Distributed Incremental Fingerprint Identification with Reduced Database Penetration Rate Using a Hierarchical Classification Based on Feature Fusion and Selection

Daniel Peralta[a,b,*], Isaac Triguero[c], Salvador García[d], Yvan Saeys[a,b], Jose M. Benitez[d], Francisco Herrera[d,e]

[a] *Department of Internal Medicine, Ghent University, Ghent, Belgium*
[b] *Data Mining and Modelling for Biomedicine group, VIB Center for Inflammation Research, Ghent, Belgium*
[c] *School of Computer Science, University of Nottingham, Jubilee Campus, Wollaton Road, Nottingham NG8 1BB, United Kingdom*
[d] *Department of Computer Science and Artificial Intelligence of the University of Granada, 18071 Granada, Spain*
[e] *Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia*

Abstract

Fingerprint recognition has been a hot research topic along the last few decades, with many applications and ever growing populations to identify. The need of flexible, fast identification systems is therefore patent in such situations. In this context, fingerprint classification is commonly used to improve the speed of the identification. This paper proposes a complete identification system with a hierarchical classification framework that fuses the information of multiple feature extractors. A feature selection is applied to improve the classification accuracy. Finally, the distributed identification is carried out with an incremental search, exploring the classes according to the probability order given by the classifier. A single parameter tunes the trade-off between identification time and accuracy. The proposal is evaluated over two NIST databases and a large synthetic database, yielding penetration rates close to the optimal values that can be reached with classification, leading to low identification times with small or no accuracy loss.

Keywords: Fingerprint recognition, fingerprint identification, fingerprint classification, large databases, feature selection, hierarchical classification

[*] Corresponding author

Email addresses: daniel.peralta@irc.vib-ugent.be (Daniel Peralta), Isaac.Triguero@nottingham.ac.uk (Isaac Triguero), salvagl@decsai.ugr.es (Salvador García), yvan.saeys@ugent.be (Yvan Saeys), J.M.Benitez@decsai.ugr.es (Jose M. Benitez), herrera@decsai.ugr.es (Francisco Herrera)

1. Introduction

Biometrics is a hot research field with many recent advances in different areas [1], such as signature [2], face [3] or fingerprint [4] recognition, template encryption [5], multimodal fusion [6] or spoofing detection [7]. They have been increasingly used during the last decades to replace tokens and passwords in many contexts. Fingerprint recognition is nowadays one of the most widespread methods for biometric identification [8]. It can be approached from two points of view: verification [9] aims to assess whether two fingerprints are taken from the same finger, whereas identification [10] consists of a search for a particular input fingerprint throughout a database of template fingerprints.

The scientific literature comprises some fast, accurate matching algorithms for verification [11, 12, 13, 14]. However, the identification of an unknown input fingerprint among a large database of templates is still a challenging problem, both in terms of accuracy and runtime [15]. The number of runs of the matching algorithm increases linearly with the number of template fingerprints. This leads to an increase of the probability of erroneous identification, due to the higher number of non-matching templates that may cause false positives, and to a linear increase of the identification time, which eventually becomes unacceptably large.

High Performance Computing (HPC) [16] puts together massive computing resources (such as CPUs and GPUs) to reduce the time needed for computationally expensive tasks. It has been steadily applied to palliate the large identification time problem [4, 17, 18]. However, for sufficiently large databases, or when limited resources and budgets come into play, an approach from a pure HPC point of view may not be enough to obtain an efficient Automatic Fingerprint Identification System (AFIS).

A solution for such cases consists of reducing the fraction of the search space that is explored for the identification, called the database penetration rate [19]. Two main families of methods can be distinguished [20]: 1) classification approaches [21] that divide the template database into a fixed number of classes, so that the input fingerprint is only searched within its own class, and 2) indexing methods [22] that map each fingerprint on a single or multi-dimensional space so that several impressions of the same fingerprint are mapped close to each other.

Classification is the most widespread of these two [21, 23]. Most current proposals are based on the five classes defined by Henry [24], which are distinguished according to their ridge pattern and have an uneven distribution: *Arch*, *Left Loop*, *Right Loop*, *Tented Arch* and *Whorl*. The global features of the fingerprint (such as orientations maps or singular points) are extracted for the classification, and encoded as a numerical vector. Then, this vector is used within a machine learning algorithm to perform the classification itself.

Some of these feature extraction methods reject the fingerprints that do not comply with certain quality criteria [25, 26, 27]. This behavior is oriented towards avoiding a harmful reduction of the penetration rate that may impede finding the correct identity after a misclassification [20]. However, the rejected fingerprints cannot be classified, restraining the reduction of the penetration rate for the posterior identification.

Another common way of improving the classification consists of joining different types of features of the image [28, 29, 26, 30]. Nevertheless, the processing of fused information also becomes more complex due to the larger amount of information that must be taken into account. This situation is typical for preprocessing methods [31], and more specifically feature selection algorithms [32, 33], which aim to simplify the data by eliminating noisy or redundant information. Such techniques have already been successfully applied to the problem of fingerprint classification [34].

To overcome these problems, we propose a complete identification system that includes a hierarchical classification framework that fuses the information of multiple feature extractors. The classification accuracy is further improved by incorporating a feature selection step. With this method we aim to maximize the reduction of the penetration rate via classification, without rejecting fingerprints whose feature extraction may have been affected by low-quality impressions. The proposed AFIS is designed to efficiently and accurately perform identifications in large fingerprint databases. The process is split into two steps:

- First, a novel hierarchical classification framework combines different sets of features and classifiers, organized into layers, depending on which feature extractors reject a certain fingerprint, eliminating the rejection whilst still benefiting from the high accuracy provided by feature extractors with rejection. Furthermore, a feature selection process is applied to eliminate redundancies and maximize the accuracy within each layer. The final outcome of the hierarchical classifier for a given an input fingerprint is an ordering of the classes according to the probability of its membership.

- Then, a distributed identification is performed, which explores the different classes in the order given by the classifier to obtain an optimized trade-off between runtime and accuracy.

The AFIS described in this paper is evaluated on three different databases: NIST-SD4, NIST-SD14 and a large synthetic database. Different feature extractors and classifiers from the scientific literature were combined to measure their accuracy, along with that of the method proposed in the current work. The impact of the classification on the posterior identification process is also tested in terms of identification accuracy and runtime.

This paper is organized as follows. Section 2 presents previous work on fingerprint recognition and penetration rate reduction. Section 3 describes the proposed hierarchical classifier and identification procedure. Section 4 details the performed experiments and analyzes the obtained results. Finally, Section 5 offers the conclusions of the study.

2. Background

The fingerprint recognition problem is described in Section 2.1. Then, Section 2.2 carries out an overview of high performance computing for fingerprint recognition. Finally, Section 2.3 presents the approaches in the current literature to reduce the penetration rate of the search in fingerprint identification.

2.1. Fingerprint recognition

Some of the properties that make fingerprints suitable for recognition purposes have been known for more than a century [24]. However, it is in the last decades, with the development of Automatic Fingerprint Identification Systems, that the full potential of fingerprints is being exploited. They are essentially patterns of ridges and valleys, from which different kinds of features can be extracted to automate the recognition process [20]. Minutiae (bifurcations and ridge endings) are the most widely used features for identification, due to their capabilities to efficiently discern fingerprints [35].

As for the variants of the fingerprint recognition problem, identification is inherently more complex than verification, since it involves searching for an input fingerprint throughout a database of n template fingerprints. This complexity increases along with the size of the database. As n grows, the number of comparisons that must be performed increases linearly, and so does the identification time. The accuracy of the identification is also degraded due to the higher probability of false positives. The identification time and accuracy are tightly coupled objectives: very precise matching algorithms are usually computationally expensive, and faster approaches are less accurate [35]. Therefore, any AFIS needs to find a trade-off between accuracy and identification time.

2.2. High Performance Computing for fingerprint identification

HPC is a common tool to speed up computationally intensive tasks. Both multi-CPU and GPU systems have been employed to optimize the fingerprint identification process [4, 36, 17, 18]. However, carrying out an effective and efficient application of HPC resources over such a problem is not straightforward. The database must be correctly distributed among the underlying hardware for a maximum speedup, and the parallel processes must be efficiently synchronized to minimize the overhead.

The use of a penetration reduction procedure increases the complexity of this task: each part of the database that can be potentially explored in parallel should be evenly distributed among the processors for a maximum performance [37]. Additionally, the processing of such parts should be performed so as to minimize the required synchronization.

2.3. Approaches for the reduction of the database penetration rate

Another way to speed up the identification time is by reducing the penetration rate of the search, which is the ratio between the number of fingerprints explored for the identification and the total size of the database [19]. There are two main approaches for this purpose: indexing (Section 2.3.1) and classification (Section 2.3.2).

