
# ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

## FACULTAD DE INGENIERIA EN ELECTRICIDAD Y COMPUTACIÓN

"Implementación de un algoritmo para la detección y conteo de células en imágenes microscópicas"

## INFORME DE MATERIA DE GRADUACIÓN

Previo a la obtención del Título de:

## INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentado por:

José Joaquín Moreira Quiroz

Pedro Vladimir Valencia Delgado

GUAYAQUIL – ECUADOR

AÑO: 2009

# AGRADECIMIENTO

Nuestra imperecedera gratitud: A Dios, por sus bendiciones, que conllevaron a la culminación de esta meta; a nuestra Alma Máter y a los profesores que durante nuestra carrera nos brindaron su amistad y sus conocimientos. A nuestros amigos y a todas las personas que colaboraron con el presente trabajo, especialmente a la Ing. Patricia Chávez por su invaluable ayuda y consejos.

# DEDICATORIA

Con el más grande afecto: A Dios, mis padres, mis hermanos, y familiares por su esfuerzo, aliento y el apoyo incondicional que me han brindado en cada instante de mi vida, lo que me permite hoy obtener este logro.

A mi madrina, Sra. Rosa Moreira y sus hijos, por haberme asistido en el camino que me ha llevado a dar este importante paso en mi vida.

**José J. Moreira Quiroz**

A Dios, por que sin él, nada podemos hacer. A mis padres, por ser la razón de seguir adelante en los momentos que quiero desmayar; y sobre todo porque me enseñaron que un Valencia Delgado, todo lo puede lograr. A mis hermanos y hermanas por su amor y ayuda a cada instante. A mi cuñada Gina, por su amistad y consejos. A mis sobrinos y sobrinas que han traído alegría a nuestras vidas. A mis abuelitas que están en el cielo, Patria y Anita por su amor, consejos y enseñanzas dejadas. A mi abuelita Teresa por todo su cariño y amor, en los momentos que la tuve conmigo. A mi tía Matilde, y sus hijas Carmita y Teodorita, por estar junto a mi familia en todo momento. A mi tía Betty, por todo su amor y aprecio para conmigo.

**Pedro Vladimir Valencia Delgado**

# TRIBUNAL DE SUSTENTACION

MSc. Patricia Chavez
PROFESOR DE LA MATERIA

MSc. Juan Carlos Avilés
PROFESOR DELEGADO DEL DECANO

# DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este Informe, me corresponde exclusivamente; y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL".

(REGLAMENTO DE GRADUACIÓN DE LA ESPOL)

_____
José Joaquín Moreira Quiroz

_____
Pedro Vladimir Valencia Delgado

# RESUMEN

En el presente trabajo, presentamos el desarrollo de una aplicación para el conteo de células en imágenes obtenidas mediante microscopio, en la cual se hace posible la utilización de 2 algoritmos (métodos) de segmentación y de 2 de los principales operadores para detectar bordes en imágenes. Previamente, se analizarán los fundamentos teóricos en los que se basa la segmentación de imágenes.

En el capítulo 1 se presentan algunas definiciones básicas sobre imágenes, entre las que se incluyen el concepto de Imagen Digital y el Filtrado de las mismas.

El capítulo 2 abarca el proceso de Segmentación de Imágenes, proceso en el que se fundamenta nuestra aplicación, e incluye los métodos para la detección de bordes, explicando como estos operan en las imágenes a través de la convolución.

El capitulo 3 se explica brevemente el procedimiento empleado para contabilizar las células una vez, que se obtiene la imagen binaria, producto de la segmentación.

En el capítulo final se presenta el desarrollo de la aplicación, junto con todos los aspectos que se tomaron en consideración para la realización de la misma.

Adicionalmente se presenta el funcionamiento de la interfaz gráfica en la cual el usuario podrá seleccionar el método de segmentación, así como el operador que utilizará para la detección de bordes.

# INDICE GENERAL

# INDICE DE FIGURAS



# INDICE DE TABLAS

## Introducción

En las ciencias médicas, existen varios tipos de exámenes, como el conteo de células T, conteo de glóbulos rojos o el Frotis de sangre que requieren determinar la cantidad de células presentes en una muestra, esto con el fin de descartar o confirmar la presencia de alguna enfermedad o cuando el médico sospecha de una anomalía de algún tipo de célula.

Uno de los métodos que se emplea en la actualidad se denomina el de la Cámara de contaje celular o Cámara de Neubauer. La exactitud y la velocidad con que se obtiene el resultado de este examen dependen en parte, de la experiencia de la persona que examina la muestra, ya que el conteo se realiza de forma manual.

En la búsqueda de una mejora de este tipo de procedimiento nos hemos propuesto desarrollar una forma diferente de conteo de células a través de un programa que permita acelerar y aportar un mayor grado de exactitud al proceso de conteo de células utilizando para esto imágenes de células obtenidas a través de un microscopio.

Para ello, nuestro trabajo se basará en técnicas de segmentación y operadores para la detección de bordes, los cuales analizaremos para encontrar los que mejor se adapten a nuestro requerimiento.

# Capítulo 1

# 1. Conceptos Generales

El procesamiento digital de señales nos permite extraer información considerada relevante, o modificarla para darle un uso apropiado, empleando para ello herramientas computacionales.

Se puede emplear el procesamiento digital de señales en una gran variedad de aplicaciones tales como: transmisión de información a través de canales de comunicación, restauración y mejoramiento de imágenes y de audio, reconocimiento de objetos en imágenes, medición de parámetros tales como la velocidad en objetos en desplazamiento capturados en video, etc.

Nuestro proyecto empleará la detección de objetos basada a su vez en el proceso conocido como *Segmentación de Imágenes*, el cual será analizado en un capitulo posterior.

## 1.1 Definición de Imagen Digital

Una imagen digital, puede ser definida como una función en dos dimensiones, *f(x,y)*, donde: *x* e *y* son coordenadas espaciales, la amplitud de

*f* en el punto *(x,y)* es conocida como la intensidad de la imagen en ese punto. Tanto, los valores de las coordenadas *x* e *y,* como de la amplitud de la función *f* son valores finitos. El término *nivel de gris*, se lo emplea para referirse a la intensidad en imágenes monocromáticas.

Las imágenes a color, están formadas por una combinación de imágenes individuales en dos dimensiones.

Es de notar, que una imagen está compuesta por un número finito de elementos (puntos) con una ubicación y valor de intensidad particular, donde cada elemento es conocido como elemento de imagen, elemento de foto, pels o pixel. Pixel es el término más ampliamente usado para denotar un elemento en una imagen digital.

## 1.2 Transformación de Intensidad

Una transformación espacial opera directamente sobre los pixeles de una imagen. La transformación de intensidad en el dominio espacial, es denotada por la expresión:

$$g(x, y) = T[f(x,y)] \qquad (1.1)$$

Donde *f(x,y)* es la imagen sobre la cual se aplicará la transformación, *g(x,y)* es la imagen procesada y *T* es un operador sobre *f* , definido sobre un vecindario especificado cerca al punto *(x,y)*. *T* puede operar sobre un conjunto de imágenes, por ejemplo sumando *K* imágenes para provocar una reducción de ruido.

La mejor aproximación para definir un vecindario espacial cerca al punto es usar un cuadrado o región rectangular centrada en *(x,y)*. El centro de la

región es movida de pixel a pixel, empezando en la parte superior izquierda de la imagen y al moverse abarca diferentes vecindarios. El operador *T* es aplicado en cada ubicación de *(x,y)* para producir la salida *g* en esa ubicación. Solo los pixeles del vecindario son usados para computar el valor de *g* en *(x,y)*.

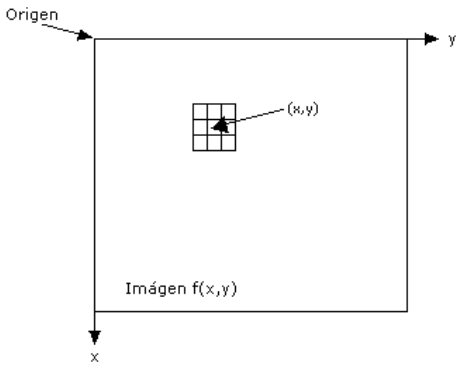


**Figura 1.1: Vecindario de 3x3 con centro en el punto (x,y)**

La transformación más simple se da cuando el vecindario en la figura 1.1 es de tamaño 1x1 (un solo pixel). En este caso, el valor de g en *(x,y)* depende solo de la intensidad de *f* en ese punto, y *T* se convierte en la función transformación de *intensidad* o *nivel de gris*. Estos dos términos son intercambiables cuando se usan imágenes monocromáticas (Ej. Escala de grises). Cuando se trata con imágenes de color, el término *intensidad* se refiere a uno de los componentes de una imagen a color.

Una forma simplificada de referirse a la transformación, es:

$$S = T(r) \qquad (1.2)$$

Donde *r* denota la intensidad de *f* y *S* la intensidad de *g*, ambos en cualquier punto *(x,y)* de la imagen.

## 1.3 Filtrado espacial

El filtrado espacial consiste básicamente en:

1) Seleccionar un punto central, *(x,y).*
2) Desarrollar una operación que involucre solo los pixeles en un vecindario predefinido cerca del punto central.
3) Dejar que el resultado de esa operación sea la “respuesta” del proceso en ese punto.
4) Repetir el proceso para cada punto en la imagen.

