Segmentación de Iris con OpenCV en Android

Oscar Fuentes Salome

Benemérita Universidad Autónoma de Puebla, Facultad de Ciencias de la Computación, Ciudad Universitaria, Av. San Claudio y 18 sur, Col. San Manuel, C.P. 72570, Puebla, Puebla, México

ofuentessalome@gmail.com

Aldrin Barreto Flores

Benemérita Universidad Autónoma de Puebla, Facultad de Ciencias de la Electrónica, Ciudad Universitaria, Av. San Claudio y 18 sur, Col. San Manuel, C.P. 72570, Puebla, Puebla, México, Teléfono: (222) 2295500

abarreto@ece.buap.mx

Verónica Edith Bautista López

Benemérita Universidad Autónoma de Puebla, Facultad de Ciencias de la Computación, Ciudad Universitaria, Av. San Claudio y 18 sur, Col. San Manuel, C.P. 72570, Puebla, Puebla, México, Teléfono: (222) 2295500

vbautista@cs.buap.mx

Resumen

En este trabajo presentamos la implementación de un algoritmo para la segmentación del iris contenido en imágenes oculares con el propósito de contribuir a la realización de un sistema de diagnóstico automático que pueda ser usado por la Iridología (detección de lesiones en el iris). Para realizar la segmentación, primeramente se localiza la pupila, posteriormente se detecta el iris y finalmente se crea una máscara para dejar únicamente el área correspondiente al iris. Se emplea la transformada circular de Hough para obtener la pupila e iris. El algoritmo fue implementado en una aplicación

móvil (app) para dispositivos con sistema operativo Android y se usaron diversos algoritmos de procesamiento digital de imágenes implementados en OpenCV.

Palabras Claves: Android, OpenCV, procesamiento digital de imágenes, segmentación de Iris, transformada circular de Hough.

1. Introducción

En el campo de la Iridología se considera que existe una estrecha conexión entre el iris y cada uno de los órganos y tejidos del cuerpo, la cual se establece mediante el cerebro y el sistema nervioso. Debido a esta conexión, el estado general de salud de un ser humano, así como la presencia de residuos tóxicos y su acumulación en el cuerpo podrían manifestarse en el iris a través de ciertas lesiones (signos, marcas o coloraciones)(véase [1], p. 6,7).

La Iridología como medicina alternativa proporciona grandes ventajas: el equipo necesario es mínimo, únicamente se requiere de una cámara fotográfica para capturar una imagen del iris, pues si bien es cierto que el diagnóstico se puede realizar directamente observando el iris del paciente, se recomienda el uso de una cámara para obtener imágenes de los ojos a fin de que este proceso sea más fácil y efectivo; es un método de diagnóstico inofensivo, es decir, no produce ningún daño, molestia o efecto secundario, debido a que sólo es necesario enfocar con una fuente luminosa los ojos del paciente y posiblemente capturar fotografías de estos (véase [2], p. 327).

El diagnóstico de padecimientos usando la Iridología se realiza analizando visualmente las características de las lesiones, por ejemplo: forma, textura, coloración, entre otras, y se asocian con un padecimiento en un determinado órgano del cuerpo de acuerdo a su localización en el iris relacionándola a su vez con alguna de las cartas iridológicas (véase [1], p. 8, 9,10, [2], p. 332). Estas cartas iridológicas son mapas de los iris (izquierdo y derecho) que muestran las áreas del iris correspondientes a cada órgano del cuerpo.

Se han creado diversas herramientas de software con el fin de apoyar al iridólogo (persona que tiene conocimientos en Iridología) en su diagnóstico, las cuales permiten principalmente: gestionar los datos del paciente y las imágenes de sus iris, aplicar operaciones a la imagen (zoom, brillo, contraste) y realizar marcas a esta. Por otra parte, también se han realizado trabajos con el propósito de automatizar el diagnóstico de algunas de las lesiones más importantes en Iridología usando algoritmos de Procesamiento Digital de Imágenes (PDI) y Visión Artificial (VA), ejemplos de esto se muestran en [3, 4, 5, 6]. En estos trabajos se identifica algún tipo de lesión, sin embargo, el reconocimiento de patrones (lesiones) en una imagen requiere de etapas previas. Una de estas etapas previas es la segmentación, la cual se refiere a dividir la imagen en las regiones u objetos que la componen y es decisiva en la identificación correcta de los objetos de interés (véase [7], p. 49).

Las técnicas que se han usado para afrontar la segmentación de iris son diversas, en [8] se utiliza difusión anisotrópica para segmentar el iris en imágenes no idealmente capturadas. En [9] se propone una metodología basada en texturas para realizar la segmentación. Con el mismo propósito, en [10] también se emplea la transformada de Hough. En [11] se utilizan contornos activos geodésicos para extraer el iris. Debido a la importancia que adquiere la segmentación de iris en el desarrollo de un sistema de diagnóstico automatizado basado en técnicas de Iridología hemos realizado este trabajo, en el cual presentamos un algoritmo para la segmentación de iris basado principalmente en la transformada de Hough.

