Augmenting Ridge Curves with Minutiae Triplets for Fingerprint Indexing

Arun Ross and Rajiv Mukherjee
Lane Department of Computer Science and Electrical Engineering,
West Virginia University, Morgantown, WV 26506

ABSTRACT

Given a query fingerprint, the goal of indexing is to identify and retrieve a set of candidate fingerprints from a large database in order to determine a possible match. This significantly improves the response time of fingerprint recognition systems operating in the identification mode. In this work, we extend the indexing framework based on minutiae triplets by utilizing ridge curve parameters in conjunction with minutiae information to enhance indexing performance. Further, we demonstrate that the proposed technique facilitates the indexing of fingerprint images acquired using different sensors. Experiments on the publicly available FVC database confirm the utility of the proposed approach in indexing fingerprints.

Keywords: biometrics, fingerprints, indexing, minutiae triplets, ridge curve, classification, filtering, clustering.

1. INTRODUCTION

The problem of automatic fingerprint identification involves comparing a query print, q, with fingerprint entries, $\mathcal{D} = \{d_1, d_2, d_3, ...d_n\}$, in a database in order to determine the identity y of the print. Each entry d_j, is assumed to be associated with an identity y_j and, hence, $y = y_k$ where $k = \arg\max_{j=1}^{n}\{S(q, d_j)\}$ and S is the matching function that assesses the similarity between two prints. The computational complexity of the identification process is primarily dictated by the number of entries, $\mid D \mid = n$, in the database. In order to reduce the number of matching operations, a filtering procedure is usually invoked to identify a subset R of prints ($R \subset D$) such that $|R| = m << n$. Filtering can be accomplished using two distinct approaches: classification and indexing. A classification scheme partitions the database of fingerprints into multiple classes and, therefore, the query print is compared only against those prints in the database belonging to the same class as the query print. Indexing schemes, on the other hand, assign an index value* to each fingerprint and, hence, the query print is compared against those prints in the database that have comparable index values. The primary difference between classification and indexing is the use of continuous values in the former.

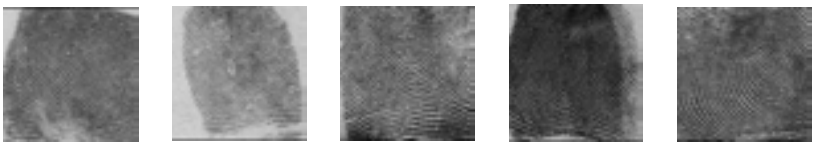
Fingerprint classification schemes based on human defined categories (such as Left Loop, Right Loop, Arch, Tented Arch and Whorl: see Figure1) have an inherent problem due to the small number of classes (e.g., 5 - 8) and the uneven distribution of fingerprints across these classes. Furthermore, most classification schemes[1–7] are based on the configuration of singular points (i.e., core and delta points) which may not be present in dab prints acquired using small-sized sensors. In this work, we focus on indexing techniques for fingerprint filtering. The proposed approach extends the indexing framework based on minutiae triplets proposed by Bhanu et al.,[8] Germain et al.,[9] and Bebis et al.[10] The primary contributions of this work are: (a) the inclusion of ridge curves associated with minutiae triplets in devising the indexing mechanism; and (b) demonstrating the efficacy of the indexing process across multiple sensors with comparable resolution.

2. CLASSIFICATION VS INDEXING

Fingerprint classification involves assigning a (usually unique) class label to a fingerprint, generally based on global features (such as ridge structure and singularities), in an efficient and repeatable way. By employing a classification strategy, the fingerprint database may be partitioned into multiple disjoint classes. Several different approaches have been suggested in the literature to classify fingerprints as whorls, loops, arches, etc.† Maltoni et al.[11] have categorized these approaches into five broad categories: syntactic-based,[1,2] structural-based,[3] statistical-based,[4] rule-based and neural network-based[5] methods.

*The index can be a vector entity.

†In some instances, a single fingerprint can be assigned two or more class labels.



(a) (b) (c) (d) (e)

Figure 1. Fingerprint classification based on singularity points (a) Left Loop. (b) Right Loop. (c) Arch. (d) Tented Arch. (e) Whorl.

Traditional fingerprint classification is impacted by the presence of a small number of classes (e.g., 5-8) and the highly uneven distribution of fingerprints within these classes. For instance, the left loop, right loop and whorl classes encompass more than 90% of the fingerprints. This does not pose much of a problem in ten-print identification systems, since a distinct code corresponding to the ten different fingers can be generated based on the class labels. This code can then be used to reduce the number of candidate prints to be examined in the database . However, when the goal is to search for a single fingerprint in a large database (containing, say, millions of identities), the classification scheme may not be successful in significantly reducing the search space. Therefore, alternate methods such as *sub-classification* and *continuous classification* schemes have to be explored in order to reduce the search space.

