Fingerprint Matching Algorithm Based on Tree Comparison using Ratios of Relational Distances

Abinandhan Chandrasekaran
Research Assistant
axc054100@utdallas.edu

Dr.Bhavani Thuraisingham
IEEE Fellow, AAAS Fellow
bxt043000@utdallas.edu

Data and Applications Security Laboratory
Department of Computer Science, University of Texas at Dallas

Abstract

We present a fingerprint matching algorithm that initially identifies the candidate common unique (minutiae)points in both the base and the input images using ratios of relative distances as the comparing function. A tree like structure is then drawn connecting the common minutiae points from bottom up in both the base and the input images. Matching score is obtained by comparing the similarity of the two tree structures based on a threshold value. We define a new term called the 'M (i) - tuple' for each minutiae point which uniquely encodes details about the local surrounding region, where i = 1 to N, and N is the number of minutiae. The proposed algorithm requires no explicit alignment of the two to-be compared fingerprint images and also tolerates distortions caused by spurious minutiae points. The algorithm is also capable of comparing and producing matching scores between two images obtained from two different kinds of sensors, hence is sensor interoperable and also reduces the FNMR in cases where there is very little overlap region between the base and the input image. We conducted evaluations on the FVC-2000 [1] datasets and have summarized the results in the concluding section.

1. Introduction

Fingerprint based biometric authentication and verification systems have gained immense popularity and acceptance ever since their inception. This is primarily because of the ease of operation, installation and easy acquisition of the biometric feature, which in this case is a fingerprint. Matching two fingerprints can be unsuccessful due to various reasons and also depends upon the method that is being used for matching. Very popular methods include minutiae based matching, correlation based matching, pattern matching etc… The quality of fingerprints often plays a major role in affecting the degree of accuracy of the result produced by the matcher. In this paper, we have solely concentrated on attacking two problems, the first being the increased FNMR when the overlap region is very less in area and the second being the sensor interoperability problem, as indicated by Jain et all [2]. The quality of fingerprints can be perceived as a problem in two major cases. [a] Fingerprint quality often deteriorates based on the acquisition device from which it was obtained with respect to overlap regions. While an optical scanner may be able to produce an image as large as 500 * 500 pixels with 500 dots per inch, a solid state scanner will be able to produce an image with dimensions of 300 *300 only. What this effectively means is that, while fingers are placed at different positions on the platen of the scanner at various instances, an optical scanner will be able to capture more area at all times, hence the overlap region between the base and the input image is always considerable enough in area to perform a match. But in a system that deploys solid state acquisition devices, this common area or the overlap region might be low enough to an extent that either a match could not be performed or the result is not accurate when the matching process is executed. Figure 1 illustrates the overlap region problem.

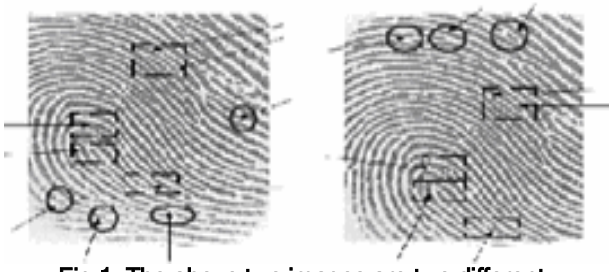

Fig 1: The above two images are two different instances of the same fingerprint.

In figure 1, the areas marked in squares encapsulate minutiae points that are common in both the images, while the areas marked in circles encapsulate minutiae points that are unique to each of the two images. The

Second International Conference on Availability, Reliability and Security (ARES'07)
0-7695-2775-2/07 $20.00 © 2007 IEEE

minutiae points that are inside the circled areas in both the images are unique to each image in the sense that, they are either not available in the other image or they are spurious minutiae due to poor image clarity. [b] The second case is the general one where the fingerprint image lacks clarity. Again, there can be many situations here too. Two or more ridge lines can merge and cause a misleading picture while detecting minutiae points and spurious minutiae may pop up in a poor image. Because of this, many minutiae based matching algorithms fail since the number of common minutiae that can be found on both the images is low enough and results in an undesirable matching score. The sensor interoperability problem was highlighted through a case study by Jain et all in 2004 [2]. Majority of the current algorithms assume that the base and the input images are obtained from the same fingerprint acquisition device. The case study points out that the evaluation protocol used in FVC 2002 matched images originating from the same sensor only, even though images from three different kinds of sensors were used. It also points out the difficulty in incorporating sensor interoperability feature into an algorithm, although there are common data formats for switching the feature sets. The reason why this feature is a desired one is that, future deployments of biometric systems might require the parallel usage of raw biometric information from various sources, hence necessitating the need for an algorithm that is sensor interoperable. Note that this is a desired feature for the general biometrics deployed security scenario, including face and iris biometrics. The proposed algorithm uses relational distances between minutiae points as ratios and how it alleviates the sensor interoperability problem is discussed in the forth coming sections of this paper. Figure 2 shows variations between two images (images of the same finger) obtained from two different sensors.

