Localización del iris en imágenes en escala de gris usando los valores de intensidad

Juan Antonio Cano Salado, Borja Moreno Fernández y Pascual Javier Ruiz Benítez

Resumen

Este trabajo aborda el problema de la localización del iris en imágenes en escala de grises.

El artículo que hemos usado como referencia es: Localization of iris in gray scale images using intensity gradient (A. Basit, M.Y. Javed).

El proceso se divide en cuatro fases: localización de la pupila, detección del límite iris-esclerótica, localización de párpados y eliminación de pestañas.

En la primera fase comenzamos aplicando un filtro de media a la imagen de entrada para suavizarla. A continuación la binarizamos para facilitar la búsqueda de un punto cualquiera dentro de la pupila.

Una vez localizado ese punto procedemos a calcular el centro y el radio de la pupila.

En la segunda fase se localiza el límite exterior del iris. Para ello aplicamos un filtro de Gauss a la imagen y calculamos los puntos con gradiente significativo dentro de un banda en torno al límite iris-esclerótica.

En la tercera fase procedemos a la localización de los párpados, que pueden aparecer sobre la región del iris calculada en la fase anterior.

Para ello buscamos puntos con gradiente significativo dentro de la region del iris.

En la última fase eliminamos las pestañas que queden dentro de la región del iris (que ya se encuentra completamente definida), si es que las hay.

Key words: iris, localización, filtro Gauss, filtro media, gradiente, histograma

1. Introducción

La identificación automática de las personas viene siendo muy importante en los últimos años. Debido a su gran fiabilidad se vienen desarrollando muchos sistemas de seguridad biométricos. La biométrica es la rama de la ciencia en la que una persona es automáticamente identificada por su comportamiento (firma, paso, voz, etc) o su físico (huella dactilar, iris, retina, geometría de la mano, etc).

El iris empieza a formarse en el tercer mes de gestación. Los patrones que forman el iris se completan en el octavo mes, aunque la pigmentación puede cambiar en el primer año de vida. El patrón complejo del iris contiene muchas características: ligamentos arqueados, surcos, crestas, anillos, coronas y pecas. El iris humano tiene características únicas y es lo suficiente complejo para ser usado como firma biométrica y no cambia en la vida de una persona.

No sólo los iris de gemelos idénticos son diferentes, incluso el izquierdo y el derecho de una misma persona son diferentes. La localización del iris en una imagen sirve como preprocesamiento en un sistema de identificación por

Email addresses: juacansal@alum.us.es (Juan Antonio Cano Salado), bormorfer@alum.us.es (Borja Moreno Fernández), pasruiben@alum.us.es (Pascual Javier Ruiz Benítez).

iris. Para cualquier sistema de reconocimiento, la precisión del sistema de reconocimiento del iris es muy dependiente de la precisión de la localización del iris.

Cuanto mejor localizado esté el iris, mejor será el rendimiento del sistema de reconocimiento. En el siguiente documento se describe la implementación de un algoritmo para la localización del iris en imágenes en escala de grises usando los valores de intensidad. La aplicación recibe como entrada una imagen en escala de grises de un ojo, tras la aplicación del algoritmo se delimitará el iris en la imagen.

2. Planteamiento teórico

El iris se modela como 2 círculos concéntricos y los párpados se modelan como 2 curvas parabólicas. El procedimiento de localización del iris consta de 4 fases.

2.1. *Localización de la pupila*

2.1.1. *Filtro de la media*

Como la pupila es una porción lisa y oscura en la imagen de un ojo, aplicamos un filtro de media con el objetivo de suavizar ligeramente la imagen.

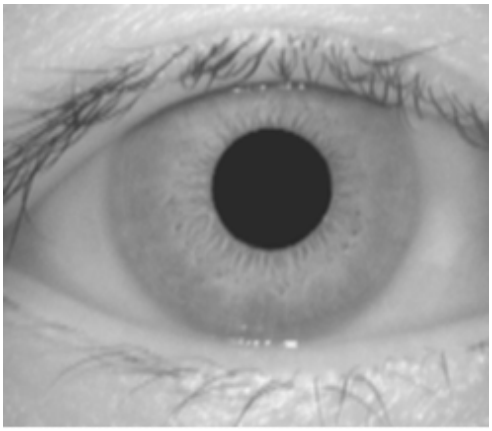

Figura 1. Filtro de media

2.1.2. *Encontrar punto dentro de la pupila*

Se binariza la imagen y se localiza un punto cualquiera del interior de la pupila. La componente X de dicho punto se corresponde con la columna que más píxeles negros tiene en la imagen binarizada. La componente Y de dicho punto se corresponde con la fila que más píxeles negros tiene en la imagen binarizada.