1. **Sub-Classification**: This approach is adopted by fingerprint experts to perform manual fingerprint matching in forensic applications. However the rules for such sub-classifications are quite complicated and highly dependent on the fingerprint under consideration. Therefore, it is not practical to be used for automatic fingerprint classification.

2. **Continuous Classification**: This approach does not partition fingerprint into disjoint classes but, rather, represents them as feature vectors, such that, similar fingerprints are mapped close to each other in the multidimensional space. Thus, the query (input) fingerprint is matched against those fingerprints in the database whose feature vectors lie in its vicinity.

This notion of continuous classification permits one to devise indexing measures that characterize a fingerprint as a feature point in metric space. This allows for the effective clustering of fingerprints. So the primary difference between classification and indexing is the use of such 'continuous' valued vectors (as opposed to fixed class labels) in the latter. The focus of this work is on indexing.

3. SEARCH AND RETRIEVAL STRATEGIES

A retrieval strategy is necessary to scan through the database after assigning a class label or index vector to the query fingerprint. For any classification or indexing technique, the following search/retrieval strategies may be used.[11]

1. **Search target class only**: The target class is defined to be the class label assigned to the input fingerprint. Only fingerprints belonging to the target class are considered. The search is terminated when a matching fingerprint is found or the entire target class is visited.

2. **Search according to predefined search order**: The order in which the classes are visited is predefined. Therefore, if a matching fingerprint is not found in the target class, fingerprint images from the other classes are examined. There may be instances when the entire database is searched using such a strategy.

3. **Search according to variable search order**: If a matching fingerprint is not found in the target class, then other classes are searched according to the class-likelihood generated for the input fingerprint by the classifier. It should be noted that the class-likelihood may be different for different fingerprints resulting in a variable search order across classes.

4. **Fixed radius search**: This search strategy is relevant to the continuous classification scheme for indexing. The search space is limited to only those fingerprints whose corresponding feature vectors are within a hypersphere of a pre-defined radius centered at the feature vector corresponding to the input fingerprint.

5. **Incremental search**: The search space is expanded in small increments if a matching fingerprint is not located within the radius specified initially.

The goal of the indexing technique is to significantly reduce the number of candidate fingerprints to be considered by the matching algorithm. While a variety of fingerprint features may be considered for indexing[7, 12–14] the use of minutiae points is especially beneficial since most matching algorithms use minutiae features. In the current literature, there are two prominent approaches for fingerprint indexing based on minutiae points: one proposed by Germain et al.[9] and another by Bhanu et al.[8] Germain et al.[9] use minutiae *triplet* features for indexing using the FLASH technique (Fast Look up Algorithm for String Homology).[15] Bhanu et al.'s[8] technique also uses minutiae triplets, but the features extracted from these triplets are significantly different. Moreover, they use Geometric Hashing[16] in order to ensure invariant features. Bebis et al.[10] suggest the use of delaunay triangulation for selecting minutiae triplets. The indexing techniques proposed by Germain et al.,[9] Bhanu et al.[8] and Bebis[10] have been summarized in Table1.

Table 1. Examples of three feature indexing schemes based on minutiae points.

Author	Features used	Performance
Germain et al.	Used triangulation of minutiae points. Length of each side. Local Orientations. Ridge count between two vertices.	The False Positive Rate (FPR) on a 10 person database is 9.5%. FPR on a 100 person database is 63%. With 32 disks distributed over an 8 node IBM SP2 system, could search a database of 10 million prints in 70 seconds. Disk parallelism can be used to reduce query time. Proprietary database used.
Bhanu et al.	Used triangulation of minutiae points. Maximum length of 3 sides Median and minimum angles Triangle handedness, type and direction. Ridge Count Minutiae Density	On NIST-4 database the Correct Indexing Performance (CIP) is 85.5% with verification limited to 10% of the database.
Bebis et al.	Used Delaunay triangulation of minutiae points. Ratio of minimum to maximum length Ratio of median side to maximum side. Cosine of the angle between two smallest sides.	In case of 3 imprints per person in the training set, average correct matching rate is 86.56% and the average false negative matching rate is 13.36%. Proprietary database was used.

