An Efficient Radical-Based Algorithm for Stroke-Order-Free Online Kanji Character Recognition

Wenjie Cai, Seiichi Uchida, and Hiroaki Sakoe
Graduate School of Information Science and Electrical Engineering, Kyushu University
Fukuoka 812-8581, Japan
{cwj, uchida, sakoe}@human.is.kyushu-u.ac.jp

Abstract

This paper investigates improvements of an online handwriting stroke-order analysis algorithm — cube search, based on cube graph stroke-order generation model and dynamic programming (DP). By dividing character into radicals, the model is decomposed into intra-radical graphs and an inter-radical graph. This decomposition considerably reduces the time complexity of stroke-order search DP. Experimental results showed an significant improvements in operational speed. Additionally, recognition accuracy was also improved by prohibiting unnatural stroke-order.

1. Introduction

There are a variety of reported radical-based online character recognition algorithms, e.g., [1]-[3]. In this paper, we report a novel, efficient and stroke-order-free recognition algorithm based on cube search (CS) and radical-based reference model.

CS algorithm, proposed by Sakoe and Shin [4], [5], is an effective stroke-order analysis algorithm for online character recognition. In which, an N-dimensional cube graph stroke-order generation model is defined for N-stroke character imposing bijection property on the stroke correspondence. Then, the stroke correspondence search problem is formulated as an optimal path search problem on the cube graph. An efficient dynamic programming (DP) is used to search for the shortest path on the cube graph, with $O(N \cdot 2^{N-1})$ time complexity. However, for multi-stroke character with large N (e.g., Chinese character or Japanese Kanji character), a more efficient algorithm is hoped for.

It is well known that Kanji characters are composed of limited number of radicals [1]. We redefine the character model as a set of component radicals and break down the analysis process into two levels — radical level and character level. This decomposition is based on the same principle as that of two-level connected speech recognition algorithm by Sakoe [6], [7]. By this reorganization the original cube graph is decomposed into several small scale intra-radical graphs and a small scale inter-radical graph, and allows considerable reduction in complexity. It also allows a higher recognition accuracy by omitting such unrealistic stroke transition connecting two consecutive radicals without completing the former radical.

[1]The radical does not remain in the morphological definition. Radicals in this paper may artificially be designed with the objective of efficiency or accuracy improvement.

2. Basic Cube Search Method (CS) [4], [5]

2.1. Basic Principle of Stroke Correspondence Search

We define the input character as a stroke sequence,

$$A = A_1 A_2 \dots A_k \dots A_N, \tag{1}$$

where the kth stroke A_k is the time sequence representation of local feature (e.g., moving direction and coordinate of pen-point).

$$A_k = a_{1k} a_{2k} \dots a_{ik} \dots a_{Ik}, I = I(k).$$

Similarly, we define the reference character as,

$$B = B_1 B_2 \dots B_l \dots B_N, \tag{2}$$

$$B_l = b_{1l} b_{2l} \dots b_{jl} \dots b_{Jl}, J = J(l).$$

We define stroke distance $\delta(k, l) = D(A_k, B_l)$ as the dissimilarity measure between the input stroke A_k and the reference stroke B_l, and calculate it by DP-matching [4]. The family of mappings $\{l = l(k)\}$ generates stroke-order variations. Therefore, it is a reasonable way to search for the

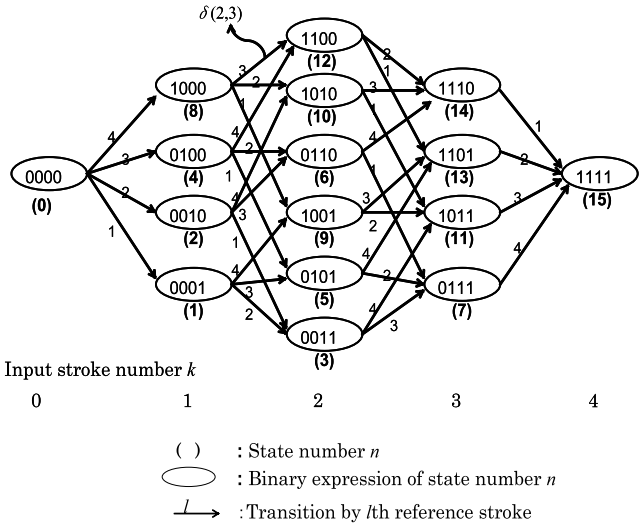


Figure 1. Cube search graph ($N = 4$).

optimal stroke-to-stroke correspondence by the following minimization problem.

