

On-line Handwritten Alphanumeric Character Recognition

Using Dominant Points in Strokes

Xiaolin Li and Dit-Yan Yeung

Department of Computer Science

Hong Kong University of Science and Technology

Clear Water Bay, Kowloon

HONG KONG

Internet: {xiaolin,dyyeung}@cs.ust.hk

Fax: +852-2358-1477

August 23, 1996

Abstract

All alphanumeric characters can be written in certain styles with strokes of different shapes and positions. An on-line handwritten character written on a digitizing tablet is represented as a sequence of strokes, which are the loci of the pen tip from its pen-down to pen-up positions. In this paper, we present an approach to on-line handwritten alphanumeric character recognition based on sequential handwriting signals. In our approach, an on-line handwritten character is characterized by a sequence of dominant points in strokes and a sequence of writing directions between consecutive dominant points. The directional information of the dominant points is used for character preclassification and the positional information is used for fine classification. Both preclassification and fine classification are based on dynamic programming matching using the idea of band-limited time warping. These techniques are elastic, in that they can tolerate local variation and deformation. The issue of reference (or template) set evolution is also addressed. A recognition experiment has been conducted with 62 character classes (0-9, A-Z, a-z) of different writing styles (Italian manuscript style and some other styles) and 21 people as data contributors. The recognition rate of this experiment is 91%, with 7.9% substitution rate and 1.1% rejection rate. The average processing time is 0.35 second per character on a 486 50MHz personal computer.

Keywords

Handwritten stroke, dominant point, direction primitive, preclassification, fine classification, time warping, dynamic programming

1 Introduction

With the development of digitizing tablets and microcomputers, on-line handwriting recognition has become an area of active research since the 1960s⁽¹⁾. One reason for this is that on-line handwriting recognition promises to provide a dynamic means of communication with computers through a pen-like stylus, not just a keyboard. This seems to be a more natural way of entering data into computers.

On-line recognition refers to the recognition mode in which the machine recognizes the handwriting while the user writes on the surface of a digitizing tablet with an electronic pen. The digitizing tablet captures the dynamic information about handwriting, such as number of strokes, stroke order, writing speed, etc., all in real time. Off-line recognition, by contrast, is performed after the handwriting has been completed and its image has been scanned in. Thus, dynamic information is no longer available.

Another advantage of on-line recognition over off-line recognition is that there is close interaction between the user and the machine. The user can thus correct any recognition error immediately when it occurs.

The problem of on-line handwriting recognition can be defined in various ways. Variables in the problem definition include character set, writing style, and desired recognition rate. In general, each problem definition lends itself to different algorithmic approaches, which in turn make use of different features for classification.

With respect to alphanumeric character recognition, it seems that the most discriminating features come from their shape. This has led many researchers in the field of handwritten

alphanumeric character recognition to use geometrical and topological features^(2,3,4), statistical features⁽⁵⁾, and other algorithms^(6,7,8) to perform recognition based on character shape.

In recent years, some researchers have made a lot of effort in modeling pen-tip movement^(9,10). In the handwriting generation model⁽¹⁰⁾, for example, the movement of the pen tip is described as a velocity vector controlled by two synergies: one is the curvilinear velocity and the other is the angular velocity. Such a model gives us a better understanding of handwriting generation.

In this paper, we present an approach to on-line handwritten alphanumeric character recognition based on sequential handwriting signals. The recognition process consists of the following stages (Figure 1):

1. Data preprocessing
2. Feature extraction
3. Character classification
4. Reference set evolution

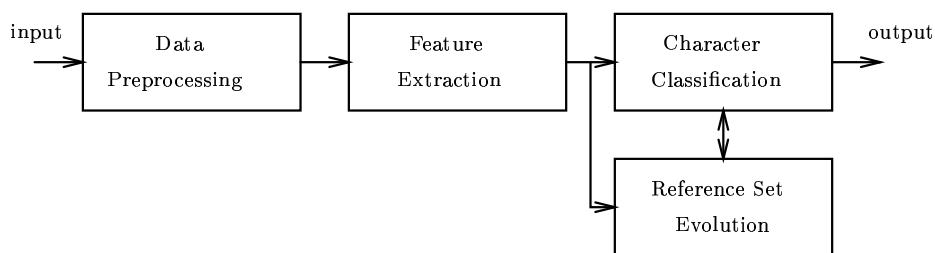


Figure 1: Different stages of the recognition process

Feature extraction is a very crucial step, as the success of a recognition system is often

attributed to a good feature extraction method. The main idea here is to come up with an efficient and reliable method for extracting features from the sequential handwriting signals. In particular, points in strokes corresponding to local extrema of curvature are detected. These points correspond to the maxima of angular velocity or the minima of curvilinear velocity of the pen-tip movement in fluent writing^(9,10). In addition, the mid-point between two consecutive points that correspond to curvature extrema or pen-down/pen-up locations is also used to represent a local minimum of angular velocity (or a local maximum of curvilinear velocity). We refer to all these points as dominant points in strokes. Figure 2 illustrates this concept, where the dominant points are indicated by circles.

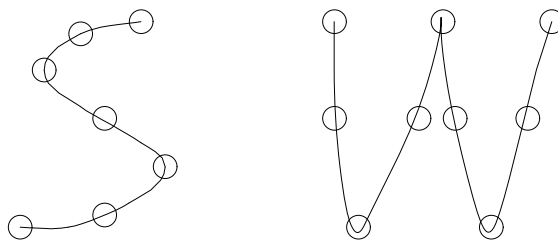


Figure 2: Dominant points in strokes

Unlike topological and statistical features such as moments, dominant points are perceptibly significant and are useful for generating rich structural descriptions. In our system, an on-line handwritten character is characterized by a sequence of dominant points in strokes and a sequence of writing directions between consecutive dominant points. The directional information of the dominant points is used for character preclassification and the positional information is used for fine classification. Both preclassification and fine classification are based on dynamic programming matching using the idea of band-limited time warping.

An expected advantage of this approach over other methods comes from the fact that our character matching processes are elastic and hence can tolerate local variation and deformation. Besides, dominant points are quite easy to extract using the technique described in Reference 11. Moreover, our approach can handle large alphabets (such as Chinese characters) due to its fast preclassification step. However, since our approach is based on sequential handwriting signals, it is intrinsically stroke-order dependent unless multiple models corresponding to different stroke orders are used.

The remainder of this paper is divided into five sections. Section 2 describes data preprocessing and feature extraction. Section 3 presents our preclassification and fine classification algorithms. Section 4 addresses the problem of reference set evolution. Section 5 presents the experimental results using our approach. Finally, Section 6 gives some concluding remarks.

