PERGAMON

Pattern Recognition 34 (2001) 2339–2352

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Model-based stroke extraction and matching for handwritten Chinese character recognition

Cheng-Lin Liu[a,*], In-Jung Kim[b], Jin H. Kim[b]

[a]*Multimedia Systems Research Department, Central Research Laboratory, Hitachi, Ltd. 1-280 Higashi-koigakubo, Kokubunji-shi, Tokyo 185-8601, Japan*

[b]*AI Lab., Department of Computer Science, KAIST, 373-1 Kusong-dong Yusong-gu, Taejon 305-701, South Korea*

Received 24 September 1999; accepted 30 October 2000

Abstract

This paper proposes a model-based structural matching method for handwritten Chinese character recognition (HCCR). This method is able to obtain reliable stroke correspondence and enable structural interpretation. In the model base, the reference character of each category is described in an attributed relational graph (ARG). The input character is described with feature points and line segments. The strokes and inter-stroke relations of input character are not determined until being matched with a reference character. The structural matching is accomplished in two stages: candidate stroke extraction and consistent matching. All candidate input strokes to match the reference strokes are extracted by line following and then the consistent matching is achieved by heuristic search. Some structural post-processing operations are applied to improve the stroke correspondence. Recognition experiments were implemented on an image database collected in KAIST, and promising results have been achieved.

Keywords: Chinese character recognition; Structural matching; Model-based stroke extraction; Heuristic search; Semi-admissible search

1. Introduction

To solve the hard problem of handwritten Chinese character recognition (HCCR), a wide variety of methods have been proposed. In terms of description scheme, the methods proposed so far can be divided into two groups: feature matching and structural matching [1–3]. Feature matching methods are efficient in recognition speed and noise immunity but is not sufficient in deformation tolerance. In contrast, structural matching methods are more flexible to tolerate shape deformation but are complicated in implementation and expensive in computation. Compared to feature matching, structural matching provides more information in addition to the class identity of input pattern. By structural matching, the structural description of input character is obtained so that structural interpretation and knowledge-based recognition become feasible. For this reason, numerous works on structural matching have been done and fruitful results have been reported.

Structural matching methods in HCCR can be further classified into contour matching [4,5], stroke matching [6,7] and hierarchical matching [8,9]. Stroke matching has been intensively pursued and the concept *stroke* has various definitions: natural stroke, line stroke, line segment, and short segment. A natural stroke is a component of trajectory from pen-down to pen-lift in regular writing, which has been adopted in off-line character recognition by Hsieh and Lee [10,11]. A natural stroke is composed of one or more collinear line strokes. Since the natural stroke is highly difficult to extract from character images, the line stroke is more frequently adopted as the structural primitive in Chinese character recognition

* Corresponding author. Tel.: + 81-42-323-1111; Ext. 3607; fax: + 81-42-327-7778.

E-mail address: liucl@crl.hitachi.co.jp (C.-L. Liu).

0031-3203/01/$20.00
PII: S0031-3203(00)00165-5

[6,7,12–14]. A line segment is a piece of stroke between two feature points (end-point, junction point, or deflection point of high curvature). It is relatively easy to extract but is susceptible to topology corruption [15]. A short segment is a portion of line segment with approximately equal length so as to overcome the instability of topology [16,17]. However, the short segment description enlarges the data amount and is weak to grasp the global structure.

Structural matching is basically a combinatorial problem, wherein the stroke correspondence between an input character and a reference character is sought so as to interpret the structure of the input character and make decision for classification. Optimization algorithms employed in structural matching of Chinese characters include the relaxation labeling [4,6–8], tree search [10,13], linear programming (LP) [12], dynamic programming (DP) [5,11,16], artificial neural network [14], etc. Out of the matching algorithms, the relaxation labeling and tree search have been most widely used. While relaxation labeling is more computationally efficient, tree search is more flexible in structural analysis and incorporating human knowledge.

Extracting strokes from 2D images is a major difficulty for structural matching. Due to the stroke interference and the ambiguity of curvature, to cut and connect line segments into collinear line strokes is not clear-cut. Suganthan and Yan [14] gave some examples of stroke ambiguity, including stroke breaking, orientation variation, accidental collinear connection, shape distortion, etc. Extracting strokes from skeleton, contour or line adjacency graph (LAG) encounters respective difficulties. So far, many works have been done to specially solve the problem of stroke extraction [18–26], but the reliability is not adequate if no structural information from the reference model is incorporated.

Model-based stroke extraction is effective in overcoming the ambiguity. It can be dated back to Tsui [27], where the stroke extraction was guided by model strokes but was not integrated with matching. Yeung and Fong suggested the integration of classification into stroke extraction but they did not implement it [23]. Generally, the model-based approach dynamically splits and merges line segments to form plausible strokes according to the model information. The strategy of dynamic line segment merging has been adopted in on-line [28] and off-line character recognition [29], as well as in shape recognition [30]. In these works the segment merging and consistent matching are integrated in a tree search framework. Cheng proposed a method to dynamically merge stroke segments during relaxation labeling [15]. Another related work was to extract candidate strokes prior to matching and then seek a consistent combination of strokes [10,11].

Since the works of Hsieh and Lee [10,11], Cheng [15], Rocha and Pavlidis [29] are closely related to our work, it is worthwhile to briefly review their methods. Hsieh and Lee [10,11] used on-line character models (composed of natural strokes) to guide the extraction and matching of strokes for off-line Chinese character recognition. Prior to stroke matching, all possible input strokes of the stroke types present in the reference model were detected from the input character, and then the stroke correspondence was obtained by tree search or Viterbi search. In the on-line character model, only the relations between consecutive strokes in writing order were exploited. In Cheng's work [15], line segments of the input and reference characters were initially matched one-to-one and then tentatively connected to form long strokes according to the matching scores during relaxation labeling. Rocha and Pavlidis [29] grouped the stroke components of an input character dynamically to match against a prototype character. They defined a number of shape transformations to accomplish the stroke grouping and matching. These operations were unified by best-first tree search. In the description of prototypes, however, they did not explore the inter-stroke relationship. Their method was tested on recognition of alpha-numeric characters.

In this paper, we propose a new model-based method for stroke extraction and structural matching in HCCR. We adopt the line stroke as the primitive of structural description and the relation between neighboring strokes in reference characters are well defined. The reference character is described in an attributed relational graph (ARG) while the input character is described with feature points and line segments. In structural matching, all the valid candidate input strokes to match a reference stroke are extracted by line following, and the optimal combination of candidate strokes is obtained by heuristic search. Based on the reference-input stroke correspondence, a distance measure can be computed for decision making of character classification. To further improve the recognition performance, the high-level structural analysis and interpretation, and knowledge-based recognition become feasible due to the stroke correspondence.

In the rest of this paper, we first introduce the modeling of reference characters in Section 2; Section 3 describes the pre-processing procedure of input characters; Section 4 presents the model-based stroke extraction and consistent matching method; Section 5 gives experimental results and Section 6 draws concluding remarks.

2. Model building

Chinese characters are composed of strokes and the inter-stroke relation is relatively stable. Hence, it is reasonable to describe Chinese characters in a relational data structure. In previous works, Chinese characters have been described in attributed relational graphs (ARGs). Very often, a stroke is denoted with a node, and

an arc linking two nodes stands for the inter-stroke relation. The stroke attribute and the relation type have various definitions. Generally, stroke attributes include stroke type, length, orientation, and inter-stroke relations include relative position, connection types, parallelism, intersection, etc. The stroke type and relation are usually assigned symbolic values, such as {*Long*, *Short*} for stroke length, {*Horizontal*, *Vertical*, *Left – slant*, *Right – slant*} for stroke orientation, {*Left*, *Right*, *Above*, *Below*} for relative position, {*L – connection*, *T – junction*, *X – intersection*, *Parallel*} for touch relation, etc.

