A Framework for Efficient Fingerprint Identification Using a Minutiae Tree

Praveer Mansukhani, Sergey Tulyakov, and Venu Govindaraju, *Fellow, IEEE*

***Abstract*—Given the existence of large fingerprint databases, including distributed systems, the development of algorithms for performing fast searches in them has become the important topic for biometric researchers. In this paper, we propose a new indexing method for fingerprint templates consisting of a set of minutia points. In contrast to previously presented methods, our algorithm is tree-based and well addresses the efficiency needs of complex (possibly distributed) systems. One large index tree is constructed and the enrolled templates are represented by the leaves of the tree. The branches in the index tree correspond to different local configurations of minutia points. Searching the index tree entails extracting local minutia neighborhoods of the test fingerprint and matching them against tree nodes. Therefore, the search time does not depend on the number of enrolled fingerprint templates, but only on the index tree configuration. This framework can be adapted for different tree-building parameters (feature sets, indexing levels, bin boundaries) according to user requirements and different enrollment and searching techniques can be applied to improve accuracy. We conduct a number of the experiments on Fingerprint Verification Competition databases, as well as the databases of synthetically generated fingerprint templates. The experiments confirm the ability of the proposed algorithm to find correct matches in the database and the minimum search time requirements.**

***Index Terms*—Biometric identification systems, fingerprint identification, indexing.**

I. Introduction

BIOMETRIC systems usually operate in one of two modes—verification or identification. In the verification mode, where the user identifies himself while providing his biometric data to the system, we only need to match the acquired template of the person against the stored template of his claimed identity. Such a 1:1 matching process relies on the two fingerprint images (or their template representation) being sufficiently similar to one another to provide a positive result. Different methods for matching two fingerprints are described in the literature [1]. Generally, the matching process consists in finding correspondences in two templates, e.g., pairs of matched minutiae. Both templates should be available during matching process, and the matching can take significant time in order to consider all possible matching correspondences.

Manuscript received January 15, 2009; revised November 10, 2009. First published May 20, 2010; current version published June 03, 2010.

The authors are with the Center for Unified Biometrics and Sensors, State University of New York at Buffalo, Buffalo, NY 14201 USA (e-mail: pdm5@cubs.buffalo.edu; tulyakov@cubs.buffalo.edu; govind@cubs.buffalo.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSYST.2009.2037286

The problem is more complicated in 1:N recognition systems, also known as fingerprint identification schemes. Given a probe fingerprint image (or template) the system must determine the identity of the owner of the print, from a database of stored templates. The stored template best matching the probe provides the identity of the input probe. A naive method of approaching this identification process is to match each user in the dataset with the probe, and the user with the highest matching score is returned as output. Even though the typical fingerprint matching algorithm can perform a matching of two templates in less than a second, the identification in larger datasets, say containing few millions templates, can require prohibitively large time.

Seemingly, there exist two practical approaches for searching in large biometric databases. The first tries to construct the efficient representation of biometrics, e.g., fixed-length feature vector, and perform fast matching, for example, consisting of calculating Euclidean distance. By distributing matching across few servers it is possible to perform identification in a few seconds, even if the database contains millions of records [2]. But it might be difficult to extract fixed length feature vector representations for fingerprint images. The second direction attempts to split biometric templates into few clusters or classes, and to restrict the database search only to templates of the same class. The example of this approach is the Henry fingerprint classification system. Ideally, though, we would like to have a third approach—indexing. Traditional indexing relies on the some tree structure; for example, we might look at the first letter of the word and select the branch with that letter, second letter corresponds to the selection of second branch in the index tree and so on. In this paper, we investigate the possible construction of such an index tree for fingerprint templates.

Fingerprint matching is not an exact process—two images acquired at different times from the same finger will never be exactly the same, even if the images are acquired under identical conditions. Moreover, this is not the case in most practical applications where two matched images could be acquired by different types of sensors, under different conditions and at different times. Translation of the fingerprint with respect to the origin, difference in orientation of the two images, distortions applied due to pressure of the finger on the sensor surface are just some of the ways in which two samples differ. Other factors such as partial acquisition of one or more images, presence of dirt or water on the finger surface or errors (quantization or algorithmic) in the feature extraction algorithm also contribute to the complexity of matching two fingerprint images. As a result, the indexing method should be able to deal with distorted or missing minutia information. Our method does this by: 1) considering localized minutia information and 2) enrolling the

same fingerprint multiple times into the index tree. This results in a significant increase of efficiency in complex (possibly distributed) systems.

The content of this paper is as follows. The existing approaches to perform fingerprint identification in large datasets are discussed in the next section. In Section III, we introduce our approach for fingerprint indexing. Section IV contains the results of the tests of our algorithm for speed and accuracy on different real and synthetic datasets. Finally, we conclude a paper with a summary of our work and present some direction for future research.

II. Current Fingerprint Identification Systems

One of the ways to reduce search space in fingerprint databases is to determine the class, e.g., whorl, arch or loop, of all enrolled fingerprints and to match the probe only to the enrolled fingerprints of the same class (*Henry classification system*). This approach has been in use from the beginning of the fingerprint usage for person identification [3] and automated fingerprint identification systems (AFIS) might include an implementation of such classification [4], [5].

In spite of a reduction in the search space, systems relying solely on classification might still be insufficient for real world applications. Fingerprints can be classified into one of six (at times even four or five, in most applications) classes which is not good enough in order to arrive at a decision while searching in large datasets. The problem is worsened by not having all fingerprint classes of the same size. So classes with a higher frequency will be searched more often, and will have a greater number of candidate prints in them. Moreover, some classification systems return the two most probable classes for a particular fingerprint, which does reduce the misclassification rate, but increases the search space. However, far greater speed-ups will be needed in case of very large datasets. The approaches bypassing Henry classification system have been found to be more effective, both in terms of better accuracies and in terms of the size of the search space considered [6].

A. Filter-Based Schemes

A set of features (or a feature vector) is extracted from the fingerprint image by applying a series of filters. Such an approach has been used in the Local Axial Symmetry (LAS) Registration scheme by Liu [7] where the system tries to identify fingerprints by locating (almost) symmetrical subregions of the image. A scheme of three filters (core, delta and parallel) have been used to transform fingerprint images to a easily indexible template by Li [8]. Another combination-based approach is used by Boer et al. [9] where three possible indexing features: directional field estimates, FingerCode, and triplets have been used, and a system combining them has worked more efficiently.

Singular point features, class information and ridge count information are used at various levels to prune the search space. However each fingerprint in the space must be individually compared to the test template to give the best match. Singular point-based indexing is also used by Liu *et al.* [10]. Another approach that has been used is by Feng [11] where ridge information around minutiae points have been used to index prints.

B. Minutiae-Point Indexing Schemes

Minutiae-point based fingerprint indexing algorithms involve a variation of a binning technique as described below. A bin array is developed to group fingerprint templates having similar minutiae point arrangements together. The features used for binning depend on the type of minutiae point arrangements to be binned. Different combinations of these feature values are used to create a n-dimensional array of bins. In the enrollment stage, information from each extracted minutiae are mapped to an arrangement of bins. The fingerprint template identifier, corresponding to the matched minutiae is stored in the bin. Hence, each bin has a list of all the templates which have an extracted minutia point corresponding to its bin boundaries. When a test fingerprint arrives into the system, minutiae extraction is done, the bins corresponding to each of the minutiae points are determined, and the template details in those bins are ordered according to the frequency of occurrence. These templates are compared, in order, with the test template, until a match is found.

