# Stochastic Model of Stroke Order Variation

Yoshinori Katayama, Seiichi Uchida, and Hiroaki Sakoe
*Faculty of Information Science and Electrical Engineering, Kyushu University, 819-0395, Japan*
{*yosinori,uchida*}*@is.kyushu-u.ac.jp*

## Abstract

*A stochastic model of stroke order variation is proposed and applied to the stroke-order free on-line Kanji character recognition. The proposed model is a hidden Markov model (HMM) with a special topology to represent all stroke order variations. A sequence of state transitions from the initial state to the final state of the model represents one stroke order and provides a probability of the stroke order. The distribution of the stroke order probability can be trained automatically by using an EM algorithm from a training set of on-line character patterns. Experimental results on large-scale test patterns showed that the proposed model could represent actual stroke order variations appropriately and improve recognition accuracy by penalizing incorrect stroke orders.*

## 1. Introduction

In on-line recognition of multi-stroke characters, such as Japanese Kanji characters and Chinese characters, stroke order variation is one of the most serious hurdles. An input pattern written in a certain stroke order has a partial or global difference from its standard pattern written in a different stroke order [1] and therefore some special treatment is necessary to compensate the variation.

Several approaches have been proposed to deal with stroke order variations [1]. One important approach is to optimize the stroke-to-stroke correspondence between input and standard patterns. If the correct stroke correspondence is determined, the similarity under the correspondence will be constant regardless of the stroke order variation of an input pattern.

This approach, however, has a side effect; the similarity between an input pattern and the standard pattern of a "wrong" category is often over-estimated by unnatural stroke correspondence. For example, an input pattern "本" written in the correct stroke order will be misrecognized as "末" under an unnatural stroke correspondence which maximizes their similarity. Note that the correct stroke order of "本" is ("—" → "|' → "/" → "\" → "–") and that of "末" is ("—" → "–" → "|' → "/" → "\" ). Thus if we allow any stroke order variation, those two characters become almost identical.

One possible remedy to suppress the misrecognitions is to penalize unnatural i.e., rare stroke order on optimizing the stroke correspondence. In fact, there are *popular* stroke orders (including the standard stroke order) and there are *rare* stroke orders. If we penalize the situation that "末" is matched to an input pattern with its very rare stroke order of ("—" → "|' → "/" → "\" → "–"), we can avoid the misrecognition of "本" as "末."

For this purpose, a stochastic model of stroke order variations is proposed. The model is a hidden Markov model (HMM) with a special topology that represents all stroke order variations. A sequence of state transitions from the initial state to the final state of the model represents one stroke order. If a reasonable probability is assigned to each state transition by a training scheme, it is possible to represent the probability distribution of stroke order variations.

This model is useful for online character recognition because it can suppress rare stroke orders by a low probability. In fact, as shown in this paper, the proposed model could improve recognition accuracy of Japanese Kanji characters, which were collected from the large Japanese Kanji character database called "HANDS-kuchibue_d-97-06-10 [2]."

## 2. Conventional stroke-order free methods

### 2.1. Related works

For stroke order-free on-line character recognition, there are mainly three approaches to deal with stroke order variations. The first approach is to prepare multiple standard patterns with popular stroke orders [3]. Although this approach is very simple, there is no well-established strategy of choosing "popular" stroke orders. In addition, it cannot cope with unexpected stroke order variations at all.

The second approach [4, 5] converts on-line character patterns into image patterns and recognizes them by off-line character recognition techniques. Although this hybrid approach is also promising and provides stroke order-free recognition systems, it cannot fully utilize the dynamics, or temporal information, in on-line character patterns.

[1] Readers who are not familiar with multi-stroke characters should pay attention to the interesting fact that each multi-stroke character has its own "correct" (or standard) stroke order. Children learn the correct stroke order in their school age. Every standard pattern is often prepared in its correct stroke order. On the other hand, an incorrect but major stroke order exists as the local rule, and the rare stroke order is taken accidentally and intentionally, especially by beginners of learning multi-stroke characters.

