BINARY TREE BASED LINEAR TIME FINGERPRINT MATCHING

Mayur D Jain[1], Nalin Pradeep S[2], Prakash C[2] and Balasubramanian Raman[3]

[1]Microsoft (R&D) India Private Ltd., Hyderabad, India,
[2]Sarnoff Innovative Technologies, Vision Group, Bangalore, India,
[3]Indian Institute of Technology, Dept of Mathematics, Roorkee, India,
[1]E-mail: mayurj@microsoft.com

ABSTRACT

Fingerprint matching algorithm is a key step in fingerprint recognition system. Though there are many existing matching algorithms, there has been inability to match fingerprints in linear time. In this paper we present a novel biometric approach to match fingerprints that run in linear time. We match the minutiae in the fingerprint by constructing a Nearest Neighbor Vector (NNV) considering its k-nearest neighbors. The consolidation of these matched minutiae points is done by incorporating them in binary tree that propagates simultaneously in both fingerprints. This helps our algorithm to run in $O(n)$ time in contrast to many existing algorithms when reference core point is available. We analyze the resulting improvement in computational complexity and present experimental evaluation over FVC2002 database.

Index Terms— *Fingerprint Identification, Binary Tree, Pattern Matching*

1. INTRODUCTION

Although a large number of fingerprint recognition algorithms have been proposed to date based on image correlation, texture and filterbanks [1], minutiae based algorithms are most widely used. This is because it is believed that the minutiae in fingerprints are most discriminating and reliable features [2,3]. Most minutiae based algorithms do not rely on pre-alignment of fingerprints and carries out one-to-many matching process which runs with time complexity of $O(n^2)$ (often $O(n^3)$), where n is the number of minutiae. This complexity of the matcher can be reduced if we were to pre-align the two fingerprints in some manner so that the problem reduces to $O(n)$. Global features of the fingerprints such as core and delta points are landmark points whose location is consistent across different impressions of the same user. Therefore, their positions can be used as a point of reference in order to align the prints.

In our proposed approach, we construct binary tree with core point as root. As any node in a binary tree can have only two children according to [4], we consider only two nearest minutiae points from a node. These two nearest minutiae points are added as children of a node if their respective NNV matches; else a dummy node is added. The binary tree is constructed by propagating through each minutiae point simultaneously in both fingerprints. This is very similar to the way a human expert matches fingerprints where each successive minutiae is considered for matching in the immediate neighborhood of previously matched minutiae. This fine matching strategy makes the algorithm robust to handle distortions in the fingerprint and also for fingerprints subjected to rotation and translation. Moreover, our algorithm runs in $O(n)$ time when reliable core point is available, where n is the number of minutiae.

The paper is organized as follows: Section 2 explains core point extraction. In Section 3, matching procedure is detailed along with time complexity analysis. Section 4 discusses the results and accuracy of matching. Section 5 concludes our work.

2. CORE POINT EXTRACTION

There have been several approaches proposed for the detection of singular points (core and delta). By far, the most popular method is the one proposed by Kawagoe and Tojo [5] and is based on the computation of Poincare index. Also, recently an interesting improvisation on the Poincare index was proposed by Bazen and Gerez [6]. However, these methods based on Poincare index are very sensitive to noise and lead to detection of false singularities in low quality images.

We use the most recent approach based on complex filtering proposed by Nilsson et al. [7,8]. Their techniques rely on detecting the parabolic and triangular symmetry associated with core and delta points. The algorithm not

only provides the most likely position of the singular points but also their associated direction. This information can be used to accurately align the fingerprints in constant time. Figure 1 shows the core point extracted from a fingerprint image.

