UNIVERSIDAD DE GUANAJUATO.

CAMPUS IRAPUATO – SALAMANCA
DIVISIÓN DE INGENIERÍAS

"SEGUIMIENTO DEL ORGANISMO UNICELULAR PARAMECIUM TETRAURELIA USANDO ALGORITMOS DE VISIÓN POR COMPUTADORA."

TESIS

QUE PARA OBTENER EL TÍTULO DE:
INGENIERO EN SISTEMAS COMPUTACIONALES
PRESENTA:

ISAAC MEJÍA FLORES

DIRECTORES:
DR. NATANAEL BENITO CUANDO ESPITIA
DR. MIGUEL TORRES CISNEROS

SALAMANCA, GTO. 2023

Dedicatoria

A mis padres Isaac M. y Aura Lizett F. por su apoyo a lo largo de toda la carrera.

Desde el inicio de mi educación, ustedes han sido mi faro, mi guía constante. Han sido testigos de mis altibajos, de mis momentos de incertidumbre y de mis triunfos más grandes. En cada paso del camino, han estado allí para impulsarme a dar lo mejor de mí y recordarme que puedo superar cualquier desafío.

Gracias por estar siempre dispuestos a escuchar mis preocupaciones, por brindarme sabios consejos y por alentarme a perseguir mis pasiones. Han sido mi soporte en los momentos difíciles y mi mayor fuente de alegría en los momentos de celebración. Su presencia constante ha sido un regalo inestimable que ha dado forma a la persona en la que me he convertido.

Padre y madre, ustedes son mi mayor inspiración. Mi éxito es su éxito y mi felicidad se multiplica cuando puedo ver una sonrisa en sus rostros. A medida que concluyo este capítulo y me embarco en nuevos desafíos, quiero que sepan que su amor y apoyo continuarán siendo mi fuerza impulsora.

Agradecimientos

Agradezco al Consejo Nacional de Ciencia (CONACyT) por la beca otorgada a mi persona y por el apoyo parcial a este trabajo, otorgado mediante los proyectos CF2019-102963 y CB-2016-01-286629

A mis asesores por su paciencia y sus grandes consejos y comentarios.

A mis amigos del SVA por siempre brindarme una gran convivencia y apoyo en estos años que hemos compartido a lo largo de la carrera.

A mi novia por siempre escucharme y aconsejarme a lo largo de los altibajos que pude tener a lo largo de este camino.

A mi Tocayo por ser mi apoyo durante este proyecto y las vivencias que compartimos a lo largo de esta experiencia

Al Centro de investigaciones en óptica (CIO) unidad Aguascalientes por darme la oportunidad de realizar este trabajo de investigación en sus instalaciones.

A la Universidad de Guanajuato por ser mi casa de estudios y proporcionarme los conocimientos que requiero para afrontar los siguientes retos que me deparen en mi vida profesional

Tabla de contenido

RESUMEN 6

Capítulo 1 Conceptos básicos de la visión por computadora 7

1.1 La imagen digital 7

1.2 Escala de grises en una imagen digital 8

1.3 La imagen binaria 8

1.4 Espacios de color. 9

1.4.1 Funciones de correspondencia de color. 9

1.4.2 Espacio de color CIE-RGB 10

1.5 Histograma de una imagen digital 12

1.5.1 Histograma de una imagen a color 13

1.5.2 Operaciones con histograma de una imagen digital. 15

1.6 Morfología matemática 18

1.6.1 Retículo (Lattice) 18

1.6.2 Operaciones elementales: Dilatación 19

1.6.3 Operaciones elementales: Erosión 20

1.7 Detección de bordes 21

1.7.1 Gradiente de una imagen digital 21

1.7.2 Algoritmo de Canny 23

Capítulo 2 Obtención y filtrado de las imágenes del Paramecium Tetraurelia 24

2.1 El Paramecium Tetraurelia 24

2.2 Medio de cultivo 24

2.3 Arreglo experimental 25

2.4 Obtención de imágenes de células del Paramecium Tetraurelia 27

2.4.1 Obtención y limpieza de la muestra 27

2.4.2 Correlación preliminar entre láser y nado del Paramecium tetraurelia 28

2.5 Filtrado de las imágenes del Paramecium tetraurelia (Fase de preprocesamiento de la imagen). 29

2.5.1 Escala de grises 29

2.5.2 Binarización de la imagen (Threshold) 30

2.5.3 Operaciones morfológicas de la imagen resultante 31

Capítulo 3 Análisis estructural de la imagen binaria 34

3.1 Detección de bordes para análisis estructural 34

3.2 Análisis estructural 36

3.2.1 Ordenamiento topológico por seguimiento de borde 36

3.3 Aproximación a una elipse 38

3.3.1 Algoritmo de ajuste de mínimos cuadrados lineales 38

3.4 Extracción y almacenamiento de datos 40

3.4.1 Obtención del centroide de la elipse 40

3.4.2 Obtención del ángulo de la elipse (matriz de covarianza) 41

3.4.3 Almacenamiento de datos 42

Capítulo 4 Conclusiones y trabajo a futuro 44

4.1 Ventajas del sistema de visión por computadora 44

4.2 El software propuesto bajo GNU General Public License 44

4.3 Trabajo a futuro 45

4.3.1 Analisis de celulas de Paramecium tetraurelia en tiempo real 45

4.3.2 Modificación de umbrales 46

Anexo: Función operativa del software 47

Compilación y ejecución del software 47

Parámetros de configuración 47

Referencias 49

RESUMEN

La detección de células individuales de microorganismos es una herramienta importante ya que ayuda a describir el comportamiento de estos organismos. Este tipo de herramientas permiten una descripción y detección de estos microorganismos y pueden dar mucha información relevante respecto a su geometría y a la estadística de movimiento. Con esta información es posible plantear análisis estadísticos para tener una serie de modelos matemáticos más descriptivos. Tradicionalmente, para hacer la detección de estos microorganismos se utilizan métodos de intervención biológica lo cual complica los procesos y dificulta la obtención de datos cuantitativos precisos.

Este trabajo de investigación se centra en el estudio del microorganismo *Paramecium tetraurelia* el cual es un organismo unicelular ciliado que se encuentra naturalmente en cuerpos de agua dulce. Al ser un organismo ciliado, poseen en su membrana celular una serie de cilios cuyo movimiento colectivo permite la locomoción del organismo en un medio líquido.

En la literatura existe un gran número de estudios que describen la taxonomía y comportamiento de este organismo. Por ejemplo, se sabe que este microorganismo regula el abatimiento de sus cilios por medio de señales eléctricas en su membrana [1]. Esto es muy relevante debido a que las respuestas eléctricas de la membrana se relacionan directamente con las características de nado que exhiben. Por ende, este microorganismo tiene el potencial para ser usado como modelo de estudio de células más complejas como las neuronas. Es por esto por lo que en la literatura al *Paramecium tetraurelia* se le ha denominado como “La neurona nadadora” [2].

Por lo anterior este trabajo centra el enfoque en el desarrollo de un novedoso software capaz de detectar células individuales del microorganismo *Paramecium tetraurelia* en imágenes de un microscopio óptico digital usando algoritmos de visión por computadora. Los resultados de este trabajo están orientados a dar información relevante para el estudio y análisis del comportamiento de estas células.

Capítulo 1

Conceptos básicos de la visión por computadora

La visión por computadora se puede definir como una serie de algoritmos que permiten a los equipos de cómputo obtener imágenes del mundo real por medio de periféricos ópticos (cámaras) y extraer información relevante a través de un análisis de dichas imágenes [3]. Debido a que el presente trabajo se centra en el desarrollo de algoritmos de visión por computadora, en este capítulo se describen brevemente los conceptos fundamentales de la visión por computadora.

1.1 La imagen digital

Una imagen digital puede ser definida como una matriz rectangular de datos en un dominio espacial el cual consiste en un arreglo de *pixeles (x, y, u)* el cual cada uno combina un punto en $(x, y) \in \mathbb{Z}^2$. Bajo esta definición, el valor de *u* representa la intensidad de cada pixel. El conjunto de estos puntos forma una malla regular [4].

De manera formal una imagen I se puede definir de acuerdo con la ecuación (1.1).

$$I = \{(x, y): 1 \leq x \leq N_{cols} \wedge 1 \leq y \leq N_{rows}\} \subset \mathbb{Z}^2 \quad (1.1)$$

Donde *I* es la imagen, *x, y* representan los puntos o pixeles en la malla de la imagen digital.

En la Figura 1.1 se muestra una imagen y el acercamiento para poder apreciar los pixeles en dicha imagen. Se observa de manera detallada el grid que compone la imagen.

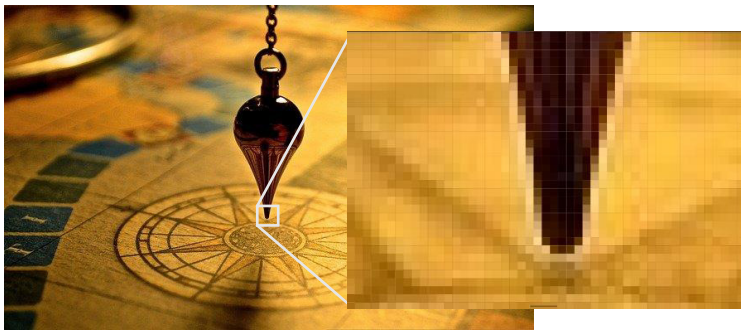

Figura 1.1 Representación de una imagen digital en el grid. La imagen mostrada ha sido recuperada de pixabay bajo licencia gratuita.

Cada pixel contiene la información de intensidad del dominio de la imagen dependiendo del tipo de imagen que se tenga en pantalla. Estas imágenes pueden ser monocromáticas (escala de grises), binarias o policromático. Mas adelante se hablará al respecto de los espacios de color y como se procesan en una computadora.

1.2 Escala de grises en una imagen digital

Una imagen de escala de grises está compuesta por valores por pixel de $\{0 \leq u \leq 255\} \subset \mathbb{Z}^2$ esto dado que se utiliza un formato de 8 bits. Un valor de 0 representa un pixel completamente negro, mientras que un valor de 255 representa un pixel completamente blanco, y los valores entre ellos describen niveles de gris intermedios.

La elección de 8 bits por canal para describir la intensidad de la luz en una imagen se basa en una combinación de factores técnicos y económicos, incluida la capacidad de procesamiento de los dispositivos y la cantidad de memoria disponible en los sistemas informáticos. Aunque existen formatos de imagen de mayor "*bit depth*", el formato de 8 bits por canal sigue siendo ampliamente utilizado debido a su equilibrio entre calidad y eficiencia.

En la Figura 1.2 se muestra una representación de esta escala.

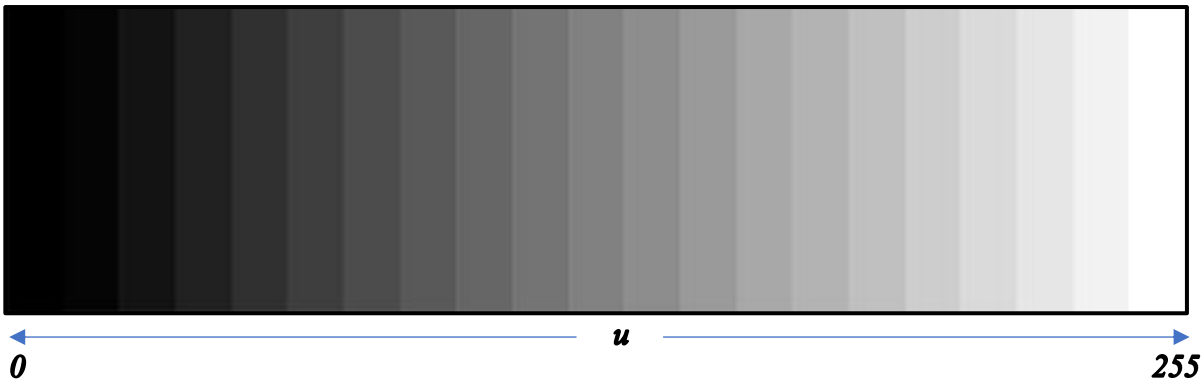

Figura 1.2 Representación de la escala de grises. Es importante mencionar que esta es una representación de la escala de grises y no muestra los 255 diferentes tonos de grises.

1.3 La imagen binaria

Se conoce como imagen binaria a aquella imagen que comprende los valores de intensidad $\{0 \leq u \leq 1\} \subset \mathbb{Z}^2$ las cuales se denotan los valores de 0 como negro representando el fondo de la imagen y el 1 como blanco representando el contenido en primer plano de la imagen (Es importante aclara que para este trabajo se toma esta interpretación de los valores de pixeles, pero otros trabajos pueden usar la interpretación contraria a la propuesta aquí).

Para esta separación de dos clases de la imagen se tiene el concepto del umbral el cual, en una imagen binaria, los pixeles con intensidades por encima del umbral se asignan a un valor de 1 y los pixeles con intensidades por debajo del umbral se asignan a un valor de 0.

La elección del valor de umbral adecuado depende de la aplicación y de la calidad de la imagen de entrada, sin embargo, en algunos casos es posible aplicar un algoritmo automático para determinar el mejor valor de umbral.

Un ejemplo de imagen binaria se muestra en la Figura 1.3. Nótese que la imagen mostrada en la Figura 1.3 es una versión binaria de la imagen de la Figura 1.1.

Figura 1.3 Ejemplo de una imagen binaria

1.4 Espacios de color.

Los espacios de color son el mecanismo para representar el color en un dispositivo electrónico. Estos espacios de color se basan en una serie de algoritmos los cuales siguen un estándar determinado. Típicamente, estos estándares son establecidos por la CIE (*Commission internationale de l'éclairage).* En general, existen distintos espacios de color y se basan en como el ser humano es capaz de percibir el color.

1.4.1 Funciones de correspondencia de color.

Los estándares del color basan su funcionalidad en las funciones de correspondencia de color (*Color matching functions*). Estas funciones buscan describir de forma numérica la respuesta cromática del ser humano. Cuando la combinación de color es lineal se puede construir un algoritmo simple el cual logra determinar como el ojo humano o una cámara digital, percibe y responde a diferentes longitudes de onda de la luz visible. En otras palabras, estos algoritmos describen la sensibilidad del sistema a diferentes colores y esta puede ser calculada a través de sumar las fuentes de longitudes de onda por separado. La suma de las fuentes a diferentes longitudes de onda se obtiene con base en cada set de funciones primarias que se pueden describir como P_1 , P_2 y P_3 [5]. Esta nomenclatura hace referencia a las funciones que describen la sensibilidad de los 3 tipos de conos en el ojo humano de los diferentes colores. El cono de tipo L, sensible a longitudes de onda larga (color rojo). El cono de tipo M, sensible a longitudes de onda mediana (color verde) y el cono de tipo S, sensible a longitudes de onda corta (color azul) [6].