4. PROPOSED TECHNIQUE

The technique proposed in this work relies on the creation of a 9-dimensional index space model based on minutiae triplets *and* the associated ridge curves. The K-means clustering scheme is invoked to partition this index space into multiple clusters. Now, each fingerprint is viewed as a *collection* of points distributed in the index space with each point characterizing a 9-dimensional feature describing a triplet and the associated ridge

curves. Each of the points is assigned to one of the pre-defined clusters based on the minimum distance rule. Thus, a cluster in index space will have a listing of all fingerprint identities that have at least one point assigned to that cluster. When a query print is presented to the system, it is first decomposed into triplets and ridge curves, and the 9-dimensional collection of points is generated. Next, these points are mapped to individual clusters in the index space. A set of possible matching identities corresponding to a small number of clusters is then determined. Thus, the query print is compared against this reduced set of fingerprints in order to retrieve the best match. We first describe the features that constitute the 9-dimensional index space.

4.1. Feature Extraction

The features are extracted by examining the structural information in the distribution of minutiae points by employing Delaunay triangulation. This allows for choosing more meaningful minutiae groups during indexing, so that structural information pertaining to a local neighborhood is preserved (Figure 2). This process is also computationally efficient[17–20] since it eliminates the need to consider all possible minutiae triplets in a fingerprint image.

Figure 2. Results of Delauanay triangulation on multiple impressions of two fingerprints (a) and (b).

Given a fingerprint image with minutiae configuration $\mathcal{M} = \{m_1, m_2, \ldots, m_o\}$, $m_i = (x_i, y_i, \theta_i)$, where (x_i, y_i) is the location of minutia m_i and θ_i is its orientation, the process of Delaunay triangulation generates triplets of the form $t = (m_i, m_j, m_k)$, $1 \leq i, j, k \leq o$. The maximum number of triplets generated as a result of this triangulation will be $9o$+1. Two sets of features are extracted from each triplet. The first set of features corresponds to the geometry of the triangle generated by the triplet, while the second set pertains to the shape of the ridges associated with the three minutiae points constituting the triplet.

Let l_1,l_2 and l_3 represent the length of the three sides of the triangle defining the triplet, such that $l_1 \leq l_2 \leq l_3$. Let α_{max} denote the maximum interior angle of the triangle. Then, the following three features are extracted based on the geometry of the triangle:

$$\alpha = \cos(\alpha_{max}), \quad (1)$$

$$\beta = \frac{p^2}{a}, \quad (2)$$

$$\gamma = \frac{l_3}{l_1}. \quad (3)$$

where,

$$\begin{aligned} p &= l_1 + l_2 + l_3, \\ a &= \sqrt{s(s-l_1)(s-l_2)(s-l_3)}, \\ s &= \frac{p}{2}. \end{aligned}$$

This triangle is retained if its quality factor, $\rho = \frac{4*\sqrt{3}*a}{\Sigma_{i=1}^{3} l_i^2}$ is below a certain threshold. The quality factor ensures that skinny triangles are eliminated. It should be noted that the orientation information is not used for feature extraction. This can potentially improve the retrieval time.

The second set of features is based on fitting a quadratic curve to the ridges associated with each triplet. For every minutiae point detected in the fingerprint, a ridge tracing algorithm is invoked that gives a set of points lying on a ridge containing the minutiae point. Since ridge tracing commences from a minutiae point, it is possible for the algorithm to proceed in more than one direction (e.g., in the case of bifurcation points). In such cases, the ridge containing the maximum number of points is selected. Each ridge curve is represented as a second order curve parameterized by the coefficients p_0, p_1 and p_2, i.e., a point (x, y) on the parameterized curve satisfies $y = p_2x^2 + p_1x + p_0$. See Figure 3 The ratio of these coefficients, viz., $\kappa = \frac{p_2}{p_1}$ and $\lambda = \frac{p_1}{p_0}$, are used as features. Since there are three ridges associated with each triplet a set of six features are obtained: $\kappa_1, \kappa_2, \kappa_3, \lambda_1, \lambda_2, \lambda_3$.

4.2. Creating and populating the index space model

Each fingerprint image can, therefore, be represented as a set of Delaunay triangles $\mathcal{T} = \{\vec{t_1}, \vec{t_2}, \ldots \vec{t_r}\}$ that is generated from its minutiae distribution. Each triplet, $\vec{t_i}$, is further characterized by a nonuple consisting of an agglomeration of geometric and ridge curve features, i.e., $\vec{t_i} = \{\alpha^i, \beta^i, \gamma^i, \kappa_1^i, \kappa_2^i, \kappa_3^i, \lambda_1^i, \lambda_2^i, \lambda_3^i\}$. This 9-dimensional entity can be viewed as a single point in hyperspace; thus, each fingerprint image will have a *collection* of points (pertaining to all Delaunay triplets) residing in this 9-dimensional space. Given a set of training fingerprint images, an index space model is first created by performing unsupervised clustering (K-means clustering) on the set of all 9-dimensional entities generated from these images (Figure 4). This results in K clusters, $c_1, c_2, \ldots c_K$ with cluster, c_j, represented by its centroid, $\vec{\mu_j}$.

