UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO

FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS

Escuela Profesional de Ingeniería Electrónica

TESIS

"Algoritmo para detección de huevos de *Trichuris trichiura* en imágenes microscópicas de muestras coprológicas - Hospital Regional de Lambayeque - 2019"

PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO ELECTRÓNICO

Investigador: Bach. Vásquez Ortiz Eduar Aníbal

Asesor: Mg. Ing. Oscar Ucchelly Romero Cortéz

Co-asesor: Dr. Heber Silva Díaz

Lambayeque, 2020

UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO

FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS

Escuela Profesional de Ingeniería Electrónica

"Algoritmo para detección de huevos de *Trichuris trichiura* en imágenes microscópicas de muestras coprológicas - Hospital Regional de Lambayeque - 2019"

Mtro. Ing. Carlos Leonardo Oblitas Vera

Presidente

Mg. Ing. Martín Augusto Nombera Lossio

Secretario

Mg. Ing. Lucía Isabel Chamán Cabrera

Vocal

Mg. Ing. Oscar Ucchelly Romero Cortéz

Asesor

Dr. Heber Silva Díaz

Co-asesor

UNIVERSIDAD NACIONAL PEDRO RUIZ GALLO
FACULTAD DE CIENCIAS FISICAS Y MATEMATICAS
DECANATO
Ciudad Universitaria - Lambayeque

ACTA DE SUSTENTACIÓN Nº 078-2019-D/FACFyM

(Sustentación Autorizada por Resolución N° 1653-2019-D/FACFyM)

En la ciudad de Lambayeque, siendo las 11:00 a.m. del día 26 de Diciembre de 2019 se reunieron en la sala de sustentaciones del laboratorio de EPLG los miembros del Jurado designados mediante Resolución Nº 1012-2019-D/FACFyM, los docentes:

Mtro. Ing. Carlos Leonardo Oblitas Vera	Presidente
Mg. Ing. Martín Augusto Nombera Lossio	Secretario
Mg. Ing. Lucía Isabel Chamán Cabrera	Vocal

Para recibir la tesis titulada:

"Algoritmo para detección de bordes de Tráchoma Folicular en Imágenes Macroscópicas de Mujeres Capacitadas en Hospitales Regionales de Lambayeque - 2019"

desarrollada por el Bachiller en Ingeniería Electrónica, **Vásquez Ortiz Eduar Anibal.**

Después de escuchar la exposición y las respuestas a las preguntas formuladas por los miembros del Jurado, se acordó APROBARLO el trabajo por UNANIMIDAD con el calificativo de MUY BUENO

En consecuencia, el Bachiller en referencia queda apto para recibir el Título Profesional de **Ingeniero Electrónico,** de acuerdo a la Ley Universitaria, el Estatuto y Reglamento de la Universidad Nacional Pedro Ruiz Gallo de Lambayeque.

Observaciones:

.....

.....

Para constancia del hecho firman.

Mtro. Ing. Carlos Leonardo Oblitas Vera
Presidente

Mg. Ing. Martín Augusto Nombera Lossio
Secretario

Mg. Ing. Lucía Isabel Chamán Cabrera
Vocal

Declaración Jurada de Originalidad

Yo, Eduar Aníbal Vásquez Ortiz, investigador principal, Mg. Ing. Oscar Ucchelly Romero Cortéz, asesor, y Dr. Heber Silva Díaz, coasesor del trabajo de investigación "Algoritmo para detección de huevos de *Trichuris Trichiura* en imágenes microscópicas de muestras coprológicas - Hospital Regional de Lambayeque - 2019" declaramos bajo juramento que este trabajo no ha sido plagiado, ni contiene datos falsos. En caso se demostrara lo contrario, asumo responsablemente la anulación de este informe y por ende el proceso administrativo al que hubiera lugar. Que puede conducir a la anulación del título o grado emitido como consecuencia de este informe.

Lambayeque, 09 de diciembre del 2019
Investigador: Bach. Eduar Aníbal Vásquez Ortiz
Asesor: Mg. Ing. Oscar Ucchelly Romero Cortéz
Co-asesor: Dr. Heber Silva Díaz

Dedicatoria

A mis padres Eduar y Gloria, y a mi hermano Víctor, quienes son mi ejemplo a seguir.

Agradecimiento

A mis padres y hermano por su apoyo incondicional a lo largo de mi vida.

Al Hospital Regional de Lambayeque por permitirme usar sus equipos y acceder a la muestra coprológica necesitada para el desarrollo del proyecto.

A Heber Silva por su asesoría en temas de microbiología.

A Oscar Romero por guiarme en el presente trabajo.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El objetivo de esta investigación fue elaborar un algoritmo de visión por computadora para detección de huevos de ***Trichuris trichiura*** *en imágenes microscópicas de muestras coprológicas con alta sensibilidad y precisión; para lograr esto se usaron 1000 imágenes, 30 % para probar el funcionamiento del algoritmo y 70 % para su aprendizaje, de 65 x 65 pixeles extraídas de 30 imágenes microscópicas de 1280 x 960 pixeles que fueron recolectadas de una sola muestra coprológica positiva procesada en solución salina; se elaboraron programas en Python utilizando librerías OpenCV, Scikit-learn, imutils, argparse, os, cPickle, Numpy y Matplotlib para obtener las subimágenes, graficar histogramas, probar y guardar el clasificador con diferentes vectores de características. El vector con mejor rendimiento fue el histograma en espacio de color HSV con 3 intervalos de matiz, 4 de saturación y 4 de brillo usando el algoritmo del vecino más cercano con métrica Manhattan y un vecino para la clasificación, llegando la sensibilidad del algoritmo al 99,35% y la precisión al 96.1%.*

Palabras clave*: Algoritmo, Trichuris trichiura, sensibilidad y precisión*

ABSTRACT

The objective of this research was to develop a computer vision algorithm for the detection of ***Trichuris trichiura*** *eggs in microscopic images of coprological samples with high recall and precision; to achieve this, 1000 images were used, 30% to test the algorithm and 70% for its learning, of 65 x 65 pixels extracted from 30 microscopic images of 1280 x 960 pixels that were collected from a single positive coprological sample processed in saline solution; Python programs were developed using OpenCV, Scikit-learn, imutils, argparse, os, cPickle, Numpy and Matplotlib libraries to obtain sub-images, graph histograms, test and save the classifier with different feature vectors. The vector with the best performance was the HSV color space histogram with 3 hue intervals, 4 saturation and 4 brightness using the algorithm of the nearest neighbor with Manhattan metric and one neighbor for classification, reaching the algorithm a recall of 99,35% and a precision of 96.1%.*

Keywords*: Algorithm, Trichuris trichiura, recall and precision*

INTRODUCCIÓN

Los médicos basan el diagnóstico de ciertas enfermedades en análisis que permiten la detección de agentes extraños en el organismo, análisis que deben ser lo más exactos posibles; sin embargo, éstos no están exentos de errores, por lo que es de suma importancia para la salud de la población encontrar métodos que los reduzcan.

Actualmente, el proceso de análisis de cualquier tipo de muestra a través de un microscopio en los hospitales se realiza mediante inspección humana, luego de haber preparado la muestra en laminillas de 22 x 22 mm (Fabián de Estrada et al, 2003). El proceso que se sigue es el siguiente: examinar las preparaciones con el microscopio con objetivos de 10x, 40x o 100x, comenzando en el ángulo superior izquierdo del cubreobjeto y desplazándose a través de toda la muestra, anotando los hallazgos que hace en cada campo microscópico (zonas en las que se divide la muestra). (Zurita, 2013)

Uno de los casos en el que se utiliza el análisis de muestras a través del microscopio es el de la detección de estructuras helmínticas parasitarias el cual se realiza mayormente con objetivo de 10x o 40x no siendo recomendable el de 100x (Fabián de Estrada et al, 2003). Una de dichas estructuras son los huevos de *Trichuris trichiura*.

Se sabe que la detección depende mucho de la habilidad del laboratorista, pudiendo verse afectada por el cansancio del mismo al tener que verificar tantos campos microscópicos; apareciendo fallas en la detección, como el confundir un elemento que no es, con una estructura parasitaria; una lista de estos elementos se puede ver en el apéndice E del Manual de Procedimientos de Laboratorio del Ministerio de Salud (Zurita, 2013), o no observar toda la muestra debido a la gran cantidad de campos. Es por ello que se necesita un método rápido y cuyo resultado no dependa de la pericia del analista.

El problema formulado fue: ¿Cómo un algoritmo de visión por computadora permite detectar huevos de *Trichuris trichiura* en imágenes microscópicas de muestras coprológicas?, siendo la hipótesis a defender: Si se elabora un algoritmo de visión por

computadora, entonces se podrá detectar huevos de *Trichuris trichiura* en imágenes microscópicas de muestras coprológicas con alto grado de sensibilidad y precisión.

Para lograr el objetivo general de elaborar un algoritmo de visión por computadora para detección de huevos de *Trichuris trichiura* en imágenes microscópicas de muestras coprológicas con alta sensibilidad y precisión; siendo los objetivos específicos:

- Procesar las imágenes microscópicas de los huevos de *Trichuris trichiura* de manera que se obtengan descriptores según su forma, color y tamaño.
- Establecer las concordancias y similitudes entre los descriptores del grupo de aprendizaje para determinar patrones de clasificación.
- Comparar los descriptores del grupo de prueba con los patrones establecidos para obtener su clasificación.
- Comparar las etiquetas dadas por el algoritmo al grupo de prueba con las de sus clases verdaderas para determinar su sensibilidad y precisión.

CAPÍTULO I

DISEÑO TEÓRICO

1.1.Antecedentes

Antecedentes Internacionales

A. Estados Unidos. "Clasificación al nivel de dermatólogo de cáncer de la piel con redes neuronales profundas" (Esteva et al, 2017)

Estudio en el cual se usó 129,450 imágenes clínicas con 2,032 diferentes enfermedades para entrenar una red neuronal convolucionada (CNN), llegando a un rendimiento que al ser comparado con el de 21 dermatólogos es casi el mismo e incluso mejor; con lo que se demuestra que estos algoritmos están a la par del desempeño de una persona por lo que sus diagnósticos son muy confiables. Se utilizó una arquitectura GoogleNet Inception v3 CNN que fue pre-entrenada con aproximadamente 1.28 millones de imágenes, y se la entrenó con el nuevo conjunto de imágenes usando aprendizaje por transferencia. Lo que buscó este estudio fue que este tipo de red neuronal sea usado en aplicaciones de smartphones y así poder proveer acceso universal de bajo costo a un rápido diagnóstico de enfermedades de la piel. El trabajo concluye que es necesaria más investigación para evaluar la performance real y poder validar esta técnica con una completa distribución de las lesiones que puedan ser encontradas.

B. Sudán. "Detección de Parásitos de Malaria usando Procesamiento Digital de Imágenes" (Bashir et al, 2017)

Estudio en el que se detecta parásitos Plasmodium y eritrocitos utilizando características de intensidad y un clasificador de red neuronal artificial entrenada con el algoritmo de propagación hacia atrás, la arquitectura del sistema consistió en seis procesos principales: adquisición, pre-procesamiento, segmentación (extrayendo 1120 sub-imágenes de eritrocitos que se usaron en el entrenamiento y prueba del sistema), extracción de las características, comparación y clasificación;

obteniéndose un 99,68% de precisión en la evaluación de la performance lo que significa que el clasificador da un buen resultado con los datos usados.

C. **Malasia. "Sistema Automatizado para el Diagnóstico de Parásitos Intestinales mediante Análisis de Imágenes Computarizadas" (Ghazali et al, 2013)**

Estudio en el cual se hace uso de un sistema de filtración con determinaciones de umbrales para la clasificación y detección de huevos de *Ascaris lumbricoides* y *Trichuris trichiura*, obteniendo una tasa de éxito de casi 93% y 94 % respectivamente en simulaciones, realizan además combinaciones diferentes de algoritmos de preprocesamiento de la imagen para así obtener el resultado más óptimo para la extracción de características. Se concluyó que el mejor detector de bordes es el método Canny y que usar un filtro de mediana dos veces da un mejor rendimiento, además la técnica empleada da resultados en segundos.

Antecedentes Nacionales

A. **"Análisis de características de forma del bacilo de Koch para detección automática de tuberculosis en imágenes digitales" (Ticona, 2017)**

En esta tesis se busca automatizar la detección de presencia de bacilos de Koch en imágenes microscópicas digitales para el diagnóstico de Tuberculosis; se emplea descriptores de forma debido a que es una de las característica más relevantes del bacilo, para ello utiliza descriptores de Fourier que permiten tener una representación numérica de la forma de un elemento. El estudio logra un acierto del 96.86% lo que implica una detección con bajo número de falsos positivos. Además logra dichos resultados con un reducido número de descriptores de Fourier (5 en total) lo que implica una alta velocidad de ejecución del programa que permite obtener resultados confiables de manera rápida.

B. **"Modelo computacional para la identificación de células espermáticas mediante el análisis automático de micrografías digitales"(Hernández, 2015)**

En esta tesis se presenta un algoritmo que permite la identificación de células espermáticas en micrografías digitales, así como, su clasificación como normales o anormales basados en la morfología de la cabeza de dichas células. Para ello se binarizó la imagen lo que permitió identificar la cabeza de las células para posteriormente hacer su respectivo análisis morfológico (largo, ancho, área,

perímetro). El estudio logró identificar un 91.5% de las células espermáticas y una tasa de acierto para la clasificación morfológica del 77.6%.