978-0-7695-3725-2/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDAR.2009.146

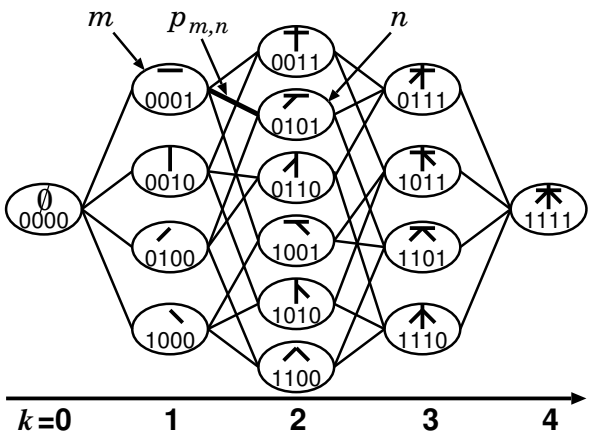


**Figure 1. Cube graph representing all stroke order variations of " 木. "**

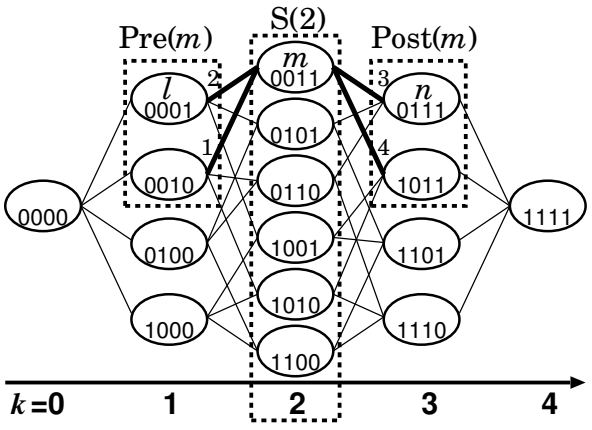


**Figure 2. Sets of states, $\mathrm{S}(k)$, $\mathrm{Pre}(m)$, and $\mathrm{Post}(m)$.**

The third and more essential approach is to optimize stroke correspondence between input and standard patterns [6, 7, 8, 9, 10]. Stroke correspondence between input and standard patterns is optimized so as to maximize total stroke similarities. Cube search [6] introduced below is also a method of this approach. In principle, this approach is robust enough to cope with any stroke order. This robustness, however, often over-estimates similarity between patterns of different categories and leads misrecognitions. In order to suppress those misrecognitions, we newly introduce a stochastic model of stroke order variations, which is a pioneering attempt to the best of authors' knowledge.

### 2.2. Cube search [6]

Cube search is a method to optimize stroke order correspondence. Its key idea is to represent all stroke order variations of a $K$-stroke character as a $K$-dimensional hyper cubic graph. The "cube" name derives from this structure. Figure 1 shows the cube graph $\mathcal{G}^c$ for the category $c =$" 木", which is a four-stroke Kanji character. All stroke order variations are represented by paths from the start node to the end node. The four-digit binary number in each node indicates the strokes matched already. Let $I_k$ be the $k$th stroke of the input pattern and $J_l^c$ be the $l$th stroke of the standard pattern of the category $c$. Then, "0101" means that $I_1$ and $I_2$ have already been matched to $J_1^c$ and $J_3^c$, or $J_3^c$ and $J_1^c$.

Each edge $e_{m,n}^c$ on the cube graph $\mathcal{G}^c$ represents a specific stroke correspondence. For example, the edge $e_{m,n}^c$ between $m = 0001$ and $n = 0101$ represents the correspondence between $I_2$ and $J_3^c$. (This is because the third bit becomes 1 at the second transition.) A stroke similarity $q_{m,n}^c$ between $I_2$ and $J_3^c$ is assigned to the edge $e_{m,n}^c$. In [6], the stroke similarity is evaluated by stroke shape and position, just like other methods [7, 8, 9, 10]. Then, the optimal stroke correspondence problem between the input pattern and the standard pattern of $c$ is considered as the optimal path problem of the cube graph. The optimal path is defined as the path with the maximum sum of stroke similarities. Dynamic programming (i.e., Viterbi algorithm) was used to find the optimal path in [6].

## 3. Cube HMM

In this section, a stochastic model of stroke order variations, called cube HMM, is newly introduced. Cube HMM is a left-to-right (and no self-state transition) HMM whose topology is defined as the cube graph. In cube HMM, each node of the cube graph is treated as a state. The stroke similarity $q_{m,n}^c$ is treated as an "symbol" output probability, where the symbol is a stroke. For example, $q_{0001,0101}^c$ is the output probability of $I_2$ and designed to become large (small) if the stroke $I_2$ is similar (dissimilar) to $J_3^c$. The probability $q_{m,n}^c$ will be detailed in 3.1.

