Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Componente para el post-procesamiento de imágenes de huellas dactilares

Autor

Ángel Noel Castro Tejas

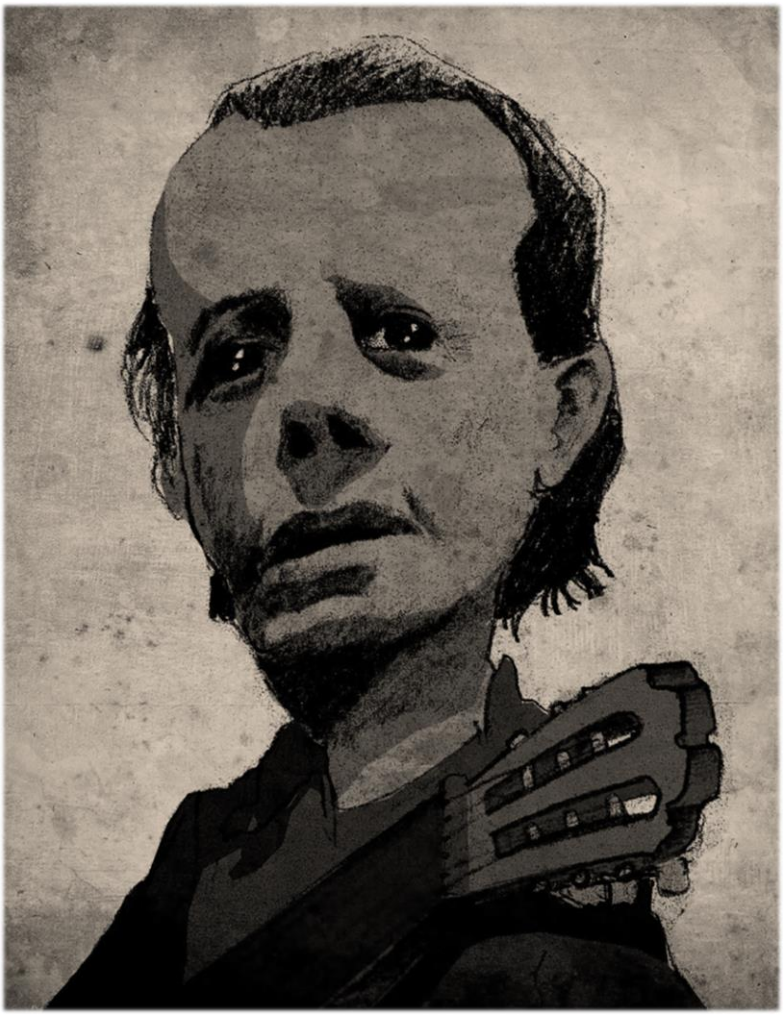
Tutor

Ing. Irina Brito Reyes

Cotutor

MSc. Damaris Cruz Amarán

La Habana, Junio de 2014



"Lo más terrible se aprende enseguida y lo más hermoso nos cuesta la vida"

Silvio Rodríguez

Declaración de la autoría

Declaro ser el único autor del presente trabajo de diploma titulado: **Componente para el post-procesamiento de imágenes de huellas dactilares** y autorizo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo el presente trabajo a los ____ días ____ del mes de _____ del año _____.

_____
Firma de Tutora
Ing. Irina Brito Reyes.

_____
Firma de Co-Tutora
Msc. Damaris Cruz Amarán.

_____
Firma del Autor
Angel Noel Castro Tejas

Tutor: Ing. Irina Brito Reyes.
Institución: Universidad de las Ciencias Informáticas
Correo electrónico: irinab@uci.cu

Co-Tutor: MSc. Damaris Cruz Amarán.
Institución: Universidad de las Ciencias Informáticas
Correo electrónico: damaris@uci.cu.

Agradecimientos

*A mis **padres** y **familia**,*

*A **Rayner**,*

*A mi pequeño amigo **Rubén** y a **Tomás,***

A todos los que me dieron su mano siempre que la necesite, y a los que no,

A todos los que están, y a los que no,

A aquellos que creyeron en mí, y a los que no,

En fin, a todos los que con su granito de arena, ayudaron a la elaboración de este trabajo.

.

Dedicatoria

*A mi **madre**, ejemplo de guerrera incansable.*

*A mi **padre**, **hermana** y **familia**, por su apoyo incondicional durante estos cinco años.*

.

Resumen

En la actualidad el reconocimiento de personas basado en rasgos característicos ha demostrado ser un método eficaz en recintos donde la seguridad es un tema de interés. Este proceso se efectúa con ayuda de los sistemas biométricos de identificación, que basan su funcionamiento en el reconocimiento de las características fisiológicas o conductuales de los seres humanos. En el departamento de Biometría, perteneciente al Centro de Identificación y Seguridad Digital de la Universidad de las Ciencias Informáticas, se lleva a cabo el proceso de desarrollo de un sistema que realice la verificación de individuos mediante huellas dactilares.

Para que este cumpla su propósito, es necesario realizar un proceso de filtrado de minucias con el objetivo de eliminar la mayor cantidad de falsas minucias mientras se conservan las de mayor valor identificativo. La presente investigación tiene como objetivo, desarrollar un componente para el post-procesamiento de imágenes de huellas dactilares.

Durante el desarrollo de la presente investigación quedó definido el empleo de las siguientes herramientas: como lenguaje de programación C#, como metodología de desarrollo XP y para el procesamiento de imágenes de huellas dactilares y realización de pruebas, el *framework Fingerprint Recognition* v2.2. Como principal aporte práctico del trabajo se destaca la implementación de los algoritmos necesarios para la discriminación de falsas minucias. Además de quedar demostrado que el algoritmo propuesto es competente y eficiente por lo que contribuye a sentar las bases para el desarrollo futuro de un sistema AFIS cubano.

Palabras claves: reconocimiento, sistemas biométricos, huellas dactilares, post-procesamiento, minucias.

Introducción

El reconocimiento biométrico hace referencia al uso de características o identificadores anatómicos o conductuales distintivos de una persona. Dentro de las características anatómicas se encuentran los patrones de la huella dactilar, la forma de la mano, el iris, entre otras, mientras que dentro de las conductuales se pueden citar la firma, el paso y el tecleo[1].

Las primeras aplicaciones de las técnicas biométricas tuvieron lugar dentro del ámbito legal, generalmente en el área forense. Sin embargo, en las últimas décadas ha surgido la necesidad de diseñar sistemas de alta seguridad capaces de identificar a los diferentes individuos a partir de sus rasgos biométricos. Para este proceso uno de los métodos más usados es el basado en huellas dactilares, debido su fácil adquisición y uso, fiabilidad y gran aceptación por parte de los usuarios[2].

Debido a los elevados costos de estos sistemas, a nuestro país se le dificulta tener acceso a estos, por lo que se ha trazado la meta de crear sus propias soluciones de modo de no quedar exenta en el tema. Tal es el caso del Sistema Automatizado de Identificación Dactilar (Biomesys AFIS) desarrollado por la empresa de Desarrollo de Aplicaciones, Tecnologías y Sistemas (DATYS) y el Centro de Aplicaciones de Tecnologías de Avanzada (CENATAV).

En la Universidad de las Ciencias Informáticas (UCI), se encuentra el Centro de Identificación y Seguridad Digital (CISED), el cual está compuesto por varias líneas de investigación y desarrollo, una de las cuáles especializa su trabajo en el área de técnicas biométricas, fomentando la creación de soluciones informáticas relacionadas con las mismas. Directamente, el equipo de trabajo relacionado con la investigación de tecnologías biométricas en la línea de huellas dactilares, también ha incursionado en el tema, obteniendo resultados positivos aunque todavía estos son algo incipientes.

El departamento de Biometría del Centro de Identificación y Seguridad Digital se encuentra actualmente implementando una aplicación para el proceso de extracción de características de la huella dactilar. Durante este proceso, la huella pasa por diferentes etapas, comenzando por la captura de la imagen de la huella por un sensor especializado. Seguidamente se aplica un conjunto de algoritmos matemáticos a esta para garantizar el mejoramiento de la calidad de la imagen, conformando la etapa de pre-procesamiento. Posteriormente son extraídas las minucias de la imagen adelgazada. Finalmente se realiza una etapa de post-procesamiento, donde se valida el conjunto de minucias extraídas usando diversas reglas, con el objetivo de eliminar falsas minucias.

La ausencia de esta última etapa trae consigo que bajo malas condiciones de la imagen puedan ser introducidas un gran número de falsas minucias o se cometan errores a la hora de decidir el tipo de minucias y su localización. Todo esto influye directamente en el rendimiento de los sistemas, aumentando las tasas de errores y los tiempos de respuesta.

Partiendo de la problemática anteriormente descrita, la presente investigación define el **problema a resolver** ¿Cómo eliminar las falsas minucias a partir un esqueleto de imagen de huella dactilar para el proceso de extracción de características implementado por la Línea de biometría del CISED?

Para dar solución al problema planteado surge el **objetivo general** de la investigación consistiendo en: Desarrollar un componente para la eliminación de falsas minucias a partir de un esqueleto de huella dactilar para el proceso de extracción de características implementado por la Línea de biometría del CISED.

Se define como **objeto de estudio**: El post-procesamiento de imágenes de huellas dactilares.

Para darle cumplimiento al objetivo general de la investigación fueron identificados los siguientes **objetivos específicos:**

1. Realizar estado del arte de algoritmos, metodologías y herramientas para el desarrollo del componente.
2. Diseñar un componente para la eliminación de falsas minucias a partir del esqueleto de una huella dactilar.
3. Implementar la solución propuesta.
4. Validar el resultado obtenido mediante la realización de pruebas al componente.

Idea a defender

Con la implementación de un componente para el post-procesamiento de la imagen de una huella dactilar para la Línea de biometría del CISED se realizará una correcta extracción de características de la huella dactilar, eliminando las falsas minucias introducidas durante el proceso de extracción.

Entre los **métodos científicos teóricos** utilizados en la investigación se encuentran:

- **Analítico-Sintético**: En el análisis de documentos para la captura y levantamiento de requisitos que permitan obtener los rasgos que caractericen el post-procesamiento de huellas dactilares así como las principales diferencias y similitudes de los algoritmos existentes en la actualidad.
- **Histórico-Lógico**: Se utiliza para el trabajo recopilatorio de información sobre los algoritmos

para el post-procesamiento y su evolución.

El presente documento está compuesto por tres capítulos, que incluyen todo lo relacionado con el trabajo investigativo, así como el análisis, diseño e implementación del sistema y las pruebas realizadas para validar la solución propuesta.

- **Capítulo 1: Fundamentación Teórica**. Se exponen los elementos teóricos que sustentan el problema científico y los objetivos del trabajo. Se realiza un análisis de las metodologías y herramientas de desarrollo disponibles para el desarrollo de la presente investigación.
- **Capítulo 2: Características del componente**. Se describen las principales características y funcionalidades del componente, se generan los artefactos de la ingeniería asociados a las fases de análisis y diseño.
- **Capítulo 3: Implementación y Pruebas**. Se realiza la implementación del componente, se describen los estándares de codificación utilizados y se ejecutan el conjunto de pruebas que validan la solución que se propone.

Capítulo 1. Fundamentación teórica

Introducción

En el presente capítulo se realiza un estudio de los principales algoritmos para el post-procesamiento de imágenes de huellas dactilares existentes, basado principalmente en el rendimiento de los mismos, así como conceptos asociados a estos para ayudar a una mejor comprensión del problema en cuestión. Se describen las principales características de las herramientas y tecnologías que se utilizarán en el desarrollo de la solución.

1.1. Conceptos asociados al dominio del problema

Sistemas biométricos

Son sistemas automatizados que realizan el reconocimiento a través de una característica física o conductual de una persona, basándose en un patrón conocido. La mayoría de sistemas funcionan de dos formas principales, una es el enrolamiento o captura de datos y la otra es la autenticación. La primera consiste en la adquisición de la información biométrica asociada a un usuario a través de un lector biométrico y posteriormente esta información es almacenada en una base de datos etiquetándola con la identidad del usuario. Mientras que en la autenticación de igual modo se extrae la información biométrica del usuario y es usada por el sistema para verificar la identidad de un usuario solicitando ser identificado o para identificar quien es el usuario.

La mayoría de los sistemas biométricos están compuestos por cuatro módulos, listados a continuación:

Módulo de lectores: Está destinado a la adquisición de la información biométrica asociada al usuario.

Módulo de extracción de características: Este módulo está encargado de extraer los valores de las características de cada rasgo biométrico.

Módulo de comparación: Es el responsable de comparar las características biométricas adquiridas con aquellas guardadas en una base de datos.

Módulo de toma de decisiones: Es donde se establece la identidad de un usuario o es aceptada o refutada una identidad solicitada. Esto se realiza basado en los resultados obtenidos por el módulo de cotejo.[3]

Actualmente estos sistemas son muy utilizados debido a la confiabilidad y rapidez con que una

persona puede ser identificada, brindando así un grado de seguridad elevado. Una contraseña o tarjeta de identificación, puede ser falsificada muy fácilmente, mientras que estos sistemas requieren de la presencia del individuo para realizar su verificación. Además de los bajos costos de administración, ya que su mantenimiento solo está enfocado al mantenimiento del lector y una persona encargada de mantener la base de datos actualizada.

Huella dactilar

Según el Diccionario de la Real Academia Española (DRAE)[4], la huella dactilar es la impresión que suele dejar la yema del dedo en un objeto al tocarlo, o la que se obtiene impregnándola previamente en una materia colorante. Esta definición es correcta, pero para un mejor entendimiento de que representa técnicamente la huella dactilar, es necesario conocer conceptos asociados a la morfología de la piel en la zona de los dedos de la mano.

Lofoscopia: Etimológicamente significa examen de las crestas papilares (*lophos:* crestas y *skopia:* examen). Más correctamente se considera a la Lofoscopia como la ciencia que estudia las crestas papilares con fines identificativos. [5]

Dactiloscopia: Etimológicamente proviene del griego (*daktilos:* dedos y *skopia:* examen), es por tanto la parte de la Lofoscopia que estudia las crestas papilares en la zona de la yemas de los dedos. Dado que la mayoría de los estudios sobre Lofoscopia se centran en esta zona, en el ámbito de la criminalística ambas palabras se utilizan como sinónimos.[5]

El Instituto Nacional de Estándares y Tecnologías (*NIST*, de sus siglas en inglés), define una huella dactilar como la imagen o impresión, parcial o total, de las crestas de fricción de cualquier dedo de la mano[6]. Otros autores convergen sobre un mismo concepto de huella dactilar, siendo el patrón de las crestas y valles en la superficie de la yema de los dedos[7-13].

De manera que se define como huella dactilar la impresión que produce el contacto de las crestas papilares de un dedo de la mano (generalmente el dedo pulgar o el índice) sobre una superficie. Siendo esta característica única y distintiva de una persona.

1.2. Objeto de estudio

En este epígrafe se abordarán temas relacionados con las características de las huellas dactilares y los procesos de extracción de características de una huella dactilar, haciendo énfasis en el post-procesamiento de la huella.

Características de las huellas dactilares

Las huellas dactilares tienen como particularidad el ser únicas para cada individuo, argumento que permite su uso como uno de los métodos de reconocimiento biométrico más utilizados a escala mundial.

Estas están constituidas por rugosidades que forman salientes y depresiones. Estas salientes son denominadas crestas papilares y las depresiones surcos inter-papilares o valles. Las crestas, en una imagen de una huella dactilar usualmente aparecen como una serie de líneas oscuras que representan los relieves, mientras que los valles entre estas crestas aparecen como espacio en blanco y están en bajo relieve[14] (Ver Figura 1).

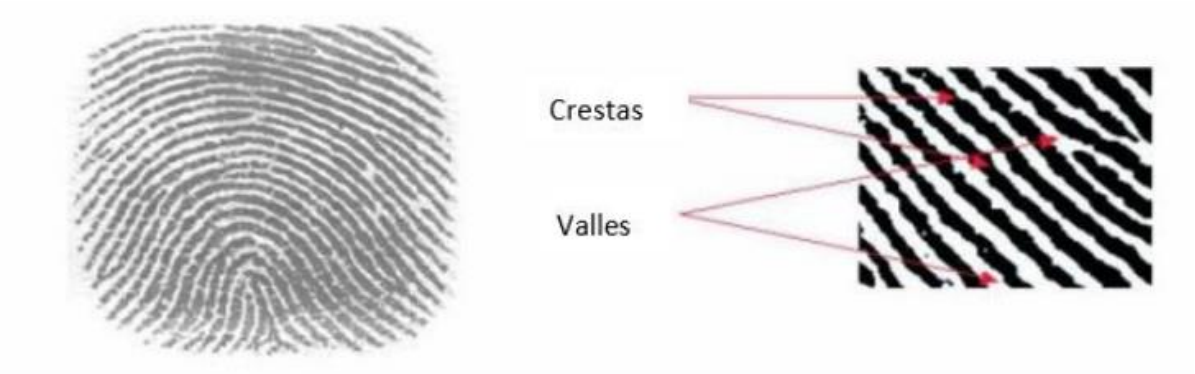


Figura. 1 Valles y crestas de una huella dactilar. Tomada de [15].