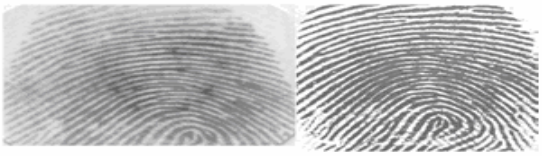

Fig 2: Left - Low cost Optical Scanner
Right - Low cost Capacitive Scanner

A sensor interoperable algorithm should be able to enroll one of the images in Fig 2 as the base image and then verify the other when it is given as the input image. That would be the ideal and desired output of the algorithm. We have concentrated on the above mentioned two problems, the first problem being the inefficient matching score due to low overlap regions, which in turn leads to high FNMR and the second problem is the sensor interoperability issue. Section 3 explains in detail about how the proposed algorithm attacks the above mentioned two problems. The rest of the paper is organized as follows. In section 2, we discuss about some related work. In section 3, the proposed algorithm is detailed. Section 4 summarizes the evaluations and the result. Section 5 concludes the discussion.

2. Background and Related Work

Numerous algorithms have already been proposed to deal with fingerprint matching and recognition. Jain et all proposed a filterbank matching algorithm [3] that employs gabor filters to obtain both local and global information which in turn becomes a FingeCode. Matching is based on comparing the Euclidean distances between two such FingerCodes. In [4], a hybrid matching algorithm that considers both minutiae point data and texture information is detailed. A few graph based algorithms ([5,][6],[7]) do matching by performing operations based on graph principles. The algorithm [8] proposed by Nalini K Ratha et all creates two Minutiae Adjacency Graphs, one each for the base and the input image, in which the vertices of the graph represent minutiae from the matched minutiae set. The algorithm proceeds in three phases to arrive at the matched minutiae set. The first phase results in a minimum set of matched node (minutiae) pairs based in their neighborhood information. The second phase results in an increased set of matched pairs by comparing the distances from the unmatched minutiae set to the matched minutiae set, which indeed was obtained because of the first phase. The algorithm proceeds to the third phase in which the neighborhood of each feature is extended, if a decision could not be arrived at the end of the second phase. Point Pattern Matching problems ([9] and [10]) are also used in fingerprint matching though they are computationally expensive. [11] proposes a solution to fingerprint matching by incorporating ideas associated with point pattern matching problem. Werner Olz and Walter Kropatsch proposed an algorithm [12] that brings the entire ridge topology into consideration. Each ridge is assigned a symbol based on their kind, which includes a ridge ending, ridge bifurcation etc… These symbols are considered to be nodes/vertices of a graph, and two such graphs that are drawn from the entire ridge topology of the base and the input image is compared to arrive at a result. Fingerprint Classification and indexing techniques are prevalent ([13], [14] and [15]), some of which discriminate between fingerprints based on the appearance of the core area. In [16], a new term called K-plet is introduced, which may either refer to

'K' nearest neighbors of a minutia or all neighbors within a circular radius etc… An adjacency graph is drawn for each of the K-plets and then the CBFS (Coupled Breadth First Search) algorithm is used to traverse through the nodes of the graphs to produce a matching score. The K-plet construction and the resulting adjacency graphs are used to match local regions within both the images, while CBFS consolidates all the local matches. In [17] and [18], multiple feature descriptors and classifiers are used in parallel to enhance the result of the matching process. Techniques discussed in [19], [20], [21] and [22] try to align the global patterns of the ridges and valleys rather than matching minutiae points only. We realized the importance of simultaneously considering both local information and global features from the above papers. The proposed algorithm in this paper considers each of the minutiae point in the base image to be a candidate common point minutiae (a common point minutiae is one that is available in both the base and the input image) initially, and derives the 'M (i) – tuple 'for all the minutiae. Then the M (i) - tuples are computed for minutiae points in the input image and are compared to the already computed tuples from the base image. Based on tuple matching, common point minutiae sets (set of points that are available in both in the base and the input image) are obtained. Note that we will have two such sets, one from the perspective of the base image and one from the perspective of the input image. The reason why we will have two sets is explained in section 3. After these sets are obtained, a tree like structure is drawn between the minutiae points in both the sets and the trees are compared to arrive at a matching score result. The above overview of the proposed algorithm is explained in great detail in the following section.

