Documentación Externa

Proyecto de Biometría

Biometría y Seguridad en Sistemas – Curso 2018/19

Alumnos:

- Juan Antonio Silva Luján
- Alberto Mangut Bustamante

UNIVERSIDAD DE EXTREMADURA

Índice de contenidos:

1. Introducción

La palabra Biometría tiene su significado del griego bios que significa vida y metron referente a medida, hoy en día se considera una ciencia que estudia características cuantitativas de los seres vivos, métodos que analizan rasgos humanos para identificar personas.

El uso de identificación por rasgos biométricos es muy útil a día de hoy ya que permite la identificación de individuos de una forma rápida, fácil y sencilla.

En este aspecto, nos centraremos en un tipo muy distintivo de biometría que depende de las características usadas para identificar a los individuos, biometría estática.

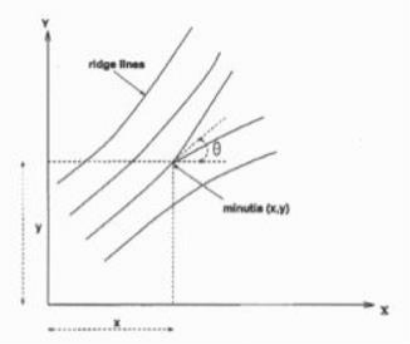
Existen numerosos tipos de biometría estática o física como la huella dactilar, retina, iris, ADN, cara, olor corporal, emisión térmica… pero en el caso que nos ocupa nos centraremos en la biometría basada en huella dactilar.

La biometría basada en huella dactilar es una cualidad exclusiva derivada de los primates, se forma en la 6º semana de vida y no varía a lo largo de toda la vida.

Toda huella dactilar se compone de:

- Rugosidades que forman salientes (crespas capilares).
- Depresiones (surcos interpapilares).

El sudor que emite el dedo se trata de un aceite que los surcos retienen, cuando se toca algo queda su negativo como residuo.



Las características de interés de una huella son la bifurcación y el punto final aunque cabe a destacar las minucias o minutia, los puntos de una huella dactilar donde una cresta termina o se divide en dos.

El emparejamiento se convierte en un problema de emparejamiento de un conjunto de puntos que son grafos o subgrados.

A continuación, se expondrán los pasos más relevantes del desarrollo del proyecto, es decir, los pasos a seguir para detectar correctamente la identidad de una huella dactilar.

2. Desarrollo del proyecto - Pasos

Antes de lograr la identificación de un individuo mediante su huella dactilar es necesario procesar la imagen ya que no puede ser procesada directamente, existen técnicas que experimentan con ciertas técnicas fundamentales. Los pasos a seguir son los siguientes:

Nuestro proyecto tiene una Interfaz de Usuario que actúa en base a los distintos eventos (pulsación del botón de acción) que invocará una función de la clase ProyectoBiometria.java desde la clase UI.java.

1) Obtención de la imagen original.

Se debe buscar una imagen de una huella dactilar con una calidad apropiada para el análisis, se examina en el directorio donde resida en nuestro equipo y se carga en el programa.

En nuestro proyecto, la clase UI (User Interface) reaccionará el evento de pulsar el botón examinar para invocar a la función del paquete Examinar y su clase. Esta ejecución cargará en el jLabel2 (panel de la izquierda) la imagen original. Esta carga está sujeta a la extensión de la imagen, en nuestro caso a las extensiones .JPG y .PNG

2) Conversión de imagen RGB a grises.

Se debe convertir la imagen RGB a una matriz de grises de 0 a 255. En nuestro proyecto, se invocará la función convertirImagenColorAImagenGris que recorre el alto y el ancho de la imagen obteniendo el valor RGB de cada iteración.
Se asignará el valor de R, G y B aplicando la operación lógica >> para mover los bits y a continuación la operación AND para cada uno de ellos. Por último, se realizará una operación final para obtener el gris de la media de los 3 colores primarios.

3) Realizar el histograma o ecualizado.

El histograma de una imagen representa la frecuencia relativa de ocurrencia de los niveles de gris en la imagen. Se define un mapeado de los niveles de gris p en nuevos niveles de gris q tal

que la distribución de niveles q es uniforme. Este mapeado expande el contraste de la imagen mejorando la detección de características.