The main difference between the cube search of 2.2 and cube HMM is that a state transition probability $p_{m,n}^c$ ($\sum_n p_{m,n}^c = 1$) is assigned to $e_{m,n}^c$, along with $q_{m,n}^c$. The state transition probability $p_{m,n}^c$ represents the probability of a specific stroke order. For example, $p_{0001,0101}^c$ represents the probability that the third standard stroke is written as the second stroke in the category $c$. Thus, if we train $p_{m,n}^c$ appropriately, cube HMM can represent the probability of stroke order variations of $c$.

The $p_{m,n}^c$ for the $K$-stroke character spends memory of $O(K \cdot 2^{K-1})$, it is hard to describe the large number of stroke character in a cube HMM. However such character consists of character components and radicals, it is useful to describe the character as cube HMMs describing radical.

### 3.1. Training algorithm

Like popular HMMs, the state transition probability $p_{m,n}$ of cube HMM can be trained by the EM algorithm. For the notational simplicity, we will drop the superscript $c$, whenever there is no confusion.

∘ Forward probability
$\alpha_0 = 1.0$;
for $k = 1$ to $K$ do
  for $\forall m \in \mathrm{S}(k)$ do
    $\alpha_m = \sum_{l \in \mathrm{Pre}(m)} \alpha_l p_{l,m} q_{l,m}$

∘ Backward probability
$\beta_M = 1.0$ where $M \in \mathrm{S}(K)$;
for $k = K - 1$ downto 0 do
  for $\forall m \in \mathrm{S}(k)$ do
    $\beta_m = \sum_{n \in \mathrm{Post}(m)} \beta_n p_{m,n} q_{m,n}$

**Figure 3. Pseudo code for calculating $\alpha_m$ and $\beta_m$.**

First, the forward probability $\alpha_m$ and the backward probability $\beta_m$ are calculated to the state $m \in \mathrm{S}(k)$, where $\mathrm{S}(k)$ is a set of states at $k$ as shown in Fig. 2. Specifically, given $p_{l,m}$ and $q_{l,m}$, $\alpha_m$ and $\beta_m$ are calculated as Fig. 3, where $\mathrm{Pre}(m)$ and $\mathrm{Post}(m)$ are the preceding and the succeeding state sets of $m$, respectively. Then, using $\alpha_m$ and $\beta_m$, the state transition probability $p_{m,n}$ is updated as

$$\hat{p}_{m,n} = \frac{\alpha_m p_{m,n} q_{m,n} \beta_n}{\sum_{n' \in \mathrm{Post}(m)} \alpha_m p_{m,n'} q_{m,n'} \beta_{n'}}. \quad (1)$$

### 3.2. Recognition using cube HMM

An unknown input pattern $I$ is classified as the category $c$ which gives the maximum likelihood by cube HMM. The likelihood is calculated as $\alpha_M$ ($M \in S(K)$) by the Viterbi algorithm, which can be derived by replacing the last line of the forward probability calculation as

$$\alpha_m = \max_{l \in \mathrm{Pre}(m)} \alpha_l p_{l,m} q_{l,m}.$$

## 4. Experimental results

### 4.1. Data sets

The public on-line Japanese Kanji character database called "HANDS-kuchibue_d-97-06-10" [2] was used in the following recognition experiments. From the database, samples with 5, 10, 15, and 20 strokes were extracted to observe the recognition performance at those stroke numbers individually. Table 1 shows statistics of the dataset. Each sample was linearly rescaled into $128 \times 128$, smoothed, and resampled.

### 4.2. Calculation of $q_{m,n}$

The stroke output probability $q_{m,n}$ was calculated by an HMM which evaluates a stroke similarity. For example,

**Table 1. Statistics of dataset. The parenthesized number shows the number of samples with regular stroke order.**

| #strokes | 5 | 10 | 15 | 20 |
|---|---|---|---|---|
| #categories | 67 | 76 | 21 | 2 |
| #samples | 28,525 (22,473) | 17,605 (10,710) | 3,336 (1,506) | 624 (155) |

$q^c_{0001,0101}$ is calculated by applying the input stroke $I_2$ to this "stroke HMM" which models the shape of the single stroke $J_3$. It should be noted that if we use an HMM for $q_{m,n}$, that is, if we *embed* the stroke HMM for every edge of the cube graph for $q_{m,n}$, cube HMM becomes a large and nested HMM.