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

A bijection constraint should be imposed on the mapping $l = l(k)$ to keep the property of actual stroke-order variation phenomena.

2.2. Cube Search (CS)

Figure 1 is an example of the mechanism for generating bijective stroke-order variations. It is an N-dimensional cube graph for N-stroke character. For each node, an N-bit binary number is allotted, which identifies each node, and controls the selection of node-to-node transition (edge) as flags. The lth bit of the node corresponds to the lth reference stroke and the flag value 1 means that the lth reference stroke has already been matched to some of past input stroke up to k. With input stroke transition $k-1 \rightarrow k$, an edge is selected which causes a new reference stroke match. Let the lth reference stroke be matched to the kth input stroke, for example, then the lth bit of node number is inverted as $0 \rightarrow 1$. With this combination of l and k, the stroke distance $D(A_k, B_l)$ is imposed to the edge as edge transition cost.

Thus, the calculation of Eq. (3) under bijection constraint, is optimum path search on the Fig. 1 cube graph, starting from the left-most node $(00...0)$ and terminating at the right-most node $(11...1)$. An efficient DP algorithm can be applied to this graph search. In DP terminologies, the input stroke number k stands for stage, the node stands for state, edge cost or equivalently stroke distance stands for local cost, and their sum total stands for objective function. The recurrence equation of DP is formulated as,

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

where G is the DP cumulative score, m and n are state numbers covering $(00...0)$ to $(11...1)$. In Fig. 1, the character distance $D(A,B)$ is obtained as $G(2^N-1)$ at the right-most state $(11...1)$.

The time complexity (or the number of edges) of CS is $O(N \cdot 2^{N-1})$. In practical implementation, beam search acceleration technique is used (for more details see [4], [5]). However, for characters with large N, CS algorithm is still time-consuming, even when the beam search is resorted to. In the next section, starting from CS, we derive an efficient radical-based method to solve the problem.

3. Radical-Based Cube Search

The main idea is described as following: On the basis of radicals, we redefine the reference character as,

$$B = R_1 R_2 \ldots R_p \ldots R_m, \tag{5}$$

where, R_p is the pth reference radical of B with n_p strokes, and $\sum_{p=1}^{m} n_p = N$. We consider an intra-radical cube graph, which generates intra-radical stroke-order, for each of these radicals. We further consider an inter-radical cube graph which control bijective property of radical-to-radical correspondence. In the inter-radical cube graph, an edge stands for a radical, and node flags control the radical-to-radical match. This reorganization is based upon a reasonable assumption that the stroke transition is unrealistic, which connects two consecutive radicals without completing the former radical. Considering the exponential property of cube graph complexity, considerable simplification can be expected, by this reorganization.

Figure 2 gives an example of character "桜", which is divided into three radicals "木", "ツ", and "女". In conventional CS algorithm, for 10-stroke character "桜", a cube graph with $10 \cdot 2^9 = 5120$ edges is required. When decomposition is applied, three radicals "木", "ツ", and "女" require 32, 12, and 12 edges cube graph, respectively (see Fig. 2a). The inter-radical cube graph includes 12 edges. By substituting intra-radical graphs for corresponding edges in the inter-radical graph, there exist a total of $4 \times 32 + 4 \times 12 + 4 \times 12 + 12 = 236$ edges. Thus, the computational burden for stroke-order search is considerably reduced.

By applying the two-level DP principle [6], [7], we propose the following search algorithm comprising a radical level process and a character level process.

Step 1 Radical Level Process :

Corresponding to n_p-stroke reference radical pattern, n_p-stroke input stroke sequence starting from input stroke $s+1$ is hypothesized as an input radical,

$$A(s, n_p) = A_{s+1} A_{s+2} \ldots A_{s+n_p}. \quad (6)$$

A stroke-based CS is carried out between $A(s, n_p)$ and reference radical R_p on the intra-radical cube graph to obtain the dissimilarity between $A(s, n_p)$ and R_p as radical distance $\lambda(s, p) = D(A(s, n_p), R_p)$. This is repeated for all possible combinations of s and p.

