Indexing and Retrieving in Fingerprint Databases Under Structural Distortions

Andrés Gago-Alonso*,[a], José Hernández-Palancar[a], Ernesto Rodríguez-Reina[a], Alfredo Muñoz-Briseño[a]

[a]*Advanced Technologies Application Center (CENATAV), Havana, Cuba.*

Abstract

This paper presents a new algorithm for fingerprint indexing, which is based on minutia triplets, and it is very tolerant to missing and spurious minutiae. In this sense, a novel representation for fingerprints is proposed by defining a triangle set based on extensions of Delaunay triangulations. Moreover, a set of robust features is used to build indices. Finally, a recovery method based on calculating the recommendation score is introduced, using a new similarity function between geometric transformations. Our proposal was tested on well known databases, showing that it outperforms most of the already reported methods, especially under conditions of distortions.

Key words: fingerprint identification, fingerprint indexing, triangular hull, redundant feature vector, Delaunay triangulation

1. Introduction

Biometrics can be defined as the automated use of physiological or behavioural characteristics to identify or verify the identity of a person. One of the most widely used techniques in biometric systems is the comparison of fingerprints. The ridge patterns found on fingers and other body parts are unique, and they provide enough information to distinguish a specific person from the rest. Also, these patterns can be extracted very easily and are very reliable compared with other biometric features.

There are two kinds of general problems related to fingerprint recognition systems: verification and identification. The purpose of verification systems is to confirm the identity of a particular individual, so comparisons are only necessary with fingerprints that belong to that person [16, 40]. On the other hand, the purpose of identification is to establish the identity of a specific person, given a query impression and a dataset of fingerprints of different individuals. As we can see, identification requires a search on all possible fingerprint candidates. However, a comparison between the query and every candidate stored in the dataset is impracticable, since modern fingerprint collections usually have millions of entries.

*Corresponding author. Tel.: +537-272-1670; fax: +537-273-0045

Email addresses: agago@cenatav.co.cu (Andrés Gago-Alonso), jpalancar@cenatav.co.cu (José Hernández-Palancar), erreina@cenatav.co.cu (Ernesto Rodríguez-Reina), amunoz@cenatav.co.cu (Alfredo Muñoz-Briseño)

There are some proposed approaches in literature that try to reduce the search space in which the comparisons are made [8]. One of these solutions is the classification of the fingerprints stored in the dataset, in the five classes of Henry (left loop, right loop, arch, tended arch and whorl) [29, 41, 19]. These classes divide the impressions in groups according to ridge patterns. In this way, comparisons are only made with fingerprints in the dataset that have the same classification as the query. This method has serious disadvantages mainly because the number of classes in which the search space is divided is small. In addition 90% of impressions belong to three classes [28], so, in most cases, the reduction of potential candidates is insignificant.

Another group of algorithms uses indexing in order to return a subset of the dataset, ordered by a recommendation score. This approach, also known as continuous classification, allows choosing the number of impressions of the dataset that will be compared to the query. However, most of these solutions do not have robust strategies to deal with missing or spurious minutiae, and the commonly used mechanisms to reduce the negative effect of noise are insufficient.

This paper proposes an indexing algorithm which is prepared for dealing with the problem of missing and spurious minutiae. This algorithm is based on minutia triplets, and it introduces a novel fingerprint representation based on an expanded triangle set obtained from Delaunay triangulations. From each of these triangles, indices are formed using fingerprint features such as ridge counters, minutia directions and triangle sign. With these indices, an index table is built in preprocessing time. In the retrieving stage, a novel method for calculating the recommendation score and a mechanism to deal with noise are also defined. Thus, we can get a list of candidates with the highest degrees of affinity with the query, considering the best geometric transformation.

The rest of this paper is organized as follows. In section 2, some basic concepts, which are useful for understanding the rest of the document, are presented. Also, a general scheme of fingerprint indexing algorithms is given. Next, in section 3, a description of the main state of the art algorithms is exposed. In section 4, a new feature extraction strategy is defined using a new criterion for selecting the set of triangles. Section 5 introduces the index function, the indexing process, and index table construction. In section 6, a novel method for recovering a list of candidates is proposed. Finally, experimental results are shown in section 7, and our conclusions are given in section 8.

2. Background

In this section, we present some basic concepts and a general scheme of fingerprint indexing algorithms. Thus, we declare the necessary background for understanding our proposal and the rest of the paper. Finally, we describe the Delaunay triangulation and its properties, considering that this kind triangulation is used for many indexing algorithms, including our approach.

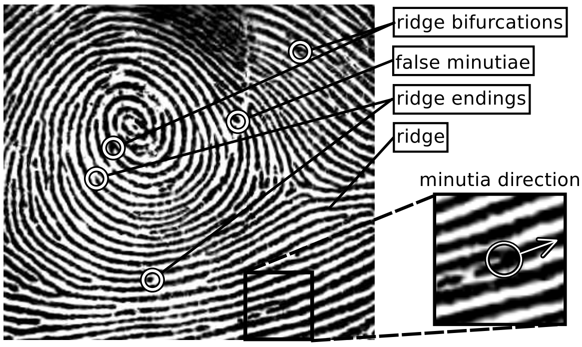


Figure 1: Fingerprint parts.

2.1. Fingerprint related concepts

Fingerprints are marks produced by the contact of a finger with a surface in a controlled environment. These marks reflect the different patterns formed by the ridges that are visible in the epidermis. For many years, fingerprint image acquisition has been accomplished from different sources like: inked finger on paper or ink-less fingerprint scanners. In section 7.1, we can see examples of some datasets of fingerprints obtained in different sources.

Most of the indexing algorithms use minutiae as basis for representing fingerprint and building indices. Minutiae are singularities in the ridge patterns, which are commonly classified in two types: bifurcations and endings. A bifurcation is a point where a ridge splits into two ridges, while an ending is an endpoint of a ridge. In Fig. 1, we can see examples of bifurcations, endings, and ridges in a real fingerprint.

The direction of a minutia is another commonly used feature in fingerprint indexing algorithms. This feature is defined as the angle formed between the horizontal axis and the tangent of the ridge associated to the corresponding minutia, in counter clockwise. In Fig. 1, a bifurcation with its respective direction is shown.

There are several published minutia extraction algorithms which have shown an allowable performance [22, 41]. However, in almost all of these methods, the possibility of finding false minutiae always exists. False minutiae are the points which are incorrectly identified as minutiae. In Fig. 1, we can see a false minutiae caused by a scar in the finger.

2.2. Indexing based systems

The general scheme of all indexing based fingerprint identification systems, is the same, see Fig. 2. Such scheme is made up of an indexing stage and a retrieving stage. Indexing stage is also known as offline stage since it is executed while the fingerprint collection is preprocessed. The queries are attended in a retrieving stage, which detects the query occurrence in the fingerprint collection.

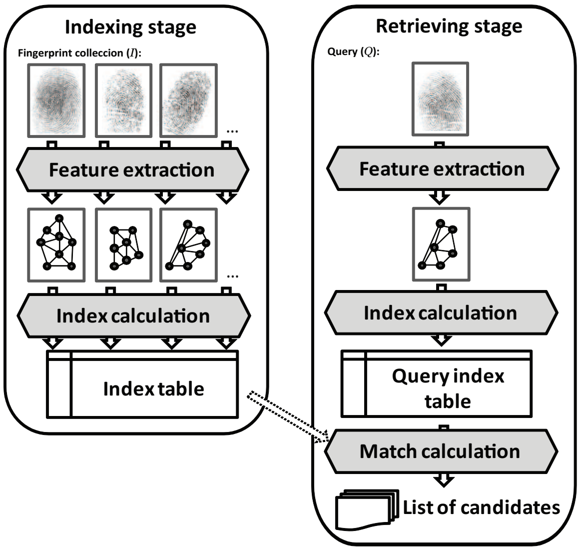


Figure 2: General scheme of an indexing based fingerprint identification system.

The indexing stage can be described as follows in almost all reported methods. Let $I = \{F_1, F_2, \ldots, F_N\}$ be a collection of fingerprints, where F_i represents the i-th stored impression, and N is the number of impressions in I. Each $F_i \in I$ is preprocessed for extracting a set of features, which is used as model for representing it. These models are the basis for calculating indices which are stored in the index table. This table is used for reducing the search space during the retrieving stage.

The retrieving stage also has the same structure in almost all reported methods. Given a query fingerprint Q, it is required to detect if there is any $F_h \in I$ such that F_h and Q represent the same finger of the same person. This stage finds a list of candidates $C_Q \subset I$, such that the probability of finding F_h in C_Q is very high, while the probability of finding F_h in $I \setminus C_Q$ is very low.

Each query Q is processed in a similar way as it was done for each fingerprint in I during the indexing stage. Thus, a set of features is calculated, and it is used as model for representing Q. This model is used for calculating the query indices, which are used for finding the matches with the already calculated index table. Finally, the list of candidates is obtained by combining these match results.

2.3. Delaunay triangulation

In general, a triangulation of a set of points, $P = \{p_1, p_2, \ldots, p_N\}$, in the plane is the set of triangles that conforms a maximal planar subdivision whose vertex set is P. A maximal planar subdivision is a subdivision S such that no edge connecting two vertices can be added to S without destroying its planarity [2].