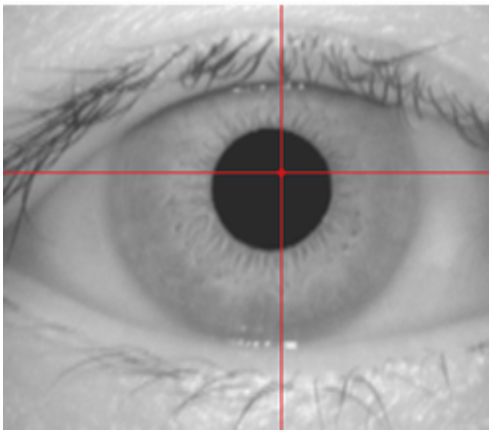

Figura 2. Punto cualquiera dentro de la pupila

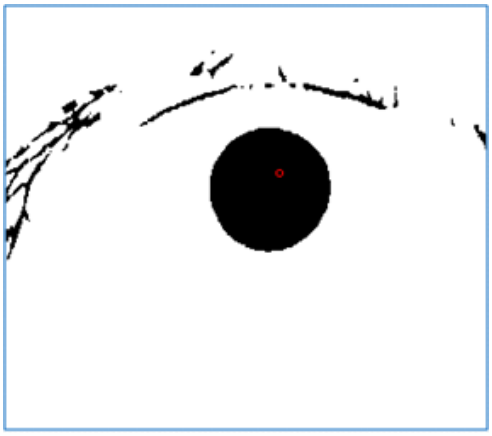

Figura 3. Imagen binarizada

2.1.3. *Encontrar centro de la pupila (método iterativo)*

Se localiza el centro de la pupila utilizando un método iterativo. Se parte del punto obtenido en el apartado anterior. En cada iteración, se actualiza la componente X del punto de manera que tenga el mismo número de píxeles negros consecutivos a izquierda y derecha. Se hace lo mismo con la componente Y. El método converge en el punto centro de la pupila siempre que ésta sea redondeada.

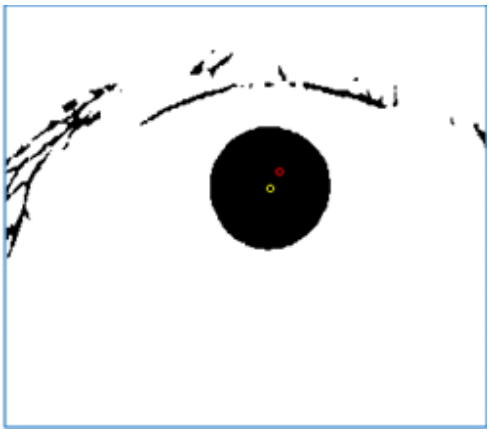

Figura 4. Centro de la pupila

2.1.4. *Encontrar radio de la pupila*

Se determina el radio de la pupila sumando el número de píxeles negros consecutivos que hay encima, debajo, a izquierda y a derecha del centro de la pupila y dividiendo el resultado obtenido entre 4.

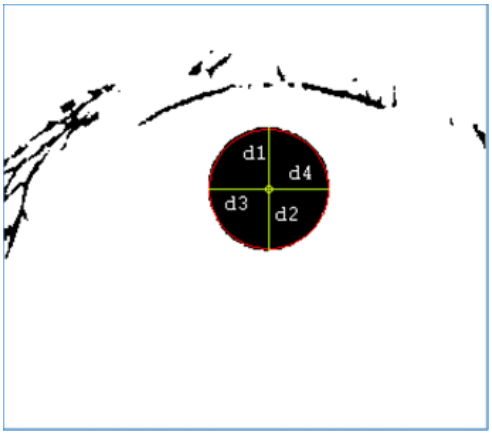

Figura 5. Radio de la pupila

2.2. *Detección del límite iris-esclerótica*

2.2.1. *Cálculo de círculos interno y externo*

Se determinan dos círculos que delimitarán la región de la imagen en la que puede encontrarse el círculo del iris. Ambos están centrados en el centro de la pupila. El círculo interno es ligeramente superior al círculo de la pupila y el círculo externo es el más grande posible tal que esté contenido en los límites de la imagen.

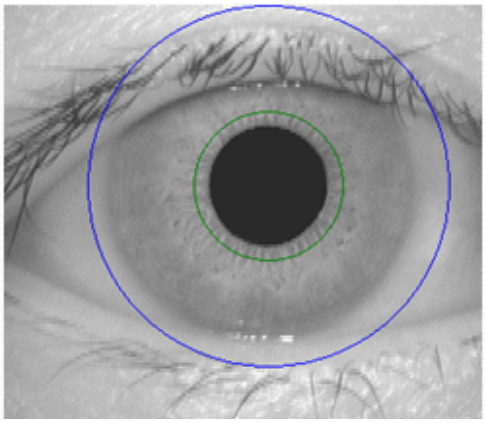

Figura 6. Círculos interno y externo

2.2.2. *Filtro Gaussiano*

Procedemos a realizar un filtro gaussiano para suavizar la imagen, eliminando los detalles.

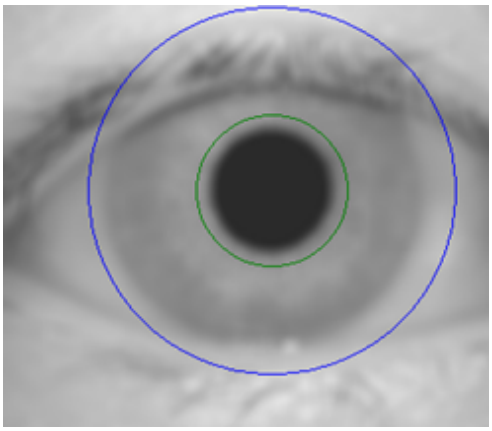

Figura 7. Filtro gaussiano

2.2.3. *Cálculo de puntos con mayor gradiente*

Se localizan los puntos de mayor gradiente de intensidad en la región delimitada por los círculos interno y externo calculados anteriormente. El cálculo del gradiente de intensidad se hace en dirección radial desde el centro de la pupila. Se buscan estos puntos solamente en los cuadrantes inferiores, alrededor de la línea horizontal que atraviesa el centro de la pupila, ya que es la región donde es menos probable encontrar interferencias de los párpados.