Se puede sintonizar la densidad de cada componente para obtener una unidad de energía de la fuente para cada longitud de onda la cual en la literatura es representada por $f_1(\lambda)$, $f_2(\lambda)$ y $f_3(\lambda)$. Ahora, para una longitud de onda λ_0 la energía se puede descomponer en términos de las funciones anteriores como se muestra en la ecuación (1.2).

$$U(\lambda_0) = f_1(\lambda_0)P_1 + f_2(\lambda_0)P_2 + f_3(\lambda_0)P_3 \quad (1.2)$$

Esta fuente es la suma de una vasta cantidad de longitudes de onda, la cual tiene diferentes intensidades. En la ecuación (1.3) se muestra cómo se logra empatar cada fuente de longitudes de onda y se obtiene la fuente final de luz [5].

$$S(\lambda) = \omega_1 P_1 + \omega_2 P_2 + \omega_3 P_3$$

$$= \left\{\int_A f_1(\lambda)S(\lambda)d\lambda\right\}P_1 + \left\{\int_A f_2(\lambda)S(\lambda)d\lambda\right\}P_2 + \left\{\int_A f_3(\lambda)S(\lambda)d\lambda\right\}P_3 \quad (1.3)$$

La ecuación (1.3) se compone de tres términos, cada uno de los cuales describe la contribución de cada uno de los tres tipos de conos en el ojo humano a la percepción del color. Cada término es una integral que describe la sensibilidad del cono a la luz S(λ), ponderada por un factor ω_i que representa la importancia relativa de cada cono en la percepción del color.

1.4.2 Espacio de color CIE-RGB

En 1931 se establece el sistema de representación tricromático por el CIE el cual su sistema se basa en tres colores principales como sus siglas en ingles lo indican R(Red), G(Green), B(Blue). Para su funcionamiento los valores del sistema RGB se han adquirido en base a experimentos con luz para poder dotar al algoritmo de funciones correctas para su descripción en una pantalla. Estas funciones se muestran en la ecuación (1.4), (1.5) y (1.6) [7].

$P_1 = R = \delta_{700}$ (1.4)

$P_2 = G = \delta_{546.1}$ (1.5)

$P_3 = B = \delta_{435.8}$ (1.6)

Las definiciones anteriores corresponden a la longitud de onda de los colores físicos. Siendo 700nm la longitud de onda para el espectro rojo, 546nm para el espectro verde y 435.8nm para el espectro azul (referente a los tres conos en el ojo humano).

Dado que el espacio de RGB tiene 3 componentes se puede representar como un vector en el espacio que en este caso es $\mathbb{R}^3$. Al ser un vector en el espacio puede tener sus tres componentes representados como $[c_1\ c_2\ c_3]$ el cual al ser variados puede resultar en diferentes combinaciones de colores e intensidades (Luminiscencia).

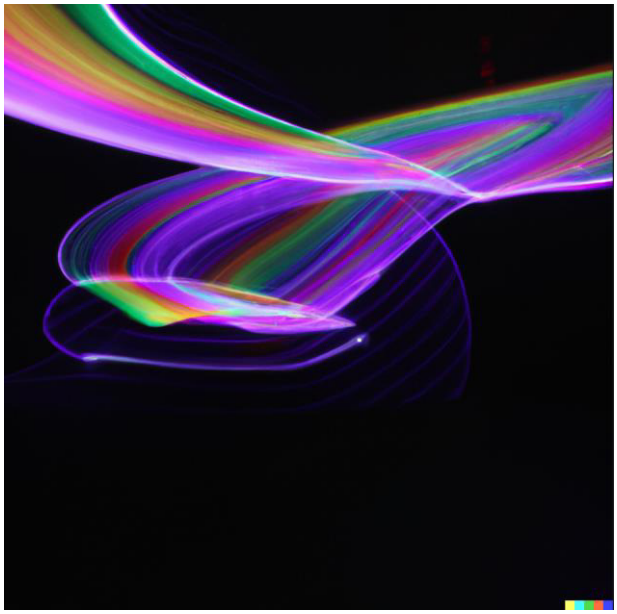

Figura 1.4 Para lograr diferentes combinaciones de colores se tiene que variar los valores de las primitivas del espacio de color. Esta imagen se generó con Dall-E de open AI pidiendo que generara una representación artística del espacio RGB en una computadora

1.4.2.1 Representación geométrica del espacio RGB

Según Jonas Gomes, Luiz Velho y Mario Costa (2012) [7] la representación geométrica del color abre paso a conceptos básicos como la luminiscencia y la cromaticidad. En general, estos conceptos se refieren a la característica de los colores y su percepción por el ojo humano. La luminiscencia describe la cantidad de luz percibida por el ojo y se relaciona con el brillo o la claridad del color. La cromaticidad describe el matiz y el tono del color.

Matemáticamente hablando se tiene que al considerar una distribución espectral en función de $C(\lambda)$ y un número real $t > 0$ tomando en cuenta la función de distribución espectral $C'' = tC$ lo cual describe que la luminiscencia de C'' es t veces la luminiscencia de C, es decir, que describe un color con la misma cromaticidad, pero con una luminiscencia t veces mayor que la del color original. Esto significa que la cantidad de luz percibida por el ojo se ha multiplicado por t, lo que resulta en un color más brillante o claro.

La percepción de color es importante ya que radica en la percepción que tenemos del mundo que nos rodea. El concepto de la percepción del color se aplica al espacio de color RGB para dar una representación de zonas más oscuras o claras en una imagen. La variación de los parámetros de las componentes del espacio de color RGB permite ajustar la luminosidad sin afectar la longitud de onda principal C(λ), lo cual es crucial en la manipulación de imágenes y la producción de efectos visuales.

Para la representación geométrica final del espacio de color RGB se utiliza una representación cubica [7], la cual se muestra en la Figura 1.5 Este modelo se basa en el

sistema cartesiano, en este se puede observar que en las equinas se tienen los colores principales del espacio los cuales son el Rojo, Verde y Azul. La combinación de estos colores primarios da pauta a los colores secundarios los cuales son Cian, Magenta y Amarillo los cuales son las otras 3 esquinas del cubo. En el punto [0,0,0] u origen se representa el negro. El color blanco se representa en un espacio de color RGB como la combinación de los valores máximos de los tres canales, cada uno de ellos con una resolución de 8 bits. Estos dos puntos son los que nos pueden acercar más a los diferentes niveles de luminiscencia que se tienen en el espacio físico.

Cuando se habla de las 3 primitivas se habla de en qué punto coordenado estamos del cubo unitario, y si hablamos de que son puntos de 0 a 255 (en una representación computacional) se habla de una combinación de $256^3 = 16{,}777{,}216$ colores posibles.

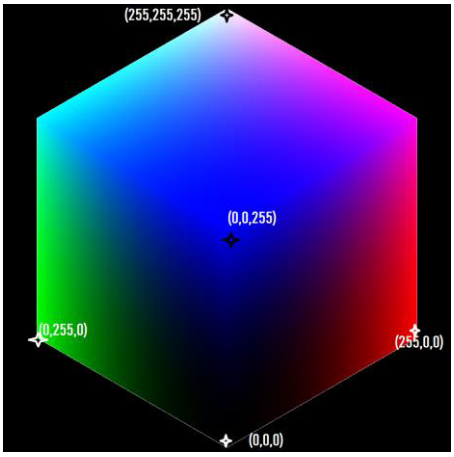

Figura 1.5 Representación del cubo para el espacio RGB

1.5 Histograma de una imagen digital

El histograma de una imagen digital puede ser definido como un gráfico el cual representa la distribución de intensidades de los pixeles en una imagen digital en un rango de [0, 255]. Los datos del histograma se pueden describir como una función discreta $h(r_k) = n_k$ donde r_k se entiende como el k-ésimo nivel de intensidad y n_k corresponde al k-ésimo número de pixeles en la imagen.

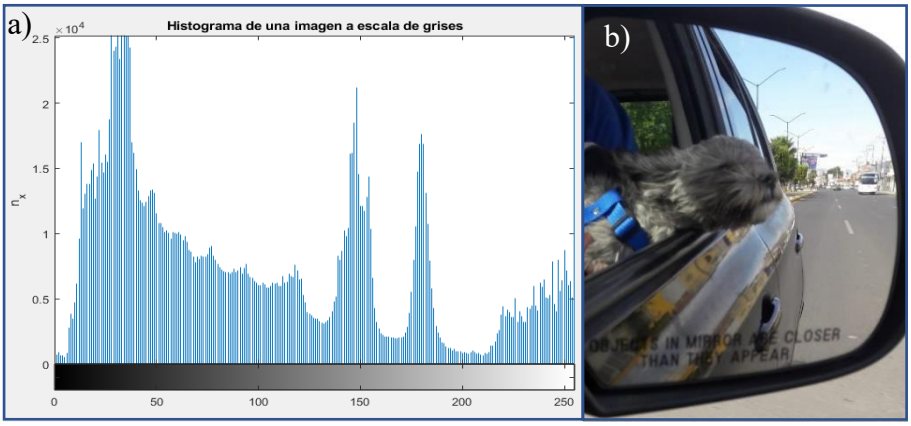

Figura 1.6. Histograma de una imagen a escala de grises. La imagen se procesó mediante Matlab para generar figuras descriptivas de los espacios de color e histogramas

En la Figura 1.6 se muestra el histograma de una imagen digital en escala de grises. Como se puede observar en a) el histograma puede brindar información relevante de una imagen como lo es la distribución de las intensidades a lo largo de la imagen. A su vez con esto se pueden hacer cálculos estadísticos interesantes para poder identificar elementos en una imagen.

1.5.1 Histograma de una imagen a color

Como se vio anteriormente, así como existen imágenes a escala de grises como la de la figura 1.6 b) también existen imágenes con representación de color. Con lo cual queda una incógnita relevante para el estudio de la visión por computadora. ¿Cómo se obtiene un histograma de una imagen a color?

El espacio de color RGB se divide en 3 componentes, por lo tanto, para que una imagen se vea a color hay que sobreponer los 3 canales uno sobre otro como se muestra en la Figura 1.7.

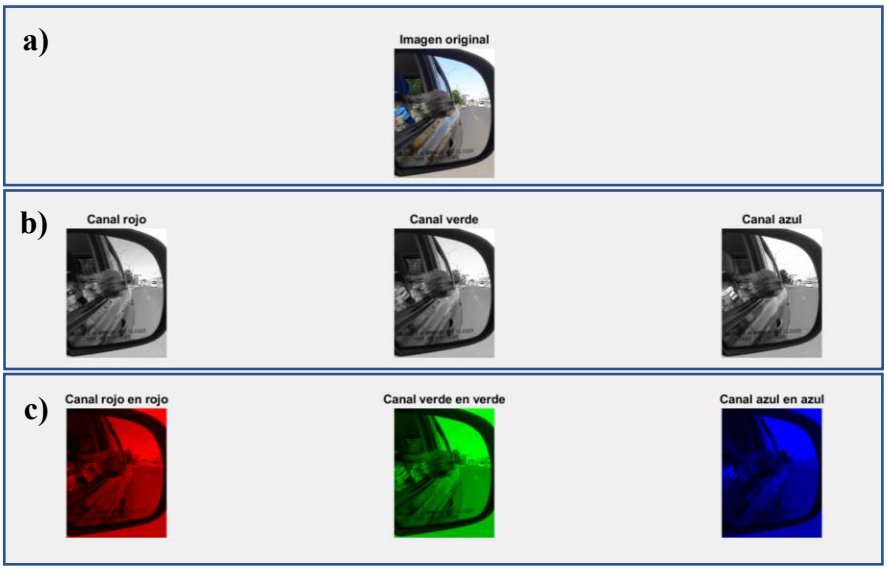

Figura 1.7 Ejemplo de separación de una imagen en el espacio de color RGB. Se muestra cómo se descompone una imagen RGB en sus primitivas

En la Figura 1.7 a) Se muestra como está constituida la imagen original con sus tres canales. En 1.7 b) se descompone la imagen en sus tres canales R, G y B respectivamente y se representan en escala de grises. Se ve en escala de grises debido a que al ser canales independientes la computadora lo interpreta como Unicanal (solo grises) sin embargo se puede apreciar cómo cambian las intensidades de los pixeles en cada canal. En la Figura 1.7 c) se hace una representación a color de cada uno de los canales para poder apreciar mejor estas primitivas. En la Figura 1.8 se muestran los tres histogramas de la imagen (uno por cada canal).

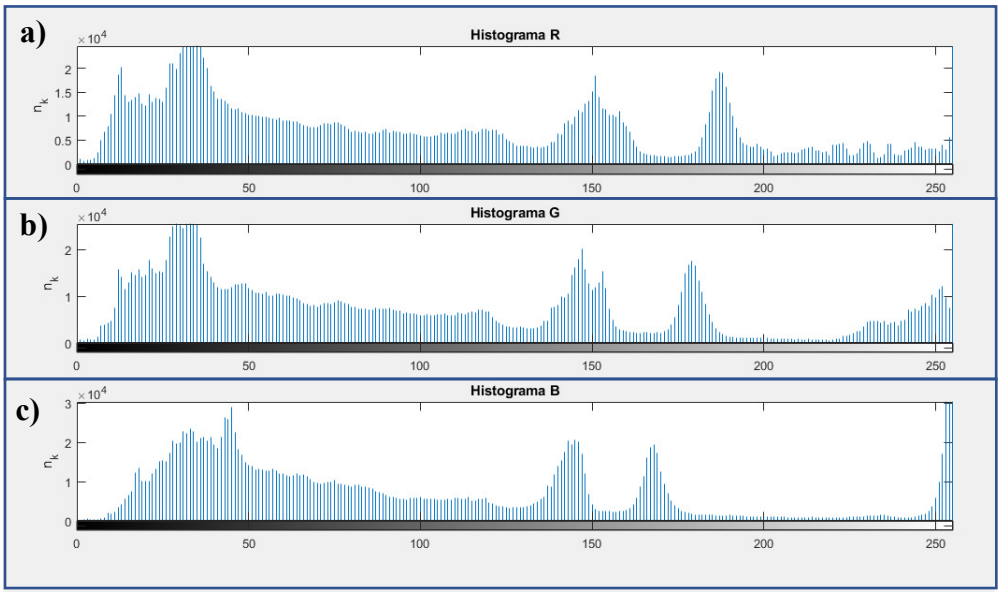

Figura 1.8. Histogramas de los 3 canales de la imagen de la Figura 1.7b, se puede observar la distribución de los colores con ayuda de las intensidades.

1.5.2 Operaciones con histograma de una imagen digital.

Como se mencionó anteriormente, el histograma contiene información relevante de una imagen digital. Con esta herramienta se puede hacer varias estimaciones estadísticas o probabilísticas gracias a la representación de la distribución de los pixeles. A continuación, se revisarán algunas operaciones que se pueden realizar con el histograma de una imagen digital.

1.5.2.1 Normalización

Una práctica común es la de la normalización de una imagen digital. Las normalizaciones digitales tienen muchos usos, entre ellos puede ser la estimación de la probabilidad de ocurrencia del nivel de intensidad de una imagen [8].