Delaunay triangulation is a specific kind of triangulation, which has been used for representing fingerprints, in some of the reported indexing algorithms [1, 27, 28]. This concept is defined as follows.

Definition 1 (Delaunay Triangulation). *Let* $P = \{p_1,\, p_2, \ldots, p_N\}$ *a set of points in the plane, and let*

T be a triangulation of P. Then, T is a Delaunay triangulation if and only if every triangle $\triangle P_i P_j P_k$ that belongs to T satisfies that its circumcircle contains no other point of P.

Based on the above, we define a Delaunay graph of a Delaunay triangulation T as a tuple $G = \langle P, E\rangle$ where P is the set of planar points that originated T, and E is the set of edges that conforms the triangles of T. Each edge has a single occurrence in E.

As we can see, the insertion of a new point in a Delaunay triangulation only affects the triangles whose circumcircles contain that point. As a result, noise only affects the Delaunay triangulation locally, which gives a good structural stability when small changes occur in the set of points.

Delaunay triangulations have some properties [2], such as:

- The union of all triangles of a Delaunay triangulation of a set of points P conforms the convex hull of P.
- The Delaunay triangulation of a set of points P has $2N - 2 - k$ triangles and $3N - 3 - k$ edges, where N is the number of points in P and k is the number of points of P forming the convex hull.
- The Delaunay triangulation maximizes the minimum angle of every formed triangle. Compared to any other triangulation of a set of points P, the smallest angle in the Delaunay triangulation is at least as large as the smallest angle in any other.
- A Delaunay triangulation of a set of points P is unique if there is no circumcircle with more than three points of P on its border.

In our paper, we use Delaunay triangulation as start point to define an expanded triangle set, which is used by our indexing proposal for representing fingerprints.

3. Related work

Using the above mentioned scheme, several indexing based approaches for fingerprint identification have been reported [1, 3, 28, 8, 33]. These approaches are classified by us according to the fingerprint features used for indexing into the following classes: minutia triangle based approaches, ridge based approaches, and image processing based approaches.

3.1. Minutia triangle based approaches

The most commonly proposed strategies are based on minutia triangle. The first column of Table 1 shows a summary of these reported methods.

Table 1: The datasets and indexing methods for which published results are available.

Reported solution	**Topological features**	**Triangle geometric features**	**Fingerprint features**
Proposed approach (2011) [in this paper]	Delaunay triangles + Redundant triangles	handedness (s_t)	relative direction of each minutia regarding the opposite triangle side ($\beta_1, \beta_2, \beta_3$) and ridge counter between pairs of minutiae (r_1, r_2, r_3)
Biswas et al. (2008) [4]	All triangles	side lengths (l_1, l_2, l_3) and angle amplitudes (α_1, α_2, α_3)	ridge curvature for each minutia neighborhood (c_1, c_2, c_3)
Ross and Mukherjee (2007) [43]	Delaunay triangles	cosines of angles ($\cos(\alpha_3)$), perimeter and area ratio (p^2/A), and side ratios (l_3/l_1)	second degree curve coefficients of incident ridges ($\kappa_1, \kappa_2, \kappa_3, \lambda_1, \lambda_2, \lambda_3$)
Liang et al. (2007) [28]	Low-order Delaunay triangles	angle amplitudes (α_1, α_2), handedness (s_t) and side lengths (l_3)	minutia types (11 values) and relative directions of each minutia regarding the incident triangle sides (ϕ_1, ϕ_2, ϕ_3)
Liang et al. (2006) [27]	Delaunay triangles	angle amplitudes (α_1, α_2), handedness (s_t) and side lengths (l_3)	minutia types (11 values) and relative directions of each minutia regarding the incident triangle sides (ϕ_1, ϕ_2, ϕ_3)
Bhanu and Tan (2003) [3]	All triangles	side lengths (l_3), angle amplitudes (α_1, α_2) and handedness (s_t)	minutia types (4 values)
Choi et al. (2003) [12]	All triangles	side lengths (l_3) and angle handedness (s_t)	minutia types and directions
Bebis et al. (1999) [1]	Delaunay triangles	side ratios (l_1/l_3, l_2/l_3), cosines of angles ($\cos(\alpha_3)$)	None
Germain et al. (1997) [15]	All triangles	side lengths (l_1, l_2, l_3)	ridge counters (r_1, r_2, and r_3) and minutia directions (θ_1, θ_2, θ_3)

Feature extraction is a basic subtask in any indexing based system. This step is used in the same way for preprocessing fingerprints in collections and queries, and the resulting feature model is used for building the index tables.

The feature extraction step in minutia triangle based solutions can be described as follows, see Fig. 3. First, the minutiae of the fingerprint are detected. Next, a set of triangles is calculated; examples of these kinds of triangle sets are shown in the second column of Table 1. After that, geometric and fingerprint features are calculated for completing the feature model; examples of these features are also shown in the last two columns of Table 1.

As we can see, there are some ways for selecting this set of triangles: using all triangles among any minutia triplets in the fingerprint [3, 4, 12, 15], only using Delaunay triangles [1, 27, 43], and using low-order Delaunay triangulations [28]. However, all of the reported solutions have at least one of the following problems:

1. Considering all triangles increases the execution times and identification errors by the possibility of

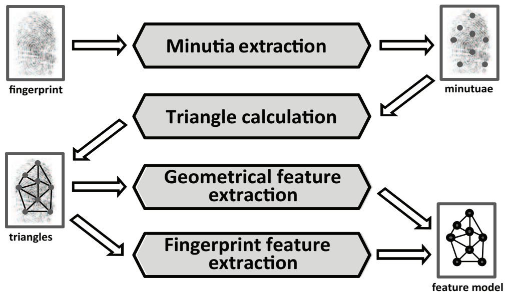


Figure 3: General scheme of feature extraction in minutia triangle based approaches.

false acceptance.

2. Delaunay triangulation may not be stable if fingerprints are affected even by tiny distortions, such as: minutia shifts, spurious minutiae, and missing minutiae.

3. Using low-order Delaunay triangles only reduces the negative effect caused by minutia shifts.

In section 4.1, we introduce a new criterion for choosing triangles, called expanded triangle set, considering Delaunay triangles and some redundant triangles. The expanded triangle set helps to reduce the negative effect produced by spurious minutiae and missing minutiae.

On the other hand, when the set of triangles is resolved, geometric and fingerprint features are calculated for building the above mentioned feature model. These features must be stable by linear distortions, such as scale, rotation, and translations, non-linear distortions, like shear, and other image source damages, like occlusion and clutter caused by scars, dryness, sweat, and smudge.

Geometric features are used in almost all reported methods, see Table 1. However, at least one of the following problems is presented in the reported solutions:

4. Geometric features based on measures, such as distances, areas, or combinations, may not be stable by even tiny distortions.

5. Geometric features based on angles are more stable than those based on measures; however, the implementation of mechanism for reducing the negative effect of distortion is required.

Fingerprint features, meanwhile, are also used in almost all reported methods, see Table 1. The effectiveness of a fingerprint indexing system widely depends on the accuracy of the feature extraction methods. Therefore, all reported solutions have the following problem:

6. The feature extraction methods reported in the literature are not unfailing ones.

To check for fingerprint feature extraction methods, other already published works can be consulted, for example the comparative study of Rajanna et al. [41] published two years ago.

In section 4.2, we propose a feature model only considering the triangle handedness, as geometric feature, relative minutia directions, and ridge counters, as fingerprint features. As it can be seen in other steps of our proposal, the goal of our research is focused on improving indexing and retrieving tasks, by dealing with the failures of feature extraction methods instead of proposing better solutions for feature extraction tasks.

Index calculation is the subsequent step after feature extraction. Each triangle t in each already calculated feature model is inserted in the index table. This table works as a hash table where indices are used as keys, and triangles are used as values. Triangles with the same features are inserted in the same key inside the index table.

In the retrieving stage, the index calculation is used for generating the query index table. In this case, the indices are calculated in similar way as it is done during the indexing stage, including some redundant information taking into account the possibility of noise distortions in data. That is, the same triangle is inserted in position keys associated with very similar features.

Using the two above mentioned tables, the index table and the query index table, the retrieving stage proceeds to the match calculation step. This step is also known as accumulating evidence step, since it performs a casting among all fingerprints in the collection.

There are two reported ways for performing the match calculation step: vote based strategy and transformation based strategy. The vote based strategies accumulate evidence by counting a vote for every entry stored in the indexed locations and picking up the ones with the high number of votes [3, 27, 28, 43]. The problem with these approaches is that it does not consider whether these votes are consistent among themselves; this situation is solved by transformation based strategies. In this sense, the transformation based strategies accumulate evidence by counting votes in the transformation space introducing a measure of coherence [15, 1, 4]. The main idea in these approaches is to consider the best geometric transformations between query and template.

The solution proposed in this paper includes a new way for match calculation, guaranteeing coherence among the votes in the transformation space and considering distortion in data, see sections 5 and 6.

3.2. Other approaches

One of the most successful approach for fingerprint indexing is uses ridge invariants as features [13]. This work proposed the creation of substructures formed by the ridges that converge in each minutia. Each ridge is subdivided in sub-ridges taking the minutiae as extreme points. These sub-ridges are labeled according to their relative positions with respect to the analysed minutia. The indices are derived from binary relations between substructures and the labels generated by its associated sub-ridges.

