SISTEMA DE AUTENTICACION DIGITAL PARA EL SERVICIO DE TAXI

UNIVERSIDAD PONTIFICIA BOLIVARIANA

Maestría en TIC– Seguridad Informática

Medellín

2015

SISTEMA DE AUTENTICACIÓN DIGITAL PARA EL SERVICIO DE TAXI

JUAN DAVID GÓMEZ GIRALDO

UNIVERSIDAD PONTIFICIA BOLIVARIANA
ESCUELA DE INGENIERÍAS
FACULTAD DE TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES
MAESTRÍA EN TIC
MEDELLÍN
2015

SISTEMA DE AUTENTICACIÓN DIGITAL PARA EL SERVICIO DE TAXI

JUAN DAVID GÓMEZ GIRALDO

Trabajo de grado para optar al título de Magister en TIC línea de Seguridad informática

Tutor
Álvaro Ospina San Juan
Magister en Software libre

Grupo de Investigación en Microelectrónica

UNIVERSIDAD PONTIFICIA BOLIVARIANA
ESCUELA DE INGENIERÍAS
FACULTAD DE TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES
MAESTRÍA EN TIC
MEDELLÍN
2015

Medellín, Agosto de 2015

Juan David Gómez Giraldo

“Declaro que esta tesis (o trabajo de grado) no ha sido presentada para optar a un título, ya sea en igual forma o con variaciones, en esta o cualquier otra universidad” Art 82 Régimen Discente de Formación Avanzada

Firma

Medellín, Agosto de 2015

Agradecimientos

A mi familia por brindarme su apoyo incondicional durante todo mi proceso de formación y las constantes voces de aliento que motivaron cada día el esfuerzo

A Dios por permitir que estos logros se presenten en mi vida guiados por su mano protectora.

Y en general a todas las personas que hicieron parte de este proyecto con sus aportes por pequeños que fueran generaron inquietudes que permitieron grandes cambios.

Una frase o idea puede cambiar la visión del mundo según la perspectiva desde donde se mire.

Contenido

Ilustraciones

Tablas

Ecuaciones

Resumen

En los últimos años ha aumentado el uso de las aplicaciones móviles en el servicio público, como una manera de aumentar la calidad y seguridad del servicio. Debido a los problemas de orden público regionales y nacionales surge la necesidad de brindar seguridad garantizando integridad y autenticidad de la información de los usuarios, tanto los usuarios como el transportador ahora requieren tener una mayor confiabilidad de quien ofrece y quien utiliza el servicio.

Para esto se plantea un método de verificación de identidad de dos vías que sea confiable para ambas partes, que ayude a la prevención de delitos como paseos millonarios, robos, estafas y extorsiones, entre otros, para este fin se desarrolló una aplicación llamada *SecurityP* basada en Android que permita la autenticación a través de la huella dactilar, utilizando una Tablet ubicada en el vehículo con el fin de poder verificar en línea la identidad tanto del conductor como del pasajero, advirtiendo la posibilidad de algún delito.

Para lograr este fin revisaremos los recursos utilizados como el dispositivo físico de lectura de huellas digitales, los algoritmos de identificación de las huellas, la topología de conexión utilizada, además de todo el proceso de diseño y desarrollo de la aplicación; y por último las pruebas realizadas que nos ayudarán a concluir factores importantes a futuro para trabajar en el proyecto productivo.

1. Introducción

Actualmente existen aplicaciones enfocadas al servicio de taxi que brindan facilidades para poder hacer uso de éste, pero muy pocas que brinden una seguridad desde ambas partes debido a la identificación directa.

Estas aplicaciones requieren de un Smartphone personal para la solicitud del servicio lo que hace que no todos los usuarios puedan acceder a estos aplicativos.

Entre algunas de las aplicaciones más utilizadas en el mercado y que han tenido mayor acogida en Colombia podemos nombrar a Easytaxi (1) y a Tappsi (2) por ejemplo, estás aplicaciones ofrecen el servicio de solicitud de taxi y brindan la facilidad de hacer el seguimiento del recorrido hasta el destino mediante una aplicación web integrada con el GPS del dispositivo, otras aplicaciones utilizadas, poco conocidas en Colombia pero reconocidas en otros países, entre ellas TaxiBeat (3) y Wannataxi (4), usan un registro inicial de los conductores, con la cédula, nombre y foto para que sea comparado por el usuario al momento de tomar el servicio, incluyen un temporizador, un localizador y la posibilidad de elegir el taxi a preferencia y calificar el servicio, sin embargo ninguna de las aplicaciones mencionadas anteriormente cuenta con una función que les brinde tanto a pasajeros como a conductores la posibilidad de un servicio confiable mediante la validación de la huella dactilar que garantice la identificación total de los usuarios, arriesgándose así a que ocurra una posible suplantación de identidad o de registros al tomar un taxi en la vía, por lo que se pretende realizar una prueba de concepto que permita establecer un método de verificación de identidad por medio de un lector de huellas digitales y así garantizar que cualquier taxi que implemente la utilización del servicio, nos permita llegar seguros a nuestro destino.

1.1. Planteamiento del problema

Durante los últimos años se han presentado en las diferentes ciudades del país y de Latinoamérica delitos relacionados con el servicio de taxi, casos como el secuestro exprés o también conocido paseo millonario, y/o hurtos a los conductores donde se ven involucrados muchos ciudadanos y que en algunos casos puede llegar hasta la muerte.

El secuestro exprés o paseo millonario se da esencialmente en las principales ciudades del país, según el DANE (5) corresponde a un 6.3% entre los tipos de hurto que se denuncian bajo la base de 2.054 miles de personas, esta modalidad es implementada por taxistas legalmente registrados, que seleccionan su víctima basados en la apariencia física, permitiendo así operar desde la normalidad y según el caso cometer el delito, lo anterior da cuenta de la falencia en las aplicaciones actuales, toda vez que cuando el servicio se toma en la vía pública, no se cuenta con registros de llamadas telefónicas que permitan hacer una validación aunque sea mínima de la información, aun cuando se realiza la llamada, se corre el riesgo de una suplantación o de omisiones que aumentan la inseguridad

Según las estadísticas de la Fiscalía General de la Nación cada día siete personas son víctimas de paseos millonarios solo en Bogotá (6) y cifras similares se manejan en las diferentes ciudades capitales del país, lo que hace necesario implementar un sistema que asegure un viaje confiable no solo al pasajero, sino también al conductor.

Las cifras dan cuenta de la creciente desconfianza e inseguridad que perciben los usuarios del servicio público de taxi, generando con ello una sensación de miedo, no solo por las pérdidas económicas que puede ocasionar ser víctimas de un paseo millonario, sino también por el temor de recibir lesiones físicas, que en el peor de los casos pueden conducir a la muerte. Según el DANE el 61% de usuarios se sienten inseguros en el transporte público (5), esta inseguridad manifestada en las estadísticas podría tener fundamento en algunas variables como un inadecuado control por parte de las autoridades, normas de seguridad deficientes, no contar con zonas de acopio cerca o un horario de servicio nocturno que facilita el accionar de los delincuentes.

Los métodos tradicionales de autenticación del conductor son físicos y pueden ser modificados fácilmente por los conductores o incluso quitados de la vista del usuario, y el control que se hace a tarjetas de los taxis es muy poca por parte de las autoridades, generando una desconfianza de parte de los usuarios hacia el conductor, ya que no existe ninguna forma que garantice la veracidad de la

información y las reales intenciones de los conductores, que en muchos casos son los que realizan los delitos de atracos o son cómplices del paseo millonario y de igual forma pero en menos cantidad el atraco a los conductores por parte de los pasajeros (7).

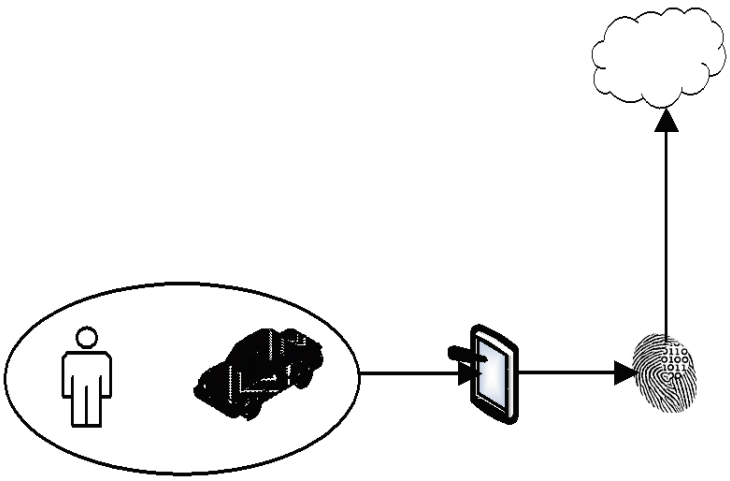


Ilustración 1: Sistema de autenticación por medio de huella digital.

Se pretende realizar una prueba de concepto que permita hacer una validación técnica para la autenticación de los usuarios del servicio de taxi, y de este modo combatir la inseguridad producto de paseos millonarios y robos en este servicio, esta autenticación se realizará mediante un dispositivo móvil asignado a el vehículo y a su vez a los diferentes conductores, de tal forma que se tenga una relación directa y así poder identificar el conductor y/o el pasajero, y establecer responsabilidades ante cualquier evento; el proceso de autenticación que se ejemplifica en la Ilustración 1 se daría por medio de huella digital garantizando los tres pilares de la seguridad de la información, integridad, confiabilidad y disponibilidad en el servicio, tanto el conductor como el usuario tendrían que autenticarse ante el aplicativo para que se garantice la legitimidad de la identidad y por tanto en el momento en que se presente el delito, tener la información necesaria para interponer la respectiva denuncia y con ello evitar la impunidad.

Después de un previo registro de la información, el sistema valida los datos del usuario en el servidor central a través del plan de datos o WIFI.

En caso de algún evento se contará con el registro de los usuarios, este registro incluye: nombre completo, cédula, correo electrónico y huella, está información se

podrá presentar como evidencia para reportar algún abuso o delito sin que se vea vulnerada la seguridad e individualizando el actor del delito.

Como valor agregado para el pasajero se pueden obtener los siguientes datos que serán enviados al correo electrónico suministrado en el momento de su registro: Datos del conductor y placa del vehículo, fecha y hora del servicio, posición inicial y final del recorrido.

Para la ejecución de este proyecto se debe tener claridad de algunos conceptos de tecnologías que se usarán para poder entender el desarrollo, además de tener claro un contexto de la problemática social y de las aplicaciones relacionadas existentes.

1.2. Objetivos:

Objetivo General

- Diseñar un método de verificación de identidad por medio de un lector de huellas digitales para aumentar la seguridad en el servicio de taxis.

Objetivos Específicos

- Establecer requisitos de seguridad fundamentales para diseñar la prueba de concepto.
- Implementar la prueba de concepto de verificación de identidad por medio de la huella digital.
- Realizar verificación necesaria para asegurar la funcionalidad del aplicativo.

2. Contextualización

Se revisarán los conceptos generales de la biometría y la identificación biométrica aplicada a la seguridad. También se harán apuntes acerca de las distintas técnicas de identificación biométrica en especial de la huella dactilar y el uso de Android, compilando el sistema de identificación por huella dactilar como medio de autenticación.

2.1. Biometría

El reconocimiento basado en Biometría o simplemente Biometría, es la ciencia de identificar o verificar la identidad de una persona fundamentada en sus características fisiológicas o de comportamiento (8); los rasgos fisiológicos están relacionados con la fisiología del cuerpo e incluyen principalmente las huellas dactilares, el rostro, el ADN, las orejas, el iris, la retina y la geometría de las manos y las palmas. Los rasgos de comportamiento están relacionados con el comportamiento de una persona y como ejemplos de esto se pueden mencionar la firma, la manera de mecanografiar, el modo de caminar, la voz etc. El reconocimiento biométrico ofrece muchas ventajas sobre el tradicional número de PIN o contraseña y los enfoques basados en fichas (como tarjetas de identificación entre otros). Un rasgo biométrico no puede ser transferido fácilmente, olvidado o perdido, el propietario legítimo de la plantilla biométrica puede ser fácilmente identificado y es difícil duplicar un rasgo biométrico (9). Existen un gran número de propiedades deseables para cualquiera de las características biométricas elegidas [4], entre ellas se tiene:

1. Universalidad: Cada persona debe poseer esa característica.
2. Unicidad: No deben haber dos personas iguales en términos de características biométricas.
3. Permanencia: Las características biométricas no deben cambiar ni siquiera en lo más mínimo con el pasar del tiempo.
4. Cobrabilidad: Las características biométricas deben ser medibles con algún dispositivo de medición (práctico).
5. Aceptabilidad: La población de usuarios y el público en general no deben tener fuertes objeciones en cuanto a la recolección y/o medición del rasgo biométrico.

Ahora bien, un sistema biométrico es esencialmente un sistema de reconocimiento de patrones que opera gracias a la adquisición de datos biométricos de un individuo,

extrayendo de estos datos un conjunto de características para después compararlas con un conjunto de plantillas en una base de datos ya existente (10). Dependiendo del contexto de la aplicación, un sistema biométrico puede operar tanto en *modo de verificación* como en *modo de identificación*.

- En el modo de verificación, un individuo provee sus datos biométricos y se identifica usualmente con un número PIN (aunque también puede ser mediante un nombre de usuario, una tarjeta inteligente etc.). Entonces el sistema verifica la identidad del individuo comparando los datos biométricos adquiridos contra una plantilla biométrica en la base de datos. La función de este sistema es básicamente hacer una comparación *uno a uno* para determinar si la identidad del sujeto es real o es mentira.
- En el modo de identificación, el sistema compara los datos biométricos dados con las plantillas de todos los usuarios registrados en la base de datos. Por lo que el sistema conduce una comparación *uno a muchos* para establecer la identidad del sujeto (o falla si el sujeto no se encuentra registrado en la base de datos), sin que este tenga que presentarse con una identidad.

La efectividad de un sistema biométrico puede ser juzgada por las siguientes características (11):

1. Rendimiento: Esto se refiere a la precisión alcanzable de reconocimiento, velocidad, robustez; la cantidad de recursos necesarios para alcanzar tanto la velocidad, como la precisión del reconocimiento deseado sin olvidar el ámbito operacional (esto involucra el ambiente laboral del individuo, por ejemplo los trabajadores de actividades manuales pueden tener un gran número de cortadas y/o heridas en las huellas dactilares) además se toman en cuento el ámbito ambiental (humedad, iluminación etc.) que afecta la precisión y la velocidad del reconocimiento.
2. Escalabilidad: Esta se refiere a la habilidad de abarcar un gran número de individuos sin que disminuya significativamente el desempeño.
3. No invasividad: Esto se refiere a la facilidad con la que la información proveniente de los individuos puede ser obtenida, sin perjudicar la integridad física e idealmente sin preparaciones especiales de estos.
4. Elusión: Esta se refiere al grado con el cual el sistema es resistente a los ataques.

Como se mencionó antes un sistema biométrico práctico debe reunir la precisión, velocidad y cantidad de recursos necesarios para el reconocimiento especificado, debe ser inofensivo para los usuarios, ser aceptado por la población que lo usará y

ser suficientemente robusto como para soportar métodos fraudulentos y ataques al sistema.

Huella dactilar como rasgo biométrico:

Una huella dactilar es una impresión de las crestas de fricción provenientes de la superficie en la punta de los dedos. Las huellas dactilares se han usado para identificación personal por muchas décadas, sin embargo recientemente han sido automatizadas gracias a los avances de las capacidades computacionales. El reconocimiento por huellas dactilares hoy en día es una de las más importantes y populares tecnologías biométricas principalmente por la inherente facilidad de adquisición, las numerosas fuentes (diez dedos) disponibles para recolección y las recolecciones y usos establecidos por agencias policiales. La identificación automática por huellas dactilares es una de las más confiables tecnologías biométricas, esto debido a la bien conocida distinción de huellas dactilares, persistencia, facilidad de adquisición y altas cifras de precisión de concordancia. Las huellas dactilares son únicas en cada individuo y no cambian con el paso del tiempo, incluso gemelos idénticos (que comparten el mismo ADN) poseen huellas dactilares diferentes. La unicidad puede ser atribuida al hecho de que los patrones de crestas y los detalles en las pequeñas áreas de crestas de fricción nunca se repiten. Estas crestas de fricción se desarrollan en el feto cuando se encuentra en su forma definitiva antes de nacer y son conocidas por ser persistentes en toda la vida a excepción de casos con cicatrices permanentes. Investigaciones científicas en áreas como biología, embriología, anatomía e histología verifican estos hallazgos (12); además la precisión de concordancia en los sistemas de autentificación basados en huellas dactilares ha probado ser bastante alta. Estos sistemas continúan dominando el mercado biométrico, contabilizando casi el 52% de los sistemas de autentificación basados en rasgos biométricos (11).

Historia del reconocimiento por huellas dactilares:

Las huellas dactilares han sido halladas en antiguos artefactos recuperados en los sitios de excavaciones de diferentes civilizaciones (13). Sin embargo, estas han sido utilizadas con motivo de identificación solo a partir del siglo IX; en la referencia (14) podemos hallar una línea de tiempo en donde se destacan importantes eventos que establecieron la fundación de la tecnología moderna biométrica basada en huellas

dactilares. Henry Fauld (15) ha sido el primero que científicamente sugirió la individualidad y la unicidad de las huellas dactilares; Sir Francis Galton ha publicado el bien conocido libro titulado *Huellas dactilares* (16) en el que se discute un detallado modelo estadístico de análisis e identificación de huellas dactilares. Galton ha introducido las características de nivel 2 definiendo puntos de minucia así como también finales de crestas o finales de bifurcaciones en una cresta local. Un importante avance en la identificación de huellas dactilares ha sido hecho por Edward Henry, quien ha establecido un sistema conocido como *"El sistema Henry"* utilizado para clasificar huellas dactilares (17).

En la referencia (18), Locard ha introducido la ciencia denominada *"Poroscopía"*, la comparación de los poros de sudor con el propósito de identificar a un individuo. Locard ha establecido que, al igual que las características de las crestas, los poros también son permanentes, inmutables y únicos por lo que son útiles para establecer la identidad especialmente cuando no se dispone de un número suficiente de crestas. Por otro lado Chatterjee ha propuesto el uso de los bordes de las crestas en combinación con otras formaciones de crestas de fricción para establecer la individualización, a la cual se refiere como *"Bordeoscopía*" (19).

En los últimos años la poroscopía y bordeoscopía han recibido una creciente atención y han sido ampliamente estudiadas por examinadores de huellas dactilares latentes (16). Se ha llamado la atención a que las formas y posiciones relativas de los poros de sudor y las formas de los bordes de cresta son tan permanentes y únicas como los puntos de minucia tradicionales y cuando estos son entendidos añaden un peso considerable a la conclusión de identificación (19).

Representación por huellas dactilares:

Los tipos de información que pueden ser recolectadas de la impresión de una cresta de fricción provenientes de una huella dactilar, pueden ser categorizadas como características de nivel 1, nivel 2 o nivel 3 así como se muestra en la Ilustración 2.

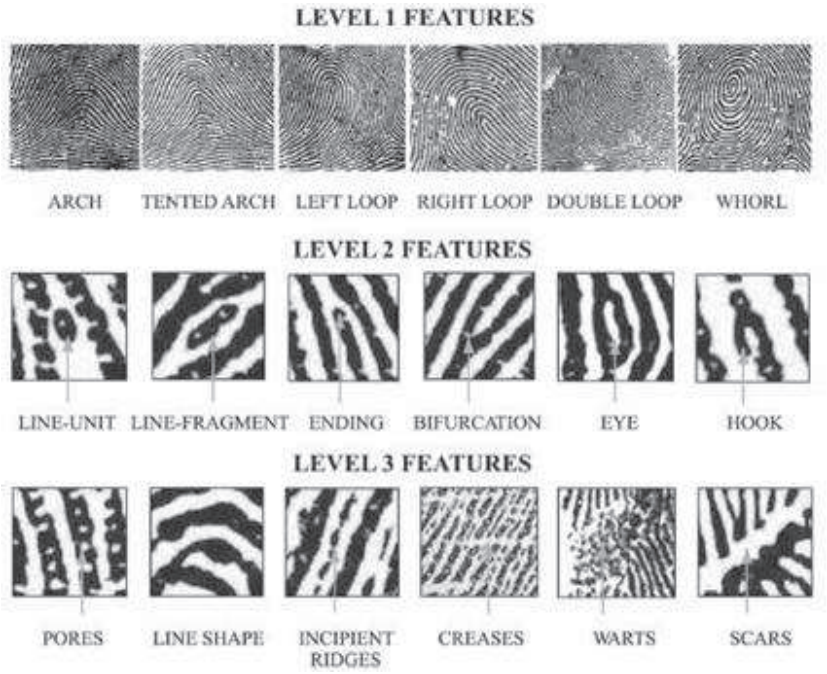


Ilustración 2: Ejemplos de las características Nivel I, II y III respectivamente. (20)

A nivel global, los patrones de huellas dactilares exhiben una o más regiones donde las líneas de cresta asumen formas distintivas caracterizadas por una alta curvatura, frecuente terminación etc. Estas regiones con ampliamente clasificadas como arco, bucle, espiral; y pueden ser además clasificadas en varias subcategorías.

- La características de nivel 1 comprenden esos patrones globales y la información morfológica; por si mismos no contienen suficiente información para identificar una huella dactilar de entre muchas, pero son usados para una amplia clasificación de ellas.
- Las características nivel 2 o *"minucias"* hacen referencia a las distintas formas en las que las crestas pueden ser discontinuas, son esencialmente *"Características de Galton"* nombradas como finales de cresta y bifurcaciones de cresta. Un final de cresta es definido como el punto en donde una cresta termina abruptamente, una bifurcación de cresta es definida como el punto en el cual una cresta se bifurca en otras dos. Las minucias son las características más prominentes generalmente estables y robustas para las condiciones de impresión de las huellas dactilares. La distribución de las minucias en una huella dactilar son consideradas únicas y la mayoría de las parejas automatizadas usan esta propiedad para identificar huellas dactilares.

La unicidad de las huellas dactilares basada en los puntos de minucia ha sido cuantificada por Galton (16), análisis estadísticos han mostrado que las características de nivel dos tienen suficiente poder discriminativo para establecer la individualidad de las huellas dactilares (21).

- Las características de nivel 3 hacen referencia a los detalles extremadamente finos dentro de las crestas presentes en las huellas dactilares (22); son esencialmente los poros de sudor y los contornos de las crestas. Los poros son las aberturas de las glándulas de sudor y están distribuidos a lo largo de las crestas; los estudios (19) han mostrado que la densidad de los poros en una cresta varía entre 25 y 45 poros por pulgadas y entre 20 y 40 poros deberían ser suficientes para determinar la identidad de un individuo. Un poro puede estar tanto abierto como cerrado dependiendo de su actividad de transpiración, un poro cerrado se encuentra enteramente encerrado por una cresta, mientras un poro abierto se intersecta con un valle ubicado entre dos crestas así como se muestra en la Ilustración 3. La información del poro (posición, número y forma) es considerada permanente, inmutable y altamente distintiva sin embargo muy pocas técnicas de concordancia automáticas usan a los poros desde que la extracción confiable de esta información requiere una alta resolución y calidad de las imágenes de las huellas dactilares. Los contornos de cresta contienen información valiosa de nivel tres incluyendo el ancho de la cresta y la forma del borde. Varias formas de los bordes de las crestas de fricción pueden ser a su vez clasificadas en ocho categorías, estas son: rectitud, convexión, picos, mesa, bolsillo, concavidad, ángulo entre otras; se muestran en la Ilustración 2. Las formas y posición relativa de las crestas de fricción son consideradas tanto permanentes como únicas.

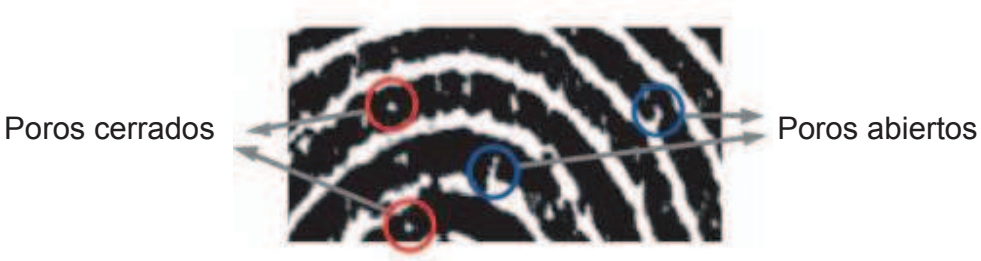


Ilustración 3: Poros cerrados y abiertos. (40)

2.2. LEVEL3

La siguiente descripción de la modalidad preferida es de naturaleza meramente ejemplar y no está intencionada de ninguna manera a limitar la invención, su aplicación o usos:

De acuerdo con un aspecto de la presente invención, las características de Nivel 1, Nivel 2 y Nivel 3 en una imagen de una huella dactilar estaban mutuamente correlacionadas entre sí. Por ejemplo, la distribución de los poros no fue al azar, pero siguió naturalmente la estructura de las crestas. Además, sobre la base de la fisiología de la huella dactilar, los poros solo estuvieron presentes en las crestas, no en los valles. Por lo tanto, fue esencial que la ubicación de las crestas fuera localizada antes de la extracción de los poros. Además de los poros, los contornos de las crestas también se consideraron como información de Nivel 3. Durante la adquisición de imágenes, se observó que el contorno de las crestas fue a menudo más fiable conservado a 1000 ppi que los poros, especialmente en la presencia de las diversas condiciones de la piel y el ruido del sensor. Con el fin de extraer automáticamente las características de Nivel 3, la presente invención proporciona los algoritmos de extracción de características utilizando, entre otras cosas, los filtros de Gabor y las transformadas de Wavelet. También debe apreciarse que la presente invención puede practicarse con imágenes que incluyen menos de o más de 1000 ppi. (23)

Para mejorar las crestas, los filtros de Gabor que se utilizaron, tienen la forma establecida en el algoritmo de la Ecuación 1 a continuación:

$$G(x, y : \theta, f) = \exp\left\{-\frac{1}{2}\left[\frac{x_\theta^2}{\delta_x^2} + \frac{y_\theta^2}{\delta_y^2}\right]\cos(2\pi f x_\theta)\right\}$$

Ecuación 1

Dónde θ y f son la orientación y la frecuencia del filtro, respectivamente, y δ_x y δ_y son las desviaciones estándar de la envolvente Gaussiana a lo largo de los ejes x e y respectivamente. Aquí, (x_θ, y_θ) representan la posición de un punto (x,y) después de que ha sido rotado en sentido horario por un ángulo (por ejemplo, 90°). Los cuatro parámetros (es decir θ, f, δ_x y δ_y) del filtro Gabor fueron determinados empíricamente basados en la frecuencia y la orientación de la imagen de la huella dactilar. Un ejemplo no limitante de una imagen de la huella dactilar que ha sido mejorada después de la filtración Gabor se muestra en la Ilustración 4. Está claro que las crestas estaban bien separadas de los valles después de la mejora.

Ilustración 4: Filtración de Gabor.

El procedimiento anterior suprime el ruido rellenando todos los huecos (o poros) en las crestas y destacando sólo las crestas. Simplemente añadiéndoselo a la imagen de la huella original, se observó que los poros abiertos y cerrados se conservan, ya que sólo aparecen en las crestas (Ilustración 5). Sin embargo, el contraste entre los poros y las crestas fue bajo en la Ilustración 5. A fin de mejorar la imagen original con respecto a los poros, se empleó un filtro de paso de banda para capturar la alta respuesta de frecuencia negativa pues los valores de la intensidad en los poros cambian abruptamente del blanco al negro. La transformada de Wavelet es conocida por sus propiedades altamente localizadas en las frecuencias y dominios espaciales. Por lo tanto la transformada Wavelet del sombrero mexicano fue aplicada a la imagen de entrada f(x, y) Є R^2 para obtener la respuesta de frecuencia W, como se establece en el algoritmo de la Ecuación 2 a continuación:

$$w(s, a, b) = \frac{1}{\sqrt{s}} \int \int_{R^2} f(x, y)\phi\left(\frac{x-a}{s}, \frac{y-b}{s}\right) dx dy,$$

Ecuación 2

Ilustración 5: Poros en crestas.

Donde s es el factor de escala (: 1.32) y (a, b) son los parámetros cambiantes. En esencia, este Wavelet fue un filtro de paso de banda con escalas. Después de normalizar la respuesta del filtro (por ejemplo, 0-255) usando la normalización mínimo-máximo, las regiones de los poros que suelen tener alta respuesta de frecuencia negativa estuvieron representados por pequeñas gotas con bajas intensidades (Ilustración 6) Mediante la adición de las respuestas de los filtros de Gabor y Wavelet, la mejora "óptima" de los poros se obtuvo debido a la restricción de que los poros se encuentran sólo en las crestas (Ilustración 7). Por último, un umbral determinado empíricamente se aplicó para extraer los poros con un tamaño de gota de menos de 40 píxeles. (Un ejemplo de extracción de los poros se muestra en la Ilustración 8), donde aprox. 250 poros, tanto abiertos como cerrados, se extrajeron con precisión a lo largo de las crestas.

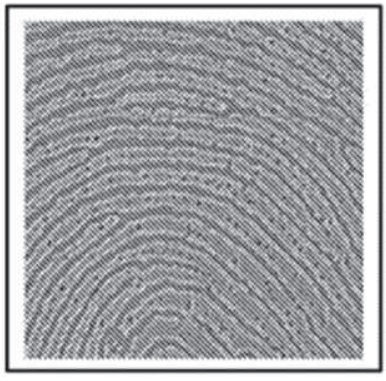


Ilustración 6: Huella 1.

Ilustración 7: Huella 2.

Ilustración 8: Huella 3.

El algoritmo para extraer los contornos de las crestas se puede describir de la siguiente manera. En primer lugar, la imagen se mejora usando los filtros de Gabor como en la ecuación 1**.** Entonces, la transformada Wavelet se aplica a la imagen de

la huella para mejorar los contornos de las crestas (Ilustración 9). Es necesario hacer notar que la escala en la ecuación 2 se aumentó a 1,74 con el fin de acomodar la variación de la intensidad de los contornos de la cresta. La respuesta Wavelet se extraerá de la imagen mejorada de Gabor para que los contornos de la cresta se mejoraran más (Ilustración 10). La imagen resultante se binariza mediante un umbral definido empíricamente δ(=10).

Ilustración 9: Huella 5.

