PERGAMON

Neural Networks 14 (2001) 1293–1305

Neural Networks

www.elsevier.com/locate/neunet

Contributed article

Fingerprints classification using artificial neural networks: a combined structural and statistical approach

Khaled Ahmed Nagaty

Faculty of Computer Sciences and Information Systems, Ain-Shams University, Cairo, Egypt

Received 9 February 1999; accepted 30 March 2001

Abstract

This paper describes a fingerprint classification algorithm using Artificial Neural Networks (ANN). Fingerprints are classified into six categories: arches, tented arches, left loops, right loops, whorls and twin loops. The algorithm extracts a string of symbols using the block directional image of a fingerprint, which represents the set of structural features for this image. The moment representing the statistical feature of the pattern is computed for this string and its Euclidean Distance Measures (EDM) are computed by using this moment. Our discrimination system uses a multilayer artificial neural network composed of six subnetworks one for each class. The classifier was tested on 1500 images of good quality in the Egyptian Fingerprints database; images with poor quality were rejected. In the six-class problem the network achieved 95% classification accuracy. In the five-class problem when we place whorls and twin loops together in the same category the classification accuracy was around 99%. In the four-class problem when we place arches and tented arches in the same class the classification accuracy was 99%. © 2001 Elsevier Science Ltd. All rights reserved.

Keywords: Fingerprints classification; Characteristic strings; Moments; Euclidean Distance Measures (EDM); Multilayer ANN; Subnetworks; Block directional image

1. Introduction

Fingerprints are imprints or impressions of patterns formed by friction ridges of the skin in fingers and thumbs. Fingerprints are an effective means of identifying individuals for long times, because the fingerprints of an individual are unique (Leung, Leung, Lau & Luk, 1991).

Presently, several organizations are using fingerprints for not only criminal investigation but also to access control of restricted areas, border control, driving licenses, liquor licenses, employee identification, financial security and verification of firearm purchasers, passports and visas. Its criminal applications include verifying the alleged identity of suspects, identifying known criminals, persons considered as potential criminals and detained suspects. Fingerprints are one of the most popular biometrics techniques in both of verification and identification modes (Hong & Jain, 1998). In the verification mode fingerprints authenticates an individual identity by comparing one-to-one comparison to determine whether the identity claimed by the individual is true or not. In the identification mode fingerprints recognizes an individual by searching the entire template database for a match. It conducts one-to-many comparison to establish the identity of the individual. The goal of fingerprint classification is to assign a fingerprint a specific class according to its geometric properties. At the coarse-level, Henry classifies fingerprint patterns into three major classes: loops, arches and whorls (Federal Bureau of Investigation, 1984). Each class is subdivided into subclasses according to specific characteristics in the class. Arches can be divided into arches and tented arches, loops can be divided into right loops and left loops while whorls can be divided into whorls and twin loops, as shown in Fig. 1. Tented arches can be divided into first type and second type tented arches, see Fig. 2. In this paper we will consider first type tented arches to represent this class, because it is difficult to discriminate between the two types based on the block directional image only. Fingerprints classification reduces the number of images to be matched during fingerprint matching, because only fingerprints with the same class should be matched. Thus fingerprint classification can determine at the coarse-level if two fingerprints can match or not, it is based on ridge patterns orientations and the singular points of fingerprints (core and deltas). If these properties can be described quantitatively and extracted automatically from a fingerprint then the fingerprint classification will become an easy task. Bad quality images decrease the classification accuracy of a classifier. These images contain a high ratio of noise, which makes them difficult to classify

E-mail address: khaled_nagaty@hotmail.com (K.A. Nagaty).

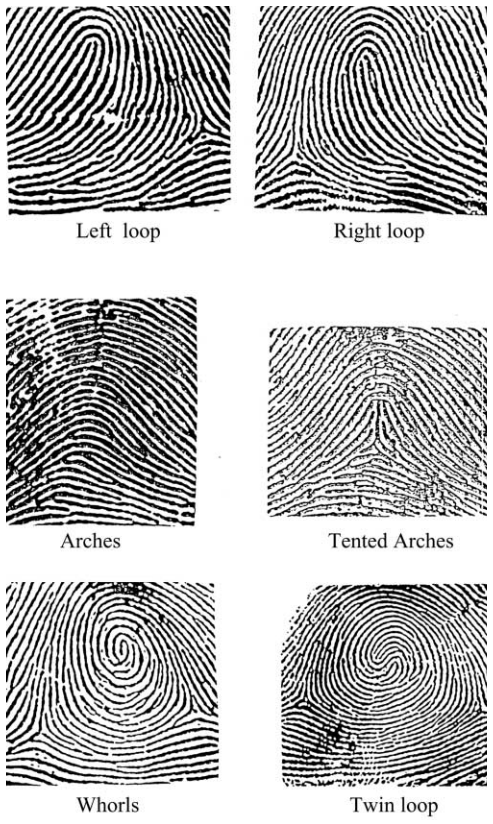


Fig. 1. Six classes of fingerprints.

even for a fingerprint expert. Singular points of a fingerprint can be smudged due to over inking or no inking during fingerprint acquisition, this causes classifiers, which depend on singular points to fail if a misdetection of these points occurs. Singular points classifiers based on singular points extraction and perform classification based on the number and location of the detected singular points. A fingerprint with a core and right delta is classified left loop, if the delta is on the left side it is classified as right loop. If there is a core and two deltas it is classified as whorl and if there are no deltas it is classified as an arch. Ridges in processing regions, which have a high ratio of noise, are smudged and no direction is assigned to them, this causes classifiers, which depend on global orientation of fingerprint ridges to fail. It is better in our algorithm to reject these images, to make such a decision a threshold based on the entire image's average density is

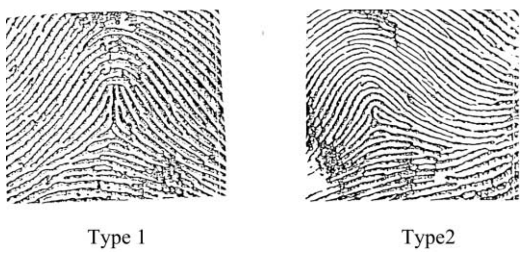


Fig. 2. The two types of tented arches.

used, where images below this threshold are rejected. The average density of the entire image is computed by averaging the pixel density of all the processing blocks during the computation of the block directional image.

Many researchers have addressed the fingerprint classification problem. Tou and Hankley (1968) proposed a syntactic approach to describe fingerprint impression, which concentrates on the topological representation of the pattern. Moayer and Fu (1975a, 1975b), suggest a syntactic method based on the ridge flow in the pattern area and the presence of deltas in the print, however misclassification occurs if the deltas are not detected exactly.