La normalización se hace dividiendo cada uno de los componentes de la imagen entre el número total de pixeles en la imagen. El número total de pixeles puede calcularse como el producto del número de pixeles a lo ancho de la imagen y el número de pixeles a lo largo de la imagen. Este producto se puede denotar como MN. A su vez, se puede expresar con la ecuación (1.7)

$$P(r_k) = \frac{n_k}{MN} \quad (1.7)$$

Donde r_k es el k- ésimo nivel de gris y n_k es el número de pixeles en la imagen que contienen ese nivel r_k

En la Figura 1.9 se pueden observar dos histogramas de la misma imagen, en 1.9 a) se observa el histograma original. En 1.9 b) se muestra el histograma después del proceso de normalización. En este último histograma se puede apreciar una reducción de n_k sin afectar la distribución de las intensidades r_k.

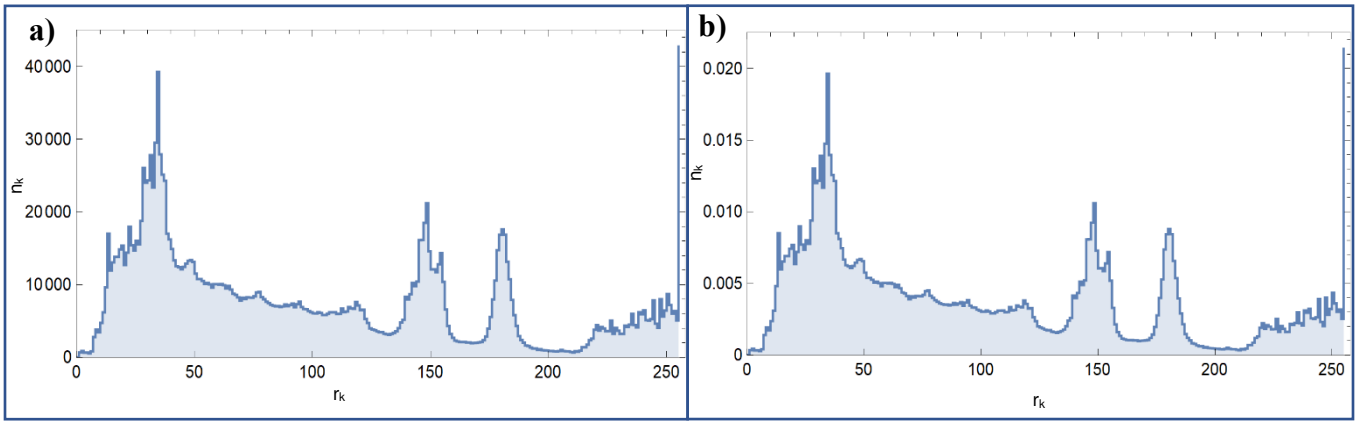

Figura 1.9 Comparativa de histogramas de una misma imagen, en a) se tiene el histograma original y en b) el histograma después de un proceso de normalización.

1.5.2.2 Ecualización de histogramas

La ecualización del histograma es una técnica muy usada en el procesamiento de imágenes digitales con la cual es posible ajustar las intensidades de los pixeles (r_k) para mejorar el contraste de una imagen y mejorar los bordes de la imagen de manera favorable. La ecualización de histogramas puede expresarse mediante la ecuación (1.8) como sigue:

$$S = T(r) \quad 0 \leq r \leq 1 \quad (1.8)$$

Donde $T(r)$ es la función de transferencia para la ecualización.

La función de transferencia del histograma debe satisfacer dos condiciones:

a) $T(r)$ debe tener un valor único que asegure que existirá la transformada inversa, y $T(r)$ debe aumentar monótonamente para asegurar que la imagen se mueve de negro a blanco
b) Y además debe ocurrir que: $0 \leq T(r) \leq 1$ que para $0 \leq r \leq 1$ para que la imagen tenga la misma entrada y salida y tengan así el mismo tamaño.

Para este punto se requiere la señal de entrada del histograma, la función de transferencia el cual se usará para generar el histograma ecualizado [9].

Según Abhishek Yadav, y Poonam Yadav "La señal del histograma de entrada puede ser generado mediante la función de densidad probabilística o PDF ya que los niveles de grises en la imagen son distribuidos de manera aleatoria y PDF es solo una herramienta la cual puede calcular el histograma" [9].

Por lo cual esto solo puede ser observado si $P_r(r)$ es igual a la PDF de los niveles de grises de la imagen de entrada y la función de transferencia de la ecualización se conoce, teniendo como salida $P_s(s)$ mediante la ecuación (1.9):

$$P_S(S) = P_r(r)\left|\frac{dr}{ds}\right| \quad (1.9)$$

Como la PDF es usada para la generación del histograma de la señal de entrada la transformación de la función puede ser en forma de una función de distribución cumulativa. Por lo tanto, se puede generar una nueva relación como se muestra en la ecuación (1.10)

$$S = T(r) = \int_0^r \mathrm{P_r}(w)\, dw \quad (1.10)$$

En donde de acuerdo con la literatura [9] *w* representa un cambio de variable intermedia para la realizar la operación de integración.

Ahora según la literatura [9], se debe verificar que la $T(r)$ generara una imagen con un histograma ecualizado por lo tanto tenemos que:

$$\frac{ds}{dr} = \frac{d}{dr}\left[\int_0^r \mathrm{P_r}(w)\, dw\right]$$

$$\left|\frac{ds}{dr}\right| = \frac{1}{P_r(r)} \quad (1.11)$$

Sustituyendo la ecuación (1.10) en la ecuación (1.9) se tiene que

$$P_S(S) = \left|\frac{dr}{ds}\right| * \frac{1}{P_r(r)}$$

$$P_S(S) = 1 \quad (1.12)$$

El resultado de la ecuación (1.12) indica que se generará un histograma ecualizado de cualquier imagen insertada. En la Figura 1.10 se muestra el resultado de ecualizar una imagen.

Como se puede observar en 1.10 d) la distribución de los pixeles con respecto a la intensidad es más uniforme con respecto a 1.10 c), esto se ve reflejado de 1.10 a) a 1.10 b) en cómo el contraste de la imagen cambia haciendo los bordes y las imperfecciones de la imagen más notorios.

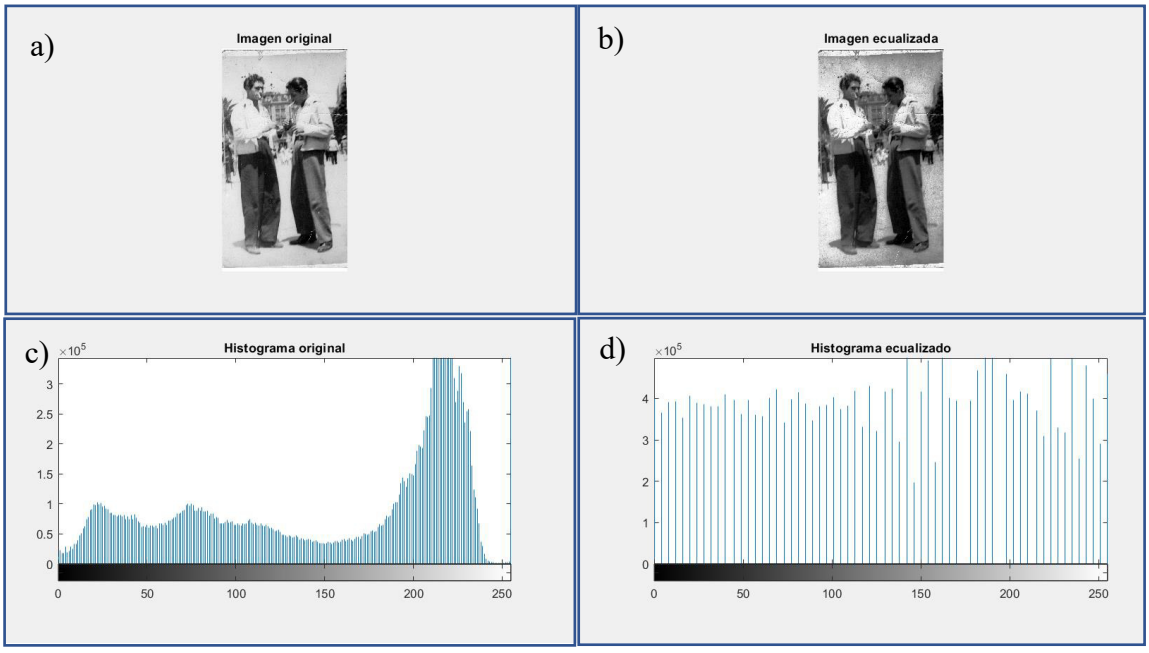

Figura 1.10 Ejemplo de una ecualización del histograma de una imagen

1.6 Morfología matemática

La morfología matemática es una técnica de análisis y procesamiento de estructuras geométricas. Esta técnica se basa en la teoría de conjuntos, la teoría de retículos, topología y funciones aleatorias [10].

La morfología matemática es usada ampliamente en el campo de la visión por computadora para refinar algún objeto estructural en una imagen binaria y hacer más fácil análisis posteriores.

Para poder definir los operadores de la morfología matemática hay que entender la noción de la teoría de retículos.

1.6.1 Retículo (Lattice)

Según Najman y Talbot[11] un retículo (lattice) es un conjunto de elementos *E* (el espacio) dotado con una relación de orden (*) la cual es reflexiva ($\forall x \in E,\ x * x$), antisimétrica ($x * y \,\&\, y * x \rightarrow x = y$) y transitiva ($x * y \,\&\, y * z \rightarrow x * z$). Esto quiere decir que en este ordenamiento para todo 'x' y 'y' se puede definir el mismo elemento más largo $x \bigvee y$ un elemento más pequeño $x \wedge y$. Najman y Talbot definen dos elementos importantes en su análisis: el *supremum* y el *infimum*. Una retícula se considera completa si cualquier subconjunto (P) de E tiene tanto un supremum como un infimum, y ambos pertenecen al espacio (E). El *supremum* es formalmente el más corto de los elementos de *E* tal que son mayores todos los elementos de *P.* Por otro lado, el *infimum* es el elemento más largo de *E* tal que es el elemento más pequeño de todos los elementos de *P.*

En una retícula el *supremum* y el *infimum* juegan un rol de simetría. En particular, si se considera la retícula $\mathcal{P}[E]$ constituida por la colección de todos los subconjuntos de *E,* dos operadores ψ y $\psi*$ son duales si para todo *X,* $\psi(X^c = [\psi^*(X)])$ Donde $X^c = E \backslash X$ es el complemento de *X* en *E*.

En la Figura 1.11 se tiene un ejemplo de una retícula. En esta figura podemos observar que el cubo RGB de la Figura 1.5 puede ser un ejemplo de retícula ya que tiene conjuntos y subconjuntos en su espacio *E.*

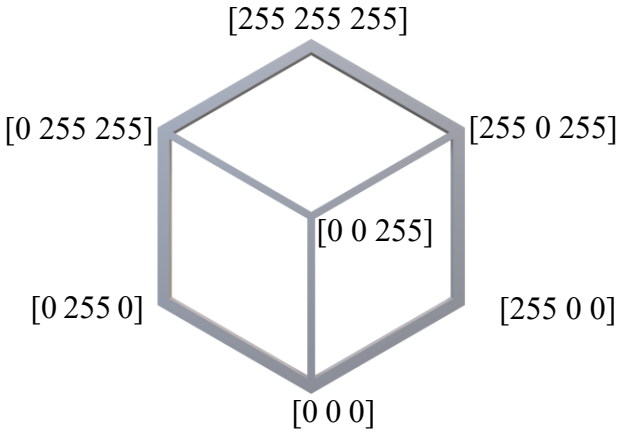

Figura 1.11. El cubo RGB es una representación de una retícula ya que es un conjunto E con valores en $\mathbb{R}^3$y subconjunto de valores P.

En la visión por computadora, la retícula se utiliza para definir la forma y la estructura de los objetos que se procesan. Por ejemplo, un elemento estructurante puede ser una matriz binaria que define la forma de un objeto que se va a dilatar o erosionar en una imagen.

En una retícula, el supremum y el infimum se utilizan para determinar el límite superior e inferior de un conjunto de elementos. En las operaciones morfológicas, el supremum y el infimum se utilizan para determinar la forma y la estructura de los elementos estructurantes utilizados para procesar imágenes.

1.6.2 Operaciones elementales: Dilatación

Para hablar acerca de las operaciones elementales que existen hay que entender el concepto de elementos estructurantes. En el estudio de la morfología matemática se tiene que usar una familia especial de conjuntos las cuales permiten sacar conclusiones de cómo la forma del elemento encaja con la imagen que se quiere operar.

Es importante tener en cuenta que la elección del elemento estructurante puede afectar significativamente el resultado de la operación, por lo que es importante elegir un elemento adecuado para cada aplicación específica.

Este conjunto de elementos estructurantes *B* no son más que imágenes binarias con una forma definida.

La dilatación de una imagen binaria en términos simples trata de tomar la imagen y expandirla al tamaño requerido. Matemáticamente hablando se tiene que la dilatación *X* por *B* es definida como un conjunto de todos los puntos x tal que B_x se intersecta con *X,* Para este punto se tiene una intersección la cual no está vacía.

$$X \oplus B \triangleq \{x: B_x \cap X \neq \phi\} \qquad (1.13)$$

En la Figura 1.12 se muestra un ejemplo de una dilatación en una imagen binaria. Para esta operación se usó un elemento estructurante cuadrado de 6 pixeles por 17 pixeles. La forma matricial numérica de la matriz estructurante se representa de la siguiente manera:

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

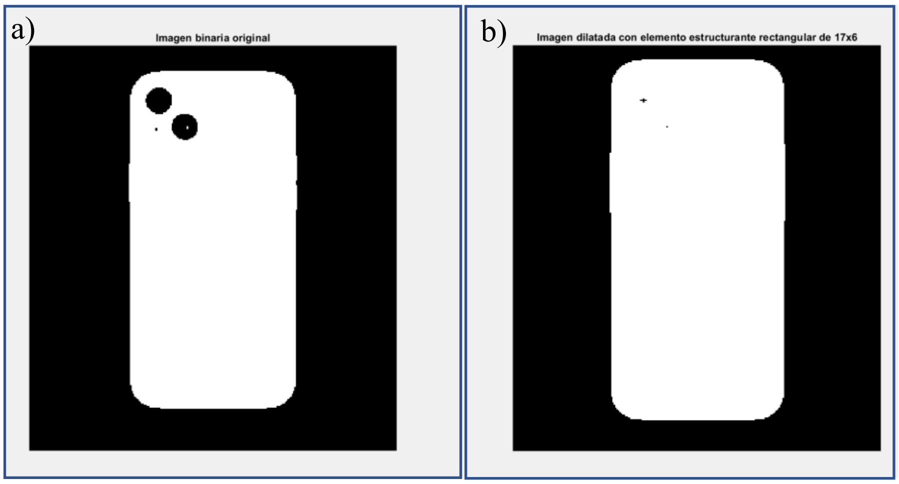

Figura 1.12 Dilatación de una imagen, en a) se observa la imagen original y en b) la imagen dilatada con un elemento estructurante de 6x17, imagen recuperada de Google.com y procesada bajo Matlab con un kernel de 6x17.