The most common indexing schemes used involve indexing fingerprints based on the local features of minutiae triplets. Different triplet-based schemes, formed by varying the features extracted and assigning different weighting schemes to matching pairs have been explored in [12], [13] and [14]. Whereas the filter-based schemes explicitly construct fixed length fingerprint representation for fast fingerprint matching, triplet-based indexing schemes do it implicitly. Basically, the features correspond to the number of triplets in a particular bin. The matching consists in determining the common numbers of triplets in two fingerprints.

C. Limitations of Current Indexing Schemes

Scalability with Large Datasets: It can be seen that indexing approaches have been able to reduce the size of the search space quite significantly, but it is still a linear fraction of the original, and hence of the order O(N), where N is the size of the dataset. Even a quick scan of the entire dataset (without a detailed matching procedure) to eliminate a bulk of the candidate templates from further consideration makes them unsuitable for many real time applications which could involve very huge datasets. Even the triplet-based binning approaches, which do have a large number of bins, would have a considerable overhead in aggregating the results of the bin constituents when the size of the datasets (and hence bins) do get very large.

Addition of Separate Matching Algorithm: Currently studied indexing systems just produce an ordered list of candidate matches, and would require a separate matching algorithm at the post-processing verification stage, to perform a 1:1 match of the test template with each candidate image left in the reduces search space. This is an additional overhead on the system.

III. Using Minutiae-Based Graphs for Fingerprint Recognition

Previously, Chikerrur *et al.* [16] proposed a fingerprint matching method using local patterns of multiple minutiae points(Fig. 1). In the k-plet approach, graph $G(V, E)$ has been generated based on the arrangement of minutiae points in the enrolled template. The matching process has two parts: local matching for two k-plets, and combination of local matches by

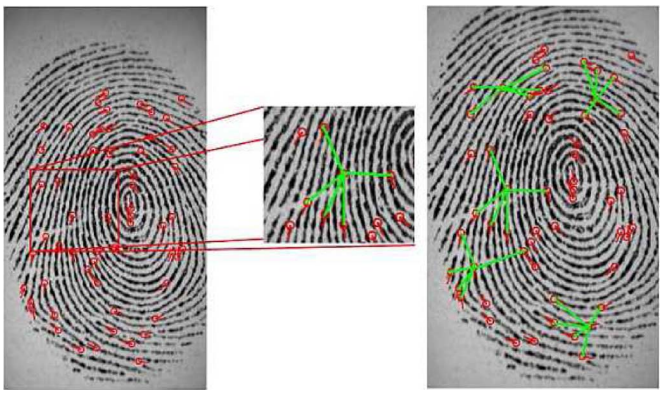


Fig. 1. k-plet technique uses local arrangements of minutiae points to match fingerprints [15].

global *coupled breadth first search*. Specifically, if two k-plets from two fingerprints are matched, we try to expand the the matching to the neighboring k-plets in both fingerprints.

Our indexing algorithm employs similar ideas: we perform local match and, in case of success, we try to expand the match to nearest local neighborhood. But instead of matching our probe fingerprint against a single enrolled fingerprint, we encode the neighborhood configuration information into one global tree and match our probe against the that tree. In order to reduce complexity, in the current system we simply take single minutia in place of k-plets, and expansion of match consists in considering the closest non-matched minutia.

A. Index Overview

Our system arranges the fingerprint dataset in a tree-based structure, as shown in Fig. 2. Each non-leaf node represents an arrangement of minutiae points based on the path from the root to that particular node. Fingerprint templates are enrolled at the leaf nodes, one or more templates can be enrolled at each leaf. Also, a fingerprint might be enrolled at multiple leaf nodes in the tree, representing multiple minutiae paths corresponding to various different minutiae arrangements in a single fingerprint image.

Note, that we build a single tree for all fingerprints in the database, and all enrolled fingerprints will have some of the leaves representing them. The tree provides true index structure for fingerprint database—when we perform identification, we traverse the tree and find leaves corresponding to minutia paths in the test fingerprint template; found leaves will most likely point to the matching enrolled fingerprints in the database.

B. Branch Selection and Binning of Minutiae Points

To arrange the minutiae points together, we use the concept of minutiae bins. Binning is done based on the relative features of a particular minutiae point and its nearest neighbor. The feature set used for binning the points will be selected based on the analysis of fingerprint matching features, discussed later in this document. A discretization technique will be used to handle continuous featured values. Hence, the number of bins for a particular minutiae point will depend on the number and the properties of each individual feature selected. An example of the binning procedure follows.

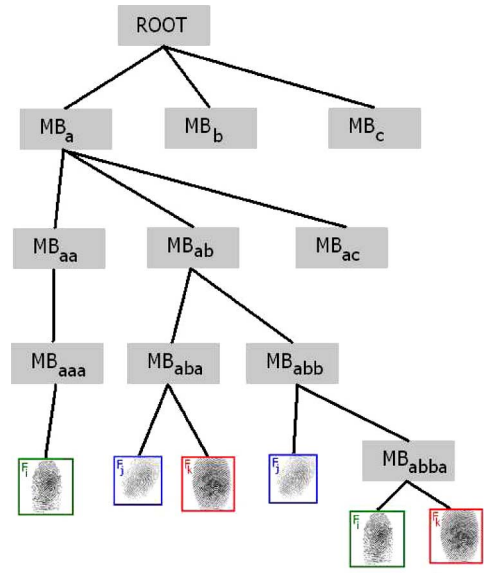
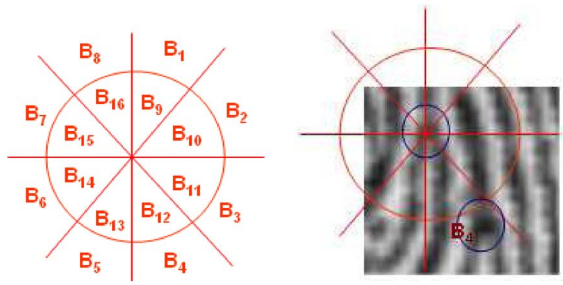


Fig. 2. Different fingerprint templates enrolled in the minutiae tree.



(a) (b)

Fig. 3. (a) Basic bin arrangement for 16 bins. (b) Bins have been applied to a pair of minutiae points.

Consider an arrangement where we use two features for binning a particular point its distance and orientation with respect to the current minutia point. We could set a single threshold value on the distance, effectively dividing all values into two clusters. For the orientation, we divide these values into eight different bins, giving us a total of 16 bins, as can be seen in Fig. 3(a).

Consider an arrangement of minutiae points as shown in Fig. 3(b). Assume that the top point is the current point, corresponding to the current node in the tree. If we evaluate the location of the nearest minutiae point to this one, located at the bottom right of the image, we can see that, using our binning scheme, it would map to bin B4. (In case two or more points are located equidistant from the current point, another metric, such as the orientation could be used to break the tie.) Hence, the path corresponding to B4 is taken (Fig. 4) and the new (bottom right) minutiae point becomes the current point. The process is repeated, until we reach the end of the tree.

C. Enrollment of Templates

One or more fingerprint templates per user might be enrolled into the system. Noise removal, smoothing and binarization of

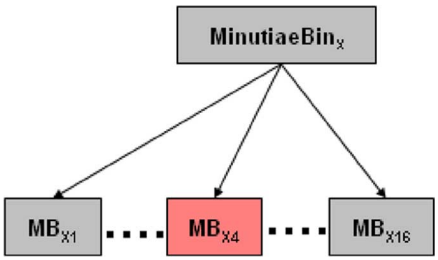


Fig. 4. Selection of branch at the lower level depends on the assignment of the minutiae point to the bin [Fig. 3(b)].