1.2.Bases teóricas

1.2.1. *Trichuris trichiura*

Huevos

- **Tamaño:** 50 μm
- **Forma:** Alargada
- **Color:** Envoltura anaranjada y contenido amarillo.
- **Otras características:** Presenta un tapón redondo y transparente en cada polo
- **Contenido:** Una masa granulosa uniforme. (Zurita, 2013) Ver Fig. 1.1.

Fig 1.1 Huevo de *Trichuris trichiura.* (Zurita, 2013)

En (Centers for Disease Control and Prevention, 2017) se define lo siguiente:

Ciclo de vida

Los huevos no embrionados se expulsan con las heces. En el suelo, los huevos pasan a una etapa de 2 células, una etapa de segmentación avanzada, y luego embrionan; los huevos se vuelven infecciosos en 15 a 30 días. Después de la ingestión (a través de manos o alimentos contaminados con el suelo), los huevos eclosionan en el intestino delgado y liberan larvas que maduran y se establecen como adultos en el colon. Los gusanos adultos (de aproximadamente 4 cm de longitud) viven en el ciego y el colon ascendente fijándose con la parte anterior en la mucosa. Las hembras comienzan a ovipositar 60 a 70 días después de la infección. Las lombrices hembras en el ciego arrojan entre 3,000 y 20,000 huevos por día. La vida útil de los adultos es de aproximadamente 1 año. Ver Fig. 1.2.

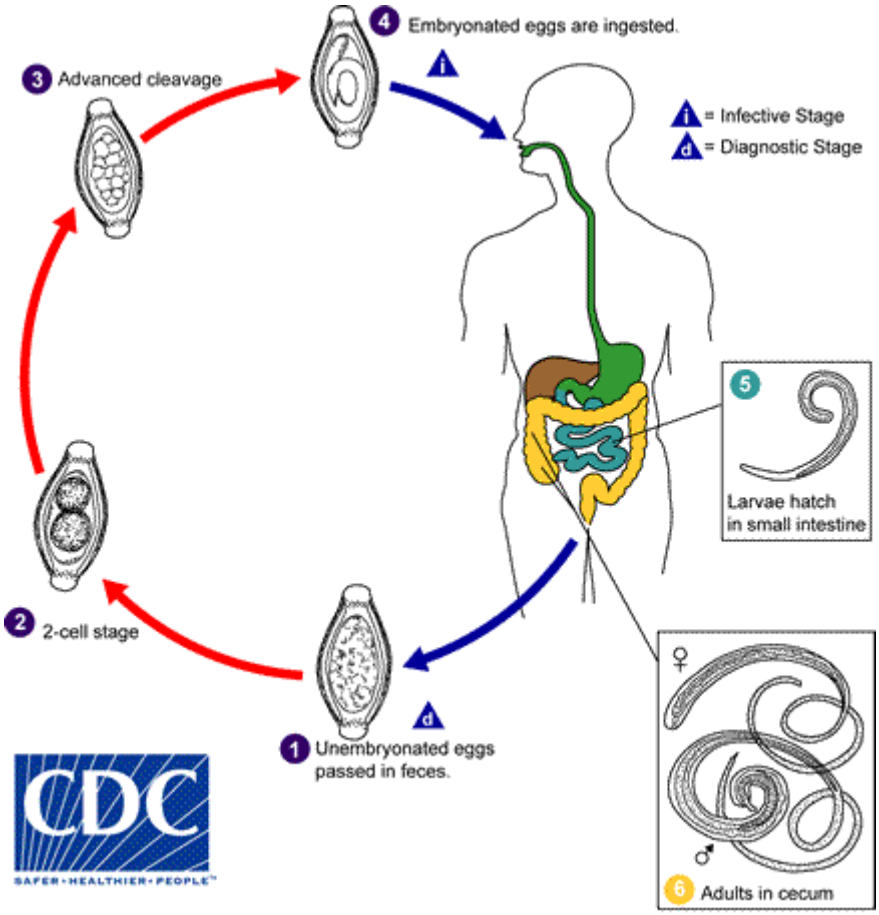

Fig 1.2 Ciclo de vida de *Trichuris trichiura (Centers for Disease Control and Prevention, 2017)*

Distribución geográfica

Es el tercer gusano redondo más común en los humanos. En todo el mundo, las infecciones son más frecuentes en áreas con clima tropical y prácticas de saneamiento deficientes, y entre los niños. Se estima que 800 millones de personas están infectadas en todo el mundo.

En un estudio sobre la prevalencia de helmintos intestinales en el Perú entre los años 1981 y 2001 para *Trichuris trichiura* se obtuvo los siguientes resultados: en 69,091 sujetos de 22 departamentos y del Callao fue 14,10% (0,25 – 85,07), en 4 provincias y una no determinada de Loreto fue 56,35% (27,50 – 59,70) y en 3 de Ayacucho 26,89% (0,73 – 42,72) en 2,741 y 412 sujetos, respectivamente, en Selva Baja 37,72% (1,70– 85,07) y Yunga 27,19% (0,90 – 65,30) en 7,019 y 7,790 sujetos, respectivamente; en 15 497 sujetos de población general 18,85% (0,37 – 85,07) y en 418 manipuladores de alimentos 18,09% (8,18 – 28,00). (Cabrera, 2003)

Presentación clínica

Con mayor frecuencia asintomática. Las infecciones graves, especialmente en niños pequeños, pueden causar problemas gastrointestinales (dolor abdominal, diarrea, prolapso rectal) y posiblemente retraso del crecimiento.

Diagnóstico de laboratorio

La identificación microscópica de los huevos de lombriz en las heces es evidencia de infección. Debido a que los huevos pueden ser difíciles de encontrar en infecciones leves, se recomienda un procedimiento de concentración. Debido a que la gravedad de los síntomas depende de la carga del gusano, la cuantificación de este último (por ejemplo, con la técnica de Kato-Katz) puede resultar útil.

El examen de la mucosa rectal por proctoscopia (o directamente en caso de prolapsos) ocasionalmente puede demostrar gusanos adultos.

1.2.2. Algoritmo

Un algoritmo es una serie de pasos organizados, que describe el proceso que se debe seguir, para dar solución a un problema específico. (Fadul, 2004).

1.2.3. Procesamiento digital de imágenes

Según (Woods & Gonzales, 2007), una imagen puede definirse como una función bidimensional $f(x,y)$, donde x y y son coordenadas y la amplitud de f en cualquier par (x,y) se llama intensidad o nivel de gris de la imagen en ese punto. Cuando x, y y f son cantidades finitas y discretas, se llama a la imagen una imagen digital.

Un procesamiento de bajo nivel se caracteriza por el hecho de que ambas entradas y salidas son imágenes, uno de nivel medio se caracteriza porque sus entradas son generalmente imágenes, pero sus salidas son atributos de esas imágenes, y un procesamiento de alto nivel involucra “dar sentido” a un conjunto de objetos reconocidos, como en el análisis de imágenes, e incluso llegar a realizar las funciones cognitivas normalmente asociadas con la visión.

Procesamiento multiespectral de imágenes es un área típica en la cual operaciones de vectores y matrices son usadas con frecuencia. Imágenes de color son formadas en espacio de color RGB mediante el uso de imágenes componentes de color rojo, verde y azul, cada píxel de una imagen RGB tiene 3 componentes, que pueden ser organizadas en la forma de un vector columna. Ver Fig. 1.3.

$$z = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}$$

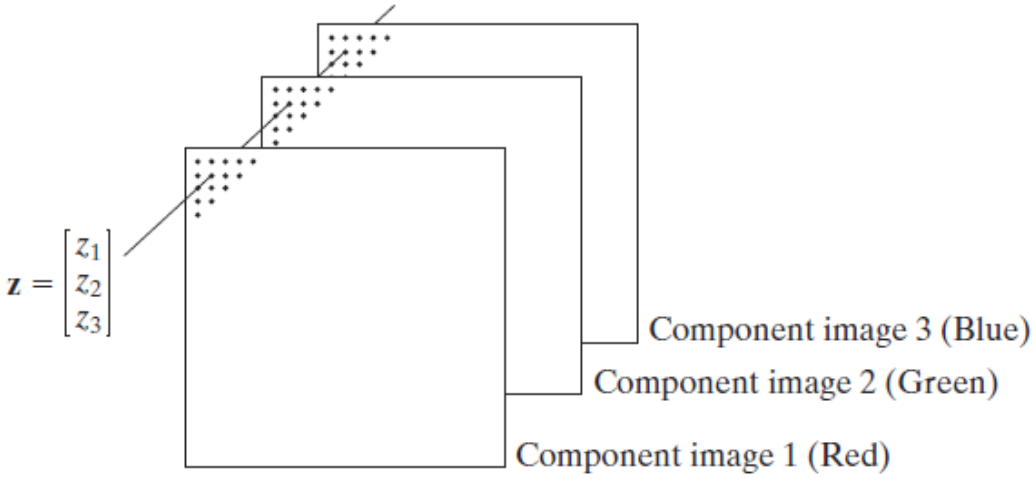

Fig 1.3 Formación de un vector a partir de los valores de píxel correspondientes en tres imágenes de componentes RGB (Woods & Gonzales, 2007)

Un dato a tener en cuenta es que no todos los programas y lenguajes para el procesamiento de imágenes utilizan el mismo orden de las componentes, por ejemplo, Matlab utiliza el orden RGB, mientras que Open CV usa BGR; esto producirá algunos cambios cuando se traslade un algoritmo desde un programa hacia otro, por lo cual se le debe prestar atención.

1.2.4. Espacio de color perceptual o HSV

En (Solomon & Breckon, 2011) se define el espacio de color perceptual como una forma alternativa de representar imágenes de color de una manera que sea más natural para la percepción humana y la comprensión del color que la representación RGB.

Los cambios dentro de este espacio de color siguen un gradiente de color perceptualmente aceptable. Desde una perspectiva de análisis de imagen, permite la separación del color de la iluminación a un mayor grado.

Los tres parámetros del espacio de color HSV (ver Fig. 1.4) se pueden interpretar de la siguiente manera:

. H (matiz / *hue*) es la longitud de onda dominante del color, por ejemplo rojo, azul, verde.

. S (saturación / *saturation*) es la "pureza" del color (en el sentido de la cantidad de luz blanca mezclada con el).

. V (valor / *value*) es el brillo del color (también conocido como luminancia).

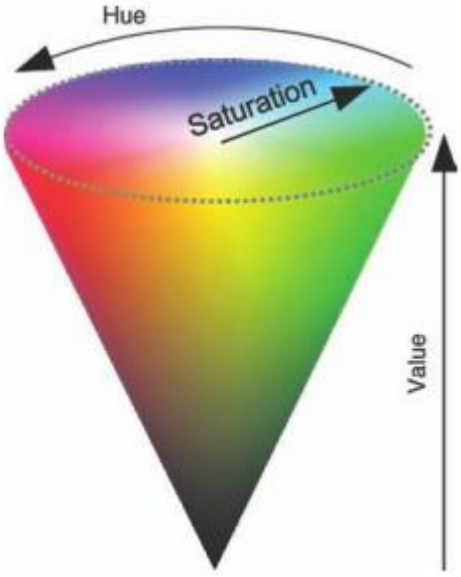

Fig 1.4 Espacio de color HSV como un cono 3-D (Solomon & Breckon, 2011)

1.2.5. Distribuciones de pixeles: histogramas

De acuerdo a (Solomon & Breckon, 2011) un histograma de imagen es un gráfico de la frecuencia relativa de ocurrencia de cada uno de los permitidos valores de píxeles en la imagen contra los valores mismos. Si normalizamos tal diagrama de frecuencia, para que la suma total de todas las entradas de frecuencia sobre el rango permitido sea uno, podemos tratar al histograma de imagen como una función de densidad de probabilidad discreta que define la posibilidad de que un valor de píxel dado ocurra dentro de la imagen. La inspección visual de un histograma de imagen puede revelar el contraste básico que está presente en la imagen y las posibles diferencias en la distribución del color de los componentes de escena de primer plano y de fondo de la imagen.

Para una imagen simple en escala de grises (ver Fig. 1.5) el histograma se puede construir simplemente contando la cantidad de veces que cada valor de escala de grises (0–255) ocurre dentro de la imagen. Cada "contenedor" (“bin”) dentro del histograma se incrementa cada vez que se encuentra su valor.