1.6.3 Operaciones elementales: Erosión

Contrario a la operación de dilatación, la erosión busca que a partir de la imagen binaria original reducir la cantidad de pixeles que se tienen en valor de 1, esto con el fin que si por ejemplo se tiene una imagen en la cual los pixeles se solapan entre sí y se quiere lograr una detección de dos objetos por separado el algoritmo permita reducir esa probabilidad de traslape entre varios cuerpos. La ecuación (1.14) describe la relación matemática de la erosión el cual B_x denota la translación de B de modo que su origen se encuentra en x. Entonces la erosión de X por B es definida como un conjunto de todos los puntos x tal que B_x es incluido en X.

$$X \ominus B \{x: B_x \subset X\} \quad (1.14)$$

La Figura 1.13 muestra la operación de la erosión con el mismo elemento estructurante de la Figura 1.12, se puede apreciar el cómo se sustraen los pixeles.

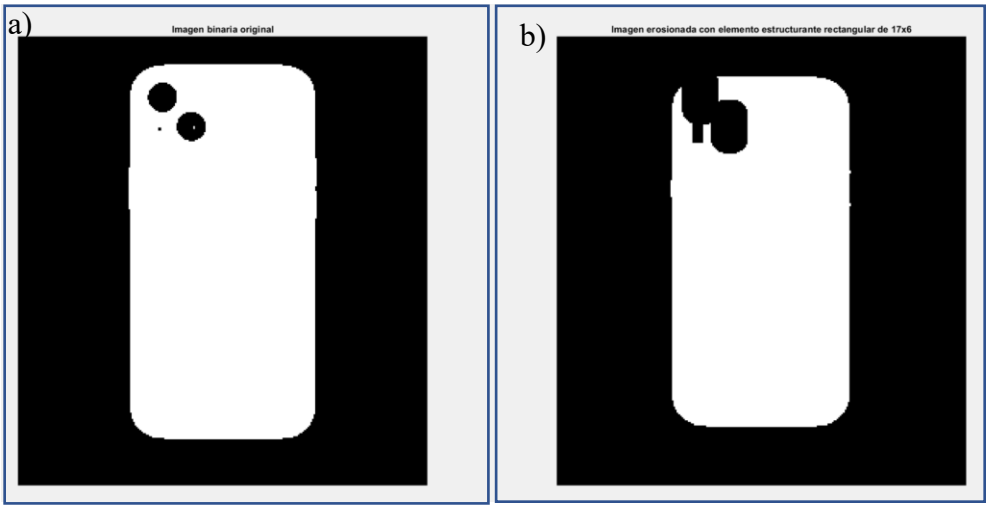

Figura 1.13 En a) se muestra la imagen original, en b) Se muestra la imagen resultante de la erosión con un elemento estructurante de 6x17, imagen recuperada de Google.com y procesada bajo Matlab con un kernel de 6x17.

1.7 Detección de bordes

La detección de bordes es una técnica importante en el análisis de imágenes y se utiliza para encontrar y destacar regiones en una imagen donde hay cambios significativos en la intensidad de los pixeles. Los bordes son áreas en la imagen donde los objetos o características importantes tienen una transición brusca en la intensidad de los pixeles. La detección de bordes es una forma de identificar estos cambios y destacarlos en la imagen.

Los métodos de detección de bordes incluyen la aplicación de filtros de gradientes, como el filtro de Sobel o el algoritmo de Canny, que destacan los bordes mediante la comparación de la intensidad de los pixeles vecinos.

Además de la detección de bordes, es importante realizar una suavización previa de la imagen para reducir el ruido y mejorar la precisión de la detección de bordes. La detección de bordes es una técnica esencial en muchas aplicaciones de visión por computadora, como la segmentación de imágenes, la identificación de objetos y la detección de formas.

1.7.1 Gradiente de una imagen digital

Los algoritmos de gradiente se basan en la idea de detectar cambios bruscos de intensidad en la imagen y utilizarlos para identificar bordes. La idea detrás de estos algoritmos es que los bordes son regiones donde hay un cambio drástico en la intensidad de los pixeles, lo que se puede detectar mediante la aplicación de un filtro de gradiente.

Hay dos tipos de gradiente: gradiente de intensidad y gradiente de dirección. El gradiente de intensidad se utiliza para medir el cambio de intensidad de la imagen en una dirección dada, mientras que el gradiente de dirección se utiliza para medir la dirección del cambio de intensidad.

Para el cálculo del gradiente de una imagen digital se implementa utilizando la magnitud del gradiente ∇f, la cual se entiende como el gradiente de *f* con coordenadas *x, y,* definido como una columna de un vector bidimensional [8].

$$\nabla f = grad(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad (1.15)$$

Este vector posee una característica geométrica crucial que indica la dirección en la que la función *f* experimenta su mayor variación en la posición (*x, y*).

Para la magnitud de este gradiente se tiene que al ser un vector se utiliza la fórmula de la magnitud de un vector. Por lo tanto, resulta en la ecuación (1.16).

$$M(x, y) = |\nabla f| = \sqrt{g_x^2 + g_y^2} \quad (1.16)$$

La propiedad geométrica del vector gradiente apunta hacia la dirección de mayor cambio de la función en una ubicación *(x, y)*. Se calcula la razón de cambio en la dirección del vector gradiente en *(x, y)*. La imagen resultante *M(x, y)* de este cálculo, conocida como la imagen del gradiente, es del mismo tamaño que la imagen original y permite variar en cada ubicación de pixel en la función.

Para el cálculo de la dirección del gradiente se tiene que:

$$\theta = \arctan\left(\frac{g_y}{g_x}\right) \quad (1.17)$$

En la Figura 1.14 se puede ver el resultado del cálculo del gradiente de una imagen donde a) es la imagen en escala de grises original y b) representa el cálculo del gradiente *M(x, y)*.

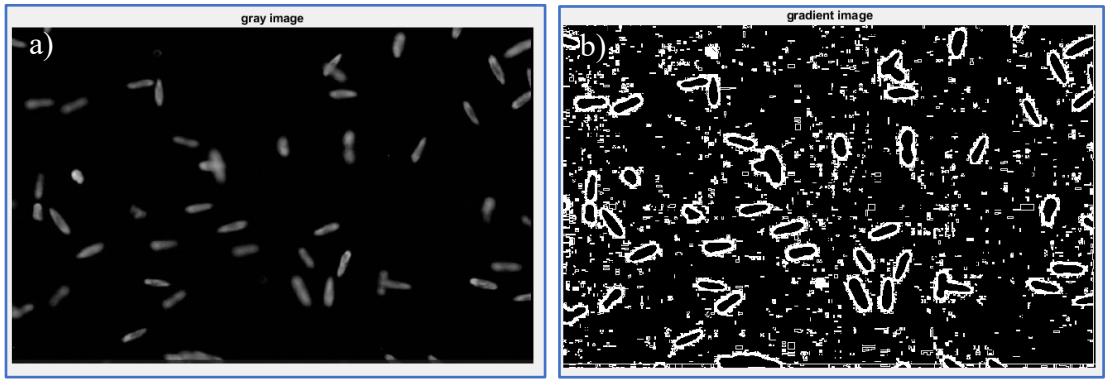

Figura 1.14 Se observa el cálculo del gradiente de una imagen digital

1.7.2 Algoritmo de Canny

El algoritmo de detección de bordes publicado en el año 1986 por Jonh Canny en la revista IEEE Transactions on Pattern Analysis and Machine Intelligence denominado "A Computational Approach to Edge Detection" [12] es un algoritmo ampliamente utilizado en el mundo de la visión por computadora.

El algoritmo de detección de bordes de Canny se basa en tres objetivos [13]:

1. Los bordes encontrados deben ser verdaderos y todos los bordes deben encontrarse. La probabilidad de encontrar un buen borde debe maximizarse y la de un borde falso debe minimizarse. Lograr este objetivo requiere una alta relación señal-ruido.
2. La ubicación del borde encontrado debe estar lo más cerca posible de la ubicación exacta.
3. No debe haber respuestas múltiples para un solo borde.

Este algoritmo funciona mediante un proceso de varios pasos que incluyen suavizado, cálculo del gradiente, histéresis y acumulación [14]. A continuación, se describen brevemente los pasos del algoritmo de Canny.

1. Suavizado: El primer paso consiste en suavizar la imagen mediante un filtro Gaussiano para reducir el ruido y mejorar la detección de bordes. Este filtro esta descrito por la ecuación (1.18) como:
$$G = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \quad (1.18)$$
Donde σ es la desviación estándar y se utiliza para controlar el grado de suavidad.
2. Cálculo del gradiente: Después de suavizar la imagen, el algoritmo de Canny calcula el gradiente para determinar la dirección y la intensidad de los bordes en la imagen. Esto se realiza a través de una combinación de operaciones de convolución y cálculo de las magnitudes y direcciones de los gradientes.
3. Histéresis: El siguiente paso consiste en aplicar la técnica de histéresis para determinar si un pixel es parte de un borde o no. Para hacer esto, se establecen dos umbrales, uno superior y otro inferior, y se decide si un pixel es parte de un borde si su valor de gradiente es mayor que el umbral superior (Conectividad de pixeles).
4. Acumulación: Por último, el algoritmo acumula los resultados y produce una imagen binaria con los bordes detectados, en la que los pixeles con valores de gradiente altos son considerados como parte de un borde y se marcan en la imagen binaria resultante

Si bien no es estrictamente necesario que el input sea una imagen binaria esta puede ayudar al algoritmo ya que es más ligera de procesar y esta imagen puede llevar operaciones elementales antes para lograr reducción de ruido y obtener bordes más suaves y exactos a la figura que se quiere resaltar.

Capítulo 2
Obtención y filtrado de las imágenes del Paramecium Tetraurelia

En este capítulo se tratará el tema del cultivo y la obtención de las imágenes del *Paramecium tetraurelia,* así como la parte de filtrado de las imágenes obtenidas para mejorar el contraste de la imagen y así dar pauta a una detección de bordes más limpia.

2.1 El Paramecium Tetraurelia

El *Paramecium Tetraurelia* es una especie de protozoo flagelado perteneciente al género Paramecium. Es un organismo unicelular que se encuentra comúnmente en aguas dulces y estuarios.

El *Paramecium Tetraurelia* se caracteriza por tener una forma alargada y ovalada, con una medida promedio de alrededor de 117-137μm de largo. Tiene una pared celular externa dura llamada película, que le brinda protección y soporte estructural. Además, cuenta con una serie de cilios, que le permiten desplazarse a través del agua.

El *Paramecium Tetraurelia* es un organismo heterótrofo, es decir, se alimenta de otras células y partículas orgánicas presentes en el medio ambiente. Utiliza sus cilios para capturar y atrapar a sus presas, y luego las digiere mediante el uso de enzimas especializadas.

Además, es un organismo muy útil para el estudio de la biología molecular y celular, ya que es relativamente fácil de cultivar y manipular en el laboratorio.

Se ha observado que este microorganismo tiene una relación de nado hacia adelante y hacia atrás al entrar con agentes externos (como estimulantes ópticos, térmicos, químicos, etc.) con sus potenciales de acción basados en calcio. Esto es relevante para el estudio ya que por esta razón los autores lo han llamado, la neurona nadadora [15].

Para el cultivo efectivo de este microorganismo, es necesario utilizar un medio de cultivo que promueva el crecimiento y reproducción de la bacteria *Enterobacter sp* la cual sirve de alimento y favorece la reproducción del *Paramecium tetraurelia*. Es importante seguir las instrucciones específicas para la preparación y uso del medio de cultivo para garantizar un cultivo efectivo y evitar problemas como contaminación. Además, es importante tener en cuenta que el control del pH (7.3) también que es crucial para el éxito del cultivo del microorganismo.

2.2 Medio de cultivo

Para fines de este trabajo de investigación se utilizó *Paramecium tetraurelia* (Carolina Biological Supply Co., Item 131560).

Este medio de cultivo se compone de KCl (4mM/L), $CaCl_2$ (1mM/L), MOPS (1Mm/L), MOPS (1mM/L), MgCl (200µM/L), Stigmasterol (5mg en 1 mL de etanol 70%) y Casaminoacidos (0.3gr/L). Para el ajuste del pH a 7.3 se usa NaOH, La cantidad tiene que ser lo suficiente para llevar el pH a ese valor deseado.

2.3 Arreglo experimental

Para este arreglo experimental se utilizó un microscopio óptico digital de la marca Dino-Lite modelo 5MP Edge AM7115MZTel cual tiene una ampliación de 10x-220x, la cual permite obtener videos con una resolución de 2592x1944 px a 30 frames por segundo (FPS) permitiendo ser los ojos del sistema de visión por computadora.

A pesar de que el microscopio digital ya tiene su propia iluminación, esta proporciona mucho ruido al resultado de la imagen final, por lo que se usó un anillo led con lo cual permitió una imagen más limpia y contrastada del *Paramecium tetraurelia.* En la Figura 2.1 se muestra el resultado con las diferentes alternativas de iluminación justificando el uso del anillo led.

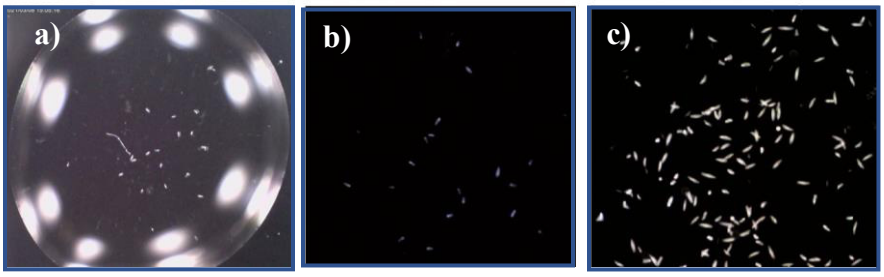

Figura 2.1. Se muestra el resultado con diferentes fuentes de iluminación a) Iluminación propia del microscopio óptico. b) Iluminación con una linterna. c) Iluminación con el anillo de leds

Para los estímulos ópticos se utilizó un diodo láser multimodo de 808 nm (Thorlabs, Inc., L808P1000MM), 1000mW, con un colimador LTN330-A, la cual permite la irradiación del microorganismo provocando la reacción de nado paranoico que se busca. Es importante mencionar que el colimador debe de ser ajustado al área del spot del láser deseado.

Para lograr el fondo que permitiera el contraste adecuado para la detección del microorganismo se utilizó una cartulina negra la cual se montó en cima del láser/colimador como se muestra en la Figura 2.2

Figura 2.2 Se muestra el fondo el cual sirve para obtener las mejores imágenes del Paramecium tetraurelia.

En la Figura 2.3 se muestra el diagrama del arreglo experimental. En particular, la Figura 2.3ª muestra el arreglo experimental esquemáticamente; mientras que la Figura 2.3b muestra una fotografía del arreglo montado en una mesa óptica. Se utiliza también un filtro pasa bajas (Thorlabs, Inc., FESH0750) para la protección del sensor del microscopio óptico ya que al irradiar con luz láser directamente esta se puede dañar con mucha facilidad.

