ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UN PROGRAMA EN MATLAB QUE PERMITA EL CONTEO DE MICROORGANISMOS EN PLACAS PETRIFILM UTILIZANDO VISIÓN COMPUTACIONAL

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

JAIME MAURICIO MONTENEGRO BENAVIDES

DIRECTOR: Ing. JORGE EDUARDO CARVAJAL RODRÍGUEZ M.Sc.

Quito, junio 2021

AVAL

Certifico que el presente trabajo fue desarrollado por Jaime Mauricio Montenegro Benavides, bajo mi supervisión.

M.Sc. Jorge Carvajal
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Jaime Mauricio Montenegro Benavides, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

Jaime Mauricio Montenegro Benavides

DEDICATORIA

A mí Madre Consuelo y mi Padre Rubén por su apoyo incondicional en momentos difíciles y además ser un ejemplo de trabajo duro y tenacidad, igualmente le dedico a mi Tío Mauro, por ser mi referente, mi soporte y ejemplo a seguir.

AGRADECIMIENTO

A mí Madre Consuelo por ser mi soporte económico y moral, a mí Padre Rubén y mi hermano Santiago, que sin su ayuda hubiera sido imposible culminar mi carrera, a mi Tío Mauro por ser como un padre, con su ejemplo y apoyo siempre fue mi referente y una inspiración de vida, a mi Tío Wilson por brindarme sus consejos que siempre son una luz en mi vida.

A mi abuelito Zenón que desde el cielo guía mis pasos, a Carito que es una persona especial en mi vida y un gran apoyo, a mi gran amigo Andrés, un hermano que la vida me brindó, que durante toda la carrera universitaria cultivamos una amistad sincera.

A mi tutor, MSc Jorge Carvajal agradezco su labor incansable de ayuda y compresión durante la carrera. Su don de gente fue vital para no desmayar y conseguir este objetivo de vida.

ÍNDICE DE CONTENIDO

AVAL I
DECLARACIÓN DE AUTORÍA II
DEDICATORIA III
AGRADECIMIENTO IV
ÍNDICE DE CONTENIDO V
RESUMEN VII
ABSTRACT VIII
1. INTRODUCCIÓN 1
1.1 OBJETIVOS 2
1.2 ALCANCE 3
1.3 MARCO TEÓRICO 4
1.3.1 VISION COMPUTACIONAL 4
1.3.2 ESPACIO DE COLOR 5
1.3.2.1 ESPACIO DE COLOR RGB 5
1.3.2.1 ESPACIO DE COLOR CIE*lab 6
1.3.3 PROCESAMIENTO DE IMÁGENES 7
1.3.3.1 IMÁGENES DE NIVEL DE GRISES 8
1.3.3.2 HISTOGRAMA 9
1.3.3.3 ECUALIZACIÓN DE HISTOGRAMA 11
1.3.3.4 ECUALIZACIÓN ADAPTATIVA DE HISTOGRAMA 12
1.3.3.5 IMÁGENES BINARIAS 12
1.3.4 OPERACIONES MORFOLÓGICAS 13
1.3.4.1 DILATACIÓN DE IMÁGENES BINARIAS 14
1.3.4.2 EROSIÓN DE IMÁGENES BINARIAS 15
1.3.4.3 APERTURA DE IMÁGENES BINARIAS 17
1.3.4.4 CIERRE DE IMÁGENES BINARIAS 17
1.3.4.5 EXTRACIÓN DE BORDES 18
1.3.4.6 ALGORITMO DE RELLENO DE HUECOS 19
1.3.5 ALGORITMO DE APRENDIZAJE AUTOMÁTICO NO SUPERVISADO K-MEANS 20
1.3.5.1 APRENDIZAJE AUTOMÁTICO SUPERVISADO 21

1.3.5.2 APRENDIZAJE AUTOMÁTICO NO SUPERVISADO 21
1.3.6 DETECCIÓN DE OBJETOS EN IMÁGENES BINARIAS 24
1.3.6.1 VECINDAD DE PIXELES..... 24
1.3.6.2 COMPONENTES CONEXAS 25
1.3.6.3 TIPOS DE CONECTIVIDAD 25
1.3.6.4 ALGORITMO DE ETIQUETADO 26
1.3.7 TRANSFORMADA DE HOUG 28
1.3.8 DESCRIPTORES GEOMÉTRICOS EN IMÁGENES BINARIAS..... 32
1.3.9 DESCRIPCIÓN DE HERRAMIENTAS UTILIZADAS 33
1.3.10 TRABAJOS ANTERIORES..... 34
2. METODOLOGÍA..... 35
2.1 SEGMENTACIÓN DE LA SECCIÓN DE INTERÉS UTILIZANDO K-MEANS…….....37
2.2 CONTEO DE LOS MICROORGANISMOS EN LA PLACA PETRIFILM..... 53
2.3 DESCRIPTORES GEOMÉTRICOS 72
2.4 INTERFAZ GRÁFICA DISEÑADA EN APP DESIGNER MATLAB..... 77
3. RESULTADOS Y DISCUSIÓN 88
3.1 RESULTADOS EN LA SEGMENTACIÓN CON K-MEANS……87
3.2 RESULTADOS DE ECUALIZACIÓN ADAPTATIVA DE HISTOGRAMA.... 91
3.3 RESULTADOS DEL CONTEO 92
3.4 RESULTADOS DE LOS DESCRIPTORES GEOMÉTRICOS 95
4. CONCLUSIONES Y RECOMENDACIONES..... 99
4.1 CONCLUSIONES..... 99
4.2 RECOMENDACIONES 100
5. REFERENCIAS BIBLIOGRÁFICAS 101

RESUMEN

Las Placas Petrifilm para conteo y análisis de diferentes tipos de microorganismos como Mohos, Levaduras y bacterias son un medio de cultivo listo para usarse con fines investigativos en varios campos de la microbiología, su ventaja es que ofrecen soluciones a través de pruebas fáciles de usar, rápidas y precisas con resultados confiables que constituyen indicadores de calidad de los productos analizados.

Para interpretar la información que se presenta en las Placas, los microbiólogos utilizan su sentido de la vista y un lente de aumento para mejorar la visualización, con lo cual, les permite contar cuantos organismos existen en las Placas Petrifilm, el conteo que realiza el microbiólogo puede presentar errores humanos debido a que depende de los sentidos de cada persona. Para solucionar este problema se desarrolló un sistema de software en Matlab que permite el conteo y análisis automático de microorganismos en placas Petrifilm.

El programa incluye: operaciones morfológicas, la técnica de aprendizaje no supervisado k-means, detectores de bordes y geometrías circulares, así como también identificación de centros, áreas y perímetros, además algoritmos basados en vecindad y conectividad de pixeles que permiten realizar el conteo total de todos los microorganismos presentes en la placa petrifilm.

El desarrollo del presente programa se efectúa sobre imágenes digitales de las placas petrifilm que fueron proporcionadas por el Departamento de Microbiología de la USFQ.

.

PALABRAS CLAVE: Placas Petrifilm, Conteo de Microorganismos, Software Matlab, Operaciones Morfológicas, Algoritmo K-means.

ABSTRACT

Petrifilm Plates for counting and analyzing different types of microorganisms such as molds, yeasts and bacteria are a culture medium ready to use for research purposes in various fields of microbiology, their advantage is that they offer solutions through easy-to-use test. Fast and precise with reliable results that constitute quality indicators of the analyzed products.

To interpret the information presented on the Plates, microbiologist use their sense of sight and a magnifying glass to improve visualization. Allowing them to count how many organisms exist on the Petrifilm Plates the count performed by the microbiologist it can present human errors because it depends on the senses of each person. To solve this problem, a software system was developed in Matlab that allows the automatic counting and analysis of microorganism in Petrifilm plates.

The program includes: morphological operations, the k-means unsupervised learning technique, edge detectors and circular geometries, as well as identification of centers, areas and perimeters, as well as algorithms based on neighborhood and pixel connectivity that allow counting total of all microorganism present on the petrifilm plate.

The development of this program is carried out on digital images of the petrifilm plates that were provided by the Department of Microbiology of the USFQ.

KEY WORDS: Petrifilm plates, Microorganism Counting, Matlab Software, Morphological Operations, K-means Algorithm.

1. INTRODUCCIÓN

Las placas petrifilm en el área de la microbiología es una herramienta que proporciona pruebas y análisis rápidos, consistentes y precisos para cumplir con los estrictos requerimientos de seguridad e higiene de diversos productos [1].

Las industrias alimentarias analizan sus productos alimenticios para garantizar su inocuidad par el ser humano, permitiendo así, identificar y cuantificar bacterias e indicadores de contaminación de forma rápida, por ejemplo; las placas petrifilm son utilizadas en la industria lechera, donde se realiza un análisis microbiológico y la resistencia a antibióticos de la leche cruda realizando un conteo y análisis de Coliformes (Aerobios), con el fin de garantizar si la cantidad de bacterias sobrepasa los límites máximos permitidos por las normas de salud existentes [2].

Para el conteo y análisis microbiológico se debe preparar las muestras, y esto consiste en inocular e incubar, para su posterior lectura, como se indica en la Figura 1.1

Figura 1.1 Proceso para la preparación de muestras y lectura de análisis.

Las Placas Petrifilm contienen un agente gelificante soluble en agua fría, nutrientes y un tinte que permite la visualización de la colonia de bacterias que se generan durante el proceso de incubación en las placas [3].

Para interpretar la información que se encuentra en las placas, los microbiólogos utilizan su sentido de la vista y un lente de aumento para mejorar la visualización, con lo cual, les permite contar cuantos organismos existen en las placas petrifilm. Cuando el número de bacterias es mayor a 250 en un cm^2 por su excesivo crecimiento, por facilidad, los conteos son estimados, se realiza el conteo solo en un cm^2 y este resultado se multiplica por 20,

así se obtiene el resultado total de la placa. Este método se aplica debido a que cada placa petrifilm está dividida en 20 secciones de un cm^2.

Los microbiólogos realizan otro procedimiento donde cuentan y clasifican visualmente en base al tamaño los microorganismos que han alcanzado un mayor crecimiento durante el proceso de incubación, además en base a la redondez de los Aerobios descartan las partículas que no son de interés o residuos del producto, ya que estos tienen una forma irregular.

Los microbiólogos emplean sus habilidades sensoriales, en especial su visión para realizar el conteo y análisis de los organismos incubados en las placas petrifilm, sin embargo, esta forma tradicional de conteo y análisis no resulta ser la más adecuada ya que los microbiólogos utilizan más tiempo y cantidad de mano de obra en la ejecución de estos procesos [4].

De ahí que, la necesidad de los microbiólogos en mejorar sus tiempos de análisis, permitiendo una optimización de recursos en el conteo de microorganismos en placas petrifilm, en el presente trabajo de titulación con colaboración de Departamento de Microbiología de USFQ, quien proporciona las imágenes digitales de las placas petrifilm en formato jpeg, se desarrolló un sistema de conteo y análisis automático utilizando técnicas de procesamiento de imágenes, reduciendo la posibilidad de error humano [4].

1.1 OBJETIVOS

El objetivo general del Proyecto Técnico es: Desarrollo de un programa en Matlab que permita el conteo de microorganismos en placas petrifim utilizando visión computacional.

Los objetivos específicos del Trabajo Técnico son:

- Describir la información relevante acerca de las placas petrifilm con fines investigativos.
- Detallar el procesamiento de imágenes, operaciones morfológicas, transformada de houg y algoritmo K-means.
- Implementar un programa en Matlab para el conteo de microorganismos.
- Analizar los resultados arrojados por la aplicación.

1.2 ALCANCE

En el siguiente proyecto de titulación se plantea implementar una aplicación en Matlab que permita el conteo y análisis de microorganismos(Aerobios) en placas petrifilm utilizando visión computacional, lo cual permite tener una herramienta automatizada para la detección de microorganismos en las placas petrifilm basada en el procesamiento de imágenes digitales.

El presente programa incluye: operaciones morfológicas, la técnica de aprendizaje no supervisado K-means, detectores de bordes y geometrías circulares, así como también identificación de sus centros, áreas y perímetros de los microorganismos de mayor tamaño que son los de mayor interés a analizar, además algoritmos basados en vecindad y conectividad de pixeles que permiten realizar el conteo total de los microorganismos presentes en la placa petrifilm. El desarrollo del presente programa se efectuará sobre imágenes digitales de las placas petrifilm que fueron proporcionadas por el departamento de Microbiología de la USFQ.

En la Figura 1.2 se muestra el diagrama de bloques del programa realizado.

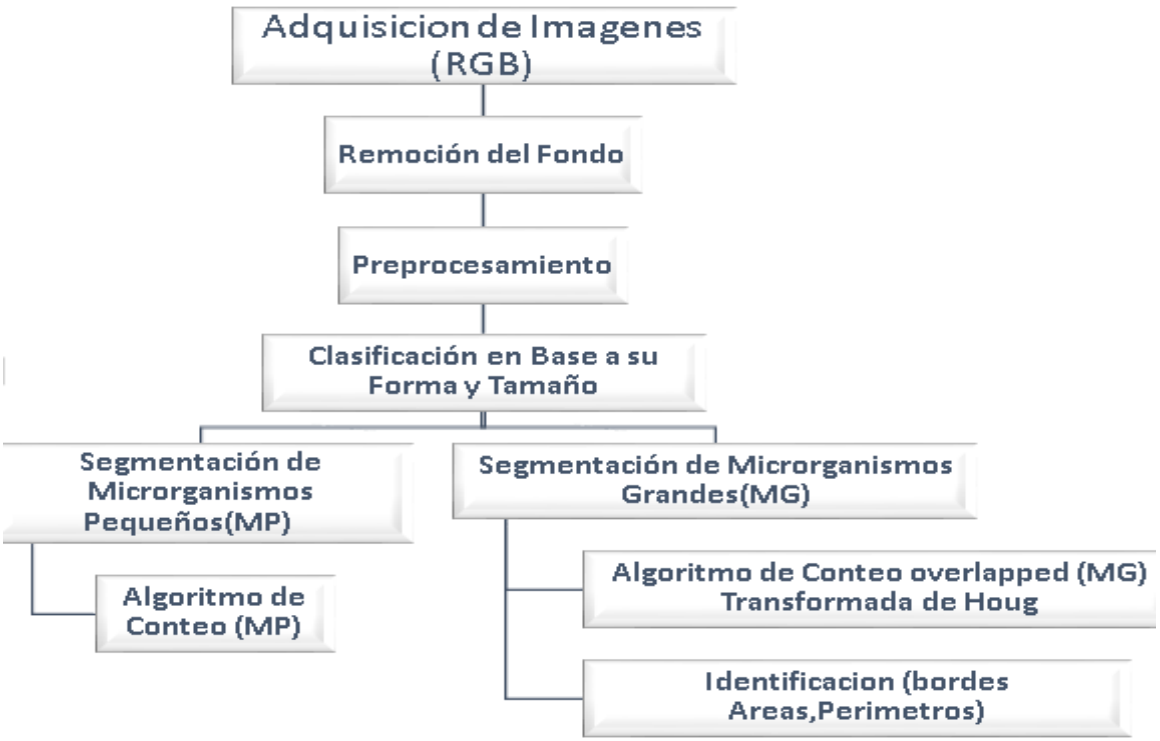


Figura 1.2 Diagrama de bloques de la secuencia del programa implementado en Matlab.

El programa comienza con la selección de la imagen que se representa en un formato RGB (Red, Green, Blue) que es transformado al espacio de color L*a*b*, donde posteriormente se utiliza operaciones morfológicas y la técnica de clustering k-means, con el objetivo que se facilite el análisis del objeto, centrándose solo en la parte de interés [5].

K-means realiza una clasificación no supervisada que tiene como fin organizar los datos en grupos (clusters), de tal manera que las características de los miembros de cada grupo sean similares entre si y diferentes a las de otro grupo.

Luego que se tiene la región de interés se realiza un pre-procesamiento de la imagen, y seguido una segmentación de los microorganismos en base a su morfología(tamaño) y así aplicar el algoritmo de conteo.La transformada de Houg facilita contar los microorganismos que están semi-solapados, también aplicaremos algoritmos para medir perímetros y circularidad de los microorganismos para finalmente los resultados indicarlos en una interfaz gráfica amigable con el usuario diseñada en App Designer Matlab [6].

1.3 MARCO TEÓRICO

En un inicio la microbiología ha sido por tradición una disciplina manual, pero con objetivo de optimizar la productividad y eficiencia de los laboratorios de microbiología el aporte de las nuevas tecnologías entre ellas la visión computacional han permitido la automatización y mejora de diagnósticos microbiológicos.

En consecuencia, en la presente sección se detallará las técnicas de visión computacional que ayudarán al desarrollo de un software que permita el conteo de microrganismos en placas petrifilm.

1.3.1 VISION COMPUTACIONAL

Nuestra capacidad de interpretar el entorno que nos rodea depende de la vista y el cerebro, que a partir de los rayos de luz que llegan a nuestros ojos identifica los objetos que nos interesan y su posición en el espacio. La Visión Computacional intenta simular o replicar la habilidad y capacidad humana del sentido de la vista en las computadoras, a partir de imágenes adquiridas, por ejemplo, imágenes de una cámara fotográfica [7].

La facultad de analizar las imágenes por un computador, para luego extraer la información útil con lleva un trabajo previo y secuenciado que incluye modelos de cálculo matemático que están inmersos en el procesamiento de imágenes, modelos que permiten analizar y mejorar propiedades como el tamaño, forma, gradiente, texturas, color, luminancia, contornos de regiones de interés, con el propósito de segmentar la imagen para su final interpretación por el computador. Cabe mencionar que el procesamiento de imágenes y la visión computacional, aunque están relacionados, existe una diferencia fundamental, el procesamiento de imágenes perfecciona la imagen para la interpretación por una persona, mientras que en la visión computacional la interpretación de las imágenes la realiza una computadora [8].

Existen diversos ejemplos de aplicación de la visión computacional que van desde la medicina, astronomía, manufactura. En la medicina una aplicación de la visión computacional proporciona diagnósticos más exactos y automáticos de tumores malignos, esto se realiza mediante el procesamiento de imágenes de radiografía, microscopía o de resonancia magnética, sin la necesidad de interpretación médica [9].

1.3.2 ESPACIO DE COLOR

El color es una sensación que se produce en respuesta al estímulo nervioso del ojo causado por una longitud de onda luminosa.

Un espacio de color es la descripción y organización de toda la gama de colores que percibe el sentido de la vista humano, expresado en un modelo matemático que permite la interpretación del color. Existen varios espacios de color, por ejemplo, RGB, HSV, L*a*b etcétera, donde cada uno posee ventajas y desventajas que dependen de su aplicación [10].

1.3.2.1 Espacio de Color RGB

La representación RGB es una de las formas más comunes de simbolizar una imagen a color. Para dar origen a toda la gama de colores visibles al ojo humano se realiza la combinación aditiva de los tres colores principales, rojo, verde y azul.

Las imágenes en color se pueden constituir utilizando tres matrices de similar tamaño en 2D, una matriz para cada canal de color: rojo (R), verde (G), azul (B), cada elemento de la matriz contiene un valor que equivale a 8 bits que a su vez origina una escala de [0, 255], la combinación de las tres matrices da un total de 24 bits, que da como resultado 16 millones de combinaciones de colores [11].

En la Figura 1.3 se indica una pequeña región de 4x4 pixeles de una imagen a color con los valores de cada componente: rojo, verde y azul.

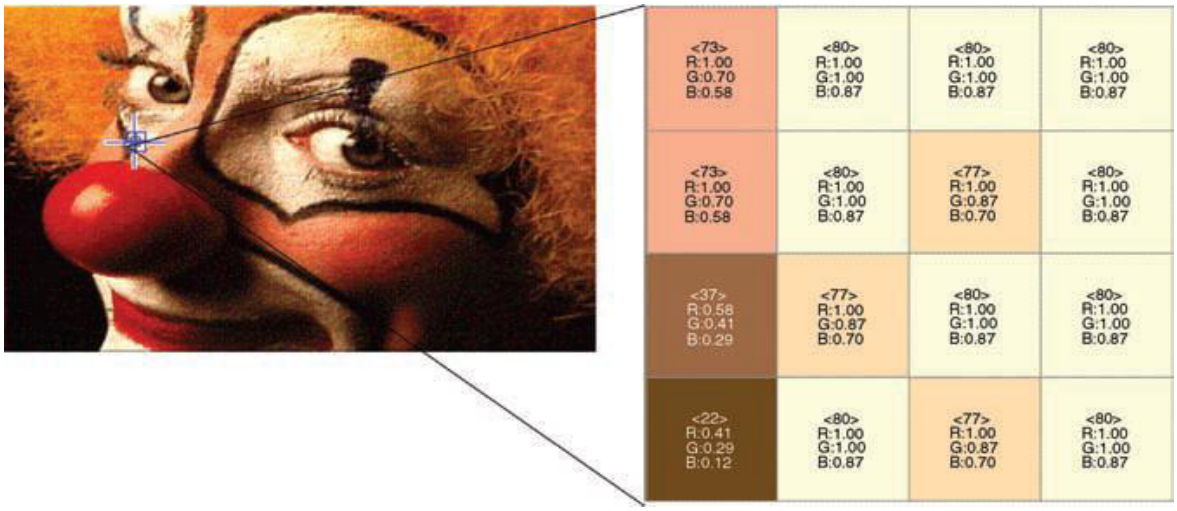


Figura 1.3 Imagen a color en formato de espacio RGB.

1.3.2.2 Espacio de Color CIE*L*a*b

La percepción e interpretación humana del color suele ser subjetiva, razón por la cual es importante expresar el color en términos objetivos. La ventaja del espacio de color L*a*b es que relaciona valores numéricos de color con la percepción visual humana, lo que permite tener una medida más objetiva del color y simular la respuesta no lineal del ojo.

El espacio de color L*a*b nació en base a la teoría de color oponente, es decir, existen colores que son totalmente opuestos en su tonalidad, por ejemplo, negro-blanco, rojo-verde y amarillo-azul, a partir de la teoría de color oponente el espacio de color L*a*b es representado en un sistema cartesiano de tres ejes, uno vertical (*L), y dos horizontales (*a y *b) [11].

El eje vertical *L representa la luminosidad en una escala de 0 (negro) hasta 100(blanco), el eje *a representa todas las tonalidades que van desde rojo(positivo) hasta verde(negativo) con una escala de -100 a +100 y el eje *b representa tonalidades desde amarillo(positivo) a azul(negativo) con una escala de -100 a +100.

El sistema cartesiano del espacio de color L*a*b se muestra en la Figura 1.4.

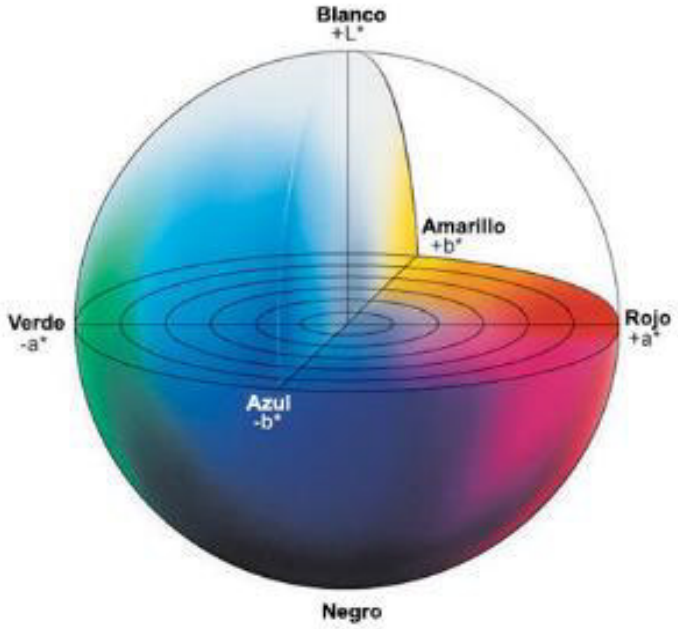


Figura 1.4 Espacio de color L*a*b [12].

El espacio de color L*a*b especifica el color en tres factores:

1. **Tonalidad**

   Atributo por el que se dice que un estímulo es verde, rojo, azul.

2. **Luminosidad**

   Atributo por el que se dice si el color es claro u oscuro.

3. **Saturación**

   Atributo por el que se dice si un color es más suave o más fuerte.

1.3.3 PROCESAMIENTO DE IMÁGENES

El procesamiento de imágenes digitales es la manipulación de la información contenida en la imagen por medio de algoritmos matemáticos utilizando un computador, con el objetivo de obtener la información útil de la imagen [13].

El procesamiento de imágenes tiene tres etapas:

Pre-procesamiento

Tiene como objetivo manipular la imagen para que el resultado sea más apropiado que el de la imagen original, por ejemplo, mejorar el contraste, mejorar contornos para que sean más lisos, resaltar detalles importantes, filtrar ruido.

Segmentación

El objetivo de esta etapa es dividir o descomponer la imagen en regiones de interés siguiendo un criterio y un fin específico.

Análisis

La etapa de análisis está ligada a la etapa de segmentación, es decir, sí se obtiene una buena segmentación se puede hacer un buen análisis de la imagen.

Para una adecuada manipulación e interpretación del contenido que se encuentra presente en una imagen hay ocasiones que es necesario trasformar la imagen mediante la aplicación de algoritmos matemáticos que permiten transformar del formato original a otro formato fácil de manipular, como, por ejemplo, transformar imágenes en formato a color a formato en escala de grises o binario [14].

1.3.3.1 Imágenes de Nivel Grises

Las imágenes en el formato de grises se representan en una matriz de dos dimensiones, de forma general cada elemento de la matriz se cuantifica con un valor de 8 bits por pixel, el valor de pixel 0 corresponde a “negro” y un valor de pixel de 255 corresponde “blanco”, mientras todos los valores intermedios entre 0 y 255 expresan las variaciones de sombras de gris. La principal característica de una imagen de grises es que se ha eliminado la componente del tono y la de saturación, conservando solo la luminancia de la imagen.

En la Figura 1.5 se indica una pequeña región de 6x6 pixeles de una imagen en escala de grises, donde los pixeles de valores grandes corresponden a los pixeles más brillantes [15].

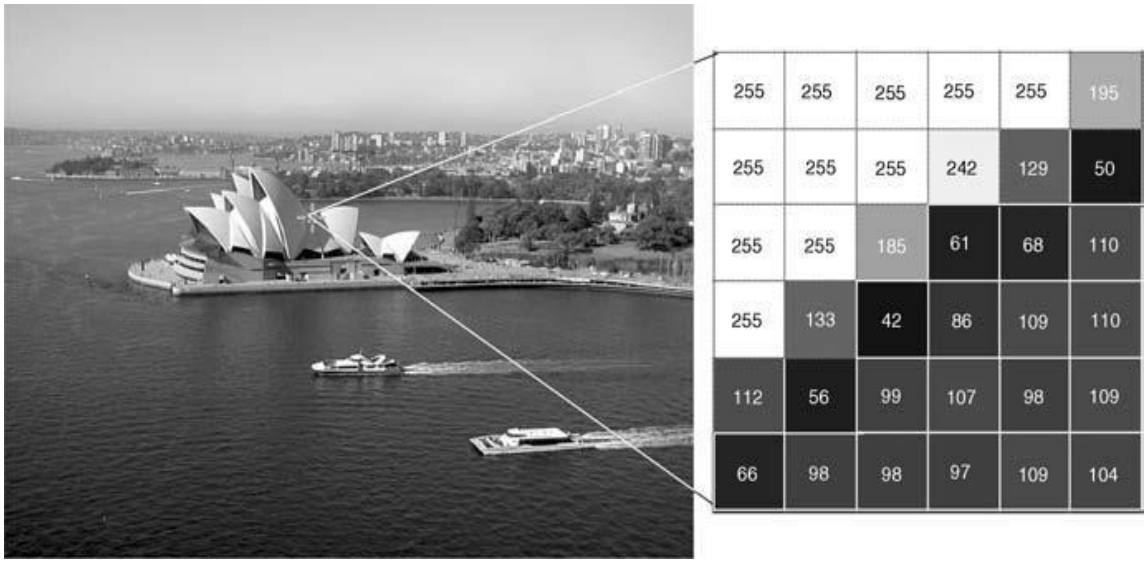


Figura 1.5 Imagen en escala de grises [16].

La ecuación que permite transformar una imagen en formato RGB a una imagen en niveles de grises está representado por la Ecuación 1.1

$$Luminancia = 0.299.R + 0.587.G + 0.114.B \quad (1.1)$$

Donde R, G y B son las componentes rojas, verde y azul de la imagen a color y los pesos para cada componente representan la sensibilidad del sistema visual humano a las frecuencias de espectro cercano a los colores primarios [17].

1.3.3.2 Histograma de una Imagen en Escala de Grises

El histograma de una imagen monocromática es una gráfica que representa la repetición de ocurrencia de cada nivel de gris en la imagen.

Los histogramas se simbolizan mediante un gráfico de barras, cada barra es un nivel de gris, la altura de la barra es el número de pixeles presentes en la imagen que corresponden a ese nivel de gris en particular [18].

Matemáticamente el histograma se representa con la Ecuación 1.2.

$$h(k) = n_k \quad (1.2)$$

Donde:

$k = 0,1, \ldots., L - 1$

L es el número de niveles de gris.

n_k es el número de pixeles que corresponden a un nivel de gris en particular.

Otra representación de un histograma es de la forma normalizada y se define matemáticamente como la ecuación 1.3.

$$p(r_k) = \frac{n_k}{n} \tag{1.3}$$

Donde:

n es el número total de pixeles en la imagen.

$p(r_k)$ es la probabilidad del k-ésimo nivel de gris.

En la Figura 1.6 se observa la representación de un histograma de siete niveles [19].

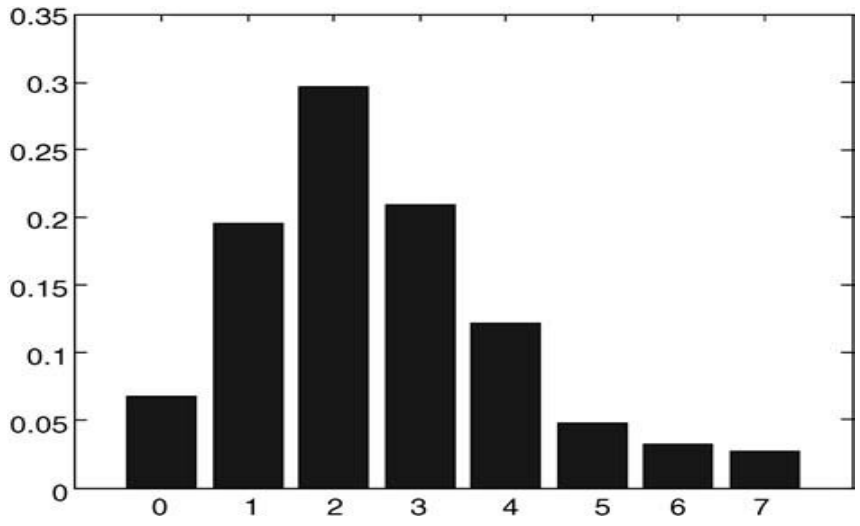


Figura 1.6 Representación de un histograma de la forma normalizada con 7 niveles de grises.

Los histogramas brindan la posibilidad de analizar las propiedades de una imagen como por ejemplo el brillo y el contraste, una imagen que tiende a ser oscura da como resultado un histograma que manifiesta una acumulación de barras en el extremo inferior del rango

del nivel de grises, mientras que en una imagen de alto brillo las barras se acumulan en el extremo superior del rango de nivel de grises, si el histograma de una imagen se muestra distribuido de una manera uniforme en todos sus rangos de grises, se dice que la imagen tiene un alto contraste [20].

1.3.3.3 Ecualización de Histograma

La técnica de ecualización de histograma tiene como objetivo cambiar la imagen de tal forma que el nuevo histograma resultante se muestre plano y uniforme, como consecuencia se obtiene una mejora en el contraste de la imagen.

En la ecualización de histograma se utiliza una función auxiliar de transformación llamada función de distribución acumulativa, dada por la Ecuación 1.4.

$$s_k = T(r_k) = \sum_{j=0}^{k} \frac{n_j}{n} = \sum_{j=0}^{k} p(r_j) \tag{1.4}$$

Donde:

s_k es el nuevo nivel de gris mapeado para todos los pixeles.

Gracias a la ecualización de histograma en la Figura 1.7 se observa una mejora en el contraste imagen literal c) con respecto a la imagen original literal a), además como resultado se obtiene una expansión del histograma literal d) [21].

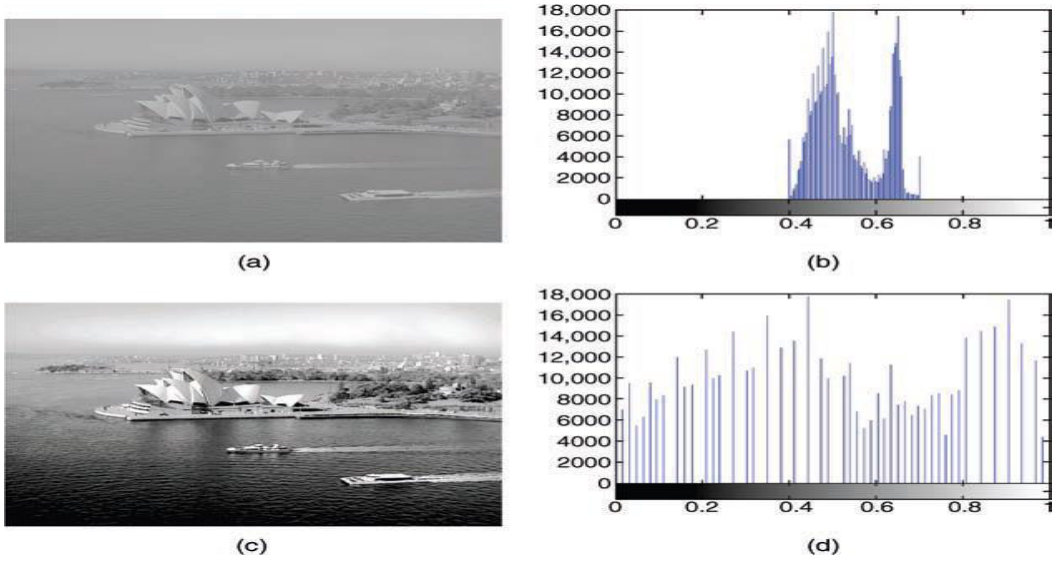


Figura 1.7 Comparativa entre imágenes e histogramas con ecualización literal c) y d) y sin ecualización literal a) y b).