Kameswara and Black (1980) adopt a syntactic approach based on a string of symbols to classify a fingerprint into one of 10 defined types. Kawagoe and Tojo (1984) describe a method to classify fingerprints according to the shape of the major flow lines obtained by tracing the flow of the direction pattern of the fingerprint. Kamijo, Hiroshi and Kojima (1992) use multilayers artificial network composed of five subnetworks to carry out macrofeature extraction on the fingerprint image. Mehtre (1993) uses the core and delta information for classification of fingerprints and for registration while matching. He identifies fingerprints by matching the minutiae features of a given fingerprint against those stored in the database. Blue, Candela, Grother, Chellappa and Wilson (1994) made a comparison between conventional statistical classifiers and neural networks classifiers for fingerprint applications. Karu and Jain (1996) use the number and location of the detected singular points (cores and deltas) in a fingerprint image to perform fingerprint classification. Neto and Borges (1997) use discrete wavelet transform as feature extraction technique and adopt artificial neural networks as decision stage to perform fingerprint classification. Lumini, Maio and Maltoni (1997) extract numerical vectors from the direction images of the fingerprints. The classification is thus performed in a multidimensional space by using similarity measures.

Senior (1997) describes a method of classification using hidden Markov models to recognize the ridge structure of the print. Hong, Yifei and Jain (1998) present a fast fingerprint enhancement algorithm, which can adaptively improve the clarity of ridge and valley structures of input fingerprint images based on the estimated local ridge orientation and frequency. Cappelli, Lumini and Maio (1999) introduce a new approach to automatic fingerprint classification based on the directional image partitioning into homogeneous connected regions according to the fingerprint topology, thus giving a synthetic representation which can be exploited as a basis for the classification. A set of dynamic masks, together with an optimization criterion are used to guide the partitioning.

Previous attempts to classify fingerprint by using either the structural features or the statistical features are considered incomplete, they do not give the desired accuracy and the problem is still an open question. In this paper, we are interested in coarse-level classification of fingerprint images

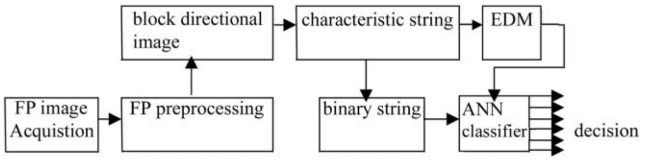


Fig. 3. Block diagram of the fingerprint classification algorithm.

by combining both the structural features and statistical measures of fingerprint patterns. After preprocessing the fingerprint image, it is normalized by adjusting its orientation and scale, then we use its block directional image to extract both the structural and statistical features required for classification.

Block directional smoothing is done for the directional image where spurious directions are smoothed.

We chose ANN as a classifier due to their ability in making non-linear distinction and their recognition of inaccurate and distorted patterns. Also it was proved that combining both the original independent variables and the predictions of a well-known statistical tool increases ANN classification performance (Markham & Ragsdale, 1995).

The algorithm for classifying the input fingerprint into six classes consists of four major steps.

1. Compute the block directional image.
2. Extract the structural features of the pattern.
3. Compute the statistical measures of the pattern.
4. Design a multilayer ANN composed of six subnetworks for the six fingerprint classes and use a multivariate input vector which is a combination of the structural features and the statistical measures.

A block diagram of this algorithm is shown in Fig. 3. Each step of the algorithm is discussed in the following sections. The algorithm was tested on images from the Egyptian Fingerprint database at the Criminal Evidence department.

In Section 2 we present an algorithm for computing a block directional image. Section 3 deals with finding the structural features of the fingerprint. Section 4 deals with the computation of the statistical measures of the pattern. Section 5 combines the structural features and the statistical measures to present them to the ANN classifier which is composed of six independent subnetworks. Section 6 presents experimental results and compares them with other classification results reported in the literature (Karu & Jain, 1996; Wilson, Candela & Watson, 1993).

2. Computation of the block directional image

The method for finding the block directional image is adopted from the paper by Mehtre (1993). The concept of block directional image divides the fingerprint image into a number of squares. To compute the local orientation of the dominant ridge in each square the following equation was used:

$$D(I,J) = \underset{\text{dir}=1}{\overset{N}{\text{Min}}} \sum_{k=1}^{L} |G(I_k, J_k) - G(I,J)| \tag{1}$$

where $D(I, J)$ is the ridge direction at pixel (I, J), $G(I, J)$ is the gray value at pixel (I, J), $G(I_k, J_k)$ is the gray value at pixel (I_k, J_k) in the direction dir, L is the number of pixels in each direction used for computation of this direction and N is the number of directions used.

Although N different directions can be evaluated owing to the continuous ridge flow over a block under consideration, we set N to four directions only where $N \in \{0, \pi/4, \pi/2, 3\pi/4\}$ which are the major local ridge orientations. The value of L changes according to the block size under consideration and the orientation. For each of the above four directions, $L = 32$ pixels for a block of size 32×32. The local orientations of the ridges in the block directional image reveals the global orientation of the ridges in the fingerprint which is sufficient to classify a fingerprint. The direction of the dominant ridge in each square can be determined and represented by directional codes as shown in Fig. 4. The directions of the ridges included in the directional image approximate the original fingerprint image and preserve the global structure of the print. Fig. 5 shows two different types of fingerprints and their corresponding block directional images with a square size of 32×32 pixels.

2.1. Block directional smoothing

After computing the directional image, we may find sampling blocks having directions which may be quite different from their neighboring directions. If these directions are used as they are, then the extracted curves will not

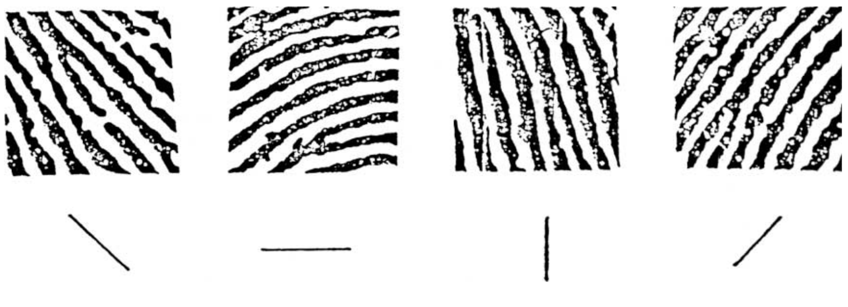


Fig. 4. Directional codes.