When a fingerprint corresponding to an identity, y, is input to the system, it is first decomposed into its constituent triplets, $\vec{t_1}, \vec{t_2}, \ldots \vec{t_r}$, which are then mapped into the 9-dimensional index space. Each $\vec{t_i}$, $i = 1, 2, \ldots, r$, will be assigned to exactly one cluster, c_j, $j = 1, 2, \ldots, K$ according to the minimum distance rule, i.e., assign $t_i \rightarrow c_j$ and $y \rightarrow c_j$ if

$$j = \arg\min_{k=1}^{K} ||\vec{t_i} - \vec{\mu_k}||, \quad (4)$$

where $||.||$ is the L2 norm. This process is repeated for every print in the database. Thus, each cluster, c_j, will have a listing of all fingerprint identities, $\{y_{j,1}, y_{j,2}, \ldots y_{j,n_j}\}$, which have at least one triplet assigned to that cluster.

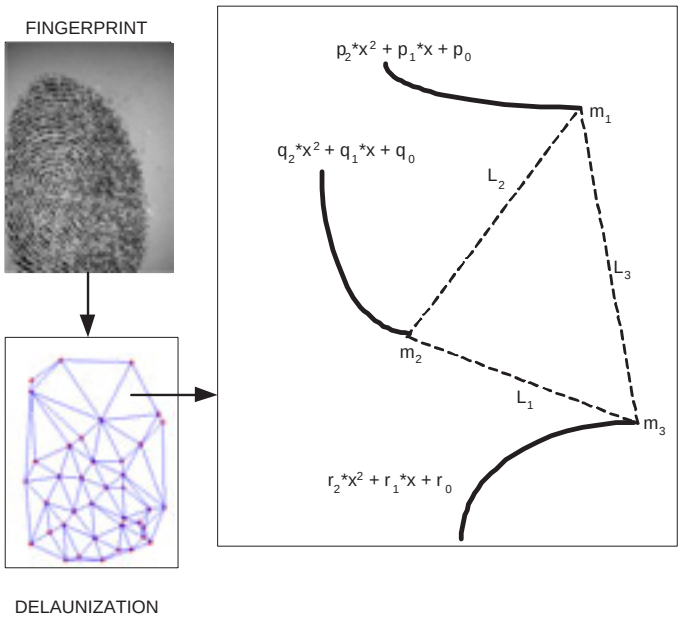


Figure 3. The set of geometric and ridge curve features that are extracted from a triplet arrangement of minutiae points

4.3. Fingerprint retrieval

Single target cluster When a query print, q, is presented to the system, it is first decomposed into its constituent triplets and ridge curves. The set of 9-dimensional points, $\vec{t_1}, \vec{t_2}, \ldots \vec{t_r}$, corresponding to the extracted features are then generated. Next, each point, t_i, is mapped onto a cluster, c_{π_i} ($\pi_i \in \{1, 2, \ldots, K\}$) in index space using the minimum distance rule (equation 4). This process identifies r (possibly) non-unique target clusters, $c_{\pi_1}, c_{\pi_2}, \ldots c_{\pi_r}$, associated with the query print. Those identities occurring frequently in the target clusters (i.e, the top-N identities) are retrieved for further matching (Figure 5).

