Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Estado de México
Escuela de Ingeniería y Ciencias

TECNOLÓGICO DE MONTERREY®

Algoritmo de indexación basado en código cilíndrico de minucias para una rápida identificación de huellas latentes dactilares.

Una disertación de
ISMAY PÉREZ SÁNCHEZ

Presentado a la
Escuela de Ingeniería y Ciencias
en cumplimiento parcial de los requisitos para el grado de

Maestro
en
Ciencias de la Computación

Atizapán de Zaragoza, Estado de México
4 de Julio, 2019

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Estado de México

Los miembros del comité, por la presente, certifican que han leído la disertación presentada por Ismay Pérez Sánchez y que es completamente adecuada en alcance y calidad como un requisito parcial para el grado de Maestro en Ciencias de la Computación.

Dr. Miguel Angel Medina Pérez
Tecnológico de Monterrey, Campus Estado de México
Asesor Principal

Dr. Raúl Monroy Borja
Tecnológico de Monterrey, Campus Estado de México
Co-asesor

Dr. Octavio Loyola González
Tecnológico de Monterrey, Campus Puebla
Miembro del comité

Dr. Ponciano Jorge Escamilla Ambrosio
Centro de Invesstigación en Computación, Instituto Politéctino Nacional
Miembro del comité

Dr. Raúl Monroy Borja
Director del Programa de Posgrado
Escuela de Ingeniería y Ciencias
Tecnológico de Monterrey, Campus Estado de México

Atizapán de Zaragoza, Estado de México, 4 de Julio, 2019

Declaración de autoría

Yo, Ismay Pérez Sánchez, declaro que esta tesis titulada, 'Algoritmo de indexación basado en código cilíndrico de minucias para una rápida identificación de huellas latentes dactilares.' y el trabajo que se presenta en ella es de mi autoría. Adicionalmente, confirmo que:

- Realicé este trabajo en su totalidad durante mi candidatura al grado de maestro en esta universidad.
- He dado crédito a cualquier parte de esta tesis que haya sido previamente sometida para obtener un grado académico o cualquier otro tipo de titulación en esta o cualquier otra universidad.
- He dado crédito a cualquier trabajo previamente publicado que se haya consultado en esta tesis.
- He dado crédito a todas las fuentes de ayuda utilizadas.
- He dado crédito a las contribuciones de mis coautores, cuando los resultados corresponden a un trabajo colaborativo.
- Esta tesis es enteramente mía, con excepción de las citas indicadas.

Firma:

Fecha:

Resumen

Algoritmo de indexación basado en código cilíndrico de minucias para una rápida identificación de huellas latentes dactilares.

por Ismay Pérez Sánchez

La identificación de huellas latentes dactilares es una de las principales actividades forenses para esclarecer hechos delictivos. El costo computacional de esta actividad dificulta la rápida toma de decisiones en la identificación de un individuo cuando las bases de datos superan el millón de huellas. Con el propósito de reducir el tiempo de búsqueda para generar el orden de las huellas a comparar, se desarrollan algoritmos de indexación de huellas que incrementen lo menos posible la tasa de error en comparación a los algoritmos de identificación.

Existen varios algoritmos de indexación de huellas reportados en la literatura que logran más de un 95 % de eficacia usando impresiones como huella consulta. Sin embargo, estos algoritmos asumen alta calidad de imagen en la huella, información completa de la misma y con esto, la existencia de la mayoría de los rasgos que componen a una huella. Nada de lo anterior se garantiza para las huellas latentes, por lo que el enfoque más utilizado en la indexación de este tipo de huellas es la combinación de descriptores para suplir la carencia de información y la extracción defectuosa de los descriptores debido a la mala calidad de este tipo huellas. La mayoría de estos algoritmos incluyen los descriptores de minucias porque son los más tolerantes a los problemas mencionados debido a su presencia en cualquier parte de la huella; además, son tolerantes a la rotación, traslación y distorsión de las huellas.

En la presente investigación proponemos un algoritmo de indexación de huellas latentes basado en códigos cilíndricos de minucias. Este tipo de descriptor de minucia presenta una estructura regular, lo que le brinda ventajas en cuanto a eficiencia. Además, en estudios recientes, este descriptor ha mostrado una tasa de error en la identificación, a nivel local, inferior a los demás descriptores reportados.

Nuestra propuesta de indexación requiere de un preprocesamiento, en el cual se agrupan los códigos cilíndricos de minucias correspondientes a las impresiones de un conjunto de bases de datos. Luego se realizan tres etapas en nuestro algoritmo de indexación. Primero, se indexan las bases de datos de interés usando criterios de disimilitud con los centros de grupos generados. Segundo, dada una huella latente, se extraen los códigos cilíndricos de minucias asociados a los índices de los grupos con menor distancia euclidiana respecto a cada uno de los descriptores de la huella latente, ajustando dicha cantidad para evitar problemas de eficacia y eficiencia. En la etapa final se integran los votos representados por las huellas obtenidas, en la segunda etapa, para seleccionar las huellas candidatas a comparar con la huella latente.

Los resultados obtenidos mejoran al algoritmo base (MCC SDK) cuando es seleccionado el 0.01 % de una base de datos con más de un millón de impresiones, lo cual representa una reducción de cuatro órdenes de magnitud. Hasta donde sabemos, no existen investigaciones similares en bases de datos de este tamaño.

A mis padres y hermana, por tanto amor y apoyo que me han brindado en este largo camino hacia mis metas.

Agradecimientos

Eternos agradecimientos a mis asesores Dr. Miguel Angel Medina Pérez y Dr. Raúl Monroy Borja, por la incansable dedicación brindada en mi formación para lograr los resultados de esta investigación.

Un agradecimiento especial a Miguel Angel, además del asesor, al amigo, por el constante apoyo que me ha brindado desde antes de mi llegada a este programa de maestría. Y en conjunto con su esposa María Esther, por ofrecerme su techo en los momentos más difíciles vividos al poco tiempo de comenzar esta experiencia.

Agradezco a los miembros del comité de revisión, Dr. Octavio Loyola González y Dr. Ponciano Jorge Escamilla Ambrosio, por sus acertados comentarios con el objetivo de mejorar el contenido de este trabajo. Especialmente a Octavio, que a lo largo de estos dos años fue como un asesor más.

Al Dr. Francisco Herrera y el Dr. Salvador García por su colaboración en la concepción de las ideas que dieron paso a esta investigación y el desarrollo de la plataforma de experimentación.

Agradecimientos a mi inmensa familia, que tanto desde Cuba, Estados Unidos, como de España, siempre me hicieron llegar palabras de ánimo hasta el final. Especialmente a mis padres, Cirilo Pérez Donet y Maricely Sánchez Febles y a mi hermana, Idania Rojas Sánchez.

A mis compañeros de posgrado, Miryam Elizabeth Villa Pérez, Kevin Islas Abud, Javier de Velasco Oriol, Raúl Salgado Ulloa, German Alamilla Peralta, Nicolás Villegas Félix y Roberto Carlos Vazquez Nava por compartir momentos y enseñarme una cultura desconocida para mí. En especial para Roberto Carlos, amigo y compañero de ideas y proyectos, por su ayuda incondicional.

A los amigos de mi tierra natal que, con su paso por mi vida, aportaron de alguna forma en las decisiones que me han llevado hasta aquí. En especial para mis amigos de la universidad, Andres Eduardo Gutierrez Rodríguez, quien propició el primer contacto para iniciar este camino, y a Danilo Valdes Ramirez por tantas conversaciones de apoyo e invaluable ayuda en la búsqueda de soluciones a problemas de toda índole.

Mis agradecimientos al Tecnológico de Monterrey y al Consejo Nacional de Ciencia y Tecnología de México (CONACyT) por poner a mi disposición recursos en favor de mi formación científica.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Los sistemas de reconocimiento biométricos, en específico los que usan huellas latentes dactilares, están entre los mecanismos principales para la identificación de los posibles autores de hechos delictivos [46]. Las huellas latentes dactilares son aquellas que las personas dejan en la superficie de los objetos que tocan con sus dedos, debido a la naturaleza grasa de la piel [45]. La creación de las huellas latentes dactilares es involuntaria, por tanto, no se garantiza que estas huellas tengan una calidad tan alta como la de las huellas adquiridas por sensores dactilares en condiciones controladas; estas últimas llamadas impresiones. Sin embargo, a pesar de la baja calidad de las huellas latentes, a partir de ellas los peritos dactiloscópicos identifican a las personas.

En el contexto de reconocimiento de huellas dactilares existen dos tipos de aplicaciones, la verificación y la identificación. El objetivo de las aplicaciones de verificación de huellas dactilares consiste en determinar si dadas dos huellas dactilares tomadas de manera voluntaria en condiciones controladas, proceden del mismo dedo. Por su parte, el objetivo de las aplicaciones de identificación de huellas latentes dactilares consiste básicamente en, dada una huella latente, buscar todas las huellas en una base de datos que procedan del mismo dedo.

Los algoritmos para verificación de huellas son usados comúnmente en aplicaciones de control de acceso y los resultados en las competencias internacionales muestran una elevada tasa de identificación. Solo por citar un ejemplo, en la competencia FVC-onGoing[1] los mejores algoritmos alcanzan valores entre 95.0 % y 98.0 % aproximadamente para los indicadores de verificación de huellas. Las aplicaciones de verificación de huellas trabajan con impresiones; esto explica los altos porcentajes de identificación alcanzados.

[1] https://biolab.csr.unibo.it/FVCOnGoing/

Aunque la literatura sobre verificación de impresiones es extensa, son menos los trabajos sobre identificación de huellas latentes debido, en gran parte, a la complejidad inherente a éstas. Los algoritmos que trabajan con huellas latentes han reportado alta tasa de error; así lo muestra un trabajo reciente [11] donde el mejor resultado reporta un error del 25.6 % al comparar 258 huellas latentes contra 100,000 impresiones.

La dificultad en la identificación de huellas latentes se atribuye, principalmente, a tres factores [12]: baja calidad en las huellas latentes en cuanto a la claridad de las crestas; área pequeña de las huellas latentes comparado con las impresiones; y grandes distorsiones no lineales debido a las características de la superficie de donde son tomadas las huellas latentes.

En la actualidad existen soluciones informáticas para el reconocimiento automatizado de huellas y la complejidad temporal en la que se incurre es lineal. A pesar de ello, la tarea de reconocer una huella latente se vuelve prohibitiva para situaciones de rápida toma de decisiones [33] a medida que la base de datos de huellas escala en tamaño. En adelante, denotaremos "huella plantilla" a cada huella almacenada en la base de datos y "huella consulta" a la huella que se desea identificar.

Los grupos de algoritmos de *clasificación* e *identificación rápida* de huellas; y este último dividido en, sub-clasificación y clasificación continua [44], se han creado para reducir la cantidad de huellas candidatas a usar en la etapa de identificación de la huella consulta. Sin embargo, la mayoría de estas estrategias tienen grandes limitaciones; o reducen, cuanto más, un orden de magnitud (clasificación), ó simplemente son muy difíciles de implementar (subclasificación) [44]. Entre los mejores métodos derivados de clasificación continua se encuentran los métodos basados en funciones *hash* aplicadas a descriptores que consideran la información de la vecindad de cada minucia de la huella [33]. Una minucia es un rasgo de las huellas que representa el punto donde dos crestas se unen o donde una cresta termina de manera abrupta. Las mismas están representadas por una tupla $\langle x, y, \theta \rangle$ que contiene la posición y el ángulo respecto al eje coordenado.

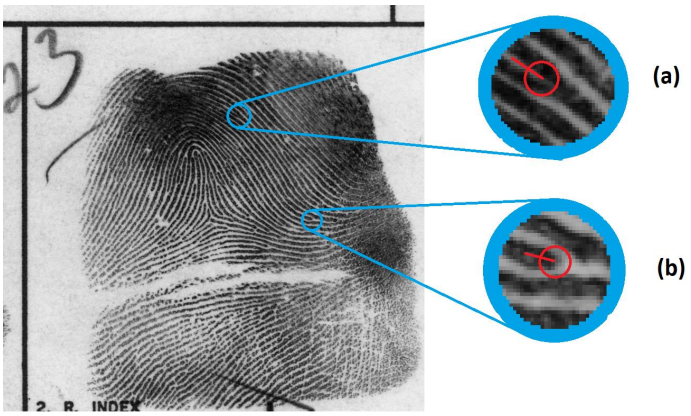


FIGURA 1.1: Ejemplo de minucias más frecuentes en las huellas: (a) bifurcación, (b) terminación. La imagen de la huella pertenece a la base de datos NIST SD27 (G001T2U.EFT).

Estos algoritmos de indexación se evalúan mediante la relación entre el error cometido al reconocer huellas consultas (ya sean impresiones o huellas latentes) y la cantidad de huellas plantillas que finalmente se usan de la base de datos [44]. Por lo que es de interés obtener el menor error con el menor porcentaje posible de la base de datos, logrando estos resultados en un tiempo cada vez menor.

Capelli et al. [16] introdujeron un algoritmo basado en códigos cilíndricos de minucias [14] el cual, para el problema de verificación de impresiones, obtiene un error del 5 % o menor para un 5 % de uso de bases de datos estándar (NIST SD4 [66], SD14 [67]). Por lo que en esta investigación se propone mejorar la tasa de error obtenida por el mencionado algoritmo cuando es usado en identificación de huellas latentes, sacrificando la eficiencia en una magnitud aceptable.

En el presente trabajo establecemos que un algoritmo **X** es más eficiente que un algoritmo **Y**, cuando dado el problema de indexación de huellas, **X** logra obtener la lista de huellas candidatas en un tiempo menor que **Y**, usando las mismas bases de datos como entrada. Esta métrica se calcula como $\frac{t(X)}{t(Y)}$, donde $t(X)$ y $t(Y)$ son los tiempos consumidos por ambos algoritmos. Valores menores a 1 indican que el algoritmo **X** es más eficiente y mayores a 1, lo opuesto.

Dada la definición anterior, se considera una pérdida de eficiencia aceptable a una diferencia no mayor de un orden de magnitud a favor del algoritmo a mejorar .

El estudio se llevará a cabo usando la misma forma de representación usada por Capelli et al. [16] con diferente estrategia de indexado y recuperación de impresiones.

A continuación en este capítulo se define el problema (sección 1.1), hipótesis (sección 1.2), objetivos (sección 1.3) y por último la estructura del presente documento.

1.1. Descripción del problema

Mejorar la eficiencia en los algoritmos de reconocimiento de huellas ha sido un campo activo de investigación [35, 36, 55]. Sin embargo, los algoritmos para la identificación rápida de huellas han sido propuestos, en su mayoría, para las aplicaciones de verificación de huellas. En este contexto, se han desarrollado nuevos algoritmos tanto en el enfoque de clasificación, como en el de indexación.

1.1.1. Clasificación de huellas

Entre los algoritmos usados para la identificación rápida de huellas se encuentran los clasificadores. Estos trabajan directamente con el flujo de líneas que forman las crestas de la huella [57, 64, 70], la orientación de la imagen [26, 38, 52, 59, 62], los puntos singulares [26, 38, 62] y filtros de Gabor [26, 37, 48]. Estos algoritmos usan el esquema de clasificación Galton-Henry [44] que divide las huellas en 5 clases (arco, arco en tienda, bucle a la izquierda, bucle a la derecha y espiral) (ver figura 1.2) y su distribución natural es de 3.7 %, 2.9 %, 33.8 %, 31.7 % y 27.9 % respectivamente [69]. Los inconvenientes que encontramos en dichos algoritmos son varios. Primero, la distribución de las huellas es muy desbalanceada entre clases y no logra reducir suficientemente las huellas candidatas; segundo, son muy pocas las clases en las que se dividen las huellas [16]. El tercero y más importante es la imposibilidad de clasificar una huella latente dactilar por causa de la mala calidad que generalmente tienen [44]. Esto trae consigo que los clasificadores den como resultado que prefieran descartar la huella por falta de información [44] y por tanto no se pueda llevar a cabo la selección de las huellas candidatas. Por estas razones los algoritmos de clasificación, mayormente, se usan en combinación con los métodos de comparación de huellas para la verificación, en el que la huella consulta también es tomada en condiciones controladas para maximizar su calidad.

