A COMPARATIVE STUDY OF STROKE CORRESPONDENCE SEARCH ALGORITHMS FOR ONLINE KANJI CHARACTER RECOGNITION

Wenjie CAI, Seiichi UCHIDA, and Hiroaki SAKOE

Graduate School of Information Science and Electrical Engineering, Kyushu University

ABSTRACT

The stroke to stroke correspondence search between an input character and a reference character plays an important role in stroke-order-free online Kanji character recognition. It has a vital influence on the recognition performance. Various stroke correspondence algorithms have been proposed, and official comparative studies for clarifying the relative superiority of those algorithms are hoped for. As the first stage of the studies, two typical algorithms — the cube search method (CS) and a method based on stable marriage (SM) —are experimentally compared, mainly in regard of recognition accuracy. The experimental results show that 99.10% and 97.58% recognition rates were attained by CS and SM, respectively. Additionally, we discuss the performance difference between CS and SM.

1. INTRODUCTION

A sufficient stroke order tolerability is indispensable for practical online handwriting recognition especially when multi-stroke character (Chinese character or Japanese Kanji character, for example) is dealt with. **Figure 1** shows a simple example of stroke order variation. Actually, there are characters with more than 20 strokes and their stroke order variations are very complicated.

There are several stroke order free recognition algorithms reported by different research groups [1]—[5]. Each of these algorithms, without exception, first evaluates dissimilarities between each of input strokes and each of reference strokes to form a stroke distance table. Then, a mapping or correspondence is searched for between input strokes and reference strokes on the table which give minimum stroke distances in some sense. A bijection constraint is preferably imposed on the mapping to simulate the actual stroke order variation. The problem is how to formulate the optimality criterion and how to solve the optimization problem. In the above quoted algorithms distance sum total or a like is used as the criterion and the major difference among them is the optimization algorithm for searching for the optimum mapping. These algorithms are proposed and evaluated by independent research groups, therefore, a comparative study to clarify the relative superiority is hoped for.

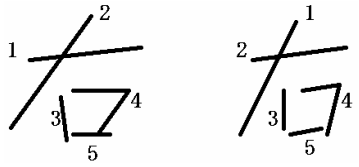

Fig. 1 Example of stroke order variation

As the first step of the comparative study, the cube search method (CS) by Sakoe and Shin [1] and the method based on stable marriage algorithm (SM) by Yokota et al. [2] are experimentally compared using a common database.

This paper is organized in the following way: In section 2 we describe the stroke to stroke correspondence search algorithms, including the basic principle and the two considered algorithms of CS and SM. In Section 3, experimental results are depicted and the performance difference between CS and SM is discussed. Finally, in Section 4 we conclude the paper.

2. ALGORITHMS STUDIED

2.1. Basic principle of stroke correspondence search

We define the input character as a stroke sequence,

$$A = A_1 A_2 \cdots A_k \cdots A_N, \qquad (1)$$

where the kth stroke A_k is the time sequence representation of local feature (moving direction and

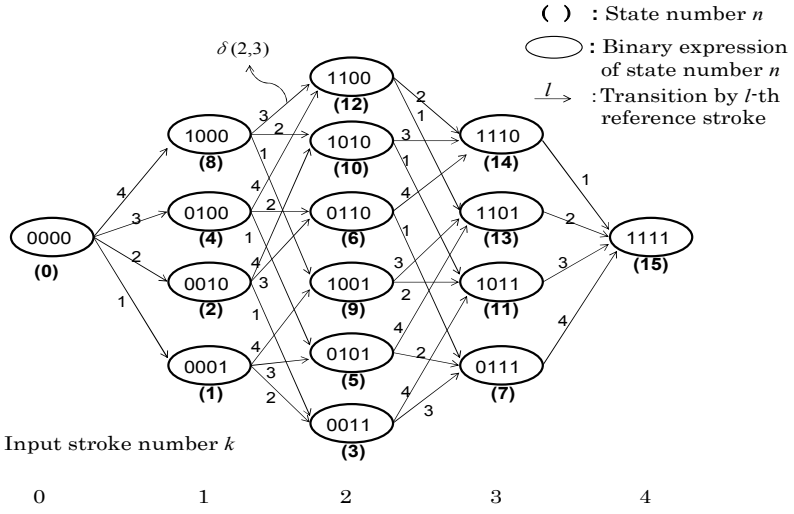

Fig. 2 Cube search graph ($N = 4$)

coordinate of pen point, for example).

$$\boldsymbol{A}_k = a_{1k}\, a_{2k} \cdot \cdot \cdot \; a_{ik} \cdot \cdot \cdot \; a_{Ik}\,, \quad I = I(k).$$