En nuestro proyecto en la función ecualize se calcula la frecuencia relativa de ocurrencia de los distintos niveles de gris en la imagen, se construye la tabla LUT y se transforma la imagen utilizándola.

4) Convertir la imagen de grises a matriz de Blanco y Negro.

Para ello se aplica una función de umbralización (por defecto 127) y se pasa la imagen de escala de grises a una con forma binaria con valor 0 o 1, o blanco o negro.
En nuestro proyecto, en la función grisesAmatrizByN se recorre la imagen según su alto y ancho y se compara cada uno de los colores primarios R, G y B con el umbral existente en el jTextField1 con un valor por defecto de 127 que se asignará como variable por parámetros.

Si el valor del color primario es menor que el valor del umbral, se le asigna 255, en caso contrario 0. Es decir, si el color es relativamente oscuro, se convierte en negro absoluto, en caso de que el color sea relativamente claro, es convierte a blanco absoluto, de esta forma transformaremos toda la imagen a blanco y negro.

5) Filtrar la imagen

Es necesario realizar un filtrado de la imagen a partir de que halla sido transformada en blanco y negro etiquetando los puntos oscuros con un 0 y los puntos iluminados con un 1 ya que el ruido produce efectos en la imagen como contornos irregulares, pequeños huecos o esquinas perdidas con puntos aislados. Se utiliza una ventana de vecindad de 3x3 aplicando dos filtros distintos.

En nuestro proyecto se recorre en alto y ancho de la imagen en la clase Filtrado.java y se obtiene el color según su RGB, a continuación, se obtienen los valores de los vecinos y se aplica el filtro:

```
p = p + b * g * (d + e) + d * e * (b + g);

colorRGB = (p << 16) | (p << 8) | p;
```

6) Aplicar adelgazamiento – Algoritmo de Zhang-Suen

Mediante el uso del algoritmo de Zhang-Suen se obtiene un adelgazamiento de las líneas de la imagen en blanco y negro filtrada erosionando sucesivamente los pixeles del contorno hasta quedarnos solo con el "esqueleto".

7) Detección de Minucias

En nuestro proyecto, la obtención de las Minucias tiene lugar en la clase MinutiasDetector.java donde se recorre la imagen en todo su alto y ancho obteniendo una serie de condiciones booleanas según la ventana de vecindad de 3x3 de todos los vecinos existentes a cada pixel.

Cada vez que se cumpla cualquiera de las 12 condiciones especificadas en la sentencia condicional, se considera que esa posición de coordenadas (x, y) es una Minucia y por tanto, se asignará esa coordenada en la imagen y se listará en una estructura de datos para ser almacenada.

Se considerarán minucias tan solo cuando se trata de una bifurcación o un punto final de la huella.

Existe también la posibilidad de mostrarla en el entorno de desarrollo o exportarla a un fichero .TXT que decida el usuario mediante la clase Exportar.java del paquete Examinar.

8) Cálculo del ángulo de una minutia

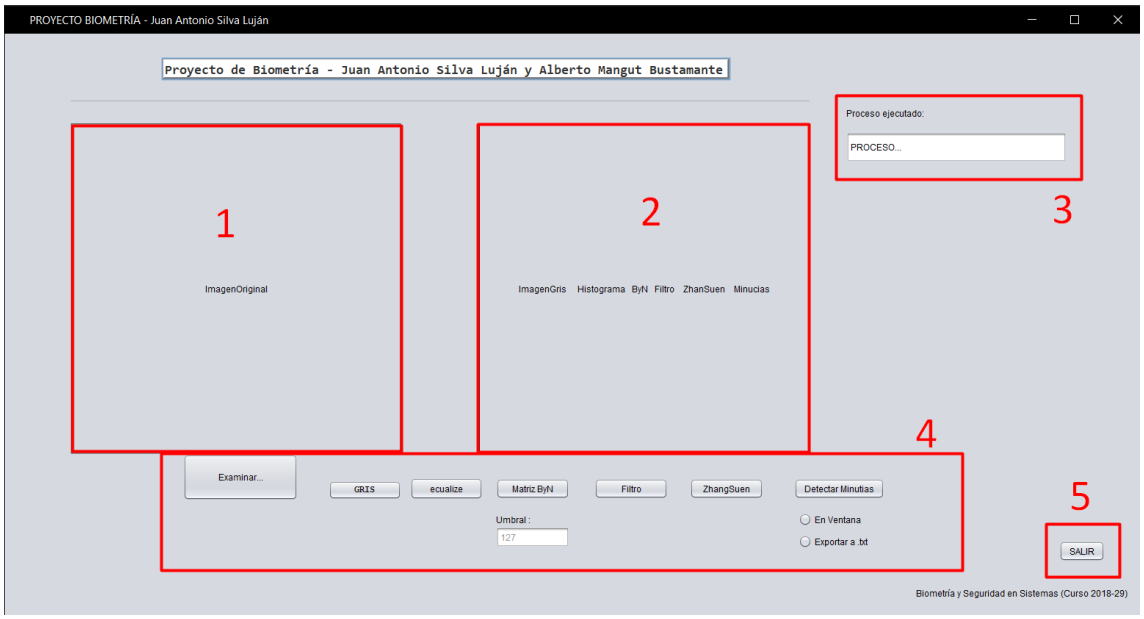
El ángulo se calcula recorriendo los 6 pixeles de la arista de la minucia a analizar, se inicia con la coordenada (x, y) y se avanza a sus siguientes vecinos aplicando el arcotangente de el resultado de la división del gradiente de X entre el de Y.