1.1.2. Subclasificación de huellas

Otro tipo de algoritmos para la identificación rápida de huellas son los sub-clasificadores. Estos se aplican en las huellas clasificadas como bucles o espiral, donde se usa el conteo de crestas entre los bucles y los deltas [53]. Este método es incluso más difícil de desarrollar que los que usan la clasificación original en 5 clases, porque las reglas varían para cada uno de los tipos

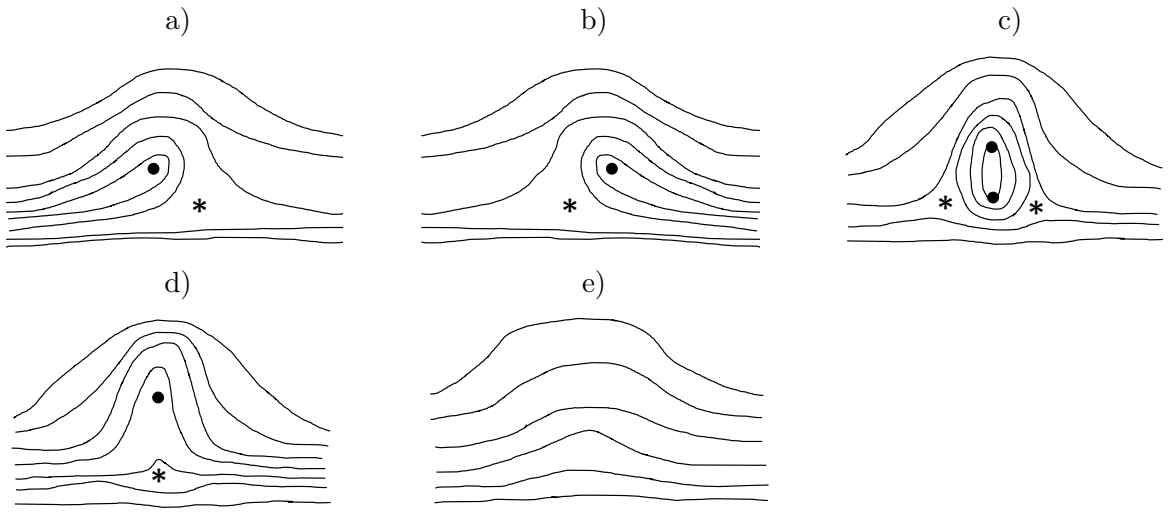


FIGURA 1.2: Clases de huellas: a) bucle a la izquierda, b) bucle a la derecha, c) espiral, d) arco en tienda, y e) arco. Los círculos indican el punto singular de los bucles y los asteriscos el delta.

de dedo al que pudiera pertenecer la huella. Además, presentan el mismo inconveniente que los algoritmos de clasificación respecto a la calidad de las huellas latentes [44].

1.1.3. Indexación de huellas

Como último grupo se encuentran los algoritmos de indexación, también llamados clasificación continua. Según Davide Maltoni et al. [44] y Capelli et al. [16], la indexación es más apropiada que los métodos anteriores cuando el propósito es identificar huellas latentes, debido a la baja calidad que presentan las huellas latentes. Estos algoritmos se basan en crear vectores numéricos que representan las características de las huellas. Mediante estos vectores se calcula el nivel de similitud de dichas características y se establecen relaciones según su cercanía. Estos algoritmos han sido utilizados con una gran variedad de técnicas que incluyen estructuras de datos espaciales, conteo de crestas, alineación de imágenes respecto a puntos singulares (núcleo y delta de la huella), plantillas para las clasificaciones Galton-Henry y agrupamiento por descriptores de minucias [44].

En nuestra opinión, los métodos basados en descriptores de minucias tienen posibilidades de obtener menores tasas de error ante los problemas presentes en una huella latente. Esto se sustenta debido a que la mayoría de estos descriptores son invariantes a la traslación y rotación de la huella [44], son tolerantes a la baja calidad de la imagen obtenida [29] y tolerantes a errores del extractor de rasgos (ausencia de minucias existentes o inclusión de minucias falsas) [31, 44].

Además, estos descriptores son los más usados en publicaciones recientes [14, 16, 33, 39, 40, 58]. Mientras que las técnicas restantes usan rasgos dependientes de una alta calidad en la huella.

Los descriptores de minucias están formados por cantidades variables de minucias y pueden encontrarse combinados con otros rasgos [33]. Entre los descriptores que se han usado para representar la topología de las minucias se encuentran: elementos de la triangulación de Delaunay [5, 61], estructuras convexas [33], estructuras estelares de longitud variable [2, 47] y códigos cilíndricos [16, 65]. De estos se han detectado diferentes problemas; los algoritmos basados en triangulación presentan problemas ante distorsiones de las huellas y requieren complejas modificaciones en el algoritmo original [39, 40, 49], las estructuras convexas y estelares dependen de la existencia de rasgos de orden uno (núcleos y deltas) y los códigos cilíndricos solo destaca como aspecto negativo el tamaño de almacenamiento del descriptor [65].

Aplicado a los anteriores descriptores, se han usado como mecanismo de agrupamiento: agrupamiento [27, 33], árboles [2, 47], algoritmo FLASH [23, 49], proyección mediante función *hash* simple [5, 39, 40, 61] y mediante familias de funciones como LSH (*Local Sensitive Hashing*) [4, 65]. Los resultados obtenidos por la combinación de códigos cilíndricos con estrategias de agrupamiento LSH en el algoritmo de Capelli et al. publicado en 2011 [16] se encuentran entre los que menor error reportan. Es válido aclarar que de este algoritmo solo se han publicado resultados en problemas de verificación.

Todos los enfoques mencionados hasta el momento tienen como limitante que usan impresiones como huella consulta, donde la calidad de las imágenes es mejor que en el caso de las huellas latentes. Por otro lado, las investigaciones realizadas para el problema de indexación con huellas latentes han sido muy pocas en comparación con las relacionadas con impresiones [55] y en la mayoría de estas se estipula que usar un solo descriptor es insuficiente, producto de la poca información disponible en una huella latente [20, 55]. Además, todos estos algoritmos que combinan descriptores usan el MCC SDK propuesto en [16]. Por lo que mejorar la tasa de error de dicho SDK, basado solamente en código cilíndrico de minucia, mejorará el desempeño de los indexadores que combinan sus resultados en sus mecanismos de integración de votos. Esta tarea es la motivación principal de nuestra investigación.

En resumen, el problema presente en la indexación de huellas latentes dactilares es que, en el propósito de obtener un reducido conjunto del espacio de búsqueda (impresiones en bases de datos disponibles), aún existe un margen de mejora significativo en el error cometido en las listas propuestas por estos algoritmos, para pequeños porcentajes de las bases de datos.

1.2. Hipótesis

Analizado el problema y los antecedentes se plantea la siguiente hipótesis: Si se crea un algoritmo de indexación basado en códigos cilíndricos de minucias, usando algoritmos de agrupamiento, entonces se puede mejorar la tasa de error de los algoritmos de indexación reportados en la literatura con el mismo descriptor, sin aumentar significativamente el tiempo de identificación de huellas latentes dactilares.

Con lo cual se dará respuesta a las siguientes preguntas de investigación:

- ¿Qué tanta eficiencia en la identificación de huellas latentes hay que sacrificar para disminuir la tasa de error usando códigos cilíndricos de minucia como descriptor?
- ¿Que tan uniforme pueden agruparse los descriptores de huellas de manera que se minimice el error cometido en la identificación?

1.3. Objetivos

El objetivo general del presente trabajo es desarrollar un algoritmo de indexación basado en minucias para una identificación rápida de huellas latentes dactilares, que disminuya la tasa de error de los algoritmos de indexación basados en códigos cilíndricos reportados en la literatura, con costo computacional aceptable.

Se considera un costo computacional aceptable siempre y cuando, en términos de complejidad temporal, no sea mayor a un orden de magnitud del costo del algoritmo propuesto por Cappelli et al. [16].

De este objetivo general se derivan los siguientes objetivos particulares:

1. Diseñar una estructura de indexación que organice los códigos cilíndricos según su cercanía.
2. Desarrollar un algoritmo para la identificación rápida de huellas latentes que sea robusto a la ausencia de información que tienen las mismas.
3. Crear mecanismos para realizar una cantidad balanceada de comparaciones con los descriptores de minucias extraídos de la estructura de indexación, para disminuir la cantidad de huellas que se usarán en la comparación con la huella latente.

La cercanía entre dos códigos cilíndricos se define como una función que toma en cuenta la cantidad de bits comunes entre dos de estas estructuras. A mayor cantidad de bits en común, mayor es la probabilidad de que las minucias correspondientes sean coincidentes.

En esta investigación consideramos como robusto a la ausencia de información que tienen las huellas latentes, aquel algoritmo que sea capaz de, dado un código cilíndrico que se desea buscar, encontrar en la estructura indexada los códigos cercanos sin que necesariamente sean iguales al código buscado.

1.4. Contribuciones

Las contribuciones de esta investigación son las siguientes:

1. Una estructura de indexación basada en agrupamiento de códigos cilíndricos de minucias, que garantiza la cercanía de estos descriptores y con ello maximizar la probabilidad de extracción de los descriptores correspondientes a la impresión que coincide con la huella latente.

2. Un algoritmo para recuperar cantidades homogéneas de descriptores, sin que se pierda el criterio de cercanía, a pesar de que los grupos de descriptores presenten distribución heterogénea.

3. Un algoritmo de indexación aplicable a huellas latentes dactilares que disminuye la tasa de error usando solo descriptores de códigos cilíndricos de minucias. Estos resultados son alcanzables para pequeños por cientos de la base de datos de impresiones, lo cual es coherente con el objetivo de este tipo de algoritmos que buscan reducir el espacio de búsqueda.

1.5. Organización de la tesis

El presente documento se organiza de la siguiente forma: en el capítulo 2 se analizan los diferentes enfoques para resolver el problema expuesto anteriormente. A continuación, el capítulo 3 presenta un nuevo algoritmo para indexar usando huellas latentes dactilares basados en códigos cilíndricos de minucias y en el capítulo 4 se exponen los resultados experimentales del nuevo

algoritmo propuesto, así como su análisis. Finalmente, se abordan las conclusiones generales y trabajos futuros.

Capítulo 2

Trabajos relacionados

En este capítulo se analizan las etapas que componen a los algoritmos usados para reducir el espacio de búsqueda de huellas candidatas. También un análisis de la literatura para los algoritmos de indexación de huellas, tanto de impresiones, como de huellas latentes.

Cómo rasgo fundamental de las huellas, para esta revisión fue seleccionada la minucia, por las características mencionadas en el capítulo 1. Se detalla una taxonomía orientada al número de minucias involucradas en los descriptores que son usados por los algoritmos de indexación de impresiones. Además, se reportan todos los trabajos encontrados para huellas latentes, los cuales son muy pocos y exponen la necesidad de combinar varios descriptores.

2.1. Marco teórico

Las dificultades intrínsecas de la clasificación y sub-clasificación automatizada de huellas, mencionadas en el capítulo 1, originó que se investigara en sistemas de identificación rápida de huellas que no estuvieran basados en clases definidas por humanos [44]; de esta forma surgieron los métodos de indexación [43]. En indexación las huellas no son separadas en diferentes clases, sino asociadas con uno o varios vectores numéricos. Usando estos vectores las huellas pueden ser representadas en un espacio multidimensional de manera tal que sus coordenadas estén próximas si pertenecen a huellas similares. Mientras más similares las huellas, más cercanas estarán en el espacio coordenado.

Luego es necesario un procedimiento de recuperación que se lleva a cabo colocando la información de la huella consulta en dicho espacio y seleccionando n huellas. Los vectores extraídos son

los más cercanos, en dependencia de una función distancia, por ejemplo, distancia euclidiana [44] o distancia de Hamming [16].

A continuación, se detallarán las etapas anteriormente mencionadas.

2.1.1. Algoritmo de indexación

Los algoritmos de indexación dividen conjuntos de huellas de manera apropiada según su similitud usando sus rasgos (minucias en el presente caso) y formando topologías mediante la cual se relacionan [33]. La forma de establecer las representaciones locales de las minucias varía en la cantidad de minucias vecinas que son relacionadas y de este número depende en gran medida la topología que las rige. Por ejemplo, cuando se usan relaciones ternarias entre las minucias más cercanas se crean triangulaciones y en caso de no existir la restricción de cercanía, se crea un entramado entre las minucias formado por todas las combinaciones de éstas, agrupadas en cantidades determinadas por el algoritmo de indexación [23].

Los diferentes enfoques que se han utilizado para indexar descriptores de minucias están representados por funciones hash [16], árboles [47] y la triangulación de Delaunay [5, 49]. En el caso de las funciones hash H, se proyecta un valor de un conjunto con muchos miembros, incluso una cantidad infinita, a un valor de otro conjunto con un número fijo de miembros [68]. De esta forma los descriptores de minucias son agrupados según el valor que retorne la función hash. También existen familias de funciones hash que tienen la propiedad de ser sensibles a localidad (LSH, por sus siglas en Inglés) [16]. Estas familias de funciones se definen como $f_{H_i} : \{0,1\}^d \rightarrow \{0,1\}^r$, siendo $0 < i <= h$ y h la cantidad de funciones hash a utilizar, d la dimensión del vector que define al descriptor de la minucia y r la dimensión del vector resultante al que es proyectado el vector original. La proyección se realiza usando los subíndices definidos por $H_i = \{s_1, s_2, ..., s_r\}$ tal que $1 <= s_j <= d$, los cuales son seleccionados aleatoriamente para cada función. Estas familias de funciones obtienen a partir del vector del descriptor de una minucia, una representación de menor dimensión y lo asignan a un índice en la tabla hash asociada a cada función, ver figura 2.1. LSH es un método genérico y probabilístico que es efectivo para encontrar los elementos más cercanos a un vector de grandes dimensiones [33].

La estructura de datos abstracta llamada “árboles” también ha sido usada para indexar los descriptores de minucias. Estos simulan la estructura jerárquica de los árboles naturales, con un nodo raíz y subárboles a partir de éste. Los subárboles, a su vez, se definen de igual forma, por

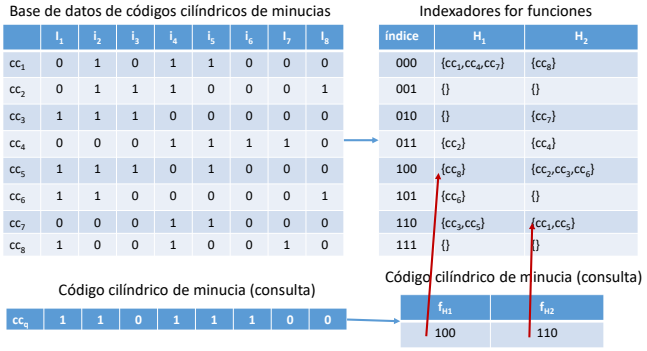


FIGURA 2.1: Ejemplo del esquema de indexación mediante familia de funciones hash. Parámetros: $d = 8$, $r = 3$, $h = 2$, $H_1 = \{i_1, i_3, i_8\}$, $H_2 = \{i_2, i_5, i_6\}$. De izquierda a derecha, de arriba hacia abajo; la base de datos con los descriptores se inserta en las tablas hash usando f_{H1} y f_{H2} para determinar los índices, luego el descriptor consulta también es evaluado por las funciones hash y se obtienen los respectivos índices. En este caso sintético no existen coincidencias, pero en casos de descriptores reales se retornan los descriptores que más coincidan a través de las tablas de indexación.

lo que van construyendo una estructura recursiva hasta llegar a los valores "hojas" o terminales [34]. La relación directa entre nodos se representa a través de arcos y se establecen, en el contexto de indexación, usando una medida entre los descriptores que corresponden a los arcos. El procedimiento de indexación en un árbol comienza seleccionando un descriptor de la impresión, equivalente al nodo raíz del árbol (figura 2.2). Luego, se selecciona un descriptor vecino según una función que establece el tipo de relación que se desee. Esta relación también es buscada en los nodos que forman la descendencia del nodo inicial en el árbol, seleccionándose el más apropiado. De esta forma se construye un camino donde al final, en un nodo 'hoja', se inserta el identificar de la huella a la que corresponden los descriptores utilizados [47], (figura 2.3).

