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

Wenjie CAI [a], Seiichi UCHIDA [b] and Hiroaki SAKOE
[a] *O-RID Company*
*1-2-2, Minami Tateishi*
*874-0839, Beppu City, JAPAN*
[b] *Department of Advanced Information Technology, Kyushu University*
*744, Motooka, Nishi-ku*
*819-0395, Fukuoka City, JAPAN*
cwj@ksn.biglobe.ne.jp, uchida@ait.kyushu-u.ac.jp.

**Abstract.** This paper investigates improvements of an online handwriting stroke-order analysis algorithm — radical-based cube search (RBCS), which based on free stroke-order generation model called cube graph and dynamic programming (DP). We propose a novel model to resolve both of the stroke-order free and the stroke-number free problems, with lower time complexity of stroke-order search 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 and spatial complexity. Within the intra-radical graphs, we resolve the stroke-number free problem by using a model called multilayer cube graph. Experimental results showed a higher recognition accuracy of 91.65%, and a practical average recognition time of 0.45s per character.

## 1. Introduction

Handwriting recognition can be divided into two categories: online and offline. This paper copes with online multi-stroke character recognition (e.g., Chinese character or Japanese Kanji character). An important difference between online and offline handwriting recognition (or OCR) is that online devices capture the temporal or dynamic information of the writing. This information consists of the stroke-numbers, the stroke-orders, the writing direction for each stroke, and the writing speed within each stroke. A stroke is the writing from pen-down to pen-up. Most online devices capture the trace of the handwriting as a sequence of coordinate points.

The online recognition of Kanji characters is different from western handwriting recognition and poses a special challenge, because current online recognition still suffers from several weaknesses involving sensitivity to stroke-order, stroke-number, and stroke characteristics variations. Figure 1 shows an example of different writing styles. It should be noted that the two handwriting Kanji characters "王" (king) have different stroke-orders and stroke-numbers. The numerals in this figure indicate the stroke-orders. Thus, for a practical system, it needs to cope with the natural handwriting of a wide range of different stroke-order and stroke-number.

Oda et al. (Oda & al., 2007) proposed a stroke-order free and stroke-number free algorithm, combining a linear-time elastic matching based online recognizer and an offline recognizer. To the authors' best knowledge, it gives highest recognition accuracy of 92.2% on the famous Kuchibue database, and with less time complexity. However, for the stroke-order and stroke-number variation problem, basically their online recognizer depends on a few of most possible variation types preregistered in the prototype dictionary, and can not cope with all possible variations.

Sakoe and Shin (Sakoe & Shin, 1997; Shin & Sakoe, 1999) proposed a stroke-order free algorithm called cube search (CS), it 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*, a more efficient algorithm is hoped for.

Cai et al. (Cai & al., 2005; Cai & al., 2006) proposed an algorithm called radical-based cube search (RBCS) to solve the time complexity problem of CS. It redefine the character model as a set of component radicals and break down the original cube graph into several small scale intra-radical graphs and inter-radical graph, to reduce the time complexity considerably. However, since it's difficult to determine the stroke scopes of radicals when stroke-number variations occur, it can only cope with stroke-number fixed characters. On the other hand, Sakoe and Shin (Sakoe & Shin, 1997; Shin & Sakoe, 1999) proposed a multilayer CS algorithm (MLCS) to solve the stroke-number variation problem for CS at some moderate cost of increase in time complexity. In MLCS, a few of prototype strokes can be concatenated for responding to the possible input stroke concatenation.

In this paper, based on RBCS, we propose a novel stroke-order generation model to solve both of stroke-order variation and stroke-number variation problems. In intra-radical graph, we employ the MLCS to cope with stroke-order variation and stroke-number variation problems within radical. In inter-radical graph, considering the stroke-number variation of radicals, the novel graph model can effectively solve the radical-order variation problem. Experimental result shows that, it can give higher recognition accuracy and practical recognition time.

