JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 18, 257-280 (2002)

A Study on Chinese Carbon-Signature Recognition

SHENG-FUU LIN, YU-WEI CHANG AND CHIEN-KUN SU
Department of Electrical and Control Engineering
National Chiao Tung University
Hsinchu, 300 Taiwan

A novel off-line algorithm and a modified chain code for recognizing signatures are proposed in the present study. Carbon paper is used to detect force distributions when people write their signatures. First of all, the signature contours are located and the upper and lower profiles are generated; then, these are used to classify the given signature. Both the gray-scale and the chain code characteristics of a signature are used to extract structure and force distribution features, which are then transformed into a normalized vector. Finally, a Supervised Fuzzy Adaptive Hamming Network (SFAHN) is employed to interpret the feature vector in order to determine whether the signature is genuine or not. Simulation results show that the proposed algorithm has a good recognition rate.

Keywords: signature recognition, upper profiles, lower profiles, chain codes, feature extraction

1. INTRODUCTION

Much research has been done on recognition of hand-written characters and signatures. The origin of character recognition goes back as far as 1870. Character recognition and signature identification are two important issues today in business activities and are classified as data processing applications. Signature recognition and character recognition are very similar in many respects. A common application of signature recognition is signature identification. Whether a signature is genuine or not can affect economic security since false identification of the signature on a bank check, a commercial form, or personal identification maybe result in huge losses. To handle such tasks, signature recognition should be quick, convenient, and highly reliable.

Although identification can be achieved using ID cards, secret hidden marks, passwords and other methods, these methods suffer from being duplicated easily or being too complex for people to use easily. However, everyone remembers his name easily and has his own particular style of handwriting. In some cases, fingerprints or voiceprints can be used for identification, though bank checks and commercial forms usually need faster processing.

Signature recognition is usually divided into two stages. In the first stage, the characteristic features of a signature are extracted using sensors (electronic pens or tablets), or else images are obtained by means of scanning. The second, judgment stage determines whether the signature is genuine or forged. Many approaches, such as statistical,

Received February 25, 2000; revised October 4, 2000; accepted November 30, 2000.
Communicated by Wen-Lian Hsu.

syntactic, and neural network methods have been investigated in recent years. The feature-extraction techniques can usually be divided into two types: *on-line* and *off-line*. An on-line method extracts features while a person is signing; an off-line method extracts features after a person has finished writing his signature. Since an on-line method must extract features in real time, it requires the use of sensorial input tools [4, 11, 13, 25, 28]. Zimmermann and Varady [28] used a digitizing graphic tablet to detect pen/writing surface contact over a fixed period of time. Herbst and Liu [11] detected acceleration while people were writing their signatures. Dimauro, Impedovo, and Pirlo [4] also used a graphic tablet to determine pen-down and pen-up positions. An on-line method is more informative than an off-line method, but it is limited by the need for special devices. Since an off-line method cannot get real-time information, it only focuses on the analysis of shape, structure, strokes, and other features of the signature. An off-line method requires some special character-recognition processing techniques, such as thinning [6, 27, 29], slope detection, angle calculation, finding the number and positions of crossing strokes and closed-loops, determining arc curvature, selective searching [30], and so on [1, 3, 9, 14, 17, 18, 20, 24, 25, 29, 30]. In an off-line technique, some useful features obtained by an on-line method are missing, for example, the force distribution and acceleration of a signature. Ammar, Yoshida, and Fukumura [1] used a pencil for signing in order to record the force employed, but they found it difficult to determine the threshold value, which is a limitation of input tools. A pencil is normally not used while conducting business, and pencil signatures can be distorted or even disappear due to external influences. Feature extraction is also important since everyone write his or her signature differently but in much the same way each time.

The methods used to determine whether a signature is genuine or not can usually be classified into three categories:

1) statistical methods [1, 5, 11, 14, 17, 18, 20],
2) syntactic methods [9, 22], and
3) neural network methods [5, 17, 18, 21, 23].

Genetic Algorithms (GAs) can also be used in the judgment stage [25], but they are not often discussed. Each method mentioned above has its own advantages and disadvantages:

1) *Statistical approaches*: Since our habits change, it is inevitable that there will be some differences between the signatures that are written by the same signer. These differences are not serious, but sometimes people may be disturbed by external or internal events, and may generate some particularly unusual signatures. Hence, special patterns must be included in the database and this requires the use of many signature patterns to create a more complete database. Thus, a statistical method is time consuming and requires more operations.

2) *Syntactic approaches*: In syntactic approaches, the structure of an entity is of paramount importance. By analyzing the structure, the classification and description can be determined. Two major rule types are:
   a) formal grammars and
   b) relational descriptions (principally graphs).

Signature skeletons are often located by means of thinning. The results of thinning [6, 27] are usually not good, especially when there are some abnormal forks at stroke intersection points. Since a syntactic method needs correct skeleton structures, Liao and Hung [15] proposed the use of Bernstein-Bezier Curve Fitting to solve abnormal forking problems in character recognition. However, more time is required for preprocessing, and this method is suitable for printed characters only. The relational description must also be suitable and robust; otherwise, one cannot use Liao and Hung's method to get appropriate features to represent the character.

3) *Neural network approaches*: The most popular neural-network-based pattern recognition algorithm, called *Back-Propagation* (BP), is widely used for signature recognition [5]. *Adaptive Resonance Theory* (ART) is another well-known neural network approach, and is usually used in the object recognition field [2]. However, neural networks can only be used to solve problems on a case by case basis. Each problem requires the use of a suitable neural network to obtain a solution. The inputs of a neural network must be fixed dimension vectors, which are obtained by means of feature extraction. Extraction of effective features can reduce the training time required and produce good performance. Thus, feature extraction is a key part of neural network approaches. In addition, arranging the training-pattern order, choosing the number of nodes, and adjusting other neural network parameters require experience and are usually done by means of a trial and error.

The off-line signature recognition method can be divided into two stages: *identification* and *verification*. Identification means determining the group that a signature belongs to. On the other hand, verification means verifying a supposition that a particular signer wrote the signature. In other words, identification is the same as classification since it also classifies signatures. Therefore, whether a signature is genuine or not is judged during the verification process. Much research has focused on verification issues, but little research has dealt with identification problems [17].

An *automatic signature recognition system* (ASRS) must accomplish both identification and verification. When we want to determine whether a signature is genuine or not, we usually go by the overall outline of the entire signature to be first. If it varies in outline from the genuine signature, then the signature can be easily and quickly determined to be forged. If not, then the fine parts of the entire signature must be analyzed in detail. For the above reason, the entire outlines of signatures are saved as features. Parisses [16] and Gupta [8] adjusted the parameters to get more accurate or coarser feature vectors. Using a statistical method to achieve easy and robust identification is a not bad choice. The *Nearest Neighbor Rule* (NNR) is a popular method of classification in statistics since it can find the minimum difference between test and database patterns.