The automatic learning of structural models [31,32] is itself a hard problem and is beyond the scope of this paper. In our experiments, we built reference models from the pen trajectory of on-line input characters for convenience of stroke extraction. In specific, the strokes of a reference model were extracted automatically from a regularly written on-line character while the inter-stroke relations were assigned manually. An assistant program was made to extract strokes from on-line characters, adjust the strokes, and assign inter-stroke relations. The models are different from the so-called on-line model of Hsieh and Lee [10,11] in that we exploit not only the relation between consecutive strokes in writing order but the relation between all closely coupled strokes. And also, we define some different types of stroke relation to better account for the structural invariance. The stroke attributes and relation types of a reference character are stored in a reference graph $G_R = (N_R, E_R)$, where N_R denotes the set of attributed strokes and E_R denotes the stroke relation types. The reference strokes and relation types are initially extracted from on-line input characters automatically and then adjusted interactively by human operators. The size of reference models is normalized to 100×100.

2.1. Stroke types and attributes

We define stroke attributes according to the stroke types in Chinese calligraphy. In description of a reference model, the coordinates of the start and end points of all strokes are stored, from which the stroke length and orientation can be derived. These attributes have continuous values. Each stroke is assigned to one of the 7 types, and each type has different tolerances of length, orientation and bend angles. The tolerances are useful in candidate stroke extraction.

The stroke types are: dot (D), horizontal stroke (H), vertical stroke (V), slash (S), back slash (B), tick (T) and hook (K). These stroke types can be found in a typical Chinese character as shown in Fig. 1. The tolerances of orientation, bend angles, and length are dependent on the stroke type. The orientation tolerance $\Delta\theta$ is the maximum orientation deviation of an input stroke relative to the standard orientation of reference stroke. A hand-produced stroke may bend to left or right side. The bend

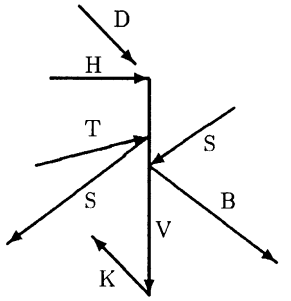


Fig. 1. Stroke types in a Chinese character D = dot, H = horizontal, V = vertical, S = slash, B = back slash,T = tick, K = hook.

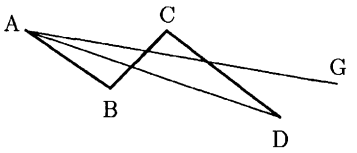


Fig. 2. Bend angles of a polyline stroke $ABCD$ $\angle DAC$ is a left bend and $\angle DAB$ is a right bend.

angle is defined as the angle formed by two end points and the maximum deviation vertex, as shown in Fig. 2, where an input stroke is the concatenation of four vertices $ABCD$, AD is the approximated straight line, B is a right-bend point and C is a left-bend point. The angle $\angle DAB$ is the right-bend angle and $\angle DAC$ is the left-bend angle. The bend ranges are applied to restrict the bend angles in candidate input stroke extraction. The left and right bend ranges are denoted by BL and BR, respectively. The variation of stroke length is restricted by length bounds. We specify a lower bound (LB) and an upper bound (UB) of length for each reference stroke. In candidate stroke extraction, LB is used as an abstract threshold, while UB is used to terminate the extension of candidate input stroke when the length is exceeded.

The tolerances are set according to stroke type. They are also related to the orientation and length of reference stroke. Some heuristics to set the tolerances are as follows. Generally, a dot stroke is very flexible in orientation and length variation. Hook and tick strokes are less variable than dot stroke but are wilder than other stroke types. The orientation variation of a long stroke is smaller than a short stroke. A long vertical stroke is more stable in orientation than a short vertical stroke. And generally, a vertical stroke is more stable in length than other types of strokes. A slash stroke has greater left bend, while a back-slash stroke has greater right bend. All the tolerances are set as flexible as possible so that the true strokes of input character are rarely excluded.

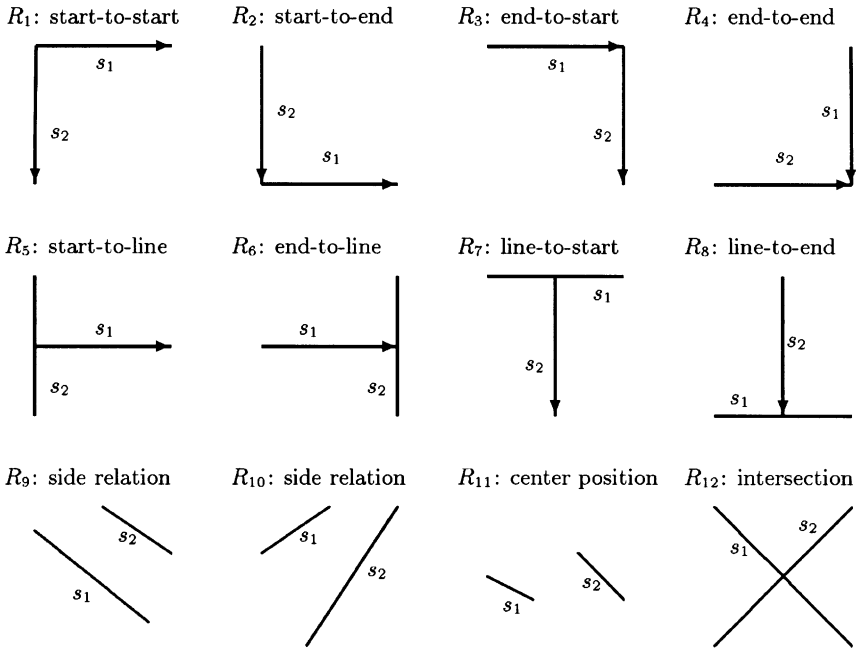


Fig. 3. Inter-stroke relation types.

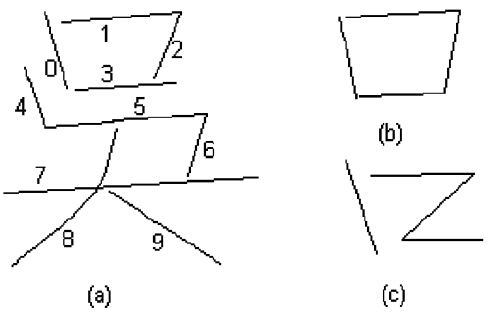


Fig. 4. A reference character with variations.

Table 1
Stroke types and relations of the model in Fig. 4

Type	Index	1	2	3	4	5	6	7	8	9
V	0	7	10	3	10	10	0	0	0	0
H	1		3	9	0	10	0	0	0	0
V	2			6	0	10	0	0	0	0
H	3				0	10	0	0	0	0
V	4					3	0	0	0	0
H	5						3	10	7	0
V	6							6	10	0
H	7								12	7
S	8									7
B	9									

2.2. Stroke relation types

We define 12 types of stroke relation, some of which are similar to those in the literature. The major difference lies in the relative position relation, which was usually assigned *left–right* and *above–below* types and we feel not stable. The relation types are shown in Fig. 3. We denote the relation types with a set $\{R_1, R_2, \ldots, R_{12}\}$. R_1–R_4 are connection relations, as called L-shape in the literature, but the real connection is not required. R_5–R_8 are stroke end to line adjacency relations, as called T-junction in the literature, but, stroke touching is not required. R_9 and R_{10} are side positional relation, under which stroke touching but not crossing is allowed. R_9 says that if a reference stroke s_2^r lies on certain (left/right) side of another reference stroke s_1^r, the corresponding input stroke s_2^i must lie on the same side of s_1^i. Considering that stroke s_2^r lies on one side of s_1^r does not guarantee that s_1^r necessarily lies on one side of s_2^r, we define the relation R_{10} indicating that s_1^r lies on certain side of s_2^r. Some time the side relation is unstable, e.g., when both two strokes are short and wild in orientation. In this case, the center positional relation R_{11} rises, which says that the relative position of centers is stable. R_{12} is the intersection relation, it requires that two strokes must touch each other, and at least one stroke passes across another one.