1.3.3.4 Ecualización Adaptativa de Histograma

Un problema que se presenta con la ecualización tradicional de imágenes es la posible deficiencia en mejorar los detalles en espacios de área diminutos, ya que el algoritmo de ecualización de histograma analiza la imagen en forma global, entonces se hace indispensable utilizar la versión sectorizada del algoritmo.

La ecualización adaptativa se fundamenta en utilizar un pequeño cuadrado deslizante que recorre toda la imagen y al mismo tiempo calcula la ecualización de histograma solo en la pequeña sección o vecindario delimitada al tamaño de la ventana deslizante, lo que da como resultado un análisis minucioso de toda la imagen, consiguiendo mejorar el contraste y los pequeños detalles de la imagen.

La ecualización adaptativa es de suma utilidad cuando se requiere visibilizar y restaurar los detalles diminutos de una imagen en escala de grises, la ecuación 1.4 también se utiliza en el método de ecualización adaptativa a diferencia que la ecuación se aplica a pequeños vecindarios de la imagen [22].

1.3.3.5 Imágenes binarias

En aplicaciones para el estudio de imágenes es necesario reducir la imagen de escala de grises a una imagen con dos niveles de gris (blanco y negro), a dicho proceso se le denomina umbralización o binarización de imágenes.

En la Figura 1.8 se observa que normalmente a la imagen binaria se la representa con un valor de un bit por pixel, donde el valor de 0 por lo general significa “negro” y el valor de 1 “blanco”.

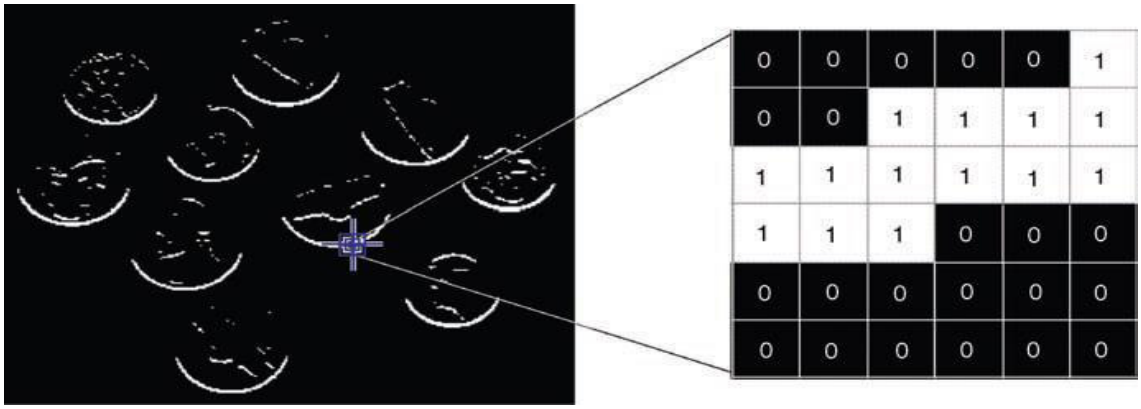


Figura 1.8 Representación de una imagen binaria, el valor de 1 es blanco y el valor de 0 es negro.

La transformación de una imagen de escala de grises a una imagen binaria habitualmente se ejecuta comparando la intensidad de cada pixel con un valor de intensidad que se toma como referencia (umbral) y se sustituye el valor de intensidad por un valor 1 ó 0 según el resultado de dicha comparación [23].

Determinar el umbral de una imagen es un paso del pre-procesamiento de imagenes, donde los objetos de interés de análisis tienen un brillo promedio más intenso que sus vecinos, matemáticamente el paso para crear el umbral de una imagen de entrada f(x,y) y originar una interpretación binarizada g(x,y) se determina como:

$$g(x,y) = \begin{cases} 1, si\ f(x,y) > T \\ 0\ , f(x,y)\ \leq T \end{cases} \quad (1.5)$$

Donde T es el umbral y $f(x,y)$ es un valor de intensidad.

1.3.4 OPERACIONES MORFOLOGICAS

Una estrategia que da buenos resultados y se utiliza en el procesamiento de imágenes es la morfología matemática aplicada a imágenes binarias, no obstante, la morfología matemática también se puede aplicar a imágenes a color e imágenes a escala de grises. La morfología matemática abarca conceptos de algebra, geometría, teoría de conjuntos, que permite procesar imágenes con el propósito de resaltar, filtrar, segmentar, rellenar regiones, detectar objetos y bordes [24].

Las dos operaciones básicas en las que se define y consolida toda la morfología matemática, son las operaciones morfológicas de erosión y dilatación, de las dos operaciones mencionadas se derivan un sin número de operaciones morfológicas con diversas aplicaciones.

La clave de la morfología matemática está en el elemento estructurante (una pequeña imagen binaria) [25]. El elemento estructurante es definido como un conjunto de pixeles de geometría conocida, simétrica o asimétrica, que se crea y escoge a priori dependiendo de la aplicación, una vez determinado el elemento estructurante se lo desplaza pixel a pixel por todo el plano de la imagen binaria ocasionando transformaciones a la imagen original.

Los elementos estructurantes pueden tener diferentes formas y tamaños, lo que origina diferentes resultados al momento de aplicar a la imagen original, de forma adicional en cada elemento estructurante se debe definir preliminarmente sus centros u orígenes como se indica en la Figura 1.9.

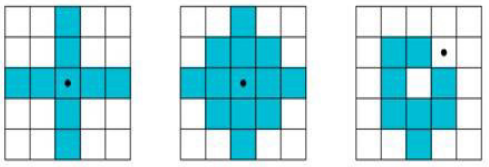


Figura 1.9 Ejemplos de elementos estructurantes.

1.3.4.1 Dilatación de Imágenes Binarias

Se define para una imagen B con un elemento estructurante A, tal que, cuándo se traslada el elemento estructurante a un punto(pixel) de la imagen original, el elemento estructurante intersección con la imagen binaria da como resultado un conjunto diferente de vacío, si esto se cumple ,el resultado es el siguiente, el pixel de la imagen binaria que esta exactamente solapado o superpuesto con el pixel central del elemento estructurante toma el valor de 1, caso contrario, si la intersección entre el elemento estructurante y la imagen es nula(no existe intersección) o un conjunto vacío, el pixel de la imagen binaria que esta superpuesto con el pixel central del elemento estructurante toma el valor de 0 [26].

En conclusión, basta solo un pixel de valor 1 que se intersecte entre el elemento estructurante y la imagen binaria para que el resultado de la dilatación tome un valor de 1.

La dilatación se expresada por la siguiente operación de conjuntos:

$$B \oplus A = \{Z|(\hat{A}) \cap B \neq \emptyset\} \qquad (1.6)$$

Donde:

A es un elemento estructurante.

$\hat{A}$ es la reflexión del elemento estructurante

Z representa todos los puntos espaciales o coordenadas (x, y) de la imagen binaria.

B es la imagen binaria.

$\emptyset$ es un conjunto vacío.

En la Figura 1.10 se observa que la operación de dilatación añade pixeles (pixeles amarillos en imagen de la derecha) a la imagen original (imagen de la izquierda), también la dilatación es útil para cerrar algunos agujeros y unir regiones próximas [27].

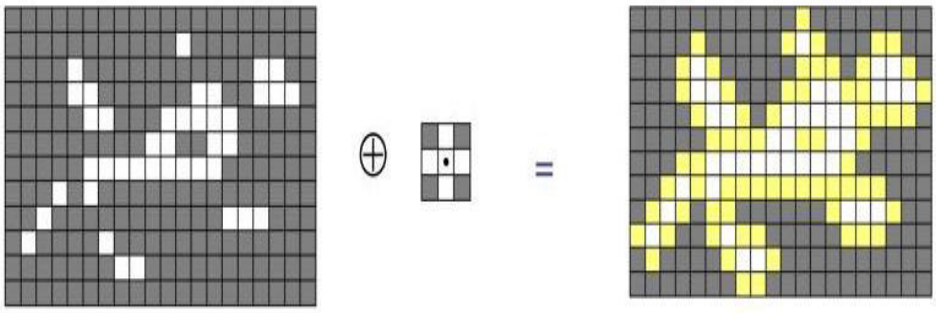


Figura 1.10 Representación de la dilatación a la imagen de la izquierda dando como resultado la imagen de la derecha [28].

1.3.4.2 Erosión de Imágenes Binarias

Se define para una imagen B con un elemento estructurante A, tal que, cuando yo traslado el elemento estructurante a un punto de la imagen, este quede totalmente contenido en un elemento de la imagen, es decir, el elemento estructurante es un subconjunto de la imagen, si esto se cumple, el resultado es el siguiente, el pixel de la imagen binaria que está superpuesto con el pixel central del elemento estructurante recibe el valor de 1, caso contrario, si el elemento estructurante no está totalmente contenido en la imagen, el pixel de la imagen binaria que esta solapado con el pixel central del elemento estructurante recibe el valor de 0 [29].

La erosión se expresa por la siguiente operación de conjuntos:

$$B \ominus A = \{Z|(A) \subseteq B\} \tag{1.7}$$

$$B \ominus A = \{Z|(A) \cap B^C = \emptyset\} \tag{1.8}$$

Donde:

A es un elemento estructurante.

Z representa todos los puntos espaciales o coordenadas (x, y) de la imagen binaria.

B es la imagen binaria.

B^C es el conjunto complemento de B.

$\emptyset$ es un conjunto vacío.

En la Figura 1.11 se observa que en la imagen erosionada de la derecha se eliminan pixeles, reduce bordes y separa objetos

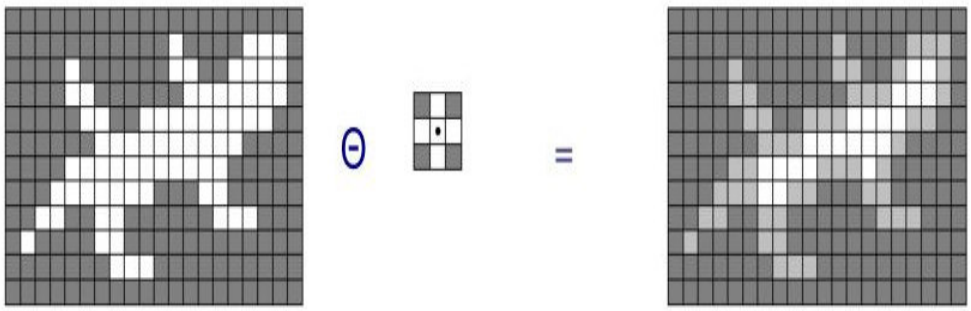


Figura 1.11 Representación de la operación de erosión a la imagen de la izquierda [30].

Cabe mencionar que la dilatación y la erosión no son operaciones inversas, es decir, si dilato la imagen y luego erosiono, o viceversa, la imagen resultado nunca vuelve a ser la imagen original.

Las operaciones morfológicas básicas como la dilatación y la erosión se pueden combinar de tal manera que su resultado origina nuevas operaciones morfológicas como es el caso de la operación morfológica de apertura y cierre que a su vez proporcionan nuevas herramientas de aplicación para la resolución de problemas en el procesamiento de imágenes binarias [31].

1.3.4.3 Apertura de Imágenes Binarias

La operación morfológica de apertura es la composición de una erosión seguida de una dilatación utilizando durante todo el proceso el mismo elemento estructurante, matemáticamente se describe con la siguiente representación:

$$B \cdot A = ((B \ominus A) \oplus A) \tag{1.9}$$

Donde:

A es un elemento estructurante.

B es la imagen binaria.

De manera alternativa la operación de apertura puede ser expresada con la notación de conjuntos de la Ecuación 1.10.

$$B \cdot A = U\{B_z \,/\, B_z \subseteq A\} \tag{1.10}$$

La apertura morfológica se usa ordinariamente para descartar algunas protuberancias finas en los bordes de los objetos que se encuentran en la imagen binaria, además corta diminutos puentes delgados de uniones entre objetos conectados entre sí, lo que permite suavizar los contornos [32].

1.3.4.4 Operación Morfológica de Cierre

La operación morfológica de cierre es la composición de una dilatación seguida de una erosión utilizando el mismo elemento estructurante, matemáticamente se describe con la siguiente representación

$$B \circ A = (B \oplus A) \ominus A \tag{1.11}$$

Donde

A es el elemento estructurante

B es la imagen binaria

La operación de cierre morfológico se maneja para llenar diminutos agujeros, unir roturas y cerrar espacios estrechos existentes entre los objetos de una imagen binaria sin cambiar los tamaños originales de los objetos [33].

1.3.4.5 Extracción de Bordes

Las operaciones morfológicas de dilatación y erosión, combinadas con conceptos relacionados con la diferencia de conjuntos consigue como resultado la extracción de bordes internos o limites externos de los objetos presentes es la imagen binaria.

Limite Interno: Consiste en extraer los pixeles internos de un objeto A que se encuentran localizados en el borde de A.

La ecuación se traduce en que a una imagen A se le resta la imagen ya erosionada de A y como resultado se obtiene el borde interno de la imagen A, el procedimiento se denota por siguiente ecuación [34].

$$B(Ai) = A - (A \ominus B) \tag{1.12}$$

Limite Externo: Consiste en extraer los pixeles externos de un objeto A que se encuentran inmediatamente localizados después de A.

La ecuación se traduce en que se dilata una imagen A y al resultado de la dilatación se le resta la imagen A, y como consecuencia se obtiene el borde externo de la imagen A, el procedimiento se denota por siguiente ecuación [24].

$$B(Ae) = (A \oplus B) - A \tag{1.13}$$

Gradiente Morfológico: Consiste en la combinación mediante una resta de las fronteras externas y las fronteras internas, el procedimiento se denota por la siguiente ecuación.

$$B(Ag) = (A \oplus B) - (A \ominus B) \tag{1.14}$$

1.3.4.6 Algoritmo de Relleno de Regiones en Imágenes Binarias

El algoritmo de llenado de regiones en imágenes binarias se basa en el uso de operaciones morfológicas y operaciones de conjuntos, su finalidad es de llenar los “huecos” que pueden existir en las regiones binarias.

Se define como “hueco” solo a los subconjuntos de pixeles con valor 0 que se encuentran presentes dentro de las regiones de los objetos, es decir, los huecos pertenecen a todo el conjunto de pixeles que conforman un objeto, la condición para que un subconjunto de pixeles de valor 0 reciba la categoría de hueco es que se encuentren rodeados o bordeados en su totalidad por pixeles de valor 1 [36].

El objetivo del algoritmo es eliminar todas las posibles discontinuidades señaladas como huecos que se presentan en un objeto, es decir, transformar los subconjuntos de pixeles con valor 0 “huecos” en subconjuntos de pixeles de valor 1, y así uniformizar en su totalidad al objeto solo con pixeles de valor 1 [37].

Las operaciones inmersas en el algoritmo de llenado de huecos son la operación morfológica de dilatación, además de los conceptos de operaciones de intersección y complemento de conjuntos, a continuación, se detalla paso a paso el funcionamiento del algoritmo.

Sea p un pixel perteneciente a un hueco bordeado en su totalidad por un límite de conectividad-8 como se observa en la Figura 1.12 literal a), la finalidad del algoritmo es llenar toda la región con 1s utilizando p como pixel de inicio.

El relleno del hueco se consigue gracias a un proceso iterativo denotado matemáticamente por la siguiente expresión:

$$X_K = (X_{K-1} \oplus B) \cap A^C \tag{1.14}$$

Donde:

B es el elemento estructurante en forma de diamante en la Figura 1.12 literal c)

A^C es el complemento de la imagen A representado en la Figura 1.12 literal b)

X_K Es el resultado parcial de acuerdo con la iteración del algoritmo descrito por la Ecuación 1.14.

Se empieza por escoger un pixel p perteneciente al hueco Figura 1.12 literal a), donde p siempre se hace que tome el valor de 1,este procedimiento equivale a la iteración 0, donde $X_0 = p$ Figura 1.12 literal d) y B es el elemento estructurante en forma de diamante literal c), a contunuación el pixel p se dilata con el elemento estructurante B y el resultado se interseca con el complemento A^C lo que da como resultado X_1 Figura 1.12 literal e) y así las iteraciónes continuan hasta que el algoritmo se detiene en la k-ésima iteración, es decir, cuando $X_K = (X_{K-1})$ representado en la Figura 1.12 literal h).

Al final de las cinco iteraciones que se realiza en este ejemplo particular el siguiente paso es la unión de X_K y A, que da como resultado concluyente que todos los pixeles del hueco obtienen el valor de 1 Figura 1.12 literal i).

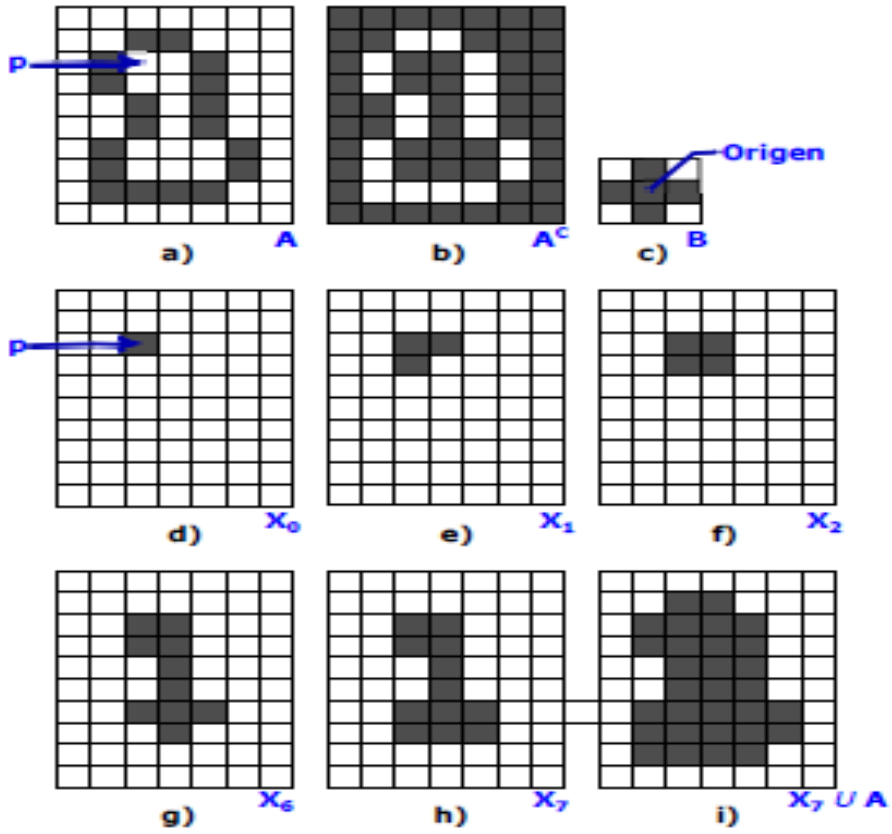


Figura 1.12 Representación del algoritmo de llenado de huecos [38].

1.3.5 ALGORITMO DE APRENDIZAJE AUTOMÁTICO NO SUPERVISADO K-MEANS

El aprendizaje automático o machine learning se basa en algoritmos matemáticos que pueden aprender de cualquier tipo de datos para después como objetivo indicar conclusiones interesantes en base a la data analizada.

El aprendizaje automático cuenta con varias clasificaciones, entre las dos principales están:

1.3.5.1 Aprendizaje Automático Supervisado

Se produce cuando un algoritmo aprende a partir de datos como ejemplos y respuestas asociadas a esos ejemplos, es decir, los algoritmos aprenden con datos donde se conoce el resultado (datos etiquetados), es decir, datos con categorías previamente definidas con el objetivo de predecir respuestas cuando se ingresa otra data semejante, el aprendizaje

supervisado es similar al humano bajo la supervisión de un maestro, el maestro proporciona buenos ejemplos para que el alumno los memorice [39].

1.3.5.2 Aprendizaje Automático no Supervisado

Se produce cuando un algoritmo aprende de datos (no etiquetados) sin ninguna respuesta asociada, es decir, datos sin categorías previamente definidas, lo que permite que el algoritmo estudie por sí mismo los datos, con el objetivo de agrupar y clasificar los datos en base características y patrones similares, son bastante útiles porque proporcionan nuevas aportaciones sobre el significado de los datos.

El análisis de datos mediante algoritmos de aprendizaje no supervisado es servicial en diversas aplicaciones que requieren clasificación de datos, cómo, por ejemplo, identificar células cancerosas dentro de una muestra grande o agrupar palabras con definiciones similares [40].

Uno de los algoritmos matemáticos que se utiliza como método de agrupamiento no supervisado es K-means, este algoritmo consiste en agrupar o segmentar datos en subconjuntos o clústeres, de modo que los datos que están dentro de cada subconjunto están íntimamente relacionados unos a otros con respecto a los datos que están en otros subconjuntos, es decir, un dato solo puede pertenecer a una clase, no a dos o más.

El algoritmo k-means agrupa los datos en k (k= 1,2,3…)clases basándose en criterios de similitud o diferencia que es cuantificada mediante el concepto de distancia euclidiana entre dos puntos (a menor distancia mayor similitud entre dos datos), además, K-means incluye el concepto de centroide que tiene como objetivo minimizar la varianza intracluster y maximizar la varianza intercluster, el centroide es representado por k puntos que vienen a constituir los centros de cada uno de los clústeres [41].

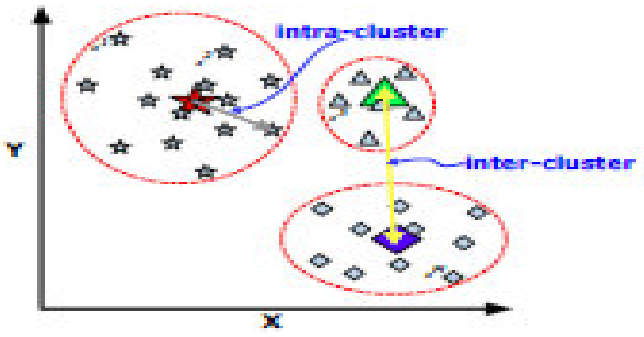


Figura 1.13 Representación de las distancias intercluster e intraculster [42].

En la Figura 1.13 se observa las distancias euclidianas dentro de los clústeres y las distancias entre los clústeres.

El algoritmo K-means está dado por los siguientes pasos:

1. Se debe especificar el número de clústeres k o subconjuntos en que se desea agrupar los datos, este número se define al principio de cada análisis, no existe un método óptimo o teórico global que nos indica cuantos clústeres son los más apropiados, un enfoque práctico es comparar los resultados de múltiples ejecuciones con diferentes valores k y elegir el mejor resultado basado en un criterio definido [43].

2. Se selecciona k puntos que sirven en un futuro como centroides de cada grupo, la ubicación de estos puntos es aleatoria en un inicio, los puntos seleccionados no necesariamente son datos que pertenecen a nuestra data, pueden ser puntos nuevos, los k puntos también llamados semillas son plantados al azar en nuestros datos y guían al momento de dividir en subconjuntos o clústeres al total de los datos [43].

3. El siguiente paso es la fase de asignación de cada dato a que clúster pertenece, esto se realiza verificando la distancia euclidiana de cada uno de los datos a cada uno de los k puntos o semillas, asignando cada dato al clúster k o semilla que se encuentra más cercano [43]. La Ecuación 1.15 calcula la distancia euclidiana entre cada semilla y cada dato.

$$Distancia\ i,j = abs(P(j) - Centroide(i))$$
$$i = 1,2,3 \dots, k\ ;\ j = 1,2,3, \dots, N \qquad (1.15)$$

   Donde k es el número de centroides y clústeres
   N es el número total de datos
   $P(j)$ es el j-ésimo dato.
   $Centroide(i)$ es el i-enesimo centroide.
   $Distancia\ i,j$ es la distancia euclidiana entre $P(j)$ y $Centroide(i)$

4. Se realiza el cálculo de centroides, y se desplaza la semilla acorde a los grupos de datos formados en el paso anterior, es decir, se cambia las coordenadas de la semilla al centro de cada clúster, ahora cada clúster está asignado a una

única semilla central [44]. La Ecuación 1.16 calcula los centroides de cada clúster

$$Centroide(i) = \frac{1}{Ni}\sum_{j=1}^{Ni} P(j)$$
$$i = 1,2,3 \dots, k\ ; \qquad (1.16)$$

Donde Ni representa el total de datos que están asignados al clúster i en el paso anterior

5. Se repite los pasos 3 y 4 en varias iteraciones hasta que ninguno de los datos cambie de clúster en rondas consecutivas, o lo que es lo mismo, hasta que los centroides permanezcan iguales, en ese momento por más iteraciones nuevas que se realiza siempre se va a obtener idénticos resultados repetitivos.

El uso de k-means como aporte a las nuevas tecnologías para la segmentación de imágenes ha sido de gran utilidad en diversas aplicaciones, debido a que las pantallas o monitores utilizan el espacio de color RGB que está compuesto por tres dimensiones (rojo, verde, azul), se aplica el algoritmo k-means a cada una de las dimensiones mencionadas, el algoritmo se encarga de agrupar y formar clústeres de pixeles en base a similitudes cromáticas o de intensidad, pero los resultados experimentales indican que se consigue una mejor segmentación a partir del espacio de color L*a*b en comparación con el espacio de color RGB ,la conclusión al porqué de la mejora es por el aporte de información de cantidad de luz, saturación y tono que ofrece y cuantifica el espacio de color L*a*b [45].

En la Figura 1.14 se observa la representación cartesiana del plano a y b.

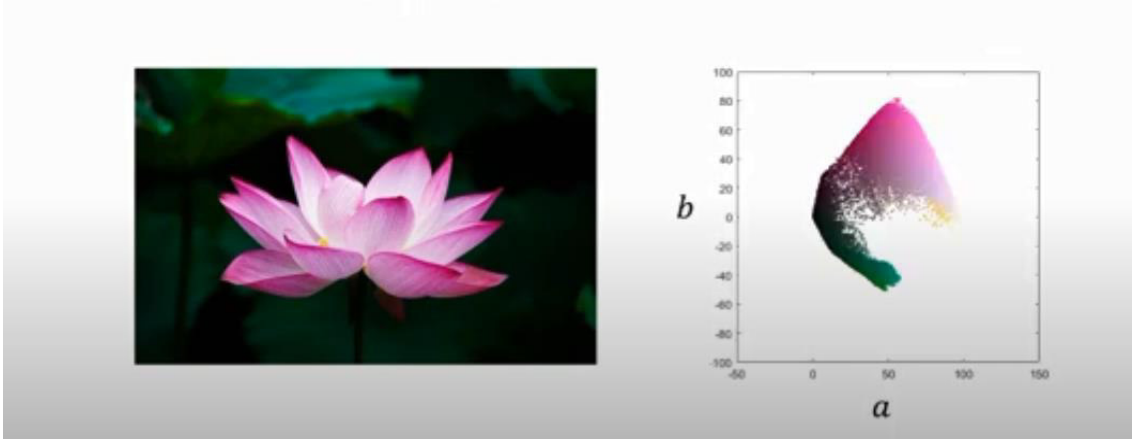


Figura 1.14 Representación cartesiana del plano a y b [46].

En la Figura 1.14 se repreresenta cada pixel en el plano a* b* del espacio de color L*a*b y se observa con claridad que la mayoría de los pixeles se encuentran alrededor de dos agrupaciones principales de color, entonces el objetivo de k-means es encontrar los centroides que mejor describen esta aglomeración de datos lo cual permite una exitosa segmentación de las imágenes [47].

1.3.6 DETECCIÓN DE OBJETOS EN IMÁGENES BINARIAS

Una imagen binaria se puede definir matemáticamente como una función F(x, y), donde las variables independientes x y y representa las coordenadas cartesianas de cada pixel en la imagen.

1.3.6.1 Vecindad de Pixeles

El concepto de vecindad de pixeles es ampliamente utilizado en diferentes campos del procesamiento de imágenes, como por ejemplo en algoritmos que permiten el conteo de objetos en imágenes binarias.

Se define como vecindad de pixeles a todos los pixeles inmediatos que rodean a un pixel central p(x, y) de referencia, el vecindario de pixeles puede ser interpretado como una pequeña matriz M que por lo general es de 3x3, donde la matriz M es subconjunto de la imagen F(x, y), los pixeles horizontales y verticales vecinos del pixel p(x, y) tienen las siguientes coordenadas:

$$(x+1, y), (x-1, y), (x, y+1), (x, y-1) \tag{1.16}$$

Los pixeles que se encuentran en las diagonales tienen las siguientes coordenadas:

$$(x-1, y-1), (x-1, y+1), (x+1, y-1), (x+1, y-1) \tag{1.17}$$

Se dice que todos los pixeles que pertenecen a un mismo vecindario están conectados entre sí, y el conjunto de vecindarios de tamaño 3x3 interconectados forman las llamadas componentes conexas de una imagen binaria [48].

1.3.6.2 Componentes Conexas

La noción de vecindad de pixeles es con frecuencia la base conceptual para algoritmos que permiten indicar el número de regiones u objetos presentes en la imagen, dichas regiones son denominadas técnicamente con el nombre de componentes conexas.

El método utilizado para la localización de componentes conexas en una imagen binaria es el denominado algoritmo de Labelling, el cual permite etiquetar con un valor numérico todas las componentes conexas en la imagen binaria, cabe recalcar que el algoritmo de Labelling arroja diferentes resultados que depende de la clase de conectividad utilizada [49].

1.3.6.3 Tipos de Conectividad

En las imágenes digitales existen dos tipos de conectividad

4-Conectividad: Se basa en escoger como vecinos solo a los pixeles que se sitúan en la dirección horizontal y vertical con respecto al pixel de referencia p(x,y) como se muestra en la Figura 1.15. el pixel central tiene 4 vecinos a su alrededor.

Figura 1.15 Representación gráfica de la conectividad-4

8-Conectividad: Se basa en escoger como vecinos conexos a los pixeles que se sitúan en la dirección horizontal y vertical, como también a los pixeles que se sitúan en las diagonales con respecto al pixel central de referencia, en la Figura 1.16 se visualiza que el pixel central tiene un total de 8 pixeles vecinos a su alrededor [50].

Figura 1.16 Representación gráfica de la conectividad-8

1.3.6.4 Algoritmo de Labelling

El algoritmo de labelling retorna como resultado una nueva imagen etiquetada con diferentes valores numéricos, donde cada etiqueta corresponden a cada componente conexa u objeto encontrado en la imagen, el algoritmo se caracteriza por etiquetar solo a los pixeles de valor 1, entonces los pixeles que pertenecen a una misma región conectada comparten igual valor numérico de etiqueta

De manera general el algoritmo de labelling consiste en hacer dos recorridos completos por cada uno de los pixeles, de izquierda a derecha y de arriba abajo.

Es importante enfatizar que al momento de aplicar el algoritmo se tiene la libertad de elegir el tipo de conectividad que se utiliza, y en base a dicha elección el algoritmo de labelling podrá dar dos resultados totalmente diferentes [51].

A continuación, se detalla los pasos en la aplicación del algoritmo de labelling.

Paso 1. En el primer recorrido se analiza uno a uno todos los pixeles de la imagen binaria, y se etiqueta con un valor numérico solo a los pixeles de valor 1 y con el mismo valor de etiqueta a sus vecinos conectados (8-conectividad o 4- conectividad).

Cabe recalcar que los únicos vecinos que se tomados en consideración por el algoritmo de labelling, ya sea con la conectividad-8 ó con la conectividad-4, son los pixeles vecinos localizados en la diagonal superior derecha y diagonal superior izquierda, además del pixel localizado en la parte superior y el localizado precisamente a la izquierda con respecto al pixel central $p(x, y)$ de referencia, en conclusión para el algoritmo de labelling cuando se utiliza la conectividad-8 considera como vecinos solo a 4 pixeles de un total de 8 vecinos posibles y en conectividad-4 considera como vecinos solo a 2 pixeles de un total de 4 vecinos posibles [52].

Al momento de hacer el recorrido pixel a pixel se puede encontrar que un pixel ya está etiquetado previamente con un valor diferente al elegido ese instante, entonces la solución será guardar los dos valores de etiqueta, la etiqueta antigua y la nueva como equivalencias, para así ser resueltas en un segundo recorrido, donde el objetivo va a ser elegir el valor más conveniente de entre todos los valores de equivalencias previamente guardados, que por lo general va a ser la etiqueta de menor valor [53].

Antes de etiquetar los pixeles existe la posibilidad de encontrar tres casos diferentes, a continuación, se detalla cada caso de aplicación.

Caso 1 En la Figura 1.17 se observa que con conectividad-8 o conectividad-4 todos los vecinos que rodean al pixel central de valor 1 se encuentran en un valor de 0, como conclusión se determina que se encontró un nuevo objeto en la imagen y de inmediato se asigna un nuevo valor de etiqueta (E), el algoritmo mantiene un contador de etiquetas que se incrementa en uno a medida que se encuentra un nuevo objeto en la imagen.

Figura 1.17 Etiquetaje (E) cuando se encuentra un nuevo objeto.

Caso 2 En la Figura 1.18 se observa que independientemente de la conectividad solo existe un pixel vecino de valor 1, entonces se asigna una etiqueta (E) idéntica al pixel central y al pixel vecino de valor 1.