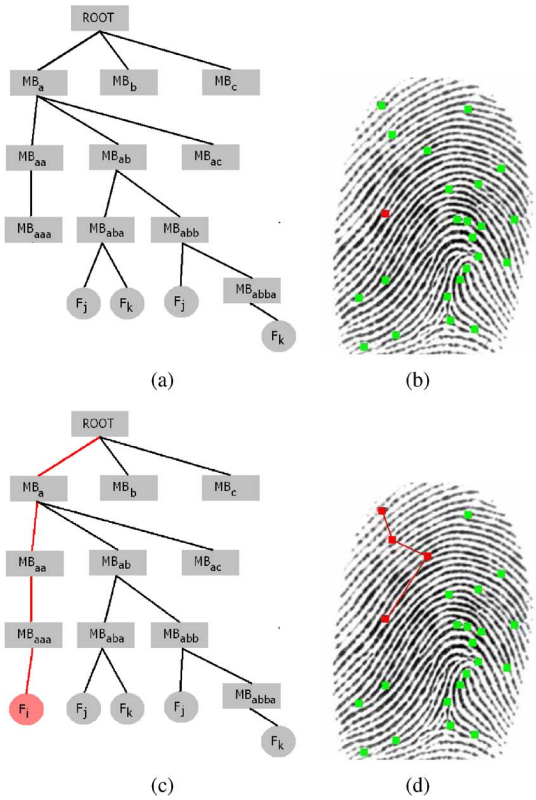


Fig. 5. Enrollment of a fingerprint template. Original minutiae tree (a), in which template (b) must be enrolled. One minutia point (red color) is selected as a root, the path from it to neighboring minutia points is found (d) and the corresponding branches in the tree are selected (c). The fingerprint gets enrolled at the corresponding leaf node of the index tree.

the fingerprint image may be carried out as a part of preprocessing, followed by minutiae extraction. One point is selected as the starting minutia; this corresponds to the root of the tree. Its nearest minutia point is found and its features are calculated relative to the current (root) minutia. Based on these feature values, the matching bin is found and the control moves to the corresponding branch of the tree, with the nearest (with respect to the current) minutia as the new current minutia. Now the nearest minutia point to the (new) current minutia is determined and the procedure is repeated as before.

Once we reach the end of the tree, we will enroll the fingerprint template at the corresponding leaf node (Fig. 5). Now another minutia is selected as the start (root) minutia and the above enrollment procedure is again carried out; here the fingerprint might get enrolled at a different location at the tree, depending on the local neighborhood of the new start (root) minutia point. We repeat this process with all minutia from the fingerprint serving as a root, so that if fingerprint template has n minutia, then up to n paths (and n leafs) of the tree will correspond to the same enrolled fingerprint template (Fig. 6).

D. Fingerprint Template Matching

The matching task involves the identification of the owner of a fingerprint template, assuming that one of the initial process carried out is similar to the enrollment scheme. Preprocessing (noise removal, smoothing, binarization) is done before the minutiae extraction stage.

The matching of the test template is similar to the enrollment procedure. One minutia point is again selected as the root and the relative properties of its nearest minutia point are calculated. These are used for the selection of the branch for traversal down the tree (Fig. 7). Now the newly selected minutia becomes the current minutia and the process is carried on as before, until a match is found (Fig. 8). In case, we encounter a node with no matching fingerprints on the branch to be traversed, we might either backtrack to one of the previous nodes or start over with another minutia point from the test template taken as the root.

The matching will succeed if during the processing of test template we find a minutia path similar to the minutia path of the enrolled template. The corresponding leaf node in the index tree will point to the enrolled template having same path of neighboring minutia. In order to increase our chances of finding corresponding minutia paths, we propose (see the next section) investigating alternative minutia paths for test fingerprint in cases when the neighboring minutia positions are near the boundaries of the bins.

In any case, the search time for the test fingerprint is approximately the same as the enrollment time: exploring n paths (n is the number of minutia in the test template) of m neighboring minutia (m is the maximum depth of the tree) will take $O(nm)$ time. The total size of the tree is equivalent to the number of leaf nodes and equals to $O(B^m)$, where B is the number of branches at each node. Generally, the search time and the tree size do not depend on the number of enrolled fingerprints in the database. But for bigger fingerprint databases we might require bigger numbers of bins B and search path lengths m, so that the chance of collision (different fingerprints having similar minutia paths and thus being enrolled at the same leaf nodes) is minimized.

E. Variation in Minutiae Feature Values

Two fingerprint images from the same finger are never exactly similar. There is inherent variation in the values of the extracted minutiae points caused due to:

i) different positioning of the finger on the scanner, which causes affine transformations on the images;
ii) pressure applied by the user causing distortions in the image;
iii) quality of the images caused do to external factors such as dryness of fingers;
iv) limitations of the feature extraction procedure such as quantization and image quality leading to missing minutiae points and addition of spurious points.

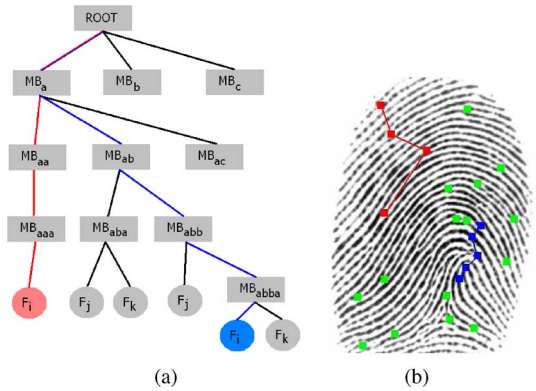


Fig. 6. Enrollment of a fingerprint template (contd.). The process of enrollment is repeated for each minutiae point in the enrolled template chosen as a root (a), giving a different path in the index tree. Consequently, the fingerprint template gets enrolled at the multiple leaf nodes of the tree (b).

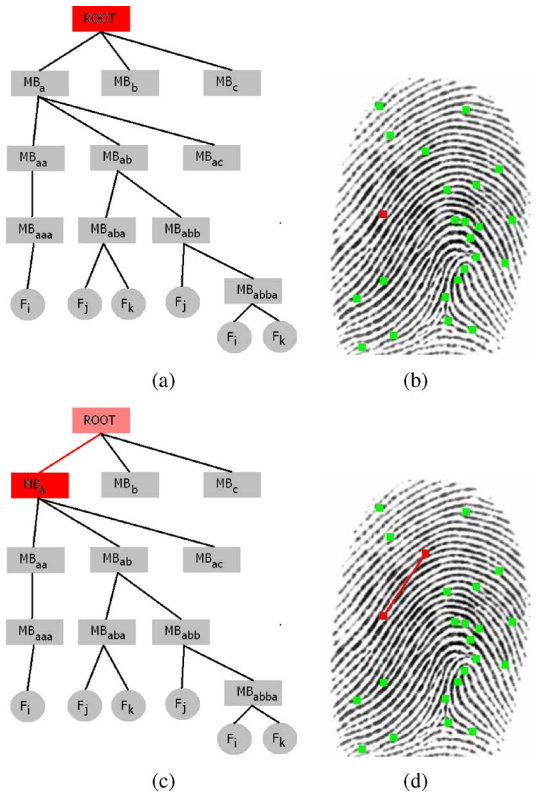


Fig. 7. Matching procedure. One point is selected as the root (a), (b). Based on the features of the nearest neighbor (d), the branch for traversal is selected (c).

These variations lead to variation in the number of correctly extracted minutiae points and their locations in the image. They get reflected in the indexing and matching procedure, and thus reduce the overall accuracy of the system. To account for errors that might be caused due to such variations, we implement the following techniques in our indexing scheme.