Paso extra) Conversión de Matriz a Imagen y viceversa

En nuestro proyecto tenemos implementados 2 algoritmos para poder transformar una imagen a una matriz y una matriz a una imagen, ha sido necesario recurrir a estos algoritmos en diversos momentos de la implementación ya que hemos procurado que todos los métodos sean de tipo BufferedImage y, por tanto que la clase principal invocadora, UI.java siempre retorne un BufferedImage para poder mostrarlo fácilmente en el panel derecho.

3. Manual de usuario – Uso del programa e Interfaz de Usuario

En la siguiente imagen se puede visualizar la interfaz de usuario con la que se puede ejecutar la práctica de la signatura y cada uno de sus componentes:



1: Panel izquierdo en el que se mostrará permanentemente la imagen original, la que se importa pulsando el botón Examinar…

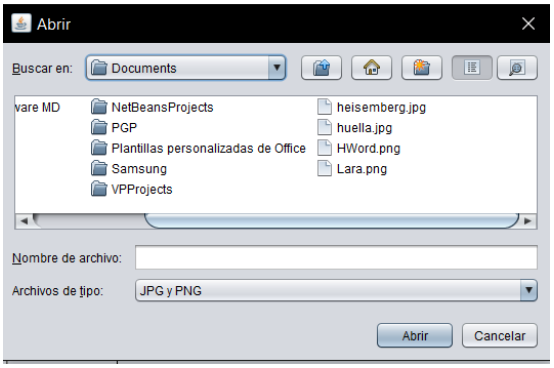
2: Panel derecho en el que se mostrarán las imágenes que se irán modificando tras la ejecución de cada uno de los botones inferiores.

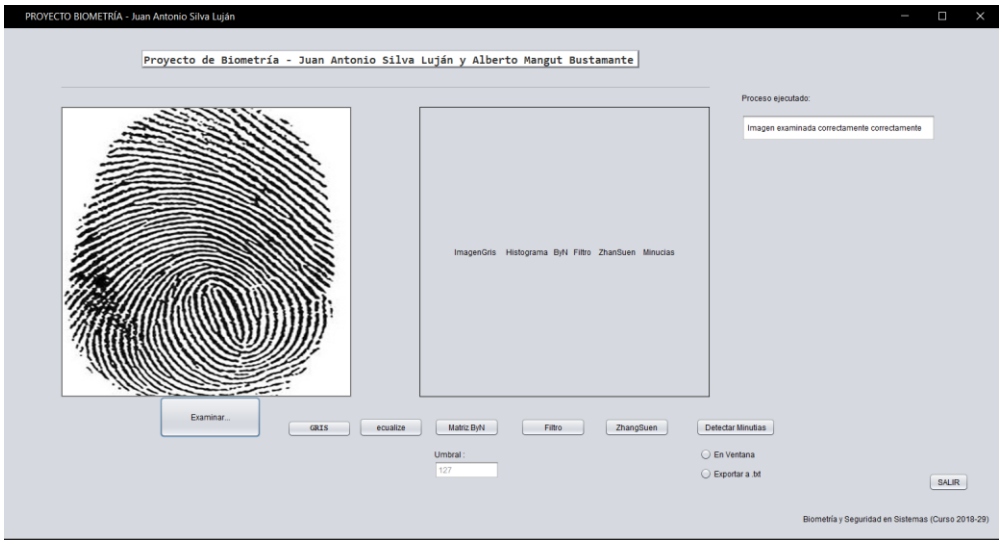
3: Panel de ejecución en el que se mostrará que proceso se está ejecutando para conocimiento del usuario.