Another very accurate approach uses the Minutiae Cylinder Code in order to generate fixed length binary indices [8]. This feature is a representation based on 3D data structures, and it is built from minutia distances and angles. In the retrieving stage, a local sensitive hashing algorithm is used for finding similarities between the binary vectors.

On the other hand, there is another kind of methods for fingerprint indexing based on image processing. For example, there is a proposal where Gabor filters have been used for processing each minutia neighborhood and obtaining the index vectors [24]. Symmetric filters from cores, deltas, and parallel patterns are used as indexing features [26]. Additionally, MACE filters are used in order to generate index vectors [30]. The SIFT, SURF, and DAISY points are also used as features for indexing tasks [18, 44]. Moreover, there is as approach where a combination of the results of different indexing methods is proposed [5]. Another algorithms use the orientation maps of fingerprints in order to generate features to construct the index vectors [9, 10, 21, 25, 32, 33, 34]. In the literature there are some methods that estimate with accuracy the orientation map [42]. Some proposals also merge different strategies of fingerprint indexing in order to fuse their advantages [5, 11].

There is other approach that defines a minutia tree in order to find sequences of minutiae with the same geometric relationships between the fingerprints [38]. Other reported method indexes the minutiae in order to find correspondences between fingerprints [47]. Another algorithm generates indices from the interaction of the queries and a fingerprint reference set; which is built with fingerprints having a good representative and discriminative power, in preprocessing stage [17].

4. Feature extraction step

In this section, we propose a new feature extraction strategy using a new criterion for selecting the set of triangles, starting from Delaunay triangulation. First, we define the expanded triangle set of a planar point set in section 4.1. Next, the feature model for representing fingerprints is proposed, see section 4.2. As in a previous work [39], our representation is an extension of the Delaunay triangulation that deals with some kind of noise in the fingerprints, in this case with spurious minutiae.

4.1. Selecting the set of triangles

Let $P = \{p_1, p_2, \ldots, p_N\}$ be a set of points in the plane, where $G = \langle P, E\rangle$ is its Delaunay graph and T is its Delaunay triangulation. To be able to formally define the expanded triangle set of P, we first define the *triangular hull* of any point $p_i \in P$.

Definition 2 (Triangular hull). *Let p_i be a point of P. The set $N_i = \{p_j | \{p_i, p_j\} \in E\}$ denoted the point set formed by all the adjacent vertices of p_i in the Delaunay graph G. The triangular hull of p_i is defined as the Delaunay triangulation of the planar point set N_i, and it is denoted by H_i.*

As we can see, the number of points in each set N_i is the degree of p_i in the graph G, and it is denoted by d_i.

Definition 3 (Expanded triangle set). *The expanded triangle set of P is defined as $R = T \cup H_1 \cup H_2 \cup \ldots \cup H_N$.*

The set R includes the triangles in the Delaunay triangulation of P and any triangle in the triangular hulls of the points in P. Therefore, $|R| \geq |T|$; however, we can prove that $|R| \in O(N)$. The following theorem gives a characterization of the number of triangles in R.

Theorem 1. *The number of triangles in R is lesser than $13N - 25$.*

PROOF. The number of triangles in R can be calculated as follows:

$$|R| \leq |T| + \sum_{i=1}^{N} |H_i|. \tag{1}$$

Since T is the Delaunay triangulation of a set with N points, the number of triangles can be bounded by $2N - 1$. The same statement can be applied to each triangular hull H_i. Thus, the inequality (1) can be transformed in (2).

$$\begin{aligned} |R| &\leq 2N - 1 + \textstyle\sum_{i=1}^{N} (2d_i - 1), \\ &= 2N - 1 + 2\textstyle\sum_{i=1}^{N} d_i - N, \\ &= N + 2\textstyle\sum_{i=1}^{N} d_i - 1. \end{aligned} \tag{2}$$

Using a basic principle of the graph theory, the sum of the degrees, d_i, of the vertices of the graph G is precisely twice the number of edges, $|E|$. Thus, the inequality (2) can be transformed in (3).

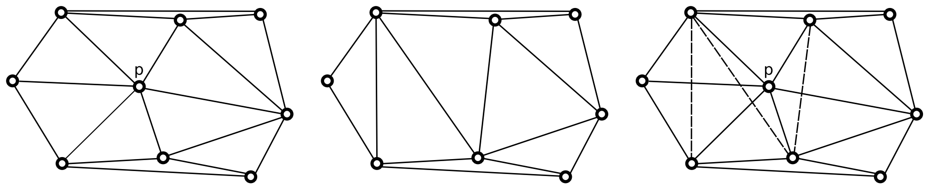
$$|R| \leq 4|E| + N - 1. \tag{3}$$

Using the Euler characteristic for planar graph [2] we obtain $|E| \leq 3N - 6$. Thus, we have $|R| \leq 13N - 25$, and we conclude the proof.

□

As we can see, despite the fact that $|R|$ is greater than $|T|$, the number of triangles of $|R|$ is still linear with respect to N. This is very desirable if we consider that the sets R will be used as a representation for fingerprints in indexing tasks. This property also has the advantage that the identification errors by false acceptance are reduced in comparison with other approaches that use all triplets or only Delaunay triangulations. On the other hand, with this representation the execution times of our proposal are very similar to methods that only use Delaunay triangulations.

The advantage of the set R is that it contains all of the Delaunay triangles that are formed when each minutia is eliminated individually. In this way, we ensure that even when the extraction method fails to find a minutia, some of the matchings will be found.



(a) Delaunay triangulation (b) Delaunay triangulation without p (c) Expanded triangle set

Figure 4: Triangle set examples.

In Fig. 4(a), we can see a Delaunay triangulation of a set of points. In Fig. 4(b), we can appreciate major structural changes in the same triangulation when removing the vertex p. On the other hand, Fig. 4(c) shows the expanded set of the points including p. As we can see, Fig. 4(c) has corresponding triangles with both, Fig. 4(a) and Fig. 4(b) due to the use of the expanded triangle set. This example shows that with the defined set R is more likely to find correspondences than with Delaunay triangulations, especially when some minutiae are not detected on the involved fingerprints.

In the present paper, the expanded triangle sets of minutiae are used for representing fingerprints in indexing and retrieving tasks. Moreover, in section 7.3 we show an experimental evaluation for checking the good accuracy obtained using expanded triangles for indexing tasks.

4.2. Triangle invariant features

Let $P = \{p_1, p_2, \ldots, p_N\}$ be the set containing all the planar points representing the minutiae in a fingerprint F. Let R be the expanded triangle set of P, and let $t \in R$ be a triangle, which represents a minutia triplet. Let $m_1 = (x_1, y_1)$, $m_2 = (x_2, y_2)$, and $m_3 = (x_3, y_3)$ be the three points of t, with their corresponding planar coordinates, which are sorted in ascending order regarding the length of the opposite side.

The feature vectors associated to t in the fingerprint F is denoted by $f(t)$, and it is defined as follows

$$f(t) = (s_t, \beta_1, \beta_2, \beta_3, r_1, r_2, r_3), \quad (4)$$

where s_t is the triangle sign, β_i is the relative direction of m_i, and r_i is the ridge counter of the opposite side to m_i in the triangle t. The seven components of this feature vector are formally defined as follows.

The twice signed area of t is calculated using the following mathematical expression

$$A_t = x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2). \quad (5)$$

Using A_t, we define the triangle sign of t as $s_t = 0$ if $A_t < 0$; otherwise $s_t = 1$. As we can see, this feature is invariant to rotation.

Let θ_i be the angle that the ridge makes with respect to the X-axis at the minutia point m_i. Let $u_1 = \overrightarrow{m_2m_3}$, $u_2 = \overrightarrow{m_3m_1}$, and $u_3 = \overrightarrow{m_1m_2}$ be the vectors representing the sides of the triangle t. Let d_i be an angle, $0 \leq d_i < 360°$, which is co terminal with $\theta_i - Arg(u_i)$, where the vector u_i forms an angle $Arg(u_i)$ with respect to the X-axis. Using d_i, we define the relative direction of m_i as $\beta_i = \lfloor d_i/45° \rfloor$, where $\lfloor \cdot \rfloor$ denotes the integer part operator or floor function. Since $0 \leq d_i < 360°$, then we have $0 \leq \beta_i < 8$.

The ridge counter r_i is defined as the number of ridges crossed by the opposite side of m_i in the triangle t. We study the statistical behavior of this feature in some real world datasets, and only in a small number of cases are greater than 16. Therefore, we remove from R those triangles with at least one value outside the interval, $0 \leq r_i < 16$.

The feature vectors presented in this section can be represented as a function $f : R \to \Phi$ called feature function, where the set $\Phi = K_1 \times K_3^3 \times K_4^3$, assuming $K_n = \{0, 1, \ldots, 2^n - 1\}$, represents the feature space. Thus, we are able to define the formal representation of a fingerprint F, which is used in this paper.