2.3.1. Indexing

Indexing methods can be based on various features, such as minutiae [22, 38], orientations [39] or ridge frequencies [40], and usually do not consider rejecting fingerprints. They transform each fingerprint into a point in some multi-dimensional space, in such a way that several impressions of the same fingerprint ideally become very close points. Therefore, the input fingerprint is only compared with templates that fall close to that point. The search area (or the number of nearest neighbors considered) controls the trade-off between accuracy and identification time: a larger area will more likely contain the searched template, but will be more expensive to explore. The main problem of indexing approaches is the degradation of the accuracy when the penetration rate is too small or the collision rate in the multi-dimensional space is too high [22]. Such difficulties increase with large databases because of the higher density of points in the target search space.

2.3.2. Classification

Classification is the most common approach for penetration rate reduction [21]. Most authors use the five classes proposed by Henry [24] (Fig. 1), which can be visually discerned according to their ridge pattern. Some methods perform a fixed classification, based on a set of rules that determine the class without explicitly training a classification model [41, 42]. However, most proposals in the literature first encode some features as a numerical vector, which is then used to train a classifier. Some authors propose their own specific classifier for this step, but general purpose models (such as neural networks or support vector machines) are commonly applied. The reader may refer to [21, 23] for a wide review of feature extraction and fingerprint classification methods.

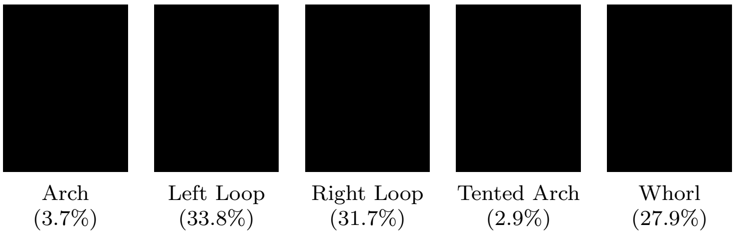


Figure 1: Five fingerprint classes defined by Henry [24] and their frequencies.

In general and with few exceptions [43], classification is built upon global level fingerprint features due to their low correlation with minutiae [20, 21]:

- Orientation map: each position contains the average direction of a block of pixels in the image [44, 45].
- Singular points (cores and deltas): points around which the patterns of the image are organized [41, 46, 29, 47].
- Ridge structure: these methods consider the shape and disposition of the ridges (ridge tracing) [48, 49, 50, 51].
- Filters: many methods apply filters on the fingerprint image. The reference method in this point is FingerCode [25], which starts by locating the core of the image using the Poincaré method [41]. Then, a circle (divided into sectors) is centered 40 pixels below the core and Gabor's filters are applied over the cropped image to enhance the ridges along four different directions. Finally, the standard

deviation of the gray levels of each filtered image is computed for each sector of the circle. When the reference core point cannot be found or is too close to the image border to fit the circle, the fingerprint is rejected. Many methods in the literature are based on FingerCode due to its efficiency and the high accuracy it provides [26, 52, 27], although most of them also inherit the rejection problem.

Table 1: Feature extraction methods

Name	**Features**	**Number of features**	**Rejection**	**Ref.**
Cappelli	Orientations	357	No	[34]
Leung	Gabor	816	No	[52]
Li	Singular points and orientations	58	No	[45]
Liu	Singular points	64	No	[47]
Nyongesa	Singular points and orientations	7	No	[29]
Cao	Orientations and ridge tracing	Variable	No	[30]
Hong	Singular points, Gabor and ridge tracing	202	Yes	[26]
Le	Gabor	1280	Yes	[27]

Several state-of-the-art feature extractors (summarized in Table 1) have been implemented for the experiments in this paper. Cappelli et al. [34] use the orientation map, registered by means of the Poincaré method [41]. It applies a dynamic mask for each class to obtain a vector with five features. Then, the orientations are stored into the feature vector in radians. Leung and Leung [52] describe a modification of FingerCode with an increased number of features and based on the R92 [53] registration method, instead of Poincaré. Furthermore, Leung's method does not reject any fingerprint; instead, the areas of the filter that fall outside of the image are marked as missing. Li et al. [45] combine singular points and orientations, extracted with complex filters and a multi-scale model respectively. Liu [47] proposes an approach based solely on relative measures between the singular points. Nyongesa et al. [29] build a vector from the angles between the singular points, complemented with the global orientation of the image. Cao et al. propose a hybrid approach in [30], which merges a fixed classification scheme with some steps involving machine learning procedures. This method builds an orientation map and evaluates successive fixed rules to determine if the fingerprint belongs to classes Arch or Whorl. In a negative case, a k-nearest neighbors classifier [54] determines the two most probable classes, one of which is selected by ridge tracing or support vector machines. Hong et al. [26] propose an extension of the FingerCode feature vector [25] with the number of cores and deltas, the distance and location between them and the pseudo-ridges traced from the center of the fingerprint. Le and Van [27] present another method based on FingerCode, also with more features, with the difference that it is invariant to rotations of the fingerprint.

3. Distributed incremental search with hierarchical classification framework

In this paper we propose a complete AFIS composed by a hierarchical classification method (Section 3.1) and a distributed identification with incremental search (Section 3.2) to efficiently cope with very large databases of fingerprints.

3.1. Hierarchical fingerprint classification with feature selection

The aim of any fingerprint classifier is to provide a high accuracy rate to narrow down the search space of the subsequent identification. The reduction or elimination of the feature extraction rejection is also an important objective. Therefore, the characteristics and the quality of the underlying feature extractors are critical. In our proposal, the problems that typically arise in such situations are dealt with in several manners:

- Classification accuracy: several feature extractors are combined to increase the precision, taking advantage of the larger amount of information that becomes available for the classifier.
- Rejection rate: if any of the fused feature extractors rejects a fingerprint, a different combination of extractors is used.

- Growth of the feature vector: as features are combined, the feature vector of a fingerprint grows, increasing the classification time and complicating the classification task. A feature selection procedure is therefore applied to obtain a representative subset of features.

The rest of this section describes the aforementioned parts of the proposal (feature fusion, hierarchical classification and feature selection), ending by a description of the overall classification procedure that integrates all of them.

Fusion of feature vectors

Suppose that m feature extractors are combined, each of which produces a feature vector $\mathbf{x}^{(i)} = \{x_1^{(i)}, ..., x_{D_i}^{(i)}\}$ $(i = 1, ..., m)$. The fusion process works as follows:

- If none of the feature extractors reject the fingerprint, all feature vectors $\mathbf{x}^{(i)}$ are concatenated as shown in Eq. 1, obtaining a vector $\mathbf{x}$ of size $D = \sum_{i=1}^{m} D_i$.

$$\mathbf{x} = \{x_1^{(1)}, ..., x_{D_1}^{(1)}, ..., x_1^{(i)}, ..., x_{D_i}^{(i)}, ..., x_1^{(m)}, ..., x_{D_m}^{(m)}\} \tag{1}$$

- If any feature extractor rejects the fingerprint, no feature vector is created and the fingerprint is rejected.

Hierarchical classification

By design, the rejection rate of the described fusion process is at least as large as the maximum rejection rate of the m underlying methods. This contributes to a very high accuracy on the non-rejected fingerprints, but the rejection rate would hinder the penetration rate. Therefore, we propose a hierarchical structure where each level combines a different set of feature extractors. Given a fingerprint, the features for the first level are extracted. If the extraction is successful, the extracted features are used to classify the fingerprint. On the contrary, if the fingerprint is rejected by any of the feature extractors in this level, the feature extractors of the second level are applied in turn. The process goes on until some level does not reject the fingerprint or the last level of the hierarchy is reached.

As a general guideline, the first level of the hierarchy should involve feature extractors with a high rejection rate and very high accuracy. Each subsequent level would contain extractors with decreasing rejection rates, until the last level is exclusively composed by extractors with no rejection. In this manner, the entire hierarchical structure works as a single classifier with no rejection that takes advantage from the high accuracy of the methods that do reject fingerprints. Note that although henceforth we will focus on a classifier with two levels, the proposal is generic and can be extended to any number of levels.

Feature selection

The hierarchical structure with fusion described so far ensures that no fingerprint is rejected, and that a large amount of information is available to train the classifier in each level of the hierarchy. However, the vectors of different feature extractors often contain overlapping information. Feature selection methods [32] are an adequate method to reduce the size of these fused vectors by removing redundant components. Although any feature selection method would be applicable, in our implementation we applied the embedded multivariate feature selection procedure described in [55]. This method trains a Random Forest classifier [56] and sums the information gain in each split over all the decision trees, obtaining a vector with a weight for each feature. The final feature set is composed of the D' features with the highest weights, which enables a very flexible use of the feature selection procedure as D' can be chosen at will.