1) Handing Missing and Spurious Minutiae Points: Fig. 9 shows two fingerprint samples taken from the same user. Minutiae points have been extracted from each of the fingerprint images. Consider the case, as shown in Fig. 9(b) and (d), where a minutiae point has been detected in one image but the extraction algorithm has missed the point in the other image. If one

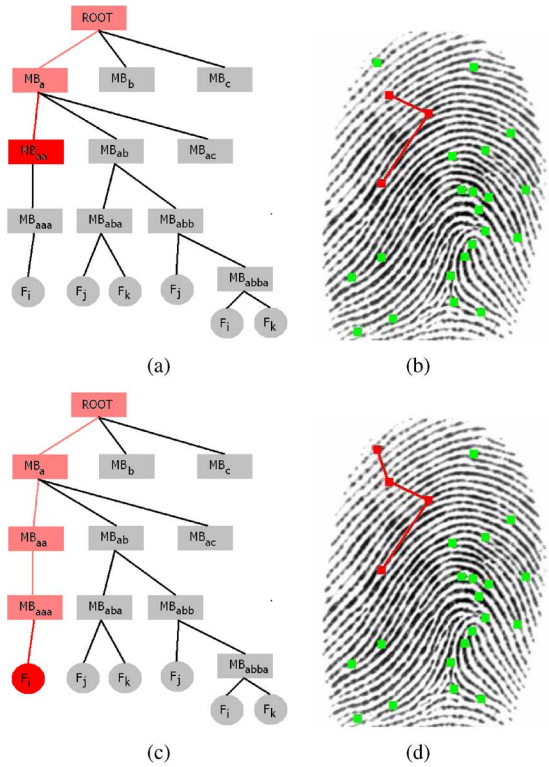


Fig. 8. Matching procedure (contd.) The traversal of the tree continues (a), based on the corresponding minutiae points (b). In the end, the leaf node is found (c), thus ensuring a match based on the local minutiae arrangement (d).

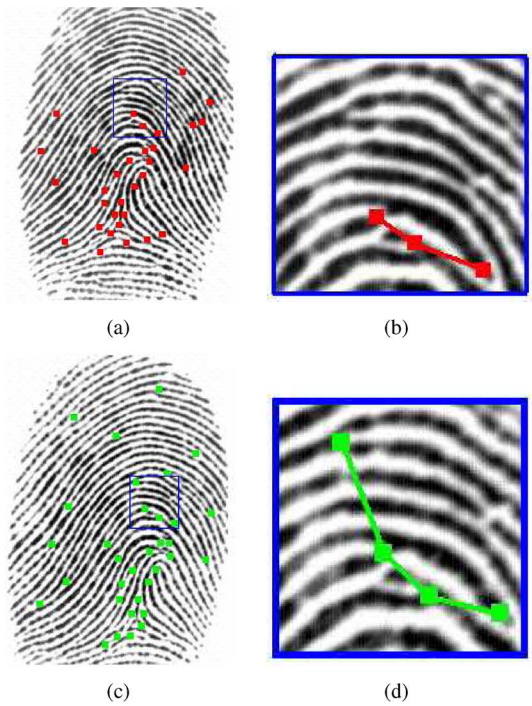


Fig. 9. Example of a missing minutia point. As the top minutia point on the upper image has not been extracted, the tree for these patterns will be traversed differently.

of these images is enrolled into the tree, and the other happens to be the probe image, they will not match each other as they would lead to different nodes in the tree.

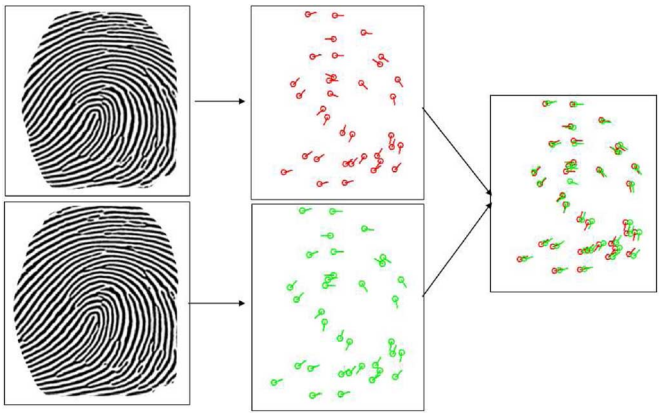


Fig. 10. Variation in minutiae points of fingerprint images.

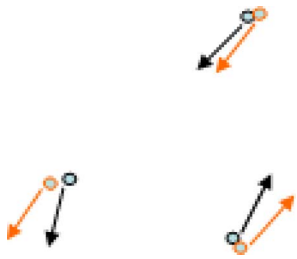


Fig. 11. Minutiae triplets are considered to be similar even though they do not match exactly.

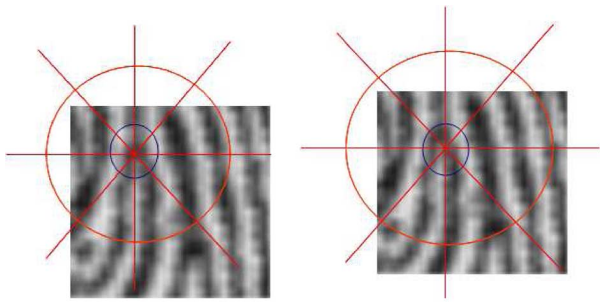


Fig. 12. Minutia point during enrollment of template (a) and during search phase (b) maps to different bins.

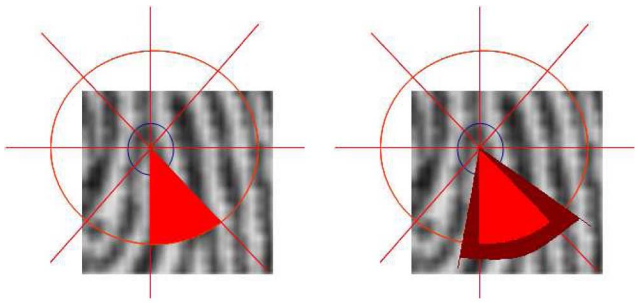


Fig. 13. Search space is limited to the bin area [12(a)]; expanding the search to multiple bins will also find minutiae points that are just outside the bin boundaries [13(b)].

Similarly due to noise in the images, the minutiae extraction algorithm might incorrectly mark certain points as minutiae in some images, leading to spurious points being introduced in the image. These also cause similar errors as described before, and hence the problem of missing and spurious minutiae points are handled similarly.

Although having clean images with minimum noise, better scanning hardware and more robust extraction algorithms are the best way to handle this problem, some amount of redundancy is built into the indexing system in the following ways.

i) Enrolling multiple points from the same fingerprint as root. For every enrolled fingerprint image, each extracted minutiae point is iteratively taken as the root and the fingerprint template is placed into the corresponding slot in the tree. This ensures that even if a small neighborhood of minutiae points is correctly extracted, the search algorithm is able to find the enrolled template for the corresponding probe.

ii) Enrolling multiple images of the same user. In case some minutiae points are missed by the extraction algorithm, their presence in another sample of the same finger will cause a correct enrollment. As a default, most of our experiments are performed using two enrolled templates per user, to mitigate the effect of any errors during the feature extraction of a single template.

2) Variation in Extracted Feature Values: Fig. 10 shows us two fingerprint images and their matching minutiae points. We can observe variability in the minutiae locations and orientations(Fig. 11), even if we account for global affine transformations to the images. Hence, an exact match between fingerprints is not possible, as discussed earlier.

Such variations in minutiae features affect consistency of binning procedures, in cases where the minutiae points to be binned are close to the bin boundaries. For example, in Fig. 12, we see how a slight shift in the position of the same minutiae point, with respect to the bin center, causes the minutiae to be enrolled in one bin and searched in another. As the searching procedure only searches for minutiae points strictly within the bin boundaries [Fig. 13(a)] the enrolled minutiae point is not found.