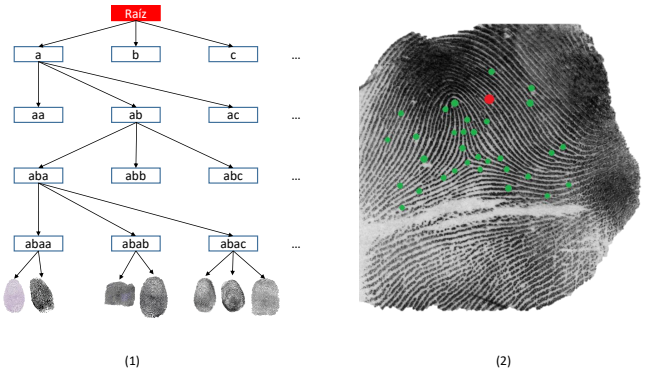


FIGURA 2.2: Ejemplo de indexación de una impresión usando árboles. Un descriptor de minucia de la impresión (2) es seleccionada como nodo raíz (punto rojo) para insertar en el árbol (1).

Como último mecanismo de indexación se analiza la triangulación de Delaunay. Este es un

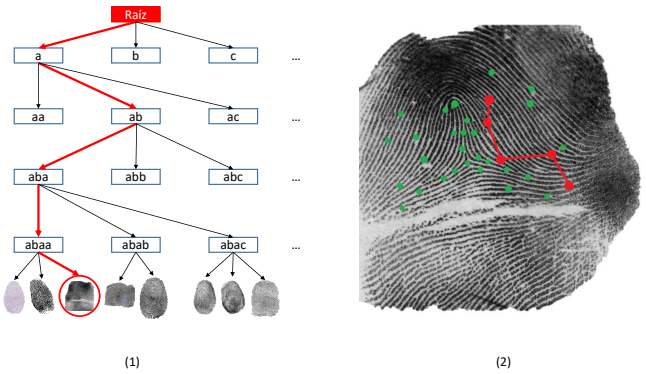


FIGURA 2.3: Continuación del ejemplo de indexación de una impresión usando árboles. Se busca un camino desde el nodo raíz a través de sus vecinos (2) y la correspondiente expansión de ramas en el árbol (1), hasta encontrar un nodo hoja, donde se inserta el identificador de la impresión.

concepto que proviene de la especialidad de geometría computacional y consta de la siguiente definición: dado un conjunto de puntos discretos P en el plano, $T(P)$ es una triangulación si ningún punto se encuentra dentro del círculo circunscrito de algún triángulo perteneciente a $T(P)$ [19]. La triangulación de Delaunay tiene la propiedad de maximizar el menor ángulo que forma cada triángulo y por tanto las longitudes de los lados serán las menores posibles y los puntos (minucias) que se conectan son los vecinos más cercanos. De esta forma se toman los tríos que conforman los triángulos como el descriptor a indexar. Al considerar solo una cantidad de triángulos correspondiente a la cantidad de minucias, este método logra una complejidad temporal y espacial de un $O(n)$ en la etapa de indexación. La indexación en estos casos se realiza, en su mayoría, proyectando los tríos de minucias que conforman los triángulos, en una representación vectorial y usando esta en conjunto con una función hash. Otros casos más particulares se analizarán en el epígrafe 2.2.3.

2.1.2. Algoritmo de recuperación

Luego de organizar la base de datos de huellas plantillas en una estructura bajo el criterio de máxima similitud, procede usar una estrategia de recuperación para conformar la lista de huellas candidatas a usar en la identificación de la huella latente. Estos métodos pueden ser técnicas incrementales o de filtrado [44].

En las técnicas de filtrado, en modo general, se aplica a los descriptores de la huella consulta las mismas operaciones que a los descriptores de las huellas plantillas, para obtener el subconjunto de descriptores al cual pertenecería si fuera a indexarse. Esta operación se le conocerá a partir

de ahora como *consulta*. Luego se construye un orden de preferencia con todas las huellas plantillas que coinciden con dicho subconjunto, en base a la similitud con los descriptores de la huella consulta [28]. El principal objetivo de estos algoritmos es que incluyan en la lista de huellas candidatas, la huella que coincide con la huella consulta para lograr su identificación correctamente.

Tanto las funciones hash simples [5, 6, 21, 30, 40, 60] como las familias de funciones LSH [16, 58, 65] son utilizados en combinación con el mecanismo general descrito anteriormente, como mecanismo de recuperación de huellas. La especificidad con las familias de funciones es que dado un descriptor de minucia se obtiene una lista por cada tabla hash. Con todas estas listas se crea un orden en las huellas en dependencia de la cantidad de descriptores de minucias coincidentes que le corresponden a cada una.

Entre las técnicas de filtrado también encontramos mecanismos de agrupamiento mediante árboles [47]. En este caso el árbol representa también la estructura de indexación y cuando se procede a la obtención de las n huellas candidatas, la huella consulta debe intentarse insertar hasta llegar a un nodo en el que el tamaño del subárbol con raíz en dicho nodo, minimice la diferencia con el valor n solicitado. Sin embargo, es común que para satisfacer un número determinado se deban hacer operaciones adicionales, En caso de completar las n huellas, se usan consultas a los subárboles vecinos siguiendo el mismo patrón de comportamiento que en una estructura de datos de consulta de rango espacial como el kd-tree [56].

Uz et al. [61] muestra un caso especial de agrupamiento, el cual es incremental. En este, primero se utilizan todas las impresiones pertenecientes a un mismo dedo, se calculan las minucias coincidentes y se asigna un peso a cada minucia. Este peso representa la cantidad de impresiones en las que fue encontrada y a mayor peso, mayor es la calidad de la minucia. Luego, los autores crean la triangulación de Delaunay de manera creciente, comenzando por las minucias de mayor calidad y adicionando progresivamente las de un nivel de calidad inferior en cada paso. De esta forma, logran dejar de considerar cierta cantidad de minucias por huellas al encontrar en las comparaciones de los primeros niveles un grado de similitud por encima de un umbral establecido.

Existen otros enfoques que son muy específicos del descriptor de minucia, pero los fundamentales son los abordados anteriormente y entre ellos los que usan indexado por tablas hash producen menos errores de identificación que los demás [65].

2.2. Trabajos relacionados con indexación de impresiones

A continuación, se analizarán las investigaciones más recientes y relevantes para mostrar las diferentes soluciones al problema de reducir el espacio de búsqueda en el reconocimiento de huellas, con el menor error posible. Todos los algoritmos expuestos a continuación están basados en minucias, como rasgo principal.

La clasificación de los algoritmos de indexación basados en el número de minucias relacionadas, según Khodadoust and Khodadoust [33], es la siguiente: enfoques basados en una minucia, pares de minucias, tríos de minucias, cuádruplos de minucias, K minucias y códigos cilíndricos de minucias (MCC, por sus siglas en inglés). En la tabla 2.1 se puede encontrar un mayor nivel de detalle en el que se organizan los diferentes algoritmos reportados.

Las métricas de evaluación presentes en la tabla 2.1 pueden consultarse en las respectivas publicaciones. Su inclusión en la tabla es para mostrar cómo la mayoría de los algoritmos reportados en la literatura no usan el protocolo de evaluación estandar que usa la tasa de error contra el porcentaje de base de datos.

TABLA 2.1: Resumen de las características de los algoritmos más relevantes basados solamente en minucias en el contexto de indexación de impresiones.

Autores	**Descriptor(es)**	**Tamaño de BD (huellas)**	**Resultados**
Jayaraman et al. (2014) [30]	MBP	800	95 % HR en 2.24 % PR
Tiwari et al. (2015) [60]	CGTC	800	100 % HR en 1.32 % PR
Wang et al. (2012) [63]	DITOM	800	7.5 % EER
Bhanu et al. (2003) [6]	Trío de minucias	5000	72.4 % CIP
Choi et al. (2003) [18]	Trío de minucias	1360	99.2 % PR en 10 % Rank
Bebis et al. (1999) [5]	Triángulos	300	94.2 % AAcc
Liang et al. (2007) [40]	Triángulos	880	100 % HR en 20.9 % PR
Muñoz-Briseño et al. (2013) [49]	Triángulos	24,000	menos del 98 % CIP en 30 % PR
Gago et al. (2013) [21]	Triángulos	24,000	98 % CIP en 30 % PR
Uz et al. (2009) [61]	Triángulos	800	1.53 % FRR en 0 % FAR
Iloanusi et al. (2011) [27]	Cuádruplos	800	100 % HR en 5.2 % PR
Iloanusi et al. (2014) [28]	Cuádruplos	800	100 % HR en 5 % PR
Chikkerur et al. (2006) [17]	k minucias vecinas	800	1.5 % EER
Mansukhani et al. (2010) [47]	Grafo	800	0.95 % AAcc
Bai et al. (2015) [2]	k minucias vecinas	2,000	menos del 94 % CIP en 10 % PR
Cappelli et al. (2011) [16]	MCC	24,000	9 % ER en 1 % PR
Wang et al. (2015) [65]	MCC compactado	20,000	92 % HR en 10 % PR
Bai et al. (2018) [4]	MCC compactado	27,000	4 % ER en 10 % PR
Bai et al. (2018) [3]	MCC compactado	27,000	3 % ER en 10 % PR

Las abreviaturas de las métricas utilizadas en los diferentes protocolos de evaluación se encuentrán en inglés.

PR:	Porcentaje de base de datos	ER:	Tasa de error
HR:	Tasa de éxito	EER:	Punto de equilibrio del error
CIP:	Potencia de índice correcto	Rank:	Posición en un ordenamiento
FRR:	Tasa de falsos rechazos	FAR:	Tasa de falsa aceptación
AAcc:	Eficacia promedio		

2.2.1. Indexación basada en una minucia

La indexación por una minucia ha sido tratada usando varias representaciones, entre ellas patrones binarios de minucia (MBP, por sus siglas en inglés) [30] el cual usa un vector de longitud fija y una tabla hash para indexar, colocando el vector sólo una vez en la estructura de indexación. Estas características propician realizar comparaciones de una manera eficiente y una complejidad espacial del algoritmo reducida ($O(n)$). Otra representación con base en una sola minucia es el "código de pista gaussiana coaxial" (CGTC, por sus siglas en inglés) [60] y también usa un vector de longitud fija. Además, ambas estrategias de indexación son invariantes a la rotación y a la traslación.

Sin embargo, este enfoque tiene una serie de inconvenientes, puesto que analizan la información por sectores y no tienen una estrategia para resolver de manera robusta situaciones en las que se encuentran minucias en los límites de dos sectores. También se encuentran en desventaja con los métodos basados en más de una minucia [33] por contener menos información de la vecindad de la minucia. Además, estas representaciones usan la información espacial (distancia) y direccional (ángulo) de la minucia respecto al núcleo de la huella y se conoce que en huellas clasificadas como arco y bucles gemelos, figuras 1.2.e y 1.2.c respectivamente, no hay núcleos o hay más de uno, respectivamente [44]. Por último, al intentar llevar este descriptor al contexto de huellas latentes, encontrar el núcleo representa un reto porque la fracción obtenida de la huella latente puede no contenerlo y la calidad de la imagen puede provocar que se detecte un núcleo falso o ninguno.

2.2.2. Indexación basada en pares de minucias

El enfoque basado en dos minucias se encuentra reportado por Wang and Hu [63]. El vector formado por las características extraídas de los pares de minucias en este algoritmo es invariante a cualquier transformación geométrica [33]. El mecanismo de indexación usa una cadena binaria y tiene propiedades adicionales útiles para el contexto de seguridad. Este enfoque cuenta con el inconveniente de que analiza todos los pares de minucias en una huella $n(n-1)/2$ y por tanto su complejidad computacional es $O(n^2)$.

2.2.3. Indexación basada en tríos de minucias

En el caso de los tríos de minucias, el primer intento de indexación usando este número de minucias y los triángulos que se forman entre ellas fue publicado en Germain et al. [23]. El método consiste en registrar la correspondencia entre los tríos de la huella consulta y la huella plantilla usando técnicas de agrupamiento sobre los parámetros que conforman la representación. Este algoritmo tuvo varias mejoras que radicaban en nuevas características en los tríos de minucias [6, 8, 18], no obstante aún todos poseen una complejidad temporal $O(n^3)$ debido a que la cantidad de tríos es $\binom{n}{3} = \frac{n!}{3!(n-3)!}$.

La triangulación de Delaunay ha sido usada para reducir la complejidad temporal de los métodos anteriores [10]. Pero aun así este método presenta problema de sensibilidad a las alteraciones en la posición y el número de las minucias que se encuentren presentes en la huella (tanto por la ausencia de minucias reales, como por la presencia de minucias falsas) ya que por cada configuración se genera una triangulación diferente [5]. Otros algoritmos se han creado para mejorar estos inconvenientes [21, 40, 49]. Entre ellos, Liang et al. [40] crea una triangulación con un subconjunto de las minucias disponibles, por lo que se pierde información para realizar una correcta indexación y solo reduce el efecto negativo de las traslaciones de las minucias en la huella. Por otro lado, Muñoz-Briseño et al. [49] y Gago-Alonso et al. [21] tienen como inconvenientes las distorsiones que se presentan en las huellas latentes y la incertidumbre que introduce uno de los elementos de la tupla que describe los tríos de minucias. Este elemento es calculado mediante el conteo de crestas interceptadas por las aristas de los triángulos y por tanto puede presentar problemas cuando uno de los lados es paralelo a la estructura de las crestas. También presenta dificultades cuando existen deformaciones producidas por una cicatriz y hay pérdida de la información de las crestas, conllevando a un conteo incorrecto.

Otro intento de usar la triangulación de Delaunay es a través de una estrategia jerárquica. Uz et al. [61] usa este enfoque y clasifica las minucias en dependencia de la calidad que presenten. En este algoritmo la indexación se realiza implícitamente durante el análisis de coincidencia de las minucias. Esto se evidencia en la generación progresiva de triangulaciones, comenzando sólo con las minucias de más alta calidad y añadiendo en cada paso las minucias categorizadas de calidad inmediata inferior, hasta que se usen todas las minucias disponibles. El algoritmo se detiene en el nivel donde el grado de coincidencia es satisfactorio, dejando sin usar cierta cantidad de minucias en dependencia del nivel. El inconveniente de este enfoque es que usa

una triangulación como la propuesta en Bebis et al. [5] con su consiguiente vulnerabilidad ante distorsiones, ruido en la huella latente, ausencia de minucias y presencia de minucias falsas.

2.2.4. Indexación basada en cuádruplos de minucias

Los cuádruplos de minucias forman diferentes simetrías clasificadas en: convexas, cóncavas y reflexivas [27]. De estos, los cuádruplos convexos son usados en conjunto con técnicas de agrupamiento para mejorar el rendimiento de la indexación de estas minucias [27, 28]. Sin embargo, la complejidad de estos algoritmos es de $O(n^4)$ [42].