Overall classification procedure

Each level of the hierarchy is based on different features, and therefore is trained independently. The training procedure is depicted in Fig. 2, which represents a case with four feature extractors, one of which (Hong) has a non-zero rejection rate. First, all the considered feature extractors are applied to the fingerprints in the training set. For each level, the corresponding features are fused and the fingerprints that are rejected by any of the underlying extractors are discarded. The features are filtered by the chosen feature selection algorithm. Finally, a classifier is trained using the filtered data. At the end of the training process, the global hierarchical model is composed by one classifier for each layer.

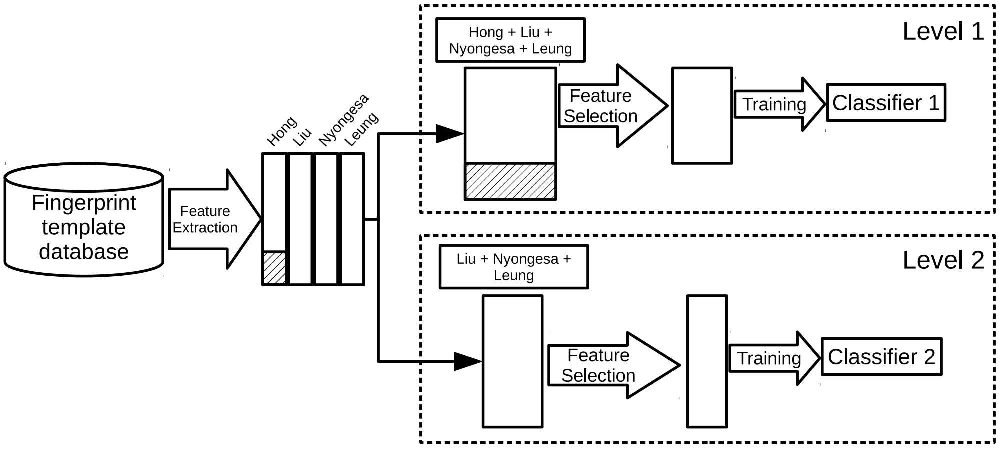


Figure 2: Training of the hierarchical classifier for four feature extractors (selected from those in Table 1) and two levels. Note that the proposal is generic and both the number of levels and the feature extractors involved in each level might be arbitrarily chosen.

When an input fingerprint is received, the system first determines which level is applicable. In most cases it is not necessary to compute the entire feature extraction procedure to determine whether a fingerprint is rejected or not. Therefore, once the system determines which feature extractors will reject the input fingerprint, it can directly apply only the feature extractors that are involved in the applicable level. Then, the features are filtered and finally the corresponding classifier is applied, as depicted in Fig. 3.

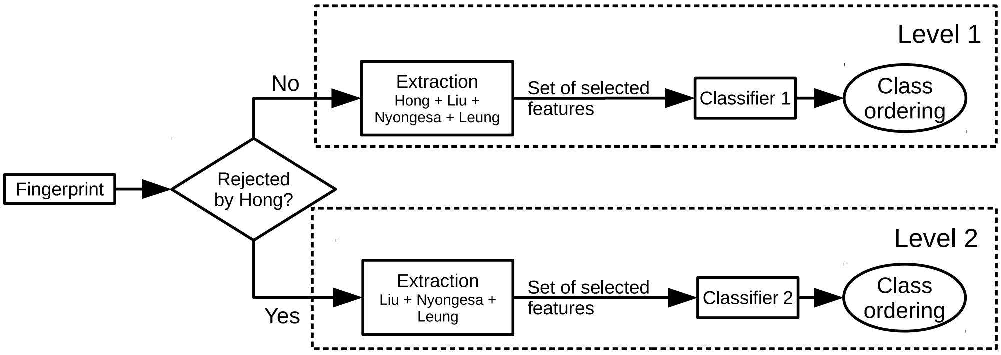


Figure 3: Application of the hierarchical classifier trained as shown in Fig. 2

Many classifiers in the literature allow us to compute an ordering of the classes according to their probability. We propose the use of this information to perform an incremental search (as described in Section 3.2). Two different classifiers are employed in this paper: Support Vector Machines (SVM) [57] and Random Forests [56]. Class probabilities for SVM are computed following the method proposed in [58]. For Random Forests, the class probabilities can be calculated from the proportion of votes for each class among the decision trees.

3.2. Parallel incremental identification with reduced penetration rate

The described classification sets up the starting point for an identification procedure with reduced penetration rate. In the context of a parallel identification system, however, there are several additional factors to take into account to maximize the performance gain obtained with this reduction. The system proposed in this paper is based upon the one described in [4], a two-level parallel framework with no classification and 100% penetration rate designed to be executed over a cluster of multi-core computers in a master-slave manner. The number of slave processes and their threads can be tuned to suit the needs of any particular configuration, and the database is accordingly distributed among the computing nodes.

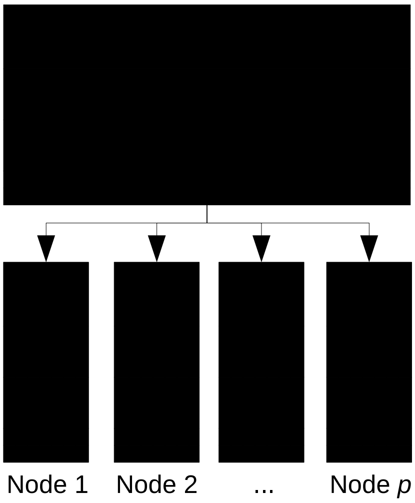


Figure 4: Database partitioning using classes. Each node is responsible for a partition of the database with the same class distribution, and in turn partitions it among its threads.

Fig. 4 shows the partitioning designed to maximize the parallelization of the identification search. Let $\mathcal{T} = \{T_1, ..., T_n\}$ be the complete template fingerprint database, where each T_i is a template fingerprint. Let $\mathcal{T}^c$ be the set of templates belonging to class c, so that the classes are disjoint and $\mathcal{T} = \bigcup\limits_c \mathcal{T}^c$. Let p be the number of slave processes, which is typically equal to the number of available computing nodes. Then each $\mathcal{T}^c$ is evenly split into p parts, so that a slave j is responsible for exploring its corresponding database portion $\mathcal{T}_j = \bigcup\limits_c \mathcal{T}_j^c$. Additionally, each $\mathcal{T}_j^c$ is further divided into logical sections that are explored in parallel by several threads within each node.

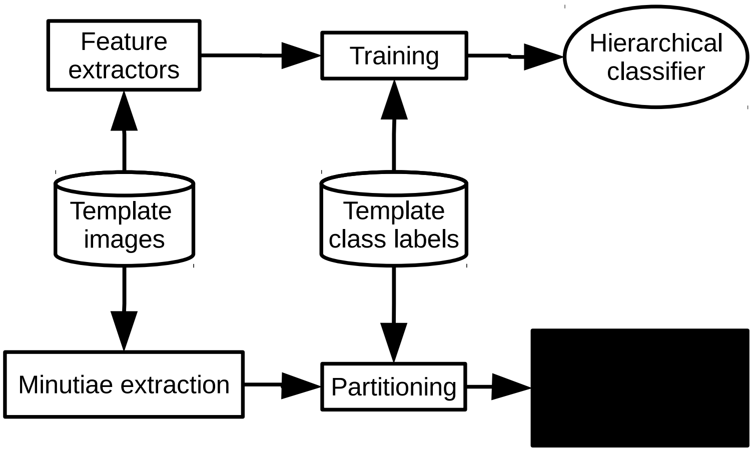


Figure 5: Setup of the identification process.

With this partitioning scheme, the exploration of each individual class can be carried out in parallel among all threads in all p nodes and each thread will have the same computational workload, thus maximizing the parallelism and minimizing the overhead. Fig. 5 shows the tasks that are performed for the identification system setup. The global features and the minutiae are extracted from the template fingerprints, and used to train the classifier and to build the partitioned template database, respectively.

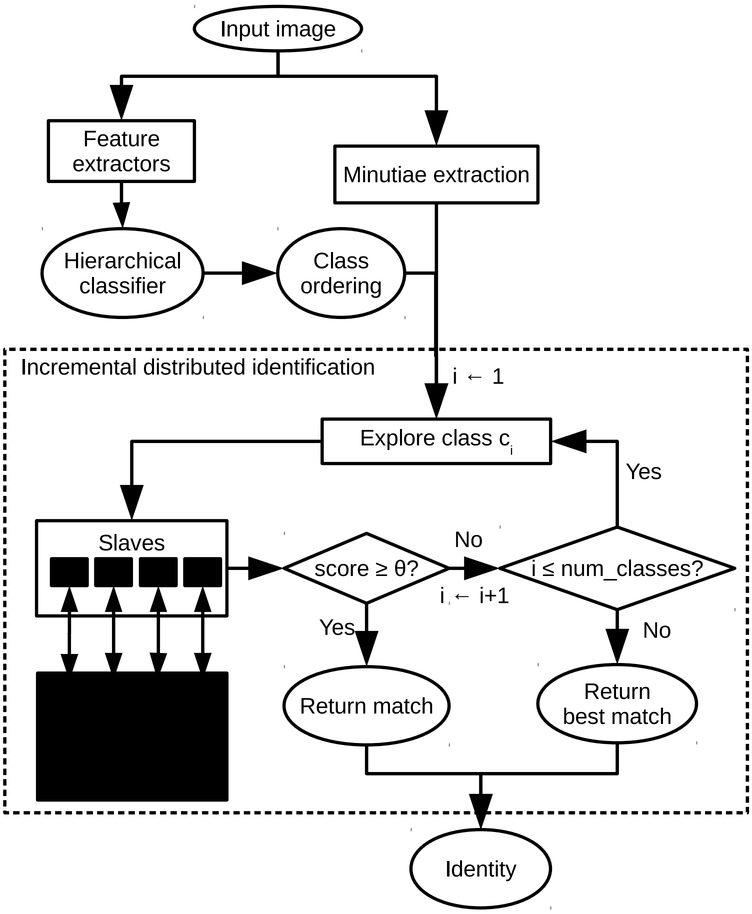


Figure 6: Identification workflow.