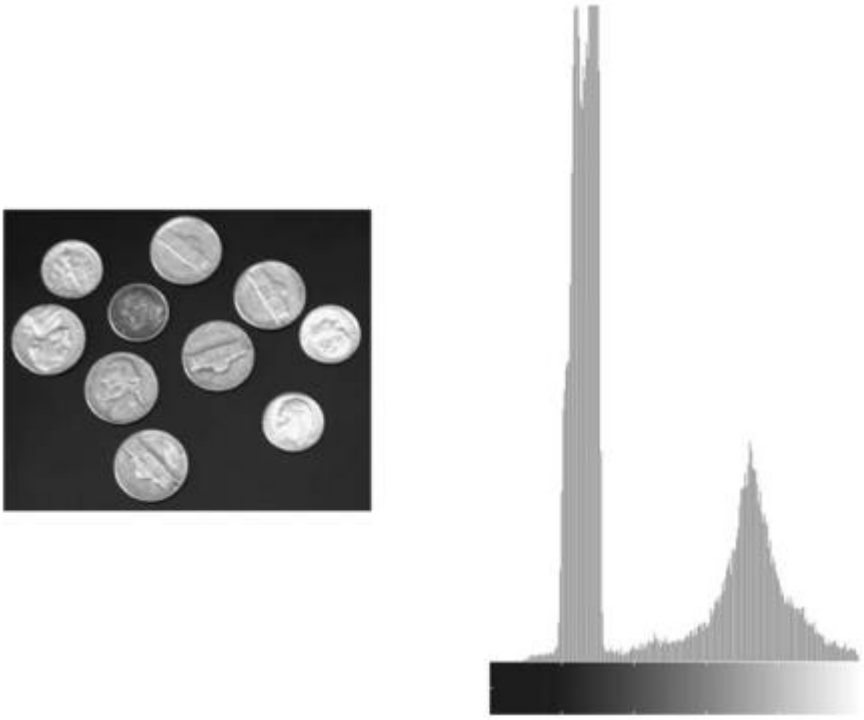

Fig 1.5 Imagen de muestra y su histograma (Solomon & Breckon, 2011)

1.2.6. Vector de características

Los objetos pueden ser comparados por la similitud basada en sus representaciones como un vector de medidas. Suponga que cada objeto es representado por exactamente *d* medidas. La coordenada í-esima de tal vector de características tiene el mismo significado para cada objeto. (Shapiro & Stockman, 2001)

1.2.7. El algoritmo del vecino más cercano

El principio del algoritmo del vecino más cercano es el de comparar patrones de imágenes de entrada con un número de paradigmas y luego clasificarlos de acuerdo a la clase del paradigma que dé la coincidencia más cercana. (Davies, 2012)

No es necesario hacer suposiciones sobre modelos para la distribución en el espacio de los vectores de características; el algoritmo usa solamente las muestras de

entrenamiento. Se calcula las distancias desde un vector de características desconocido a todas las muestras en la base de datos y se recuerda su distancia mínima. (Shapiro & Stockman, 2001)

El concepto de "distancia" en un espacio de características es flexible y admite otras definiciones aparte de la medida Euclidiana. En la Tabla 1.1 se aprecian fórmulas de tres métricas entre 2 vectores N-dimensionales x y y. (Solomon & Breckon, 2011)

Tabla 1.1
Métricas para el algoritmo del vecino más cercano

Métrica	**Fórmula**
Distancia Euclidiana	$L(x,y) = \left[\sum_{i=1}^{N}(x_i - y_i)^2\right]^{\frac{1}{2}}$
Distancia Manhattan	$L(x,y) = \sum_{i=1}^{N}\lvert x_i - y_i\rvert$
Métrica Minkowski	$L(x,y) = \left[\sum_{i=1}^{N}(x_i - y_i)^k\right]^{\frac{1}{k}}$

Nota: Tomada de (Solomon & Breckon, 2011)

Una mejor clasificación puede ser hecha examinando los k vectores de características más cercanos en la base de datos. $k > 1$ permite un mejor muestreo de la distribución de vectores de características en el espacio. En teoría, es mejor usar $k > 1$; pero, efectivamente usar un k mayor depende en tener un número más grande de muestras en cada vecindario para evitar tener que buscar muy lejos del vector desconocido por muestras. (Shapiro & Stockman, 2001).

En la Fig. 1.6, se muestra un ejemplo del uso del algoritmo para un problema de dos clases en un espacio vectorial de dos dimensiones; si k=1 entonces el patrón de prueba será de clase 1, mientras que si se aumenta el valor de k la clasificación será más compleja debido a que ambas clases están cerca una de la otra en el espacio de características.

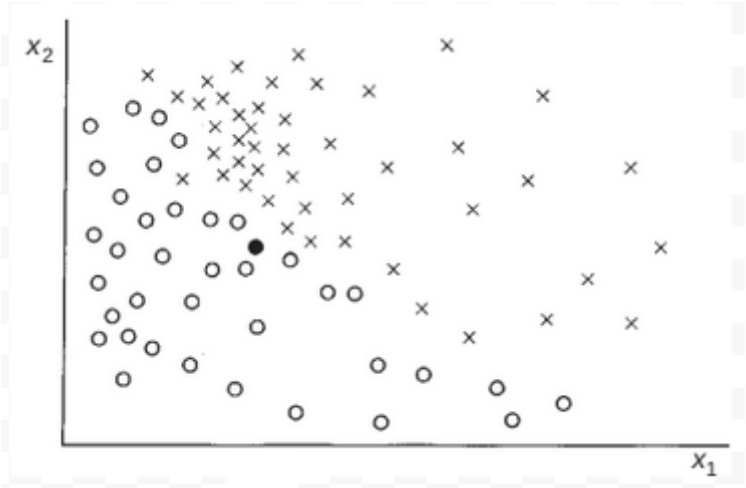

Fig 1.6 Principio del algoritmo del vecino más cercano para un problema de dos clases. ○, conjunto de patrones de la clase 1; ✕, conjunto de patrones de la clase 2; y ● patrón de prueba. (Davies, 2012)

1.2.8. Algunos términos comunes de clasificación

En (Solomon & Breckon, 2011) se definen los siguientes términos:

- **Espacio de características:** El espacio matemático abstracto (N dimensional) generado por los vectores de características utilizados en el problema de clasificación.
- **Datos de entrenamiento:** Una colección de vectores de características utilizados para construir un clasificador.
- **Datos de prueba:** Una colección de vectores de características utilizados para probar el rendimiento de un clasificador.
- **Clase de patrón:** Un término genérico que encapsula un grupo de vectores de características que comparten un origen estadístico o conceptual común.
- **Función discriminante:** Una función cuyo valor determinará la asignación a una clase u otra; típicamente, una evaluación positiva se asigna a una clase y una negativa a otra.

1.2.9. Diseño de un clasificador

De acuerdo a (Solomon & Breckon, 2011), el objetivo de la clasificación es identificar rasgos característicos, patrones o estructuras dentro de una imagen y usar estos para asignarles una clase particular. El aporte humano es esencial en el diseño y entrenamiento de clasificadores automatizados en dos áreas claves:

A. **Especificación de la tarea:** El diseñador de un sistema de clasificación necesitará decidir qué clases serán consideradas y que variables o parámetros serán importantes para lograr la clasificación.
B. **Etiquetado de clases:** el proceso de entrenamiento de un clasificador automatizado puede requerir usualmente "etiquetado manual" en la etapa inicial, un proceso en el que un usuario humano experto asigna ejemplos a clases específicas basado en propiedades seleccionadas e importantes. Esto forma parte del proceso en la generación de los llamados clasificadores supervisados.

1.2.10. Clasificación supervisada y no supervisada

Las técnicas de clasificación se pueden agrupar en dos tipos principales: supervisadas y no supervisadas. La clasificación supervisada se basa en tener patrones de ejemplo o vectores de características que ya hayan sido asignados a una clase definida. Usando una muestra de dichos vectores de características como datos de entrenamiento, se diseña un sistema de clasificación con la intención de que nuevos ejemplos de vectores de características que no se utilizaron en el diseño se clasifiquen posteriormente de manera precisa. En la clasificación supervisada, entonces, el objetivo es utilizar ejemplos de entrenamiento para diseñar un clasificador que se generalice bien a nuevos ejemplos. Por el contrario, la clasificación no supervisada no se basa en la posesión de ejemplos existentes de una clase de patrón conocida. Los ejemplos no están etiquetados y se busca identificar grupos directamente dentro del cuerpo general de datos y características que permitan distinguir un grupo de otro. (Solomon & Breckon, 2011)

1.2.11. Diseño de sistemas de clasificación

En (Solomon & Breckon, 2011) se describen los pasos a seguir para el diseño de un sistema de clasificación.

1. **Definición de clase:** Claramente, la definición de las clases es específica del problema. Por ejemplo, un sistema automatizado de procesamiento de imágenes que analice mamografías podría solamente pretender clasificar las imágenes en solo dos categorías de interés: normal y anormal. Esto sería un clasificador binario o, como a veces se le llama, un dicotomizador. Por otro

lado, un sistema más ambicioso podría intentar un diagnóstico más detallado, clasificar los escaneos en varias clases diferentes según el diagnóstico preliminar y el grado de confianza que se tiene.

2. **Exploración de datos:** En este paso, un diseñador explorará los datos para identificar posibles atributos que permitirán la discriminación entre las clases. No hay manera fija o mejor de abordar este paso, pero generalmente dependerá de un grado de intuición y sentido común. Los atributos relevantes pueden relacionarse con absolutamente cualquier propiedad de una imagen o región de imagen que podría ser útil para discriminar una clase de otra. Las medidas o atributos más comunes se basarán ampliamente en la intensidad, el color, la forma, textura o alguna mezcla de los mismos.

3. **Selección y extracción de características:** La selección de las características discriminatorias define el espacio de características. Esto, por supuesto, supone implícitamente que se ha establecido procedimientos de procesamiento de imágenes para realizar la extracción de características necesarias. En general, es crucial seleccionar características que posean dos propiedades clave. La primera es que el conjunto de características sea lo más compacto posible y la segunda que deben poseer poder discriminatorio. Un conjunto compacto de características es básicamente un conjunto pequeño y es de suma importancia, ya que un mayor número de características seleccionadas requieren una cantidad cada vez mayor de muestras de entrenamiento para entrenar eficazmente al clasificador. Segundo, obviamente tiene sentido seleccionar atributos cuya distribución sobre las clases definidas está tan ampliamente separada como sea posible y que el conjunto seleccionado de atributos deba ser estadísticamente independientes entre sí. Un conjunto de atributos que posean estas dos últimas propiedades tendría un poder de discriminación máximo.

4. **Construir el clasificador usando los datos de entrenamiento:** La etapa de entrenamiento primero requiere que se encuentre una muestra de ejemplos que se pueda asignar de manera confiable a cada una de las clases seleccionadas. Asumiendo que se han seleccionado N funciones que se han anticipado serán suficientes para lograr la tarea de discriminación y, por lo

tanto, de clasificación. Para cada ejemplo de entrenamiento por lo tanto, se registran las mediciones en las N características seleccionadas como los elementos de un vector de características N-dimensional $\mathbf{x}= [x_1, x_2, \ldots, x_N]$. De esta manera, todos los ejemplos de entrenamiento proporcionan vectores de características que pueden considerarse que ocupan una ubicación específica en un espacio de características N-dimensional. Si la selección de características se ha realizado juiciosamente, entonces los vectores de características para cada clase de interés formarán más o menos distintos conglomerados o grupos de puntos en el espacio de características.

5. **Probar el clasificador:** El rendimiento del clasificador debe evaluarse en una nueva muestra de vectores de características para ver qué tan bien puede generalizarse a nuevos ejemplos. Si el rendimiento es insatisfactorio (lo que constituye un rendimiento insatisfactorio es obviamente dependiente de la aplicación), el diseñador volverá al tablero de dibujo, considerando si las clases seleccionadas son adecuadas, si la selección de características debe ser reconsiderada o si el algoritmo de clasificación en sí mismo necesita ser revisado. La totalidad del procedimiento se repetiría hasta lograr un rendimiento satisfactorio.

1.2.12. Evaluación del error de un sistema de clasificación

Para (Shapiro & Stockman, 2001), la tasa de error de un sistema de clasificación es una medida de que tan bien el sistema resuelve el problema para el cual ha sido diseñado. El rendimiento está determinado por los errores y los rechazos hechos; clasificar todas las entradas dentro de la clase de rechazo significa que el sistema no comete errores, pero es inservible.

- **Error de clasificación:** Ocurre cuando el sistema clasifica un objeto de entrada como clase C_i, cuando la clase verdadera es clase C_j; $i \neq j$ y $C_i \neq C_r$, siendo C_r la clase de rechazo.
- **Tasa de error empírico:** Es el número de errores hecho en datos de prueba independientes dividido por el número de intentos de clasificación.
- **Tasa de rechazo empírica**: Es el número de rechazos hechos en datos de prueba independientes dividido por el número de intentos de clasificación.

- **Datos de prueba independientes:** Son objetos de muestra con clase verdadera conocida, incluyendo objetos de la "clase de rechazo", que no fueron usados en los algoritmos de extracción de características y clasificación.
- **Falsa alarma y falso rechazo:** Consideremos que nuestro sistema sirve para determinar si un paciente tiene o no una enfermedad *D*. Si el sistema dice incorrectamente que la persona tiene la enfermedad *D* entonces el error es llamado *falsa alarma o falso positivo*; mientras que, si el sistema dice incorrectamente que la persona no tiene la enfermedad *D*, entonces el error es llamado *falso rechazo o falso negativo*.

1.2.13. Precisión y sensibilidad

De acuerdo a (Shapiro & Stockman, 2001), en la aplicación de recuperación de documentos o imágenes, , el objetivo es recuperar objetos de interés de clase C_1 y no muchos objetos sin interés de clase C_2 de acuerdo a las características proporcionadas. El rendimiento de un sistema así es caracterizado por su precisión y sensibilidad.

- **Precisión:** Es el número de documentos relevantes (verdaderos C_1) recuperados dividido entre el número total de documentos recuperados (verdaderos C_1 más falsos positivos en realidad de C_2)
- **Sensibilidad:** Es el número de documentos relevantes recuperados por el sistema dividido entre el número total de documentos relevantes en la base de datos. Equivalentemente, es el número de documentos verdaderos C_1 recuperados dividido por el total de documentos verdaderos C_1 y falsos negativos.