This problem is solved by searching the neighboring bins for those points that have a strong likelihood of crossing the bin boundaries across multiple samples. Based on the choice of a suitable threshold, those points that are sufficiently close to the bin boundaries, have the next closest bin also searched while traversing the tree. Relaxing the bin boundaries as shown in Fig. 13(a) will lead to a greater number of correct minutiae points being matched and a robust method of traversal down the tree.

IV. Speed and Accuracy Analysis of Indexing System

In this section, we empirically obtain and discuss the performance of the indexing system on both live-scan and synthetic datasets.

A. Experimental Setup

We have built and implemented the indexing system described, and have tested it on the Fingerprint Verification Competition (FVC) 2002 [17] Database 1 and FVC 2004

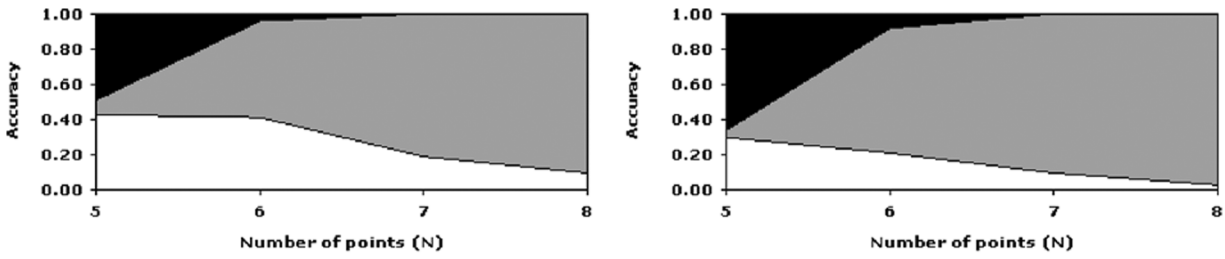


Fig. 14. Results for single path search—number of unmatched prints increases with greater depth in the tree. (a) FVC 2002 DB1, (b) FVC 2004 DB1.

Database 1 [18]images. Each dataset has eight images each of 100 different users, each scanned at 300 dpi (dots per inch). Of these images, three images per user have been enrolled in the tree and the remaining five have been used in the test mode as probes, to determine if the system can correctly identify the owner of the fingerprint.

Each minutiae point is binned into one of 128 bins, based on value of its distance to the previous point ($\text{threshold} = 30$ pixels, two classes), its angle to the vertical (eight classes, 45 degrees each) and the difference in the orientation of the ridge directions of the current and the preceding point in the tree (eight classes, 45 degrees each).

In the current implementation, we could have three outcomes of the database search: found correct fingerprint ("white" color in performance graphs below), found incorrect fingerprint ("black"), and did not find matching fingerprint ("gray").

B. System Accuracy

We have tested the system going various levels deep into the tree. An arrangement of N minutiae points considered enrolls the fingerprint (N-2) levels deep, as the first two points are the root and the alignment minutiae point respectively. In addition, we have built the system to work in two modes.

i) *Single Path Mode*: Here, we traverse the tree on a single path, and hence the minutiae point graph must exactly match the enrolled graph.

ii) *Multiple Path Mode*: As described in Section III-E.2 if the minutiae point is close to the bin boundaries, we search the corresponding minutiae bin and its neighboring bin. Thresholds have been set to 0.1 times the angle and orientation bin sizes and 0.2 times the distance threshold.

Fig. 14 shows us the performance of the system while we have considered a single path, rather than traversing the tree along multiple paths while searching for the identity of the test fingerprint. We can observe that as we increase the number of levels, there are a greater number of unmatched/unidentified fingerprints. This is because, to find a matched template at a deeper level in the index tree implies that a larger number of minutiae points in the enrolled and the probed template must match one another. We can also observe that while searching at lower levels in the tree, a larger number of matching candidates per fingerprint have been returned.

While we traverse multiple paths during the search for a matching enrolled template, we allow for the fact that, due to some minutiae points being located near the bin boundaries, distortions of the fingerprint might cause them to be mapped to a nearby bin. This leads to a greater number of correct matches as compared to the previous case, but we do have a larger number of incorrect matches also. As before, we see (Fig. 15) that searching deeper into the index tree, i.e., as the value of N increases, the number of test fingerprints that do not match any of the enrolled fingerprints increases.

TABLE I
VARYING NUMBER OF TEMPLATES SEARCHED PER USER (SINGLE PATH SEARCH)

No. User Templates	1	2	3	4	5
Accuracy	0.81	0.91	0.91	0.95	0.96
Matching Rate	0.22	0.52	0.72	0.84	0.87

TABLE II
VARYING NUMBER OF TEMPLATES SEARCHED PER USER (MULTIPLE PATH SEARCH)

No. User Templates	1	2	3	4	5
Accuracy	0.73	0.76	0.81	0.80	0.77
Matching Rate	0.19	0.32	0.50	0.57	0.58

Table I shows how an improvement to the system performance can be achieved by increasing the number of templates searched per user, and aggregating the individual results of each template to arrive at a single candidate for that user. This gives us a higher accuracy (ratio of correct matches to total number of probed users, who have returned candidate matches) and matching rate (ratio of the number of users whose searching has returned a candidate list to the total number of users probed).

In this case, using multiple paths while searching does not improve the performance of the system, as seen in Table II. Though larger number of test samples return a list of potential candidates, leading to a higher matching rate, the accuracy of the system is affected due to a greater possibility of incorrect matches being returned.

C. Running Time of the System

One of the main requirements of an indexing system is that it should be able to produce an output in a short duration of time, ideally a real-time system is what is desired. Running time analysis of our indexing system is carried out with respect to both enrollment time for a fingerprint template into the tree structure and the search time required for the system to determine the identity of the owner of the template.

We have performed all these tests using the same FVC datasets as reported in the previous sections, and all indexing parameters (number of features and the binning thresholds) have been set to their default values. All experiments have been

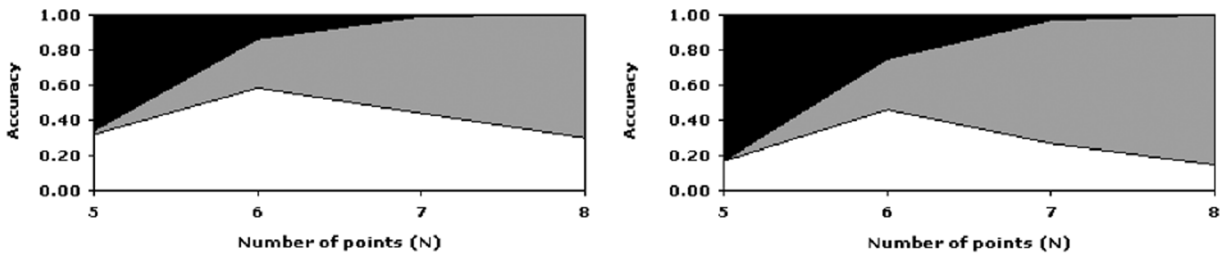


Fig. 15. Results for multiple path search—searching multiple paths leads to greater number of matches returned. (a) FVC 2002 DB1, (b) FVC 2004 DB1.