2 Data Preprocessing and Feature Extraction

2.1 Data Preprocessing

In the current context, a handwritten stroke refers to the locus of the pen tip from its pen-down to the next pen-up position. It can, therefore, be described as a sequence of consecutive points on the x - y plane:

$$S = p_1 p_2 \cdots p_L \tag{1}$$

where $p_l = (x_l, y_l)$, $1 \leq l \leq L$, p_1 is the pen-down point, p_L is the pen-up point, and L is the number of points in the stroke. Based on this representation, a handwritten character can then

be described as a sequence of strokes

$$C = S_1 S_2 \cdots S_N \quad (2)$$

where N is the number of strokes in the character.

Since handwritten characters often have large variations in size and position, it is necessary to normalize the input data to facilitate subsequent processing. In our system, size normalization is performed by scaling each character both horizontally and vertically:

$$x_i = \frac{x_i^o - x_{min}}{x_{max} - x_{min}} W \quad (3)$$

$$y_i = \frac{y_i^o - y_{min}}{y_{max} - y_{min}} H \quad (4)$$

where (x_i^o, y_i^o) denotes the original point, (x_i, y_i) is the corresponding point after transformation, $x_{min} = \min_i\{x_i^o\}$, $y_{min} = \min_i\{y_i^o\}$, $x_{max} = \max_i\{x_i^o\}$, $y_{max} = \max_i\{y_i^o\}$, W and H are the width and height of the normalized character, respectively.

Following this normalization step, stroke smoothing and linear interpolation are performed so that the point sequence representing a stroke satisfies the following relations:

$$p_{l+1} \neq p_l \quad (5)$$

$$|x_{l+1} - x_l| \leq 1 \quad (6)$$

$$|y_{l+1} - y_l| \leq 1 \quad (7)$$

We impose the above conditions on each stroke so that its eight-neighbor chain code is:

$$D = d_1 d_2 \cdots d_{L-1} \quad (8)$$

where $d_l \in \{E, SE, S, SW, W, NW, N, NE\}$, $1 \leq l \leq L - 1$, describes the direction from a point to the next one. These directions are shown in Figure 3(a) and are coded as 0 to 7. Due to

the interpolation process, d_l can be determined simply from the coordinate offsets as shown in Table 1. An example of stroke chain code is given in Figure 3(b) where the stroke has been interpolated. Its chain code is 43221007.

d_l	0	1	2	3	4	5	6	7
Δx	1	1	0	-1	-1	-1	0	1
Δy	0	1	1	1	0	-1	-1	-1

Table 1: Lookup table for direction codes

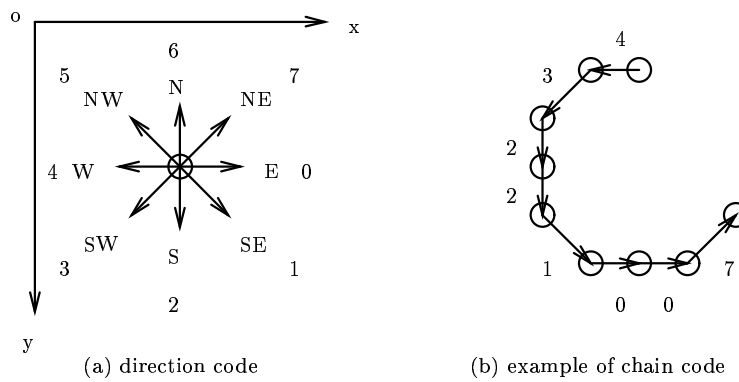


Figure 3: Stroke chain code

2.2 Feature Extraction

2.2.1 Feature Types

The stroke-based features used in our system are:

- Dominant points in strokes:

Dominant points refer to points of the following types: (a) pen-down and pen-up points; (b) points corresponding to local extrema of curvature; and (c) mid-points between two consecutive points of type (a) or (b).

- Direction primitives between dominant points:

A direction primitive refers to one of the eight chain-code directions: E, SE, S, SW, W, NW, N, and NE (see Figure 3(a)). It represents the writing direction from a dominant point to the next one.

2.2.2 Dominant Points

Based on the chain coding scheme, consecutive *exterior angles* and *contour angles* formed by pairs of arrows along the stroke can be defined as shown in Figure 4. In Figure 4, the exterior angle a_i at point p_i is formed by the pair of arrows, d_{i-1} and d_i , and located on the left-hand side of the arrows. The value of a_i can be obtained by looking up Table 2. Denoting the sequence of exterior angles in a stroke as

$$A = a_2 a_3 \cdots a_{L-1} \quad (9)$$

and performing low-pass filtering on A , one can segment the stroke into a sequence of convex/concave/plain regions.

$(d_{i-1} - d_i) \bmod 8$	0	1	2	3	4	5	6	7
a_i	180°	135°	90°	45°	0°/360°	315°	270°	225°

Table 2: Lookup table for exterior angles

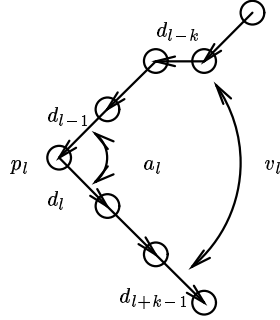


Figure 4: Exterior angle and contour angle

The contour angle v_l at p_l is defined within a support region and its value is estimated as

$$v_l = \frac{1}{K} \sum_{k=1}^K a_{lk} \quad (10)$$

where for $k = 1, 2, \dots, K$, angle a_{lk} is formed by the pair of arrows d_{l-k} and d_{l+k-1} . The value of a_{lk} can also be obtained through table lookup as before. The contour angle v_l is a good estimate of the curvature at p_l . Denoting the sequence of contour angles in the stroke as

$$V = v_2 v_3 \cdots v_{L-1} \quad (11)$$

one can easily obtain the maximum within a convex region and the minimum within a concave region. All such maxima and minima constitute the local extrema of curvature along a stroke. More details of the above technique can be found in Reference 11.

After detecting the extreme points, a mid-point between two consecutive points of type (a) (i.e. pen-down and pen-up points) or (b) (i.e. local extrema of curvature) along a stroke is then located to approximate the point of local minimum in angular velocity. The pen-down and pen-up points, the local extrema of curvature and these mid-points together then constitute the dominant points in a stroke.

2.2.3 Direction Primitives

Based on the dominant points extracted, a direction primitive can be defined as a vector from a dominant point to the following one, after quantization into one of the eight directions E, SE, S, SW, W, NW, N, and NE.

2.2.4 Feature Sequences

After feature extraction, a character C can be represented as a sequence P of dominant points and a sequence D of direction primitives:

$$P = p_1 p_2 \cdots p_M \quad (12)$$

$$D = d_1 d_2 \cdots d_{M-1} \quad (13)$$

where p_m is a dominant point, and $d_m \in \{E, SE, S, SW, W, NW, N, NE\}$ is the (quantized) direction from p_m to p_{m+1} .