In recent years, the use of a neural network in the verification step has often been proposed. Drouhard, Sabourin, and Godbout [5] used a back-propagation neural network for verification. Although Back-Propagation (BP) algorithms are popular learning methods, they have the disadvantages of requiring a large amount of training time and usually finding a local-minimum solution. The adaptive resonance theory (ART) has also been applied to pattern recognition. It is an on-line learning architecture that does not need prior knowledge of the total number of pattern classes. When a new pattern is presented,

ART does not need to learn all the patterns, including those that have been previously learned. ART is much more convenient to use than BP. Tseng and Huang [23] used the ART-1 neural network to perform verification. In a study on verification, an experiment comparing six methods was conducted [17]. Pottier [17] showed that using a 2-layer artificial neural network with fully connected perceptrons gave the best recognition rate compared with 1-NNR, 3-NNR, etc. Adaptive Hamming networks have been proved to be equivalent to the ART-1 model with a fast-learning architecture that minimizes the searching process [12]. The fuzzy adaptive Hamming network [13], which evolved from the adaptive Hamming network, can deal with analog input patterns. Furthermore, the *Supervised Fuzzy Adaptive Hamming Net* (SFAHN) [13] not only accepts analog input patterns, but also can operate in a supervised-learning mode. Since neural networks have good recognition rates, we use SFAHN in the verification phase.

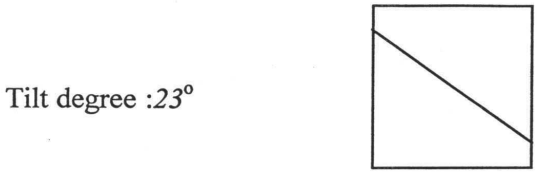
In order to save force distributions as features when signatures are written, we use carbon paper as the force sensor. Since the characteristics of ink allow it to diffuse rapidly and consistently on a normal paper, the intensity of the strokes hardly varies, making it difficult to get a fine force distribution record. However, carbon paper generates darker and lighter strokes on the paper beneath it when people use more or less strength. We scan the signatures duplicated by the carbon paper using a scanner and save them as images. Then, a fast and easy thinning algorithm [6] is used to get signature skeletons. In order to get features that are not influenced by abnormal forks, a modified algorithm has been developed from the chain code algorithm. Finally, SFAHN is used to verify whether the signatures are genuine or not. Since SFAHN can be trained on-line, it requires less training time and has a good recognition rate.

2. CHAIN CODES AND SUPERVISED FUZZY ADAPTIVE HAMMING NETWORK

2.1 Chain Codes

Chain codes [6] have been used in pattern recognition, where the contours tracked and chain codes used to record variation in direction. They are also used to represent a boundary as a sequence of connected straight-line segments. Digital images are usually acquired and processed in a grid format with equal spacing in the x and y directions, so that chain codes can be generated by following a boundary in a clockwise direction and assigning a specific direction to the segment connected with a pair of pixels. An 8-directional chain code was used in this study. The code words 0, 1, 2, 3, 4, 5, 6, and 7 represented the changes in direction between pixels, and they represented right, right-up, up, left-up, left, left-down, down, and right-down, respectively.

Although chain codes are usually used within closed boundaries, we can also apply them in signature recognition. In Fig. 1, there are three lines with tilt degrees of 23°, 45°, and 80°, respectively. These lines were scanned from left to right and from top to bottom. Skeletons of these lines comprise the right, down, and right-down directions mostly, so the numbers for the right, down, and right-down code words could be determined. We calculated the ratio of a codeword by dividing the number of the specific codeword by the total number of codewords, and the ratios of code words 0, 6, and 7 could be used to



Code	0	5	6	7
Number of code	73	0	0	54
Ratio (%)	57.48%	0%	0%	42.519%

(a)

Code	0	5	6	7
Number of code	0	0	0	127
Ratio (%)	0%	0%	0%	100%

(b)

Code	0	5	6	7
Number of code	0	0	105	22
Ratio (%)	0%	0%	82.677%	17.322%

(c)

Fig. 1. Analyses of three lines with different tilt degrees using chain codes.

describe the tilt state of each line in Fig. 1. The larger the ratio for code 6, the greater the tilt degree. The distribution of the ratio for each code reveals the useful features. People were asked to sign their names on a horizontal line, called the **base-line**, a common requirement for bank checks and other documents, to obtain horizontal signatures. If the signatures were not written along the horizontal line, we would get useless chain codes because it would not represent the real structure of the signature.

Chain codes can be used to generate fixed dimension feature vectors that represent structures, and these features can be used as inputs of a neural network. By adding variation of gray values to chain codes, the force distribution features can be extracted. Chain codes can be modified and used to analyze signature structures and force distributions for the purpose of verification.

2.2 Supervised Fuzzy Adaptive Hamming Network

In this section, the Supervised Fuzzy Adaptive Hamming network (SFAHN) will be introduced as a feature classifier. SFAHN receives analog input patterns and learns them in a supervised mode. If an input pattern is similar to one of its stored patterns, the weights connected to the neuron of the stored pattern are updated. On the other hand, if the input pattern is not similar to any of the stored patterns, then a new neuron is created, and the weights are updated to store the new pattern. When a new input pattern is learned, the SFAHN does not need to acquire the patterns learned previously. SFAHN operates at a fast speed since it does not waste any time on searching. Hence, the SFAHN feature classifier was adopted for our system in order to achieve quick recognition.

SFAHN consists of complement coding, a score-matching net, MAXNET, map field, and match-tracking. Although a detailed explanation of SFAHN was given by Hung and Lin [13], a brief description is given below.

When SFAHN receives an M-dimensional analog input pattern, it produces $2M$ features using complement coding. These $2M$ features are input to the score-matching net to generate N outputs. After applying a piecewise linear function to the score-matching operation, a choice function is used to select a maximum value among the N outputs. MAXNET is employed to choose the neuron that has the maximum choice function value, and then a comparison is made between the class label of the chosen neuron and the desired input class label according to the map field. If these two classes are the same, then the weights of the chosen neuron are updated. Otherwise, match tracking is used to increase the threshold value of the score-matching net, and another neuron with a maximum choice function value is found and selected. After searching all the neurons with previously learned patterns, if no neuron with the same class label as the desired input class label is found, then a new neuron is created and assigned to the desired class label, and its weights are updated to store the new input pattern. We will now describe the recognition phase of SFAHN. In the recognition phase, the operating components consist of complement coding, a score-matching net, and MAXNET. When a test pattern is input and a complement is coded, the features are input to the score-matching net, and the choice function value is computed. MAXNET is used to select the neuron with the maximum choice function value, and from the map field, the class label of the chosen neuron corresponding to the output is the recognized result.

In the identification phase, a statistical method known as 1-NNR is employed to classify signatures. The upper and lower profiles of each test signature are created. Then, after finding the minimum distance between the test and the database profiles, we assign a numerical label (0, 1, 2, ...) to each classified signature. SFAHN is used as a feature verifier to determine whether the signatures are genuine or not. The features of each signature are extracted and saved in vector form. After the internal operations are performed, the output of SFAHN is the number of the chosen neuron with the maximum choice

function value. This number is the result of the given signature recognized by SFAHN. For example, if the number of the output node with the maximum value of SFAHN is not the same as the label given in the identification, we conclude that the signature is forged.

3. THE ALGORITHM FOR IDENTIFICATION AND VERIFICATION

3.1 System Overview

A block diagram of our signature recognition system is shown in Fig. 2. Two major processes are performed by the proposed system: *identification* and *verification*.