Además usamos algunos de los algoritmos de OpenCV (véase [14], p. 158) dado que es una librería de código abierto que contiene una gran cantidad de algoritmos de VA. OpenCV está disponible para usarse en Android, por lo que la implementación del algoritmo propuesto la realizamos en este sistema operativo.

La organización de este documento es la siguiente: en la sección 2 describimos las etapas del algoritmo propuesto, así como las técnicas de PDI empleadas en cada una

de estas. Posteriormente, en la sección 3 describimos los resultados de las pruebas experimentales.

Finalmente, se presentan las conclusiones y el trabajo a futuro.

2. Desarrollo

El algoritmo propuesto en este trabajo se compone de dos etapas. En la primera etapa se identifica la pupila y en la segunda, el iris. Este algoritmo lo hemos desarrollado para su funcionamiento con imágenes capturadas bajo un ambiente no controlado y con características de captura (como la iluminación, ángulo y distancia de captura, cámara, resolución) diferentes, ya que parte del objetivo del trabajo desarrollado fue implementarlo en una aplicación móvil para que sea utilizado por cualquier persona.

2.1. Identificación de la pupila

La primera etapa se inicia aplicando un filtro de suavizado a la imagen ocular. Específicamente se usó la media aritmética y determinamos experimentalmente que era necesario aplicar dos iteraciones. Este filtro reduce las transiciones fuertes, lo cual ayuda para evitar la detección de falsos bordes (véase [12], p. 269). La aplicación de este filtro se realiza mediante convolución con el kernel que se muestra en la Fig. 1. Una vez suavizada la imagen original, se transforma a escala de grises.

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Fig. 1. Matriz de convolución para suavizar la imagen

Posteriormente, se debe calcular el histograma de la imagen. Siguiendo una idea similar a la que se muestra en [13], se analizó el histograma de varias imágenes oculares en escala de grises y se pudo observar que la pupila aparecía como la montaña más cercana al origen, debido a que generalmente, corresponde a los pixeles con tonos más oscuros, un ejemplo de esto se puede apreciar en la Fig. 2. En base al patrón que representa la pupila en el histograma, se dedujo que la identificación del límite interior del iris en la imagen, es decir, la pupila, se podía segmentar mediante umbralización (véase [20], p. 108).

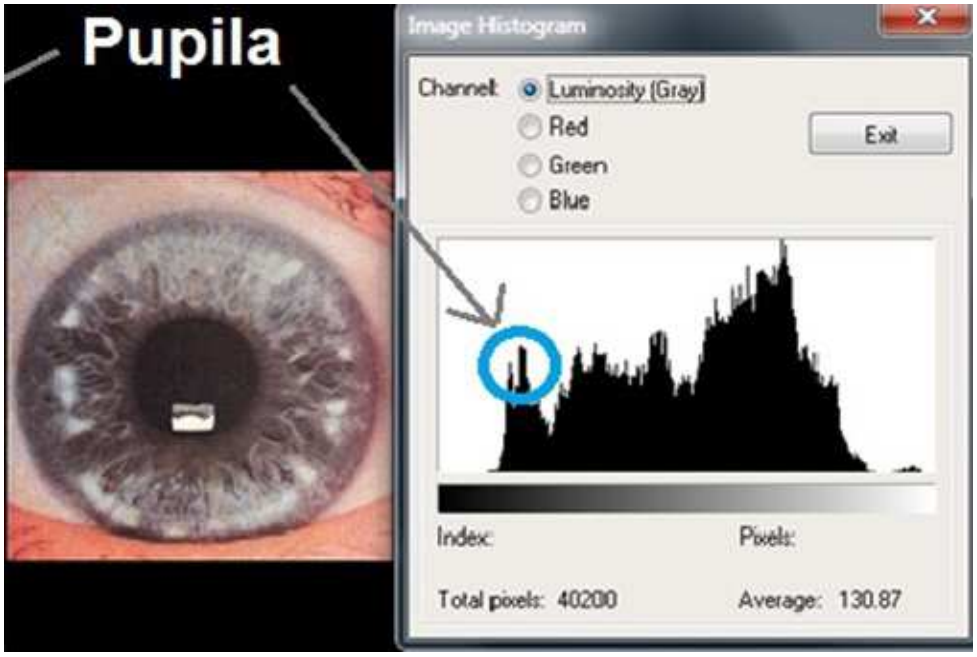

Fig. 2. La primera montaña en el histograma representa la pupila

La umbralización, sin embargo, requiere de un umbral (u), es decir, un valor mínimo para discriminar entre dos clases de pixeles, los que pertenecen y los que no pertenecen a la pupila. Visualmente, mediante la observación del histograma, podía aproximarse el valor del umbral, pero establecer un valor específico para todas las imágenes no era posible debido a que, como se mencionó en párrafos anteriores, las condiciones de captura variaban drásticamente entre una imagen y otra. En general, el valor de u es el valor máximo del histograma de la imagen más cercano al origen, pero

antes de determinar dicho valor es necesario suavizar el histograma, esto a fin de evitar falsos máximos que pudieran aparecer.

