Reconocimiento de Huellas Dactilares

Proyecto de Biometría

Samuel Moreno Vicent

Miguel Torres Pérez

Tabla de contenido

Algoritmos utilizados

Consideraciones a tener en cuenta:

- En el programa se hace uso de un Array de matrices donde guardaremos el estado de la imagen según el paso en el que nos encontremos, de esta forma podremos retroceder si así lo necesitamos.

Escala de Grises

Uno de los primeros pasos implementado es el llamado "escala de grises". Este algoritmo se encarga de transformar una imagen RGB a color en una imagen en escala de grises.

Para ello, el algoritmo recorre la matriz que conforma la imagen y por cada pixel sustituye el valor en R (rojo) G (verde) y B (azul) por la media de estos valores y lo coloca en las tres posiciones de tal manera que el gris tendrá una intensidad proporcional a la siguiente fórmula:

$$nivel\ de\ Gris = \frac{(R + G + B)}{3}$$

Una vez calculado el valor de cada pixel la matriz se almacenará en el array de matrices en la posición relativa al primer paso.

Ecualizar

El segundo paso implementado es el llamado "ecualizar". Este algoritmo se encarga de distribuir uniformemente los valores de una imagen en escala de grises.

Para ello, el algoritmo recorre la matriz que conforma la imagen y calcula el histograma. El histograma es una representación gráfica de la distribución de los distintos tonos de una imagen. A continuación, recorremos de nuevo la matriz de la imagen y sustituimos los pixeles con el nuevo valor calculado para distribuir los tonos de la imagen.

La idea de este algoritmo es que existan valores entre 0 y 255 y además estén distribuidos uniformemente. Sin embargo, el método ecualizar suele estropear la imagen y por eso hemos hecho que sea **OPCIONAL**.

Una vez calculado el valor de cada pixel la matriz se almacenará en el array de matrices en la posición relativa al segundo paso.

Calcular Umbral Automáticamente

Este algoritmo es un **EXTRA** que hemos implementado. La idea de este algoritmo es liberar al usuario de establecer un umbral manualmente. Aunque el usuario sigue teniendo la oportunidad de establecerlo.

Para calcular el umbral, el algoritmo recorre la matriz que conforma la imagen y calcula el valor medio acumulado de los pixeles de la imagen. Sin embargo, no todos los valores tienen el mismo peso. Se ha implementado una función que tiene en cuenta los valores repetidos para que cada vez pesen menos. De esta forma el fondo blanco no tendrá tanto peso en el cálculo umbral.

El algoritmo ha sido probado con éxito en varias imágenes con distintas distribuciones de grises y con distintas cantidades de fondo blanco.

Una vez calculado el umbral este se utiliza para binarizar una imagen.

Binarizar

El tercer paso implementado es el llamado "binarizar". Este algoritmo se encarga de transformar la imagen en escala de grises a **blanco y negro**.

Para ello, el algoritmo recorre la matriz que conforma la imagen y para cada pixel escribe un 0 (negro) o un 1 (blanco) en función del umbral. Si es menor escribirá 0 y si es mayor un 1.

La idea de este algoritmo es que existan únicamente valores 0's ó 1's. Como ya hemos mencionado anteriormente el usuario puede establecer un valor de umbral por medio de una barra deslizable en la interfaz, pero sino se establece un cálculo automático del umbral para ahorrar tiempo al usuario.

Una vez calculado el valor binario cada pixel, la matriz se almacenará en el array de matrices en la posición relativa al tercer paso.

Filtrar

El cuarto paso implementado es el llamado "filtra". Este algoritmo se encarga de transformar la imagen eliminando el **ruido binario**.

Para ello, el algoritmo recorre la matriz que conforma la imagen y para cada pixel decide si es un blanco o un negro en función de sus pixeles vecinos. Con este filtro se reducen efectos como: contornos irregulares, pequeños huecos, esquinas perdidas y puntos aislados.

Para cada pixel aplica una ventana de vecindad de 3x3 en la que se aplican 2 filtros:

- La función del filtro Binario 1:

$$B1=p+b.g.(d+e)+d.e.(b+g)$$

- La función del filtro Binario 2:

$$B2=p[(a+b+d).(e+g+h)+(b+c+e).(d+f+g)]$$

a	b	c
d	p	e
f	g	h

Vecindad de 3x3

Los "." son AND lógicos y los "+" son OR lógicos.

Una vez calculado el estado de cada pixel, la matriz se almacenará en el array de matrices en la posición relativa al cuarto paso.

Adelgazamiento

El quinto paso implementado es el llamado "adelgazamiento". Este algoritmo se encarga de adelgazar los trazos de la huella.

Para ello, usamos el conocido algoritmo de Zhang-Suen para java. El algoritmo trabaja por iteraciones. En cada iteración, se tienen dos sub-iteraciones en cada una de las cuales, se evalúa cada pixel de la imagen en base a cuatro condiciones que, de cumplirse, permiten que el pixel en cuestión sea borrado por no tratarse de una parte fundamental del esqueleto de la huella. El resultado de la imagen son los pixeles que hayan cumplido las condiciones de borrado, y serán la entrada para la siguiente sub-iteración. La última iteración se da cuando se cumplen dos sub-iteraciones donde ningún pixel se borra.

Una vez adelgazada la huella, la matriz se almacenará en el array de matrices en la posición relativa al tercer paso.

Detección de minutias

El sexto paso implementado es el llamado "detección de minutias". Este algoritmo se encarga de encontrar las minutias de una huella y almacenarlas en un array de minutias

Para ello, recorremos la matriz y para cada pixel comprobamos si tiene vecinos. En el caso de que un pixel tenga uno o tres vecinos será una minutia. Y se añadirá en el array de minutias con las coordenadas y el tipo de la minutia. Que será tipo uno o tres en función de los vecinos que tuviera.

Una vez procesada la huella para encontrar las minutias se pasa al siguiente paso.

Calculo del Ángulo

El séptimo y último paso implementado es el llamado "Calcular ángulo". Este algoritmo encarga de calcular el **ángulo** de las minutias de una huella y guardar los datos de la minutia en un **fichero** para que los datos persistan.