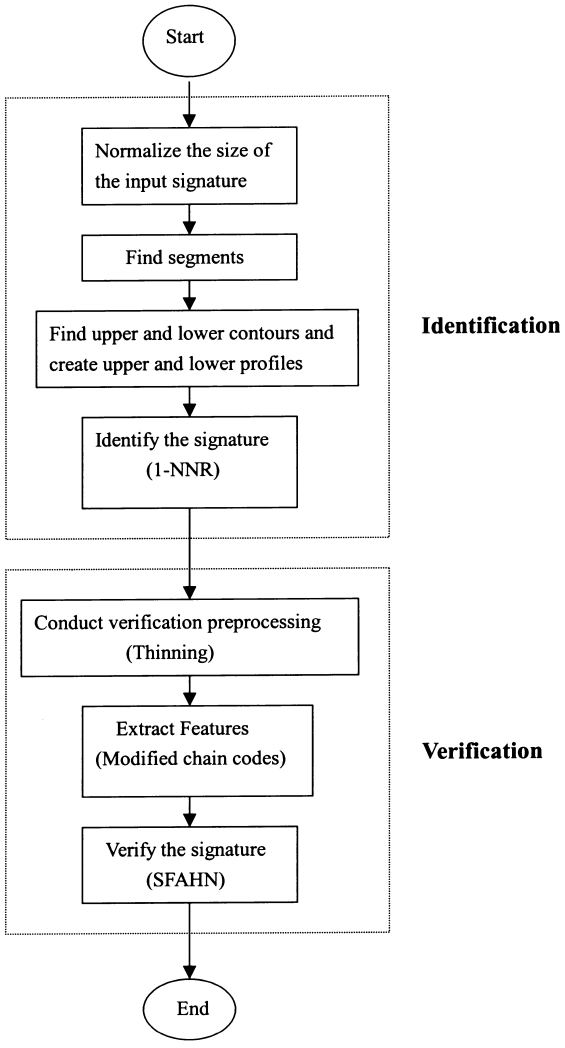


Fig. 2. Block diagram of the signature recognition system.

In the identification phase, first of all, the size of an input signature is normalized. Second, the input signature is segmented into several parts. Third, the *upper contour* and the *lower contour* of each segment are determined, and these contours are used to create an *upper profile* and a *lower profile* for the input signature. Finally, these profiles are used to classify the signature, and a numerical label is assigned to each of the classified signatures.

In the verification phase, first of all, the thinning process is used to obtain the signature skeletons and the force distributions. There are many redundant pixels around the signature boundaries, and we must avoid influence from redundant pixels in order to get useful force distributions. After the thinning process, a modified chain code is used to perform feature extraction. Finally, SFAHN as a feature verifier is employed to determine whether the input signature is genuine or not.

It is assumed that all the signatures are written from left to right. The process is shown in Fig. 3. A person signs his name on the "A" sheet of paper, and a carbon paper duplicates the signature on the "B" sheet of paper. The "A" sheet is graph paper whereas the "B" sheet is plain white paper. Generally, a signature duplicated via carbon paper has better contrast than the original one, so we can extract better features from the duplicated signature.

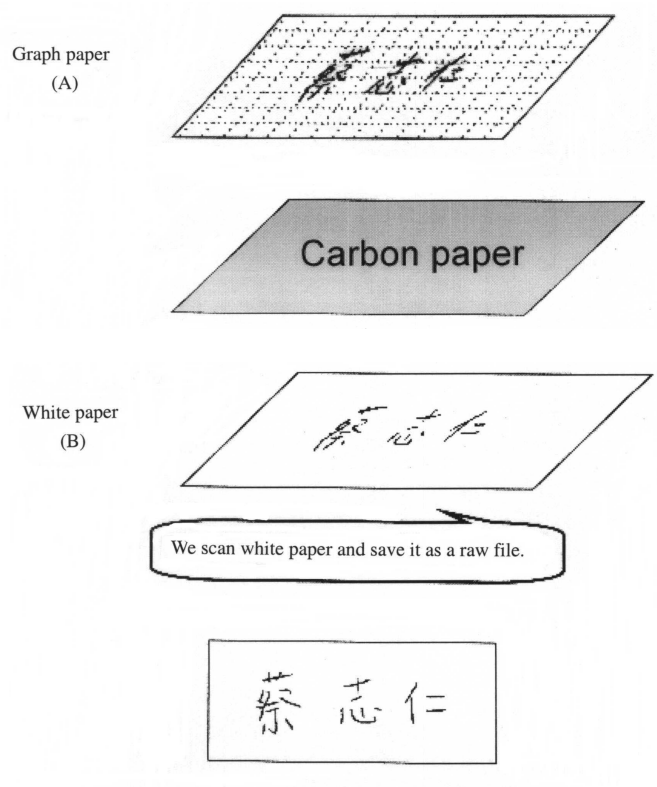
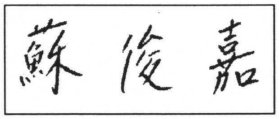
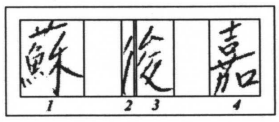


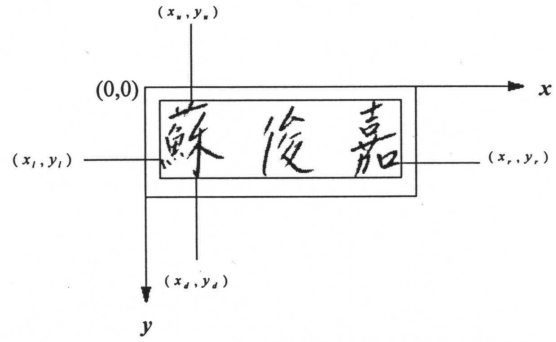
Fig. 3. The process for getting signatures.



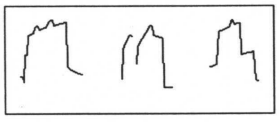
(a) Normalize the digitalized signature.



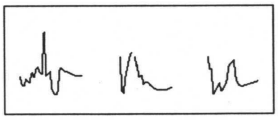
(c) Segment number is 4.



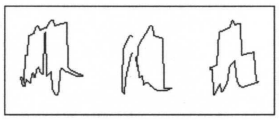
(b) Minimum rectangle that can include all the signature pixels.



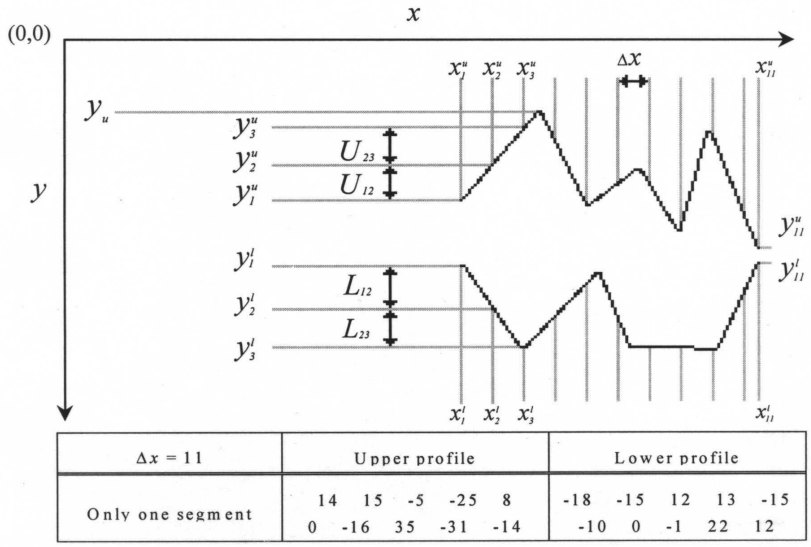
(d) Upper contour.



