Universidad Nacional de Córdoba
Facultad de Matemática, Astronomía y Física

Licenciatura en Ciencias de la Computación
Trabajo final de Grado

"Algoritmos de origen computacional como una alternativa eficaz para la segmentación y comparación digital del iris humano"

Rocchietti Marco A.
Scerbo Alejandro L.A.

Autores:
Alejandro Luis Angel Scerbo scerbo@famaf.unc.edu.ar
Marco Augusto Rocchietti mrocchie@famaf.unc.edu.ar
Supervisor:
Dra Silvia M. Ojeda ojeda@famaf.unc.edu.ar
Clasificación ACM:

'I.4.6' (consultar http://www.acm.org/about/class/ccs98-html)

Marzo-2012

“Algoritmos de origen computacional como una alternativa eficaz para la segmentación y comparación digital del iris humano”

Abstract. *Las técnicas empleadas en la actualidad para reconocimiento e identificación del iris humano son de muy reciente aplicación. La mayoría de los desarrollos disponibles están basados en modelos matemáticos complejos, que aunque demuestran buenos resultados para imágenes digitales en general, no aprovechan propiedades específicas de imágenes del ojo humano. Este trabajo es una respuesta de concepción computacional a esta problemática. Se propone un algoritmo de segmentación de la pupila, y en consecuencia un método computacional de reconocimiento del iris humano.*

1. INTRODUCCIÓN

1.1 El iris en la biometría, conceptos básicos

Desde hace tiempo, se considera al iris del ojo humano como una característica de interés biométrico, por sus cualidades especiales y diferentes de otras biométricas. Además de identificar unívocamente a cada humano (propósito de la biometría), las características que hacen del iris una biométrica se forman desde los tres meses de gestación y permanecen prácticamente idénticas durante toda la vida del individuo. Por otro lado no es necesario el contacto físico para extraer una lectura y su falsificación es prácticamente remota, o por lo menos demasiado dificultosa.
Basándose en diversos estudios recogidos en [2], en el patrón visual del iris hay más información que identifica unívocamente a una persona, que en una huella dactilar.

Su diámetro es prácticamente uniforme en los seres humanos oscilando entre los 11.5mm y 12mm aunque por el efecto de la córnea se observa una longitud horizontal aproximada de 13mm, siendo esta particularidad de gran importancia en este trabajo pues nos provee de una cota (13mm) para el diámetro del iris.

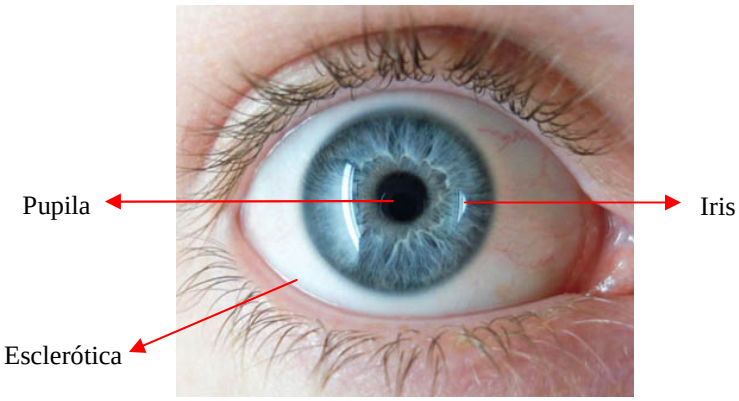


Fig. 1: Secciones del ojo

1.2 Problemática y oportunidad: La concepción de los métodos

Como es de conocimiento en el área, la gran mayoría (si no la totalidad) de las técnicas de moda en reconocimiento e identificación del iris humano, tienen un origen común y relacionado desde sus comienzos, con el análisis estadístico de imágenes digitales [7]. Muchos de los mejores algoritmos de tratamiento, filtrado y compresión de imágenes digitales deben su éxito a la influencia de este enfoque estadístico, el cual revolucionó en un comienzo, la forma en la que se modela al objeto de estudio para luego aplicar resultados teóricos.

Estos "paradigmas de modelo" han establecido diversos objetos estadístico-matemáticos a lo largo de las últimas décadas, tales como por ejemplo:

- Transformada de Fourier
- Componentes principales
- Matrices principales
- "Imagen como base de un espacio vectorial"

En la actualidad la Ciencia de la Computación ha madurado lo suficiente como para establecer modelos útiles para problemas específicos; quedando así expuesta una de las motivaciones de este trabajo: ***"establecer un modelo de concepción computacional"*** para abordar el problema de la segmentación y comparación digital del iris humano.

La "oportunidad" reside en el hecho de que los algoritmos basados en modelos estadístico-matemáticos, muchas veces dependen de filtrados estadísticos, pre-tratamientos, descomposición matemática en componentes, y demás mecanismos que pueden resultar costosos desde un punto de vista computacional [5], [6].

El trabajo, surge como una respuesta de concepción computacional a esta problemática, la idea detrás de la iniciativa se basa en considerar a la imagen como una "estructura de datos" (paradigma propio de las ciencias de la computación) y no como un "objeto matemático".

Específicamente buscaremos un algoritmo de segmentación de la pupila, como una primera aproximación al problema de reconocimiento e identificación de individuos a partir del iris.

Bajo esta concepción fundamental, el desarrollo que propone este trabajo persigue los siguientes objetivos:

1. Producir mejoras en la complejidad (menor costo computacional) con respecto a los algoritmos del estado del arte.

2. Lograr un adecuado balance entre performance y polivalencia.

3. Adquirir independencia del banco de datos que se utilice para mostrar resultados.

1.3 Lineamientos preliminares

"Si podemos efectuar la etapa de segmentación de una manera más eficiente, podremos poner más énfasis en la etapa de comparación, en espera de resultados globales competentes"

Gran parte de los algoritmos o métodos utilizados en el estado del arte se basan en métodos complejos que no aprovechan características comunes en las imágenes digitales del ojo humano, e invierten toda su complejidad en determinar con exactitud los bordes del iris. Por otro lado en la actualidad es de conocimiento común que la información de interés biométrico ("patrón visual") presente en el iris es vasta (o mucho más que suficiente) respecto de otras bio-métricas. Combinando estos hechos, nos vimos motivados a creer que aún si una mejora substancial en la performance de la etapa de segmentación conllevara una pérdida de exactitud en los límites de la misma, esto podría no determinar un deterioro en los resultados, poniendo el énfasis adecuado en las etapas subsiguientes del sistema.