Figura 1.18 Etiquetaje (E) cuando se encuentra un solo vecino.

Caso 3 En la Figura 1.19 el pixel central tiene más de un vecino con valor de 1, entonces puede ocurrir que existen pixeles vecinos ya etiquetados, la solución se determina asignando una de las etiquetas vecinas E1 o E2 al pixel central de referencia, lo común es asignar el valor de etiqueta del vecino que se encuentra más a la izquierda del pixel central de referencia, para luego guardar todas las equivalencias de etiquetas existentes.

Figura 1.19 Etiquetaje (E) cuando se encuentra más de un vecino.

Paso 2 Una vez que finaliza el proceso de etiquetar todos los pixeles, se hace un segundo recorrido pixel a pixel, ahora con el objetivo de resolver las equivalencias guardadas en el

Paso 1 con el propósito de asignar un solo valor de etiqueta a los pixeles que constituyen la componente conexa de la imagen, lo común es signar la etiqueta de menor valor de entre todas las equivalencias [54].

En el ejemplo particular que se indica en la Figura 1.20 se observa los diferentes resultados que arroja el algoritmo de labelling según la conectividad elegida, con conectividad-8 el algoritmo encuentra dos objetos en la imagen, mientras que con conectividad-4 el algoritmo encuentra a 4 objetos en la imagen, para obtener un resultado más preciso y detallado se recomienda utilizar la conectividad-8, ya que este tipo de conectividad permite incluir como vecinos a los pixeles localizados en las diagonales superiores respecto al pixel central de referencia, lo que origina una mayor eficiencia en los resultados finales.

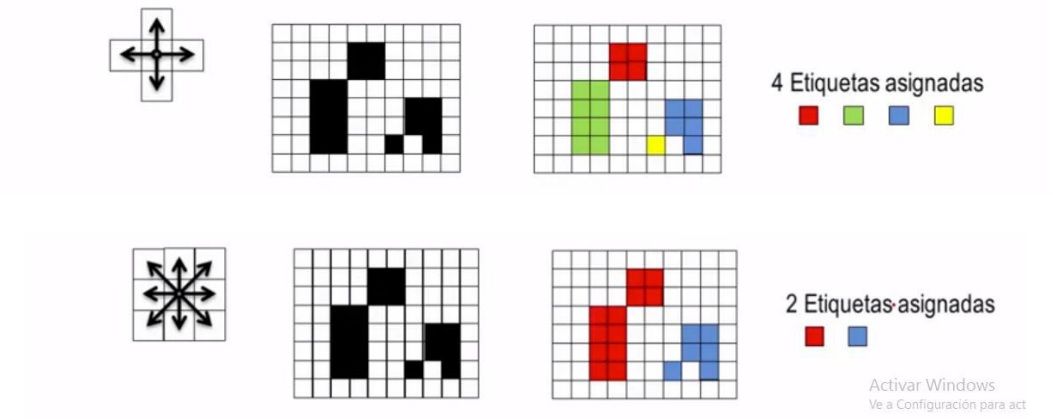


Figura 1.20 Resultados al aplicar el algoritmo de labelling con los dos tipos de conectividad existentes.

1.3.7 TRANSFORMADA DE HOUGH

El algoritmo de la transformada de Hough es un procedimiento matemático planteado para tratar de detectar las líneas rectas o círculos presentes en una imagen binaria, solo se analiza los pixeles de valor 1, cada pixel se representa en el plano bidimensional por una coordenada (x, y)

Debido a que los pixeles de valor 1 en una imagen binaria se encuentran distribuidos de una manera dispersa, una solución difícil y poco efectiva es tratar de ajustar los puntos a una recta o una circunferencia con el método de regresiones y mínimos cuadrados, por ello se hace imperativo utilizar otro método matemático como la transformada de Hough [55].

Para entender la transformada de Hough de una manera sencilla, se parte del caso más simple, que es la transformada de Hough para encontrar líneas rectas en una imagen binaria, un concepto similar se extiende y se aplica para encontrar círculos.

Se tiene una recta L en el plano cartesiano, y un punto p (pixel de valor 1) sobre dicha recta con coordenadas (xi, yi), entonces dichas coordenadas (xi, yi) cumplen la Ecuación 1.18 que es la ecuación de la recta, donde m es la pendiente y b es el punto de intersección de la recta con el eje y:

$$yi = mxi + b \tag{1.18}$$

.

Para la representación del algoritmo de Hough se deriva una fórmula alternativa de la ecuación de la recta, caracterizada por los nuevos parámetros (ϱ, θ) que reemplaza a los parámetros m y b [56].

En la Figura 1.21 ϱ es la distancia perpendicular desde el origen cartesiano hasta la recta L y θ es el ángulo que forma dicha recta con el eje x ,

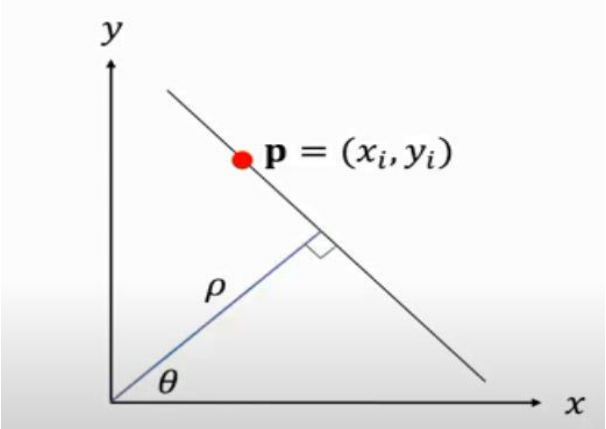


Figura 1.21 Gráfica de la recta en el plano cartesiano

Donde la ecuación alternativa de la recta es:

$$\varrho = xi.cos\theta + yi.sin\theta \tag{1.19}$$

Además la pendiente m y el valor de b se expresan en función de los nuevos parámetros (ϱ, θ)

$$m = -\frac{1}{tan\theta} \tag{1.20}$$

$$b = -\frac{\varrho}{sin\theta} \tag{1.21}$$

La Ecuación 1.19 representa todas las infinitas rectas que pueden pasar por la coordenada (xi, yi),por ende también existe infinitas combinaciones de (ϱ, θ) para representar las rectas, esto significa que cada punto(pixel) en la imagen tiene su propia Ecuación 1.19.

El algoritmo crea un arreglo bidimensional H(ϱ, θ), donde cada elemento de la matriz está asociado a una pareja de coordenadas (ϱ, θ) y en cada elemento se guarda el valor que representa el número de intersecciones entre las curvas (ϱ, θ) [57].

El objetivo del algoritmo es buscar todas las intersecciones posibles entre el total de curvas paramétricas (ϱ, θ) que representan a cada pixel existente de valor 1 en la imagen binaria, los puntos en el plano paramétrico(ϱ, θ) donde ocurre el mayor número de intersecciones representa la existencia de una línea recta con parámetros ϱ y θ, dicha recta a su vez incluye a todas las coordenadas posibles (xi, yi) que están sobre dicha línea recta, en la Figura 1.22 se ejemplifica la gráfica de dos curvas en el plano paramétrico(ϱ, θ) pertenecientes a las coordenadas $(x1, y1)$ y $(x2, y2)$, donde el número de intersecciones es registrado por un contador numérico que guarda los valores en la matriz H(ϱ, θ), el procedimiento continua hasta finalizar con la coordenada (xN, yN).

Los pares ordenados (ϱ, θ) en la matriz H que maximizan el número de intersecciones son reemplazados en la Ecuación 1.20 y Ecuación 1.21 para encontrar el valor de m y b correspondientes, para así encontrar la ecuación de la recta representada por la Ecuación 1.19.

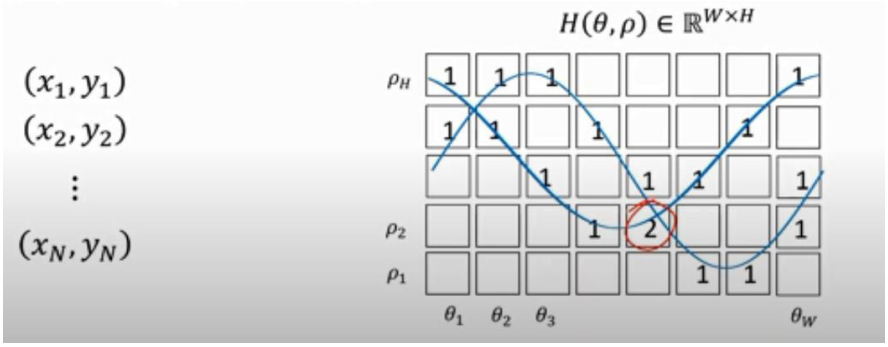


Figura 1.22 Intersección en un punto común de dos curvas (ϱ, θ), donde cada grafica representan la coordenada de un pixel.

La transformada de Hough para detectar círculos sigue una estrategia similar a la que se utiliza para detectar rectas.

Como se observa en la Figura 1.23 si un punto p de coordenadas (xi, yi) pertenece a un circulo de radio r y centro (xo, yo), el punto p satisface la Ecuación 1.22 del circulo:

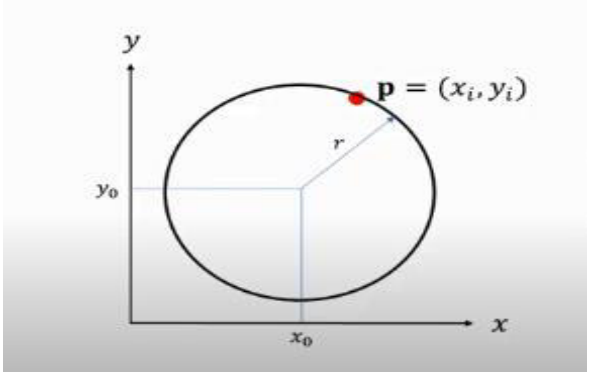


Figura 1.23 Gráfica de una circunferencia en el plano cartesiano

$$r^2 - (xi - xo)^2 + (yi - yo)^2 \tag{1.22}$$

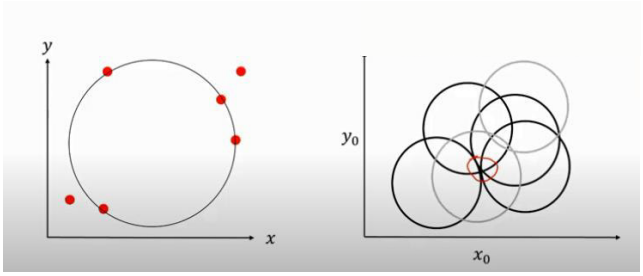
A partir de la Ecuación 1.22 la relación entre (xi, yi) se puede convertir en una relación entre los parámetros (xo, yo), esta relación queda establecida siempre y cuando se conozca el valor de radio r, la representación matemática está dada en la Ecuación 1.23

$$yo = yi \pm \sqrt{r^2 - (xi - xo)^2} \tag{1.23}$$

Para la detención de círculos con la transformada de Hough vamos asumir que se conoce el radio r del circulo y por lo tanto los parámetros a encontrar son los del centro del circulo (xo, yo),

En el espacio paramétrico (xo, yo) cada pixel de valor 1 en la imagen binaria equivale a una curva también con forma de circulo que relaciona a xo y yo. Se observa en la Figura 1.24 literal a) que todos los pixeles que se encuentran sobre un mismo circulo van a generar curvas paramétricas que siempre se intersectan en un punto común que corresponde a la pareja de valores (xo, yo) Figura 1.24 literal b), si se grafica las curvas paramétricas de puntos (xi, yi) que no se encuentran sobre el círculo dichas curvas no se intersectan [58].

Los puntos de intersección se guardan en un arreglo bidimensional H(xo, yo), donde cada elemento está asociado a una pareja (xo, yo) y el número de intersecciones es registrado por un contador numérico que guarda los valores en la matriz H(xo, yo)



a) **b)**

Figura1.24 Gráfica de puntos que pertenecen y no pertenecen a la circunferencia literal a) .Grafica de los puntos en el plano paramétrico (xo, yo) literal b).

Algoritmicamente la deteccion de circulos de Hough se describe en los siguientes pasos:

1. Se define un rango de valores de radio [$(rmin, rmax)$] para la busqueda del centro del circulo, además de un rango de valores de [$(xmin, xmax]$), [$(ymin, ymax)$].

2. Se encuentra las coordenadas (xi, yi) de cada pixel.

3. Se inicializa el arreglo bidimencional H(xo, yo), donde cada elemento esta asociado a una pareja (xo, yo), en este arreglo se asume un valor de radio fijo que pertenece al rango [$(rmin, rmax)$].

4. Se repeti el paso 3 para cada valor de radio dentre del rango [$(rmin, rmax)$].

5. Finalmente se escoge la terna de parametros que maximice el arreglo H(xo, yo).

1.3.8 DESCRIPTORES GEOMÉTRICOS EN IMÁGENES BINARIAS

Perímetro

Existen diversos métodos que permiten calcular el perímetro de un objeto presente en una imagen binaria, uno de uso común consiste en extraer primero su borde y luego contar los pixeles en el borde resultante, debido a algunas imperfecciones inevitables en el proceso

de digitalización, como por ejemplo contornos de curvas irregulares o bordes dentados, el valor del perímetro deducido con cualquier método no es el 100% preciso [58].

La Ecuación 1.24 representa el conteo de todos los pixeles que pertenecen al borde del objeto.

$$P = \sum_{f=0}^{f} \sum_{c=0}^{c} g(x,y) \;\; V\, g(x,y) con\ un\ pixel\ vecino\ V_4\big(g(x,y)\big) = 0 \tag{1.24}$$

Área

Para determinar el área de un elemento binario se realiza el conteo de todos los pixeles que pertenecen a ese objeto, la Ecuación 1.25 indica el recorrido por cada fila y columna del objeto [59].

$$A = \sum_{f=0}^{f} \sum_{c=0}^{c} g(x,y) \tag{1.25}$$

Factor de Redondez

El factor de redondez es una métrica que se encarga de definir qué tan redondo es un objeto, la métrica es una relación entre el área y el perímetro elevado al cuadrado del objeto y viene expresada por la ecuación 1.26

$$FR = \frac{4\pi A}{P^2} \tag{1.26}$$

La Ecuación 1.26 arroja un valor de 1 cuando es un circulo, para valores menores a 1 es otra figura geométrica diferente, se puede tomar un umbral por ejemplo de 0.90 para especificar que valores entre el rango [0.90-1] son círculos [60].

1.3.9 DESCRIPCIÓN DE HERRAMIENTAS UTILIZADAS

Para el desarrollo de la aplicación que permite el conteo de microorganismos en placas petrifilm utilizando visión computacional se hace el uso de la herramienta de software MATLAB.

MATLAB es un software numérico que se basa en el manejo de matrices con el objetivo de implementar algoritmos matemáticos, representar datos y crear interfaces dinámicas y

fáciles de majear para el usuario, entre sus diversas aplicaciones y bondades en los diversos campos de estudio, Matlab ofrece el paquete Image Processing Toolbox que suministra un diverso conjunto de algoritmos que consienten el procesamiento, la investigación y visualización de imágenes, además proporciona el ambiente idóneo para la creación de nuevos algoritmos [61].

Con la herramienta Image Processing Toolbox se automatiza los flujos de programas lo que facilita la manipulación de imágenes cuando se tiene un gran volumen de datos, esto con el objetivo de segmentar y mejorar las imágenes permitiendo realizar análisis geométricos y comparativos entre las diversas técnicas aplicadas a imágenes

1.3.10 TRABAJOS ANTERIORES

En los recursos literarios científicos que se investigó para el presente trabajo de titulación no se ha encontrado un trabajo similar en el contexto especifico aplicado al conteo de microorganismos en placas petrifilm utilizando el software MATLAB, existe diversos trabajos de estudio relacionados con el procesamiento de imágenes en otros campos de investigación que sirvieron como guía para el desarrollo del presente trabajo de titulación que en su mayor parte fue creado en base a la experimentación mediante prueba y error de todos los recursos que proporciona el software Matlab para el tratamiento de imágenes, con el objetivo que los algoritmos utilizados encajen a conveniencia con las imágenes proporcionadas por la USFQ y así obtener los resultados esperados.

2. METODOLOGÍA

En la presente sección se describirá el desarrollo de un programa en Matlab que permita el conteo de microorganismos en las placas petrifilm utilizando visión computacional. Para facilidad de interacción entre el usuario y el programa se implementó una interfaz gráfica en la plataforma App Designer de Matlab que permite indicar los resultados de una manera clara.

El presente capítulo constituye de cuatro secciones principales, la sección 2.1 indica el procedimiento para aislar o segmentar la sección de interés de la imagen, la sección 2.2 detalla el conteo de los microorganismos en la placa petrifilm, luego la sección 2.3 muestra el uso de descriptores geométricos para el análisis de áreas, perímetros en los microorganismos de mayor tamaño, y finalmente la sección 2.4 describe la implementación de la interfaz gráfica en app designer que indica los resultados.

El presente trabajo de titulación se desarrolló a partir de un conjunto de imágenes digitales de prueba de las placas petrifilm que fueron entregadas por el departamento de Microbiología de USFQ.

En la Figura 2.1 se observa la foto de una placa petrifilm, donde los puntos rojos son los microorganismos incubados presentes en la placa. Si se analiza el tamaño de los microorganismos, se observa que existe dos grupos de microorganismos, los microorganismos grandes con un tamaño estándar, que son de fácil reconocimiento visual, lo que proporciona un fácil conteo, mientras que, a su vez existen microorganismos diminutos de difícil visualización, lo que ocasiona que los técnicos del laboratorio se ayuden de un lente de aumento para facilitar el conteo.

Las líneas cuadriculadas de color amarillo que se observa en la Figura 2.1 de la placa petrifilm sirve como una guía a los técnicos de laboratorio al momento de realizar el conteo utilizando el sentido de la vista, se visualiza que también existen microorganismos sobre las líneas cuadriculadas.

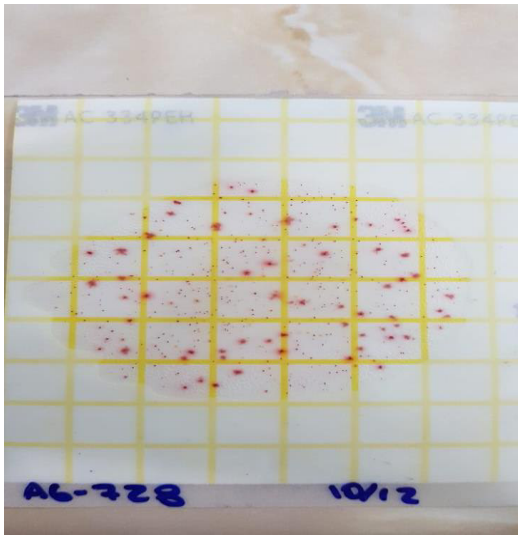


Figura 2.1 Imagen de placa petrifilm.

En la Figura 2.2 se resume en un diagrama de bloques las etapas del algoritmo que se detallará en las siguientes secciones.

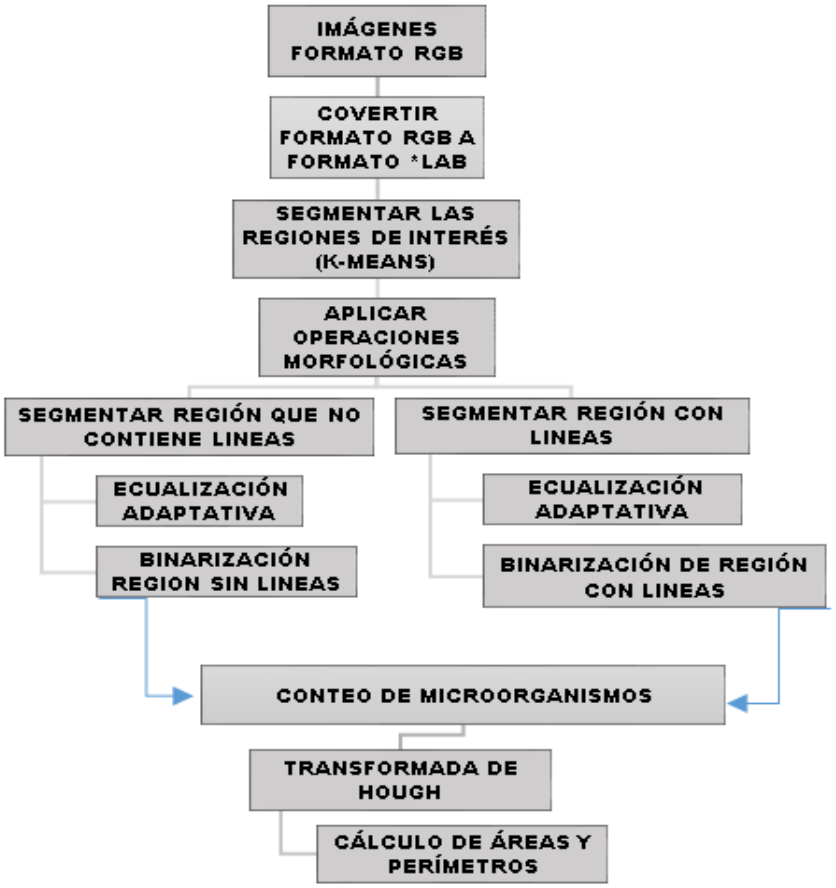


Figura 2.2 Diagrama de bloques de los algoritmos aplicados.

2.1 SEGMENTACIÓN DE LA SECCIÓN DE INTERÉS UTILIZANDO EL ALGORITMO DE INTELEGENCIA NO SUPERVISADA K-MEANS.

En la Figura 2.1 se visualiza que además de la región de interés (región donde se encuentran los microorganismos) existen diferentes regiones con diversas tonalidades, regiones que si no se logran eliminar, al momento de procesar la imagen pueden causar serios problemas que originen confusión en la interpretación y análisis de resultados, por lo cual, se hace imperativo encontrar un método que permita segmentar la imagen solo en la región de interés (región donde se encuentran los microorganismos).

Antes de empezar con el análisis, es obligatorio que las imágenes se guarden en una PC, que previamente este instalado el software Matlab, las imágenes se pueden guardar en un registro de carpetas de libre elección por el usuario, ya que desde la aplicación App Designer se cargará la imagen elegida por el técnico de laboratorio para su posterior conteo automático.

Por intermedio del comando `uigetfile` se permite al usuario ingresar desde Matlab a cualquier dirección de carpeta guardada en la PC y elegir la imagen que desea analizar, las variables de salida del comando `uigetfile` son el nombre de la imagen y la ruta del archivo donde se encuentra la imagen. El procedimiento antes mencionado se detalla en el Segmento de código 2.1.

```
% LOCALIZAR LAS CARPETA DONDE SE ENCUENTRAN LAS IMÁGENES %
[FileName,PathName]=uigetfile('*.jpeg','select an image ');
```

Segmento de código 2.1: Permite elegir y Cargar la imagen

La función `uigetfile` utilizó los siguientes argumentos de entrada y salida:

- `*.jpeg:` Es la variable de entrada que especifica que formato de imagen se va a buscar, en este caso la imagen está en formato jpeg.
- `select an imagen:` Es la variable de entrada que indica que el tipo de archivo va a ser una imagen.
- `FileName:` Es la variable de salida que representa el nombre del archivo.

- `PathName`: Es la variable de salida que hace referencia a la ruta del archivo donde se encuentra la imagen.

Una vez que se eligió la imagen se procede a leer la misma con el comando `imread` y para su posterior visualización se utiliza el comando `imshow,` el procedimiento se indica en el Segmento de código 2.2.

```
% LEER Y VISUALIZAR LA IMAGEN ELEGIDA EN EL REGISTRO DE CARPETAS %
% LEER Y VISUALIZAR LA IMAGEN ELEGIDA EN EL REGISTRO DE CARPETAS
app.he=imread(strcat(PathName,FileName));
%app.he:   Es   la   variable   donde   se   guarda   la   imagen.
imshow(app.he,'parent',app.UIAxes);
```

Segmento de código 2.2: Leer y visualizar imagen.

La función `imread` utilizó los siguientes argumentos de entrada y salida:

- Las variables `PathName` y `FileName` contienen el nombre y la ruta del archivo, además se utiliza la función `strcat` que permite concatenar cadenas de caracteres, con el objetivo de unir el nombre del archivo seguido de la ruta, con el objetivo de guardar en una sola variable que servirá como entrada a la función `imread.`

- `app.he`: Es la variable de salida de la función `imread,` donde se guarda la imagen que será utilizada durante todo el procesamiento.

En la Figura 2.3 se observa una pequeña parte de los valores que arroja la matriz en formato RGB, los valores deben ir de 0 a 255.

203	204	201	206	211	210	205	201	205	208	216	216	217	211	211	213	219	224	221	217	218
200	199	201	204	212	208	204	200	202	202	214	218	212	212	209	212	217	224	221	218	216
198	198	196	205	214	209	204	198	202	202	211	211	214	207	209	209	216	222	223	218	218
191	193	195	202	213	209	204	199	199	203	212	209	210	208	203	206	214	219	220	214	217
196	192	190	203	213	212	208	199	199	200	213	209	209	204	204	206	211	219	213	211	208
191	190	185	194	213	212	206	196	194	199	206	204	205	201	199	202	209	215	215	207	203
187	177	183	189	207	204	199	191	188	187	197	209	205	198	192	193	199	214	208	205	204
179	171	165	178	191	188	182	172	171	168	183	183	180	171	171	176	192	219	213	206	201
184	173	177	181	186	175	165	155	150	149	140	136	132	129	138	141	172	209	220	211	209

Figura 2.3 Matriz en formato RGB.

Para segmentar la imagen y obtener la sección de interés (región donde se encuentran los microorganismos), se realizó pruebas con diversos algoritmos de inteligencia no supervisada, el algoritmo con el que se obtuvo mejores resultados fue el algoritmo k-mens.

Para trabajar con k-means y obtener efectos óptimos en la segmentación se debe transformar la imagen del espacio de color RGB (Espacio de color por defecto) al espacio de color L*a*b.

El comando que se utiliza para dicha transformación de espacio de color es `rgb2lab`, el cual recibe como argumento de entrada la imagen en formato RGB y devuelve como salida la imagen en formato L*a*b como se indica en el Segmento de código 2.3.

El espacio de color L*a*b está formado por tres planos (Plano L, plano*a y *b), al graficar cada plano utilizando la aplicación" Color Theresholder" de Matlab, se forma una representación del color en tres dimensiones de la imagen de la placa petrifilm como se observa en la Figura 2.4.

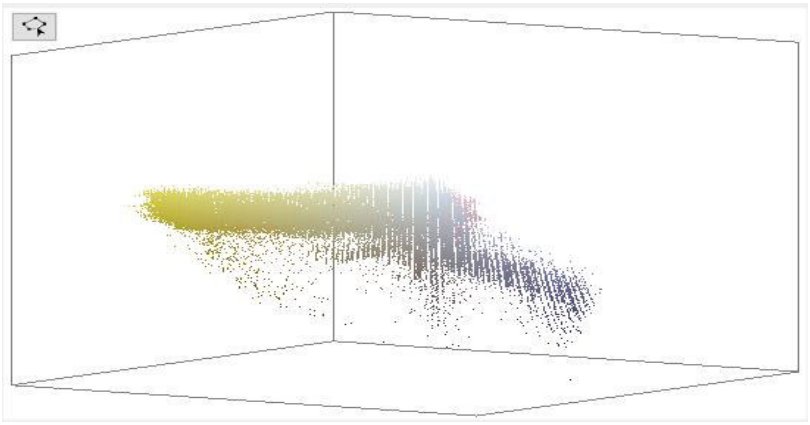


Figura 2.4 Representación gráfica de la placa petrifilm en tres dimensiones del espacio de color L*a*b.

Para el análisis con el algoritmo K-means solo es necesario trabajar con los planos de color *a y * b, ya que solo estos planos son los que contienen la información de la tonalidad, mientras que el plano L contiene la información de luminosidad que es menos relevante para el presente análisis, para mayor facilidad con el manejo de datos se transforma cada plano al formato single, el cambio de formato es una recomendación de Matlab debido a que single reorganiza cada dato con un valor de 4 bytes, lo que permite una mejor precisión del algoritmo, el comando que se utiliza para dicha transformación es `im2sing` el proceso mencionado se detalla en la Sección de código 2.3.

```
%Convertir la imagen del formato RGB a formato L*a*b
%Solo se elige los planos a* y b* para el análisis ya que estos
contiene la informacion de la tonalidad.
lab_he = rgb2lab(he)
```

Segmento de código 2.3: Transformar del espacio de color RGB al espacio L*a*b.

En la Figura 2.5 se observa una pequeña parte de los valores que arroja la matriz ya transformada al espacio de color L*a*b, los valores deben ir de -100 a 100.

71.0878	71.4557	70.8412	72.4618	74.6193	73.7699	71.4533	70.0190	71.5137	72.8867	75.8086	76.0377
70.2544	69.8852	70.6232	71.7461	74.9841	73.0379	71.0853	69.6495	70.4084	70.6804	75.0802	76.7643
70.0079	70.0442	69.3231	72.4234	75.7514	73.9667	71.6487	69.4347	69.5952	70.4084	73.9851	74.4862
68.2226	68.9645	68.9528	71.3205	75.3873	73.9274	72.1338	69.8047	68.4851	70.1750	74.5401	73.4435
69.4896	67.7854	67.0415	71.8864	75.0397	74.7131	73.5616	69.7648	69.2396	69.8228	74.5184	73.5794
67.6361	67.8510	65.9875	69.3710	75.5438	75.1984	73.6280	69.7315	68.4404	70.2528	73.0059	72.2722
66.7341	63.5265	65.5530	68.6253	75.2493	74.4668	72.6725	69.7596	67.8886	67.5160	71.4412	75.8391
64.1068	61.6479	59.4025	64.8715	69.7596	68.9259	66.6902	62.9701	61.8418	60.7019	66.9028	66.9028
66.4567	62.5995	64.1097	65.8329	67.4832	63.8996	60.1071	56.3094	53.8567	53.2211	50.2492	48.6672
58.6772	54.3891	47.0761	50.6433	48.3125	45.1158	39.8360	34.6202	31.9747	29.4159	29.5699	25.2314
43.3811	40.1247	28.7093	27.4859	29.6501	27.0199	25.0115	21.9600	21.4736	20.5796	20.5307	19.1370
23.9118	23.9528	21.5194	22.1751	21.7772	19.9759	19.5328	18.6537	20.8882	20.8882	20.8320	19.1065
17.8514	17.8934	21.2790	20.9044	21.1894	18.9411	20.7318	21.6361	25.2330	24.3543	27.8068	25.1838
21.0701	23.0670	25.7685	25.3551	27.3549	27.8467	31.6498	32.4987	37.5832	38.8232	44.1207	40.0459
28.5393	29.4009	34.1282	42.1782	41.0309	45.0357	52.5737	55.1410	53.6380	60.1927	73.1005	66.4195

Figura 2.5 Matriz en formato L*a*b.

En el presente trabajo solo se requiere analizar las componentes de color, por tal motivo se toma los planos de color para luego ser transformados a un tipo de dato single.

```
ab1 = lab_he(:,:,2:3);
%Transformar los  parámetros a* y b* a tipo de dato single. La
variable %Ab1 es la imagen de salida de la función im2single.
ab1 = im2single(ab1);
```

Segmento de código 2.4: Separar los planos de color y trasformar al tipo de datos single.

Una vez que se preparó los datos contenidos en la imagen de una manera adecuada, los mismos se utilizan como entrada en el algoritmo de inteligencia no supervisada k-means, la función que representa el algoritmo k-means en Matlab es `imsegkmeans`, el procedimiento de segmentación se indica en la Sección de código 2.5.

La finalidad de k-means es dividir la imagen en grupos denominados clústeres, el número de clústeres es de libre elección por los usuarios.

Sintaxis Función Kmeans:

```
L = imsegkmeans(I,k)
```

```
L = imsegkmeans(I,k,Name,Value)
```

Descripción:

`L = imsegkmeans(I,k)` segmenta la imagen(`I`) en clústeres(k) realizando la agrupación en clústeres k-means y devuelve la salida segmentada (L) etiquetada en una imagen en formato binario.

`L = imsegkmeans(I,k,Name,Value)` utiliza argumentos nombre-valor para controlar aspectos del algoritmo de agrupación en clústeres k-means.

Los argumentos de entrada y salida de la función `imsegkmeans` para la aplicación particular son los que se indican a continuación.

- `ab1:` Es un argumento de entrada y significa la imagen que se requiere segmentar.
- *`NColores:`* Es un argumento de entrada que indica el número de clústeres en el que se desea segmentar la imagen, es arbitrario, y se elige al inicio del proceso, su elección se basa en la experimentación. Para la presente aplicación se eligió dos clústeres.
- NumAttempts*:* Número de veces que se repetirá el proceso de agrupación en clústeres utilizando nuevas posiciones centroide de clúster iniciales, especificadas como el par separado por comas que consta de y un entero positivo.'NumAttempts'
- *`NormalizeInput:`* Indica que se va a normalizar los datos de entrada.
- *`MaxIterations:`* K-means funciona como un proceso iterativo, donde en cada repetición se redefine el centroide y el número de elementos que pertenecen al clúster, se determina que el proceso ha finiquitado cuando se consigue que no existan más cambios en los clústeres y en los centroides.

  El valor por defecto es de 100 repeticiones, el cual fue aprovechado de manera satisfactoria para la presente aplicación.
- *`Threshold:`* Indica la precisión del algoritmo, que se encuentra definida por un umbral, ayuda a detener el algoritmo cuando los centroides en dos iteraciones consecutivas no cambian su posición dentro de un cierto rango de umbralidad. El

umbral elegido experimentalmente es de valor 5. Si cumple este aspecto el algoritmo puede parar antes de llegar a las 100 repeticiones.