(e) Lower contour.



(f) The entire contour.



$\Delta x = 11$	Upper profile	Lower profile
Only one segment	14 15 -5 -25 8 0 -16 35 -31 -14	-18 -15 12 13 -15 -10 0 -1 22 12

(g) Example of sampling contours and creating profiles.

Fig. 4. Contours and profiles.

$\Delta x = 3$	Upper profile	Lower profile
segment (1)	-5 25 19 1 7 -4 2 2 5 -5 1 1 1 -41 -3 -1 -2 -1 0	-4 5 -2 5 8 -4 35 -37 3 -20 -1 -24 -2 -1 -2 -1 0 0
segment (2)	32 6 1	11 13 11
segment (3)	19 5 5 6 -9 -1 -1 -46 0 0 0	-10 -7 2 -3 -2 -2 0 0 0 2 0
segment (4)	1 1 31 1 1 2 5 -2 0 -28 1 0 1 -27 0	-23 4 6 -6 7 15 4 -17 -5 -1 1 1 1 1 -1

(h) The profiles of Figure 3.4 (a).

Fig. 4. (Cont'd) Contours and profiles.

The person is asked to write his name along the horizontal line on sheet A. This line serves as a **base-line**. Then, some people are asked to forge the signature, after showing them sheet A. The forgers are allowed to see the structures of the genuine signature, but they do not know its force distributions. Assuming that the signature on sheet B has a uniform background, it is scanned using a scanner, and the image is saved as a raw data file. The scanned files are saved in gray-level mode where the white pixels are considered to be background, and all the others are considered to be signatures. Eight-bit-per-pixel resolution is used to represent the gray level of each pixel.

3.2 Identification

The aim of identification is to get an approximate outline of the signature. Parisse [16] proposed a method for obtaining global word shape, though his original goal was to perform off-line handwritten recognition. He used several vectors to approximate the outline of a signature. Parisse's method can be used to perform identification, and it also serves as a classifier. However, his coordinate system is not suitable for our signature recognition system. In order to define an effective and appropriate coordinate system, we refer to the algorithm in [8] and propose a modified version for the purpose of identification.

1) *Normalization*: We choose a suitable fixed-size rectangle as a normalization standard size. If an input signature is larger or smaller than this rectangle, then the size of the input signature is changed such that the signature fits exactly in this rectangle.
2) *Segmentation*: For any given signature (Fig. 4 (a)), we assume that the origin point is in the top-left corner of the image, defined as (*0,0*). The rightward direction is defined as positive *x* and the downward direction as positive *y*; then, the uppermost, lowest, leftmost, and rightmost pixels in the image are located. The uppermost pixel (x_u, y_u) in the image is the point that belongs to the signature with the lowest *y* component value. If more than one pixel qualifies, then the pixel with the lowest *x* component value is selected as the uppermost pixel. Similarly, the lowest pixel (x_d, y_d) has the highest *y* component value. The leftmost pixel (x_l, y_l) and the rightmost pixel (x_r, y_r) are deter-

mined in a similar way. Then, these four points are employed to generate the minimum rectangle size that can include all the pixels of the signature (Fig. 4 (b)). Since signatures are horizontal, they are separated into several segments based on the spaces between the signature elements in the minimum rectangle. For example, each of the columns in the minimum rectangle is scanned from left to right. If the column being scanned has no pixel that belongs to the signature, whereas its right/left column has pixels that belong to the signature, then this column is saved as a start/end column of a segment. The column between a start column and an end column, the leftmost column in the minimum rectangle, and the rightmost column in the minimum rectangle are defined as **segments**. For example, the number of segments shown in Fig. 4 (a) is four (Fig. 4(c)).

3) *Contours*: The shapes of signatures can be used as features for classification. The upper contour and the lower contour can be used to describe the shape of a signature. The definitions of an upper contour, a lower contour, and an entire contour are given in the following:
   a) *Upper contour*: For every segment, each column is scanned from y_u to y_d, and the first pixel of each column is located. All the first pixels in a segment are connected to produce the *upper contour* (Fig. 4(d)).
   b) *Lower contour*: The definition for a lower contour is the same as for an upper contour except that the scanning direction is reversed, that is from y_d to y_u. Similarly, for each column, the first pixel that belongs to the signature is determined and becomes a lower contour point. All the lower contour points of a segment comprise the *lower contour* (Fig. 4 (e)).
   c) *Entire contour*:The entire contour can be obtained by linking the upper and the lower contours, as shown in Fig. 4 (f).
4) *Profile*: The profile of a segment is determined by transforming a sequence of points, which are sampled from a lower/upper contour, into a special set. In other words, some pixels from the sampling contours are used to approximate the upper and the lower contours.
   a) *Upper profile*: Given a segment, progressing from left to right, we find the first pixel of the upper contour and have its coordinate (x_1^u, y_1^u). Now, we define $x_2^u = x_1^u + \Delta x$, where Δx is a small positive integer. There exists a unique point on this upper contour with x-coordinate x_2^u. Then, its y-coordinate y_2^u is also located. The above steps are repeated to get pixels (x_3^u, y_3^u), (x_4^u, y_4^u), and so on. The relationship among these points is:

$$x_i^u = x_{i-1}^u + \Delta x.$$

If x_i^u is beyond the last pixel of the upper contour, the last pixel of the upper contour is chosen to replace x_i^u. These pixels, (x_k^u, y_k^u), are defined as **sampled pixels**. For two consecutive sampled pixels, (x_{i-1}^u, y_{i-1}^u) and (x_i^u, y_i^u), the y-component difference is defined as follows:

$$U_{i-1,i} = y_{i-1}^u - y_i^u .$$

The set of all $U_{i-1,i}$ is called the **upper profile**.

b) *Lower profile*: The steps followed to generate a lower profile are similar to those for an upper profile. First of all, the first pixel of a lower contour and its coordinates (x_1^l, y_1^l) is determined. Then, we find the sampled pixels (x_2^l, y_2^l), (x_3^l, y_3^l), and so on. The relationship among them is:

$$x_i^l = x_{i-1}^l + \Delta x.$$

We use these sampled pixels from the lower contour to generate $L_{i-1,i}$:

$$L_{i-1,i} = y_{i-1}^l - y_i^l .$$

The set of all $L_{i-1,i}$ is called the **lower profile**.

Fig. 4 (g) shows an example of generating profiles of one segment, and Fig. 4(h) shows the profiles of the signature shown in Fig. 4(a). This method is employed to create database profiles. When a test signature is given, its segment number, upper profile, and lower profile are determined. These profiles are used to find the minimum distance between the input signature and each signature stored in the database. The identification algorithm is described below.

Step 1. Input a testing signature.
Step 2. Find its segments, upper profile and lower profile.
Step 3. Find the signatures from the database that have the same number of segments as the testing signature.
Step 4. For the same segment sequence, find the minimum distance between the test and the candidate's signature segment by segment, and then add them together.
Step 5. Compare all the database profiles found in Step 3 with the test profiles based on the total minimum distances found in Step 4; thus, the test signature can be classified.