2.3 Other Considerations

There are a few other special considerations in our system design.

1. Recognition of different character sets:

Our system aims at recognizing handwritten characters from different character sets, including Chinese characters¹, English letters, and Arabic numerals. Some English letters

¹Our system processes handwritten Chinese characters in a similar manner. However, Chinese character recognition is beyond the scope of this paper.

and Arabic numerals look very similar to each other (e.g., ‘o’ and ‘0’, ‘s’ and ‘5’), making it very difficult, if not impossible, to resolve such ambiguities without using context information. In our system, we designate different writing areas for different character sets.

2. Recognition of ascenders and descenders:

Furthermore, since ascenders (“poles”) and descenders (“legs”) are very useful for discriminating between some English letters having a similar shape (e.g., ‘h’ and ‘n’), ledger lines (defining three ledger regions) are used to facilitate character classification for English handwriting. In our system, the ascender-descender property of an English letter is determined by the position of its first stroke. If the upper portion of this stroke is above the mid-level of the top ledger region, it is considered to be a letter with ascender. On the other hand, if the lower portion of the stroke is below the mid-level of the bottom ledger region, it is considered to be a letter with descender. Otherwise, the letter has neither an ascender nor a descender.

3. Use of pseudo extrema of curvature:

Since the curvature along a perfect circle is constant, theoretically no local extrema of curvature can be detected along the circle. This may lead to misclassification if a large portion of a stroke is part of a circle or a very smooth arc. When this happens, we identify the mid-point of that region as a pseudo extreme of curvature. For any region defined by two dominant points of type (a) or (b), we consider both the length of the arc and the ratio of the arc length to the chord length. If both measures exceed their respective thresholds, its mid-point will be regarded as a pseudo extreme of curvature.

2.4 An Example

Figure 5 shows an example of the letter ‘B’.² The original handwriting with two “inked” strokes is located in the designated writing area for English letters. The “inking” process is simulated simply by connecting consecutive points in the pen-tip trace with line segments. The discrete version of the pen-tip trace sampled by the digitizing tablet is re-displayed in the top-right region. As soon as the writing is completed, the input character is normalized in size, with the data points of each stroke interpolated and chain coded. Local extrema of curvature in each stroke are then detected from the chain code at the pixel level. After that, mid-points between every two consecutive local extrema or pen-down/pen-up points are located. All these points are dominant points in strokes. The direction between every two consecutive dominant points is then quantized into one of the eight categories, forming a sequence of direction primitives for each stroke.

In the lower region of Figure 5, the two preprocessed strokes of the letter ‘B’ are displayed, with the local extrema of curvature marked. Each stroke is also characterized by a sequence of direction primitives. The first stroke has three dominant points (no curvature extrema) with sequence ‘22’. The second stroke has nine dominant points (three extreme points) with sequence ‘01340134’. Since this character is written in the area designated for (lowercase or uppercase) English letters and the upper portion of the first stroke is above the mid-level of the top ledger region, it is categorized as an English letter with ascender.

²The generic user interface shown in Figure 5 and some later figures is intended for a more general system which uses more or less the same approach as described in this paper for the recognition of Chinese characters as well. Readers may ignore the Chinese character box in these figures since Chinese character recognition is beyond the scope of this paper.

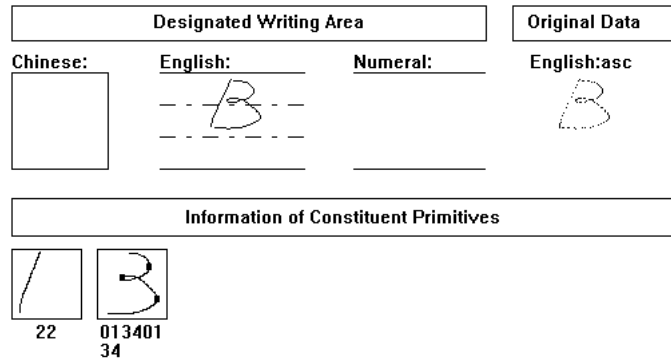


Figure 5: Example of data preprocessing and feature extraction

3 Character Classification

Our classification stage is divided into two steps. First, candidate selection is performed during the preclassification step to reduce the number of candidate classes that need to be investigated further in detail. Second, fine classification is performed on the candidate classes that survive the preclassification step to finally classify an input character. Both the preclassification and fine classification steps perform matching using dynamic programming.

3.1 Dynamic Programming for Elastic Matching

3.1.1 Band-limited Time Warping

Time warping is a useful technique for finding the correspondence between two strings (sequences) which may be distorted^(12,13). In dealing with such distortions, each symbol in one string can be mapped to one or more symbols in the other string, provided that the sequential order is preserved.

Given two strings, many time warps are possible. A cost (or gain) function can be defined to evaluate each warp. If time warping is intuitively seen as stretching or shrinking one string (nonlinearly) to make it look more similar to another string, then the best warp corresponds to the best compromise between minimizing the stretching and shrinking costs and maximizing the similarity between the corresponding symbols in the two strings. A similarity or dissimilarity (i.e. distance) measure can be defined between two strings based on the best warp.

For a string of length m and another string of length n , the best warp between them can be found out using dynamic programming, and the matching process is usually represented by an $m \times n$ graph (matrix) with three basic operations (i.e. compression, expansion, and substitution) associated with each node^(13,14). The computational complexity is thus $O(mn)$. This could be time-consuming if the basic operations form complex distortions. However, if we assume that there are only local distortions (i.e. variation and deformation of handwriting in our case) within each string, we can limit the extent of possible positional shift of each symbol. This results in a reduction in the number of possible warps that need to be investigated using dynamic programming. As a result, the matching graph is reduced to a sub-diagonal one, and hence the computational complexity is also reduced accordingly. The sub-diagonal range characterized by an integer indicates the extent of possible positional shift of symbols during the matching process (see Figure 8). This kind of time warping is referred to here as band-limited time warping. Similar ideas have been suggested by Tappert *et al.*⁽¹⁾, who use lines of slopes 1/2 and 2 to represent the limits of compression and expansion in stroke-by-stroke elastic matching.

3.1.2 Dynamic Programming Algorithm

Dynamic programming is a useful technique for finding the shortest path from one node to another in a graph. It has also been used successfully for string matching^(14,15). In our system, we use a dynamic programming approach to perform elastic matching of feature sequences.

