Universidad de Granada

Departamento de Ciencias de la Computación
e Inteligencia Artificial

Programa de Doctorado en Tecnologías de la Información y la Comunicación

Minería de datos en computación de altas prestaciones para identificación en base a huellas dactilares

Tesis Doctoral

Daniel Peralta Cámara

Granada, Junio de 2016

Editor: Universidad de Granada. Tesis Doctorales
Autor: Daniel Peralta Cámara
ISBN: 978-84-9163-027-2
URI: http://hdl.handle.net/10481/44550

Universidad de Granada

Minería de datos en computación de altas prestaciones para identificación en base a huellas dactilares

MEMORIA PRESENTADA POR

Daniel Peralta Cámara

PARA OPTAR AL GRADO DE DOCTOR EN INFORMÁTICA

Junio de 2016

DIRECTORES

Francisco Herrera Triguero y José Manuel Benítez Sánchez

Departamento de Ciencias de la Computación
e Inteligencia Artificial

La memoria titulada "*Minería de datos en computación de altas prestaciones para identificación en base a huellas dactilares*", que presenta D. Daniel Peralta Cámara para optar al grado de doctor, ha sido realizada dentro del Programa Oficial de Doctorado en "*Tecnologías de la Información y la Comunicación*", en el Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada bajo la dirección de los doctores D. Francisco Herrera Triguero y D. José Manuel Benítez Sánchez.

El doctorando, D. Daniel Peralta Cámara, y los directores de la tesis, D. Francisco Herrera Triguero y D. José Manuel Benítez Sánchez, garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección de los directores de la tesis, y hasta donde nuestro conocimiento alcanza, en la realización del trabajo se han respetado los derechos de otros autores a ser citados cuando se han utilizado sus resultados o publicaciones.

Granada, Junio de 2016

El Doctorando | Los directores

Fdo: Daniel Peralta Cámara | Fdo: Francisco Herrera Triguero | Fdo: José Manuel Benítez Sánchez

Esta tesis doctoral ha sido desarrollada con la financiación del Programa de Becas de Formación de Profesorado Universitario del Ministerio de Educación y Ciencia, en su Resolución del 28 de Febrero de 2013, bajo la referencia FPU12/04902.

Agradecimientos

Son tantas las personas a las que estoy agradecido y tantos los motivos por los que lo estoy que no hay imprenta que pueda encuadernarlo todo. Así que os pido disculpas a todos y todas por no corresponder a lo que me habéis dado en estos años.

Empezando por los que han estado ahí desde el principio, agradezco infinitamente a mis padres su esfuerzo, que nunca les podré devolver porque me faltarían vidas. Si ahora tengo algo bueno es porque lo aprendí de vosotros primero. Gracias también a Víctor, por hacerme ver que las cosas inesperadas se consiguen con entusiasmo y dedicación. A mi abuela, tíos y primos, por esas estupendas comidas hablando de postres y de barriguillas. A Julia, por hacerme reír cuando lo he necesitado, y a Rosario y Miguel, por hacerme ver la vida de otra forma. Y un enorme agradecimiento a los que ya no estáis, porque siempre os echaremos de menos en los buenos momentos.

Debo agradecer a mis directores el enorme esfuerzo invertido en esta tesis y en enseñarme cosas que sólo se enseñan con el ejemplo. Paco, gracias por tener siempre claro el camino a seguir, por los e-mails de colores en los momentos necesarios, y por enseñarme que cuando uno hace lo que le gusta el entusiasmo nunca se pierde. José Manuel, gracias por aportar siempre el punto de vista práctico, por esas conversaciones a deshora, por aficionarme a la mitología griega, y por enseñarme y demostrarme que el saber no ocupa lugar.

Tras los directores van dos personas que casi lo han sido. Isaac, gracias por estar siempre ahí sea para revisar artículos, hacerme correr o prestarme almohadas. Sabes igual que yo que nada de esto habría sido igual sin ti. Salva, gracias por esos cafés que en 20 minutos me han hecho reorientar meses de trabajo, y por tantas risas juntos (aunque algunos no se lo crean).

Evidentemente no pueden faltar mis compañeros, siempre ahí para compartir sudores y emociones: Manu, Sergio, Pablo, Sara, José Antonio, Juanan, Kasia, Rafa, Manolo, Jorge, Natalia, Lala... ¡Y los seniors! Sin tener como modelo a Christoph, Joaquín, Victoria, Alberto, Julián, Fran, y un largo etcétera, los que vamos detrás no sabríamos ni por dónde empezar. Finalmente, un enorme abrazo a la sangre nueva, que dentro de unos pocos *deadlines* estarán quebrándose la cabeza para escribir sus propios agradecimientos: Paco, Jesús, Sergio, Elena, y todos los demás.

Huge thanks to Yvan Saeys and the entire DaMBi research group, for the amazing treatment that made me feel at home so many times. Everything is always better with an awesome playlist!

Un agradecimiento muy especial a la piña de Albolote y extrarradio, porque cualquier esfuerzo se hace ligero sabiendo que siempre se puede contar con vosotros para desconectar. Y no me olvido de esos personajes del C.C. Chana, por demostrarme que es posible trabajar arduamente, tomarse la vida con humor y nunca ser globero ni penco (salvo en algún caso).

Y terminando por lo más importante, gracias Ana. Porque si yo he invertido horas en esta tesis, tú has invertido muchas más (aunque no te des cuenta) en hacerme feliz. Porque al fin y al cabo, tú empezaste todo esto y mucho más.

Table of Contents

Chapter I

PhD dissertation

1 Introduction

Personal identification has been an increasingly important issue in a wide variety of fields, such as access control, criminology, forensics or automatic payment. In particular, in the last few years the amount of people that must be identified has been hugely increased: the identification requirements of big companies, law-enforcement departments or public administrations reach the hundreds of millions of individuals [iaf, uid]. This problem has been dealt with in various manners, some of the most popular of which are passwords and tokens. However, these solutions present some problems: passwords can be forgotten, tokens can be lost, and both can be stolen with relative easiness. Therefore, there has been a great interest in the scientific community to find a mean for identification that is not based on *what we know* (like passwords) nor *what we have* (like tokens), but rather on *who we are* [JFR07].

Biometrics provide an answer to that question by using features that are intrinsic to each person for the identification. The biometric features that can be used to identify individuals are diverse, such as face, fingerprints, ear, palmprint, finger veins, DNA, iris, and many others [JFR07]. Among these, fingerprints are the most used ones due to their desirable properties [MMJP09]:

- **Universality**: everybody has fingerprints, except in rare cases of severe amputations.
- **Uniqueness**: every finger of every person in the planet has a unique fingerprint.
- **Invariability**: fingerprints do not change along a person's life.
- **Easiness of use**: collecting fingerprint images is fast, cheap and non-invasive, especially with the development of specific electronic devices for this purpose.

A fingerprint is essentially a pattern of ridges and valleys located on a fingertip. These ridges and valleys form different types of patterns that can be used for their recognition. Although fingerprint patterns have been scientifically studied for more than a century [Hen00], manual comparison is a tedious and time-consuming process. In this context, automatic fingerprint recognition aims to speed up this process by registering the image of a fingerprint in a computer support, where the matching between fingerprints can be carried out in a systematic and efficient way.

Although fingerprints can be compared directly at the image level, such approaches usually do not yield good results due to the variances between different images of the same fingerprints,

such as rotations, translations and deformations of the skin. Image level comparison is also time consuming due to the high number of pixels of the image matrices. Therefore, the first step in the fingerprint recognition process is feature extraction [MMJP09]. This process consists of obtaining the relevant information of the image, so as to use it in further steps of the recognition. The various types of features that can be extracted from fingeprints are usually grouped into three levels: global (singular points, orientation maps and pseudoridges), local (minutiae) and detail (pores and intra-ridge features).

Among these, minutiae are by far the most used features for fingerprint recognition [PGT+15]. Minutiae are the bifurcations and the endings of the fingerprint ridges. They are easily described by their position and angle, and their number allows for efficient comparison algorithms. Minutiae-based fingerprint matching algorithms compare two sets of minutiae to determine whether they belong to the same fingerprint or not. The final output of the matching function is a measure of the similarity between the two compared fingerprints, which is usually either a binary truth-or-false value either a real-valued similarity score. Most of the current matching algorithms in the specialized literature start by computing a set of local structures, usually involving minutiae neighborhoods. Then, the local structures are compared with each other and a final consolidation step is applied to obtain the final similarity score or matching decision. Some matching methods involve complex computations and are very accurate, whilst others can compare sets of minutiae very fast, with a slightly lower accuracy.

The fingerprint recognition problem can be addressed from two points of view, each of which represents a different problem on its own [MMJP09]:

- **Verification** [JHB97] consists of comparing two fingerprint captures to determine whether they were taken from the same finger or not. It is a 1:1 comparison problem that typically involves a single application of a matching function.
- **Identification** [JHPB97] consists of exploring a database of template fingerprints to find the match of an input fingerprint, that is, a 1:n comparison problem for a database of n templates.

This thesis focuses on the identification problem. The general steps of an Automatic Fingerprint Identification System (AFIS) are depicted in Figure 1. First, the fingerprints of all the users that are to be identified are captured in a process called enrollment, to build the template fingerprint database. Then, when a new input fingerprint has to be identified, it is compared with each template fingerprint and the best match is returned. The main requirements of an AFIS can be synthesized into the following:

- **Accuracy**: the error rate of the identification. It should be as low as possible, to avoid both false positives (accepted impostors) and false negatives (rejected genuine users).
- **Efficiency**: the time needed to locate a fingerprint in the database. It should be kept as small as possible. Many real applications of fingerprint identification involve real-time constraints, so that a late response of the system is equivalent to a system failure. Very often, this time threshold is in the order of a few seconds.
- **Scalability**: the current needs of large-scale identification systems involve the possibility that template databases are likely to grow in all contexts. Therefore, an AFIS must be able to efficiently cope with such increased sizes, for instance by an adequate increase of the hardware resources.

- **Flexibility**: the system should be able to cope with any database size, any fingerprint characteristics (such as low-quality images or rolled prints), any performance requirement and any hardware configuration.

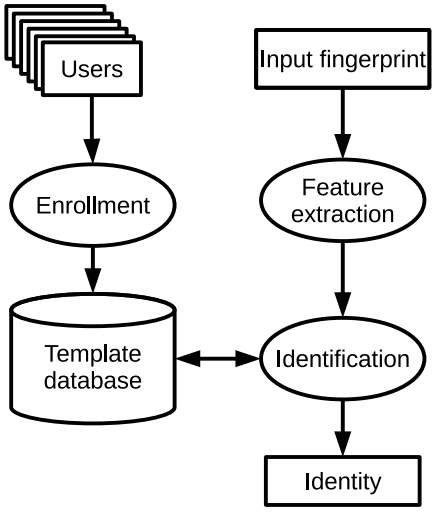


Figure 1: Workflow of an AFIS

It is immediate to deduce that identification is intrinsically more difficult than verification, as it can be approached as a succession of n verification steps. In fact, this is the approach followed by many AFIS [MMJP09]: the input fingerprint is compared to each fingerprint in the database, and the identity that yields the highest matching score is returned. Most approaches also apply a certain threshold on the response, to take into account the possibility that the input fingerprint is actually not in the database. The difficulty of identification with respect to verification is twofold:

- **High identification time**: a basic identification system takes at least n times longer than the underlying verification algorithm to identify a given fingerprint.
- **Accuracy loss**: an identification system not only has to find the single correct match for the input fingerprint among all database templates; it must also ensure that non-matching templates will not be detected as genuine matches. As the number of non-matching templates is at least $n - 1$, the probability of a false matching is usually not negligible.

By definition, these difficulties increase along with the value of n. Therefore, a direct brute-force approach for an AFIS that must identify within a database of more than a few thousands of people is not possible [PTSR$^+$14]. From the identification time point of view, most of these systems have real time constraints; for instance, the identification to access a building should not take much more than one second. From the accuracy point of view, a large value of n implies a large number of non-matching templates in the database, with the consequent increase of the probability of a wrong identification. As current society necessities for identification are reaching the order of hundreds of millions of people [iaf, uid], there is a strong need of scalable, accurate solutions that can adequately deal with this problem.

High Performance Computing (HPC) is one of the tools that support the modern Science, as it enables the computation of multiple calculations in a reasonable time by means of massive computational resources [Sto92]. The scientific literature thrives with successful applications of

HPC systems to real problems, and most big companies and public institutions implement their own HPC infrastructures to process their data. As such, HPC is a promising tool for the problem of fingerprint identification in large databases.

In this context, one of the hot research topics during the last few years has been big data, which can be defined is as the amount of data that cannot be processed within a single machine. Big data poses an interesting challenge in many fields, but also offers an opportunity to extract better and more valuable knowledge. So far, big data techniques have been successfully applied to various problems [FdRL+14]. Several HPC-based frameworks have been developed to help with this task; two of the most popular ones are Apache Hadoop [Whi12] (which implements the MapReduce paradigm) and Apache Spark [KKWZ15].

However, a problem of this scale cannot be solved by HPC alone. It is necessary to delve deeper into the fingerprint identification problem to find new ways to tackle very large scale identification more efficiently. In this context, Data Mining consists of evaluating sets of data to extract new knowledge from them, usually by detecting patterns that were previously unknown [WFH11]. Such knowledge can be used to design an implement new ways of approaching problems related with that kind of data.

Preprocessing is one of the key components of any process involving data mining, as it is necessary to obtain all the benefits from the data mining techniques [GLH15]. In the context of fingerprint identification, preprocessing techniques can be applied in several manners to enhance the accuracy and the runtime of the identification itself, for instance to enhance the features extracted from the fingerprints or to gather information to improve the identification process.

One of the most extended ways to improve the identification time of an AFIS is the use of classification techniques [DHS12]. Fingerprints can be divided into five classes according to the visual pattern of their ridges [Hen00]. If the class of an input fingerprint is correctly determined, then it is possible to perform the identification by comparing it only with the template fingerprints belonging to the same class [MMJP09]. The number of comparisons with template fingerprints with respect to the database size is called *penetration rate* and its reduction can be key in the performance of identification system. However, the misclassification of a fingerprint can lead to identification errors; therefore, it is crucial to reduce the classification error as much as possible.

Information fusion is a paradigm used in many disciplines to improve the overall precision of a given process, including biometrics [RJ03]. Biometric problems are intrinsically adapted to information fusion approaches; the fusion of information can be performed at many levels [LN17]: data level, feature level, score level or decision level. In particular, fusion approaches in fingerprint identification are usually grouped into two categories: using several fingerprint images [JFR07] and using several matching algorithms [JPC99]. Information fusion has been steadily used to increase the accuracy of fingerprint and other biometric recognition systems. However, it affects negatively the runtime, as more computation has to be performed to obtain redundant information. In this context, it would be desirable to use information fusion from a different point of view to tackle both the accuracy and the runtime problems.

This thesis starts by presenting a deep study of the scientific literature on minutiae-based local matching matching techniques, establishing a taxonomy of the available local structures and consolidation methods, and highlighting the main advantages and drawbacks of each of them. Then, we will present a minutiae filtering algorithm that removes spurious or misleading minutiae to improve both the identification time and the accuracy of the recognition process. After that, we will describe two frameworks for massively parallel fingerprint identification, which are able to execute different matching algorithms adapting to the underlying hardware for maximum performance and

full scalability. We will also develop a framework to combine the information of two fingerprints and the capabilities of two different matching algorithms to address both problems that hinder identification in large databases: the high identification time and the loss of accuracy. Finally, we describe a new classification strategy to reduce the penetration rate of the identification. Finally

After this introduction section, Section 2 describes in detail the background of the main areas addressed in this thesis: fingerprint feature extraction (Section 2.1), fingerprint identification (Section 2.2), high performance computing (Section 2.3), database penetration reduction and fingerprint classification (Section 2.4) and information fusion for fingerprint identification (Section 2.5).

After that, Section 3 presents the justification of this memory, describing the open problems addressed throughout this thesis. The objectives pursued to address these problems are detailed in Section 4, along with the methodology followed along the thesis in Section 5. Section 6 summarizes the works that compose this memory, while Section 7 presents the results obtained in them, performing an analysis in relation with the tackled objectives and how they have been reached. Section 8 presents the conclusions after the work carried out for this thesis. Finally, in Section 9 we point out several future lines of work that have been derived from the results achieved.

Introducción

La identificación de personas ha sido un problema de importancia creciente en una amplia variedad de campos, como el control de acceso, la criminología, el análisis forense o el pago automático. En particular, durante los últimos años la cantidad de personas que deben ser identificadas ha aumentado enormemente: los requisitos de identificación de grandes empresas, fuerzas de seguridad o administraciones públicas alcanzan tamaños de los cientos de millones de individuos [iaf, uid]. Este problema se ha intentado resolver de diversas formas; algunas de las más populares son las contraseñas y los objetos (como llaves o tarjetas de identificación). Sin embargo, estas soluciones presentan algunos problemas: las contraseñas pueden ser olvidadas, los objetos se pueden perder, y ambos pueden ser robados con relativa facilidad. Por lo tanto, ha habido un gran interés en la comunidad científica por encontrar un medio para la identificación que no esté basado en *qué sabemos* (como las contraseñas) ni en *qué tenemos* (como los objetos) sino más bien en *quién somos* [JFR07].

La Biometría proporciona una respuesta a esta pregunta utilizando características intrínsecas a cada persona para la identificación. Las características biométricas que se pueden utilizar para identificar individuos son diversas, como el rostro, huellas y venas dactilares, oreja, palma de la mano, ADN, iris, y muchas otras [JFR07]. Entre ellas, las huellas dactilares son las más utilizadas debido a sus propiedades [MMJP09]:

- **Universalidad**: todo el mundo tiene huellas, excepto en casos raros de amputaciones graves.
- **Unicidad**: cada dedo de cada persona en el planeta tiene una huella única.
- **Invariabilidad**: las huellas no cambian a lo largo de la vida de una persona.
- **Facilidad de uso**: la recolección de huellas es rápida, económica y no invasiva, especialmente tras el desarrollo de dispositivos electrónicos específicos para este propósito.

Una huella dactilar es en esencia un patrón de crestas y valles situado en la yema de un dedo. Estas crestas y valles forman distintos tipos de patrones que se pueden utilizar para el reconocimiento de las huellas. Aunque se han estudiado científicamente durante más de un siglo [Hen00], la comparación manual es un proceso largo y tedioso. En este contexto, el reconocimiento automático de huellas busca acelerar este proceso registrando la imagen de la huella en un soporte computacional, donde el emparejamiento entre huellas se pueda llevar a cabo de forma sistemática y eficiente.

Aunque las huellas se pueden comparar directamente a nivel de imagen, habitualmente este enfoque no proporciona buenos resultados debido a la variabilidad existente entre distintas imágenes de la misma huella, como rotaciones, traslaciones y deformaciones de la piel. La comparación de imágenes también es computacionalmente costosa debido al elevado número de píxeles de sus matrices. Por tanto, el primer paso en el proceso de reconocimiento de huellas es la extracción de características [MMJP09]. Este proceso consiste en obtener información relevante a partir de la imagen para utilizarla en pasos posteriores del reconocimiento. Los diversos tipos de características que se pueden extraer de las huellas se suelen agrupar en tres niveles: global (puntos singulares, mapas de orientación y pseudocrestas), local (minucias) y detalle (poros y características intracresta).

Entre ellas, las minucias son con diferencia la característica más utilizada para el reconocimiento de huellas dactilares [PGT+15]. Las minucias son las bifurcaciones y finales de las crestas de la huella. Se describen fácilmente mediante su posición y ángulo, y su número permite el diseño

de algoritmos de comparación eficientes. Los algoritmos de emparejamiento (o *matching*) basados en minucias comparan dos conjuntos de minucias para determinar si pertenecen a la misma huella o no. La salida final de la función de *matching* es una medida de la similitud entre las dos huellas comparadas, la cual suele ser o bien un valor binario (verdadero o falso) o un *score* de similitud de valor real. La mayor parte de los algoritmos de *matching* en la literatura especializada empiezan calculando un conjunto de estructuras locales, que habitualmente se basan en los vecindarios de las minucias. A continuación, las estructuras locales se comparan entre ellas y se aplica un paso final de consolidación para obtener el *score* de similitud final. Algunos métodos de *matching* utilizan cálculos complejos y son muy precisos, mientras que otros pueden comparar conjuntos de minucias de forma muy rápida, con una precisión algo menor.

El problema del reconocimiento de huellas dactilares se puede abordar desde dos puntos de vista, cada uno de los cuales representa un problema diferente [MMJP09]:

- **Verificación** [JHB97]: consiste en comparar dos capturas de huellas para determinar si fueron tomadas del mismo dedo o no. Es una comparación 1:1 que implica típicamente una única aplicación de una función de *matching*.
- **Identificación** [JHPB97]: consiste en explorar una base de datos de huellas para encontrar la pareja de una huella de entrada, es decir, es un problema de comparación 1:n para una base de datos de n huellas (denominadas *template*).

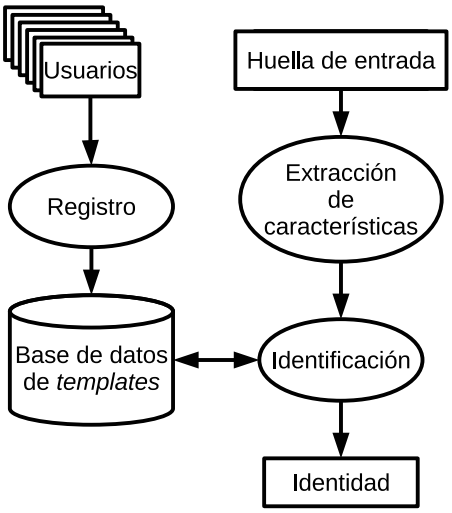


Figura 2: Pasos generales de un sistema de identificación automático

Esta tesis se centra en el problema de la identificación. Los pasos generales de un sistema automático de identificación de huellas (*Automatic Fingerprint Identification System*, AFIS) se muestran en la Figura 2. Primero, las huellas de todos los usuarios que se deban identificar son capturadas en un proceso de registro, para construir la base de datos de huellas *template*. Luego, cuando una nueva huella debe identificarse, se compara con cada una de las huellas *template* y se devuelve el mejor emparejamiento. Los principales requisitos de un AFIS se pueden sintetizar en los siguientes:

- **Precisión**: la tasa de error de la identificación. Debe ser lo más baja posible, para evitar tanto falsos positivos (impostores aceptados) como falsos negativos (usuarios genuinos rechazados).

- **Eficiencia**: el tiempo requerido para localizar una huella en la base de datos. Debe mantenerse lo más pequeño posible. Muchas aplicaciones reales de la identificación con huellas implican restricciones de tiempo real, de forma que un resultado tardío equivale a un fallo del sistema. Muy a menudo, este umbral de tiempo está en el orden de unos pocos segundos.
- **Escalabilidad**: las necesidades actuales de los sistemas de identificación a gran escala implican una alta probabilidad de que las bases de datos de huellas crezcan en todos los ámbitos. Por lo tanto, un AFIS debe ser capaz de trabajar de forma eficiente con bases de datos de tamaño creciente, por ejemplo mediante un incremento adecuado de los recursos computacionales.
- **Flexibilidad**: el sistema debe ser capaz de enfrentarse a bases de datos de cualquier tamaño, huellas con cualquier tipo de propiedades (como imágenes de baja calidad o impresiones rodadas), cualquier requisito de rendimiento y cualquier configuración *hardware*.

Es inmediato deducir que la identificación es intrínsecamente más compleja que la verificación, dado que se puede ver como una sucesión de n pasos de verificación. De hecho, éste es el enfoque seguido por muchos AFIS [MMJP09]: la huella de entrada se compara con cada una de las huellas en la base de datos, y se devuelve la identidad que proporciona el *score* de *matching* más alto. La mayor parte de este tipo de soluciones también aplican un cierto umbral sobre la respuesta, para tener en cuenta la posibilidad de que la huella de entrada no esté en la base de datos. La dificultad de la identificación con respecto a la verificación es doble:

- **Elevado tiempo de identificación**: un sistema de identificación básico tarda n veces más en identificar una huella que el algoritmo de verificación subyacente.
- **Pérdida de precisión**: un sistema de identificación no solamente debe encontrar el único emparejamiento correcto para la huella de entrada entre todas las huellas de la base de datos; también debe asegurar que las huellas no emparejadas no se detectarán como parejas genuinas. Al ser el número de *templates* no emparejados al menos $n - 1$, la probabilidad de un falso emparejamiento habitualmente no es despreciable.

Por definición, estas dificultades se incrementan junto con el valor de n. Por lo tanto, no es factible un enfoque directo por fuerza bruta para un AFIS que debe identificar en una base de datos de más de unos pocos miles de huellas [PTSR+14]. Desde el punto de vista del tiempo de identificación, la mayoría de esos sistemas tiene restricciones de tiempo real; por ejemplo, la identificación para acceder a un edificio no debería tardar mucho más de un segundo. Desde el punto de vista de la precisión, un valor alto de n implica una gran cantidad de *templates* en la base de datos que no emparejan con la huella de entrada, con el consiguiente aumento de la probabilidad de una identificación errónea. Las necesidades de identificación actuales de la sociedad alcanzan el orden de los cientos de millones de personas [iaf, uid], por lo que hay una fuerte necesidad de soluciones precisas y escalables que resuelvan este problema de forma adecuada.

La computación de altas prestaciones (*High Performance Computing*, HPC) es una de las herramientas que dan soporte a la Ciencia moderna, al posibilitar el cómputo de múltiples cálculos en un tiempo razonable mediante el uso de recursos computacionales masivos [Sto92]. La literatura científica presenta múltiples aplicaciones exitosas de sistemas HPC sobre problemas reales, y la mayor parte de grandes empresas e instituciones públicas implementan sus propias infraestructuras HPC para procesar sus datos. Por tanto, la HPC es una herramienta prometedora para el problema de identificación de huellas dactilares en grandes bases de datos.

En este contexto, una de las líneas de investigación de más alta actividad en los últimos años es el *big data*, que se puede definir como la cantidad de datos que no se puede procesar con una única máquina. El *big data* crea un desafío interesante en muchos campos, pero también ofrece una oportunidad para extraer conocimiento de mayor calidad y valor. Hasta el momento, las técnicas *big data* se han aplicado con éxito sobre varios problemas [FdRL+14]. Varias plataformas basadas en HPC han sido desarrolladas para dar soporte a esta tarea; dos de las más populares son Apache Hadoop [Whi12] (que implementa el paradigma MapReduce) y Apache Spark [KKWZ15].

No obstante, un problema de esta escala no se puede solucionar mediante HPC por sí sola. Es necesario ahondar más en el problema de la identificación con huellas para encontrar nuevas vías de abordar de forma eficiente la identificación a muy gran escala. En este contexto, la minería de datos consiste en evaluar conjuntos de datos para extraer nuevo conocimiento a partir de ellos, por lo habitual mediante la detección de patriones que eran previamente desconocidos [WFH11]. Este conocimiento se puede usar para diseñar e implementar nuevas formas de abordar problemas relacionados con el tipo de datos analizado.

El preprocesamiento es uno de los componentes clave de cualquier proceso que implique minería de datos, puesto que es necesario para obtener todos los beneficios de tales técnicas [GLH15]. En el contexto de la identificación con huellas dactilares, las técnicas de preprocesamiento se pueden aplicar de varias maneras para mejorar la precisión y el tiempo de la identificación, por ejemplo mejorando las características extraídas de las huellas o recogiendo información que permita optimizar el proceso de identificación.

Una de las maneras más extendidas de mejorar el tiempo de identificación de un AFIS es el uso de técnicas de clasificación [DHS12]. Las huellas se pueden dividir en cinco clases en función del patrón visual de sus crestas [Hen00]. Si la clase de una huella de entrada se determina correctamente, es posible realizar la identificación comparándola solamente con las huellas *template* pertenecientes a la misma clase [MMJP09]. El número de comparaciones con huellas *template* con respecto al tamaño de la base de datos se llama *tasa de penetración*, y su reducción puede ser clave en el rendimiento de un sistema de identificación. Sin embargo, una clasificación errónea de una huella puede conllevar errores de identificación; por consiguiente es crucial reducir el error de clasificación tanto como sea posible.

La fusión de información es un paradigma utilizado en muchas disciplinas para mejorar la precisión global de un determinado proceso, incluyendo la biometría [RJ03]. Los problemas biométricos están intrínsecamente adaptados a los enfoques de fusión de información, dado que ésta puede aplicarse en varios niveles [LN17]: datos, características, *score* o decisión. En particular, las técnicas de fusión en identificación de huellas se suelen agrupar en dos categorías: el uso de varias imágenes de huellas [JFR07] y el uso de varios algoritmos de *matching* [JPC99]. La fusión de información se ha utilizado con regularidad para aumentar la precisión de los sistemas de identificación con huellas y otras características biométricas. Sin embargo, afecta negativamente al tiempo de ejecución, puesto que la obtención de información redundante requiere de un cómputo mayor. En este contexto, es deseable utilizar la fusión de información desde un punto de vista diferente para abordar conjuntamente los problemas de precisión y tiempo de ejecución.

Esta tesis empieza presentando un profundo estudio de la literatura científica sobre técnicas de *matching* locales basadas en minucias, estableciendo una taxonomía de los tipos de estructuras locales y métodos de consolidación, y señalando las principales ventajas e inconvenientes de cada uno de ellos. A continuación, se presenta un algoritmo de filtrado de minucias que elimina minucias espurias o engañosas para mejorar tanto el tiempo de identificación como la precisión del proceso de reconocimiento. Después, se describen dos plataformas para identificación de huellas en arquitecturas masivamente paralelas, que son capaces de ejecutar diferentes algoritmos de *mat-*

ching adaptándose al *hardware* subyacente para un rendimiento máximo y escalabilidad completa. También se desarrolla una plataforma para combinar la información de dos huellas dactilares y las capacidades de dos algoritmos de *matching* diferentes, para abordar de forma conjunta los dos problemas que dificultan la identificación en grandes bases de datos: el elevado tiempo de identificación y la pérdida de precisión. Finalmente, se describe una nueva estrategia de clasificación para reducir la tasa de penetración de la búsqueda.

Tras esta sección introductoria, la Sección 2 describe en detalle las áreas principales a las que hace referencia esta tesis: extracción de características de las huellas dactilares (Sección 2.1), identificación de huellas (Sección 2.2), computación de altas prestaciones (Sección 2.3), reducción de la penetración en la base de datos y clasificación de huellas (Sección 2.4) y fusión de información para identificación con huellas (Sección 2.5).

Posteriormente, la Sección 3 presenta la justificación de esta memoria, describiendo los problemas abiertos abordados a lo largo de esta tesis. Los objetivos perseguidos en la búsqueda de la solución a estos problemas se detallan en la Sección 4, junto con la metodología seguida para el desarrollo de la tesis en la Sección 5. La Sección 6 resume los trabajos que componen esta memoria, y la Sección 7 presenta los resultados obtenidos en ellos, realizando un análisis en relación con los objetivos considerados y cómo se han alcanzado. La Sección 8 presenta las conclusiones tras el trabajo llevado a cabo para esta tesis. Finalmente, en la Sección 9 se destacan varias líneas de trabajo futuro que se derivan de los resultados conseguidos.

2 Preliminaries

This section points out the preliminary background on the topics addressed by this thesis. First, Section 2.1 explains the features extracted from fingerprints. Section 2.2 details the main characteristics of the fingerprint identification process. Section 2.3 presents some knowledge on high performance computing and its previous applications on fingerprint identification. Then, Section 2.4 explains the highlights on database penetration reduction and fingerprint classification. Finally, the use of information fusion for fingerprint identification is detailed in Section 2.5.

2.1 Feature extraction

Most fingerprint recognition approaches in the literature do not work with the images themselves. Rather, a set of features is first extracted from the image [MMJP09] to describe useful patterns; then, these features are used for the different operations that can be performed on fingerprints, such as verification, identification or classification.

There are various types of features that can be extracted from fingerprints. These features are usually grouped into three categories, ranging from coarse to fine-grain patterns [MMJP09]:

- **Level 1 (global)**: refers to the global ridge line flow. The most used features in this category are the following:
  - **Orientation map**: a matrix containing the direction of the ridge line flow for each block of the fingerprint image.
  - **Singular points**: points around which the ridge lines are structured. There are two main types of singular points: deltas and cores.
- **Level 2 (local)**: considers minutiae details extracted from the ridge skeleton. Although there are multiple types of minutiae, most works use the two most easily extracted ones: ridge endings and bifurcations.
- **Level 3 (fine-detail)**: involves intra-ridge details such as width, shape, ridge contours, sweat pores and creases.

Each of these different types of features is better suited to a different purpose. On the one hand, the distinctiveness of level 1 features is not enough to perform an accurate verification; however, they are suited to carry out a classification process [GDP+15b]. On the other hand, the distinctiveness properties and average number of minutiae make them the best candidate for verification and identification [CFM10]. Finally, level 3 features are mainly used in combination with minutiae to improve the accuracy of the recognition [JCD07].

There are many proposals for feature extraction in the literature. Each type of feature requires a different approach for its extraction [GDP+15b]. For instance, most orientation map extractors are based on the gradient of the image [JPC99, Liu10] or the slits sum method [CGW+95, CLMM99]. Singular point detection often uses the Poincaré method [KJ96] or complex filters [Liu10]. Finally, minutiae extractors can be categorized into one the following [MMJP09]:

- **Binarization-based methods:** are the largest group. They require a binary fingerprint image. The image passes through a thinning process that reduces the ridge width to a single

pixel, to produce a skeleton image. Then, the minutiae can be easily extracted with a scan of the image. Some methods are explained in [RCJ95, WGT+10].

- **Direct gray-scale extraction**: some methods do not apply binarization or thinning methods to the image. Instead, the detection is carried out directly from the gray-scale pixel map [LHC00, CF01].

2.2 Fingerprint identification

Fingerprint identification consists of exploring a template fingerprint database, looking for the mate of a given input fingerprint. Therefore, the identification typically involves comparing the input fingerprint to each template in order to select the most likely match. In general, fingerprints are not directly compared at the image level due to the large intra-class variance of the images (caused by factors such as translations, rotations and skin deformation). Therefore, in this section we will focus on minutiae-based matching, which is the most extended approach [PPJ02]. Instead, a set of minutiae is extracted from each fingerprint, constituting the basic information for a comparison algorithm.

A template fingerprint T and an input fingerprint I (or rather, their respective minutiae sets) are compared in a process called *matching*, which returns a similarity value q. The matching is a function Q that maps pairs of fingerprints to the similarity domain, so that $Q(I,T) = q$. In verification approaches the value q is usually boolean, indicating if the two compared fingerprints are the same or not. Identification approaches are more often based on numeric scores that rate the similarity level of the compared fingerprints.

Minutiae-based fingerprint matching can be carried out at two different levels [MMJP09]:

- **Global matching**: the entire minutiae sets are directly compared. This type of matching is more sensitive to image distortions, rotations and translations, although it takes advantage from a global view of the fingerprints. Some proposals are presented in [RKCJ96, CCHW97].
- **Local matching**: local structures are computed from the minutiae set, usually involving the immediate neighborhood of each minutiae. This neighborhood can be computed either as the nearest neighbors [JY00] or using a fixed radius [RBPV00, CTY06, CFM10]. Then, these local structures are compared among themselves to compute the matching score. Thanks to the use of local information, local matching algorithms are intended to be invariant to rotations and translations, and more robust to skin deformations.

In practice, most modern matching algorithms follow a hybrid approach: a local matching process is used to find the most similar local structures and extract a common alignment for both fingerprints; then, a global matching procedure (called in this context *consolidation*) is applied to obtain the final matching score.

In an identification system the matching process has to be applied once for each template T_i in the database. In such a context, the identity returned by the identification system follows this form:

$$\text{Identity} = \arg\max_i Q(I, T_i) \quad i \in \{1, 2, ..., n\} \tag{I.1}$$

Identification implies a heavy computational effort, especially as the size of the database n grows. Some proposals have arisen in the last few years to palliate this problem, such as the

use of high performance computing (Section 2.3) or the reduction of the database penetration rate (Section 2.4).

There are several measures that quantify the goodness of verification and identification approaches. Some of the most used measures throughout the literature are the following:

- **Measures for verification**:
  - **False Match Rate (FMR)**: probability of stating that two different fingerprints are the same.
  - **False Non-Match Rate (FNMR)**: probability of denying the equality of two captures of the same fingerprint.
  - **Equal Error Rate (ERR)**: point at which the FMR and FNMR are equal.
  - **FMR100**: lowest achievable FNMR for a FMR $\leq 1\%$.
  - **ROC**: curve that plots the Genuine Matching Rate (GMR $= 1 -$ FNMR) versus the FMR.
- **Measures for identification**:
  - **True Positive Rate (TPR)**: percentage of test fingerprints that are correctly identified in the database, when only the best matching identity is returned.
  - **R100**: lowest rank (i.e. the number of identities returned from the database) that allows an error lower than 1%.
  - **Cumulative Match Curve (CMC)**: curve that represents the error associated to each rank.

These measures allow to compare objectively different matching algorithms and identification systems, as they give information on different aspects of their behavior.

2.3 High Performance Computing (HPC)

HPC systems are widely used for distributed and parallel computing in many fields of Science and industry. Their use is oriented at obtaining diverse advantages [Sto92]:

- **Efficiency**: the computation can be carried out in parallel so that the results are obtained faster.
- **Robustness**: the processing workload is spread among different computers, allowing the system to be fault tolerant. If one machine fails, the others can assume its work.
- **Scalability**: nowadays, the hardware industry evolves towards integrating a higher number of cores and collaborating processors. Thus, an adequately designed algorithm that is able to solve bigger problems just by using more computing power could solve arbitrarily big problems in the future, without being modified.

Hardware has evolved in two main ways to support HPC, both of which are focused on increasing and exploiting the parallelism. On the one hand, several computers can be integrated with a high-throughput network to conform a cluster. This approach provides a great flexibility when the computing capacity has to be increased, as this can be done merely by purchasing more

machines. However, the performance can become limited by the network speed. On the other hand, each single processor is composed nowadays by an ever growing number of cores that are able to handle several execution threads in parallel. These threads can communicate very efficiently using shared memory or internal mechanisms. The limitation of this approach is that the number of cores within a processor is limited and cannot be changed over time. Most practical deployments nowadays implement a hybrid approach: clusters of multicore computers. Such clusters are able to take advantage of the strengths of both approaches.

Similarly, several lines of software aim to support the development of applications for HPC infrastructures. In multi-computer environments, tools like the Message Passing Interface (MPI) provide the means of communicating several machines using the intermediary network. Within a single multi-core machine, libraries that handle parallel or concurrent threads (such as OpenMP or POSIX threads) offer an optimal performance.

There are several performance measures for parallel systems. The most widely used is the speedup ($S = t_s/t_p$), which measures the ratio between the execution time of a sequential approach and that of the parallelized implementation. If a calculation is executed in p processing cores, and a fraction f of it is performed in parallel, Amdahl's Law [Amd67] provides the maximum attainable speedup:

$$S^* = \frac{1}{(1-f) + \frac{f}{p}} \tag{I.2}$$

According to the equation, a fully parallelizable ($f = 1$) algorithm would have a maximum speedup of p. However, in practice there are several factors that hinder this speedup:

- All algorithms include some non-parallelizable computation, so $f < 1$. Even if it represents a small amount with respect to the whole runtime, the impact on the maximum speedup can be important, as shown in Equation I.3, which shows the maximum speedup with an infinite number of processors:
$$\lim_{p \to +\infty} \frac{1}{(1-f) + \frac{f}{p}} = \frac{1}{1-f} \tag{I.3}$$
- The parallel version of a sequential algorithm usually introduces more operations, such as communication and synchronizations, to organize the workload and ensure correct results.
- In many cases, such communication and synchronization times can become the bottleneck of the parallel approach, especially for high values of p.

The scientific literature presents some examples of applications of HPC infrastructures to the fingerprint identification problem [ISA11, DGLN11]. These approaches can be grouped into two main categories: client-server systems, where the server forwards identification requests to the clients, which explore the database [HAL+08, MLH11]; and agent-based systems, where each agent performs some processing task and then shares the results with other agents [NH04, BGMB08].

In recent years, a number of HPC frameworks for big data processing have been developed. Such frameworks provide several characteristics that can be summarized into high availability, fault-tolerance, distributed data storage, massive parallelization and high data throughput. This allows to develop specific applications following certain programming paradigms, such as MapReduce [DG08]. In exchange, they introduce a certain overhead in processing and communication times. Such frameworks have been successfully applied to various problems in several fields, including biometric recognition [SR11, KS13].

2.4 Database penetration reduction and fingerprint classification

Despite the enormous power of modern HPC infrastructures, the acquisition and deployment costs required for very large-scale databases can become high. Therefore, other solutions must be found to tackle the problem. One of them consists of reducing the number of template fingerprints that are compared with the input. The percentage of database that is explored is called penetration rate [RB04].

In this context, there are two main families of methods that attempt to reduce the database penetration rate [MMJP09]. Indexing algorithms [Cap11] perform a mapping of the fingerprints to a multi-dimensional space, so that different captures of the same fingerprint should be close to each other in the target space. Classification algorithms [GDP$^+$15b] split the database into a certain number of disjoint classes, so that the input fingerprint is only compared with those fingerprints that belong to its same class. In this thesis, we focus on the classification approach to the problem.

In a machine learning context, classification [DHS12] consists of extracting knowledge from a set of n input examples $x_1, ..., x_n$, each of which is labeled with one of m classes $y_i \in \{c_1, ..., c_m\}$ and characterized by p features $a_1, ..., a_p$. The aim of a classifier is to be able to correctly predict the class of a new unseen example. In fingerprint classification, the most widely used system is the five-class model defined by Henry [Hen00] (Figure 3).

(a) Arch (b) Left loop (c) Right loop (d) Tented arch (e) Whorl

Figure 3: Cinco fingerprint classes defined by Henry [Hen00]

From this point of view, each fingerprint corresponds to a single example, represented by a feature vector. In fingerprint classification, unlike what occurs typically in machine learning, the construction of such feature vectors is often considered to be included into the classification process, so that the classification refers to the problem as a whole. Therefore, many proposals in the literature cope both the feature extraction process and the classifier itself working on top of the proposed features.

The features used for fingerprint classification are usually based on level 1 features, such as orientation maps [CLMM99], ridge structure [Sen97] and singular points [NAKMM04]. Other approaches are also based on the application of filters to the fingerprint image; Gabor filters are one of the most popular for this purpose [JPC99]. Finally, many authors combine several of the previously mentioned features to maximize the accuracy yielded by the classifier [HMCC08, Liu10].

A review of the most prominent feature extraction and classification methods in the scientific literature was carried out in [GDP$^+$15b, GDP$^+$15a], describing a taxonomy of the various types of features and classification approaches, and featuring an extensive experimental study. The knowledge gathered in this review has been used as starting point for the new proposals for fingerprint classification presented in this thesis.

2.5 Information fusion for fingerprint identification

Information fusion can be applied to the fingerprint recognition problem in several ways and at multiple levels. For instance, the use of different types of features or the inclusion of classification strategies already represents a type of information fusion [LN17]. When we focus on the specific minutiae-based fingerprint identification problem, information fusion can be carried out in two main ways.

On the one hand, several fingers [JFR07] can be used altogether to increase the distinctiveness of the identities and to avoid the difficulties posed by injured fingertips or low quality scans. A matching function for multiple (k) fingerprints follows the form $Q(\mathcal{I}, \mathcal{T}_i)$ where $\mathcal{I} = \{I_j \mid j \in \{1, ..., k\}\}$ and $\mathcal{T}_i = \{T_{ij} \mid j \in \{1, ..., k\}\}$.

On the other hand, several matching algorithms can be combined [JPC99, NM06] to profit from their advantages while leaving aside their weaknesses. Multi-algorithm techniques work in a similar way as multi-finger ones, so that the fused score for k algorithms is $Q(I, T_i) = \mathcal{F}\left(Q_1(I, T_i), ..., Q_k(I, T_i)\right)$, where $\mathcal{F}$ is an aggregation function and each Q_j is a different matching function.

A third category of approaches uses several captures of each fingerprint, but it has already been proven that the multi-finger approach is more effective due to the use of less correlated information [MMJP09].

Multi-finger and multi-algorithm approaches can be categorized together according to the type of fusion they perform:

- **Feature fusion** [JS02, NSV07]: merges all k fingerprints of an identity into a single structure. This structure is compared to all n template structures, avoiding the need of performing k matchings per identity but requiring specific matching algorithms to handle such structures as well as an additional conversion step.
- **Score fusion** [JPC99, MRD13]: performs one application of the matching function for each fingerprint or algorithm, and aggregates the results into a single score, typically by adding or multiplying them. It does not need a specific matching algorithm; however, the use of k fingerprints or k matchings multiplies the identification time by k.
- **Decision fusion** [RKCJ96, PJ02]: can be seen as a special case of score fusion, where matching is performed hierarchically. When the k input fingerprints are compared with some k template fingerprints for a given identity, the first pair is compared first. If the resulting score meets a certain condition, the second pair is compared, and so on, until a decision is reached.

Most fusion approaches in the literature attempt to improve the matching accuracy and they have already proven to yield good results; nevertheless, the runtime is not considered. Such methods are therefore not suitable to tackle the identification in large databases because the required time is higher than it is for single-fingerprint or single-matcher approaches.

3 Justification

As explained in the previous sections, there is a need of efficient, scalable automatic fingerprint identification systems that are able to carry out identifications in very large databases. Various tools and techniques, such as preprocessing, HPC, classification or the combination of several fingerprints and matchers have the potential of improving the identification process to obtain both faster and more accurate solutions.

To obtain such an identification system, the following issues should be addressed:

- In the last decades, a large number of fingerprint matching algorithms have been proposed. All these approaches should be revised and systematically analyzed, to allow the study of their common structures, so as to determine their strengths and weaknesses. Specifically, approaches combining a local matching process with a global consolidation phase have proven to offer a good performance within a reasonable computing time.
- Spurious minutiae are one of the problems that hinder the performance of any matching algorithm. A correct handling of this issue would allow to obtain more accurate fingerprint identification systems. Additionally, the removal of spurious minutiae has a positive impact on the identification time, reducing the computational needs for each matching operation.
- When dealing with large fingerprint databases, the matching time becomes a very important bottleneck for any identification system. Furthermore, the identification accuracy is degraded as the size of the database grows. It is necessary to tackle both problems with suitable tools, such as:
  - Parallelization techniques to use the computing power of HPC infrastructures, so as to eliminate that bottleneck in a scalable manner. The new frameworks for big data could also provide an interesting support for very large scale proposals.
  - Multi-finger and multi-matcher approaches that have already been proven to enhance the accuracy of verification and identification systems. However, there has been no attempt to use such ideas to also improve the identification time along with the accuracy. In combination with an adequate HPC infrastructure and implementation, a multi-finger and multi-matcher solution could be decisive in obtaining an accurate, fully scalable fingerprint identification system.
- Finally, tackling the penetration rate of the identification search would allow for a better performance of the identification system. There are multiple fingerprint classification algorithms published in the specialized literature; however, there is no proposal yet to evaluate its impact on the actual subsequent identification process. Additionally, some of the feature extractors that produce accurate results in classification reject some of the fingerprints and are unable to extract any features for them. This problem should be addressed in order to maximize the reduction of the penetration rate.

All the aforementioned issues can be encompassed within the subject of this thesis: the development and application of data mining techniques in high performing computing architectures for fingerprint identification in very large databases.

4 Objectives

After an adequate study of the current state of all the areas described in the previous sections, it is possible to focus on the actual objectives of this thesis. These will include the research and analysis of the background fields described before, and the development of advanced models for fingerprint identification based on the most promising properties of each field. More specifically, the objectives are:

- **To study the current state of the art in minutiae-based fingerprint matching.** In particular, matching algorithms based on local structures and global consolidation have proven to reach a good trade-off between accuracy and computational complexity. The common structures and processes followed by all published methods, as well as their particularities, their weaknesses and their advantages, should be fully understood in order to settle the basis for the remaining work of this thesis.
- **To propose a new preprocessing method to eliminate spurious minutiae.** Spurious minutiae have a negative impact on any matching algorithm, both in terms of accuracy and runtime. The developed preprocessing method should remove as many spurious minutiae as possible, while keeping genuine minutiae unmodified, which would speed up and improve the accuracy of matching algorithms.
- **To develop efficient, scalable and accurate approaches for fingerprint identification.** Such methods are necessary to address the identification in very large databases. To tackle this objective, we will follow two main paths:
  - *Strategies based on HPC*: from this point of view, two alternatives are considered: an implementation based on MPI for multi-node and multi-core clusters, and another that takes advantage of big data frameworks. Furthermore, the side advantages of the development of such frameworks are twofold. First, such proposals would certainly prove to be of interest for a broad scope of the scientific community; the development of a publicly available software would settle the basis for reproducible research, and other researchers could benefit from the knowledge acquired for this thesis. Second, they would constitute a valuable support for the experimentation needed for other parts of this thesis.
  - *Strategies based on information fusion*: the fusion of several matching algorithms and several fingers should improve the two tackled goals: accuracy and efficiency. Any identification system must find a trade-off between them; in our proposal, this trade-off should be flexible so as to be able to obtain fast results with a fair accuracy, or very accurate results with a slightly higher time delay.

  Actually, both paths are not conflicted with each other and can be applied altogether to an identification system.
- **To reduce the database penetration rate of the identification search.** The use of a classification step previous to the identification would enhance the performance of the overall process by reducing the database penetration rate. The proposal should be generic enough to allow for different types of classifiers that could suit different needs. Moreover, it is very desirable to avoid the rejection of fingerprints in the feature extraction step that precedes the classification. Finally, the identification process itself should be able to draw the maximum information from the classification phase, so as to reduce the penetration rate as much as possible.

5 Methodology

This thesis requires the application of a methodology that is both theoretical and practical. Therefore, we need a strategy that, while maintaining the guidelines of the traditional scientific method, is able to provide the special needs of such methodology. In particular, the following guidelines for the research work and experiments will be applied:

1. **Observation**: detailed study of the fingerprint identification problem and its specific characteristics, as well as the possibilities offered by HPC systems and data mining techniques. Local minutiae-based fingerprint matching has to be accorded special attention, as it is the core of the performance of an automatic fingerprint identification system.

2. **Hypothesis formulation**: design of new identification methods that make use of the approaches that have been highlighted as promising to improve the performance of the identification, such as classification, preprocessing and high performance computing. The new methods should implement the characteristics described in the previously mentioned objectives to face the problem of identification in very large databases.

3. **Observation gathering**: getting the results obtained by the application of the new methods, on different types of fingerprint databases and using different types of performance measures. Both the efficiency and the accuracy have to be taken into account.

4. **Contrasting the hypothesis**: comparison of the results obtained with those published by other methods related to fingerprint identification in the specialized literature. For a fair comparison, the compared methods should be evaluated on the same hardware and the same databases, in a generic manner so as not to obtain biased conclusions.

5. **Hypothesis proof or refusal**: acceptance or rejection and modification, in due case, of the developed techniques as a consequence of the performed experiments and the gathered results. If necessary, the previous steps should be repeated to formulate new hypothesis that can be proven.

6. **Scientific thesis**: extraction, redaction and acceptance of the conclusions obtained throughout the research process. All the proposals and results gathered along the entire process should be synthesized into a memory of the thesis.

6 Summary

This section presents a summary of the proposals described in the publications associated to this thesis. Afterwards, Section 7 will show an overview of the obtained results. The research carried out for this thesis and the results obtained in each case are collected into the following published papers:

- D. Peralta, M. Galar, I. Triguero, D. Paternain, S. García, E. Barrenechea, J. M. Benítez, H. Bustince, F. Herrera, A Survey on Fingerprint Minutiae-Based Local Matching for Verification and Identification: Taxonomy and Experimental Evaluation. Information Sciences 315 (2015) 67–87, doi: 10.1016/j.ins.2015.04.013.

- D. Peralta, M. Galar, I. Triguero, O. Miguel-Hurtado, J.M. Benítez, F. Herrera. Minutiae Filtering to Improve Both Efficacy and Efficiency of Fingerprint Matching Algorithms. Engineering Applications of Artificial Intelligence, 32 (2014) 37–53. doi: 10.1016/j.engappai.2014.02.016

- D. Peralta, I. Triguero, R. Sanchez-Reillo, F. Herrera, J.M. Benítez. Fast Fingerprint Identification for Large Databases. Pattern Recognition 47:2 (2014) 588–602. doi: 10.1016/j.patcog.2013.08.002

- D. Peralta, I. Triguero, S. García, F. Herrera, J.M. Benítez. DPD-DFF: A Dual Phase Distributed Scheme with Double Fingerprint Fusion for Fast and Accurate Identification in Large Databases. Information Fusion 32 (2016) 40–51. doi: 10.1016/j.inffus.2016.03.002

- D. Peralta, I. Triguero, Y. Saeys, S. García, J.M. Benítez, F. Herrera. Clasificación Jerárquica de Huellas Dactilares con Selección de Características. VII Symposium of Theory and Applications of Data Mining (TAMIDA), CAEPIA 2015, Albacete (España), pp. 831-840, 09-12 November 2015.

Additionally, several other works are currently under different stages of development and submission in specialized journals:

- D. Peralta, S. García, J.M. Benítez, F. Herrera. Fingerprint Identification in MapReduce and Spark.
  - The aim of this work is to establish a flexible decomposition paradigm for matching algorithms, with the final goal of adapting them to big data environments such as MapReduce or Spark. Thus, the identification in very large databases would benefit from the advantages of such frameworks, such as robustness and fault-tolerance.

- D. Peralta, I. Triguero, Y. Saeys, S. García, J.M. Benítez, F. Herrera. Complete Fingerprint Identification System with Classification.
  - In this work, we present a complete identification system whose first step is a novel classification strategy that combines features from different sources to optimize the classification accuracy. The identification itself is performed in a progressive way to further improve the performance of the proposal.

- D. Peralta, I. Triguero, Y. Saeys, S. García, J.M. Benítez, F. Herrera. Deep Learning for Fingerprint Classification.

– The capabilities of deep learning approaches are explored concerning their application on the fingerprint classification problem. Different types of architectures are analyzed and tested on various types of fingerprint databases to test their behavior and highlight their weaknesses and strengths. The aim of this study is also to provide the starting point for further research.

Due to the constraints imposed by auto-plagiarism detection systems, the specific proposals and results presented in these papers cannot be included into this memory; instead, the draft version of the submitted material will be provided separately.

The remainder of this section is organized into the four objectives defined in Section 4. First, Section 6.1 summarizes the review performed on minutiae-based matching algorithms. Then, Section 6.2 shows the proposed preprocessing algorithm for minutiae filtering. Section 6.3 details the proposed approaches to improve the efficiency and efficacy of the identification process. Finally, Section 6.4 explains our proposals to take advantage of the reduction of the database penetration rate.

6.1 Review on minutiae-based local matching algorithms

Fingerprint identification has become a topic of main interest in the last decades. Automatic recognition systems have been widely used with great achievements in many practical applications. In particular, minutiae-based fingerprint matching has proven to yield a good performance for both verification and identification purposes. More than 80 different methods have been proposed so far in the specialized literature, showing the possibilities of the approach and the interest of the scientific community.

However, these methods do not follow a common design procedure and the terminology is sometimes unclear. Despite this, many of the published methods share similar or repeated parts that might even be named differently in different sources. Although there are some reviews on the topic [YA04, MMJP09, JFN10], at the time of writing this thesis there is no general categorization of matching methods and their particularities.

We have identified the main characteristics in minutiae-based local matching algorithms, including the topology of the local structures (nearest neighbors, fixed radius, texture mixed, minutiae triplets, K-plets and cylinders), the type of transformation applied in the global consolidation process (single, consensus, multiple, complex or incremental), the use of additional features (ridge frequencies, core points, local orientation and gray-scale images), the peculiarities of the minutiae (type, ridge count and ridge properties), and finally the type of parameter learning (matching score or local similarity). This thorough analysis allowed us to build a taxonomy of the methods published so far, highlighting which characteristics are common between them and how they interact.

In addition, an experimental study with the most representative matching methods of each category was carried out on several fingerprint datasets, including public datasets and one collected by the authors' research group with different sensors in a controlled environment. Multiple accuracy and runtime measures were accounted for, so as to evaluate the impact of the different characteristics of the matching on them.

The journal paper associated to this part is:

- D. Peralta, M. Galar, I. Triguero, D. Paternain, S. García, E. Barrenechea, J. M. Benítez, H. Bustince, F. Herrera, A Survey on Fingerprint Minutiae-Based Local Matching for Verification

and Identification: Taxonomy and Experimental Evaluation. Information Sciences 315 (2015) 67–87, doi: 10.1016/j.ins.2015.04.013.

6.2 Minutiae filtering

Minutiae show several properties that make them very suited to perform fingerprint recognition tasks: they are unique, universal, invariable and easy to use. As a consequence, most matching algorithms rely on minutiae to compute the similarity between two fingerprints. Minutiae extraction is therefore a key component of the fingerprint recognition process, as these matching methods often rely exclusively on the information provided by the minutiae extractor.

Although there are several minutiae extractors proposed in the literature, the problems encountered with the process are common for all of them and can be classified into two types:

- **Missing minutiae** that are not detected by the extractor and therefore cannot be used for the matching.
- **Spurious minutiae** that are erroneously detected by the extractor, introducing noise into the resulting minutiae set. Most spurious minutiae are detected on the borderline of the fingerprint.