1.2.14. Matriz de confusión

Es comúnmente usada para reportar los resultados de los experimentos de clasificación. La Fig. 1.7. muestra un ejemplo. La entrada en la fila i, y en la columna j el número de veces que un objeto etiquetado como verdaderamente de clase i fue clasificado como clase j.

La diagonal de la matriz de confusión, donde i=j, indica lo éxitos. Con resultados de clasificación perfecta, todos los elementos fuera de la diagonal son cero. Valores altos fuera de la diagonal indican confusión entre clases y fuerzan a reconsiderar los procedimientos de extracción de características y/o el proceso de clasificación. Si se

han hecho pruebas exhaustivas, la matriz indica los tipos y tasas de errores esperados en un sistema en funcionamiento. En el ejemplo mostrado hay 1000 vectores de entrada. Tres entradas etiquetadas como 9 fueron incorrectamente calificadas como 4, mientras dos entradas etiquetadas como 4 fueron incorrectamente clasificadas como 9. En total, 25 de los vectores de entrada fueron clasificados mal. Asumiendo que los datos de prueba fueron independiente de los usados para entrenar el sistema de clasificación, se tendría una tasa de rechazo empírica de 7/1000=0,007 y una tasa de error total de 25/1000=0,025. La tasa de error para solamente los 9s, sin embargo, es 5/100=0,05. (Shapiro & Stockman, 2001)

		Salida dada por el sistema de reconocimiento de patrones de clase j										
		'0'	'1'	'2	'3'	'4'	'5'	'6'	'7'	'8'	'9'	'R'
Objeto de clase verdadera i	'0'	97	0	0	0	0	0	1	0	0	1	1
	'1'	0	98	0	0	1	0	0	1	0	0	0
	'2'	0	0	96	1	0	1	0	1	0	0	1
	'3'	0	0	2	95	0	1	0	0	1	0	1
	'4'	0	0	0	0	98	0	0	0	0	2	0
	'5'	0	0	0	1	0	97	0	0	0	0	2
	'6'	1	0	0	0	0	1	98	0	0	0	0
	'7'	0	0	1	0	0	0	0	98	0	0	1
	'8'	0	0	0	1	0	0	1	0	96	1	1
	'9'	1	0	0	0	3	0	0	0	1	95	0

Fig 1.7 Matriz de confusión hipotética para reconocimiento de dígitos. 'R' es la clase de rechazo (Shapiro & Stockman, 2001)

CAPÍTULO II

MÉTODO Y MATERIALES

2.1. Diseño de Contrastación de Hipótesis

Usando los descriptores obtenidos del procesamiento de la base de datos, se realizó el reconocimiento de patrones en el grupo de aprendizaje para poder detectar los huevos de *Trichuris trichiura*. Los resultados obtenidos por el algoritmo al clasificar el grupo de prueba fueron comparados con sus respectivas clases verdaderas, para probar así su sensibilidad y precisión.

En Fig. 2.1 se presenta la lógica a seguir para la contrastación de la hipótesis.

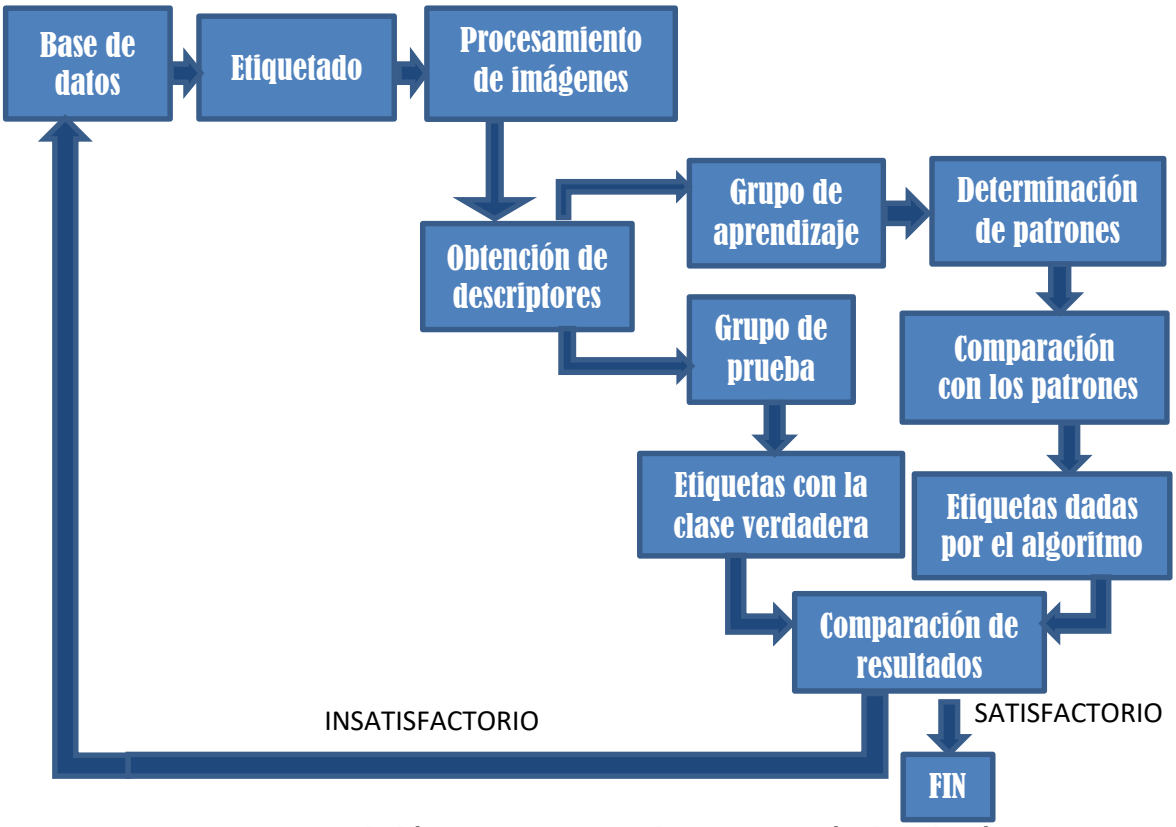

Fig 2.1 Esquema con la lógica a seguir para la contrastación de la hipótesis

2.2. Población, muestra

La población estuvo constituida por las imágenes microscópicas de las muestras coprológicas de los pacientes que se atienden en el Hospital Regional de Lambayeque las cuales fueron procesadas para detectar la presencia de huevos de *Trichuris trichiura*.

El tamaño de muestra de imágenes fue no probabilístico por conveniencia del investigador, siendo un total de 1000 imágenes de 65 x 65 pixeles (500 positivas y 500 negativas), 30 % para probar el funcionamiento del algoritmo y 70 % para su aprendizaje, obtenidas de la segmentación de 30 imágenes microscópicas de 1280 x 960 pixeles que fueron recolectadas de una sola muestra coprológica positiva procesada en solución salina.

2.3. Técnicas, instrumentos, equipos, materiales y consideraciones éticas

- **Técnicas:** Las técnicas usadas fueron de adquisición y segmentación de imágenes, extracción de histogramas del espacio de color HSV, y el algoritmo del vecino más cercano con las métricas Euclidiana y Manhattan.

- **Instrumentos:** Se utilizaron programas escritos en lenguaje Python para procesar los datos y obtener la sensibilidad y precisión del algoritmo.

- **Equipos:**

  Microscopio Biológico Binocular Olympus CX31: Dispositivo gracias al cual se analizó la muestra coprológica, dicho análisis fue realizado por el investigador con asistencia del biólogo co-asesor. Cuenta con objetivos de 4x, 10x, 40x y 100x, se utilizó el objetivo de 10x que combinado con el ocular de 10x se logró un aumento de 100 veces.

  Celular con cámara: Equipo con el cual se obtuvieron las imágenes a usar en el software, con una resolución de 4128 x 3096 píxeles, que luego fueron redimensionadas a 1280 x 960 píxeles.

  Computadora: Equipo electrónico donde se realizó la programación y se almacenó la base de datos. Fue una laptop Lenovo con sistema operativo Ubuntu 18.04 en Dual Boot, con 197,3 GB de memoria, 7,7 GB de RAM y procesador Intel Core I7.

- **Materiales:**

  Imágenes microscópicas digitales de muestras coprológicas

  Lenguaje Python: Lenguaje de programación usado para la elaboración del algoritmo en su versión 2.7.15+

  OpenCV: Biblioteca libre de visión artificial en su versión 4.0.1 (Bradski & Kaehler, 2000)

  Scikit-learn: Biblioteca para aprendizaje de máquina de software libre para lenguaje Python (Pedregosa et al, 2011) en su versión 0.20.4.

  imutils: Paquete que incluye una serie de funciones convenientes para hacer más fácil funciones de procesamiento de imágenes básicas con OpenCV en su versión 0.5.2.

  argparse: Librería para análisis de argumentos y opciones de línea de comandos, versión 1.1.

  os: Módulo que proporciona una forma portátil de utilizar la funcionalidad dependiente del sistema operativo. Se usó para leer las etiquetas de las imágenes.

  cPickle: Módulo que implementa un algoritmo en C para convertir un objeto arbitrario de Python en una serie de bytes. Se usó la versión 1.71 y sirvió para guardar el clasificador.

  Numpy: Paquete fundamental para cálculo científico con Python. Se usó para algunas operaciones matriciales, versión 1.16.5.

  Matplotlib: Librería para gráficos en 2D, versión 2.2.4. Se utilizó para plotear los histogramas.

- **Consideraciones éticas**

  Se solicitó permiso al hospital para hacer uso de su microscopio y de una muestra anónima, por lo que no se conoció la identidad del paciente. Además no se trabajó directamente con personas.

CAPÍTULO III

RESULTADOS Y DISCUSIÓN

3.1.Programa para obtención de subimágenes

Para la obtención de subimágenes, se realizó un programa en Python que permitió extraer regiones de 65 x 65 píxeles de las imágenes microscópicas obtenidas, como la de la Fig.3.1, y etiquetarlas. Se eligió ese tamaño de subimagen pues se observó que en ese espacio podía alcanzar un huevo completo. Los nombres de las subimágenes a guardar tuvieron la siguiente estructura “si.#de imagen.jpg” y “no.#de imagen.jpg”.

El programa recibía 3 argumentos: la ruta de la imagen y el número de imágenes positivas y negativas ya extraídas. Luego por medio de dos barras de desplazamiento se trasladaba a través de toda la imagen hasta donde se deseaba y, presionando las teclas “s” o “n” se guardaban las sub imágenes positivas (con huevo) (Fig. 3.2) o negativas (sin huevo) (Fig. 3.3) respectivamente; se salía del programa presionando la tecla “q”. En la figura 3.4 se aprecia la interfaz del programa.

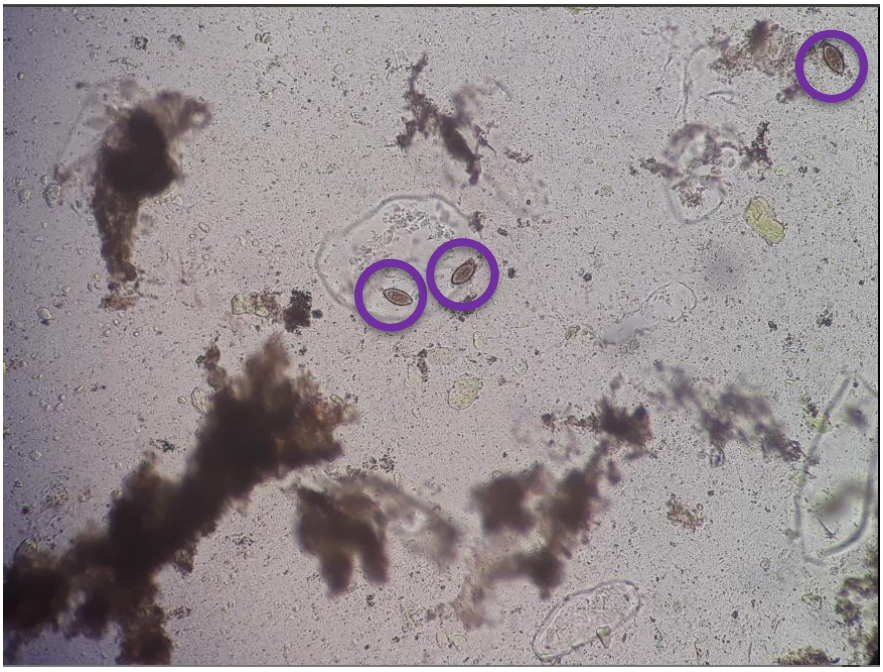

Fig 3.1 Imagen microscópica obtenida a 100x con 3 huevos de *Trichuris trichiura*