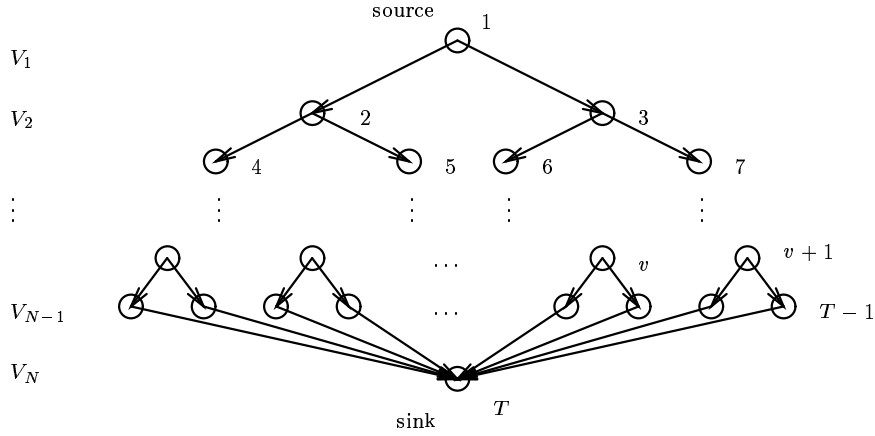


Figure 6: Dynamic programming matching

Suppose G is a matching graph with nodes in N levels, as shown in Figure 6. Each node in the graph is associated with an index and a cost value. Let V_n be the set of nodes at level n , $1 \leq n \leq N$. The singleton sets $V_1 = \{1\}$ and $V_N = \{T\}$ are for the source and sink nodes, respectively. Starting from the source there are a number of paths leading to the sink, with each path associated with a total cost value. Denoting the cost value of node v as c_v and the minimum partial cost (over all possible partial paths) from a node $u \in V_n$ to the sink V_N as $C_n(u)$, the minimum total cost from the source to the sink can then be determined as follows:

1. Initialization step:

$$C_N(T) = c_T \quad (14)$$

2. Induction step:

$$\forall u \in V_n, n = N - 1, N - 2, \dots, 1$$

$$C_n(u) = c_u + \min_{\langle u,v \rangle \in G} \{C_{n+1}(v)\} \quad (15)$$

where $\langle u, v \rangle$ denotes an edge in G .

The algorithm for dynamic programming based on a gain measure can be derived in a similar manner.

3.2 Preclassification

The preclassification step is based on finding the maximum gain of band-limited time warping to define the similarity between two strings. The maximum gain is found by dynamic programming.

3.2.1 Similarity Between Two Sequences

Let D_I and D_R be the sequences of direction primitives of an input character C_I and a reference character C_R , respectively:

$$D_I = d_{i_1} d_{i_2} \cdots d_{i_{M-1}} \quad (16)$$

$$D_R = d_{r_1} d_{r_2} \cdots d_{r_{N-1}} \quad (17)$$

Without loss of generality, assume that $M \leq N$. Using the idea of band-limited time warping, one can construct a matching graph $G_s(D_I, D_R)$ with nodes in $N+1$ levels, as shown in Figure 7.

As shown in the figure, a direction primitive in D_R is allowed to have only one match in D_I , but a direction primitive in D_I may have multiple matches in D_R . Furthermore, virtual source

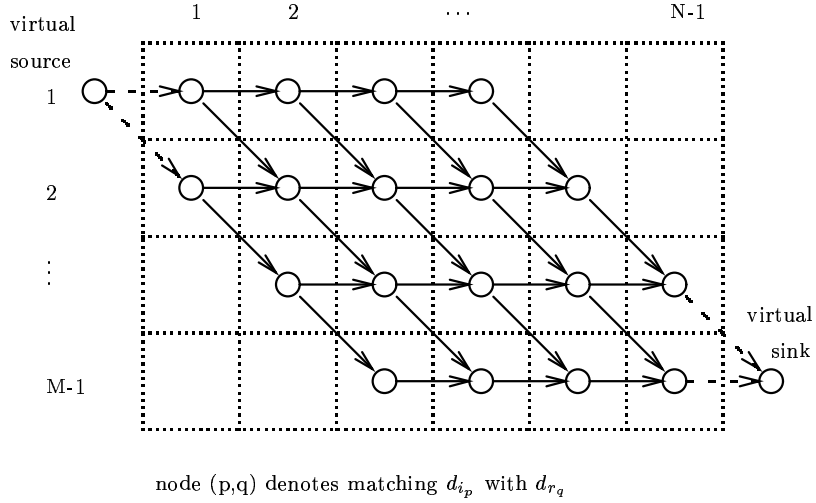


Figure 7: Matching graph $G_s(D_I, D_R)$

and sink nodes are introduced so that matching of the two ends is not enforced. The gain at a node (p, q) can be interpreted as the similarity between d_{i_p} and d_{r_q} , which is defined in Table 3. The gains at the source and the sink are set to zero.

The maximum total gain G_{max} from the source to the sink can be found by dynamic programming as discussed above. The similarity between D_I and D_R is then defined as

$$S(D_I, D_R) = \frac{G_{max}}{N-1} \quad (18)$$

if both D_I and D_R are non-empty. It is obvious that $0 \leq S(D_I, D_R) \leq 1$.

3.2.2 Candidate Classes

Suppose C_I and C_R belong to the same character set and have the same ascender-descender property if the character set is the set of English letters. C_R is said to be a candidate class for

	E	SE	S	SW	W	NW	N	NE
E	1	0.6	0	0	0	0	0	0.6
SE	0.6	1	0.6	0	0	0	0	0
S	0	0.6	1	0.6	0	0	0	0
SW	0	0	0.6	1	0.6	0	0	0
W	0	0	0	0.6	1	0.6	0	0
NW	0	0	0	0	0.6	1	0.6	0
N	0	0	0	0	0	0.6	1	0.6
NE	0.6	0	0	0	0	0	0.6	1

Table 3: Similarity between direction primitives

C_I if

$$S(D_I, D_R) \geq T_S \quad (19)$$

where T_S is a threshold.

Based on the above rule, we can select a (limited) number of candidate classes for subsequent fine classification. In the case when no candidate classes can be selected, the input character will be rejected because it is sufficiently different from all the character patterns in the current reference set. (The issue of modifying or evolving the current reference set will be addressed in a later section.)

3.3 Fine Classification

The fine classification step is based on finding the minimum cost of band-limited time warping to define the distance (i.e. dissimilarity) between two strings. The minimum cost is found by dynamic programming.

3.3.1 Distance Between Two Sequences

Let P_I and P_R be the sequences of dominant points of the input character C_I and a reference character C_R , respectively:

$$P_I = p_{i_1} p_{i_2} \cdots p_{i_M} \quad (20)$$

$$P_R = p_{r_1} p_{r_2} \cdots p_{r_N} \quad (21)$$

Using the idea of band-limited time warping, a matching graph $G_d(P_I, P_R)$ with nodes in N levels can be constructed (Figure 8), similar to that for the preclassification step.