Some relation types are symmetrical or compatible with one another. R_2 and R_3 are symmetrical, i.e.,

$s_1R_2s_2 \leftrightarrow s_2R_3s_1$. Similarly, we have $s_1R_5s_2 \leftrightarrow s_2R_7s_1$, $s_1R_6s_2 \leftrightarrow s_2R_8s_1$, and $s_1R_9s_2 \leftrightarrow s_2R_{10}s_1$. Some relations are compatible. For example, R_7 is compatible to R_1, i.e., a pair of strokes under relation R_1 does not violate relation R_7. Similarly, R_6 is compatible to R_3 and R_4. The symmetry and compatibility between relation types provides us much flexibility in building reference models.

Fig. 4 shows the strokes of a reference character, and Table 1 lists the stroke types and inter-stroke relations of this model. In the relation table, number 0 denotes null relation, which implies that the relation between two strokes is not critical in structural matching. We define stroke relations only for closely coupled strokes in order to reduce the computation effort in structural matching. Note that the relation between strokes s_0 and s_1 is set as R_7 rather than R_1. These two strokes are written in start-to-start adjacency relation in regular style as in Fig. 4b, but the shape often deforms as in Fig. 4c. R_7 covers both the two cases. Similarly, the relation between strokes s_2 and s_3 is set as R_6 instead of R_4 so as to tolerate deformation.

3. Preprocessing

The preprocessing is performed in three phases: skeletonization, line approximation and graph description. The input character is thinned to reduce the stroke width to unit. Then the skeleton segments each demarcated by two control points (stroke end or junction) are traced. Each skeleton segment is approximated into line segments by corner detection and polyline approximation. All feature points (control and corner points) are stored in a graph as nodes and a line segment linking two feature points is an edge.

For character image thinning, we took the algorithm proposed by Suzuki and Abe [33]. It performs fairly well in respect of skeleton shape from our view. After thinning, all skeleton segments demarcated by control points are traced and spurious branches (whose length is short enough) are removed. The merging of neighboring control points (possibly dichotomy of an intersection) is postponed to the graph level. In line approximation, the high-curvature corner points are detected using Rosenfeld and Johnston's technique [34] to split the curve into smooth pieces. Each piece is again split by the polygonal approximation technique of Ramer [35] to generate more break points.

After line approximation, all feature points and line segments are stored in a graph. In this graph, neighboring nodes are merged, if they are very close to each other compared to the estimated stroke width. The graph is further normalized in size such that the span of feature points in two axes is confined to 100, being coherent to the size of reference characters. The graph of an input character is denoted by $G_I = (N_I, E_I)$, where N_I is the set of feature points and E_I is the set of line segments each linking two feature points. An example of preprocessing is shown in Fig. 5. In Fig. 5, a is the input binary image with skeleton drawn in bold pixels, Fig. 5b is the traced skeleton with feature points drawn in bold pixel, and Fig. 5c is the graph representation with size normalized, where the numbers denote the indices of feature points.

4. Structural matching

4.1. Candidate stroke extraction

An input stroke is an edge (line segment) or a path (polyline) in the input graph $G_I = (N_I, E_I)$. The set of candidate input strokes to match a reference stroke is stored in a candidate list. All candidate strokes to match a reference stroke are extracted by a line following procedure guided by depth-first search. In depth-first search, an OPEN list is used to store the newly generated candidates, and a CLOSED list is used to store the candidate

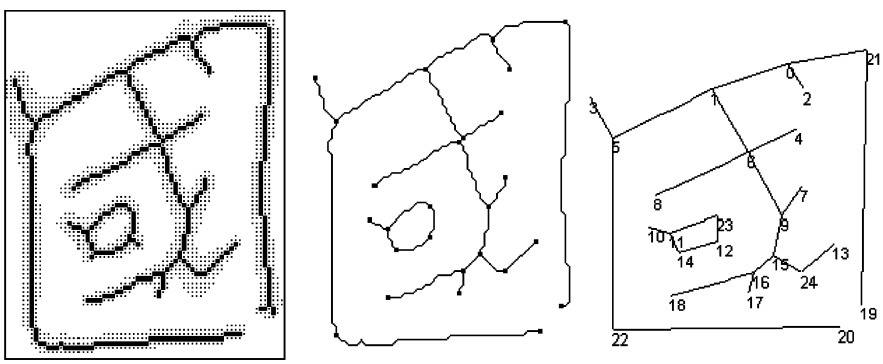


Fig. 5. Preprocessing of character image to obtain graph description.

strokes that have been extended or have failed to extend. The extension of candidate strokes proceeds recursively until the OPEN list is empty. Finally, the candidate strokes are selected from the elements of CLOSED.

The reference and input strokes are considered in a coordinate system with downward y-axis conforming to the pen-movement direction. Denote the reference stroke to match by s^r, which has attributes of start and end points $(p_s^r, p_e^r) = ((x_s^r, y_s^r), (x_e^r, y_e^r))$, center point $p_c^r = (x_c^r, y_c^r)$, length l^r, orientation θ^r, orientation tolerance $\Delta\theta$, bend ranges BL and BR, length bounds LB and UB. Let the candidate input stroke be a sequence of feature points $(p_1^i, p_2^i, \ldots, p_n^i)$, which has overall orientation θ^i and start orientation θ_0^i (the orientation of the first line segment), left- and right-bend angles bl and br. The turning angle from θ^r to θ^i is denoted by $\mathrm{d}\theta = \mathrm{d}\theta(\theta^r, \theta^i)$. The domain of orientation is $\theta \in [-90, 270)$, and the domain of turning angle is $\mathrm{d}\theta \in [-180, 180)$. The turning angle is positive when the input stroke turns right relative to the reference stroke and is negative when turning left. See Fig. 2, where AG is a reference stroke and $ABCD$ is an input stroke, we have $\mathrm{d}\theta = \angle GAD$, $bl = |\angle DAC|$, and $br = |\angle DAB|$. Note that the bend angles bl and br have abstract values but are differentiated into left and right bends.

The candidate stroke extraction is performed in four steps: initialization, depth-first search, candidate stroke verification and candidate stroke sorting. Initially, the reference stroke is matched with a line segment (the start line piece of an input stroke). The distance between the start points of reference and input strokes is restricted by

$$d(p_s^r, p_1^i) \leqslant 40.$$

Recalling that the size of normalized characters is 100×100, this condition is very loose. And the maximum orientation deviation of the start piece is restricted by

$$\begin{cases} \mathrm{d}\theta \leqslant \Delta\theta + BR & \text{if } \mathrm{d}\theta \geqslant 0, \\ -\mathrm{d}\theta \leqslant \Delta\theta + BL & \text{if } \mathrm{d}\theta < 0. \end{cases}$$

This ensures that the overall orientation of the input stroke is within the orientation tolerance $\Delta\theta$.

During the depth-first search, a candidate stroke is extended via connecting the end point to one more adjacent feature point. In extension, the local condition of feature point deviation and the global condition of reference stroke constraints are examined. For three consecutive feature points, say, BCD in Fig. 2, the turning angle from BC to CD and the distance from point C to line BD should be small. The global condition guarantees that the candidate stroke and its extensions do not exceed the orientation tolerance:

$$\mathrm{d}\theta + br \leqslant \Delta\theta + BR \quad \text{and} \quad bl - \mathrm{d}\theta \leqslant \Delta\theta + BL.$$

The search algorithm is described as below.

Algorithm 1. Tree search for candidate stroke extraction

1. Initialization. All line segments initially matched with a reference stroke is stored in OPEN list. The CLOSED list is initially empty.
2. Extension.
   2.1. Move the last element of OPEN into CLOSED, and extend it. If the length of the candidate stroke exceeds the upper bound UB, it does not extend. Append the newly generated candidates into OPEN.
   2.2. Repeat 2.1 until the OPEN list is empty.
3. Select candidate strokes from the elements of CLOSED.