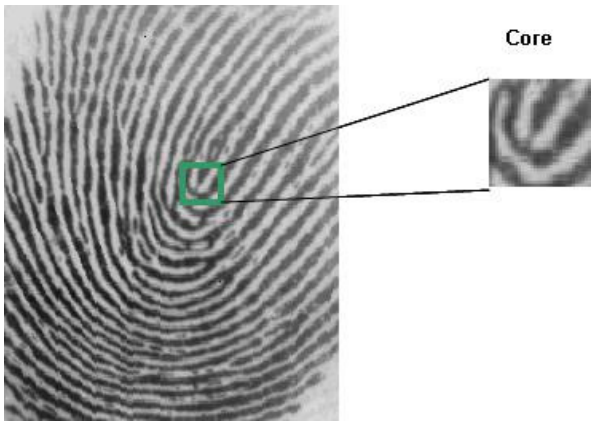

Figure 1: Core Point

3. MATCHING

We propose a matching algorithm that runs in linear time based on binary tree construction. From the reliable core point that is extracted, binary tree is constructed. The features of the algorithm can be put as (1) Representation of NNV (2) Matching NNV (3) Incorporating NNV matches into a Binary tree (4) Analysis of matching time complexity.

3.1. Nearest Neighbor Vector Representation

Each minutia in the fingerprint is represented by attributes (x, y, θ) that represents the coordinate and orientation of minutiae. To construct a NNV structure we use 'k' nearest neighbors of a minutia. As shown in Figure 2, it demonstrates an example of NNV using four nearest neighbors of a minutia. If *'a'* is the center minutia, *b, c, d* and *e* are its four neighboring minutiae points. We call the line that connects the center minutia with neighboring minutia as edge and each minutiae point as vertex. We construct NNV by extracting the following four features:

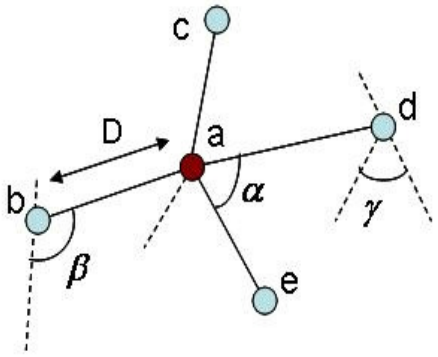

Figure 2: Nearest Neighbor Vector formation

1. The Euclidean distance between the center minutiae and neighboring minutiae, as D_i, $i = 1 \text{ to } 4$.
2. The angle difference between two neighboring edges, as $\alpha_i, i = 1 \text{ to } 4$
3. The angle difference between minutiae orientation and its respective edge, as $\beta_i, i = 1 \text{ to } 4$
4. The angle difference between center minutiae orientation and its neighboring minutiae orientation, as $\gamma_i, i = 1 \text{ to } 4$.

The NNV vector can be represented as $\left(D_i, \alpha_i, \beta_i, \gamma_i\right)$, where i = 1 to 4.

3.2. NNV Matching

Our matching algorithm is based on matching the NNV formed in local neighborhood and propagating the match by incorporating the minutiae points in binary tree structure. Let $\left(D_{1i}, \alpha_{1i}, \beta_{1i}, \gamma_{1i}\right)$ & $\left(D_{2i}, \alpha_{2i}, \beta_{2i}, \gamma_{2i}\right)$, for i = 1 to 4 be two NNVs from two different fingerprints. In order to compare the two NNVs, we calculate NNV Score through relative distance and angle differences of α, β, γ.

1. Relative distance computed as
$$Rd_i = \left|D_{1i} - D_{2i}\right| / \min\left(D_{1i}, D_{2i}\right) \quad (1)$$
min( ) means to select minimal of two values

2. The angle difference computed as
$$\Delta\alpha_i = \left|\alpha_{1i} - \alpha_{2i}\right| \quad (2)$$
$$\Delta\beta_i = \left|\beta_{1i} - \beta_{2i}\right| \quad (3)$$
$$\Delta\gamma_i = \left|\gamma_{1i} - \gamma_{2i}\right| \quad (4)$$
$$NNVScore = f_d \sum_{i=0}^{4} Rd_i + f_\alpha \sum_{i=0}^{4} \Delta\alpha_i + f_\beta \sum_{i=0}^{4} \Delta\beta_i + f_\gamma \sum_{i=0}^{4} \Delta\gamma_i \quad (5)$$

where f_d is a weighting factor for the relative distance and f_α, f_β, f_γ are weighting factors for the respective orientations. These weighting factors are a linear combination of sum to calculate the NNV score.