Para suavizar el histograma de una imagen se desarrolló y empleó el algoritmo 1, Fig. 3. Este algoritmo requiere que el histograma de la imagen se haya previamente calculado (*histograma []),* al cual se le aplica una mediana con una ventana de 5 elementos (líneas de la 3 a la 9) y a continuación una media entre dos valores seguidos (línea 11, 12 y 13).

De la línea 4 a la 6 se copian del histograma 5 elementos consecutivos a *subarreglo[]*, los cuales se ordenan en la línea 7 y en la línea 8 se obtiene la mediana del subarreglo para asignarla al histograma en la posición *i*. Este proceso se repite hasta la posición 253 del histograma (líneas de la 3 a la 9).

De la línea 11 a la 12 se suaviza el histograma parcial mediante la media entre dos elementos consecutivos y se redondea la división con el propósito de obtener un valor entero para el histograma suavizado. Este segundo proceso se repite hasta la posición 255 del histograma.

En la Fig. 4 se puede observar una imagen, su histograma original y su histograma suavizado. Tras suavizar el histograma se puede observar cómo queda dividido en regiones y los falsos máximos y mínimos desaparecen. Habiendo suavizado el histograma se puede localizar el valor de u.

Pistas Educativas Año XXXVI - ISSN 1405-1249

Certificado de Licitud de Título 6216; Certificado de Licitud de Contenido 4777; Expediente de Reserva 6 98 92

Algoritmo 1 Suaviza el histograma de una imagen

Entrada: Histograma []
Salida: HistogramaSuavizado []
1: *subarreglo [5]*
2: //Se aplica la mediana
3: **para** *i* ← 2 **hasta** *i* < = *histograma.tamaño* – 2 **hacer**
4: **para** *s* ← -2 **hasta** *s* < = 2 **hacer**
5: *subarreglo [s+2]* ← *histograma [i+s]*
6: **fin para**
7: ***Ordenar*** (*subarreglo*)
8: *histograma [i]* ← *subarreglo [2]*
9: **fin para**
10: // Se aplica la media
11: **para** *i* ← *1* **hasta** *i* <= *histograma.tamaño* – 1 **hacer**
12: *histogramaSuavizado [i]* ← ***Redondear*** *((histograma [i - 1]+ histograma [i + 1] / 2))*
13: **fin para**

Fig. 3. Algoritmo para suavizar el histograma de una imagen

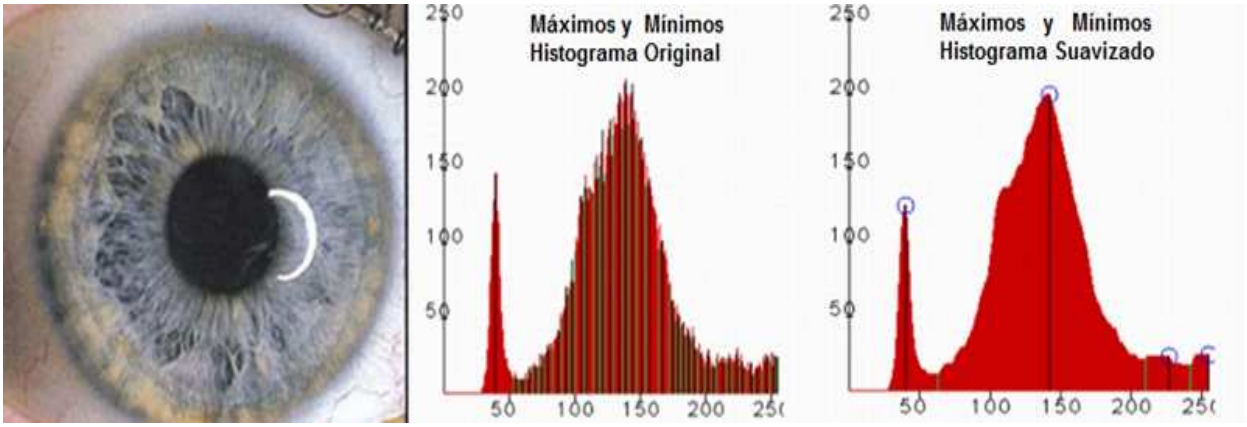

Fig. 4. Histograma original y suavizado

Con el objetivo de determinar u se desarrolló y empleó el algoritmo 2, Fig. 5, el cual asigna a un arreglo (MaxMin, inicializado con ceros) de 256 elementos, los siguientes valores: 1 si el tono correspondiente al índice en el histograma suavizado representa un máximo y -1 si corresponde a un mínimo. Se usan dos banderas DIRARRIBA que toma

el valor de 1 y DIRABAJO que toma el valor de -1 (línea 1 y 2) para que durante el recorrido del histograma suavizado en busca de los máximos y mínimos se pueda saber si se asciende o desciende, y dependiendo de esto, la variable sentido tomara el valor de DIRARRIBA o DIRABAJO respectivamente, de esta manera se sabe cuándo hay un cambio de sentido y se puede determinar si se trata de un máximo o un mínimo (línea 7 y 11).