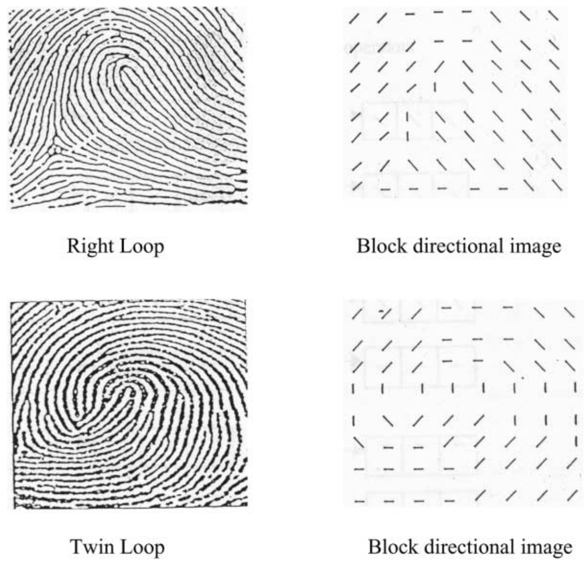


Fig. 5. Two fingerprints and their block directional image.

represent the original image, so it is required to smooth these spurious directions. We used a set of horizontal and vertical operators where each operator has three components. The direction of the middle component is changed according to the direction of its surrounding two components. The block directional image is raster scanned by using both the horizontal operators and the vertical operators in any order. This process is done iteratively, until no more corrections are done and the algorithm terminates. If the algorithm enters into a loop of corrections, i.e. a set of operators undo what the other set has done, then the number of corrections between successive iterations remains constant but greater than zero and the algorithm can terminate. This method for block smoothing corrects only the spurious directions and leaves the correct directions unsmoothed which gives better results. Fig. 6 shows the set of vertical operators and Fig. 7 shows the set of horizontal operators where / indicates a direction $<90°$, \ indicates a direction $>90°$, | for the vertical direction and $-$ for the horizontal direction. We notice that the number of vertical operators is less than the number of horizontal operators so that we can avoid smoothing the delta regions which have vertical structures.

It is possible to describe these operators formally by using the IF-THEN rule, suppose that the middle component of the operator is θ and its two surrounding components are θ_1,θ_2, respectively. Then we have the following two rules:

```
Rule1: (horizontal and vertical operators)
  If $(\theta_1 = \theta_2 \neq \theta)$ Then $(\theta_1 = \theta_2 = \theta)$
Rule 2: (horizontal operators only)
  If $(\theta_1 \neq \theta_2 \neq \theta)$ Then
    If
    $(\theta_1 = (0° \cup 90°) \cap \theta \neq (0° \cup 90°) \cap$
    $\theta_2 = (45° \cup 135°))$ Then $\theta = \theta_2$
    If
    $(\theta_1 = (45° \cup 135°) \cap \theta \neq (0° \cup 90°) \cap$
    $\theta_2 = (0° \cup 90°))$ Then $\theta = \theta_1$
```

It is important to exempt the following structures in Fig. 8 from smoothing so that we can keep the original structure of the fingerprint. Fig. 9(a), shows an original fingerprint image. Fig. 9(b), shows its block directional image obtained by using Eq. (1). Fig. 9(c)–(g) show the effects of the smoothing iterations on the block directional image in Fig. 9(b), where the images on the left side show the effect of using first the horizontal operators and then the vertical operators, while the images on the right side show the effect of using first the vertical operators and then the horizontal operators. Fig. 9(c)–(g) show the first, second, third, fourth and fifth iterations of smoothing respectively. In Fig. 9(f), the algorithm on the right side terminates when the number of its horizontal and vertical corrections is zero. In Fig. 9(g) the algorithm on the left side terminates when the number of its horizontal and vertical corrections is zero. For the sake of clarity we used rectangles to denote the directions required smoothing. Vertical rectangles denote vertical directions that require smoothing using vertical operators. Horizontal rectangles denote horizontal directions that require smoothing using horizontal operators. We shall discuss the left-side algorithm only and the same is applied for the right-side

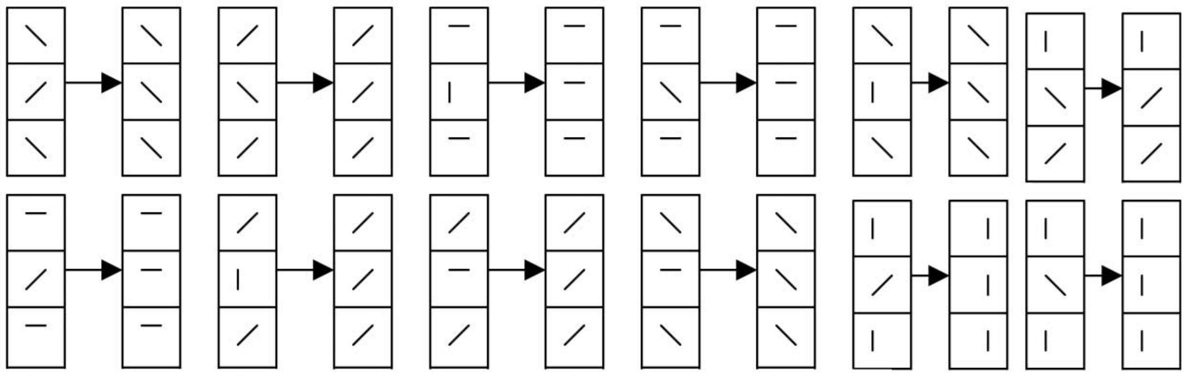


Fig. 6. The set of vertical operators.

