Original Research Paper

Enrolment and Matching of Fingerprints using Minutiae Tree

Praseetha, V.M. and S. Vadivel

Department of Computer Science, BITS Pilani, International Academic City, UAE

Article history
Received: 11-02-2019
Revised: 04-03-2019
Accepted: 14-03-2019

Corresponding Author:
Praseetha, V.M.
Department of Computer Science, BITS Pilani, International Academic City, UAE
Email: praseethasunil@gmail.com

Abstract: Automated fingerprint matching is considered as the most challenging phase of fingerprint recognition since it can be affected by a variety of factors such as noise, skin condition, rotation, distortions and displacements. When there is a large database, the search time to get a matching fingerprint will be relatively high. To reduce the searching time in the database, we have proposed a minutiae tree based indexing method in this paper. The database is represented in the form of a minutiae tree and the fingerprint matching is done by visiting the nodes in the tree where the local configuration of the minutiae is stored. By using this tree structure it is found that the search time or matching time can be considerably reduced and the matching time is independent of the number of fingerprints enrolled in the database. Our framework is scalable and the experiments conducted explores its ability to find correct matches with minimal search time.

Keywords: Minutia, Fingerprint Matching, Minutiae Tree, Binning, Indexing

Introduction

Fingerprint verification and fingerprint identification are the two different modes of operations of a fingerprint based biometric system. Fingerprint verification is considered as a simple process of 1:1 matching whereas fingerprint identification is a complex process of 1:N matching (Jain *et al*., 2010). In both the modes fingerprint matching is an important step. In verification the acquired image is matched with the stored image. The matching algorithm will find the correspondence between the two images and gives a positive result if they are significantly similar. In identification the given probe image is matched with all the images in the dataset to generate a degree of similarity. The algorithm returns the name of the person who has a highest degree of similarity. Fingerprint matching is very challenging since two images acquired at different time exactly under same conditions need not be exactly the same. The difficulty in matching is due to several reasons (Sheng *et al*., 2007). The translation and rotation of the fingerprint images, application of a poor feature extraction algorithm, displaced, false and missing minutiae and the non linear deformations of the images are some of the reasons. To overcome these variations a powerful matching algorithm is needed. So the matching algorithm should be invariant to translation and rotation of the images. It should return the correct result while comparing fingerprints from the same finger even the feature extractor has missed some features or even when the fingerprint images are affected by non linear distortions.

The fingerprint matching algorithm can generally be classified into three different classes:

- Correlation based matching algorithms (Hatano *et al*., 2002; Lindoso *et al*., 2007) which use superimpose two fingerprint images to find the correlation among pixels for different displacement and rotation
- Minutiae based matching algorithms (Jiang and Yau, 2000; Luo *et al*., 2000) which use the extracted minutiae of two fingerprints to find the matching pairs of minutiae
- Non minutiae based matching algorithms (Yang and Park, 2008; Nanni and Lumini, 2009) which use the orientation, shape or frequency of ridges to perform matching between two fingerprints

Among the three classes minutiae based algorithms are the most common and minutiae based matching is considered as a point pattern matching problem.

Our goal is to develop a fingerprint matching algorithm which can reduce the database search time during matching. Towards this goal a minutiae tree based algorithm is proposed in this paper, which enrol the fingerprints at the leaf node of the tree and the matching is done by comparing the values related to each minutia at each node. A minutiae tree based indexing technique with fast retrieval rate is our contribution. The organization of this paper is as follows. After the introduction the related works are given. The following sections include the Proposed Method, Performance Analysis and Conclusion.

Related Works

The uniqueness of fingerprints has made them as the popularly using modality in automatic person identification (Sankaran *et al*., 2014). The features in a fingerprint are called minutiae which contain most reliable and discriminating information. Ridge ending and ridge bifurcation are the two most prominent minutiae used for automatic fingerprint matching. A set of minutiae are detected from the fingerprint during feature extraction and the minutiae set is characterized by the attributes like minutiae type (ridge end or ridge bifurcation), position of the minutiae (x and y coordinates) and the orientation of the minutiae. The minutiae list is considered as a pattern of points and so the fingerprint matching problem can be considered as a point pattern matching problem.

The most popular technique for fingerprint matching is minutiae based matching (Lahby *et al*., 2016). Each minutia is represented by a 3-tuple $M = (x, y, \theta)$ where (x, y) is the location and θ is the orientation of the minutiae. Let T be the template fingerprint and M_T is the minutiae in template fingerprint. Similarly P is the probe fingerprint and M_P is the minutiae in the probe fingerprint. A minutiae based matching algorithm goes through the following steps in general:

- Define a tolerance box for M_T in terms of the permitted maximum spatial distance and direction difference
- M_P is considered matched with M_T if M_P comes within the tolerance box of M_T. This is repeated for all the minutiae in M_P
- A matching score for the two fingerprints is calculated

Inorder to maximize the matching score an optimal displacement and rotation alignment should be obtained. The match or non-match decision is based on the threshold value which varies depending on the application.

The fingerprint database contain hundreds or thousands of images and one method applied to reduce the search space is to classify these images into different classes like left loop, right loop, arch, whorl etc. This approach will reduce the search space since the probe is matched only with the fingerprints which are enrolled in the same class.

Another technique to speed up the identification process is indexing. Fingerprint indexing is applied to reduce the search space and thus to reduce the search time in a large database. Lot of fingerprint indexing techniques have been proposed and applied by researchers to reduce the response time for improving the effectiveness of the identification system (Li *et al*., 2014; Iloanusi *et al*., 2011; Zhou *et al*., 2014; Mngenge *et al*., 2015). Another indexing method known as Minutiae Cylinder Code (MCC) uses a 3D data structure known as cylinder to speed up the fingerprint identification (Cappelli *et al*., 2010). The invariant distances and angles in the neighbourhood of each minutiae are used to develop the cylinder. This approach is a fixed radius approach and so the problem with spurious minutiae are efficiently handled. In this approach, a fixed length bit vector has been used for encoding the neighbour of each minutiae. Locality-Sensitive Hashing is used to index these bit vectors. The similarity between two fingerprints are found by considering the number of collisions of binary vectors. Experimental results have shown that MCC is more accurate than well known minutiae-only local matching techniques.

Multiple features can be used for fingerprint indexing as explained in (de Boer *et al*., 2001). In this approach the authors have made use of the registered directional field estimate, Finger Code and minutiae triplets for indexing. They have found that the database search is more effective by this method when searching in a large fingerprint database. An approach based on minutia neighbourhood structure for indexing is explained in (Liang *et al*., 2007). This is a more stable triangulation algorithm which is insensitive to fingerprint distortion. The minutia details and attributes of low-order Delaunay triangle are used for indexing. Experiments show that the proposed algorithm considerably narrows down the search space in fingerprint databases and is stable for various fingerprints.

A graph based fingerprint matching algorithm has been proposed in (Chikkerur *et al*., 2006). This approach is named as k-plet approach and in this approach a graph has been generated by considering multiple minutiae and their local patterns. For matching two fingerprints, the k-plets are matched by local matching and the local matches are combined by a global coupled breadth first search (Chikkerur *et al*., 2006). The same idea has been used for indexing in (Mansukhani *et al*., 2010). In this approach, the fingerprints are enrolled as a global tree and for matching, the probe is matched against the tree. If the local match is a success, the match will be expanded to the nearest neighbourhood. Here, instead of k-plets a single minutiae is considered to reduce the complexity.

Proposed Method

Our algorithm is the modified version of the algorithm proposed in (Mansukhani *et al*., 2010) which uses a tree structure in which the fingerprint templates are enrolled at the leaf nodes. Mansukhani *et al*. (2010) an arbitrary minutiae is selected as the root of the tree and the features of the nearest minutiae relative to the root are calculated and the particular fingerprint is enrolled at the leaf node. The size of the tree will be very higher since one fingerprint is enrolled in many leaves.