El valor de umbralización se estableció a partir del primer máximo y mínimo empleando la ecuación 1.

$$u = (max + min) / 2 \quad (1)$$

Algoritmo 2 Obtiene los máximos y mínimos de un histograma

```
Entrada: histogramaSuavizado []
Salida: MaxMin [] // Arreglo con máximos y mínimos
 1: DIRARRIBA ← 1
 2: DIRABAJO ← −1
 3: sentido ← DIRARRIBA
 4: para i ← 2 hasta i <= histogramaSuavizado.tamaño − 1 hacer
 5:     si sentido == DIRARRIBA & histogramaSuavizado [i] > histogramaSuavizado [i+1] entonces
 6:             MaxMin[i] ← 1
 7:             sentido ← DIRABAJO
 8:             i++
 9:     de lo contrario si sentido == DIRABAJO & histogramaSuavizado [i] < histogramaSuavizado [i+1] entonces
10:             MaxMin[i] ← −1
11:             sentido ← DIRARRIBA
12:             i++ fin si
13:     fin si
14: fin para
15: si histogramaSuavizado [histograma.tamaño −1] < histogramaSuavizado [histogramaSuavizado.tamaño− 2] entonces
16:     MaxMin [histogramaSuavizado.tamaño− 1] ← −1
17: de lo contrario
18: MaxMin [histogramaSuavizado.tamaño− 1] ← 1
19: fin si
```

Fig. 5. Algoritmo para obtener los máximos y mínimos de un histograma

Una vez obtenido u se realiza una umbralización con ese valor, es decir se binariza la imagen mediante la ecuación 2.

$$I(x,y) = \begin{cases} 0, & si\ I(x,y) < u \\ 1, & si\ I(x,y) \geq u \end{cases} \quad (2)$$

Después de binarizar la imagen se obtiene segmentada la pupila, sin embargo, en algunos casos también permanecen puntos o regiones innecesarias (artefactos), este problema se puede apreciar en la Fig. 6. Dichos artefactos podrían afectar en la detección correcta de la pupila. Para evitar este problema se utiliza la transformada circular de Hough (véase [14], p. 158), ya que podemos considerar que la pupila es muy semejante a esta forma geométrica.

Fig. 6. Pupila segmentada y artefactos

La Transformada de Hough es una técnica que permite el reconocimiento de patrones globales en el espacio de la imagen, reconociendo patrones locales (idealmente un punto) en un espacio de parámetros transformado, en otras palabras permite reconocer formas paramétricas simples en una imagen usando para ello un espacio diferente, llamado espacio de Hough. Este método se usa principalmente para la detección de rectas, la cual es la forma más básica de implementarla, sin embargo, también se puede utilizar para reconocer polinomios, circulos, etc. Utilizando la transformada de Hough se puede detectar la pupila sin importar que posiblemente existan artefactos en la imagen, ver Fig. 7.

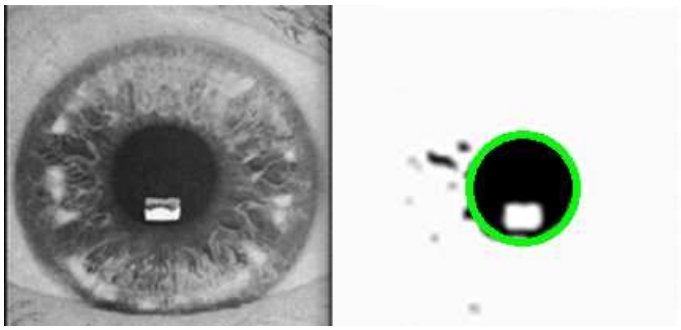

Fig. 7. Pupila detectada con la transformada circular de Hough

Para usar la transformada circular de Hough en nuestro trabajo, no hemos implementado el algoritmo como tal, sino utilizamos el método HoughCircles de la clase Imgproc de OpenCV, el cual tiene la siguiente sintaxis:

Imgproc.HoughCircles(Mat imag, Mat circulos, int metodo, int dp, int minDist, int param1, int param2, int minRadio, int maxRadio)

Dónde:

imag.- es la imagen en la que se desean detectar los círculos.

circulos.- es el vector donde se almacenaran los valores de las coordenadas del centro y el radio de cada círculo encontrado.

metodo.-indica el método que será usado para la detección. Por ahora, OpenCV únicamente utiliza el método 21HT, el cual se indica mediante la constante Imgproc.CV_HOUGH_GRADIENT.

dp.- indica la proporción inversa de la resolución del acumulador respecto a la resolución de la imagen. Si dp = 1 el acumulador tendrá la misma resolución que la imagen, si dp = 2 el acumulador tendrá la mitad de la resolución de la imagen.

minDist.- es la distancia mínima entre círculos detectados. Si el parámetro es demasiado pequeño podrían ser detectados múltiples círculos falsos. Si es demasiado grande, algunos podrían ser omitidos.

param1.- este parámetro es un umbral usado por el detector de bordes Canny, el cual es usado internamente en este método (HoughCircles).

param2.- se refiere al valor del acumulador. Los círculos con valor de acumulador más altos serán retornados primero.

minRadio.- radio mínimo de los círculos.

maxRadio.- radio máximo de los círculos.

La llamada al método Imgproc.HoughCircles se realizó con los siguientes valores:

dp = 1, minDist = 50, param1 = 150, param2 = 13, minRadio = mImag.height()/8, maxRadio = mImag.height()/5.