2.2.5. Indexación basada en k minucias

Tanto el enfoque de k minucias, como el de códigos cilíndricos usan la información de las minucias a su alrededor relacionadas con la minucia central sólo por la distancia euclidiana. Los dos enfoques primarios que se usan en estos métodos tienen las siguientes bases: las n minucias más cercanas a la minucia $m = \{x_m, y_m, \theta_m\}$ y las minucias localizadas en una circunferencia con centro $\left(x_m, y_m\right)$ y radio R. Según Cappelli et al. [14], el primer enfoque tiene la ventaja que al tener un descriptor de longitud fija proporciona una mayor eficiencia en las comparaciones. El inconveniente de esta forma de representación radica en que, debido a la ausencia de información en las huellas latentes dactilares, esas últimas minucias vecinas podrían estar tan alejadas que introducen información que no es útil localmente. Mientras que el segundo enfoque, al tener sólo la información de las minucias dentro en un radio R, en principio, son más tolerantes a la ausencia de minucias y a la introducción de minucias falsas por parte de los extractores.

La representación mediante k minucias es una extensión de los algoritmos del vecino más cercano. Al escoger sólo k minucias de su vecindad local la complejidad espacial y temporal es $O(nk)$ y dependerá de cuál sea la diferencia entre las k minucias vecinas que se toman por cada una de las n minucia que existen en la huella para que se considere lineal o cuadrática [33]. Estos algoritmos son invariantes a las rotaciones y traslaciones de las minucias. Chikkerur et al. [17] fue la primera propuesta con esta representación de minucias formando un grafo para realizar la indexación. En su lugar, otros usan árboles, donde el tiempo de análisis de coincidencia de las minucias depende solamente de la configuración de los índices del árbol y no del número de las huellas plantillas [47]. Bai et al. [2] propone un híbrido, donde crea un mecanismo de indexación de múltiples vías basado en k minucias y árboles de 3 niveles. El principal problema que presenta

esta estructura es que no garantiza conexión en toda la huella. Además, los algoritmos que usan triangulación sólo necesitan 2.6 arcos, en promedio, para garantizar conectividad en el grafo que forman [25], mientras que en la presente forma de representación se necesitan como mínimo cuatro arcos por minucia, en detrimento de la complejidad temporal del algoritmo.

2.2.6. Indexación basada en códigos cilíndricos de minucias

La representación por códigos cilíndricos de minucias [14] está basada en la información de las minucias en un radio r y consiste en un vector v de dimensión d que contiene la vecindad de una minucia m, figura 2.4. Este vector está formado por la representación binaria de varias secciones cilíndricas y en cada una se encuentra sólo la información aportada por las minucias con una diferencia angular respecto a la minucia m en un rango determinado. Estos rangos de diferencias angulares son del mismo tamaño y cubren los 2π radianes, por lo que, dado q rangos, estos cubrirán las siguientes sectores angulares: $\left[0, \frac{1}{q}2\pi\right)$, $\left[\frac{1}{q}2\pi, \frac{2}{q}2\pi\right)$ ... $\left[\frac{q-1}{q}2\pi, 2\pi\right)$.

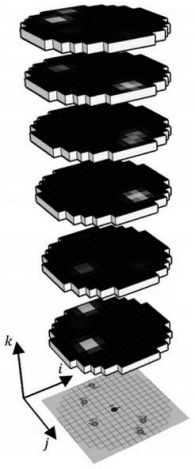


FIGURA 2.4: Representación gráfica de un código cilíndrico de minucia.

La representación por códigos cilíndricos de minucias tiene las siguientes ventajas: usa una estructura basada en radio fijo y el vector generado para las minucias en la vecindad geométrica es también de longitud fija, por lo que los cálculos entre sí son menos complicados; los problemas de las minucias que se encuentran cerca de los bordes de las circunferencias son tratados de manera que no introducen carga adicional en la etapa de identificación y verificación. Por último, la representación puede ser binaria, de esta forma la comparación de estos descriptores puede llevarse a cabo de manera optimizada al más bajo nivel de instrucciones en un procesador.

El algoritmo de Capelli et al. [16] usa este descriptor de minucias en combinación con funciones hash LSH obteniendo un mecanismo que usa diferentes configuraciones del vector original (valores determinados por un subconjunto arbitrario de bits que pertenecen a varios rangos de diferencias angulares). Este aspecto podría indicar que, bajo varias funciones de indexado, el margen de error en discriminar dos minucias como no coincidentes se minimizaría debido a que las funciones hash generan diferentes configuraciones de minucias coincidentes y por tanto la unión de todos los resultados considerarían más combinaciones de minucias. Sin embargo, este método presenta el inconveniente de que no son usadas todas las funciones para indexar los códigos cilíndricos, sino que son seleccionadas aleatoriamente, por lo que no todos los códigos cilíndricos serían indexados mediante las mismas funciones. Por la razón anterior, el grado de incertidumbre de cómo se lleva a cabo la selección de minucias candidatas a coincidentes es alto y no se reporta el estudio probabilístico pertinente para demostrar que su comportamiento es fiable. Además, esta estrategia tiene problemas ante la ausencia de minucias o la presencia de minucias falsas, principalmente en huellas pequeñas o de baja calidad [33].

Los códigos cilíndricos de minucias han sufrido alteraciones en su estructura a través de compactaciones para realizar comparaciones más rápidas [4, 65] en las estructuras de indexación. En el mismo sentido de este enfoque se han introducido las redes neuronales convolucionales (CNN, por sus siglas en Inglés) [3]. Los cuales, por una parte, tienen la ventaja de que la salida de estos algoritmos es una versión compacta de los códigos cilíndricos de minucias que se usan directamente como llaves en las tablas hash, haciendo la búsqueda más rápida. Por otra parte, tienen el inconveniente de que su tamaño no debe exceder de 32 bits para que la búsqueda en las llaves de la estructura de indexación sea más eficiente que una búsqueda secuencial [65]. Esta restricción de tamaño también provoca que se agrupen una mayor cantidad de descriptores por llave, afectando el poder de reducción en el espacio de búsqueda.

2.3. Trabajos relacionados con indexación de huellas latentes

El número de investigaciones orientadas a indexación con huellas latentes es muy pequeño en comparación con los mencionados en la sección 2.2. Estos se enfocan en la combinación de varios descriptores para filtrar las impresiones y de esta forma tratar de resolver el problema de la falta de información en las huellas latentes (tabla 2.2).

TABLA 2.2: Resumen de las características de los algoritmos reportados en el contexto de indexación de huellas latentes dactilares.

Autores	**Rasgo(s)**	**Descriptor(es)**	**No. de huellas**	**Resultados**
Feng and Jain (2008) [20]	Clase de la huella, puntos singulares, crestas	Clasificación; núcleo y orientación del delta; campo de orientación y curvatura	10,258	97.3 % HR en 10 % PR
Paulino et al. (2013) [55]	Minucia, cresta, puntos singulares	Tríos de minucias y MCC; campo de orientación y frecuencia de crestas; núcleo y deltas	267,258	81.8 % HR en 10 % PR
Krish et al. (2014) [35]	Crestas	Tensor de orientación	88	cerca del 68 % HR para rank-1
Krish et al. (2015) [36]	Crestas	Tensor de orientación	258	80.62 % HR para rank-1

Las abreviaturas de las métricas utilizadas en los diferentes protocolos de evaluación se encuentran en inglés.

PR: Porcentaje de base de datos ER: Tasa de error

HR: Tasa de éxito Rank-1: Primera posición del ordenamiento

Los trabajos analizados se presentan bajo dos categorías: los que usan varios indexadores (un indexador por cada descriptor que se utiliza), llamados en este trabajo 'indexadores múltiples' y los que reducen el número de huellas de las bases de datos de huellas usando la combinación de varios descriptores en uno más complejo. Estos últimos son conocidos en la literatura como 'pre-registro', no tienen control del porcentaje a reducir en las bases de datos y en su mayoría reportan sus resultados combinados con algoritmos de identificación de huellas.

2.3.1. Indexadores múltiples

Feng et al. [20] desarrollaron un método de filtrado usando tres descriptores. En primer lugar, usan la clase de la huella teniendo en cuenta sus cinco clasificaciones. Como segundo caso, se utilizan los puntos singulares como el núcleo y delta, detectados mediante el método de índice de Poincaré y la dirección de los deltas calculados a través de los campos de orientación. Por último, se usa el campo de orientación y de curvatura de la huella en la vecindad de los puntos singulares mencionados. Para que una impresión sea considerada como una huella candidata debe formar parte del resultado de los 3 filtros. Este método tiene el inconveniente de que la extracción de estos descriptores es un proceso complejo y en ocasiones ni siquiera es posible cuando la huella es de muy mala calidad [54].

Otro ejemplo de combinaciones de descriptores es planteado por Paulino et al. [55]. En este trabajo se usan tríos de minucias, códigos cilíndricos, campo de orientación, puntos singulares y período de crestas. El proceso de indexación según el descriptor se lleva de la siguiente forma:

- Los códigos cilíndricos de minucias se indexan mediante el MCC SDK, pero usando 64 funciones hash en vez de las 32 utilizadas por defecto.
- El campo de orientación alrededor de cada minucia es convertido en un vector binario de longitud fija, invariante a rotación y traslación. Esto se realiza muestreando 76 puntos en total en cuatro círculos concéntricos en la minucia. La indexación de este descriptor se realiza usando funciones hash y la similitud entre descriptores se calcula de manera similar al método de los códigos cilíndricos de minucias.
- Los tríos de minucias usan una indexación convencional [23] y en la etapa de recuperación la búsqueda es acelerada incorporando restricciones de rotación, es decir, diferencias de más de 60 grados descartan al descriptor bajo análisis. En este caso, cada impresión obtendrá como valor de similitud el número de tríos coincidentes con la huella latente.
- En el caso de los puntos singulares y el periodo de crestas, se usan como filtros. Con los puntos singulares se forman pares (núcleo-núcleo, núcleo-delta, delta-delta), entre la impresión y la huella latente. De cada par se calcula la distancia entre los puntos singulares y los ángulos de las direcciones de dichos puntos con la recta que los une. Con estos datos se calcula el grado de similitud mediante una fórmula a intervalos.

- Finalmente, se calcula el promedio del período de crestas en un círculo alrededor del núcleo para cada huella. Como función de similitud se utiliza $e^{-\frac{\|R_l - R_i\|}{\omega_R}}$, donde R_l y R_i son los promedios del período de crestas en la huella latente y la impresión, respectivamente, y ω_R es un término de normalización calculado por el promedio de $\|R_l - R_i\|$.

Luego, se integran los índices calculados para determinar la lista candidata de impresiones. Para ello primero se normalizan los índices para el caso de los códigos cilíndricos de minucias, los tríos de minucias y el descriptor para el campo de orientación, ya que son comparables y se les puede aplicar la regla de suma [50] a estos índices. En el caso de los puntos singulares se suma una unidad al índice normalizado, puesto que, en caso de no existir núcleo, ni delta, entonces los valores de similitud se establecen a 0. Por su parte, el índice del período de crestas ya se encuentra en el intervalo $[0, 1]$ y finalmente se multiplican estos tres últimos valores para obtener el grado de similitud de las huellas. Este método de indexación también presenta el problema de la dependencia con los puntos singulares. Además, incluye cómputo excesivo para el cálculo de los descriptores relacionados con las minucias [71].

En ambos trabajos se expone el criterio de no usar, solamente, un rasgo de las huellas para realizar la indexación. Su justificación se sustenta en que la información parcial que existe en las huellas latentes y el ruido de fondo que adquieren estas huellas al ser extraídas, crean condiciones difíciles para realizar una indexación correcta.

2.3.2. Pre-registro

También se han realizados trabajos que, si bien no clasifican como indexación, intentan agilizar la selección de impresiones para el paso de identificación de la huella latente. El trabajo de Krish et al. [35] se enfoca en la construcción de tensores basados en el campo de orientación de las huellas y su función de similitud está basada en la desigualdad de Schwarz [41]. A dicho tensor se le calcula la estructura envolvente y usándose este último como patrón, se busca en los tensores de las impresiones. Este proceso se repite rotando el patrón hasta 45 grados y donde la similitud sea máxima, se calcula el centro del patrón en la impresión (punto de registro). Luego se seleccionan las minucias dentro del círculo con centro en el punto de registro y radio igual al diámetro del patrón, y se calculan las distancias mínimas entre las minucias de la huella latente y la impresión. Si estas distancias no logran ser menores a 12 píxeles, la impresión es desechada como candidata.

En una extensión al estudio anterior realizada por los mismos autores, Krish et al. [36], se usaron no solo la similitud basada en la desigualdad de Schwarz, sino también disimilitudes basadas en la distancia de Manhattan y Euclidiana, así como una función de disimilitud basada en consistencia [32]. El método en general es el mismo, solo que, al obtener un índice final, se promedian los valores de disimilitud normalizados. En estos casos, si bien son resistentes a la falta de información, aun es sensible a altas deformaciones por la rigidez con la que se discriminan las impresiones. Además, su objetivo principal es reducir el número de minucias por huellas a ser comparadas por un algoritmo de identificación, pero no evita hacer el chequeo contra todas las huellas.

2.4. Conclusiones

La mayoría de los algoritmos analizados, provenientes de soluciones al problema de indexación para la verificación de impresiones, presentan muy buenos resultados en las tasas de error a medida que los descriptores incluyen más información producto del número de minucias que usan. El inconveniente radica en que de igual forma aumenta el tiempo en el que obtienen esos resultados. En el caso particular de los tríos de minucias, usados para triangulación, mejoran en esa deficiencia, pero resultan muy sensibles a la aparición de falsas minucias y ausencia de las originales. Mientras, en el enfoque de usar un número arbitrario de minucias, para el caso de las huellas latentes, trae como consecuencia que se incluyan minucias alejadas de la minucia de referencia, de tal forma que las deformaciones de las huellas comienzan a perjudicar el porcentaje de error de los algoritmos.

Los algoritmos que usan códigos cilíndricos de minucias presentan varias ventajas. Estos se benefician de este descriptor por ser invariante a rotaciones y traslaciones, además de poseer una representación binaria de tamaño fijo, lo cual le brinda la posibilidad de realizar operaciones rápidas utilizando directamente instrucciones optimizadas de bajo nivel. Estas características lo convierten, en nuestra opinión, en el mejor descriptor a usar en solitario. Sin embargo, los algoritmos de indexación con mejores resultados que utilizan este descriptor, generalmente, usan familias de funciones LSH, los cuales realizan selección aleatoria de las funciones hash y distribuyen estos descriptores bajo diferentes condiciones. Esto provoca que la tasa de error esté controlada por un comportamiento probabilístico cuyo impacto no ha sido estudiado en los algoritmos de indexación.

En el caso de los trabajos realizados en indexación para el problema de identificación de huellas latentes, se establece que la utilización de un solo descriptor es insuficiente para capturar la información relevante en estos casos, tal como se criticó en los trabajos sobre indexación de impresiones. En estos algoritmos, la principal deficiencia que se señala es el uso de descriptores que consideran los núcleos y deltas de las huellas. Estos representan un problema porque ya es conocido que en algunas clases de huellas no existen núcleos, además, debido a la mala calidad y la información incompleta que logra obtenerse en las huellas latentes, pueden no contener estos puntos singulares en general.

