración de los 3 episodios se pueden apreciar en la Tabla 5.3 y los resultados de la especificidad se muestran en la Tabla 5.4.

Tabla 5.3: Sensibilidad del algoritmo de clasificación con los k vecinos más cercanos como método de discriminación y empleando imágenes sin segmentar para el entrenamiento.

	It 1	It 2	It 3	It 4	It 5	It 6	It 7	It 8	It 9	It 10	Media
Episodio 1	0.875	1	1	1	1	1	1	1	1	1	0.987
Episodio 2	1	1	1	0.83	1	1	1	1	1	1	0.983
Episodio 3	1	0.83	1	1	1	1	1	1	1	1	0.983

Tabla 5.4: Especificidad del algoritmo de clasificación con los k vecinos más cercanos como método de discriminación y empleando imágenes sin segmentar para el entrenamiento.

	It 1	It 2	It 3	It 4	It 5	It 6	It 7	It 8	It 9	It 10	Media
Episodio 1	0.8	1	1	0.57	0.778	0.75	1	0.71	1	0.88	0.85
Episodio 2	0.5	0.667	1	0.83	1	0.75	0.83	0.625	1	0.875	0.808
Episodio 3	0.5	1	0.83	0.667	1	0.83	0.83	1	1	0.667	0.833

Es posible observar que la sensibilidad para ambas pruebas se mantiene en 0.98 mientras que la especificidad es de 0.8 para el entrenamiento con segmentación, y toma valores desde 0.8 hasta 0.85 con el entrenamiento con imágenes sin segmentar. El segmentador, por si mismo, obtiene resultados similares, lo cual indica que la discriminación por los k vecinos más cercanos consigue clasificar correctamente la totalidad de las imágenes positivas que cumplieron con el criterio de segmentación sin importar el preprocesamiento realizado al conjunto de entrenamiento. En cuanto a las imágenes negativas, alrededor del 80 % son despreciadas por la segmentación. Por lo tanto existe cerca de un 20 % de las imágenes negativas que contienen elementos teñidos que cumplen con los criterios de segmentación. Los resultados de la etapa de discriminación entrenada con imágenes segmentadas no muestran mejoras respecto a los resultados obtenidos en la etapa de segmentación. Sin embargo, al usar imágenes sin segmentar en el entrenamiento, se obtiene una ligera mejoría

en la especificidad respecto a los resultados de la segmentación, alcanzando hasta un 0.85 de especificidad. Lo anterior indica que el algoritmo clasifica correctamente las imágenes positivas, sin embargo, la mayoría de los elementos segmentados que no corresponden al objeto de interés son clasificados erróneamente como positivos.

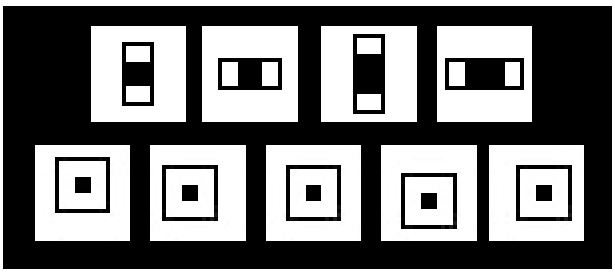

Figura 5.3: Características parecidas al tipo Haar empleadas para generar el clasificador mediante AdaBoost.

5.4.2. AdaBoost

El segundo método de discriminación empleado es un algoritmo de *boosting* conocido como AdaBoost, el cual es descrito en la sección 2.3.3. Al igual que el algoritmo de los k vecinos más cercanos, este método fue implementado en C++ y enlazado con la salida de la segmentación para la discriminación final. En el entrenamiento se usan las características parecidas al tipo Haar presentados en la Fig. 5.3. No se realiza alguna segmentación previa durante el proceso de aprendizaje. Los resultados de la segmentación no son procesados directamente por este clasificador, en lugar de eso se emplea la subimagen original de entrada en los casos donde la segmentación es exitosa. Las imágenes que no contienen elementos segmentados son despreciadas inmediatamente. El número de iteraciones del algoritmo de entrenamiento para AdaBoost se estableció en 10. La sensibilidad y la especificidad para el clasificador general con AdaBoost como etapa de discriminación se muestran en las Tablas 5.5 y 5.6 respectivamente.