El proceso de mover el punto central crea nuevos vecindarios, uno para cada pixel en la imagen *f*. Si los cálculos realizados sobre los pixeles de un vecindario son lineales, la operación es llamada *filtrado espacial lineal,* de otra manera es llamada *filtrado espacial no lineal*. El *filtrado espacial* también es conocido como *procesamiento de vecindario*.

### 1.3.1 Filtrado Espacial Lineal

Las operaciones lineales consisten en multiplicar cada pixel en el vecindario por un correspondiente coeficiente y sumar los resultados para obtener la respuesta en cada punto *(x,y)*. Si el vecindario es de tamaño m x n, se necesitarán mn coeficientes. Los coeficientes son ordenados como una matriz llamada *filtro*, *máscara*, *máscara de filtro*, *kernel*, *plantilla* o *ventana*. Otro término empleado para referirse a la matriz de coeficientes es *filtro de convolución*.

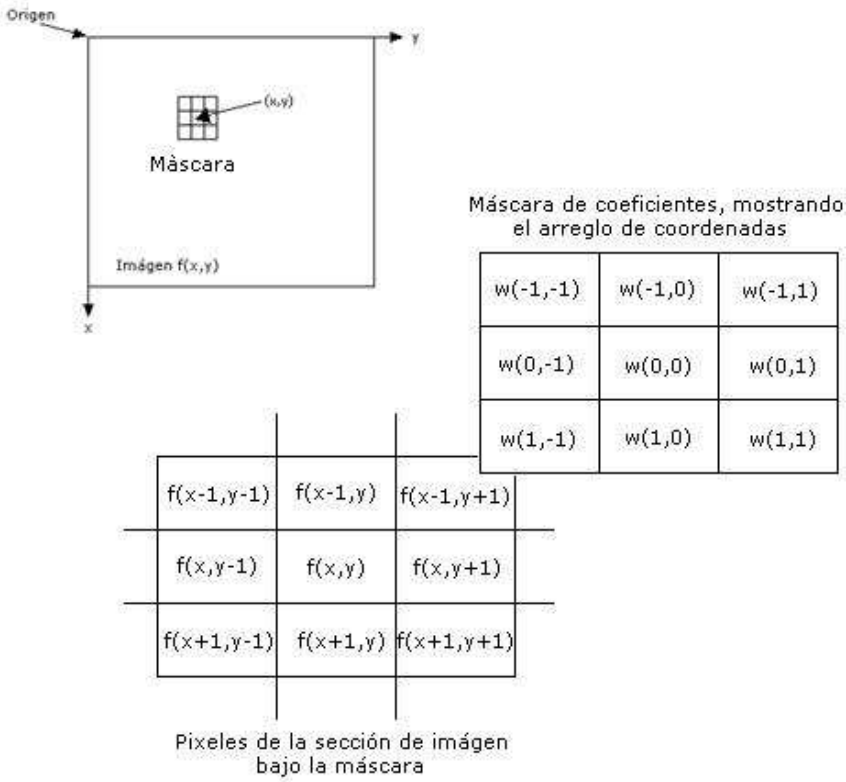


**Figura 1.2: Mecanismo del filtrado espacial**

El procedimiento del *filtrado espacial lineal* se muestra en la figura 1.2. El proceso consiste en mover el centro de la máscara de filtro *w*, sobre cada punto de la imagen *f*. En cada punto *(x,y)*, la respuesta del filtro en ese punto es la suma de los productos de los coeficientes del filtro y los correspondientes pixeles del vecindario en el área abarcada por la máscara de filtro. Normalmente se utilizan máscaras de tamaño impar, esto es que, para una máscara de tamaño m x n, se asume m = 2a + 1 y n = 2b + 1, donde a y b son enteros positivos, esto con el fin de facilitar el trabajo con ellas ya que tienen un único punto central. Sin embargo no se descarta el uso de máscaras de tamaño par.

Existen dos conceptos relacionados estrechamente que deben ser entendidos claramente cuando se desarrolla el filtrado espacial lineal. Uno es la *correlación*; el otro es la *convolución*. La Correlación es el proceso de pasar la máscara *w* por el arreglo de imagen f, en la forma descrita en la figura 1.2. La Convolución es el mismo proceso, excepto que *w* es rotado 180º antes de ser pasada sobre *f*.

### 1.3.2 Explicación Gráfica de la Correlación y de la Convolución

Para esta explicación emplearemos la figura 1.3. La figura 1.3(a) será la función que representará la imagen que vamos a filtrar. La función *w(x,y)* en la figura 1.3(b) será empleada como máscara de filtro en la Correlación o en el caso de realizar la Convolución, la máscara que se utilizará será la representada por la función *–w(x,y)* en la figura 1.3(c), que es la versión rotada 180º de la función *w(x,y)*.

En la figura 1.3(d) se muestra la función *f(x,y)* a la cual se la ha rellenado con dos hileras de ceros en su contorno, esto con el fin de que ninguno de los elementos en la máscara se quede sin un elemento correspondiente en la imagen con el cual traslaparse.

En la figura 1.3(e) se muestra el inicio de la Correlación, donde se observa que esta empieza con el elemento en la parte inferior derecha de la máscara (número 9) traslapado con el elemento en el origen de la función *f(x,y)* (número 0). La ubicación del elemento central de la máscara (número 5), (que se encuentra resaltado en rojo) será finalmente la ubicación del resultado de la Correlación. Las figuras

1.3(f) y 1.3(g) muestran el desplazamiento que se realiza de la máscara sobre la imagen. La figura 1.3(h) muestra un instante antes de que la máscara se traslape con el primer elemento diferente de cero en la imagen (el número 1), la figura 1.3(i), cuando se ha producido este traslape y en la figura 1.3(j), se muestra el instante en que el último elemento de la máscara se ha traslapado con el único elemento diferente de cero en la imagen. Finalmente, en la figura 1.3(k) se aprecia el resultado de la correlación desarrollada. En caso de utilizar la máscara de la figura 1.3(c), lo que se llevará a cabo es la *convolución*, cuyo resultado se puede apreciar en la Figura 1.3(l).

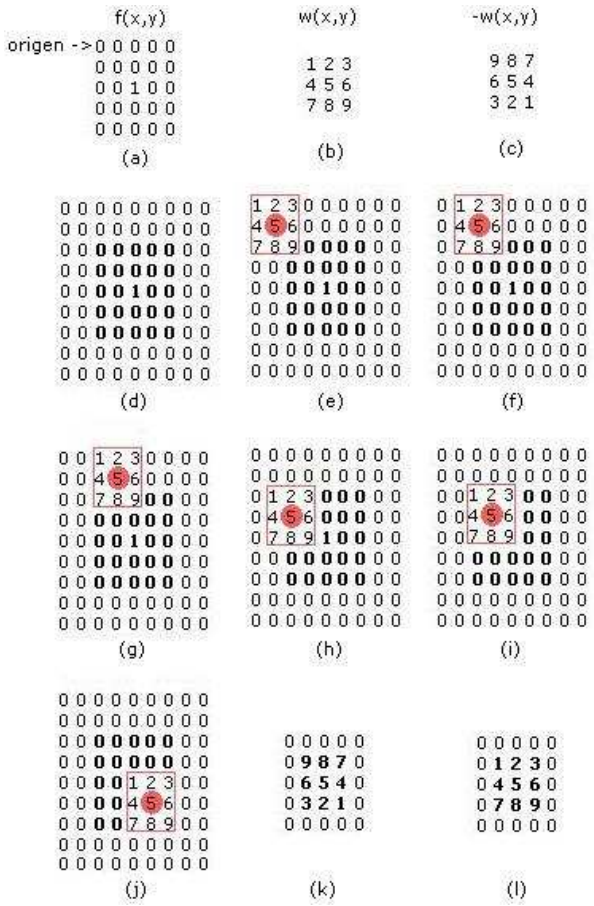


*Figura 1.3: Explicación Gráfica de la Correlación y de la Convolución*

# Capítulo 2

# 2. Segmentación de Imágenes

La segmentación, es un proceso en el cual una imagen es sub-dividida en las regiones u objetos que la componen. La segmentación concluye cuando los objetos de interés han sido aislados. En nuestro caso, la segmentación debería de concluir cuando las células de las imágenes, hayan sido aisladas.

Los algoritmos de segmentación para imágenes monocromáticas generalmente están basados en una de dos características básicas de los valores de intensidad en las imágenes: discontinuidad y similaridad.

En la primera categoría, y en la cual estará basado nuestro proyecto, el enfoque es para particionar una imagen basado en cambios abruptos de intensidad, tal como ocurre en los bordes de una imagen. La detección de bordes, es uno de los algoritmos básicos que existen en el tema de segmentación. Los principales enfoques en la segunda categoría están basados en la partición de la imagen en regiones que son similares de acuerdo a un conjunto de criterios predefinidos.

## 2.1 Detección de Discontinuidades basada en valores de Intensidad

La forma básica para encontrar discontinuidades, es pasar una máscara a través de la imagen, en la forma descrita en las secciones 1.3.1 y 1.32. Para

una máscara de 3x3 este procedimiento involucra computar la suma de productos de los coeficientes con los niveles de intensidad contenidos en la región abarcada por la máscara. Esto es, la respuesta *R* de la máscara en cualquier punto (*x*, *y*) de la imagen está dado por:

$$R = \sum_{k=1}^{9} w_k z_k \quad (2.1)$$

$$R = w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9 \quad (2.2)$$

Donde $z_k$ es la intensidad del pixel asociado con el coeficiente $w_k$ de la máscara.

