K-plet and Coupled BFS: A Graph Based Fingerprint Representation and Matching Algorithm

Sharat Chikkerur, Alexander N. Cartwright, and Venu Govindaraju

Center for Unified Biometrics and Sensors, University at Buffalo, NY, USA
{ssc5, anc, govind}@buffalo.edu

Abstract. In this paper, we present a new fingerprint matching algorithm based on graph matching principles. We define a new representation called K-plet to encode the local neighborhood of each minutiae. We also present CBFS (Coupled BFS), a new dual graph traversal algorithm for consolidating all the local neighborhood matches and analyze its computational complexity. The proposed algorithm is robust to non-linear distortion. Ambiguities in minutiae pairings are solved by employing a dynamic programming based optimization approach. We present an experimental evaluation of the proposed approach and showed that it exceeds the performance of the NIST BOZORTH3 [3] matching algorithm.

1 Introduction

Clearly the most important stage of a fingerprint verification system is the matching process. The purpose of the matching algorithm is to compare two fingerprint images or templates and return a similarity score that corresponds to the probability of match between the two prints. Minutiae features are the most popular of all the existing representation for matching and also form the basis of the process used by human experts [7]. Each minutiae may be described by a number of attributes such as its position (x,y) its orientation θ, its quality etc. However, most algorithms consider only its position and orientation. Given a pair of fingerprints, their minutiae features may be represented as an unordered set given by

$$I_1 = \{m_1, m_2 ....m_M\} \text{ where } m_i = (x_i, y_i, \theta_i) \tag{1}$$

$$I_2 = \{m'_1, m'_2 ....m'_N\} \text{ where } m'_i = (x'_i, y'_i, \theta'_i) \tag{2}$$

Usually points in I_2 is related to points in I_1 through a geometric transformation $T()$. Therefore, the technique used by most minutiae matching algorithms is to recover the transformation function T() that maps the two point sets . While there are several well known techniques for doing this, several challenges are faced when matching the minutiae point sets. The fingerprint image is obtained by capturing the three dimensional ridge pattern on the finger on to a two-dimensional surface. Therefore apart from skew and rotation assumed under most distortion models, there is also considerable stretching. Most matching algorithms assumed the prints to be rigidly transformed(strictly rotation and displacement) between different instances and therefore perform poorly under such situations. (See Figure 1).

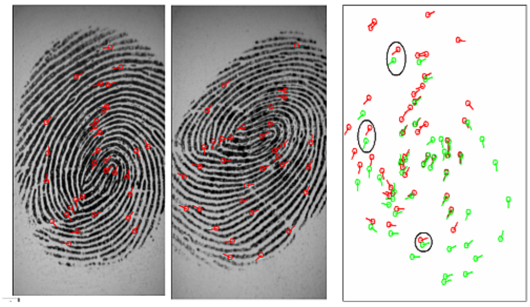


Fig. 1. An illustration of the non-linear distortion

1.1 Prior Related Work

A large number of recognition algorithms have been proposed in literature to date. The problem of matching minutiae can be treated as an instance of generalized *point pattern matching* problem. It is assumed that the two points sets are related by some geometrical relationship and the problem reduces to finding the most optimal geometrical transformation that relates these two sets. Most existing algorithms can be broadly classified as follows