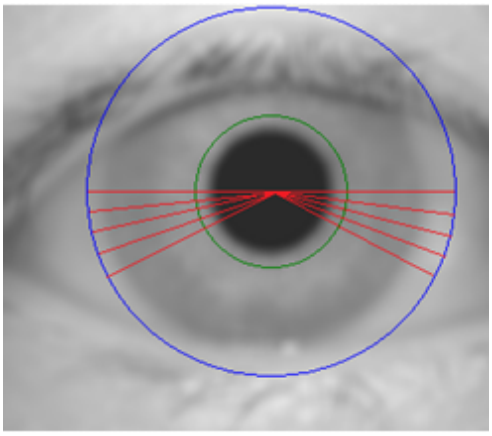

Figura 8. Barrido en cuadrantes inferiores

Y nos quedamos con los puntos de mayor gradiente:

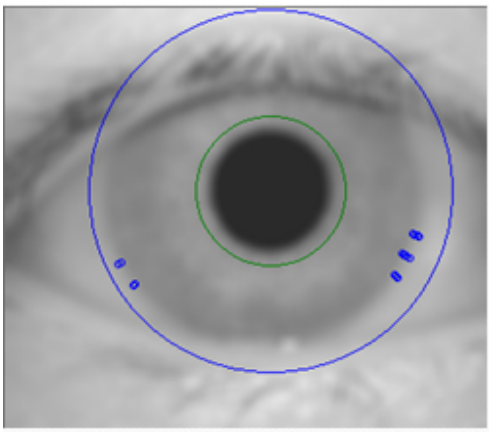

Figura 9. Puntos con mayor gradiente

2.2.4. *Determinar el radio del círculo que separa el iris de la esclerótica*

Se calcula la banda de anchura 16 que agrupa mayor cantidad de puntos de gradiente significativo. El radio del iris será la media de las distancias de los puntos comprendidos en la banda al centro de la pupila.

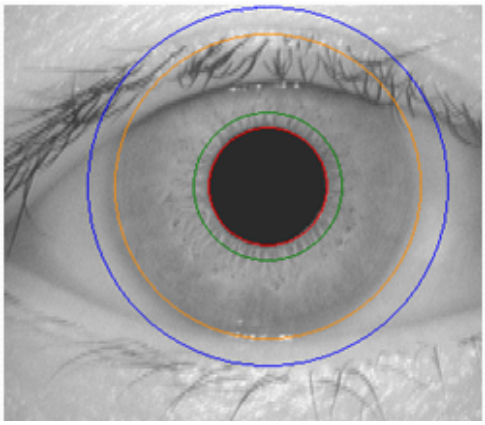

Figura 10. Radio del iris

2.3. *Localización de párpados*

2.3.1. *Determinar una parábola virtual superior e inferior*

La parábola virtual superior pasa por el punto (CentroPupila.X - RadioIris, CentroPupila.Y) así como por el punto (CentroPupila.X + RadioIris, CentroPupila.Y). Tiene su vértice en el punto (CentroPupila.X, CentroPupila.Y - RadioPupila - 10).

La parábola virtual inferior pasa por el punto (CentroPupila.X - RadioIris, CentroPupila.Y) así como por el punto (CentroPupila.X + RadioIris, CentroPupila.Y). Tiene su vértice en el punto (CentroPupila.X, CentroPupila.Y + RadioPupila + 10).

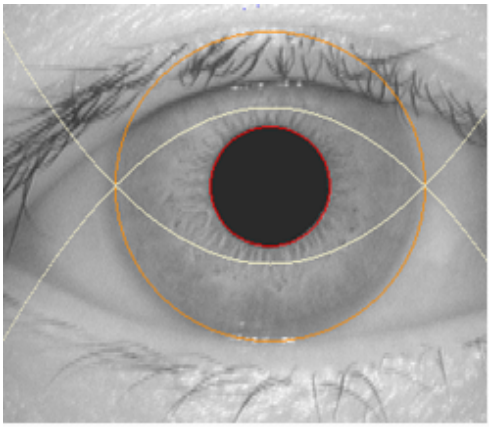

Figura 11. Parábolas virtuales

Buscaremos el párpado superior en la región comprendida entre la parábola virtual superior y el semicírculo superior del iris, y el párpado inferior en la región comprendida entre la parábola virtual inferior y el semicírculo inferior del iris.

2.3.2. *Cálculo de puntos con mayor gradiente en los párpados*

Se localizan los puntos de mayor gradiente de intensidad en la región delimitada entre la parábola virtual superior (inferior) y el semicírculo superior (inferior) del iris. El cálculo del gradiente de intensidad se lleva a cabo en dirección vertical.