## 2.2 Detección de Bordes

Al ser un borde, dentro de una imagen, un tipo de discontinuidad, esta puede ser detectada usando derivadas de primer y segundo orden.

La derivada de primer orden para una imagen es el *gradiente*, el cual, para una función en dos dimensiones, *f(x,y)* viene dado por:

$$\boldsymbol{\nabla f} = \begin{bmatrix} G_x & G_y \end{bmatrix} = \begin{bmatrix} \frac{df}{dx} & \frac{df}{dy} \end{bmatrix} \quad (2.3)$$

La magnitud de este vector es:

$$\nabla f = mag(\nabla f) = \left[G_x^2 + G_y^2\right]^{1/2} \quad (2.4)$$

$$= \left[\left(\frac{df}{dx}\right)^2 + \left(\frac{df}{dy}\right)^2\right]^{1/2} \quad (2.5)$$

Esta cantidad se puede simplificar en algunos casos, omitiendo la raíz cuadrada de la operación:

$$\nabla f = G_x^2 + G_y^2 \quad (2.6)$$

O usando el valor absoluto de los términos:

$$\nabla f = |G_x| + \left|G_y\right| \quad (2.7)$$

Esta aproximación, todavía se comporta como una derivada, esto quiere decir, que se hace cero en áreas de intensidad constante y sus valores son proporcionales al grado de cambio de intensidad, en áreas donde los valores de los pixeles son variables. Es común referirse a la magnitud del gradiente o su aproximación simplemente como “el gradiente”.

Una propiedad fundamental del vector gradiente es que este, apunta en la dirección de la tasa máxima de cambio de *f* en las coordenadas *(x,y)*. El ángulo en el cual esta tasa máxima de cambio ocurre, es:

$$\theta(x,y) = tan^{-1}\left(\frac{G_y}{G_x}\right) \quad (2.8)$$

### 2.2.1 Detector de bordes *Sobel*

El detector de borde *Sobel* usa la máscara que se muestra en la figura 2.2 para aproximar digitalmente las primeras derivadas $G_x$ y $G_y$. En otras palabras, el gradiente en el punto central de un vecindario es computado por el detector *Sobel* como sigue:

$$\mathrm{g} = \left[G_x^2 + G_y^2\right]^{\frac{1}{2}} \quad (2.9)$$

$$= \{[(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)]^2 + [(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)]^2\}^{1/2} \quad (2.10)$$

Entonces, decimos que un pixel en la ubicación *(x,y)* es un pixel de borde si g≥*T* en esa ubicación, donde *T* es un umbral que debe ser especificado.

| | | |
|---|---|---|
| z1 | z2 | z3 |
| z4 | z5 | z6 |
| z7 | z8 | z9 |

**Figura 2.1: Vecindario de la imagen sobre la cual se aplica el operador**

| | | |
|---|---|---|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

Gx = (z7+2z8+z9 ) - (z1+2z2+z3 )

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

Gy = (z3+2z6+z9 ) - (z1+2z4+z7 )

**Figura 2.2: Máscaras del operador Sobel**

### 2.2.2 Laplaciano de un detector Gausiano (LoG)

Consideremos la función Gausiana:

$$h(r) = -e^{\frac{r^2}{2\sigma^2}} \qquad (2.11)$$

Donde $r^2 = x^2+y^2$ y $\sigma$ es la desviación estándar. Si se convoluciona esta función con una imagen la volverá borrosa. El grado de borrosidad es determinado por $\sigma$. El Laplaciano de esta función (la segunda derivada con respecto a $r$) es:

$$\nabla^2 \mathrm{h(r)} = -\left(\frac{\mathrm{r^2-\sigma^2}}{\mathrm{\sigma^4}}\right) e^{\frac{r^2}{2\sigma^2}} \qquad (2.12)$$

Ahora se hace obvio el por qué este detector es llamado el Laplaciano de un Gausiano.

Ya que la segunda derivada es una operación lineal, al convolucionar (filtrar) una imagen con $\nabla^2 \mathrm{h(r)}$ es lo mismo que convolucionar primero la imagen con la función h(r), y luego computar el Laplaciano del

resultado. Este es el concepto clave del detector LoG. Convolucionamos la imagen con $\nabla^2 \mathrm{h}(\mathrm{r})$, sabiendo que esto tiene dos efectos: suaviza la imagen (reduciendo el ruido), y computa el Laplaciano, el cual produce una imagen con bordes dobles. Entonces, la tarea de localizar los bordes se centra en encontrar el cruce por cero entre los bordes dobles.

### 2.2.3 Detector de bordes *Canny*

El detector *Canny* es de todos, el más potente detector de bordes, debido a su eficacia. El método que emplea para detectar bordes puede resumirse como sigue:

1. La imagen es suavizada usando un filtro Gausiano con una desviación estándar, esto para reducir el ruido.
2. El gradiente local, $\mathrm{g} = \left[G_x^2 + G_y^2\right]^{\frac{1}{2}}$ y la dirección del borde $\theta(x, y) = tan^{-1}\left(\frac{G_y}{G_x}\right)$, son computadas en cada punto. Cualquier operador, de los anteriormente citados, puede ser usado para computar $G_x$ y $G_y$. Un punto de borde se define como un punto cuyo peso es localmente máximo en la dirección del gradiente.
3. Los puntos de borde determinados en (2) dan elevación a crestas en la magnitud del gradiente de la imagen. El algoritmo luego rastrea a lo largo de la cima de estas crestas, y lleva a cero los pixeles que no están en realidad sobre la cima de la cresta, produciendo una línea delgada en la salida, un proceso conocido

como *supresión no máxima*. Los pixeles de crestas son luego comparados usando dos umbrales, *T1* y *T2* con *T1* < *T2*. Los pixeles de crestas con valores mayores que *T2* se dice que son “probables candidatos” para ser pixeles de borde. Los pixeles en crestas con valores entre *T1* y *T2 se dice que son “candidatos poco probables”* para ser pixeles de borde.

4. Finalmente, el algoritmo realiza la unión de los pixeles incorporando *“candidatos débiles”* que están 8-conectados a los pixeles “probables”.

# Capítulo 3

# 3. Etiquetado de componentes en imágenes binarias

Una representación común de una imagen digital en dos dimensiones, es el de un arreglo de pixeles asociados con colores en un mapa de color. El caso particular de una imagen digital que puede ser representada usando un mapa de dos colores (*Primer Plano* y *Fondo)* define la clase de imágenes digitales binarias. En las imágenes binarias, la información está contenida en forma de componentes y en la interrelación entre ellos.

## 3.1 Definición de Primer Plano en una Imagen

El *Primer Plano* es el conjunto *F* de puntos los cuales poseen valores iguales a 1. Por convención, el *Primer Plano* corresponde al conjunto de pixeles negros en una imagen binaria.

## 3.2 Definición de Fondo de una Imagen

El *Fondo* es el complemento del conjunto *F*, denotado por $F^c$. Este, es el conjunto de puntos asociados con un valor cero. Por convención, el fondo corresponde al conjunto de todos los pixeles blancos en la imagen.

## 3.3 Vecindad

Un pixel *p* con coordenadas *(x,y)* tiene dos vecinos verticales y dos horizontales cuyas coordenadas son *(x+1, y)*, *(x-1,y)*, *(x,y+1)* y *(x,y-1)*. Este conjunto de 4-vecinos de *p*, denotado, $V_4(p)$ se muestra en la figura 3.1(a).

Los cuatro vecinos diagonales de *p* tienen coordenadas *(x+1, y+1), (x+1, y-1), (x-1, y+1)* y *(x-1, y-1)*. Este conjunto de puntos denotado como $D_4(p)$ es mostrado en la figura 3.1(b). La unión de $V_4(p)$ y $D_4(p)$, en la figura 3.1(c), son los *8-vecinos* de *p*, denotado como $V_8(p)$.

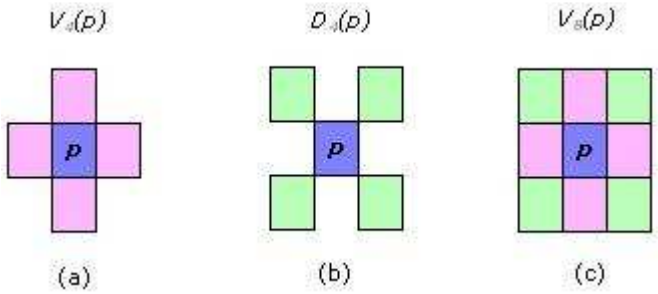


**Figura 3.1: Tipos de vecindad**

## 3.4 Adyacencia

Dos pixeles *p* y *q* se dice que son *4-adyacentes* si $q \in V_4(p)$. De igual manera, se dice que *p* y *q* son *8-adyacentes* si $q \in V_8(p)$.