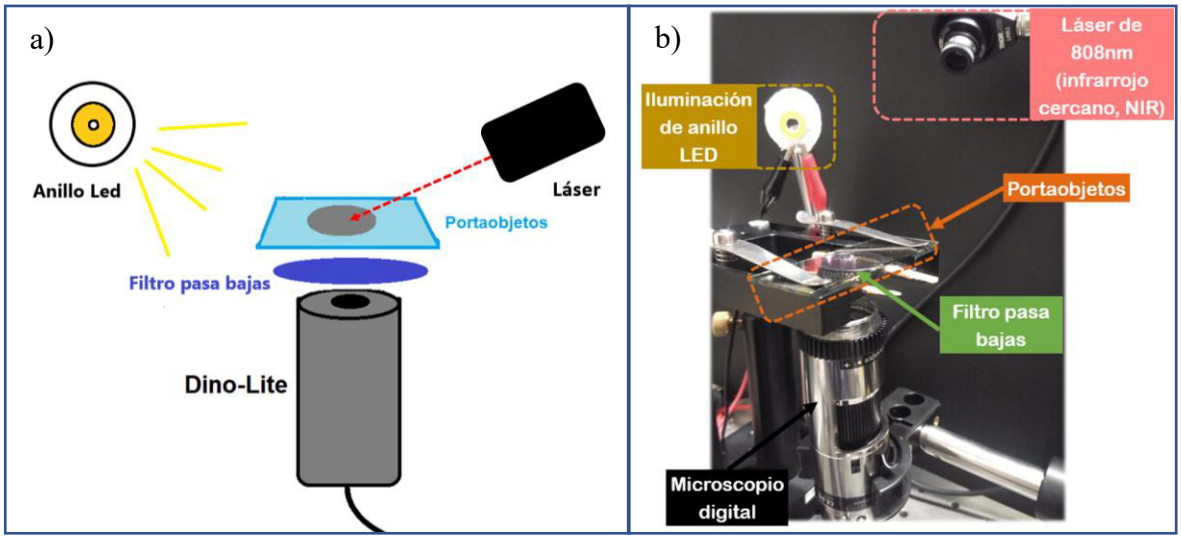

Figura 2.3. Arreglo experimental. a) Muestra el diagrama a seguir para realizar el arreglo experimental. b) Se muestra el arreglo experimental ya realizado.

Una vez realizado el arreglo se pueden obtener imágenes muy limpias de buen contraste, los cuales son suficientes para la obtención de imágenes fáciles de procesar.

En la Figura 2.4 se muestran las imágenes finales del arreglo experimental.

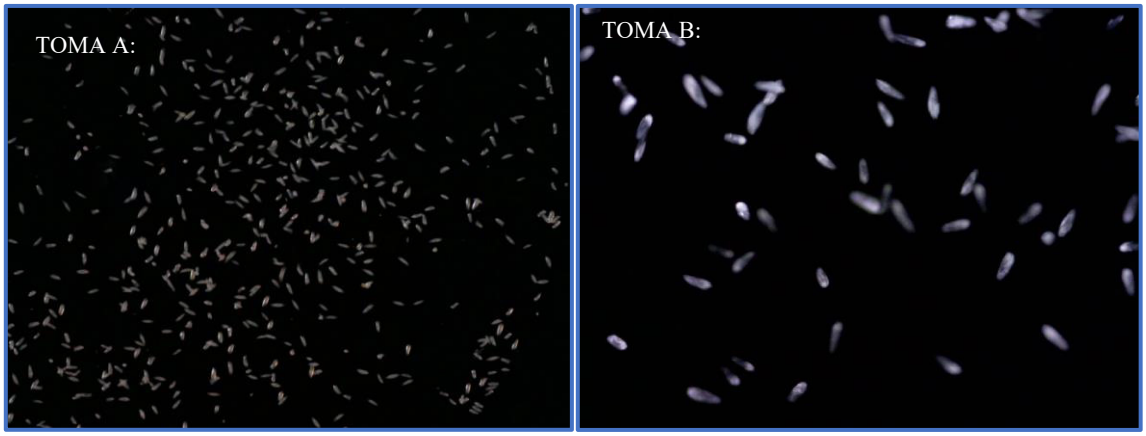

Figura 2.4 En la toma A se muestra una imagen tomada del Paramecium con un zoom menor al de la toma B

Las imágenes seleccionadas serán del estilo de la toma B dado a que se muestran de mejor manera las células de *Paramecium Tetraurelia*.

2.4 Obtención de imágenes de células del Paramecium Tetraurelia

Para la preparación de la escena se buscó la mejor manera de que las imágenes fueran limpias para disminuir el número de filtraciones y el margen de error de la detección al momento de realizar el procesamiento de la imagen con algoritmos de visión por computadora. Lo ideal para la toma de la imagen y el muestreo es un fondo negro contrastando de la mejor manera posible el microorganismo. Además, se utilizó una cámara de alta resolución y una lente de alta calidad para asegurar la claridad y precisión de las imágenes capturadas. También se utilizó un sistema de iluminación controlado para minimizar la distorsión de la imagen y para ajustar el nivel de iluminación según sea necesario.

Utilizando una adecuada técnica de preparación y un equipo óptico de alta calidad, es posible obtener imágenes muy nítidas y de buen contraste. Esto es especialmente importante para el procesamiento de las imágenes utilizando algoritmos de visión por computadora, ya que una buena calidad de imagen es crucial para obtener resultados precisos. Además, la obtención de imágenes de alta calidad también es útil para la observación y análisis manual de las muestras, ya que permite una mejor visualización de los detalles y características del microorganismo.

2.4.1 Obtención y limpieza de la muestra

Debido a la naturaleza del *Paramecium tetraurelia* este también tiene desechos como todos los seres vivos. En consecuencia, requieren de un proceso de selección de la mejor muestra posible para obtener imágenes más claras del *Paramecium.*

Para ello se toma una muestra 1ml de dos tubos de cultivo que estén lo más limpio posible. Después se trasladan a un tubo falcón nuevo y estéril, para luego centrifugarlos por 2 minutos a 3000 RPM. Una vez pasados los 2 minutos se toma el volumen experimental del fondo del tubo falcón, esto es así debido a que el *Paramecium* tiende a acumularse en el fondo del tubo después del proceso de centrifuga. En la Figura 2.5 se muestra un diagrama del proceso para la limpieza de la muestra.

Figura 2.5 Diagrama de limpieza de la muestra experimental

2.4.2 Correlación preliminar entre láser y nado del Paramecium tetraurelia

Se tomaron diferentes imágenes del comportamiento del *Paramecium* para su posterior análisis, cada una variando la potencia del láser.

Paralelo a esto se realizó un estudio que tiene como objetivo medir la fluencia del láser esto con el fin de corroborar la potencia del láser por unidad de área y así tener más información con la cual respaldar los análisis finales obtenidos por medio del software.

la fluencia del láser puede ser calculada por medio de la ecuación (2.1)

$$F = \frac{(P)(T)}{A} \qquad (2.1)$$

Donde F es la fluencia en Ws/m^2, P es la potencia del láser en W y T es el tiempo de exposición en S.

En la Figura 2.6 se observa el área del spot del láser (hay que recordar que está regulado por el colimador), esto como un área base para el cálculo de su fluencia en esa medida, dada como 16.04 mm^2.

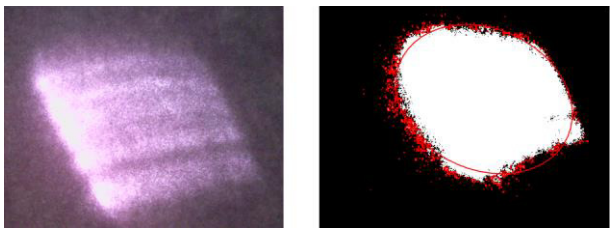

Figura 2.6 Área del spot de 16.04 mm²

Para este punto se tomaron diferentes videos con diferentes corrientes inducidas. Un video del nado normal del *Paramecium* y 3 videos con 900mA, 1000mA y 1100mA de corriente inducido al láser. Estas muestras deberían de ser suficientes como un primer acercamiento a la fase de preprocesamiento de la imagen.

2.5 Filtrado de las imágenes del Paramecium tetraurelia (Fase de preprocesamiento de la imagen).

En esta sección se estudia la fase de preprocesamiento/filtrado de las imágenes obtenidas en la sección 2.4 de este trabajo de investigación. Esta parte es una de las más importantes del trabajo debido a que es una fase previa para hacer el análisis estructural de una imagen binaria (Capítulo 1).

2.5.1 Escala de grises

Para este proceso se tiene que un video es un conjunto de imágenes digitales secuenciales en un periodo n de tiempo, con lo cual se pueden procesar *n* imágenes y esto es precisamente lo que se realiza en esta sección, esto con el fin de poder acercarse a la binarización de la imagen. La conversión de una imagen RGB a una escala de grises se realiza mediante la combinación de los tres canales de color que conforman la imagen. Los tres canales de color, rojo, verde y azul son combinados mediante un peso específico para crear una representación en escalas de grises de la imagen. Este proceso es importante ya que la información en una escala de grises es más fácil de procesar y analizar que una imagen en RGB. Además, la escala de grises tiene menos información que una imagen en RGB, lo que facilita el procesamiento y acelera los tiempos de ejecución. La conversión de una imagen RGB a una escala de grises es fundamental para el análisis de imágenes y es un paso importante en el procesamiento de imágenes en este trabajo. En código con la librería opencv (usada en este trabajo de investigación) se realiza con la función cvtColor, la cual permite convertir una imagen RGB a escala de grises mediante la ecuación [16] (2.2).

$$\text{RGB to Gray: } Y = 0.299R + 0.587G + 0.114B \qquad (2.2)$$

La ecuación (2.2) se vale de un concepto llamado *luma* el cual es un componente que funge como el componente representativo de la de la luminosidad. Los coeficientes se basan en la sensibilidad relativa del ojo humano a diferentes longitudes de onda de luz.

La función requiere de tres argumentos cuando se usa en C++: el primer argumento es la imagen (frame) a procesar, el segundo es el dato *Mat* destino del procesamiento y el tercer argumento es el tipo de conversión deseado, en este caso es la conversión a escala de grises (COLOR_BGR2GRAY). En la Figura 2.7 se tiene la muestra de la conversión donde en a) es el input frame y b) es el output frame.

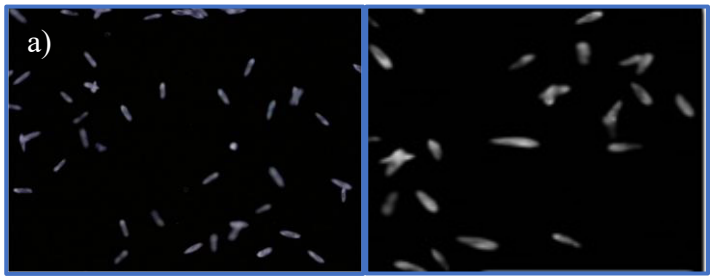

Figura 2.7 Conversión de RGB a escala de grises para una mejor eficiencia del procesamiento del frame

2.5.2 Binarización de la imagen (Threshold)

Como se vio en el capítulo 1, la binarización es un proceso con el cual podemos normalizar los pixeles en valores de entre 0 y 1 para indicar a la computadora si lo que está pasando en cámara es un objeto o no. Esta técnica es útil en el proceso de análisis de imágenes y se utiliza en muchas aplicaciones, como la detección de objetos, la segmentación de imágenes y la compresión de imágenes. La binarización se realiza mediante el uso de un umbral, donde los pixeles con intensidad de luz por encima del umbral se asignan a 1 y los demás se asignan a 0. Este proceso simplifica la imagen y facilita la tarea de la computadora al trabajar con solo dos valores. Para este proyecto se está utilizando un umbral de 50 a 255 dado que este umbral en fase de pruebas dio como resultado una imagen resultante con *Paramecium Tetraurelia* más completo, permitiendo tener un análisis estructural del organismo lo más exacto posible.

Estos valores de umbral se utilizan en la función *threshold* como argumentos ya que es un indicativo de el rango de intensidad de pixeles que tiene que existir en la imagen a escala de grises para asignar el valor de 1 (que en este caso es el primer plano de la imagen)

El algoritmo utilizado es conocido dentro de la librería de OpenCV como *binary* y este se utiliza con la flag *THRESH_BINARY* dentro de la función *threshold.* Como se muestra en la ecuación[17] (2.3)

$$dst(x,y) = \begin{cases} 1 & if\ src(x,y) > thresh \\ 0 & otherwise \end{cases}$$

El resultado de la binarización de una imagen se muestra en la Figura 2.8

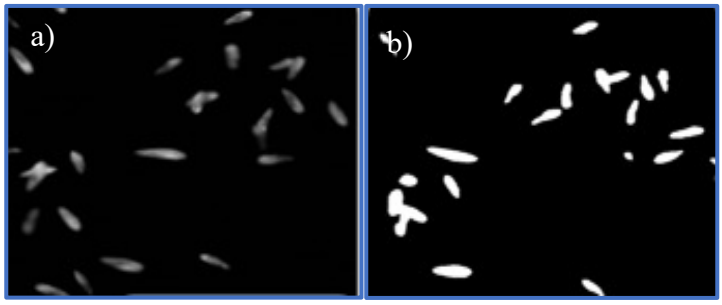

Figura 2.8 Paso de la imagen en escala de grises a) hacia una imagen binaria partiendo de un umbral de 50 a 255

2.5.3 Operaciones morfológicas de la imagen resultante

Como se vio en el capítulo 1, las operaciones morfológicas son de suma ayuda al momento de trabajar con imágenes binarias ya que permiten realizar transformaciones en la forma de los objetos binarizados y mejorar su representación. Algunas de las operaciones morfológicas más comunes incluyen la erosión, la dilatación, la apertura y el cierre.

La erosión reduce el tamaño de los objetos y suprime los detalles no importantes. Por otro lado, la dilatación aumenta el tamaño de los objetos y llena los huecos. La apertura combina la erosión seguida de una dilatación y es útil para eliminar el ruido y suavizar las bordes. Por último, el cierre combina una dilatación seguida de una erosión y es útil para llenar los huecos y unificar los objetos separados.

Es importante mencionar que para estas operaciones se utiliza la función *morphologyDefaultBorderValue()* que se usa para establecer el valor por defecto para los bordes de la imagen durante las operaciones morfológicas.

En OpenCV, durante las operaciones morfológicas se realiza una dilatación o erosión de la imagen utilizando un elemento estructurante, y los pixeles en los bordes de la imagen pueden ser afectados por el tamaño del elemento estructurante. Para evitar problemas con los bordes, es posible establecer un valor por defecto para los pixeles en los bordes de la imagen antes de realizar la operación morfológica. Esta función ayuda mucho a que los resultados del análisis estructural sean precisos y consistentes.

Para este proyecto se realiza primero una dilatación a la imagen binaria con ayuda de un elemento estructurante elíptico inscrito en un rectángulo de tamaño de 6 px × 6 px este elemento en opencv se denomina como MORPH_ELLIPSE.