Capítulo 3

Nuevo algoritmo de indexación usando códigos cilíndricos de minucias

En el presente capítulo explicaremos los diferentes pasos para lograr los objetivos propuestos. La figura 3.1 muestra el esquema general de nuestra propuesta indicando el orden de las etapas a seguir. Primero, realizamos el preprocesamiento de un conjunto de bases de datos de impresiones para crear los grupos representativos de las mismas. Con estos grupos calculamos sus centros geométricos que harán la función de índices en la estructura de indexación. Segundo, la etapa de indexación, donde creamos la estructura, asociando los códigos cilíndricos de minucias de cada impresión de las bases de datos con el índice más cercano. Tercero, la etapa de recuperación; en esta etapa realizamos las consultas de las huellas latentes, donde para cada uno de sus descriptores se calcula a qué índice corresponden. De estos índices obtenemos listas de descriptores asociados, que pueden ser ampliadas o no, en dependencia de un umbral definido que tiene el objetivo de homogenizar la cantidad de descriptores extraídos. Y, por último, la etapa de integración de votos; aquí se combinan los votos por impresión de tal forma que permita crear una lista de impresiones candidatas que maximice la probabilidad de contener la huella correspondiente a la huella consulta.

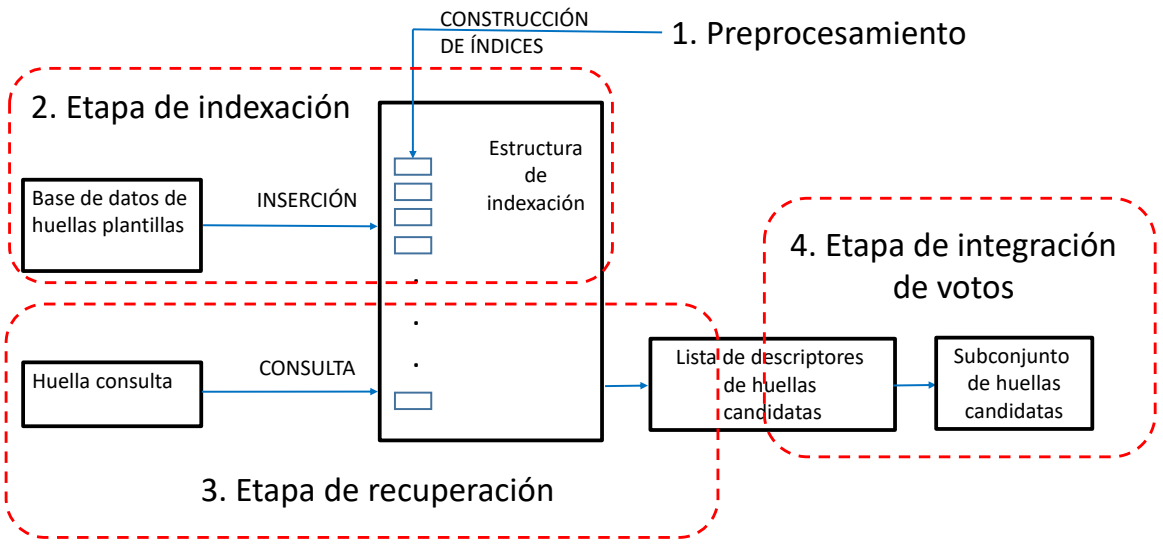


FIGURA 3.1: Esquema del algoritmo de indexación propuesto con cada etapa enumerada.

A continuación se encuentran detalladamente las diferentes etapas la figura 3.1, así como sus pasos previos.

3.1. Preprocesamiento

En la primera etapa de esta propuesta se procesan los códigos cilíndricos de minucias de las impresiones de varias bases de datos (figura 3.2), con el objetivo de alcanzar una asignación de los descriptores en la estructura de indexación, lo más homogénea posible. Para ello se agrupan dichos descriptores en k grupos, obteniéndose como resultado de interés, los centros de los grupos y no los grupos en sí. Donde el j-ésimo centro de grupo se define de la siguiente forma: $c_j = \langle v_{j1}, v_{j2}, ..., v_{jl} \rangle$; $v_{jl} \in [0, 1]$, donde l es la dimensión de los códigos cilíndricos y depende de la forma en la que se construyen. En nuestro caso particular, $l = 1280$. Una vez calculados los centros de grupos estos se convierten en los índices de la estructura de indexación.

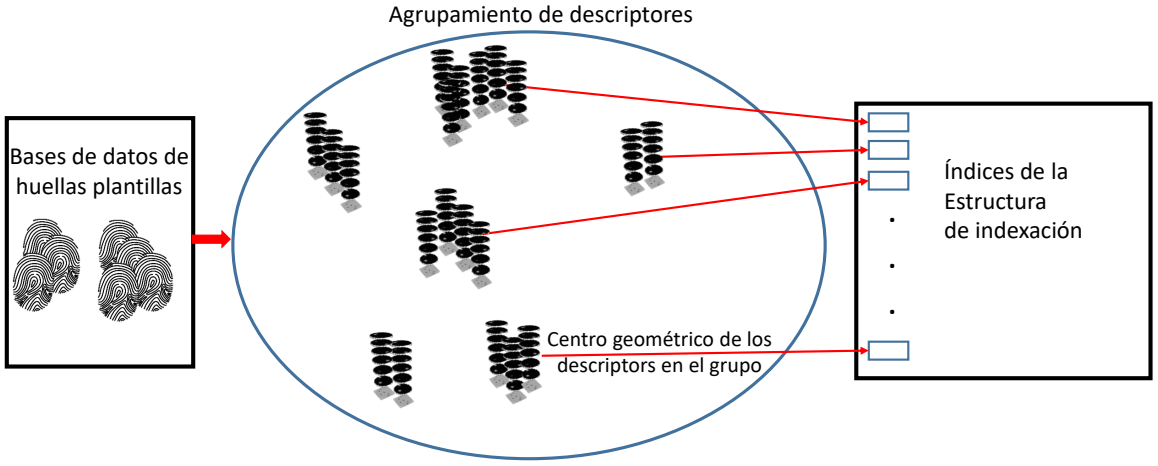


FIGURA 3.2: Esquema del preprocesamiento aplicado a bases de datos de impresiones para la construcción de los índices de la estructura de indexación.

El uso posterior de los centros de grupos se justifica bajo el supuesto de que las bases de datos utilizadas son representativas del conjunto de huellas usado en todos nuestros experimentos. Por tanto, en la indexación todos estos centros se usarán para construir grupos y estimamos que seguirán la misma distribución de los códigos cilíndricos usados para obtener dichos centros.

3.2. Etapa de indexación

El segundo paso planteado en la figura 3.1 consiste en asignar los códigos cilíndricos de minucias de cada impresión a la entrada que le corresponde en una tabla de búsqueda (ver algoritmo 1). Para esto, por cada código cilíndrico de minucia cc se calcula la distancia euclideana con cada centro de grupo c_j, obtenido previamente, y se selecciona el centro de grupo más cercano. Luego asociamos a este centro de grupo una lista de pares formadas por el código cilíndrico en cuestión y el índice de la impresión a la que pertenece.

El uso de la función de distancia euclidiana para realizar la asignación de un centro de grupo a un código cilíndrico se determinó de manera empírica. Se experimentó con un conjunto de funciones de disimilitud, obteniendo los mejores resultados (Capítulo 4) con la función de distancia escogida para el algoritmo presentado.

Una vez creada la estructura de indexación H, podemos proseguir con la obtención de la lista de impresiones candidatas para identificar una huella latente, comenzando con la recuperación de impresiones (epígrafe 3.3).

Algoritmo 1: Indexación de impresiones.

function `construirEstructuraIndexación`(T, C)
Data:

- $T = \{T_i : i \in [1, n]\}$, donde T_i es el conjunto de códigos cilíndricos de minucias de la i-ésima impresión y n el número de impresiones en T.
- $C = [c_1, c_2, ..., c_k]$, donde c_j es el j-ésimo centro de grupo de los códigos cilíndricos de minucías y k el número de grupos.
- Sea d la función de distancia euclideana.

Result: Estructura de indexación H
begin
  $H \leftarrow \{\}$
  foreach $T_i \in T$ **do**
    foreach *código cilíndrico* $cc \in T_i$ **do**
      $c \leftarrow argmin_{c_j \in C}\ d(cc, c_j)$
      $H \leftarrow H \cup \{\langle c, \langle cc, i\rangle\rangle\}$
    end
  end
  return H
end

3.3. Etapa de recuperación de impresiones

La etapa de recuperación tiene el objetivo de obtener un subconjunto de impresiones que facilite el reconocimiento de la huella consulta en un menor tiempo. Para ello debe maximizar la probabilidad de tener entre sus impresiones la huella correspondiente.

La cantidad de impresiones obtenidas, a menos que los descriptores indexados se distribuyan de forma homogénea, suele variar en dependencia de la huella consulta. En el algoritmo 2, propuesto a continuación, establecemos una salida homogénea en cada consulta ($u = a/k$), donde a es la cantidad de descriptores indexados y k el número de grupos creados. De esta forma mejoramos en la búsqueda de la impresión correcta en aquellos casos que el grupo correspondiente contenga menos códigos cilíndricos que los definidos por el umbral u.

El algoritmo en esta etapa primero ordena los centros de grupos atendiendo a la distancia de estos con el código cílindrico usado como consulta. Luego se seleccionan los pares $\langle cc, i\rangle$ asociados a los centros de grupos más cercanos, hasta lograr una cantidad mayor o igual que u. Con cada par obtenido de H se forma una tupla de salida con el grado de similitud entre los códigos cilíndrico cc y qcc, más el índice de la huella a la que pertenece cc. La similitud es calculada mediante la fórmula diseñada en Cappelli et al. [14].

Algoritmo 2: Recuperación de impresiones.

function `recuperarImpresiones`(qcc, C, H)
Data:

- qcc, código cilíndrico de consulta.
- $C = [c_1, c_2, ..., c_k]$, donde c_j es el j-ésimo centro de grupo de los códigos cilíndricos de minucías y k el número de grupos.
- H, estructura de indexación creada en el algoritmo 1.
- Sea d la función de distancia euclideana.
- Sea s la función de similitud local entre códigos cilíndricos definida en Cappelli et al. [14].

Result: L_{qcc}, conjunto de pares $\langle s, i\rangle$, donde s es el grado de similitud de qcc con un código cilíndrico de minucia asociado a la i-ésima huella plantilla indexada en H.

```
begin
    Sea u ← a/k el umbral para cada búsqueda, donde a es el número total de códigos
      cilíndricos indexados en H.
    D ← lista de los centros de grupos (c_j ∈ C) ordenados ascendentemente según el valor
      retornado por d(qcc, c_j)
    indice ← 1
    L_qcc ← {}
    repeat
        foreach {⟨cc, i⟩ : ⟨c_indice, ⟨cc, i⟩⟩ ∈ H ∧ c_indice ∈ D} do
            L_qcc ← L_qcc ∪ {⟨s(qcc, c_j), i⟩}
        end
        indice ← indice + 1
    until ||L|| >= u
    return L_qcc
end
```

El algoritmo 2 se lleva a cabo con cada código cilíndrico de la huella latente y finalmente se retorna una lista por cada descriptor.

El algoritmo desarrollado por Capelli et. al. [16] realiza las comparaciones locales entre minucias teniendo en cuenta rotaciones de hasta 45 grados, por lo que la función de similitud local s realiza sus cálculos de igual manera para lograr una comparación en igualdad de condiciones.

3.4. Etapa de integración de votos

En la sección anterior, el algoritmo 2 obtiene una lista de pares $\langle s, i\rangle$ por cada código cilíndrico de la huella consulta. Cada par se forma por la impresión i-ésima de la base de datos y el grado de similitud local obtenido con uno de sus códigos cilíndricos y el código cilíndrico de la huella

consulta en cuestión. En el contexto de nuestro trabajo llamaremos a dicho par, 'voto', ya que es analizada como un voto a favor de la impresión relacionada. Como es de notar, no existe ninguna restricción en el algoritmo 2 para evitar que una impresión se repita, ya sea en una misma lista o entre varias.

Como resultado final de este esquema de solución es necesario retornar una lista de impresiones donde las mismas no estén repetidas, ya que carece de sentido. Para unificar las listas obtenidas previamente, desarrollamos a continuación una variante del conteo de Borda ([9]) como integración de votos. Esta variante se ajusta a la creación de la lista de impresiones candidatas (algoritmo 3) el cual también debe garantizar que la lista resultante sea insensible a la falta de información de las huellas latentes.

Algoritmo 3: Integración de votos.

function `integrarVotos`(L)
Data:

- $L = [L_1, ..., L_m]$, donde m es la cantidad de códigos cilíndricos en la huella consulta, $L_c = \{\langle s, i\rangle_t, t \in [1, p_c]\}$ y p_c la cantidad de impresiones recuperadas del código cilíndrico c-ésimo de la huella consulta.

Result: Lista I de impresiones candidatas.

begin
  $R \leftarrow \{\}$
  foreach $L_c \in L$ **do**
    Ordenar L_c por el grado de similitud de manera descendente.
    Sea P un conjunto, inicialmente vacío, para controlar la unicidad de los identificadores de las impresiones.
    Sea v un escalar, inicializado a 1, para establecer el orden de prioridad de los identificadores de las impresiones.
    foreach $\langle s, i\rangle \in L_c$ **do**
      if $i \notin P$ **then**
        $R \leftarrow R \cup \{\langle v, i\rangle\}$
        $v \leftarrow v + 1$
        $P \leftarrow P \cup \{i\}$
      end
    end
  end
  Sea n el número de impresiones indexadas.
  foreach $\langle v, i\rangle \in R$ **do**
    $c_i \leftarrow c_i + (n - v)$
  end
  Ordenar los identificadores de las impresiones por su valor c_i de manera descendente y almacenar el nuevo orden en I.
  return I
end

Las listas en $L = [L_1, ..., L_m]$ se ordenan de mayor a menor similitud, es decir, de menor a mayor según la distancia calculada entre los códigos cilíndricos. Por cada lista L_c se toma la ocurrencia de cada impresión una sola vez, formando una lista R de tuplas representadas por el orden en el que aparecen las impresiones en L_c y el índice de la impresión en cuestión. Iterando por R, se acumula por cada impresión el orden invertido en relación a todas las huellas. Finalmente, se ordenan las huellas según el valor acumulado, de mayor a menor, dando más peso a aquellas impresiones que se hayan encontrado más veces en las primeras posiciones de las listas de entrada a esta etapa del algoritmo.

3.5. Conclusiones

En el presente capítulo se desarrollaron los algoritmos correspondientes a las diferentes etapas del esquema de indexación propuesto. Con ellos se logra construir una estructura de indexación que asocie los códigos cilíndricos basados en su similitud, además de obtener resultados de tamaño homogéneo a cada consulta realizada para así lograr estabilizar el rendimiento y mejorar las posibles deficiencias en aquellas consultas que retornen listas con tamaño inferior al umbral definido. También se desarrolla una variante del conteo de Borda como mecanismo de integración de votos a partir de listas de impresiones obtenidas por cada código cilíndrico de la huella consulta. Estos resultados son corroborados en el siguiente capítulo mediante resultados experimentales.

Capítulo 4

Protocolo de evaluación y resultados experimentales

En este capítulo analizamos los resultados del algoritmo de indexación de huellas dactilares propuesto, tanto de tasa de error, como de eficiencia. Mostramos el desempeño del algoritmo en más de un millón de huellas, para ello utilizamos la base de datos NIST SD27 [22] y dos bases de datos adicionales; una base de datos propietaria y otra de naturaleza sintética. Nuestra propuesta es evaluada contra los resultados del algoritmo en Cappelli et al. [16]. En la literatura encontramos varias soluciones posteriores, tanto para el problema de indexación de impresiones, como de huellas latentes, pero en todas se combinan varios descriptores. Como se argumentó en el epígrafe 1.1, los trabajos más recientes incluyen entre sus descriptores los códigos cilíndricos de minucias mediante el MCC SDK, de ahí nuestra selección del algoritmo a comparar. Además, ninguna otra implementación se encuentra disponible y no se proveen detalles suficientes para realizar una implementación fidedigna. Adicionalmente, tampoco fue atendida nuestra petición a los autores para reproducir sus experimentos en las bases de datos escogidas en esta investigación y los resultados reportados, al menos en indexación de huellas latentes, no usaban el protocolo estandar de evaluación para indexación, sino el de identificación.