3. Proposed Algorithm

The algorithm proposes to match two fingerprints provided that their minutiae points are identified already. In order to test and verify our algorithm, we used the algorithm proposed by Sharat et all in [23], to extract minutiae points from a given fingerprint image. In short, [23] does fingerprint image enhancement based on STFT (Short Time Fourier Transform) analysis to improve the overall clarity of a fingerprint image and also provides it in a binary format. We use [23] to obtain the enhanced binary image, after which we thin down the binary image down to a width of one pixel so as to retrieve minutiae points from the image. What we have now is a pair of fingerprint images with their minutiae points identified. The proposed algorithm in this paper involves two phases in order to produce a matching score. In the first phase, the methodology used to obtain the common minutiae point set (minutiae points present in both the base and the input image) is explained. In the second phase, the technique to perform actual matching based on the common minutiae point set is detailed. Figure 3 explains how the minutiae points are obtained using [23].

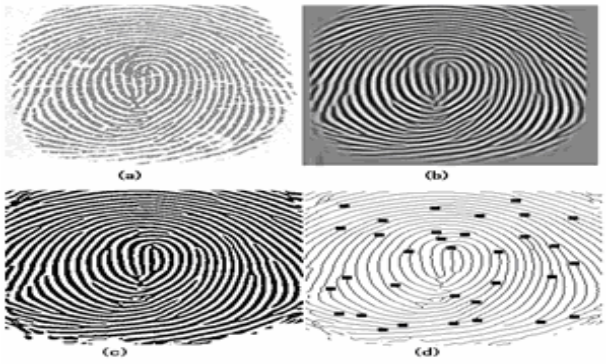

Fig 3: (a) Original image (b) Enhanced image (c) Enhanced Binary image (d) Thinned image

3.1. Finding Common Points – Phase 1

The prime purpose of this phase is to find the number of common minutiae points available in a pair of fingerprint images. Given two fingerprint images with 'N1' and 'N2' identified minutiae points respectively (where N1 need not be equal to N2), this phase outputs the 'M' common minutiae points, which would be available in both the images. Effectively, if N1 represents the set of minutiae points in image 1 and N2 represents the set of minutiae points in image 2, M would be the intersection of N1 and N2 ($M = N1 \cap N2$). We define a new term called the 'M (i) – tuple' to represent information about a minutiae that would identify it uniquely among the set of all minutiae. The M (i) – tuples of a pair of minutiae can be compared/matched to find if they both are the same or not. The method followed to arrive at M follows in the next sub-section. When two images with identified minutiae points are given as input, the algorithm considers one image to be the base image (BM) and the other image to be the input image (IM). Either of them can be BM or IM and vice versa. Figure 4 shows the ideal output after phase 1.

3.1.1. M(I) – Tuples in base image (BM)

The base image has 'N1' number of minutiae points and N (BM) is the set of all minutiae in the base image. Now, the M (i) – tuple (i = 1 to N1) for each minutiae point is calculated as follows:

Step 1: For each minutiae i = 1 to N1, the 5 nearest minutiae points are found. This is done by calculating the Euclidean Distances from the 'i'th minutiae point to all the other minutiae points in the set N (BM) and noting down the 5 nearest minutiae points with respect to Euclidean Distances.

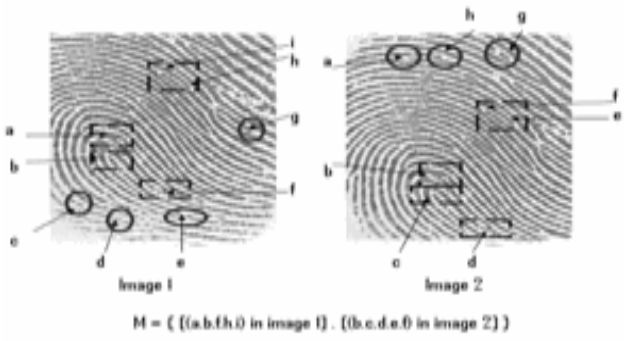

Fig 4: Common points (in squares) are listed in M, others (in circles) are not listed.