Tabla 5.5: Sensibilidad del algoritmo de clasificación empleando AdaBoost como método de discriminación.

	It 1	It 2	It 3	It 4	It 5	It 6	It 7	It 8	It 9	It 10	Media
Episodio 1	0.571	0.875	0.71	0.6	0.667	0.625	1	0.8	1	1	0.785
Episodio 2	0.875	0.5	0.71	0.667	0.75	0.875	1	0.75	0.857	1	0.799
Episodio 3	0.83	0.83	0.83	0.83	0.83	0.5	0.83	0.83	1	0.83	0.817

Tabla 5.6: Especificidad del algoritmo de clasificación empleando AdaBoost como método de discriminación.

	It 1	It 2	It 3	It 4	It 5	It 6	It 7	It 8	It 9	It 10	Media
Episodio 1	1	1	1	1	1	1	1	0.857	1	1	0.986
Episodio 2	1	1	1	1	1	1	1	0.875	1	1	0.987
Episodio 3	1	1	1	1	1	1	1	1	1	1	1

AdaBoost presenta una mejora significativa en la especificidad del clasificador tomando valores cercanos a uno, esto indica que el clasificador presenta una menor cantidad de falsos positivos. Sin embargo, también presenta una disminución de la sensibilidad y por lo tanto un aumento en los falsos negativos. Aun así, los valores de sensibilidad oscilan alrededor de 0.8, lo cual representa un valor aceptable. Al contrario de lo que ocurría con el algoritmo de los k vecinos más cercanos, AdaBoost desprecia la mayoría (cerca del 100 %) de los ejemplos negativos que lograron ser segmentados por la primera etapa del clasificador propuesto. Pero también desprecia cerca del 10 % de las imágenes positivas que fueron segmentadas, lo que presenta una desventaja para este algoritmo.

6. Conclusiones y Trabajo Futuro

6.1. Conclusiones

En este trabajo se abordó el tema de detección de *Trypanosoma cruzi* en imágenes obtenidas a partir de muestras sanguíneas. El *Trypanosoma cruzi* es un protozoario parásito causante de la enfermedad de Chagas. Para esta labor se emplearon métodos de visión computacional para la extracción de información de las imágenes y métodos de aprendizaje automático para el análisis e interpretación de la información obtenida. El procedimiento planteado estuvo fundamentado en los trabajos realizados para la detección de *Plasmodium*, causante de la enfermedad malaria.

El método de clasificación propuesto siguió las etapas planteadas por la mayoría de los trabajos realizados en la detección de *Plasmodium*, los cuales son la segmentación, la obtención de un vector de características y la clasificación final de los resultados de los pasos anteriores. La mayor parte del trabajo se enfoca en los algoritmos y métodos para realizar estas tareas.

Los clasificadores basados en la distancia de Mahalanobis lograron segmentar la imagen de forma adecuada en la mayoría de los casos, sin embargo en algunas imágenes, el colorante disperso presentaba confusión con las células teñidas y era segmentada junto con

ellos. Los atributos de color que obtiene la imagen como resultado de la tinción demostraron su utilidad al momento de implementar la segmentación basada en la diferencia de los canales verde y azul de la imagen a color. La unión de dos técnicas de segmentación logró aislar al parásito del resto de la imagen, cumpliendo de este modo, con el objetivo de la segmentación.

Los k vecinos más cercanos, como método de discriminación, demostraron tener alta sensibilidad, es decir, clasifican correctamente los ejemplos positivos, sin embargo, no muestran ventaja al momento de clasificar los elementos negativos, algunos de los cuales son asignados erróneamente como positivos. AdaBoost, por otra parte, demostró ser robusto ante los ejemplos negativos, con un menor número de falsos positivos, aunque con el precio de incrementar los falsos negativos o, equivalentemente, clasificar erróneamente elementos positivos como si fueran negativos.