Instead of starting with an arbitrary minutiae as root, the proposed algorithm starts with the core point in the fingerprint. If the fingerprint contains a core point we are making the core point as the root. The minutiae points in the fingerprint are considered in the increasing order of their Euclidian distance from the root (core). Then the required features of the minutiae points are considered and based on these features nodes are created for each minutiae in the fingerprint. For a fingerprint having n minutiae, our tree structure does not contain n enrolments. So the space required for the tree con be reduced considerably.

The proposed method has the following objectives:

- Enrol the fingerprints on the minutiae tree using the local configuration of the minutiae
- Perform the fingerprint matching efficiently by doing a search operation on the minutiae tree

All the steps carried out in the proposed method are explained in detail in the following subsections.

Preprocessing

To improve the quality of the image by improving the clarity between the ridges and valleys, the following steps are applied to each fingerprint image. These steps are the preprocessing steps and will help the feature extraction process. The pre-processing steps applied are:

- Normalization
- Segmentation
- Binarization
- Thinning
- Alignment
- Ridge Orientation

Normalization

The variation in the gray level values of ridges and valleys can be reduced by the process of normalization. Normalization is done on the image so as to get a pre specified mean and variance for the image. The normalized gray level *N*(*I*, *j*) of the pixel at (*i*, *j*) is calculated as:

$$N(i,j)=\begin{cases} M_0+\sqrt{\dfrac{V_0\times\left[I(i,j)-M_I\right]^2}{V_I}}, & if\ I(i,j)<M_I \\ M_0-\sqrt{\dfrac{V_0\times\left[I(i,j)-M_I\right]^2}{V_I}}, & otherwise \end{cases} \quad (1)$$

where, M_0 and V_0 denote the desired mean and variance, M_I and V_I denote the estimated mean and variance and *I*(*i*, *j*) is the gray level of the image *I*.

Segmentation

Segmentation is an important and unavoidable pre processing step in which the foreground and background of the fingerprint image is differentiated. Lot of segmentation algorithms have been proposed by researchers (Mehtre *et al*., 1987; Mehtre and Chatterjee, 1989; Bazen and Gerez, 2001). We have used the algorithm proposed in (Ratha *et al*., 1995) for segmentation. The input image is divided into non overlapping blocks of size $w \times w$. Then the mean *M*(*I*) and standard deviation *std*(*I*) of each block are computed as:

$$M(I)=\frac{1}{w^2}\sum_{i=-\frac{w}{2}}^{\frac{w}{2}}\sum_{j=-\frac{w}{2}}^{\frac{w}{2}} I(i,j) \quad (2)$$

$$std(I)=\sqrt{\frac{1}{w^2}\sum_{i=-\frac{w}{2}}^{\frac{w}{2}}\sum_{j=-\frac{w}{2}}^{\frac{w}{2}}\left(I(i,j)M(I)\right)^2} \quad (3)$$

If the standard deviation of the block is greater than an empirically selected threshold, then the corresponding block is considered as the foreground region otherwise background.

Binarization

Binarization of the grey-scale image: the ridges and valleys of the fingerprint image are highlighted with black and white colors respectively. The intensity of each pixel in the grey-scale image is transformed to 0 or 1 which represent black or white in binary intensity. It can be done by using the equation:

$$I(x,y)=\begin{cases}1 & if\ I(x,y)\geq t\\ 0 & otherwise\end{cases} \quad (4)$$

where, *t* is the global threshold.

Thinning

In this process the foreground pixels are successively eroded until the ridge width is reduced to one pixel size. An iterative algorithm explained in (Guo and Hall, 1989) is used for thinning.

Alignment

Alignment is a very important process to be done before fingerprint matching since the misalignment may lead to false results. Here we have applied a PCA based alignment algorithm (Kour *et al*., 2012). PCA can find the direction in which the data is spread more. The amount of spread is given by eigen values and the direction of spread is given by eigen vectors. The fingerprint alignment is done by using the following steps:

- Find the mean of the coordinates of all points in the thinned image and find the covariance matrix
- Find the eigen value and eigen vectors based on the covariance matrix
- Compute the angle of rotation by finding the eigen vector corresponding to the maximum eigen value
- Since it is assumed that fingerprints will not have rotation more than 60, if the obtained angle of rotation is more than 60, then use another eigen vector to find the angle of rotation
- The fingerprint is rotated according to the angle of rotation

Ridge Orientation

We have used a gradient based algorithm (Hong *et al*., 1998) to find the orientation of ridges on the fingerprint. Using this algorithm, the local ridge orientation at each pixel (i, j) is computed as:

$$\theta(i,j)=\frac{1}{2}\tan^{-1}\left(\frac{\phi'_x(i,j)}{\phi'_y(i,j)}\right) \tag{5}$$

where:

$$\phi'_x(i,j)=\sum_{u=-\frac{l}{2}}^{\frac{l}{2}}\sum_{v=-\frac{l}{2}}^{\frac{l}{2}}w(u,v)\phi_x(i-uw,j-vw) \tag{6}$$

$$\phi'_y(i,j)=\sum_{u=-\frac{l}{2}}^{\frac{l}{2}}\sum_{v=-\frac{l}{2}}^{\frac{l}{2}}w(u,v)\phi_y(i-uw,j-vw) \tag{7}$$

$$\phi_x=\cos(2\theta(i,j)), \tag{8}$$

and:

$$\phi_y=\sin(2\theta(i,j)) \tag{9}$$

ϕ_x and ϕ_y are the x and y components of the image vector field and w is a low pass filter of size $l \times l$.

Minutiae Extraction

After preprocessing the minutiae are extracted from the image. The two types of minutiae that are extracted for the experiment are ridge ending and ridge bifurcation. The location of the minutiae are calculated by analysing each pixel in the thinned image. The crossing number method (Tamura, 1978) is applied to extract the minutiae as shown in Fig. 1. In this method a 3×3 window centred at pixel p is traversed circularly:

$$cn(p)=\frac{1}{2}\sum_{1\ldots8}|P_i-P_{i+1}|, P_9=P_1 \tag{10}$$

where, P_i is the neighbouring pixel of P. The neighbouring pixels of P are scanned in the anti-clockwise direction as shown below. Ridge endings will have a value 1 for *cn* and bifurcations will have a value 3.

A set of minutiae are extracted and from the extracted minutiae the spurious minutiae are eliminated. The core points and the delta points are also extracted from the fingerprint images as core point is used as the root of the minutiae tree.

Core Point Detection

The detection of core point is very important since the minutiae tree is constructed by keeping the core point as the root of the tree. Poincare index method is a widely accepted method for finding the singular points in a fingerprint (Jirachaweng *et al*., 2011; Joshi *et al*., 2009; Kawagoe and Tojo, 1984; Karu and Jain, 1996). The Poincare index of a element θ_{ij} in the orientation image at position (i, j) is calculated as:

$$P(i,j)=\sum_{k=0}^{N}\Delta(k) \tag{11}$$

where,

$$\Delta(k)=\begin{cases}\delta(k), & if\,|\delta(k)|<90^\circ\\ \delta(k)+180 & if\,|\delta(k)|\le-90^\circ\\ \delta(k)-180, & otherwise\end{cases}$$

Here $\delta(k)$ is the ordered difference between neighbouring elements of θ_{ij} and is calculated as:

$$\delta(k)=\theta\left(i_{(k+1)\bmod N},j_{(k+1)\bmod N}\right)-\theta(i_k,j_k)$$

Under closed curves the Poincare index assumes one of the three values for singular points (Kumar *et al*., 2016). Then:

$$P(i,j)=\begin{cases}0^\circ, & if\ pixel(i,j)\,does\,not\,belong\,to\,singular\,region\\ 360^\circ, & if\ pixel(i,j)\,belong\,to\,whorl\\ 180^\circ & if\ pixel(i,j)\,belong\,to\,core\\ -180^\circ & if\ pixel(i,j)\,belong\,to\,delta\end{cases}$$

Figure 2 shows a fingerprint image and its core point and minutiae.