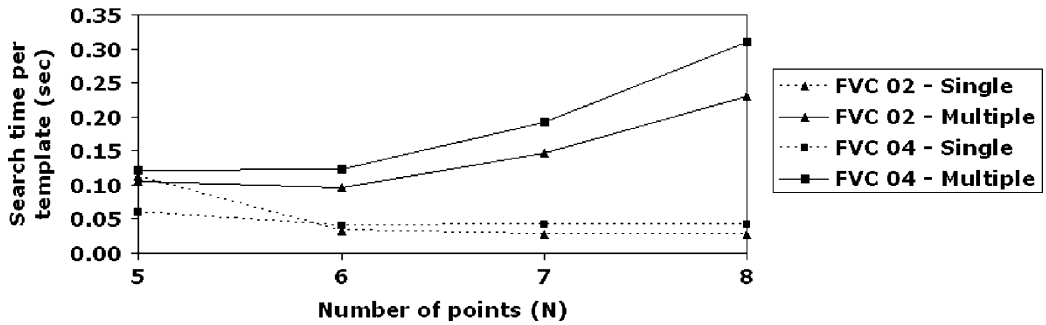


Fig. 16. Running time for a identification of test template.

carried out on a Dell Dimension 2400 PC, with a 2.2-GHz Pentium 4 CPU and 512 MB of RAM, running Windows XP with Service Pack 2.

Enrollment of a Fingerprint Template Into the Tree: The tree structure is designed so that it is easily update-able. If a certain number of users have already been enrolled into the system, and a new user enters the system, it is just the new users details that have to be added to the corresponding nodes in the tree; the details of the other users are unaffected. This means that we do not have to rebuild the tree at every instance, which would have lead to a large time spent if we had a large number of users enrolled.

The enrollment time of a template has been found to depend on two factors:

1) the number of minutiae points in the enrolled template, which can be bounded by a constant value;
2) the number of levels deep into which the fingerprint must be enrolled, which is kept constant by implementing the indexing structure as an NTFS directory tree.

Experiments show us the indexing time has been found to be approximately 1 s per template enrolled per user, independent of the dataset size and number of templates enrolled per user.

Searching the Index for a Test Template: We have empirically found out average search times for the test templates during our previous experiments with the FVC datasets and have plotted them in Fig. 16.

For a single path search mode, we can see that the time remains almost constant as we increase the depth of the search. This result holds up well for results on both the FVC 2002 and FVC 2004 datasets. We do observe that the time taken for a 5-level deep tree on the FVC 2002 database is slightly higher, but we could attribute that to a significantly larger I/O time, due to a larger number of candidates returned.

TABLE III
AVERAGE RUNNING TIME FOR TEMPLATE IDENTIFICATION

	N = 5	N = 6	N = 7	N = 8
FVC 2002 - Single Path	0.112	0.032	0.028	0.028
FVC 2004 - Single Path	0.106	0.096	0.146	0.230
FVC 2002 - Multiple Path	0.060	0.040	0.042	0.042
FVC 2002 - Multiple Path	0.122	0.124	0.192	0.310

For a multiple path search mode, there is a significant increase in the search time as we go deeper into the tree. This is attributed to greater instances where the minutia point is close to the bin boundary, leading to a forking of the search path. Each fork effectively doubles the search time, and hence as we go deeper into the tree, we do traverse a greater overall distance in the search, increasing the time required. The results are tabulated in Table III.

D. System Performance on Large Synthetic Datasets

The difficulty to obtain a large number of willing users to hand over multiple prints and a significant amount of resources needed to manually enroll each user make it impossible to create large size datasets from actual users. Hence, we have tested the indexing system on generated (synthetic) fingerprint minutiae point arrangements. Here, enrolled and probed fingerprint templates have been generated in a controlled environment, so that we can vary, based on our requirements, one or more characteristics of the templates and observe its effect on the accuracy.

Our synthetic template generation method is divided into two steps.

1) Generation of the master template—A master template is generated for each user; it is not used for enrollment or matching but to generate multiple templates of the user from the master template. Fingerprint area (height and

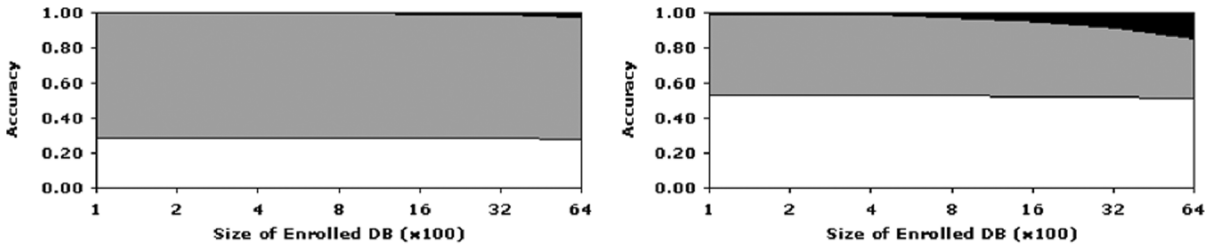


Fig. 17. Number of correct matches remains constant with increase in database size. (a) Single path search. (b) Multiple path search.

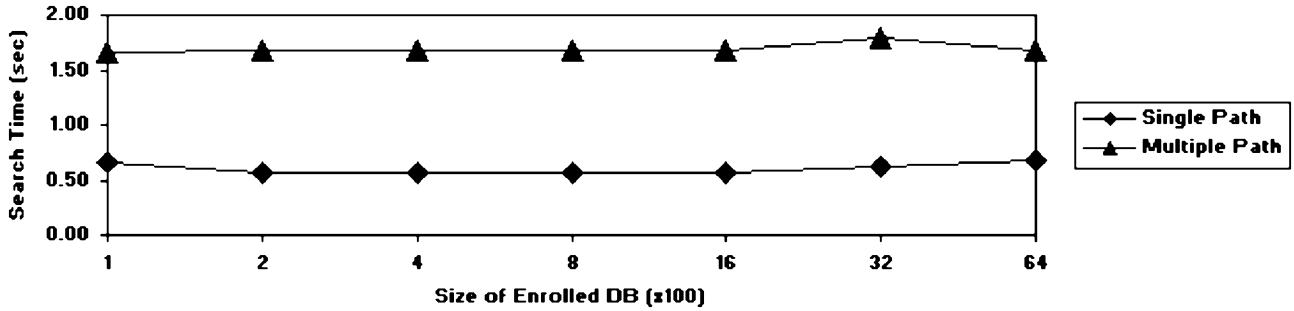


Fig. 18. Running time for an identification of test template.

width), the number of minutiae points and the locations and orientations of the minutiae points are stored in the master template.

2) Creating multiple sample templates—From each master template one or more sample templates are generated per user, by introducing affine transformations on the entire template, location and orientation-based distortions on individual minutiae points and introducing spurious and eliminating genuine (missing) minutiae points.

In this section, we look at how the accuracy and running time scale with an increase in the size of the dataset. We have applied a default value to each distortion to simulate those found in real life datasets. Then, we have varied the magnitude of each distortion individually and then together to study the effect on the system performance.

E. Varying Size of Enrolled Fingerprint Database

One of the central advantages of our indexing scheme is that the system does scale very well with an increase in the size of the enrolled dataset. To validate this, we have progressively increased the size of the enrolled dataset and tested the performance of the system. Three templates per user, with the number of users varying from 100 to 6400, have been enrolled, keeping the default distortion parameters. The tree has been searched six levels deep with a 100 probed users, five templates per user.

We can see from Fig. 17 that as the size of the enrolled dataset increases, the number of correct matches remains constant. For a search into the tree a finite number of levels deep, the template is matched on the basis of the minutiae pattern matching, and even though increasing the number of enrolled templates would cause a greater probability of a wrong pattern matching, it does not affect the probability of a correct match. Observe that the average number of templates returned for a probed fingerprint template does slightly increase with larger datasets. This is more prominent in the multiple path search, as the search branches down a larger number of nodes, and hence has a greater chance of matching any other enrolled template. A higher accuracy, but a larger number of incorrect matches in larger datasets can be observed when we search multiple paths.