A la vez, poseen en su composición puntos singulares y únicos en cada una de ellas, denominados **minucias**. Estas se encuentran en el trazo de la crestas y no se encuentran distribuidas uniformemente en la huella. Durante el proceso de identificación o verificación existen varios tipos de minucias conocidas, como es el caso de las lagunas, las líneas independientes, el punto o isla, el espolón (*spur*, de su traducción al inglés), el cruce, la terminación y la bifurcación (Ver Figura 2), siendo estos dos últimos los tipos de minucias que más información aportan de una huella dactilar, debido a que como se puede observar, estas se encuentran incluidas como parte de los demás tipos de minucias.

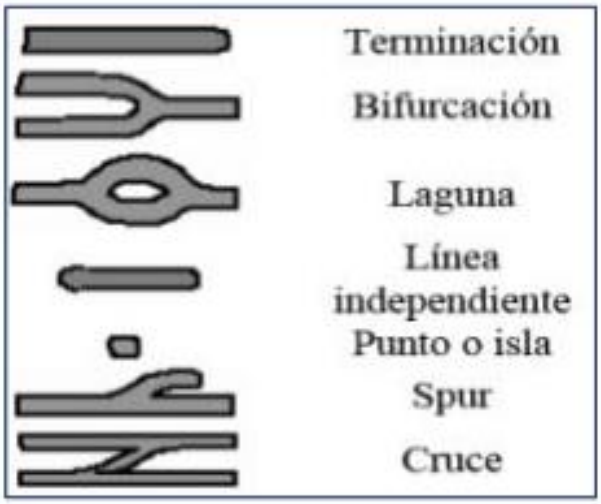


Figura. 2 Tipos de minucias. Tomada de [15].

Terminación: Se pueden encontrar donde naturalmente termina una cresta.

Bifurcación: Se pueden encontrar donde se divide una cresta en dos.

Niveles de representación de la huella dactilar

Cuando un patrón de huella dactilar es analizado a diferentes niveles, este presenta diferentes tipos de características:

- **Nivel 1**: A nivel global, el flujo de las líneas de crestas delinea patrones (Ej. Lazos, Deltas, etc.). Estos puntos particulares y los bosquejos de las líneas de crestas son de vital uso para la indexación y clasificación de las huellas dactilares, pero sus características distintivas no son suficientes para garantizar un cotejo de buena calidad. También forma parte de este nivel la forma externa de la huella dactilar y la orientación de la misma.
- **Nivel 2**: A nivel local, son encontradas las minucias. Este nivel de representación es uno de los más usados debido a su alta confiabilidad a la hora del cotejo de una huella dactilar.
- **Nivel 3**: A nivel muy refinado, los detalles internos de las crestas pueden ser detectados. Entre estos están el ancho, forma, curvatura y contornos de las crestas, pero el más distintivo de estos detalles son los poros sudoríficos. Sin embargo, extraer estas características es solamente factible en imágenes de huellas dactilares de alta definición[1].

Proceso de extracción de características en huellas dactilares

Pre-procesamiento

El pre-procesamiento de una huella dactilar es la etapa donde se aplica un conjunto de técnicas y algoritmos a la imagen capturada de la huella, con el objetivo de mejorar su calidad y de este modo garantizar una mejor extracción de las minucias. La calidad de la imagen se corresponde con la claridad de las estructuras de las crestas en la imagen de la huella dactilar; además, se considera que posee buena calidad cuando el contraste es alto y están bien definidas crestas y valles[16].

A grandes rasgos el pre-procesamiento de una huella dactilar es la ejecución de modo secuencial de los siguientes procesos: segmentación, normalización, cálculo de la orientación, binarización y adelgazamiento.

Segmentación

Existen dos regiones que describen una imagen de huella dactilar, la región localizada en el primer plano y otra en la región del fondo (Ver Figura. 3). La primera se corresponde con la zona de la huella dactilar que posee las crestas y los valles, y a su vez contiene los puntos característicos (minucias) de la imagen de la huella dactilar. Por otra parte, la región del fondo es aquella que queda fuera de los límites de la huella dactilar y es el área donde mayormente se introduce ruido en la imagen durante su adquisición. La esencia de la segmentación radica en reducir la carga asociada, asegurándose de que solamente se enfoquen en la región del primer plano[17].

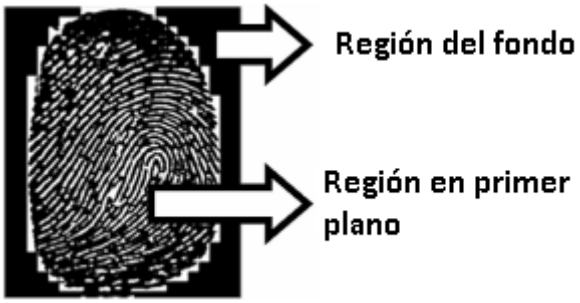


Figura. 3 Regiones de imagen de huella dactilar. Tomado de [17].

Normalización

La normalización se realiza sobre la imagen de la huella dactilar previamente segmentada con el principal objetivo de estandarizar el grado de variaciones en los valores de la imagen en escala de grises. Mediante este proceso de normalización los valores de escala de grises son llevados a un cierto rango lo suficientemente asequibles para mejorar el brillo y el contraste de la imagen[1].

Cálculo de la orientación

En cada huella dactilar, las crestas forman patrones que fluyen en diferentes direcciones. La dirección del flujo de una en particular durante un intervalo de pixeles, como se muestra en la figura constituye la orientación de la cresta (Ver Figura 4.)[17].

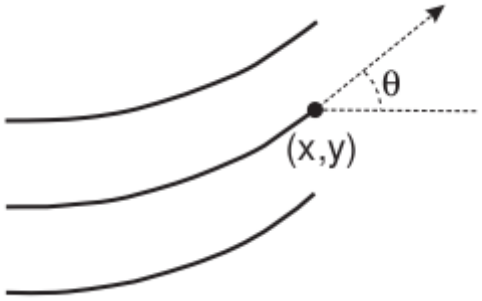


Figura. 4 Orientación de un pixel de cresta en una huella dactilar. Tomado de [17].

Binarización

Es el proceso que transforma la imagen a escalas de grises en una imagen binaria, donde cada pixel toma valor de 0 o 1. Varios algoritmos de extracción de minucias operan en imágenes binarias[2] donde hay sólo dos niveles de interés: los pixeles negros (con valor de 0) que representan las crestas y los blancos (con valor de 1) que representan los valles. Este proceso mejora el contraste entre las crestas y los valles en la imagen de la huella dactilar, y por consiguiente facilita la extracción de minucias (Ver Figura 5 a) [18].

Adelgazamiento

Es aplicado sobre la imagen binarizada y como resultado todas las crestas son adelgazadas al grosor de un pixel, haciendo que los puntos característicos de la huella dactilar se puedan identificar con más facilidad. Este proceso se implementa normalmente a través de operaciones morfológicas, que permiten reducir el ancho de las crestas. Después de esta etapa, la imagen está lista para la extracción de características (Ver Figura 5 b))[18].

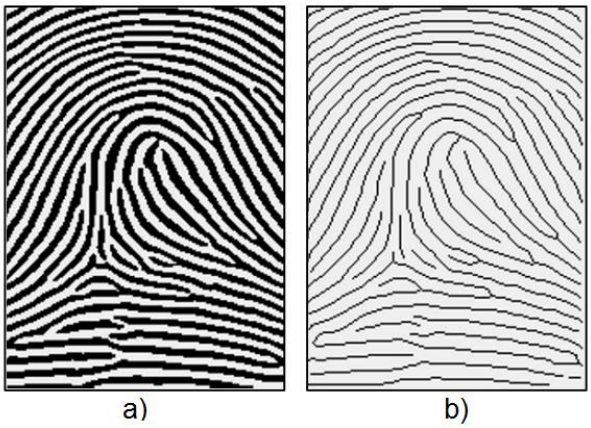


Figura. 5 a) Imagen binarizada b) Imagen adelgazada. Tomada de [1].

Extracción de minucias

Este es el proceso donde se extraen las minucias de una huella dactilar, trabajando sobre la imagen adelgazada. Los puntos de minucias son aquellos que tienen en su vecindad, un pixel con valor uno (terminación de cresta) o más de dos (bifurcación) (Ver Figura 6)[18].

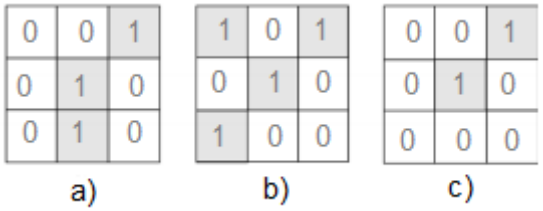


Figura. 6 a) Bloque sin minucias b) Bloque con minucia de bifurcación c) Bloque con minucia de terminación. Elaboración propia a partir de [18].

Post-procesamiento

El conjunto de puntos obtenidos generalmente contienen gran cantidad de falsas minucias (Ver Figura. 7). Esto puede ocurrir debido a regiones de la imagen de la huella dactilar con muy baja calidad, o debido a cicatrices o cortadas que no pudieron ser eliminadas en procesos anteriores. El problema de encontrar y eliminar las falsas minucias ha sido atacado por varios investigadores y con tal objetivo se han desarrollado diferentes algoritmos, algunos de los cuales se listan y explican a continuación.

Algoritmos para el post-procesamiento

A nivel internacional existen hoy en día varios estudios[19-24] realizados alrededor del tema del post-

procesamiento de imágenes de huellas dactilares, todos ellos enfocados en logar eliminar las minucias sin valor identificativo acerca de un individuo, manteniendo las que si lo tienen, ya que de ello depende que un cotejo de huella dactilar sea o no confiable.

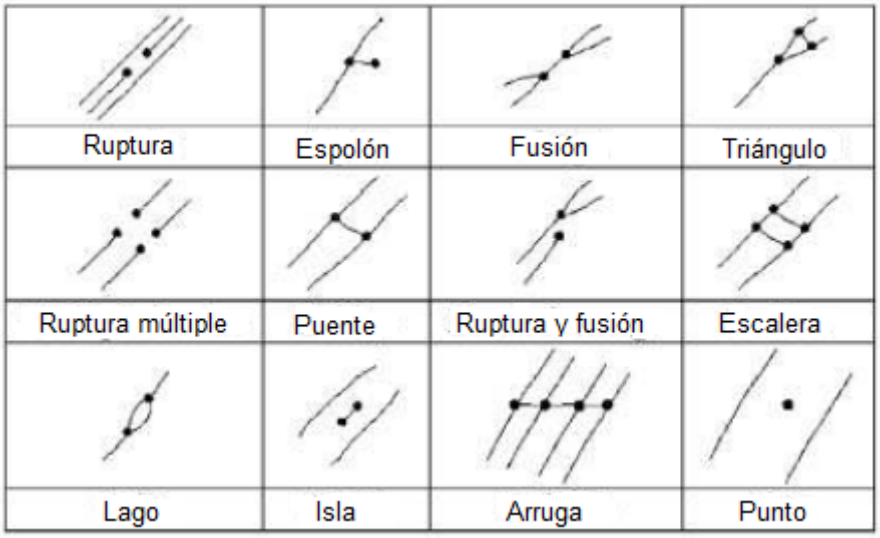


Figura. 7 Ejemplo de varios tipos de falsas minucias (puntos negros). Tomada de [2].

Ratha et al[23] proponen una fase de post-procesamiento para eliminar las falsas minucias basado en relaciones estructurales y espaciales de las minucias. Un ejemplo de ello sería el siguiente: dos minucias en una huella dactilar real no pueden encontrarse a una distancia muy corta una de otra. Las siguientes heurísticas son usadas para validar las minucias encontradas en el momento de la extracción:

Eliminación de ruptura en una cresta: Dos puntos de terminaciones con la misma orientación y dentro de un umbral de distancia (T_1 = 10) son eliminados.

Eliminación de espolón: Un punto de terminación el cual está conectado a un punto de bifurcación dentro de una distancia umbral (T_2 = 15) es eliminado.

Efectos de frontera: Las minucias detectadas dentro de un límite especificado de la frontera de las áreas exteriores de la huella son eliminadas.

El modulo entero de extracción de características propuesto por Ratha et al[23] fue probado en un ordenador SPARCstation 20 modelo 30, con un procesador de 200Mhz y una memoria RAM de 512MB. El mismo arroja tiempos de ejecución para todo el proceso de 32,2 segundos y particularmente para el proceso de extracción de minucias y post-procesado 1,8 segundos.

Para evaluar el rendimiento de su propuesta comparan las minucias extraídas con un conjunto de minucias seleccionadas por un experto en la misma huella. Definen $F_a = (f_a^1, f_a^2, \ldots, f_a^n)$ como un conjunto *N* de minucias detectadas por el algoritmo en una huella y a $F_g = (f_g^1, f_g^2, \ldots, f_g^n)$ como un conjunto *M* de minucias detectadas por un experto en la misma huella. Para calcular el Índice de Bondad (*IB*) (*GI* del inglés, *Goodness Index*) se necesitaron los siguientes conceptos:

Minucias emparejadas: Se cumple cuando una minucia f_a y una f_g se encuentran dentro de un área de tolerancia (Ver Figura. 8).

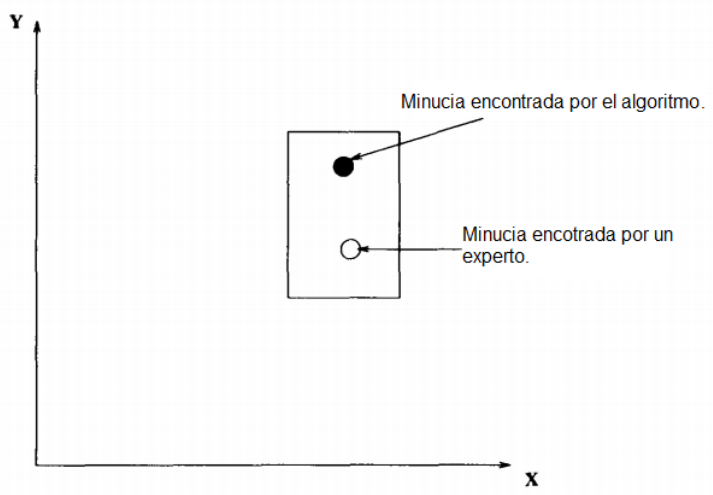


Figura. 8 Representación del Índice de Bondad según Ratha et al[23]. Tomada de [23].

Minucia faltante: Se cumple cuando una minucia f_a no se encuentra en el conjunto F_g. Es decir, la minucia no fue encontrada por el algoritmo.

Minucia espuria: Se cumple cuando una minucia del conjunto F_a no se encuentra en el conjunto F_g.

Luego de realizar el cálculo del IB se obtuvieron los siguientes resultados:

Tabla. 1 Índices de Bondad para el algoritmo de Ratha et al[23]. Elaboración propia a partir de resultados en [23].

ID de la huella	IB
hd_1	0,48
hd_2	0,475
hd_3	0,34
hd_4	0,285
hd_5	0,263
hd_6	0,135
hd_7	0,118

hd_8	0,11
hd_9	0,102
hd_10	0,1

Teniendo en cuenta que el IB quedó definido por el autor en un intervalo de $[-3,1]$. Se analizaron los valores obtenidos y se llegó a la conclusión de que el método propuesto es robusto y preciso. Se debe señalar que solo está enfocado a eliminar rupturas en las crestas, espolones y lo que el autor reconoce como efecto de la frontera, existiendo la posibilidad de no eliminar otro tipo de falsas minucias (Ver Figura. 7).

Kim et al[20] presentan un conjunto de algoritmos para el post-procesamiento de imágenes de huellas dactilares basado en la orientación y el flujo de las crestas, así como también la distancia y conectividad entre las minucias. Se define un orden para eliminar las falsas minucias, quedando de la siguiente forma:

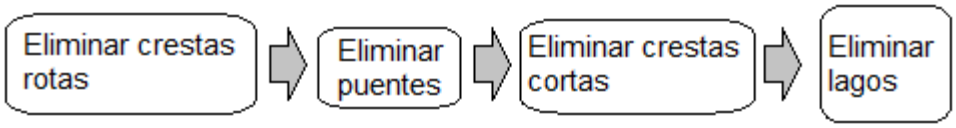


Figura. 9 Orden para eliminar falsas minucias. Elaboración propia a partir de [20].