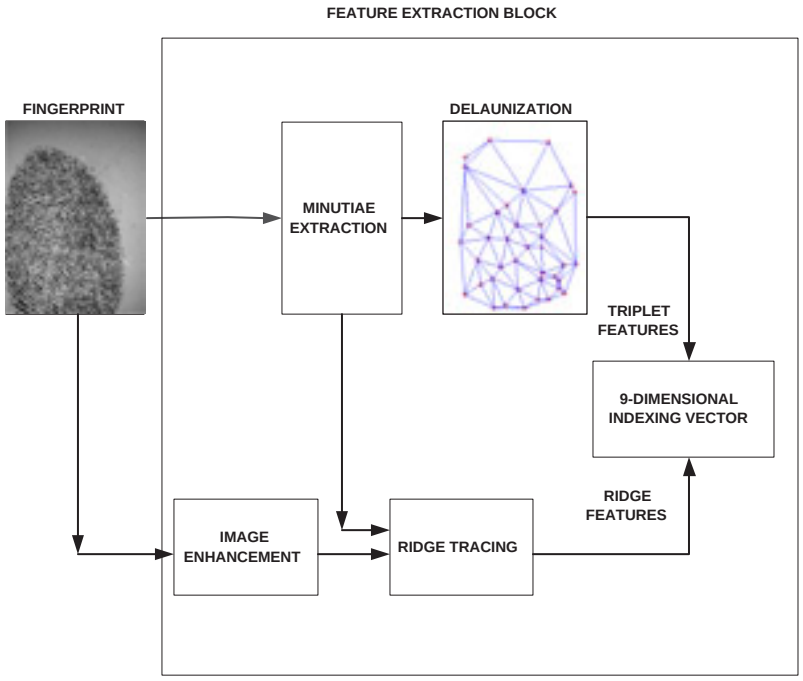
Multiple target clusters The algorithm for fingerprint retrieval can be slightly modified by associating more than one cluster (the top m nearest centroids) with every $\vec{t_i}$ of the query print, i.e., assign $\vec{t_i} \rightarrow \{c_{\pi_i,1}, c_{\pi_i,2}, \ldots c_{\pi_i,m}\}$ such that $||\vec{t_i} - \vec{\mu_{\pi_i,1}}|| \leq ||\vec{t_i} - \vec{\mu_{\pi_i,2}}|| \leq \ldots \leq ||\vec{t_i} - \vec{\mu_{\pi_i,m}}||$ and $||\vec{t_i} - \vec{\mu_{\pi_i,k}}|| \geq ||\vec{t_i} - \vec{\mu_{\pi_i,m}}|| \; \forall k \notin \{1, 2, \ldots m\}$. The retrieval procedure is not affected by this modification. However, using multiple target clusters results in increased computational complexity because it involves scanning the index space model in order to find the m^{th} nearest centroid.

5. EXPERIMENTAL EVALUATION

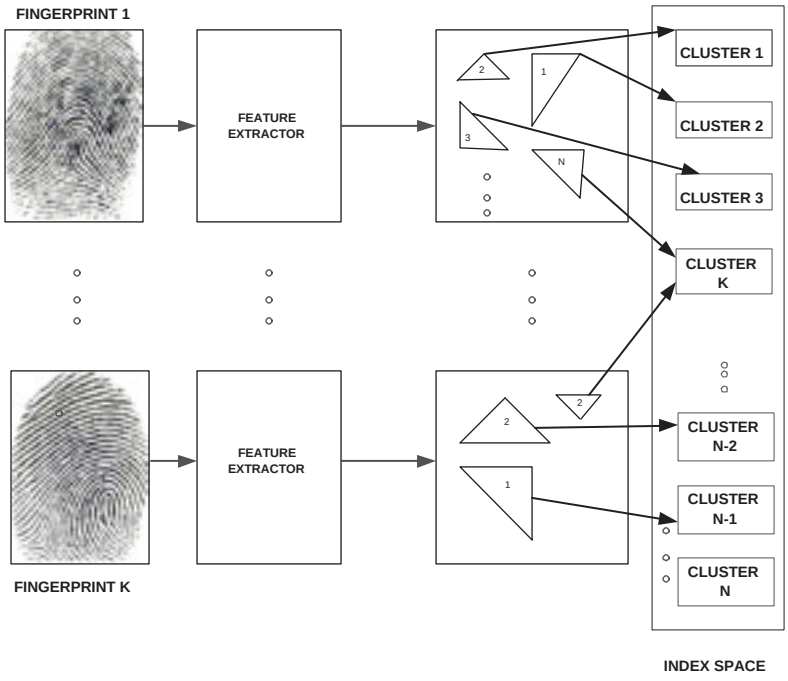
The performance of the proposed indexing mechanism is summarized using two measures namely the *penetration rate* and the *hit rate*. The penetration rate defines the fraction of user identities retrieved from the database upon presentation of the query print. The hit rate is defined as the probability that the correct user identity is retrieved. In the context of our experiments the following procedure was adopted to compute these measures. Suppose that a database has n fingerprints and there are s query fingerprints. For query print, q_i, we define p_i to be the minimum number of fingerprints that have to be retrieved from the database (based on the retrieval technique described in the previous section) in order to guarantee a hit. Further, without loss of generality let us assume that $p_1 \geq p_2 \geq p_3 \ldots \geq p_s$. Thus, the value $\frac{\Sigma_{i=1}^{z} p_i}{n}$ will be the penetration rate corresponding to a hit rate of $\frac{z}{s}$ since the entries, p_i, are sorted.