F. Running Time on Large Datasets

An important characteristic of the indexing system is that it is scalable with large datasets. We have already seen that our tree building technique allows for additional users to be enrolled without any need to rebuild the tree. A constant time taken to enroll a template into the system means that the time taken to enroll a particular dataset is proportional to the size of the dataset.

For our experiments in evaluating the search time for fingerprint templates, we have used similar hardware configuration as in the previous section and have calculated the average time per probed template, required by the system to return a result for a corpus of 100 users, five templates per user. The number of users in the enrolled dataset has been varied from 100 users onwards with three templates enrolled per user. Testing has been done with a largest enrolled dataset size of 6400 users, which was the maximum permissible given the hardware and I/O constraints permitted by the operating system. It is possible to overcome the encountered constraints by migrating the system from the index stored in file directories to the index stored in a single file. But note that most other indexing systems (such as [13], [10], [19], [7]) have reported their results on datasets of only 100 users.

We can see that even as we successively double the size of the enrolled dataset, the time taken per user remains constant. For multiple path search, we observe that an increase in dataset size does not have a significant impact on the search time. However, as expected, searching multiple paths does take more time than a single path search (Fig. 18).

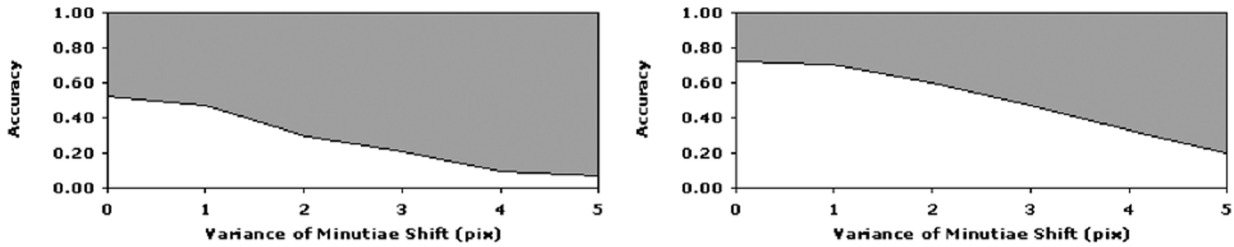


Fig. 19. Effect of minutiae point shifting on indexing system. (a) Single path search. (b) Multiple path search.

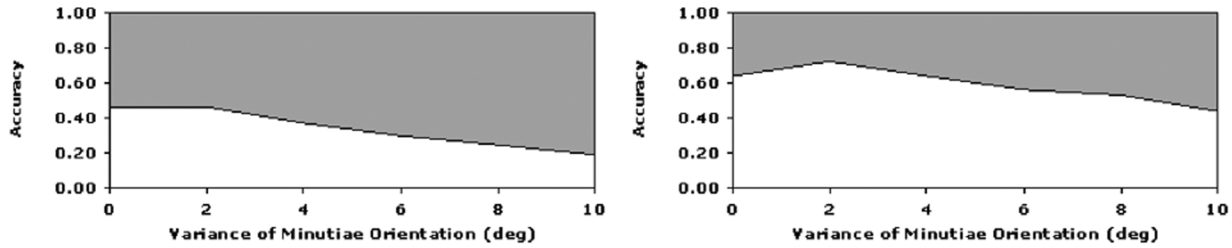


Fig. 20. Effect of minutiae point rotation on indexing system. (a) Single path search. (b) Multiple path search.

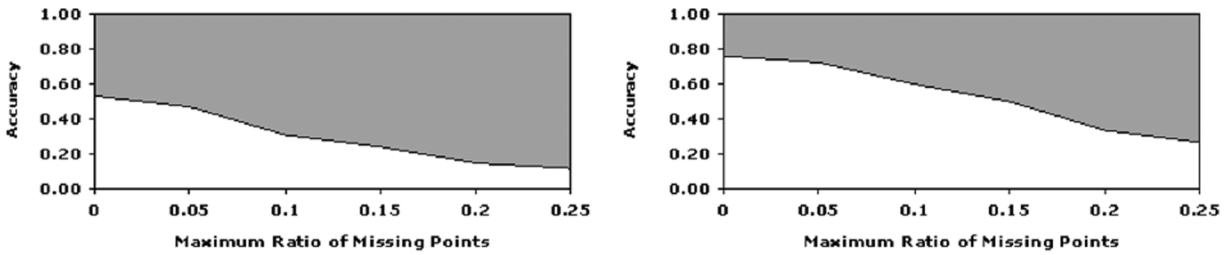


Fig. 21. Effect of missing points in fingerprint template. (a) Single path search. (b) Multiple path search.

1) Shifting of Minutiae Points: We have shifted the location of the minutiae points in the template randomly. The magnitude of the shift is determined by a normal distribution with a 0 mean and a parameterized variance. A random angle, between 0 and 359 is calculated and the location of the minutiae point is shifted based on the distance and angle for that particular point. All other parameters of distortion are kept at their default value.

Fig. 19 shows how the identification system is affected by the shifting in minutiae points. We observe that the accuracy of the system is not seriously affected by increased deviation, up to a certain limit. After this point, there is a drop in accuracy for increase in point distortions. This is because, in a binning structure, a slight deviation in the locations of the points still cause the corresponding points from templates belonging to the same user to get mapped to the same bin. As we increase distortions, the chances of the same minutiae mapping to different bins, across templates increases, and affects the retrieval accuracy.

2) Changes in Minutiae Orientation: Here we vary the orientation of the minutiae points and observe the effect on the system. For each minutiae point, a value is picked from a normal distribution with 0 mean and a selected variance, based on the amount of distortion to by applied to the template. The ridge orientation calculated for the point is either increased or decreased based on the selected value. All other distortion parameters have been set to their default values. We can observe (Fig. 20) that initially there is no significant change in system accuracy. However, beyond a threshold, we see a drop in performance, due to the points getting distorted beyond the bin boundaries. The system has been tested on both a single and multiple path-based searching.

3) Missing Minutiae Points: From a particular template, we calculate the probability of elimination of a minutiae point in it, based on randomly selecting a value up to $\max_{\text{missing}}$. Greater values of $\max_{\text{missing}}$ should lead to a larger number of points being eliminated from it, and might lead to a lesser number of minutiae patterns getting correctly matched, affecting the accuracy of the system. To test this hypothesis, we have varied the value of $\max_{\text{missing}}$, keeping all other distortion parameters at their default value, and have plotted the system accuracy in Fig. 21.

We can see that with an increase in the number of missing minutiae points, the accuracy of the system steadily decreases. The same trend is observed during a multiple path search, although it performs better than in the single path mode.

4) Spurious Minutiae Points: While generating the sample template from the master, we add up to $\max_{\text{spurious}}$ fraction of extra minutiae points into the template, representing spurious

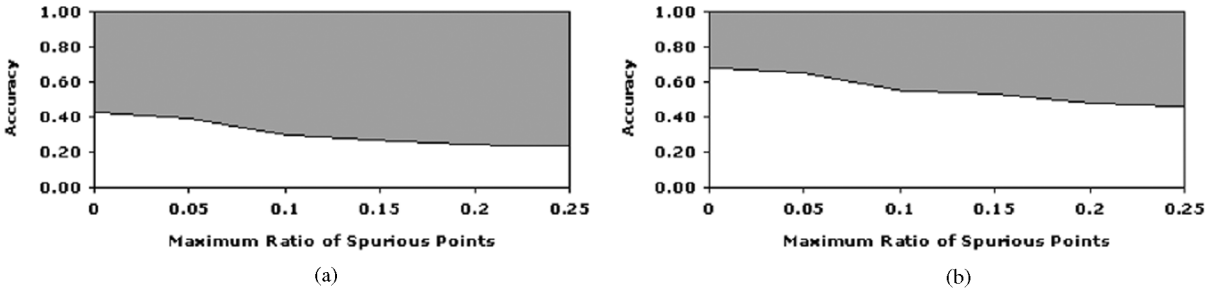


Fig. 22. Effect of spurious points in fingerprint template. (a) Single path search. (b) Multiple path search.

minutiae points typically introduced due to errors in feature extraction. Just as having missing points, we can see from Fig. 22 that a larger number of spurious minutiae points negatively impacts the system, as they cause incorrect tree traversal.

