Desarrollo de un Software para la Corrección de Segmentación de Células Endoteliales en Imágenes de Microscopía Especular en Córnea Guttata

---

Juan Sebastián Sierra Bravo

Universidad Tecnológica de Bolívar

Facultad de Ingeniería
Programa de Ingeniería Mecatrónica

Trabajo de grado

Desarrollo de un Software para la Corrección de Segmentación de Células Endoteliales en Imágenes de Microscopía Especular en Córnea Guttata

Juan Sebastián Sierra Bravo

1. Revisor Sonia H. Contreras, Ph.D.
Facultad de Ingeniería
Universidad Tecnológica de Bolívar

2. Revisor Gloria I. Bautista, M.Sc.
Facultad de Ingeniería
Universidad Tecnológica de Bolívar

Supervisores Andrés G. Marrugo, Ph.D. y Lenny A. Romero, Ph.D.

Juan Sebastián Sierra Bravo
Desarrollo de un Software para la Corrección de Segmentación de Células Endoteliales en Imágenes de Microscopía Especular en Córnea Guttata
Trabajo de grado,
Revisores: Sonia H. Contreras, Ph.D. y Gloria I. Bautista, M.Sc.
Supervisores: Andrés G. Marrugo, Ph.D. y Lenny A. Romero, Ph.D.

Universidad Tecnológica de Bolívar
Programa de Ingeniería Mecatrónica
Facultad de Ingeniería
Cartagena, Bolívar

Resumen

Es una práctica común en la oftalmología el hacer el estudio del estado de la córnea en los pacientes mediante la microscopía especular. El objetivo de dicho estudio es, principalmente, conocer la densidad celular (CD, por sus siglas en inglés), de manera que el profesional encargado pueda dar un diagnóstico de la salud de la córnea del paciente en cuestión. Realizar esta tarea manualmente resulta tedioso y consume mucho tiempo, razón por la cual en muchas clínicas se ha optado por usar softwares capaces de medir la densidad celular de manera automática o semi-automática (guiada por el operador).

Sin embargo, distintas enfermedades pueden afectar a la córnea y dificultar la evaluación automática mediante software. En este trabajo tratamos específicamente las imágenes de pacientes con córnea guttata. Para esto desarrollamos un software que permite corregir la segmentación de células en imágenes de microscopía especular. Este ha demostrado ser robusto y flexible a las distintas necesidades del especialista en estos casos.

Abstract

It is a common practice in ophthalmology to study the state of the cornea in patients using specular microscopy. The objective of this kind of study is, mainly, to determine the cell density (CD), so that the physician can give a diagnosis of the health of the patient's cornea. To manually perform this task is tedious and time consuming, which is why in many clinics has chosen to use softwares capable of measuring cell density automatically or semi-automatically (guided by the operator).

However, different diseases can affect the cornea and make it difficult to evaluate automatically using software. In this work we specifically treat the images of patients with guttate cornea. For this we developed a software that allows to correct the segmentation of cells in images of specular microscopy. The software has proven to be quite robust and flexible to the different needs of the specialist in these cases.

Agradecimientos

Quiero agradecer a mis padres, Enrique Sierra y Yuly Bravo, y a mis hermanos Enrique y Valentina, quienes me han brindado su apoyo incondicional en cada una de las etapas tanto en el desarrollo de mi carrera profesional, como en el ámbito personal; también agradezco a mis supervisores de trabajo de grado, los profesores Andrés Marrugo y Lenny Romero, quienes estuvieron siempre al pendiente de mi proyecto, brindándome apoyo académico y moral.

Agradezco a la Fundación Oftalmológica de Santander FOSCAL, en particular a los doctores Alejandro Tello y Virgilio Galvis por facilitarnos las imágenes de prueba para el desarrollo de este trabajo. También a la profesora María Millán de la Universidad Politécnica de Cataluña (UPC) por sus consejos y al Centro de Cooperación y Desarrollo de la UPC bajo el proyecto CCD2018-U005 en el que ha estado enmarcado este trabajo.

Por último, agradezco a mis compañeros Daniel Pajaro y Alvaro Ortega, quienes me introdujeron al lenguaje Python y al diseño de interfaces gráficas en el mismo.

Índice general

Índice de figuras

1 Introducción

El estudio del estado de la córnea en los pacientes mediante la microscopía especular es una práctica común en la oftalmología. Para este estudio se utiliza el microscopio especular, el cual es un instrumento que permite capturar imágenes de las células de la córnea. A partir de estas imágenes es posible obtener parámetros como el número de células, la densidad celular y tamaños de células (Bourne y Kaufman, 1976). Tomando estos datos como criterio, el especialista realiza una valoración del estado de la córnea y da un diagnóstico final, para asignar tratamientos al paciente de ser necesario.

1.1 Estado del arte

Diferentes metodologías se han planteado para la segmentación de las células en imágenes de microscopía especular. Luc Vincent y Barry Masters propusieron en 1992 utilizar la transformación de sombrero de copa y binarizar la imagen de entrada para obtener los bordes de las células (Vincent y Masters, 1992). Este método resultó tener ciertos defectos, por lo que posteriormente emplearon otro más eficiente pero más complejo, utilizando segmentación de línea divisoria.

En 1999 Francisco J. Sanchez-Marin, desarrolló un método completamente automático para la segmentación de contornos de células endoteliales corneales (Sanchez-Marin, 1999). En este trabajo la imagen de entrada es realzada mediante la extracción de la imagen suavizada por un filtro gaussiano de la imagen original, logrando de esta manera distinguir mejor entre el área las células y el área que hay entre ellas, posterior a ello, el contraste es aumentado extendiendo el histograma de la imagen, y por último esta es binarizada, obteniendo de esta manera la segmentación de las células.

En 2005 investigadores crearon un software de detección de contornos para analizar la morfometría de las células en microscopia especular (Giasson y col., 2005). El método utilizado para la segmentación consiste en convolucionar la imagen de entrada con un *sombrero Mexicano* (un tipo de kernel gaussiano), el cual combina un realce de los bordes y un filtro de paso bajo para hacer más uniforme la luminosidad

de la imagen; y luego binarizar y erosionar la imagen para disminuir el espacio entre las regiones.

Nótese que los trabajos mencionados, a diferencia del presentado en este documento, no tiene en cuenta las córneas con patologías (córnea guttata), sino que únicamente trabajan en base a imágenes de córneas saludables. Por lo tanto, en este proyecto se buscará plantear una metodología para desarrollar un software que permita corregir la segmentación de las células del endotelio corneal incluyendo en los cálculos dicha afección de la córnea; esto como una primera aproximación a la problemática.

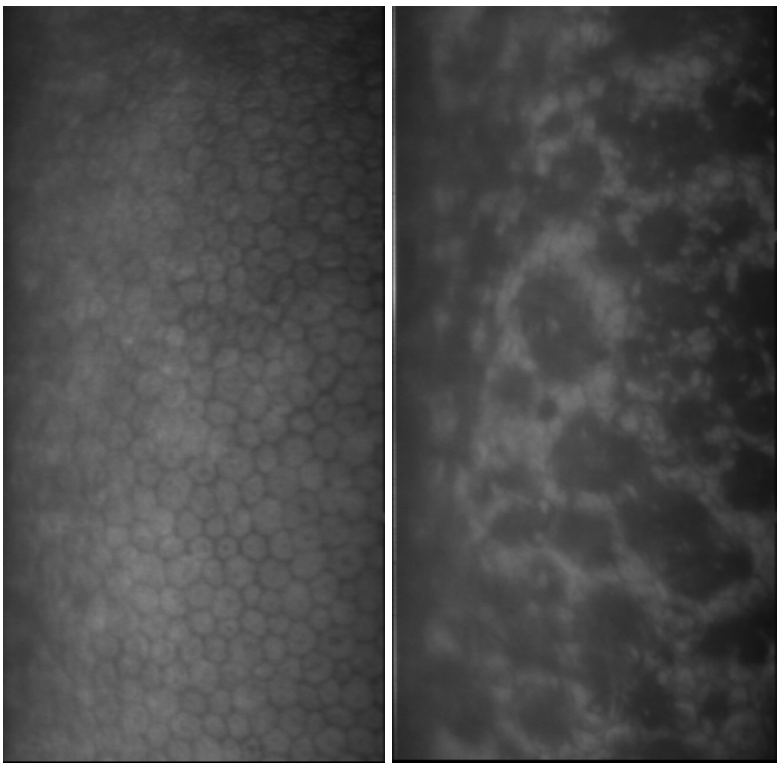
1.2 Planteamiento del problema

Una de las afecciones de la córnea que puede producir resultados no deseados en la segmentación de células, es la córnea guttata. Esta es una afección del endotelio, la capa más profunda de la córnea, cuyas células son las encargadas de mantener transparencia corneal (Oftalmología Barran, s.f.). Dicha afección puede ser causada por factores como el envejecimiento, drogas, cirugías e inflamaciones (Maruoka y col., 2018). Esta condición presenta irregularidades en la distribución de las células del endotelio y por consiguiente en la imagen obtenida por el microscopio. En la Figura 1.1 se puede observar la diferencia entre la imagen de microscopía especular de un paciente sano (Figura 1.1a) y un paciente con córnea guttata (Figura 1.1b); donde se puede notar que en la figura de la izquierda están bien definidas las células del endotelio, con una estructura tipo panal de abeja, mientras que en la figura de la derecha las células con forma regular se encuentran aisladas por regiones, además se logran apreciar áreas negras en la misma, lo que corresponde a células muertas o guttas.

El efecto de la córnea guttata en las imágenes de microscopía especular hace necesario desarrollar nuevas metodologías para que el proceso de segmentación de células o obtención de parámetros sea más robusto.

Este trabajo es realizado en colaboración con la clínica oftalmológica FOSCAL, ubicada en Bucaramanga, Colombia, la cual proporcionó las imágenes que se utilizarán en este trabajo. Esta clínica utiliza el microscopio especular SP-3000P para obtener imágenes de la córnea de los pacientes y posteriormente conocer datos como la densidad celular (en número de células por milímetro cuadrado), el área de la célula más grande y más pequeña (en micrómetros), entre otros parámetros.

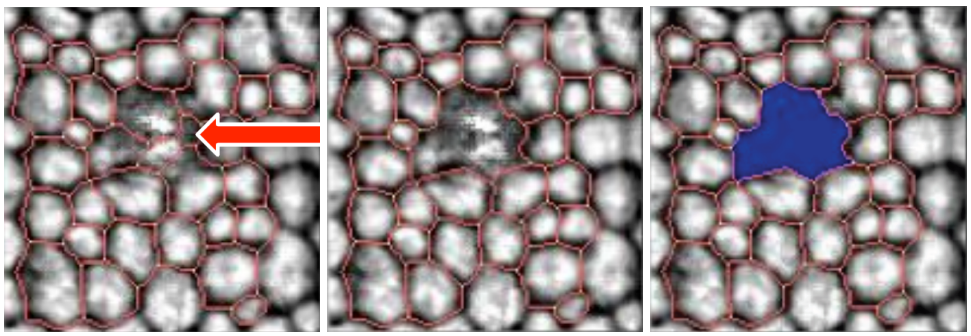
El software encargado de realizar la segmentación celular implementado en el microscopio SP-3000P realiza un mal conteo de células, y por lo tanto de densidad



(a) (b)

Figura 1.1: Dos imágenes de microscopía especular. *(a)* Paciente sano. *(b)* Paciente con córnea guttata.

celular, si la córnea del sujeto es poco saludable y contiene guttas. Cuando se presentan estos casos, el software del microscopio en algunas ocasiones identifica células dentro de las guttas. Un ejemplo de esta situación se puede observar en la Figura 1.2a, donde las líneas rojas corresponden a los bordes de las segmentaciones obtenidas. En estos casos, el usuario tiene la opción de implementar herramientas para eliminar manualmente los bordes que no se desean, y así hacer nuevamente el cálculo de tamaños y conteo de células y de densidad celular. En la Figura 1.2b se muestra el resultado de eliminar los bordes erróneos de la figura 1.2a . Usar este método para "corregir" el cálculo tiene una particularidad: las guttas son tomadas como células, y por lo tanto los cálculos siguen siendo erróneos. Otra herramienta que tiene el usuario, le permite seleccionar las áreas que no corresponden a células y eliminarlas por completo de los cálculos, como se muestra en la Figura 1.2c, lo cual da como resultado un buen conteo celular, pero una densidad celular mayor a la real, puesto que el área que se está teniendo en cuenta es menor. Por esta razón, nos propusimos desarrollar una herramienta que permita corregir la segmentación y obtener cálculos correctos, sin tomar erróneamente guttas como células y utilizando el valor real de área.



(a) (b) (c)

Figura 1.2: Caso de segmentación errónea de células endoteliales en imagen de microscopía especular. Las líneas rojas corresponden a los bordes de las segmentaciones obtenidas. *(a)* Gutta con segmentaciones internas que no corresponden a verdaderas células. La flecha señala la gutta. *(b)* Bordes erróneos eliminados. *(c)* Región de gutta resaltada en azul para ser excluida de los cálculos de parámetros.

1.3 Estructura y alcance del documento

El documento consta de seis capítulos. En el capítulo 2 del documento se explican los fundamentos de procesamiento de imágenes a tener en cuenta para el desarrollo del software. El capítulo 3 habla de la metodología llevada a cabo para la corrección de conteo celular. En el capítulo 4 se explica la implementación del software en Python. El capítulo 5 muestra los resultados obtenidos en este trabajo. Por último, las conclusiones son mencionadas en el capítulo 6.

2 Fundamentos de procesamiento de imágenes

Para el desarrollo del software fue esencial tener conocimientos de conceptos básicos de procesamiento de imágenes. Por esta razón, en esta sección se describirán las operaciones de procesamiento de imágenes utilizadas, así como también las características de algunos tipos de imágenes que son de interés para el trabajo aquí planteado.

2.1 Imagen digital

Una imagen digital es un arreglo bidimensional compuesto por un número finitos de elementos, los cuales cada uno consta de una posición y un valor llamado *intensidad*, o en imágenes monocromáticas, *nivel en escala de grises*; estos elementos son comúnmente llamados *píxeles*, aunque también se les puede referir como *elementos de imagen* (Gonzalez, 2004). Cabe mencionar que los elementos que componen las imágenes monocromáticas, o en escala de grises, tienen valores comprendidos entre 0 y 255, siendo el primero el más oscuro y el último el más claro.

En la Figura 2.1 se muestra en la parte izquierda la imagen digital *cameraman.tif* (imagen digital en escala de grises) obtenida de MATLAB, y a la derecha recorte de 11x11 píxeles de la misma, situada sobre su respectiva representación matricial.

Las imágenes a color están compuestas por la combinación de imágenes monocromáticas. Como ejemplo tenemos las imágenes RGB, las cuales se crean a partir de la combinación de tres canales llamados *rojo* (R), *verde* (G) y *azul* (B) los cuales contienen imágenes monocromáticas, lo que hace posible que las técnicas de procesamiento de imágenes puedan ser usadas en imágenes a color, trabajando individualmente sobre cada una de los canales que la conforman (Gonzalez, 2004).