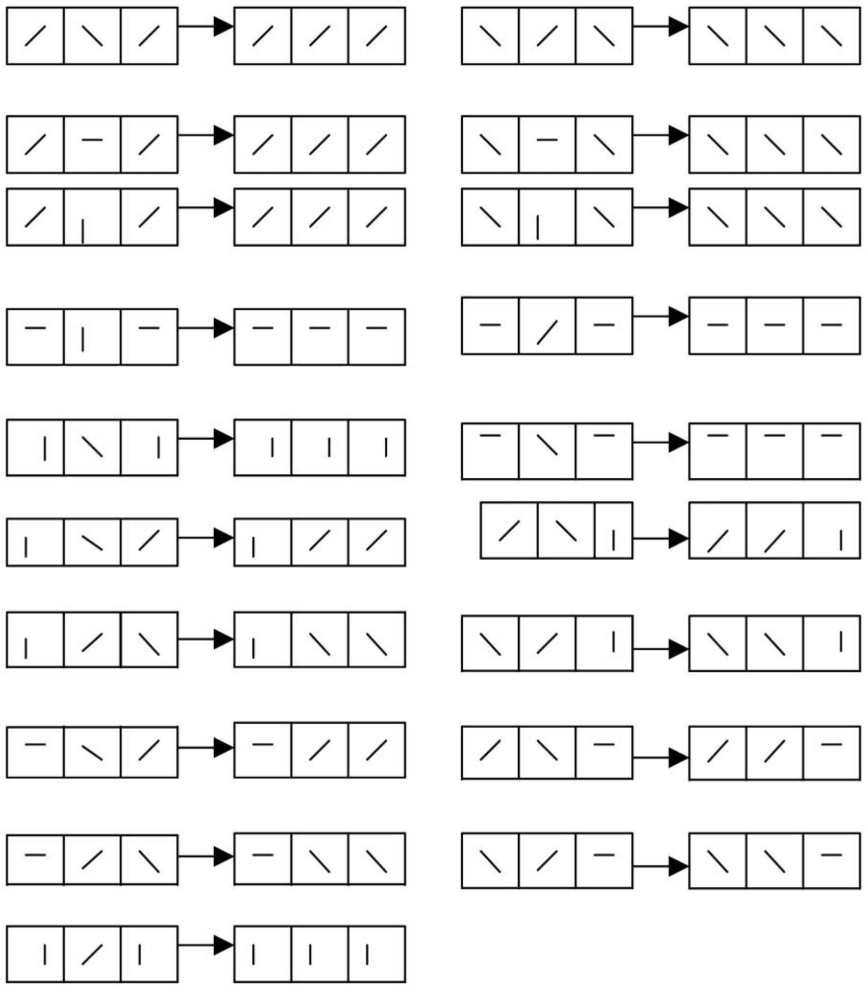


Fig. 7. The set of horizontal operators.

algorithm. In Fig. 9(c), the horizontal directions in horizontal rectangles were smoothed, the horizontal corrections are two. In Fig. 9(d), vertical directions in vertical rectangles were smoothed, the number of vertical corrections is five. The result of this vertical smoothing is three horizontal spurious directions in three horizontal rectangles. These horizontal spurious directions are smoothed in Fig. 9(e), and the number of horizontal corrections is three. In Fig. 9(f), no more vertical spurious directions that require smoothing are found,and the number of vertical corrections is zero. In Fig. 9(g), there are no horizontal directions that require smoothing are found,and the number of horizontal corrections is zero. Now the algorithm terminates at the fifth iteration. Notice that the algorithm will terminate when the number of horizontal and vertical corrections is zero. It is easy to see that the order of using the horizontal and vertical operators in smoothing does not effect the end result.

2.2. Summary

In this section we use a method for the computation of the block directional image. We present an algorithm to smooth the spurious directions in this image. The algorithm smoothes the image by applying a set of horizontal and vertical operators in any orders to perform vertical and horizontal smoothing. These operators are in effect only when it meets a region that requires smoothing, other regions which do not need

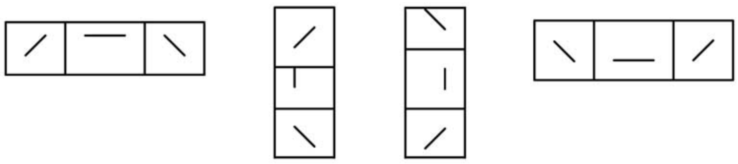


Fig. 8. Structures exempted from the smoothing process

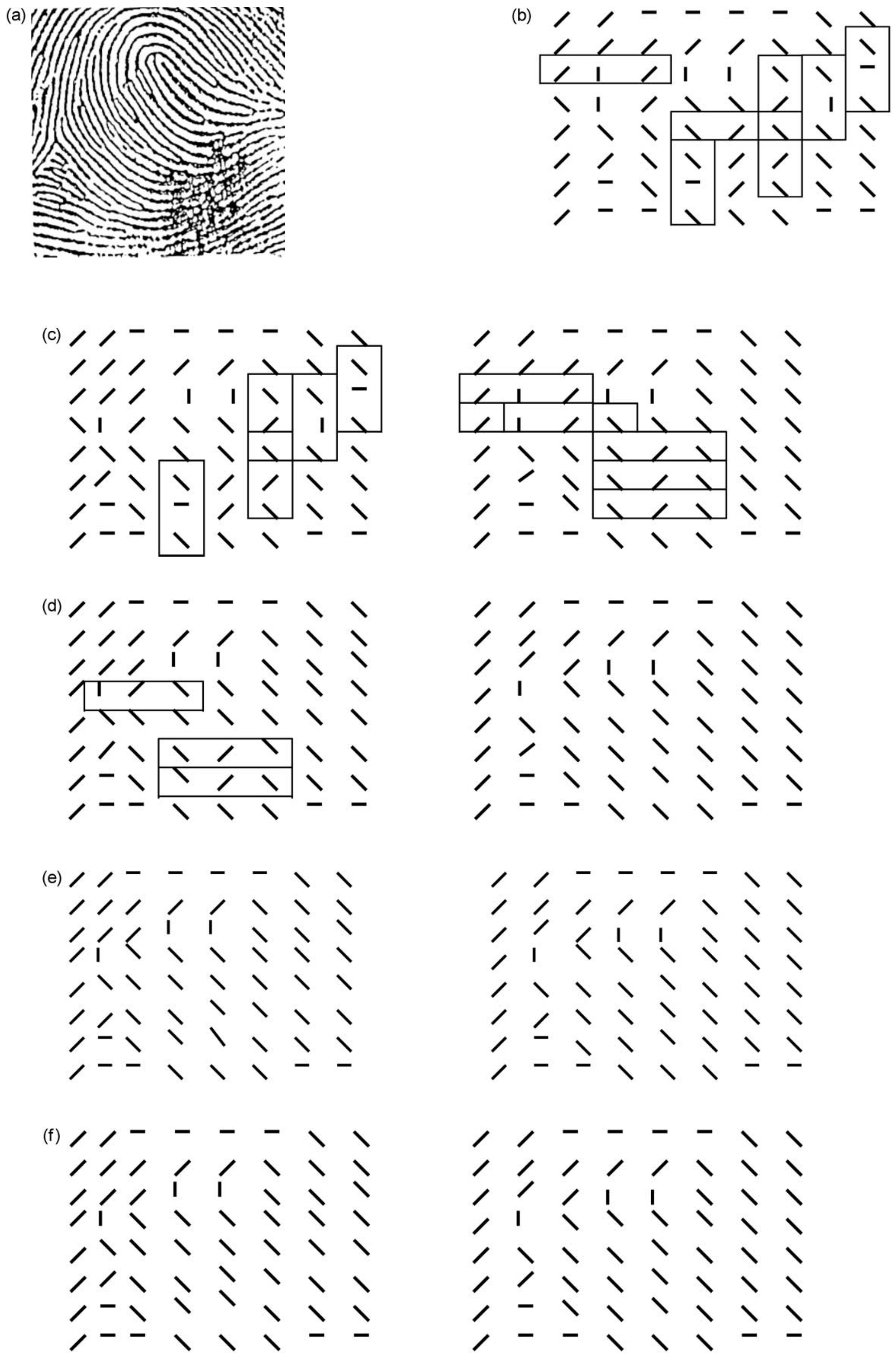


Fig. 9. (a) A fingerprint, (b) unsmoothed directional image for (a), (c) directional image for (b), (d) directional image for (c), (e) directional image for (d), (f) directional image for (e), (g) directional image for (f).