- *pixeles_labelss:* Es el resultado y argumento de salida del algoritmo k-means, guarda todos los clústeres en el que se fragmento o dividió la imagen. Cada clúster se representa con la forma de una imagen binaria, que sirve como máscara que posteriormente se multiplica por la imagen original y como resultado se consigue la segmentación deseada.

```
%Se elige dos clusters se segmentación
NColores = 2;
%Paramentros de entrada
%ab1:Imagen de entrada.
%MaxIteracciones: Es el número de Iteraciones.
%Thereshold: Muestra el umbral de precisión
%NormalizeInput: Muestra la normalizacion de los datos.
pixeles_labelss =
imsegkmeans(ab1,NColores,'NumAttempts',2,'NormalizeInput',false,'MaxI
terations',100,'Threshold',5);
title('Image Labeled by Cluster Index');
```

Segmento de código 2.5: Aplicación del algoritmo k-means

En base a la experimentación, se concluyó que el número de clústeres más adecuado para la aplicación debe tener un valor dos (`NColores= 2`), es decir, se segmenta la imagen en dos conjuntos de pixeles, donde cada pixel perteneciente a un grupo tiene similitudes en su tonalidad.

Para extraer cada clúster guardado en la variable de salida `pixeles_labelss` se utiliza el número de clúster correspondiente, en la presente aplicación se obtuvo dos clústeres, clúster 1 y clúster 2. El procedimiento de obtención del clúster 1 se observa en la Segmento de código 2.6

```
%Obtenemos el cluster 1 resultado del algoritmo k-means
mask1=pixel_labels==1; %Se extrae el cluster numero 1 denomidado
mask1
figure
imshow(mask1)
title('MASCARA 1');
```

Segmento de código 2.6: Obtención del clúster número 1

El cluster número 1 está representado como una imagen binaria que se observa en la Figura 2.6.

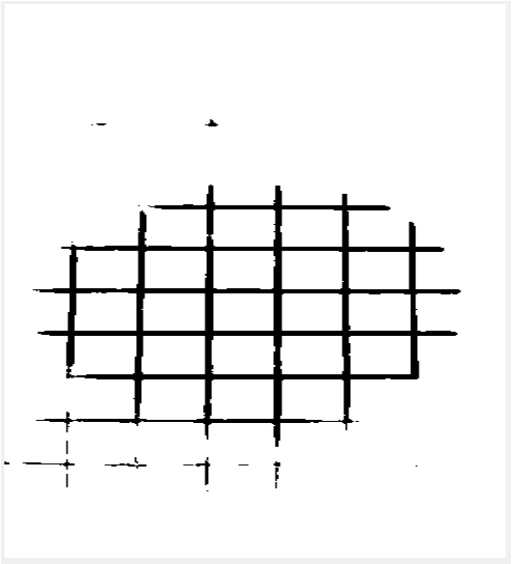


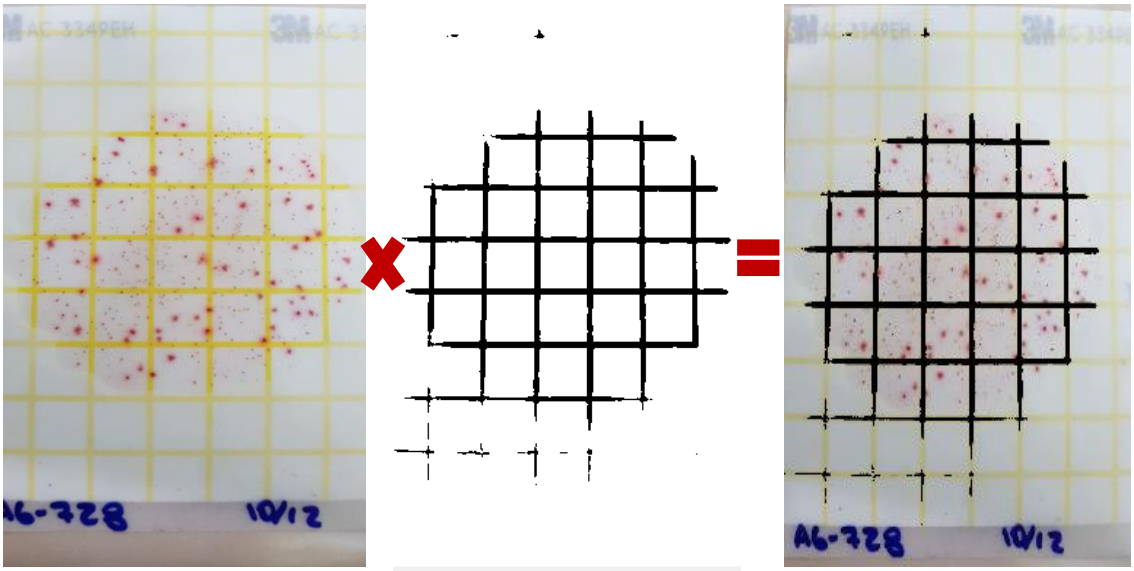
Figura 2.6 Imagen del Clúster 1

Los clústeres que se obtiene como salida del algoritmo k-meas no representa la imagen a color segmentada, para conseguir la imagen segmentada, se multiplica la máscara o clúster 1 por la imagen original. El procedimiento se visualiza en el Segmento de código 2.7.

```
%%obtener la segmentacion a color del cluster 1
imagen_color_cluster1 = app.he.*uint8(mask1); %Se multiplica el
cluster por la imagen original, el resultado es imagen_color_cluster1
figure
imshow(imagen_color_cluster1)
title('imagen segmentada a color por el cluster 1');
```

Segmento de código 2.7: Obtención de la imagen segmentada a color

Antes de multiplicar la imagen original que está representada por la Figura 2.7 literal a) y la máscara literal b), se transforma los datos pertenecientes al clúster 1 a tipo de dato `uint8` el procedimiento de conversión de tipo de dato es una recomendación de Matlab para mejorar la eficiencia en memoria, ya que el tipo de dato `uint8` ocupa 1byte y es suficiente para manejar valores binarios. El resultado de la segmentación a color por el clúster 1 se observa en la Figura 2.7 literal c).



a) b) c)

Figura 2.7 Imagen a color segmentada por el clúster 1 literal c).

Similar procedimiento se realiza para la extracción del clúster 2. Los pasos de extracción del clúster 2 y segmentación de la imagen a color por el clúster 2 se detalla en el Segmento de código 2.8

```
%Extraccion de  la máscara binaria correspondiente al cluster
numero2, producto del algoritmo k-means
maska2=pixel_labels==2;
figure
imshow(maska2)
title('MASCARA 2');
%% Obtener la segmentacion a color del cluster 2
imagen_color_cluster2 = app.he.*uint8(maska2);
figure
imshow(imagen_color_cluster2)
title('imagen segmentada a color por el cluster 2');
```

Segmento de código 2.8: Extracción del clúster 2 y segmentación de la imagen a color mediante el clúster 2.

En la Figura 2.8 se visualiza la imagen binaria del clúster 2

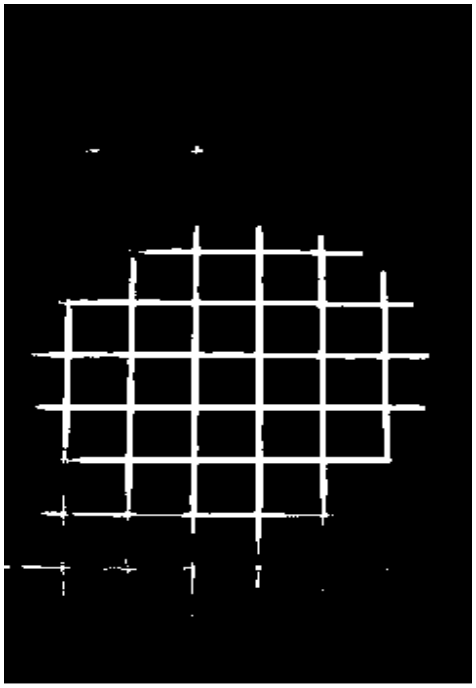
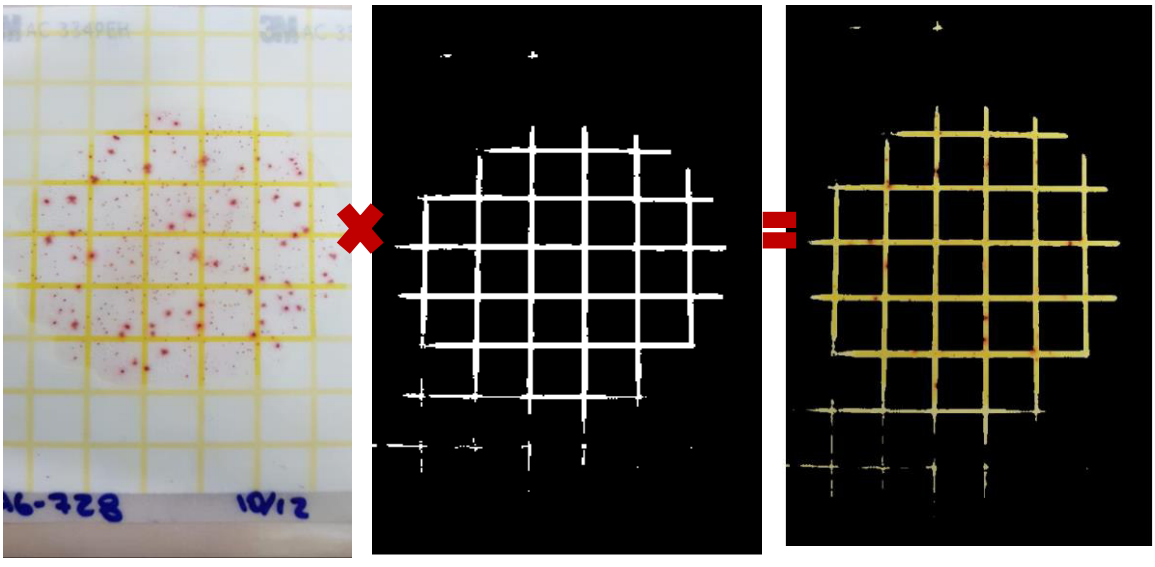


Figura 2.8 Imagen del clúster 2

Para obtener la segmentación a color del clúster 2 en la Figura 2.9 se observa que se multiplica la imagen original literal a) por el clúster 2 literal b) y se consigue como resultado la imagen a color segmentada literal c).



a) b) c)

Figura 2.9 Imagen a color segmentada por el clúster 2 literal c).

El objetivo de utilizar k-means es aislar la sección de interés(región donde se encuentran los microorganismos), mediante la división de la imagen en grupos que contengan tonalidades similares, con el propósito, que al momento del procesamiento se evite errores en el conteo de los microorganismos, en la Figura 2.9 literal c) se observa que se consiguió el objetivo, ya que se logró aislar la sección de la imagen que contiene solo a las líneas (cuadrícula de líneas amarillas), que a su vez presentan tonalidades similares, lo que facilita el procesamiento de esa sección de la imagen.

Al analizar la Figura 2.7 literal c) se visualiza que no se logró una segmentación adecuada, porque aún existen diferentes tonalidades presentes en la imagen, que al momento del procesamiento van a traer graves problemas en el conteo. La solución que se encontró a este problema se centraliza a partir del clúster 2, este clúster en su forma es el que más se aproxima o se asemeja para construir una máscara y segmentar la sección de interés (sección donde se encuentran los microorganismos).

El clúster 2 se procesará utilizando operaciones morfológicas con la finalidad de encontrar una máscara adecuada para la segmentación de la sección de interés.

Se inicia eliminando las secciones diminutas de pixeles que se encuentran dispersas alrededor de la sección principal del clúster 2, esto se consigue con la función `bwareafilt`, donde los argumentos de entrada son el clúster 2 y el valor numérico de 1, dicho valor numérico indica que solo se desea conservar la sección de mayor tamaño, mientras que las secciones pequeñas restantes se eliminan. En el Segmento de código 2.9 se detalla el procedimiento antes mencionado.

Sintaxis función `bwareafilt:`

```
BW2 = bwareafilt(BW,n)
```

Descripción:

`BW2 = bwareafilt(BW, n)` mantiene los `n` objetos más grandes de la imagen `BW` y los demás los elimina, donde n=1,2,..m. La imagen `BW2` de salida contiene los objetos de mayor tamaño.

```
%Eliminamos los pequeños pixeles que se encuentran alrededor de los
%objetos grandes.
%El argumento de entrada es el cluster 2(maska2) y el valor numérico
de 1
maska2=bwareafilt(maska2,1);
figure
```

```
imshow(maska2)
title('Eliminar las pequeñas secciones');
```

Segmento de código 2.9: Se eliminan las pequeñas secciones de pixeles que se encuentran alrededor del objeto más grande.

En la Figura 2.10 se visualiza que se eliminó las secciones diminutas presentes en el clúster 2.

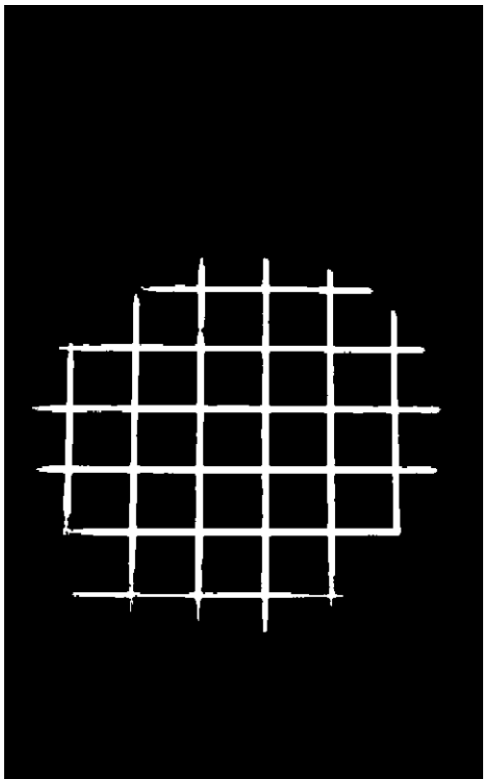


Figura 2.10 Se elimina las secciones pequeñas de pixeles

A continuación, se utiliza la operación morfológica para el llenado de agujeros `imfill`, dicha función considera como "huecos" a todos los pixeles de valor 0 (fondo de color negro) que están completamente rodeados por pixeles de valor 1 (color blanco), el objetivo del algoritmo es llenar todos los huecos con pixeles de valor 1. Después de aplicar el algoritmo de llenado de huecos, de nuevo se aplica la función bwareafilt, para eliminar las pequeñas secciones de pixeles que son el resultado de aplicar la función `imfill`. La resolución se encuentra detallada en el segmento de código 2.10.

Sintaxis imfill:

```
BW2 = imfill(BW,'holes')
```

Descripción:

`BW2 = imfill(BW,'holes')` rellena los agujeros de la imagen binaria de entrada BW. En esta sintaxis, un agujero es un conjunto de píxeles de fondo que se encuentran bordeados por pixeles de valor 1.

```
%Operación morfológica de Relleno de Agujeros.
%Argumento de entrada es el cluster 2 (maska2)
c1=imfill(maska2,'holes');
figure
imshow(c1)
title('Rellenos de huecos')
%Escoger el objeto más grande y eliminar los Pequeños.
a1=bwareafilt(c1,1);
```

Segmento de código 2.10: Relleno de agujeros en el cluster2.

Los argumentos de entrada de la función `imfill son:`

- *'holes':* Especifica que los pixeles del fondo son de valor 0.
- maska2: Imagen de entrada.

El resultado del relleno de huecos se visualiza en Figura 2.11.

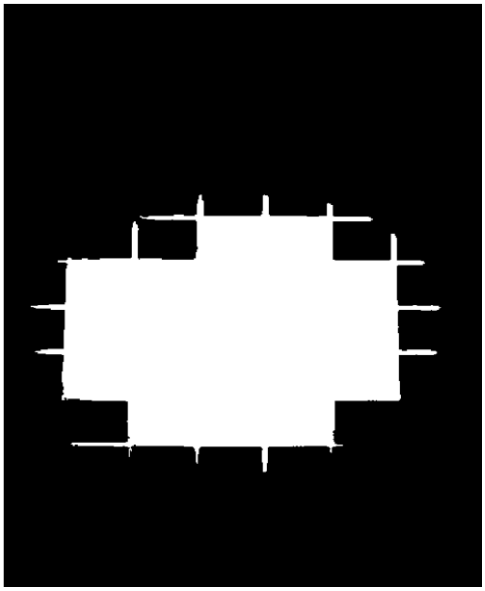


Figura 2.11 Relleno de agujeros del clúster 2

A continuación, se aplica la operación morfológica de dilatación, para ello se utiliza la función `imdilate,` el objetivo es cerrar algunos agujeros y unir regiones próximas.

La función `strel` permite la creación del elemento estructurante, el elemento estructurante que se utiliza es un circulo de radio 40. En el segmento de código 2.11 se detalla el procedimiento.

Sintaxis Función imdilate:

```
J = imdilate(I,SE)
```

Descripción:

`J = imdilate(I,SE)` dilata la imagen binaria (`I`), devolviendo la imagen dilatada (`J`), `SE` es un objeto de elemento estructurante o matriz de objetos de elemento de estructuración

```
%Se aplica una operación morfológica de dilatación
%Se crea un elemento estructurante en forma de disco de radio 40.
s2=strel('disk',40); %Dilatacion con un elemento estructurante en
forma de disco.
c2=imdilate(a1,s2);
figure
imshow(c2)
title('Operación Morfológica de Dilatación ')
```

Segmento de código 2.11: Operación Morfológica de Dilatación.

Los argumentos de entrada y de salida de la función `strel` son los siguientes:

- *'disk':* Muestra que el elemento estructurante va a ser un disco, separado por una coma se indica el valor del radio del circulo que es de 40 pixeles.
- El argumento de salida de la función es `s2` que representa el elemento estructurante.

Para la función `imdilate` se utiliza los argumentos de entrada y salida nombrados a continuación:

- a1: Es la imagen de entrada que se desea dilatar
- s2: Es el elemento estructurante definido por la función `strel.`

Seguido de aplicar la dilatación se aplica la operación morfológica de llenado de huecos, el procedimiento se detalla en el Segmento de código 2.12.

```
%Operación morfológica de Relleno de Agujeros.
c3=imfill(c2,'holes');
```

```
figure
imshow(c3)
title('Relleno de huecos')
```

Segmento de código 2.12: Operación Morfológica de Relleno de agujeros.

Para unir roturas y suavizar la imagen, sin cambiar los tamaños originales del objeto se aplica la operación morfológica de cierre, que consiste en una dilatación seguida de una erosión utilizando el mismo elemento estructurante. La función que se utiliza para el cierre morfológico es `imclose.` El elemento estructurante que se utiliza es un circulo con radio de 40, el procedimiento se desarrolla en el Segmento de código 2.13.

Sintaxis de `imclose:`

```
J = imclose(I,SE)
```

Descripción:

`J = imclose(I,SE)` realiza el cierre morfológico en la imagen binaria (I), devolviendo la imagen cerrada J, donde el elemento estructurante es SE

```
%Operacion Morfológica de Cierre
%Se crea un elemento estructurante en forma de disco de radio 40.
s1=strel('disk',40);
%Se aplica la operación morfológica de cierre
c4=imclose(c3,s1);
figure
imshow(c4)
title('Operación Morfológica de Cierre')
```

Segmento de código 2.13: Operación Morfológica de Cierre.

La función `imclose` utiliza los argumentos de entrada y salida nombrados a continuación:

- c3: Es la imagen de entrada a la que se desea aplicar el cierre morfológico.
- s1: Es el elemento estructurante definido por la función `strel.`

En la Figura 2.12 se visualiza el resultado de aplicar las operaciones morfológicas de dilatación y cierre.

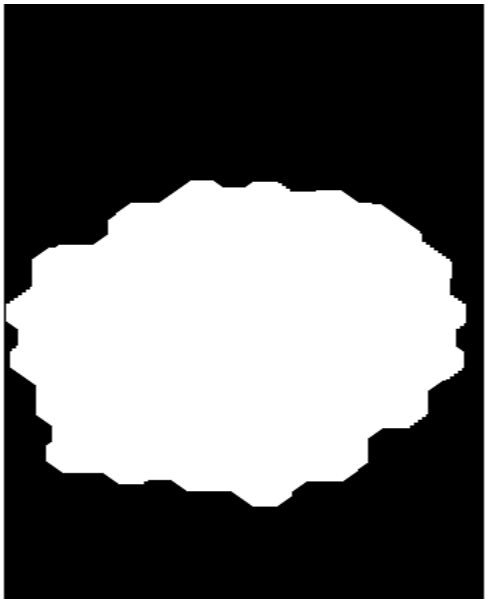


Figura 2.12 Resultado de aplicar las operaciones de dilatación y cierre morfológico

En resumen, a partir del clúster 2 que se obtuvo con K-means se aplicó una serie de operaciones morfológicas y se consiguió una máscara (Figura 2.12) que encaja a la perfección en forma y tamaño con el objetivo de aislar la sección de interés (sección donde se localizan los microorganismos) de la imagen de la placa petrifilm.

Mediante las diversas pruebas que se realizó con diferentes imágenes se consiguió una segmentación cien por ciento efectiva, pero en contados casos se observó, que a pesar de la adecuada segmentación, en algunas imágenes existía una intersección entre la sección segmentada y el borde de la imagen, lo que dificultaba el conteo de los microorganismos, el problema se solucionó construyendo una imagen binaria de tamaño idéntico a la imagen analizada, dicha imagen creada tiene como condición que todos los pixeles de los bordes siempre tomen el valor de 0 y todos los demás pixeles sobrantes tomen el valor de 1, la imagen construida se detalla en el segmento de código 2.14. La imagen creada sirve como máscara que posteriormente se multiplicará con la imagen de la Figura 2.12, y así se consigue que nunca exista dicha intersección entre la sección de interés y los bordes de la imagen, evitando problemas en el conteo de los microorganismos, cabe recalcar que este inconveniente se presentó en un grupo aislado de imágenes.

```
%El objeto segmentado puede intersectarse con los bordes de la
imagen.
%lo que origina una dificultad al momneto del conteo.
%Para solucionar este inconveniente se crea una imagen binaria v que
%permite que se realice un mejor conteo.
for i=1:1280
```

```
    for j=1:720
       if j>=1 & j<=8 | j>=711 & j<= 720 |i>=1000
       v(i,j)=0;
       else
        v(i,j)=1;
       end

   end
end
%Multiplicación entre la imagen binaria v y la imagen segmentada.
mascara=v.*c4;
figure
imshow(mascara)
title('Máscara de imagen segmentada');
```

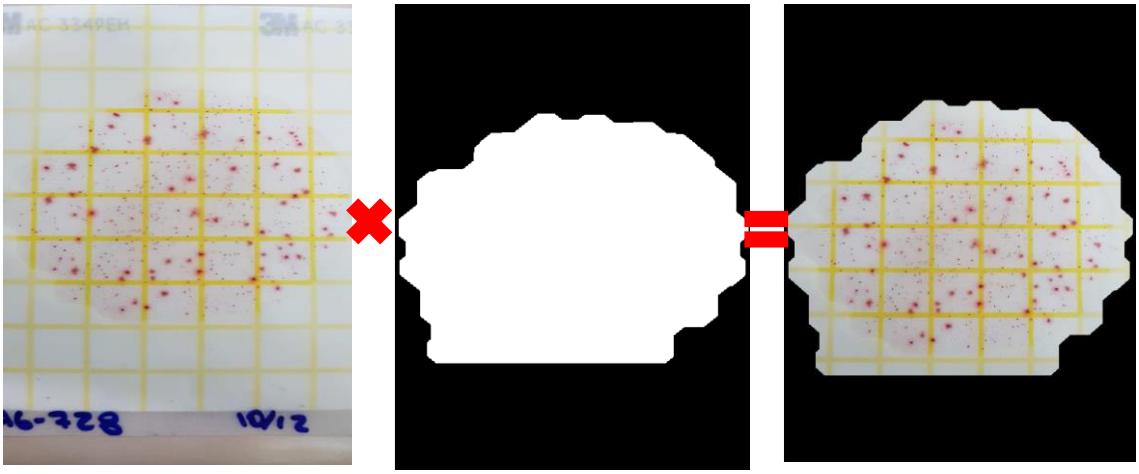
Segmento de código 2.14: Código que crea una imagen binaria con los pixeles del borde con valor de 0 y los pixeles del centro con un valor de 1.

Una vez que se obtiene una máscara para segmentar la región de interés, el siguiente paso es obtener la imagen a color segmentada, para ello se multiplica la máscara Figura 2.12 por la imagen original. La resolución se detalla en el Segmento de código 2.15.

```
%IMAGEN DE INTERES SEGMENTADA%
%Se multiplica la imagen a color original por la máscara
imagen_segmentada_color = app.he.*uint8(mascara);
figure
imshow(imagen_segmentada_color)
title('imagen segmentada');
```

Segmento de código 2.15: Segmentación de la sección de interés.

En la Figura 2.13 literal c) se observa que se consiguió segmentar la sección de interés al multiplicar el literal b) que representa la máscara, con la imagen original representada en el literal a).



a) b) c)

Figura 2.13 Segmentación a color de la sección de interés literal c).

2.2 CONTEO DE LOS MICROORGANISMOS PRESENTES EN LA IMAGEN DE LA PLACA PETRIFILM

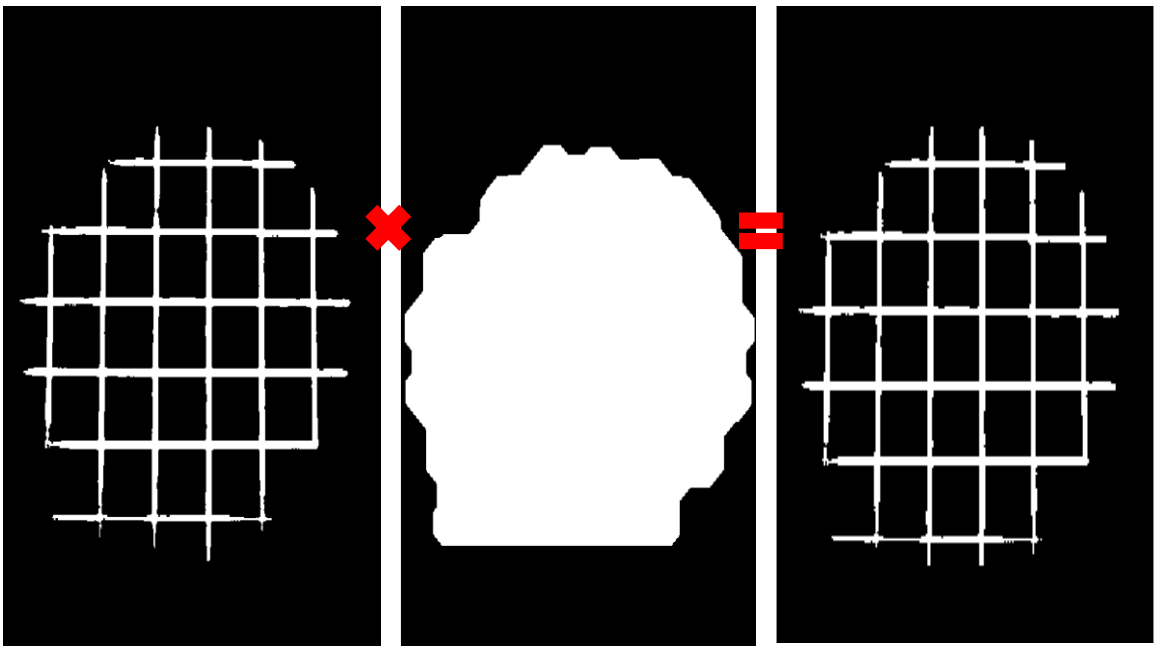
El conteo de los microorganismos presentes en la placa petrifilm se lo realiza en dos partes, esto ocurre porque la sección de la imagen que contiene las líneas (cuadricula de líneas amarillas), donde también están presentes microorganismos, es diferente a la tonalidad del resto de la placa, entonces primero se empieza por procesar solo la sección de la imagen que contiene las líneas, para después procesar la sección de la imagen que no contiene las líneas.

Para el procesamiento de la sección de la imagen que contiene las líneas se parte del clúster 2, Figura 2.10, ya que esta contiene la parte segmentada solo de las líneas. A continuación, se multiplica la máscara Figura 2.13 literal b) con el clúster 2, esto asegura que el clúster 2 de líneas tenga las mismas dimensiones de contorno que la máscara de segmentación. En el Segmento de código 2.16 se detalla el procedimiento antes mencionado.

```
%Se multiplica la máscara con el cluster 2 de lineas, esto asegura
que el cluster 2 de lineas tenga las tenga las mismas dimensiones del
máscar
maska2=maska2.*mascara;
figure
imshow(maska2)
title('CLUSTER DE LINEAS')
```

Segmento de código 2.16: Clúster 2 nuevo de líneas.

En la Figura 2.14 literal c) se visualiza el nuevo clúster 2 de líneas que servirá como máscara.



a) b) c)

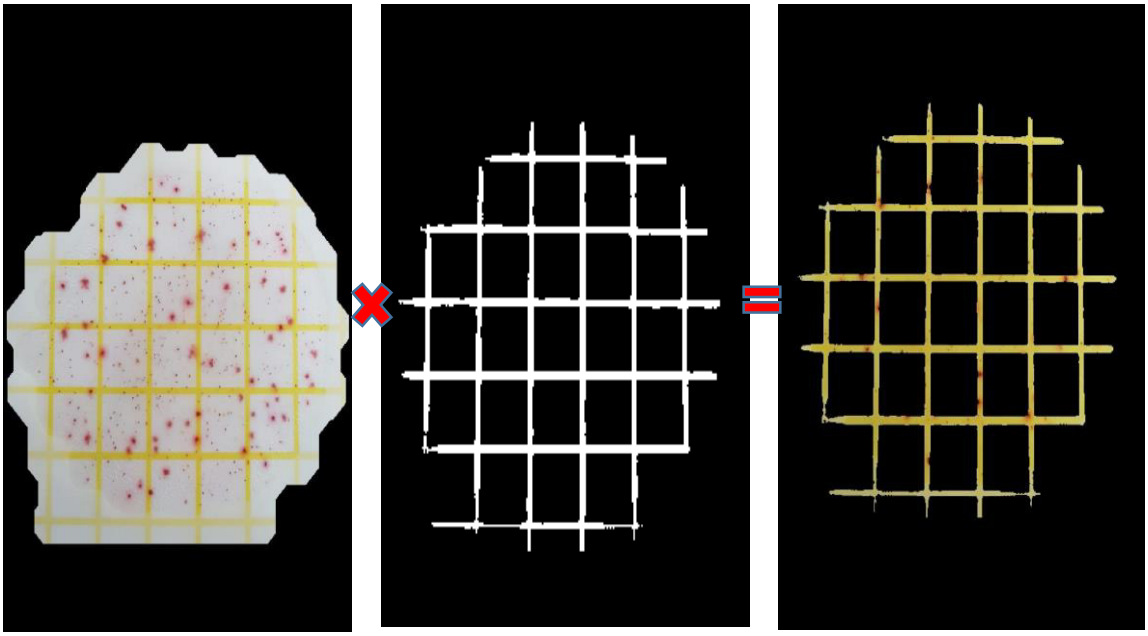
Figura 2.14 Máscara nueva de líneas

La máscara de la Figura 2.14 literal c) se usa para segmentar la sección de las líneas, para conseguir este propósito se multiplica la imagen segmentada Figura 2.13 literal c) por la máscara de líneas Figura 2.14 literal c). La resolución se detalla en el Segmento de código 2.17.

```
%Se Multiplica la máscara de lineas con la imagen a color segmentada
para obtener la imagen segmentada solo de lineas.
Imagen_lineas= imagen_segmentada_color.*uint8(maska2);
figure
imshow(Imagen_lineas)
title('IMAGEN SEGMENTADA SOLO CON LINEAS')
```

Segmento de código 2.17: Imagen de líneas.

En la Figura 2.15 literal c) se observa que se consiguió segmentar solo las líneas, dicha sección cuenta con tonalidades similares, lo que permite mayor facilidad en el análisis.



a) b) c)

Figura 2.15 Imagen de líneas

La imagen a color de líneas de la Figura 2.15 se transforma a escala de grises mediante el comando `rgb2gray`, donde el argumento de entrada a la función es la imagen de líneas `(Imagen_lineas)`.

El procedimiento se detalla en el Segmento de código 2.18.

```
%Imagen de lineas en escala de Grises
Imagen_lineas_gris=rgb2gray(Imagen_lineas);
figure
imshow(Imagen_lineas_gris)
title('Imagen en escala de Grises')
```

Segmento de código 2.18: Imagen de líneas en escala de gris

En la Figura 2.16 se observa la imagen de líneas en escala de grises lo que facilitará la manipulación para mejorar el contraste.

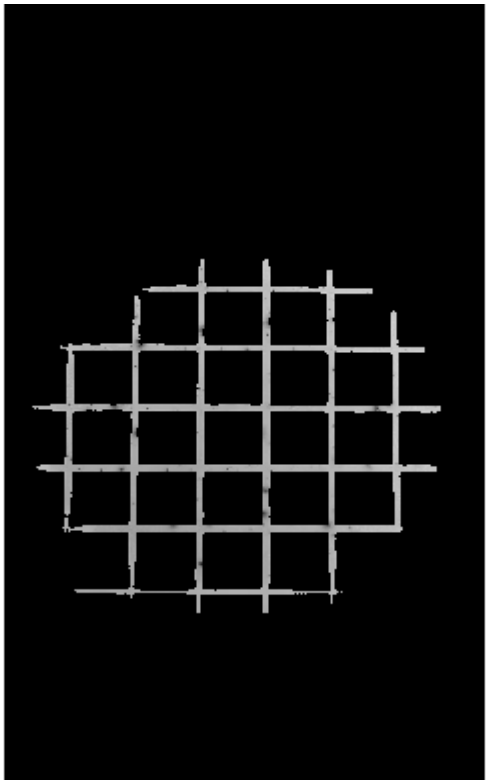


Figura 2.16 Imagen de líneas en escala de gris

El objetivo es no perder ningún detalle en la imagen de la Figura 2.16, en especial las secciones donde se encuentran los microorganismos que son los diminutos puntos de color gris oscuro.