## 2. Stroke-order Free Kanji Recognition Algorithm Based on CS

### 2.1. General Problem of Determining Optimal Stroke Correspondence

Let *A* denote an input character pattern with *N* strokes,

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

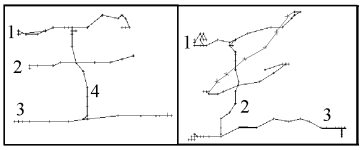



**Figure 1**: Two handwriting Kanji character "王" with different stroke-orders and stroke-numbers.

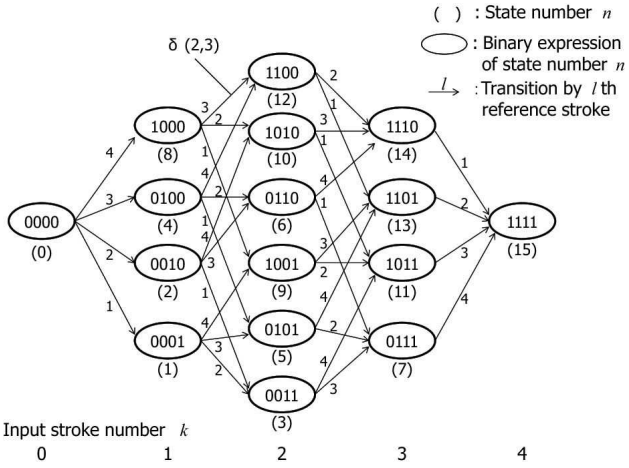



**Figure 2**: Cube search graph ($N$ = 4).

where $\boldsymbol{A}_k$ is the $k$th stroke and represented as a sequence of feature vectors. For example, $\boldsymbol{A}_k$ is a sequence of three-dimensional vectors, each of which is comprised of the local direction and $x$-$y$ coordinates. Similarly, let $B$ denote the reference character pattern,

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

Let $\delta(k, l)$ denote a stroke distance between the input stroke $\boldsymbol{A}_k$ and the reference stroke $\boldsymbol{B}_l$. DP-matching distance (Sakoe & Shin, 1997) has been often utilized for calculating a distance between a pair of strokes with different lengths.

We consider a mapping $l = l(k)$ for representing the stroke correspondence between $\boldsymbol{A}_k$ and $\boldsymbol{B}_l$. Under the mapping $l(k)$, the stroke $\boldsymbol{A}_k$ corresponds to $\boldsymbol{B}_{l(k)}$. Thus, the character distance between $\boldsymbol{A}$ and $\boldsymbol{B}$ becomes $\sum_k \delta(k, l(k))$. The mapping $l(k)$ should be bijective (one-to-one) from { $\boldsymbol{A}_k$ } onto { $\boldsymbol{B}_l$ }. The optimal stroke-order of $\boldsymbol{A}$ can be obtained as $l(1)$, …, $l(k)$, …, $l(N)$, which will minimize the criterion $\sum_k \delta(k, l(k))$.

## 2.2. Cube Search (CS)

Figure 2 is an example of the mechanism for generating bijective stroke correspondence. 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 $l$th bit of the node corresponds to the $l$th reference stroke and the flag value 1 means that the $l$th reference stroke has already been matched to some of past input stroke up to $k$. With input stroke transition $k$-1 → $k$, an edge is selected which causes a new reference stroke match. Let the $l$th reference stroke be matched to the $k$th input stroke, for example, then the $l$th bit of node number is inverted as 0 → 1. With this combination of $l$ and $k$, the stroke distance $\delta(k, l)$ is imposed to the edge as edge transition cost.

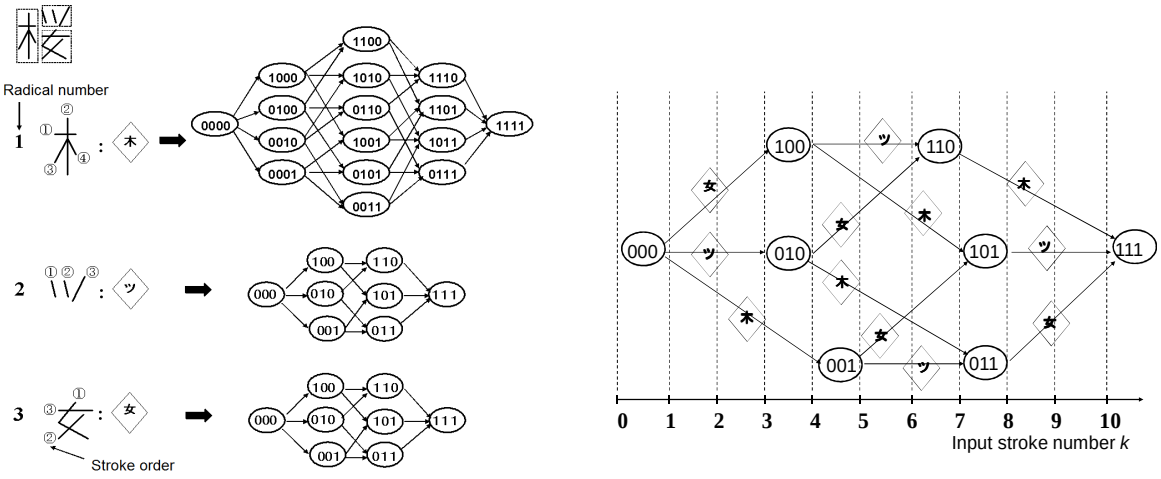
Thus, the calculation of $\sum_k \delta(k, l(k))$ under bijection constraint, is optimum path search on the Fig. 2 cube graph, starting from the left-most node (00…0) and terminating at the right-most node (11…1). An efficient DP algorithm can be applied to this graph search. In DP terminologies, the 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,

