Fingerprint Identification Using Delaunay Triangulation

George Bebis†, Taisa Deaconu† and Michael Georgiopoulos‡

†Department of Computer Science, University of Nevada, Reno
‡Department of Electrical and Computer Engineering, University of Central Florida
{bebis, deaconu}@cs.unr.edu, mng@ece.engr.ucf.edu

Abstract

This paper presents a new indexing-based approach to fingerprint identification. Central to the proposed approach is the idea of associating a unique topological structure with the fingerprint minutiae using the Delaunay triangulation. This allows for choosing more "meaningful" minutiae groups (i.e., triangles) during indexing, preserves index selectivity, reduces memory requirements without sacrificing recognition accuracy, and improves recognition time. Specifically, assuming N minutiae per fingerprint on the average, the proposed approach considers only $O(N)$ minutiae triangles during indexing or recognition. This compares favorably to $O(N^3)$, the number of triangles usually considered by other approaches, leading to significant memory savings and improved recognition time. Besides their small number, the minutiae triangles we used for indexing have good discrimination power since, among all possible minutiae triangles, they are the only ones satisfying the properties of the Delaunay triangulation. As a result, index selectivity is preserved and indexing can be implemented in a low-dimensional space. Some key characteristics of the Delaunay triangulation are (i) it is unique (assuming no degeneracies), (ii) can be computed efficiently in $O(NlogN)$ time, and (iii) noise or distortions affect it only locally. The proposed approach has been tested on a database of 300 fingerprints (10 fingerprints from 30 persons), demonstrating good performance.

Keywords: fingerprint recognition, Delaunay triangulation, indexing

1. Introduction

Fingerprint matching is one of the most popular and reliable biometric techniques used in automatic personal identification. There are two main applications involving fingerprints: *fingerprint verification* [1][2] and *fingerprint identification* [3]-[5]. While the goal of fingerprint verification is to verify the identity of a person, the goal of fingerprint identification is to establish the identity of a person. Specifically, fingerprint identification involves matching a query fingerprint against a fingerprint database to establish the identity of an individual. To reduce search time and computational complexity, *fingerprint classification* is usually employed to reduce the search space by splitting the database into smaller parts [5][6]. Matching is usually based on lower-level features determined by singularities in the finger ridge pattern known as *minutiae*.

Given the minutiae representation of fingerprints, fingerprint matching can simply be seen as a point matching problem [7]-[9]. In this context, matching two fingerprints implies finding a subset of minutiae in the first fingerprint that best match to a subset of minutiae in the second fingerprint through a geometric transformation in an optimal sense (i.e., least-squares). Besides matching two fingerprints together, the main issue when dealing with large fingerprint databases is how to select the most similar fingerprints to the query fingerprint from the database. Both of these problems appear very often in computer vision, particularly, in object recognition. As a result, methods for object recognition have much in common with fingerprint identification methods.

Indexing-based methods [10]-[12] are quite attractive when dealing with large object databases, since the query object does not have to be compared with every other object in the database. Specifically, indexing is a mechanism which, when provided with a key value, is able to rapidly access some associated data. Thus, instead of having to search the space of all possible matches and explicitly reject invalid ones, indexing inverts the process so that only the most feasible matches are considered. The main idea behind indexing is to prestore information about the models in a table. For each model, groups of features are extracted and an index is computed from each group. Information about each group is then stored in the indexed location. During recognition, the information stored in the table is used to quickly eliminate non-compatible matches. Recently, Germain et al. have proposed using the FLASH algorithm [4], an indexing-based object recognition algorithm, for fingerprint identification.

There are important issues to be considered when using indexing for fingerprint identification including the issues of memory requirements and index selectivity. In terms of memory requirements, the number of table entries will be of the order of $O(N^S)$, where N is the

0-7695-0446-9/99 $10.00 © 1999 IEEE

average number of object features and S is the size of groups. Fingerprints usually contain between 30 and 80 minutiae [1][2], a number which is higher than the average number of features (e.g., curvature extrema) found in a typical object. This, along with the common practice to store more than one imprints of the same finger in the database to improve robustness and accuracy [1][2], leads to much higher space requirements.