Definition 4 (The feature model). *Let F be a fingerprint. The model of F is defined as a triplet $M = \langle P, R, f \rangle$, where P is the planar point set representing the minutiae of F, R is the expanded triangle set of P, and f is a feature function $f : R \to \Phi$.*

This feature model is used during indexing and retrieving stages for representing fingerprints and queries respectively. The triangle sign used in our approach is a very robust feature. Additionally, we use ridge counters and minutia directions, which are widely used in other triplet based indexing algorithms. These features, combined with the mechanism defined in section 6.1 to reduce the negative effects of noise, show a good performance when they are used in indexing tasks, see section 7.

5. The indexing stage

In this section, we introduce the index function, and we describe the index building process. We also define the index table that will contain those indices. This table is built in the preprocessing stage of our proposal in order to collect information that will be used in the recovery stage.

5.1. The index function

Let $M = \langle P, R, f \rangle$ be a feature model, according to the definition 4, and let $t \in R$ be a minutia triangle. The feature vector $f(t) \in \Phi$ has seven components. One of them get values in K_1, three of them get values in K_3, and the other ones get values in K_4. As we can see, for all $n \geq 1$ the numbers in K_n have a binary representation with only n bits. Therefore, concatenating the binary representations of the components of $f = f(t)$, we can conform an integer number represented with 22 bits. This integer number is denoted by $h(f)$. Using theses facts, we define the index function as follow.

Definition 5 (Index function). *The index function in the feature model M is defined as $h : \Phi \to K_{22}$, such that for each $t \in R$, $h(f(t))$ is the integer number obtained by concatenating the binary representation of the components of the feature vector $f(t)$.*

5.2. The index table

Let $D = \{M_1, M_2, \ldots, M_N\}$ be a collection of feature models, where $M_i = \langle P_i, R_i, f_i \rangle$ for each i, $1 \leq i \leq N$. To be able to formally define the index table of D, we start by defining the record of a triangle $t \in R_i$ in a model M_i.

Definition 6 (Record of a triangle). *Let i be an integer number, $1 \leq i \leq N$, and let $t \in R_i$ be a triangle in a model $M_i \in D$. The record of t is defined as the vector $r(t) = (i, x_1, y_1, x_2, y_2, x_3, y_3)$, where (x_i, y_i) are the coordinates of each point m_i in the triangle t, and i is the fingerprint identifier.*

Let A_k be the set of records of triangles such that their corresponding index value is k; that is

$$A_k = \{r(t) | t \in R_i, 1 \leq i \leq N \text{ and } h(t) = k\}. \tag{6}$$

The index table of D is a hash table that uses the index values (k) as key and the list of records (A_k) as values. In this paper, we use minimal perfect hashing [7] for implementing such hash table.

The algorithm 1 shows the pseudo-code for building the index table of the collection D. First, an empty hash table H is created; this table is populated in the following lines. Lines 2 and 3 traverse all models in D and all of the triangles in such models. In the iteration of the lines 4-9, the function index is evaluated in the triangle t for calculating the index k, which is used as key in the hash table H. If the key k has been calculated in H, the corresponding set A_k is updated by adding a new record $r(t)$.

6. The retrieving stage

In this section, we propose a novel method for recovering a list of candidates based on the index table, which was previously constructed. Details of the used algorithm for computing similarities between the query and the stored models are also described in this section.

6.1. Processing the query

The query Q is processed in a similar way as it was done for each fingerprint in I during the indexing stage. First, the feature model $M_Q = \langle P_Q, R_Q, f_Q \rangle$ of Q is calculated. Next, a query index table of M_Q is built using the algorithm 2.

The algorithm 2 works as follows. In line 1, an empty hash table is initialized, and it is populated by traversing every triangle $t \in R_Q$. The record of each triangle t is calculated in line 3; in this case, we use $r(t) = (x_1, y_1, x_2, y_2, x_3, y_3)$ only including the coordinates of the points of t. Next, the feature vector $f_Q(t)$ is calculated in line 4. For each feature vector f, the set of redundant feature vector is defined as follows.

Function `CreateIT`(D)

Input: $D = \{M_1, M_2, \ldots, M_N\}$ - feature models representing the fingerprint collection

Output: H - the index table of D

$H \leftarrow$ an empty hash table without keys and without values.
foreach $M_i = \langle P_i, R_i, f_i \rangle \in D$ *with* $1 \leq i \leq N$ **do**
    foreach $t \in R_i$ **do**
        $k \leftarrow h(f(t))$;
        $A_k = \emptyset$;
        if k *is a key of* H **then**
            $A_k \leftarrow$ the value associated to the key k in H;
        $A_k \leftarrow A_k \cup \{r(t)\}$;
        Insert the key k and the value A_k in H;
return H;

Algorithm 1: Pseudo-code for creating the index table.

Definition 7 (Redundant feature vectors). *Let $f \in \Phi$ be a feature vector such that $f = (s_t, \beta_1, \beta_2, \beta_3, r_1, r_2, r_3)$. The set of redundant feature vector of f is defined as*

$$J(f) \quad = \quad \{f' \,|\|f' - f\|_\infty \leq 1 \wedge \|f' - f\|_1 \leq e_1\},$$

where $\|x\|_\infty = \max(|x_1|, |x_2|, \ldots, |x_k|)$, $\|x\|_1 = \sum_{i=1}^{k} |x_i|$, for $x = (x_1, x_2, \ldots, x_k)$. Besides, e_1 is an user defined threshold, according to the application.

The set of redundant feature vector $J(f)$ is used for considering noise distortion during the retrieval stage. In our work, we do not accept noise distortion in the first component s_t, in order to reduce the search space and improve the index selectivity; moreover, s_t is a geometric feature very stable in the presence of tiny distortions. The presence of noise in the other fingerprint features is quite expected; therefore, redundant vectors are required for facing the instability problems, which were described in section 3.1. Thus, we consider e_1^3 redundant feature vectors, one of them is f, and the others ones differ in ± 1 for at least one component.

For each triangle $t \in R_Q$, the set $J(f)$ is traversed in line 5. In the iteration of the lines 6-11, the function index is evaluated for each redundant feature vector f' for calculating the index k, which is used as key in the hash table H_Q. If the key k has been calculated in H_Q, the corresponding set A_k is updated by the record r. It is important to remark that, in the case of query index table, the same record r can be inserted in different keys in H_Q. This fact will be useful in next stages during the retrieving stage, for increasing the accuracy of our proposal.

Function `CreateQIT(`M_Q`)`

Input: $M_Q = \langle P_Q, R_Q, f_Q \rangle$ - the query feature model

Output: H_Q - the query index table of Q

$H_Q \leftarrow$ an empty hash table without keys and without values.
foreach $t \in R_Q$ **do**
    $r \leftarrow r(t)$;
    $f \leftarrow f_Q(t)$;
    foreach $f' \in J(f)$ **do**
        $k \leftarrow h(f')$;
        $Q_k = \emptyset$;
        if *k is a key of H_Q* **then**
            $Q_k \leftarrow$ the value associated to the key k in H_Q;
        $Q_k \leftarrow Q_k \cup \{r\}$;
        Insert the key k and the value Q_k in H_Q;
return H_Q;

Algorithm 2: Pseudo-code for processing the query.

6.2. Recovering index matches

Let us suppose that the index table H of the collection of feature models D is given, and let H_Q be the query index table of a query Q. Using H and H_Q, we can compute the index matches by means of the algorithm 3.

The line 1 of the algorithm 3 starts by initializing M as an empty hash table. Next, all of the keys k in the table H_Q are traversed, see line 2. After that, the sets of triangle records Q_k and A_k are obtained from H_Q and H respectively. The set Q_k is the set of triangle record with index k in H_Q whereas A_k is the set of triangle record with the same index in H. Tentatively, each record $q \in Q_k$ can be matched with each record $r \in A_k$, since q and t represent triangles with similar feature vectors.

A match between two records q and t is defined by three geometric transformations among corresponding sides in such triangles. Let $\tau_1 = (\lambda, \omega, x, y)$ be the geometric transformation between the smallest sides in the triangles represented by q and t. In this case, λ is the positive real scale factor, ω is the rotation angle in the range $-180° < \omega \leq 180°$, and (x, y) is the translation vector. In the same way, we define τ_2 as the geometric transformation between the second smallest sides, and τ_3 as the geometric transformation between the highest sides.

It is known that small local distortions can cause sizable global deformations [23]. In this way, the

Function `FindMatches(`H, H_Q`)`

Input: H - the index table, H_Q - the query index table

Output: M - the match table

1 $M \leftarrow$ an empty hash table without keys and without values.
2 **foreach** *key* k *of* H_Q **do**
3     $Q_k \leftarrow$ the value associated to the key k in H_Q;
4     $A_k \leftarrow$ the value associated to the key k in H;
5   **forall** *record* $q \in Q_k$ **do**
6   **forall** *record* $r \in A_k$ **do**
7             $i \leftarrow$ the fingerprint identifier in r;
8             $T_i \leftarrow \emptyset$;
9   **if** i *is a key of* M **then**
10                 $T_i \leftarrow$ the value associated to the key i in M;
11             $\Upsilon \leftarrow \{\tau_1, \tau_2, \tau_3\}$, where τ_1, τ_2 and τ_3 are the geometric transformation between the corresponding sides in the triangles of q and r;
12   Remove from Υ the non valid transformations;
13             $T_i \leftarrow T_i \cup \Upsilon$;
14   Insert the key i and the value T_i in M;
15 **return** M;

Algorithm 3: Pseudo-code for detecting all matches.

disparity between two pairs of analogous minutiae grows as their corresponding distances increase in size [20]. Since the lengths of each triangle sides are almost always different, the deformation between corresponding sides can also be distinct. On the other hand, in a false matched pair of triangles we can find a true matched pair of minutiae. For these reasons, we use a geometric transformation for each side instead of a single one from the whole triangle. Moreover, in Fig. 6(a) of the section 7.3 we present an empirical justification for our proposal.

