It appears that binary morphology has found a new field of application in the area of cluster analysis. As a natural extension of this work, the authors intend to publish a follow-up paper that will discuss the application of multivalued set-theoretic operations to pattern classification.

ACKNOWLEDGMENT

The authors wish to thank O. M'hirit, Professor at the School of Forestry of Sale, Morocco, for providing the Cedrus data. They are also grateful to B. Ceurstemont for his programming assistance.

REFERENCES

[1] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.
[2] G. Matheron, *Random Sets and Integral Geometry*. New York: Wiley, 1985.
[3] J. Serra, *Image Analysis and Mathematical Morphology*. New York: Academic, 1982.
[4] R. M. Haralick, S. R. Sternberg, and X. Zhuang, "Image analysis using mathematical morphology," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-9, no 4, pp. 532–550, 1987.
[5] H. Minkowski, "Volumen und Oberflche," *Math. Ann.*, vol. 57, pp. 447–495, 1903.
[6] J. -G. Postaire and C. P. A. Vasseur, "An approximate solution to normal mixture identification with application to unsupervised pattern classification," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-3, no 2, pp. 163–179, 1981.
[7] A. Touzani and J. -G. Postaire, "Mode detection by relaxation," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 10, no. 6, pp. 970–978, 1988.
[8] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Trans Inform. Theory*, vol. IT-13, pp. 21–27, 1967.
[9] G. H. Ball and D. J. Hall, "Isodata, A novel method of data analysis and pattern classification," NTIS Rep. AD699 616, Stanford Res. Inst., Stanford, CA, 1965.
[10] J. Macqueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Symp. Math. Stat. Prob.*, 1967, pp. 281–297.
[11] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-1, no 2, pp. 224–227, 1979.
[12] J. T. Tou Dynoc, "A dynamic optimal cluster-seeking technique," *Int. J. Comput. Inform. Sci.*, vol. 8, no. 6, pp. 541–547, 1979.
[13] K. C. Gowda and G. Krishna, "Agglomerative clustering using the concept of mutual nearest neighborhood," *Patt. Recogn.*, vol. 10, pp. 105–112, 1978.
[14] J. -G. Postaire and O. M'Hirit, "Application of pattern recognition to volume estimation in forest inventory," *Forest Sci.*, vol. 31, no. 1, pp. 53–65, 1985.
[15] P. M. Narendra, "A separable median filter for image noise," *Proc. IEEE Conf. Patt. Recogn. Image Processing*, 1978.
[16] P. A. Golder and K. A. Yeomans, "The use of cluster analysis for stratification," *Appl. Stat.*, vol. 22, pp. 213–219, 1973.

Attributed String Matching by Split-and-Merge for On-Line Chinese Character Recognition

Yih-Tay Tsay and Wen-Hsiang Tsai

***Abstract*— Consecutive strokes of Chinese characters tend to be connected in fast writing, and this causes a problem for most stroke-based recognition approaches. In this correspondence, we propose a recognition scheme to recognize cursive Chinese characters under the constraint of correct stroke writing orders. The proposed recognition scheme consists of two phases: candidate character selection and detailed matching. In the former phase, an input script with N strokes is used to split the strokes of each reference character into N corresponding parts. In the latter phase, the connected input strokes are broken into multiple strokes under the guidance of candidate characters. In both phases, dynamic programming is employed for stroke or character matching. Good experimental results prove the feasibility of the proposed approach for cursive Chinese character recognition.**

***Index Terms*—Character recognition, on-line Chinese character recognition, string matching.**

I. INTRODUCTION

On-line Chinese character recognition has been studied for many years. Many techniques have been published to solve the problem. They can basically be classified into two major approaches: namely, the statistical method [1], [2] and the structural method [3]–[11]. In the statistical method, a feature vector is usually extracted from the strokes composing an input character. Character recognition is performed by selecting the reference character with the minimum distance from the input character.

In the structural method, a set of basic strokes are usually selected as the primitives [5]–[9], and stroke recognition is based on the use of certain geometrical features like line segment directions, stroke lengths, corner numbers, etc. Stroke numbers, stroke orders, stroke relations, etc., are also found to be useful for character recognition. The recognition schemes are based on the use of decision tree [3], string matching [7], syntax and/or semantics analysis [5], radical decomposition [4], etc. Most of the recognition schemes depend heavily on the recognition of strokes. Stroke-based recognition, however, is effective under the constraint of careful writing, which places a burden on the users. Furthermore, in many applications, there is a need for fast writing for data entry. When writing fast, a user tends to connect consecutive strokes. The stroke shape in a character also varies from time to time, even when it is written by the same user. This makes stroke-based recognition a very difficult problem that does not have very high recognition rates. Recently, several attempts have been made to remove unnatural writing constraints, and some algorithms [8]–[11] have been proposed to compensate for the variations of stroke connections, stroke orders, and stroke distortion.