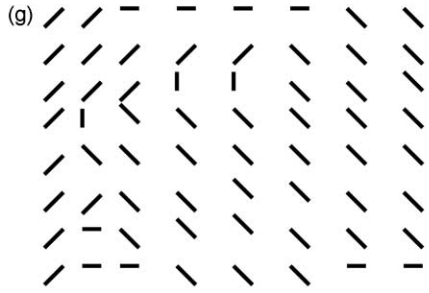


Fig. 9. (*continued*)

smoothing are kept unsmoothed. The effectiveness of this method is shown using a sample fingerprint image.

3. Structural feature extraction

The block directions of the directional image are converted into binary blocks so that the resultant curves can be traced using a line-tracing algorithm. The line-tracing algorithm can extract the curve features such as endpoints and curvatures and represent them in a symbolic form to be ready for structural matching. The binarization process takes place by mapping each direction to its corresponding binary operator. Fig. 10, shows the directional operators for the mapping process where λ_i is the direction for block i. Fig. 11, shows the binary version for the block directional image in Fig. 9(g).

It is clear in Fig. 11, that it is difficult for the line tracing algorithm to trace the ridge flow in each type of the bifurcations in Fig. 12, mainly because we do not know which path to follow.

To remove these bifurcations we have to apply the filter operators in Fig. 13 in the four symmetric directions. Fig. 14, shows the binary block directional image after removing bifurcations

3.1. Line tracing algorithm

The transformed block directional image is composed of ones, which act as a link between various curves of the pattern. The pattern is scanned from left to right, top to bottom by using a set of feature masks, as shown in Fig. 15. This constellation of feature masks represents all the possible structures the algorithm can find when tracing the binary block directional image. Each mask is assigned a symbol, and each connected curve consists of a string of symbols that represent this curve without any loss of information. We assume a column of zeros as the first column and the last column in the matrix, also we consider a row of zeros as the first row and the last rows in the matrix. All the zeros in the matrix are omitted for clarification. The algorithm starts with a search for one of the endpoint symbols H, I, K, L, M, N, O. If one is found the algorithm traces its line using the connecting operators (non-end symbols) which are: P, Q, R, S, T, U, V, W, X, Y, Z, A and store the symbols as they contain valuable information on the flow of this line. During the tracing process the algorithm deletes the line to avoid picking it up again. We must note that the algorithm keeps its original position to return to it again after finishing tracing a line to continue scanning the matrix. The algorithm stops when there are no more lines to trace. At the end, symbols on closed curves form the characteristic string for the pattern (Kameswara & Black, 1980). Generally, there is one characteristic string to represent the pattern.

For example, the strings below are corresponding to the curves in the binary block directional in Fig. 14.

(1) IM, (2) OK, (3) MPQK, 4) IM, (5) IRWK, (6) IRWK, (7)OK, (8) IRWK, (9) OK, (10) IM, (11) OK, (12) KQPRWK, (13) IM, (14) OK, (15) IM, (16) IM, (17) MPQK, (18) MPQK

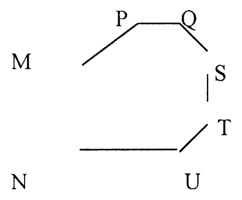
The strings representing lines OK and IM do not represent any valuable information of the interesting curves in the pattern, so we can delete these curves. MPQK and IRWK strings represent stray curves and do not have significant information. The KQPRWK string represents the right loop curve, it starts on the right side and ends on the same side.

$$\begin{array}{cccc}
\begin{matrix}0&0&0\\1&1&1\\0&0&0\end{matrix} & \begin{matrix}0&1&0\\0&1&0\\0&1&0\end{matrix} & \begin{matrix}1&0&0\\0&1&0\\0&0&1\end{matrix} & \begin{matrix}0&0&1\\0&1&0\\1&0&0\end{matrix} \\
\lambda_i = 0^\circ & \lambda_i = 90^\circ & \lambda_i > 90^\circ & \lambda_i < 90^\circ
\end{array}$$

Fig. 10. Directional binary operators.

3.2. Symbol transformation

The alphabetic symbols used to represent the curve strings must be transformed to map an input vector to an ANN. An efficient mapping is to transform the geometrical binary pattern corresponding to a given symbol to a binary string. This transformation can be done by scanning row wise the binary representation for each symbol in Fig. 15 to form a one-dimensional binary string. For example, the binary representation for the H symbol is mapped to 000010010. The characteristic strings represent the various types of fingerprint patterns can be transformed to their binary strings, first by transforming each symbol in the characteristic string to its binary representation and then concatenating these binary representations to form a one-dimensional binary string. For example, the characteristic string S for the left loop is:



S = MPQSTUN

Its one-dimensional binary representation is:
001010000000011100000110001100010010010010100-001110000000011000.

```
    1     1                         1     1
  1     1   1 1 1 1 1 1 1 1 1 1 1 1   1     1
1     1                                 1     1
    1     1     1     1     1 1     1     1
  1     1     1     1     1     1     1     1
1     1     1     1     1         1     1     1
    1     1     1   1     1   1     1     1
  1     1     1     1     1     1     1     1
1     1     1       1     1       1     1     1
    1   1   1     1     1     1     1     1
  1     1     1     1     1     1     1     1
1       1       1     1     1     1     1     1
    1 1     1     1     1     1     1     1
  1     1     1     1     1     1     1     1
1         1     1     1     1     1     1     1
    1     1 1     1     1     1     1     1
  1     1     1     1     1     1     1     1
1     1         1     1     1     1     1     1
    1       1     1     1     1     1     1
  1   1 1 1   1     1     1     1     1     1
1               1     1     1     1     1     1
    1       1     1     1     1     1     1
  1   1 1 1   1     1     1     1     1     1
1               1     1     1     1     1     1
```

Fig. 11. Binary block directional image before removing bifurcations.

3.3. Summary

In this section we use a line tracing algorithm to extract the structural features from the block directional image by using a set of feature masks. The block directional image is first binarized by a set of directional codes and its bifurcations are removed by a set of filter operators. The output of this algorithm is a set of strings which contains the characteristic string for the fingerprint under investigation. This string is then transformed to its one-dimensional binary representation to form the structural part of the feature vector.

```
x 1 x          x 1 x          x 1 x          x x x
1 0 1          1 0 x          x 0 1          1 0 1
x x x          x 1 x          x 1 x          x 1 x

x x x          1 x x          x x 1          1 1 1
x 1 x          1 1 x          x 1 1          x 1 x
1 1 1          1 x x          x x 1          x x x
```

Fig. 12. Types of bifurcations.

```
x 1 x          x 0 x     x x x          x x x
1 0 1 ───────▶ 1 1 1     x 1 x ───────▶ x 0 x
x x x          x x x     1 1 1          1 1 1
```

Fig. 13. Filter operators to remove bifurcations.