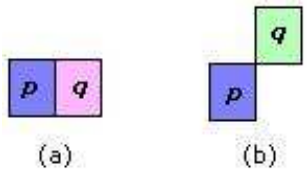


**Figura 3.2: Tipos de adyacencia**

En la figura 3.2(a), p y q son *4-adyacentes* y *8-adyacentes*, mientras que en la figura 3.2(b) p y q son *8-adyacentes* pero no *4-adyacentes*.

### 3.5 Caminos

Un *camino* entre los pixeles $p_1$ y $p_2$, es una secuencia de pixeles $p_1, p_2, \ldots p_{n-1}, p_n$ tal que $p_k$ es adyacente a $p_{k+1}$, para 1≤ k ≤n. Un camino puede ser *4-conectado* u *8-conectado*, dependiendo de la definición de adyacencia que se utilice. En la figura 3.3(a), se muestra un camino *4-conectado*, donde algunos puntos (pixeles) no han sido tomados en cuenta aunque forman parte del primer plano de la imagen, ya que estos no cumplen la condición de ser *4-adyacentes*. Así mismo, en la figura 3.3(b) se muestra un camino 8-*conectado*, donde también existen puntos (pixeles) que no han sido tomados en cuenta, en este caso por no cumplir la condición de ser 8-*adyacentes*.

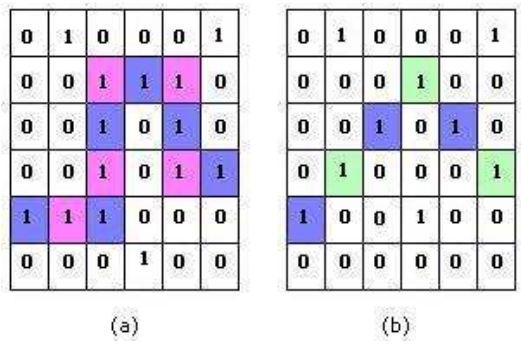


**Figura 3.3: Tipos de caminos**

Dos pixeles *p* y *q,* pertenecientes al primer plano, se dice que son *4-conectado*, si existe entre ellos un camino *4-conectado*, consistente enteramente de pixeles de primer plano. En las figuras 3.3(a), cualquier par de puntos dentro del camino resaltado, pueden ser considerados como *4-conectados*. A su vez, los pixeles *p* y *q* son *8-conectados* si existe un camino

*8-conectado* entre ellos. En la figura 3.3(b), cualquier par de puntos dentro del camino resaltado, pueden ser considerados como *8-conectados*.

## 3.6 Componentes conectados

Para cualquier pixel *p* del primer plano, el conjunto de todos los pixeles pertenecientes al *primer plano* y conectados al pixel *p*, es llamado el *componente conectado* que contiene a *p*.

El término *componente conectado* está definido en términos de un camino, y la definición de un camino depende, como ya lo analizamos, del tipo de adyacencia que se utilice. Esto implica que la naturaleza de un *componente conectado*, depende de qué tipo de adyacencia se escoja, siendo 4- y 8-adyacencia las más utilizadas.

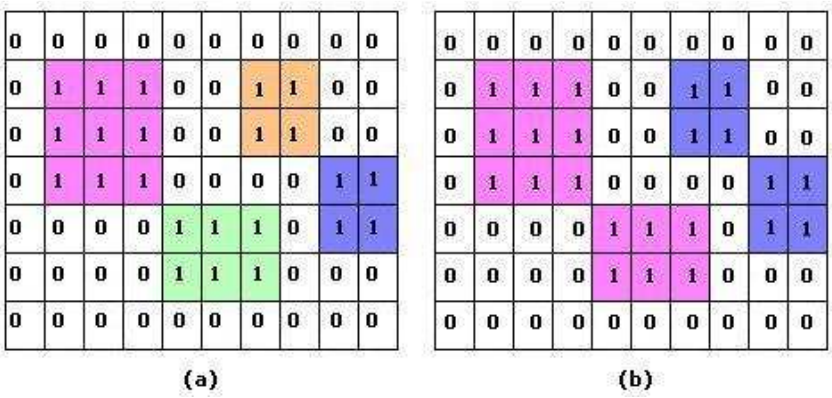


**Figura 3.4: Componentes conectados**

En la figura 3.4, se muestran la diferencia de resultados al utilizar 4- y 8-adyacencia. En la figura 3.4(a) se obtienen 4 componentes *4-conectados*,

producto de usar *4-adyacencia*. En la figura 3.4(b), se obtienen 2 componentes *8-conectados*, producto de usar *8-adyacencia*.

### 3.7 Definición de Borde

Dado un conjunto *P* de puntos *k-conectados*, el complemento de *P* denotado por $P^C$, se define una relación de conectividad dual (denotada *k'-conectividad,* con *k*=8 y *k'*=4). El *borde* de *P* es el conjunto de puntos *B* definido como el conjunto de puntos *k-conectados* en *P* que tienen al menos un *k'-vecino* en $P^c$.

### 3.8 Etiquetado de Segmentos

El resultado de la segmentación satisfactoria de una imagen es el etiquetado de cada pixel correspondiente a un segmento determinado. El objetivo del etiquetado es asignar a cada pixel de una imagen el número de etiqueta o índice del segmento al que pertenece.

De manera general este procedimiento puede ser descrito de la siguiente manera:

Dada una imagen binaria y el conjunto de pixeles pertenecientes al *primer plano F* de la imagen, la meta es asociar una etiqueta de componente LABEL(p) a cada pixel *p* en la imagen, tal que:

1) LABEL(p) = 0 si $p \in F^c$ y LABEL(p) = +∞ si $p \in F$
2) LABEL(p) = LABEL(q) si y solo si los pixeles $p \in F$ y $q \in F$ están en el mismo componente conectado. De otra forma LABEL(p) ≠ LABEL(q).

El mapa de etiquetas $(LABEL(p))_p^{(0)}$ primero es inicializado usando la siguiente ecuación:

$$\begin{cases} LABEL^{(0)}(p) = +\infty & \forall p \epsilon F \\ LABEL^{(0)}(p) = 0 & \forall p \epsilon F^c \end{cases} \quad (3.1)$$

Luego, para una iteración en *t>0*, los valores en el mapa de etiquetas $(LABEL(p))_p^{(t)}$ son actualizados usando la siguiente ecuación:

$$LABEL^{(t)}(p) = min\{LABEL^{(t-1)}(p)\ ;\ \lambda\ ;\ L_{min}(p)\} \quad (3.2)$$

Donde

$$L_{min}(p) = \min\ \{\ LABEL^{(t-1)}(q)/q = (x_p + k, y_p + k)\ \epsilon F\} \quad (3.3)$$

Y el parámetro $\lambda$>0 es un contador de componentes que es incrementado en 1 cada vez que el valor de $LABEL^{(t)}(p)$ se establece como $\lambda$. El valor del parámetro $\lambda$ es siempre mayor en 1 que el número de componentes detectados. Este proceso es repetido hasta que no existan cambios que realizar en esta iteración. Al finalizar el proceso de etiquetado de los componentes, los pixeles *p* asociados con etiquetas LABEL(p)=0 pertenece al fondo, es decir a $F^c$. Todos los pixeles *p* en un componente conectado dado de *F* están asociados con la misma etiqueta LABEL(p)>0.

# Capítulo 4

# 4. Desarrollo de la Aplicación

En esta aplicación, hemos trabajado con una serie de imágenes de células obtenidas en análisis de diferentes fluidos y tejidos, encontradas en su mayoría en la Internet, que para propósitos de demostrar el desempeño de nuestra aplicación, son aceptables.

Hemos desarrollado dos métodos para el conteo de células, a los que hemos llamado *Método 1* y *Método 2* ya que encontramos que para determinadas imágenes se obtienen mejores resultados con un método que con el otro.

Para la realización de la misma, hemos empleado el muy conocido paquete computacional Matlab 7.7.0 (R2008b), de la compañía MathWorks, el que dispone de una serie de funciones que facilitan el procesamiento de imágenes.

## 4.1 Estructura de la Aplicación

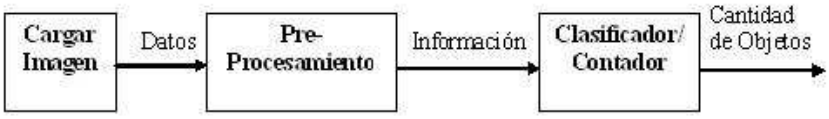


**Figura 4.1: Diagrama en bloques de la Aplicación**

El proceso que seguimos para obtener el número de células en las imágenes, se muestra en la figura 4.1, y la parte principal del mismo se presenta en el procesamiento que efectuamos sobre las imágenes, ya que de esto depende la calidad del resultado del proceso.

Las imágenes que se presentarán a en la siguiente sección, y con las cuales explicaremos el proceso, son las que obtuvimos empleando el detector de bordes *Canny*, sin embargo, nuestra aplicación adicionalmente, da la opción al usuario de utilizar también el detector de bordes *Sobel*.