In this correspondence, attributed string matching with split-and-merge based on dynamic programming (DP) techniques is proposed to recognize cursive Chinese characters under the constraint of correct

Manuscript received Aug. 29, 1989; revised January 7, 1992. This work was supported by National Science Council, Republic of China under Grant NSC 81-0404-E-009-017. Recommended for acceptance by Editor-in-Chief A. K. Jain.

Y. -T. Tsay is with the Department of Computer and Engineering Science, Yuan-Ze Institute of Technology, Taoyuan, Taiwan, R.O.C.
W.-H. Tsai is with the Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan, R.O.C.

IEEE Log Number 9203754.

stroke orders. In order to reduce the computing time, the proposed recognition scheme is decomposed into two phases: candidate character selection and detailed matching. The detailed descriptions of the two phases are given in the following sections.

II. Attributed String Representation

An input character obtained from a digitization tablet can be represented as an attributed string by concatenating all the strokes according to the stroke writing order. Two kinds of primitives, namely, real primitive and pseudo primitive, are used in the attributed string. Each real primitive is a line segment that approximates part of a stroke, and each pseudo primitive is an implicit line segment that connects two consecutive strokes. Attributed string representations of on-line handwritten Chinese characters are easily obtained from the results of appropriate preprocessing and angle filtering processes [15].

Define stroke string $A_s = S_1 S_2 \cdots S_N$ to represent character A with N strokes, where each substring $S_j = s_i r_{i+1} r_{i+2} \cdots r_{i+n_j} (j = 1, 2, \cdots, N)$ represents the jth stroke of A, where n_j is the number of line segments contained in S_j. The ith primitive s_i of A is also the leading pseudo primitive of S_j, and the primitive r_{i+k} is the kth real primitive of S_j. To describe the geometric properties of the primitives, an attribute vector (q_i, l_i) is associated with each real primitive r_i, where q_i and l_i are the direction and the length attributes of r_i, respectively. Let P_{i1} and P_{i2} be the starting and the end feature points of primitive r_i. The direction attribute q_i of r_i is defined as the slope of the line segment $P_{i1}P_{i2}$, which is quantized into one of 64 directions. The length attribute l_i of r_i is defined as the Euclidean distance between P_{i1} and P_{i2}. In addition, an attribute vector $(q_j, l_j, x_{j1}, y_{j1}, x_{j2}, y_{j2})$ is associated with each pseudo primitive p_j, where q_j and l_j are the direction and length attributes of s_j, respectively, and the coordinates (x_{j1}, y_{j1}) and (x_{j2}, y_{j2}) specify the positions of the starting and the end feature points of s_j, respectively. Fig. 1(a) shows the stroke string of a character with seven strokes, whose pseudo primitives are drawn by dashed line segments.

Define another attributed string representation of a character (called the pseudo string) as the concatenation of all the pseudo primitives of the character in order. Let attributed string $A_p = p_1 p_2 \cdots p_N$ represent the pseudo string of character A with N strokes where p_i represents the ith pseudo primitive. Instead of using the attribute vector defined for a pseudo primitive in the stroke string, another attribute vector $(q_{i1}, q_{i2}, q_{i3}, x_{i1}, y_{i1}, x_{i2}, y_{i2})$ is used to specify the attributes of each pseudo primitive p_i in the pseudo string, where q_{i2} is the direction of p_i, and q_{i1} and q_{i3} are the directions of the preceding and the following real primitives in the stroke string, respectively. In addition, (x_{i1}, y_{i1}) and (x_{i2}, y_{i2}) are the coordinates of the starting and the end feature points of p_i, respectively. The pseudo string shown in Fig. 1(b) of an input script is $p_1 p_2 \cdots p_5$, where $p_1 = s_1, p_2 = s_6, p_3 = s_8, p_4 = s_{12}$, and $p_5 = s_{15}$.

III. Candidate Character Selection

In this section, the candidate characters for an input script are determined by matching the pseudo string of the input (called the input pseudo string henceforth) with those of the reference characters. The matching scheme may be regarded as a process of splitting the stroke sequence of each reference character by the pseudo primitives of the input script. The validity of this "reference character splitting" for candidate character selection is based on the following observation. When a user writes an input script in a cursive way, it rarely happens that a stroke in the input script is broken into two or more strokes. On the contrary, two consecutive strokes in the reference character are often connected together to form a single stroke, i.e., the pseudo primitive between the two consecutive strokes in the reference character often disappears in the input pseudo string. Therefore, under the assumption that the input script is written in a correct stroke order, a pseudo primitive in the input script always corresponds to a pseudo primitive in the reference character, and the stroke number of the input script is usually no greater than that of the reference character. In other words, the input pseudo string can be considered to be a partial string of that of the reference character. Define the pseudo string distance between two characters as the minimum of the sums of the distances of the corresponding pseudo primitives in the two characters. Then, the reference characters with smaller pseudo string distance values can be selected as the candidates for detailed matching.