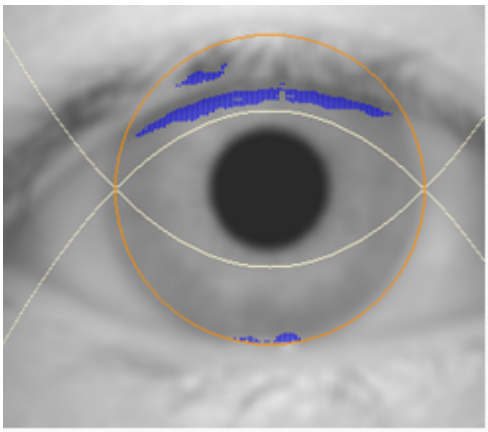

Figura 12. Mayor gradiente en zona de párpados

2.3.3. *Aplicar filtro a puntos candidatos*

De los puntos seleccionados anteriormente nos quedaremos sólo con aquellos que cumplan las tres condiciones siguientes:

1. Tener un valor de intensidad inferior a 120.
2. Tener un valor de intensidad muy similar al de alguno de sus vecinos situados a su izquierda.
3. Tener un valor de intensidad muy similar al de alguno de sus vecinos situados a su derecha.

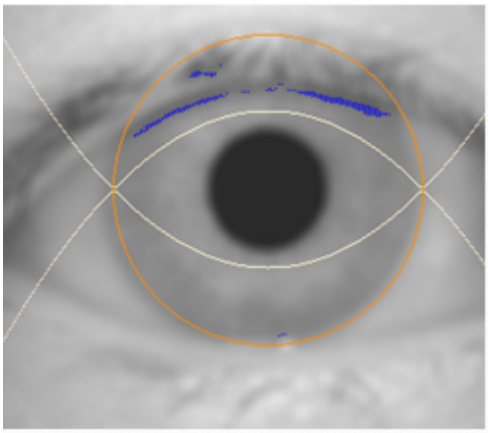

Figura 13. Filtrado de puntos en zona de párpados

2.3.4. *Determinar parábolas de mínimos cuadrados*

Si una cantidad considerable de puntos situados en la zona superior (inferior) del iris pasaron el filtro anterior, se calcula la parábola de mínimos cuadrados para estos puntos. El resultado será la aproximación al párpado superior (inferior).

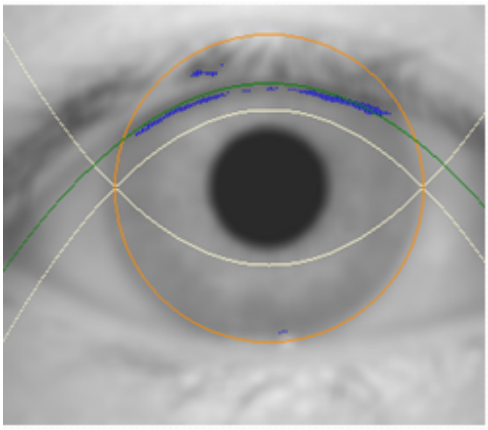

Figura 14. Parábolas de mínimos cuadrados

Llegados a este punto ya conocemos la localización de la pupila, el iris y los párpados. Eliminamos el resto de la imagen pues contiene información que no nos interesa.

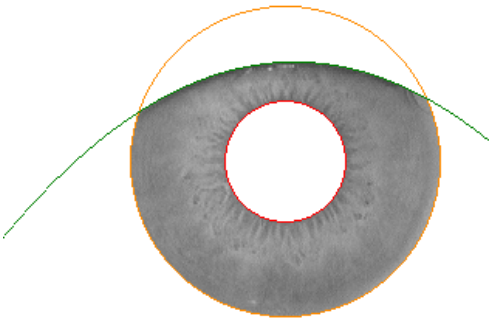

Figura 15. Iris con posibles pestañas

2.4. *Eliminación de pestañas*

Se localizan aquellos píxeles con una intensidad inferior a 100, que serán candidatos a pertenecer a una pestaña, ya que las pestañas son más oscuras que el iris.

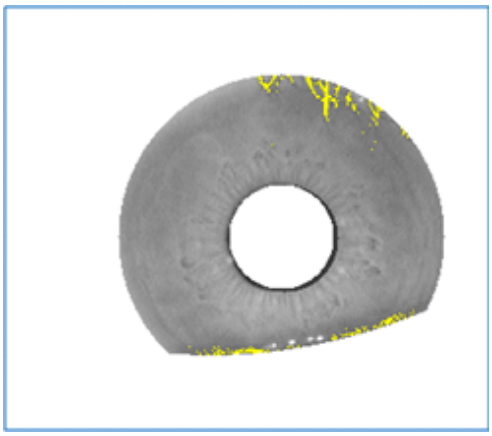

Figura 16. Puntos con valor de intensidad inferior a 100

Si se detectó un número de píxeles oscuros superior a 10 se considera que la imagen original tenía pestañas sobre el iris y se procede a la eliminación de estos píxeles oscuros. Con esto concluye el proceso, el iris ha quedado completamente delimitado y libre de pestañas.

Figura 17. Eliminación de posibles pestañas

Si no se detectó un número suficiente de píxeles, se considera que no hay pestañas en la imagen original.

3. Resolución práctica

En esta sección comentaremos los pasos que hemos implementado para que así quede más claro su posterior estudio si se decide hacer una ampliación de este trabajo.