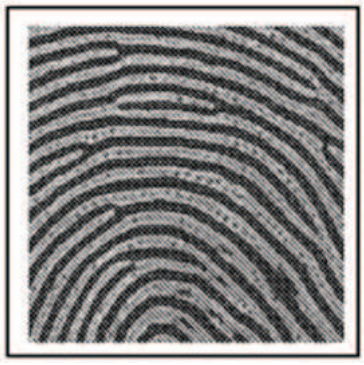


Ilustración 10: Huella 4.

Por último, los contornos de las crestas pueden ser extraídos mediante la convolución de la imagen binarizada $f^b(x, y)$ con un filtro H, dado por el algoritmo en la ecuación 3, a continuación en donde el filtro H: (0, 1, 0; 1, 0, 1; 0, 1, 0) cuenta el número de puntos de borde vecinos para cada píxel. Un punto (por ejemplo x, y) se clasifica como un punto de contorno de cresta si r(x, y)= 1 o 2. Ilustración 11 muestra los contornos de crestas extraídos.

$$r(x,y)=\Sigma_{n,m} f^b(x,y)H(x-n,\ y-m)$$

Ecuación 3

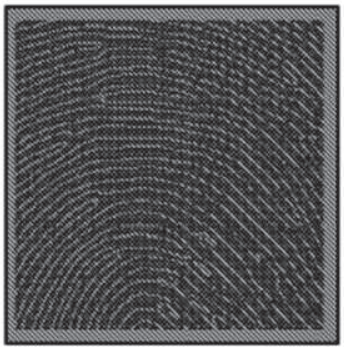


Ilustración 11: Huella 6.

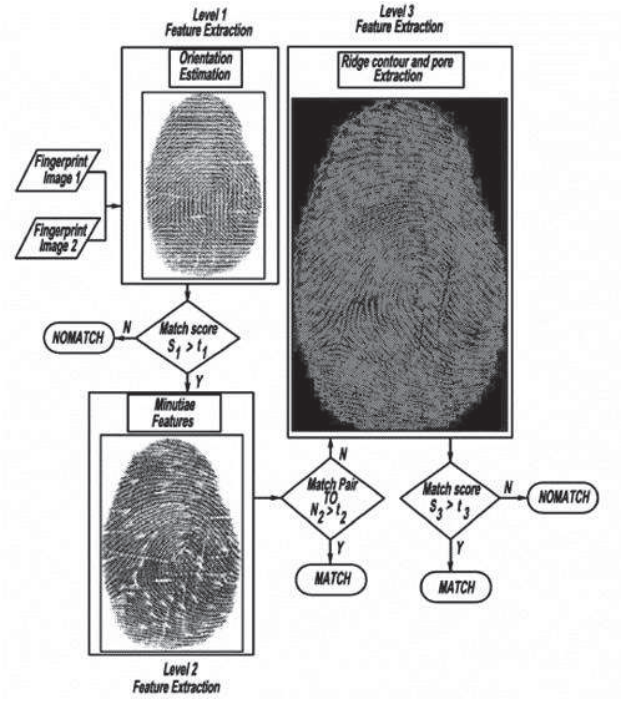


Ilustración 12: Proceso Level3.

La Ilustración 12 ilustra el diseño arquitectónico del sistema de concordancia propuesto de la presente invención. Cada capa en las características utilizadas en el sistema con su nivel correspondiente. Todas las características que se utilizaron en el sistema se muestran en la Ilustración 13.

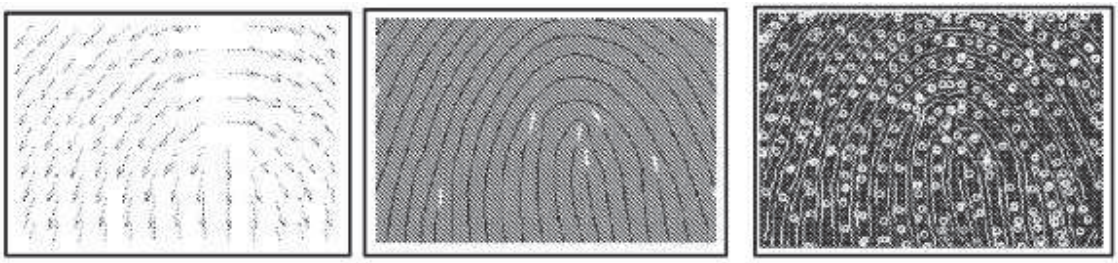


Ilustración 13: Características según niveles.

Dadas dos imágenes de huellas dactilares, el sistema extrae las características de nivel 1 (por ejemplo, orientación de campo) y Nivel 2 (por ejemplo, los puntos característicos) y establece la alineación de las dos imágenes usando un algoritmo de comparación de cadena. El acuerdo entre los campos de orientación de las dos imágenes fue entonces calculado utilizando el producto punto. Si los campos de orientación no estuvieron de acuerdo (por ejemplo, $S_1 < t_1$), el comparador rechazó la consulta y se detuvo en el Nivel 1. De lo contrario, el comparador procedió al Nivel 2, donde la correspondencia entre puntos característicos se establecieron utilizando cuadros delimitadores y el resultado de la puntuación de comparación S_2 se calculó de acuerdo con el algoritmo en la ecuación 4, a continuación:

$$S_2 = w_1 \times S_1 + w_2 \times \frac{1}{2}\left(\frac{N_2^{TQ} - 0.20 \times \left(N_2^{T} - N_2^{TQ}\right)}{N_2^{T} + 1} 1 + \frac{N_2^{TQ} - 0.20 \times \left(N_2^{Q} - N_2^{TQ}\right)}{N_2^{Q} + 1}\right)$$

Ecuación 4

Cuando w_1 y w_2 (=1-w_1) son ponderados combinando la información en el Nivel 1 y Nivel 2, N_2^{TQ} es el número de puntos característicos coincidentes y N_2^{T}; y N_2^{Q} son el número de puntos característicos dentro de la región de superposición de la plantilla (T) y la consulta (Q), respectivamente. Tenga en cuenta que se requiere que $0 \leq S_2 \leq 100$. A continuación, el umbral T_2 se fijó en 12, de tal manera que si $N_2^{TQ} > 12$, la identificación positiva en muchos tribunales de justicia, la comparación terminada en el nivel 2 y la puntuación final de comparación permanezca como S_2. De lo contrario, la investigación de las características de nivel 3 continuó. El umbral T_2 fue elegido en base a la directriz de 12 puntos que se consideró como evidencia suficiente para hacer una toma de identificación positiva en muchos tribunales de justicia.

A medida que la comparación procedía al Nivel 3, los puntos característicos coincidentes en el Nivel 2 fueron examinados en el contexto de la vecindad de las características de Nivel 3. Por ejemplo, dado un par de puntos característicos coincidentes, las características de Nivel 3 se compararon en la vecindad y se volvió

a calcular la correspondencia basada en los acuerdos de las características de Nivel 3. Asumiendo que se haya establecido una alineación en el Nivel 2, (x_i, y_i), para i= 1, 2,.. N_2^{TQ} fue la locación de los i-ésimos puntos característicos comparados y (x_i, y_i) fue la ubicación principal de todos los puntos característicos coincidentes. La región asociada de cada punto característico comparado (x_i, y_i) se definió como una ventana rectangular R_I. Con tamaño 60xl20 centrada en la ecuación 5:

$$\frac{x_i + \overline{x}}{2}, \frac{y_i + \overline{y}}{2}$$

Ecuación 5

Cabe señalar que es posible que los puntos característicos estén fuera de su región asociada, pero la selección de estos aseguró una región suficientemente grande de primer plano para la extracción de características de Nivel 3.

Con el fin de comparar las características de Nivel 3 en cada región local, el hecho de que el número de características detectadas (por ejemplo los poros y los puntos de contorno de las crestas) debían tenerse en cuenta; en la práctica, sería diferente entre la consulta y la plantilla, debido a la degradación de la calidad de la imagen (por ejemplo, la deformación de la piel). El algoritmo Iterativo de Punto más Cercano (ICP) fue una buena solución para este problema, ya que apunta a minimizar las distancias entre puntos en una imagen a entidades geométricas (en lugar de puntos) en el otro sin requerir una correspondencia de 1:1. Otra ventaja del ICP fue que cuando se aplica localmente, proporciona la corrección de alineación para compensar la deformación no lineal, asumiendo que la estimación inicial de la transformación era razonable.

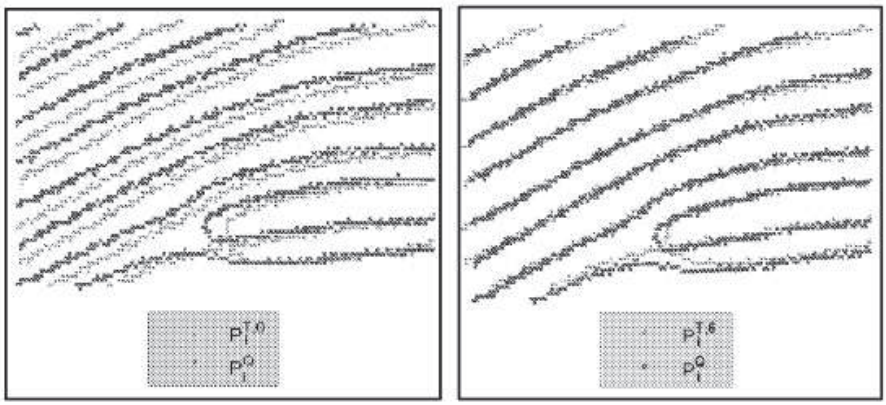


Ilustración 14: Corrección de alineación.

Para cada conjunto de puntos característicos combinados (x_i, y_i), para i= 1, 2,… N_2^{TQ} , las regiones asociadas desde T y Q fueron definidas para ser R_i^T y R_i^Q, respectivamente, y las característica de nivel 3 extraídas establece P_i^T: ($a_{i,j}$, $b_{i,j}$, $t_{i,j}$), para J=l, 2,… $N_{3,i}^T$ y P_i^Q: ($a_{i,k}$, $b_{i,k}$, $t_{i,k}$), para k= 1, 2,…. $N_{3,i}^Q$, en consecuencia. Cada conjunto de características incluye tripletas que representan la ubicación de cada punto de característica y su tipo (por ejemplo, los poros o los contornos de los puntos de cresta). Cabe señalar que los poros que coincidan con los puntos de contorno de crestas fueron evitados. Los detalles de la comparación de cada conjunto de características de Nivel 3 P_i^T y P_i^Q utilizando el algoritmo ICP (Ilustración 14) se dan a continuación:

1. Inicializar el índice de iteración K – 0;
2. Inicializar $P_i^{TO} = P_i^T$ y la transformación rígida W_i^O = I;
3. Inicializar indicador de convergencia Diff = 10^{10};
4. Establecer el criterio de distancia para la parada D = 0.03;
5. Establecer el criterio de parada para la iteración Itr = 15;
6. While (Diff > D)

{

6.1. If (k ≥ Itr) break;
6.2. k = k + I
6.3. aplicar W_i^{k-1} a la consulta $P_i^{T,k} = W_i^{k-1} P_i^{T,k-1}$;
6.4. For (j = 1 to $N_{3,i}^Q$)

{

Encuentra índice del punto más cercano para ($a_{i,j}^Q$, $b_{i,j}^Q$, $t_{i,j}^Q$);

$C^k(j) = \text{argmin}_g (d((a_{i,g}^{T,k}, b_{i,g}^{T,k}, t_{i,g}^{T,k}), (a_{i,j}^Q, b_{i,j}^Q, t_{i,j}^Q)))$;

g = 1, 2, … , $N_{3,i}^Q$

}

6.5. Comprueba la distancia principal entre $P_i^{T,k}$ y P_i^Q

$$E_i^k\left(P_i^{T,k}, P_i^Q\right) = \frac{1}{N_{3,i}^Q} \sum_{j=1}^{N_{3,i}^Q} d\left(\left(a_{i,C^k(j)}^{T,k}, b_{i,C^k(j)}^{T,k}, t_{i,C^k(j)}^{T,k}\right), \left(a_{i,j}^Q, b_{i,j}^Q, t_{i,j}^Q\right)\right);$$

6.6. Obtener la nueva transformación W_i^k que minimice E_i^k;

6.7. Estimar la convergencia de iteración k
Diff = E_i^k ($P_i^{T,k}$ y P_i^Q) - E_i^{k-1} ($P_i^{T,k-1}$ y P_i^Q);

7. Obtener la distancia de coincidencia $E_i = E_i^k$ ($P_i^{T,k}$, P_i^Q)

La transformación inicial W_i^O en el Paso 2 se fijó de igual modo tanto a la matriz identidad I como a P_i^T y había sido pre-alineada en el nivel 2. En los pasos 6.4 y 6.5, d(.,.) denota la distancia euclidiana entre los conjuntos de puntos, cabe señalar que la ICP requiere $N_{3,i}^Q$, el número de características de nivel 3 en la región de consulta R_i^Q, siempre será menor que $N_{3,i}^T$, el número de características de Nivel 3 en R_i^T. Esto podría satisfacerse mediante la elección del conjunto de características con el tamaño más grande para ser la plantilla. La rápida convergencia del algoritmo ICP generalmente estaba asegurada porque la alineación inicial basada en puntos característicos en el Nivel 2 en general era buena. Cuando el algoritmo convergía o terminaba cuando k= 15, la distancia de comparación E se obtenía.

Dada N_2^{TQ} igualada a los puntos característicos entre T y Q en el Nivel 2, N_2^{TQ} compara las distancias E_i, para un i= 1, 2,… N_2^{TQ} basado en las características de Nivel 3 que se obtuvieron utilizando el algoritmo anterior. Cada distancia E_i, se compara con un umbral t_d y si $E_i < t_d$, la correspondencia de los puntos característicos asociados es asegurada, de lo contrario, la correspondencia es negada. $N_{2,3}^{TQ}$ fue el número actualizado de los puntos coincidentes, $N_{2,3}^{TQ} \leq N_2^{TQ}$ (Ilustración 15) La puntuación de coincidencia S_3 se definió de acuerdo con el algoritmo establecido en la ecuación 6, a continuación:

$$S_3 = w_1 \times S_1 + w_2 \times \frac{1}{2}\left(\frac{N_{2,3}^{TQ} - 0.20 \times \left(N_2^T - N_{2,3}^{TQ}\right)}{N_2^T + 1} 1 + \frac{N_{2,3}^{TQ} - 0.20 \times \left(N_2^Q - N_{2,3}^{TQ}\right)}{N_2^Q + 1}\right)$$

Ecuación 6

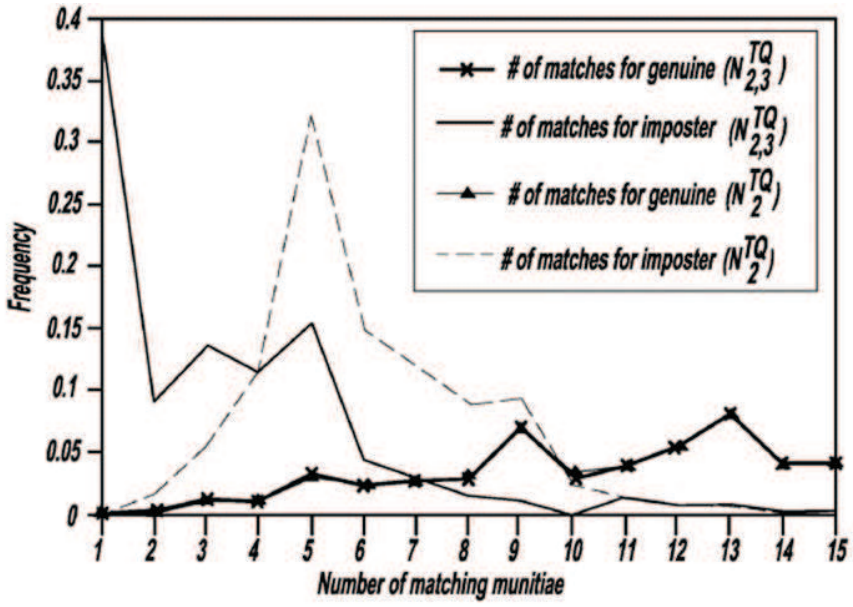


Ilustración 15: Coincidencias.

En donde N_2^T y N_2^Q como antes, son el número de puntos característicos dentro de la región solapada de la plantilla y de la consulta respectivamente. Cabe señalar que $0 \leq S_3 \leq 2100$.

El comparador jerárquico propuesto utiliza un esquema de fusión que integra la información de las características en el Nivel 2 y Nivel 3 en forma de cascada. Un enfoque alternativo que integra las puntuaciones de coincidencia en el Nivel 2 y Nivel 3 de forma paralela también se propuso, en donde se emplea la normalización min-máx y la suma de fusionar las puntuaciones de coincidencia. Aunque este último es un enfoque más directo y es el más comúnmente usado, es más lento porque la comparación tanto del Nivel 2 como del Nivel 3 deberá realizarse para cada consulta. Además, la fusión de puntajes paralelos es sensible al esquema de normalización seleccionado y a la regla de fusión. Por otro lado, el comparador jerárquico propuesto permite que la presente invención controle el nivel de información, o características que serán usadas en diferentes etapas de comparación de huellas dactilares.

En resumen, la presente invención proporciona un sistema de comparación de huellas dactilares automatizado que utiliza características de huellas dactilares en imágenes de 1000 ppi en los tres niveles. Para obtener información discriminatoria en el nivel 3 se proporcionaron los algoritmos basados en filtros de Gabor y transformadas wavelet, para así extraer automáticamente los poros y los contornos de las crestas. Un algoritmo ICP modificado fue empleado para hacer coincidir las características de Nivel 3. Los resultados experimentales demostraron que las

características de Nivel 3 deben ser examinadas para refinar el establecimiento de puntos de coincidencias proporcionados en el Nivel 2. Más importante aún, las ganancias de rendimiento consistentes también se observaron tanto en imágenes de alta calidad e imágenes de baja calidad, lo que sugiere que la extracción automática de características de Nivel 3 puede ser informativa y robusta, especialmente cuando la región de huellas dactilares, o el número de características de nivel 2, es pequeño.

2.3. MINDTCT

Los algoritmos utilizados en MINDTCT fueron inspirados por el Sistema de Reconocimiento de Huellas Dactilares Automático del Ministerio del Interior; específicamente, este conjunto de algoritmos se conoce comúnmente como "HO39." (24) El software NIST es una implementación totalmente original superior a las capacidades de HO39. Incorpora nuevos algoritmos, un diseño modular, la asignación dinámica y control de parámetros flexibles que proporcionan un marco de apoyo a la mejora y adaptación de la tecnología del futuro. Cabe señalar que los algoritmos y parámetros del software se han diseñado y puesto a procesar las imágenes escaneadas en 19,69 pixeles por milímetro (ppmm) (500 píxeles por pulgada) y cuantificadas a 256 niveles de gris de manera óptima.

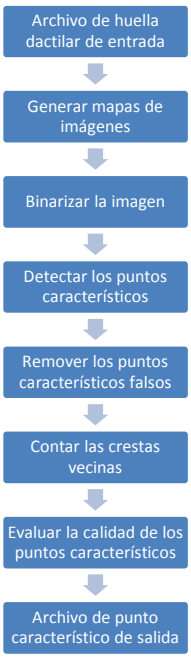


Ilustración 16: Proceso MINDTCT.

Una vez que el software está exitosamente instalado y compilado, el programa, MINDTCT está disponible para la detección de puntos característicos de una imagen de huella dactilar. En esta sección se describe cada uno de los pasos más importantes en el proceso de detección de puntos característicos. A MINDTCT se le conoce comúnmente como "*rutinas de segunda generación*" (o versión 2). Las rutinas versión 1 se incluyen en las bibliotecas para la comparación, pero en

general, su desempeño será menos satisfactorio. La Ilustración 16 enumera los pasos funcionales ejecutados.

El software ha sido diseñado de forma modular de manera que cada uno de los pasos que se indican en la Ilustración 16 se ejecuta principalmente por una sola subrutina. Esto permite que otros enfoques alternativos sean implementados y sustituidos en el proceso y el impacto global sobre el rendimiento pueda ser evaluado. Para apoyar a los muchos parámetros de funcionamiento necesarios, una estructura única de control global se utiliza para registrar tamaños, tolerancias y umbrales.

Archivos de entrada de imágenes de huellas dactilares:

Mindtct introduce una imagen de la huella dactilar y automáticamente detecta puntos característicos de esta. Los algoritmos y parámetros se han desarrollado con el fin de buscar imágenes digitalizadas en 19.69 ppmm y cuantificados en 256 niveles de gris. En archivos con formato ANSI/NIST se busca la estructura de archivos de un registro de huellas dactilares en escala de grises. Una vez encontrado, la imagen de la huella en este registro se procesa. La aplicación es capaz de procesar registros tipo-4 ANSI/NIST, tipo-13 y tipo-14 de imágenes de huellas dactilares. (25) En la actualidad, sólo se procesa el primer registro de huella dactilar en escala de grises del archivo ANSI/NIST, pero la aplicación podría ser cambiada para procesar todas las huellas dactilares en escala de grises del archivo ANSI/NIST.

Mindtct tiene una opción que le permitirá mejorar las imágenes de muy bajo contraste. Si se selecciona la opción, Mindtct evaluará el histograma de la imagen de entrada; si la imagen es de contraste muy bajo, será optimizada para mejorar el contraste, de lo contrario no se modificará.

Generar mapas de calidad de imagen:

Debido a que la calidad de la imagen de una huella dactilar puede variar, especialmente en el caso de huellas dactilares latentes, es crítico ser capaz de analizar la imagen y determinar las áreas que se degradan y que puedan causar problemas. Se pueden medir varias características que están diseñadas para transmitir información sobre la calidad de regiones localizadas en la imagen. Estas incluyen la determinación de la dirección del flujo de las crestas en la imagen y además la detección de las regiones de bajo contraste, el bajo flujo de bordes y la alta curvatura. Estas tres últimas condiciones representan áreas inestables en la imagen donde la detección de los puntos característicos no es fiable, y juntas se

pueden utilizar para representar los niveles de calidad en la imagen. Cada una de estas características se discute a continuación.

Mapas de dirección:

Uno de los pasos fundamentales en este proceso de detección de puntos característicos es derivar un mapa direccional de flujo de crestas, o "*mapa de dirección*". El propósito de este mapa es representar las áreas de la imagen con suficiente estructura de cresta; crestas bien formadas y claramente visibles son esenciales para detectar de forma fiable los puntos de extremo y bifurcación de estas. Además, el mapa de dirección registra la orientación general de las crestas a medida que fluyen a través de la imagen.

Para analizar la huella dactilar localmente, la imagen se divide en una cuadrícula de bloques. A todos los píxeles dentro de un bloque se le asignan los mismos resultados. Por lo tanto, en el caso del mapa de dirección, a todos los píxeles de un bloque les serán asignados la misma dirección del flujo de crestas. Ahora bien, se deben hacer varias consideraciones cuando se utiliza un enfoque basado en bloques; en primer lugar, debe determinarse cuanta información local se requiere para derivar de forma fiable la característica deseada. Esta área se conoce como "*la ventana*". La característica medida dentro de la ventana se asigna a cada píxel del bloque. Es típicamente deseable compartir datos utilizados para calcular los resultados asignados a los bloques vecinos. De esta forma parte de la imagen que ha contribuido a los resultados de un bloque se incluye en los resultados del bloque vecino también. Esto ayuda a minimizar la discontinuidad en los valores de los bloques cruzando la frontera de un bloque a su vecino. Este "*suavizado*" se puede implementar utilizando un sistema en el cual un bloque es más pequeño que la ventana que lo rodea, y las ventanas se superponen de un bloque a otro. Esto se ilustra en la Ilustración 17

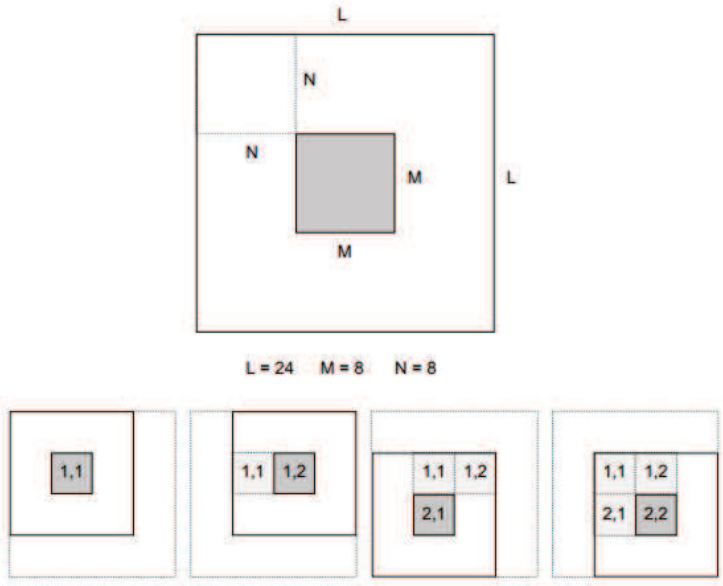


Ilustración 17: Suavizado por bloques.

El gran marco en la parte superior de la Ilustración representa una ventana (en blanco) que rodea a un bloque más pequeño (en gris). Suponiendo que los bloques vecinos son adyacentes y no se solapan, este escenario está definido por tres parámetros: *el tamaño de la ventana* "L", *el tamaño de bloque* "M" y *el desplazamiento* del bloque de origen de la ventana "N." En la estructura de control global, lfsparms_V2, estos parámetros se definen como “MAP_WINDOWSIZE_V2 = 24”, “MAP_BLOCKSIZE_V2 = 8”, y “MAP_WINDOWOFFSET_V2 = 8” respectivamente. Como resultado, la imagen se divide en una cuadrícula de bloques de píxeles de 8 × 8, a cada bloque se le asigna un resultado de una ventana más grande de 24 × 24 píxeles circundantes, y el área para las ventanas de bloques vecinos se superponen hasta 2/3.

La fila inferior de cuadros en la Ilustración 17 ilustra cómo funciona esto en la práctica. Designando la dirección de un bloque por sus índices (fila, columna), se calcula el marco de la izquierda que muestra el primer bloque (1,1). La siguiente trama avanza al siguiente bloque adyacente a la derecha, bloque (1,2). Correspondientemente, la ventana se desplaza 8 píxeles, y el nuevo bloque recibe sus resultados. Se debe tener en cuenta que hay dos copias de la imagen que se utiliza. Cada ventana opera sobre los datos originales de la imagen, mientras que los resultados de los bloques se escriben en una imagen de salida separada. El tercer fotograma de la ilustración muestra la configuración de la ventana para el bloque (2,1), y la cuarta trama muestra a su vecina de la derecha siendo calculada.

Se debe hacer una consideración adicional si se utilizan bloques, esta es determinar cómo manejar los bordes de la imagen. Las dimensiones de la imagen

probablemente no serán un múltiplo par de bloques y las ventanas circundantes de los bloques a lo largo del perímetro de la imagen se pueden extender fuera de esta. En este programa, la imagen se rellena por un margen de píxeles grises medianos (ajustados a la intensidad 128). Este margen es suficientemente grande para contener las ventanas perimetrales en la imagen. El procesamiento de los bloques parciales también se contabiliza en la parte derecha e inferior de la imagen.

Dado el enfoque anterior para el cálculo de los resultados de bloque con una ventana superpuesta, se puede describir la técnica utilizada para determinar la dirección del flujo de crestas en la imagen. Para cada bloque en la imagen, la ventana que rodea se hace girar de forma incremental y un análisis de una Transformada Discreta de Fourier (DFT) se lleva a cabo en cada orientación. La Ilustración 18 ilustra la rotación incremental de la ventana. El cuadro de la parte superior izquierda en la Ilustración representa una ventana con sus filas rotadas 90° en sentido anti horario de modo que quedan alineadas verticalmente. Esto se considera la orientación "0" en el software. Los parámetros “NUM_DIRECTIONS” en la estructura de control global lfspars_V2, especifican el número de orientaciones que deben analizarse en un semicírculo. Este parámetro se establece en 16, creando un incremento en el ángulo de 11,25° entre cada orientación. Estas orientaciones se representan en el círculo en la Ilustración 18. La fila inferior de la Ilustración muestra la rotación incremental de la filas de la ventana en cada orientación definida.

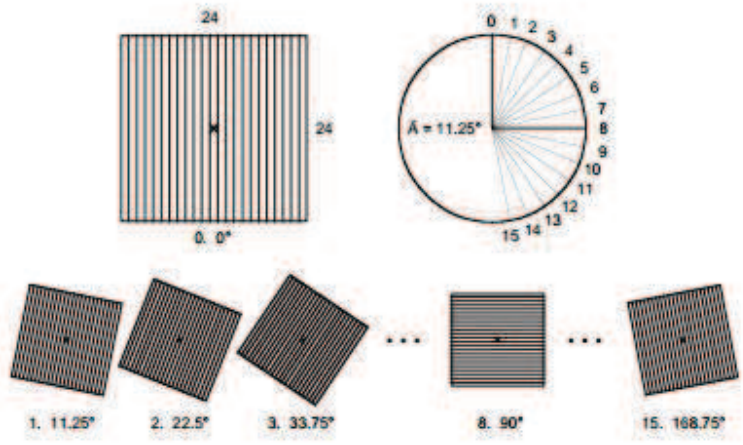


Ilustración 18: Orientación de la huella.

Al determinar la dirección del flujo de crestas para un bloque, cada una de sus orientaciones de ventana es analizada. Dentro de una orientación, los píxeles a lo largo de cada fila rotada de la ventana se suman entre sí, formando un vector de sumas de filas de 24 píxeles. Las 16 orientaciones producen 16 vectores de filas de

sumas. Cada vector de filas de sumas se convoluciona con 4 formas de onda de frecuencia cada vez mayor. Estos se ilustran en la Ilustración 19. La forma de onda superior en la Ilustración tiene un solo período que se extiende a través de la longitud de todo el vector. La frecuencia de la segunda forma de onda es el doble de la primera; la tercera se duplica a partir de la segunda y así sucesivamente. Los valores discretos para las funciones seno y coseno en las 4 frecuencias diferentes se calculan para cada unidad a lo largo del vector. Las filas de sumas en un vector se multiplican luego a sus correspondientes valores de seno discretos y los resultados se acumulan y encuadran. El mismo cálculo se realiza entre las filas de sumas en el vector y sus correspondientes valores de coseno discretos. El componente de seno cuadrado a continuación se añade al componente de coseno al cuadrado, produciendo un coeficiente de resonancia que representa lo bien que el vector se ajusta a la frecuencia de la forma de onda específica.