Fig. 6 depicts the global workflow of the proposal. The classes are explored following the order determined by the hierarchical classifier. If the best score obtained within a class is higher than a threshold θ, the corresponding fingerprint is returned as matched identity. Otherwise, the AFIS explores the next most probable class. If all the classes are explored and no score above θ is found, the match with the highest score among all classes is returned. This workflow has been implemented in a master-slave manner.

The master process (Algorithm 1) is responsible for extracting the minutiae and the global features of the input fingerprint I, and for classifying it. The minutiae set M and the class ordering are then broadcast to the slaves. For each class, the master collects the best scores found by the slaves, and determines if the ending criterion is met. A signal is broadcast to the slaves indicating whether the next most probable class should be explored or not. The loop goes on until a score higher than the threshold θ is found, or all classes have been explored. Finally, the best match found so far is returned.

Each slave process (Algorithm 2) starts with the reception of the minutiae and the class ordering from the master. For each class, the input minutiae are matched with the templates, and the best score is sent to the master. The slave explores the classes until the master sends the signal to stop.

This design ensures that a sufficiently high value of θ (namely $\theta = 1$, assuming matching scores in the range $[0, 1]$) yields an AFIS with no classification that returns the best match in the entire database. With $\theta = 0$, the system only explores the single most probable class for each input fingerprint, thus behaving as most previous approaches in the literature. The range of values $0 < \theta < 1$ enables a flexible environment, where the classes are explored until a score is found that is sufficiently high to ensure a high probability of genuine match. This allows us to tune the trade-off between penetration rate and accuracy with a single real parameter θ.

Algorithm 1: Identification procedure (master)

```
Input: $I$, $\theta$
$M \leftarrow$ I.extractMinutiae()
class_order $\leftarrow$ hierarchicalClassification(I)
MPIbroadcast($M$, class_order)
bestmatch $\leftarrow \emptyset$
finish $\leftarrow$ FALSE
$i \leftarrow 1$
while $i \leq$ length(class_order) and not finish do
    matches $\leftarrow$ getBestScoresFromSlaves()
    bestmatch $\leftarrow$ max(bestmatch, matches)
    if bestmatch.score() $\geq \theta$ then
        finish $\leftarrow$ TRUE
    end
    MPIbroadcast(finish)
    $i \leftarrow i + 1$
end
return bestmatch
```

Algorithm 2: Identification procedure (slave j)

```
Input: $\mathcal{T}_j$
MPIreceiveBroadcast($M$, class_order)
finish $\leftarrow$ FALSE
$i \leftarrow 1$
while $i \leq$ length(class_order) and not finish do
    bestmatch $\leftarrow \emptyset$
    $c \leftarrow$ class_order[$i$]
    foreach $T \in \mathcal{T}_j^c$ do
        match $\leftarrow$ matching(T, M)
        bestmatch $\leftarrow$ max(bestmatch, match)
    end
    sendToMaster(bestmatch)
    MPIreceiveBroadcast(finish)
    $i \leftarrow i + 1$
end
```

The identification time for each input fingerprint is given by $t_{fe} + t_c + np_r t_m$, where t_{fe} is the feature extraction time, t_c is the classification time, p_r is the penetration rate and t_m is the average matching time. In practice, the large magnitude of n causes the last term to dominate the formula, so that the complexity order becomes $O(np_r)$. When such a system is parallelized across p machines with q cores each, the lower bound of the identification time becomes $t_{fe} + t_c + np_r t_m/(pq)$ and the complexity becomes $O(np_r/(pq))$. Note that for an AFIS without any classification step, these expressions still hold with $t_c = 0$ and $p_r = 1$.

4. Experimental study

This section describes the experiments carried out to test the behavior of the proposed AFIS with hierarchical classification, along with an analysis of the results. First, Section 4.1 describes the datasets and algorithms used for the experiments. Then, Section 4.2 analyzes the results obtained in every step of the proposed hierarchical classifier, comparing them with those obtained by the reference methods from the literature. Section 4.3 provides a study on the database penetration rate reduction that can be estimated from the classification results. Finally, the results of the overall identification procedure in terms of identification time and accuracy, for each value of the threshold parameter, are evaluated in Section 4.4.

4.1. Datasets and algorithms

This section describes the fingerprint databases employed in the experiments, along with the algorithms that were selected to apply the feature extraction, minutiae extraction, matching and classification steps of the overall AFIS. Three different databases have been used for this work:

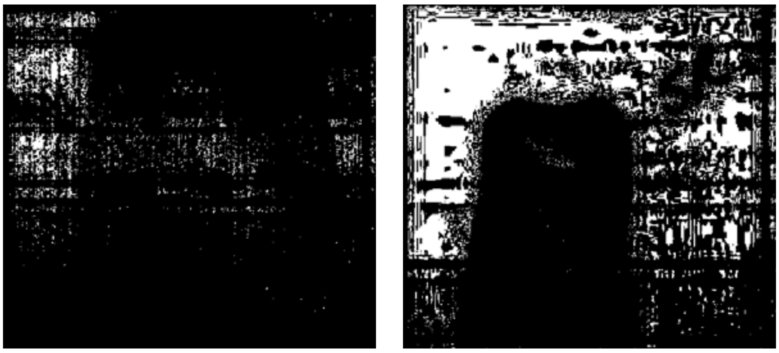
- SFinGe: artificial database generated by the SFinGe software [59, 20], using the parameters presented in Table 2 so as to include low quality fingerprints with heavy distortions, rotations and translations. A set of 10 000 fingerprints has been considered for the study of the classifiers (Section 4.2), while a large database composed of 3 impressions of 400 000 fingerprints was used for the identification experiments (Sections 4.3 and 4.4).

- NIST-SD4: this database is formed by two rolled ink impressions of 2000 manually classified fingerprints [60], uniformly distributed among the five classes. We segmented the images with he NIGOS *nfseg* algorithm [53]. A subset of 350 fingerprints are labeled with two different classes. Different authors handle this in different ways; for this study, we chose to follow the approach in [23], consisting of removing the doubly labeled fingerprints from the dataset, obtaining a final dataset of 1650 fingerprint pairs. This procedure avoids the necessity of replicating the fingerprints with two classes, which would hinder the identification process.

- NIST-SD14: it is formed by pairs of rolled ink impressions of 27 000 fingerprints, also manually classified [61] and segmented with *nfseg*. In some cases, the two impressions of the same fingerprint are labeled with different classes; in such situations (443 in total) the second impression has been relabeled with the class of the first impression. Note that 28 fingerprints of class "Scar" were removed for the experiments, as well as two fingerprints whose segmentation failed, leaving a total of 26 970 pairs (examples in Fig. 7).

Table 2: Parameter specification for SFinGe

Scanner parameters	**Generation parameters**	**Output settings**
Acquisition area: 14.6mm x 19.6mm.	Impressions per finger: 3.	Output file type: WSQ.
Resolution: 500 dpi.	Class distribution: Natural.	
Image size: 288 x 384.	Varying quality and perturbations.	
Background type: Optical.	Generate pores: enabled.	
Background noise: Default.	Save ISO templates: enabled.	
Crop borders: 0 x 0.		



(a) f0007289
failed segmentation

(b) s0002119
Scar class

Figure 7: Two examples of fingerprints discarded from NIST-SD14.

All the fingerprint images of the databases were stored in Wavelet Scalar Quantization (WSQ) format. The fingerprint global features for the classification were extracted using the state-of-the-art algorithms listed in Table 1 and described in Section 2.3.2, named by the first author's last name of their respective publications. The minutiae used for the matching during the identification were extracted using NIGOS *mindtct* [53]. Two different classifiers have been tested: Random Forest [56] and SVM [57], with the parameters shown in Table 3. The former was also used for the feature selection process, as described in [55]. Finally, the well-known Minutia-Cylinder-Code (MCC) [14] was selected as matching algorithm for the identification experiments. The authors used their own implementation of the algorithm, following the guidelines and parameters specified in the original publication. The software developed and executed for the experiments is publicly available from `http://sci2s.ugr.es/Fingerprint`.