In lines 7-14 of the algorithm 3, all the matches among the records q and t are calculated. The geometric transformations between these matches are inserted in the multiset T_i (T_i is a multiset since it can contain the same transformation several times). As we can see, the hash table M uses the fingerprint identifiers as keys and the multiset of geometric transformations as values. Each multiset T_i is called the transformation space of the fingerprint F_i.

Since these concepts are used in fingerprint recognition context, we apply some restrictions for filtering the set of geometric transformations used during retrieving stage. In this sense, we say that $\tau = (\lambda, \omega, x, y)$ is a valid transformation if $1 - E_\lambda \leq \lambda \leq 1 + E_\lambda$ and $-E_\omega \leq \omega \leq E_\omega$, where E_λ and E_ω are user defined thresholds, see line 12 the algorithm 3.

6.3. Computing candidate list

Let T_i be the transformation space of the fingerprint F_i. This multiset contains geometric transformation candidates for aligning F_i with the query Q. Moreover, we need to choose the best transformation of T_i for aligning F_i with Q. In this section, we present a criterion for choosing such transformation.

Definition 8 (Similarity function). *Let $\tau_1 = (\lambda_1, \omega_1, x_1, y_1)$ and $\tau_2 = (\lambda_2, \omega_2, x_2, y_2)$ be two geometric transformations. The similarity function of these transformations is defined as*

$$\varphi(\tau_1, \tau_2) \quad = \quad \min\{f_\lambda(\lambda_1, \lambda_2), f_\omega(\omega_1, \omega_2), f_t(x_1, x_2), f_t(y_1, y_2)\},$$

where f_λ, f_ω, and f_t are the partial similarity functions between the components of the geometric transformations, which are defined as follows

$$\begin{aligned} f_\lambda(\lambda_1, \lambda_2) &= b_{e_\lambda}\left(\lambda_1/\lambda_2 - 1\right), \\ f_\omega(\omega_1, \omega_2) &= b_{e_\omega}(\omega_1 - \omega_2), \\ f_t(x_1, x_2) &= b_{e_t}(x_1 - x_2). \end{aligned}$$

In this case, the family of bell shaped functions $b_e(\xi) = exp(-\xi^2/2e^2)$ is used for defining the partial similarity functions. Moreover, the values e_λ, e_ω, and e_t are user defined thresholds, according to the kind of application.

In this definition, we choose the bell shaped function in order to describe in fuzzy terms, see [6], the degree of closeness between each corresponding pair of transformation components. Using the similarity function, we can define a weight of a transformation $\tau \in T_i$.

Definition 9 (Weight of a transformation). *The weight of a transformation $\tau \in T_i$ is defined as follows*

$$w(\tau) = \sum_{\tau' \in T_i} \varphi(\tau, \tau').$$

This weight allows us to proportionally describe the level of matching between the corresponding minutiae in these triangles by aligning F_i and Q with the transformation τ. The highest weight is associated to the best geometric transformation between these fingerprints. Using the weight of the best geometric transformation, we are able to define the recommendation score of the fingerprint F_i.

Definition 10 (Recommendation score). *The recommendation score of a fingerprint F_i is defined as $\rho(i) = \max\{w(\tau) | \tau \in T_i\}$.*

Our recommendation score proposal is quite similar to the one proposed by Germain et al. [15]. Our main contribution is the the use of similarity values and weights for taking into account the degree of closeness, during the score computation. In Fig. 6(c) of the section 7.3, we show a evaluation of our proposal using other methods for calculating the recommendation score.

The main task in the retrieving stage is processing the list T_i for each fingerprint F_i, keeping the recommendation score in each identifier i, see algorithm 4. Line 1 initializes an empty candidate list. Next, the match table M is traversed for computing the recommendation score, see lines 2-5. Finally, the candidate list is sorted in descending order according to the score values, see line 6. The outputs of this stage are the fingerprints with the N largest score, where N is defined by the user, see line 7.

Function `FindCandidates(`M, N`)`

Input: M - the match table, N - number of elements in the candidate list

Output: L - sorted list of candidates

1 $L \leftarrow \emptyset$;
2 **foreach** *key* i *of* M **do**
3     $T_i \leftarrow$ the value associated to the key i in M;
4     $\rho \leftarrow \rho(i)$ according to the definition 10;
5     $L \leftarrow L \cup \{(i, \rho)\}$;
6 Sorting L in descending order according ρ components;
7 **return** a sublist of L with the first N elements;

Algorithm 4: Pseudo-code for calculating candidate list.

The computational complexity of this step can be studied as follows. This algorithm depends on the number of keys in the match table M, which is denoted by us as m, and the number of geometric transformations of each set T_i, which is denoted by us as t_i. The complexity of line 4 is $O(t_i^2)$, since it is the cost of computing the weights (see the summation of definition 9) and the recommendation score (see the maximum value of definition 10). Line 6 complexity is the cost of sorting a list with m elements, that is $O(m \log m)$. Thus, the complexity of algorithm 4 must be described using the following formula:

$$O\left(\sum_i t_i^2 + m \log m\right), \tag{7}$$

where $\sum_i$ sums over the keys of M. In section 7.3, we present an analysis of the values for t_i in an specific experiment. As we can see in Table 3, in the 96% of cases, the value of t_i is lesser than 10, and in the 99.36% of cases, the value of t_i is lesser than 40. Moreover, values of t_i greater than 41 are only considered for processing the template associated with the query (right hit); this fact only takes place, at most, once per query. It means that the number of transformations in the sets T_i is very small, in almost all occasions. Therefore, the computational cost for calculating recommendation score is quite insignificant.

With the description of the retrieving stage, we have concluded the last step for the description of our proposed indexing algorithm.

7. Experimental results

In this section, we describe and discuss the experiments made in order to evaluate the accuracy of our proposal, and we compare the results with some of the best state of the art algorithms.

7.1. Data sets description

In order to characterize our algorithm, experiments were conducted in the following well known datasets:

- NIST DB4: The NIST Special Database 4 contains 4000 8-bit gray scale rolled impressions from 2000 fingers (2 impressions per finger) [46]. The image size is 512 × 512 pixels and they are uniformly distributed in the five classes defined by Henry (arch, tended arch, whorl, left loop and right loop).

- NIST DB4 (natural): This dataset is a subset of NIST DB4 [46], and it was obtained by reducing the cardinality of the less frequent classes in nature, in order to resemble a natural distribution. In this way, the size of this dataset is decreased to 1204 impressions.

- NIST 14 (reduced): This dataset is composed by the last 2700 fingerprint pairs of NIST Special Database 14 [45]. This dataset contains rolled impressions of size 832 × 768 and its distribution resembles the fingerprint distribution in nature.

- FVC2000 DB2: The second FVC2000 dataset is composed by 800 fingerprints from 100 fingers (8 impressions per finger) [35]. These images were captured using a low-cost capacitive sensor "TouchChip" by ST Microelectronics. The size of the images is 256 × 364 pixels.

- FVC2000 DB3: The second FVC2000 dataset have 800 fingerprints from 100 fingers (8 impressions per finger) [35]. The images where captured using an optical sensor "DF-90" by Identicator Technology, resulting in images of 448 × 478.

- FVC2002 DB1: This dataset consists of 800 fingerprints from 100 fingers (8 impressions per finger) [36]. All of these images were captured using the optical optical sensor "TouchView II" by Identix, resulting in images of 388 × 374 pixels in 8-bit gray scale.

- FVC2004 DB1: This dataset is composed by 800 fingerprints from 100 fingers (8 impressions per finger) [37]. These image were captured with an optical sensor "V300" by CrossMatch, resulting in images of 640 × 480.

- FVC2006 DB2: The second FVC2006 dataset have 1680 fingerprints from 140 fingers (12 impressions per finger) [14]. The images where captured using an optical sensor, resulting in images of 400×560.

Table 2: The datasets and indexing methods for which published results are available.

Dataset	**Methods with published results**
NIST DB4	Germain et al. (1997) [15]
	Bhanu and Tan (2003) [3]
	Jiang et al. (2006) [21]
	Gyaourova and Ross (2008) [17]
	Capelli et al. (2011) [8]
	Liu et al. (2012) [33]
NIST DB4 (Natural)	Lumuni et al. (1997) [34]
	Capelli et al. (1999) [9]
	Lee et al. (2005) [25]
	Jiang et al. (2006) [21]
	Li et al. (2006) [26]
	Liu et al. (2006) [31]
	Liu et al. (2007) [32]
	Capelli et al. (2011) [8]
NIST DB14	Lumuni et al. (1997) [34]
	Capelli et al. (1999) [9]
	Capelli et al. (2002) [10]
	Capelli et al. (2011) [8]
FVC 2000 DB2	De Boer et al. (2001) [5]
	Jiang et al. (2006) [21]
	Liang et al. (2006) [27]
	Shuai et al. (2008) [44]
	Cappelli et al. (2011) [8]
FVC 2000 DB3	Jiang et al. (2006) [21]
	Capelli et al. (2011) [8]
FVC 2002 DB1	Feng and Cai (2006) [13]
	Liang et al. (2007) [28]
	Shuai et al. (2008) [44]
	He et al. (2009) [18]
	Capelli et al. (2011) [8]
	Liu et al. (2012) [33]
FVC 2004 DB1	Liang et al. (2007) [28]
	Zhang et al. (2008) [44]

In Table 2, we can see the indexing methods for which published results are available and the datasets in which they are reported.