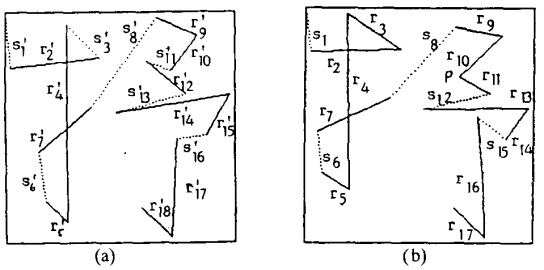

Fig. 1. String representations of a reference character and an input script: (a) Reference character and its stroke string representation; (b) input script and its stroke string representation.

More specifically, the phase of candidate character selection can be divided into three stages. The first stage is to determine an initial set R of reference characters from which candidate characters can be selected. Under the assumption that each stroke in the reference character is seldom broken into two or more strokes, the number N of strokes in the input script can be utilized as the lower limit of the number of strokes for each reference character in R. On the other hand, because each stroke in the input script may correspond to one or more strokes in each reference character, the numbers of line segments in the strokes of the input script can be used to estimate the upper limit of the possible number U of strokes for each reference character in R. This can be done by the use of a heuristic function F_h in the following way:

$$U = \sum_{i=1}^{N} F_h(n_i)$$

where

$$\begin{aligned} F_h(n_i) &= n_i, && \text{if } n_i \le 2; \\ &= n_i - 1, && \text{if } 3 \le n_i \le 4; \\ &= n_i - 2, && \text{if } n_i \ge 5 \end{aligned}$$

and n_i is the number of line segments that approximate the ith stroke of the input script. The heuristic function defined above is based on the fact that if the line segments in the input stroke is longer, the strokes in the reference will be longer. Finally, the reference characters, each having its stroke numbers between N and U, are selected to compose the set R. This completes the first stage of candidate character selection.

In the second stage of candidate character selection, the pseudo string of each reference character in R is matched with that of the input script. Let attributed string $A_p = p_1 p_2 \cdots p_N$ represent the pseudo string of the input script A with N strokes and $B_p = p'_1 p'_2 \cdots p'_M$ represent the pseudo string of reference character B

in R with M strokes. Assume that $M \geq N$. In addition, let $(q_{i1}, q_{i2}, q_{i3}, x_{i1}, y_{i1}, x_{i2}, y_{i2})$ and $(q'_{j1}, q'_{j2}, q'_{j3}, x'_{j1}, y'_{j1}, x'_{j2}, y'_{j2})$ be the attribute vectors of p_i and p'_j that represent the ith and the jth pseudo primitives of A and B, respectively. The primitive distance $d(p_i, p'_j)$ between primitives p_i and p'_j is defined as follows:

$$\begin{aligned} d(p_i, p'_j) = {} & H(q_{i1}, q'_{j1}) + H(q_{i2}, q'_{j2}) + H(q_{i3}, q'_{j3}) \\ & + W_1(|x_{i1} - x'_{j1}| + |y_{i1} - y'_{j1}| \\ & + |x_{i2} - x'_{j2}| + |y_{i2} - y'_{j2}|) \end{aligned}$$

where H is a function specifying direction difference values, and W_1 is a preselected constant used to normalize the weights of the difference measures of the direction and position attributes. The direction difference function $H(q_{ik}, q'_{jk})$ is defined to be

$$H(q_{ik}, q'_{jk}) = \min(|q_{ik} - q'_{jk}|, 64 - |q_{ik}, q'_{jk}|).$$

The pseudo string distance $D_p(A, B)$ between A and B is defined as the minimum of the sums of the distances of the corresponding pseudo primitives between A and B, i.e., $D_p(A, B)$ can be defined as follows:

$$D_p(A, B) = \min_f \sum_{i=1}^{N} d(p_i, p'_{f(i)})$$

where the primitive-correspondence function f must satisfy the following constraints:

$$\begin{aligned} & i \leq f(i) \leq i + M - N; \\ & f(i-1) < f(i). \qquad \text{for } i = 1 \text{ through } N. \end{aligned}$$

The constraint $f(i-1) < f(i)$ comes from the assumption that the strokes in the input script are written in a correct order.

The pseudo string distance $D_p(A, B)$ defined above can be computed as the minimum cost of a sequence of primitive edit operations that transform B_p into A_p. The mapping of each primitive p'_j of B_p by the primitive-correspondence function f falls into one of the following two cases. 1) There exists one p_i of A_p corresponding to p'_j. This case can be considered to be a change operation that transforms p'_j into p_i, and the cost of the change operation is just the pseudo primitive distance $d(p_i, p'_j)$; 2) there does not exist any p_i of A_p corresponding to p'_j. This case can be considered to be a delete operation that deletes p'_j from B_p, and the cost of the delete operation can be taken to be zero. Note that exactly $M - N$ pseudo primitives in B_p should be deleted.