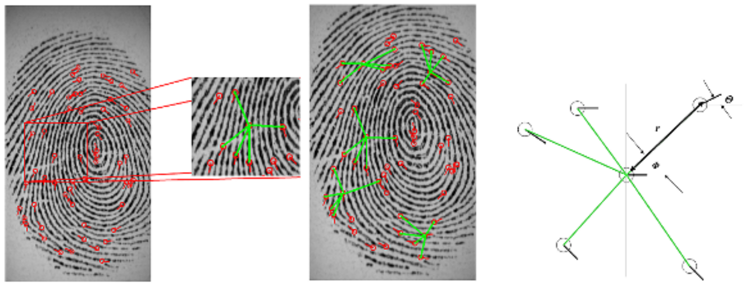
1. **Global Matching:** In this approach, the matching process tries to simultaneously align all points at once. The global matching approach can be further categorized into
   (a)*Implicit Alignment:* Here the process of finding the point correspondences and finding the optimal alignment are performed simultaneously. This includes the iterative approach proposed by Ranade and Rosenfield [8] and the generalized Hough Transform based approach of Ratha et al. [9]
   (b)Explicit Alignment In this approach, the optimal transformation is obtained after explicitly aligning one of more corresponding points. The alignment may be absolute (based on singular point such as core and delta) or relative(based on a minutiae pair). Absolute alignment approaches are not very accurate since singular point location in poor quality prints is unreliable. Jain et al [4] proposed a relative alignment approach based on alignment of ridges.
2. **Local Matching:** In local matching approaches, the fingerprint is matched by accumulating evidence from matching local neighborhood structures. Each local neighborhood is associated with structural properties that are invariant under translation and rotation. Therefore, local matching algorithms are more robust to non-linear distortion and partial overlaps when compared to global approaches. However, local neighborhood do not sufficiently capture the global structural relationships making false accepts very common. Thefore in practice, matching algorithms that rely on local neighborhood information are implemented in two stages (a) *Local structure matching*: In this step, local structures are compared to derive candidate matches for each structure in the reference print. (b) *Consoldiation*: In this step, the candidate matches are validated based on how it agrees to the global match and a score is

generated by consolidating all the valid matches. Examples of matching algorithm based on local properties can be found in Jian and Yau [6],Jea and Govindaraju [5] and Ratha et al. [10].

2 Proposed Approach: Graph Based Matching

We propose a novel graph based algorithm for robust fingerprint recognition. We define a new representation called *K-plet* to represent local neighorhood of a minutiae that is invariant under translation and rotation. The local neighborhoods are matched using a dynamic programming based algorithm. The consolidation of the local matches is done by a novel *Coupled Breadth First Search* algorithm that propagates the local matches simultaneously in both the fingerprints. In the following section, we describe our approach using the following three aspects (i)Representation, (ii)Local Matching and (iii)Consolidation.

Table 1. Left: An illustration of K-plets defined in a fingerprint, Right:Local co-ordinate system of the K-plet



2.1 Representation: K-plet

The *K-plet* consists of a central minutiae m_i and K other minutiae $\{m_1, m_2...m_K\}$ chosen from its local neighborhood. Each neigbhorhood minutiae is defined in terms of its local radial co-ordinates $(\phi_{ij}, \theta_{ij}, r_{ij})$ (See Table 1) where r_{ab} represents the Euclidean distance between minutiae m_a and m_b. θ_{ij} is the relative orientation of minutia m_j w.r.t the central minutiae m_i. ϕ_{ij} represents the direction of the edge connecting the two minutia. The angle measurement is made w.r.t the X-axis which is now aligned with the minutia direction of m_i. Unlike the *star* representation, the K-plet does not specify how the K neighbors are chosen. We outline two different approaches of doing this althought this is not meant to be an exhaustive enumeration of ways to construct the K-plet. (i)In the first approach we construct the K-plet by considering the K-nearest neighbors of each minutia. This is not very effective if the minutia are clustered since it cannot propagate matches globally. (ii) In the second approach, in order to maintain high connectivity between different parts of the fingerprint, we chose K neighboring minutia such that a nearest neighbor is chosen in each of the four quadrant sequentially. Our results are reported based on this construction.