While any HMM can be used as the stroke HMM, $(xy/\theta)$-HMM [11] was employed in the following experiment. This stroke HMM is a left-to-right HMM and comprised of $1 \sim 4$ states (except for the start and the end states) according to the number of line segments of the stroke to be modeled. For example, a "¬"- shaped stroke is modeled by an stroke HMM with 2 states.

### 4.3. Training cube HMM

The 2/3 samples of the dataset were used as training samples and the remaining 1/3 sample was used as test sample. The training samples were used not only for training $p_{m,n}$ of cube HMM but also for training the parameters of stroke HMMs.

The samples from the database show their stroke order variations. For example, among 1496 samples of "用", 1315 samples were written in the standard stroke order 1-2-3-4-5 ("ノ"→ "¬" → "–" → "–"→ "|") and 178 samples were written in the order of 1-2-5-3-4 (the fifth stroke "|" was written before the third stroke "–" and the fourth stroke "–"). The remaining three samples were written in the orders of 4-2-1-3-5, 1-3-5-2-4, and 2-3-1-4-5, respectively.

Figure 4 shows the trained cube HMM of "用." The edges with $p_{m,n} \geq 10^{-3}$ are drawn by solid lines. Among them, dominant edges with larger $p_{m,n}$ are drawn by thicker solid lines. It can be observed that stroke orders found in the database were successfully acquired in the cube HMM by the training. Note that there are "fake" edges with high probabilities. The fake edges are caused by local stroke similarities $q_{m,n}$, and Eq. (1). However the stroke order with the fake edges is negligible at the branch node by the major stroke order with actual stroke patterns.

Figure 5 is the trained cube HMM of another five stroke character, "末." This is an extreme case; the 228 samples of "末" in the database have the same standard stroke order. It can be observed that this cube HMM has only one dominant path and it represents the standard stroke order 1-2-3-4-5.

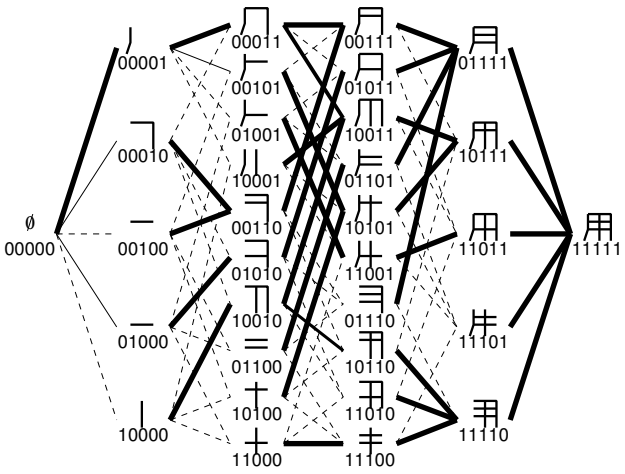

**Figure 4. Trained stroke order variation "用."**

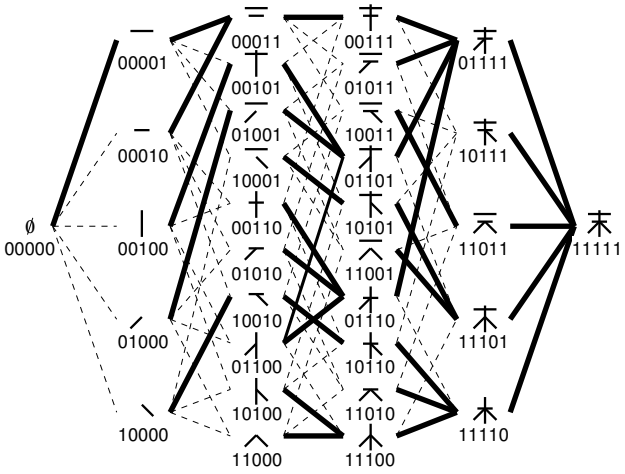

**Figure 5. Trained stroke order variation "末."**

### 4.4. Recognition results

Table 2 shows the misrecognition rates by the conventional cube search and cube HMM. The difference of these two methods is that $p_{m,n}$ is constant in the former and trained in the latter. These rates were calculated by three-hold cross validation. The "open" item was the misrecognition rate of open dataset and the "closed" item was that of closed dataset used for training. From this table, cube HMM shows better recognition performance especially for five-stroke characters. It is well known that recognition of fewer-stroke characters are generally difficult because they are affected drastically even by small variations in their shapes and, of course, their stroke orders. This fact indicates that the large improvement by cube HMM on five-stroke characters was achieved by compensating stroke order variations appropriately while avoiding unnatural stroke correspondences between different categories.