The issue of index selectivity relates to the discrimination power of the groups considered for indexing. Groups with low discrimination power give rise to very similar indices (low index selectivity). As a result, a large number of hypothetical matches are generated during recognition, making indexing ineffective. The main cause of reduced index selectivity is the small size of groups used for indexing and that almost every possible group is considered for indexing. One way to deal with this problem is by increasing the index dimensionality using larger size groups, however, this will also increase memory requirements since the number of groups increases exponentially with size. Alternatively, additional information can be computed from each group and added to the index to increase its dimensionality. FLASH is based on this idea (7-dimensional indices were used in [11] and 9-dimensional in [4]). Although this approach is effective, it also increases time requirements and raises the issue of computing the additional information fast and reliably.

In this paper, we propose a new approach to fingerprint identification based on indexing and the Delaunay triangulation [13][14]. The problem of triangulation is a fundamental one in computational geometry with applications in surface or function interpolation. Here, the Delaunay triangulation is used to associate a unique topological structure with the fingerprint minutiae. The goal is to use the Delaunay minutiae triangles for indexing. This yields reduced memory requirements without sacrificing recognition accuracy, preserves index selectivity without resorting to high-dimensional indexing schemes, and improves recognition time. It can be shown that if N is the number of minutiae, the Delaunay triangulation produces $O(N)$ triangles [14][15]. Thus, the number of table entries will be of the order of $O(N)$. This compares favorably to $O(N^3)$, the number of all possible triangles considered by other approaches [4][10]. In addition, the Delaunay minutiae triangles have good discrimination power since, among all possible triangles, they are the ones satisfying the properties of the Delaunay triangulation [14][15]. This, along with their small number, leads to faster recognition and low-dimensional indexing.

The key characteristics of the Delaunay triangulation of a set of points is that it is unique. Also, it can be computed efficiently in $O(NlogN)$ time [16]. One problem is that it is sensitive to noise and distortions (e.g, introduced by missing or spurious minutiae points), however, both noise and distortion have only a local effect on it. This means that correct identification will be possible if some region of the fingerprint has not been seriously affected.

The paper is organized as follows: in Section 2, we discuss the use of indexing for fingerprint identification. Section 3 reviews the Delanuay triangulation and Section 4 describes how the Delaunay triangulation is used for fingerprint identification. Section 5 presents our experimental results and finally, our conclusions are given in Section 6.

2. Using indexing for fingerprint identification

The problem of fingerprint identification has much in common with the problem of 2D *model-based* object recognition where recognition relies upon the existence of a set of predefined models. Given an unknown scene, recognition implies: (i) the identification of a set of features from the scene which approximately match a set of features from a model, (ii) the recovery of the geometric transformation that the model has undergone and, (iii) verification that other features coincide with the predictions. Similarly, fingerprint identification refers to the process of matching a query fingerprint against a fingerprint database in order to establish the identity of an individual. In both cases, the goal is to quickly determine if an object or fingerprint is in the database and to retrieve those objects or fingerprints which are most similar with the unknown scene or query fingerprint. Since usually there is no *a-priori* knowledge of possible feature correspondences, matching can be computationally too expensive, even for a moderate number of models in the database.

Indexing has received considerable attention in the literature [10]-[12] since it does not require considering each model separately, thus, it is less dependent on the database size. Indexing-based methods have two phases of operation: *preprocessing* and *recognition*. During preprocessing, features which remain unchanged under geometric transformations (*invariants*) are extracted from groups of model points and used to form indices. The indexed locations are filled with entries containing references to the models. During recognition, features from groups of image points are extracted and used to form indices again. The models listed in the indexed entries are collected into a list of candidate models and the most often indexed models are selected for further verification. Verification works by computing the transformation between the model(s) and the image and then by aligning the model(s) with the image using the computed transformation. Then, the similarity of the model with the image is estimated (e.g., by finding the percentage of model features that have been aligned with image features).