$$G(n) = \min_m \quad [G(m) + \delta(k, l)] \tag{3}$$

where $G$ is the DP cumulative score, $m$ and $n$ are state numbers covering (00…0) to (11…1). In Fig. 2, the character distance $\sum_k \delta(k, l(k))$ is obtained as $G(2^N\text{-}1)$ at the right-most state (11…1). The time complexity (or the number of edges) of CS is $O(N \cdot 2^{N-1})$. For multi-stroke character with large $N$, it is time-consuming.

## 2.3 Radical-Based Cube Search (RBCS)

It is well known that Kanji characters are composed of limited number of radicals. For RBCS, the main idea is




(a) Intra-radical cube graphs (b) Inter-radical cube graph

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

described as following: On the basis of radicals, we redefine the reference character as,

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

where, $R_p$ is the $p$th 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 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. In fact, according to our investigation on a 17,983-character online dataset (882 classes, 1-20 strokes, written by 30 persons), such stroke transitions are rare and only 26 samples are found, based on our radical definition.

Considering the exponential property of cube graph complexity, considerable simplification can be expected by above reorganization (Cai & al., 2005; Cai & al., 2006). Figure 3 gives an example of character "桜" (cherry), 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. Three radicals "木", "ツ", and "女" require 32, 12, and 12 edges cube graph, respectively (see Fig. 3a). 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 4×32+4×12+4×12+12 = 236 edges. Thus, the computational burden for stroke-order search is considerably reduced.

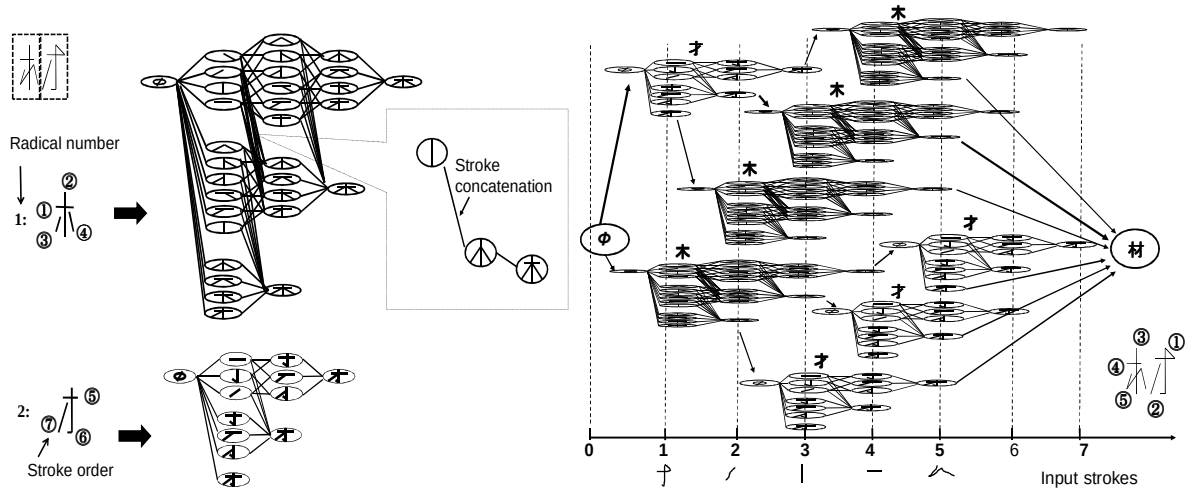
## 3. Novel Stroke-order Free and Stroke-number Free Kanji Recognition Algorithm

This Section, we propose a RBCS-based model. It is also composed of intra-radical cube graph and inter-radical cube graph, corresponding to two levels of processes — the radical level process and character level process.