Una vez con la imagen dilatada se puede observar en la Figura 2.9 que hay elementos que no sirven para el análisis estructural el cual podrían contaminar el resultado final.

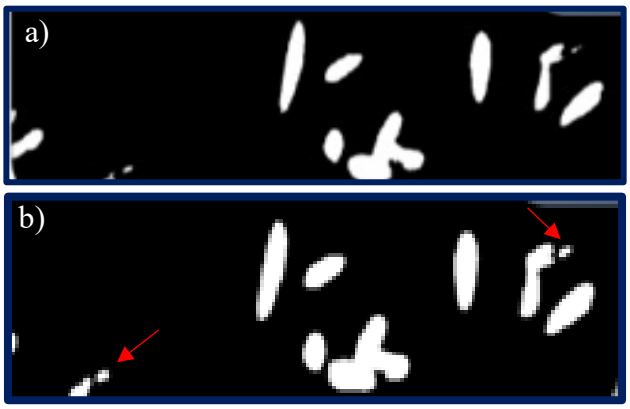

Figura 2.9 En a) Se observa la imagen original sin operaciones morfológicas, en b) Se observan puntos donde pixeles se han amplificado después de implementar la operación de dilatación.

Para dejar más limpia la imagen se procede a realizar una erosión a la imagen, esta con un elemento estructurante cuadrado de 10 px × 10 px. En la Figura 2.10 se observa como luego de un proceso de erosión se reducen de manera significativa las detecciones las cuales no son relevantes para el proceso de análisis, pero aun la imagen muestra pequeños puntos de ruido. Si se hace una erosión más agresiva se tiene que la estructura del *Parameciu*m *Tetraurelia* se vería radicalmente comprometida ya que empezaría a ser más pequeña y filosa la estructura, haciendo cada vez más notorios los dientes de sierra de los pixeles de la imagen binaria.

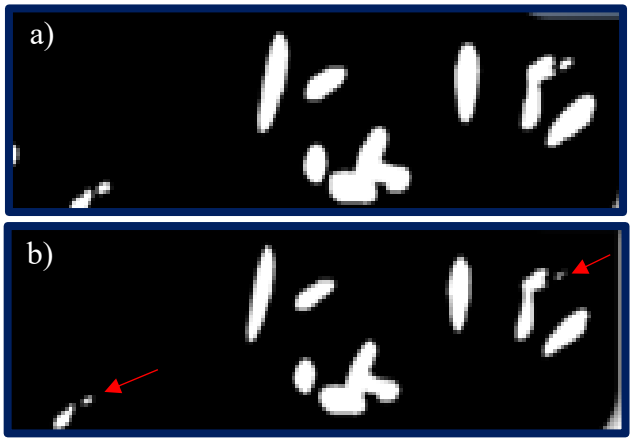

Figura 2.10 en b) se observan aun los puntos no deseados en la imagen, pero se puede apreciar una reducción significativa con respecto de a)

Por lo cual la siguiente operación morfológica usada para la imagen es la apertura con la cual como ya se mencionó se puede dejar la estructura más limpia y eliminar los puntos de ruido que claramente no se desean.

Para esta operación se utilizó un elemento estructurante elíptico inscrito en un rectángulo de tamaño de 10 px × 10 px lo cual permitió lograr una imagen adecuada para el proceso de análisis estructural. En la Figura 2.11 se puede apreciar claramente como la apertura ha

eliminado el ruido presente en la imagen y ha dejado la estructura más limpia y clara. La elección del elemento estructurante elíptico inscrito en un rectángulo de tamaño 10 px * 10 px ha resultado ser adecuada para lograr este resultado, ya que ha permitido suavizar los bordes de los objetos y preservar su forma general.

Este resultado es esencial para el análisis estructural de la imagen, ya que permite a la computadora identificar con mayor claridad los objetos presentes en la imagen y su relación entre ellos. Además, la eliminación del ruido mejora la precisión de los resultados de análisis estructural.

Figura 2.11. Se observa el resultado final del filtrado de la imagen la cual se nota más limpia y con bordes más cercanos a la forma de un Paramecium Tetraurelia sin puntos no deseados como en la Figura 3.10 b)

Capítulo 3
Análisis estructural de la imagen binaria

En el capítulo anterior se describió el proceso de obtención de la muestra de *Paramecium Tetraurelia* y la creación de los videos a ser analizados. Se detalló cómo se obtiene la imagen binaria y se realizaron operaciones morfológicas para mejorar la calidad de la imagen y disminuir el ruido en ella. En este capítulo, se describirán los métodos utilizados para extrapolar la elipse correspondiente a cada célula de *Paramecium Tetraurelia* detectado en cada frame.

Estos métodos incluyen técnicas de detección de bordes, análisis estructural, así como algoritmos de ajuste de elipses para obtener una representación precisa y estable de la forma de la célula en cuestión.

El objetivo final es tener una base de datos completa y detallada de las elipses correspondientes a cada *Paramecium Tetraurelia* en el video. La información obtenida será esencial para entender mejor la biología de estos organismos y para mejorar la comprensión de cómo interactúan con su entorno.

3.1 Detección de bordes para análisis estructural

Para este trabajo de investigación, se ha elegido utilizar el algoritmo de Canny para el resaltado de los bordes en las imágenes de Paramecium. El algoritmo de Canny es ampliamente utilizado en el procesamiento de imágenes debido a su alta precisión y bajo nivel de errores. Este algoritmo utiliza una combinación de suavizado, detección de gradientes y umbralización para encontrar los bordes en la imagen (Como se menciona en el capítulo 1. Después de aplicar el algoritmo de Canny, se podrían extraer características importantes de las células de Paramecium .

Para lograr un análisis estructural efectivo, es importante configurar adecuadamente los parámetros del algoritmo de Canny. Este algoritmo, implementado en la función "Canny" de OpenCV, es una herramienta útil para detectar bordes en imágenes digitales. La configuración óptima de los parámetros dependerá de las características específicas de cada imagen y de los requisitos del análisis que se pretende realizar. Al ajustar adecuadamente los parámetros de Canny, es posible lograr un rendimiento óptimo que permita detectar con precisión los bordes estructurales y otros detalles relevantes de la imagen. Es importante tener en cuenta que la configuración de los parámetros de Canny puede ser un proceso de prueba y error, por lo que se recomienda experimentar con diferentes valores hasta encontrar la combinación adecuada para cada caso.

Para los valores umbrales de la función de Canny (en OpenCV) se probaron distintos umbrales entre 0 y 255 y se analizaron visualmente las imágenes resultantes. De acuerdo con este análisis heurístico, los valores óptimos encontrados fueron:

1. Treshhold 1 (Primer umbral para el proceso de histéresis): 50
2. Treshhold 2 (Primer umbral para el proceso de histéresis): 100

Estos valores de umbrales generan imágenes con bordes definidos correctamente, siendo bordes completos y de alto contraste lo cual es fundamental para el análisis estructural de la imagen binaria.

Cabe resaltar que si bien estos valores son óptimos para el tipo de imágenes obtenidas experimentalmente en este trabajo, los valores óptimos del umbral pueden ser distintos para las imágenes tomadas con diferente iluminación o magnificación. Al usar los resultados del presente trabajo para otra aplicación será necesario realizar una optimización a los valores del umbral.

Además de estos valores, también se pueden incluir otros dos parámetros importantes: el tamaño de apertura para el operador de Sobel y el gradiente L2. En este caso, se ha decidido dejar estos parámetros en sus valores por defecto, ya que los resultados no muestran una gran diferencia en el resultado final.

Es importante mencionar que el gradiente L2 se puede calcular utilizando la ecuación (3.1), que consiste en la suma de las magnitudes de las derivadas en las direcciones x, y [18]:

$$\text{Edge\_Gradient}(G)=|Gx|+|Gy| \quad (3.1)$$

En la Figura 3.1 se muestran dos imágenes: a) la imagen original binaria y b) el resultado obtenido tras aplicar el algoritmo de Canny a la imagen original. La imagen original binaria, a), presenta regiones de interés blancas y fondo negro. Por otro lado, el resultado obtenido tras aplicar el algoritmo de Canny a esta imagen, b), presenta los bordes detectados de las regiones de interés en color blanco y el fondo en color negro.

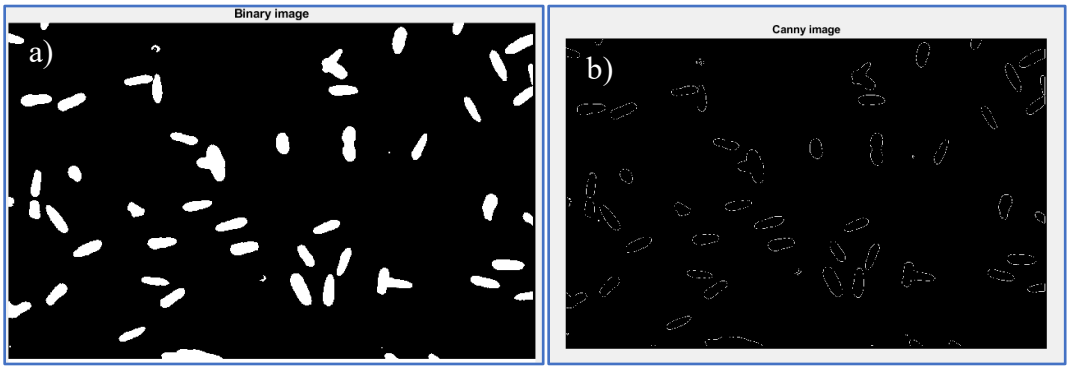

Figura 3.1 En a) se muestra la imagen binaria input para el algoritmo de Canny. En b) Se muestra la imagen resultante del algoritmo de Canny

La detección de bordes utilizando el algoritmo de Canny ha permitido identificar con precisión las regiones de interés en la imagen analizada. Estos resultados son cruciales para un análisis estructural efectivo, ya que permiten identificar de manera clara las características de la estructura analizada. A partir de estos resultados, se puede concluir que la detección de bordes mediante el algoritmo de Canny es una herramienta útil y eficaz para el análisis estructural de imágenes digitales.

3.2 Análisis estructural

En esta sección, abordaremos el análisis estructural, una fase fundamental para obtener información relevante de la imagen resultante en el proceso de detección de bordes. El análisis estructural nos permite comprender la relación entre los elementos presentes en la imagen, su posición y su influencia en el conjunto. Este análisis es crucial para extraer datos significativos que se utilizarán en la posterior toma de decisiones y en la implementación posterior de la interpolación de una elipse.

3.2.1 Ordenamiento topológico por seguimiento de borde

El algoritmo de ordenamiento topológico por seguimiento de borde a grandes rasgos es un algoritmo utilizado en teoría de grafos para ordenar un grafo dirigido acíclico (DAG) linealmente de tal manera que, para cada arista dirigida del grafo, el nodo de origen aparece antes que el nodo de destino en la secuencia lineal. Este algoritmo se basa en un método de exploración de los bordes del grafo.

El algoritmo comienza seleccionando un nodo de inicio en el grafo que no tiene aristas entrantes. A continuación, se sigue el borde saliente de ese nodo y se elimina la arista que lo conecta con su nodo destino. Luego se comprueba si el nodo destino tiene alguna arista entrante restante. Si no hay más aristas entrantes, ese nodo se agrega a la secuencia lineal del orden topológico y se convierte en el nuevo nodo de inicio. Si todavía hay aristas entrantes, se sigue el borde saliente del nodo destino y se repite el proceso.

Este proceso se repite hasta que se han agregado todos los nodos del grafo a la secuencia lineal. Si en algún momento del proceso no hay nodos disponibles para continuar, entonces el grafo contiene al menos un ciclo y no se puede obtener un orden topológico [19].

Si se ve este grafo como el arreglo de pixeles se tiene que el algoritmo de ordenamiento topológico por seguimiento de borde comienza seleccionando un pixel de inicio y se sigue el borde de los pixeles adyacentes, marcando cada pixel visitado. El proceso se repite hasta que se han visitado todos los pixeles del contorno del objeto. El resultado es un arreglo de valores que describe la forma y el tamaño del objeto detectado.

En caso de que el algoritmo detecte dos o más pixeles adyacentes al pixel actual se seleccionará el siguiente pixel adyacente que se encuentre en sentido contrario al punto de entrada. Este proceso continúa hasta que se llega al punto de partida y se completa el contorno.

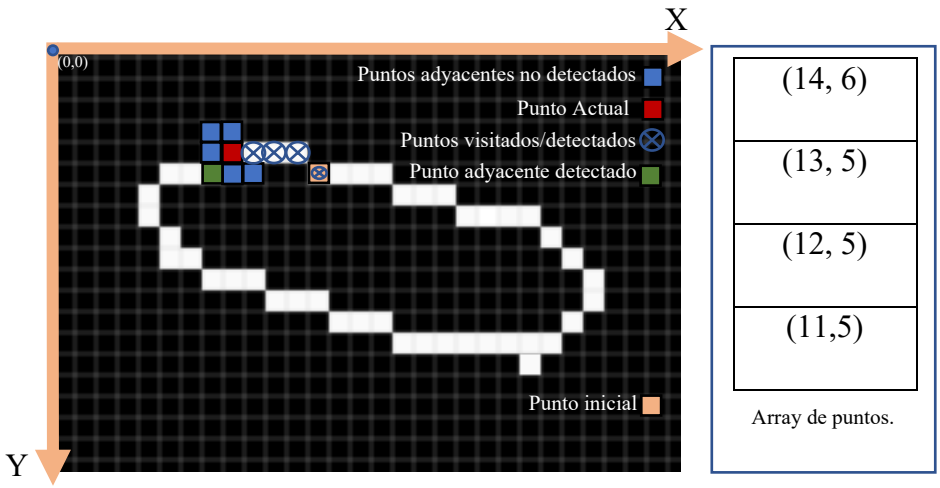

Figura 3.2 Se puede observar como el algoritmo recorre la figura punto por punto almacenando los resultados en un array de puntos.

En la Figura 3.2 se muestra una guía más visual de cómo actúa el algoritmo de detección de contornos sobre la imagen binaria obtenida mediante Canny. En primer lugar, se parte de la imagen binaria que contiene los bordes detectados por el algoritmo de Canny. Luego, se aplica el algoritmo de ordenamiento topológico por seguimiento de borde para obtener una lista de puntos ordenados que representan el contorno de la figura.

Esto se hace para toda la imagen por lo que se tiene un vector de vectores que almacena cada punto por contorno detectado, dando como resultado una estructura similar a el siguiente arreglo (este arreglo puede interpretarse como un vector de vectores):

[[[10, 20], [12, 25], [14, 30], [16, 35], [18, 40]],

[[100, 150], [102, 155], [104, 160], [106, 165], [108, 170], [110, 175]]]

En C++ al momento de declarar una estructura capaz de contener esto se usa el template "*Vector"* El cual permite un fácil manejo de los datos que arroja el algoritmo.

La estructura quedaría de la siguiente manera:

```
std::vector<std::vector<cv::Point>> cnts;
```

Donde el tipo de dato *Point* es un tipo de dato que la librería OpenCV tiene para manejar los puntos coordenados `<Number, Number>` (valores numéricos).