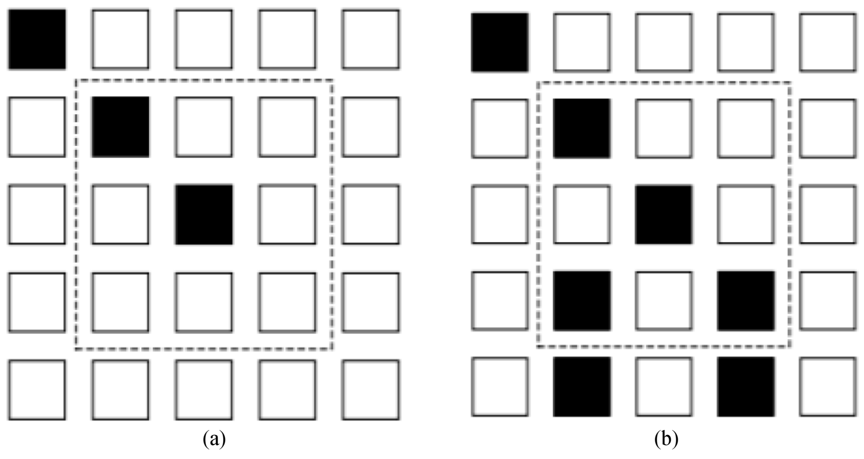

Fig. 1: Crossing number values for ridge ending and bifurcation (a) *cn* = 1 for ridge ending (b) *cn* = 3 for bifurcation

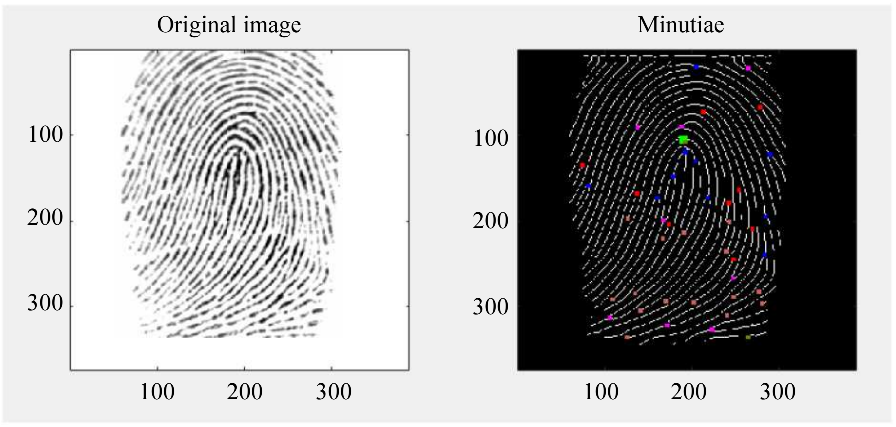

Fig. 2: Original fingerprint image and its core point and minutiae

Pre-Enrolment Steps

After the preprocessing steps on each image, the required features are extracted from the images in the database. Our method extracts two kinds of features from each fingerprint: ridge ending and ridge bifurcation. Before enrolling the fingerprints, we need to carry out some steps which makes the enrolment easy and thus gives an efficient tree structure. The following are the steps carried out before enrolling each fingerprint:

1. Get the extracted core point and let the co-ordinates be (x, y)
2. Find the *Euclidean distance* between the core point and each minutia and arrange the points in the increasing order of their distance
3. Prepare the list of minutiae in the increasing order of their distance from the core point
4. For each minutia m_i calculate two angles. (1) the angle between mi and m_{i+1} (2) the angle between mi and m_{i+2}
5. Create a list of minutiae with required information (i.e., x, y, Euclidean distance, angle1 and angle2)

We are not considering the fingerprints from which a core point cannot be detected. The minutiae points for enrolment are selected according to the Euclidean distance from the core point and we can select as many number of points as required. Since the searching time on a tree depends on the height/depth of the tree, this organization will give an improved search time with an average number of minutiae.

The fingerprints in the database are arranged in the form of a tree. A single tree is built to represent all the fingerprints and the fingerprints are enrolled at the leaf nodes. The root of the tree is the core point of the fingerprint. The minutiae points are arranged in the internal nodes and they are arranged based on the orientation and distance. The depth of each path in the tree depends on the number of minutiae points considered while enrolling the fingerprint. This arrangement provides an index structure for the database. For matching a fingerprint we traverse the tree from root and when we reaches a correct leaf we will obtain a perfect match.

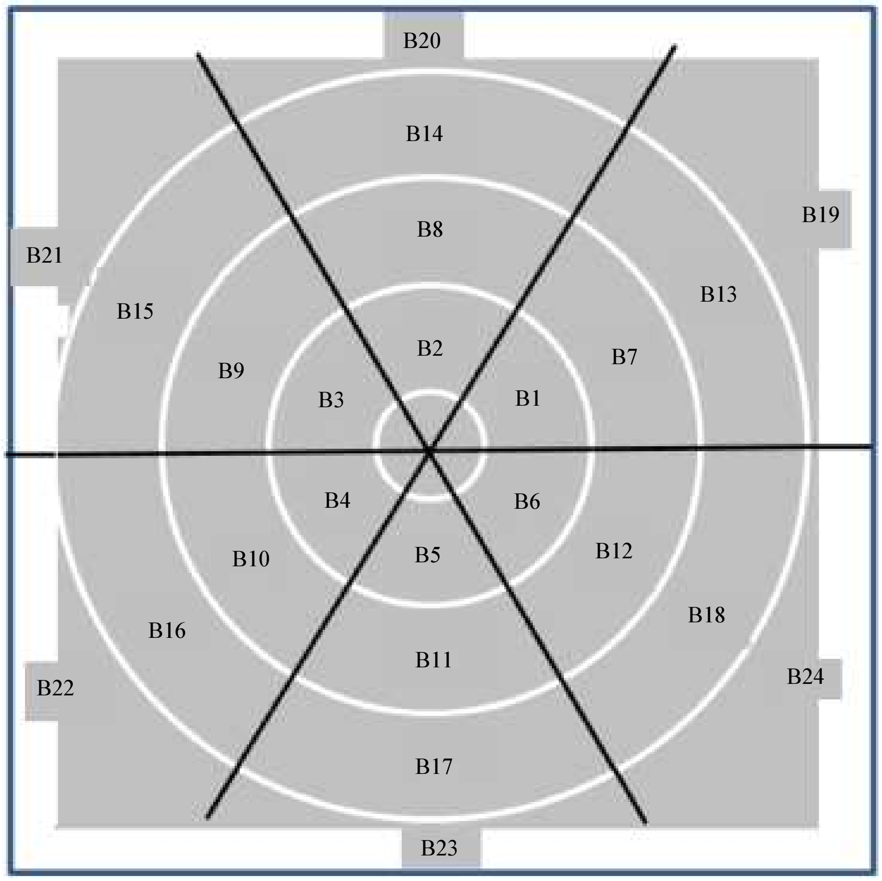

Fig. 3: Basic bin arrangement of 24 bins

How to Create the Minutiae Tree?