Los parámetros del método HoughCircles más decisivos para obtener una segmentación satisfactoria son el valor del acumulador, y el tamaño mínimo y máximo de los radios buscados en la imagen. El valor del acumulador (param2) se estableció en 13, lo que significa que un conjunto de puntos equidistantes a un punto en común se debía considerar como círculo si por lo menos tenía 13 elementos, esto debido a que en ocasiones al umbralizar la mayor parte de la periferia de la pupila se elimina o queda demasiado discontinua. El radio mínimo y máximo se determinó mediante la comparación del radio de la pupila con la altura de la imagen que la contenía. Para determinar el valor del radio mínimo se consideró el caso en el que la pupila se encontraba contraída; y para el del radio máximo, el caso en el que la pupila se encontraba dilatada. En la Fig.8, en el lado izquierdo se muestra el modelo de una imagen ocular donde la pupila se encuentra dilatada (de una forma un poco exagerada). Se puede observar que el radio de esta pupila es aproximadamente una quinta parte de la altura de imagen. De manera similar, en el lado derecho de la figura, se muestra un modelo de una imagen ocular donde la pupila se encuentra contraída (de una forma también un poco exagerada) la cual tiene un radio aproximadamente igual a una octava parte de la altura de dicha imagen. Tomando en consideración lo anterior, se estableció

que los valores óptimos para el radio mínimo y máximo eran mImag.height()/8 y mImag.height()/5 respectivamente.

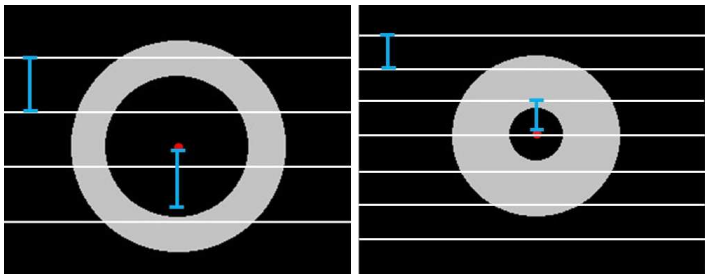

Fig. 8. Modelo de una pupila dilatada (izquierda) y una contraida (derecha)

Una vez que se ejecuta el método HoughCircles, se obtiene en el vector que se pasa como segundo parámetro los valores de las coordenadas del centro y el radio del círculo encontrado, es decir la pupila.

2.2. Identificación del borde exterior del iris

Para la detección del borde exterior del iris, el procedimiento que se utilizó es similar al utilizado en la detección de la pupila. Esta etapa inicia aplicando el filtro de la mediana a la imagen original para reducir el ruido de la imagen ocular. El siguiente paso es calcular su histograma y suavizarlo mediante el algoritmo 1. Después se calculan todos los máximos y mínimos empleando el algoritmo 2. A partir de que se obtuvo el histograma dividido en intervalos, se busca el intervalo más grande, el cual se pudo observar, analizando el histograma de varias imágenes, es el rango de tonos correspondientes al iris. Para averiguar el intervalo se usó el algoritmo 3, ver Fig. 9.

El algoritmo 3 obtiene el punto inicial (*intervalo [0]*) y el punto final (*intervalo[1]*) de la región con mayor área en el histograma. Se definió como región al intervalo que existe entre dos puntos mínimos consecutivos. La búsqueda de la primera región inicia estableciendo como punto inicial (punto1) la posición 0 del histograma (línea 8) y después la posición inicial de la búsqueda es donde se encuentre un mínimo (-1). Mientras la bandera punto2 encontrado se mantenga en falso la búsqueda del punto

final de la región continuará (línea 7 a la 16). Una vez encontrado el intervalo de la región (punto1 y punto2), se calcula el área de la región y se almacena en la variable *suma*. Posteriormente si el número de intervalos (*numInter*) es mayor que uno (línea 23), es decir, ya se ha calculado otra área, se compara el área recién calculada con la anterior (línea 24), si esto también es cierto *sumaAnterior* tomará el valor de *suma* que es mayor al área y se establece el intervalo en el que se encuentra. Si es la primera región que se encuentra entonces suma Anterior toma el valor de suma (línea 30). Este proceso se repite dependiendo del número de mínimos en el histograma suavizado.

Con el valor de u = (p1 + p2) /2 se umbraliza la imagen previamente suavizada con la matriz de convolución mostrada en la Fig.1. Después de haber aplicado la umbralización, se utiliza nuevamente el método HoughCircles de OpenCV para detectar el círculo que representa el iris en la imagen ocular, es decir, con este método se obtienen los valores de las coordenadas del centro y de su radio, Fig.10.