6.2. Trabajo Futuro

Con todo esto, fue posible verificar la factibilidad de realizar una detección automática de *Trypanosoma. cruzi*, y este trabajo de tesis marca un punto de inicio en futuras investigaciones sobre el tema. El algoritmo AdaBoost empleado corresponde a la primera etapa de entrenamiento de un clasificador compuesto por una cascada de clasificadores, cada uno entrenado con el algoritmo AdaBoost. Futuros planteamientos contemplan la implementación de la cascada de clasificadores propuesto por Viola y Jones, ademas de realizar pruebas con otros clasificadores como lo son las redes neuronales o los clasificadores basados en vector de soporte. La etapa de segmentación también puede ser mejorada para reducir los efectos del tinte disperso en la imagen. La segmentación permite establecer

zonas de búsqueda, sin embargo, la forma de recorrer esta zonas y el tamaño de la misma son dos factores que se deben considerar en futuros trabajos.

Los trabajos realizados utilizaron un total de 120 imágenes dividas en 60 positivas y 60 negativas. Aún cuando los clasificadores presentan resultados aceptables con este conjunto de muestras, es necesario incrementar este número para el diseño y prueba de clasificadores más robustos. Las aplicaciones de análisis de imágenes microscópicas operan sobre muestras tratadas con colorante y las técnicas son realizadas por diferentes especialistas, por lo que es necesario asegurar la robustez del sistema ante diferentes escenarios, por lo que el primer paso, antes de realizar las mejoras a los métodos y las pruebas con otros clasificadores, es reunir una mayor cantidad de imágenes positivas y negativas que permitan entrenar y validar los algoritmos y, al mismo tiempo, contribuir a la formación de una colección de imágenes para futuras investigaciones en el área.

Anexo A

Implementación de los Algoritmos

Como parte del desarrollo del proyecto se implementaron los algoritmos en el lenguaje de programación C++. Esta sección presenta el código de algunos de los principales métodos implementados. Los algoritmos fueron implementados con la ayuda de la biblioteca OpenCV.

Diferencia de Canales

```
void tsRestaCanales(CvMat* src, CvMat* dst) {
    uchar *iptr, *icols;
    uchar *dptr, *dcols;
    iptr = src->data.ptr;
    dptr = dst->data.ptr;
    for (int rows = 0; rows < src->rows;
                rows++, iptr+= src->step, dptr+= dst->step) {
        icols = iptr;
        dcols = dptr;
        for (int cols = 0; cols < src->cols;
                cols++, icols+=3, dcols++) {
            if (*icols > *(icols + 1)) {
```

```
                *dcols = *icols - *(icols + 1);
            }
            else
                *dcols = *(icols + 1) - *icols;
            if (*dcols > 50) {
                *dcols = 255;
            }
            else
                *dcols = 0;
        }
    }
}
```

Clasificación Gaussiana

```
void TsGaussClass::segmentImage(CvMat* colorMat,
        CvMat* segmentedMat) {
    uchar *ptr, *rows, *ptrSeg, *rowsSeg;
    double distStained, distNotStained;
    cvSmooth(colorMat, colorMat, CV_MEDIAN, 7, 7);
    CvMat *sample = cvCreateMat(3, 1, CV_32FC1);
    cvSet(segmentedMat, cvScalar(255));
    rows = colorMat->data.ptr;
    rowsSeg = segmentedMat->data.ptr;
    for(int row = 0; row < colorMat->rows;
        row++, rows+= colorMat->step, rowsSeg+= segmentedMat->step)
    {
```

```
        ptr = rows;
        ptrSeg = rowsSeg;
        for(int cols = 0; cols < colorMat->cols;
            cols++, ptr+= 3, ptrSeg++)
        {
            *(float*)CV_MAT_ELEM_PTR( *sample, 0, 0) =
                                    (float)(*ptr);
            *(float*)CV_MAT_ELEM_PTR( *sample, 1, 0) =
                                    (float)(*(ptr + 1));
            *(float*)CV_MAT_ELEM_PTR( *sample, 2, 0) =
                                    (float)(*(ptr + 2));
            distStained = cvMahalanobis(sample, this->meanPos,
                                    this->invCovPos);
            distNotStained = cvMahalanobis(sample, this->meanNeg,
                                    this->invCovNeg);
            if(distStained > distNotStained + this->threshold)
            {
                *ptrSeg = 0;


            }
        }
    }
}
```