Fig 3.2 Subimagen positiva

Fig 3.3 Subimagen negativa

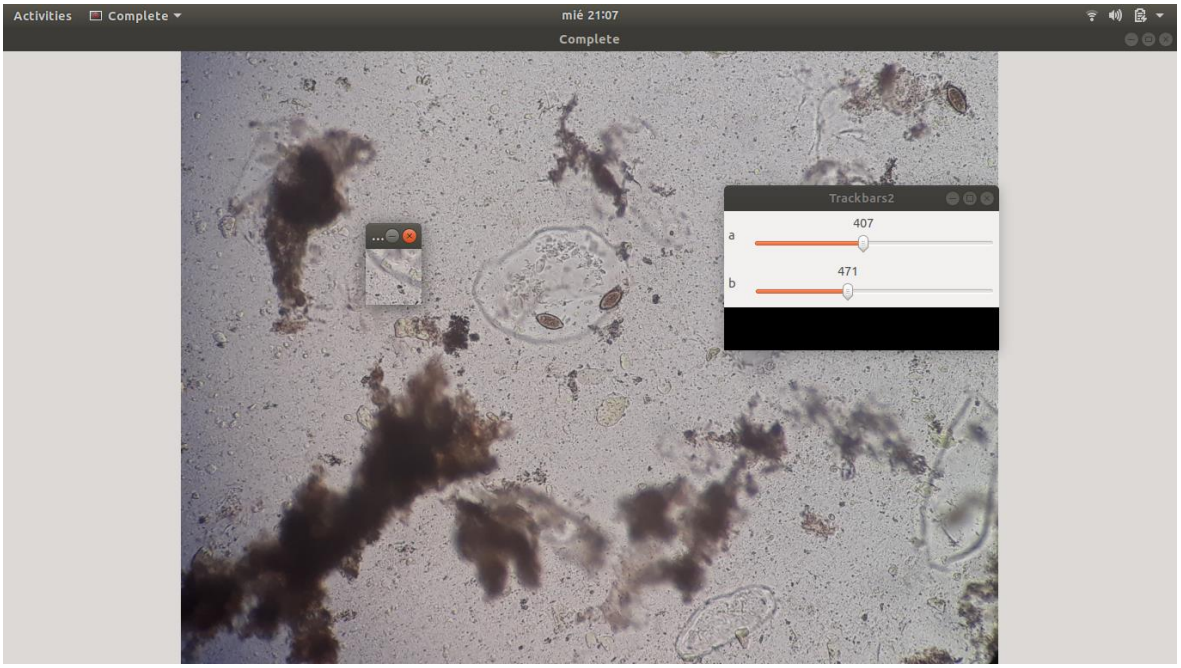

Fig 3.4 Interfaz del programa para obtención de subimagenes

Código en lenguaje Python del programa para obtención de subimágenes:

```
import cv2

import argparse

import imutils

import numpy as np
```

```
def callback(value):
        pass
def get_arguments():
        ap = argparse.ArgumentParser()
        ap.add_argument('-i', '--image', required=True, help='Ruta a la imagen')
        ap.add_argument('-s-', '--si', required=True, help='Imagenes positivas')
        ap.add_argument('-n-', '--no', required=True, help='Imagenes negativas')
        args = vars(ap.parse_args())
        return args
def main():
        args = get_arguments()
        image = cv2.imread(args['image'])
        image = imutils.resize(image, height =960)
        cv2.namedWindow('Complete', cv2.WINDOW_NORMAL)
        cv2.imshow("Complete",image)
        a=0
        b=0
        s=int(args['si'])
        n=int(args['no'])
        cv2.namedWindow("Trackbars2", 0)
        cv2.createTrackbar("a" , "Trackbars2", a, image.shape[0]-65, callback)
        cv2.createTrackbar("b" , "Trackbars2", b, image.shape[1]-65, callback)
        while True:
```

```
            a=cv2.getTrackbarPos("a" , "Trackbars2")
            b=cv2.getTrackbarPos("b" , "Trackbars2")
            img = np.array(image[a:a+65,b:b+65])
            cv2.imshow("Seccion",img)
            Key=cv2.waitKey(1)
            if Key is -1:
                continue
            else:
                if Key & 0xFF is ord('s'):
                    cv2.imwrite("train/si.%s.jpg" %(s),img)
                    s=s+1
                if Key & 0xFF is ord('n'):
                    cv2.imwrite("train/no.%s.jpg" %(n),img)
                    n=n+1
                if Key & 0xFF is ord('q'):
                    print s
                    print n
                    break
if __name__ == '__main__':
    main()
```

El programa inicia importando las librerías a usar, luego se define la función para extraer los argumentos; después se define la función principal en la cual se redimensiona la imagen, se crean las barras de desplazamiento y se extraen sus

valores, con éstos se muestra una sección de la imagen y según la tecla presionada se define que debe realizar el programa; por último se inicializa la función principal.

3.2.Conversión de histogramas en espacio HSV a vector

Se graficó los histogramas (Fig. 3.8 a 3.16) de 3 subimágenes diferentes (Fig. 3.5 a 3.7) en espacio de color HSV para determinar si es que serían un buen descriptor para la clasificación.