An example of signature identification is shown in Fig. 5.

$$\begin{matrix}\text{Upper profile}\\ \text{Lower profile}\end{matrix}\left\{\begin{matrix}(U_{12},U_{23},\cdots,U_{i_1-1,i_1}) & (U_{12},U_{23},\cdots,U_{i_2-1,i_2}) & \cdots & (U_{12},U_{23},\cdots,U_{i_n-1,i_n})\\ \underbrace{(L_{12},L_{23},\cdots,L_{i_1-1,i_1})}_{S_1} & \underbrace{(L_{12},L_{23},\cdots,L_{i_2-1,i_2})}_{S_2} & \cdots & \underbrace{(L_{12},L_{23},\cdots,L_{i_n-1,i_n})}_{S_n}\end{matrix}\right\}$$

(a) The profile form of a signature, where the number of segments in n. For every segment, there are i_n-1 dimension vectors for the upper profile and lower profile.

Fig. 5. An example of using comparison to perform identification.

Database profiles	S_1	S_2
Type 1	(3, 26, -18) (4, 8, -9)	(-4, -16, 5, 3) (2, -1, 8, -13)
Type 2	(-4, -6, 5, -9) (3, 4, 22, -1)	(-3, 2, -12, -3, 6) (6, 2, -4, 11, -7)

Test profiles	S_1	S_2
Unknown type	(-5, 8, -12) (6, 27, 2)	(-9, 2, 8) (-2, 8, -1)

(b) We wish to classify the test signature as type 1 or type 2.

Compared with type 1:	
S_1	$\begin{vmatrix}(3,26,-18)_{\text{Type1}}\\ -(-5,8,-12)_{\text{Test}}\end{vmatrix} = \lvert 3-(-5)\rvert + \lvert 26-8\rvert + \lvert(-18)-(-12)\rvert = 32$, upper profile distance. $\begin{vmatrix}(4,8,-9)_{\text{Type1}}\\ -(6,27,2)_{\text{Test}}\end{vmatrix} = \lvert 4-6\rvert + \lvert 8-27\rvert + \lvert(-9)-2\rvert = 32$, lower profile distance.
	The distance for segment (1) is 32 + 32 = 64.
S_2	$\begin{vmatrix}(-4,-16,5,3)_{\text{Type1}}\\ -(-9,2,8,0)_{\text{Test}}\end{vmatrix} = \lvert(-4)-(-9)\rvert + \lvert(-16)-2\rvert + \lvert 5-8\rvert + \lvert 3\rvert = 25$, upper profile distance. $\begin{vmatrix}(2,-1,8,-13)_{\text{Type1}}\\ -(-2,8,-1,0)_{\text{Test}}\end{vmatrix} = \lvert 2-(-2)\rvert + \lvert(-1)-8\rvert + \lvert 8-(-1)\rvert + \lvert -13\rvert = 35$, lower profile distance.
	$\begin{vmatrix}(-4,-16,5,3)_{\text{Type1}}\\ -(0,-9,2,8)_{\text{Test}}\end{vmatrix} = \lvert -4\rvert + \lvert(-16)-(-9)\rvert + \lvert 5-2\rvert + \lvert 3-8\rvert = 19$, upper profile distance. $\begin{vmatrix}(2,-1,8,-13)_{\text{Type1}}\\ -(0,-2,8,-1)_{\text{Test}}\end{vmatrix} = \lvert 2\rvert + \lvert(-1)-(-2)\rvert + \lvert 8-8\rvert + \lvert(-13)-(-1)\rvert = 15$, lower profile distance.
	The distance for segment (2) is min(60, 34) = 34.

(c) The minimum distance over all the segments between the test signature and the type 1 signature is 64+34=98. For the same process, we find that the minimum distance over all the segments between the test signature and the type 2 signature is 24+34=58. Therefore, we classify this test signature as type 2.

Fig. 5. (Cont'd) An example of using comparison to perform identification.

3.3 Verification Preprocessing

Before verification, the signatures are thinned in order to acquire their skeletons. This is helpful for obtaining the force distributions and the structure of a signature. SThere are many redundant pixels around the boundary of a signature, and the force distributions may be influenced by these redundant pixels. In order to get useful force distributions, a thinning process is employed, and a common thinning algorithm [6] is adopted.

Since the thinning method is very simple, the resulting skeleton might not be very accurate. Hence, an average gray value is adopted in order to get more robust results. The operations for representing the average gray values must be performed in parallel to avoid changing gray values of other pixels during execution.

4. MODIFIED CHAIN CODES

The goal of verification is to determine whether a signature is genuine or not. The major features of any given signature consist of its structure and force distributions. Some studies have only focused on structural features, such as the height/width ratio, principle-axis orientation, handwriting slope, hole and cavity attributes, pixel density distributions and so on. A forger can imitate these features easily, for example, by tracing the original signature, but a force distribution is difficult to imitate.

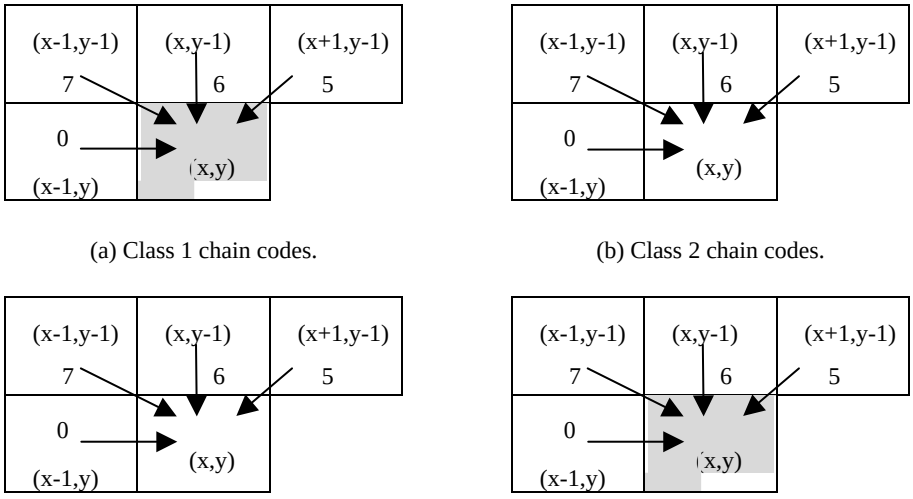
In order to get more local information, we divide the minimum rectangle into several equal parts proceeding from left to right. The experimental environment determines the number of parts. For a three-character Chinese name, three parts are recommended. In each part, the proposed modified chain code is used to perform feature extraction. To solve the problem of obtaining force distributions, we propose using a sheet of carbon paper to detect force distributions, and this method has been proved to be suitable for obtaining signatures with good contrast. The problem of transforming force distributions into features has not yet been solved. An algorithm is needed to effectively determine the direction and the variation of a force distribution. Hence, the chain code combined with gray value variation is used to solve this problem.

Since the images are scanned from left to right and from top to bottom, for a pixel at (x, y), only its left, upper-left, up, and upper-right pixels need to be checked. This can only generate four directions (which are right, down-left, down, and down-right). Referring to the 8-direction chain code, the code words of these four directions are 0, 5, 6, and 7, respectively. Let the gray value of a pixel at coordinate (x, y) be $g(x, y)$. If $g(x, y) \leq g(x', y')$, where the pixel at (x', y') represents anyone of the above four pixels, then the gray level variation is said to be *darkening*. On the other hand, if $g(x, y) > g(x', y')$, then the gray level variation is *lightening*. Suppose that we process a pixel at (x, y). From the gray level variations between $g(x, y)$ and $g(x-1, y)$, $g(x-1, y-1)$, $g(x, y-1)$, $g(x+1, y-1)$, respectively, we can create *class 1 chain codes* and *class 2 chain codes*.