Eliminar crestas rotas: Debido a cicatrices y a la poca presión del dedo en el dispositivo de captura, una cresta se puede dividir y crear dos puntos de terminación (Ver Figura. 10). De este modo se introducirían dos falsas minucias y son eliminadas de la siguiente forma:

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} < Dist_1 \quad (1)$$

La línea construida entre dos puntos de terminaciones, pertenecientes a dos crestas debe fluir en la misma dirección:

$$tan^{-1}\left(\frac{y_2 - y_1}{x_2 - x_1}\right) \cong \frac{1}{2}(OR_A + OR_B) \quad (2)$$

Eliminar puentes: El método usado para detectar dos falsas minucias de tipo bifurcación (Ver Figura. 10), conformando una estructura de puente es el siguiente:

1) Se siguen las tres crestas conectadas a una bifurcación (Punto A).
2) Si una de las crestas se encuentra con otra bifurcación (Punto B), se calcula la orientación de la cresta conectada por dos bifurcaciones (OR_{AB}) y la distancia de la misma ($Dist_{AB}$).
3) Si la distancia $Dist_{AB}$ es menor que un valor umbral ($Dist_2$) y la diferencia entre OR_{AB} y la

orientación promedio de las dos bifurcaciones (OR_A, OR_B) es mayor que un ángulo especificado (se definió $\frac{\pi}{4}$), entonces quedan identificadas las dos bifurcaciones como falsas minucias.

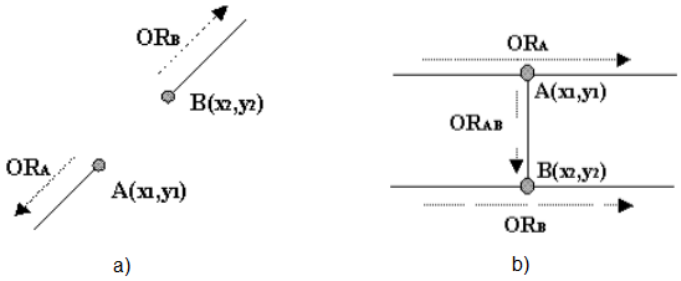


Figura. 10 Estructuras de falsas minucias: a) Cresta rota, b) Puente. Tomada de [20].

Eliminando crestas cortas: Comenzando de una terminación se sigue la cresta y si se encuentra una terminación o una bifurcación a una distancia ($Dist_3$) las dos minucias son consideradas falsas. También si una bifurcación se encuentra con otra son consideradas falsas minucias (Ver Figura. 11 c) d) y e)).

Eliminando lagos: Se siguen tres crestas conectadas formando una bifurcación. Si dos de las crestas se encuentran para formar otra bifurcación y las dos bifurcaciones están a una distancia ($Dist_4$), entonces las dos son consideradas falsas minucias (Ver Figura. 11 f)).

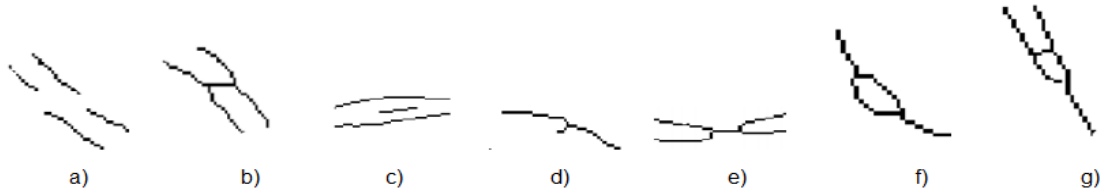


Figura. 11 Estructuras de falsas minucias según Kim et al. a) Ruptura de crestas, b) Puente, c), d) y e) Crestas cortas, f) Lago y g) Triángulo. Tomada de [20].

Para definir los umbrales el autor se basa en la propiedad elástica de los dedos. Esto se debe a que la diferencia entre las crestas cambia debido a diferentes presiones aplicadas por el dedo, en el momento de la captura de la huella. Para lidiar con este problema, el autor define los umbrales de forma adaptativa basado en la frecuencia de las crestas (Ver Tabla. 2).

Tabla. 2 Umbrales definidos por Kim et al[20] para el post-procesamiento. Elaboración propia a partir de [20].

Distancia	Descripción	Umbral

$Dist_1$	Cresta rota	$2/frecuencia$
$Dist_2$	Puente	$1{,}5/frecuencia$
$Dist_3$	Cresta corta	$1{,}7/frecuencia$
$Dist_4$	Lago	$2/frecuencia$

Para evaluar el rendimiento del post-procesamiento propuesto en (), se capturaron las imágenes de las huellas dactilares con un sensor manufacturado por Nitgen, con un tamaño de imagen de 248x292 y una resolución de 450dpi. La base de datos de huellas estuvo compuesta por 1000 imágenes de calidades diferentes, 10 capturas de 100 individuos diferentes.

Los términos usados para estas pruebas se describen de la siguiente manera:

Minucias verdaderas (MV): Minucias escogidas por un experto.

Minucias emparejadas (ME): Son minucias extraídas por el sistema que se encuentran dentro de **MV**.

Minucias falsas (MF): Minucias extraídas por el sistema que no se encuentran dentro de **MV**.

Minucias descartadas (MD): Minucias escogidas por el experto que no son extraídas por el sistema.

Minucias cambiadas (MC): Minucias que extrajo el sistema y se encuentran en **MV** pero con diferente tipo.

Tabla. 3 Rendimiento del algoritmo con respecto a la extracción de minucias. Elaboración propia a partir de [20].

	Método A	Método B
TMD (%)	9.8	12.3
TMC (%)	6.1	5.8
TMV (%)	84.1	81.9
TMF (%)	54.2	21.2

En la tabla anterior se muestra el rendimiento del algoritmo propuesto. El Método A indica la extracción de minucias sin post-procesamiento y el Método B indica los resultados obtenidos usando el algoritmo. Esta revela que la Tasa de Minucias Falsas disminuye en un 33%, mientras la Tasa de Minucias Verdaderas solo un 2.2%. En adición, para mostrar el efecto de usar su post-procesamiento en el momento de la comparación, los autores se apoyan en las tasas de Falsos Aceptados y Genuinos Aceptados, arrojando una considerable mejora en el rendimiento del sistema de comparación (Ver Figura. 12).

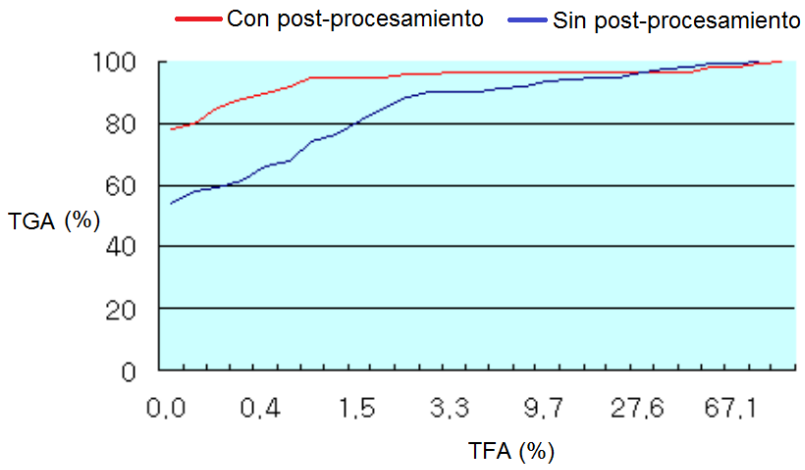


Figura. 12 Relación de las tasas de falsos aceptados y genuinos aceptados. Tomada de [20].

Lu et al[21] proponen un post-procesamiento efectivo y eficiente para la extracción de minucias falsas. Este algoritmo dedica un gran empeño en diferenciar confiablemente las falsas minucias de las verdaderas, además de hacer uso de la información del número asociado a cada cresta obtenida de la imagen original a escala de grises. Diseña y arregla varias técnicas de procesado correctamente y selecciona varios parámetros cuidadosamente.

Para la realización del post-procesamiento se trabaja con la lista de minucias una vez extraídas, usando información de la minucia y la imagen adelgazada. La imagen original a escala de grises se hace referencia a ella cuando se necesite. Otro parámetro de gran utilidad es el promedio de distancia de una cresta en pixeles.

El correcto funcionamiento de este algoritmo depende de que las minucias hayan sido extraídas con un algoritmo propuesto por los autores en otra publicación[25], dicho algoritmo trabaja directamente sobre la imagen a escala de grises. Por lo que no se tiene en cuenta para la presente investigación debido a que la misma está enfocada en la extracción de minucias de la imagen adelgazada y de usarse podría introducir una gran cantidad de minucias falsas.

Popovic et al[22] enfocan su trabajo en detectar y eliminar las falsas minucias en regiones de crestas con ruptura, a menudo definidas como arrugas. Desde que la mayoría de las técnicas de mejoramiento usan información acerca de campos de orientación, se ha tratado de analizar si y como esta información puede ser usada en algoritmos de filtrado de minucias. En una extensión de su trabajo anterior[26], esta vez incluyen información acerca de la posición de los puntos característicos

en su algoritmo. Aunque la mejora por filtrado direccional puede reconectar algunas rupturas de crestas originadas por arrugas estrechas, las más anchas permanecen. Entonces la información direccional multi escala es usada para detectar y eliminar falsas minucias en regiones de crestas rotas. El presente algoritmo está enfocado a tratar solamente las rupturas en las crestas, por lo que su implementación no resolvería el problema de esta investigación.

Zhao et al[24] proponen usar el valle de la huella dactilar en vez de las crestas para el proceso de extracción de minucias. Se usan diferentes pasos de pre-procesamiento en la imagen binarizada con el fin de eliminar falsas minucias como lagos y puntos, también para reducir el número de minucias del tipo isla, puentes y espolones en la imagen de esqueleto. Finalmente usando la propiedad de dualidad intrínseca de una huella dactilar (Ver Figura. 13), la cual versa que por cada terminación de cresta, se corresponde una bifurcación de valle y viceversa, con la única excepción de los puntos característicos delta y núcleo. Especialmente, definieron una estructura de punto-H para la eliminación de varias falsas minucias entre las que se encuentran los puentes, triángulos, escalera y arrugas todas juntas.

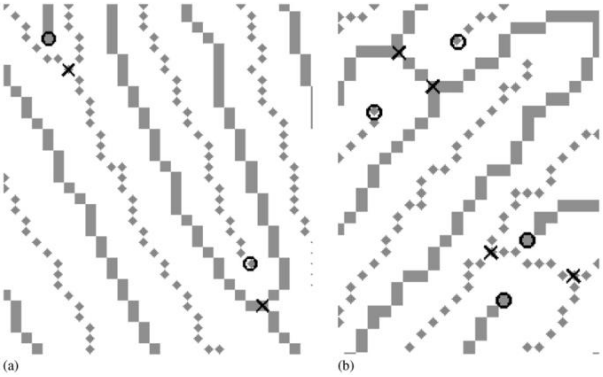


Figura. 13 Dualidad de las crestas y valles (Cuadros grandes representan los valles, los pequeños las crestas). Ejemplo de punto-H (¨o¨ son las terminaciones y ¨x¨ las bifurcaciones). Tomada de [24].

Se establece un orden específico para eliminar las falsas minucias, de modo que se optimicen los resultados en etapas posteriores y queda definido de la siguiente forma (Ver Figura 14):

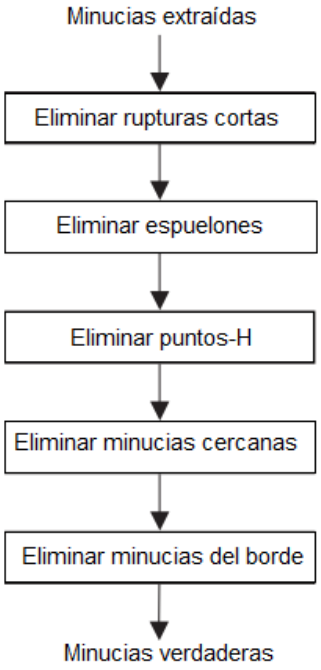


Figura. 14 Proceso de eliminación de falsas minucias. Tomada de [24].

En un primer momento se eliminan las rupturas cortas basados en que si dos terminaciones de crestas cumplen las siguientes condiciones serán eliminadas:

- Si la distancia entre dos terminaciones está por debajo de un umbral T_1.
- La diferencia entre los ángulos de orientación de dos terminaciones (Ang_1, Ang_2) está dentro de un intervalo $[\theta_1, \theta_2]$.
- La diferencia entre los ángulos de orientación de la línea que conecta las dos terminaciones (Ang_3) y alguno de los ángulos Ang_1 o Ang_2 está dentro un intervalo $[\theta_3, \theta_4]$.

Para calcular el ángulo de orientación de la terminación se busca el octavo vecino conectado a la misma. Durante la búsqueda, el algoritmo continúa incluso si se encontrara con un punto de bifurcación (Ver Figura. 15).

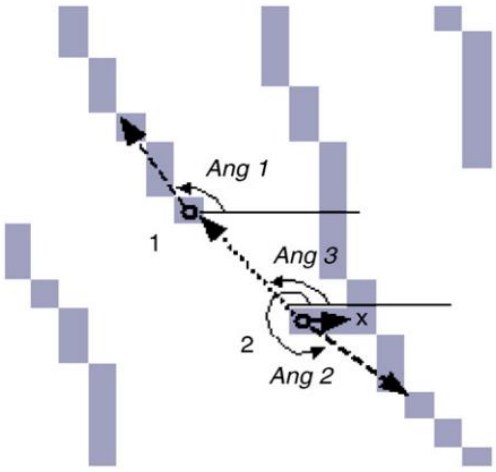


Figura. 15 Eliminación de rupturas cortas. Tomado de [24].

En un segundo momento se etiquetan los pixeles conectados en la imagen adelgazada. Si la distancia (d) entre un punto de bifurcación y uno de terminación está por debajo de un umbral T_2 y sus etiquetas son las mismas, se etiquetan una vez más los pixeles conectados, pero usando una ventana pequeña ($2d + 1\ x\ 2d + 1$) centrada en el punto de bifurcación o el de terminación. Si sus etiquetas siguen siendo las mismas, son eliminadas ambos puntos (Ver Figura. 16).

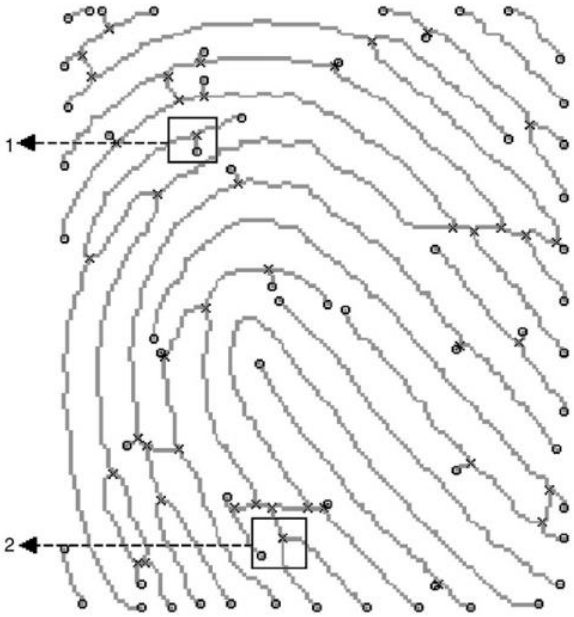


Figura. 16 Eliminación de espolones. 1) Los dos puntos conservan la misma etiqueta. 2) No se conservan las mismas etiquetas. Tomado de [24].

En un tercer momento son detectados y eliminados los Puntos-H. Si una estructura de puente en una cresta (o valle) de la imagen adelgazada y una ruptura en un valle (o cresta) cumplen con las siguientes condiciones, estos forman un Punto-H (Ver Figura. 17).

- El punto de intersección se encuentra entre dos terminaciones y dos bifurcaciones.
- La distancia entre el punto medio de la estructura de puente M_1 y el de la ruptura M_2 esta dentro de un umbral T_3.
- El ángulo de intersección θ entre la estructura de puente y la ruptura está dentro de un intervalo $[\theta_5, \theta_6\ ]$.