Step 2: If i1, i2, i3, i4 and i5 are the 5 nearest minutiae points of i, then we calculate M (i) – tuple in the following way: (a) Calculate distances i – i1, i – i2, i – i3, i – i4, and i – i5. Note that distance 'i – iN' means the Euclidean Distance between the points i and iN. So here, distance i – i1 means the Euclidean distance between minutiae point i and i1 and so on. (b) Find the following 10 ratios (i - i1): (i - i2), (i - i1): (i – i3), (i - i1): (i – i4), (i - i1): (i – i5) , (i – i2) : (i – i3), (i – i2) : (i –i4), (i – i2) : (i – i5), (i – i3) : (i – i4), (i – i3) : (i – i5), (i – i4) : (i – i5) according to the following equation : (a – b): (a – c) =

Max {(a-b), (a-c)} / Min {(a-b), (a-c)}. ------- {1}

So all the ten ratios when calculated using {1} would yield a number value that is definitely greater than 1. The decimals are rounded to two digits and the 10 values are entered in to the tuple. (c) While calculating the ratio of distances between (a – b): (a – c), the angle that is formed between 'bac' or 'cab' at 'a' is also calculated and entered in the tuple correspondingly near to where the ratio of (a – b): (a – c) is entered. Calculation of angles is also explained below.

In figure 5, While finding the ratio (i – i1): (i – i2), the angle between them can be found by the following way. Extend any one of the edges (i – i1) or (i – i2) beyond point 'i'. Here, the extended edge is (i – i1). The angle formed by (i1 – i –extended line) will be 180 degrees always, since it is just an extension (Angle 1, in figure 5). The remaining 180 degrees is split up by two angles, Angle 2 which is (Extended line – i –i2), while the other angle is the one that we want which is angle (i1 - i - i2) or (i2 - i - i1). So always this angle would never be greater than 180 degrees. Table 1 displays how M(i)– tuple of each minutiae would look like. This is how the tuple for each minutia in the set N (BM) is constructed.

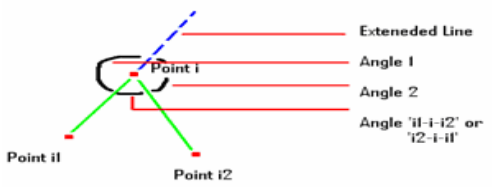

Fig 5: Finding the angle between two edges.

3.1.1. M(I) – Tuples in input image (IM)

Steps that were followed for the Base Image apply to the Input image also. If N2 represents the number of minutiae in the input image and if N (IM) represents the set of all minutiae in the input image, run Step 1 and 2 as described in 3.1.1. Now we have the tuples of all the minutiae points in the input image as well. Now, for all the M (i) – tuples in the input image, where i = 1 to N2, compare with all the tuples of the base image.

Table 1. Displaying an example tuple

M (i) – Tuple

S.No	Ratios	Degrees
1	1.34	39
2	2.23	78
3	2.67	145
4	1.98	122
5	2.12	101
6	1.09	77
7	2.22	100
8	1.67	34
9	1.09	90
10	2.00	169

While comparing two tuples, one from the input image and one from the base image, those two tuples are considered to be the same if a minimum of 2 or more ratios along with the corresponding angles match. A matching of two ratios along with the angles means that for that particular minutiae ('i'), there is a similar corresponding minutiae ('isim') in the other image which also has two minutiae points at relational distances with specified angles, like 'i'. So the threshold for two tuples to match is two ratios along with the specified angles. All such matched tuples, which invariably are minutiae points are strictly Candidate Common Points, which would be available in the both the images. No conclusion is arrived at, at the end of phase 1. Two things will have to be explained here before proceeding to the matching phase. The first thing is to explain how two matching ratios is reason enough to accept a tuple/minutiae point as a Candidate Common Point. Consider the following scenario. In fig 6, when ratios of distances of AB: AC, AC: AD along with their internal angles from the base image match with the ratios of AB: AD, AD: AE along with the internal angles from the input image, it is fair

enough to consider the point A from the both the images as a Candidate Common point because, it is evident that there is some minutiae point 'X' in both the images that have 3 similar minutiae points in its neighborhood, at the same relational distances and internal angles. Note that the distances AB,AC and distances AB,AD from the left and the right images respectively may not be the same, but the ratios AB: AC (left) = AB: AD(right) might be the same using {1}. Similarly, the internal angles might be the same for the following reason. The internal angles are given a range of plus or minus 3.5 degrees to match, since, when comparing an image acquired from a solid state device with an image acquired from an optical device, the size of the images may differ by around more than 300 * 300 pixels. For example, the base image may be a 300 * 300 image from a solid state device, while the input image may be from an optical device with size 600 * 600. For these sensor interoperable matches, the 3 degree range is given. If both the images are from the same sensor, a 1.5 degree range is more than enough.