Experiments were conducted using the Fingerprint Verification Competition 2004 (FVC2004) database that is partitioned into four (DB1, DB2, DB3, DB4).[‡] Each partition has fingerprint images acquired using a particular

[‡]http://bias.csr.unibo.it/fvc2004/



(a)



(b)

Figure 4. Creating and populating the index space model. (a) Extracting features for creating the index space model. The extracted features are a collection of 9-dimensional entities. (b) Mapping fingerprints in a database to the proposed index space by using the 9-dimensional points extracted from each image. Clustering is performed in the index space by using the K-means algorithm.

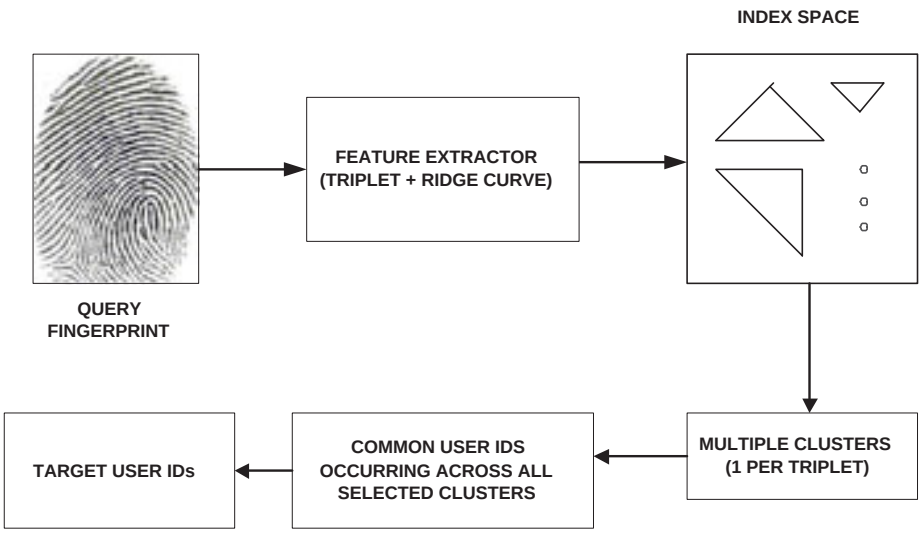


Figure 5. Retrieving the top few identities corresponding to the given query fingerprint.

sensor (see Table 7). There are 880 images in each partition corresponding to 110 fingers (with 8 images per finger). There is no correspondence indicated between fingers across these four databases.

1. The index space was first generated using the minutiae points and ridge curves extracted from the images in the synthetic database (DB4). *Note that* true *fingerprint images are not required to create the index space model*. This is, perhaps, an interesting characteristic of the proposed approach. A total of 600 clusters (i.e., K = 600) were created in the index space. This number may be arbitrarily increased for larger databases.

2. The FVC2004 DB1 database was partitioned into two sets, DB1-S1 and DB1-S2. The first three samples of each of the 110 fingers were used to create DB1-S1; the remaining five samples of each of the 110 fingers were used to create DB1-S2. All images in DB1-S1 were then projected onto the clusters in the index space. The images in DB1-S2 were used as query prints to test the efficacy of the indexing model. The performance of the indexing model can be seen in Table 2.

3. The above experiment was repeated using the FVC2004 DB2 and DB3 databases also. The resulting performance can be seen in Table 2.

4. In order to demonstrate the significance of incorporating ridge curve features $(\kappa_1, \kappa_2, \kappa_3, \lambda_1, \lambda_2, \lambda_3)$ in addition to the geometric features of the triplet (i.e., α, β, γ), the performance with and without using the ridge curve features is presented in Table 3.

5. The importance of assigning a triplet to multiple target clusters is borne out in Table 4 where the indexing performance is observed to improve upon the consideration of multiple clusters.

In order to test the performance of the indexing space model generated using FVC2004 DB4 on other databases, experiments were conducted using the FVC2002 database that is partitioned into four (DB1, DB2, DB3, DB4) sets.[§] Each partition has fingerprint images acquired using a particular sensor (see Table 7). There are 880 images in each partition corresponding to 110 fingers (with 8 images per finger).

1. The FVC2002 DB1 database was partitioned into two sets, DB1-S1 and DB1-S2. The first three samples of each of the 110 fingers were used to create DB1-S1; the remaining five samples of each of the 110 fingers

[§]http://bias.csr.unibo.it/fvc2002/

were used to create DB1-S2. All images in DB1-S1 were then projected onto the clusters in the index space. The images in DB1-S2 were used as query prints to test the efficacy of the indexing model. The performance of the indexing model can be seen in Table 5.