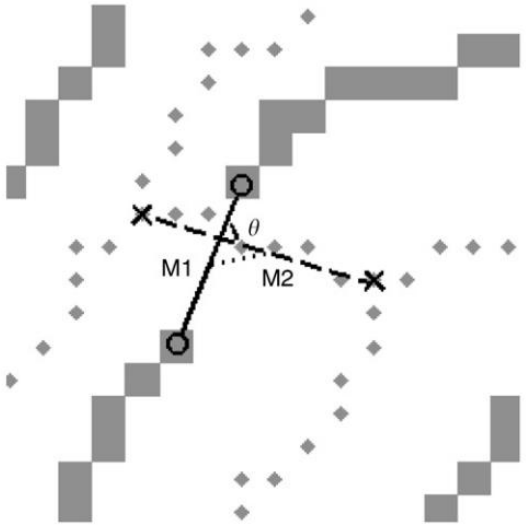


Figura. 17 Eliminación de estructuras de Puntos-H. Tomada de [24].

Finalmente son eliminadas aquellas minucias que están muy cerca una de otra, a una distancia umbral T_4 y todas las que se encuentren en la frontera de la imagen de la huella dactilar. Este algoritmo luego de realizado el post-procesamiento, es capaz de eliminar un elevado por ciento de las falsas minucias, mientras que preserva las verdaderas.

Para evaluar el rendimiento del algoritmo se realizaron experimentos con una base de datos de huellas adquiridas por un sensor óptico (*StarTek FM100*). Dicha base de datos estuvo compuesta por 6780 imágenes de huellas dactilares tomadas de 113 individuos. Las dimensiones de las imágenes fueron de $170\ x\ 180$, después de ser redimensionadas de las imágenes reales ($256\ x\ 256$) eliminando regiones con mala calidad pertenecientes al borde de la misma. Se utilizó el Índice de Bondad propuesto por Ratha et al[23] arrojando los resultados mostrados en la Tabla. 4.

Tabla. 4 Índice de Bondad para el algoritmo de Zhao et al[24]. Elaboración propia a partir de [24].

ID de la huella	IB
1	0,75
2	0,68
3	0,54
4	0,53
5	0,5
6	0,5
7	0,44
8	0,43
9	0,41
10	0,18

En una etapa final se hicieron pruebas del algoritmo sobre la base de datos DB1 de la FVC2002, la cual contiene 880 imágenes de huellas dactilares ($388\ x\ 374$) de 110 individuos. Se tuvo en cuenta el proceso completo, incluyendo la comparación para obtener las tasas de falsos y genuinos aceptados. Se puede concluir que después de obtener estas tasas sin usar el post-procesamiento, usando el algoritmo propuesto por Kim et al[20] y usando el de Zhao et al[24], este último mejora considerablemente el proceso de comparación (Ver Figura. 18).

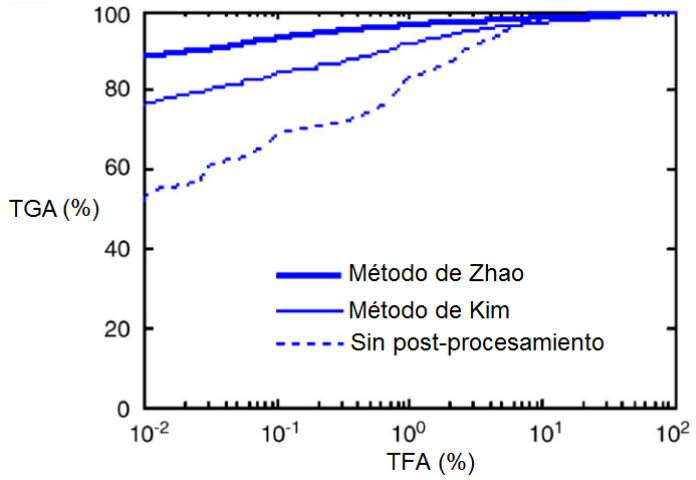


Figura. 18 Relación de las tasas de falsos aceptados y genuinos aceptados. Tomada de [24].

Bansal et al[19] proponen un algoritmo de extracción de minucias el cual usa la Transformada de Aciertos y Fallas (TAF, *Hit or Miss Transform* de su traducción del inglés). El autor señala que las regiones separadas o mal conectadas en la imagen binarizada pueden significar la aparición de un

gran número de falsas minucias en la imagen adelgazada. Por lo que se aplican operadores morfológicos a la imagen binarizada para eliminar espolones, estructuras de puentes, lagos e islas.

Para la extracción de minucias es usada la transformada de aciertos o fallos. Con este objetivo se desarrollaron varios elementos estructurados con la forma de minucias presentes en una imagen de huella dactilar para ser comparados (Ver Figura. 19).

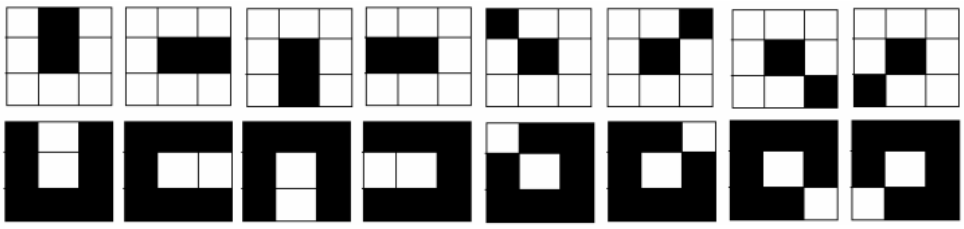


Figura. 19 Estructuras usadas por la TAF para detectar minucias de tipo terminación. Tomada de [19]:

Este algoritmo centra su atención en la etapa del pre-procesamiento, aplicando operadores morfológicos sobre la imagen. Los autores afirman que el factor clave para el correcto funcionamiento está en la realización de un buen pre-procesado y en la calidad de la imagen. Por ende la fase de post-procesado es orientada a validar reglas simples que eliminen falsas minucias que hayan permanecido.

Se realizaron experimentos sobre una base de datos de huellas la cual está compuesta por impresiones de 40 dedos y 8 de cada uno. Se extrajeron las minucias en 4 etapas diferentes como se muestra a continuación:

- Imágenes originales.
- Imágenes mejoradas y adelgazadas.
- Imágenes adelgazadas pre-procesadas.
- Luego del post-procesado.

También se usaron de medidas cuantitativas para la medida del rendimiento del algoritmo, estas son la Sensibilidad y la Especificidad. Estas indican la capacidad de detectar minucias genuinas y eliminar falsas minucias respectivamente (Ver Tabla. 5), teniendo en cuenta las minucias fieles (minucias extraídas por un experto de la imagen original de la huella dactilar).

Tabla. 5 Resultados de las variables Sensibilidad y Especificidad definidas por Bansal. Tomada de [19].

	Antes de post-procesado			Luego de post-procesado			Sensibilidad	Especificidad
Imagen	Total	Descartadas	Falsas	Total	Descartadas	Falsas	%	%
I1	215	1	157	62	1	4	98,3	93,2

I2	61	1	40	23	1	1	95,7	95,7
I3	64	1	43	22	1	0	95,7	100
I4	76	2	45	31	2	1	93,6	96,9

Los resultados arrojados por el algoritmo muestran una alta precisión. Extraer las minucias usando la Transformada de Aciertos y Fallas permite obtener un menor número de falsas minucias, lo cual contribuye a la disminución de tiempos dedicados al tratamiento de estas (Otros resultados son mostrados en el Anexo 8).

Luego de haber realizado un análisis de los métodos anteriormente citados se arribó a las siguientes conclusiones. Teniendo como base las principales características de cada método y la similitud de estas con el proceso en particular a usar, la forma en que son validados los resultados experimentales y el nivel científico de cada publicación.

Se definieron tres variables para realizar la evaluación, las cuales se describen a continuación:

Similitud: está relacionada con la semejanza del pre-procesamiento y extracción de minucias entre el método propuesto por cada autor y el método a usar. De este modo se garantiza también, una semejanza entre resultados. Esta variable tiene tres clasificadores (Alta, Media y Baja).

Un ejemplo de cómo se realizó el proceso de calificación para esta variable puede verse en la asignación de una evaluación Baja al método de Bansal2010[19]. En este se realiza el pre-procesamiento basado en operadores morfológicos, así como la extracción de minucias con la Transformada de Aciertos y Fallos.

Validación: está relacionada con la forma en que son validados los resultados, es decir ¿De qué modo se realizan las validaciones del algoritmo?, ¿Qué peso tienen estas validaciones?, ¿Qué bases de datos son usadas para los experimentos? Esta variable tiene tres clasificadores (Alta, Media y Baja).

Un ejemplo de cómo se realizó el proceso de calificación para esta variable puede verse en la asignación de una evaluación Alta al método de Bansal2010[19]. Los resultados son validados usando una base de datos de la FVC2002, mostrando los resultados de la extracción de minucias en diferentes etapas de la extracción de características. Por otro lado usa dos variables (Sensibilidad y Especificidad) para medir la precisión del método propuesto, además de varias tablas donde expone el Índice de Bondad y cantidad de minucias de ambos tipos antes y después del post-procesamiento.

Nivel Científico: está relacionado con los factores de impacto de las revistas que publican los

artículos, así como los publicadores. También se tuvo en cuenta la cantidad de citas y referencias. Esta variable se calcula empleando la fórmula siguiente:

$$NC = CC(\%) + FI + CR(\%)$$

Donde $CC(\%)$ se refiere a la cantidad de citas del artículo en porciento, FI al factor de impacto de la revista o el publicador y $CR(\%)$ a la cantidad de referencias del artículo en porciento (Los cálculos asociados a esta esta variable se realizan en el Anexo 7).

A continuación se establece una comparación entre los métodos anteriores usando las variables anteriormente definidas (Ver Tabla. 6).

Tabla. 6 Tabla comparativa de los métodos investigados. Elaboración propia.

Autor	Similitud	Validación	Nivel Científico
Ratha1995[23]	Alta	Media	8.362
Kim2001[20]	Media	Media	1.28
Lu2002[21]	Baja	Media	1.23
Popovic2007[22]	Alta	Baja	0.641
Zhao2007[24]	Alta	Alta	3.202
Bansal2010[19]	Baja	Alta	1.486

De la comparación anteriormente realizada se concluye implementar el componente de post-procesamiento usando los métodos propuestos por Zhao et al[24] para eliminar minucias de tipo rupturas cortas, espolones y minucias cercanas (lagos, islas, puntos y fusiones). De forma auxiliar, se emplea parte de la solución de Ratha et al[23] para eliminar las minucias asociadas a la frontera y las rupturas cortas, quedando así conformado el componente propuesto.

1.3. Ambiente de desarrollo

En este apartado se realiza un análisis relacionado con varias metodologías, herramientas y tecnologías existentes, y se establece la definición de aquellas que en conjunto, conforman el

ambiente de desarrollo.

1.3.1.Metodología de desarrollo de software

La selección de una metodología adecuada para el desarrollo de software es un factor importante en el éxito de un proyecto. No existe una metodología mejor que otra, sino que algunas se ajustan mejor a las características y necesidades específicas de los proyectos.

Las metodologías de desarrollo se dividen en dos grandes grupos, las tradicionales y las ágiles. Las primeras enfatizan en el uso exhaustivo de documentación durante todo el ciclo de vida del proyecto y es recomendada para los proyectos con grandes equipos de desarrollo. Por otra parte, las ágiles dan mayor importancia a la capacidad de respuesta a los cambios, se enfatiza en la satisfacción del cliente y promueve el trabajo en equipo.

Metodologías pesadas. Proceso Unificado de Desarrollo

Las metodologías pesadas están basadas en normas provenientes de estándares seguidos por el entorno de desarrollo, ofrecen por lo general cierta resistencia a los cambios que puedan producirse durante el ciclo de desarrollo del producto, son procesos muy bien controlados con muchas políticas y normas a seguir. En este grupo se puede encontrar la metodología: Proceso Unificado de Desarrollo (*RUP*, en inglés *Rational Unified Process)*, actualmente la más utilizada de esta alternativa. La misma se caracteriza por ser dirigida por casos de uso, estar centrada en la arquitectura y ser iterativa e incremental. Entre sus principios se encuentran adaptar el proceso a la necesidad del cliente, equilibrar prioridades, demostrar valor iterativamente, fomentar la colaboración entre equipos, elevar el nivel de abstracción y enfocarse en la calidad. Sin embargo *RUP* presenta algunos inconvenientes para su aplicación en algunos proyectos como son: límite de tiempo demasiado ajustado, la documentación al ser tan exhaustiva, hace mucho énfasis en la planificación, no siempre se puede llevar a cabo la implementación del software de acuerdo a su planificación, pues aparecen cambios los cuales solo pueden ser percibidos en el momento de la codificación, entre otras.

Metodologías ágiles. Programación Extrema

Por otro lado las metodologías ágiles están basadas en heurísticas provenientes de prácticas de producción de código, y están especialmente preparadas para sufrir cambios durante el proyecto, son un proceso menos controlado y con pocos principios, por lo que desarrollan software en cortos períodos de tiempo y exigen poca documentación. Entre ellas se puede encontrar la metodología

Programación Extrema (*XP*, en inglés *Extreme Programming)*.

La *XP* fue concebida por Kent Beck, como un proceso de creación de software diferente al convencional. En palabras de Beck: "*XP* es una metodología ligera, eficiente, con bajo riesgo, flexible, predecible y divertida para desarrollar software"[27]. *XP* es una de las metodologías ágiles para el desarrollo de software más exitosas de la actualidad. Se utiliza en proyectos con pequeños equipos de desarrollo y con corto plazo de entrega. Se basa en la retroalimentación entre el cliente y el equipo de desarrollo, buena comunicación entre los participantes y simplicidad en las soluciones implementadas. Consiste en una programación rápida, cuya particularidad es tener como miembro del equipo al usuario final. Es adecuada para proyectos con requisitos imprecisos, muy cambiantes, y donde existe un alto riesgo técnico[16].

Para la presente investigación se seleccionó la metodología de desarrollo XP debido a los elementos que se precisan a continuación:

- Permite un grupo pequeño de desarrolladores.
- Necesidad de resultados tangibles a corto plazo.
- Imposibilidad para un grupo de desarrollo pequeño, de asumir una metodología robusta debido a la cantidad excesiva de roles y documentación generada en el ciclo de vida del proyecto.
- La necesidad de una retroalimentación continua entre el cliente y el equipo de desarrollo.
- Diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto.

1.3.2.Lenguaje de programación

Para el desarrollo de la presente investigación se usa el lenguaje de programación C#. Esto se debe a que el proceso de extracción implementado por el Centro de Identificación y Seguridad Digital, para el cual se desarrolla el componente, tiene definido el uso de este lenguaje.

1.3.3.Entorno de desarrollo

Se seleccionó como entorno integrado de desarrollo (IDE, por sus siglas en inglés) Visual Studio Ultimate 2013 Ultimate. Este incluye potentes herramientas que simplifican todo el proceso de desarrollo de aplicaciones de principio a fin. Los equipos pueden observar una mayor productividad y ahorro de costes al utilizar características de colaboración avanzadas, así como herramientas de

pruebas y depuración integradas que le ayudarán a crear siempre un código de gran calidad. Entre las ventajas del mismo están:

- Interpreta rápidamente el código.
- Crea enriquecidas experiencias de usuario.
- Utiliza las habilidades existentes: Trabaja el desarrollo de SharePoint, incluyendo herramientas para componentes web, listas, los flujos de trabajo, eventos y más.
- Dedica menos tiempo a la depuración: La jerarquía de llamada en línea ayuda rápidamente rastrea el flujo de ejecución de un programa sin invocar al depurador. También puede utilizar la marca de etiquetas de punto de interrupción para realizar una depuración más sencilla[28].

1.3.4. Framework de desarrollo

En el desarrollo del presente trabajo, debido a la necesidad de realizar procesos involucrados en la extracción de características de una imagen de huella dactilar ajenos del alcance del mismo, se decide usar el *framework Fingerprint Recognition* (FR) en su versión 2.2, ya que este posee las herramientas necesarias para la realización del pre-procesamiento de la huella y de esta forma la obtención de una imagen adelgazada de la misma, siendo esta de vital importancia para la realización de esta investigación[7].

Entre sus principales características se encuentran:

- Esta implementado en el lenguaje de programación C# usando el *framework* .NET.
- Permite la realización de pruebas de verificación con las bases de datos B de las FVC2000, FVC2002 y FVC2004 y en las bases de datos A de las FVC2002 y FVC2004 (Ver interfaz del *framework* en Anexo 6).
- Posee una amplia gama de algoritmos implementados para la extracción de características de la imagen de una huella dactilar.