Although indexing is an attractive approach, very often it becomes less effective due to limited index selectivity. The heart of the problem is the low dimensionality of the invariants used to form the indices. In addition, indexing has high memory requirements. In the case of fingerprints, memory requirements can become much higher since fingerprints contain more features on the average than typical objects. Indexing-based methods usually consider every possible group of features (of a specific size) for building the table. There are two main reasons for this: first, it is desirable to build some degree of redundancy in the table so recognition can become more robust and second, there is usually no *a-priori* knowledge for choosing certain groups over others. Although redundancy can improve robustness, redundancy with limited index selectivity increase false positives, slowing recognition time significantly.

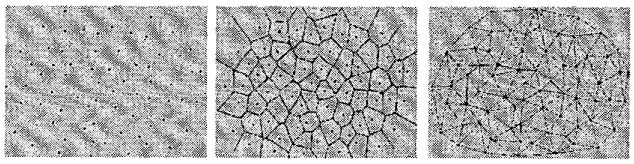
One way to deal with the problem of limited index selectivity is by choosing larger size groups. However, this will further increase memory requirements since the number of groups increases exponentially with size [20]. To get around this problem, "grouping" has been suggested in object recognition to identify important groups of features only [21]. Using grouping in fingerprint recognition, however, will not be a good idea since the minutiae have a rather random distribution. Another idea to improve index selectivity is by adding new invariants to the index, thus, increasing its dimensionality. The FLASH algorithm is based on this idea [11]. In [4], the FLASH algorithm was used for fingerprint identification. FLASH considers triangles of minutiae to compute a 9-dimensional index which includes information about the lengths of the sides of the triangle formed by the triangle, the ridge count between each pair, and angle information. Although the idea of using high-dimensional invariants does improve index selectivity, new issues arise since we need to consider how the high-dimensional invariants will be computed fast and reliably.

In this paper, we propose a new indexing-based approach to fingerprint identification. Central to the new approach is the idea of associating a unique topological structure with the minutiae using Delaunay triangulation. The minutiae triangles of the Delaunay triangulation are then used for indexing. There are several advantages behind this idea. First of all, we only consider $O(N)$ triangles for indexing, implying lower memory requirements and less redundancy. Second, the minutiae triangles of the Delaunay triangulation have good discrimination power since, among all possible triangles, they are the only ones satisfying the properties of the Delanuay triangulation. The improved index selectivity along with the less redundancy of the information stored in the table yield less false positives and improve recognition time. Finally, indexing can be implemented in a low-dimensional space.

3. Background on Delaunay triangulation

Triangulation is a process that takes a region of space and divides it into subregions. The space may be of any dimension, however, a 2D space is considered here since we are dealing with 2D points (minutiae). In this case, the subregions are simply triangles. Triangulation has many applications in finite elements simulation, surface approximation and nearest neighbor identification [16]. Here, however, our goal, is to associate a 2D topological structure with the minutiae.

Given a set S of points $p_1, p_2, \ldots, P_N$, we can compute the Delaunay triangulation of S by first computing its *Voronoi* diagram. The Voronoi diagram decomposes the 2D space into regions around each point such that all the points in the region around p_i are closer to p_i than they are to any other point in S. Given the Voronoi diagram, the Delaunay triangulation can be formed by connecting the centers of every pair of neighboring Voronoi regions. Figure 1a shows a set of 2D points, their Voronoi diagram is shown in Figure 1b while their Delaunay triangulation is shown in Figure 1c. Delaunay triangulation has certain properties, including: (1) the Delaunay triangulation of a non-degenerate set of points is unique, (2) a circle through the three points of a Delaunay triangle contains no other points and (3) the minimum angle across all the angles in all the triangles in a Delaunay triangulation is greater than the minimum angle in any other triangulation of the same points.



(a) (b) (c)

Figure 1. (a) A set of points, (b) its Voronoi diagram, and (c) its Delaunay triangulation.

Property 1 supports the use the Delaunay triangles for indexing. Property 2 implies that the insertion of a new point in a Delaunay triangulation affects only the triangles whose circumcircles contain that point. As a result, noise affects the Delaunay triangulation only locally. This is very important in the context of our application. The last property implies that the triangles obtained are not "skinny". This is also very desirable in our application since the computation of the geometric transformations between fingerprints is based on corresponding minutiae triangles. Using skinny triangles can lead to instabilities and errors [10]. In a comparison study that involved several well known topological structures [18], the Delaunay triangulation was found to have the best structural stability under random positional perturbations.