## 4.2 Procesamiento de las imágenes: Explicación del Método 1

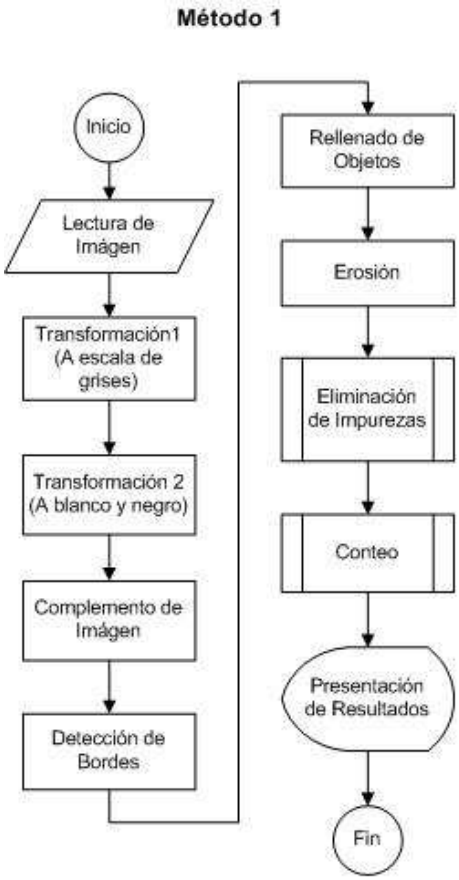


**Figura 4.2: Algoritmo Método 1**

El algoritmo de conteo utilizando el método 1 se muestra en la figura 4.2. Empieza con la adquisición de la imagen, este que no es un paso complicado, lo realizamos utilizando la función *imread*, y emplearemos la imagen que se muestra en la figura 4.3[1].

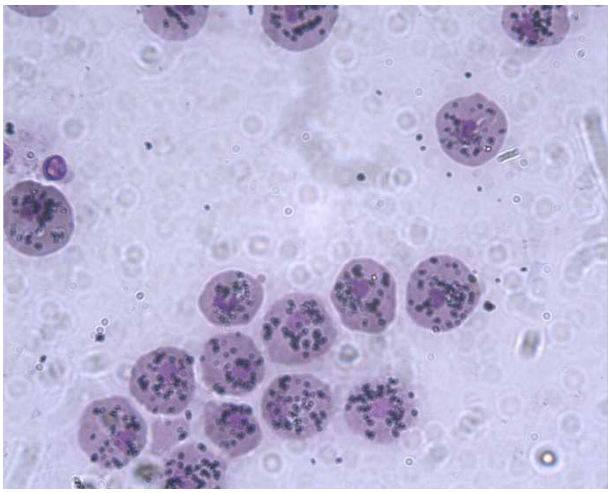

**Figura 4.3: Imagen de un conjunto de células**

Enseguida, llevamos la imagen a escala de grises con la función *rgb2gra*y aplicándole un ajuste a la imagen con la función *imadjust*, con umbrales de 0.2 y 0.8, siendo el resultado el que se muestra en la figura 4.4.

Enseguida, convertimos la imagen en una imagen en blanco y negro con la función *im2bw*. El resultado que se obtiene se muestra en la figura 4.5.

En este punto, son detectados los bordes que existen en el interior de la imagen. Para ello empleamos la función *edge* junto con el operador que

[1] Fuente: Dr. Marcelo Muñoz, F.I.M.C.M - ESPOL

seleccionemos, que para el caso de nuestra explicación, es el operador *Canny*.

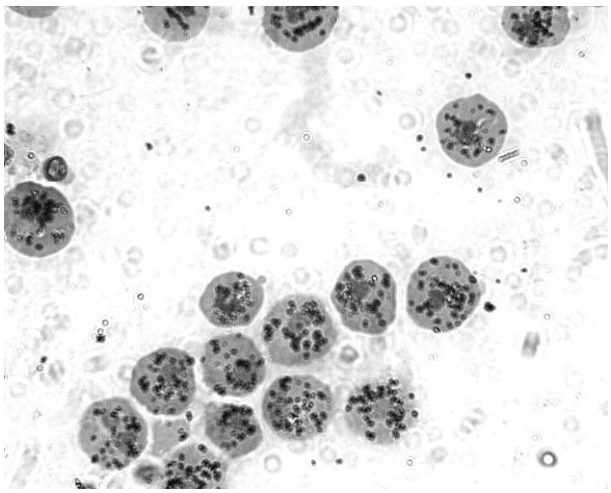

**Figura 4.4: Imagen llevada a escala de gris**

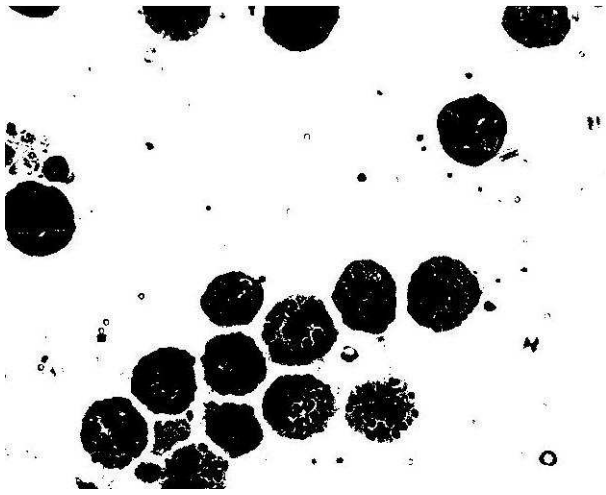

**Figura 4.5: Imagen en blanco y negro**

A continuación obtenemos el complemento de la imagen en la figura 4.5 utilizando la función *imcomplement*. El resultado se muestra en la figura 4.6.

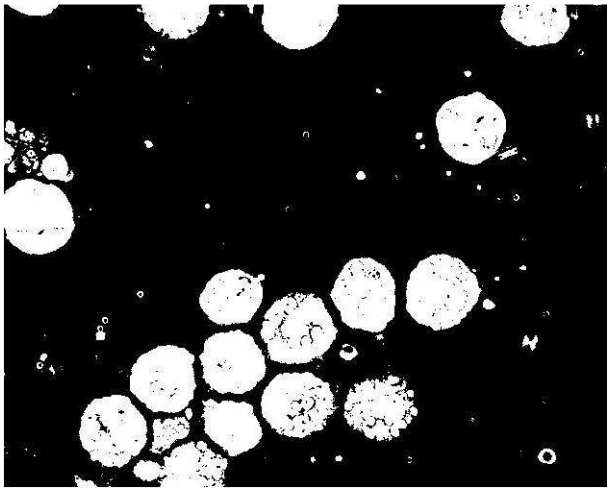

**Figura 4.6: Complemento de la imagen en figura 4.4**

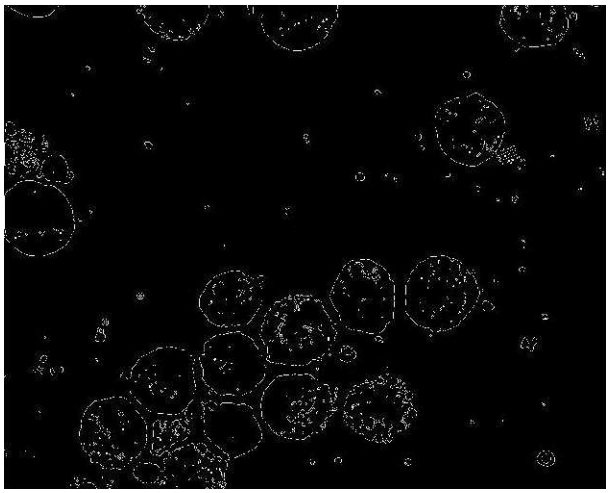

**Figura 4.7: Borde detectados**

Luego, rellenamos y erosionamos los objetos (células) que han sido detectados en la imagen, empleando para ello, las funciones *imfill* e i*merode*. Estas funciones reciben como argumento con que elementos vamos a rellenar y erosionar los objetos que han sido detectados. Estos elementos

son creados con la función *strel*. El resultado del relleno y la erosión, se muestra en la figura 4.8.

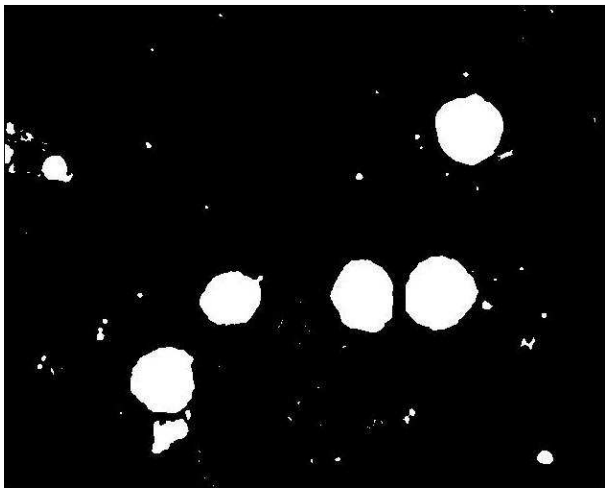

**Figura 4.8: Imagen segmentada**

En este punto, tenemos una buena aproximación de la forma de los objetos (células) presentes en la imagen. Sin embargo, están presentes pequeñas zonas que no representan a los objetos de nuestro interés, que de no ser eliminadas, pueden afectar el resultado de la contabilización.