Fig 3.5 Subimagen A

Fig 3.6 Subimagen B

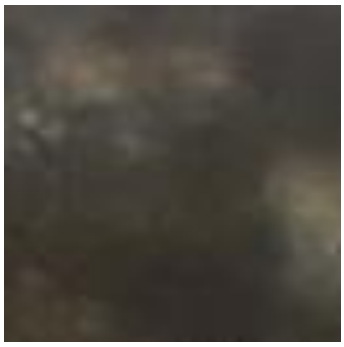

Fig 3.7 Subimagen C

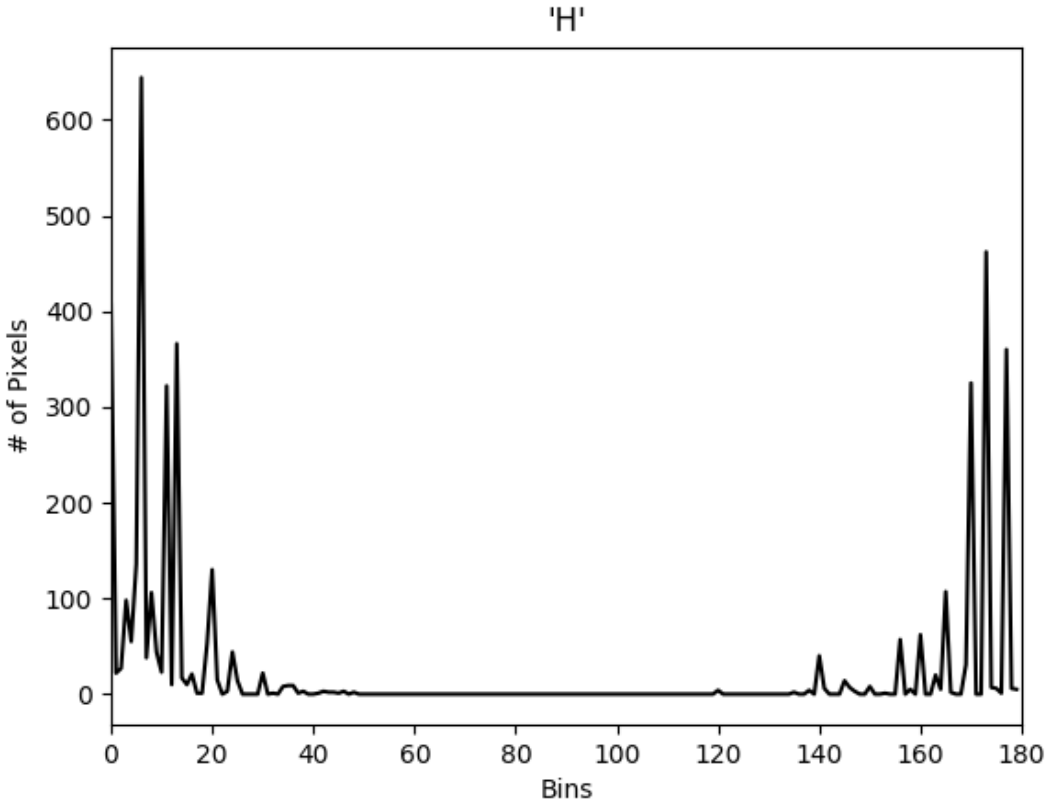

Fig 3.8 Histograma de ‘H’ de la subimagen A

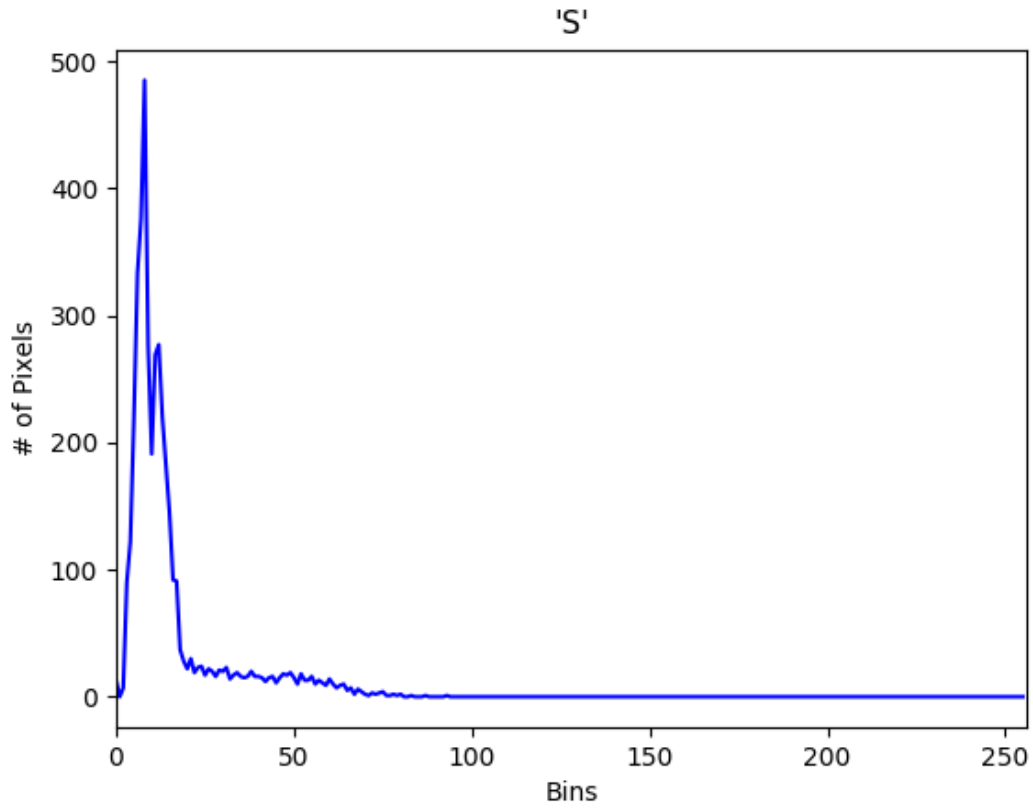

Fig 3.9 Histograma de ‘S’ de la subimagen A

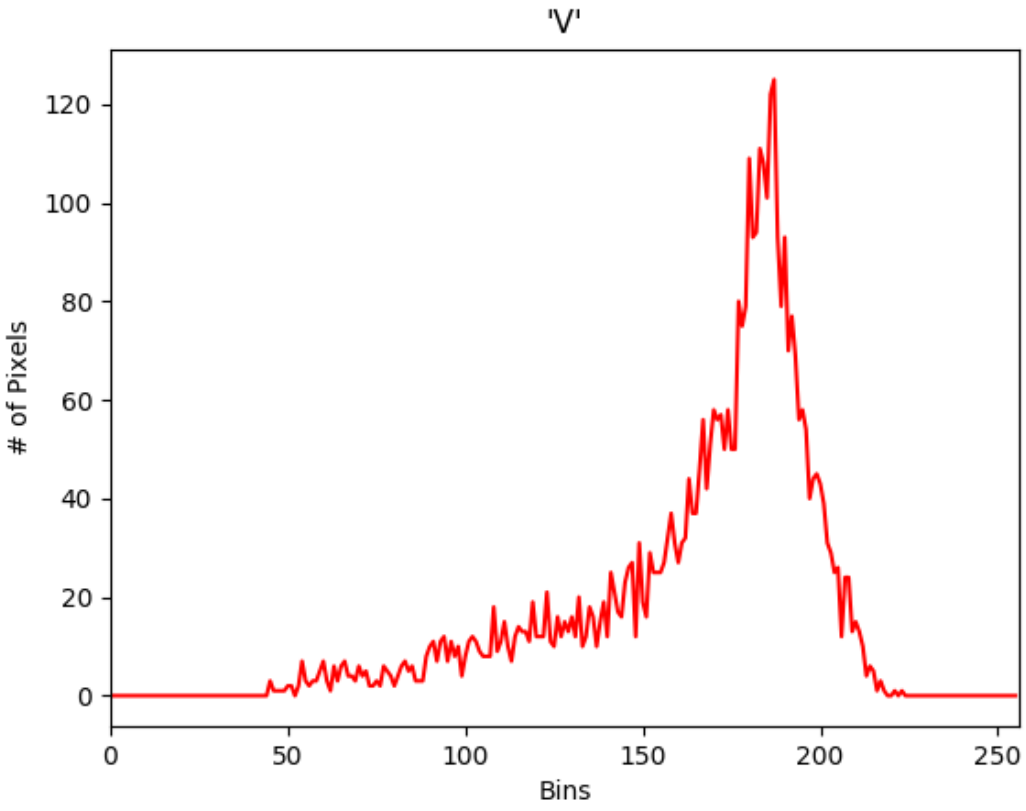

Fig 3.10 Histograma de ‘V’ de la subimagen A

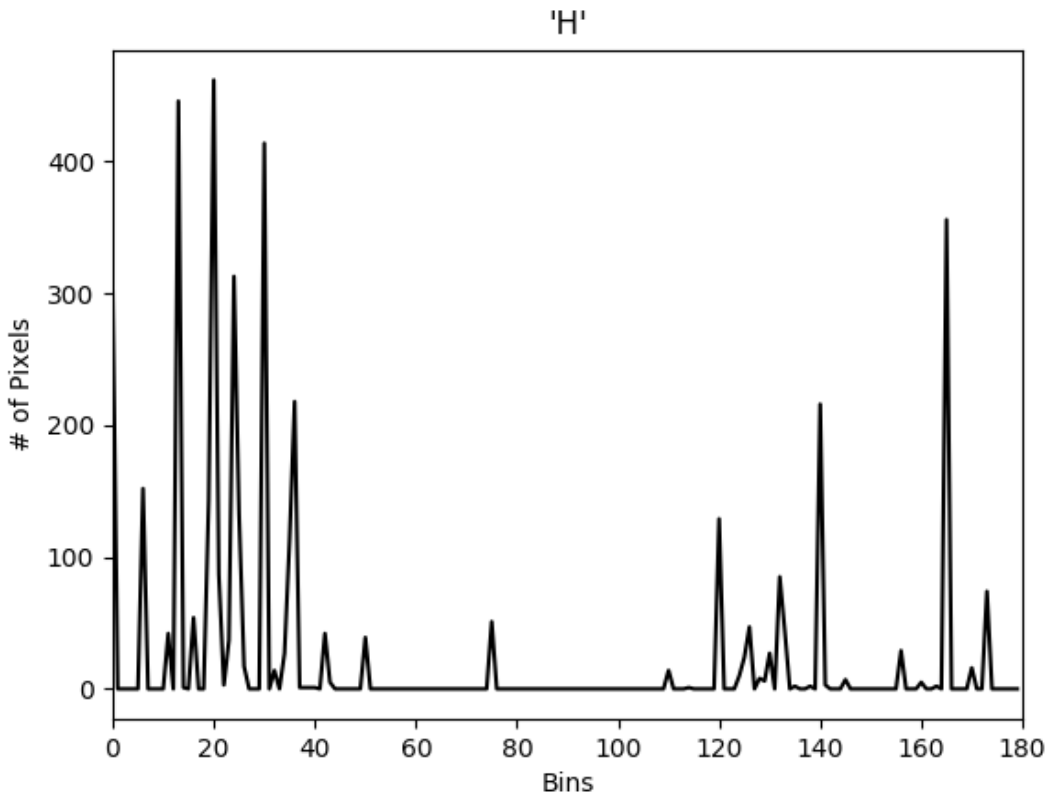

Fig 3.11 Histograma de ‘H’ de la subimagen B

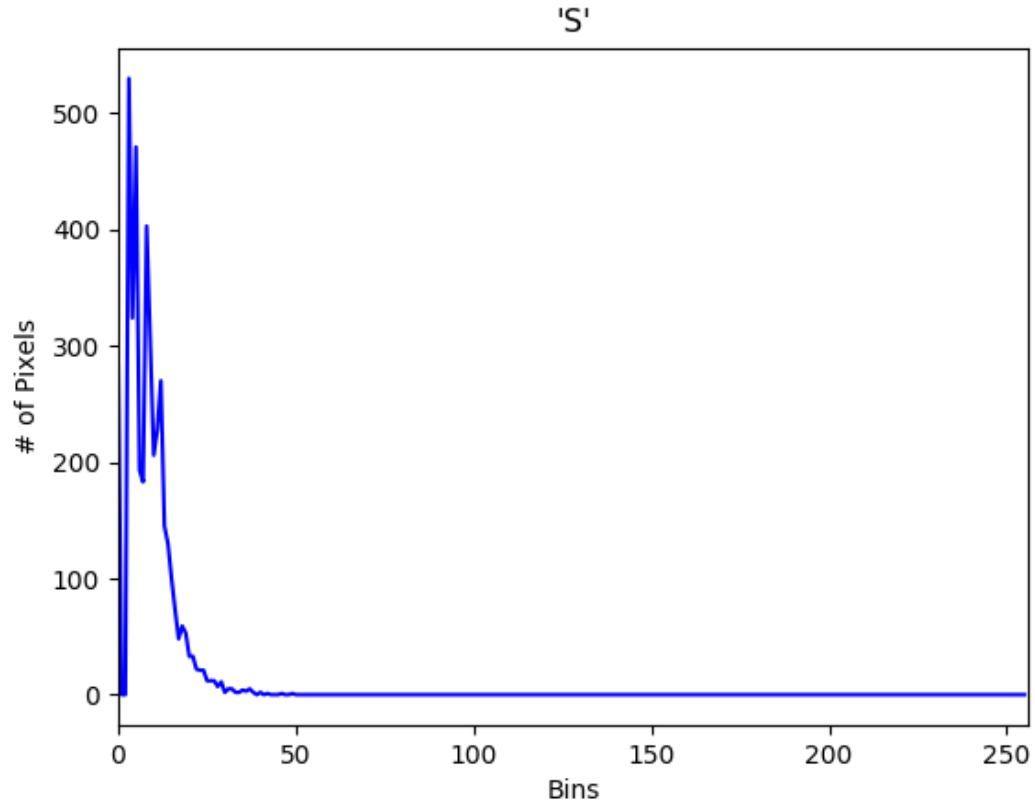

Fig 3.12 Histograma de ‘S’ de la subimagen B

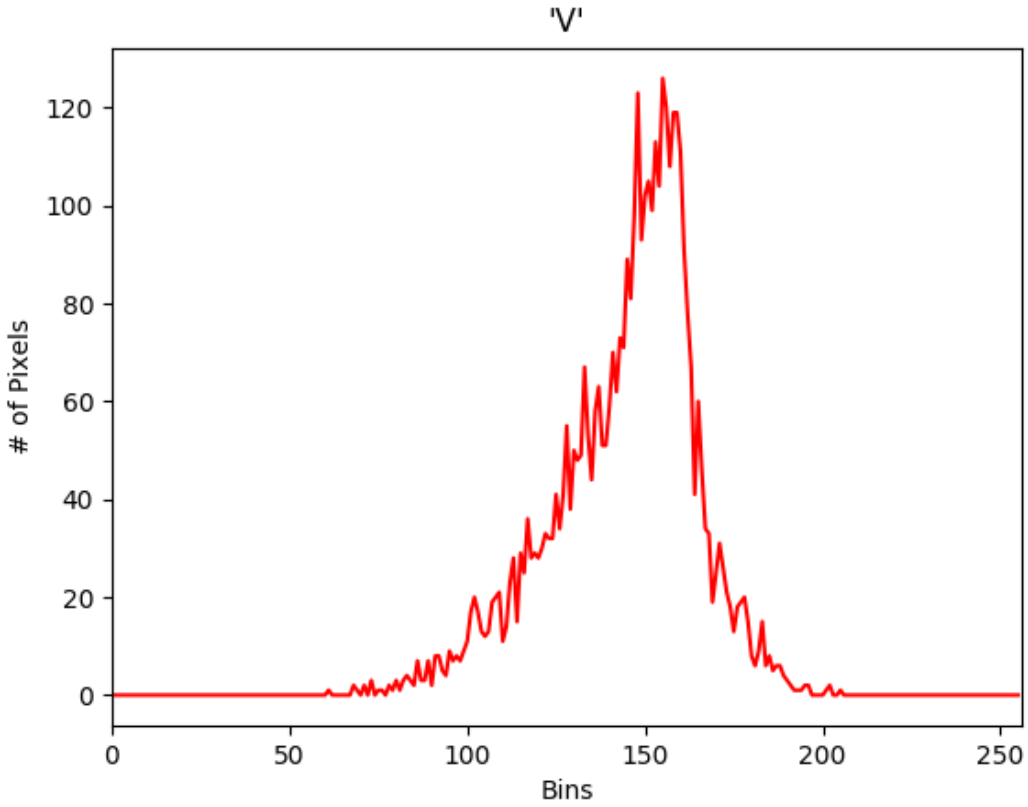

Fig 3.13 Histograma de 'V' de la subimagen B

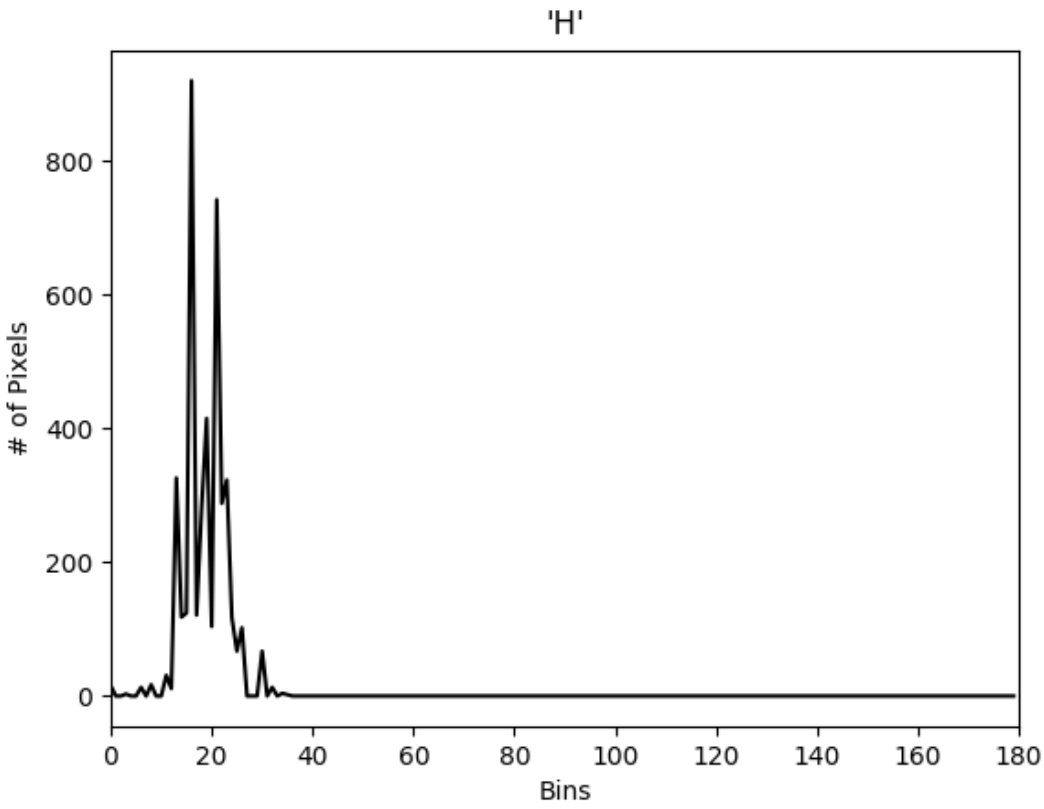

Fig 3.14 Histograma de 'H' de la subimagen C

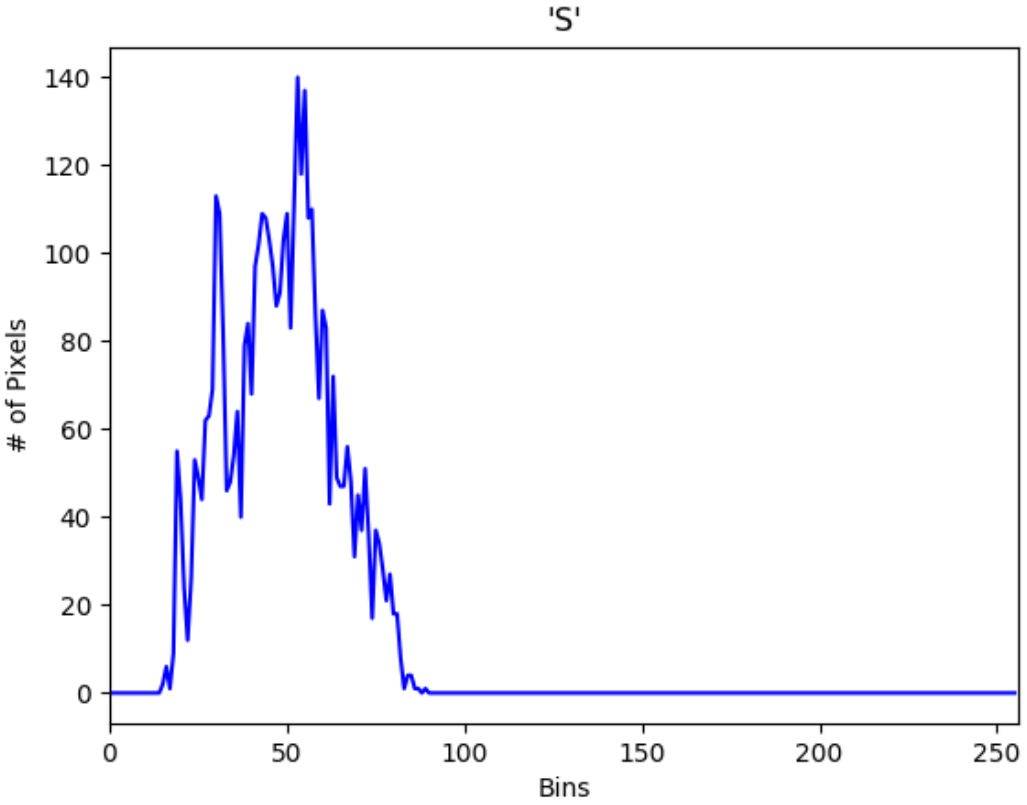

Fig 3.15 Histograma de ‘S’ de la subimagen C

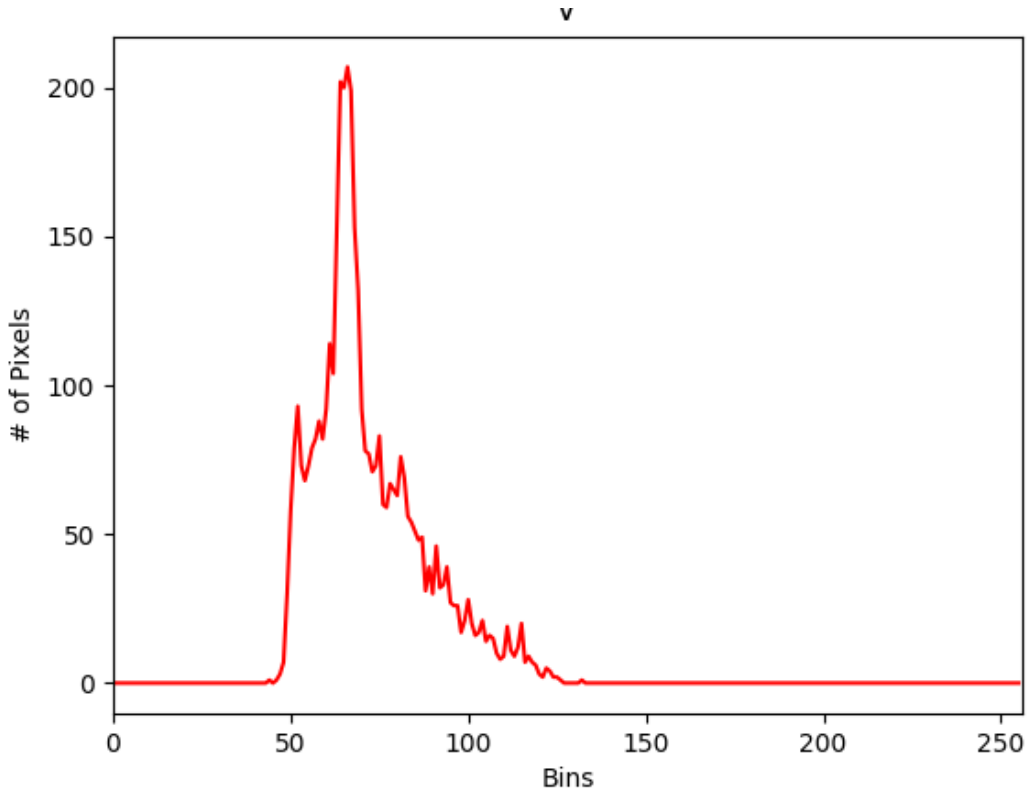

Fig 3.16 Histograma de ‘V’ de la subimagen C

Los histogramas fueron obtenidos con 180 ‘bins’ (intervalos) para ‘H’ y 256 para ‘S’ y ‘V’. Se observó que los histogramas tenían una característica particular dependiendo de la subimagen, por ejemplo el histograma de ‘S’ de la subimagen C tiene un pico entre 50 y 60, mientras que los otros dos tienen picos muy cerca de 0. Luego de esto, se realizó la conversión de los histogramas a un vector, sin embargo no se recomienda colocar todos los datos completos ya que el vector resultante tendría 11’796,480 elementos, obtenido de una matriz de 180*256*256; por lo tanto se debe disminuir el número de intervalos. Se decidió usar 8 intervalos para matiz, saturación y brillo, obteniéndose un vector de 512 elementos, que es un valor adecuado para iniciar las pruebas.

Código de la función

```
def extract_color_histogram(image, bins=(8, 8, 8)):
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    hist = cv2.calcHist([hsv], [0, 1, 2], None, bins, [0, 180, 0, 256, 0, 256])
    return hist.flatten()
```

3.3. Uso de rangos de color para obtener un descriptor:

Se elaboró un programa para determinar en qué rangos del espacio de colores HSV se encuentran los huevos, eliminando la mayor cantidad de colores sin perjudicar la forma básica de los mismos; se usaron las subimagenes A, B y C obteniéndose los siguientes rangos y las siguientes imágenes en blanco y negro (Fig. 3.17 a 3.19):

$0 < H < 17$ $22 < S < 80$ $85 < V < 170$

Fig 3.17 Imagen en B/N obtenida de A

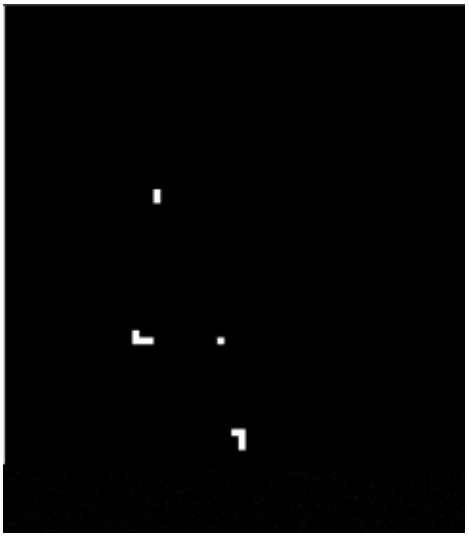

Fig 3.18 Imagen en B/N obtenida de B

Fig 3.19 Imagen en B/N obtenida de C

De esto, se obtuvieron dos vectores de características: uno de 4,225 elementos que es la nueva imagen completa en blanco y negro, y otro de 512 elementos que es el histograma de la nueva imagen sin los colores fuera de rango.

Código de la función para la imagen en blanco y negro y el nuevo histograma: Para pasar de un descriptor al otro se elimina el numeral (#) de las líneas comentadas y se comenta la penúltima línea (return mask1.flatten()).

```
def image_to_feature_vector(image):
        lower_egg = np.array([0,22,85])
        upper_egg = np.array([17,80,170])
        hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
        mask1 = cv2.inRange(hsv, lower_egg, upper_egg)
        #img2=cv2.bitwise_or(image,image,mask=mask1)
        #hist = cv2.calcHist([img2], [0, 1, 2], None,
                                (8,8,8),  [0, 180, 0, 256, 0, 256])
        return mask1.flatten()
        #return hist.flatten()
```

3.4. Sensibilidad y precisión del clasificador

Para la clasificación de las imágenes se utilizó el algoritmo del vecino más cercano con los parámetros (número de vecinos y métrica) elegidos por la función GridSearchCv de la librería Scikit-learn, que busca los más adecuados según el vector de características usado.

En la tabla 3.1 se aprecia que los valores más altos de sensibilidad y precisión se consiguieron con el histograma de la imagen completa como vector, un vecino y métrica Manhattan. La imagen en blanco y negro como vector, luego de eliminar colores fuera de rango, da como resultado valores bajos de sensibilidad y precisión, esto probablemente se deba a que el color de los huevos es variable por lo que existe el riesgo que colores fuera del rango también se encuentren dentro del huevo, dando una forma que no es buen descriptor de lo que se desea clasificar; sin embargo, el histograma de los colores dentro del rango sí da un alto valor de sensibilidad, pero la baja precisión hace que se descarte; por último, la imagen en blanco y negro como vector concatenada con el histograma de la imagen completa da valores aceptables de sensibilidad y precisión, pero no lo suficientemente buenos para la clasificación que se realiza, pues esto se traduce en posibles errores en el diagnóstico médico.

Tabla 3.1
Sensibilidad y precisión del clasificador de acuerdo al vector de características

Vector de características	# de vecinos	Métrica	Sensibilidad (%)	Precisión (%)
Imagen en Blanco y negro como vector, luego de eliminar colores fuera del rango	3	Euclidiana	57.05	69.67
Histograma de los colores dentro del rango	3	Manhattan	97.32	68.72
Histograma de la imagen completa	1	Manhattan	97.99	94.19
Imagen en B/N como vector concatenada con el histograma de la imagen completa	1	Manhattan	76.51	85.07