The Delaunay triangulation and the Voronoi diagram are very efficient algorithms since the number of edges in both of them is proportional to a small constant times the number of points ($O(N)$). Since each edge belongs to at most two triangles or polygons, then the number of triangles generated by the Delaunay triangulation is also linear to the number of points. In our experiments, we have used Fortune's implementation which is available from http://netlib.bell-labs.com/netlib/voronoi. The complexity of the algorithm is $O(Nlog(N))$.

4. Indexing using Delaunay triangulation

4.1. Minutiae triangulation

The proposed fingerprint identification system represents fingerprints in terms of their minutiae. The two most prominent minutiae, which are also the ones used here, correspond to ridge endings and ridge bifurcations. Each minutiae is represented by its coordinates (x, y). Once the minutiae have been extracted (we use the algorithm in [5]), their Delaunay triangulation is computed. Figure 2 demonstrates the Delaunay triangulation of the minutiae extracted from one of the fingerprints in our database.

Figure 2. The Delaunay triangulation of the minutiae.

4.2. Building the index table

The index table is built by considering the minutiae triangles formed by the Delaunay triangulation. Before deciding what invariants will be computed from each minutiae triangle, the geometric transformation that relates different fingerprint instances should be defined. Usually, it is assumed to be a rigid or similarity transformation [2]-[4][5]. In this paper, we assume similarity transformations (translation, rotation, and scaling) with a refinement step based on affine transformations (see Section 4.4). From each minutiae triangle, information invariant to similarity transformations is thus computed. Then, an index is formed using the invariants and appropriate information is stored in the indexed table location.

Without using the Delaunay triangulation, we would have to consider every possible triangle. Assuming N minutiae on the average, the number of possible triangles is $O(N^3)$. In contrast, the Delaunay triangulation yields only $O(N)$ triangles. Since these triangles satisfy the properties of the Delaunay triangulation, they can be found through a well defined procedure and have good discrimination power. Using these triangles for indexing preserves index selectivity and allows for implementing a low dimensional indexing scheme.

Given a minutiae triangle (e.g., see Figure 3), we compute three invariants which are then used to form a 3-dimensional index. The invariants are based on the sides and angles of the minutiae triangle. First of all, we sort the sides of the triangle to avoid considering all possible orders of three points:

$$l_1 \le l_2 \le l_3$$

Then, the following invariants are computed:

$$0 \le \frac{l_1}{l_3} \le 1$$

$$0 \le \frac{l_2}{l_3} \le 1$$

$$-1 \le cos(A) \le 1$$

where A is the angle between the smallest two sides.

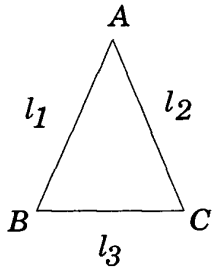


Figure 3. Invariants using the minutiae triangles.

The reason for using the cosine of the angle and not the angle itself is because the value of the angle is sensitive to noise introduced by the minutiae extraction algorithm while the cosine can filter out part of that noise. It should be mentioned that the angle we consider for indexing is the largest of the three angles in the triangle under consideration. Obviously, very large angles yield triangles whose points are almost collinear. Although the Delanuay triangulation tends to avoid such "skinny" triangles as mentioned in Section 3, this cannot always be guaranteed (unless extra points are inserted in the minutiae set, e.g., see [16]). Such triangles are not desirable since the computation of the geometric transformation becomes unstable (small errors in the minutiae locations yield large errors in the computation of the parameters of the transformation). Thus, we reject triangles whose largest angle is greater than a threshold (168 degrees).

After the invariants have been computed, linear scaling followed by quantization yields an integer index. For each index computed, information about the fingerprint and its minutiae is stored in the index table. Specifically, the entries stored in the table have the following format:

$$(person\_ID, imprint\_ID, m_1(x, y), m_2(x, y), m_3(x, y))$$

where *person_ID* corresponds to an identification code for the person, *print_ID* is an identification code for the particular imprint of that person (i.e., each person can have more than one imprints stored in the database), and $m_i(x, y)$ are the (x, y) coordinates of the m_i point in the group of minutiae.