El procesamiento de imágenes digitales se realiza mediante softwares computacionales especializados. Una imagen digital se almacena en forma de matriz en el computador, y sobre esta se pueden hacer operaciones como realzar la imagen para su visualización o para obtener medidas y extraer características, como es el caso del conteo de células endoteliales. El procesamiento de este tipo de imágenes tiene

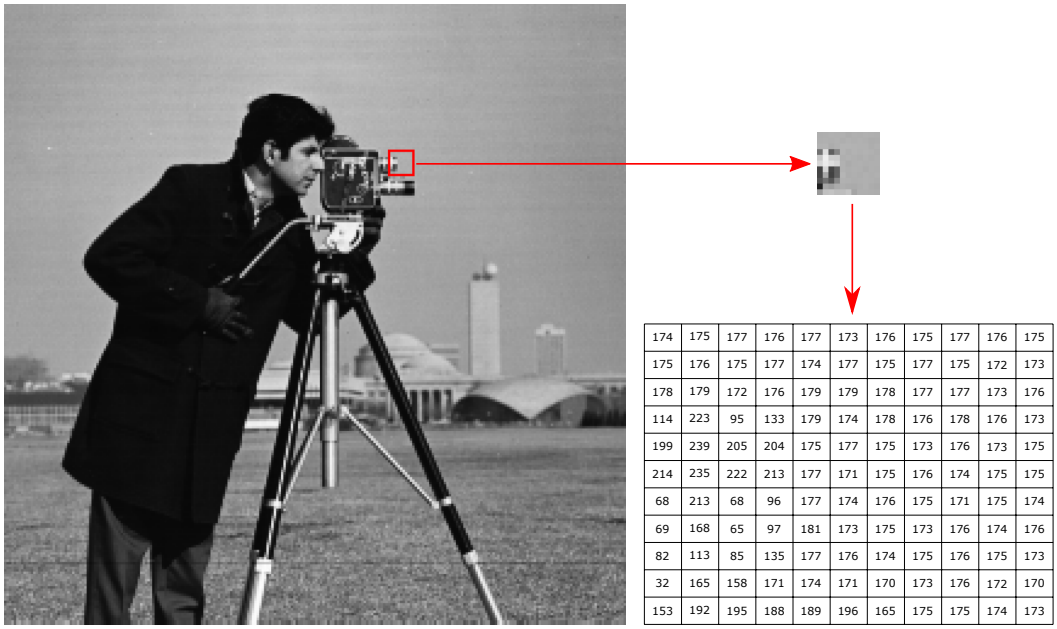


Figura 2.1: Imagen digital y representación como matriz.

aplicaciones, por ejemplo, en el campo de la visión artificial, el cual intenta simular la visión humana, y abarca una rama de lo que se conoce como *inteligencia artificial* (Gonzalez, 2004).

El procesamiento de imágenes se basa en realizar operaciones matemáticas entre matrices (arreglos bidimensionales, o imágenes), y tiene como objetivo analizar los elementos que la conforman de manera computarizada para obtener información interés de las mismas.

Existen además otro tipo de imágenes llamadas *binarias*, las cuales poseen ciertas características que permiten usar funciones especiales de procesamiento de imágenes. Este tipo de imágenes será expuesto en la Sección 2.2.

2.2 Propiedades de imágenes binarias

Una imagen binaria está compuesta por píxeles únicamente con valores de cero (negro) y uno (blanco). Las imágenes en escala de grises pueden ser transformadas en binarias, definiendo un valor umbral entre 0 y 255, con el cual se compararán todos los elementos que componen a la imagen, convirtiendo en 1 los que estén por encima de este valor, y en 0 todos los demás.

Este tipo de imágenes están conformadas por una o más regiones blancas, como se muestra en la Figura 2.2, donde se pueden apreciar 4 regiones circulares, cada una con sus respectivas características.

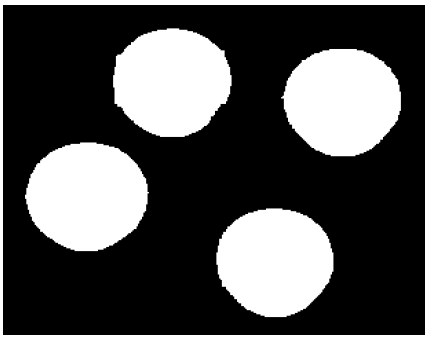


Figura 2.2: Imagen binaria.

Estas regiones tienen propiedades especiales con las que es posible trabajar el procesamiento de imágenes. Dichas propiedades son:

- **Área:** Espacio que ocupa la región sobre la imagen. Se mide en píxeles y está dado por la fórmula

$$A = \sum_{i=1}^{n} m_i \ , \tag{2.1}$$

  donde A es el área; n es la cantidad de píxeles en la región; y m_i es el valor de intensidad del píxel.

- **Centroide:** Se define como la media aritmética de las posiciones de todos los puntos que conforman la figura (Protter y Morrey, 1977). Las coordenadas del centroide se calculan con las fórmulas

$$X_c = \frac{1}{M} \sum_{i=1}^{n} x_i m_i \ , \tag{2.2}$$

$$Y_c = \frac{1}{M} \sum_{i=1}^{n} y_i m_i \ , \tag{2.3}$$

  donde X_c y Y_c son las coordenadas del centroide en los ejes X y Y, respectivamente; n es la cantidad de píxeles en la región; x_i y y_i representan la localización del píxel horizontal y verticalmente, respectivamente; m_i es el valor de intensidad del píxel; y M es la sumatoria de las intensidades m_i.

- **Bounding box:** O cuadro delimitador, es el rectángulo más pequeño que cubre un área definida. Este es obtenido a partir de un vector de 4 posiciones *[x y a b]*, donde las dos primeras posiciones indican la coordenada de la esquina superior izquierda del rectángulo, y las otras dos determinan el ancho y alto del mismo, como se puede apreciar en la Figura 2.3, donde se muestra el cuadro delimitador de una región blanca en una imagen binaria.

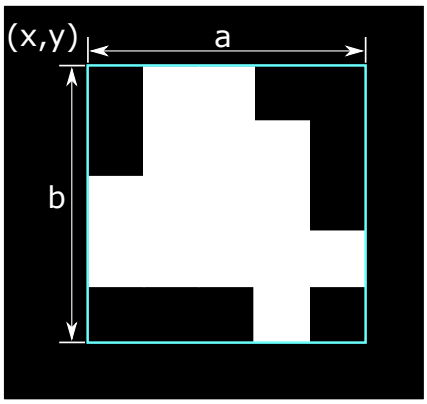


Figura 2.3: Bounding box de una región.

Gracias a las propiedades mencionadas, es posible realizar con estas imágenes operaciones conocidas como *operaciones de morfología matemática* (véase Sección 2.3).

2.3 Operaciones de morfología matemática en imágenes binarias

La morfología matemática se refiere a herramientas usadas en el procesamiento de imágenes para extraer componentes de las mismas que son útiles en el análisis de las características de las regiones. Algunas de estas operaciones usan un elemento estructurante. Esto es una matriz binaria de pequeñas dimensiones, la cual se desplazará sobre la imagen, y dependiendo de su forma determinará cuáles píxeles tenidos en cuenta en la operación.

Entre las operaciones de morfología matemática en imágenes binarias usadas en este trabajo encontramos:

- **Dilatación:** Es una operación morfológica que expande o hace más grueso un objeto (área blanca) en una imagen binaria. La manera en que el objeto se dilata es determinado por el elemento estructurante. La dilatación de un objeto A por un elemento estructurante B está definida por la operación (Gonzalez, 2004)

$$A \oplus B = \{z|(\widehat{B})_z \cap A \neq \varnothing\} \ , \tag{2.4}$$

donde $\varnothing$ es un conjunto vacío y $\widehat{B}$ es el reflejo de B. Esto quiere decir que la dilatación de A por B es el conjunto de todos las ubicaciones donde la versión reflejada de B se superpone con al menos un elemento del objeto A.

- **Erosión:** Esta operación, contrario a la dilatación, reduce o adelgaza un objeto en una imagen binaria. La erosión de un objeto A por un elemento estructurante B está definida por la operación (Gonzalez, 2004)

$$A \ominus B = \{z|(B)_z \subseteq A\} \ , \tag{2.5}$$

donde la notación $C \subseteq D$ significa que C es un subconjunto de D. Esto quiere decir que la erosión de A por B es el conjunto de todas las posiciones z en las que B es contenido por A. Dicho de otra manera, corresponde a las posiciones de B en las cuales no comparte ningún elemento con el fondo (regiones con valor 0), lo cual se puede expresar con la ecuación

$$A \ominus B = \{z|(B)_z \cap A^c = \varnothing\} \ . \tag{2.6}$$

- **Closing:** Consiste en aplicar una dilatación seguida de una erosión. Esta operación elimina las regiones negras (con valor 0) en la imagen binaria que son más pequeñas que el elemento estructurante. La operación *closing* de un objeto A por un elemento estructurante B está dada por la ecuación

$$A \bullet B = (A \oplus B) \ominus B \ . \tag{2.7}$$

En la Figura 2.4 se muestran los resultados obtenidos al realizar las operaciones morfológicas expuestas previamente sobre una misma imagen con un elemento estructurante en forma de disco, con radio de 5 píxeles. Nótese que al dilatar la imagen, las regiones blancas se expanden, al contrario que al realizar la erosión, donde estas se contraen. Por último, al realizar el closing, las regiones cercanas se unen, manteniendo su tamaño original.

- **Bwlabel:** Esta operación tiene como parámetros de entrada una imagen binaria y un valor (4 u 8) que establece la conectividad. Este último define cómo se

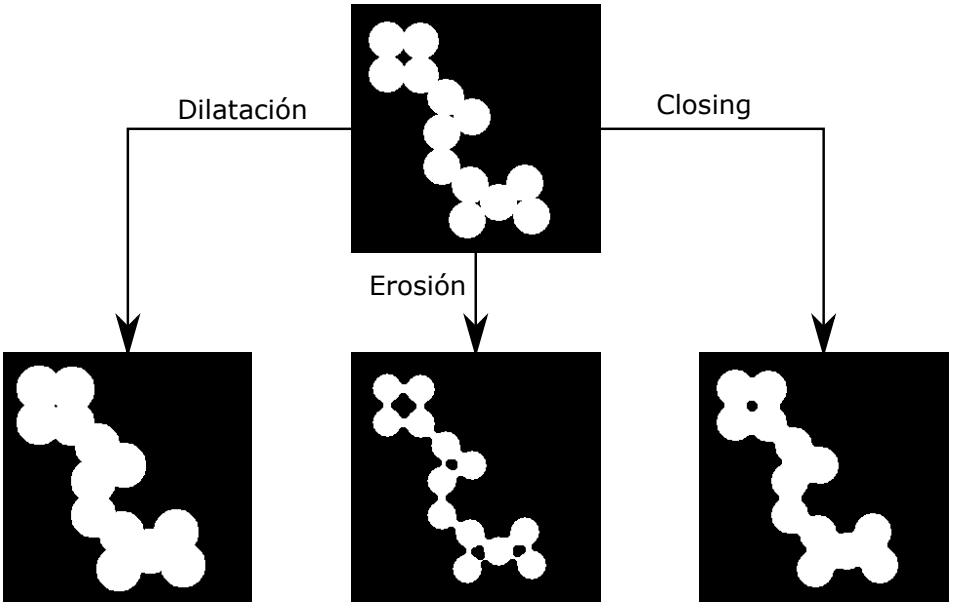


Figura 2.4: Resultados al realizar distintas operaciones morfológicas sobre la misma imagen, con un elemento estructurante en forma de disco.

analizarán los elementos de la imagen; el valor de 4 (4-conexión) sólo tendrá en cuenta los píxeles que se encuentran al norte, sur, este y oeste del elemento que está siendo analizado, mientras que con el valor de 8 (8-conexión) se realiza el proceso con los 8 vecinos del mismo.

La operación tiene como objetivo crear una imagen de etiquetas, enumerando cada una de las regiones blancas encontradas en la imagen de entrada, haciendo el recorrido por columnas o por filas. Dos píxeles vecinos con valor de 1 tienen la misma etiqueta dependiendo de la conectividad establecida.

En la Figura 2.5 se puede apreciar un ejemplo de una entrada (Figura 2.5a), donde se observan algunas regiones separada, y los resultados obtenidos a partir de esta función, haciendo uso de la 4-conexión (Figura 2.5b) y la 8-conexión (Figura 2.5c), entre los cuales, la diferencia es que en el primero los elementos unidos sólo por una esquina no hacen parte de la misma región, al contrario que en el segundo, donde estos sí se incluyen en la misma región.

2.4 Crecimiento de regiones

El crecimiento de regiones es una técnica de segmentación que consiste en crear regiones partiendo de puntos o *semillas* en una imagen digital en escala de grises. Estas regiones crecen al añadir píxeles que cumplen un criterio predefinido de intensidad a las semillas (Gonzalez, 2004). Una vez obtenidas las regiones con

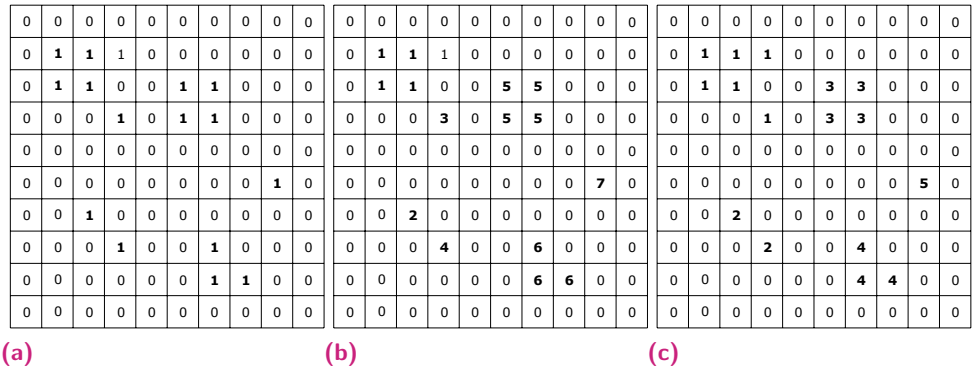


Figura 2.5: Ejemplo de uso de la función *bwlabel*. *(a)* Matriz que contiene la imagen de entrada. *(b)* Resultado implementando la 4-conectividad. *(c)* Resultado implementando la 8-conectividad.

este método de segmentación, es posible determinar sus áreas, así como otras propiedades.

Existen dos métodos comúnmente usados para realizar el crecimiento de regiones. Para comprender ambas alternativas, supongamos que i es el valor de intensidad en escala de grises del píxel seleccionado, y está ubicado en la posición (x_i, y_i); t es un valor de la tolerancia; y j es la matriz que contiene la imagen.

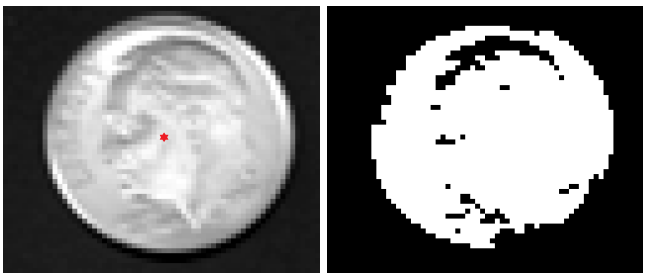
1. El primer método consiste en seleccionar un píxel, analizar los 8 vecinos y determinar cuáles poseen un valor de intensidad que se encuentre dentro del rango de tolerancia $i - t <= i <= i + t$. Aquellos que estén dentro del rango, harán parte de la región en crecimiento, y serán analizados del mismo modo, cambiando únicamente el valor de intensidad *i* por el correspondiente al píxel en cuestión. De esta manera, la región se expanderá hasta cubrir el área circundante con valores de intensidad similares al inicial.

   En la Figura 2.6a se muestra un ejemplo de una imagen de entrada cuyo píxel semilla está ubicado cerca del centro de la misma, marcado con un punto rojo. Al realizar el crecimiento de regiones con este método, se obtiene la Figura 2.6b, donde se aprecia que la moneda no fue completamente abarcada, debido a las diferencias de valores de intensidad dentro del objeto.