2. The above experiment was repeated using the FVC2002 DB2, DB3 and DB4 databases also. The resulting performance can be seen in Table 5.

3. The effect of considering multiple clusters is shown in Table 6.

Table 2. Performance of the proposed indexing model on the DB1, DB2 and DB3 partitions of the FVC2004 database. The index space model was created using the DB4 partition in all three cases

Database	**Hit Rate (%)**	**Penetration Rate (%) 1 Target Cluster**
FVC2004DB1	100	51.40
FVC2004DB1	95	48.75
FVC2004DB1	90	45.97
FVC2004DB1	85	43.03
FVC2004DB1	80	40.04
FVC2004DB2	100	52.00
FVC2004DB2	95	49.34
FVC2004DB2	90	46.45
FVC2004DB2	85	43.61
FVC2004DB2	80	40.79
FVC2004DB3	100	52.41
FVC2004DB3	95	49.83
FVC2004DB3	90	46.68
FVC2004DB3	85	44.43
FVC2004DB3	80	41.68

Table 3. Indexing performance improvement due to inclusion of ridge features.

Database	**Hit Rate (%)**	**Penetration Rate (%) Geometric Features**	**Penetration Rate (%) Geometric+Ridge Features**
FVC2004DB1	100	54.0	51.40
FVC2004DB1	95	51.43	48.75
FVC2004DB1	90	48.72	45.97
FVC2004DB1	85	45.99	43.03
FVC2004DB1	80	43.25	40.04

6. SUMMARY AND FUTURE WORK

The purpose of this paper was to highlight the improvement in indexing performance when augmenting minutiae-triplet based features with the associated ridge curve information. Each fingerprint image is characterized by a collection of nine-dimensional feature vectors that is mapped into index space. This index space is partitioned into clusters and can be generated using minutiae points obtained from synthetic images. The proposed method was tested on different fingerprint databases from the FVC2002 and FVC2004 repository. It will be interesting to observe the performance of this scheme across databases having different sensor characteristics (especially,

Table 4. Indexing performance when multiple target clusters are identified for each Delaunay triplet. Using more than two clusters does not seem to have any benefit.

Database	Hit Rate (%)	Penetration Rate (%) 1 Target Cluster	Penetration Rate (%) 2 Target Clusters	Penetration Rate (%) 3 Target Clusters
FVC2004DB2	100	52.00	47.05	46.97
FVC2004DB2	95	49.34	44.22	44.17
FVC2004DB2	90	46.45	41.36	41.33
FVC2004DB2	85	43.61	38.54	38.53
FVC2004DB2	80	40.79	35.78	35.66

Table 5. Performance of the proposed indexing model on the DB1, DB2, DB3 and DB4 partitions of the FVC2002 database. The index space model was created using the FVC2004 DB4 in all four cases.

Database	Hit Rate (%)	Penetration Rate (%)
FVC2002DB1	100	47.32
FVC2002DB1	95	44.22
FVC2002DB1	90	41.11
FVC2002DB1	85	38.11
FVC2002DB1	80	35.10
FVC2002DB2	100	47.07
FVC2002DB2	95	44.28
FVC2002DB2	90	41.54
FVC2002DB2	85	38.83
FVC2002DB2	80	36.12
FVC2002DB3	100	50.27
FVC2002DB3	95	47.54
FVC2002DB3	90	44.69
FVC2002DB3	85	41.81
FVC2002DB3	80	38.93
FVC2002DB4	100	45.39
FVC2002DB4	95	42.46
FVC2002DB4	90	39.61
FVC2002DB4	85	36.79
FVC2002DB4	80	33.93

Table 6. Indexing performance on FVC2004 DB4 and FVC2002 DB4 when varying number of target clusters are considered.

Database	Hit Rate (%)	Penetration Rate (%) 1 Target Cluster	Penetration Rate (%) 2 Target Clusters	Penetration Rate (%) 3 Target Clusters
FVC2002DB4	100	51.46	45.39	45.41
FVC2002DB4	95	48.76	42.46	42.51
FVC2002DB4	90	45.63	39.61	39.62
FVC2002DB4	85	42.57	36.79	36.75
FVC2002DB4	80	39.53	33.93	33.95
FVC2004DB4	100	51.42	48.00	47.86
FVC2004DB4	95	48.72	45.19	45.08
FVC2004DB4	90	45.68	42.29	42.30
FVC2004DB4	85	42.60	39.54	39.54
FVC2004DB4	80	39.52	36.87	36.78

Table 7. Sensor Characteristics.