Fingerprint images are usually very noisy due to various factors such as fingerprint morphology and imaging conditions. Also, certain amount of noise is introduced by the minutiae extraction process. In order to account for variations in the fingerprint images of the same finger, it is often imperative to store in the database information from several different images of the same finger taken at different times. Although this increases memory requirements, it makes the system more robust to noise and distortions. In Section 5, we report a number of experiments by varying the number of imprints stored in the database.

4.3. The identification step

During identification, each index generated by a query fingerprint is used to retrieve all model fingerprints stored in the database under the same index. While processing the query fingerprint, the minutiae points are extracted and their Delaunay triangulation is computed. For each Delaunay minutiae triangle, the lengths of the sides are calculated, sorted in ascending order, and the invariants are computed. Then, the invariants are quantized as in preprocessing. The resulting index is used to retrieve from the database all the entries stored at the same index table location. To account for noise, we also retrieve entries stored in a small neighborhood (i.e., a circle of radius 2) around the indexed location.

Most indexing-based approaches accumulate evidence about a model by casting a vote for every entry stored in the indexed locations and by "histograming" the entries to pick the ones which have received a high number of votes [10]. The problem with this approach is that it takes into consideration only the number of votes received by a particular entry and does not consider whether these votes are consistent among themselves. To introduce a measure of coherence, Lamiroy and Gros [19] have proposed voting in the transformation space. The key idea behind this approach is to consider transformations which form large clusters in the transformation space. The same idea was also used in [4].

We have also adopted this idea in our work since it is very effective. Specifically, each of the entries retrieved from the index table represents a hypothesized correspondence between three minutiae in the query fingerprint and three minutiae in the model fingerprint. Given this information, the transformation that best maps the query triangle to the model triangle is computed. The computed transformation parameters are binned and, along with the *person_ID* and *imprint_ID*, form a key that indexes another data structure used for evidence accumulation. An 8-dimensional integer array was used in order to store the number of votes in the transformation space (six dimensions for the parameters of the transformation, one for the *person_ID* and one for the *imprint_ID*).

If a large number of minutiae can be brought into correspondence by a transformation, then the indices generated by the triangles formed by those minutiae will generate the same or very similar transformation parameters. Hence, a larger number of votes for a correct match will be accumulated. There might be a number of random correspondences between minutiae triangles in the query fingerprint and some model fingerprint, however, the likelihood of a number of consistent transformation parameters being generated by random correspondences is small, and the verification step will eliminate most of them.

4.4. The verification step

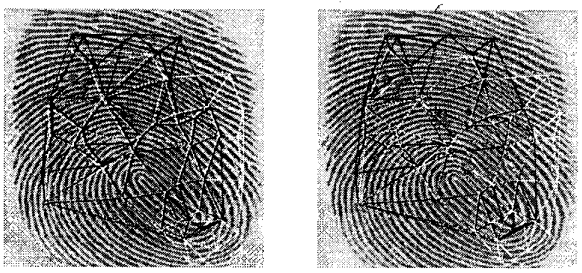
The transformations that are further considered for verification are the ones with the 4 largest number of votes. The verification stage determines whether two fingerprints correspond to the same finger or not. This is performed by aligning the two fingerprints using the transformation computed in the previous step and by computing the amount of overlap. Specifically, given a query fingerprint, a list of candidate fingerprints which possibly match the query fingerprint is generated. For each candidate match, a transformation is computed. Then, the computed transformation is applied on the candidate fingerprint to align it with the query fingerprint. If a large number of minutiae from the candidate fingerprint are "close" (i.e., less that 15 pixels) to a large number of minutiae from the query fingerprint, then it is very likely that the two fingerprints come from the same finger. To compute the percentage of overlap, we use the following formula:

$$p = \frac{2n}{m+q} \; x \; 100$$

where n is the number of matched minutiae, m is the number of minutiae in the candidate fingerprint and q is the number of minutiae in the query fingerprint.