Para ello, Este algoritmo consta de dos partes bien diferenciadas una para las minutias de tipo1 y otra paras las de tipo 3.

Para las de tipo 1 el algoritmo va recorriendo hasta 6 vecinos de la minutia y cuando llega al último calcula el gradiente en el eje X y el gradiente en el eje Y. Dado que nuestro sistema de coordenadas x e y difiere del eje cartesiano es necesario realizar algunos ajustes:

- Invertir de signo el valor del gradiente Y.

Por último, realizamos el arco tangente de la división del gradiente Y entre el X:

$$angulo = \arctan\left(\frac{Gy}{Gx}\right)$$

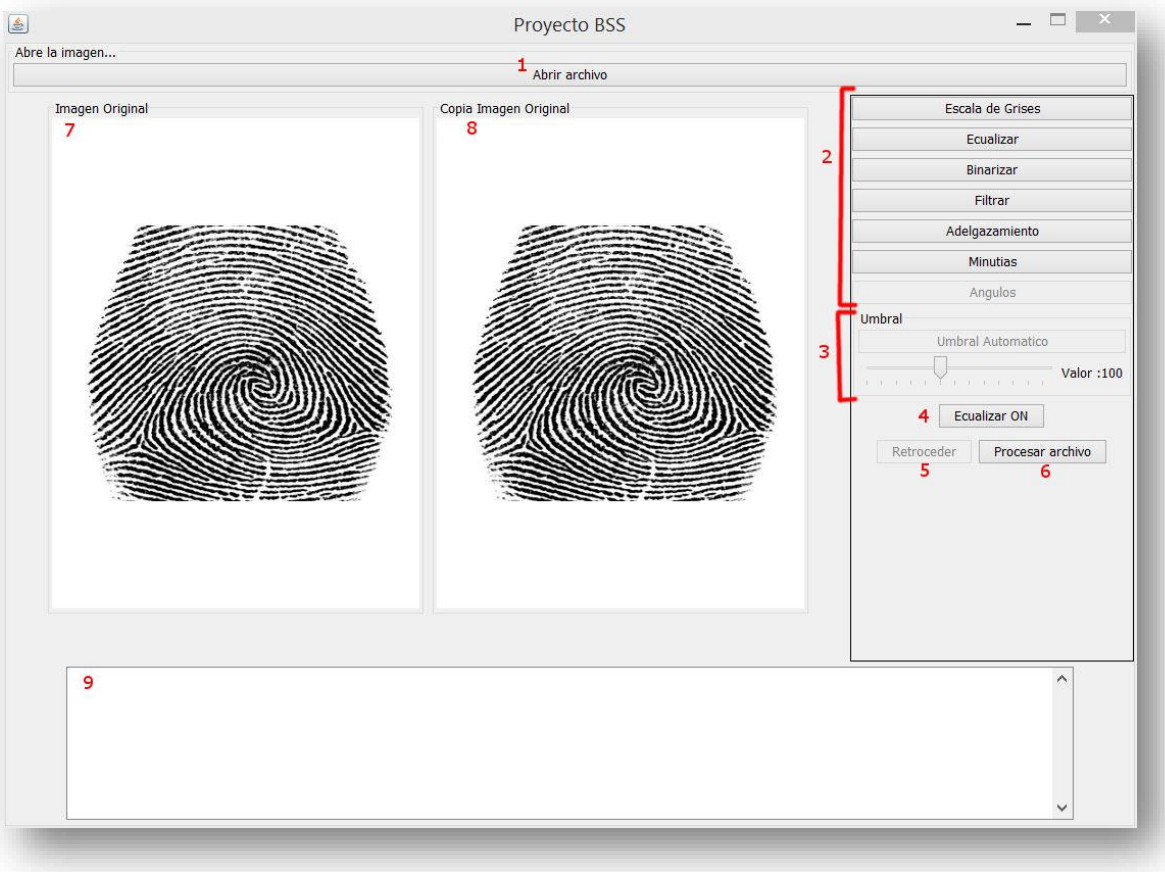
Para las de tipo 3 como la minutia tiene 3 vectores el algoritmo va recorriendo hasta 6 vecinos de cada uno de los vectores y cuando llega al último calcula el gradiente en el eje X y el gradiente en el eje Y de cada vector y luego al final se suman las componentes X's e Y's de los 3 vectores. De esta manera obtendremos el ángulo resultante de una minutia de bifurcación. Dado que nuestro sistema de coordenadas x e y difiere del eje cartesiano es necesario realizar algunos ajustes:

- Invertir de signo el valor del gradiente Y.

Por último, realizamos el arco tangente de la división del gradiente Y entre el X:

$$angulo = \arctan\left(\frac{Gy}{Gx}\right)$$

Interfaz de usuario



La Interfaz de usuario consiste en una única ventana donde se mostrara toda la información necesaria para la realización de la extracción de minucias.

1 El botón "Abrir archivo" sirve para elegir la imagen que se va a abrir dentro del directorio del ordenador, por defecto abrirá la carpeta Huellas si existiera.

2 El menú de procesos consta de 7 botones que representan los 7 pasos a seguir en el procesamiento de la imagen. Gracias a este menú podemos ir al paso que queramos directamente, realizando todos los pasos intermedios de manera automática sin necesidad de ir de uno en uno. El botón del proceso actual quedara desactivado para saber por cual vamos. **Importante**: el botón de Ángulos es independiente al resto. Cuando vayamos paso por paso este método no se realizará, solo se podrá realizar cuando hayamos extraído las minutias. El resultado quedara escrito en el documento generado.

3 La barra de umbral podrá ser utilizada desde que realicemos el método de binarizado para cambiar el umbral de binarización, el botón de Umbral Automático estará disponible también en ese momento y calculara el mejor umbral para la imagen.

4 Como hay algunas imágenes para las que el ecualizar no las mejora, decidimos poner un botón que desactivara este método, de manera que con el ecualizar desactivado (OFF) al dar a ecualizar nos dejara la imagen en grises que teníamos antes.

5 El botón retroceder nos lleva a la tarea anterior a la actual, en caso de que estemos en la primera tarea se desactivara el botón.