Database	Sensor Type	Image Size	Resolution
FVC2004DB1	Optical	640x480	500 dpi
FVC2004DB2	Optical	328x364	500 dpi
FVC2004DB3	Thermal Sweep	300x480	512 dpi
FVC2004DB4	Synthetic	288x384	about 500 dpi
FVC2002DB1	Optical	388x374	500 dpi
FVC2002DB2	Optical	296x560	569 dpi
FVC2002DB3	Capacitive	300x300	500 dpi
FVC2002DB4	Synthetic	288x384	about 500 dpi

resolution) but containing the same set of subjects. Future work will include analyzing the improvement in the efficiency of the indexing scheme by using parallel computing. The effect of varying the total number of clusters (i.e., K) on the indexing performance will also be studied. In order to obtain data regarding the computation time and real-time performance, the indexing algorithm should be implemented on a field programmable gate array (FPGA) based array processor.[13]

REFERENCES

1. K. Rao and K. Black, "Type classification of fingerprint: A syntactic approach," *IEEE Transactions on PAMI* , pp. 223–231, 1980.
2. B. Moayer and K. Fu, "A syntactic approach to fingerprint pattern recognition," *Pattern Recognition* **7**, pp. 1–23, 1975.
3. M. Chong et. al., "Geometric framework for fingerprint image classification," *Pattern Recognition* **30**(9), pp. 1475–1488, 1997.
4. A. Senior, "A hidden markov model fingerprint classifier," *Asilomar Conf. Signals, Systems and Computers* **1**, pp. 306–310, November 1997.
5. A. Jain, S. Prabhakar, and L. Hong, "A multi channel approach to fingerprint classification," *IEEE Transactions on PAMI* **21**, pp. 348–359, April 1999.
6. K. Karu and A. Jain, "Fingerprint classification," *Pattern Recognition* **29**(3), pp. 389–404, 1996.
7. T. Liu, G. Zhu, C. Zhang, and P. Hao, "Fingerprint indexing based on singular points," *International Conference on Image Processing* **3**, pp. 293–296, September 2005.

8. B. Bhanu and X. Tan, "Fingerprint indexing based on novel features of minutiae triplet," *IEEE Transactions on PAMI* **25**, pp. 616–622, May 2003.
9. R. Germain, A. Califano, and S. Colville, "Fingerprint matching using transformation parameter clustering," *IEEE Computational Science and Engineering* **4**, pp. 42–49, October 1997.
10. G. Bebis, T. Deaconu, and M. Georgiopoulos, "Fingerprint identification using delaunay triangulation," *IEEE International Conference on Intelligence, Information, and Systems (ICIIS)* , pp. 452–459, 1999.
11. D. Maltoni, D. Maio, and A. Jain, *Handbook of Fingerprint Recognition*, Springer, New York, 2003.
12. J. Boer, A. Bazen, and S. Gerez, "Indexing fingerprint database based on multiple feature," *ProRISC 2001 Workshop on Circuits, Systems and Signal Processing* , November 2001.
13. N. K. Ratha, K. Karu, S. Chen, and A. K. Jain, "A real-time matching system for large fingerprint databases," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **18**(8), pp. 799–813, 1996.
14. X. Liang, T. Asano, and A. Bishnu, "Distorted fingerprint indexing using minutiae detail and delaunay triangle," *3rd International Symposium on Voronoi Diagrams in Science and Engineering (ISVD'06)* , pp. 217–223, 2006.
15. A. Califano and I. Rigoutsos, "Flash: A fast look-up algorithm for string homology," *Proceedings CVPR'93* , pp. 353–359, June 1993.
16. H. Wolfson and I. Rigoutsos, "Geometric hashing: an overview," *IEEE Computational Science and Engineering* **4**, pp. 10–21, October 1997.
17. S. Fortune, "A sweepline algorithm for voronoi diagrams," *Algorithmica* **2**(2), pp. 153–174, 1987.
18. D. Lee and B. Schachter, "Two algorithms for constructing a delaunay triangulation," *International Journal of Computer and Information Sciences* **9**(3), pp. 219–242, 1980.
19. J. Shewchuk, "Triangle: Engineering a 2d quality mesh generator and delaunay triangulator," *Applied Computational Geometry: Towards Geometric Engineering* **1148**, pp. 203–222, 1996.
20. C. Lawson, *Software for C Surface Interpolation, Mathematical Software*, Academic Press, New York, iii ed., 1977.