1.4. Conclusiones parciales

El estudio realizado en este capítulo permitió:

- Analizar los conceptos asociados al post-procesamiento de imágenes de huellas dactilares para obtener un mejor dimensionamiento y entendimiento del problema.
- Evaluar los diferentes métodos para el post-procesamiento para obtener conocimiento referente a los algoritmos existentes para este proceso y así seleccionar el de mayor similitud con las características del problema planteado.
- Seleccionar la metodología, tecnologías y herramientas para lograr la satisfacción de los requisitos establecidos y posibilitando la obtención de un producto final con la calidad requerida.

Capítulo 2. Características del componente

Introducción

El presente capítulo está dedicado a las características del componente. Se abordan temas relacionados con la propuesta de solución para responder a la situación problemática en cuestión, además quedarán definidas las principales funcionalidades y requisitos no funcionales del sistema. Las funcionalidades se describen mediante las historias de usuario (HU). Se lleva a cabo la realización del plan de entrega, donde se indican las historias de usuario que se crearán para cada versión de la aplicación propuesta, así como las fechas en las que se publicarán dichas versiones. Se realiza el plan de iteraciones donde se muestran las HU a realizar en cada iteración según su prioridad en el negocio.

2.1 Propuesta de solución

El presente trabajo propone el desarrollo de un componente para el post-procesamiento de imágenes de huellas dactilares que realiza el filtrado de un conjunto de minucias con el objetivo de eliminar las que no aporten información identificativa para el posterior proceso de comparación en el Centro de Identidad y Seguridad Digital(CISED). Este componente está compuesto por algoritmos propuestos en Zhao2007[24] y en Ratha et al1995[23], quedando conformado como se describe a continuación:

1) Se elimina el tipo de minucia falsa correspondiente a la frontera de la huella dactilar y las minucias que entre su ángulo (α_m) y el de la orientación de la imagen (α_o) exista una diferencia (d_1). Donde $d_1 = 30^\circ$.
2) Se procede a eliminar las minucias de tipo ruptura corta, desechando cualquier par de minucias que se encuentren por debajo de una distancia umbral (d_2), luego todos los pares de minucias de tipo terminación, que se encuentren por debajo de una distancia umbral (d_3) y que la diferencia entre sus ángulos (α_{min1}, α_{min2}) sea menor que un ángulo umbral (α_1). Por último se elimina todo par de minucias de tipo terminación que cumpla con que la diferencia entre la orientación de sus ángulos (α_{min1}, α_{min2}) se encuentre dentro de un intervalo [α_2; α_3] y que la diferencia entre la orientación del ángulo de la línea que conecta las dos terminaciones y cualquiera de los ángulos α_{min1} o α_{min2} esté dentro de un intervalo [α_4, α_5]. Donde $d_2 = 6$, $d_3 = 10$, $\alpha_1 = 15^\circ$, $\alpha_2 = 145^\circ$, $\alpha_3 = 225^\circ$, $\alpha_4 = -25^\circ$, $\alpha_5 = 25^\circ$
3) En un primer momento se etiquetan los pixeles conexos de la imagen adelgazada, luego si la

distancia (*D*) entre una minucia de terminación y otra de bifurcación es menor que una distancia umbral (d_4) y sus etiquetas son las mismas, se etiquetan los pixeles nuevamente en una ventana pequeña (2D + 1x2D + 1) y si sus etiquetas siguen siendo las mismas entonces se eliminan ambas minucias. Donde $d_4 = 6$.

Lo anteriormente descrito conforma la estructura del componente, y precisamente en ese orden se ejecutarán las acciones para eliminar las minucias falsas.

2.2 Principales funcionalidades

El objetivo principal de la definición de las principales funcionalidades es identificar lo que se necesita para el correcto funcionamiento del producto. La definición y descripción de estos, ayuda a que exista una mayor comunicación entre el cliente y el equipo de desarrollo. Se han detectado como principales funcionalidades las siguientes:

1. Eliminar efecto fronterizo.
   1.1. Eliminar minucias de la frontera.
   1.2. Eliminar minucias con diferente orientación.
2. Eliminar rupturas cortas.
   2.1. Eliminar minucias cercanas.
   2.2. Eliminar minucias de rupturas cortas.
3. Eliminar espolones.
   3.1. Etiquetar pixeles conexos.
   3.2. Eliminar minucias de tipo espolones.

2.3 Requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Las propiedades o cualidades que hacen este producto más confiable y robusto se definen a continuación:

Rendimiento

- Los tiempos de respuestas del componente deben ser menores o iguales a lo establecido por estándares internacionales.

Software

- Windows XP o superior.
- Framework .NET en su versión 4.5.

Hardware

- Un microprocesador CPU 1.5 GHz o superior y 512MB de memoria RAM o superior.

2.4 Metáfora

En XP no se enfatiza la definición temprana de una arquitectura estable para el sistema. Dicha arquitectura se asume evolutiva y los posibles inconvenientes que se generarían por no contar con ella explícitamente en el comienzo del proyecto se solventan con la existencia de una metáfora. El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo. Una metáfora es una historia compartida que describe cómo debería funcionar el sistema. La práctica de la metáfora consiste en formar un conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema. Este conjunto de nombres ayuda a la nomenclatura de clases y métodos del sistema[29].

El componente para el post-procesamiento de imágenes de huellas dactilares se divide en tres subprocesos. Este toma como entrada un conjunto de minucias extraídas del esqueleto de la huella dactilar, el mismo esqueleto y una matriz con la orientación de la imagen en diferentes bloques de la huella. Todas estas entradas son obtenidas del *framework* FR, el cual es el encargado de realizar la etapa de pre-procesamiento.

En primer lugar se eliminan las falsas minucias que correspondan al área de la frontera y las que tengan cierta diferencia con la orientación de la imagen en su área local. Como resultado de esto se obtiene un conjunto menor de minucias, quedando aun falsas minucias por ser eliminadas.

En un segundo subproceso, son tratadas las minucias con el objetivo de eliminar las de tipo rupturas cortas y las que se encuentren muy cerca unas de otras, como son las estructuras de lagos, islas y fusiones (Ver Tabla. 7). Para esto son calculadas las distancias entre los puntos de minucias. También se tienen en cuenta los ángulos de orientación de dichos puntos, así como el de la línea que los une.

Finalmente, un último subproceso es el encargado de eliminar falsas minucias de tipo espolones. Es necesario para esta fase crear una nueva matriz donde los pixeles conexos del esqueleto de la huella sean etiquetados. De este modo se puede saber si un punto de bifurcación y uno de terminación forman parte de este tipo de estructura.

2.5 Fase de exploración

Para llevar a cabo la correcta implementación de la herramienta propuesta, es necesario realizar la

especificación de requisitos, los detalles del tiempo que requiere la implementación y la estimación del riesgo. La fase de Exploración de la metodología XP se desarrolla en un periodo de tiempo en el que se realizan un conjunto de funcionalidades conocidas como Historias de Usuarios (HU) uno de los artefactos fundamentales que genera dicha metodología.

2.5.1 Historias de usuario

La metodología XP utiliza las historias de usuario para especificar y administrar los requisitos del sistema, permitiendo de esta manera tener una visión general de lo que se desea crear. Representan una breve descripción del comportamiento del sistema y emplea terminología del cliente sin lenguaje técnico, se realiza una HU por cada característica principal del sistema.

Debido a lo inestable que suele ser la planificación de horas dedicadas al desarrollo de cada historia de usuario, se decide establecer el aproximado del tiempo de estimación en días ideales (la medida de tiempo más utilizada es por semanas). El tiempo de estimación expuesto en las tablas es un aproximado, así como la prioridad de cada historia de usuario.

Para el presente trabajo de diploma se obtienen un total de 3 HU que son implementadas en varias iteraciones. A continuación se muestran las HU asociadas a los requerimientos funcionales antes mencionados.

Tabla. 7 HU Eliminar efecto fronterizo. Elaboración propia.

Historia de usuario	
Número: 1	**Nombre:** Eliminar efecto fronterizo.
Usuario: Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Estimación aproximada: 10	**Iteración asignada:** 1
Descripción: dado un conjunto de minucias y la imagen de orientación de una huella son eliminadas las minucias que: • Cumplan con la condición de estar en la zona de la frontera de la huella dactilar (**Eliminar minucias de la frontera**). • Su ángulo de orientación difiera de un intervalo determinado con la orientación de la imagen en su área local (**Eliminar minucias con diferente orientación**).	
• **Observaciones:** Se debe disponer de un conjunto de minucias extraídas de un esqueleto de huella dactilar y la imagen de orientación de esta.	

Tabla. 8 HU Eliminar rupturas cortas. Elaboración propia.

Historia de usuario	
Número: 2	**Nombre:** Eliminar rupturas cortas.
Usuario: Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Estimación aproximada: 20	**Iteración asignada:** 1 y 2
Descripción: dado un conjunto de minucias y el esqueleto de una huella se debe: • Eliminar cualquier par de estas que se encuentre a una distancia determinada (**Eliminar minucias cercanas**). • Eliminar las de tipo terminación que se encuentren a una distancia determinada, la diferencia entre sus ángulos de orientación este de un intervalo determinado y la diferencia entre el ángulo de orientación de los puntos que unen las minucias y el de cualquiera de las dos se encuentre en un intervalo de ángulos (**Eliminar minucias de rupturas de crestas**).	
• **Observaciones:** Se debe haber eliminado las minucias pertenecientes a la frontera de la huella dactilar. Además es necesario disponer del esqueleto de la huella procesada y del conjunto de minucias previamente procesado.	

Tabla. 9 HU Eliminar espolones. Elaboración propia.

Historia de usuario	
Número: 3	**Nombre:** Eliminar espolones.
Usuario: Usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Estimación aproximada: 15	**Iteración asignada:** 1 y 2
Descripción: dado un conjunto de minucias y el esqueleto de una huella: • Se debe crear un nuevo esqueleto de la huella donde los pixeles conexos estén etiquetados (**Etiquetar pixeles conexos**). • Si una minucia de tipo bifurcación y otra de tipo terminación tienen la misma etiqueta, se crea una nueva matriz centrada en el punto de bifurcación. Las dimensiones de esta serán definidas por la distancia entre los dos puntos. Si en esta nueva imagen aún conservan las mismas etiquetas, ambas son eliminadas (**Eliminar minucias de tipo espolón**).	
• **Observaciones:** Se debe haber eliminado las minucias cercanas y de tipo ruptura corta. Además es necesario disponer del esqueleto de la huella procesada y del conjunto de minucias previamente procesado.	

2.6 Fase de planificación

En esta fase el cliente prioriza las historias de usuario y se realiza por parte de los desarrolladores una estimación del esfuerzo. La medida para calcular dicho esfuerzo es el punto y por lo general, un punto equivale a una semana ideal, en el caso particular de la presente investigación se mantendrá durante todo el documento como valor de un punto, un día ideal debido a las características del equipo de desarrollo y el entorno de trabajo. Se acuerda el alcance de la primera entrega y se define en conjunto con el cliente, el cronograma de entrega. La duración del cronograma de la primera release normalmente no excede dos meses. La fase de planeamiento solo toma unos pocos días.

La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según el alcance del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación[30].

2.6.1 Prioridad de las historias de usuario.

Asignar una prioridad a las historia de usuario por parte del cliente, es lo que decide el orden en el cual serán implementadas las mismas, siempre y cuando estén de acuerdo usuario y desarrollador. Las historias de usuario con menor número de prioridad son las más importantes, quedando conformadas de la siguiente manera:

Tabla. 10 Prioridad de las historias de usuarios. Elaboración propia.

Historia de usuario	Prioridad
Eliminar efecto fronterizo.	1
Eliminar rupturas cortas	1
Eliminar espolones	1

2.6.2 Estimación de esfuerzos

Para que el proceso de desarrollo de la propuesta de solución sea satisfactorio y eficiente, se lleva a cabo la estimación de esfuerzos para cada una de las Historias de Usuarios identificadas en el

anteriormente. Esta estimación se basa principalmente en la velocidad del equipo de desarrollo y en la semejanza con historias de usuario desarrolladas con anterioridad. Las HU de la presente investigación tienen un valor entre 15-25 puntos cada una.

Los puntos estimados son expresados en días ideales, se debe tener en cuenta que muy pocas veces el cronograma se lleva a cabo exactamente como se planifica. El esfuerzo necesario para construir las historias de usuario está basado en la técnica de estimación para el desarrollo ágil, quedando conformada de la siguiente manera:

Tabla. 11 Estimación de esfuerzos por Historia de usuario. Elaboración propia.

Historia de usuario	Esfuerzo necesario (Puntos estimados)
Eliminar efecto fronterizo.	15
Eliminar rupturas cortas	20
Eliminar espolones	25

2.7 Liberaciones

En el momento de planificar la liberación de un software se debe llevar un balance de esta, pues si se realiza demasiado pronto no se recogerán las funcionalidades necesarias para que ameriten dicha liberación, mientras que esperar mucho tiempo, conllevaría a que el software desarrollado quede atrás respecto a la competencia. La idea es producir rápidamente versiones del sistema que sean operativas, aunque obviamente no cuenten con toda la funcionalidad pretendida para el sistema pero que constituyan un resultado de valor para el negocio. Una entrega no debería tardar más de 3 meses.

Entre los elementos a tener en cuenta en el cronograma de liberación, es fundamental saber que rara vez un plan marcha respecto a lo acordado, por esa razón el plan de liberación está en constante cambio. Un ejemplo de ello se tiene, cada vez que el usuario cambia la prioridad de una historia, cuando el desarrollador divide o une historias de usuario o cuando aparecen imprevistos en las tecnologías a utilizar. Es por esto que los cambios deben ser aceptados. El cronograma de liberación de la presente investigación queda conformado de la siguiente manera:

Tabla. 12 Cronograma de liberación. Elaboración propia.

Iteración	Fecha de liberación
Primera iteración	3ra semana de Marzo de 2014
Segunda iteración	1ra semana de Mayo de 2014

2.8 Fase de iteración

Una vez descritas e identificadas las historias de usuario, además de estimado el esfuerzo necesario para el desarrollo de cada una de ellas, se procede a realizar la planificación de las etapas de implementación del sistema. Con este fin se define un plan que contiene las iteraciones a realizar y la relación de las historias de usuario que serán implementadas y en qué orden para cada iteración, teniéndose en cuenta la duración de las mismas. La cantidad de iteraciones necesarias (IN) para realizar todas las historias de usuario identificadas es obtenida sumando todos los puntos de esfuerzo (PE) de las historias de usuario y dividiéndolos entre la velocidad de iteración del equipo (VIE).

IN = PE + VIE
PE = 60 días (12 semanas)
IN = 12 / 8
IN = 1.5

Para calcular la velocidad de iteración del equipo (VIE) se divide la cantidad de desarrolladores (CD) entre el factor de dedicación (FD) al proyecto (para la presente investigación es de 2 [50%]) y se multiplica por el tiempo de duración máximo de una iteración (DMI) (en el caso de la presente investigación el tiempo es de 16 días máximo).

*VIE = (CD / FD) * DMI*
CD = 1, FD = 50% (2), DMI = 16
*VIE = (1 / 2) * 16*
VIE = 8

Del cálculo anterior se obtiene que el valor de la velocidad de equipo es 8, y las iteraciones necesarias para desarrollar las historias de usuario 2. Las iteraciones quedaron conformadas de la siguiente manera:

Tabla. 13 Primera iteración. Elaboración propia.

Iteración		
Numero: 1	**H.U (por orden):** 1,2,3	**Duración total:** 40
Descripción: Se comienzan a implementar las HU priorizando la numero 1, la cual debe quedar terminada en esta primera iteración. Se debe obtener una versión funcional del componente, que luego será sometida a pruebas para rectificar umbrales empleados.		
Fecha de liberación: Marzo de 2014		

Tabla. 14 Segunda iteración. Elaboración propia.

Iteración		
Numero: 2	**H.U (por orden):** 2, 3	**Duración total:** 20
Descripción: Se continúa en la implementación de las HU 2 y 3. Se rectifican los errores detectados de la iteración anterior, además de la optimización del código. Al concluir esta iteración se obtendrá una versión 1.0 del componente.		
Fecha de liberación: Mayo de 2014		

2.9 Diseño del componente

2.9.1 Patrones

Los patrones constituyen soluciones a problemas recurrentes que ocurren en el entorno. Luego de llegar a la solución se encapsulan todas las variables y factores de la misma, lo cual conforma una guía para resolver una y otra vez el mismo problema. Entre sus características se encuentran: describir el problema de forma sencilla, describir el contexto en el que ocurre, puntualizar los pasos a seguir, hacer énfasis en los puntos fuertes y débiles de la solución, referir otros patrones asociados, entre otras. En la presente investigación se abordan los patrones arquitectónicos y los patrones de diseño que son utilizados para conformar el diseño de la aplicación propuesta[31].