3.3. Incorporating NNV Match in Binary Tree

The binary tree construction occurs simultaneously in both reference and test fingerprint. In binary tree, nodes are added in each level from left to right as shown in Figure 3 and proceeds to next level only after the completion of previous level. We maintain binary tree as a list such that each node could be accessed in $O(1)$ time. Suppose k represents a node in the binary tree then it's left and right child could be accessed according to [4] as $leftnode = 2k + 1$ and $rightnode = 2k + 2$. Figure 3 shows example of binary tree that has 3 levels. The numbers in each node from $\{0,1,.....14\}$ shows the order in which a minutia gets added as a node.

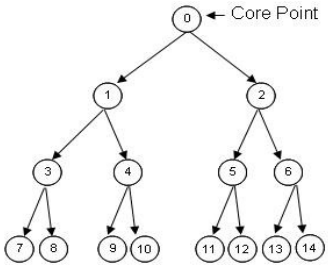

Figure 3: Example Binary Tree

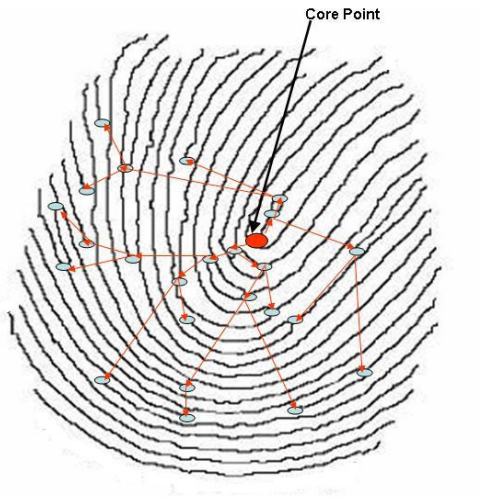

Figure 4: Binary tree representation of minutiae points in fingerprint image

Assuming that R and S are two sets of minutiae points from reference and test fingerprint image then each minutia's NNV is matched as explained in above section and gets added as a node in their respective binary tree. A dummy node gets added in the binary tree of test fingerprint when matching fails. This mainly helps in propagating the binary tree even when matching fails. In Figure 5 overview of constructing this binary tree by validating the NNV match is shown as an algorithm. Figure 4 shows the resultant binary tree formed for a fingerprint image.

1.Let R and S be set of minutiae pairs corresponding to reference and test images
2.Let C_R & C_S be core points from two fingerprints
3.Let M represent set of matched minutiae pairs <R,S>
4.Let RQ & SQ represent a FIFO queue
5.Let Bt_1 & Bt_2 represent binary tree list of two fingerprints with roots (C_R,C_S)
6.Initialize a.ENQUEUE (RQ,C_R)
b.ENQUEUE (SQ,C_S)
7.While (RQ is not empty or SQ is not empty)
a. rq=DEQUEUE(RQ)
b. sq=DEQUEUE(SQ)
c. (<rq_1,rq_2>,<sq_1,sq_2>) =Find two nearest minutiae of (rq,sq)
d. ENQUEUE (RQ,(rq_1,rq_2))
e. ENQUEUE(SQ,(sq_1,sq_2))
f. If (NNV Match(rq_1,sq_1) = true)
Add (rq_1,sq_1) as leftnode to Bt_1,Bt_2list
M= M + <rq_1,sq_1>
else
Add (rq_1,dummy) as leftnode to Bt_1,Bt_2list
g. If (NNV Match(rq_2,sq_2) = true)
Add (rq_2,sq_2) as rightnode to Bt_1,Bt_2list
M= M + <rq_2,sq_2>
else
Add (rq_2,dummy) as rightnode to Bt_1,Bt_2list
8.RETURN M (Match count obtained from M)

Figure 5: An Overview of binary tree construction algorithm

We finally consider the maximum number of matches return to compute the matching score. The score is generated using

$$score = \frac{M^2}{M_R M_S} \quad (6)$$

Here M represents the minutiae match count and M_R and M_S represent the number of minutiae in reference and test prints respectively.

3.4. Time Complexity Analysis

In order to examine the complexity of minutiae matching, we need to examine the construction of binary tree. Based

on the algorithm in Figure 5, we can see that the minutiae are handled in a queue for binary tree formation. Each queue operation like ENQUEUE or DEQUEUE takes $O(1)$ according to [4]. And at each node before adding it to the binary tree, we perform NNV match considering k neighbors whose complexity is $O(k^2) = O(1)$. We perform these operations on each minutia and they get added as matched or dummy node to the tree. Further more addition of each node in their respective binary tree list takes $O(1)$ time. So the average time complexity of consolidating NNV matches into a binary tree takes $T_M = O(1)O(k^2)O(1)O(n) = O(n)$, making the match time linear.