4.1. Bases de datos

Las diferentes etapas del algoritmo usan bases de datos de diversas fuentes. La primera fuente es el Instituto Nacional de Estándar y Tecnología (NIST [51] por sus siglas en inglés) de los

Estados Unidos de América, al cual pertenecen las siguientes bases de datos:

- SD4: Contiene 4,000 impresiones rodadas (2,000 pares) con un tamaño de 480x512 píxeles y una resolución de 500ppp [66].
- SD14: Contiene 54,000 impresiones rodadas (27,000 pares) con un tamaño de 832x768 píxeles y una resolución de 500ppp [67].
- SD27: Contiene 258 impresiones y 258 huellas dactilares latentes emparejadas [22]. Las dimensiones son de 800x768 y una resolución también de 500ppp.

Nótese que cada par de huellas en las bases de datos corresponde a dos instancias del mismo dedo.

También utilizamos una base de datos propietaria de 106,921 impresiones y 284 huellas latentes para llevar a cabo el entrenamiento del algoritmo. Por último, incluimos una base de datos sintética generada con el programa SFinGe versión 4.1 (build 1746) Demo [44]. Esta base de datos contiene 997,377 impresiones artificiales y se usa para aproximar el comportamiento del algoritmo en bases de datos mayores a un millón de huellas. Estas impresiones generadas artificialmente son pertinentes para evaluar los resultados de la investigación porque el software SFinGe se ha usado en competencias de verificación y los resultados han sido similares a los obtenidos con bases de datos reales [13].

4.2. Protocolo de evaluación PR vs ER

En esta investigación usamos el protocolo de evaluación para algoritmos de indexación documentado en el 'Handbook of Fingerprint Recognition' [44]. Este protocolo evalúa los algoritmos de indexación de huellas mediante el balance entre la tasa de error y la tasa de penetración en la base de datos (ER y PR por sus siglas en Inglés, respectivamente), figura 4.1 . El término ER se define como el porcentage de las huellas no encontradas en relación al total de huellas buscadas y PR como la porción de la base de datos que se desea retornar en promedio. El balance ER/PR se calcula para todos los valores de $max_{PR} \in [1, 100]$. Cada valor de max_{PR} está asociado a un número máximo de huellas candidatas $max_C = max_{PR} * n/100$, donde n es el número total de las huellas en la base de datos. En caso de que la lista de candidatos sea mayor que max_C, solo

serán considerados los primeros max_C candidatos para calcular los valores correspondientes de PR y ER. La descripción detallada de este procedimiento se encuentra en [7].

Los algoritmos de indexación, basados en sus estrategias internas, pueden retornar listas de candidatos de diferentes tamaños en dependencia de la huella consulta. A su vez, estas listas producen valores de PR variables y con el objetivo de reducir el espacio de búsqueda, un primer enfoque es retornar listas de candidatos de menor tamaño. Pero este enfoque podría tener el inconveniente de producir más errores, por lo que debe seleccionarse cuidadosamente [15, 16].

La compararación de algoritmos de indexación mediante este protocolo tiene diferentes maneras de interpretarse. El mejor caso de análisis es cuando la curva de un algoritmo domina al resto de las curvas en toda la extensión del rayo numérico, ver curva que representa el algoritmo B en la figura 4.1. Esto significa que para cualquier valor de PR (eje de las abcisas), dicha curva tiene un valor menor de ER (eje de las ordenadas) que las curvas del resto de los algoritmos; por tanto, se ubica más próxima al eje horizontal en la figura.

El caso ideal se presenta con la linea verde oscuro, indicando el algoritmo A, el cual no comete errores para ningún porcentage de la base de datos. Mientras el peor caso sería la linea roja, algoritmo D, donde el error siempre es el máximo, excepto para el 100 % de PR, en cuyo caso ya el algoritmo deja de cumplir su objetivo de reducir el espacio de búsqueda. Un algoritmo de indexación que presente este comportamiento nunca lograría devolver la huella que corresponde. En otros casos se pueden obtener curvas que se intersecten, para las cuales lo importante es evaluar qué PR interesa, ya que no se puede determinar un algoritmo mejor a otro en su totalidad. También se encuentran curvas incompletas (ver ejemplos en los artículos [16, 21, 30]) y en estos casos se analizan sus comportamientos en el rango en el que todas estén definidas.

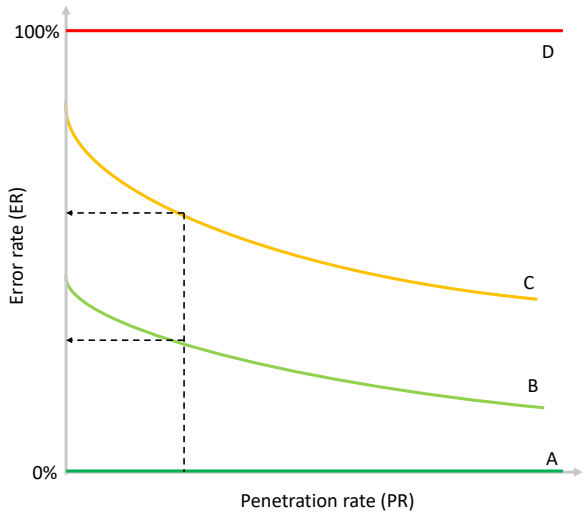


FIGURA 4.1: Protocolo de evaluación de algoritmos de indexación [44]. Las métricas utilizadas son tasa de error y porcentaje de la base de datos; Error rate y Penetration rate en la literatura en inglés, respectivamente. Con letras de la A a la D representan cuatro hipotéticos algoritmos, siendo A el ideal y D, el peor. Las líneas discontinuas indican como evaluar los algoritmos B y C dado un valor dado de porcentaje de la base de datos, siendo B el mejor por presentar menos error.

4.3. Análisis de resultados experimentales

4.3.1. Resultados del preprocesamiento

En el preprocesamiento agrupamos las impresiones de las bases de datos del NIST (SD4, SD14 y SD27) y la base de datos propietaria. En los casos de las bases de datos SD4 y SD14, todas las huellas fueron incluidas en el agrupamiento debido a que son impresiones.

Para agrupar los códigos cilíndricos usamos la implementación de k-means++ [1] de Weka [24] con parámetros que facilitaran la reducción de dos o más órdenes de magnitud en la etapa de recuperación. De esta forma seleccionamos la cantidad de grupos ($k = 100$), asumiendo que su comportamiento, en la etapa de indexación, cumpla con la consideración hecha sobre la distribución de las bases de datos escogidas. Se realizaron varios experimentos con diferentes valores de k; para valores mayores que 100 los resultados de los experimentos empeoraron y en los menores no se llevaron a cabo, puesto que no cumplen con la reducción mínima de dos órdenes de magnitud. En los experimentos realizados obtuvimos que uno de los centros acumulaba más de tres veces la cantidad promedio de códigos cilíndricos de minucias. Esta distribución tiene consecuencias negativas en el rendimiento de nuestro algoritmo de recuperación debido a que es mucho mayor la cantidad de descriptores a filtrar para lograr el umbral deseado. Nótese que

no solo afectaría las consultas que seleccionen dicho grupo, sino también aquellos grupos que no contengan la cantidad esperada y lo tengan en su lista de vecinos de donde completar la cantidad de descriptores. Por esta razón decidimos distribuir los descriptores asignados a ese grupo entre los demás centros de grupos bajo las mismas condiciones con que fueron creados los grupos, resultando ahora 99 grupos. La figura 4.2 muestra el histograma de la distribución de los códigos cilíndricos en sus respectivos grupos. Estos datos presentan un promedio de 143,248.16 descriptores por grupo y una desviación estándar aproximadamente de 61,902.92. Si bien es una desviación alta, el número de picos se equipara aproximadamente con las depresiones, por lo que el algoritmo de recuperación no demora en encontrar un grupo con el cual balancear su salida cuando el código cilíndrico consulta le corresponda a un grupo con pocos descriptores.

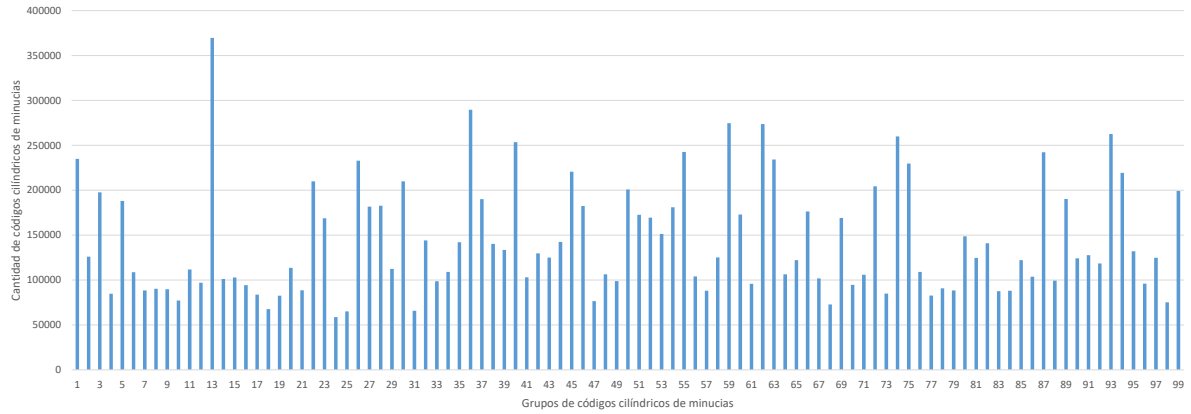


FIGURA 4.2: Histograma de frecuencia de código cilíndrico de minucias por grupos.

4.3.2. Resultados de pruebas

Las pruebas que realizamos se llevaron a cabo en diferentes bases de datos. La base de datos propietaria se usó con el objetivo de optimizar el algoritmo, mientras que la base de datos NIST SD27 aumentada con la base de datos sintética se usó para comprobar el comportamiento en condiciones parecidas a un entorno real (más de un millón de huellas).

4.3.2.1. Resultados en la base de datos propietaria

En los experimentos con la base de datos propietaria se usaron sus huellas latentes como huellas consulta y sus impresiones como huellas plantilla. En estos obtuvimos una distribución de los códigos cilíndricos de minucias por grupos parecidos al obtenido en el preprocesamiento. En la figura 4.3.a mostramos como el experimento con la base de datos propietaria, al ser un

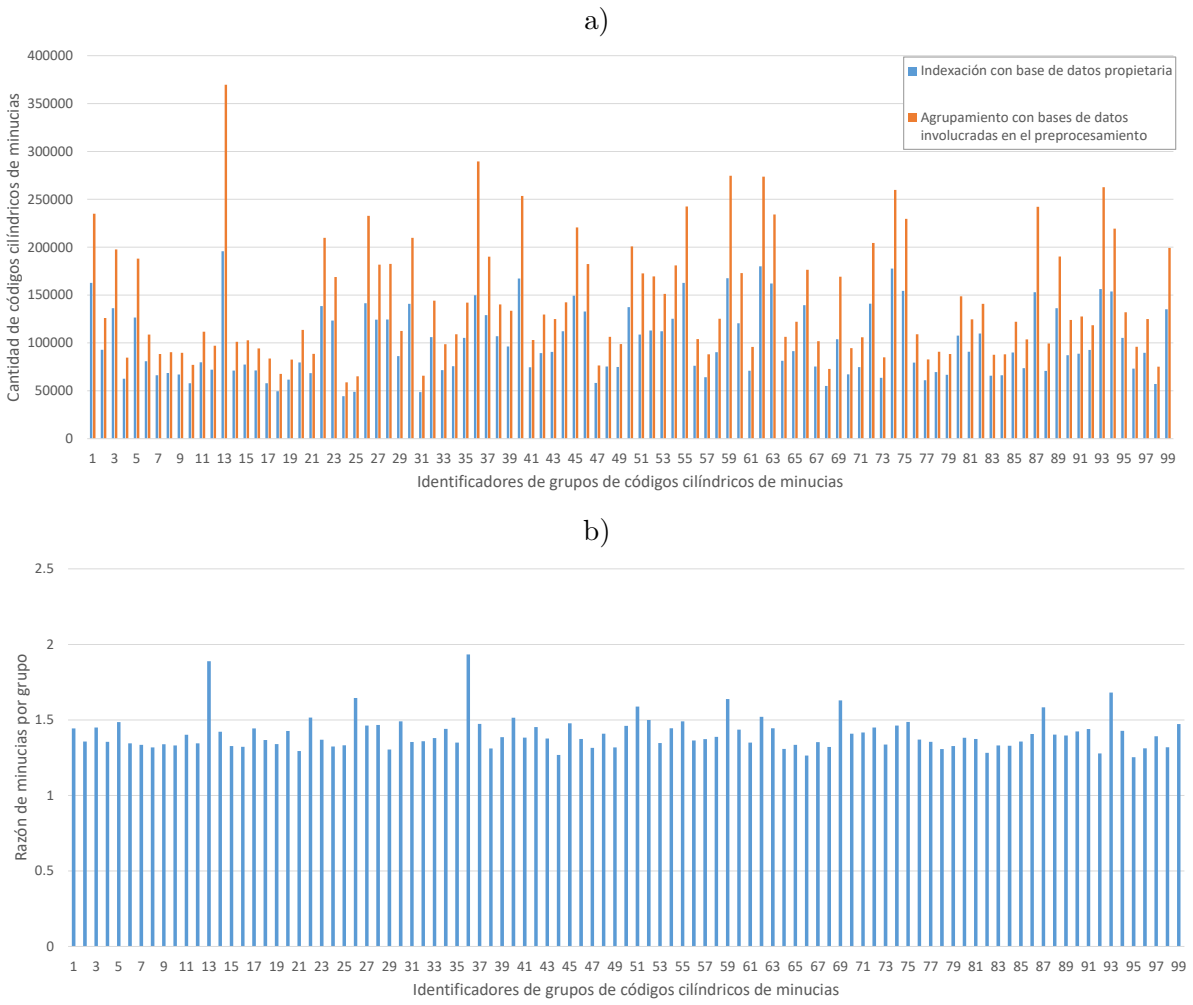


FIGURA 4.3: Histogramas del comportamiento de la distribución de códigos cilíndricos en el preprocesamiento y en la indexación de la base de datos propietaria; a) cantidades por grupo de ambos conjuntos de bases de datos; b) razón entre las cantidades.

subconjunto del preprocesamiento, mantiene un nivel inferior en las cantidades de descriptores por grupos y en la figura 4.3.b presentamos la proporción entre ambas cantidades por grupos; gráficamente se nota que la mayoría de los grupos mantienen la proporción y se corrobora con una desviación estándar de solo 0.11. En el caso de las cantidades de descriptores por grupos obtuvimos una desviación estándar de 36,217.16, lo cual indica que en relación al preprocesamiento (61,902.92), la diferencia de códigos cilíndricos entre grupos disminuyó y se encuentran mejor distribuidos. Esto se debe a que el conjunto de prueba es un subconjunto del usado para generar los grupos y las bases de datos adicionales siguen también aproximadamente la misma distribución.

Los resultados en el protocolo de evaluación (figura 4.4) muestran cómo el algoritmo propuesto (MCCClustering) mejora la eficacia del algoritmo base de comparación (Capelli et al, [16]) de

manera consistente hasta para un 10 % de uso de la base de datos.