Habiendo elegido el histograma de la imagen completa como vector de características a usar, se procedió a probar con diferentes intervalos de matiz (H), saturación (S) y brillo (V); como se observa en la tabla 3.2.

Tabla 3.2
Sensibilidad y precisión del clasificador de acuerdo a los intervalos del histograma

Intervalos	# de vecinos	Métrica	Sensibilidad (%)	Precisión (%)
(8,8,8)	1	Manhattan	97.99	94.19
(10,10,10)	1	Manhattan	97.99	94.81
(12,12,12)	1	Manhattan	97.99	96.05
(14,14,14)	1	Manhattan	98.66	93.04
(6,6,6)	1	Manhattan	98.66	93.63
(4,4,4)	1	Manhattan	98.66	94.23
(2,2,2)	3	Manhattan	94.63	85.98
(3,3,3)	1	Manhattan	93.96	89.74
(3,3,4)	1	Manhattan	97.99	94.81
(3,4,4)	1	Manhattan	98.66	95.45

Se obtuvo el mejor rendimiento en dos combinaciones de intervalos (12,12,12) con 97.99% de sensibilidad y 96.05% de precisión, y (3,4,4) con 98.66% de sensibilidad y 95.45% de precisión. Se le dio prioridad al de mayor sensibilidad, puesto que en el diagnóstico médico conviene un bajo número de falsos negativos, además en este caso conviene tener menos valores para que el procesamiento de los datos sea más rápido y, puesto que, como se observan en las figuras 3.8 a 3.15, los histogramas tienen pocos picos altos permitiendo que con estos pocos valores se obtenga un buen descriptor de la imagen.

Por último, se realizaron diferentes pruebas agrupando de diferentes maneras las imágenes, llegándose a obtener un 99.35% de sensibilidad en una prueba y 96.1% de precisión en otra. Además en la tabla 3.3 se puede observar que la sensibilidad siempre se mantiene mayor de 97%, mientras que la precisión fluctúa entre 92% y 96%.

Tabla 3.3
Sensibilidad y precisión del clasificador con distintos datos de aprendizaje, con intervalo de (3,4,4), métrica Manhattan y un vecino más cercano.

Prueba	**Sensibilidad (%)**	**Precisión (%)**
1	98.66	95.45
2	97.37	96.1
3	99.35	93.87
4	98.72	92.22
5	98.76	93.53
6	97.44	95.00

La cantidad de subimagenes extraídas se decidió basándose en (Bashir et al, 2017), y lo obtenido en esta investigación llega a superar en resultados para *Trichuris trichiura* a la investigación realizada en Malasia (Ghazali et al, 2013).

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1.Conclusiones

1. El algoritmo de visión por computadora elaborado para detección de huevos de *Trichuris trichiura* contiene las siguientes fases: obtención de imágenes - extracción de histogramas en HSV - clasificación con el algoritmo del vecino más cercano.
2. Se procesaron imágenes microscópicas de 65 x 65 pixeles obteniéndose cuatro descriptores:

- Imagen en B/N, luego de eliminar colores fuera del rango.
- Histograma de la imagen luego de eliminar los colores fuera del rango.
- Histograma de la imagen completa.
- Imagen en B/N como vector concatenada con el histograma de la imagen completa.

3. En un espacio de características N-dimensional se ubicaron los descriptores del grupo de aprendizaje, obteniéndose patrones de clasificación.
4. La clasificación del grupo de prueba se obtuvo utilizando el algoritmo del vecino más cercano con distintos parámetros.
5. Con el histograma con 3 intervalos de matiz (H), 4 de saturación (S) y 4 de brillo (V) y el algoritmo del vecino más cercano con métrica Manhattan y un solo vecino, se obtuvo 99.35% de sensibilidad y 96.1% de precisión para el algoritmo elaborado

4.2.Recomendaciones

- Utilizar un microscopio con cámara fija.
- La toma de imágenes microscópicas se debe hacer con la misma iluminación debido a que ésta puede alterarlas.
- Aumentar la base de datos con distintas muestras coprológicas positivas y negativas.

BIBLIOGRAFÍA REFERENCIADA

1. Bashir, A., Mustafa, Z. A., Abdelhameid, I., & Ibrahem, R. (2017, January). Detection of malaria parasites using digital image processing. In *2017 International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE)* (pp. 1-5). IEEE.
2. Bradski, G., & Kaehler, A. (2000). OpenCV. *Dr. Dobb's journal of software tools*, *3*.
3. Cabrera, R. (2003). Helmintos intestinales en el Perú: análisis de la prevalencia (1981-2001). *Lima: Ministerio de Salud. Oficina General de Epidemiología*
4. Centers for Disease Control and Prevention (2017). *Trichuriasis.* Recuperado de https://www.cdc.gov/dpdx/trichuriasis/index.html
5. Davies, E. R. (2012). *Computer and machine vision: theory, algorithms, practicalities* (4th ed). Amsterdam ; Boston: Elsevier.
6. Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, *542*(7639), 115.
7. Fabián de Estrada, M., Tello Casanova, R., & Náquira Velarde, C. (2003). *Manual de Procedimientos de Laboratorio para el Diagnóstico de los Parásitos Intestinales del Hombre*.
8. Fadul, A. O. (2004). Diseño Estructurado de Algoritmos. Colombia: Sincelejo.
9. Ghazali, K. H., Hadi, R. S., & Mohamed, Z. (2013). Automated system for diagnosis intestinal parasites by computerized image analysis. *Modern Applied Science*, *7*(5), 98.
10. Hernández Bretón, H. (2015). Modelo computacional para la identificación de células espermáticas mediante el análisis automático de micrografías digitales.
11. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. *Journal of machine learning research*, *12*(Oct), 2825-2830.
12. Shapiro, L. G., & Stockman, G. C. (2001). *Computer Vision*. Upper Saddle River, NJ: Pearson.