Let D be an array such that $D(i, j)$ means the minimum edit cost of transforming $p'_1 p'_2 \cdots p'_j$ into $p_1 p_2 \cdots p_i$ or, equivalently, the minimum pseudo string distance between $p_1 p_2 \cdots p_i$ and $p'_1 p'_2 \cdots p'_j$. Based on the above concept of pseudo string edition, the array element $D(i, j)$ can be obtained by the following DP technique:

$$\begin{aligned} D(i, i) = {} & D(i-1, i-1) + d(p_i, p'_i), \\ & \text{for } i = 1, 2, \cdots, N \\ D(i, j) = {} & \min(D(i-1, j-1) + d(p_i, p'_j), D(i, j-1)), \\ & \text{for } i = 1, 2, \cdots, N \text{and} j = i+1, i+2, \cdots, M. \end{aligned}$$

The final element $D(N, M)$ is simply the minimum edit cost of transforming B_p into A_p, which is also the desired minimum pseudo string distance $D_p(A, B)$ between A and B.

To obtain the index j of the corresponding p'_j of each p_i, the above procedure can be easily modified by keeping the index flow of $D(i, j)$ into an array C. After the final element $D(N, M)$ is obtained, the corresponding pseudo primitive pairs can be obtained by tracing back through C. By this correspondence information, it is easy to find the corresponding strokes in B for each input stroke in A. From these corresponding pairs, the stroke string of B can then be split into N substrings, where each contains one or more strokes. We call this process reference character splitting. Fig. 1 shows an example in which the pseudo strings of an input script A and a reference character B are represented by $A_p = p_1 p_2 \cdots p_5 = s_1 s_6 s_8 s_{12} s_{15}$, and $B_p = p'_1 p'_2 \cdots p'_7 = s'_1 s'_3 s'_6 s'_8 s'_{11} s'_{13} s'_{16}$, respectively. Five corresponding pairs $(s_1, s'_1), (s_6, s'_6), (s_8, s'_8), (s_{12}, s'_{13})$, and (s_{15}, s'_{16}) can be obtained from the above procedure. Therefore, the stroke string of the reference character is split into five substrings $s'_1 r'_2 s'_3 r'_4 r'_5$, $s'_6 r'_7$, $s'_8 r'_9 r'_{10} s'_{11} r'_{12}$, $s'_{13} r'_{14} r'_{15}$, and $s'_{16} r'_{17} r'_{18}$, which correspond, respectively, to the input strokes represented by substrings $s_1 r_2 r_3 r_4 r_5$, $s_6 r_7$, $s_8 r_9 r_{10} r_{11}$, $s_{12} r_{13} r_{14}$, and $s_{15} r_{16} r_{17}$.

The final stage of the candidate character selection phase is to find out from the set R the 15 reference characters with the smallest pseudo string distances from the input script as the candidate characters for detailed matching described next.

IV. Detailed Matching

In the detailed matching phase, each input stroke is matched with the corresponding strokes in each candidate character. Each stroke in the input or the reference is represented by a substring of pseudo and real primitives. The basic idea of matching here is to use the pseudo primitive in the substring of a candidate character to split the substring of an input stroke if the input stroke is matched with more than one stroke in the candidate character. This process is called input stroke splitting.

More specifically, when more than one stroke in the candidate character are considered to correspond to one input stroke, each pseudo primitive in the strokes of the candidate character may fall into one of the following two cases: 1) The pseudo primitive corresponds to a real primitive in the input script. For example, the pseudo primitive s'_3 in Fig. 1(a) corresponds to the real primitive r_3 in Fig. 1(b). It is reasonable here to match the two feature points of s'_3 with the corresponding feature points of r_3 and to split the first input stroke in Fig. 1(b) into r_2 and $r_4 r_5$. This match will be denoted as $r_3 \rightarrow s'_3$; 2) the pseudo primitive disappears in the input script, i.e., the pseudo primitive reduces to one feature point. For example, the pseudo primitive s'_{11} in Fig. 1(a) becomes a feature point P that connects the real primitives r_{10} and r_{11} in Fig. 1(b). In this case, the two feature points of s'_{11} should be matched with the feature point P, and the third input stroke in Fig. 1(b) should be split into $r_9 r_{10}$ and r_{11}. This match will be denoted as $\lambda_{11} \rightarrow s'_{11}$, where λ_{11} is a null primitive connecting r_{10} and r_{11}. The cost function of the above split operations will be described later, after the detailed matching process is described in the following.