Although we use similarity transformations to related different fingerprint instances, differences in the

pressure of the figure on the sensor or skin elasticity produce deformations which are not modeled very well by similarity transformations. In our experiments, we have found that a refinement of the computed similarity transformations using affine transformations can align the fingerprints more accurately. Based on this observation, our verification procedure has two stages. In the first stage, the two fingerprints are aligned through a similarity transformation since the invariants computed are invariants to similarity transformations. In the second stage, however, we find additional minutiae correspondences and we attempt improve the alignment by computing an affine transformation. Figure 4a shows the alignment of two fingerprints using just three minutiae and similarity transformation. Figure 4b shows the aligned fingerprints using more minutiae correspondences and affine transformation.



(a) (b)

Figure 4. Aligning the two fingerprints using a similarity transformation, (b) improving the alignment through an affine transformation.

In each stage, a different threshold is used to define the percentage of minutiae from the candidate fingerprint that are "close" to minutiae from the query fingerprint. In particular, a smaller threshold (20%) is used in the first stage to make sure that we consider as many candidate matches as possible, thus, reducing the number of false negatives. In the second stage, however, we filter out the false positives introduced by the first stage by using a higher threshold (40%). The threshold used to define the "closeness" between minutiae is also different is each stage (10 pixels in stage one and 15 pixels in stage two).

5. Experimental results

5.1. The data set

The fingerprint images used in this study have been captured using an inkless fingerprint scanner. Our database contains 300 fingerprints, captured from 30 individuals (10 images per finger for each individual). The size of these images is 400 by 400 pixels. When these fingerprint images were captured, no restriction on the position and the orientation of fingers were imposed.

5.2. Experiments

To characterize system's performance, we have conducted several experiments. In the first set of experiments, we vary the number of imprints stored in the database for each person. Thus, a subset of the 300 fingerprint images was used to build the database while the rest images were used for testing. We have experimented with storing 3, 5, and 7 images per person in the database. In each case, six experiments were conducted. In the first five experiments, the images stored in the database were chosen randomly while in the last experiment, the "best" images were chosen (in terms of image quality according to our opinion).

We classify our results in four categories: (a) *correct*: the query fingerprint has been correctly matched to one or more fingerprints from the same person, (b) *false positive*: the query fingerprint has been matched to one or more fingerprints from an incorrect person (c) *false negative*: the query fingerprint has not been matched to any fingerprint in the database (we assume that the database contains fingerprints from each person), and (d) *mixed*: there is not enough evidence to assign the query fingerprint to one of the previous three categories. The reason we have mixed matches is because we store more than one imprints in the database for each person. Usually, the query fingerprint is matched to more than one imprints from the same person (see Figure 5). Sometimes, however, the list of matches contains fingerprints from other persons as well. We call this case a "mixed" match. Mixed matches require further processing. Here, we resolve the mixed results using a "majority" rule. According to this rule, we assign the query fingerprint to the individual with the maximum number of imprints in the list of matches. In our experiments, we were able to resolve all mixed matches correctly using this rule (the percentage of mixed matches is shown in the last column for completeness).

(a) (b)

Figure 5. Several imprints from the same finger match the same query image; (a) imprint1, (b) imprint2. The black lines correspond to the query image while the white lines correspond to the model.

Tables 1-3 show the results obtained storing various number of imprints (3, 5, and 7) in the database for

each person. From these results, we can infer that the recognition accuracy depends on the number of imprints stored in the database for each person. In particular, the last row of each table shows that if the imprints stored in the database are of good quality, then recognition accuracy can be improved significantly. The number of false negatives are relatively high compared to the number of false postives. Obviously, the threshold used for matching (40% of points should match) has some effect on this. Although this threshold was chosen experimentally here, we plan to optimize its choice in our future work.

Table 1. Stored: 3 imprints per person; Tested: 210.