Binning

The minutiae points are selected based on the Euclidean distance from the core point. The minutia which is very near to the core point is selected first and binned into one of the bins in the first level of the tree. For our experiment we have constructed 24 different bins based on the orientation and distance from the core point (6 for orientation and 4 for distance). The basic bin arrangement is shown in Fig. 3. The number of bins can be varied according to the maximum Euclidean distance allowed for enrolling. Here we have selected 60 pixels as the maximum distance and so the minutiae which are up to 60 pixels away from the core point are considered for enrolment. At each level of the tree, maximum *n* number of minutiae are binned where *n* is the number of fingerprint images in the dataset.

Table 1 shows the range of values for orientation and distance that come under various bins. Figure 4 shows the structure of bins associated with each branch of the tree and Fig. 5 shows the dynamic list associated with each bin.

If more than one minutia is binned into a particular bin, then these minutiae can be maintained by using a dynamic list. The list keep the minutiae in the increasing order of their distance from the core point. Each minutiae in the list hold the x and y coordinates, the euclidean distance, the angles calculated.

Creating Minutiae Tree by Enrolling the Fingerprints

Let *T* is the template fingerprint which we want to insert in the tree which contain n minutiae points. Then *T* can be defined as a set of *n* minutiae which are arranged in the increasing order of the Euclidean distance from the core point. *T* can be represented as:

$$T = m_1^T, m_2^T, m_3^T, \dots, m_n^T$$

where, each minutiae m_i^T is defined as:

$$m_i^T = \left(x_i, y_i \theta_i, t_i\right)$$

where, x_i and y_i are the *x* coordinate and y coordinates of the i^{th} minutiae, θ_i is the orientation of the minutia and t_i

is the type of the minutia of the fingerprint template *T*. The tree is constructed by creating nodes for each minutiae and thus for a fingerprint with *n* minutiae the corresponding branch of the tree grows up to a depth *n*, with the fingerprint enrolled at the leaf node.

Our method applies the concept of bins for categorizing the minutiae. Each minutia is put in a particular bin by considering some of the features of minutia itself and also by considering some features relative to the root (core point). The branch selection is done by comparing the type of the minutia. Since we are considering only two types of minutiae, our minutiae tree is having only two branches in a high level view. The high level structure of the minutiae tree is shown in Fig. 6. Each node in the tree is having a number of bins to accommodate the minutiae based on their relative feature values. After selecting the appropriate branch, our algorithm selects a suitable bin for the minutiae. The binning is done based on the orientation of the current minutia and the distance between the root and the current minutia. We have divided the orientation into 6 bins and distance into 4 bins. So in effect a minutia could enter in any one of the available 24 bins. Each bin in turn contains a dynamic list which can grow up to n. The list stores the required information about the minutia which falls in a particular bin.

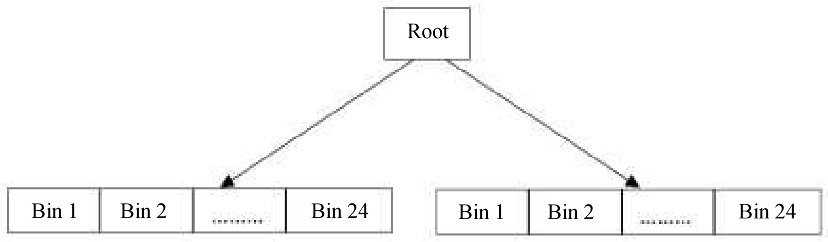

Fig. 4: Bins associated with the branches of the minutiae tree

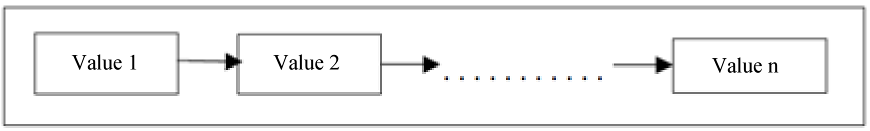

Fig. 5: Dynamic list for the Bin *x*

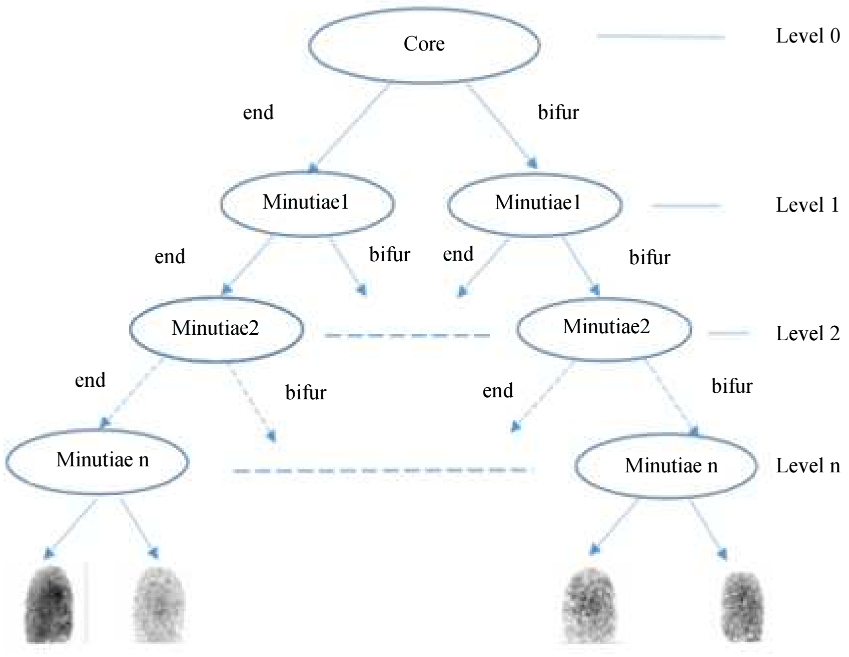

Fig. 6: The high level structure of the minutiae tree

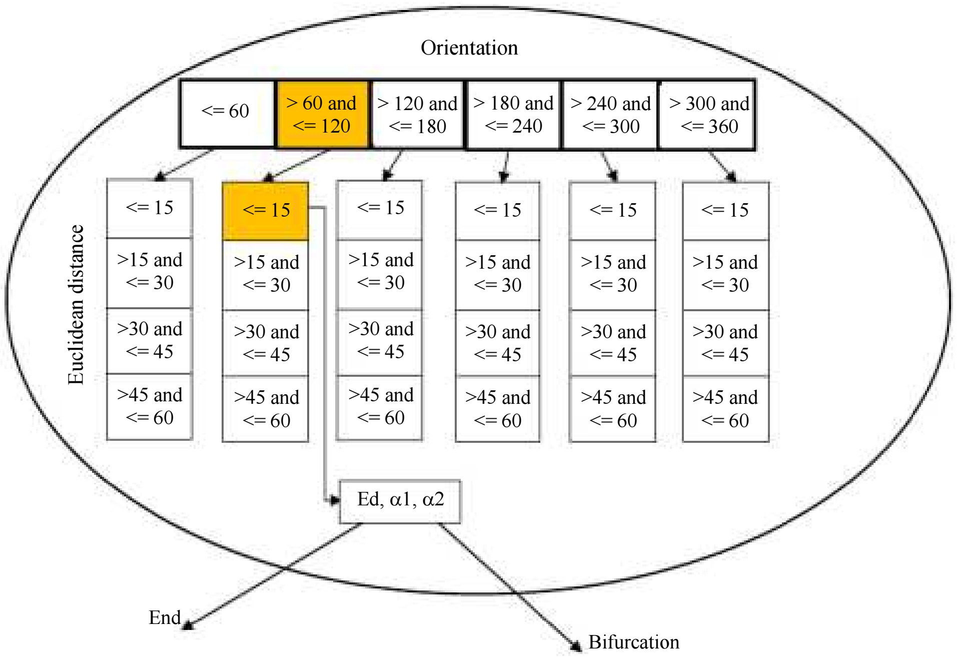

Fig. 7: Structure of a node in the minutiae tree

Table 1: Range of values of angle and distance for various bins

	≤ 60	> 60 and ≤ 120	> 120 and ≤ 180	> 180 and ≤ 240	> 240 and ≤ 300	> 300 and ≤ 360
<= 15	Bin 1	Bin 2	Bin 3	Bin 4	Bin 5	Bin 6
> 15 and ≤ 30	Bin 7	Bin 8	Bin 9	Bin 10	Bin 11	Bin 12
> 30 and ≤ 45	Bin 13	Bin 14	Bin 15	Bin 16	Bin 17	Bin 18
> 45 and ≤ 60	Bin 19	Bin 20	Bin 21	Bin 22	Bin 23	Bin 24