Intersección de Resultados

```
void TsGaussClass::getInterestRegions(CvMat* colorMat,
```

```
TsCvPointArray interestPoint, CvMat* graySegmented,
CvMat* colorSegmented, TsBooleanArray &status) {
 TsImagePointer ptr1, col1, ptr2, col2,
    ptrColor1, colColor1,
    ptrColor2, colColor2;
 int intAuxiliar;
 TsIntArray tags;
 TsCvPointArrayIterator it1;
 TsIntArrayIterator it2;
 CvMat* colorAuxiliar = cvCreateMat(colorMat->rows,
     colorMat->cols, CV_8UC3);
 CvMat* grayAuxiliar = cvCreateMat(colorMat->rows,
     colorMat->cols, CV_8UC1);
 CvMat* grayILabeled = cvCreateMat(colorMat->rows,
     colorMat->cols, CV_32SC1);
 cvCopy(colorMat, colorAuxiliar);
 status.clear();
 cvSetZero(graySegmented);
 cvSetZero(colorSegmented);
 this->segmentImage(colorAuxiliar, grayAuxiliar);
 tsEtiquetaI(grayAuxiliar, grayILabeled);
 for (it1 = interestPoint.begin();
      it1 != interestPoint.end(); it1++) {
     intAuxiliar = *(int*)CV_MAT_ELEM_PTR(*grayILabeled,
      it1->y, it1->x);
     if (intAuxiliar != 0) {
         tags.push_back(intAuxiliar);
```

```
            status.push_back(true);
        }
        else
            status.push_back(false);
    }
    ptr1.i = grayILabeled->data.i;
    ptr2.ptr = graySegmented->data.ptr;
    ptrColor1.ptr = colorMat->data.ptr;
    ptrColor2.ptr = colorSegmented->data.ptr;
    for (int rows = 0; rows < grayILabeled->rows;
          rows++, ptr1.ptr+= grayILabeled->step,
          ptr2.ptr+= graySegmented->step,
          ptrColor1.ptr+= colorMat->step, ptrColor2.ptr+=
          colorSegmented->step ) {
        col1.i = ptr1.i;
        col2.ptr = ptr2.ptr;
        colColor1.ptr = ptrColor1.ptr; //Imagen de entrada a color
        colColor2.ptr = ptrColor2.ptr; //Imagen a color segmentada
        for (int cols = 0; cols < grayILabeled->cols;
            cols++, col1.i++, col2.ptr++,  colColor1.ptr+= 3,
            colColor2.ptr+= 3) {
            if (*col1.i) {
                for (it2 = tags.begin();
                      it2 != tags.end(); it2++) {
                    if (*col1.i == *it2) {
                        *col2.ptr = 255;
                        *colColor2.ptr =
```

```
                        *colColor1.ptr;
                        *(colColor2.ptr + 1) =
                        *(colColor1.ptr + 1);
                        *(colColor2.ptr + 2) =
                        *(colColor1.ptr + 2);
                        break;
                    }
                }
            }
        }
    }
    cvReleaseMat(&colorAuxiliar);
    cvReleaseMat(&grayAuxiliar);
    cvReleaseMat(&grayILabeled);
}
```

Los K Vecinos más Cercanos

```
double tsNNbClassifier::kNNb(TsFloatArray x)
{
    map<double, TsIntArray> nb;
    map<double, TsIntArray>::iterator mapit;
    std::vector<TsFloatArray>::iterator itF;
    double aux;
    int veci, poscount, negcount;
    if (this->intervals.size() == 0) {
        std::cerr << "No se ha establecido un arreglo
```

```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣de␣intervalos\n";
        return 0;
    }
    for (itF = poshist.begin(); itF != poshist.end(); itF++) {
        aux = tsArrayEuclideanDistance(x, *itF);
        nb[aux].push_back(1);
    }
    for (itF = neghist.begin(); itF != neghist.end(); itF++) {
        aux = tsArrayEuclideanDistance(x, *itF);
        nb[aux].push_back(-1);
    }
    veci = 0;
    poscount = 0;
    negcount = 0;
    for (mapit = nb.begin(); mapit != nb.end() &&
        veci < this->k; mapit++) {
        for (int i = 0; i < mapit->second.size() &&
         veci < this->k; i++) {
            if(mapit->second[i] > 0)
                poscount++;
            else
                negcount++;
            veci++;
        }
    }
    if (poscount > negcount) {
        return 1.0;
```

```
    }
    return -1.0;
}
```