2.9.2 Patrones arquitectónicos

Los patrones arquitectónicos expresan fundamentalmente esquemas de organización estructural para sistemas de software, brindando un conjunto de subsistemas predefinidos, especificando sus

responsabilidades e incluyendo reglas y guías con el objetivo de lograr la organización de las relaciones entre estos[32].

Para el desarrollo del componente propuesto por el presente trabajo se seleccionó el patrón arquitectónico por capas, con el objetivo de separar el código del programa dependiendo de su naturaleza. En este caso particular está compuesto solamente por 2 capas debido a que no se necesita una capa de acceso a datos.

Capa de presentación: Representa la interfaz gráfica de prueba para la interacción con el componente. A través de varias opciones, muestra los subprocesos que se realizan hasta completar el post-procesamiento (Interfaz gráfica de prueba, ver Anexo 5).

Capa de negocio: Refleja la lógica del sistema representando la organización de las clases y relaciones entre estas. (Ver Figura. 15).

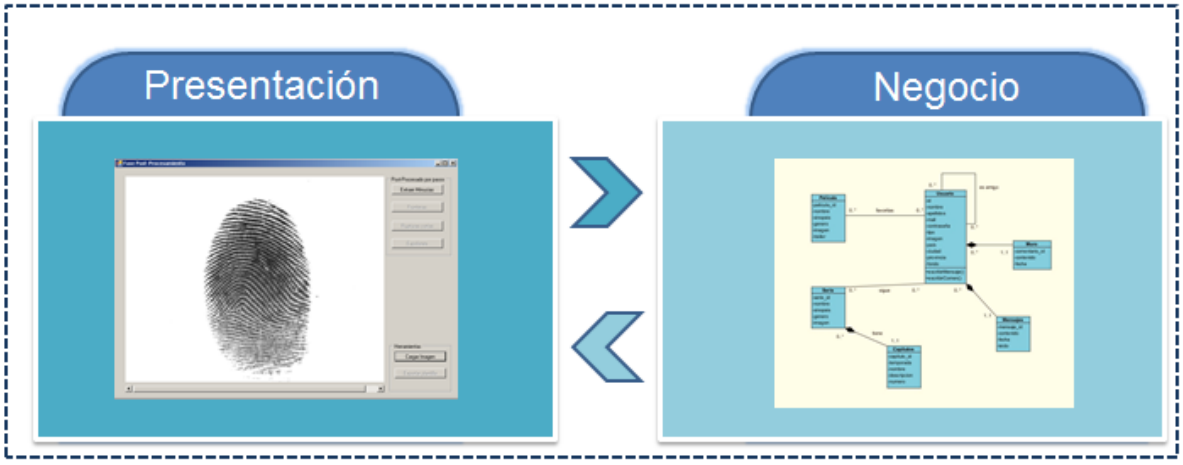


Figura. 20 Patrón arquitectónico por capas. Elaboración propia.

Debido a que los procesos que se realizan dentro del componente pueden ser descritos como un subconjunto de tareas a realizarse con el objetivo de filtrar una entrada de datos, guiados por un orden especifico en forma de tubería y resultando en una salida de datos procesados, se seleccionó con patrón arquitectónico de apoyo el de tuberías y filtros, ya que este coincide perfectamente con la descripción anteriormente planteada [32]. A continuación se muestra como queda estructurado el componente usando el patrón arquitectónico escogido.

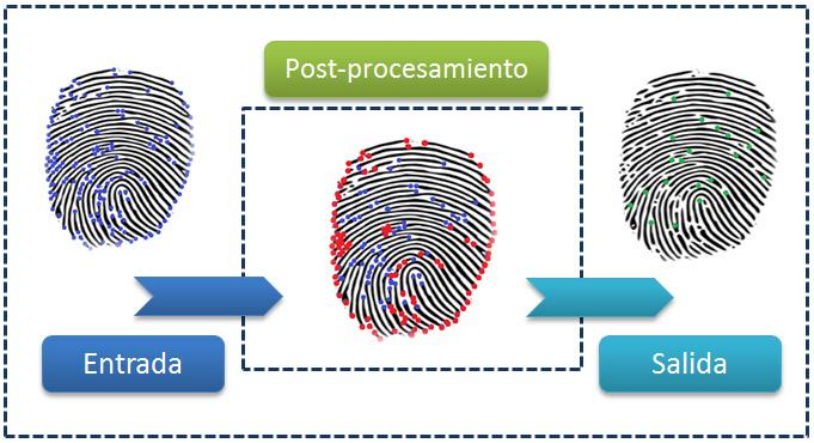


Figura. 21 Patrón arquitectónico tuberías y filtros. Elaboración propia.

2.9.3 Patrones de diseño

Los patrones de diseño son descripciones de clases y objetos relacionados que están particularizados para resolver un problema de diseño general en un determinado contexto [33]. Entre los más conocidos se encuentran los patrones de Asignación de Responsabilidades (GRASP, de sus siglas inglés, *General Responsibility Asignment Software Patterns*) y los patrones de la Banda de los Cuatro (GoF, de sus siglas en inglés, *Gang of Four*).

Dentro de los patrones **GRASP** se usaron:

Controlador: es un objeto de interfaz no destinada al usuario que se encarga de efectuar las asignaciones en cuanto al manejo de los eventos del componente y definir tanto las operaciones como el orden de las mismas. Se evidencia en la clase Controller.cs que gestiona todo el flujo de datos del componente (Ver Figura. 22).

```
public List<Minutia> PostProcessing(List<Minutia> listMinutiae, OrientationImage
orImage, ImageMatrix skeleton)
{
    var minutiaeList = new BoundaryEffects().BorderMinutiae(listMinutiae, orImage);
    minutiaeList = new ShortBreaks().ShortBreaksMinutiae(minutiaeList, skeleton);
    minutiaeList = new Spur().SpurMinutiae(minutiaeList, skeleton);
    sw.Stop();

    return minutiaeList;
```

Figura. 22 Ejemplo de patrón controlador. Elaboración propia.

Experto: Es uno de los más utilizados ya que define quien asumirá la responsabilidad en cada caso, buscando siempre al experto en la información. Permite darle solución al problema el cual sería el

principio fundamental para asignar responsabilidades en el diseño orientado a objetos. El cumplimiento de una responsabilidad requiere a menudo información distribuida en varias clases de objetos[33]. El uso de este patrón permitió a los objetos valerse de su propia información para hacer lo que se les pide, favoreció la existencia de mínimas relaciones entre las clases, lo que permite contar con un sistema sólido y fácil de mantener. Ejemplo de este patrón es la clase Neighbor.cs, esta es experta en la información referente a su tipo de datos (Ver Figura. 23).

```
public class Neighbor
{

    public Neighbor(int x, int y, bool labeled, int value)
    {

    public bool Labeled { set; get; }
    public int X { set; get; }
    public int Y { set; get; }
    public int Value { set; get; }

}
```

Figura. 23 Ejemplo de patrón Experto. Elaboración propia.

Creador: Para guiar la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El diseño bien asignado permitirá soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y la reutilización. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento. El uso de este patrón permitió crear las dependencias mínimas necesarias entre las clases, beneficiando el mantenimiento del sistema y brindando mejores oportunidades de reutilización. Este patrón se pone de manifiesto en varias clases, una de ellas es ConnectedComponentLabeling.cs, que es la encargada de etiquetar los componentes conexos en una imagen adelgazada de huella dactilar, creando para este propósito objetos de tipo Neighbor (Ver Figura. 24).

```
public class ConnectedComponentLabeling
{
    public ImageMatrix LabeledMatrix(ImageMatrix imageMatrix)
    {

        var neighborMatrix = new Neighbor[imageMatrix.Height,imageMatrix.Width];
        var neighborList = new List<Neighbor>();
```

Figura. 24 Ejemplo de patrón Creador. Elaboración propia.

Bajo acoplamiento: Este patrón asigna una responsabilidad para mantener el bajo acoplamiento, de modo que no dependa de muchas otras. La clase no tendrá muchas dependencias, proporcionando la reutilización. Este patrón no puede verse separado del patrón Experto y del patrón Alta Cohesión.

Ventajas:

- No afectan los cambios en otros componentes.
- Fácil de entender de manera aislada.
- Conveniente para reutilizar[33].

El empleo de este patrón se evidencia en la clase ShortBreaks.cs, dicha clase no depende de ninguna otra (Ver Figura. 19).

Alta cohesión: Este patrón define que una clase tiene responsabilidades moderadas en un área funcional y colabora con otras clases para llevar a cabo las tareas. Una clase con alta cohesión tienen un número relativamente pequeño de métodos y no realiza mucho trabajo. Colabora con otros objetos para compartir el esfuerzo si la tarea es extensa. Además es ventajosa porque es relativamente cómodo de mantener, entender y reutilizar.

Ventajas:

- Se incrementa la claridad y facilita la comprensión del diseño.
- Se simplifican el mantenimiento y las mejoras[33].

Un ejemplo del uso de este patrón en la presente solución se puede apreciar en las clases ShortBreak.cs, BoundaryEffects.cs y Spur.cs las cuales contienen responsabilidades estrechamente relacionadas, encargadas de controlar las acciones en sus respectivas partes del negocio. Esto hace posible que el software sea flexible a cambios sustanciales con efecto mínimo.

2.9.4 Tarjetas CRC

Cuando se lleva a cabo el diseño de un sistema, *XP* propone la idea de llevar a cabo las tareas de la

manera más simple posible, aplicando prácticas especializadas que la metodología propone. Dichas prácticas inciden directamente en la realización y elaboración del diseño de un software pues, cuando se tiene un diseño simple se presta mayor atención a codificar, tarea sobre la cual hace más énfasis esta metodología.

Ciertamente la metodología *XP* no presta especial atención a los diagramas *UML,* pues existe la creencia de que los verdaderos seguidores de *XP* no diagraman. Esta creencia viene dada por 2 causas principales: primero está el hecho de que algunas personas encuentran los diagramas útiles y otras no, y la segunda causa es que los diagramas tienden a asociarse con procesos pesados. Tales procesos invierten mucho tiempo en dibujar diagramas que no ayudan y pueden dañar más que auxiliar. Para hacer un buen uso de los diagramas *UML,* lo primero que se debe tener en cuenta es saber para qué se dibujan, debido a que la comunicación es el principal objetivo de dichos diagramas. Un problema común con el uso de los diagramas es que por lo general se intenta hacerlos extensos, convirtiéndose la exhaustividad en la enemiga de su comprensión[30].

Partiendo de lo anteriormente descrito, se puede apreciar que la metodología *XP* no recomienda la utilización de diagramas de clases, debido a la gran cantidad de documentación e inconvenientes que ello requiere, aunque tampoco excluye del todo su elaboración, en su lugar orienta la realización de Tarjetas de Contenido, Responsabilidad, Colaboración, (*CRC*, por sus siglas en inglés). Dichas tarjetas, en cada una de las iteraciones van siendo definidas como herramientas de reflexión en el diseño de software orientado a objetos. Estas tarjetas brindan un acercamiento a la identificación de las clases y la información referente a los objetos desarrollados en el proyecto. Cada tarjeta identifica una clase, sus responsabilidades y relaciones con otras clases.

Cada tarjeta CRC contiene el nombre de la clase que representa, además existe una descripción de las responsabilidades (métodos) asociados con la clase, así como una lista de otras clases relacionadas mediante el envío de mensajes. El sistema CRC se usa principalmente como herramienta didáctica y como metodología para estudiar la conducta de los diseñadores orientados a objetos. Las tarjetas CRC son también un recordatorio y ayudan a los programadores experimentados y principiantes a comunicarse entre sí acerca de la modelación del entorno con objetos[34].

A continuación se muestran como quedaron conformadas las tarjetas CRC para el componente:

Tabla. 15 Tarjeta CRC PostProcesadoIU. Elaboración propia.

Tarjeta CRC	
Clase: PostProcesadoIU	
Responsabilidades	**Colaboraciones**
– Cargar imagen. – Mostrar post-procesado de la huella dactilar. .	– FR.Componente.dll – FR.Ratha et al1995.dll – FR.Sherlock1994.dll – FR.Core.dll

Tabla. 16 Tarjeta CRC MinutiaeExtractor. Elaboración propia.

Tarjeta CRC	
Clase: MinutiaeExtractor	
Responsabilidades	**Colaboraciones**
– Extraer las minucias. – Calcular ángulo de minucias. – Obtener vecinos de minucias	– FR.Core.dll

Tarjeta CRC	
Clase: Controller	
Responsabilidades	**Colaboraciones**
– Definir el orden de ejecución de los filtros para el post-procesamiento.	– -

Tabla. 17 Tarjeta CRC BoundaryEffects. Elaboración propia.

Tarjeta CRC	
Clase: BoundaryEffects	
Responsabilidades	**Colaboraciones**
– Eliminar efecto fronterizo.	– FR.Core.dll

Tabla. 18 Tarjeta CRC ShortBreaks. Elaboración propia.

Tarjeta CRC	
Clase: ShortBreaks	
Responsabilidades	**Colaboraciones**
– Eliminar rupturas cortas. – Calcular ángulo de minucias por su 8vo vecino. – Obtener vecinos de minucias.	– FR.Core.dll – ImageProcessingTools.dll

Tabla. 19 Tarjeta CRC Spur. Elaboración propia.

Tarjeta CRC	
Clase: Spur	
Responsabilidades	**Colaboraciones**
– Eliminar espolones. – Obtener vecinos de minucias.	– FR.Core.dll – ImageProcessingTools.dll – ConnectedComponentLabeling.cs

Tabla. 20 Tarjeta CRC ConnectedComponentLabeling. Elaboración propia.

Tarjeta CRC	
Clase: ConnectedComponentLabeling	
Responsabilidades	**Colaboraciones**
– Etiquetar componentes conexos en imagen adelgazada.	– ImageProcessingTools.dll – Neighbor.cs

Tabla. 21 Tarjeta CRC Neighbor. Elaboración propia.

Tarjeta CRC	
Clase: Neighbor	
Responsabilidades	**Colaboraciones**
– Manejar datos asociado el tipo de	–

dato Neighbor.	

2.10 Conclusiones parciales

El componente para el post-procesamiento de imágenes de huellas dactilares le permitirá al CISED eliminar las falsas minucias extraídas de un esqueleto de huella. Del mismo modo que se disminuirán los tiempo de respuesta del módulo de comparación y las tasas de errores de estos.

- Se definieron los requisitos no funcionales del sistema, la metáfora y la elaboración de historias de usuario permitieron un mejor entendimiento del funcionamiento del sistema y las diferentes funcionalidades con las que este cumple, según las necesidades del cliente.
- La selección de una arquitectura y de los patrones de diseño para el componente, permitieron una mejor organización de la información.
- El tiempo estimado, el plan de iteraciones y de entrega permitieron fijar metas para el cumplimiento de las diferentes iteraciones.

Capítulo 3. Implementación y pruebas

Introducción

En este capítulo se exponen todas las particularidades de la implementación del componente, en aras de obtener un producto final que esté en correspondencia con los requisitos definidos por el cliente, lo cual se validará aplicando los diferentes tipos de prueba. Para ello se definen los estándares de codificación que debe cumplir el equipo de desarrollo, además de presentarse las interfaces creadas para comprobar el funcionamiento del componente.

3.1. Estándares de codificación

Un estándar de codificación son reglas que se siguen para la escritura del código fuente. Para una mejor comprensión del código se definieron una serie de estándares basados en diversas reglas.

3.1.1. Estilos para la capitalización

Utilización de la convención Pascal: Se define el primer caracter de cada palabra en mayúscula y el resto en minúscula. Ejemplo: PostProcessing.

Utilización de la convención Camel: Se define el primer caracter de cada palabra en mayúscula (excepto la primera palabra) y el resto en minúscula. Ejemplo: postProcessing.

Convenciones para los nombres

```
public class Neighbor
{
    public Neighbor(int x, int y, bool labeled, int value)
    {
```

Figura. 25 Convención Pascal para el nombre de las clases. Elaboración propia.

```
public ImageMatrix LabeledMatrix(ImageMatrix imageMatrix)...
```

Figura. 26 Convención Pascal para el nombre de los métodos. Elaboración propia.

```
var result = new List<Minutia>();
var toErase = new bool[listMinutiae.Count];
var distance = new MtiaEuclideanDistance();
```

Figura. 27 Convención Camel para nombre de las variables. Elaboración propia.