After depth-first search, the CLOSED list contains some candidate strokes that mismatch the reference stroke. So we have a verification phase to remove these mismatched strokes. The surviving candidates satisfy the conditions

$$d(p_c^r, p_c^i) \leqslant 40, \quad |\mathrm{d}\theta| \leqslant \Delta\theta, \quad bl \leqslant BL \quad br \leqslant BR.$$

In addition, the length of the input stroke from start to end point must be greater than the lower bound LB. The matched candidate strokes are sorted in increasing order of inter-stroke distance. The distance between the reference stroke and an input stroke is calculated by

$$d_s(s^r, s^i) = \lambda_\theta |\mathrm{d}\theta| + \lambda_l |l^r - l^i| + \lambda_{pos} d(p_c^r, p_c^i) + \lambda_{merge}(a_l h_l + a_r h_r), \quad (1)$$

where λ_θ is the coefficient of orientation deviation, λ_l is the coefficient of length difference, λ_{pos} is the coefficient of center position distance and λ_{merge} is the coefficient of line segment merging cost. h_l is the maximum point-to-line distance on left side of input stroke and h_r is the maximum point-to-line distance on right side, while a_l and a_r are coefficients for them. $a_l = 0.5$ for slash strokes and $a_l = 1$ otherwise, $a_r = 0.5$ for back slash strokes and $a_r = 1$ otherwise.

4.2. Consistent matching

In stroke matching, each reference stroke is matched with an input stroke and the input strokes are consistent with respect to the inter-stroke relations. The stroke matching can be viewed as a consistent labeling problem [36] like graph matching. Our problem of stroke matching is special in that a node of the reference model G_R is matched with a path of the input character G_I. We will formulate the node-to-path matching as a combinatorial optimization problem and solve it using heuristic search strategies.

Graph matching is to seek a consistent mapping of nodes and edges between two graphs under the relational constraints. The error-correcting matching of ARGs has been attacked by a number of researchers [37–40]. Tsai et al. [39] and Wong et al. [40] formulated the ARG

matching as a problem of objective optimization and applied heuristic search algorithms [41,42] to solve it. We solve the node-to-path matching problem similarly in the framework of combinatorial optimization, wherein the reference-input stroke correspondence is obtained by seeking the optimal solution of an objective function using heuristic search strategies. We further use some structural post-processing operations to improve the stroke correspondence. As for heuristic search, we use the A* algorithm to seek the optimal solution, or use a semi-admissible search algorithm to seek a sub-optimal solution in much less time.

Suppose G_R has n_R nodes and a node is denoted by s_i^r, its corresponding candidate input stroke is denoted by s_i^i. Denote the distance between the reference and input strokes as $d_s(s_i^r, s_i^i)$ and the cost of stroke relation matching of stroke pairs as $d_e((s_i^r, s_k^r), (s_i^i, s_k^i))$. The distance between G_I and G_R is

$$D(G_R, G_I) = \sum_{i=1}^{n_R} \left[d_s(s_i^r, s_i^i) + \sum_{k,(s_i^r, s_k^r) \in E_R} d_e((s_i^r, s_k^r), (s_i^i, s_k^i)) \right]. \tag{2}$$

The optimal matching problem is to seek a node-to-path mapping so that the distance $D(G_R, G_I)$ is minimized. In order to reduce the computation effort, the relation cost is not calculated. Instead, the consistency of relation between two stroke pairs is examined. If two stroke pairs are not compatible with respect to the relation constraint, the relation cost is infinity, otherwise 0. Thus, the objective function can be re-written as

$$D(G_R, G_I) = \sum_{i=1}^{n_R} \frac{d_s(s_i^r, s_i^i)}{\prod_{k,(s_i^r, s_k^r) \in E_R} C((s_i^r, s_k^r), (s_i^i, s_k^i))}, \tag{3}$$

where the compatibility function $C((s_i^r, s_k^r), (s_i^i, s_k^i))$ has value 1 if two stroke pairs are compatible, otherwise 0. The compatibility is examined only if the two reference strokes are related in the model character. The constraints of compatibility are dependent on the stroke relation type. But in all case, two input strokes matched with two different reference strokes should not share common line segment except when the two corresponding reference strokes have relation of intersection R_{12} or stroke end connection R_1–R_4 and the common line segment is short.

The constraints of compatibility for the relation types are given in Table 2. For convenience of illustration, we denote the reference strokes as s_1^r and s_2^r, and the corresponding input strokes as s_1^i and s_2^i. If the superscript is not given, the notation refers to either reference or input stroke. The start and end points of a stroke are denoted by p_s and p_e, respectively, and the center point is denoted by p_c. Note that some of relation types are not treated in Table 2. Since R_2 is symmetric to R_3, examination of R_2 can be readily converted to relation R_3 by swapping the two strokes. Similarly, R_5 is converted to R_7, R_8 is converted to R_6, and R_{10} is converted to R_9.

Table 2
Type specific constraints of stroke relation matching

Relation type	Constraints (necessary conditions on the input stroke pair)
R_1–R_4	The side relation of point p_{c1} relative to s_2, and the side relation of point p_{c2} relative to s_1 are consistent between the reference pair and the input pair.
R_6	Not crossing. The side relation of p_{s1} relative to s_2 is consistent. The projection point of p_{e1}^i to s_2^i, or the crossing point of extended s_1^i with s_2^i is within the line segment of s_2^i.
R_7	Not crossing. The side relation of p_{s2} relative to s_1 is consistent. The projection point of p_{s2}^i to s_1^i, or the crossing point of extended s_2^i with s_1^i is within the line segment of s_1^i.
R_9	Both p_{s2}^i and p_{e2}^i are one the same side of s_1^i, and the side relation is consistent to the side relation of s_2^r relative to s_1^r. In addition, the constraints of R_{11} should be met.
R_{11}	Considering the orientation of the line connecting p_{c1} and p_{c2}, the orientation difference between the reference pair and the input pair is confined. The threshold of difference is dependent on the distance $d(p_{c1}^r, p_{c2}^r)$. The shorter is the distance, the larger is the threshold.
R_{12}	Two input strokes must touch each other, and at least one stroke crosses through another one.

On compatibility of stroke relation, the cost of reference-input stroke matching is calculated as in Eq. (1). A reference stroke may also correspond to a null input stroke, like the node deletion operation in graph matching [38]. The cost of null correspondence is computed as

$$d_s(s^r, 0) = \lambda_{ndel} l^r + C_{ndel}, \tag{4}$$

where λ_{ndel} is a coefficient to the reference stroke length and C_{ndel} is a constant. When a reference stroke is matched with a null input stroke, the relation of this stroke with any other stroke is considered as compatible, i.e., the compatibility function takes value 1.

The search space of stroke matching is represented as a tree data structure. A node in the search tree stands for a reference-input stroke pair and a path from the root node to an intermediate node stands for a partial character matching. A node is denoted as $N = (s_i^r, s_i^i)$, where s_i^i is a real input stroke or a null stroke. The root node

$N_0 = (0,0)$ represents a null matching of strokes. A path from the root node to an intermediate node is denoted by

$$P = \{(s_1^r, s_1^i), (s_2^r, s_2^i), \ldots, (s_i^r, s_i^i)\}. \quad (5)$$

The cost of a node is

$$c_N(N) = c_N(s_i^r, s_i^i) = \frac{d_s(s_i^r, s_i^i)}{\prod_{k=1}^{i-1} C((s_i^r, s_k^r), (s_i^i, s_k^i))}, \quad (6)$$

and the cost of a path is

$$c_P(N) = c_P(s_i^r, s_i^i) = \sum_{k=1}^{i} c_N(s_k^r, s_k^i). \quad (7)$$

The essential operation in search space is the expansion of intermediate nodes, wherein the succeeding reference stroke is matched with its candidate input strokes to generate new nodes. The nodes are stored in two lists depending on whether they have been expanded or not. The nodes that have not been expanded are stored in an OPEN list, and those which have been expanded are stored in a CLOSED list. The root node is initially in OPEN. When an open node is to be expanded, it is moved from OPEN to CLOSED, and the new nodes generated in expansion are added into OPEN. Specifically, in the expansion of a node $N = (s_i^r, s_i^i)$, a new reference stroke s_{i+1}^r is matched with its candidate input strokes s_{i+1}^i under relation constraints to generate new nodes (s_{i+1}^r, s_{i+1}^i), or correspond to a null stroke to generate a new node $(s_{i+1}^r, 0)$.