Hemos modelado las fases como clases que heredan de la clase padre Fase, y los pasos como clases que heredan de la clase padre Paso.

3.1. *Localización de la pupila*

3.1.1. *Filtro de la media*

Se utiliza la clase PasoFiltroMedia, y se realiza la media utilizando el método Apply de la clase Mean() de la biblioteca aForge.NET:

```
Mean filter = new Mean();
Bitmap ImagenFiltroMedia = filter.Apply(Estado.Instance.ImagenInicial);
```

3.1.2. *Encontrar punto dentro de la pupila*

Se utiliza la clase PasoPuntoPupila. Se binariza la imagen mediante el método Apply de la clase Threshold de la biblioteca aForge.NET:

```
Threshold filter = new Threshold(CorteBinarizacion);
Bitmap imagenBinarizada = filter.Apply
 (Grayscale.CommonAlgorithms.RMY.Apply
 (Estado.Instance.ImagenFiltroMedia));
```

Una vez binarizada la imagen calculamos un punto cualquiera en el interior de la pupila. En concreto calcularemos aquel punto cuya componente X se corresponda con la columna que más píxeles negros tiene,

```
for (int fila = 0; fila < filas; fila++)
{
  sumaActual = 0;
  for (int columna = 0; columna < columnas; columna++)
  {
    //como es una imagen en blanco y negro nos da igual tomar la componente R, G o B
    sumaActual += imagenBinarizada.GetPixel(columna, fila).R;
  }
  if (sumaActual < sumaMinima)
  {
    sumaMinima = sumaActual;
    filaSumaMinima = fila;
  }
 }
```

y cuya componente Y se corresponde con la fila de mayor número de píxeles negros y se calcula de forma análoga.

3.1.3. *Encontrar centro de la pupila (método iterativo)*

Se utiliza la clase PasoCentroPupila. Se localiza el centro de la pupila con un método iterativo:

```
int numIter = 0;
do
{
  puntoNuevo = CalculaPuntoNuevo(puntoActual);
  distanciaEntrePuntos = puntoNuevo.Distancia(puntoActual);
  puntoActual = puntoNuevo;
  numIter++;
}
while ((distanciaEntrePuntos > ErrorMaximoPermitido)
       && (numIter < NumIteracionesMax));
```

Utilizamos un método auxiliar para calcular el punto nuevo en cada iteración:

```
private Vector2 CalculaPuntoNuevo(Vector2 puntoActual)
```

Dicho método nos devuelve el punto centro de la pupila.
El proceso por el que se obtiene la componente X centralizada es:

```
while(columna < columnas)
{
   if (imagenBinarizada.GetPixel(columna, fila).R > 0) break;
   columna++;
}
int columnaDerecha = columna;
```

De esta forma obtenemos el número de columnas de pupila que hay a la derecha del punto.

```
while (columna >= 0)
{
  if (imagenBinarizada.GetPixel(columna, fila).R > 0)  break;
  columna--;
}

int columnaIzquierda = columna;
```

Y de esta forma el número de columnas de pupila a la izquierda del mismo.
Haciendo una media aritmética entre esos 2 valores,

```
int mediaColumna = (columnaDerecha + columnaIzquierda) / 2;
```

obtenemos una columna más centralizada que la del punto inicial.
De forma análoga se procede con la componente Y.
El método suele converger al centro de la pupila, pero se ha puesto un límite de iteraciones ya que se han encontrado algunos casos (muy poco comunes) en los que esto no ocurre.

3.1.4. *Encontrar radio de la pupila*
Se utiliza la clase PasoRadioPupila. Se determina el radio de la pupila sumando el número de píxeles negros consecutivos que hay encima, debajo, a izquierda y a derecha del centro de la pupila y dividiendo el resultado obtenido entre 4.

```
int distancia1 = Math.Abs(PuntoPupila.X - columnaDerecha);
int distancia2 = Math.Abs(PuntoPupila.X - columnaIzquierda);
int distancia3 = Math.Abs(PuntoPupila.Y - filaAbajo);
int distancia4 = Math.Abs(PuntoPupila.Y - filaArriba);

RadioPupila = (distancia1 + distancia2 + distancia3 + distancia4) / 4;
```

3.2. *Detección del límite iris-esclerótica*

3.2.1. *Cálculo de círculos interno y externo*
Se utiliza la clase PasoCirculosExternoInterno. El círculo interno se calcula de forma inmediata:

```
RadioCirculoInterno = RadioPupila + SumaRadioPupila;
```

El círculo externo es el más grande posible que quepa en la imagen y cuyo centro sea CentroPupila:

```
int filas = imagenInicial.Height;
int columnas = imagenInicial.Width;

int distanciaHorizontal1 = CentroPupila.X;
int distanciaVertical1 = CentroPupila.Y;
int distanciaHorizontal2 = columnas - CentroPupila.X;
int distanciaVertical2 = filas - CentroPupila.Y;

int radioCirculoExterno =
      Math.Min(Math.Min(distanciaHorizontal1, distanciaHorizontal2),
      Math.Min(distanciaVertical1, distanciaVertical2));
```

3.2.2. *Filtro Gaussiano*

Se utiliza la clase PasoFiltroGaussiano, y para realizar el filtro se utiliza el método Apply de la clase GaussianBlur de la biblioteca aForge.NET:

```
GaussianBlur filter = new GaussianBlur(tamanoKernel, sigma);
Bitmap ImagenFiltroGaussiano = filter.Apply(ImagenInicial);
```