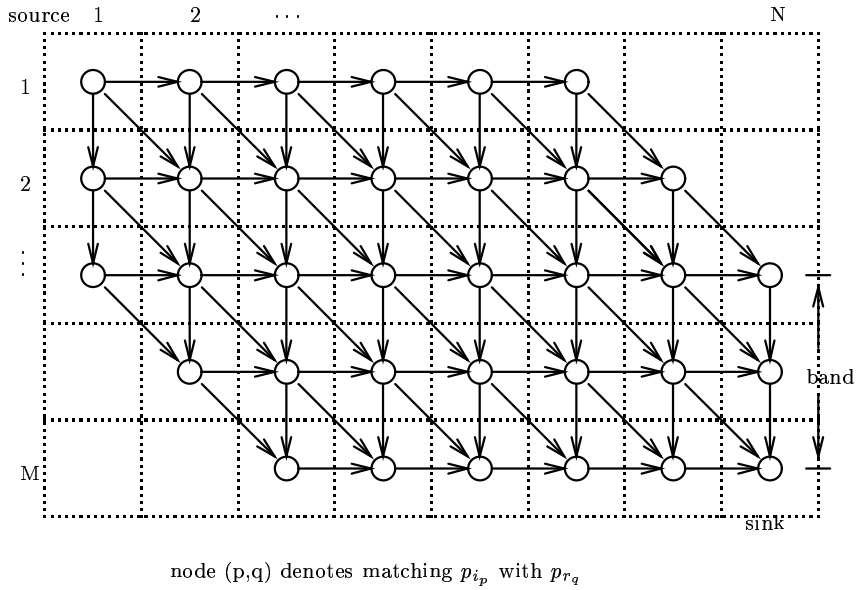


Figure 8: Matching graph $G_d(P_I, P_R)$

Unlike in the matching graph $G_s(D_I, D_R)$ a dominant point in either sequence may have multiple matches in the other sequence. The cost at a node (p, q) is defined as the Euclidean distance between dominant points p_{i_p} and p_{r_q} . The minimum total cost C_{min} from the source to the sink can be found by dynamic programming. With this cost, the distance between P_I

and P_R is defined as

$$D(P_I, P_R) = \frac{C_{min}}{N} \quad (22)$$

It is obvious that $D(P_I, P_R)$ is well defined and ≥ 0 as long as P_I and P_R are both non-empty.

3.3.2 Character Classification

Let $\{P_R\}$ be the set of dominant point sequences corresponding to the set of candidate classes $\{C_R\}$ obtained from the preclassification step. The input character C_I is classified as $C_R^* \in \{C_R\}$ if P_R^* corresponding to C_R^* satisfies

$$P_R^* = \arg \min_{P_R} \{D(P_I, P_R)\} \quad (23)$$

and

$$D(P_I, P_R^*) \leq T_D \quad (24)$$

where T_D is a threshold. Otherwise, the input character will be rejected.

3.4 An Example

We have defined the similarity and distance between strings based on dynamic programming matching using different matching graphs. Figure 9 shows an example to illustrate the ideas.

Two character patterns C_I and C_R are shown. For simplicity, we only consider dominant points of types (a) and (b) (see Section 2) in this example. Let the sequences of direction primitives of C_I and C_R be

$$D_I = d_{i_1} d_{i_2} d_{i_3} d_{i_4} = WSEN \quad (25)$$

$$D_R = d_{r_1} d_{r_2} d_{r_3} d_{r_4} = SENW \quad (26)$$

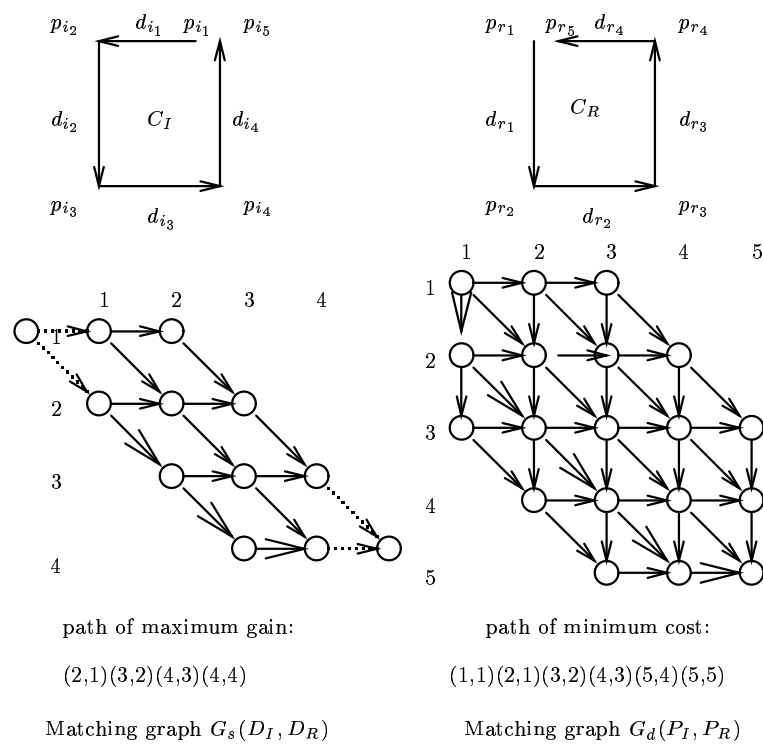


Figure 9: Example of similarity and distance measures

and the sequences of their dominant points be

$$\begin{aligned} P_I &= p_{i_1}p_{i_2}p_{i_3}p_{i_4}p_{i_5} \\ &= (91, 0)(0, 0)(0, 127)(107, 127)(80, 0) \end{aligned} \quad (27)$$

$$\begin{aligned} P_R &= p_{r_1}p_{r_2}p_{r_3}p_{r_4}p_{r_5} \\ &= (0, 0)(0, 127)(107, 127)(107, 0)(27, 0) \end{aligned} \quad (28)$$

The maximum gain G_{max} determined from the matching graph $G_s(D_I, D_R)$ is 3, while the minimum cost C_{min} determined from the matching graph $G_d(P_I, P_R)$ is 160. Thus, we get

$$S(D_I, D_R) = \frac{3}{4} = 0.75 \quad (29)$$

$$D(D_I, D_R) = \frac{160}{5} = 32 \quad (30)$$

4 Reference Set Evolution

In general, a character class may need more than one reference pattern (or template) to represent different commonly encountered variations in writing style, e.g., numeral ‘7’ and lowercase English letter ‘z’. Besides, variations in stroke order and the number of strokes may also be handled in the same way. However, due to speed and memory considerations, it may not be feasible to allow the reference set to be expanded without bound. Moreover, class separability may be affected if the reference patterns are not chosen properly. We propose a method for choosing the appropriate set of reference patterns in an iterative manner.³ This process is referred to as reference set evolution.