In general, minutiae extractors tend to suffer more from the latter, so as not to omit real minutiae that might be crucial for the comparison. These problems increase the difference between captures of the same fingerprint, whilst they can also cause false similarities between captures of different fingers. They become more acute when dealing with low quality fingerprints or with large-scale identification, in which the huge amount of non-matching templates requires an accurate matching algorithm to avoid identification errors.

In this thesis we have tackled the problem of removing spurious borderline minutiae after their extraction. For this goal we have applied different strategies to extract a candidate set of spurious minutiae for a fingerprint:

- Computing the convex hull of the extracted minutiae set; all the minutiae in the convex hull are included into the candidate set.
- Using an image segmentation-based approach aimed at discerning the background and the foreground of the fingerprint image. The approach includes four steps: normalization, block-wise variance computation, thresholding and refinement. The minutiae that are detected within background areas are included into the candidate set.

Both strategies have been combined with the quality information provided by the minutiae extractor, so that the minutiae selected by the preprocessing that fall under a certain quality threshold are eliminated prior to the recognition itself. As a side objective, the influence of the spurious minutiae on several different matching methods has also been studied.

The experiments for this proposal have been carried out on multiple different databases, some of them widely used and publicly available, some others collected by the research group, and five artificially generated databases. The advantage of the latter is that, unlike for real captures, the ground-truth minutiae of a fingerprint can be known and compared with the extracted set. This allowed us to evaluate and quantify the amount of noise introduced by the minutiae extraction algorithm.

The journal paper associated to this part is:

- D. Peralta, M. Galar, I. Triguero, O. Miguel-Hurtado, J.M. Benítez, F. Herrera. Minutiae Filtering to Improve Both Efficacy and Efficiency of Fingerprint Matching Algorithms. Engineering Applications of Artificial Intelligence, 32 (2014) 37–53. doi: 10.1016/j.engappai.2014.02.016

6.3 Efficient, scalable and accurate fingerprint identification

The increasing number of applications of identification applications in many contexts, such as public institutions, police departments or forensic purposed, has led in many cases to a growth of the size of the fingerprint databases used for the identification. When dealing with large fingerprint databases, the bottleneck of the identification process is the matching algorithm because it must be applied once for each template fingerprint in the database. Although the literature offers fingerprint matching algorithms that provide a very good accuracy, they are often computationally expensive and their use becomes prohibitive in large-scale environments.

HPC has already been successfully applied to many different pattern recognition problems; a parallel computing infrastructure allows to speed up the execution of computationally heavy algorithms. Although there are some proposals in the literature that use HPC tools for fingerprint recognition, at the time of writing this thesis none of them focused on the large-scale identification problem. Rather, they tackled other problems such as high availability or database distribution.

For this thesis, we developed a parallel and distributed framework for fingerprint identification that takes advantage both of multi-computer clusters and multi-core processors with a two-level parallelism. The framework focuses on parallelizing the matching step as much as possible, while reducing the communication and synchronization needs between the different threads and processes. Moreover, the proposal can be used for any matching algorithm as it does not involve modifying the computation; the result obtained for the identification is guaranteed to be the same as that of a sequential approach.

Taking this idea one step further, a decomposition scheme for fingerprint matching algorithms has been proposed. The knowledge gathered in the review on minutiae-based matching has been applied to develop the proposal as a generic strategy that can be applied to any algorithm. The ultimate purpose of such decomposition is to facilitate the adaptation of matching algorithms to big data frameworks, so as to take advantage of all the possibilities offered by them.

The experiments performed for the developed systems involved an artificially generated database of hundreds of thousands of fingerprints, along with one of the largest public datasets available, to evaluate the speedup of the proposals under different hardware contraints.

Although the previously described HPC-based proposals do not suffer from any accuracy loss, they do not tackle the task of enhancing the accuracy of the underlying matching algorithms. The use of several fingerprints or several matching algorithms has already been proven to enhance the accuracy of the recognition process by several authors. However, there is yet no attempt to use these strategies to lower the identification time; instead, this time is increased by the successive application of the matching algorithms.

We propose an identification system (nicknamed DPD-DFF) that incorporates two different fingerprints per identity, as well as two matching algorithms. Once an input fingerprint pair is available, the identification is carried out in two phases. In the first (*fast*) phase, a fast matching algorithm is used to quickly perform a first scan of the database to select a set of candidate identities. Then, a very accurate matching algorithm is used in the *accurate* phase to select the matching identity among those in the candidate set. Each of these phases can either use a single or both

fingerprints, yielding different trade-offs between accuracy and identification time. Additionally, the size of the candidate set can be varied to finely tune that balance, so as to suit the environmental constraints and the available computational resources.

DPD-DFF has been extensively tested on several sets of publicly available databases, as well as a large artificially generated dataset. All possible combinations of single and double fingerprint were put to the test, along with two different criteria for the selection of the set of candidate identities.

The journal papers associated to this part are:

- D. Peralta, I. Triguero, R. Sanchez-Reillo, F. Herrera, J.M. Benítez. Fast Fingerprint Identification for Large Databases. Pattern Recognition 47:2 (2014) 588–602. doi: 10.1016/j.patcog.2013.08.002
- D. Peralta, I. Triguero, S. García, F. Herrera, J.M. Benítez. DPD-DFF: A Dual Phase Distributed Scheme with Double Fingerprint Fusion for Fast and Accurate Identification in Large Databases. Information Fusion 32 (2016) 40–51. doi: 10.1016/j.inffus.2016.03.002
- D. Peralta, S. García, J.M. Benítez, F. Herrera. Fingerprint Identification in MapReduce and Spark. Sometido.

6.4 Database penetration rate reduction

When dealing with sufficiently large databases, a large computing capacity might not be enough to solve the performance constraints. It can also happen that the cost of such an infrastructure could be high, which leads to the search of other solutions to allow for fast fingerprint identification. Therefore, there is a need to consider other approaches that will speed up the identification time in very large databases, in combination with HPC infrastructures.

In this line of work, the reduction of the database penetration is one of the most pursued goals. Different approaches can be followed to tackle this objective; among them, fingerprint classification is one of the most popular. Different classification procedures, each with a particular feature extraction process to encode the fingerprints, have been proposed in the literature.

Some of the feature extraction methods that lead to the best classification accuracy reject the fingerprints that do not meet certain criteria, so that no feature vector is extracted, making the classification impossible.

We proposed a classification approach that combines several feature extractors in several ways to improve the classification accuracy while eliminating the rejection rate. This, in combination with an incremental search procedure, allows to reduce the database penetration rate for the identification, while maintaining a good identification accuracy.

Several experiments were performed to assess the quality of the proposal and evaluate the behavior of the subsequent identification search, following the guidelines established in the previous review. The experiments used a large synthetically generated database as well as the well-known NIST-SD14 public dataset.

The research papers associated to this part is:

- D. Peralta, I. Triguero, Y. Saeys, S. García, J.M. Benítez, F. Herrera. Clasificación Jerárquica de Huellas Dactilares con Selección de Características. VII Symposium of Theory and Appli-

cations of Data Mining (TAMIDA), CAEPIA 2015, Albacete (España), pp. 831-840, 09-12 November 2015.

- D. Peralta, I. Triguero, Y. Saeys, S. García, J.M. Benítez, F. Herrera. Complete Fingerprint Identification System with Classification. Sometido.
- D. Peralta, I. Triguero, Y. Saeys, S. García, J.M. Benítez, F. Herrera. Deep Learning for Fingerprint Classification. Sometido.

7 Discussion of results

The following subsections summarize and discuss the results obtained in each specific stage of the thesis.

7.1 Review on minutiae-based local matching algorithms

Both classic and recent minutiae-based fingerprint matching algorithms have been thoroughly analyzed within the published review. The different characteristics that define them have been pointed out and served as a basis to define a taxonomy of matching methods, which allows establish guidelines for further work.

The extensive experimental study carried out has compared the accuracy and runtime of the main matching algorithms of each family described in the taxonomy. The obtained results have been further analyzed with statistical tests. The results of the comparison have highlighted the potential of cylinder-based approaches when accuracy is at stake. Also, some simple nearest neighbors approaches with relatively simple consolidations can obtain a decent accuracy within a very fast computing time. In general, no single approach is preferred above all others: the suitability of a matching method will depend on the specific requirements of the application. However, the results obtained in this review allow to reduce the breadth of the decision and sets the guidelines to reach a better decision.

Additionally, all the results obtained are publicly available at `http://sci2s.ugr.es/MatchingReview` for their download.

7.2 Minutiae filtering

We have proposed a preprocessing algorithm for fingerprint matching that tackles the problem of the spurious minutiae detection. The proposal implements two different strategies (the convex hull and a new segmentation method) to delimit the borders of the fingerprint, where most spurious minutiae are located. Then, a threshold based on the quality provided by the minutiae extractor is applied to filter the minutiae.

Twelve different databases have been used to perform experiments; one of them was captured by the authors' research groups, six are public and five were artificially generated, allowing the analysis of the difference between the ground-truth minutiae sets and the corresponding extracted minutiae sets. The experiments were carried out with four different matching algorithms to assess the robustness of the proposal. The results show that the proposed filters allowed us to reduce the number of spurious minutiae without damaging the correctly detected ones. This lead to an increase of the accuracy of the applied matching algorithms, along with a reduction of their runtime due to the lower number of minutiae. The accuracy improvement reached up to 2% for good quality databases and 25% for the public FVC. The runtime reduction attained up to 60%.

7.3 Efficient, scalable and accurate fingerprint identification

We have designed two types of parallel frameworks to allow for the full scalability of fingerprint identification systems.

The first proposal is based on Message Passing Interface (MPI), and describes a two-level parallel and distributed framework that is generic for any fingerprint matching algorithm. The

system ensures that there is no loss in the accuracy with respect to a sequential approach.

The extensive experiments carried out on databases of different sizes and different volumes of hardware resources show that the proposal offers a near-linear scalability for fingerprint identification. For instance, a database of 400 000 fingerprints was explored in 0.5s. Moreover, the availability of more RAM memory when using several computers allows to maintain larger amounts of preprocessed fingerprints in main memory, accelerating the identification process. The associated software is publicly available (`https://github.com/dperaltac/mpi-afis`). Our results have been used by other authors as a baseline for the evaluation of their own proposals, assessing a good acceptation in the scientific community. The results of the experiments and details on the implementation can be found at `http://sci2s.ugr.es/ParallelMatching`.

The second proposal is a generic fingerprint matching decomposition methodology that enables the use of big data paradigms (such as MapReduce or Spark) to enhance the identification speed and scalability. The proposed decomposition has been applied to two different well-known matchers.

The Hadoop and Spark implementations of these algorithms were compared in the experimental study, highlighting the different characteristics provided by each big data framework. Additionally, the proposal was compared with those of the parallel framework previously described. The results showed that the execution times with Spark were even lower that those obtained with MPI, assessing the good behavior of the proposed decomposition, which allows to avoid part of the computation by early dropping of non promising local structure matches.

We proposed a flexible identification system, DPD-DFF, that involves two fingers per identity and two matching algorithms. First, the database is quickly explored with a fast matching algorithm to extract a set of candidate identities; then, an accurate matcher is applied to select the most similar match in the candidate set. Both phases can use either one or two fingerprints, providing several variant of the algorithm that allow different accuracy-efficiency trade-offs. Additionally, the size of the candidate set can be controlled by several criteria, enabling a fine-grain tuning of that trade-off.

The resulting software is publicly available at `https://github.com/dperaltac/mpi-afis`. A thorough experimentation was carried out to verify the behavior of the proposal, involving many public databases, a large artificially generated database, and a database composed of fingerprint pairs gathered among more than 300 people specifically for this study, using a sensor that captures two fingerprints at a time.

The results obtained showed that DPD-DFF outperforms traditional, single fngerprint systems both in terms of identification accuracy and identification time. Moreover, the tested double fingerprint and double matcher approaches that conform the state-of-the-art in the topic also were outperformed by our approach. Finally, an additional study carried out using impostor fingerprints showed an outstanding performance for the early detection of such cases, obtaining very high True Negative Rate and True Positive Rate altogether. The complete results obtained for this study can be consulted at `http://sci2s.ugr.es/DPDDFF`.

7.4 Database penetration rate reduction

The reduction of the penetration rate has been tackled by following a multi-level classification approach, combining several feature extractors in a way that avoids rejecting any fingerprint while taking advantage of the extractors that do reject those that do not meet their quality requirements. The subsequent identification process, which is carried out in an incremental manner, has also been

considered in the approach so as to provide a system as a whole.

The experiments, involving a large database and a public database of rolled fingerprints, showed that the approach is able to outperform the state-of-the-art classifiers in terms of classification accuracy, with the additional advantage of not rejecting any fingerprint. Furthermore, the experiments involving the identification process revealed that the reduction of the penetration rate is very close to the theoretical maximum that can be obtained with the 5-class approach (around 70%), while maintaining a high identification accuracy.

8 Concluding Remarks

In this thesis, we have addressed the problem of fingerprint identification in large databases, with the aim of analyzing, designing and implementing different strategies for efficient, scalable and accurate identification systems.

The initial objective for the thesis was to gain a deep understanding of the field, especially regarding minutiae-based local matching algorithms, which constitute the bottleneck of any large-scale identification approach. To do so, we have carried out a theoretical and empirical survey of the main methods proposed in the literature, focusing on extracting the characteristics that are common between them and those that are unique to each approach. The results of this study constituted the ground basis for the subsequent research during this thesis.

The second objective tackled was the improvement of both the accuracy and the runtime of matching algorithms by a preprocessing algorithm to filter minutiae. We have revealed the impact of spurious minutiae on the performance of several different matchers, and we have proposed two different schemes to carry out their filtering. The results showed improvements in both accuracy and runtime, assessing the value of the proposal.

The parallelization of the matching process has been carried out through two different proposals. On the one hand, a two-level parallel scheme has been proposed, which runs processes in different machines as well as several threads within each process. The experiments performed using the proposal revealed near-linear speedup independent of the underlying matching algorithm. Moreover, the speedup was super-linear when the most complex database was used, showing the benefits of the synchronized work of several machines. On the other hand, a methodology for the decomposition of matching algorithms (based on the knowledge gathered for the review) was designed and applied to some of the most relevant algorithms of the state-of-the-art. The aim of such decomposition is to provide a generic methodology to implement any matching algorithm within a big data framework, thus taking advantage of their benefits. The decomposition allows for an early detection of poorly-matching fingerprints, so as to reduce the computational load. The results of the experiments outperformed those obtained with the two-level parallel approach, revealing the extreme scalability capabilities of the decomposition scheme.

In combination with the development of such scalable system, we have applied information fusion strategies to reduce both the identification time and the identification error. In this line of work, we have proposed an identification system that fuses two fingerprints and two matching algorithms in a dual-phase scheme, which provides a flexible trade-off between accuracy and efficiency by means of a single parameter. The proposal dominated the approaches of the state-of-the-art in both goodness measures.

Finally, another strategy to improve the identification time involved fingerprint classification. To do so, we combined different feature extractors in a multi-level manner to maximize the classification accuracy and eliminate the fingerprint rejection of the feature extractors. The accuracy of the resulting classifier outperformed that of all the compared methods. The further tests carried out involving the entire identification process assessed the good performance of the approach in terms of the reduction of the database penetration rate.

Conclusiones

En esta tesis se ha abordado el problema de la identificación de huellas dactilares en grandes bases de datos, con el objetivo de analizar, diseñar e implementar distintas estrategias para sistemas de identificación eficientes, escalables y precisos.

El objetivo inicial de la tesis era obtener un profundo conocimiento del área, especialmente en cuanto a algoritmos de *matching* locales basados en minucias, que constituyen el cuello de botella de cualquier sistema de identificación a gran escala. Para ello, se ha llevado a cabo un estudio teórico y empírico de los principales métodos propuestos en la literatura, centrado en extraer las características que son más comunes entre ellos y las que son únicas para cada enfoque. Los resultados de este estudio constituyen la base para la investigación realizada durante esta tesis.

El segundo objetivo fue la mejora de la precisión y el tiempo de ejecución de los algoritmos de *matching* mediante un algoritmo de preprocesamiento para el filtrado de minucias. Se ha revelado el impacto de las minucias espurias sobre distintos algoritmos, y se han propuesto dos estrategias diferentes para su filtrado. Los resultados muestran mejoras tanto en precisión como en tiempo, certificando la valía de la propuesta.

La paralelización del proceso de matching se ha llevado a cabo mediante dos propuestas diferentes. Por una parte, se ha propuesto un esquema paralelo a dos niveles que lanza procesos en distintas máquinas, y varias hebras en cada proceso. Los experimentos revelaron una aceleración super-lineal al utilizar la base de datos más compleja, mostrando los beneficios del trabajo sincronizado de varias máquinas. Por otra parte, se ha diseñado una metodología para la descomposición de algoritmos de *matching* (basada en el conocimiento adquirido para la revisión previamente mencionada), y se ha aplicado sobre algunos de los algoritmos más relevantes del ámbito. El objetivo de tal descomposición es proporcionar una metodología genérica para implementar cualquier algoritmo de *matching* en una plataforma de *big data*, aprovechando las ventajas que proporciona. La descomposición permite una pronta detección de huellas con escaso emparejamiento, para reducir la carga computacional. Los resultados de los experimentos mejoraron los obtenidos con el enfoque paralelo a dos niveles, revelando las extremas capacidades de escalabilidad del método de descomposición.

En combinación con el desarrollo de estos sistemas escalables, se han aplicado estrategias de fusión de información para reducir tanto el tiempo como el error de identificación. En esta línea de trabajo, se ha propuesto un sistema de identificación que fusiona dos huellas y dos algoritmos de *matching* en un esquema de dos fases, proporcionando un equilibrio flexible entre precisión y eficiencia con un único parámetro. La propuesta dominó a las del estado del arte para las dos medidas estudiadas.

Finalmente, otra estrategia para mejorar el tiempo de identificación implicó la clasificación de huellas. Para ello, se han combinado distintos extractores de características de forma jerárquica para maximizar el acierto de la clasificación y eliminar el rechazo de las huellas producido por los extractores. La precisión del clasificador resultante mejoró la de todos los métodos comparados. Las pruebas llevadas a cabo posteriormente, abarcando el proceso completo de identificación, ratificaron el buen rendimiento de la propuesta en términos de reducción de la tasa de penetración en la base de datos.

9 Future Work

The work carried out during this thesis has highlighted new promising research lines, either to further enhance the performance of the models proposed, or to apply them to new challenging problems.

Applying deep learning techniques to enhance the fingerprint recognition process: In the last few years, deep neural networks [LBH15] have arisen as a very powerful way to solve many complex real-world problems, in particular those based on images [KSH+12]. Their use on several parts of the fingerprint recognition process is therefore very promising.

Deep learning has been proven to provide outstanding performances in classification problems. Therefore, their use to improve the fingerprint classification task is of great interest.

Deep neural networks can also be applied in an unsupervised manner to extract patterns that intrinsically define a given input [LGRN09]. From this perspective, deep learning can be used as a feature extractor to obtain relevant information from the fingerprints.

New big data approaches: Big data is expected to be one of the main challenges for data mining in the near future [FdRL+14]. The huge quantities of data that are available in many fields offer a two-fold field of research. On the one hand, there is a need of systems that are able to deal with such large amounts of data in an efficient and scalable manner. On the other hand, all that data can provide new knowledge and information to solve new problems.

In the Biometrics field, the increasing needs to identify people is leading to ever larger databases of different biometric features. The use of big data approaches to evaluate all that information could lead to a better understanding of the problem.

Multi-modal biometrics: One of the natural solutions that have arisen for biometrics is the hybridization of several biometric features [RNJ06]. By combining the information from several sources such as fingerprint, face, iris, etc., it is possible to increase the accuracy and reliability of the recognition, as well as to provide more robust systems in case of injuries or amputations. However, the identification time is also increased in a similar manner to what happens with multi-finger or multi-matcher methods.

Therefore, there is also a need to adapt multi-modal biometric approaches to a large-scale point of view to deal with their increased scalability and identification times issues.

Fingerprint identification on co-processors: Although most high performance computing systems rely on clusters of multi-core processors, during the last years there has been a vast effort to implement computationally heavy tasks in GPU support with frameworks such as NVIDIA CUDA [cud]. GPUs have already very recently successfully used for fingerprint identification systems with outstanding performance [GLHB14, CFM15, LCG+15].

In addition to GPUs, other co-processors have been developed to give support to computationally intensive algorithms. One example is the Intel Xeon Phi [JR13], which includes several dozens of cores and local memory to execute highly parallel tasks. Unlike GPUs, such co-processors have an instruction set that is closer to that of a general purpose processor, which makes them

easier to use for general purpose applications. Therefore, the study of the impact that this kind of hardware can have on the performance of fingerprint identification systems would be of interest.

Hybrid architectures: The previous line of future work would also arise the possibility of developing more complex identification systems that put together multiple types of hardware to obtain a maximum performance. A single server hosting several CPUs, GPUs and co-processors could provide a very interesting performance for a reasonably low price and above all low maintenance costs, as there is no need for inter-connecting networks or shared resources. An approach of this kind could be of great interest for small or medium sized corporations.

Chapter II

Publications: Published Papers

1 A Survey on Fingerprint Minutiae-Based Local Matching for Verification and Identification: Taxonomy and Experimental Evaluation

- D. Peralta, M. Galar, I. Triguero, D. Paternain, S. García, E. Barrenechea, J. M. Benítez, H. Bustince, F. Herrera, A Survey on Fingerprint Minutiae-Based Local Matching for Verification and Identification: Taxonomy and Experimental Evaluation. Information Sciences 315 (2015) 67–87, doi: 10.1016/j.ins.2015.04.013.
  - Status: **Published**.
  - Impact Factor (JCR 2015): 3.364
  - Subject Category: Computer Science, Information Systems. Ranking 8 / 143 (**Q1**).

A Survey on Fingerprint Minutiae-based Local Matching for Verification and Identification: Taxonomy and Experimental Evaluation

Daniel Peralta[a], Mikel Galar[c], Isaac Triguero[d,a], Daniel Paternain[c], Salvador García[a,b,*], Edurne Barrenechea[c], José M. Benítez[a], Humberto Bustince[c], Francisco Herrera[a]

[a]*Dept. of Computer Science and Artificial Intelligence. University of Granada, 18071 Granada, Spain*
[b]*Faculty of Computing and Information Technology - North Jeddah, King Abdulaziz University, 21589, Jeddah, Saudi Arabia*
[c]*Departamento de Automática y Computación, Universidad Pública de Navarra, Pamplona, Spain*
[d]*Inflammation Research Center, a VIB-UGent Dept. UGent Dept. of Internal Medicine, Respiratory Medicine (GE01) Technologiepark 927, B-9052 Zwijnaarde, Belgium.*

Abstract

Fingerprint recognition has found a reliable application for verification or identification of people in biometrics. Globally, fingerprints can be viewed as valuable traits due to several perceptions observed by the experts; such as the distinctiveness and the permanence on humans and the performance in real applications. Among the main stages of fingerprint recognition, the automated matching phase has received much attention from the early years up to nowadays. This paper is devoted to review and categorize the vast number of fingerprint matching methods proposed in the specialized literature. In particular, we focus on local minutiae-based matching algorithms, which provide good performance with an excellent trade-off between efficacy and efficiency. We identify the main properties and differences of existing methods. Then, we include an experimental evaluation involving the most representative local minutiae-based matching models in both verification and evaluation tasks. The results obtained will be discussed in detail, supporting the description of future directions.

Keywords: Biometrics, fingerprint verification, fingerprint identification, local matching, minutiae.

*Corresponding author. Tel.: +34 958 240598; fax: +34 958 243317.
Email addresses: dperalta@decsai.ugr.es (Daniel Peralta), mikel.galar@unavarra.es (Mikel Galar), Isaac.Triguero@irc.vib-UGent.be (Isaac Triguero), daniel.paternain@unavarra.es (Daniel Paternain), salvagl@decsai.ugr.es (Salvador García), edurne.barrenechea@unavarra.es (Edurne Barrenechea), J.M.Benitez@decsai.ugr.es (José M. Benítez), bustince@unavarra.es (Humberto Bustince), herrera@decsai.ugr.es (Francisco Herrera)

1. Introduction

Automatic fingerprint recognition has been one of the most known and used biometric authentication systems during the last decades. It has been used for personal verification and identification with great achievements [76]. A vast number of applications incorporate fingerprint recognition as basics, such as forensics, building accessing, ATM authentication or secure payment [113]. There are some other human characteristics that can be used as traits of a biometric system, such as the person's face, the retina or iris [16], the voice, etc. There is no trait that highlights as the best one. However, on average, fingerprints offer good capabilities in all properties analyzed by the experts and excellent results in distinctiveness [126], permanence and global performance [113]. Although the recognition is not as accurate as with other traits, it provides a good balance between accuracy, speed, resource requirements and robustness.

Independent of the type of task, either verification [72] (one-to-one comparison) or identification (search for an input fingerprint in a database) [80], it is necessary to perform a sequence of operations to build a template database and later use the system. Assuming that there is a database and that proper enrollments have been previously taken, the order of the operations for both tasks is given by: a capture of the fingerprint, a feature extraction stage, a matching and a pre-selection or filtering [85] (which is associated to identification tasks only). The capture of the fingerprint obtains an image that is not usually stored as such in the database. Instead, a feature extraction process is applied to obtain up to three levels of features [60]: level 1 features provide, at the global level, information of singular points and ridge line flow or orientation; level 2 features, at a local level, refer to minutiae details which usually correspond to bifurcations and ridge endings; and level 3 features, at the very-fine level, include features inside the ridges such as width, shape, curvature, dots, etc. These features are only observable in high resolution images.

Once a set of features is extracted from the fingerprint image, the final goal is to find (or confirm) the identity of a person whose fingerprint has been previously enrolled into the system. The matching mechanism is the responsible to provide a likeliness score between two fingerprints. Most of the efforts in matching are with the use of minutiae details, although there are other types of matching methods based on correlations of images, other types of features and even on level 3 features. Minutiae matching consists of finding the alignment between two templates that results in the maximum number of minutiae pairings. Furthermore, minutiae matching can be classified as local or global [81], aligned or not [189], etc; all the categories will be detailed in this paper.

Many fingerprint matching algorithms have been proposed in the literature, and the operations with features they use are sometimes similar or even repeated. In spite of the existence of some reviews on the topic, such as [174, 113, 71], they are not explicitly focused on matching and the characteristics of the methods are not completely studied or categorized. This issue may lead to a lack of unification and even to propose very similar matching methods in the future. Moreover, there are few attempts to empirically compare them.

In this sense, the motivation of this paper can be segregated into three main objectives:

- To gather and briefly describe all the matching methods proposed in the specialized

literature.

- To offer an entire taxonomy based on the main processes and properties observed in the matching methods. It allows us to understand the reasons to choose the most suitable matching algorithm depending on the circumstances.
- To conduct an empirical study analyzing the most important local minutiae-based matching algorithms in terms of accuracy and speed throughput when they are applied to both verification and identification tasks.

The rest of this paper is organized as follows. Section 2 provides the necessary background in fingerprint minutiae matching. In Section 3, we introduce the main properties and the taxonomy for the matching methods. Next, Section 4 overviews the current trends in fingerprint matching. In Section 5, experiments on several data sets compare some of the most important local minutiae-based matching methods. Finally, Section 6 concludes the paper, including some original opinions for instruction in theory and application and future research directions. Additional material to the paper can be found at `http://sci2s.ugr.es/MatchingReview/`.

2. Background in Fingerprint Minutiae Matching

Fingerprint matching is a crucial step in both verification and identification problems. Roughly, a fingerprint matching algorithm compares two fingerprints and returns either a degree of similarity (a real number bounded into an interval) or a dichotomic output (matched or non-matched). Hereafter, we use the representation of the fingerprint acquired by enrollment as the template (T) and the representation of the input fingerprint (I). Two fingerprints are called *genuine* if they represent the same finger, and *impostor* when they are different.

Several factors make fingerprint matching a very challenging problem [113]: image noise, skin condition, distortions, rotations, displacement, etc. There are two well-known properties in fingerprints: large variability in different impressions of the same finger (large *intra-class* variations) and much similarity between two images from different fingers (small *interclass* variations).

The most popular and used technique is the minutiae-based matching. Subsequent subsections will detail the main concepts of minutiae-based matching (Subsection 2.1), including the distinction between global and local matching (Subsection 2.2) and feature extraction techniques that are commonly used to obtain the minutiae for matching (Subsection 2.3).

2.1. Minutiae-based Matching

The output of a minutiae extraction stage is, at least, a set of minutiae. Each minutia is represented by its location coordinates and orientation angles, forming a 3-tuple $M = (x, y, \theta)$. T and I fingerprints have m and n minutiae, respectively. A minutia M_j in I is considered matched with a minutia M_i in T when it falls within the tolerance box of M_i. The tolerance box is defined as the maximum spatial distance and direction difference

permitted to compensate unavoidable errors made by minutiae extractors and positioning changes produced by distortions.

Obviously, it is mandatory to obtain the optimal displacement and rotation alignment of fingerprints in order to maximize the number of minutiae matched. This also includes scaling and advanced geometrical transformations. After alignment, a matching score for the two fingerprints is calculated. To do this, the pairing function between minutiae M_i and M_j must be found, assuming that each minutia has either exactly one matched minutia in the other fingerprint or has none at all. Achieving the optimal pairing is not a trivial task when the correct alignment is not known, as it usually happens in practice. For instance, a minutia of I may fall within the tolerance box of two or more minutiae of T. An assignment algorithm, preferably fast or greedy, is usually employed for this task.

Finally, the matching score could be formulated as follows:

$$matching\_score = \frac{k}{(n+m)/2}$$

where k is the number of matched minutiae. It is a simple expression usually shared among matching algorithms. However, advanced models normally exploit further information such as the minutiae quality and adjusted parameters by using optimization techniques.

2.2. Global and Local Minutiae Matching

Fingerprint minutiae matching can be firstly divided into two families of methods:

- Global minutiae matching: the algorithms of this kind tackle the alignment process by taking into consideration all the minutiae as a whole set in a global manner. Since the number of components to be aligned are, at least, three (two directions and the angle), they may require high computational resources and often the usage of a pre-alignment stage that is based on other features extracted such as singular points or orientation maps.

- Local minutiae matching: they consist of comparing two fingerprints according to local structures of minutiae. These structures are formed by considering different relationships based on proximity between closer minutiae. They are characterized by properties that are invariant regarding global transformations, such as translations and rotations. Thus, they do not take into account global relationships and allow to make matching with partial information.

The benefits of local minutiae matching are simplicity, low computational complexity and distortion tolerance, whereas global minutiae matching techniques lead to high distinctiveness. However, all of these benefits could be achieved by using hybrid strategies that perform a local minutiae matching followed by a consolidation stage. The former step determines pairs of minutiae that locally match and extracts a subset of candidate alignments for I and T. The latter step, which is not strictly mandatory, is aimed at checking the degree in which local matches support global matching.

Table 1: Enumeration of representative global minutiae matching algorithms

References	Main Property
[138, 101]	Hough transform-based approaches
[72, 107, 37]	Ridge-based relative pre-alignment
[47, 189]	Global matching of clusters of minutiae
[157, 11, 28, 163]	Algebraic geometry-based approaches
[30, 83]	Singularity-based relative pre-alignment
[140, 98, 118]	Warping modeling-based approaches
[120]	Minutiae matching with tesselated local information
[161]	Global minutiae matching with image correlation
[56, 104, 175, 82]	Orientation image-based relative pre-alignment
[151, 145, 144]	Global matching by evolutionary algorithms
[78, 92]	Weighted global matching with adjustment of scores
[32, 160]	Hierarchical and/or multilevel minutiae matching

Recently, most of the proposals of fingerprint minutiae matching designed to be implemented in real systems have given up the idea of global matching in favor of local matching. Nevertheless, although the focus of this paper is to review the properties and methods belonging to local minutiae matching, we also provide an enumeration of the most influential global minutiae matching methods proposed in the specialized literature (see Table 1).

2.3. Feature Extraction Techniques

This section is devoted to briefly identify the subset of feature extraction techniques frequently used in conjunction with fingerprint minutiae matching. It is worth mentioning that an exhaustive review of existing techniques can be found in [113]. Next, we will summarize the most representative algorithms according to their usage in practice and in subsequent matching approaches proposed in the literature:

- Fingerprint segmentation [108, 34].
- Local orientation map estimation [125, 137, 4].
- Local ridge frequencies estimation [65, 109].
- Singular and core points searching [85, 74, 139, 86].
- Alignment of local orientations and ridge frequencies [27].
- Fingerprint binarization [125, 65].
- Fingerprint skeletonization [180, 58, 106].
- Minutiae extraction [1, 108].

- Spurious minutiae removal [153, 12, 184, 95, 129].

3. Local Minutiae Matching: Properties, Methods and Taxonomy

In the following, we present the taxonomy of minutiae-based local matching methods and the properties used to build it. First, in Subsection 3.1, the essential characteristics, which will define the categories of the taxonomy, will be outlined. Next, in Subsection 3.2, we will enumerate all the minutiae-based local matching methods proposed in the scientific literature. Then, each method will be categorized according to the studied properties to provide a comprehensive taxonomy.

3.1. Properties for Categorizing Local Matching

This subsection provides a framework for the organization of the matching methods that will be presented in Subsection 3.2. The aspects discussed here include (1) topology of local structure, (2) type of consolidation, (3) usage of additional features, (4) minutiae peculiarities and (5) parameter learning. These mentioned facets are involved in the definition of the taxonomy, because they determine the way of operation of each matching technique.

3.1.1. Topology of local structure

Local matching is based on the computation of the similarity between local regions of two fingerprints, for the sake of achieving the desired invariance regarding translations and rotations. In minutiae matching, regions are associated with subsets of minutiae that present some kind of relationship, mainly based on location and proximity. Hence, the subsets of minutiae are organized into local structures and they can be built under different assumptions:

- Nearest Neighbors (**NN**): local structures are formed by a central minutia and a certain number of its nearest neighbor minutiae. The number of neighbors is specified as an input parameter and the local structures are usually defined by distances, directions and angles between pairs of minutiae.

- Fixed **Radius**: it creates a local structure from a central minutia by using a maximum distance (d_{max}) in the graph (V_i, E_i) defined as: (1) a set of vertices V_i containing all the minutiae whose spatial distance is less than or equal to d_{max} and, (2) a set of edges E_i connecting the central minutia and every vertex in V_i. The distance d_{max} is specified as an input parameter and the local structures are defined by the set of edges in clockwise traversing, by using distances as well as absolute and relative angles.

- **Texture** mixed: a local structure is defined as a feature vector that contains proper information extracted from the minutia and other types of information coming from additional features extracted from the fingerprint image, such as local orientation, ridge frequency, gray-scale image properties or sampling of equidistant points following the ridge starting from the minutia, from neighbor ridges or organized in a circular pattern around a central minutia. This aspect is closely related to the use of additional features

(third property described in this subsection), which indicates the source of the extra information used in the local structure. Also, if the matcher has the *Ridge Properties* (within the *Peculiarities in Minutia* aspect), activated, this is a symptom of using the aforementioned sampling.

- Minutiae **Triplets**: firstly used for indexing approaches, they are also interesting to yield local structures. Triplets may be built by some type of triangulation or by using all possible combinations of triplets in local regions. The local structures use information regarding angles of the vertices, length of the sides and some triangle properties such as direction, orientation, etc.

- **K-Plet**: it is an extension of the NN local based structure where it is ensured that the nearest neighbors minutiae are equally distributed in the four quadrants around the minutia.

- Minutia **Cylinder**: as an extension of fixed radius local structures, it allows a fixed length invariant coding for each minutia based on a discretization of a cuboid into cells. The cylinder is set up by using the radius as the base and the direction difference between minutiae as the height. It also allows binary representation of local structures for fast matching.

3.1.2. Type of consolidation

Although the partial scores obtained from the comparison of local structures could straightaway get a final matching score, it is common to develop a further consolidation stage in order to check whether the local similarity is supported at the global level or not. It adds an extra stage to evaluate the coherence among spatial relationships taking the local structures as basic elements. It is very useful in some cases, in which local structures could match in fingerprints from different fingers, independent of the fingerprint region that they represent. Different consolidation techniques have been proposed and can be easily isolated from the rest of the properties studied in this section:

- **Single** transformation: it is the simplest consolidation idea, based on the alignment of T and I by using the best transformation resulting from a local structure matching. A common procedure is to estimate a very limited number of pairs of local structures that received the highest matching scores and then to use the translation and rotation obtained from them to carry out a global alignment for the remaining minutiae.

- **Consensus** of transformations: it tries to evaluate to what extent each transformation obtained from a local structure matching is consistent with the others. Another manner is to assess the maximum number of consistent individual transformations. There are different approaches to calculate this estimator, although the most common one is to check that a subset of the most similar local structures remains consistent.

- **Multiple** transformations: due to the fact that the best transformation coming from the most similar local structures is not the best transformation at the global level,

multiple transformations may be used by: (1) selecting the final transformation according to the highest score achieved in the final pairing stage, (2) restricting the global matching to regions adjacent to each reference pair, or (3) fusing the results of multiple registrations.

- **Complex** transformation: this group includes transformations which are based on complex models to alleviate deformations and plastic distortions. For instance, there are models that apply a thin-plate spline to represent elastic deformations, or use the Parzen window to estimate the probability density.

- **Incremental** consolidation: when arranging the local structures into a graph, connecting the minutiae by the edges, the matching can be performed trough a dual graph traversal algorithm in a breadth-first fashion. At the end of the route, the algorithm returns the number of matched nodes. This process is repeated for every pair of minutiae and the best solution is finally chosen.

3.1.3. Use of additional features

We call as additional features those cases in which local structures also incorporate information gathered from other external sources. They may come from other feature extraction processes such as the local orientation image or the local ridge frequency estimation. Once again, we would like to emphasize that the additional features must be external with respect to the minutiae extraction algorithm. Thus, these additional features can cooperate with the mandatory features associated to minutiae (minutiae position and direction) defined by standards like ISO/IEC 19794-2. The external additional features used are the following:

- Ridges Frequency (**RF**): a local ridges frequency represents the local average pixel distance between ridges. It can be used either as a local feature associated to a certain region (or minutia) of the fingerprint image, when it is relativized with respect to the global ridges frequency of the fingerprint, or to normalize distances between two minutiae as a method of palliating the effect of distortion.

- **Core** points: the locations and orientations of core singularities are extracted from the fingerprint images for supporting the decision made by the local matching. For instance, they could be used to perform a relative pre-alignment, discarding those minutiae that are far from the original directions, or to involve only those minutiae that are close to them.

- Local Orientation (**LO**): locally, a fingerprint has a well-defined orientation field given by the ridge direction in a certain region of the image. In order to estimate it, it is normal to define a window size (ranging from 8×8 to 16×16) in order to quantize the average direction into 8 or 16 angles. The local orientation is then a number associated to a region of the fingerprint and it can be also associated to a central minutia of a local structure.

- Gray-Scale Images (**GSI**): they include texture information such as regions of gray-scale fingerprint images enhanced by filters, derived from variances among pixels, obtained by Gabor expansion or FingerCode textures [75].

3.1.4. Peculiarities in minutiae

Unlike the previous property, we define as a peculiarity in minutiae the additional information closely related to the minutia that can be extracted by using an advanced minutiae extractor. They are considered as supplementary features, different of position and direction, directly obtained from the minutiae set and being essential for the performance of a concrete matching technique. In what follows we present the most important ones:

- **Types** of minutia: one of the most common peculiarities required by many matchers is the type of minutia, dividing them into two classical types: bifurcations and ridge ends.

- Ridge Count (**RC**): this peculiarity is associated to each central minutia of the local structure and represents the number of ridges that are cut across the line joining two minutiae. The minutiae extractor requires access to the binarized or skeletonized fingerprint image to be computed.

- Ridge Properties (**RP**): the ridge which the minutia belongs to is analyzed in terms of its degree of curvature or by sampling some equidistant points along the curve to form relationships with respect to the central minutia. Here, the minutiae extractor requires to explore the skeletonized fingerprint image to walk through the ridges.

3.1.5. Parameter learning

Finally, with the term of parameter learning we refer to the application of machine learning based techniques to optimize the separation between genuine and impostor fingerprints. They are usually employed in the optimization of the similarity score that determines the final decision. The parameters typically involved in the learning process are the weights associated to the contribution of each pair of matched minutiae to the computation of the final score. This and other forms of parameter learning are the following:

- Matching Score (**MS**): a function receiving as input the feature vectors that represent two local structures and obtaining as output the similarity score is learned by means of neural networks or other regression schemes. The learning process is supervised and it is focused on optimizing the final matching score between genuine or impostor fingerprints.

- Local Similarity (**LS**): an off-line learning process is performed to learn the genuine similarity between local structures or to adjust the contribution weights associated to each component of the feature vector.

3.2. Taxonomy of Minutiae-Based Local Matching Methods

Nowadays, more than 80 minutiae-based local matching methods have been proposed in the specialized literature. This section is focused on enumerating and categorizing them according to the properties studied before. Table 2 presents an enumeration of the methods reviewed in this paper. In this field, the authors do not usually give a name for their proposal, with few exceptions. Thus, we will use the reference of the paper as their identifier.

As we can see in Table 2, the most common proposals use the *Texture* based topology, being the main baseline method the one proposed in [154]. Regarding other topologies, almost all the NN and *Radius* approaches provide from the matchers [81] and [136]. Referring to consolidation and the additional features, we can observe that all categories are spread over all methods without a clear norm. The access to the RP is more common in recent methods. Moreover, the RC and the use of the *Types* of minutiae are in decline in recent years, due to their lack of uniformity in different prints obtained from the same finger. Finally, few techniques require the use of parameter learning.

4. Related and Current Work on Matching

Once we have provided a comprehensive review on minutiae-based fingerprint matching methods, it is meaningful to also provide other kinds of procedures using for matching. They can be seen as related techniques that could be connected with matching, and current work in other ways of improving matching in different application areas. In this sense, this section gathers the most relevant developments in different issues (Subsection 4.1), distinguishing among correlation-based matching techniques (Subsection 4.2), indexing algorithms and advanced progresses in matching (Subsection 4.3).

4.1. Correlation-based Techniques and Matching without Minutiae

Generically, matching by correlation of images occurs when two fingerprint images are superimposed and their similarity is computed through the correlation between corresponding pixels for different alignments. However, this apparently simple operation rarely leads to acceptable results, mainly due to undesirable changes of global structure and brightness and contrast of the image, both depending on distortions and skin condition. Moreover, this process may involve high computational costs.

In the specialized literature, there are various alternatives coped to palliate some of the problems associated with correlation-based matching. For example, to alleviate the distortion problem, some proposals use local windows around the minutiae [90], singular points alignment before correlation [124] or advanced correlation filters [159]. To reduce the computational complexity, the correlation is performed in local regions in the Fourier domain [168], or using the Fourier-Mellin transform to maintain rotation and translation invariance [149, 84], the symmetric phase only filter to reduce noise [66] and the curvelet transform [57]. Recently, there is a promising trend that transforms minutiae positions and orientations to spectral representations in fixed-length feature vectors invariant to translations, rotations and scale. They are suitable to be reduced by dimensionality reduction techniques to speed up the matching process [171, 121].

Table 2: Enumeration and classification of minutiae-based local matching methods

References	Local Structure Topology	Type of Consolidation	Additional Features	Minutiae Peculiarities	Parameter Learning
[165]	NN	Incremental	None	None	None
[81, 6]	NN	Single	None	Types + RC	None
[136]	Radius	Consensus	None	RC	None
[94]	Texture	Multiple	RF	Types	None
[181]	NN	Single	Core	None	None
[5]	NN	Complex	None	None	None
[63]	Texture	Single	LO	Types + RP	None
[141]	Not defined	Not defined	RF + GSI	Not defined	None
[150]	Triplets	None	None	RC	None
[154, 166, 117]	Texture	Multiple	LO	None	None
[29]	NN	Single	None	Types + RP	None
[123, 182]	Texture	Multiple	LO	RP	None
[127, 116]	Triplets	Multiple	None	None	None
[142]	Texture	Multiple	LO	Types	None
[170]	Radius	Multiple	None	None	None
[41]	K-Plet	Incremental	Core	Types	None
[46]	Triplets	Multiple	None	Types + RC	None
[77]	NN	Consensus	None	None	MS
[134, 167]	Texture	Single	LO	None	None
[132]	Texture + Triplets	Single	LO	None	None
[155, 156]	Texture	Single	LO	RC	None
[178, 183]	Triplets	Single	None	None	None
[179]	NN	Single	None	Types	None
[13]	NN	None	None	RC	None
[35, 15]	Radius	Consensus	None	None	None
[36]	Texture + Triplets	Consensus	LO	None	LS
[39]	K-Plet	Incremental	None	Types	None
[40]	Texture	Single	GSI	None	None
[50, 133, 187]	Texture	Multiple	None	RP	None
[62, 96]	Texture	Consensus	GSI	RP	None
[93]	K-Plet	Complex	None	None	None
[143]	NN	Consensus	None	RC	None
[7]	Texture	Multiple	GSI	None	None
[48, 2]	Texture	None	None	RP	None
[61]	Texture	Complex	GSI	RP	None
[131]	Radius	Consensus	LO	None	None
[135]	Texture	Multiple	None	Types + RP	None
[172, 177]	Triplets	Incremental	None	None	None
[164]	Texture	Consensus	LO	RP	None
[8]	Texture	Single	Core + GSI	RC	None
[49]	Radius + Texture	Multiple	RF + Core + LO	Types + RP	MS
[88]	Triplets	None	None	Types	None
[89]	Radius	Incremental	None	None	None
[115]	K-Plet	Single	GSI	Types	None
[162]	K-Plet	Single	RF + GSI	RP	None
[188]	NN	None	None	Types + RC	None
[14]	Texture	Consensus	None	RC + RP	None
[20, 152]	Texture	Complex	None	RP	None
[21]	Radius + Texture	Multiple	RF + LO	RP	MS
[87]	NN	Single	GSI	Types + RC	None
[114]	Texture	Incremental	None	Types + RP	None
[146]	Radius	None	None	None	None
[147]	Texture	None	LO	None	None
[173]	Radius	Consensus	None	Types + RC	None
[186]	Triplets	Multiple	None	Types	None
[19]	Texture	Multiple	RF + LO	RP	None
[26, 67]	Cylinder	Consensus	None	None	None
[148]	NN	Multiple	None	None	None
[31]	Texture + Triplets	None	GSI	None	None
[42]	K-Plet	Incremental	None	RC + RP	None
[53]	NN	None	None	None	None
[100]	Texture + Radius	Consensus	GSI	None	None
[185]	Triplets	Single	LO	None	None
[17]	Texture	Multiple	LO + Core	RP	None
[18]	Radius + Texture	Incremental	Core + LO	RP	None
[23]	Texture + Cylinder	Consensus	RF + LO + Core	None	None
[33]	Texture	None	GSI	None	None
[43]	Radius	Multiple	None	Types	None
[55]	Radius + Texture	None	LO + GSI	None	None
[119]	Texture + Triplets	None	Core + GSI	None	None

Other approaches perform fingerprint matching without the use of minutiae. They use the so-called texture information, being the most popular the *FingerCode* approach [75], which chains tessellated areas related to core points with Gabor filter to capture useful texture information. *FingerCode* features have been used in later research [7, 122, 176]. Isolated orientation [91] or ridge information [169] can also be used for matching. Finally, when high resolution images are available, level-3 features such as sweat pores, dots and incipient ridges can be used instead of minutiae [68, 103].

4.2. Fingerprint Indexing

Fingerprint indexing arises from the necessity of quick access to the fingerprint templates database in identification tasks. Some indexing techniques use partial information provided by the extracted minutiae of the fingerprint and build local structures centered on each minutia to establish similarity relationships between fingerprints and key indexes. This allows the ordering of candidate templates to increase the probability to match true paired fingerprints. Actually, these approaches can be viewed as minutiae-based matching approaches if the matching score is proportionally related to the number of coincident local structures.

The pure indexing proposals found in the literature are those based on minutiae triplets, which consider triangle-based characteristics to compute similarity among fingerprints, such as lengths, angles, handleless [10], etc.; and triangulations to improve efficiency [99]. Other indexing approaches utilize LO [105] and also RF [22]. Finally, several criteria for narrowing the candidate list obtained from indexing are evaluated in [24].

4.3. Current Progress in Matching

Nowadays, the matching field is continually in progress, offering new developments to improve personal identification. In the following, we briefly mention different matching related issues being currently tackled:

- Accelerating fingerprint matching: many efforts have been performed to speed up the matching process, for instance, by means of FPGA-based [79], GPU-based [59] parallel architectures or distributed computing [130].
- Fingerprint matching in embedded systems: sensors [3] and smart cards [9].
- Latent fingerprint matching: it is a more complicated problem because these fingerprints are inadvertent impressions left by fingers on surfaces [70, 128].
- Palmprint matching: based on ridges [44], minutiae [25, 32] and also effective approaches for latent matching [69, 102].
- Combinations with other traits and multiple matching: with face recognition [64], multiple matching [73], multiple sample [38] and minutiae-based synthesis for matching [158].

- Privacy protection in fingerprint matching: which tries to avoid the traditional encryption with its associated decryption, which exposes the fingerprint to the attacker. Two examples of recent techniques are the reverse MCC representation [51] and the combination of two different fingerprints into a new identity, based on minutiae, orientations and singular points [97].

5. Experimental Evaluation of Local Minutiae Matching Methods

This section is devoted to perform an experimental evaluation of the most important local minutiae-based matching algorithms. Subsection 5.1 establishes the experimental framework, presenting information about the used databases, the performance measures, the algorithms and their parameters. Then, Subsection 5.2 shows the analysis of the results of the used methods over the public FVC databases. Subsection 5.3 presents a study over four databases captured by the authors.

5.1. Experimental Set Up

This section describes the databases (Subsection 5.1.1), the accuracy measures (Subsection 5.1.2) and the framework (Subsection 5.1.3) used to carry out the experimental evaluation of the matchers.

5.1.1. Databases

We have used a wide variety of databases to test the performance and behavior of the matching algorithms. Table 3 presents their characteristics, showing their size and the average number of minutiae of the template and input fingerprints.

Table 3: Summary description of the used databases.

Denomination	Number of Fingerprints	Impressions per finger	Average template minutiae number	Average input minutiae number
FVC2000_db1a	100	8	49.51	48.93
FVC2000_db2a	100	8	58.43	57.97
FVC2000_db3a	100	8	132.97	144.18
FVC2000_db4a	100	8	36.88	37.10
FVC2002_db1a	100	8	53.11	49.69
FVC2002_db2a	100	8	61.87	56.93
FVC2002_db3a	100	8	58.23	57.52
FVC2002_db4a	100	8	50.52	49.78
FVC2004_db1a	100	8	49.01	62.84
FVC2004_db2a	100	8	64.45	64.19
FVC2004_db3a	100	8	94.52	98.63
FVC2004_db4a	100	8	55.00	52.61
DB1	1228	10	45.26	45.20
DB2	1228	10	145.79	142.94
DB3	1228	10	44.36	43.34
DB4	1228	10	44.50	43.35

First, we apply the algorithms over twelve of the well-known FVC databases, using the first impression of each finger as template, and the other seven impressions as input. These

databases are designed for verification competitions, and therefore their fingerprints have bad quality on purpose. More information about the FVCs databases can be found in [110, 112, 111].

Four additional databases, captured by the authors' research groups, are used for the study. They simulate a real environment for identification with consented fingerprints captures of reasonable quality. All of them are composed by the same fingers, captured by four different sensors (Table 4).

Table 4: Sensors used to capture the fingerprints.

Database	**Sensor**	**Sensor type**	**Fingerprint type**
DB1	Upek Eikon	Capacitive	Swipe
DB2	Suprema RealScan-D	Optical	Rolled
DB3	Suprema BioMini	Optical	Plain
DB4	SecuGen Hamster IV	Optical	Plain

A total of 308 people participated in the study. The fingerprints of the thumb, forefinger and middle finger of both their hands were captured along three different sessions. After removing the failed captures, we selected three random input fingerprints per session and a single template fingerprint for each finger and sensor. After this manner we get four final databases that contain the same 1228 fingers captured by four different sensors.

5.1.2. Accuracy measures

The accuracy of a fingerprint matcher can be measured from two different perspectives:

- Verification: consists of matching two fingerprints to determine whether they correspond to the same finger or not.
- Identification: tries to find the match of an input fingerprint in a database, comparing it to all the templates.

Each perspective employs different accuracy measures. In this paper, we use the following verification measures:

- False Matching Rate (FMR): rate of different fingerprints that are considered to be the same by the matcher. Each possible score has an FMR associated; the higher the score, the lower the FMR.
- False Non-Matching Rate (FNMR): rate of corresponding fingerprints that are erroneously considered different.
- Equal-Error Rate (EER): value (corresponding to a certain score threshold) where FMR and FNMR are equal.
- ROC: curve that plots the Genuine Matching Rate (GMR $= 1 -$ FNMR) versus the FMR.

- FMR100: lowest achievable FNMR for a FMR $\leq 1\%$.
- FMR1000: lowest achievable FNMR for a FMR $\leq 0.1\%$.
- ZeroFMR: lowest achievable FNMR for a FMR $= 0\%$.

Within an identification process, most of the accuracy measures are related to the rank, which is the position of the genuine score if all the obtained scores are ordered in descending order. In other words, the rank is the minimum number of database fingerprints that have to be returned by the identification system to ensure that the correct identity is included. We use the following identification accuracy measures:

- True positive rate (TPR): percentage of test fingerprints that are correctly identified in the database, when only the best matching score is retrieved. The TPR is the error obtained when using a rank of 1.
- R100: lowest rank that allows an error lower than 1%.
- ZeroR: lowest rank that does not allow errors.
- Cumulative Match Curve (CMC): curve that represents the error associated to each rank.

The optimum value for R100 and ZeroR is 1, whereas the worst one is the size of the database.

In addition to all these values, the average matching time is also important to determine if a matching algorithm is suitable for a certain identification system.

For reasons of space and concision, not all of these measures are presented in the paper. The full set of results is accessible at `http://sci2s.ugr.es/MatchingReview/`.

Statistical tests allow to establish a fair comparison between the methods and to detect significant differences. In this paper, we use the nonparametric tests recommended in [45, 54], which claim to be simple, safe and robust.

Furthermore, we apply the Friedman test [52] to measure the differences between the methods with a multiple comparison analysis. The Holm procedure is applied to find out which algorithms are distinctive. [1]

5.1.3. Experimental framework

To compute these measures it is necessary to perform all the matching comparisons between template and input fingerprints. In order to obtain the results within a reasonable time, and to fix a common execution environment, all the experiments have been carried out within the parallel framework proposed in [130], which speeds up the computation while ensuring that the results are the same as in a sequential execution. The NIGOS

[1] Additional information about these tests, as well as the corresponding software, are available at `http://sci2s.ugr.es/sicidm/`.

mindtct algorithm [165] has been used for the minutiae extraction. All executions have been performed in a cluster of 12 machines, each of them with two Intel(R) Xeon(R) E5-2620 CPU at 2.00 GHz and 64GB RAM.

The empirical study involves 12 matching algorithms from those listed in Table 2. We want to outline that all the implementations of the matching algorithms, excepting the proposed in [165], were developed by us and they are only based on the descriptions and specifications given by the respective authors according to their papers. It is also noteworthy that our implementation of Feng's algorithm only uses the minutiae features provided by the minutiae extractor, and therefore is not as complex as the original algorithm. The parameter values used for all matchers have been extracted from these papers and are shown in Table 5. In the cases where the parameter values are not given in the original paper, we experimentally selected values that suit the general case. We have not performed any training to adapt these parameters, because our objective is not to maximize the accuracy, but to fairly compare the matchers and their robustness in a common environment and upon different databases.

5.2. Analysis and Empirical Results on FVC Databases

This section analyzes the results obtained over the 12 FVC databases, in terms of verification and identification.

5.2.1. Verification

Tables 6 and 7 present the EER and FMR100, respectively, as the error percentage obtained for all tested algorithms over the 700 input fingerprints of each FVC database. The best result for each database is stressed in boldface. Additionally, Figure 1 plots the ROC curve for the most difficult FVC database (FVC2002_db3a, which obtains the highest average EER).