7.2. Preprocessing and thresholds

To extract the features in our new algorithm, we use a minutia extraction method similar to the one reported in the state of the art [22]. This method computes black-white transition count around each point in the skeletonized image of each fingerprint. If the value of this count is 1 or 3, we will be in presence of a termination or a bifurcation, respectively. In this way, the minutiae are located, and their directions are computed from the associated ridges. In order to eliminate noise, the minutiae that are in the border of the impressions or in bad quality areas (false minutiae) are eliminated. We use a method already described in the literature for finding bad quality areas, which is based on the coherence and orientation maps [41] . Also, we developed a simple method to extract ridge counters between minutiae.

For testing our proposal, the thresholds described in section 6 are fixed to the following values: $e_1 = 2$ (two distortion errors, see definition 7), $E_\lambda = 0.25$, $E_\omega = 60°$, $e_\lambda = 0.125$, $e_\omega = 10°$, and $e_t = 15$ pixels (see definition 8).

7.3. Evaluation of our proposal

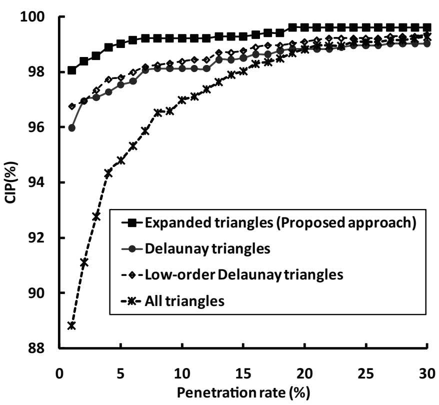
The Correct Index Power (CIP) is one of the most used and reliable measure for the evaluation of indexing algorithms accuracy. For this reason, in our experimentation the trade off between Penetration Rate (PR) and CIP is used to illustrate the results.

Formally, we can define the Correct Index Power and the Penetration Rate as: $CIP(N) = 100 \times c(N)/E$ and $PR(N) = 100 \times N/E$ respectively, where E is the number of experiments, and $c(N)$ is the number of times where the correct result is within the list with the first N hypothesis.

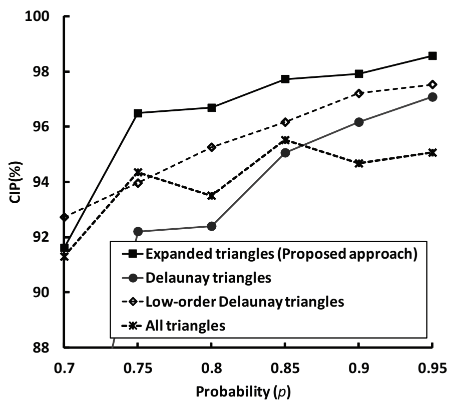
In order to evaluate the impact of our contributions we have conducted some experiments in the FVC 2006 DB2 dataset. In this way, we choose the first impression of each finger to conform the experimental collection with 100 fingerprints, while the other 11 prints of each finger are used as queries (1540 fingerprints are used as queries).

In Fig. 5(a), we can see our approach evaluated using different methods to select the triangle set. In this experiment, everything was fixed with the exception of the criterion for choosing triangles. As we can see, the variant that use the expanded triangle set proposed by us is better than the others. This occurs because, in our proposal, we solve the problem generated by missing and spurious minutiae, and we also represent fingerprints with a linear number of triangles (see section 4).

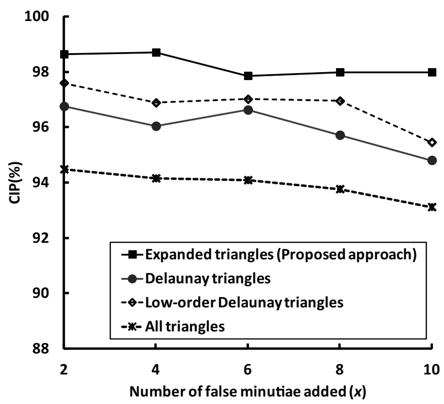
We also performed an exhaustive experimentation to prove the impact of our proposal in situations where some minutiae are missing, see Fig. 5(b). In order to simulate these conditions, we deliberately erase some of the minutiae obtained in the feature extraction process of each fingerprint. In this way, the following methodology was used: for each minutia we generate a random value between 0 and 1, that is used as the probability of keeping this minutia from the original minutia set. If this value is lesser than a predefined probability threshold p, the minutia is kept, otherwise it is removed. In Fig. 5(b), the results of these experimentations with different values of threshold p, and a value of penetration rate of 5% are shown. Thus, we empirically checked that the use of expanded triangles is the most immune option for facing the missing minutia distortions.



(a) FVC 2006 DB2



(b) Varying the probability (p) of keeping in a minutia in the original minutia set.



(c) Varying the number (x) of false minutiae added to the original minutia set.

Figure 5: Results of different methods for triplets selection.

In similar way, we tested our proposal for situations where some spurious minutiae are found. In this case, we added x false minutiae inside the area of each fingerprint. The coordinates and features of each false minutia are randomly generated. Some experimentations where conducted with different values of x and a penetration rate of 5%. The results are shown in Fig. 5(c). Thus, we can see that using expanded triangles we can achieve the best indexing results, under situations of appearance of spurious minutiae.

On the other hand, we prepared another experiment to justify the generation of a geometric transformation for each triangle side instead of a single one from the whole triangle, see Fig. 6(a). In this experiment, everything was fixed with the exception of the method for calculating set of transformations for each pair of triangles. Thus, we conclude that our proposal is a good choice for recovering index matches in fingerprint identification.

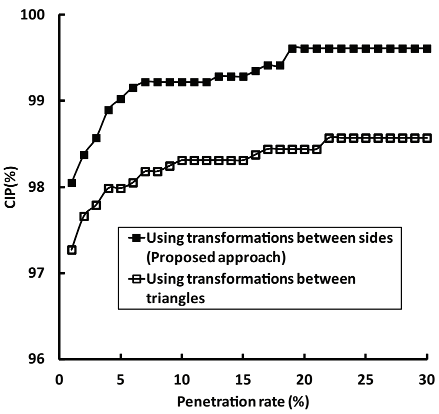
Moreover, we conducted an experiment to evaluate our proposal with 4 different values for the threshold e_1 (see definition 7). This threshold is very important because it has a great influence in the similarity function performance. In this experiment, everything was fixed with the exception of this threshold. As we can see in Fig. 6(b), the best accuracy was achieved with $e_1 = 2$.

Another experiment was done to prove the advantages of our defined similarity function between geometric transformations. In Fig. 6(c), we can appreciate our approach using the proposed similarity function and using fixed thresholds in order to decide if two transformations are similar. In the second case, instead of using weights to determine the recommendation score, a binary value is returned: 1 if the transformations are similar and 0 if they are not. Moreover, we include in this comparison the results of our approach but using the vote-based method for calculating such score. In this experiment, everything was fixed with the exception of the method for calculating the above mentioned score. The similarity function has a positive impact in the accuracy of our algorithm because provides a value of closeness between transformation components, in fuzzy terms.

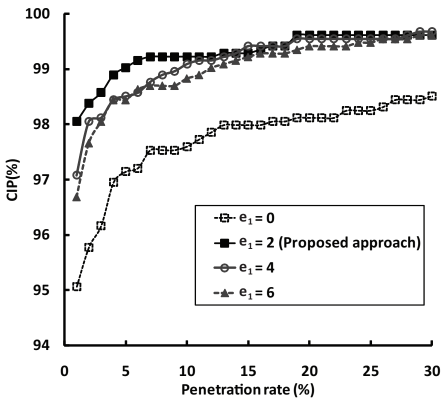
Table 3: The number of elements in the sets T_i checked in FVC 2006 DB2.

Number of transformations T_i (Intervals)	$[1, 10]$	$[11, 40]$	$[41, 200]$	$[201, \ldots)$
Wrong hits	96.16%	3.13%	0	0
Right hits	0.01%	0.06%	0.4%	0.25%
Total	96.17%	3.19%	0.4%	0.25%

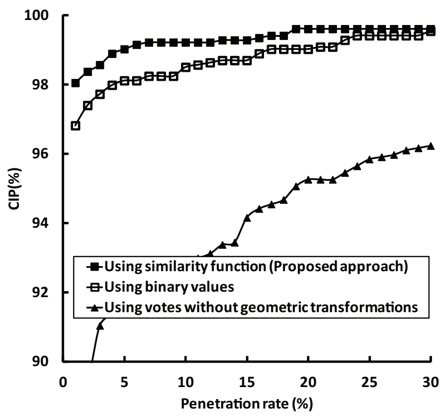
The last experiment was focussed on analysing the computational complexity for calculating the recommendation scores during the last step of retrieval stage, see Table 3. This experiment was performed by counting the number of geometric transformations in each set T_i for all executions of line 4 of the algorithm 4, and distributing this value among the four intervals considered in Table 3. For example, in the 96.17% of cases, the value of t_i is lesser than 10, see the last row and second column of the table. Moreover, we also count the number of cases of wrong or right hits. A right hit refers to a case where the set T_i belong



(a) Using different methods for calculating geometric transformations.