Así, nos vimos impulsados a valernos de hipótesis como las siguientes para comenzar el desarrollo de la 1er etapa (segmentación):

- La pupila se manifestará en la imagen como un discoide de textura relativamente homogénea.
- La pupila es lo suficientemente grande como para conformar la región oscura de mayor superficie en la imagen.
- En todas las imágenes oftalmológicas hay exactamente un ojo.
- Las muestras de un mismo dataset respetan la distancia del sensor al individuo.
- Dado que el ojo humano promedio presenta escasa variación en el diámetro final del iris el límite exterior de la segmentación quedaría automáticamente determinado por la resolución espacial de cada muestra.
- Debido a la cantidad de información biométrica presente en el iris, determinar el límite exterior de la segmentación utilizando características anatómicas no producirá pérdidas substanciales de información.

Nuestro objetivo es entonces tomar estas características comunes en la generalidad de los bancos de datos y tomarlas como precondiciones para desarrollar un método veloz y eficaz para la segmentación de la pupila y con ello comenzar un sistema biométrico eficiente.

En esta ocasión trabajamos con el banco de datos CASIA IrisV1 [1] ya que el mismo es empleado para pruebas de algoritmos en diversos trabajos de actualidad [6] en el área.

Un sistema biométrico tipo [6], se divide en cuatro etapas**: "Localización", "Normalización", "Extracción" y "Comparación".** Nuestro trabajo concreto fue posicionar una alternativa propia para cada una de estas etapas

A continuación explicamos en qué consiste cada una y exponemos nuestro desarrollo sobre las mismas.

2. DESARROLLO

Convenciones de notación

Trabajaremos sobre la base del siguiente marco de referencia:

- Una imagen **I** será una matriz de **dimensión NxM** (N columnas y M filas). Para referirnos al elemento de la matriz I ubicado en "columna i" de la "fila j" escribiremos **I[i, j]** donde $0 \leq i < N$ y $0 \leq j < M$.
- Usaremos el término *celda* para denotar un elemento de la matriz I, considerando tanto sus coordenadas (número de columna y número de fila) como su valor. En este sentido también se usara el termino *píxel*.
- Al valor de una celda de la imagen I se le denominará **valor de intensidad.** Los valores bajos de intensidad estarán relacionados con menor luminosidad en la imagen (el valor de luminosidad más baja es 0, considerado el negro absoluto).
- Una segmentación de **pupila** será un **disco** representado por un par **(c, r)** donde el parámetro 'c' tiene las coordenadas de la celda de la imagen correspondiente al centro de la pupila y el escalar 'r' es el radio de la misma.
- Una segmentación de **iris** será un **anillo** de parámetros **(c, r, R)** donde 'c' es el centro en común de las dos circunferencias que lo delimitan, 'r' el radio del círculo que separa la pupila del iris y 'R' el radio del círculo que delimita al iris con la esclerótica. Este modelo exige que $0 < r < R$.

2.1 Primera etapa: Localización de la Característica

Como ya se mencionó, en un sistema biométrico el primer paso es localizar dentro de la imagen la región en donde se encuentra la información biométrica (el iris en nuestro caso).

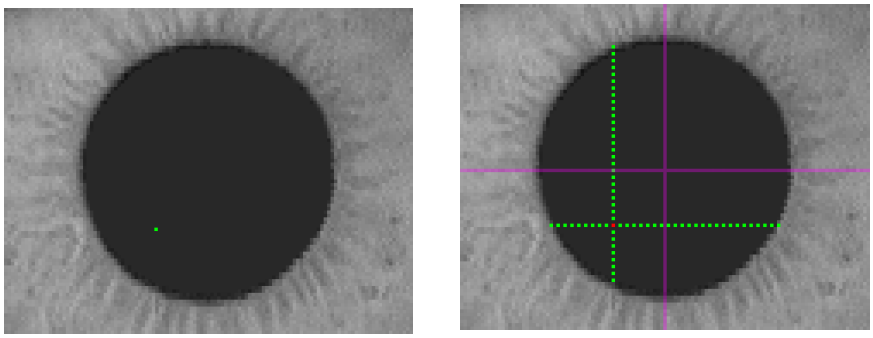
Para localizar el iris necesitamos dos radios: el interior correspondiente al borde pupilar y el exterior correspondiente a la frontera entre el iris y la esclera.

Los sub-pasos que siguen a continuación, desarrollan cada una de estas tareas.

Segmentación Pupilar

Suponiendo que tenemos las coordenadas de una celda correspondiente a la región interior de la pupila y que la misma es un disco perfecto, propondremos un algoritmo simple y eficaz para ayudar a determinar su centro y radio. Dicho algoritmo lo hemos llamado Cruz y trabaja de la siguiente manera:

Se realizan cuatro recorridos (o trazas) a partir del píxel inicial. Dos recorridos en sentido vertical (direcciones "norte" y "sur") y dos en sentido horizontal (direcciones "este" y "oeste"). Todos los recorridos finalizan al detectar que el próximo píxel a visitar presenta una **diferencia de intensidad considerable** respecto a la del píxel inicial. El valor *"de corte"* para esta diferencia se ajusta con un parámetro T (de tolerancia).



(a) **(b)**

Fig. 2. En (a) un píxel perteneciente a la pupila. En (b) las trazas realizadas por el algoritmo Cruz. El píxel rojo corresponde al píxel inicial. Los píxeles verdes corresponden a las trazas. El espacio entre píxeles es solo por motivos estéticos, el algoritmo revisa trazas conexas.

El algoritmo se sustenta en las propiedades simétricas que posee un disco para calcular un centro. En efecto, si uno considera la recta vertical que pasa por el punto que divide la traza horizontal (generada por el método) en dos partes iguales (la recta vertical violeta en la Fig. 2 b), dicha recta también divide al disco a la mitad. Análogamente sucede lo mismo con la recta horizontal que pasa por el punto medio de la traza vertical. La intersección de dichas rectas es el centro del disco. El centro, entonces, se calcula de la siguiente manera:

Sean T_N, T_S, T_E, T_o las longitudes de los recorridos norte, sur, este y oeste respectivamente obtenidos por el algoritmo Cruz a partir de las coordenadas (x, y). Se define:

$$x_c = \frac{(x+T_E)+(x-T_O)}{2} \qquad y_c = \frac{(y+T_S)+(y-T_N)}{2}$$

Las salidas del algoritmo serán, entonces, el centro del disco (x_c, y_c) y las longitudes de los cuatro recorridos T_N, T_S, T_E, T_o.

Una vez obtenidas las salidas del algoritmo ya tenemos la mitad del problema resuelto (pues el centro de la pupila está determinado). Solo nos queda calcular el radio. Para lograrlo se puede realizar una segunda ejecución del algoritmo a partir del centro previamente calculado, ya que de esta manera las trazas realizadas en esta ocasión serán las proyecciones radiales en los cuatro sentidos cardinales a partir del centro, como se puede observar en la Fig. 3. Así cualquiera de las longitudes de traza puede usarse como radio.