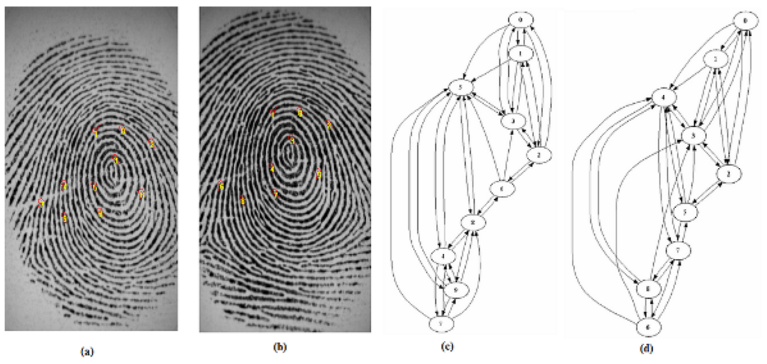


Fig. 2. Illustration of two fingerprints of the same user with marked minutiae and the corresponding adjacency graph based on the K-plet representaion. It is to be noted that the topologies of the graphs are different due to an extra unmatched minutiae in the left print.

2.2 Graphical View

We encode the local structural relationship of the K-plet formally in the form of a graph $G(V, E)$. Each minutiae is represented by a vertex v and each neighboring minutiae is represented by a directed edge (u, v) (See Figure 2). Each vertex u is colored with attributes $(x_u, y_u, \theta_u, t_u)$ that represents the co-ordinate, orientation and type of minutiae(ridge ending or bifurcation). Each directed edge (u, v) is labelled with the corresponding K-plet co-ordindates $(r_{uv}, \phi_{uv}, \theta_{uv})$

2.3 Local Matching: Dynamic Programming

Our matching algorithm is based on matching a local neighborhood and propagating the match to the K-plet of all the minutiae in this neighborhood successively. The accuracy of the algorithm therefore depends critically on how this local matching is performed. We convert the unordered neighbors of each K-plet into an ordered sequence by arranging them in the increasing order of the radial distance r_{ij}. The problem now reduces to matching two ordered sequences $S\{s_1, s_2...s_M\}$ $T\{t_1, t_2...t_N\}$. We utilize a dynamic programming approach based on string alignment algorithm [2].

Formally, the problem of string alignment can be stated as follows: Given two strings or sequences S and T, the problem is two determine two auxiliary strings S' and T' such that

1. S' is derived by inserting spaces (_) in S
2. T' is derived by inserting spaces in T
3. $length(S') = length(T')$
4. The cost $\sum_{i=1}^{|S'|} \sigma(s'_i, t'_i)$ is maximized.

For instance, the result of aligning the sequences $S = \{acbcdb\}$ and $T = \{cadbd\}$ is given by

$$S' = ac\_\_bcdb \tag{3}$$
$$T' = \_cadb\_d\_ \tag{4}$$

A trivial solution would be to list all possible sequences S' and T' and select the pair with the least/most alignment cost. However, this would require exponential time. Instead we can solve this using dynamic programming in O(MN) time as follows. We define D[i,j]$(i \in \{0, 1...M\}, j \in \{0, 1...N\})$ as the cost of aligning substrings S(1..i) and T(1..j). The cost of aligning S and T is therefore given by D[M,N]. Dynamic programming uses a recurrence relation between D[i,j] and already computed values to reduce the run-time substantially. It is assumed ofcourse that D[k,l] is optimal $\forall k < i, l < j$. Given that the previous sub-problems have been optimally defined, we can match s_i and t_j in three ways