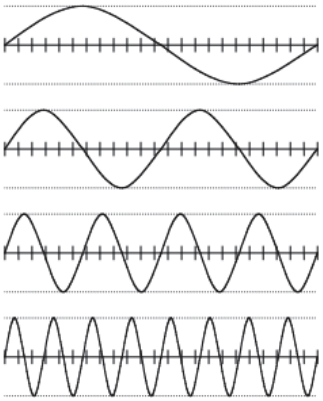


Ilustración 19: Frecuencia de la onda.

La frecuencia espacial de la forma de onda superior en la Ilustración 19 representa discretamente crestas y valles con una anchura de aproximadamente 12 píxeles. La segunda forma de onda representa crestas anchas y valles de 6 píxeles. La tercera forma de onda representa crestas anchas y valles de 3 píxeles. Por último, la cuarta forma de onda representa crestas anchas y valles de 1,5 píxeles. Dada una imagen escaneada en 19.69 ppmm, estas formas de onda cubren crestas y valles que van en anchura desde 0,6 mm hasta 0.075 mm. Los coeficientes de resonancia producidos a partir de la convolución de cada uno de los 16 vectores filas de suma de orientación con las 4 formas de onda discretas diferentes, se almacenan y luego son analizados. Generalmente, la dirección del flujo de crestas dominante para el bloque se determina por la orientación con la máxima resonancia de forma de onda. Los detalles se encuentran en el código fuente.

En la Ilustración 20, se muestra una imagen de la huella original a la izquierda. La imagen de la derecha, es la misma imagen de la huella con las direcciones de flujo

de crestas grabadas en el mapa de dirección resultante. Cada dirección en el mapa se representa como un segmento de línea centrado girado dentro de su correspondiente bloque de imagen de píxeles de 8×8.

Ilustración 20: Comparación huellas de bajo contraste.

Mapa de bajo contraste:

Es difícil, si no imposible, determinar con precisión un flujo de crestas dominante en ciertas porciones de una imagen de huella dactilar. Es así, el caso de las zonas de bajo contraste que contienen la imagen de fondo y las manchas. Es deseable detectar estas zonas y evitar artificialmente la asignación de direcciones de flujo de crestas donde realmente no hay aristas bien definidas, es problemático derivar un flujo de cresta arbitrario estrictamente a partir de los datos dentro de estas áreas. Un mapa de imagen llamado "*mapa de bajo contraste*" se calcula donde se marcan los bloques con un contraste suficientemente bajo; este mapa separa el fondo de la imagen de la huella dactilar, y traza manchas y áreas ligeramente entintadas de la huella dactilar, los puntos característicos no se detectan dentro de los bloques de bajo contraste en la imagen.

Una forma de distinguir un bloque de bajo contraste de un bloque que contiene crestas bien definidas es comparar las distribuciones de intensidad de sus píxeles. Por definición, hay poco rango dinámico en intensidad de los píxeles en un área de bajo contraste por lo que la distribución de intensidades de los píxeles será muy estrecha. Un bloque que contiene crestas bien definidas tendrá por otro lado, una gama de intensidad de los píxeles considerablemente más amplia, ya que habrá píxeles que van desde "muy ligero" en el medio de valles a "muy oscuros" en el medio de las crestas. Con el fin de determinar si un bloque es de bajo contraste,

este software calcula la distribución de intensidad de los píxeles dentro de la ventana que rodea el bloque. Un porcentaje especificado de colas de alta y baja distribución se recortan y se mide la anchura de la distribución restante. Si la anchura medida es suficientemente pequeña, entonces el bloque se encuentra marcado en el mapa como con bajo contraste.

En la estructura global de control lfsparms_V2, el parámetro PERCENTILE_MIN_MAX=10 hace que tanto el 10% de las más bajas y las más altas intensidades de los píxeles en la distribución se recorte. Debido al recorte de las colas, la medida del ancho posterior se efectúa en una porción mucho más estable de la distribución. El parámetro MIN_CONTRAST_DELTA=5 es el umbral de intensidad de los píxeles menos de lo que indica un bloque de bajo contraste; este umbral se derivó empíricamente a partir de una muestra de entrenamiento de bloques de bajo y alto contraste extraídos de imágenes de huellas dactilares reales. Los mapas de imágenes son en realidad computados en este software en una imagen con intensidad de píxel de 6 bits con 64 niveles de gris. El umbral de aquí de 5 en realidad corresponde a un umbral de 10 tonos de gris en la imagen original con intensidad de pixel de 8 bits con 256 niveles de gris. En otras palabras, si el rango dinámico del centro es 80% de un píxel de un bloque, la distribución de intensidad no es mayor que 10 tonos de gris, la cual se determina de bajo contraste. Las cruces blancas en la esquina de la imagen de la huella en la Ilustración 21 etiquetan bloques con contraste suficientemente bajo.

Ilustración 21: Huella con contraste bajo.

Mapa de bajo flujo:

Es posible cuando se deriva el mapa inicial de dirección que algunos bloques para resulten sin un flujo de crestas dominante, estos bloques corresponden normalmente a las zonas de baja calidad en la imagen. Inicialmente no se les asigna una orientación en el mapa de dirección, pero posteriormente a algunos de estos

bloques se le pueden asignar una orientación interpolando el flujo de crestas de bloques vecinos. El mapa de bajo flujo marca los bloques a los que no podrían serles asignados inicialmente un flujo de crestas dominante, en el caso de que los puntos característicos se detecten en estos bloques su calidad asignada se reduce debido a que se han detectado dentro de una parte menos fiable de la imagen. Las cruces blancas en la imagen de la huella en la Ilustración 22 etiquetan bloques sin flujo de crestas dominante.

Ilustración 22: Bloques sin flujo de cresta dominante.

Mapa de curva alta:

Otra parte de la imagen de la huella que es problemática cuando se trata de detectar de forma fiable puntos característicos son las zonas de alta curvatura. Esto es especialmente cierto en las regiones centrales y delta de una huella dactilar (26). El mapa de curva alta marca bloques que se encuentran en zonas de alta curvatura de la huella dactilar, se utilizan dos medidas diferentes; la primera llamada "*vorticidad*" mide la variación acumulada en la dirección del flujo de crestas alrededor de todos los vecinos de un bloque. La segunda llamada "*la curvatura*" mide el cambio más grande en la dirección del flujo de crestas de entre un bloque y el flujo de crestas de cada uno de sus vecinos, los detalles se encuentran en el código fuente. En el caso de que los puntos característicos se detecten en estos bloques su calidad asignada se reduce debido a que se han detectado dentro de una parte menos fiable de la imagen. Las cruces blancas en la imagen de la huella en la Ilustración 23 etiquetan bloques con las crestas de alta curvatura.

Ilustración 23: Bloques con las crestas de alta curvatura.

Mapa de calidad:

El mapa final de imagen producido por este paquete es un mapa de calidad. Como se mencionó el mapa de bajo contraste, bajo mapa de flujo y el mapa de curva elevada apuntan a diferentes regiones de baja calidad de la imagen. La información de estos mapas se integra en un mapa general, como se muestra en la Ilustración 24, y contiene 5 niveles de calidad. La calidad asignada a un bloque específico se determina en base a su proximidad a los bloques marcados en estos diversos mapas. Los detalles se encuentran en el código fuente.

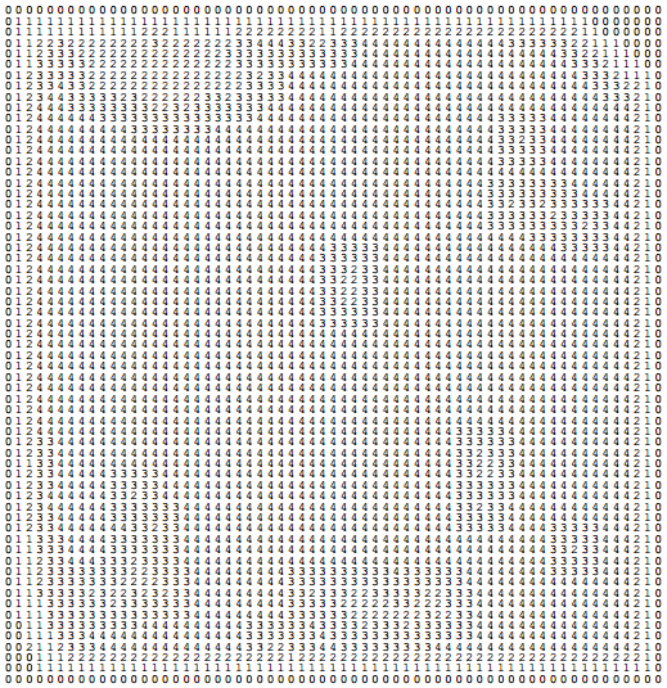


Ilustración 24: Calidad asignada.

Imagen binarizada:

El algoritmo de detección de puntos característicos en este sistema está diseñado para operar en una imagen de dos niveles (o binaria) en donde los píxeles negros representan las crestas y los píxeles blancos representan los valles en la piel de un dedo. Para crear esta imagen binaria, cada píxel de la imagen de entrada en escala de grises debe ser analizado con el fin de determinar si se debe asignar un pixel negro o blanco. Este proceso se conoce como "*binarizar la imagen"*. A un píxel se le asigna un valor binario basado en la dirección del flujo de crestas asociado con el bloque dentro del cual está el pixel. Si no hay un flujo de crestas detectable para el bloque del píxel actual, entonces el pixel se establece en blanco. Si no se detecta un flujo de crestas a continuación, las intensidades de los píxeles que rodean el pixel actual se analizan dentro de una cuadrícula girada como se ilustra en la Ilustración 25**.**

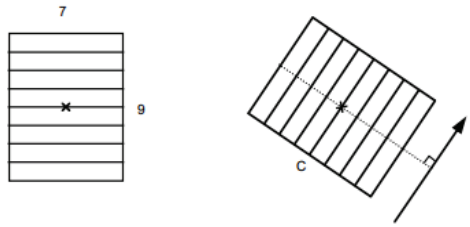


Ilustración 25: Dirección del flujo de cresta.

Esta cuadrícula es definida dentro de la estructura de control global lfsparms_V2, con el ancho de columna ajustado a 7 píxeles y una altura de fila ajustada a 9 píxeles. Con el píxel de interés en el centro, la cuadrícula se gira de manera que sus filas son paralelas a la dirección del flujo de crestas local, las intensidades de los píxeles en escala de grises se acumulan a lo largo de cada fila girada en la red, formando un vector fila de sumas. El valor binario que se asignará al píxel central se determina multiplicando la fila central suma por el número de filas de la cuadrícula y comparando este valor para las intensidades en escala de grises acumuladas dentro de toda la red. Si la fila central suma multiplicada es menos que la intensidad total de la red, entonces el píxel central se establece en negro; de lo contrario, se establece en blanco.

Ilustración 26: Transformación en escala de grises.

Cabe señalar que la etapa de binarización es crítica para la detección exitosa de puntos característicos en este enfoque. Los resultados de la binarización deben ser robustos en términos “efectividad” lidiando con diversos grados de calidad de imagen y “fiable” en cuanto a la representación con precisión de las estructuras de crestas y valles. Estos son un reto y a veces son objetivos contradictorios. Es deseable conservar tanta información de la imagen como estructura de crestas y valles como sea posible para que no se pierdan puntos característicos y sin embargo no es deseable acentuar áreas degradadas en la imagen hasta el punto de introducir falsos puntos característicos. Se han invertido importantes esfuerzos para promover imágenes binarias tanto robustas como fiables y sin embargo, el sistema actual tiende a producir un considerable número de puntos característicos falsos. Esto es problemático particularmente a la hora de procesar imágenes latentes de huellas dactilares.

Detección de puntos característicos:

Este paso analiza metódicamente la imagen binaria de una huella dactilar, la identificación de los patrones de píxeles localizados que indican el final o el fraccionamiento de una cresta, los patrones buscaron unos muy compactos como se ilustra en la Ilustración 27. El patrón de más a la izquierda contiene seis pixeles binarios en una configuración de 2×3, este patrón puede representar el final de una cresta negra que sobresale en el patrón de la derecha; es cierto también para el siguiente patrón de 2×4. La única diferencia entre este patrón y el primero es que el par de pixeles del medio se repite. De hecho, es así para todos los patrones representados, esta "familia" de patrones al final de la cresta pueden ser representados por el patrón que se encuentra más a la derecha, donde el par de pixeles de en medio (representado por "*") se puede repetir una o más veces.

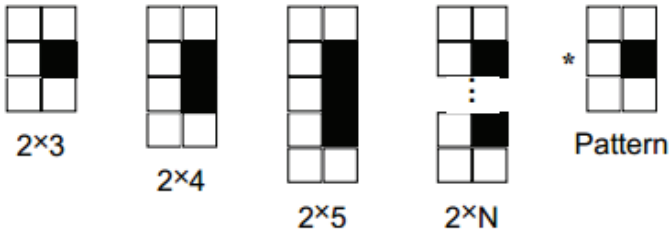


Ilustración 27: Patrones de binarización.

Los finales de crestas candidatos se detectan en la imagen binaria mediante el escaneo de pares consecutivos de píxeles en busca de secuencias que coincidan con este patrón. La exploración del patrón se lleva a cabo tanto de forma vertical como de forma horizontal, el patrón como se ilustra está configurado para la exploración vertical debido a que los pares de píxeles están apilados uno encima del otro. Para llevar a cabo la exploración horizontal, los pares de píxeles están se deben "desapilar", girar 90 ° hacia la derecha y colocarlos en secuencia de izquierda a derecha. Usando la representación anterior, se utilizan una serie de patrones de puntos característicos para detectar puntos candidatos en la imagen binaria de la huella. Estos patrones se ilustran en la Ilustración 28. Hay dos patrones que representan los candidatos finales de crestas, el resto representan diversas bifurcaciones de crestas. Un atributo secundario "aparecer/desaparecer" se asigna a cada patrón, designando la dirección en donde una cresta o valle sobresale en el patrón. Todas las secuencias de pares de píxeles que coincidan con estos patrones (como la imagen se escanea tanto vertical como horizontalmente), forman una lista de puntos característicos candidatos.

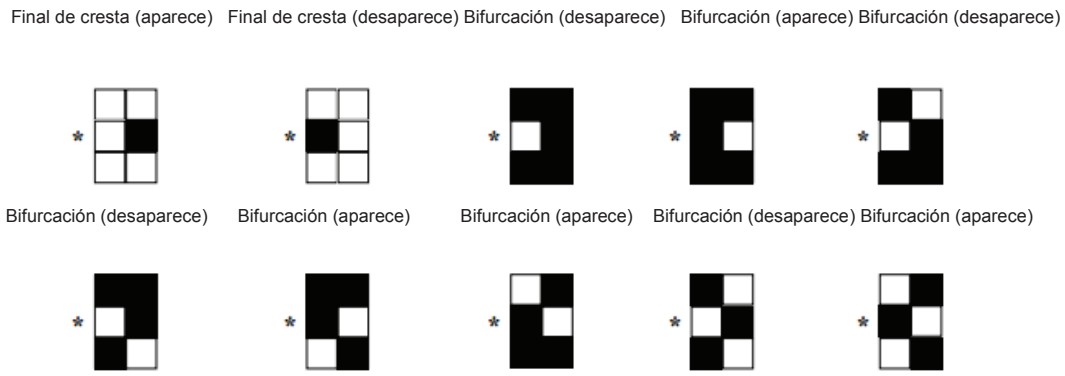


Ilustración 28: Puntos característicos.

Remover los puntos característicos falsos:

Usando los patrones en la Ilustración 28, se detectan los puntos característicos candidatos con tan sólo seis píxeles. Esto facilita un esquema de detección particularmente codicioso que minimiza la posibilidad de perder ciertos puntos; sin embargo, muchos puntos característicos falsos están incluidos en la lista de candidatos, debido a esto se gasta mucho esfuerzo en la eliminación de los puntos falsos. Estos pasos incluyen la eliminación de islas, lagos, agujeros, puntos característicos en regiones donde hay baja calidad de imagen, ganchos, superposiciones, puntos característicos que son demasiado amplios y puntos característicos que son demasiado estrechos (poros). Se proporciona una breve descripción de cada uno de estos pasos en el orden en que se ejecutan.

Remover islas y lagos:

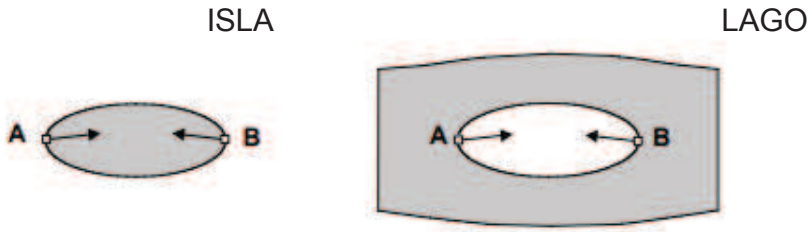


Ilustración 29: Islas y lagos.

En este paso, los fragmentos al final de la cresta y las marcas de tinta espurias (islas), junto con los huecos interiores de las crestas (lagos) son identificados y eliminados. Estas características son un poco más grandes que el tamaño de los poros en la piel y son a menudo de forma elíptica; por lo tanto tendrán típicamente un par detectado de puntos característicos candidatos en los extremos opuestos. Un ejemplo de estos tipos de características se muestra en la Ilustración 29. Se incluye en la parte inferior de la Ilustración los criterios utilizados para detectar las islas y lagos. Un par de puntos deben estar dentro de 16 pixeles de distancia uno del otro. Si es así, entonces las direcciones de los dos puntos característicos deben ser casi opuesta (≥ 123,75°) entre sí. A continuación ambos puntos característicos deben residir en el borde de un mismo lazo y el perímetro del bucle debe ser ≤60 pixeles. Si todos estos criterios se cumplen, entonces el par de puntos característicos candidatos se retiran de la lista y la imagen binaria se altera de manera que se llena la isla o lago. Tenga en cuenta que esta es la única etapa de eliminación que modifica la imagen de la huella binaria.

Remover hoyos:

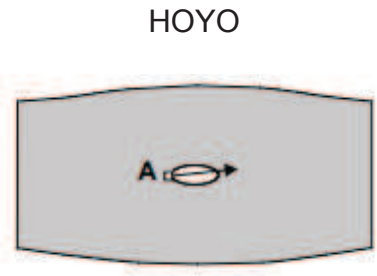


Ilustración 30: Hoyo.

Aquí un agujero se define de forma similar a una isla o un lago sólo que más pequeño y el bucle sólo necesita tener un punto característico en él. Los criterios para la eliminación de un agujero se ilustran en la Ilustración 30. Si un punto candidato se encuentra en el borde de un bucle con una longitud de perímetro ≤15 pixeles, entonces el punto se retira de la lista de candidatos.

Remover puntos para invalidar los bloques:

Este paso y el siguiente identifican y elimina puntos característicos candidatos que se encuentran cerca de los bloques que no contienen un flujo de crestas detectable. Estos bloques son denominados como contenedores “una dirección del flujo de crestas inválida” y representan zonas de baja calidad en la imagen de la huella.

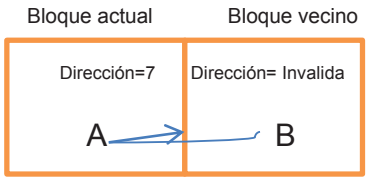


Ilustración 31: Remoción de puntos.

Este paso se ilustra en la Ilustración 31. Un punto característico se traslada 4 pixeles en la dirección en la que el punto está apuntando. Si el punto trasladado se encuentra dentro de un bloque con la dirección del flujo de crestas inválida, entonces el punto característico original es removido de la lista.

Remover bloques inválidos cercanos:

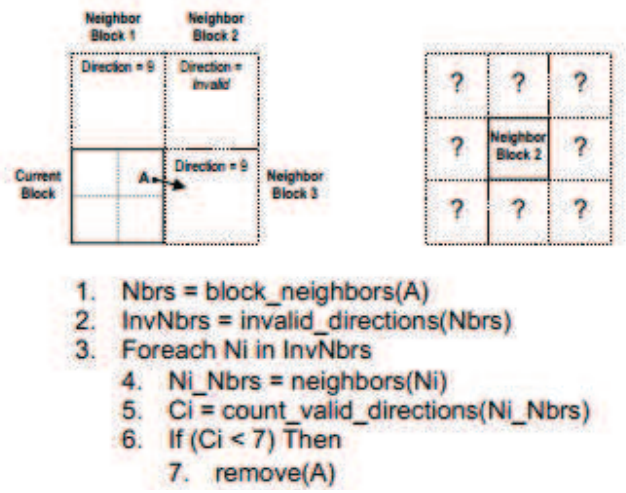
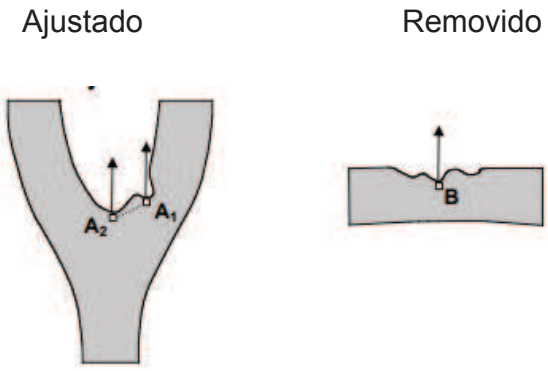


Ilustración 32: Remover bloques.

Aquí, se evalúa la proximidad de un punto característico candidato a un número de bloques circundantes con la dirección del flujo de crestas válida. Dado un punto característico, los bloques lo suficientemente cerca de este punto (detalles dejados al código fuente) y además los vecinos inmediatos del bloque en el que reside el punto, se ponen a prueba en turnos. Si uno de estos bloques vecinos tiene la dirección del flujo de crestas válida, sus 8 vecinos son probados. El número de bloques circundantes con la dirección del flujo de crestas válida se cuentan, y si el número de bloques válidos es <7, entonces el punto característico original se quita de la lista de candidatos. La Ilustración 32 ilustra este paso.

Remover o ajustar puntos característicos laterales:




```
1. Pts = trace_contours(A, 7 pixels)
2. R_Pts = rotate_points_vertical(Pts, direction(A))
3. {Min_Ys,Max_Ys} = min_and_max_Ys(R_Pts)
4. If (#Min_Ys == 1) Then
       Adjust(A, Pts[Min_Y1])
5. Else If ({Min_Ys,Max_Ys} ==
       {Min_Y1,Max_Y1,Min_Y2}) Then
           Min_Y = point_at_min_Y(R_Pts, Min_Ys)
           Adjust(A, Pts[Min_Y])
6. Else revove(A)
```

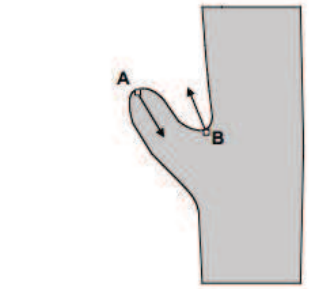
Ilustración 33: Remover puntos característicos laterales.

Este paso cumple con dos propósitos. El primero consiste en afinar la posición de un punto característico para que se coloque más simétricamente al final de una cresta o un valle, en el proceso se puede determinar que no hay una forma simétrica clara para el contorno en el que se encuentra el punto característico candidato; esto sucede a menudo en el caso de los puntos detectados a lo largo del lado de una cresta o de un valle en lugar de al final de estos. En este caso, se retira el punto característico fuera de lugar. En la Ilustración 33, la ilustración de la izquierda representa el ajuste de un punto característico desde el punto A_1 a A_2. La ilustración de la derecha representa la eliminación de un punto de lado B.

Para lograr esto, comenzando en el punto característico candidato, o bien el borde de la cresta o del valle se remonta a la derecha y a la izquierda 7 pixeles, produciendo una lista de 15 puntos de contorno. Las coordenadas de estos puntos de contorno se giran de manera que la dirección del punto candidato esté apuntando verticalmente. Las coordenadas giradas se analizan entonces para determinar el número y la secuencia de máximos y mínimos relativos en las coordenadas giradas. Si sólo hay una coordenada “y” mínima entonces el punto del mínimo se asume que se recaerá en la parte inferior de un contorno girado en forma de cuenca y el punto candidato se mueve para corresponder a esta posición en la imagen original. Si hay

más de una coordenada “y” para un mínimo, deberá existir una secuencia específica de mínimos-máximos-mínimos, en cuyo caso el punto candidato se mueve al punto en la imagen original correspondiente a la menor coordenada “y” de los mínimos. De nuevo, esto se supone que es la parte inferior de un contorno girado relativamente en forma de cuenca. Si hay más de un coordenada “y” para los mínimos y no hay una secuencia exacta mínimos-máximas-mínimos a lo largo del contorno girado entonces el punto característico es determinado en residir a lo largo de un lado de una cresta o valle y es removido de la lista de candidatos.

Remover ganchos:



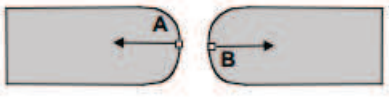

```
1. If (distance(A,B) <= 16 pixels) Then
2.   If (direction_angle(A,B) >= 123.75°) Then
3.     If (type(A) != type(B)) Then
4.       Pts = trace_contours(A, 30 pixels)
5.       If (in_points(Pts, B)) Then
6.         remove(A,B)
```

Ilustración 34: Remover ganchos.

Un gancho no es más que un pico o espolón que sobresale por el lateral de una cresta o de un valle. Un ejemplo se ilustra en la Ilustración 34. Esta característica tiene típicamente dos puntos característicos de tipo opuesto, uno en un pequeño trozo de la cresta y el otro en un pequeño valle que está relativamente cerca el uno al otro. Los dos puntos deben estar en un rango menor de 16 píxeles uno del otro, sus direcciones deben ser casi opuesta (≥123,75 °), que debe ser de tipo opuesto y deben estar en la mismo borde (ya sea de la cresta o del valle) dentro de los 30 píxeles de contorno el uno del otro. Si todo esto es cierto, entonces los dos puntos característicos se eliminan de la lista de candidatos.

Remover superposiciones:

En este paso una superposición es una discontinuidad en una cresta o un valle. Estos artefactos se introducen típicamente por el proceso de impresión de la huella dactilar. Una rotura en una cresta causa 2 terminaciones finales de crestas falsas que deben ser detectados, mientras que una ruptura en un valle hace 2 bifurcaciones falsas. Los criterios para la detección de una superposición se ilustran en la Ilustración 35. Dos puntos característicos deben estar en un rango de 8 pixeles uno del otro y sus direcciones deben ser casi opuesta (≥123,75 °). Si es así, entonces se calcula la dirección de la línea que une los dos puntos característicos. Si la diferencia entre la dirección del primer punto característico y la línea de unión es (≤90°), entonces los dos puntos característicos se eliminan de la lista de candidatos. De lo contrario, si los puntos característicos se encuentran dentro de un rango de 6 pixeles uno del otro y no hay transiciones de valor de píxel a lo largo de la línea de unión, entonces los puntos son retirados de la lista de candidatos.

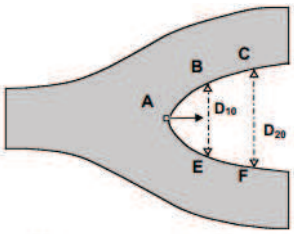



```
1. If (distance(A,B) <= 8 pixels) Then
   2. If (direction_angle(A,B) >= 123.75°) Then
      3. If(type(A) == type(B)) Then
         4. J = join_direction(A,B)
         5. If(direction_angle(180°-A,J) <= 90° Then
            6. remove(A,B)
         7. Else If (distance(A,B) <= 6 pixels &&
                     free_path(A,B)) Then
            8. remove(A,B)
```

Ilustración 35: Remover superposiciones.

Remover puntos característicos muy anchos:

Los dos pasos siguientes identifican puntos característicos falsos que se encuentran en las estructuras de crestas y valles malformados. Un extremo de una cresta generalizada se compone de un valle en forma de Y que envuelve una barra de negro. Lo contrario es cierto para una bifurcación generalizada. Las pruebas simples se aplican para evaluar la calidad de esta forma de Y.

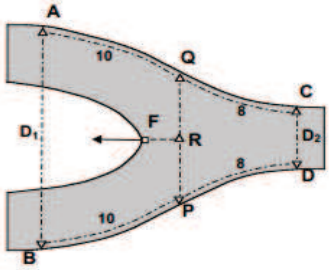



```
1.  Pts1 = trace_contour(A, 20 pixels)
2.  Pts2 = trace_contour(A, -20 pixels)
3.  B = Pts1[10]; C = Pts1[20]
4.  E = Pts2[10]; F = Pts2[20]
5.  D10 = distance(B,E)
6.  D20 = distance(C,F)
7.  If ((D20/D10) > 2.0) Then
    8.  remove(A)
```

Ilustración 36: Remover puntos característicos muy anchos.

Este paso evalúa si la estructura que envuelve un extremo de una cresta o valle es relativamente en forma de “Y” y no demasiado ancho. La Ilustración 36 ilustra los criterios aplicados. El borde de la cresta o del valle se remonta hacia la izquierda y hacia la derecha 20 píxeles, produciendo 2 listas de puntos de contorno. En cada contorno, se coordina en el índice de píxeles 10 (B&E) y se almacenan en el índice de píxeles 20 (C&F). La distancia entre los píxeles en el índice 10 se calcula como la distancia entre los píxeles en el índice 20. La relación de estas dos distancias se calcula entonces (D20/D10), y si la relación es mayor que 2,0, entonces el punto característico se elimina de la lista de candidatos. Cabe señalar que en base a estos criterios, la bifurcación en la ilustración no podría ser eliminada.

Remover puntos característicos muy angostos:




```
1.  T = 180° - direction(F)
2.  R = translate(F, 3 pixels, T)
3.  Q = find_edge(R, Up, 12 pixels)
4.  P = find_edge(R, Down, 12 pixels)
5.  Pts = trace_contour(Q, 10 pixels)
6.  A = Pts[10]
7.  Pts = trace_contour(Q, -8 pixels)
8.  C = Pts[8]
9.  Pts = trace_contour(P, 10 pixels)
10. B = Pts[10]
11. Pts = trace_contour(P, -8 pixels)
12. D = Pts[8]
13. D1 = distance(A,B)
14. D2 = distance(C,D)
15. If ((D1/D2) <= 2.25) Then
    16. remove(F)
```

Ilustración 37: Remover puntos característicos muy angostos.