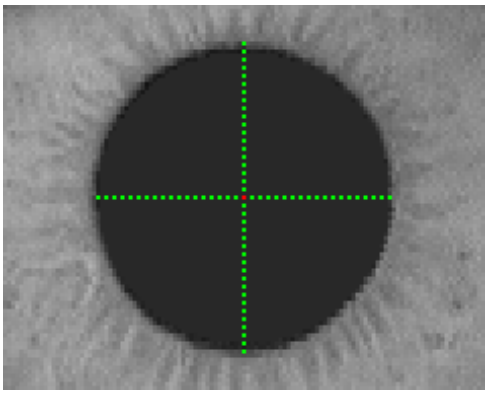


Fig. 3. Representación de la salida del algoritmo Cruz a partir del píxel central de la pupila.

En principio con el método anteriormente explicado se resuelve la segmentación pupilar, pero aun queda un pequeño detalle: La pupila no es un disco perfecto, ya que presenta irregularidades sobre los bordes variando de individuo a individuo, condiciones de iluminación, etc. Por ello hay dos cuestiones a resolver:

1. En la segunda ejecución del algoritmo Cruz las longitudes de traza serán ligeramente distintas, por lo que hay que fijar un criterio para poder, a partir de ellas, elegir la mas conveniente como radio.
2. Cuanto más alejado del centro de la pupila esté el píxel inicial, peor será el centro calculado por la ejecución de Cruz debido a las irregularidades en el borde pupilar de la imagen.

En la primera cuestión nuestra política fue entonces elegir el mínimo de entre las longitudes de traza como radio, priorizando no dejar fuera de la segmentación final algún píxel del iris a no incluir píxeles de la pupila dentro de la misma.

La segunda problemática obliga a realizar una ejecución de Cruz más para poder confiar en el centro calculado, es decir, la primera ejecución nos acercará bastante al verdadero *"centro pupilar"* y luego necesitaremos hacer una nueva ejecución para obtener el centro definitivo. Por último se obtiene el radio como se mencionó anteriormente haciendo una última ejecución del algoritmo.

Encontremos la pupila

Hasta aquí hemos trabajado con la hipótesis de tener un píxel que efectivamente pertenece a la pupila en la imagen. Para encontrar dicho píxel nos basamos en la siguiente idea: Al ser la pupila la región oscura de la imagen con mayor superficie, si tomamos una muestra de píxeles uniformemente espaciados, la mayor parte de píxeles oscuros (de bajos valores de intensidad) serán provenientes de la pupila. Los demás pueden provenir de pestañas o algún ruido en la imagen

La siguiente figura ilustra tal situación.

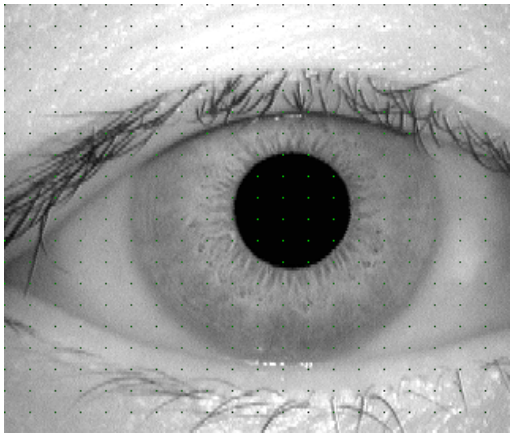


Fig. 4. Muestra de píxeles con distancia uniforme.

Para poder descartar píxeles utilizaremos nuevamente el algoritmo Cruz haciendo el siguiente análisis*:*

"Cuando Cruz se ejecute a partir de un píxel proveniente de las pestañas, las longitudes de traza obtenidas serán 'muy pequeñas' y/o 'muy distintas' entre sí"

El mismo análisis puede hacerse para los píxeles oscuros producto del ruido. El criterio para determinar si una traza es pequeña se fija con un parámetro ρ que especifica el radio más chico esperado para la pupila. Se calcula entonces la media de las longitudes de traza y se verifica que sea mayor a dicho parámetro. Para decidir si las trazas no son lo suficientemente similares se establece un criterio de "*circularidad esperada"*. Para ajustar el criterio se establece un parámetro σ (entre 0 y 100) como porcentaje de semejanza deseada. El método, entonces, trabaja de la siguiente manera:

Definimos

$$T_< = \min\{T_N, T_S, T_E, T_O\} \; ; \; T_> = \max\{T_N, T_S, T_E, T_O\} \; ; \; \overline{T} = \frac{T_N + T_S + T_E + T_O}{4}$$

1. Se ejecuta Cruz sobre el píxel candidato y se evalúa $\overline{T} \geq \rho$. Si no se cumple la condición se descarta el candidato.
2. Se procede a hacer una segunda ejecución a partir del centro calculado para obtener una mejor aproximación al centro pupilar y le llamamos (x_0, y_0).
3. Se realiza una última ejecución de Cruz para analizar las trazas obtenidas a partir de (x_0, y_0).
4. Se evalúa $T_< \geq T_> \sigma/100$. Si se cumple la condición el método determina la segmentación como ((x_0, y_0) , $T_<$). Caso contrario, se rechaza el candidato.

Por último, para hacer la búsqueda más eficiente, ordenamos los candidatos de menor a mayor según su nivel de intensidad (puesto que la pupila debe ser oscura) y además sólo se prueba con los píxeles candidatos provenientes de regiones relativamente centrales de la imagen. Para definir dicha región **A** como área de interés, usamos el triple del parámetro del radio más chico (3ρ) asumiendo arbitrariamente un lado de distancia que aproxime *"una pupila y media"*.

$$A = \{(x, y) \in I \mid 3\rho \leq x \leq N - 3\rho \wedge 3\rho \leq y \leq M - 3\rho\}$$

Segmentación del iris

Una vez determinados los parámetros de la pupila, para segmentar el iris solo debemos encontrar el radio mayor del anillo que define al mismo.

La córnea cubre perfectamente al iris y su diámetro está acotado por 13mm en los humanos. Por lo tanto usaremos el parámetro del dataset resolución espacial o "escala" como referencia para calcular el radio estándar del iris (6,5mm) en píxeles. Hecho esto tenemos todos los parámetros necesarios para nuestro modelo de segmentación del iris y por lo tanto concluye la etapa de localización de la característica biométrica.

2.2 Segunda etapa: Normalización

El propósito de la etapa de normalización es el de proveer alguna forma de estandarización entre las muestras capturadas y/o a contrastar.