6 El botón procesar archivo realizara la tarea siguiente, mostrando los resultados. La ultima tarea es el cálculo de minutias, una vez ahí el botón quedará desactivado y el cálculo de los ángulos se hará solo al pulsar ese botón.

7 En este hueco se mostrará la imagen anterior a la tarea actual, de manera que podamos compararla con la procesada.

8 Aquí se muestra la imagen procesada después de la realización de la tarea actual.

9 En el cuadro de texto se mostrarán los mensajes del sistema necesarios para saber que se está ejecutando en cada momento.

Clases y Módulos implementados

Clase Main

Esta clase se encarga únicamente de iniciar la ejecución del programa. Inicializa la interfaz y deja que ella haga el resto.

Clase Interfaz

En esta clase se realizan todas las tareas relacionadas con la interfaz, inicializarla, añadir los botones, cambiar las imágenes, cambiar los textos, etc.

En el **constructor** de la interfaz creamos la ventana de ejecución de la aplicación, inicializando todos los valores necesarios, añadiendo los botones, las barras y todo lo que está dentro de la ventana.

El método **actionPerformed** es el Listener encargado de asignar a cada botón (“abrir archivo”, “Escala de Grises”, “Ecualizar”, “Binarizar”, “Filtrar”, “Adelgazamiento”, “Minutias”, “Ángulos”, “Umbral Automatico”, “Ecualizar ON/OFF”, “Retroceder” y “Procesar”) la acción que debe realizar al ser pulsado.

El método **ecualizarOnOff** se encarga de modificar las acciones necesarias para que el ecualizar quede activado o desactivado, se llama desde el botón "Ecualizar ON/OFF" y además cambia el texto del botón en la interfaz.

El método **resize** se encarga de, al abrir una imagen nueva, redimensionarla a un máximo de 390*500 en el caso de ser superior a estas dimensiones. Esto se hace para que la imagen quepa dentro de la ventana de la interfaz y se pueda ver completa.

El método **abrirArchivo** se encarga de abrir una imagen nueva y reiniciar todas las variables de imágenes anteriores para que no interfieran con la nueva.

El método **llenarArray** es el encargado de procesar la imagen hasta la tarea actual, esto significa que, si se cambia el umbral del binarizado en cualquier momento, este método volverá a calcular las tareas anteriores necesarias para que se pueda mostrar la actual sin ningún problema.

Los métodos **pintarTarea** y **cambiarNombres** se encargan de mostrar la tarea actual en la interfaz, cambiando la disponibilidad de los botones, las imágenes y sus nombres.

Clase Procesar Imagen:

El método **escalaGrises** se encarga de dada la imagen inicial, hacer la media de los 3 colores RGB y pasarlos a gris.

El método **ecualizar** calcula el histograma de la imagen en gris y mediante la Lookup table transformamos la imagen para que se vea más nítida.

El método **calcularUmbral** calcula el mejor umbral para binarizar la imagen, se utilizar en el botón calcular umbral.

El método **binarizar** pasa la imagen de escala de grises a blanco y negro, para cada pixel, si es mayor que el valor de umbralizar se pasa a blanco (1) y si es menor se pasa a negro (0).

El método **filtrar** toma la imagen en blanco y negro y la mejora rellenando huecos blancos y borrando puntos negros sueltos.

El método **numNeighbors** se utiliza en el método de adelgazamiento, sirve para calcular el número de vecinos negros de cada pixel.

El método **numTransitions** se utiliza en el método de adelgazamiento y en la extracción de minucias, sirve para calcular el número de aristas que salen del pixel actual, es decir, el número de vecinos separados entre sí.

El método **atLeastOneIsWhite** se utiliza en el método de adelgazamiento, sirve para comprobar que al menos un pixel de los de alrededor es blanco.

El método **adelgazamiento** toma la imagen en blanco y negro y reduce las líneas negras de manera que acaben siendo de un solo pixel de grosor.

El método **extraccionDeMinutias** toma la imagen adelgazada y para cada pixel negro comprueba el número de aristas que salen de él, si es 1 o 3 es una minucia y se añade al vector de m

El método **pintarMinutias** utiliza la imagen adelgazada y el vector de minucias para crear una nueva imagen que solo se utiliza para mostrar las minucias con un círculo rojo o naranja (dependiendo de si es tipo 1 o 3).

El método **calcularAngulo** utilizar la imagen adelgazada y el vector de minucias para calcular el ángulo de cada una de ellas, recorre 6 pixeles de cada arista y calcula el ángulo para cada una de ellas, para las de tipo 3 realizamos la suma de los tres vectores que hemos obtenido en formato módulo argumento.

El método **calcularMargen** busca el inicio de la huella en la imagen y guarda los valores de los márgenes para que la extracción de minucias no tenga en cuenta las terminaciones del exterior.

Clase Minutia:

Guarda los datos de una minutia, sus coordenadas X e Y, el tipo (1 si es una terminación y 3 si es una bifurcación) y el ángulo. Sus métodos sirven únicamente para acceder y cambiar estos valores.

Clase Matriz:

Contiene una matriz de short donde se guardan los valores de los pixeles de la imagen, además guarda el valor del largo y del alto de la imagen.

Sus métodos **getWidth** y **getHeight** sirven para extraer los valores de alto y ancho

setPixel y **getPixel** sirven para obtener el valor de un pixel, asignar el valor de un pixel respectivamente

getImagenGris devuelve una imagen de la matriz en nivel de grises

getImagenBinariaConRojo devuelve una imagen de la matriz en blanco y negro (también tiene la opción de mostrar el rojo y el naranja para mostrar un circulo en la posición de las minucias).

Clase Procesar Imagen

El método **escalaGrises** se encarga de dada la imagen inicial, hacer la media de los 3 colores RGB y pasarlos a gris.

El método **ecualizar** calcula el histograma de la imagen en gris y mediante la Lookup table transformamos la imagen para que se vea más nítida.

El método **calcularUmbral** calcula el mejor umbral para binarizar la imagen, se utilizar en el botón calcular umbral.