Let attributed strings $S_t = s_1 r_2 r_3 \cdots r_n$ and $S'_t = s'_1 r'_2 \cdots s'_i r'_{i+1} \cdots r'_m$ represent the substrings of the ith stroke of the input script A and the corresponding strokes in a candidate character B, respectively. Note that the leading pseudo primitive s_1 is the only pseudo primitive in S_t, and it should be matched with the leading pseudo primitive s'_1 of S'_t. In addition, note that S'_t may contain more than one pseudo primitive. Let the direction and the length attributes of r_i and r'_j be denoted as (q_i, l_i) and (q'_j, l'_j), respectively. In addition, let the attribute vectors of pseudo primitives s_i and s'_j be denoted as $(q_i, l_i, x_{i1}, y_{i1}, x_{i2}, y_{i2})$ and $(q'_j, l'_j, x'_{j1}, y'_{j1}, x'_{j2}, y'_{j2})$, respectively. Generally speaking, the interstroke distance between S_t and S'_t can be defined in terms of the minimum-cost edit sequence from S_t to S'_t. Allowable edit operations and their cost functions R dependent on the types of primitives are described as follows:

1) *Change:* To replace the first pseudo primitive s_1 of S_t with s'_1 of S'_t or to replace a real primitive r_i with another r'_j, denoted

as $s_1 \rightarrow s_1'$ or $r_i \rightarrow r_j'$, respectively, with costs

$$R(s_1 \rightarrow s_1') = H(q_1, q_1') + W_2 \cdot [|l_i - l_i'| + E_d \cdot ((x_{11}, y_{11}), (x_{11}', y_{11}')) + E_d((x_{12}, y_{12}), (x_{12}', y_{12}'))],$$

$$R(r_i \rightarrow r_j') = H(q_i, q_j') + W_2 \cdot |l_i - l_j'|.$$

2) *Insert:* To insert a real primitive r_j' into S_t, denoted as $\lambda \rightarrow r_j'$, where λ is a null symbol, with cost

$$R(\lambda \rightarrow r_j') = K_i + W_2 \cdot l_j'.$$

3) *Delete:* To delete a real primitive r_i from S_t, denoted as $r_i \rightarrow \lambda$, with cost

$$R(r_i \rightarrow \lambda) = K_d + W_2 \cdot l_i.$$

4) *Split:* To split S_t into two parts; denoted as $r_i \rightarrow s_j'$ or $\lambda_i \rightarrow s_j'$, i.e., for the case $r_i \rightarrow s_j', S_t = s_1 r_2 \cdots r_n$ is split into $s_1 r_2 \cdots r_{i-1}$ and $r_{i+1} r_{i+2} \cdots r_n$, whereas for the case $\lambda_i \rightarrow s_j', S_t = s_1 r_2 \cdots r_n$ is split into $s_1 r_2 \cdots r_{i-1}$ and $r_i r_{i+1} \cdots r_n$, with costs

$$R(r_i \rightarrow s_j') = H(q_1, q_j') + W_2 \cdot [|l_i - l_j'| + E_d((x_{j1}', y_{j1}'), (x_i, y_i)) + E_d((x_{j2}', y_{j2}'), (x_{i+1}, y_{i+1}))],$$

$$R(\lambda_i \rightarrow s_j') = K_s + W_2 \cdot [l_j' + E_d((x_{j1}', y_{j1}'), (x_i, y_i)) + E_d((x_{j2}', y_{j2}'), (x_i, y_i))].$$

5) *Merge:* To merge k consecutive real primitives in S_t into a longer real primitive, denoted as $r_i r_{i+1} \cdots r_{i+k-1} \rightarrow r_i^*$

$$R(r_i r_{i+1} \cdots r_{i+k-1} \rightarrow r_i^*) = (l_i^*/L_r) \cdot N_r \cdot \sum_{j=1}^{i+k-1} (H(q_i^*, q_j) \cdot l_j / l_i^*))$$

where (q_i^*, l_i^*) is the attribute vector of the combined real primitive r_i^*, $E_d((x_1, y_1), (x_2, y_2))$ is the Euclidean distance between points (x_1, y_1) and (x_2, y_2), L_r and N_r are the total length and the number of all the real primitives in the input script, and W_2, K_i, K_d, and K_s are preselected constants. Detailed explanations of the cost functions and the attribute vector of the combined primitive defined above can be found in [13] and [14].

The interstroke distance $D_s(S_t, S_t')$ from S_t to S_t' is defined as the minimum of the cost of all the edit sequences taking S_t to S_t', and the character distance from the input script A to the candidate character B is defined as

$$D_c(A, B) = \sum_{t=1}^{N} D_s(S_t, S_t')$$

where N is the number of strokes in A.

Because merge operations are only applied to the real primitives of the input stroke, the merge costs and the attributes of all possible combined real primitives can be computed independently and stored in an array before matching the input with the candidate characters. In addition, to reduce the computation time, it is assumed that the maximum number of real primitives allowed in a merge operation is three. This has been found to be reasonable according to our experimental results.

The interstroke distance $D_s(S_t, S_t')$ between the tth stroke $S_t = s_1 r_2 \cdots r_n$ of A and its corresponding strokes $S_t' = s_1' r_2' \cdots s_i' r_{i+1}' \cdots r_m'$ in candidate character B is computed as follows. Let D be an array such that $D(i, j)$ means the minimum cost for transforming the attributed string $s_1 r_2 \cdots r_i$ into the attributed string $s_1' r_2' \cdots x_j'$ (where x_j' is either a real primitive r_j' or a pseudo primitive s_j'), i.e., let $D(i, j) = D_s(s_1 r_2 \cdots r_i, s_1' r_2' \cdots x_j')$. The distance array D can be computed by a DP technique whose essence is described below.