En nuestro caso, para implementar esta etapa del sistema nos basamos en el método de desdoblamiento (o "*unwrapping*") propuesto por Daugman [4], ya que posee buenas propiedades como:

1. El tamaño de las muestras normalizadas es mucho menor al de las muestras originales de cada dataset.
2. Con las normalizaciones ya se necesitan parámetros como "centro" y "radio".
3. La normalización de Daugman[4] es robusta frente a dilatación/contracción del iris (i.e. dadas dos muestras del mismo individuo, una contraída y la otra dilatada, sus formas normalizadas tienden a ser similares).
4. La manipulación de estas imágenes normalizadas es mucho más rápida que si se utilizaran las imágenes originales del dataset.

Básicamente se usa la representación polar de los píxeles correspondientes al iris para construir una nueva imagen como se muestra en la Fig. 5.

Daugman[4] propone representar la versión normalizada de un anillo de iris, como una matriz **N** compuesta por una extracción parametrizada de información dentro del iris. Es decir; una selección de sub-circunferencias concéntricas pertenecientes al anillo que lo compone. La imagen normalizada dependerá entonces de dos parámetros:

- **Resolución radial 'r':** La cantidad de sub-circunferencias que tomaremos del anillo del iris, los cuales conformarán las filas de N.
- **Resolución angular 'θ':** La cantidad de radios seleccionados, los cuales conformarán las columnas de N.

Es importante fijar estas resoluciones pues de variar según la imagen el objetivo de normalización no se alcanza.

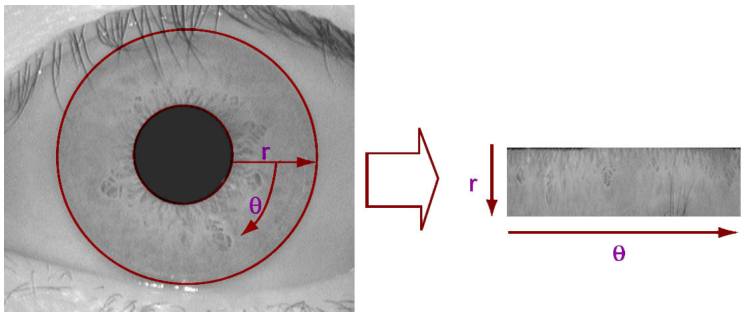


Fig. 5*:* Representación gráfica del "unwrapping" propuesto por Daugman [4].

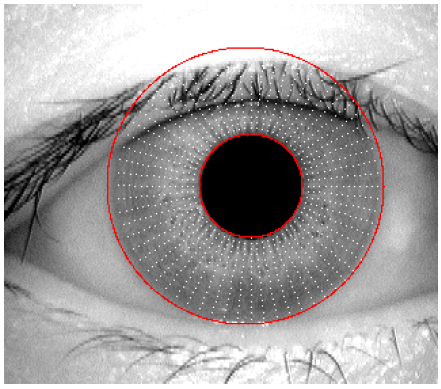


Fig. 6. Ejemplo de los píxeles seleccionados en una normalización con 65 píxeles de resolución angular y 15 píxeles de resolución radial.

La matriz normalizada del iris N de dimensiones θxr se caracteriza entonces de la siguiente manera:

$$N[x_N, y_N] = I[x_i, y_i]$$

donde $x_N = 0...\theta - 1, y_N = 0...r - 1$ y además:

$$\begin{cases} x_i = \cos(\alpha) r_i + x_0 \\ y_i = sen(\alpha) r_i + y_0 \end{cases}$$

Donde (x_0, y_0) son las coordenadas del centro pupilar y:

$$\alpha = \frac{2\pi}{\theta} x_N \qquad r_i = \frac{r_> - r_<}{r} y_N + r_<$$

Siendo $r_<$ el radio de la pupila y $r_>$ el radio de la circunferencia que separa al iris de la esclera. Es decir siendo $((x_0, y_0), r_<, r_>)$ la segmentación obtenida en la etapa anterior.

Nuestra propuesta

En la normalización propuesta por Daugman, se toma una muestra de píxeles arbitrarios y poco representativos del iris. Por otro lado, tampoco es viable trabajar con la totalidad de ellos puesto que manipular objetos tan grandes tiene un costo computacional alto. Nuestra idea es usar información más representativa de la textura. Proponemos extraer estadísticos de distintas regiones del iris en la imagen y formar una nueva a partir de ellos. En este trabajo las regiones serán bloques rectangulares y se utilizará a la media muestral como estadístico ya que mostró mejores resultados que la mediana y la desviación estándar. El mecanismo para generar la nueva imagen se muestra en la siguiente figura.

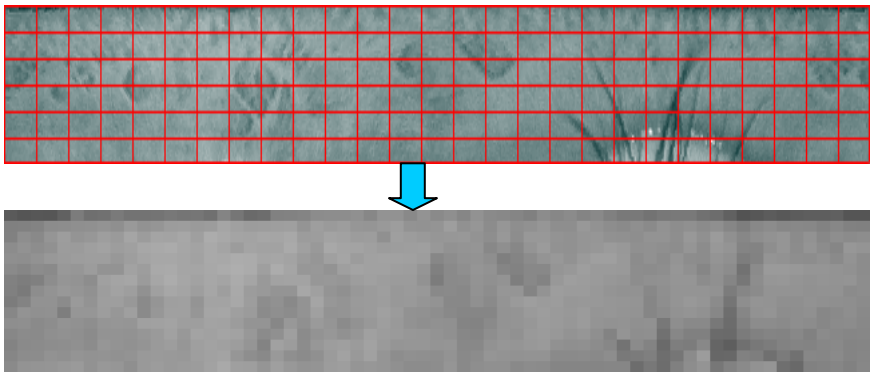


Fig. 7. Arriba la normalización que contiene todos los píxeles el iris de la muestra original, superpuesta la grilla que determinaría la resolución de los códigos a almacenar. Abajo el código ya generado a partir de las medias de las celdas de arriba.

Nuestra innovación aquí consiste en construir una imagen normalizada del iris que contenga **todos** los píxeles de la muestra original y luego se descompone en una grilla según la resolución convenida para almacenar los códigos de iris (θxr).

Finalmente se calcula la intensidad media de cada celda y se utiliza este resultado como valor de intensidad de cada píxel del código generado (en el orden topológico).

Sea N la matriz correspondiente al iris normalizado (con todos sus píxeles) de dimensión $\theta_N x\, r_N$ definiremos la matriz del código de iris C de dimensión θxr:

$$C[x, y] = \overline{B}_{xy} \text{ con } x = 0...\theta - 1,\ y = 0...r - 1$$

$$B_{ij} = \{N[x, y] \mid i\ell_H \le x < (i+1)\ell_H \min\theta_N \wedge j\ell_V \le y < (j+1)\ell_V \min r_N\}$$

Donde ℓ_H, ℓ_V son las longitudes horizontal y vertical de los bloques respectivamente y se definen como:

$$\ell_H = \left\lceil \frac{\theta_N}{\theta} \right\rceil ;\ \ell_V = \left\lceil \frac{r_N}{r} \right\rceil$$

2.3 Tercera etapa: Extracción de característica

En esta etapa se busca capturar dentro de la muestra normalizada, la característica elegida para identificar al individuo y almacenarla conformando así el llamado "código biométrico".