El método **binarizar** pasa la imagen de escala de grises a blanco y negro, para cada pixel, si es mayor que el valor de umbralizar se pasa a blanco (1) y si es menor se pasa a negro (0).

El método **filtrar** toma la imagen en blanco y negro y la mejora rellenando huecos blancos y borrando puntos negros sueltos.

El método **numNeighbors** se utiliza en el método de adelgazamiento, sirve para calcular el número de vecinos negros de cada pixel.

El método **numTransitions** se utiliza en el método de adelgazamiento y en la extracción de minucias, sirve para calcular el número de aristas que salen del pixel actual, es decir, el número de vecinos separados entre sí.

El método **atLeastOneIsWhite** se utiliza en el método de adelgazamiento, sirve para comprobar que al menos un pixel de los de alrededor es blanco.

El método **adelgazamiento** toma la imagen en blanco y negro y reduce las líneas negras de manera que acaben siendo de un solo pixel de grosor.

El método **extraccionDeMinutias** toma la imagen adelgazada y para cada pixel negro comprueba el número de aristas que salen de él, si es 1 o 3 es una minucia y se añade al vector de m

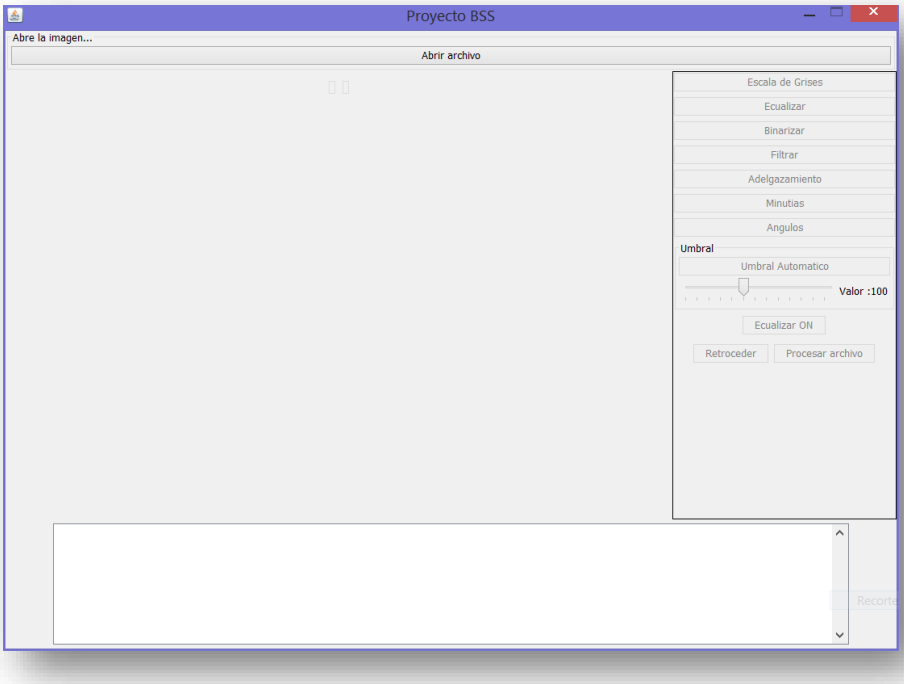
El método **pintarMinutias** utiliza la imagen adelgazada y el vector de minucias para crear una nueva imagen que solo se utiliza para mostrar las minucias con un círculo rojo o naranja (dependiendo de si es tipo 1 o 3).

El método **calcularAngulo** utilizar la imagen adelgazada y el vector de minucias para calcular el ángulo de cada una de ellas, recorre 6 pixeles de cada arista y calcula el angulo para cada una de ellas, para las de tipo 3 realizamos la suma de los tres vectores que hemos obtenido en formato módulo argumento.

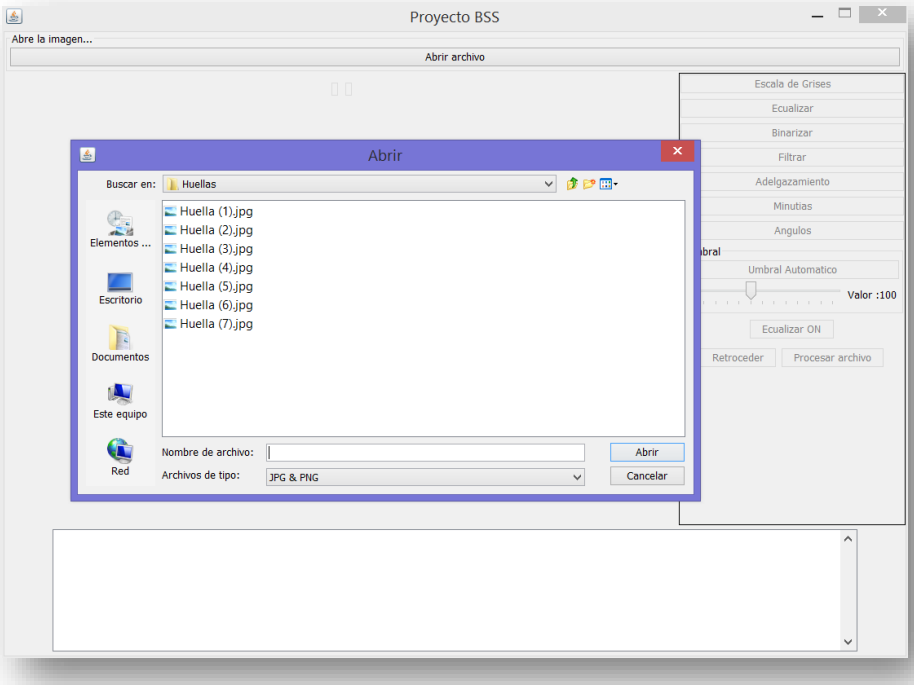
El método **calcularMargen** busca el inicio de la huella en la imagen y guarda los valores de los márgenes para que la extracción de minucias no tenga en cuenta las terminaciones del exterior.