2. Para el segundo método, usaremos la ecuación (Gonzalez, 2004)

$$T = |j - i| <= t \ , \tag{2.8}$$

   para obtener una imagen binaria T en la que los píxeles con valor de 1 corresponden a las posiciones en las que la matriz j tiene un valor dentro del



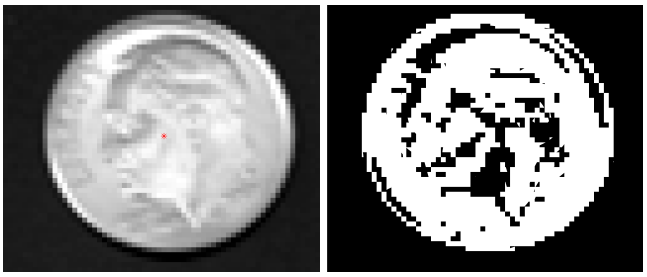
(a) (b)

Figura 2.6: Método 1 de crecimiento de regiones. *(a)* Píxel semilla sobre la imagen. *(b)* Resultado del crecimiento de regiones.

rango $i - t <= i <= i + t$, por lo que posiblemente hagan parte del mismo objeto.

Teniendo la matriz T, se aplica la función *bwlabel*, de donde se extraerán sólo los píxeles en los que la etiqueta es igual al valor de la del píxel en la posición (x_i, y_i). De este modo nos quedaremos con la región que abarca el objeto seleccionado.

La Figura 2.7a muestra el origen del crecimiento de región sobre la imagen, a partir del cual se analizarán los elementos cercanos. En la Figura 2.7b se muestra la región obtenida luego del proceso, nótese que aunque no identifica la totalidad del área del objeto, mantiene su forma, por lo que mediante morfología matemática se puede obtener el área deseada. Cabe resaltar que el resultado varía si se cambia el valor de tolerancia T.



(a) (b)

Figura 2.7: Método 2 de crecimiento de regiones. *(a)* Píxel semilla sobre la imagen. *(b)* Resultado del crecimiento de regiones.

3 Metodología

El objetivo de este trabajo es crear un software que le permita al médico oftalmólogo encargado de llevar a cabo el procedimiento de captura de imagen de las células corneales al paciente usando el microscopio especular, realizar modificaciones a la segmentación calculada por el software del microscopio, para lograr que los parámetros calculados sean más precisos.

Inicialmente, con el fin de realizar pruebas haciendo uso de las operaciones morfológicas para desarrollar un software que brinde herramientas para modificar la segmentación celular calculada por el microscopio especular, se creó una GUI (Graphical User Interface) en MATLAB, un sistema de cómputo numérico que trabaja en base a matrices, por lo que brinda cierta facilidad para trabajar con este tipo de imágenes. Posterior a ello, debido a que MATLAB requiere de licencia para su uso, nuestro software será implementado en en lenguaje Python, el cual es un lenguaje de código abierto, rápido y capaz de ejecutar un programa sin necesidad de licencias.

Este software tiene como entrada un archivo .TIF de dos canales (el primero es una imagen en escala de grises, y el segundo una imagen de 16 niveles), el cual es obtenido por el microscopio especular. El software desarrollado en este trabajo le ofrece al usuario algunas opciones con la capacidad de realizar modificaciones a la segmentación para corregir ciertas imperfecciones y obtener datos más acertados.

3.1 Captura de imagen

El proceso de captura de la imagen de la córnea y análisis de la misma, se hace mediante el microscopio especular SP-3000P. Para llevar a cabo este proceso, una vez el paciente ubique el ojo frente al lente de la cámara, el médico operará el microscopio siguiendo las fases mostradas en la Figura 3.1 (Corporation, s.f.).

1. **Alineación:** El médico seleccionará entre tres tipos de alineación: automática, semi-automática y manual. Aquí se debe llevar el centro del segmento anterior ocular, que es la tercera parte frontal del ojo e incluye la córnea, el iris, el cuerpo ciliar y el cristalino (Cassin y col., 1984), al centro de la pantalla.

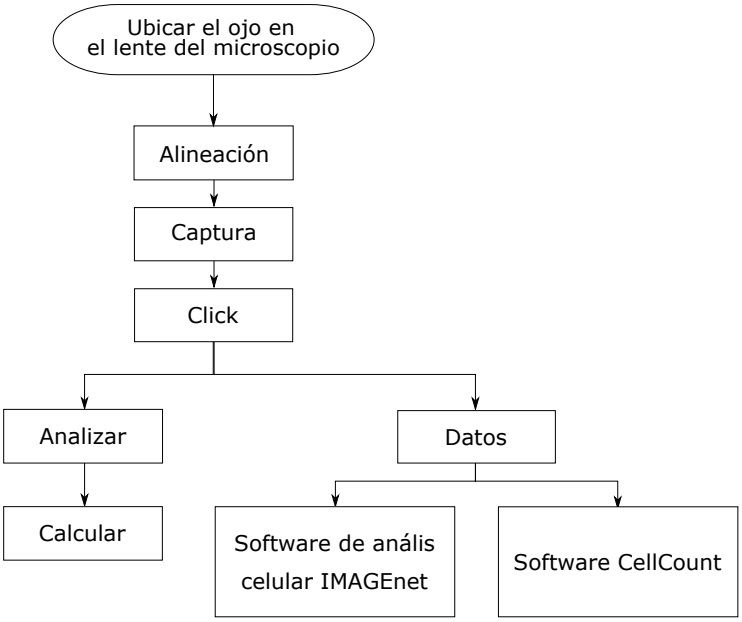


Figura 3.1: Proceso para la captura de imágenes.

2. **Captura:** Una vez ubicada correctamente la imagen, el usuario hará la captura de la misma.

3. **Click:** Aparece a un costado de la pantalla una sección de la imagen que contiene las células de la córnea, a partir de la cual se le dan dos opciones al operario:

    - **Análisis.** En pantalla se observan los centros de algunas de las células en la imagen, y da la opción de calcular, mediante la cual en cuestión de algunas decenas de segundos es obtenida la información referente a cantidad de células, tamaño mínimo, máximo y promedio, densidad celular, desviación estándar, coeficiente de valor y hexagonalidad.

    - **Datos.** Para obtener los datos en archivos guardables en la computadora, se puede usar el software de análisis celular *IMAGEnet*, el cual adquiere información mejorada sobre la capa de células endoteliales, o el sofware *CellCount*, que realiza análisis altamente precisos de la capa de células endoteliales y es totalmente automático.

En la Figura 3.2 (Corporation, s.f.) se pueden apreciar las características técnicas del microscopio especular SP-3000P, usado por la clínica para este proceso y de la imagen capturada, donde encontramos información del área de la imagen, distancia de trabajo, también características físicas del equipo, como el peso, tamaño del monitor, entre otras.

Cobertura fotografica	0.25 x 0.5 mm
Captura de aumento	150 x
Medición de Paquimetría	paso de 0.01 mm
Distancia de trabajo	25 mm
Intensidad de flash	Alto / Bajo
Objetivo de fijación	1 Central / 4 Periférico(S,T,I,N/12,2,6,10 en punto)
Modo foto	Automático/Semi-automático/Manual
Memoria de imagen	5 imágenes por cada ojo (total 10 imágenes)
Monitor	5.6” LCD a Color
Recorrido de base	X ± 46 mm, Y ±14 mm, Z ± 20 mm
Recorrido de cabeza	X ± 10 mm, Y ± 14 mm, Z ± 10 mm
Ajuste de la barbilla	Motorizado 65mm
Voltaje de alimentación	100V - 240V
Consumo de energía	Normal 65VA Máximo 190 VA
Dimensiones	Base 274 **x** Profundidad 485 **x** Altura 445
Peso	22 Kg

Figura 3.2: Características técnicas del equipo y la imagen.

Luego de este proceso, el software del microscopio arroja un archivo .TIF de dos canales que podrá ser guardado en la computadora e introducido al software desarrollado en este trabajo.

3.2 Imagen de 16 niveles

Como se mencionó anteriormente, el médico oftalmólogo hace uso del microscopio especular para capturar una imagen de las células de la córnea. Esta será posteriormente procesada por el software del microscopio para crear la segmentación, formando líneas que pasen a través del espacio que separa dos células adyacentes. A partir de la segmentación obtenida, este mismo software calcula ciertos parámetros (grosor de la córnea, número de células, tamaño mínimo y máximo, tamaño promedio, densidad celular, entre otros), y por último, genera un archivo .TIF de dos canales. En la Figura 3.3 se muestra el primer canal del archivo generado por el microscopio, en este se guarda la imagen tomada de la córnea del paciente en la parte izquierda, y a la derecha hay un espacio oscuro vacío para que las dimensiones de los dos canales sean iguales.

Por otro lado, en la Figura 3.4 tenemos el canal 2. Esta es una imagen indexada de 16 niveles que consta de dos partes: a la izquierda muestra la segmentación calculada por el microscopio al analizar la imagen obtenida (ver Figura 3.4a), y en la

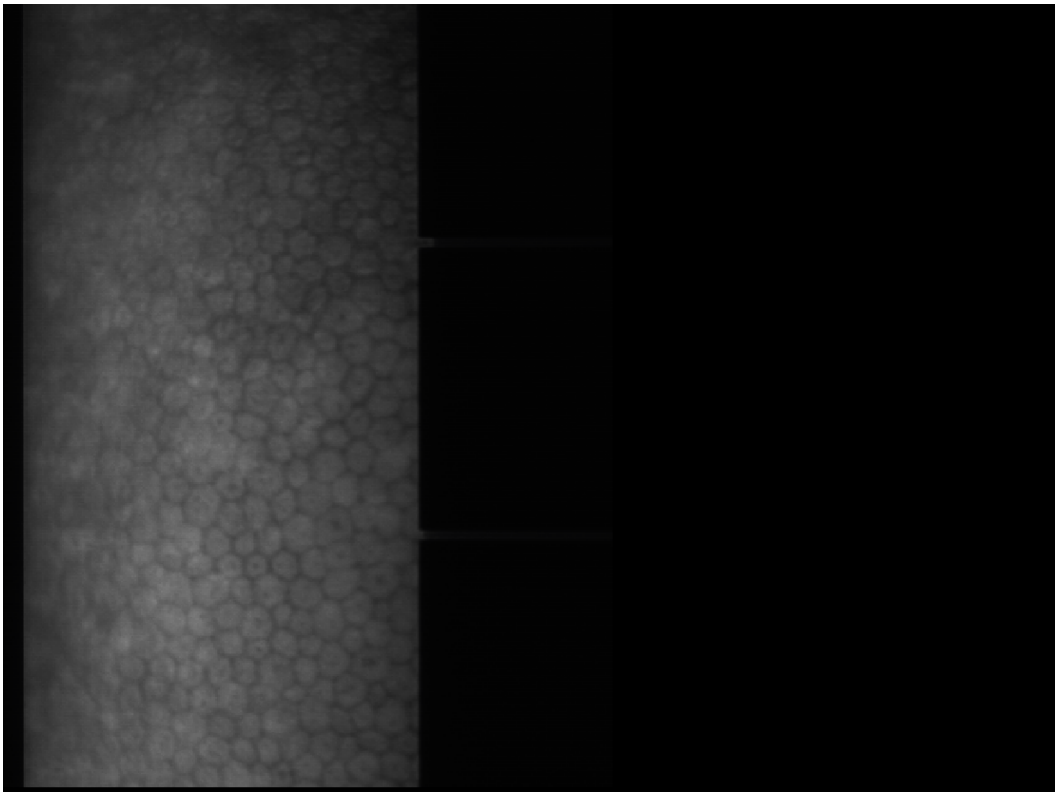


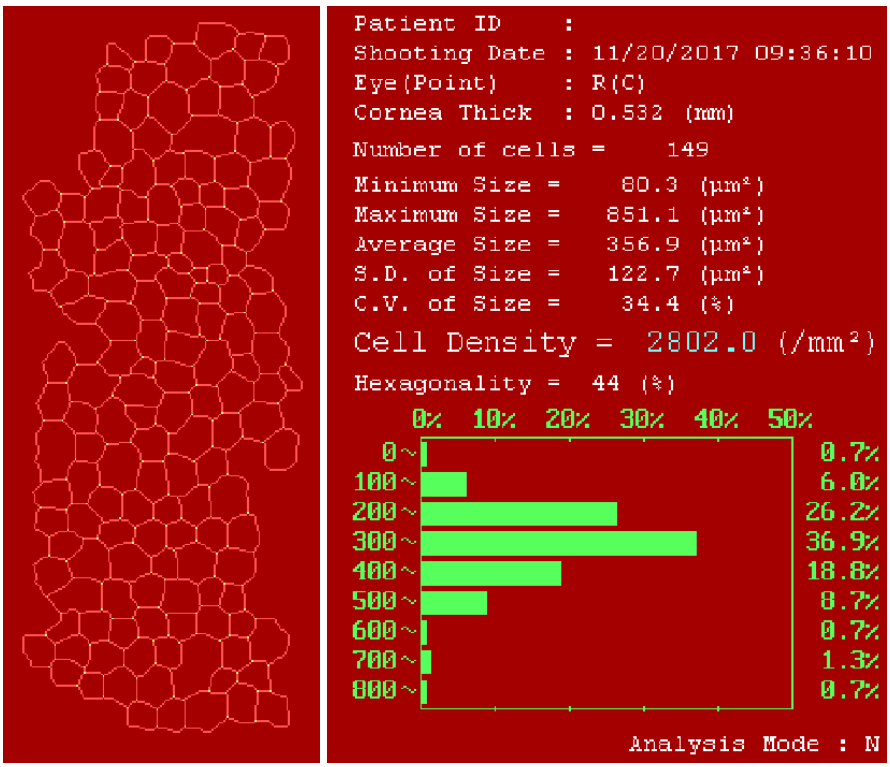
Figura 3.3: Canal 1. Imagen de células corneales.

parte derecha de la misma se tienen los datos del paciente y de la córnea analizada, como se observa en la Figura 3.4b.

Esta imagen de 16 niveles posee su propia paleta de colores, la cual consta de 16 colores enumerados del 0 al 15. Los píxeles que conforman la imagen del canal 2 adoptan un color dentro de la paleta de colores para expresar cierta información, lo que permite que al analizar la imagen con un software especializado, se pueda extraer la información necesaria y omitir el resto, quedándonos sólo con los píxeles cuyos valores sean los de nuestro interés. De este modo, podemos crear una imagen binaria de la segmentación celular (donde los bordes tendrán el valor de cero y las células el valor de uno) separando los píxeles que se requieran de los demás, de manera que sea posible realizar las operaciones morfológicas necesarias para modificar y corregir la segmentación.

3.3 Algoritmo de corrección

El software que se desea desarrollar deberá cumplir con ciertos requerimientos funcionales, de manera que permitan al usuario realizar una corrección de segmentación rápida y óptimamente. El software:



(a) (b)

Figura 3.4: Canal 2. *(a)* Segmentación calculada por el microscopio. *(b)* Parámetros calculados a partir de la segmentación.

- Tendrá como entrada un archivo .TIF de dos páginas obtenido por el microscopio especular
- Permitirá la creación de guttas y células
- Constará de una herramienta capaz de unir dos o más regiones cercanas
- Permitirá dibujar nuevas regiones, en caso de que esté siendo excluida información importante de la segmentación original
- Realizará los cálculos de los parámetros necesarios, tanto para células como para guttas
- Tendrá como salida un archivo .TIF de características similares al usado de entrada

Para ello nos hemos planteado la metodología expuesta en la Figura 3.5, donde se muestra el flujo general con el que constará el software para realizar la corrección de segmentación de células endoteliales. En este se tiene como entrada la imagen

I, la cual es un archivo .TIF de dos canales, que contenga tanto la imagen de las células de la córnea como la segmentación de la misma, al que se le realizarán las modificaciones necesarias usando las herramientas del software, para obtener nuevos datos de la segmentación, y este arroja como salida S, que es un nuevo archivo .TIF multipáginas con las mismas características que el de entrada.