El paso anterior prueba los puntos característicos candidatos que son demasiado anchos. Este paso prueba los puntos que se encuentran en las estructuras que son demasiado angostos. Esto es típico por ejemplo, en los poros de la piel. La Ilustración 37 ilustra esta prueba. Comenzando con el punto característico candidato F, sus coordenadas se trasladan 3 pixeles en la dirección opuesta al borde superior. El borde superior y los bordes inferiores y de la estructura envolvente, se encuentran entonces en (Q&P). A partir de estos dos puntos, el borde se remonta a la izquierda 10 pixeles y a la derecha 8 píxeles. Los puntos al final de los contornos de 10 píxeles se almacenan (A&B) y los puntos al final de los contornos de 8 píxeles se almacenan (C&D). A continuación, se calculan las distancias entre estos pares de puntos y se calcula la relación (D1/D2). Si la relación es ≤2,25, entonces el punto característico se elimina de la lista de candidatos. De hecho, si el proceso no encuentra cualquiera de los puntos en la ilustración, entonces se retira el punto candidato. Cabe señalar que, mindtct, sólo busca puntos que sean demasiado angostos dentro de las regiones de alta curvatura o zonas en donde el flujo de dirección de la cresta no es determinable.

Cuenta de crestas vecinas:

Los comparadores de puntos característicos de huellas dactilares a menudo usan información además de sólo la información de los puntos propios. La información auxiliar normalmente incluye la dirección de los puntos, su tipo y puede incluir además información relacionada con los puntos vecinos. Más allá de la posición un punto característico, la orientación y el tipo, no hay esquemas estándares de vecinos. Diferentes sistemas AFIS utilizan varias topologías y atributos para vecinos. Un atributo común es el número de aristas que intervienen (llamados cruces de crestas) entre un punto y cada uno de sus vecinos. Por ejemplo, el IAFIS del FBI utiliza cruces de cresta entre un punto y sus 8 vecinos más cercanos, en donde cada vecino es el más cercano dentro de un octante especificado (27). El esquema vecino distribuido con este sistema ha sido heredado directamente de HO39 (24) se reportan de este modo hasta los 5 vecinos más cercanos. Dado un punto característico, se seleccionan los vecinos más cercanos (en la misma columna de píxeles), y a la derecha (dentro de las columnas enteras de píxeles) de la imagen. Estos vecinos más cercanos están ordenados en orden por su dirección, empezando en sentido vertical y siguiendo en sentido horario. Usando esta topología, se calcula el número de crestas y se registran entre un punto y cada uno de sus vecinos más cercanos.

Evaluar la calidad de los puntos característicos:

Uno de los objetivos del desarrollo de este paquete de software fue calcular una relación calidad/fiabilidad asociada con cada punto característico detectado. Incluso con la larga lista de remoción (pasos arriba), los puntos característicos falsos permanecen potencialmente en la lista de candidatos. A medida de calidad robusta puede ayudar a gestionar esto y a esos puntos característicos falsos se les debe asignar una calidad inferior a la de los puntos verdaderos. A través de una señal dinámica puede determinarse una solución de compromiso entre el mantenimiento de puntos falsos y el arrojar puntos verdaderos; teniendo esto en cuenta, mindtct calcula y reporta las cualidades de los puntos. Se combinan dos factores para producir una medida de calidad para cada punto detectado, el primer factor “L” se toma directamente de la ubicación del punto en el mapa de calidad descrito anteriormente. Se le asigna inicialmente uno de los cinco niveles de calidad, siendo 4 el de más alta calidad y siendo 0 el de más baja.

El segundo factor se basa en simples estadísticas de intensidad de píxel (media y desviación estándar) en las inmediaciones del punto de característico. El tamaño de la vecindad se establece en 11 píxeles, esto es lo suficientemente grande como para contener porciones generosas de una cresta de tamaño medio y valles. Una región de alta calidad de imagen de una huella dactilar dentro tendrá un contraste significativo que cubrirá el espectro completo de escala de grises. En consecuencia, la intensidad media de los pixeles de una vecindad será muy cerca de 127. Por razones similares, las intensidades de los píxeles de un vecindario ideal tendrán una desviación estándar ≥64.

Usando esta lógica, la siguiente medida de fiabilidad “R”, se calcula dando una vecindad significante “μ” y una desviación estándar “σ”:

$$F_\mu = 1.0 - \frac{|\mu - 127|}{127}$$

$$F_\sigma = \begin{cases} 1.0 & if\ \sigma > 64 \\ \dfrac{\sigma}{64} & \end{cases}$$

$$R = \min(F_\mu, F_\sigma)$$

Los puntos característicos de alta calidad “Q”, se calculan utilizando el nivel de mapa de calidad “L” y la fiabilidad “R” como:

$$Q = \begin{cases} .50 + (.49 * R) & if\ L = 4 \\ .25 + (.24 * R) & if\ L = 3 \\ .10 + (.14 * R) & if\ L = 2 \\ .05 + (.04 * R) & if\ L = 1 \\ .01 & if\ L = 0 \end{cases}$$

Esto resulta en un valor de calidad en el rango de 0,01 a 0,99. Un valor bajo de la calidad representa un punto característico detectado en una región de menor calidad de la imagen, mientras que un valor de alta calidad representa un punto característico detectado en una región de mayor calidad.

Archivo de punto característico de salida:

Al finalizar, mindtct toma los puntos característicos resultantes y les da salida a un archivo. Si el archivo de entrada era un archivo tipo ANSI/NIST formateado mindtct añade dos nuevos registros y escribe un nuevo archivo con formato ANSI/NIST a <oroot> .mdt, donde <oroot> se pasa como un parámetro para mindtct. Los nuevos registros son un registro de Tipo 9 sosteniendo los puntos característicos detectados, se construye y se inserta junto con un registro de Tipo 13 o Tipo 14, sosteniendo los resultados de la binarización de la imagen. Si la imagen de entrada es de una huella dactilar latente, a continuación los resultados de la binarización se almacenan en un registro de Tipo 13; de lo contrario, los resultados de la imagen se almacenan en un registro de Tipo 14. Cabe señalar que los puntos en el registro de Tipo 9 están formateados en los campos asignados por el NIST 5.12 de acuerdo a la norma ANSI/NIST (26). Las utilidades “an2k7toiaf” y “iaf2an2k7” pueden ser usadas para convertir entre estos campos y los campos del FBI/IAFISassigned (27). Si el archivo de entrada no está en formato ANSI/NIST, a los puntos resultantes se les puede acceder en el archivo de texto <oroot>.min y no hay ningún archivo de salida ANSI/NIST creado pero se crea un archivo de píxeles en bruto que tiene los resultados de la binarización de la imagen.

Para todos los tipos de entrada los puntos característicos detectados también se escriben en un archivo de texto <oroot>.xyt que está formateado para su uso con los comparadores bozorth3. Este archivo tiene una línea de espacio delimitada por puntos característicos que contienen su coordenadas X e Y, un ángulo de dirección theta y la calidad de los puntos. Los puntos característicos de salida están en el formato ANSI/NIST, que tienen el origen en la parte inferior izquierda de la imagen y también las direcciones apuntando hacia afuera de los finales de la cresta o de la bifurcación del valle. Hay una opción de salida para el punto en la representación

M1 (ANSI INCITS 378-3004), que tiene el origen del píxel en la parte superior izquierda de la imagen y las direcciones apuntando hacia el final de la cresta o de la bifurcación del valle. Si esta opción (-m1) se utiliza cuando se ejecuta mindtct también debe ser usada por bozorth3 al emparejar los archivos de los puntos característicos.

Se produce así una serie de otros archivos de salida, estos incluyen un archivo para cada uno de los mapas de imágenes que se describieron antes y un archivo de registro listando todos los puntos característicos detectados asociados y sus atributos. Todos estos son archivos de texto y son creados por mindtct en el directorio de trabajo actual con los nombres de archivos fijos. El mapa de dirección se almacena en <oroot>.dm; el mapa de bajo contraste se almacena en <oroot>.lcm; el mapa de bajo flujo se almacena en <oroot>.lfm; el mapa de curva alta se almacena en <oroot>.hcm; y el mapa de calidad se almacena en <oroot>.qm. Los mapas están representados por una cuadrícula de números, cada uno correspondiente a un bloque en la imagen de la huella. El último archivo de salida de texto, <oroot>.min, contiene una lista formateada de atributos asociados a cada punto característico detectados en la imagen de la huella. Entre estos atributos encontramos localización coordinada de los píxeles, su dirección y el tipo.

Ilustración 38: Marcación de puntos característicos.

2.4. BOZORTH3

Algoritmo útil para la toma de características extraídas de dos huellas dactilares (tales como los puntos característicos detectados en MINDTCT) y la comparación de ambas, ya sea para el propósito de una verificación de uno a uno, o uno a muchos. Este tipo de algoritmo se conoce comúnmente como un “comparador de huellas dactilares”.

Antecedentes:

De 1995 a 2002, el FBI tenía una pantalla de demostración para exhibir el Sistema Integrado Automatizado de Identificación (AFIS) y su interoperabilidad con el Centro de Información de Crímenes Nacionales (NCIC2000). Uno de los propósitos de esta demostración era simular la adquisición y búsqueda en tiempo real de huellas dactilares y la respuesta de forma remota desde el campo.

Antes de este proyecto de demostración, un empleado del FBI con el nombre de Allan S. Bozorth, había partido en un esfuerzo por investigar la noción de un algoritmo invariante de traslación y rotación para hacer coincidir dos huellas dactilares entre sí. Allan tuvo éxito en el diseño de un algoritmo de este tipo, e implementó el software. Probó el comparador de huellas dactilares ampliamente usando la base de datos de huellas dactilares NIST que estaba disponible, y en las propias palabras de Allan, "*me sorprendieron gratamente los resultados*", pero *"aun no tengo un buen presentimiento acerca de su desempeño".*

Fue en ese momento que comenzó la construcción de la pantalla de demostración. En la demostración, se utilizó el algoritmo de “*Home Office*” para la detección de puntos característicos (el algoritmo en el que se basa MINDTCT); y cuando la necesidad de un comparador de huellas dactilares surgió, el algoritmo de Allan fue seleccionado e integrado tanto en las porciones del AFIS como del NCIC de la pantalla. El comparador se desempeñó en un nivel adecuado para apoyar la demostración, donde un invitado podía ser registrado y buscado contra un pequeño antecedente.

Para sorpresa de Allan, este algoritmo ya ha sido ampliamente utilizado como punto de referencia por el NIST para apoyar el trabajo en virtud de la Ley Patriota de Estados Unidos. El algoritmo ha demostrado desempeñarse respetablemente bien tanto para aplicaciones de verificación como de identificación.
En honor del duro trabajo y los logros de Allan, el NIST ha elegido nombrar este comparador "*el comparador Bozorth*" o en corto "*Bozorth*". A continuación se hace una descripción del algoritmo:

Algoritmo de bozorth:

Dos cosas claves son importantes a tener en cuenta con respecto a este comparador de huellas dactilares:

1. Las características de los puntos característicos son exclusivamente usadas y limitadas a la posición (x, y) y a la orientación 't', representado como {x, y, t}.
2. El algoritmo está diseñado para ser invariante en cuanto a la rotación y traslación.

El algoritmo está compuesto de tres pasos principales:

1. Construir tablas de comparación de puntos característicos localizados dentro de las huellas dactilares.
   - Una tabla para la huella digital de prueba y una tabla para cada huella dactilar de la galería y así comparar ambas.
2. Construir una tabla de compatibilidad entre huellas dactilares
   - Comparar una tabla de comparación de impresiones de prueba de puntos característicos contra una tabla de comparación de la galería de impresiones de puntos característicos y construir una nueva tabla de compatibilidad.
3. Cruzar la tabla de compatibilidad entre huellas dactilares:
   - Cruzar y unir las entradas de la Tabla con los clústeres.
   - Combinar clústeres compatibles y acumular una puntuación de coincidencia.

Construcción de tablas de comparación de puntos característicos dentro de las huellas dactilares:

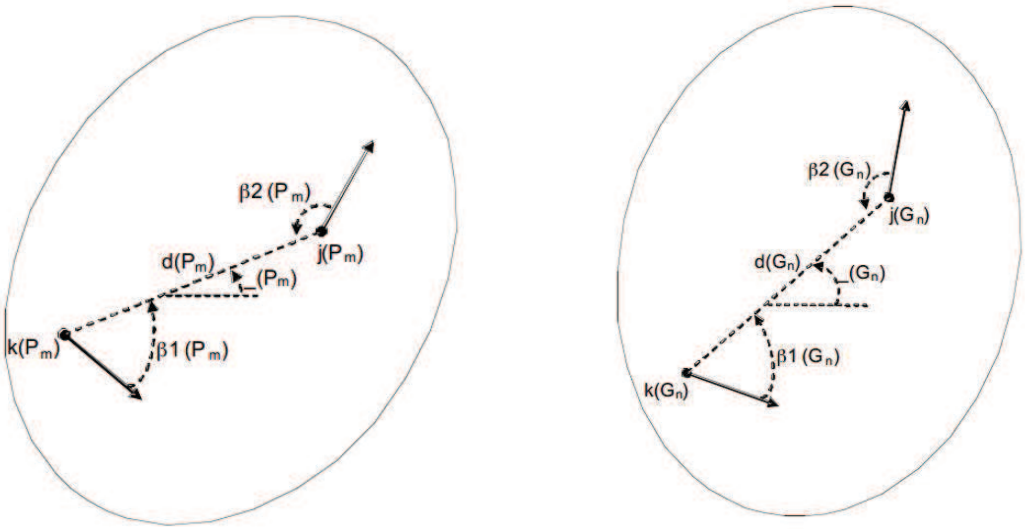


Ilustración 39: Comparación de puntos característicos de las huellas dactilares.

El primer paso en el comparador Bozorth es calcular las mediciones relativas de cada punto característico de una huella dactilar y llevarla a todos los demás puntos característicos de la misma huella dactilar. Estas mediciones relativas se almacenan en una "*tabla de comparación de puntos característicos*" y son los que proporcionan la invariación de rotación y traslación del algoritmo.

La Ilustración 39 ilustra las mediciones entre puntos característicos que se utilizan. Hay dos puntos característicos que se muestran en este ejemplo. Los puntos k se encuentran en la parte inferior izquierda de la "huella dactilar" y su ubicación es descrita por la representación de puntos (x_k, y_k). Por otro lado la flecha que apunta hacia abajo y hacia la derecha representa la orientación t_k. Un segundo punto característico j está en la parte superior derecha. Para tener en cuenta la posición de translación relativa, la distancia d_{kj} se calcula mediante las ubicaciones de ambos puntos característicos. Esta distancia se mantendrá relativamente constante entre los puntos correspondientes de dos impresiones dactilares diferentes, independientemente de cuanto desplazamiento y rotación pueda haber entre ambas.

Para realizar medidas relativas rotacionales es un poco más complicado. El objetivo de cada uno de los puntos de comparación en la comparación por pares, es calcular el ángulo entre la orientación de cada punto y la línea intermedia entre ellos. De esta manera, estos ángulos permanecen relativamente constantes a la línea de intervención independientemente de cuánto se gire la huella dactilar. En la ilustración anterior, el ángulo θ_{kj} de la línea de intervención entre los puntos k y j se calcula tomando el arco tangente de la pendiente de la línea de intervención. Los ángulos β_k y β_j se calculan con respecto a la línea de intervención como se muestra mediante la incorporación de θ_{kj} y la orientación t de cada punto característico. Cabe señalar que la comparación de puntos por pares se lleva a cabo en posiciones de puntos ordenados primero en la coordenada x y a continuación en la coordenada y, y que todas las orientaciones están limitadas al periodo (-180 °, 180 °] con 0 ° apuntando a la horizontal dirigida hacia la derecha y los grados crecientes van en sentido anti horario. Para cada comparación de puntos por pares, se realiza una entrada en una tabla de comparación. Cada entrada consta de:

- {dkj, β1, β2, k, j, θkj}

  En donde: β1 = min(βk,βj) and β2 = max(βk,βj)

  Entonces de la Ilustración 39

- β1 = βk and β2 = βj

Las entradas se almacenan en la tabla de comparación con el fin de aumentar la distancia y la tabla se recorta en el punto en el cual se alcanza el umbral de distancia máxima. Con el fin de hacer estas mediciones entre pares de puntos, una tabla de comparación debe ser construida con todas y cada una de las huellas dactilares que se desea comparar.

Construcción de una tabla de comparación entre huellas dactilares.

El siguiente paso en el algoritmo de coincidencia Bozorth es tomar las tablas de comparación de los puntos característicos de dos huellas dactilares distintas y buscar las entradas "compatibles" entre ambas tablas. La Ilustración 40 representa dos impresiones de la misma huella dactilar con ligeras diferencias en la rotación y la escala. Dos puntos característicos correspondientes se muestran en cada huella.

La impresión superior izquierda representa una impresión probada en la cual todos sus puntos han sido comparados por pares las con mediciones relativas almacenadas en la tabla de comparación de puntos característicos P. Las mediciones calculadas a partir de cada par de puntos en este ejemplo han sido almacenadas como la entrada m^{th} en la tabla P, denotada P_m. La notación de los valores individuales almacenados en la tabla se representa como funciones de búsqueda en una entrada de la tabla dada. Por ejemplo, el índice del punto característico inferior izquierdo se almacena en entrada de la tabla de la tarde y se hace referencia como $k(P_m)$, mientras que la distancia entre los dos puntos característicos también se almacena en la entrada P_m y se le hace referencia como $d(P_m)$. La huella inferior derecha representa una impresión de la galería, y utiliza una notación similar, excepto que todas sus comparaciones de puntos por pares han sido almacenadas en la tabla G, y las mediciones realizadas de los dos puntos correspondientes en la galería de impresión han estado guardadas en la entrada G_n de la tabla. Las tres pruebas siguientes se llevan a cabo para determinar si las entradas P_m y G_n son "compatibles". Los primeros chequeos de prueba para ver si las distancias correspondientes se encuentran a una tolerancia especificada T_d. Las dos últimas pruebas verifican si los ángulos relativos de los puntos dentro se encuentran dentro de una tolerancia especificada T_β. $\Delta_d()$ y $\Delta_\beta()$ son delta o funciones de diferencias.

- $\Delta_d\ (d(P_m),d(G_n)) < T_d$
- $\Delta_\beta\ (\beta1(P_m),\beta1(G_n)) < T_\beta$
- $\Delta_\beta\ (\beta2(P_m),\beta2(G_n)) < T_\beta$

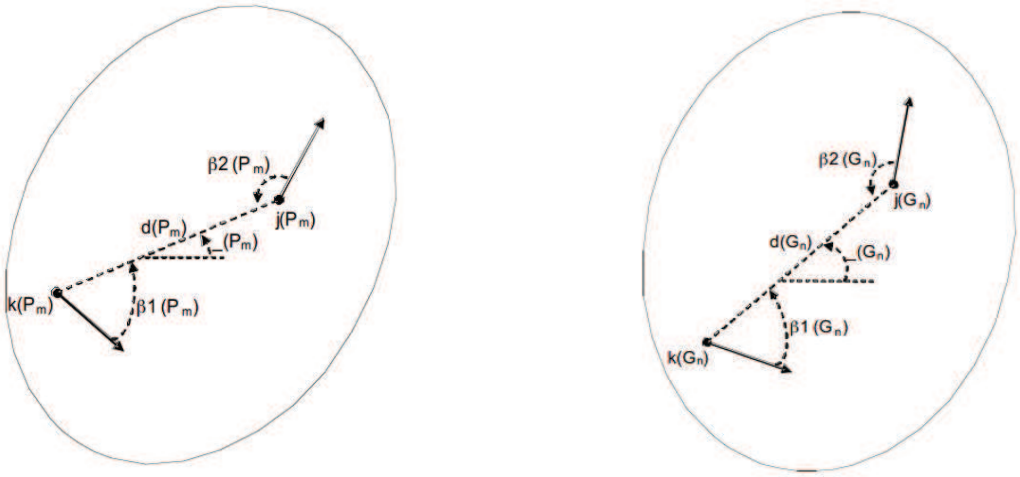


Ilustración 40: Mediciones de puntos característicos por pares compatibles entre dos huellas dactilares que generan una entrada en una tabla de compatibilidad.

Si la distancia relativa y los ángulos de los puntos característicos entre las dos entradas de la tabla de comparación están dentro de la tolerancia aceptable, entonces se introduce la siguiente entrada la una tabla de compatibilidad:

- $\{\Delta_\beta(\theta(P_m),\theta(G_n)), k(P_m), j(P_m), k(G_n), j(G_n)\}$

Por lo tanto, un elemento de la tabla de compatibilidad incorpora dos pares de puntos característicos, un par de la huella dactilar de prueba (k (P_m), j (P_m)) y el otro de la galería de huellas dactilares (k (G_n), j (G_n)). La entrada en la tabla de compatibilidad indica entonces que k (P_m) corresponde a k (G_n) y j(P_m) corresponde a j(G_n). El primer término de la entrada de la tabla, $\Delta_\beta(\theta (P_m), \theta (G_n))$, se utiliza posteriormente para combinar clústeres que compartan una cantidad similar de rotación global entre los puntos característicos de prueba y los de la galería.

Cruzando la tabla de compatibilidad entre huellas dactilares:

En este punto del proceso, hemos construido una tabla de compatibilidad que consiste en una lista de asociación de compatibilidad entre dos pares de puntos característicos potencialmente correspondientes. Estas asociaciones representan enlaces individuales en una "*gráfica de compatibilidad*". Para determinar qué tan bien las dos huellas dactilares coinciden entre sí, un objetivo simple sería atravesar el gráfico de compatibilidad encontrando el camino más largo de las asociaciones de compatibilidad enlazadas. La puntuación de compatibilidad sería entonces la longitud de la trayectoria más larga.

Hay algunos serios desafíos que enfrentar en un enfoque tan simple como este. Entre estos encontramos:

1. La tabla de compatibilidad no es un gráfico coherente, sino más bien una colección disjunta de enlaces individuales dentro de un gráfico.
2. Cada nodo en el gráfico es potencialmente relacionado con muchos otros nodos.
3. Esto conduce al potencial de circuitos.
4. No hay un nodo raíz evidente en el gráfico que pueda ser predicho a conducir al camino de máxima longitud.
5. Oclusiones y/o huecos dentro de cualquiera de las dos huellas dactilares que son comparadas provocarán discontinuidades en el gráfico.

Para tener en cuenta estas cuestiones, Allan Bozorth implementó un algoritmo que procesa la tabla de compatibilidad para que las transversales sean iniciadas desde distintos puntos de vistas. Así como las transversales son realizadas, las porciones o clústeres de la gráfica de compatibilidad se crean mediante el enlazamiento de las entradas en la tabla. Una vez que las transversales están completas, los clústeres "compatibles" se combinan y el número de entradas de la tabla enlazadas a través de los clústeres combinados se acumulan para formar la puntuación de coincidencia. Cuanto mayor sea el número de asociaciones de compatibilidad enlazadas, mayor será la puntuación de coincidencia, y será más probable que las dos huellas dactilares sean de la misma persona, el mismo dedo.

Implementación de bozorth3:

Dado el algoritmo de coincidencia anterior, el programa original de Allan Bozorth calcula una puntuación de coincidencia de un solo par de huellas dactilares. Una primera versión de este comparador *uno a uno*, llamado "bozorth98," se ha utilizado ampliamente por el NIST en estudios de referencia [25]. La implementación incluida en esta distribución, llamada BOZORTH3, ha sido modificada para calcular una puntuación de coincidencia entre los puntos característicos de cualquier número de pares de huellas dactilares. Por ejemplo, se puede calcular puntuaciones de coincidencia entre una huella dactilar de prueba y cualquier número de huellas dactilares de la galería. Para procesar un conjunto de prueba de tamaño P contra una galería de tamaño G, el programa original necesitaba leer y pre procesar 2*(P*G) archivos. El pre procesamiento consiste en analizar los puntos característicos {x, y, t}'s desde el archivo, ordenar los {x, y, t}'s y recortar el periodo en las orientaciones, para entonces calcular la tabla de compatibilidad de puntos

característicos. La capacidad de *uno a muchos* significa que cada archivo de prueba se lee y se pre procesa sólo una vez, reduciendo así el número de archivos leídos y pre procesados a P+(P*G). El único requisito de memoria adicional es el espacio necesario para almacenar internamente una tabla de compatibilidad de puntos característicos de huellas dactilares de prueba pre procesadas. Adicionalmente de las modificaciones para apoyar más eficiente la comparación de *uno a muchos*, BOZORTH3 se ha reorganizado, optimizado, y se ha hecho más portátil.

Por defecto, BOZORTH3 produce una línea por cada comparación que se calcula, conteniendo sólo la puntuación de coincidencia. Idealmente, la puntuación de coincidencia es alta si los dos conjuntos de puntos característicos de entrada son del mismo dedo de un sujeto, y baja si son de diferentes dedos. La implementación de la tabla de coincidencias transversales descrito anteriormente es inagotable y por lo tanto no garantiza un resultado óptimo. La puntuación de coincidencia resultante representa aproximadamente (pero no exactamente) el número de puntos característicos que pueden ser igualados entre las dos huellas dactilares. Como regla general, una puntuación de coincidencia de más de 40 por lo general indica una coincidencia verdadera. Para la evaluación del desempeño, ver (28) (29) (30). Las puntuaciones de correspondencia de BOZORTH3 son generalmente, pero no siempre, idénticas a esas publicadas de "bozorth98." Los cambios en los cálculos de puntos flotantes, una corrección lógica efectuando el cruce de la tabla de coincidencias, y las modificaciones pertinentes para evitar índices de arreglos ilegales son los principales responsables de las diferencias de puntuaciones.

Por defecto, sólo los 150 puntos característicos de mejor calidad (la calidad de los puntos característicos es determinada por el algoritmo de extracción de los puntos característicos, actualmente denominado MINDTCT) de cada huella digital de entrada se utilizan. Eso debería ser más que suficiente, un dedo normalmente tiene menos de 80 puntos característicos por lo que un extractor de puntos característicos se puede utilizar incluso si es demasiado sensible produciendo muchos falsas minucias.

2.5. PARSE

Parse proporciona servicios de back-end basados en la nube para desarrolladores de aplicaciones móviles. Ofrece una pila completa de servicios móviles para que los desarrolladores puedan centrarse en el desarrollo de aplicaciones y dejar la infraestructura a Parse. Ofrece la administración de servidores de más de 180.000 aplicaciones móviles para Android, iOS y Windows, que funcionan con más de 200 millones de dispositivos móviles.

Parse opera con alto rendimiento, I/O MongoDB clusters intensivos y necesita mejorar la escalabilidad y velocidad. Maneja la administración de cuentas de usuario, almacenamiento de datos y almacenamiento en caché de disco para sus clientes y su uso puede fluctuar diariamente.

Parse está utilizando 1.000 volúmenes aprovisionados para IOPS en los clústeres de MongoDB que se ejecutan en Amazon Elastic Compute Cloud (Amazon EC2). Los ingenieros utilizan Elastic Load Balancing en la parte superior de la pila para distribuir las conexiones a servidores web, que se conectan a los servidores de aplicaciones que se comunican con las bases de datos. La Ilustración 41 ilustra el entorno de Parse.

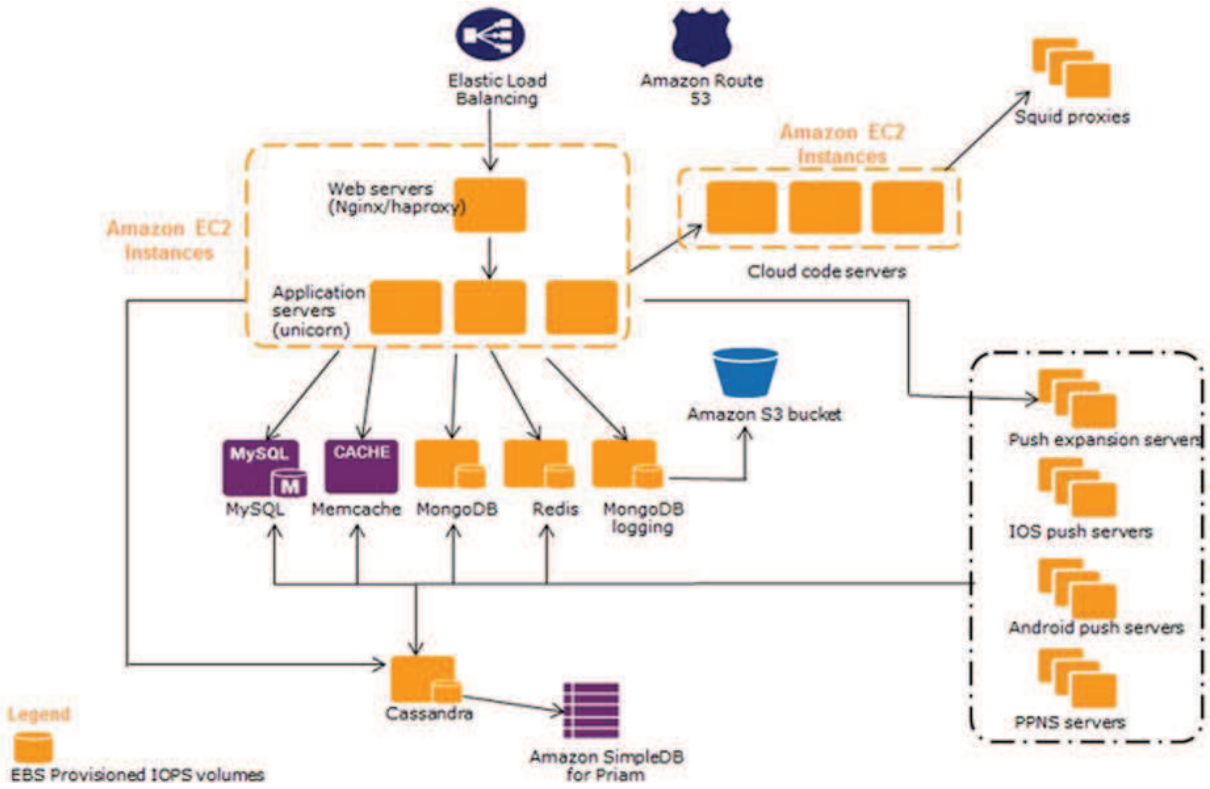


Ilustración 41: Arquitectura de Parse sobre AWS.

Parse ejecuta sus bases de datos MongoDB en conjuntos de réplicas que incluyen una base de datos primaria y dos secundarias. Además utiliza Amazon Elastic Block Store (Amazon EBS) para crear instantáneas con frecuencia para cada fragmento MongoDB, que luego se sube a un Simple Storage Service de Amazon (Amazon S3). Si es necesario, Parse puede abrir un nuevo nodo en cuestión de minutos utilizando Amazon S3 y unirlo a un clúster (31).

2.6. Android

Conceptos generales:

Android es un sistema operativo basado en el kernel de Linux diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes o Tablets.

La principal razón y, es la ventaja de Android, es que es código libre y posee una gran comunidad de desarrolladores trabajando en la constante mejora de las aplicaciones que utiliza este sistema y en el propio desarrollo del sistema operativo.