4.2. Hierarchical classifier with feature selection

This study evaluates the behavior of the first part of the proposed AFIS: the hierarchical classifier. The classifier is compared with the proposals that obtained the best results in [23], which are listed in Table 1.

Table 3: Parameters of the classifiers

Algorithm	Parameters
Random Forest	Trees: 1000; Variables per split: $\sqrt{D'}$
SVM	RBF Kernel; $C = 1$; Tolerance = 0.001; Epsilon = 0.1; $\gamma = 1/D'$

The study focuses on two performance measures—classification accuracy and rejection rate—that have to be maximized and minimized, respectively. The experiments in this section have been performed on 10 000 fingerprints generated with SFinGe, following a 5-fold cross validation procedure.

Table 4 shows the average test accuracy and rejection rates of the tested feature extractors and classifiers as a starting point for the current study. Note that Cao's hierarchical approach uses SVM internally, so the Random Forest classifier was not applied in this case. The most accurate results with and without rejections are respectively bold-faced, indicating that Hong's algorithm was the most precise one, although with a significant rejection rate, whilst Liu's method obtained the best precision without rejections. Even though Le's proposal is also based on FingerCode, its accuracy was outperformed by other approaches. Cao's approach obtained a fair accuracy, but also lower than that obtained by other methods. These discrepancies with the results presented in their original publications can be caused by the higher difficulty of classifying plain fingerprints [20], and by the deliberately low quality of our SFinGe database.

Table 4: Accuracy and rejection rates for each feature extractor and classifier

Feature Extractor	Random Forest	SVM-RBF	Rejection
Cappelli	0.8908	0.8494	0.00%
Leung	0.9027	0.9064	0.00%
Li	0.7981	0.6763	0.00%
Liu	**0.9259**	0.9130	0.00%
Nyongesa	0.7033	0.7353	0.00%
Cao	–	0.8651	0.00%
Hong	0.9359	**0.9490**	15.90%
Le	0.7755	0.7946	15.22%

Table 5 shows the accuracy obtained by fusing different feature vectors with and without rejection, limiting the results to the best combinations in each case for the sake of simplicity. Besides an increase of the accuracy, it can be seen that in general combinations that are affected by rejection got a higher precision with the SVM classifier, while combinations without rejection performed better with Random Forest.

Table 5: Accuracy and rejection for the best combinations of feature extractors. The best accuracy is bold-stressed for each combination.

Combination of extractors	Random Forest	SVM-RBF	Rejection
Hong-Liu-Nyongesa-Leung	0.9490	**0.9568**	15.90%
Hong-Liu-Leung	0.9486	**0.9564**	15.90%
Hong-Liu-Leung-Li	0.9493	**0.9549**	15.90%
Hong-Liu-Nyongesa-Leung-Li	0.9505	**0.9549**	15.90%
Hong-Liu-Nyongesa	0.9457	**0.9540**	15.90%
Hong-Liu	0.9452	**0.9539**	15.90%
Hong-Liu-Cappelli-Li	**0.9534**	0.9384	15.90%
Liu-Nyongesa-Leung	0.9424	**0.9462**	0.00%
Liu-Cappelli-Leung-Li	**0.9456**	0.9394	0.00%
Liu-Cappelli-Nyongesa-Leung-Li	**0.9454**	0.9400	0.00%
Liu-Leung	0.9419	**0.9454**	0.00%
Liu-Cappelli-Nyongesa-Li	**0.9448**	0.9193	0.00%
Liu-Cappelli-Nyongesa-Leung	**0.9444**	0.9418	0.00%
Liu-Cappelli-Leung	**0.9441**	0.9416	0.00%

Once the different features have been combined, the embedded feature selection method (based on Random Forests) is applied to reduce the size of the feature vectors and increase the accuracy. Fig. 8 depicts the behavior of the classification accuracy according to the number of features selected from several combinations of feature extractors. The figure shows that the accuracy of Random Forest was only slightly increased by the feature selection procedure. This is caused by the implicit behavior of Random Forests, which perform the splits within each tree according to the feature with the best information gain. This defines a selection of the features, which in fact constitutes the basis for the applied embedded feature selection mechanism in our proposal. On the other hand, the feature selection improved the accuracy of SVM. The case that combines all feature extractors without rejection is especially illustrative, as the best accuracy obtained needed only 20% of the features and outperformed the Liu-Nyongesa-Leung combination, which yielded a better accuracy when all features were used.

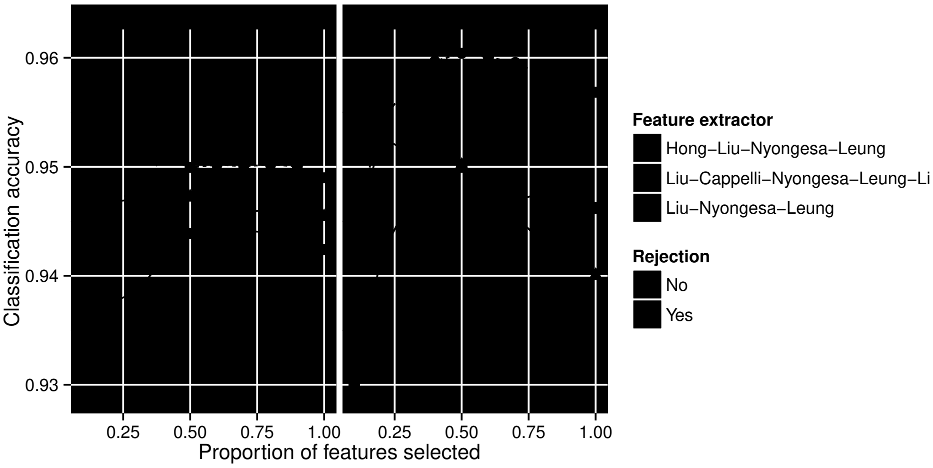


Figure 8: Classification accuracy when using feature selection

Finally, two different variants of the complete hierarchical procedure were tested. Both cases involved a two-level hierarchy: the first level contained the best performing combination with rejection (Hong-Liu-Nyongesa-Leung) after selecting 50% of the features, whilst two different combinations of feature extractors were inserted into the second level, both selecting 60% of features. We only consider the results of the SVM classifier, which performs significantly better than Random Forests after the application of the feature selection step. The results in Table 6 show that the proposal reached an accuracy rate of 95.40% with no rejection at all, which was better than any of the reference results shown in Table 4.

Table 6: Accuracy of the two variants tested for the hierarchical classifier with SVM (0% rejection)

	Option A	**Option B**
Level 1	Hong-Liu-Nyongesa-Leung	Hong-Liu-Nyongesa-Leung
Level 2	Liu-Nyongesa-Leung	Liu-Cappelli-Nyongesa-Leung-Li
Hierarchical classification (no Feature selection)	0.9497	0.9502
Hierarchical classification (with Feature selection)	0.9530	0.9540

The results of the study carried out so far assess that the proposed hierarchical classifier is able to outperform the state-of-the-art approaches, both in terms of classification accuracy and rejection rate. The fusion of feature extractors, in combination with an adequate feature selection, increases the accuracy of the classification while avoiding an overgrowth of the number of features. Finally, the hierarchical structure enables the elimination of the rejection rate, while maintaining the high accuracy provided by the underlying feature combinations.

4.3. Database penetration rate reduction

Once the classifier is trained, it is possible to estimate the penetration reduction from the distribution of the fingerprint classes, the classifier confusion matrix and its rejection rate [44, 62]. Additionally, to estimate the penetration of the proposed incremental search, it is necessary to take into account the ordering of the classes obtained from the classification. This section performs a preliminary study of the expected performance on the tested databases in terms of penetration rate reduction:

- The classifiers for SFinGe were trained using 10 000 fingerprints (different from the ones used in the previous section, to avoid any bias) and were tested over 3 impressions of 400 000 fingerprints.
- The classifiers for NIST-SD4 and NIST-SD14 were trained using the first half of each database: both impressions of 825 and 13 485 fingerprints, respectively. The remaining fingerprints were used to compute the statistics shown in the subsequent tables.

Table 7 indicates how often the real class is obtained in each position of the class ordering for each database, when the proposed hierarchical classifier is applied. It is observed that in most of the misclassified fingerprints the actual class is the second of the options provided by the classifier. The decreasing trend shown in the table highlights the incremental search as a very promising way to reduce the penetration rate while maintaining a high accuracy, because the classification errors can be corrected by the identification process.

Table 7: Distribution of the position of the correct class in the class ordering.