Depending on the order of node expansion, the search strategies are classified into breadth-first search, depth-first search, and best-first search. The A* algorithm is enhanced from best-first search with estimation of remaining path cost. We first describe the best-first search as below.

Algorithm 2. Best-first search

1. Initialization. Generate a root node $N_0 = (0,0)$ with cost $f(N_0) = 0$. N_0 is initially stored in OPEN list, and the CLOSED list is initially empty.
2. Expansion.
   2.1. Select the best node in OPEN which has least path cost, move it to CLOSED and expand it. The new nodes generated in expansion are added to OPEN.
   2.2. Repeat 2.1 until the selected best node is a goal node (all reference strokes have been matched).
3. Back-track the path from the goal node to the root node to record the corresponding input strokes of all reference strokes.

In heuristic search, each node in the search space has a heuristic function to estimate the remaining cost of a complete solution (a path from the root to a goal node). The estimated cost is added to the partial cost to order the nodes. It was proven that if the estimated cost is a lower bound of the true remaining cost, the best-first search (A* in this case) is guaranteed to find the global optimal solution [41,42]. Including the estimated remaining cost is efficient to speed up the search process via expanding much fewer nodes.

Some heuristic functions have been proposed for graph matching [39,40]. The heuristic function that we use is a simplification of previous forms, and is similar to the forward-checking function of Ref. [37]. The heuristic function is the sum of minimum cost for each remaining reference stroke under the constraints of relation between this stroke and all preceding strokes in the partial matching corresponding to an intermediate node $N = (s_i^r, s_i^i)$:

$$h(N) = \sum_{j=i+1}^{n_R} \min_{s_j^i} \frac{d_s(s_j^r, s_j^i)}{\prod_{k=1}^{i} C((s_j^r, s_k^r), (s_j^i, s_k^i))}. \quad (8)$$

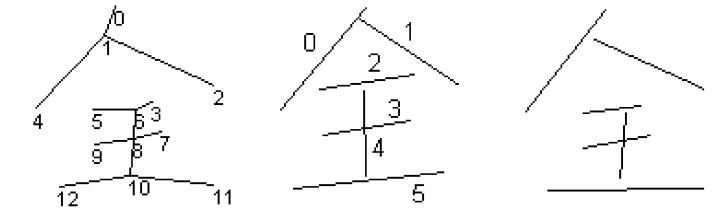
In A* search, the sum of partial cost and estimated remaining cost

$$f(N) = c_P(N) + h(N) \quad (9)$$

is used instead of $c_P(N)$ alone to order the nodes in OPEN list. Each time the node in OPEN list that has least sum of cost is selected to expand.

Fig. 6 shows an example of A* search for consistent stroke matching. Fig. 6a is the input character, with numbers denoting the vertex indices; Fig. 6b is the reference model, with numbers denoting the stroke indices; And Fig. 6c is the result of stroke matching, with the extracted input strokes drawn as straight lines. Fig. 6d lists the candidate input strokes for each reference stroke. A candidate stroke is represented as a sequence of vertices, with the reference-input stroke distance given in parentheses. Fig. 6e shows the search tree, where each node is represented as a reference-input stroke pair and is attached with the partial cost and estimated remaining cost in parentheses (the goal nodes have no remaining cost). The sequential number of node generation is given left to the node frame. In expanding the root node, the first reference stroke s_0^r is matched with two candidate input strokes as well as a null input stroke ("*d*" denotes "deletion"). The three nodes 1, 2, and 3 are stored in OPEN list, whereas the root node is moved to CLOSED list. At this moment, the node 2 has minimum sum of partial cost and remaining cost. It is moved to CLOSED and is expanded to generate nodes 4 and 5. Then the expansion of node 5 generates nodes 6–11. In this way, the node of minimum sum cost in OPEN is expanded recursively until the minimum cost node is a goal node. Backtracking the goal node 22 gives the consistent input strokes as shown in Fig. 6c.

The A* search algorithm is admissible in that if the heuristic function $h(N)$ is a lower bound estimate of the remaining cost, the search will converge to the global optimal solution. However, to search for the global optimum is expensive because in practical problems, there



(a) Input character (b) Reference character (c) Matched result

(d) Candidate strokes

s0: 0-1-4 (11), 1-4 (33)
s1: 1-2 (22)
s2: 5-6-3 (43), 9-8-7 (52), 5-6 (65), 9-8 (66), 6-3 (93), 1-2 (105)
s3: 9-8-7 (22), 5-6-3 (38), 9-8 (39), 12-10 (45), 5-6 (54), 8-7 (58), 10-11 (73), 6-3 (83), 12-10-11 (97)
s4: 6-8-10 (28), 8-10 (51)
s5: 12-10-11 (39), 9-8-7 (84)

(e) A* search

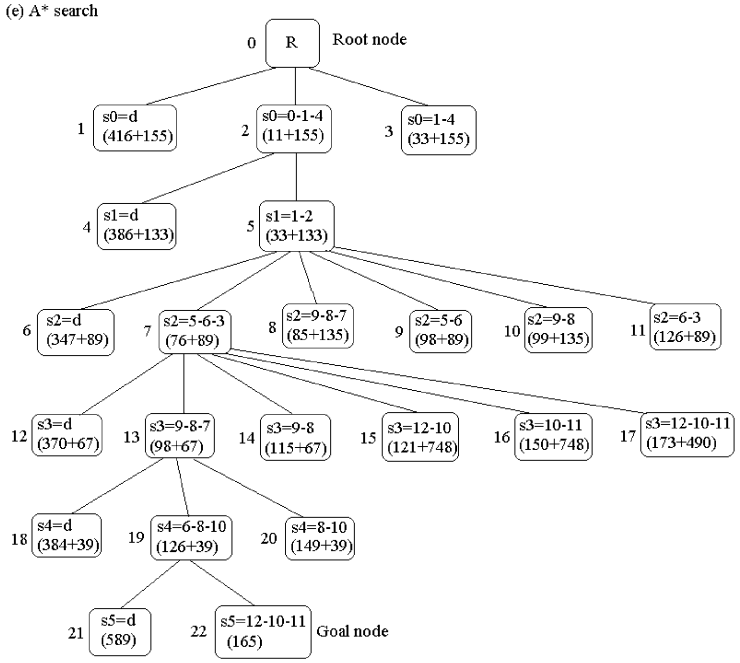


Fig. 6. An example of A* search for stroke matching.

may be many feasible solutions whose costs do not vary significantly from each other. In this case, it spends undesirably long time to differentiate between the competing paths. To speed up the search process, the ε-admissible algorithm A*ε was proposed to search an approximate solution whose cost is not more than $1+\varepsilon$ of the optimum [43,44].

We view the sub-optimal search as a hybrid of best-first search and depth-first search, where the search efficiency is improved by encouraging the expansion of deeper nodes. In our matching problem, the search space has n_R levels corresponding to n_R nodes of G_R. The terminal node has depth n_R and the intermediate nodes have depth $1,\ldots,i,\ldots,n_R-1$. To speed up the search process, we replace the cost function $f(N)$ with

$$f'(N) = f(N)\times\left(1+\varepsilon\frac{n_R-i}{n_R}\right) \tag{10}$$

and select the node in OPEN with minimum $f'(N)$ to expand. This algorithm can be proven ε-admissible in the same way as Ref. [44].