Fig 6: Left: Point A & its 5 nearest neighbors from BM. Right: Point A & 5 nearest neighbors from IM.

The second thing is to answer the question "Why don't we keep a threshold of such tuples to match and straightaway declare the result?" Refer to figure 7. The Ratios of distances of ab: ad, ac: ae along with their internal angles from the base image (left) match with the ratios of ab: ad, ac: ae along with their internal angles. When observed carefully, we can infer that only ratios of ab : ad with their internal angles match in both the images, because when we consider point A to be origin of a X – Y co-ordinate system in both the images, point b falls on the 3rd quadrant and point d falls on the 4th quadrant in both the images. But with respect to the ratios and angles of ac: ae, points c and e fall in the 3rd and 4th quadrant respectively in the base image, while the same points c and e fall in the 2nd and the 1st quadrant respectively in the input image.

Fig 7: Left: Point A & its 5 nearest neighbors in BM. Right: Point A & 5 nearest neighbors in IM.

Hence, only one ratio matches, which is not good enough to say that the point A is a Common Point that lies in both the images. But checking the quadrant information for every minutiae point while trying to match the local neighborhood is a complex task computationally. Hence these points are also included in the Candidate Common Point List for the time being, but are removed in the matching phase. This is explained in the matching section. In the following section, we discuss the matching phase by using a tree like structure. At the end of this phase, after checking each of the tuples from the input image with all the tuples from the base image, the algorithm results in producing the Candidate Common Point List.

3.2. Matching phase

The matching phase of the algorithm does two functions. (1) Separates the Candidate Common Points List into two lists, (a) Confirmed Common Points List and (b) Spurious / Unconfirmed Point List. (2) Uses the Confirmed Common Points List to generate a Matching Score between the Base and the Input image.

3.2.1. Finding confirmed common points list

From N (BM), which is the set of minutiae points in the base image, the algorithm considers only those points that feature in the Candidate Common Point List to create the tree. The remaining points in the set N(BM) are listed in the set N'(BM). After those points are considered, a tree like structure is drawn from bottom up. Similarly from N(IM), algorithm considers points that feature in the Candidate Common Point List to create the tree and the remaining points are listed in the set N'(IM). This is illustrated from figure 9. The Candidate Common Points in the Base and the Input image are ordered as follows. The lowest common point in both the images is considered to be the origin of an X –Y co-ordinate system. All the other points that are above this point are ordered with respect to their Y values (lower the Y value, lower the order, so the origin point is order 0, the next is order 1 and so on), and when two points happen to have the same Y value, the point with the lower X value is given the lower order.

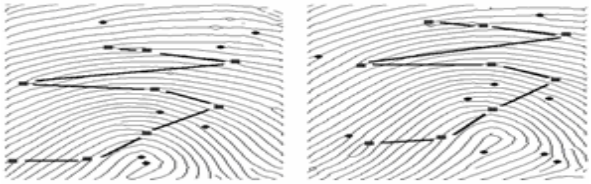

Fig 8: Tree connects candidate common points [squares] & circles are spurious minutiae points.

Effectively the order increases bottom up in the image. After ordering all the Candidate Common Points, they are connected from bottom up with respect to their order in both the images, as shown in figure 9. Then, the algorithm does the following to eliminate spurious minutiae points by doing the following: In both the tree structures, if we consider the Candidate Minutiae Points to be Vertices and the lines connecting the vertices as edges, then the algorithm encodes the following information in each of the edges: Each edge carries information about (1) the quadrant to which it is moving, (2) ratio of its own length with the previous edge length calculated using {1}, hence this information is a number value greater than 1, (3) Angle created by the edge along with the extension of the previous edge. Dark lines in figure 9 represent edges flowing through the Candidate Common Points, while semi dark lines represent the X – Y coordinate system with the starting vertex of the edge or particular minutiae as the origin, and dark dotted lines indicate extension of the previous edge. The edge from B to C would contain the below mentioned information: (a) Quadrant to which it is moving, which in this case is 1st quadrant, and this is calculated by projecting an X – Y Co-ordinate system from the starting vertex of the edge, which is B here. The edge BC obviously moves to the first quadrant. (b) Ratio of edge lengths AB and BC is calculated based on {1}. (c) The minimum angle that is created by the edge BC with the extension line of the previous edge is noted. Similarly, these three information is calculated for each of the edges that connect the Candidate Common Points in both the images. If there were 'n' Candidate Common Points, there would be (n-1) edges. These edges (in the two trees) of the base image and the input image are compared and the target vertex of the edge is removed from the Candidate Common Point List if the edges don't match.