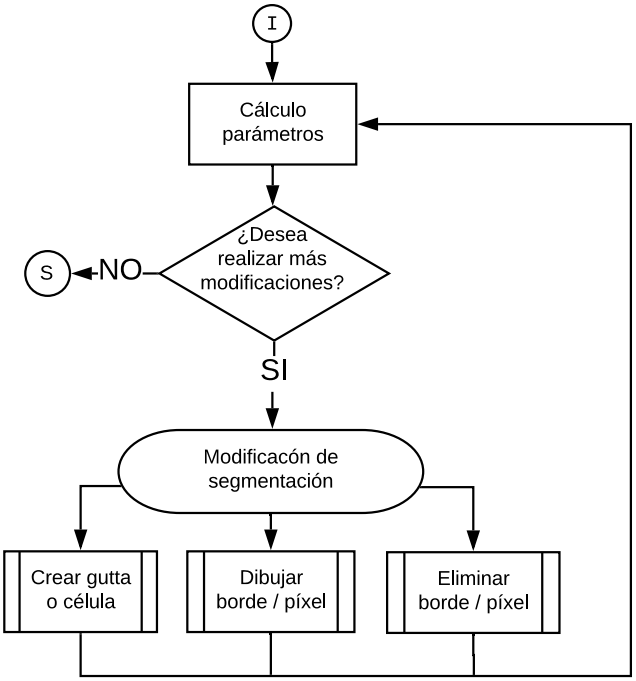


Figura 3.5: Diagrama de corrección de segmentación de células endoteliales.

A continuación se describirá la metodología llevada a cabo para el correcto funcionamiento de las diferentes herramientas que ofrece el software al usuario para corregir la segmentación celular propuesta por el microscopio especular. Cabe mencionar que un costado de la ventana se habrá una imagen recortada que muestra la imagen del canal 1 con la segmentación superpuesta (esta imagen la llamaremos $\boldsymbol{R}$) para lograr una mejor interpretación por parte del usuario de los cambios que se vayan realizando.

3.3.1 Cálculo de parámetros

Posterior a seleccionar la imagen con la que se trabajará, el algoritmo identificará automáticamente el canal 2 y extraerá la segmentación de la imagen, creando una nueva imagen binaria que la contenga, luego procederá a usar la función *bwlabel*, la cual enumera las regiones blancas en una imagen binaria (haciendo que cada píxel dentro de la región tome el valor de la respectiva etiqueta), las cuales, a su vez, representan a las células, por lo que esta función nos permite conocer la cantidad de células en la imagen.

Al cargar la imagen, no se dispone del área en μm^2 que ocupan las células segmentadas, por lo tanto el usuario deberá introducir el valor de densidad celular calculado por el microscopio, con el fin de hallar un factor de conversión de píxeles a μm^2, usando la fórmula

$$FC = \frac{N}{pix * CD} \ , \tag{3.1}$$

donde FC es el factor de conversión de píxeles a μm^2; N es la cantidad de células; pix es la cantidad de píxeles que ocupa el grupo de células en la imagen; y CD es la densidad celular introducida por el usuario.

En este punto sólo resta conocer la cantidad de píxeles que conforman cada región. Para esto, se usa nuevamente la imagen de etiquetas; en esta se recorrerá elemento a elemento la matriz que la representa para hacer un conteo de la cantidad de píxeles cuyos valores sean iguales a cada una de las etiquetas. Estas cantidades son guardadas en un vector de *N* posiciones, para luego comparar cada posición dentro de este y determinar el tamaño de la célula más grande y la más pequeña, además de poder calcular el área promedio de células.

Es importante mencionar que en esta parte del algoritmo se realiza nuevamente el cálculo de densidad celular, despejando la incógnita de la ecuación 3.1, puesto que este procedimiento se repite siempre que la segmentación es modificada usando alguna de las herramientas que se mencionarán más adelante.

3.3.2 Crear gutta o célula

Como ha sido mencionado con anterioridad, la segmentación propuesta por el microscopio no es del todo acertada, debido a que identifica células en partes donde no las hay. Para corregir esto, el software tendrá una opción llamada *Select Guttae*, que le permitirá al usuario seleccionar una región que esté siendo equivocadamente considerada como una célula, para transformarla en una gutta. Visualmente, en la imagen ***R***, las guttas serán marcadas con colores más oscuros.

El píxel seleccionado es detectado, posterior a lo cual se hace una búsqueda en la imagen de etiquetas para conocer el valor del mismo. Luego, se buscarán todos los píxeles cuyos valores sean iguales al encontrado, para oscurecerlos en la imagen R, hacerlos cero (negro) en la imagen de células, y uno (blanco) en la imagen G. Esta última es creada inicialmente a partir de una matriz de ceros del mismo tamaño de la matriz que contiene la imagen R; es en esta en la que se ven reflejadas todas las

modificaciones realizadas a las guttas creadas, y a partir de la misma, se obtiene toda la información requerida de estas, realizando el procedimiento descrito en la Sección 3.3.1.

Como contraparte, el software tendrá una herramienta llamada *Select Cell* que haría el proceso inverso a *Select Guttae*. Para usar esta herramienta se debe seleccionar una región que esté siendo mostrada actualmente como una gutta, para transformarla en una célula realizando el mismo procedimiento expuesto anteriormente. En la imagen ***R***, los píxeles en cuestión retomarán su color original; en la imagen de células, los valores serán cambiados a 1 (blanco), y en la imagen de guttas estos serán cero (negro) nuevamente.

Como método un poco más rápido y menos tedioso para marcar guttas dentro de la segmentación, el software también constará de una opción con la cual, por medio de un clic, se "siembra una semilla" en un píxel que contenga una gutta, a partir de este se analizarán los píxeles cercanos para determinar hasta dónde se expande la gutta seleccionada e incluirla en los cálculos; este método es conocido como *crecimiento de regiones* (ver Sección 2.4). En esta ocasión, el software no trabajará en base a la imagen binaria, sino que usará la imagen del canal 1.

Para la GUI desarrollada en MATLAB, el criterio que define el contorno de la gutta creada por crecimiento de regiones, es que todas las regiones de la segmentación que tengan al menos un elemento en común con la región creada, será unida a esta, debido a que existe la posibilidad de que correspondan a células no saludables.

3.3.3 Dibujar borde o píxel

Como se ha podido notar, la segmentación calculada por el microscopio no abarca toda la imagen, por lo que es posible que queden elementos importantes por fuera de esta. El botón *Draw Border* permitirá al usuario crear líneas rectas negras, que harán parte de la segmentación celular.

En la Figura 3.6 se muestra el procedimiento para dibujar un línea del borde. Para dibujar la primera línea, el usuario deberá seleccionar dos puntos (P_1 y P_2) dentro de la imagen, como se aprecia en la Figura 3.6a, los cuales quedarán guardados en memoria. Con estas dos coordenadas como entrada, se dibuja sobre la segmentación una línea recta que vaya desde P_1 hasta P_2, como en la Figura 3.6b. Posterior a esto, el usuario tendrá la posibilidad de seleccionar un nuevo punto, por lo que P_1 tomará el valor de P_2, y P_2 será el nuevo punto seleccionado, repitiendo nuevamente el proceso, tal como se muestra en la Figura 3.6c.

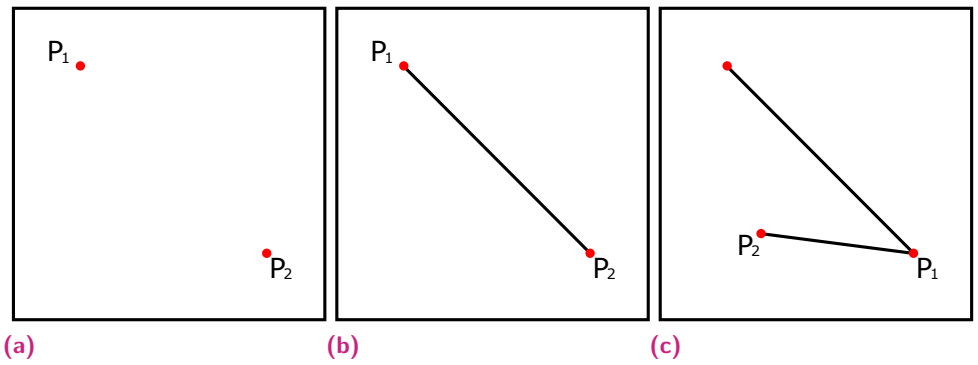


Figura 3.6: Ejemplo de dibujar borde. *(a)* Selección del usuario. *(b)* Dibujar línea. *(c)* Nueva coordenada para dibujar línea.

Como complemento a esta función, el software también posee una herramienta para cambios más pequeños y precisos: *Draw Pixel*. Con este botón se pueden dibujar píxeles sobre la imagen y hacerlos parte de la segmentación.

3.3.4 Eliminar borde o píxel

Mediante análisis, se detectó que la segmentación arrojada por el microscopio, en ocasiones crea más de una región donde evidentemente sólo hay una, como se puede notar en la Figura 3.7. Para este tipo de situaciones, el software tendrá una opción que le permita al usuario eliminar la línea que separa dos regiones vecinas.

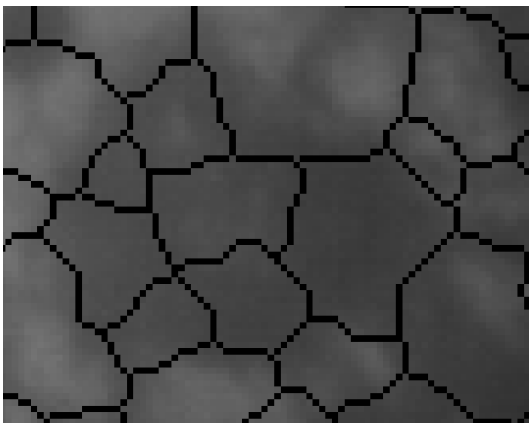


Figura 3.7: Recorte de una sección donde se pueden apreciar más de una región segmentada sobre una misma gutta.

Esto es, a partir de dos clic. El usuario seleccionará dos regiones vecinas donde quiera eliminar la línea que las separa, como se ve en la Figura 3.8a, en la que los asteriscos rojos representan la selección del usuario. Se creará internamente una imagen binaria que contenga estas dos regiones, para poder trabajar de manera más cómoda (véase Figura 3.8b). Posterior a esto, se aplicará sobre la imagen la función *imclose* con un elemento estructurante de 3x3 píxeles, para eliminar la línea

negra que separa las regiones, obteniendo como resultado una única región, como la mostrada en la Figura 3.8c).

Llegados a este punto, hay que mencionar que en la imagen binaria se están teniendo en cuenta las intersecciones de la segmentación (los puntos amarillos del canal 2), los cuales indican los puntos en donde se tocan 3 regiones vecinas, por lo que no pueden ser eliminados, a menos que se deseen eliminar los bordes entre estas tres regiones. Por esta razón, se restan las intersecciones, para eliminarlas de la imagen binaria, lo que nos da como resultado la Figura 3.8d. Por último, se obtiene la Figura 3.8e restando la imagen *b* a la *d*, lo que resulta en la línea que se desea eliminar, para obtener la Figura 3.8f, donde tenemos las dos regiones iniciales unidas.

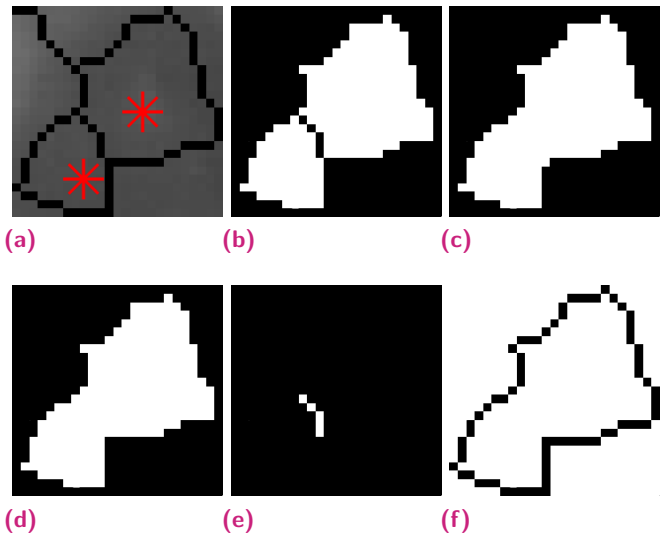


Figura 3.8: Trancisión de las imágenes binarias. *(a)* Selección de dos regiones vecinas. *(b)* Binarización de las regiones seleccionadas. *(c)* Imagen binaria luego del *imclose*. *(d)* Imagen binaria excluyendo las intersecciones. *(e)* Línea resultante de la resta entre la imagen *d* y *b*. *(f)* Regiones unidas.

Como resultado obtenemos lo mostrado en la Figura 3.9, en la que se aprecia que la línea que antes separaba dos regiones ya no está, dando como resultados una única región.

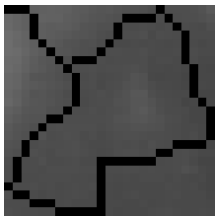


Figura 3.9: Resultado de usar la herramienta *Delete Border*.

Del mismo modo que en la sección anterior, aquí se tiene una herramienta para cambios que requieren mayor precisión. Con esta se pueden eliminar píxeles de la segmentación.

3.4 Interfaz en MATLAB (GUI)

Esta interfaz gráfica contiene cada una de las herramientas mencionadas en la Sección 3.3, y se encuentra organizada como se muestra en la Figura 3.10, donde fueron enumeradas cada una de las partes de la misma, para ser descritas posteriormente.

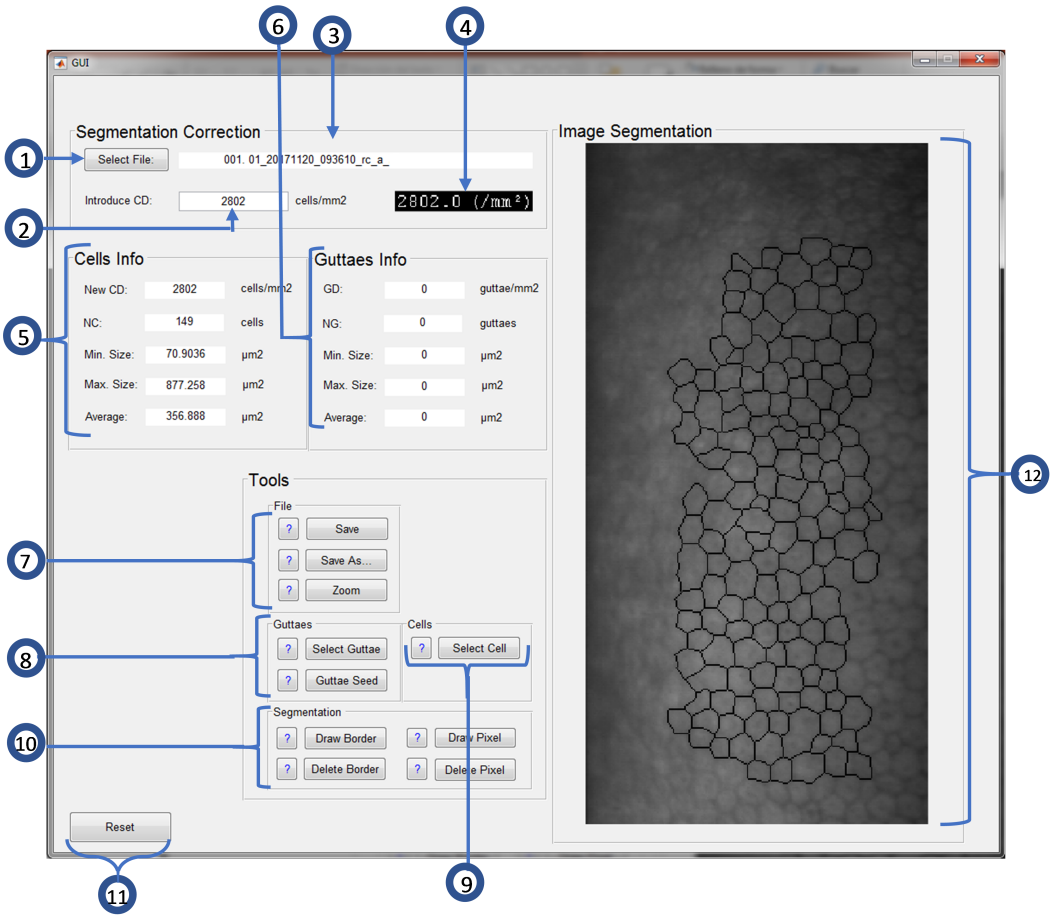


Figura 3.10: Interfaz gráfica de MATLAB.