Reglas de codificación

- No usar caracteres simples para el nombre de las variables i, n, s, etc. Una excepción de esta regla son las variables dentro de los ciclos.
- No usar nombres de variables que coincidan con palabras reservadas.
- Los comentarios deben estar en el mismo nivel del código.
- No escribir comentarios para cada línea de código o para cada variable declarada.
- Las llaves se deben poner al mismo nivel del código que las contiene.
- Usar una línea en blanco para separar agrupaciones lógicas del código.
- Debe dejarse una y solo una línea en blanco entre cada método dentro de las clases.
- Las llaves deben ser utilizadas sobre líneas separadas y no sobre la misma línea como en if, for etc.
- Usar un espacio simple antes y después de cada operador y llave.

3.2. Implementación del sistema

Para describir las tareas llevadas a cabo en la fase de implementación, se emplea un lenguaje técnico el cual, no necesariamente debe ser entendible por el cliente. Dichas tareas son asignadas al equipo o programador responsable, normalmente la codificación se lleva a cabo por una pareja de programadores. Esta labor se lleva a cabo con el objetivo de detallar mejor las historias de usuario, lo cual facilita el entendimiento en el proceso de implementación. Cada historia de usuario puede contener una o más tareas de ingeniería, explicando de forma general las acciones que se realizan en la misma.

3.2.1. Tareas de las historias de usuario

A continuación son presentadas las tareas definidas para el desarrollo de la presente investigación, mostrando las tareas de las HU Cargar imagen, Extraer minucias, Eliminar efecto fronterizo, Eliminar rupturas cortas, Eliminar espolones y Exportar plantilla de minucias.

Tabla. 22 Tarea: Eliminar las falsas minucias de tipo fronterizas. Elaboración propia.

Tareas de Historia de Usuario		
ID: 1	**H.U**. Eliminar efecto fronterizo.	**Iteración:**1
Nombre: Eliminar las falsas minucias de tipo fronterizas.		
Tipo: Desarrollo.	**Puntos estimados:** 12	

Fecha inicio: 29/1/14	**Fecha Fin:** 18/2/14
Responsable: Angel Noel Castro Tejas.	
Descripción: Aplicar reglas de validación a cada minucia para determinar si son verdaderas, mientras se eliminan las falsas.	

Tabla. 23 Tarea: Eliminar las falsas minucias de tipo ruptura corta. Elaboración propia.

Tareas de Historia de Usuario		
ID: 2	**H.U**. Eliminar rupturas cortas.	**Iteración:**1, 2
Nombre: Eliminar las falsas minucias de tipo ruptura corta.		
Tipo: Desarrollo.	**Puntos estimados:** 14	
Fecha inicio: 19/2/14	**Fecha Fin:** 30/4/14	
Responsable: Angel Noel Castro Tejas.		
Descripción: Aplicar reglas de validación a minucias en pares, para determinar si son verdaderas, mientras se eliminan las falsas.		

Tabla. 24 Tarea: Eliminar las falsas minucias de tipo espolones. Elaboración propia.

Tareas de Historia de Usuario		
ID: 3	**H.U**. Eliminar espolones.	**Iteración:**1, 2
Nombre: Eliminar las falsas minucias de tipo espolones.		
Tipo: Desarrollo.	**Puntos estimados:** 12	
Fecha inicio: 5/3/14	**Fecha Fin:** 30/4/14	
Responsable: Angel Noel Castro Tejas.		
Descripción: Aplicar reglas de validación a minucias en pares, para determinar si son verdaderas, mientras se eliminan las falsas.		

Tabla. 25 Tarea: Etiquetar las crestas de una imagen adelgazada de huella dactilar. Elaboración propia.

Tareas de Historia de Usuario		
ID: 4	**H.U**. Eliminar espolones.	**Iteración:**1, 2
Nombre: Etiquetar las crestas en una imagen adelgazada de huella dactilar.		
Tipo: Desarrollo.	**Puntos estimados:** 4	

Fecha inicio: 15/3/14	**Fecha Fin:** 30/4/14
Responsable: Angel Noel Castro Tejas.	
Descripción: Asignar un identificador a cada cresta de la imagen adelgazada de huella dactilar.	

3.3. Pruebas

Dentro de las buenas prácticas de la metodología *XP*, se encuentra la de llevar a cabo un Desarrollo Guiado por Pruebas (TDD por su siglas en inglés), pues de esta manera se reduce el número de errores no detectados, así como el tiempo entre la introducción de estos en el sistema y su descubrimiento. Es por esta razón que el sistema en desarrollo está siendo testeado constantemente. La metodología *XP* divide las pruebas en 2 grupos fundamentales, las pruebas unitarias y las pruebas de aceptación.

Una prueba unitaria consiste en la verificación de un módulo (unidad de código) determinado dentro de un sistema. Las pruebas unitarias aseguran que un determinado módulo cumpla con un comportamiento esperado en forma aislada antes de ser integrado al sistema. Para el componente desarrollado en el presente trabajo la realización de pruebas unitarias resulta un proceso altamente engorroso debido a que como se menciona anteriormente se necesita conocer los resultados que arrojará la funcionalidad a probar.

Con el objetivo de ilustrar lo anteriormente planteado, se realiza el siguiente análisis; para realizarle pruebas unitarias a la funcionalidad de eliminar efecto fronterizo se debe conocer de antemano la cantidad exacta de minucias que pertenecen a la frontera, su localización y orientación exactas en la huella, para de esta forma saber si el resultado arrojado en la prueba es correcto o no. Esta última parte es la que resulta compleja, pues no se cuenta en el Centro o en la Universidad con expertos en el tema (peritos), capaces de identificar estas características en una imagen de huella dactilar.

Por las razones anteriormente planteado al presente componente no se le realizan pruebas unitarias, si no pruebas de aceptación

3.3.1. Pruebas de fiabilidad

Para la validación del componente de post-procesamiento asociado a la presente investigación, se realizó una comparación con el componente que cuenta el *framework* FR. Para dicha comparación se usó una muestra aleatoria de imágenes de huellas dactilares de las base de datos de la FVC2000,

conformando un conjunto de 10 huellas, una por cada base de datos. Se capturaron las cantidades de minucias antes de realizar el post-procesamiento, y luego de realizado con ambos componentes (Ver Tabla. 28).

Tabla. 26 Cantidad de minucias antes y después del post-procesamiento. Elaboración propia.

Huella	Sin post-procesado		Post-procesado componente		Post-procesado *framework*	
	Bifurcaciones	Terminaciones	Bifurcaciones	Terminaciones	Bifurcaciones	Terminaciones
1_3	14	138	9	53	9	52
101_2	7	133	7	41	7	41
4_5	24	129	8	40	10	41
101_3	2	85	2	23	2	23
3_1	13	156	10	59	10	60
103_6	23	139	10	38	10	41
20_6	9	119	9	35	9	35
103_3	2	109	2	29	2	29
Totales	94	1008	57	318	59	322
PT (%)	8,53%	91,47%	5,17%	28,86%	5,35%	29,22%

Los resultados arrojados por este experimento corroboran que el componente perteneciente a la presente investigación elimina una gran cantidad de minucias falsas luego de aplicado el post-procesamiento. Por otro lado, en comparación con el componente del *framework* FR, se puede observar que se disminuyó en un 0,18% en las bifurcaciones y en un 0,36% en las de tipo bifurcaciones.

De forma adicional, se utilizó el componente en el proceso de extracción de características implementado por el *framework* FR y se obtuvieron las tasas de error asociadas a varios algoritmos de comparación. De este modo se obtuvo una medida de cuanto influye el componente en la precisión y el rendimiento de un módulo de comparación.

Existen diversas tasas de error, las cuales permiten arribar a conclusiones de que tan certero puede ser un algoritmo. Una de estas es la Tasa de Comparaciones Falsas (False matching rate, de sus siglas en inglés *FMR*) y es calculada de la siguiente manera: $FMR = \frac{IA}{TII}$ donde IA es el número de impostores aceptados y TII es el total de intentos de impostores. Esto significa que mientras menor sea el FMR mejor será el algoritmo en términos de precisión. Por otro lado está la Tasa de Comparaciones No Falsas (*False non matches rate,* de sus siglas en inglés *FNMR*) y es calculada de

la siguiente forma: $FNMR = \frac{GR}{TIG}$ donde GR es el número de genuinos rechazados y TIG es el total de intentos genuinos. Igual que la anterior tasa, entre más pequeño sea su valor, más preciso será el algoritmo.

La manera más sencilla y exacta de describir el rendimiento de un algoritmo de comparación dactilar es usando la Tasa de Error Igual (*Equal error rate*, de sus siglas en inglés *EER*). Esta no es más que el punto donde el FMR y el FNMR son iguales y mientras menor sea este menor será el error del algoritmo[35].

Para la realización de este experimento se usó el *framework* FR, ya que este cuenta con un módulo de pruebas que usa las bases de datos de huellas dactilares de la FVC2000. Para dicho caso de prueba se usaron las bases de datos DB1_A y DB1_B, para las cuales se usó un lector óptico de tipo KeyTronic. La primera cuenta con 800 huellas tomadas de 100 individuos, 8 por cada uno y la segunda cuenta con 80 huellas tomadas de 10 individuos, igualmente 8 por cada uno[35].

Para la realización de esta tarea, se tuvo en cuenta las prestaciones del ordenador donde se llevaron a cabo las pruebas. Se usó un microprocesador de 3.10Ghz con 2GB de memoria RAM. Por otro lado, con el objetivo de tener una comparación, se ejecutaron las pruebas usando el componente propuesto y sin el este (Ver Tablas 26 y 27).

Tabla. 27 Resultados de las pruebas sobre la base de datos DB1_A. Elaboración propia.

Algoritmo de comparación	TPC (ms)		EER (%)	
	SinComp	ConComp	SinComp	ConComp
Jiang2000[36]	29,70735	2,1751	45,922	33,916
Medina2012[37]	17,55935	0,82606	35,764	23,447
Liu2005[38]	257,43484	1,89123	23,165	20,19
Medina2012*[39]	86,61794	8,35123	27,606	24,787
Medina2011[40]	38,948	2,71729	39,752	35,294

Tabla. 28 Resultados de las pruebas sobre la base de datos DB1_B. Elaboración propia.

Algoritmo de comparación	TPC (ms)		EER (%)	
	SinComp	ConComp	SinComp	ConComp
Jiang2000[36]	24,77813	1,84531	42,5	27,937
Medina2012[37]	15,55	0,60781	24,683	16,19
Liu2005[38]	168,8438	1,19375	18,452	16,409

Medina2012*[39]	74,20469	7,84063	19,365	19,821

Como se puede observar en las Tablas 26 y 27 el componente reduce el tiempo promedio de comparación (TPC) considerablemente con los algoritmos de comparación empleados[36-40], igual sucede con la tasa de error igual (EER).

De los resultados anteriormente expuestos, se puede concluir que el componente propuesto en la presente investigación cumple con su objetivo, eliminando una gran cantidad de minucias falsas. Además de quedar demostrado que es capaz de influir en el rendimiento de un módulo de comparación, disminuyendo la tasa de error y los tiempos de respuesta.

3.4. Conclusiones parciales

- El uso de los estándares de codificación y las tareas de ingeniería facilitó la implementación del componente.
- La realización de pruebas de rendimiento al componente demostró que el componente disminuye los tiempos de respuesta de la etapa de comparación.
- Los resultados obtenidos en las pruebas de fiabilidad demostraron que el componente desarrollado es capaz de eliminar un gran número de falsas minucias y de disminuir las tasas de error de un algoritmo de comparación.

Conclusiones

1. Como resultado del estudio realizado se definió utilizar los algoritmos de Ratha et al y Zhao et al debido a que estos son capaces de eliminar un gran número de falsas minucias.
2. El uso de patrones de diseño y arquitectónicos brindó como resultado un diseño sólido y flexible para la codificación del componente.
3. Se conformó una solución capaz de eliminar una gran cantidad de minucias falsas durante el post-procesamiento de una imagen de huella dactilar.
4. El componente implementado logró realizar el proceso de filtrado de minucias eficientemente según los resultados de las pruebas de rendimiento realizadas.

Recomendaciones

1. Implementar otros algoritmos de post-procesamiento de imágenes de huellas dactilares citados en la presente investigación, de modo que sea posible usar combinaciones de los mismos en vistas de obtener un mejor resultado.

Bibliografía

1. Davide Maltoni, D.M., Anil K. Jain, Salil Prabhakar, *Handbook of Fingerprint Recognition*. 2009.
2. Mateos, M.T. and J.A.S. Pizarro, *Tecnologías biométricas aplicadas a la seguridad*. 2005: Ra-Ma.
3. Igor Bohm, F.T., *Biometric Systems.* 2004.
4. [En_Linea]. *Diccionario de la Real Academia Española*. 2014; Available from: http://lema.rae.es/drae/.
5. Piñol, M.J.L., *Analisis del algoritmo de Maio para la extracción de huellas dactilares. Optimización para una implementación hardware.* 2004.
6. NIST, *Data Format for the Interchange of Fingerprint, Facial & Other Biometric Information*. 2011, National Institute of Standards and Technology: United States of America.
7. D. Ashok Kumar, T.U.S.B., *A Comparative Study on Fingerprint Matching Algorithms for EVM.* Journal of Computer Sciences and Applications, 2013. **1**(4): p. 55-60.
8. Feng, J., *Combining minutiae descriptors for fingerprint matching.* Pattern Recognition, 2008: p. 342-352.
9. J.Venkatesh, V.D.A.K., *Advanced Filter and Minutia Matching For Fingerprint Recognition.* International Journal of Science and Research (IJSR), 2013. **2**(1): p. 52-57.
10. K. S. Sim, Y.K.T., M. E. Nia, and G. D. Lee, *Rotation-invariant Reference Point Location Detection Using Complex Filtering for Fingerprint Matching.* International Journal of Future Computer and Communication, 2012. **1**(3): p. 321-322.
11. Koichi Ito, H.N., Koji Kobayashi, Takafumi Aoki, Tatsuo Higuchi, *A Fingerprint Matching Algorithm Using Phase-Only Correlation.* IEICE Trans. Fundamentals, 2004. **E87-A**(3): p. 682-691.
12. M.A. Wahby Shalaby, M.O.A., *A multilevel structural technique for fingerprint representation and matching.* Elsevier, 2013: p. 56–69.
13. Rosario Arjona, I.B., *A hardware solution for real-time intelligent fingerprint acquisition*, in *Journal Real-Time Image Processing*. 2012.
14. Cruz, A.R., *Clasificación de huellas digitales mediante minucias*. 2009.
15. Hernández, A.D., *Componente de clasificación de huellas dactilares*, in *Centro de*

Identificación y Seguridad Digital. 2013, Universidad de las Ciencias Informáticas: Cuba. p. 77.

16. Luis Miguel Echeverry Tobón, L.E.D.C., *Curso práctico de la metodólogia agil xp al desarrollo de software*, in *Facultad de Ingenieria: Eléctrica, Electrónica, Física y Ciencias de la Computación*. 2007, Universidad Tecnológica de Pereira: Pereira.
17. Gabriel Babatunde Iwasokun, et al., *A Multi-Level Model for Fingerprint Image Enhancement.* Journal of Pattern Recognition Research 7, 2012.
18. Rajinikannan, M., D. Ashok Kumar, and R. Muthuraj, *Estimating the impact of fingerprint image enhancement algorithms for better minutia detection.* Int. J. Comput. Appl, 2010. **2**(1): p. 36-42.
19. Bansal, R., P. Sehgal, and P. Bedi, *Effective morphological extraction of true fingerprint minutiae based on the hit or miss transform.* International Journal of Biometric and Bioinformatics, 2010.
20. Kim, S., L.D., and J. Kim. *Algorithm for detection and elimination of false minutiae in fingerprint images*. in *Proceedings of the Third International Conference on Audio and Video Based Biometric Person Authentication*. 2001.
21. Lu, H., J.X., and W. Yau, *Effective and efficient fingerprint image postprocessing*, in *7th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. 2002.
22. Popovic, B., M.L., and M. Bandur, *Spurious fingerprint minutiae detection based on multiscale direc-tional information.* 2007.
23. Ratha, N.K., S. Chen, and A.K. Jain, *Adaptive flow orientation-based feature extraction in fingerprint images.* Pattern Recognition 28(11), 1995.
24. Zhao, F. and X. Tang, *Preprocessing and postprocessing for skeleton-based fingerprint minutiae extraction.* 2007.
25. X. D. Jiang, W. Y. Yau, and W. Ser., *Detecting the fingerprint minutiae by adaptive tracing the gray-level ridge.* Pattern Recognition, 2001. **34**.
26. Popovic, B.M. and L. Maskovic, *Fingerprint Minutiae Filtering Based on Multiscale Directional Information.* 2007.
27. Kent Beck, M.F., *Planning Extreme Programming*. 2000: Addison Wesley.
28. Gousset, M., B. Keller, and A. Krishnamoorthy, *Professional Application Lifecycle Management*

with Visual Studio 2012. 2012.