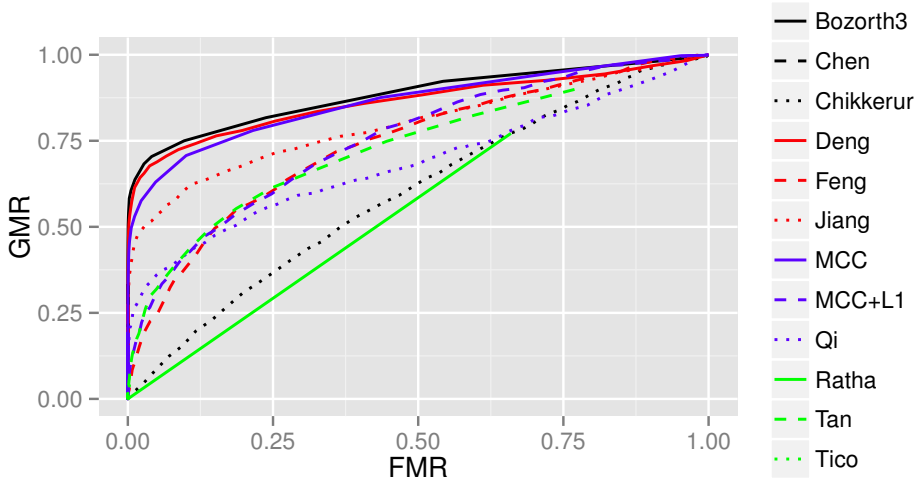


Figure 1: ROC curve for FVC2002_db3a

Bozorth3 is the best performing algorithm in general. If we focus on the EER, MCC also obtains good results, while Deng is more accurate in terms of FMR100. The ROC curves

Table 5: Parameters for the methods used in the experiments

Algorithm	**Parameters**
Mindtct [165]	output format = ANSI INCITS 378-2004 image enhancement = enabled
Bozorth3 [165]	input format = ANSI INCITS 378-2004, Maximum number of minutiae = 150, Minimum number of minutiae = 10
Jiang [81]	$w_d = 1, w_\theta = 54\pi, w_\phi = 54\pi, w_n = 0, w_t = 0$ Consolidation step iterations = 5, Minutiae neighborhood size = 2 $BG_1 = 8, BG_2 = \frac{\pi}{6}, BG_3 = \frac{\pi}{6}$
Ratha [136]	Neigh=6, F_{min}=0.4, TM=8, RelDist=0.2, RidgesDiff=10, EdgesDiff=0.1, MisMatch=10000
Tan [150]	Δ_α=10, Δ_τ=20, WindowsSize=32, $Triplet_{angle}$=2, $Triplet_{side}$=15,$Triplet_{minutieaDensity}$=2,$Triplet_{RidgeCount}$=2, TS=0.15, T_Θ=30, T_1=150, T_2=100, T_D=12, T_N=8, MaxTriangleWidth=300
Tico [154]	TH_{RV} = 25, Block = 16, NumRadius=4, THR_t=Π, THR_{Dist}=6, THR_{angle}=$\frac{\Pi}{6}$, MTI=6, μ=0.25
Deng [46]	$Minutiae_{minDelanuy}$=20, TH_1=36, TH_{num}=20, TH_{edge}=15, W_0=1, W_1=W_2=$\frac{0.3 \cdot 180}{\Pi}$, W_3=3, W_4=W_5=6, TH_d=8, TH_Θ=TH_ϕ=TH_{ang}= $\frac{\Pi}{6}$, TH_{rc}=3, TH_{SL}=0.2
Qi [132]	TH_{RV} = 25, Block = 16, PointSeg=3,MinutiaeSeg=6, LongSeg=18, THR_t=Π, THR_O=$\frac{\Pi}{2}$, THR_{Dist}=10, THR_{angle}=$\frac{\Pi}{4}$, W_M=0.6,W_O=0.4
Chen [35]	$Thr_L = 55, Thr_H = 80, R = 80, RS = 100, \theta_L = 0.25, \theta_H = 0.4$ $\text{len}_L = 5, \text{len}_H = 20, Thr_{topo} = 0.7$
Chikkerur [39]	$K = 8$, Bounding box = $\{8, \pi/6, \pi/6\}$
Feng [49]	Neighborhood radius = 60, Translation Tolerance Box = 8, Rotation Tolerance Box = $\pi/6$, Rotation maximum threshold = $5\pi/9$, Minimum normalized similarity = π
MCC [26]	$R = 70, N_s = 16, N_d = 6, \sigma_s = \frac{28}{3}, \sigma_d = \frac{2\pi}{9}, \mu_\Psi = 0.01, \tau_\Psi = 400$ $\omega = 50, min_{VC} = 0.75, min_M = 2, min_{ME} = 0.60, \sigma_\theta = \frac{\pi}{2}, max_{n_p} = 12$ Floating-point-based version: enabled, consolidation scheme= LSSR, $\mu_P = 20$ $w_R = 0.5, \mu_1^\rho = 5, \tau_P = 0.6, min_{n_p} = 4$ $\mu_2^\rho = \frac{\pi}{12}, \mu_3^\rho = \frac{\pi}{12}, \tau_1^\rho = -1.6, \tau_2^\rho = -30, \tau_3^\rho = -30, n_{rel} = 5$
MCC+L1 [23]	$Block_w = 16, Block_l = 32, FV_{long} = 36, FV_{radius} = 4$ W_1=W_3=0.16, W_2=0.37, W_4=0.31, Threshold=0.4

show that Bozorth3, MCC and Deng dominate all methods, followed by Jiang.

These four algorithms are substantially different from each other. For example, MCC uses cylinders as local structure, while Deng uses the texture and Jiang and Bozorth3 use

Table 6: EER percentages for FVC databases

Database	Bozorth3	Jiang	Ratha	Tan	Tico	Deng	Qi	Chen	Chikkerur	Feng	MCC	MCC+L1
FVC2000_db1a	**7.481**	12.945	43.557	25.711	83.286	7.633	25.446	69.143	37.840	20.013	8.207	24.804
FVC2000_db2a	8.751	16.451	42.499	37.143	87.429	9.308	20.124	66.143	39.406	22.252	**8.578**	20.725
FVC2000_db3a	18.750	24.954	41.996	30.965	95.857	**14.814**	29.978	64.714	43.219	40.022	20.216	23.152
FVC2000_db4a	**5.817**	8.166	42.042	24.228	91.857	17.006	41.777	46.571	36.498	32.259	6.026	20.144
FVC2002_db1a	15.286	16.312	41.761	26.366	80.000	16.676	34.640	63.571	40.776	**15.067**	15.325	23.287
FVC2002_db2a	14.564	13.404	38.141	27.708	79.571	12.959	27.852	46.000	37.840	15.254	**12.553**	22.166
FVC2002_db3a	**20.062**	27.686	46.093	33.002	95.286	21.258	37.346	86.714	43.462	31.922	21.867	32.015
FVC2002_db4a	**21.003**	23.281	42.641	29.839	88.857	24.352	39.921	90.286	36.369	23.692	23.988	26.181
FVC2004_db1a	**17.374**	24.999	44.405	40.286	98.429	20.409	42.930	83.571	47.938	23.209	19.562	28.592
FVC2004_db2a	**17.183**	23.798	45.195	38.728	48.000	20.766	35.354	85.143	42.102	29.003	19.786	31.675
FVC2004_db3a	**6.265**	13.834	43.545	31.792	79.000	9.396	31.119	29.714	43.415	35.287	10.037	18.699
FVC2004_db4a	**26.189**	31.438	42.315	33.712	65.286	28.372	40.106	93.286	40.029	29.240	28.122	27.160

Table 7: FMR100 percentages for FVC databases

Database	Bozorth3	Jiang	Ratha	Tan	Tico	Deng	Qi	Chen	Chikkerur	Feng	MCC	MCC+L1
FVC2000_db1a	13.334	20.269	98.704	67.550	90.591	**12.764**	50.339	100.000	97.107	76.583	14.239	79.610
FVC2000_db2a	18.712	27.582	98.647	80.952	94.609	**18.435**	47.920	100.000	96.458	89.613	18.564	71.068
FVC2000_db3a	37.876	54.204	98.619	89.996	98.250	**28.632**	95.592	100.000	98.114	97.057	46.071	82.449
FVC2000_db4a	14.012	15.944	98.621	77.234	100.000	31.254	88.820	100.000	95.597	89.744	**9.657**	66.374
FVC2002_db1a	24.967	25.806	98.605	67.114	92.114	22.380	56.529	100.000	97.442	52.155	**22.192**	76.952
FVC2002_db2a	22.645	20.300	98.378	64.810	94.427	**17.357**	46.073	56.559	97.010	48.794	19.363	72.116
FVC2002_db3a	**37.324**	56.179	98.830	85.811	100.000	39.946	74.281	100.000	97.974	90.282	47.915	86.162
FVC2002_db4a	52.152	**51.010**	98.655	80.990	96.435	51.858	75.166	100.000	96.911	85.916	51.861	86.607
FVC2004_db1a	**36.286**	51.935	98.748	92.937	98.631	41.550	77.401	100.000	98.048	87.091	36.503	86.848
FVC2004_db2a	**35.089**	52.450	98.787	91.740	97.416	37.457	74.450	91.571	97.785	95.038	41.324	86.201
FVC2004_db3a	**13.618**	34.910	98.704	86.950	95.782	28.127	79.643	40.986	98.521	98.007	28.420	71.332
FVC2004_db4a	**60.790**	66.630	98.637	85.705	97.055	69.738	83.779	100.000	97.156	87.176	66.055	89.465

the nearest neighbors. The consolidation type is also different. However, it is noteworthy that none of them use any additional features: Jiang and Deng use both the minutia type and the ridge count, while Bozorth3 and MCC only use the basic minutia information.

It is also interesting that, even though MCC+L1 obtains good results when the GMR is high, it does not improve the results obtained with the bare use of MCC. Note that the MCC+L1 algorithm uses a different, less accurate variant of MCC (with binary encoding and a different consolidation), meant to be very efficiently implemented on hardware.

This states that none of the characteristics described in Subsection 3.1 can be discarded as worse than the rest: the verification performance is determined by the matching algorithm as a whole, and each local structure and consolidation can supply useful information. Nevertheless, the use of additional features does not always lead to more accurate results.

Along with the accuracy, the computational performance is a very important characteristic of a fingerprint matching algorithm, especially when it has to deal with large fingerprint

databases.

Table 8 summarizes the average matching times for the tests performed so far. Note that these times are measured in computational time, and therefore are not affected by the parallel framework in which the tests have been carried out.

Table 8: Average matching times (in milliseconds) for FVC databases

Database	Bozorth3	Jiang	Ratha	Tan	Tico	Deng	Qi	Chen	Chikkerur	Feng	MCC	MCC+L1
FVC2000_db1a	1.026	**0.382**	2.865	161.339	21.319	2.409	0.739	3.936	5.051	3.577	10.094	0.762
FVC2000_db2a	1.719	**0.541**	3.938	513.535	34.927	4.400	0.913	7.712	6.187	6.632	13.710	0.979
FVC2000_db3a	6.187	**4.149**	22.220	81277.765	228.140	91.468	4.653	51.330	91.320	46.758	82.510	4.854
FVC2000_db4a	3.145	**0.234**	1.606	49.119	11.242	1.284	0.466	2.249	3.330	2.409	5.733	0.436
FVC2002_db1a	1.349	**0.422**	3.189	279.543	21.319	3.386	0.749	4.541	5.532	4.047	10.880	0.784
FVC2002_db2a	1.233	**0.551**	4.149	442.721	33.742	3.989	0.955	4.777	6.725	4.554	14.559	0.901
FVC2002_db3a	1.235	**0.534**	3.964	436.841	28.712	4.222	0.986	6.771	6.301	5.920	14.168	1.054
FVC2002_db4a	1.268	**0.397**	2.915	338.823	18.921	3.697	0.784	5.163	5.046	4.770	10.334	0.745
FVC2004_db1a	1.488	**0.491**	3.656	313.619	16.544	3.849	0.942	6.434	5.563	5.553	12.253	0.801
FVC2004_db2a	1.534	**0.680**	4.845	853.888	33.527	5.448	1.292	9.432	7.482	7.922	17.974	1.275
FVC2004_db3a	16.566	**1.850**	10.815	10575.240	99.901	21.829	2.336	27.278	18.136	25.183	40.523	2.535
FVC2004_db4a	1.312	**0.461**	3.340	540.880	25.047	4.189	0.872	6.484	5.325	5.913	11.737	0.840

We can notice that in all cases, Jiang is the fastest algorithm, followed by Qi. The former performs a simple consolidation and does not use any additional features, which makes the computation very fast. The latter does not involve any consolidation, and therefore performs all the matching process from a local point of view.

In the other extreme, the Tan's algorithm is extremely slow, especially for databases with more minutiae per fingerprint. This algorithm computes all the triplets of the fingerprints, and compares them. This computation has factorial order and therefore takes a long time for fingerprints with a certain number of minutiae. This is an example of an algorithm that could be improved by a previous minutiae filtering.

It is curious to note that the Qi's algorithm is very fast, although it also uses triplets. However, it includes a first candidate selection using the texture, avoiding the creation of all possible triplets.

If we compare the overall performance of the algorithms, we can observe that the consolidation bears a high weight in the runtime. Complex consolidations require more computing time, as for MCC, Deng and Tico.

Another observation that can be made is that MCC+L1 is considerably faster than MCC. This is due to the structure of MCC+L1, which first compares the L1 features of the fingerprints, and applies MCC only if they are similar enough. This hierarchical matching is able to save a lot of computing time, but also explains why MCC+L1 is often less accurate than MCC.

Table 9 shows the results of the statistical tests for several accuracy measures, highlighting Bozorth3, MCC and Deng as the best algorithms.

Table 9: Statistical tests over the verification measures over the FVC databases

Algorithm	EER	FMR100	FMR1000	ZeroFMR
Bozorth3	**1.500**	**2.083**	**1.750**	**1.292**
Jiang	4.000	3.500	3.833	3.458
Ratha	9.917	11.083	11.250	10.042
Tan	7.250	6.833	7.250	7.167
Tico	11.750	9.750	10.208	10.042
Deng	3.000	**2.083**	2.250	3.083
Qi	7.583	5.583	6.333	8.833
Chen	10.833	10.500	7.250	5.500
Chikkerur	9.000	9.917	10.625	10.042
Feng	5.333	7.500	7.833	9.458
MCC	2.500	2.333	2.167	2.250
MCC+L1	5.333	6.833	7.250	6.833
Friedman P-value	6.18e-011	6.13e-11	5.33e-11	7.34e-11

5.2.2. Identification

Tables 10 and 11 summarize the R100 and TPR values, respectively. Finally, Figure 2 displays the CMC curves for the FVC2002_db3a database.

Table 10: R100 values for FVC databases (100 templates)

Database	Bozorth3	Jiang	Ratha	Tan	Tico	Deng	Qi	Chen	Chikkerur	Feng	MCC	MCC+L1
FVC2000_db1a	100	91	100	100	96	100	100	100	100	100	**72**	79
FVC2000_db2a	100	96	100	100	94	72	97	100	100	100	**64**	77
FVC2000_db3a	100	98	100	94	97	**81**	98	100	100	100	98	84
FVC2000_db4a	100	79	100	100	100	100	100	100	100	100	**33**	95
FVC2002_db1a	100	95	100	100	100	100	100	100	100	100	89	**85**
FVC2002_db2a	100	92	100	100	100	100	98	100	100	100	88	**80**
FVC2002_db3a	100	98	100	100	99	100	99	100	100	100	91	**86**
FVC2002_db4a	100	95	100	100	98	100	100	100	100	100	91	**84**
FVC2004_db1a	100	94	100	100	100	100	99	100	100	100	**87**	91
FVC2004_db2a	100	96	100	100	99	98	99	100	100	100	**94**	**94**
FVC2004_db3a	100	90	100	100	96	**49**	99	100	100	100	64	84
FVC2004_db4a	100	96	100	100	96	100	99	100	100	100	94	**89**

It is curious to observe that, while MCC+L1 is the best algorithm if we focus on the rank, MCC obtains better numeric results (for example for FVC2000_db4a) and Deng and Bozorth3 have higher TPR in most cases. The CMC curves explain this behavior. For low ranks, Deng and Bozorth3 perform better, and therefore have a lower TPR. MCC is slightly below Deng in accuracy, while MCC+L1 obtains good results for very high ranks.

The high values obtained denote the difficulty of the FVC databases: the algorithms need to return the majority of the databases in order to ensure that the genuine fingerprint is returned. Note that the methods that have a value of 100 return the entire database. Chen's algorithm has a very low CMC curve because the matching score is often exactly zero (when the compared fingerprints do not match some conditions). This causes some

Table 11: TPR percentage for FVC databases

Database	Bozorth3	Jiang	Ratha	Tan	Tico	Deng	Qi	Chen	Chikkerur	Feng	MCC	MCC+L1
FVC2000_db1a	**87.857**	78.571	37.714	50.143	10.857	86.286	49.286	29.286	3.000	28.571	85.286	43.429
FVC2000_db2a	82.857	71.714	42.857	45.286	5.143	**85.286**	49.000	33.286	4.429	12.714	81.571	50.714
FVC2000_db3a	63.143	45.286	26.571	17.857	2.714	**72.571**	10.714	33.143	2.286	2.143	54.000	50.714
FVC2000_db4a	88.571	82.429	24.857	38.429	8.429	70.286	10.000	51.143	3.429	7.429	**91.571**	52.000
FVC2002_db1a	77.714	75.857	53.143	49.571	8.000	**78.286**	42.571	35.143	3.571	47.286	77.429	44.000
FVC2002_db2a	80.143	81.429	69.286	52.000	6.571	**83.143**	54.286	47.571	4.000	56.429	80.429	58.143
FVC2002_db3a	**63.571**	41.000	18.714	20.286	12.286	60.857	25.000	13.286	1.857	8.571	50.143	21.286
FVC2002_db4a	48.000	44.571	28.714	31.000	2.143	**50.857**	23.857	8.714	3.143	14.000	46.857	29.000
FVC2004_db1a	**65.286**	46.714	23.143	9.000	1.429	61.857	23.857	16.143	2.571	29.143	60.857	23.143
FVC2004_db2a	**64.714**	46.429	20.286	11.714	3.000	63.000	25.571	12.286	2.571	5.000	58.429	21.571
FVC2004_db3a	**86.714**	65.571	16.000	29.286	6.714	83.857	19.429	61.429	2.571	1.000	74.429	57.286
FVC2004_db4a	**37.286**	34.143	21.000	26.429	1.857	29.000	17.571	4.714	3.286	9.857	32.714	25.429

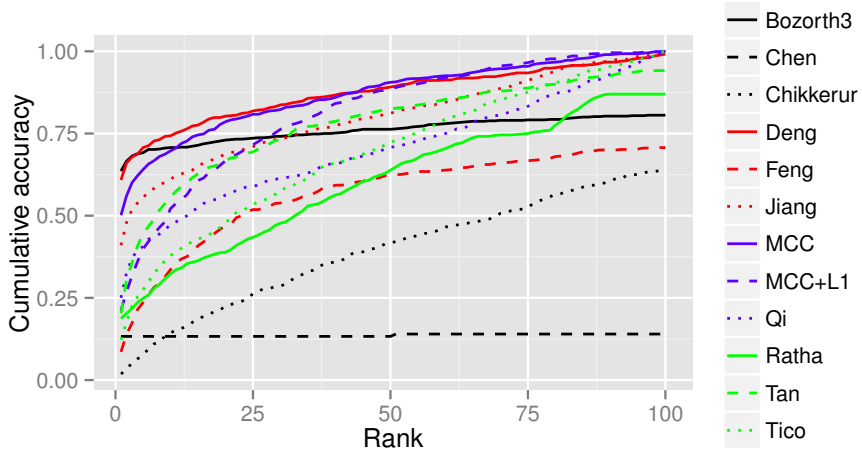


Figure 2: CMC curves for FVC2002_db3a

genuine scores to be the lowest ones in the fingerprint database, and therefore the rank necessary to ensure a certain identification accuracy is greatly increased.

Table 12 displays the results of the statistical tests.

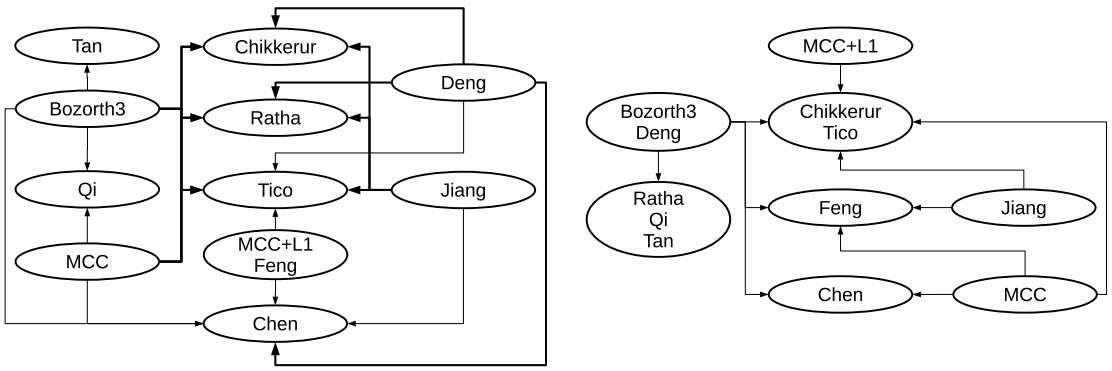
In general, the identification results of these algorithms are similar to the ones obtained for verification, and their behavior remains the same.

To conclude the study, Figure 3 outlines two directed-graphs for verification and identification statistical results respectively. Each method is represented as a vertex, and the edges connect two methods in which the Holm test has detected significant differences. Specifically, in Figure 3a, those methods that receive an arrow are outperformed by the linked algorithm in terms of EER, whereas in Figure 3b, we focus on the TPR measure. A Thick line means that a method statistically outperform another considering all the verification or identification measures. To simplify the graphs, the methods with identical differences with the others have been grouped in the same nodes.

Table 12: Statistical tests for the identification measures over the FVC databases

Algorithm	**TPR**	**R100**	**ZeroR**
Bozorth3	**1.667**	8.875	7.375
Jiang	3.667	3.458	**4.083**
Ratha	7.208	8.875	7.375
Tan	7.083	8.292	7.375
Tico	10.917	5.625	6.917
Deng	1.917	6.333	6.792
Qi	7.250	6.167	7.000
Chen	8.417	8.875	7.375
Chikkerur	11.583	8.875	7.375
Feng	9.417	8.875	7.375
MCC	2.833	1.958	4.625
MCC+L1	6.042	**1.792**	4.333
Friedman P-value	5.652e-11	5.652e-11	0.089

The figure ratifies the analysis of the accuracy measures: Bozorth3, MCC, Jiang and Deng are the most accurate algorithms for the FVC databases, with statistically significant differences with respect to the other methods.



(a) Verification (b) Identification

Figure 3: Significant differences among the tested methods

5.3. Analysis and Empirical Results on Captured Databases

In the preceding section, the algorithms of Bozorth3 [165], Jiang [81], Deng [46] and MCC [26] were highlighted as the most accurate for the FVC databases, as they are statistically better than other methods both for verification and identification. This section performs a deeper study upon the four captured databases described, focusing on these four algorithms.

5.3.1. Verification

Table 13 presents the results obtained in terms of EER, FMR100 and FMR1000. Figure 4 displays the ROC curves.

Table 13: Verification performance measures (in percentages)

	EER				FMR100				FMR1000			
Database	**Bozorth3**	**Jiang**	**Deng**	**MCC**	**Bozorth3**	**Jiang**	**Deng**	**MCC**	**Bozorth3**	**Jiang**	**Deng**	**MCC**
DB1	**2.763**	6.292	4.288	3.448	**4.852**	15.092	9.337	6.908	**11.144**	29.223	22.733	15.638
DB2	0.686	3.712	3.393	**0.350**	0.617	6.546	6.056	**0.180**	1.219	14.131	15.309	**0.623**
DB3	0.839	2.518	1.018	**0.414**	0.779	4.013	1.025	**0.280**	2.103	9.177	2.845	**0.889**
DB4	0.788	2.512	0.951	**0.443**	0.701	3.958	0.929	**0.303**	1.951	8.806	2.624	**0.834**

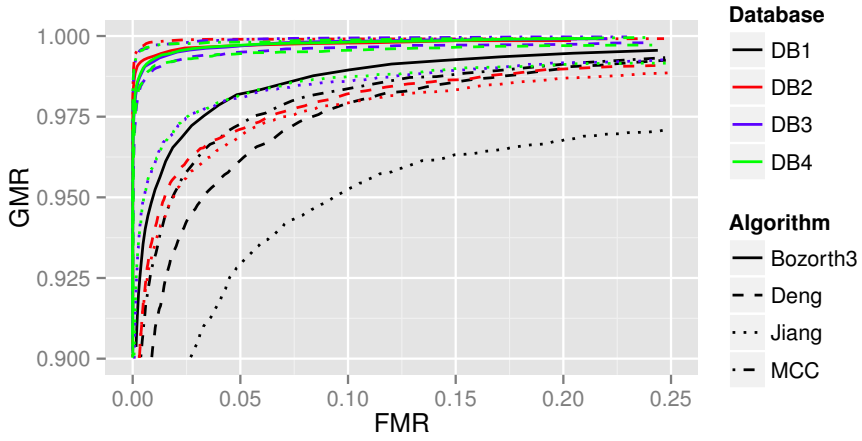


Figure 4: ROC curves for captured databases

Note that the error values for these databases are far better than those obtained for the FVC ones, which are designed for test purposes and whose quality is deliberately bad.

In this case, MCC obtains the best results for all measures and databases except DB1, in which Bozorth3 is better, and the ROC curves follow the same behavior. Jiang gets the worst values among the three tested algorithms.

MCC and Bozorth3 only use the basic minutiae information to build their local structures, while Deng takes into account texture information and some minutiae peculiarities such as the ridge count and the type. Therefore, the fact that Deng is able to obtain good results with the FVC databases—even though it is outperformed by MCC and Bozorth3 for the captured ones—suggests that the texture is less affected than the minutiae in the FVC bad quality images.

It is also noteworthy that Jiang and Deng perform better with the DB3 and DB4 databases (plain fingerprints), while Bozorth3 excels on DB1 (swipe fingerprints), and MCC obtains better results with DB2 (rolled fingerprints). This could happen due to the convex hull computation carried out by MCC, which filters the minutiae on the borders of the fingerprint. Bozorth3, Deng and Jiang do not carry out any special treatment on those areas,

which are more prone to errors. In all cases, the DB1 database (captured with a narrow swipe sensor) is the most difficult one for the verification.

As for the computing times, we observe the same behavior as with the FVC databases (Table 14). Jiang is the fastest algorithm, followed by Bozorth3, MCC and Deng, which involve more complex consolidations and more information.

Table 14: Average matching times (in milliseconds)

Database	Bozorth3	Jiang	Deng	MCC
DB1	3.679	**0.469**	11.178	6.061
DB2	12.076	**7.501**	175.132	64.826
DB3	3.227	**0.415**	9.057	5.884
DB4	3.184	**0.423**	9.054	5.797

5.3.2. Identification

To conclude this study, Table 15 and Figure 5 present the identification performance measures and the CMC for the four tested algorithms over the four captured databases.

Table 15: Identification performance values (1228 templates)

	R100				ZeroR				TPR			
Database	Bozorth3	Jiang	Deng	MCC	Bozorth3	Jiang	Deng	MCC	Bozorth3	Jiang	Deng	MCC
DB1	1228	866	**147**	237	1228	1228	1228	**1169**	**90.264%**	69.942%	85.125%	84.057%
DB2	**1**	297	121	**1**	1228	1228	1220	**1202**	99.077%	87.559%	93.838%	**99.222%**
DB3	**1**	172	6	**1**	1228	1224	1228	**1027**	99.050%	90.771%	98.082%	**99.285%**
DB4	**1**	118	4	**1**	**1228**	**1228**	**1228**	**1228**	99.168%	90.879%	98.172%	**99.358%**

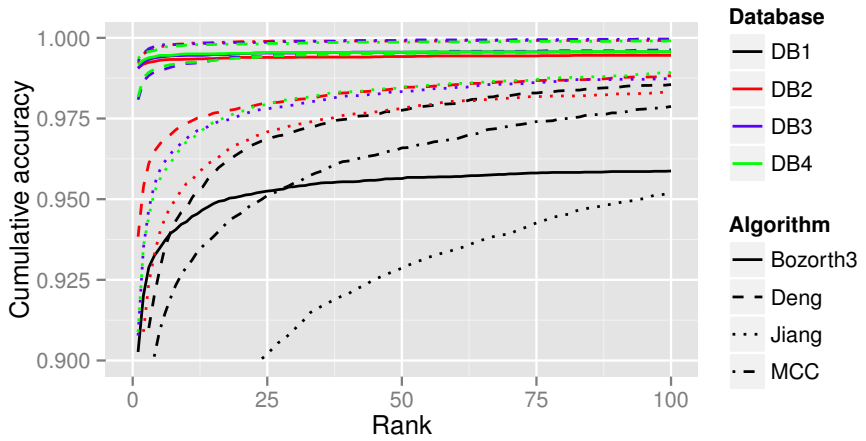


Figure 5: CMC curves for the captured databases

Again, MCC highlights as the most accurate algorithm, except for the DB1 database, for which Deng obtains better R100 and Bozorth obtains better TPR. The CMC curves

illustrate these results, showing that for low ranks, Bozorth3 performs better than MCC and Deng over DB1. As the rank increases, the cumulative accuracy of Deng increases too. This result contrasts with the verification analysis, which stated that both Bozorth3 and MCC outperform Deng for all databases.

The explanation of this fact is that the verification performance measures are calculated considering a fixed score. That is, each point of the ROC curve plots the FMR and FNMR obtained with a certain score. However, the rank is independent of the numerical value of the scores: it only takes into account their order. The different behavior of ROC and CMC means that the score variability over these databases is higher for Deng than for MCC and Bozorth3. This means that, given an input fingerprint, Deng can ensure with a high confidence that the genuine score is higher than the impostor ones; however, it does not ensure with the same confidence that the genuine and impostor scores of all fingerprints can be separated by a certain fixed threshold.

Otherwise, the relative performance of the databases is maintained: the swipe sensor provides fingerprints that are more difficult to recognize, as well as the DB2 sensor for rolled fingerprints.

6. Conclusions

In this paper, we have compiled the most relevant work in the scientific literature about fingerprint local minutiae-based matching. We have described the background in the field, including some references about global matching and feature extraction techniques. Then, we have studied the main properties of the local matching algorithms, as well as the information they are based on, distinguishing between five different aspects: topology of local structure, type of consolidation, usage of additional features, minutiae peculiarities and parameter learning. Using all this information we have built a taxonomy of more than 80 local minutiae matching methods.

In order to complete the study, we have designed and implemented an experimental framework using two sets of databases: 12 from the FVC competitions, which are publicly available, and 4 databases captured by the authors' research groups. The study analyzes the results of 12 of the studied matchers, both in terms of verification and identification performance measures.

After the work realized in this paper, the following conclusions can be drawn:

- Fingerprint matching is a very active field, with dozens of proposed matching methods.
- The obtained results reveal big differences in the accuracy of the matchers, highlighting some of them as more precise than the others.
- The best performing algorithms do not share any special characteristics, although none of them uses any fingerprint features apart in addition to the minutiae coordinates, angle, type and ridge count.
- Furthermore, it has been seen that for different databases, different matchers may be the most accurate. An especially revealing result is that there is a big difference

between processing rolled, plain and swipe fingerprints, as the different number of minutiae and the presence or not of minutiae on the borders affects the behavior of the matchers.

- This states that some of the different approaches to design matching algorithms are equally valid, and depend on the particular fingerprints.
- There is also a big difference in the computational complexity of the methods: the fastest methods are more suitable for systems with very large fingerprint databases.
- This paper can help nonexperts to choose an appropriate matching algorithm that suits their particular problem.
- It can also help other researchers in the field to develop new matching methods, using the components and properties described in this paper.

In our opinion, the specialized literature contains lots of ideas related to minutiae fingerprint matching, some of them are quite similar and even it may be possible to find overlap among them. Most of the fingerprint matching approaches introduced in the last four decades are minutiae based. One of the reasons to expect minutiae-based algorithms to perform well is the sheer amount of research done on this approach. Original ideas are those which have served as inspiration of the rest of the matching methods. The majority of them were analyzed in this paper with empirical studies, trying to fix one of the main problems observed in this respect in the literature: almost all the proposals were compared under different configurations and without a standard. However, this task is very tedious due to the fact that the papers do not provide all the details to achieve a perfect implementation of the idea presented, especially the information related to the values of the parameters employed.

In the theoretical slope, we realize that the usage of isolated minutiae for matching, although is enough to achieve competitive performance, falls short in more complex scenarios. This is the reason that justifies the fact of real life implementations of fingerprint systems that fuse fingerprints with other traits or employ double fingerprint inputs. The world-wide large scale deployment of fingerprint systems demands a new generation of accurate and highly interoperable algorithms; therefore the development of minutiae-only matching algorithms will not be abandoned for a long time.

In the practical slope, the experiments have shown that none of the features established in the taxonomy can be considered as better than the others, and that the matching algorithms work as a whole. The same algorithms have also been proven to perform differently in different databases. Therefore, all the local structures, consolidations and features described in the taxonomy can be useful for future developments, as the key of an accurate matching algorithm is an adequate use of these parts and not the parts themselves. It has also been noted that the difference in the identification time can be huge depending on the used methods, especially for rolled fingerprints. When time is a limited resource, care must be taken on choosing local structures and consolidations that are at most linear or quadratic with respect to the number of minutiae.

As future research directions, we particularize the following ones:

- Biometric Fusion: the main advantage of fusion in the context of biometrics is an improvement in the overall matching accuracy. This is commonly known as multi-factor authentication and is considered more secure than using fingerprints alone as these other factors have some of their own strengths. Combining fingerprints with other biometric traits offers several advantages, such as the improvement of the universality or the problems caused by the acquisition of poor quality images due to external factors.

- Indexing and Big Data: as we mention in Section 4.2, the indexing is particularly useful when large volumes of fingerprints are stored every day. Identification task in large data bases could become in a real challenge for obtaining quick responses for each query. The employment of Big Data solutions to fingerprint matching and indexing is incoming in the near future.

- High quality images: in certain applications, it is possible to acquire high resolution images in which at the very-fine level, intra-ridge details can be detected. These include width, shape, curvature, edge contours of ridges as well as other permanent details such as dots and incipient ridges. One of the most important fine-level details is the finger sweat pores, whose positions and shapes are considered highly distinctive.

Acknowledgements

This work was supported by the Research Projects CAB(CDTI), TIN2011-28488, and TIN2013-40765-P. D. Peralta holds an FPU scholarship from the Spanish Ministry of Education and Science (FPU12/04902).

References

[1] C. Arcelli, G.S. di Baja, A width-independent fast thinning algorithm., IEEE Transactions on Pattern Analysis and Machine Intelligence 7 (1985) 463–474.

[2] A. Balti, M. Sayadi, F. Fnaiech, Improved features for fingerprint identification, in: Proceedings of the Mediterranean Electrotechnical Conference - MELECON, 2012, pp. 878–883.

[3] S. Bayram, H.T. Sencar, N. Memon, Efficient sensor fingerprint matching through fingerprint binarization, IEEE Transactions on Information Forensics and Security 7 (2012) 1404–1413.

[4] A.M. Bazen, S.H. Gerez, Systematic methods for the computation of the directional fields and singular points of fingerprints., IEEE Transations on Pattern Analysis and Machine Intelligence 24 (2002) 905–919.

[5] A.M. Bazen, S.H. Gerez, Fingerprint matching by thin-plate spline modelling of elastic deformations., Pattern Recognition 36 (2003) 1859–1867.

[6] A. Bengueddoudj, S. Akrouf, F. Belhadj, D. Nada, Improving fingerprint minutiae matching using local and global structures, in: 8th International Workshop on Systems, Signal Processing and Their Applications, WoSSPA 2013, 2013, pp. 279–282.

[7] F. Benhammadi, M.N. Amirouche, H. Hentous, K. Bey-Beghdad, M. Aissani, Fingerprint matching from minutiae texture maps., Pattern Recognition 40 (2007) 189–197.

[8] F. Benhammadi, K.B. Beghdad, H. Hentous, Fingerprint verification based on core point neighbourhoods minutiae, in: AICCSA 08 - 6th IEEE/ACS International Conference on Computer Systems and Applications, 2008, pp. 530–536.
[9] F. Benhammadi, K.B. Bey, Embedded fingerprint matching on smart card, International Journal of Pattern Recognition and Artificial Intelligence 27 (2013).
[10] B. Bhanu, X. Tan, Fingerprint indexing based on novel features of minutiae triplets, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (2003) 616–622.
[11] P. Bhowmick, B.B. Bhattacharya, Approximate fingerprint matching using kd-tree., in: International Conference on Pattern Recognition (ICPR (1)), 2004, pp. 544–s547.
[12] Z. Bian, D. Zhang, W. Shu, Knowledge-based fingerprint post-processing, International Journal of Pattern Recognition and Artificial Intelligence 16 (2002) 53–67.
[13] S. Bistarelli, F. Santini, A. Vaccarelli, An asymmetric fingerprint matching algorithm for java card tm, Pattern Analysis and Applications (2006) 359–376.
[14] J. Bohn, V. Despigel, Fingerprint skeleton matching based on local descriptor, in: IEEE 3rd International Conference on Biometrics: Theory, Applications and Systems, BTAS 2009, 2009.
[15] J. Bringer, V. Despiegel, Binary feature vector fingerprint representation from minutiae vicinities, in: IEEE 4th International Conference on Biometrics: Theory, Applications and Systems, BTAS 2010, 2010.
[16] M.J. Burge, K.W. Bowyer, Handbook of Iris Recognition, Springer Publishing Company, Incorporated, 2013.
[17] K. Cao, X. Yang, X. Chen, X. Tao, Y. Zang, J. Liang, J. Tian, Minutia handedness: A novel global feature for minutiae-based fingerprint matching., Pattern Recognition Letters 33 (2012) 1411–1421.
[18] K. Cao, X. Yang, X. Chen, Y. Zang, J. Liang, J. Tian, A novel ant colony optimization algorithm for large-distorted fingerprint matching., Pattern Recognition 45 (2012) 151–161.
[19] K. Cao, X. Yang, X. Tao, P. Li, Y. Zang, J. Tian, Combining features for distorted fingerprint matching, Journal of Network and Computer Applications 33 (2010) 258–267.
[20] K. Cao, X. Yang, X. Tao, Y. Zhang, J. Tian, A novel matching algorithm for distorted fingerprints based on penalized quadratic model, in: IEEE 3rd International Conference on Biometrics: Theory, Applications and Systems, BTAS 2009, 2009.
[21] K. Cao, X. Yang, J. Tian, Y. Zhang, P. Li, X. Tao, Fingerprint matching based on neighboring information and penalized logistic regression, in: International Conference on Advances in Biometrics (ICB), volume 5558 of *Lecture Notes in Computer Science*, 2009, pp. 617–626.
[22] R. Cappelli, Fast and accurate fingerprint indexing based on ridge orientation and frequency., IEEE Transactions on Systems, Man, and Cybernetics, Part B 41 (2011) 1511–1521.
[23] R. Cappelli, M. Ferrara, A fingerprint retrieval system based on level-1 and level-2 features., Expert Systems with Applications 39 (2012) 10465–10478.
[24] R. Cappelli, M. Ferrara, D. Maio, Candidate list reduction based on the analysis of fingerprint indexing scores., IEEE Transactions on Information Forensics and Security 6 (2011) 1160–1164.
[25] R. Cappelli, M. Ferrara, D. Maio, A fast and accurate palmprint recognition system based on minutiae, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 42 (2012) 956–962.
[26] R. Cappelli, M. Ferrara, D. Maltoni, Minutia cylinder-code: A new representation and matching technique for fingerprint recognition., IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (2010) 2128–2141.
[27] R. Cappelli, A. Lumini, D. Maio, D. Maltoni, Fingerprint classification by directional image partitioning., IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (1999) 402–421.
[28] C. Carvalho, H. Yehia, Fingerprint alignment using line segments., in: International Conference on Biometric Authentication (ICBA), volume 3072 of *Lecture Notes in Computer Science*, Springer, 2004, pp. 380–387.
[29] J.H. Cha, H. Jang, G.Y. Kim, H.I. Choi, Fingerprint matching based on linking information structure of minutiae., in: International Conference on Computational Science and its Applications (ICCSA (1)), volume 3043 of *Lecture Notes in Computer Science*, 2004, pp. 41–48.

[30] K.C. Chan, Y.S. Moon, P.S. Cheng, Fast fingerprint verification using subregions of fingerprint images, IEEE Transactions on Circuits Systems and Video Technology 14 (2004) 95–101.
[31] A.C. Chau, C.P. Soto, Hybrid algorithm for fingerprint matching using delaunay triangulation and local binary patterns, in: 16th Iberoamerican Congress on Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications (CIARP), volume 7042 of *Lecture Notes in Computer Science*, 2011, pp. 692–700.
[32] F. Chen, X. Huang, J. Zhou, Hierarchical minutiae matching for fingerprint and palmprint identification, IEEE Transactions on Image Processing 22 (2013) 4964–4971.
[33] K. Chen, A. Hu, Fingerprint matching using texture feature extracted from minutiae neighborhood, in: Proceedings - 4th International Conference on Computational Intelligence and Communication Networks, CICN 2012, 2012, pp. 322–326.
[34] X. Chen, J. Tian, J. Cheng, X. Yang, Segmentation of fingerprint images using linear classifier., EURASIP Journal of Advanced Signal Processing 4 (2004) 480–494.
[35] X. Chen, J. Tian, X. Yang, A new algorithm for distorted fingerprints matching based on normalized fuzzy similarity measure., IEEE Transactions on Image Processing 15 (2006) 767–776.
[36] X. Chen, J. Tian, X. Yang, Y. Zhang, An algorithm for distorted fingerprint matching based on local triangle feature set, IEEE Transactions on Information Forensics and Security 1 (2006) 169–177.
[37] J. Cheng, J. Tian, H. Chen, Fingerprint minutiae matching with orientation and ridge, in: International Conference on Biometric Authentication (ICBA), 2004, pp. 351–358.
[38] X. Cheng, S. Tulyakov, V. Govindaraju, Minutiae-based matching state model for combinations in fingerprint matching system, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2013, pp. 92–97.
[39] S. Chikkerur, A.N. Cartwright, V. Govindaraju, K-plet and coupled BFS: A graph based fingerprint representation and matching algorithm., in: International Conference on Biometrics (ICB), volume 3832 of *Lecture Notes in Computer Science*, 2006, pp. 309–315.
[40] S. Chikkerur, S. Pankanti, A. Jea, N.K. Ratha, R.M. Bolle, Fingerprint representation using localized texture features., in: International Conference on Pattern Recognition (ICPR (4)), 2006, pp. 521–524.
[41] S. Chikkerur, N.K. Ratha, Impact of singular point detection on fingerprint matching performance., in: workshop on Automatic Identification Advanced Technologies, 2005, pp. 207–212.
[42] H. Choi, K. Choi, J. Kim, Fingerprint matching incorporating ridge features with minutiae., IEEE Transactions on Information Forensics and Security 6 (2011) 338–345.
[43] V. Conti, G. Vitello, F. Sorbello, S. Vitabile, An advanced technique for user identification using partial fingerprint, in: Proceedings of the 7th International Conference on Complex, Intelligent, and Software Intensive Systems, CISIS 2013, 2013, pp. 236–242.
[44] J. Dai, J. Feng, J. Zhou, Robust and efficient ridge-based palmprint matching, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (2012) 1618–1632.
[45] J. Demšar, Statistical comparisons of classifiers over multiple data sets, The Journal of Machine Learning Research 7 (2006) 1–30.
[46] H. Deng, Q. Huo, Minutiae matching based fingerprint verification using delaunay triangulation and aligned-edge-guided triangle matching, in: Proceedings of the 5th International Conference on Audio- and Video-Based Biometric Person Authentication, AVBPA, 2005, pp. 270–278.
[47] K.C. Fan, C.W. Liu, Y.K. Wang, A randomized approach with geometric constraints to fingerprint verification., Pattern Recognition 33 (2000) 1793–1803.
[48] G. Fang, S.N. Srihari, H. Srinivasan, P. Phatak, Use of ridge points in partial fingerprint matching, in: SPIE: Biometric Technology for Human Identification IV, 2007.
[49] J. Feng, Combining minutiae descriptors for fingerprint matching., Pattern Recognition 41 (2008) 342–352.
[50] J. Feng, Z. Ouyang, A. Cai, Fingerprint matching using ridges, Pattern Recognition 39 (2006) 2131–2140.
[51] M. Ferrara, D. Maltoni, R. Cappelli, Noninvertible minutia cylinder-code representation, IEEE Transactions on Information Forensics and Security 7 (2012) 1727–1737.

[52] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, Journal of the American Statistical Association 32 (1937) 675–701.
[53] Z. Gao, X. You, L. Zhou, W. Zeng, A novel matching technique for fingerprint recognition by graphical structures, in: International Conference on Wavelet Analysis and Pattern Recognition, 2011, pp. 77–82.
[54] S. García, F. Herrera, An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons, Journal of Machine Learning Research 9 (2008) 2677–2694.
[55] R. Garg, S. Rane, A keypoint descriptor for alignment-free fingerprint matching, in: ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings, 2013, pp. 2994–2998.
[56] J. Gu, J. Zhou, C. Yang, Fingerprint recognition by combining global structure and local cues, IEEE Transactions on Image Processing 15 (2006) 1952–1964.
[57] H. Guesmi, H. Trichili, A.M. Alimi, B. Solaiman, Fingerprint verification system based on curvelet transform and possibility theory, Multimedia Tools and Applications, in press. DOI: 10.1007/s11042-013-1785-1 (2014) 1–20.
[58] Z. Guo, R.W. Hall, Parallel thinning with two-subiteration algorithms., Communications of the ACM 32 (1989) 359–373. Corrigendum: CACM 32(6): 759 (1989).
[59] P.D. Gutiérrez, M. Lastra, F. Herrera, J.M. Benitez, A high performance fingerprint matching system for large databases based on GPU, IEEE Transactions on Information Forensics and Security 9 (2014) 62–71.
[60] H. Hasan, S.A. Kareem, Fingerprint image enhancement and recognition algorithms: a survey, Neural Computing and Applications 23 (2013) 1605–1610.
[61] X. He, J. Tian, L. Li, Y. He, X. Yang, Modeling and analysis of local comprehensive minutia relation for fingerprint matching, IEEE Transactions on Systems, Man, and Cybernetics, Part B 37 (2007) 1204–1211.
[62] Y. He, J. Tian, L. Li, H. Chen, X. Yang, Fingerprint matching based on global comprehensive similarity, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (2006) 850–862.
[63] Y. He, J. Tian, X. Luo, T. Zhang, Image enhancement and minutiae matching in fingerprint verification, Pattern Recognition Letters 24 (2003) 1349–1360.
[64] L. Hong, A. Jain, Integrating faces and fingerprints for personal identification, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (1998) 1295–1307.
[65] L. Hong, Y. Wan, A. Jain, Fingerprint image enhancement: Algorithm and performance evaluation, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (1998) 777–789.
[66] K. Ito, H. Nakajima, K. Kobayashi, T. Aoki, T. Higuchi, A fingerprint matching algorithm using phase-only correlation, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E87-A (2004) 682–691.
[67] M.H. Izadi, L. Mirmohamadsadeghi, A. Drygajlo, Introduction of cylinder quality measure into minutia cylinder-code based fingerprint matching, in: 2012 IEEE 5th International Conference on Biometrics: Theory, Applications and Systems, BTAS 2012, 2012, pp. 353–358.
[68] A.K. Jain, Y. Chen, M. Demirkus, Pores and ridges: High-resolution fingerprint matching using level 3 features, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (2007) 15–27.
[69] A.K. Jain, J. Feng, Latent palmprint matching, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (2009) 1032–1047.
[70] A.K. Jain, J. Feng, Latent fingerprint matching, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (2011) 88–100.
[71] A.K. Jain, J. Feng, K. Nandakumar, Fingerprint matching, IEEE Computer 43 (2010) 36–44.
[72] A.K. Jain, L. Hong, R.M. Bolle, On-Line Fingerprint Verification, IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (1997) 302–314.
[73] A.K. Jain, S. Prabhakar, S. Chen, Combining multiple matchers for a high security fingerprint verification system, Pattern Recognition Letters 20 (1999) 1371–1379.
[74] A.K. Jain, S. Prabhakar, L. Hong, A multichannel approach to fingerprint classification., IEEE Trans-

actions on Pattern Analysis and Machine Intelligence 21 (1999) 348–359.
[75] A.K. Jain, S. Prabhakar, L. Hong, S. Pankanti, Filterbank-based fingerprint matching, IEEE Transactions on Image Proccessing 9 (2000) 846–859.
[76] A.K. Jain, A.A. Ross, K. Nandakumar, Introduction to Biometrics, Springer Publishing Company, Incorporated, 2011.
[77] T.Y. Jea, V. Govindaraju, A minutia-based partial fingerprint recognition system, Pattern Recognition 38 (2005) 1672–1684.
[78] J. Jia, L. Cai, P. Lu, X. Liu, Fingerprint matching based on weighting method and the svm., Neurocomputing 70 (2007) 849–858.
[79] R.M. Jiang, D. Crookes, Fpga-based minutia matching for biometric fingerprint image database retrieval, Journal of Real-Time Image Processing 3 (2008) 177–182.
[80] X. Jiang, M. Liu, A.C. Kot, Fingerprint retrieval for identification., IEEE Transactions on Information Forensics and Security 1 (2006) 532–542.
[81] X. Jiang, W.Y. Yau, Fingerprint minutiae matching based on the local and global structures, in: International Conference on Pattern Recognition (ICPR), 2000, pp. 6038–6041.
[82] X. Jiang, X. You, Y. Yuan, M. Gong, A method using long digital straight segments for fingerprint recognition, Neurocomputing 77 (2012) 28–35.
[83] Y. Jie, Y. Yi fang, Z. Renjie, S. Qifa, Fingerprint minutiae matching algorithm for real time system, Pattern Recognition 39 (2006) 143–146.
[84] A.T.B. Jin, D.N.C. Ling, O.T. Song, An efficient fingerprint verification system using integrated wavelet and fourier-mellin invariant transform., Image and Vision Computing 22 (2004) 503–513.
[85] K. Karu, A.K. Jain, Fingerprint classification, Pattern Recognition 29 (1996) 389–404.
[86] M. Khalil, D. Muhammad, M. Khan, K. Alghathbar, Singular points detection using fingerprint orientation field reliability., International Journal of Physical Sciences 5 (2010) 352–357.
[87] U.M. Khan, S.A. Khan, N. Ejaz, R.U. Rehman, A fingerprint verification system using minutiae and wavelet based features, in: 2009 International Conference on Emerging Technologies, ICET 2009, 2009, pp. 291–296.
[88] H. Khazaei, A. Mohades, Fingerprint matching algorithm based on voronoi diagram, in: Proceedings - The International Conference on Computational Sciences and its Applications, ICCSA 2008, 2008, pp. 433–440.
[89] A. Kisel, A. Kochetkov, J. Kranauskas, Fingerprint minutiae matching without global alignment using local structures, Informatica 19 (2008) 31–44.
[90] Z.M. Kovács-Vajna, A fingerprint verification system based on triangular matching and dynamic time warping, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (2000) 1266–1276.
[91] J.V. Kulkarni, B.D. Patil, R.S. Holambe, Orientation feature for fingerprint matching., Pattern Recognition 39 (2006) 1551–1554.
[92] R. Kumar, B.R.D. Vikram, Fingerprint matching using multi-dimensional ann., Engineering Applications of Artificial Intelligence 23 (2010) 222–228.
[93] D. Kwon, I.D. Yun, D.H. Kim, S.U. Lee, Fingerprint matching method using minutiae clustering and warping., in: International Conference on Pattern Recognition (ICPR (4)), 2006, pp. 525–528.
[94] D. Lee, K. Choi, J. Kim, A robust fingerprint matching algorithm using local alignment., in: International Conference on Pattern Recognition (ICPR (3)), 2002, pp. 803–806.
[95] S. Lee, H. seung Choi, K. Choi, J. Kim, Fingerprint-quality index using gradient components., IEEE Transactions on Information Forensics and Security 3 (2008) 792–800.
[96] P. Li, X. Yang, Q. Su, Y. Zhang, J. Tian, A novel fingerprint matching algorithm using ridge curvature feature, in: International Conference on Advances in Biometrics (ICB), volume 5558 of *Lecture Notes in Computer Science*, 2009, pp. 607–616.
[97] S. Li, A. Kot, Fingerprint combination for privacy protection, IEEE Transactions on Information Forensics and Security 8 (2013) 350–360.
[98] X. Liang, T. Asano, Fingerprint matching using minutia polygons, International Conference on Pattern Recognition (ICPR) 1 (2006) 1046–1049.

[99] X. Liang, A. Bishnu, T. Asano, A robust fingerprint indexing scheme using minutia neighborhood structure and low-order delaunay triangles., IEEE Transactions on Information Forensics and Security 2 (2007) 721–733.
[100] C. Liu, J. Cao, X. Gao, X. Fu, J. Feng, A novel fingerprint matching algorithm using minutiae phase difference feature, in: Proceedings - International Conference on Image Processing, ICIP, 2011, pp. 3201–3204.
[101] C. Liu, T. Xia, H. Li, A hierarchical hough transform for fingerprint matching., in: International Conference on Biometric Authentication (ICBA), volume 3072 of *Lecture Notes in Computer Science*, Springer, 2004, pp. 373–379.
[102] E. Liu, A.K. Jain, J. Tian, A coarse to fine minutiae-based latent palmprint matching, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (2013) 2307–2322.
[103] F. Liu, Q. Zhao, D. Zhang, A novel hierarchical fingerprint matching approach, Pattern Recognition 44 (2011) 1604–1613.
[104] L. Liu, T. Jiang, J. Yang, C. Zhu, Fingerprint registration by maximization of mutual information, IEEE Transactions on Image Processing 15 (2006) 1100–1110.
[105] M. Liu, P.T. Yap, Invariant representation of orientation fields for fingerprint indexing, Pattern Recognition 45 (2012) 2532–2542.
[106] X. Luo, J. Tian, Knowledge based fingerprint image enhancement., in: International Conference on Pattern Recognition (ICPR), 2000, pp. 4783–4786.
[107] X. Luo, J. Tian, Y. Wu, A minutia matching algorithm in fingerprint verification., in: International Conference on Pattern Recognition (ICPR), 2000, pp. 4833–4836.
[108] D. Maio, D. Maltoni, Direct gray-scale minutiae detection in fingerprints, IEEE Transactions on Pattern Analysis and Machine Inteligence 19 (1997) 27–40.
[109] D. Maio, D. Maltoni, Ridge-line density estimation in digital images, in: International Conference on Pattern Recognition (ICPR(1)), 1998, pp. 534–538.
[110] D. Maio, D. Maltoni, R. Cappelli, J. Wayman, A. Jain, FVC2000: fingerprint verification competition, Pattern Analysis and Machine Intelligence, IEEE Transactions on 24 (2002) 402–412.
[111] D. Maio, D. Maltoni, R. Cappelli, J. Wayman, A. Jain, FVC2004: Third fingerprint verification competition, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 3072 (2004) 1–7.
[112] D. Maio, D. Maltoni, R. Cappelli, J.L. Wayman, A.K. Jain, FVC2002: Second fingerprint verification competition, in: Pattern Recognition, 2002. Proceedings. 16th International Conference on, volume 3, 2002, pp. 811–814.
[113] D. Maltoni, D. Maio, A.K. Jain, S. Prabhakar, Handbook of Fingerprint Recognition, Springer Publishing Company, Incorporated, 2nd edition, 2009.
[114] K. Mao, G. Wang, C. Yu, Y. Jin, A novel multi-reference points fingerprint matching method, in: International Conference on Advances in Multimedia Modeling (MMM), volume 5371 of *Lecture Notes in Computer Science*, 2009, pp. 356–366.
[115] K. Mao, G. Wang, G. Yu, A novel fingerprint matching method by excluding elastic distortion, in: International Conference on Database Systems for Advanced Applications (DASFAA), volume 4947 of *Lecture Notes in Computer Science*, 2008, pp. 348–363.
[116] M.A. Medina-Pérez, M. García-Borroto, A.E. Gutierrez-Rodríguez, L. Altamirano-Robles, Improving fingerprint verification using minutiae triplets, Sensors 12 (2012) 3418–3437.
[117] M.A. Medina-Prez, A. Gutirrez-Rodrguez, M. Garca-Borroto, Improving fingerprint matching using an orientation-based minutia descriptor, in: 14th Iberoamerican Conference on Pattern Recognition: Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications (CIARP), volume 5856 of *Lecture Notes in Computer Science*, 2009, pp. 121–128.
[118] P. Meenen, A. Ashrafi, R.R. Adhami, The utilization of a taylor series-based transformation in fingerprint verification., Pattern Recognition Letters 27 (2006) 1606–1618.
[119] P.I. Mistry, C.N. Paunwala, Fusion fingerprint minutiae matching system for personal identification, in: 4th International Conference on Computing, Communications and Networking Technologies, ICCCNT

2013, 2013.
[120] K.A. Nagaty, An adaptive hybrid energy-based fingerprint matching technique., Image and Vision Computing 23 (2005) 491–500.
[121] K. Nandakumar, Fingerprint matching based on minutiae phase spectrum., in: International Conference on Biometrics (ICB), 2012, pp. 216–221.
[122] L. Nanni, A. Lumini, Local binary patterns for a hybrid fingerprint matcher, Pattern Recognition 41 (2008) 3461–3466.
[123] G.S. Ng, X. Tong, X. Tang, D. Shi, Adjacent orientation vector based fingerprint minutiae matching system, in: International Conference on Pattern Recognition (ICPR (1)), 2004, pp. 528–531.
[124] K. Nilsson, J. Bigun, Localization of corresponding points in fingerprints by complex filtering, Pattern Recognition Letters 24 (2003) 2135–2144.
[125] L. O'Gorman, J.V. Nickerson, An approach to fingerprint filter design., Pattern Recognition 22 (1989) 29–38.
[126] S. Pankanti, S. Prabhakar, A.K. Jain, On the individuality of fingerprints, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (2001) 1010–1025.
[127] G. Parziale, A. Niel, A fingerprint matching using minutiae triangulation., in: International Conference on Biometric Authentication (ICBA), volume 3072 of *Lecture Notes in Computer Science*, 2004, pp. 241–248.
[128] A.A. Paulino, J. Feng, A.K. Jain, Latent fingerprint matching using descriptor-based hough transform, IEEE Transactions on Information Forensics and Security 8 (2013) 31–45.
[129] D. Peralta, M. Galar, I. Triguero, O. Miguel-Hurtado, J.M. Benitez, F. Herrera, Minutiae filtering to improve both efficacy and efficiency of fingerprint matching algorithms, Engineering Applications of Artificial Intelligence 32 (2014) 37 – 53.
[130] D. Peralta, I. Triguero, R. Sanchez-Reillo, F. Herrera, J.M. Benitez, Fast fingerprint identification for large databases, Pattern Recognition 47 (2014) 588–602.
[131] V.N. Perminov, A.M. Fartukov, Method for fingerprint minutiae matching based on their alignment, Pattern Recognition and Image Analysis 17 (2007) 631–638.
[132] J. Qi, Y. Wang, A robust fingerprint matching method, Pattern Recognition 38 (2005) 1665–1671.
[133] J. Qi, M. Xie, W. Wang, A novel fingerprint matching method using a curvature-based minutia specifier, in: Proceedings - International Conference on Image Processing, ICIP, 2008, pp. 1488–1491.
[134] J. Qi, S. Yang, Y. Wang, Fingerprint matching combining the global orientation field with minutia, Pattern Recognition Letters 26 (2005) 2424–2430.
[135] C.J. Ran, M. Xie, A new fingerprint matching method based on ridge tracing, in: Proceedings of the 2007 International Conference on Wavelet Analysis and Pattern Recognition, ICWAPR '07, volume 3, 2007, pp. 402–407.
[136] N. Ratha, R. Bolle, V. Pandit, V. Vaish, Robust fingerprint authentication using local structural similarity, in: Workshop on Applications of Computer Vision, 2000, pp. 29–34.
[137] N.K. Ratha, S. Chen, A.K. Jain, Adaptive flow orientation-based feature extraction in fingerprint images., Pattern Recognition 28 (1995) 1657–1672.
[138] N.K. Ratha, K. Karu, S. Chen, A.K. Jain, A real-time matching system for large fingerprint databases., IEEE Transactions on Pattern Analysis and Machine Intelligence 18 (1996) 799–813.
[139] K. Rerkrai, V. Areekul, A new reference point for fingerprint recognition., in: International Conference on Image Processing (ICIP), 2000, pp. 499–502.
[140] A. Ross, S.C. Dass, A.K. Jain, A deformable model for fingerprint matching., Pattern Recognition 38 (2005) 95–103.
[141] A. Ross, A.K. Jain, J. Reisman, A hybrid fingerprint matcher., Pattern Recognition 36 (2003) 1661–1673.
[142] L. Sha, X. Tang, Orientation-improved minutiae for fingerprint matching, in: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR) Volume 4, 2004, pp. 432–435.
[143] L. Sha, F. Zhao, X. Tang, Minutiae-based fingerprint matching using subset combination, in: International Conference on Pattern Recognition (ICPR (4)), 2006, pp. 566–569.