4. Statistical features extraction

Methods for the statistical description of texture can be utilized to describe the curves extracted from the fingerprint. The method of moments is one of these and we will use it in this paper, where the characteristic string can be described quantitatively by using moments. Table 1 represents the rank table where each string symbol s_i is assigned a rank r_i.

Measures of texture computed by using only histograms suffer from the limitation that they carry no information

Table 1
Feature symbols ranks

Feature symbol	Rank
H	1
L	2
I	3
J	4
K	5
M	6
N	7
O	8
X	9
Y	10
P	11
Q	12
R	13
S	14
Z	15
T	16
U	17
V	18
W	19
A	20

```
    1                                     1
  1     1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1     1
1     1                                 1     1
    1     1     1     1     1 1     1     1
  1     1     1     1     1     1     1     1
1     1     1       1     1       1     1     1
    1     1     1   1     1   1     1     1
  1     1     1     1     1     1     1     1
1       1   1       1     1       1     1     1
    1   1   1       1     1   1     1     1
  1     1     1     1     1     1     1     1
1       1       1     1     1     1     1     1
    1   1   1     1     1     1     1     1
  1     1     1     1     1     1     1     1
1         1     1     1     1     1     1     1
    1       1     1     1     1     1     1
  1     1     1     1     1     1     1     1
1     1         1     1     1     1     1     1
                  1     1     1     1     1
  1 1 1 1 1 1 1     1     1     1     1     1
1               1     1     1     1     1     1
                  1     1     1     1     1
  1 1 1 1 1 1 1     1     1     1     1     1
1               1     1     1     1     1     1
```

Fig. 14. Binary block directional image after removing bifurcations.

regarding the relative position of symbols in the string. One way to bring this type of information into the texture analysis process is to consider not only the ranks of symbols but also their positions in the string. The weighed rank ω_i of every symbol s_i is computed by multiplying its rank r_i obtained from Table 1 by its position ρ_i in the string:

$$\omega_i = r_i \times \rho_i \tag{2}$$

The total weighed ranks T is computed by:

$$T = \sum_{i=1}^{L} \omega_i \tag{3}$$

where L is the length of the characteristic string.

The probability $g(r_i)$ can be regarded as the corresponding histogram for the characteristic string. It is computed by dividing the weighed rank ω_i associated with the symbol s_i by the total weighed ranks T:

$$g(r_i) = \frac{\omega_i}{T} \tag{4}$$

If a characteristic string has a length L then the method of moment is based on computing the nth moment μ_n for this string around its mean rank m:

$$\mu_n = \sum_{i=1}^{L} (r_i - m)^n g(r_i) \tag{5}$$

The mean rank m is given by:

$$m = \sum_{i=1}^{L} r_i g(r_i) \tag{6}$$

where L is the length of the characteristic string, $g(r_i)$: is the probability that the symbol s_i represented by its rank r_i takes the weight ω_i, μ_n is the nth moment for the characteristic string, it is directly related to the shape of the curve represented by this string.

For example, consider the characteristic string MPQK for the arch type. From Table 1, symbol M has rank 6 with position 1, symbol P has rank 11 with position 2, symbol Q has rank 12 with position 3 and symbol K has rank 5 with position 4. We calculate the weighed rank for each symbol as follows:

$$\omega_1 = 6 \times 1 = 6 \quad \omega_2 = 11 \times 2 = 22$$

$$\omega_3 = 12 \times 3 = 36 \quad \omega_4 = 5 \times 4 = 20$$

```
0 0 0        x 1 x        0 0 0        x 0 0
0 1 0 → H    0 1 0 → L    x 1 0 → I    1 1 0 → J
x 1 x        0 0 0        1 x 0        x 0 0

1 x 0        0 x 1        0 0 x        0 0 0
x 1 0 → K    0 1 x → M    0 1 1 → N    0 1 x → O
0 0 0        0 0 0        0 0 x        0 x 1

0 0 0        0 0 0        0 0 1        1 0 0
0 1 1 → P    1 1 0 → Q    0 1 0 → R    0 1 0 → S
1 0 x        x 0 1        0 1 x        x 1 0

x 1 0        x 0 1        1 0 x        0 1 x
0 1 0 → T    1 1 0 → U    0 1 1 → V    0 1 0 → W
1 0 0        0 0 0        0 0 0        0 0 1

0 0 0        0 1 0        0 1 0        0 0 0
1 1 0 → X    1 1 0 → Y    0 1 1 → Z    0 1 1 → A
0 1 0        0 0 0        0 0 0        0 1 0
```

Fig. 15. Feature masks and their symbols.

The total weighed ranks for this characteristic string is:

$$T = 6 + 22 + 36 + 20 = 84$$

The probability of finding symbol s_i with rank r_i is:

$$g(\mathrm{M} = 6) = 6/84 = 0.071$$

$$g(\mathrm{P} = 11) = 22/84 = 0.262$$

$$g(\mathrm{Q} = 12) = 36/84 = 0.429$$

$$g(\mathrm{K} = 5) = 20/84 = 0.238$$

The mean value m for the random variable r is:

$$m = 6 \times 0.071 + 11 \times 0.262 + 12 \times 0.429 + 5 \times 0.238$$

$$= 9.646$$

We shall use the second moment μ_2 to represent the texture for a given string, to compute the second moment:

$$\mu_2(r) = (6 - 9.646)^2 \times 0.0714 + (11 - 9.646)^2 \times 0.262 + (12 - 9.646)^2 \times 0.429 + (5 - 9.646)^2 \times 0.238 = 8.943.$$

4.1. Minimum distance classifier (Euclidean Distance Measure)

Suppose that the number of pattern classes is N, then for each characteristic string representing pattern i belongs to class j we associate a second moment μ_{2ij} where $j = 1, 2, 3 \ldots N$. If the number of patterns in class j is M_j then the mean second moment $\bar{\mu}_{2j}$ which considered the center for this class is:

$$\bar{\mu}_{2j} = 1/M_j \sum_{i=1}^{M_j} \mu_{2ij} \tag{7}$$

If the unknown string has a moment $\mu_{2\mathrm{new}}$, then its distance from the center of class j is given by the Euclidean Distance Measure (EDM):

$$dj(\mu_{2\mathrm{new}}) = \|\mu_{2\mathrm{new}} - \bar{\mu}_{2j}\| \tag{8}$$

where $j = 1, 2, \ldots N$ and $dj(\mu_{2\mathrm{new}})$ is the Euclidean distance between the second moment of the unknown pattern $\mu_{2\mathrm{new}}$ and the center of the class j $(\bar{\mu}_{2j})$.