Graphs for single path search mode and multiple path search show us a similar pattern as observed for missing points. A steady decrease in performance can be seen as the number of spurious points increase. We can also see that the system performance is slightly worse as compared to the corresponding numbers for the missing minutiae experiments.

Hence, we can observe that due to a binning approach, slight distortions wont affect the system as corresponding minutiae from the enrolled and probed templates will most probably map to the same bin, in spite of slightly different features calculated. A multiple path-based search will improve the search accuracy, because of it will also search in neighboring bins, and eliminate some of the errors caused due to bin misses. But in both cases, as the distortions in the templates starts increasing, there is a significant decrease in system performance.

V. Conclusion

In this paper, we described the developed indexing system for fingerprint images based on minutiae point patterns, particularly suited to enhance the performance in a complex (possibly distributed) biometric system. Fingerprint templates are binned based on local arrangement of minutiae points. This allows for some amount of distortion in the templates, without loss of accuracy. We empirically observe that methodology is scalable to large datasets. The accuracy of the identification system does not decrease with an increase in the size of the enrolled dataset. The response time for identifying a test template scales well with respect to an increase in search depth of the tree, and is almost unaffected with an increase in the size of the enrolled dataset. The time required for enrollment of a template is not significant, and it remains constant as the indexing tree increases. Experiments have been performed on real world and synthetic datasets to validate these assumptions. Moreover, our indexing system is dynamic which allows us to enroll new fingerprint templates into the tree without any significant changes to the rest of the index.

The system has been designed such that it can be easily customized according to the needs of the user. For example, the number of levels up to which fingerprints are enrolled and searched can be varied—either increasing the probability of getting a correct match or a confidence in the result. This indexing methodology does not depend on any particular feature set to be used as indexing parameters. Based on a feature study of the fingerprint dataset to be enrolled, the best feature set could be identified, and the system performance could be improved.

We consider the developed system as the first prototype of the minutia-based indexing algorithm which can be further developed and improved in multiple ways. For example, we can expand our index tree search to include backtracking based on some matching cost function. This will be a generalization of multipath method which we tested. Or we can take k-plets [16] $(k \geq 2)$ instead of currently considered single minutia used for tree branching. In this case, we have to deal with bigger number of bins at each level, but a smaller number of levels in the index tree.

References

[1] A. K. Jain and D. Maltoni, *Handbook of Fingerprint Recognition*. New York: Springer-Verlag, 2003.
[2] J. Daugman and I. Malhas, "Iris recognition border-crossing system in the uae," *Int. Airport Rev.*, no. 2, 2004.
[3] F. Galton, *FingerPrints*. New York: McMillan, 1892.
[4] G. Candela, P. Grother, C. Watson, R. Wilkinson, and C. Wilson, PCASYS-A Pattern-Level Classification Automation System for Fingerprints National Inst. Standards and Technol., Tech. Rep. NISTIR 5647, Apr. 1995.
[5] N. Ratha, K. Karu, S. Chen, and A. Jain, "A real-time matching system for large fingerprint databases," *IEEE Pattern Anal. Mach. Intell.*, vol. 18, no. 8, pp. 799–813, Aug. 1996.
[6] X. Tan, B. Bhanu, and Y. Lin, "Fingerprint identification: Classification vs. indexing," in *Proc. IEEE Conf. Advanced Video and Signal Based Surveillance*, 2003.
[7] T. Liu, C. Zhang, and P. Hao, "Fingerprint indexing based on las registration," in *Proc. Int. Conf. Image Processing (ICIP)*, Atlanta, GA, 2006.
[8] J. Li, W. Yau, and H. Wang, "Fingerprint indexing based on symmetrical measurement," in *Proc. 18th Int. Conf. Pattern Recognition (ICPR)*, Hong Kong, 2006.
[9] J. de Boer, A. M. Bazen, and S. Gerez, "Indexing fingerprint databases based on multiple features," in *ProRISC 2001 Workshop on Circuits, Systems and Signal Processing*, Utrecht, The Netherlands, 2001.
[10] T. Liu, G. Zhu, C. Zhang, and P. Hao, "Fingerprint indexing based on singular point correlation," in *Proc. Int. Conf. Image Processing (ICIP)*, Genoa, Italy, 2005.
[11] J. Feng and A. Cai, "Fingerprint indexing using ridge invariants," in *Proc. 18th Int. Conf. Pattern Recognition (ICPR)*, Hong Kong, 2006.
[12] R. Germain, A. Califano, and S. Colville, "Fingerprint matching using transformation parameter clustering," *IEEE Comput. Sci. Eng.*, vol. 4, no. 4, pp. 42–49, Oct.-Dec. 1994.
[13] B. Bhanu and X. Tan, "Fingerprint indexing based on novel features of minutiae triplets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 616–622, May 2003.

[14] K. Choi, D. Lee, S. Lee, and J. Kim, "An improved fingerprint indexing algorithm based on the triplet approach," in *Proc. Audio and Video Based Biometric Person Authentication (AVBPA)*, Guildford, U.K., 2003.
[15] S. Chikkerur, "Online Fingerprint Verification System," M.S. thesis, Univ. Buffalo, Buffalo, NY, 2005.
[16] S. Chikkerur, A. Cartwright, and V. Govindaraju, "K-plet and coupled bfs: A graph based fingerprint representation and matching algorithm," in *Proc. Int. Conf. Biometrics*, Hong Kong, 2006.
[17] Second International Fingerprint Verification Competition 2002 [Online]. Available: http://bias.csr.unibo.it/fvc2002/
[18] Third International Fingerprint Verification Competition 2004 [Online]. Available: http://bias.csr.unibo.it/fvc2004/
[19] X. Liang, A. Bishnu, and T. Asano, "A robust fingerprint indexing scheme using minutia neighborhood structure and low-order Delaunay triangles," *IEEE Trans. Inf. Forensics Security*, vol. 2, no. 4, pp. 721–733, Dec. 2007.

Praveer Mansukhani received the Master's degree in 2004 and the Ph.D. degree in 2008 from the Department of Computer Science and Engineering (CSE), University of Buffalo, State University of New York (SUNY).

His research interests are biometric authentication, pattern recognition, and document image processing.

Sergey Tulyakov received the Ph.D. degree in computer science and engineering from the University of Buffalo, State University of New York (SUNY), in 2006.

Currently, he is a Research Scientist at the Center for Unified Biometrics and Sensors, University of Buffalo. His research interests handwriting recognition, fingerprint and face biometric person authentication, and combination of pattern classifiers.

Venu Govindaraju (F'06) received the B.Tech. (Hons.) degree from the Indian Institute of Technology (IIT), Kharagpur, India, in 1986 and the Ph.D. degree from the University of Buffalo (UB), State University of New York (SUNY), in 1992.

He is a UB Distinguished Professor of Computer Science and Engineering at the UB SUNY. He has authored more than 300 scientific papers, including over 60 journal papers and 30 book chapters. He has been the primary investigator(PI)/co-PI of projects funded by the government and the industry for about 50 million dollars in the last 15 years.

Dr. Govindaraju is a Fellow of the International Association of Pattern Recognition (IAPR).