Este problema puede solucionarse en gran medida aplicando una especie de filtro, que consiste en eliminar los objetos con un área menor al área promedio de los objetos presentes en la imagen. Esta parte del algoritmo se muestra en la figura 4.10. Primero debemos encontrar cuantos componentes (objetos) existen en nuestra imagen utilizando la función *bwlabel*, la cual realiza el etiquetado de componentes conectados en la imagen binaria que debe recibir como parámetro, y que en nuestro caso será, la de la figura 4.8. Luego, con la ayuda de la función *bwareaopen*, la cual recibe como

parámetro el número de pixeles que debe contener un objeto en la imagen para ser eliminado, que en nuestro caso será un porcentaje del área promedio calculada (por ejemplo, 75% del área promedio calculada). El resultado de este procedimiento se muestra en la figura 4.9.

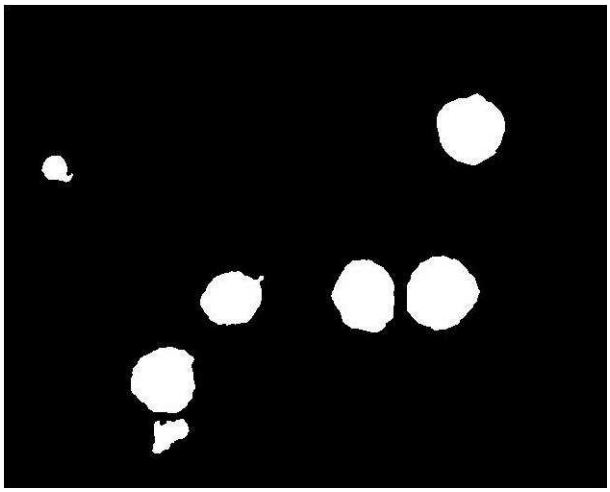

**Figura 4.9: Imagen segmentada mejorada**

Finalmente, el marcado de las células detectadas y el conteo de las mismas, es realizado obteniendo el centroide de los componentes (objetos) detectados en la imagen, por lo que nuevamente utilizamos la función de etiquetado *bwlabel* y luego nos valemos de la función *regionprops* utilizando la opción *centroid*, para obtener como resultado una matriz con la ubicación del centroide de cada objeto detectado en la imagen. El resultado del conteo es el número de elementos que contiene la matriz de centroides. El algoritmo de conteo se muestra en la figura 4.10 y el resultado final en la figura 4.11.

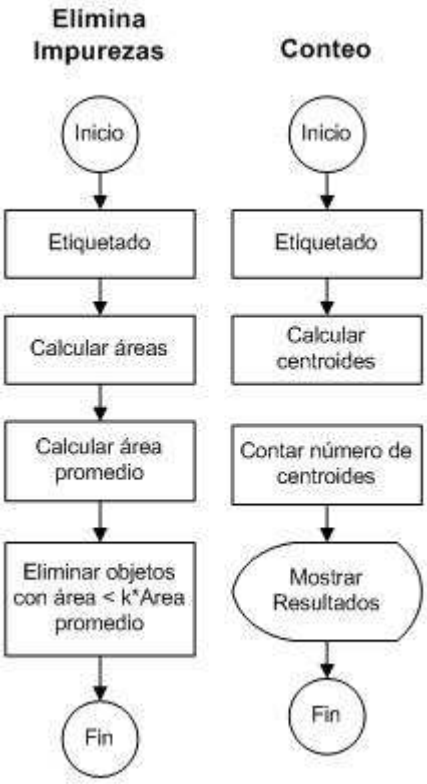


**Figura 4.10: Eliminación de impurezas y Conteo de células**

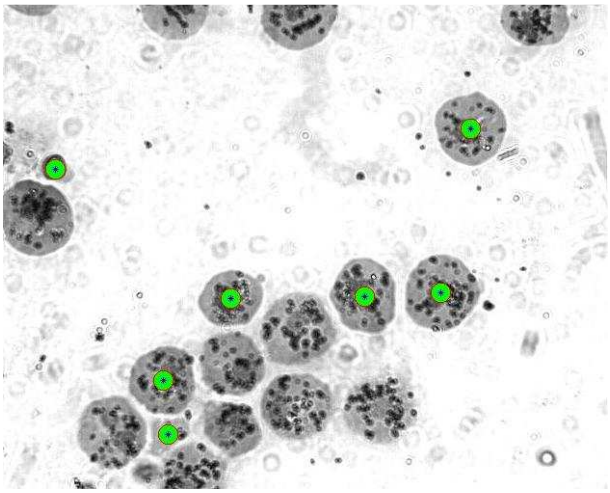

**Figura 4.11: Células contabilizadas con método 1**

### 4.3 Explicación del Método 2

El algoritmo del Método 2 se muestra en la figura 4.12.

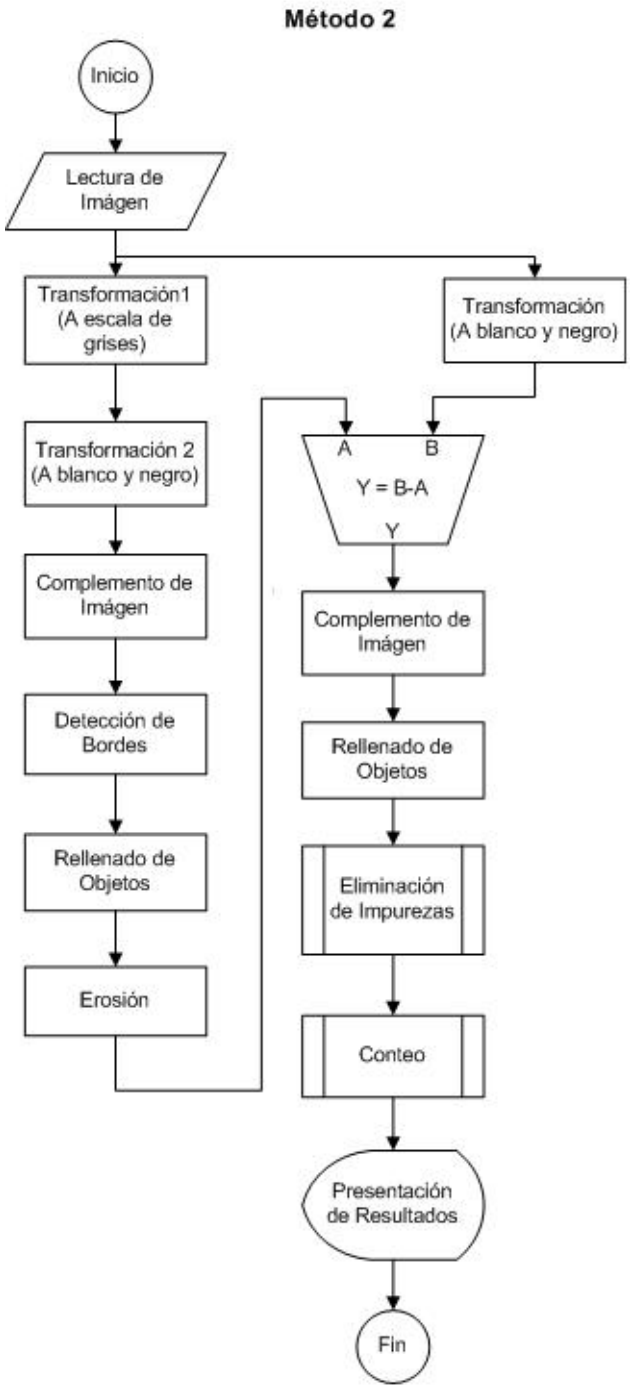


**Figura 4.12: Algoritmo del método 2**

La diferencia entre los dos métodos radica en que el Método 2 luego de realizar la erosión, resta el resultado de este proceso que se muestra en la figura 4.8, de la imagen original llevada a blanco y negro, mostrada en la figura 4.5. Esta estrategia provoca un mejor desempeño al momento de eliminar objetos indeseados presentes en la imagen. En la figura 4.13 se muestra el resultado de este procedimiento.

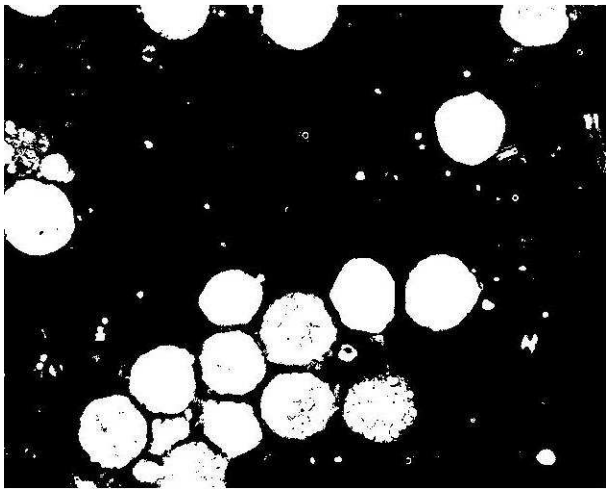

**Figura 4.13: Imágen que se obtiene de la resta de las imágenes en las figuras 4.8 y 4.5**

Nuevamente aplicamos el mismo algoritmo de la figura 4.10, esto, para eliminar objetos indeseados que se hayan podido pasar por alto.

Finalmente, se realiza el mismo procedimiento para la contabilización de las células y su marcado, que realizamos en el Método 1. Como podemos observar en la figura 4.15, para esta imagen y con el Método 2, logramos que algunas células que no habían sido tomadas en cuenta ahora si lo sean.