Una vez almacenados los datos, el siguiente paso es realizar la interpolación de la elipse, que consiste en encontrar la mejor elipse que se ajusta a los datos obtenidos.

3.3 Aproximación a una elipse

El proceso de aproximación a una elipse es fundamental en el análisis estructural de imágenes, ya que permite obtener información precisa sobre la forma y orientación de los objetos presentes en la imagen. Es un tema de gran relevancia en el contexto del análisis morfológico del *Paramecium Tetraurelia*, ya que de aquí se extrae la información de salida ("*output*") que permite interpretar y comprender los datos obtenidos en la etapa de procesamiento de la imagen.

Para este fin, se utilizan los datos obtenidos en la fase de análisis estructural para alimentar el algoritmo utilizado por OpenCV en su función de fitEllipse el cual utiliza el algoritmo de ajuste de mínimos cuadrados lineales o LADEA ("Linear Algebraic Distance Estimation Algorithm") por sus siglas en inglés.

3.3.1 Algoritmo de ajuste de mínimos cuadrados lineales

El algoritmo de ajuste de mínimos cuadrados lineales es una técnica matemática utilizada para encontrar la mejor línea recta o el mejor ajuste de una forma geométrica a un conjunto de datos. Este algoritmo se utiliza en diversas aplicaciones, desde la estadística hasta la visión por computadora y el procesamiento de imágenes.

En el caso de la detección de formas geométricas en imágenes, el algoritmo de ajuste de mínimos cuadrados lineales puede utilizarse para ajustar una elipse a un conjunto de puntos que representan el contorno de un objeto. Esto puede proporcionar información valiosa sobre la forma y orientación del objeto y permitir la segmentación y análisis de la imagen de manera más eficiente.

La función de OpenCV FitEllipse funciona de la siguiente manera [21]:

1. Se obtienen los puntos que forman el contorno del objeto de interés en la imagen.
2. Se calcula el centroide del conjunto de puntos.
3. Se calculan las desviaciones de los puntos con respecto al centroide y se construye la matriz de covarianza de los datos. La matriz de covarianza es una matriz simétrica que representa cómo están relacionadas las diferentes variables (en este caso, las coordenadas x e y) en el conjunto de datos.
4. Se calculan los eigenvalores y eigenvectores de la matriz de covarianza. Los eigenvectores son vectores ortogonales que indican las direcciones principales de variabilidad en los datos. Los eigenvalores representan la varianza de los datos a lo largo de cada eigenvector.

5. Se seleccionan los dos eigenvectores asociados con los dos eigenvalores más grandes. Estos dos eigenvectores definen los ejes principales de la elipse.
6. Se calcula el ángulo de inclinación de la elipse a partir del ángulo que forma el eigenvector correspondiente al eigenvalor más grande con el eje x de la imagen.
7. Se calculan los semiejes de la elipse a partir de la longitud de los eigenvectores correspondientes a los dos eigenvalores más grandes.
8. Se calculan las coordenadas del centro de la elipse utilizando el centroide calculado en el paso 2 y la dirección del eigenvector correspondiente al eigenvalor más grande.
9. Se construye la ecuación paramétrica de la elipse utilizando las coordenadas del centro, los semiejes y el ángulo de inclinación. Como se muestra en la ecuación (3.1) y (3.2)

$$x = x_0 + a\cos(\theta)\cos(t) - bsen(\theta)sen(t) \quad (3.1)$$

$$y = y_0 + asen(\theta)\cos(t) + bcos(\theta)sen(t) \quad (3.2)$$

Donde 'x_0' y 'y_0' son las coordenadas del centro de la elipse, a y b son los semiejes de la elipse, (θ) es el ángulo de inclinación y t es un parámetro el cual varia de 0 a 2π

10. Se obtienen los puntos de la elipse evaluando la ecuación paramétrica en diferentes valores de t.

Una vez obtenidos los parámetros de la elipse ajustada mediante el algoritmo de ajuste de mínimos cuadrados lineales, se pueden obtener diversas medidas y características de la elipse. Por ejemplo, se puede obtener el centro de la elipse a partir de las coordenadas del punto medio de los puntos extremos de los ejes mayor y menor, así como la orientación de la elipse con respecto al sistema de coordenadas. Además, se puede calcular el área de la elipse y su perímetro, lo que proporciona información importante sobre la forma y el tamaño de la elipse. También se pueden obtener medidas como el eje mayor y menor, la excentricidad y la distancia focal, lo que permite una caracterización completa de la elipse.

Para este caso el punto de interés se encuentra en las posiciones centrales de cada uno de los microorganismos. En la Figura 3.3 se muestra el resultado del algoritmo de interpolación de elipses.

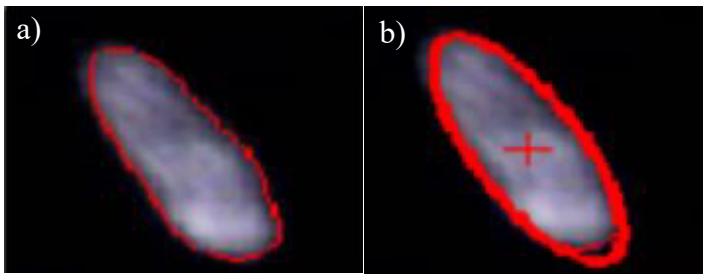

Figura 3.3 en a) se muestra los puntos calculados en la fase de análisis estructural. En b) se muestran los resultados de la interpolación de la elipse basado en el algoritmo LIN

3.4 Extracción y almacenamiento de datos

Como se explicó en la sección anterior, la interpolación de la elipse permite obtener sus propiedades matemáticas, lo que permite obtener información relevante sobre la posición del Paramecium Tetraurelia. En este estudio, se utilizan las propiedades de posición en el plano de la imagen, eligiendo el centro *(x, y)* y el ángulo θ como parámetros importantes, como se ilustra en la Figura 3.4.

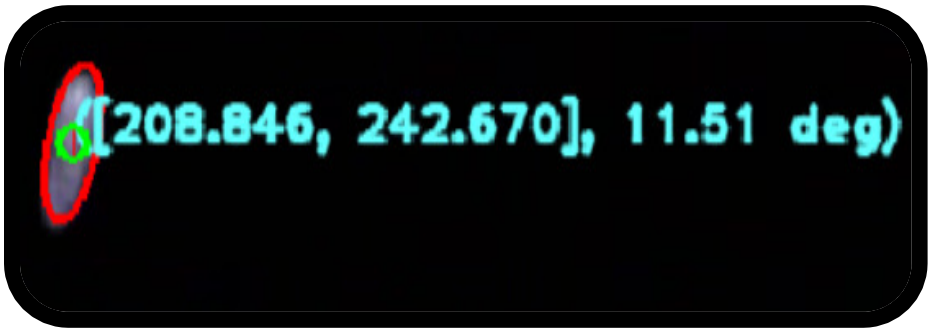

Figura 3.4 información extraída de la elipse sobrepuesta en el Paramecium Tetraurelia

Es importante destacar que la matriz de covarianza es un factor clave en el cálculo del ángulo de la elipse. Aunque la figura 3.4 muestra el ángulo en grados, al utilizar la matriz de covarianza se obtiene el resultado en radianes.

A continuación se explicará más a detalle como OpenCV Obtiene cada uno de los parámetros de la elipse.

3.4.1 Obtención del centroide de la elipse

El cálculo del centroide de la elipse interpolada se realiza utilizando la fórmula del centroide de un conjunto de puntos. En primer lugar, se obtienen los puntos que forman la elipse y se almacenan en una matriz. Luego, se calcula el promedio de las coordenadas en *(x, y)* de todos los puntos, lo que da como resultado el centroide de la elipse. Esta fórmula se basa en

la propiedad de que el centroide de un conjunto de puntos es el punto promedio de las coordenadas de todos los puntos [19], especificada en la ecuación (3.3).

$$C = \frac{1}{n}\sum_{i=1}^{n}(x_i, y_i) \quad (3.3)$$

Donde n es el número de puntos de la elipse y (x_i, y_i) son las coordenadas de cada punto de la elipse y C es el centroide (*x, y*).

Por lo que en cada frame se hará el cálculo de cada centroide de cara *Paramecium Tetraurelia,* detectado.

3.4.2 Obtención del ángulo de la elipse (matriz de covarianza)

Para la obtención del ángulo se utiliza la matriz de covariancia la cual describe la relación entre la posición de los puntos en la imagen y la forma y orientación de la elipse ajustada.

La matriz de covarianza de los puntos se calcula y se diagonaliza para obtener los vectores propios de la matriz, que corresponden a los ejes principales de la elipse ajustada. La dirección del eje principal que tiene la mayor varianza indica la orientación de la elipse.

Aquí la pregunta radica en: ¿Como es calculada la matriz de covarianza?

Para este fin se tienen dos conceptos de probabilidad las cuales son la varianza y la covarianza definidas por las ecuaciones 3.4 y 3.5 respectivamente

$$\left.\begin{aligned} Var(x) = \sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n-1} \\ Var(y) = \sigma^2 = \frac{\sum_{i=1}^{n}(y_i - \mu)^2}{n-1} \end{aligned}\right\} \quad (3.4)$$

Donde n es el número total de datos, x_i e y_i es el *i*- ésimo dato y μ es la media de los datos.

$$Cov(x, y) = \frac{\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)}{n} \quad (3.5)$$

Con esto conceptos la matriz de covarianza se define por medio de la ecuación (3.6)

$$\begin{bmatrix} Var(x) & Cov(x,y) \\ Cov(x,y) & Var(y) \end{bmatrix} \quad (3.6)$$

Una vez obtenida la matriz de covarianza se calculan los eigenvalores de esta misma matriz dada la ecuación (3.7)

$$|A - \lambda I| = 0 \quad (3.7)$$

Donde A es la matriz de covarianza (en este caso), λ es el eigenvalor e I es la matriz identidad.

Luego, el cálculo de los eigenvectores se encuentran resolviendo el sistema de ecuaciones lineales homogéneas asociado [22]:

$$(A - \lambda I)x = 0 \quad (3.8)$$

después se elige el eigenvector con el eigenvalor más alto y se calcula el ángulo entre el eigenvector y el eje x de la imagen para dar un ángulo final que es el que se encuentra en la elipse.

3.4.3 Almacenamiento de datos

Para el almacenamiento de los datos del programa concluido se pensaron muchas maneras de realizarlo, sin embargo se optó por un archivo de valores separados por comas (CSV).

El uso de archivos CSV es comúnmente aceptado como una forma conveniente y versátil de almacenar datos estructurados en una variedad de aplicaciones de software. Los archivos CSV permiten la separación de los datos mediante comas, lo que los hace fácilmente legibles y editables en cualquier editor de texto plano o programa de hoja de cálculo. Además, el formato CSV es compatible con una amplia variedad de lenguajes de programación y herramientas de análisis de datos, lo que lo convierte en una opción popular para el intercambio de datos entre sistemas.

Dado que se está hablando de una matriz de 3 dimensiones (vector de vectores de puntos) la opción que se tomo es dividir el output del software en 3 archivos CSV diferentes: Uno con los componentes X, Uno con los componentes en Y, y uno con el ángulo θ. Esta separación se muestra esquemáticamente en la Figura 3.5.

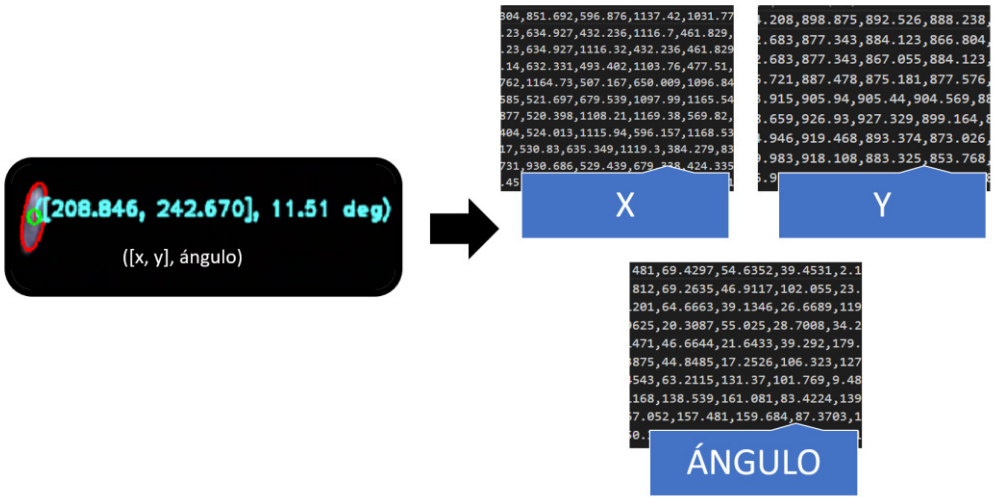

Figura 3.5 Ejemplo de la separación de los datos en 3 archivos diferentes

La razón para esta división es que se están registrando muchos fotogramas y cada salto de línea se toma como un dato por frame.

Por cada célula de *Paramecium Tetraurelia* se obtiene un valor separado por coma. Es decir, la cantidad de datos antes del salto de línea (\n) es el número de microorganismos detectados en ese frame especifico y juntando los tres archivos CSV se describe completamente sus posiciones y ángulo de inclinación de todo el video. Es importante aclarar que los datos solo son de los microorganismos que aparecen en la región de interés por cada frame. Como se muestra en la Figura 3.6

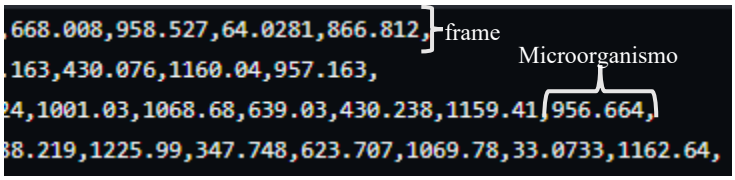

Figura 3.6. Detalle de los datos obtenidos por el programa desarrollado en este trabajo. En particular, la figura y muestra como está conformada la distribución de datos en cada frame.

Los datos almacenados en un archivo CSV pueden ser utilizados para realizar análisis estadísticos, generar gráficos y visualizaciones, realizar predicciones, entrenar modelos de aprendizaje automático, y mucho más. La separación de los datos en tres archivos CSV diferentes (componentes X, componentes Y, y ángulo θ) facilita la manipulación y análisis de los datos de manera independiente, lo que puede ser útil para realizar comparaciones entre diferentes variables o para realizar análisis específicos.

Capítulo 4
Conclusiones y trabajo a futuro

En este trabajo se ha presentado un método para la obtención de datos espaciales de células individuales de *Paramecium tetraurelia* mediante el uso de algoritmos de visión por computadora. Los resultados obtenidos demuestran la viabilidad de este enfoque para estudiar la trayectoria y el comportamiento de estos organismos unicelulares.

En este capítulo, se presentan las conclusiones generales del trabajo realizado, así como posibles áreas de mejora y futuras investigaciones en este campo. Además, se discute la utilidad de los datos obtenidos y las posibles aplicaciones de estos resultados para la comunidad científica y para aquellos interesados en el estudio del *Paramecium Tetraurelia*.