³Note that this issue is related to clustering.

4.1 Intraclass and Interclass Distances

Let A_M be an alphabet of M character classes with N example instances per class:

$$A_M = \{C_m | 1 \leq m \leq M\} \quad (31)$$

$$C_m = \{C_{mn} | 1 \leq n \leq N\} \quad (32)$$

For each class C_m we want to select K_m patterns from the N instances to form the reference set:

$$C_m^r = \{C_{mn_k} | C_{mn_k} \in C_m, 1 \leq k \leq K_m\} \quad (33)$$

For each subset C_m^r , the following distance measures are defined:

- Intraclass distance:

$$D_{int}(C_m^r, C_m) = \frac{1}{K_m N} \sum_{k=1}^{K_m} \sum_{n=1}^N D(P_{mn_k}, P_{mn}) \quad (34)$$

where P_{mn_k} and P_{mn} are the sequences of dominant points of C_{mn_k} and C_{mn} , respectively.

- Interclass distance:

$$\begin{aligned} D_{ext}(C_m^r, C_l^r) &= \min_{l \neq m, n_k, n_j} \{D(P_{mn_k}, P_{ln_j})\} \\ &= \min_{l \neq m} \{ \min_{n_k} \{ \min_{n_j} \{ D(P_{mn_k}, P_{ln_j}) \} \} \} \end{aligned} \quad (35)$$

where P_{mn_k} and P_{ln_j} are the sequences of dominant points of C_{mn_k} and C_{ln_j} , respectively.

Intuitively, for each character class C_m , we would like to find the optimal reference set \hat{C}_m^r such that

$$\hat{C}_m^r = \arg \min_{C_m^r} \{D_{int}(C_m^r, C_m)\} \quad (36)$$

and

$$\hat{C}_m^r = \arg \max_{C_m^r} \{D_{ext}(C_m^r, C_l^r)\} \quad (37)$$

so that it can represent the class with minimum intraclass distance and maximum interclass distance. In practice, however, this may not always be realizable. A heuristic method is proposed here to find a (suboptimal) reference set through an iterative procedure.

4.2 Iterative Evaluation and Deletion

Let P_{mn} be the sequence of dominant points of an instance C_{mn} in class C_m . We define the following evaluation function for C_{mn} :

$$J(C_{mn}) = \sum_{i=1}^N D(P_{mn}, P_{mi}) + \min_{l \neq m, n_j} \{D(P_{mn}, P_{ln_j}) \mid C_{ln_j} \in C_l^r\} \quad (38)$$

where C_l^r is the current reference set for class C_l and can be arbitrarily set at the beginning.

We delete the instance $C_{mn_d} \in C_m$ such that

$$C_{mn_d} = \arg \min_n \{J(C_{mn})\} \quad (39)$$

By repeatedly deleting instances, we can get a reference set C_m^r with K_m instances. The same procedure is performed on all M classes.

By going through the character classes for multiple passes using an iterative evaluation and deletion procedure, the resultant reference set will have small intraclass distance and large interclass distance.

5 Recognition Experiment

We have discussed the several stages of our system for data preprocessing, feature extraction, character classification, and reference set evolution. In this section, some experimental results are discussed.

Our system has been implemented on a 486 50MHz personal computer running Microsoft Windows 3.1, with a WACOM digitizing tablet as input device. The width and the height of each normalized character are set to $W = 108$ and $H = 128$, and the thresholds for preclassification and fine classification are set to $T_S = 0.6$ and $T_D = 32$. Two rectangular writing areas each of size 110×72 are assigned to numerals and English letters.

5.1 Character Classes

We use 62 character classes with different writing styles in our experiment, i.e., numerals 0-9, uppercase letters A-Z, and lowercase letters a-z. The basic writing templates for these classes are shown in Figures 10 to 12. These writing templates are derived from the Italian manuscript style and some other writing styles. Most characters are written from top to bottom and from left to right. For characters having more than one stroke, the stroke order is indicated by associated numbers.

As was mentioned above in Section 2, three ledger regions are used for English writing. The ledger lines are shown as dotted lines in Figures 11 and 12.

Other than the writing templates shown above, some character classes also have other

0	1	2	3	4	5	6	7	8	9
0	1	2 2	3	4 ¹ ₂	5 ¹ ₂	6	7 ¹ ₂	8	9

Figure 10: Basic writing templates for numerals

A	A ¹ ₂ ₃	B	B ¹ ₂	C	C	D	D ¹ ₂	E	E ¹ ₂ ₃
F	F ¹ ₂ ₃	G	G	H	H ¹ ₂ ₃	I	I	J	J
K	K ¹ ₂	L	L	M	M ¹ ₂	N	N ¹ ₂	O	O
P	P ¹ ₂	Q	Q ¹ ₂	R	R ¹ ₂	S	S	T	T ¹ ₂
U	U	V	V	W	W	X	X ¹ ₂	Y	Y ¹ ₂
Z	Z								

Figure 11: Basic writing templates for uppercase letters

a		b		c		d		e	
f		g		h		i		j	
k		l		m		n		o	
p		q		r		s		t	
u		v		w		x		y	
z									

Figure 12: Basic writing templates for lowercase letters

associated templates. For example, the letter ‘T’ in the Italian manuscript style is written with two strokes: the first one is a vertical stroke and the second, a horizontal stroke. However, some people write this character with a different stroke order, i.e., the horizontal stroke first followed by the vertical stroke. In order to handle such variation, an extra template is also included. For brevity, we do not show all the variants here.

5.2 Data Collection and Reference Set

5.2.1 Data Collection

In order to evaluate the performance of our system, 21 participants were invited to contribute handwriting data for our recognition experiment. The data for each participant are stored in three data files, one each for numerals, uppercase and lowercase letters.

When producing the first data file, each participant was asked to write the numerals 0-9 twice in two different writing styles corresponding to the writing templates in Figure 10. This file thus contains 20 character instances (0-9, 0-9). For the second data file, each participant was asked to write the uppercase letters A-Z once in the specified stroke order according to Figure 11, and then in whatever stroke order the participant prefers. Thus, this file contains a total of 52 character instances (A-Z, A-Z). For the third data file, each participant was asked to write the lowercase letters a-z twice in two different writing styles corresponding to the writing templates in Figure 12. This file contains a total of 52 character instances (a-z, a-z). Figures 13 and 14 show two examples of such data files produced by two participants. Note that in Figure 13 the two instances of characters 'G', 'I', and 'J' are quite different with respect to the number of strokes. Such variations are not consistent between different writers.