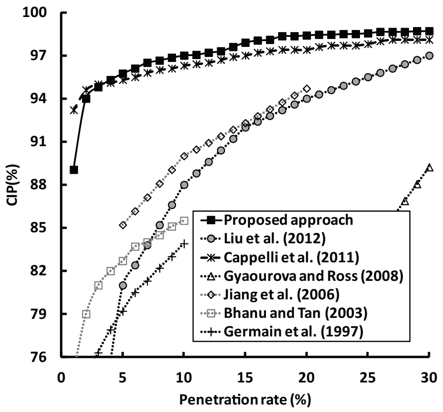


(b) Using different values for e_1.

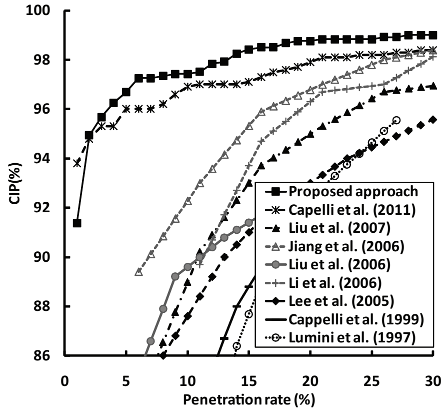


(c) Using different methods for calculating the recommendation score.

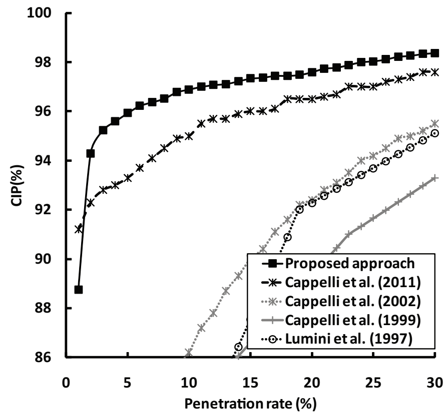
Figure 6: Results of other experiments for evaluating our proposal.



(a) NIST DB4



(b) NIST DB4 (natural)



(c) NIST DB14

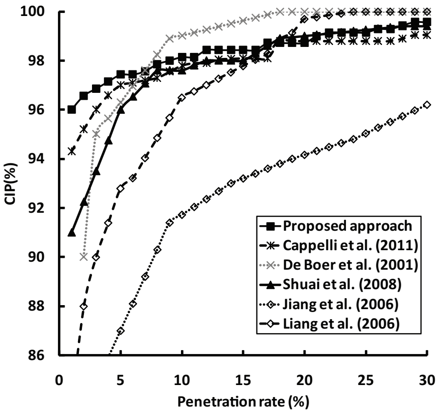
Figure 7: Results in NIST databases.

to the template associated with the query; the other cases were called wrong hits. For example, there are not computed values greater than 41 for wrong hits, see second row and the last two columns of the table. In summary, we can conclude that the number of transformations in the sets T_i is very small, in almost all cases. Therefore, the computational cost for calculating recommendation score is quite insignificant. This experiment was repeated in all databases presented in section 7.1; however, the conclusion was the same in all of the tests. For this reason, we only include the results in FVC 2006 DB2.

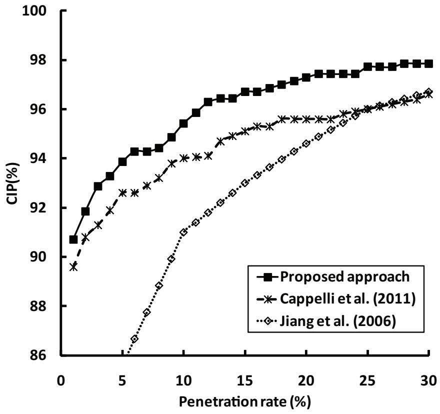
7.4. Comparison with other reported approaches

The results reported in the NIST datasets have been obtained by using the first impressions to build the experimental collection, while the second prints are used as queries to test the indexing performance.

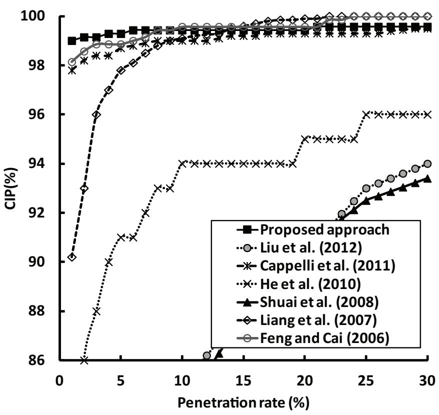
In Fig. 7(a), Fig. 7(b) and Fig. 7(c) we can see that our method is worse than Capelli et al. [8] only with very small values of penetration rate: less than 3 in NIST DB4 and less than 1 in NIST DB4 (natural). All



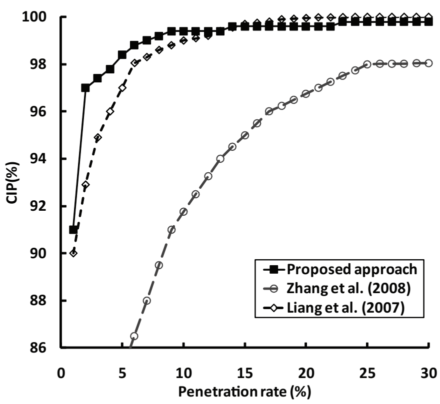
(a) FVC 2000 DB2



(b) FVC 2000 DB3



(c) FVC 2002 DB1



(d) FVC 2004 DB1

Figure 8: Results in FVC databases.

of the others algorithms are considerably less accurate than our approach.

In the FVC datasets, the results reported by all methods except for Liang et al. [28] and our approach in Fig. 8(d), have been obtained by selecting one impressions randomly for each finger to conform the experimental collection with 100 fingerprints, while the other seven prints for each finger are used as queries (700 fingerprints are used as queries). In the case of FVC 2004, the results shown by Liang et al. [28] and our approach (see Fig. 8(d)) were obtained using an experimental collection conformed by randomly choosing 3 impressions for each finger of the dataset while the other five prints for each finger are used as queries (500 fingerprints are used as queries). We adopt this experimental setup in order to show an impartial comparison between Liang et al. [28] and our proposal.

In Fig. 8(c), we can appreciate that Liang et al. [28] has better performance than our proposal with penetration rate (PR) values higher than 15. However this algorithm shows poor results in the experiments, for smaller values of PR. In addition, the difference of methodology in the experiment may have influenced the results. Also, we can see that our proposal is slightly less accurate than Feng and Cai (2006) [13] with PR values higher than 10, but this method adopts other features based on ridge while our approach only uses ridge counters.

The Fig. 8(a), Fig. 8(b) and Fig. 8(d), show the superior performance of our approach over most of the state of the art methods in FVC2000 DB2, FVC2000 DB3, and FVC2004 DB1 datasets. In the case of De Boer et al. [5], the good reported accuracy was obtained by combining three different algorithms. Also in one of these algorithms, in the 13 percent of the dataset the singular points where manually corrected and 1 percent of the dataset was eliminated since no registration point could be found. Our proposal also outperforms the method of Liang et al. [27] for values of penetration rate less than 18.

8. Conclusions

In this paper, a new fingerprint indexing algorithm based on minutia details and triplets is proposed. It has been shown that this algorithm is able to search a fingerprint dataset more efficiently and stably than previous triangle-based algorithms. Also, a new fingerprint representation is defined based on the Delaunay triangulations and triangular hulls. This representation is very robust under situations in which some minutiae are not detected. Also, the number of triangles of this representation is linear with respect to the number of minutiae in the fingerprint.

The proposed approach uses robust features combined with a new defined mechanism for dealing with the effects of noise, based on redundant feature vectors. A novel recovery strategy based on geometric transformations is also introduced. In this sense, a similarity function between geometric transformations is defined. Several experiments have been conducted in well known FVC and NIST databases. The obtained results show that our approach outperforms the majority of the best algorithms reported in the literature.

Only in a few cases the results reported by other methods are comparable with our proposal. This is due to the use of ridge based features or combinations of different methods.

Future work will be devoted to define a new representation of fingerprints more tolerant to elastic distortions, by combining higher order Delaunay triangles with our proposal. This will allow us to have an even more robust algorithm capable of dealing with displaced minutiae, using a relative small number of triangles.

References

[1] G. Bebis, T. Deaconu, and M. Georgiopoulos. Fingerprint Identification Using Delaunay Triangulation. In *Proceedings International Conference Information Intelligence and Systems (ICIIS'99)*, pages 452–459, Maryland, USA, 1999.

[2] M. Berg, M. Krevelt, M. Overmars, and O. Scharzkopf. Computational Geometry (Algorithms and Applications), Chapter 9. Springer-Verlag, Berlin Heidelberg, 1997.