Donde:

1. El botón permite al usuario seleccionar el archivo .TIF que contiene la segmentación de la que se desea trabajar.

2. Este espacio inicialmente está en blanco. Aquí el usuario deberá introducir la densidad celular calculada por el microscopio.

3. Se muestra el nombre del archivo seleccionado.

4. Recorte de la imagen del canal 2 donde se muestra la densidad celular calculada por el microscopio.

5. Parámetros calculados en base a la información de las células.

6. Parámetros calculados en base a la información de las guttas.

7. Opciones de guardado y zoom con sus respectivos botones de ayuda.

8. Herramientas para crear guttas con sus respectivos botones de ayuda.

9. Herramientas para crear células con sus respectivos botones de ayuda.

10. Herramientas para modificar la segmentación con sus respectivos botones de ayuda.

11. Botón de reset. Deshace todas los cambios y regresa toda la información a su estado inicial.

12. Segmentación superpuesta sobre la imagen de las células de la córnea. Aquí el usuario interactuará con la imagen para usar las herramientas.

3.4.1 Ejemplo ilustrativo

En esta sección se mostrará un ejemplo paso a paso del uso de las herramientas ofrecidas por la GUI creada en MATLAB, para realizar una corrección de segmentación de las células del endotelio corneal.

El primer paso será seleccionar la imagen sobre la que se desea trabajar e introducir la densidad celular que indica la imagen original, para que sea posible realizar los cálculos. A un costado se muestra un recorte de la imagen del canal 2 del archivo seleccionado, donde se indica la densidad celular (ver Figura 3.11).

La Figura 3.12a muestra una sección de la imagen seleccionada en la que se pueden apreciar algunas guttas, donde es usada la herramienta de dibujar bordes para separar lo que parecen ser dos guttas diferentes en la 3.12b. En las Figuras 3.12c y 3.12d fueron seleccionadas algunas regiones para transformarlas en guttas, entre las cuales se hizo una mala selección, que es corregida en la Figura 3.12e usando la herramienta de seleccionar células.

En la parte izquierda de la imagen se puede notar que hay una gutta más pequeña cuya área está dentro de 5 regiones diferentes; en esta ocasión usaremos el método 1 de crecimiento de regiones, expuesto en la Sección 2.4. Al seleccionar un punto dentro de la gutta con esta herramienta, la región se expande hasta ocuparla por

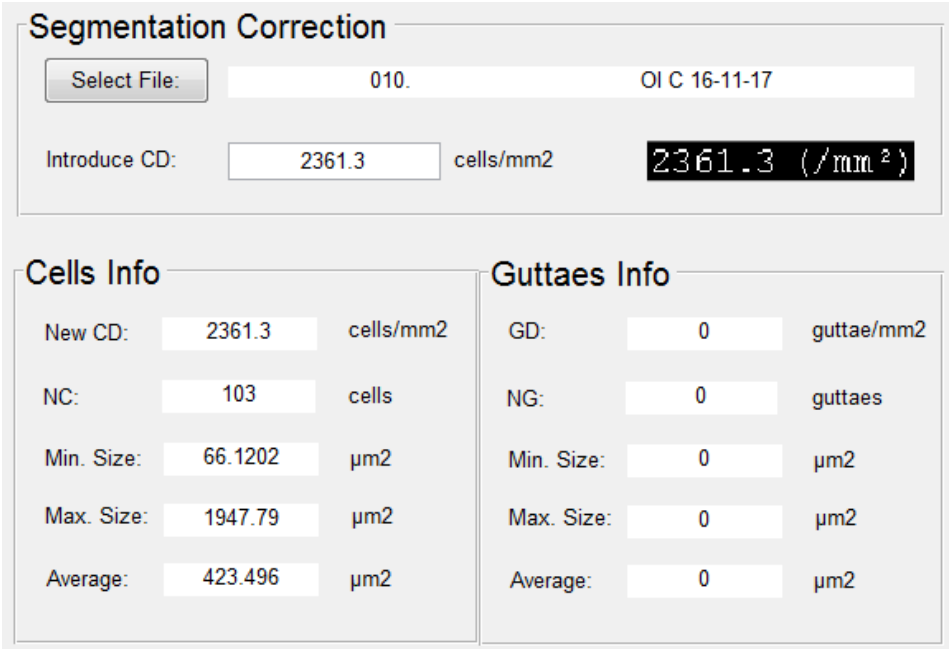


Figura 3.11: Parámetros calculados inicialmente al introducir la densidad celular.

completo,luego, debido a que está dentro de otras 5 regiones, el software identifica estas como guttas. Esto lo podemos apreciar en la Figura 3.12f.

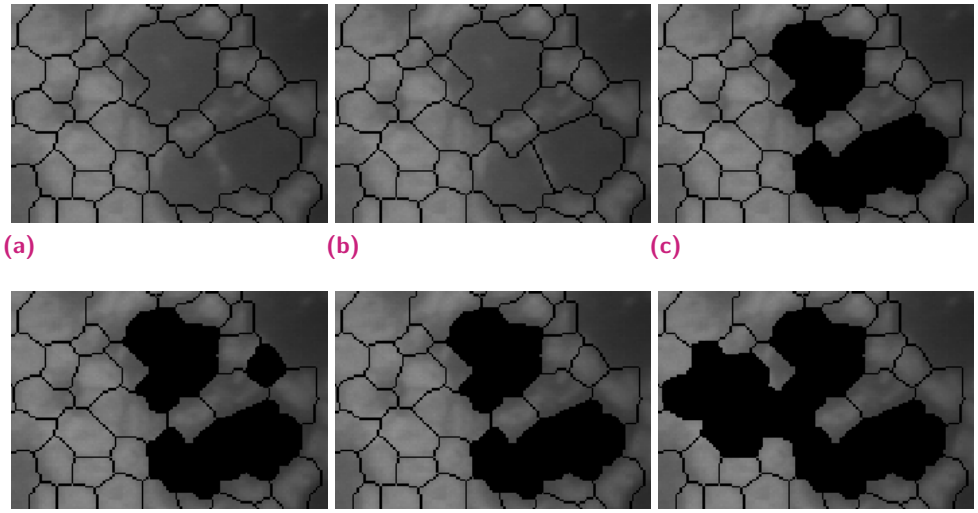


Figura 3.12: Transición de cambios realizados a la segmentación. *(a)* Segmentación inicial. *(b)* Línea dibujada. *(c)* Selección de guttas. *(d)* Selección equivocada de gutta. *(e)* Selección de célula. *(f)* Crecimiento de regiones

Cabe resaltar que la GUI de MATLAB, al detectar dos o más guttas muy cercanas (separadas sólo por una línea de un píxel de grosor), las une en una sola. Por esta razón, en los resultados obtenidos de las modificaciones realizadas previamente, mostrados en la Figura 3.13, solamente hay una gutta de gran tamaño.

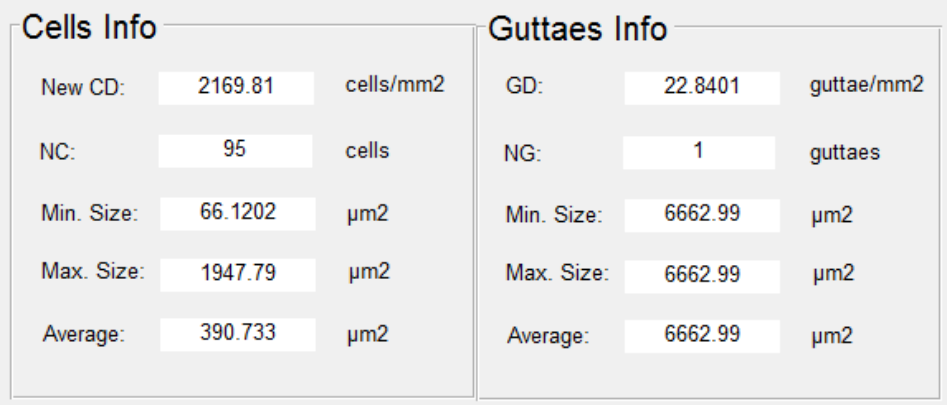


Figura 3.13: Resultados obtenidos de la corrección de segmentación usando la interfaz gráfica desarrollada en MATLAB.

4 Implementación en Python

El uso de MATLAB tiene algunas ventajas y desventajas. Este tiene una gran cantidad de operaciones y funciones robustas incluidas, por lo que es fácil desarrollar un algoritmo que permita realizar pruebas. Por otro lado, para el uso de este software es necesario tener licencia, la cual es costosa.

Por lo tanto, se decidió portar el código de MATLAB a Python. Este es un lenguaje de programación de código abierto, por lo que no necesita licencia para ejecutar un programa, y brinda cierta facilidad para realizar operaciones, puesto que al ser un lenguaje muy usado, la cantidad de librerías y operaciones creadas es grande. En este capítulo se describirá el proceso llevado a cabo para la implementación del software en Python.

4.1 Preprocesamiento

Como ha sido mencionado en anteriores secciones, el software está diseñado para funcionar con las imágenes obtenidas del microscopio especulas SP-3000P. Las características de imagen son claves para poder ser procesada de manera correcta. La Figura 4.1 resume las características que debe tener el archivo que será leído por el software.

Formato	TIFF (Tagged Image File Format) [.tiff, .tif]
Cantidad de canales	**2.** Canal 1: fotografía original; canal 2: segmentación y datos. **3.** Canal 1: fotografía original; canal 2: segmentación y datos; canal 3: segmentación y datos originales.
Resolución	640x480

Figura 4.1: Características de la imagen de entrada al software.

4.2 Procesamiento de la imagen

Una vez escogida la imagen a trabajar, se extrae la segmentación a partir del canal 2 de la imagen y se utiliza para etiquetar las respectivas regiones en la imagen original.

Posteriormente, es necesario que el usuario ingrese un valor de densidad celular para calcular todas las características de las regiones. Esto genera una entrada inicial en el historial de valores del software, que almacenara cualquier futura modificación sobre las variables de las regiones y la imagen mostrada en la interfaz, lo que permitirá retroceder o rehacer modificaciones hechas. El calculo de las nuevas características de las regiones es realizado cada vez que el usuario realice una modificación.

El procesamiento y cálculo de las características de las regiones se realiza utilizando la siguiente serie de pasos:

1. Extracción de la segmentación de las regiones a partir del canal 2 del archivo

2. Realizar un *closing* sobre la matriz de guttas para unificar guttas cercanas

3. Superponer la segmentación sobre la imagen original (canal 1), generando la imagen $\boldsymbol{R}$

4. Superponer la matriz de guttas sobre la imagen $\boldsymbol{R}$

5. Realizar un etiquetado sobre la segmentación para obtener el número de células

6. Realizar un etiquetado sobre la matriz de guttas, para obtener el número de guttas

7. Analizar las regiones de células y guttas, para calcular los valores de área mínima, máxima, promedio y densidad tanto de células como de guttas

8. Añadir todos los nuevos valores como una nueva entrada en el historial y actualizar todos los datos mostrados en la interfaz y la imagen

Luego de esto, el software queda a la espera de la próxima acción del usuario. Este podrá continuar y realizar una nueva modificación sobre la imagen o puede finalizar la edición, guardando los cambios o generando una nueva imagen. Esta última opción borra todo registro previo en el historial y los valores actuales se vuelven la primera (y, por el momento, única) entrada en el mismo.

4.3 Estructura del código

Para llevar a cabo la correcta implementación del software en Python, se estableció inicialmente un flujo de los procesos que este debe llevar a cabo para realizar una corrección de segmentación.

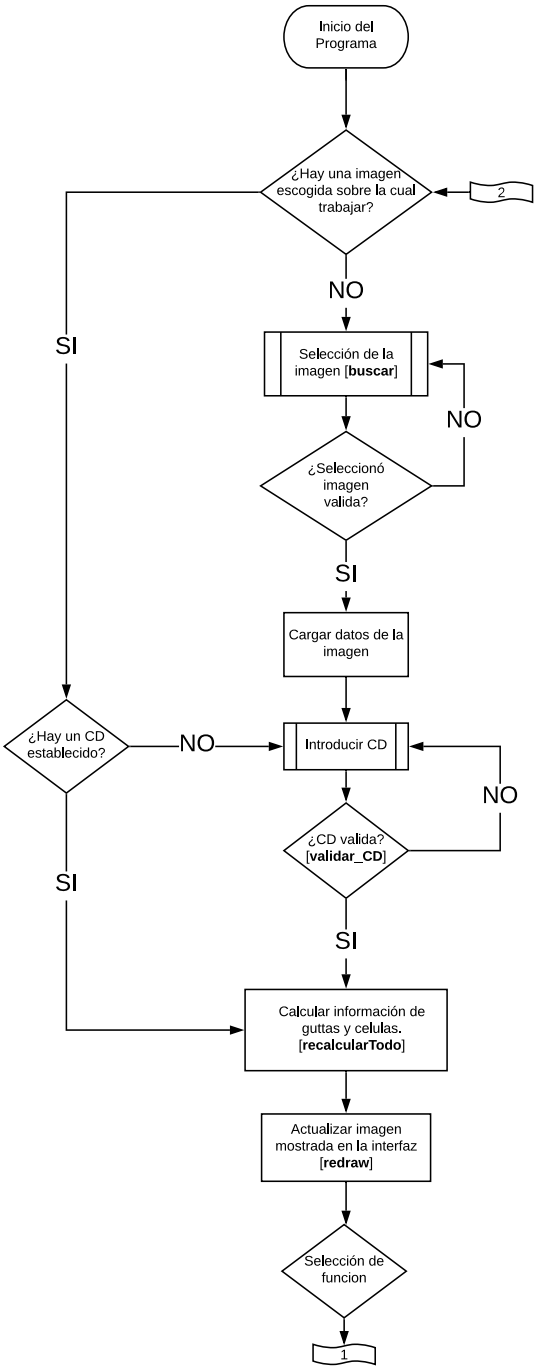


Figura 4.2: Diagrama de corrección de segmentación de células endoteliales en lenguaje Python. Parte 1.