[144] W. Sheng, G. Howells, M. Fairhurst, F. Deravi, K. Harmer, Consensus fingerprint matching with genetically optimised approach, Pattern Recognition 42 (2009) 1399–1407.
[145] W. Sheng, G. Howells, M.C. Fairhurst, F. Deravi, A memetic fingerprint matching algorithm., IEEE Transactions on Information Forensics and Security 2 (2007) 402–412.
[146] J. Shi, K.Y. Lam, Minucode: A fixed-value representation of fingerprint minutiae for biometric cryptosystem, in: International Conference on Advances in Information Security and Assurance (ISA), volume 5576 of *Lecture Notes in Computer Science*, 2009, pp. 382–391.
[147] Z. Shi, V. Govindaraju, Robust fingerprint matching using spiral partitioning scheme, in: International Conference on Advances in Biometrics (ICB), volume 5558 of *Lecture Notes in Computer Science*, 2009, pp. 647–655.
[148] F. Su, P. Sun, L. Wang, X. Xie, An efficient minutiae-based fingerprint matching algorithm for resource constrained implementation, in: Proceedings - 2010 2nd IEEE International Conference on Network Infrastructure and Digital Content, IC-NIDC 2010, 2010, pp. 214–218.
[149] V.A. Sujan, M.P. Mulqueen, Fingerprint identification using space invariant transforms., Pattern Recognition Letters 23 (2002) 609–619.
[150] X. Tan, B. Bhanu, A robust two step approach for fingerprint identification, Pattern Recognition Letters 24 (2003) 2127–2134.
[151] X. Tan, B. Bhanu, Fingerprint matching by genetic algorithms, Pattern Recognition 39 (2006) 465–477.
[152] N.T.H. Thuy, H.X. Huan, N.N. Ky, An efficient method for fingerprint matching based on local point model, in: International Conference on Computing, Management and Telecommunications, ComManTel 2013, 2013, pp. 334–339.
[153] M. Tico, P. Kuosmanen, An algorithm for fingerprint image postprocessing, in: In Proceedings of the Thirty-Fourth Asilomar Conference on Signals, Systems and Computers, 2000, pp. 1735–1739.
[154] M. Tico, P. Kuosmanen, Fingerprint matching using an orientation-based minutia descriptor., IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (2003) 1009–1014.
[155] X. Tong, J. Huang, X. Tang, D. Shi, Fingerprint minutiae matching using the adjacent feature vector., Pattern Recognition Letters 26 (2005) 1337–1345.
[156] X. Tong, S. Liu, J. Huang, X. Tang, Local relative location error descriptor-based fingerprint minutiae matching, Pattern Recognition Letters 29 (2008) 286–294.
[157] R. Udupa, G. Garg, P.K. Sharma, Fast and accurate fingerprint verification., in: International Conference on Audio and Video based Biometric Person Authentication (AVBPA), volume 2091 of *Lecture Notes in Computer Science*, 2001, pp. 192–197.
[158] T. Uz, G. Bebis, A. Erol, S. Prabhakar, Minutiae-based template synthesis and matching for fingerprint authentication, Computer Vision and Image Understanding 113 (2009) 979–992.
[159] K. Venkataramani, B.V.K.V. Kumar, Performance of composite correlation filters in fingerprint verification., Optical Engineering 43 (2004) 1820–1827.
[160] M.A. Wahby-Shalaby, M. Omair Ahmad, A multilevel structural technique for fingerprint representation and matching, Signal Processing 93 (2013) 56–69.
[161] D. Wan, J. Zhou, Fingerprint recognition using model-based density map, IEEE Transactions on Image Processing 15 (2006) 1690–1696.
[162] C. Wang, G. Ding, Z. Zheng, Fingerprint matching combining the adjacent feature with curvature of ridges, in: Proceedings of the World Congress on Intelligent Control and Automation (WCICA), 2008, pp. 6807–6810.
[163] W. Wang, J. Li, W. Chen, Fingerprint minutiae matching based on coordinate system bank and global optimum alignment, in: International Conference on Pattern Recognition (ICPR (4)), 2006, pp. 401–404.
[164] X. Wang, J. Li, Y. Niu, Fingerprint matching using orientationcodes and polylines., Pattern Recognition 40 (2007) 3164–3177.
[165] C.I. Watson, M.D. Garris, E. Tabassi, C.L. Wilson, R.M. Mccabe, S. Janet, K. Ko, User's Guide to NIST Biometric Image Software (NBIS), Technical Report, NIST, 2010.

[166] H. Wei, M. Guo, Z. Ou, Fingerprint verification based on multistage minutiae matching., in: International Conference on Pattern Recognition (ICPR (2)), 2006, pp. 1058–1061.
[167] H. Wei, D. Liu, A multi-stage fingerprints matching algorithm, in: Proceedings of the 2009 IEEE International Conference on Automation and Logistics, ICAL 2009, 2009, pp. 197–199.
[168] C.L. Wilson, C.I. Watson, E.G. Paek, Effect of resolution and image quality on combined optical and neural network fingerprint matching., Pattern Recognition 33 (2000) 317–331.
[169] X. Xie, F. Su, A. Cai, Ridge-based fingerprint recognition., in: International Conference on Biometrics (ICB), volume 3832 of *Lecture Notes in Computer Science*, 2006, pp. 273–279.
[170] X. Xie, F. Su, A. Cai, J. Sun, A robust fingerprint minutiae matching algorithm based on the support model., in: International Conference on Biometric Authentication (ICBA), volume 3072 of *Lecture Notes in Computer Science*, 2004, pp. 316–323.
[171] H. Xu, R.N.J. Veldhuis, T.A.M. Kevenaar, A.H.M. Akkermans, A fast minutiae-based fingerprint recognition system., IEEE Systems Journal 3 (2009) 418–427.
[172] W. Xu, X. Chen, J. Feng, A robust fingerprint matching approach: Growing and fusing of local structures., in: International Conference on Biometrics (ICB), volume 4642 of *Lecture Notes in Computer Science*, 2007, pp. 134–143.
[173] L. Xuzhou, Y. Fei, A new fingerprint matching algorithm based on minutiae, in: Proceedings of 2009 IEEE International Conference on Communications Technology and Applications, IEEE ICCTA2009, 2009, pp. 869–873.
[174] N. Yager, A. Amin, Fingerprint verification based on minutiae features: a review, Pattern Analysis and Applications 7 (2004) 94–113.
[175] N. Yager, A. Amin, Fingerprint alignment using a two stage optimization., Pattern Recognition Letters 27 (2006) 317–324.
[176] J. Yang, S. Xie, S. Yoon, D. Park, Z. Fang, S. Yang, Fingerprint matching based on extreme learning machine, Neural Computing and Applications 22 (2013) 435–445.
[177] W. Yang, J. Hu, M. Stojmenovic, Ndtc: A novel topology-based fingerprint matching algorithm using n-layer delaunay triangulation net check, in: Proceedings of the 2012 7th IEEE Conference on Industrial Electronics and Applications, ICIEA 2012, 2012, pp. 866–870.
[178] Y. Ying, H. Zhang, X. Yang., A method based on delaunay triangulation for fingerprint matching., in: SPIE Conference on Biometric Technology for Human Identification II, 2005.
[179] K.D. Yu, S. Na, T.Y. Choi, A fingerprint matching algorithm based on radial structure and a structure-rewarding scoring strategy., in: International Conference on Audio- and Video-based Biometric Person Authentication (AVBPA), volume 3546 of *Lecture Notes in Computer Science*, 2005, pp. 656–664.
[180] T.Y. Zhang, C.Y. Suen, A fast parallel algorithm for thinning digital patterns, Communications of the ACM 27 (1984) 236–239.
[181] W. Zhang, Y. Wang, Core-based structure matching algorithm of fingerprint verification, in: International Conference on Pattern Recognition (ICPR (1)), 2002, pp. 70–74.
[182] Y. Zhang, X. Yang, Q. Su, J. Tian, Fingerprint recognition based on combined features., in: International Conference on Biometrics (ICB), volume 4642 of *Lecture Notes in Computer Science*, 2007, pp. 281–289.
[183] D. Zhao, F. Su, A. Cai, Fingerprint registration using minutia clusters and centroid structure, in: International Conference on Pattern Recognition (ICPR (4)), 2006, pp. 413–416.
[184] F. Zhao, X. Tang, Preprocessing and postprocessing for skeleton-based fingerprint minutiae extraction., Pattern Recognition 40 (2007) 1270–1281.
[185] X. Zhao, X. Zhang, G. Zhao, X. Li, K. Zhang, R. Qian, Triangle matching combined with singular features in fingerprints, in: Proceedings 2011 International Conference on Mechatronic Science, Electric Engineering and Computer, MEC 2011, 2011, pp. 2069–2072.
[186] J.D. Zheng, Y. Gao, M.Z. Zhang, Fingerprint matching algorithm based on similar vector triangle, in: Proceedings of the 2009 2nd International Congress on Image and Signal Processing, CISP'09, 2009.
[187] X. Zheng, Y. Wang, Fingerprint matching based on ridge similarity, in: ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings, 2008, pp. 1701–1704.

[188] W.B. Zhong, X.B. Ning, C.J. Wei, A fingerprint matching algorithm based on relative topological relationship among minutiae, in: 2008 IEEE International Conference Neural Networks and Signal Processing, ICNNSP, 2008, pp. 225–228.
[189] E. Zhu, J. Yin, G. Zhang, Fingerprint matching based on global alignment of multiple reference minutiae, Pattern Recognition 38 (2005) 1685–1694.

2 Minutiae Filtering to Improve Both Efficacy and Efficiency of Fingerprint Matching Algorithms

- D. Peralta, M. Galar, I. Triguero, O. Miguel-Hurtado, J.M. Benítez, F. Herrera. Minutiae Filtering to Improve Both Efficacy and Efficiency of Fingerprint Matching Algorithms. Engineering Applications of Artificial Intelligence, 32 (2014) 37–53. doi: 10.1016/j.engappai.2014.02.016

  - Status: **Published**.
  - Impact Factor (JCR 2014): 2.207
  - Subject Category: Automation & Control Systems Science. Ranking 16 / 58 (**Q2**).
  - Subject Category: Computer Science, Artificial Intelligence. Ranking 30 / 123 (**Q1**).
  - Subject Category: Engineering, Multidisciplinary. Ranking 12 / 85 (**Q1**).
  - Subject Category: Engineering, Electrical & Electronic. Ranking 51 / 249 (**Q1**).

Minutiae Filtering to Improve Both Efficacy and Efficiency of Fingerprint Matching Algorithms

Daniel Peralta[a], Mikel Galar[b], Isaac Triguero[a], Oscar Miguel-Hurtado[c], Jose M. Benitez[a], Francisco Herrera[a]

[a]*Department of Computer Science and Articial Intelligence, CITIC-UGR (Research Center on Information and Communications Technology), University of Granada, 18071 Granada, Spain*
[b]*Department of Automatics and Computation, Universidad Pública de Navarra, 31006 Pamplona, Spain*
[c]*Instituto Científico de Innovación y Tecnologías Aplicadas (INCITA), R and D Department, C/Santa Leonor 65, Bloque C, Planta 2, 28037 Madrid, Spain*

Abstract

Fingerprint minutiae extraction is a critical issue in fingerprint recognition. Both missing and spurious minutiae hinder the posterior matching process. Spurious minutiae are more frequent than missing ones, but they can be removed by post-processing. In this work, we study the usage of a state-of-the-art minutiae extractor, MINDTCT, and we analyze its major drawback: the presence of spurious minutiae lying on the borders of the fingerprint and out its area. In order to overcome this problem, we use two different filtering approaches based on the convex hull of the minutiae and the segmentation of the fingerprint. We will analyze, supported by an exhaustive experimental study, the efficacy of these methods to remove spurious minutiae. We will evaluate both the effect on different state-of-the-art matchers and the goodness of the minutiae, by comparing the extracted minutiae with the ground-truth ones. For this purpose, the experiments have been performed on several databases of both real and synthetic fingerprints. The filters used allow us to remove spurious minutiae, resulting in more accurate results even in the case of robust matchers. The EER is improved up to 2% for good quality databases, and up to 25% for FVC databases. Additionally, the matching time is accelerated, since less minutiae are processed, attaining up to a 60% runtime reduction for the tested database.

Keywords: Fingerprint recognition, Minutiae filtering, Fingerprint segmentation, Fingerprint enhancement

1. Introduction

Fingerprints are the most used features for biometric identification. Fingerprints are present on the surface of fingertips and they are patterns formed of ridges and valleys. Their individuality, which is determined by the local ridge characteristics and their relationships (Hong et al., 1998), makes them appropriate for identification purposes, since each individual has unique fingerprints. The characteristics or features of a fingerprint are usually classified into three levels (Maltoni et al., 2009; Feng and Jain, 2011):

- *Level 1 (Global)* refers to the global ridge line flow (orientations) and the features derived from it (singular points).
- *Level 2 (Local)* considers minutiae details extracted from the ridge skeleton.
- *Level 3 (Fine-detail)* includes intra-ridge details such as width, shape, ridge contours, sweat pores, creases, etc.

Level 2 features (minutiae) are the most commonly used ones for fingerprint matching (Jiang and Yau, 2000; Deng and Huo, 2005; Chen et al., 2006; Cappelli et al., 2010), that is, to check whether two fingerprints belong to the same individual. Notice that neither Level 1 features nor Level 3 features are usually considered for matching. The former ones because their distinctiveness is not sufficient to accurately perform the matching, whereas the latter ones because they require high quality fingerprints, which are not usually available.

In the last years, fingerprint recognition has acquired a big importance due to its advantages for identifying people, but there are also legal concerns about its use. One of the hot topics in research is the encoding of fingerprints, which seeks to store the fingerprint in the database in a template format from which the original image cannot be retrieved. There are many minutiae-based algorithms that perform this task (Lee and Kim, 2010; Ahmad et al., 2011).

A minutia is defined as a local discontinuity in the fingerprint pattern (ridge skeleton). Many minutiae types can be found in fingerprints, but only two of them are considered by most of the Automatic Fingerprint Identification Systems (AFISs): ridge endings and ridge bifurcations. The usage of minutiae provides several advantages to the matching process: they are distinctive, compact and human experts also use them to match fingerprints. Nevertheless, the extraction of minutiae from fingerprint images is a difficult task (Maio and Maltoni, 1997; Hong et al., 1998). Moreover, the difficulty increases as the quality of the fingerprint is degraded.

As a consequence, minutiae extraction becomes a key component in the development of AFISs (Ratha and Bolle, 2004; Maltoni et al., 2009). Two different types of errors can be attributed to these methods: missing minutiae (non-detected real minutiae) and spurious minutiae (non-existing detected minutiae). Such errors might be produced by poor-quality images, but also due to creases or scars in the fingerprint pattern. In addition to these errors, the quality of the minutiae should also be assessed, that is, how close are the estimated minutiae positions and angles from the real minutiae ones (Ratha et al., 1995; Gao et al., 2010). Regarding erroneous minutiae, a missing minutia can only be recovered (detected) improving the minutiae extraction method; otherwise, a spurious minutiae can be detected and removed from the minutiae set by post-processing techniques (Chikkerur et al., 2005).

This post-processing step is of great importance, since a large number of spurious minutiae are usually detected on poor-quality fingerprints, whereas the number of correct minutiae detected might be enough to perform a successful matching (around 12 correctly matched minutiae are usually sufficient to claim the individuality of a fingerprint (Dass, 2010)). The removal of spurious minutiae, maintaining the correct ones, improves the results of the matching process (Hong et al., 1998).

In this paper, we focus on the well-known MINDTCT minutiae extractor, which is provided with NBIS software package (Watson et al., 2010). Our aim is to study the behavior of two different approaches to remove borderline minutiae and thus enhance the minutiae set given by MINDTCT: the usage of the convex hull formed by the minutiae and a segmentation-based approach (presented in Section 3). In this manner, we have a two-fold objective:

- First, we aim to improve the minutiae set obtained by MINDTCT eliminating the spurious borderline minutiae, using the two mentioned approaches. Then, we will compare the quality of the original and the filtered minutiae sets to validate the studied post-processing mechanisms.
- Second, we will investigate the influence of the spurious minutiae on several matching methods to show that an adequate post-processing can be effective to enhance the results obtained in AFISs. More specifically, we will show that robust methods are not severely affected by spurious minutiae in terms of accuracy, whereas simpler ones can be highly influenced by their presence. Furthermore, reducing the number of minutiae the computational complexity of the matching is reduced, obtaining faster matching times, which are also evaluated in this paper.

In order to carry out these objectives, we have developed an exhaustive experimental study, where we aim to evaluate both the quality of the minutiae extracted (with and without filtering) and the effect of the filtering methods in different state-of-the-art matching methods (both in terms of accuracy and complexity). In total, twelve databases of three different types have been evaluated: five databases artificially generated with SFinGe (Cappelli et al., 2004; Maltoni et al., 2009), six databases from the FVC competitions, and one database captured by the authors' research groups. The parallel architecture presented in (Peralta et al., 2014) has been used to allow the execution of huge amounts of matches in a reasonable time. We will show that almost 75% of the spurious minutiae detected by MINDTCT can be removed by an adequate post-processing, highly reducing the error rates of the matchers (the reduction varies depending on their robustness) and their execution times.

The rest of the paper is organized as follows. In Section 2.1, we recall several related works to minutiae filtering and quality evaluation. In Section 2.2, we describe the quality measures used in this paper. In Section 3, we present two different post-processing methods for borderline minutiae filtering. The experimental framework used to develop the experiments is presented in Section 4. Next, Section 5 presents the experimental study carried out to evaluate both the quality of the minutiae and the performance of the matching methods. Finally, Section 6 concludes the paper and presents some future research lines based on the obtained results.

2. Fingerprint minutiae

In this section, we first recall some related works to minutiae extraction and post-processing. Next, we review some metrics already used to assess the quality of the extracted minutiae with respect to the ground-truth ones.

2.1. Minutiae extraction in AFISs

Two types of minutiae detection algorithms can be found in the specialized literature depending on how they deal with the fingerprint image. Binarization-based methods (Jain et al., 1997; Watson et al., 2010) carry out a binarization process followed by the thinning of the obtained image, producing a new image from which the minutiae can be easily extracted. Thus, errors in both phases bring along the detection of spurious minutiae. Otherwise, gray-level intensities-based methods (Maio and Maltoni, 1997; Jiang et al., 2001) directly extract the minutiae from the gray-scale image without requiring to pre-process the image. Although

the pre-processing is avoided, these methods also produce spurious minutiae in low quality fingerprints. Hence, independently of the method used to extract the minutiae, two types of errors can be produced, as we have previously mentioned: spurious and missing minutiae.

Focusing on the spurious minutiae, post-processing techniques can be considered in order to prune them. Two different approaches can be found in the literature (Chikkerur et al., 2005; Maltoni et al., 2009):

1. *Structural post-processing*: heuristics based on the relative location and the length of the ridges, among other structural information, are used to remove minutiae (Jiang et al., 2001). These rules are usually strongly related to the minutiae extraction algorithm. For example, in Xiao and Raafat (1991), a number of structures resulting from the thinning of the image leading to spurious minutiae were identified and different heuristics were proposed to eliminate such minutiae. In Hung (1993) and Zhao and Tang (2007), the authors take advantage of the duality property of the ridge endings and bifurcations using the negative and positive gray-level images to detect and prune minutiae.
2. *Filtering based on gray-level*: the gray-scale values in the neighborhood of the minutiae are used to verify the minutiae. Most of these approaches rely on previously labeled spurious and correct minutiae to train a classifier able to decide whether each minutia is spurious or not. In Prabhakar et al. (2000), the gray-scale values are directly used to train a Learning Vector Quantifier classifier. Neural networks are used in Maio and Maltoni (1998), Santhanam et al. (2007) and Kumar and Deva Vikram (2010) to learn to filter minutiae with different pre-processing steps and features to represent the minutiae as the input for the classifier.

MINDTCT extracts the minutiae from an input image following six steps: 1) generation of image maps, 2) binarization of the image, 3) detection of the initial minutiae set, 4) removal of spurious minutiae, 5) ridge counting between neighboring minutiae and 6) assessment of the minutiae quality. Both the 4^{th} and the 6^{th} steps are related to minutiae post-processing. The former aims to remove spurious minutiae by structural post-processing, whereas the latter assigns a quality index to each minutia, which allows for further processing of the minutiae. However, despite these processes, a certain number of spurious minutiae are still present in the extracted minutiae set, as it can be observed in Figure 1. Observe that most of the spurious minutiae lie on the border between the fingerprint and the background.

Although we are centering our attention on MINDTCT, we should notice that

there are several minutiae extractors in the literature (Maio and Maltoni, 1997; Jain et al., 1997; Gao et al., 2010), but MINDTCT is still competitive, despite a number of spurious minutiae are detected (Dass, 2010).

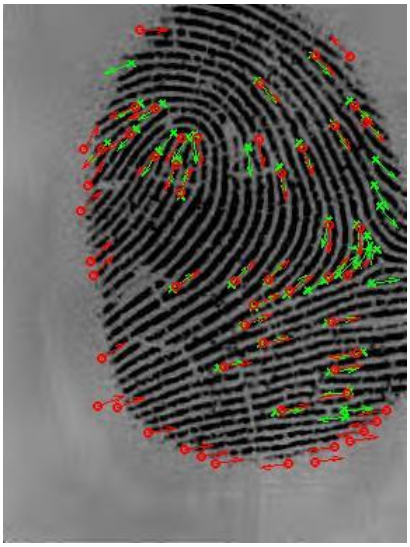


Figure 1: Ground-truth (marked with green crosses) and extracted (drawn as red circles) minutiae.

2.2. Minutiae Quality Evaluation

Minutiae quality can be understood in two ways. It can be related to the confidence of the minutiae extractor algorithm on the extracted minutia. In this case, the quality might help the matching process to reject fingerprints, remove low quality minutiae or assign weights depending on the quality. The quality given to each minutia by MINDTCT is an example of this type of quality. In MINDTCT the quality is assigned depending on the gray-level of the pixels in the neighborhood of the minutiae and on the local quality assigned to the minutiae location (quality map). However, this quality is not adequate by itself for post-processing, since low quality is assigned to areas near singular points, even though the minutiae are correctly detected. This fact will be clearly shown carrying out several experiments using the quality as filtering criterion (Section 5).

In this paper, we refer to minutiae quality (and its evaluation) as the quality of the minutiae obtained with respect to the ground-truth ones. However, obtaining ground-truth minutiae is a very difficult task. For this reason, we take advantage of the SFinGe software tool (Cappelli et al., 2004; Maltoni et al., 2009), which

generates realistic synthetic fingerprints, allowing a straightforward evaluation of AFISs. Moreover, since the fingerprints are generated from minutiae, the ground-truth minutiae become available (among other interesting ground-truth data) and hence, it makes possible to properly evaluate the performance of minutiae post-processing methods and to observe their influence on different matching algorithms. Furthermore, the experiments carried out with SFinGe are easily reproducible by other researchers using the same parameters for the generation process, which are provided in Section 4. In addition, we carry out an indirect evaluation of the minutiae in the fingerprint verification and identification problem considering databases from the FVC and a real one.

In the specialized literature the Goodness Index (GI) has been considered to measure the quality of minutiae with respect to the ground-truth ones (Hong et al., 1998; Zhao and Tang, 2002). The GI combines the number of correctly detected minutiae (paired with ground-truth minutiae, P), the number of spurious minutiae (S) and the number of missing minutiae (M) in a unique value, easing the comparison between different methods. Its computation is shown in Equation 1, where T is the total number of ground-truth minutiae. Notice that for the minutiae pairing a tolerance box centered around each ground-truth minutia is used (in our experiments this box is a 10 pixels radius circle). The maximum value of GI is 1, meaning that all ground-truth minutiae are correctly paired with the corresponding detected minutiae and there are no missing minutiae ($P = T$ and $M = S = 0$). Hence, the greater the GI is, the greater the quality of the detected minutiae is.

$$\text{GI} = \frac{P - M - S}{T} \tag{1}$$

Besides from the GI, other metrics to quantify the quality of the minutiae have been used in the literature. That is the case of the True Positive Rate (TPR) and the Positive Predictive Value (PPV). The TPR is the percentage of correctly detected minutiae with respect to the number of ground-truth minutiae (TPR $= P/T = P/(P + M)$), whereas the PPV is the percentage of correctly detected minutiae among all the minutiae detected (PPV $= P/(P+S)$). Hence, the former is related to the number of missing minutiae, whereas the latter is related to the number of spurious minutiae.

In addition to these measures quantifying the accuracy of the minutiae, in the sense that they measure the proportion of correctly detected minutiae, we propose other metrics in Section 4.3.1 to measure, among the correctly detected minutiae, how close are the localization and the orientation estimations of the minutiae with respect to the corresponding ground-truth ones. This measurement is interesting in

order to analyze the deviations produced by the minutiae extractors. Reasonably, if a minutia is correctly detected but its localization or angle difference is large, the extracted minutiae can be considered as spurious, whereas the real one can be considered as missing.

3. Post-processing Methods for Borderline Minutiae Filtering

In this section we describe two alternatives for minutiae filtering. The former uses a convex hull approach to find spurious minutiae (Section 3.1), whereas the latter considers the segmentation of the fingerprint to filter the minutiae in the borders (Section 3.2).

3.1. Convex hull-based filtering

The convex hull of a set of points S is defined as the convex polygon that contains all the elements of S with the smallest area. Figure 2 depicts an example of the convex hull for a set of points. It shows the concept of convex hull as the analogy of an elastic-band that surrounds the complete set S and then is released, so that the enclosed points form the convex hull.

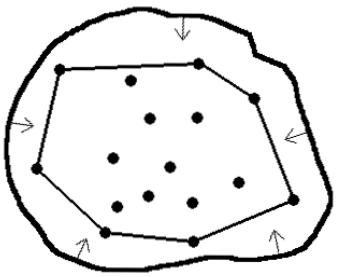


Figure 2: Convex hull example. Elastic-band analogy.

This idea has been widely used in the specialized literature as a simple, but efficient, mechanism to remove spurious minutiae (Wen and Guo, 2009; Cappelli et al., 2010). In this work, we implement the convex hull idea in conjunction with the quality assessment mechanism of MINDTCT, providing an intuitive way to filter the minutiae set.

A minutia m_i is formed by four components $(x_i, y_i, \theta_i, q_i)$, where:

- (x_i, y_i) are the coordinates in the fingerprint image.
- θ_i is the orientation or minutia angle.

- q_i refers to the quality of the minutia.

Thus, after the minutiae detection process accomplished by MINDTCT, a fingerprint F can be represented as a vector of r minutiae $\mathbf{m} = \{m_1, m_2, ..., m_r\}$. To determine which minutiae of a given fingerprint F are susceptible of being spurious, we focus on their coordinates x_i and y_i and their quality q_i (with $i \in \{1, \ldots, r\}$).

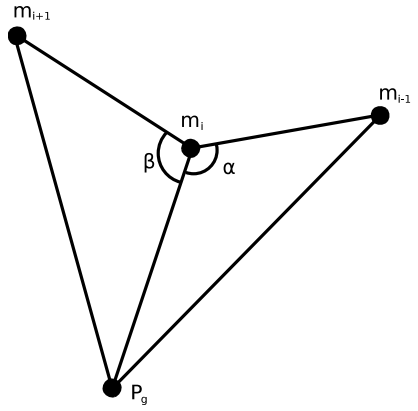


Figure 3: Example of minutiae triplet in the Graham's scan algorithm. In this case, the minutiae m_i does not belong to the convex hull.

Initially, the convex hull of the fingerprint minutiae set is calculated using the Graham's scan algorithm (Graham, 1972). This algorithm places a random point P_g among the minutiae, and converts their coordinates to polar. Then, the minutiae are ordered by their newly calculated polar angle. The minutiae are grouped in triplets according to their order, forming triangles as depicted in Figure 3. The minutiae m_i belongs to the convex hull if and only if $\alpha + \beta < \pi$. The process is repeated for all triplets, obtaining the set of minutiae that form the convex hull ($\mathbf{m}_{CH}$).

Then, minutiae in $\mathbf{m}_{CH}$ with their quality lower than the threshold ϕ are included in the set of candidate spurious minutiae $\mathbf{m}_s$. These minutiae are removed from the original minutiae set $\mathbf{m}$. If the resulting set has at least r_{min} minutiae, it is taken as the final filtered minutiae set. Otherwise, the information loss is considered too high and the filtering is not applied. Note that if ϕ is set to the maximum quality value (100), the algorithm will remove all the minutiae in the convex hull.

The main highlights of the convex hull approach are its simplicity and intuitiveness. The convex hull can be very easily superposed to the fingerprint image,

and the correctness of the filtered minutiae can be visually checked. As for the computational impact, our implementation has the complexity of the Graham's scan algorithm, which is of O(n log n), with n being the number of minutiae of the fingerprint. Thus, the convex hull computation is reasonably fast.

However, this method has some drawbacks. The most important one is that it is very sensitive to image translations, as it can be clearly observed in Figure 4, which shows two captures of the same fingerprint. The fingerprint in Figure 4a is correctly centered, and the convex hull is correctly detected, including most of the image spurious minutiae. Otherwise, the fingerprint in Figure 4b is translated and many of the minutiae that lie on the image borders are not spurious. The usage of the MINDTCT quality parameter that we propose intends to reduce the impact of this kind of translations. However, the quality assignment process of MINDTCT assigns low qualities to the minutiae near the image borders, even when they are correctly detected, and this may difficult the filtering process.

(a) (b)

Figure 4: Convex hull of two captures of the same fingerprint.

3.2. Segmentation-based filtering

This method consists of detecting the fingerprint area in the image, that is, to define a segmentation mask for the fingerprint. Once the mask is defined, all the minutiae that were detected out of the fingerprint region or near the borders are

deleted. In such a way, we aim to decrease the number of spurious minutiae while maintaining the number of correctly detected ones.

The segmentation process combines the concepts presented in Ratha et al. (1995); Hong et al. (1998); Bazen and Gerez (2001). A fingerprint image usually has two well-differentiated areas: the background and the fingerprint itself. In the former there are usually low variations of the gray-level intensities, since it tends to be homogeneous (although the intensity may vary and can be different depending on the device used). In opposite, the latter presents a high variance of gray-level intensities due to the presence of ridges and valleys. On this account, an effective methodology for fingerprint segmentation can be developed considering the local variance (block-wise variance) of the pixel intensity values.

The operating procedure of the whole algorithm is presented hereafter:

1. *Normalization*: A desired mean M_0 and variance V_0 for the fingerprint image are established. The image is then normalized in such a way that its mean and variance take values M_0 and V_0, respectively. This pre-processing reduces the gray-level variations along ridges and valleys, which facilitates further processing. In our case, this phase allows us to set a global threshold for all the images. Normalization is a pixel-wise operation in which a new image I_{norm} is created starting from the original image I (with original mean and variance M and V, respectively) as follows.

$$I_{norm}(i,j) = \begin{cases} M_0 + \sqrt{\frac{V_0 \cdot (I(i,j)-M)^2}{V}} \text{ if } I(i,j) > M \\ M_0 - \sqrt{\frac{V_0 \cdot (I(i,j)-M)^2}{V}} \text{ otherwise.} \end{cases} \quad (2)$$

2. *Block-wise variance computation*: The gray-level variance of each block in the normalized image is computed (blocks of 8×8 pixels are used in our experiments).
3. *Thresholding*: Each block is assigned to the background or to the fingerprint depending on its variance following a global threshold (T_v). Blocks with variance greater than the threshold belong to the fingerprint, whereas the rest are assigned to the background.
4. *Refinement*: In order to obtain a unique fingerprint area, three iterations of hole filling are carried out, where blocks discordant with more than half of its 8-neighbors are changed. Then, an erosion process is performed to ensure that the fingerprint region does not contain any background zone.

After carrying out the segmentation of the fingerprint, all the minutiae located on blocks belonging to the background and whose quality is lower than ϕ

are pruned. In addition to these minutiae, those lying on the borderline blocks, that is, in blocks having a background block in its 8-neighborhood, are also removed. An example of the application of this process to the minutiae obtained using MINDTCT algorithm can be shown in Figure 5. The shaded area corresponds to the background detected by the segmentation algorithm. We can observe that the studied method has effectively removed a great amount of spurious minutiae lying on the borders of the fingerprint.

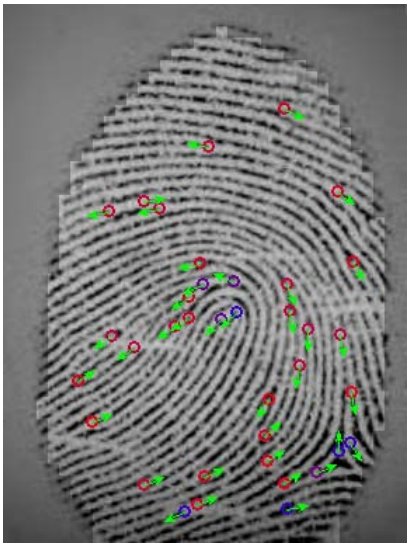


Figure 5: Minutiae detected by MINDTCT algorithm and filtered by the segmentation-based approach. Blue means low quality and red means high quality.

4. Experimental Framework

In this section, we show the main aspects related to the experimental setup that we will use in this work. Section 4.1 details the databases used. Section 4.2 summarizes the methods used in this study with their respective parameters. Finally, Section 4.3 describes the performance measures needed to provide a faithful comparison of the obtained results.

4.1. Databases

This section describes the 12 databases used for this study, grouped into three categories: 5 SFinGe-generated (Section 4.1.1), 6 FVC databases (Section 4.1.2) and a real database captured by the authors (Section 4.1.3).

4.1.1. SFinGe databases

To evaluate the efficacy of the analyzed minutiae filtering techniques we will use the SFinGe tool (Cappelli et al., 2004; Maltoni et al., 2009) to generate five synthetic databases with different sizes. This software allows us to control the quality and other features of the generated fingerprints. In order to make the generation process reproducible, Table 1 shows the configuration parameters of the SFinGe tool that we have used to generate the databases. These parameters have been selected aiming to obtain realistic fingerprints that offer a wide range of image qualities, including very low quality fingerprints with highly corrupted areas.

Scanner parameters
Acquisition area: 0.58" x 0.77" (14.6mm x 19.6mm). Resolution: 500 dpi, Image size: 288 x 384. Background type: Optical, Background noise: Default. Crop borders: 0 x 0.
Generation parameters
Impression per finger: 25. Class distribution: Natural. Set all distributions as: "Varying quality and perturbations" Generate pores: enabled, Save ISO templates: enabled.
Output settings
Output file type: WSQ.

Table 1: Parameter specification used with SFinGe tool.

In the five databases, we have generated 25 impressions of each fingerprint. For each fingerprint, we have to differentiate between template and input impressions. In order to perform a more real setup, we carry out an enrolment process in the synthetic databases. This process selects a good quality impression as template, ensuring a minimum of 40 ground-truth minutiae. In case that all 25 impressions have less than 40 minutiae, the sample with the highest number of minutiae is taken as template. The remaining 24 samples are considered as input fingerprints. Table 2 presents the characteristics of the databases generated, showing the size of the databases and the average number of ground-truth minutiae of the template and input fingerprints.

4.1.2. FVC databases

In order to get the most realistic behavior, we have only taken those FVC databases that contain real fingerprints. Thus, we have used DB1A and DB2A from FVC2000 (Maio et al., 2002a), FVC2002 (Maio et al., 2002b) and FVC2004 (Maio et al., 2004), making a total of six real databases, whose main features are described in Table 2.

Denomination	Number of Fingerprints	Impressions per finger	Average template minutiae number	Average input minutiae number
BD1	1000	25	40.79	36.84
BD2	1000	25	40.90	36.79
BD3	1000	25	41.29	37.28
BD4	2000	25	40.84	36.81
BD5	5000	25	40.97	36.98
FVC2000_db1a	100	8	49.51	48.93
FVC2000_db2a	100	8	58.43	57.97
FVC2002_db1a	100	8	53.11	49.69
FVC2002_db2a	100	8	61.87	56.93
FVC2004_db1a	100	8	49.01	62.84
FVC2004_db2a	100	8	64.45	64.19
Captured	1530	9	44.63	45.92

Table 2: Summary description of the databases.

4.1.3. Captured database

To complete this study, the experiments have been repeated with a database of real fingerprints, which have been captured by the authors' research groups in three different cities. The fingerprints have been captured with an optical sensor (SecuGen Hamster Plus), and belong to the thumb, forefinger and middle finger of both hands of 356 people. The captures were taken within three different sessions, between two and three weeks apart, obtaining two template images and twelve test images (four per session) per fingerprint.

After removing failed captures and selecting a single random template image and three random test images per finger and per session, the final database is formed by 1530 template fingerprints and 13770 input fingerprints, whose overall statistics are shown in Table 2.

4.2. Algorithms and configuration of the parameters

Minutiae-based matching algorithms can work at different levels with the minutiae sets, comparing small groups of them (local approaches), using the whole minutiae sets (global approaches) and combining both philosophies (hybrid approaches). In this work, we will use four well-known minutiae-based matchers: Jiang (Jiang and Yau, 2000), Deng (Deng and Huo, 2005), Chen (Chen et al., 2006) and MCC (Cappelli et al., 2010). These matchers are briefly describe hereafter:

- Jiang's algorithm is a classical hybrid matching algorithm in which each minutia is represented with a feature vector that is related with its neighboring minutiae. Thus, the most similar pair of feature vectors should

correspond to the same minutia, and the remaining minutiae are globally matched, obtaining the similarity score.

- Deng's algorithm also presents a hybrid approach, but is based on the minutiae graph triangulation. Once the triangle set of each fingerprint is computed using the Delaunay triangulation, the algorithm calculates the global score associated to each triangle pair. The final score is the maximum global score.
- Chen's method is local and focuses on getting robustness despite of the fingerprint distortion. It calculates a local topology for each minutia with a fixed radius. Then, it compares local topologies of fingerprints to establish the similarity. If they are similar enough, it includes a second comparison with a higher radius, aiming to avoid image distortion problems.
- MCC uses both local and global information to perform the matching, building tridimensional data structures (called cylinders) from minutiae distances and angles. This method includes its own filtering process that is based on the convex hull idea. In our experiments we will test two versions of this algorithm depending on the cylinder's size (8 and 16), disabling its own filtering process when it is used in combination with the analyzed filtering methods to avoid a double removing stage. We will call these modified variants MCC8n and MCC16n, to distinguish them from the original algorithm with the embedded filtering method.

Algorithm	**Parameters**	**Reference**
Mindtct	output format = ANSI INCITS 378-2004, image enhancement = enabled	(Watson et al., 2010)
Convex hull	r_{min} = 12	Section 3.1
Segmentation	M_0 = 100, V_0 = 1000, T_v = 30	Section 3.2
Chen	$THR_L = 55, THR_H = 80, R = 80, RS = 100,$ $LEN_L = 5, LEN_H = 20, THETA_L = 0.25$ $THETA_H = 0.4, THRTOPO = 0.7$	(Chen et al., 2006)
Deng	$W_1 = 1, W_2 = 0.3 \cdot 180/\pi, W_3 = 0.3 \cdot 180/\pi, W_4 = 6, W_5 = 6,$ $TH_1 = 36, TH_{num} = 20, TH_{edge} = 15, TH_d = 8, TH_\theta = \pi/6,$ $TH_\phi = \pi/6, TH_{rc} = 3, TH_{SL} = 0.2, TH_{ang} = \pi/6,$	(Deng and Huo, 2005)
Jiang	$W_1 = 1, W_2 = 0.3 \cdot 180/\pi, W_3 = 0.3 \cdot 180/\pi, W_4 = 3, W_5 = 3,$ Consolidation step iterations = 5, Minutia neighborhood size = 2 $BG_1 = 8, BG_2 = \pi/6, BG_3 = \pi/6$	(Jiang and Yau, 2000)
MCC	R = 70, $N_s \in \{8, 16\}$, N_d = 6, $\sigma_s = 28/3$, $\sigma_d = 2/9 \cdot \pi$, $\mu_\Psi = 1/100$ τ_Ψ = 400, $\omega = 50, min_{VC} = 0.75, min_M = 2, min_{ME} = 0.60,$ Floating-point-based version: enabled, $\mu_P = 20, \tau_P = 2/5,$ $min_{n_p} = 4, max_{n_p} = 12, w_R = 0.5, \mu_1^\rho = 5, \tau_1^\rho = -8/5,$ $\mu_2^\rho = \pi/12, \tau_2^\rho = -30, \mu_3^\rho = \pi/12, \tau_3^\rho = -30, n_{rel} = 5, \sigma_\theta = \pi/2$	(Cappelli et al., 2010)

Table 3: Specification of the parameters.

Note that these algorithms are translation and rotation invariant. The configuration parameters of all the methods used in this study are common for all databases, and they were selected according to the recommendation of the corresponding authors (Table 3). This can be done because the study is performed comparing the algorithms with themselves. Furthermore, this parameter setting allows future comparisons for other studies, and produces a more realistic setup, avoiding the overfitting that may arise from specific parameter optimization. The value for the r_{min} parameter was selected because at least 12 minutiae are needed to claim the individuality of a fingerprint (Dass, 2010).

4.3. Performance measures

In this section we present the three different types of performance measures that we used to evaluate the minutiae filtering methods from different perspectives. In Section 4.3.1, we describe the first group of metrics, which are related to the quality of the detected minutiae in comparison with ground-truth minutiae. These metrics aim to extend those defined in Section 2.2. The second ones are devoted to measure the performance of the matching algorithms in the fingerprint verification problem (Section 4.3.2). Finally, the third group aims to measure the identification performance of the algorithms (Section 4.3.3).

4.3.1. New minutiae quality evaluation metrics

In addition to the measures explained in Section 2.2, in this work we propose a simple similarity measure not to replace the previous proposals, but to expand them providing information about the difference between ground-truth minutiae and the corresponding correctly extracted minutiae. In conjunction with GI, TPR and PPV, the proposed metrics provide more information about how similar are two sets of minutiae.

These metrics measure the difference between the location and angle estimation of the extracted minutiae and the ground truth ones. They are computed as follows. For each ground-truth minutia, the nearest extracted minutia in a ten pixels radius is selected as matched minutia (which cannot be then matched with a different ground-truth minutiae). If there are no extracted minutiae in this area, the ground-truth minutia is marked as missing. When all the ground-truth minutiae are categorized as matched or missing ones, the remaining extracted minutiae (if any) are considered to be spurious. After this process, for each paired minutiae their euclidean distance in pixels is computed. Finally, the average of the euclidean distances of all pairs is obtained to measure the location error. This

error is denoted as Mean Euclidean Distance (MED). Similarly, the absolute average difference between the angles of the matched minutiae pairs (in degrees) is computed and denoted as mean angle distance (MAD).

4.3.2. Measuring the performance of the matchers in the verification problem

In order to measure the effectiveness of the matchers, we consider the well-known False Matching Rate (FMR), False Non-Matching Rate (FNMR), and Equal-Error Rate (EER), which indicates the value where FMR and FNMR are equal. Furthermore, we use other useful indicators such as FMR100 (the lowest achievable FNMR for a FMR $\leq 1\%$) and FMR1000 (the lowest FNMR for a FMR $\leq 0.1\%$).

4.3.3. Measuring the performance of the matchers in the identification problem

The measures presented in the preceding section offer average values for 1vs1 comparisons. Thus, we use some additional values to complete the scope of our study, which use the concept of rank. Within an identification process, where the input fingerprint is compared to all template fingerprints in a database, the rank is the position of the genuine score if all the obtained scores are ordered in descending order. In other words, the rank is the minimum number of database fingerprints that have to be returned by the identification system to ensure that the correct identity is included. The accuracy measures in these case are R100 (lowest rank that allows an error lower than 1%) and R1000 (lowest rank that allows an error lower than 0.1%). The optimum value for these measures is 1, whereas the worst one is the size of the database.

Additionally, the CMC (Cumulative Match Curve) is used to show graphically the behavior of a matching algorithm. The curve shows the error associated to each rank.

5. Experimental Study

This section presents all the experiments that have been designed for this work. There is one section for each of the tested databases: SFinGe (Section 5.1), FVC (Section 5.2) and captured (Section 5.3). As we have mentioned, this experimentation has several objectives, and we have divided this section accordingly:

1. To check the accuracy of the different matchers when using the minutiae extracted by MINDTCT in comparison the accuracy obtained with a perfect minutiae extractor that obtains ground-truth minutiae (Section 5.1.1).

2. To verify the usefulness of the MINDTCT minutia quality value. The same accuracy measures from the verification and identification problems are used to compare the results when only minutiae over a certain quality threshold are used for the matching (Section 5.1.2).
3. To measure the quality of the extracted minutiae with respect to the filtered ones. The quality measures described in Sections 2.2 and 4.3.1 are computed for all the minutiae sets, comparing whether the quality of the filtered minutiae overcome that of the original ones (Section 5.1.3).
4. To test the effect of the convex hull and segmentation filters in both types of matching problems considered. We perform an analysis of the verification and identification accuracy measures (Sections 5.1.4, 5.1.5, 5.2 and 5.3).

5.1. SFinGe databases

In this section, we use the SFinGe databases, which provide the ground-truth minutiae, to study the minutiae statistics and the behavior of the minutiae extractor and all the proposed filtering schemes along with several minutiae matchers.

5.1.1. Performance evaluation with MINDTCT vs ground-truth minutiae

The first step of this experimental study consists of comparing the results obtained with the minutiae extracted by MINDTCT with those obtained using ground-truth minutiae, in order to quantify the loss of accuracy.

Following the experimental framework established, Table 4 shows the verification performance measures obtained for each matching algorithm and database. In this table, we observe that the matching algorithms maintain their respective performance ranking independently of the minutiae considered (ground-truth or MINDTCT). MCC16 is the best performing algorithm in all the databases, while Deng is by far the worst algorithm. Nevertheless, except for the Deng algorithm, the achieved error rates are very low for ground-truth minutiae, whereas they suffer great increase when MINDTCT minutiae are used. The bad behavior of Deng's algorithm could be explained because it uses additional information (the ridge count) that is not included in the ground-truth information provided by SFinGe, and therefore it has to be obtained from the data extracted by MINDTCT. However, the element that provides the most information to Deng's algorithm is still the minutiae set itself.

The results on the identification problem for the same algorithms and databases are also shown in Table 4. It can also be observed that MINDTCT introduces noise and erroneous information deteriorating the results obtained, as expected. It can be noted that although Deng is the less accurate algorithm, Chen obtains

		EER		FMR100		FMR1000		R100		R1000	
	Matcher	**GT**	**MIND.**	**GT**	**MIND.**	**GT**	**MIND.**	**GT**	**MIND.**	**GT**	**MIND.**
BD1	Chen	0.2291	5.6618	0.1750	10.4083	0.3750	19.3000	1	1000	1000	1000
	Deng	13.0702	15.6673	22.3917	23.5375	24.8125	26.7208	869	873	983	983
	Jiang	0.1047	2.3267	0.0375	3.3958	0.1292	8.4833	1	58	1	789
	MCC8	0.0137	0.5712	0.0000	0.7292	0.0042	1.1375	1	1	1	163
	MCC16	0.0091	0.5368	0.0042	0.7708	0.0042	1.1833	1	1	1	148
BD2	Chen	0.2510	5.2323	0.1833	9.5208	0.3875	17.9458	1	1000	1000	1000
	Deng	13.3764	15.3131	22.6708	23.4917	24.725	26.3875	851	866	977	981
	Jiang	0.0964	2.2146	0.0333	3.2500	0.1250	7.6042	1	33	1	648
	MCC8	0.0252	0.3988	0.0083	0.4458	0.0125	0.7792	1	1	1	59
	MCC16	0.0148	0.3792	0.0000	0.4208	0.0000	0.7333	1	1	1	30
BD3	Chen	0.1988	5.4133	0.1500	10.8167	0.3250	19.4750	1	1000	1000	1000
	Deng	13.8259	14.3113	22.9417	23.8042	25.225	26.6875	863	848	981	978
	Jiang	0.1531	2.3961	0.0750	3.6417	0.1625	8.0125	1	51	1	825
	MCC8	0.0256	0.5017	0.0042	0.8042	0.0042	1.2167	1	1	1	97
	MCC16	0.0161	0.4663	0.0042	0.7000	0.0083	1.1125	1	1	1	88
BD4	Chen	0.2399	5.4464	0.1792	9.9646	0.3812	18.6313	1	2000	2000	2000
	Deng	13.1484	15.2916	22.4583	23.5833	24.4813	26.525	1722	1739	1959	1964
	Jiang	0.1013	2.2647	0.0354	3.3792	0.1313	7.7250	1	90	1	1514
	MCC8	0.0207	0.4910	0.0042	0.5771	0.0083	0.9479	1	1	1	177
	MCC16	0.0119	0.4538	0.0021	0.6125	0.0021	0.9958	1	1	1	152
BD5	Chen	0.2122	5.4377	0.1667	11.0333	0.3308	19.3142	1	5000	5000	5000
	Deng	12.8995	13.8054	22.3058	23.5075	24.7092	26.9992	4309	4311	4912	4909
	Jiang	0.1106	2.3017	0.0533	3.5650	0.1325	7.9992	1	250	1	3923
	MCC8	0.0225	0.4860	0.0033	0.7542	0.0050	1.1925	1	2	1	430
	MCC16	0.0178	0.4558	0.0025	0.6642	0.0050	1.0967	1	2	1	362

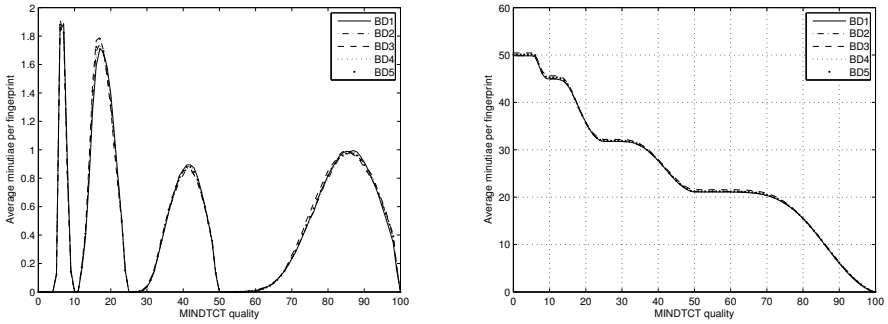
Table 4: Verification and identification results for ground-truth and MINDTCT minutiae.

the maximum possible values in most of the cases. This is due to some outliers, corresponding to genuine scores with value zero. These outliers appear because by definition the rank is a maximum value. The other matchers do not have outliers in their results, although it can be observed that MINDTCT minutiae perform considerably worse than the perfect results achieved using ground-truth minutiae.

5.1.2. MINDTCT quality value

The previous results confirm that the minutiae extraction process is a critical step in a fingerprint identification system, and thus a deeper study of the extracted minutiae must be performed. As explained in Section 2.2, MINDTCT algorithm provides a quality value for each extracted minutia. Figure 6a shows the average quality distribution per fingerprint of the five considered databases. As all databases have been generated using the same parameters with SFinGe, the average characteristics are very similar. Figure 6b shows how much the average number of minutiae per fingerprint is reduced when we filter the minutiae with a

basic quality threshold. These figures follow a very clear pattern, marking three sharp separations: 10, 30 and 50. According to Figure 6b, these thresholds leave each fingerprint with an average of 45, 32 and 21 minutiae, respectively.



(a) Distribution (b) Cumulative

Figure 6: Average quality distribution of the minutiae for each fingerprint

		EER				FMR100				FMR1000			
	Matcher	**MIND.**	**Th-10**	**Th-30**	**Th-50**	**MIND.**	**Th-10**	**Th-30**	**Th-50**	**MIND.**	**Th-10**	**Th-30**	**Th-50**
BD1	Chen	5.6618	5.7528	**5.3506**	9.4004	**10.4083**	10.5167	11.0500	24.3593	19.3000	21.1500	**19.0958**	49.2020
	Deng	15.6673	15.5481	**14.8782**	19.2270	23.5375	**23.4917**	25.1292	43.9963	**26.7208**	27.3333	29.3583	91.3285
	Jiang	2.3267	**2.3030**	2.8475	5.5472	3.3958	**3.3292**	4.2083	8.5779	8.4833	**7.3208**	8.0208	13.3169
	MCC8	**0.5712**	0.6833	1.4845	4.8255	**0.7292**	0.9917	2.7167	6.2521	**1.1375**	1.5458	3.6625	8.6728
	MCC16	**0.5368**	0.7152	1.5154	3.2427	**0.7708**	0.8792	2.4083	5.9386	**1.1833**	1.8750	3.3250	8.3382
BD2	Chen	5.2323	5.4240	**4.8810**	8.8163	**9.5208**	9.8583	11.6667	22.6625	17.9458	19.5583	**17.6333**	57.9000
	Deng	15.3131	15.2669	**15.1866**	17.1155	23.4917	**23.3208**	24.5500	49.1909	26.3875	**26.0875**	29.9583	90.9910
	Jiang	2.2146	**2.1769**	2.7535	5.0819	3.2500	**3.0375**	3.9000	7.6144	7.6042	**7.0708**	8.0208	12.5740
	MCC8	**0.3988**	0.5862	1.4593	4.7469	**0.4458**	0.5542	2.4250	6.1073	**0.7792**	1.5625	3.2208	8.3406
	MCC16	**0.3792**	0.4814	1.5318	2.9973	**0.4208**	0.5667	2.1667	5.7223	**0.7333**	1.0750	2.8875	7.8217
BD3	Chen	5.4133	5.7154	**5.1666**	9.0655	**10.8167**	14.8458	12.2917	24.0875	**19.4750**	21.8500	19.7417	54.9708
	Deng	14.3113	**14.1980**	15.9426	17.0402	**23.8042**	24.1667	25.2000	50.4087	**26.6875**	27.5667	31.2750	92.3757
	Jiang	2.3961	**2.2342**	2.5970	5.0475	**3.6417**	3.7083	4.1958	8.0364	8.0125	**7.1333**	7.6583	11.8586
	MCC8	**0.5017**	0.8578	1.7641	4.8308	**0.8042**	0.9250	2.4500	6.0547	**1.2167**	1.4042	4.0583	8.4749
	MCC16	**0.4663**	0.7144	1.8139	3.1029	0.7000	**0.4875**	2.2042	5.7415	**1.1125**	1.2958	3.7375	7.8945
BD4	Chen	5.4464	5.5880	**5.1159**	9.1090	**9.9646**	10.1875	12.4354	23.5109	18.6313	20.5292	**18.3646**	54.3595
	Deng	15.2916	15.3520	**15.1611**	17.2930	23.5833	**23.4479**	24.7979	48.5937	**26.5250**	27.0562	29.1187	90.7194
	Jiang	2.2647	**2.2230**	2.8559	5.1915	3.3792	**3.2042**	4.2292	7.9669	7.7250	**7.2792**	7.9583	12.7704
	MCC8	**0.4910**	0.6360	1.4514	4.7531	**0.5771**	0.7583	2.6021	6.1901	**0.9479**	1.2542	3.4875	8.5213
	MCC16	**0.4538**	0.5909	1.4862	3.1016	**0.6125**	0.7500	2.3146	5.8868	**0.9958**	1.2417	3.1396	8.1405
BD5	Chen	5.4377	5.6067	**5.1790**	9.2726	**11.0333**	14.6792	12.5333	24.8677	19.3142	19.1517	**18.5242**	54.5388
	Deng	**13.8054**	14.1424	13.8936	17.3177	**23.5075**	24.0117	25.7600	49.3406	26.9992	**26.6008**	31.2533	91.7535
	Jiang	2.3017	**2.2976**	2.8379	5.2062	3.5650	**3.2367**	4.3425	7.9578	7.9992	**7.2158**	8.1425	12.8853
	MCC8	**0.4860**	0.6420	1.4924	4.8992	0.7542	**0.7500**	2.6583	6.5453	**1.1925**	1.2167	3.5500	8.9515
	MCC16	**0.4558**	0.6182	1.4716	3.2772	**0.6642**	0.6875	2.4325	6.2481	**1.0967**	1.7033	3.2825	8.5172

Table 5: Verification results obtained using different quality thresholds to filter the minutiae.