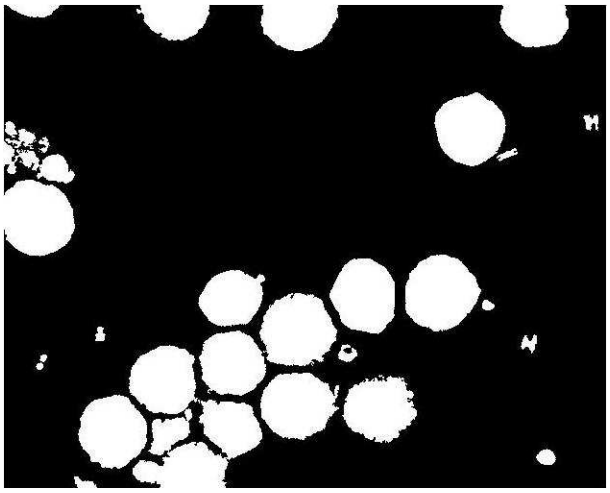

Figura 4.14: Imagen mejorada

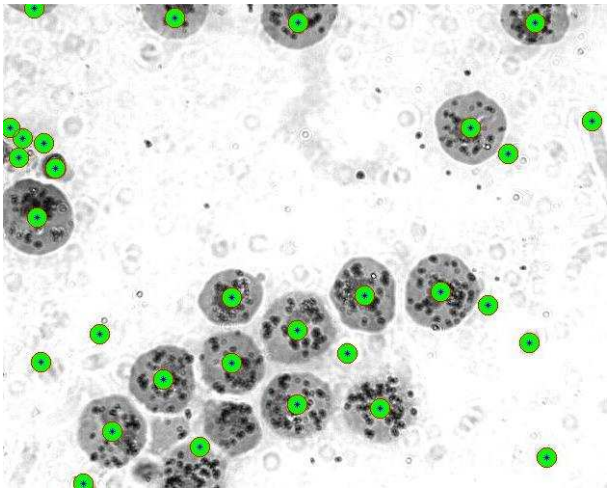

Figura 4.15: Células contabilizadas con el método 2

### 4.4 Funcionalidad de la Aplicación

La aplicación está diseñada para cargar una imagen por medio de una venta que se abrirá al presionar un botón. El nombre de la imagen aparecerá en un área apropiada de la ventana de la aplicación y en otra ubicación se

mostrará la imagen que se cargó y una adicional, se mostrará la imagen con las células que han sido detectadas.

Así mismo, en un recuadro aparecerá el número de células que han sido contabilizadas, dando la opción al usuario de guardar la imagen con las células marcadas. El código de la aplicación se muestra en el ANEXO B.

### 4.5 Resultados obtenidos

Fueron analizadas 24 imágenes, de las cuales, en algunos casos la aplicación funcionó de mejor manera con un método que con el otro. En otros casos, ambos métodos arrojaban resultados muy parecidos, con una muy buena aproximación al número de células presentes en la imagen. La mayoría de las imágenes con las que trabajamos no excedían 1 Mbyte , y con ellas trabajamos sin inconvenientes. Pasado este tamaño, la aplicación puede tardarse en arrojar el resultado. Así mismo, las dimensiones de las imágenes promediaron los 600 x 400 pixeles.

| | | | |
|---|---|---|---|
| Imágenes Analizadas | | | 24 |
| Número de células presentes en las imágenes | | | 2031 |
| Método 1 | Operador Canny | Falsos Positivos | 0 |
| | | Falsos Negativos | 781 |
| | Operador Sobel | Falsos Positivos | 0 |
| | | Falsos Negativos | 974 |
| Método 2 | Operador Canny | Falsos Positivos | 174 |
| | | Falsos Negativos | 778 |
| | Operador Sobel | Falsos Positivos | 169 |
| | | Falsos Negativos | 771 |

**Tabla 1: Resultados obtenidos**

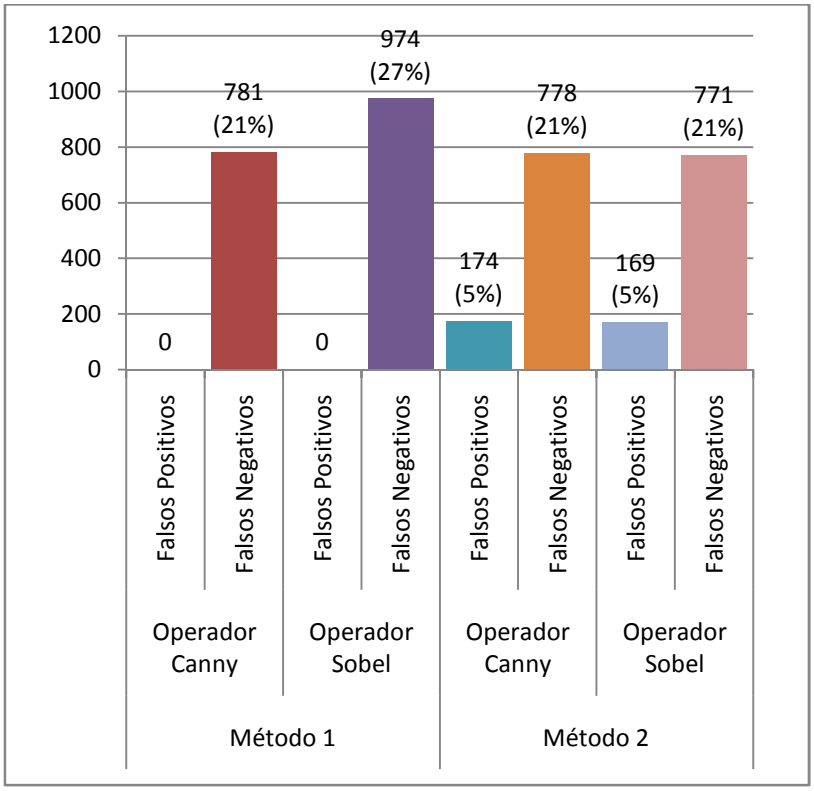


**Figura 4.16: Resultado del análisis**

## CONCLUSIONES

1) De acuerdo a los resultados obtenidos, de entre las combinaciones posibles de métodos y operadores, la menor cantidad de errores se produjeron al utilizar el Método 1 junto con el operador Canny, ya que la suma de la proporción de falsos positivos y falsos negativos llega aproximadamente al 21%. Además Al emplear el Método 1 observamos que no existen falsos positivos (detección de objetos diferentes a los de nuestro interés) lo que le da a nuestra aplicación un cierto grado de selectividad.

2) En cuanto a la superioridad de los operadores, podemos señalar que el operador Canny mantiene una ventaja sobre el operador Sobel. Esta diferencia se nota sobre todo al momento de emplear el Método 1, en el cual la cantidad de falsos negativos (cuando el programa no detecta los objetos que nos interesan) es inferior al utilizar el operador Canny que al utilizar el operador Sobel.

3) Los falsos positivos que se registraron, fueron en algunos casos, elementos adicionales que se encuentran en las muestras juntos con las células, los cuales no pudieron ser removidos en su totalidad en la parte del procesamiento. Mientras que, la gran parte de falsos negativos, se debieron

a que algunas células se encontraban agrupadas en las imágenes, lo que hizo que el programa detectara como una sola célula dichos grupos.

4) En lo que tiene que ver a la velocidad con que se obtiene el resultado del conteo utilizando nuestra aplicación, es evidente que esta, es mucho más veloz comparada con los procedimientos que existen en la actualidad. Sin embargo la exactitud del conteo realizado con nuestra aplicación todavía difiere con la que se obtiene en el conteo manual.

5) Finalmente, este trabajo no pretende reemplazar el procedimiento que se emplea actualmente en este tipo de análisis, solo deja planteada esta alternativa que puede servir como base de partida para futuros trabajos de investigación en la Medicina apoyados en el Procesamiento Digital de Señales.

## RECOMENDACIONES

1) La aplicación muestra un buen desempeño cuando se utilizan imágenes en las que las células se encuentran separadas o en imágenes con grupos de células, en cuyos grupos las células tienen bordes claramente diferenciables (caso de las células de cebolla), por lo que se recomienda que las imágenes a analizar tengan estas características.

2) La elección final del método de contabilización así como del operador depende del resultado que visualicemos al mandar a contar las células de la imagen. Es decir, que si observamos que al utilizar un método junto con un operador determinado, un mayor número de células es marcado, por obvias razones estos deberán ser el método y el operador elegidos.

3) El tamaño de las imágenes incide en la cantidad de recursos, sobre todo de memoria, empleados en el procesamiento por lo que es recomendable que estas no sean demasiado grandes, en nuestro caso obtuvimos buenos resultados con imágenes del orden de los 300Kb y la dimensión de las imágenes, promediaron los 600x400 pixeles.

4) Como trabajo futuro, se podría incluir el cálculo de otros parámetros de las células tales como el tamaño de las mismas. Adicionalmente, esta aplicación podría servir de base para contabilizar otros tipos de objetos tales como granos en general.