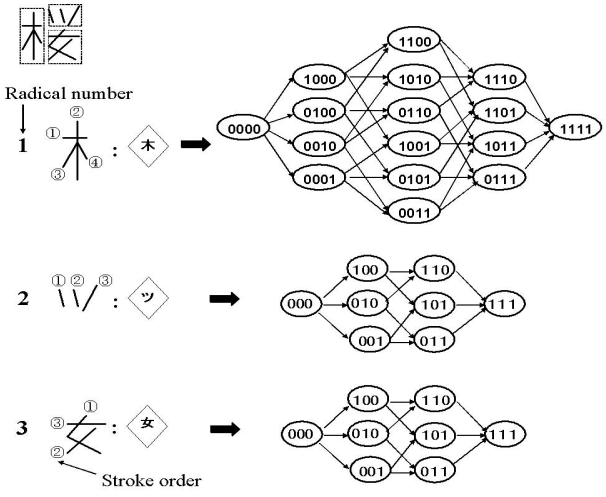
Step 2 Character Level Process :

Radical distances $\lambda(s, p)$ are summed up according to the inter-radical graph, and the optimum path is searched for which gives the minimum radical distance sum. This minimum value gives character distance,

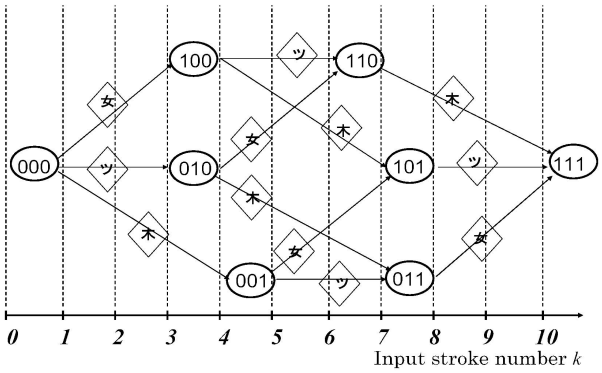
$$D(A, B) = \min_{p=p(q)} \left[\sum_{q=1}^{m} \lambda(s, p) \right]. \quad (7)$$

Constraints are (i) $p(q)$ is bijective and (ii) $\sum n_{p(q)} = N$. These constraints are equivalent to the regulation by inter-radical graph. The minimization is accomplished by DP similar to Eq. (4), except for the edge cost is the radical distance $\lambda(s, p)$.

In the implementation, some practical techniques can be applied. From Fig. 2b, we can see that we can limit the calculation of radical distances $\lambda(s, p)$ to those (s, p), where the radical R_p follows to the node at $k = s$. Consider that some considerable number of radicals are common to different characters. A technique can be implemented where radical distances of those common radicals are stored in a suitable table and accessed to when different character is processed, thus avoiding repeated calculations of the same radical distance.



(a) Intra-radical cube graphs



(b) Inter-radical cube graph

Figure 2. Radical-based cube search graph (character "桜", N=10, m=3, n_1=4, n_2=n_3=3).

4. Design Policy of Reference Character

Complexity of the proposed radical-based cube search algorithm depends on the design of radical units. A simple mathematics derives the following minimality condition for time complexity,

$$m = n_1 = n_2 = \ldots = n_m = \sqrt{N}. \quad (8)$$

Taking into account of the integer approximation of Eq. (8), the design policy will be to compromise the following requirements.

(1) Radicals in morphological meaning are bases.

(2) Number of radicals m, and the size of each radical n_p should be near by $\sqrt{N}$.

(3) When specific radical is frequently written with stroke transition to external without completing the radical, the radical is to be divided at the transition point.