Para mejorar el contraste de la imagen en escala de grises y resaltar los microorganismos con poca visibilidad, se aplica en la imagen el método de ecualización adaptativa de histograma. Si se visualiza el histograma de la Figura 2.17 (Antes de aplicar el método de ecualización adaptativa) se observa que todos los pixeles están agrupados en una determinada zona, esto origina que la imagen tenga poco contraste.

La obtención del histograma se realiza con la función `imhist`, el procedimiento se detalla en el Segmento de código 2.19.

```
%HISTOGRAMA DE IMAGEN DE LINEAS EN ESCALA DE GRISES
figure
imhist(Imagen_lineas_gris);
title('HISTOGRAMA DE IMAGEN DE LINEAS')
```

Segmento de código 2.19: La Creación del Histograma de la imagen de líneas en escala de gris

En la Figura 2.17 se visualiza el histograma de la imagen a grises, los pixeles se agrupan en su mayoría en el intervalo de 160 a 200 esto origina que la imagen tenga poco contraste.

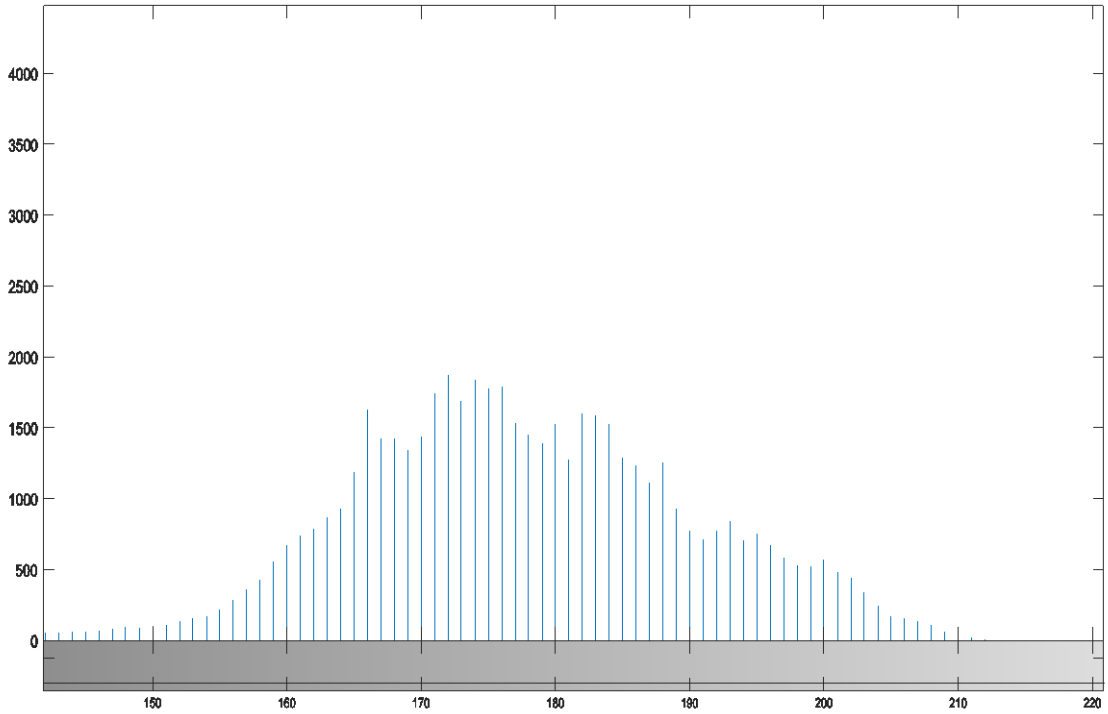


Figura 2.17 Histograma de Imagen de líneas en escala de gris

Para aplicar la ecualización adaptativa de histograma se requiere utilizar la función de Matlab `adapthisteq` que tiene como objetivo expandir y aplanar el histograma.

En el Segmento de código 2.20 se detalla la aplicación de la ecualización adaptativa de histograma.

Sintaxis adapthisteq:

```
J = adapthisteq(I,Name,Value)
```

Descripción:

`J = adapthisteq(I,Name,Value)` mejora el contraste de la imagen `I` en escala de grises mediante la transformación de los valores mediante la ecualización de histograma adaptativo.

```
%Se utiliza la ecualización adaptativa de Histograma para mejorar el
%contraste.
claheI = adapthisteq(I1,'NumTiles',[10
10],'clipLimit',0.02,'Distribution','uniform');
claheI = imadjust(claheI);
figure
imshow(claheI);
title('Imagen con Ecualización Adaptativa de Histograma')
```

Segmento de código 2.20: Ecualización Adaptativa de Histograma de la imagen de líneas en escala de gris

La función `adapthisteq` tiene los siguientes argumentos de entrada que se describen a continuación:

- *'NumTiles':* Indica que se va a dividir la imagen en mosaicos, en cada mosaico se aplica la ecualización adaptativa de histograma, mediante la experimentación se determinó que para la presente aplicación se divide la imagen en mosaicos de 10 por 10 pixeles.
- *'clipLimit':* Es un valor que representa un factor de contraste, que se encuentra en el rango que va desde 0 hasta el 1, este valor impide la sobresaturación de la imagen, para la presente aplicación se utilizó el valor de 0.2.
- *'Distribution':* Indica la forma en la que se representa el histograma. Mediante la experimentación, se determinó que la distribución de histograma que mejor se ajusta en la presente aplicación es la distribución uniforme 'uniform'.

La Figura 2.18 es el resultado de aplicar la técnica de ecualización adaptativa de histograma, se visualiza que hubo una mejora significativa en el contraste de la imagen, lo que permite que los detalles poco legibles se visualicen de una mejor manera.

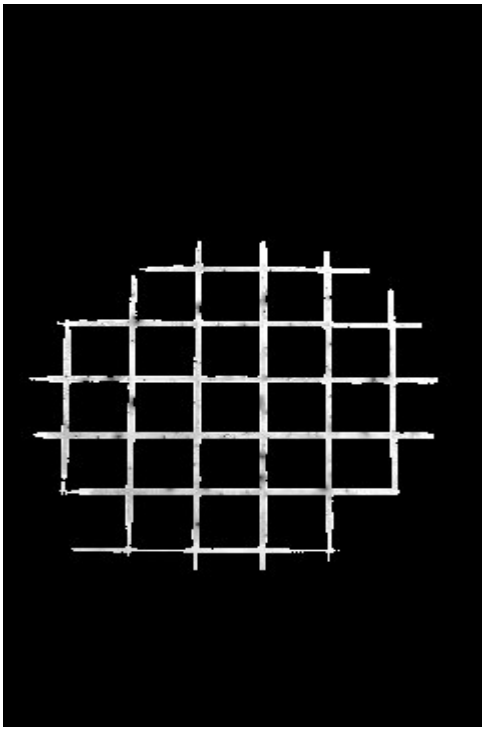


Figura 2.18 Imagen de líneas en escala de gris con ecualización adaptativa de histograma

En el Segmento de código 2.21 se recrea el histograma una vez aplicado la técnica de ecualización adaptiva de histograma.

```
%Histograma de imagen  de lineas en escala de grises aplicando
latecnica de ecualizacion adaptativa de histograma
figure
imhist(claheI);
title('Histograma con Ecualización Adaptativa ')
```

Segmento de código 2.21: Histograma de la técnica de Ecualización Adaptativa de Histograma de la imagen de líneas en escala de gris

En la Figura 2.19 se visualiza el histograma cuando ya se aplicado la técnica de ecualización adaptativa de histograma, lo que permitió aplanar el histograma y redistribuir los valores de los pixeles de una manera más uniforme

Ahora los pixeles se centralizan en el intervalo de 180 a 256, esto originó una mejora en el contraste de la imagen.

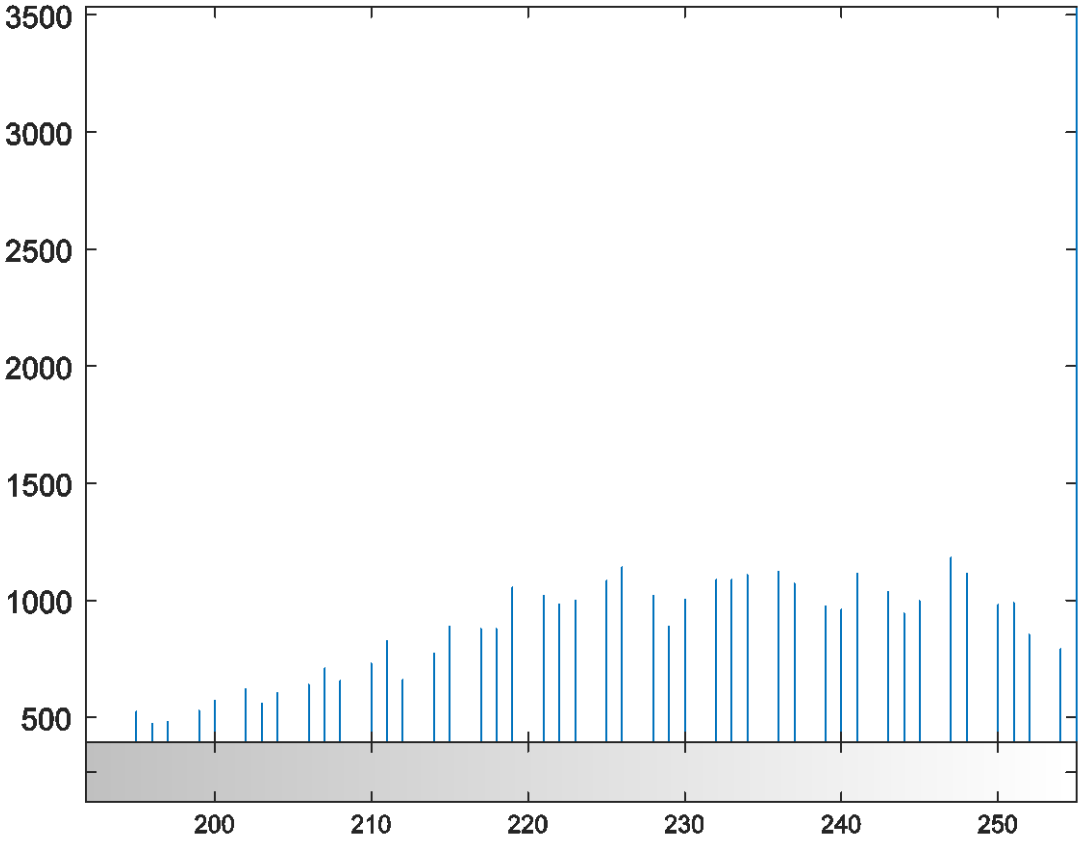


Figura 2.19 Histograma de Ecualización Adaptativa de la imagen de líneas en escala de gris

A continuación, se requiere umbralizar o binarizar la imagen de la Figura 2.18, existen diversos métodos para dicho propósito, pero a partir de la experimentación el que mejor resultado mostró fue el método que a partir del histograma se encuentra un valor empírico de umbralización, en base a las diversas pruebas que se realizó el valor más adecuado de umbralización fue el de 145.

En el Segmento de código 2.22 se detalla el proceso de umbralización de la imagen solo con líneas, se programó un código para que valores menores a 145 tomen el valor de 0, y valores mayores o iguales a 145 tomen el valor de 1, de esta forma se consigue el objetivo de binarizar la imagen sin perder los pequeños microorganismos que se encuentran en las líneas de las placas petrifilm.

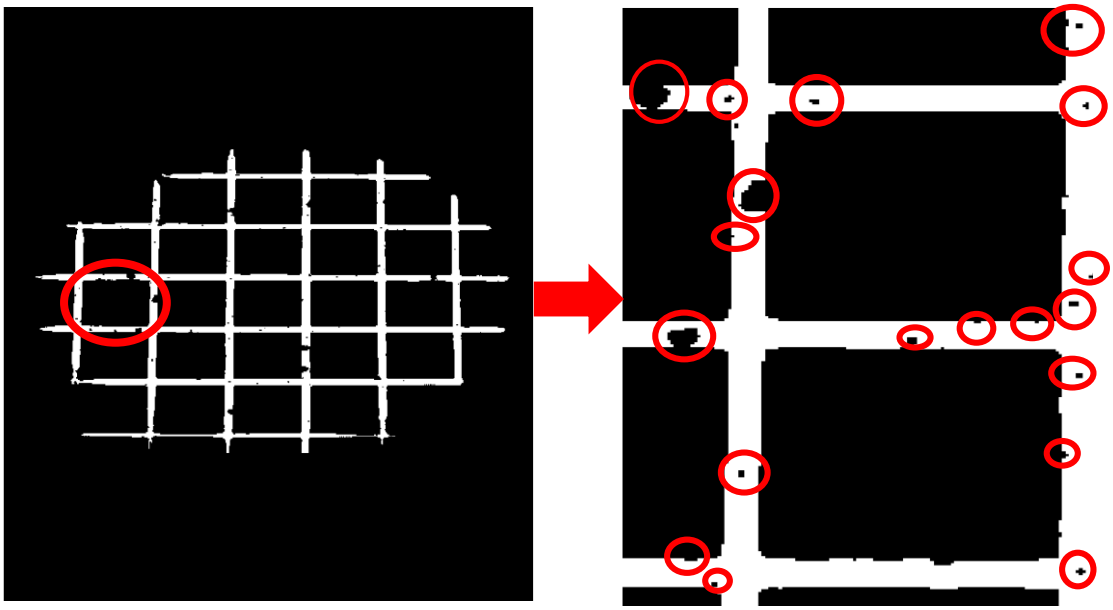
```
%De una manera empírica se encontró el valor de Umbralización más
adecuado que fue de 145, y mediante un pequeño código se binariza la
imagen que solo contiene lineas.
[f,c]=size(claheI);
```

```
for i=1:f
    for k=1:c
        if claheI(i,k)<=145
           claheI(i,k)=0;
        else
          claheI(i,k)=255;
        end
    end
end

figure
imshow(claheI)
title('Imagen de lineas Binarizada')
figure
```

Segmento de código 2.22: Umbralización de la imagen de líneas en escala de gris

La Figura 2.20 representa la imagen binaria de las líneas en la placa petrifilm, el literal b) es una ampliación de una pequeña sección de la imagen del literal a), los círculos rojos en la imagen del literal b) ayudan a identificar a los microorganismos presentes en las líneas.



a) **b)**

Figura 2.20 Literal a) Imagen binaria de las líneas, literal b) Ampliación de un segmento de la imagen, los círculos rojos identifican los microorganismos presentes en la imagen de líneas

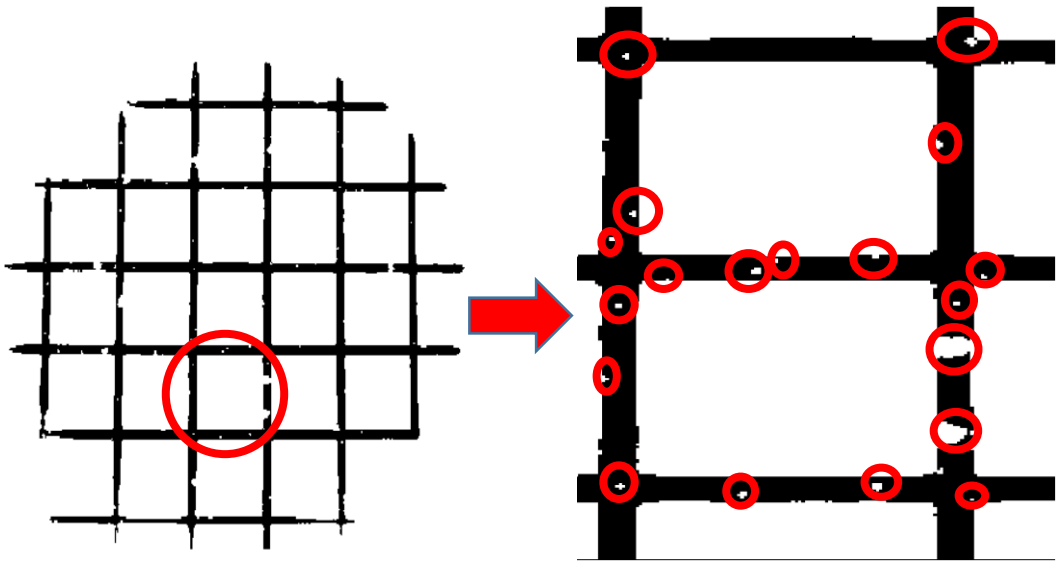
En la Figura 2.20 literal b) se observa que los microorganismos están representados por el color negro (valor de 0) y las líneas están representadas por el color blanco (valor de 1).

Para aislar los microorganismos que se encuentran en las líneas en su totalidad, primero se calcula el negativo de la Figura 2.20. En el segmento de código 2.23 se detalla el procedimiento.

```
%NEGATIVO DE LA FIGURA 2.17
m=~claheI;
figure
imshow(m)
title('NEGATIVO DE LA IMAGEN')
```

Segmento de código 2.23: Negativo de la imagen de la Figura 2.17.

En la Figura 2.21 se visualiza el resultado del segmento de código 2.23, los microorganismos ahora tienen el valor de 1 (color blanco) y las líneas ahora tienen el color negro (valor 0).



a) b)

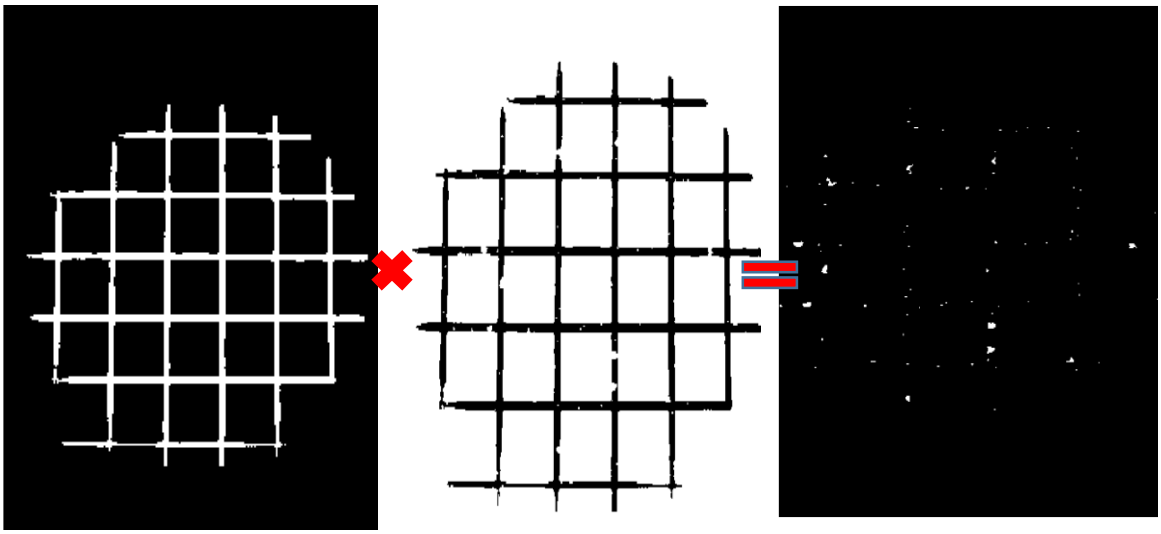
Figura 2.21 Negativo de la imagen de la Figura 2.20.

A continuación, si se multiplica pixel a pixel la imagen de la Figura 2.21 con la imagen del clúster 2 de líneas Figura 2.14 literal c) (Máscara de líneas), se consigue aislar por completo los microorganismos que se encuentran en las líneas. El procedimiento se detalla en el segmento de código 2.24.

```
%Aislar los microorganismos de la lineas
heI_lineas=maska2.*m;
figure
imshow(heI_lineas)
title('MICROORGANISMOS EN LAS LINEAS')
```

Segmento de código 2.24: Microorganismos presentes en las líneas de las placas petrifilm aislados.

En la Figura 2.22 literal c) se observa que se logró aislar por completo los microorganismos presentes en las líneas.



a) b) c)

Figura 2.22 Microorganismos presentes en las líneas de las placas petrifilm aislados.

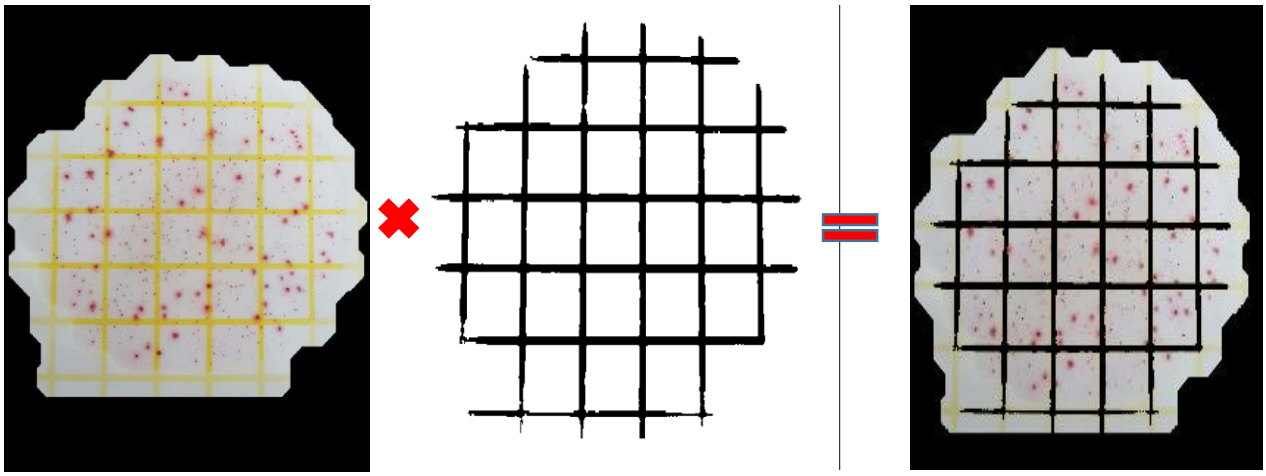
Una vez que se aisló los microorganismos presentes en las líneas, a continuación, se procede a aislar los microorganismos restantes que se encuentran en la placa petrifilm

Para aislar los microorganismos restantes (microorganismos que no se encuentran en las líneas), se utiliza una máscara que permita desaparecer las líneas y a su vez los microrganismos presentes en las líneas, ya que estos fueron aislados anteriormente. Para conseguir este objetivo se vuelve a utilizar la máscara del clúster 2 de líneas (Figura 2.10 Máscara de líneas), pero esta vez se calcula el negativo de dicha imagen, con el objetivo de multiplicar este resultado con la imagen Figura 2.13 (Segmentación a color de la sección de interés). Cabe recalcar que la máscara de la Figura 2.10 contiene las líneas, pero sin los microorganismos.

El procedimiento antes mencionado se detalla en el segmento de código 2.25

```
%%CREAMOS UNA MASCARA PARA ELIMINAR LAS LINEAS DE LA IMAGEN DE LA
FIGURA 2.9
%Se vuelve a utilizar la imagen del clúster 2 de líneas (Figura 2.10
Máscara de líneas),
%pero esta vez se calcula el negativo de dicha imagen
mask22_negativo=1-maska2;% Imagen en negativo
Imagen_sin_lineas= imagen_segmentada_color.*uint8(mask22_negativo);
%Multiplicación entre la mascara para eliminar las lineas y la imagen a
color.
figure
imshow(Imagen_sin_lineas)
title('IMAGEN SEGMENTADA A COLOR SIN LINEAS');
```

Segmento de código 2.25: Imagen segmentada a color sin líneas.



a) **b)** **c)**

Figura 2.22 Imagen a color sin líneas literal c).

En la imagen de la Figura 2.22 literal c) se visualiza que ya no se cuenta con las líneas ni con los microorganismos presentes en las líneas.

A continuación, se realiza la transformación de la imagen de la Figura 2.22 literal c) a una imagen en escala de grises. El procedimiento se detalla en el segmento de código 2.26.

```
%Se transforma a imagen en escala de grises
Imagen_sin_lineas_gris=rgb2gray(Imagen_sin_lineas);
```

```
figure
imshow(Imagen_sin_lineas_gris)
title('IMAGEN EN ESCALA DE GRISES')
```

Segmento de código 2.26: Imagen a escala de grises sin líneas

En la Figura 2.23 se visualiza la imagen sin líneas a escala de grises.

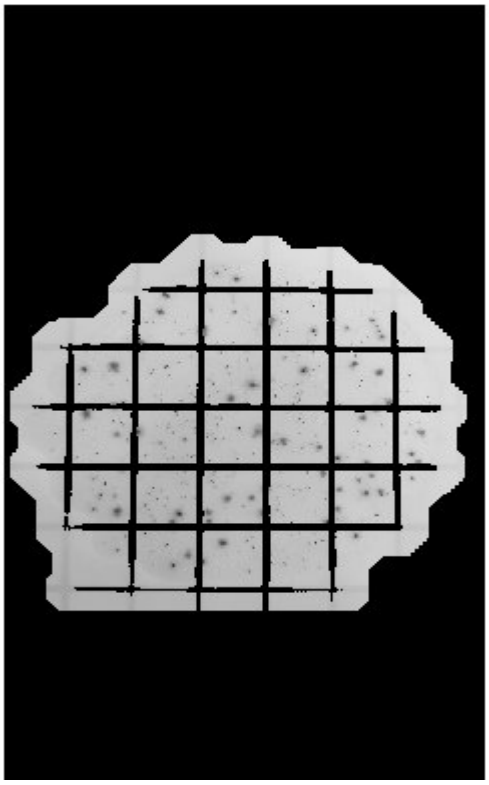


Figura 2.23 Imagen a escala de grises sin líneas

Para mejorar el contraste, en la Figura 2.23 se aplica la técnica de ecualización adaptativa de histograma mediante la función `adapthisteq`. En el segmento de código 2.27 se detalla el procedimiento en la aplicación de la ecualización adaptativa de histograma.

```
%Se aplica Ecualización Adaptativa de histograma
claheI_1 =adapthisteq(Imagen_sin_lineas_gris,'NumTiles',[10
2],'clipLimit',0.03,'Distribution','uniform');
claheI =imadjust(claheI_1);
figure
imshow(claheI_1)
title('IMAGEN EN ESCALA DE GRISES MEJORADA CON ECUALIZACIÓN DE
HISTOGRAMA')
```

Segmento de código 2.27: Ecualización adaptativa de histograma

Los parámetros de entrada a la función `adapthisteq` presentan diferentes valores a los aplicados a la imagen de líneas de la Figura 2.16 (Imagen de líneas en escala de gris aplicando ecualización adaptativa de histograma).

Mediante la experimentación se observó que los valores de entrada que mejor funcionan para la aplicación son:

- *'NumTiles'* con un mosaico de [10 2], en 'clipLimit' se emplea un valor de 0.03, y también se utiliza una distribución uniforme. En la Figura 2.24 se observa el resultado de aplicar la ecualización adaptativa de histograma.

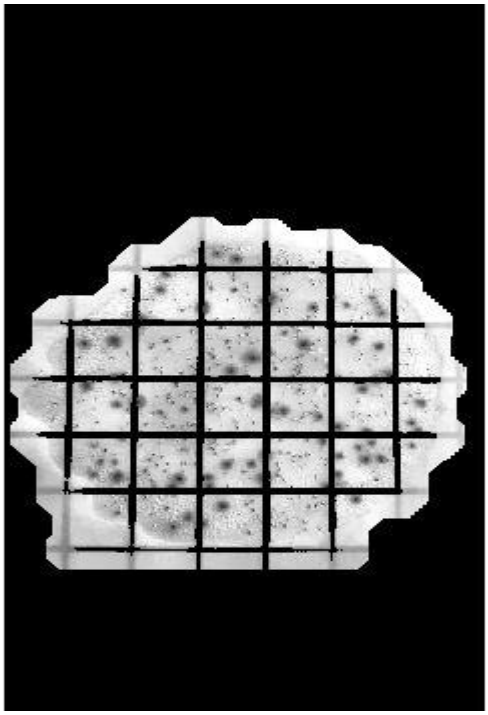


Figura 2.24 Imagen aplicada ecualización adaptativa de histograma

Para umbralizar o binarizar la imagen de la Figura 2.24 se utiliza la función `imbinarize`, el procedimiento se detalla en el segmento de código 2.27.

Sintaxis imbinarize:

```
BW = imbinarize(I,'adaptive',Name,Value)
```

Descripción:

`BW = imbinarize(I,'adaptive',Name,Value)` crea una imagen binaria a partir de la imagen utilizando pares nombre-valor para controlar aspectos del umbral adaptativo.I

```
%IMAGEN BINARIZADA SIN LOS MICRORGANISMOS EN LAS LINEAS %
a1_binaria=
imbinarize(claheI_1,'adaptive','ForegroundPolarity','dark','Sensitivi
ty',0.4); %Imagen binaria
figure
imshow(a1_binaria)
title('IMAGEN BINARIZADA SIN LAS LINEAS')
```

Segmento de código 2.28: Binarizar imagen de la Figura 2.23.

La función `imbinarize` tiene los siguientes parámetros de entrada y salida:

- *'adaptive':* Calcula el umbral solo en pequeñas secciones de la imagen o en pequeños vecindarios alrededor de cada pixel, escoge el valor de umbral de acuerdo a la intensidad media del sector analizado.
- *'ForegroundPolarity':* Clasifica la imagen en dos planos, un plano perteneciente al fondo, y otro plano denominado primer plano.
- *'dark'*: Muestra que el fondo es más brillante que el primer plano
- *'Sensitivity' :* Es el umbral de sensibilidad que controla que pixeles pertenecen al primer plano, el rango de sensibilidad va desde 0 hasta 1, un valor alto originaría que pixeles del fondo pasen a ser pixeles del primer plano, lo que podría ocasionar graves errores. Mediante la experimentación se concluyó que el valor más adecuado para la presente aplicación es el valor de 0.4.
- a1_binaria: Representa la imagen binaria de salida de la función `imbinarize`.

En la Figura 2.25 se observa la imagen ya binarizada, los puntos negros equivalen a los microorganismos, y toman el valor de 0, mientras que el fondo está representado por el color blanco y toma el valor 1.

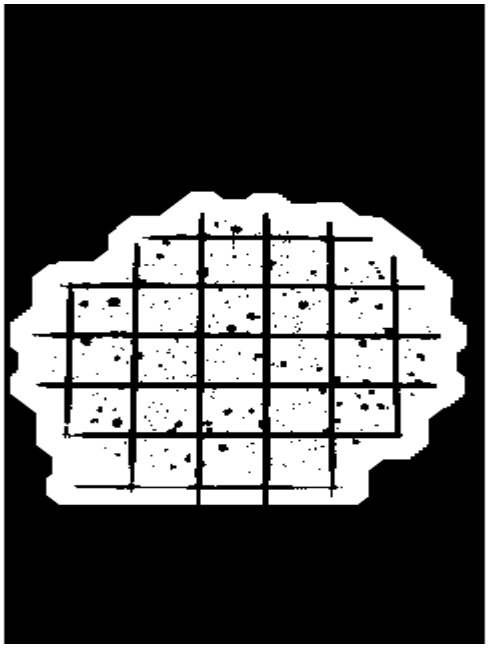
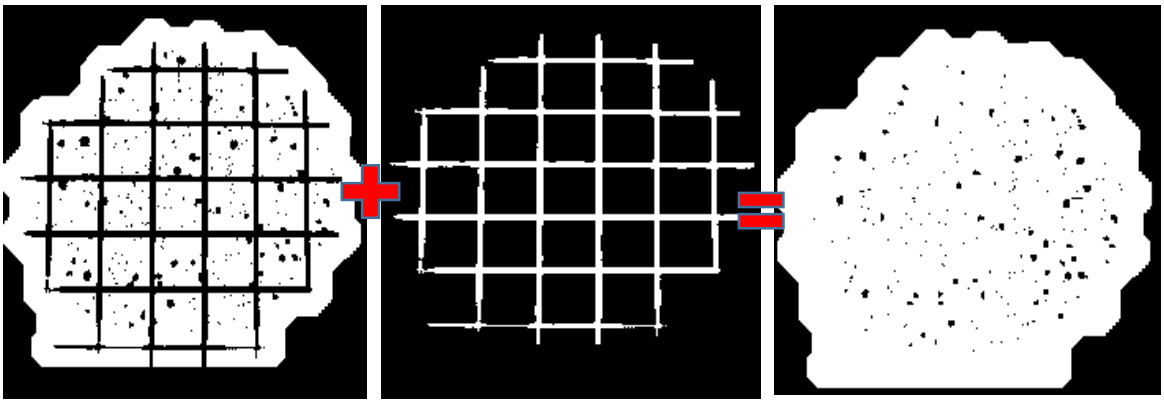


Figura 2.25 Imagen binarizada

En la imagen de la Figura 2.25 se visualiza que contiene las líneas de color negro, para homogenizar la imagen con todo el fondo blanco, se suma el clúster 2 de líneas de la Figura 2.10 (Máscara de líneas) con la imagen de la Figura 2.25.

En la Figura 2.25 literal c) se observa el resultado de la suma de las imágenes mencionadas.



a). **b).** **c).**

Figura 2.25 Suma de imagen a) y la imagen b), para homogenizar el fondo blanco, que da como resultado la imagen c)

El segmento de código 2.29 representa el procedimiento de la suma de imágenes de la Figura 2.25 del literal a) y literal b).

```
figure%Imagen binaria sin microorganismos presentes en las lineas
d1_sin_microorganismos_lineas=a1_binaria+maska2;
imshow(d1_sin_microorganismos_lineas);
title('IMAGEN BINARIA SIN LOS MICROORGANISMOS DE LAS LINEAS')
```

Segmento de código 2.29: Suma de imágenes para homogenizar el fondo blanco

Para cambiar el color de los microorganismos de negro a blanco, y el fondo de la imagen de blanco a color negro, en la imagen de la Figura 2.25 literal c) se toma el negativo, lo cual da como resultado la imagen de la Figura 2.26.