[3] B. Bhanu and X. Tan. Fingerprint Indexing Based on Novel Features of Minutiae Triplets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5): 616–622, 2003.

[4] S. Biswas, N. Ratha, G. Aggarwal, and J. Connell. Exploring Ridge Curvature for Fingerprint Indexing. In *Proceedings of the 2nd IEEE International Conference on on Biometrics: Theory, Applications and Systems*, pages 1–6, Virginia, USA 2008.

[5] J.D. Boer, A.M. Bazen, and S.H. Cerez. Indexing Fingerprint Database Based on Multiple Features. In *Proceedings of the 12th Annual Workshop on Circuits*, pages 300–306, 2001.

[6] G. Bojadziev and M. Bojadziev. Fuzzy sets, fuzzy logic and applications. World Scientific, London, 1995.

[7] F.C. Botelho, Y. Kohayakawa, and N. Ziviani. A Practical Minimal Perfect Hashing Method. In *Proceedings of the 4th International Workshop on Efficient and Experimental Algorithms (WEA'05)*, pp. 488–500, 2005.

[8] R. Cappelli, M. Ferrara, and D. Maltoni. Fingerprint Indexing Based on Minutia Cylinder-Code. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5): 1051–1057, 2011.

[9] R. Cappelli, A. Lumini, D. Maio, and D. Maltoni. Fingerprint Classification by Directional Image Partitioning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5): 402–421, 1999.

[10] R. Cappelli, D. Maio, and D. Maltoni. A Multi-Classifier Approach to Fingerprint Classification. *Pattern Analysis and Applications*, 5(2): 136–144, 2002.

[11] R. Cappelli, M. Ferrara. A Fingerprint Retrieval System Based on Level-1 and Level-2 Features. Expert Systems with Applications, Vol. 39, Issue 12, pages 10465-10478, Sept. 2009.

[12] K. Choi, D. Lee, S. Lee, and J. Kim. An Improved Fingerprint Indexing Algorithm Based on the Triplet Approach. In *Proceedings of the 4th International Conference Audio and Video Based Biometric Person Authentication (AVBPA'03)*, pages 584–591, 2003.

[13] J. Feng and A. Cai. Fingerprint Indexing Using Ridge Invariants. In *Proceedings 18th International Conference on Pattern Recognition (ICPR'06)*, Vol. 4, pages 433-436, Hong Kong, 2006.

[14] Fingerprint verification Competition 2006, [WWW document] http://bias.csr.unibo.it/fvc2006, (Accessed 17th October 2011)

[15] R.S. Germain, A. Califano, and S. Colville. Fingerprint Matching Using Transformation Parameter Clustering. *IEEE Computing in Science and Engineering*, 4(4): 42–49, 1997.

[16] M.R. Girgisa, A.A. Sewisyb, R.F. Mansourb A robust method for partial deformed fingerprints verification using genetic algorithm. Expert Systems with Applications, Vol. 36, Issue 2, pages 20082016, March 2009.

[17] A. Gyaourova and A. Ross. A Novel Coding Scheme for Indexing Fingerprint Patterns. In *Proceedings of the 7th International Workshop on Statistical Pattern Recognition (S+SSPR)*, pages 765–774, 2008.

[18] S. He, C. Zhang, and P. Hao. Clustering-Based Descriptors for Fingerprint Indexing and Fast Retrieval. In *Proceedings of the 9th Asian Conference on Computer Vision (ACCV'09)*, Part I, pages 354–363, 2009.

[19] C. Hung, J. Liung, C. Yi Optical sensor measurement and biometric-based fractal pattern classifier for fingerprint recognition. Expert Systems with Applications, Vol. 38, Issue 5, pages 50815089, May 2011.

[20] T.Y. Jea and V. Govindaraju. A minutia-based partial fingerprint recognition system. *Pattern Recognition*, 38(10): 1672–1684, 2005.

[21] X. Jiang, M. Liu, and A.C. Kot. Fingerprint Retrieval for Identification. *IEEE Transactions on Information Forensics and Security*, 1(4): 532–542, 2006.

[22] S. Kasaei and B. Boashash. Fingerprint feature extraction using block-direction on reconstructed images. In *Proceedings of IEEE Region Conference on Speech and Image Technologies for Computing and Telecommunications (TENCON'97)*, pages 303–306, 1997.

[23] 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(11): 1266–1276, 2000.

[24] J. Kumar. A Clustering and Indexing Technique suitable for Biometric Databases. Indian Institute Of Technology, Kampur, 2009.

[25] S.O. Lee, Y.G. Kim, and G.T. Park. A Feature Map Consisting of Orientation and Inter-ridge Spacing for Fingerprint Retrieval. In *Proceedings of the 5th International Conference Audio and Video Based Biometric Person Authentication (AVBPA'05)*, pages 184–190, 2005.

[26] J. Li, W. Yau, and H. Wang. Fingerprint Indexing Based on Symmetrical Measurement. In *Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06)*, Vol. 1, pages 1038–1041, 2006.

[27] X. Liang, T. Asano, and A. Bishnu. Distorted Fingerprint Indexing Using Minutia Detail and Delaunay Triangle. In *Proceedings of the 3rd International Symposium on Voronoi Diagrams in Science and Engineering (ISVD'06)*, pages 217–223, 2006.

[28] X. Liang, A. Bishnu, and T. Asano. A Robust Fingerprint Indexing Scheme Using Minutia Neighborhood Structure and Low-Order Delaunay Triangles. *IEEE Transactions on Information Forensics and Security*, 2(4): 721–733, 2007.

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

[30] T. Liu, G. Zhu, C. Zhang, and P. Hao. Fingerprint Indexing Based on Singular Point Correlation. In *Proceedings of the International Conference on Image Processing (ICIP)*, Vol. 3, pages 293–296, 2005.

[31] M. Liu, X. Jiang, and A.C. Kot. Fingerprint Retrieval by Complex Filter Responses. In *Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06)*, Vol. 1, pages 1042–1046, 2006.

[32] M. Liu, X. Jiang, and A.C. Kot. Efficient Fingerprint Search based on Database Clustering. *Pattern Recognition*, 40(6): 1793–1803, 2007.

[33] M. Liu and P.T. Yap. Invariant Representation of Orientation Fields for Fingerprint Indexing. *Pattern Recognition*, 45(7): 2532–2542, 2012.

[34] A. Lumini, D. Maio, and D. Maltoni. Continuous versus Exclusive Classification for Fingerprint Retrieval. *Pattern Recognition Letters*, 18(10): 1027–1034, 1997.

[35] D. Maio, D. Maltoni, R. Cappelli, J.L. Wayman, and A.K. Jain FVC2000: Fingerprint Verification Competition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3): 402–412, 2002.

[36] D. Maio, D. Maltoni, R. Cappelli, J.L. Wayman, and A.K. Jain FVC2002: Second Fingerprint Verification Competition. In *Proceedings of the 16th International Conference on Pattern Recognition (ICPR'02)*, pages 811–814, 2002.

[37] D. Maio, D. Maltoni, R. Cappelli, J.L. Wayman, and A.K. Jain FVC2004: Third Fingerprint Verification Competition. In *Proceedings of the 1st International Conference on Biometric Authentication*, pages 1–7, 2004.

[38] P. Mansukhani, S. Tulyakov, and V. Govindaraju. A Framework for Efficient Fingerprint Identification Using a Minutiae Tree. *IEEE System Journal*, 4(2): 126–137, 2010.

[39] A. Muñoz, A. Gago, J. Hernández Fingerprint Indexing with Bad Quality Areas. To appear in: Expert Systems with Applications, DOI: http://dx.doi.org/10.1016/j.bbr.2011.03.031.

[40] L. Nanni, A. Lumini Descriptors for image-based fingerprint matchers. Expert Systems with Applications, Vol. 36, Issue 10, pages 1241412422, Dec 2009.

[41] U. Rajanna, A. Erol, and G. Bebis. A Comparative Study on Feature Extraction for Fingerprint Classification and Performance Improvements Using Rank-Level Fusion. *Pattern Analysis and Applications*, 13(3): 263–272, 2010.

[42] S. Ram, H. Bischof, J.A. Birchbauer. Modelling fingerprint ridge orientation using Legendre polynomials. *Pattern Recognition*, 43(1): 342–357, 2010.

[43] A. Ross and R. Mukherjee. Augmenting Ridge Curves with Minutiae Triplets for Fingerprint Indexing. In *Proceedings of the SPIE Biometric Technology for Human Identification IV*, DOI: 10.1117/12.720820, Orlando, USA, 2007.

[44] X. Shuai, C. Zhang, and P. Hao. Fingerprint Indexing Based on Composite Set of Reduced SIFT Features. In *Proceedings of the 19th International Conference on Pattern Recognition (ICPR'08)*, 2008.

[45] C.I. Watson. NIST Special Database 14, NIST Mated Fingerprint Card Pairs 2. National Institute of Standards and Technology, 1993.

[46] C.I. Watson and C.L. Wilson. NIST Special Database 4, Fingerprint Database. National Institute of Standards and Technology, 1992.

[47] Y. Zhang, J. Tian, K. Cao, P. Li, and X. Yang. Improving Efficiency of Fingerprint Matching by Minutiae Indexing. In *Proceedings of the 19th International Conference on Pattern Recognition (ICPR'08)*, 2008.