For enrolling the fingerprints the minutiae list which is ordered based on the Euclidean distance is considered. The first minutia in the list is considered and the type of the minutia is checked. If the minutia is of type *ridge ending*; it will be stored as the left child of the root and if the minutia is of type *bifurcation*; it will be stored as the right child of the root. Then the orientation and the Euclidean distance of the minutia are compared with the range of values associated with the bins. A suitable bin is found out and the minutia is put in the dynamic list of that bin. The list is maintained with each bin to accommodate all the minutia falling in the bin. The structure of a node is shown in Fig. 7.

We have used the following algorithm for enrolling the fingerprints in the tree. The output of the pre-enrolment stage is used as input for the enrolment.

Algorithm:

Input: Core point and the list from the pre- enrolment stage
Output: A minutiae tree with fingerprints enrolled as Leaves

Steps:

1. Core point is made as the root of the tree
2. For each minutia in the list
    (a) if the type of minutia is end branch to left else branch to right
    (b) find a suitable bin according to the orientation and *Euclidean distance*
    (c) store the details of the minutia in the dynamic list of that bin
3. Enrol the corresponding fingerprint in the leaf node

A general tree structure is implemented for enrolling the fingerprint. The core point in the fingerprint is made as the root of the tree. The minutiae points in the fingerprint are considered in the increasing order of their Euclidian distance from the root (core point). Then the required features of the minutiae points are considered and based on these features nodes are created for each minutiae in the fingerprint. Our algorithm enrols the fingerprint at the leaf nodes. At each level of the tree the values associated with each minutia are stored with a pointer to the parent(previous minutia in the list) as well as a pointer to the child (or the next minutia in the list). For a fingerprint having n minutiae, our tree structure allows only one enrolment for the fingerprint. So the space required for the tree can be reduced considerably. Fig. 8 shows a sample tree created while enrolling a fingerprint with first five minutiae of the fingerprint.

The internal organization of the tree is shown in Fig. 9. The root of the tree is having pointers to the left node and right node. Each node of the tree has 24 bins and each bin has a dynamic list associated with it to manage the entries. So each minutiae from N fingerprints is binned into any one of the 24 bins associated with a node at each level. The minutiae are stored in the dynamic list with parent address, child address and with the values related to the minutiae. The parent address and child address are stored to get the exact position of the minutiae in the bin. In each bin the values associated with the minutiae are stored along with the parent node address and the child node address which makes searching easier. Figures 10 and 11 shows the structure of the tree after enrolling two fingerprints. In this way all the fingerprints in the database are enrolled in the minutiae tree. The depth of the tree depends on the maximum allowed euclidean distance and the number of minutiae selected for enrolling. We can make the height of each path identical by selecting equal number of minutiae for enrolment.

Fingerprint Matching

The next important phase is matching a probe image. The pre-processing steps and the pre-enrolment steps are applied to the probe image prior to matching. Matching is done in a similar way enrolment is done. The list of minutiae of the probe image may contain *p* number of minutiae which are arranged in the increasing order of their distance from the core point. The first minutia in the list will be considered and based on the type of the minutia the appropriate branch of the tree is selected. Then based on the feature values of the current minutia the search is progressed towards a particular bin and hence to the list in that bin. The feature values of the current minutia of the probe image are now compared with the values in the list. A matched value will be found with the help of a threshold value and now the search or the matching process progresses with next minutiae by selecting the suitable branch from the previous minutia. This process continues and when the matching reaches the leaf node, that means a matched fingerprint is found.

For the minutiae values which are on the boundaries of the bins, the search will progress through multiple paths. This approach is helpful to increase the chances of finding the correct minutiae paths. Also in situations where the search process is unable to move further, backtracking is applied to find a correct path.

How to Handle Missing Minutiae

An important issue that can occur and which will lead to a wrong search path is the problem of missing minutiae. If the features extracted from the probe fingerprint miss some minutiae compared with the enrolled fingerprints, the matching process will not give a correct result. To overcome this situation we have applied a prediction logic in our method. By applying this prediction logic it is found that our algorithm progresses through the correct path even if some minutiae are missing and gives the correct result. Thus we are able to reduce the error rate up to an extent.

If m_1, m_2, …, m_n is the minutiae set of the query fingerprint in the increasing order of the Euclidean distance from the root, at each node or with each minutiae we are recording the angle that m_i and m_{i+1} makes with the root. When the query fingerprint comes, at each node this angle will be checked and if it is greater than a particular threshold it is assumed that the minutia m_{i+1} is missing in the query fingerprint and the search will be continued with the next node in the search path if the angle between mi and m_{i+2} in the enrolled fingerprint satisfies with that of the query fingerprint. By using this simple method it is found that the mismatch due to a missing minutia in the query fingerprint could be controlled.

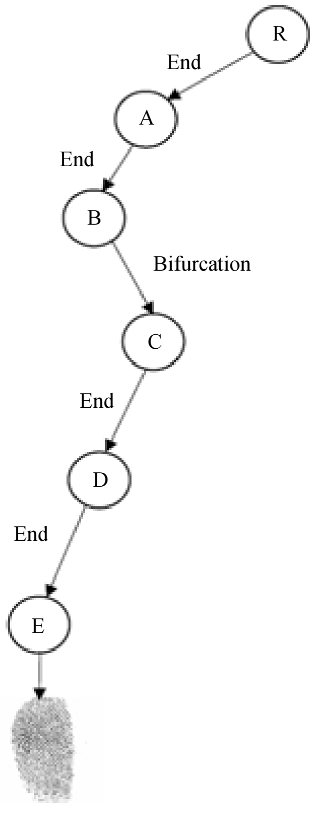

Fig. 8: Sample tree created when enrolling one fingerprint

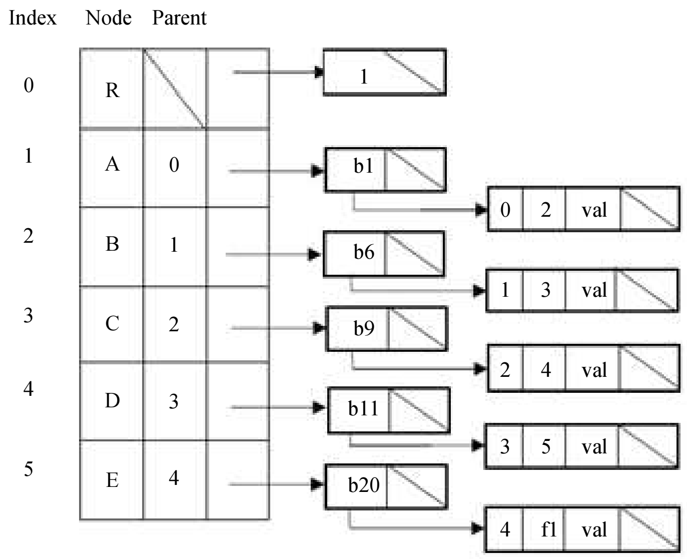

Fig. 9: Internal structure of the sample tree created when enrolling one fingerprint