**Table 2. Error rates (%) by the proposed method and the conventional method.**

| #strokes, $K$ | | 5 | 10 | 15 | 20 |
|---|---|---|---|---|---|
| conventional | closed | 3.43 | 0.37 | 0.15 | 0.1 |
| | open | 3.80 | 0.47 | 0.39 | 0.0 |
| proposed | closed | 2.49 | 0.11 | 0.03 | 0.0 |
| | open | 2.87 | 0.56 | 0.45 | 0.0 |

**Table 3. Major improved samples for five-stroke characters.**

| #samples | input | | recognition result | | | |
|---|---|---|---|---|---|---|
| | | | conventional | | proposed | |
| | cat. | order | cat. | (#sam) order | cat. | order |
| 285 | 本 | 1-2-3-4-5 | 末 | (160) 1-3-4-2-5, (113) 1-3-4-5-2, ( 3) 2-3-4-5-1 | 本 | 1-2-3-4-5 |
| | | | | ( 9) 1-3-4-2-5 | | 1-2-3-5-4 |
| 154 | 本 | 1-2-3-4-5 | 末 | ( 84) 1-3-4-2-5, ( 58) 1-3-4-5-2, ( 7) 2-3-4-5-1 | 本 | 1-2-3-4-5 |
| | | | | ( 5) 1-3-4-2-5 | | 1-2-3-5-4 |
| 41 | 田 | 1-2-3-4-5 | 旧 | ( 21) 1-3-2-4-5 | 田 | 1-2-3-4-5 |
| | | 1-2-4-3-5 | | ( 19) 1-3-4-2-5 | | 1-2-4-3-5 |
| | | 1-2-4-5-3 | | ( 1) 1-3-4-5-2 | | 1-2-4-5-3 |

**Table 4. Major degraded samples for five-stroke characters.**

| #samples | input | | recognition result | | | |
|---|---|---|---|---|---|---|
| | | | conventional | | proposed | |
| | cat. | order | cat. | (#sam) order | cat. | order |
| 7 | 用 | 1-2-5-3-4 | 用 | (6)1-2-5-3-4 | 田 | 1-2-3-4-5 |
| | | 1-3-5-2-4 | | (1)1-3-5-2-4 | | 1-2-3-5-4 |
| 6 | 民 | 1-2-3-5-4 | 民 | (6)1-2-3-5-4 | 末 | 1-2-3-4-5 |
| 5 | 世 | 1-3-4-2-5 | 世 | (5)1-3-4-2-5 | 北 | 1-2-3-4-5 |

Table 3 shows the major improved samples on five-stroke characters. For example, 285 samples of input pattern "本" written in its standard stroke order (1-2-3-4-5) were misrecognized as "末" by the cube search. This means that the cube search allow "末" to have the stroke orders of 1-3-4-2-5 etc. In other words, cube HMM did not allow "末" to have those stroke orders. This is because the cube HMM of "末" only allows the standard stroke order as we saw in Fig. 5 and the over-fitting of "末" to "本" was prohibited.

Table 4 shows the major degraded samples for five-stroke characters. This table shows that input patterns with rare stroke orders were misrecognized by cube HMM. Clearly, this is the side effect of cube HMM, which suppresses rare stroke orders by training popular stroke orders. This side

**Table 5. Error rate (%) by cube HMM after additional training.**

| #strokes, $K$ | 5 | 10 | 15 | 20 |
|---|---|---|---|---|
| closed | 1.38 | 0.03 | 0.03 | 0.0 |
| open | 1.80 | 0.44 | 0.51 | 0.0 |

effect is not serious since characters with rare stroke order variations are rare.

### 4.5. Additional training

As noted before, cube HMM can be considered as a large and nested HMM when stroke HMMs are embedded for evaluating $q_{m,n}$. This large HMM is still a single HMM and its all the parameters can be trained simultaneously by the EM algorithm, although in the above experiment the stroke HMM was independently trained before the training of $p_{m,n}$.