3.2.3. *Cálculo de puntos con mayor gradiente*

Se utiliza la clase PasoPuntosGradienteIris. Se trata de hacer un barrido en la zona situada entre el círculo interno y el externo calculados anteriormente:

```
//zona izquierda
for (int d = 0; d < 5; d++)
{
  double direccion = Math.PI + d * 0.15;
  for (int i = distanciaInicial; i < distanciaFinal; i++)
     EncuentraPuntosGradiente(i, direccion);
}

//zona derecha
for (int d = 0; d < 5; d++)
{
  double direccion = -d * 0.15;
  for (int i = distanciaInicial; i < distanciaFinal; i++)
     EncuentraPuntosGradiente(i, direccion);
}
```

3.2.4. *Determinar el radio del círculo que separa el iris de la esclerótica*

Se utiliza la clase PasoRadioIris. Se calcula la banda de anchura 16 que agrupa mayor cantidad de puntos de gradiente significativo:

```
int sumaMaxima = 0, radioSumaMaxima = 0;

for (int i = radioMinimo; i < radioMaximo; i++)
{
  int suma = NumeroValoresEnBanda(puntos, i);
  if (suma > sumaMaxima)
  {
    sumaMaxima = suma;
    radioSumaMaxima = i;
  }
 }
```

Se utiliza el método auxiliar

```
private int NumeroValoresEnBanda(List<Vector2> lista, int centroBanda)
```

para contar el número de valores de la lista de puntos que entran en la banda que le pasamos como parámetro.
El radio del iris será la media de las distancias de los puntos comprendidos en la banda al centro de la pupila:

```
int radioIris = MediaValoresEnBanda(puntos, radioSumaMaxima);
```

3.3. *Localización de párpados*

3.3.1. *Determinar una parábola virtual superior e inferior*

Para este paso se utilizan las clases PasoParabolaVirtualSuperior y PasoParabolaVirtualInferior que son hijas de la clase PasoParabolaVirtual.

El código de la clase PasoParabolaVirtual es:

```
InicializaPuntos();

double denominador = Math.Pow(puntoParabola1.X - verticeParabola.X, 2);
double t = (puntoParabola1.Y - verticeParabola.Y) / denominador;

parabola = new Parabola();
parabola.a = t;
parabola.b = -2 * t * verticeParabola.X;
parabola.c = (t * Math.Pow(verticeParabola.X, 2)) + verticeParabola.Y;

AsignaParabolaAEstado();
```

Donde los métodos InicializaPuntos() y AsignaParabolaAEstado() son definidos por las clases PasoParabolaVirtualSuperior y PasoParabolaVirtualInferior de forma que la parábola virtual superior pasa por el punto (CentroPupila.X - RadioIris, CentroPupila.Y) así como por el punto (CentroPupila.X + RadioIris, CentroPupila.Y) y tiene su vértice en el punto (CentroPupila.X, CentroPupila.Y - RadioPupila - 10). Análogamente se procede con la parábola virtual inferior.

3.3.2. *Cálculo de puntos con mayor gradiente en los párpados*

Para este paso se utiliza una clase padre PasoPuntosGradienteParpado, con sus dos clases hijas PasoPuntosGradienteParpadoSuperior y PasoPuntoGradienteParpadoInferior. El código de la clase padre es:

```
//además de los puntos anterior y posterior en la dirección
Vector2 puntoAnterior = new Vector2(puntoX, puntoY - 1);
Vector2 puntoPosterior = new Vector2(puntoX, puntoY + 1);

//calculamos la variación de intensidad del punto anterior al posterior
int intensidadPuntoAnterior =
      imagenFiltroGaussiano.GetPixel(puntoAnterior.X, puntoAnterior.Y).R;
int intensidadPuntoPosterior =
      imagenFiltroGaussiano.GetPixel(puntoPosterior.X, puntoPosterior.Y).R;

int variacion = Math.Abs(intensidadPuntoAnterior - intensidadPuntoPosterior);

//nos quedamos sólo con aquellos con una variación considerable
if (variacion > limiteInferior)
{
  Vector2 punto = new Vector2(puntoX, puntoY);
  puntosGradiente.Add(punto);
}
```

3.3.3. *Aplicar filtro a puntos candidatos*

Se utiliza la clase PasoFiltroCandidatosGradiente. Para el párpado superior, de los puntos seleccionados anteriormente nos quedaremos sólo con aquellos que cumplan las tres condiciones siguientes: 1. Tener un valor de intensidad inferior a 130.

```
byte intensidad = imagen.GetPixel(p.X, p.Y).R;
criterio1 = intensidad < 130;
```

2. Tener un valor de intensidad muy similar al de alguno de sus vecinos situados a su izquierda.

```
  byte? intensidadVecino1 =
AyudanteImagenes.Intensidad(imagen, p.X - 1, p.Y - 1);
  byte? intensidadVecino2 =
AyudanteImagenes.Intensidad(imagen, p.X - 1, p.Y);
  byte? intensidadVecino3 =
AyudanteImagenes.Intensidad(imagen, p.X - 1, p.Y + 1);

  criterio2 = Similar(intensidad, intensidadVecino1)
 || Similar(intensidad, intensidadVecino2)
 || Similar(intensidad, intensidadVecino3);
```