Ejemplo de Ejecución

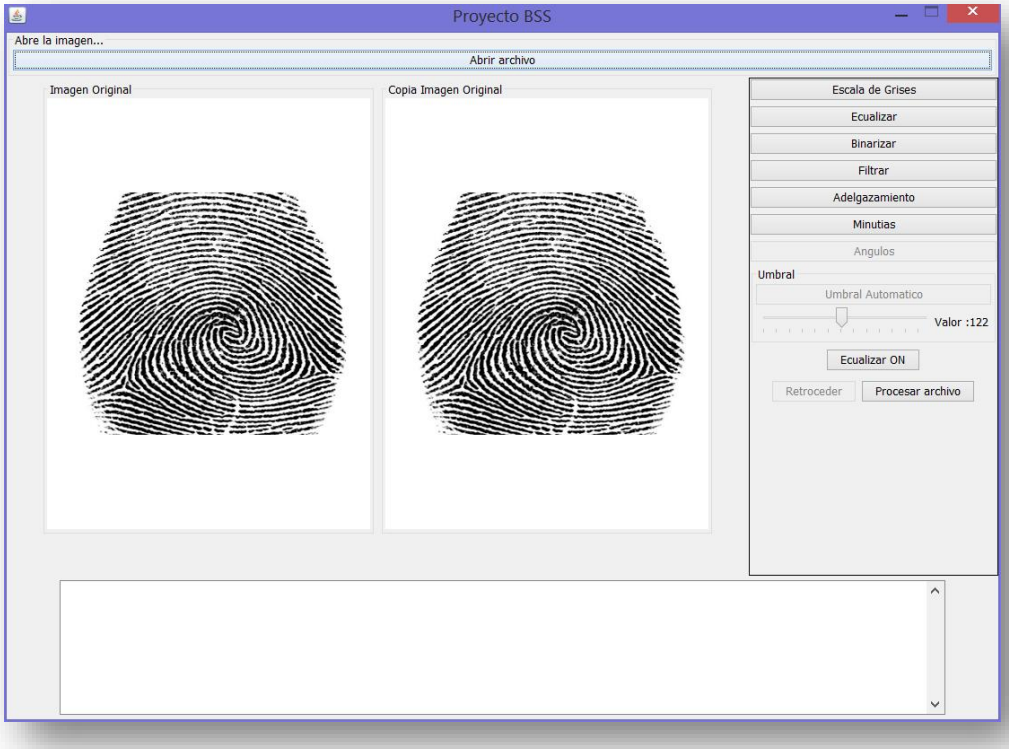
Ventana inicial al ejecutar la aplicación:



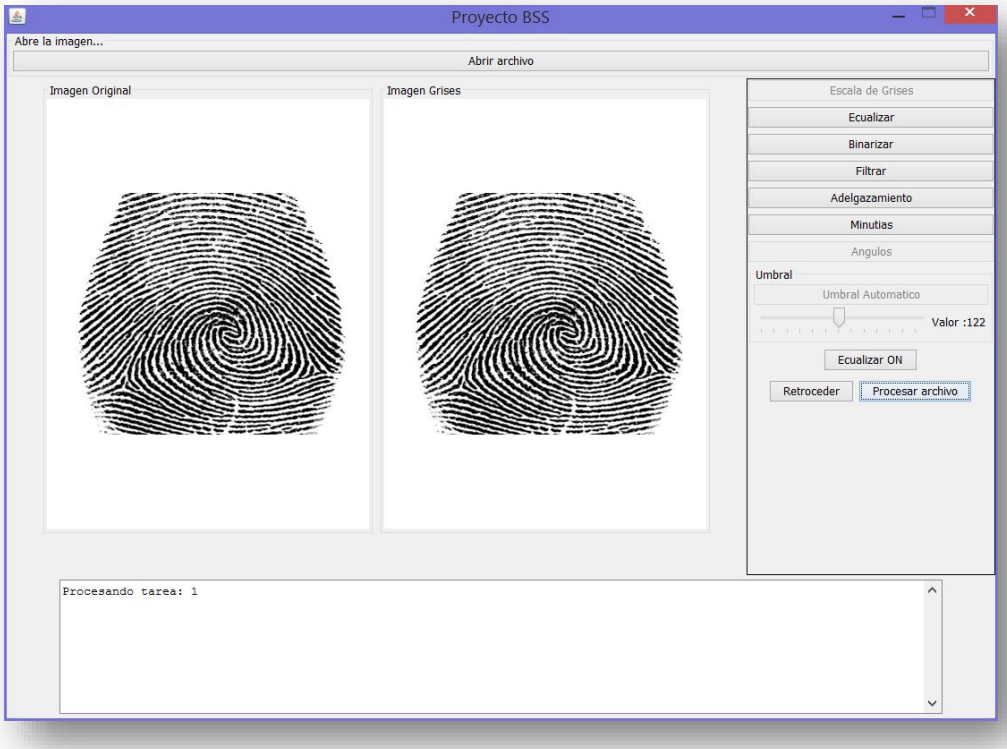
Damos al botón de Abrir archivo para seleccionar una imagen:



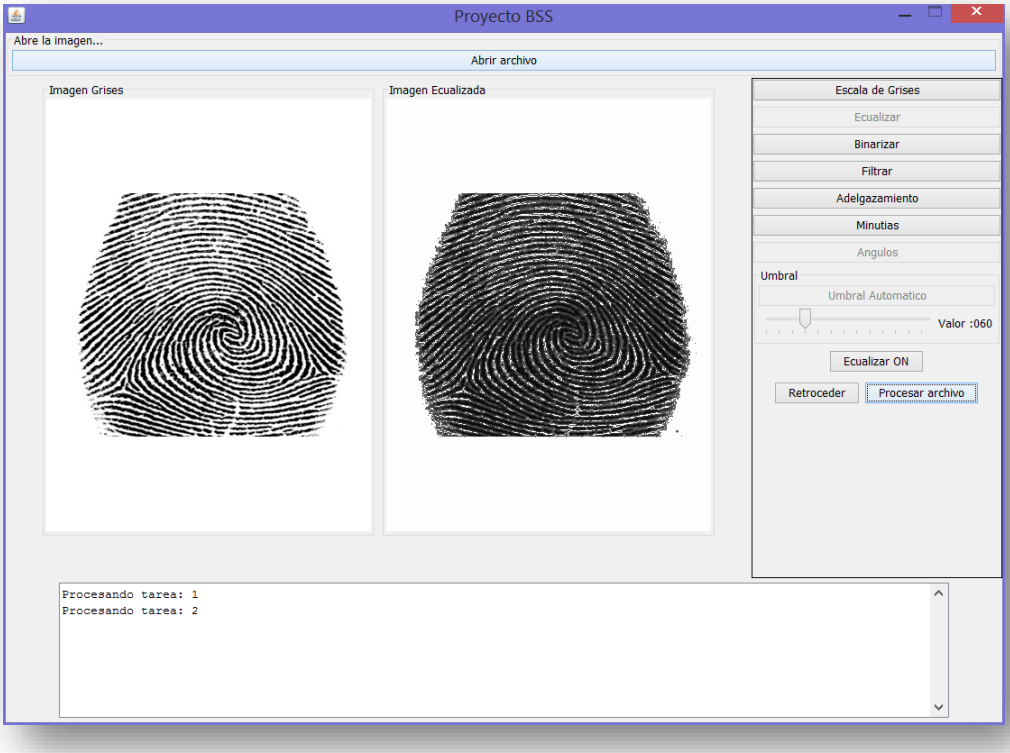
Obtenemos así la imagen abierta:



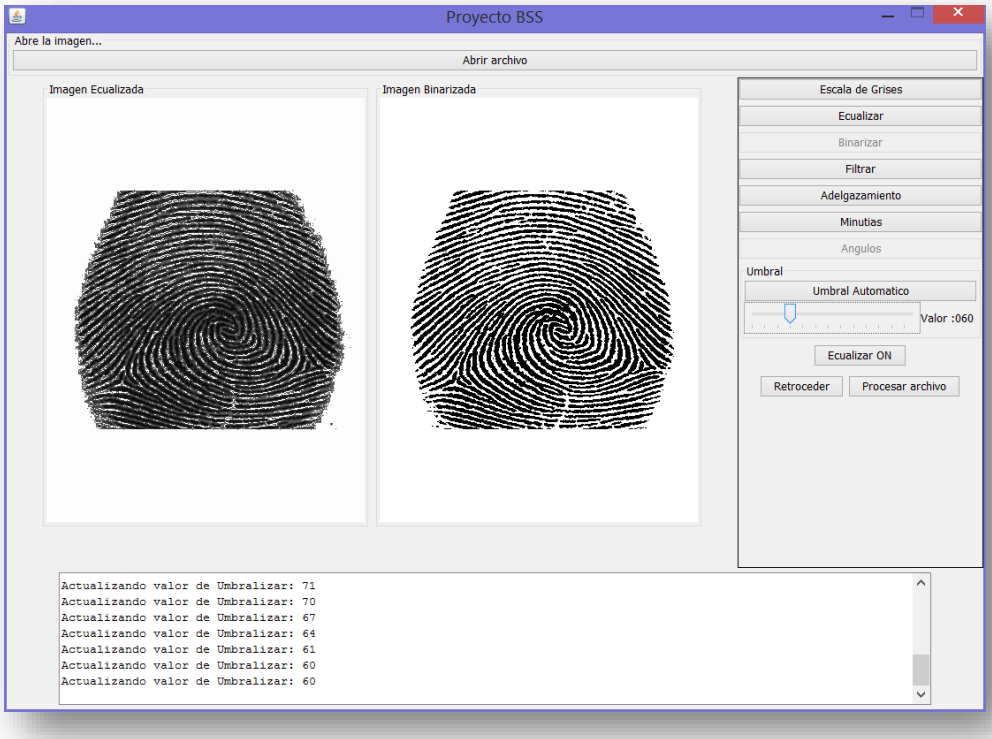
Damos a procesar archivo o a Escala de grises, obteniendo la a la izquierda la imagen original y a la derecha la imagen en gris:



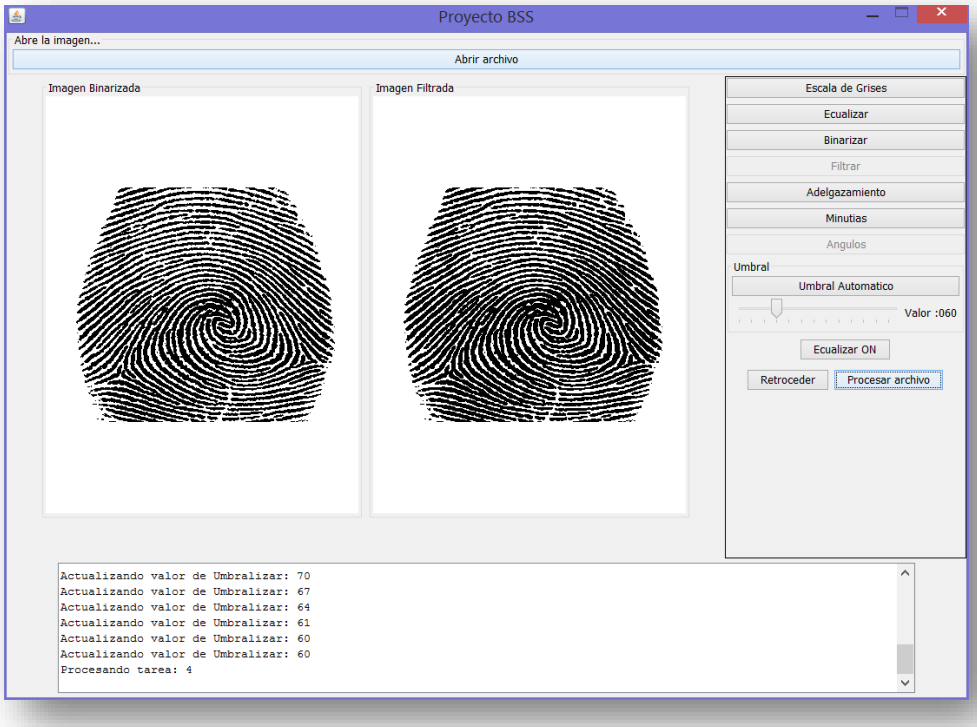
Al volver a dar a procesar archivo ecualizamos la imagen:



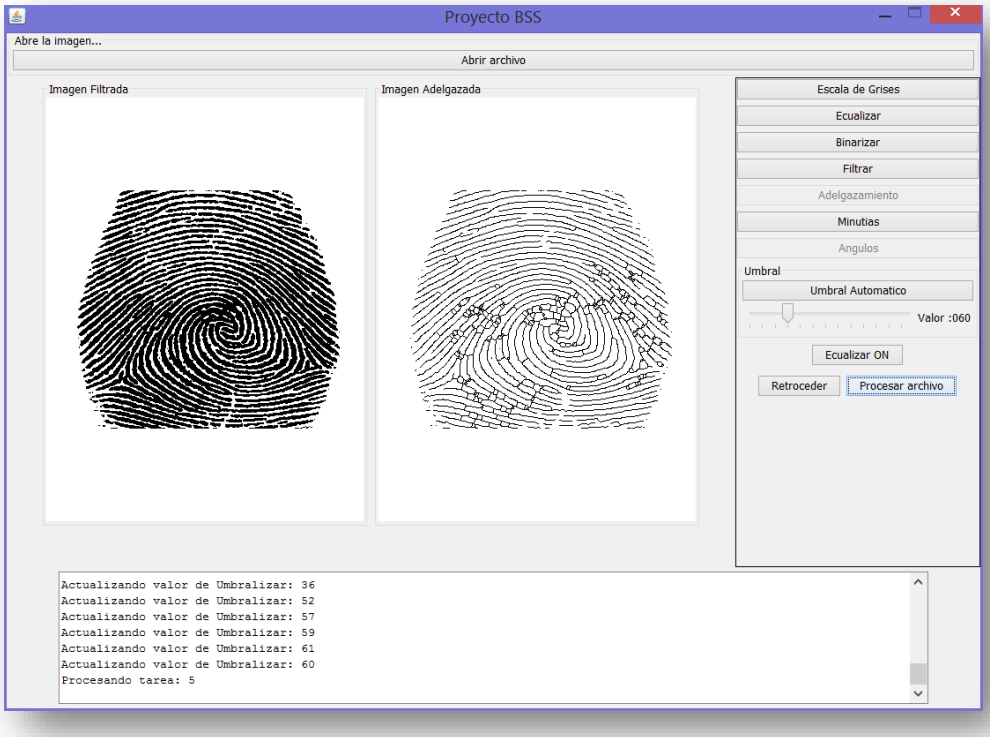
Ahora al dar a procesar archivo podemos cambiar el valor de la barra del umbral hasta encontrar una en la que la imagen se vea nítida, por ejemplo 60:



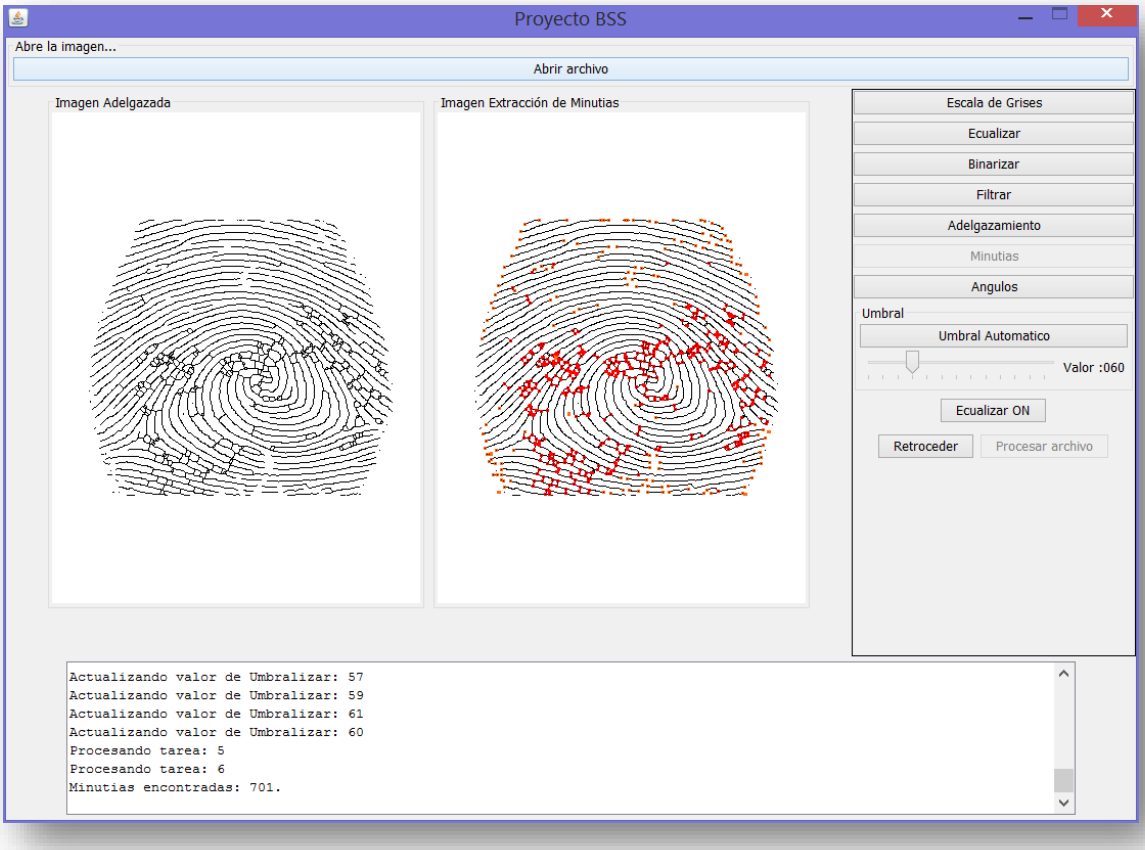
Al pasar a la siguiente tarea obtenemos la imagen filtrada, de manera que queda mas precisa.