1) *Class 1 chain code*: If the variation of gray levels is *darkening*, then it is defined as a **class 1 chain code** (Fig. 6 (a)).
2) *Class 2 chain code*: If the variation of gray levels is *lightening*, then it is defined as a **class 2 chain code** (Fig. 6(b)).

For a thinned signature (Fig. 7(a)), the number of class 1 and of class 2 chain codes is determined, and the percentage for each class is computed (Fig. 7(b)). These data are useful since they can reduce the influence of false forks. We define the number of pixels between two gray values G_1 and G_2 as N_{G_1, G_2}. For each part, the maximum gray value G_{max} and the minimum gray value G_{min} can be easily located. A threshold value, G_{th}, is determined as follows:

$$\frac{N_{G_{\min}, G_{th}}}{N_{G_{\min}, G_{\max}}} = S, \quad 0 \le S \le 1.$$



(a) Class 1 chain codes.

(b) Class 2 chain codes.

(c) Class 3 chain codes.

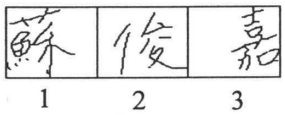
(d) Class 4 chain codes.

Fig. 6. The modified chain codes.

Then, the threshold of each part is used to get a binary image (Fig. 7(c)). If a pixel at (x, y) satisfies $g(x, y) > G_{th}$, then we define it as a **white** pixel. If a pixel at (x, y) satisfies $g(x, y) \le G_{th}$, then we define this pixel as a **black** pixel. Thus, the corresponding binary image can be found. If the parameter S is large, then the threshold G_{th} will be large, and there will be more black pixels in the binary image. If the parameter S is small, then the threshold G_{th} will be small and there will be more white pixels in the binary image. Thus, we can find the ratio of the number of black pixels to the total number of non-background pixels for each part (Fig. 7 (d)). After applying threshold values, the four adjacent scanned pixels are rechecked, and the class 3 and class 4 chain codes are created. Let us assume that we next process the pixel at (x, y), and that pixel (x', y') is one of the four adjacent scanned neighboring pixels.

1) *Class 3 chain code*: If both (*x*, *y*) and (*x'*, *y'*) are black pixels, then the corresponding chain code is defined as a **class 3 chain code** (Fig. 6 (c)).
2) *Class 4 chain code*: If both (*x*, *y*) and (*x'*, *y'*) are white pixels, then the corresponding chain code is defined as a **class 4 chain code** (Fig. 6 (d)).

The number of class 3 and if class 4 chain codes is calculated, and the corresponding percent- ages for each part are computed (Fig. 7 (e)).



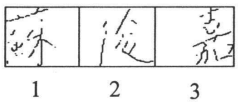
(a) A thinned singnature.

Part(1)	Code 0	Code 5	Code 6	Code 7
Class 1	54 (29.67%)	58 (31.868%)	21 (11.538%)	49 (26.923%)
Class 2	34 (23.943%)	48 (33.802%)	19 (13.38%)	41 (28.873%)

Part(1)	Code 0	Code 5	Code 6	Code 7
Class 1	41 (26.451%)	61 (39.354%)	9 (5.806%)	44 (28.387%)
Class 2	33 (33.333%)	38 (33.383%)	11 (11.111%)	17 (17.171%)

Part(1)	Code 0	Code 5	Code 6	Code 7
Class 1	32 (22.222%)	41 (28.472%)	17 (11.805%)	54 (37.5%)
Class 2	32 (26.666%)	34 (28.333%)	13 (10.833%)	41 (34.166%)

(b) The distribution of class 1 and 2 chain codes.



(c) A binary image.

Fig. 7. Feature extraction for the purpose of verification.

S = 0.75	Part (1)	Part (2)	Part (3)
	52.14%	69.802%	96.153%

(d) The percentage of black pixels in each part.

Part(1)	Code 0	Code 5	Code 6	Code 7
Class 1	38 (25.675%)	54 (36.486%)	16 (10.81%)	40 (27.027%)
Class 2	41 (29.078%)	41 (29.078%)	19 (13.475%)	40 (28.368%)

Part(1)	Code 0	Code 5	Code 6	Code 7
Class 1	47 (27.647%)	68 (40%)	17 (10%)	38 (22.352%)
Class 2	18 (30.508%)	25 (42.372%)	1 (1.694%)	15 (25.423%)

Part(1)	Code 0	Code 5	Code 6	Code 7
Class 1	60 (24%)	72 (28.8%)	29 (11.6%)	89 (35.6%)
Class 2	2 (33.333%)	2 (33.333%)	0 (0%)	2 (33.333%)

(e) The distribution of class 3 and 4 chain codes.

Fig. 7. (Cont'd) Feature extraction for the purpose of verification.

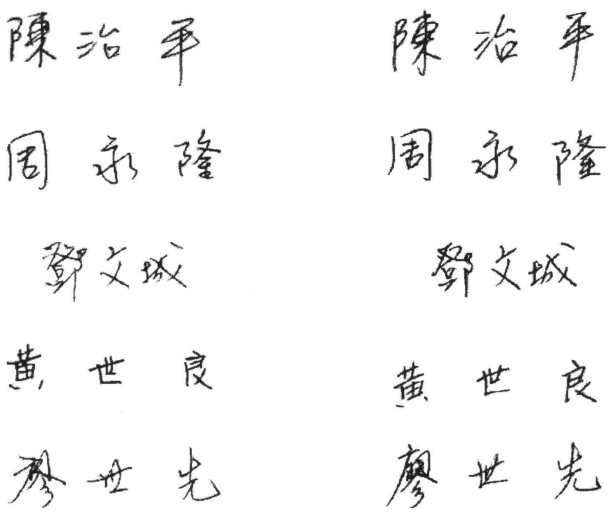
We assign a unique class label to each signature template. In the identification phase, the input signature is classified to an appropriate class no matter whether it is a genuine signature or not. In the verification phase, the modified chain codes are used to extract signature features. These features are used to form an input vector of SFAHN, and the output of SFAHN is a class label. If the output of SFAHN is the same as the label found in the identification phase, then the input signature is regarded as a genuine one. On the other hand, if they are not the same, then the input signature is regarded as a forged one.

5. SIMULATIONS AND RESULTS

5.1 Experiment Environment

In this section, we will evaluate the performance of the proposed signature recognition system. Signatures of 34 people were collected (Fig. 8 (a)). A sheet of graph paper

and a sheet of carbon paper placed underneath were used to obtain each signature. The same genuine signatures were used in both the identification and verification phases to generate a database and to train SFAHN.



(a) Genuine signatures. (b) Forged signatures.

Fig. 8. Genuine and forged signatures.

2,394 genuine signatures were collected. To create the database profiles and to train the neural network, SFAHN, 1,100 signatures were selected, and the remaining 1,294 signatures were used as test signatures. 602 forged signatures were also collected (Fig. 8 (b)) and combined with 1294 genuine test signatures to form a test set.