Data from shirley, 5 July 1995

A B C D E F G H I J
K L M N O P Q R S T
U V W X Y Z A B C D
E F G H I J K L M N
O P Q R S T U V W X
Y Z

Figure 13: Uppercase letters written by a participant

Data from james, 15 June 1995

a b c d e f g h i j
k l m n o p q r s t
u v w x y z a b c d
e f g h i j k l m n
o p q r s t u v w x
y z

Figure 14: Lowercase letters written by a participant

5.2.2 Reference Set

All the 124 characters written by the first participant were used as reference patterns to form the initial reference set. This set includes all the basic writing templates as shown in Figures 10 to 12, as well as some variations with respect to the number of strokes and the stroke order, such as in the characters ‘E’, ‘H’, ‘I’, and ‘T’.

Reference set evolution was then performed by using the data from five randomly chosen participants. A maximum of four reference patterns were allowed for each character class. However, the two initial reference patterns for each class were treated as “seeds” and were never deleted. After running the procedure as described in Section 4, 56 new reference patterns were selected to give a total of 180 patterns in the reference set. This reference set was then used in the recognition of the characters written by the other 15 participants.

5.3 Recognition Results

By using an elastic matching approach, our system can tolerate local variation and deformation such as stroke splitting and slanting. In our experiment, it was found that whenever a person wrote a character in accordance with one of the writing templates in the reference set, the input character was always recognized correctly. Figure 15 shows an example.


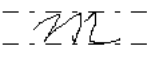
Designated Writing Area			Identity
Chinese:	English:	Numeral:	
			m
			No.2 d=15.88 s=0.893
Information of Character Candidates			
m	u	n	
No.2	No.2	No.2	
d=15.88	d=19.81	d=20.42	
s=0.893	s=0.787	s=0.707	

Figure 15: Recognition of a single character

The input character in Figure 15 was written in accordance with the second writing template for the lowercase letter ‘m’ in Figure 12. Although the stroke was split into two parts due to fast writing, a reference pattern for the correct class ‘m’ was still selected during the preclassification step. The three candidate reference patterns are reference pattern 2 of ‘m’, reference pattern 2 of ‘u’, and reference pattern 2 of ‘n’. As shown in Figure 15, the candidates are sorted in ascending order of the distance. In addition, the distance and similarity values are also shown. Since reference pattern 2 of class ‘m’ has the smallest distance value ($d = 15.88$) for the input character, the input is recognized as character ‘m’.

To collect more statistics about the performance, the data from 15 participants were tested.

Figures 16 to 18 show some examples.

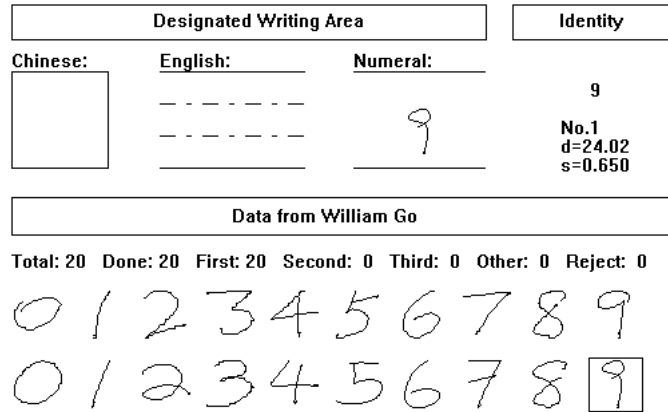


Figure 16: Recognition of numerals

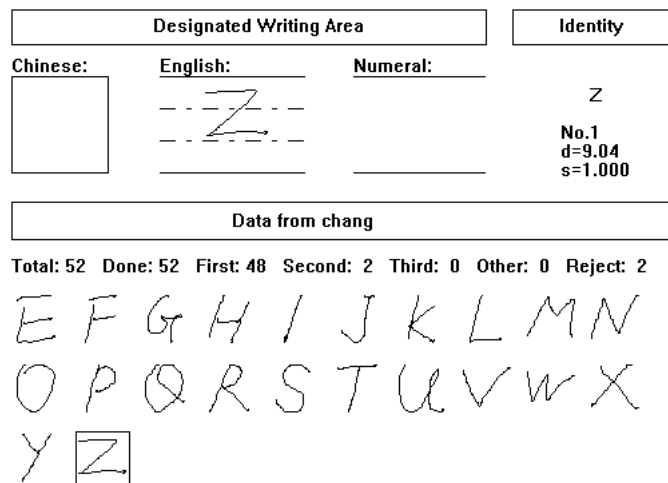


Figure 17: Recognition of uppercase letters

In each figure, the handwritten characters from a data file are displayed in the lower area. The character currently being processed is highlighted by a box, with the original writing and the recognized class displayed in the corresponding areas. The information line in the middle area provides the following statistics:

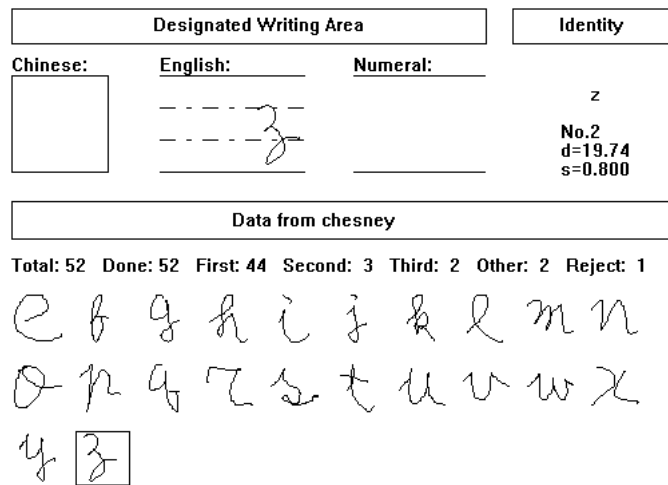


Figure 18: Recognition of lowercase letters

- total number of characters in the data file;
- number of characters processed (done) so far;
- number of characters recognized as the first candidate;
- number of characters recognized as the second candidate;
- number of characters recognized as the third candidate;
- number of characters recognized as some other candidates; and
- number of characters rejected by the system.

As part of the results, all the 20 numerals in Figure 16 were recognized correctly. Among the 52 characters (partially) shown in Figure 17, 48 were recognized as the first candidate, two as the second candidate, and two were rejected. For the lowercase letters in Figure 18, 44 were recognized as the first candidate, three as the second, two as the third, two as others, and one was rejected.