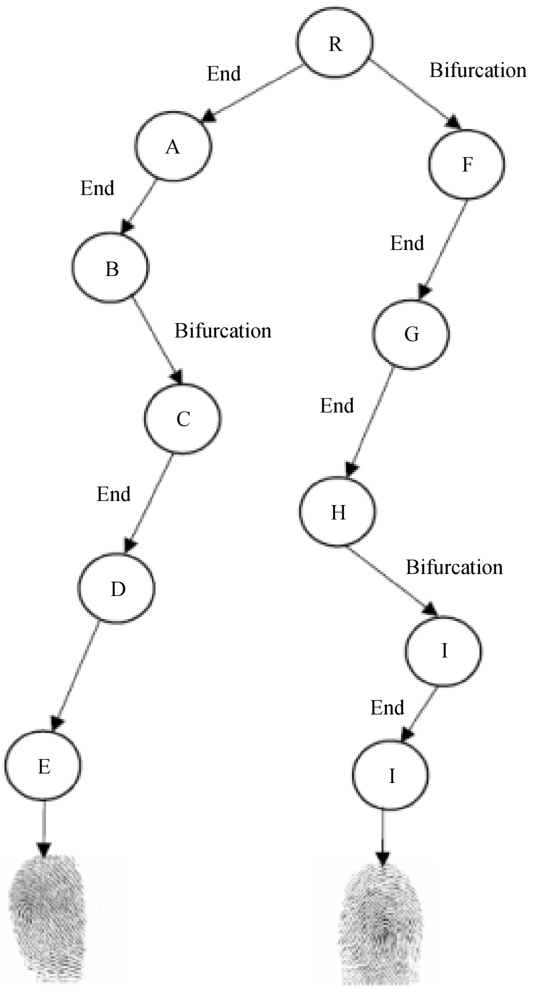

Fig. 10: Sample tree Created when enrolling two fingerprints

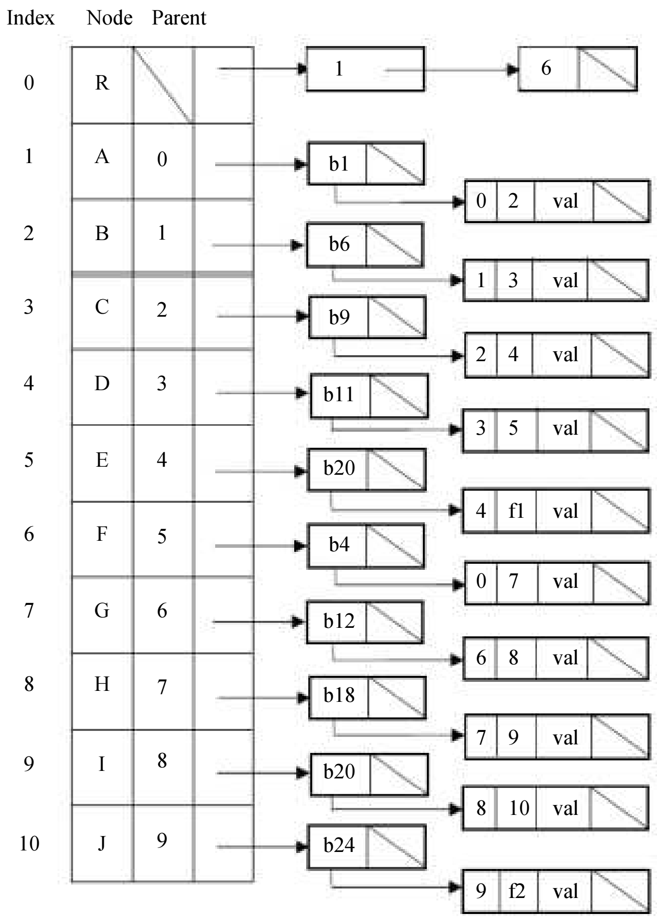

Fig. 11: Internal structure of the sample tree created when enrolling two fingerprints

Suppose m_{T1}, m_{T2}, m_{T3}, m_{T4} are the minutiae of the template fingerprint and m_{P1}, —-, m_{P3}, m_{P4} are the minutiae of the probe fingerprint. The —- indicates a missing minutia. Let C_T and C_P be the core points of the template and probe fingerprints. When the first minutia is considered in the pre-enrolment stage the angles $< m_{T1}\ C_T\ m_{T2}$ and $< m_{T1}\ C_T\ mT_3$ are calculated and stored in the corresponding bin of the minutia. When the first minutiae of the probe fingerprint is considered the angle $< m_{P1}\ C_P\ m_{P3}$ is calculated. The angles $< m_{T1}\ C_T\ m_{T2}$ and $< m_{P1}\ C_P\ m_{P3}$ are compared and since the second minutia is missing in the probe fingerprint these two will not match. So our algorithm will compare $< m_{T1}\ C_T\ m_{T3}$ and $< m_{P1}\ C_P\ m_{P3}$ which gives a positive result. So it is understood that the second minutia is missing in the probe fingerprint and the search is continued with the third minutia of the template omitting the second minutia.

Performance Analysis

The dataset used for this experiment is FVC 2002 in which the images are of size 300×300. The dataset contains 8 images each of different users.

The performance of the above described system is analysed in terms of running time and accuracy. In a single node of the tree there are 24 static bins and a dynamic list corresponding to each bin. A minutia based on its local feature values is added to one of the lists of these 24 bins. The matching time or searching time of the above mentioned system is same as the enrolment time. The maximum possible children for each node is n where n is the number of enrolled fingerprints. So the maximum possible size of the tree is $O(nd)$ where d is the depth of the tree which is equal to the number of minutiae used to build the tree.

Running Time

The running time of the algorithm is comprised of the time required to enrol a fingerprint into the tree and the time required to carry out a search. The running time of the algorithm is found to be slightly increasing as the depth of the tree increases. Compared to the single path search, multiple path search takes more time as it need to search multiple paths. The running time of the algorithm is compared with previously proposed algorithm

(Mansukhani *et al*., 2010). The comparison of the average running time for single search path and multiple path search when the depth of the tree increases is given in Table 2 and 3.

Figure 12 shows the comparison of running time of the proposed algorithm during single path search and multiple path search when the number of minutiae used for enrolling the fingerprint is increasing. It can be seen from the graph that the running time of the algorithm is slightly increasing as the number of minutiae is increasing.

Efficiency

From the database fingerprint images of 10 persons were selected. 8 images of the same finger per person are considered for constructing the tree. Then there will be 7 test cases for one instance of the fingerprint. So for one person, there will be (8*7)/2 = 28 test cases since we are avoiding repeated matching. So the total test cases for 10 persons is 28*10 = 280. The efficiency of the method is calculated as:

$$Efficiency = \frac{Number\ of\ positive\ \ matches}{Total\ number\ of\ text\ \ cases}$$

The proposed method is tested by varying the number of minutiae selected for enrolling the fingerprints. We conducted various experiments for testing the efficiency of the proposed method. The experiments took 5, 10, 15, 20 and 25 minutiae which are near to the core point to create the tree. When the tree is created with 5 minutiae points for each fingerprint, the efficiency obtained is 258/280 = 91.07%. Hence the Genuine Acceptance Rate (GAR) is 91.07% and the False Rejection Rate (FRR) is 8.93%. Thu positive matches obtained with 10 minutiae points is 259. So the efficiency obtained is 92.5%. The GAR is 92.5% and FRR is 7.5%. The positive matches obtained when considering 15 minutiae points for constructing the tree and for matching is 258. Then the efficiency is 92.14%. The GAR is 92.14% and FRR is 7.86%. We got 260 positive matches when tried with 20 min points and so the efficiency is calculated as 92.85%. The GAR is 92.85% and the FRR is 7.15%. The positive matches obtained when considering 25 min points for constructing the tree and for matching is 260. Then the efficiency is 92.85%. The GAR is 92.85% and FRR is 7.15%. Table 4 shows the FRR and GAR values obtained for single path search.