En nuestro caso particular, basados en la observación de que al momento de captura de la imagen ocular puede haber variaciones en el nivel de luminosidad, se nos planteó como objetivo desarrollar un método de extracción de característica que sea robusto ante esta situación. Por ello nuestro método se enfoca en la relación de cambio que hay píxel a píxel en la textura normalizada del iris y no en los valores de intensidad en sí. Más precisamente el vector de característica o código de iris desarrollado se construye de la siguiente manera:

$$D[i, j] = C[i+1, j] - C[i, j]$$

Con i = 0…θ-2 j = 0…r-1 donde C es la matriz θxr correspondiente a la textura del iris codificada con el método propuesto.

A la matriz D, de dimensión $\theta-1$xr, la llamaremos matriz diferencial. La denominación que le hemos dado proviene de la similitud que hay en la forma de calcularla con el concepto de derivada de una función. Más aun, la característica de representar el crecimiento por fila que posee la imagen N es un rasgo importante, pues si calculamos la matriz diferencial de N y la matriz diferencial de la imagen obtenida al sumar a cada componente cierta constante, dichas matrices serán iguales. Esto hace que al momento de la comparación, el sistema sea robusto a la variación de los niveles de luminosidad entre las imágenes adquiridas.

2.4 Etapa final: Comparación (*matching*)

Esta es la última etapa en un sistema biométrico y la razón de ser de todo este trabajo. Aquí la tarea es medir el grado de similitud del código de iris obtenido a partir de la imagen de entrada con los demás códigos almacenados en el sistema. De esta manera se intentará determinar la identidad del individuo. Para medir la similitud entre dos códigos de iris usaremos la llamada distancia de Hamming.

La distancia de hamming entre dos matrices diferenciales $\theta-1 \text{x} r$ se define a continuación:

$$d(D,D') = \sum_{j=0}^{r-1}\sum_{i=0}^{\theta-2} \frac{\left|D[i,j] \oplus D'[i,j]\right|_1}{(\theta-1)rB}$$

Donde la constante B corresponde a la cantidad de bits necesarios para representar cada elemento de la matriz, la operación $\oplus$ corresponde a la operación XOR y $|\,.\,|_1$ cuenta la cantidad de bits no nulos en la representación binaria.

3. RESULTADOS, VALIDACION

3.1 Segmentación

Criterio de efectividad propuesto

En este apartado vamos a establecer un criterio de éxito/fracaso al momento de segmentar pupila, el cual será el empleado para definir estadísticos de efectividad más adelante.

Cabe aclarar que lo que se busca, es formalizar una idea bastante simple, tal como: ***"tendremos una segmentación exitosa cuando la pupila esté visualmente bien segmentada".***

Dicho esto, la definición formal que elegimos más conveniente, fue la siguiente:

"*Una segmentación de pupila (definida como una circunferencia) será exitosa cuando esté contenida dentro de la pupila y al menos dos puntos de la misma; separados por al menos 90° (en ambas direcciones), se correspondan con la frontera entre pupila e iris. Salvando casos excepcionales de pupilas poco circulares donde se requiere el criterio de un supervisor*".

Resultados

Con el criterio de éxito recientemente explicado, obtuvimos una efectividad del 100% en 4.45 segundos (es decir, unos 5 milisegundos por muestra).

Los pasos promedio del método (cada paso representa la corrida de Cruz sobre un candidato diferente para cada imagen) resultaron 1.067.

3.2 Matching

Criterio de efectividad propuesto

Ya que el banco de datos CASIA-Iris V1[1] provee 4 muestras por individuo para la etapa de pruebas *(y 3 distintas para entrenamiento)*, hay; en principio, dos criterios para definir efectividad en la comparación (matching):

- **"Identificación de individuo":** *Tendremos un éxito cada vez que se identifica al individuo correcto con alguna de sus 4 imágenes, y la efectividad será el número de éxitos sobre el total de individuos*
- **"Identificación de muestra":** *Tendremos un éxito cada vez que una muestra se corresponde con el individuo correcto, y la efectividad será el número de éxitos sobre el total de muestras (cuatro veces el total de individuos).*

Resultados

La bibliografía empleada para contrastar nuestros resultados [6] no aporta documentación detallada acerca de cómo se obtuvieron los tiempos en milisegundos de las técnicas del estado del arte en siguiente cuadro comparativo, pero explica que es el tiempo que le tomó a cada técnica (implementada de alguna manera) en algún hardware determinado llevar a cabo los mismos 4 pasos que desarrollamos en nuestro trabajo sobre cada muestra del mismo dataset [1].

Comparación Ilustrativa					
Método	Daugman	LiMa and Tan	Boles and Boashashe	Wavelet Multiscale *1	Método Propuesto
Costo Computacional	285	95	55	81	73
Efectividad identificación	99,90%	99,23%	93,20%	99,60%	95,8% / 100% *2

Peores Valores
Mejores Valores
Nuestros Resultados

(*1): Nombre para representar el método basado en detección de bordes multiescala usando "wavelet maxima" como paso de pre-procesamiento propuesto en [6].

(*2): Como en la bibliografía tampoco se menciona qué criterio de éxito se utiliza para medir la efectividad, incluimos los dos resultados que obtuvimos: 95,8% de efectividad para el criterio "identificación de muestra", y 100% para el criterio "identificación de individuo".

4. CONCLUSIONES

4.1. Segmentación

Encontramos el resultado obtenido altamente positivo, pues analizar, segmentar y recrear las 756 imágenes que componen el dataset le tomó a nuestro método aproximadamente 4.45 segundos en una PC estándar actual de bajo perfil, (CPU Intel(R) Core(TM) i3 M330 @2.13GHz) mientras que con el mismo equipo y sobre el mismo Framework, ejecutar un **"filtro de mayoría de ventana 3"** (una tarea trivial en el ámbito de las imágenes digitales) toma aproximadamente 0.56 segundos sobre una única muestra del mismo dataset.

El valor observado en "pasos promedio" denota que en la gran mayoría de los casos el primer píxel elegido para correr Cruz, ya pertenecía a la pupila.