The recognition results based on data from 15 people are summarized in Tables 4 to 7. The recognition rate for numerals is the highest while that for lowercase letters is the lowest.⁴ The overall recognition rate is 91.0%, with 7.9% substitution rate and 1.1% rejection rate. If we consider the three best candidates, the hit rate is 97.1%.

Total	1st	2nd	3rd	Others	Rejected
300	289	8	0	1	2
100%	96.3%	2.7%	0.0%	0.3%	0.7%

Table 4: Recognition results for numerals

Total	1st	2nd	3rd	Others	Rejected
780	725	31	1	10	13
100%	92.9%	4.0%	0.1%	1.3%	1.7%

Table 5: Recognition results for uppercase letters

Total	1st	2nd	3rd	Others	Rejected
780	679	61	13	22	5
100%	87.1%	7.8%	1.7%	2.8%	0.6%

Table 6: Recognition results for lowercase letters

The average time for processing one data file of 20 numerals is about 6 seconds, while that for 52 English letters is about 18 seconds. Thus, the average processing time is estimated to be about 0.35 second per character.

⁴Postprocessing techniques based on context information may be used to further increase the recognition accuracy. However, such issue is beyond the scope of this paper.

Total	1st	2nd	3rd	Others	Rejected
1860	1693	100	14	33	20
100%	91.0%	5.4%	0.7%	1.8%	1.1%

Table 7: Overall recognition results

6 Conclusion and Remarks

In this paper, we have presented an approach to on-line handwritten alphanumeric character recognition. In our approach, an on-line handwritten character is characterized by a sequence of dominant points in strokes and a sequence of writing directions between consecutive dominant points. The directional information of the dominant points is used for character preclassification and the positional information is used for fine classification. Both preclassification and fine classification are based on dynamic programming matching using the idea of band-limited time warping. These techniques are elastic, in that they can tolerate local variation and deformation. The issue of reference (or template) set evolution is also addressed.

Our system has been built on a 486 50MHz personal computer running Microsoft Windows 3.1, with a WACOM digitizing tablet as input device. A recognition experiment was conducted using this approach, with 62 character classes (0-9, A-Z, a-z) of different writing styles (Italian manuscript style and some other styles) and data collected from 21 people. The data from six participants were used to build the reference set, which was then tested using data from the other 15 participants. The recognition rate of this experiment is 91%, with 7.9% substitution rate and 1.1% rejection rate. The system is reasonably fast, with average processing time of 0.35 second per character.

A remarkable aspect of our approach is that it is easily extensible to different character sets and different writing styles. For example, the system can also recognize symbols such as ‘+’, ‘-’, ‘\$’, ‘£’ if the corresponding templates are added into the reference set. Furthermore, our approach can handle large character sets (such as Chinese characters) due to its fast preclassification step.

A limitation of our system is its dependence on stroke order, since our approach is based on sequential features rather than 2-D spatial information. Thus, in general, a different stroke order requires an additional reference pattern.

In our future work, we will develop an approach insensitive to variations in stroke order. A possible solution to this problem is to cast it under a probabilistic framework so that such variations can be represented by a probability distribution over all possible sequences for a pattern class. An early approach using (observable) Markov chains⁽¹⁶⁾ and a more recent approach using hidden Markov models^(17,18) are examples in this direction.

Acknowledgement

The research reported in this paper was supported by a Sino Software Research Centre Research Award (SSRC 94/95.EG11) and a Research Infrastructure Grant (RI 92/93.EG08) of the Hong Kong University of Science and Technology. The authors like to thank all the volunteers for participating in the recognition experiment.

References

1. C.C. Tappert, C.Y. Suen and T. Wakahara, The state of the art in on-line handwriting recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.12, no.8, 787–808 (1990)
2. D.D. Kerrick and A.C. Bovik, Microprocessor-based recognition of handprinted characters from a tablet input, *Pattern Recognition*, vol.21, no.5, 525–537 (1988)
3. M. Shridhar and A. Badreldin, Recognition of isolated and simply connected handwritten numerals, *Pattern Recognition*, vol.19, no.1, 1–12 (1986)
4. L. Lam and C.Y. Suen, Structural classification and relaxation matching of totally unconstrained handwritten zip-code numbers, *Pattern Recognition*, vol.21, no.1, 19–31 (1988)
5. T. Taxt, J.B. Olafsdottir and M. Daehlen, Recognition of handwritten symbols, *Pattern Recognition*, vol.23, no.11, 1155–1166 (1990)
6. D.J. Burr, Designing a handwriting reader, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.5, no.5, 554–559 (1983)
7. M.S. El-Wakil and A.A. Shoukry, On-line recognition of handwritten isolated Arabic characters, *Pattern Recognition*, vol.22, no.2, 97–105 (1989)
8. J.S. Huang and K. Chuang, Heuristic approach to handwritten numeral recognition, *Pattern Recognition*, vol.19, no.1, 15–19 (1986)
9. R. Plamondon, Handwriting generation: the delta lognormal theory, *Proceedings of the Fourth International Workshop on Frontiers in Handwriting Recognition*, 1–10 (1994)
10. R. Plamondon, A model-based segmentation framework for computer processing of handwriting, *Proceedings of the Eleventh IAPR International Conference on Pattern Recognition*, 303–307 (1992)
11. X. Li and N.S. Hall, Corner detection and shape classification of on-line handprinted Kanji strokes, *Pattern Recognition*, vol.26, no.9, 1315–1334 (1993)

12. J.B. Kruskal, An overview of sequence comparison, *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, D. Sankoff and J.B. Kruskal (eds.), Reading, Massachusetts: Addison-Wesley, 1–43 (1983)
13. J.B. Kruskal and M. Liberman, The symmetric time-warping problem: from continuous to discrete, *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, D. Sankoff and J.B. Kruskal (eds.), Reading, Massachusetts: Addison-Wesley, 125–161 (1983)
14. F.H. Cheng, W.H. Hsu and M.Y. Chen, Recognition of handwritten Chinese characters by modified Hough transform techniques, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.11, no.4, 429–439 (1989)
15. C.K. Lin, K.C. Fan and F.T. Lee, On-line recognition by deviation-expansion model and dynamic programming matching, *Pattern Recognition*, vol.26, no.2, 259–268 (1993)
16. R.F.H. Farag, Word-level recognition of cursive script, *IEEE Transactions on Computers*, vol.28, no.2, 172–175 (1979)
17. K.S. Nathan, J.R. Bellegarda, D. Nahamoo and E.J. Bellegarda, On-line handwriting recognition using continuous parameter hidden Markov models, *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol.5, 121–124 (1993)
18. S.R. Veltman and R. Prasad, Hidden Markov models applied to on-line handwritten isolated character recognition, *IEEE Transactions on Image Processing*, vol.3, no.3, 314–318 (1994)