4: Botones de ejecución de cada uno de los pasos a realizar detallados en el apartado anterior, en el último paso existen las opciones de mostrar las minucias en la ventana del IDE o exportarlas al TXT en la ubicación que deseemos.

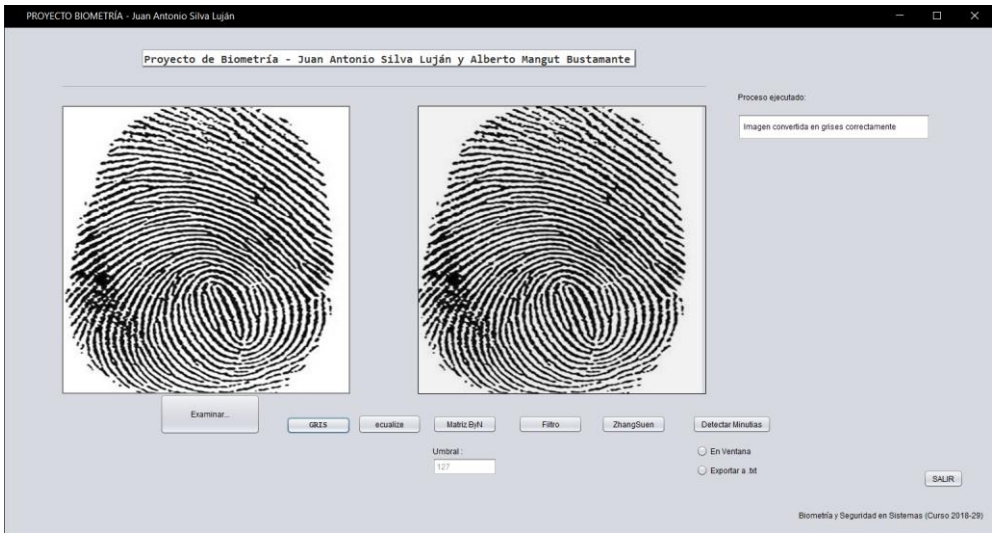
5: Salir del programa.

En primer lugar, podemos pulsar sobre el botón **[Examinar…]** donde se abrirá una ventana como la que se puede ver a la derecha, seleccionaremos la imagen de la huella a analizar, pulsaremos abrir y aparecerá en el panel de la izquierda de la interface principal.