La Imagen Integral

```
void tsCompIntImg(CvMat* src, CvMat* dst) {
    int Sxy = 0;
    int Sxm1y = 0;
    uchar *rows_src, *ptr_src;
    cvSet(dst, cvScalar(1));
    TsImagePointer rows_dst, ptr_dst, aux;
    rows_src = src->data.ptr;
    rows_dst.i = dst->data.i;
    for (int y = 0; y < src->rows;
                y++, rows_src+= src->step,
                rows_dst.ptr+= dst->step) {
        Sxm1y = 0;
        ptr_dst.i = rows_dst.i;
        ptr_src = rows_src;
        for (int x = 0; x < src->cols;
                  x++, ptr_src++, ptr_dst.i++) {
            Sxy = Sxm1y + (*ptr_src);
            if (y == 0)
                *ptr_dst.i = Sxy;
            else{
                aux.i = rows_dst.i;
```

```
                aux.ptr-= dst->step;
                aux.i+= x;
                *ptr_dst.i = *(aux.i) + Sxy;
            }
            Sxm1y = Sxy;
        }
    }
}
```

Selección del Clasificador Débil

```
void TsStrongClassifierManager::generateWeakClassifier
        (TsTrainingSet trainingset,
        TsFloatArray positivedistribution,
        TsFloatArray negativedistribution){

    double Tplus = 0; //Suma total pesos positivos
    double Tminus = 0;//Suma total pesos negativos

    double Splus; //Suma de ejemplos positivos por debajo
    double Sminus; //Suma de ejemplos negativos por debajo
    double e; //error
    double e1, e2; //Error por debajo y por arriba del umbral actual
    double eminimo  = 100000; //Para almacenar error minimo
    int th; //Umbral
    int haar_for_classifier;
    int parity; //paridad
```

```
int auxiliar_parity; //un auxiliar para la paridad
int yp_class; //el resultado del clasdificador h(x);
std::map<int, TsCvPointArray> responses;
TsIntArray wavelets_th; //Conjunto de umbrales para cada wavelet
TsIntArray wavelets_parities; //paridades para cada wavelet
TsFloatArrayIterator itfloat1;
TsCvPointArrayIterator itpoint1;
std::vector<TsHaarWavelet>::iterator itwavelet1;
TsIntArrayIterator itint1;
TsTrainingSetIterator ittraining1;
std::map<int, TsCvPointArray>::iterator itmap1;
std::map<int, TsCvPointArray>::iterator itmap2;

itfloat1 = positivedistribution.begin();
for (itfloat1; itfloat1 != positivedistribution.end();
             itfloat1++) {
    Tplus+= *itfloat1;
}
itfloat1 = negativedistribution.begin();
for (itfloat1; itfloat1 != negativedistribution.end();
             itfloat1++) {
    Tminus+= *itfloat1;
}
//Se asume que se calcularon las respuestas en una etapa anterior
for (int i = 0; i <  this->wavelet_space.size(); i++) {
    for (ittraining1 = trainingset.begin();
             ittraining1 != trainingset.end(); ittraining1++) {
```

```
        responses[ittraining1->haar_responses[i]].push_back
            (cvPoint(ittraining1->identifier,
            ittraining1->y_class));
    }
    for (itmap1 = responses.begin();
            itmap1 != responses.end(); itmap1++) {
        Splus = 0;
        Sminus = 0;
        for (itmap2 = responses.begin();
             itmap2 != itmap1; itmap2++) {
            itpoint1 = (itmap2->second).begin();

            for (itpoint1; itpoint1
                    != (itmap2->second).end(); itpoint1++) {

                if (itpoint1->y > 0) {
                    Splus+= positivedistribution
                            [trainingset[itpoint1
                             ->x].distribution_position];
                }
                else
                    Sminus+= negativedistribution
                    [trainingset[itpoint1
                     ->x].distribution_position];
            }

        }
```

```
            e1 = Splus + (Tminus - Sminus);
            e2 = Sminus + (Tplus - Splus);
            if (e1 < e2) {
                e = e1;
                auxiliar_parity = -1;
            }
            else{
                e = e2;
                auxiliar_parity = 1;
            }
            if (e < eminimo) {
                eminimo = e;
                th = trainingset[itmap1->
                second[0].x].haar_responses[i];

                parity = auxiliar_parity;
            }
        }
        wavelets_th.push_back(th);
        wavelets_parities.push_back(parity);
        eminimo = 10000;
        Splus = 0; //Se puede optimizar
        Sminus = 0; //Se puede optimizar
        responses.clear();
    }

    //ahora se elige el clasificador que minmice E[y != h(x)]
```

```
eminimo = 100000;
e1 = 0;
//e2 = 0;
for (int i = 0; i < this->wavelet_space.size(); i++) {
    e1 = 0;
    for (ittraining1 = trainingset.begin();
            ittraining1 != trainingset.end(); ittraining1++) {
        yp_class = (wavelets_parities[i]*(ittraining1
                    ->haar_responses[i]) <
                     wavelets_parities[i]*wavelets_th[i])?
                      1: -1;
        if (ittraining1->y_class != yp_class) {
            if (ittraining1->y_class == 1) {
                e1 += positivedistribution[ittraining1->
                    distribution_position];
            }
            else
                e1 +=  negativedistribution[ittraining1->
                    distribution_position];
        }
    }
    if (e1 < eminimo) {
        eminimo = e1;
        haar_for_classifier = i;
    }
}
```

```
    this->current_weak = this->
                wavelet_space[haar_for_classifier];
    this->current_weak_position = haar_for_classifier;
    this->current_threshold = wavelets_th[haar_for_classifier];
    this->current_parity = wavelets_parities[haar_for_classifier];
    this->current_weight = 1;
    this->current_error = eminimo;
}
```