3. Tener un valor de intensidad muy similar al de alguno de sus vecinos situados a su derecha.

```
byte? intensidadVecino4 =
        AyudanteImagenes.Intensidad(imagen, p.X + 1, p.Y - 1);
byte? intensidadVecino5 =
```

```
AyudanteImagenes.Intensidad(imagen, p.X + 1, p.Y);
  byte? intensidadVecino6 =
          AyudanteImagenes.Intensidad(imagen, p.X + 1, p.Y + 1);

  criterio3 = Similar(intensidad, intensidadVecino4)
         || Similar(intensidad, intensidadVecino5)
         || Similar(intensidad, intensidadVecino6);
```

Finalmente añadimos los puntos que cumplen los 3 criterios:

```
  if(criterio1 && criterio2 && criterio3) nuevaListaPuntosSup.Add(p);
```

Para el párpado inferior, procederemos de forma análoga.
Finalmente añadimos los puntos que cumplen los 3 criterios:

```
  if (criterio1 && criterio2 && criterio3) nuevaListaPuntosInf.Add(p);
```

3.3.4. *Determinar parábolas de mínimos cuadrados*

Se utiliza la clase PasoParabolaMinimosCuadrados. Determinaremos que se trata de un párpado si una cantidad considerable de puntos situados en la zona superior(inferior) del iris pasaron el filtro anterior, esto lo detectamos con el método ParpadoEncontrado() que es definido en las clases hijas PasoParabolaMinimosCuadradosSuperior y PasoParabolaMinimosCuadradosInferior.

A continuación se calcula la parábola de mínimos cuadrados para estos puntos:

```
  foreach (Vector2 punto in listaPuntos)
  {
    BigInteger puntoX = punto.X;
    BigInteger puntoY = punto.Y;

    s10 += puntoX; // Suma de x
    s20 += puntoX * puntoX; // Suma de x cuadrada
    s30 += puntoX * puntoX * puntoX; // Suma de x cúbica
    s40 += puntoX * puntoX * puntoX * puntoX; // Suma x cuarta

    s01 += puntoY; // Suma de y
    s11 += puntoX * puntoY; // Suma de xy
    s21 += puntoX * puntoX * puntoY;
  }

  BigInteger numA = (s21 * (s20 * s00 - s10 * s10)
            - s11 * (s30 * s00 - s10 * s20)
            + s01 * (s30 * s10 - s20 * s20));
  BigInteger numB = (s40 * (s11 * s00 - s01 * s10)
            - s30 * (s21 * s00 - s01 * s20)
            + s20 * (s21 * s10 - s11 * s20));
  BigInteger numC = (s40 * (s20 * s01 - s10 * s11)
            - s30 * (s30 * s01 - s10 * s21)
            + s20 * (s30 * s11 - s20 * s21));
  BigInteger deno = (s40 * (s20 * s00 - s10 * s10)
```

```
        - s30 * (s30 * s00 - s10 * s20)
        + s20 * (s30 * s10 - s20 * s20));
```

El resultado será la aproximación al párpado superior (inferior).

3.4. *Eliminación de pestañas*

Se localizan aquellos píxeles candidatos a pertenecer a una pestaña, ya que las pestañas son más oscuras que el iris:

```
for (int x = 0; x < imagenIrisDelimitado.Width; x++)
{
  for (int y = 0; y < imagenIrisDelimitado.Height; y++)
  {
    Vector2 punto = new Vector2(x, y);

    //si es lo suficientemente oscuro, lo marcamos como posible pestaña
    if (imagenIrisDelimitado.GetPixel(x, y).R < LimiteOscuro)
       puntosPosiblesPestanas.Add(punto);
  }
}
```

Si se detectó un número de píxeles oscuros superior al parámetro SuficientesPuntos, se considera que la imagen original tenía pestañas sobre el iris y se procede a la eliminación de estos píxeles oscuros:

```
int numeroPuntos = PuntosPosiblesPestanas.Count;

//si hay suficientes puntos
if (numeroPuntos > SuficientesPuntos)
{
  //los borramos
  foreach (Vector2 punto in Estado.Instance.PuntosPosiblesPestanas)
     imagenIrisDelimitado.SetPixel(punto.X, punto.Y, Color.White);
}
```

Con esto concluye el proceso, el iris ha quedado completamente delimitado y libre de pestañas.

Si no se detectó un número suficiente de píxeles, se considera que no hay pestañas en la imagen original y no se hace nada.

4. Manual de usuario

Hemos desarollado dos aplicaciones, una con interfaz gráfica (IrisPID) y otra por consola (IrisPIDConsola).

La primera consta de una ventana principal dividida en dos regiones. En la región derecha vemos una imagen en la que se van mostrando los progresos del algoritmo. En la región izquierda se puede observar información acerca del siguiente paso a ejecutar en el algoritmo, incluyendo la fase a la que dicho paso pertenece y una descripción del mismo. En esta sección aparecen también los botones que detallaremos a continuación.

- Botón “Aplicar Paso”: aplica el siguiente paso del algoritmo.

- Botón “Aplicar Completo”: aplica de manera automática todos los pasos a realizar generando la imagen resultante del proceso.