4.2 Matching

Los resultados obtenidos fueron muy alentadores, ya que; aunque las comparaciones con el estado del arte no sean tan exhaustivas como hubiese sido lo ideal, las pruebas bastaron para poner nuestro desarrollo en tiempos del mismo orden de los que hay publicados en la bibliografía, y la efectividad en cuanto a "identificación de individuo" (la cual creemos es muy razonable si pensamos en el campo de acción forense) arrojó resultados inmejorables. Pudimos identificar a todos (los 108) individuos del dataset por al menos una de sus 3 capturas biométricas disponibles en el subconjunto de entrenamiento, y; en la mayoría de los casos, las 3 capturas apuntaron al mismo individuo (95,8% de efectividad según el criterio "identificación de muestras").

4.3 Globales

En líneas generales, verificamos ideas informales como

"Se podía resolver el problema de la segmentación de una manera mucho más simple y sin modelos matemáticos complejos"

"Los métodos de moda vuelcan la complejidad a la detección exacta de los bordes del iris, apostemos a la comparación"

Se obtuvieron en efecto; resultados competitivos dedicando menos computación a la segmentación (resignando quizás un poco de exactitud en estos bordes), y proponiendo una mejora a la comparación "tradicional" por Hamming.

Logramos una drástica reducción de complejidad segmentando el borde interno con nuestro algoritmo Cruz y valiéndonos de estándares anatómicos (resolución espacial y cota anatómica de 13mm mencionada en sección 1.1) para obtener el borde exterior, perdiendo así exactitud en la segmentación lo cual atenuamos con el empleo de la "matriz diferencial" antes del paso final: el matching.

REFERENCIAS

[1]. Chinese Academy of Sciences, CASIA-IrisV1, http://biometrics.idealtest.org/

[2]. J. G. Daugman. "*High Confidence Visual Recognition of Persons by a Test of Statistical Independence"*. IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 15, nº 11. Noviembre 1993.pp. 1148-1161.

[3]. "Iris Recognition" del NSTC, Comitee on Technology, Comitee on Homeland and National Security, Subcomitee on Biometrics- w*ww.biometrics.gov*, 7 Aug 2006.

[4]. J. Daugman, "How iris recognition works", IEEE Transactions on Circuits and Systems for Video Technology, vol. 14, pp. 21–30, 2004

[5]. R. Wildes, "Iris recognition: an emerging biometric technology", Proceedings of the IEEE,vol. 85, pp. 1348–1363, 1997.

[6]. A. Bouridane, Imaging for Forensics and Security, Signals and CommunicationTechnology, DOI 10.1007/978-0-387-09532-5 1.

[7]. Bernd Jähne. (2004) "Practical Handbook on Image Processing for Scientific and Technical Applications", 2nd Ed. CRC Press LLC.

APENDICE 'A': PSEUDOCODIGOS

A continuación se mostrarán los pseudocódigos más relevantes correspondientes a la implementación de las rutinas y funciones que componen el sistema desarrollado. Primeramente fijaremos un poco más de notación y nomenclatura.

A.1 Convenciones de notación

Trabajaremos sobre la base del siguiente marco de referencia:

- Una imagen **I** se representa como una matriz de **tamaño NxM** (N columnas y M filas) de valores enteros. Para referirnos al elemento de la matriz I ubicado en "columna i" de la "fila j" escribiremos **I[i, j]** donde $0 \leq i < N$ y $0 \leq j < M$.
- Si I es una imagen NxM, para referirnos a la posición determinada por la "columna x" de la "fila y", escribiremos "**celda (x, y) de I**" siempre que se cumpla $0 \leq x < N \wedge 0 \leq y < M$.
- Un **píxel p** será una terna **(x, y, NI)** tal que NI es un entero entre 0 y 255.
- Dada una imagen I de tamaño NxM y p=(x, y, NI) escribiremos $\mathbf{p \in I}$ para denotar que (x, y) sea una celda de I y se cumpla que I[x, y] = NI.
- Dados un par de píxeles **p**=(x, y, NI) y **p'**= (x', y', NI'), $\mathbf{p \leq p'}$ si y solo si se cumple que $NI \leq NI'$.

A.2 Pseudocódigo

Para poder describir los algoritmos que desarrollamos, a continuación vamos a establecer un pseudocódigo el cual no pertenece a ningún lenguaje de programación en particular. La idea es simplemente capturar la esencia de los algoritmos sin perder esfuerzo en cuestiones muy finas de implementación. Así pues cuando, por ejemplo, se necesite usar conjuntos no habrá que diseñar mecanismos sofisticados para implementarlos sino que directamente usaremos la notación de conjuntos habitual, con sus operaciones. El mismo criterio es usado para el resto de los objetos matemáticos.

A continuación describiremos la sintaxis del pseudocódigo junto con una breve e informal descripción de su significado.

Asignación de variable

```
v := E
```

Es la clásica asignación de un lenguaje imperativo, donde "E" es una expresión y "v" la variable donde se guarda el valor resultante de "E".

Otra de las formas de asignación que usaremos en el pseudocódigo es la asignación múltiple:

$v_1, v_2, \ldots, v_n := E_1, E_2, \ldots, E_n$

Aquí, a cada variable "$\mathbf{v_i}$" se le asigna el valor de la expresión "$\mathbf{E_i}$" correspondiente de manera simultánea.

Condicional

if $C_1 \longrightarrow S_1$
[] $C_2 \longrightarrow S_2$
:
[] $C_n \longrightarrow S_n$
fi

Cuando alguna de las condiciones "$\mathbf{C_i}$" se cumple, se ejecuta la sentencia (o grupo de sentencias) "$\mathbf{S_i}$" correspondiente. Si no se cumple ninguna de las condiciones, simplemente no realiza ninguna acción. En el caso que más de una condición se cumpla se elegirá no determinísticamente alguna de las sentencias para ejecutar (en los algoritmos desarrollados todas las condiciones son excluyentes entre si, por lo que esta característica del condicional no será de importancia).

Iteraciones condicionales

while C **do**
S_1
⋮
S_n
od

Este tipo de iteración ejecuta las sentencias S_1,..., S_n mientras la condición C sea verdadera.

Otra de las formas de iteración que usaremos es la siguiente:

do $C_1 \longrightarrow S_1$
[] $C_2 \longrightarrow S_2$
:
[] $C_n \longrightarrow S_n$
od

En este caso, $\mathbf{S_i}$ se ejecutará cuando se cumpla la condición $\mathbf{C_i}$; luego se vuelven a evaluar las condiciones para determinar que sentencia se ejecutará en el próximo paso del ciclo. De esta manera el ciclo de iteraciones finalizara cuando todas las condiciones $\mathbf{C_i}$ sean falsas. Si más de una condición es verdadera, se elije no determinísticamente alguna de las sentencias correspondientes y se ejecuta.

```
for v := a to b do
  S1
  :
  Sn
od
```

En este caso antes de comenzar a iterar se inicializa la variable "v" con el valor (entero) denotado por la expresión "a". Se iterarán las sentencias S_1,..., S_n siempre que $v \leq b$. Al final de cada iteración el valor de "v" se incrementa en una unidad y se evalúa la condición de iteración.