The Euclidean norm:

$$\|\mu_{2\mathrm{new}} - \bar{\mu}_{2j}\| = [(\mu_{2\mathrm{new}} - \bar{\mu}_{2j})^T(\mu_{2\mathrm{new}} - \bar{\mu}_{2j})]^{1/2} \tag{9}$$

we then assign the unknown moment $\mu_{2\mathrm{new}}$ to class j if $dj(\mu_{2\mathrm{new}})$ has the smallest difference.

4.2. Summary

In this section we use the method of moments to extract the statistical features for a fingerprint from its characteristic string which can be described quantitatively by using the second moment. The Euclidean distance measure is used to measure the distance between the second moment of an unknown pattern and the center of each class which is the mean second moment of this class. These distance measures form the statistical part of the feature vector.

5. ANN classification

To confirm the efficiency of our combined extracted features, a three-layer feed-forward ANN was designed, see Fig. 16. For the six-class problem the network is composed of six independent subnetworks with one subnetwork for each class and has connections from the combined structural variables and statistical measures in the input layer to the inputs of the first hidden layer. The input layer is composed of n independent variables $x_1, x_2, \ldots x_n$ of the binary string representing the structural features extracted from the block directional image of the pattern, where $x_i \in \{0, 1\}$. We let the size of $n = 180$ which is the

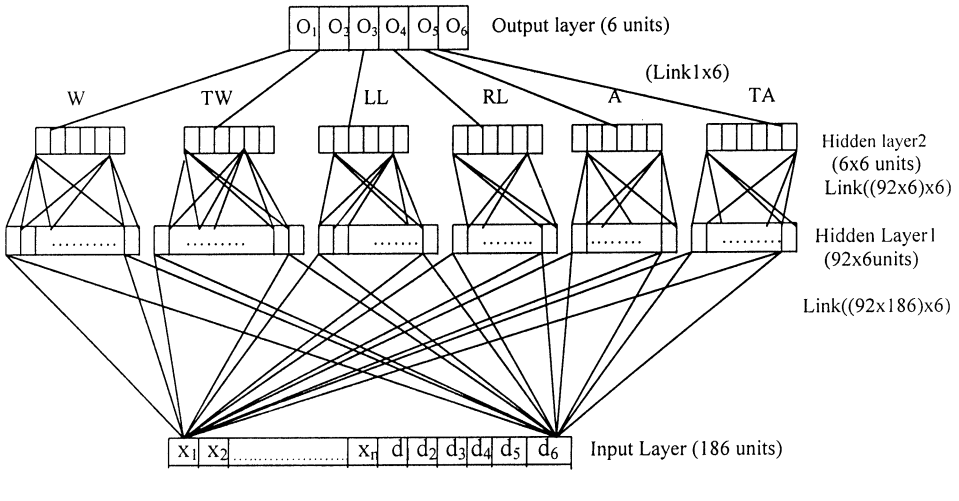


Fig. 16. Network structure.

binary representation for the plain whorls which have the longest curves. Also we have $d_1, d_2, ...d_k$ where d_j is the Euclidean Distance Measure for class j, where $j = 1, 2...k$ and k is the number of pattern classes which is in our case is six classes. Now, our input vector to the input layer is a multivariate vector, which has the following general form: $(x_1, x_2, ...x_n, d_1, d_2, ...d_k)$. The labels of the subnetworks are A for arches, TA for tented arches, LL for left loops, RL for right loops, W for whorls and TW for twin loops. Decomposing the network into subnetworks one for each class improve the classification rate than using a single network for all classes. Some of the fingerprint classes are not well separated and there is a possibility of classes overlapping such as the overlap between plain whorls and twin loops, loops and twin loops. In case of single network for all classes the learning rate may stuck to a local minimum thus reducing the classification rate of the network. In case of decomposing the network into subnetworks, these subnetworks are independent of each other and trained independently on its data set thus excluding the possibility of classes overlapping which causes the decreases in the classification rate. The output layer is used only to display the outputs of the subnetworks, it has no role in the learning process. It is composed of six units ($O_1, O_2...,O_6$) with no weights on its connections with the second hidden layer. Each subnetwork was trained independently with a data set carrying the characteristics of the class. Weights are adjusted using the error backpropagation learning rule.

5.1. Summary

In this section we present a three-layer feed-forward ANN which is composed of six subnetworks one for each class. The input layer is composed of 180 independent variables of the binary string and six Euclidean distance measures one for each class of the fingerprints, i.e. the input vector is a multivariate vector composed of 186 elements. Each subnetwork is trained independently on its data set by using the error backpropagation learning rule.

6. Experimental results and discussion

Our fingerprint classification algorithm was tested on the Egyptian Criminal Evidence Fingerprint database, which contains 30,000 fingerprints. Most of these images are of poor quality and contain a high ratio of noise due to over-inking or no inking. Most of them contain text, additional lines,tabulations in the background, they are not centered or properly oriented (see Fig. 17). In order to reject fingerprints with bad qualities we took a threshold value 65% of the entire image's average density. Images below this threshold were rejected. The test was done on a random sample of 1500 fingerprints, 446 whorls, 240 twin loops, 327 left loops, 303 right loops, 114 arches and 70 tented arches. Each subnetwork was trained by using 100 patterns, which carries the characteristics of its class. The error in classifying the 1500 fingerprints into six classes was 4%, the largest source of error was due to the classification of twin loops as whorls and whorls as twin loops. Fingerprints that were assigned different classes, their classification was assumed to be correct, because even fingerprint experts can assign more than one class to the same pattern. The confusion matrix for the six-class classification problem is shown in Table 2. If we combine whorls and twin loops into the same class, then the classification error for the resulting five-class problem reduces to 1%. The confusion matrix for the five-class classification problem is given in Table 3. If we combine arches and tented arches in the same class then the classification error for the resulting four-class problem reduces to 1%. The confusion matrix for the four-class classification problem is given in Table 4. The rows in Tables 2–4 do not sum up to the sample size assigned to that class, because some of the fingerprints were assigned different classes. The values in the column labeled as 'Unknown' denotes the number of fingerprints the algorithm failed to assign them a class.