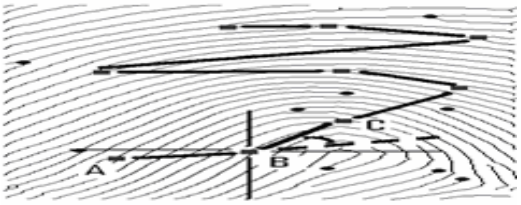

Fig 9: Drawing edges between the common minutiae points.

For instance, in the above example, if edge BC does not match with its corresponding edge in the other image, the vertex C is removed from Candidate Common Point Lists of both the images. An example illustration is given below: When the two images in figure 10 are being matched, the following will happen. When edge 'i – i+1' is being compared in both the images, the information that both the 'i - i+1' edges possess in the respective images are completely different. The edge 'i-i+1' in the left image proceeds in the first quadrant, whereas the edge 'i – i+1' in the right image proceeds in the second quadrant. Moreover the ratios of "i – (i +1): i – (i -1)" in both the images are different, because, the distance "i – i +1" is visibly large in the left image, which would cause the ratios to change.

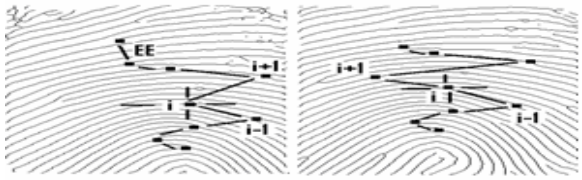

Fig 10: Trees in both images have same no. of nodes though some may not match.

What happens now is that, the minutiae point "i + 1" is removed in one of the images and the entire procedure is tested, and if the problem occurs again, minutiae "i+1" is removed in the other image and the procedure is tested. In the above case, when point "i + 1" is removed in the left image, the edges don't match again, so "i + 1" is retained in the left image and the removed from the right image and the procedure is tested. The edges will now match in this case. In the third case that the edges don't match after trying in the second image also, the algorithm removes "i + 1" from both the images and proceeds. Observe carefully figure 10 to note that, when edge "i – i +1" is removed in the right image, the number of edges in the right edge will be one less than the number of edges in the left image. But this problem is taken care of by the algorithm because, all the algorithm has to do is to find the extra edge in the left image and then remove it. In this case, the edge marked "EE" in the left image is the extra edge, which is quite evident from seeing the picture. While the algorithm runs in the matching phase, this extra edge is removed since it finds no matching edge in the other image. This step is repeated until a stage comes where no edges can further be removed. Now the Candidate Common Point set is updated. At this juncture, the following procedure is followed. Note that there might be Common Minutiae Points that were not identified by phase 1. This could have happened if there was a minutiae point "i" in both the images such that, their M(i) – tuples did not match because of large number of spurious minutiae in their vicinity in both the images. In this case, though the point "i" is actually present in both the images, it is not identified because the M(i) – tuple possesses local information only. So in the matching stage, when no edges could further be removed, the algorithm tries one last time to find Common Minutiae Points to add them to the tree. The

procedure follows as thus: for each minutiae in the set N'(BM), the 5 nearest points that feature in the current updated Candidate Common Points are found, and the M(i) – tuples are created. Similarly, for each minutiae in the N'(IM) set, M(i) – tuples are created. These tuples are matched to find any new Candidate Common Point. If any such points are found, they are added to the tree and the matching procedure by comparing edges and removing spurious points is once again followed till no edges can further be removed. Now, essentially what the algorithm has resulted in is a tree whose vertices feature in the Confirmed Common Points List, which in turn indicates the common minutiae points in both the images. If C (N) is the number of points in the Confirmed Common Points List and N is the Maximum {Number of points in the [base, input] images}, then C (N) >= (N/2). If this is true, then the two images are said to be the same, else a negative score is displayed.