Esta ventaja hace que realizar esta prueba de concepto tenga un manejo flexible y económico antes de una implementación robusta y a la medida, además de esto las funcionalidades de lectura de huella dactilar embebidas en la Tablet son compatibles con este sistema operativo, evitando un problema de adaptación al hardware.

Si bien Android es un sistema operativo con muchas cualidades se debe considerar unos riesgos de seguridad al momento del desarrollo de la aplicación para la prueba de concepto, y adicionalmente si se implementara como proyecto productivo, dentro de los factores a tener en cuenta serian: garantizar el envío exitoso de la información, establecer conexiones con el servidor con contraseñas seguras, dejar exclusivamente la Tablet para el uso del aplicativo y deshabilitar los botones físicos innecesarios, no todos estos aspectos se tienen en cuenta en la prueba de concepto pero productivamente serían necesarios.

Versiones:

Con el pasar de los años el sistema operativo ha evolucionado de manera vertiginosa, lo cual ha llevado a desarrollar diferentes versiones en muy poco tiempo. Cada versión tiene como característica identificativa su propio nivel de API (Application Programming Interface) diferente. Una API no es más que un protocolo diseñado para utilizarse al momento de desarrollar el software, en un lenguaje más técnico, es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizada por otro software como una capa de abstracción, esta puede incluir especificaciones para distintas rutinas, estructuras de datos y clases para diferentes objetos.

Es muy importante tener en cuenta la versión en la que se desarrollara la aplicación, para que cuente con la posibilidad de actualizaciones que puedan mejorar el servicio a futuro, teniendo en cuenta las versiones de Android la aplicación se desarrolló principalmente para la versión KitKat (v4.4).

A continuación se resaltan las diferentes versiones de este sistema operativo junto con la distribución en el mercado de cada una de ellas, pero se debe tener en cuenta que las primeras versiones ya están desactualizadas y por lo tanto salieron del mercado, la información que se muestra es de las versiones más actuales que aún siguen siendo usadas por los diferentes usuarios: (32)

Versión	Nombre de referencia	API	Distribución (%)
2.2	Froyo	8	0,6
2.3.3 2.3.7	Gingerbread	10	9,8
4.0.3 4.0.4	Ice Cream Sandwich	15	8,5
4.1	Jelly Bean	16	22.8
4.2		17	20,8
4.3		18	7,3
4.4	KitKat	19	30,2

Ilustración 42: Porcentaje de distribución de las versiones de Android.

Ventajas:

Viendo muchas de las cualidades de este sistema operativo, se mencionan a continuación aquellas por las que fue la plataforma elegida para desarrollar esta tesis:

- *Implantación:* Actualmente y como se mencionó anteriormente Android es el sistema operativo que mayor porcentaje de usuarios tiene en el mercado, de

ahí que sea estratégico implementar esta plataforma pues se llegará a un mayor número de usuarios.

- *Código abierto:* Es un sistema operativo que le da la posibilidad a cualquier desarrollador no solo de crear nuevas aplicaciones sino además de mejorar las ya existentes, para de esta forma optimizar y otorgarles a los dispositivos un mayor desempeño.
- *Libertad:* Android le da la posibilidad al usuario de instalar y eliminar aquella aplicación que este desee sin ningún tipo de inconveniente lo que permite que cada persona tenga un control total sobre su dispositivo.
- *Comunidad:* Al ser Android un sistema altamente usado, una ventaja es que posee una gran comunidad de desarrolladores, de ahí que la cantidad de ideas y consejos que se encuentran al momento de desarrollar código es inimaginable; de este modo le facilita mucho al usuario el poder diseñar su propio código.
- *Multitarea:* Una característica importante de esta plataforma, permite al usuario tener diversas aplicaciones en ejecución suspendiendo aquellas que no esté utilizando y cerrándolas en caso de que no resulten útiles, con el fin de ahorrar batería o memoria.
- *Personalización:* Android es totalmente personalizable pues le permite a los usuarios cambiar fondos de pantalla, añadir "widgets" etc… Y a los fabricantes adaptar el sistema operativo para las propias necesidades del dispositivo.

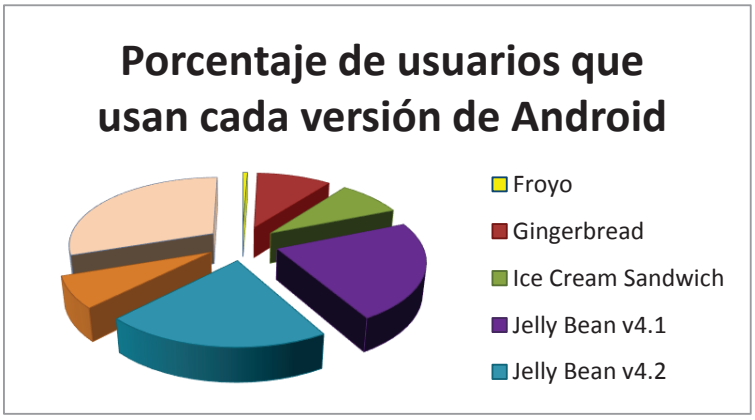


Ilustración 43: Distribución de versiones de Android.

Estructura:

Android por ser intuitivo y simple posee una estructura basada en “activities”. Una *activity* no es más que una actividad que el usuario puede hacer y la mayoría de estas interactúan directamente con la persona. Hablando a nivel de software, una activity es una clase (en Java), cuya función principal es crear una ventana en la cual se pueda programar para mostrar o hacer cualquier cosa que se desee.

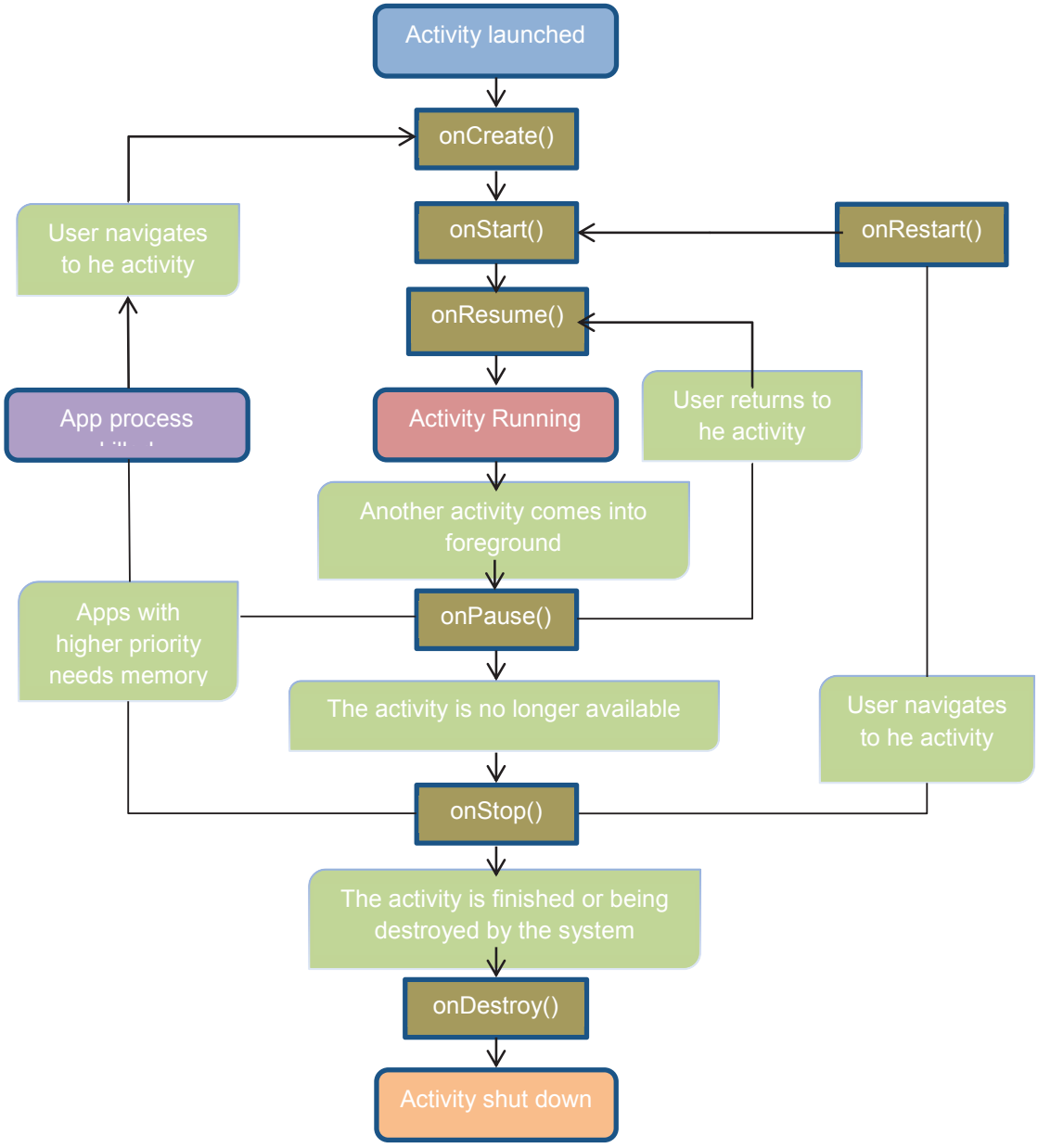


Ilustración 44: Proceso de una actividad. [30]

Para crear la ventana mencionada antes el usuario debe asociar la activity con un view (vista), en la que se incluirán todas las formas, botones, textos, colores, sonidos etc… Que aparecerán en la *activity.* A continuación se muestra un diagrama indicando el ciclo de vida de una *activity* en Android desde que se lanza hasta que se cierra, así como todos los caminos intermedios que puede tomar (33).

3. SecurityP (App)

En esta sección se explicará todo lo relacionado con el aplicativo desarrollado para la autenticación de dos vías por medio de la huella dactilar para la implementación en el servicio de taxi, se observarán los aspectos de diseño del prototipo funcional y los elementos que los componen.

Descripción

La aplicación SecurityP es desarrollada para dispositivos Android, especialmente Tablet con el fin de ser usadas en el servicio de taxi para la autenticación por medio de huella dactilar y así garantizar la identidad de las personas que hacen parte del servicio.

La aplicación como prueba de concepto se concentra en hacer una validación de la identidad, además de registrar la fecha del servicio y los datos de los usuarios para que pueda quedar una constancia y posterior utilización de esta información.

En un primer intento el usuario se debe registrar en el aplicativo la huella digital con el nombre, la cédula y el correo, mientras que el conductor debe estar ya registrado.

La interfaz gráfica es sencilla para facilitar la compresión del usuario.

3.1. Elicitación

Se quiere brindar a los usuarios del aplicativo un servicio con el cual pueda sentirse confiado y tener la seguridad que va a llegar a su destino sano y salvo, además de registrar la información del servicio en caso de ser solicitada posteriormente, que sea de fácil manejo y sencilla.

Para cumplir con la seguridad y la confianza entre las partes, se requiere conocer e identificar la otra persona que hace parte del servicio, en el caso de un conductor la otra parte seria el pasajero y en el caso de este, sería el conductor, para poder brindar este servicio se requiere de una identificación que siempre vaya con los usuarios y que no sea fácilmente falsificable, por lo cual se requiere del uso de la biometría, específicamente la huella dactilar la cual se usa como verificación de la identidad al tomar el servicio de taxi junto con el número de la cédula, de tal forma que después de un registro previo muestre los datos de las personas y la huella dactilar, confirmando la identificación de la persona y de igual manera el conductor que va a estar previamente registrado con su foto y vehículo, los usuarios se autenticarán en presencia del otro para brindar mayor confianza.

El dispositivo de autenticación por medio de huella dactilar debe ser móvil para facilitar el manejo dentro del vehículo y facilitar la manipulación por parte del usuario y conductor.

La primera vez que un pasajero se registre debe ingresar el nombre, el número de cédula, el correo electrónico y la huella dactilar para realizar el registro y posteriormente verificar el registro, luego de la verificación del usuario se autentica el conductor con su número de cédula y huella, cuando es verificado muestra la foto del conductor y del taxi, validando la autenticidad.

El dispositivo tiene que contar con conectividad por medio de plan de datos celular, WIFI y GPS con el fin de transmitir la información de la huella dactilar para compararla con la base de datos y devolver la confirmación o denegación de identidad, adicionalmente se utiliza el GPS para establecer la posición donde se toma el servicio y la posición donde finaliza, junto con estos datos se envía al correo electrónico del usuario el tiempo del servicio con el fin de llevar el registro de los viajes.

En caso de no tener conexión el dispositivo muestra un mensaje indicando un error en la comunicación o en caso de no encontrar la huella envía un mensaje indicando que la validación no fue exitosa.

3.2. Requisitos del sistema

3.2.1. Requisitos funcionales

ID	Nombre	Descripción
RF1	Obtener información	Permite leer los datos de la persona.
RF2	leer huella dactilar	La aplicación debe leer la huella de los usuarios.
RF3	Procesar la huella	La aplicación procesa y transforma la huella.
RF4	Transmitir información	La aplicación debe transmitir la información obtenida a través de la red.
RF5	Verificar información	Compara y verificar la huella de los usuarios.
RF6	registrar usuarios	Permite el registro de nuevos pasajeros.
RF7	Registrar ubicación	La aplicación debe almacenar la ubicación inicial y final del servicio.
RF8	Enviar correo	Envía información del servicio al correo registrado.

Tabla 1: Estructura del framework grails.

3.2.2. Especificación de requisitos funcionales

Nombre	Obtener información
ID	RF1
descripción	Permite obtener la información del cliente por medio de cuadros de texto para ser verificada
Frecuencia	Siempre
Precondición	Estar ubicado en los cuadros de texto para ingresar la información, no estar vacío el campo.
Post condición	Ingresar otros datos o huella dactilar.
Excepción	Los campos no pueden estar nulos para obtener la información.

Tabla 2: Requisito – Obtener información.

Nombre	Leer huella dactilar
ID	RF2

descripción	Lee la huella dactilar desde el dispositivo de lectura en la Tablet para ser enviada y procesada.
Frecuencia	Siempre
Precondición	Presionar el botón leer huella antes de posicionar el dedo en el lector.
Post condición	Presionar el botón verificar huella para enviar la imagen al servidor.
Excepción	En caso de no detectar huella advierte sobre la no lectura de la huella.

Tabla 3: Requisito – Leer huella dactilar

Nombre	Procesar la huella
ID	RF3
descripción	Realiza el procesamiento de la imagen para extraer las minucias de la huella y las demás características únicas.
Frecuencia	Siempre
Precondición	Haber obtenido una lectura y trasmisión positiva de la huella dactilar.
Post condición	Tramite el resultado del procesamiento de las huellas dactilares enviadas por el dispositivo al proceso de verificación y comparación de las huellas.
Excepción	En caso de no recibir las huellas genera error al no tener los parámetros necesarios.

Tabla 4: Requisito – Procesar la huella

Nombre	Transmitir información
ID	RF4
descripción	Envía la información obtenida junto con la imagen de la huella dactilar al servidor, a través de plan de datos o WIFI.
Frecuencia	Siempre
Precondición	Tener la información y la imagen de la huella dactilar.

Post condición	N/A
Excepción	En caso de un error de comunicación muestra un mensaje indicando el error.

Tabla 5: Requisito – Transmitir información

Nombre	Verificar información
ID	RF5
descripción	Realiza la comparación de las huellas dactilares para determinar si cumplen con los patrones de similitud.
Frecuencia	Siempre
Precondición	Recibe la huella dactilar procesada para realizar la comparación de las huellas.
Post condición	Retorna respuesta positiva o negativa de comparación de las huellas dactilares.
Excepción	En caso de no poder enviar la respuesta intentar enviarla durante un tiempo corto y sino desecharla.

Tabla 6: Requisito – Verificar información

Nombre	Registrar pasajeros
ID	RF6
descripción	Permite el registro de los nuevos pasajeros solicitando el nombre completo, cédula, correo y huella dactilar.
Frecuencia	Siempre
Precondición	No permitir espacios nulos en el nombre y la cédula.
Post condición	Trasmitir la información al servidor
Excepción	Si falla el registro de la información muestra un mensaje de error indicando la falla.

Tabla 7: Requisito – Registrar pasajeros

Nombre	Registrar servicio
ID	RF7

descripción	Obtiene la ubicación actual y final por medio del GPS, plan de datos o WIFI, hora del servicio, duración y datos del taxi, como nombre del conductor, placa y empresa.
Frecuencia	Siempre.
Precondición	Haber iniciado el servicio en el aplicativo.
Post condición	Trasmitir la información al servidor.
Excepción	Si falla el registro de la información muestra un mensaje de error indicando la falla.

Tabla 8: Requisito – Registrar servicio

Nombre	Enviar correo.
ID	RF8
descripción	Envía un correo electrónico al usuario con la ubicación actual y final, hora del servicio, duración y datos del taxi, como nombre del conductor, placa y empresa.
Frecuencia	Siempre.
Precondición	Haber finalizado el servicio en el aplicativo.
Post condición	Trasmitir la información al servidor.
Excepción	Si falla el registro de la información muestra un mensaje de error indicando la falla.

Tabla 9: Requisito – Enviar correo.

3.2.3. Requisitos no funcionales

ID	Nombre	Descripción
RNF1	mostrar video	La aplicación debe mostrar un video mientras no se interactúa con la aplicación.
RNF2	Interfaz grafica	La interfaz gráfica debe ser amigable y fácil de manejar.
RNF3	Responsividad	La aplicación debe responder de manera ágil para el usuario.
RNF4	Abstracción de hardware	La aplicación pueda instalarse en varios dispositivos diferentes.

Tabla 10: Requisitos no funcionales.

4. Diseño

El sistema de autenticación por medio de huellas dactilares para el servicio de taxi funciona con un lector de huellas integrado en una Tablet ubicada en el interior de cada vehículo, el dispositivo tiene comunicación por medio de WIFI o plan de datos móvil con el fin de validar la información de la huella electrónica ante la base de datos centralizada.

El proceso de autenticación consiste en validar la huella dactilar ingresada para verificar y la huella almacenada anteriormente identificada con la cédula de la persona de esta manera se establece una relación entre la huella dactilar y la persona, se usa la plataforma web Parse para almacenar los registros y la huellas dactilares y mediante un servicio se envía al servidor de procesamiento y comparación la imagen tomada por el lector de huellas en formato JPG para ser procesada, analizada y comparada por los algoritmos MINDTCT y BOZORTH3 quien retorna la confirmación o fallo de la comparación de la información.

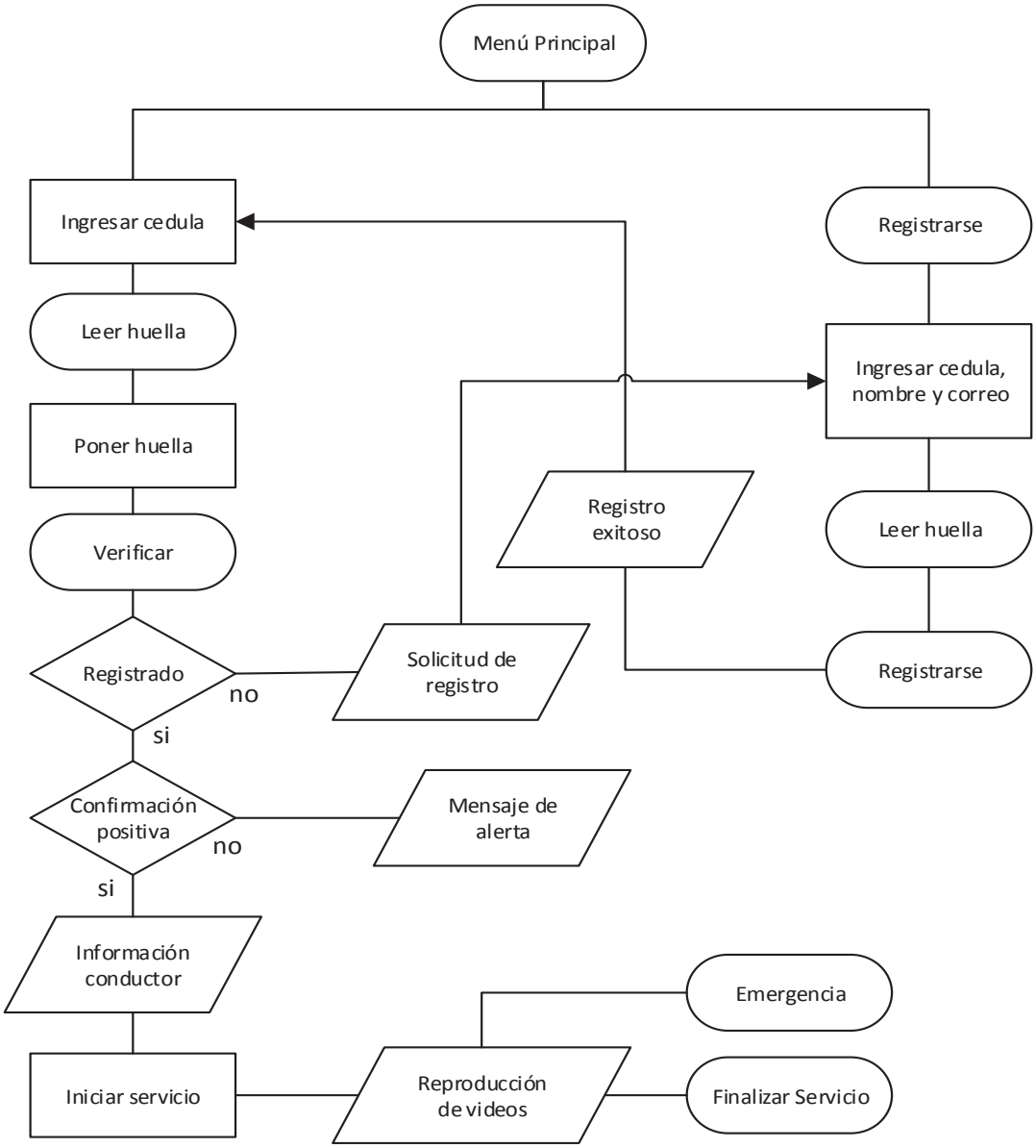


Ilustración 45: Diagrama de flujo del sistema.

4.1. Casos de uso

4.1.1. Identificación de actores del sistema

En este sistema identificamos los siguientes actores del sistema:

Conductor: es el usuario que estará previamente registrado y se encargará de guiar y usar el sistema de autenticación digital por medio de huella dactilar, podrá además verificar la información del pasajero, iniciar y parar el servicio.

Pasajero: es el usuario del sistema que en caso de no estar registrado podrá hacerlo con sus datos personales, además podrá autenticarse y verificar la información del conductor, además podrá iniciar el servicio y finalizarlo, recibirá al correo electrónico la información del servicio.

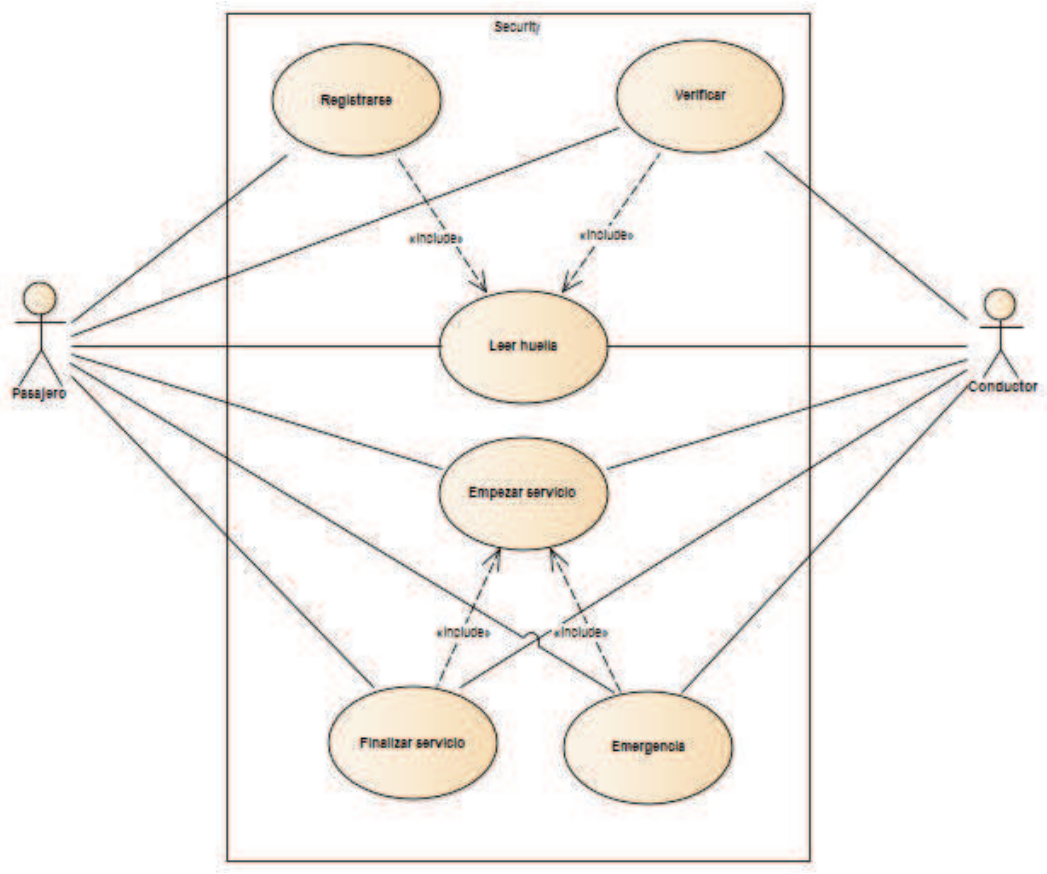


Ilustración 46: Diagrama de casos de uso.

4.1.2. Especificación de Casos de uso

Identificador	[CU01]
Nombre	Leer huella
resumen	Activa el sensor de lectura de huella y obtiene la imagen de la huella dactilar cuando el usuario pone el dedo sobre el sensor.
actor	Pasajero – Conductor
Precondición	Poner la huella sobre el sensor.
Post condición	Confirmación de lectura de huella
flujo normal	1) El usuario arranca el aplicativo y aparece una pantalla donde se le pide que introduzca la cédula y la huella dactilar. 2) El usuario introduce la información solicitada y le presiona leer huella. 3) El sistema comprueba que la huella se leyó correctamente y muestra la imagen obtenida.
flujo alternativo	Cancelar la lectura
inclusiones	Ninguna
extensiones	Ninguna

Tabla 11: Caso de uso - leer huella.

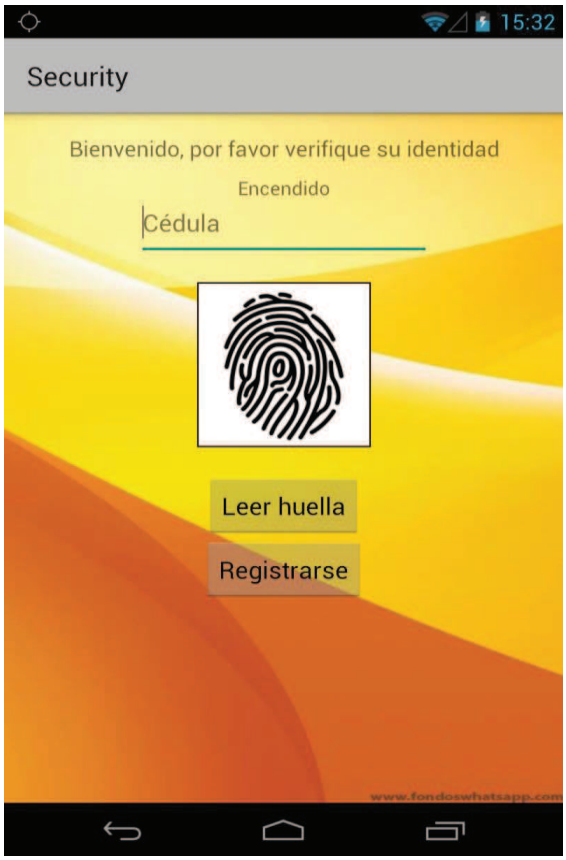


Ilustración 47: Menú principal.

Identificador	[CU02]
Nombre	Verificar
resumen	Trasmite la imagen de la huella dactilar y la cédula, para ser verificada y comparada por el servidor.
actor	Pasajero – Conductor
Precondición	Ingresar la cédula y leer la huella dactilar.
Post condición	Confirmación de identidad del usuario y presentación de información.
flujo normal	1) El usuario arranca el aplicativo y aparece una pantalla donde se le pide que introduzca la cédula y la huella dactilar. 2) El usuario introduce la información solicitada y le presiona leer huella.

	3) El sistema comprueba que la huella se leyó correctamente y muestra la imagen obtenida. 4) El usuario presiona verificar. 5) El sistema responde con la confirmación de la identidad y la información para el usuario.
flujo alternativo	Registrarse
inclusiones	[CU01] Leer huella
extensiones	Ninguna

Tabla 12: Caso de uso - Verificar

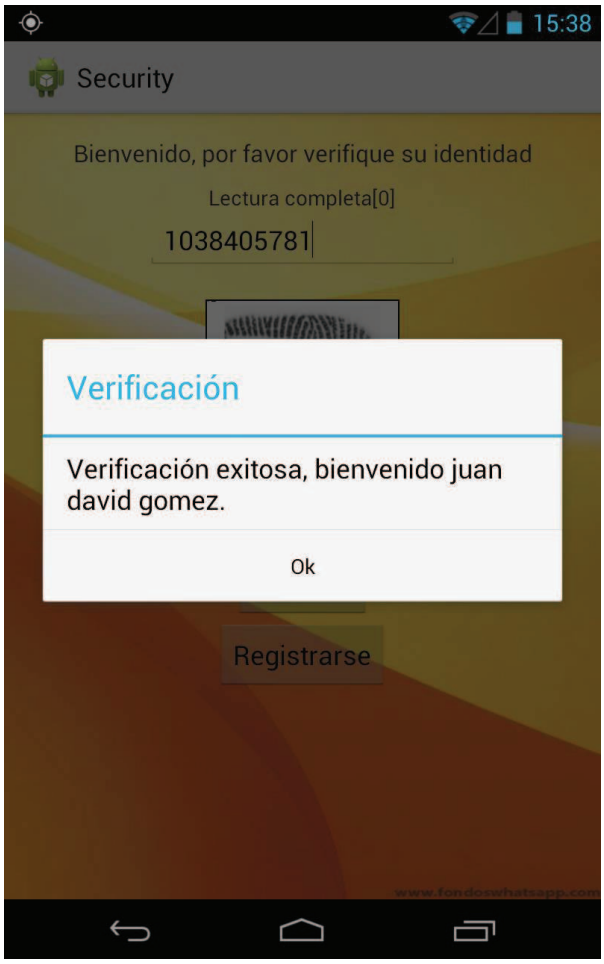


Ilustración 48: Verificación de usuario.