29. Fowler, M. *¿Ha muerto el diseño?* 2004; Available from: http://martinfowler.com/articles/designDead.html.
30. Patricio Letelier, M.C.P., *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP)*. 2006, Universidad Politécnica de Valencia
31. Erick Gamma, R.H., Ralph Jonhson, Jonh Vlissides, *Patrones de Diseño: elementos de software orientados a objetos reutilizables.*, ed. Addison-Wesley. 1997.
32. Frank Buschmann, R.M., Hans Rohnert, Peter Sommerlad, Michael Stad, *Arquitectura de software orientada a patrones.*, ed. Wiley. Vol. 1. 1996.
33. Larman, C., *UML y Patrones: Introducción al análisis y diseño orientado a objetos.* 1999, PRENTICE HALL, Mexico. 536.
34. Sandra Casas, H.R., *Aspectos Tempranos: un enfoque basado en Tarjetas CRC*. 2009: Colombia.
35. Raffaele Cappelli, D.M., Davide Maltoni, James L. Wayman, Anil K. Jain, *Performance Evaluation of Fingerprint Verification Systems.* IEEE Transactions on Pattern Analisys and Machine Intelligence, 2006. **Vol. 28**.
36. Xudong Jiang, W.-Y.Y. *Fingerprint minutiae matching based on the local and global structures*. in *International Conference on Pattern Recognition* 2000: IEEE.
37. M. A. Medina Pérez, M.G.B., A. E. Gutierrez Rodriguez, L. Altamirano Robles, *Improving Fingerprint Verification Using Minutiae Triplets.* Sensors, 2012. **12**.
38. Ning Liu, Y.Y., Hongwei Zhang *A Fingerprint Matching Algorithm Based On Delaunay Triangulation Net*. in *The Fifth International Conference on Computer and Information Technology*. 2005: IEEE.
39. M. A. Medina Pérez, M.G.B., A. E. Gutierrez Rodriguez, L. Altamirano Robles, *Improving the multiple alignments strategy for fingerprint verification.* Computer Science, 2012. **7329**.
40. M. A. Medina Pérez, M.G.B., A. E. Gutierrez Rodriguez, L. Altamirano Robles, *Robust fingerprint verification using m-triplets*, in *International Conference on Hand-Based Biometrics (ICHB 2011)*. 2011.
41. [En_Linea]. *Journal Rankings*. 2012 [cited 2014; Available from: http://www.scimagojr.com/journalrank.php?category=1707.

Anexos

Anexo 1: Plantilla para especificación de Historias de Usuario.

Tabla. 29 Plantilla para la especificación de las HU. Elaboración propia.

Historia de Usuario		
ID:	**Nombre:**	
Descripción:		
Estimación aproximada:	**Prioridad:**	**H.U original:**
Observaciones:		

ID: número de identificación de la Historia de Usuario.

Nombre: Nombre de la Historia de Usuario.

Descripción: Breve descripción del proceso que define la historia.

Estimación aproximada: Duración aproximada (en días), de duración de la confesión de las tareas relacionadas con la Historia de Usuario.

Prioridad: Prioridad estimada de manera general de la Historia de Usuario.

H.U original: Si la presente Historia de Usuario es producto de la división de otra, aquí se pone el nombre de la Historia de Usuario original.

Observaciones: Alguna acotación importante a señalar acerca de la Historia de Usuario.

Anexo 2: Plantilla para la especificación de Iteraciones

Tabla. 30 Plantilla para la especificación de las iteraciones. Elaboración propia.

Iteración		
Numero:	**H.U (por orden):**	**Duración total:**
Descripción:		
Fecha de liberación:		

Número: Número de la Iteración.

H.U: Historias de Usuario ordenadas, a implementar en la iteración.

Duración total: Duración total en días de la iteración.

Descripción: Breve descripción del proceso que define la iteración.

Fecha de liberación: Especifica la fecha de la liberación.

Anexo 3: Plantilla para la especificación de las Tarjetas CRC.

Tabla. 31 Plantilla para la especificación de las Tarjetas CRC. Elaboración propia.

Clase:	
Responsabilidades:	**Colaboraciones:**

Clase: Nombre de la clase.

Responsabilidades: Funcionalidades que realiza la clase.

Colaboraciones: Clases que tienen relación con la clase de la tarjeta.

Anexo 4: Plantilla para la especificación de las Tareas de Historias de Usuario.

Tabla. 32 Plantilla para la especificación de las Tareas de Historias de Usuarios. Elaboración propia.

Tareas de Historia de Usuario		
ID:	**H.U:**	**Iteración:**
Nombre:		
Tipo:	**Puntos estimados:**	
Fecha inicio:	**Fecha Fin:**	
Responsable:		
Descripción:		

ID: Número de identificación de la tarea

H.U: Historia de Usuario a la que pertenece la tarea.

Iteración: Iteración a la que pertenece la tarea.

Nombre: Nombre de la tarea.

Tipo: Tipo de tarea.

Puntos estimados: Duración aproximada (en días), de la tarea.

Fecha de inicio: Fecha en la que comienza la tarea.

Fecha de fin: Fecha en la que termina la tarea.

Responsable: Nombre del miembro del equipo que realizará la tarea.

Descripción: Breve descripción del proceso que define la tarea.

Anexo 5: Capturas de pantalla de la interfaz de prueba del componente.

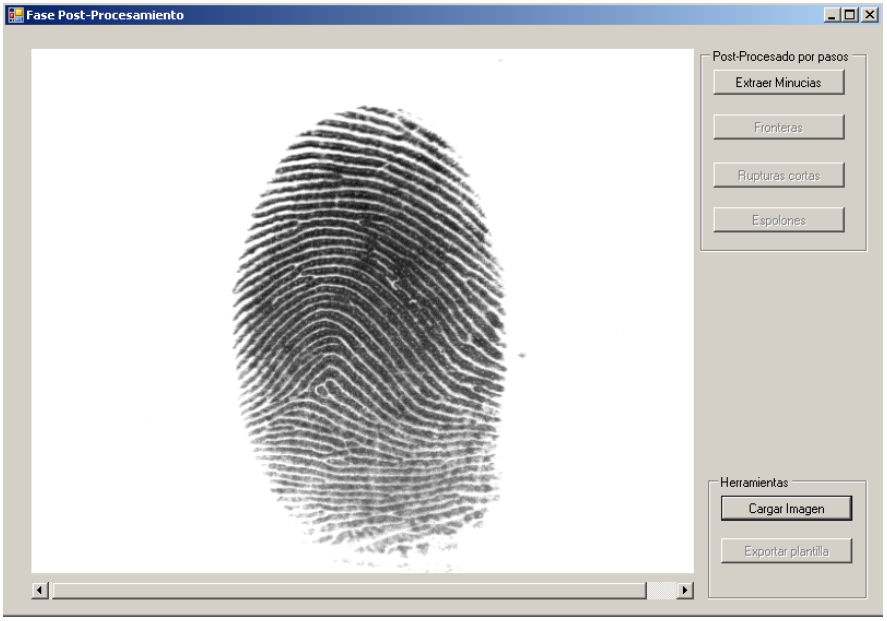


Figura. 28 Captura de pantalla: Cargar imagen. Elaboración propia.

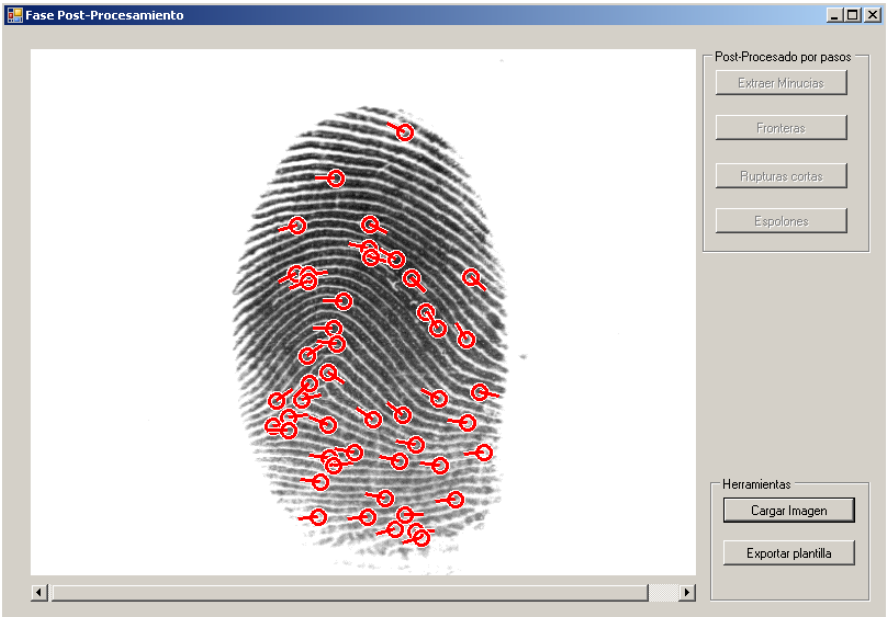


Figura. 29 Captura de pantalla: Post-procesamiento finalizado. Elaboración propia.

Anexo 6: Captura de pantalla de *framework Fingerprint Recognition* v2.2.

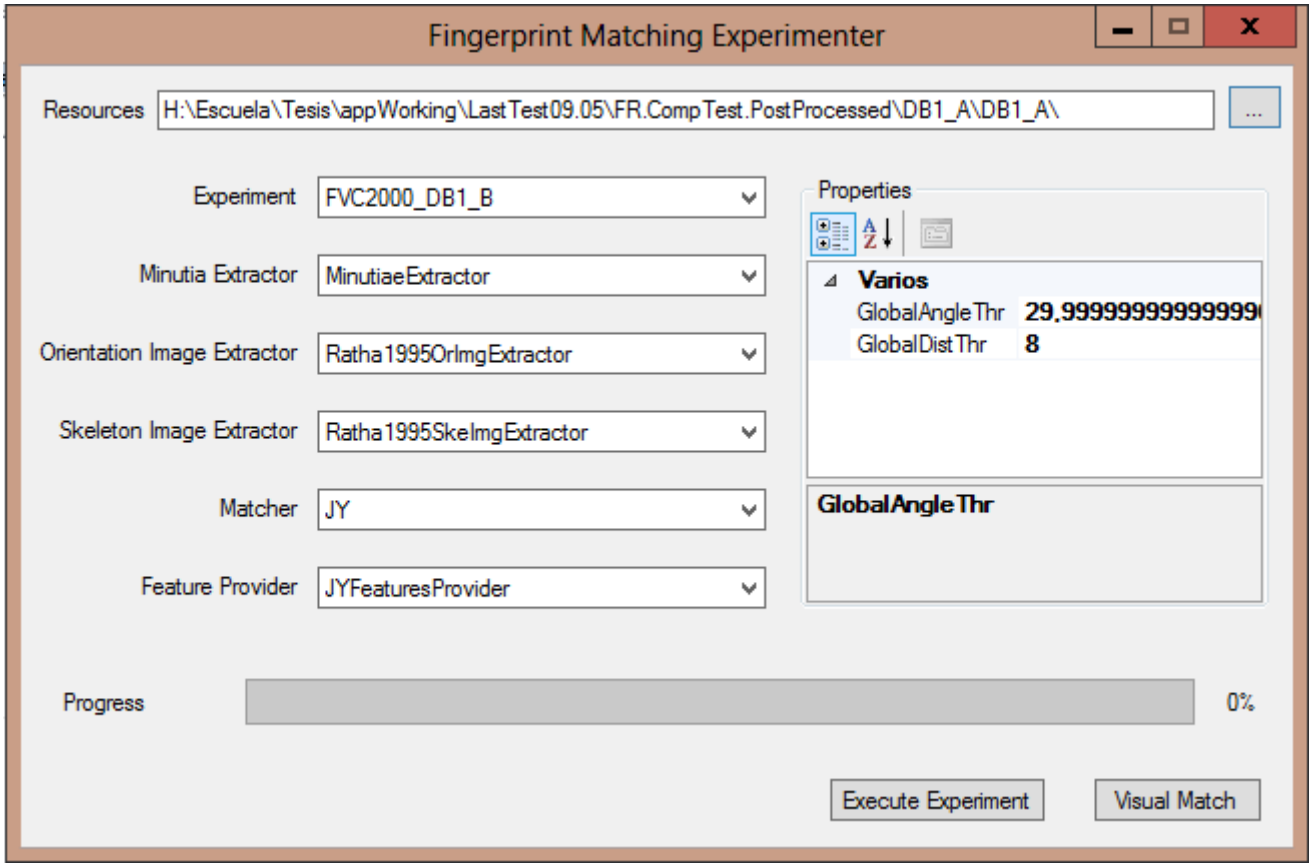


Figura. 30 Captura de pantalla: Interfaz de prueba del framework Fingerprint Recognition v2.2. Elaboración propia.

Anexo 7: Cálculo de la variable **Nivel Científico** usando datos recopilados de [41].

Tabla. 33 Datos necesarios para el cálculo de la variable Nivel Científico. Elaboración propia.

Método	Cantidad de citas (CC)	Cantidad de referencias (CR)	Factor impacto revista o publicador (FI)
Ratha1995[23]	580	20	Pattern Recognition (**2.362**)
Kim2001[20]	20	8	Springer-Verlag (**1**) – *Publicador*
Lu2002[21]	14	9	IEEE (**1**) – *Publicador*
Zhao2007[24]	62	22	Pattern Recognition (**2.362**)
Popovic2007[22]	4	19	Electronics and Electrical Engineering (**0.411**)
Bansal2010[19]	12	34	IJBB (**1.026**)

- NC (Ratha1995) = 5.8 + 2.362 + 0.2
  NC (Ratha1995) = 8.362
- NC (Kim2001) = 0.20 + 1 + 0.08
  NC (Kim2001) = 1.28
- NC (Lu2002) = 0.14 + 1 + 0.09
  NC (Lu2002) = 1.23
- NC (Zhao2007) = 0.62 + 2.362 + 0.22
  NC (Zhao2007) = 3.202
- NC (Popovic2007) = 0.04 + 0.411 + 0.19
  NC (Popovic2007) = 0.641
- NC (Bansal2010) = 0.12 + 1.026 + 0.34
  NC (Bansal2010) = 1.486

Anexo 8: Otros resultados expuestos en Bansal et al[19].

Tabla 1 Número total de bifurcaciones y terminaciones con su respectiva reducción de falsas minucias luego de cada etapa. Elaboración propia a partir de [19]-

Imagen	Minucias Fieles	Etapa	Total Minucias	Terminaciones	Bifurcaciones	Reducción (%)	Reducción Terminaciones	Reducción Bifurcaciones
I1	59(36+23)	Original	347	201	146	x	x	x
		Mejorada	315	180	135	9,2	10,4	7,5
		Pre-procesada	215	101	114	31,7	43,9	15,6
		Post-procesada	62	38	24	71,1	62,4	78,9
I2	23(12+11)	Original	114	75	39	x	x	x
		Mejorada	88	56	29	22,8	25,3	25,6
		Pre-procesada	61	42	19	30,7	25	34,5
		Post-procesada	23	11	12	62,2	73,8	36,8
I3	23(9+14)	Original	201	106	95	x	x	x
		Mejorada	73	30	43	63,7	71,7	54,7
		Pre-procesada	54	20	34	26	33,3	20,9
		Post-procesada	22	8	14	57,4	60	58,8
I4	32(13+19)	Original	360	110	250	x	x	x
		Mejorada	121	52	89	66,3	52,7	64,4
		Pre-procesada	76	24	52	37,2	53,9	41,6
		Post-procesada	33	14	17	56,6	41,7	67,3

Tabla. 34 Comparación del método empleado por Bansal et al[19] para la extracción de minucias con el de Número Cruzado (CN).Elaboración propia a partir de [19].

Etapa	Imagen	Método propuesto			Método Número Cruzado (CN)		
		Total	Terminaciones	Bifurcaciones	Total	Terminaciones	Bifurcaciones
Antes de post-procesado	I1	215	101	114	589	303	286
	I2	61	42	19	189	109	80
	I3	54	20	34	183	69	114
	I4	76	24	52	189	85	104
Luego de post-procesado	I1	62	38	24	145	67	78
	I2	23	11	12	38	24	14
	I3	22	8	14	78	26	52
	I4	33	14	17	41	15	26