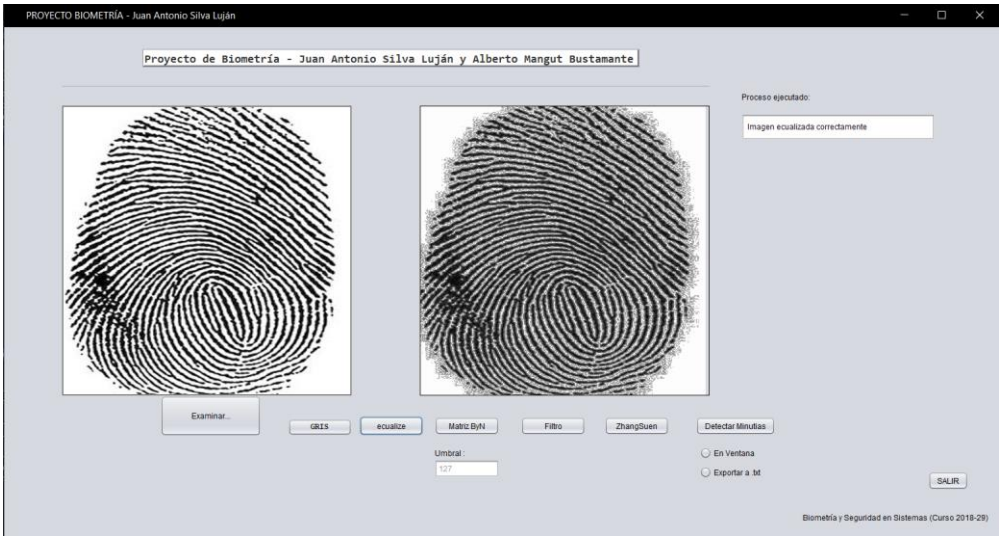




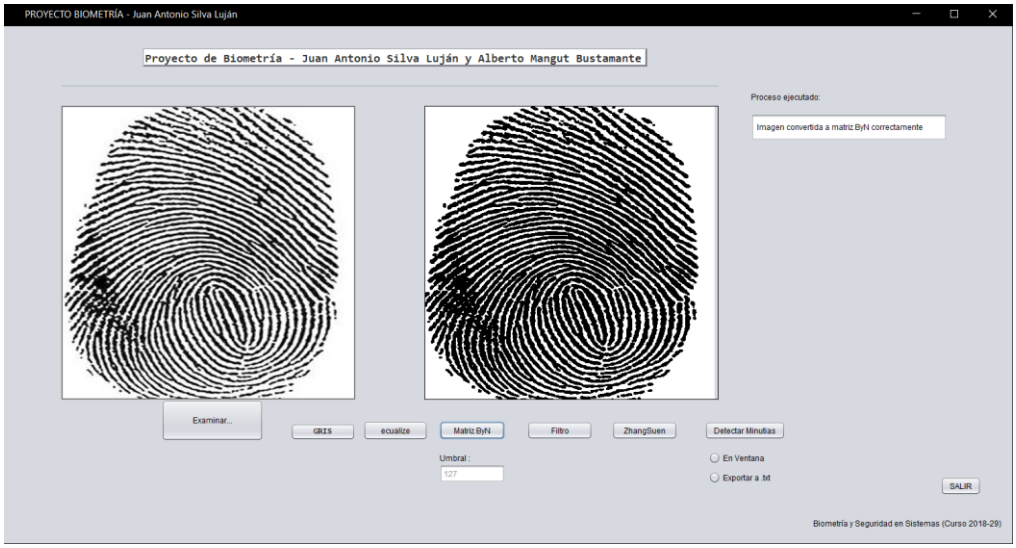
Cuando pulsemos en el botón **[GRIS]** aparecerá la imagen convertida a escala de grises en el panel de la derecha.



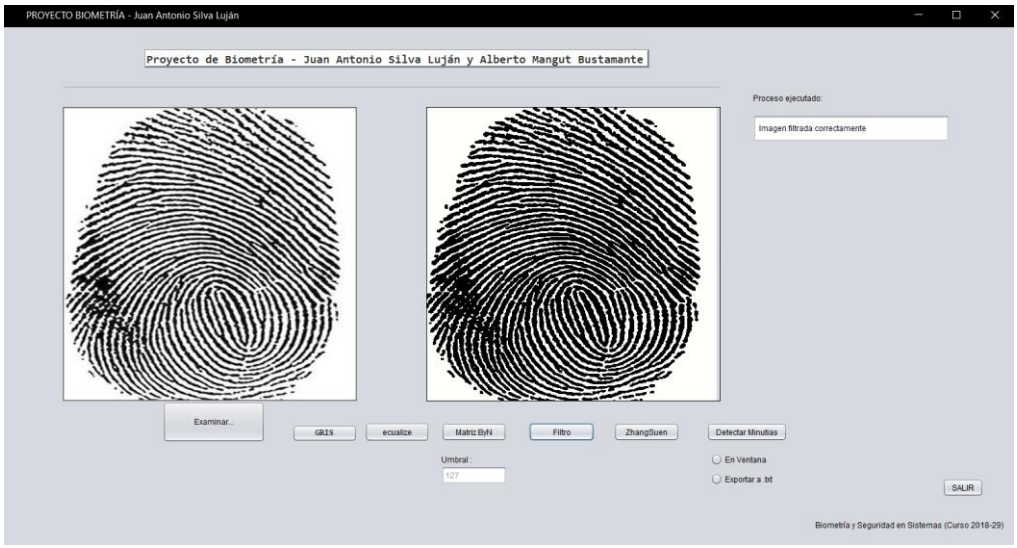
A continuación, pulsamos en el botón **[ecualize]** para realizar el histograma de la imagen.