Identificador	[CU03]
Nombre	Registrarse
resumen	Obtiene la información necesaria del pasajero, cédula, nombre, correo e imagen de la huella dactilar.
actor	Pasajero.
Precondición	No estar registrado anteriormente en el sistema.
Post condición	Almacenar la información en la base de datos.
flujo normal	1) El usuario arranca el aplicativo y aparece una pantalla donde se le pide que introduzca la cédula y la huella dactilar. 2) El usuario presiona el botón registrarse y carga el menú de registro con los campos para introducir, el nombre, la cédula, el correo y la huella dactilar. 3) El usuario introduce la información solicitada y le presiona leer huella. 4) El sistema comprueba que la huella se leyó correctamente y muestra la imagen obtenida. 5) El usuario presiona nuevamente el botón registrarse. 6) El sistema confirma el registro exitoso.
flujo alternativo	Cancelar registro
inclusiones	[CU01] Leer huella
extensiones	Ninguna

Tabla 13: Caso de uso – Registrarse

Ilustración 49: Registro Usuario.

Identificador	[CU04]
Nombre	Empezar Servicio
resumen	Toma los datos de los usuarios y los almacena junto con la hora y la posición inicial del servicio, y carga el video preestablecido.
actor	Pasajero – Conductor
Precondición	Haber autenticado a los usuarios del servicio.
Post condición	Iniciar el temporizador del servicio.
flujo normal	1) El pasajero arranca el aplicativo y aparece una pantalla donde se le pide que introduzca la cédula y la huella dactilar. 2) El pasajero introduce la información solicitada y le presiona leer huella.

	3) El sistema comprueba que la huella se leyó correctamente y muestra la imagen obtenida. 4) El usuario presiona verificar. 5) El sistema responde con la confirmación de la identidad y la información para el usuario. 6) El sistema solicita la autenticación del conductor. 7) El conductor introduce la información solicitada y le presiona leer huella. 8) El sistema comprueba que la huella se leyó correctamente y muestra la imagen obtenida. 9) El usuario presiona verificar. 10) El sistema responde con la confirmación de la identidad y la información para el usuario. 11) El usuario presionar el botón empezar servicio. 12) El sistema almacena la información del servicio y comienza la reproducción del video y el contador de tiempo.
flujo alternativo	Ninguno
inclusiones	Ninguno
extensiones	Ninguna

Tabla 14: Caso de uso – Empezar servicio

Ilustración 50: Información conductor.

Identificador	[CU05]
Nombre	Finalizar Servicio
resumen	Toma los datos del servicio y les agrega la duración del mismo y la posición final, además envía un correo electrónico al usuario con los datos obtenidos del servicio.
actor	Pasajero – Conductor
Precondición	Haber iniciado el servicio
Post condición	Enviar el correo al usuario con la información del servicio.
flujo normal	1) El pasajero arranca el aplicativo y aparece una pantalla donde se le pide que introduzca la cédula y la huella dactilar.

	2) El pasajero introduce la información solicitada y le presiona leer huella. 3) El sistema comprueba que la huella se leyó correctamente y muestra la imagen obtenida. 4) El usuario presiona verificar. 5) El sistema responde con la confirmación de la identidad y la información para el usuario. 6) El sistema solicita la autenticación del conductor. 7) El conductor introduce la información solicitada y le presiona leer huella. 8) El sistema comprueba que la huella se leyó correctamente y muestra la imagen obtenida. 9) El usuario presiona verificar. 10) El sistema responde con la confirmación de la identidad y la información para el usuario. 11) El usuario presionar el botón empezar servicio. 12) El sistema almacena la información del servicio y comienza la reproducción del video y el contador de tiempo. 13) El usuario presiona Finalizar servicio. 14) El sistema guarda la posición final y para el contador de tiempo, además envía un correo al usuario con la información del servicio.
flujo alternativo	Emergencia
inclusiones	[CU04] Empezar Servicio
extensiones	Ninguna

Tabla 15: Caso de uso – Finalizar servicio

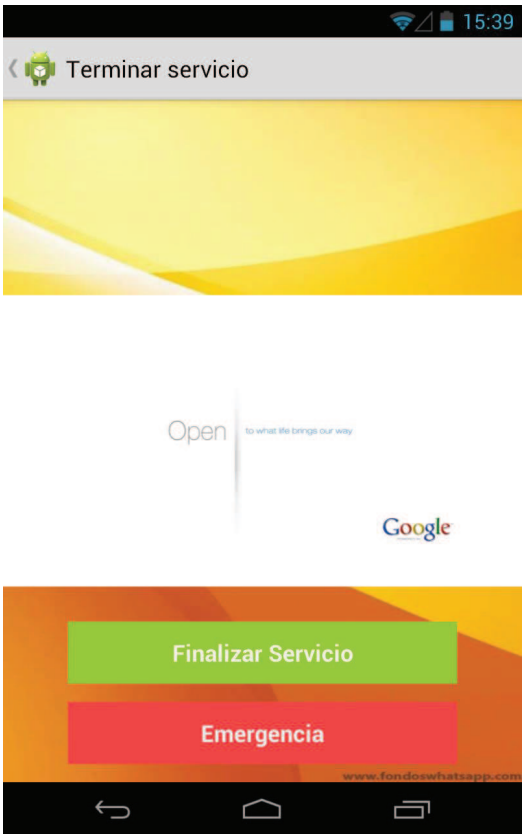


Ilustración 51: Video y Finalización de servicio.

Identificador	[CU06]
Nombre	Emergencia
resumen	Toma los datos del servicio y les agrega la duración del mismo y la posición final, además envía un correo electrónico a un correo específico con los datos obtenidos del servicio.
actor	Pasajero – Conductor
Precondición	Haber iniciado el servicio
Post condición	Enviar el correo al usuario con la información del servicio.
flujo normal	1) El pasajero arranca el aplicativo y aparece una pantalla donde se le pide que introduzca la cédula y la huella dactilar. 2) El pasajero introduce la información solicitada y le presiona leer huella.

	3) El sistema comprueba que la huella se leyó correctamente y muestra la imagen obtenida. 4) El usuario presiona verificar. 5) El sistema responde con la confirmación de la identidad y la información para el usuario. 6) El sistema solicita la autenticación del conductor. 7) El conductor introduce la información solicitada y le presiona leer huella. 8) El sistema comprueba que la huella se leyó correctamente y muestra la imagen obtenida. 9) El usuario presiona verificar. 10) El sistema responde con la confirmación de la identidad y la información para el usuario. 11) El usuario presionar el botón empezar servicio. 12) El sistema almacena la información del servicio y comienza la reproducción del video y el contador de tiempo. 13) El usuario presiona Emergencia. 14) El sistema guarda la posición final y para el contador de tiempo, además envía un correo a la cuenta del sistema con la información del servicio.
flujo alternativo	Finalizar Servicio
inclusiones	[CU04] Empezar Servicio
extensiones	Ninguna

Tabla 16: Caso de uso – Emergencia.

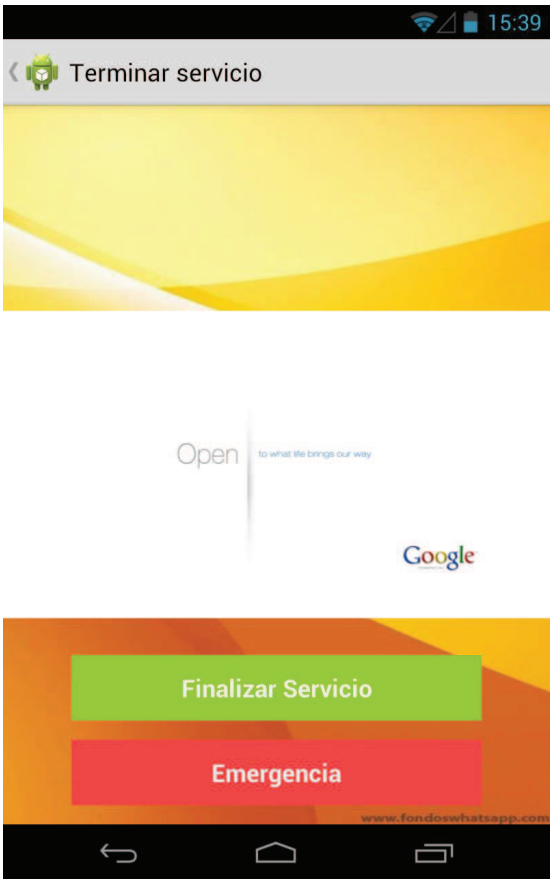


Ilustración 52: Terminar Servicio.

4.1.3. Diagrama de Componentes

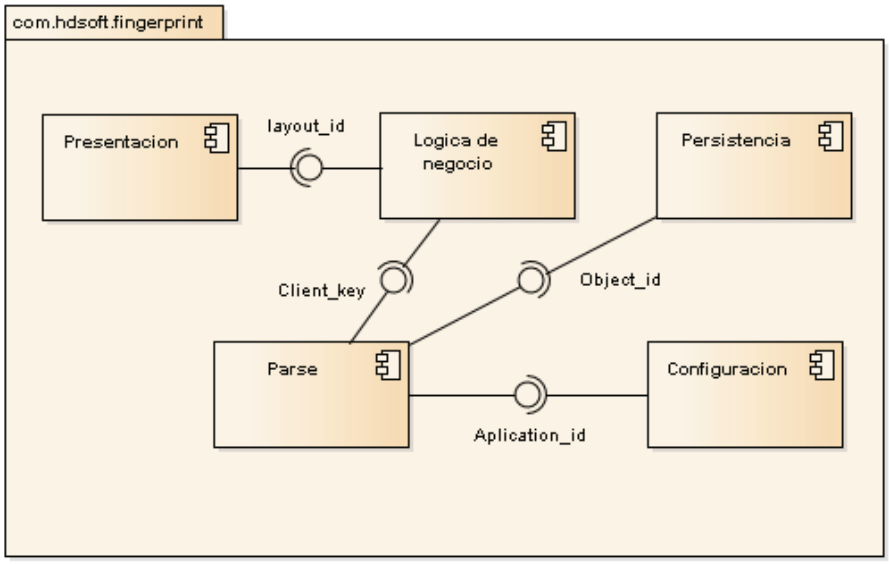


Ilustración 53: Diagrama de Componentes.

4.1.4. Diagrama de procesos

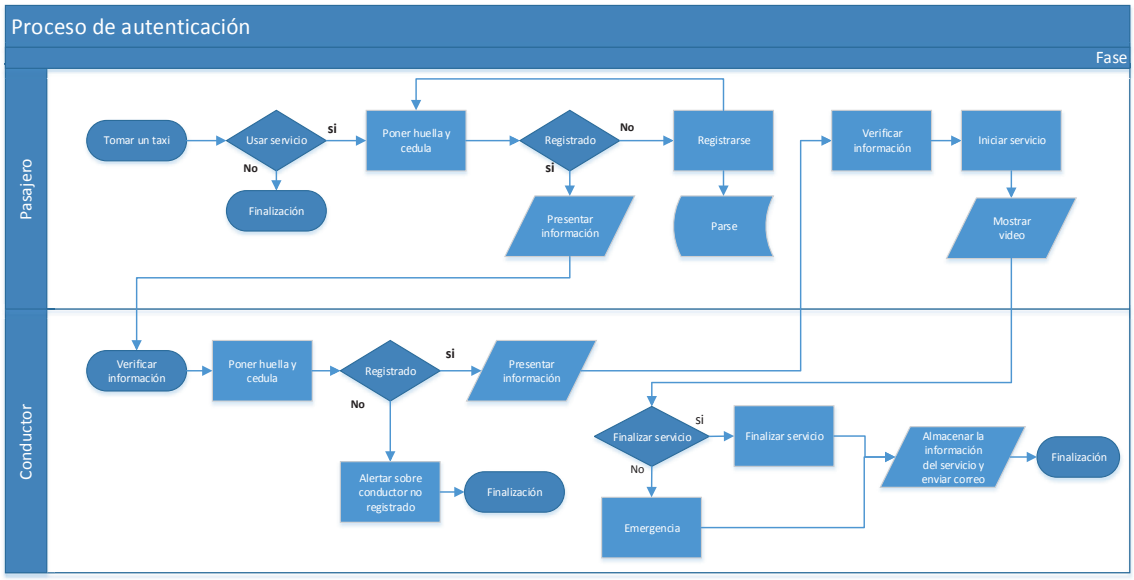


Ilustración 54: Diagrama de procesos.

5. Implementación

5.1. Arquitectura

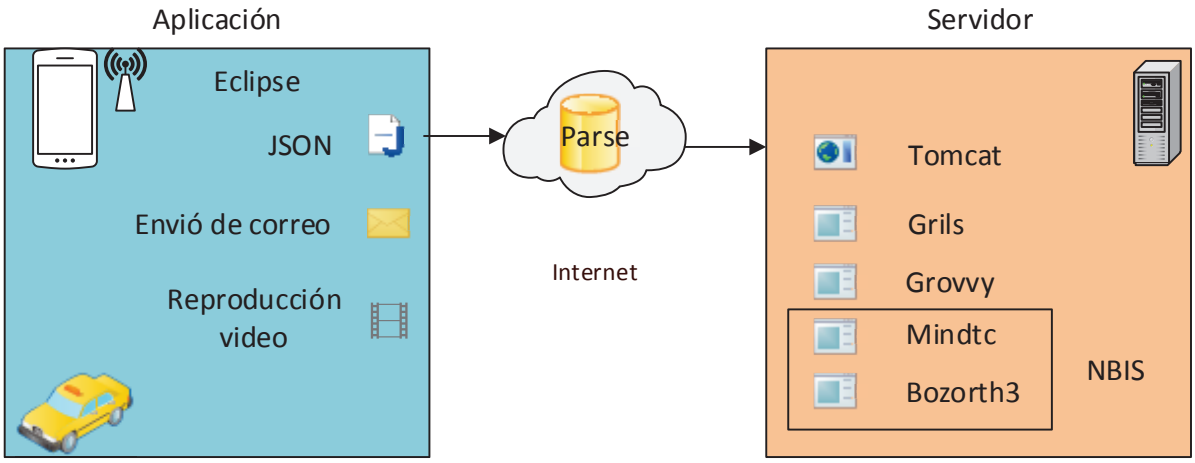


Ilustración 55: Arquitectura del sistema.

5.2. Aplicación

La mayoría de las aplicaciones Android deben su existencia a Eclipse (34). Es un IDE de código abierto (entorno de desarrollo integrado) para proyectos de Java. Básicamente, el lugar donde la aplicación se hace a mano, siendo apoyado por diversas etapas de su ciclo de vida. Google oficialmente apoya, y ha creado el *Android Development Tools Plugin* para Eclipse e integrado a su AVD Administrador de gestión de dispositivo virtual.

En otras palabras, no sólo se puede construir programas Java, sino rápidamente crear código orientado a Android, y probar con el apoyo de los emuladores (los dispositivos virtuales) que muestran cómo el código se ejecutaría en versiones estandarizadas de dispositivos Android (en el base a su nivel de API, en lugar de detallar cada teléfono).

Plugin ADT

Android Development Tools (ADT) es un plugin para el IDE Eclipse que está diseñado para darle un poderoso entorno integrado para construir aplicaciones Android. ADT amplía las capacidades de Eclipse para que pueda configurar rápidamente nuevos proyectos para Android, crear una aplicación IU, añadir componentes basados en la API de Android Framework, depurar sus aplicaciones utilizando las herramientas del SDK de Android, e incluso exportar archivos firmados .apk con el fin de distribuir la aplicación.

El desarrollo en Eclipse con ADT es muy recomendable y es la manera más rápida de comenzar. Así como la integración de herramientas, editores XML personalizados, y el panel de salida de depuración, ADT da un impulso increíble en el desarrollo de aplicaciones de Android.

Administrador de dispositivos Android Virtual

La AVD Manager es una herramienta de interfaz de usuario fácil de usar para administrar sus (dispositivo virtual Android) configuraciones AVD. Un AVD es una configuración de dispositivo para el emulador de Android que permite modelar diferentes configuraciones de dispositivos con Android. Al iniciar el Administrador de AVD en Eclipse o ejecutar la herramienta de Android en la línea de comandos, verá la AVD Manager como se muestra

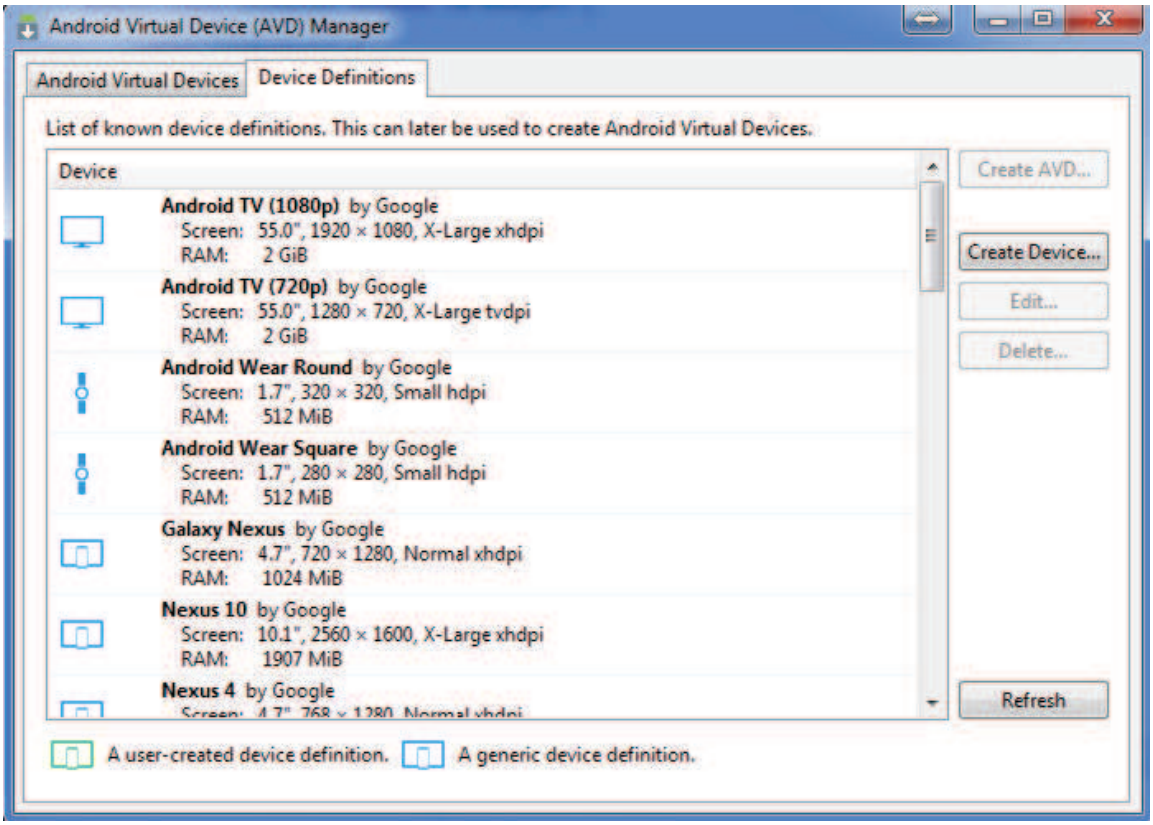


Ilustración 56: Android Virtual Device (AVD).

Este mecanismo de plug-in es un framework de software ligero. Además de permitir que Eclipse se extienda usando otros lenguajes de programación como C y Python, el framework plug-in permite a Eclipse trabajar con lenguajes de composición tipográfica como LaTeX, aplicaciones como sistemas de gestión de bases de telnet y la creación de redes. La arquitectura plug-in admite escribir cualquier extensión deseada con el medio ambiente, tales como la gestión de configuraciones. Soporte para Java y CVS proporcionada por el SDK de Eclipse, y con soporte para otros sistemas de control de versiones proporcionadas por plug-ins de terceros.

La construcción de los proyectos de Android

En cuanto a la construcción de proyectos de Android, la secuencia necesaria de eventos es la siguiente:

- Descargar SDK Java SE
- Descargar el SDK de Android
- Descargar el IDE de Eclipse (versión clásica)
- Descargar el ADT Plugin para Eclipse

El Plugin central de Eclipse se puede encontrar en marketplace.eclipse.org/, donde, por ejemplo entre los muchos disponibles relacionados con Android, se puede encontrar "*Testdroid Recorder*", lo que "permite una fácil prueba UI automatizado de aplicaciones Android. Utilice la aplicación para registrar los casos de prueba y reproducir en cualquier dispositivo o emulador de Android.

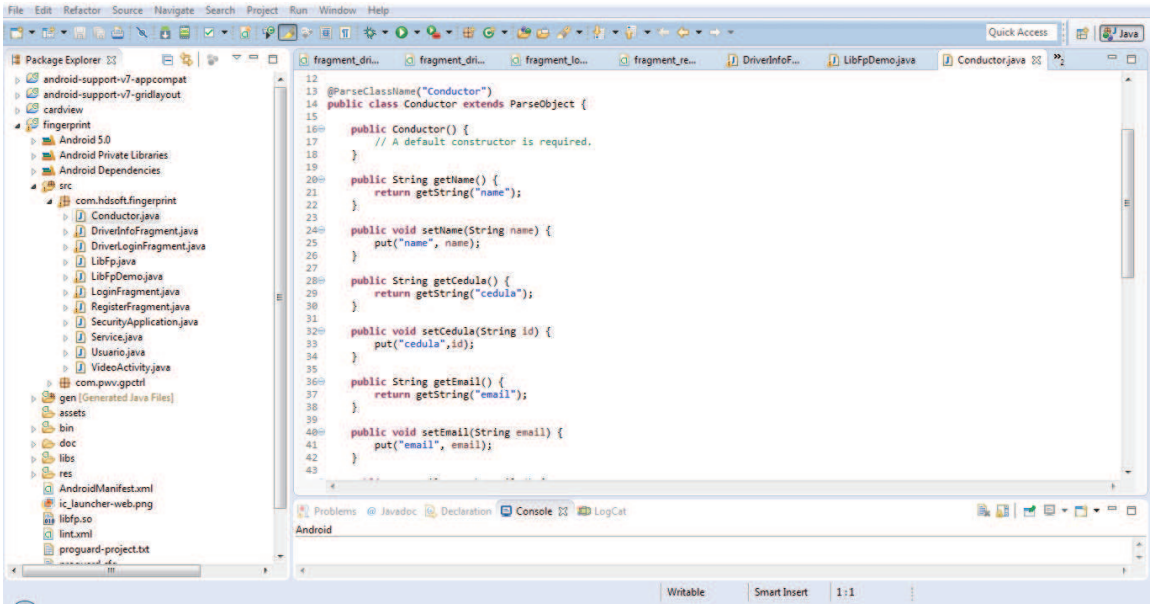


Ilustración 57: Eclipse.

Definición de Clases en la aplicación

Conductor.java
Estructura de lo que sería el conductor para la aplicación.

DriverinfoFragment.java
Muestra la información del conductor

DriverloginFragment.java
Es la vista para realizar el login del conductor, donde se realiza el procedimiento de verificación del conductor

LibFp.java
Clase con las librerías necesarias para entender la huella dactilar.

LibFpDemo.java
Es la actividad principal donde se hace la parte de las verificaciones, es la clase tipo main de la aplicación, en esta clase se hace la verificación del usuario, el registro del usuario, la verificación del taxista, mostrar la información del taxista, usando la librería tanto LibFp.java como el resto de librerías, implementando los métodos de las otras vistas, y se dice que lo implementa porque llama los métodos desde los fragmentos.

LoginFragment.java
Es la vista para realizar el login del usuario, donde se realiza el procedimiento de verificación del conductor

RegisterFragment.java
Es la clase que permite realizar el registro de los pasajeros a la aplicación, preguntando la cédula, nombre, correo y la huella dactilar.

SecurityApplication.java
Clase para configurar la aplicación en Parse, donde sirve para trasformar las variables de tal forma que puedan ser entendidas por Parse y conectarlo con los accesos dados por Parse.

Service.java
Es la clase que se utiliza para registrar los datos del servicio, como el tiempo, la ubicación, los usuarios del servicio.

Usuario.java
Estructura de lo que sería el usuario para la aplicación

VideoActivity.java
Realiza el proceso de reproducción del video durante el servicio como una actividad independiente.

5.3. Servidor

En esta sección se explicará los componentes que se instalaron en el servidor y como se realizó el servicio web utilizando el framework grails que explota el uso de groovy como lenguaje potenciado para JVM y publicado en tomcat, JSON para trasmitir la información de manera ágil y el uso del software biométrico de la NIST “NBIS” para el procesamiento de la huella dactilar y la comparación;

El código de parte del servidor se realizó en groovy que es un es un lenguaje ágil y dinámico para la máquina virtual de Java. Se basa en los puntos fuertes de Java, pero tiene características adicionales de energía inspirados en lenguajes como *Python, Ruby y Smalltalk.* Hace las funciones de programación modernas disponibles para los desarrolladores de Java con casi cero curva de aprendizaje. Proporciona la capacidad para verificación de tipo estático y compilación estáticamente del código con robustez y rendimiento. Apoya lenguajes específicos de dominio y otra sintaxis compacta, por lo que el código se convierte en fácil de leer y mantener. Hace que la escritura de shell y scripts de creación fáciles, con sus potentes primitivas de procesamiento, capacidades programación orientada a objetos y *Ant DSL*. Aumenta la productividad de los desarrolladores mediante la reducción de código *scaffold* en el desarrollo web, GUI, bases de datos o aplicaciones de consola. Simplifica pruebas mediante pruebas de unidades y bocetos fuera de la caja. Se integra perfectamente con todas las clases y las bibliotecas *Java* existentes. Compila directamente a código de bytes de Java para que pueda utilizarlo en cualquier lugar se puede utilizar Java (35).

Grails es un código abierto, pila completa, *framework* de aplicaciones web para la JVM. Aprovecha el lenguaje de programación *Groovy* y convenciones sobre configuración para proporcionar una experiencia de desarrollo productivo y aerodinámico (36).

Apache Tomcat es una implementación de software de código abierto de las tecnologías *Java Servlet y JavaServer Pages*. Las especificaciones *Java Servlet y JavaServer Pages* se desarrollan bajo *Java Community Process*. *Apache Tomcat* se desarrolla en un entorno abierto y participativo y liberado bajo la licencia *Apache* versión 2 (37).

El NIST Biometric Image Software (NBIS) la distribución es desarrollado por el Instituto Nacional de Estándares y Tecnología (NIST) de la Oficina Federal de Investigaciones (FBI) y el Departamento de Seguridad Nacional (DHS). Este software ha sido determinado para estar fuera del alcance de la Export Administration Regulation (EAR), ya que ha sido creado exclusivamente por empleados del Gobierno de los Estados Unidos; se distribuye libremente, sin requisitos de concesión de licencias; y se considera de dominio público. Por lo tanto, es permisible para distribuir este software como una descarga gratuita desde Internet. La distribución NBIS contiene código fuente C para su uso con el procesamiento y análisis de datos biométricos (38).

JSON (JavaScript Object Notation) es un formato de intercambio de datos ligero. Es fácil para los seres humanos leer y escribir. Es fácil para las máquinas analizar y generar. Se basa en un subconjunto del lenguaje de programación JavaScript.

JSON es un formato de texto que es completamente independiente del lenguaje, pero utiliza las convenciones que son familiares para los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen JSON un lenguaje ideal para intercambio de datos (39).

El framework Grails tiene una estructura especifica de archivos para el manejo de las librerías y compilaciones nativas, en la siguiente ilustración se observa la estructura base.

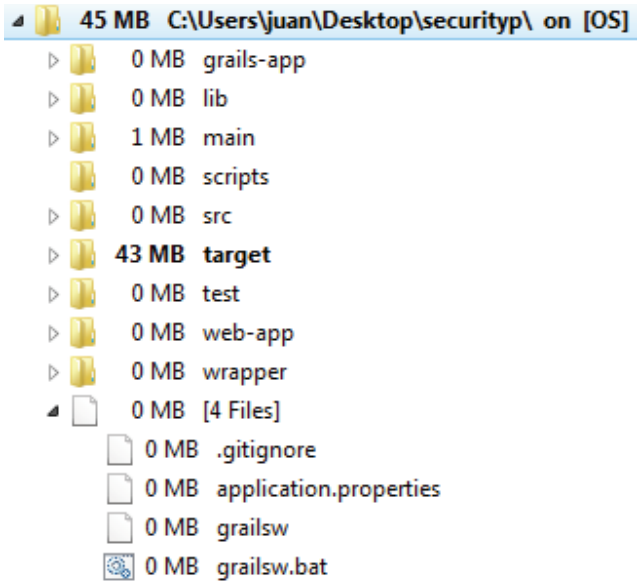


Ilustración 58: Estructura del framework grails.

En esta estructura del framework se destacan esencialmente 4 archivos de código los cuales se explicarán a continuación y un paquete .war generado para ser cargado al tomcat para publicar el servicio, los archivos son:

LoginController.groovy: Recibe el post tipo JSON desde el aplicativo.

FPValidatorService.groovy: Es el archivo principal que llama al extractor de minucias y comparador de minucias

MinutiaExtractor.groovy: Procesa la imagen de las huellas dactilares y genera los archivos en texto plano para ser comparado.

MinutiaComparator.groovy: Compara las minucias para determinar el grado de similitud entre las huellas.

A continuación se explicará el código de cada uno de estos archivos.

5.3.1. LoginController.groovy

Recibe la solicitud de servicio de comparación y es el encargado de enviar la respuesta positiva o negativa en caso de no pasar el nivel de paridad entre las huellas.

Ruta: Securityp/grails-app/controllers/fpvalidator/

Código:

```
package fpvalidator

class LoginController {

   static format = ['json']

   def FPValidatorService

//Recibe la solicito JSON y la envía a FPValidatorService y devuelve el resultado 200 si es positiva la comparación y 401 se es negativa la comparación.

   def index() {
      def json = request.getJSON()
      final img1_url = json['img1']
      final img2_url = json['img2']
      def resp = ['url1':img1_url, 'url2': img2_url]
      if (FPValidatorService.validate(img1_url, img2_url))
         render status: 200
      else
         render status:401
   }

}
```

5.3.2. FPValidatorService.groovy

Este archivo realiza el llamado a los métodos de extracción de minucias y comparación, y determina el límite mínimo en el que se establece una comparación positiva.