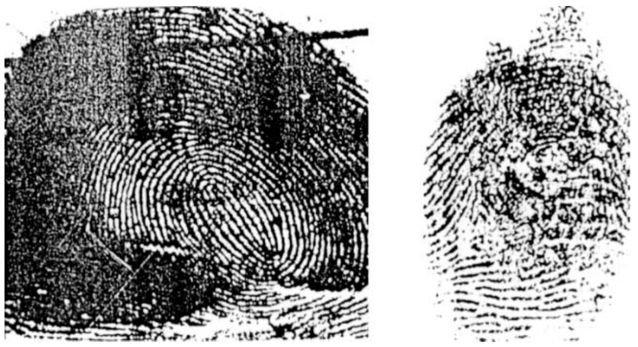


Fig. 17. Examples of poor quality images.

For the five-class problem, Karu and Jain (1996) have used a classification program based on singular points extraction and performs classification based on the number and location of the detected singular points. The classifier was tested on the NIST-4 database the best classification accuracy was 85.4% for the five-class problem and 91.1% for the four-class problem on the NIST-9 database. Also for the five-class problem, Wilson et al. (1993), tested their classifier on NIST-4 database and the best reported error were 17% with equally spaced grid and 14% with unequally spaced grid with 10% rejection. They trained their classifier on 2000 images and tested on another 2000 images of the same fingers. Our algorithm was trained on 100 patterns per class and tested on 1500 images with an error rate of 1% and classification accuracy of 99% for the four-class problem,an error rate 1% and classification accuracy around 99% for the five-class problem and an error rate 5% and classification accuracy of 95% for the six-class problem.

6.1. Computational complexity

The above procedures have been implemented on a PC

Table 2
Six-class classification problem

True classification	Assigned class							Classification accuracy (%)
	Whorls	Twin loops	Left loops	Right loops	Arches	Tented arches	Unknown	
Whorls	434	33	16	4	4	0	12	97
Twin loops	25	192	0	5	0	8	10	80
Left loops	0	0	321	1	0	0	5	98
Right loops	0	0	0	302	0	0	1	99
Arches	0	0	0	0	114	0	0	100
Tented arches	0	0	4	2	3	61	0	87

Table 3
Five-class classification problem

True classification	Assigned class						Classification accuracy (%)
	Whorls	Left loops	Right loops	Arches	Tented arches	Unknown	
Whorls	684	18	10	4	16	2	99
Left loops	0	321	1	0	0	5	98
Right loops	0	0	302	0	0	1	99
Arches	0	0	0	114	0	0	100
Tented arches	0	4	2	3	61	0	87

Table 4
Four-class classification problem

True classification	Assigned class					Classification accuracy (%)
	Whorls	Left loops	Right loops	Arches	Unknown	
Whorls	684	18	10	4	2	99
Left loops	0	321	1	0	5	98
Right loops	0	0	302	0	1	99
Arches	0	4	2	178	0	97

with 100 Mhz. Processor, math coprocessor and 16 MB RAM. The procedure for computing the block directional image occupies about 15 k. words of computer memory and it takes about 5 s to compute a directional image for a 256×256 fingerprint image. The smoothing procedure used for block smoothing occupies 8 k words of computer memory and takes about 4 s to smooth the block directional image. The line tracing algorithm used for the characteristic string extraction occupies 40 k words of computer memory and takes about 2 s to extract the set of strings. The procedure used to transform a characteristic string to its one-dimensional binary string and compute its moment and the Euclidean distance measures occupies 14 k words of computer memory and takes less than 0.5 s of execution time. For the ANN part, Each subnetwork occupies 161 k words of computer memory and takes about 40 min of training.

Acknowledgements

In memory of my mother, Professor Olfat Y. Mansour, for her continuous encouragement. I would like also to acknowledge Miss Afifi for editing the manuscript.

References

Blue, J. L., Candela, G. T., Grother, P. J., Chellappa, R., & Wilson, C. L. (1994). Evaluation of pattern classifiers for fingerprints and OCR applications. *Pattern Recognition*, *27* (4), 485–501.

Cappelli, R., Lumini, A., & Maio, D. (1999). Fingerprint classification by directional image partitioning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *21* (5), 402–421.

Federal Bureau of Investigation (1984). *The science of fingerprints: Classification and uses*, Washington, DC: U.S. Government Printing Office.

Hong, L., & Jain, A. (1998). Integrating faces and fingerprints for personal identification. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, *20* (12), 1295–1307.

Hong, L., Yifei, W., & Jain, A. (1998). Fingerprint image enhancement: algorithm and performance evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20* (8), 777–789.

Kameswara, R., & Black, K. (1980). Type classification of fingerprints: A syntactic approach. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, *PAMI-2* (3), 223–231.

Kamijo, M., Hiroshi, M., & Kojima, K. (1992). Classification of fingerprint images using an artificial network. *Systems and Computers in Japan*, *23* (3), 89–101.

Karu, K., & Jain, A. K. (1996). Fingerprint classification. *Pattern Recognition*, *29* (3), 389–404.

Kawagoe, M., & Tojo, A. (1984). Fingerprint pattern classification. *Pattern Recognition*, *17*, 295–303.

Leung, W. F., Leung, S. H., Lau, W. H., & Luk, A. (1991). Fingerprint recognition using artificial network, artificial networks for signal processing. *Proceedings of the 1991 IEEE Workshop* (pp. 227–235).

Lumini, A., Maio, D., & Maltoni, D. (1997). Continuous versus exclusive classification for fingerprint retrieval. *Pattern Recognition Letters*, *18*, 1027–1034.

Markham, I. S., & Ragsdale, C. T. (1995). Combining neural networks and statistical predictions to solve the classification problem in discriminant analysis. *Decision Sciences*, *26* (2), 229–241.

Mehtre, B. M. (1993). Fingerprint image analysis for automatic identification. *Machine Vision and Applications*, *6*, 124–139.

Moayer, B., & Fu, K. S. (1975a). A syntactic approach to fingerprint pattern recognition. *Pattern Recognition*, *7*, 1–23.

Moayer, B., & Fu, K. S. (1975b). tree system approach for fingerprint pattern recognition. *IEEE Trans. Comput.*, *C-25*, 262–274.

Neto, H. V., & Borges, D. L. (1997). Fingerprint classification with artificial networks. *Proceedings of the Fourth Brazelian Symposium on Artificial Networks* (pp. 66–72).

Senior, A. (1997). Hidden Markov model fingerprint classifier. *Proceedings of the 31ST Asilomar Conference on Signals, Systems and Computers*, Part 1 (of 2) (pp. 306–310). Pacific Grove, CA, USA.

Tou, J.T., & Hankley, W. J. (1968). Automatic fingerprint identification and classification analysis via contextual analysis and topological coding. *Pictorial pattern recognition* (pp. 411–456). Thompson Book Company.

Wilson, C. L., Candela, G. T., & Watson, C. I. (1993). Neural network fingerprint classification. *J. Artificial Neural Networks*, *1* (2), 1–25.