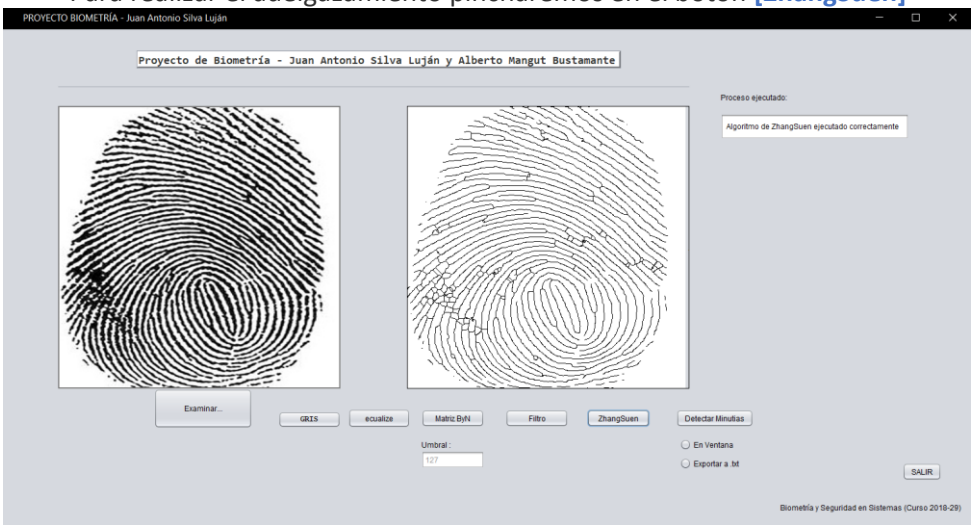
Con el **Umbral** por defecto a **127** (modificable) se pincha en el botón **[Matriz ByN].**



Y pulsando en el botón **[Filtro]** se aplica el filtrado a la imagen de la derecha.



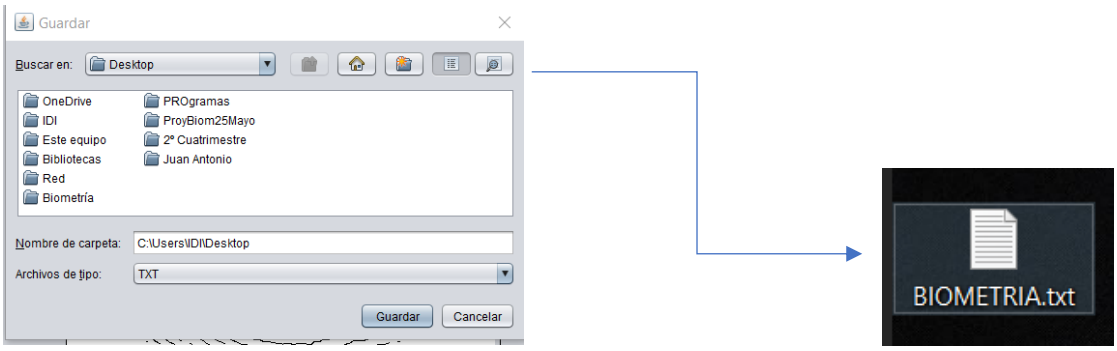
Para realizar el adelgazamiento pincharemos en el botón **[ZhangSuen]**



Por último, pincharemos en el botón **[Detectar Minutias],** podemos seleccionar como opción adicional:

- **En ventana:** Mostrará las minucias en la terminal del IDE.
- **Exportar a .txt:** Exportará las minucias a la raíz que desee el usuario con el nombre de `BIOMETRIA.txt`.

Vamos a exportar las minucias al escritorio tal como se ha detallado en la documentación:



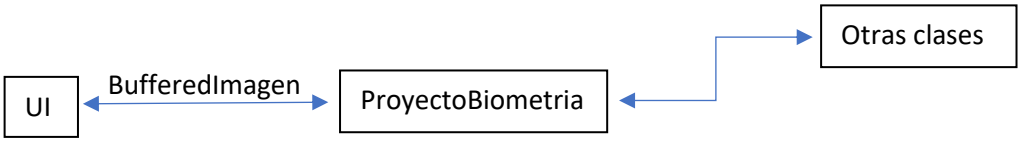
Si abrimos el archivo **BIOMETRIA.txt** se pueden observar todas las minucias detectadas separadas por un guión "-".

Por último, pinchamos en el botón **[Salir]** para cerrar el programa o bien sobre la X de la esquina superior derecha.