Ruta: securityp/grails-app/services/fpvalidator/

Código:

```
package fpvalidator

import co.jdg.fpvalidator.MinutiaComparator
import co.jdg.fpvalidator.MinutiaExtractor
import grails.transaction.Transactional
```

```
import java.nio.file.Path
import java.nio.file.Paths

@Transactional
class FPValidatorService {

//establece la ruta de las imágenes a comparar y el valor mínimo permitido para confirmar las similitudes de las huellas dactilares.

    public static final String minutiaTemplatePath = "/tmp/minext"
    public static final int BOZORTH_THRESHOLD = 40

//Recibe las dos imágenes y las descarga con downloadImage, ejecuta el procesamiento de las imágenes, ejecuta el comparador de minucias y devuelve el resultado estableciendo si paso el umbral permitido anteriormente.

    def Boolean validate(url1, url2) {
        def img1Path = downloadImage(url1)
        def img2Path = downloadImage(url2)
        extractMinutia(img1Path, img2Path)
        def bozorthScore = compareMinutia(img1Path, img2Path)
        return bozorthScore >= BOZORTH_THRESHOLD
    }
//Recibe las imágenes y las procesa de a una en la ruta establecida, haciendo llamado al método MinutiaExtractor

    private void extractMinutia(Path img1Path, Path img2Path) {
        def extractor1 = new MinutiaExtractor.Builder(Paths.get(minutiaTemplatePath), img1Path).build()
        extractor1.process()
        def extractor2 = new MinutiaExtractor.Builder(Paths.get(minutiaTemplatePath), img2Path).build()
        extractor2.process()
    }

//toma los dos templates generados por el proceso de extracción de minucias y las compara hacienda llamado al método MinutiaComparator.

    private int compareMinutia(Path img1Path, Path img2Path){
        def minutia1Path = Paths.get(minutiaTemplatePath + '/MinTemplates/' +
getMinutiaTemplateName(img1Path))
        def minutia2Path = Paths.get(minutiaTemplatePath + '/MinTemplates/' +
getMinutiaTemplateName(img2Path))
        MinutiaComparator.execute(minutia1Path, minutia2Path)
    }

//Obtiene la imagen desde la ruta con el nombre de la imagen agregándole _m.xyt que es el formato en que lo exporta el método MinutiaExtractor

    private String getMinutiaTemplateName(Path imgFileName){
        return imgFileName.getFileName().toString().substring(0,
imgFileName.getFileName().toString().indexOf('.')) + '_m.xyt'
    }
// Descarga la imagen desde la url que llega.

    protected Path downloadImage(String address){
        Path filePath = Paths.get("/tmp/" + address.tokenize("/")[-1] + '.bmp')
        def file = new FileOutputStream(filePath.toFile())
        def out = new BufferedOutputStream(file)
        out << new URL(address).openStream()
        out.close()
        return filePath
    }
```

```
}
```

5.3.3. MinutiaExtractor.groovy

Este archive realiza la extracción y procesamiento de las minucias de la huella dactilar, generando los puntos característicos de cada huella y devolviendo un archivo con dichos puntos.

Ruta: securityp/src/groovy/co/jdg/fpvalidator/

Código:

```
package co.jdg.fpvalidator

import nij.qrfrp.extract.ExtractionManager
import nij.qrfrp.extract.RidgeExtractor
import nij.qrfrp.util.ImageUtil

import javax.imageio.ImageIO
import java.nio.file.Path

class MinutiaExtractor {

    private final processMin
    private final processRidge
    private final dpi
    private final outputDirectory
    private final image
    private final minDir
    private final diagnosticsDir
    private inputImage


    public static class Builder{
        private processMin = true
        private processRidge = false
        private dpi = 400
        private outputDirectory
        private image
        private minDir
        private diagnosticsDir

        public Builder(Path outputDirectory, Path image){
         this.outputDirectory = outputDirectory.toFile()
         this.image = image.toFile()
         this.minDir = new File(this.outputDirectory.path + File.separator + "MinTemplates/");
         this.diagnosticsDir = new File(this.outputDirectory.path + File.separator+"Diagnostics/");
        }

        public Builder withDPI(int dpi){
            this.dpi = dpi
            return this
        }

        public Builder withProcessRidge(Boolean processRidge){
            this.processRidge = true
```

```
            return this
        }

        public Builder withProcessMinutia(Boolean processMinutia){
            this.processMin = true
            return this
        }


        public MinutiaExtractor build () {
            return new MinutiaExtractor(this)
        }
    }

    private MinutiaExtractor(Builder builder){
        this.processMin = builder.processMin
        this.processRidge = builder.processRidge
        this.dpi = builder.dpi
        this.outputDirectory = builder.outputDirectory
        this.image = builder.image
        this.minDir = builder.minDir
        this.diagnosticsDir = builder.diagnosticsDir
    }

//Crea los directorios si no existen y carga la imagen y la procesa.
    public process(){
        createMinutiaOutputDirectory()
        loadAndProcessImage()
    }

    private void createMinutiaOutputDirectory() {
        if(!outputDirectory.isDirectory())
            outputDirectory.mkdirs();
        if (!minDir.exists() && processMin)
            minDir.mkdirs();
        if(!diagnosticsDir.exists())
            diagnosticsDir.mkdirs();
    }

//Carga la imagen, la convierte en escala de grises, extrae las minucias y las crestas de la huella dactilar,
procesa la información y genera los templates de cada huella.

    private void loadAndProcessImage(){
        this.inputImage = ImageIO.read(this.image)
        if(inputImage == null)
        {
            System.out.println("Image could not be read");
            throw RuntimeException("Image could not be read")
        }

        if(inputImage.getData().getNumBands()!=1)
        {
            System.out.println("Number of bands!=1; converting to gray.");
            try{
                inputImage = ImageUtil.convertRGBToGray(inputImage)
            }catch(Exception ee)
            {
                throw RuntimeException("Image could not be converted to gray")
            }
        }

        String root = image.getName().substring(0, image.getName().indexOf("."));
```

```
        File diagnosticsSubDir = new File(diagnosticsDir.path + File.separator + root + File.separator)

        def ext = new ExtractionManager(inputImage, dpi)
        if(processMin)
        {
            if(!processRidge)
            {
                ext.extractRidges();
                ext.getRidgeExtr().writeDiagnosticImages(diagnosticsSubDir, root)
                ext.getRidgeExtr().nullImages()
                System.gc()
            }
            ext.extractMinutiae()

if(Boolean.parseBoolean(ext.getRidgeExtr().getConf().get(RidgeExtractor.RidgeExtractorConf.Key.DIAGNOSTICS_KEY)))
                ext.getMinExtr().writeDiagnosticImages(diagnosticsSubDir, root)
            ext.getMinExtr().nullImages()

            if(!processRidge)
                ext.setRidgeExtr(null)
            System.gc()
        }
        ext.dumpTtemplates(null, null, minDir, null, root);

    }

}
```

5.3.4. MinutiaComparator.groovy

Este archivo realiza la comparación de los archivos devueltos por el código anterior y determinando si cumplen o no con el margen de comparación para determinar la paridad de las huellas.

Ruta: securityp/src/groovy/co/jdg/fpvalidator/

Código:

```
package co.jdg.fpvalidator

import java.nio.file.Path

import java.nio.file.Paths


class MinutiaComparator {

//ejecuta el método Bozorth3.exe con los parámetros indicados en execute.

    public static final String EXECUTABLE_PATH = "./main/lib/bozorth3.exe";
    private static final boolean DEBUG = false;

//ejecuta el comando con la opción –p para comparar, seguido de los templates generados por el método MinutiaExtractor, y retorna el valor de comparación entre los templates.
```

```
    public static int execute(Path minutia1, Path minutia2) throws FileNotFoundException
    {
        File executable = Paths.get(EXECUTABLE_PATH).toFile()
        if(!executable.exists())
            throw new FileNotFoundException("Bozorth3 executable not found. ")

        String arg1 = "", arg2="", arg3="";
        arg1 = "-p"
        arg2 = minutia1.toFile().path
        arg3 = minutia2.toFile().path

        String[] command = null

        List<String> argList = new Vector<String>();
        argList.add(executable.path);
        argList.add(arg1)
        argList.add(arg2)
        argList.add(arg3)
        command = argList.toArray(new String[argList.size()]);
        if(DEBUG) System.out.println("COMMAND> " + java.util.Arrays.toString(command String[] envp = null;

        Runtime rt = Runtime.getRuntime();

        try {
            Process proc = rt.exec(command, envp);
            if(DEBUG) System.out.println("Bozorth started.");
            int exitVal = proc.waitFor();
            String bozorthVal = proc.getText()
            if(DEBUG) System.out.println("Bozorth return value = " + exitVal);
            return Integer.parseInt(bozorthVal.trim())

        } catch(Exception e){

            System.out.println("Error");
            e.printStackTrace();
        }
    }

}
```

6. Pruebas

6.1. Objetivo y aceptación de las pruebas

El principal objetivo de la prueba es medir los tiempos de respuesta con diferentes tecnologías disponibles, con el fin de determinar si se ajustan los tiempos a un valor inferior a los quince segundos por cada autenticación y en total a un minuto por todo el proceso, contando tiempos de espera entre autenticaciones, ya que el servicio sería poco aceptado si el proceso es lento y por lo tanto la prueba de concepto sería un fracaso. La identificación exitosa y en corto tiempo de las personas, facilita la masificación del aplicativo, lo que conlleva a tener un mayor número de registros de los servicios que se realicen, permitiendo dejar en evidencia los delincuentes y persuadir a los mismos de ejecutar las acciones punibles.

6.2. Pruebas de procesamiento de la huella.

Estas pruebas se realizan directamente en el servidor que se encarga de hacer la comparación de las dos huellas dactilares, la registrada en el servidor y la que se ingresa al momento del servicio y se contabiliza sumando los tres procesos ejecutados a cada huella dactilar, que son image preprocessing time, ridge extraction time y minutiae extraction, la suma de los 6 valores es el tiempo de procesamiento de la imagen antes del envío al dispositivo.

Los datos obtenidos en el muestreo son:

Autenticación usuario (seg)	Autenticación conductor (seg)
0,657	0,634
0,638	0,667
0,66	0,687
0,695	0,646
0,625	0,684
0,675	0,624
0,638	0,689
0,687	0,656
0,691	0,681

0,645	0,629
0,674	0,683
0,658	0,674
0,67	0,693
0,692	0,68
0,637	0,655
0,688	0,671
0,672	0,662
0,638	0,683
0,668	0,653
0,695	0,664
0,684	0,663
0,663	0,633
0,633	0,67
0,695	0,634
0,625	0,634

Tabla 17: Procesamiento huella.

Des. estándar	0,023682483	0,021245549
Mediana	0,668	0,664
Promedio	0,66412	0,66196
Mínimo	0,625	0,624
Máximo	0,695	0,693

Tabla 18: Procesamiento huella (estadísticas en segundos)

De estos datos podemos obtener que la desviación estándar del servidor es mínima debido a que el servidor está dedicado solo a esperar la solicitud de comparación de las huellas y cuenta con los suficientes recursos para procesarla rápidamente, con un promedio de 0,66412 segundos para la autenticación del usuario y 0,66196 segundos para la autenticación del conductor.

A continuación se muestra el registro del servidor de una de las huellas de donde se sacó la información se muestra.

==> /var/log/tomcat7/catalina.out <==

config parameters set: {ip_gaussianvarianceperdpi=0.001, ip_verbose=true, ip_minimumdiagnostics=false, ip_macroblocksizeperdpi=0.015, ip_diagnostics=false}

BEGIN IMAGE PREPROCESSOR:
---
image processed with gaussian filter.
image processed with block equalization filter.
image processed using level adjustment.
ridge flow successfully calculated.
image preprocessing time = 0.039 seconds.

config parameters set: {re_minimumdiagnostics=false, re_threshold=127, re_verbose=true, re_diagnostics=false, re_pixelspacingperdpi=0.004, re_borderperdpi=0.006, re_minimumblobfillsizeperdpi=0.0, mindtctpath=./main/lib/mindtct.exe, re_microblocksizeperdpi=0.002, re_maximumblobfillsizeperdpi=0.125, re_suppressionradiusperdpi=0.015}

BEGIN RIDGE EXTRACTION:
---
image thresholded.
blobs filled in.
image edged with sobel filter.
edges thinned using Hildtich.
border applied to image.
local ridge orientation calculated.
Number of ridge contour segments = 724.
Ridge strength = 120.39832024263163.
Core displacement = 36.6742416417845.
Ridge to valley ratio = 0.7988581466842336.
template created.
ridge extraction time = 0.253 seconds.

BEGIN MINUTIAE EXTRACTION:
---
image resized and written to temporary location.
mindtct executed.
mindtct output successfully created.
xyt file loaded.

minutiae coordinates plotted.
template created.
minutiae extraction time = 0.069 seconds.

config parameters set: {ip_gaussianvarianceperdpi=0.001, ip_verbose=true, ip_minimumdiagnostics=false, ip_macroblocksizeperdpi=0.015, ip_diagnostics=false}

BEGIN IMAGE PREPROCESSOR:
---
image processed with gaussian filter.
image processed with block equalization filter.
image processed using level adjustment.
ridge flow successfully calculated.
image preprocessing time = 0.038 seconds.

config parameters set: {re_minimumdiagnostics=false, re_threshold=127, re_verbose=true, re_diagnostics=false, re_pixelspacingperdpi=0.004, re_borderperdpi=0.006, re_minimumblobfillsizeperdpi=0.0, mindtctpath=./main/lib/mindtct.exe, re_microblocksizeperdpi=0.002, re_maximumblobfillsizeperdpi=0.125, re_suppressionradiusperdpi=0.015}

BEGIN RIDGE EXTRACTION:
---
image thresholded.
blobs filled in.
image edged with sobel filter.
edges thinned using Hildtich.
border applied to image.
local ridge orientation calculated.
Number of ridge contour segments = 613.
Ridge strength = 122.0885784716516.
Core displacement = 88.0056816347672.
Ridge to valley ratio = 0.5959823361329986.
template created.
ridge extraction time = 0.231 seconds.

BEGIN MINUTIAE EXTRACTION:
---
image resized and written to temporary location.
mindtct executed.

mindtct output successfully created.
xyt file loaded.
minutiae coordinates plotted.
template created.
minutiae extraction time = 0.065 seconds.

==> /var/log/tomcat7/localhost_access_log.2014-11-23.txt <==
181.51.107.27 - - [23/Nov/2014:11:37:12 -0500] "POST /fpvalidator-0.1/login/index HTTP/1.1" 200 –

6.3. Pruebas de tiempos con WIFI

Autenticación usuario (seg)	Autenticación conductor (seg)
6,1	6,36
10,56	7,25
7,21	7,03
5,74	6,81
6,2	6,88
6,28	6,18
6,14	8,16
6,3	6,36
7,1	6,8
5,9	5,43
6,32	7,1
7,12	6,43
8,1	7,54
7,29	6,54
6,32	7,2
7,11	7,21
5,87	6,43
5,65	5,79
7,45	8,1
6,43	5,34
8,31	7,3
6,21	6,43
7,61	6,23
6,6	6,78
6,34	5,65

Tabla 19: Tiempos con WIFI

Des. estándar	1,060752406	0,726244449
Mediana	6,34	6,78
Promedio	6,8104	6,6932
Mínimo	5,65	5,34
Máximo	10,56	8,16

Tabla 20: Tiempos con WIFI (estadísticas en segundos)

Se puede observar que el tiempo promedio de autenticación del usuario es mayor a la autenticación del conductor debido a que la huella con la que se compara al conductor ya está en el servidor guardada por lo tanto no tiene que ser enviada por la aplicación, se evidencia también que la desviación estándar es mayor para el

usuario, esto puede producirse debido a la información adicional que tiene que enviarse y la variación de la velocidad del WIFI, posiblemente por interferencias inherentes a la tecnología, obstáculos o interferencias.

6.4. Pruebas de tiempos con plan de datos

Tecnología 2G

Autenticación usuario (seg)	Autenticación conductor (seg)
36,4	44.54
49,1	33,11
40,03	34,8
46,05	36,7
42,41	32,6
41,4	37,67
45,6	36,51
50,21	39,6
39,8	33,99
37,67	38,65
43,23	37,96
45,26	34,28
49,1	33,81
40,32	40,1
40,38	39,32
42,14	36,27
48,6	39,21
45,67	34,67
51,1	37,49
41,23	35,11
38,56	37,21
39,76	35,61
38,2	34,19
42,33	33,91
44,39	35,72

Tabla 21: Tiempos con tecnología 2G

Des. estándar	4,17629449	2,20670023
Mediana	42,33	35,995
Promedio	43,1576	36,18708333

Mínimo	36,4	32,6
Máximo	51,1	40,1

Tabla 22: Tiempos con tecnología 2G (estadísticas en segundos)

Tecnología 3G

Autenticación usuario (seg)	Autenticación conductor (seg)
13,43	12,54
14,65	11,65
15,68	13,54
15,13	12,78
16,7	11,45
17,25	11,75
16,43	12,73
17,41	13,59
13,69	11,47
14,55	11,01
16,12	13,75
15,65	13,8
17,12	14,01
13,76	13,71
15,75	13,45
16,41	11,92
14,26	11,25
13,56	12,81
14,92	13,16
15,87	13,61
13,63	12,12
13,48	13,61
13,23	13,6
12,61	13,19
13,56	14,22

Tabla 23: Tiempos con tecnología 3G.

Des. estándar	1,436717903	0,977907289
Mediana	14,92	13,16
Promedio	14,994	12,8288
Mínimo	12,61	11,01
Máximo	17,41	14,22

Tabla 24: Tiempos con tecnología 3G (estadísticas en segundos).

Con plan de datos móviles se evidencia un mayor tiempo, teniendo en cuenta que es más rápido a mayor generación de tecnología móvil, evidenciando casi la mitad de tiempo entre 2G y 3G, se evidencia de igual manera lo ocurrido con el WIFI, mayor tiempo de respuesta en la autenticación para el usuario que para el conductor, al igual que las desviaciones estándar con un mayor valor para el pasajero y manejando picos extraños esporádicos.

7. Conclusiones

- Tomando como referencia las estadísticas expuestas en el planteamiento del problema y analizando la problemática social que se evidencia en los medios de comunicación y en la experiencia de la vida cotidiana, es posible tener una visión positiva con respecto al impacto que puede generar este proyecto en uno de los aspectos más relevantes de la situación social actual, la inseguridad, enfocada como se ha descrito anteriormente en el servicio público que ofrecen los taxis.

- Teniendo en cuenta que la huella dactilar se concibe como una señal única e intransferible de la identidad, el obtener este registro durante el servicio permite tener una unicidad, lo que proporciona total confiabilidad de la información facilitando así el accionar de las autoridades ante un evento delictivo durante el servicio de taxi y cohibiendo el actuar de los criminales, por tanto ayudando a la reducción del índice en los casos de inseguridad.

- Parte del éxito del proyecto radica en el tiempo que tome el proceso completo de autenticación de los usuarios, ya que puede determinar la aceptación del sistema y masificación de su uso, permitiendo recopilar una mayor cantidad de registros que ayuden a identificar patrones de movilidad y que sirvan de repositorios para consultas posteriores de las autoridades y seguimiento en las investigaciones de paseos millonarios y otros delitos en el servicio.

- El sistema de autenticación de huellas dactilares permite realizar las comparaciones de las huellas una a una, para esto el sistema necesita realizar un apareamiento previo por medio de la cédula y así optimizar el tiempo de búsqueda, ya que el realizar el proceso de comparación de toda la base de datos de huellas puede ser traumático en caso de un uso masivo del sistema. Si bien el procesamiento de la imagen dura milisegundos en el

servidor, realizar este proceso en una base de datos grande puede causar demora de segundos o incluso minutos que hacen inviable el proceso.

- Si bien, el software de reconocimiento de huellas dactilares reconoce giros en las huellas y es capaz de procesar esta información, el dispositivo lector de huellas integrado a la Tablet es muy pequeño, por lo cual se generan muchos errores al momento de la toma de la huella, como es el caso de una posición por encima o por debajo de la huella registrada o un giro de la huella, haciendo que los puntos característicos resaltados por el software queden por fuera de la imagen a comparar, los errores que se presenten podrían ocasionar insatisfacción en el uso del servicio.

- Hay factores inherentes al proyecto que no son controlables, entre ellos la tecnología 3G con la que cuente el dispositivo, que actualmente presenta una congestión debido a la saturación de los operadores celulares en el área de prueba, generando retrasos en el envío y recepción de la información transmitida, además en vehículos en movimiento pueden presentarse puntos sin conexión que pueden generar intermitencias en el envío de la información.

8. Trabajo futuro

- Con el fin de minimizar los tiempos en las bases de datos que contienen numerosos registros, se pretende realizar un filtrado por zonas para la búsqueda e implementar una base de datos jerárquica para acelerar la exploración de la huella.

- Los dispositivos para producción deben incluir tecnología 4G que permitan una mayor velocidad en el envío y recepción de la información, minimizando el tiempo de respuesta del aplicativo y por tanto la experiencia de usuario.

- A futuro se pretende implementar otros servicios aprovechando la infraestructura tecnológica desplegada y permitiendo cubrir necesidades adicionales de seguridad como es el pago en línea, que reduce el manejo de dinero en efectivo, solicitud de servicios por medio del móvil y una aplicación para que los usuarios puedan tener una bitácora de sus servicios con la información completa de fecha, duración, recorrido, posicionamiento, etc.

9. REFERENCIAS

1. EasyTaxi. EasyTaxi. [Online]; 2014. Available from: http://www.easytaxi.com/co/page/conditions.

2. Tappsi. Tappsi. [Online]; 2014. Available from: http://tappsi.co/sobre-tappsi/.

3. Taxibeat. Taxibeat. [Online]; 2014. Available from: http://taxibeat.com.mx/como-funciona/.

4. Wannataxi. Wannataxi. [Online]; 2014. Available from: http://www.wannataxi.com/user/.

5. DIAN. Encuesta de convivencia y seguridad ciudadana. Boletín de prensa. Bogota: DIAN; 2012.

6. Policía Nacional de Colombia. Los taxistas como factor significativo en la seguridad de Bogotá. Criminalidad. 2008 Noviembre; 50(2): p. 152.

7. Ficalia general de la Nación. Ficalia general de la Nación. [Online]; 2013. Available from: http://www.fiscalia.gov.co/colombia/tag/paseo-millonario/.

8. Bolle R, Connell J, Pank S. Guide to Biometrics. SpringerVerlag. 2003.

9. Dass S, Jain A. Fingerprint-based recognition. Technometrics. 2007; 49(3): p. 262-276.

10. Jain , Ross , Prabhakar S. An introduction to biometrics recognition. IEEE Transactions on circuits and systems for video technology. 2004; 14(1): p. 4-20.

11. Jain AK, Maltoni D. Handbook of Fingerprint Recognition. SpringerVerlag New York. 2003.

12. History and science of fingerprints identification, technology and legal issues. Riged and Furrows. [Online]; 2014. Available from: http://ridgesandfurrows.homestead.com/fingerprint.html.

13. Krlk M, Nejman L. Fingerprints on artifacts and historical items: examples and documents. Journal of Ancients Fingerprints. 2007 Agosto.

14. Onin. The history of fingerprints. [Online]; 2014. Available from: http://www.onin.com/fp/fphistory.html.

15. H. Faulds. 2. On the skin-furrows of the hand. Nature. 1880; 22: p. 605.

16. Galton F. Fingerprints. Macmillan. 1892.

17. Sodhi GS, Kaur J. The forgotten Indian pioneers of fingerprint science. Current Science. 2005 Enero; 88.

18. Locard E. Les Pores et L´identification Des Criminels. Biologica: Revue Scientifique de Medicine. 1912; 2: p. 357-365.

19. Asbaugh DR. Quantitative-Qualitative Friction Ridge Analysis: An Introduction To Basic and Advanced Ridgeology. CRC Press. 1999.

20. The Thin Blue Line. The Thin Blue Line. [Online]; 2014. Available from: http://www.policensw.com/info/fingerprints/finger06.html.

21. Pankanti S, Prabhakar S, Jain AK. On the individuality of Fingerprints. IEEE Trans. 2002; 24(8): p. 1010-1025.

22. ANSI/NIST. The ANSI/NIST Committe to Define an Extended Fingerprint Feature Set. [Online]; 2006. Available from: http://fingerprint.nist.gov/standart/cdeffs/index.html.

23. Kong A, Zhang D, Kamel M. Palmprint identification using feature-level fusion. Pattern Recognition. 2006; 39(3): p. 478-487.

24. Home Office Automatic Fingerprint Recognition System (HOAFRS). Science and Technology Group. London:; 1993. Report No.: 16-93-0026.

25. R.M. McCabe ",J.M. Data Format for the Interchange of Fingerprint, Facial, Scar Mark & Tattoo (SMT) Information. Available from R.M. McCabe at NIST, 100 Bureau Drive, Stop 8940, Gaithersburg, American National Standard ANSI/NIST-ITL 1-2000; 2000. Report No.: 2089.

26. Federal Bureau of Investigation. The Science of Fingerprints. Washington D.C: Federal Bureau of Investigation, U.S. Department of Justice. Report No.: Rev. 12-84.

27. Federal Bureau of Investigation. Electronic Fingerprint Transmission Specification. Washington D.C: Federal Bureau of Investigation. Report No.: 20535.

28. Wilson C, Garris M, Watson C, Hicklin A. Studies of Fingerprint Matching Using the NIST Verification Test Bed (VTB). [Online]; 2003 [cited 2003 Julio. Available from: http://www.itl.nist.gov/iad/894.03/pact/pact.html.]

29. C. Wilson AHBUHKMZPGRMCWSOMB. Fingerprint Vendor Technology Evaluation (FpVTE) 2003, Technical Report NISTIR 7123. [Online]; 2004 [cited 2014 Octubre. Available from: http://fpvte.nist.gov/.

30. C. Watson CWMIRSKM. Studies of One-to-One Matching with Vendor SDK Matchers. Technical Report NISTIR 7119. ; 2004.

31. Amazon. AWS Case Study: Parse. [Online]. [cited 2014 Diciembre. Available from: http://aws.amazon.com/solutions/case-studies/parse/.]

32. adslzone. ¿El fin de la fragmentación en Android? Más del 80% con Ice Cream Sandwich o superior. [Online]. [cited 2014 Diciembre. Available from: http://www.adslzone.net/2014/11/04/el-fin-de-la-fragmentacion-en-android-mas-del-80-con-ice-cream-sandwich-o-superior/.]

33. Android Comunity. Understand the Lifecycle Callbacks. [Online]. [cited 2014 Diciembre. Available from: http://developer.android.com/training/basics/activity-lifecycle/starting.html.]

34. Eclipse. About the Eclipse Foundation. [Online]. [cited 2015 Abril. Available from: https://www.eclipse.org/org/.]

35. groovy. A dynamic language for the Java platform. [Online]; 2014. Available from: http://groovy.codehaus.org/.

36. Grails. What is Grails? [Online]. [cited 2014 Noviembre. Available from: https://grails.org/.]

37. Apache Tomcat. Apache Tomcat. [Online]; 2014. Available from: http://tomcat.apache.org/.

38. NIST. NIST Biometric Image Software. [Online]; 2014. Available from: http://www.nist.gov/itl/iad/ig/nbis.cfm.

39. JSON. Introducing JSON. [Online]; 2014. Available from: http://www.json.org/.

40. Jain AK, Chen Y, Demirkus M. Pores and Ridges: High-Resolution Fingerprint Matching Using Level 3 Features. PAMI. 2007 Enero; 29(1): p. 15-27.

10. Anexos

- Manual instalación Servidor de procesamiento y comparación de huellas
- Manual mindtct
- Manual bozorth3
- Articulo publicable
- Documento de elicitación

ANEXOS

1. Manual instalación Servidor de procesamiento y comparación de huellas

El servidor permite realizar el proceso de análisis de las huellas dactilares que le llegan desde el servidor *Parse* utilizando el algoritmo *MINDTCT* para su análisis y el algoritmo *BOZOTH3* para la comparación de las huellas enviando como resultado una confirmación de autenticidad o una negación.

Instalación del servidor

Para el proceso de instalación del servidor se debe tener en cuenta que se realizó con la versión Ubuntu 14.04 LTS que es la última publicada, realizando previamente un proceso de actualización mejoramiento con los comandos "*apt-get update*" y el comando "*apt-get upgrade*" y se realiza la instalación de los paquetes necesarios para que el servicio funcione correctamente.

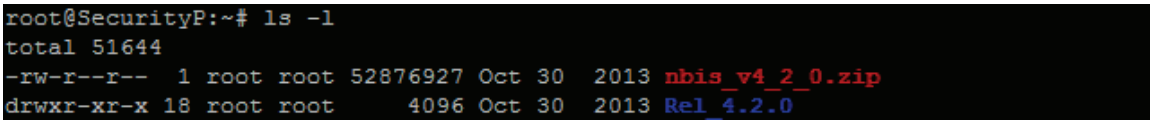
```
root@SecurityP:~# uname -a
Linux SecurityP 3.13.0-32-generic #57-Ubuntu SMP Tue Jul 15 03:51:08 UTC 2014 x8
6_64 x86_64 x86_64 GNU/Linux
root@SecurityP:~#
```

Para descargar los paquetes *MINDTCT* y el *BOZOTH3* se ingresa a la página del *NIST* y descarga la versión Release 4.2.0 del *NBIS Software* que contiene estos paquetes y la almacenamos en el servidor.

```
root@SecurityP:~# wget http://nigos.nist.gov:8080/nist/nbis/nbis_v4_2_0.zip
--2014-11-13 10:33:37--  http://nigos.nist.gov:8080/nist/nbis/nbis_v4_2_0.zip
Resolving nigos.nist.gov (nigos.nist.gov)... 129.6.24.104, 2610:20:6005:24::104
Connecting to nigos.nist.gov (nigos.nist.gov)|129.6.24.104|:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 52876927 (50M) [application/zip]
Saving to: 'nbis_v4_2_0.zip'

74% [=========================>          ] 39,224,599  3.13MB/s  eta 4s
```

De ser necesario debemos descargar el paquete *unzip* con el comando "*apt-get install unzip*" para descomprimir el archivo que acabamos de descargar de tal forma que tengamos la carpeta interna sin compresión como se evidencia en la figura.



Antes de realizar la instalación del *nbis* debemos estar seguros de tener instalado otros componentes para el funcionamiento correcto del *MINDCTC* y el *BOZORTH3*, por lo tanto realizamos la instalación de los paquetes *gcc*, make y build-essential *tomcat* y la administración del *tomcat* con el siguiente comando:

"*apt-get install gcc tomcat7 tomcat7-admin make build-essential*"

A continuación se hace la descripción de cada paquete utilizado.