1) If x_j' is a real primitive, then it is matched with the input real primitives, and $D(i, j)$ is computed as follows:

$$m1 = D(i, j-1) + R(r_j \rightarrow \lambda);$$
$$m2 = D(i-1, j) + R(\lambda \rightarrow r_i');$$
$$m3 = D(i-1, j-1) + R(r_j \rightarrow r_i');$$
$$m4 = D(i-1, j-2) + R(r_{j-1} r_j \rightarrow r*) + R(r* \rightarrow r_i');$$
$$m5 = D(i-1, j-3) + R(r_{j-2} r_{j-1} r_j \rightarrow r*) + R(r* \rightarrow r_i');$$
$$D(i, j) = \min(m1, m2, m3, m4, m5).$$

2) If x_j' is a pseudo primitive, then the input stroke S_t is split into two parts, and $D(i, j)$ is computed as follows:

$$m1 = D(i-1, j-1) + R(r_j \rightarrow s_i');$$
$$m2 = D(i-1, j) + R(\lambda_{j+1} \rightarrow s_i');$$
$$D(i, j) = \min(m1, m2).$$

The final element $D(n, m)$ is just the interstroke distance $D_s(S_t, S_t')$, which is then added iteratively to the character distance $D_c(A, B)$.

The final stage in detailed matching is simply to select, as the recognition result of the input script, the candidate character with the smallest character distance from the input script.

V. Experimental Results

In our experiments, the handwriting area of each character is 15 by 15 mm on a digitization tablet whose spatial resolution is 20 lines/mm. After the preprocessing and angle filtering processes are performed to each stroke of a character, a size and position normalization operation is applied to reduce the character data into a size of 200 by 200 pixels, from which the attributed string representation is obtained. The attribute weighting factors W_1 described in Section III and W_2 described in Section IV were both set to be 0.25, which equalizes the weights of the difference measures of the direction and length attributes. The constants K_i and K_d for insertions and deletions described in Section IV were both chosen to be 16, which is one half of the maximum value of the direction difference function. In addition, the constant K_s for the split cost was chosen to be 8. The choice of the above preselected constants was based on our experimental results. In addition, the recognition rates changed only slightly when the preselected values were perturbed by small amounts.

A database composed of 3100 Chinese characters was constructed as the references. The stroke numbers of the characters range from 1 to 24. To create the input data, in total, 200 characters chosen from famous Chinese poems were written by ten persons. Each scripter was instructed to write characters in identical character style and with handwriting speed identical to writing daily notes. The scripter was also informed of the correct stroke writing order of each character. Table I shows the experimental results. Out of the 2000 input scripts, only 76 were misrecognized. The recognition rate, thus, was 96.2%. The fourth through the 11th columns of Table I show the statistics of the numbers of input scripts based on the number of connected strokes in the scripts. Out of the 2000 input scripts, only 1061 have

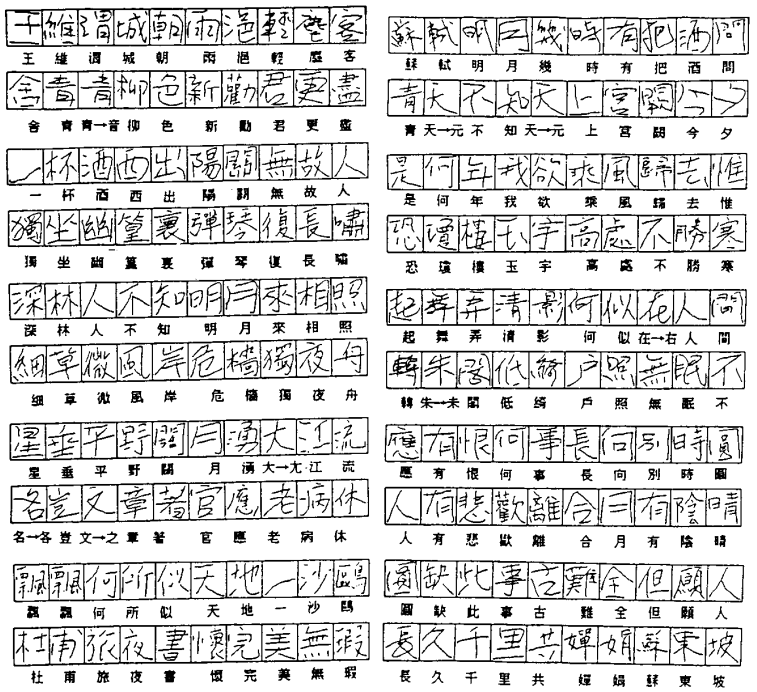

Fig. 2. Partial recognition results of 200 input scripts chosen from ten scripters. Each error is marked by the notation $A \rightarrow B$, which means that the script actually is A but was recognized to be B.