4.3. Post-processing and distance measure

The stroke correspondence obtained by heuristic search is not necessarily consistent to human perception

due to the imperfection of cost functions. The imperfection of matching may also be produced by writing variations. For example, in handwritten Chinese characters, two strokes may accidentally connect end-to-end with no prominent deviation point on skeleton such that they cannot be split in stroke extraction stage. To improve the matching result, we built a structural post-processing module performing 3 tasks: stroke extension, stroke re-split and redundancy interpretation.

The input stroke corresponding to a reference stroke is extended to be as long as possible under the constraints of common line segment sharing and stroke relations. For each reference stroke, consider its candidate input strokes that embrace the current corresponding input stroke, check the constraints of line segment sharing and relations with the corresponding input strokes of all other reference strokes. Select the longest candidate input stroke that pass all constraints as the new correspondence.

In stroke re-split stage, we investigate all the stroke pairs with relation R_5–R_8 or R_{12}. For a pair of reference strokes s_1^r and s_2^r with relation R_{12}, if one end of a corresponding input stroke does not protrude from the crossing point. Say, if the start of s_1^i does not protrude, then investigate the line segment linked to the crossing point and occupied by a third reference stroke, say, s_3^r. If this line segment is collinear to s_1^i, we let the two reference strokes s_1^r and s_3^r occupy half of the line segment, respectively, by generating a new break point in middle of the line segment and adjust the corresponding input strokes. Similarly, we examine the protrusion of s_2^i for R_5 and R_6, and the protrusion of s_1^i for R_7 and R_8, and adjust the corresponding input strokes if appropriate.

In the input character, a redundant stroke can be appropriately interpreted and discounted in the inter-character distance. Generally, for input strokes corresponding to two reference strokes, if the end of one stroke is connected to the start of another stroke by an un-occupied input stroke, this un-occupied input stroke is probably redundant. Another kind of redundant stroke is the hook, which is linked to the end of another long stroke. In previous works, redundant hooks were removed prior to stroke matching [45]. By incorporating model information, the detection of redundant hook is more reliable. In our implementation, we have detected the redundant hooks but not other redundant strokes.

Fig. 7 shows two examples of post-processing, where some line segments are split by new break points (marked with a cross). In each row, the left image is the input character, the middle one is the reference model, and the right one is the matching result with extracted input strokes drawn as straight lines.

After post-processing, a distance measure is computed between the reference and input characters. This distance measure will be used in recognition, wherein an input character is matched with a number of hypothesized reference models (selected by coarse classification) and the reference model with minimum distance to the input character is the recognition result. The distance is calculated by

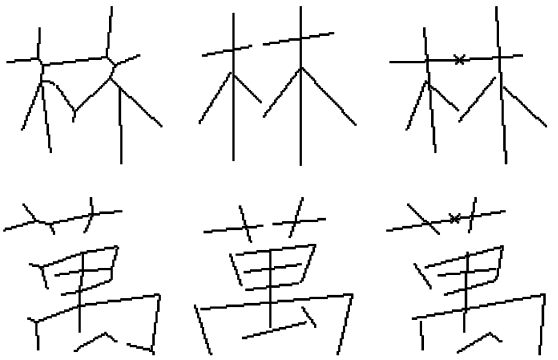


Fig. 7. Examples of stroke re-split after matching.

$$D'(G_R, G_I) = \left[\sum_{i=1}^{n_R} d_s(s_i^r, s_i^i) + \sum_{i=1}^{n_E} d_i(e_i)\right] \Big/ \sum_{i=1}^{n_R} l_i^r, \qquad (11)$$

where $d_s(s_i^r, s_i^i)$ is the distance between a reference stroke s_i^r and its corresponding input stroke s_i^i, as in Eqs. (1) or (4). n_E is the number of edges (line segments) in G_I. The cost $d_i(e_i)$ is zero when the line segment e_i is occupied by a reference stroke; else if e_i is interpreted to be redundant, $d_e(e_i) = 0.2\lambda_{ndel} \cdot l^i(e_i)$, where $l^i(e_i)$ is the length of the line segment e_i; otherwise $d_e(e_i) = \lambda_{ndel} \cdot l^i(e_i)$. The distance is normalized with respect to the total stroke length of the reference character.

5. Experimental results

We implemented experiments on a database of handwritten Chinese characters collected by the AI laboratory of KAIST (Korea Advanced Institute of Science and Technology). The database has 783 Chinese characters, which are frequently used in Korean names. Each character has 200 sample images, with indices 1–200. The images are sorted approximately in decreasing order of image cleanness. We refer to a set of samples with one image from each class as a data set. We used 80 sets of odd number index $(1, 3, \ldots, 159)$ for training a coarse classifier and 20 sets of even number index $(2, 4, \ldots, 40)$ for testing. As stated in Section 2, the reference models were constructed from on-line input characters and the off-line samples were not used to adjust the models.

The coarse classifier extracts contour direction features from the character image without size normalization but the features are normalized with respect to the character size. The details of feature extraction can be found in

Ref. [46]. The dimensionality of feature vector is 144 and the template vector of each class is obtained by averaging the training vectors of this class. For an input pattern, the city block distance between the input feature vector and the template vector of each class is computed and the top 10 classes with minimum distances are selected as candidates for structural matching. On the 20 test data sets, the cumulative accuracy of coarse classification is 99.78%. In fine classification, the structural description of the input character is matched with the reference models of the candidate classes by heuristic search and a distance measure as in Eq. (11) is computed for each candidate class. Finally, the class with the minimum distance is the recognition result. An input character is rejected when the minimum distance is greater than 2.0.

The parameters of structural matching were set as $\lambda_{ndel} = 5$, $C_{ndel} = 50$, $\lambda_{\theta} = 2$, $\lambda_{pos} = 1$, $\lambda_{l} = 1$, and $\lambda_{merge} = 2$. Actually, the matching result is insensitive to these parameters. We set the parameters from our intuition while the variation of parameters to certain extent does not influence the matching result. The experiments were implemented on PC/300 MHz with programming language C + + on Windows 95. The A* search algorithm and the semi-admissible search with variable ε have been tested for stroke matching. The recognition speed (average time for recognizing one character image) and accuracies are given in Table 3.

It is reasonable that the A* search algorithm ($\varepsilon = 0$) yields the best recognition performance. The correct recognition rate is as high as 97.89% and the error rate is 1.50%. However, the recognition speed of A* algorithm, averagely 0.85 s for one image, is too slow. The semi-admissible search improves the recognition speed with a little loss of recognition accuracy. When $\varepsilon = 4.0$, the recognition speed (0.22 s) is much faster than that of A* search, but the loss of accuracy (96.65 versus 97.89%) is considerable. The parameter $\varepsilon = 2.0$ provides a good compromise between recognition accuracy (97.44%) and speed (0.30 s). Note that the recognition rates of $\varepsilon = 1.0$ and 2.0 are higher than that of $\varepsilon = 0.5$ even though smaller value of ε is deemed to converge to better solution. This phenomenon might result from the structural post-processing, in which the incomplete matching of sub-optimal search was remedied.