Figura 2.26 Negativo de la Figura 2.25 literal c)

En el Segmento de código 2.30 se detalla el procedimiento para calcular el negativo de la imagen Figura 2.25 literal c).

```
%SE TOMA EL NEGATIVO DE LA IMAGEN FIGURA 2.28 LIERAL C)
cc2_negativo=1- d1_sin_microorganismos_lineas;
figure
imshow(cc2_negativo);
title('NEGATIVO DE IMAGEN BINARIA SIN MICROORGANISMOS DE LAS LINEAS')
```

Segmento de código 2.30: Negativo de la imagen binaria Figura 2.25 literal c).

En la Figura 2.26 no se encuentran presentes los microorganismos de las líneas, pero en el segmento de código 2.23 y la Figura 2.22 se observa que ya se logró extraer a los microorganismos presentes en las líneas. Para unir a todos los microorganismos que se

encuentran en la placa petrifilm, se suma la Figura 2.26 y la Figura 2.22 literal c) (Microorganismos presentes en las líneas), el resultado de la suma origina una imagen con todos los microorganismos presentes en la placa petrifilm.

En el Segmento de código 2.31 se detalla el procedimiento de la suma de las imágenes antes mencionado.

```
%%OBTENER EL CONTEO TOTAL DE TODOS LOS MICRORGANISMOS BINARIZADOS
% SE SUMA LA IMAGEN CON LINEAS Y LA IMAGEN CON MICROORGANISMOS
cc3_total= cc2_negativo + heI_lineas;
figure
imshow(cc3_total,8);
title('Imagen completa con todos los microorganismos')
total=bwconncomp(cc3_total);
```

Segmento de código 2.31: Total de los microorganismos presentes en la imagen petrifilm

En la Figura 2.27 literal c) se observa el resultado de la suma de las imágenes a) y b), como consecuencia se consigue el aislamiento de todos los microorganismos presentes en la imagen de la placa petrifilm, donde el literal b) son los microorganismos presentes en las líneas, y el literal a) son los microorganismos que se encuentran en el resto de la placa petrifilm excepto las líneas.

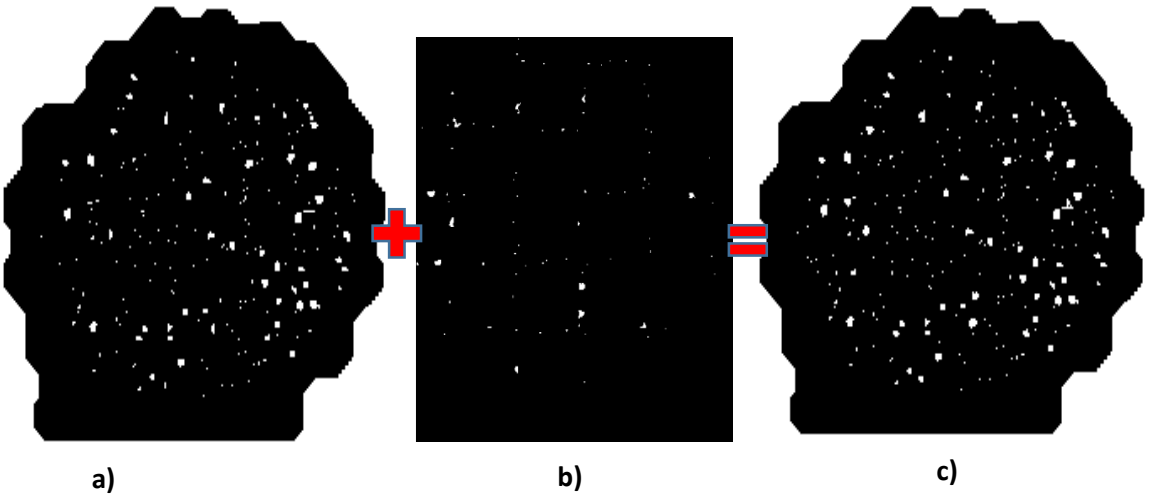


Figura 2.27 Suma de la imagen a) y b) para obtener el total de microorganismos presentes en la imagen petrifilm, literal c)

Una vez que se obtienen todos los microorganismos aislados Figura 2.27 literal c), se facilita el objetivo final que es el conteo de los mismos, para ello se utiliza la función de Matlab `bwconncomp` como se detalla en el Segmento de código 2.31.

La función `bwconncomp` se basa en el algoritmo de etiquetado de labelling que se encarga de encontrar las componentes conexas u objetos presentes en la imagen binaria, para ello en los argumentos de entrada de la función se especifica qué tipo de conexión de pixeles se va a utilizar, en la presente aplicación se utiliza la conexión-8 que es la que mejores resultados consigue.

Los argumentos de entrada de la función bwconncomp son la imagen de la Figura 2.27 literal c), seguido por el tipo de conectividad de pixeles que se usa, la salida de la función es el número de componentes conexas u objetos que s e encontró en la imagen.

En la Figura 2.28 se visualiza los datos de salida entregados por la función `bwconncomp,` la salida es de tipo `struct,` donde `NumObjects` es el número de objetos presentes en la imagen.

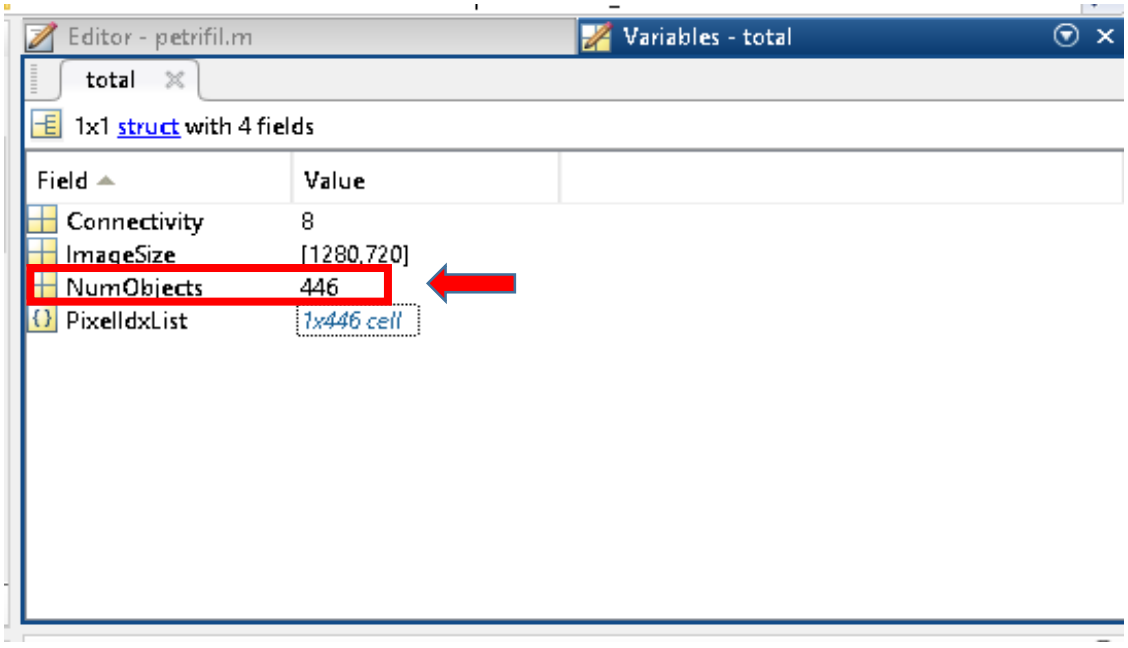


Figura 2.28 Resultados de la función `bwconncomp` de Matlab que permite encontrar el número de objetos presentes en una imagen binaria

2.3 DESCRIPTORES GEOMÉTRICOS

En la sección 2.2 se realizó el conteo total de los microorganismos presentes en la imagen, en la presente sección se realizará un análisis de los microorganismos de mayor tamaño, se utilizará detectores de bordes y geometrías circulares, así como también funciones de Matlab que permitan la identificación de sus centros, áreas y perímetros.

Se necesita aislar los microorganismos de mayor tamaño, para conseguir este objetivo se realizó un estudio y en base a la experimentación se concluyó que los microorganismos claramente visibles en la placa petrifilm tienen un radio mayor a 20 pixeles, se utilizó la función bwareaopen que elimina los microorganismos que se encuentran menores al rango de 20 pixeles, además se programa para que el fondo de la imagen este completamente oscuro, esto facilita la manipulación de los microorganismos. El desarrollo del programa se detalla en la Segmento de código 2.32.

```
%%OBTENER EL CONTEO DE LOS MICRORGANISMOS MAS GRANDES
BW11_MICROOGANISMOS_GRANDES=bwareaopen(cc3_total,20); %%La funcion
bwareopen aisla los microorganismos menores a 20 pixels
BW1_MICROOGANISMOS_GRANDES = BW11_MICROOGANISMOS_GRANDES; Se crea una
imagen con las mismas caracteristicas y se llama
BW1_MICROOGANISMOS_GRANDES.
figure
imshow(BW1_MICROOGANISMOS_GRANDES);
title('Imagen con los microorganismos mas Grandes')
numPixels = cellfun(@numel,total.PixelIdxList);
[biggest,idx] = max(numPixels);
BW1_MICROOGANISMOS_GRANDES(total.PixelIdxList{idx})=0;
figure
imshow(BW1_MICROOGANISMOS_GRANDES)
title('CONTEO DE LOS MICROORGANISMOS DE MAYOR TAMAÑO EN LA IMÁGENES
')
```

Segmento de código 2.32: Se los microorganismos de mayor tamaño

En la Figura 2.29 se visualiza que se consiguió separar los microorganismos de mayor tamaño.

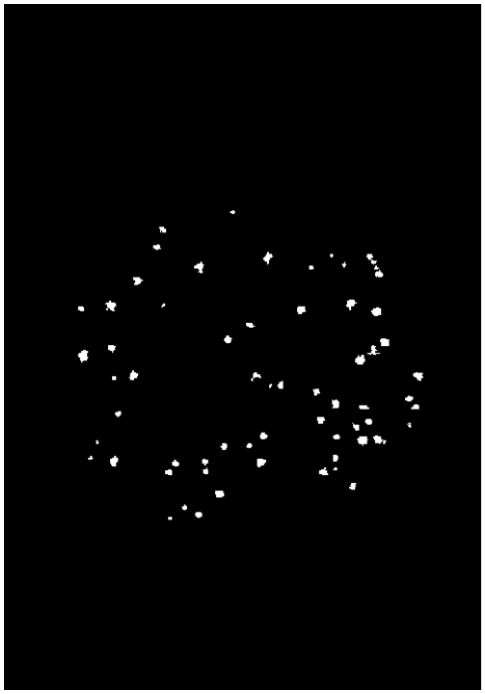


Figura 2.29 Se aísla los microorganismos de mayor tamaño

Para analizar si existen microorganismos solapados se utiliza la transformada de Hough, los detalles de funcionamiento del algoritmo se describen en la sección 1.3.7, el algoritmo se caracteriza por encontrar las formas circulares de los objetos presentes en una imagen, el algoritmo calcula los radios y los centros de cada objeto, para después en base a la ecuación de la circunferencia trazar el perímetro del posible círculo.

Una ventaja de utilizar la transformada de hough es que podría existir la posibilidad que dos microorganismos estén semi-solapados, entonces la transformada de hough detecta y calcula el centro y radio de cada microorganismo, y a pesar que se encuentren los microorganismos semi-solapados la transformada entiende que son diferentes objetos.

En la presente aplicación en base a todas las imágenes utilizadas no se encontró el caso en el que se presenten microrganismos semi-solapados.

Para la aplicación de la trasformada de hough se utilizó la función `imfindcircles`, el algoritmo posee diferentes variables de entrada, que de acuerdo a ciertas diligencias pueden ser útiles o no, en base a la experimentación se determinó los parámetros y valores más adecuados. En el Segmento de código 2.33 se detalla la aplicación de la transformada de hough.

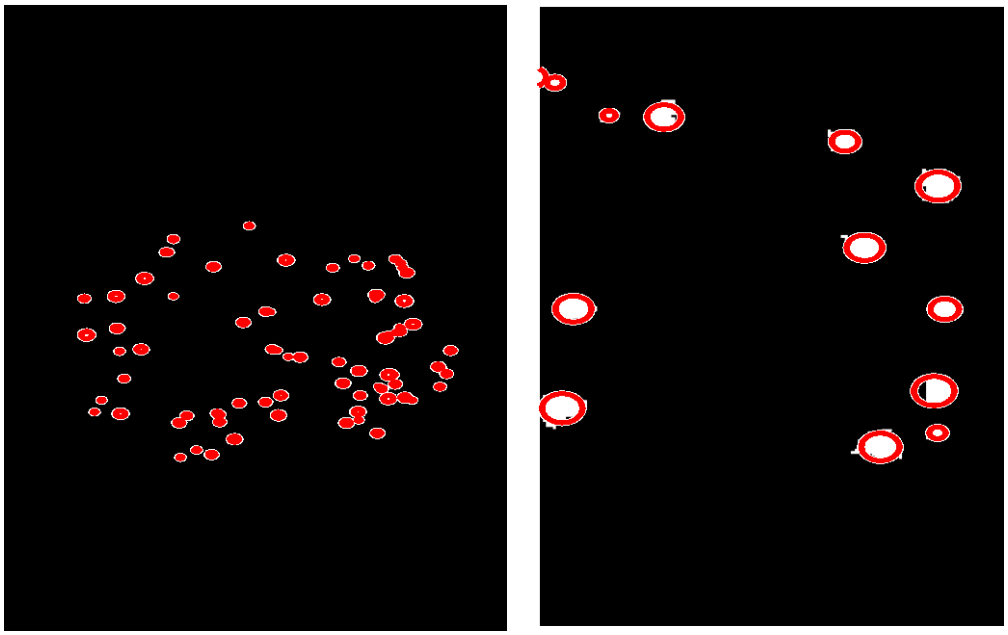
```
%Transformada de Houg.
[centers,radii] = imfindcircles(BW1_MICROOGANISMOS_GRANDES,[20
40],'ObjectPolarity','bright', ...
    'Sensitivity',0.95)
figure
imshow(BW1_MICROOGANISMOS_GRANDES) %Mediante la funcion viscircles se
dibuja los circulos a partir de los centros y radios detectados por
la transformada de houg,
h=viscircles(centers,radii,'Color','r');
%
```

Segmento de código 2.33: Transformada de houg

La transformada de hough tiene las siguientes variables de entrada:

- Se especifica un rango de radios donde la transformada de hough va a buscar las formas circulares, se determinó el rango [20 40].
- *'ObjectPolarity':* Indica si el fondo es más claro u oscuro que los objetos que se desean detectar, en nuestro caso el fondo es más oscuro que los objetos, por este motivo se elige 'bright'
- La sensibilidad es un valor de búsqueda que indica la precisión con la que va a detectar los círculos, se encuentran en el rango de 0 al 1, para la presente aplicación se utilizó 0.95. Un valor alto de sensibilidad podría ocasionar que se encuentren falsos positivos, es decir que se detecten círculos donde no los hay.
- Los valores de salida de la función son el radio y el centro de cada circulo detectado, los centros se representan en coordenadas cartesianas
- La función que permite dibujar los círculos es `viscircles`.
- Los valores de entrada de la función `viscircles` son los radios y los centros de cada círculo detectado, además del color con el que se dibuja los perímetros, en la presente aplicación se utilizó el color rojo.

En la Figura 2.30 litera a) se observa los resultados de la transformada de hough, el literal b) es la misma imagen, pero amplificada para una mejor visualización de los microorganismos. La transformada de hough se encarga de dibujar los perímetros de las circunferencias.



a) b)

Figura 2.30 Transformada de Hough a) Imagen aplicada la transformada de Hough b) Imagen amplificada para mejor visualización.

Para encontrar el perímetro de cada microorganismo primero se utiliza la función `bwboundaries`, como salida de dicha función se obtiene la variable (B) que contiene las coordenadas cartesianas de los limites exteriores de cada objeto.

Con las coordenadas cartesianas de los bordes que se consiguen como resultado de función `bwboundaries`, se procede a utilizar la ecuación matemática para el cálculo de la longitud de una curva. Mediante la unión de cada punto o coordenada cartesiana (mientras más puntos más precisión en el resultado del perímetro) se aproxima un cálculo del perímetro de cada microorganismo.

Una vez calculado el perímetro, se procede a calcular el área de cada microorganismo, esto se realiza con la ayuda de la función `regionprops`.

Un parámetro geométrico que se calcula es base al perímetro y el área de un objeto es la circularidad Ecuación 1.26, este parámetro indica que tan redondo es un objeto, un valor de 1 muestra que el objeto es completamente redondo, mientras que valores menores que 1 sugieren que el objeto es menos redondo. La fórmula que representa la circularidad de un objeto es la razón entre el área del objeto y el perímetro al cuadrado.

En el Segmento de código 2.34 se visualiza en detalle el proceso de cálculo de áreas, perímetros y circularidades de los microorganismos de mayor tamaño.

```
% OBTENER EL PERIMTRO , AREA Y LA REDONDES DE LOS MICROOGANISMOS
BW2=bwareaopen(cc3,40);
figure
imshow(BW2);
numPixels = cellfun(@numel,total.PixelIdxList);
[biggest,idx] = max(numPixels);
BW2(total.PixelIdxList{idx})=0;
figure
imshow(BW2)
title('Imagen para determinar el factor de redondez')
CCC=bwconncomp(BW2);
hold on;
%CÁLCULO DE LOS PUNTOS DEL BORDE EXTERNO DE CADA MICROOGANISMO
[B,L,N] = bwboundaries(BW2);
imshow(label2rgb(L,@jet,[.5 .5 .5]))
hold on

for k=1:length(B),
    boundary = B{k};
    plot(boundary(:,2), boundary(:,1),'.r');

end
```

```
% CÁLCULO DEL AREA DE CADA MICROOGANISMO
stats = regionprops(L,'Area','Centroid');
threshold = 0.94;
%recorrer los límites
%calcular una precisión simple del perímetro del objeto.
%recorrer los límites
```

```
% CALCULO DEL PERIMETRO DE CADA MICROORGANISMO
for k = 1:length(B)
    %obtener coordenadas de límite (X, Y) correspondientes a la
etiqueta 'k'
    boundary = B{k};
    %calcular una estimación simple del perímetro del objeto
    delta_sq = diff(boundary).^2;
    perimeter = sum(sqrt(sum(delta_sq,2)));

    %obtener el cálculo del área correspondiente a la etiqueta 'k'
    area = stats(k).Area;

    %CÁLCULO DE LA CIRCULARIDAD DE CADA OBJETO
```

```
    metric = 4*pi*area/perimeter^2;

    % muestra los resultados
    metric_string = sprintf('%2.2f',metric);

   %marcar objetos por encima del umbral con un círculo negro
    if metric > threshold
        centroid = stats(k).Centroid;
        plot(centroid(1),centroid(2),'ko');
    end
    text(boundary(1,2)-
35,boundary(1,1)+13,metric_string,'Color','y','FontSize',10,'FontWeig
ht','bold') ;
end
title(['Metrics closer to 1 indicate that ','the object is
approximately round']);
```

Segmento de código 2.34: Calculo de áreas, perímetros y circularidad de los microorganismos

2.4 INTERFAZ GRÁFICA DISEÑADA EN APP DESIGNER MATLAB

App Designer es un medio interactivo que permite el diseño de interfaces de usuario fáciles de manipular, en la presente sección se detallará el proceso de creación de una interfaz gráfica, con el objetivo de visualizar los resultados arrojados por el programa que realiza el conteo de microorganismos en placas petrifilm.

Para ingresar a la App Designer en el command Window de Matlab se digita el comando appdesigner, lo que nos llevará a la interfaz de diseño que se muestra en la Figura 2.31.

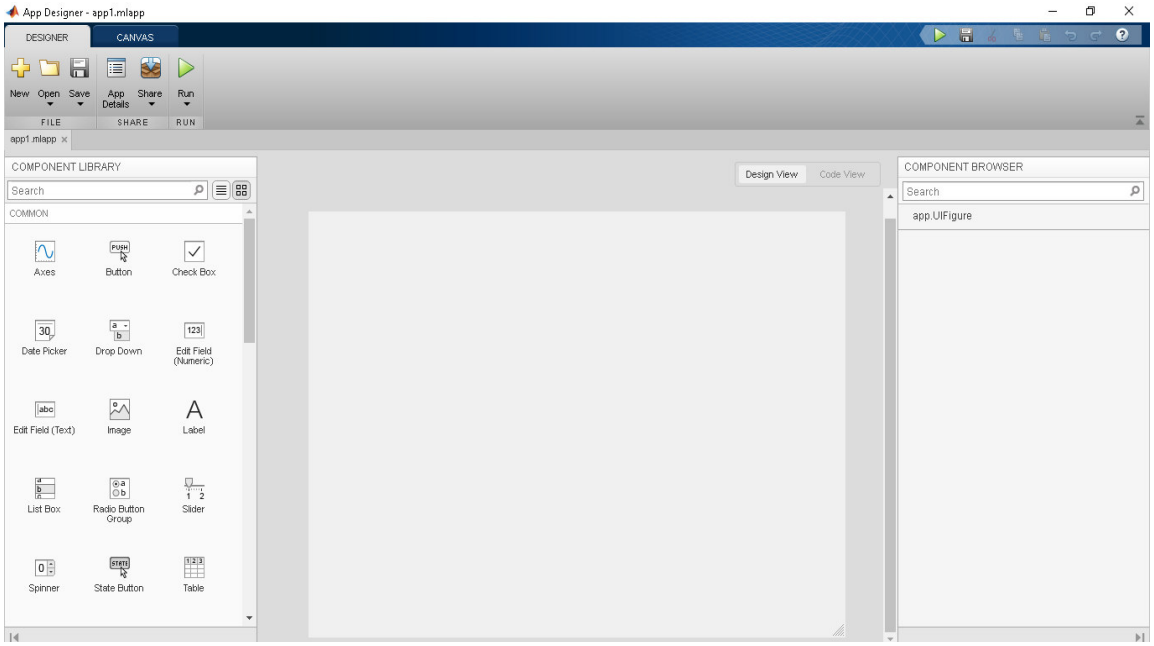


Figura 2.31 Interfaz de diseño App Designer

Dentro de las diferentes opciones que se encuentra para el diseño de la interfaz, se empieza por elegir la opción Tab-Group Figura 2.32, que permitirá al usuario elegir dentro de las diversas secciones a cuál se dirige.

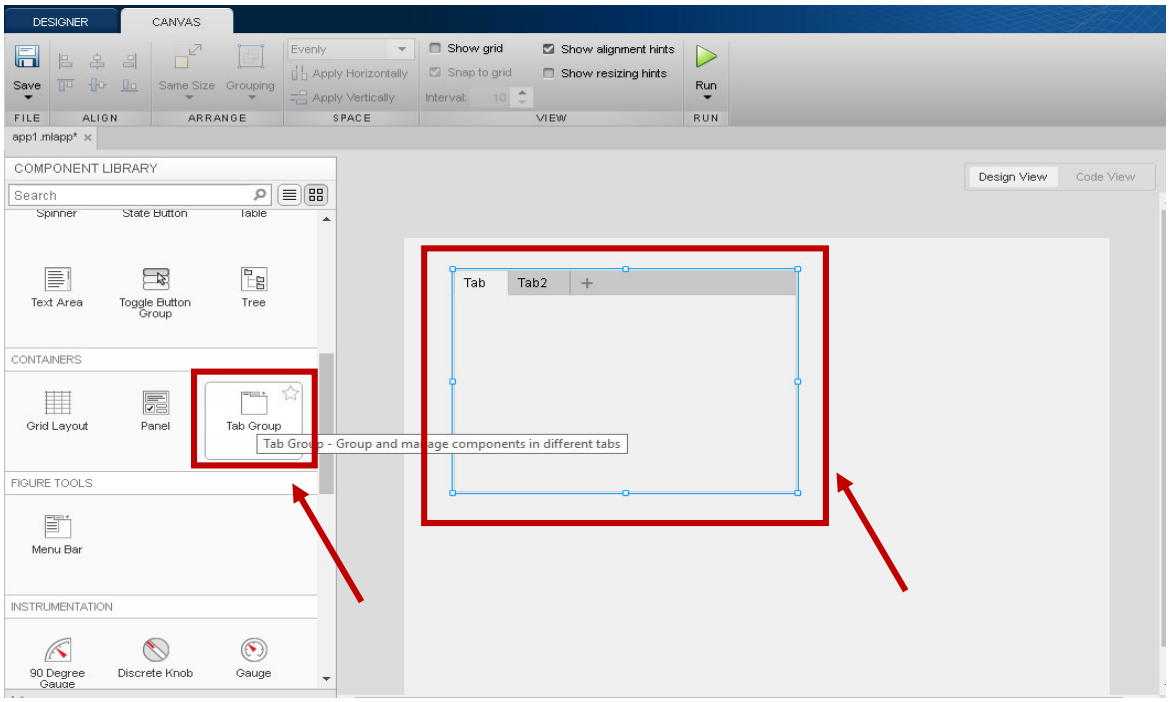


Figura 2.32 Selección Tab-Group

En la opción Tab-Grop se creará tres secciones diferentes donde el usuario puede dirigirse.

- Cargar Imagen: En esta sección el usuario cargará la imagen desde cualquier ruta donde se encuentren los archivos con las imágenes que se desea analizar.
- Conteo de Microorganismos: En esta sección se visualizarán los resultados del conteo de los microorganismos presentes en la palca petrifilm.
- Descriptores Geométricos: En esta sección se visualizarán los resultados de la circularidad de los microorganismos de mayor tamaño.

En la Figura 2.33 se observa las tres secciones creadas, cabe recalcar que siempre el primer paso va a ser cargar la imagen que se desea analizar, para esto se debe dirigir a la sección “Cargar imagen”, cumplido este requerimiento el usuario podrá obtener los resultados de las otras dos secciones restantes, como por ejemplo visualizar los resultados en la sección de conteo y la sección de descriptores geométricos.

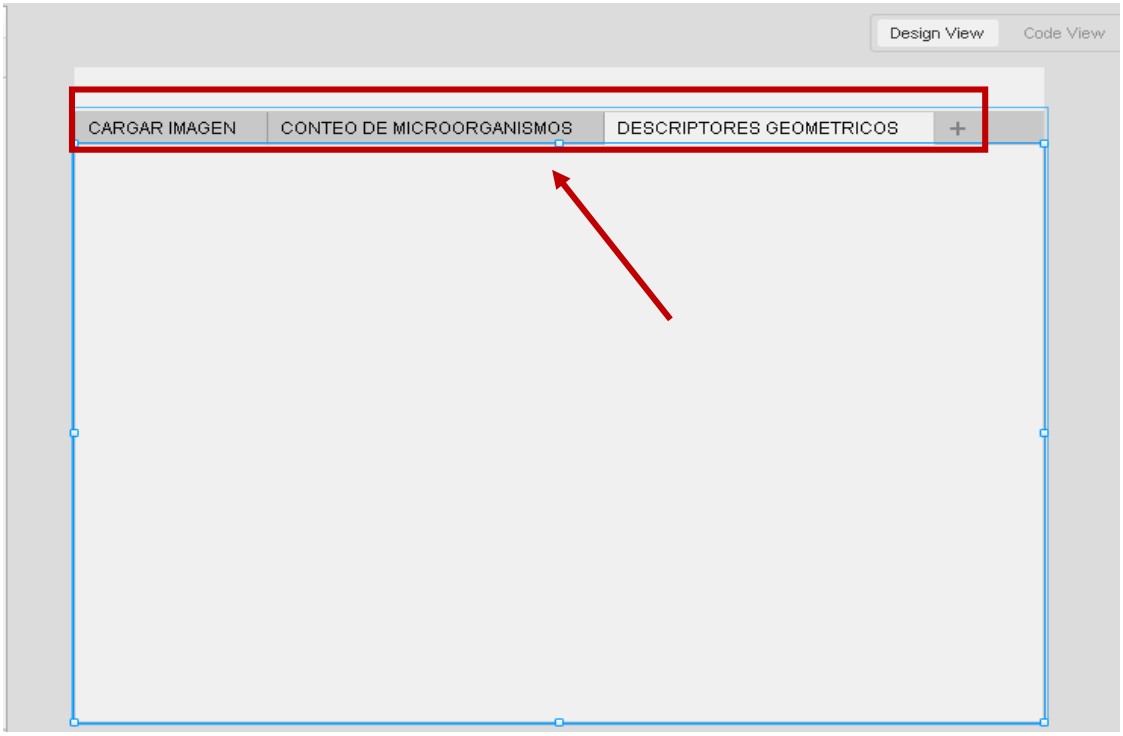


Figura 2.33 Tres secciones diferentes que puede elegir el usuario.

La aplicación App Designer se fundamenta en el uso de funciones, desde este punto de vista, cada elemento interactivo de la interfaz como botones, graficas etc son asociados solo a una única función, que a su vez contiene un segmento especifico de código que pertenece solo a esa función o elemento gráfico de la interfaz, entonces, eso significa que, cada variable creada dentro de una función solo puede ser utilizada dentro de la función, y no podrá ser usada por otras funciones, para solucionar este inconveniente se crea variables globales que serán visibles por todas las funciones existentes.

Para declarar variables globales se lo hace en el segmento de código `propierties (private=private)` como se indica en la Figura 2.34.

Para ingresar a la sección de código se dirige el cursor y se hace clic en code view , como se observa en la Figura 2.34.

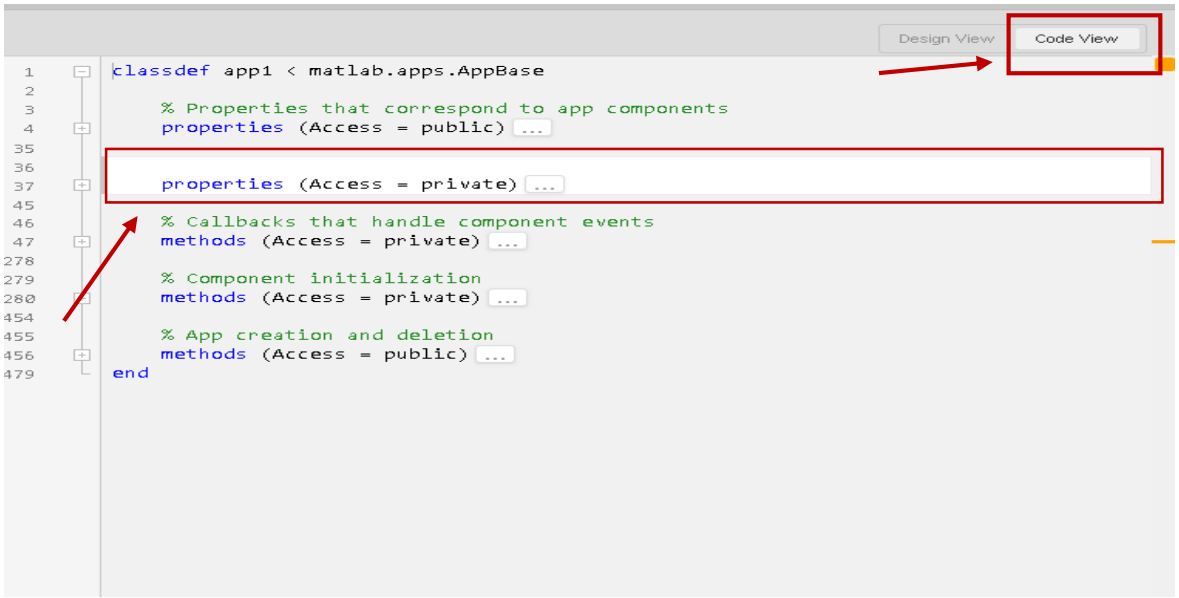


Figura 2.34 Sección de código donde se declaran las variables globales

En el Segmento de código 2.35 se visualiza las variables globales declaradas que serán utilizadas en la configuración de la interfaz gráfica de usuario

```
properties (Access = private)
        he; % Imagen cargada para ser analizada
        cc3; % Imagen con el total de los microorganismos
        BW3; % Iamgen con los microorganismos pequeños
        BW1; % Imagen con microorganismos grandes
    end
```

Segmento de código 2.35: Variables globales utilizadas en la interfaz gráfica

A continuación, se regresa a la interfaz gráfica, y en la sección cargar imagen se crea dos Botón y dos Axes, el primer Botón se lo nombra Cargar imagen y el segundo botón se lo nombra Procesamiento, y en el Axes 1 se indicará la imagen que fue elegida por el usuario para el análisis y en el Axes 2 se observará la imagen binaria ya procesada con todos los microorganismos. En la Figura 2.35 se observa el procedimiento antes mencionado.