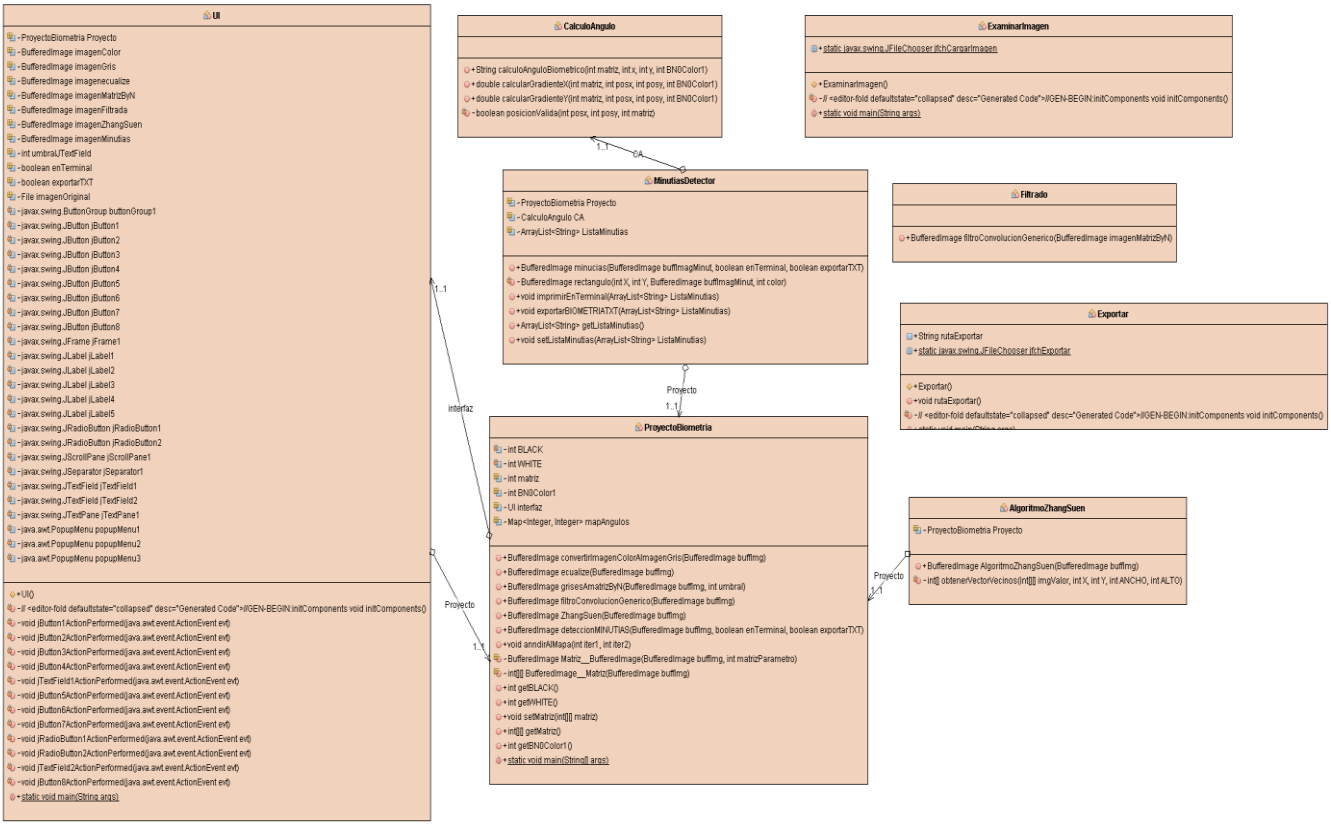
4. Manual del programador

La clase principal UI.java interactúa con las acciones o eventos del usuario, cuando pincha en un botón se ejecuta una función vinculada con dicho botón. Esta función invoca a su función asociada en la clase ProyectoBiometria.java, realiza la acción pertinente retornando siempre un dato de tipo BufferedImage, es decir, una imagen a la clase UI de nuevo que se encargará de mostrarla en el panel pertinente.

En algunos algoritmos, debido a su complejidad y longitud se usan sus propias clases como en el filtro, adelgazamiento o detección de minucias.



A continuación, se muestra un diagrama UML de la definición e implementación de las clases que actúan en el proyecto de biometría.