Declaración de funciones

Para definir las funciones en el pseudocódigo usaremos la siguiente sintaxis:

```
fun nombre_funcion(p1, …, pn)
  S1
  :
  Sr
  return E1, E2, …, Ek
end
```

Las variables $\mathbf{p_1}$,..., $\mathbf{p_n}$ tendrán los valores de entrada de la función. Los valores de salida serán los resultados de las expresiones $\mathbf{E_1}$,..., $\mathbf{E_k}$. Cuando k > 1 la forma de recuperar las salidas de la función será:

```
v1, v2, …, vk := nombre_funcion(p1, …, pn)
```

La sentencia de terminación por excepción

Cuando una función no logre cumplir con su propósito, y este sea crucial para el funcionamiento del sistema, usaremos la sentencia "**abort**" que simplemente termina el programa en un estado de error.

El tipo de dato "lista"

Usaremos en algunas funciones el tipo *lista* para almacenar datos. Este tipo de datos es muy simple y solo nos hace falta definir dos operaciones y un constructor:

- La lista vacía: **[]**
- Agregar por derecha un elemento a la lista: ◄
- Tomar el primer elemento de la lista y sacarlo: **take()**

El funcionamiento del operador "◄" se ilustra en la siguiente ecuación:

```
([]◄ 5) ◄ 2 = [5, 2]
```

En el caso de la función "**take()**" es importante notar que además de devolver el primer elemento de la lista pasada como parámetro, modifica a la misma quitándole dicho elemento. Un ejemplo:

```
lista := [5, 2]
e := take(lista)
```

Luego de ejecutar el programa anterior, los valores de las variables serán:

```
e = 5 ∧ lista = [2]
```

A.3 Implementación del sistema

El siguiente esquema, es una descripción TOP-DOWN de la estructura de funciones de nuestro sistema.

En otras palabras se comienza (arriba) por la función de más "alto nivel" (la que captura la funcionalidad final del sistema) y cada flecha simboliza la llamada al procedimiento, función o funcionalidad señalada.

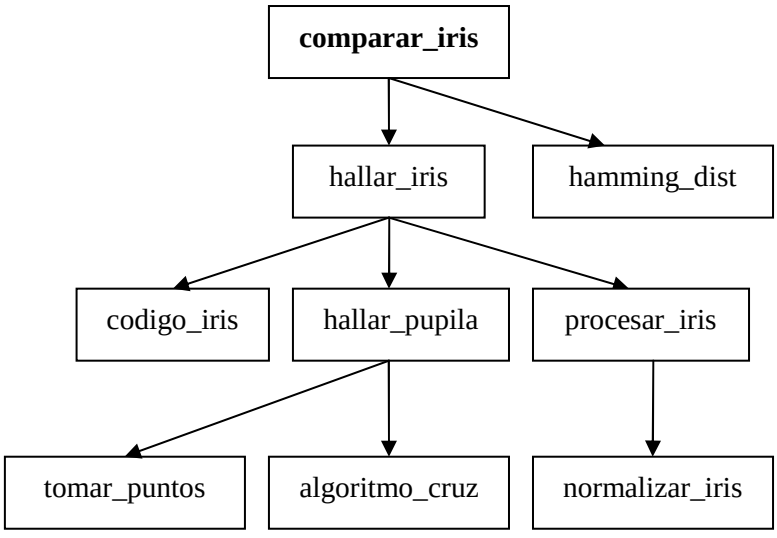


Fig 5.9: Descripción TOP-DOWN del sistema completo.

A continuación se muestran los pseudocódigos de los componentes del sistema.

Algoritmo Cruz (algoritmo_cruz)

Valores de entrada:

- Una imagen **I** de tamaño NxM
- Un píxel inicial **p** = (x , y, NI)
- Un entero **T** correspondiente a la diferencia de intensidad tolerada entre cada píxel de una traza y **p**.

Las salidas son:

- Un par ordenado **C**=(xc, yc) correspondiente a las coordenadas del centro calculado.
- Cuatro enteros positivos **TN**, **TS**, **TE** y **TO** correspondientes a las longitudes de los recorridos en cada punto cardinal a partir de **p**.

Y las variables propias:

- **YN**, **YS** correspondientes a las coordenadas norte y sur de la traza vertical.
- **XE**, **XO** correspondientes a las coordenadas este y oeste de la traza horizontal.

El algoritmo Cruz queda entonces:

```
fun algoritmo_cruz(I, p, T)
  (x, y, NI) := p
  YN, YS := y, y
  XE, XO := x, x
  do |I[x, YN] – NI| ≤ T ∧ YN > 0 ⟶ YN := YN – 1
  [] |I[x, YS] – NI| ≤ T ∧ YS < M – 1 ⟶ YS := YS + 1
  [] |I[XE, y] – NI| ≤ T ∧ XE < N – 1 ⟶ XE := XE + 1
  [] |I[XO, y] – NI| ≤ T ∧ XO > 0 ⟶ XO := XO – 1
  od
  C := (XE + XO, YN + YS) / 2
  TN, TS := y – YN,  YS – y
  TE, TO := XE – x,  x – XO
  return TN, TS, TE, TO, C
end
```

Algoritmo para hallar la pupila (hallar_pupila)

Las entradas:

- Una imagen **I** de tamaño NxM
- Un entero **T** correspondiente a la tolerancia a distancias de intensidad para el algoritmo Cruz.

- Un entero **TC** de valores entre 0 y 100 que indica la tolerancia circular.
- Un entero **RM** que contiene el radio mínimo para la pupila

Las salidas son:
- Un valor **E** booleano (verdadero o falso) que indica si se encontró una presunta pupila.
- Un par ordenado **C**=(xc, yc) correspondiente a las coordenadas del centro de la segmentación pupilar.
- Un entero **R** que contendrá el radio de la segmentación pupilar.

Las variables propias relevantes:
- **TN**, **TS** correspondientes a las longitudes norte y sur de las trazas verticales.
- **TE**, **TO** correspondientes a las longitudes este y oeste de las trazas horizontales,
- "**pixeles**" es una variable que contiene la lista de píxeles obtenidos de la muestra uniforme en la imagen.
- "**exito**" contiene un valor de verdad que indica si el método pudo encontrar a la pupila.