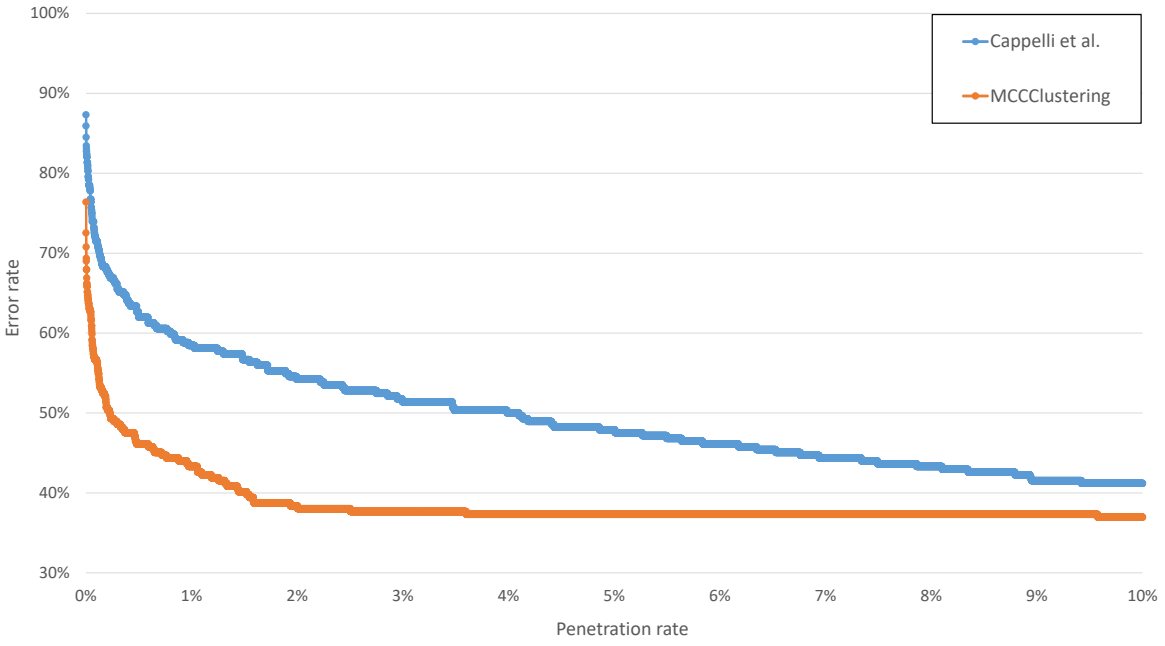


FIGURA 4.4: Protocolo de evaluación de los algoritmos en la base de datos propietaria hasta un 10 % de la misma.

En este protocolo a menor porciento de uso de la base de datos, menor el tiempo para procesar la lista de huellas candidatas retornadas por el algoritmo, por lo que los resultados serán analizados en los percentiles 0.01, 0.1, 0.5 y 1 % como se muestra en la tabla 4.1. La taza de error se obtuvo en los valores superiores más próximos a dichos percentiles debido a que a ese nivel de precisión numérica no se obtuvieron exactamente los percentiles mencionados.

TABLA 4.1: Tasa de error de los algoritmos en los percentiles claves en la base de datos propietaria.

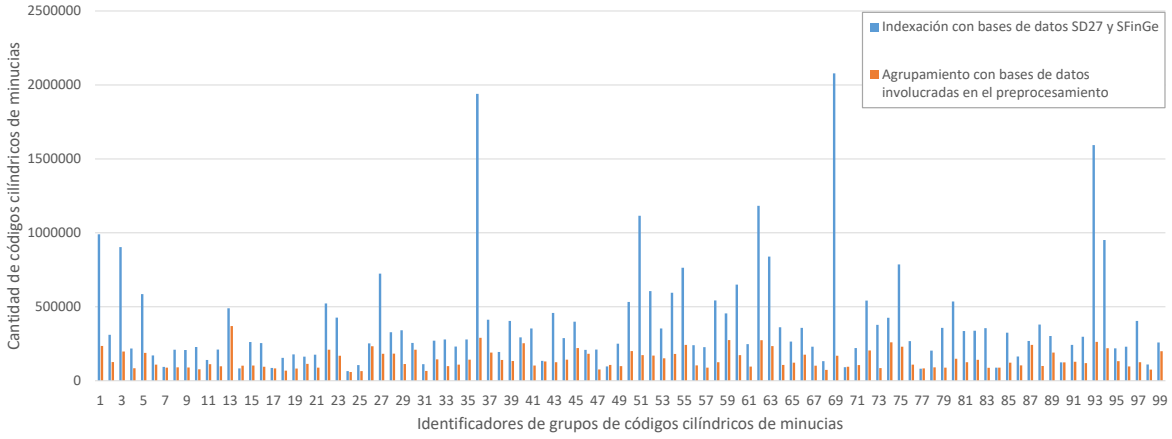
	Percentiles			
Algoritmo	**0.01 %**	**0.1 %**	**0.5 %**	**1 %**
Cappelli et al. [16]	82.0 %	71.5 %	61.9 %	58.4 %
MCCClustering	65.8 %	56.7 %	46.1 %	43.3 %

4.3.2.2. Resultados en la base de datos NIST SD27

En el actual experimento usamos como huellas consulta las huellas latentes de la base de datos NIST SD27 y como huellas plantilla las impresiones de la misma, la base de datos propietaria y

las huellas de la base de datos sintética. En este caso obtuvimos un histograma (figura 4.5.a) que difiere en la distribución de códigos cilíndricos por grupos con respecto a los resultados obtenidos en el agrupamiento del preprocesamiento. En la figura 4.5.b se enfatiza, mediante la proporción de las agrupaciones de este experimento y del preprocesamiento, el carácter de las asignaciones cuando son usadas los códigos cilíndricos de la base de datos sintética. Esto indica que dichas huellas sintéticas no siguen la distribución de las bases de datos usadas originalmente para crear los grupos y brinda la oportunidad para experimentar en un contexto más heterogeneo.

a)



b)

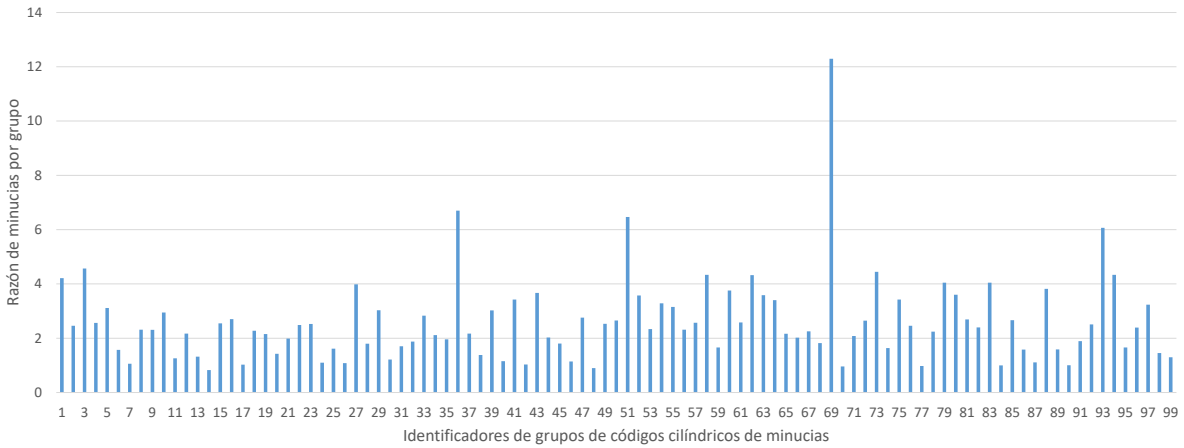


FIGURA 4.5: Histogramas del comportamiento de la distribución de códigos cilíndricos en el preprocesamiento y en la indexación de las bases de datos: NIST SD27, propietaria y sintética; a) cantidades por grupo de ambos conjuntos de bases de datos; b) razón entre las cantidades.

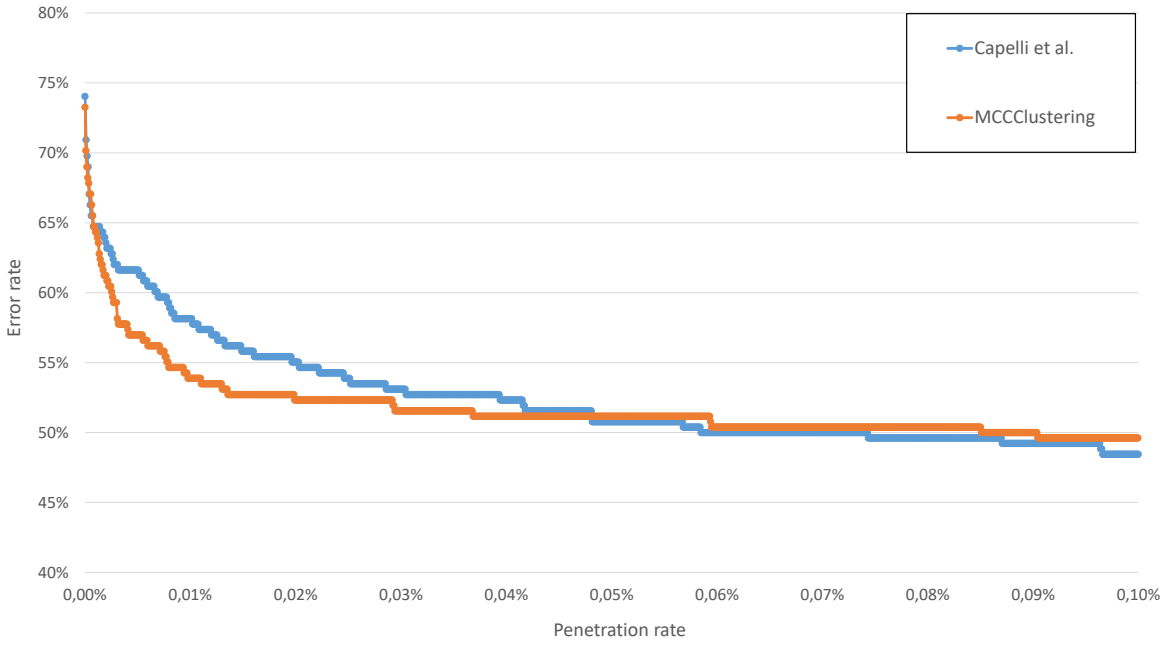


FIGURA 4.6: Protocolo de evaluación de los algoritmos en las bases de datos NIST SD27, propietaria y sintética.

En los resultados del protocolo de evaluación obtuvimos un desempeño de nuestro algoritmo favorable hasta un 0.043 % de uso de la base de datos (figura 4.6). El comportamiento general en los percentiles claves usados para comparar ambos algoritmos se muestra en la tabla 4.2.

TABLA 4.2: Tasa de error de los algoritmos en los percentiles claves para las bases de datos NIST SD27, propietaria y sintética.

	Percentiles			
Algoritmo	**0.01 %**	**0.1 %**	**0.5 %**	**1 %**
Cappelli et al. [16]	58.1 %	48.4 %	39.9 %	32.2 %
MCCClustering	53.9 %	49.6 %	42.6 %	39.5 %

En otras palabras, nuestro algoritmo obtiene mejores resultados al retornar hasta 475 huellas candidatas (0.043 %) de un total de 1,104,556 que conforman las tres bases de datos de este experimento. Esta cantidad ya es un número suficientemente grande a tener en cuenta por un perito para determinar la pertenencia o no de una huella a una persona, por lo que el resultado del algoritmo es positivo al no ser necesario un porcentaje mayor de la base de datos.

4.3.3. Análisis temporal

Los experimentos realizados persiguen como objetivo secundario obtener los resultados en un tiempo menor o igual a un orden de magnitud que el tiempo reportado por la implementación disponible de Cappelli et al. [16]. El tiempo a medir es el tiempo promedio de búsqueda de los descriptores de minucias para conformar la lista de huellas candidatas por cada huella consulta. En la tabla 4.3 se muestra el comportamiento respecto a este tiempo.

TABLA 4.3: Tiempo promedio de búsqueda de los códigos cilíndricos de minucias en la estructura de indexación, medidos en milisegundos. Los resultados se muestran para los dos experimentos.

	Experimento	
Algoritmo	propietaria	sd27+propietaria+sintética
Cappelli et al.	627	2,626
MCCClustering	4,483	23,218
Proporción en tiempo de ejecución	7.14	8.84

En ambos casos la proporción de los tiempos es menor al umbral establecido, siendo satisfactorio el objetivo propuesto.

4.4. Conclusiones

La distribución de los códigos cilíndricos en las agrupaciones del preprocesamiento no son homogéneas, por lo que se justifica una etapa de recuperación de estos descriptores que realice un balance de la cantidad de los mismos.

Las diferentes bases de datos utilizadas en el preprocesamiento aportan cantidades proporcionales a los grupos generados, al ser de diversas fuentes, se explica en parte la representatividad de estas con respecto al universo de huellas y en consecuencia la utilidad de una indexación basada en agrupamientos para ser usada con cualquier otro conjunto de bases de datos.

Según la distribución de los códigos cilíndricos de las huellas sintéticas, estas no captan la naturaleza de las huellas reales al introducir una variabilidad muy alta en su distribución.

Nuestros algoritmos logran mejores resultados de eficacia para el reconocimiento de huellas latentes en porcentajes de bases de datos significativos para los peritos, tanto en conjuntos de huellas reales, como sintéticas.

Por último, según la literatura consultada hasta la fecha de culminación de este trabajo (junio del 2019), nuestras investigaciones son las primeras en usar bases de datos que superen el millón de huellas.

Capítulo 5

Conclusiones

Los algoritmos de indexación basados en minucias presentan menor tasa de error a medida que el número de minucias aumenta en los descriptores usados, pero a expensas de la eficiencia de los mismos. No obstante, reportan los menores errores, cuando se usa únicamente un solo descriptor para el problema de indexación de impresiones.

En el caso del problema de indexación de huellas latentes dactilares, existe consenso general en la literatura en la necesidad de usar más de un descriptor para poder lograr el menor error posible. El resultado final en estos algoritmos integra los resultados de la indexación por cada descriptor usado. Por tanto, el propósito de este trabajo se enfoca en mejorar la tasa de error de indexación para el descriptor de códigos cilíndricos de minucias, el cual es muy utilizado en la literatura.

Los algoritmos que usan códigos cilíndricos de minucias mejoran su rendimiento en comparación a los demás descriptores de minucias, debido a su representación vectorial de longitud fija, lo que posibilita realizar comparaciones en instrucciones de bajo nivel del procesador. También reportan menor tasa de error puesto que involucra información de las minucias existentes en un radio determinado, volviéndolo más versátil que otros descriptores en cuanto a tolerancia a problemas de traslación, rotación y distorsión en huellas latentes dactilares.

En esta investigación se propuso un algoritmo de indexación basado en códigos cilíndricos de minucias que agrupa estos descriptores usando los centros geométricos de grupos construidos previamente con bases de datos de entrenamiento. Estos grupos no generan distribuciones homogéneas por lo que se propuso un algoritmo de recuperación que retorne una cantidad uniforme en cada consulta, logrando de esta forma homogenizar el rendimiento por cada código cilíndrico

de minucia y disminuir los errores en los casos de descriptores que no alcancen el umbral definido. Finalmente, se implementó una versión del algoritmo de conteo de Borda para integrar los votos de cada código cilíndrico de minucia de una misma huella latente.

El algoritmo propuesto para la indexación de huellas latentes dactilares se probó en experimentos con diferentes bases de datos, entre ellas una de más de un millón de huellas sintéticas. Según nuestro análisis de la literatura hasta el momento (junio de 2019), ninguna investigación se ha reportado en semejante escenario para el problema de indexación, tanto de impresiones, como de huellas latentes.

El algoritmo propuesto logra menor tasa de error, en pequeños porcentajes de las bases de datos, que el MCC SDK (único algoritmo del que existe disponible una implementación para experimentar). Además, estos resultados fueron obtenidos tantos para bases de datos reales, como sintéticas. También, los tiempos promedios de búsqueda por huella en los experimentos se logran en mayor tiempo que el algoritmo reportado por el MCC SDK, pero sin superar un orden de magnitud.