3. Evaluations and Results

We conducted tests on Databases 1 and 2 from the FVC 2000 datasets. Database 1 contains images obtained from a low cost optical scanner while Database 2 contains images obtained from a low cost capacitive scanner. Three tests were performed on the databases. (1) To find the algorithm's efficiency when base and the input images are obtained from the same sensor. (2) To find the efficiency when the base and input images originated from different sensors. The above two tests are based on Genuine Accept Rates and False Reject Rates. (3) To find the False Accept Rate. **Case 1:** Database 1 contains 8 images [8 instances of the same finger] each of 10 different persons. So there are totally 80 images. For each person, 56 test cases were formulated. That is, out of the 8 instances, one instance would be set as the base image and would be tested against the remaining 7 images, which were considered to be input images. So there would be 7 test cases for one instance of one user. Hence, there would be 8 * 7 = 56 test cases for one person. And there would be 10 * 56 = 560 test cases in total for database 1. We express the efficiency of the algorithm in the following way: Efficiency in % = Number of Positive Matches / Total Number of Test cases. We got the following results for Database 1: **Efficiency = 539 / 560 = 96.25 %.** Hence the **Genuine Accept Rate = 96.25 %** and the **False Reject Rate = 3.75 %.** Similarly there are 560 test cases for database 2. **Efficiency** in this case = 541 / 560 = **96.61%. Genuine Accept Rate = 96.61%**, **False Reject Rate = 3.39%.** The cumulative results can be given as the following: **Efficiency = (539) + (541) / (560) + (560). Efficiency = 1080 / 1120 = 96.43 %.** So the overall **Genuine Accept Rate = 96.43% and the overall False Reject Rate = 3.57%. Case 2:** In both databases cumulatively, for one person, there are 16 instances of the same finger, 8 of which are taken in a low cost optical scanner and the other eight are taken in a low cost capacitive scanner. Hence, each of the 8 instances in Database 1 is kept as the base image once and tested 8 times with the other 8 instances of the same finger in Database 2. This results in 8 * 8 = 64 test cases. Similarly, the vice versa is done, meaning that each of the 8 instances in Database 2 is kept as the base image once and tested 8 times with the other 8 instances of the same finger in Database 1. These further result in another 64 test cases. So for every person, there are 64 + 64 = 128 possible test cases. Hence there are 10 * 28 = 1280 test cases. **Efficiency = 1127 / 1280 = 88.05 %.** Hence the **Genuine Accept Rate = 88.05 %** and the **False Reject Rate = 11.95%.** In general, there were about 20 out of the 560 images that yielded very less number of detectable minutiae points in the first place, because of poor image quality. **Case 3:** There are 160 images totally in Databases 1 and 2 (80 each). Out of these 160 images, there are 10 different fingerprints (each fingerprint has 16 instances, 8 each from sensor 1 and 2). To find the False Accept Rate, we did the following: We randomly considered one image, and also it's 15 other corresponding images (since they would match with the considered image), and tested that one image against the remaining 144 images. 10 such random images were considered leading to 10 * 144 = 1440 test cases. False Accept rate = Number of Positive Matches / Total test cases. We obtained 5 positive matches, hence **F.A.R = 5 / 1440 = 0.35%.** The following graph maps the genuine accept rate with false accept rate at four different instances. Remember that the base and the input images are the same if and only if C (N) >= (N/2) where C (N) is the number of common points inn the both the images and N is the Maximum {Number of points in the [base, input] images}. The five instances in the graph represent when C (N) >= (N/2), C (N) >= (N/1.8), C(N) >= (N/1.6), and C (N) >= (N/1.4), C (N) >= (N/1.2). In case 1, we obtained highest G.A.R of 96.43% with a F.A.R of 0.35%. This was obtained when C(N)>= (N/2). Similarly in case 2, the highest obtained G.A.R was 88.05% with the same F.A.R of 0.35%. This result was also obtained when C(N)>=(N/2).

5. Conclusion and future work

We have proposed a new fingerprint matching algorithm based on ratios of relational distances. We

extended the same concept to make the algorithm sensor – independent and also obtained satisfactory results. The proposed algorithm is able to generate a matching score when it obtains a minimum of (N/2) + 1 common minutiae points between the base and the input image, where N is the maximum of Number of detected minutiae points in either the base or the input image. FNMR does not increase considerably since there is no threshold on no. of minutaie that needs to be matched. We hope to make the algorithm more efficient by incorporating more classifiers in addition to the existing matching criterion which would aid in producing enhanced results.

6. References

[1] Fingerprint Verification Competition, August 2000, organized by *Biometric Systems Lab* (University of Bologna), *U.S National Biometric Test Center* (SJSU), *and Pattern Matching and Image Processing Laboratory* (MSU).