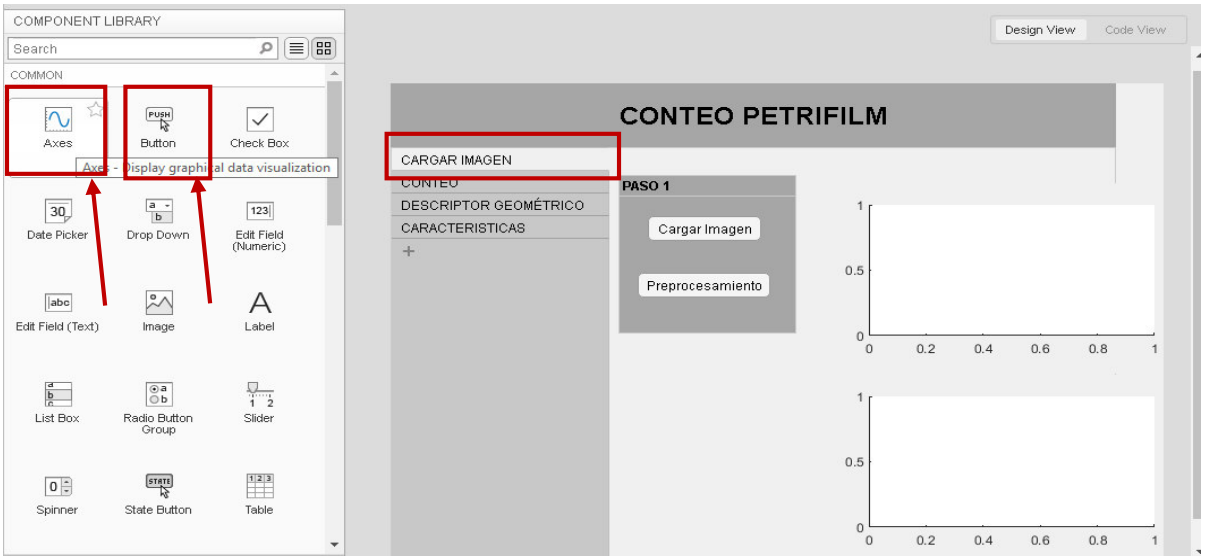


Figura 2.35 En la sección Cargar imagen se crea del botón de nombre Cargar Imagen y el botón de nombre Procesamiento, además de los Axes.

A continuación, se crea y asocia una función al botón denominado Cargar Imagen, para esto se hace clic izquierdo en el botón cargar imagen, y se señala la opción Callback como se visualiza en la Figura 2.36.

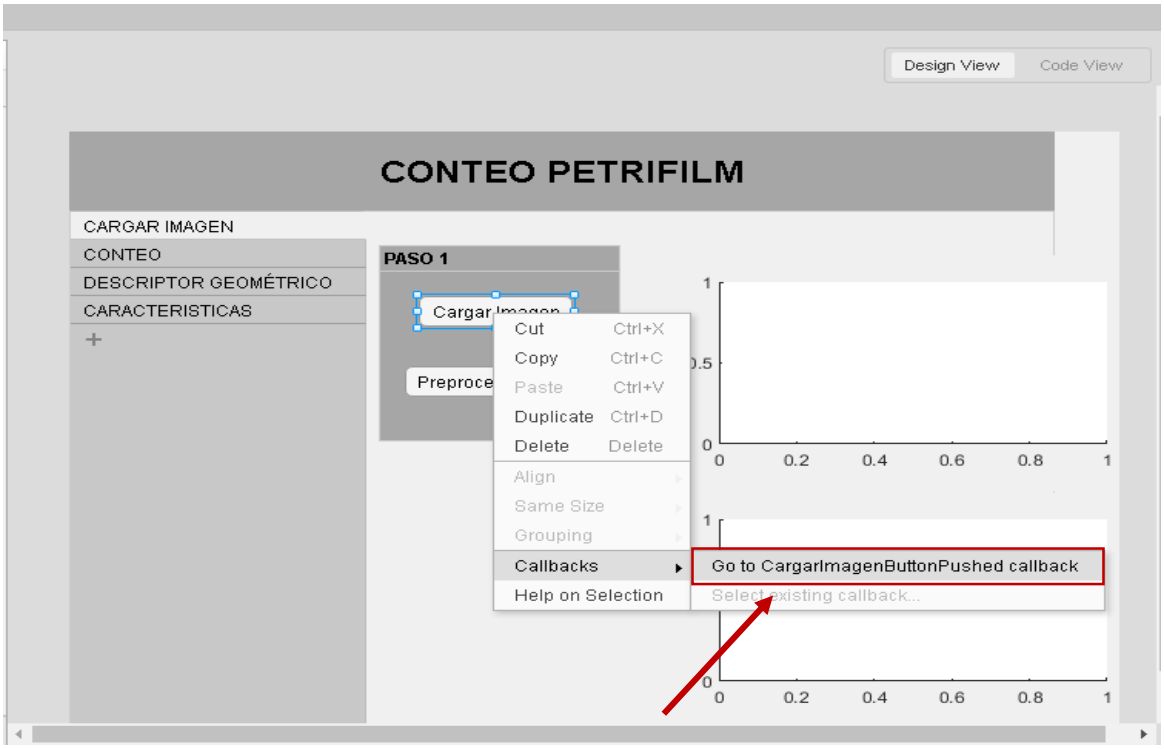


Figura 2.36 Creación de una función asociada al botón Cargar Imagen.

En la Figura 2.37 se indica la función creada y que a su vez se relaciona con el botón Cargar imagen.

Figura 2.37 Segmento de código donde se programa la función asociada al botón Cargar Imagen

En el Segmento 2.36 de código se detalla la función que se creó asociada al Botón “Cargar Imagen” y el código que se utilizó dentro de la función.

El código 2.36 permite que, el usuario al momento de hacer clic en “Cargar Imagen” se despliegue la opción de búsqueda para que el usuario pueda dirigirse a las carpetas donde se encuentran las imágenes para ser analizadas, una vez que se elige la imagen se carga automáticamente para su posterior visualización en el Axes 1 llamado app.UIAxes.

```
% Button pushed function: CargarImagenButton
  function CargarImagenButtonPushed(app, event)
   [FileName,PathName]=uigetfile('*.jpeg','select an image ');
     app.he=imread(strcat(PathName,FileName));
     imshow(app.he,'parent',app.UIAxes);
     app.UIAxes_2.cla;
     app.EditField.Value=0;
     app.EditField_2.Value=0;
  end
```

Segmento de código 2.36: Función asociada al Botón Cargar Imagen.

Una vez que se cargó la imagen en la aplicación, el usuario después tiene que hacer clic en el botón “Procesamiento”, con el objetivo de obtener todos los resultados del conteo y de descriptores geométricos, dichos resultados serán visualizados en la sección “Conteo” y la sección “Descriptores Geométricos”.

A continuación, se asocia el botón “Procesamiento” a una función, se sigue un procedimiento similar al que se realizó con el botón “Cargar Imagen”. El procedimiento se visualiza en la Figura 2.34.

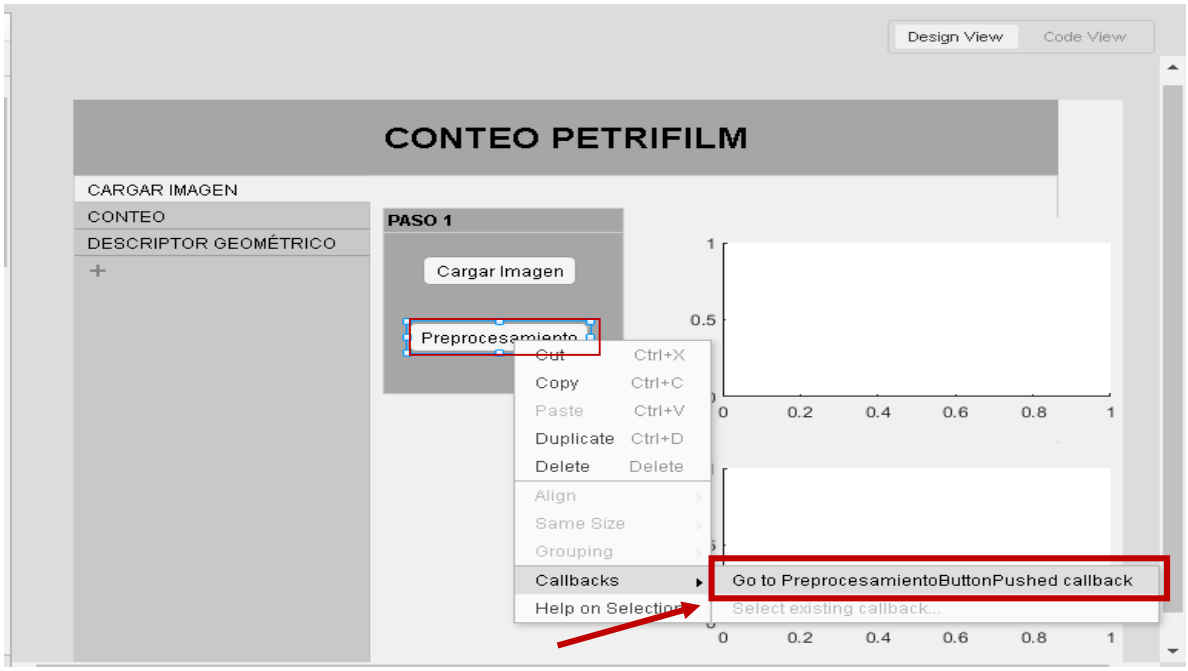


Figura 2.38 Creación de una función asociada al botton Procesamiento.

En la función creada y asociada al botón" Procesamiento", se carga todos los Segmentos de código indicados en las secciones 2.3 y 2.4.

```
   % Button pushed function: PreprocesamientoButton
function PreprocesamientoButtonPushed(app, event)
  % Button pushed function: PreprocesamientoButton
       function PreprocesamientoButtonPushed(app, event)
lab_he = rgb2lab(app.he);
ab = lab_he(:,:,2:3);
ab = im2single(ab);
nColors = 2;
pixel_labels =
imsegkmeans(ab,nColors,'NumAttempts',2,'NormalizeInput',false,'MaxIterations',100,'Thr
eshold',5);
title('Image Labeled by Cluster Index');
%MASCARA DE LINEAS%
mask2=pixel_labels==2;
figure
imshow(mask2)
cluster2=(app.he).*(uint8(mask2));
figure
imshow(cluster2)
 % CONTEO DE MICRORGANISMOS EN LINEAS
I1=rgb2gray(cluster2);
imhist(I1);
figure
imshow(I1)
claheI = adapthisteq(I1,'NumTiles',[10 10],'clipLimit',0.02,'Distribution','uniform');
claheI = imadjust(claheI);
figure
```

```
imshow(claheI);
imhist(claheI);
[f,c]=size(claheI);
for i=1:f
    for k=1:c

        if claheI(i,k)<=160
           claheI(i,k)=0;
        else
          claheI(i,k)=250;

        end

    end
end
m=~claheI;
heI=mask2.*m;
imshow(heI);
CCC = bwconncomp(heI)
% MASCARA PARA AISLAR IMAGEN DE INTERES ,INCLUYE LINEAS
c1=imfill(mask2,'holes');
a1=bwareafilt(c1,1);
s1=strel('disk',40);
s2=strel('disk',40);
s3=strel('disk',10);
c2=imdilate(a1,s2);
c3=imfill(c2,'holes');
c4=imclose(c3,s1);
for i=1:1280
    for j=1:720

       if j>=1 & j<=8 | j>=711 & j<= 720 |i>=1100
       v(i,j)=0;
       else
        v(i,j)=1;
       end

   end
end
imshow(v)
c4=v.*c4
```

Segmento de código 2.37: Función asociada al Botón “Procesamiento”.

Para todas las demás secciones se efectúa el mismo procedimento realizado con el botón “Cragar imagen” y el botón “Procesamiento”.

A continuación, se indica la creación de la sección denominada Conteo, ver Figura 2.39, en la sección “Conteo” es donde se visulizarán los resultados del conteo de los microorganismsos en la placa petrifilm, dentro de la sección conteo existen tres campos, el primer campo indica el conteo total de todos los microoganismos presentes en la placa petrifim, el segundo campo el usuario al hacer clic en “Conteo BC” se muestra el conteo de

los microoganismos de mayor tamaño y el campo último “Conteo SC” indica el conteo de los microorganismos de menor tamaño. En la Figura 2.39 se observa la creación de los tres campos donde se visualizarán los resultados del conteo de los microorganismos.

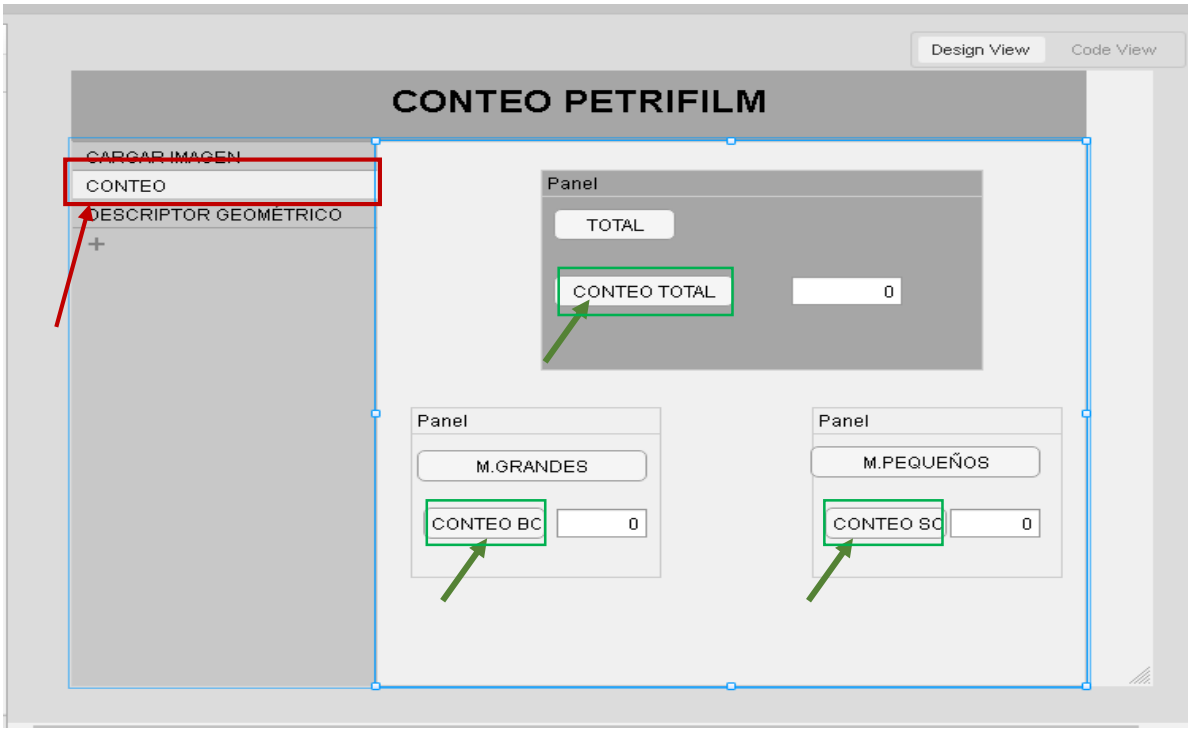


Figura 2.39 Sección donde se visualiza los resultados del conteo de los microorganismos.

Cada uno de los tres campos está representado por una función diferente, en el Segmento de código 2.38 se detalla la creación de las tres funciones, gracias a las variables globales creadas en el Segmento de código 2.35, las tres funciones pueden interactuar entre si y a su vez con las funciones “Procesamiento” y la función “Cargar imagen”.

```
% Button pushed function: CONTEOBCButton
        function CONTEOBCButtonPushed(app, event)
           app.EditField.Value=app.b;
        end

 % Button pushed function: CONTEOSCButton
        function CONTEOSCButtonPushed(app, event)
        x=app.a-app.b;
        app.EditField_2.Value=x;
        end

 % Button pushed function: CONTEOTOTALButton
```

```
function CONTEOTOTALButtonPushed(app, event)
app.EditField_3.Value =app.a
end
```

Segmento de código 2.38: Funciones que permiten visualizar los resultados del conteo de los microorganismos en la interfaz gráfica.

A continuación, se configura la última sección denominada descriptores geométricos, ver Figura 2.40, donde el usuario tiene que hacer clic en visualizar, para observar los resulatdos de la circularidad, y la identificación los microorganismos de mayor tamaño en la imagen original.

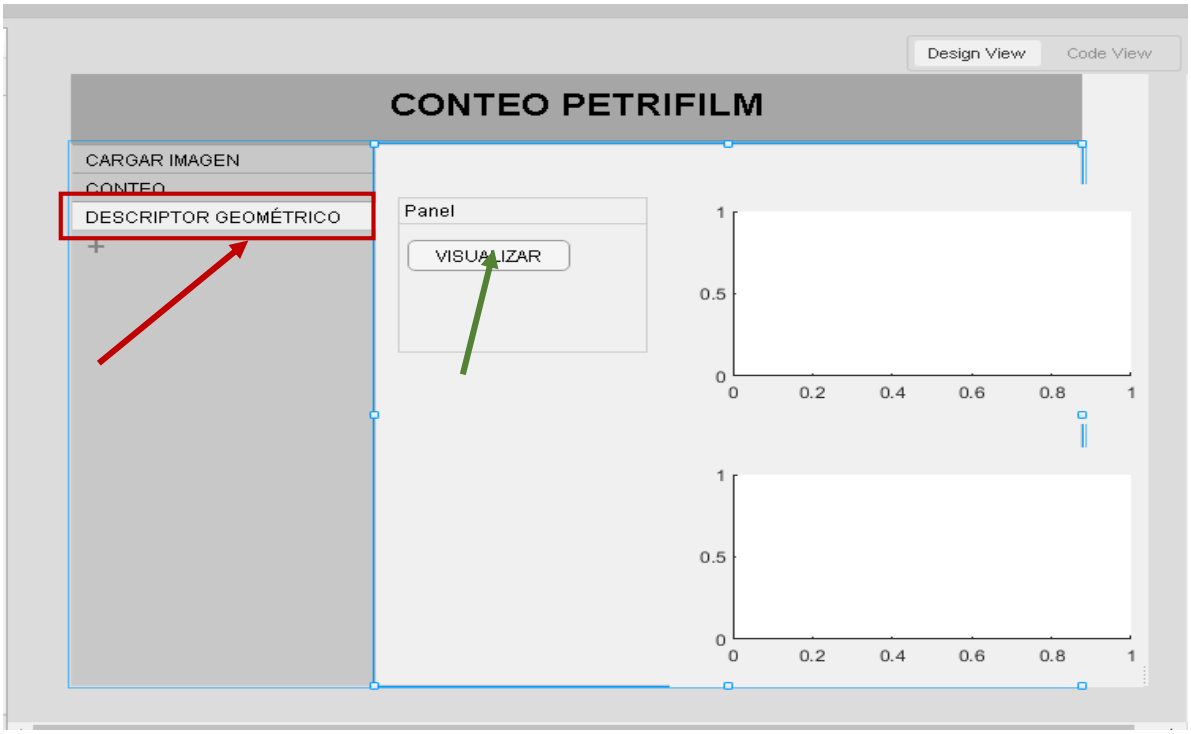


Figura 2.40 Sección donde se visualiza los resultados de la circularidad

3. RESULTADOS Y DISCUSIÓN

En el presente capítulo se describe los resultados de la segmentación de imágenes con k-means, de forma similar se indica los resultados de utilizar la ecualización de histograma adaptativa para mejorar el contraste de la imagen, además el resultado del conteo de los microorganismos.

3.1 RESULTADOS CONSEGUIDOS EN LA SEGMENTACIÓN CON K-MEANS.

En la Figura 3.1 literal a) se observa que la imagen original contiene diversas tonalidades, lo que dificulta aplicar un método único para la extracción de los microorganismos presentes en la imagen, lo que hizo indispensable segmentar la imagen en diversas imágenes para después ser analizadas cada una por separado.

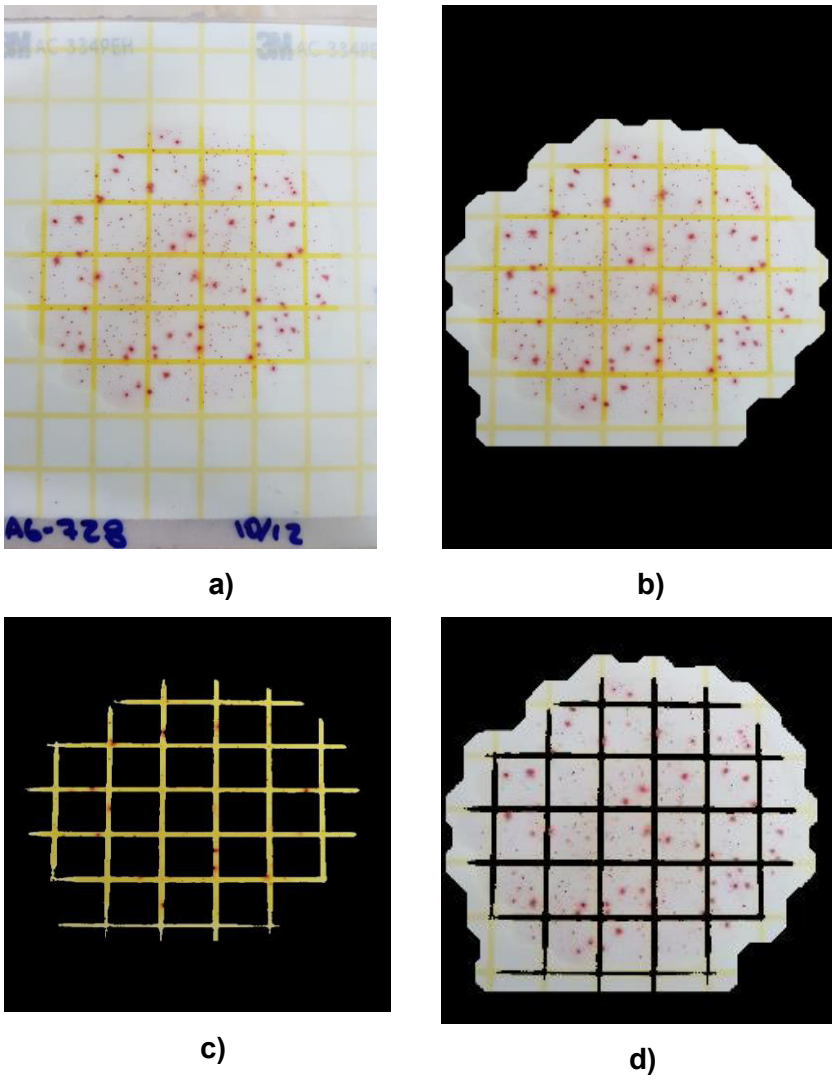


Figura 3.1 Etapas de segmentación de la imagen en diversas secciones

En la Figura 3.1 literal b) se segmentó la imagen en una sección donde solo se encuentran los microorganismos de la placa petrifilm, y a partir de la sección del literal b) se derivó dos secciones más como, la sección c) solo de lineas, y la sección d) imagen que no contiene lineas.La sección c) y d) se analizaron por separado, ya que por sus tonalidades semejantes se facilita el procesamiento y la extracción de los microorganismos presentes en cada imagen.

Un problema que se presentó es el que se observa en la Figura 3.2, que a pesar de la correcta segmentación de la sección de interés, en el posterior procesamiento de la imagen hubo problemas para el conteo de los microorganismos, las dificultades fueron originadas por causa que la sección segmentada de interés presenta intersecciones con el borde de la imagen, la solución para este problema se detalla en el segmento de código 2.13, la cual fue construir una imagen binaria que funcione como máscara, pero con la condición que todos los pixeles del borde tengan un valor de 0 y los demás pixeles tomen un valor de 1. Al multiplicar la máscara creada por la Figura 3.2 origina que no exista intersección en los bordes.

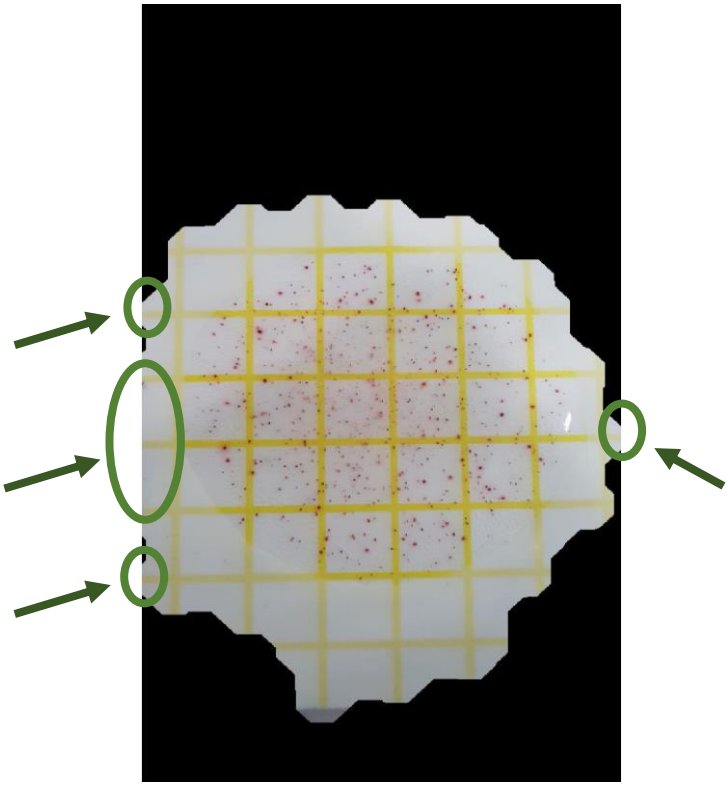


Figura 3.2 Sección segmentada presenta 4 intersecciones en los bordes de la imagen

La Figura 3.3 es resultado de multiplicar la máscara por la Figura 3.2, se observa que la sección de interés no presenta intersección con los bordes de la imagen.

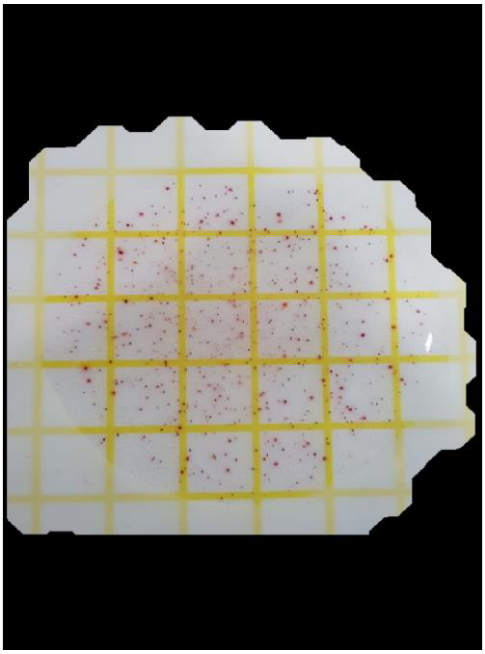


Figura 3.3 Sección segmentada de la Figura 3.2 sin intersección en los bordes

Otro caso que originó inconvenientes es cuando existe dificulatd en el enfoque de la cámara al momento de tomar la foto de la placa petrifilm, esto causó problemas al momento de extarer con presición los microorganismos de la placa petrifilm.

En la Figura 3.4 se observa un ejemplo del problema antes mencionado de una mal enfoque, que podría ocasionar problemas en el conteo.

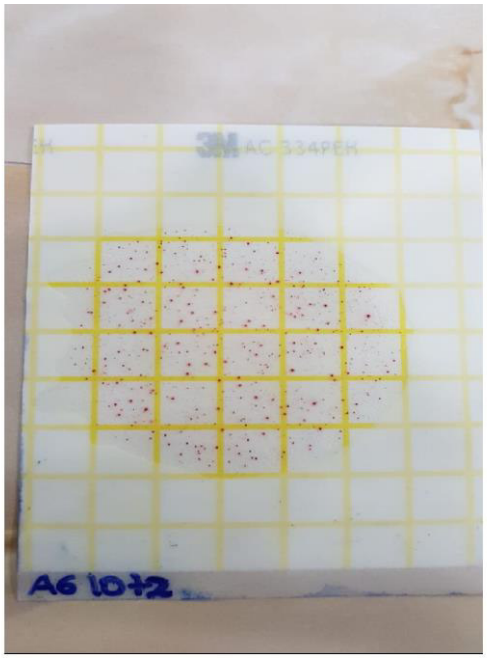
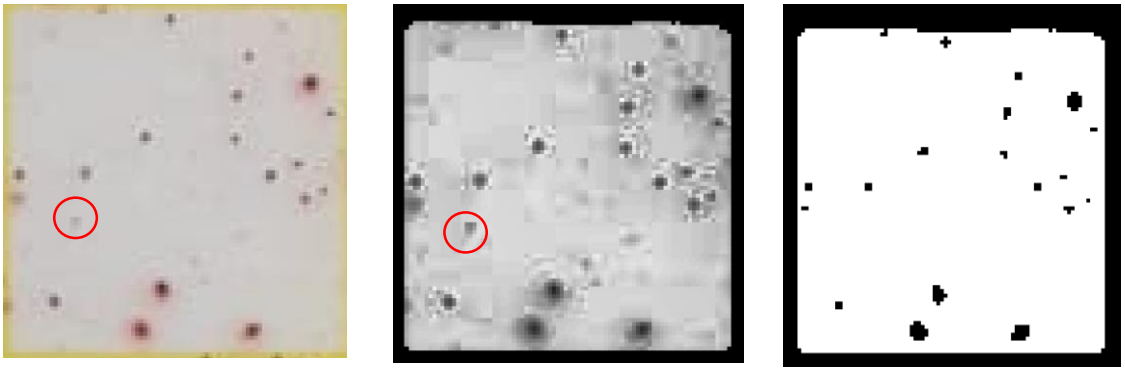


Figura 3.3 Ejemplo de imagen de placa petrifilm con un mal enfoque que origina problemas en el conteo.

3.2 RESULTADOS EN LA APLICACIÓN DEL ALGORITMO DE ECUALIZACIÓN ADAPTATIVA DE HISTOGRAMA.

La ecualización adaptativa de histograma se aplicó a las imágenes en escala de grises para mejorar el contraste, con el objetivo de optimizar la visualización de los microorganismos que eran opacos y poco legibles.Hubieron casos en que los microorganismos poco legibles no eran detectados por el algoritmo de ecualización adaptativa de histograma, y al momento de binarizar la imagen los microorganismos desaparecian y no eran contabilizados.

En la Figura 3.4 literal a) se observa un pequeño segmento de la imagen original amplificada para una mejor visualización, el literal b) indica una imagen con ecualización adaptativa de histograma y el literal c) la misma imagen binarizada.



a) b) c)

Figura 3.4 Segmento de la imagen original amplificada a) , imagen en escala de grises literal b) , imagen binarizada c).

En la Figura 3.4 se observa que a pesar que se aplicó la técnica de ecualización adaptativa de histograma el microorganismo encerrado en el círculo rojo al momento de binarizar no fue contabilizado.

3.3 RESULTADOS DEL CONTEO UTILIZANDO LA INTERFAZ GRÁFICA EN APP DESIGNER.

Para que el usuario obtenga el conteo de los microorganismos presentes en la imagen de la placa petrifilm, primero tiene que hacer click en el botón “Cargar la Imagen” y de forma inmediata se despliegan los archivos donde el usuario tienen que elegir la imagen que desea analizar como se observa en la Figura 3.5.

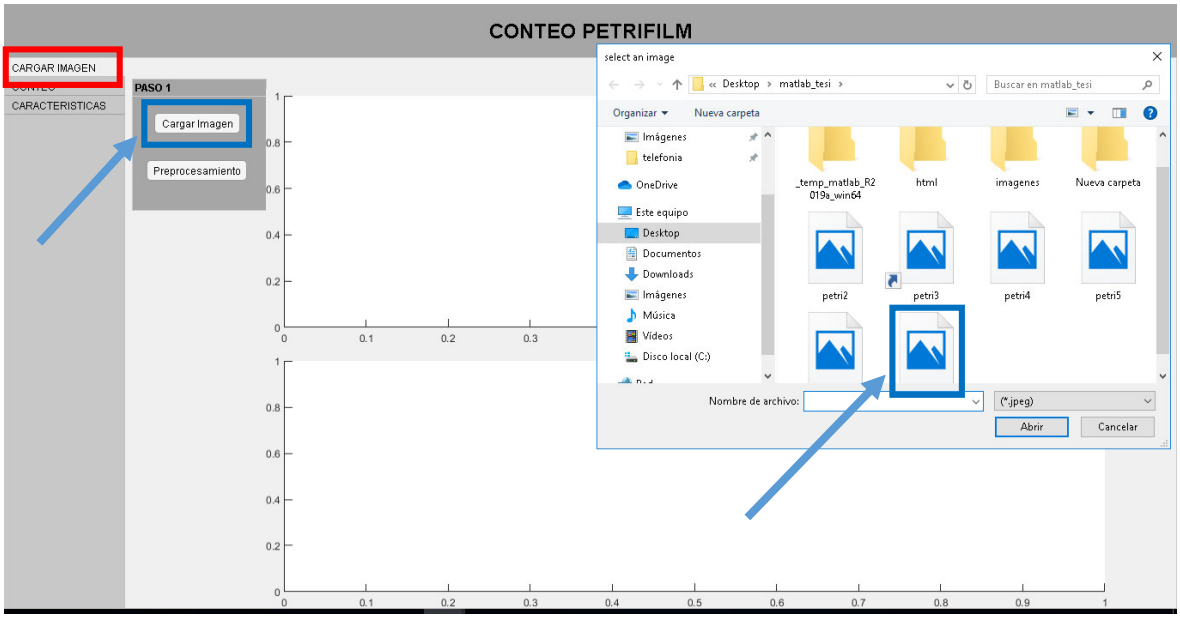


Figura 3.5 Procedimiento para cargar la imagen que se desea analizar.

Si por algún motivo el usuario se olvida de cargar la imagen y presiona cualquier otro botón antes del botón “Cargar imagen”, inmediantamente se despliega un mensaje de error, donde le informa al usuario que primero debe cargar la imagen, el procedimiento se visualiza en la Figura 3.6.