Similarly, we define the reference pattern as,

$$\boldsymbol{B} = \boldsymbol{B}_1\, \boldsymbol{B}_2 \cdot \cdot \cdot \quad \boldsymbol{B}_l \cdot \cdot \cdot \quad \boldsymbol{B}_N\,, \tag{2}$$

$$\boldsymbol{B}_l = b_{1l}\, b_{2l} \cdot \cdot \cdot \; b_{jl} \cdot \cdot \cdot \; b_{Jl}\,, \quad J = J(l).$$

We use $\delta(k, l) = D\,(\boldsymbol{A}_k, \boldsymbol{B}_l)$, which is called stroke distance and is calculated by utilizing dynamic programming (DP) matching [1], [7], to express the dissimilarity measure between stroke $\boldsymbol{A}_k$ and $\boldsymbol{B}_l$.

The distance (or dissimilarity measure) between characters $\boldsymbol{A}$ and $\boldsymbol{B}$ is the summation of $\delta(k, l(k))$, where $l(k)$ is a unique surjection (correspondence) from $\{\boldsymbol{A}_k\}$ onto $\{\boldsymbol{B}_{l(k)}\}$. If the optimal correspondence $l(k)$ is given by a stroke to stroke correspondence search, the optimal character distance is obtained.

Thus, the problem is how to get the optimal correspondence $l(k)$. In this paper, we compare two algorithms described below.

2.2. Cube search (CS) [1]

In the CS algorithm, the objective function is the summation of $\delta(k, l)$ along the correspondence $l = l(k)$. An N-dimensional cube graph, shown in **Fig. 2**, is used to impose bijection property on the mapping $l(k)$. The obtained minimum summation is used as character distance $D(\boldsymbol{A}, \boldsymbol{B})$. It is formulated as,

$$D(\boldsymbol{A}, \boldsymbol{B}) = \min_{\{l(k)\}} \left[\sum_{k=1}^{N} \delta\big(k, l(k)\big) \right]. \tag{3}$$

As shown in **Fig. 2**, in every state of N bits, each bit's position corresponds to the reference pattern stroke number l, and the bits of *1* means that these reference strokes have already been matched to some input stroke from *1* to k. The state transition from column k-*1* to column k causes a bit l to invert from '*0*' to '*1*'. Thus, the stroke correspondence search problem is equivalent to the optimal path search problem from state (*0*, *0*, · · · *0*) to state (*1*, *1*, · · · *1*) on the N-dimensional cube graph with edge-cost $\delta(k, l)$.

An efficient DP algorithm is used to search for the shortest path on the cube graph. The recurrence equation of DP is formulated as,

$$G(n) = \min_{m} \big[G(m) + \delta(k, l) \big]\,, \tag{4}$$

where G is the DP work area covering states in **Fig. 2**. m and n are state numbers standing for (*0*, *0*, · · · *0*) to (*1*, *1*, · · · *1*).

The computational complexity of CS is $\boldsymbol{O}(N \cdot 2^{N-l})$. In the practical implementation, beam search (BS) acceleration technique is used. For more details see [1].

2.3. Stable marriage (SM) [2]

Respecting everyone's preferences, the SM is an algorithm to find the most stable one-to-one matching

between two groups (input strokes and reference strokes) with equal number. As for the stroke correspondence problem, a more natural way to express the preferences is to have each stroke list in the order of value of $\delta(k, l)$. Clearly, these preferences often conflict. The sense of the stable matching is to remove unstable couples one at a time, until some stroke finds a spouse stroke which can match the stroke stably [6].

As for the kth input stroke, SM solves the stroke correspondence problem in the following manner.

Step 1 Rank the N reference strokes in the order of value of stroke distance $\delta(k, l)$, where $l = 1, \cdots, N$. If $\delta(k, l') = \min\{\delta(k, l) | l = 1, \cdots, N\}$, the reference stroke l' is selected as the candidate stroke for *Step 2*.

Step 2 If the candidate stroke l' hasn't been selected by another input stroke yet, the input stroke k will match with the reference stroke l'. Otherwise, go to *Step 3*.

Step 3 If the candidate stroke l' has been selected by another input stroke k' and $\delta(k, l') < \delta(k', l')$, the input stroke k will match with the reference stroke l'. Otherwise, if $\delta(k, l') > \delta(k', l')$, go to *Step 2* with selecting the lower order reference stroke ranked in *Step 1* as the candidate stroke and repeat the same operation as above, until the input stroke k finds a spouse stroke.

On the basis of the correspondence $l(k)$ attained by the above method, the summation of stroke distances $\delta(k, l(k))$ gives character distance $D(\boldsymbol{A}, \boldsymbol{B})$.

$$D(\boldsymbol{A}, \boldsymbol{B}) = \sum_{k=1}^{N} \delta(k, l(k)). \quad (5)$$

The computational complexity of SM is $\boldsymbol{O}(N^2)$.