La Figura 4.2 muestra la primera parte del proceso. En esta, se dice que el software inicialmente realiza una validación de la imagen de entrada sobre la cual se trabajará. Posterior a esto, solicita ingresar un valor de densidad celular, para calcular la cantidad de células/guttas y las áreas de estas.

Una vez obtenida la información anterior, se ejecuta el proceso de la Figura 4.2, donde se le ofrecen al usuario diferentes herramientas para corregir la segmentación de la imagen seleccionada, y obtener datos más acertados a partir de esta.

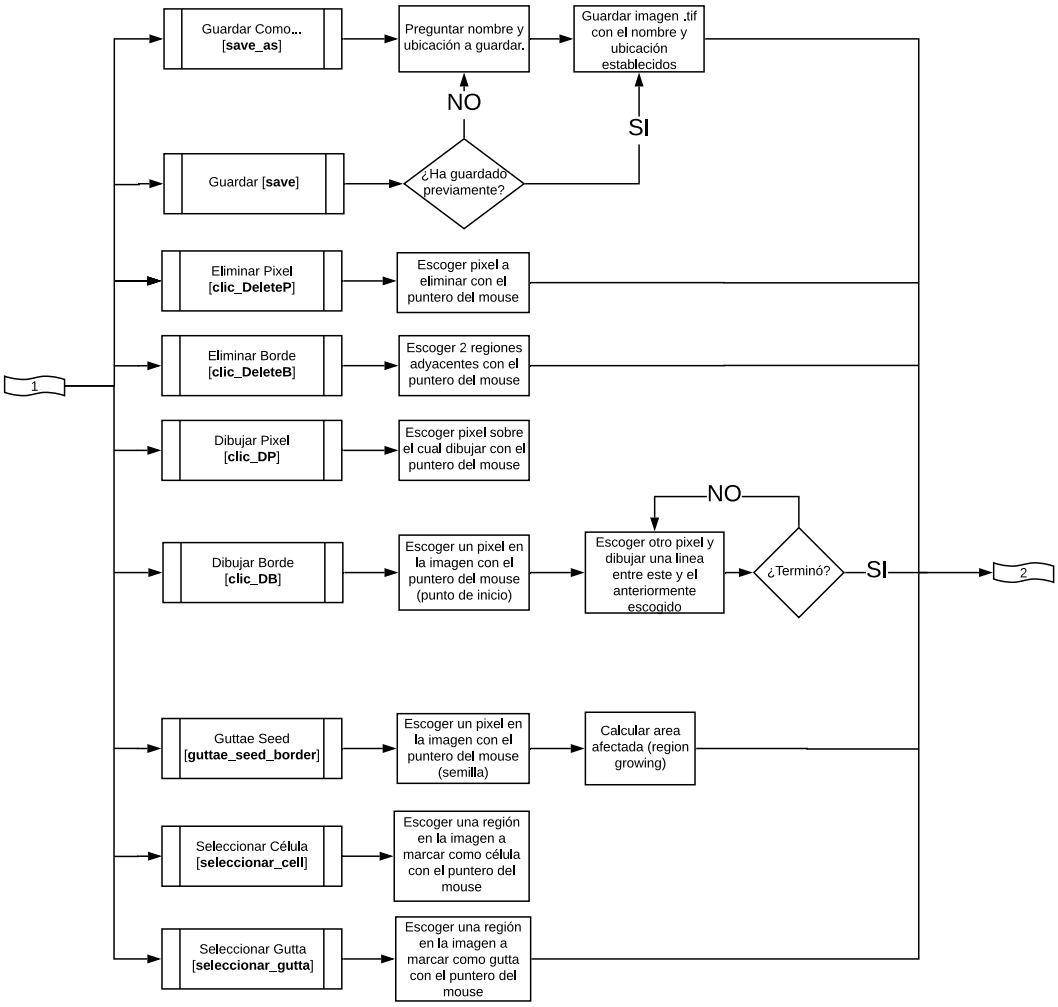


Figura 4.3: Diagrama de corrección de segmentación de células endoteliales en lenguaje Python. Parte 2.

Para lograr esto, se estableció la estructura del código en lenguaje Python del software de la siguiente manera:

1. **Librerías:** Las librerías que contienen las funciones necesarias para el desarrollo del software son

- **tkinter:** El módulo Tkinter (interfaz Tk) es la interfaz estándar de Python para el kit de herramientas GUI de Tk. Toda la interfaz gráfica del programa está diseñada con Tkinter.

- **PIL:** Librería de imágenes de Python. Esencial para el manejo de las distintas imágenes con las que trabaja el programa.

- **numpy:** Librería para mayor soporte en el manejo de vectores y matrices, constituyendo una biblioteca de funciones matemáticas de alto nivel para operar con esos vectores o matrices de manera ágil.

- **skimage:** Colección de distintos algoritmos para el procesamiento de imágenes.

- **scipy.ndimage:** Librería para el procesamiento de imágenes multidimensionales. Utilizada para realizar closing binario.

2. **Declaraciones para GUI (botones, rectángulos, imágenes, íconos):** Creación de la interfaz gráfica.

3. **Funciones de modificación de segmentación:** Estas funciones serán explicadas a continuación en la Sección 4.3.1, teniendo en cuenta tanto los parámetros de entrada como de salida de cada una de ellas.

4.3.1 Funciones de modificación de segmentación

buscar*(resetear=None)*

Permite escoger la imagen inicial sobre la cual se trabajará. Esta función establece los valores iniciales de cada uno de los parámetros de la información de la imagen en cero, a excepción del numero de guttas y células. Es llamada al presionar el botón de *Select File* en la interfaz. Elimina cualquier modificación previamente hecha sobre la imagen si es llamada nuevamente.

Parámetros de entrada:

- **resetear:** Parámetro (bool) para determinar si fue llamada por presionar el botón de reset.

validar_CD*(x)*

Verifica que el número ingresado en CD sea válido, haciendo distinción entre separación decimal con comas y puntos.

Parámetros de entrada:

- **x:** Cadena (String) a validar.

Salida:

- Numero (float). Es caso de x ser un número valido, será x convertido en número. En caso contrario, devolverá -1.

recalcularTodo*(by_button=False)*

Recalcula todos los datos de las células y guttas en la imagen a partir de las últimas modificaciones realizadas a la imagen.

Parámetros de entrada:

- **by_button:** Parámetro (bool) para determinar si fue llamada por uno de los botones de cambio de visualización de la imagen.

redraw*(event=None)*

Redibuja la imagen actualmente mostrada en la interfaz, ya sea porque los datos, centro de dibujado, o el nivel de zoom o canal de visualización cambiaron. Esta imagen es redibujada desde cero, teniendo en cuenta los valores actuales en el historial de las diferentes matrices para su creación: Guttas, segmentación e imagen original.

Parámetros de entrada:

- **event:** Variable de evento en la interfaz. Utilizada para obtener las coordenadas del clic del usuario (event.x, event.y). Es un parámetro opcional, ya que es posible que no se haya llamado por una acción directa de pulsación del usuario, ejemplo, como resultado de la finalización de una función de cálculo.

save_as*(just_save=False)*

Guarda la imagen con la que se está trabajando, agregando un tercer canal que contiene la información original de la imagen.

Parámetros de entrada:

- **just_save:** Parámetro (bool) que define si es necesario preguntar o no por el nombre del archivo. En caso de ser falso, simplemente se sobrescribe el archivo sobre el cual se está trabajando (predeterminado).

save*(just_save=False)*

Guarda la imagen con la que se está trabajando, agregando un tercer canal que contiene la información original de la imagen. Dependiendo de la función que lo haya llamado, puede o no preguntar por el nombre y ubicación del archivo a guardar. Sin embargo, siempre lo preguntará si es la primera vez en ser llamada en la actual ejecución del programa.

Parámetros de entrada:

- **just_save:** Parámetro (bool) que define si es necesario preguntar o no por el nombre del archivo. En caso de ser falso, simplemente se sobrescribe el archivo sobre el cual se está trabajando (predeterminado).

clic_DeleteP*(event=None)*

Permite eliminar un píxel específico de la segmentación en la imagen. Esta función no actuara cuando se haga clic sobre una gutta, para no generar micro-regiones de un píxel de área que afectaran el calculo de los parámetros.

Parámetros de entrada:

- **event:** Variable de evento en la interfaz. Utilizada para obtener las coordenadas del clic del usuario (event.x, event.y).

clic_DeleteB*(event)*

Elimina el borde entre 2 secciones adyacentes. Esta función debe ser ejecutada de forma sucesiva por lo menos 2 veces. La primera vez, guarda la posición de la primera sección marcada. La segunda, elimina el borde entre la sección actualmente marcada y la anterior, realizando un *closing* entre ambas regiones y luego superponiendo el área resultante sobre la imagen de la segmentación. El área de intersección entre ambas imágenes es la línea (borde) de reparación que había entre ambas regiones. Las coordenadas de dicha intersección son eliminadas de la segmentación.

Parámetros de entrada:

- **event:** Variable de evento en la interfaz. Utilizada para obtener las coordenadas del clic del usuario (event.x, event.y).

clic_DP*(event=None)*

Permite marcar un píxel específico en la imagen como parte de la segmentación (negro). Modifica directamente la imagen de la segmentación.

Parámetros de entrada:

- **event:** Variable de evento en la interfaz. Utilizada para obtener las coordenadas del clic del usuario (event.x, event.y).

clic_DB*(event=None)*

Crea un borde entre 2 secciones adyacentes. Esta función debe ser ejecutada de forma sucesiva por lo menos 2 veces. En la primera ejecución, guarda las coordenadas del píxel donde se realizó el click. Mientras, en la segunda y posteriores ejecuciones, crea una línea entre el píxel previamente seleccionado y el actual.

Parámetros de entrada:

- **event:** Variable de evento en la interfaz. Utilizada para obtener las coordenadas del clic del usuario (event.x, event.y).

guttae_seed_border*(img,seed)*

Esta función realiza el algoritmo de crecimiento de regiones sobre la imagen establecida tomando como base las coordenadas dadas. A partir del resultado, realiza un relleno del área calculada para omitir posibles secciones incompletas dentro de la gutta, posterior a esto extrae el borde la gutta. Este borde será posteriormente utilizado para modificar la segmentación de la imagen.

Parámetros de entrada:

- **img:** Imagen (PIL) sobre la cual se realizará el crecimiento de regiones.
- **seed:** Coordenadas (x,y) a partir de la cual se realizará.

Salida:

- Imagen (PIL) del borde de la sección de gutta calculada.
- Imagen (PIL) de la sección de gutta calculada (rellena).

seleccionar_cell*(event=None)*

Permite seleccionar, por medio de un clic, una región que esté siento tomada equivocadamente como una gutta para transformarla en una célula.

Parámetros de entrada:

- **event:** Variable de evento en la interfaz. Utilizada para obtener las coordenadas del clic del usuario (event.x, event.y).

seleccionar_gutta*(event=None)*

Permite seleccionar, por medio de un clic, una región que esté siento tomada equivocadamente como una célula para transformarla en una gutta.

Parámetros de entrada:

- **event:** Variable de evento en la interfaz. Utilizada para obtener las coordenadas del clic del usuario (event.x, event.y).

4.4 Interfaz gráfica

En esta sección se mostrará la interfaz gráfica desarrollada en Python, explicando cada una de sus partes.

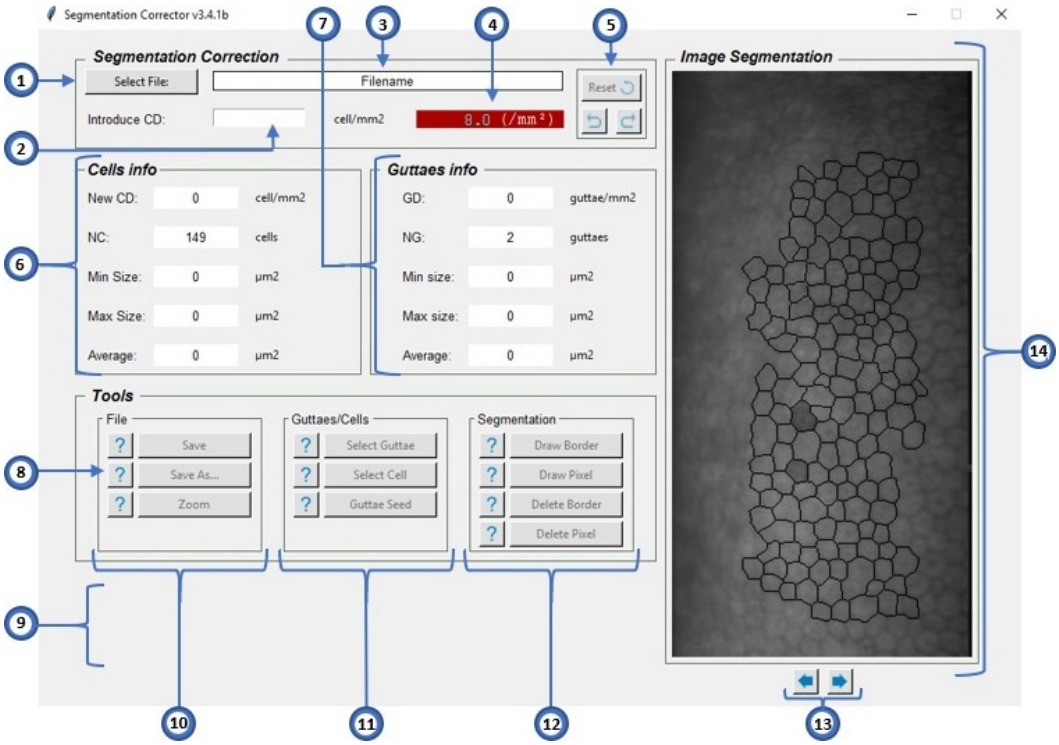


Figura 4.4: Interfaz gráfica desarrollada en Python.

Donde:

1. Botón para la selección del fichero sobre el cual se trabajará. Las demás funciones estarán deshabilitadas mientras no se haya seleccionado un archivo válido.

2. Entrada para introducir la densidad celular, la cual será validada.

3. Nombre del archivo seleccionado.

4. Densidad celular obtenida de la información en la imagen.

5. Controles para rehacer o deshacer los últimos cambios hechos sobre la imagen de las células corneales, así como un botón de reinicio a su estado inicial.

6. Muestra de información sobre las células en la imagen. Todos los parámetros son inicialmente mostrados en cero hasta introducir un CD, con excepción del número de células.

7. Muestra de información sobre las guttas en la imagen. Todos los parámetros son inicialmente mostrados en cero hasta introducir un CD, con excepción del número de guttas.

8. Botones de información sobre cada función del programa.

9. Área de mensajes para el usuario.

10. Funciones de archivo. Estas no realizan ninguna modificación sobre la imagen, simplemente permiten guardar los cambios hechos, además de hacer zoom sobre la imagen mostrada.

11. Funciones para modificación de células y/o guttas. Incluyen las funciones de seleccionar célula, seleccionar gutta y sembrar una semilla para guttas.

12. Funciones para modificación de la segmentación. Incluyen dibujado de borde, dibujado de píxel individual, borrado de borde y borrado de píxel individual.

13. Botones para la navegación entre 3 vistas de la imagen:

    - Vista1: Canal 1 del archivo, imagen capturad por el microscopio.

    - Vista2: Imagen de las células corneales con la segmentación y guttas superpuestas.

    - Vista3: Visualizar únicamente las guttas en una imagen binaria.

14. Visualización de la imagen actual trabajada, dependiendo de la vista actual escogida.

5 Experimentos y resultados

En esta sección se mostrarán los resultados obtenidos al corregir algunas segmentaciones realizadas en córneas guttata, usando el software desarrollado en el lenguaje de programación Python, para comparar los valores de los parámetros obtenidos con los iniciales.