[2] A.Ross, A.K.Jain, "Biometric Sensor Interoperability – A case study in Fingerprints," *International ECCV workshop on Biometric Authentication (BioAW), Prague, Czech Republic*, LNCS vol. 3087, pp. 134-145, May 2004.

[3] A.K.Jain, S.Prabhakar, L.Hong, S.Pankanti, "Filterbank based Fingerprint Matching," *IEEE Transactions on Image Processing*, vol. 9, no. 5, May 2000.

[4] A.K.Jain, A.Ross, S.Prabhakar, "Fingerprint Matching using Minutiae and Texture Features," *International Conference on Image Conference ,* vol. 3, 282 – 285, 2001.

[5] S.Gold and A.Rangarajan, "A graduated assignment algorithm for graph matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*., vol. 18, no.4, pp. 377 – 388, 1996.

[6] A.K.Hrechak and J.A.Mchugh, "Automated Fingerprint Recognition using Structural Matching," *Pattern Recognition*, vol. 23, pp. 893-904, 1990.

[7] M.Eshera and K.S.Fu, "A Similarity Measure between Attributed Relational Graphs for Image Analysis," in *Proceedings of the 7th International Conference on Pattern Recognition*, Montreal, P.Q., Canada, July 30 – Aug 3 1984.

[8] N.K Ratha, V.D Pundit, R.M Bolle, V.Vaish, "Robust Fingerprint Authentication Using Local Structural Similarity," *Workshop on Applications of Computer Vision*, 29 – 34, '00.

[9] A.Ranade & A.Rosenfeld, "Point Pattern Matching by Relaxation," Pattern Recog., vol, no:12,2; 269-275, 1993.

[10] J.Ton and A.K.Jain, "Registering landsat images by point matching," IEEE Transactions on GeoSci. Remote Sensing, vol.27, pp. 642-651, May 1989.

[11] A.K.Jain, L.Hong, S.Pankanti and R.Bolle, "An identity authentication system using fingerprints," Proc. IEEE, vol.85, pp. 1365-1388, Sept.1997.

[12] W.Ölz and W.G.Kropatsch, "Graph Representation of Fingerprint Topology," *Computer Vision - CVWW'04, Slovenien Pattern Recognition Society,* pp. 51–58, 2004.

[13] G.T.Candela, P.J.Grother, C.I.Watson, R.A.Wilkinson and C.L.Wilson, "PCASYS: A Pattern-level classification automation system for fingerprints," *NIST Tech.Rep.NISTIR 5647*, Aug.1995.

[14] A.K.Jain, S.Prabhakar and L.Hong, "A Multichannel approach to fingerprint classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.21, no.4, pp. 348-359, 1999.

[15] L.Hong and A.K.Jain, "Classification of fingerprint images," in *11th Scandinavian Conference on Image Analysis*, Kangerlussuaq, Greenland, June 7 – 11, 1999.

[16] S.Chikkerur, A. N. Cartwright and V. Govindaraju, "Kplet and CBFS: A Graph based Fingerprint Representation and Matching Algorithm," *ICB 2006, Hong Kong.*

[17] R.O.Duda and P.E.Hart, *Pattern Classification and Scene Analysis*, New York: Wiley, 1973.

[18] A.K.Jain, S.Prabhakar and S.Chen, "Combining multiple matchers for a high security fingerprint verification system," *Pattern Rcognit.Lett*., vol.20, no.11-13, pp.1371-1379, 1999.

[19] A.Sibbald, "Method and apparatus for fingerprint characterization and recognition using auto-correlation pattern," *U.S.Patent 5633947*, 1994.

[20] E.C.Driscoll, C.O.Martin, K.Ruby, J.J.Russel and J.G.Watson, "Method and apparatus for verifying identity using image correlation" *U.S.Patent 5067162*, 1991.

[21] K.Nandakumar and A.K.Jain, "Local Correlation-based fingerprint matching," *ICVGIP, Kolkata,* December 2004.

[22] A. M. Bazen, G. T. B. Verwaaijen, S. H. Gerez, L. P. J. Veelenturf, and B. J. van der Zwaag, "A correlation-based fingerprint verification system," *Proc. Workshop on Circuits System and Signal Processing, ProRISC*, 205--213, 2000.

[23] Sharat Chikkerur, Alexander N. Cartwright, and Venu Govindaraju, "Fingerprint Image Enhancement Using STFT Analysis". *ICAPR 2005, UK.*