### 3.1 Radical Level Process

In our proposed RBCS, MLCS is employed in the intra-radical cube graph, which copes with intra-radical stroke-number variations for each of these radicals. MLCS is an extension of the CS, to solve the stroke-number free problem at some moderate cost of increase in time complexity. Figure 4a shows examples of MLCS in radicals of character pattern "材" (material). As shown in Fig. 4a of MLCS, based on the CS graph of Fig. 2, responding to the possible stroke concatenation, two or more reference strokes can be concatenated for calculating the stroke distance cost. It should be noted that, in Fig. 4, instead of binary bits '1', we mark reference strokes themselves straightly to express the matched reference strokes of each node. The original bits of binary expression of those concatenated reference strokes are inverted from '0' to '1' in one state transition accompanied by a layer transition. The number of MLCS layers expresses the upmost number of reference strokes that can be concatenated. For example, here, upmost three reference strokes can be concatenated (two stroke concatenations).

During the radical level process, stroke correspondence between input radical and reference radical can be given. The input stroke with concatenated original strokes should be matched with a couple of relevant




(a) Intra-radical cube graphs (MLCS is employed) (b) Inter-radical cube graph

**Figure. 4** RBCS-based stroke-order free and stroke-number free cube search graph (pattern "材", $N$ = 7, $m$ = 2, $n_1$ = 4, $n_2$ = 3, two stroke concatenations).

reference strokes.

### 3.2 Character Level Process

In the inter-radical cube graph, the radical correspondence between input and reference character should be given for those input radicals, which have unsteady stroke-numbers because of possible stroke-number variations in them. As shown in Fig. 4b, responding to the possible stroke-number variations of radicals, state transitions between intra-radical cube graphs (i.e., MLCS) can start from multiple right-most state of former MLCS graphs. In addition, in order to reduce the time complexity, only promising latter MLCS graphs are selected to transit to, under the constraint of total stroke-number of input character. It should be noted that, this reorganization is also based upon the reasonable assumption that the stroke transition and stroke concatenation are unrealistic, which connects two consecutive radicals without completing the former radical.

For the inter-radical cube graph of Fig. 4b, starting from the left-most node and terminating at the right-most node, an optimal radical correspondence is searched to achieve radical-order free. Meanwhile, by applying the intra-radical MLCS, the optimal stroke correspondence between two matched radicals is given, with both of stroke-order free and stroke-number free characteristics.

## 4. Experimental Results

In order to clarify the effectiveness of the proposed algorithm, we conducted experiments on a PC with Genuine Intel(R) CPU T2300 1.66GHz processor and 1.49GB RAM. A total of 373,816 stroke-order free and stroke-number free online Kanji (882 classes, 1-20 strokes) of HANDS-kuchibue d-97-06-10 database were used as test data. A set of 1,009 reference radicals were designed for the 882 classes. By these radicals, reference characters were divided into 1-4 reference radicals. In our experiment, upmost four stroke concatenations were permitted.

Our experiment gave a higher recognition rate of 91.65%, and the average recognition time of 0.45s per character. By our experimental results, our proposed algorithm is proved to be able to provide higher recognition accuracy at practical processing speed.

## 5. Conclusion

In this paper, we proposed a novel stroke-order free and stroke-number free online Kanji character recognition algorithm. It is based on the RBCS and improved it. By employing the MLCS in the intra-radical cube graph, it solved the stroke-number variation problem within the radical. Furthermore, by efficiently determining the stroke scopes of input radicals with stroke-number variations, the proposed algorithm successfully reduced the time complexity for the time-consuming recognition algorithm of stroke-number free. The experimental results proved the proposed algorithm can give higher recognition accuracy and practical processing speed.

## References

Oda, H., et al. (2007). Size Reduction of an On-Line Handwritten Character Recognizer Combining On-Line and Off-Line Recognizers. *IEICE Trans. Inf. & Syst., vol. J90-D, no. 9,* pp. 2583-2594 (in Japanese).

Sakoe, H., & Shin, J. (1997). 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 (in Japanese).

Shin, J. & Sakoe, H. (1999). 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 (in Japanese).

Cai, W., Uchida, S. & Sakoe, H. (2005). An Efficient Stroke-Order-Free On-Line Character Recognition Algorithm Based on Radical Reference Pattern. *IEICE Trans. Inf. & Syst., vol. J88-D-II, no. 7,* pp. 1187-1195 (in Japanese).

Cai, W., Uchida, S. & Sakoe, H. (2006). An Efficient Radical-Based Algorithm for Stroke-Order-Free Online Kanji Character Recognition. *Proc. 18th Int. Conf. Pattern Recognition, vol. 2,* pp. 986-989.