5.1 Corrección de segmentación en córnea guttata

Una vez abierto el software, se selecciona la imagen sobre la que se desea trabajar usando el botón *Select File*, y posteriormente se introduce la densidad celular en el recuadro *Introduce CD* para que este pueda realizar los cálculos (ver Figura 5.1).

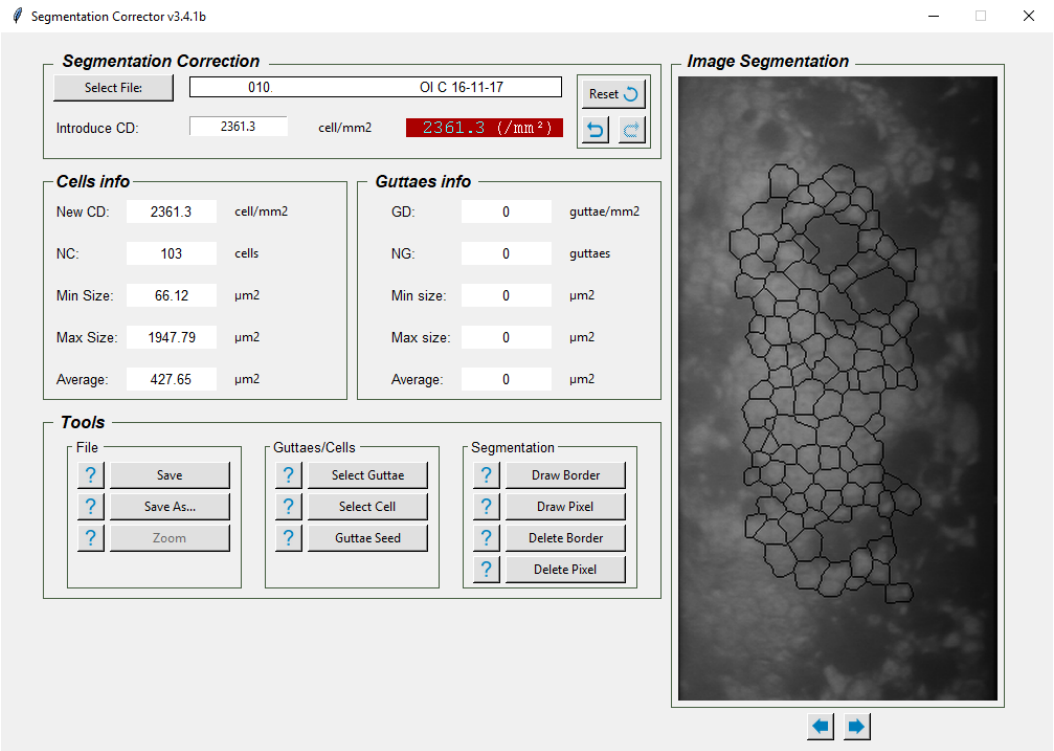


Figura 5.1: Selección de archivo e introducir densidad celular.

La Figura 5.2a muestra una sección extraída de la imagen seleccionada, en esta sección es posible visualizar algunas guttas. En las Figuras 5.2b y 5.2c fueron

seleccionadas algunas regiones para transformarlas en guttas, entre las cuales se hizo una mala selección, que es corregida en la Figura 5.2d usando la herramienta de seleccionar células.

Para complementar esta gutta, se usará la herramienta *Guttae Seed*; el software usa el método 2 de crecimiento de regiones, expuesto en la Sección 2.4. Al seleccionar un punto dentro de la gutta. como se puede apreciar en la Figura 5.2e, donde se nota que en la parte inferior de la gutta se ha expandido un poco la región.

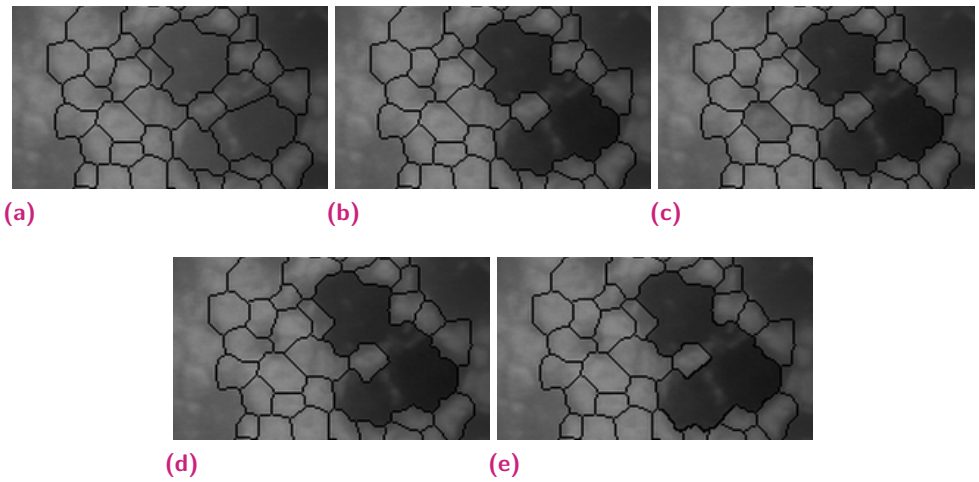


Figura 5.2: Corrección de segmentación. *(a)* Segmentación inicial. *(b)* selección de guttas. *(c)* Selección equivocada de gutta. *(d)* Selección de célula. *(e)* Crecimiento de región.

Luego de hacer estas correcciones, se obtuvieron los resultados mostrados en la Figura 5.3, donde se aprecia una diferencia entre la densidad celular inicial y la nueva de 92.99 células/mm^2, y de 4 entre las cantidades de células, siendo la nueva la que tiene menor cantidad de células. Las células que fueron excluidas del conteo, ahora son guttas, por lo que es posible calcular la densidad de guttas, lo que dio como resultado 22.91 guttas/mm^2.

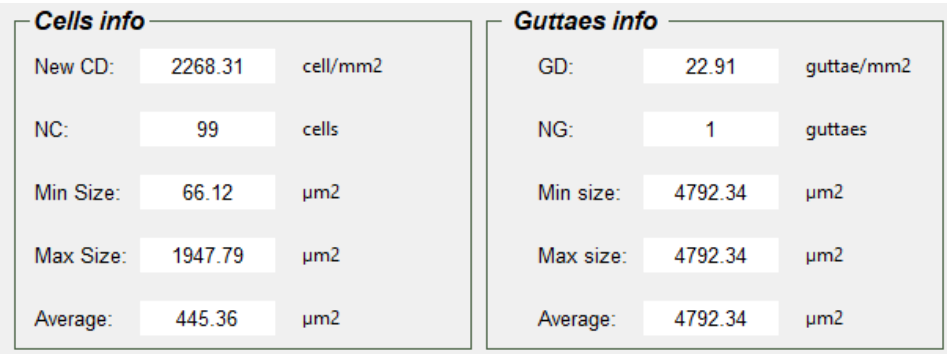


Figura 5.3: Resultados de la corrección de segmentación.

En los datos obtenidos podemos notar que inicialmente había una sobrevaloración de la densidad celular calculada por el microscopio. Con la nueva información, el médico encargado podrá dar una valoración más acertada del estado de la córnea del paciente.

Dado que los médicos necesitan estas imágenes en un archivo digital, el software arroja como salida un nuevo archivo .TIF, pero esta vez de 3 canales. El primer canal se guarda la imagen de las células de la córnea del paciente, la cual se encuentra en el canal 1 del archivo con el que se trabajó.

En la Figura 5.4 se puede apreciar el canal 2 del archivo guardado por el software. En este se guarda la segmentación corregida usando las herramientas que el mismo ofrece, junto a los parámetros calculados de tamaños, densidades y cantidades tanto de células como de guttas. Es importante mencionar que a diferencia de la segmentación obtenida por el microscopio, donde no tienen en cuenta las guttas, en este canal se marcan las guttas con un color gris, que contrasta respecto a los demás colores.

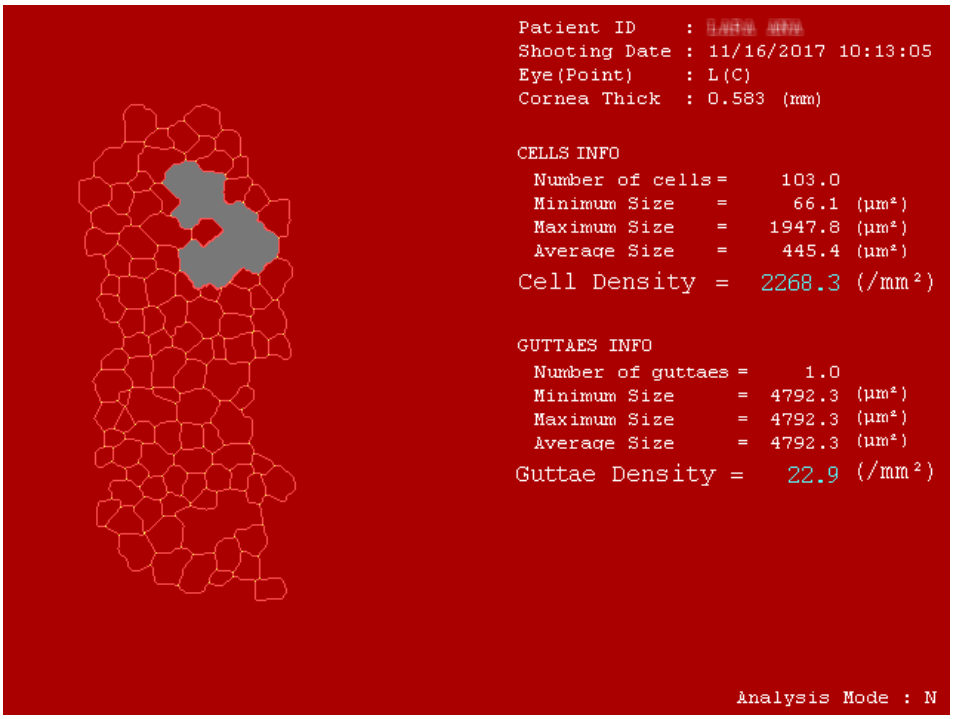


Figura 5.4: Canal 2 de salida. Segmentación corregida y nuevos datos.

Por último, en la Figura 5.5 se guarda la información inicial. En este se tiene la imagen del canal 1, sobre la cual se encuentra la segmentación original, y al costado derecho se tienen todos los datos originales.

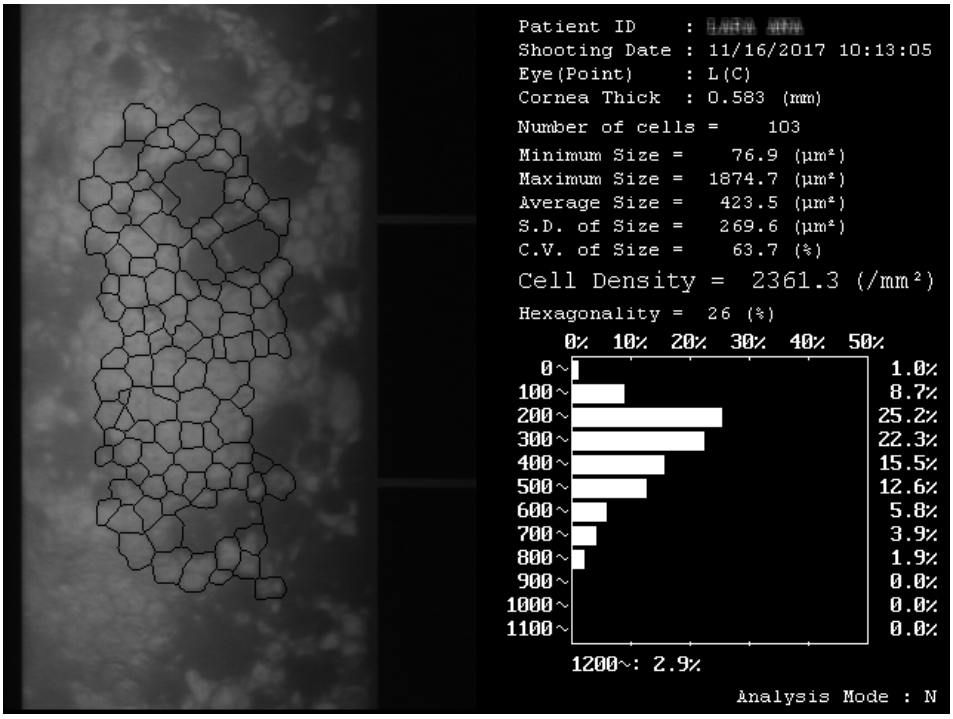


Figura 5.5: Canal 3 de salida. Segmentación y datos anteriores.

5.1.1 Otras pruebas

Prueba 1

En la Figura 5.6 podemos ver los datos iniciales calculados por el software al analizar la segmentación propuesta por el microscopio de la Figura 5.7a, donde la cantidad de células es 113, y la densidad celular tiene un valor de 2235.4 células/mm^2, lo que indica que es una córnea saludable, pero se puede notar a simple vista que no es correcto.

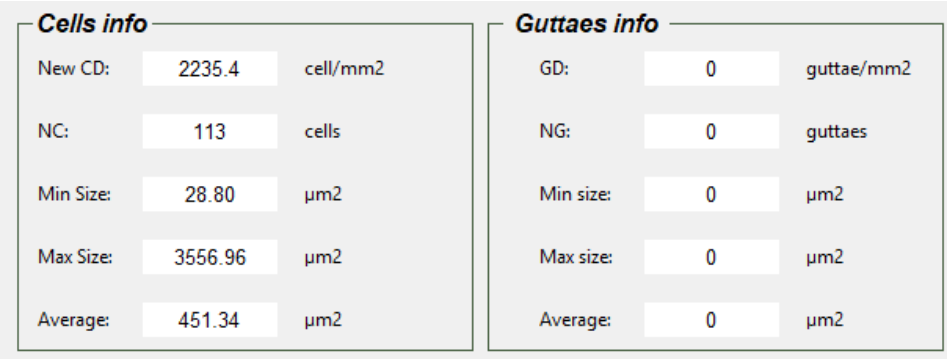
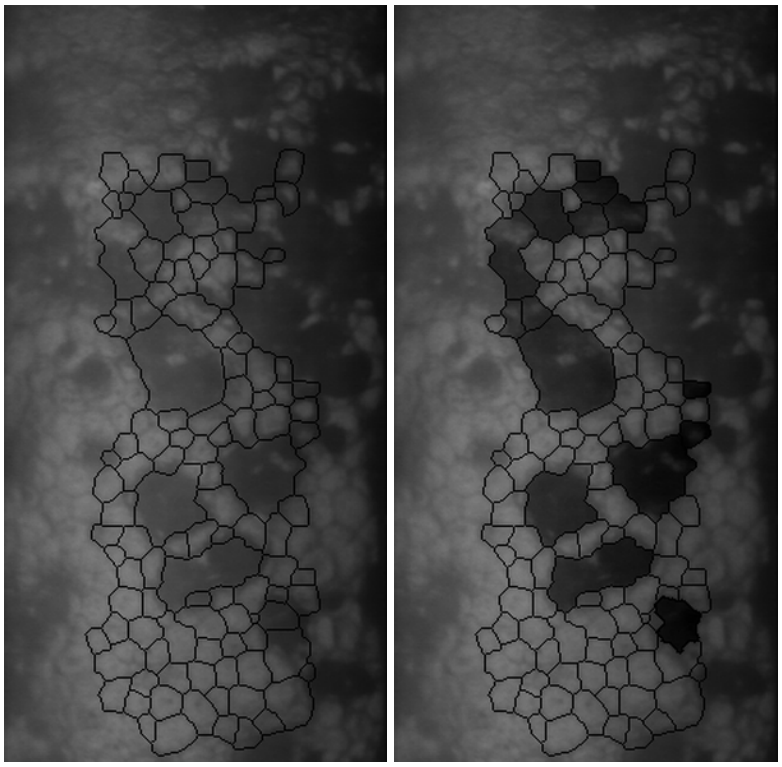


Figura 5.6: Prueba 1.Datos calculados por el microscopio especular.