Algoritmo 3 Obtiene el intervalo de los tonos correspondientes al iris

```
Entrada: histogramaSuavizado [], valoresMaxMin []
Salida: intervaloIris []
 1: p2Encontrado ← falso
 2: sumaAnterior ← 0
 3: numInter ← 0
 4: i ← 0
 5: punto2 ← 0
 6: //Se obtiene el intervalo
 7: mientras i < valoresMaxMin.tamaño hacer
 8: punto1 ← i
 9: mientras punto2Encontrado  !=  true hacer
10:        si valoresMaxMin [i] == −1 o i == valoresMaxMin.tamaño − 1
entonces
11:        punto2 ← i
12:         punto2Encontrado ← verdadero
13:              numInter + +
14:         fin si
15:    i + +
16:   fin mientras
17:  //Se calcula el área del histograma en el intervalo
18:  suma ← 0
19:  para  j ← punto1 hasta j < punto2 hacer
20:    suma ← suma + histogramaSuavizado[j]
21:  fin para
22:  // Comparación de áreas, se decide qué área es la más grande
23:  si numInter > 1 entonces
24:       si suma > sumaAnterior entonces
25:        sumaAnterior ← suma
26:        intervalo [0] ← punto1
27:        intervalo [1] ← p2
28:      fin si
29:   de lo contrario
30:    sumaAnterior ← suma
31:   fin si
32:   p2encontrado ← false
33: fin mientras
```

Fig. 9. Algoritmo para obtener el intervalo de los tonos correspondientes al iris

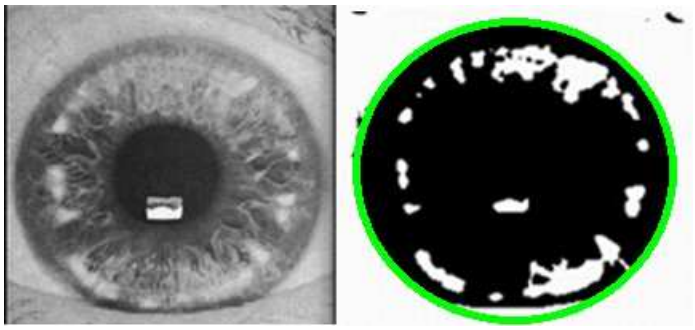

Fig. 10. Iris detectado con la transformada circular de Hough

2.3. Obtención del iris segmentado

Finalmente, habiendo obtenido los centros y radios que representan la pupila y el iris, se crea una máscara con estos datos y los métodos bitwise_not y bitwise_and de la clase Core de OpenCV para obtener el iris totalmente segmentado tal como se muestra en la Fig. 11.

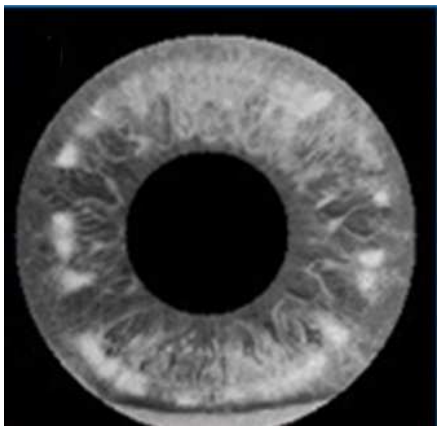

Fig. 11. Iris Segmentado

3. Resultados

El algoritmo propuesto se evaluó tanto en un smartphone como en un emulador, ver Fig. 12, y las características técnicas de ambos se pueden observar en la tabla 1. Para realizar las pruebas se usaron 20 imágenes de la base de datos CASIAIRISV3-Interval

[21]. Además, también se usaron 25 imágenes de dos tipos de lesiones: rosario linfático y anillo de colesterol, estas imágenes se obtuvieron de sitios web dedicados a la Iridología [15],[16],[17],[18],[19]. Los resultados de las pruebas en el emulador y el celular mostraron algunas diferencias con las mismas imágenes. A pesar de las diferencias imprevistas, los resultados alcanzados en la segmentación del iris fueron satisfactorios en la mayoría de los casos, el criterio para considerar una segmentación satisfactoria ha sido si se obtiene al menos 90% del iris y a lo más el 10% de otras partes, en caso contrario se consideró como segmentación no satisfactoria, este criterio se comprobó visualmente. Algunos ejemplos de casos considerados como segmentación satisfactoria se pueden observar en la Fig. 13 y de casos no satisfactorios en la Fig. 14.