Desde el menu superior se puede abrir una nueva imagen para su tratamiento, salir de la aplicación y ver la ayuda.

La aplicación por consola se usa principalmente para la experientación. Pide al usuario una carpeta que posteriormente recorre, aplicando el método de localización del iris. Almacena los resultados en una carpeta 'salida' para su posterior análisis.

5. Experimentación

Para realizar la experimentación hemos utilizado la base de imágenes de iris CASIA v1, que contiene 756 imágenes de iris y es la base de imágenes en la que se basaron los autores del artículo que hemos seguido como referencia. Debemos agradecer al Center for Biometrics and Security Research de la Chinese Academy of Sciences que nos dejaran usar esta base de imágenes privada.
Los resultados esperimentales nos muestran que para un ordenador Pentium IV, 2.0GHz Dual Core, 4GB RAM el tiempo medio empleado en el procesado de una imagen es de aproximadamente un segundo.
El porcentaje de éxito en la localización del iris es del 96.43 %, con tan solo 27 imágenes problemáticas de las 756. Además, de estas 27 imágenes problemáticas, 22 se deben a que el iris no se encuentra contenido completamente en los límites de la imagen, por lo que su correcta localización no es posible.
Podría intentarse ajustar los diferentes parámetros del algoritmo para conseguir que las 5 imágenes restantes (0.6 % de la base de imágenes original) funcionaran. Cada una de estas 5 imágenes falla porque o bien el límite exterior del iris aparece difuso, o bien hay una presencia masiva de pestañas.
En cualquier caso, consideramos que el porcentaje de acierto es muy alto y por tanto la aplicación muy fiable.

6. Carga de trabajo

Fecha de la reunión	Inicio	Fin	Tiempo total empleado	Miembro del grupo	Actividad
22-11-2010	15:30	17:30	2h	Juan, Borja, Pascual	Comprensión del artículo de referencia
24-11-2010	15:30	17:30	2h	Juan, Borja, Pascual	Comprensión del artículo de referencia
24-11-2010	17:30	18:30	1h	Borja, Pascual	Comprensión del artículo de referencia
29-11-2010	15:00	19:00	4h	Juan, Borja, Pascual	Análisis teórico
01-12-2010	15:30	17:30	2h	Juan, Borja, Pascual	Interfaz Gráfica
01-12-2010	17:30	19:30	2h	Juan, Pascual	Interfaz Gráfica
06-12-2010	15:30	19:30	4h	Juan, Borja, Pascual	Interfaz Gráfica
08-12-2010	15:30	17:30	2h	Juan, Borja, Pascual	Comprensión del artículo de referencia
08-12-2010	17:30	19:30	2h	Juan, Pascual	Implementación
13-12-2010	15:30	19:30	4h	Borja	Implementación
15-12-2010	09:30	14:30	5h	Juan	Implementación
15-12-2010	09:30	14:30	5h	Borja	Implementación
16-12-2010	09:30	14:30	5h	Pascual	Implementación
20-12-2010	09:00	12:00	3h	Juan	Implementación
21-12-2010	15:30	19:30	2h	Pascual	Implementación
21-12-2010	09:00	14:00	5h	Borja	Implementación
22-12-2010	15:30	19:30	4h	Juan	Implementación
22-12-2010	09:00	12:00	3h	Pascual	Implementación
23-12-2010	08:30	17:30	9h	Borja	Implementación
23-12-2010	09:00	17:00	8h	Juan	Implementación
23-12-2010	08:00	16:00	8h	Pascual	Implementación
29-12-2010	09:00	12:00	3h	Juan	Documentación
29-12-2010	10:30	14:30	4h	Borja	Documentación
30-12-2010	09:00	12:00	3h	Borja	Documentación
02-01-2011	10:00	14:00	4h	Pascual	Documentación
02-01-2011	09:00	14:00	5h	Juan	Documentación
02-01-2011	15:30	17:30	2h	Pascual	Documentación

7. Conclusiones

El algoritmo implementado cumple sus expectativas, es decir, la localización del iris en imágenes en escala de grises. La fotografía a procesar ha de tener unas características adecuadas: una foto de un único ojo en escala de grises con resolución de 320x280 pixeles y una iluminación adecuada.

Encontramos que la mayoría de los pasos se realizan de forma casi instantánea, el único paso que se demora un poco es el filtro gaussiano, sería interesante mejorar los tiempos de respuesta.

Puesto que el artículo no dejaba claro algunos detalles con respecto a la distancia de Mahalanobis hemos decidido usar distancia euclidea.

Aunque se han alcanzado todos los objetivos del trabajo proponemos como posibles mejoras las siguientes:

- Podría adaptarse para funcionar con otras versiones de esta base de datos de imágenes o incluso con bases de datos diferentes.

- Se podría aumentar la compatibilidad permitiendo la entrada de imágenes de diferentes tamaños, formatos de imagen y color.

Posibles ampliaciones:

- Una vez localizado el iris, podría procederse a realizar una labor de reconocimiento.

Referencias

[1] A. Basit, M.Y. Javed, "Localization of iris in gray scale images using intensity gradient", *Image and Vision Computing*, Vol. 27, pp. 1134-1142, 2009.

[2] `http://www.cbsr.ia.ac.cn/IrisDatabase.htm`