# ANEXOS

## ANEXO A

## MANUAL DEL USUARIO

Esta es la ventana principal de la aplicación:

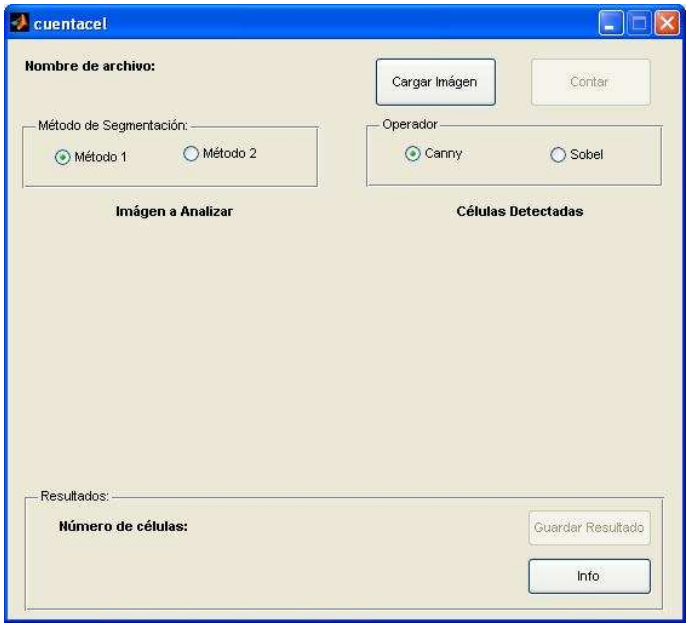


**Figura A- 1: Ventana principal de la Aplicación**

Con este botón se abrirá una ventana desde donde se podrá seleccionar la imagen que queremos analizar:


**Figura A- 2: Botón para cargar la imagen**

Este botón sirve para guardar el resultado del análisis:


**Figura A- 3: Botón para contabilizar**

En este panel se selecciona el operador para la detección de bordes:


Figura A- 4: Panel de operadores

En este panel se selecciona el Método de segmentación:

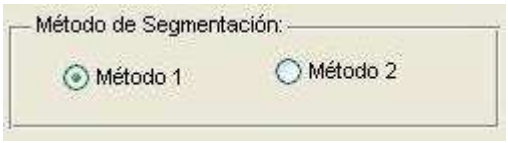


Figura A- 5: Panel de Métodos

Botón para guardar los resultados:

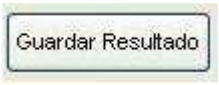


Figura A- 6: Botón para guardar los resultados

Presentación de los resultados:

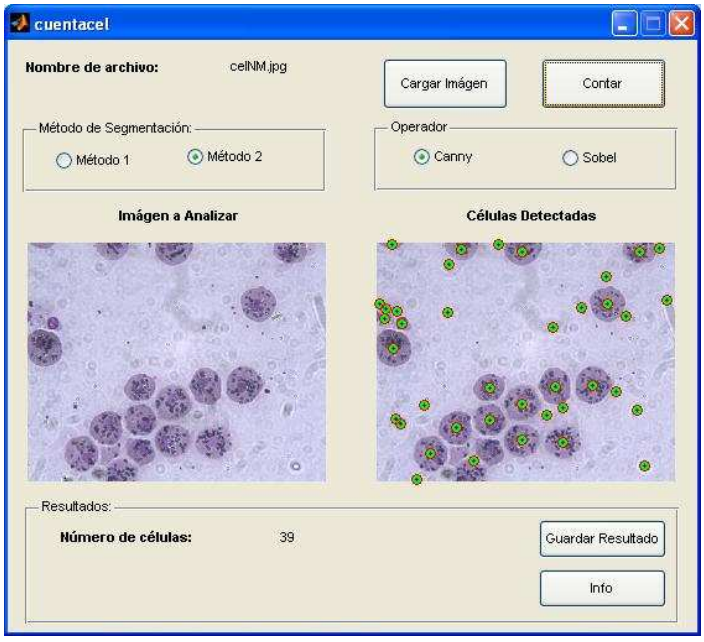


Figura A- 7: Resultado obtenido

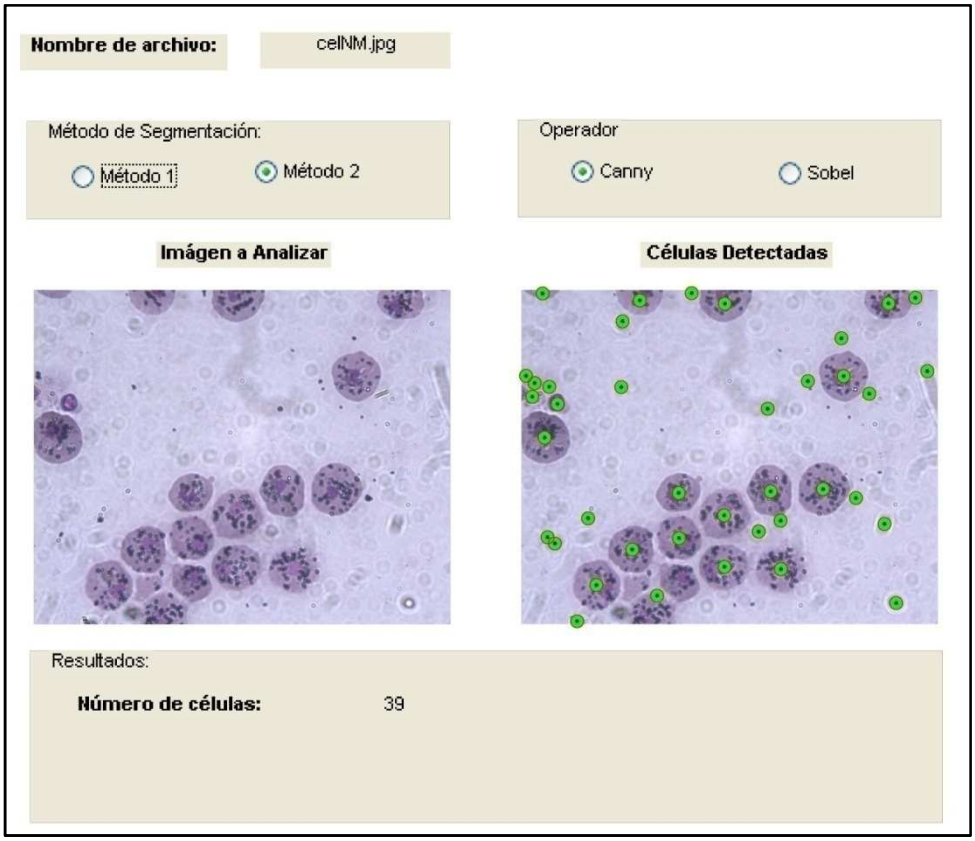


**Figura A- 8: Resultados Guardados**

## ANEXO B

## CODIGO DE LA APLICACIÓN

```
function varargout = cuentacel(varargin)
% Código de inicialización del programa "cuentacel"

function cuentacel_OpeningFcn(hObject, eventdata, handles, varargin)
% Esta función se ejecuta justo antes de que la aplicación sea
% visible.
% En esta función se cargan los valores iniciales de algunas
% variables utilizadas en la aplicación.

function varargout = cuentacel_OutputFcn(hObject, eventdata,
handles)
% Las salidas de esta función son mostradas en la línea de comando.

function pushbutton1_Callback(hObject, eventdata, handles)
% Esta función se ejecuta al presionar el botón Cargar Imagen. Al
% presionarlo se abre la ventana desde donde seleccionamos la imagen
% a analizar.

function fileTypes = supportedImageTypes
% Función auxiliar: aquí se ingresa la lista de formatos de imágenes
% con que trabajaremos.

function pushbutton2_Callback(hObject, eventdata, handles)
% Se ejecuta al presionar el botón Contar. Dentro de esta función se
% encuentra el código que procesa la imagen (Segmenta), Contabiliza
% los objetos (Células) y muestra el resultado, según el método y el
% operador seleccionados.

function pushbutton3_Callback(hObject, eventdata, handles)
% Se ejecuta al presionar el botón Info, para mostrar información
sobre los autores.

function pushbutton4_Callback(hObject, eventdata, handles)
% Al presionar el botón Guardar, se ejecuta esta función con la que
guardamos los resultados.

function fileTypes1 = supportedImageTypes1
% Función auxiliar: lista de formatos de imágenes que se pueden
% utilizar para guardar los resultados.

function uipanel2_SelectionChangeFcn(hObject, eventdata, handles)
% Función para seleccionar uno de los métodos de segmentación.

function uipanel3_SelectionChangeFcn(hObject, eventdata, handles)
% Función para seleccionar uno de los operadores de detección de
bordes.
```

## BIBLIOGRAFIA


- BLANCHET Gerard, CHARBIT Maurice, “Digital Signal and Image Processing using Matlab”, Iste, 2006.

- GONZALEZ Rafael C., WOODS Richards E., EDDINS Stevens L., “Digital Image Processing using Matlab”, Prentice Hall, 2002.

- HUNT Brian, LIPSMAN Ronald, “A guide to Matlab for beginners and experienced users”, Cambridge University Press, 2001.

- JAIN Anil K., “Fundamentals of Digital Image Processing”, Prentice Hall, 1989.

- MARCHAND-MAILLET Stéphane, SHARAIA Yazid, “Binary Digital Image Processing”, Academic Press, 2000.

- PRATT William K., “Digital Image Processing”, Jhon Wiley & Sons, 2001

- SANCHEZ Omar, “Contar objetos en una imagen basado en grupos próximos”, http://omarsanchez.net/cuentaima.aspx, 2009