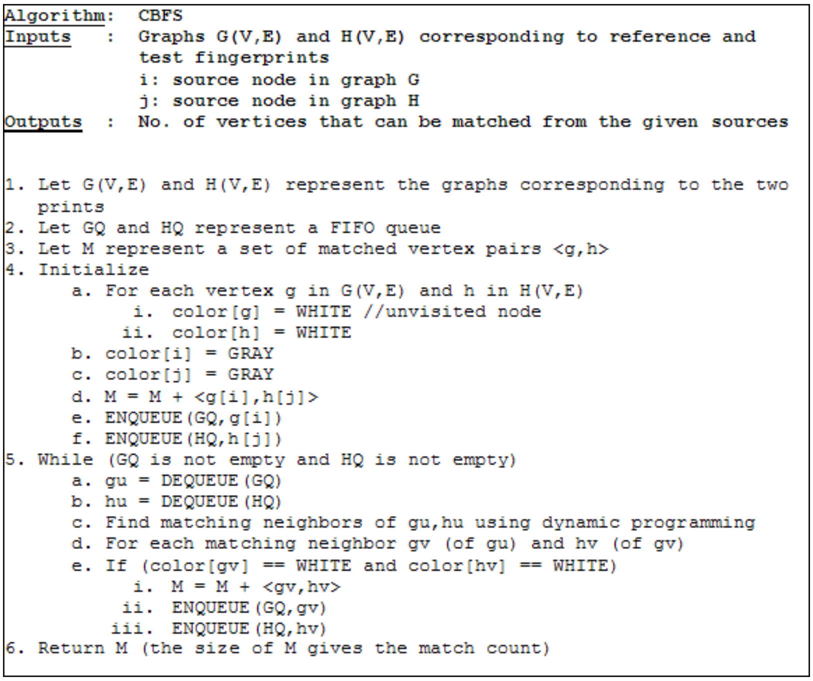
1. the elements s[i] and t[j] match with cost $\sigma(s[i], t[j])$,
2. a gap is introduced in t (s[i] is matched with a gap) with cost $\sigma(s[i], \_)$
3. a gap is introduced in s (t[j] is matched with a gap) with cost $\sigma(\_, t[j])$

Therefore, the recurrence relation to compute D[i,j] is given by

$$D[i,j] = max \left\{ \begin{array}{c} D[i-1, j-1] + \sigma(s[j], t[i]) \\ D[i-1, j] \quad + \quad \sigma(s[i], \_) \\ D[i, j-1] \quad + \quad \sigma(\_, t[j]) \end{array} \right\} \tag{5}$$

2.4 Consolidation: Coupled Breadth First Search

The most important aspect of the new matching algorithm is a formal approach for consolidating all the local matches between the two fingerprints without requiring explicit



```
Algorithm:  CBFS
Inputs   :  Graphs G(V,E) and H(V,E) corresponding to reference and
            test fingerprints
            i: source node in graph G
            j: source node in graph H
Outputs  :  No. of vertices that can be matched from the given sources


1. Let G(V,E) and H(V,E) represent the graphs corresponding to the two
   prints
2. Let GQ and HQ represent a FIFO queue
3. Let M represent a set of matched vertex pairs <g,h>
4. Initialize
      a. For each vertex g in G(V,E) and h in H(V,E)
            i.  color[g] = WHITE //unvisited node
           ii.  color[h] = WHITE
      b. color[i] = GRAY
      c. color[j] = GRAY
      d. M = M + <g[i],h[j]>
      e. ENQUEUE(GQ,g[i])
      f. ENQUEUE(HQ,h[j])
5. While (GQ is not empty and HQ is not empty)
      a. gu = DEQUEUE(GQ)
      b. hu = DEQUEUE(HQ)
      c. Find matching neighbors of gu,hu using dynamic programming
      d. For each matching neighbor gv (of gu) and hv (of gv)
      e. If (color[gv] == WHITE and color[hv] == WHITE)
            i.  M = M + <gv,hv>
           ii.  ENQUEUE(GQ,gv)
          iii.  ENQUEUE(HQ,hv)
6. Return M (the size of M gives the match count)
```

Fig. 3. An overview of the CBFS algorithm

alignment. We propose a new algorithm called *Coupled BFS algorithm*(CBFS) for this purpose. CBFS is a modification of the regular breadth first algorithm [2] except for two special modifications. (i) The graph traversal occurs in two directed graphs G and H corresponding to reference and test fingerprints simultaneously. (The graphs are constructured as mentioned in Section 2.2) (ii) While the regular BFS algorithm visits each vertex v in the adjacency list of , CBFS visits only the the vertices $v_G \in V$ and $v_H \in H$ such that v_G and v_H are locally matched vertices. The overview of the CBFS algorithm is given in Figure 3

2.5 Matching Algorithm

It is to be noted that the CBFS algorithm requires us to specify two vertices as the source nodes from which to begin the traversal. Since the point correspondences are not known apriori, we execute the CBFS algorithm for all possible correspondence pairs $g[i], h[j])$. We finally consider the maximum number of matches return to compute the matching score. The score is generated by using [1] $s = \frac{m^2}{M_R M_T}$. Here m represents the number of matched minutiae and M_R and M_T represent the number of minutiae in the reference and template prints respectively.