Capítulo 6

Trabajos futuros

Existen varias potencialidades derivadas de esta investigación. La primera tarea de interés es combinar nuestra propuesta de investigación con otros indexadores que usen diferentes descriptores, tal como se desarrolla en la literatura consultada ([20, 55]). Obteniendo de esta forma un indexador que integre los resultados de diferentes descriptores provenientes, potencialmente, de diversos rasgos. Esto con el objetivo de lograr mejores resultados de tasa de error en las combinaciones de descriptores reportadas que usen el SDK provisto por Cappelli et al. [16].

Analizar diferentes criterios de desempeño de los algoritmos de indexación de huellas latentes. Entre ellos, tasa de error cometido contra el tamaño de las bases de datos a indexar y el tamaño de los grupos asociados a las llaves de la estructura de indexación.

Como caso particular del último criterio de desempeño mencionado, se propone investigar la generación de distribuciones homogéneas que logren mantener o superar los niveles de tasa de error obtenidos en esta investigación. En caso de no lograrlo, sería útil analizar el comportamiento de la tasa de error contra el tamaño de los grupos, ahora sí todos iguales. Nótese que para tamaños de grupos menores al umbral definido en nuestra investigación es de esperar que exista una mejoría en el tiempo promedio de búsqueda.

Finalmente, combinar nuestra propuesta de indexación o una combinación de indexadores, incluyendo el nuestro, con un algoritmo de identificación [35, 36] . En este caso la lista de huellas candidata obtenida por el indexador se convierte en la entrada al algoritmo de identificación. En este tipo de trabajo es de interés evaluar las métricas de porcentaje de base de datos retornadas por el algoritmo de indexación contra el error cometido por el algoritmo de identificación para diversas posiciones del ranking.

Bibliografía

[1] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[2] Chaochao Bai, Tong Zhao, Weiqiang Wang, and Min Wu. An efficient indexing scheme based on k-plet representation for fingerprint database. In *International Conference on Intelligent Computing*, pages 247–257. Springer, 2015.

[3] Chao-chao Bai, Wei-qiang Wang, Tong Zhao, Ru-xin Wang, and Ming-qiang Li. Deep learning compact binary codes for fingerprint indexing. *Frontiers of Information Technology & Electronic Engineering*, 19(9):1112–1123, 2018.

[4] Chaochao Bai, Weiqiang Wang, Tong Zhao, and Mingqiang Li. Fast exact fingerprint indexing based on compact binary minutia cylinder codes. *Neurocomputing*, 275:1711–1724, 2018.

[5] George Bebis, Taisa Deaconu, and Michael Georgiopoulos. Fingerprint identification using delaunay triangulation. In *International Conference on Information Intelligence and Systems*, pages 452–459. IEEE, 1999.

[6] Bir Bhanu and Xuejun Tan. Fingerprint indexing based on novel features of minutiae triplets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):616–622, 2003.

[7] Biometric System Laboratory . Benchmark area: Fingerprint indexing. https://biolab.csr.unibo.it/fvcongoing/UI/Form/BenchmarkAreas/BenchmarkAreaFIDX.aspx, 2019. Accessed: 2019-03-05.

[8] Soma Biswas, Nalini K. Ratha, Gaurav Aggarwal, and Jonathan Connell. Exploring ridge curvature for fingerprint indexing. In *2nd IEEE International Conference on Biometrics: Theory, Applications and Systems*, pages 1–6. IEEE, 2008.

[9] Duncan Black. Partial justification of the borda count. *Public Choice*, 28(1):1–15, Dec 1976.

[10] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc., 2008.

[11] Kai Cao and Anil K. Jain. Automated latent fingerprint recognition. *IEEE transactions on pattern analysis and machine intelligence*, 41(4):788–800, 2019.

[12] Kai Cao, Eryun Liu, and Anil K. Jain. Segmentation and enhancement of latent fingerprints: A coarse to fine ridge structure dictionary. *IEEE transactions on pattern analysis and machine intelligence*, 36(9):1847–1859, 2014.

[13] Raffaele Cappelli, Dario Maio, Davide Maltoni, James L. Wayman, and Anil K. Jain. Performance evaluation of fingerprint verification systems. *IEEE transactions on pattern analysis and machine intelligence*, 28(1):3–18, 2006.

[14] Raffaele Cappelli, Matteo Ferrara, and Davide Maltoni. Minutia cylinder-code: A new representation and matching technique for fingerprint recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(12):2128–2141, 2010.

[15] Raffaele Cappelli, Matteo Ferrara, and Dario Maio. Candidate list reduction based on the analysis of fingerprint indexing scores. *IEEE Transactions on Information Forensics and Security*, 6(3):1160–1164, 2011.

[16] Raffaele Cappelli, Matteo Ferrara, and Davide Maltoni. Fingerprint indexing based on minutia cylinder-code. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):1051–1057, 2011.

[17] Sharat Chikkerur, Alexander N Cartwright, and Venu Govindaraju. K-plet and coupled bfs: a graph based fingerprint representation and matching algorithm. In *International Conference on Biometrics*, pages 309–315. Springer, 2006.

[18] Kyoungtaek Choi, Dongjae Lee, Sanghoon Lee, and Jaihie Kim. An improved fingerprint indexing algorithm based on the triplet approach. In *5th International Conference on Audio- and Video-Based Biometric Person Authentication*, pages 584–591. Springer, 2003.

[19] Mark De Berg, Otfried Cheong, Marc Van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition, 2008.

[20] Jianjiang Feng and Anil K. Jain. Filtering large fingerprint database for latent matching. In *19th International Conference on Pattern Recognition*, pages 1–4. IEEE, 2008.

[21] Andrés Gago-Alonso, José Hernández-Palancar, Ernesto Rodríguez-Reina, and Alfredo Muñoz-Briseño. Indexing and retrieving in fingerprint databases under structural distortions. *Expert Systems with Applications*, 40(8):2858–2871, 2013.

[22] Michael D. Garris and R. Michael McCabe. *NIST Special Database 27: Fingerprint minutiae from latent and matching tenprint images.* U.S. Department of Commerce, U.S. National Institute of Standards and Technology, 2000.

[23] Robert S. Germain, Andrea Califano, and Scott Colville. Fingerprint matching using transformation parameter clustering. *IEEE Computational Science and Engineering*, 4(4):42–49, 1997.

[24] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[25] Mabel Iglesias Ham, Yilian Bazán Pereira, and Edel B García Reyes. A multiple substructure matching algorithm for fingerprint verification. In *Iberoamerican Congress on Pattern Recognition*, pages 172–181. Springer, 2007.

[26] Jin-Hyuk Hong, Jun-Ki Min, Ung-Keun Cho, and Sung-Bae Cho. Fingerprint classification using one-vs-all support vector machines dynamically ordered with naïve bayes classifiers. *Pattern Recognition*, 41(2):662–671, 2008.

[27] Ogechukwu Iloanusi, Aglika Gyaourova, and Arun Ross. Indexing fingerprints using minutiae quadruplets. In *Workshop on Biometrics at Conference on Computer Vision and Pattern Recognition*, pages 127–133, Colorado Springs, USA, June 2011. IEEE.

[28] Ogechukwu Iloanusi. Fusion of finger types for fingerprint indexing using minutiae quadruplets. *Pattern Recognition Letters*, 38:8–14, 2014.

[29] Anil K Jain and Jianjiang Feng. Latent fingerprint matching. *IEEE Transactions on pattern analysis and machine intelligence*, 33(1):88–100, 2011.

[30] Umarani Jayaraman, Aman Kishore Gupta, and Phalguni Gupta. An efficient minutiae based geometric hashing for fingerprint database. *Neurocomputing*, 137:115–126, 2014.

[31] Tsai-Yang Jea. *Minutiae-based Partial Fingerprint Recognition*. PhD thesis, State University of New York at Buffalo, Buffalo, NY, USA, 2005. AAI3203911.

[32] Xudong Jiang, Manhua Liu, and Alex C. Kot. Fingerprint retrieval for identification. *IEEE Transactions on Information Forensics and Security*, 1(4):532–542, Dec 2006.

[33] Javad Khodadoust and Ali Mohammad Khodadoust. Fingerprint indexing based on minutiae pairs and convex core point. *Pattern Recognition*, 67:110–126, 2017.

[34] Donald Knuth. *The Art of Computer Programming: Fundamental algorithms.*, volume 1 of *Computer Science and Information Processing*. Addison-Wesley, 3rd edition, 1997.

[35] Ram P. Krish, Julian Fierrez, Daniel Ramos, Javier Ortega-Garcia, and Josef Bigun. Preregistration for improved latent fingerprint identification. In *22nd International Conference on Pattern Recognition (ICPR)*, pages 696–701. IEEE, 2014.

[36] Ram Prasad Krish, Julian Fierrez, Daniel Ramos, Javier Ortega-Garcia, and Josef Bigun. Pre-registration of latent fingerprints based on orientation field. *IET Biometrics*, 4(2):42–52, 2015.

[37] Terje Kristensen, Jostein Borthen, and Kristian Fyllingsnes. Comparison of neural network based fingerprint classification techniques. In *International Joint Conference on Neural Networks*, pages 1043–1048. IEEE, 2007.

[38] Jun Li, Wei-Yun Yau, and Han Wang. Combining singular points and orientation image information for fingerprint classification. *Pattern Recognition*, 41(1):353–366, 2008.

[39] Xuefeng Liang, Tetsuo Asano, and Arijit Bishnu. Distorted fingerprint indexing using minutia detail and delaunay triangle. In *3rd International Symposium on Voronoi Diagrams in Science and Engineering*, pages 217–223. IEEE, 2006.

[40] Xuefeng Liang, Arijit Bishnu, and Tetsuo Asano. A robust fingerprint indexing scheme using minutia neighborhood structure and low-order delaunay triangles. *IEEE Transactions on Information Forensics and Security*, 2(4):721–733, 2007.

[41] Edgar R. Lorch. The cauchy-schwarz inequality and self-adjoint spaces. *Annals of Mathematics*, 46(3):468–473, 1945.

[42] László Lovász, Katalin Vesztergombi, Uli Wagner, and Emo Welzl. Convex quadrilaterals and k-sets. *Contemporary Mathematics*, 342:139–148, 2004.

[43] Alessandra Lumini, Dario Maio, and Davide Maltoni. Continuous versus exclusive classification for fingerprint retrieval. *Pattern Recognition Letters*, 18(10):1027–1034, 1997.

[44] Davide Maltoni, Dario Maio, Anil K. Jain, and Salil Prabhakar. *Handbook of Fingerprint Recognition*. Springer Publishing Company Incorporated, 2nd edition, 2009.

[45] Davide Maltoni, Dario Maio, Anil K. Jain, and Salil Prabhakar. Introduction, fingerprint sensing and storage. In *Handbook of Fingerprint Recognition*, pages 36–38. Springer Publishing Company Incorporated, 2nd edition, 2009.

[46] Davide Maltoni, Dario Maio, Anil K. Jain, and Salil Prabhakar. Preface, overview. In *Handbook of Fingerprint Recognition*, page xi. Springer Publishing Company Incorporated, 2nd edition, 2009.

[47] Praveer Mansukhani, Sergey Tulyakov, and Venu Govindaraju. A framework for efficient fingerprint identification using a minutiae tree. *IEEE Systems Journal*, 4(2):126–137, 2010.

[48] Jun-Ki Min, Jin-Hyuk Hong, and Sung-Bae Cho. Effective fingerprint classification by localized models of support vector machines. In *International Conference on Biometrics*, pages 287–293. Springer, 2006.

[49] Alfredo Muñoz-Briseño, Andrés Gago-Alonso, and José Hernández-Palancar. Fingerprint indexing with bad quality areas. *Expert Systems with Applications*, 40(5):1839–1846, 2013.

[50] Karthik Nandakumar, Yi Chen, Sarat C. Dass, and Anil Jain. Likelihood ratio-based biometric score fusion. *IEEE transactions on pattern analysis and machine intelligence*, 30(2):342–347, 2008.

[51] National Institute of Standards and Technology. Main page. https://www.nist.gov/, 2019. Accessed: 2019-03-27.

[52] Michel Neuhaus and Horst Bunke. A graph matching based approach to fingerprint classification using directional variance. In *Audio-and Video-Based Biometric Person Authentication*, pages 455–501. Springer, 2005.

[53] Federal Bureau of Investigation. *The Science of Fingerprints: Classification and Uses*. U.S. Government Publication, Washington, DC, 1984.

[54] Alessandra A. Paulino, Jianjiang Feng, and Anil K. Jain. Latent fingerprint matching using descriptor-based hough transform. *IEEE Transactions on Information Forensics and Security*, 8(1):31–45, 2013.

[55] Alessandra A. Paulino, Eryun Liu, Kai Cao, and Anil K. Jain. Latent fingerprint indexing: Fusion of level 1 and level 2 features. In *Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, pages 1–8. IEEE, 2013.

[56] V. Ramasubramanian and K.K. Paliwal. A generalized optimization of the k-d tree for fast nearest-neighbour search. In *Fourth IEEE Region 10 International Conference TENCON*, pages 565–568. IEEE, 1989.

[57] Andrew Senior and Ruud Bolle. Fingerprint classification by decision fusion. *Automatic Fingerprint Recognition Systems*, pages 207–227, 2004.

[58] Yijing Su, Jianjiang Feng, and Jie Zhou. Fingerprint indexing with pose constraint. *Pattern Recognition*, 54:1–13, 2016.

[59] Xuejun Tan, Bir Bhanu, and Yingqiang Lin. Fingerprint classification based on learned features. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(3):287–300, 2005.

[60] Kamlesh Tiwari and Phalguni Gupta. Indexing fingerprint database with minutiae based coaxial gaussian track code and quantized lookup table. In *International Conference on Image Processing*, pages 4773–4777. IEEE, 2015.

[61] Tamer Uz, George Bebis, Ali Erol, and Salil Prabhakar. Minutiae-based template synthesis and matching for fingerprint authentication. *Computer Vision and Image Understanding*, 113(9):979–992, 2009.

[62] Lin Wang and Mo Dai. Application of a new type of singular points in fingerprint classification. *Pattern recognition letters*, 28(13):1640–1650, 2007.

[63] Song Wang and Jiankun Hu. Alignment-free cancelable fingerprint template design: A densely infinite-to-one mapping (ditom) approach. *Pattern Recognition*, 45(12):4129–4137, 2012.

[64] Xin Wang and Mei Xie. Fingerprint classification: An approach based on singularities and analysis of fingerprint structure. *Biometric Authentication*, pages 3–11, 2004.

[65] Yi Wang, Lipeng Wang, Yiu-Ming Cheung, and Pong C. Yuen. Learning compact binary codes for hash-based fingerprint indexing. *IEEE Transactions on Information Forensics and Security*, 10(8):1603–1616, 2015.

[66] Craig I. Watson and Charles L. Wilson. Nist special database 4. *Fingerprint Database, U.S. National Institute of Standards and Technology*, 17(77), 1992.

[67] Craig I. Watson. Nist special database 14. *Fingerprint Database, U.S. National Institute of Standards and Technology*, 1993.

[68] Eric W. Weisstein. Hash function – a wolfram web resource. http://mathworld.wolfram.com/HashFunction.html. (Accessed on 27/06/2019).

[69] Charles L. Wilson, Gerald T. Candela, and Craig I. Watson. Neural network fingerprint classification. *Journal of Artificial Neural Networks*, 1(2):203–228, 1994.

[70] Qinzhi Zhang and Hong Yan. Fingerprint classification based on extraction and analysis of singularities and pseudo ridges. *Pattern Recognition*, 37(11):2233–2243, 2004.

[71] Wei Zhou, Jiankun Hu, Song Wang, Ian Petersen, and Mohammed Bennamoun. Fingerprint indexing based on combination of novel minutiae triplet features. In *International Conference on Network and System Security*, pages 377–388. Springer, 2015.