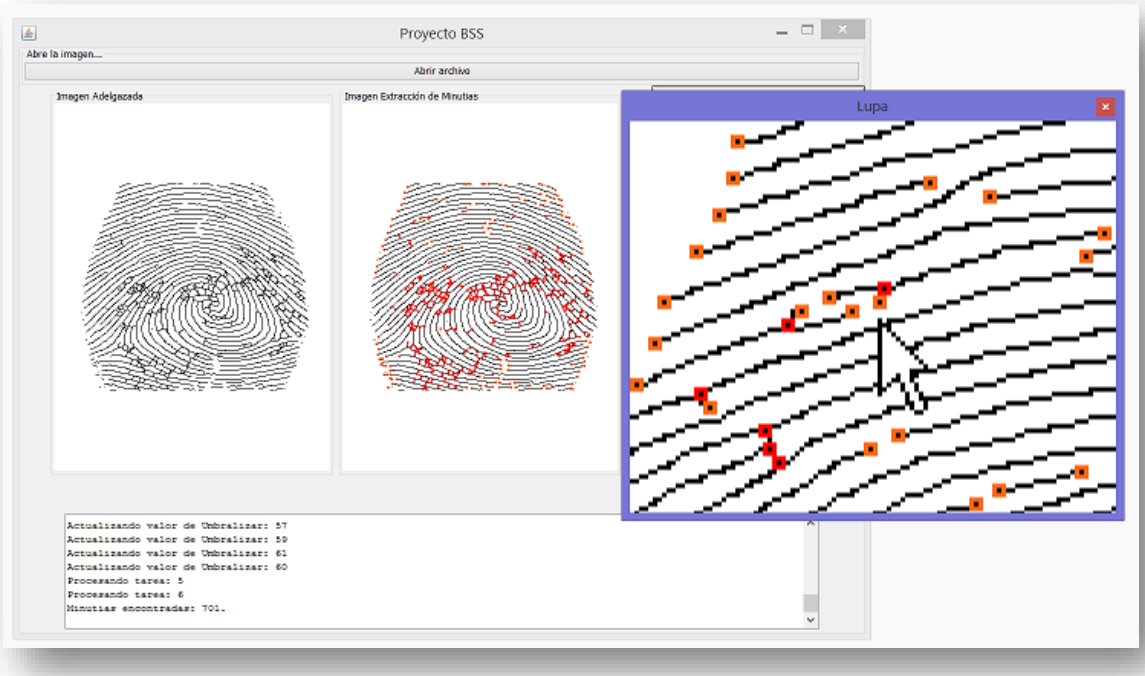


A continuación, obtenemos la imagen adelgazada, siempre usando el botón de procesar archivo.

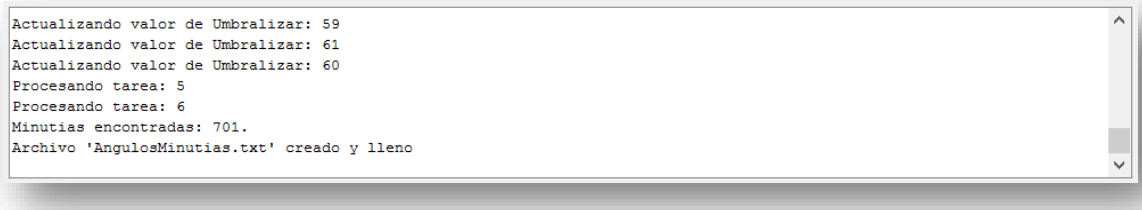


La ultima tarea a realizar por el procesar archivo es la extracción de minutias, estas se mostrarán con un círculo rojo alrededor, si nuestra pantalla es e alta resolución y no se ve bien podemos utilizar la lupa de Windows para acercar las minutias.

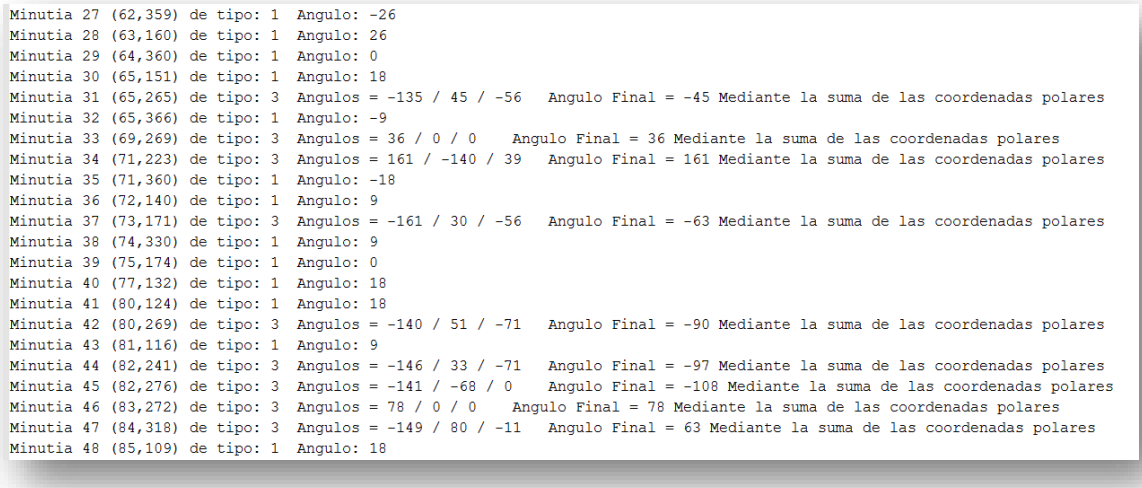




Para finalizar utilizaremos el botón de Ángulos que nos generará un fichero "AngulosMinutias.txt" en la carpeta donde tengamos el ejecutable



Esta es una muestra del interior del fichero:



Conclusiones

Este proyecto ha sido de gran utilidad académica, hemos aprendido a cómo tratar imágenes mediante el uso de matrices y como se resuelven los problemas del análisis biométrico de una huella dactilar.

Además, nos ha sido útil para aprender a manejar interfaces en java, con exposición de imágenes, que hasta ahora no lo habíamos usado.

Por último, ha sido una experiencia enriquecedora el hecho de poder hacer el trabajo en grupo.