When the minutiae are at the boarders of the bin, sometimes the search will progress through multiple paths. Table 5 shows the FRR and GAR values obtained during multiple path search. The FRR and GAR values obtained during single path search and multiple path search when the tree is created with varying number of minutiae are shown in Fig. 13.

Table 2: Average running time in seconds for the method in (Mansukhani *et al*., 2010)

Number of minutiae considered	Single path search	Multiple path search
5	0.112	0.060
6	0.032	0.040
7	0.028	0.042
8	0.028	0.042

Table 3: Average running time in seconds for the proposed method on FVC 2002

Number of minutiae considered	Single path search	Multiple path search
5	0.066	0.074
10	0.069	0.082
15	0.073	0.090
20	0.081	0.099
25	0.089	0.109

Table 4: The FRR and GRR values in different cases of single path search

	Number of Points	FRR	GAR
Case 1	5	8.93%	91.07%
Case 2	10	7.5%	92.5%
Case 3	15	7.86%	92.14%
Case 4	20	7.15%	92.85%
Case 5	25	7.15%	92.85%

Table 5: The FRR and GRR values in different cases of multiple path search

	Number of Points	FRR	GAR
Case 1	5	10.00%	90.00%
Case 2	10	9.64%	90.36%
Case 3	15	9.64%	90.36%
Case 4	20	10.36%	89.64%
Case 5	25	10.00%	90.00%

Table 6: The penetration rate of the proposed method

Hit Rate	Penetration rate
60%	1.0
70%	1.4
80%	1.9
85%	2.5
90%	3.1

Table 7: The average penetration rate of the proposed method compared with existing methods

Method|Hit Rate	60%	70%	80%	90%	95%	100%
Minutiae Quadruplets (Iloanusi *et al*., 2011)	6.8	8.68	10.5	15.0	17.6	21.5
Minutiae Neighbourhood (Vij, 2013)	1.0	1.90	3.9	8.6	14.0	57.0
Proposed method	1.0	1.40	1.9	3.1	4.2	8.5

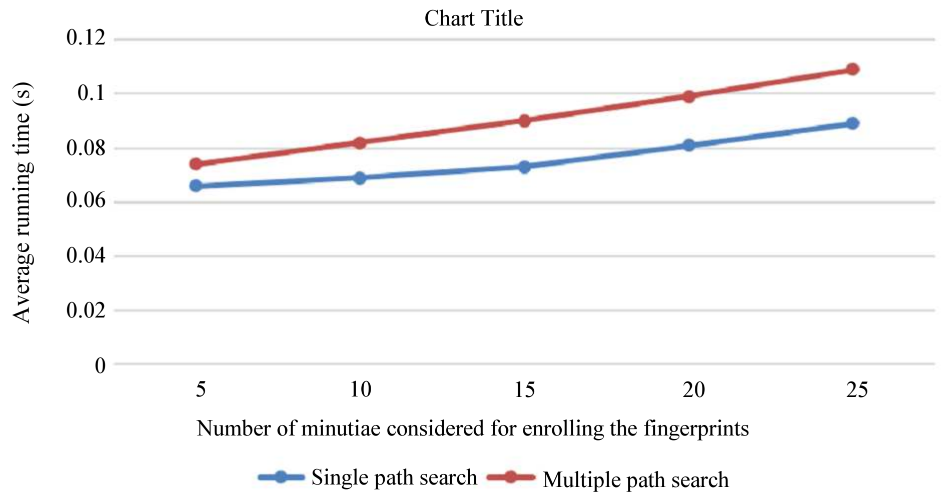

Fig. 12: Average running time in seconds

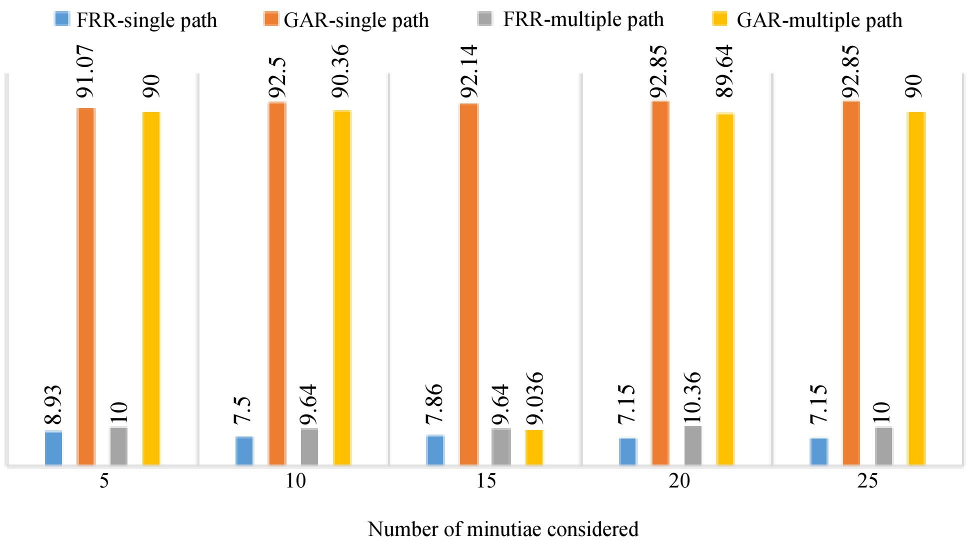

Fig. 13: Comparioson of FRR and GAR values in single path searching and multiple path searching

Penetration rate is the average percentage of database searched over all test fingerprints. The *penetration rate* for the proposed method is given in Table 6 and 7 shows a comparison between the proposed method and some other indexing methods. The penetration rate of the proposed method is less compared to other methods since we have applied binning based on the relative features of the fingerprint minutiae by which the percentage of database searched is decreased.

Conclusion

Fingerprints have been used for personal identification from long back since they are very unique. Fingerprint matching plays an important role in fingerprint verification as well as in fingerprint identification. For a better response time or for getting a real time response we have to reduce the search space while matching. The method proposed in this paper considerably reduces the search space and thus reduces the matching time.

A fingerprint matching system with a tree based indexing structure has been implemented and tested. Our system is robust to translation and rotation of fingerprints. Another important quality of our system is that it is scalable and the time required for enrolling a fingerprint remains same irrespective of the size of the database. The searching or matching time found to be slightly increasing depending on the number of minutiae included in the tree. The penetration rate of the proposed method is found to be very less compared to other indexing techniques. The accuracy can be increased by including more relative characteristics of the minutiae.

Author's Contributions

All authors equally contributed in this work.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that the coauthor has read and approved the manuscript and there are no ethical issues involved.

References

Bazen, A.M. and S.H. Gerez, 2001. Segmentation of fingerprint images. Proceedings of the Workshop on Circuits, Systems and Signal Processing, (SSP' 01), Veldhoven, The Netherlands, pp: 276-280.