4.1 Ventajas del sistema de visión por computadora

En este trabajo se ha propuesto un software de detección automatizada de células del *Paramecium tetraurelia* mediante algoritmos de visión por computadora. La utilización de un software de este tipo presenta numerosas ventajas, tales como el ahorro de tiempo y recursos, una mayor precisión y consistencia en la detección de células, la posibilidad de análisis cuantitativo de los datos obtenidos, la escalabilidad del análisis y la aplicabilidad del software a diferentes áreas de investigación.

La detección manual de células individuales es un proceso laborioso y tedioso que puede requerir mucho tiempo y recursos humanos. Un software automatizado puede realizar esta tarea de forma rápida y precisa, liberando al personal de tareas más complejas y permitiendo un uso más eficiente de los recursos disponibles.

Además, el software propuesto permite una mayor precisión y consistencia en la detección de células, ya que no presenta errores humanos y puede detectar células individuales de forma precisa y consistente. Esto es especialmente importante cuando se manejan grandes cantidades de datos.

Otra ventaja importante de este software es su capacidad para proporcionar análisis cuantitativo de los datos obtenidos. Al detectar y seguir células individuales a lo largo del tiempo, el software puede proporcionar datos cuantitativos acerca de la posición de la célula y abre pauta a utilizar esa información sobre numerosos estudios acerca de su comportamiento y puede llegar a ser de gran utilidad en futuros estudios e investigaciones.

4.2 El software propuesto bajo GNU General Public License

GNU GPL (General Public License) es una licencia de software libre creada por el proyecto GNU. Esta licencia establece los términos y condiciones bajo los cuales se puede distribuir, modificar y utilizar el software. El objetivo de la GPL es garantizar que el software sea libre

y que cualquier persona tenga la libertad de usarlo, copiarlo, distribuirlo y modificarlo. La GPL se considera una licencia "copyleft", lo que significa que las modificaciones y derivados del software también deben ser licenciados bajo la GPL [24].

La elección de la licencia GNU GPL para el software propuesto se justifica por varias razones. En primer lugar, la licencia permite que cualquier persona pueda acceder y utilizar el software de forma gratuita, lo que es importante para promover la colaboración y el intercambio de conocimientos en la comunidad científica. Además, al ser una licencia de código abierto, permite que otros desarrolladores puedan acceder al código fuente del software y mejorar o personalizar el mismo para adaptarlo a sus necesidades específicas.

La licencia GNU GPL también proporciona protección legal para el software, lo que es importante para garantizar que el software se utilice de manera responsable y ética. Al establecer condiciones claras para el uso del software, la licencia ayuda a prevenir la explotación comercial no ética y asegura que el software siempre esté disponible para su uso en la investigación científica.

En general, la elección de la licencia GNU GPL para el software propuesto se justifica por su capacidad para fomentar la colaboración, el intercambio de conocimientos y la protección legal del software. Al hacer que el software sea libre y accesible para todos, se espera que el software pueda contribuir significativamente al avance de la investigación científica en el área de la detección de células de *Paramecium*.

4.3 Trabajo a futuro

Dado a que este proyecto de investigación es relevante presentar dos puntos fundamentales que podrían complementar de muy buena manera el software propuesto.

4.3.1 Analisis de celulas de Paramecium tetraurelia en tiempo real

Dada la versatilidad de fuentes que OpenCV tiene al momento de utilizar inputs de videos es posible trabajar con la idea de extraer las imágenes del microorganismo directamente del microscopio digital ya que este es detectado como una cámara digital. Esto es especialmente importante en áreas de investigación que requieren una respuesta rápida y una evaluación en tiempo real, como la microbiología ambiental o la investigación de medicamentos. Además, la capacidad de análisis en tiempo real permite la automatización de procesos de laboratorio y la optimización de la productividad y eficiencia en el manejo de datos.

Normalmente OpenCV toma los inputs de video bajo identificadores lo cual en el parámetro input de ejecución podría ponerse el identificador de la cámara conectada. Estos identificadores pueden variar según el sistema operativo y la plataforma utilizados. En general, los identificadores de cámara se asignan automáticamente por el sistema operativo en el orden en que se conectan las cámaras.

Al estar el parámetro input ligado a una clase *VideoCapture* dentro de la librería OpenCV esto es posible y si se complementa con el trabajo a futuro de implementar una interfaz gráfica mucho más amigable con el usuario se puede lograr una herramienta de detección bastante versátil para el estudio del comportamiento de este microorganismo.

4.3.2 Modificación de umbrales

La variación de los umbrales es una tarea fundamental en la detección de objetos mediante visión por computadora. La elección de los valores adecuados de umbralización es clave para el éxito de la detección, ya que puede influir significativamente en la precisión de la localización del objeto. Además, es importante considerar la iluminación ambiental, ya que puede cambiar la apariencia de los objetos y afectar su detección.

Para este trabajo, se han establecido umbrales específicos en la fase de binarización y en la técnica de Canny con el objetivo de asegurar una detección precisa y efectiva de las *paramecia* en los videos utilizados. Es importante destacar que estos umbrales han sido establecidos en función de las características específicas de los videos utilizados en este estudio, por lo que si se utilizan otros videos, puede ser necesario ajustar estos umbrales para lograr una detección óptima.

En la actualidad, los umbrales se encuentran definidos en el código de las líneas 118 y 131, lo que implica que se debe realizar una modificación manual si se desea ajustarlos. Sin embargo, se podría considerar la posibilidad de implementar en el futuro una interfaz gráfica que permita modificar estos umbrales de forma más sencilla y dinámica, por ejemplo, mediante el uso de un *trackbar*. De esta manera, se facilitaría el proceso de ajuste de los umbrales y se permitiría una mayor flexibilidad en la configuración del software propuesto.

Por último, es necesario enfatizar que la herramienta desarrollada durante este proyecto puede ser de gran utilidad en áreas de microbiología y electrofisiología de organismos ciliados. Esto es relevante ya que aporta una herramienta tecnológica robusta para estudiar el comportamiento de nado en organismos celulares simples que a su vez puede brindar información sobre la dinámica de su membrana celular. Si bien la herramienta computacional puede ser mejorada en varios sentidos, se puede concluir que representa un medio de análisis novedoso y fiable para futuros estudios estadísticos de parámetros como velocidad promedio y velocidad de giro.

Anexo: Función operativa del software

Compilación y ejecución del software

El software planteado está programado en su totalidad en C++, incluyendo la librería de OpenCV en su versión 4.5.0 para su funcionamiento.

Como dependencias de compilación/ejecución se requiere mingw de 64 bits y OpenCV precompilado, es decir, los binarios e includes de la librería.

Parámetros de configuración

En el repositorio de GitHub se encuentra un archivo[23] .bat el cual al momento de ejecutar (Windows) Hace una compilación directa bajo el compilador de g++ el cual se muestra su contenido a continuación.

```
g++ -o run main.cpp -I[~INCLUDES DIR] –L[~BINARIES DIR] -
llibopencv_core450 -llibopencv_highgui450 -llibopencv_imgcodecs450 -
llibopencv_imgproc450 -llibopencv_photo450 -llibopencv_video450 -
llibopencv_videoio450
```

Desglosando el comando se tiene que indicar en donde se encuentran los includes y los binarios de la librería de OpenCV previamente compilada. Después hay que indicar que módulos utilizara el compilador para general el archivo binario "run" lo cual son los argumentos que aparecen después de indicar los directorios.

Una vez compilado se pasa a la fase de ejecución con la cual se tienen diferentes argumentos los cuales se muestran a continuación:

```
run.exe input [params]
```

En el caso de input este tiene que contener el directorio de ubicación del video a procesar en formato wmv.

Si en params se inserta -? o -h o –help se desplegará en consola los diferentes parámetros de ejecución el proyecto los cuales se detallan a continuación.

- -d o –delay {Value} → Este parámetro indica el tiempo en el que los fotogramas transcurren siendo el valor por defecto el 1 el cual indica que la velocidad ira al tiempo que se grabó el video sin ninguna clase de delay.
- -s o –save → es una flag la cual indica si el software tiene que generar o no los archivos CSV de almacenamiento de datos.

- -fN o –file_name {FileName output name}→ Este parámetro le indica al software con que nombre tiene que guardar los CSV de salida. Esto para más fácil identificación de los archivos si es que se procesan varios videos, en caso de no especificar se asignara el nombre “No_Name” por default.

A continuación se muestra un ejemplo del comando de ejecución del software.

```
run.exe videos/1000w_swim.wmv -s -fN 1000wSwim_Data -d 1
```

En la Figura A1 se muestra la interfaz al momento de ejecutar el software propuesto en este trabajo de investigación.

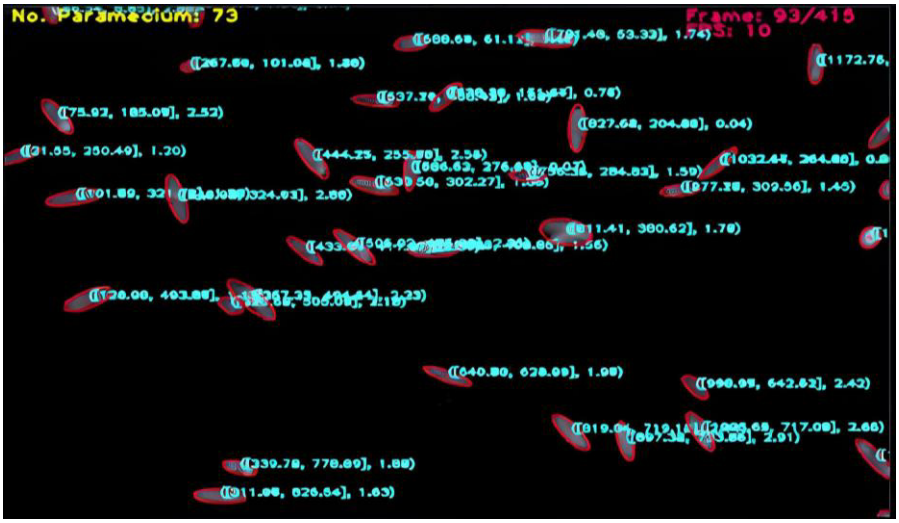

Figura A1. Captura de pantalla representativa de la interfaz completa con los datos de cada célula de Paramecium Tetraurelia detectado en el frame correspondiente

A su vez a la interfaz se le ha dotado con el número de *Paramecia* detectados (esquina superior izquierda) en el fotograma actual, así como un contador en un formato de “{current frame}/{total frames}” y los fotogramas por segundo (esquina superior derecha).

Se propone como trabajo a futuro la recompilación de la librería de OpenCV para permitir la integración de la librería Qt de C++ y así crear interfaces gráficas para la configuración de los umbrales de detección, esto facilitaría la integración del software propuesto en futuras investigaciones y su posible uso por parte de la comunidad científica y académica.

Referencias

[1] Lodh, S., Yano, J., Valentine, M. S., & van Houten, J. L. (2016). Voltagegated calcium channels of Paramecium cilia. Journal of Experimental Biology, 219(19), 3028–3038. https://doi.org/10.1242/jeb.141234

[2] Brette, R. (2021). Integrative Neuroscience of Paramecium, a "Swimming Neuron". eneuro, 8(3), ENEURO.0018-21.2021. https://doi.org/10.1523/eneuro.0018-21.2021

[3] Marketing. (2022, 30 mayo). Visión por Computador ✓ Qué es, Aplicaciones y Objetivos. EDS Robotics. Recuperado 22 de septiembre de 2022, de https://www.edsrobotics.com/blog/vision-computador-que-es/

[4] Klette, R. (2014, 20 enero). Concise Computer Vision: An Introduction Into Theory and Algorithms (2014 ed.). Springer.

[5] Forsyth, D. A. & Ponce, J. (2012). Computer Vision: A Modern Approach (2.a ed.). Pearson.

[6] Lennie, Peter. 2003. "The Physiology of Color Vision". Center for Neural Science. New York University, New York, NY 10003, USA.

[7] Gomes, J., Velho, L. & Sousa, C. M. (2012, 24 abril). Computer Graphics: Theory and Practice. A K Peaters. Cap. 5, pp. 119

[8] R. C. Gonzalez and R. E. Woods: Digital Image Processing (Prentice-Hall, New Jersey, 2002), Cap. 6, pp. 424; Cap. 3, pp. 142;

[9] Abhishek Yadav, & Poonam Yadav. (2009). Digital Image Processing: Vol. First edition. Laxmi Publications Pvt Ltd. Cap. 2, pp. 23-24

[10] Mathematical morphology: from theory to applications, Laurent Najman and Hugues Talbot (Eds). ISTE-Wiley. ISBN 978-1-84821-215-2. June 2010

[11] Najman, L., & Talbot, H. (2013). Mathematical Morphology: From Theory to Applications. John Wiley & Sons. Cap. 1, pp. 4

[12] Canny, J., A Computational Approach To Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986

[13] Jain, A. K. (1989). Fundamentals of Digital Image Processing. Prentice Hall. Cap. 9, pp. 348, 355

[14] Liu, R., & Mao, J. (2018). Research on Improved Canny Edge Detection Algorithm. MATEC Web of Conferences, 232, 03053. https://doi.org/10.1051/matecconf/201823203053

[15] Ondarza, J., Symington, S. B., Houten, J. l., & Clark, J. M. (2003). G-Protein Modulators Alter the Swimming Behavior and Calcium Influx of Paramecium tetraurelia. The Journal of Eukaryotic Microbiology, 50(5), 349– 55. https://doi.org/10.1111/j.1550-7408.2003.tb00147.x

[16] OpenCv: Color conversions (n.d) https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html

[17] OpenCV: Miscellaneous Image Transformations. (n.d.). https://docs.opencv.org/4.x/d7/d1b/group__imgproc__misc.html#gaa9e58d2860d4afa658ef70a9b1115576

[18]OpenCV:FeatureDetection.(s. f.). https://docs.opencv.org/3.4/dd/d1a/group__imgproc__feature.html#ga04723e007ed888ddf11d9ba04e2232de

[19] Suzuki, S., & be, K. (1985). Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics, and Image Processing, 30(1), 32–46. doi:10.1016/0734-189x(85)90016-7

[20] Fitzgibbon, A., Pilu, M., & Fisher, R. (1996). Direct least square fitting of ellipses. IEEE Transactions on Pattern Analysis and Machine Intelligence, 21(5), 476–480. https://doi.org/10.1109/34.765658

[21] Fitzgibbon, A., & Fisher, R. N. (1995). A Buyer's Guide to Conic Fitting. British Machine Vision Conference. https://doi.org/10.5244/c.9.51

[22] Strang, G. (1998). Linear Algebra and Its Applications. Harcourt College Pub. Cap. 5, pp. 254;

[23] https://github.com/imejia-code/PARAM_PROCESSCPP

[24] The GNU General Public License v3.0 - GNU Project - Free Software Foundation. (n.d.). https://www.gnu.org/licenses/gpl-3.0.en.html