3. EXPERIMENT AND DISCUSSION

In order to clarify the performance difference between CS and SM, we conducted a comparative recognition experiment on a workstation (DELL Precision WorkStation 530, 1.7GHz). A total of 18041 Kyouiku Kanji (882 classes) written by 30 persons were used as test data with no stroke connection.

The experiment was conducted in stroke number fixed mode. Which means that the reference patterns with same stroke number as the input pattern were submitted to matching. In advance of the stroke correspondence search, stoke distances were calculated by DP-matching to form a distance table. Then CS and SM were run on this common table.

Table 1 Experimental results

Algorithm	CS	SM
Recog. rate (%)	99.10	97.58
Recog. time/per character (*ms*)	410	310
Search time/per character (*ms*)	102.4	2.2

Table 2 Error rate dependency on stroke number

Stroke number N	5	10	15	20
Number of character	65	81	27	3
CS (%)	1.7	0.5	0.0	0.0
SM (%)	2.2	2.9	0.9	0.0

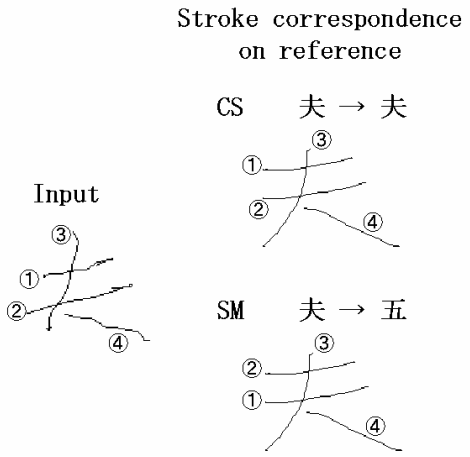

Fig. 3 A misrecognition example of SM. The stroke to stroke correspondence was given correctly by CS and was given incorrectly by SM.

Table 1 shows the experimental results including recognition rates, recognition times, and correspondence search times for each of CS and SM.

Table 2 shows error rate dependency on N, the number of strokes in input character. It is observed that in both CS and SM error reduces as N increases. This tendency might come from the fact that the information conveyed by a character pattern increases as the stroke number increases, and sufficient information is available to discriminate characters each other with some many strokes. With CS, error monotonically

reduces. With SM, on the other hand, the error rates have a peak at N =10. This might be due to the concentration of a large number of characters around there.

Consistently better performances of CS are observed from **Table 1** and **2** by which a relative superiority of CS over SM is established.

It may be said that the high accuracy of CS comes from the fact that it is based on the well defined objective function (3), and the global optimum solution is achieved by DP.

Figure 3 shows a misrecognition example of SM. As shown in **Fig. 3**, unlike the CS, the SM gave an incorrect stroke to stroke correspondence between the input character "夫" and the reference character "夫". It caused a bigger character distance between the input character and the reference character. Resultantly, the input character " 夫 " was misrecognized as the character "五".

With respect to the recognition speed, SM is much faster than CS. SM may be suitably applied to portable data terminal.

4. CONCLUSION

We discussed two kinds of one-to-one stroke correspondence search algorithms, CS and SM. Through the experiment performance superiority of CS is established. Since this paper is the first stage of the comparative studies, future studies will focus on other promising algorithms.

REFERENCES

[1] H. Sakoe, and J. Shin, "A Stroke Order Search Algorithm for Online Character Recognition," *Research Reports on Information Science and Electrical Engineering of Kyushu University*, Vol.2, No.1, pp. 99-104, 1997 (in Japanese).

[2] T. Yokota, et al., "An On-line Cuneiform Modeled Handwritten Japanese Character Recognition Method Free from Both the Number and Order of Character Strokes," *IPSJ Journal*, Vol.44, No.3, pp. 980-990, 2003 (in Japanese).

[3] K. Odaka, T. Wakahara, and I. Masuda, "Stroke Order Free On-line Handwritten Character Recognition Algorithm," *IEICE Trans. Inf. & Syst.*, Vol. J65-D, No. 6, pp. 679-686, 1982 (in Japanese).

[4] A.J. Hsieh, K.C. Fan, and T.I. Fan, "Bipartite Weighted Matching for On-line Handwritten Chinese Character Recognition," *Pattern Recognition*, Vol. 28, No. 2, pp. 143-151, 1995.

[5] C.K. Lin, K.C. Fan, and F.T.P. Lee, "On-line Recognition by Deviation-expansion Model and Dynamic Programming Matching," *Pattern Recognition*, Vol. 26, No. 2, pp. 259-268, 1993.

[6] R. Sedgewick, *Algorithms*, Addison-Wesley, second edition, pp. 499-504, 1988.

[7] H. Sakoe, and S. Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. ASSP-26, No. 1, pp. 43-49, 1978.