Además se usan las funciones **`tomar_puntos()`** y **`ordenar_puntos()`** que hacen lo esperado. El segundo parámetro de la primera función es un porcentaje que determina la distancia entre los píxeles de la muestra que se devolverá. Por ejemplo, si la imagen tiene dimensión 320x280 y pasamos 5%, la distancia entre un punto y el siguiente en sentido horizontal será de 16 píxeles (pues representa el 5% de 320) y la distancia entre un punto y el siguiente en sentido vertical será de 14 píxeles.

```
fun hallar_pupila(I, T, TC, RM)
  exito := falso
  pixeles := tomar_puntos(I, 5)
  pixeles := ordenar_puntos(pixeles)
  while pixeles ≠ [] ∧ ¬exito do
    p := take(pixeles)
    if p∈{(x,y,NI)∈I|x∈[RM*3,N-RM*3] ∧ y∈[RM*3,M-RM*3]}→
      TN, TS, TE, EO, C := algoritmo_cruz(I, p, T)
      traza_media := media {TN, TS, TE, TO}
      if traza_media ≥ RM ⟶
        TN, TS, TE, TO, C := algoritmo_cruz(I, C, T)
        TN, TS, TE, TO, c_aux := algoritmo_cruz(I, C, T)
        traza_menor := min {TN, TS, TE, TO}
        traza_mayor := max {TN, TS, TE, TO}
        R := traza_menor
        exito := traza_menor ≥ traza_mayor * (TC / 100)
      fi
    fi
  od
  return exito, C, R
end
```

Implementación de la normalización de Daugman (normalizar_iris)

Valores de entrada:

- Una imagen **I** de dimensión NxM.
- Un par de variables enteras "**x_centro**" e "**y_centro**" que son las coordenadas del centro del anillo que representa al iris.
- "**r_menor**" que es el radio de la pupila.
- "**r_mayor**" que es el radio de la circunferencia que divide al iris de la esclera.
- Un par de variables enteras "**A**" y "**R**" que indican la resolución angular y la resolución radial (respectivamente) de la matriz de iris normalizada que se generará.

Valor de salida:

- Una matriz "**matriz_iris**" de dimensión AxR que contiene los pixeles extraídos del iris.

```
fun normalizar_iris(I, x_centro, y_centro, r_menor, r_mayor, A, R)
  for code_y := 0 to R – 1 do
    for code_x := 0 to A – 1 do
    alpha := (2 * π / A) * code_x
    iris_r := ((r_mayor – r_menor) / R) * code_y + r_menor
    iris_x := cos(alpha) * iris_r + x_centro
    iris_y := sen(alpha) * iris_r + y_centro
    if celda (iris_x, iris_y) ∈ I ⟶
      matriz_iris[code_x, code_y] := I[iris_x, iris_y]
    fi
    od
  od
  return matriz_iris
end
```

Algoritmo propuesto para normalizar el iris (procesar_iris)

Valores de entrada:

- Una imagen **I** de dimensión NxM.
- Un par de variables enteras "**x_centro**" y "**y_centro**" que son las coordenadas del centro del anillo que representa al iris.
- "**r_menor**" que es el radio de la pupila.
- "**r_mayor**" que es el radio de la circunferencia que divide al iris de la esclera.

- Un par de variables '**N**' y **M**' correspondientes a la dimensión de la matriz de iris a generar.

Variables propias importantes:
- **NT** y **MT** son las dimensiones necesarias para almacenar en una matriz todos los píxeles del iris.
- La variable "**iris**" contiene la matriz normalizada con la totalidad de píxeles del iris.
- **BX** y **BY** contienen la longitud horizontal y vertical de un bloque respectivamente.

Valor de salida:

- Una matriz "**matriz_iris**" de dimensión N'xM' con el iris normalizado procesado.

```
fun procesar_iris(I,x_centro,y_centro, r_menor, r_mayor, N', M')
  MT := Ri - Rp
  NT := ⌈2 * π * radio_mayor⌉
  iris := normalizar_iris(I,x_centro,y_centro,r_menor,r_mayor,NT,MT)
  BX := ⌈NT / N'⌉
  BY := ⌈MT / M'⌉
  for code_y := 0 to M' - 1 do
    for code_x := 0 to N' - 1 do
      bloque := { iris[i,j] | i∈[code_x*BX,(code_x + 1)*BX min NT)
                            ∧ j∈[code_y*BY,((code_y + 1)*BY min MT) }
      matriz_iris[code_x, code_y] := media(bloque)
    od
  od
  return matriz_iris
end
```

Generación de la matriz diferencial (codigo_iris)

Más precisamente el vector de característica o código de iris desarrollado se construye de la siguiente manera:

La rutina toma:
- Una imagen **I** de tamaño NxM correspondiente al iris normalizado.

Devuelve:
- Una matriz **D** de dimensión (N-1)xM que llamamos matriz diferencial.

```
fun codigo_iris(I)
  for j := 0 to M - 1 do
    for i := 0 to N - 2 do
      D[i, j] := I[i + 1, j] – I[i, j]
    od
  od
  return D
end
```

Método de segmentación del iris (hallar_iris)

Finalmente el método que combina las funciones para poder, a partir de una imagen, obtener el código de iris definitivo se detalla a continuacion:

Valores de entrada:

- Una imagen **I** de dimensión NxM
- Una constante **RC** que representa a 6.5mm en píxeles
- Un par de variables '**N**' y '**M**' correspondientes a la dimensión del código de iris a generar.
- Un par de variables **T** y **TC** correspondientes a la tolerancia de intensidad y a la tolerancia circular respectivamente.
- Una constante **RM** que indica el radio más chico esperado para la pupila en píxeles.

Valor de salida:

- Una matriz **iris_code** de dimensión N'xM' que contiene el código de iris calculado.

```
fun hallar_iris(I, RC, N', M', T, TC, RM)
  exito, centro, r_menor := hallar_pupila(I, T, TC, RM)
  r_mayor := RC
  if exito →
    matriz_iris := procesar_iris(I, centro, r_menor, r_mayor, N', M')
    iris_code := codigo_iris(matriz_iris)
    return iris_code
  [] ¬exito →
    abort
  fi
end
```

Cálculo de Hamming para matrices diferenciales (hamming_dist)

Para calcular esta distancia entre matrices usamos el siguiente algoritmo:

Valores de entrada:

- **D** y **D'**, las matrices de dimensión NxM correspondientes al par de códigos de iris para comparar.

La salida es:

- La variable "**hamming**" que contiene un valor entre 0 y 1 correspondiente a la distancia calculada.

Además usaremos la constante "**B**" que simboliza la cantidad de bits necesarios para representar un elemento de las matrices D y D', y la función **bits_no_nulos()** que cuenta la cantidad de unos que hay en la representación binaria de un número dado.

```
fun hamming_dist(D, D')
  hamming := 0
  for j := 0 to M - 1 do
    for i := 0 to N - 1 do
      hamming := hamming+bits_no_nulos(D[i,j] XOR D'[i,j])
    od
  od
  hamming := hamming / (B * N * M)
  return hamming
end
```