Position	**SFinGe**	**NIST-SD4**	**NIST-SD14**
1st	94.38%	92.97%	93.76%
2nd	4.27%	5.82%	5.02%
3rd	1.02%	0.91%	0.80%
4th	0.27%	0.30%	0.31%
5th	0.05%	0.00%	0.11%

We define two measures for the evaluation of the expected penetration rate:

- Ideal penetration rate: the penetration rate that could be achieved with a perfect classifier (100% accuracy and no rejection) and a perfect identification (always succeeds in finding the correct fingerprint within the class). Thus, it is the minimum penetration required to attain a 100% True Positive Rate (TPR). As it assumes a perfect classifier, this measure depends solely on the class distribution of the dataset and the rejection of the feature extraction.
- Optimal penetration rate: the penetration achieved with a trained classification model, in combination with a perfect identification. This estimation also depends on the type of search of the identification:
  - Basic search: the most probable class is explored first. If the fingerprint is not found, all other classes are searched.
  - Incremental search: the classes are explored in the order given by the classifier, as described in Section 3.2, until the match is found.

  . The optimal penetration rate can be computed from the confusion matrix of the classification, weighted by the frequencies of each class in the dataset, and the class probabilities for each fingerprint.

Table 8 shows the values of these measures obtained when applying SVM in combination with each of the listed feature extraction methods. Note that Cao's method includes the classification as well as the feature extraction and does not consider the computation of the class probabilities. A reduction in the penetration rate is observed for all considered classifiers and databases. It can also be seen that rejections cause an increase of the ideal penetration rate, because the entire database must be explored in such cases. Although this behavior intends to reduce identification errors, it hinders the performance of accurate identification algorithms. In consequence, feature extractors without rejection obtain lower ideal penetration rates, and also lower estimated penetration rates with the basic search. Furthermore, the difference between them is substantially reduced when the incremental search is introduced. Finally, the table shows that the proposed hierarchical classifier yields the minimum penetration rate for all three tested databases, ratifying the good performance obtained in Section 4.2.

An additional experimental study has also been carried out to evaluate the impact of the image segmentation that is usually applied on NIST images before the extraction of their features. Table 9 shows

Table 8: Estimated penetration rates for each feature extractor and database. The best values for each database are bold-stressed.

	Ideal penetration rate			Optimal PR (basic search)			Optimal PR (incremental search)		
Feature Extractor	**SFinGe**	**NIST-SD4**	**NIST-SD14**	**SFinGe**	**NIST-SD4**	**NIST-SD14**	**SFinGe**	**NIST-SD4**	**NIST-SD14**
Cappelli	0.2948	0.2176	0.3105	0.3850	0.3444	0.4167	0.3444	0.2667	0.3684
Leung	0.2948	0.2176	0.3105	0.3394	0.2849	0.3823	0.3255	0.2370	0.3430
Li	0.2948	0.2176	0.3105	0.5875	0.5198	0.5744	0.4824	0.3738	0.4878
Liu	0.2948	0.2176	0.3105	0.3728	0.3578	0.4237	0.3354	0.2761	0.3729
Nyongesa	0.2948	0.2176	0.3105	0.4919	0.4251	0.4447	0.4405	0.3185	0.3853
Cao	0.2948	0.2176	0.3105	0.3980	0.4773	0.4790	–	–	–
Hong	0.3970	0.2313	0.3644	0.5122	0.3346	0.4607	0.4111	0.2635	0.3864
Le	0.4354	0.2745	0.3812	0.6160	0.4861	0.6994	0.4809	0.3471	0.5647
Proposal	0.2948	0.2176	0.3105	**0.3356**	**0.2748**	**0.3558**	**0.3102**	**0.2339**	**0.3287**

the estimated penetration rates in NIST-SD14 for all considered algorithms, respectively with and without the application of the *nfseg* segmentation algorithm. It can be observed that the segmentation increases the performance of most feature extractors. The proposed hierarchical algorithm again outperformed all others in terms of both classification accuracy and estimated penetration rate when no segmentation is applied. These results demonstrate the robustness of the proposal: it obtains a high accuracy even from the non-preprocessed input images, which is further increased when the segmentation algorithm is used.

Table 9: Results on the NIST-D14 database, with and without the application of the *nfseg* segmentation algorithm

	Classification accuracy		Optimal PR (basic search)		Optimal PR (incremental search)	
Feature Extractor	**Segmented**	**Not segmented**	**Segmented**	**Not segmented**	**Segmented**	**Not segmented**
Cappelli	0.8511	0.8547	0.4167	0.4145	0.3684	0.3628
Leung	0.8970	0.8842	0.3823	0.3911	0.3430	0.3485
Li	0.6214	0.5601	0.5744	0.6169	0.4878	0.5206
Liu	0.8429	0.8187	0.4237	0.4402	0.3728	0.3833
Nyongesa	0.8123	0.7350	0.4447	0.4985	0.3853	0.4340
Cao	0.7676	0.7366	0.4790	0.5018	–	–
Hong	0.8540	0.7834	0.4607	0.5184	0.3864	0.4243
Le	0.4871	0.6546	0.6994	0.6010	0.5647	0.4812
Proposal	**0.9376**	**0.9189**	**0.3558**	**0.3687**	**0.3287**	**0.3359**

4.4. Parallel identification with reduced penetration rate

This section presents the results of the complete identification process, including the classification, with the aim of verifying the impact of the hierarchical classification on the the identification accuracy and penetration rate. For SFinGe, the template database was composed of a single impression of 400 000 fingerprints, and the input set by a different impression of 3000 fingerprints. For NIST-SD4 and NIST-SD14, the first impression of all fingerprints composed the template database, and respectively 1650 and 3000 second impressions where used as input.

All experiments were executed on a cluster of 12 computing nodes, each containing 2 Intel(R) Xeon(R) CPU E5-2620 processors (6 cores at 2.00GHz each) and 64GB RAM, connected by a QDR InfiniBand network (40Gbps). The master node contains the same processors and 32GB RAM. The number of parallel threads within each slave node was 24 (2 threads per core). Only 2 nodes were used for NIST-SD4 due to its smaller size.

The identification accuracy measures used in this section are the number of true positives (TP) and false positives (FP). These values are also summarized into the TPR for graphical representations. Table 10 shows the results obtained by the AFIS without applying the classification step, with 100% penetration. These results establish the reference for the comparison with those that will be described henceforth.

Table 10: Results of the AFIS without the hierarchical classification step.

Database	Templates	TP	FP	Avg. identif. time (s)
SFinGe	400 000	2842	158	5.1931
NIST-SD4	1650	1485	165	1.1279
NIST-SD14	26 970	2436	564	4.9228

Table 11 shows the results when the classification was used, for various values of the threshold θ. Fig. 9 depicts the TPR and the average identification time with respect to the average penetration rate for each

Table 11: Results of the AFIS with the hierarchical classification step.

Dataset	θ	TP	FP	Avg. identif. time (s)	Avg. penetration	Avg. clasif. time (s)
SFinGe	0.000	2708	292	1.5613	0.2915	0.0395
	0.020	2708	292	1.5608	0.2915	0.0382
	0.040	2741	259	1.6161	0.2972	0.0400
	0.050	2806	194	1.6869	0.3159	0.0396
	0.060	2835	165	1.8330	0.3435	0.0401
	0.070	**2842**	**158**	**2.0077**	**0.3783**	**0.0394**
	0.080	2843	157	2.1599	0.4091	0.0400
	0.090	2842	158	2.3599	0.4477	0.0393
	0.150	2842	158	3.7196	0.7120	0.0407
	0.250	2842	158	5.0482	0.9670	0.0404
NIST-SD4	0.000	1424	226	0.3224	0.2223	0.0139
	0.020	1428	222	0.3344	0.2231	0.0142
	0.022	1463	187	0.3700	0.2561	0.0147
	0.024	1479	171	0.4374	0.3052	0.0136
	0.026	**1484**	**166**	**0.4929**	**0.3470**	**0.0137**
	0.028	1483	167	0.5301	0.3759	0.0147
	0.040	1483	167	0.7649	0.5666	0.0146
	0.070	1483	167	1.0904	0.8712	0.0138
	0.100	1483	167	1.1898	0.9683	0.0134
NIST-SD14	0.000	2341	659	1.5490	0.2959	0.0685
	0.020	2343	657	1.5500	0.2959	0.0660
	0.022	2375	625	1.6335	0.3102	0.0688
	0.024	2401	599	2.0567	0.3744	0.0688
	0.026	2425	575	2.4848	0.4533	0.0685
	0.028	**2435**	**565**	**2.7600**	**0.5027**	**0.0714**
	0.030	2438	562	3.0123	0.5481	0.0670
	0.050	2436	564	4.8354	0.8544	0.0666

database, along with those obtained without classification. The relation between penetration rate and identification time is clear: the latter decreases linearly as the former is reduced. The very low classification time is a key factor to attain this behavior. The decrease of the identification time with respect to that shown in Table 8 was two orders of magnitude higher than the time invested in the classification, highlighting the suitability of the proposal.