The simultaneous training has a merit that we can train the stroke HMMs without care of writing order of training samples. If we train the stroke HMM of the third stroke ("–") of "用" independently, we must collect this stroke from training samples by manually checking their stroke orders. (This is because the target stroke ("–") might be the fourth stroke under a stroke order variation.) This merit also allows us to do additional training, where the stroke HMM is firstly trained independently and then trained simultaneously with cube HMM, i.e., $p_{m,n}$.

Table 5 shows the error rates by the cube HMM with additional training. Comparing with Table 2, it is clear that additional training could improve recognition performance especially for five-stroke and ten-stroke characters.

As for scriptor adaptation of the cube HMM, additional training using the personal on-line data is the method apt. It is effective for the cube HMM tuned by on-line database, because the cube HMM has all stroke order variations.

## 5. Conclusion

A new HMM, called cube HMM, has been proposed to model stroke order variations in a stochastic manner. Cube HMM was a left-to-right HMM with the topology of a cube graph. Each path from the start node to the end node of the cube graph represents one possible stroke order. Thus, if a probability is assigned to each edge of the graph, it is possible to control the probability of a certain stroke order. This probability can be trained automatically by using the EM algorithm like popular HMMs.

The results of training cube HMM by a large-scale on-line Kanji character database showed that cube HMM could acquire the probability distribution of the stroke orders of each category. Cube HMM was also applied to stroke-order free on-line character recognition. The experimental result showed the superiority of the cube HMM over the conventional cube search especially on fewer-stroke characters, which were easily misrecognized by insufficient or unnatural compensation of their stroke order variations.

**Acknowledgment**: This work is partially supported by Microsoft Research Asia Mobile Computing in Education Theme Program.

## References

[1] C. L. Liu, S. Jaeger, and M. Nakagawa, "Online recognition of Chinese characters: the state-of-the-art," *IEEE Trans. PAMI*, vol. 26, no. 2, pp. 198–213, 2004.

[2] M. Nakagawa and K. Matsumoto, "Collection of on-line handwritten Japanese character pattern databases and their analyses," *IJDAR*, vol. 7, no. 1, pp. 69–81, 2004.

[3] M. Kobayashi *et al.*, "RAV (reparameterized angle variations) algorithm for online handwriting recognition," *Proc. IJDAR*, vol. 3, pp. 181–191, Mar. 2001.

[4] M. Hamanaka, K. Yamada, and J. Tsukumo, "On-line Japanese character recognition based on flexible pattern matching method using normalization-cooperative feature extraction," *IEICE Trans. Inf. & Syst.*, vol. E77-D, pp. 825–831, Jul. 1994.

[5] M. Okamoto and K. Yamamoto, "On-line handwriting character recognition using direction-change features that consider imaginary strokes," *Pattern Recognit.*, vol. 32, pp. 1115–1128, Jul. 1999.

[6] J. P. Shin and H. Sakoe, "Stroke correspondence search method for stroke-order and stroke-number free on-line character recognition — multilayer cube search —," *IEICE Trans.*, vol. J82-D-II, no. 2, pp. 230–239, 1999 (in Japanese).

[7] K. Odaka, T. Wakahara, and I. Masuda, "Stroke-order-independent on-line character recognition algorithm and its application," *Rev. Elec. Commun. Lab.*, vol. 34, pp. 79–85, 1986.

[8] C. K. Lin, K. C. Fan, and F. T. P. Lee, "On-line recognition by deviation-expansion model and dynamic programming matching," *Pattern Recognit.*, vol. 26, no. 2, pp. 259–268, 1993.

[9] A. J. Hsieh, K. C. Fan, and T. I. Fan, "Bipartite weighted matching for on-line handwritten Chinese character recognition," *Pattern Recognit.*, vol. 28, no. 2, pp. 143–151, 1995.

[10] T. Yokota *et al.*, "An on-line cuneiform modeled handwritten Japanese character recognition method free from both the number and order of character strokes," *Trans. IPSJ*, vol. 44, pp. 980–990, Mar. 2003 (in Japanese).

[11] Y. Katayama, S. Uchida, and H. Sakoe, "A new HMM for on-line character recognition using pen-direction and pen-coordinate features," *Proc. ICPR*, Dec. 2008.