3 Experimental Evaluation

In order to measure the objective performance, we run the matching algorithm on images from FVC2002 DB1 database. The database consists of 800 images (100 distinct fingers, 8 instances each). In order to obtain the performance characteristics such as EER (Equal Error Rate) we perform a total of 2800 genuine comparision and 4950 impostor comparisons .We present the comparative results in Table 2. The improvement in the ROC characteristic can be seen from Figure 4.

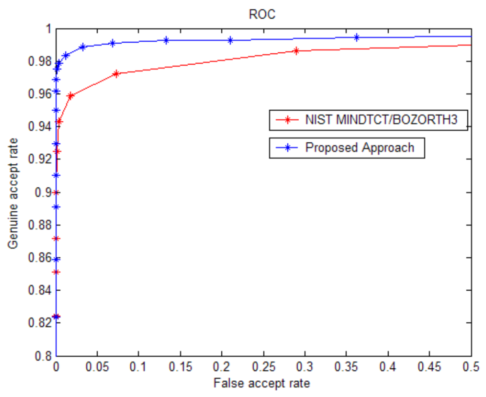


Fig. 4. A comparision of ROC curves for FVC2002 DB1 database

Table 2. A summary of the comparative results

Database	NIST MINDTCT/BOZORTH3		Proposed Approach	
	EER	FMR100	EER	FMR100
FVC2002 DB1	3.6%	5.0%	1.5%	1.65%

4 Summary

We presented a novel minutia based fingerprint recognition algorithm that incorporates three new ideas. Firstly, we defined a new representation called K-plet to encode the local neighborhood of each minutia. Secondly, we also presented a dynamic programming approach for matching each local neighborhood in an optimal fashion. Lastly, we proposed CBFS (Coupled Breadth First Search), a new dual graph traversal algorithm for consolidating all the local neighborhood matches and analyze its computational complexity. We presented an experimental evaluation of the proposed approach and showed that it exceeds the performance of the popular NIST BOZORTH3 matching algorithm.

References

1. Asker M. Bazen and Sabih H. Gerez. Fingerprint matching by thin-plate spline modeling of elastic deformations. *Pattern Recognition*, 36:1859–1867, 2003.
2. Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to algorithms*. McGraw-Hill Book Company, 1998.
3. M. D. Garris, C. I. Watson, R. M. McCabe, and C. L. Wilson. Users guide to nist fingerprint image software (nfis). Technical Report NISTIR 6813, National Institute of Standards and Technology, 2002.
4. A. Jain, L. Hong, and R. Bolle. On-line fingerprint verification. In *Pattern Analysis and Machine Intelligence*, volume 19, pages 302–313, 1997.
5. Tsai-Yang Jea and Venu Govindaraju. A minutia-based partial fingerprint recognition system. *Submitted to Pattern Recognition*, 2004.
6. Xudong Jiang and Wei-Yun Yau. Fingerprint minutiae matching based on the local and global structures. In *International Conference on Pattern Recognition*, pages 1038–1041, 2000.
7. D. Maio, D. Maltoni, A. K. Jain, and S. Prabhakar. *Handbook of Fingerprint Recognition*. Springer Verlag, 2003.
8. A. Ranade and A. Rosenfeld. Point pattern matching by relaxation. *Pattern Recognition*, 12(2):269–275, 1993.
9. N. K. Ratha, K. Karu, S. Chen, and A. K. Jain. A real-time matching system for large fingerprint databases. *Transactions on Pattern Analysis and Machine Intelligence*, 18(8):799–813, 1996.
10. N. K. Ratha, V. D. Pandit, R. M. Bolle, and V. Vaish. Robust fingerprint authentication using local structure similarity. In *Workshop on applications of Computer Vision*, pages 29–34, 2000.