4. EXPERIMENTAL EVALUATION

In order to evaluate our algorithm on fingerprint matching, we considered both the accuracy (FAR and FRR) and verification time. FVC2002 DB1 was used for the evaluation. The database consists of 800 images made up of 100 individuals with 8 prints for each. The test results are shown in Table 1. The comparison with Jain's algorithm [9] is also presented in Table 2.

Table 1: Matching result of our algorithm

	False Accept Rate	False Reject Rate
1	5.0 %	0.23 %
2	4.0 %	0.64 %
3	3.0 %	0.97 %
4	2.0 %	1.38 %
5	1.0 %	1.97 %

Table 2: Matching result of Jain's algorithm

	False Accept Rate	False Reject Rate
1	5.0 %	0.78 %
2	4.0 %	1.74 %
3	3.0 %	1.82 %
4	2.0 %	3. 16%
5	1.0 %	5.66 %

From the table we can see that, our algorithm is more efficient and has a better performance. Moreover our proposed algorithm runs in $O(n)$ time thereby with very less matching time of 34.8 ms on PIII 450M Hz.

5. CONCLUSION

In this paper we proposed an algorithm that consolidates NNV matches into a binary tree after core point extraction. The algorithm runs in linear time whereas most other fingerprint algorithms have atleast $O(n^2)$ time complexity. We also experimentally evaluated the performance and efficiency of fingerprint matching. More refined pruning would be used in future to further improve the speed of the algorithm without any further compromise in accuracy.

11. REFERENCES

[1] D.Maio, D. Maltoni, A.K. Jain and S.Prabahakar, *Handbook of fingerprint Recognition*. Springer Verlag, 2003.

[2] Federal Bureau of Investigation, The Science of Fingerprints: Classification and Uses, Washington, D.C.: GPO, 1984.

[3] H.C. Lee and R.E. Gaensslen, Eds., "Advances in Fingerprint Technology", New York: Elsevier, 1991.

[4] Thomas H. Cormen, Ronald L. Rivest, Charles E. Leiserson and Clifford Stein, Introduction to Algorithms, 2nd edition , MIT Press, 2001

[5] M. Kawagoe and A. Tojo, "Fingerprint Pattern Classification", Pattern Recognition, vol. 17, no. 3, pp. 295-303, 1984.

[6] A.M. Bazen and S.H. Gerez, "Extraction of Singular Points from Directional Fields of Fingerprints", Mobile Comm. in Perspective, CTIT Workshop Mobile Comm., Univ. of Twente, Enschede, The Netherlands, pp. 41-44, Feb. 2001.

[7] K. Nilsson, "Symmetry Filters Applied to Fingerprints", PhD thesis, Chalmers University of Technology, Sweden, 2005.

[8] K. Nilsson and J. Bigun, "Localization of corresponding points in fingerprints by complex filtering", Pattern Recognition Letters, Vol 24, 2003.

[9] Anil Jain, Lin Hong, Ruud Bolle, "On-Line Fingerprint Verification", IEEE Transactions on Pattern Analysis and Machine Intelligence, v.19 n.4, p.302-314, April 1997.

[10] Gonzalez and Woods, *Digital Image Processing* 2nd edition (Prentice Hall, 2002).

[11] Second International Competition for Fingerprint Verification Algorithms (FVC2002). http://bias.csr.unibo.it/fvc2002/

[12] L. Sha, F. Zhao and X. Tang, "Fingerprint Matching Using Minutiae and Interpolation-based Square Tessellation Fingercode", ICIP, Vol 2, 41-44, 2005.

[13] T. Ohtsuka and A. Kondo, "Improvement of the fingerprint core detection using extended relation graph", Nonlinear Signal and Image Processing, May 2005.

[14] C. Jia, M. Xie and Q. Li, "A fingerprint minutiae matching approach based on vector triangle method and ridge structure", Communications, Circuits and Systems, Vol 2, 871-875, 2004.