	Results			
Trial	Correct	False Positive	False Negative	Mixed
1	86.6%	0.4%	13%	0%
2	87%	0%	13%	0%
3	87%	0%	13%	0%
4	86.2%	0%	13.8%	0%
5	86%	0%	14%	0%
Average	86.56%	0.08%	13.36%	0%
Best	96.7%	0%	3.3%	0%

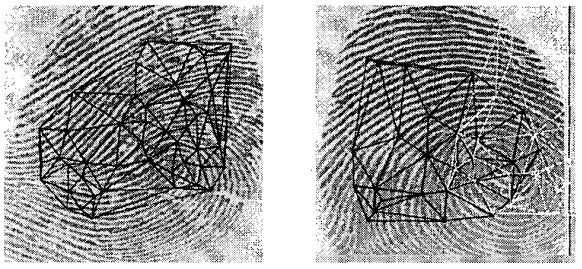
Table 2. Stored: 5 imprints per person; Tested: 150.

	Results			
Trial	Correct	False Positive	False Negative	Mixed
1	93%	0%	7%	0.6%
2	95.4%	0.6%	4%	0.6%
3	93.4%	0%	6.6%	2.6%
4	92%	0%	8%	1.3%
5	92%	0%	8%	0.6%
Average	93.16%	0.12%	6.72%	1.14%
Best	99.4%	0%	0.6%	0%

Table 3. Stored: 7 imprints per person; Tested: 90.

	Results			
Trial	Correct	False Positive	False Negative	Mixed
1	90%	0%	10%	3%
2	95%	0%	5%	3%
3	97%	0%	3%	1%
4	93%	0%	7%	3%
5	96%	0%	4%	1%
Average	94.2%	0%	5.8%	2%
Best	100%	0%	0%	0%

We have also tested all possible combinations of storing 9 fingerprints from each person in the database. Recognition accuracy improved even more in this case, however, our results indicate that there are certain fingerprints that are very difficult to identify. The main reasons for this are: (i) the quality of the query fingerprint is so bad that it does not resemble very well any of the other 9 fingerprints and (ii) the query fingerprint does not have many minutiae in common with the rest of the fingerprints (e.g., two fingerprints might correspond to different poses of the finger on the scanner or the minutiae extraction algorithm failed to detect certain minutiae). Figure 6 shows examples of these cases.



(a) (b)

Figure 6. (a) The query fingerprint is of bad quality, (b) the query fingerprint has a small number of minutiae in common with other fingerprints from the same finger (the black lines correspond to the query fingerprint).

In the next experiment, we wanted to test how false positives increase with the database size. Also, we were interested in testing how the system performs on fingerprints from people not represented in the database. In order to test this assumption, we fixed the number of query fingerprints to 50 by randomly choosing 5 persons out of the 30 persons contained in our database. Then, we used the fingerprints of the rest 25 persons (25 x 10 = 250) to build the database. Five experiments were conducted by storing in the database 250, 200, 150, 100 and 50, randomly chosen, fingerprints.

As can be seen from Table 4, false positives increase slowly with the database size. Usually, if a query fingerprint has a match in the database, there will be no room for false positives since the matched model will receive a large number of votes. However, if a match does not exist, we have noticed that false alarms usually occur at around the threshold value (all the false alarms encountered in our experiments had less than 45% common points while the threshold was 40%). One way to improve the results is by using additional information for verification. Currently, we just use the minutiae locations for verification. However, additional information such as local orientation can improve the results. Another way is to increase the threshold for the number of matched minutiae but this will of course increase the number of false negatives. The answer to this dilemma depends on the application.

The response time of our system depends on the query fingerprint. If the query fingerprint has a match in the database, then the response time is of order of a few seconds (usually, 4-5 - no code optimization was performed, all the experiments were run on an Ultra Sun 30). In this case, a few hypotheses are generated. However, if a match does not exist, then the response time can double or even triple. In this case, a large list of hypothe-

ses are created which are then subject to verification.

Table 4. False positives versus database size assuming that the query fingerprints correspond to people not represented in the database.