Three kinds of paper were used, arranged from top to bottom: graph paper, carbon paper, and plain white paper. After each person signed his or her name on the graph paper, his or her carbon signature showing force distributions was obtained on the plain white paper. The carbon signature was scanned and saved as an image file in the typical 256-gray-level mode.

5.2 Experiment Results and Analyses

After constructing the database and training the neural network, we began to test this system. In the identification phase, all 1,294 genuine signatures and 602 forged signatures were tested. For each tested signature, a numerical label was given after it was classified in the identification phase. In the verification phase, we determined whether the signature tested was genuine or not according to the identification label and the verification label. When an unknown signature was inserted, it was first identified, and then the genuineness of this unknown signature was determined.

The most important part of identification is classifying the signature. During the process of identification, if Δx is small, then the dimension of the profile vectors is high. However, this will lead to misidentification of genuine signatures. In other words, the tested signature must look more or less like the genuine signature. For three different Δx values, the results are shown in Table 1. Most of the genuine signatures were misidentified due to their small sizes. Since these small signatures had lower-dimension profile vectors, it was easy to find the best match position, and then the overall distance become smaller. When $\Delta x = 1$ or $\Delta x > 4$, the performance worsened.

Table 1. The number of misidentified signatures.

	$\Delta x = 2$	$\Delta x = 3$	$\Delta x = 4$
Genuine	28	26	30
Forgery	37	32	35

In the identification phase, the tested signature was assigned to a certain class. Whether this signature was genuine or not was determined during the verification phase. For every tested signature, its features were extracted and verified using SFAHN. After the inner operation, SFAHN generated a label that indicated whether the input signature really belonged to a designated person or not. If the label given in the identification phase did not match the output result from SFAHN, then this input signature was regarded as forged.

An error-rate definition was adopted from [5]. There are two conditions that will lead to incorrect verification:

Type I: Genuine signatures are determined to be forged.
Type II: Forged signatures are determined to be genuine.

Therefore, the total error rate is defined as the ratio of the number of incorrectly verified signatures to the total number of all test signatures. The threshold parameter (S) and vigilance parameter (ρ) can be adjusted to get different total error rates. The analysis results for ρ and S are given in Table 2 and Table 3, respectively.

In this present system, the best total error rate is 10.495% ($S = 0.75$ and $\rho = 0.95$). The vigilance parameter ρ is used to measure the degree of similarity. If ρ is large, then the test signature must look very like the genuine signature, or it will be verified as a forged signature. This will cause the number of type I errors to increase. On the other hand, if ρ is small, then the number of type II errors will increase. A forger usually does not clearly understand the force strength variations although he often does know quite well where he should use more (or less) force. In other words, the difference of G_{max} and G_{min} may be the same, but the gray level distributions of pixels are different between these two values. The threshold G_{th} is used to make these different distributions reveal on white/black pixels and class 3/class 4 chain codes. Thus, the parameter S is adjusted to find the most effective value of G_{th} for distinguishing genuine signatures from forged signatures.

Table 2. The error rate with $\Delta x = 3$ and $S = 0.6$, 0.65 and 0.7.

S =0.6	*Vigilance* (ρ)		
	0.94	0.95	0.96
Type Ⅰ	59/1294	81/1294	190/1294
Type Ⅱ	205/602	169/602	131/602
Total error rate	13.924%	13.185%	16.930%

S =0.65	*Vigilance* (ρ)		
	0.94	0.95	0.96
Type Ⅰ	64/1294	83/1294	124/1294
Type Ⅱ	235/602	161/602	112/602
Total error rate	15.770%	12.869%	12.447%

S =0.7	*Vigilance* (ρ)		
	0.94	0.95	0.96
Type Ⅰ	57/1294	82/1294	217/1294
Type Ⅱ	229/602	172/602	96/602
Total error rate	15.084%	13.296%	16.508%

If the minimum rectangle is not divided into several small parts, then it may generate features that are not robust enough. Hence, the total error rate will not decrease. The minimum rectangle is divided into six parts (2×3) in order to get more local features. However, the result shows an increase in the number of type I and type II errors. If it is divided into too many small parts, it may extract features that are very local. Hence, these features cannot respond to the characteristics of signatures and, therefore, will not help in distinguishing genuine signatures from forged signatures.

6. CONCLUSIONS

In this study, a sheet of carbon paper was used to detect the force distributions in signatures, and an off-line signature recognition system was employed. The proposed system performs identification and verification. For the purpose of identification, we use contours and profiles to classify signatures. This involves in implementing a real automatic signature recognition system (ASRS). In most of the reported signature recognition systems, the identification phase is ignored. If there are signatures whose shapes are very irregular, then systems which do not perform identification can be used to determine whether these input signatures are genuine or forged but cannot determine what classes unknown signatures belong to. Carbon paper is widely used to duplicate what we write or

Table 3. The error rate with $\Delta x = 3$ and $S = 0.75$, 0.8 and 0.85.

S =0.75	*Vigilance* (ρ)		
	0.94	0.95	0.96
Type Ⅰ	70/1294	102/1294	251/1294
Type Ⅱ	231/602	97/602	84/602
Total error rate	15.875%	10.495%	17.668%

S =0.8	*Vigilance* (ρ)		
	0.94	0.95	0.96
Type Ⅰ	64/1294	121/1294	389/1294
Type Ⅱ	227/602	102/602	67/602
Total error rate	15.348%	11.761%	24.050%

S =0.85	*Vigilance* (ρ)		
	0.94	0.95	0.96
Type Ⅰ	75/1294	132/1294	354/1294
Type Ⅱ	209/602	142/602	89/602
Total error rate	14.978%	14.451%	23.364%

type, and it is easy to find some applications in daily life for the algorithm we have proposed in this paper. For example, when there is an argument over a credit card statement, the algorithm can be applied to the bank copy that is generated by a carbon copy. By using this method, a forgery signature can be easily found out.

Usually, an off-line system does not have any special features to extract. Hence, most of the off-line systems focus on signature structures. Since signature structures can be imitated by means of tracing, a very strict and accurate verification algorithm is required. However, it is impossible for people to write their signatures in exactly the same manner every time, so a very strict system focusing on signature structure is not practical. Therefore, a sheet of carbon paper is used in our method to detect force distributions that indeed have useful features. The proposed modified chain code includes the structure and force distribution features, thus making it easy to identify and trace forgeries. Also, a new neural network model, the supervised fuzzy adaptive Hamming network (SFAHN), is adopted as the feature verifier due to its ability to learn new classes and input patterns, and to refine existing classes without destroying previously learned patterns. SFAHN not only exhibits good performance, but also offers fast verification.

The proposed system can perform both identification and verification of signatures. However, it still cannot satisfy the requirement of real ASRS specifications. In the case of a slanted signature, the present modified chain code will become invalid, since the chain codes will be different from the ones in the database.

Further studies will improve the performance of our signature recognition in many ways. One avenue of research will be to explore the unused features hidden in the internal contours of signatures, besides the upper and the lower contours. This will result in more complicated, accurate, and robust contour features. Thus, we can achieve the goal of reducing the type I error rate while allowing more (all) genuine signatures to pass the identification phase. However, a more complex feature extraction and comparison algorithm is needed. We can also use some transforms, like Fourier transform and the wavelet transform, to get special features in the frequency domain. With the aid of these special features, we should be able to improve the performance of this system. In order to get more robust local features, we can divide up the minimum rectangle according to the segments found in the identification phase. However, this will generate unfixed-dimension vectors that will make the neural network difficult to design unless each neural network is trained according to different numbers of segments. This will require more operations and preprocessing, and will be time consuming. Furthermore, a local or global histogram can be used to analyze force distributions. However, it should be noted that analyzing very basic local features might be of no value.