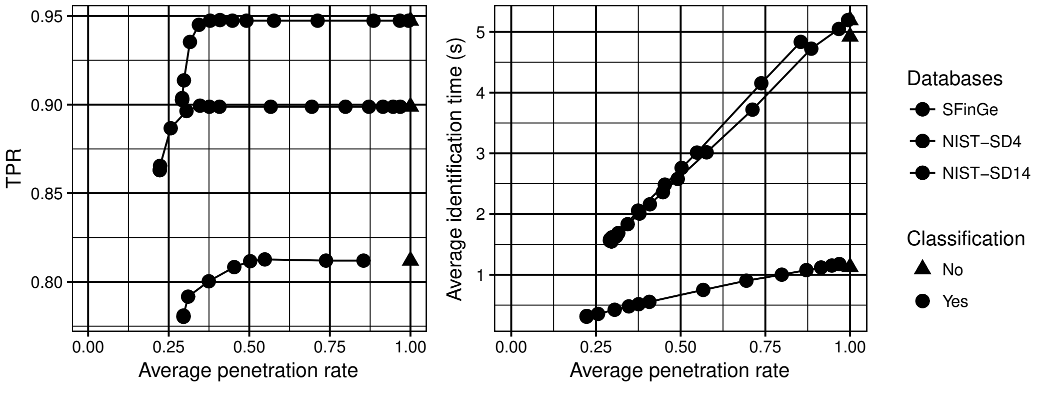


Figure 9: Classification accuracy and average identification time for each penetration rate

It is noteworthy that both SFinGe and NIST-SD14 present a peculiar behavior: the penetration rate obtained with $\theta = 0$ was lower than the ideal penetration rate presented in Table 8. Although this might seem counter-intuitive, it is a consequence of the classification errors: a fingerprint belonging to a large class (such as Left, Right or Whorl) that is misclassified as one of the small classes (Arch or Tented) may be wrongly identified, but the penetration rate is very small. This kind of behavior is very desirable in such a system, because the performance impact of such misclassifications is low.

As for the identification accuracy, for any of the tested databases a threshold can be found that yields the same TP as an AFIS without classification, with a much lower average penetration, establishing a trade-off that should be suitable for most practical systems. Such thresholds are bold-stressed in Table 11. Different thresholds can be applied in order to obtain either faster or more robust results, in each case at the expense of the other objective.

As a complementary note to conclude this study, Table 12 shows the average identification time obtained by other HPC-based proposals in the literature, most of them using GPU infrastructures. Note that each paper performed its study with different hardware, fingerprints and matching algorithms; therefore the presented times are mostly orientative. However, the table shows that the average times obtained by our proposal are competitive with the state-of-the-art, despite the higher number of minutiae of the fingerprints used (with respect to which the matchers are of at least quadratic complexity) and the number of template fingerprints in the database.

Table 12: Size of the databases and average identification time obtained by other HPC-supported AFIS.

	Hardware	**Average minutiae**	**Templates**	**Cores**	**Matcher**	**Time (s)**
Le et al. [63]	NVIDIA GeForce GTX 680	30	200 000	1536	MCC (LSS, $N_s = 8$)	1.1050
Peralta et al [4]	12 × 2 × Intel(R) Xeon(R) CPU E5-2620	55.5	400 000	144	Jiang [64]	1.0368
					Chen [65]	10.4387
					MCC (LSSR, $N_s = 16$)	20.6486
Gutiérrez et al. [36]	2 × Nvidia Tesla M2090 GPUs	40.7	100 000	1024	MCC (LSS, $N_s = 8$)	1.0240
					MCC (LSS, $N_s = 16$)	2.3939
					MCC (LSSR, $N_s = 8$)	1.8395
					MCC (LSSR, $N_s = 16$)	3.2886
Lastra et al. [18]	2 × Nvidia Tesla K20m 2 × Nvidia Tesla M2090	51.8	800 000	6016	Jiang [64]	0.5333
Cappelli et al. [17]	4 × Tesla C2075 GPUs	32.3	250 000	1792	MCC (LSS, $N_s = 8$)	0.0071
Proposal	12 × 2 × Intel(R) Xeon(R) CPU E5-2620	55.5	400 000	144	MCC (LSSR, $N_s = 8$)	2.0077

5. Conclusion

Fingerprint classification is a widely extended technique to reduce the penetration rate of the search in fingerprint identification systems. This reduction enables a faster identification process by limiting the number of template fingerprints that are compared with the searched input fingerprint. Feature extraction methods are a central part of such classification approaches; each algorithm obtains a different set of features from the fingerprint image and encodes them as a numerical vector. Some of these feature extractors reject fingerprints that do not comply with certain properties, increasing the accuracy of the classification for non-rejected fingerprints, but hindering the reduction of the penetration rate for those that are rejected.

This paper describes a hierarchical classification model, where each level combines several feature extractors into a more complete feature vector. A feature selection algorithm is then applied to eliminate noise and redundancies. The hierarchy allows for rejected fingerprints to be processed by subsequent levels, maintaining a high accuracy while completely eliminating the rejection. Furthermore, the hierarchical classification returns an ordering of the classes that can be used to perform an incremental search within a parallel environment. The template database must be conveniently partitioned among the computing nodes to maintain a balanced distribution of the classes. When a fingerprint is to be identified, the most probable class is explored; if it is not found with a sufficient confidence level the second class is explored, etc. The confidence level is determined by a matching score threshold, a single parameter that sets the trade-off between accuracy and identification time.

The results obtained over several databases highlight the very good classification accuracy obtained by the proposal, while eliminating the rejection rate. Additionally, the study performed on the identification process with incremental search demonstrates the benefits of the classification and reveal that the obtained penetration rate was very close to the optimum that can be attained.

Acknowledgments

This work was supported by the research projects TIN2014-57251-P, TIN2013-47210-P and P12-TIC-2958. Y. Saeys is an ISAC Marylou Ingram Scholar.

References

[1] R. Singh, A. Ross, K. W. Bowyer, Special issue on information fusion in biometrics, Information Fusion 32 (2016) 1–2.

[2] S. Jabin, F. J. Zareen, Biometric signature verification, International Journal of Biometrics 7 (2) (2015) 97.

[3] Z. Zhang, L. Wang, Q. Zhu, S.-K. Chen, Y. Chen, Pose-invariant face recognition using facial landmarks and Weber local descriptor, Knowledge-Based Systems 84 (2015) 78–88.

[4] D. Peralta, I. Triguero, R. Sanchez-Reillo, F. Herrera, J. M. Benitez, Fast Fingerprint Identification for Large Databases, Pattern Recognition 47 (2) (2014) 588–602.

[5] Y. J. Chin, T. S. Ong, A. B. J. Teoh, K. O. M. Goh, Integrated biometrics template protection technique based on fingerprint and palmprint feature-level fusion, Information Fusion 18 (1) (2014) 161–174.

[6] A. Kumar, A. Kumar, Adaptive management of multimodal biometrics fusion using ant colony optimization, Information Fusion 32 (2016) 49–63.

[7] D. Menotti, G. Chiachia, A. Pinto, W. Robson Schwartz, H. Pedrini, A. Xavier Falcao, A. Rocha, Deep Representations for Iris, Face, and Fingerprint Spoofing Detection, IEEE Transactions on Information Forensics and Security 10 (4) (2015) 864–879.

[8] A. K. Jain, A. A. Ross, K. Nandakumar, Introduction to Biometrics, Springer, New York, 2011.

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

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

[11] D. Maio, D. Maltoni, R. Cappelli, J. L. Wayman, A. K. Jain, FVC2000: fingerprint verification competition, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (3) (2002) 402–412.

[12] D. Maio, D. Maltoni, R. Cappelli, J. L. Wayman, A. K. Jain, FVC2002: Second fingerprint verification competition, in: Proceedings - International Conference on Pattern Recognition, Vol. 16, 2002, pp. 811–814.

[13] D. Maio, D. Maltoni, R. Cappelli, J. L. Wayman, A. K. Jain, FVC2004: Third fingerprint verification competition, Lecture Notes in Computer Science 3072 (2004) 1–7.

[14] 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 (12) (2010) 2128–2141.

[15] D. Peralta, I. Triguero, S. García, F. Herrera, J. M. Benitez, 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.

[16] H. S. Stone, High-performance computer architecture, Addison-Wesley, Reading, USA, 1992.

[17] R. Cappelli, M. Ferrara, D. Maltoni, Large-scale fingerprint identification on GPU, Information Sciences 306 (2015) 1–20.

[18] M. Lastra, J. Carabaño, P. D. Gutierrez, J. M. Benitez, F. Herrera, Fast fingerprint identification using GPUs, Information Sciences 301 (2015) 195–214.

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

[20] D. Maltoni, D. Maio, A. K. Jain, S. Prabhakar, Handbook of fingerprint recognition, Springer, New York, 2009.