Gcc: Es una colección de compiladores de GNU incluye front-ends para C, C ++, Objective-C, Fortran, Java, Ada, y Go, así como las bibliotecas de idiomas (libstdc ++, libgcj, ...). GCC fue escrito originalmente como el compilador para el sistema operativo GNU. El sistema GNU fue desarrollado para ser 100% software libre.

tomcat7: Este es el punto de entrada de nivel superior del paquete de documentación para el contenedor *Apache Tomcat Servlet / JSP*. *Apache Tomcat* versión 7.0 implementa el *Servlet 3.0 y JavaServer Pages 2.2* e incluye muchas características adicionales que lo convierten en una plataforma útil para el desarrollo y despliegue de aplicaciones y servicios web.

tomcat7-admin: Es el acceso de administración de tomcat7 vía http, Apache Tomcat implementa el servlet de Java y *JavaServer Pages* (JSP) y ofrece un entorno de servidor web HTTP de código Java.

Make: Es una herramienta que controla la generación de ejecutables y otros archivos que no sean fuente de un programa, a partir de archivos de código fuente del programa. *make* obtiene la información de cómo construir su programa desde otro archivo, que enumera cada uno de los archivos que no son de origen y la forma de calcular de otros archivos. Cuando se escribe un programa, se debe escribir un *makefile* para ello, por lo que es posible hacer uso de *make* para instalar el programa.

build-essential: Este paquete contiene una lista informativa de los paquetes considerados esenciales para la creación de paquetes *Debian*.

```
Selecting previously unselected package libalgorithm-merge-perl.
Preparing to unpack .../libalgorithm-merge-perl_0.08-2_all.deb ...
Unpacking libalgorithm-merge-perl (0.08-2) ...
Selecting previously unselected package libfile-fcntllock-perl.
Preparing to unpack .../libfile-fcntllock-perl_0.14-2build1_amd64.deb ...
Unpacking libfile-fcntllock-perl (0.14-2build1) ...
Processing triggers for man-db (2.6.7.1-1ubuntu1) ...
Setting up libstdc++-4.8-dev:amd64 (4.8.2-19ubuntu1) ...
Setting up g++-4.8 (4.8.2-19ubuntu1) ...
Setting up g++ (4:4.8.2-1ubuntu6) ...
update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mode
Setting up make (3.81-8.2ubuntu3) ...
Setting up libdpkg-perl (1.17.5ubuntu5.3) ...
Setting up dpkg-dev (1.17.5ubuntu5.3) ...
Setting up build-essential (11.6ubuntu6) ...
Setting up libfakeroot:amd64 (1.20-3ubuntu2) ...
Setting up fakeroot (1.20-3ubuntu2) ...
update-alternatives: using /usr/bin/fakeroot-sysv to provide /usr/bin/fakeroot (fa
keroot) in auto mode
Setting up libalgorithm-diff-perl (1.19.02-3) ...
Setting up libalgorithm-diff-xs-perl (0.04-2build4) ...
Setting up libalgorithm-merge-perl (0.08-2) ...
Setting up libfile-fcntllock-perl (0.14-2build1) ...
root@SecurityP:~/Rel_4.2.0#
```

Ya con los requisitos instalados debemos ejecutar el *setup* que se encuentra en el paquete con la ruta de instalación y las librerías estándar y sin el paquete X11, para la arquitectura del equipo, según el siguiente comando.

"./setup.sh /usr/local/nbis --without-X11 --STDLIBS –64"

```
root@SecurityP:~/Rel_4.2.0# ./setup.sh /usr/local/nbis --without-X11 --STDLIBS --64
root@SecurityP:~/Rel_4.2.0# make config
```

Y se ejecuta "*make* config" para realizar la configuración y verificación de los paquetes necesarios y posteriormente se realiza el "make" para que compile el paquete.

```
/bin/cat /root/Rel_4.2.0/doc/catalogs/catalog_apps.txt catalog_pcasys.txt > \
               /root/Rel_4.2.0/doc/catalogs/catalog_apps.txt.temp
/bin/mv -f /root/Rel_4.2.0/doc/catalogs/catalog_apps.txt.temp /root/Rel_4.2.0/doc/c
atalogs/catalog_apps.txt
make[4]: Leaving directory `/root/Rel_4.2.0/pcasys/src/bin/pcasys'
make[4]: Entering directory `/root/Rel_4.2.0/pcasys/src/bin/rwpics'
/bin/mv -f catalog.txt catalog_rwpics.txt
/bin/cat /root/Rel_4.2.0/doc/catalogs/catalog_apps.txt catalog_rwpics.txt > \
               /root/Rel_4.2.0/doc/catalogs/catalog_apps.txt.temp
/bin/mv -f /root/Rel_4.2.0/doc/catalogs/catalog_apps.txt.temp /root/Rel_4.2.0/doc/c
atalogs/catalog_apps.txt
make[4]: Leaving directory `/root/Rel_4.2.0/pcasys/src/bin/rwpics'
make[4]: Entering directory `/root/Rel_4.2.0/pcasys/src/bin/stackms'
/bin/mv -f catalog.txt catalog_stackms.txt
/bin/cat /root/Rel_4.2.0/doc/catalogs/catalog_apps.txt catalog_stackms.txt > \
               /root/Rel_4.2.0/doc/catalogs/catalog_apps.txt.temp
/bin/mv -f /root/Rel_4.2.0/doc/catalogs/catalog_apps.txt.temp /root/Rel_4.2.0/doc/c
atalogs/catalog_apps.txt
make[4]: Leaving directory `/root/Rel_4.2.0/pcasys/src/bin/stackms'
make[3]: Leaving directory `/root/Rel_4.2.0/pcasys/src/bin'
make[2]: Leaving directory `/root/Rel_4.2.0/pcasys/src'
make[1]: Leaving directory `/root/Rel_4.2.0/pcasys'
End: Generating "nbis" catalogs.
root@SecurityP:~/Rel_4.2.0#
```

Hasta este paso se realizó la instalación del paquete *Nbis* y debemos preparar el servidor web para que reciba las solicitudes de análisis y comparación de las huellas dactilares.

Para configurar el *tomcat* debemos ingresar a el archivo de configuración tomcat-users.xml y agregar un rol y un usuario para la administración mediante http.

```
root@SecurityP:~/Rel_4.2.0# nano /etc/tomcat7/tomcat-users.xml
root@SecurityP:~/Rel_4.2.0# service tomcat7 restart
```

```
  GNU nano 2.2.6        File: /etc/tomcat7/tomcat-users.xml              Modified

  NOTE:  The sample user and role entries below are wrapped in a comment
  and thus are ignored when reading this file. Do not forget to remove
  <!.. ..> that surrounds them.
-->
<!--
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <user username="tomcat" password="tomcat" roles="tomcat"/>
  <user username="both" password="tomcat" roles="tomcat,role1"/>
  <user username="role1" password="tomcat" roles="role1"/>
-->
<role rolename="manager-gui"/>
<user username="juan" password="juan" roles="manager-gui"/>
</tomcat-users>
```

Agregamos a la parte inferior del archivo de configuración las líneas siguientes donde se establece el rol de administración de la interface gráfica y el usuario que va a hacer uso de ese rol.

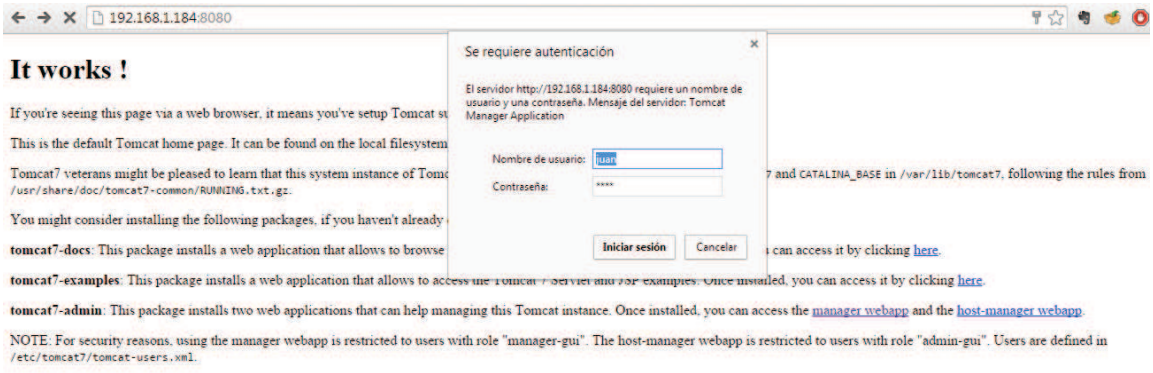
“<role rolename="manager-gui"/>”

“<user username="juan" password="juan" roles="manager-gui"/>”

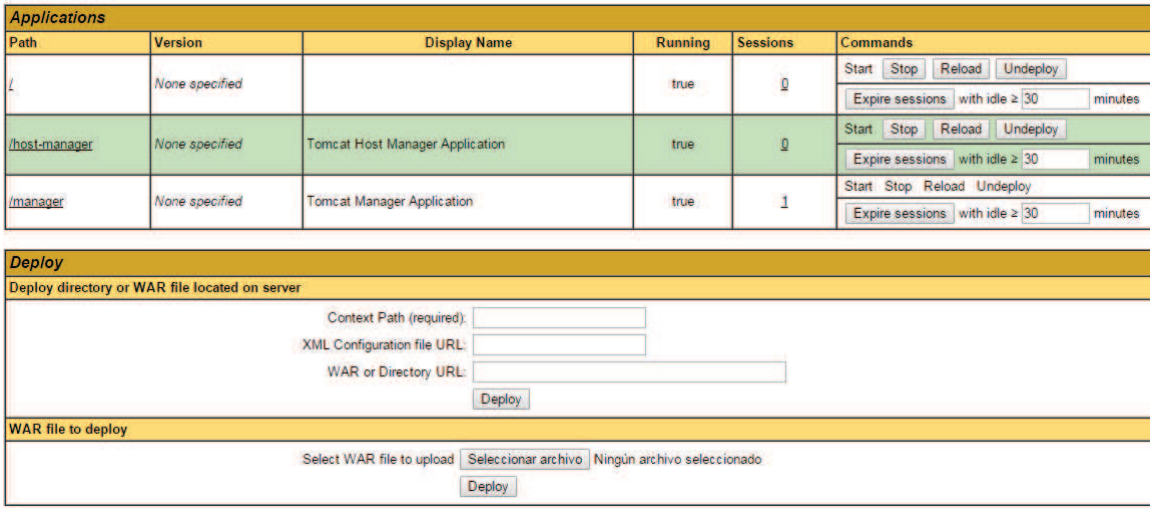
Se guarda el archivo y se reinicia el servicio de tomcat7 para que tome los cambios con el comando “*service tomcat7 restart”.*

```
root@SecurityP:~/Rel_4.2.0# nano /etc/tomcat7/tomcat-users.xml
root@SecurityP:~/Rel_4.2.0# service tomcat7 restart
 * Stopping Tomcat servlet engine tomcat7                              [ OK ]
 * Starting Tomcat servlet engine tomcat7                              [ OK ]
```

Se abre un navegador y se ingresa a la ip del servidor o el nombre en el Puerto 8080 e ingresamos con el usuario y contraseña que acabamos de registrar en el archivo de configuración del servicio tomcat7.



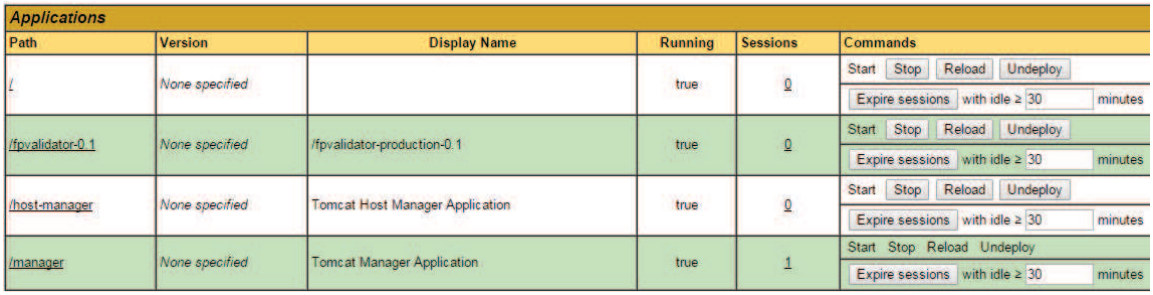
Nos va a cargar la página del Tomcat Web Application Manager donde subiremos el archivo WAR para desplegar el proyecto. Este es un archivo JAR utilizado para distribuir una colección de JavaServer Pages, servlets, clases Java, archivos XML, librerías de tags y páginas web estáticas (HTML y archivos relacionados) que juntos constituyen una aplicación web.



Ante de cargar el archivo WAR debemos dar permisos de propietario al tomcat7 en la carpeta tomcat7 dentro de la ruta de instalación como se muestra en la siguiente Figura.

```
root@SecurityP:/usr/local/nbis/bin# chown tomcat7:tomcat7 tomcat7/
```

Cuando el archivo WAR este cargado iniciamos el aplicativo fpvalidator-0.1 desde el Tomcat Web Application Manager como se muestra en la figura.



Applications					
Path	Version	Display Name	Running	Sessions	Commands
/	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/fpvalidator-0.1	None specified	/fpvalidator-production-0.1	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Con el aplicativo corriendo solo nos queda agregar los links simbólicos de MINDTCT y el BOZOTH3 en la ruta *"/var/lib/tomcat7/main/lib"* la cual debemos crear de la siguiente manera:

"*mkdir –p /var/lib/tomcat7/main/lib"*

Ingresamos a esa ruta y creamos los link simbólicos a los ejecutables MINDTCT y el BOZOTH3 en dicha ruta donde el aplicativo los espera.

"*ln -s /usr/local/nbis/bin/mindtct mindtct.exe"*

"ln -s /usr/local/nbis/bin/bozorth3 bozorth3.exe"

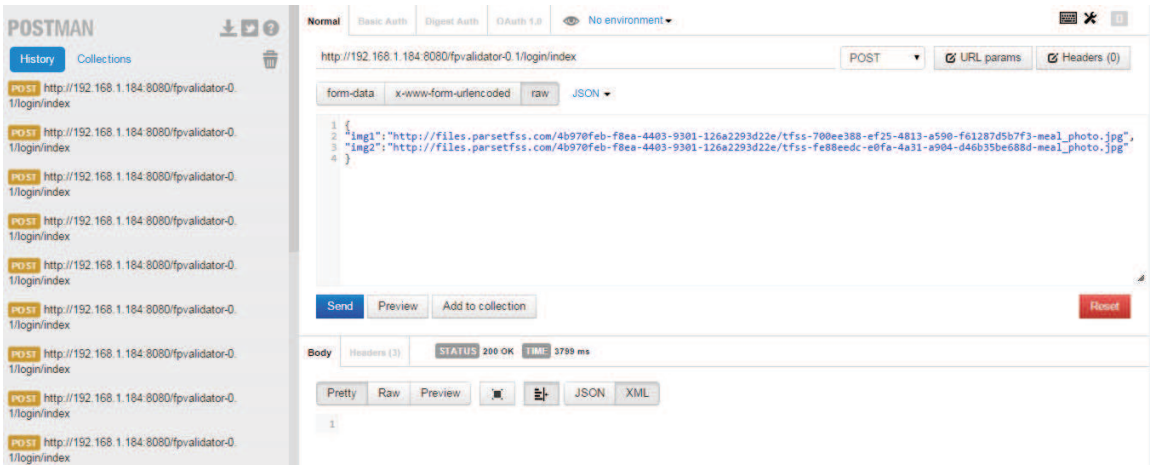
Y comprobamos que se esté ejecutando tanto el mindtct como el bozorth3

```
root@SecurityP:/var/lib/tomcat7/main/lib# ./mindtct.exe
```

Ya instalado y funcionando debemos realizar la prueba por medio de la funcionalidad postman del google Chrome.

Instalamos la extensión postman ingresamos la url del servicio donde se instaló anteriormente http://192.168.1.184:8080/fpvalidator-0.1/login/index seleccionamos tipo POST y seleccionamos un raw tipo JSON como se muestra en la figura y le enviamos las dos imágenes para que el servidor analice y compare, devolviendo una aceptación o una denegación.

Si envía STATUS 200 OK indica que confirmo la huella dactilar, si por el contrario envía STATUS 401 unauthorized indica que las huellas no tienen coincidencia.



Si queremos podemos visualizar los registros de cada huella dactilar y determinar tiempos y análisis de cada huella dactilar en la ruta *"/var/logs/tomcat7/catalina.out*" y en la ruta *"/tmp/minext/MinTemplates/"* encontraremos la huellas ya procesadas por el *mindtct* para ser enviadas al *bozorth*, que compara los archivos y envía el resultado afirmativo o negativo.

De esta forma se publicara el servicio para que la huella enviada por el dispositivo lector de huellas dactilares autentique la huella dactilar y envié la información a los usuarios del servicio.

2. Manual mindtct

NAME

mindtct − detects minutiae from a fingerprint image that is either an ANSI/NIST 2007 formatted file or a WSQ compressed file.

SYNOPSIS

mindtct *[-b] [-m1] <finger_img_in> <oroot>*

DESCRIPTION

Mindtct takes either a WSQ compressed image file or parses a standard compliant ANSI/NIST-ITL 1-2007 file searching for the first occurrence of a grayscale fingerprint image record. The fingerprint image is processed and minutiae are automatically detected.

If the input file was in ANSI/NIST 2007 format the minutiae results are formatted and stored using the NIST fields 5-12 in a Type-9 record. Upon successful completion, the input ANSI/NIST record sequence is augmented with two new records, the Type-9 minutiae record and a tagged field image record containing the results of image binarization. This augmented record sequence is then written to the output file <oroot>.mdt. The minutiae values are also written to a text file <oroot>.xyt in "x y theta quality" format. This <oroot>.xyt file is the format used by the bozorth3 matcher.

If the input image is a WSQ compressed file the minutiae are only written to the text file <oroot>.xyt. In addition a file called <oroot>.brw is created which is a raw pixel file of the binarized image created by mindtct when detecting the minutiae points.

The default is minutiae points computed based on the pixel origin being at the bottom left of the image and directions are pointing out and away from the ridge ending or bifurcation valley.

Mindtct also generates the following text files in the current working directory: <*oroot*>*.dm*, <*oroot*>*.hcm*, <*oroot*>*.lcm*, <*oroot*>*.lfm*, <*oroot*>*.qm*, and <*oroot*>*.min*. These files are described below.

OPTIONS

[-b] perform image enhancement on low contrast images. Only affects low contrast images, others are unchanged.

[-m1] write the minutiae points (in files <oroot>.{mdt,xyt,min}) according to ANSI INCITS 378-2004. This format has the pixel origin at the top left of the image and directions are pointing up the ridge ending or bifurcation valley. The default for **mindtct** has the pixel origin at the bottom left of the image and directions are pointing out and away from the ridge ending or bifurcation valley. NOTE: If this flag is used when extracting the mintuiae points it must also be used with the **bozorth3** matcher.

<*finger_img_in*>
the fingerprint file to be processed

<*oroot*>
the root name for the output files (<oroot>.???)

TEXT OUTPUT FILES

<*oroot*>*.dm*
The *Direction Map* represents the direction of ridge flow within the fingerprint image. The map contains a grid of integer directions, where each cell in the grid represents an 8x8 pixel neighborhood in the image. Ridge flow angles are quantized into 16 integer bi-directional units equally spaced on a semicircle. Starting with vertical direction 0, direction units increase clockwise and represent incremental jumps of 11.25 degrees, stopping at direction 15 which is 11.25 degrees shy of vertical. Using this scheme, direction 8 is horizontal. A value of -1 in this map represents a neighborhood where no valid ridge flow was determined.

<*oroot*>*.hcm*
The *High-Curvature Map* represents areas in the image having high-curvature ridge flow. This is especially true of core and delta regions in the fingerprint image, but high-curvature is not limited to just these cases. This is a bi-level map with same dimension as the Direction Map. Cell values of 1 represent 8x8 pixel neighborhoods in the fingerprint image that are located within a high-curvature region, otherwise cell values are set to 0.

<*oroot*>*.lcm*
The *Low-Contrast Map* represents areas in the image having low-contrast. The regions of low contrast most commonly represent the background in the fingerprint image. This is a bi-level map with same dimension as the Direction Map. Cell values

of 1 represent 8x8 pixel neighborhoods in the fingerprint image that are located within a low-contrast region, otherwise cell values are set to 0.

<oroot>.lfm

The *Low-Flow Map* represents areas in the image having non-determinable ridge flow. Ridge flow is determined using a set of discrete cosine wav e forms computed for a predetermined range of frequencies. These wave forms are applied at 16 incremental orientations. At times none of the wave forms at none of the orientations resonate sufficiently high within the region in the image to satisfactorily determine a dominant directional frequency. This is a bi-level map with same dimension as the Direction Map. Cell values of 1 represent 8x8 pixel neighborhoods in the fingerprint image that are located within a region where a dominant directional frequency could *not* be determined, otherwise cell values are set to 0. The Direction Map also records cells with nondeterminable ridge flow. The difference is that the Low-Flow Map records *all* cells with nondeterminable ridge flow, while the Direction Map records only those that remain non-determinable after extensive *interpolation* and *smoothing* of neighboring ridge flow directions.

<oroot>.qm

The *Quality Map* represents regions in the image having varying levels of quality. The maps above are combined heuristically to form 5 discrete levels of quality. This map has the same dimension as the Direction Map, with each value in the map representing an 8x8 pixel neighborhood in the fingerprint image. A cell value of 4 represents highest quality, while a cell value of 0 represent lowest possible quality.

<oroot>.xyt

This text file reports the minutiae detection results. This reports only the x,y coordinates, theta, and quality of the minutie points for the image. Each line in this file contains the space delimited information for one minutiae point. The <oroot>.xyt is the minutiae format used by the **bozorth3** matching algorithm.

<oroot>.min

This text file reports the minutiae detection results. The majority of the results listed in this text file are also encoded and stored in a Type-9 record in the output ANSI/NIST file. The first nonempty line in the text file lists the number of minutiae that were detected in the fingerprint image. Following this, the attributes associated with each detected minutia are recorded, one line of text per minutia. Each minutia line has the same format. Fields are separated by a ':', subfields are separated by a ';', and items within subfields are separated by a ','. A minutia line may be represented as:

MN : *MX*, *MY* : *DIR* : *REL* : *TYP* : *FTYP* : *FN* : *NX1*, *NY1*; *RC1* : ...

where:

MN is the integer identifier of the detected minutia.

MX is the x-pixel coordinate of the detected minutia. *MY* is the y-pixel coordinate of the detected minutia.

DIR is the direction of the detected minutia. Minutia direction is represented similar to ridge flow direction, only minutia direction is uni-directional starting at vertical pointing up with unit 0 and increasing clockwise in increments of 11.25 degrees completing a full circle. Using this scheme, the angle of a detected minutia is quantized into the range 0 to 31 with 8 representing

horizontal to the right, 16 representing vertical pointing down, and 24 representing horizontal to the left.

REL is the reliability measure assigned to the detected minutia. This measure is computed by looking up the quality level associated with the position of the minutia from the Quality Map. The quality level is then heuristically combined with simple neighborhood pixel statistics surrounding the minutia point. The results is a floating point value in the range 0.0 to 1.0, with 0.0 representing lowest minutia quality and 1.0 representing highest minutia quality.

TYP is the type of the detected minutia.
bifurcation =
"BIF" ridge
ending = "RIG"

FTYP is the type of feature detected.
appearing = "APP"
disappearing = "DIS"
(This attribute is primarily useful for purposes internal to the minutia detection algorithm.)

FN is the integer identifier of the type of feature detected. (This attribute is primarily useful for purposes internal to the minutia detection algorithm.) *NX1* is the x-pixel coordinate of the first neighboring minutia. *NY1* is the y-pixel coordinate of the first neighboring minutia.

*RC1*is the ridge count calculated between the detected minutia and its first neighbor.

... for each additional neighbor ridge count computed, the pixel coordinate of the neighbor and the ridge count to that neighbor are reported.

EXAMPLES

From *test/mindtct/execs/mindtct/mindtct.src*:

% mindtct ../../data/g001t2u.eft g001t2u

SEE ALSO

an2k7totxt(1F), **an2k7tool**(1F), **dpyan2k7**(1F), **bozorth3**(1E)

AUTHOR

NIST/ITL/DIV894/Image Group

3. Manual bozorth3

NAME

bozorth3 − Computes match scores between fingerprints

SYNOPSIS

bozorth3 [*options*] *probe-file.xyt gallery-file.xyt*
...
bozorth3 [*options*] **−M** *mates.lis*
bozorth3 [*options*] **−P** *probe.lis*
gallery.xyt* **bozorth3** [*options*] **−G**
gallery.lis probe.xyt*

bozorth3 [*options*] **−p** *probe-file.xyt*
gallery.xyt* **bozorth3** [*options*] **−p** *probe-*
file.xyt **−G** *gallery.lis* **bozorth3** [*options*] **−g**
gallery-file.xyt probe.xyt* **bozorth3** [*options*]
−g *gallery-file.xyt* **−P** *probe.lis*

DESCRIPTION

The program *bozorth3* computes match scores from fingerprint minutiae files. The files are expected to be in xyt-format, a simple text file format that is produced by the minutiae detector program *mindtct*, which is also part of the NFIS distribution.

By default, each pair of arguments on the command line is considered to be a probe file and a gallery file, in that order, that are to be matched to yield a score of similarilty. The higher the score, the more closely the minutiae in them match. The match score for known mates is often close to the number of minutiae in the probe or gallery file, but it can be lower or higher than that number, sometimes much higher.

There are two main mechanisms that allow running the *bozorth3* matcher other than by simply specifying pairs of xyt-files on the command line. The mechanisms are useful or necessary under several circumstances. For example, with large data sets, the number of pairs of files to be matched could easily exceed the maximum size the user's shell permits on a command line, or that's permitted by *exec*()* system calls. And there are cases where it's just more logical to have input filenames stored in a file. So one mechanism uses *list files* — they contain xyt-filenames, one per line, with newline characters as line-endings. The other mechanism fixes the probe (or gallery) file for an entire run, so that the filename doesn't hav e to be specified over and over again.

One form of the list file mechanism allows the pairs of files to be read from a single file. The *-M mates.lis* option requires a single list file of filenames to be matched against each other. The probe filenames are on the odd lines, and the gallery filenames are on the even lines.

Similarly, the *-P probes.lis* option specifies that the probe filenames are in the file, and the gallery filenames come from the command line. The *-G gallery.lis* option specifies just the opposite. Both options may be present, in which case all filenames will be read from the two files, and there will be no xyt-files on the command line.

The other subset of mechanisms fix a single file to be matched against a gallery (or probe) set of any size. For example, *-p probe-file* fixes the probe file for the entire run; it will be

matched against a gallery consisting of all other files on the command line (or, if *-G gallery.lis* is specified, against a gallery read from a file).

The *-g gallery-file* option specifies just the opposite. While it may seem illogical to reverse the notion of probe and gallery files by allowing a single gallery file to be compared against a probe set, it's allowed both for consistency and to make it easier to test how close scores are when the files are matched in reverse order.
Fixing both the probe and gallery file is legal, but it's equivalent to having just a single pair of filenames on the command line without the *-p* and *-g*.

The score for a probe file *a* matched to a gallery file *b* is often identical to the score for *b* matched to *a*. One one data set, the scores were the same more than 75% of the time, and only a very small number were different by more than 3.

Minutiae file format

Each line in a minutiae file contains three integers, representing the x- and y-coordinates and direction of the minutiae, and an optional fourth column of integers representing the quality of the minutiae at those coordinates. If the quality column isn't present in a file, all minutiae are assumed to be of the same quality.

A finger typically has 40-80 minutiae. Any automated minutiae extractor will, of course, flag some things as minutiae that aren't. To work with highly sensitive minutiae detectors such as *mindtct*, the *bozorth3* matcher allows each xyt-file to contain as many as 1000 minutiae lines. However, by default, only the 150 highest-quality minutiae are used to compute the match score. That number may be changed to any number from 0 to 200. If multiple minutiae have the same quality value at the cut-off point, the tie-breaking method is simple truncation of the list, sorted by quality but with an undefined sort order among its equalquality elements.

The optimal number of minutiae that should be used depends on the fingerprint images and the minutiae detector that processes them. Using more than is necessary typically reduces the accuracy of the matcher and increases its run time.

To compute a match score between two fingerprints, both sets must have at least a minimum number of minutiae. That number is 10 by default, and can be changed to any non-zero integer. Otherwise the computation returns a match score of 0.

OPTIONS

The command line options can be logically grouped into four classes:

General options

-h Print a help screen detailing the command line options.

-version
Print ANSI/NIST stardand and NBIS software version.

-v Enable verbose mode.

-A verbose=<section>
Enable verbose mode in a section of the code; the recognized sections are: main, load, bozorth, threshold.

Input options

-m1 all xyt files use representation according to ANSI INCITS 378-2004. This flag must be used if it was used by the **mindtct** algorithm when extracting the minutiae points.

-n max-minutiae
Set maximum number of minutiae to use from any file [150]; the legal range is [0,200].

-A minminutiae=#
Set minimum number of minutiae required for the match score to be more than 0 [10].

-A maxfiles=#
Set maximum number of files in any gallery, probe, or mates list file [10000].

-A plines=#-#
Process a subset of files in the probe file.

-A glines=#-#
Process a subset of files in the gallery file.

-A dryrun
Test mode only. Do not compute and print any match scores, just print the filenames between which match scores would be computed.

Thresholding options

-T threshold
Set match score threshold. By default, all match scores are printed. However, when a threshold specified, only match scores meeting or exceeding that value are printed.

-q Quit processing the probe file when a gallery file is found for which the match score meets or exceeds the specified threshold.

Output options

-A nooutput
Compute match scores, but don't print them.

-A outfmt=[spg]*
Output lines will contain (s)core, (p)robe and/or (g)allery filename. By default, only scores are output.

-O score-dir
Set the directory to write score files in.

-o score-file
Set the filename to store scores in.

-e stderr-file
Set the filename to store all other output in.

-b Use the default Standard I/O buffering to print the match scores. This is equivalent to line-buffering when the output is being printed to a terminal, and to block-buffering when the output is being printed to a file.

-l Use line-buffering to print the match scores. By default, output lines are stored and printed just prior to the *bozorth3* exiting.

SEE ALSO

mindtct (1C)

AUTHOR

Allan S. Bozorth of the FBI; modified by Michael Garris and Stan Janet, both of NIST/ITL/DIV894/Image Group.