Luego de realizar las correcciones usando las herramientas que brinda el software, se obtuvo la segmentación mostrada en la Figura 5.7b, donde se puede apreciar que

las regiones oscuras en la Figura 5.7a están siendo marcadas como guttas, por lo que se puede esperar que los datos que se obtengan sean más acertados.



(a) (b)

Figura 5.7: Segmentación prueba 1. *(a)* Segmentación calculada por el microscopio especular. *(b)* Segmentación corregida.

A partir de la corrección realizada, se calcularon los datos mostrados en la Figura 5.8, donde se logra apreciar una diferencia considerable de células, y por lo tanto también de densidad celular (de 14 y 2754.73, respectivamente), respecto a los datos iniciales. Teniendo estos resultados, es posible realizar una nueva valoración de la córnea para realizar el mejor diagnóstico posible.

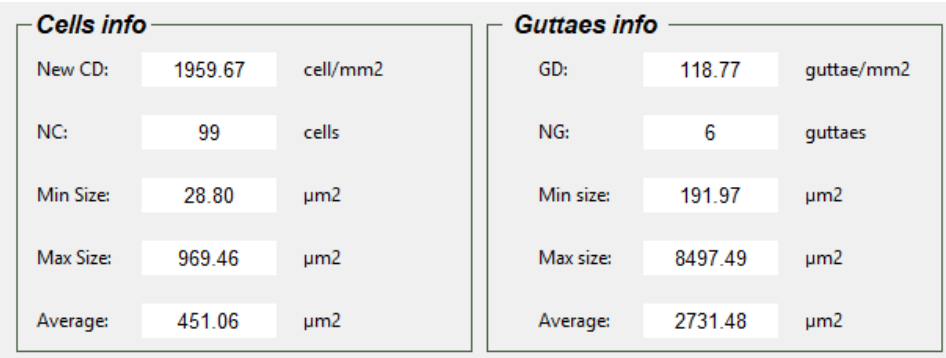


Figura 5.8: Prueba 1. Resultados de la corrección de segmentación.

Prueba 2

Del mismo modo, podemos ver los parámetros iniciales de la segmentación calculada por el microscopio especular en la Figura 5.9, donde se obtuvieron datos de densidad celular y cantidad de células de 1175.7 y 52, respectivamente.

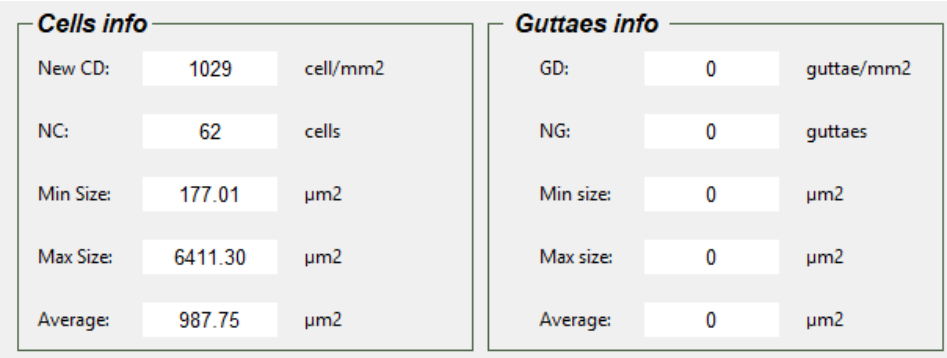
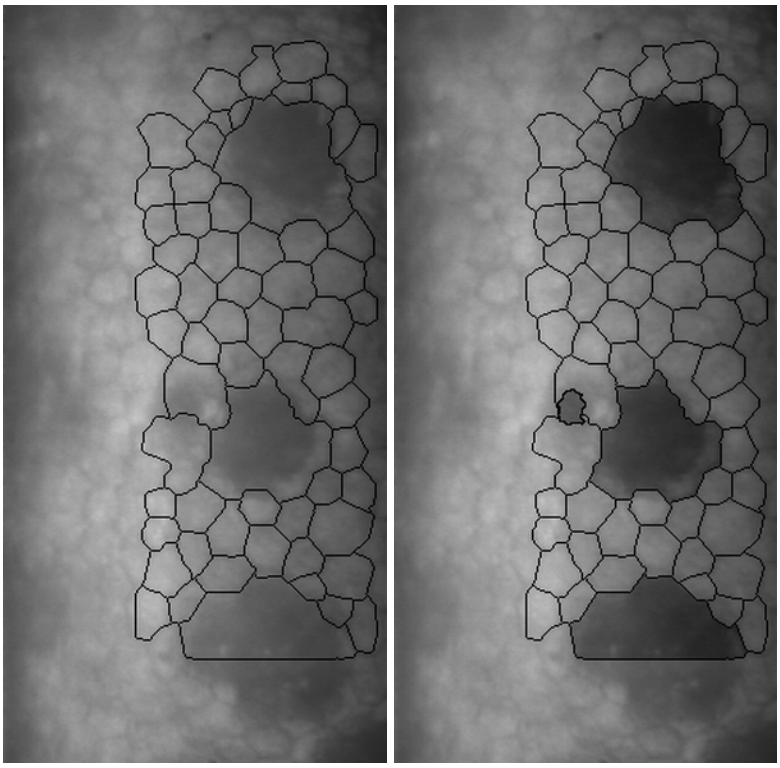


Figura 5.9: Prueba 2. Datos calculados por el microscopio especular.

En la Figura 5.10 se puede notar que existe diferencia de luminosidad y contraste respecto a la Figura 5.7, pero igualmente intentaremos usar las herramientas del software para corregir la segmentación.



(a) (b)

Figura 5.10: Segmentación prueba 2. *(a)* Segmentación calculada por el microscopio especular. *(b)* Segmentación corregida.

En las Figuras 5.10a y 5.10b podemos ver la segmentación original y la corregida, respectivamente. Nótese que la gutta más pequeña seleccionada fue usada la herramienta de crecimiento de regiones satisfactoriamente, puesto a que ha abarcado la gutta en su totalidad sin expandirse demás, y a diferencia de la GUI de MATLAB, este no transforma las regiones que compartan elementos con la nueva en guttas, sino que dibuja un contorno al rededor de esta, que es agregado a la segmentación.

En los resultados obtenidos, mostrados en la Figura 5.11, se puede notar principalmente grandes diferencias entre los tamaños de células y guttas respecto a los datos iniciales, información que el médico oftalmólogo puede tener en cuenta para realizar un nuevo diagnóstico al paciente.

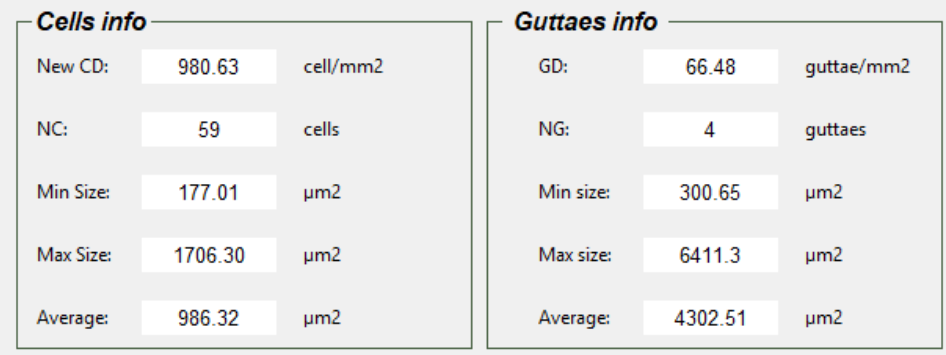


Figura 5.11: Prueba 2. Resultados de la corrección de segmentación.

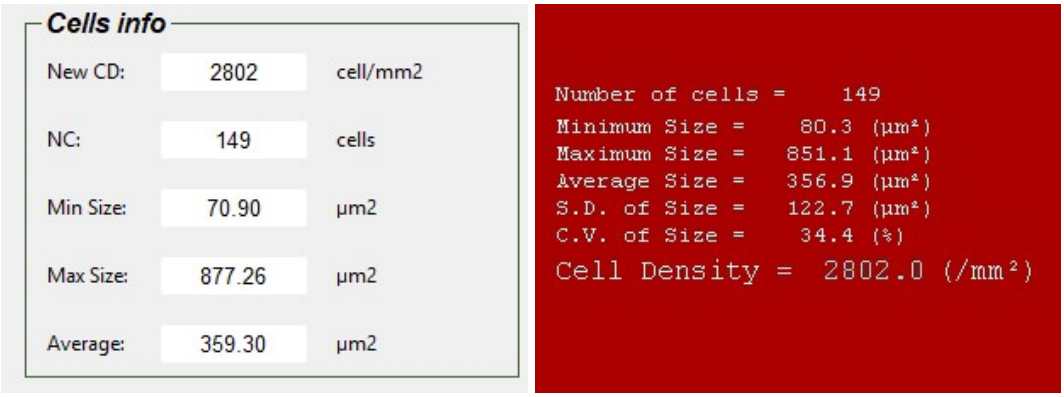
5.2 Limitaciones y trabajo futuro

5.2.1 Información inicial

La información inicial del tamaños de células (Figura 5.12a) no es igual que la obtenida por el microscopio (Figura 5.12b). Los valores son cercanos, pero no iguales. Esto puede deberse a algún parámetro interno del microscopio que no se tuvo en cuenta en el desarrollo del software.

5.2.2 Velocidad de procesamiento

Ciertas funciones del programa, como la función de crecimiento de regiones, pueden tardar un tiempo considerable en finalizar su ejecución, debido a su naturaleza de iteración sobre cada píxel en la imagen, llegando a durar hasta decenas de segundos. Este tiempo de calculo y, por ende, el rendimiento del programa, se verá mejorado si se utiliza un equipo de mejores características. El software fue diseñado y probado en principalmente en un equipo con procesador Intel Core I3 de 4° generación a



(a) (b)

Figura 5.12: Información inicial. *(a)* Datos obtenidos a partir del software. *(b)* Datos arrojados por el microscopio especular.

1.6Ghz. Sin embargo, básicamente cualquier equipo es capaz de ejecutar el programa, teniendo en cuenta que los tiempos de espera pueden llegar a ser más altos o más bajos, dependiendo de sus especificaciones técnicas.

5.2.3 Zona de extensión del crecimiento de regiones

En ocasiones, el área obtenida como resultado de utilizar la herramienta *Guttae Seed* es mayor a la deseada e inclusive puede sobrepasar los limites iniciales de la segmentación (Figura 5.13). Esto se debe al bajo contraste y la iluminación uniforme de la imagen, lo que dificulta el cálculo al hacer uso de este método de crecimiento de regiones.

Para corregir esto, como trabajo futuro se planea lograr una segmentación más acertada, usando métodos más robustos como técnicas redes convolucionales profundas.

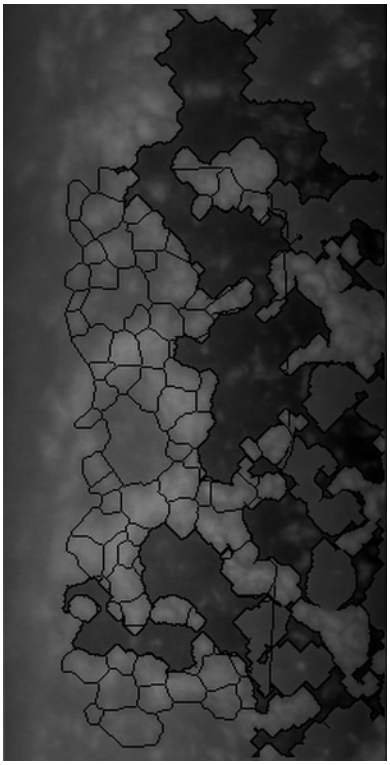


Figura 5.13: Mala extensión de crecimiento de regiones.

6 Conclusiones

En este trabajo se desarrolló satisfactoriamente un software capaz de modificar segmentaciones de células del endotelio corneal calculadas por el microscopio especular SP-3000P, mediante el uso de operaciones de morfología matemática y técnicas procesamiento de imágenes digitales. A partir de esto se puede concluir que:

1. Los parámetros obtenidos por el software son una mejor referencia a la hora de evaluar el estado de la córnea del paciente, puesto que excluye las guttas del conteo celular, pero incluye el área en el cálculo de densidad celular.

2. Existen muchos métodos de segmentación celular para obtener datos de tamaño y densidad celular, pero estos no incluyen el cálculo de dichos parámetros en las guttas, por lo que este software a demás de ser útil para la medicina, puesto que brinda a los médicos información más acertada respecto a la salud de la córnea del paciente, resulta innovador.

Como trabajo futuro, se planea hacer más robustas algunas funciones que dependen de la luminosidad de la imagen y el contraste en la misma; además, se tomará el código desarrollado en este trabajo como base para implementar nuevos módulos al software, de manera que este sea capaz de realizar su propia segmentación celular a partir de la imagen del endotelio corneal e identificar las guttas de manera automática, reduciendo así el tiempo que tarda el usuario en hacer una corrección de segmentación.

Bibliografía

Bourne, William M y Herbert E Kaufman (1976). „Specular microscopy of human corneal endothelium in vivo“. En: *American journal of ophthalmology* 81.3, págs. 319-323 (vid. pág. 1).

Cassin, Barbara, Sheila Solomon y Melvin L Rubin (1984). *Dictionary of eye terminology*. Triad Pub. Co. (vid. pág. 13).

Corporation, Topcon (s.f.). *Product information SP-3000P*. URL: https://www.topcomed.cz/pdf/SP_3000P.pdf (vid. págs. 13, 14).

Giasson, Claude J, Andrew Graham, Jean-François Blouin y col. (2005). „Morphometry of cells and guttae in subjects with normal or guttate endothelium with a contour detection algorithm“. En: *Eye & contact lens* 31.4, págs. 158-165 (vid. pág. 1).

Gonzalez, RC (2004). „Digital Image Processing Using Matlab-Gonzalez Woods & Eddins. pdf. Education“. En: (vid. págs. 5, 6, 8-11).

Maruoka, Sachiko, Shunsuke Nakakura, Naoko Matsuo y col. (2018). „Comparison of semi-automated center-dot and fully automated endothelial cell analyses from specular microscopy images“. En: *International ophthalmology*, págs. 1-13 (vid. pág. 2).

Oftalmología Barran, Centro de (s.f.). *Córnea Guttata y distrofia de Fuchs*. URL: https://www.barraquer.com/que-tratamos/cornea-guttata-y-distrofia-de-fuchs/ (vid. pág. 2).

Protter, Murray H y Charles Bradfield Morrey (1977). *College calculus with analytic geometry*. Addison-Wesley (vid. pág. 7).

Sanchez-Marin, Francisco J (1999). „Automatic segmentation of contours of corneal cells“. En: *Computers in biology and medicine* 29.4, págs. 243-258 (vid. pág. 1).

Vincent, Luc M y Barry R Masters (1992). „Morphological image processing and network analysis of cornea endothelial cell images“. En: *Image Algebra and Morphological Image Processing III*. Vol. 1769. International Society for Optics y Photonics, págs. 212-227 (vid. pág. 1).