Con lo detallado anteriormente, se mostrarán los métodos implementados de la clase ProyectoBiometria.java, son los siguientes:

public BufferedImage convertirImagenColorAImagenGris(BufferedImage buffImg)

➔ Convierte una imagen a color en una imagen a escala de grises

public BufferedImage ecualize(BufferedImage buffImg)

➔ Realiza el histograma de la imagen gris

public BufferedImage grisesAmatrizByN(BufferedImage buffImg, int umbral)

➔ Convierte la imagen anterior en una solo de Blanco y Negro

public BufferedImage FILTRO(BufferedImage buffImg)

➔ Aplica el filtrado sobre la imagen anterior

public BufferedImage ZhangSuen(BufferedImage buffImg)

➔ Adelgada la imagen filtrada mediante el algoritmo de Zhang Suen

public BufferedImage deteccionMINUTIAS(BufferedImage buffImg, boolean enTerminal, boolean exportarTXT)

➔ Deteca las municias de la huella dactilar

BufferedImage Matriz__BufferedImage(BufferedImage buffImg, int matrizParametro[][])

➔ Transforma una MATRIZ a una IMAGEN

int[][] BufferedImage__Matriz(BufferedImage buffImg)

➔ Transforma una IMAGEN a una MATRIZ.

La documentación interna en estilo javadoc está presente en todo el proyecto, se adjuntará a la entrega de la práctica como anexo.

5. Problemas encontrados

Algoritmo / Paso	Descripción del problema
Conversión de MATRIZ a IMAGEN	Al comienzo del desarrollo de la práctica estábamos un poco perdidos y no nos era fácil familiarizarnos con este tipo de prácticas en java así como el tratamiento de la imagen en el lenguaje.
FILTRO	Hemos tenido muchos problemas con el filtro y ha sido necesario preguntar dudas en las sesiones de prácticas e incluso al final del todo ya que el filtrado se realiza con librerías internas de java sin usarlas como binarización.
ZhangSuen	Este algoritmo ha sido uno de los más complicados de implementar a pesar de que se ha usado código detallado en clase y en las indicaciones de la guía de desarrollo de la práctica y de rosetaCode.
Detección de Minucias	Ha sido lo último implementado en el desarrollo del proyecto y han surgido varios contratiempos ya que al inicio, en diversas huellas, llegaba a detectar cientos de Minucias, creíamos que era correcto pero tras las indicaciones del docente de la asignatura hemos ido decreciendo la cantidad de las minucias centrándonos en desarrollar implementaciones que detecten no más sino minucias más prometedoras.

6. Conclusiones

La práctica ha sido tan laboriosa como didáctica en diversos aspectos, hemos afrontado el problema de forma práctica tal como se indica en el apartado teórico de la asignatura, resulta bastante atractiva puesto que se ven los resultados que se ven obteniendo de forma inmediata.

Un detalle a tener en cuenta es el desarrollo de la interfaz gráfica de usuario, JAMÁS se ha implementado un proyecto con interfaz en toda la carrera por lo que ha sido interesante conocer como se diseña y como se implementan las asociaciones entre botones, eventos e invocaciones a métodos aunque halla resultado complicado en un inicio.

Hay pasos y algoritmos que han resultado más complejos de lo que a priori pensábamos pero pese a todas las dudas que han ido surgiendo, se han ido solucionando poco a poco en las sesiones prácticas y en las últimas clases de ayuda de la asignatura.

Aunque no apliquemos algunas mejores presentes en el tema 6 de la asignatura, la mayoría de los conceptos quedan bastante claros puesto que sería notablemente complicado tener en cuenta todos los detalles para mejorar algoritmos y tener en cuenta requisitos que mejoren la identificación de la huella dactilar.

Aclaraciones sobre el proyecto:

- Ha sido desarrollado en el IDE Netbeans - https://netbeans.org/
- Todas las imágenes de muestra tienen un tamaño de 450 x 450
- En la ejecución fuera de IDE, debe usarse la opción de exportar y no mostrar en terminal a no ser que se ejecute dentro del entorno.
- Documentación Interna completa.

La presente documentación interna así como el proyecto en su totalidad han sido desarrollados por los alumnos Juan Antonio Silva Luján y Alberto Mangut Bustamante para la asignatura de Biometría y Seguridad en Sistemas durante el curso académico 2018/19.