		R100				R1000			
	Matcher	**MIND.**	**Th-10**	**Th-30**	**Th-50**	**MIND.**	**Th-10**	**Th-30**	**Th-50**
BD1	Chen	1000	1000	1000	**999**	**1000**	**1000**	**1000**	**1000**
	Deng	873	**869**	880	906	**983**	984	985	990
	Jiang	58	**53**	107	855	789	**709**	915	992
	MCC8	**1**	2	44	102	163	**156**	630	210
	MCC16	**1**	2	34	130	148	**112**	548	272
BD2	Chen	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**
	Deng	866	**862**	869	896	981	**980**	982	981
	Jiang	33	**25**	76	809	**648**	671	925	994
	MCC8	**1**	**1**	35	51	**59**	104	667	114
	MCC16	**1**	**1**	27	62	**30**	88	660	145
BD3	Chen	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**
	Deng	848	**838**	859	898	**978**	981	979	981
	Jiang	51	**39**	87	779	825	**786**	879	991
	MCC8	**1**	**1**	42	85	97	**85**	596	192
	MCC16	**1**	**1**	27	91	**88**	97	507	221
BD4	Chen	2000	2000	2000	**1999**	**2000**	**2000**	**2000**	**2000**
	Deng	1739	**1731**	1747	1798	**1964**	1966	1967	1971
	Jiang	90	**79**	187	1657	1514	**1377**	1832	1986
	MCC8	**1**	2	77	148	**177**	251	1296	316
	MCC16	**1**	2	58	179	**152**	208	1202	390
BD5	Chen	5000	5000	5000	**4999**	5000	5000	5000	**4999**
	Deng	4311	**4267**	4337	4487	4909	4904	**4898**	4907
	Jiang	250	**209**	465	3798	3923	**3706**	4475	4958
	MCC8	**2**	3	214	477	**430**	647	3356	1015
	MCC16	**2**	3	152	559	**362**	523	3095	1195

Table 6: Identification results obtained using different quality thresholds to filter the minutiae.

As a first approach to reduce the loss of accuracy when using MINDTCT, we have filtered the extracted minutiae using their quality value. This filtering consists of removing the minutiae whose quality is below a fixed threshold. Note that MINDTCT results shown in Table 4 correspond to a threshold with value 0. Table 5 shows the verification results obtained for all algorithms and databases when this simple filter is used with the three selected thresholds. We stress in bold-face the best result for each database and each algorithm. In general, the table shows that the variation in the error rate with respect to the quality threshold depends on the matching algorithm. On the one hand, Jiang and Deng obtain better results when the 10 threshold is used (variant Th-10). The same occurs for Chen and the Th-30 variant. However, with the other thresholds the accuracy is often lower than the one obtained considering the whole minutiae set. On the other hand, the performance loss of MCC is dramatic when the threshold is high. This is due to the embedded filtering technique of MCC, which is based on the

convex hull.

Additionally, Table 6 shows the identification results for the same algorithms and databases. Now only Jiang and Deng show a certain improvement in the results, when the threshold 10 is used. In all other cases the obtained results are worse than when the whole set of minutiae extracted by MINDTCT is used.

Therefore, we can conclude that this simple threshold-based technique is not useful to improve the accuracy, but the improvement that can be observed in some specific cases (such as Jiang with 10 threshold or Chen with 30 threshold) suggests that there may be some way to obtain the desired performance gain eliminating minutiae. It is also clear that the quality value provided by MINDTCT is not valid by itself to filter the minutiae sets, since minutiae in critical zones of the fingerprint (such as singular points) are deleted even though they are correctly detected.

5.1.3. Minutiae quality comparison: MINDTCT and filtering with convex hull and segmentation

		BD1	**BD2**	**BD3**	**BD4**	**BD5**
Ground-Truth	Mean	36.9936	36.9549	37.4404	36.9742	37.1392
	Std.	9.6787	9.4596	9.6266	9.5698	9.5539
	Max.	77	86	75	86	86
	Min.	8	9	5	8	5
MINDTCT	Mean	49.8367	49.8643	50.4599	49.8505	50.1133
	Std.	13.0665	12.8395	13.0434	12.9531	13.0357
	Max.	115	127	116	127	127
	Min.	10	15	12	10	9
CH-10	Mean	45.9267	46.0320	46.5839	45.9793	46.2658
	Std.	12.3741	12.1213	12.3295	12.2484	12.3221
	Max.	111	119	114	119	119
	Min.	10	15	8	10	8
CH-100	Mean	37.9738	37.9731	38.5253	37.9735	38.2162
	Std.	11.9769	11.7203	11.8661	11.8492	11.8935
	Max.	105	115	106	115	116
	Min.	10	14	8	10	8
Seg-10	Mean	45.5284	45.6393	46.2042	45.5838	45.8872
	Std.	12.0370	11.7531	12.0042	11.8959	11.9748
	Max.	108	114	113	114	114
	Min.	7	13	8	7	7
Seg-20	Mean	37.7164	37.5088	37.8590	37.6126	37.6798
	Std.	10.6260	10.2919	10.4139	10.4607	10.4599
	Max.	104	104	95	104	104
	Min.	5	7	7	5	5
Seg-100	Mean	33.7098	33.5650	34.0063	33.6374	33.8091
	Std.	9.8163	9.4293	9.5950	9.6249	9.6280
	Max.	104	103	90	104	104
	Min.	5	7	5	5	5

Table 7: Comparison of the number of minutiae obtained when filtering the minutiae with CH-ϕ ($\phi \in \{10, 100\}$) and Seg-ϕ ($\phi \in \{10, 20, 100\}$).

Once we have shown that the usage of the quality given by MINDTCT is not useful by itself, in this section we analyze the effect of the application of the filtering techniques described in Section 3 to the minutiae extracted by MINDTCT. In order to do so, we use the minutiae quality measures presented in Sections 2.2 and 4.3.1. This way, we are able to study whether the filtering techniques allow us to improve the quality of the minutiae.

		MSD	**MAD**	**Matches**	**Spur**	**Missing**	**GI**	**TPR**	**PPV**
BD1	MINDTCT	4.3287	18.6132	**30.5779**	19.2474	6.4157	0.1265	**0.8308**	0.6242
	CH-10	4.3283	18.4609	30.1679	15.7588	6.8256	0.2006	0.8192	0.6676
	CH-100	4.3298	18.0510	27.8580	10.1158	9.1286	0.2282	0.7523	0.7502
	Seg-10	4.3246	18.4616	30.4098	15.1186	6.5837	0.2324	0.8263	0.6781
	Seg-20	4.3139	18.0300	29.8012	7.9152	7.1869	**0.4040**	0.8098	0.8034
	Seg-100	**4.3135**	**18.0216**	27.8824	**5.8274**	9.0846	0.3586	0.7576	**0.8417**
BD2	MINDTCT	4.3150	18.2277	**30.7011**	19.1633	6.2538	0.1362	**0.8347**	0.6252
	CH-10	4.3153	18.0810	30.2894	15.7425	6.6655	0.2080	0.8230	0.6675
	CH-100	4.3178	17.6707	27.9835	9.9896	8.9652	0.2386	0.7563	0.7517
	Seg-10	4.3115	18.0807	30.5370	15.1023	6.4179	0.2403	0.8304	0.6779
	Seg-20	4.2998	17.5808	29.9147	7.5940	7.0357	**0.4197**	0.8136	0.8091
	Seg-100	**4.2982**	**17.5301**	28.0243	**5.5407**	8.9076	0.3749	0.7622	**0.8473**
BD3	MINDTCT	4.3548	18.4689	**31.0070**	19.4186	6.4334	0.1267	**0.8306**	0.6242
	CH-10	4.3547	18.3087	30.5954	15.9884	6.8449	0.1981	0.8191	0.6662
	CH-100	4.3574	17.9001	28.3180	10.2073	9.1168	0.2306	0.7542	0.7496
	Seg-10	4.3511	18.3187	30.8433	15.3609	6.5971	0.2296	0.8263	0.6764
	Seg-20	**4.3407**	17.8158	30.2044	7.6547	7.2297	**0.4106**	0.8092	0.8090
	Seg-100	4.3411	**17.7903**	28.3309	**5.6754**	9.0822	0.3657	0.7591	**0.8452**
BD4	MINDTCT	4.3218	18.4205	**30.6395**	19.2053	6.3348	0.1314	**0.8328**	0.6247
	CH-10	4.3218	18.2710	30.2287	15.7506	6.7456	0.2043	0.8211	0.6675
	CH-100	4.3238	17.8608	27.9208	10.0527	9.0469	0.2334	0.7543	0.7510
	Seg-10	4.3181	18.2711	30.4734	15.1105	6.5008	0.2363	0.8283	0.6780
	Seg-20	4.3069	17.8054	29.8580	7.7546	7.1113	**0.4118**	0.8117	0.8062
	Seg-100	**4.3059**	**17.7759**	27.9533	**5.6840**	8.9961	0.3668	0.7599	**0.8445**
BD5	MINDTCT	4.3400	18.4312	**30.7833**	19.3170	6.3560	0.1300	**0.8325**	0.6246
	CH-10	4.3398	18.2732	30.3732	15.8926	6.7659	0.2017	0.8209	0.6667
	CH-100	4.3417	17.8590	28.0841	10.1321	9.0482	0.2335	0.7550	0.7506
	Seg-10	4.3363	18.2827	30.6185	15.2687	6.5207	0.2332	0.8281	0.6769
	Seg-20	**4.3251**	17.7922	29.9839	7.6959	7.1497	**0.4128**	0.8110	0.8083
	Seg-100	**4.3251**	**17.7862**	28.1038	**5.7053**	9.0082	0.3672	0.7602	**0.8448**

Table 8: Minutiae quality comparison.

Table 7 shows the average number of minutiae found in each database and each method, while Table 8 presents the differences between ground-truth, extracted and filtered minutiae sets. Observing these tables, we can make the following observations about the weaknesses of MINDTCT:

- It is clear that extracting real minutiae is a very difficult task due to noise

and distortions of the fingerprint images. Thus, we observe that wide distance and angle differences are found between matched minutiae. This fact affects negatively to the performance of matching algorithms. Nevertheless, MINDTCT is able to detect about 31 of the 37 real minutiae in each fingerprint (on average, see Table 2).

- MINDTCT overlooks about 6 minutiae per fingerprint (on average). These missing minutiae may produce a performance degradation of the matching algorithms.
- The number of detected spurious minutiae is high. As we observed in the example presented in Figure 1, an important number of these spurious minutiae are near to the borders of the fingerprint.

This experimental study confirms our premises about MINDTCT. Among these drawbacks, the most remarkable problem of this minutiae extractor is the high number of spurious minutiae detected. However, these minutiae can be addressed by a post-processing step, as we have mentioned earlier.

Otherwise, the analyzed filtering procedures remove up to 16 of the extracted minutiae. Thus, the resulting number of filtered minutiae can be lower than the number of real minutiae, but the absolute difference is smaller than when using MINDTCT. Moreover, this fact could be due to missing minutiae rather than to the removal of real ones. For this reason, we need to investigate the results in terms of the quality of the minutiae. It is also clear that CH-10 and Seg-10 produce very similar minutiae statistics. Due to their low threshold value, they remove only minutiae that are spurious with a very high probability, and thus these filters are those removing the lowest number of minutiae (the most conservative ones). When the threshold is increased, so does the number of filtered minutiae, but this also brings along an increase in the probability of removing real minutiae.

MINDTCT has the greatest number of both spurious and matched minutiae, and the highest TPR. For all the described methods an important decrease in the number of spurious minutiae in comparison with the bare use of MINDTCT can be observed. Additionally, the number of matches is always very close to the number achieved by MINDTCT, meaning that the vast majority of pruned minutiae are spurious.

If we focus on the GI measure, Seg-20 filtering always reaches the highest values. Hence, this approach offers the best balance between spurious and missing minutiae. Otherwise, Seg-100 obtains the best PPV in all cases, meaning that it is the method that removes the most spurious minutiae, and hence most of the remaining minutiae are real, although the number of missing minutiae is slightly

higher than with other approaches. It is also the approach with the lowest MSD and MAD, meaning that the removed minutiae are those that are the furthest from their respective ground-truth minutiae.

5.1.4. Performance evaluation for the verification problem

		EER			**FMR100**			**FMR1000**		
	Matcher	**MIND.**	**CH-10**	**CH-100**	**MIND.**	**CH-10**	**CH-100**	**MIND.**	**CH-10**	**CH-100**
BD1	Chen	5.6618	**5.5307**	6.1476	**10.4083**	13.9167	11.5000	19.3000	**18.6042**	21.8708
	Deng	**15.6673**	34.2749	34.4205	**23.5375**	66.6333	68.4917	**26.7208**	78.9208	78.6625
	Jiang	**2.3267**	2.3365	2.7583	3.3958	**3.2375**	5.1542	8.4833	**7.2417**	10.1917
	MCC8n	0.5712	**0.5409**	0.7598	**0.7292**	0.7833	0.7625	1.1375	**1.1333**	1.8542
	MCC16n	0.5368	**0.5185**	0.8275	0.7708	0.7583	**0.7000**	1.1833	**1.1208**	1.6875
BD2	Chen	5.2323	**5.1250**	5.7623	**9.5208**	12.9000	10.6542	17.9458	**17.3417**	20.9042
	Deng	**15.3131**	34.8337	34.5040	**23.4917**	68.5375	66.9625	**26.3875**	79.4750	79.7750
	Jiang	2.2146	**2.1452**	2.5900	3.2500	**3.0625**	4.2458	7.6042	**7.3333**	9.6375
	MCC8n	0.3988	**0.3816**	0.5819	**0.4458**	0.4500	0.5458	0.7792	**0.7208**	1.5417
	MCC16n	**0.3792**	0.3852	0.5804	**0.4208**	0.4542	0.5000	0.7333	**0.6708**	1.4417
BD3	Chen	**5.4133**	5.4506	6.1277	**10.8167**	14.3583	11.3458	**19.4750**	21.3500	21.2417
	Deng	**14.3113**	34.6242	33.9043	**23.8042**	69.5458	69.7875	**26.6875**	80.0500	80.0792
	Jiang	2.3961	**2.3006**	3.1027	**3.6417**	3.6542	4.9833	**8.0125**	8.1208	10.3292
	MCC8n	**0.5017**	0.5527	0.6143	0.8042	**0.4875**	0.7375	1.2167	1.2792	**1.1708**
	MCC16n	**0.4663**	0.5704	0.6201	0.7000	**0.4042**	0.6458	**1.1125**	1.1875	1.6583
BD4	Chen	5.4464	**5.3276**	5.9549	**9.9646**	13.4104	11.0771	18.6313	**17.9792**	19.3813
	Deng	**15.2916**	34.6161	32.8453	**23.5833**	68.2396	68.5292	**26.5250**	79.0979	79.1438
	Jiang	2.2647	**2.2058**	2.7430	3.3792	**3.2479**	4.9438	7.7250	**7.5188**	9.8771
	MCC8n	0.4910	**0.4587**	0.6848	**0.5771**	0.6146	0.6375	0.9479	**0.9354**	1.6958
	MCC16n	0.4538	**0.4456**	0.7044	0.6125	0.6083	**0.5896**	0.9958	**0.9229**	1.5854
BD5	Chen	5.4377	**5.3687**	6.0499	**11.0333**	14.1642	11.2117	19.3142	**18.8967**	21.9500
	Deng	**13.8054**	32.8154	32.9439	**23.5075**	67.1075	69.3317	**26.9992**	79.5333	80.9400
	Jiang	2.3017	**2.2935**	2.8867	3.5650	**3.2917**	5.0258	7.9992	**7.5017**	9.8742
	MCC8n	**0.4860**	0.5909	0.6546	0.7542	**0.4342**	0.7050	**1.1925**	1.2517	1.7667
	MCC16n	**0.4558**	0.5573	0.6797	0.6642	**0.4367**	0.6458	**1.0967**	1.2742	1.6542

Table 9: Verification results obtained with the convex hull filtering.

So far, we have observed that the filter methods improve the minutiae quality. However, the goal of this operation is to perform more accuracte matchings. This section analyzes the results obtained with the different filtering approaches presented in this work in the framework of fingerprint verification. Table 9 shows the results for the two convex hull variants. In almost all cases, the CH-10 approach performs better than the CH-100 approach, because the additional quality criterion avoids the removal of real minutiae. Focusing on the CH-10 variant, it can be observed that the results with Chen and Jiang matchers are slightly improved. In the case of MCC, the results are similar to those obtained with MINDTCT. However,

the obtained results are better than those shown in Table 5, where the filtering criteria was only based on the quality assigned to the minutiae by MINDTCT. Finally, Deng shows a deterioration of the accuracy when the filters are applied.

		EER				FMR100				FMR1000			
	Matcher	**MIND.**	**Seg-10**	**Seg-20**	**Seg-100**	**MIND.**	**Seg-10**	**Seg-20**	**Seg-100**	**MIND.**	**Seg-10**	**Seg-20**	**Seg-100**
BD1	Chen	5.6618	5.2916	4.0232	**3.6916**	10.4083	13.5458	9.6458	**6.3917**	19.3000	18.1125	14.7875	**12.6000**
	Deng	15.6673	14.9290	**13.8725**	14.5227	23.5375	**23.1042**	23.6542	24.4333	**26.7208**	27.0875	27.7000	28.2500
	Jiang	2.3267	2.1747	1.7953	**1.5705**	3.3958	2.9292	2.2333	**1.9917**	8.4833	6.6250	5.0542	**4.2458**
	MCC8n	0.5712	**0.5070**	0.6413	0.5771	0.7292	0.7542	**0.5250**	0.7000	1.1375	1.1292	1.2500	**1.0667**
	MCC16n	0.5368	**0.4877**	0.5628	0.5911	0.7708	0.7083	**0.6167**	0.6667	1.1833	1.0583	**0.9708**	1.0583
BD2	Chen	5.2323	4.9994	3.8063	**3.4353**	9.5208	12.3917	8.6542	**7.3625**	17.9458	18.3375	13.0333	**12.7875**
	Deng	15.3131	**13.9528**	14.7480	15.1588	23.4917	**22.7083**	23.1458	23.2583	26.3875	**26.0458**	26.5292	27.6792
	Jiang	2.2146	1.8747	1.5268	**1.5206**	3.2500	2.7000	**1.8875**	2.0292	7.6042	6.3625	4.7250	**4.3000**
	MCC8n	0.3988	0.3555	0.3865	**0.3432**	0.4458	0.3958	**0.3250**	0.5583	0.7792	**0.5917**	0.6042	1.0208
	MCC16n	0.3792	0.3296	0.3362	**0.3223**	0.4208	0.4125	**0.3833**	0.4667	0.7333	**0.6417**	0.6917	0.8958
BD3	Chen	5.4133	5.2407	3.8875	**3.5594**	10.8167	13.4500	8.9917	**7.8000**	19.4750	20.0292	13.7125	**13.2417**
	Deng	14.3113	13.6681	**13.3721**	14.1480	23.8042	23.2917	**23.1750**	25.4250	26.6875	**26.6708**	27.6083	30.8375
	Jiang	2.3961	2.0966	1.7268	**1.5443**	3.6417	3.2125	2.2042	**2.1208**	8.0125	6.4000	4.7875	**4.4125**
	MCC8n	0.5017	0.5742	**0.4097**	0.4903	0.8042	**0.4333**	0.4833	0.5292	1.2167	1.0958	0.8417	**0.8125**
	MCC16n	0.4663	**0.3902**	0.4287	0.5294	0.7000	**0.3917**	0.4667	0.4750	1.1125	0.9625	0.7708	**0.7333**
BD4	Chen	5.4464	5.1454	**3.9135**	3.5632	9.9646	13.0562	9.1500	**7.8292**	18.6313	17.4312	13.9104	**12.0250**
	Deng	15.2916	**14.0606**	14.9685	15.3625	23.5833	23.5854	**23.3042**	23.4729	26.5250	**26.2229**	26.7021	27.7479
	Jiang	2.2647	1.9363	1.7517	**1.5556**	3.3792	2.8292	**2.0646**	2.1854	7.7250	6.5167	4.9000	**3.8333**
	MCC8n	0.4910	**0.4229**	0.4674	0.4445	0.5771	0.5896	**0.4792**	0.7563	0.9479	0.8792	**0.8104**	1.2667
	MCC16n	0.4538	**0.4082**	0.4164	0.4291	0.6125	0.5604	**0.5396**	0.6750	0.9958	**0.8563**	0.9000	1.1146
BD5	Chen	5.4377	5.1490	3.9190	**3.5600**	11.0333	13.6717	9.1092	**7.8617**	19.3142	18.1183	13.8075	**13.3158**
	Deng	**13.8054**	14.1212	13.9448	14.7570	**23.5075**	23.8667	23.6258	25.0317	26.9992	**26.3950**	29.2192	30.2117
	Jiang	2.3017	1.9899	1.6747	**1.5863**	3.5650	2.8283	2.2275	**2.1225**	7.9992	6.4783	4.6358	**4.5058**
	MCC8n	0.4860	0.5046	0.4829	**0.4337**	0.7542	**0.4300**	0.4717	0.7275	1.1925	**0.7683**	0.8142	1.2142
	MCC16n	0.4558	0.5384	0.4752	**0.4240**	0.6642	**0.3950**	0.4600	0.6542	1.0967	1.1358	**0.7917**	1.0850

Table 10: Verification results obtained with the segmentation-based filtering.

The matching results obtained with the minutiae sets filtered using the segmentation method are shown in Table 10, where we observe that the segmentation filter has reduced the error rates achieved by all algorithms in almost all the cases. These results are closer to the obtained with ground-truth minutiae for all the considered thresholds. The results with Deng and MCC have also been improved by the applied filter. Moreover, in general the most accurate variant is Seg-100, which is the one that reduces the most the number of minutiae and has the highest PPV. This highlights this measure as a good one to determine the verification accuracy.

Besides the accuracy improvements, the reduction of spurious minutiae reduces the computational complexity of the matching process. Figure 7 and Table 11 show the average runtime[1] of the matching algorithms. In the figure, the filters are ordered in decreasing order of the number of minutiae. It is clearly

[1]These results have been obtained with an Intel(R) Xeon(R) E5-2620 CPU at 2.00 GHz.

shown that the matching time is proportional to the average number of minutiae in each database. Thus, Seg-100 approach is both the most precise and the one that produces the fastest matchings, and thus is highlighted as the optimal among the tested filters. Otherwise, the use of non-filtered MINDTCT minutiae leads to the slowest execution times.

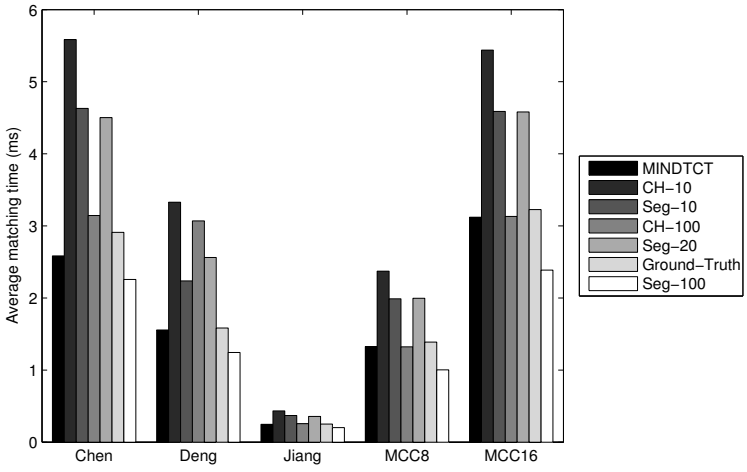


Figure 7: Average matching time in BD1 for each filtering scheme.

	Chen	**Deng**	**Jiang**	**MCC8**	**MCC16**
Ground-truth	2.5841	1.5568	0.2479	1.3269	3.1207
MINDTCT	5.5846	3.3289	0.4329	2.3722	5.4394
CH-10	4.6302	2.2378	0.3688	1.9884	4.5880
CH-100	3.1440	3.0702	0.2570	1.3232	3.1317
Seg-10	4.5026	2.5617	0.3575	1.9956	4.5807
Seg-20	2.9099	1.5836	0.2501	1.3879	3.2255
Seg-100	2.2575	1.2431	0.2021	1.0027	2.3862

Table 11: Average matching time (in milliseconds) in BD1 for each filtering scheme.

5.1.5. Performance evaluation for the identification problem

In this section we analyze the obtained results in terms of the identification performance measures, and more specifically using R100 and R1000 as defined in Section 4.3.3.

Table 12 shows the results for the convex hull-based filter. As in the previous section, when we compare these results with those obtained with MINDTCT, it

		R100			R1000		
	Matcher	**MIND.**	**CH-10**	**CH-100**	**MIND.**	**CH-10**	**CH-100**
BD1	Chen	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**
	Deng	**873**	973	977	**983**	997	998
	Jiang	58	**50**	75	789	**706**	765
	MCC8	**1**	**1**	2	163	**98**	144
	MCC16	**1**	**1**	2	148	**88**	123
BD2	Chen	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**
	Deng	**866**	973	974	**981**	997	997
	Jiang	33	**28**	46	648	**646**	787
	MCC8	**1**	**1**	**1**	59	**28**	79
	MCC16	**1**	**1**	**1**	30	**34**	81
BD3	Chen	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**
	Deng	**848**	972	970	**978**	995	997
	Jiang	51	**41**	76	825	**785**	820
	MCC8	**1**	**1**	**1**	97	**45**	104
	MCC16	**1**	**1**	**1**	88	**43**	108
BD4	Chen	**2000**	**2000**	**2000**	**2000**	**2000**	**2000**
	Deng	**1739**	1945	1950	**1964**	1994	1995
	Jiang	90	**77**	113	1514	**1411**	1535
	MCC8	**1**	**1**	2	177	**95**	228
	MCC16	**1**	**1**	2	152	**95**	223
BD5	Chen	**5000**	**5000**	**5000**	**5000**	**5000**	**5000**
	Deng	**4311**	4859	4867	**4909**	4981	4986
	Jiang	250	**217**	353	3923	3658	3954
	MCC8	2	**1**	3	430	**232**	637
	MCC16	2	**1**	3	362	**241**	618

Table 12: Identification results obtained with the convex hull-based filter.

can be shown that the CH-10 filter improves the results, mainly with MCC. However, the CH-100 variant removes too many minutiae and causes a certain loss of accuracy. The Deng algorithm performs better when no minutiae are removed, and Chen shows a high sensibility to outliers, as in the previous sections.

Finally, Table 13 contains the results of the segmentation-based filter. Now the improvement in the obtained results is very clear, meaning that the difference between genuine and impostor scores has been increased as a result of the removal of spurious minutiae.

Therefore, it can be concluded that the filtering that that removes the greatest number of minutiae (Seg-100) is also the one that achieves the best results in most of the cases. Consequently, the identification time obtained with this approach is also the fastest among all tested schemes, as it can be observed in Figure 8 and Table 14, which show the average identification time in BD1. This time corresponds to the initial processing of the input fingerprint plus 1000 matchings. Again, the figure orders the filter in decreasing order of the number of minutiae. Hence, this

		R100				R1000			
	Matcher	**MIND.**	**Seg-10**	**Seg-20**	**Seg-100**	**MIND.**	**Seg-10**	**Seg-20**	**Seg-100**
BD1	Chen	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**
	Deng	873	865	**849**	863	983	977	**975**	981
	Jiang	58	43	21	**20**	789	703	727	**63**
	MCC8	**1**	**1**	**1**	**1**	163	**72**	77	77
	MCC16	**1**	**1**	**1**	**1**	148	**68**	82	90
BD2	Chen	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**
	Deng	866	857	**852**	863	**981**	982	982	983
	Jiang	33	23	9	**7**	648	655	513	**446**
	MCC8	**1**	**1**	**1**	**1**	59	**20**	24	23
	MCC16	**1**	**1**	**1**	**1**	30	**20**	28	24
BD3	Chen	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**
	Deng	848	**836**	847	866	978	984	**977**	980
	Jiang	51	33	16	**12**	825	772	**724**	766
	MCC8	**1**	**1**	**1**	**1**	97	**34**	38	43
	MCC16	**1**	**1**	**1**	**1**	88	**31**	35	68
BD4	Chen	**2000**	**2000**	**2000**	**2000**	**2000**	**2000**	**2000**	**2000**
	Deng	1739	1725	**1707**	1721	1964	1959	**1956**	1963
	Jiang	90	63	28	**23**	1514	1363	1359	**1209**
	MCC8	**1**	**1**	**1**	**1**	177	**79**	91	106
	MCC16	**1**	**1**	**1**	**1**	152	**78**	93	99
BD5	Chen	**5000**	**5000**	**5000**	**5000**	**5000**	**5000**	**5000**	**5000**
	Deng	4311	**4249**	4250	4316	4909	4901	**4886**	4904
	Jiang	250	164	70	**61**	3923	3684	3427	**3360**
	MCC8	2	**1**	**1**	**1**	430	**212**	228	280
	MCC16	2	**1**	**1**	**1**	362	**200**	252	303

Table 13: Identification results obtained with the segmentation-based filter.

filtering method produces both faster and more accurate results, outperforming the usage of MINDTCT without post-processing, and even improving the results of the MCC internal filtering mechanism.

	Chen	**Deng**	**Jiang**	**MCC8**	**MCC16**
Ground-truth	2.5884	1.5573	0.2501	1.3450	3.1787
MINDTCT	5.5944	3.3344	0.4392	2.4053	5.5241
CH-10	4.6399	2.2563	0.3740	2.0163	4.6783
CH-100	3.1478	3.0707	0.2581	1.3408	3.1954
Seg-10	4.5112	2.5840	0.3625	2.0192	4.6715
Seg-20	2.9086	1.5953	0.2508	1.4055	3.2957
Seg-100	2.2631	1.2545	0.1973	1.0210	2.4449

Table 14: Average identification time (in seconds) in BD1 for each filtering scheme.

Finally, we conclude the study on the identification framework showing the CMC curves for the largest database considered (BD5) for all algorithms and all filtering schemes, on Figure 9. For the Jiang and Chen algorithms, Seg-100 filter dominates all solutions (except the ground-truth minutiae), closely followed by

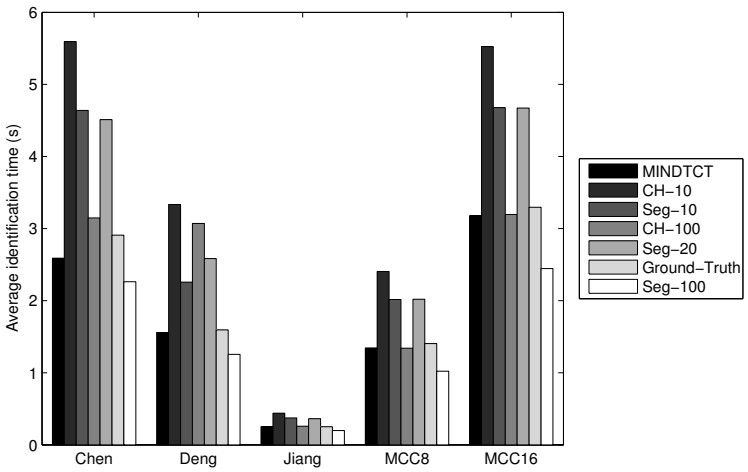
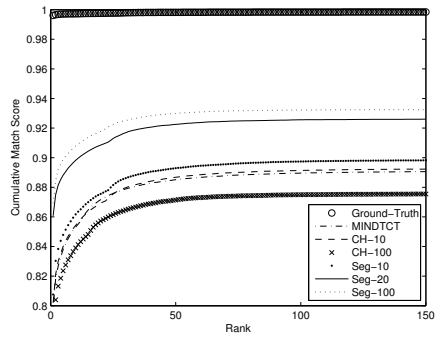


Figure 8: Average identification times in BD1 (1000 matchings per identification) for each filtering scheme.

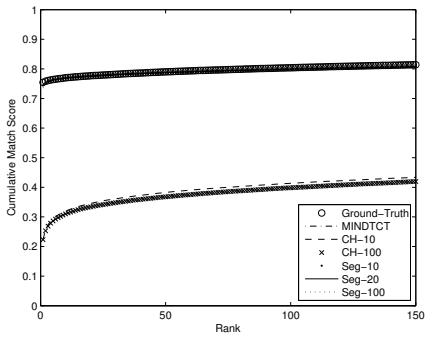
Seg-20. The usage of MINDTCT alone obtains one of the worst CMC curves, showing the importance of a good pre-processing step. For Deng, both convex hull variants have very bad results, while all the other approaches present very similar CMC curves. In the case of MCC, all variants have very high and similar scores because of its higher robustness, but again the segmentation filter provides the best results.

5.2. FVC databases

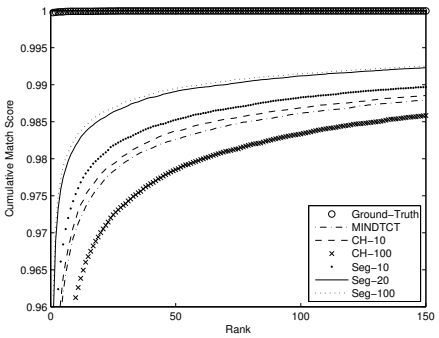
Some of the experiments have been repeated with the well-known FVC databases, to allow reproducible results in a benchmark manner and to avoid the bias in the conclusions due to the usage of a unique source of fingerprints. In order to verify the behavior of the segmentation and convex hull approaches, we have repeated the experiments with the best configuration for each one of them (CH-10 and Seg-100), as well as for the initial minutiae set extracted by MINDTCT. The minutiae statistics of the obtained databases are shown in Table 15. The results in the table show that between 20% and 40% of the extracted minutiae are deleted when using Seg-100, which is more than when using the SFinGe databases, due to the lower quality of the FVC fingerprints. This deletion is reflected in an important reduction of the matching and identification times, independently of the accuracy results achieved.



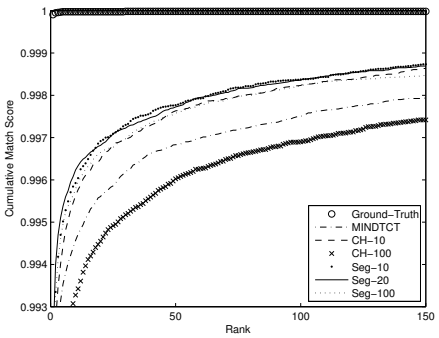
(a) Chen



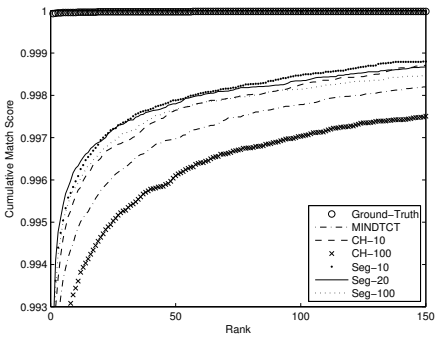
(b) Deng



(c) Jiang



(d) MCC8



(e) MCC16

Figure 9: CMC curves for BD5 database.

		FVC2000		FVC2002		FVC2004	
		DB1A	**DB2A**	**DB1A**	**DB2A**	**DB1A**	**DB2A**
MINDTCT	**Mean**	49.0063	58.0313	50.1188	57.5438	61.1138	64.2238
	Std.	11.9596	17.5767	12.9701	14.9358	18.4753	17.1335
	Max.	90	132	94	148	156	135
	Min.	18	24	11	12	20	21
CH-10	**Mean**	43.6150	53.1538	43.4525	54.0713	52.7700	59.0600
	Std.	11.4751	16.7629	11.9027	14.4751	17.4217	16.2433
	Max.	84	128	85	139	141	128
	Min.	18	21	11	12	19	21
Seg100	**Mean**	38.7838	45.4088	35.9675	45.2950	42.4500	36.9225
	Std.	10.6328	15.5173	10.5342	14.0905	14.5261	13.4584
	Max.	76	127	73	97	118	114
	Min.	12	12	7	8	10	7

Table 15: Comparison of the number of minutiae obtained when filtering the minutiae of the FVC databases with CH-10 and Seg-100.

		EER			FMR100			**FMR1000**		
		MIND.	**CH-10**	**Seg-100**	**MIND.**	**CH-10**	**Seg-100**	**MIND.**	**CH-10**	**Seg-100**
FVC2000_db1a	Chen	34.8405	33.0981	**11.9452**	69.1429	65.8571	**28.7143**	81.0000	79.7143	**37.0000**
	Deng	7.1284	6.8391	**4.8730**	11.7143	12.5714	**11.2857**	**22.0000**	23.1429	25.5714
	Jiang	13.1436	11.3016	**8.5765**	21.0000	19.5714	**15.5714**	31.8571	29.2857	**24.2857**
	MCC8n	5.3196	6.7807	**3.5440**	8.5714	15.2857	**7.8571**	13.5714	25.7143	**9.8571**
	MCC16n	7.8759	6.8413	**3.2922**	14.2857	13.8571	**8.1429**	15.0000	22.4286	**9.8571**
FVC2000_db2a	Chen	33.0960	33.1760	**11.4704**	66.1429	66.1429	**22.5714**	66.1429	77.0000	**33.2857**
	Deng	7.9856	8.3716	**5.1631**	20.4286	17.8571	**14.0000**	29.0000	28.4286	**23.0000**
	Jiang	16.2482	15.1984	**9.7763**	28.0000	26.4286	**18.5714**	39.2857	37.0000	**30.4286**
	MCC8n	5.4699	8.4892	**2.7143**	10.0000	18.8571	**4.5714**	18.5714	34.1429	**11.2857**
	MCC16n	4.2669	7.9257	**2.5765**	15.7143	18.7143	**6.7143**	19.2857	34.2857	**10.2857**
FVC2002_db1a	Chen	31.8795	30.7150	**6.7374**	63.5714	61.1429	**13.1429**	72.8571	71.7143	**20.4286**
	Deng	16.9921	15.7532	**11.9235**	22.5714	22.0000	**16.8571**	26.2857	**25.4286**	26.1429
	Jiang	16.6400	15.4062	**4.9156**	25.8571	23.7143	**9.8571**	34.4286	31.7143	**17.1429**
	MCC8n	12.4436	16.2294	**1.3903**	20.0000	22.7143	**1.5714**	27.8571	26.0000	**2.5714**
	MCC16n	12.0489	15.3434	**0.8110**	20.0000	22.8571	**1.2857**	24.2857	25.1429	**2.7143**
FVC2002_db2a	Chen	24.2237	23.2266	**3.9726**	56.7143	62.5714	**9.1429**	77.2857	77.0000	**19.2857**
	Deng	12.3254	14.2994	**8.7590**	17.2857	18.0000	**14.2857**	**20.7143**	21.7143	21.2857
	Jiang	13.5058	13.1032	**4.2648**	20.5714	20.5714	**8.1429**	28.1429	28.1429	**14.4286**
	MCC8n	14.3045	11.3997	**0.6962**	24.2857	21.4286	**0.7143**	25.0000	23.5714	**1.7143**
	MCC16n	13.6842	10.6883	**0.6111**	23.5714	21.2857	**0.4286**	27.1429	23.5714	**1.4286**
FVC2004_db1a	Chen	42.0209	41.0462	**21.6421**	83.5714	81.2857	**42.5714**	92.2857	92.2857	**55.0000**
	Deng	19.3160	19.4127	**13.7994**	40.7143	38.7143	**37.4286**	54.7143	**52.1429**	61.7143
	Jiang	25.2958	23.3398	**14.0411**	52.0000	49.1429	**31.4286**	67.4286	61.0000	**48.0000**
	MCC8n	22.1053	17.3088	**7.1443**	45.0000	38.0000	**13.1429**	55.7143	45.8571	**23.7143**
	MCC16n	23.0075	17.8074	**7.0390**	46.4286	38.0000	**13.0000**	62.8571	47.7143	**23.4286**
FVC2004_db2a	Chen	43.4942	43.1212	**18.1075**	91.5714	91.4286	**35.7143**	97.8571	97.0000	**51.8571**
	Deng	19.1320	19.5404	**15.3449**	36.4286	37.7143	**26.8571**	51.8571	49.5714	**49.2857**
	Jiang	23.7020	22.8911	**12.4437**	53.1429	50.0000	**25.0000**	67.0000	65.1429	**38.4286**
	MCC8n	22.8008	21.5678	**8.4185**	46.4286	43.4286	**12.7143**	49.2857	52.7143	**18.7143**
	MCC16n	23.5902	19.3153	**8.3629**	45.7143	42.1429	**12.0000**	53.5714	53.0000	**18.5714**

Table 16: Verification performance values for FVC databases.

Table 16 shows the verification results obtained with these databases. The first fact that should be noted is that the FVC complexity increases over the years, and hence the lowest EER rates are obtained for FVC2000, and the highest ones are seen for FVC2004.

		R100			**R1000**		
		MIND.	**CH-10**	**Seg-100**	**MIND.**	**CH-10**	**Seg-100**
FVC2000_db1a	Chen	**100**	**100**	**100**	**100**	**100**	**100**
	Deng	73	67	**34**	97	97	**85**
	Jiang	89	86	**83**	**99**	100	100
	MCC8n	62	65	**32**	99	**96**	97
	MCC16n	67	80	**30**	97	**94**	99
FVC2000_db2a	Chen	**100**	**100**	**100**	**100**	**100**	**100**
	Deng	63	69	**50**	**92**	96	98
	Jiang	96	91	**82**	100	99	**94**
	MCC8n	68	64	**17**	94	91	**47**
	MCC16n	65	67	**23**	98	95	**39**
FVC2002_db1a	Chen	**100**	**100**	**100**	**100**	**100**	**100**
	Deng	88	95	**84**	100	100	**99**
	Jiang	94	94	**53**	100	100	**87**
	MCC8n	88	80	**2**	100	100	**99**
	MCC16n	85	83	**2**	98	99	**94**
FVC2002_db2a	Chen	**100**	**100**	**100**	**100**	**100**	**100**
	Deng	83	80	**78**	99	99	**95**
	Jiang	90	88	**12**	98	98	**84**
	MCC8n	79	85	**1**	96	98	**13**
	MCC16n	82	81	**1**	98	97	**12**
FVC2004_db1a	Chen	**100**	**100**	**100**	**100**	**100**	**100**
	Deng	92	94	**79**	100	**99**	**99**
	Jiang	94	93	**89**	**99**	**99**	100
	MCC8n	90	83	**54**	**97**	100	99
	MCC16n	93	88	**55**	98	**96**	100
FVC2004_db2a	Chen	**100**	**100**	**100**	**100**	**100**	**100**
	Deng	95	94	**91**	**99**	100	100
	Jiang	96	94	**81**	**100**	**100**	**100**
	MCC8n	93	87	**74**	**99**	**99**	100
	MCC16n	95	91	**68**	100	100	**99**

Table 17: Identification performance values for FVC databases.

With the FVC databases, the Seg-100 variant greatly improves the results in all cases, except sometimes for the FMR1000 value with the Deng algorithm. Otherwise, the results of the convex hull approach are more similar to those obtained without filtering. The table also shows that the improvement obtained with Seg-100 depends on both the matcher and the database, although it is always remarkable. Moreover, the improvement is higher for the most difficult database, where a huge difference can be observed between the ERR in the MINDTCT column and that of the Seg-100. This result, along with the large number of removed

minutiae, highlight the Seg-100 approach as a very good tool to improve both the matching accuracy and its runtime, especially for low-quality databases such as those of FVC2004.

The identification accuracy results are shown in Table 17. Again, the best results are those obtained by the Seg-100 approach.

5.3. Captured database

The same experiments carried out for the FVC databases in the previous section have been executed in this case. Table 18 presents the minutiae statistics of this database. In this case, the number of minutiae removed by the filters is not as high as for the FVC databases, but still the Seg-100 variant is the one that removes the most of them.

	MINDTCT	**CH-10**	**Seg-100**
Mean	44.7602	42.1176	37.4990
Std.	11.5560	11.0870	10.8468
Max.	137	127	133
Min.	14	14	9

Table 18: Comparison of the number of minutiae obtained when filtering the minutiae of the captured database with CH-10 and Seg-100.

	EER			FMR100			**FMR1000**		
	MIND.	**CH-10**	**Seg-100**	**MIND.**	**CH-10**	**Seg-100**	**MIND.**	**CH-10**	**Seg-100**
Chen	3.9693	4.0764	**3.6698**	7.2404	7.5018	**6.8918**	15.6790	15.6935	**13.2534**
Deng	**1.1954**	1.2611	1.5208	**1.5977**	1.8954	2.5055	**5.4248**	5.5846	6.4488
Jiang	2.5860	2.5433	**2.3769**	4.1685	4.3500	**3.9797**	11.3653	10.0436	**9.2302**
MCC8	0.8691	**0.5401**	0.6417	1.4016	0.8351	**0.5882**	2.4474	**1.5686**	2.0044
MCC16	0.8393	**0.5045**	0.5605	0.7480	0.8424	**0.6245**	2.5490	1.6122	**1.1474**

Table 19: Verification performance values for the captured database.

Table 19 shows the verification results obtained with the captured database. It can be seen in the table that Seg-100 still improves the results, although the difference is not as high as for other databases. In some cases the CH-10 filter performs better, and for the Deng algorithm both filters produce worse results than when the non-filtered minutiae set is used. However, this matcher obtains more precise results for this database, especially if they are compared to those obtained with the SFinGe databases. This behavior reveals the lack of robustness of this algorithm, whose accuracy tends to be affected by the particular fingerprint

features, although for this database its results are close to those obtained by the best matcher (MCC).

The particular error values are similar to those presented in Section 5.1, except for Deng. This highlights the SFinGe databases as being reasonably realistic, since the behavior of the algorithms is similar as with a database of real fingerprints.

The identification accuracy results are shown in Table 20. Again, the Deng algorithm is the only one for which Seg-100 does not produce the best results, whereas Jiang and MCC significantly improve their accuracy. Concretely, the R rates for MCC are reduced by about 50%.

	R100			**R1000**		
	MIND.	**CH-10**	**Seg-100**	**MIND.**	**CH-10**	**Seg-100**
Chen	**1530**	**1530**	**1530**	**1530**	**1530**	**1530**
Deng	**6**	**6**	8	**454**	503	768
Jiang	164	169	**148**	1391	1363	**1352**
MCC8	4	2	**1**	298	165	**162**
MCC16	3	**1**	**1**	237	174	**137**

Table 20: Identification performance values for the captured database.

6. Conclusions and Future Lines

In this contribution, we have studied two filters to improve the minutiae extraction of MINDTCT: a convex hull-based filter and a segmentation-based filter. These filters have been tested with several configurations and databases. We have performed an analysis of the number of spurious and missing minutiae that arise when using MINDTCT, and we have shown that the studied filters can reduce the number of spurious minutiae without compromising the number of matched ones.

In our experiments, we have analyzed the influence of these spurious minutiae in several state-of-the-art minutiae-based matchers. The compared schemes allow one to remove spurious minutiae providing more accurate results even for robust matching algorithms such as MCC. The segmentation based filter is especially powerful, and the variant that removes the most minutiae is also the one that provides the best accuracy. Therefore, in addition to the accuracy improvements, the resulting reduction of number of minutiae leads to a faster matching process. These positive results are even better when the fingerprint database is of low quality, because the segmentation removes a higher number of spurious

minutiae, leading both to better accuracy and runtime, as it can be seen with the FVC databases. This fact shows the importance of an appropriate fingerprint post-processing both for the accuracy and the efficiency of the matching algorithms.

The results of this study have also shown that the PPV is a good minutiae quality measure, and the filters with high PPV tend to produce better matching accuracy. Finally, it has been observed that the SFinGe databases have a reasonably realistic behavior, as the results obtained with it are similar to those obtained with a database of real fingerprints, captured by the authors' research groups in controlled conditions.

As future work, we aim to consider new strategies to remove harmful minutiae and speed up the matching process, as well as to develop a novel matching algorithm that includes the minutiae filtering process.

Acknowledgments

This work was supported by the research projects CAB (CDTI), TIN2011-28488 and TIN2009-14575. D. Peralta holds an FPU scholarship from the Spanish Ministry of Education and Science (FPU12/04902).

References

Ahmad, T., Hu, J., Wang, S., 2011. Pair-polar coordinate-based cancelable fingerprint templates. Pattern Recognition 44, 2555–2564.

Bazen, A.M., Gerez, S.H., 2001. Segmentation of fingerprint images, in: ProRISC 2001 Workshop on Circuits, Systems and Signal Processing, pp. 276–280.

Cappelli, R., Ferrara, M., Maltoni, D., 2010. Minutia cylinder-code: A new representation and matching technique for fingerprint recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 32, 2128–2141.

Cappelli, R., Maio, D., Maltoni, D., 2004. Sfinge: an approach to synthetic fingerprint generation, in: Eighth International Conference on Control, Automation, Robotics and Vision (ICARCV2004), Kunming, China. pp. 147–154.

Chen, X., Tian, J., Yang, X., 2006. A new algorithm for distorted fingerprints matching based on normalized fuzzy similarity measure. IEEE Transactions on Image Processing 15, 767–776.

Chikkerur, S., Govindaraju, V., Pankanti, S., Bolle, R.M., Ratha, N.K., 2005. Novel approaches for minutiae verification in fingerprint images, in: WACV/MOTION, pp. 111–116.

Dass, S., 2010. Assessing fingerprint individuality in presence of noisy minutiae. IEEE Transactions on Information Forensics and Security 5, 62–70.

Deng, H., Huo, Q., 2005. Minutiae matching based fingerprint verification using Delaunay triangulation and aligned-edge-guided triangle matching, in: Lecture Notes in Computer Science, pp. 270–278.

Feng, J., Jain, A.K., 2011. Fingerprint reconstruction: From minutiae to phase. IEEE Transactions on Pattern Analysis and Machine Intelligence 33, 209–223.

Gao, X., Chen, X., Cao, J., Deng, Z., Liu, C., Feng, J., 2010. A novel method of fingerprint minutiae extraction based on gabor phase, in: Proc. of the International Conference on Image Processing, ICIP, pp. 3077–3080.

Graham, R., 1972. An efficient algorithm for determining the convex hull of a finite planar set. Information Processing Letters 1, 132–133.

Hong, L., Wan, Y., Jain, A., 1998. Fingerprint image enhancement: algorithm and performance evaluation. IEEE Transactions on Pattern Analysis and Machine Intelligence 20, 777–789.

Hung, D., 1993. Enhancement and feature purification of fingerprint images. Pattern Recognition 26, 1661–1671.

Jain, A., Hong, L., Bolle, R., 1997. On-line fingerprint verification. IEEE Transactions on Pattern Analysis and Machine Intelligence 19, 302–314.

Jiang, X., Yau, W., 2000. Fingerprint minutiae matching based on the local and global structures, in: Proc. of the 15th International Conference on Pattern Recognition, IEEE. pp. 1038–1041.

Jiang, X., Yau, W.., Ser, W., 2001. Detecting the fingerprint minutiae by adaptive tracing the gray-level ridge. Pattern Recognition 34, 999–1013.

Kumar, R., Deva Vikram, B., 2010. Fingerprint matching using multi-dimensional ann. Engineering Applications of Artificial Intelligence 23, 222–228.

Lee, C., Kim, J., 2010. Cancelable fingerprint templates using minutiae-based bit-strings. Journal of Network and Computer Applications 33, 236–246.

Maio, D., Maltoni, D., 1997. Direct gray-scale minutiae detection in fingerprints. IEEE Transactions on Pattern Analysis and Machine Intelligence 19, 27–40.

Maio, D., Maltoni, D., 1998. Neural network based minutiae filtering in fingerprints, in: Proc. of Fourteenth International Conference on Pattern Recognition, pp. 1654–1658.

Maio, D., Maltoni, D., Cappelli, R., Wayman, J.L., Jain, A.K., 2002a. FVC2000: Fingerprint verification competition. IEEE Transactions on Pattern Analysis and Machine Intelligence 24, 402–412.

Maio, D., Maltoni, D., Cappelli, R., Wayman, J.L., Jain, A.K., 2002b. FVC2002: Second fingerprint verification competition, in: Proceedings - International Conference on Pattern Recognition, pp. 811–814.

Maio, D., Maltoni, D., Cappelli, R., Wayman, J.L., Jain, A.K., 2004. FVC2004: Third fingerprint verification competition, in: Biometric Authentication, pp. 1–7.

Maltoni, D., Maio, D., Jain, A., Prabhakar, S., 2009. Handbook of fingerprint recognition. Second ed., Springer-Verlag, New York.

Peralta, D., Triguero, I., Sanchez-Reillo, R., Herrera, F., Benitez, J.M., 2014. Fast fingerprint identification for large databases. Pattern Recognition 47, 588–602.

Prabhakar, S., Jain, A., Wang, J., Pankanti, S., Bolle, R., 2000. Minutia verification and classification for fingerprint matching, in: Proc of the 15th International Conference on Pattern Recognition, pp. 25–29.

Ratha, N., Bolle, R., 2004. Automatic Fingerprint Recognition Systems. Springer, New York.

Ratha, N.K., Chen, S., Jain, A.K., 1995. Adaptive flow orientation-based feature extraction in fingerprint images. Pattern Recognition 28, 1657–1672.

Santhanam, T., Sumathi, C., Easwarakumar, K., 2007. Fingerprint minutiae filtering using artmap. Neural Computing and Applications 16, 49–55.

Watson, C.I., Garris, M.D., Tabassi, E., Wilson, C.L., Mccabe, R.M., Janet, S., Ko, K., 2010. User's Guide to NIST Biometric Image Software (NBIS). Technical Report. NIST.

Wen, C., Guo, T., 2009. An efficient algorithm for fingerprint matching based on convex hulls, in: Proc. of the 2009 International Conference on Computational Intelligence and Natural Computing, IEEE Computer Society. pp. 66–69.

Xiao, Q., Raafat, H., 1991. Combining statistical and structural information for fingerprint image processing classification and identification, in: Plamondon, R., Cheng, H. (Eds.), Pattern Recognition: Architectures, Algorithms and Applications, pp. 335–354.

Zhao, F., Tang, X., 2002. Duality-based post-processing for fingerprint minutiae extraction, in: Proc. of International Conference on Information Security, Shanghai, China, pp. 36–42.

Zhao, F., Tang, X., 2007. Preprocessing and postprocessing for skeleton-based fingerprint minutiae extraction. Pattern Recognition 40, 1270–1281.

3 Fast Fingerprint Identification for Large Databases

- D. Peralta, I. Triguero, R. Sanchez-Reillo, F. Herrera, J.M. Benítez. Fast Fingerprint Identification for Large Databases. Pattern Recognition 47:2 (2014) 588–602. doi: 10.1016/j.patcog.2013.08.002
  - Status: **Published**.
  - Impact Factor (JCR 2014): 3.096
  - Subject Category: Computer Science, Artificial Intelligence. Ranking 15 / 123 (**Q1**).
  - Subject Category: Engineering, Electrical & Electronic. Ranking 20 / 249 (**Q1**).