REFERENCES

1. M. Ammar, Y. Yoshida, and T. Fukumura, "Off-line preprocessing and verification of signatures," *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 2, 1988, pp. 589-602.
2. G. A. Carpenter and W. D. Ross, "ART-EMAP: A neural network architecture for object recognition by evidence accumulation," *IEEE Transactions on Neural Networks*, Vol. 6, 1995, pp. 805-818.
3. J. M. Chen, J. A. Ventura, and C. H. Wu, "Segmentation of planar curves into circular arcs and line segments," *Image and Vision Computing 14*, 1996, pp. 71-83.
4. G. Dimauro, S. Impedovo, and G. Pirlo, "Component-oriented algorithms for signature verification," *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 8, 1994, pp. 771-793.
5. J. P. Drouhard, R. Sabourin, and M. Godbout, "A neural network approach to off-line signature verification using directional PDF," *Pattern Recognition*, Vol. 29, 1996, pp. 415-424.
6. R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Addison Wesley, Reading, Mass, 1992.
7. V. Govindaraju and R. K. Krishnamurthy, "Holistic handwritten word recognition using temporal features derived from off-line images," *Pattern Recognition Letters 17*, 1996, pp. 537-540.
8. L. Gupta, T. Sortrakul, A. Charles, and P. Kisatsky, "Robust automatic target recognition using a localized boundary representation," *Pattern Recognition*, Vol. 28, 1995, pp. 1587-1598.
9. K. Han and I. K. Sethi, "Handwritten signature retrieval and identification," *Pattern Recognition Letters 17*, 1996, pp. 83-90.
10. R. M. Haralick, S. R. Sternberg, and X. Zhuang, "Image analysis using mathematical morphology," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, 1987, pp. 532-550.

11. N. M. Herbst and C. N. Liu, "Automatic signature verification based on accelerometry," *IBM Journal Research and Development*, 1997, pp. 245-253.
12. C. A. Hung and S. F. Lin, "Adaptive hamming net: A fast-learning ART model without searching," *Neural Networks*, Vol. 8, 1995, pp. 605-618.
13. C. A. Hung and S. F. Lin, "Supervised adaptive hamming net for classification of multiple valued patterns," *International Journal of Neural Systems*, Vol. 8, 1997, pp. 181-200.
14. S. Lee and Jack C. Pan," Offline tracing and representation of signatures," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 22, 1992, pp. 755-771.
15. C. W. Liao and J. S. Hung, "Stroke segmentation by Bernstein-Bezier curve fitting," *Pattern Recognition*, Vol. 23, 1990, pp. 475-484.
16. C. Parisse, "Global word shape processing in off-line recognition of handwriting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, 1996, pp. 460-464.
17. I. Pottier and G. Burel, "Identification and authentication of handwritten signatures with a connectionist approach," in *Proceedings of IEEE International Conference on Neural Networks*, 1994, pp. 2984-2951.
18. Y. Qi and B. R. Hunt, "A multi-resolution approach to computer verification of handwritten signatures," *IEEE Transactions on Image Processing*, Vol. 4, 1995, pp. 870-874.
19. J. L. McClelland, D. E. Rumelhart, and the PDP Research Group, *Parallel Distributed Processing*, MIT Press, Cambridge, Vol. 1, 1986.
20. R. Sabourin, M. Cheriet, and G. Genest, "An extended-shadow-code based approach for off-line signature verification," in *Proceedings of the Second International Conference on Document Analysis and Recognition*, 1993, pp. 1-5.
21. N. G. See and O. H. Seng, "A neural network approach for off-line signature verification," in *Proceedings of IEEE International Conference on Computer, Communication, Control and Power Engineering*, 1993, pp. 770-773.
22. J. W. Tai, Y. J. Liu, and L. Q. Zhang, "A model based detecting approach for feature extraction of off-line handwritten Chinese character recognition," in *Proceedings of the Second International Conference on Document Analysis and Recognition*, 1993, pp. 826-829.
23. L. Y. Tseng and T. H. Huang, "An online Chinese signature verification scheme based on the ART1 neural network," *International Joint Conference on Neural Networks*, 1992, pp. III-624-III-630.
24. M. J. Wang, W. Y. Wu, L. K. Hung, and D. M. Wang, "Corner detection using bending value," *Pattern Recognition Letters 16*, 1995, pp. 575-583.
25. Y. Xuhua, T. Furuhashi, K. Obata, and Y. Uchikawa, "Constructing a high performance signature verification system using a GA method," in *Proceedings of International Two-Stream Conference on Artificial Neural Networks and Expert Systems*, 1995, pp. 170-173.
26. I. Yoshimura and M. Yoshimura, "Off-line writer verification using ordinary character as the object," *Pattern Recognition*, Vol. 24, 1991, pp. 909-915.
27. R. W. Zhou, C. Quek, and G. S. Ng, "A novel single-pass thinning algorithm and an effective set of performance criteria," *Pattern Recognition Letters 16*, 1995, pp. 1267-1275.

28. K. P. Zimmermann and M. J. Varady, "Handwritten identification from one-bit quantilized pressure patterns," *Pattern Recognition*, Vol. 18, 1985, pp. 63-72.
29. H. H. Chang and H. Yan, "Analysis of stroke structures of handwritten Chinese characters," *IEEE Transactions on System, Man and Cybernetics*, Vol. 29, 1999, pp. 47-61.
30. J. J. Zou and H. Yan, "Extracting strokes from static line images based on selective searching," *Pattern Recognition*, Vol. 32, 1999, pp. 935-946.

Sheng-Fuu Lin (林昇甫) was born in Tainan, the Republic of China, in 1954. He received the B.S. and M.S. degrees in mathematics from National Taiwan Normal University in 1976 and 1979, respectively, the M.S. degree in computer science from the University of Maryland in 1985, and the Ph.D. degree in electrical engineering from the University of Illinois, Champaign, in 1988.

Since 1988, he has been on the faculty of the Department of Electrical and Control Engineering at National Chiao Tung University, Hsinchu, Taiwan, where he is currently an associate professor.

His research interests include fuzz theory, automatic target recognition, scheduling, image processing, and image recognition. Dr. Lin is a member of the IEEE Control Society, Chinese Fuzzy System Association, and Chinese Automatic Control Society.

Yu-Wei Chang (張育維) received his B.S. degree from the Electrical Engineering Department at TamKang University, Taipei, Taiwan, in 1995, and he received his M.S. degree from the Electrical and Control Engineering Department at National Chiao Tung University, Hsinchu, Taiwan, in 1997. His research interests include document analysis, pattern recognition, fuzzy theory, and artificial neural networks.

Chien-Kun Su (蘇建焜) received his B.S. degree from National Taiwan University in1989, and he received the M.S. degree from the University of Southern California in 1992. Currently, he is a graduate student of the Ph.D. program of the Electrical and Control Engineering Department at National Chiao Tung University. Mr. Su is also an instructor of the Electrical Engineering Department of Chung Hua University in Hsinchu Taiwan. His research interests include image processing and 3D image reconstruction.