Table 3
Recognition results (speed and accuracy)

Parameter (ε)	Speed (s)	Correct (%)	Reject (%)	Error (%)
0 (A*)	0.85	97.89	0.61	1.50
0.5	0.70	97.43	0.82	1.75
1.0	0.50	97.52	0.77	1.71
2.0	0.30	97.44	0.84	1.72
4.0	0.22	96.65	1.24	2.11

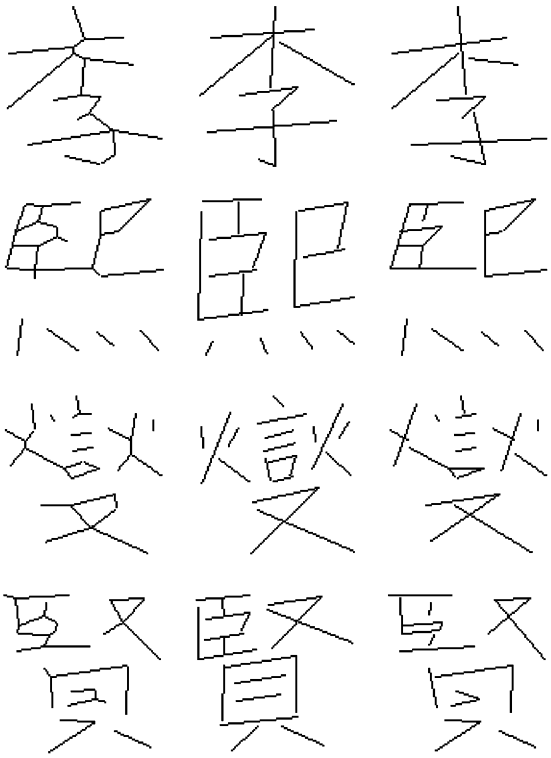


Fig. 8. Examples of correct recognition.

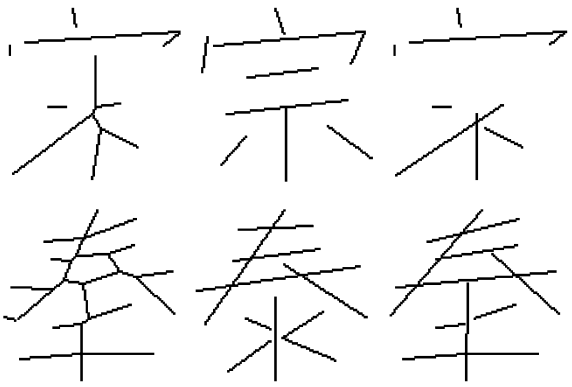


Fig. 9. Examples of mis-recognition.

Generally speaking, the proposed model-based stroke matching method is powerful and robust to extract reliable strokes and obtain correct correspondence. Particularly, It can overcome the stroke ambiguity that cannot be solved prior to incorporating the model information, including stroke interference, bending, and touching. Some examples of correct recognition are shown in Fig. 8. Examples of mis-recognition and rejection are shown in Figs. 9 and 10, respectively.

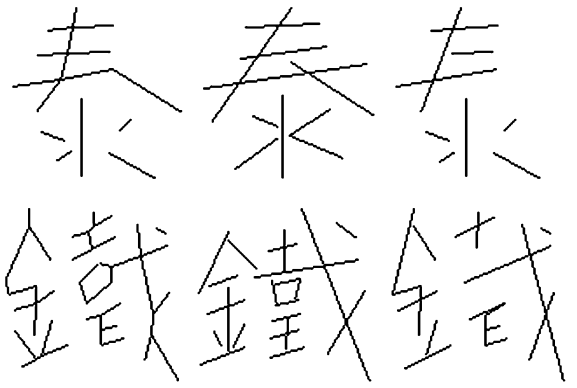


Fig. 10. Examples of rejection.

Mis-recognition and rejection are generally caused by inappropriate inter-character distance measure (e.g., the lower example of Fig. 9), structural corruption of input character (e.g., the upper example of Fig. 9), or abnormal writing styles (as in Fig. 10). The inappropriate distance measure is a major source of mis-recognition. Because we did not account for the stroke relation cost in inter-character distance, similar characters are prone to be confused. However, based on the structural matching, it is possible to construct a knowledge-based system to discriminate confusing characters.

6. Concluding remarks

In this paper we proposed a structural matching method for recognition of handwritten Chinese characters via model-based stroke extraction and heuristic search-based stroke correspondence. The intention of model-based stroke extraction is to overcome the ambiguities prevalent in practical character images. The model-based stroke matching is accomplished in two stages: candidate input stroke extraction and consistent matching. The two-stage strategy is aimed to avoid repeated generation of an input stroke even though to extract strokes during heuristic search is possible. As for heuristic search, the A* search algorithm and the semi-admissible search algorithm have been tested. The semi-admissible search provides a good compromise between recognition accuracy and speed.

The proposed method is readily applicable to on-line character recognition, where the stroke extraction is much easier and the reference models as in this paper can be used directly. It is worthy discussing, however, when the proposed method is applied to larger character set, say, more than 3000 classes. This situation gives rise to two problems: computation efficiency of classification and shape confusion between similar characters. The first problem can be well solved by precise candidate selection [47]. It was shown in Ref. [47] that in case of 3036 classes, a probabilistic method can select about 10 candidates with precision 99.67%. Since the structural matching is performed on the candidate classes only, the precise candidate selection renders it feasible for large character set. To solve the second problem, the structural matching method needs some elaborations. We herein suggest two possibilities. First, refined inter-character distance measure or knowledge-based decision can be applied based on the stroke correspondence obtained from heuristic search. Second, the attributes of structural models can be adjusted in discriminative learning to reflect the boundary between similar characters.

More suggestions toward performance improvement are as follows. (1) Enhancing the candidate stroke extraction module to handle broken strokes. (2) Pruning candidate input strokes prior to consistent matching to accelerate recognition. (3) Radical-based structural description and hierarchical matching. This is helpful to speed up recognition and grasp global structure. (4) Automatic construction of structural models in order to alleviate manual labor and to improve the recognition performance.

Acknowledgements

The authors would like to thank the anonymous referees, whose comments led to the improvement of presentation quality of this paper.

References

[1] S. Mori, K. Yamamoto, M. Yasuda, Research on machine recognition of handprinted characters, IEEE Trans. Pattern Anal. Mach. Intell. 6 (4) (1984) 386–405.
[2] T.H. Hilderbrand, W. Liu, Optical recognition of Chinese characters: advances since 1980, Pattern Recognition 26 (2) (1993) 205–225.
[3] M. Umeda, Advances in recognition methods for handwritten Kanji characters, IEICE Trans. Inform. Systems E79-D (5) (1996) 401–410.
[4] K. Yamamoto, A. Rosenfeld, Recognition of handprinted Kanji characters by a relaxation method, Proceedings of Sixth ICPR, 1982, pp. 395–398.
[5] H. Yamada, Contour DP matching method and its application to handprinted Chinese character recognition, Proceedings of Seventh ICPR, 1984, pp. 389–392.
[6] C.H. Leung, Y.S. Cheung, Y.L. Wong, A knowledge-based stroke-matching method for Chinese character recognition, IEEE Trans. System Man Cybernet 17 (6) (1987) 993–1003.
[7] S.-L. Chou, W.-T. Tsai, Recognizing handwritten Chinese characters by stroke-segment matching using an iteration scheme, Int. J. Pattern Recognition Artif. Intell. 5 (1/2) (1991) 175–197.