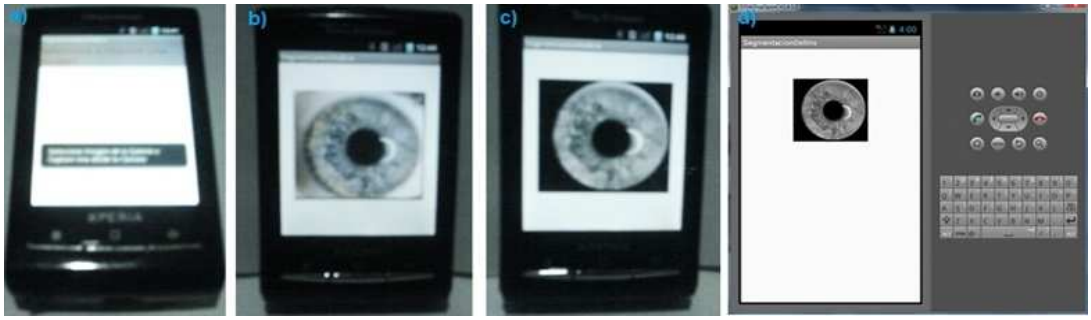

Fig. 12. App para segmentar el iris funcionando en dispositivo móvil y emulador

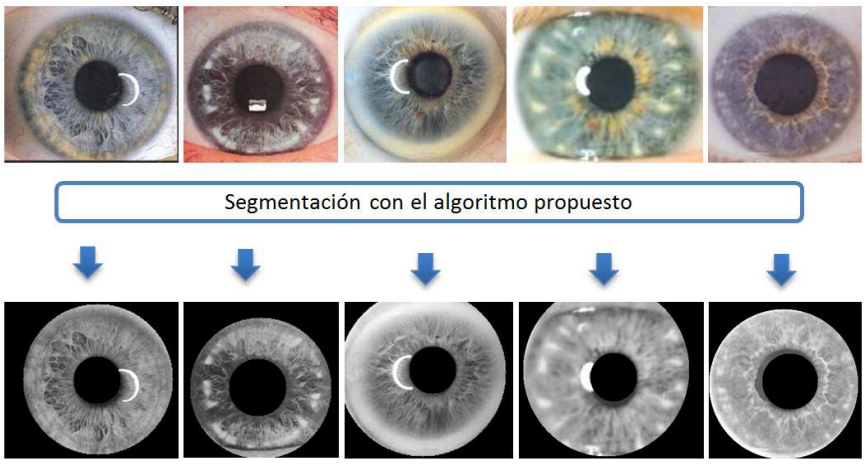

Fig. 13. Ejemplos de segmentación satisfactoria

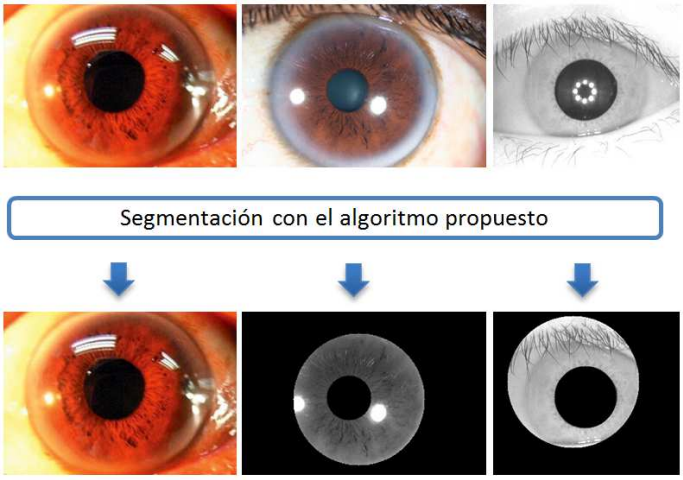

Fig. 14. Ejemplos de segmentación no satisfactoria

	Smartphone	Emulador
Versión de Android	2.1	4.1.2
Procesador	600 MHz Qualcomm MSM7227	ARM (armeabi - v7a)
Resolución	240x320 pixeles (QVGA)	WVGA800
Memoria interna	120 MB	256 MB
Memoria externa	2GB	2GB

Tabla 1.- Características del Smartphone y emulador usados para evaluar el algoritmo.

El porcentaje de aciertos obtenidos en las pruebas que se ejecutaron en el celular, así como, en el emulador, se muestran en la tabla 2.

	Smartphone	Emulador
Detección de la pupila	84.4 %	95.5 %
Detección del borde exterior del iris	80.0 %	88.8 %
Segmentación completa	75.5 %	84.4 %

Tabla 2.- Tasa de aciertos en la segmentación del iris

4. Discusión

Se detectó que las diferencias en los resultados del algoritmo obtenidos en el emulador y el smartphone aparecían desde el momento en que se obtenía el histograma de la imagen, afectándose de manera inmediata las demás etapas del proceso de segmentación. El problema probablemente se debió a la forma de representación interna de la imagen.

Como se puede apreciar, los porcentajes de aciertos más altos se consiguieron en el emulador. Cabe mencionar que los resultados que se esperaban obtener como salida de los algoritmos en cada etapa del proceso de segmentación fueron los obtenidos en este.

Por otra parte, se puede notar que la detección de la pupila tiene una tasa de acierto más alta a la de la detección del borde exterior del iris, esto se debe principalmente a que la pupila generalmente se encuentra bien definida mientras que por el contrario el borde exterior del iris puede confundirse con la esclera, especialmente en lesiones de colesterol ya que esta se presenta como un anillo en tonos muy claros.

5. Conclusiones

En este trabajo se ha presentado un algoritmo eficaz para la segmentación de iris, el cual se ha implementado aprovechando los diversos métodos de procesamiento de imágenes de OpenCV.

Por otra parte, se experimentó la posibilidad de realizar procesamiento de imágenes oculares digitales en dispositivos móviles con sistema operativo Android, demostrando de esta manera, que esto es factible en cuanto a las capacidades técnicas de estos dispositivos, que día a día rápidamente se incrementan.

Finalmente, como trabajo a futuro, se pretende utilizar el algoritmo de segmentación aquí presentado como etapa inicial en el desarrollo del sistema completo de diagnóstico basado en Iridología y que este a su vez se implemente como App, con el objetivo de

que cualquier persona que cuente con un dispositivo con este sistema operativo pueda emplearlo.