Fast Fingerprint Identification for Large Databases

D. Peralta[a,*], I. Triguero[a], R. Sanchez-Reillo[b], F. Herrera[a], J.M. Benitez[a]

[a]*Department of Computer Science and Artificial Intelligence of the University of Granada, CITIC-UGR, Granada, Spain, 18071*
[b]*University Group for Identification Technologies (GUTI) at the Electronics Technology Dpt., Carlos III University of Madrid, Avda. Universidad, 30, 28911 Leganes (Madrid), Spain*

Abstract

Fingerprint matching has emerged as an effective tool for human recognition due to the uniqueness, universality and invariability of fingerprints. Many different approaches have been proposed in the literature to determine faithfully if two fingerprint images belong to the same person. Among them, minutiae-based matchers highlight as the most relevant techniques because of their discriminative capabilities, providing precise results. However, performing a fingerprint identification over a large database can be an inefficient task due to the lack of scalability and high computing times of fingerprint matching algorithms. In this paper, we propose a distributed framework for fingerprint matching to tackle large databases in a reasonable time. It provides a general scheme for any kind of matcher, so that its precision is preserved and its time of response can be reduced. To test the proposed system, we conduct an extensive study that involves both synthetic and captured fingerprint databases, which have different characteristics, analyzing the performance of three well-known minutiae-based matchers within the designed framework. With the available hardware resources, our distributed model is able to address up to 400 000 fingerprints in approximately half a second. Additional details are provided at `http://sci2s.ugr.es/ParallelMatching`.

Keywords:

Distributed computing, large databases, minutiae matching, parallel computing, real-time fingerprint identification

*Corresponding author. Tel.: +34 958244019; fax: +34 958243317

Email addresses: `dperalta@decsai.ugr.es` (D. Peralta), `triguero@decsai.ugr.es` (I. Triguero), `rsreillo@ing.uc3m.es` (R. Sanchez-Reillo), `herrera@decsai.ugr.es` (F. Herrera), `J.M.Benitez@decsai.ugr.es` (J.M. Benitez)

Under consideration for publication in Pattern Recognition *June 21, 2016*

1. Introduction

Personal identification is one of the largest problems in the society today in a wide variety of fields: from access control to criminology and forensic identifications, payments and identification in computer systems [1]. Among all the biometric features that can be used for identification, such as voice, iris, DNA, etc, fingerprints are the most widely used [2]. They are very suitable for human recognition because of their uniqueness, universality, invariability and extraction facilities.

A fingerprint is basically a pattern of ridges and valleys captured from a finger by inked press, capacitive or optical sensors, etc. Fingerprint recognition has been studied for many years and a great number of fingerprint matching algorithms have been proposed in the specialized literature [3, 4]. Minutiae-based matching algorithms highlight as the most relevant approaches because minutiae are considered the most discriminating and reliable features [5, 6]. The design of Automatic Fingerprint Identification Systems (AFISs) [7] is an important task in pattern recognition. Although very effective solutions are currently available, many problems still remain [8]. Among them, the performance and speed of AFISs for large databases need to be improved.

Fingerprint recognition can be categorized into two different problems: verification [9] and identification [10]. The former consists of determining whether two images belong to the same fingerprint, that is, a one-to-one comparison. The latter is devoted to search for the matching of an input fingerprint in a template database, so that the owner of this fingerprint can be identified. Thus, identification can be seen as a generalization of the verification problem that conducts one-to-many comparisons. In this paper, we will focus on identification.

In general, matching algorithms are designed to carry out a fingerprint verification and their generalization to address identification is straightforward. Most of them are focused on achieving very accurate matchings, what usually negatively affects the time consumption. This factor is determinant in most real time systems where a high response time is equivalent to a system failure. Furthermore, this weakness is especially harmful when the number of templates in the database is increased. Although some approaches have been designed to be as fast as possible [6], they are not suitable to tackle large databases maintaining their precision.

High Performance Computing (HPC) is one of the tools that support the modern Science, allowing the execution of multiple calculations in a reasonable time [11] by using an adequate massive computational structure [12]. HPC has been successfully used in many different pattern recognition problems [13, 14, 15], and more concretely in real-time image comparison [16] and other artificial intelligence systems [17]. Given the complexity order of an AFIS, HPC is a promising resource that has already been proven to reduce the identification time [18, 19]. However, the proposals in the current scientific literature focus on objectives other than performance, such as high availability or database distribution. Real-time response times can only be obtained through a correct algorithm design and implementation in order to exploit the available resources as flexibly and efficiently as possible.

In this paper, we design a two-level distributed framework to provide matching algorithms the capacity of dealing with arbitrarily large databases by adapting the underlying hardware. According to the so far presented reasons, three

objectives are defined for this paper:

- To analyze the behavior of matching algorithms when dealing with large databases.
- To verify the scalability of the proposed system.
- To provide a real-time answer.

To check the performance of the proposed system, we will conduct experiments involving up to 400 000 fingerprints. Because of the absence of large captured fingerprint databases, we use the SFinGe software tool [20, 2] to generate a large synthetic database. This database is used for experiments both with the ground-truth minutia provided by SFinGe and using the NIGOS *mindtct* [21] minutiae extractor in a seek of a more realistic framework. Furthermore, in order to validate the results we also include experiments with captured databases: NIST DB4 [22] and DB14 [23].

Due to the space constraints not every experiment could be included in the paper. Complementary material about the work done for this paper can be found at the URL `http://sci2s.ugr.es/ParallelMatching`.

The rest of this paper is organized as follows: Section 2 provides a description of the fingerprint recognition process, defining in detail the most important steps. In Section 3, the HPC paradigm is presented, showing its hardware and software requirements, theoretical benefits and current applications to AFISs. Section 4 explains the proposed distributed system for tackling the fingerprint identification problem in a reasonable time. Section 5 describes the experimental framework. Section 6 examines the results obtained, presenting a discussion of them. Finally, Section 7 concludes the paper.

2. Background

A considerable research effort has been carried out in the fingerprint recognition field over the last decades. This section sums up the state-of-the-art in that field, starting with the fingerprint recognition problem (Section 2.1), and explaining the generalities of feature extraction (Section 2.2) and fingerprint matching (Section 2.3).

2.1. Fingerprint recognition

Because of its different application fields, most authors divide the fingerprint recognition problem into two variants that constitute by themselves different problems [2]:

- **Verification** consists of determining whether two fingerprint images P_1 and P_2 belong to the same person, performing a 1:1 comparison [9]. The system output is an acceptation or a refusal of the claimed identity depending on the similarity level (called score) of both fingerprints.
- **Identification** aims to find the fingerprint that matches with the input fingerprint in a database, so that its owner can be identified [10]. A fingerprint database is a set T of N template fingerprints $T = \{T_1, T_2, ..., T_N\}$ that are used as reference for the identification. Thus, identification is a problem of 1:N comparison as the input

fingerprint I needs to be compared with all T_i template fingerprints (with $i \in \{1, 2, ..., N\}$) to find the matching that provides the highest score. This score is called m_{best}. It is defined in Equation 1, where $Q(I, T_i)$ is the matching function (see Section 2.3). If m_{best} is lower than a certain threshold ϕ, then the system may consider that the input fingerprint has no corresponding template in the database. Hence, the system output can be the matched identity, a "not found" notification, or a set of candidate identities. This paper focuses on a system that considers only the maximum score, so the last case is not detailed, as shows Equation 2. A description of the system behavior in the case with a set of candidates can be found in the web site.

$$m_{best} = \max\{Q(I, T_i) \quad | \quad i \in \{1, 2, ..., N\}\} \tag{1}$$

$$Id(I) = \begin{cases} \text{not found}, & \text{if } m_{best} > \phi \\ \arg\max_i Q(I, T_i) \quad i \in \{1, ..., N\}, & \text{otherwise} \end{cases} \tag{2}$$

The identification problem can be seen as a verification performed once per each fingerprint in the database. The main difference between these problems is therefore a matter of complexity order. The objective in a verification problem is to obtain a very precise result, reducing the error rates as much as possible. However, complex verification methods are not useful for identification because the overall response time would be excessive.

So far, the general characteristics of the identification problem have been defined. The requirements needed by an AFIS to deal with large databases can be fixed:

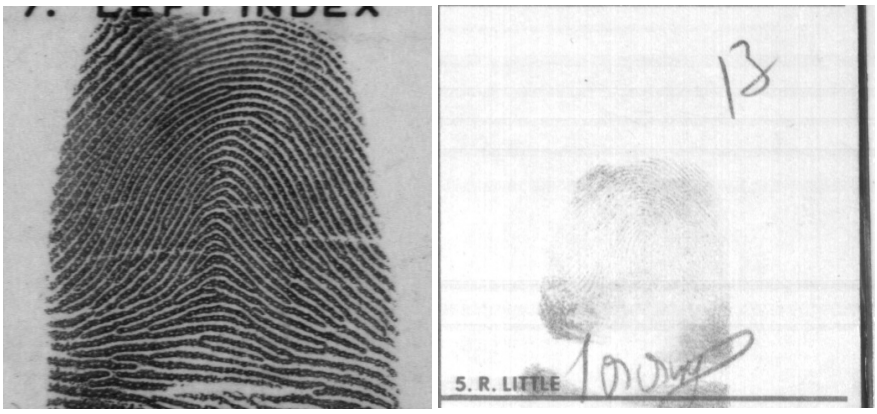
- **Precision**: error rates have to be as low as possible in order to get an accurate system. Additional information about error rates can be found in the web site associated with this paper.
- **Efficiency**: the time that is needed to locate a fingerprint in the database should be as small as possible. In a real-time system, for example, a high delay can be equivalent to a system failure [24]. The delay threshold depends on the specific system but it is very often within the order of a few seconds.
- **Scalability**: it reveals the system capabilities to deal with databases of almost arbitrary size, in a reasonable amount of time, maintaining the precision requirement. This can be done by guaranteeing that a large database can be explored in the same time than a smaller one by increasing correspondingly the underlying hardware resources.
- **Flexibility**: the system has to fit easily and efficiently any database size, any database features (such as noisy fingerprints or rollings), as well as any hardware configuration (different architectures, varying cluster size, different processors, etc.).

Although there are several solutions to the fingerprint identification problem, the general search process structure is composed of the following steps [2]:

1. Input fingerprint fetching
2. Feature extraction
3. Search of a similar fingerprint in the database
4. Returning the result

2.2. Feature extraction

A fingerprint is basically formed by ridges and valleys. They can be easily appreciated in a good quality image (Fig. 1a), or on the contrary they can be blurred or even indistinguishable (Fig. 1b), difficulting the knowledge extraction process.



(a) Good quality image (b) Bad quality image

Figure 1: Good and bad quality images.

As they are analyzed at different degrees theses ridges and valleys present some patterns that can be used to perform the fingerprint comparison. The most relevant features, ordered from the most global to the most local, are the following [2]:

- **Singular points**: they are detected at the most global level. They are points around which the ridge patterns are wrapped. There are two kinds of them: loops and deltas, and a fingerprint can have between zero and five singular points.
- **Orientation map**: it belongs to the same level as singular points and contains the direction of the fingerprint lines for each coordinate in the image.
- **Minutiae**: they are the ridge bifurcations and endings, which are detected at a more detailed level (Fig. 2).

Among these kinds of patterns, minutiae are the most used features for fingerprint recognition [2]. Some studies state that they are the most reliable features for these purposes [25, 26], and that twelve perfectly matching minutiae

Figure 2: Fingerprint minutiae with their orientation.

between two fingerprints can ensure they are the same [27]. However, in bad quality images their extraction can be troublesome [28].

A minutia M_i is typically described with five parameters $(x_i, y_i, \theta_i, t_i, q_i)$:

- (x_i, y_i): coordinates in the picture
- θ_i: orientation or minutia angle
- t_i: type (ridge ending or bifurcation)
- q_i: quality

Therefore, a fingerprint F with r minutiae can be represented as a minutiae vector $\{M_1, M_2, ..., M_r\}$.

The number of minutiae r is typically between 30 and 100. Thus, minutiae can be efficiently stored and easily handled in a computing environment, and fingerprint comparison can be treated as a similarity calculation between minutiae sets.

There are two main types of minutiae extractors [2]:

- **Binarization-based methods**: most of the methods require a binary fingerprint image. The image usually passes through a thinning process that reduces the line thickness to one pixel, resulting in a skeleton image. Although these steps are time-consuming and may cause some information loss, they allow the minutiae detection with a simple image scan and they greatly benefit from previous enhancement processes such as the approaches presented in [29, 30, 31]. Some methods of this type are NIGOS *mindtct* [21], and an approach based on peak detection along sections orthogonal to the ridge orientation [32]. Additionally, other methods improve the image quality before the thinning step, for example by using adaptive windows to follow the ridges and find the gaps and holes [33].

- **Direct gray-scale extractors**: some methods do not use binarization nor thinning. Therefore, there is no information loss and the time spent on binarization and thinning steps is avoided, but these methods do not benefit from a priori enhancements. One of the most used methods uses the orientation map to follow the ridges [5], and is used as a basis by further proposals [34, 35, 36, 37, 38]. Other methods use alternatives to ridge-line tracking, such as neural networks [39] or spatial filtering [40].

2.3. Matching

A matching algorithm compares the features of two fingerprints and returns a similarity score. The algorithm and the data structures it uses depend on the specific features that are extracted from the fingerprint image, allowing the following classification of matchers [2]:

- Correlation-based [41, 42]
- Minutiae-based [43, 28, 6, 44]
- Non-minutiae feature-based [33, 4]

This paper focuses on minutiae-based matchers, whose usual data structures are the following:

- Distance between minutiae
- Minutiae neighborhood
- Number of ridges between minutiae (ridge count)

A matching algorithm performs some calculations from these structures and the fingerprint features themselves and returns a score (typically a real number) that describes the similarity level ranging from completely different fingerprints to the totally identical pictures.

The minutiae-based matching process can be performed at three different levels [2, 6]:

- **Global**: The minutiae of the whole image are compared. This matching type is more sensitive to image distortions, rotations and translations, although the usage of information of the whole image at the same time provides a complete view of the fingerprint. Some proposals are presented in [45, 46].
- **Local**: Small groups of minutiae close to each other are compared. Problems due to rotations and translations are softened because the use of relative angles and coordinates makes the method rotation and translation invariant. The distortion problem is also reduced because close minutiae are less affected by distortions. However, not considering the fingerprint as a whole implies a loss of information that can affect the precision of the algorithm. Some approaches are described in [47, 28].

- **Hybrid**: Most reliable algorithms use a hybrid approach, combining both philosophies. First, a local matching extracts the most similar minutiae groups of both fingerprints. These minutiae are considered to be the same, and then a global matching based on this correspondence is executed. Some of the most relevant proposals are [43, 6].

3. High Performance Computing

HPC systems are normally used for distributed and parallel computing, providing several advantages:

- **Efficiency**: the parallel processing in several cores and computers can be used to get results faster.
- **Robustness**: the use of several machines allows the system to be fault-tolerant, because if one machine fails, the rest can assume its work and the system still provides a correct response.
- **Scalability**: hardware evolves towards a higher number of cores and collaborating processors. Thus, an algorithm that is able to solve bigger problems just by using more computers could solve arbitrarily big problems without being modified.

In Sections 3.1 and 3.2, the hardware and software that give support to an HPC system are described. Section 3.3 presents the theoretical expectation of improvement in the execution times of a generic system that uses HPC. Finally, the state-of-the-art about distributed AFISs is studied in Section 3.4.

3.1. Hardware support

Hardware has evolved in two ways to support HPC. On the one hand, several computers can be integrated with a high-speed network to form a cluster. This provides a great flexibility when the processing capacity has to be increased, but the performance can become limited by the network speed.

On the other hand, a single computer can have several processors, a single processor can have several cores, and a single core can handle several execution threads (for example, with the Intel Hyperthreading technology [48]). All these processors and cores can communicate using shared memory, which is very fast, as long as the synchronization is efficiently performed. This is not always easy and may imply great design and implementation efforts. However, the number of cores in a single computer is still quite limited, and nowadays is not higher than about 12 or 24.

A typical computing cluster is formed by a bunch of computers, and each one of them has one or several multicore processors, where all the cores share the main memory and some cache. This kind of clusters are called *hybrid* clusters (Fig. 3).

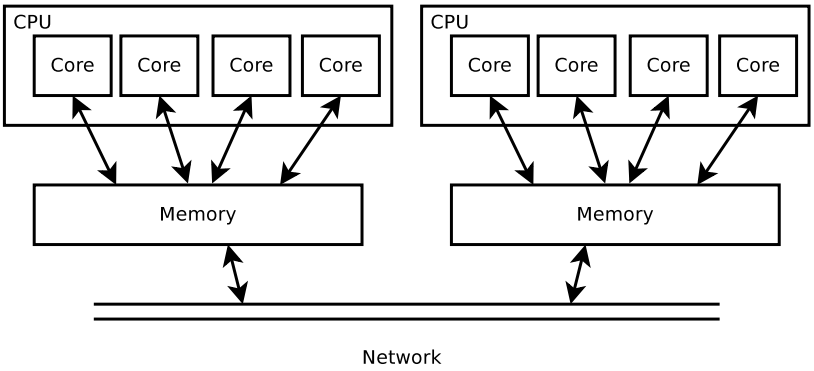


Figure 3: Typical hybrid architecture of the cluster used for the experiments in Section 6.

3.2. Software support

According to the current evolution of technology, the parallel paradigm for software development is bound to be increasingly necessary in the next years (and most likely in a longer term too).

Within a hybrid cluster, the computing program is typically divided into several processes and each process is run in a different node. These processes communicate using Message Passing Interface (MPI) [1] . Again, each process can be divided into several execution threads that can communicate using shared memory, which is faster than MPI. The maximum performance is usually reached when each computing node executes a single process that contains one or two threads per node core. Thus, the adequate implementation of a system in a computing cluster is a complex task.

The execution of these processes and threads can be tackled by the operating system (for example when using C++), or by a virtual machine (for instance with Java, Scala or Erlang).

3.3. Theoretical expectations

There are several formulas to measure the performance of a parallel system. The most widely used is the speedup ($S = t_s/t_p$), which measures the relation between the execution times of the sequential (t_s) and parallel (t_p) versions of a same calculation.

If a calculation is executed in n processing cores, and a portion f of the calculation is performed in parallel, the maximum attainable speedup would be S^*, according to the Amdahl's Law [49], which is shown in Equation 3.

$$S^* = \frac{1}{(1-f) + \frac{f}{n}} \tag{3}$$

Therefore, if the calculation is fully parallelizable ($f = 1$) the maximum speedup would be equal to the number of cores (n). However, in practice the attained speedup is lower than this maximum due to several factors:

[1] `http://www.mpi-forum.org/`

- There is always some part of the calculation that is not parallelizable $(1 - f)$. Even if this part is very small it can represent a very big speedup loss when the number of parallel cores is high, as it can be seen in Equation 4, which shows the maximum speedup for a certain f even if the number of processors is arbitrarily high. Therefore, it is crucial to reduce as much as possible the fraction of non-parallelizable calculation.

$$\lim_{n \to +\infty} \frac{1}{(1 - f) + \frac{f}{n}} = \frac{1}{1 - f} \tag{4}$$

- A parallel application includes extra communication and synchronization workloads that are not necessary in sequential programs.

- When some threads or processes finish their workload before others, the hardware does not work at full capacity any more because some of the processing cores remain idle, waiting for new tasks to be assigned.

However, there are some cases where a superlinear speedup can be attained. One of them is when the amount of processed information does not fit in the main memory of a single computer. If several computers collaborate, the total amount of available memory is higher and then the necessity of slow hard-disk accesses can be removed.

Finally, the relationship between processing (t_{pr}) and communication (t_c) workload as the problem size increases is also important ($R_{pc} = t_{pr}/t_c$). If the processing workload is higher, a bigger cluster would be useful in order to improve the performance. However, if there is more communication as the problem size increases, there would be a bottleneck and the use of more machines would not imply faster results.

3.4. Distributed AFISs: proposals in the specialized literature

According to the preceding sections, the features provided by HPCs are very similar to the AFIS objectives described in Section 2. HPC is a promising tool for the design of a flexible and scalable AFIS because it would allow a parallel search through the fingerprint database, providing an increased system performance [18, 50].

At the time of writing this paper there are several AFISs in the specialized literature and also in the commercial market. Most of these systems have an acceptable performance when they deal with small databases. Nevertheless, in most real world problems there is a need of finding a person among databases whose sizes can range from tens of thousands to tens of millions. These identifications must be performed in a reasonable time, often shorter than a threshold of a few seconds. Furthermore, as explained in Section 2, the larger the size of the fingerprint database, the harder it is to obtain a good identification accuracy.

Within this context, the bottleneck step in the identification process is the matching algorithm, because it must be performed once per each database fingerprint to determine which one is the most similar to the input.

The proposals in the specialized literature can be classified into different categories:

- **Client-server systems**: in [51], the authors propose a server-like AFIS where the fingerprint database is distributed among several servers. When a client requests an identification from a server, it searches the input

fingerprint in its database portion. If it succeeds, it sends the response to the client. However, if the fingerprint is not found in the server, the request is forwarded to other servers and the server acts as a client. Therefore, this system does not process the information in parallel. The distribution only affects the database and not the processing, and the overall processing time is higher than in a sequential AFIS. This makes this architecture unsuitable for very large databases with hundreds of thousands of fingerprints. A similar system is described in [19], that additionally includes a GPS-based system for an increased security.

The objective of these systems is to provide an AFIS for distributed databases, whereas this paper focuses on attaining low identification times in large databases. Thus, no comparison can be performed between these systems.

- **Agent-based systems**: in [52] an agent-based system is presented, mostly oriented to heterogeneous hardware architectures. The novelty of this work is that it uses the idle times of a bunch of computers that are mainly used for other purposes, especially desktop machines. The main part of this proposal is therefore a load-balancing algorithm. The system has a master-slave structure where a set of slave agents compare fingerprints and a master agent distributes and organizes the computing workload. The proposed architecture is divided into layers that isolate the resource monitoring, the agent manager and the matching algorithm. A similar, less centralized approach is presented in [53], where slave agents are able to communicate and share their found scores. Several processes are dynamically created when an input fingerprint is received to better distribute the database exploration. Although this may improve the system flexibility, there is a negative impact on the identification time.

  Again, the objective in these systems is not performance, but load-balancing between shared machines. The execution times shown in [52] are of 3 minutes and 14 second for performing 700 matchings in a set of 20 Pentium IV machines. This result shows that this approach is not able to handle identifications through hundreds of thousands of fingerprints in no more than a few seconds, as is the requirement for most real-time biometric systems.

To sum up, there are some solutions and ideas to improve the efficiency and the availability of AFISs; however, there is no really scalable AFIS available in the current scientific literature.

4. Distributed and scalable AFIS framework

As it has been explained in previous sections, the bottleneck in a traditional AFIS is the matching process, that has to be executed once for each fingerprint in the database. This makes the system less usable when it comes to deal with large or very large databases (from tens of thousands of fingerprints onwards) as the response time becomes too high. However, the fingerprint identification problem is naturally parallelizable, because the comparisons of the input fingerprint I with each one of the N fingerprints T_i in the database are entirely independent. This feature can

be exploited by designing a flexible and efficient parallel identification system based on the HPC paradigm, which eliminates the bottleneck.

The proposed system is described as follows: Section 4.1 details its parallel structure, Section 4.2 describes the database distribution and Section 4.3 explains the distributed search process.

4.1. Two-level parallelization

As described in Section 3.1, a typical computer cluster has two parallelism levels. Both nodes and cores contribute to the system performance and can execute processes by themselves; however, they must be handled by the software in a different way if a maximum performance must be attained.

The proposed software system (which is implemented in C++) consequently has a two-level parallelization:

- **Processes**: typically one per node, they are handled with MPI. [1]
- **Threads**: one or several per process, they are handled with OpenMP. [2]

There is a single process (called "master") which reads the input fingerprint and gathers the results at the end of a search. All the other processes are called "slaves", and perform parts of the search executing the matching algorithm. Each slave loads its corresponding fraction of the database and searches the input fingerprint in it. Additionally, each slave process is itself formed by one or more threads, therefore its database fraction can be divided over again and the threads perform a parallel search within each process.

4.2. Database distribution

Suppose a generic system with N fingerprints, p nodes and h threads per node. The database would be divided into one portion per node, so that each process searches in its corresponding portion of N/p fingerprints (Fig. 4). This distribution can be physical, if the fingerprints are stored in their corresponding nodes in order to improve the access time and avoid the bottlenecks of a centralized database, or merely logical if the database is centralized.

Inside each node, the process performs a logical partition of its database portion. Hence, each thread searches through only $N/(ph)$ fingerprints. This scheme allows N, p and h to be modified in a totally flexible way, so they can be adjusted to any hardware (from single-core computers to hybrid clusters), any environment conditions and any database to obtain a maximum performance gain.

Fig. 5 represents the interval size for each thread as a function of the total number of threads ph.

4.3. Distributed search process

The logic of the system remains the same independently of how many nodes or threads are used, and is depicted in Fig. 6.

[2] `http://openmp.org/wp/`

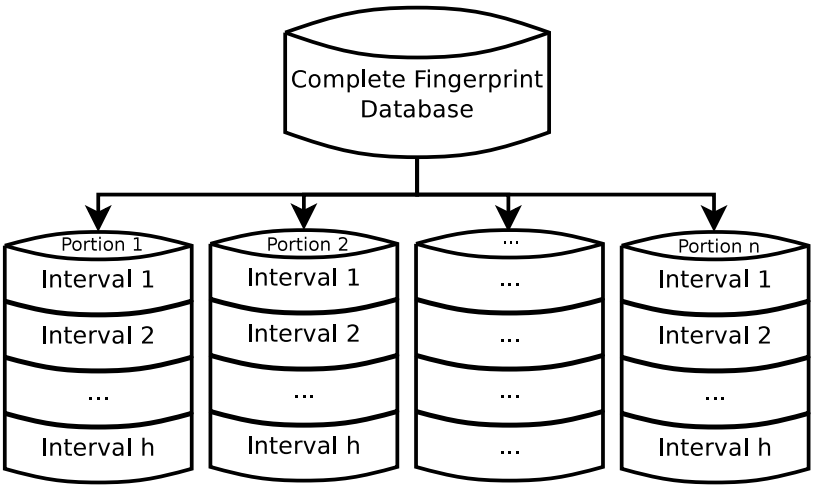


Figure 4: Database partition for nodes and threads

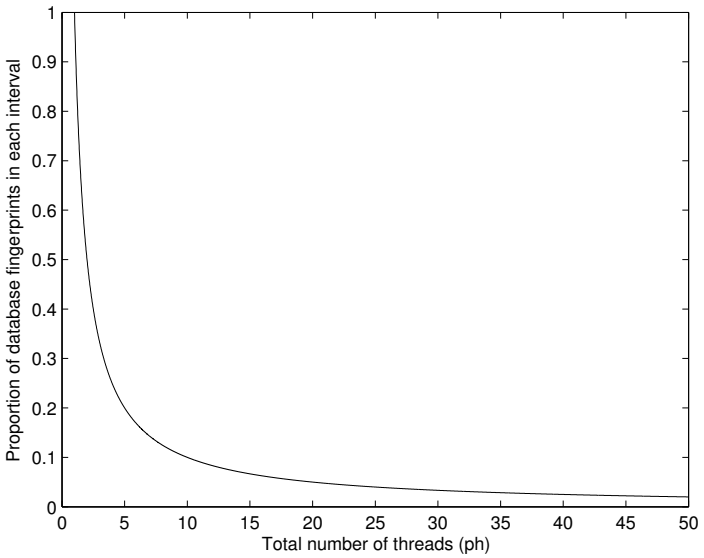


Figure 5: Relative interval size for each thread

1. **Initialization**: this step is executed only once, and then the system can perform as many identifications as required. The database partitions are established, each slave loads its part of the database, preprocesses it if necessary, and the master waits for input fingerprints.
2. **Identification loop**
   (a) The master receives an input fingerprint. As the feature extraction is performed only once, it can be computed either in the master process or in each of the slaves independently, depending on the system features.
   (b) When a slave gets the fingerprint features, each one of its threads performs $\frac{N}{ph}$ matchings to compare it with the template fingerprints in its database portion.
   (c) Each slave sends its successful matches to the master process.
   (d) The master computes the results and gives a response to the user.

This scheme ensures that the bottleneck step (number 2.b) is executed with two levels of parallelism, in order to accelerate the execution as much as possible and eliminate the bottleneck. Moreover, as the matching algorithm remains the same as in a sequential search, there is no loss of precision and the system is guaranteed to find exactly the same solution as the sequential model in much less time. This also makes the system independent of the matching algorithm, which can be easily replaced.

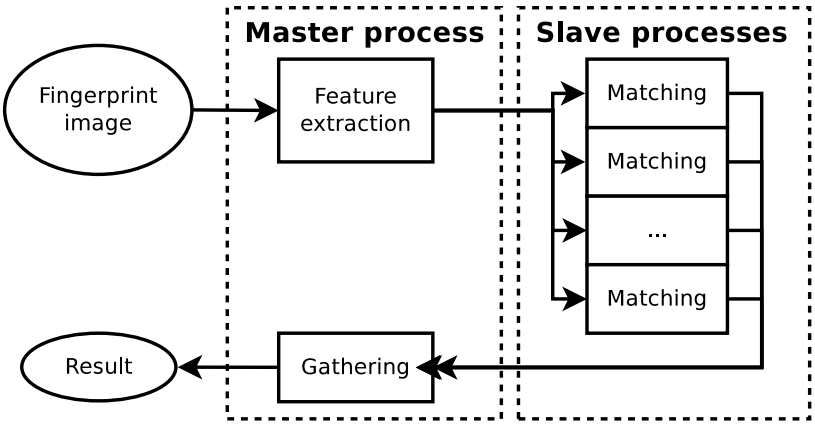


Figure 6: Processing in the proposed distributed model

According with these data and Amdahl's Law (Section 3.3), this system can obtain a maximum speedup of $\frac{1}{ph}$. In that case the identification time plot would have the same hyperbolic shape as the partition size in Fig. 5.

The proposed distributed system shows several important advantages:

- Very high speedup because of several factors:
  - Independent processing among slave processes
  - Independent processing among threads within each slave

  - Minimum communication overhead
  - Optimal exploitation of the hardware structure
- Adaptability to multiple sequential or distributed platforms and architectures
- Flexibility for centralized or distributed databases
- Flexibility for any matcher or feature extractor
- Same precision as the sequential model

5. Experimental setup

This section describes the experimental framework for this paper. The aim of this experimental study is to check the system scalability —along with its adaptability to the underlying hardware system— in several aspects:

- The number of computing nodes
- The number of threads in each node
- The size of the database

All the performed experiments have the same structure: first, a fingerprint database is loaded in the system and all its fingerprints are preprocessed according to the corresponding matching algorithm; then, a set of input fingerprints are searched throughout the database for their identification. All the presented results are averages of the identification times obtained for 1000 input fingerprints. For the sake of readability and reasons of space, the standard deviations are not included, but they can be found in the web site associated with this paper. The penetration rate is 100% for this setup, as there is no stopping criterion for the search. As explained in Section 4.3, the proposed system provides the same identifications as a traditional sequential AFIS. A large description of the possible stopping criterion is presented in the associated web site, as well as the precision results for all the databases used in this paper.

Firstly, the hardware and software support are defined and detailed in Section 5.1. Then, Section 5.2 describes the large synthetic databases created with SFinGe. Finally, Section 5.3 details the captured databases that are used in the experiments.

5.1. Hardware and software environment

The experiments have been carried out on up to twelve nodes in a cluster. Each of these nodes has the following features:

- **Processors**: 2 x Intel Xeon CPU E5-2620
- **Cores**: 6 per processor (12 threads)

- **Clock Speed**: 2.00 GHz
- **Cache**: 15 MB
- **Network**: Gigabit Ethernet (1 Gbps)
- **RAM**: 64 GB

One of the nodes acts as the interface with the user and hosts the master process. However, as this process does not perform any major processing tasks, a slave process can also be executed on the same node without compromising the performance and thus the hardware is more efficiently exploited.

The proposed distributed model has been implemented in C++, using the OpenMPI 1.6 library[3] for the communication and synchronization of processes. Similarly, the OpenMP library [2] has been used for handling the threads within each process. In all databases where the fingerprint features had to be extracted, the NIGOS *mindtct* [21] algorithm was used.

Table 1: Parameters for the methods used in the experimentation

Algorithm	**Parameters**	**Reference**
Mindtct	output format = ANSI INCITS 378-2004 image enhancement = enabled	[21]
Jiang	$w_d = 1, w_\theta = 54\pi, w_\phi = 54\pi, w_n = 0, w_t = 0$ Consolidation step iterations = 5, Minutiae neighborhood size = 2 $BG_1 = 8, BG_2 = \frac{\pi}{6}, BG_3 = \frac{\pi}{6}$	[43]
Chen	$Thr_L = 55, Thr_H = 80, R = 80, RS = 100, \theta_L = 0.25, \theta_H = 0.4$ $\text{len}_L = 5, \text{len}_H = 20, Thr_{topo} = 0.7$	[28]
MCC16	$R = 70, N_s = 16, N_d = 6, \sigma_s = \frac{28}{3}, \sigma_d = \frac{2\pi}{9}, \mu_\Psi = 0.01, \tau_\Psi = 400$ $\omega = 50, min_{VC} = 0.75, min_M = 2, min_{ME} = 0.60, \sigma_\theta = \frac{\pi}{2}, max_{n_p} = 12$ Floating-point-based version: enabled, $\mu_P = 20$ $w_R = 0.5, \mu_1^\rho = 5, \tau_P = 0.6, min_{n_p} = 4$ $\mu_2^\rho = \frac{\pi}{12}, \mu_3^\rho = \frac{\pi}{12}, \tau_1^\rho = -1.6, \tau_2^\rho = -30, \tau_3^\rho = -30, n_{rel} = 5$	[6]

Three different matching algorithms of the state-of-the-art literature have been used within the framework:

- **Jiang** is a classical hybrid matching algorithm [43]. Each minutia is described with a feature vector that depends on its neighboring minutiae, and the feature vectors of both fingerprints are compared in pairs. The algorithm

[3] `http://www.open-mpi.org/`

assumes that the most similar pair corresponds to the same minutia in both fingerprints and compares the rest of the minutiae using relative coordinates and angles (avoiding the translation and rotation problems).

- **Chen** focuses on getting robustness despite of the fingerprint distortion [28]. The algorithm is mostly local, as it calculates the local topology for each minutiae given a fixed radius. Then, it compares the local topologies of both fingerprints, and if they are similar enough, it repeats the comparison with a modified radius to avoid problems with the image distortion.

- **Minutia-Cylinder-Code (MCC)** uses both local and global information to perform the matching [6]. For each minutia, a tridimensional cylinder is built and discretized in cells. Each cell is given a value that depends on its position and the relative position of neighboring minutiae. According to this number, the cell can be declared either valid or invalid, so that only cylinders with a minimum number of valid cells are taken into account for the matching process. This process compares the cylinders of both fingerprints cell by cell and merges the results (global matching) to get the score.

  This algorithm has a binary and a real version. In this paper, we focus on the latter, which is more precise and more suited for general purpose machines. We also fix 16 cells as the cylinder side size in order to get the most accurate configuration, which is also the most computationally complex. Results for the version with $Ns = 8$ are included in the web site associated to this paper.

All three algorithms have been implemented by the authors of this paper, with the only help of the information shown at each of the referred papers. All the used methods parameters are common for all databases, and they were selected according to the recommendation of the corresponding authors (Table 1).

5.2. SFinGe databases: ground-truth minutiae and NIGOS mindtct extraction

A correct scalability study requires a very large database. However, there is no public captured fingerprint database big enough to cover this need, so we used SFinGe [20, 2] to generate a database with 400 000 synthetic fingerprints, using the parameters described in Table 2 to ensure the generation of realistic fingerprints.

SFinGe randomly generates the fingerprint minutiae and calculate a fingerprint image from them, following patterns so that the resulting synthetic fingerprints behave as natural captures. As SFinGe is able to provide the generated minutiae as an additional output, this paper has used both the returned ground-truth minutiae and the extracted minutiae (using *mindtct*), obtaining two databases with the same fingerprints but slightly different characteristics.

For each fingerprint 25 impressions have been generated. One of the impressions is selected as template, and the rest are considered input fingerprints. Then, several subsets of the whole database (each of them of increasing sizes) have been selected, respecting the natural class distribution, in such a way that each database contains the immediately smaller one. The whole enrollment process is described in the associated web site.

The size of all database subsets are presented in Table 3, along with the average number of minutiae for both template and input fingerprints. As it can be seen, the number of average minutiae is higher for the extracted feature

Table 2: Parameter specification used with the SFinGe tool

Scanner parameters
Acquisition area: 0.58” x 0.77” (14.6mm x 19.6mm).
Resolution: 500 dpi.
Image size: 288 x 384.
Background type: Optical.
Background noise: Default.
Crop borders: 0 x 0.
Generation parameters
Impression per finger: 25.
Class distribution: Natural.
Set all distributions as: “Varying quality and perturbations”.
Generate pores: enabled.
Save ISO templates: enabled.
Output settings
Output file type: WSQ.

Table 3: SFinGe databases size and average number of minutiae

DB Size	**Ground-truth**		**Extracted**	
	Template	**Input**	**Template**	**Input**
1000	40.79	36.84	55.35	49.60
2000	40.84	36.81	55.47	49.61
5000	40.97	36.98	55.64	49.87
10 000	40.79	36.77	55.48	49.61
50 000	40.72	36.70	55.44	49.58
100 000	40.73	36.71	55.46	49.62
200 000	40.74	36.71	55.50	49.63
400 000	40.70	36.68	55.47	49.66

vectors due to the noise introduced by the image generation and the processing steps. This implies that *mindtct* extracts an average of 15 spurious minutiae per fingerprint.

Finally, we have selected one random input impression for each fingerprint in the smallest database, obtaining a

test set of 1000 input fingerprints that is valid for the experiments with all the generated databases.

Table 4: Execution times and speedup with the MCC16 algorithm

Slaves	*DB size*	**1 thread**		**4 threads**		**12 threads**		**24 threads**	
		Time (s)	*Speedup*	*Time (s)*	*Speedup*	*Time (s)*	*Speedup*	*Time (s)*	*Speedup*
1	1000	6.5377	1.0000	1.6867	3.8759	0.6444	10.1447	0.5043	12.9646
	2000	12.9968	1.0000	3.2963	3.9429	1.2041	10.7936	0.9059	14.3463
	5000	32.5242	1.0000	8.1634	3.9841	2.9056	11.1937	2.1228	15.3212
	10 000	64.7076	1.0000	16.2010	3.9940	5.7484	11.2566	4.1308	15.6645
	50 000	321.8924	1.0000	81.9340	3.9287	28.2669	11.3876	20.1745	15.9554
	100 000	624.3297	1.0000	160.9759	3.8784	56.5323	11.0438	40.2844	15.4980
	200 000	1250.2594	1.0000	314.9397	3.9698	113.1789	11.0468	80.4265	15.5454
	400 000	–	–	–	–	–	–	–	–
2	1000	3.3728	1.9384	0.9062	7.2142	0.3633	17.9976	0.3081	21.2203
	2000	6.5398	1.9873	1.6969	7.6592	0.6457	20.1289	0.5144	25.2667
	5000	16.3442	1.9900	4.1567	7.8246	1.4826	21.9367	1.1156	29.1537
	10 000	32.5416	1.9885	8.1786	7.9118	2.9220	22.1446	2.1294	30.3882
	50 000	161.8845	1.9884	40.3953	7.9686	14.1720	22.7132	10.1626	31.6743
	100 000	324.4350	1.9244	80.6912	7.7373	28.2606	22.0919	20.1649	30.9612
	200 000	625.5980	1.9985	161.2985	7.7512	56.5520	22.1081	40.2440	31.0669
	400 000	1248.7902	–	322.3448	–	109.9070	–	80.4963	–
4	1000	1.8610	3.5129	0.5247	12.4603	0.2329	28.0741	0.2133	30.6491
	2000	3.3892	3.8347	0.9121	14.2499	0.3680	35.3140	0.3172	40.9691
	5000	8.2977	3.9197	2.1465	15.1521	0.7898	41.1816	0.6155	52.8417
	10 000	16.3510	3.9574	4.1657	15.5334	1.4917	43.3775	1.1096	58.3179
	50 000	80.8244	3.9826	20.3091	15.8497	7.1278	45.1600	5.1219	62.8463
	100 000	161.5434	3.8648	40.4924	15.4185	14.2047	43.9523	10.1475	61.5253
	200 000	312.9654	3.9949	80.7448	15.4841	28.2520	44.2538	20.1662	61.9977
	400 000	621.7593	–	158.2815	–	56.4944	–	40.2569	–
8	1000	0.9849	6.6382	0.3098	21.1005	0.1681	38.8978	0.1750	37.3486
	2000	1.8604	6.9859	0.5302	24.5143	0.2335	55.6547	0.2195	59.2200
	5000	4.2340	7.6816	1.1434	28.4443	0.4392	74.0569	0.3691	88.1285
	10 000	8.3241	7.7735	2.1611	29.9425	0.7880	82.1115	0.5932	109.0777
	50 000	40.4281	7.9621	10.2904	31.2810	3.5670	90.2412	2.5289	127.2860
	100 000	80.9415	7.7133	20.2252	30.8689	7.0044	89.1343	4.9369	126.4614
	200 000	161.5014	7.7415	40.3357	30.9963	14.1784	88.1803	10.1298	123.4243
	400 000	315.6620	–	80.5549	–	28.2313	–	20.1449	–
12	1000	0.6955	9.4000	0.2366	27.6279	0.1384	47.2470	0.1616	40.4603
	2000	1.2825	10.1340	0.3841	33.8357	0.1885	68.9394	0.1909	68.0741
	5000	2.9076	11.1858	0.8120	40.0537	0.3320	97.9537	0.2866	113.4908
	10 000	5.5884	11.5789	1.4935	43.3259	0.5632	114.8979	0.4399	147.0949
	50 000	27.1921	11.8377	7.0135	45.8961	2.4538	131.1832	1.7928	179.5466
	100 000	54.1348	11.5329	13.5321	46.1370	4.8029	129.9911	3.4581	180.5417
	200 000	104.8093	11.9289	26.9958	46.3132	9.4815	131.8623	6.7807	184.3857
	400 000	213.1696	–	51.5151	–	18.8523	–	13.4805	–

The execution parameters take the following values for the experiments with these database subsets, producing a total of 480 executions of 1000 identifications each:

- **Number of nodes**: 1, 2, 4, 8 and 12
- **Number of threads per node**: 1, 4, 12 and 24

Table 5: Execution times and speedup with Jiang and Chen algorithms

Slaves	*DB size*	**Jiang**								**Chen**							
		1 thread		**4 threads**		**12 threads**		**24 threads**		**1 thread**		**4 threads**		**12 threads**		**24 threads**	
		Time (s)	*Speedup*	*Time (s)*	*Speedup*	*Time (s)*	*Speedup*	*Time (s)*	*Speedup*	*Time (s)*	*Speedup*	*Time (s)*	*Speedup*	*Time (s)*	*Speedup*	*Time (s)*	*Speedup*
1	1000	0.2223	1.0000	0.0572	3.8865	0.0210	10.6091	0.0322	6.9010	2.6166	1.0000	0.6707	3.9014	0.2631	9.9439	0.2158	12.1275
	2000	0.4491	1.0000	0.1143	3.9292	0.0418	10.7486	0.0469	9.5809	5.2484	1.0000	1.3442	3.9046	0.5200	10.0937	0.4094	12.8186
	5000	1.1262	1.0000	0.2871	3.9232	0.1048	10.7434	0.0898	12.5358	13.2122	1.0000	3.3910	3.8963	1.3043	10.1294	1.0083	13.1038
	10 000	2.2432	1.0000	0.5710	3.9286	0.2072	10.8245	0.1670	13.4285	26.3182	1.0000	6.7189	3.9170	2.6000	10.1224	1.9823	13.2764
	50 000	11.2532	1.0000	2.8536	3.9434	1.0443	10.7753	0.7408	15.1909	130.0427	1.0000	33.3673	3.8973	12.9129	10.0708	9.7984	13.2719
	100 000	22.3477	1.0000	5.6989	3.9214	2.0743	10.7737	1.4444	15.4715	261.2312	1.0000	66.7052	3.9162	25.8390	10.1099	19.5891	13.3355
	200 000	44.7627	1.0000	11.4571	3.9070	4.1789	10.7115	2.8860	15.5104	520.8561	1.0000	133.8296	3.8919	51.7934	10.0564	39.2616	13.2663
	400 000	89.4130	1.0000	22.8261	3.9171	8.3137	10.7549	5.7485	15.5541	1041.2164	1.0000	268.2014	3.8822	103.6681	10.0437	78.6457	13.2393
2	1000	0.1142	1.9461	0.0294	7.5499	0.0108	20.6412	0.0292	7.6174	1.3740	1.9044	0.3538	7.3956	0.1360	19.2343	0.1180	22.1809
	2000	0.2233	2.0118	0.0575	7.8125	0.0213	21.1162	0.0364	12.3264	2.6151	2.0069	0.6745	7.7813	0.2643	19.8543	0.2176	24.1182
	5000	0.5647	1.9942	0.1455	7.7383	0.0522	21.5809	0.0579	19.4507	6.6614	1.9834	1.7315	7.6305	0.6522	20.2588	0.5073	26.0449
	10 000	1.1266	1.9911	0.2917	7.6913	0.1041	21.5529	0.0940	23.8630	13.2512	1.9861	3.4134	7.7102	1.3075	20.1283	1.0074	26.1243
	50 000	5.5878	2.0139	1.4407	7.8108	0.5222	21.5507	0.3782	29.7538	65.3116	1.9911	16.8256	7.7289	6.4966	20.0171	4.9242	26.4089
	100 000	11.2132	1.9930	2.8599	7.8141	1.0442	21.4017	0.7306	30.5885	130.0469	2.0087	33.5380	7.7891	12.9157	20.2258	9.7965	26.6656
	200 000	22.3400	2.0037	5.7152	7.8323	2.0770	21.5513	1.4477	30.9198	263.4494	1.9771	67.1486	7.7568	25.8201	20.1725	19.5915	26.5858
	400 000	44.7918	1.9962	11.4334	7.8203	4.1682	21.4514	2.8645	31.2138	494.2013	2.1069	134.0177	7.7692	51.7896	20.1047	39.2861	26.5034
4	1000	0.0636	3.4950	0.0160	13.8813	0.0060	36.8029	0.0237	9.3672	0.7832	3.3409	0.2000	13.0831	0.0773	33.8613	0.0773	33.8581
	2000	0.1154	3.8922	0.0297	15.1395	0.0111	40.4738	0.0294	15.2556	1.3838	3.7928	0.3590	14.6187	0.1362	38.5442	0.1155	45.4296
	5000	0.2848	3.9547	0.0731	15.4160	0.0268	42.0524	0.0382	29.4869	3.3920	3.8951	0.8811	14.9944	0.3295	48.0000	0.2660	49.6665
	10 000	0.5649	3.9707	0.1456	15.4113	0.0526	42.6104	0.0582	38.5593	6.6707	3.9454	1.7267	15.2417	0.6550	40.1810	0.5116	51.4412
	50 000	2.8063	4.0099	0.7184	15.6639	0.2631	42.7751	0.1996	56.3777	33.6899	3.8600	8.4567	15.3775	3.2448	40.0772	2.4605	52.8523
	100 000	5.5953	3.9940	1.4330	15.5947	0.5248	42.5867	0.3764	59.3691	65.3531	3.9972	16.8876	15.4688	6.4745	40.3479	4.9155	53.1444
	200 000	11.2069	3.9942	2.8843	15.5196	1.0416	42.9750	0.7293	61.3813	130.2397	3.9992	33.7397	15.4375	12.9011	40.3730	9.7735	53.2930
	400 000	22.3816	3.9949	5.7571	15.5309	2.0919	42.7426	1.4346	62.3251	244.8766	4.2520	67.2316	15.4870	25.8130	40.3369	19.5542	53.2478
8	1000	0.0316	7.0401	0.0083	26.7494	0.0035	64.2974	0.0290	7.6733	0.4037	6.4812	0.1054	24.8257	0.0428	61.1117	0.0515	50.8454
	2000	0.0620	7.2497	0.0161	27.9362	0.0062	72.6729	0.0212	21.1652	0.7753	6.7691	0.2062	25.4555	0.0775	67.7110	0.0758	69.2445
	5000	0.1451	7.7641	0.0378	29.7743	0.0137	82.4991	0.0132	85.4633	1.7007	7.7686	0.4545	29.0717	0.1694	77.9951	0.1482	89.1464
	10 000	0.2842	7.8943	0.0736	30.4730	0.0269	83.4649	0.0189	118.5300	3.3823	7.7811	0.9019	29.1795	0.3319	79.2880	0.2626	100.2375
	50 000	1.3982	8.0483	0.3657	30.7747	0.1338	84.0739	0.0865	130.1263	16.2990	7.9786	4.3068	30.1949	1.6056	80.9917	1.2014	108.2427
	100 000	2.8018	7.9761	0.7160	31.2100	0.2617	85.4097	0.1676	133.3246	32.6356	8.0045	8.3803	31.1721	3.2778	79.6972	2.3699	110.2273
	200 000	5.5918	8.0050	1.4278	31.3501	0.5252	85.2298	0.3771	118.6958	65.4597	7.9569	16.7699	31.0590	6.4707	80.4948	4.9169	105.9329
	400 000	11.2094	7.9766	2.8535	31.3340	1.0465	85.4367	0.7311	122.2957	130.3891	7.9855	33.5594	31.0261	12.9884	80.1652	9.8312	105.9092
12	1000	0.0217	10.2596	0.0059	37.8051	0.0026	84.7512	0.0355	6.2588	0.2860	9.1492	0.0750	34.8743	0.0319	82.0586	0.0462	56.6480
	2000	0.0420	10.7056	0.0111	40.5654	0.0045	100.4881	0.0305	14.7041	0.5355	9.8015	0.1402	37.4315	0.0555	94.5845	0.0605	86.7839
	5000	0.0987	11.4113	0.0257	43.7860	0.0099	113.8505	0.0245	46.0084	1.1946	11.0598	0.3207	41.1937	0.1207	109.4223	0.1097	120.3981
	10 000	0.1913	11.7289	0.0496	45.2401	0.0180	124.5452	0.0294	76.2192	2.2545	11.6735	0.6144	42.8372	0.2249	117.0009	0.1826	144.0929
	50 000	0.9391	11.9824	0.2479	45.3899	0.0886	127.0723	0.0769	146.2598	11.6401	11.1720	2.9209	44.5208	1.0921	119.0792	0.8393	154.9394
	100 000	1.8793	11.8913	0.4780	46.7535	0.1757	127.1641	0.1298	172.2133	21.8480	11.9568	5.6096	46.5684	2.1689	120.4465	1.6707	156.3570
	200 000	3.7270	12.0104	0.9538	46.9313	0.3514	127.3876	0.2573	173.9799	43.5101	11.9709	11.1898	46.5475	4.3222	120.5079	3.2824	158.6822
	400 000	7.4420	12.0146	1.9056	46.9214	0.7015	127.4627	0.4922	181.6637	83.1721	12.5188	22.3146	46.6609	8.6148	120.8634	6.5342	159.3485

- **Matchers**: Jiang, Chen and MCC16

For the extracted minutiae databases not all parameter combinations are necessary, and the experiments are limited to the sequential and fully parallel cases to compare if the system behavior when using the extracted minutiae is the same as when using the ground-truth database. Considering the 3 matching algorithms and the 8 database sizes, this produces a total of 48 experiments. It is important to note that the increase in the number of minutiae shown in Table 3 implies a slow down in the identification process.

5.3. NIST DB4 and DB14 databases

In addition to the above mentioned experiments, NIST DB4 and DB14 databases have also been used. These databases are provided by the National Institute of Standards and Technology (NIST). They contain 2000 and 27 000 rolled fingerprint pairs, respectively, and thus the number of minutiae extracted by NIGOS *mindtct* is very high

(135.87 in DB4 and 206.90 in DB14). The aim of using DB4 and DB14 is to test if the proposed system can deal with captured databases, and also with rolled fingerprints. The used parameters for the minutiae extraction and the matching algorithms are the same as for the SFinGe extracted database.

6. Experimental study

This section describes the results of the performed experiments. Sections 6.1, 6.2 and 6.3 present the results for SFinGe ground-truth, SFinGe extracted and NIST databases, respectively.

6.1. Speedup with SFinGe ground-truth minutiae

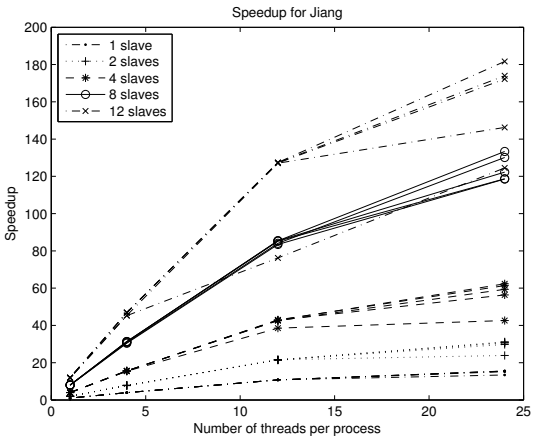
The obtained results are presented in Tables 4 and 5. Note that the experiments with the 400 000 fingerprints database combined with the MCC16 algorithm could not be performed within a single machine because the preprocessed database is bigger than the whole RAM space in the used computers (64GB), and thus no speedups can be calculated. The sequential tests could be run using virtual memory, but the performance loss would be dramatic and the speedup when using several machines would be superlinear, as described in Section 3.3. This is a very clear case of a problem that cannot be solved in a sequential manner, but can be successfully tackled using a distributed approach.

In the rest of the results, the decrease in the execution time and the corresponding increase in the speedup as the amounts of threads and processes are augmented can be seen. It is also clear that the speedup is generally almost linear with the total number of threads that perform the distributed search.

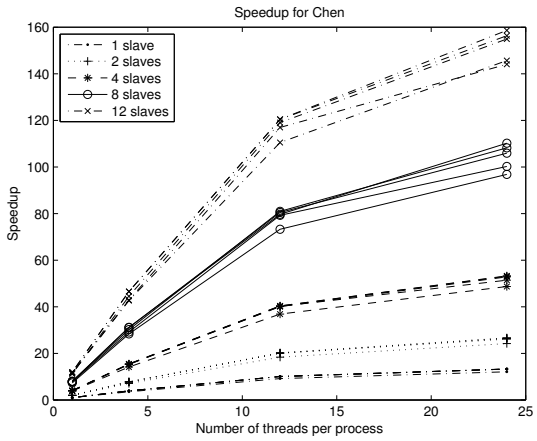
However, there are some exceptions to this statement:

- When the number of threads per computing node is 24, the performance gain is not proportional to this number of threads, but lower. Nevertheless, this behavior is normal because each node in the used cluster only has 12 cores. The Intel Hyperthreading technology is able to handle two threads in each core, but its performance is not as high as when these threads are executed in parallel within different cores, and strongly depends on the specific instructions executed by the threads.

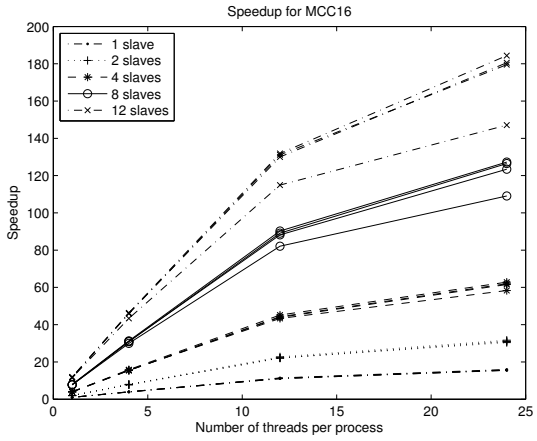
- When the database size is small and the computing resources are high, the performance is not optimal. This behavior is especially clear with the Jiang algorithm, which is the fastest method tested in this paper. For example, the execution time for 10 000 fingerprints is lower than for 1000 fingerprints when the full cluster (12 nodes and 24 threads) is used. This is due to the ratio between processing and communication times (R_{pc}), as explained in Section 3.3. With small databases and big resources, the processing time t_{pr} is much smaller than the communication time t_c and thus no gain is obtained. Furthermore, t_c is increased due to the synchronization between threads and processes. The response time of the overall system depends on the response time of the slowest thread; when the database is small, the chunks assigned to each thread are very small and the difference are high. This problem disappears as the database grows, and it explains why bigger databases can be explored faster than smaller ones when large resources are employed.