[8] L.-H. Chen, J.-R. Lieh, Handwritten character recognition using a 2-layer random graph model by relaxation matching, Pattern Recognition 23 (11) (1990) 1189–1205.
[9] S.W. Lu, Y. Ren, C.Y. Suen, Hierarchical attributed graph representation and recognition of handwritten Chinese characters, Pattern Recognition 24 (7) (1991) 617–632.
[10] C.-C. Hsieh, H.-J. Lee, Off-line recognition of handwritten Chinese characters by on-line model-guided matching, Pattern Recognition 25 (11) (1992) 1337–1352.
[11] C.-C. Hsieh, H.-J. Lee, A probabilistic stroke-based Viterbi algorithm for handwritten Chinese character recognition, Int. J. Pattern Recognition Artif. Intel. 7 (2) (1993) 329–352.
[12] F.-H. Cheng, W.-H. Hsu, C.-A. Chen, Fuzzy approach to solve the recognition problem of handwritten Chinese characters, Pattern Recognition 22 (2) (1989) 133–141.
[13] K.P. Chan, Y.S. Cheung, Fuzzy-attribute graph with application to Chinese character recognition, IEEE Trans. System Man Cybernet. 22 (1) (1992) 153–160.
[14] P.N. Suganthan, H. Yan, Recognition of handprinted Chinese characters by constrained graph matching, Image Vision Comput. 16 (3) (1998) 191–201.
[15] F.-H. Cheng, Multi-stroke relaxation matching method for handwritten Chinese character recognition, Pattern Recognition 31 (4) (1998) 401–410.
[16] H.-J. Lee, B. Chen, Recognition of handwritten Chinese characters via short line segments, Pattern Recognition 25 (5) (1992) 543–552.
[17] X. Huang, J. Gu, Y. Wu, A constrained approach to multifont Chinese character recognition, IEEE Trans. Pattern Anal. Mach. Intell. 15 (8) (1993) 838–843.
[18] H. Ogawa, K. Taniguchi, Thinning and stroke segmentation for handwritten Chinese character recognition, Pattern Recognition 15 (4) (1982) 299–308.
[19] C.W. Liao, J.S. Huang, Stroke segmentation by Bernstein-Bezier curve fitting, Pattern Recognition 23 (5) (1990) 475–484.
[20] H.-D. Chang, J.-F. Wang, A robust stroke extraction method for handwritten Chinese characters, Int. J. Pattern Recognition Artif. Intell. 8 (5) (1994) 1223–1239.
[21] K. Liu, Y.S. Huang, C.Y. Suen, Robust stroke segmentation method for handwritten Chinese character recognition, Proceedings of Fourth ICDAR, 1997, pp. 211–215.
[22] J-R. Lin, C.-F. Chen, Stroke extraction for Chinese characters using a trend-followed transcribing technique, Pattern Recognition 29 (11) (1996) 1789–1805.
[23] D.S. Yeung, H.S. Fong, A fuzzy substroke extractor for handwritten Chinese characters, Pattern Recognition 29 (12) (1996) 1963–1980.
[24] C. Lee, B. Wu, A Chinese character stroke extraction algorithm based on contour information, Pattern Recognition 31 (6) (1998) 651–663.
[25] H.-H. Chang, H. Yan, Analysis of stroke structures of handwritten Chinese characters, IEEE Trans. System Man Cybernet. Part B: Cybernet. 29 (1) (1999) 47–61.
[26] J.W. Kim et al., Decomposition of Chinese character into strokes using mathematical morphology, Pattern Recognition Lett. 20 (3) (1999) 285–292.
[27] H.T. Tsui, Guided stroke structure extraction for the recognition of handprinted Chinese characters, Proceedings of Sixth ICPR, 1982, pp. 786–788.
[28] H. Nishida, Model-based shape matching with structural feature grouping, IEEE Trans. Pattern Anal. Mach. Intell. 17 (3) (1995) 315–320.
[29] J. Rocha, T. Pavlidis, A shape analysis model with applications to a character recognition system, IEEE Trans. Pattern Anal. Mach. Intell. 16 (4) (1994) 393–404.
[30] S. Chaudhury, A. Acharyya, S. Subramanian, Recognition of occluded objects with heuristic search, Pattern Recognition 23 (6) (1990) 617–635.
[31] H. Nishida, S. Mori, An algebraic method to automatic construction of structural models, IEEE Trans. Pattern Anal. Mach. Intell. 15 (12) (1993) 1298–1311.
[32] A. Amin, S. Singh, Recognition of hand-printed Chinese characters using decision trees/machine learning C4.5 system, Pattern Anal. Appl. 1 (2) (1998) 130–144.
[33] S. Suzuki, K. Abe, Binary picture thinning by an iterative parallel two-subcycle operation, Pattern Recognition 20 (3) (1987) 297–307.
[34] A. Rosenfeld, E. Johnston, Angle detection on digital curves, IEEE Trans. Comput. 22 (1973) 875–878.
[35] U. Ramer, An iterative procedure for the polygonal approximation of plane closed curves, Comput. Graphics Image Process. 1 (1972) 244–256.
[36] R.M. Haralick, The consistent labeling problem: part I, IEEE Trans. Pattern Anal. Mach. Intell. 1 (2) (1979) 173–184.
[37] L.G. Shapiro, R.M. Haralick, Structural descriptions and inexact matching, IEEE Trans. Pattern Anal. Mach. Intell. 3 (5) (1981) 504–519.
[38] A. Sanfelieu, K-S. Fu, A distance measure between attributed relational graphs for pattern recognition, IEEE Trans. System Man Cybernet. 13 (3) (1983) 353–362.
[39] W.-H. Tsai, K.-S. Fu, Subgraph error-correcting isomorphisms for syntactic pattern recognition, IEEE Trans. System Man Cybernet. 13 (1) (1983) 48–62.
[40] A.K.C. Wong, M. You, S.C. Chan, An algorithm for graph optimal monomorphism, IEEE Trans. System Man Cybernet. 20 (3) (1990) 628–636.
[41] N.J. Nillson, Principles of Artificial Intelligence, Springer, Berlin, 1980.
[42] P.H. Winston, Artificial Intelligence, 3rd Edition, Addison-Wesley, Reading, MA, 1992.
[43] J. Pearl, J.H. Kim, Studies in semi-admissible heuristics, IEEE Trans. Pattern Anal. Mach. Intell. 4 (4) (1982) 392–399.
[44] J. Pearl, Heuristics: Intelligent Search Strategies for Computer Problem Solving, Addison-Wesley, Reading, MA, 1984.
[45] A.-B. Wang, K.C. Fan, J.S. Huang, Recognition of handwritten Chinese characters by modified relaxation methods, Image Vision Comput. 12 (8) (1994) 509–520.
[46] C.-L. Liu, M. Nakagawa, A probabilistic model for candidate selection in recognition of large character set, Proceedings of Sixth IWFHR, Taejon, Korea, 1998, pp. 259–268.
[47] C.-L. Liu, M. Nakagawa, Precise candidate selection for large character set recognition by confidence evaluation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (6) (2000) 636–642.

About the Author—CHENG-LIN LIU received the B.S. degree in Electronics from Wuhan University, the M.E. degree in Electronic Engineering from Beijing Polytechnic University, the Ph.D. degree in Pattern Recognition from the Institute of Automation, Chinese Academy of Sciences, in 1989, 1992 and 1995, respectively. He was a post-doctoral fellow in Korea Advanced Institute of Science and Technology (KAIST) and later in Tokyo University of Agriculture and Technology from March 1996 to March 1999. Since then he has been a researcher at the Central Research Laboratory, Hitachi, Ltd., where he is engaged in development of algorithms for pattern classification and handwriting recognition. His research interests include pattern recognition, artificial intelligence, neural networks, image processing, character recognition and document analysis.

About the Author—IN-JUNG KIM received the B.S. and M.S. degrees in Computer Science from Korea Advanced Institute of Science and Technology (KAIST), in 1994 and 1995, respectively. Currently, he is a Ph.D. candidate at KAIST and he is going to receive the Ph.D. degree in Feb. 2001. His research interests include artificial intelligence and pattern recognition, especially handwritten character recognition.

About the Author—JIN H. KIM received the B.S. degree in Engineering from Seoul National University, Korea, in 1971, and the M.S. and Ph.D. degrees in Computer Science from University of California, Los Angeles, in 1979 and 1983, respectively. He was a Research Engineer at Korea Institute of Science and Technology from 1973 to 1976, and Engineering Programmer at the Country of Orange, California, USA, from 1976 to 1977, and a senior staff member in Computer Science at Hughes Artificial Intelligence Center, Calabasas, California, USA, from 1981 to 1985. He joined the faculty of KAIST in 1985. He was a visiting scientist at IBM Watson Research Center from 1990 to 1991. He is now on the editorial board of International Journal of Information Processing and Management, International Journal of Robotics and Autonomous Systems, and Journal of Computer Processing of Oriental Languages. He was conferred the fellowship of IAPR in 2000. His research interests include pattern recognition and artificial intelligence.