13. Solomon, C., & Breckon, T. (2011). *Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab* (1st edition). Chichester, West Sussex ; Hoboken, NJ: Wiley.
14. Ticona Huaroto, J. E. (2017). Análisis de características de forma del bacilo de koch para detección automática de tuberculosis en imágenes digitales.
15. Woods, R., & Gonzales, R. (2007). *Digital Image Processing*. Upper Saddle River, N.J: Prentice Hall.
16. Zurita Macalupú, S. (2013). *Procedimientos de laboratorio: Laboratorios locales I - Laboratorios locales II* (2nd ed.).

ANEXOS

Fig A.1 Microscopio Olympus CX31

Fig A.2 Carnet de Investigador del Hospital Regional de Lambayeque

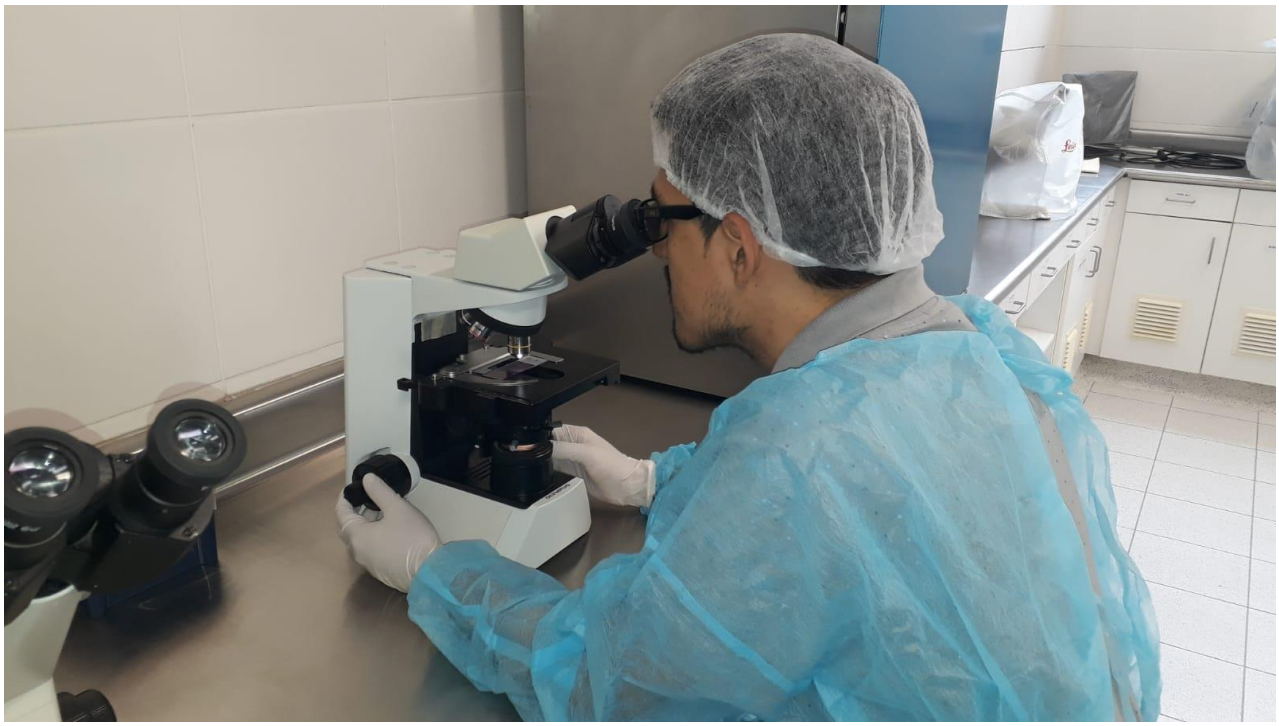

Fig A.3 Observación de muestras a través del microscopio

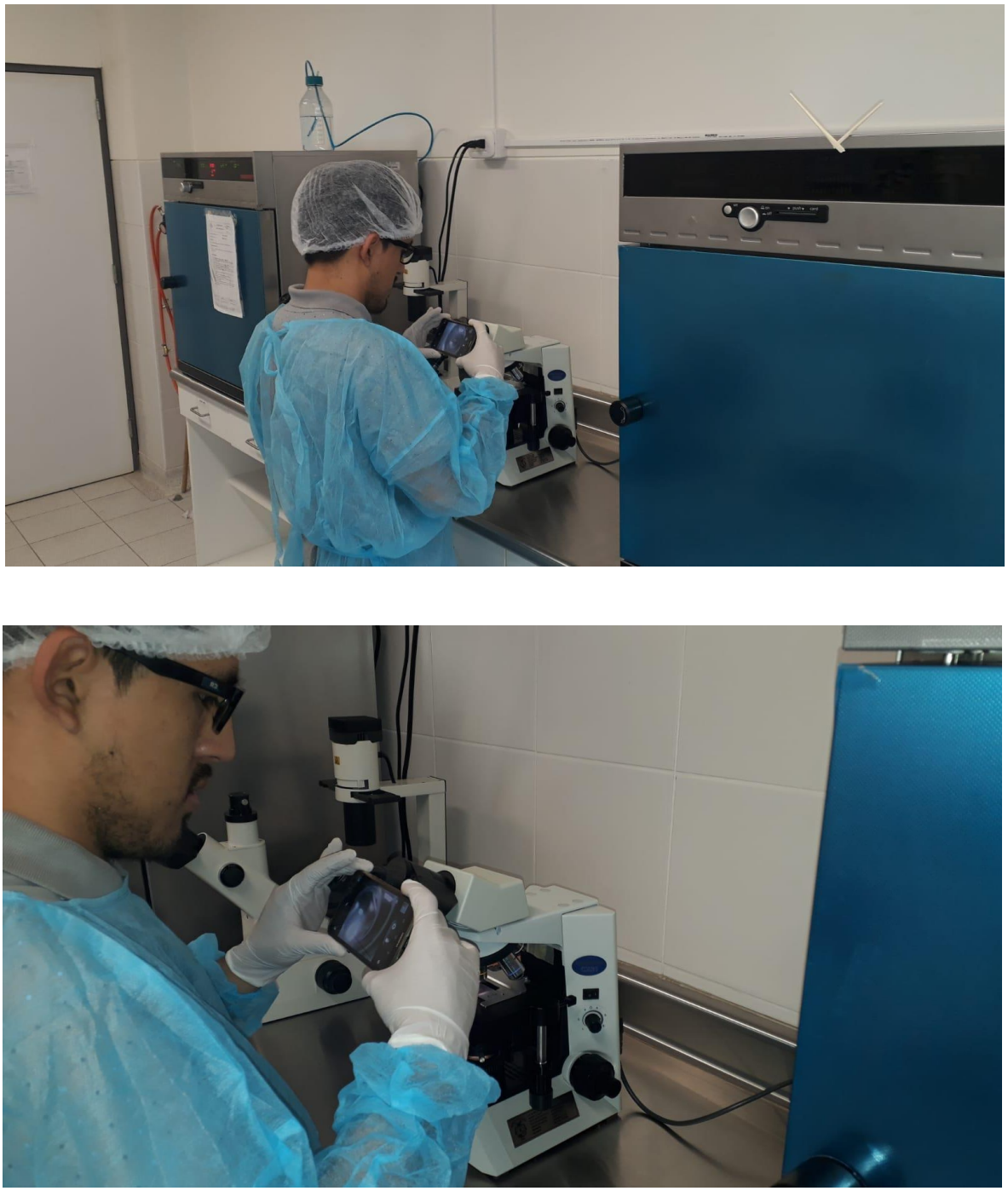

Fig A.4 Captura de las imágenes microscópicas

GOBIERNO REGIONAL LAMBAYEQUE
DIRECCIÓN REGIONAL DE SALUD
HOSPITAL REGIONAL LAMBAYEQUE - CHICLAYO

PERÚ | Ministerio de Salud

"AÑO DE LA LUCHA CONTRA LA CORRUPCIÓN Y LA IMPUNIDAD"

3405202 - 0

CONSTANCIA DE APROBACIÓN DE PROYECTOS DE INVESTIGACIÓN

El Comité de Ética en Investigación, luego de haber revisado de manera expedita el proyecto de investigación, **"ALGORITMO PARA DETECCIÓN DE HUEVOS DE *Trichuris trichiura* EN IMÁGENES MICROSCÓPICAS DE MUESTRAS COPROLÓGICAS - HOSPITAL REGIONAL DE LAMBAYEQUE-2019** ,otorga la presente constancia al autor:

VÁSQUEZ ORTIZ EDUAR ANÍBAL (Investigador Externo)

Y se resuelve:

1. Aprobar la ejecución del mencionado proyecto.
2. Se extiende esta constancia para que el proyecto pueda ser ejecutado en **Laboratorio de Investigación.**
3. El investigador deberá presentar el informe de la investigación.
4. La presente constancia es válida hasta el mes hasta **Marzo 2020.**

Chiclayo, 31 de octubre del 2019

GOBIERNO REGIONAL LAMBAYEQUE
GERENCIA REGIONAL DE SALUD
HOSPITAL REGIONAL LAMBAYEQUE

M.Sc. EMMA VANESSA ARRIAGA DEZA
PRESIDENTE DE ETICA EN INVESTIGACIÓN HRL
CPP. N° 6789

Código_Inv: 0111-052-19 CIEI

Código del programa para obtener la precisión y sensibilidad

```
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from imutils import paths
import numpy as np
import argparse
import time
import cv2
import cPickle
import os

def image_to_feature_vector(image):
	lower_egg = np.array([0,22,85])
	upper_egg = np.array([17,80,170])
	hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
	mask1 = cv2.inRange(hsv, lower_egg, upper_egg)
	#img2=cv2.bitwise_or(image,image,mask=mask1)
	#hist = cv2.calcHist([img2], [0, 1, 2], None, (8,8,8),
	#		[0, 180, 0, 256, 0, 256])
	return mask1.flatten()
	#return hist.flatten()

def extract_color_histogram(image, bins=(3,4,4)):
	hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
	hist = cv2.calcHist([hsv], [0, 1, 2], None, bins,
		[0, 180, 0, 256, 0, 256])
	return hist.flatten()

ap = argparse.ArgumentParser()
ap.add_argument("-d", "--dataset", required=True,
	help="path to input dataset")
ap.add_argument("-j", "--jobs", type=int, default=-1,
	help="# of jobs for k-NN distance (-1 uses all available cores)")
args = vars(ap.parse_args())

print("[INFO] describing images...")
imagePaths = list(paths.list_images(args["dataset"]))

rawImages = []
features = []
labels = []
both=[]
for (i, imagePath) in enumerate(imagePaths):
	image = cv2.imread(imagePath)
	label = imagePath.split(os.path.sep)[-1].split(".")[0]
	#pixels = image_to_feature_vector(image)
```

```
		hist = extract_color_histogram(image)
		#pi=np.concatenate((pixels,hist))
		#rawImages.append(pixels)
		features.append(hist)
		labels.append(label)
		#both.append(pi)
		if i > 0 and i % 100 == 0:
			print("[INFO] processed {}/{}".format(i, len(imagePaths)))
#rawImages = np.array(rawImages)
#print rawImages.shape
#print type(rawImages)
features = np.array(features)
print features.shape
#print type(features)
labels = np.array(labels)
print labels.shape
print type(labels)
#both=np.array(both)
#print both.shape
#print type(both)
#print("[INFO] pixels matrix: {:.2f}MB".format(
#		rawImages.nbytes / (1024 * 1000.0)))
print("[INFO] features matrix: {:.2f}MB".format(
		features.nbytes / (1024 * 1000.0)))
#print("[INFO] both matrix: {:.2f}MB".format(
#		both.nbytes / (1024 * 1000.0)))

#(trainRI, testRI, trainRL, testRL) = train_test_split(
#		rawImages, labels, test_size=0.30, random_state=9)
(trainFeat, testFeat, trainLabels, testLabels) = train_test_split(
		features, labels, test_size=0.30, random_state=9)
#(trainpi, testpi, trainpil, testpil) = train_test_split(
#		 both, labels, test_size=0.30, random_state=9)


params = {"n_neighbors": np.arange(1, 31, 2),
		"metric": ["euclidean", "cityblock"]}

print("[INFO] tuning hyperparameters via grid search")
model = KNeighborsClassifier(n_jobs=args["jobs"])
grid = GridSearchCV(model, params)
start = time.time()
#grid.fit(trainRI, trainRL)
grid.fit(trainFeat, trainLabels)
#grid.fit(trainpi, trainpil)
print("[INFO] grid search took {:.2f} seconds".format(
		time.time() - start))
#acc = grid.score(testRI, testRL)
acc = grid.score(testFeat, testLabels)
#acc = grid.score(testpi, testpil)
```

```
print("[INFO] grid search accuracy: {:.2f}%".format(acc * 100))
print("[INFO] grid search best parameters: {}".format(
            grid.best_params_))
#g=grid.predict(testRI)
#rec=recall_score(testRL,g,average="binary",pos_label='si')
#pre=precision_score(testRL,g,average="binary",pos_label='si')
g=grid.predict(testFeat)
rec=recall_score(testLabels,g,average="binary",pos_label='si')
pre=precision_score(testLabels,g,average="binary",pos_label='si')
#g=grid.predict(testpi)
#rec=recall_score(testpil,g,average="binary",pos_label='si')
#pre=precision_score(testpil,g,average="binary",pos_label='si')
print("[INFO] recall: {:.2f}%".format(rec*100))
print("[INFO] precision: {:.2f}%".format(pre*100))
```