Cappelli, R., M. Ferrara and D. Maltoni, 2010. Minutia cylinder-code: A new representation and matching technique for fingerprint recognition. IEEE Trans. Patt. Anal. Mach. Intell., 32: 2128-2141. DOI: 10.1109/TPAMI.2010.52

Chikkerur, S., A.N. Cartwright and V. Govindaraju, 2006. K-plet and coupled BFS: A graph based fingerprint representation and matching algorithm. Proceedings of the International Conference on Advances in Biometrics, Jan. 05-07, Springer, Hong Kong, China, pp: 309-315. DOI: 10.1007/11608288_42

de Boer, J., A.M. Bazen and S.H. Gerez, 2001. Indexing fingerprint databases based on multiple features. Proceedings of the 12th Annual Workshop on Circuits, Systems and Signal Processing, Nov. 25-27, Veldhoven, The Netherlands, pp: 300-306.

Guo, Z. and R.W. Hall, 1989. Parallel thinning with twosubiteration algorithms. Commun. ACM, 32: 359-373. DOI: 10.1145/62065.62074

Hatano, T., T. Adachi, S. Shigematsu, H. Morimura and S. Onishi *et al*., 2002. A fingerprint verification algorithm using the differential matching rate. Proceedings of the 16th International Conference on Pattern Recognition, Aug. 11-15, IEEE Xplore Press, Quebec City, Quebec, Canada, pp: 799-802. DOI: 10.1109/ICPR.2002.1048139

Hong, L., Y. Wan and A. Jain, 1998. Fingerprint image enhancement: Algorithm and performance evaluation. IEEE Trans. Patt. Anal. Mach. Intell., 20: 777-789. DOI: 10.1109/34.709565

Iloanusi, O., A. Gyaourova and A. Ross, 2011. Indexing fingerprints using minutiae quadruplets. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, Jun. 20-25, IEEE Xplore Press, Colorado Springs, CO, USA, pp: 127-133. DOI: 10.1109/CVPRW.2011.5981825

Jain, A.K., J. Feng and K. Nandakumar, 2010. Fingerprint matching. Computer, 43: 36-44. DOI: 10.1109/MC.2010.38

Jiang, X. and W.Y. Yau, 2000. Fingerprint minutiae matching based on the local and global structures. Proceedings of the 15th International Conference on Pattern Recognition, Sept. 3-7, IEEE Xplore Press, Barcelona, Spain, pp: 1038-1041. DOI: 10.1109/ICPR.2000.906252

Jirachaweng, S., Z. Hou, W.Y. Yau and V. Areekul, 2011. Residual orientation modeling for fingerprint enhancement and singular point detection. Patt. Recogn., 44: 431-442. DOI: 10.1016/j.patcog.2010.08.019

Joshi, T., S. Dey and D. Samanta, 2009. A two-stage algorithm for core point detection in fingerprint images. Proceedings of the IEEE Region 10 Conference, Jan. 23-26, IEEE Xplore Press, Singapore, pp: 1-6. DOI: 10.1109/TENCON.2009.5396214

Karu, K. and A.K. Jain, 1996. Fingerprint classification. Patt. Recognit., 29: 389-404. DOI: 10.1016/0031-3203(95)00106-9

Kawagoe, M. and A. Tojo, 1984. Fingerprint pattern classification. Patt. Recognit., 17: 295-303. DOI: 10.1016/0031-3203(84)90079-7

Kour, J., M. Hanmandlu and A.Q Ansari, 2012. Fast fingerprint image alignment. Proceedings of the 2nd International Conference on Computer Science, Engineering and Applications, (SEA' 12), Springer, At New Delhi, pp: 93-99. DOI: 10.1007/978-3-642-30157_10

Kumar, R., P. Chandra and M. Hanmandlu, 2016. A robust fingerprint matching system using orientation features. J. Inform. Process. Syst., 121: 83-99.

Lahby, M., Y. Ismaili, A. Attioui and A. Sekkaki, 2016. Performance analysis of minutia-based fingerprint matching algorithms. Proceedings of the 11th International Conference on Intelligent Systems: Theories and Applications, Oct. 19-20, IEEE Xplore Press, Mohammedia, Morocco, pp: 1-5. DOI: 10.1109/SITA.2016.7772324

Li, G., B. Yang and C. Busch, 2014. A score-level fusion fingerprint indexing approach based on minutiae vicinity and minutia cylinder-code. Proceedings of the 2nd International Workshop on Biometrics and Forensics, Mar. 27-28, IEEE Xplore Press, Valletta, Malta, pp: 1-6. DOI: 10.1109/IWBF.2014.6914238

Liang, X., A. Bishnu and T. Asano, 2007. A robust fingerprint indexing scheme using minutia neighborhood structure and low-order Delaunay triangles. IEEE Trans. Inform. Forens. Security, 2: 721-733. DOI: 10.1109/TIFS.2007.910242

Lindoso, A., L. Entrena, J. Liu-Jimenez and E. San Millan, 2007. Correlation-based fingerprint matching with orientation field alignment. Proceedings of the International Conference on Biometrics, Aug. 27-29, Springer, Seoul, Korea, pp: 713-721. DOI: 10.1007/978-3-540-74549-5_75

Luo, X., J. Tian and Y. Wu, 2000. A minutiae matching algorithm in fingerprint verification. Proceedings of the 15th International Conference on Pattern Recognition, IEEE Xplore Press, pp: 833-836.

Mansukhani, P., S. Tulyakov and V. Govindaraju, 2010. A framework for efficient fingerprint identification using a minutiae tree. IEEE Syst. J., 4: 126-137. DOI: 10.1109/JSYST.2009.2037286

Mehtre, B.M. and B. Chatterjee, 1989. Segmentation of fingerprint images-a composite method. Patt. Recognit., 22: 381-385. DOI: 10.1016/0031-3203(89)90047-2

Mehtre, B.M., N.N. Murthy, S. Kapoor and B. Chatterjee, 1987. Segmentation of fingerprint images using the directional image. Patt. Recognit., 20: 429-435. DOI: 10.1016/0031-3203(87)90069-0

Mngenge, N.A., L. Mthembu, F.V. Nelwamondo and C.H. Ngejane, 2015. An integrated approach to fingerprint indexing using spectral clustering based on minutiae points. Proceedings of the Science and Information Conference, Jul. 28-30, IEEE Xplore Press, London, UK, pp: 1222-1229. DOI: 10.1109/SAI.2015.7237300

Nanni, L. and A. Lumini, 2009. Descriptors for image-based fingerprint matchers. Expert Syst. Applic., 36: 12414-12422. DOI: 10.1016/j.eswa.2009.04.041

Ratha, N.K., S. Chen and A.K. Jain, 1995. Adaptive flow orientation-based feature extraction in fingerprint images. Patt. Recognit., 28: 1657-1672. DOI: 10.1016/0031-3203(95)00039-3

Sankaran, A., M. Vatsa and R. Singh, 2014. Latent fingerprint matching: A survey. IEEE Access, 2: 982-1004. DOI: 10.1109/ACCESS.2014.2349879

Sheng, W., G. Howells, M. Fairhurst and F. Deravi, 2007. A memetic fingerprint matching algorithm. IEEE Trans. Inform. Forens. Security, 2: 402-412. DOI: 10.1109/TIFS.2007.902681

Tamura, H., 1978. A comparison of line thinning algorithms from digital geometry viewpoint. Proceedings of the 4th International Conference on Pattern Recognition, (CPR' 78), pp: 715-719.

Vij, A., 2013. Minutiae local structures for fingerprint indexing and matching. PhD Thesis, International Institute of Information Technology Hyderabad.

Yang, J.C. and D.S. Park, 2008. A fingerprint verification algorithm using tessellated invariant moment features. Neurocomputing, 71: 1939-1946. DOI: 10.1016/j.neucom.2007.12.034

Zhou, W., J. Hu, S. Wang, I. Petersen and M. Bennamoun, 2014. Fingerprint indexing based on combination of novel minutiae triplet features. Proceedings of the International Conference on Network and System Security, (NSS' 14), Springer, pp: 377-388.