Results	
# images in the database	Average # of False Positives (5 trials)
50	0.6(1.2%)
100	0.8(1.6%)
150	1(2%)
200	1.4(2.8%)
250	1.6(3.2%)

6. Discussion and conclusions

We have proposed a indexing-based new approach to fingerprint identification using the Delaunay triangulation. The most important characteristics of the proposed approach are: low storage requirements, improved index selectivity, low dimensional indexing requirements, and fast identification. Our results indicate that the accuracy of a fingerprint recognition system can be improved by storing in the database imprints of good quality. This is a reasonable assumption since in most applications, the quality of the imprints is (or can be) controlled during acquisition. Our experiments have shown that most misses occur in the case of images of poor quality or images that do not have a good representant in the database.

References

[1] A. Jain, L. Hong, and R. Bolle, "On-line fingerprint verification", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 302-314, 1997.

[2] P. Baldi and Y. Chauvin, "Neural networks for fingerprint recognition", *Neural Computation*, vol. 5, pp. 402-418, 1993.

[3] B. Mehtre, "Fingerprint image analysis for automatic identification", *Machine Vision and Applications*, vol. 6, pp. 124-139, 1993.

[4] R. Germain, A. Califano, and S. Colville, "Fingerprint matching using transformation parameter clustering", *IEEE Computational Science and Engineering*, pp. 42-49, October-November 1997.

[5] N. Ratha, K. Karu, S. Chen and A. Jain, "A real-time system for large fingerprint databases", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp. 799-813, 1996.

[6] R. Cappelli, A. Lumini, D. Maio, and D. Maltoni, "Fingerprint classification by directional image partitioning", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 402-421, 1999.

[7] S. Ranade and A. Rosenfeld, "Point pattern matching by relaxation", *Pattern Recognition*, vol. 12, pp. 269-275, 1980.

[8] J. Starink and E. Backer, "Finding point correspondences using simulated annealing", *Pattern Recognition*, vol. 28, no. 2, pp. 231-240, 1995.

[9] G. Bebis, S. Louis, and Y. Varol, "Using genetic algorithms for model-based object recognition", *International Conference on Imaging Science, Systems, and Technology*, pp. 1-6, 1998, Las Vegas.

[10] Y. Lamdan, J. Schwartz, and H. Wolfson, "Affine invariant model-based object recognition", *IEEE Trans. on Robotics and Automation*, vol. 6, no. 5, pp. 578-589, October 1990.

[11] A. Califano and R. Mohan, "Multidimensional indexing for recognizing visual shapes", *IEEE Pattern Analysis and Machine Intelligence*, vol. 16, no. 4, pp. 373-392, 1994.

[12] G. Bebis, M. Georgiopoulos, M. Shah, and N. La Vitoria Lobo, "Indexing based on algebraic functions of views", *Computer Vision and Image Understanding*, vol. 72, no. 3, pp. 360-378, 1998.

[13] F. Aurenhammer, "Voronoi diagrams - A survey of a fundamental geometric data structure", *ACM Computing Surveys*, vol. 23, no. 3, pp. 345-405, 1991.

[14] N. Ahuja, "Dot pattern processing using voronoi neighborhoods", *IEEE Pattern Analysis and Machine Intelligence*, vol. 4, no. 3, pp. 336-343, 1982.

[15] F. Preparata and M. Shamos, *Computational Geometry*, Springer-Verlag, NY, 1985.

[16] S. Skiena, *The algorithm design manual*, Springer-Verlag, NY, 1998.

[17] R. Sibson, "The Dirichlet tessellation as an aid in data analysis", *Scandinavian Journal of Statistics*, vol. 7, pp. 14-20, 1980.

[18] M. Tuceryan and T. Chorzempa, "Relative sensitivity of a family of closest-point graphs in computer vision applications", *Pattern Recognition*, vol. 24, no. 5, pp. 361-373, 1991.

[19] B. Lamiroy and P. Gros, "Rapid object indexing and recognition using enhanced geometric hashing", *European Conference on Computer Vision (ECCV)*, pp. 59-70, 1996.

[20] D. Clemens and D. Jacobs, "Space and time bounds on indexing 3D models from 2D images", *IEEE Pattern Analysis and Machine Intelligence*, vol. 13, no. 10, pp. 1007-1017, 1991.

[21] D. Jacobs, "Robust and efficient detection of convex groups", *IEEE Pattern Analysis and Machine Intelligence*, vol. 18, no. 1, pp. 23-37, 1996.