5. Experimental Results and Discussions

In order to clarify the effectiveness of the proposed algorithm, we conducted experiments on a workstation (DELL Precision WorkStation 530, 1.7GHz). A total of 17983 stroke-order-free Kyouiku Kanji (882 classes, 1 — 20 strokes) written by 30 persons were used as test data with no stroke connection. As the feature information a_{ik} and b_{jl} of each point on the stroke, combination of the coordinates of the pen-point and eight directional vector approximation of the stroke direction was used. Based on the policy of

Table 1. Experimental results.

Algorithm	Stroke base	Radical base
Recog. rate(%)	99.15	99.31
Calculation time of stroke distances(ms)	39	30
Cube search time(ms)	43	10
Average recog. time(ms)	82	40

Table 2. Cube search time for an input "興" (ms).

Beam search	without	with
(a) Stroke base	215.20	5.71
(b) 2 radicals	43.15	6.95
(c) 4 radicals	0.62	0.43

Sec. 4, a set of 1009 reference radicals were designed. By these radicals reference characters were divided into 1 — 4 reference radicals.

Table 1 shows the experimental results for the conventional stroke-based algorithm and the proposed radical-based algorithm, including recognition rate, average stroke distance calculation time, average stroke correspondence analysis time, and average recognition time. The correspondence analysis speed is considerably improved, contributing to a significant improvement in recognition speed. Unignorable accuracy improvement is also attained.

In order to confirm the validity of the Eq. (8), we tested morphologically reasonable three decomposition ways of a character "興", shown in Fig. 3. This character is of 16 strokes, and the optimum decomposition is m = 4, n_1 = n_2 = n_3 = n_4 = 4. Type (c) decomposition (m = 4, n_1 = 4, n_2 = 6, n_3 = n_4 = 3) with the nearest approximation to this condition attained the highest speed. The test result is shown in Table 2.

6. Conclusions

An improved method is described for online handwriting stroke correspondence search. Stroke-based cube graph model for generating stroke-orders is decomposed into intra-radical cube graphs and an inter-radical cube graph. An efficient two-level DP algorithm is presented for searching for the optimum path on these graphs, which give the optimum stroke correspondence. By the radical decomposition, a considerable enhancement in search speed and a significant improvement in recognition accuracy are achieved.

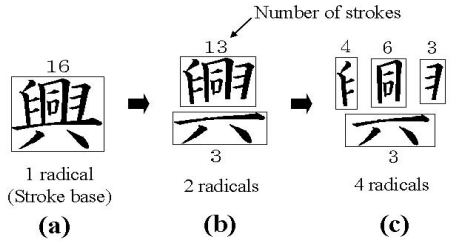


Figure 3. Three ways of radical decomposition of character "興".

References

[1] A. Kitadai, and M. Nakagawa, "Prototype Learning for Structured Character Pattern Representation Used in On-Line Recognition of Handwritten Japanese Characters," IEICE Trans. Inf. Syst., vol. J86-D-II, no. 1, pp. 1-11, Jan. 2003 (in Japanese).

[2] X. H. Xiao and R. W. Dai, "On-Line Handwritten Chinese Character Recognition Directed by Components with Dynamic Templates," Int'l J. Pattern Recognition and Artificial Intelligence, vol. 12, no.1, pp. 143-157, 1998.

[3] C. L. Liu, S. Jaeger, and M. Nakagawa, "Online Recognition of Chinese Characters: The State-of-the-Art," IEEE Trans. Pattern Anal. Mach. Intell., vol. 26, no. 2, pp.198-213, Feb. 2004.

[4] 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, March 1997 (in Japanese).

[5] J. Shin, and H. Sakoe, "Stroke Correspondence Search Method for Stroke-Order and Stroke-Number Free On-Line Character Recognition — Multilayer Cube Search — ," IEICE Trans. Inf. Syst., vol. J82-D-II, no. 2, pp. 230-239, Feb. 1999 (in Japanese).

[6] H. Sakoe, "Two-Level DP-Matching—A Dynamic Programming-Based Pattern Matching Algorithm for Connected Word Recognition," IEEE Trans. Acoust. Speech Signal Process., vol. ASSP-27, no. 6, pp. 588-595, Dec. 1979.

[7] H. Sakoe, "A Generalized Two-Level DP-Matching Algorithm for Continuous Speech Recognition," Trans. IECE Japan, vol. E65, no. 11, pp. 649-656, Nov. 1982.