6. Referencias

[1] B. Jensen, Iridology Simplified: An Introduction to the Science of Iridology and its Relation to Nutrition. 1980. Iridologist International. California.

[2] D. W. Morrison, How We Heal: Understanding the Mind-Body-Spirit Connection. 2006. North Atlantic Books. California.

[3] L. Yu, K. Wang, D. Zhang, “Extracting the autonomic nerve wreath of iris based on an improved snake approach”. Neurocomputing. Vol. 70. 2007. 743-748 pp.

[4] R. A. Ramlee, S. Ranjit, “Using iris recognition algorithm, detecting cholesterol presence”. International Conference on Information Management and Engineering. Vol. 61. 2009.714-717 pp.

[5] J. Wang, L. Ma, K. Wang, N. Li, “Radii solaris extraction through primitive modelling” ICBM. Vol. 2. 2010. 94-103 pp.

[6] L. Ma, K. Wang, D. Zhang, “A universal texture segmentation and representation scheme based on ant colony optimization for iris image processing”. Computers and Mathematics with Applications. Vol. 57. 2009. 1862-1868 pp.

[7] R. C. Gonzalez, R. E. Woods, Digital Image Processing. 2007. Prentice Hall, New Jersey.

[8] W. Hong-Lin, L. Zhi-Cheng, Q. Jian-Ping, L. Bao-Sheng, “Non-ideal iris segmentation using anisotropic dicusion”. IET Image Process. Vol. 7. 2013. 111-120 pp.

[9] J. Zuo, N. A. Schmid, “On a methodology for robust segmentation of nonideal iris images”. IEEE Transactions on Systems, Man, and Cybernetics - part B: cibernetics. Vol. 40. 2010. 703-718 pp.

[10] S. Chawla, A. Oberoi, “A robust segmentation method for iris recognition”. International Journal of Advanced Research in Computer Science. Vol. 2. 2011. 340-343 pp.

[11] S. Shah, A. Ross, “Iris segmentation using geodesic active contours”. IEEE Transactions on Information Forensics and Security. Vol. 4. 2009.

[12] Digital Image Processing: PIKS Scientific inside. John Wiley & Sons, New Jersey. 2007.

[13] C. Diaz, Y. Torres, O. Torres, “Extracción digital de características biométricas basadas en patrones del iris humano”. Revista Colombiana de Física. Vol. 43. 2011. 244-249 pp.

[14] Learning OpenCV: Computer Vision with the OpenCV. O’Reilly Media, Inc. Sebastopol. 2008.

[15] Iridologist. Disponible en: http://www.betterhealththruresearch.com/Iridologist.htm Consultada en: Julio 2013.

[16] Net irisdiagnosis. Disponible en: http://www.bionordic.dk/page.asp?id=374 &sprog=se. Consultada en: Julio 2013.

[17] Iridología. Disponible en: http://canalelsalvador.wordpress.com/category/ saludnatural/iridiologia/. Consultada en: Julio 2013.

[18] Curso de iridología digital. Disponible en: http://saludbio.com/curso-de-iridolog %C3 %ADa/curso-de-iridolog%C3%Ada. Consultada en: Julio 2013.

[19] Iridología. Disponible en: http://iridologiaprofesorrodriguez.blogspot.mx/. Consultada en: Julio 2013.

[20] L. E. Sucar, G. Gómez, Visión Computacional. Instituto Nacional de Astrofísica, Optica y Electrónica. Puebla, México.

[21] CASIAIRISV3-Interval, Biometric Ideal Test. Disponible en: http://biometrics.idealtest.org/findDownloadDbByMode.do?mode=Iris. Consultada en: Julio 2013.

7. Autores

Oscar Fuentes Salome egresado de la Ingeniería en Ciencias de la Computación en la Benemérita Universidad Autónoma de Puebla. Áreas de interés: procesamiento digital de imágenes, reconocimiento de patrones y aprendizaje automático.

Dr. Aldrin Barreto Flores Estudió en el Instituto Tecnológico de Veracruz obteniendo el grado de Ingeniero en Electrónica en el año de 1998. Realizó estudios de Maestría en Ciencias de la Electrónica en el Instituto Nacional de Astrofísica, Óptica y Electrónica en el año 2000, posteriormente en la misma Institución obtuvo el grado de Doctor en Ciencias en el área de Ciencias de la Computación en el año 2005. Sus áreas de interés son visión por computadora, sistemas embebidos y aplicaciones en la industria.

M.C. Verónica Edith Bautista López estudió en la Facultad de Ciencias de la Computación de la Benémerita Universidad Autónoma de Puebla, obteniendo el grado de Lic. en Ciencias de la computación en 2001, con la especialidad de programación de sistemas. En 2007 obtuvo el grado de Maestra en Ciencias de la Computación, con especialidad en sistemas distribuidos en la misma Institución. Actualmente es candidata a Dra. en Educación de las Ciencias, Ingenierías y Tecnologías por la Universidad de las Américas, Puebla y sus áreas de interés son el desarrollo de aplicaciones móviles y criptográficas, sistemas embebidos, así como la evaluación del aprendizaje e innovación educativa.