(a) Jiang



(b) Chen



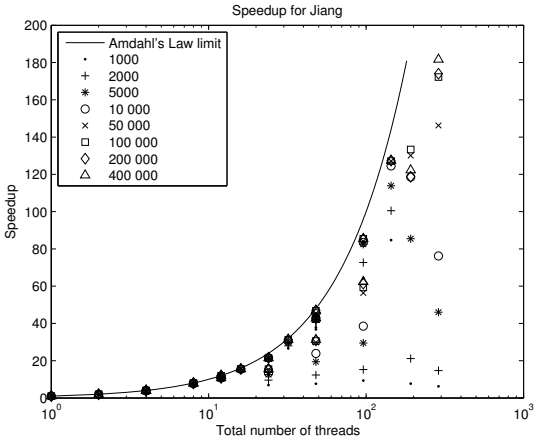
(c) MCC16

Figure 7: Speedup when varying the number of slave processes and threads, for databases with 10 000 or more fingerprints

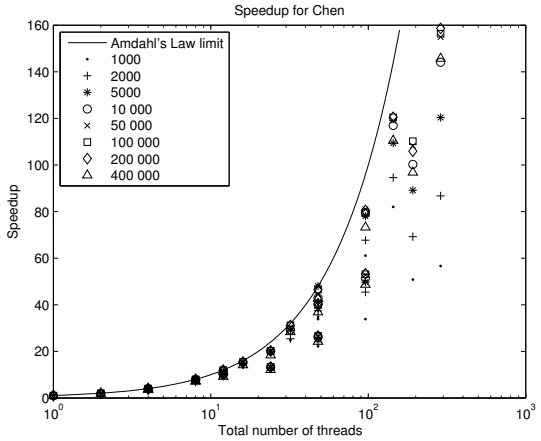
Since we are mainly concerned with large databases, this is no issue. Anyway, it can be easily solved by using a specific configuration for databases with a size smaller than a given threshold.

Additionally, Fig. 7 shows the speedup when the number of threads and processes is varied, with one line per database size and number of slaves. For the sake of readability and to avoid the irregular behavior described in the preceding paragraph, only databases from 10 000 fingerprints onwards have been drawn in these plots.

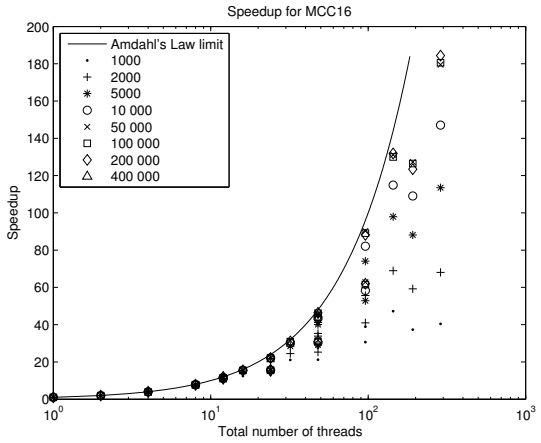
These figures clearly state that when there are more threads than physical cores the speedup does not increase linearly. It can also be seen that the lines corresponding to different database sizes are grouped. Thus, the database size does not affect the speedup when it is reasonably large. This result, along with the fact that the flexible proposed system allows the identification in arbitrarily large databases, ensures full scalability regarding the database size.



(a) Jiang



(b) Chen



(c) MCC16

Figure 8: Speedup when varying the database size

In other words, if the database size is doubled, the identification time can be kept constant by simply doubling the computing resources.

Fig. 8 shows the speedup as a function of the total number of threads that are performing the parallel search. The theoretical limit imposed by the Amdahl's Law is also displayed, and it can be seen that the results are close to the line. Moreover, the larger the database size is, the closer to the line the results are, showing that the system scalability increases along with the database size. This is the best possible situation, as it ensures a maximum scalability and performance when it is most needed. As before, there are exceptions where the speedup is much lower than the limit, when the database size is small and when the number of used threads is higher than the number of physical computing cores. The right side of the plots in Fig. 8 shows this behavior very clearly: then the full cluster is used, the speedup increases when the database grows.

Table 6: Sequential (t_s) and parallel (t_p) execution times and speedup with the SFinGe extracted and NIST databases

	Jiang			**Chen**			**MCC16**		
DB size	***t_s (s)***	***t_p (s)***	***Speedup***	***t_s (s)***	***t_p (s)***	***Speedup***	***t_s (s)***	***t_p (s)***	***Speedup***
1000	0.4735	0.0253	18.7070	5.9325	0.0661	89.6952	10.4988	0.2023	51.9038
2000	0.9330	0.0255	36.5825	11.4148	0.1004	113.6886	19.7435	0.2519	78.3921
5000	2.3099	0.0325	71.0363	28.6567	0.2061	139.0185	47.5580	0.3912	121.5540
10 000	4.6722	0.0446	104.8213	56.6954	0.3477	163.0649	93.2347	0.6227	149.7332
50 000	22.9674	0.1475	155.6925	283.3022	1.6784	168.7886	459.9514	2.5605	179.6316
100 000	46.3680	0.2734	169.6040	569.8500	3.3252	171.3708	922.1267	4.9613	185.8622
200 000	92.3980	0.5288	174.7348	1144.3038	6.6053	173.2413	1851.2036	9.7530	189.8089
400 000	183.2521	1.0368	176.7501	2303.3817	13.2144	174.3086	–	19.3620	–
DB4	7.7601	0.0795	97.5911	81.7127	0.5893	138.6534	192.5607	1.6131	119.3743
DB14	307.4337	2.0310	151.3740	2454.5287	10.4387	235.1381	6460.3050	20.6486	312.8684

6.2. SFinGe databases: extracted minutiae

Once the speedup behavior when changing the computing resources has been studied, more experiments have been performed in order to validate the results with more realistic databases. For this purpose, we have compared the sequential execution times with the times obtained when using the best configuration (12 nodes and 24 threads per node). Table 6 presents the sequential and parallel times, along with their quotient (speedup).

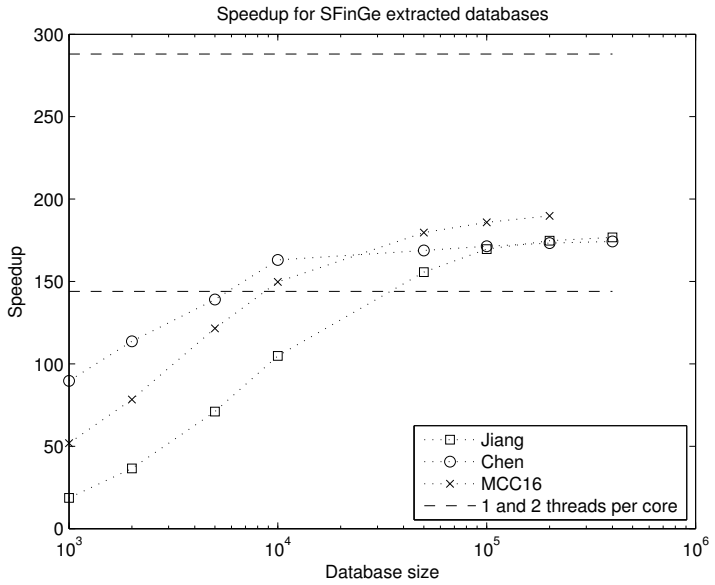


Figure 9: Speedup with the SFinGe extracted database. The dashed lines represent theoretical maximum speedups considering one or two threads per core.

It becomes clear that the good speedups obtained with the ground-truth database are also obtained with extracted

minutiae. This is due to the flexibility of the database partitioning scheme, which distributes the database statically among nodes and dynamically among threads. Another fact that can be seen in the table is that the execution times are higher than with the ground-truth minutiae. This is a consequence of the higher number of obtained minutiae when using NIGOS *mindtct*.

Fig. 9 shows the obtained speedups depending on the database size (note the logarithmic scale on the horizontal axis). It can be seen that the larger the database, the higher the speedup. It is due to the same reasons explained in the preceding section. Thus, the system behavior remains the same even when the database is changed. The plot also shows how the obtained speedup when using two threads per core is far from the theoretical maximum with the Intel Hyperthreading technology, but it is also considerably higher than the maximum of 144 (12 nodes with 12 cores) that would be attainable if this technology were not implemented in the microprocessors. This proves again that we are in an optimal case for the application of a distributed system and that the proposed system has been optimally designed and implemented.

6.3. NIST DB4 and DB14 databases

Finally, the NIST DB4 and DB14 databases have been used to test the proposed system in the same conditions as the SFinGe extracted fingerprints. The results are presented in Table 6 and Fig. 10.

Again, the speedup values are similar to those obtained with the SFinGe ground-truth database, proving that the proposed system is database-independent and can achieve very good results both with plain and rolled fingerprints, whose matching times are totally different.

The plot also shows that the speedups are higher when the search is performed in DB14, which is by far the biggest database. This result is in the same line as those obtained with the SFinGe databases, where bigger databases allow better scalability. If the figure is compared with Fig. 9, it can be seen that both NIST databases reach a better speedup than SFinGe databases of the same size. This is due to the higher number of minutiae of the rolled fingerprints: the matching process is more computationally complex, and thus HPC is able to improve the time results even further because the impact of the sequential preprocessing is reduced.

The different matching algorithms show the same behavior with the NIST and SFinGe databases, as it can be seen when comparing Fig. 9 and 10: the Jiang algorithm has less speedup because its processing workload is very small, and thus the communication time has a bigger impact on the overall time. On the other extreme, the most computationally expensive algorithm (MCC) obtains a superlinear speedup when inserted in the proposed framework, although in theory this situation should not be possible. In this case, as mentioned in Section 3.3, a higher number of computers also means more main memory and more caches. As the database chunks explored by each node are also smaller, they can fit more easily in the cache memory and thus can be explored even faster.

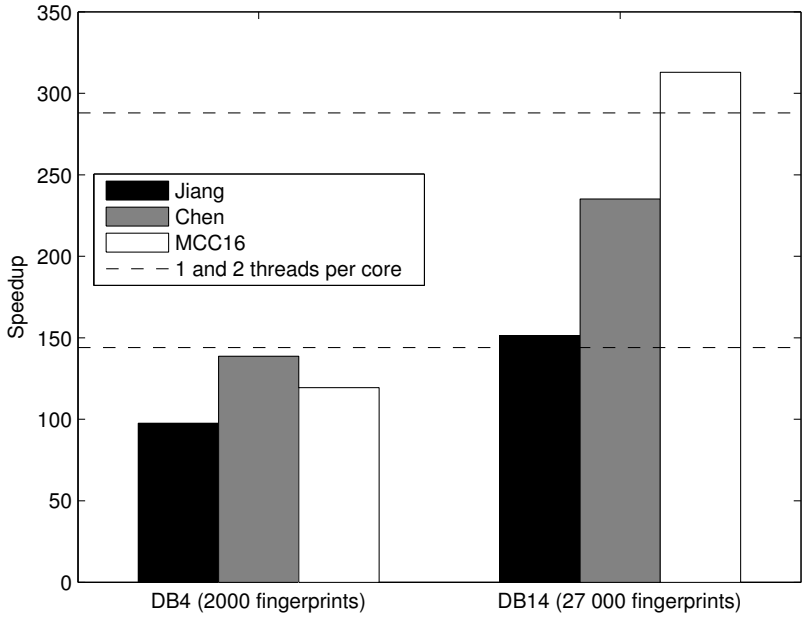


Figure 10: Speedup with the NIST databases.

7. Conclusions

In this paper, we have introduced a novel two-level parallelized automatic fingerprint identification system. The proposed framework combines process-level and thread-level parallelism in order to obtain a maximum speedup for any kind of underlying hardware architecture from monocore processors to large hybrid clusters. It also abstracts the fingerprint matching algorithm, in such a way that the inclusion of a new algorithm is straightforward and does not affect neither the algorithm nor the global framework.

In order to verify the capabilities of the system, we have used the SFinGe software [20, 2] to generate a database of 400 000 fingerprints that has been used for identification in a set of experiments on a hybrid cluster, ranging from sequential to massively parallel runs. In a search for more realistic fingerprints, we have applied the NIGOS *mindtct* minutiae extractor on the database and performed more experiments. Finally, another set of experiments has been executed using two large real-world databases from the NIST. All these experiments have been run with three well-known fingerprint matching algorithms [43, 28, 6].

After detailing the obtained results, we can conclude that the proposed framework fulfills the expectations. It has a linear scalability regarding to the fingerprint database, as well as an optimal adaptability to the underlying hardware. In theory, this allows the identification in databases of arbitrary size as long as there is enough computing power. In practice, the identification time can be kept constant against the database growth just by augmenting the computing resources in the same proportion. The framework has also proven to maintain its good behavior independently of the

underlying matching algorithms and fingerprint features.

8. Acknowledgements

This work was supported by the research projects CAB(CDTI), TIN2011-28488 and TIN2009-14575. D. Peralta holds an FPU scholarship from the Spanish Ministry of Education and Science (FPU12/04902).

References

[1] A. K. Jain, R. M. Bolle, S. Pankanti, Biometrics: Personal Identification in Networked Society, Springer, 2005.

[2] D. Maltoni, D. Maio, A. Jain, S. Prabhakar, Handbook of fingerprint recognition, Springer-Verlag New York Inc, 2009.

[3] A. Jain, S. Prabhakar, L. Hong, S. Pankanti, Filterbank-based fingerprint matching, IEEE Trans. Image Process. 9 (5) (2000) 846–859.

[4] F. Liu, Q. Zhao, D. Zhang, A novel hierarchical fingerprint matching approach, Pattern Recognition 44 (8) (2011) 1604–1613.

[5] D. Maio, D. Maltoni, Direct gray-scale minutiae detection in fingerprints, IEEE Trans. Pattern Analysis and Machine Intelligence 19 (1) (1997) 27–40.

[6] R. Cappelli, M. Ferrara, D. Maltoni, Minutia cylinder-code: A new representation and matching technique for fingerprint recognition, IEEE Trans. Pattern Analysis and Machine Intelligence 32 (12) (2010) 2128–2141.

[7] N. Ratha, R. Bolle, Automatic Fingerprint Recognition Systems, Springer, New York, 2004.

[8] S. Pankanti, S. Prabhakar, A. Jain, On the individuality of fingerprints, IEEE Trans. Pattern Analysis and Machine Intelligence 24 (8) (2002) 1010–1025.

[9] A. Jain, L. Hong, R. Bolle, On-line fingerprint verification, IEEE Trans. Pattern Analysis and Machine Intelligence 19 (4) (1997) 302–314.

[10] A. Jain, L. Hong, S. Pankanti, R. Bolle, An identity-authentication system using fingerprints, Proc. IEEE 85 (9) (1997) 1365–1388.

[11] H. Stone, High-performance computer architecture, Addison-Wesley Longman Publishing Co., Inc., 1992.

[12] R. Armstrong, D. Gannon, A. Geist, K. Keahey, S. Kohn, L. McInnes, S. Parker, B. Smolinski, Toward a common component architecture for high-performance scientific computing, in: Proc. 8th IEEE Int. Symp. High Performance Distributed Computing, 1999, pp. 115–124.

[13] M. S. Seidenberg, J. L. McClelland, A distributed, developmental model of word recognition and naming, Psychological review 96 (4) (1989) 523–568.

[14] A. Datta, S. Soundaralakshmi, Fast parallel algorithm for distance transform, IEEE Trans. Syst., Man, Cybern. A, Syst., Humans 33 (4) (2003) 429–434.

[15] A. Stamatakis, M. Ott, Exploiting fine-grained parallelism in the phylogenetic likelihood function with mpi, pthreads, and openmp: A performance study, in: Proc. 3rd Int. Conf. Pattern Recognition in Bioinformatics, Springer-Verlag, 2008, pp. 424–435.

[16] M. Gong, Y. Zhang, Y. H. Yang, Near-real-time stereo matching with slanted surface modeling and sub-pixel accuracy, Pattern Recognition 44 (10.11) (2011) 2701–2710.

[17] T. Y. Ho, P. M. Lam, C. S. Leung, Parallelization of cellular neural networks on GPU, Pattern Recognition 41 (8) (2008) 2684–2692.

[18] G. Danese, M. Giachero, F. Leporati, N. Nazzicari, An embedded multi-core biometric identification system, Microprocessors and Microsystems 35 (5) (2011) 510–521.

[19] M. Hulea, A. AÅtilean, T. LeÅ£ia, R. Miron, S. Folea, Fingerprint recognition distributed system, in: Proc. 16th IEEE Int. Conf. Automation, Quality and Testing, Robotics, Vol. 3, 2008, pp. 423–428.

[20] R. Cappelli, D. Maio, D. Maltoni, Synthetic fingerprint-database generation, in: Proc. 16th Int. Conf. Pattern Recognition, Vol. 3, 2002, pp. 744–747.

[21] C. I. Watson, M. D. Garris, E. Tabassi, C. L. Wilson, R. M. Mccabe, S. Janet, K. Ko, User's guide to NIST biometric image software (NBIS), Tech. rep., NIST (2010).

[22] C. Watson, C. Wilson, NIST special database 4, Tech. rep., NIST (1992).

[23] C. Watson, NIST special database 14, Tech. rep., NIST (1993).

[24] G. K. Manacher, Production and stabilization of real-time task schedules, J. ACM 14 (3) (1967) 439–465. doi:10.1145/321406.321408.

[25] F. B. of Investigation (Ed.), The Science of Fingerprints: Classification and Uses, U.S. Government Printing Office, 1984.

[26] H. Lee, R. Gaensslen, Advances in fingerprint technology, CRC, 2001.

[27] Q. Fang, N. Bhattacharjee, Incremental fingerprint recognition model for distributed authentication, in: Proc. Int. Conf. Security and Management, 2008, pp. 41–47.

[28] X. Chen, J. Tian, X. Yang, A new algorithm for distorted fingerprints matching based on normalized fuzzy similarity measure, IEEE Trans. Image Process. 15 (3) (2006) 767–776.

[29] C. Gottschlich, C. Schnlieb, Oriented diffusion filtering for enhancing low-quality fingerprint images, IET Biometrics 1 (2) (2012) 105–113.

[30] J. Liu-Jimenez, R. Sanchez-Reillo, L. Mengibar-Pozo, O. Miguel-Hurtado, Optimisation of biometric id tokens by using hardware/software co-design, IET Biometrics 1 (3) (2012) 168–177.

[31] P. Sutthiwichaiporn, V. Areekul, Adaptive boosted spectral filtering for progressive fingerprint enhancement, Pattern Recognition 46 (9) (2013) 2465–2486.
URL http://www.sciencedirect.com/science/article/pii/S0031320313000782

[32] N. K. Ratha, S. Chen, A. K. Jain, Adaptive flow orientation-based feature extraction in fingerprint images, Pattern Recognition 28 (11) (1995) 1657–1672.

[33] L. Coetzee, E. C. Botha, Fingerprint recognition in low quality images, Pattern Recognition 26 (10) (1993) 1441–1460.

[34] X. Jiang, W. Y. Yau, W. Ser, Minutiae extraction by adaptive tracing the gray level ridge of the fingerprint image, in: IEEE Int. Conf. Image Process., Vol. 2, 1999, pp. 852–856.

[35] X. Jiang, W. Yau, W. Ser, Detecting the fingerprint minutiae by adaptive tracing the gray-level ridge, Pattern Recognition 34 (5) (2001) 999–1013.

[36] J. Liu, Z. Huang, K. L. Chan, Direct minutiae extraction from gray-level fingerprint image by relationship examination, in: IEEE Int. Conf. Image Processing, Vol. 2, 2000, pp. 427–430.

[37] J. Chang, K. Fan, Fingerprint ridge allocation in direct gray-scale domain, Pattern Recognition 34 (10) (2001) 1907–1925.

[38] M. Fons, F. Fons, N. Canyellas, E. Cant, M. Lpez, Hardware-software co-design of an automatic fingerprint acquisition system, in: IEEE Int. Symp. Industrial Electronics, Vol. III, 2005, pp. 1123–1128.

[39] M. Leung, W. Engeler, P. Frank, Fingerprint image processing using neural networks, in: Conf. Comput. and Commun. Syst., IEEE, 1990, pp. 582–586.

[40] K. Nilsson, J. Bigun, Using linear symmetry features as a pre-processing step for fingerprint images, in: Audio and Video-Based Biometric Person Authentication, Springer, 2001, pp. 247–252.

[41] S. H. Lee, H. B. Chae, S. Y. Yi, E. S. Kim, Optical fingerprint identification based on binary phase extraction joint transform correlator, in: Proc. Int. Soc. for Optical Eng., Vol. 2752, 1996, pp. 224–232.

[42] B. V. K. V. Kumar, M. Savvides, C. Xie, K. Venkataramani, J. Thornton, A. Mahalanobis, Biometric verification with correlation filters, Applied Optics 43 (2) (2004) 391–402.

[43] X. Jiang, W. Yau, Fingerprint minutiae matching based on the local and global structures, in: Proc. 15th Int. Conf. Pattern Recognition, Vol. 2, IEEE, 2000, pp. 1038–1041.

[44] K. Cao, X. Yang, X. Chen, Y. Zang, J. Liang, J. Tian, A novel ant colony optimization algorithm for large-distorted fingerprint matching, Pattern Recognition 45 (1) (2012) 151–161. doi:10.1016/j.patcog.2011.04.016.
URL http://www.sciencedirect.com/science/article/pii/S0031320311001750

[45] N. Ratha, K. Karu, S. Chen, A. Jain, A real-time matching system for large fingerprint databases, IEEE Trans. Pattern Analysis and Machine Intelligence 18 (8) (1996) 799–813.

[46] S. Chang, F. Cheng, W. Hsu, G. Wu, Fast algorithm for point pattern matching: invariant to translations, rotations and scale changes, Pattern Recognition 30 (2) (1997) 311–320.

[47] A. Hrechak, J. McHugh, Automated fingerprint recognition using structural matching, Pattern Recognition 23 (8) (1990) 893–904.

[48] D. Koufaty, D. Marr, Hyperthreading technology in the netburst microarchitecture, IEEE Micro 23 (2) (2003) 56–65. doi:10.1109/MM.2003.1196115.

[49] G. M. Amdahl, Validity of the single processor approach to achieving large scale computing capabilities, in: Proc. Spring Joint Comput. Conf., ACM, 1967, pp. 483–485.

[50] G. Indrawan, B. Sitohang, S. Akbar, Parallel processing for fingerprint feature extraction, in: Int. Conf. Electr. Eng. and Informat., 2011, pp. 1–6. doi:10.1109/ICEEI.2011.6021606.

[51] R. F. Miron, T. S. Letia, M. Hulea, Two server topologies for a distributed fingerprint-based recognition system, in: 15th Int. Conf. System Theory, Control and Computing, 2011, pp. 1–6.

[52] K. Beghdad Bey, Z. Guessoum, A. Mokhtari, F. Benhammadi, Agent based approach for distribution of fingerprint matching in a metacomputing environment, in: Proc. 8th Int. Conf. New Technologies in Distributed Systems, 2008, pp. 1–7.

[53] K. Nagaty, E. Hattab, An approach to a fingerprints multi-agent parallel matching system, in: IEEE Int. Conf. Syst., Man and Cybern., Vol. 5, 2004, pp. 4750–4756. doi:10.1109/ICSMC.2004.1401282.

Daniel Peralta received the M.Sc. degree in Computer Science in 2011 from the University of Granada, Granada, Spain. He is currently a Ph.D. student in the Department of Computer Science and Artificial Intelligence, University of Granada. His research interests include data mining, biometrics and parallel and distributed computing.

Isaac Triguero received the M.Sc. degree in Computer Science from the University of Granada, Granada, Spain, in 2009. He is currently a Ph.D. student in the Department of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain. His research interests include data mining, data reduction, biometrics, evolutionary algorithms and semi-supervised learning.

Raul Sanchez-Reillo graduated as Telecommunication Engineer by the Polytechnic University of Madrid, obtaining his PhD at the same University in 2000, based on Biometric Authentication in Smart Cards. From 1994 he has been researching at the University Group for Identification Technologies, managing the group since 2000. He is currently Associate Professor at Carlos III University of Madrid. He is also member of ISO/IEC JTC1 SC17, SC27 and SC37 standardization bodies, holding some management position in national and international standardization bodies. His interests in R&D cover all Personal Identification Technologies, including Smart Cards, Biometrics and Secure Authentication Systems. He is founder of IDTestingLab, an evaluation facility for identification products.

Francisco Herrera received his M.Sc. in Mathematics in 1988 and Ph.D. in Mathematics in 1991, both from the University of Granada, Spain. He is currently a Professor in the Department of Computer Science and Artificial Intelligence at the University of Granada. He has published more than 230 papers in international journals. He is coauthor of the book "Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases" (World Scientific, 2001). He currently acts as Editor in Chief of the international journal "Progress in Artificial Intelligence" (Springer). He acts as area editor of the International Journal of Computational Intelligence Systems and associated editor of the journals: IEEE Transactions on Fuzzy Systems, Information Sciences, Knowledge and Information Systems, Advances in Fuzzy Systems, and International Journal of Applied Metaheuristics Computing; and he serves as member of several journal editorial boards, among others: Fuzzy Sets and Systems, Applied Intelligence, Information Fusion, Evolutionary Intelligence, International Journal of Hybrid Intelligent Systems, Memetic Computation, and Swarm and Evolutionary Computation. He received the following honors and awards: ECCAI Fellow 2009, 2010 Spanish National Award on Computer Science ARITMEL to the "Spanish Engineer on Computer Science", International Cajastur "Mamdani" Prize for Soft Computing (Fourth Edition, 2010), IEEE Transactions on Fuzzy System Outstanding 2008 Paper Award (bestowed in 2011), and 2011 Lotfi A. Zadeh Prize Best paper Award of the International Fuzzy Systems Association. His current research interests include computing with words and decision making, bibliometrics, data mining, biometrics, data preparation, instance selection, fuzzy rule based systems, genetic fuzzy systems, knowledge extraction based on evolutionary algorithms, memetic algorithms and genetic algorithms.

José Manuel Benítez is an Associate Professor at the Department Computer Science and Artificial Intelligence (`http://decsai.ugr.es`), Universidad de Granada, Granada, Spain. Dr. Benítez holds an M.S. Degree and a Ph. D. in Computer Science, both from the Universidad de Granada. He is a member of the Research Group "Soft Computing and Intelligent Information Systems" (SCI2S, `http://sci2s.ugr.es`) and head of the "Distributed Computational

Intelligence and Time Series" research lab (DICITS, `http://sci2s.ugr.es/DiCITS`). He is an active researcher in the Computational Intelligence field where his work covers the whole spectrum from foundations to applications in a number of engineering and scientific areas. In particular, his current fields of interest are time series analysis and modeling, distributed/parallel computational intelligence, high performance computing, cloud computing, data mining, biometrics, and statistical learning theory. He is a member of a number of scientific associations, including IEEE, IEEE Computational Intelligence Society, and EUSFLAT.

4 DPD-DFF: A Dual Phase Distributed Scheme with Double Fingerprint Fusion for Fast and Accurate Identification in Large Databases

- D. Peralta, I. Triguero, S. García, F. Herrera, J.M. Benítez. DPD-DFF: A Dual Phase Distributed Scheme with Double Fingerprint Fusion for Fast and Accurate Identification in Large Databases. Information Fusion 32 (2016) 40–51. doi: 10.1016/j.inffus.2016.03.002

  - Status: **Published**.
  - Impact Factor (JCR 2015): 4.353
  - Subject Category: Computer Science, Artificial Intelligence. Ranking 9 / 130 (**Q1**).
  - Subject Category: Computer Science, Theory & Methods. Ranking 4 / 105 (**Q1**).

DPD-DFF: A Dual Phase Distributed Scheme with Double Fingerprint Fusion for Fast and Accurate Identification in Large Databases

Daniel Peralta[a,*], Isaac Triguero[b,c], Salvador García[a,d], Francisco Herrera[a], Jose M. Benitez[a]

[a]*Department of Computer Science and Artificial Intelligence, CITIC-UGR (Research Center on Information and Communications Technology), University of Granada, 18071 Granada, Spain*
[b]*Department of Respiratory Medicine, Ghent University, 9000 Gent, Belgium*
[c]*VIB Inflammation Research Center, 9052 Zwijnaarde, Belgium*
[d]*Department of Information Systems, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia*

Abstract

Nowadays, many companies and institutions need fast and reliable identification systems that are able to deal with very large databases. Fingerprints are among the most used biometric traits for identification. In the current literature there are fingerprint matching algorithms that are focused on efficiency, whilst others are based on accuracy.

In this paper we propose a flexible dual phase identification method, called DPD-DFF, that combines two fingers and two matchers within a hybrid fusion scheme to obtain both fast and accurate results. Different alternatives are designed to find a trade-off between runtime and accuracy that can be further tuned with a single parameter.

The experiments show that DPD-DFF obtains very competitive results in comparison with the state-of-the-art score fusion techniques, especially when dealing with large databases or impostor fingerprints.

Keywords: Real-time identification, large databases, minutiae matching, fingerprint fusion, decision fusion, score fusion, parallel computing, biometrics

1. Introduction

Personal identification has arisen as an important issue in the last few years for many companies and institutions [1]. Identification databases grow larger every year, ranging from a few tens of people for small companies to several millions for institutions such as the police. Although there are various biometric traits that allow for identification, fingerprints are widely used because of their uniqueness and universality, among other properties [2, 3]. Fingerprint recognition can be tackled from two different perspectives: verification [4] and identification [5]. The former consists of matching two fingerprints to determine whether they belong to the same finger or not. The latter aims to identify an input fingerprint from a set of fingerprints and determine which of them matches with the input. In

*Corresponding author. Tel.: +34 958244019; fax: +34 958243317
Email addresses: dperalta@decsai.ugr.es (Daniel Peralta), Isaac.Triguero@irc.vib-UGent.be (Isaac Triguero), salvagl@decsai.ugr.es (Salvador García), herrera@decsai.ugr.es (Francisco Herrera), J.M.Benitez@decsai.ugr.es (Jose M. Benitez)

this context, an Automatic Fingerprint Identification System (AFIS) is a tool that allows us to perform identifications in fingerprint databases [3].

Fingerprints are composed of a pattern of ridges and valleys, from which diverse features can be extracted. Among these features, minutiae are widely used for fingerprint matching, mostly due to their distinctiveness [2, 6]. When two fingerprints are to be compared, the minutiae are extracted from the images, and then a matching algorithm is applied over the two minutiae sets to determine a similarity level. There are multiple proposals of minutiae-based matching algorithms in the literature [7]. Some of them are very efficient due to their simplicity [8], while others are very accurate [9]. However, these two objectives are usually not reached together because accurate algorithms tend to be complex, and therefore time-consuming. This restriction complicates the development of AFIS that are able to identify people in very large databases in a suitable time frame without precision loss.

Moreover, as the overall response time of an identification procedure is linear with respect to the size of the database, even the fastest matching algorithms may become useless when the database grows too large. Moreover, the huge number of matchings causes an accuracy loss.

Information fusion is a widely used paradigm that improves overall precision in many fields, including biometrics [10, 11, 12]. In particular, two main approaches have been proven to enhance the recognition capabilities: the use of several fingerprint images [13] and the use of several matching algorithms [14]. The information fusion can be performed at different levels:

- Feature fusion approaches merge the characteristics extracted from different fingerprint images, coming either from the same finger or different fingers [15, 16].
- Score fusion methods perform separate matchings and then sum up the scores [14, 17].
- Decision fusion methods apply the matching algorithms in a hierarchical mode over the fingerprints [11, 18].

Although these approaches increase the accuracy of the AFIS, they also slow the identification down because the processing workload is higher. In this work, we combine the ideas of multi-finger and multi-algorithm identification to improve the runtime along with the accuracy.

High Performance Computing (HPC) is an important tool to speed up the runtime of a system [19, 20], and several proposals in the literature apply it to AFIS. However, these systems focus on objectives other than precision, such as high availability [21], load balancing [22] or reduced matching times [18]. Other systems provide the ability to identify in very large databases [23, 24, 25], but their accuracy is not improved with respect to a sequential AFIS.

There are currently several systems in the world that maintain large fingerprint databases. For example, as of September 2015, India's UIDAI system [26] stores the fingerprints of around 907 million people, although so far they are only used for verification purposes, not identification. FBI IAFIS [27] (now included within Next Generation Identification, NGI) keeps the fingerprints (among other data) for around 104 million subjects, and is able to perform searches in an average time of 72 minutes.

In this paper, we propose a flexible, Dual Phase Distributed AFIS with Double Fingerprint Fusion (called DPD-DFF) that integrates two fingerprints and two matching algorithms, aiming to overcome the weaknesses of isolated approaches: high identification time and accuracy loss. To do so, the identification is split into two phases, each of which can either use a single fingerprint or fuse two of them, conforming a mixed score fusion and decision fusion process:

- In the first phase, the database is explored by a fast matching algorithm to select a candidate set. Jiang's algorithm [8] has been selected for this phase due to its high running speed [7].
- Then, the second phase applies a more accurate algorithm to identify the correct identity within this candidate set. The matcher used in this phase is Minutia Cylinder-Code (MCC) [9], which is very precise [7].

With this design, the fingerprint fusion is powerful and flexible as it is performed at two separate levels. Furthermore, this strategy has been integrated within the parallel framework proposed in [23] in order to reach full scalability for arbitrarily large databases.

This manuscript is structured as follows. First, Section 2 provides the background information on the problem at hand. Section 3 presents DPD-DFF, the approach proposed in this paper. Section 4 describes the experiments performed and their results. Finally, Section 5 details the conclusions. Complementary material to the paper including tables, plots and identification times as well as additional studies over other databases can be found at `http://sci2s.ugr.es/DPDDFF` and in the associated Technical Report [28].

2. Preliminaries

A fingerprint is a pattern of valleys and ridges located on a fingertip. Although there are several ways to perform a matching between two fingerprints, many matching algorithms use the minutiae [3, 7, 29], comparing two minutiae sets to return a similarity score. The matching is performed once for each comparison between two fingerprints. Some of the existing matching algorithms offer very good matching precision [9], and others provide a fast response with slightly diminished accuracy [8], according to the taxonomy and results presented in [7].

There are two main variants of the fingerprint recognition problem [3]. Verification [4] is a 1:1 comparison to check if two fingerprints represent the same finger. Identification [5] consists of determining which fingerprint in a database of previously captured and stored templates $T = \{T_1, T_2, ..., T_n\}$ corresponds to a given input fingerprint I. An identification algorithm compares I to every T_i and returns the identity with the best matching score as shown in Eq. 1, where $Q(I, T_i)$ is the matching function. Thus, identification is a 1:n comparison.

$$\text{Identity} = \arg\max_i Q(I, T_i) \quad i \in \{1, 2, ..., n\} \tag{1}$$

This paper is focused on identification. Section 2.1 explains the current proposals for fast and scalable identification within large databases. Then, Section 2.2 presents the previous work about fingerprint fusion to improve the identification accuracy.

2.1. Scalable fingerprint recognition in large databases

The bottleneck of an AFIS when attempting to identify within a large database is the matching algorithm. Several proposals in the literature aim to overcome this problem.

FPGA-based systems implement the matching into a Field Programmable Gate Array [18, 30], a hardware device that performs some operations very quickly, so that the overall identification time is reduced.

Other approaches reduce the penetration rate in the database by using a previous classification or indexing step [31, 32, 33, 34]. Nevertheless, in large databases this step may become the bottleneck, and the size of the subsets can become too large. Accuracy is degraded when the penetration rate is too small or the collision rate too high [33].

HPC is a common solution for reducing high execution times [19, 20]. By using q computers with c cores each to perform a parallel search, the execution time can be reduced by up to a factor of qc. Moreover, the availability of more RAM memory allows more template fingerprints to be kept in a fast access device, avoiding slow access to secondary memory. Therefore, an adequate parallel framework can constitute a suitable tool for solving the identification problem in large databases [23, 24, 25].

2.2. Fingerprint information fusion

This section introduces two of the main trends to improve the accuracy of fingerprint recognition. On the one hand, the use of several fingers [13] increases the distinctiveness of the identities and tries to avoid the difficulties posed by injured fingertips or low quality scans. The matching function for f fingerprints becomes of the form $Q(\mathcal{I}, \mathcal{T}_i)$ where $\mathcal{I} = \{I_j \mid j \in \{1, ..., f\}\}$ and $\mathcal{T}_i = \{T_{ij} \mid j \in \{1, ..., f\}\}$. This approach has been successfully applied over latent fingerprints, which are of very low quality [35].

On the other hand, the combination of several matchers [14, 36] aims to profit from their advantages, while leaving aside their weaknesses. Multi-algorithm techniques work in a similar way as multi-finger ones, so that the fused score obtained for f algorithms is $Q(I, T_i) = \mathcal{F}\left(Q_1(I, T_i), ..., Q_f(I, T_i)\right)$, where $\mathcal{F}$ is an aggregation function.

Multi-finger and multi-algorithm approaches can be categorized together according to the type of fusion they perform:

- **Feature fusion** [15, 16, 37, 38]: this approach merges all f fingerprints of an identity into a single structure, which is compared to all n template structures. This avoids the necessity of performing f matchings per identity, but requires specific matching algorithms to handle such structures, as well as an additional conversion step.
- **Score fusion** [14, 17, 36, 39, 40]: this method applies several matchings (one for each fingerprint or algorithm) and aggregates the results into a single score. Although it does not need a specific matching algorithm, the use of f fingerprints or f matchings multiplies the identification time by f.

- **Decision fusion** [10, 11, 18, 32]: can be seen as a special case of score fusion, where matching is performed hierarchically. When the f input fingerprints are compared with some f template fingerprints for a given identity, the first pair is compared first. If the resulting score meets a certain condition, the second pair is compared, and so on.

Most fusion approaches are focused on improving accuracy, without considering runtime. Therefore, they are not adapted to address the identification in large databases because the execution time is higher than it is for simpler approaches. Empirical results obtained by some of the methods mentioned above can be found in the Technical Report associated with this paper [28].

3. Dual Phase Distributed Scheme with Double Fingerprint Fusion

DPD-DFF carries out a hybrid fusion between two matching algorithms and two fingers within a flexible dual phase scheme that is implemented in a parallel HPC system. The proposal seeks to tackle large fingerprint databases with a good trade-off between two seemingly opposed objectives:

- **Accuracy**: identification accuracy must be better than it is for isolated models.
- **Efficiency and scalability**: the system should provide a real-time response. The runtime threshold depends on the specific application; it can vary between a few milliseconds and several minutes. Ideally the identification time should be lower than when using an isolated AFIS.

First, a fast matcher explores the whole database and extracts a set of candidate identities C. Then, an accurate matcher compares the input fingerprints with the templates in C. This corresponds to a decision fusion identification method as described in Section 2.2, in which the separate use of both algorithms avoids the necessity of transforming their respective outputs to a common domain and the consequent loss of precision, as it does for traditional multi-algorithm score fusion approaches. The overall identification procedure is applied as follows:

1. **Fast phase**: according to the results obtained in [7], Jiang's algorithm [8] has been selected to perform this first identification phase, because of its speed and its appropriate accuracy. Two different criteria may be used to compose the set C:
   - **Rank**: given a rank r, select the r identities that provide the best scores. Thus, C has a fixed size $|C| = r$.
   - **Threshold**: all templates $\mathcal{T}_i$ whose score is higher than a fixed threshold ϕ when compared to the input fingerprint $\mathcal{I}$ are included in C. Therefore, the size of C is not previously known and will likely be different for each input fingerprint pair. The set can be described as $C = \{\mathcal{T}_i \mid Q_{Jiang}(\mathcal{I}, \mathcal{T}_i) \geq \phi\}$.
2. **Accurate phase**: the MCC algorithm [9] has been chosen for this phase due to its high accuracy. After comparing the input fingerprints with the templates in C, the identity with the best score is returned as the found match, as shown in Eq. 2.

$$\text{Identity} = \arg\max_i \{Q_{MCC}(\mathcal{I}, \mathcal{T}_i) \mid \mathcal{T}_i \in C\} \tag{2}$$

$$T_{AB} = \{\mathcal{T}_i \mid \mathcal{T}_i = \{T_{iA}, T_{iB}\}\ ,\ i \in \{1, 2, ..., n\}\} \tag{3}$$

In addition to this multi-algorithm scheme, we also use two different fingers (let them be finger A and finger B) per identity to even further improve identification accuracy. Two template fingerprints per person are stored, constituting a database T_{AB} with n fingerprints pairs as described in Eq. 3. An identification requires an input set of two fingerprints $\mathcal{I} = \{I_A, I_B\}$. According to this structure, each of the previously described identification phases can be carried out using either a single fingerprint or both fingerprints:

- **Single finger**: a single fingerprint of each identity is compared, as shown in Eq. 4. This alternative is proposed in a search for speed, minimizing the computation load.

$$\begin{aligned} Q_{\text{Jiang}}(\mathcal{I}, \mathcal{T}_i) &= Q_{\text{Jiang}}(I_A, T_{iA}) \quad \text{(fast phase)} \\ Q_{\text{MCC}}(\mathcal{I}, \mathcal{T}_i) &= Q_{\text{MCC}}(I_B, T_{iB}) \quad \text{(accurate phase)} \end{aligned} \tag{4}$$

- **Double finger**: both fingerprints are used for the comparison. This constitutes in itself a fusion method. Thus, a score-based fusion has been implemented, using the average as the aggregation function (Eq. 5), as recommended by the results of [17]. This approach is obviously slower than using a single finger, but it is much more accurate.

$$\begin{aligned} Q_{\text{Jiang}}(\mathcal{I}, \mathcal{T}_i) &= \frac{Q_{\text{Jiang}}(I_A, T_{iA}) + Q_{\text{Jiang}}(I_B, T_{iB})}{2} \quad \text{(fast phase)} \\ Q_{\text{MCC}}(\mathcal{I}, \mathcal{T}_i) &= \frac{Q_{\text{MCC}}(I_A, T_{iA}) + Q_{\text{MCC}}(I_B, T_{iB})}{2} \quad \text{(accurate phase)} \end{aligned} \tag{5}$$

Table 1: Names of the eight considered variants of DPD-DFF

Fingers used			Candidate set criterion		
First phase	Second phase	Prefix	Rank (*R)	Threshold (*T)	Objective
Single (A)	Single (B)	SS*	**SSR**	**SST**	High speed
Single (A)	Double (A,B)	SD*	**SDR**	**SDT**	Trade-off
Double (A,B)	Single (B)	DS*	**DSR**	**DST**	Trade-off
Double (A,B)	Double (A,B)	DD*	**DDR**	**DDT**	High accuracy

The described method performs a hybrid fusion that uses both score and decision fusion to combine two fingers and two algorithms. The overall workflow is depicted in Figure 1. A pseudocode of the identification procedure

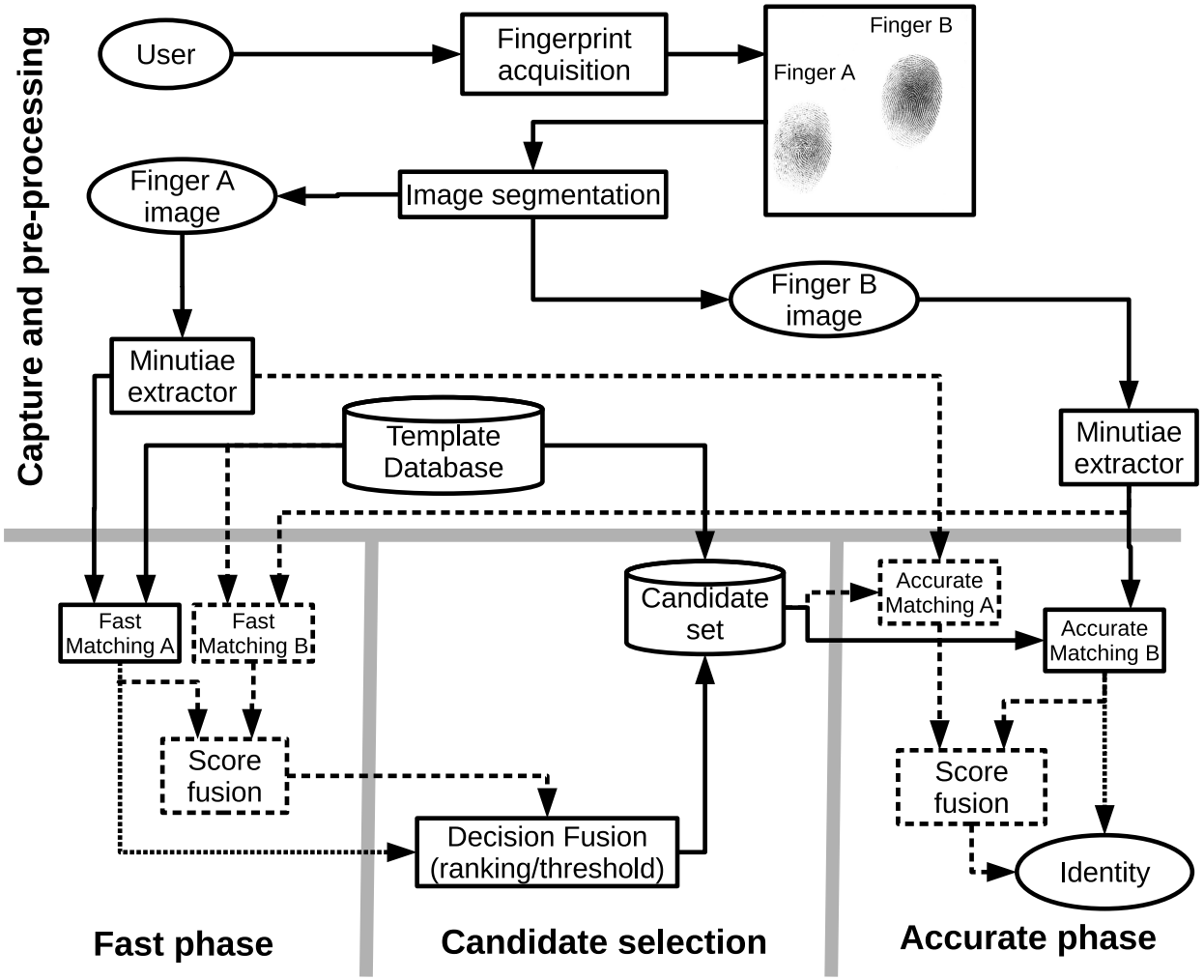


Figure 1: Workflow of DPD-DFF. Dashed lines are pathways that correspond to double finger variants. Dotted lines correspond to single finger variants. Continuous lines are pathways that are always taken.

is shown in Algorithm 1. Out of this design, we take eight variants of the algorithm into consideration, which are denoted with three letters as shown in Table 1. The first two letters represent the fingers that are taken for the fast and accurate phases respectively (S for single and D for double). The last letter stands for the criterion to build the candidate set (R for rank, T for threshold).

Note that the variants that use both fingers within a same phase (SD*, DS* and DD*) will eventually apply both matching algorithms over the fingerprints in C. This can enhance the identification accuracy, due to the synergy between two algorithms that perform the matching differently [14].

Along with the algorithm variant, the choice of the parameters to build the candidate set (r or θ) is critical, as it will determine its size $|C|$, which in turn determines the trade-off between speed and accuracy: a small candidate set relies more on the fast phase and provides faster results (though less accurate), whilst a large candidate set leads to

Input: T, $\mathcal{I}$, crit, r, θ

$C \leftarrow \emptyset$

`// Fast phase`

foreach $\mathcal{T}_i \in T$ **do**

  $q \leftarrow Q_{Jiang}(\mathcal{I}, \mathcal{T}_i)$

  if *crit* == *"Ranking"* **then**

    if $|C| < r$ **then** C.append($\mathcal{T}_i$);

    else

      $min_C = \arg\min_i \{Q_{Jiang}(\mathcal{I}, \mathcal{T}_i) \mid \mathcal{T}_i \in C\}$

      if $Q_{Jiang}(\mathcal{I}, \mathcal{T}_{min_C}) < q$ **then**

        C.remove($\mathcal{T}_{min_C}$)

        C.append($\mathcal{T}_i$)

      end

    end

  else if *crit* == *"Threshold"* ***and*** $Q_{Jiang}(\mathcal{I}, \mathcal{T}_i) \geq \theta$ **then**

    C.append($\mathcal{T}_i$)

  end

end

`// Accurate phase`

$max_q \leftarrow 0$

$identity \leftarrow null$

foreach $\mathcal{T}_i \in C$ **do**

  if $Q_{MCC}(\mathcal{I}, \mathcal{T}_i) > max_q$ **then**

    $max_q \leftarrow Q_{MCC}(\mathcal{I}, \mathcal{T}_i)$

    $identity \leftarrow \mathcal{T}_i$

  end

end

return *identity*

Algorithm 1: DPD-DFF algorithm

more accurate results but needs longer runtime.

Despite the separation between fast and accurate phases, if the structure proposed so far is implemented in a sequential manner the scalability problem will eventually appear for arbitrarily large databases. To achieve high scalability, DPD-DFF has been developed within the two-level parallel framework proposed in [23], as described in the Technical Report [28]. Hence, the scheme can be efficiently executed in a cluster of computers.

4. Experiments and results

This section describes the experiments performed over several fingerprint databases: a large database of SFinGe-generated fingerprints (Section 4.2), a database captured by the authors (Section 4.3), the well-known NIST-DB14 (Section 4.4) and several other public databases (Section 4.5). Section 4.1 describes the hardware and software used for these experiments.

The number of True Positives (TP), False Positives (FP) and False Negatives (FN) are used as accuracy measures, along with the True Positive Rate (TPR). The average identification time is denoted by t_{avg} and measured in seconds in all cases. For the threshold variants, the average candidate set size $|C|_{avg}$ is also given. The plots include the accuracy and identification time of three reference AFIS: an isolated one that uses a single finger and a single matcher (as described in [8, 9]), and two score fusion approaches, one multi-finger (as described in [17, 41]) and one multi-algorithm (as described in [11, 17, 36]).

Note that for a fair comparison, both DPD-DFF and the reference AFIS were implemented in the framework proposed in [23] and executed over the same hardware. It is out of the scope of this paper to analyze the performance of the parallel procedure; the study is focused on the behavior of the proposed hybrid fusion method.

Additional details and results (such as tables, figures, database statistics, identification times, hardware configuration and results with more databases) are available in the associated Technical Report [28] and at `http://sci2s.ugr.es/DPDDFF`.

4.1. Hardware and software environment

The experiments carried out for this paper have been executed in a cluster of 12 nodes, each of them with two Intel Xeon E5-2620 processors (6 cores each). The executions were performed with 12 slave processes (one in each node), each of them composed of 24 threads. Note that a smaller subset of nodes was used for the databases of small size.

All fingerprint minutiae were extracted using the NIGOS *mindtct* software [42], whose parameters are detailed in Table 2. The authors have written their own implementation of the underlying matching algorithms [8, 9], with the sole aid of their respective original publications. The parameters used for these algorithms are also presented in the table.

To ensure a fair comparison, the same parameters were used for all the tested databases, so as to avoid any kind of over-fitting of the results. Even though this may produce low accuracy values for some of the databases, this setup aims to assess the robustness of the proposed method in different use cases.

4.2. SFinGe database

This section describes the experiments performed over a database of 50 000 fingerprint pairs, synthetically generated with the SFinGe software [3, 43]. First, Section 4.2.1 details the used fingerprint database. Then, Section 4.2.2 describes the experiments carried out and the obtained results.

Table 2: Parameters for the methods used in the experimentation

Algorithm	Parameters	Reference
Jiang	$w_d = 1, w_\theta = 0.3\frac{180}{\pi}, w_\phi = 0.3\frac{180}{\pi}$ $w_n = 0, w_t = 0$, Consolidation step iterations = 5 Minutiae neighborhood size = 2 $BG_1 = 8, BG_2 = \frac{\pi}{6}, BG_3 = \frac{\pi}{6}$	[8]
MCC	$R = 70, N_s = 8, N_d = 6, \sigma_s = \frac{28}{3}, \sigma_d = \frac{2\pi}{9}$ $\mu_\Psi = 0.01, \tau_\Psi = 400, \omega = 50, min_{VC} = 0.75$ $min_M = 2, min_{ME} = 0.60, \sigma_\theta = \frac{\pi}{2}, max_{n_p} = 12$ Floating-point-based version: enabled, $\mu_P = 20$ $w_R = 0.5, \mu_1^\rho = 5, \tau_P = 0.6, min_{n_p} = 4, \tau_1^\rho = -1.6$ $\mu_2^\rho = \frac{\pi}{12}, \tau_2^\rho = -30, \mu_3^\rho = \frac{\pi}{12}, \tau_3^\rho = -30, n_{rel} = 5$	[9]
mindtct	output format = ANSI INCITS 378-2004 image enhancement = enabled	[42]

4.2.1. Database generation and parameters of the algorithms

In order to obtain very large databases and to control the fingerprint characteristics, we used the SFinGe software [3, 43] to generate synthetic fingerprints using the parameters specified in Table 3. The fingerprint pairs are composed by joining two synthetic fingerprints. A fingerprint cannot be included in more than one pair to ensure that all pairs are unique and disjoint in the database.

Table 3: Parameter specification used with the SFinGe tool

Scanner parameters	Generation parameters	Output settings
Acquisition area: 14.6mm x 19.6mm. Resolution: 500 dpi. Image size: 288 x 384. Background type: Optical. Background noise: Default. Crop borders: 0 x 0.	Impression per finger: 25. Class distribution: Natural. Varying quality and perturbations. Generate pores: enabled. Save ISO templates: enabled.	Output file type: WSQ.

The test set for all the experiments carried out in this paper with the SFinGe database is composed of 1000 random input pairs, which are used to perform 1000 different identifications in the database of 50 000 template fingerprint pairs. Each input pair is formed by a different impression of each fingerprint of a template pair. Therefore, we obtain accuracy measures that range from 0 to 1000.

4.2.2. *Discussion of the results*

This section discusses the obtained results for all the described variants of DPD-DFF over the SFinGe database. For the experiments, the rank values used to build the candidate set have been taken among the multiples of the number of cores of the cluster (144 in our setup), to maximize the throughput. However, we have also used lower values of the rank those in order to enrich the study and obtain more information about the behavior of the obtained accuracy.

Table 4: Results of DPD-DFF with 1000 test identifications (rank)

	SSR				**SDR**				**DSR**				**DDR**			
r	TP	FP	FN	t_{avg} (s)	TP	FP	FN	t_{avg} (s)	TP	FP	FN	t_{avg} (s)	TP	FP	FN	t_{avg} (s)
12	927	73	0	0.1588	928	72	0	0.1845	994	6	0	0.2870	996	4	0	0.3174
24	948	52	0	0.1597	949	51	0	0.1872	994	6	0	0.2776	997	3	0	0.3190
48	956	44	0	0.1589	958	42	0	0.1884	993	7	0	0.2815	997	3	0	0.3199
96	968	32	0	0.1597	970	30	0	0.1882	993	7	0	0.2889	997	3	0	0.3190
144	972	28	0	0.1606	974	26	0	0.1889	995	5	0	0.2833	999	1	0	0.3212
288	973	27	0	0.1603	976	24	0	0.1890	995	5	0	0.2916	999	1	0	0.3230
576	980	20	0	0.1696	983	17	0	0.2095	994	6	0	0.2977	999	1	0	0.3447
1152	984	16	0	0.1889	988	12	0	0.2680	992	8	0	0.3215	999	1	0	0.3788
2304	990	10	0	0.2412	995	5	0	0.3345	993	7	0	0.3485	1000	0	0	0.4483
4608	989	11	0	0.2897	996	4	0	0.4547	990	10	0	0.4185	1000	0	0	0.5887
9216	987	13	0	0.4313	996	4	0	0.7369	990	10	0	0.5665	1000	0	0	0.8665
18432	988	12	0	0.7379	998	2	0	1.3333	990	10	0	0.8481	1000	0	0	1.4356
36864	989	11	0	1.4270	1000	0	0	2.5521	989	11	0	1.4728	1000	0	0	2.5772

Table 5: Results of DPD-DFF with 1000 test identifications (threshold)

	SST					**SDT**				**DST**					**DDT**			
ϕ	$\lvert C\rvert_{avg}$	TP	FP	FN	t_{avg} (s)	TP	FP	FN	t_{avg} (s)	$\lvert C\rvert_{avg}$	TP	FP	FN	t_{avg} (s)	TP	FP	FN	t_{avg} (s)
0.05	46528.2	989	11	0	1.5153	1000	0	0	2.9382	49242.4	989	11	0	1.7302	1000	0	0	3.1920
0.10	22913.0	989	11	0	0.8381	999	1	0	1.6204	23580.8	990	10	0	1.0345	1000	0	0	1.8055
0.15	5404.7	989	11	0	0.3223	993	7	0	0.5450	2511.3	993	7	0	0.3727	999	1	0	0.4911
0.20	685.4	964	30	6	0.1925	967	27	6	0.2319	81.7	995	4	1	0.2865	997	2	1	0.3197
0.25	39.9	911	54	35	0.1583	912	53	35	0.1878	1.8	969	0	31	0.2819	969	0	31	0.3170
0.30	1.5	796	26	178	0.1515	796	26	178	0.1792	0.9	884	0	116	0.2722	884	0	116	0.3115

Tables 4 and 5 present the results of the eight variants of DPD-DFF. Note that columns $|C|_{avg}$ and t_{avg} contain average values over the 1000 performed identifications. Accordingly, Figure 2 plots the TPR along with the average identification time (note the logarithmic scale) for each variant of DPD-DFF and each reference AFIS. The following highlights can be extracted:

- The accuracy increases along with the amount of used information, so that DS* and DD* approaches are the most accurate ones.
- For a same average candidate set size, the rank approach produces more accurate results than the threshold variants, especially for small candidate sets. This might seem surprising because given an input pair, if both variants produce a candidate set of the same size, then these candidate sets are the same. However, recall that

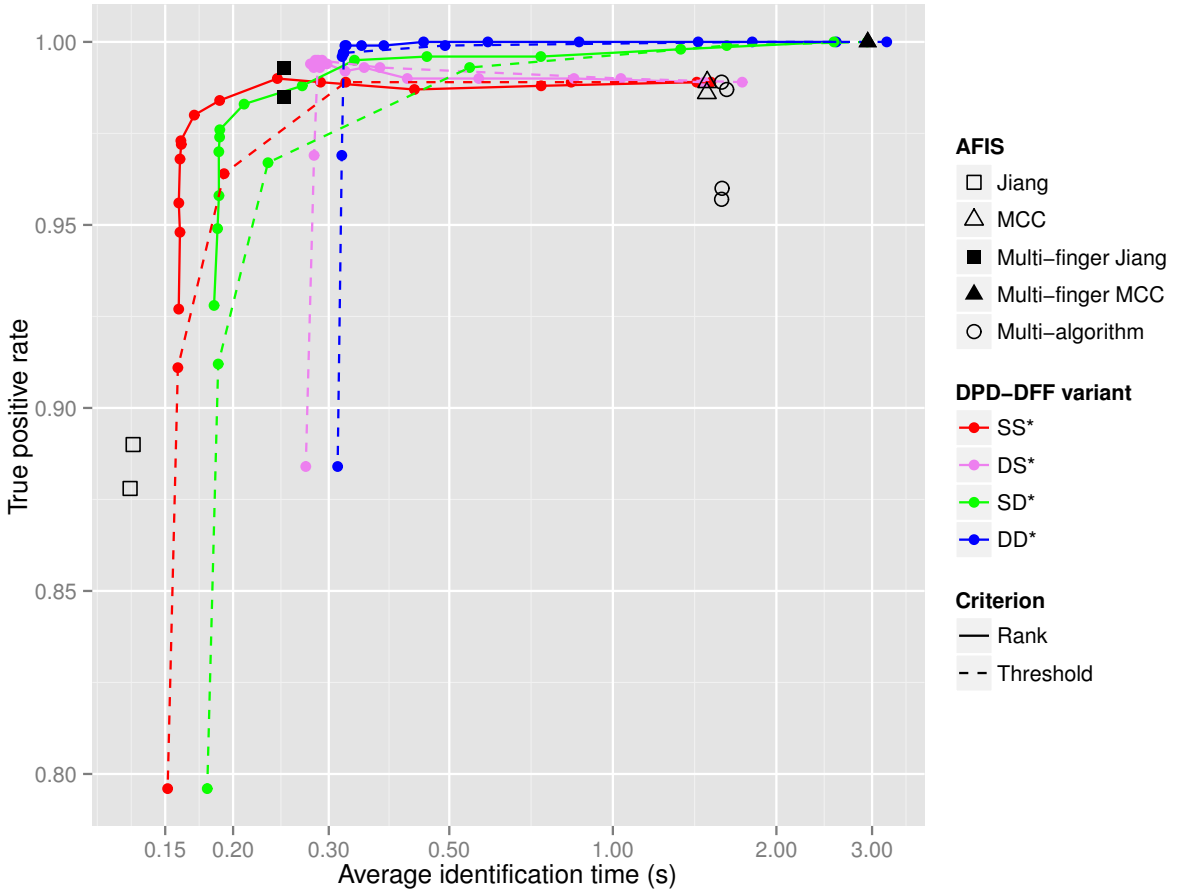


Figure 2: Average identification time and accuracy with the SFinGe database (N = 50000)

Table 5 shows the average set size. The actual value of $|C|$ in the threshold variant is different for each input pair, which makes the variant less robust.