[21] 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 Systems 81 (2015) 76–97.

[22] R. Cappelli, Fast and Accurate Fingerprint Indexing Based on Ridge Orientation and Frequency, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 41 (6) (2011) 1511–1521.

[23] 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 II: Experimental analysis and ensemble proposal, Knowledge-Based Systems 81 (2015) 98–116.

[24] E. Henry, Classification and Uses of Finger Prints, George Routledge and Sons, Broadway, Ludgate Hill, United Kingdom, 1900.

[25] A. K. Jain, S. Prabhakar, A multichannel approach to fingerprint classification, IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (4) (1999) 348–359.

[26] J. H. Hong, J. K. Min, U. K. Cho, S. B. Cho, Fingerprint classification using one-vs-all support vector machines dynamically ordered with naive Bayes classifiers, Pattern Recognition 41 (2) (2008) 662–671.

[27] T. H. Le, H. T. Van, Fingerprint reference point detection for image retrieval based on symmetry and variation, Pattern Recognition 45 (9) (2012) 3360–3372.

[28] Y. Yao, G. L. Marcialis, M. Pontil, P. Frasconi, F. Roli, Combining flat and structured representations for fingerprint classification with recursive neural networks and support vector machines, Pattern Recognition 36 (2) (2003) 397–406.

[29] H. O. Nyongesa, S. Al-Khayatt, S. M. Mohamed, M. Mahmoud, Fast robust fingerprint feature extraction and classification, Journal of Intelligent and Robotic Systems 40 (1) (2004) 103–112.

[30] K. Cao, L. Pang, J. Liang, J. Tian, Fingerprint classification by a hierarchical classifier, Pattern Recognition 46 (12) (2013) 3186–3197.

[31] S. García, J. Luengo, F. Herrera, Data Preprocessing in Data Mining, 1st Edition, Springer, New York, 2015.

[32] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, Journal of Machine Learning Research 3 (2003) 1157–1182.

[33] Y. Saeys, I. Inza, P. Larrañaga, A review of feature selection techniques in bioinformatics, Bioinformatics 19 (2007) 2507–2517.

[34] R. Cappelli, D. Maio, D. Maltoni, A multi-classifier approach to fingerprint classification, Pattern Analysis & Applications 5 (2) (2002) 136–144.

[35] 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, Information Sciences 315 (2015) 67–87.

[36] P. D. Gutierrez, 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 (1) (2014) 62–71.

[37] D. Taniar, C. H. C. Leung, W. Rahayu, S. Goel, High Performance Parallel Database Processing and Grid Databases, John Wiley & Sons, Hoboken, NJ, USA, 2008.

[38] Y. Su, J. Feng, J. Zhou, Fingerprint indexing with pose constraint, Pattern Recognition 54 (2016) 1–13.

[39] X. Jiang, M. Liu, A. C. Kot, Fingerprint retrieval for identification, IEEE Transactions on Information Forensics and Security 1 (4) (2006) 532–542.

[40] M. Liu, X. Jiang, A. Chichung Kot, Efficient fingerprint search based on database clustering, Pattern Recognition 40 (6) (2007) 1793–1803.

[41] M. Kawagoe, A. Tojo, Fingerprint pattern classification, Pattern Recognition 17 (3) (1984) 295–303.

[42] L. Wang, M. Dai, Application of a new type of singular points in fingerprint classification, Pattern Recognition Letters 28 (13) (2007) 1640–1650.

[43] U. Rajanna, A. Erol, G. Bebis, A comparative study on feature extraction for fingerprint classification and performance improvements using rank-level fusion, Pattern Analysis and Applications 13 (3) (2010) 263–272.

[44] R. Cappelli, A. Lumini, D. Maio, D. Maltoni, Fingerprint classification by directional image partitioning, IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (5) (1999) 402–421.

[45] J. Li, W. Y. Yau, H. Wang, Combining singular points and orientation image information for fingerprint classification, Pattern Recognition 41 (1) (2008) 353–366.

[46] K. Karu, A. K. Jain, Fingerprint classification, Pattern Recognition 29 (3) (1996) 389–404.

[47] M. Liu, Fingerprint classification based on Adaboost learning from singularity features, Pattern Recognition 43 (3) (2010) 1062–1070.

[48] A. Senior, A hidden Markov model fingerprint classifier, in: Proceedings of the 31st Asilomar Conference on Signals, Systems & Computers, Vol. 1, IEEE Comput. Soc, Pacific Grove, CA, 1997, pp. 306–310.

[49] A. Jain, S. Minut, Hierarchical kernel fitting for fingerprint classification and alignment, in: Proceedings of the 16th International Conference on Pattern Recognition (ICPR'02), Vol. 2, IEEE Comput. Soc, Quebec, Canada, 2002, pp. 469–473.

[50] Q. Zhang, H. Yan, Fingerprint classification based on extraction and analysis of singularities and pseudo ridges, Pattern Recognition 37 (11) (2004) 2233–2243.

[51] H. W. Jung, J. H. Lee, Fingerprint classification using the stochastic approach of ridge direction information, in: IEEE International Conference on Fuzzy Systems, IEEE, 2009, pp. 169–174.

[52] K. C. Leung, C. H. Leung, Improvement of fingerprint retrieval by a statistical classifier, IEEE Transactions on Information Forensics and Security 6 (1) (2011) 59–69.

[53] 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).

[54] T. M. Cover, P. E. Hart, Nearest Neighbor Pattern Classification, IEEE Transactions on Information Theory 13 (1) (1967) 21–27.

[55] V. A. Huynh-Thu, Y. Saeys, L. Wehenkel, P. Geurts, Statistical interpretation of machine learning-based feature importance scores for biomarker discovery, Bioinformatics 28 (13) (2012) 1766–1774.

[56] L. Breiman, Random forests, Machine Learning 45 (1) (2001) 5–32.

[57] M. A. Hearst, S. T. Dumais, E. Osman, J. Platt, B. Scholkopf, Support vector machines, IEEE Intelligent Systems and their Applications 13 (4) (1998) 18–28.

[58] H. T. Lin, C. J. Lin, R. C. Weng, A note on Platt's probabilistic outputs for support vector machines, Machine Learning 68 (3) (2007) 267–276.

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

[60] C. I. Watson, C. L. Wilson, NIST Special Database 4, Tech. rep., NIST (1992).

[61] C. I. Watson, NIST Special Database 14, Tech. rep., NIST (1993).

[62] A. Senior, A combination fingerprint classifier, IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (10) (2001) 1165–1174.

[63] H. H. Le, H. Nguyen, T. T. Nguyen, A Complete Fingerprint Matching Algorithm on GPU for a Large Scale Identification System, in: International Conference on Information Science and Applications, Vol. 376, Minh City, Vietnam, 2016, pp. 679–688.

[64] X. Jiang, W. Y. 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.

[65] 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.

Daniel Peralta received the M.Sc. and Ph.D. degrees in Computer Science from the University of Granada, Granada, Spain, in 2011 and 2016 respectively. He is currently a post-doctoral researcher in the Vlaams Instituut voor Biotechnologie (Ghent, Belgium), within the Data Mining and Modeling for Biomedicine research group. His research interests include data mining, biometrics and parallel and distributed computing.

Isaac Triguero received the M.Sc. and Ph.D. degree in Computer Science from the University of Granada, Spain, in 2009 and 2014, respectively. He is currently an assistant professor in data science at the University of Nottingham, United Kingdom. He has published more than 25 papers in international journals. His research interests include data mining, data reduction, evolutionary algorithms, semi-supervised learning, bioinformatics and big data learning.

Salvador García received his M.Sc. and Ph.D. degrees in computer science from the University of Granada, Spain, in 2004 and 2008, respectively. He is currently an Associate Professor at the Department of Computer Science and Artificial Intelligence, University of Granada, Spain. He has had more than 50 papers published in international journals. His research interests include data mining, data reduction, imbalanced learning, semi-supervised learning, statistical inference and Big Data.

Yvan Saeys obtained his M.Sc. (2000) and Ph.D. (2004) in Computer Science at Ghent University. He is currently leading the DAMBI research group (Data Mining and Modeling for Biomedicine), where his research focuses on the development and application of data mining and machine learning techniques for biological and medical applications.

José Manuel Benítez (M'98) received the M.S. and Ph.D. degrees in Computer Science, both from the University of Granada, Granada, Spain. He is currently an Associate Professor at the Department of Computer Science and Artificial Intelligence, University of Granada, Spain. His research interests include time series analysis and modeling, distributed/parallel computational intelligence, data mining, and statistical learning theory.

Francisco Herrera received his M.Sc. (1988) and Ph.D. (1991) in Mathematics from the University of Granada and is a professor at the Department of Computer Science and Artificial Intelligence. He is an Editor in Chief of "Information Fusion" and "Progress in Artificial Intelligence" and on the editorial board of several more.