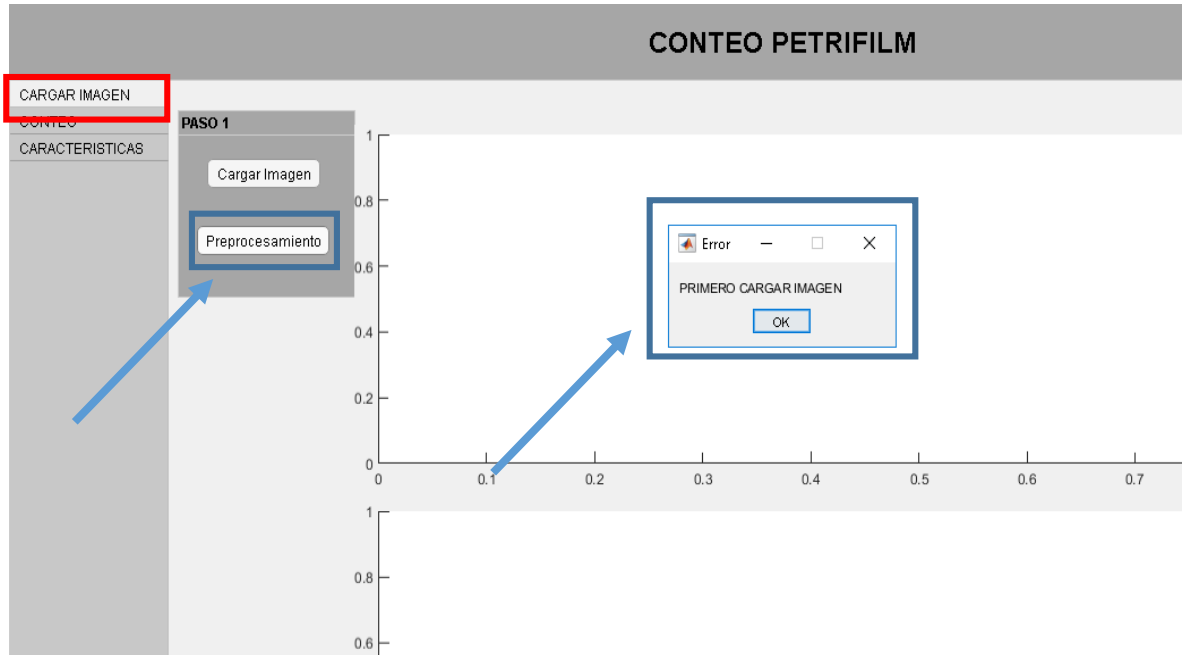


Figura 3.6 Despliegue de mensaje de error

Una prueba que la imagen de análisis se cargó correctamente es que, la imagen aparecerá en la parte derecha de la interfaz gráfica como se visualiza en la Figura 3.7.

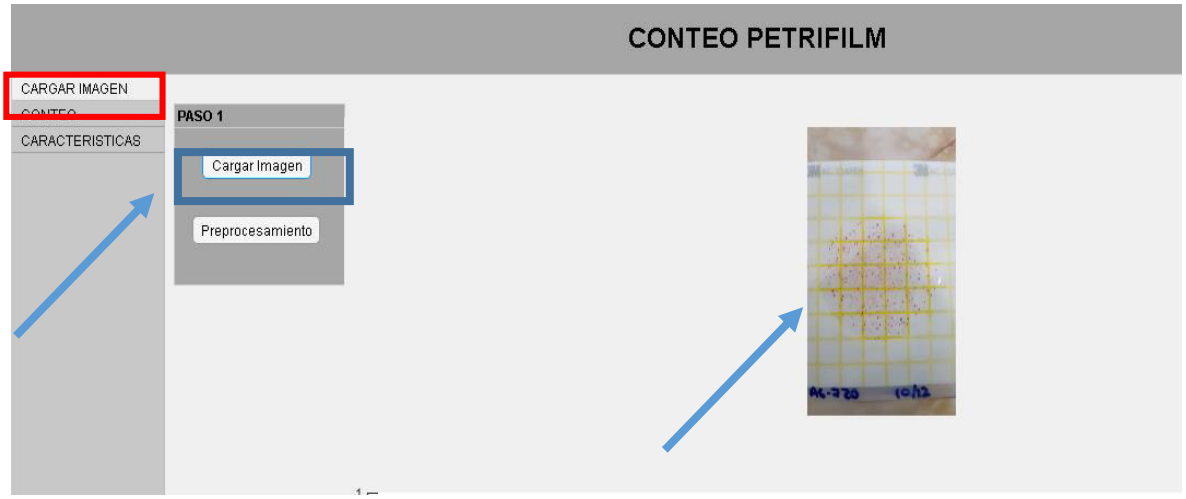


Figura 3.7 Imagen de placa petrifilm ya cargada

Una vez que se cargó la imagen que será analizada, el usuario debe presionar el botón “Procesamiento”, con el objetivo de procesar la imagen y obtener todos los resultados deseados. Para comprobar que el procesamineto se realizó de una forma adecuada la imagen binaria con todos los microorganismos extraidos se visualizarán en el extremo inferior derecho como se observa en la figura 3.8.

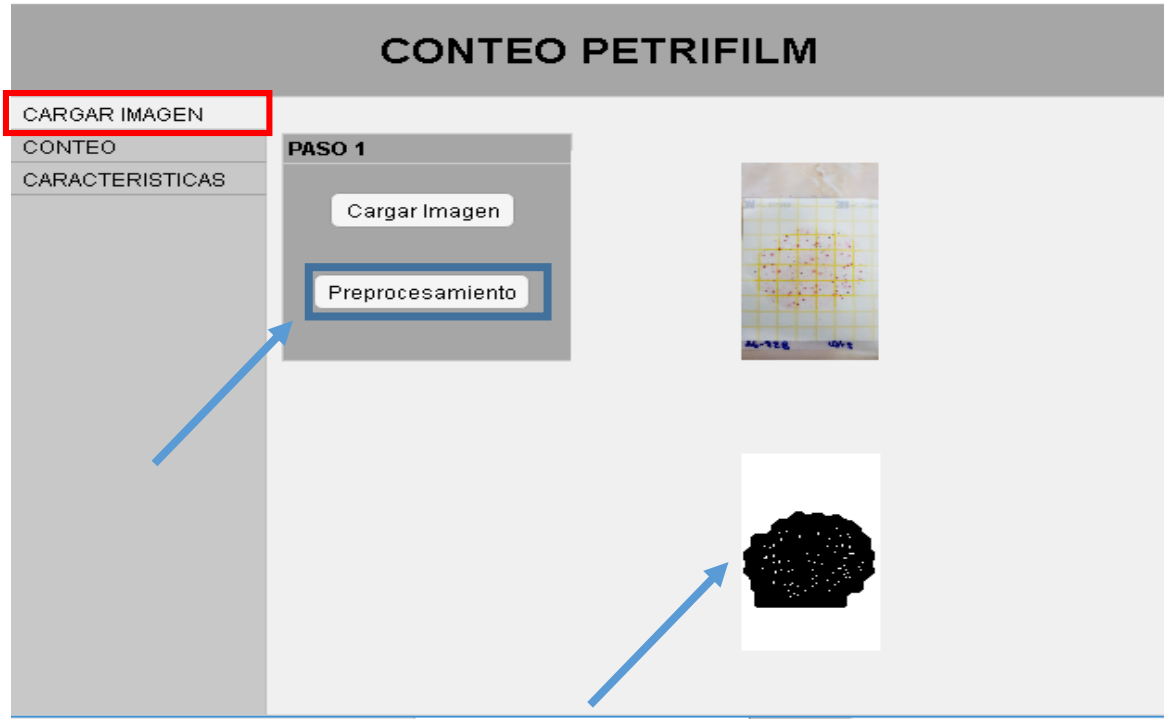


Figura 3.8 Procesamiento de Imagen de la placa petrifilm.

Una vez que se ha cargado y se ha procesado la imagen, el usuario se dirige a la sección de “Conteo”, donde puede observar el resultado de cuantos microorganismos en total están presentes en la placa petrifilm, además de contabilizar cuantos microoganismos son de mayor tamaño.

El usuario puede dar click en cualquiera de los tres botones para obtener los resultados del conteo de los microorganismos grandes, pequeños y el total , en la Figura 3.9 se visualiza la sección de la interfaz gráfica que permite el conteo.

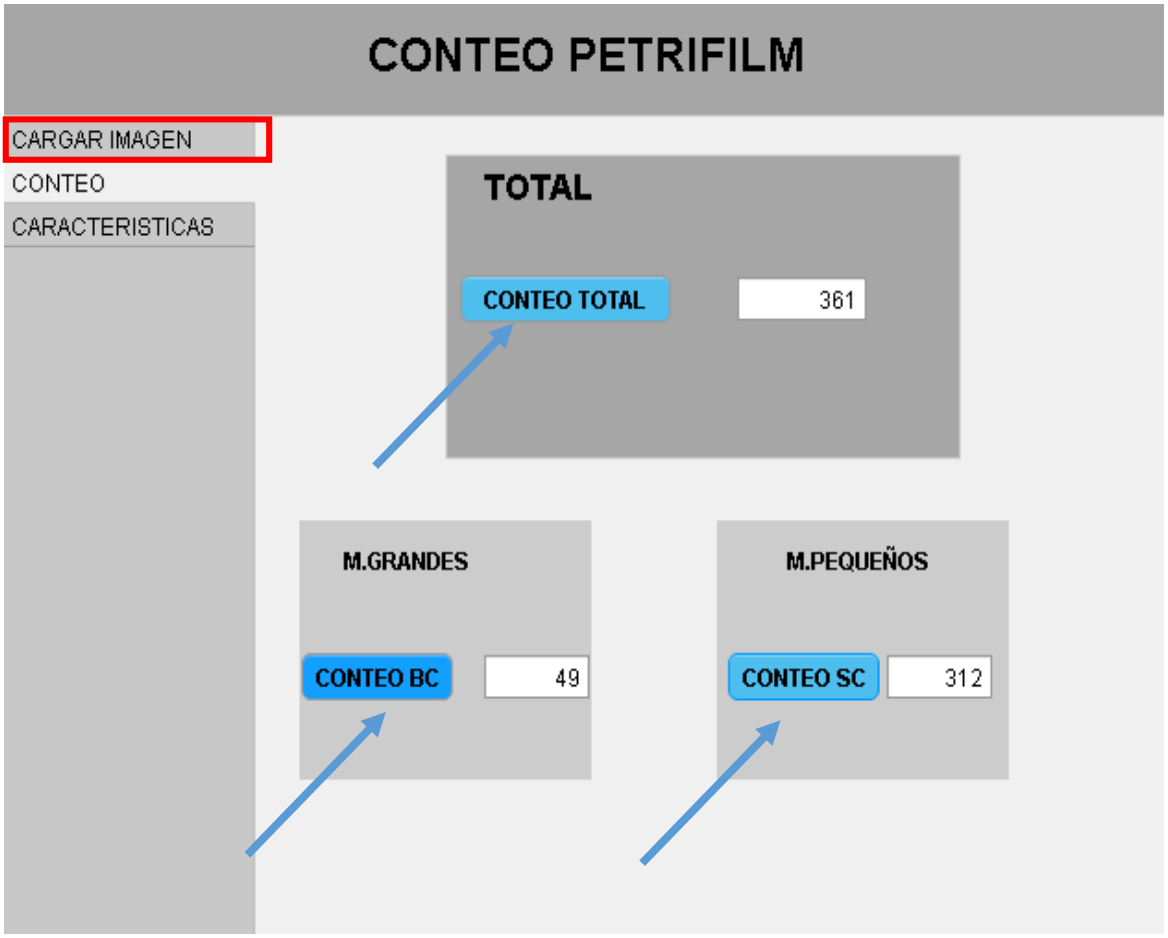


Figura 3.9 Conteo de los microorganismos.

3.4 RESULTADOS DE LOS DESCRIPTORES GEOMÉTRICOS

En la sección de descriptores geométricos se calcula el perímetro, área y redondez de los microorganismos de mayor tamaño, en la Figura 3.10 se observa el cálculo de la redondez de los microorganismos,

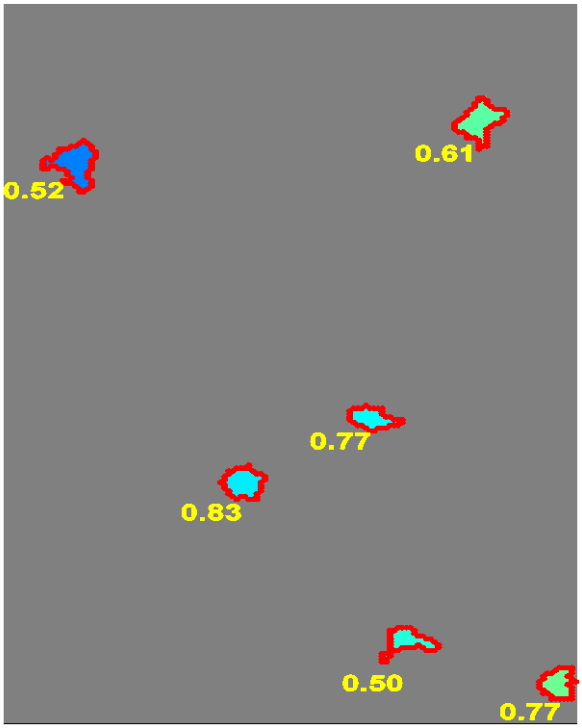


Figura 3.10 Redondez de los microorganismos.

En la Figura 3.11 se resalta los microorganismos que alcanzaron un mayor tamaño.

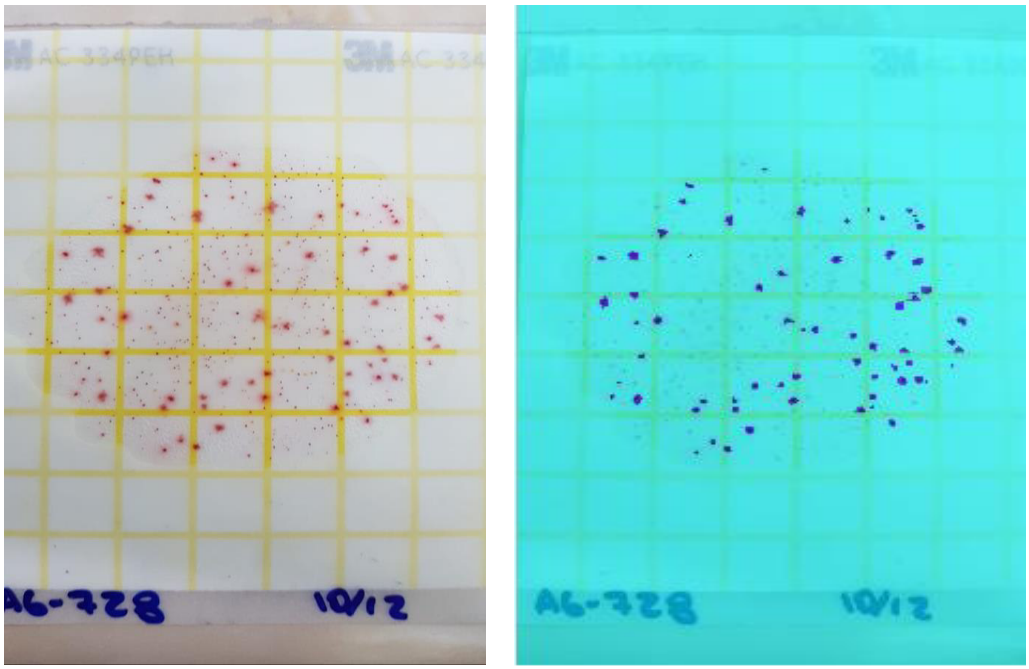


Figura 3.11 Microorganismos de mayor tamaño.

La Tabla 3.1 contiene los valores del conteo de la USFQ y los resultados en Matlab, además se calculó el error promedio, del cual se obtuvo un valor de 2.79%

NOMBRE	CONTEO USFQ	CONTEO MATLAB	ERROR
petri1	480	509	4.91
petri2	596	611	2.454991817
petri4	802	829	3.256936068
petri5	402	432	6.944444444
petri6	583	590	1.186440678
petri7	695	712	2.387640449
petri8	720	709	1.551480959
petri9	696	708	1.694915254
petri10	702	712	1.404494382
petri11	810	825	1.818181818
petri12	815	827	1.451027811
petri13	823	832	1.081730769
petri14	793	821	3.41047503
petri15	427	434	1.612903226
petri16	410	432	5.092592593
petri17	415	428	3.037383178
petri18	412	430	4.186046512
ERROR PROMEDIO			**2.793040293**

Tabla 3.1 Cálculo del error promedio entre valores de USFQ y Matlab

En la Figura 3.12 muestra los valores de la USFQ (Gráfica Naranja) y los valores Matlab(Gráfica Azul).

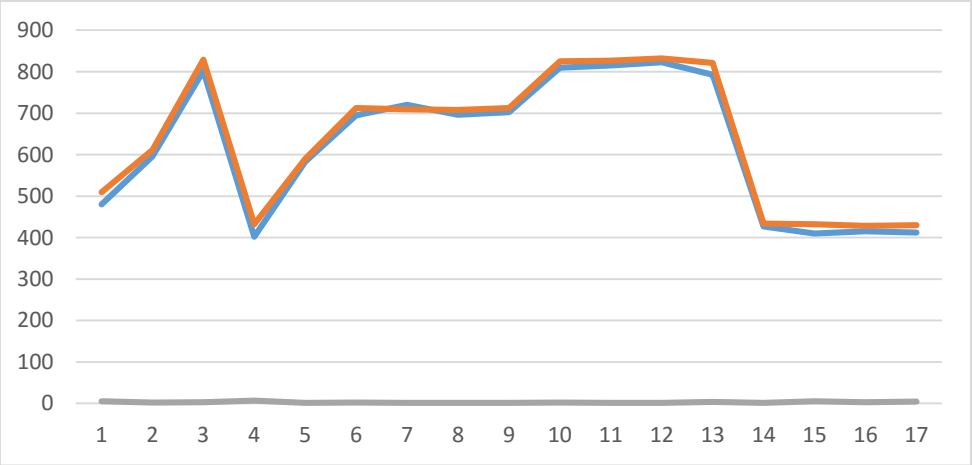


Figura 3.12 Gráfica de valores USFQ y Matlab.

La validación de resultados ante el departamento de Microbiología de la Universidad San Francisco de Quito se realizó mediante la herramienta de video conferencia Zoom, asistieron como representantes de la USFQ la Phd. Sonia Zapata y los técnicos del laboratorio, el resultado fue satisfactorio, ya que manifestaron su conformidad porque la herramienta será de valiosa utilidad en el Departamento de Microbiología.

4. CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- Ya finalizado el presente Proyecto de Titulación se consigue como resultado un programa en Matlab que permite el conteo de microorganismos presentes en una placa petrifilm utilizando visión computacional. Para segmentar la sección de interés el programa incluye operaciones morfológicas y la técnica de aprendizaje no supervisado k-means, además se utiliza algoritmos basados en vecindad y conectividad de pixeles que permiten realizar el conteo total de todos los microorganismos presentes en la placa petrifilm, finalmente se usa funciones que permiten detectar bordes y geometrías circulares, así como también identificación de centros, áreas y perímetros.

- Uno de los propósitos principales del programa desarrollado en el presente Trabajo de Titulación fue conseguir procesar la imagen sin que se pierda ningún detalle, por más minúsculo que este sea, por este motivo se utilizó la técnica de ecualización de histograma que consiguió mejorar significativamente el contraste en la imagen de escala de grises, pero hubo casos aislados con los microorganismos de menor tamaño que presentaban casi nula legibilidad, esto ocasionó que no fueran detectados por el algoritmo de ecualización de histograma y se perdieran al momento de binarizar la imagen.

- Para el conteo de los microorganismos se utilizó la técnica de etiquetado de componentes conexas que está basada en conectividad-8 ,que en su aplicación funcionó a la perfección, además se utilizó la transformada de houg para analizar los microorganismos de mayor tamaño que estén parcialmente solapados, cabe recalcar que este algoritmo fallará cuando los microorganismos estén completamente solapados, además que la transformada de houg causaba inconvenientes y falsos resultados cuando se analizaba los microorganismos extremadamente pequeños, por tal motivo solo se utilizó en los microorganismos de mayor tamaño.

- Los elementos y métricas utilizadas para aislar los microorganismos de mayor tamaño, como por ejemplo los rangos de radios fueron elegidos en función de las imágenes disponibles en la base de datos, por lo tanto, el método aplicado en el programa limita su funcionalidad a las imágenes que se encuentran presentes en la base de datos.

- Las métricas manejadas en el algoritmo k-means solo son funcionales para las imágenes utilizadas en el presente trabajo de titulación, si se requiere hacer el conteo de microorganismos con imágenes de tonalidades totalmente diferentes a las que se encuentra en la base de datos se puede tener errores significativos en los resultados.
- El error promedio que se calculó es del 2.79%, pero también se observa que al analizar valores individuales la gran mayoría son valores menores al 2%, existen valores aislados, como por ejemplo un valor de error del 6%, estos valores coinciden con las imágenes en las que las tomas de la placa son desalineadas.

4.2 RECOMENDACIONES

- El programa creado en el presente Trabajo de Titulación es efectivo cuando la imagen de placa petrifilm es capturada y enfocada de una manera adecuada, la recomendación al momento de tomar la foto es que sea lo más cercano posible a la placa petrifilm, para así no perder ningún detalle de la imagen, además sería de vital importancia la calidad de la cámara fotográfica, que la misma presente una excelente resolución.
- El presente programa presenta algunas limitaciones, como por ejemplo que nuestro conjunto de datos contiene pocas muestras de las fotos de las placas petrifilm, además que todas las imágenes se encuentran bajo las mismas condiciones de iluminación, entonces valdría la pena conseguir imágenes en condiciones más variadas para expresar una generalización al programa.
- Los algoritmos utilizados en el presente Trabajo de Titulación no representan un gran trabajo computacional, lo que permitiría en un futuro ampliar las ideas desarrolladas en el presente programa para implementar una aplicación móvil, generalizando métricas para que sea funcional para cualquier tipo de placa petrifilm.
- El método de solución implementado en el presente trabajo de titulación mediante algoritmos de procesamiento digital indica una visión amplia de cómo se realiza el conteo de microorganismos utilizando visión computacional, pero se puede extender el mismo raciocinio para el conteo de objetos en otras ramas de aplicación.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] 3MTM PetrifilmTM (2015), “Guía de Interpretación conteo placas petrifilm ” [En línea]. Disponible: http://jornades.uab.cat/workshopmrama/sites/jornades.uab.cat.workshopmrama/files/Petrifilm_guias.pdf

[2] 3M Food Safety Department (2014, Noviembre 13), “Protegiendo al consumidor. Mejorando la eficiencia” [En línea]. Disponible: https://multimedia.3m.com/mws/media/1419814O/3m-petrifilm-plates-and-reader-spain.pdf

[3] 3M™ Placas Petrifilm™ (2015) “Recuento de Aerobios AC” [En línea]. Disponible: https://multimedia.3m.com/mws/media/1409674O/guia-interpretacin-petrifilm-aerobios.pdf

[4] Evaluación del método petrifilm (2014, junio 9), “Recuento de microorganismos aerobios “[En línea]. Disponible:

http://www.ugr.es/~ars/abstract/41-383-00.pdf

[5] P. Parra, T. Negrete, J. Llaguno and N. Vega, "Simulación y Conteo de Unidades Formadoras de Colonias," IEEE ANDESCON Conference, Santiago de Cali, Aug. 2018, pp. 1-10.

[6] M. Guerrón, K.Torres, “Image processing algorithm for improving the identification of patterns in mosquitoes wings”, Tesis para obtener el título de Ingeniero Electrónico, USFQ, Quito, 2016

[7] L. Yat-Tung, S. Pang-Chui, “Studies on organoleptic properties of food products from fresh egg and egg powder through principal component analysis”, Food Chemestry, 240(767), 2018.

[8] Gonzalez, R. Digital Image Processing Using Matlab. New Jersey:Prentice Hall, 2002, pp. 519-547.

[9] V. Bonilla, S. Hidrobo, “Diseño e implementación de un sistema de control de calidad de rosas utilizando técnicas de visión artificial para la empresa Bosqueflowers”, Tesis para obtener el título de Ingeniero en Electrónica y Control, EPN, Quito, 2014.

[10] J. Marfugás (2007), “Evaluación Sensorial de los Alimentos”, Editorial Universitaria, Universidad de la Habana, Cuba. pp. 11-16.

[11] Universidad Nacional Autónoma de México."Tratamiento de imágenes en Matlab"n.d. 25 Feb. 2006. [En línea]. Disponible: https://www.fedecacao.com.co/portal/index.php/es/2015-04-23-20-00-33/documentos-tecnicos

[12] R. C. Gonzalez, & R. E. Wood, "Digital image processing", Upper Saddle River, New Jersey: Prentice Hall, 2002, pp. 519-547.

[13] N. Dhanachandra, K. Manglem, & Yambem JinaChanu, "Image Segmentation Using K -means Clustering Algorithm and Subtractive Clustering Algorithm", Procedia Computer Science Journal, vol. 54, Aug. 2015, pp. 764-771.

[14] J. Oña "Sistema de clasificación de granos de cacao", Tesis para obtener el título de Ingeniero en Electrónica y Telecomunicaciones, EPN, Quito, 2020.

[15] P. J. Baldevbhai and R. Anand, "Color image segmentation for medical images using l* a* b* color space," IOSR Journal of Electronics and Communication Engineering, vol. 1, no. 2, pp. 24–45, 2012.

[16] N. Roncagliolo, "Image Segmentation Using Matlab", Procedia Computer Science Journal, vol. 1, Aug. 2015, pp. 100-120.

[17] P. Parra, "Computer vision techniques," in *2018 IEEE ANDESCON*. IEEE, 2018, pp. 1–10.

[18] H. Lewis and M. Brown, "A generalized confusion matrix for assessing area estimates from remotely sensed data," International Journal of Remote Sensing, vol. 22, no. 16, pp. 3223–3235, 2001.

[19] N. Dhanachandra, K. Manglem, and Y. J. Chanu, "Image segmentation using k-means clustering algorithm and subtractive clustering algorithm," Procedia Computer Science, vol. 54, pp. 764–771, 2015.

[20] V. S. Nalwa, A guided tour of computer vision. Addison-Wesley Longman Publishing Co., Inc., 1994.

[21] R. Szeliski, Computer vision: algorithms and applications. Springer Science & Business Media, 2010.

[22] P. Soille, Morphological image analysis: principles and applications. Springer Science & Business Media, 2013.

[23] H. Ng, S. Ong, K. Foong, P. Goh, and W. Nowinski, “Medical image segmentation using k-means clustering and improved watershed algorithm,” in *2006 IEEE* Southwest Symposium on Image Analysis and Interpretation. IEEE, 2006, pp. 61–65.

[24] X. Peng, L. Wang, X. Wang, and Y. Qiao, “Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice,” *Computer Vision and Image Understanding*, vol. 150, pp. 109–125, 2016.

[25] H. Bay, T. Tuytelaars, and L. Van Gool, “Surf: Speeded up robust features,” in *European conference on computer vision*. Springer, 2006, pp. 404–417.

26] A. P. Bradley, “The use of the area under the roc curve in the evaluation of machine learning algorithms,” *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[27] T. Fawcett, “An introduction to roc analysis,” Pattern recognition letters, vol. 27, no. 8, pp. 861–874, 2006.

[28] L. v. d. Maaten and G. Hinton, “Visualizing data using t-sne,” *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[29] Rhody,H. ”Lecture 10: Hough Circle Transformation”. 11 Oct 2005.Web. 15 Mar 2016. [En línea]. Disponible:hhttps://www.cis.rit.edu/class/simg782/lectures/lecture 10/lec782 05 10. pdfi

[30] Rizon, M.,” Object Detection Using Circular Hough Transform”.,15 Mar 2016. [En línea]. Disponible:

hhttp://dspace.unimap.edu.my/xmlui/bitstream/handle/123456789/11349/ajas2121606-1609.pdf? sequence=1i

[31] Matlab and Image Processing Toolbox R2016a. ”Identifying Objects”. 2 Mar 2016. [En línea]. Disponible:

hhttp://www.mathworks.com/help/images/examples/identifying-round-Objects.html?refresh=truei

[32] Poly, D.” Geometric Morphometrics”.2012. [En línea]. Disponible:

hhttp://www.indiana.edu/~g562/Lectures/Lecture%201%200Introduction%20to%20Geometric%20Morphometrics.pdfi.

[33] Universidad Nacional de Quilmes.” Segmentación por UmbralizaciónMétodo de Otsu”. [En línea]. Disponible:

hhttp://iaci.unq.edu.ar/materias/vision/archivos/apuntes/Segmentacin%20por%20umbralizacin%20-%20Mtodo%20de%20Otsu.pdfi

[34] Utrecht University. ”Mathematical Morphology”. [En línea]. Disponible:

http://www.cs.uu.nl/docs/vakken/ibv/reader/chapter6.pdf.

[35] Escolano, F., Cazorla, M., Galipienso, M., Pardo, O., y Lozano, M., (2003). Inteligencia Artificial. Modelos, Técnicas y Áreas de Aplicación. Madrid, España: Paraninfo.

[36] Technology Robotix Society. Tutorial: Shape Detection. [En línea].Disponible: http://www.robotix.in/tutorials/categ/opencv/shape_detection.

[37] Rojas, T., Sanz, W., y Arteaga F. (2008). “Sistema de visión por computadora para la detección de objetos esféricos a través de la transformada de Hough”. Revista Ingeniería UC, 15 (1), 77-87.

[38] Atiquzzaman, M. “Multiresolution Hough transform: an efficient method of detecting patterns in images”. IEEE Transactions on pattern analysis and machine intelligence. 1992, 14(11), 1090-1095.

[39] Zhu, T., Jeong-Hyun, K., Dong-Joong, K. “Ellipse detection: a simple and precise method based on randomized Hough transform. Optical Engineering”. 2012, 51(5).

[40] Duda, R.O., Hart, P.E. “Use of the Hough transform to detect lines and curves in pictures”. Commun. ACM. 1972, 15(1), 11-15.

[41] Hanahara, K., Matuyama, T., Uchiyama, T. A real-time processor for the Hough transform. IEEE “transactions on pattern analysis and machine intelligence”. 1988, 10(1), 121-125.

[42] R. C. Gonzalez, & R. E. Wood, “Digital image processing”, Upper Saddle River, New Jersey: Prentice Hall, 2002, pp. 19

[43] L. v. d. Maaten and G. Hinton, “Visualizing data using t-sne,” Journal of machine learning research, vol. 9, no. Nov, pp. 2579–2605, 2008.

[44] Gonzalez, R. Digital Image Processing Using Matlab. New Jersey:Prentice Hall, 2002, pp. 519-547.

[45] R. C. Gonzalez, & R. E. Wood, “Digital image processing”, Upper Saddle River, New Jersey: Prentice Hall, 2002, pp. 51-54.

[46] Gonzalez, R. Digital Image Processing Using Matlab. New Jersey:Prentice Hall, 2002, pp. 39.

[47] Instituto tecnológico de Costa Rica. ”Mathematical Morphology”. [En línea]. Disponible:

http://www.ie.tec.ac.cr/palvarado/PAID/paid.pdf.

[48] Escuela Politécnica de Litoral. ”Implementación de un algoritmo para la detección de objetos ”. [En línea]. Disponible:

https://www.dspace.espol.edu.ec/bitstream/123456789/8375/1/Implementaci%C3%B3n%20de%20un%20algoritmo%20para%20la%20detecci%C3%B3n%20y%20conteo%20de%20c%C3%A9lulas%20en%20im%C3%A1genes%20microsc%C3%B3picas.pdf

[49] Utrecht University. ”Mathematical Morphology”. [En línea]. Disponible:

http://www.cs.uu.nl/docs/vakken/ibv/reader/chapter6.pdf.

[50] Coursera.org/: Shape Detection de objetos. [En línea]. Disponible:

https://es.coursera.org/lecture/deteccion-objetos/l1-4-componentes-conexas-zsHxk

[51] Gonzalez, R. Digital Image Processing Using Matlab. New Jersey:Prentice Hall, 2002, pp. 403-410

[52] Reporte de busqueda: “Detection y Conteo” [En línea]. Disponible:

https://www.cimat.mx/~alram/VC/ramirez_segmObjDetecc.pdf

[53] Rizon, M.,” Object Detection Using Circular Hough Transform”.,15 Mar 2016. [En línea]. Disponible:

hhttp://dspace.unimap.edu.my/xmlui/bitstream/handle/123456789/11349/ajas2121606-1609.pdf? sequence=1i

[53] N. Dhanachandra, K. Manglem, and Y. J. Chanu, “Image segmentation using k-means clustering algorithm and subtractive clustering algorithm,” Procedia Computer Science, vol. 54, pp. 764–771, 2015.

[54] V. S. Nalwa, A guided tour of computer vision. Addison-Wesley Longman Publishing Co., Inc., 1994.

[55] R. Szeliski, Computer vision: algorithms and applications. Springer Science & Business Media, 2010.

[56] P. Soille, Morphological image analysis: principles and applications. Springer Science

& Business Media, 2013.

[57] H. Ng, S. Ong, K. Foong, P. Goh, and W. Nowinski, “Medical image segmentation using k-means clustering and improved watershed algorithm,” in *2006 IEEE* Southwest Symposium on Image Analysis and Interpretation. IEEE, 2006, pp. 61–65.

[58] X. Peng, L. Wang, X. Wang, and Y. Qiao, “Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice,” Computer Vision and Image Understanding, vol. 150, pp. 109–125, 2016.

[59] H. Bay, T. Tuytelaars, and L. Van Gool, “Surf: Speeded up robust features,” in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[60] A. P. Bradley, “The use of the area under the roc curve in the evaluation of machine learning algorithms,” Pattern recognition, vol. 30, no. 7, pp. 1145–1159, 1997.

[61] T. Fawcett, “An introduction to roc analysis,” Pattern recognition letters, vol. 27, no. 8, pp. 861–874, 2006.

ORDEN DE EMPASTADO