- The rank ensures that there are no false negatives because it allows us to fix the size of the candidate set, ensuring $|C| > 0$ and offering better control of the overall identification time.
- Even for a small $|C|$, the rank variants outperform Jiang over a single fingerprint.
- Similarly, for a very large $|C|$, DPD-DFF relies more on the second phase and therefore the results get closer to those obtained by MCC. The DS* variants are a particular case because the accuracy decreases as $|C|$ increases. As the candidate set grows, they rely less on multi-finger Jiang, and more on single-finger MCC, which is less accurate than the former.
- The most accurate variant is DDR, which uses the two fingerprints with both algorithms and the rank.

- The DS* and DD* variants of DPD-DFF outperform all reference AFIS, reaching 100% TPR along with the multi-finger approach with MCC.

If the average identification time is also taken into account, the following conclusions arise:

- As expected, in general the more precise variants also take more time. These results show how DPD-DFF can be tuned according to the system needs, so that reasonably good results can be obtained very quickly (SSR variant, $r = 576$), and very precise results can be obtained with slower configurations (DDR variant, $r = 2304$).

- These tables also show that given a certain variant, the configurations with small candidate sets have a very similar runtime because all the matchings can be performed in parallel, but the accuracy is better for bigger candidate sets. Therefore, in a real environment, configurations with less than one candidate per core are usually not interesting, as they do not use the whole capacity of the cluster.

In summary, the DPD-DFF model dominates in time and accuracy all the tested AFIS, even the multi-finger approaches which provide very good accuracy. The DDR variant reaches 100% TPR in about 0.45 seconds, while the only reference AFIS that reaches this accuracy (multi-finger MCC) takes 3 seconds.

4.3. DBSpain654

A database of 654 fingerprint pairs was captured by the authors to test DPD-DFF on a controlled framework. This section describes the database (Section 4.3.1), the results obtained (Section 4.3.2), and an additional study with impostor fingerprints (Section 4.3.3). Due to the size of this database, all experiments described in this section were carried out using a single computer.

4.3.1. Database description

The fingerprints belong to the forefinger and middle finger of both hands of 334 non-experienced subjects from three different cities. Note that 14 fingerprint pairs failed in their enrolment and therefore were excluded from the database, making the resulting number of 654 pairs.

Both fingerprints of each pair were captured within the same image using a Suprema RealScan-D sensor. Each pair was captured 2 times as a template and 12 as an input over 3 different sessions several weeks apart. To compose the database and the test input set for this study, a single template and a single random input capture were selected for each pair. Then, the NIGOS *nfseg* algorithm [42] was used to segment the image and separate both fingerprints of each pair before applying the minutiae extraction.

4.3.2. Discussion of the results

Tables 6 and 7 present the results of the eight variants of the proposed DPD-DFF. Figure 3 depicts both accuracy and the average identification time of all tested AFIS. The following conclusions can be extracted from these results:

- The algorithms behave in the same way as in the previously studied databases: Jiang is less precise than MCC, and the multi-finger approaches obtain the best results both among the reference AFIS and the DPD-DFF variants.
- Again, the rank variants show more robust behavior than the threshold ones for the same average size of the candidate set. The DDR variant obtains the best performance possible for any number of candidates.
- DDR and DSR dominate all the considered multi-algorithm AFIS, and get the same TPR as multi-finger MCC in a much faster time.
- The multi-finger Jiang algorithm is faster than DPD-DFF. Actually, it corresponds to the first phase of the DS* and DD* variants, and it is clear that its accuracy is significantly improved with a small time overhead.

Table 6: Results of DPD-DFF with 654 test identifications (rank)

	SSR				**SDR**				**DSR**				**DDR**			
r	TP	FP	FN	t_{avg} (s)	TP	FP	FN	t_{avg} (s)	TP	FP	FN	t_{avg} (s)	TP	FP	FN	t_{avg} (s)
2	614	40	0	0.0622	614	40	0	0.1000	652	2	0	0.0891	653	1	0	0.1264
4	623	31	0	0.0641	623	31	0	0.1020	652	2	0	0.0902	654	0	0	0.1277
8	629	25	0	0.0655	629	25	0	0.1030	651	3	0	0.0915	654	0	0	0.1296
12	634	20	0	0.0658	635	19	0	0.1040	648	6	0	0.0919	654	0	0	0.1302
24	639	15	0	0.0666	642	12	0	0.1058	647	7	0	0.0927	654	0	0	0.1315
48	642	12	0	0.0805	645	9	0	0.1330	647	7	0	0.1073	654	0	0	0.1598

Table 7: Results of DPD-DFF with 654 test identifications (threshold)

	SST					**SDT**				**DST**					**DDT**			
ϕ	$\|C\|_{avg}$	TP	FP	FN	t_{avg} (s)	TP	FP	FN	t_{avg} (s)	$\|C\|_{avg}$	TP	FP	FN	t_{avg} (s)	TP	FP	FN	t_{avg} (s)
0.15	113.5	641	13	0	0.1165	646	8	0	0.2114	61.1	647	7	0	0.1205	654	0	0	0.1863
0.20	19.3	631	17	6	0.0694	632	16	6	0.1128	4.1	645	1	8	0.0893	646	0	8	0.1266
0.25	2.0	600	19	35	0.0619	600	19	35	0.0972	1.0	616	2	36	0.0867	616	2	36	0.1211
0.30	0.8	545	2	107	0.0590	545	2	107	0.0924	0.9	562	0	92	0.0857	562	0	92	0.1194

4.3.3. Results using impostor fingerprints

This section provides additional accuracy results for the DBSpain654 database.

In this section, the introduction of impostor fingerprints in the database requires additional error measures to study the behavior of DPD-DFF:

- **False Acceptance Rate (FAR)**: rate of impostor fingerprints that are erroneously identified as genuine ones.
- **False Rejection Rate (FRR)**: rate of genuine fingerprints that are erroneously rejected.
- **Equal Error Rate (EER)**: error when FAR and FRR are equal.

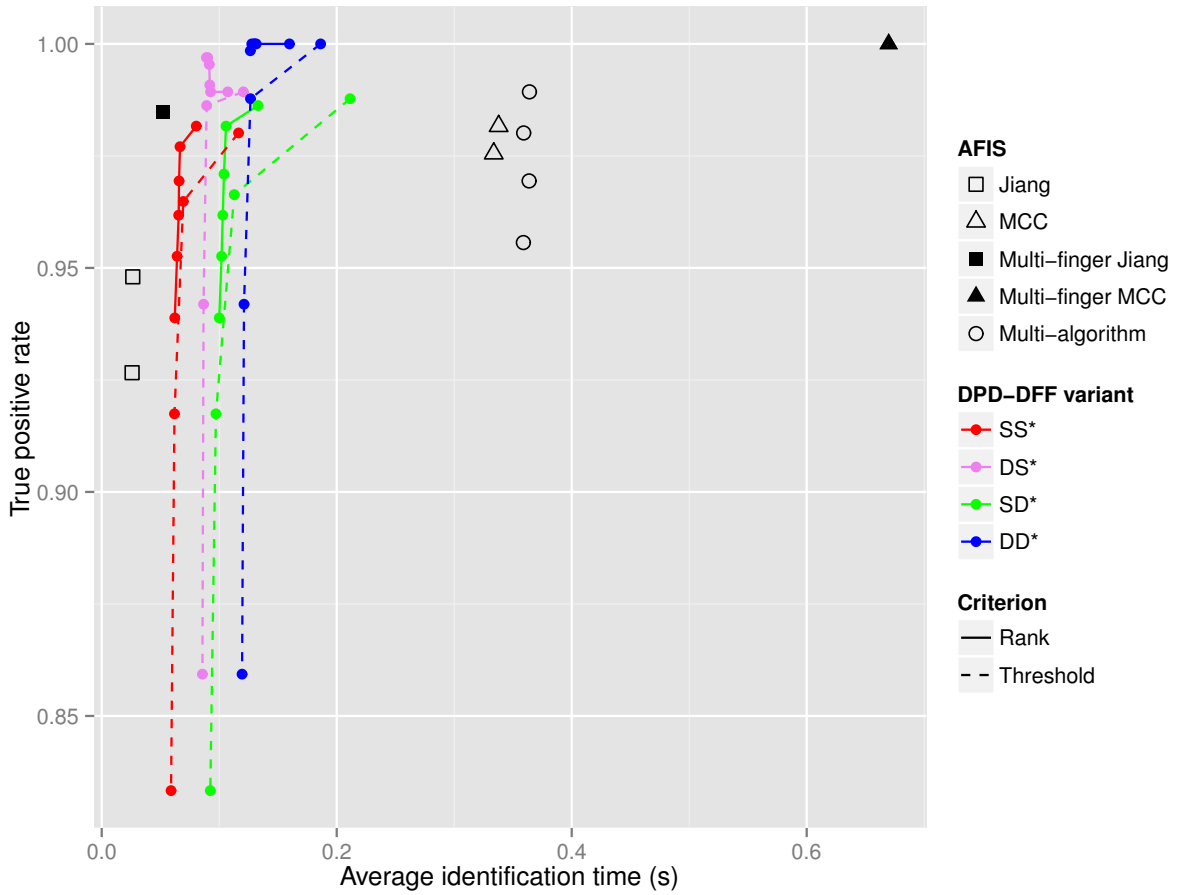


Figure 3: Runtime and accuracy with the captured database ($N = 654$)

- **FAR100, FAR1000**: FRR when FAR is 1% and 0.1%, respectively.
- **True Negatives (TN)**: number of input fingerprints that are not in the database, and are correctly detected as such.
- **True Negative Rate (TNR)**: quotient of TN and the number of impostor test fingerprints.

Tables 8 and 9 depict these error measures for all tested variants of DPD-DFF. To calculate these values, we took 3 random input fingerprint pairs for each of the 654 templates, and matched them with all the templates, making a total of 1 283 148 matchings for each matcher and each finger.

These tables show that the error rates become very low when the candidate set is big enough, especially for the DDR variant. Additionally, the FAR100 and FAR1000 values are very similar in most cases, meaning that the FAR drops quickly while the FRR remains almost constant, stating the robustness of DPD-DFF. The DRR variant obtains a very low FAR1000 when $r = 48$, which corresponds to a system that is robust against attacks (0.1% FAR), while avoiding rejections of genuine identities (0.25% FRR).

Table 8: Additional error measures (in percentages) using DPD-DFF (rank)

	SSR			SDR			DSR			DDR		
r	EER	FAR100	FAR1000	EER	FAR100	FAR1000	EER	FAR100	FAR1000	EER	FAR100	FAR1000
2	6.2691	6.2691	6.3462	6.2691	6.2691	6.2691	1.3252	1.3252	1.4547	1.3252	1.3252	1.3252
4	5.3007	5.3007	5.4536	5.3007	5.3007	5.3007	0.9684	0.9684	1.1879	0.9684	0.9684	0.9684
8	4.4852	4.4907	4.6406	4.4852	4.4852	4.4852	0.8396	0.7875	1.2571	0.7645	0.7645	0.7645
12	3.8226	3.9012	4.0571	3.8226	3.8226	3.8226	0.7715	0.7344	1.1956	0.6116	0.6116	0.6116
24	2.7405	2.8525	3.2449	2.7013	2.7013	2.7077	0.5750	0.5607	1.0508	0.3568	0.3568	0.3818
48	2.0346	2.0897	2.5592	1.8858	1.8858	1.9217	0.6132	0.4497	1.2210	0.2039	0.2039	0.2519

Table 9: Additional error measures (in percentages) using DPD-DFF (threshold)

	SST			SDT			DST			DDT		
θ	EER	FAR100	FAR1000	EER	FAR100	FAR1000	EER	FAR100	FAR1000	EER	FAR100	FAR1000
0.15	1.9888	2.0829	2.7641	1.7848	1.7848	1.8358	0.6938	0.5347	1.3761	0.3058	0.3058	0.3851
0.20	4.5385	4.6810	4.9608	4.5385	4.5385	4.5385	1.6820	1.6820	1.8941	1.6820	1.6820	1.6820
0.25	9.5360	9.5360	9.6470	9.5360	9.5360	9.5360	5.8104	5.8104	5.8104	5.8104	5.8104	5.8104
0.30	18.2050	18.2050	18.2050	18.2050	18.2050	18.2050	15.4944	15.4944	15.4944	15.4944	15.4944	15.4944

To conclude this section, we performed a new test, for which half of the fingerprints were randomly removed from the database, so that 50% of the input fingerprints become impostors trying to break into the system. The criterion used by DPD-DFF to determine if a fingerprint does not belong to the database is a score threshold within the accurate phase: if the fingerprint selected as the most similar to the input does not reach that threshold, the input is considered to be an impostor. The threshold used for these tests was the one that gives 0.01% FAR for the standalone MCC algorithm.

Table 10: Results of DPD-DFF with impostors and 654 test identifications (rank)

	SSR				**SDR**				**DSR**				**DDR**			
r	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
2	303	327	0	24	309	324	3	18	321	326	1	6	327	325	2	0
4	305	327	0	22	311	324	3	16	321	326	1	6	327	325	2	0
8	308	327	0	19	314	324	3	13	321	326	1	6	327	325	2	0
12	310	327	0	17	316	323	4	11	321	325	2	6	327	323	4	0
24	313	326	1	14	319	322	5	8	321	322	5	6	327	321	6	0
48	317	324	3	10	323	320	7	4	321	320	7	6	327	320	7	0

The results presented in Tables 10 and 11 show that, in contrast to the behavior of the TP, the TN decreases as the candidate size grows. This happens because a smaller candidate set allows the impostors to be detected during the first phase, while a bigger set makes the system more vulnerable to such attacks. This behavior provides good flexibility for the system: it can focus either on rejecting impostors or avoiding false rejections by modifying the rank parameter.

All rank variants of DPD-DFF show very good accuracy results when detecting impostors while identifying genuine fingerprints, keeping both FP and FN very low. As an example, DDR obtains the best result with the smallest r, and therefore the fastest configuration, without any false negatives in all cases. Similarly, the SSR variant is the one

Table 11: Results of DPD-DFF with impostors and 654 test identifications (threshold)

	SST				SDT				DST				DDT			
θ	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN	TP	TN	FP	FN
0.15	316	323	4	11	322	320	7	5	321	319	8	6	327	321	6	0
0.20	308	326	1	19	314	323	4	13	319	326	1	8	325	324	3	2
0.25	295	327	0	32	301	325	2	26	307	327	0	20	311	327	0	16
0.30	260	327	0	67	264	327	0	63	279	327	0	48	281	327	0	46

that is more robust against false positives, although this happens at the cost of a worse false negative rate.

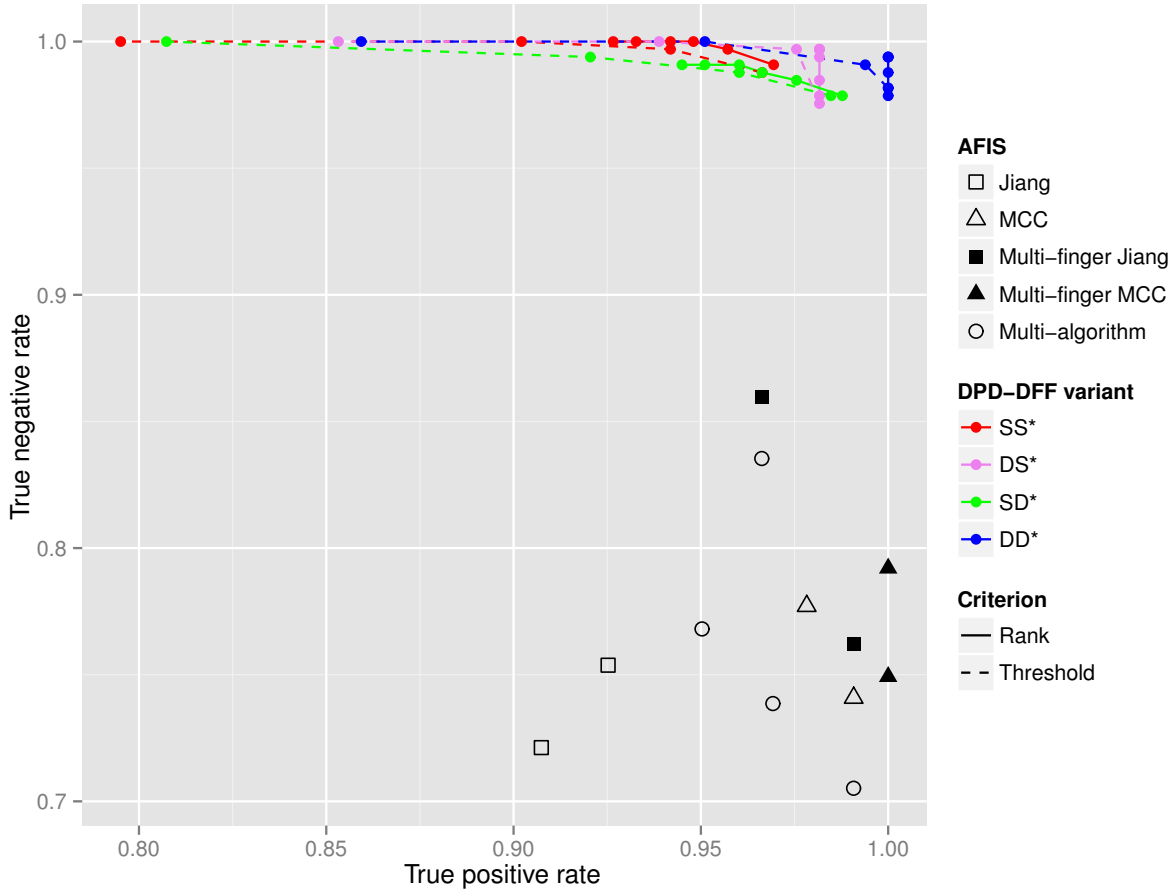


Figure 4: True positive rate and true negative rate with the captured database

Figure 4 shows how the proposed system dominates by far all reference AFIS in terms of the trade-off between false rejections and false acceptances. It can be seen that the DDR variant reaches almost 100% of both measures at the same time.

4.4. NIST-DB14 database

The NIST-DB14 database is composed of 27 000 rolled fingerprints, each of which was captured twice [44]. Tables 12 and 13 present the results of DPD-DFF over the NIST-DB14 database for the rank and threshold variants, respectively, using 12 slave machines. Figure 5 displays graphically the values of the tables.

Table 12: Results of DPD-DFF with 1000 test identifications (rank)

	SSR				**SDR**				**DSR**				**DDR**			
r	TP	FP	FN	t_{avg} (s)	TP	FP	FN	t_{avg} (s)	TP	FP	FN	t_{avg} (s)	TP	FP	FN	t_{avg} (s)
12	244	756	0	2.0345	275	725	0	2.3242	335	665	0	3.5431	357	643	0	3.8281
24	259	741	0	2.0564	312	688	0	2.3920	357	643	0	3.5769	393	607	0	3.9234
48	272	728	0	2.0726	340	660	0	2.4209	367	633	0	3.5816	418	582	0	3.9301
96	287	713	0	2.0865	369	631	0	2.4353	382	618	0	3.5872	442	558	0	3.9446
144	294	706	0	2.0891	383	617	0	2.4418	388	612	0	3.5965	458	542	0	3.9531
288	303	697	0	2.1033	405	595	0	2.4644	393	607	0	3.6070	485	515	0	3.9681
576	311	689	0	2.2739	431	569	0	2.8361	391	609	0	3.7861	507	493	0	4.3407
1152	324	676	0	2.5856	458	542	0	3.5211	393	607	0	4.1169	528	472	0	5.0399
2304	318	682	0	3.1847	482	518	0	4.8292	395	605	0	4.7404	544	456	0	6.3622
4608	331	669	0	4.3706	499	501	0	7.3712	381	619	0	5.9571	556	444	0	8.9160
9216	336	664	0	6.6912	518	482	0	12.3243	369	631	0	8.2953	559	441	0	13.8824

Table 13: Results of DPD-DFF with 1000 test identifications (threshold)

	SST					**SDT**				**DST**					**DDT**			
ϕ	$\lvert C\rvert_{avg}$	TP	FP	FN	t_{avg} (s)	TP	FP	FN	t_{avg} (s)	$\lvert C\rvert_{avg}$	TP	FP	FN	t_{avg} (s)	TP	FP	FN	t_{avg} (s)
0.10	13065.3	344	656	0	8.9011	535	465	0	17.0552	13380.5	360	640	0	10.7978	555	445	0	19.1454
0.15	3820.7	329	670	1	4.2518	484	515	1	7.2158	2197.2	394	605	1	4.9861	529	470	1	6.8838
0.20	367.3	298	684	18	2.2591	389	593	18	2.8427	49.7	342	540	118	3.5687	380	502	118	3.9022
0.25	6.0	191	476	333	1.9899	202	465	333	2.2579	0.2	137	31	832	3.4112	137	31	832	3.5791

It has to be noted that the TPR is surprisingly low, in discrepancy with other studies that highlight these matchers as accurate for the NIST-DB14 database. However, they may require specific tuning to be optimized for rolled fingerprints, which falls beyond the scope of this study. Therefore, we focus on the results obtained with general-purpose parameters that can highlight the robustness of the tested AFIS. For this database, which is difficult and computationally expensive, DPD-DFF outperforms all other approaches, in terms of both identification time and accuracy. The DDR variant is able to obtain better accuracy than the multi-finger MCC in about 25% more of the time than that required by the latter.

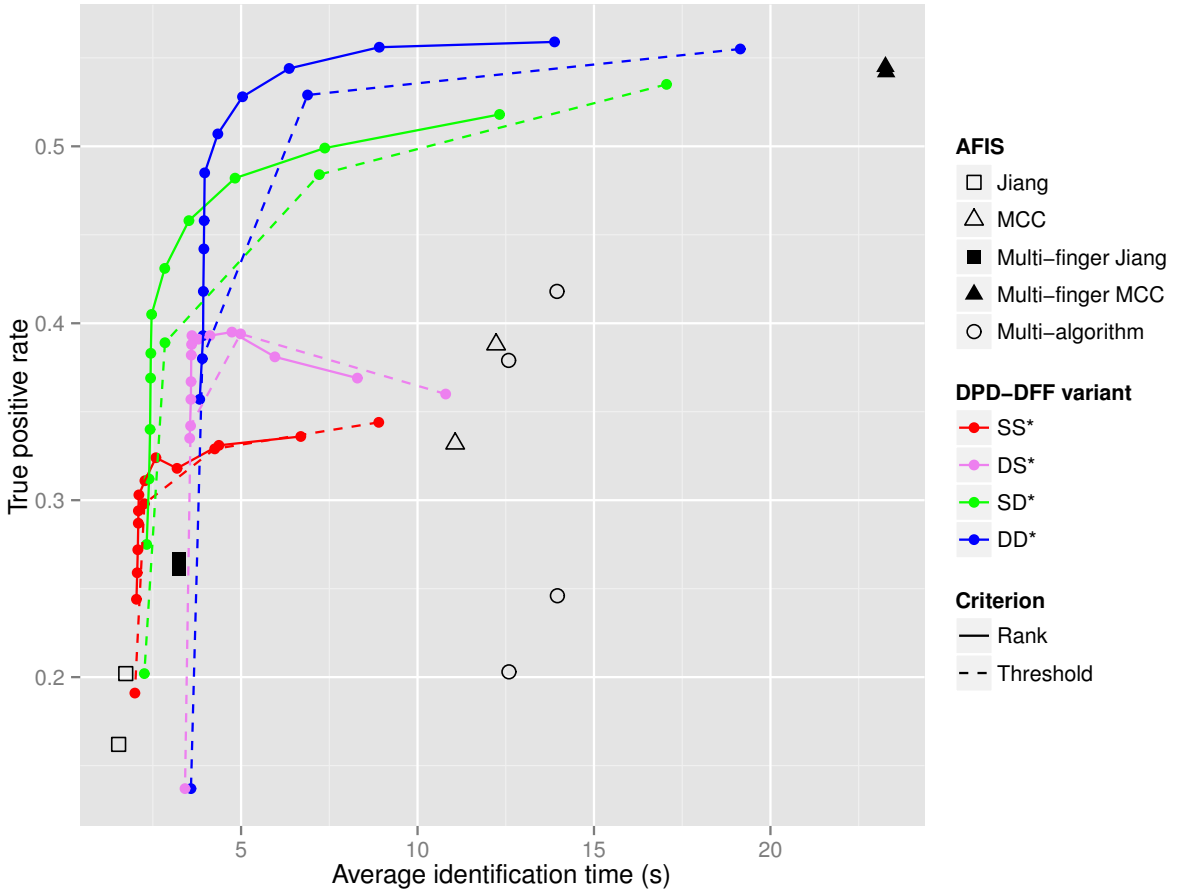


Figure 5: Average identification time and accuracy with NIST-DB14 (N = 21600)

4.5. Other real databases

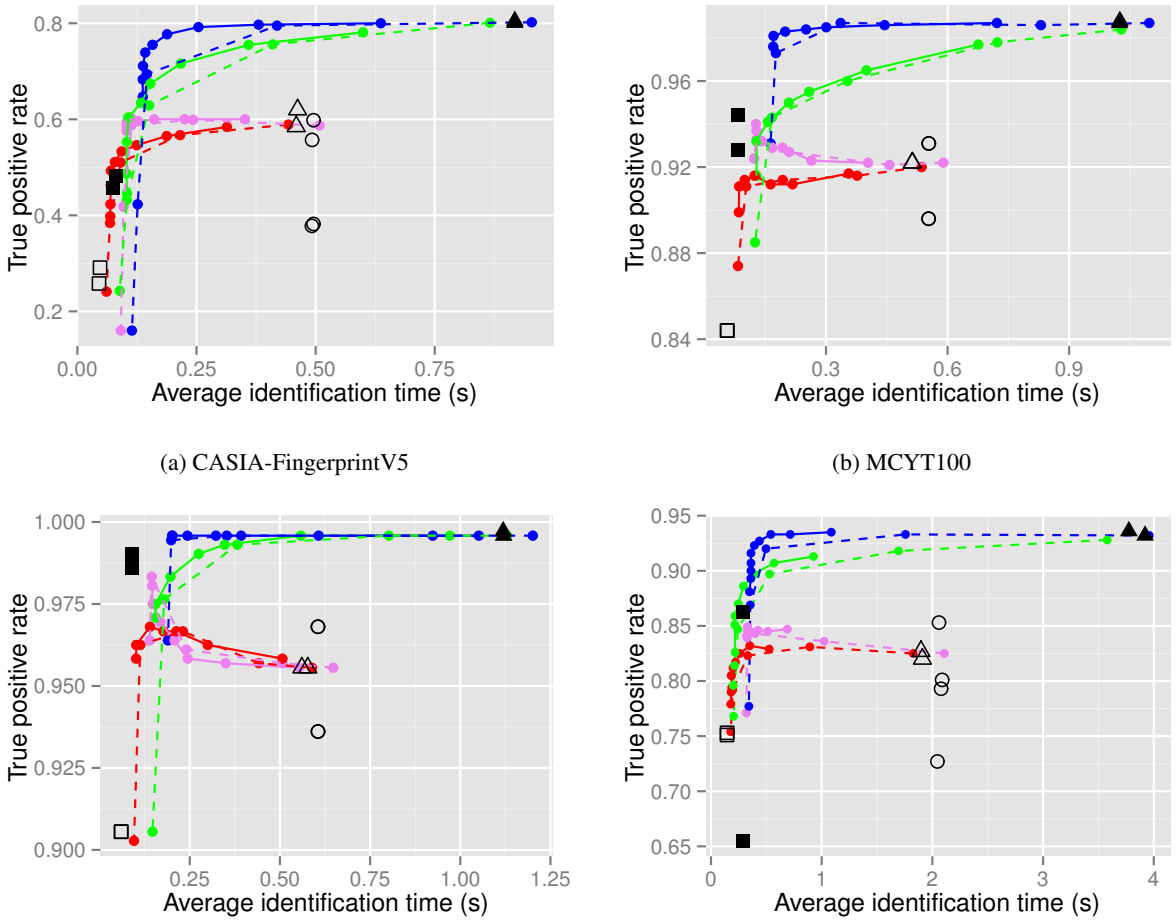
We have performed an extensive experimental study over several publicly available databases composed of real fingerprints. The objective of this study is to analyze the behavior of DPD-DFF in several realistic systems, where the fingerprints have been captured by different sensors and techniques, and to do so in a reproducible way. DB25496 is a mixture of the other four real databases formed by plain fingerprints (DBSpain654, CASIA-FingerprintV5, MCYT100 and FingerPass), where four captures of each fingerprint pair were included into the template database. Table 14 summarizes the characteristics of the three selected databases. Figure 6 displays graphically the time and accuracy values obtained for them.

Again, the DDR variant is able to obtain the same results as the multi-finger MCC, in a much shorter time frame. For the smaller databases, the multi-finger Jiang AFIS is able to obtain results that are faster than any of the variants of DPD-DFF. However, this time difference is less than 0.1s, which is usually an acceptable time to spend if the accuracy is improved. For a bigger database with the same characteristics, the relative difference in time would decrease as the accurate phase overhead would represent a smaller proportion of the overall time, thus making DPD-DFF even more

Table 14: Summary of the used real databases

Database	Subjects	Fingers	Template pairs	Input pairs	Machines used	Reference
CASIA-FingerprintV5	500	8	4000	1000	4	[45]
MCYT100	100	10	1000	1000	1	[46]
FingerPass	90	8	720	720	1	[47]
DB25496	1024	–	25 496	1000	12	[28]

suitable for the identification. This is reflected in the largest and most difficult databases (CASIA-FingerprintV5 and DB25496), as well as in the previously analyzed NIST-DB14, where DPD-DFF improves accuracy in all the reference results.

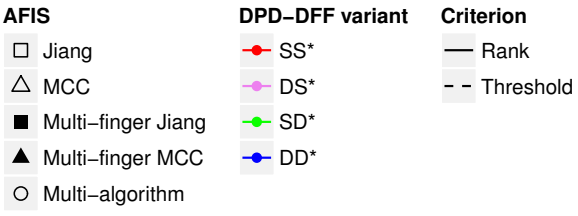


(a) CASIA-FingerprintV5

(b) MCYT100

(c) FingerPass

(d) DB25496



(e) Legend

Figure 6: Average identification time and accuracy with the additional real databases

5. Conclusions

In this paper, we have proposed a novel dual phase identification model (denoted DPD-DFF) to address the identification problem in large fingerprint databases. Its goal is to overcome the two problems that arise when dealing with this kind of database: the accuracy loss and the long runtime. To do so, the model combines two matching algorithms and two fingerprints per identity, using a mixed decision-level and score-level fusion, and has been implemented in a distributed system.

One of the main strengths of the proposed system is its flexibility, so that it can be tuned to the desired balance between accuracy and speed. Furthermore, the proposal has been tested over six fingerprint databases of diverse characteristics. The attained results have shown that the solutions obtained by our model dominate both in time and in accuracy over those obtained by using a single fingerprint or score fusion with either two fingerprints or two matchers, especially when large or complex databases are involved.

With a database of 50 000 fingerprint pairs, the algorithm reaches 100% TPR for identification taking only 0.44 seconds in a cluster of 12 machines. As for the fast results, 98.0% accuracy is obtained within 0.17 seconds.

The experiments carried out over the remaining databases have confirmed these conclusions. The additional study including impostor scores claims that DPD-DFF is much more precise than the three reference AFIS in terms of the trade-off between TPR and TNR, being able to eliminate any false negatives within a fast identification time.

Acknowledgments

This work was supported by the research projects TIN2014-57251-P, TIN2013-47210-P and P12-TIC-2958. D. Peralta holds an FPU scholarship from the Spanish Ministry of Education and Science (FPU12/04902). I. Triguero holds a BOF postdoctoral fellowship from the Ghent University.

Portions of the research in this paper use the CASIA-FingerprintV5 collected by the Chinese Academy of Sciences' Institute of Automation (CASIA).

References

[1] A. K. Jain, R. M. Bolle, S. Pankanti, Biometrics: Personal Identification in Networked Society, Springer, 2005.

[2] S. Pankanti, S. Prabhakar, A. K. Jain, On the individuality of fingerprints, IEEE Trans. Pattern Anal. Mach. Intell. 24 (8) (2002) 1010–1025.

[3] D. Maltoni, D. Maio, A. K. Jain, S. Prabhakar, Handbook of fingerprint recognition, Springer-Verlag New York Inc, 2009.

[4] A. Jain, L. Hong, R. Bolle, On-line fingerprint verification, IEEE Trans. Pattern Anal. Mach. Intell. 19 (4) (1997) 302–314.

[5] A. K. Jain, L. Hong, S. Pankanti, R. Bolle, An identity-authentication system using fingerprints, Proc. IEEE 85 (9) (1997) 1365–1388.

[6] N. Ratha, R. Bolle, Automatic Fingerprint Recognition Systems, Springer, New York, 2004.

[7] D. Peralta, M. Galar, I. Triguero, D. Paternain, S. García, E. Barrenechea, J. M. Benitez, H. Bustince, F. Herrera, A Survey on Fingerprint Minutiae-based Local Matching for Verification and Identification: Taxonomy and Experimental Evaluation, Inf. Sci. 315 (2015) 67–87.

[8] X. Jiang, W. Y. Yau, Fingerprint minutiae matching based on the local and global structures, in: Proc. 15th Int. Conf. Pattern Recognit., Vol. 2, IEEE, 2000, pp. 1038–1041.

[9] R. Cappelli, M. Ferrara, D. Maltoni, Minutia cylinder-code: A new representation and matching technique for fingerprint recognition, IEEE Trans. Pattern Anal. Mach. Intell. 32 (12) (2010) 2128–2141.

[10] P. Verlinde, G. Chollet, M. Acheroy, Multi-modal identity verification using expert fusion, Inf. Fusion 1 (1) (2000) 17–33.

[11] S. Prabhakar, A. K. Jain, Decision-level fusion in fingerprint verification, Pattern Recognit. 35 (4) (2002) 861–874.

[12] A. Ross, A. Jain, Information fusion in biometrics, Pattern Recognit. Lett. 24 (13) (2003) 2115–2125.

[13] A. K. Jain, P. Flynn, A. A. Ross, Handbook of biometrics, Springer, 2007.

[14] A. K. Jain, S. Prabhakar, S. Chen, Combining multiple matchers for a high security fingerprint verification system, Pattern Recognit. Lett. 20 (1999) 1371–1379.

[15] X. Jiang, W. Ser, Online fingerprint template improvement, IEEE Trans. Pattern Anal. Mach. Intell. 24 (8) (2002) 1121–1126.

[16] H. Xu, R. N. J. Veldhuis, Spectral minutiae representations for fingerprint recognition, in: Proc. 6th Int. Conf. Intell. Inf. Hiding Multimed. Signal Process., 2010, pp. 341–345.

[17] G. L. Marcialis, F. Roli, L. Didaci, Multimodal fingerprint verification by score-level fusion: An experimental investigation, J. Intell. Fuzzy Syst. 24 (2013) 51–60.

[18] N. K. Ratha, K. Karu, S. Chen, A. K. Jain, A real-time matching system for large fingerprint databases, IEEE Trans. Pattern Anal. Mach. Intell. 18 (8) (1996) 799–813.

[19] H. S. Stone, High-performance computer architecture, Addison-Wesley Longman Publishing Co., Inc., 1992.

[20] R. Armstrong, D. Gannon, A. Geist, K. Keahey, S. Kohn, L. McInnes, S. Parker, B. Smolinski, Toward a Common Component Architecture for High-Performance Scientific Computing, in: Proc. 8th IEEE Int. Symp. High Perform. Distrib. Comput., 1999, pp. 115–124.

[21] R. F. Miron, T. S. Letia, M. Hulea, Two server topologies for a distributed fingerprint-based recognition system, in: 15th Int. Conf. Syst. Theory, Control Comput., 2011, pp. 1–6.

[22] K. Beghdad Bey, Z. Guessoum, A. Mokhtari, F. Benhammadi, Agent based approach for distribution of fingerprint matching in a metacomputing environment, in: Proc. 8th Int. Conf. New Technol. Distrib. Syst., 2008, pp. 1–7.

[23] D. Peralta, I. Triguero, R. Sanchez-Reillo, F. Herrera, J. M. Benitez, Fast Fingerprint Identification for Large Databases, Pattern Recognit. 47 (2) (2014) 588–602.

[24] P. D. Gutierrez, M. Lastra, F. Herrera, J. M. Benitez, A high performance fingerprint matching system for large databases based on GPU, IEEE Trans. Inf. Forensics Secur. 9 (1) (2014) 62–71.

[25] R. Cappelli, M. Ferrara, D. Maltoni, Large-scale fingerprint identification on GPU, Inf. Sci. 306 (2015) 1–20.

[26] Unique Authentication Authority of India.
URL `http://uidai.gov.in/`

[27] Integrated Automated Fingerprint Identification System.
URL `http://www.fbi.gov/about-us/cjis/fingerprints\_biometrics/iafis/iafis`

[28] D. Peralta, I. Triguero, S. García, F. Herrera, J. M. Benitez, Supplementary material for "DPD-DFF: A Dual Phase Distributed Scheme with Double Fingerprint Fusion for Fast and Accurate Identification in Large Databases", Tech. rep., Soft Computing and Intelligent Information Systems, University of Granada (2015).
URL `http://sci2s.ugr.es/sites/default/files/files/ComplementaryMaterial/DPDDFF/techrep.pdf`

[29] D. Peralta, M. Galar, I. Triguero, J. M. Benitez, F. Herrera, Minutiae Filtering to Improve Both Efficacy and Efficiency of Fingerprint Matching Algorithms, Eng. Appl. Artif. Intell. 32 (2014) 37–53.

[30] A. Lindoso, L. Entrena, J. Izquierdo, FPGA-based acceleration of fingerprint minutiae matching, in: Proc. 3rd South. Conf. Program. Log., 2007, pp. 81–86.

[31] X. Jiang, M. Liu, A. C. Kot, Fingerprint retrieval for identification, IEEE Trans. Inf. Forensics Secur. 1 (4) (2006) 532–542.

[32] M. Liu, X. Jiang, A. Chichung Kot, Efficient fingerprint search based on database clustering, Pattern Recognit. 40 (6) (2007) 1793–1803.

[33] R. Cappelli, M. Ferrara, D. Maltoni, Fingerprint indexing based on minutia cylinder-code, IEEE Trans. Pattern Anal. Mach. Intell. 33 (5) (2011) 1051–1057.

[34] M. Galar, J. Derrac, D. Peralta, I. Triguero, D. Paternain, C. Lopez-Molina, S. García, J. M. Benitez, M. Pagola, E. Barrenechea, H. Bustince, F. Herrera, A survey of fingerprint classification Part I: Taxonomies on feature extraction methods and learning models, Knowledge-Based Syst. 81 (2015) 76–97.

[35] M. Vatsa, R. Singh, A. Noore, K. Morris, Simultaneous latent fingerprint recognition, Appl. Soft Comput. 11 (7) (2011) 4260–4266.

[36] L. Nanni, D. Maio, Combination of different fingerprint systems: A case study FVC2004, Sens. Rev. 26 (1) (2006) 51–57.

[37] A. Noore, R. Singh, M. Vatsa, Robust memory-efficient data level information fusion of multi-modal biometric images, Inf. Fusion 8 (4) (2007) 337–346.

[38] T. Uz, G. Bebis, A. Erol, S. Prabhakar, Minutiae-based template synthesis and matching for fingerprint authentication, Comput. Vis. Image Underst. 113 (2009) 979–992.

[39] Y. Li, J. Yin, E. Zhu, Score-based fusion in multi-unit biometric recognition system, Appl. Mech. Mater. 48–49 (2011) 1010–1013.

[40] D. Gafurov, C. Busch, P. Bours, B. Yang, Fusion in fingerprint authentication: Two finger types vs. two scanner types, in: Proc. ACM Symp. Appl. Comput., 2011, pp. 13–20.

[41] A. K. Jain, S. Prabhakar, A. Ross, Fingerprint matching: Data acquisition and performance evaluation, Tech. Rep. TR99-14, MSU (1999).

[42] C. I. Watson, M. D. Garris, E. Tabassi, C. L. Wilson, R. M. Mccabe, S. Janet, K. Ko, User's Guide to NIST Biometric Image Software (NBIS), Tech. Rep. NISTIR-7392, NIST (2010).

[43] R. Cappelli, D. Maio, D. Maltoni, Synthetic fingerprint-database generation, in: Proc. 16th Int. Conf. Pattern Recognit., Vol. 3, 2002, pp. 744–747.

[44] C. I. Watson, NIST Special Database 14, Tech. rep., NIST (1993).

[45] CASIA-FingerprintV5 database.
URL `http://biometrics.idealtest.org/`

[46] J. Ortega-Garcia, J. Fierrez-Aguilar, D. Simon, J. Gonzalez, M. Faundez-Zanuy, V. Espinosa, A. Satue, I. Hernaez, J.-J. Igarza, C. Vivaracho, D. Escudero, Q.-I. Moro, MCYT baseline corpus: A bimodal biometric database, IEEE Proc. Vision, Image Signal Process. 150 (6) (2003) 395–401.

[47] X. Jia, X. Yang, Y. Zang, N. Zhang, J. Tian, A cross-device matching fingerprint database from multi-type sensors, in: Proc. Int. Conf. Pattern Recognit., 2012, pp. 3001–3004.

Bibliography

[Amd67] Amdahl G. M. (apr 1967) Validity of the single processor approach to achieving large scale computing capabilities. In *Proc. Spring Jt. Comput. Conf.*, pp. 483–485. ACM.

[BGMB08] Beghdad Bey K., Guessoum Z., Mokhtari A., and Benhammadi F. (2008) Agent based approach for distribution of fingerprint matching in a metacomputing environment. In *Proc. 8th Int. Conf. New Technol. Distrib. Syst.*, pp. 1–7.

[Cap11] Cappelli R. (2011) Fast and Accurate Fingerprint Indexing Based on Ridge Orientation and Frequency. *IEEE Trans. Syst. Man, Cybern. Part B Cybern.* 41(6): 1511–1521.

[CCHW97] Chang S. H., Cheng F. H., Hsu W. H., and Wu G. Z. (1997) Fast algorithm for point pattern matching: invariant to translations, rotations and scale changes. *Pattern Recognit.* 30(2): 311–320.

[CF01] Chang J. and Fan K. (2001) Fingerprint ridge allocation in direct gray-scale domain. *Pattern Recognit.* 34(10): 1907–1925.

[CFM10] Cappelli R., Ferrara M., and Maltoni D. (2010) Minutia cylinder-code: A new representation and matching technique for fingerprint recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* 32(12): 2128–2141.

[CFM15] Cappelli R., Ferrara M., and Maltoni D. (2015) Large-scale fingerprint identification on GPU. *Inf. Sci.* 306: 1–20.

[CGW+95] Candela G. T., Grother P. J., Watson C. I., Wilkinson R. A., and Wilson C. L. (1995) {PCASYS}- A Pattern-Level Classification Automation System for Fingerprints. Technical Report 5647, NIST.

[CLMM99] Cappelli R., Lumini A., Maio D., and Maltoni D. (may 1999) Fingerprint classification by directional image partitioning. *IEEE Trans. Pattern Anal. Mach. Intell.* 21(5): 402–421.

[CTY06] Chen X., Tian J., and Yang X. (2006) A new algorithm for distorted fingerprints matching based on normalized fuzzy similarity measure. *IEEE Trans. Image Process.* 15(3): 767–776.

[cud] NVIDIA CUDA.

[DG08] Dean J. and Ghemawat S. (jan 2008) {MapReduce}: simplified data processing on large clusters. *Commun. ACM* 51(1): 107–113.

[DGLN11] Danese G., Giachero M., Leporati F., and Nazzicari N. (2011) An embedded multicore biometric identification system. *Microprocess. Microsyst.* 35(5): 510–521.

[DHS12] Duda R. O., Hart P. E., and Stork D. G. (2012) *Pattern classification.* John Wiley & Sons.

[FdRL+14] Fernández A., del Río S., López V., Bawakid A., del Jesus M. J., Benitez J. M., and Herrera F. (2014) Big Data with Cloud Computing: an insight on the computing environment, {MapReduce}, and programming frameworks. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* 4(5): 380–409.

[GDP+15a] Galar M., Derrac J., Peralta D., Triguero I., Paternain D., Lopez-Molina C., García S., Benitez J. M., Pagola M., Barrenechea E., Bustince H., and Herrera F. (2015) A survey of fingerprint classification Part II: Experimental analysis and ensemble proposal. *Knowledge-Based Syst.* 81: 98–116.

[GDP+15b] Galar M., Derrac J., Peralta D., Triguero I., Paternain D., Lopez-Molina C., García S., Benitez J. M., Pagola M., Barrenechea E., Bustince H., and Herrera F. (2015) A survey of fingerprint classification Part I: Taxonomies on feature extraction methods and learning models. *Knowledge-Based Syst.* 81: 76–97.

[GLH15] García S., Luengo J., and Herrera F. (2015) *Data Preprocessing in Data Mining.* Springer, New York, 1st edition.

[GLHB14] Gutierrez P. D., Lastra M., Herrera F., and Benitez J. M. (2014) A high performance fingerprint matching system for large databases based on GPU. *IEEE Trans. Inf. Forensics Secur.* 9(1): 62–71.

[HAL+08] Hulea M., Aştilean A., Leţia T., Miron R., and Folea S. (may 2008) Fingerprint recognition distributed system. In *Proc. 16th IEEE Int. Conf. Autom. Qual. Testing, Robot.*, volumen 3, pp. 423–428.

[Hen00] Henry E. (1900) *Classification and Uses of Finger Prints.* George Routledge and Sons, Broadway, Ludgate Hill, United Kingdom.

[HMCC08] Hong J. H., Min J. K., Cho U. K., and Cho S. B. (2008) Fingerprint classification using one-vs-all support vector machines dynamically ordered with naive Bayes classifiers. *Pattern Recognit.* 41(2): 662–671.

[iaf] Integrated Automated Fingerprint Identification System.

[ISA11] Indrawan G., Sitohang B., and Akbar S. (jul 2011) Parallel Processing for Fingerprint Feature Extraction. In *Int. Conf. Electr. Eng. Informat.*, pp. 1–6.

[JCD07] Jain A., Chen Y., and Demirkus M. (jan 2007) Pores and Ridges: High-Resolution Fingerprint Matching Using Level 3 Features. *IEEE Trans. Pattern Anal. Mach. Intell.* 29(1): 15–27.

[JFN10] Jain A. K., Feng J., and Nandakumar K. (feb 2010) Fingerprint Matching. *Computer (Long. Beach. Calif).* 43(2): 36–44.

[JFR07] Jain A. K., Flynn P., and Ross A. A. (2007) *Handbook of biometrics.* Springer.

[JHB97] Jain A. K., Hong L., and Bolle R. (1997) On-line fingerprint verification. *IEEE Trans. Pattern Anal. Mach. Intell.* 19(4): 302–314.

[JHPB97] Jain A. K., Hong L., Pankanti S., and Bolle R. (1997) An identity-authentication system using fingerprints. *Proc. IEEE* 85(9): 1365–1388.

[JPC99] Jain A. K., Prabhakar S., and Chen S. (1999) Combining multiple matchers for a high security fingerprint verification system. *Pattern Recognit. Lett.* 20: 1371–1379.

[JR13] Jeffers J. and Reinders J. (2013) *Intel Xeon Phi Coprocessor High Performance Programming*. Newnes.

[JS02] Jiang X. and Ser W. (aug 2002) Online fingerprint template improvement. *IEEE Trans. Pattern Anal. Mach. Intell.* 24(8): 1121–1126.

[JY00] Jiang X. and Yau W. Y. (2000) Fingerprint minutiae matching based on the local and global structures. In *Proc. 15th Int. Conf. Pattern Recognit.*, volumen 2, pp. 1038–1041. IEEE.

[KJ96] Karu K. and Jain A. K. (1996) Fingerprint classification. *Pattern Recognit.* 29(3): 389–404.

[KKWZ15] Karau H., Konwinski A., Wendell P., and Zaharia M. (2015) *Learning Spark: Lightning-Fast Big Data Analysis*. O'Reilly Media, Inc.

[KS13] Kitano T. and Su L. (jun 2013) SPOAN: Load Balancing Replica Placement Strategy for Large Scale Biometric Identification Service. In *2013 IEEE Int. Congr. Big Data*, pp. 326–333. IEEE.

[KSH+12] Krizhevsky A., Sulskever I., Hinton G. E. G. E., Sutskever I., and Hinton G. E. G. E. (2012) ImageNet Classification with Deep Convolutional Neural Networks. *Adv. Neural Inf. Process. Syst.* pp. 1097–1105.

[LBH15] LeCun Y., Bengio Y., and Hinton G. (2015) Deep learning. *Nature* 521(7553): 436–444.

[LCG+15] Lastra M., Carabaño J., Gutierrez P. D., Benitez J. M., and Herrera F. (apr 2015) Fast fingerprint identification using GPUs. *Inf. Sci.* 301: 195–214.

[LGRN09] Lee H., Grosse R., Ranganath R., and Ng A. Y. (2009) Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proc. 26th Annu. Int. Conf. Mach. Learn. - ICML '09*, pp. 1–8. ACM Press, New York, New York, USA.

[LHC00] Liu J., Huang Z., and Chan K. L. (2000) Direct minutiae extraction from gray-level fingerprint image by relationship examination. In *IEEE Int. Conf. Image Process.*, volumen 2, pp. 427–430.

[Liu10] Liu M. (2010) Fingerprint classification based on Adaboost learning from singularity features. *Pattern Recognit.* 43(3): 1062–1070.

[LN17] Lumini A. and Nanni L. (2017) Overview of the combination of biometric matchers. *Inf. Fusion* 33: 71–85.

[MLH11] Miron R. F., Letia T. S., and Hulea M. (2011) Two server topologies for a distributed fingerprint-based recognition system. In *15th Int. Conf. Syst. Theory, Control Comput.*, pp. 1–6.

[MMJP09] Maltoni D., Maio D., Jain A. K., and Prabhakar S. (2009) *Handbook of fingerprint recognition*. Springer, New York.

[MRD13] Marcialis G. L., Roli F., and Didaci L. (2013) Multimodal fingerprint verification by score-level fusion: An experimental investigation. *J. Intell. Fuzzy Syst.* 24: 51–60.

[NAKMM04] Nyongesa H. O., Al-Khayatt S., Mohamed S. M., and Mahmoud M. (2004) Fast robust fingerprint feature extraction and classification. *J. Intell. Robot. Syst.* 40(1): 103–112.

[NH04] Nagaty K. A. and Hattab E. (2004) An approach to a fingerprints multi-agent parallel matching system. In *IEEE Int. Conf. Syst., Man Cybern.*, volumen 5, pp. 4750–4756.

[NM06] Nanni L. and Maio D. (2006) Combination of different fingerprint systems: A case study FVC2004. *Sens. Rev.* 26(1): 51–57.

[NSV07] Noore A., Singh R., and Vatsa M. (2007) Robust memory-efficient data level information fusion of multi-modal biometric images. *Inf. Fusion* 8(4): 337–346.

[PGT+15] Peralta D., Galar M., Triguero I., Paternain D., García S., Barrenechea E., Benitez J. M., Bustince H., and Herrera F. (2015) A survey on fingerprint minutiae-based local matching for verification and identification: Taxonomy and experimental evaluation. *Inf. Sci.* 315: 67–87.

[PJ02] Prabhakar S. and Jain A. K. (2002) Decision-level fusion in fingerprint verification. *Pattern Recognit.* 35(4): 861–874.

[PPJ02] Pankanti S., Prabhakar S., and Jain A. K. (2002) On the individuality of fingerprints. *IEEE Trans. Pattern Anal. Mach. Intell.* 24(8): 1010–1025.

[PTSR+14] Peralta D., Triguero I., Sanchez-Reillo R., Herrera F., and Benitez J. M. (feb 2014) Fast Fingerprint Identification for Large Databases. *Pattern Recognit.* 47(2): 588–602.

[RB04] Ratha N. and Bolle R. (2004) *Automatic Fingerprint Recognition Systems*. Springer, New York.

[RBPV00] Ratha N., Bolle R., Pandit V., and Vaish V. (2000) Robust fingerprint authentication using local structural similarity. In *Proc. Fifth IEEE Work. Appl. Comput. Vis.*, pp. 29–34. IEEE Comput. Soc.

[RCJ95] Ratha N. K., Chen S., and Jain A. K. (1995) Adaptive flow orientation-based feature extraction in fingerprint images. *Pattern Recognit.* 28(11): 1657–1672.

[RJ03] Ross A. A. and Jain A. K. (2003) Information fusion in biometrics. *Pattern Recognit. Lett.* 24(13): 2115–2125.

[RKCJ96] Ratha N. K., Karu K., Chen S., and Jain A. K. (1996) A real-time matching system for large fingerprint databases. *IEEE Trans. Pattern Anal. Mach. Intell.* 18(8): 799–813.

[RNJ06] Ross A. A., Nandakumar K., and Jain A. K. (2006) *Handbook of multibiometrics*, volumen 6. Springer.

[Sen97] Senior A. (1997) A hidden Markov model fingerprint classifier. In *Proc. 31st Asilomar Conf. Signals, Syst. Comput.*, volumen 1, pp. 306–310. IEEE Comput. Soc, Pacific Grove, CA.

[SR11] Shelly and Raghava N. S. (sep 2011) Iris recognition on Hadoop: A biometrics system implementation on cloud computing. In *2011 IEEE Int. Conf. Cloud Comput. Intell. Syst.*, pp. 482–485. IEEE.

[Sto92] Stone H. S. (1992) *High-performance computer architecture*. Addison-Wesley, Reading, USA.

[uid] Unique Authentication Authority of India.

[WFH11] Witten I. H., Frank E., and Hall M. A. (2011) *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kauffman Series in Data Management Systems.

[WGT+10] Watson C. I., Garris M. D., Tabassi E., Wilson C. L., Mccabe R. M., Janet S., and Ko K. (2010) User's Guide to NIST Biometric Image Software (NBIS). Technical report, NIST.

[Whi12] White T. (2012) *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 3rd edition.

[YA04] Yager N. and Amin A. (apr 2004) Fingerprint verification based on minutiae features: a review. *Pattern Anal. Appl.* 7(1): 94–113.