AdaBoost

```
void TsAdaBoost::learnClassifier(int T)
{
    double et;
    double at;
    int actual_class;
    //Inicializamos los pesos como 1/2m, 1/2n para m muestras
        positivas y n muestras negativas
    for (int i = 0; i < this->num_of_class1; i++) {
        this->class1_distribution.push_back(1.0/(2*this
                ->num_of_class1));
    }
    for (int i = 0; i < this->num_of_class2; i++) {
        this->class2_distribution.push_back(1.0/(2*this
                ->num_of_class2));
    }
```

```
for (int t = 0; t < T; t++) {
    std::cout << "Training iteration " << t + 1 << std::endl;
    this->normalizeDistributions();

    this->classifier.generateWeakClassifier(this->training_set,
            this->class1_distribution,
            this->class2_distribution); //Se genera ht(x)
    et = this->classifier.getCurrentError();
    if (et > 0.5) {
        std::cout << "Error e" << t << " exedio 0.5\n";
        break;
    }

    at = 0.5*log((1 - et)/et); // El peso para ht(x)
    actual_class =
             this->classifier.getCurrentClassifierPosition();
    //Se actualiza la distribucion

    TsTrainingSetIterator it1 = this->training_set.begin();
    for (it1; it1 != this->training_set.end(); it1++) {
        if (it1->y_class > 0) {
            this->class1_distribution[it1
                    ->distribution_position] *=
                    exp(-1*at*(it1->y_class)*(this
                    ->classifier.evaluateCurrentWeakClassifier
                    (*it1)));
        }
```

```
            else
                this->class2_distribution[it1
                        ->distribution_position] *=
                        exp(-1*at*(it1->y_class)*(this
                        ->classifier.evaluateCurrentWeakClassifier
                        (*it1)));
        }
        this->classifier.setWeightForCurrent(at);
        this->classifier.pushCurrentWeakClassifier();

    }
    std::cout << "end training\n";
}
```

Bibliografía

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

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

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

[4] Chagas, C. 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* (1909), 159–218.

[5] Colunga, M. C., Siordia, O. S., and Maybank, S. J. Leukocyte recognition using em-algorithm. 2009.

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

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

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

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

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

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

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

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

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

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

[16] Rojas, Y. *Estandarización del frotis sanguíneo y su tinción para facilitar el reconocimiento microscópico de Trypanosoma cruzi (Kinetoplastida:Trpanosomatidae) en un dispositivo automatizado.* Tesis de licenciatura, Universidad Autónoma de Yucatán, 2012.

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

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

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

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

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

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

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

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

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

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

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

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

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

[30] Xiaojuan, L., and Cunshe, C. 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).