TABLE I
RECOGNITION RESULTS OF 200 INPUT SCRIPTS WRITTEN BY TEN SCRIPTERS

scripter	error count	recognition rate	distribution of input scripts based on number k of connected strokes							
			k=0	k=1	k=2	k=3	k=4	k=5	k=6	k≥7
1	12	94.0%	59	47	41	34	8	6	4	1
2	4	98.0%	128	50	18	3	0	1	0	0
3	6	97.0%	89	58	37	9	5	2	0	0
4	15	92.5%	73	57	44	13	7	3	2	1
5	5	97.5%	142	40	12	1	4	1	0	0
6	16	92.0%	50	66	34	22	14	8	4	2
7	7	96.5%	119	52	23	5	1	0	0	0
8	8	96.0%	138	36	11	11	1	1	1	1
9	1	99.5%	146	45	7	1	1	0	0	0
10	2	99.0%	117	53	18	10	1	1	0	0
total	76	96.2%	1061	504	245	109	42	23	11	5

correct input stroke counts (i.e., $k = 0$), and therefore, each of the remaining 939 input scripts has at least one stroke connected to another (i.e., $k \leq 1$). The maximum number of connected strokes among the input scripts is up to ten (i.e., $k = 10$), and all characters with $k = 10$ were correctly recognized. This shows that the proposed approach is quite powerful for recognizing highly cursive scripts. Due to different writing habits, each scripter shows a different degree of stroke connection. The recognition rate tends to decrease slightly when the input strokes are heavily connected. The recognition results of some input scripts are shown in Fig. 2.

It has been found that three factors lead to the above 76 misrecognitions of the input scripts. The first is that the candidate selection phase fails to select correct references. Five errors come from this factor. The second factor is that similar character structures exist in Chinese characters. In total, 66 errors were caused by this factor. The third factor is the distortion existing in the input scripts. Five input scripts were misclassified due to this factor. Define the cumulative recognition rate in the Lth order as the rate at which the correct character for an input script appears in the list of the first L candidates selected according to the character distance values. Table II shows the results of the cumulative recognition rates. Only the first three candidates need to be selected for the rate to reach 99%.

TABLE II
CUMULATIVE RECOGNITION RATES

order	1st	2nd	3rd	5th	15th
cumulative recognition rate	96.2%	98.6%	99.35%	99.55%	99.75%

A prototype recognition system has been constructed on an IBM PC/AT. The C language was used for programming. The average time to recognize an input script is about 2.5 s. Due to the large number of Chinese character categories, most of the processing time was spent in the candidate selection phase. The number of references selected in the candidate character selection phase tends to be large when the strokes of the input script are heavily connected, causing the processing time to increase.

VI. CONCLUDING REMARKS

A new method for on-line handwritten Chinese character recognition has been proposed in this paper. Two kinds of attributed string representations are proposed to represent handwritten Chinese characters. By using the split concept, an input script can be used to split the strokes of each of the references to obtain the desired stroke correspondence in the candidate selection phase. In addition, in the detailed matching phase, the proposed split operation can be employed to solve the input stroke connection problem caused by fast writing. Good experimental results show the effectiveness of the proposed recognition approach in handling the stroke connection problem.

Most errors in the experimental results were caused by the problem that there are too many Chinese characters with similar structures. At least two methods can be used to solve this problem. One is to perform more detailed recognition using more subtle features after the detailed matching phase. The other is to modify the attributed string representations of similar reference characters to make the substrings of their identical parts (such as their common radical) the same. This will enhance the discrimination power of the distances between the input script and similar reference characters.

In addition, the constraint of correct stroke orders, which increases burden to the user and limits writing speed, imposed by the proposed approach can be relaxed by adding all the possible stroke orders of each Chinese character in the database. However, enlarging the database will increase the recognition time. It also seems difficult to handle all possible stroke orders of different users. Because the stroke order of each character seems stable for each particular user, it is feasible to use the proposed approach to design a writer-dependent system. Future research can be directed toward investigating a more intelligent method for handling stroke orders and making the proposed recognition scheme more flexible.

REFERENCES

[1] H. Arakawa, "On-line recognition of handwritten characters—Alphanumerics, hiragana, katakana, kanji," *Patt. Recogn.*, vol. 16, no. 1, pp. 9–16, 1983.

[2] K. Odaka *et al.*, "On-line recognition of handwritten characters by approximating each stroke with several points," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-12, pp. 898–903, 1982.

[3] S. Hanaki and T. Yamazaki, "On-line recognition of handprinted Kanji characters," *Patt. Recogn.*, vol. 12, pp. 421–429, 1980.

[4] E. F. Yhap and E. C. Greanias, "An on-line Chinese character recognition system," *IBM J. Res. Devel.*, vol. 25, no. 3, pp. 187–195, May 1981.

[5] C. Hsu *et al.*, "A syntactic-semantic approach to recognize handwritten Chinese characters by using a digitizing table as the input device," *CPCOL*, vol. 2, no. 4, pp. 198–215, 1986.

[6] M. Nakagawa *et al.*, "On-line recognition of handwritten Japanese character in JOLIS-1," in *Proc. 6th ICPR*, 1982, pp. 776–779.

[7] S. L. Shiaw *et al.*, "On-line handwritten Chinese character recognition by string matching," in *Proc. 1988 ICCPCOL*, 1988, pp. 76–80.

[8] T. Wakahara and M. Umeda, "On-line cursive script recognition using stroke linkage rules," in *Proc. 7th ICPR*, 1984, pp. 1065–1068.

[9] K. Ishigaki and T. Morishita, "A top-down online handwritten character recognition method via the denotation of variation," in *Proc. 1988 ICCPCOL*, 1988, pp. 141–145.

[10] T. Morishita *et al.*, "A Kanji recognition method which detects writing errors," *IJPRAI*, vol. 2, no. 1, pp. 181–195, Mar. 1988.

[11] K. J. Chen *et al.*, "A system for on-line recognition of Chinese characters," *IJPRAI*, vol. 2, no. 1, pp. 139–148, Mar. 1988.

[12] R. A. Wagner and M. J. Fisher, "The string-to-string correction problem," *JACM*, vol. 21, pp. 168–173, Jan. 1974.

[13] W. H. Tsai and S. S. Yu, "Attributed string matching with merging for shape recognition," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-7, pp. 453–462, July 1985.

[14] Y. T. Tsay, "Model-guided attributed string matching by split-and-merge for shape recognition and on-line Chinese character recognition," Ph.D. thesis, National Chiao Tung Univ., Taiwan, R.O.C., 1988.

[15] C. C. Tappert *et al.*, "The state of the art in on-line handwritten recognition," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-12, pp. 787–808, Aug. 1990.

Breaking Substitution Cyphers Using Stochastic Automata

B. J. Oommen, *Senior Member, IEEE,* and J. R. Zgierski

***Abstract*— Let Λ be a finite plaintext alphabet and V be a cypher alphabet with the same cardinality as Λ. In all one-to-one substitution cyphers, there exists the property that each element in V maps onto exactly one element in Λ and vice versa. This mapping of V onto Λ is represented by a function T^*, which maps any $v \in V$ onto some $\lambda \in \Lambda$ (i.e., $T^*(v) = \lambda$). In this correspondence, we consider the problem of learning the mapping of T^* (or its inverse $(T^*)^{-1}$) by processing a sequence of cypher text. The fastest reported method to achieve this is an elegant relaxation scheme due to Peleg *et al.* [8], [9] that utilizes the statistical information contained in the unigrams and trigrams of the plaintext language. In this correspondence, we present a new learning automaton solution to the problem called the cypher learning automaton (CLA). The proposed scheme is fast, and the advantages of the scheme in terms of time and space requirements over the relaxation method have been listed. The correspondence contains simulation results comparing both cypher-breaking techniques.**

***Index Terms*— Cytography, learning automata, relaxation methods, substitution cyphers.**

I. INTRODUCTION

The art of cryptography has very probably existed ever since writing was invented. The purpose of it was, of course, to hide a message by systematically transforming the original message into a form that is unintelligible to the reader unless it is first decyphered. In order to prevent unauthorized readers (or eavesdroppers) from decyphering the original message, the strategy with which the transformed message is decyphered is maintained as a well-kept secret between the sender and the intended receiver. Hopefully, in this way, the communication between the authentic sender and receiver cannot be intercepted by others for whom it is not meant. From the point of view of the eavesdropper, the heart of the problem is to break the cypher without a knowledge of the decyphering process, but this is usually not an easy task.

This correspondence concerns the breaking of substitution cyphers using learning automata (LA).

Throughout this correspondence, we shall use the word plaintext to refer to the orginial message and the term cyphertext to refer to the transformed message. The verbs cypher, encrypt, and encode will be used interchangeably, and decrypt, decypher, and decode will be used interchangeably.

Cyphers that use a direct one-to-one mapping are known as monoalphabetic substitution cyphers. By monoalphabetic, we mean that only one alphabet is used in encyphering a message, and this is referred to as the cypher alphabet. In general, if Λ is a finite plaintext alphabet and V is a cypher alphabet with the same cardinality as Λ, a substitution cypher may use any of the $|\Lambda|!$ permutations as its key. Clearly, finding the correct key to decypher such an encoded message is an extremely tedious task if one merely randomly tries all the different mappings. Even with a computer doing the job, this

Manuscript received November 7, 1990; revised January 21, 1992. This work was supported by the Natural Sciences and Engineering Research Council of Canada. A preliminary version of some of this work appears in the *Proceedings of the Fourth IEA/AIE International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Hawaii, June, 1991. Recommended for acceptance by Associate Editor S. Tanimoto.

The authors are with the School of Computer Science, Carleton University, Ottawa, Canada K1S5B6.

IEEE Log Number 9204243.